PY32E407 系列

32 位 ARM® Cortex®-M4F 微控制器

参考手册

Puya Confidential

Puya Semiconductor (Shanghai) Co., Ltd.

目录

1. 寄存器描述中使用的缩写列表 52

2. 系统架构框图 53

3. 系统及存储器架构 54

3.1 ARM® CORTEX®-M4F 处理器简介 54

3.2 系统架构 54

3.2.1 I_bus 55

3.2.2 D_bus 55

3.2.3 S_bus 55

3.2.4 DMA1/DMA2 BUS 55

3.2.5 ETH DMA BUS 55

3.2.6 总线矩阵 56

3.2.7 AHB/APB 总线桥 56

3.3 存储器组织架构 56

3.3.1 简介 56

3.3.2 存储器映射 57

3.4 嵌入式 SRAM 60

3.4.1 奇偶校验 61

3.4.2 CCM SRAM 写保护 61

3.4.3 CCM SRAM 读保护 62

3.4.4 CCM SRAM 擦除 62

3.5 嵌入式 FLASH 62

3.6 位段 63

3.7 启动配置 63

3.7.1 系统存储启动程序 64

3.7.2 物理重映射 64

4. 嵌入式 FLASH 接口 65

4.1 简介 65

4.2 主要特征 65

4.3 闪存结构 65

4.4 错误代码纠正（ECC） 66

4.5 读访问延迟 66

4.6 自适应实时存储器加速器 67

4.7 擦除和编程操作 67
4.7.1 闪存解锁 68
4.7.2 闪存擦除操作 68
4.7.2.1 闪存页擦除 68
4.7.2.2 闪存扇区擦除 68
4.7.2.3 闪存块擦除 69
4.7.2.4 闪存 Bank 擦除 69
4.7.3 闪存写操作 69
4.8 FLASH 选项字节 70
4.8.1 Flash 选项字节描述 70
4.8.1.1 Flash 用户选项 1 的选项字节 71
4.8.1.2 Flash 用户选项 2 的选项字节 72
4.8.1.3 Flash WRP 地址的选项字节 73
4.8.1.4 Flash PCROP0SR 地址的选项字节 73
4.8.1.5 Flash PCROP0ER 地址的选项字节 73
4.8.1.6 Flash PCROP1SR 地址的选项字节 74
4.8.1.7 Flash PCROP1ER 地址的选项字节 74
4.8.2 Flash 选项字节写 74
4.8.2.1 修改选项字节 75
4.8.2.2 加载选址字节 75
4.9 闪存用户 OTP 字节 76
4.10 闪存保护 76
4.10.1 闪存读保护(RDP) 77
4.10.1.1 Level 0：no protection 77
4.10.1.2 Level 1：Read protection 77
4.10.2 闪存写保护（WRP) 78
4.10.3 专有代码读出保护(PCROP) 78
4.10.4 强制从 Main flash 启动 78
4.11 闪存中断 79
4.12 寄存器描述 79
4.12.1 Flash 访问控制寄存器(FLASH_ACR) 79
4.12.2 Flash 密钥寄存器(FLASH_KEYR) 80
4.12.3 Flash 选项密钥寄存器（FLASH_OPTKEYR) 80
4.12.4 Flash 状态寄存器(FLASH_SR) 80
4.12.5 Flash 控制寄存器(FLASH_CR) 81

4.12.6 Flash ECC 寄存器（FLASH_ECCR) *83*
4.12.7 Flash 选项 1 寄存器(FLASH_OPTR1) *84*
4.12.8 Flash 选项 2 寄存器(FLASH_OPTR2) *85*
4.12.9 Flash WRP 地址寄存器(FLASH_WRPR) *85*
4.12.10 Flash PCROP0 起始地址寄存器（FLASH_PCROP0SR) *86*
4.12.11 Flash PCROP0 结束地址寄存器（FLASH_PCROP0ER) *86*
4.12.12 Flash PCROP1 起始地址寄存器（FLASH_PCROP1SR) *87*
4.12.13 Flash PCROP1 结束地址寄存器（FLASH_PCROP1ER) *87*

5. 电源控制（PWR） **88**

5.1 简介 88
5.2 电源结构 88
5.2.1 系统主要电源 *89*
5.2.1.1 独立 A/D 转换器电源和参考电压 89
5.2.1.2 备份电池域 89
5.2.1.3 VDDD 调压器 90
5.2.2 电源检测 *90*
5.2.2.1 POR/PDR 90
5.2.2.2 PVD 91
5.3 系统低功耗模式 91
5.3.1 正常运行模式 *94*
5.3.2 低功耗运行模式 *94*
5.3.3 低功耗模式进入和退出 *95*
5.3.4 睡眠模式 *96*
5.3.5 低功耗睡眠模式 *96*
5.3.6 停止模式 *96*
5.3.7 待机模式 *98*
5.3.8 停止模式和待机模式的调试 *99*
5.3.9 低功耗模式下的自动唤醒（AWU） *99*
5.4 寄存器描述 99
5.4.1 电源控制寄存器 1(PWR_CR1)(0x00) *99*
5.4.2 电源控制寄存器 2(PWR_CR2)(0x04) *100*
5.4.3 电源控制寄存器 3(PWR_CR3)(0x08) *101*
5.4.4 电源控制寄存器 4(PWR_CR4)(0x0C) *102*
5.4.5 电源状态寄存器(PWR_SR)(0x10) *103*
5.4.6 电源状态清除寄存器(PWR_SCR)(0x14) *104*

6. 复位与时钟控制（RCC） 105

6.1 简介 105

6.1.1 主要特征 105

6.2 系统复位 105

6.2.1 NRST 引脚（外部复位） 105

6.2.2 软件复位 105

6.2.3 低功耗管理复位 106

6.2.4 选项字节加载复位 106

6.3 电源复位 106

6.4 备份域复位 106

6.5 备份域复位之外的复位的统一处理 106

6.6 时钟结构 107

6.7 时钟源 108

6.7.1 HSE 时钟 108

6.7.2 HSI16 时钟 108

6.7.3 HSI48 时钟 108

6.7.4 PLL 时钟 108

6.7.5 LSE 时钟 108

6.7.6 LSI 时钟 109

6.7.7 HSI10M 时钟 109

6.8 时钟安全系统（CSS） 109

6.9 外设时钟使能寄存器 109

6.10 寄存器描述 109

6.10.1 时钟控制寄存器（RCC_CR） 109

6.10.2 时钟配置寄存器（RCC_CFGR） 111

6.10.3 时钟中断寄存器（RCC_CIR） 113

6.10.4 AHB1 外设复位寄存器（RCC_AHB1RSTR） 115

6.10.5 AHB2 外设复位寄存器（RCC_AHB2RSTR） 116

6.10.6 APB1 外设复位寄存器 1（RCC_APB1RSTR1） 117

6.10.7 APB1 外设复位寄存器 2（RCC_APB1RSTR2） 119

6.10.8 APB2 外设复位寄存器（RCC_APB2RSTR） 120

6.10.9 AHB1 外设时钟使能寄存器（RCC_AHB1ENR） 121

6.10.10 AHB2 外设时钟使能寄存器（RCC_AHB2ENR） 122

6.10.11 APB1 外设时钟使能寄存器 1（RCC_APB1ENR1） 123

6.10.12 APB1 外设时钟使能寄存器 2（RCC_APB1ENR2） 125

6.10.13 APB2 外设时钟使能寄存器（RCC_APB2ENR） 126
6.10.14 外设独立时钟配置寄存器（RCC_CCIPR） 127
6.10.15 RTC domain 控制寄存器（RCC_BDCR） 129
6.10.16 控制/状态寄存器（RCC_CSR） 130
6.10.17 时钟配置寄存器 1（RCC_CFGR1） 131
6.10.18 时钟配置寄存器 2（RCC_CFGR2） 132
6.10.19 时钟配置寄存器 3（RCC_CFGR3） 133
7. 备份寄存器（BKP） 135
7.1 简介 135
7.2 主要特性 135
7.3 功能描述 135
7.3.1 侵入检测 135
7.3.2 RTC 校准 136
7.3.2.1 概述 136
7.3.2.2 RTC 校准方式 136
7.3.2.3 计算所需的校准量 138
7.3.3 备份寄存器的擦除 138
7.4 寄存器描述 139
7.4.1 备份控制寄存器（BKP_CR） 139
7.4.2 备份控制/状态寄存器（BKP_CSR） 139
7.4.3 RTC 时钟校准寄存器（BKP_RTCCR） 140
7.4.4 备份数据寄存器（BKP_DRx）（x=0…31） 141
8. CRC 计算单元（CRC） 142
8.1 简介 142
8.2 CRC 主要特性 142
8.3 CRC 功能描述 142
8.3.1 CRC 结构框图 142
8.3.2 CRC 操作 143
8.3.3 多项式可编程性 143
8.4 寄存器描述 144
8.4.1 数据寄存器（CRC_DR） 144
8.4.2 独立数据寄存器（ CRC_IDR） 144
8.4.3 控制寄存器（CRC_CR） 144
8.4.4 CRC 初始值寄存器(CRC_INIT) 145

8.4.5 CRC 多项式(CRC_POL)..... 145

9. 通用 IO（GPIO）..... 146

9.1 简介..... 146

9.2 主要特征..... 146

9.3 功能说明..... 146

9.3.1 通用IO（GPIO）..... 147

9.3.2 IO 引脚复用器和映射..... 148

9.3.3 IO 端口控制寄存器..... 148

9.3.4 IO 端口数据寄存器..... 148

9.3.5 IO 数据位操作..... 149

9.3.6 GPIO 锁定机制..... 149

9.3.7 IO 复用功能输入/输出..... 149

9.3.8 外部中断线/唤醒线..... 149

9.3.9 输入配置..... 149

9.3.10 输出配置..... 150

9.3.11 复用功能配置..... 151

9.3.12 模拟配置..... 152

9.3.13 使用HSE 或LSE 振荡器引脚作为GPIO..... 153

9.3.14 使用RTC 电源域中的GPIO 引脚..... 153

9.3.15 使用PB8 作为GPIO..... 153

9.3.16 使用PF5 作为GPIO..... 153

9.4 寄存器描述..... 154

9.4.1 GPIO 端口模式寄存器（GPIOx_MODER）(x= A..F)..... 154

9.4.2 GPIO 端口输出类型寄存器（GPIOx_OTYPER）(x= A..F)..... 154

9.4.3 GPIO 端口输出速度寄存器（GPIOx_OSPEEDR）(x= A..F)..... 155

9.4.4 GPIO 端口上拉/下拉寄存器（GPIOx_PUPDR）(x= A..F)..... 155

9.4.5 GPIO 端口输入数据寄存器（GPIOx_IDR）(x= A..F)..... 155

9.4.6 GPIO 端口输出数据寄存器（GPIOx_ODR）(x= A..F)..... 156

9.4.7 GPIO 端口置位/复位寄存器（GPIOx_BSRR）(x= A..F)..... 156

9.4.8 GPIO 端口配置锁定寄存器（GPIOx_LCKR）(x= A..F)..... 157

9.4.9 GPIO 复用功能低寄存器（GPIOx_AFRL）(x= A..F)..... 157

9.4.10 GPIO 复用功能高寄存器（GPIOx_AFRH）(x= A..F)..... 158

9.4.11 GPIO 端口位复位寄存器（GPIOx_BRR）(x= A..F)..... 158

10. 系统配置控制器（SYSCFG）..... 159

10.1 主要特性 159
10.2 寄存器描述 159
10.2.1 SYSCFG 配置寄存器 1（SYSCFG_CFGR1） 159
10.2.2 SYSCFG 配置寄存器 2 （SYSCFG_CFGR2) 161
10.2.3 SYSCFG 配置寄存器 3（SYSCFG_CFGR3） 162
10.2.4 SYSCFG 配置寄存器 4（SYSCFG_CFGR4） 162
10.2.5 SYSCFG 配置寄存器 5（SYSCFG_CFGR5） 163
10.2.6 外部中断配置寄存器 1（SYS_EXTICR1） 163
10.2.7 外部中断配置寄存器 2（SYS_EXTICR2） 164
10.2.8 外部中断配置寄存器 3（SYS_EXTICR3） 164
10.2.9 外部中断配置寄存器 4（SYS_EXTICR4） 165
10.2.10 GPIOA 滤波使能寄存器（PA_ENS） 165
10.2.11 GPIOB 滤波使能寄存器（PB_ENS） 165
10.2.12 GPIOC 滤波使能寄存器（PC_ENS） 166
10.2.13 GPIOD 滤波使能寄存器（PD_ENS） 166
10.2.14 GPIOE 滤波使能寄存器（PE_ENS） 166
10.2.15 GPIOF 滤波使能寄存器（PF_ENS） 167
10.2.16 GPIOA 模拟通道 2 使能寄存器(PA_ANA2ENR) 167
10.2.17 GPIOB 模拟通道 2 使能寄存器(PB_ANA2ENR) 168
10.2.18 GPIOC 模拟通道 2 使能寄存器(PC_ANA2ENR) 168
10.2.19 GPIOD 模拟通道 2 使能寄存器(PD_ANA2ENR) 169
10.2.20 GPIOE 模拟通道 2 使能寄存器(PE_ANA2ENR) 170
10.2.21 PVD_IN 模拟通道 2 使能寄存器(PVDIN_ANA2ENR) 170
10.2.22 SYSCFG CCM SRAM 控制与状态寄存器(SYSCFG_SCSR) 170
10.2.23 SYSCFG CCM SRAM 写保护寄存器（SYSCFG_SWPR) 171
10.2.24 SYSCFG CCM SRAM 键值寄存器（SYSCFG_SKR) 171
11. 外设互联矩阵 172
11.1 简介 172
11.2 互联详情 173
11.2.1 Input trigger(itr) 173
11.2.2 External trigger(etr) 173
11.2.3 Ocref clear 174
11.2.4 TI1 174
11.2.5 TI2 174
11.2.6 TI3 174

11.2.7 TI4 175
11.2.8 break1 175
11.2.9 break2 175
11.2.10 比较器消隐源 175
11.2.11 IRTIM control signal 175
11.2.12 System error 176
12. DMA 控制器（DMA） 177
12.1 概述 177
12.1.1 主要特性 177
12.1.2 基本定义 177
12.1.3 顶层结构图 179
12.2 功能描述 179
12.2.1 接口定义 179
12.2.2 存储器外设 180
12.2.3 DMA 传输 180
12.2.4 仲裁器 180
12.2.5 DMA 通道 181
12.2.5.1 传输数据量可编程 181
12.2.5.2 地址指针增量 181
12.2.5.3 存储器到存储器模式 181
12.2.6 错误处理 181
12.2.7 中断 182
12.2.8 DMA 外设请求映射 182
12.3 寄存器描述 183
12.3.1 DMA 模块使能寄存器（DMA_EN） 183
12.3.2 DMA 通道使能寄存器（DMA_CH_EN） 183
12.3.3 DMA 控制寄存器（DMA_CH_CTRLLx） 184
12.3.4 DMA 控制寄存器（DMA_CH_CTRLHx） 185
12.3.5 DMA 配置寄存器（DMA_CH_CFGLx） 186
12.3.6 DMA 配置寄存器（DMA_CH_CFGHx） 187
12.3.7 DMA 通道 x 源地址寄存器（DMA_SAx） 188
12.3.8 DMA 通道 x 目标地址寄存器（DMA_DAx） 188
12.3.9 DMA 半块传输中断寄存器（DMA_STATUSHALFBLOCK） 188
12.3.10 DMA 半块传输中断屏蔽寄存器（DMA_MASKHALFBLOCK） 189
12.3.11 DMA 半块传输完成中断清除寄存器（DMA_CLEARHALFBLOCK） 189

12.3.12 DMA 传输完成中断寄存器（DMA_STATUSTFR）.....189
12.3.13 DMA 块传输完成中断寄存器（DMA_STATUSBLOCK）.....190
12.3.14 DMA 源事务完成中断寄存器（DMA_STATUSSRCTRAN）.....190
12.3.15 DMA 目标事务完成中断寄存器（DMA_STATUSDSTTRAN）.....191
12.3.16 DMA 传输错误中断寄存器（DMA_STATUSERR）.....191
12.3.17 DMA 传输完成中断屏蔽寄存器（DMA_MASKTFR）.....191
12.3.18 DMA 块传输完成中断屏蔽寄存器（DMA_MASKBLOCK）.....192
12.3.19 DMA 源事务完成中断屏蔽寄存器（DMA_MASKSRCTRAN）.....192
12.3.20 DMA 目标事务完成中断屏蔽寄存器（DMA_MASKDSTTRAN）.....193
12.3.21 DMA 传输错误中断屏蔽寄存器（DMA_MASKERR）.....194
12.3.22 DMA 传输完成中断清除寄存器（DMA_CLEARTFR）.....194
12.3.23 DMA 块传输完成中断清除寄存器（DMA_CLEARBLOCK）.....195
12.3.24 DMA 源事务完成中断清除寄存器（DMA_CLEARSRCTRAN）.....195
12.3.25 DMA 目标事务中断清除寄存器（DMA_CLEARDSTTRAN）.....195
12.3.26 DMA 传输错误中断清除寄存器（DMA_CLEARERR）.....196
12.3.27 DMA 通道中断状态寄存器（DMA_STATUSINT）.....196
12.3.28 DMA 源事务软件握手寄存器（DMA_REQSRC）.....197
12.3.29 DMA 目标事务软件握手寄存器（DMA_REQDST）.....197
12.3.30 DMA 源单事务软件握手寄存器（DMA_SGLRQSRC）.....198
12.3.31 DMA 目标单事务软件握手寄存器（DMA_SGLRQDST）.....198
13. 中断和事件.....200
13.1 嵌套向量中断控制器(NVIC).....200
13.1.1 主要特性.....200
13.1.2 系统滴答（SysTick）校准值寄存器.....200
13.1.3 中断和异常向量表.....200
13.2 外部扩展中断/事件控制器（EXTI）.....203
13.2.1 主要特性.....203
13.2.2 唤醒事件管理.....203
13.2.3 功能说明.....203
13.2.3.1 EXTI 框图.....204
13.2.3.2 硬件中断选择.....204
13.2.3.3 硬件事件选择.....204
13.2.3.4 软件中断/事件的选择.....204
13.2.4 外部中断/事件线映射.....204
13.3 寄存器描述.....206

13.3.1 中断屏蔽寄存器（0x00：EXTI_IMR1）.....206
13.3.2 事件屏蔽寄存器（0x04：EXTI_EMR1）.....206
13.3.3 上升沿触发选择寄存器（0x08：EXTI_RTSR）.....207
13.3.4 下降沿触发选择寄存器（0x0C：EXTI_FTSR）.....207
13.3.5 软件中断事件寄存器（0x10：EXTI_SWIER）.....207
13.3.6 挂起寄存器（0x14：EXTI_PR）.....208
13.3.7 中断屏蔽寄存器（0x18：EXTI_IMR2）.....208
13.3.8 事件屏蔽寄存器（0x1C：EXTI_EMR2）.....209
14. 模拟/数字转换（ADC）.....210
14.1 简介.....210
14.2 主要特征.....210
14.3 功能说明.....211
14.3.1 ADC 模块框图.....211
14.3.2 ADC1/2/3 连接关系.....212
14.3.3 ADC1/2/3 使用限制.....214
14.3.4 单端和差分输入通道.....215
14.3.5 ADC 校准.....215
14.3.5.1 ADC 软件校准.....215
14.3.6 ADC 开-关控制.....216
14.3.7 写 ADC 控制位的限制.....216
14.3.8 ADC 时钟.....217
14.3.9 ADC 通道选择.....217
14.3.10 强制停止 ADC.....217
14.3.11 动态低功耗特性.....218
14.3.11.1 自动延迟转换模式(AUTDLY).....218
14.3.12 转换模式.....218
14.3.12.1 单次转换模式.....218
14.3.12.2 连续转换模式.....219
14.3.12.3 扫描模式.....219
14.3.12.4 间断转换模式.....220
14.3.13 注入通道管理.....221
14.3.13.1 触发注入.....221
14.3.13.2 自动注入.....221
14.3.14 模拟看门狗.....222
14.3.14.1 AWDx 标志和中断.....222

14.3.14.2 模拟看门狗 1 描述 222
14.3.14.3 看门狗 1 的模拟看门狗滤波器 223
14.3.14.4 看门狗 2 和 3 的描述 223
14.3.14.5 ADCy_AWDx_OUT 信号输出产生 224
14.3.14.6 带增益和偏移补偿的模拟看门狗 224
14.3.15 数据处理 224
14.3.15.1 数据对齐 224
14.3.15.2 offset 225
14.3.15.3 增益补偿 227
14.3.15.4 Offset 补偿 227
14.3.16 过采样器 227
14.3.16.1 过采样时不支持的 ADC 操作模式（单个 ADC 模式和双 ADC 模式） 229
14.3.16.2 过采样时支持的 ADC 操作模式（单个 ADC 模式） 229
14.3.16.3 模拟看门狗 229
14.3.16.4 触发模式 229
14.3.16.5 仅对规则通道进行过采样 230
14.3.16.6 仅对注入通道进行过采样 230
14.3.16.7 自动注入模式 231
14.3.16.8 过采样时支持双 ADC 模式 231
14.3.16.9 模式组合小结 231
14.3.17 可编程采样时间 231
14.3.18 外部触发转换 232
14.3.19 可配置分辨率 233
14.3.20 DMA 请求 233
14.3.21 双 ADC 模式 233
14.3.21.1 同步注入模式 234
14.3.21.2 同步规则模式 235
14.3.21.3 快速交叉模式 235
14.3.21.4 慢速交叉模式 236
14.3.21.5 交替触发模式 237
14.3.21.6 独立模式 238
14.3.21.7 混合的规则/注入同步模式 238
14.3.22 温度传感器和内部参考电压 239
14.3.23 电池监视 239
14.3.24 ADC 中断 240

14.4 寄存器描述（对于每个 ADC) 240
14.4.1 ADC 中断和状态寄存器（0x00:ADC_ISR) *240*
14.4.2 ADC 中断使能寄存器（0x04:ADC_IER) *242*
14.4.3 ADC 控制寄存器（0x08:ADC_CR) *243*
14.4.4 ADC 配置寄存器（0x0C:ADC_CFGR) *244*
14.4.5 ADC 配置寄存器 2（0x10:ADC_CFGR2) *247*
14.4.6 ADC 采样时间寄存器 1(0x14:ADC_SMPR1) *249*
14.4.7 ADC 采样时间寄存器 2(0x18:ADC_SMPR2) *249*
14.4.8 ADC 看门狗阈值寄存器 1（0x20:ADC_TR1) *250*
14.4.9 ADC 看门狗阈值寄存器 2（0x24:ADC_TR2) *250*
14.4.10 ADC 看门狗阈值寄存器 3（0x28:ADC_TR3) *251*
14.4.11 ADC 规则序列寄存器 1(0x30:ADC_SQR1) *251*
14.4.12 ADC 规则序列寄存器 2(0x34:ADC_SQR2) *252*
14.4.13 ADC 规则序列寄存器 3(0x38:ADC_SQR3) *253*
14.4.14 ADC 规则序列寄存器 4(0x3C:ADC_SQR4) *254*
14.4.15 ADC 规则数据寄存器（0x40:ADC_DR) *254*
14.4.16 ADC 注入序列寄存器(0x4C:ADC_JSQR) *254*
14.4.17 ADC offset y 寄存器(0x60+0x04(y-1):ADC_OFRy) *256*
14.4.18 ADC 注入数据寄存器（0x80+0x04(y-1):ADC_JDRy) *257*
14.4.19 ADC 模拟看门狗 2 配置寄存器（0xA0:ADC_AWD2CR) *257*
14.4.20 ADC 模拟看门狗 3 配置寄存器（0xA4:ADC_AWD3CR) *257*
14.4.21 ADC 差分模式选择寄存器（0xB0:ADC_DIFSEL) *258*
14.4.22 ADC 校准因子寄存器（0xB4:ADC_CALFACT) *258*
14.4.23 ADC 增益补偿寄存器（0xC0:ADC_GCOMP) *259*
14.5 寄存器描述(对于公共 ADC) 259
14.5.1 ADC 公共控制寄存器（0x308:ADC_CCR) *259*
15. 数字/模拟转换（DAC） 262
15.1 简介 262
15.1.1 主要特征 *262*
15.1.2 模块框图 *263*
15.2 DAC 功能描述 264
15.2.1 使能 DAC 通道 *264*
15.2.2 使能 DAC 输出缓存 *264*
15.2.3 DAC 数据格式 *264*
15.2.4 DAC 转换 *265*

15.2.5 DAC 输出电压 266
15.2.6 选择DAC 触发 266
15.2.7 DMA 功能 267
15.2.7.1 DMA 请求 267
15.2.7.2 DMA 下溢检测 267
15.2.8 噪声生成 267
15.2.9 三角波生成 268
15.2.10 双DAC 通道转换 269
15.2.10.1 不使用波形发生器的独立触发 269
15.2.10.2 使用相同 LFSR 的独立触发 270
15.2.10.3 使用不同 LFSR 的独立触发 270
15.2.10.4 产生相同三角波的独立触发 270
15.2.10.5 产生不同三角波的独立触发 271
15.2.10.6 同时非硬件转换 271
15.2.10.7 不使用波形发生器的同时触发 271
15.2.10.8 使用相同 LFSR 的同时触发 271
15.2.10.9 使用不同 LFSR 的同时触发 272
15.2.10.10 使用相同三角波发生器的同时触发 272
15.2.10.11 使用不同三角波发生器的同时触发 272
15.2.11 DAC 输出时钟 273
15.3 寄存器描述 273
15.3.1 DAC 控制寄存器(DAC_CR) 273
15.3.2 DAC 软件触发寄存器(DAC_ SWTRIGR) 276
15.3.3 DAC 通道 1 的 12 位右对齐数据保持寄存器(DAC_DHR12R1) 276
15.3.4 DAC 通道 1 的 12 位左对齐数据保持寄存器(DAC_DHR12L1) 276
15.3.5 DAC 通道 1 的 8 位右对齐数据保持寄存器(DAC_DHR8R1) 277
15.3.6 DAC 通道 2 的 12 位右对齐数据保持寄存器(DAC_DHR12R2) 277
15.3.7 DAC 通道 2 的 12 位左对齐数据保持寄存器(DAC_DHR12L2) 277
15.3.8 DAC 通道 2 的 8 位右对齐数据保持寄存器(DAC_DHR8R2) 278
15.3.9 双DAC 的 12 位右对齐数据保持寄存器(DAC_DHR12RD) 278
15.3.10 双DAC 的 12 位左对齐数据保持寄存器(DAC_DHR12LD) 278
15.3.11 双DAC 的 8 位右对齐数据保持寄存器(DAC_DHR8RD) 279
15.3.12 DAC 通道 1 数据输出寄存器(DAC_DOR1) 279
15.3.13 DAC 通道 2 数据输出寄存器(DAC_DOR2) 280
15.3.14 DAC 状态寄存器(DAC_SR) 280

15.3.15 DAC 输出选择寄存器(DAC_OC) 280

16. 电压基准缓冲器(V_{REFBUF}) 282

16.1 简介 282

16.2 功能描述 282

17. 比较器(COMP) 283

17.1 简介 283

17.1.1 主要特性 283

17.1.2 模块框图 284

17.2 COMP 功能描述 284

17.2.1 COMP 的输入和输出 284

17.2.2 COMP 的时钟和复位 285

17.2.3 COMP 的滤波功能 286

17.2.4 COMP 的锁定机制 286

17.2.5 COMP 的输出消隐 287

17.2.6 COMP 的窗口模式 287

17.2.7 COMP 的迟滞功能 288

17.2.8 COMP 的功耗模式 288

17.2.9 COMP 的低功耗模式 289

17.2.10 COMP 的中断 289

17.2.11 COMP1 控制和状态寄存器(COMP1_CSR) 289

17.2.12 COMP1 滤波寄存器 (COMP1_FR) 291

17.2.13 COMP2 控制和状态寄存器(COMP2_CSR) 291

17.2.14 COMP2 滤波寄存器(COMP2_FR) 293

17.2.15 COMP3 控制和状态寄存器(COMP3_CSR) 293

17.2.16 COMP3 滤波寄存器(COMP3_FR) 295

17.2.17 COMP4 控制和状态寄存器(COMP4_CSR) 295

17.2.18 COMP4 滤波寄存器(COMP4_FR) 297

18. 运算放大器(OPA) 298

18.1 概述 298

18.1.1 主要特性 298

18.1.2 模块框图 298

18.2 OPA 功能描述 299

18.2.1 OPA 输出重定向到内部 ADC 通道 299

18.2.2 OPA 复位和时钟 299

18.2.3 初始配置 299
18.2.4 信号连接 299
18.3 OPA 模式 300
18.3.1 独立模式(外部增益设置模式) 300
18.3.2 可编程增益放大器模式 300
18.3.3 带外部滤波的可编程增益放大器模式 301
18.3.4 可编程增益放大器，带外部偏置的正向或反相模式 302
18.3.5 可编程增益放大器，带外部偏置的正向或带滤波的反相模式 302
18.3.6 电流采样模式 303
18.4 OPA PGA 增益 304
18.5 定时器控制 OPA VINP/VINM 选择模式 304
18.6 OPA 低功耗模式 305
18.7 寄存器描述 305
18.7.1 OPA1 控制/状态寄存器（OPA1_CSR） 305
18.7.2 OPA2 控制/状态寄存器（OPA2_CSR） 307
18.7.3 3 控制/状态寄存器（OPA3_CSR） 309
18.7.4 OPA1 timer 控制模式寄存器（OPA1_TCMR） 311
18.7.5 OPA2 timer 控制模式寄存器（OPA2_TCMR） 312
18.7.6 OPA3 timer 控制模式寄存器（OPA3_TCMR） 313
19. 高级定时器（TIM1 和 TIM8） 315
19.1 简介 315
19.1.1 TIM1/TIM8 主要特征 315
19.1.2 模块框图 316
19.2 TIM1/TIM8 功能描述 316
19.2.1 时基单元 316
19.2.2 计数器模式 318
19.2.3 重复计数器 325
19.2.4 外部触发输入 326
19.2.5 时钟选择 326
19.2.6 捕获/比较通道 329
19.2.7 输入捕获模式 331
19.2.8 PWM 输入模式 331
19.2.9 强置输出模式 332
19.2.10 输出比较模式 332
19.2.11 PWM 模式 333

19.2.12 非对称 PWM 模式 339
19.2.13 组合 PWM 模式 339
19.2.14 三相 PWM 组合模式 340
19.2.15 互补输出和死区插入 341
19.2.15.1 重定向 OCxREF 到 OCx 或 OCxN 343
19.2.16 使用刹车功能 343
19.2.17 双向刹车输入 347
19.2.17.1 对刹车电路进行启用和重启用 349
19.2.18 在外部事件时清除 OCxREF 信号 349
19.2.19 产生六步 PWM 输出 350
19.2.20 单脉冲模式 351
19.2.20.1 特殊情况：OCx 快速使能： 352
19.2.21 可重触发的单脉冲模式 352
19.2.22 比较模式生成脉冲 352
19.2.23 编码器接口模式 354
19.2.23.1 正交编码器 354
19.2.23.2 时钟加方向编码器模式 356
19.2.23.3 定向时钟编码器模式 357
19.2.23.4 索引输入 358
19.2.24 方向位输出 369
19.2.25 UIF 位重映射 369
19.2.26 定时器输入异或功能 369
19.2.27 与霍尔传感器的接口 370
19.2.28 TIMx 定时器和外部触发的同步 371
19.2.28.1 从模式：复位模式 371
19.2.28.2 从模式：门控模式 372
19.2.28.3 从模式：触发模式 373
19.2.28.4 从模式：复位加触发组合模式 373
19.2.28.5 从模式：门控加复位组合模式 373
19.2.28.6 从模式：外部时钟模式 2 + 触发模式 373
19.2.29 DMA 突发模式 374
19.2.30 TIM1/TIM8 DMA 请求 375
19.2.31 定时器同步 375
19.2.32 调试模式 375
19.2.33 TIM1/TIM8 中断 375

19.3 寄存器描述 376
19.3.1 TIM1 和 TIM8 控制寄存器 1（TIMx_CR1） 376
19.3.2 TIM1 和 TIM8 控制寄存器 2（TIMx_CR2） 377
19.3.3 TIM1 和 TIM8 从模式控制寄存器（TIMx_SMCR） 380
19.3.4 TIM1 和 TIM8 DMA/中断使能寄存器（TIMx_DIER） 383
19.3.5 TIM1 和 TIM8 状态寄存器（TIMx_SR） 385
19.3.6 TIM1 和 TIM8 事件产生寄存器（TIMx_EGR） 387
19.3.7 TIM1 和 TIM8 捕获/比较模式控制寄存器 1（TIMx_CCMR1） 388
19.3.7.1 输出比较模式 389
19.3.7.2 输入捕获模式 391
19.3.8 TIM1 和 TIM8 捕获/比较模式控制寄存器 2（TIMx_CCMR2） 392
19.3.8.1 输出比较模式 392
19.3.8.2 输入捕获模式 394
19.3.9 TIM1 和 TIM8 捕获/比较使能寄存器（TIMx_CCER） 394
19.3.10 TIM1 和 TIM8 计数器（TIMx_CNT） 397
19.3.11 TIM1 和 TIM8 预分频器（TIMx_PSC） 398
19.3.12 TIM1 和 TIM8 自动重装载寄存器（TIMx_ARR） 398
19.3.13 TIM1 和 TIM8 重复计数寄存器（TIMx_RCR） 398
19.3.14 TIM1 和 TIM8 捕获/比较寄存器 1（TIMx_CCR1） 399
19.3.15 TIM1 和 TIM8 捕获/比较寄存器 2（TIMx_CCR2） 400
19.3.16 TIM1 和 TIM8 捕获/比较寄存器 3（TIMx_CCR3） 400
19.3.17 TIM1 和 TIM8 捕获/比较寄存器 4（TIMx_CCR4） 401
19.3.18 TIM1 和 TIM8 刹车和死区寄存器（TIMx_BDTR） 402
19.3.19 TIM1 和 TIM8 捕获/比较寄存器 5（TIMx_CCR5） 405
19.3.20 TIM1 和 TIM8 捕获/比较寄存器 6（TIMx_CCR6） 406
19.3.21 TIM1 和 TIM8 捕获/比较模式控制寄存器 3（TIMx_CCMR3） 407
19.3.22 TIM1 和 TIM8 死区时间寄存器 2（TIMx_DTR2） 408
19.3.23 TIM1 和 TIM8 编码器控制寄存器（TIMx_ECR） 409
19.3.24 TIM1 和 TIM8 输入选择寄存器（TIMx_TISEL） 410
19.3.25 TIM1 和 TIM8 备用功能选项寄存器 1（TIMx_AF1） 411
19.3.26 TIM1 和 TIM8 备用功能选项寄存器 2（TIMx_AF2） 413
19.3.27 TIM1 和 TIM8 DMA 控制寄存器（TIMx_DCR） 415
19.3.28 TIM1 和 TIM8 连续模式的 DMA 地址（TIMx_DMAR） 416
20. 通用定时器（TIM2 到 TIM5/TIM18） 417
20.1 简介 417

20.1.1 TIM2/3/4/5/18 主要特征 *417*
20.1.2 模块框图 *418*
20.2 TIM2/3/4/5/18 功能描述 418
20.2.1 时基单元 *418*
20.2.2 计数器模式 *420*
20.2.3 外部触发输入 *427*
20.2.4 时钟选择 *427*
20.2.5 捕获/比较通道 *429*
20.2.6 输入捕获模式 *431*
20.2.7 PWM 输入模式 *431*
20.2.8 强置输出模式 *432*
20.2.9 输出比较模式 *432*
20.2.10 PWM 模式 *433*
20.2.11 非对称 PWM 模式 *438*
20.2.12 组合 PWM 模式 *439*
20.2.13 在外部事件时清除 OCxREF 信号 *440*
20.2.14 单脉冲模式 *441*
20.2.14.1 特殊情况：OCx 快速使能： 442
20.2.15 可重触发的单脉冲模式 *443*
20.2.16 比较模式生成脉冲 *443*
20.2.17 编码器接口模式 *445*
20.2.17.1 正交编码器 445
20.2.17.2 时钟加方向编码器模式 447
20.2.17.3 定向时钟编码器模式 448
20.2.17.4 索引输入 449
20.2.18 方向位输出 *460*
20.2.19 UIF 位重映射 *460*
20.2.20 定时器输入异或功能 *460*
20.2.21 TIMx 定时器和外部触发的同步 *461*
20.2.21.1 从模式：复位模式 461
20.2.21.2 从模式：门控模式 461
20.2.21.3 从模式：触发模式 462
20.2.21.4 从模式：复位加触发组合模式 462
20.2.21.5 从模式：门控加复位组合模式 463
20.2.21.6 从模式：外部时钟模式 2 + 触发模式 463

20.2.22 定时器同步 *463*
20.2.22.1 使用一个定时器作为另一个定时器的预分频器 464
20.2.22.2 使用一个定时器使能另一个定时器 465
20.2.22.3 使用一个定时器去启动另一个定时器 466
20.2.22.4 使用一个外部触发同步地启动 2 个定时器 467
20.2.23 DMA 突发模式 *468*
20.2.24 TIM2/3/4/5/18 DMA 请求 *469*
20.2.25 调试模式 *469*
20.2.26 TIM2/3/4/5/18 低功耗模式 *469*
20.2.27 TIM2/3/4/5/18 中断 *469*
20.3 寄存器描述 470
20.3.1 TIMx 控制寄存器 1（TIMx_CR1）（x=2/3/4/5/18） *470*
20.3.2 TIMx 控制寄存器 2（TIMx_CR2）（x=2/3/4/5/18） *471*
20.3.3 TIMx 从模式控制寄存器（TIMx_SMCR）（x=2/3/4/5/18） *472*
20.3.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x=2/3/4/5/18） *476*
20.3.5 TIMx 状态寄存器（TIMx_SR）（x=2/3/4/5/18） *477*
20.3.6 TIMx 事件产生寄存器（TIMx_EGR）（x=2/3/4/5/18） *479*
20.3.7 TIMx 捕获/比较模式控制寄存器 1（TIMx_CCMR1）（x=2/3/4/5/18） *480*
20.3.7.1 输出比较模式 480
20.3.7.2 输入捕获模式 482
20.3.8 TIMx 捕获/比较模式控制寄存器 2（TIMx_CCMR2）（x=2/3/4/5/18） *483*
20.3.8.1 输出比较模式 483
20.3.8.2 输入捕获模式 485
20.3.9 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x=2/3/4/5/18） *485*
20.3.10 TIMx 计数器（TIMx_CNT）（x=2） *487*
20.3.11 TIMx 计数器（TIMx_CNT）（x=3/4/5/18） *487*
20.3.12 TIMx 预分频器（TIMx_PSC）（x=2/3/4/5/18） *487*
20.3.13 TIMx 自动重装载寄存器（TIMx_ARR）（x=2） *488*
20.3.14 TIMx 自动重装载寄存器（TIMx_ARR）（x=3/4/5/18） *488*
20.3.15 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x=2） *489*
20.3.16 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x=3/4/5/18） *489*
20.3.17 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x=2） *490*
20.3.18 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x=3/4/5/18） *490*
20.3.19 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x=2） *491*
20.3.20 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x=3/4/5/18） *492*

20.3.21 TIMx 捕获比较寄存器 4（TIMx_CCR4）（x=2） 492
20.3.22 TIMx 捕获比较寄存器 4（TIMx_CCR4）（x=3/4/5/18） 493
20.3.23 TIMx 编码器控制寄存器（TIMx_ECR）（x=2/3/4/5/18） 494
20.3.24 TIMx 输入选择寄存器（TIMx_TISEL）（x=2/3/4/5/18） 495
20.3.25 TIMx 备用功能选项寄存器 1（TIMx_AF1）（x=2/3/4/5/18） 496
20.3.26 TIMx 备用功能选项寄存器 2（TIMx_AF2）（x=2/3/4/5/18） 496
20.3.27 TIMx DMA 控制寄存器（TIMx_DCR）（x=2/3/4/5/18） 497
20.3.28 TIMx 连续模式的 DMA 地址（TIMx_DMAR）（x=2/3/4/5/18） 498

21. 通用定时器（TIM9/TIM12） **499**

21.1 简介 499
21.1.1 TIM9 和 TIM12 主要特征 499
21.1.2 模块框图 500
21.2 TIM9 和 TIM12 功能描述 500
21.2.1 时基单元 500
21.2.2 计数器模式 501
21.2.3 时钟选择 503
21.2.4 捕获比较通道 505
21.2.5 输入捕获模式 506
21.2.6 PWM 输入模式 507
21.2.7 强置输出模式 508
21.2.8 输出比较模式 508
21.2.9 PWM 模式 509
21.2.10 单脉冲模式 510
21.2.10.1 特殊情况：OCx 快速使能： 511
21.2.11 TIMx 定时器和外部触发的同步 511
21.2.11.1 从模式：复位模式 511
21.2.11.2 从模式：门控模式 512
21.2.11.3 从模式：触发模式 512
21.2.12 定时器同步 513
21.2.13 调试模式 513
21.3 寄存器描述 513
21.3.1 TIM9 和 TIM12 控制寄存器 1（TIMx_CR1） 513
21.3.2 TIM9 和 TIM12 从模式控制寄存器（TIMx_SMCR） 514
21.3.3 TIM9 和 TIM12 DMA/中断使能寄存器（TIMx_DIER） 515
21.3.4 TIM9 和 TIM12 状态寄存器（TIMx_SR） 516

21.3.5 TIM9 和 TIM12 事件产生寄存器（TIMx_EGR） *517*
21.3.6 TIM9 和 TIM12 捕获/比较模式控制寄存器 1（TIMx_CCMR1） *518*
21.3.6.1 输出比较模式 518
21.3.6.2 输入捕获模式 519
21.3.7 TIM9 和 TIM12 捕获/比较使能寄存器（TIMx_CCER） *520*
21.3.8 TIM9 和 TIM12 计数器（TIMx_CNT） *522*
21.3.9 TIM9 和 TIM12 预分频器（TIMx_PSC） *522*
21.3.10 TIM9 和 TIM12 自动重装载寄存器（TIMx_ARR） *522*
21.3.11 TIM9 和 TIM12 捕获/比较寄存器 1（TIMx_CCR1） *523*
21.3.12 TIM9 和 TIM12 捕获/比较寄存器 2（TIMx_CCR2） *523*
22. 通用定时器（TIM10/TIM11/TIM13/TIM14/TIM19） **524**
22.1 简介 524
22.1.1 TIMx 主要特征 *524*
22.1.2 模块框图 *524*
22.2 TIMx 功能描述 525
22.2.1 时基单元 *525*
22.2.2 计数器模式 *526*
22.2.3 时钟选择 *529*
22.2.4 捕获/比较通道 *529*
22.2.5 输入捕获模式 *530*
22.2.6 强置输出模式 *531*
22.2.7 输出比较模式 *531*
22.2.8 PWM 模式 *532*
22.2.9 单脉冲模式 *533*
22.2.10 定时器同步 *533*
22.2.11 调试模式 *533*
22.3 寄存器描述 534
22.3.1 TIMX 控制寄存器 1（TIMx_CR1） *534*
22.3.2 TIMX DMA/中断使能寄存器（TIMx_DIER） *535*
22.3.3 TIMX 状态寄存器（TIMx_SR） *535*
22.3.4 TIMX 事件产生寄存器（TIMx_EGR） *536*
22.3.5 TIMX 捕获/比较模式控制寄存器 1（TIMx_CCMR1） *537*
22.3.5.1 输出比较模式 537
22.3.5.2 输入捕获模式 538
22.3.6 TIMX 捕获/比较使能寄存器（TIMx_CCER） *539*

22.3.7 TIMX 计数器（TIMx_CNT） *540*
22.3.8 TIMX 预分频器（TIMx_PSC） *540*
22.3.9 TIMX 自动重装载寄存器（TIMx_ARR） *540*
22.3.10 TIMX 捕获/比较寄存器 1（TIMx_CCR1） *541*
23. 通用定时器（TIM15/TIM16/TIM17） **542**
23.1 简介 542
23.1.1 TIM15 主要特征 *542*
23.1.2 模块框图 *543*
23.1.3 TIM16_17 主要特征 *543*
23.1.4 模块框图 *544*
23.2 TIM15_16_17 功能描述 544
23.2.1 时基单元 *544*
23.2.2 计数器模式 *545*
23.2.3 重复计数器 *548*
23.2.4 时钟选择 *549*
23.2.5 捕获/比较通道 *551*
23.2.6 输入捕获模式 *552*
23.2.7 PWM 输入模式（仅 tim15） *553*
23.2.8 强置输出模式 *554*
23.2.9 输出比较模式 *554*
23.2.10 PWM 模式 *555*
23.2.11 组合 PWM 模式（仅 tim15） *558*
23.2.12 互补输出和死区插入 *559*
23.2.12.1 重定向 OCxREF 到 OCx 或 OCxN 561
23.2.13 使用刹车功能 *561*
23.2.14 双向刹车输入 *564*
23.2.14.1 对刹车电路进行启用和重启用 565
23.2.15 在外部事件时清除 OCxREF 信号 *565*
23.2.16 产生六步 PWM 输出 *566*
23.2.17 单脉冲模式 *567*
23.2.17.1 特殊情况：OCx 快速使能： 568
23.2.18 可重触发的单脉冲模式(仅 tim15) *568*
23.2.19 UIF 位重映射 *569*
23.2.20 定时器输入异或功能（仅 tim15） *569*
23.2.21 TIMx 定时器和外部触发的同步 *569*

23.2.21.1 从模式：复位模式 569
23.2.21.2 从模式：门控模式 570
23.2.21.3 从模式：触发模式 570
23.2.21.4 从模式：复位加触发组合模式（仅 tim15） 571
23.2.21.5 从模式：门控加复位组合模式（仅 tim15） 571
23.2.22 定时器同步（仅 tim15） 571
23.2.23 DMA 突发模式 571
23.2.24 DMA 请求 572
23.2.25 调试模式 572
23.2.26 TIM15/TIM16/TIM17 中断 572
23.3 TIM15 寄存器描述 573
23.3.1 TIM15 控制寄存器 1（TIMx_CR1） 573
23.3.2 TIM15 控制寄存器 2（TIMx_CR2） 574
23.3.3 TIM15 从模式控制寄存器（TIMx_SMCR） 575
23.3.4 TIM15 DMA/中断使能寄存器（TIMx_DIER） 577
23.3.5 TIM15 状态寄存器（TIMx_SR） 578
23.3.6 TIM15 事件产生寄存器（TIMx_EGR） 580
23.3.7 TIM15 捕获比较模式控制寄存器 1（TIMx_CCMR1） 581
23.3.7.1 输出比较模式 581
23.3.7.2 输入捕获模式 583
23.3.8 TIM15 捕获比较使能寄存器（TIMx_CCER） 584
23.3.9 TIM15 计数器（TIMx_CNT） 586
23.3.10 TIM15 预分频器（TIMx_PSC） 587
23.3.11 TIM15 自动重装载寄存器（TIMx_ARR） 587
23.3.12 TIM15 重复计数寄存器（TIMx_RCR） 588
23.3.13 TIM15 捕获比较寄存器 1（TIMx_CCR1） 588
23.3.14 TIM15 捕获比较寄存器 2（TIMx_CCR2） 589
23.3.15 TIM15 刹车和死区寄存器（TIMx_BDTR） 589
23.3.16 TIM15 死区时间寄存器 2（TIMx_DTR2） 592
23.3.17 TIM15 输入选择寄存器（TIMx_TISEL） 593
23.3.18 TIM15 备用功能选项寄存器 1（TIMx_AF1） 593
23.3.19 TIM15 备用功能选项寄存器 2（TIMx_AF2） 595
23.3.20 TIM15 DMA 控制寄存器（TIMx_DCR） 596
23.3.21 TIM15 连续模式的 DMA 地址（TIMx_DMAR） 597
23.4 TIM16_TIM17 寄存器描述 597

23.4.1 TIM16_TIM17 控制寄存器 1（TIMx_CR1） 597
23.4.2 TIM16_TIM17 控制寄存器 2（TIMx_CR2） 598
23.4.3 TIM16_TIM17 DMA/中断使能寄存器（TIMx_DIER） 599
23.4.4 TIM16_TIM17 状态寄存器（TIMx_SR） 600
23.4.5 TIM16_TIM17 事件产生寄存器（TIMx_EGR） 601
23.4.6 TIM16_TIM17 捕获比较模式控制寄存器 1（TIMx_CCMR1） 602
23.4.6.1 输出比较模式 602
23.4.6.2 输入捕获模式 603
23.4.7 TIM16_TIM17 捕获比较使能寄存器（TIMx_CCER） 604
23.4.8 TIM16_TIM17 计数器（TIMx_CNT） 606
23.4.9 TIM16_TIM17 预分频器（TIMx_PSC） 607
23.4.10 TIM16_TIM17 自动重装载寄存器（TIMx_ARR） 607
23.4.11 TIM16_TIM17 重复计数寄存器（TIMx_RCR） 607
23.4.12 TIM16_TIM17 捕获比较寄存器 1（TIMx_CCR1） 608
23.4.13 TIM16_TIM17 死区时间寄存器 2（TIMx_DTR2） 608
23.4.14 TIM16_TIM17 刹车和死区寄存器（TIMx_BDTR） 609
23.4.15 TIM16_TIM17 输入选择寄存器（TIMx_TISEL） 612
23.4.16 TIM16_TIM17 备用功能选项寄存器 1（TIMx_AF1） 612
23.4.17 TIM16_TIM17 备用功能选项寄存器 2（TIMx_AF2） 614
23.4.18 TIM16_TIM17 option 寄存器（TIMx_OR1） 615
23.4.19 TIM16_TIM17 DMA 控制寄存器（TIMx_DCR） 615
23.4.20 TIM16_TIM17 连续模式的 DMA 地址（TIMx_DMAR） 616
24. 基本定时器（TIM6 和 TIM7） **617**
24.1 简介 617
24.1.1 TIM6 和 TIM7 主要特征 617
24.1.2 模块框图 617
24.2 TIMx 功能描述 618
24.2.1 时基单元 618
24.2.2 计数器模式 619
24.2.3 调试模式 621
24.3 寄存器描述 622
24.3.1 TIM6 和 TIM7 控制寄存器 1（TIMx_CR1） 622
24.3.2 TIM6 和 TIM7 控制寄存器 2（TIMx_CR2） 623
24.3.3 TIM6 和 TIM7DMA/中断使能寄存器（TIMx_DIER） 623
24.3.4 TIM6 和 TIM7 状态寄存器（TIMx_SR） 624

24.3.5 TIM6 和 TIM7 事件产生寄存器（TIMx_EGR） *624*
24.3.6 TIM6 和 TIM7 计数器（TIMx_CNT） *624*
24.3.7 TIM6 和 TIM7 预分频器（TIMx_PSC） *625*
24.3.8 TIM6 和 TIM7 自动重装载寄存器（TIMx_ARR） *625*
25. 低功耗定时器(LPTIM) **626**
25.1 简介 626
25.1.1 主要特性 *626*
25.1.2 模块框图 *627*
25.2 外部信号说明 627
25.2.1 LPTIM 输入和触发映射 *627*
25.3 LPTIM 功能描述 628
25.3.1 复位和时钟 *628*
25.3.2 毛刺滤波器 *628*
25.3.3 预分频 *629*
25.3.4 触发选择 *629*
25.3.5 操作模式 *629*
25.3.5.1 单次计数模式 629
25.3.5.2 连续计数模式 630
25.3.5.3 波形产生 631
25.3.6 寄存器更新 *631*
25.3.7 计数器模式 *632*
25.3.8 定时器使能 *632*
25.3.8.1 编码器模式 632
25.3.9 Debug 模式 *633*
25.3.10 LPTIM 低功耗模式 *633*
25.3.11 LPTIM 中断 *634*
25.4 寄存器描述 634
25.4.1 LPTIM 中断和状态寄存器，Address=0x00(LPTIM_ISR) *634*
25.4.2 LPTIM 中断清零寄存器，Address=0x04(LPTIM_ICR) *635*
25.4.3 LPTIM 中断使能寄存器，Address=0x08(LPTIM_IER) *636*
25.4.4 LPTIM 配置寄存器，Address=0x0C(LPTIM_CFGR) *637*
25.4.5 LPTIM 控制寄存器，Address=0x10(LPTIM_CR) *639*
25.4.6 LPTIM 比较寄存器，Address=0x14(LPTIM_CMP) *640*
25.4.7 LPTIM 自动重载寄存器，Address=0x18(LPTIM_ARR) *640*
25.4.8 LPTIM 计数寄存器，Address=0x1C(LPTIM_CNT) *640*

25.4.9 LPTIM 选择寄存器，Address=0x20(LPTIM_OR) *641*

26. 红外接口(IRTIM) **642**

27. 实时时钟（RTC） **643**

27.1 概述 643

27.2 主要特性 643

27.3 功能描述 644

27.3.1 概述 *644*

27.3.2 寄存器复位 *645*

27.3.3 读 RTC 寄存器 *645*

27.3.4 配置 RTC 寄存器 *645*

27.3.5 RTC 标志设置 *646*

27.3.6 RTC 时序 *646*

27.4 寄存器描述 647

27.4.1 RTC 控制寄存器高位(RTC_CRH)(0x00) *647*

27.4.2 RTC 控制寄存器低位(RTC_CRL)(0x04) *648*

27.4.3 RTC 预分频装载寄存器高位(RTC_PRLH)(0x08) *649*

27.4.4 RTC 预分频装载寄存器低位(RTC_PRLL)(0x0C) *649*

27.4.5 RTC 预分频余数寄存器高位(RTC_DIVH)(0x10) *650*

27.4.6 RTC 预分频余数寄存器低位(RTC_DIVL)(0x14) *650*

27.4.7 RTC 计数寄存器高位(RTC_CNTH)(0x18) *651*

27.4.8 RTC 计数寄存器低位(RTC_CNTL)(0x1C) *651*

27.4.9 RTC 闹钟寄存器高位(RTC_ALRH)(0x20) *651*

27.4.10 RTC 闹钟寄存器低位(RTC_ALRL)(0x24) *652*

28. 独立看门狗（IWDG） **653**

28.1 简介 653

28.2 主要特性 653

28.3 功能说明 653

28.3.1 模块框图 *653*

28.3.2 硬件看门狗 *654*

28.3.3 寄存器保护 *654*

28.3.4 调试模式 *654*

28.3.5 低功耗冻结 *654*

28.4 寄存器描述 654

28.4.1 键值寄存器（IWDG_KR） *654*

28.4.2 预分频寄存器（IWDG_PR）..... 655
28.4.3 重装载寄存器（IWDG_RLR）..... 655
28.4.4 状态寄存器（IWDG_SR）..... 656
29. 窗口看门狗（WWDG）..... 657
29.1 简介..... 657
29.2 主要特性..... 657
29.3 WWDG 功能描述..... 657
29.4 如何编写看门狗超时程序..... 658
29.5 调试模式..... 659
29.6 寄存器描述..... 659
29.6.1 控制寄存器（WWDG_CR）..... 659
29.6.2 配置寄存器（WWDG_CFR）..... 659
29.6.3 状态寄存器（WWDG_SR）..... 660
30. 外部串行存储控制器（ESMC）..... 661
30.1 简介..... 661
30.2 主要特征..... 661
30.3 ESMC 功能描述..... 661
30.3.1 ESMC 外部存储器连接图..... 661
30.3.2 ESMC 命令序列..... 662
30.3.3 ESMC 接口协议模式..... 664
30.3.4 ESMC 的工作模式..... 666
30.3.4.1 ESMC 间接模式..... 667
30.3.4.2 ESMC 内存映射模式..... 667
30.3.5 ESMC 的可编程选项..... 667
30.3.5.1 字节计数..... 667
30.3.5.2 极性和相位..... 668
30.3.5.3 波特率..... 668
30.3.5.4 时序控制..... 669
30.3.5.5 ESMC 命令发送..... 669
30.3.5.6 ESMC 命令序列配置..... 669
30.3.5.7 Dummy 周期控制..... 671
30.3.5.8 从机选择输出控制..... 672
30.3.5.9 FIFO 的大小..... 672
30.3.5.10 采样延伸..... 672

30.3.6 DMA 模式 672
30.3.7 ESMC 中断 673
30.4 ESMC 寄存器描述(只支持字节访问) 673
30.4.1 配置寄存器（ESMC_CR） 674
30.4.2 配置寄存器 2（ESMC_CR2) 674
30.4.3 时序控制寄存器（ESMC_TCR） 675
30.4.4 波特率寄存器(ESMC_BAUD) 676
30.4.5 SPI Flash 命令寄存器(ESMC_SFCR) 676
30.4.6 SPI 输出控制寄存器(ESMC_SOCR) 676
30.4.7 Dummy 控制寄存器(ESMC_DCR) 677
30.4.8 配置寄存器 3(ESMC_CR3) 677
30.4.9 状态寄存器(ESMC_SR) 678
30.4.10 状态寄存器 2(ESMC_SR2) 678
30.4.11 RX_FIFO 状态寄存器(ESMC_RXSTAT) 679
30.4.12 中断标志寄存器(ESMC_IFR1) 679
30.4.13 中断使能寄存器(ESMC_IER1) 680
30.4.14 中断标志寄存器(ESMC_IFR2) 681
30.4.15 中断使能寄存器(ESMC_IER2) 681
30.4.16 采样延伸寄存器（ESMC_STRR） 682
30.4.17 接收超时寄存器(ESMC_RXTOUT) 682
30.5 XIP 模式控制寄存器–读操作 683
30.5.1 XIP SPI Flash 命令寄存器(ESMC_XSFCR) 683
30.5.2 XIP SPI 输出控制寄存器(ESMC_XSOCR) 683
30.5.3 XIP 读 Dummy 控制寄存器(ESMC_XDCR) 684
30.5.4 XIP 配置寄存器 3(ESMC_XCR3) 684
30.6 XIP 模式控制通用寄存器 684
30.6.1 XIP 模式寄存器（ESMC_XMODE） 684
30.6.2 XIP 写模式寄存器（ESMC_XMODE_WE） 685
30.6.3 XIP 从选择输出控制寄存器(ESMC_XSSOCR) 685
30.6.4 XIP 从选择输出控制寄存器(ESMC_XSSOCR_WE) 685
30.6.5 XIP 模式采样拉伸寄存器(ESMC_XSTRR) 686
30.7 XIP 模式控制寄存器–写入操作 686
30.7.1 XIP 模式写入 SPI flash 命令寄存器(ESMC_XSFCR_WE) 686
30.7.2 XIP 模式读取 SPI 输出控制寄存器（ESMC_XSOCR_WE） 687
30.7.3 XIP 写 Dummy 周期控制寄存器(ESMC_XDCR_WE) 687

30.7.4 XIP 写入配置寄存器 3(ESMC_XCR3_WE) 688
30.8 ESMC 寄存器描述 688
30.8.1 24 位地址寄存器（ESMC_ADDR24） 688
30.8.2 32 位地址寄存器（ESMC_ADDR32） 689
30.8.3 数据寄存器（ESMC_DATA） 689
30.8.4 传输计数寄存器(ESMC_BCR) 690
31. SDIO 接口（SDIO） 691
31.1 简介 691
31.2 SDIO 适配器 692
31.2.1 控制单元 692
31.2.2 命令单元 692
31.2.3 数据单元 695
31.2.4 SDIO 时钟控制 696
31.3 SDIO AHB 接口 697
31.3.1 SDIO 中断 697
31.3.2 SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程 697
31.4 卡功能描述 697
31.4.1 卡识别模式 697
31.4.2 卡复位 698
31.4.3 操作电压范围确认 698
31.4.4 卡识别过程 698
31.4.5 写数据块 699
31.4.6 读数据块 699
31.4.7 擦除：成组擦除和扇区擦除 700
31.4.8 宽总线选择和解除选择 700
31.4.9 保护管理 700
31.4.10 卡状态寄存器 700
31.4.11 SD 状态寄存器 702
31.4.12 SD 的 I/O 模式 705
31.4.13 命令和响应 706
31.4.13.1 应用相关命令和通用命令 706
31.4.13.2 命令类型 707
31.4.13.3 命令格式 707
31.5 响应格式 709
31.5.1 R1(普通响应命令) 709

31.5.2 R1b 709
31.5.3 R2(CSD、CSD 寄存器) 709
31.5.4 R3(OCR 寄存器) 710
31.5.5 R4（快速 I/O） 710
31.5.6 R4b 710
31.5.7 R5（中断请求） 711
31.5.8 R6（中断请求） 711
31.6 编程序列 712
31.6.1 卡识别 712
31.6.1.1 卡复位 712
31.6.1.2 工作电压范围验证 712
31.6.1.3 卡识别过程 713
31.6.2 无数据命令 713
31.6.3 写数据命令 713
31.6.4 单个数据块或多个数据块读 714
31.7 寄存器描述 714
31.7.1 SDIO 电源控制寄存器（SDIO_POWER） 714
31.7.2 SDIO 时钟控制寄存器（SDIO_CLKCR） 715
31.7.3 SDIO 参数寄存器（SDIO_ARG） 716
31.7.4 SDIO 命令寄存器（SDIO_CMD） 716
31.7.5 SDIO 命令响应寄存器（SDIO_RESPCMD） 718
31.7.6 SDIO 命令响应寄存器（SDIO_RESPX） 718
31.7.7 SDIO 数据定时器寄存器（SDIO_TMOUT） 719
31.7.8 SDIO 数据块长度寄存器（SDIO_BLKSIZE） 719
31.7.9 SDIO 数据长度寄存器（SDIO_DLEN） 719
31.7.10 SDIO 控制寄存器（SDIO_CTRL） 720
31.7.11 SDIO 状态寄存器（SDIO_STA） 721
31.7.12 SDIO 中断状态寄存器（SDIO_INTSTS） 722
31.7.13 SDIO 中断屏蔽寄存器（SDIO_INTMASK） 723
31.7.14 SDIO FIFO 阈值寄存器（SDIO_FIFOTH） 724
31.7.15 SDIO 发送到卡计数器（SDIO_TCBCNT） 725
31.7.16 SDIO 发送到 FIFO 计数器(SDIO_TBBCNT) 725
31.7.17 SDIO 数据 FIFO 寄存器（SDIO_FIFODATA） 725
32. USB OTG 全速（OTG_FS） 727
32.1 概述 727

32.1.1 主要特性 727
32.1.1.1 通用功能 727
32.1.1.2 主机模式功能 728
32.1.1.3 设备模式功能 728
32.1.2 结构框图 728
32.2 功能描述 729
32.2.1 OTG 全速控制器 *729*
32.2.2 全速 OTG PHY(物理接口) *729*
32.2.3 OTG 双角色设备(DRD) *730*
32.2.3.1 ID 信号检测 730
32.2.3.2 HNP 双角色设备 730
32.2.3.3 SRP 双角色设备 731
32.2.4 USB 设备模式 *731*
32.2.4.1 支持 SRP 功能的设备 732
32.2.4.2 设备状态 732
32.2.4.3 设备端点 732
32.2.5 USB 主机 *734*
32.2.5.1 具备 SRP 功能的主机 735
32.2.5.2 USB 主机状态 735
32.2.5.3 主机通道 737
32.2.5.4 主机调度器 738
32.2.6 SOF 触发 *739*
32.2.6.1 主机 SOF 739
32.2.6.2 设备 SOF 740
32.2.7 供电选项 *740*
32.2.8 USB 数据 FIFO *741*
32.2.9 设备模式下的 FIFO 结构 *742*
32.2.9.1 设备模式下的接收 FIFO 742
32.2.9.2 设备模式下的发送 FIFO 742
32.2.10 主机模式下的 FIFO 结构 *743*
32.2.10.1 主机模式下的接收 FIFO 743
32.2.10.2 主机模式下的发送 FIFO 744
32.2.11 USB 系统性能 *744*
32.2.12 OTG_FS 中断 *745*
32.3 USBFS 寄存器 746

32.3.1 CSR 存储器映像 746
32.4 OTG_FS 全局寄存器 749
32.4.1 OTG_FS 控制和状态寄存器(OTG_FS_GOTGCTL) 750
32.4.2 OTG_FS 中断寄存器(OTG_FS_GOTGINT) 751
32.4.3 OTG_FS AHB 配置寄存器(OTG_FS_GAHBCFG) 752
32.4.4 OTG_FS 配置寄存器(OTG_FS_GUSBCFG) 753
32.4.5 OTG_FS 复位寄存器(OTG_FS_GRSTCTL) 754
32.4.6 OTG_FS 控制器中断寄存器(OTG_FS_GINTSTS) 756
32.4.7 OTG_FS 中断屏蔽寄存器(OTG_FS_GINTMSK) 760
32.4.8 OTG_FS 接收状态调试读OTG 状态读和 POP 寄存器(OTG_FS_GRXSTSR /OTG_FS_GRXSTSP) 762
32.4.9 OTG_FS 接收 FIFO 长度寄存器(OTG_FS_GRXFSIZ) 763
32.4.10 OTG_FS 非周期性 TX FIFO 长度寄存器(OTG_FS_GNPTXFSIZ) 763
32.4.11 OTG_FS 非周期性 TX FIFO/请求队列状态寄存器(OTG_FS_GNPTXSTS) 764
32.4.12 OTG_FS 通用控制器配置寄存器(OTG_FS_GCCFG) 765
32.4.13 OTG_FS 控制器 ID 寄存器(OTG_FS_CID) 766
32.4.14 OTG_FS 主机周期性发送 FIFO 长度寄存器(OTG_FS_HPTXFSIZ) 766
32.4.15 OTG_FS 设备 IN 端点发送 FIFO 长度寄存器(OTG_FS_DIEPTXFx)(其中 x 是 FIFO 的编号, x=1…3) 766
32.5 主机模式下的寄存器 767
32.5.1 OTG_FS 主机模式配置寄存器(OTG_FS_HCFG) 767
32.5.2 OTG_FS 主机帧间隔寄存器(OTG_FS_HFIR) 768
32.5.3 OTG_FS 主机帧号/帧时间剩余寄存器(OTG_FS_HFNUM) 768
32.5.4 主 OTG_FS 主机周期性发送 FIFO/请求队列寄存器(OTG_FS_HPTXSTS) 769
32.5.5 OTG_FS 主机所有通道中断寄存器(OTG_FS_HAINT) 770
32.5.6 OTG_FS 主机所有通道中断屏蔽寄存器(OTG_FS_HAINTMSK) 770
32.5.7 OTG_FS 主机端口控制和状态寄存器(OTG_FS_HPRT) 770
32.5.8 OTG_FS 主机通道 x 特性寄存器(OTG_FS_HCCHARx)(此处 x 代码通道号，x = 0...7) ... 773
32.5.9 OTG_FS 主机通道 x 中断寄存器(OTG_FS_HCINTx)(其中 x 代表通道号，x=0...7) 774
32.5.10 OTG_FS 主机通道 x 中断屏蔽寄存器(OTG_FS_HCINTMSKx)(其中 x 为通道号，x=0...7) 775
32.5.11 OTG_FS 主机通道 x 传输长度寄存器(OTG_FS_HCTSIZx)(其中 x 为通道号，x=0...7) 776
32.6 设备模式下的寄存器 776
32.6.1 OTG_FS 设备配置寄存器(OTG_FS_DCFG) 776
32.6.2 OTG_FS 设备控制寄存器(OTG_FS_DCTL) 777
32.6.3 OTG_FS 设备状态寄存器(OTG_FS_DSTS) 779

32.6.4 OTG_FS 设备 IN 端点通用中断屏蔽寄存器(OTG_FS_DIEPMSK) *780*
32.6.5 OTG_FS 设备 OUT 端点通用中断屏蔽寄存器(OTG_FS_DOEPMSK) *780*
32.6.6 OTG_FS 设备所有端点中断寄存器(OTG_FS_DAINT) *781*
32.6.7 OTG_FS 所有端点中断屏蔽寄存器(OTG_FS_DAINTMSK) *782*
32.6.8 OTG_FS 设备 VBUS 放电时间寄存器(OTG_FS_DVBUSDIS) *783*
32.6.9 OTG_FS 设备 VBUS 脉冲时间寄存器(OTG_FS_DVBUSPULSE) *783*
32.6.10 OTG_FS 设备 IN 端点 FIFO 空中断屏蔽寄存器(OTG_FS_DIEPEMPMSK) *783*
32.6.11 OTG_FS 设备控制 IN 端点 0 控制寄存器(OTG_FS_DIEPCTL0) *784*
32.6.12 OTG 设备端点 x 控制寄存器(OTG_FS_DIEPCTLx)(其中 x 为端点号，x=1…3) *785*
32.6.13 OTG_FS 设备控制 OUT 端点 0 控制寄存器(OTG_FS_DOEPCTL0) *787*
32.6.14 OTG_FS 设备 OUT 端点 x 控制寄存器(OTG_FS_DOEPCTLx)(其中 x 为端点号，x=1…3) *789*
32.6.15 OTG_FS 设备端点 x 中断寄存器(OTG_FS_DIEPINTx)(其中 x 为端点号，x=0…3) *791*
32.6.16 OTG_FS 设备端点 x 中断寄存器(OTG_FS_DOEPINTx)(其中 x 为端点号，x=0…3) *792*
32.6.17 OTG_FS 设备 IN 端点 0 传输长度寄存器(OTG_FS_DIEPTSIZ0) *793*
32.6.18 OTG_FS 设备 OUT 端点 0 传输长度寄存器(OTG_FS_DOEPTSIZ0) *794*
32.6.19 OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DIEPTSIZx)(其中 x 为端点号，x=1…3) *794*
32.6.20 OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DOEPTSIZx)(其中 x 为端点号，x=1…3) *795*
32.6.21 OTG_FS 设备 IN 端点传输 FIFO 状态寄存器(OTG-FS_DTXFSTSx)(其中 x 为端点号，x=0…3) *796*
32.7 OTG_FS 电源和时钟门控寄存器(OTG_FS_PCGCCTL) 796
33. 以太网（ENET） **798**
33.1 以太网简介 798
33.2 以太网特征 798
33.2.1 DMA 特征 *799*
33.2.2 PTP 特征 *799*
33.2.3 模块框图 *800*
33.3 以太网功能描述：SMI、MII 和 RMII 800
33.3.1 状态管理接口 SMI *800*
33.3.1.1 SMI 帧格式 801
33.3.1.2 SMI 写操作 801
33.3.1.3 SMI 读操作 802
33.3.1.4 SMI 时钟选择 802
33.3.2 独立介质接口 MII *803*
33.3.2.1 MII 时钟源 804

33.3.3 简化的独立介质接口 RMII *804*
33.3.3.1 RMII 时钟源 805
33.3.4 MII/RMII 选择 *805*
33.4 以太网功能描述：MAC 802.3 805
33.4.1 MAC 802.3 帧格式 *806*
33.4.2 MAC 帧发送 *808*
33.4.3 MAC 帧接收 *814*
33.4.4 MAC 中断 *818*
33.4.5 MAC 过滤 *818*
33.4.6 MAC 回环模式 *820*
33.4.7 MAC 管理计数器：MMC *820*
33.4.8 电源管理：PMT *821*
33.4.9 精确时间协议（IEEE1588 PTP） *823*
33.4.9.1 参考时钟源 824
33.4.9.2 具有 PTP 特性发送帧 825
33.4.9.3 具有 PTP 特性接收帧 825
33.4.9.4 系统时间校正方法 825
33.4.9.5 初始化生成系统时间的编程步骤 826
33.4.9.6 粗调校正方法中系统时间的更新步骤 827
33.4.9.7 细调校正方法中系统时间的更新步骤 827
33.4.9.8 PTP 触发内部连接到 TIM2 827
33.4.9.9 PTP PPS 输出信号 827
33.5 DMA 控制器 828
33.5.1 使用 DMA 传输的初始化过程 *828*
33.5.2 主机突发访问 *829*
33.5.3 主机数据缓存区对齐 *829*
33.5.4 缓存区大小计算 *829*
33.5.5 DMA 仲裁 *830*
33.5.6 DMA 错误状态 *830*
33.5.7 Tx DMA 的配置 *830*
33.5.7.1 非 OSF 模式（默认模式） 830
33.5.7.2 OSF 模式 832
33.5.7.3 发送帧的进程 833
33.5.7.4 发送轮询被暂停 834
33.5.7.5 发送 DMA 描述符 834

33.5.8 Rx DMA 的配置 *837*
33.5.8.1 获取接收描述符 838
33.5.8.2 接收帧进程 838
33.5.8.3 接收进程挂起 839
33.5.8.4 DMA 接收描述符 839
33.5.9 使能 IEEE1588-2005 时间戳的描述符格式 *842*
33.5.9.1 发送描述符 842
33.5.9.2 接收描述符 843
33.5.9.3 DMA 中断 844
33.6 以太网中断 844
33.7 寄存器描述 845
33.7.1 以太网 MAC 配置寄存器（ETH_MACCR） *845*
33.7.2 以太网 MAC 帧过滤寄存器（ETH_MACFFR） *847*
33.7.3 以太网 MAC 哈希表高位寄存器（ETH_MACHTHR） *848*
33.7.4 以太网 MAC 哈希表低位寄存器（ETH_MACHTLR） *849*
33.7.5 以太网 MAC MII 地址滤寄存器（ETH_MACMIIAR） *849*
33.7.6 以太网 MAC MII 数据寄存器（ETH_MACMIIDR） *850*
33.7.7 以太网 MAC 流控寄存器（ETH_MACFCR） *850*
33.7.8 以太网 MAC VLAN 标签寄存器（ETH_MACVLANTR） *851*
33.7.9 以太网 MAC 远程唤醒帧过滤寄存器（ETH_MACRWUFFR） *852*
33.7.10 以太网 MAC 控制和状态寄存器（ETH_MACPMTCSR） *852*
33.7.11 以太网 MAC 中断状态寄存器（ETH_MACSR） *853*
33.7.12 以太网 MAC 中断屏蔽寄存器（ETH_MACIMR） *854*
33.7.13 以太网 MAC 地址 0 高位寄存器（ETH_MACA0HR） *854*
33.7.14 以太网 MAC 地址 0 低位寄存器（ETH_MACA0LR） *855*
33.7.15 以太网 MAC 地址 1 高位寄存器（ETH_MACA1HR） *855*
33.7.16 以太网 MAC 地址 1 低位寄存器（ETH_MACA1LR） *856*
33.7.17 以太网 MAC 地址 2 高位寄存器（ETH_MACA2HR） *856*
33.7.18 以太网 MAC 地址 2 低位寄存器（ETH_MACA2LR） *857*
33.7.19 以太网 MAC 地址 3 高位寄存器（ETH_MACA3HR） *857*
33.7.20 以太网 MAC 地址 3 低位寄存器（ETH_MACA3LR） *858*
33.7.21 以太网 MMC 控制寄存器（ETH_MMCCR） *858*
33.7.22 以太网 MMC 接收中断寄存器（ETH_MMCRIR） *859*
33.7.23 以太网 MMC 发送中断寄存器（ETH_MMCTIR） *859*
33.7.24 以太网 MMC 接收中断屏蔽寄存器（ETH_MMCRIMR） *860*

33.7.25 以太网 MMC 发送中断屏蔽寄存器（ETH_MMCTIMR） *860*
33.7.26 以太网 MMC 单次冲突后发送好帧计存器（ETH_MMCTGFSCCR） *861*
33.7.27 以太网 MMC 多次冲突后发送好帧计数寄存器（ETH_MMCTGFMSCCR） *861*
33.7.28 以太网 MMC 发送好帧计数寄存器（ETH_MMCTGFR） *861*
33.7.29 以太网 MMC CRC 错误接收帧计数寄存器（ETH_MMCRFCECR） *861*
33.7.30 以太网 MMC 对齐错误接收帧计数寄存器（ETH_MMCRFAECR） *862*
33.7.31 以太网 MMC 对齐错误接收帧计数寄存器（ETH_MMCRFAECR） *862*
33.7.32 以太网 PTP 时间戳控制寄存器（ETH_PTPTSCR） *862*
33.7.33 以太网 PTP 亚秒递增寄存器（ETH_PTPSSIR） *863*
33.7.34 以太网 PTP 时间戳高位寄存器（ETH_PTPTSHR） *863*
33.7.35 以太网 PTP 时间戳低位寄存器（ETH_PTPTSLR） *864*
33.7.36 以太网 PTP 时间戳高位更新寄存器（ETH_PTPTSHUR） *864*
33.7.37 以太网 PTP 时间戳低位更新寄存器（ETH_ PTPTSLUR） *864*
33.7.38 以太网 PTP 时间戳加数寄存器（ETH_PTPTSAR） *865*
33.7.39 以太网 PTP 时间戳目标时间高位寄存器（ETH_PTPTTHR） *865*
33.7.40 以太网 PTP 时间戳目标时间低位寄存器（ETH_PTPTTLR） *865*
33.7.41 以太网 DMA 总线模式寄存器（ETH_DMABMR） *866*
33.7.42 以太网 DMA 发送查询请求寄存器（ETH_DMATPDR） *867*
33.7.43 以太网 DMA 接收查询请求寄存器（ETH_DMARPDR） *868*
33.7.44 以太网 DMA 接收描述符列表地址寄存器（ETH_DMARDLAR） *868*
33.7.45 以太网 DMA 发送描述符列表地址寄存器（ETH_DMATDLAR） *868*
33.7.46 以太网 DMA 状态寄存器（ETH_DMASR） *869*
33.7.47 以太网 DMA 工作模式寄存器（ETH_DMAOMR） *871*
33.7.48 以太网 DMA 中断使能寄存器（ETH_DMAIER） *873*
33.7.49 以太网 DMA 丢失帧和缓存溢出计数寄存器（ETH_DMAMFBOCR） *875*
33.7.50 以太网 DMA 当前发送描述符寄存器（ETH_DMACHTDR） *875*
33.7.51 以太网 DMA 当前接收描述符寄存器（ETH_DMACHRDR） *876*
33.7.52 以太网 DMA 当前发送缓存地址寄存器（ETH_DMACHTBAR） *876*
33.7.53 以太网 DMA 当前接收缓存地址寄存器（ETH_DMACHRBAR） *876*
34. CAN 总线控制器 **877**
34.1 简介 877
34.1.1 主要特性 *877*
34.1.2 模块框图 *878*
34.2 FDCAN 功能描述 879
34.2.1 工作模式 *879*

34.2.1.1 软件初始化 879
34.2.1.2 正常工作 880
34.2.1.3 CAN FD 操作 881
34.2.1.4 收发器延迟补偿 882
34.2.1.5 受限工作模式 883
34.2.1.6 总线监听模式 884
34.2.1.7 禁止自动重发（DAR）模式 884
34.2.1.8 掉电（休眠模式） 885
34.2.1.9 测试模式 885
34.2.1.10 外部回环模式 885
34.2.1.11 内部回环模式 886
34.2.2 时间戳生成 *886*
34.2.3 超时计数器 *886*
34.2.4 接收处理 *887*
34.2.4.1 接收过滤 887
34.2.4.2 ID 范围过滤器 888
34.2.4.3 专用 ID 过滤器 888
34.2.4.4 ID 位屏蔽过滤器 888
34.2.4.5 标准消息 ID 过滤 888
34.2.4.6 扩展消息 ID 过滤 889
34.2.4.7 接收 FIFO 890
34.2.4.8 接收 FIFO 阻塞模式 891
34.2.4.9 接收 FIFO 覆盖模式 891
34.2.4.10 专用接收缓冲区 892
34.2.4.11 接收缓冲区处理 893
34.2.5 发送处理 *893*
34.2.5.1 发送暂停 893
34.2.5.2 专用发送缓冲区 894
34.2.5.3 发送 FIFO 894
34.2.5.4 发送队列 895
34.2.5.5 混合使用专用发送缓冲区和发送 FIFO 895
34.2.5.6 混合使用专用发送缓冲区和发送队列 896
34.2.5.7 发送取消 896
34.2.5.8 发送事件处理 896
34.2.6 FIFO 确认处理 *897*

34.2.7 消息 RAM *897*
34.2.7.1 消息 RAM 配置 897
34.2.7.2 接收缓冲区和 FIFO 元素 898
34.2.7.3 发送缓冲区元素 900
34.2.7.4 发送事件 FIFO 元素 901
34.2.7.5 标准消息 ID 过滤器元素 903
34.2.7.6 扩展消息 ID 过滤器元素 904
34.3 寄存器描述 905
34.3.1 FDCAN 字节序寄存器（FDCAN_ENDN），Address=0x04 ..... *905*
34.3.2 FDCAN 数据位时间和预分频寄存器（FDCAN_DBTP），Address=0x0C *905*
34.3.3 FDCAN 测试寄存器（FDCAN_TEST），Address=0x10 ..... *906*
34.3.4 FDCAN RAM 看门狗寄存器（FDCAN_RWD），Address=0x14 ..... *908*
34.3.5 FDCAN CC 控制寄存器（FDCAN_CCCR），Address=0x18 ..... *908*
34.3.6 FDCAN 标称位时间和预分频寄存器（FDCAN_NBTP），Address=0x1C *910*
34.3.7 FDCAN 时间戳计数器配置寄存器（FDCAN_TSCC），Address=0x20 ..... *911*
34.3.8 FDCAN 时间戳计数器值寄存器（FDCAN_TSCV），Address=0x24 ..... *911*
34.3.9 FDCAN 超时计数器配置寄存器（FDCAN_TOCC），Address=0x28 ..... *912*
34.3.10 FDCAN 超时计数器值寄存器（FDCAN_TOCV），Address=0x2C *912*
34.3.11 FDCAN 错误计数器寄存器（FDCAN_ECR），Address=0x40 ..... *913*
34.3.12 FDCAN 协议状态寄存器（FDCAN_PSR），Address=0x44 ..... *913*
34.3.13 FDCAN 发送器延迟补偿寄存器（FDCAN_TDCR），Address=0x48 ..... *916*
34.3.14 FDCAN 中断寄存器（FDCAN_IR），Address=0x50 ..... *917*
34.3.15 FDCAN 中断使能寄存器（FDCAN_IE），Address=0x54 ..... *919*
34.3.16 FDCAN 中断线选择寄存器（FDCAN_ILS），Address=0x58 ..... *921*
34.3.17 FDCAN 中断线使能寄存器（FDCAN_ILE），Address=0x5C *924*
34.3.18 FDCAN 全局过滤器配置寄存器（FDCAN_GFC），Address=0x80 ..... *924*
34.3.19 FDCAN 标准 ID 过滤器配置寄存器（FDCAN_SIDFC），Address=0x84 ..... *925*
34.3.20 FDCAN 扩展 ID 过滤器配置寄存器（FDCAN_XIDFC），Address=0x88 ..... *925*
34.3.21 FDCAN 扩展 ID 与掩码寄存器（FDCAN_XIDAM），Address=0x90 ..... *926*
34.3.22 FDCAN 高优先级消息状态寄存器（FDCAN_HPMS），Address=0x94 ..... *926*
34.3.23 FDCAN 新数据 1 寄存器（FDCAN_NDAT1），Address=0x98 ..... *927*
34.3.24 FDCAN 新数据 2 寄存器（FDCAN_NDAT2），Address=0x9C *927*
34.3.25 FDCAN 接收 FIFO0 配置寄存器（FDCAN_RXF0C），Address=0xA0 ..... *928*
34.3.26 FDCAN 接收 FIFO0 状态寄存器（FDCAN_RXF0S），Address=0xA4 ..... *928*
34.3.27 FDCAN 接收 FIFO0 确认寄存器（FDCAN_RXF0A），Address=0xA8 ..... *929*

34.3.28 FDCAN 接收缓冲区配置寄存器（FDCAN_RXBC），Address=0xAC 929
34.3.29 FDCAN 接收 FIFO1 配置寄存器（FDCAN_RXF1C），Address=0xB0 930
34.3.30 FDCAN 接收 FIFO1 状态寄存器（FDCAN_RXF1S），Address=0xB4 931
34.3.31 FDCAN 接收 FIFO1 确认寄存器（FDCAN_RXF1A），Address=0xB8 931
34.3.32 FDCAN 接收缓冲区和 FIFO 元素大小配置寄存器（FDCAN_RXESC），Address=0xBC 932
34.3.33 FDCAN 发送缓冲区配置寄存器（FDCAN_TXBC），Address=0xC0 933
34.3.34 FDCAN 发送 FIFO/队列状态寄存器（FDCAN_TXFQS），Address=0xC4 933
34.3.35 FDCAN 发送缓冲区元素大小配置寄存器（FDCAN_TXESC），Address=0xC8 934
34.3.36 FDCAN 发送缓冲区请求挂起寄存器（FDCAN_TXBRP），Address=0xCC 935
34.3.37 FDCAN 发送缓冲区添加请求寄存器（FDCAN_TXBAR），Address=0xD0 935
34.3.38 FDCAN 发送缓冲区取消请求寄存器（FDCAN_TXBCR），Address=0xD4 936
34.3.39 FDCAN 发送缓冲区发送已发生寄存器（FDCAN_TXBTO），Address=0xD8 936
34.3.40 FDCAN 发送缓冲区取消已完成寄存器（FDCAN_TXBCF），Address=0xDC 937
34.3.41 FDCAN 发送缓冲区发送中断使能寄存器（FDCAN_TXBTIE），Address=0xE0 937
34.3.42 FDCAN 发送缓冲区取消已完成中断使能寄存器（FDCAN_TXBCIE），Address=0xE4 937
34.3.43 FDCAN 发送事件 FIFO 配置寄存器（FDCAN_TXEFC），Address=0xF0 938
34.3.44 FDCAN 发送事件 FIFO 状态寄存器（FDCAN_TXEFS），Address=0xF4 938
34.3.45 FDCAN 发送事件 FIFO 确认寄存器（FDCAN_TXEFA），Address=0xF8 939
35. 串行外设接口/片上音频接口(SPI/I²S) 940
35.1 概述 940
35.1.1 SPI 特征 940
35.1.2 I²S 特征 941
35.2 SPI 功能描述 941
35.2.1 单主机和单从机通信 941
35.2.1.1 全双工通信 941
35.2.1.2 半双工通信 942
35.2.2 多从机通信 943
35.2.3 多主机通信 944
35.2.4 从机选择(NSS)引脚管理 945
35.2.5 通讯格式 946
35.2.5.1 时钟极性和相位 946
35.2.5.2 数据帧格式 947
35.2.6 SPI 配置 947
35.2.7 SPI 使能流程 948

35.2.8 数据传输和接收流程 *948*
35.2.8.1 接收 FIFO 和发送 FIFO 948
35.2.8.2 处理序列 948
35.2.8.3 关闭 SPI 的流程 949
35.2.8.4 数据包 949
35.2.8.5 通信时序 950
35.2.9 状态位 *951*
35.2.9.1 TX FIFO 空标志(TXE) 951
35.2.9.2 接收缓冲非空标志(RXNE) 951
35.2.9.3 忙标志(BSY) 951
35.2.10 错误标志 *952*
35.2.10.1 主模式失效(MODF) 952
35.2.10.2 溢出错误（OVR） 952
35.2.11 SPI 中断 *952*
35.2.12 CRC 计算 *952*
35.2.12.1 CRC 标准 953
35.2.12.2 CPU 管理的 CRC 传输 953
35.2.12.3 DMA 下的 CRC 传输 953
35.2.13 TI 模式 *953*
35.3 I²S 功能描述 954
35.3.1 简介 *954*
35.3.2 音频协议 *954*
35.3.2.1 I²S 飞利浦协议 955
35.3.2.2 MSB 对齐标准 956
35.3.2.3 LSB 对齐标准 957
35.3.2.4 PCM 标准 958
35.3.3 时钟 *959*
35.3.4 发送与接收 *960*
35.3.4.1 主模式 960
35.3.4.2 从模式 960
35.3.5 标志位 *961*
35.3.5.1 状态标志 961
35.3.5.2 错误标志 962
35.3.6 DMA 功能 *962*
35.3.7 中断 *962*

35.4 寄存器描述 963
35.4.1 SPI_CR1 寄存器 *963*
35.4.2 SPI_CR2 寄存器 *965*
35.4.3 SPI_SR 寄存器 *966*
35.4.4 SPI_DR 寄存器 *967*
35.4.5 SPI_CRCPR 寄存器 *967*
35.4.6 SPI_RXCRCR 寄存器 *968*
35.4.7 SPI_TXCRCR 寄存器 *968*
35.4.8 SPI_I2S_CFGR 寄存器 *969*
35.4.9 SPI_I2SPR 寄存器 *970*
36. I²C 接口 972
36.1 简介 972
36.2 主要特征 972
36.3 I²C 功能描述 974
36.3.1 简介 *974*
36.3.2 时钟 *974*
36.3.3 包错误校验（PEC） *974*
36.3.4 I²C 的从模式 *975*
36.3.4.1 从机发送模式 976
36.3.4.2 从机接收模式 976
36.3.4.3 关闭通信 977
36.3.5 I²C 的主模式 *977*
36.3.5.1 主机产生时钟 978
36.3.5.2 起始条件 978
36.3.5.3 从机地址发送 978
36.3.5.4 主机发送模式 979
36.3.5.5 主机接收模式 980
36.3.6 错误标志 *983*
36.3.7 DMA 功能 *984*
36.3.8 支持从 STOP 模式唤醒 *985*
36.3.9 中断 *985*
36.3.10 系统管理总线 SMbus *985*
36.3.10.1 地址解析协议(ARP) 986
36.3.10.2 SMBus 提醒模式（ALERT） 986
36.3.10.3 超时错误（TIMEOUT） 986

36.3.10.4 使用 SMBus 模式的接口 987
36.3.10.5 总线空闲检测 987
36.3.10.6 PMbus 987
36.3.10.7 SMBus：I2C_TIMEOUTR 寄存器配置示例 987
36.4 寄存器描述 988
36.4.1 I²C 控制寄存器 1 (I2C_CR1) *988*
36.4.2 I²C 控制寄存器 2 (I2C_CR2) *991*
36.4.3 I²C 自身地址寄存器 1 (I2C_OAR1) *992*
36.4.4 I²C 自身地址寄存器 2 (I2C_OAR2) *993*
36.4.5 I²C 数据寄存器(I2C_DR) *993*
36.4.6 I²C 状态寄存器 1 (I2C_SR1) *994*
36.4.7 I²C 状态寄存器 2 (I2C_SR2) *997*
36.4.8 I²C 时钟控制寄存器(I2C_CCR) *998*
36.4.9 I2C 上升时间寄存器 (I2C_TRISE) *999*
36.4.10 I2C 超时寄存器(I2C_TIMEOUTR) *1000*
37. 通用同步收发器(USART) **1002**
37.1 简介 1002
37.2 主要特征 1002
37.2.1 USART 拓展功能 *1002*
37.2.2 USART 功能框图 *1003*
37.3 USART 功能描述 1004
37.3.1 传输 pin 脚 *1004*
37.3.1.1 USART 特征 1004
37.3.1.2 USART 硬件流控 1004
37.3.1.3 同步模式和智能卡模式 1004
37.3.2 USART 特性描述 *1004*
37.3.3 USART FIFO 和阈值 *1005*
37.3.4 发送器 *1006*
37.3.4.1 字符发送 1006
37.3.4.2 配置停止位 1006
37.3.4.3 单字节通讯 1007
37.3.4.4 中断字符 1008
37.3.4.5 空闲字符 1008
37.3.5 接收器 *1008*
37.3.5.1 起始位检测 1008

37.3.5.2 字符接收 1009
37.3.5.3 中断字符 1010
37.3.5.4 空闲字符 1010
37.3.5.5 上溢错误 1010
37.3.5.6 噪声错误 1010
37.3.5.7 帧错误 1011
37.3.5.8 在接收时配置停止位 1012
37.3.6 分数波特率的产生 1012
37.3.7 USART 接收器容忍时钟的变化 1013
37.3.8 USART 自动波特率检测 1013
37.3.9 多处理器通信 1014
37.3.9.1 空闲总线检测(WAKE=0) 1014
37.3.9.2 地址标记(address mark)检测(WAKE=1)（4 位或者 7 位地址检测） 1015
37.3.10 USART Modbus 通信 1015
37.3.11 校验控制 1016
37.3.12 LIN(局域互联网)模式 1016
37.3.12.1 发送 1016
37.3.12.2 接收 1016
37.3.13 同步模式 1018
37.3.13.1 主模式 1018
37.3.14 单线半双工通信 1019
37.3.15 USART 接收超时 1019
37.3.16 智能卡 1019
37.3.16.1 T = 0（character mode) 1020
37.3.16.2 T = 1（block mode) 1021
37.3.17 IrDA SIR ENDEC 功能模块 1022
37.3.17.1 IrDA 正常模式 1022
37.3.17.2 IrDA 低功耗模式 1022
37.3.18 利用 DMA 连续通信 1023
37.3.19 硬件流控制 1025
37.3.19.1 RS232 硬件流控 1025
37.3.19.2 RS485 硬件流控 1027
37.3.20 中断请求 1027
37.4 寄存器描述 1027
37.4.1 状态寄存器（SR） 1027

37.4.2 发送数据寄存器（DR） 1031
37.4.3 波特率寄存器（BRR） 1031
37.4.4 控制寄存器1（CR1） 1032
37.4.5 控制寄存器2（CR2） 1035
37.4.6 控制寄存器3（CR3） 1037
37.4.7 保护时间和预分频（GTPR） 1039
37.4.8 接收超时寄存器（RTOR） 1040
38. 通用异步收发器(UART) 1041
38.1 简介 1041
38.2 主要特征 1041
38.3 UART 功能描述 1041
38.3.1 UART（RS232）串行协议 1041
38.3.2 9 位数据传输 1042
38.3.3 波特率 1045
38.3.4 UART 接收器容忍时钟的变化 1046
38.3.5 利用 DMA 连续通讯 1046
38.3.6 中断 1047
38.4 寄存器描述 1048
38.4.1 UART DR, Address=0X00(DR) 1048
38.4.2 UART BRR, Address=0X04(BRR) 1048
38.4.3 UART SR, Address=0X10(SR) 1048
38.4.4 UART CR1, Address=0X14(CR1) 1050
38.4.5 UART CR2, Address=0X18(CR2) 1052
38.4.6 UART CR3, Address=0X1C(CR3) 1052
38.4.7 UART RAR, Address=0X20(RAR) 1053
38.4.8 UART TAR, Address=0X24(TAR) 1054
38.4.9 UART BRRF, Address=0X28(BRRF) 1054
39. 低功耗通用异步收发器（LPUART） 1055
39.1 LPUART 主要特性 1055
39.2 LPUART 功能描述 1056
39.2.1 LPUART 框图 1056
39.2.2 LPUART 信号 1056
39.2.3 LPUART 字符描述 1057
39.2.4 LPUART 发送 1057

39.2.4.1 起始位检测 1058
39.2.4.2 数据字符接收 1058
39.2.4.3 特殊帧及错误处理 1059
39.2.4.4 时钟源选择 1059
39.2.4.5 帧错误 1060
39.2.4.6 配置停止位个数 1060
39.2.5 LPUART 波特率产生 1060
39.2.6 LPUART 接收容忍度 1061
39.2.7 LPUART 多处理器通信 1061
39.2.7.1 空闲总线检测 1062
39.2.7.2 4 位/7 位地址标志检测 1062
39.2.8 LPUART 奇偶校验 1063
39.2.8.1 偶校验 1063
39.2.8.2 奇校验 1063
39.2.8.3 接收校验 1063
39.2.8.4 发送校验位 1063
39.2.9 LPUART 单线半双工通信 1063
39.2.10 DMA 连续通信 1063
39.2.10.1 利用 DMA 发送 1063
39.2.10.2 利用 DMA 接收 1064
39.2.10.3 错误标志和中断 1065
39.2.11 RS232 硬件流控制和 RS485 驱动器使能 1065
39.2.11.1 RS232 RTS 流控制 1066
39.2.11.2 RS232 CTS 流控制 1066
39.2.11.3 RS485 驱动使能 1067
39.2.12 LPUART 低功耗模式 1067
39.2.12.1 根据 WUS 产生中断唤醒请求 1067
39.2.12.2 从静默模式进入低功耗模式 1067
39.2.12.3 低功耗模式时内核时钟关闭 1067
39.3 LPUART 中断 1068
39.4 LPUART 寄存器 1069
39.4.1 LPUART 控制寄存器 1（LPUART_CR1） 1069
39.4.2 LPUART 控制寄存器 2（LPUART_CR2） 1071
39.4.3 LPUART 控制寄存器 3（LPUART_CR3） 1072
39.4.4 LPUART 波特率寄存器（LPUART_BRR） 1074

39.4.5 LPUART 请求寄存器（LPUART_RQR）..... 1075
39.4.6 LPUART 中断和状态寄存器（LPUART_ISR）..... 1075
39.4.7 LPUART 中断标志清零寄存器（LPUART_ICR）..... 1078
39.4.8 LPUART 接收数据寄存器（LPUART_RDR）..... 1078
39.4.9 LPUART 发送数据寄存器（LPUART_TDR）..... 1079
39.4.10 LPUART 预分频器寄存器（LPUART_PRESC）..... 1079
40. 时钟校准控制器（CTC）..... 1080
40.1 简介..... 1080
40.2 主要特征..... 1080
40.2.1 模块框图..... 1080
40.3 功能描述..... 1081
40.3.1 REF 同步脉冲发生器..... 1081
40.3.2 CTC 校准计数器..... 1081
40.3.3 频率评估和自动校准过程..... 1082
40.3.4 软件编程指南..... 1083
40.4 寄存器描述（基址 0x4000_C800）..... 1083
40.4.1 CTC 控制寄存器 0(CTC_CTL0)..... 1083
40.4.2 CTC 控制寄存器 1(CTC_CTL1)..... 1084
40.4.3 CTC 状态寄存器（CTC_SR）..... 1085
40.4.4 CTC 中断清除寄存器（CTC_INTC）..... 1087
41. 随机数生成器(RNG)..... 1088
41.1 概述..... 1088
41.1.1 主要特性..... 1088
41.1.2 顶层结构图..... 1088
41.2 功能描述..... 1089
41.2.1 随机数的产生..... 1089
41.2.2 RNG 的软件操作..... 1089
41.2.3 RNG 的随机源..... 1089
41.3 寄存器描述..... 1089
41.3.1 RNG 控制寄存器（RNG_CR）..... 1090
41.3.2 RNG 线性反馈移位寄存器(RNG_LFSR)..... 1090
41.3.3 RNG 状态寄存器(RNG_SR)..... 1090
41.3.4 RNG 数据寄存器(RNG_DR)..... 1091
42. AES 硬件加速器(AES)..... 1092

42.1 简介 1092
42.1.1 主要特征 1092
42.1.2 模块框图 1092
42.2 功能说明 1093
42.2.1 ECB Mode 1093
42.2.2 CBC Feedback Mode 1093
42.2.3 CTR Mode 1094
42.2.4 CMAC Authentication Mode 1095
42.2.5 GCM Operation 1095
42.2.6 CCM Operation 1097
42.3 寄存器描述 1098
42.3.1 信息控制寄存器(AES_MSG_CFG) 1099
42.3.2 密钥控制寄存器(AES_CTX_CFG) 1100
42.3.3 信息长度寄存器(AES_MSG_TOTAL_BYTES) 1100
42.3.4 附加信息长度寄存器（AES_MSG_AAD_BYTES） 1101
42.3.5 GCM 模式附加信息长度寄存器（AES_GCM_AAD_INFO） 1101
42.3.6 密钥寄存器 0（AES_CTX_KEY0） 1101
42.3.7 密钥寄存器 1（AES_CTX_KEY1） 1102
42.3.8 密钥寄存器 2（AES_CTX_KEY2） 1102
42.3.9 密钥寄存器 3（AES_CTX_KEY3） 1102
42.3.10 密钥寄存器 4（AES_CTX_KEY4） 1102
42.3.11 密钥寄存器 5（AES_CTX_KEY5） 1103
42.3.12 密钥寄存器 6（AES_CTX_KEY6） 1103
42.3.13 密钥寄存器 7（AES_CTX_KEY7） 1103
42.3.14 CBC 模式密钥寄存器 0（AES_CTX_CBC_KEY0） 1103
42.3.15 CBC 模式密钥寄存器 1（AES_CTX_CBC_KEY1） 1104
42.3.16 CBC 模式密钥寄存器 2（AES_CTX_CBC_KEY2） 1104
42.3.17 CBC 模式密钥寄存器 3（AES_CTX_CBC_KEY3） 1104
42.3.18 CTR 模式算子寄存器 0（AES_CTX_CTR0） 1104
42.3.19 CTR 模式算子寄存器 1（AES_CTX_CTR1） 1105
42.3.20 CTR 模式算子寄存器 2（AES_CTX_CTR2） 1105
42.3.21 CTR 模式算子寄存器 3（AES_CTX_CTR3） 1105
42.3.22 初始化向量寄存器 0（AES_CTX_IV0） 1105
42.3.23 初始化向量寄存器 1（AES_CTX_IV1） 1106
42.3.24 初始化向量寄存器 2（AES_CTX_IV2） 1106

42.3.25 初始化向量寄存器 3（AES_CTX_IV3）..... 1106
42.3.26 消息验证码寄存器 0（AES_CTX_MAC0）..... 1106
42.3.27 消息验证码寄存器 1（AES_CTX_MAC1）..... 1107
42.3.28 消息验证码寄存器 2（AES_CTX_MAC2）..... 1107
42.3.29 消息验证码寄存器 3（AES_CTX_MAC3）..... 1107
42.3.30 输入 FIFO（AES_INGRESS_FIFO）..... 1107
42.3.31 DMA 输入爆发传输设置寄存器（AES_ING_DBCFG）..... 1108
42.3.32 DMA 输出长度寄存器（AES_DMA_ENG_LEN）..... 1108
42.3.33 DMA 输出爆发传输设置寄存器（AES_ENG_DBCFG）..... 1108
42.3.34 DMA 输入长度寄存器（AES_DMA_ING_LEN）..... 1109
42.3.35 输入 FIFO 状态寄存器（AES_INGF_STATUS）..... 1109
42.3.36 输出 FIFO（AES_ENGRESS_FIFO）..... 1110
42.3.37 输出 FIFO 状态寄存器（AES_ENGFIFO_STATUS）..... 1110
42.3.38 DMA 输入完成状态寄存器（AES_ING_DMA_DONE）..... 1110
42.3.39 DMA 输出完成状态寄存器（AES_ENG_DMA_DONE）..... 1111
42.3.40 完成状态寄存器（AES_DONE_STATUS）..... 1111

43. CORDIC 协处理器(CORDIC)..... 1112

43.1 概述..... 1112
43.1.1 主要特性..... 1112
43.2 功能描述..... 1112
43.2.1 概述..... 1112
43.2.2 CORDIC 函数..... 1112
43.2.2.1 Cosine..... 1113
43.2.2.2 Sine..... 1113
43.2.2.3 Phase..... 1114
43.2.2.4 Modulus..... 1114
43.2.2.5 Arctangent..... 1114
43.2.2.6 Hyperbolic cosine..... 1115
43.2.2.7 Hyperbolic sine..... 1115
43.2.2.8 Hyperbolic arctangent..... 1116
43.2.2.9 Natural logarithm..... 1116
43.2.2.10 Square root..... 1117
43.2.3 定点表示法..... 1117
43.2.4 缩放因子..... 1117
43.2.5 精度..... 1118

43.2.6 Zero-overhead mode 1119
43.2.7 Polling mode 1120
43.2.8 Interrupt mode 1120
43.2.9 DMA mode 1120
43.3 寄存器描述 1121
43.3.1 CORDIC 控制/状态寄存器（CORDIC_CSR) 1121
43.3.2 CORDIC 参数寄存器（CORDIC_WDATA) 1123
43.3.3 CORDIC 结果寄存器（CORDIC_RDATA） 1124
44. 6800/8080 控制器（LCDC） 1125
44.1 概述 1125
44.2 主要特性 1125
44.3 功能描述 1125
44.3.1 使能 LCDC 1125
44.3.2 工作模式 1125
44.3.3 发送 LCD 数据 1125
44.3.4 发送 LCD 指令 1126
44.3.5 从 LCD 读取指令 1126
44.3.6 LCDC 时序 1126
44.3.7 DMA 请求 1128
44.3.8 LCDC 忙机制 1128
44.4 寄存器描述 1129
44.4.1 LCDC 控制与状态寄存器(LCDC_CSR) 1129
44.4.2 LCD0 数据寄存器(LCD0_DR) 1130
44.4.3 LCD0 命令寄存器(LCD0_CMDR) 1130
44.4.4 LCD0 写时序配置寄存器(LCD0_WCFGR) 1130
44.4.5 LCD0 读时序配置寄存器(LCD0_RCFGR) 1131
44.4.6 LCD1 数据寄存器(LCD1_DR) 1131
44.4.7 LCD1 命令寄存器(LCD1_CMDR) 1132
44.4.8 LCD1 写时序配置寄存器(LCD1_WCFGR) 1132
44.4.9 LCD1 读时序配置寄存器(LCD1_RCFGR) 1133
44.4.10 LCD2 数据寄存器(LCD2_DR) 1133
44.4.11 LCD2 命令寄存器(LCD2_CMDR) 1134
44.4.12 LCD2 写时序配置寄存器(LCD2_WCFGR) 1134
44.4.13 LCD2 读时序配置寄存器(LCD2_RCFGR) 1135
44.4.14 LCD3 数据寄存器(LCD3_DR) 1135

44.4.15 LCD2 命令寄存器(LCD3_CMDR) 1136
44.4.16 LCD3 写时序配置寄存器(LCD3_WCFGR) 1136
44.4.17 LCD3 读时序配置寄存器(LCD3_RCFGR) 1137
45. MCU 调试接口 1138
45.1 简介 1138
45.2 主要特性 1138
45.3 功能描述 1138
45.3.1 支持调试低功耗模式 1138
45.4 寄存器描述 1139
45.4.1 ID 编码（DBGMCU_IDCODE） 1139
45.4.2 调试配置寄存器（DBGMCU_CR1） 1139
45.4.3 调试配置寄存器（DBGMCU_CR2） 1142
46. 更新历史 1144

1. 寄存器描述中使用的缩写列表

缩写	描述
Read/Write（RW）	软件能读写此位
Read-only（R）	软件只能读此位
Write-only（W）	软件只能写此位，读此位将返回复位值
Read/Clear Write0（RC_W0)	软件可以读此位，也可以通过写 0 清除此位，写 1 对此位无影响
Read/Clear Write1（RC_W1)	软件可以读此位，也可以通过写 1 清除此位，写 0 对此位无影响
Read/Clear Write（RC_W)	软件可以通过写入寄存器来读取和清除该位，写入该位的值并不重要
Read/Clear by read（RC_R)	软件可以读取这个位。读取此位会自动将其清除为 0，写入此位不会影响位值
Read/Set by Read（RS_R)	软件可以读取这个位。读取此位会自动将其设置为 1，写入此位不会影响位值
Read/Set（RS)	软件可以读此位，也可以设置此位为 1，写 0 对此位无影响
Toggle（T)	软件可以通过写入 1 来切换此位，写入 0 无效
保留（Res)	保留位，必须保持在重置值

2. 系统架构框图

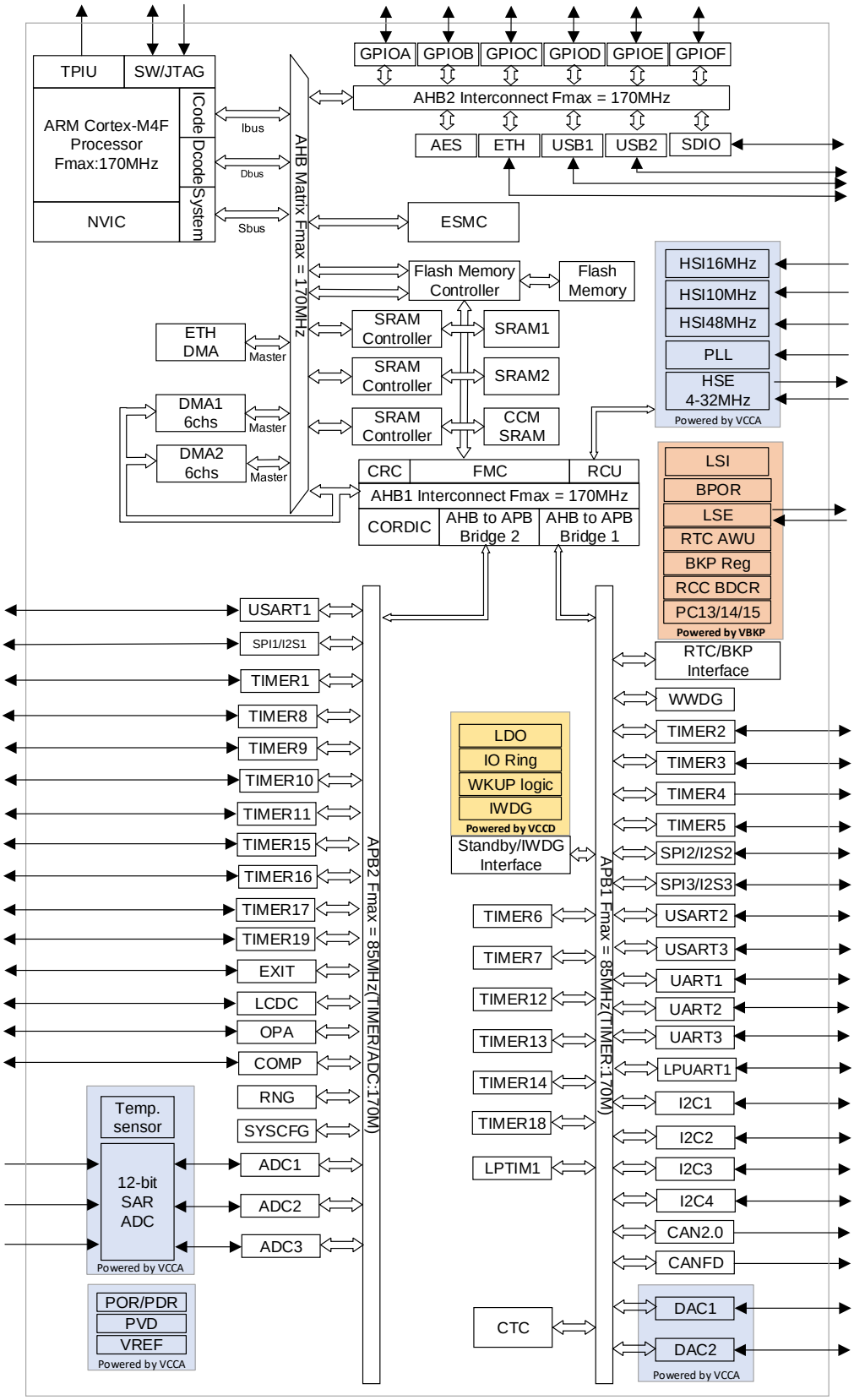


图 2-1 功能模块

3. 系统及存储器架构

PY32E407 系列器件是基于 Arm® Cortex®-M4F 处理器的 32 位通用微控制器。Arm® Cortex®-M4F 处理器包括三条 AHB 总线分别称为 I-CODE 总线、D-Code 总线和系统总线。Cortex®-M4F 处理器的所有存储访问，根据不同的目的和目标存储空间，都会在这三条总线上执行。存储器的组织采用了哈佛结构，预先定义的存储器映射和高达 4 GB 的存储空间，充分保证了系统的灵活性和可扩展性。

3.1 Arm® Cortex®-M4F 处理器简介

Cortex®-M4F 处理器是一个具有低中断延迟时间和低成本调试特性的 32 位处理器。高集成度和增强的特性使 Cortex™-M4F 处理器适合于那些需要高性能和低功耗微控制器的市场领域。Cortex®-M4F 处理器基于 Armv7 架构，并且支持一个强大且可扩展的指令集，包括通用数据处理 I/O 控制任务、增强的数据处理位域操作、DSP(数字信号处理)和浮点运算指令。下面列出由 Cortex®-M4F 提供的一些系统外设：

- 内部总线矩阵，用于实现 I-Code 总线、D-Code 总线、系统总线、专用总线(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（NVIC）
- 闪存地址重载及断点单元（FPB）
- 数据观测点及跟踪单元（DWT）
- 指令跟踪宏单元（ITM）
- 嵌入式跟踪宏单元（ETM）
- 串行线和 JTAG 调试接口（SWJ-DP）
- 跟踪端口接口单元（TPIU）
- 内存保护单元（MPU）
- 浮点运算单元（FPU）

3.2 系统架构

系统采用 32 位多层 AHB 总线矩阵，可以保证多主机和多从机之间的并行通信：

- 六条主控总线：
  - Cortex™-M4F 内核 I -Code 总线、D-Code 总线和 系统总线
  - DMA1 总线
  - DMA2 总线
  - ETH DMA 总线
- 八条被控总线：
  - 内部 Flash I-Code 总线
  - 内部 Flash D-Code 总线
  - 内部 SRAM1 总线
  - 内部 SRAM2 总线
  - 内部 CCM SRAM 总线
  - AHB1 外设总线（包括 AHB 到 APB 的总线桥和 APB 外设）
  - AHB2 外设总线

— ESMC

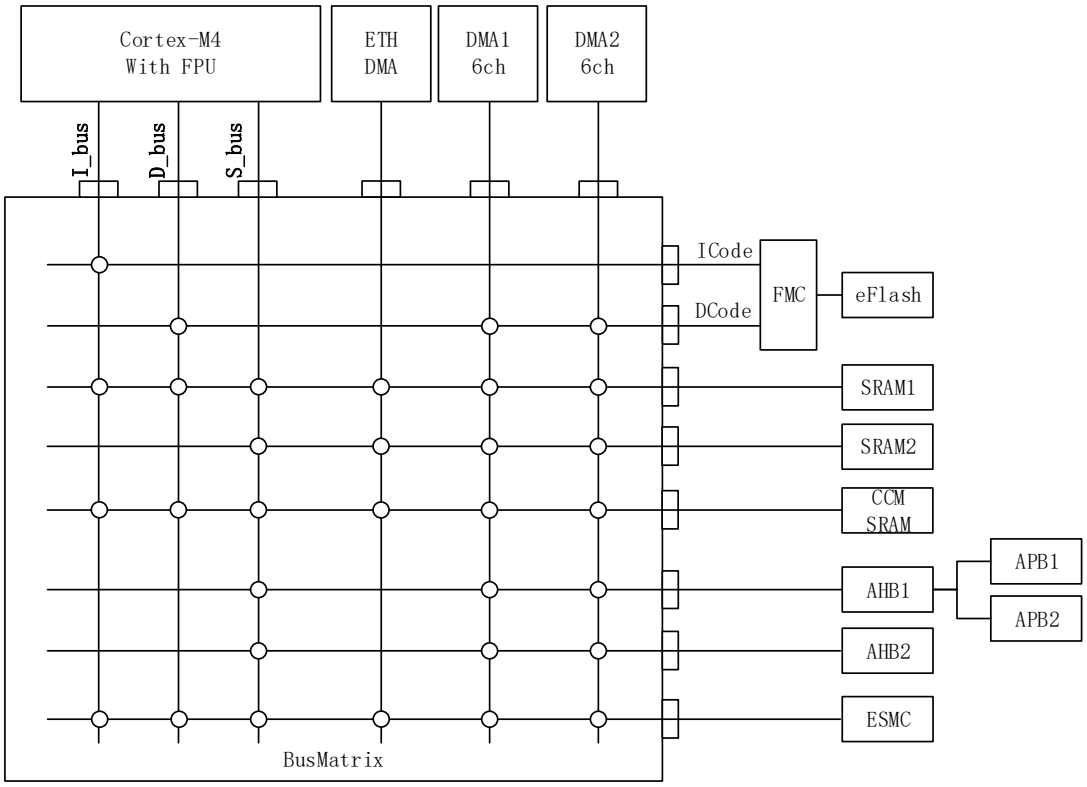


图 3-1 系统架构图

3.2.1 I_bus

此总线用于将 Cortex™-M4F 内核的指令总线连接到总线矩阵。内核通过此总线获取指令。此总线访问的对象是包含代码的存储器（内部 Flash/SRAM1/CCM SRAM 或通过 ESMC 的外部存储器）。

3.2.2 D_bus

此总线用于将 Cortex™-M4F 数据总线连接到总线矩阵。内核通过此总线进行立即数加载和调试访问。此总线访问的对象是包含代码或数据的存储器（内部 Flash/SRAM1/CCM SRAM 或通过 ESMC 的外部存储器）。

3.2.3 S_bus

此总线用于将 Cortex™-M4F 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据。此总线访问的对象是 144 KB 的内部 SRAM1/SRAM2/CCM SRAM、包括 AHB 外设在内的 APB1 外设、APB2 外设以及通过 ESMC 的外部存储器。

3.2.4 DMA1/DMA2 BUS

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线访问的对象是 144KB 的内部 SRAM1、SRAM2、CCM SRAM、包括 AHB 外设在内的 APB1 外设、APB2 外设以及通过 ESMC 的外部存储器。

3.2.5 ETH DMA BUS

此总线用于将 ETH DMA 存储器总线主接口连接到总线矩阵。ETH DMA 通过此总线访问的对象是 144 KB 的内部 SRAM1、SRAM2、CCM SRAM 以及通过 ESMC 的外部存储器。

3.2.6 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。系统包含 6 个主设备 Master，分别为 ARM® Cortex™-M4F 的 I_BUS、D_BUS、S_BUS、DMA1、DMA2 及 ETH DMA；及 8 个从设备 Slave，分别为内部的 Flash 存储器、内部的 SRAM1、SRAM2、CCM SRAM、AHB1 外设（AHB2APB 桥 1 对应的外设和 AHB2APB 桥 2 对应的外设）、AHB2 外设、ESMC。

3.2.7 AHB/APB 总线桥

借助两个 AHB/APB 总线桥 ，APB1 和 APB2 可在 AHB 总线与两个 APB 总线之间实现完全同步的连接，从而灵活选择外设频率。

每次芯片复位后，所有外设时钟都被关闭（SRAM 和 Flash 接口除外）。使用外设前，必须在 RCC_AHBxENR 或 RCC_APBxENR 寄存器中使能其时钟。

有关 AHB 和 APB 外设地址映射信息，参考 3.3.2。

3.3 存储器组织架构

3.3.1 简介

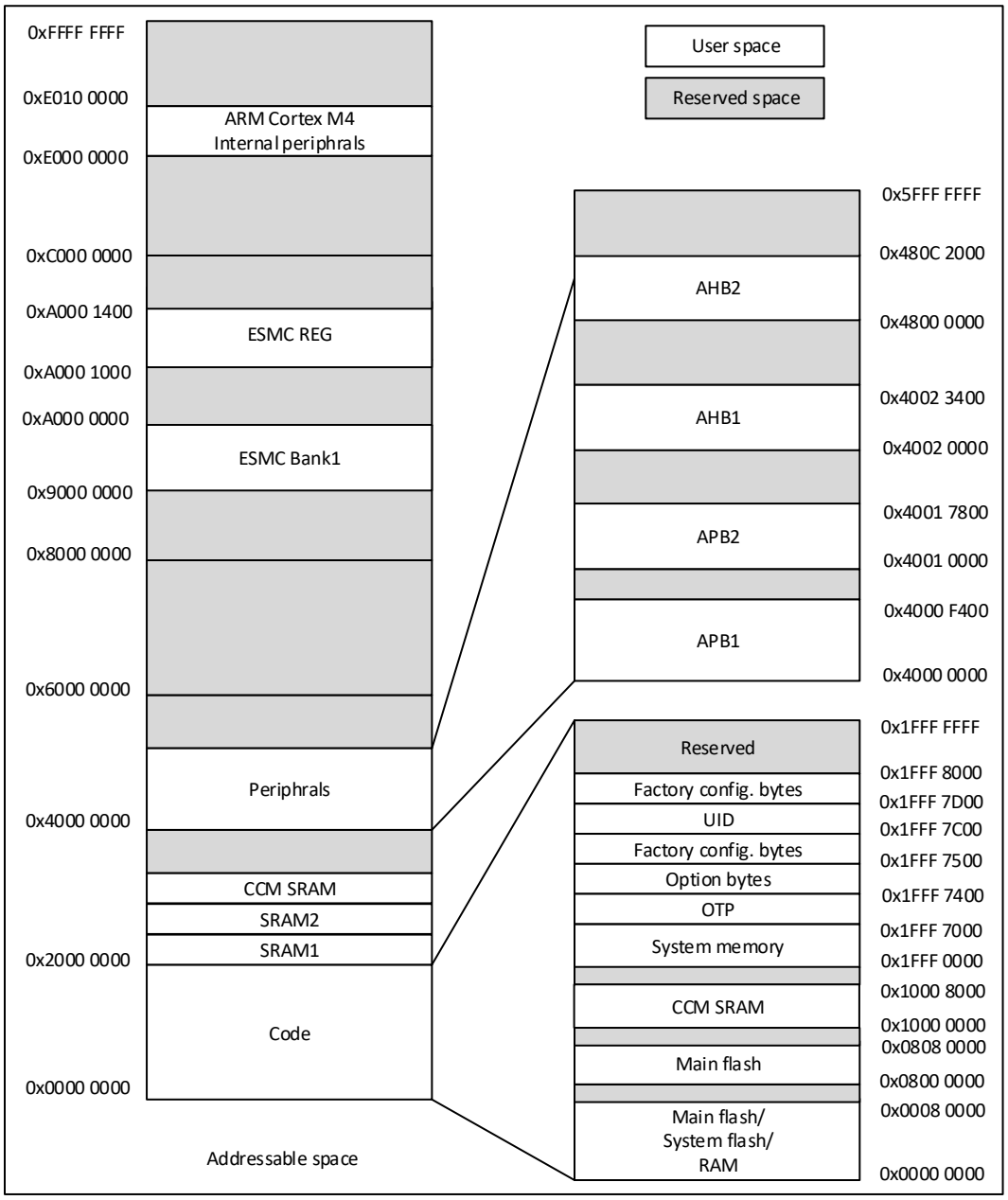
Arm® Cortex®-M4F 处理器采用哈佛结构，可以使用相互独立的总线来读取指令和加载/存储数据。指令代码和数据都位于相同的存储器地址空间，但在不同的地址范围。程序存储器，数据存储器，寄存器和 I/O 端口都在同一个线性的 4 GB 的地址空间之内。

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

有关外设寄存器映射的详细信息，请参见相关章节。

可寻址的存储空间分为 8 个主要块，每个块为 512 MB。

3.3.2 存储器映射


0xFFFF FFFF
0xE010 0000
ARM Cortex M4
Internal periphrals
0xE000 0000
0xC000 0000
0xA000 1400
ESMC REG
0xA000 1000
0xA000 0000
ESMC Bank1
0x9000 0000
0x8000 0000
0x6000 0000
Periphrals
0x4000 0000
CCM SRAM
SRAM2
SRAM1
0x2000 0000
Code
0x0000 0000
Addressable space
User space
Reserved space
0x5FFF FFFF
0x480C 2000
AHB2
0x4800 0000
0x4002 3400
AHB1
0x4002 0000
0x4001 7800
APB2
0x4001 0000
0x4000 F400
APB1
0x4000 0000
0x1FFF FFFF
Reserved
0x1FFF 8000
Factory config. bytes
0x1FFF 7D00
UID
0x1FFF 7C00
Factory config. bytes
0x1FFF 7500
Option bytes
0x1FFF 7400
OTP
0x1FFF 7000
System memory
0x1FFF 0000
0x1000 8000
CCM SRAM
0x1000 0000
0x0808 0000
Main flash
0x0800 0000
0x0008 0000
Main flash/
System flash/
RAM
0x0000 0000

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。详细的存储和寄存器空间映射参考下面的表格。

下表中给出了产品中可使用的外设及对应的地址边界。

表 3-1 存储器地址

Type	Boundary Address	Size	Memory Area	Description
SRAM	0x2002 4000 - 0x3FFF FFFF	511 MB	保留	1.CPU 读写该空间时产生 Response error,进而进入 HardFault 异常 2.DMA 访问时产生 TEIF 状态位
	0x2000 0000 - 0x2002 3FFF	144 KB	SRAM	如果硬件上电配置 SRAM 为 144 KB,则 SRAM 地址空间为 0x2000 0000 - 0x20023FFF
Code	0x1FFF 8000 - 0x1FFF FFFF	32KB	保留	-
	0x1FFF 7E00 - 0x1FFF 7FFF	512 Bytes	UID bytes	Unique ID
	0x1FFF 7C00 - 0x1FFF 7DFF	512 Bytes	Factory config. bytes	-
	0x1FFF 7A00 - 0x1FFF 7BFF	512 Bytes	HSI16 Trim	-
	0x1FFF 7800 - 0x1FFF 79FF	512 Bytes	Factory config. bytes	-
	0x1FFF 7600 - 0x1FFF 77FF	512 Bytes	Factory config. bytes	-
	0x1FFF 7400 - 0x1FFF 75FF	512 Bytes	Option bytes	芯片软硬件 option bytes 信息
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 KB	System memory	存放 Boot loader
	0x1008 0000 - 0x1FFE FFFF	256 MB	保留	-
	0x1000 0000 - 0x1000 7FFF	32 KB	CCM SRAM	-
	0x0808 0000 - 0x0FFF FFFF	127 MB	保留	-
	0x0800 0000 - 0x0807 FFFF	512 KB	Main flash memory	-
	0x0008 0000 - 0x07FF FFFF	127 MB	保留	1.CPU 读写该空间时产生 Response error，进而进入 Hard fault 异常； 2.DMA 访问时产生 TEIF 状态位；
	0x0000 0000 - 0x0007 FFFF	512 KB	根据 Boot 配置选择是： 1）Main flash memory 2）System memory 3）SRAM	-

1. 上述空间标注为**保留**的空间,无法进行写操作,读为 0，且产生 response error。

表 3-2 外设寄存器地址

总线	存储器起止地址	外设
AHB3	0xA000 1000 - 0xA000 13FF	ESMC
AHB2	0x480C 2000 - 0x5FFF FFFF	保留
	0x480C 0000 - 0x400C 1FFF	ETH
	0x4808 0000 - 0x480B FFFF	USB2 OTG FS
	0x4804 0000 - 0x4807 FFFF	USB1 OTG FS
	0x4800 2800 - 0x4803 FFFF	保留
	0x4800 2400 - 0x4800 27FF	AES

总线	存储器起止地址	外设
	0x4800 2000 - 0x4800 23FF	SDIO
	0x4800 1800 - 0x4800 1FFF	保留
	0x4800 1400 - 0x4800 17FF	GPIOF
	0x4800 1000 - 0x4800 13FF	GPIOE
	0x4800 0C00 - 0x4800 0FFF	GPIOD
	0x4800 0800 - 0x4800 0BFF	GPIOC
	0x4800 0400 - 0x4800 07FF	GPIOB
	0x4800 0000 - 0x4800 03FF	GPIOA
AHB1	0x4002 3400 - 0x4002 FFFF	保留
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2400 - 0x4002 2FFF	保留
	0x4002 2000 - 0x4002 23FF	FMC
	0x4002 1400 - 0x4002 1FFF	保留
	0x4002 1000 - 0x4002 13FF	RCC
	0x4002 0C00 - 0x4002 0FFF	CORDIC
	0x4002 0800 - 0x4002 0BFF	保留
	0x4002 0400 - 0x4002 07FF	DMA2
	0x4002 0000 - 0x4002 03FF	DMA1
APB2	0x4001 7800 - 0x4001 FFFF	保留
	0x4001 7400 - 0x4001 77FF	LCDC
	0x4001 7000 - 0x4001 73FF	OPA
	0x4001 6C00 - 0x4001 6FFF	COMP
	0x4001 6800 - 0x4001 6BFF	RNG
	0x4001 6400 - 0x4001 67FF	TIMER19
	0x4001 6000 - 0x4001 63FF	TIMER17
	0x4001 5C00 - 0x4001 5FFF	TIMER16
	0x4001 5800 - 0x4001 5BFF	TIMER15
	0x4001 5400 - 0x4001 57FF	TIMER11
	0x4001 5000 - 0x4001 53FF	TIMER10
	0x4001 4C00 - 0x4001 4FFF	TIMER9
	0x4001 4000 - 0x4001 4BFF	保留
	0x4001 3C00 - 0x4001 3FFF	ADC3
	0x4001 3800 - 0x4001 3BFF	USART1
	0x4001 3400 - 0x4001 37FF	TIMER8
	0x4001 3000 - 0x4001 33FF	SPI1/I^2S1
	0x4001 2C00 - 0x4001 2FFF	TIMER1
	0x4001 2800 - 0x4001 2BFF	ADC2
	0x4001 2400 - 0x4001 27FF	ADC1
	0x4001 0800 - 0x4001 23FF	保留
	0x4001 0400 - 0x4001 07FF	EXTI
	0x4001 0000 - 0x4001 03FF	SYSCFG
APB1	0x4000 F400 - 0x4000 FFFF	保留
	0x4000 E000 - 0x4000 F3FF	CAN2.0
	0x4000 CC00 - 0x4000 DFFF	CANFD
	0x4000 C800 - 0x4000 CBFF	CTC
	0x4000 B400 - 0x4000 C7FF	保留

总线	存储器起止地址	外设
	0x4000 AC00 - 0x4000 B3FF	CANMEM
	0x4000 8800 - 0x4000 ABFF	保留
	0x4000 8400 - 0x4000 87FF	I^2C4
	0x4000 8000 - 0x4000 83FF	LPUART1
	0x4000 7C00 - 0x4000 7FFF	LPTIM1
	0x4000 7800 - 0x4000 7BFF	I^2C3
	0x4000 7400 - 0x4000 77FF	DAC
	0x4000 7000 - 0x4000 73FF	PWR
	0x4000 6D00 - 0x4000 6FFF	保留
	0x4000 6C00 - 0x4000 6CFF	BKP
	0x4000 6000 - 0x4000 6BFF	保留
	0x4000 5C00 - 0x4000 5FFF	UART3
	0x4000 5800 - 0x4000 5BFF	I^2C2
	0x4000 5400 - 0x4000 57FF	I^2C1
	0x4000 5000 - 0x4000 53FF	UART2
	0x4000 4C00 - 0x4000 4FFF	UART1
	0x4000 4800 - 0x4000 4BFF	USART3
	0x4000 4400 - 0x4000 47FF	USART2
	0x4000 4000 - 0x4000 43FF	保留
	0x4000 3C00 - 0x4000 3FFF	SPI3/I^2S3
	0x4000 3800 - 0x4000 3BFF	SPI2/I^2S2
	0x4000 3400 - 0x4000 37FF	保留
	0x4000 3000 - 0x4000 33FF	IWDG
	0x4000 2C00 - 0x4000 2FFF	WWDG
	0x4000 2800 - 0x4000 2BFF	RTC
	0x4000 2400 - 0x4000 27FF	TIMER18
	0x4000 2000 - 0x4000 23FF	TIMER14
	0x4000 1C00 - 0x4000 1FFF	TIMER13
	0x4000 1800 - 0x4000 1BFF	TIMER12
	0x4000 1400 - 0x4000 17FF	TIMER7
	0x4000 1000 - 0x4000 13FF	TIMER6
	0x4000 0C00 - 0x4000 0FFF	TIMER5
	0x4000 0800 - 0x4000 0BFF	TIMER4
	0x4000 0400 - 0x4000 07FF	TIMER3
	0x4000 0000 - 0x4000 03FF	TIMER2

1. 上表 AHB 标注为**保留**的地址空间，无法写操作,读回为 0，且产生 Hard fault。

3.4 嵌入式 SRAM

PY32E407 系列器件内置最大 144 KB 字节的静态 SRAM，根据不同产品定义，SRAM 容量会有差异，具体请参考数据手册。它可以以字节、半字(16 位)或全字(32 位)访问。SRAM 的起始地址是 0x2000 0000。

96K SRAM1(映射到地址 0x2000 0000)

16K SRAM2(映射到地址 0x2001 8000)

32K CCM SRAM(映射到地址 0x1000 0000 和 SRAM2 的末端 0x2001 C000)

当选择从 SRAM1 启动或选择物理重新映射时，CPU 可以通过系统总线或通过 ICode/DCode 总线访问 SRAM1。

CCM SRAM 映射到地址 0x1000 0000。

由于通过 ICode 总线访问，可以从 CCM SRAM 执行最高性能，无需任何重新映射。

CCM SRAM 接续在 SRAM2 结束后的地址，与 SRAM1 和 SRAM2 提供连续的地址空间。

3.4.1 奇偶校验

SRAM1 的前 32 KB 和整个 CCM SRAM 带有奇偶校验。

用户可以使用用户选项字节中的选项位 SRAM_PE 启用奇偶校验。

数据总线宽度为 36 位，因为 4 位可用于奇偶校验(每字节 1 位)。

在写入 SRAM 时自动计算并存储奇偶校验位。然后，读取时自动检查。如果有 1 位失败，则生成 NMI 终中断。错误也可以链接到 TIM1/TIM8/TIM15/TIM16/TIM17 的 BRK_IN。

注:启用 SRAM 奇偶校验时，建议使用软件初始化 SRAM，以避免读取时出现奇偶校验错误。

3.4.2 CCM SRAM 写保护

CCM SRAM 可以采用 1 K 字节的页面粒度进行写保护。

写保护可以在 SYSCFG 中的 SYSCFG CCM SRAM 写保护寄存器(SYSCFG_SWPR)中启用。这是一个具有一次性写入“1”机制的寄存器，这意味着通过一次性写入“1”为 SRAM 的该页设置“写入保护”，并且只能通过系统复位来清除本设置。

表 3-3 CCM SRAM 配置

Page number	Start address	End address
Page 0	0x1000 0000 / 0x2001 C000	0x1000 03FF / 0x2001 C3FF
Page 1	0x1000 0400 / 0x2001 C400	0x1000 07FF / 0x2001 C7FF
Page 2	0x1000 0800 / 0x2001 C800	0x1000 0BFF / 0x2001 CBFF
Page 3	0x1000 0C00 / 0x2001 CC00	0x1000 0FFF / 0x2001 CFFF
Page 4	0x1000 1000 / 0x2001 D000	0x1000 13FF / 0x2001 D3FF
Page 5	0x1000 1400 / 0x2001 D400	0x1000 17FF / 0x2001 D7FF
Page 6	0x1000 1800 / 0x2001 D800	0x1000 1BFF / 0x2001 DBFF
Page 7	0x1000 1C00 / 0x2001 DC00	0x1000 1FFF / 0x2001 DFFF
Page 8	0x1000 2000 / 0x2001 E000	0x1000 23FF / 0x2001 E3FF
Page 9	0x1000 2400 / 0x2001 E400	0x1000 27FF / 0x2001 E7FF
Page 10	0x1000 2800 / 0x2001 E800	0x1000 2BFF / 0x2001 EBFF
Page 11	0x1000 2C00 / 0x2001 EC00	0x1000 2FFF / 0x2001 EFFF
Page 12	0x1000 3000 / 0x2001 F000	0x1000 33FF / 0x2001 F3FF
Page 13	0x1000 3400 / 0x2001 F400	0x1000 37FF / 0x2001 F7FF
Page 14	0x1000 3800 / 0x2001 F800	0x1000 3BFF / 0x2001 FBFF
Page 15	0x1000 3C00 / 0x2001 FC00	0x1000 3FFF / 0x2001 FFFF
Page 16	0x1000 4000 / 0x2002 0000	0x1000 43FF / 0x2002 03FF
Page 17	0x1000 4400 / 0x2002 0400	0x1000 47FF / 0x2002 07FF
Page 18	0x1000 4800 / 0x2002 0800	0x1000 4BFF / 0x2002 0BFF

Page 19	0x1000 4C00 / 0x2002 0C00	0x1000 4FFF / 0x2002 0FFF
Page 20	0x1000 5000 / 0x2002 1000	0x1000 53FF / 0x2002 13FF
Page 21	0x1000 5400 / 0x2002 1400	0x1000 57FF / 0x2002 17FF
Page 22	0x1000 5800 / 0x2002 1800	0x1000 5BFF / 0x2002 1BFF
Page 23	0x1000 5C00 / 0x2002 1C00	0x1000 5FFF / 0x2002 1FFF
Page 24	0x1000 6000 / 0x2002 2000	0x1000 63FF / 0x2002 23FF
Page 25	0x1000 6400 / 0x2002 2400	0x1000 67FF / 0x2002 27FF
Page 26	0x1000 6800 / 0x2002 2800	0x1000 6BFF / 0x2002 2BFF
Page 27	0x1000 6C00 / 0x2002 2C00	0x1000 6FFF / 0x2002 2FFF
Page 28	0x1000 7000 / 0x2002 3000	0x1000 73FF / 0x2002 33FF
Page 29	0x1000 7400 / 0x2002 3400	0x1000 77FF / 0x2002 37FF
Page 30	0x1000 7800 / 0x2002 3800	0x1000 7BFF / 0x2002 3BFF
Page 31	0x1000 7C00 / 0x2002 3C00	0x1000 7FFF / 0x2002 3FFF

3.4.3 CCM SRAM 读保护

CCM SRAM 具有读保护(RDP)。详细信息请参见 4.10.1 节:读保护(RDP)。

3.4.4 CCM SRAM 擦除

CCM SRAM 可以在复位时使用选项配置位 CCMSRAM_RST 进行内容的擦除（参考 4.8.1 选项字节描述)。同样也可以使用软件方法设置 CCMSR 控制位实现（参考 10.2.22 SYSCFG CCM SRAM 控制与状态寄存器（SYSCFG_SCSR)）。

3.5 嵌入式 Flash

Flash 存储器有两个不同的物理区域组成:

- Main flash 区域，512 KB，有两个 Bank 组成，它包含应用程序和用户数据。根据不同产品定义，Flash 容量会有差异，具体请参考数据手册。用于存储用户程序和用户数据。软件对设定范围外空间的访问会产生 Hard fault。
- Information 区域，32 KB，包括以下部分:
  - Factory config. bytes:
    - 芯片上电读校验码
    - 芯片硬件 Trimming 配置值
    - Device ID code
  - UID：512 Bytes，用于存放芯片的 UID
  - OTP：1024 Bytes，用于存放用户 OTP 数据
  - 选项字节：256 Bytes，用于存放芯片硬件和存储保护的配置值
  - System memory（系统存储器)：28 KB，用于存放 Boot loader

Flash 接口实现基于 AHB 协议的指令读取和数据访问，它通过寄存器实现了 Flash 的基本 program/erase 等操作。

3.6 位段

Cortex™-M4F 存储器映射包括两个位段区域。这些区域将存储器别名区域中的每个字映射到存储器位段区域中的相应位。在别名区域写入字时，相当于对位段区域的目标位执行读—修改—写操作。

在 PY32E407 器件中，外设寄存器和 SRAM 均映射到一个位段区域，这样可实现单个位段的读写操作。这些操作仅适用于 Cortex™-M4F 访问，对于其它总线主接口（如 DMA）无效。

可通过一个映射公式说明别名区域中的每个字与位段区域中各个位之间的对应关系。映射公式为：

$$bit_word_addr = bit_band_base + (byte_offset \times 32) + (bit_number \times 4)$$

其中：

- bit_word_addr 代表别名区域中将映射到目标位的字的地址
- bit_band_base 代表别名区域的起始地址
- byte_offset 代表目标位所在位段区域中的字节编号
- bit_number 代表目标位的位位置（0-7）

示例

下例说明如何将 SRAM 地址 0x2000 0300 处字节的位 2 映射到别名区域：

0x2200 6008 = 0x2200 0000 +（0x300*32）+（2*4)

对地址 0x2200 6008 执行写操作相当于在 SRAM 地址 0x2000 0300 处字节的位 2 执行读—修改—写操作。

对地址 0x2200 6008 执行读操作将返回 SRAM 地址 0x2000 0300 处字节的位 2 的值（0x01 表示位置位，0x00 表示位复位）。

有关位段的详细信息，请参见 Cortex™-M4F 编程手册（DUI508B_cortex_m4_ugrm.pdf）。

3.7 启动配置

通过配置位 BOOT0 pin，nBOOT0 / nBOOT1 / nSWBOOT0（存放于 option bytes 中)，可选择三种不同的启动模式，如下表所示：

表 3-4 Boot 配置

Boot mode configuration					Mode
BOOT_LOCK	nBOOT1 FLASH_OPTR2[8]	nBOOT0 FLASH_OPTR2[14]	BOOT0 Pin PB8	nSWBOOT0 FLASH_OPTR2[13]	
1	X	X	X	X	选择 Main flash 作为启动区
0	X	X	0	1	选择 Main flash 作为启动区
0	X	1	X	0	选择 Main flash 作为启动区
0	0	X	1	1	选择 SRAM1 作为启动区
0	0	0	X	0	选择 SRAM1 作为启动区
0	1	X	1	1	选择 System flash 作为启动区
0	1	0	X	0	选择 System flash 作为启动区

Boot loader 程序存储在 System memory,用于通过 USART、USB、SPI、I^2C 接口下载 Flash 程序。在系统复位后，BOOT0 引脚或用户选项配置位(取决于 FLASH_OPTR 寄存器中的 nSWBOOT0 位值)的值和 nBOOT1 位将被锁存。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 SRAM1 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 0000）访问它。
- 从内置 SRAM1 启动：SRAM1 被映射到启动空间（0x0000 0000），但仍然能够在它原有的地址（0x2000 0000）访问它。

在启动后，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(典型的从 Flash 启动)。PY32E407 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，重新映射向量表到 SRAM 中。

3.7.1 系统存储启动程序

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 USART 或者 USB 接口对闪存存储器进行重新编程。

3.7.2 物理重映射

选择启动引脚后，应用程序软件可以将某些存储器设定为从代码空间进行访问（这样，可通过 ICode 总线而非系统总线执行代码）。这样的修改通过在 SYSCFG 控制器中编程 SYSCFG.MEM_MODE 来实现。因此可重映射以下存储器：

- Main flash
- 系统存储器
- 嵌入式 SRAM1
- ESMC

表 3-5 物理重映射地址

地址	Main Flash 启动/重映射	嵌入式 SRAM1 启动/重映射	系统存储器 启动/重映射	ESMC 重映射
0x2000 0000 - 0x2001 7FFF	SRAM1	SRAM1	SRAM1	SRAM1
0x1FFF 0000 - 0x1FFF 7FFF	系统存储器	系统存储器	系统存储器	系统存储器
0x0808 0000 - 0x0FFF FFFF	保留	保留	保留	保留
0x0800 0000 - 0x0807 FFFF	Flash	Flash	Flash	Flash
0x0400 0000 - 0x07FF FFFF	保留	保留	保留	ESMC
0x0000 0000 - 0x03FF FFFF	Flash （使用别名）	SRAM1 （使用别名）	系统存储器（28 KB） （使用别名）	ESMC

4. 嵌入式 Flash 接口

4.1 简介

Flash 接口可管理 CPU 通过 AHB I-Code 和 D-Code 总线对 Flash 进行访问。该接口可针对 Flash 执行擦除和编程操作，并实施读写保护机制。

Flash 接口通过指令预取和缓存机制加速代码执行。

4.2 主要特征

- 存储器组织结构：
  - Main memory：最大 512 KB
  - Information memory：32 KB
  - Page size：512 Bytes
  - Sector size：4 KB
  - Block size：64 KB
  - 137-bit wide data read(128 bits data+9 bits ecc)
  - 137-bit wide data write(128 bits data+9 bits ecc)
- Flash 读操作
- Flash 编程操作（页编程）
- Flash 擦除操作（页擦/扇区擦/块擦/全擦）
- 读/写保护
- I-Code 上的预取操作
- I-Code 上的分支缓存
- D-Code 上的数据缓存
- 选项字节加载功能

4.3 闪存结构

Flash 存储器由 137 位宽(128 bits data+9 bits ecc)的 cell 组成，具有支持边读边写功能（RWW）的双 Bank 架构，可以用作程序和数据的存储。Page 大小为 512 Bytes，Sector 大小为 4 KB，Block 大小为 64 KB。

从功能上，Flash 存储器分为 Main flash 和 Information flash，前者容量最大是 512 KB，后者容量为 32 KB，如下：

表 4-1 闪存结构及边界地址

Area	Space Name				Address	Size(bytes)
Main memory	Bank0	Block 0	Sector 0	Page 0-7	0x0800 0000 - 0x0800 0FFF	4 K
			Sector 1	Page 8-15	0x0800 1000 - 0x0800 1FFF	4 K
			…	…	…	…
			Sector 15	Page 120-127	0x0800 F000 - 0x0800 FFFF	4 K
		…	…	…	…	…
		Block 3	Sector 48	Page 384-391	0x0803 0000 - 0x0803 0FFF	4 K
			Sector 49	Page 392-399	0x0803 1000 - 0x0803 1FFF	4 K
			…	…	…	…
			Sector 63	Page 504-511	0x0803 F000 - 0x0803 FFFF	4 K
	Bank1	Block 4	Sector 64	Page 512-519	0x0804 0000 - 0x0804 0FFF	4 K

			Sector 65	Page 520-527	0x0804 1000 - 0x0804 1FFF	4 K
			…	…	…	…
			Sector 79	Page 632-639	0x0804 F000 - 0x0804 FFFF	4 K
		…	…	…	…	…
		Block 7	Sector 112	Page 896-903	0x0807 0000 - 0x0807 0FFF	4 K
			Sector 113	Page 904-911	0x0807 1000 - 0x0807 1FFF	4 K
			…	…	…	…
			Sector 127	Page 1016-1023	0x0807 F000 - 0x0807 FFFF	4 K
Informatin block	Bank0	System memory			0x1FFF 0000 - 0x1FFF 6FFF	28 K
		OTP			0x1FFF 7000 - 0x1FFF 73FF	1 K
		选项字节			0x1FFF 7400 - 0x1FFF 75FF	512
		Trim0			0x1FFF 7600 - 0x1FFF 77FF	512
		Normal Temper Param			0x1FFF 7800 - 0x1FFF 79FF	512
		High Temper Param			0x1FFF 7A00 - 0x1FFF 7BFF	512
		FT infor0			0x1FFF 7C00 - 0x1FFF 7DFF	512
		UID			0x1FFF 7E00 - 0x1FFF 7FFF	512

4.4 错误代码纠正（ECC）

闪存中的数据为 137 位：每个 128 位增加 9 位 ECC。

ECC 机制支持:

- 一位错误检测和纠正
- 两位错误检测

当检测和纠正一位错误时，在 FLASH_ECCR 寄存器中设置标志 ECCC（ECC 纠正）。如同时设置了 ECCCIE，则会生成一个中断。

当检测到两位错误时，在 FLASH_ECCR 寄存器中设置标志 ECCD（ECC 检测）。在这种情况下，会生成一个 NMI 中断。

当检测到 ECC 错误时,故障地址及其相关的 Bank 被保存在 FLASH_ECCR 寄存器中的 ADDR_ECC，SYSF_ECC 和 BK_ECC 中。

当 ECCC 或 ECCD 被置位时，如果出现新的 ECC 错误，则不更新 ADDR_ECC，SYSF_ECC 和 BK_ECC。FLASH_ECCR 仅在清除 ECC 标志后更新。

注意：当访问初始数据:0x1FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF 时，会检测到一位错误并纠正。

4.5 读访问延迟

为了正确的从Flash存储器读取指令/数据,需要根据CPU时钟(HCLK)的频率,设置寄存器FLASH_ACR的等待周期数（Latency）。

在芯片复位后（包括上电复位，系统复位，退出 Standby 复位），HCLK 时钟频率为 16 MHz，Flash 读周期数为 0。

当改变CPU时钟频率或范围时,必须按照以下软件步骤来调整访问Flash存储器所需要的等待周期数。

提高 CPU 频率时的操作步骤

1. 将新的等待周期数编程到 FLASH_ACR 寄存器中的 LATENCY 位
2. 通过读取 FLASH_ACR 寄存器，检查新的等待周期是否设置成功
3. 通过改写 RCC_CFGR 寄存器中的 SW 位来修改 CPU 时钟源
4. 如有需要，可通过改写 RCC_CFGR 中的 HPRE 位来修改 CPU 时钟预分频器

5. 通过读取 RCC_CFGR 寄存器中相应的时钟源状态（ SWS 位）和/或 AHB 预分频值（ HPRE 位），检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功

降低 CPU 频率时的操作步骤

1. 通过改写 RCC_CFGR 寄存器中的 SW 位来修改 CPU 时钟源
2. 如有需要，可通过改写 RCC_CFGR 中的 HPRE 位来修改 CPU 时钟预分频器
3. 通过读取 RCC_CFGR 寄存器中相应的时钟源状态（ SWS 位）和/或 AHB 预分频值（ HPRE 位），检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功
4. 将新的等待周期数编程到 FLASH_ACR 中的 LATENCY 位
5. 通过读取 FLASH_ACR 寄存器，检查新的等待周期是否设置成功

4.6 自适应实时存储器加速器

专有的自适应实时(ART)存储器加速器面向 ARM® Cortex™-M4F 处理器 进行了优化。该加速器很好的体现了 ARM Cortex M4F 的固有性能优势，克服了通常条件下，高速的处理器在运行中需要经常等待 FLASH 读取的情况。

为了发挥处理器的全部性能，该加速器将实施指令预取队列和分支缓存，从而提高了 128 位 Flash 的程序执行速度。根据 CoreMark 基准测试，凭借 ART 加速器所获得的性能相当于 Flash 在 CPU 频率高达 170 MHz 时以 0 个等待周期执行程序。

- **指令预取**

每个 Flash 读操作可读取 128 位，可以是 4 行 32 位指令，也可以是 8 行 16 位指令，具体取决于烧写在 Flash 中的程序。因此对于顺序执行的代码，至少需要 4 个 CPU 周期来执行前一次读取的 128 位指令行。在 CPU 请求当前指令行时，可使用 I-Code 总线的预取操作读取 Flash 中的下一个连续存放的 128 位指令行。

- **指令缓存存储器**

为了减少因指令跳转而产生的时间损耗，可将 64 行 128 位的指令保存到指令缓存存储器中。每当出现指令缺失（即请求的指令未存在于当前使用的指令行、预取指令行或指令缓存存储器中）时，系统会将新读取的行复制到指令缓存存储器中。如果 CPU 请求的指令已存在于指令缓存区中，则无需任何延时即可立即获取。指令缓存存储器存满后，可采用 LRU（最近最少使用）策略确定指令缓存存储器中待替换的指令行。此特性非常适用于包含循环的代码。

- **数据管理**

在 CPU 流水线执行阶段，将通过 D-Code 总线访问 Flash 中的数据缓冲区。CPU 流水线只有获得请求的数据后才会继续执行。为了减少因此而产生的时间损耗，通过 AHB 数据总线 D-Code 进行的访问优先于通过 AHB 指令总线 I-Code 进行的访问。此特性的工作原理与指令缓存存储器类似，但保留的数据大小限制在 8 行 128 位以内。

4.7 擦除和编程操作

通过 ICP（In-circuit programming）或者 IAP（In-application programming）可以对 Flash 进 program 操作。

ICP：用来更新整个 Flash 存储器的内容，可以使用 JTAG/SWD 协议或者 Boot loader，把用户应用装入 MCU 中。

IAP：可以使用芯片支持的通讯接口，下载要编程的数据到 Flash 中。IAP 允许用户在应用运行时，再次 program flash 存储器。

如果在进行 Flash program 或 erase 操作时，发生了复位，则 Flash 存储器的内容是不被保护的。

在对一个 Bank 进行 program 或者 erase 操作期间，可以对另外一个 Bank 进行读访问。对正在 program 或者 erase 操作中 BANK 进行读写操作时，会产生 Hard fault。

对于 program 和 erase 操作，必须打开 HSI。

4.7.1 闪存解锁

在复位后，Flash 控制寄存器（FLASH_CR）不允许执行写操作，以防止因电气干扰等原因出现对 Flash 的意外操作。每次对 Flash 的 erase 和 program 操作，都必须通过写 FLASH_KEYR 来解锁，以启用对 FLASH_CR 寄存器的访问。

具体步骤如下：

步骤 1：向 FLASH_KEYR 寄存器写入 KEY1=0x4567 0123

步骤 2：向 FLASH_KEYR 寄存器写入 KEY2=0xCDEF 89AB

任何错误的时序都会锁定住 FLASH_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，会产生总线 Hard fault 中断。这样的错误包括第一个写周期的 KEY1 不匹配，或者 KEY1 匹配但第二个写周期的 KEY2 不匹配。

FLASH_CR 寄存器可以通过软件写 FLASH_CR 寄存器的 LOCK 位被再次锁定。

注意：当 FLASH_SR 寄存器的 BSY 位被置位时，FLASH_CR 寄存器不能进行写操作。

4.7.2 闪存擦除操作

Flash 擦除操作支持 page、sector、block、bank erase，执行 Bank0 erase 时不会对 information memory 起作用。

4.7.2.1 闪存页擦除

Page erase 用来对 512 Bytes 的 Main flash 进行 erase 操作，但对 Information 区不起作用。

当某个 page 被 WRP 保护，它是不会被 erase 的，此时 WRPERR 位被置位。

page erase 操作的具体步骤如下：

1）检查 FLASH_SR 寄存器 BSY 位，确认没有正在进行的 flash 操作

2）向 FLASH_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH_CR 寄存器的保护

3）设置 FLASH_CR 寄存器的 PER 位和 EOPIE 位

4）向该 page 写任意数据（必须 32 位 数据）

5）等待 BSY 位被清零

6）检查 EOP 标志位被置位

7）清除 EOP 标志

4.7.2.2 闪存扇区擦除

Sector erase 用来对 4 KB 的 Main flash 进行 erase 操作，但对 Information 区不起作用。

当某个 sector 被 WRP 保护，它是不会被 erase 的，此时 WRPERR 位被置位。

sector erase 操作的具体步骤如下：

1）检查 BSY 位，确认是否没有正在进行的 Flash 操作

2）向 FLASH_KEYR 寄存器依次写 KEY1、KEY2，解除 FLASH_CR 寄存器保护
3）设置 FLASH_CR 寄存器的 SER 位和 EOPIE 位
4）向该 sector 写任意数据（必须 32 位数据）
5）等待 BSY 位被清零
6）检查 EOP 标志位被置位
7）清除 EOP 标志

4.7.2.3 闪存块擦除

Block erase 用来对 32 KB 的 Main flash 进行 erase 操作，但对 Information 区不起作用。
当某个 Block 被 WRP 保护，它是不会被 erase 的，此时 WRPERR 位被置位。
block erase 操作的具体步骤如下：
1）检查 BSY 位，确认是否没有正在进行的 Flash 操作
2）向 FLASH_KEYR 寄存器依次写 KEY1、KEY2，解除 FLASH_CR 寄存器保护
3）设置 FLASH_CR 寄存器的 BER 位和 EOPIE 位
4）向该 block 写任意数据（必须 32 bits 数据）
5）等待 BSY 位被清零
6）检查 EOP 标志位被置位
7）清除 EOP 标志

4.7.2.4 闪存 Bank 擦除

Bank erase 用来对 256 KB 的 Main flash 中进行 erase 操作，但对 Information 区不起作用。
当 WRP 被使能，Bank erase 功能无效，不会产生 Bank erase 操作，并且 WEPERR 位被置位。
Bank erase 操作的具体步骤如下：
1）检查 BSY 位，确认是否没有正在进行的 Flash 操作
2）向 FLASH_KEYR 寄存器依次写 KEY1,KEY2，解除 FLASH_CR 寄存器保护
3）设置 FLASH_CR 寄存器的 MER1 位或 MER0 位和 EOPIE 位
4）向对应 Bank Flash 的任意 Flash 空间写任意数据（必须 32 bits 数据）
5）等待 BSY 位被清零
6）检查 EOP 标志位被置位
7）清除 EOP 标志

4.7.3 闪存写操作

Flash 存储器每次以 32 bits(word)为单位（进行 half word 或者 byte 操作会产生 Hard fault）进行整个 page 的 program 操作。当 FLASH_CR 寄存器的 PG 位被置位，CPU 向 Flash 存储器地址空间写 32 bits 数据时，program 操作开始启动。
如果要 program 的 flash 地址空间，是被 FLASH_WRPR 寄存器设置为保护的区域，则 program 操作会被忽略掉，同时 FLASH_CR 寄存器 WRPRTERR 位会被置位。Program 操作的结束，FLASH_CR 寄存器的 EOP 位会被置位。
具体 Flash program 的操作步骤如下所示：
1）检查 FLASH_SR 寄存器的 BSY 位，判断是否当前没有正在继续的 Flash 操作
2）如果没有正在进行的 Flash erase 或者 program 操作，则软件读出该 Page 的 128 个 word（如果需要保留该 page 已有的数据，则进行该步骤，否则跳过该步骤）

3）向 FLASH_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH_CR 寄存器的保护

4）设置 FLASH_CR 寄存器的 PG 位和 EOPIE 位

5）向目标地址进行第 1 到第 127 个 word 的 program 操作（只接受 32 位 的 program）

6）设置 FLASH_CR 寄存器的 PGSTRT

7）写第 128 个 word

8）等待 FLASH_SR 寄存器的 BSY 位被清零

9）检查 FLASH_SR 寄存器的 EOP 标志位（当 program 操作已经成功，该位被置位），然后软件清除该位

10）如果不再有 program 操作，则软件清除 PG 位

当上述步骤 7）成功执行，则 program 操作自动启动，同时 BSY 位被硬件置位。

4.8 Flash 选项字节

4.8.1 Flash 选项字节描述

芯片内的 Flash 的 Information 区域的部分区域作为选项字节使用，用来存放芯片或者用户针对应用需要对硬件进行的配置。比如，看门狗可以选择为硬件或者软件模式。

为了数据的安全性，选项字节以正文及反码形式分别存储。

表 4-2 选项字节格式

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

选项字节的内容可以从表 4-3 选项字节结构所述的存储器地址读到，也可以从以下选项字节的相关寄存器读到：

- Flash 用户选项寄存器 1（FLASH_OPTR1）
- Flash 用户选项寄存器 2（FLASH_OPTR2）
- Flash WRP 地址寄存器（FLASH_WRPR）
- Flash PCROP0SR 地址寄存器（FLASH_PCROP0SR）
- Flash PCROP0ER 地址寄存器（FLASH_PCROP0ER）
- Flash PCROP1SR 地址寄存器（FLASH_PCROP1SR）
- Flash PCROP1ER 地址寄存器（FLASH_PCROP1ER）

表 4-3 选项字节结构

Word Address	Description
0x1FFF 7400	用户选项字节 1 及反码
0x1FFF 7404	用户选项字节 2 及反码
0x1FFF 7408	保留
0x1FFF 740C	WRP 地址选项字节及反码
0x1FFF 7410	PCROP0SR 地址选项字节及反码
0x1FFF 7414	PCROP0ER 地址选项字节及反码
0x1FFF 7418	PCROP1SR 地址选项字节及反码
0x1FFF 741C	PCROP1ER 地址选项字节及反码

…	保留
0x1FFF 74FF	保留

4.8.1.1 Flash 用户选项 1 的选项字节

Flash memory address：0x1FFF 7400

Production value：0x0955 F6AA

在上电复位（POR/OBL_LAUNCH）释放后,从 Flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~WWDG_SW	~nRST_STDBY	~nRST_STOP	~IWDG_SW	~BFB	~IWDG_STDBY	~IWDG_STOP	Res	~RDP[7:0]							
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
WWDG_SW	nRST_STDBY	nRST_STOP	IWDG_SW	BFB	IWDG_STDBY	IWDG_STOP	Res	RDP[7:0]							
R	R	R	R	R	R	R		R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	~WWDG_SW	R	WWDG_SW 的反码
30	~nRST_STDBY	R	NRST_STDBY 的反码
29	~nRST_STOP	R	NRST_STOP 的反码
28	~IWDG_SW	R	IWDG_SW 的反码
27	~BFB	R	BFB 的反码
26	~IWDG_STDBY	R	IWDG_STDBY 的反码
25	~IWDG_STOP	R	IWDG_STOP 的反码
24	保留	-	-
23：16	~RDP	R	RDP 的反码
15	WWDG_SW	R	0：硬件窗口 watchdog 1：软件窗口 watchdog
14	nRST_STDBY	R	0：进入 Standby 模式时产生 Reset 1：无 Reset 产生
13	nRST_STOP	R	0：进入 Stop 模式时产生 Reset 1：无 Reset 产生
12	IWDG_SW	R	0：硬件独立 watchdog 1：软件独立 watchdog
11	BFB	R	0：从 Bank0 启动(Bank0 映射在 0x0800 0000 Bank1 映射在 0x0804 0000) 1：从 Bank1 启动(Bank1 映射在 0x0800 0000 Bank0 映射在 0x0804 0000)
10	IWDG_STDBY	R	设置 IWDG 在 Standby 模式下定时器运行状态 0：freeze 定时器 1：正常运行
9	IWDG_STOP	R	设置 IWDG 在 Stop 模式下定时器运行状态 0：freeze 定时器 1：正常运行
8	保留	-	-
7：0	RDP	R	0xAA：level 0，读保护无效 非 0xAA：level 1，读保护有效

4.8.1.2 Flash 用户选项 2 的选项字节

Flash memory address：0x1FFF 7404

Production value：0x80FF 7F00

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~ BOOT_LOCK	~ nBOOT0	~ nSW-BOOT0	~ CCMSRAM_RST	~ SRAM_PE	~ NRST_MODE		~ nBOOT1	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
BOOT_LOCK	nBOOT0	nSW-BOOT0	CCMSRAM_RST	SRAM_PE	NRST_MODE		nBOOT1	Res							
BOOT_LOCK	R	R	R	R	R	R	R	-							

Bit	Name	R/W	Function
31	~BOOT_LOCK	R	BOOT_LOCK 的反码
30	~nBOOT0	R	nBOOT0 的反码
29	~nSWBOOT0	R	nSWBOOT0 的反码
28	~CCMSRAM_RST	R	CCMSRAM_RST 的反码
27	~ SRAM_PE	R	SRAM_PE 的反码
26：25	~ NRST_MODE	R	NRST_MODE 的反码
24	~ nBOOT1	R	nBOOT1 的反码
23：16	保留	-	-
15	BOOT_LOCK	R	用于强制从用户 Flash 区域启动 0：基于 pad/选项位配置启动 1：从主闪存强制启动
14	nBOOT0	R	nBOOT0 选项位 0:nBOOT0=0 1:nBOOT0=1
13	nSWBOOT0	R	软件 BOOT0 0:BOOT0 取自选项位 nBOOT0 1:BOOT0 取自 PB8/BOOT0 引脚
12	CCMSRAM_RST	R	系统复位时 CCM SRAM 擦除： 0:系统复位发生时 CCM SRAM 擦除 1:系统复位发生时 CCM SRAM 不擦除
11	SRAM_PE	R	SRAM1 和 CCM SRAM 奇偶校验许可位 0：许可 1：禁止
10：9	NRST_MODE	R	PF5 pad mode 00：Reset 输入/输出 01：Reset 输入 10：GPIO 11：Reset 输入/输出
8	nBOOT1	R	启动配置与 BOOT0 引脚决定从闪存主存储器、SRAM1 或系统存储器中选择启动模式。 请参阅第 3.7 节：启动配置。
7：0	保留	-	-

4.8.1.3 Flash WRP 地址的选项字节

Flash memory address：0x1FFF 740C

Production value：0xFF00 00FF

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Function
31：24	保留	-	-
23：16	~WRP	R	WRP 的反码
15：8	保留	-	-
7：0	WRP	R	0：block[y]被保护 1：block [y]无保护 y=0~7

4.8.1.4 Flash PCROP0SR 地址的选项字节

Flash memory address：0x1FFF 7410

Production value：0xFE00 01FF

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							~ PCROP0SR[8:0]								
-							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
Res							PCROP0SR[8:0]								
-							R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31：25	保留	-	-
24：16	~PCROP0SR	R	PCROP0SR 的反码
15：9	保留	-	-
8：0	PCROP0SR	R	BANK0 PCROP 区域起始地址(以 Page 为单位)

4.8.1.5 Flash PCROP0ER 地址的选项字节

Flash memory address：0x1FFF 7414

Production value：0xFFFF 0000

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							~ PCROP0ER[8:0]								
-							R	R	R	R	R	R	R	R	R
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							PCROP0ER[8:0]								
-							R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function

31：25	保留	-	-
24：16	~PCROP0ER	R	PCROP0ER 的反码
15：9	保留	-	-
8：0	PCROP0ER	R	BANK0 PCROP 区域结束地址(以 Page 为单位)

4.8.1.6 Flash PCROP1SR 地址的选项字节

Flash memory address：0x1FFF 7418

Production value：0xFE00 01FF

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							~ PCROP1SR[8:0]								
-							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
Res							PCROP1SR[8:0]								
-							R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31：25	保留	-	-
24：16	~PCROP1SR	R	PCROP1SR 的反码
15：9	保留	-	-
8：0	PCROP1SR	R	BANK1 PCROP 区域起始地址(以 Page 为单位)

4.8.1.7 Flash PCROP1ER 地址的选项字节

Flash memory address：0x1FFF 741C

Production value：0xFFFF 0000

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							~ PCROP1ER[8:0]								
-							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
Res							PCROP1ER[8:0]								
-							R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31：25	保留	-	-
24：16	~PCROP1ER	R	PCROP1ER 的反码
15：9	保留	-	-
8：0	PCROP1ER	R	BANK1 PCROP 区域结束地址(以 Page 为单位)

4.8.2 Flash 选项字节写

复位后，FLASH_CR 寄存器中与选项字节相关的位是被写保护的。当对选项字节进行相关操作前，FLASH_CR 寄存器中的 OPTLOCK 位必须被清零。

以下步骤用来解锁该寄存器：

1. 通过解锁流程，解除 FLASH_CR 寄存器的写保护
2. 向 FLASH_OPTKEYR 寄存器，写 OPTKEY1=0x0819 2A3B
3. 向 FLASH_OPTKEYR 寄存器，写 OPTKEY2=0x4C5D 6E7F

任何错误的时序都会锁住 FLASH_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，会产生总线 Hard Fault 中断。

User option（information flash 的选项字节）可以通过软件写 FLASH_CR 寄存器的 OPTLOCK 位，被保护住，以防止不要的 erase/program 操作。

如果软件设置 Lock 位，则 OPTLOCK 位也被自动置位。

4.8.2.1 修改选项字节

选项字节的 program 操作，和 Main flash 的操作不一样。为修改选项字节，需要进行如下步骤：

1. 用之前描述的步骤，清零 OPTLOCK 位
2. 检查 BSY 位，确认没有正在进行的 Flash 操作
3. 向选项字节寄存器 FLASH_OPTR1/FLASH_OPTR2/FLASH_WRPR/FLASH_PCROR0SR/FLASH_PCROR0ER/FLASH_PCROR1SR/FLASH_PCROR1ER 写期望的值（1~7 个 word）
4. 设置 OPTSTRT 位和 EOPIE 位
5. 向 0x1FFF7400 地址写任意数据（必须 32 bits 数据）
6. 等待 BSY 位被清零
7. 检查 EOP 标志位被置位
8. 清除 EOP 标志

任何对选项字节的改动，硬件都会先把选项字节对应的整个 page erase 掉，然后用 FLASH_OPTR1、FLASH_OPTR2、FLASH_WRPR、FLASH_PCROR0SR、FLASH_PCROR0ER、FLASH_PCROR1SR、FLASH_PCROR1ER 寄存器的值，写入选项字节中。并且，硬件自动计算相应的补码，并把计算值写入选项字节的相应区域。

4.8.2.2 加载选址字节

在 BSY 位被清零后，所有新的选项字节被写入了 Flash information 存储器中，但是未应用于芯片系统。对选项字节寄存器进行读操作，仍然返回上一次被装载的选项字节里的值。仅当他们（新值）被装载后，才对芯片系统起作用。

选项字节的装载，在以下两种情况下进行：

- 当 FLASH_CR 寄存器中的 OBL_LAUNCH 位被置位
- 在上电复位后（POR）

“装载选项字节”进行的操作是：对 information memory 区域的选项字节进行读操作，再把读出的数据存储在内部 option 寄存器中（FLASH_OPTR1、FLASH_OPTR2、FLASH_WRPR、FLASH_PCROR0SR、FLASH_PCROR0ER、FLASH_PCROR1SR、FLASH_PCROR1ER）。

每个 option 位在它相同的双字地址（下一个 half word）有相应的补码。在选项字节装载期间，会对 option bit 和其反码进行验证，确保装载被正确进行。

如果正反码匹配，则选项字节被复制到 option 寄存器中。

如果正反码不匹配，则 FLASH_SR 寄存器的 OPTVERR 状态位被置位。不匹配的值被写入 option 寄存器：

- 对于 OPTR
  - RDP 位写成 0xff（即 level 1）
  - BFB 为写为 0（即不进行 bank 交换）
  - 其余不匹配的值都写成 1
- 对于 WRP option，不匹配的值是默认值“无保护”
- 对于 PCROP option，不匹配的值是默认值“全闪存保护”

在系统复位后，选项字节的内容被复制到下面的 option 寄存器（软件可读可写）：

- FLASH_OPTR1
- FLASH_OPTR2
- FLASH_WRPR
- FLASH_PCROP0SR
- FLASH_PCROP0ER
- FLASH_PCROP1SR
- FLASH_PCROP1ER

这些寄存器也被用来修改选项字节。

4.9 闪存用户 OTP 字节

芯片内的 Flash 的 information 区域的部分区间作为 Flash USER OTP memory Bytes。

共计 2 Page，每个 Page 分别控制。

表 4-4 USER OTP 结构

Page	Word	Address	Contents
1	0	0x1FFF 7000	Bit[31:16]:存放用户数据 Bit[15:0]：USER OTP MEMORY_LOCK
	1	0x1FFF 7004	存放用户数据
	…	…	存放用户数据
	127	0x1FFF 71FC	存放用户数据
2	0	0x1FFF 7200	Bit[31:16]:存放用户数据 Bit[15:0]：USER OTP MEMORY_LOCK
	1	0x1FFF 7204	存放用户数据
	…	…	存放用户数据
	127	0x1FFF 73FC	存放用户数据

这 2 个 Page 配置在 information 区域,对本 Page 区域 program 和擦是按照 Main flash 的方法来处理。

另外，Main Flash 区域的 mass erase 对本区域无效。

设定 USER OTP MEMORY_LOCK 内容不会立刻更新，直到上电复位（POR/PDR），从会起到保护功能。

对本 Page write 有如下保护。

表 4-5 Flash USER OTP 写保护

USER OTP MEMORY_LOCK	Write protection
0xAA55	读：可以 program 和擦操作：不可以
除(0xAA55)之外的任何值	读、program 和擦操作：可以

4.10 闪存保护

对 Flash main memory 的保护包括以下几种机制：

- Read protection（RDP），防止来自外部的访问。
- Wrtie protection（WRP）控制，以防止不要的写操作（由于程序存储器指针 PC 的混乱）。写保护的粒度设计为 32 KB。

- 选项字节写保护，有专门的解锁设计。

4.10.1 闪存读保护(RDP)

通过设置 RDP 选项字节，并进行系统复位（POR 或者 OBL 复位)装载新的 RDP 选项字节，可以激活读保护功能。RDP 保护 flash main memory、CCM SRAM 和备份寄存器。

如果通过 SWD 的 debug 仍在连接时，读保护被设置，需要进行上电复位而不是系统复位。

当 RDP 选项字节和反码成对正确存在于选项字节时，Flash memory 会被保护。

表 4-6 闪存读保护状态

RDP 选项字节值	RDP 选项字节反码值	等保护等级
0xAA	0x55	Level 0
除[0xAA,0x55]组合的任何值		Level 1

无论任何保护级别，System memory 都是可读的，但不能进行 program 和 erase 操作。

4.10.1.1 Level 0：no protection

对 Main flash 的读、编程和擦除操作是可以的，对选项字节、CCM SRAM 和备份寄存器也是可以进行任何操作。

4.10.1.2 Level 1：Read protection

当选项字节里的 RDP 及其反码包含任何除了[0xAA,0x55]之外的组合，则 level 1 read protection 生效，Level 1 是默认的保护级别。

- User mode：在用户模式下执行的程序（boot from main flash），可以对 Main flash、选项字节、CCM SRAM 和备份寄存器进行所有操作。
- Debug、boot from SRAM、boot from system memory modes：在 debug 模式，或者当从 SRAM 或者 system memory 启动，main flash、CCM SRAM 和备份寄存器是不能被访问的。在这些模式下，对 main flash 读或者写访问会产生总线 hard fault 中断。

读保护级别可以改变：

- 从 Level 0 到 Level 1，将 RDP 字节的值更改为除 0xAA 之外的任何值
- 从 Level 1 到 Level 0，将 RDP 字节的值更改为 0xAA

Level 1 变为 Level 0，会触发硬件 mass erase Main flash。

表 4-7 访问状态与保护级别和执行模式的关系

Area	Protection level	User execution（BootFromFlash）			Debug/ BootFromRam/ BootFromLoader		
		Read	Write	Erase	Read	Write	Erase
Main memory	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	Yes	No	No	No
System memory	x	Yes	No	No	Yes	No	No
选项字节	x	Yes	Yes	N/A	Yes	Yes	N/A
OTP	x	Yes	Yes	Yes	Yes	Yes	Yes
Backup registers	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	Yes	No	No	No
CCM SRAM	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	Yes	No	No	No

注：

(1) Information 区域只读，和保护等级及执行模式无关。

(2) RDP 从 Level 1 修改为 Level 0，会触发硬件对 Main flash 的 mass erase。

(3) 对于从 SRAM 或者 system memory 执行程序包括两种情况：

一个是 Boot from，另一个是从别的存储器 boot，程序跳转到 SRAM 或者 system memory 执行。

4.10.2 闪存写保护（WRP)

Flash 可以被设置成写保护，以应对不要的写操作。定义 WRP 寄存器每 bit 的控制粒度为 64 KB 的写保护（WRP）区域，即 1 个 block 大小。具体参见 WRP 寄存器的描述。

当 WRP 的区域被激活，则不允许进行 erase 或者 program 操作。相应的，即使只有一个区域被设定为写保护，则 bank erase 功能不起作用。此外，如果尝试对设为写保护的区域进行 erase 或者 program 操作，则 FLASH_SR 寄存器的写保护错误标识（WRPERR）会被置位。

注：写保护仅对 Main flash 起作用，对 system memory 不起作用。

4.10.3 专有代码读出保护(PCROP)

除了闪存之外，还可以防止来自第三方的读写。保护区仅执行：它只能由 CPU 作为指令代码访问，而严格禁止所有其他访问（DMA、调试和 CPU 数据读取、写入和擦除）。当 RDP 保护从级别 1 更改为级别 0 时，包括 PCROP 区域也会被擦除（请参阅更改读取保护级别）。

每个 PCROP 区域由与物理闪存地址相关的起始页和结束页地址定义。这些地址定义在 PCROP 地址寄存器 FLASH PCROP 起始地址寄存器（FLASH_PCROP0SR，FLASH_PCROP1SR）、FLASH PCROP 结束地址寄存器（FLASH_PCROP0ER ，FLASH_PCROP1ER）。

通过 D-code 总线对 PCROP 保护区执行的任何读取访问都会触发 RDERR 标志错误。

任何受 PCROP 保护的地址也是写保护的，对这些地址之一的任何写访问都会触发 WRPERR。

任何 PCROP 区域也受到擦除保护。因此，对该区域中的页面的任何擦除都是不可能的（包括包含该区域的开始地址和结束地址的页面）。此外，如果一个区域受到 PCROP 保护，则不能执行软件 Bank Mass 擦除。

只有当 RDP 从级别 1 更改为级别 0 时，才能禁用 PCROP。如果通过修改用户选项试图清除 PCROP 或缩小 PCROP 区域，可启动选项编程，但 PCROP 区域保持不变。相反，可以增加 PCROP 区域。

表 4-8 PCROP 保护

PCROPx 寄存器值(x = 0,1)	PCROP 保护区
PCROPxSR >PCROPxER	无 PCROP 区域。
PCROPxSR <=PCROPxER	PCROPxSR 和 PCROPxER 之间的区域受到保护。

4.10.4 强制从 Main flash 启动

为了提高安全性，FLASH_OPTR2 寄存器的 BOOT_LOCK 选项位允许强制系统从主闪存启动，从而无视其他启动选项。重置 BOOT_LOCK 只有在以下情况：

- RDP 设置为级别 0
- RDP 级别 1 修改为级别 0

4.11 闪存中断

表 4-9 Flash 中断请求

中断事件	事件标志	时间标志/中断清除方法	控制位使能
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE
Read error	RDERR	Write RDERR=1	RDERRIE
ECC correction	ECCC	Write ECCC=1	ECCCIE

以下事件没有单独的中断标识，但会产生 Hard fault：

- 解锁 Flash memory 的 FLASH_CR 寄存器的流程错误
- 解锁 Flash 选项字节的写操作流程错误
- Flash program,erase 操作未进行 32 位数据的对齐
- 对 program,erase 中的 Bank 进行读写访问
- 一个 Bank program,erase 中，另一个 Bank 再进行 program,erase

4.12 寄存器描述

4.12.1 Flash 访问控制寄存器(FLASH_ACR)

Address offset：0x00

Reset value：0x0000 0700

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

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	-
10:8	保留		3’h7	只能写初始值
7:4	保留	-	-	-
3:0	LATENCY[3:0]	RW	4’h0	Flash 读操作对应的等待状态： HCLK <= 25 MHz 0：flash 读操作没有等待状态，即每次读 flash 需要 1 个系统时钟周期 25 MHz < HCLK <= 50 MHz 1：flash 读操作有 1 个等待状态，即每次读 flash 需要 2 个系统时钟周期 50 MHz < HCLK <= 75 MHz 3：flash 读操作有 3 个等待状态，即每次读 flash 需要 4 个系统时钟周期 75 MHz < HCLK <= 100 MHz 4：flash 读操作有 4 个等待状态，即每次读 flash 需要 5 个系统时钟周期 100 MHz < HCLK <= 125 MHz 5：flash 读操作有 5 个等待状态，即每次读 flash 需要 6 个系统时钟周期

				125 MHz < HCLK <= 150 MHz 6：flash 读有 6 个等待状态，即每次读 flash 需要 7 个系统时钟周期 150 MHz < HCLK <= 170 MHz 7：flash 读操作有 7 个等待状态，即每次读 flash 需要 8 个系统时钟周期 其余值不建议配置 注：在配置 latency 位时，该寄存器其他位不能写入任何值

4.12.2 Flash 密钥寄存器(FLASH_KEYR)

Address offset：0x08

Reset value：0x0000 0000

所有寄存器位是 write-only，读出返回 0。

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

Bit	Name	R/W	Reset Value	Function
31:0	KEY[31:0]	W	32’h0	下面的值必须被连续的写入，才能解锁 FLASH_CR 寄存器，并使能 flash 的 program/erase 操作 KEY1：0x4567 0123 KEY2：0xCDEF 89AB

4.12.3 Flash 选项密钥寄存器（FLASH_OPTKEYR)

Address offset:0x0C

Reset value:0x0000 0000

所有寄存器位是 write-only，读出返回 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEY[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**
OPTKEY[15:0]															
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:0	OPTKEY[31:0]	W	32’h0	下面的值必须被连续的写入，才能解锁 flash 的 option 寄存器，并使能选项字节的 program/erase 操作 KEY1：0x0819 2A3B KEY2：0x4C5D 6E7F

4.12.4 Flash 状态寄存器(FLASH_SR)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY1	BSY0
-														R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	RD ERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP ERR	Res.	Res.	Res.	EOP
RC_W1	RC_W1	-									RC_W1	-			RC_W1

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	BSY1	R	0	Busy 位 该位表示 flash bank1 的操作正在进行。该位在 flash 操作的开始被硬件置位，当操作完成或者错误产生时由硬件清零。
16	BSY0	R	0	Busy 位 该位表示 flash bank0 的操作正在进行。该位在 flash 操作的开始被硬件置位，当操作完成或者错误产生时由硬件清零。
15	OPTVERR	RC_W1	0	选项字节加载错误 当 option 及其反码不匹配时，硬件置位该位。装载不匹配的选项字节，被强制成安全值，参考 FLASH 选项字节。 软件写 1，清零。
14	RDERR	RC_W1	0	PCROP 读取错误 当要通过 D-code 总线读取的地址属于闪存的读取保护区域时由硬件置位（PCROP 保护）。 软件写 1，清零。
13:5	保留	-	-	-
4	WRPERR	RC_W1	0	写保护错误 当要被 program/erase 的地址处于被写保护的 flash 区域时（WRP），硬件置位该位。 软件写 1，清零。
3:1	保留	-	-	-
0	EOP	RC_W1	0	当 flash 的 program/erase 操作成功完成，硬件置位。该位仅当 FLASH_CR 寄存器的 EOPIE 位使能才会被置位。 软件写 1，清零。

4.12.5 Flash 控制寄存器(FLASH_CR)

Address offset:0x14

Reset value:0xC000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	Res.	OBL_ LAUN CH	RD ERR IE	ERR IE	EOP IE	Res.	Res.	Res.	Res.	PG STRT	Res.	OPT STRT	Res.
RS	RS	-		RC_W1	RW	RW	RW	-				RW	-	RW	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BER	SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER1	MER0	PER	PG
-			RW	RW	-							RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Lock	RS	1	FLASH_CR Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器被锁住。当成功给出解锁流程后，该位被硬件清零。

				【软件要在 program/erase 操作完成后，置位该位】 当不成功的解锁流程给出，该位仍然保持置位状态，直到下一次系统复位。
30	OPTLOCK	RS	1	选项字节 Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器中与选项字节有关的位被锁住。当成功给出解锁流程后，该位被硬件清零。 【软件要在 program/erase 操作完成后，置位该位】 当不成功的解锁时序给出，该位仍然保持置位状态，直到下一次系统复位。
29：28	保留	-	-	-
27	OBL_LAUNCH	RC_W1	0	强制选项字节加载 当置位时，该位强制系统进行选项字节的重装载。该位仅当选项字节装载被完成后被硬件清零。如果 OPTLOCK 位被置位，该位不能被写. 0：选项字节加载完成 1：产生选项字节加载请求，产生系统复位，进行选项字节的重装载。
26	RDERRIE	RW	0	PCROP 读取错误中断使能 当 FLASH_SR 中的 RDERR 位被置位，如果该位使能，则产生中断请求。 0：无中断产生 1：有中断产生
25	ERRIE	RW	0	错误中断使能 当 FLASH_SR 寄存器的 WRPERR 位被置位，如果该位使能，则产生中断请求。 0：无中断产生 1：有中断产生
24	EOPIE	RW	0	操作结束中断使能 当 FLASH_SR 寄存器的 EOP 位被置位，该位使能中断的产生。 0：EOP 中断关闭 1：EOP 中断使能
23：18	保留	-	-	-
19	PGSTRT	RW	0	Flash main memory 的 program 操作的启动位。 该位启动了 Flash main memory 的 program 操作，软件置位，在 FLASH_SR 寄存器的 BSY 位被清零后，硬件清零该位。
18	保留	-	-	-
17	OPTSTRT	RW	0	Flash 选项字节修改的启动位 该位启动了对选项字节的修改。软件置位，在 FLASH_SR 寄存器的 BSY 位被清零后，硬件清零该位。 注意：当对 flash 选项字节进行修改时，硬件自动把整个 128Bytes 的 page 进行 erase 操作，再进行 program 操作，其中也包括自动进行反码的写入。
16:13	保留	-	-	-

12	BER	RW	0	Block erase 操作 0：未选择 flash 的 block erase 操作 1：选择 flash 的 block erases 操作
11	SER	RW	0	Sector erase 操作 0：未选择 flash 的 sector erase 操作 1：选择 flash 的 sector erases 操作
10:4	保留	-	-	-
3	MER1	RW	0	Bank 1 erase 操作 0：未选择 flash 的 bank1 erase 操作 1：选择 flash 的 bank1 erases 操作
2	MER0	RW	0	Bank 0 erase 操作 0：未选择 flash 的 bank0 erase 操作 1：选择 flash 的 bank0 erases 操作
1	PER	RW	0	Page erase 操作 0：未选择 flash 的 page erase 操作 1：选择 flash 的 page erase 操作
0	PG	RW	0	Program 操作 0：未选择 flash 的 program 操作 1：选择 flash 的 program 操作

4.12.6 Flash ECC 寄存器（FLASH_ECCR)

Address offset:0x18

Reset value：0x0000 0000。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res	Res	Res	Res	Res	ECCCIE	Res	SYSF_ECC	BK_ECC	Res	Res	Res	ADDR_ECC[17:16]	
RC_W1	RC_W1	-					RW	-	R	R	-			R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR_ECC[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31	ECCD	RC_W1	0	ECC 检测标志位 在检测到 ECC 两位错误时由硬件置位，当该位被设置时，生成 NMI 中断。 通过写入 1 清除。
30	ECCC	RC_W1	0	ECC 纠正标志位 当检测到并纠正了一位 ECC 错误时由硬件置位，当 ECCCIE 设置时，产生中断。 通过写入 1 清除。
29:25	保留	-	-	-
24	ECCCIE	RW	0	ECC 纠正中断使能 0：禁用 ECCC 中断 1：启用 ECCC 中断。
23	保留	-	-	-

22	SYSF_ECC	R	0	ECC 故障 memory 0：Main memory 1：information memory
21	BK_ECC	R	0	ECC Main memory 故障 bank 0:bank 0 1:bank 1
20:18	保留	-	-	-
17:0	ADDR_ECC[17:0]	R	18'h0	ECC 故障地址 记录故障地址

4.12.7 Flash 选项 1 寄存器(FLASH_OPTR1)

Address offset:0x20

Reset value：0x0000 XXXX。

在上电复位（POR/OBL_LAUNCH）释放后，从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WWDG_SW.	nRSTSTDBY	nRSTSTOP	IWDG_SW	BFB	IWDG_STDBY	IWDG_STOP	Res.	RDP[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	WWDG_SW	RW		0：硬件窗口 watchdog 1：软件窗口 watchdog
14	nRST_STDBY	RW		0：进入 Standby 模式时产生 Reset 1：无 Reset 产生
13	nRST_STOP	RW		0：进入 Stop 模式时产生 Reset 1：无 Reset 产生
12	IWDG_SW	RW		0：硬件 watchdog 1：软件 watchdog
11	BFB	RW		0：从 Bank0 启动(Bank0 映射在 0x0800 0000 Bank1 映射在 0x0804 0000) 1：从 Bank1 启动(Bank1 映射在 0x0800 0000 Bank0 映射在 0x0804 0000)
10	IWDG_STDBY	RW		设置 IWDG 在 Standby 模式下定时器运行状态 0：freeze 定时器 1：正常运行
9	IWDG_STOP	RW		设置 IWDG 在 Stop 模式下定时器运行状态 0：freeze 定时器 1：正常运行
8	Res			
7:0	RDP	RW		0xAA：level 0，读保护无效 非 0xAA：level 1，读保护有效

4.12.8 Flash 选项 2 寄存器(FLASH_OPTR2)

Address offset:0x24

Reset value：0x0000 XXXX。

在上电复位（POR/OBL_LAUNCH）释放后,从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_LOCK	nBOOT0	nSWBOOT0	CCMSRAM_RST	SRAM_PE	NRST_MODE		nBOOT1	Res							
RW	RW	RW	RW	RW	RW	RW	RW	-							

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15	BOOT_LOCK	RW		用于强制从用户 Flash 区域启动 0：基于 pad/选项位配置启动 1：从主闪存强制启动
14	nBOOT0	RW		nBOOT0 选项位 0:nBOOT0=0 1:nBOOT0=1
13	nSWBOOT0	RW		软件 BOOT0 0:BOOT0 取自选项位 nBOOT0 1:BOOT0 取自 PB8/BOOT0 引脚
12	CCMSRAM_RST	RW		系统复位时 CCM SRAM 擦除: 0:系统复位发生时 CCM SRAM 擦除 1:系统复位发生时 CCM SRAM 不擦除
11	SRAM_PE	RW		SRAM1 和 CCM SRAM 奇偶校验许可位 0：许可 1：禁止
10:9	NRST_MODE	RW		PF5 pad mode 00：Reset 输入/输出 01：Reset 输入 10：GPIO 11：Reset 输入/输出
8	nBOOT1	RW		启动配置与 BOOT0 引脚决定从闪存主存储器、SRAM1 或系统存储器中选择启动模式。 请参阅第 3.7 节：启动配置。
7:0	保留	-	-	-

4.12.9 Flash WRP 地址寄存器(FLASH_WRPR)

Address offset:0x2C

Reset value：0x0000 00XX。

在上电复位（POR/OBL_LAUNCH）释放后，从 Flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Reset Value	Function
31：8	保留	-	-	-
7：0	WRP[7:0]	RW		Write protection，每一位对应一个 block 0：block N，写保护有效 1：block N，写保护无效 N=0-7

4.12.10 Flash PCROP0 起始地址寄存器（FLASH_PCROP0SR)

Address offset:0x30

Reset value：0x0000 0XXX。

在上电复位（POR/OBL_LAUNCH）释放后，从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP0SR[8:0]								
-							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	保留	-	-	-
8：0	PCROP0SR	RW		BANK0 PCROP 区域起始地址(以 Page 为单位)

4.12.11 Flash PCROP0 结束地址寄存器（FLASH_PCROP0ER)

Address offset:0x34

Reset value：0x0000 0XXX。

在上电复位（POR/OBL_LAUNCH）释放后，从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP0ER[8:0]								
-							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	保留	-	-	-
8：0	PCROP0ER	RW		BANK0 PCROP 区域结束地址(以 Page 为单位)

4.12.12 Flash PCROP1 起始地址寄存器（FLASH_PCROP1SR)

Address offset:0x38

Reset value：0x0000 0XXX。

在上电复位（POR/OBL_LAUNCH）释放后，从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1SR[8:0]								
-							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	保留	-	-	-
8：0	PCROP1SR	RW		BANK1 PCROP 区域起始地址(以 Page 为单位)

4.12.13 Flash PCROP1 结束地址寄存器（FLASH_PCROP1ER)

Address offset:0x3C

Reset value：0x0000 0XXX。

在上电复位（POR/OBL_LAUNCH）释放后，从 flash information memory 的选项字节区域读出相应的值，写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP1ER[8:0]								
-							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：9	保留	-	-	-
8：0	PCROP1ER	RW		BANK1 PCROP 区域结束地址(以 Page 为单位)

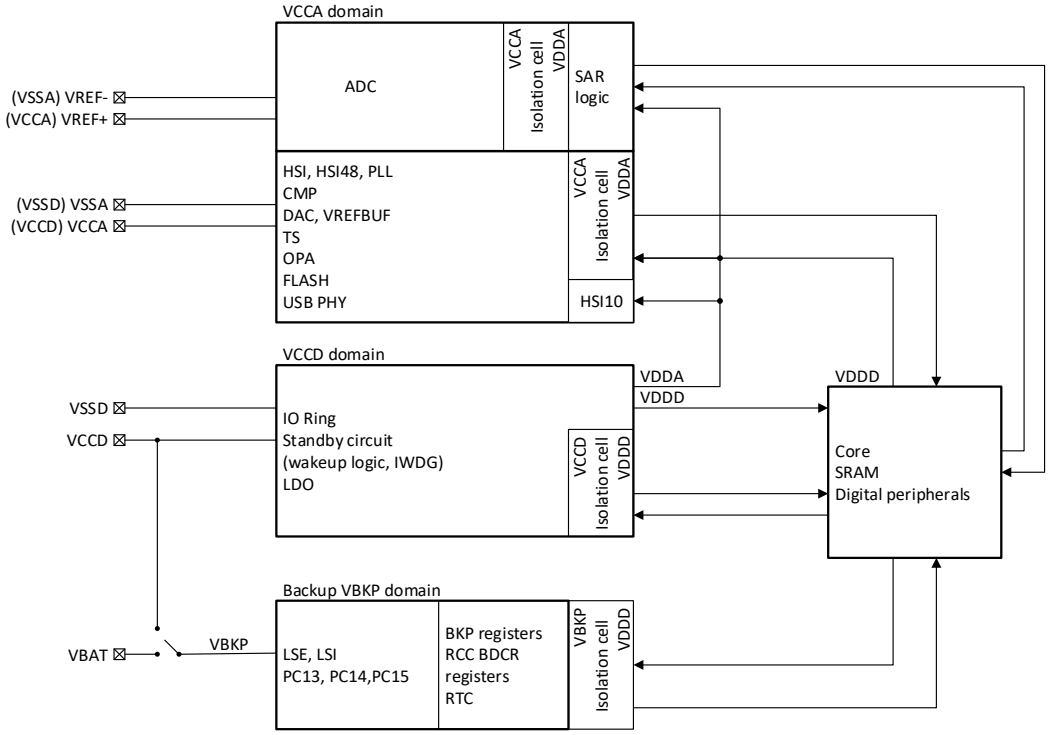
5. 电源控制（PWR）

5.1 简介

PWR 模块主要完成如下功能：

- 芯片上电和掉电流程控制
- 模拟 PMU 控制信号产生
- 存储器低功耗控制信号产生
- 低功耗模式进入和唤醒时序控制
- PVD 功能

5.2 电源结构



- 芯片工作电压为 2.3 ~ 3.6 V
- 全部低功耗进入和退出逻辑，都在 V_{CCD} 电源域实现。
- V_{BKP} 域逻辑有独立的上电复位 BPOR
- V_{DDD} 域数字逻辑的上电复位由 V_{CCD} 模块产生
- 使用 HSI16 产生的 V_{CC} 域时钟作为待机模式进入和唤醒逻辑的时钟，HSI_ON 需要为 V_{CCD} 域信号
- 待机模式下，可以配置使能 IWDG 工作，其计数时钟为 LSI，所以 LSI 时钟需要在待机模式下输出，即 LSI 时钟也会输出 V_{CCD} 域时钟。

5.2.1 系统主要电源

5.2.1.1 独立 A/D 转换器电源和参考电压

为了提高转换精度，ADC 配有独立电源，可以单独滤波并屏蔽 PCB 上的噪声。

- ADC 电压源从单独的 V_{CCA} 引脚输入。
- V_{SSA} 引脚提供了独立的电源接地连接。

为了提高低压输入的精度，用户可以在 V_{REF+} 上连接单独的 ADC 外部参考电压输入。V_{REF+}上的电压范围为 2.3 V 到 V_{CCA}。

5.2.1.2 备份电池域

要在 V_{CCD} 断电后保留 RTC 备份寄存器的内容并为 RTC 供电，可以将 V_{BAT} 引脚连接到电池或其他备用电源上。

要使 RTC 即使在主数字电源（V_{CCD}）关闭后仍然工作，V_{BAT}引脚需为以下各模块供电：

- RCC BDCR 寄存器
- RTC 模块(RTC_CR 寄存器除外)
- LSE 振荡器
- BKP 寄存器
- PC13 到 PC15 I/O

V_{BAT} 电源的开关由复位模块中内置的掉电复位电路（BPOR）进行控制。

注意：

- 在 $t_{RSTTEMPO}$（V_{CC}启动时间)或检测到 PDR 后，V_{BAT}和 V_{CCD}之间的电源开关仍然连接到 V_{BAT}。
- 在启动阶段，如果建立的 V_{CCD}小于 $t_{RSTTEMPO}$(参见数据表中的 $t_{RSTTEMPO}$值)和 $V_{CCD} > V_{BAT} + 0.6$ V，则可以通过连接 V_{CCD}和电源开关(V_{BAT})之间的内部二极管向 V_{BAT}注入电流。

如果连接到 V_{BAT}引脚的电源/电池不能支持此电流注入，强烈建议在此电源和 V_{BAT}引脚之间连接一个二极管。

如果应用中未使用任何外部电池，建议将 V_{BAT} 引脚连接到 V_{CCD}，V_{CCD} 上需并联 100 nF 去耦陶瓷电容。

通过 V_{CCD}对备份域供电时（模拟开关连接到 V_{CCD}）：

- PC14 和 PC15 可用作 GPIO 或 LSE 引脚
- PC13 可用作 GPIO，也可配置为其他功能 (TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出)

注：由于该开关的灌电流能力有限（3 mA)，因此在输出模式下使用 GPIO PC13 到 PC15 时存在以下限制: 速率不得超过 2 MHz，最大负载为 30 pF，并且这些 I/O 不能用作电流源 (如用于驱动 LED)。

通过 V_{BAT} 对备份域供电时（由于不存在 V_{CCD}，内部电源开关连接到 V_{BAT}），可实现以下引脚：

- PC14 和 PC15 只可用作 LSE 引脚
- PC13 可用作 RTC 附加功能引脚（TAMPER 引脚、RTC 闹钟或秒输出）

备份域访问

复位后，备份域（RTC 寄存器和 RTC 备份寄存器）将受到保护，以防止意外的写访问。要使能对备份域（RTC 和 RTC 备份寄存器）的访问，需要按如下步骤配置：

- 将 RCC_APB1ENR1 寄存器中的 PWREN 位置 1，使能电源接口时钟
- 将 PWR_CR1 寄存器中的 DBP 位置 1，使能对备份域的访问
- 配置 RCC 备份域控制寄存器（RCC_BDCR）中的 RTCSEL 寄存器选择 RTC 时钟源

- 配置 RCC 备份域控制寄存器（RCC_BDCR）中的 RTCEN，使能 RTC 时钟

5.2.1.3 VDDD 调压器

嵌入式线性 VR 为备份域和待机（唤醒）电路以外的所有数字电路供电。

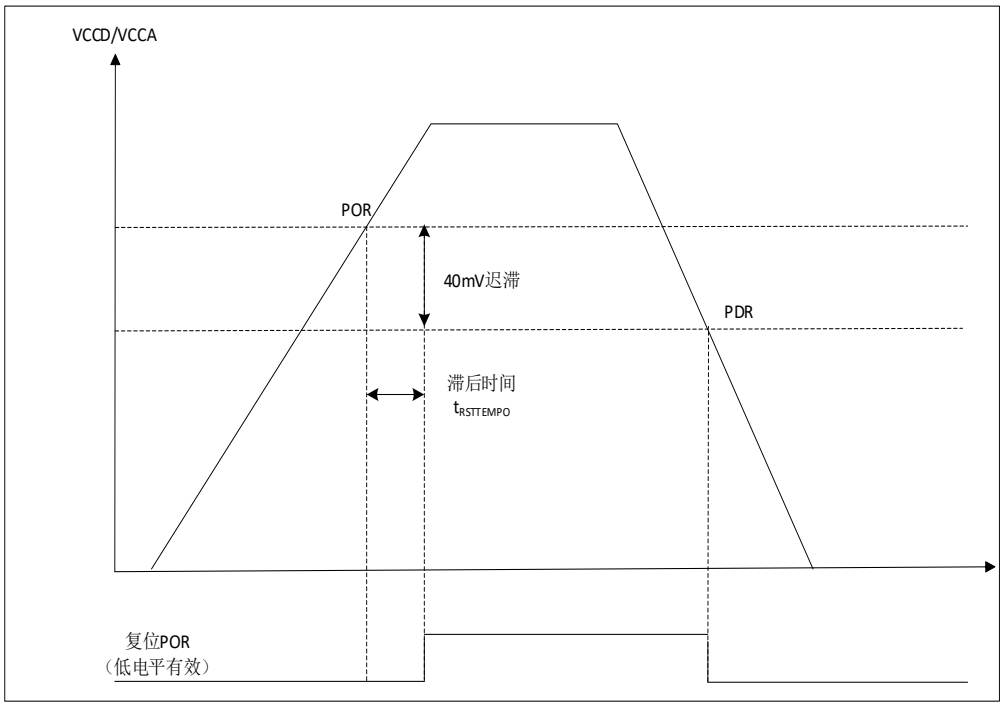
通过软件激活时，调压器在复位后始终处于使能状态。根据应用模式的不同，可采用三种不同的模式工作：

- 在 Run 模式中，调压器为内核、存储器和数字外设提供全功率。在该模式下，调压器输出电压可通过软件调节为不同的电压值（可通过 PWR_CR 寄存器的 VOS[1:0]位配置）。
- 在停止模式下，主调压器或低功耗调压器为 V_{DDD} 域提供低功率电压，从而保存寄存器和内部 SRAM 的内容。还可以将调压器置于主调压器模式（MR）或低功耗模式（LPR)。
- 在待机模式中，调压器掉电。除待机（唤醒）电路和备份域外，寄存器，SRAM1 和 CCM SRAM 的内容都将丢失，SRAM2 的内容可选择是否保留（由 PWR_CR3 的 RRS 位控制）。

5.2.2 电源检测

5.2.2.1 POR/PDR

芯片包含 Power-on reset（POR）、Power-off reset（PDR）模块，提供电源相关复位。其中 POR/PDR 在所有 power 模式下都默认工作。POR/PDR 输出复位低电平有效。



5.2.2.2 PVD

PVD 模块是检测 V_{CC} 是否低于 PWR_CR2.PLS 设置的阈值。

通过配置 PWR_CR2.PVDE 寄存器使能 PVD 功能。

电源控制/状态寄存器（PWR_CSR）中的 PVDO 标志用来表明 V_{CC} 是高于还是低于 PVD 的电压阀值。该事件在内部连接到 EXTI 的 Line16，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 V_{CC} 下降到 PVD 阀值以下或当 V_{CC} 上升到 PVD 阀值之上时，根据 EXTI Line16 的上升/下降沿触发设置，就会产生 PVD 中断。实际应用中，这一特性可用于执行紧急关闭任务。

5.3 系统低功耗模式

默认情况下，系统复位或上电复位后，微控制器进入运行模式。在运行模式下，CPU 通过 HCLK 提供时钟，并执行程序代码。系统提供了多个低功耗模式，可在 CPU 不需要运行时（例如等待外部事件时）节省功耗。由用户根据应用选择具体的低功耗模式，在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。

器件有五种低功耗模式：

- 睡眠模式（Sleep mode）：内核停止运行，内核的外设（NVIC，Sys Tick）等可以配置为运行;
- 低功耗运行模式（Low-power run）：系统时钟配置为 HSI16 并且 HCLK 8 分频以下（即 HCLK 不大于 2MHz），V_{DDD} 由 LPR 供电;
- 低功耗睡眠模式（Low-power sleep）：从 Low-power run 进入 Sleep 模式;
- 停止模式（Stop mode）：V_{DDD} 域所有高速时钟关闭，SRAM 和寄存器内容保持。LSI 和 LSE 可以根据唤醒源选择是否开启。可以配置进入 STOP 时的 LDO 工作模式，支持 MR 和 LPR 两种配置。这两种种模式功耗依次降低，唤醒时间依次变长。
- 待机模式（Standby mode）：V_{DDD} 域断电，仅 V_{CCD} 和 VBKP 域工作。可通过配置（PWR_CR3.RRS），选择 SRAM2 是否保电；LSI 和 LSE 可以根据唤醒源选择是否开启。

芯片存在 V_{BAT} 电源，所以在 V_{CC} 掉电时，芯片仅 V_{BKP} 域工作。

此外，可通过下列方法之一降低运行模式的功耗：

- 降低系统时钟速度
- 不使用 APBx 和 AHBx 外设时，将对应的外设时钟关闭。

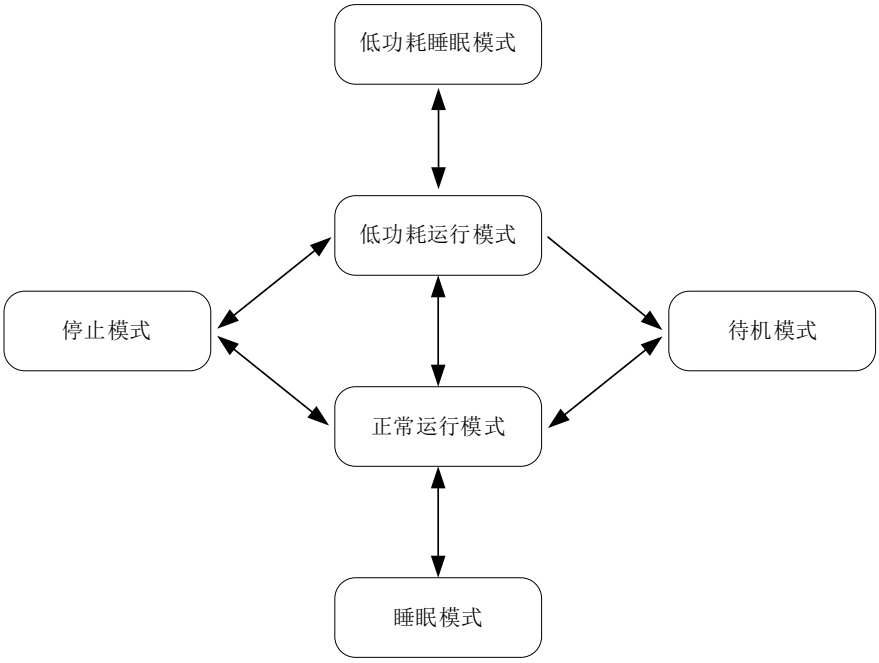


图 5-1 低功耗模式状态转移图

表 5-1 低功耗模式概要

模式	时钟状态	进入条件	退出条件	唤醒延时	唤醒后系统时钟	MR 状态	LPR 状态
Sleep	CPU 时钟关闭	WFI 或者从中断返回 SLEEPDEEP = 0	任意中断	6 个系统时钟 (1)	进入前时钟	ON	OFF
		WFE	唤醒事件				
Low-power run	系统时钟:HSI 4 分频	LPR=1	LPR=0	2us(2)		OFF	ON
Low-power Sleep	CPU 时钟关闭	WFI 或者从中断返回 SLEEPDEEP = 0 LPR=1	任意中断	6 个系统时钟 (1)			
		WFE LPR=1	唤醒事件				
Stop0	HSI/LSI/LSE 可以选择打开 其他时钟关闭	LPMS=00 SLEEPDEEP=1 WFI 或者从中断返回 或者 WFE	任意 EXTI 特定外设唤醒事件	5.5us(3)		ON	OFF
Stop1	除 LSE,LSI 外其他时钟关闭	LPMS=01 SLEEPDEEP=1 WFI 或者从中断返回 或者 WFE		5.5/7.5us (4)		OFF	ON
SRAM 保持	除 LSE,LSI 外其他时钟关闭	LPMS=11 SLEEPDEEP=1 RRS=1 WFI 或者从中断返回 或者 WFE	WKUP IO RTC TAMP 外部复位 PIN 复位 IWDG 复位	37us (5)	HSI	OFF	OFF
SRAM 不保持	除 LSE,LSI 外其他时钟关闭	LPMS=11 SLEEPDEEP=1 RRS=0 WFI 或者从中断返回 或者 WFE				OFF	OFF

1. 系统时钟为进入 SLEEP 前配置的 sys_clk（参考 RCC 模块）周期。
2. MR 稳定时间约为 2 us;
3. 包括 HSI16 稳定时间和存储器（Flash）低功耗退出稳定时间;

4. 有两种配置，HSI16 和 MR 同时开启（5.5 us）和 HSI16 等待 MR 稳定后再开启(7.5 us)。
5. 包括 MR 启动到稳定时间 10 us，HSI16 稳定时间 2 us，Flash 稳定时间 5.1 us 以及上电内部校准数据加载时间 20 us.

以上皆为仿真值，实际值预估小于上述时间（具体参数请参考数据手册）

下表为不同模式下，各个功能模块是否存在的情况：

Peripheral	Run	Sleep	LP Run	LP Sleep	Stop		Standby		V_{BAT}
					-	唤醒	-	唤醒	
CPU Core	Y	-	Y	-	-	-	-	-	-
Flash memory	Y	Y(2)	Y	Y(2)	-	-	-	-	-
SRAM1	Y	Y(3)	Y	Y(3)	Y	-	-	-	-
SRAM2	Y	Y(3)	Y	Y(3)	Y	-	O(4)	-	-
CCM SRAM	Y	Y(3)	Y	Y(3)	Y	-	-	-	-
ESMC	O	O	O	O	-	-	-	-	-
BKP	Y	Y	Y	Y	Y	-	Y	-	Y
PVD	O	O	O	O	O	O	-	-	-
DMA	O	O	O	O	-	-	-	-	-
HSI16	O	O	O	O	O(5)	-	-	-	-
HSI48	O	O	-	-	-	-	-	-	-
HSE	O	O	O	O	-	-	-	-	-
LSI	O	O	O	O	O	-	O	-	O
LSE	O	O	O	O	O	-	O	-	O
PLL	O	O	-	-	-	-	-	-	-
HSE Clock Security System (CSS)	O	O	O	O	-	-	-	-	-
RTC/Auto Wakeup	O	O	O	O	O	O	O	O	O
Number of RTC TAMP pin	1	1	1	1	1	O	1	O	1
USBx(x=1,2)	O	O	-	-	-	O	-	-	-
Ethernet MAC	O	O	-	-	-	O	-	-	-
USARTx（x=1,2,3）	O	O	O	O	-	-	-	-	-
UARTx（x=1,2,3）	O	O	O	O	-	-	-	-	-
LPUART1	O	O	O	O	O(6)	O(6)	-	-	-
I2Cx(x=1,2,3,4)	O	O	O	O	O(7)	O(7)	-	-	-
SPIx(x=1,2,3)	O	O	O	O	-	-	-	-	-
CANx（x=1,2）	O	O	O	O	-	-	-	-	-
ADCx(x=1,2,3)	O	O	O	O	-	-	-	-	-
DACx(x=1,2)	O	O	O	O	-	-	-	-	-
VREFBUF	O	O	O	O	-	-	-	-	-
OPAx(x=1,2,3)	O	O	O	O	-	-	-	-	-
COMPx(x=1,2,3,4)	O	O	O	O	O	O	-	-	-
Temperature sensor	O	O	O	O	-	-	-	-	-
Timers	O	O	O	O	-	-	-	-	-

LPTIM	O	O	O	O	O	O	-	-	-
IWDG	O	O	O	O	O	O	O	O	-
WWDG	O	O	O	O	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-
RNG	O	O	O	O	-	-	-	-	-
AES	O	O	O	O	-	-	-	-	-
CORDIC	O	O	O	O	-	-	-	-	-
SDIO	O	O	O	O	-	-	-	-	-
CRC	O	O	O	O	-	-	-	-	-
GPIOs	O	O	O	O	O	O	O(8)	O(9)	-

注：

1. Y = YES(ENABLE)；O= Optional(默认关闭，可以软件使能)； - = not available
2. Flash 在 Sleep 和 LP_Sleep 模式下，可由软件选择时钟是否打开
3. SRAM 在 Sleep 和 LP_Sleep 模式下，可由软件选择时钟是否打开
4. SRAM2 可以被保持
5. 在 Stop 模式下，HSI16 可由软件控制开关
6. LPUART 在 Stop 模式下可以接收，产生 START 和地址匹配中断唤醒
7. I^2C 在 Stop 模式下提供地址匹配和超时中断唤醒
8. 在 Standby 模式下，IO 可以配置为上拉下拉或者浮空
9. Standby 模式下具有唤醒功能的 IO:PA0 PA2 PC5 PC13 PE6

5.3.1 正常运行模式

在正常运行模式下，可以通过降低系统时钟（SYSCLK，HCLK，PCLK）来降低功耗；在进入 SLEEP 模式前，也可以通过分频来降低外设时钟频率；

在正常运行模式下，不同的外设和存储器，都可以通过门控单独关闭时钟，达到降低功耗作用；在进入 SLEEP 前，可以关闭不是用作唤醒的外设时钟；

5.3.2 低功耗运行模式

为了进一步降低运行模式下的功耗，可以关闭 MR，V_{DDD} 域由 LPR 供电。同时降低系统的工作频率，不能超过 2 MHz；

低功耗运行模式下的 IO 状态

在低功耗运行模式下，所有 IO 状态保持和运行状态一致。

低功耗运行模式的进入

1. 可选：程序跳转到 SRAM 运行，同时配置 Flash 进入低功耗模式；
2. 降低系统时钟，要低于 2 MHz；
3. 设置 V_{DDD} 使用 LPR 供电，MR 关闭。

低功耗运行模式的退出

1. 设置 V_{DDD} 使用 MR 供电；
2. 等待 MR 上电完成；
3. 恢复正常的系统时钟。

低功耗运行	描述
模式进入	降低系统时钟低于 2 MHz LPR=1
模式退出	LPR=0 等待 MR 上电完成 恢复正常时钟
唤醒延时	MR 开启到稳定时间

5.3.3 低功耗模式进入和退出

进入：

通过执行 WFI（等待中断）或 WFE（等待事件）指令进入低功耗模式，也可通过在从 ISR 返回时系统控制寄存器中的 SLEEPONEXIT 位置 1 进入低功耗模式。

仅当没有中断或事件挂起时，才可通过 WFI 或 WFE 进入低功耗模式，即 CPU 在进入低功耗前需要清零挂起中断或者事件。

在进入低功耗模式的进程中，出现唤醒信号，系统会根据当前状态判断如何处理：

1. 在系统时钟关闭之前，退出进入流程，处理唤醒中断或事件；
2. 在系统时钟关闭之后，先保持住唤醒信号，再正常进入低功耗模式，进入低功耗模式后，再按正常唤醒流程唤醒。

退出：

根据进入睡眠和停止模式的方式退出这两种低功耗模式：

- 如果使用 WFI 指令或从 ISR 返回进入低功耗模式，通过 NVIC 确认的任何外设中断都能唤醒器件。
- 如果使用 WFE 指令进入低功耗模式，MCU 将在有事件发生时立即退出低功耗模式。唤醒事件可通过以下方式产生：

  — NVIC IRQ 中断：

  SEVONPEND = 0 时，通过使能 NVIC 和对应的外设的中断使能控制位，可以从 WFE 唤醒。当系统唤醒后，必须清除 NVIC 和外设中的中断挂起位；

  SEVONPEND = 1 时，只用使能外设的中断使能控制位，NVIC 中的中断使能可以选择是否使能，就可以从 WFE 唤醒。当系统唤醒后，必须清除外设中的中断挂起位，如果 NVIC 中的中断使能，也要同时清除；

  所有的 NVIC 中断都可以唤醒 CPU，NVIC 中没有使能的也可以。

  — 事件：

  配置一个外部或内部的 EXTI 线为事件模式。当 CPU 从 WFE 恢复时，因为对应事件线的挂起位没有被置 1，不必清零 EXTI 外设的中断挂起位或 NVIC IRQ 通道挂起位。可能需要清零外设中的中断标志。该模式唤醒所需的时间最短，因为没有时间花费在中断的进入或退出上。

  当发生外部复位（NRST 引脚）、IWDG 复位、使能的 WKUPx 引脚上出现上升沿或者触发 RTC 唤醒事件时，MCU 退出待机低功耗模式。

  从待机模式唤醒后，程序将按照复位（启动 boot 引脚采样、复位向量获取等）后的方式重新执行。

5.3.4 睡眠模式

进入睡眠模式之前，必须清零所有的中断挂起位。

睡眠模式下的 IO 状态

在睡眠模式下，所有 IO 状态保持和运行状态一致。

睡眠模式	描述
模式进入	WFI 或 WFE，并且： --SLEEPDEEP=0 以及 --没有中断挂起。
	从最低优先级的 ISR 返回，并且： --SLEEPDEEP=0 以及 --SLEEPONEXIT=1 --没有中断挂起。
模式退出	使用 WFI 或从最低优先级 ISR 返回进入：中断（使能） 使用 WFE 进入且 SEVONPEN=0：唤醒事件 使用 WFE 进入且 SEVONPEN=1：外设使能中断（即使 NVIC 中禁止）
唤醒延时	无

5.3.5 低功耗睡眠模式

低功耗睡眠模式下的 IO 状态

在低功耗睡眠模式下，所有 IO 状态保持和运行状态一致。

低功耗睡眠	描述
模式进入	低功耗睡眠模式只能从低功耗运行模式进入 WFI 或 WFE，并且： --SLEEPDEEP=0 以及 --没有中断挂起。
	低功耗睡眠模式只能从低功耗运行模式进入 从最低优先级的 ISR 返回，并且： --SLEEPDEEP=0 以及 --没有中断挂起。
模式退出	使用 WFI 或从最低优先级 ISR 返回进入：中断（使能） 使用 WFE 进入且 SEVONPEN=0：唤醒事件 使用 WFE 进入且 SEVONPEN=1：外设使能中断（即使 NVIC 中禁止） 从低功耗睡眠模式退出，系统进入低功耗运行模式。
唤醒延时	无

5.3.6 停止模式

停止模式基于内核 SleepDeep 模式与外设时钟门控。V_{DDD} 域 LDO 既可以配置为正常模式（MR），也可以配置为低功耗模式（LPR）。在停止模式下，V_{DDD} 域的所有高频时钟都会停止，停止的时钟包括 PLL、HSI16、HSI48 和 HSE，其中 PLL、HSI48 和 HSE 由软件控制关闭，HSI16 由硬件关闭。内部 SRAM 和寄存器内容将保留。

停止模式下的 IO 状态

在停止模式下，所有 IO 状态保持和运行状态一致。

停止模式的进入

如果正在执行 FLASH 编程，停止模式的进入将延迟到存储器访问结束后执行（由 FLASH 控制器接口的 HREADY 信号以及操作标志控制，软件可以在结束后再执行 WFI 或者 WFE 指令）。

进入停止模式前，如果正在访问 APB 域，停止模式的进入将延迟到 APB 访问结束。（通过软件保证）

在停止模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗（IWDG）：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。
- 实时时钟（RTC）：通过 RCC 备份域控制寄存器（RCC_BDCR）中的 RTCEN 位进行配置。
- 内部 RC 振荡器（LSI RC）：通过 RCC 时钟控制和状态寄存器（RCC_CSR)中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器（LSE OSC）：通过 RCC 备份域控制寄存器(RCC_BDCR）中的 LSEON 位进行配置。

在 Stop 模式下，一些低功耗外设也可以配置为唤醒源，例如 LPUART 和 I^2C；

在进入停止模式前，一些模拟模块（ADC，DAC）也可以工作，为了进一步降低功耗，进入前请关闭。

停止模式的退出

退出停止模式，将选择 HSI16 作为系统时钟。如果配置为退出停止模式后，进入低功耗运行模式，需要在进入停止模式前，将系统时钟配置为低于 2 MHz；同时选择 V_{DDD} 的供电方式 MR（STOP0）或者 LPR（STOP1）。

在进入 Stop 时，根据配置的 V_{DDD} 供电模式不同，启动后需要的延时不同。

停止模式	描述
模式进入	WFI 或 WFE，并且： --没有中断（针对 WFI）或者事件（针对 WFE）挂起； --SLEEPDEEP=1； --将 PWR_CR 寄存器中的 LPMS 配置为 00（STOP0）或者 01（STOP1）
	从最低优先级的 ISR 返回，并且： --SLEEPDEEP=1； --没有中断挂起； --将 PWR_CR 寄存器中的 LPMS 配置为 00（STOP0）或者 01（STOP1）
模式退出	使用 WFI 或从 ISR 返回进入：任何配置为中断模式的 EXTI line（同时使能 NVIC 中对应的 EXTI 中断）。中断源为外部中断或者具有唤醒功能的外设产生的中断。 使用 WFE 进入且 SEVONPEND=0：任何配置为事件模式的 EXTI line。 使用 WFE 进入且 SEVONPEND=1： --任何配置为中断模式的 EXTI line（即使在 NVIC 中断对应的 EXTI 中断禁止）。中断源为外部中断或者具有唤醒功能的外设产生的中断。 --唤醒事件 NRST 复位 IWDG 复位
唤醒延时	HSI16 唤醒时间+（LDO 从低功耗模式退出时间（LDO 不同模式时间不同））+FLASH 唤醒的时间（根据配置时间可有重合）

5.3.7 待机模式

待机模式下，VR（包括 MR 和 LPR）停止工作，V_{DDD} 域断电。PLL，HSI16，HSI48，HSE 都关闭，其中 PLL、HSI48 和 HSE 由软件控制关闭，HSI16 由硬件关闭。除 V_{BKP} 域（RTC 寄存器和备份寄存器）和待机电路的寄存器外，其他 SRAM1 和 CCM SRAM 和寄存器内容都会丢失，根据配置 SRAM2 可选择是否由 LPR 单独供电。

待机模式下的 IO 状态

在待机模式，IO 可以配置为上拉下拉，或者模拟模式；

RTC 的输出口 PC13，LSE 的两个端口 PC14 和 PC15，WKUP IO 和 TAMPER IO，都处于工作状态；

待机模式的进入

在待机模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗（IWDG）：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。
- 实时时钟（RTC）：通过 RCC 备份域控制寄存器（RCC_BDCR）中的 RTCEN 位进行配置。
- 内部 RC 振荡器（LSI RC）：通过 RCC 时钟控制和状态寄存器（RCC_CSR)中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器（LSE OSC）：通过 RCC 备份域控制寄存器(RCC_BDCR）中的 LSEON 位进行配置。

待机模式	描述
模式进入	WFI 或 WFE，并且： --没有中断（针对 WFI）或者事件（针对 WFE）挂起； --SLEEPDEEP=1； --将 PWR_CR 寄存器中的 LPMS 配置为 11； --配置 PWR_CSR 寄存器中的 WUF 位清零； --将与选择唤醒源对应的标志清零；
	从 ISR 返回，并且： --SLEEPDEEP=1； --SLEEPONEXIT=1； --没有中断挂起； --将 PWR_CR 寄存器中的 LPMS 配置为 11； --将 PWR_CSR 寄存器中的 WUF 位清零； --将与选择唤醒源对应的标志清零；
模式退出	WKUPx 引脚（x=1，2，3，4，5）
	RTC 唤醒事件产生
	NRST 引脚复位
	IWDG 复位
唤醒延迟	HSI16 唤醒时间+LDO 上电时间+LDO 等待时间（可配置 0us/5us/10us/20us）+FLASH 等待时间 3us（根据配置时间可有重合）+内部校准值加载时间

在待机模式下，除以下各部分外，其余 IO 都处于高阻态：

- 复位引脚

- 被设置为防侵入或者校准输出时的 TAMPER 引脚
- 5 个 WKUP 引脚（PA0/PC13/PE6/PA2/PC5）（如果使能）

5.3.8 停止模式和待机模式的调试

在停止和待机模式，默认情况下，调试功能将中断，因为 CPU 核无时钟。但如果配置了 DBGMCU_CR 中相关寄存器，则即使 CPU 进入 SLEEPDEEP 模式，仍可进行调试。

5.3.9 低功耗模式下的自动唤醒（AWU）

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 MCU(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_BDCR)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 32.768 kHz 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 1 μA)

- 低功耗内部 RC 振荡器(LSI)

使用该时钟源，节省了一个 32.768 kHz 晶振的成本。但是 RC 振荡器将少许增加电源消耗。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 EXTI17 为上升沿触发。
- 配置 RTC 使其可产生唤醒事件。

如果要从待机模式中唤醒，不必配置 EXTI17（直接使用 RTC 闹钟唤醒事件唤醒待机）。

该外设的寄存器可以通过 half-word 或者 word 访问。

5.4 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。

5.4.1 电源控制寄存器 1(PWR_CR1)(0x00)

Address offset：0x00

Reset value：0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.											STDBY_MRRDY_WAIT		FLS_WUPT		HSION_CTRL
-											RW		RW		RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	LPR	Res	SRAM_RETV		VOS		DBP	Res	Res	Res	Res	Res	Res	LPMS	
-	RW	-	RW		RW		RW	-						RW	

Bit	Name	R/W	Reset Value	Function
31:21	保留	-	-	-
20:19	STDBY_MRRDY_WAIT	RW	2'h0	待机模式唤醒，MR ready 后，需要等待的时间。 00：3us；01：5us；10：10us；11：20us
18:17	FLS_WUPT	RW	2'b00	停止模式唤醒时序中，在 HSI 稳定后，在 FLASH 操作前需要等待时间。 00：3us；01：5us；10：2us；11：0us；

16	HSION_CTRL	RW	1	从停止模式唤醒时，HSI 打开时间控制。 0：等待 MR 稳定后，使能 HSI； 1：与 VR 同时打开，即唤醒时立刻使能 HSI。
15	保留	-	-	-
14	LPR	RW	0	VR 工作模式选择； 0：工作在主模式（MR） 1：工作在低功耗模式（LPR）
13	保留	-	-	-
12:11	SRAM_RETV	RW	2’b00	用于控制 SRAM2 在 STANDBY 模式下的电压 SRAM_RETV=2b’00,高电压模式 SRAM_RETV=2b’01,中电压模式 SRAM_RETV=2b’10,中低电压模式 SRAM_RETV=2b’11,低电压模式
10:9	VOS	RW	2’b00	选择 VR 输出电压级别。 此位用来控制内部 VR 的输出电压，以便实现器件与功耗的平衡。 当 VR 为 LPR 模式时： 00：高电压模式； 01：中电压模式； 10：中低电压模式； 11：低电压模式 当 VR 为 MR 模式时： 0x：高电压模式； 1x：低电压模式
8	DBP	RW	0	Backup 域寄存器写保护。 0：不可访问 RTC 和备份寄存器； 1：可以访问 RTC 和备份寄存器；
1:0	LPMS	RW	0	低功耗模式选择 00：Stop0 （MR 工作）01：Stop1 （LPR 工作） 10：保留 11：STANDBY （VR 关闭）

5.4.2 电源控制寄存器 2(PWR_CR2)(0x04)

Address offset：0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	FLT_CTRL			FLTDIS	Res.	Res.	Res.	Res.	PLS			PVDE
-				RW			RW	-				RW			RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-

11:9	FLT_CTRL	RW	3’b000	PVD 滤波时间配置。 111：保留； 110：滤波时间大约为 1024 个滤波时钟（LSI 或者 LSE 时，大约为 30.7ms）； 101：滤波时间大约为 128 个滤波时钟（LSI 或者 LSE 时，大约为 3.8ms）； 100：滤波时间大约为 64 个滤波时钟（LSI 或者 LSE 时，大约为 1.92ms）； 011：滤波时间大约为 16 个滤波时钟（LSI 或者 LSE 时，大约为 480us）； 010：滤波时间大约为 4 个滤波时钟（LSI 或者 LSE 时，大约为 120us）； 001：滤波时间大约为 2 个滤波时钟（LSI 或者 LSE 时，大约为 60us）； 000：滤波时间大约为 1 个滤波时钟（LSI 或者 LSE 时，大约为 30us）；
8	FLTEN	RW	0	PVD 数字滤波使能。 0：PVD 数字滤波禁止 1：PVD 数字滤波使能
7:4	保留	-	-	-
3:1	PLS[2:0]	RW	0	PVD 电平选择。 由软件写入，用于选择 PVD 的电压阈值。 000：保留 001：保留 010：保留 011：VPVD3（around 2.4 V) 100：VPVD4（around 2.6 V) 101：VPVD5（around 2. 8 V) 110：VPVD6（around 3.0 V) 111：外部 IO 输入
0	PVDE	RW	0	PVD 使能 0：禁止 PVD 1：使能 PVD

5.4.3 电源控制寄存器 3(PWR_CR3)(0x08)

Address offset：0x08

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRS	Res.	Res.	Res.	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
-							RW	-			RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-

8	RRS	RW	0	SRAM2 在 STANDBY 模式下数据保持 0：SRAM2 在 STANDBY 模式下断电； 1：SRAM2 在 STANDBY 模式下由 LPR 供电，数据保持。
7:5	保留	-	-	-
4	EWUP5	RW	0	使能 WKUP5 引脚. 此位由软件置 1 和清零。 0：WKUP5 引脚用于通用 I/O。WKUP5 引脚上的事件不会从待机模式唤醒器件。 1：WKUP5 引脚用于从待机模式唤醒，唤醒沿由 WP5 决定。
3	EWUP4	RW	0	使能 WKUP4 引脚. 此位由软件置 1 和清零。 0：WKUP4 引脚用于通用 I/O。WKUP4 引脚上的事件不会从待机模式唤醒器件。 1：WKUP4 引脚用于从待机模式唤醒，唤醒沿由 WP4 决定。
2	EWUP3	RW	0	使能 WKUP3 引脚. 此位由软件置 1 和清零。 0：WKUP3 引脚用于通用 I/O。WKUP3 引脚上的事件不会从待机模式唤醒器件。 1：WKUP3 引脚用于从待机模式唤醒，唤醒沿由 WP3 决定。
1	EWUP2	RW	0	使能 WKUP2 引脚. 此位由软件置 1 和清零。 0：WKUP2 引脚用于通用 I/O。WKUP2 引脚上的事件不会从待机模式唤醒器件。 1：WKUP2 引脚用于从待机模式唤醒，唤醒沿由 WP2 决定。
0	EWUP1	RW	0	使能 WKUP1 引脚. 此位由软件置 1 和清零。 0：WKUP 引脚用于通用 I/O。WKUP 引脚上的事件不会从待机模式唤醒器件。 1：WKUP 引脚用于从待机模式唤醒，唤醒沿由 WP1 决定。

5.4.4 电源控制寄存器 4(PWR_CR4)(0x0C)

Address offset： 0x0C

Reset value： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WP5	WP4	WP3	WP2	WP1
-											RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	-	-
4	WP5	RW	0	WKUP5 引脚极性选择，此位由软件置 1 和清零。 0：WKUP 引脚上升沿唤醒； 1：禁止；
3	WP4	RW	0	WKUP4 引脚极性选择，此位由软件置 1 和清零。

				0：WKUP 引脚上升沿唤醒； 1：禁止；
2	WP3	RW	0	WKUP3 引脚极性选择，此位由软件置 1 和清零。 0：WKUP 引脚上升沿唤醒； 1：禁止；
1	WP2	RW	0	WKUP2 引脚极性选择，此位由软件置 1 和清零。 0：WKUP 引脚上升沿唤醒； 1：禁止；
0	WP1	RW	0	WKUP1 引脚极性选择，此位由软件置 1 和清零。 0：WKUP 引脚上升沿唤醒； 1：禁止；

5.4.5 电源状态寄存器(PWR_SR)(0x10)

Address offset： 0x10

Reset value： 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO
-															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MR_RDY	SBF	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
-						R	R	-			R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	PVDO	R	0	PVD 输出标志。 此位通过硬件置 1 和清零。仅当 PVD 工作时 此位才有效。 0：V_{CC} 高于 PLS[2:0] 位选择的 PVD 阈值。 1：V_{CC} 低于 PLS[2:0] 位选择的 PVD 阈值。 注：PVD 在进入待机模式时关闭。因此，待机模式后或复位后，此位等于 0。等到 PVD 使能后，根据检测电压阈值输出对应值。
15:10	保留	-	-	-
9	MR_RDY	R	1	用于指示 MR 工作状态 0：MR 关闭； 1：MR 工作；
8	SBF	R	0	待机模式标志。 此位由硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CSBF 位置 1 来实现。 0：器件未进入待机模式 1：器件已进入待机模式 注：待机模式唤醒后，需软件清除该位
7:6	保留	-	-	-
4	WUF5	R	0	WKUP5 唤醒标志。 该位通过硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CWUF5 位置 1 清零。 0：未发生唤醒事件 1：收到 WKUP5 唤醒事件

3	WUF4	R	0	WKUP4 唤醒标志。 该位通过硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CWUF4 位置 1 清零。 0：未发生唤醒事件 1：收到 WKUP4 唤醒事件
2	WUF3	R	0	WKUP3 唤醒标志。 该位通过硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CWUF3 位置 1 清零。 0：未发生唤醒事件 1：收到 WKUP3 唤醒事件
1	WUF2	R	0	WKUP2 唤醒标志。 该位通过硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CWUF2 位置 1 清零。 0：未发生唤醒事件 1：收到 WKUP2 唤醒事件
0	WUF1	R	0	WKUP1 唤醒标志。 该位通过硬件置 1，清零则只能通过 POR/PDR 或将 PWR_SCR 寄存器中的 CWUF1 位置 1 清零。 0：未发生唤醒事件 1：收到 WKUP1 唤醒事件

5.4.6 电源状态清除寄存器(PWR_SCR)(0x14)

Address offset： 0x14

Reset value： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res.	Res	Res	Res	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBF	Res.	Res.	Res.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
-							W	-			W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	-	-
8	CSBF	W	0	SBF 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 SBF 信号
7:6	保留	-	-	-
4	CWUF5	W	0	WKUP5 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 WUF5 信号
3	CWUF4	W	0	WKUP4 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 WUF4 信号
2	CWUF3	W	0	WKUP3 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 WUF3 信号
1	CWUF2	W	0	WKUP2 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 WUF2 信号
0	CWUF1	W	0	WKUP1 唤醒标志清除信号。 通过设置该位为 1，清除 PWR_SCR 中的 WUF1 信号

6. 复位与时钟控制（RCC）

6.1 简介

6.1.1 主要特征

本模块完成如下功能：

- 产生系统时钟和系统复位
- 产生各个模块时钟和复位
- 产生时钟源控制信号

6.2 系统复位

除了时钟控制寄存器 CSR 中的复位标志和备份域中的寄存器外，系统复位会将其他全部寄存器都复位为复位值。

只要发生以下事件之一，就会产生系统复位：

- NRST 引脚低电平（外部复位）
- 窗口看门狗计数结束（WWDG 复位）
- 独立看门狗计数结束（IWDG 复位）
- 软件复位（SYSRESETREQ 复位）
- 低功耗管理复位（NRST_STDBY/NRST_STOP）
- 选项字节加载复位

6.2.1 NRST 引脚（外部复位）

通过特定的选项位（NRST_MODE），NRST 引脚可配置为：

- 复位输入/输出（设备交付时的默认值）

复位输入/输出（设备交付时默认）引脚上的任何有效复位信号都会传播到设备内部逻辑，所有内部复位源都会通过脉冲发生器驱动到此引脚。GPIO 功能（PF5）不可用。脉冲发生器保证每个内部复位源在 NRST 引脚上输出的最小复位脉冲持续时间为 20μs。该模式在每次设备上电复位期间(直到加载选项字节)始终处于有效状态(独立于选项字节设置)。

- 复位输入

在此模式下，NRST 引脚上的任何有效复位信号都会传播到设备内部逻辑，设备内部产生的复位。在此配置中，GPIO 功能（PF5）不可用。

- GPIO

在此模式下，引脚可以用作 PF5 标准 GPIO。引脚的复位功能不可用。复位只能从设备内部复位源进行，并且不会传播到此引脚。

6.2.2 软件复位

可通过查看 RCC 时钟控制和状态寄存器（RCC_CSR）中的复位标志确定。要对器件进行软件复位，必须将 CPU 中断和复位控制寄存器中的 SYSRESETREQ 位置 1。

6.2.3 低功耗管理复位

引发低功耗管理复位的方式有两种：

- 进入待机模式时产生复位：

此复位的使能方式是清零用户选项字节中的 nRST_STDBY 位。且 PWR_CR.PDDS 寄存器配置为待机模式，进入低功耗模式时，器件将复位，而非进入待机模式。

- 进入停止模式时产生复位：

此复位的使能方式是清零用户选项字节中的 nRST_STOP 位。且 PWR_CR.PDDS 寄存器配置为停止模式，进入低功耗模式时，器件将复位，而非进入停止模式。

6.2.4 选项字节加载复位

当 FLASH_CR 寄存器中设置 OBL_LAUNCH 位（位 27）时，会生成选项字节加载复位。此位用于通过软件启动选项字节加载。

6.3 电源复位

只要发生以下事件之一，就会产生电源复位：

- 上电/掉电复位（POR/PDR 复位）
- 在退出待机模式时

除备份域外，电源复位会将所有寄存器复位。

复位入口向量在存储器映射中固定在地址 0x0000_0004。

6.4 备份域复位

备份域复位会将所有备份域寄存器复位，备份域寄存器包括 RCC_BDCR,备份寄存器以及 RTC 部分寄存器。

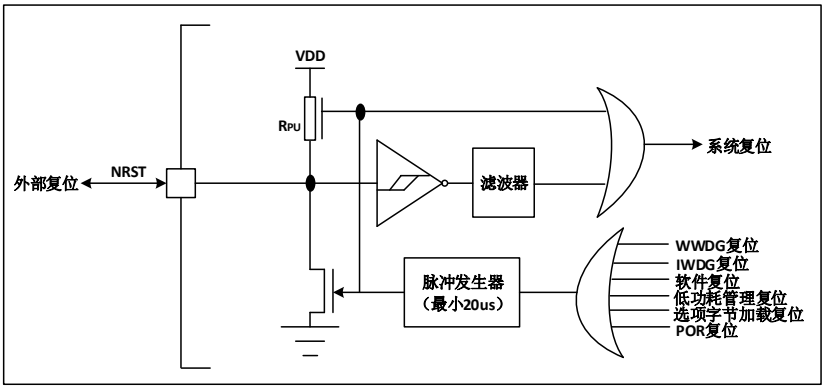
只要发生以下事件之一，就会产生备份域复位：

- 软件复位：通过将 RCC 备份域控制寄存器（RCC_BDCR）中的 BDRST 位置 1 触发。
- 在电源 V_{CC} 和 V_{BAT} 都已掉电后，其中任何一个又再上电。

6.5 备份域复位之外的复位的统一处理

除备份域复位外，NRST 引脚复位输入/输出模式下，其他源复位都作用于 NRST 引脚，该引脚在复位过程中时钟保持低电平。

电路如下图所示：



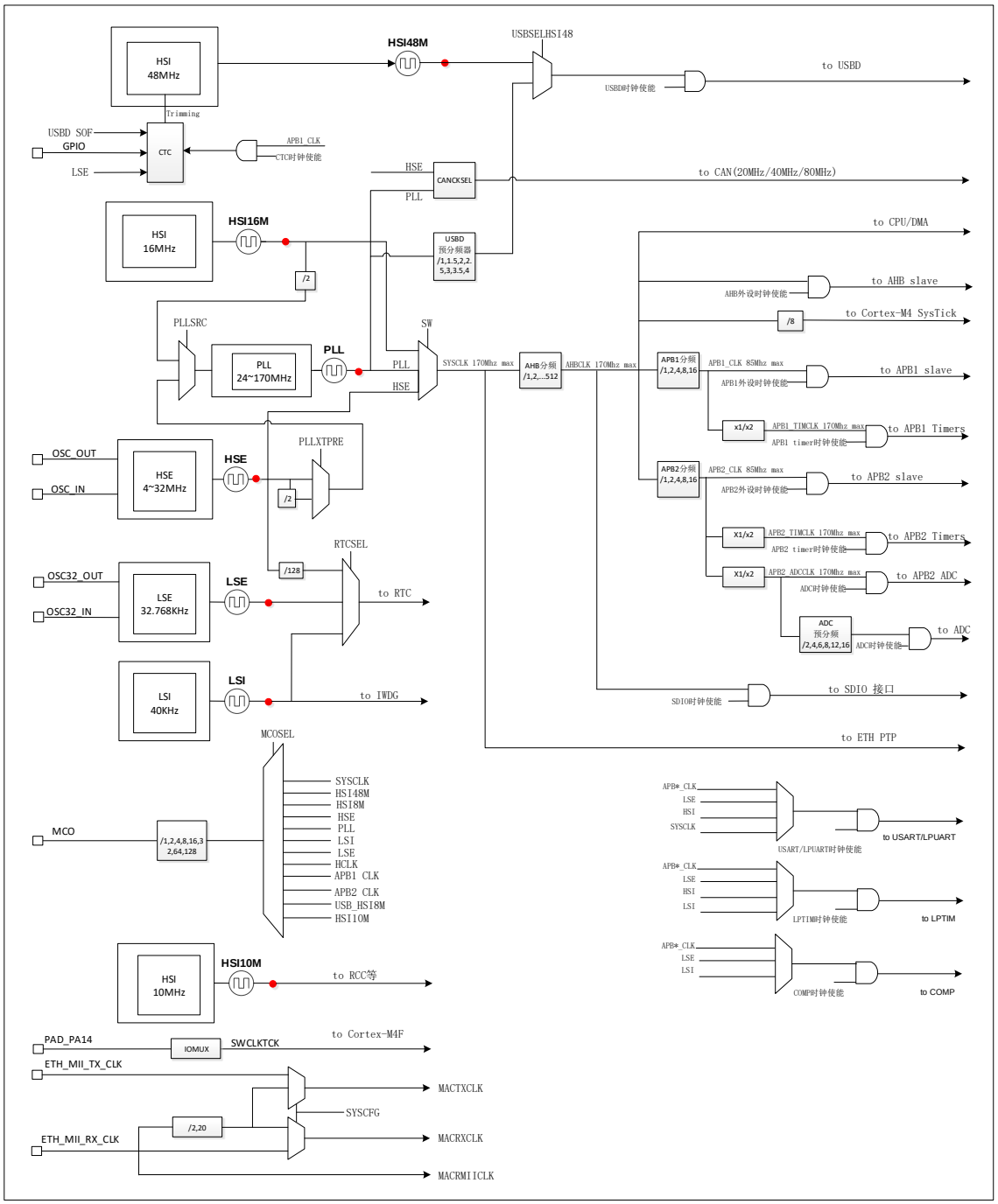
当系统发生内部复位时（高电平有效），脉冲发生器开始产生脉冲信号，并保证该脉冲信号持续至少 20 us。此脉冲信号使 N 管导通。N 管导通会不断拉低 NRST 引脚电压，当拉低到 VIL 时，NRST 引脚产生低电平信号。该信号经过施密特触发器和滤波后，产生系统复位（低电平有效）。

注 1：除外部 PIN 复位源之外的复位源，即脉冲发生器的源，不会经过 30 us 的滤波后产生系统复位；

注 2：脉冲发生器的复位源，在产生系统复位后，除了本身复位标志置位外，外部引脚复位标志也会置位。

6.6 时钟结构

时钟结构如下图所示：



6.7 时钟源

系统存在如下时钟源：

6.7.1 HSE 时钟

外部高频 OSC. 频率范围 4 ~ 32 MHz.
HSE 时钟的稳定时间由模拟 HSE 模块计数，模拟 HSE 模块在启动并且稳定后产生 HSE_RDY 信号。
HSE 时钟可以有两个工作模式：

- 外部 XTAL OSC+内部起振电路（HSEBYP=0）
- 通过 OSC_IN IO 输入的外部时钟（HSEBYP=1）

为保证 CLOCK 输出第一个时钟就为稳定时钟，HSE 内部包含一个计数器，计数完成后输出第一个时钟上升沿。HSEBYP=1 时计数周期是 HSEBYP=0 的一半。

6.7.2 HSI16 时钟

内部 16 MHz RC 振荡器。相较于 HSE，功耗低，稳定时间短，但精度低。模拟 HSI 模块在启动并且稳定后产生 HSI_RDY 信号。
上电复位后，在加载阶段，HSI 的校准值加载到 RCC_CR.HSICAL 寄存器。
从停止模式唤醒后，HSI 将作为系统时钟源。

6.7.3 HSI48 时钟

HSI48 时钟支持通过 CTC 模块自校准，HSI48 用做 USBD 模块的时钟。

6.7.4 PLL 时钟

PLL 模块用来倍频 HSI 或者 HSE 时钟，PLL 输入时钟频率范围为 2 ~ 32 MHz，输出时钟频率范围为 24 ~ 170 MHz。
PLL lockout 时间典型值为 25 us，最大值为 100 us。

6.7.5 LSE 时钟

外部 32.768 kHz OSC，用作低功耗时钟。
可以通过配置 LSEDRV 在稳定时间和功耗之间做平衡。LSE 稳定时间为 0.5 s（典型值）。LSE 稳定信号由 LSE 模拟模块产生。
与 HSE 来源类似，LSE 也有两个来源：

- 32.768 kHz XTAL+内部起振电路（LSEBYP=0）
- 通过 OSC32_IN 输入的外部时钟（LSEBYP=1）

为保证 CLOCK 输出第一个时钟就为稳定时钟，LSE 内部包含一个计数器，计数完成后输出第一个时钟上升沿。LSEBYP=1 时计数周期是 LSEBYP=0 的一半。

在 LSEBYP=1 情况下，MCO 输出的时钟为原始外灌时钟，且不会受 LSE_ON 信号的控制。但使用 LSE 时钟的逻辑会受 LSE_ON 信号控制。

6.7.6 LSI 时钟

内部低频 40 kHz 时钟。用作 IWDG、RTC 时钟。

6.7.7 HSI10M 时钟

该时钟作为低精度时钟，用作 NRST pin 的滤波计数。每次产生系统复位，经过脉冲发生器后，都通过 NRST pin 返回，所以每次复位产生时都会使能 HSI10M 时钟。

6.8 时钟安全系统（CSS）

时钟安全系统可以通过软件激活。在这种情况下，时钟检测器在 HSE 振荡器启动延迟后启用，并在该振荡器停止时禁用。

如果在 HSE 时钟上检测到故障，HSE 振荡器将被自动禁用，时钟故障事件将发送到定时器（TIM1/TIM8 和 TIM15/16/17）的中断输入，并生成一个中断以通知软件故障（时钟安全系统中断 CSSI），允许 MCU 执行救援操作。CSSI 通过 FPU NMI（不可屏蔽中断）异常向量链接到 Cortex®-M4F。

注：一旦启用 CSS，如果 HSE 时钟出现故障，则会发生 CSS 中断，并自动生成 NMI。除非 CSS 中断挂起位被清除，否则 NMI 将无限期执行。因此，在 NMI ISR 中，用户必须通过在时钟中断寄存器（RCC_CIR）中设置 CSSC 位来清除 CSS 中断

如果 HSE 振荡器直接或间接用作系统时钟（间接表示：它用作 PLL 输入时钟，PLL 时钟用作系统时钟），则检测到的故障会导致系统时钟切换到 HSI16 振荡器，并禁用 HSE 振荡器。如果 HSE 时钟是用作系统时钟的 PLL 的时钟源，则 PLL 也被禁用。

6.9 外设时钟使能寄存器

每个外围时钟可以由 RCC_AHBxENR、RCC_APBxENRy 寄存器的 xxxxEN 位启用。

当外围时钟未激活时，不支持外围寄存器的读或写访问。

使能位具有同步机制，在使能位被设置之后，在时钟被激活之前有 2 个时钟周期的延迟。

注意：在为外围设备启用时钟后，软件必须等待一段时间后才能访问外围寄存器。

6.10 寄存器描述

该模块寄存器可以按字节（8 位）[RCC_BDCR 除外]、半字（16 位）或字（32 位）访问。

6.10.1 时钟控制寄存器（RCC_CR）

Address offset：0x00

Reset value：0x0080_2083

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.						PLLRDY	PLLON	HSICAL[10:8]			Res.	CSSON	HSEBYP	HSERDY	HSEON
-						R	RW	R	R	R	-	RW	RW	R	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSICAL[7:0]								HSITRIM[4:0]					HSIKERON	HSIRDY	HSION
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	R	RW

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	-
25	PLLRDY	R	0	PLL 时钟 ready 标志。

				硬件置位，表明 PLL 时钟稳定。 0：PLL 没有 ready； 1：PLL ready；
24	PLLON	RW	0	PLL 使能。 当 PLL 时钟被用作（SWS 为 PLL）或将要被用作系统时钟（SW 选择 PLL）时，该位不能被清零。 0：PLL OFF； 1：PLL ON；
23:21	HSICAL[10:8]	R	3’h4	HSI16 时钟校准值高 3bit。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSICAL+HSITRIM，当 HSITRIM 值改变时，该寄存器值也会更新。
20	保留	-	-	-
19	CSSON	RW	0	HSE 时钟安全系统使能。 当该位为 1 时，如果 HSE OSC ready 则硬件会使能时钟检测模块；如果 HSE 检测失败，则关闭时钟检测模块。(该位只能写 1，系统复位清零) 0：时钟安全系统关闭（时钟检测关闭）； 1：时钟安全系统开启（如果 HSE 时钟稳定则时钟检测开启，否则关闭时钟检测）；
18	HSEBYP	RW	0	HSE 屏蔽晶振，选择管脚输入时钟。 该位只有当 HSEON=0 时才能写. 0：HSE 晶振不屏蔽，外部高速时钟选择晶振； 1：HSE 晶振屏蔽，外部高速时钟选择外部管脚输入时钟源；
17	HSERDY	R	0	HSE 晶振时钟 ready 标志。 该位由硬件置位表明 HSE 晶振稳定。 0：HSE 晶振没有 ready； 1：HSE 晶振 ready；
16	HSEON	RW	0	HSE 晶振使能。 当 HSE 直接或者间接作为系统时钟源时，该位不能被置 0. 0：HSE 晶振 OFF； 1：HSE 晶振 ON；
15:8	HSICAL[7:0]	R	8’h20	HSI16 时钟校准值。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSICAL+HSITRIM，当 HSITRIM 值改变时，该寄存器值也会更新。
7:3	HSITRIM[4:0]	RW	5’h10	HSI16 时钟 Trimming 值。 软件写入该寄存器调整 HSI16 时钟，叠加到 HSICAL 后输出给模拟 HSI16。 系统复位会复位该寄存器。
2	HSIKERON	RW	0	HSI16 作为内核时钟配置提供给外设 USART 和 I2C 等。保持 HSI16 在停止模式下打开可避免由于 HSI16 启动时间而降低通信速度。此位对 HSION 值没有影响。 0：对 HSI16 振荡器没有影响。 1：HSI16 振荡器即使在停止模式下也被强制使能。
1	HSIRDY	R	1	HSI16 时钟 ready 标志。

				硬件置位表明 HSI OSC 稳定。该位只有当 HSION=1 时才有效。 0：HSI16 OSC 没有 ready； 1：HSI16 OSC ready； 当 HSION 清零后，HSIRDY 将在 6 个 HSI16 时钟之后拉低。
0	HSION	RW	1	HSI16 时钟使能。 硬件在进入停止、待机模式时，会根据需要清零该寄存器停止 HSI16. 当 HSI16 直接或者间接作为系统时钟时，该寄存器不能为 0. 0：HSI16 OSC OFF； 1：HSI16 OSC ON； 硬件在如下情况会使能 HSI16： 1） 硬件从停止或者待机模式唤醒后； 2） HSE 直接/间接作为系统时钟但出现 HSE CSS fail；

6.10.2 时钟配置寄存器（RCC_CFGR）

Address offset：0x04

Reset value：0x0000_0000

当时钟源切换时，访问该寄存器有 1 或者 2 个时钟的等待周期。

当 APB 或者 AHB 分频值更新时，访问该寄存器可能有 0 ~15 个时钟的等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCO[3:0]				SRAM2PRE	USBPRE[2:0]			Res.	ADCPRE[2:0]		
-				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**
Res.	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:24	MCO [3:0]	RW	4’h0	MCO 输出时钟选择。 0000：MCO 不输出 0001：LSE 0010：LSI40K 0011：HSI48 0100：SYSCLK 0101：HSI16 0110：HSE 0111：PLL 1000：HCLK 1001：APB1 时钟 1010：APB2 时钟 1011：HSI10M others：不输出 注 1：MCO 输出选择系统时钟时，需要保证输出时频率不超过 IO 的最大允许频率。

23	SRAM2PRE	RW	0	SRAM2 时钟分频系数。基于 HCLK 的分频系数。 0：不分频 1：2 分频
22:20	USBPRE[2:0]	RW	3’h0	USB 时钟分频。 000：PLL 时钟 1.5 分频 001：PLL 时钟 010：PLL 时钟 2.5 分频 011：PLL 时钟 2 分频 100：PLL 时钟 3 分频 101：PLL 时钟 3.5 分频 11x：PLL 时钟 4 分频
19	保留	-	-	-
18:16	ADCPRE[2:0]	RW	3’h0	配置 ADC 时钟的分频因子。 000：APB2（ADC）2 分频 001：APB2（ADC）4 分频 010：APB2（ADC）6 分频 011：APB2（ADC）8 分频 100：APB2（ADC）2 分频 101：APB2（ADC）12 分频 110：APB2（ADC）8 分频 111：APB2（ADC）16 分频
15:14	保留	-	-	-
13:11	PPRE2[2:0]	RW	3’h0	高速 APB（APB2）时钟分频系数。 PCLK 基于 HCLK 的分频系数。 0xx：不分频 100：2 分频 101：4 分频 110：8 分频 111：16 分频
10:8	PPRE1[2:0]	RW	3’h0	高速 APB（APB1）时钟分频系数。 PCLK 基于 HCLK 的分频系数。 0xx：不分频 100：2 分频 101：4 分频 110：8 分频 111：16 分频
7:4	HPRE[3:0]	RW	4’h0	AHB 时钟 HCLK 基于 SYSCLK 的分频系数。 0xxx：不分频 1000：2 分频 1001：4 分频 1010：8 分频 1011：16 分频 1100：64 分频 1101：128 分频 1110：256 分频

				1111：512 分频 为了保证系统正常工作，需要根据 VR 电源情况配置合适频率。 注：建议逐级切换分频系数。
3:2	SWS[1:0]	R	2’h0	系统时钟选择状态。 该位由硬件控制，表明系统时钟源的选择状态。 00：HSI16 01：HSE 10：PLL CLK Others：保留
1:0	SW[1:0]	RW	2’h0	系统时钟源选择。 该位由硬件和软件控制，表明系统时钟源的选择。 00：HSI16 01：HSE 10：PLL CLK Others：保留 硬件配置为 HSI 的情况： ● MCU 从停止、待机模式退出 ● 软件配置为 01(HSE)，但 HSE 检测失败 ● 软件配置为 10，并且 PLL 源选择 HSE，但 HSE 检测失败

6.10.3 时钟中断寄存器（RCC_CIR）

Address offset：0x08

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	HSI48RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
-								W	-	W	W	W	W	W	W
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res		HSI48RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res.	HSI48RDYF	PLLRDYF	HSERDYF	HSIRDYIF	LSERDYF	LSIRDYF
-		RW	RW	RW	RW	RW	RW	R	-	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	CSSC	W	0	HSE 时钟安全系统（CSS）中断标志清零。 0：没有影响； 1：清零 CSSF 标志
22	保留	-	-	-
21	HSI48RDYC	W	0	HSI48 ready 中断标志清零。 0：没有影响； 1：清零 HSI48RDYF 标志；
20	PLLRDYC	W	0	PLL ready 中断标志清零。

				0：没有影响； 1：清零 PLLRDYF 标志；
19	HSERDYC	W	0	HSE ready 中断标志清零。 0：没有影响； 1：清零 HSERDYF 标志；
18	HSIRDYC	W	0	HSI ready 中断标志清零。 0：没有影响； 1：清零 HSIRDYF 标志；
17	LSERDYC	W	0	LSE ready 中断标志清零。 0：没有影响； 1：清零 LSERDYF 标志；
16	LSIRDYC	W	0	LSI ready 中断标志清零。 0：没有影响； 1：清零 LSIRDYF 标志；
15:14	保留	-	-	-
13	HSI48RDYIE	RW	0	HSI48 时钟 ready 中断使能。 0：禁止； 1：使能；
12	PLLRDYIE	RW	0	PLL ready 中断使能。 0：禁止； 1：使能；
11	HSERDYIE	RW	0	HSE 时钟 ready 中断使能。 0：禁止； 1：使能；
10	HSIRDYIE	RW	0	HSI16 时钟 ready 中断使能。 0：禁止； 1：使能；
9	LSERDYIE	RW	0	LSE 时钟 ready 中断使能。 0：禁止； 1：使能；
8	LSIRDYIE	RW	0	LSI 时钟 ready 中断使能。 0：禁止； 1：使能；
7	CSSF	R	0	时钟安全系统中断标志。 当 HSE 时钟出现故障时，由硬件置 1。 0：HSE 时钟安全系统中断未产生； 1：HSE 时钟安全系统中断产生； 写 CSSC 寄存器 1 清零该位。
6	保留	-	-	-
5	HSI48RDYF	R	0	HSI48 时钟 ready 中断标志。 HSI48 时钟稳定并且 HSI48RDYIE=1 时，硬件置位该寄存器。 0：HSI48 时钟 ready 中断未产生； 1：HSI48 时钟 ready 中断产生； 写 HSI48RDYC 寄存器 1 清零该位。
4	PLLRDYF	R	0	PLL 时钟 ready 中断标志。

				PLL 时钟稳定并且 PLLRDYDIE=1 时，硬件置位该寄存器。 0：PLL 时钟 ready 中断未产生； 1：PLL 时钟 ready 中断产生； 写 PLLRDYC 寄存器 1 清零该位。
3	HSERDYF	R	0	HSE 时钟 ready 中断标志。 当 HSE 时钟稳定并且 HSERDYIE=1 时，硬件置位该寄存器。 0：HSE 时钟 ready 中断未产生； 1：HSE 时钟 ready 中断产生； 写 HSERDYC 寄存器 1 清零该位。
2	HSIRDYF	R	0	HSI16 时钟 ready 中断标志。 HSI16 时钟稳定并且 HSIRDYIE=1 时，硬件置位该寄存器。 0：HSI16 时钟 ready 中断未产生； 1：HSI16 时钟 ready 中断产生； 写 HSIRDYC 寄存器 1 清零该位。
1	LSERDYF	R	0	LSERDY 时钟 ready 中断标志。 LSE 时钟稳定并且 LSERDYDIE=1 时，硬件置位该寄存器。 0：LSERDY 时钟 ready 中断未产生； 1：LSERDY 时钟 ready 中断产生； 写 LSERDYC 寄存器 1 清零该位。
0	LSIRDYF	R	0	LSIRDY 时钟 ready 中断标志。 LSI 时钟稳定并且 LSIRDYIE=1 时，硬件置位该寄存器。 0：LSIRDY 时钟 ready 中断未产生； 1：LSIRDY 时钟 ready 中断产生； 写 LSIRDYC 寄存器 1 清零该位。

6.10.4 AHB1 外设复位寄存器（RCC_AHB1RSTR）

Address offset：0x28

Reset value：0x0000_0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res -															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ESMCRST	Res.	Res.	Res.	Res	Res.	CRCRST	CORDICRST	Res			DMA2RST	DMA1RST
-			RW	-					RW	RW	-			RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	ESMCRST	RW	0	ESMC(QSPI)模块复位。 0：没有影响； 1：该模块复位；

11:7	保留	-	-	-
6	CRCRST	RW	0	CRC 模块复位。 0：没有影响； 1：该模块复位；
5	CORDICRST	RW	0	CORDIC 模块复位。 0：没有影响； 1：该模块复位；
4:2	保留	-	-	-
1	DMA2RST	RW	0	DMA2 模块复位。 0：没有影响； 1：该模块复位；
0	DMA1RST	RW	0	DMA1 模块复位。 0：没有影响； 1：该模块复位；

6.10.5 AHB2 外设复位寄存器（RCC_AHB2RSTR）

Address offset：0x2C

Reset value：0x0000_0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETHRST	USB2RST	USB1RST	Res.	Res.	AESRST	SDIORST	Res.	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Res.	Res.
RW	RW	RW	-		RW	RW	-	RW	RW	RW	RW	RW	RW	-	

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	ETHRST	RW	0	ETH 模块复位。 0：没有影响； 1：该模块复位；
14	USB2RST	RW	0	USB2 模块复位。 0：没有影响； 1：该模块复位；
13	USB1RST	RW	0	USB1 模块复位。 0：没有影响； 1：该模块复位；
12:11	保留	-	-	-
10	AESRST	RW	0	AES 模块复位。 0：没有影响； 1：该模块复位；
9	SDIORST	RW	0	SDIO 模块复位。 0：没有影响； 1：该模块复位；
8	保留	-	-	-

7	IOPFRST	RW	0	IOPF 模块复位。 0：没有影响； 1：该模块复位；
6	IOPERST	RW	0	IOPE 模块复位。 0：没有影响； 1：该模块复位；
5	IOPDRST	RW	0	IOPD 模块复位。 0：没有影响； 1：该模块复位；
4	IOPCRST	RW	0	IOPC 模块复位。 0：没有影响； 1：该模块复位；
3	IOPBRST	RW	0	IOPB 模块复位。 0：没有影响； 1：该模块复位；
2	IOPARST	RW	0	IOPA 模块复位。 0：没有影响； 1：该模块复位；
1:0	保留	-	-	-

6.10.6 APB1 外设复位寄存器 1（RCC_APB1RSTR1）

Address offset：0x38

Reset value：0x0000_0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTCRST	I2C3RST	DACRST	PWRRST	Res.	CAN2RST	CAN1RST	Res.	UART3RST	I2C2RST	I2C1RST	UART2RST	UART1RST	USART3RST	USART2RST	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**
SPI3RST	SPI2RST	Res.	Res.	WWDGRST	Res.	TIM18RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
RW	RW	-		RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	CTCRST	RW	0	CTC 模块复位。 0：没有影响； 1：该模块复位；
30	I2C3RST	RW	0	I2C3 模块复位。 0：没有影响； 1：该模块复位；
29	DACRST	RW	0	DAC1 模块复位。 0：没有影响； 1：该模块复位；
28	PWRRST	RW	0	Power 接口模块复位。 0：没有影响； 1：该模块复位；
27	保留	-	-	-
26	CAN2RST	RW	0	CANFD 模块复位。

				0：没有影响； 1：该模块复位；
25	CAN1RST	RW	0	CAN2.0 模块复位。 0：没有影响； 1：该模块复位；
24	保留	-	-	-
23	UART3RST	RW	0	UART3 模块复位。 0：没有影响； 1：该模块复位；
22	I2C2RST	RW	0	I2C2 模块复位。 0：没有影响； 1：该模块复位；
21	I2C1RST	RW	0	I2C1 模块复位。 0：没有影响； 1：该模块复位；
20	UART2RST	RW	0	UART2 模块复位。 0：没有影响； 1：该模块复位；
19	UART1RST	RW	0	UART1 模块复位。 0：没有影响； 1：该模块复位；
18	USART3RST	RW	0	USART3 模块复位。 0：没有影响； 1：该模块复位；
17	USART2RST	RW	0	USART2 模块复位。 0：没有影响； 1：该模块复位；
16	保留	-	-	-
15	SPI3RST	RW	0	SPI3 模块复位。 0：没有影响； 1：该模块复位；
14	SPI2RST	RW	0	SPI2 模块复位。 0：没有影响； 1：该模块复位；
13:12	保留	-	-	-
11	WWDGRST	RW	0	WWDG 模块复位。 0：没有影响； 1：该模块复位；
10	保留	-	-	-
9	TIM18RST	RW	0	TIM18 模块复位。 0：没有影响； 1：该模块复位；
8	TIM14RST	RW	0	TIM14 模块复位。 0：没有影响； 1：该模块复位；
7	TIM13RST	RW	0	TIM13 模块复位。

				0：没有影响； 1：该模块复位；
6	TIM12RST	RW	0	TIM12 模块复位。 0：没有影响； 1：该模块复位；
5	TIM7RST	RW	0	TIM7 模块复位。 0：没有影响； 1：该模块复位；
4	TIM6RST	RW	0	TIM6 模块复位。 0：没有影响； 1：该模块复位；
3	TIM5RST	RW	0	TIM5 模块复位。 0：没有影响； 1：该模块复位；
2	TIM4RST	RW	0	TIM4 模块复位。 0：没有影响； 1：该模块复位；
1	TIM3RST	RW	0	TIM3 模块复位。 0：没有影响； 1：该模块复位；
0	TIM2RST	RW	0	TIM2 模块复位。 0：没有影响； 1：该模块复位；

6.10.7 APB1 外设复位寄存器 2（RCC_APB1RSTR2）

Address offset：0x3C

Reset value：0x0000_0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	I2C4RST	LPUART1RST	LPTIM1RST
-													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	I2C4RST	RW	0	I2C4 模块复位。 0：没有影响； 1：该模块复位；
1	LPUART1RST	RW	0	LPUART1 模块复位。 0：没有影响； 1：该模块复位；
0	LPTIM1RST	RW	0	LPTIM1 模块复位。 0：没有影响； 1：该模块复位；

6.10.8 APB2 外设复位寄存器（RCC_APB2RSTR）

Address offset：0x40

Reset value：0x0000_0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LCDC RST	OPA RST	COM PRST	RNG RST	TIM1 9RST	TIM1 7RST	TIM1 6RST	TIM1 5RST	TIM1 1RST	TIM1 0RST	TIM9 RST	Res.	Res.	Res.
-		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**
ADC3 RST	USART 1RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	Res								SYSCF GRST
RW	RW	RW	RW	RW	RW	RW	-								RW

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	LCDCRST	RW	0	LCDC 模块复位。 0：没有影响； 1：该模块复位；
28	OPARST	RW	0	OPA 模块复位。 0：没有影响； 1：该模块复位；
27	COMPRST	RW	0	COMP 模块复位。 0：没有影响； 1：该模块复位；
26	RNGRST	RW	0	RNG 模块复位。 0：没有影响； 1：该模块复位；
25	TIM19RST	RW	0	TIM19 模块复位。 0：没有影响； 1：该模块复位；
24	TIM17RST	RW	0	TIM17 模块复位。 0：没有影响； 1：该模块复位；
23	TIM16RST	RW	0	TIM16 模块复位。 0：没有影响； 1：该模块复位；
22	TIM15RST	RW	0	TIM15 模块复位。 0：没有影响； 1：该模块复位；
21	TIM11RST	RW	0	TIM11 模块复位。 0：没有影响； 1：该模块复位；
20	TIM10RST	RW	0	TIM10 模块复位。 0：没有影响； 1：该模块复位；
19	TIM9RST	RW	0	TIM9 模块复位。

				0：没有影响； 1：该模块复位；
18:16	保留	-	-	-
15	ADC3RST	RW	0	ADC3 模块复位。 0：没有影响； 1：该模块复位；
14	USART1RST	RW	0	USART1 模块复位。 0：没有影响； 1：该模块复位；
13	TIM8RST	RW	0	TIM8 模块复位。 0：没有影响； 1：该模块复位；
12	SPI1RST	RW	0	SPI1 模块复位。 0：没有影响； 1：该模块复位；
11	TIM1RST	RW	0	TIM1 模块复位。 0：没有影响； 1：该模块复位；
10	ADC2RST	RW	0	ADC2 模块复位。 0：没有影响； 1：该模块复位；
9	ADC1RST	RW	0	ADC1 模块复位。 0：没有影响； 1：该模块复位；
8:1	保留	-	-	-
0	SYSCFGRST	RW	0	SYSCFG 模块复位。 0：没有影响； 1：该模块复位；

6.10.9 AHB1 外设时钟使能寄存器（RCC_AHB1ENR）

Address offset：0x48

Reset value：0x0000_0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res -															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ESMCEN	Res.	Res.	Res.	Res	Res.	CRCEN	CORDICEN	FMCEN	Res.	SRAMEN	DMA2EN	DMA1EN
-			RW	-					RW	RW	RW	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	ESMCEN	RW	0	ESMC(QSPI)模块时钟使能。 0：禁止； 1：使能；
11:7	保留	-	-	-

6	CRCEN	RW	0	CRC 模块时钟使能。 0：禁止； 1：使能；
5	CORDICEN	RW	0	CORDIC 模块时钟使能。 0：禁止； 1：使能；
4	FMCEN	RW	1	FLASH 接口模块时钟使能，针对 Sleep 模式。 0：禁止； 1：使能；
3	保留	-	-	-
2	SRAMEN	RW	1	SRAM 存储区时钟使能，针对 Sleep 模式。 0：禁止； 1：使能；
1	DMA2EN	RW	0	DMA2 模块时钟使能。 0：禁止； 1：使能；
0	DMA1EN	RW	0	DMA1 模块时钟使能。 0：禁止； 1：使能；

6.10.10 AHB2 外设时钟使能寄存器（RCC_AHB2ENR）

Address offset：0x4C

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETHRXEN	ETHTXEN
-														RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETHEN	USB2EN	USB1EN	Res.	Res.	AESEN	SDIOEN	Res.	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.	Res.
RW	RW	RW	-		RW	RW	-	RW	RW	RW	RW	RW	RW	-	

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	ETHRXEN	RW	0	ETH 模块 RX 时钟使能。 0：禁止； 1：使能；
16	ETHTXEN	RW	0	ETH 模块 TX 时钟使能。 0：禁止； 1：使能；
15	ETHEN	RW	0	ETH 模块时钟使能。 0：禁止； 1：使能；
14	USB2EN	RW	0	USB2 模块时钟使能。 0：禁止； 1：使能；
13	USB1EN	RW	0	USB1 模块时钟使能。

				0：禁止； 1：使能；
12:11	保留	-	-	-
10	AESEN	RW	0	AES 模块时钟使能。 0：禁止； 1：使能；
9	SDIOEN	RW	0	SDIO 模块时钟使能。 0：禁止； 1：使能；
8	保留	-	-	-
7	IOPFEN	RW	0	IOPF 模块时钟使能。 0：禁止； 1：使能；
6	IOPEEN	RW	0	IOPE 模块时钟使能。 0：禁止； 1：使能；
5	IOPDEN	RW	0	IOPD 模块时钟使能。 0：禁止； 1：使能；
4	IOPCEN	RW	0	IOPC 模块时钟使能。 0：禁止； 1：使能；
3	IOPBEN	RW	0	IOPB 模块时钟使能。 0：禁止； 1：使能；
2	IOPAEN	RW	0	IOPA 模块时钟使能。 0：禁止； 1：使能；
1:0	保留	-	-	-

6.10.11 APB1 外设时钟使能寄存器 1（RCC_APB1ENR1）

Address offset：0x58

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTCEN	I2C3EN	DACEN	PWREN	BKPEN	CAN2EN	CAN1EN	Res.	UART3EN	I2C2EN	I2C1EN	UART2EN	UART1EN	USART3EN	USART2EN	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**
SPI3EN	SPI2EN	Res	Res.	WWDGEN	Res.	TIM18EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
RW	RW	-		RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	CTCEN	RW	0	CTC 模块时钟使能。 0：禁止； 1：使能；
30	I2C3EN	RW	0	I2C3 模块时钟使能。

				0：禁止； 1：使能；
29	DACEN	RW	0	DAC 模块时钟使能。 0：禁止； 1：使能；
28	PWREN	RW	0	PWR 模块时钟使能。 0：禁止； 1：使能；
27	BKPEN	RW	0	BACKUP 模块时钟使能。 0：禁止； 1：使能；
26	CAN2EN	RW	0	CANFD 模块时钟使能。 0：禁止； 1：使能；
25	CAN1EN	RW	0	CAN2.0 模块时钟使能。 0：禁止； 1：使能；
24	保留	-	-	-
23	UART3EN	RW	0	UART3 模块时钟使能。 0：禁止； 1：使能；
22	I2C2EN	RW	0	I2C2 模块时钟使能。 0：禁止； 1：使能；
21	I2C1EN	RW	0	I2C1 模块时钟使能。 0：禁止； 1：使能；
20	UART2EN	RW	0	UART2 模块时钟使能。 0：禁止； 1：使能；
19	UART1EN	RW	0	UART1 模块时钟使能。 0：禁止； 1：使能；
18	USART3EN	RW	0	USART3 模块时钟使能。 0：禁止； 1：使能；
17	USART2EN	RW	0	USART2 模块时钟使能。 0：禁止； 1：使能；
16	保留	-	-	-
15	SPI3EN	RW	0	SPI3 模块时钟使能。 0：禁止； 1：使能；
14	SPI2EN	RW	0	SPI2 模块时钟使能。 0：禁止； 1：使能；

13:12	保留	-	-	-
11	WWDGEN	RW	0	WWDG 模块时钟使能。 0：禁止； 1：使能；
10	保留	-	-	-
9	TIM18EN	RW	0	TIM18 模块时钟使能。 0：禁止； 1：使能；
8	TIM14EN	RW	0	TIM14 模块时钟使能。 0：禁止； 1：使能；
7	TIM13EN	RW	0	TIM13 模块时钟使能。 0：禁止； 1：使能；
6	TIM12EN	RW	0	TIM12 模块时钟使能。 0：禁止； 1：使能；
5	TIM7EN	RW	0	TIM7 模块时钟使能。 0：禁止； 1：使能；
4	TIM6EN	RW	0	TIM6 模块时钟使能。 0：禁止； 1：使能；
3	TIM5EN	RW	0	TIM5 模块时钟使能。 0：禁止； 1：使能；
2	TIM4EN	RW	0	TIM4 模块时钟使能。 0：禁止； 1：使能；
1	TIM3EN	RW	0	TIM3 模块时钟使能。 0：禁止； 1：使能；
0	TIM2EN	RW	0	TIM2 模块时钟使能。 0：禁止； 1：使能；

6.10.12 APB1 外设时钟使能寄存器 2（RCC_APB1ENR2）

Address offset：0x5C

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C4EN	LPUART1EN	LPTIM1EN
-													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	I2C4 EN	RW	0	I2C4 模块时钟使能。 0：禁止； 1：使能；
1	LPUART1EN	RW	0	LPUART1 模块时钟使能。 0：禁止； 1：使能；
0	LPTIM1EN	RW	0	LPTIM1 模块时钟使能。 0：禁止； 1：使能；

6.10.13 APB2 外设时钟使能寄存器（RCC_APB2ENR）

Address offset：0x60

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LCD CEN	OPA EN	COM PEN	RNG EN	TIM1 9EN	TIM1 7EN	TIM1 6EN	TIM1 5EN	TIM1 1EN	TIM1 0EN	TIM9 EN	Res		
-		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**
ADC 3EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC 2EN	ADC1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCF GEN
RW	RW	RW	RW	RW	RW	RW	-								RW

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	LCDCEN	RW	0	LCDC 模块时钟使能。 0：禁止； 1：使能；
28	OPAEN	RW	0	OPA 模块时钟使能。 0：禁止； 1：使能；
27	COMPEN	RW	0	COMP 模块时钟使能。 0：禁止； 1：使能；
26	RNGEN	RW	0	RNG 模块时钟使能。 0：禁止； 1：使能；
25	TIM19EN	RW	0	TIM19 模块时钟使能。 0：禁止； 1：使能；
24	TIM17EN	RW	0	TIM17 模块时钟使能。 0：禁止； 1：使能；
23	TIM16EN	RW	0	TIM16 模块时钟使能。 0：禁止；

				1：使能；
22	TIM15EN	RW	0	TIM15 模块时钟使能。 0：禁止； 1：使能；
21	TIM11EN	RW	0	TIM11 模块时钟使能。 0：禁止； 1：使能；
20	TIM10EN	RW	0	TIM10 模块时钟使能。 0：禁止； 1：使能；
19	TIM9EN	RW	0	TIM9 模块时钟使能。 0：禁止； 1：使能；
18:16	保留	-	-	-
15	ADC3EN	RW	0	ADC3 模块时钟使能。 0：禁止； 1：使能；
14	USART1EN	RW	0	USART1 模块时钟使能。 0：禁止； 1：使能；
13	TIM8EN	RW	0	TIM8 模块时钟使能。 0：禁止； 1：使能；
12	SPI1EN	RW	0	SPI1 模块时钟使能。 0：禁止； 1：使能；
11	TIM1EN	RW	0	TIM1 模块时钟使能。 0：禁止； 1：使能；
10	ADC2EN	RW	0	ADC2 模块时钟使能。 0：禁止； 1：使能；
9	ADC1EN	RW	0	ADC1 模块时钟使能。 0：禁止； 1：使能；
8:1	保留	-	-	-
0	SYSCFGEN	RW	0	SYSCFG 模块时钟使能。 0：禁止； 1：使能；

6.10.14外设独立时钟配置寄存器（RCC_CCIPR）

Address offset：0x68

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res	Res	Res	Res	COMP4SEL[1:0]		COMP3SEL[1:0]		COMP2SEL[1:0]		COMP1SEL[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**
LPTIM1SEL[1:0]		LPUART1SEL[1:0]		Res.						USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
RW	RW	RW	RW	-						RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23:22	COMP4SEL[1:0]	RW	2'h0	COMP4 时钟源选择。 00：PCLK； 01：LSI； 10：LSE； 11：保留； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
21:20	COMP3SEL[1:0]	RW	2'h0	COMP3 时钟源选择。 00：PCLK； 01：LSI； 10：LSE； 11：保留； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
19:18	COMP2SEL[1:0]	RW	2'h0	COMP2 时钟源选择。 00：PCLK； 01：LSI； 10：LSE； 11：保留； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
17:16	COMP1SEL[1:0]	RW	2'h0	COMP1 时钟源选择。 00：PCLK； 01：LSI； 10：LSE； 11：保留； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
15:14	LPTIM1SEL[1:0]	RW	2'h0	LPTIM1 时钟源选择。 00：PCLK； 01：LSI； 10：HSI； 11：LSE； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
13:12	LPUART1SEL[1:0]	RW	2'h0	LPUART1 时钟源选择。 00：PCLK；

				01：SYSCLK； 10：HSI； 11：LSE； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
11:6	保留	-	-	-
5:4	USART3SEL[1:0]	RW	2'h0	USART3 时钟源选择。 00：PCLK； 01：SYSCLK； 10：HSI； 11：LSE； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
3:2	USART2SEL[1:0]	RW	2'h0	USART2 时钟源选择。 00：PCLK； 01：SYSCLK； 10：HSI； 11：LSE； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。
1:0	USART1SEL[1:0]	RW	2'h0	USART1 时钟源选择。 00：PCLK； 01：SYSCLK； 10：HSI； 11：LSE； 注：切换时钟源前，需要先关闭时钟使能，切换完成后再开启时钟使能。

6.10.15 RTC domain 控制寄存器（RCC_BDCR）

Address offset：0x70

Reset value：0x0000_0000

该寄存器中 LSEON、LSEBYP、LSEDRV、RTCSEL、RTCEN 位在备份域中，仅备份域复位时能被复位，备份域复位包括备份域上电复位(BPOR)和备份域软复位(BDRST)。

当连续访问该寄存器时，0 ≤ wait state ≤ 3。

该寄存器不属于 V_{DDD} 域，而属于 RTC 域（V_{BAK} 域）。在复位后，该寄存器需要处于写保护状态，即需要设置 PWR_CR1.DBP 位为 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															BDRST
-															RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	Res.	Res.	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
RW	-					RW	RW	-			RW	RW	RW	R	RW

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	BDRST	RW	0	备份域软复位。 0：没有影响； 1：复位；
15	RTCEN	RW	0	RTC 和 TAMP 时钟使能。 0：禁止； 1：使能；
14:10	保留	-	-	-
9:8	RTCSEL[1:0]	RW	2'h0	RTC 时钟源选择。 00：No clock 01：LSE 10：LSI 11：HSE divided by 128。 一旦 RTC 时钟源选择后不能再改变，除非备份域复位被清零。
7：5	保留	-	-	-
4:3	LSEDRV[1:0]	RW	2'h0	LSE 驱动能力设置 00：gm 2.5uA/V 01：gm 3.75uA/V 10：gm 8.5uA/V 11：gm 13.5uA/V
2	LSEBYP	RW	0	LSE OSC bypass 0:低速外部时钟选择晶振； 1:低速外部时钟选择外部接口输入时钟； 注：只有当外部 32.768KHz OSC 禁止（LSEON=0 并且 LSERDY=0）时才能写该位。
1	LSERDY	R	0	LSE OSC ready. 硬件配置该位为 1 表明 LSE 时钟稳定。 在 LSEON 清零后，该位需要 6 个 LSE 时钟后再清零。
0	LSEON	RW	0	LSE OSC 使能。 0：禁止； 1：使能；

6.10.16控制/状态寄存器（RCC_CSR）

Address offset：0x74

Reset value：0x0800_0000

该寄存器中复位标志位只能由 power reset 复位，其他由 system reset 复位。

当连续访问该寄存器时，0 ≤ wait state ≤ 3。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
R	R	R	R	R	R	R	RW	-							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
-														R	RW

Bit	Name	R/W	Reset Value	Function
31	LPWRRSTF	R	0	低功耗复位标志。 当进入停止/待机低功耗模式时，硬件置位该寄存器。 只有当 nRST_STOP, nRST_STDBY 选项字节为清零（有效）状态时才能操作该寄存器。 RMVF 置 1 会清零该位。
30	WWDGRSTF	R	0	窗口 WDG 复位标志。 RMVF 置 1 会清零该位。
29	IWDGRSTF	R	0	IWDG 复位标志。 RMVF 置 1 会清零该位。
28	SFTRSTF	R	0	软复位标志。 RMVF 置 1 会清零该位。
27	PWRRSTF	R	1	POR/PDR 复位标志。 RMVF 置 1 会清零该位。
26	PINRSTF	R	0	外部 NRST 管脚复位标志。 RMVF 置 1 会清零该位。
25	OBLRSTF	R	0	选项字节 loader 复位标志。 RMVF 置 1 会清零该位。
24	RMVF	RW	0	软件置位清零复位标志。 写该位为 1 的操作会清零 bit25 开始的复位标志，软件写 1 后该位会维持为 1，硬件不会清零。
23:2	保留	-	-	-
1	LSIRDY	R	0	LSI OSC 稳定标志。 0：LSI OSC 没有 ready； 1：LSI OSC ready；
0	LSION	RW	0	LSI OSC 使能。 0：禁止； 1：使能； 硬件开启模拟 LSI 的情况： 1） 硬件 IWDG 使能；

6.10.17 时钟配置寄存器 1（RCC_CFGR1）

Address offset：0x78

Reset value：0x1080_8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USBSELHSI48	PVDSEL[1:0]		HSI48CAL[12:0]												
RW	RW	RW	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**
HSI48TRIM[6:0]							Res.	Res.	Res.	HSI48RDY	HSI48ON	Res.	MCOPRE[2:0]		
RW							-			R	RW	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	USBSELHSI48	RW	0	USBD 模块时钟选择。 0：USBD 时钟选择 PLL（分频）； 1：USBD 时钟选择 HSI48；

30:29	PVDSEL[1:0]	RW	2’h0	PVD 滤波时钟选择。 00：PVD 滤波时钟选择 LSI； 01：PVD 滤波时钟选择 LSE； 10：PVD 滤波时钟选择 PWR PCLK； 11：保留； 注：在配置 PWR_CR.PVDE=1 使能 PVD 功能之前需要先配置该寄存器选择 PVD 滤波时钟，并在整个过程中时钟选择不会变化。
28:16	HSI48CAL[12:0]	R	13’h1080	HSI48 时钟校准值。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSI48CAL+HSI48TRIM，当 HSI48TRIM 值改变时，该寄存器值也会更新。
15:9	HSI48TRIM[6:0]	RW	7’h40	HSI48 时钟 Trimming 值。 软件写入该寄存器调整 HSI48 时钟，叠加到 HSI48CAL 后输出给模拟 HSI48。 系统复位时该寄存器会被复位。
8:6	保留	-	-	-
5	HSI48RDY	R	0	HSI48 时钟 ready 标志。 硬件置位表明 HSI48 稳定。该位只有当 HSI48ON=1 时才有效。 0：HSI48 没有 ready； 1：HSI48 ready； 当 HSI48ON 清零后，HSI48RDY 将在 6 个 HSI48 之后拉低。
4	HSI48ON	RW	0	HSI48 时钟使能。 0：HSI48 OFF； 1：HSI48 ON；
3	保留	-	-	-
2:0	MCOPRE[2:0]	RW	3’h0	MCO（microcontroller clock output）分频系数。软件控制这些位，设置 MCO 输出的分频系数： 000：1 001：2 010：4 011：8 100：16 101：32 110：64 111：128 在 MCO 输出使能前，设置这些位。

6.10.18时钟配置寄存器 2（RCC_CFGR2）

Address offset：0x7C

Reset value：0x0000_0008

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

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9:8	HSE_DRV[1:0]	RW	2'h0	HSE 驱动能力配置 00：4~8MHz 01：8~16MHz 10：16~32MHz 11：保留
7:4	保留	-	-	-
3:0	CANCKSEL[3:0]	RW	4'h8	CAN 通讯时钟选择。 0000：PLL 时钟； 0001：PLL 时钟 2 分频； 0010：PLL 时钟 3 分频； 0011：PLL 时钟 4 分频； 0100：PLL 时钟 5 分频； 0101：PLL 时钟 6 分频； 0110：PLL 时钟 7 分频； 0111：PLL 时钟 8 分频； 1000：HSE 时钟； others：无时钟；

6.10.19时钟配置寄存器 3（RCC_CFGR3）

Address offset：0x80

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL_FBDIV[7:0]								Res.		PLL_POSTDIV[1:0]		PLL_PREDIV[1:0]		PLLXTPRE	PLLSRC
RW	RW	RW	RW	RW	RW	RW	RW	-		RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:8	PLL_FBDIV[7:0]	RW	8'h0	配置 PLL 倍频因子。 00000000：x2 00000001：x3 00000010：x4 ….. 11111101：x255 11111110：x256

				11111111：x257
7:6	保留	-	-	-
5:4	PLL_POSTDIV[1:0]	RW	2'h0	锁相环后分频比设置 00：1 分频 01：2 分频 10：4 分频 11：8 分频
3:2	PLL_PREDIV[1:0]	RW	2'h0	锁相环前分频比设置 00：1 分频 01：2 分频 10：4 分频 11：8 分频
1	PLLXTPRE	RW	0	HSE 分频作为 PLL 源。 只能在关闭 PLL 时才能写入此位。 0：HSE 时钟作为 PLL 源 1：HSE 时钟 2 分频作为 PLL 源
0	PLLSRC	RW	0	PLL 时钟源选择。 只能在关闭 PLL 时才能写入此位。 0：HSI16 2 分频后作为 PLL 输入时钟 1：PLLXTPRE 配置后的 HSE 时钟作为 PLL 输入时钟

7. 备份寄存器（BKP）

7.1 简介

备份寄存器由 32 个 32 位的寄存器组成，可用来存储 128 个字节的用户数据。该模块处在备份域中，当 V_{CC} 电源被切断，它们仍然由 V_{BAT}（备份域电源）维持供电。当系统在待机模式下被唤醒，系统复位或电源复位(POR)时，它们也不会被擦除。

备份寄存器由备份域上电复位（BPOR）或软件复位（BDRST）所复位，且在检测到侵入事件时，擦除备份寄存器内容。

BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

复位后，不可访问备份寄存器和 RTC，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 的写访问：

- 通过设置寄存器 RCC_APB1ENR1 的 PWREN 和 BKPEN 位来打开电源和备份域模块时钟
- 将电源控制寄存器 1（PWR_CR1）的 DBP 位设为 1 来使能对备份寄存器和 RTC 的写访问。

7.2 主要特性

- 支持 128 字节数据备份寄存器
- 管理侵入检测并具有中断功能的状态和控制寄存器
- 存储 RTC 校验值
- 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 闹钟脉冲或者秒脉冲
- 备份寄存器受 FLASH 的 RDP（读保护）保护

7.3 功能描述

7.3.1 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 BKP_CR 的 TPAL 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 TPAL=0 时：如果在启动侵入检测 TAMPER 引脚前（通过设置 TPE 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 TPE 位置‘1’后并没有出现上升沿）。
- 当 TPAL=1 时：如果在启动侵入检测引脚 TAMPER 前（通过设置 TPE 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 TPE 位置‘1’后并没有出现下降沿）。

设置 BKP_CSR 寄存器的 TPIE 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 TPE 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：当 V_{CC} 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。

7.3.2 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（BKP_RTCCR）的 CCO 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 121 ppm。

7.3.2.1 概述

实时时钟（RTC）精度是大多数嵌入式应用程序的要求，但由于外部环境温度变化，时钟的晶体频率变化引起 RTC 精度可能不像预期的那样准确。使用数字时钟校准电路，允许应用程序补偿晶体和温度变化。

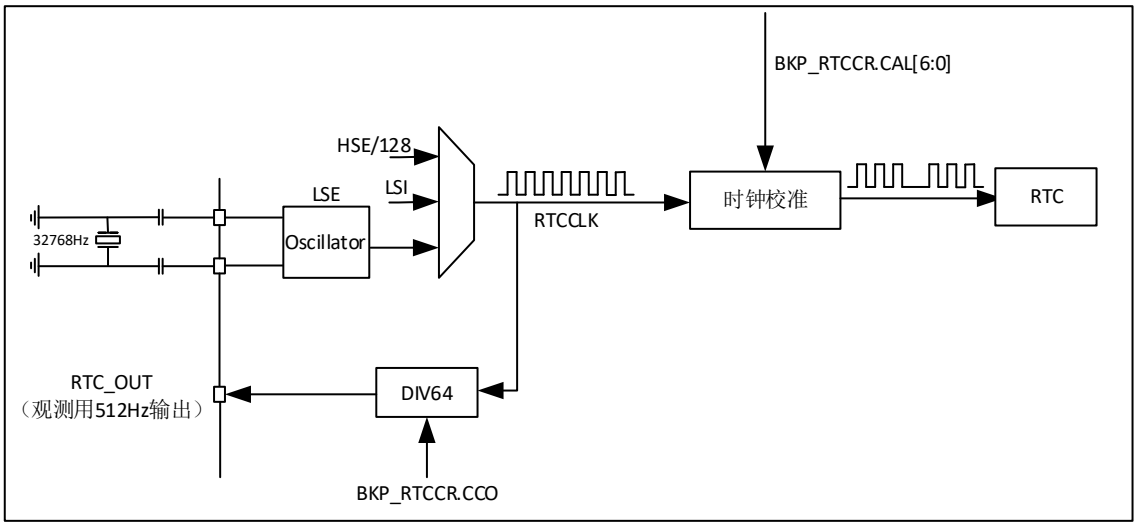
7.3.2.2 RTC 校准方式

RTC 时钟可选择一个额定频率为 32.768kHz 的石英晶体控制振荡器驱动。晶体振荡器是提供固定频率的最精确电路之一。时钟误差有两个原因：

1. 温度变化
2. 晶体变化

如前所述，大多数时钟芯片通过使用繁琐的微调电容器来补偿晶体频率和温度变化。设计采用周期性计数器校正。数字校准电路每 2^{20} 个时钟周期消除 0 至 127 个周期。脉冲被消隐的个数取决于加载到 BKP 的 RTC 时钟校准寄存器 BKP_RTCCR.CAL 的值。由于 RTC 时钟校准寄存器在备份域中，如果电池连接到 V_{BAT} 引脚，即使设备断电，校准值也不会丢失。

结构框图如下所示：



注：引脚上输出的时钟是校准前的 RTC 时钟，因此校准不会改变其值。

每个校准步骤的效果是每 1 048 576（2^{20}）实际振荡器周期减去 1 个振荡器周期。即校准寄存器中每个校准步骤的调整量为 0.954（1000000/2^{20}）ppm。因此，振荡器时钟可以从 0 放慢到 121 ppm。

下表显示了每个位在每个月（30 天）中实时表示的 ppm 和秒数。

校准值	四舍五入到最接近的 ppm	值以秒为单位，每月(30 天)四舍五入到最近的秒	校准值	四舍五入到最接近的 ppm	值以秒为单位，每月(30 天)四舍五入到最近的秒
0	0	0	64	61	158
1	1	2	65	62	161
2	2	5	66	63	163
3	3	7	67	64	166
4	4	10	68	65	168
5	5	12	69	66	171
6	6	15	70	67	173
7	7	17	71	68	176
8	8	20	72	69	178
9	9	22	73	70	180
10	10	25	74	71	183
11	10	27	75	72	185
12	11	30	76	72	188
13	12	32	77	73	190
14	13	35	78	74	193
15	14	37	79	75	195
16	15	40	80	76	198
17	16	42	81	77	200
18	17	44	82	78	203
19	18	47	83	79	205
20	19	49	84	80	208
21	20	52	85	81	210
22	21	54	86	82	213
23	22	57	87	83	215
24	23	59	88	84	218
25	24	62	89	85	220
26	25	64	90	86	222
27	26	67	91	87	225
28	27	69	92	88	227
29	28	72	93	89	230
30	29	74	94	90	232
31	30	77	95	91	235
32	31	79	96	92	237
33	31	82	97	93	240
34	32	84	98	93	242
35	33	87	99	94	245
36	34	89	100	95	247
37	35	91	101	96	250
38	36	94	102	97	252
39	37	96	103	98	255
40	38	99	104	99	257
41	39	101	105	100	260
42	40	104	106	101	262
43	41	106	107	102	264
44	42	109	108	103	267
45	43	111	109	104	269
46	44	114	110	105	272
47	45	116	111	106	274
48	46	119	112	107	277
49	47	121	113	108	279
50	48	124	114	109	282
51	49	126	115	110	284
52	50	129	116	111	287
53	51	131	117	112	289
54	51	133	118	113	292
55	52	136	119	113	294
56	53	138	120	114	297
57	54	141	121	115	299
58	55	143	122	116	302
59	56	146	123	117	304
60	57	148	124	118	307
61	58	151	125	119	309

62	59	153	126	120	311
63	60	156	127	121	314

如上所述，RTC 时钟校准电路只从晶体时钟中减去周期。基于 RTC 预分频器值默认设置为 32768，更快的晶体频率(>32.768 kHz)可以被校准，而较慢的晶体频率(<32.768 kHz)不能被补偿（只校准变快的晶体，变慢的会加剧变慢的情况）。所以只有晶体频率量程[32 772,32 768]可以校准。

由于晶体频率可能在 32.768 kHz 附近变化，比如可以通过设置 RTC 预分频为 32766(而不是 32768)来调整晶体频率从 32.768 kHz 变为 32766 kHz。这样，可以补偿晶体频率为[32770,32766]的范围。

7.3.2.3 计算所需的校准量

为了确定在给定的应用程序中需要多少校准，需要根据具体情况，借助 RTC 时钟的输出功能。RTC 时钟输出可以用来测量晶体振荡器的精度，如果晶体振荡器的频率是精确的 32.768 kHz，则 RTC 时钟输出的频率正好是 512 Hz。

该方法可分为以下步骤:

1.启用低速外部振荡器(LSE)，选择 LSE 作为 RTC 时钟源，然后启用 RTC 时钟。

2.在 RTC_OUT 引脚(PC13)上输出 RTC 时钟的 64 分频率，用于晶体频率测量。通过将 BKP_RTCCR 中的 CCO 位置 1 来实现。

3.计算晶体频率偏差(ppm)。假设 RTC 预分频器值为 32766，通过将 511.968Hz(32766/64≈511.968)的测量偏差除以 511.968，并将结果乘以 100 万，就可以快速计算出 ppm 中的偏差。使用表 1 找到最近的校准值。此表是基于 ppm 表示的变化值的校准值的直接查找表。

4.向 RTC 校准寄存器中写入校准值以补偿晶体偏差。

注意:要将 RTC 预分频器设置为 32766，请将 32765 写入 RTC 预分频器寄存器。

例如，如果在实测中测量到的频率为 511.982 Hz，则δ为 0.014。除以 511.968，再乘以 100 万，结果是 27.35 ppm。在这种情况下，最接近的补偿值是 28。误差将从 27.35 ppm(每月~71 秒)减少到 0.65 ppm(每月~1.7 秒)。

注意:由于 RTC 校准的原理是忽略部分 RTC 时钟脉冲，它不能改善短时间内的计数，它只改善长时间内的计数。例如，在没有校准的情况下，使用 RTC 计数 0.01 秒会比有校准的情况下更准确。由于校准周期在考虑的时间范围内可能发生或不发生，因此结果值可能会发生显著变化。因此，根据应用情况，最好不要使用校准。

7.3.3 备份寄存器的擦除

以下方式可以使得备份寄存器被擦除：

①V_{BAT} 域上电复位

②侵入检测电路成功捕捉到侵入事件

③寄存器 RCC_BDCR 的 BDRST（备份域软复位）被写入 1

④RDP level 从 1 变为 0

⑤TEF 标志位为 1

注意：正在擦除时，总线会处于非应答态，总线无法对 BKP 进行任何读写操作。

7.4 寄存器描述

注：BKP 模块所有寄存器都只能以字方式访问

7.4.1 备份控制寄存器（BKP_CR）

Address offset：0x00

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	TPAL	RW	0	侵入检测 TAMPER 引脚有效电平。 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（TPE=1 时）； 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（TPE=1 时）； 注：当 TPE=1 时，无法更改此位。
0	TPE	RW	0	启动侵入检测 TAMPER 引脚。 0：侵入检测 TAMPER 引脚作为通用 IO 口使用； 1：侵入检测 TAMPER 引脚作为侵入检测使用；

7.4.2 备份控制/状态寄存器（BKP_CSR）

Address offset：0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						TIF	TEF	Res.					TPIE	CTI	CTE
-						R	R	-					RW	W	W

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	TIF	R	0	侵入中断标志。 当检测到侵入事件且 TPIE 位为 1 时，此位由硬件置 1.通过向 CTI 位写 1 来清除此标志位（同时也清除中断）。如果 TPIE 位被清除，此位也会被清除。 此标志能被系统复位清零,但不能被备份域复位清零。 微控制器从待机模式唤醒时,此标志清零 0：无侵入中断； 1：产生侵入中断；
8	TEF	R	0	侵入事件标志。 当检测到侵入事件时此位由硬件置 1.通过向 CTE 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 BKP_DRx 寄存器。当此位被置 1 时，若对 BKP_DRx 进行写操作，写入的值不会被保存。
7:3	保留	-	-	-
2	TPIE	RW	0	允许侵入 TAMPER 引脚中断。 0：禁止侵入检测中断； 1：允许侵入检测中断（BKP_CR 寄存器的 TPE 位也必须被置 1） 此标志能被系统复位清零,但不能被备份域复位清零.当微控制器从待机模式唤醒时,此标志清零
1	CTI	W	0	清除侵入检测中断。 此位只能写入 1，读出值为 0. 0：无效； 1：清除侵入检测中断和 TIF 侵入检测中断标志。
0	CTE	W	0	清除侵入检测事件。 此位只能写入 1，读出值为 0. 0：无效； 1：清除 TEF 侵入检测事件标志（并复位侵入检测器）。

7.4.3 RTC 时钟校准寄存器（BKP_RTCCR)

Address offset：0x08

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						ASOS	ASOE	CCO	CAL[6:0]						
-						RW	RW	RW	RW						

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	ASOS	RW	0	闹钟或者秒输出选择。 当设置了 ASOE 位，ASOS 位可以用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲或闹钟脉冲。 0：输出 RTC 闹钟脉冲； 1：输出秒脉冲； 注：该位只能被备份域复位清零。
8	ASOE	RW	0	允许输出闹钟或者秒脉冲。 根据 ASOS 位配置，该位允许 RTC 秒脉冲或闹钟脉冲输出到 TAMPER 引脚上。 输出脉冲的宽度为一个 RTC 时钟的周期。设置了 ASOE 位时不能开启 TAMPER 的功能。 注：该位只能被备份域复位清零。
7	CCO	RW	0	校准时钟输出。 0：无影响；

				1：在侵入检测引脚输出 RTC 64 分频时钟。 当 CCO 位置 1 后，必须关闭侵入检测功能以避免检测到无效的侵入事件。 注：当 V_{CC} 供电断开时，该位清零。
6:0	CAL[6:0]	RW	6'h0	校准值。 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 0~121 ppm。

注：①当 CCO 和 ASOE 同时为 1 时，ASOS 优先级高。

②该寄存器的数据需要从 APB 同步到 RTC 时钟域，确保写入值确认无误后可以再写入。

7.4.4 备份数据寄存器（BKP_DRx）（x=0…31）

Address offset：0x80 + 0x04 * x,（x = 0 to 31）（0x80~0xFC）

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	D[31:0]	RW	32'h0	备份数据。这些位可以被用来保存用户数据。 注：BKP_DRx 寄存器不会被系统复位、电源复位、从待机模式唤醒复位所复位。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。

8. CRC 计算单元（CRC)

8.1 简介

循环冗余校验(CRC)单元可以利用 7 位、8 位、16 位、32 位的自定义生成多项式，从 8 位、16 位、32 位的生成多项式中计算出 CRC 码。

在其他的应用中，基于 CRC 的技术用于验证数据传输或存储的完整性。在功能安全标准的领域内，基于 CRC 的技术提供了一种验证闪存完整性的方法。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。

8.2 CRC 主要特性

- 使用 CRC-32 以太网多项式：0x4C11DB7

$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

- 支持可编程的多项式，多项式位宽也是可编程的(支持 7，8，16，32 比特的多项式)
- 支持对 8、16、32 比特的数据的计算
- 可自定义 CRC 初始值
- 通用 32 位寄存器(可用于存放临时数据)
- 具备输入缓存功能
- 一个 32 位数据寄存器用于输入 / 输出
- CRC 计算时间：对于 32 位数据，需要 4 个 AHB 时钟周期(HCLK)；对于 16 位数据，需要 2 个 AHB 时钟周期；对于 8 位数据，需要 1 个 AHB 时钟周期(HCLK)
- 支持输入数据反转，输出数据反转
- CRC_DR 寄存器能被 AHB 总线以右对齐 8 位、右对齐 16 位、32 位访问，其他的寄存器只能以 32 位被 AHB 总线访问

8.3 CRC 功能描述

8.3.1 CRC 结构框图

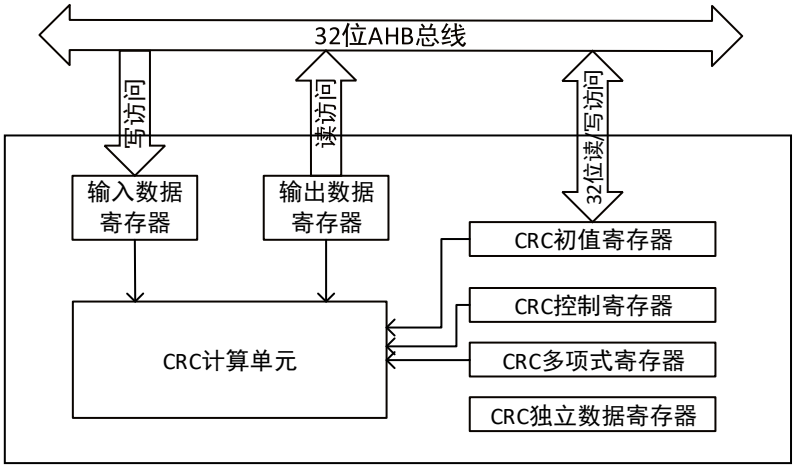


图 8-1 CRC 模块的结构

8.3.2 CRC 操作

CRC 计算单元含有 1 个 32 位数据寄存器(CRC_DR)。对于写操作，它用于保存输入数据；对于读操作，它用于保存之前的 CRC 计算结果。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)，具体取决于所写数据的格式。CRC_DR 寄存器能被字(32 位)、右对齐半字(16 位)、右对齐字节(8 位)访问，而其他的 CRC 寄存器只允许 32 位访问。

对于不同位宽的数据，CRC 计算模块需要不同的时间，具体如下：

- 计算 32 位数据需要 4 个 AHB 时钟周期
- 计算 16 位数据需要 2 个 AHB 时钟周期
- 计算 8 位数据需要 1 个 AHB 时钟周期

输入缓存允许用户在写入第一个数据之后立刻写入第二个数据，无需等待之前的数据计算完毕。

用户可以动态地调整数据大小以减少计算特定字节数的数据时，访问本模块的次数。例如，计算一个 5 字节大小的数据，可以往本模块写入 1 个字的数据，再写入 1 个字节的数据以得到计算结果。

本模块支持输入数据反转功能，以适应不同的大小端。根据 CRC_DR 寄存器的 REV_IN[1:0]，输入数据反转能以 8 位，16 位，32 位为单位进行反转。

例如，对于需要 CRC 运算的输入数据 0x1A2B3C4D：

- 以字节(8 位)为单位进行反转之后的数据是 0x58D43CB2
- 以半字(16 位)为单位进行反转之后的数据是 0xD458B23C
- 以字(32 位)为单位进行反转之后的数据是 0xB23CD458

CRC_CR 寄存器中 REV_OUT 位能将输出数据反转。这项操作是在位层次完成的，例如：输出数据是 0x112233544,被反转后成为 0x22CC4488

将 CRC_CR 寄存器的 RESET 位置 1，能使 CRC 计算器的初始值装载一个可编程的值(默认值为 0xFFFFFFFF)

CRC 初始值能被编程成 CRC_INIT 寄存器所包含的值。当 CRC_INIT 寄存器被写时，CRC_DR 寄存器会被自动地初始化。

CRC_IDR 寄存器可以用来保存与 CRC 计算相关的临时值。它不受 CRC_CR 寄存器中 RESET 位的影响。

8.3.3 多项式可编程性

多项式的系数可以通过 CRC_POL 寄存器完全地进行自定义，多项式的位宽可以根据 CRC_CR 寄存器的 POLYSIZE[1:0]配置为 7 位，8 位，16 位，32 位。偶多项式是不被支持的，例如：偶多项式的形式是 $x+x^2+\ldots+x^n$，而奇多项式的形式是 $1+x+x^2+\ldots+x^n$。

如果 CRC 数据小于 32 比特，CRC 的值将会从 CRC_DR 的最低有效位开始读起。

为了得到可信的 CRC 计算结果，不能在 CRC 的计算期间，更改多项式的值或大小。在更改 CRC 多项式之前，必须复位 CRC。

默认的多项式是 CRC-32（以太网）多项式：0x4C11DB7。

8.4 寄存器描述

8.4.1 数据寄存器（CRC_DR）

Address offset:0x00

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[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
DR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DR[31:0]	RW	32’hFFFFFFFF	数据寄存器位。 写入 CRC 计数器的新数据时，作为输入寄存器。 读取时返回之前的 CRC 计算的结果。 如果 CRC 数据小于 32 位，该寄存器的低位用于读或写正确的数据。

8.4.2 独立数据寄存器（ CRC_IDR）

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[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
IDR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	IDR[31:0]	RW	32’h0	通用 32 位数据寄存器位。可临时存放数据。 寄存器 CRC_CR 的 RESET 位产生的 CRC 复位对本寄存器无影响。 注：此寄存器不参与 CRC 计算，可以存放任何数据。

8.4.3 控制寄存器（CRC_CR）

Address offset:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	REV_OUT	RW	0	反转输出数据

				这个位决定输出数据的位顺序是否反转 0：输出数据的位顺序不反转 1：输出数据的位顺序反转
6:5	REV_IN[1:0]	RW	2’b0	反转输入数据 这 2 个位决定了输入数据的顺序是否反转 00：顺序不反转 01：输入数据每个 byte 都反转 10：输入数据每个 half-word 都反转 11：输入数据每个 word 都反转
4:3	POLYSIZE[1:0]	RW	2’b0	CRC 多项式位宽 这 2 个位决定了 CRC 多项式的位宽 00：32 位 CRC 多项式 01：16 位 CRC 多项式 10：8 位 CRC 多项式 11：7 位 CRC 多项式
2:1	保留	-	-	-
0	RESET	w	0	软件置位后将会复位 CRC 模块，数据寄存器会加载 CRC_INIT 寄存器的值。软件只能写 1，由硬件清零。

8.4.4 CRC 初始值寄存器(CRC_INIT)

Address offset:0x10

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[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**
CRC_INIT[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	CRC_INIT[31:0]	RW	32’hFFFFFFFF	可编程 CRC 初值 这个寄存器用于存储 CRC 初值

8.4.5 CRC 多项式(CRC_POL)

Address offset:0x14

Reset value:0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[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**
POL[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	POL[31:0]	RW	32’h04C11DB7	可编程 CRC 多项式 这个寄存器用于存储 CRC 多项式的系数 如果 CRC 多项式小于 32 位,将以右对齐的方式存放在本寄存器内

9. 通用 IO（GPIO）

9.1 简介

每个通用 IO 端口包括 4 个 32 位配置寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR 和 GPIOx_PUPDR）、2 个 32 位数据寄存器（GPIOx_IDR 和 GPIOx_ODR）、1 个 32 位置位/复位寄存器（GPIOx_BSRR)、一个 32 位复位寄存器(GPIOx_BRR）、1 个 32 位锁定寄存器（GPIOx_LCKR）和 2 个 32 位复用功能选择寄存器（GPIOx_AFRH 和 GPIOx_AFRL，但 GPIOF 只有 GPIOF_AFRL）。

9.2 主要特征

- 受控 IO 多达 86 个
- 输出状态：推挽或开漏 + 上拉/下拉
- 从输出数据寄存器（GPIOx_ODR）或外设（复用功能输出）输出到引脚
- 可为每个 IO 选择不同的速度
- 输入状态：浮空、上拉/下拉、模拟
- 将引脚输入到数据寄存器（GPIOx_IDR）或外设（复用功能输入）
- 置位/复位寄存器（GPIOx_BSRR)，对 GPIOx_ODR 具有按位写权限
- 复位寄存器（GPIOx_BRR)，对 GPIOx_ODR 具有按位写权限
- 锁定寄存器（GPIOx_LCKR)，可冻结 IO 配置
- 复用功能输入/输出选择寄存器（一个 IO 最多可具有 16 个复用功能）
- 快速翻转，每次翻转最快只需要两个时钟周期
- 引脚复用非常灵活，允许将 IO 引脚用作 GPIO 或多种外设功能中的一种

9.3 功能说明

根据数据手册中列出的每个 IO 端口的特性，可通过软件将通用 IO（GPIO）端口的各个端口位分别配置为多种模式：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟功能
- 具有上拉或下拉功能的开漏输出
- 具有上拉或下拉功能的推挽输出
- 具有上拉或下拉功能的复用功能推挽
- 具有上拉或下拉功能的复用功能开漏

每个 IO 端口位均可自由编程，但 IO 端口寄存器必须按字、半字或字节进行访问。GPIOx_BSRR 和 GPIOx_BRR 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。

下图显示了 5 V 兼容和基本 IO 端口位的基本结构。表 9-1 给出了可能的端口位配置方案。

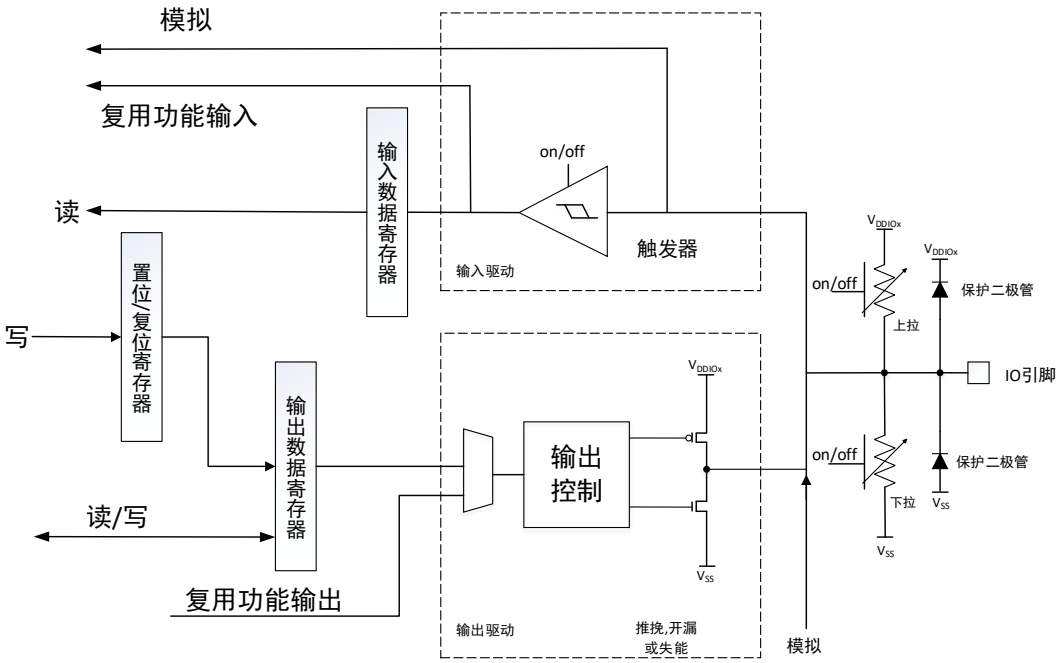


图 9-1 IO 端口基本结构

表 9-1 端口位配置表

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		IO configuration	
01	0	SPEED[1:0]		0	0	GP output	PP
	0			0	1	GP output	PP+PU
	0			1	0	GP output	PP+PD
	0			1	1	保留	
	1			0	0	GP output	OD
	1			0	1	GP output	OD+PU
	1			1	0	GP output	OD+PD
	1			1	1	Reserved(GP output OD)	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP+PU
	0			1	0	AF	PP+PD
	0			1	1	保留	
	1			0	0	AF	OD
	1			0	1	AF	OD+PU
	1			1	0	AF	OD+PD
	1			1	1	保留	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved(input floating)	
11	x	x	x	0	0	Input/Output	Analog
	x	x	x	0	1	保留	
	x	x	x	1	0	Input/Output	Analog,PD
	x	x	x	1	1	保留	

GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

9.3.1 通用 IO（GPIO）

在复位期间及复位刚刚完成后，复用功能尚未激活，大部分 IO 端口被配置为模拟模式。复位后，调试引脚处于复用功能上拉/下拉状态，BOOT0 引脚处于输入模式下拉状态：

- PA15：JTDI 处于上拉状态

- PA14：JTCK/SWCLK 处于下拉状态
- PA13：JTMS/SWDAT 处于上拉状态
- PB4：NJTRST 处于上拉状态
- PB8：BOOT0 处于下拉状态

输入数据寄存器（GPIOx_IDR）每隔 1 个 AHB 时钟周期捕获一次 IO 引脚的数据。

所有 GPIO 引脚都具有内部弱上拉及下拉电阻，可根据 GPIOx_PUPDR 寄存器中的值来打开/关闭。

9.3.2 IO 引脚复用器和映射

IO 引脚通过一个复用器连接到板载外设/模块，该复用器一次仅允许一个外设的复用功能（AF）连接到 IO 引脚。这可以确保共用同一个 IO 引脚的外设之间不会发生冲突。

每个 IO 引脚都有一个复用器，该复用器采用 16 路复用功能输入（AF0 到 AF15），可通过 GPIOx_AFRL（用于引脚 0 到 7）和 GPIOx_AFRH（用于引脚 8 到 15）寄存器对这些输入进行配置。

要将 IO 配置成所需功能，请按照以下步骤操作：

调试功能

系统复位后，调试相关的引脚被复用为 AF，供系统调试接口使用。

GPIO

在 GPIOx_MODER 寄存器中将所需 IO 配置为输出或输入。

外设复用功能

- 在 GPIOx_AFRL 或 GPIOx_AFRH 寄存器中，设置 IO 的复用功能
- 通过 GPIOx_OTYPER、GPIOx_PUPDR 和 GPIOx_OSPEEDR 寄存器，分别选择输出类型、上拉/下拉以及输出速度
- 在 GPIOx_MODER 寄存器中将所需 IO 配置为复用功能

额外功能

- 对于 ADC，DAC，OPA 和 COMP 等，需要通过寄存器 GPIOx_MODER 配置相应的 IO 为模拟模式，并在 ADC，DAC，OPA 和 COMP 等中配置相应功能；需要特别注意，对于附加功能(如 DAC 或 OPA)的输出，需要先配置相应 IO 为模拟模式，再使能外设的控制寄存器。
- 对于额外功能 TAMP_RTC，WKUPx,PVD_IN,OSC 和 OSC32，需要在 RTC，PWR 和 RCC 等模块配置相应功能，这些功能优先于 GPIO 设定。

9.3.3 IO 端口控制寄存器

每个 GPIO 有 4 个 32 位控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR），可配置多达 16 个 IO。GPIOx_MODER 寄存器用于选择 IO 工作模式（输入、输出、复用、模拟）。GPIOx_OTYPER 用于选择输出类型（推挽或开漏）。GPIOx_OSPEEDR 寄存器用于设定 IO 速度 。GPIOx_PUPDR 寄存器用于选择上拉/下拉。

9.3.4 IO 端口数据寄存器

每个 GPIO 都具有 2 个 16 位数据寄存器：输入和输出数据寄存器（GPIOx_IDR 和 GPIOx_ODR）。GPIOx_ODR 用于存储待输出数据，可对其进行读/写访问。IO 输入数据存储到 GPIOx_IDR 中，它是一个只读寄存器。

9.3.5 IO 数据位操作

置位复位寄存器（GPIOx_BSRR）是一个 32 位寄存器，它允许应用程序在输出数据寄存器(GPIOx_ODR)中对各个单独的数据位执行置位和复位操作。

GPIOx_ODR 中的每个数据位对应于 GPIOx_BSRR 中的两个控制位：BS（i)和 BR（i)。当写入 1 时，BS(i）位会置位对应的 ODR(i）位。当写入 1 时，BR(i)位会清零 ODR(i)对应的位。

在 GPIOx_BSRR 中向任何位写入 0 都不会对 GPIOx_ODR 中的对应位产生任何影响。如果在 GPIOx_BSRR 中同时尝试对某个位执行置位和清零操作，则置位操作优先。

使用 GPIOx_BSRR 寄存器更改 GPIOx_ODR 中各个位的值是一个“单次”操作，不会锁定 GPIOx_ODR 位。随时都可以直接访问 GPIOx_ODR 位。GPIOx_BSRR 寄存器提供了一种执行原子按位处理的方法。

在对 GPIOx_ODR 进行位操作时，软件无需禁止中断：在一次原子 AHB 写访问中，可以修改一个或多个位。

复位寄存器（GPIOx_BRR）是一个 32 位寄存器，它允许应用程序在输出数据寄存器(GPIOx_ODR）中对各个单独的数据位执行复位操作。复位寄存器的作用与置位复位寄存器相似，但仅能复位输出数据寄存器的一个或多个位。

9.3.6 GPIO 锁定机制

通过将特定的写序列应用到 GPIOx_LCKR 寄存器，可以冻结 GPIO 控制寄存器。冻结的寄存器包括 GPIOx_MODER 、 GPIOx_OTYPER 、 GPIOx_OSPEEDR 、 GPIOx_PUPDR 、 GPIOx_AFRL 和 GPIOx_AFRH。

要对 GPIOx_LCKR 寄存器执行写操作，必须用特定的写/读序列，软件以正确的锁定序列操作此寄存器的 LCKR[16]后，会使用 LCKR[15:0]的值来锁定 IO 的配置（在写序列期间，LCKR[15:0]的值必须保持不变）。某个或多个端口被锁定后，无法解锁，直到发生系统复位，或 GPIO 被复位。每个 GPIOx_LCKR 位都会冻结控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL 和 GPIOx_AFRH）中的对应位。

锁定序列只能通过字访问对 GPIOx_LCKR 寄存器操作。

具体操作参考 GPIOx_LCKR 寄存器描述。

9.3.7 IO 复用功能输入/输出

有两个寄存器可用来从每个 IO 可用的 16 个复用功能输入/输出中进行选择。

不同的复用信号有不同的输入/输出方向，具体情况由各自的外设模块决定。

9.3.8 外部中断线/唤醒线

所有端口都具有外部中断功能。要使用外部中断线，必须将端口配置为输入模式，请参见中断和事件章节。

9.3.9 输入配置

当 IO 口配置为输入：

- 输出缓冲器不使能

- 施密特触发器输入使能
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 每个 AHB 时钟采样引脚电平，并存放于输入数据寄存器
- 读取输入数据寄存器可得知引脚电平

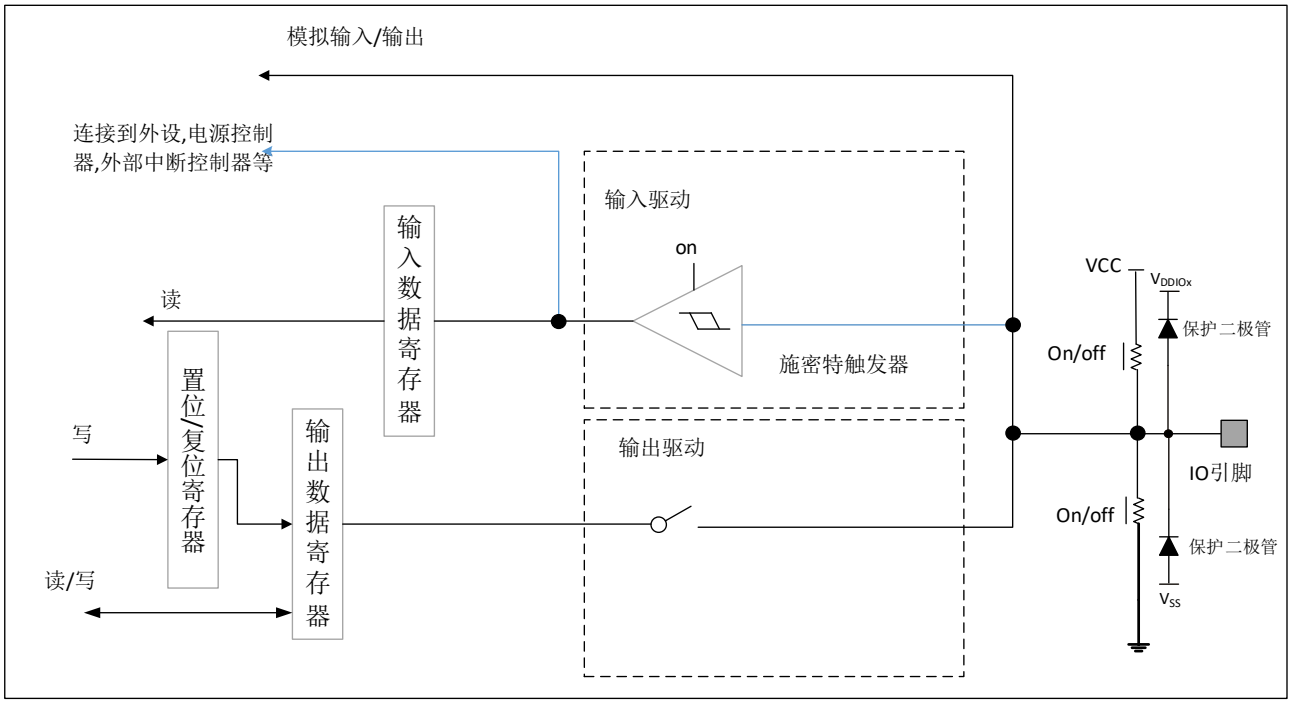


图 9-2 输入浮空/上拉/下拉配置

9.3.10 输出配置

当 IO 端口被配置为输出时：

- 输出缓冲器使能

– 开漏模式：输出寄存器上的'0'导通 N-MOS，而输出寄存器上的'1'将端口置于高阻状态(PMOS 不导通)。

– 推挽模式：输出寄存器上的'0'导通 N-MOS，而输出寄存器上的'1'将导通 P-MOS。

- 施密特触发输入使能
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 每个 AHB 时钟采样引脚电平，并存放于输入数据寄存器
- 读取输入数据寄存器可得知引脚电平
- 读取输出数据寄存器可得知上一次写 GPIO 的值

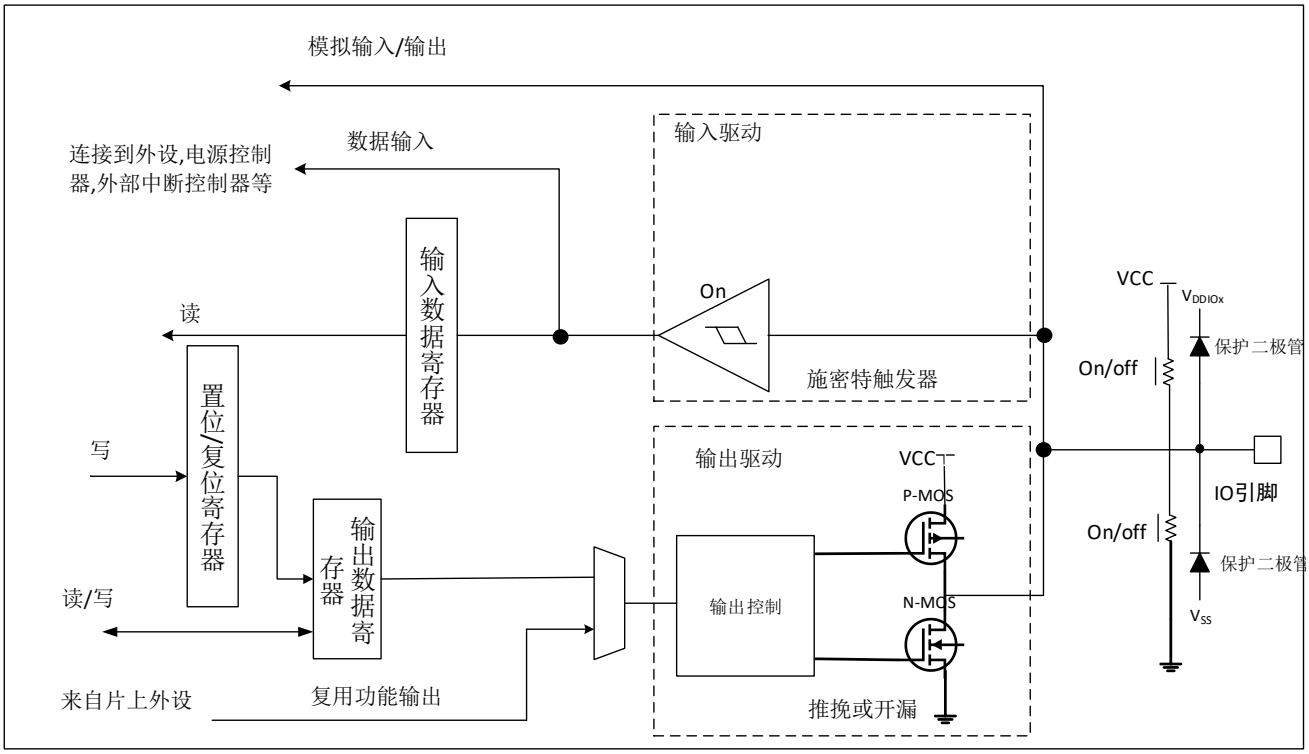


图 9-3 输出配置

9.3.11 复用功能配置

当 IO 端口被配置为复用功能时：

- 输出缓冲器使能
- 输出缓冲器的输出数据和输出使能由外设模块驱动
- 施密特触发输入使能
- 根据寄存器 GPIOx_PUPDR 配置可使能/不使能上下拉电阻
- 每个 AHB 时钟采样引脚电平，并存放于输入数据寄存器
- 读取输出数据寄存器可得知上一次写 GPIO 的值

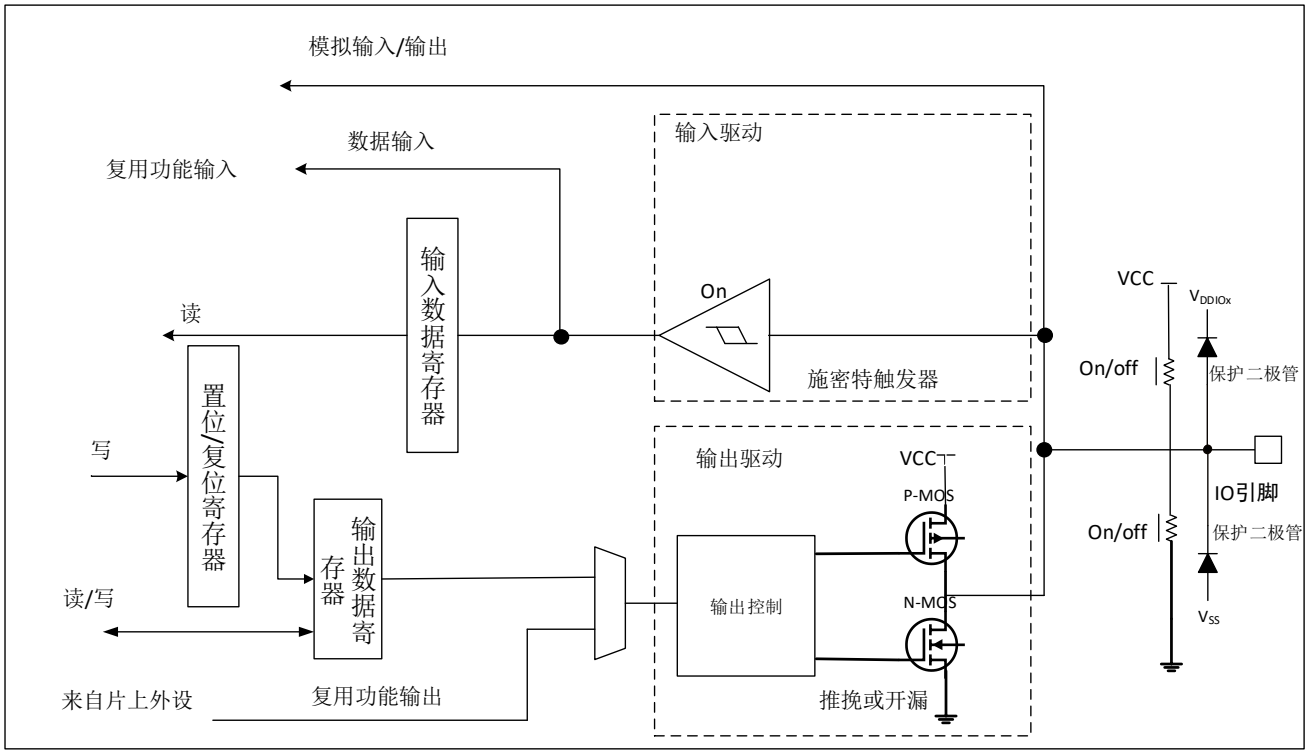


图 9-4 复用功能配置

9.3.12 模拟配置

当 IO 端口被配置为模拟模式时：

- 输出缓冲器不使能；
- 施密特触发器输入不使能，实现了每个模拟 IO 引脚上的零功耗。施密特触发输出值被强置为’0’；
- 弱上拉和下拉电阻被禁止（需要软件设定）；
- 读取输入数据寄存器时数值为’0’。

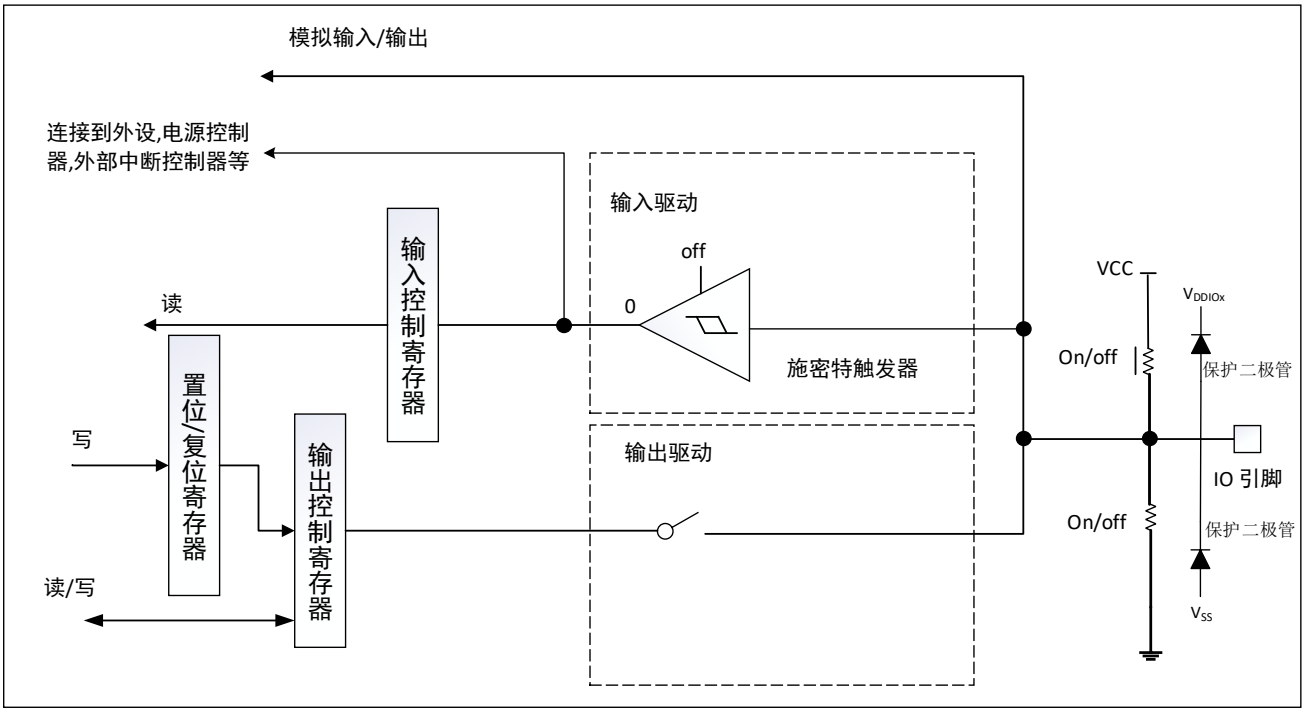


图 9-5 高阻抗模拟配置

9.3.13 使用 HSE 或 LSE 振荡器引脚作为 GPIO

当 HSE 或 LSE 功能不使能（复位后的默认），相应的管脚可以当作正常的 GPIO。

当 HSE 或 LSE 功能使能（RCC 寄存器中置位 HSEON 或 LSEON），相关引脚的功能由振荡器控制，不受这些 GPIO 寄存器的设定影响。

当晶振配置为用户外部时钟模式，只有 OSC_IN 或 OSC32_IN 用于接收外部时钟输入，而 OSC_OUT 或 OSC32_OUT 脚仍然可以用作正常 GPIO。

9.3.14 使用 RTC 电源域中的 GPIO 引脚

当 V_{DDD} 断电时（当设备进入待机模式时），PC13/PC14/PC15 GPIO 功能失效。如果 RTC 配置未覆盖其 GPIO 设定，则在待机唤醒后，这些引脚为模拟输入/输出状态。

9.3.15 使用 PB8 作为 GPIO

PB8 可以用作 boot 引脚（BOOT0）或 GPIO。根据用户选项字节中的 nSWBOOT0 位选择：

- nSWBOOT0=1，复位中为 BOOT0 引脚。
- nSWBOOT0=0，复位中为 GPIO 引脚。

注意事项:

PB8 在复位后和 nSWBOOT0 无关均为 GPIO 引脚。

9.3.16 使用 PF5 作为 GPIO

PF5 可以用作复位引脚（NRST）或 GPIO。根据用户选项字节中的 NRST_MODE 位，它切换到以下模式：

- 复位输入/输出：NRST_MODE=2’b11(默认值)或 NRST_MODE=2’b00

- 仅复位输入：NRST_MODE=2’b01
- GPIO PF5 模式：NRST_MODE=2’b10

9.4 寄存器描述

该章节详细描述 GPIO 的寄存器功能

有关寄存器位、寄存器偏移地址和复位值的汇总，请参见 GPIO 寄存器映射表；

可通过字节（8 位）、半字（16 位）或字（32 位）对 GPIO 寄存器进行访问。

GPIOA 基地址：0x4800_0000

GPIOB 基地址：0x4800_0400

GPIOC 基地址：0x4800_0800

GPIOD 基地址：0x4800_0C00

GPIOE 基地址：0x4800_1000

GPIOF 基地址：0x4800_1400

9.4.1 GPIO 端口模式寄存器（GPIOx_MODER）(x= A..F)

偏移地址:0x00

复位值：0xABFF_FFFF（端口 A）

复位值:0xFFFF_FEBF（端口 B）

复位值:0xFFFF FFFF（其他 C .. F）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
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**
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	MODEy[1:0]	RW	PA13~PA15、PB3~PB4：2’b10 其他:2’b11	y = 15..0 软件通过这些位配置相应的 IO 模式 00：输入模式 01：通用输出模式 10：复用功能模式 11：模拟模式(复位状态)

9.4.2 GPIO 端口输出类型寄存器（GPIOx_OTYPER）(x= A..F)

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	OTy	RW	0	y = 15..0

				软件配置 IO 的输出类型 0：推挽输出（复位状态) 1：开漏输出

9.4.3 GPIO 端口输出速度寄存器（GPIOx_OSPEEDR）(x= A..F)

偏移地址:0x08

复位值:0x0C00 0000（端口 A）

复位值:0x0000 0000（其他）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
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**
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	OSPEEDy[1:0]	RW	PA13：2’b11 其他：2’b00	y = 15..0 软件配置 IO 口的输出速度 00：非常低 01：低速 10：高速 11：非常高

9.4.4 GPIO 端口上拉/下拉寄存器（GPIOx_PUPDR）(x= A..F)

偏移地址:0x0C

复位值:0x6400 0000（端口 A）

复位值:0x0002 0100（端口 B）

复位值:0x0000 0000（其他）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
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**
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	PUPDy [1:0]	RW	2’b01：PA13、PA15、PB4 2’b10：PA14、PB8 2’b00：其他	y = 15..0 软件配置 IO 口上拉或者下拉 00：无上下拉 01：上拉 10：下拉 11：保留

9.4.5 GPIO 端口输入数据寄存器（GPIOx_IDR）(x= A..F)

偏移地址:0x10

复位值:0x0000 XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
R															

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	IDy	R		y = 15..0 这是只读的，读出值位对应 IO 口的状态

9.4.6 GPIO 端口输出数据寄存器（GPIOx_ODR）(x= A..F)

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
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15：0	ODy	RW	0	y = 15..0 对 GPIOx_BSRR 或 GPIOx_BRR 寄存器(x=A,B,C,D,E,F)，可以分别对各个 ODR 位进行独立的设置/清除。

9.4.7 GPIO 端口置位/复位寄存器（GPIOx_BSRR）(x= A..F)

偏移地址:0x18

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
W															

Bit	Name	R/W	Reset Value	Function
31:16	BRy	W	0	y = 15..0 读返回值是 0 0：对应的 ODRy 位不受影响 1：清除对应的 ODRy 位 注：如果同时写 BSy 和 BRy 的对应位，BSy 位起作用
15：0	BSy	W	0	y = 15..0 读返回值是 0 0：对应的 ODRy 位不受影响 1：设置对应的 ODRy 位

9.4.8 GPIO 端口配置锁定寄存器（GPIOx_LCKR）(x= A..F)

当执行正确的写序列置位 LCKK 时，该寄存器用来锁定端口位的配置。LCKR[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 LCKR[15:0]。对相应的端口执行锁定序列后，在下次系统复位前将不能再更改端口位的配置。注：特殊写时序用来写 GPIOx_LCKR 寄存器。在锁定时序中只允许字访问。

每个锁定位冻结一种特定的配置寄存器（控制和复用功能寄存器）

偏移地址:0x1C

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
-															RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	LCKK	RW	0	该位可随时读出，它只能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_LCKR 寄存器被锁定 锁键值写序列: 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 LCK[15:0]的值。锁键时序的任何错误都会终止锁键被激活。对端口的任何一位首次锁键时序之后，读 LCKK 位都是返回 1,直到系统复位或者 GPIO 复位。
15：0	LCKy	RW	0	y = 15..0 只能在 LCKK 位为 0 时写入。 0：不锁定端口的配置 1：锁定端口配置

9.4.9 GPIO 复用功能低寄存器（GPIOx_AFRL）(x= A..F)

偏移地址:0x20

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
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**
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0]	RW	4'h0	y = 7..0 配置复用功能 IO AFSELy 选择:

				0000:AF0	1000：AF8
				0001:AF1	1001：AF9
				0010:AF2	1010：AF10
				0011:AF3	1011：AF11
				0100:AF4	1100：AF12
				0101:AF5	1101：AF13
				0110:AF6	1110：AF14
				0111:AF7	1111：AF15

9.4.10 GPIO 复用功能高寄存器（GPIOx_AFRH）(x= A..F)

偏移地址:0x24

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
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**
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0]	RW	4’h0	y = 15..8 配置复用功能 IO AFSELy 选择: 0000:AF0 1000：AF8 0001:AF1 1001：AF9 0010:AF2 1010：AF10 0011:AF3 1011：AF11 0100:AF4 1100：AF12 0101:AF5 1101：AF13 0110:AF6 1110：AF14 0111:AF7 1111：AF15

9.4.11 GPIO 端口位复位寄存器（GPIOx_BRR）(x= A..F)

偏移地址:0x28

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	**14**	**13**	**2**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W															

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	BRy	W	0	y = 15..0 读返回值是 0 0：对应的 ODy 位受影响 1：清除对应的 ODy 位

10. 系统配置控制器（SYSCFG）

10.1 主要特性

SYSCFG 模块主要完成如下功能：

- I²C IO 噪声滤波器控制
- I²C Fm+模式控制
- 所有 IO 噪声滤波器控制
- EXTI IO 选择
- 模拟通道 2 控制
- 根据不同 boot 模式，映射初始程序区
- DMA 外设通道选择
- 定时器刹车输入和外部触发控制
- CCM 擦除控制,CCM 忙状态标志
- SRAM1 和 CCM 的奇偶校验控制
- CCM 页写保护

10.2 寄存器描述

10.2.1 SYSCFG 配置寄存器 1（SYSCFG_CFGR1）

Address offset：0x00

Reset value：0x0000 000x(x 取决于 BOOT0 引脚以及选项字节里启动模式配置)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2C_EIIC[15:0]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP	CTC_SOF_SEL	PTP_PPS_REMAP	TIM2ITR1_REMAP[1:0]		ETH_PHYSEL	MEM_MODE[1:0]	
-					RW	RW	RW	RW	RW	RW	RW		RW	RW	

Bit	Name	R/W	Reset Value	Function
31:16	I2C_EIIC[15:0]	RW	16'h0	I²C 相关 IO 的模拟滤波使能,每一位控制一个 IO 0：模拟滤波关闭 1：模拟滤波使能 I2C_EIIC[0]：控制 GPIOA8 的 I²C 模拟滤波使能 I2C_EIIC[1]：控制 GPIOA9 的 I²C 模拟滤波使能 I2C_EIIC[2]：控制 GPIOA13 的 I²C 模拟滤波使能 I2C_EIIC[3]：控制 GPIOA14 的 I²C 模拟滤波使能 I2C_EIIC[4]：控制 GPIOA15 的 I²C 模拟滤波使能 I2C_EIIC[5]：控制 GPIOB5 的 I²C 模拟滤波使能 I2C_EIIC[6]：控制 GPIOB7 的 I²C 模拟滤波使能 I2C_EIIC[7]：控制 GPIOB8 的 I²C 模拟滤波使能 I2C_EIIC[8]：控制 GPIOB9 的 I²C 模拟滤波使能 I2C_EIIC[9]：控制 GPIOC3 的 I²C 模拟滤波使能 I2C_EIIC[10]：控制 GPIOC6 的 I²C 模拟滤波使能 I2C_EIIC[11]：控制 GPIOC7 的 I²C 模拟滤波使能

				I2C_EIIC[12]：控制 GPIOC8 的 I2C 模拟滤波使能 I2C_EIIC[13]：控制 GPIOC9 的 I^2C 模拟滤波使能 I2C_EIIC[14]：控制 GPIOC11 的 I^2C 模拟滤波使能 I2C_EIIC[15]：控制 GPIOF0 的 I^2C 模拟滤波使能
15:11	保留	-	-	-
10	I2C4_FMP	RW	0	I2C4 Fm+模式驱动能力(Fast-mode Plus driving capability)激活 0：I2C4 被 AF 通道所选择的引脚的 Fm+驱动模式不使能 1：I2C4 被 AF 通道所选择的引脚的 Fm+驱动模式使能 该位配置成 1，会自动使得以下 IO 的驱动能力配置成最大值: PA13,PB7,PC6,PC7
9	I2C3_FMP	RW	0	I2C3 Fm+模式驱动能力(Fast-mode Plus driving capability)激活 0：I2C3 被 AF 通道所选择的引脚的 Fm+驱动模式不使能 1：I2C3 被 AF 通道所选择的引脚的 Fm+驱动模式使能 该位配置成 1，会自动使得以下 IO 的驱动能力配置成最大值: PA8,PB5,PC8,PC9,PC11
8	I2C2_FMP	RW	0	I2C2 Fm+模式驱动能力(Fast-mode Plus driving capability)激活 0：I2C2 被 AF 通道所选择的引脚的 Fm+驱动模式不使能 1：I2C2 被 AF 通道所选择的引脚的 Fm+驱动模式使能 该位配置成 1，会自动使得以下 IO 的驱动能力配置成最大值: PA8,PA9,PC3,PF0
7	I2C1_FMP	RW	0	I2C1 Fm+模式驱动能力(Fast-mode Plus driving capability)激活 0：I2C1 被 AF 通道所选择的引脚的 Fm+驱动模式不使能 1：I2C1 被 AF 通道所选择的引脚的 Fm+驱动模式使能 该位配置成 1，会自动使得以下 IO 的驱动能力配置成最大值: PA13,PA14,PA15,PB7,PB8,PB9
6	CTC_SOF_SEL	RW	0	选择输出到 ctc 的 usb sof 0：usb sof 来自 USB1 1：usb sof 来自 USB2
5	PTP_PPS_REMAP	RW	0	以太网 PTP PPS 映射 它控制以太网的 MAC PPS_PTS 是否输出到 PB5 引脚 0：PTP_PPS 不输出到 PB5 引脚 1：PTP_PPS 输出到 PB5 引脚
4:3	TIM2ITR1_REMAP[1:0]	RW	2'h0	TIM2 内部触发 1 重映射 它控制 TIM2 ITR1 的内部重映射。 00 ： 低电平 01 ： usb1_sof 10 ： usb2_sof 11 ： eth_ptp_intr
2	ETH_PHYSEL	RW	0	ETH 物理层接口(Ethernet PHY Interface)选择位

				0：PHY 接收 MII 时钟，PHY MII 接口工作 1：PHY 接收 RMII 时钟，PHY RMII 接口工作
1:0	MEM_MODE [1:0]	RW	x	启动模式选择位 控制存储器的 0x0000 0000 地址的映射。 00：Main flash，映射到 0x0000 0000 01：System flash，映射到 0x0000 0000 10：ESMC，映射到 0x0000 0000 11：SRAM，映射到 0x0000 0000

10.2.2 SYSCFG 配置寄存器 2 (SYSCFG_CFGR2)

Address offset：0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.							SPF	Res.				ECCL	PVDL	SPL	CLL
-							RC_W1	-				RS	RS	RS	RS

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	SPF	RC_W1	0	SRAM1 和 CCM SRAM 的奇偶校验错误标志位 当 SRAM1 或 CCM SRAM 的奇偶校验出现错误时，此位由硬件置 1。软件往此位写 1 即可清零 0:没有检测到奇偶校验错误 1:检测到奇偶校验错误
7:4	保留	-	-	-
3	ECCL	RS	0	ECC 锁定使能位 此位由软件置 1，且只能被系统复位清零。此位能用于使能和锁定 FLASH ECC 的错误连接到 TIM1/8/15/16/17 的刹车输入(break input) 软件写 0 到该位不会改变该位的值 0：ECC 错误不与 TIM1/8/15/16/17 的刹车输入相连 1：ECC 错误与 TIM1/8/15/16/17 的刹车输入相连
2	PVDL	RS	0	PVD 锁定使能位 软件置位，系统复位清零。它可以被用作使能和锁定 PVD 连接给 TIM1/8/15/16/17 的刹车输入，也锁定 PWR_CR 寄存器的 PVDE。 软件写 0 到该位不会改变该位的值 0：PVD 中断不与 TIM1/8/15/16/17 的刹车输入连接。PVDE 位可以被软件写入。 1：PVD 中断与 TIM1/8/15/16/17 的刹车输入连接。PVDE 位只读。
1	SPL	RS	0	SRAM1 和 CCM SRAM 奇偶校验锁定使能位 此位由软件置 1，且只能被系统复位清零。此位能用于使能和锁定 SRAM1 或 CCM SRAM 的奇偶校验错误信号连接到 TIM1/8/15/16/17 的刹车输入(break input) 软件写 0 到该位不会改变该位的值

				0：CCM SRAM 的奇偶校验错误信号不与 TIM1/8/15/16/17 的刹车输入相连 1：CCM SRAM 的奇偶校验错误信号与 TIM1/8/15/16/17 的刹车输入相连
0	CLL	RS	0	Cortex-M4F LOCKUP 使能位 软件置位，系统复位清零。它可以使能和锁定 Cortex-M4F 的 LOCKUP(hardfault)输出给 TIM1/8/15/16/17 的刹车输入。 软件写 0 到该位不会改变该位的值 0：Cortex-M4F 的 LOCKUP 输出不与 TIM1/8/15/16/17 的刹车输入连接； 1：Cortex-M4F 的 LOCKUP 输出与 TIM1/8/15/16/17 的刹车输入连接；

10.2.3 SYSCFG 配置寄存器 3（SYSCFG_CFGR3）

Address offset:0x8

Reset value:0x7F7F 7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DMA1_MAP4[6:0]							Res.	DMA1_MAP3[6:0]						
-	RW							-	RW						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DMA1_MAP2[6:0]							Res.	DMA1_MAP1[6:0]						
-	RW							-	RW						

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:24	DMA1_MAP4[6:0]	RW	7’h7F	DMA1 通道请求映射。详见 DMA 章节
23	保留	-	-	-
22:16	DMA1_MAP3[6:0]	RW	7’h7F	DMA1 通道请求映射。详见 DMA 章节
15	保留	-	-	-
14:8	DMA1_MAP2[6:0]	RW	7’h7F	DMA1 通道请求映射。详见 DMA 章节
7	保留	-	-	-
6:0	DMA1_MAP1[6:0]	RW	7’h7F	DMA1 通道请求映射。详见 DMA 章节

10.2.4 SYSCFG 配置寄存器 4（SYSCFG_CFGR4）

Address offset:0XC

Reset value:0x7F7F 7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DMA2_MAP2[6:0]							Res.	DMA2_MAP1[6:0]						
-	RW							-	RW						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DMA1_MAP6[6:0]							Res.	DMA1_MAP5[6:0]						
-	RW							-	RW						

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-

30:24	DMA2_MAP2[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节
23	保留	-	-	-
22:16	DMA2_MAP1[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节
15	保留	-	-	-
14:8	DMA1_MAP6[6:0]	RW	7'h7F	DMA1 通道请求映射。详见 DMA 章节
7	保留	-	-	-
6:0	DMA1_MAP5[6:0]	RW	7'h7F	DMA1 通道请求映射。详见 DMA 章节

10.2.5 SYSCFG 配置寄存器 5（SYSCFG_CFGR5）

Address offset:0x10

Reset value：0x7F7F 7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DMA2_MAP6[6:0]							Res	DMA2_MAP5[6:0]						
-	RW							-	RW						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DMA2_MAP4[6:0]							Res	DMA2_MAP3[6:0]						
-	RW							-	RW						

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:24	DMA2_MAP6[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节
23	保留	-	-	-
22:16	DMA2_MAP5[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节
15	保留	-	-	-
14:8	DMA2_MAP4[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节
7	保留	-	-	-
6:0	DMA2_MAP3[6:0]	RW	7'h7F	DMA2 通道请求映射。详见 DMA 章节

10.2.6 外部中断配置寄存器 1（SYS_EXTICR1）

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	EXTIx[3:0]	RW	4'h0	EXTIx 配置(x = 0~3)（EXTI x configuration) 用于选择 EXTIx 外部中断的输入源。详见 EXTI 外部中断事件映射。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚

				0011：PD[x]引脚 0100：PE[x]引脚 0101：PF[x]引脚

10.2.7 外部中断配置寄存器 2（SYS_EXTICR2）

Address offset:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	EXTIx[3:0]	RW	4’h0	EXTIx 配置(x =4 … 7)（EXTI x configuration) 用于选择 EXTIx 外部中断的输入源。详见 EXTI 外部中断事件映射。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚 0100：PE[x]引脚 0101：PF[x]引脚(仅当当 x=4,5)

10.2.8 外部中断配置寄存器 3（SYS_EXTICR3）

Address offset:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	EXTIx[3:0]	RW	4’h0	EXTIx 配置(x = 8 … 11)（EXTI x configuration) 用于选择 EXTIx 外部中断的输入源。详见 EXTI 外部中断事件映射。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚 0100：PE[x]引脚

10.2.9 外部中断配置寄存器 4（SYS_EXTICR4）

Address offset:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	EXTIx[3:0]	RW	4’h0	EXTIx 配置(x = 12 … 15)（EXTI x configuration) 用于选择 EXTIx 外部中断的输入源。详见 EXTI 外部中断事件映射。 0000：PA[x]引脚 0001：PB[x]引脚 0010：PC[x]引脚 0011：PD[x]引脚 0100：PE[x]引脚

10.2.10 GPIOA 滤波使能寄存器（PA_ENS）

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PA_ENS[15:0]	RW	16’h0	GPIOA0~15 的噪声滤波使能 0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.11 GPIOB 滤波使能寄存器（PB_ENS）

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PB_ENS[15:0]	RW	16’h0	GPIOB0~15 的噪声滤波使能

				0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.12 GPIOC 滤波使能寄存器（PC_ENS）

Address offset:0x2C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PC_ENS[15:0]	RW	16’h0	GPIOC0~15 的噪声滤波使能 0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.13 GPIOD 滤波使能寄存器（PD_ENS）

Address offset:0x30

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PD_ENS[15:0]	RW	16’h0	GPIOD0~15 的噪声滤波使能 0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.14 GPIOE 滤波使能寄存器（PE_ENS）

Address offset:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PE_ENS[15:0]	RW	16’h0	GPIOE0~15 的噪声滤波使能 0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.15 GPIOF 滤波使能寄存器（PF_ENS）

Address offset:0x38

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	PF_ENS[5:0]	RW	6'h0	GPIOF0~5 的噪声滤波使能 0：关闭噪声滤波功能 1：开启噪声滤波功能

10.2.16 GPIOA 模拟通道 2 使能寄存器(PA_ANA2ENR)

Address offset:0x40

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					PA10_ ENA2	Res.		PA7_ ENA2	Res.	PA5_ ENA2	PA4_ ENA2	PA3_ ENA2	PA2_ ENA2	PA1_ ENA2	PA0_ ENA2
-					RW	-		RW	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	-
10	PA10_ENA2	RW	0	PA10 模拟通道 2 使能：USB1_OTG_FS_ID 0：不使能 1：使能
9:8	保留	-	-	-
7	PA7_ENA2	RW	0	PA7 模拟通道 2 使能：COMP2_INP 0：不使能 1：使能
6	保留	-	-	-
5	PA5_ENA2	RW	0	PA5 模拟通道 2 使能：COMP2_INM, 0：不使能 1：使能
4	PA4_ENA2	RW	0	PA4 模拟通道 2 使能：COMP1_INM 0：不使能 1：使能
3	PA3_ENA2	RW	0	PA3 模拟通道 2 使能：COMP2_INP 0：不使能 1：使能
2	PA2_ENA2	RW	0	PA2 模拟通道 2 使能：COMP2_INM

				0：不使能 1：使能
1	PA1_ENA2	RW	0	PA1 模拟通道 2 使能：COMP1_INP 0：不使能 1：使能
0	PA0_ENA2	RW	0	PA0 模拟通道 2 使能： COMP1_INM,COMP3_INP 0：不使能 1：使能

10.2.17 GPIOB 模拟通道 2 使能寄存器(PB_ANA2ENR)

Address offset:0x44

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.													PB2_ENA2	PB1_ENA2	PB0_ENA2
-													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	PB2_ENA2	RW	0	PB2 模拟通道 2 使能：COMP4_INM 0：不使能 1：使能
1	PB1_ENA2	RW	0	PB1 模拟通道 2 使能：COMP1_INP 0：不使能 1：使能
0	PB0_ENA2	RW	0	PB0 模拟通道 2 使能：COMP4_INP 0：不使能 1：使能

10.2.18 GPIOC 模拟通道 2 使能寄存器(PC_ANA2ENR)

Address offset:0x48

Reset value:0x0000 C000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC15_ENA2	PC14_ENA2	Res.										PC3_ENA2	PC2_ENA2	PC1_ENA2	PC0_ENA2
RW	RW	-										RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	PC15_ENA2	RW	1	PC15 模拟通道 2 使能：LSE(OSC32) 0：不使能 1：使能
14	PC14_ENA2	RW	1	PC14 模拟通道 2 使能：LSE(OSC32)

				0：不使能 1：使能
13:4	保留	-	-	-
3	PC3_ENA2	RW	0	GPIOC3 模拟通道 2 使能：ADC1_CH9/ADC2_CH9 0：不使能 1：使能
2	PC2_ENA2	RW	0	PC2 模拟通道 2 使能： ADC1_CH8/ADC2_CH8/USB2_OTG_FS_ID 0：不使能 1：使能
1	PC1_ENA2	RW	0	PC1 模拟通道 2 使能：COMP3_INP 0：不使能 1：使能
0	PC0_ENA2	RW	0	PC0 模拟通道 2 使能：COMP3_INM 0：不使能 1：使能

10.2.19 GPIOD 模拟通道 2 使能寄存器(PD_ANA2ENR)

Address offset:0x4C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		PD13_ENA2	PD12_ENA2	PD11_ENA2	PD10_ENA2	Res.									
-		RW	RW	RW	RW	-									

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13	PD13_ENA2	RW	0	PD13 模拟通道 2 使能：ADC3_CH10 0：不使能 1：使能
12	PD12_ENA2	RW	0	PD12 模拟通道 2 使能：ADC3_CH9 0：不使能 1：使能
11	PD11_ENA2	RW	0	PD11 模拟通道 2 使能：ADC3_CH8 0：不使能 1：使能
10	PD10_ENA2	RW	0	PD10 模拟通道 2 使能：ADC3_CH7 0：不使能 1：使能
9:0	保留	-	-	-

10.2.20 GPIOE 模拟通道 2 使能寄存器(PE_ANA2ENR)

Address offset:0x50

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.			PE12_ENA2	PE11_ENA2	PE10_ENA2	Res.		PE7_ENA2	Res.						
-			RW	RW	RW	-		RW	-						

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	PE12_ENA2	RW	0	PE12 模拟通道 2 使能：ADC3_CH15 0：不使能 1：使能
11	PE11_ENA2	RW	0	PE11 模拟通道 2 使能：ADC3_CH14 0：不使能 1：使能
10	PE10_ENA2	RW	0	PE10 模拟通道 2 使能：ADC3_CH13 0：不使能 1：使能
9:8	保留	-	-	-
7	PE7_ENA2	RW	0	PE7 模拟通道 2 使能：COMP4_INP 0：不使能 1：使能
6:0	保留	-	-	-

10.2.21 PVD_IN 模拟通道 2 使能寄存器(PVDIN_ANA2ENR)

Address offset:0x54

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															PVDIN_ENA2
-															RW

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	PVDIN_ENA2	RW	0	使能 PVD_IN 引脚(PB7)的模拟通道 2

10.2.22 SYSCFG CCM SRAM 控制与状态寄存器(SYSCFG_SCSR)

Address offset：0x70

Reset value：0x0000_0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.														CCMBSY	CCMER
-														R	W

Bit	Name	R/W	Reset Value	Function
31：2	保留	-	-	-
1	CCMBSY	R	0	CCM SRAM 擦除标志位 0:无 CCM SRAM 擦除操作正在进行 1：CCM SRAM 擦除操作正在进行
0	CCMER	W	0	CCM SRAM 擦除 将该位设为 1 将会启动 CCM SRAM 的硬件擦除操作，该位会被硬件清零

10.2.23 SYSCFG CCM SRAM 写保护寄存器（SYSCFG_SWPR)

Address offset：0x74

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS

Bit	Name	R/W	Reset Value	Function
31:0	PxWP(x=0~31)	RS	0	CCM SRAM 第 x 页(page)写保护 这些位由软件置 1，且只能被系统复位清零。 0:CCM SRAM 第 x 页写保护关闭 1:CCM SRAM 第 x 页写保护使能

10.2.24 SYSCFG CCM SRAM 键值寄存器（SYSCFG_SKR)

Address offset：0x78

Reset value：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
KEY[7:0]	KEY[7:0]	W	8’h0	软件擦除 CCM SRAM 的写保护键值 只有遵循以下的步骤，才能解除 SYSCFG_SCSR 寄存器的 CCMER 位的写保护: 1. 写”0xCA”到本寄存器的 KEY[7:0] 2. 写”0x53”到本寄存器的 KEY[7:0] 若写入序列错误，将重新激活写保护

11. 外设互联矩阵

11.1 简介

一些外围设备之间有连接关系。

这允许在外设之间进行自主通信，从而节省 CPU 资源，同时也节省电源消耗。此外这些硬件连接消除了软件延迟。根据外设的不同，这些互联可以在运行、睡眠和停止的模式下运行。

源	目标																																
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM8	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14	TIM15	TIM16	TIM17	TIM18	TIM19	LPTIM	ADC1	ADC2	ADC3	DAC1	DAC2	COMP1	COMP2	COMP3	COMP4	IRTIM
TIM1	-	1	1	1	1			1	1	1	1	1	1	1	1	1	1	1	1		13	13	13	14	14	10	10	10	10	
TIM2	1	-	1	-	1			1	1	1	1	1	1	1	1	1	1	1	1		13	13	13	14	14	10	10	10	10	
TIM3	1	1		1	1			1	1	1	1	1	1	1	1	1	1	1	1		13	13	13	14	14	10	10	10	10	
TIM4	1	1	1		1			1	1	1	1	1	1	1	1	1	1	1	1		13	13	13	14	14	10	10	10	10	
TIM5	1	1	1	1				1	1	1	1	1	1	1	1	1	1	1	1					14	14					
TIM6																					13	13	13	14	14					
TIM7																					13	13	13	14	14					
TIM8	1	1	1	1	1				1	1	1	1	1	1	1	1	1	1	1		13	13	13	14	14	10	10	10	10	
TIM9	1	1	1	1	1			1		1	1	1	1	1	1	1	1	1	1		13	13	13							
TIM10	1	1	1	1	1			1	1		1	1	1	1	1	1	1	1	1		13	13	13							
TIM11	1	1	1	1	1			1	1	1		1	1	1	1	1	1	1	1		13	13								
TIM12	1	1	1	1	1			1	1	1	1		1	1	1	1	1	1	1											
TIM13	1	1	1	1	1			1	1	1	1	1		1	1	1	1	1	1				13							
TIM14	1	1	1	1	1			1	1	1	1	1	1		1	1	1	1	1				13							
TIM15	1	1	1	1	1			1	1	1	1	1	1	1		1	1	1	1		13	13		14	14	10	10	10	10	
TIM16	1	1	1	1	1			1	1	1	1	1	1	1	1		1	1	1											11
TIM17	1	1	1	1	1			1	1	1	1	1	1	1	1	1		1	1				13							11
TIM18	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1		1		13	13		14	14	10	10	10	10	
TIM19	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1	1			13	13								
LPTIM																														
ADC1	2	-	2	-																										
ADC2								2																						
ADC3								2																						
DAC1																														
DAC2																														
HSE																4	4													
LSE					4											4	4													
HSI16																														
LSI					4											4	4													
MCO																4	4													
EXTI																								14	14					
RTC					4											4	4													
COMP1	4 2 3 8 9	4 2 3 5	4 2 3 5 7	4 2 3 5	4 2 3 5			4 2 3 8 9							3 4 8	3 8	3 8	4 2 3 5												
COMP2	4 2 3 8 9	4 2 3 5	4 2 3 5 7	4 2 3 5 6	4 2 3 5			4 2 3 8 9							3 4 5 8	3 8	3 8	4 2 3 5												
COMP3	4 2	4 2	4 2	4 2	4 2			4 2							3 5	3 8	3 8	4 2												

	3 8 9	3 5	3 5 6	3 5	3 5			3 8 9							8			3 5											
COMP4	4 2 3 8 9	4 2 3 5 6	4 2 3 5	4 2 3 5	4 2 3 5			4 2 3 8 9							3 5 8	3 8	3 8	4 2 3 5											
SYST ERR	1 2							1 2							1 2	1 2	1 2												

11.2 互联详情

11.2.1 Input trigger(itr)

表 11-1 Interconnect 1

Timer input trigger signal	Timer 输入触发源分配									
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM9	TIM12	TIM15	TIM18
Timx_itr0	-	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo	Tim1_trgo
Timx_itr1	Tim2_trgo	-	Tim2_trgo	Tim2_trgo	Tim2_trgo	Tim2_trgo	Tim2_trgo	Tim2_trgo	Tim2_trgo	Tim2_trgo
Timx_itr2	Tim3_trgo	Tim3_trgo	-	Tim3_trgo	Tim3_trgo	Tim3_trgo	Tim3_trgo	Tim3_trgo	Tim3_trgo	Tim3_trgo
Timx_itr3	Tim4_trgo	Tim4_trgo	Tim4_trgo	-	Tim4_trgo	Tim4_trgo	Tim4_trgo	Tim4_trgo	Tim4_trgo	Tim4_trgo
Timx_itr4	Tim5_trgo	Tim5_trgo	Tim5_trgo	Tim5_trgo	-	Tim5_trgo	Tim5_trgo	Tim5_trgo	Tim5_trgo	Tim5_trgo
Timx_itr5	Tim8_trgo	Tim8_trgo	Tim8_trgo	Tim8_trgo	Tim8_trgo	-	Tim8_trgo	Tim8_trgo	Tim8_trgo	Tim8_trgo
Timx_itr6	Tim9_oc1	Tim9_oc1	Tim9_oc1	Tim9_oc1	Tim9_oc1	Tim9_oc1	-	Tim9_oc1	Tim9_oc1	Tim9_oc1
Timx_itr7	Tim9_oc2	Tim9_oc2	Tim9_oc2	Tim9_oc2	Tim9_oc2	Tim9_oc2	-	Tim9_oc2	Tim9_oc2	Tim9_oc2
Timx_itr8	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc	Tim10_oc
Timx_itr9	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc	Tim11_oc
Timx_itr10	Tim12_oc1	Tim12_oc1	Tim12_oc1	Tim12_oc1	Tim12_oc1	Tim12_oc1	Tim12_oc1	-	Tim12_oc1	Tim12_oc1
Timx_itr11	Tim12_oc2	Tim12_oc2	Tim12_oc2	Tim12_oc2	Tim12_oc2	Tim12_oc2	Tim12_oc2	-	Tim12_oc2	Tim12_oc2
Timx_itr12	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc	Tim13_oc
Timx_itr13	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc	Tim14_oc
Timx_itr14	Tim15_trgo	Tim15_trgo	Tim15_trgo	Tim15_trgo	Tim15_trgo	Tim15_trgo	Tim15_trgo	Tim15_trgo	-	Tim15_trgo
Timx_itr15	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc	Tim16_oc
Timx_itr16	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc	Tim17_oc
Timx_itr17	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	Tim18_trgo	-
Timx_itr18	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc	Tim19_oc

一些 TIMx 定时器在内部连接在一起，用于定时器的同步或者连接。当一个定时器为主模式时，它可以重置、启动、停止从模式定时器。

11.2.2 External trigger(etr)

表 11-2 Interconnect 2

Timer 外部触发输入信号	Timer 外部触发信号源分配						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM18
Timx_etr0	TIM1_ETR	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR	TIM8_ETR	TIM18_ETR
Timx_etr1	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out
Timx_etr2	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out
Timx_etr3	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out

Timx_etr4	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out
Timx_etr5	adc1_awd1	TIM3_ETR	TIM2_ETR	TIM3_ETR	TIM2_ETR	adc2_awd1	TIM3_ETR
Timx_etr6	adc1_awd2	TIM4_ETR	TIM4_ETR	TIM5_ETR	TIM3_ETR	adc2_awd2	TIM4_ETR
Timx_etr7	adc1_awd3	TIM5_ETR	-	TIM18_ETR	TIM18_ETR	adc2_awd3	TIM5_ETR
Timx_etr8		TIM18_ETR	adc1_awd1			adc3_awd1	
Timx_etr9			adc1_awd2			adc3_awd2	
Timx_etr10			adc1_awd3			adc3_awd3	

比较器的输出可以被连接到定时器（TIMx）输入捕获或者 TIMx_ETR 信号或者 TIMx_OCREFCLR 信号，也可以产生定时器的刹车输入信号。

11.2.3 Ocref clear

表 11-3 Interconnect 3

Timer ocref 清除信号	Timer ocref 清除信号分配							
	TIM1	TIM2	TIM3	TIM8	TIM15	TIM16	TIM17	TIM18
timx_ocref_clr0	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out
timx_ocref_clr1	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out
timx_ocref_clr2	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out
timx_ocref_clr3	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out

11.2.4 TI1

表 11-4 Interconnect 4

Timer TI1 输入信号	Timer TI1 信号分配									
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM15	TIM16	TIM17	TIM18
timx_ti1_in0	TI1	TI1	TI1	TI1	TI1	TI1	TI1	TI1	TI1	TI1
timx_ti1_in1	Comp1_out	Comp1_out	Comp1_out	Comp1_out	LSI	Comp1_out				Comp1_out
timx_ti1_in2	Comp2_out	Comp2_out	Comp2_out	Comp2_out		Comp2_out	Comp1_out	MCO	MCO	Comp2_out
timx_ti1_in3	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Rtc_wakeup	Comp3_out	Comp2_out	HSE_Div32	HSE_Div32	Comp3_out
timx_ti1_in4	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp1_out	Comp4_out				Comp4_out
timx_ti1_in5					Comp2_out					
timx_ti1_in6					Comp3_out			LSI	LSI	
timx_ti1_in7					Comp4_out					

11.2.5 TI2

表 11-5 Interconnect 5

Timer T2 输入信号	Timer TI2 信号分配							
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM15	TIM18
timx_ti2_in0	TI2	TI2	TI2	TI2	TI2	TI2	TI2	TI2
timx_ti2_in1		Comp1_out	Comp1_out	Comp1_out	Comp1_out		Comp2_out	Comp1_out
timx_ti2_in2		Comp2_out	Comp2_out	Comp2_out	Comp2_out		Comp3_out	Comp2_out
timx_ti2_in3		Comp3_out	Comp3_out	Comp3_out	Comp3_out			Comp3_out
timx_ti2_in4		Comp4_out	Comp4_out	Comp4_out	Comp4_out			Comp4_out

11.2.6 TI3

表 11-6 Interconnect 6

Timer TI3 输入信号	Timer TI3 信号分配						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM18
timx_ti3_in0	TI3	TI3	TI3	TI3	TI3	TI3	TI3
timx_ti3_in1		Comp4_out	Comp3_out	Comp2_out			

11.2.7 TI4

表 11-7 Interconnect 7

Timer TI4 输入信号	Timer TI4 信号分配						
	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM18
timx_ti4_in0	TI4	TI4	TI4	TI4	TI4	TI4	TI4
timx_ti4_in1		Comp1_out					
timx_ti4_in1		Comp2_out					

11.2.8 break1

表 11-8 Interconnect 8

	Timer 刹车 1 信号分配				
	TIM1_break	TIM8_break	TIM15_break	TIM16_break	TIM17_break
Timer 刹车 1 信号源	TIM1_BKINpin	TIM8_BKINpin	TIM15_BKINpin	TIM16_BKINpin	TIM17_BKINpin
	Comp1_out	Comp1_out	Comp1_out	Comp1_out	Comp1_out
	Comp2_out	Comp2_out	Comp2_out	Comp2_out	Comp2_out
	Comp3_out	Comp3_out	Comp3_out	Comp3_out	Comp3_out
	Comp4_out	Comp4_out	Comp4_out	Comp4_out	Comp4_out

定时器的刹车输入有两个通道：

1. break1 channel ,它包括应用程序故障和系统故障。

2. break2 channel 仅包括应用程序故障。

11.2.9 break2

表 11-9 Interconnect 9

	Timer 刹车 2 信号分配	
	TIM1_break2	TIM8_break2
Timer 刹车 2 信号源	TIM1_BKIN2pin	TIM8_BKIN2pin
	Comp1_out	Comp1_out
	Comp2_out	Comp2_out
	Comp3_out	Comp3_out
	Comp4_out	Comp4_out

11.2.10 比较器消隐源

表 11-10 Interconnect 10

	比较器消隐源分配			
	COMP1	COMP2	COMP3	COMP4
比较器消隐源	tim1_oc5	tim1_oc5	tim1_oc5	tim3_oc4
	tim2_oc3	tim2_oc3	tim3_oc3	tim8_oc5
	tim3_oc3	tim3_oc3	tim2_oc4	tim15_oc2
	tim8_oc5	Tim8_oc5	Tim8_oc5	tim1_oc5
	tim18_oc3	tim18_oc3	tim18_oc4	tim18_oc4
	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1
	tim4_oc3	tim4_oc3	Tim4_oc3	tim4_oc3

TIMx 可以用作 COMPx 的消隐窗口。

定时器的输出信号 TIMx_OCx 是 COMPx 的消隐源输入。

11.2.11 IRTIM control signal

表 11-11 Interconnect 11

IRTIM 控制信号	IRTIM 控制信号分配
调制包络信号	Tim16_oc1
载波信号	Tim17_oc1

通用定时器（TIM16/TIM17）输出通道 TIMx_OC1 用于产生红外信号的波形输出。

11.2.12 System error

表 11-12 Interconnect 12

系统错误信号源	系统错误信号分配给 Timer 信号				
	TIM1	TIM8	TIM15	TIM16	TIM17
Flash ECC error	TIM1_BK	TIM8_BK	TIM15_BK	TIM16_BK	TIM17_BK
PVD output					
SRAM1/CCM parity					
Cortex-M4F Lockup					
CSS					

12. DMA 控制器（DMA）

12.1 概述

直接存储器访问（DMA）用来提供在外设和存储器之间、存储器和存储器之间或者外设和外设之间的高速数据传输。搬移数据无需 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 控制器共有 12 个通道（DMA1 有 6 个通道，DMA2 有 6 个通道），每个通道专门用来管理来自于一个或多个外设访问请求。DMA 还有一个仲裁器来协调各个 DMA 请求的优先权。

12.1.1 主要特性

主要功能如下：

- 单 AHB master
- 有可编程的通道优先级
- 支持外设到存储器，存储器到外设，存储器到存储器和外设到外设的数据传输
- 支持片上存储器设备（如 Flash，SRAM, AHB 和 APB 外设）作为源和目标
- 支持可编程的源和目标地址，地址可选递增，递减或不变
- 可以通过通道寄存器自动重载和块之间的连续地址实现多块传输功能
- 支持可编程的突发事务大小
- 每个通道生成一个中断请求。每个中断请求都是由六个 DMA 事件中的任何一个引起的:传输完成、块传输完成、源事务完成、目标事务完成、传输错误或半块传输完成
- 通道 0~4 的 FIFO 深度为 8 字节，通道 5 的 FIFO 深度为 32 字节

12.1.2 基本定义

- 源外设——位于 AHB 层上的设备，DMA 可以从中读取数据，然后 DMA 将数据存储在通道 FIFO 中。源外设与目标外设一起形成通道。源外设是 AHB 或 APB 从机。如果源是 APB 从机，则通过 AHB-APB 桥对其访问。
- 目标外设——DMA 将 FIFO 中存储的数据 (之前从源外设读取) 写入到的设备。目标外设是 AHB 或 APB 从机。如果目标是 APB 从机，则通过 AHB-APB 桥对其进行访问。
- 存储器——对于 DMA 传输来说，源或目标总是“准备好”的，并且不需要与 DMA 有握手接口。
- 通道——在一个已配置的 AHB 层上的源外设和位于相同或不同 AHB 层上的目标外设之间的通过 FIFO 发生读/写数据的路径。如果源外设不是存储器，那么将源握手接口分配给该通道。如果目标外设不是存储器，则将目标握手接口分配给该通道。源和目标握手接口可以通过编程通道寄存器动态分配。
- 流量控制器——确定 DMA 块传输长度并终止它的设备（DMA 源或目标外设）。
- 传输层次结构——图 12-1 展示了非存储器外设的 DMA 传输、块传输、事务 (单个或突发) 和 AHB 传输（单个或突发）之间的层次结构

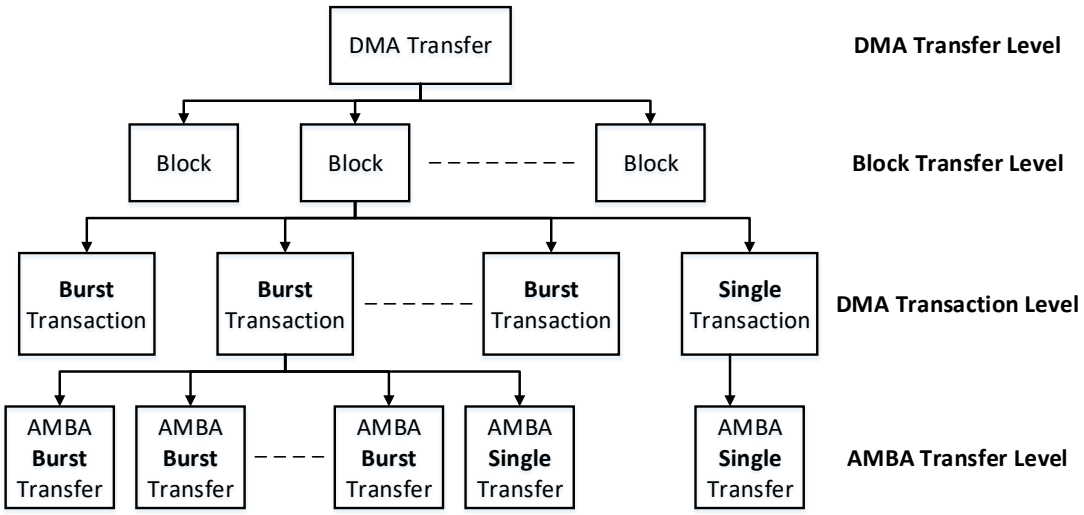


图 12-1 非存储器外设的 DMA 传输层次结构

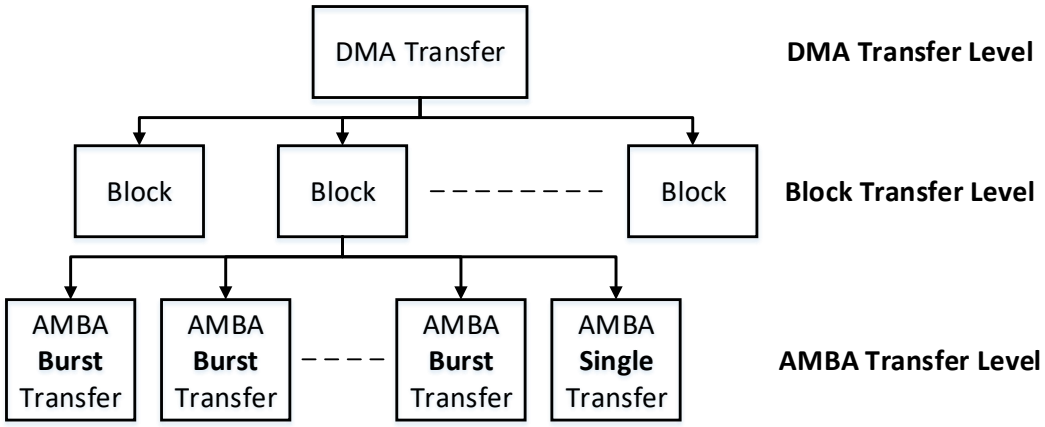
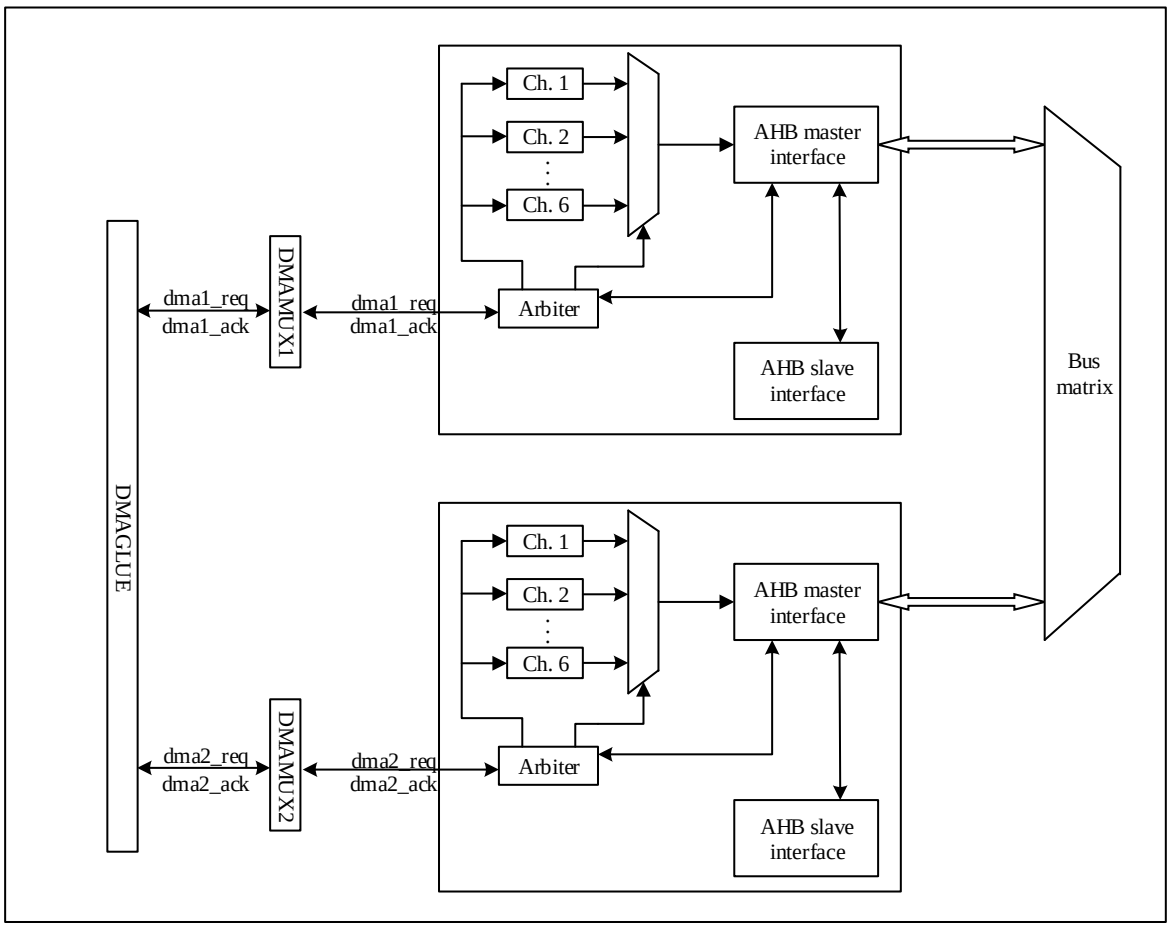


图 12-2 存储器的 DMA 传输层次结构

- 块——DMA 数据的块，其数量是块长度，由流量控制器决定。对于 DMA 和存储器之间的传输，一个块被直接分解为一系列突发和单个传输。对于 DMA 和非存储器外设之间的传输，一个块被分解成 DMA 事务序列（单个和突发）。
- 事务——DMA 传输的基本单位，由硬件或软件握手接口决定。如果外设是非存储器设备，那么事务只与 DMA 的源或目标外设之间的传输相关。
- 有以下两种事务：
  - 单个事务——单个事务的长度总是 1 并转换成单个 AHB 传输。
  - 突发事务——突发事务的长度可以被配置。突发事务被转换成一系列突发和 AHB 单个传输。突发事务长度由程序控制，通常与 DMA、源外设和目标外设的 FIFO 大小有一定的关系。
- DMA 传输——软件控制 DMA 传输的块数量。DMA 传输完成后，DMA 中的硬件禁用通道并生成一个中断来表示 DMA 传输完成。然后可以为新的 DMA 传输重新编程通道。
  - 单块 DMA 传输——由单个块组成。
  - 多块 DMA 传输——DMA 传输可以由多个 DMA 块组成。通过自动重新加载通道寄存器和连续块支持多块 DMA 传输。
    - 自动重新加载——DMA 自动在每个块的末尾重新加载通道首次启用时的值到通道寄存器。

- 连续块——选择连续块之间的地址作为前一个块末尾的延续。

12.1.3 顶层结构图



DMA 和 CPU 共用系统总线。如果 DMA 和 CPU 访问同一目标（存储器或者外设），DMA 请求会停止 CPU 访问系统总线一些时钟周期。总线矩阵仲裁用 round-robin 方式，保证 CPU 占据至少一半系统总线带宽。

12.2 功能描述

12.2.1 接口定义

源单个事务大小（以字节为单位）： src_single_size_bytes = DMA_CH_CTRLLx.SRC_TR_WIDTH/8

源突发事务大小（以字节为单位）： src_burst_size_bytes = DMA_CH_CTRLLx.SRC_MSIZE * src_single_size_bytes

目标单个事务大小（以字节为单位）： dst_single_size_bytes = DMA_CH_CTRLLx.DST_TR_WIDTH/8

目标突发事务大小（以字节为单位）： dst_burst_size_bytes = DMA_CH_CTRLLx.DST_MSIZE * dst_single_size_bytes

块大小（以字节为单位）： blk_size_bytes_dma = DMA_CH_CTRLHx.BLOCK_TS * src_single_size_bytes

12.2.2 存储器外设

DMA_CH_CTRLLx.SRC_MSIZE 和 DMA_CH_CTRLLx.DST_MSIZE 仅对具有握手接口的外设有效，它们不能用于定义存储器外设的突发长度。

因此：

- 到存储器的突发传输的长度始终等于通道 FIFO 中可用数据项的数量或完成块传输所需数据项的数量，以较小者为准。
- 来自存储器的突发传输的长度始终等于通道 FIFO 中可用空间或完成块传输所需数据项的数量，以较小者为准。

12.2.3 DMA 传输

DMA 块传输可以从外设请求或者在存储器到存储器传输的情况下由软件触发。

事件发生后，单个 DMA 传输步骤:

- 外设向 DMA 控制器发送一个 DMA 请求信号。
- DMA 控制器根据与这个外设请求相关联的通道的优先级响应请求。
- 一旦 DMA 控制器授予外设，则启动传输，当传输完成，DMA 控制器就向外设发送一个应答。
- 当外设从 DMA 控制器获得应答时，就释放它的请求。
- 一旦外设取消了请求断言，DMA 控制器就释放当前外设占有的通道。

当外设是传输的源或目标时，使用请求/确认协议。例如，在存储器到外设的传输中，外设通过将其单个请求信号驱动到 DMA 控制器来启动传输。DMA 控制器然后将 FIFO 中的数据写入外设。

对于给定的通道 x, DMA 块传输由以下重复序列组成:

- 从外设数据寄存器或存储器中的某个位置读取的数据，数据位宽在 DMA_CH_CTRLLx.SRC_TR_WIDTH 寄存器中配置，数据长度在 DMA_CH_CTRLLx.SRC_MSIZE 寄存器中配置，通过内部当前源地址寄存器寻址，用于第一次传输的起始地址在 DMA_SARx 寄存器中配置。
- 数据写入到外设数据寄存器或存储器中的某个位置，数据位宽在 DMA_CH_CTRLLx.DST_TR_WIDTH 寄存器中配置，数据长度在 DMA_CH_CTRLLx.DST_MSIZE 寄存器中配置，通过内部当前目标地址寄存器进行寻址，用于第一次传输的起始地址在 DMA_DARx 寄存器中配置。
- DMA_CTLx.BLOCK_TS 寄存器包含剩余要传输的数据项的数量，DMA 启动传输之后该寄存器指示已经从源读取的数据项的总数。

重复以上序列，直到达到设定的 DMA_CTLx.BLOCK_TS 寄存器的值。

12.2.4 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。

当一个请求通道 x 被仲裁器授予（硬件请求或软件触发）访问权限时，就会发出单个或突发 DMA 传输。然后，仲裁器再次仲裁所有请求通道，并选择优先级最高的通道。

每个通道的优先权可以在 DMA_CH_CFGx.寄存器中设置，有 6 个等级：

- ✓ 0
- ✓ 1
- ✓ 2
- ✓ 3
- ✓ 4
- ✓ 5

优先级 5>4>3>2>1>0。

相同优先级时通道号较低的通道具有更高优先级。

12.2.5 DMA 通道

12.2.5.1 传输数据量可编程

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进行 DMA 传输。DMA 传输数据量是可编程的，最大到 4095。

12.2.5.2 地址指针增量

通过设置 DMA_CH_CTRLLx 寄存器中的 SINC 和 DINC 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。

当设置为递增模式时，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 8/16/32bit。第一个传输的地址是存放在 DMA_SAx/DMA_DAx 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（在内部的当前外设/存储器地址寄存器中）。

当通道配置为单块传输时，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_CH_CTRLHx.BLOCK_TS 寄存器中重新写入传输数目。

12.2.5.3 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_CH_CTRLLx.TT 寄存器之后，在软件设置了 DMA_CH_ENx 寄存器中的 CH_EN 位启动 DMA 通道时，DMA 传输将马上开始。当传输数据个数达到设定的 DMA_CH_CTRLHx.BLOCK_TS 寄存器的值时，DMA 传输结束。

12.2.6 错误处理

在 AHB 传输中出现错误时，会发生以下情况：

1. 正在进行的 DMA 传输立即停止
2. 相关通道被禁用
3. 发出中断（如果没有被屏蔽）

如果开启了多个通道，只关闭检测到 AHB 错误的通道。

FIFO 的内容不会被清除，一旦通道重新启用以启动新传输，就变得不可访问并被覆盖。

不支持从发生错误的点自动恢复传输，想要重新传输必须重新启动整个块传输才行。

12.2.7 中断

对于每个通道，有六种类型的中断源:

- 块传输完成中断

此中断是在 DMA 块传输完成时产生的。

- 目标事务完成中断

此中断是在从目标的握手接口（硬件或软件）完成请求的单个/突发事务的最后一次 AHB 传输后生成的。

如果通道的目标是存储器，那么该通道将永远不会生成目标事务完成中断。因此，该中断的对应使能不会起作用。

- 错误中断

当在 DMA 传输期间从总线上的 AHB 从机接收到错误响应时产生这个中断。此外，产生错误中断还会取消 DMA 传输并且禁用通道。

- 源事务完成中断

此中断是在从源的握手接口（硬件或软件）完成请求的单个/突发事务的最后一次 AHB 传输后生成的。

如果源是存储器，那么源事务完成中断应该被忽略，因为存储器没有“DMA Transaction Level”的概念。

- DMA 传输完全中断

此中断是在完成到目标外设的 DMA 传输时产生的。

- 半块传输完成中断

此中断是在 DMA 块的一半传输完成到目标外设时产生的。

12.2.8 DMA 外设请求映射

DMA 外设请求映射到 DMA1（6 通道）或者 DMA2（6 通道）的各个通道，由 SYSCFG 的 DMAx_MAP 寄存器控制，每个外设请求通过配置可以映射到 12 个通道中的任意一个。

序号与外设请求的关系如下表所示:

请求 MUX 输入序号	源	请求 MUX 输入序号	源	请求 MUX 输入序号	源
0	ADC1	32	TIM1_TRIG	64	TIM5_CH4
1	ADC2	33	TIM1_COM	65	TIM5_UP
2	ADC3	34	TIM8_CH1	66	TIM5_TRIG
3	DAC1	35	TIM8_CH2	67	TIM18_CH1
4	DAC2	36	TIM8_CH3	68	TIM18_CH2
5	SPI1_RX	37	TIM8_CH4	69	TIM18_CH3
6	SPI1_TX	38	TIM8_UP	70	TIM18_CH4
7	SPI2_RX	39	TIM8_TRIG	71	TIM18_UP
8	SPI2_TX	40	TIM8_COM	72	TIM18_TRIG
9	SPI3_RX	41	TIM6_UP	73	TIM15_CH1
10	SPI3_TX	42	TIM7_UP	74	TIM15_CH2
11	USART1_RX	43	TIM2_CH1	75	TIM15_UP
12	USART1_TX	44	TIM2_CH2	76	TIM15_TRIG

13	USART2_RX	45	TIM2_CH3	77	TIM15_COM
14	USART2_TX	46	TIM2_CH4	78	TIM16_CH1
15	USART3_RX	47	TIM2_UP	79	TIM16_UP
16	USART3_TX	48	TIM2_TRIG	80	TIM17_CH1
17	LPUART1_RX	49	TIM3_CH1	81	TIM17_UP
18	LPUART1_TX	50	TIM3_CH2	82	AES_IN
19	I2C1_RX	51	TIM3_CH3	83	AES_OUT
20	I2C1_TX	52	TIM3_CH4	84	Cordic_Read
21	I2C2_RX	53	TIM3_UP	85	Cordic_Write
22	I2C2_TX	54	TIM3_TRIG	86	ESMC_RX
23	I2C3_RX	55	TIM4_CH1	87	ESMC_TX
24	I2C3_TX	56	TIM4_CH2	88	SDIO
25	I2C4_RX	57	TIM4_CH3	89	UART1_RX
26	I2C4_TX	58	TIM4_CH4	90	UART1_TX
27	TIM1_CH1	59	TIM4_UP	91	UART2_RX
28	TIM1_CH2	60	TIM4_TRIG	92	UART2_TX
29	TIM1_CH3	61	TIM5_CH1	93	UART3_RX
30	TIM1_CH4	62	TIM5_CH2	94	UART3_TX
31	TIM1_UP	63	TIM5_CH3	95	LCDC_TX

12.3 寄存器描述

12.3.1 DMA 模块使能寄存器（DMA_EN)

Address offset:0x00

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	DMA_EN	RW	0	DMA 使能 0：禁止 1：使能

12.3.2 DMA 通道使能寄存器（DMA_CH_EN)

Address offset:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-

13:8	CH_EN_WE	W	6’h0	通道使能的写使能（每一 bit 对应一个通道，例如 CH_EN_WE[0]对应 CH0） 0：禁止 1：使能
7:6	保留	-	-	-
5:0	CH_EN	RW	6’h0	通道使能（每一 bit 对应一个通道，例如 CH_EN[0]对应 CH0） 0：禁止 1：使能 CH_EN 与 CH_EN_WE 一一对应，CH_EN 与 CH_EN_WE 必须同时写入，且仅当 CH_EN_WE 等于 1 时 CH_EN 才能写入; DMA_EN 为 0 时该寄存器硬件清零; 该寄存器在最后一次往目标地址传输数据完成后硬件清零;

12.3.3 DMA 控制寄存器（DMA_CH_CTRLLx）

Address offset:0x10+x*0x58（x 为通道编号）

Reset value:0x0030 4801

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.										TT		Res.			SRC_MSIZE
										RW	RW				RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_MSIZE		DST_MSIZE			SINC		DINC		SRC_TR_WIDTH			DST_TR_WIDTH			INT_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21:20	TT	RW	2’h3	传输类型 00：mem 到 mem 传输 01：mem 到外设传输 10：外设到 mem 传输 11：外设到外设传输
19:17	保留	-	-	-
16:14	SRC_MSIZE	RW	3’h1	源突发事务长度 3’h0：1 3’h1：4 3’h2：8 3’h3：16 3’h4：32 3’h5：64 3’h6：128 other：禁止 数据个数，每个数据位宽由 SRC_TR_WIDTH 决定
13:11	DST_MSIZE	RW	3’h1	目标突发事务长度 3’h0：1

				3’h1：4 3’h2：8 3’h3：16 3’h4：32 3’h5：64 3’h6：128 other：禁止 数据个数，每个数据位宽由 DST_TR_WIDTH 决定
10:9	SINC	RW	2’h0	源地址增量 00：递增 01：递减 10：不变 11：不变
8:7	DINC	RW	2’h0	目标地址增量 2’h0：递增 2’h1：递减 2’h2：不变 2’h3：不变
6:4	SRC_TR_WIDTH	RW	3’h0	源传输位宽 3’h0：8 bits 3’h1：16 bits 3’h2：32 bits other：禁止
3:1	DST_TR_WIDTH	RW	3’h0	目标传输位宽 3’h0：8 bits 3’h1：16 bits 3’h2：32 bits other：禁止
0	INT_EN	RW	1	中断使能 0：禁止 1：使能

12.3.4 DMA 控制寄存器（DMA_CH_CTRLHx）

Address offset：0x14+x*0x58（x 为通道编号）

Reset value:0x0000 0002

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

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	BLOCK_TS	RW	12’h2	块传输大小

				传输开始后读取该寄存器为已经从源地址读取的数据项的总数

12.3.5 DMA 配置寄存器（DMA_CH_CFGLx）

Address offset:0x18+x*0x58（x 为通道编号）

Reset value:0x0000 0e(x<<1)0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RELOAD_DST	RELOAD_SRC	MAX_ABRST										Res.			
RW	RW	RW													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				HS_SEL_SRC	HS_SEL_DST	FIFO_EMPTY	CH_SUSP	CH_PRIOR			INT_SUSP	Res.			
				RW	RW	R	RW	RW			RW				

Bit	Name	R/W	Reset Value	Function
31	RELOAD_DST	RW	0	目标自动重载 0：禁止 1：使能 DA 寄存器会在多块传输的每个块传输结束时自动重载为初始值
30	RELOAD_SRC	RW	0	源自动重载 0：禁止 1：使能 SA 寄存器会在多块传输的每个块传输结束时自动重载为初始值
29:20	MAX_ABRST	RW	10’h0	最大突发长度 0：不限制最大突发长度 other：对应设置的突发长度
19:12	保留	-	-	-
11	HS_SEL_SRC	RW	1	源软件或硬件握手选择 0：硬件握手 1：软件握手 当源是存储器时，该寄存器不起作用
10	HS_SEL_DST	RW	1	目标软件或硬件握手选择 0：硬件握手 1：软件握手 当目标是存储器时，该寄存器不起作用
9	FIFO_EMPTY	R	1	通道 fifo 状态 0：非空 1：空
8	CH_SUSP	RW	0	通道挂起 0：来自源的 DMA 传输不会被挂起 1：来自源的 DMA 传输被挂起
7:5	CH_PRIOR	RW	3’hx	通道优先级 默认值为通道号 该寄存器值越大优先级越高 默认每个通道的优先级为通道号，即

				通道 0 优先级为 0 通道 1 优先级为 1 通道 2 优先级为 2 通道 3 优先级为 3 通道 4 优先级为 4 通道 5 优先级为 5
4	INT_SUSP	RW	1	块中断中挂起（多块传输） 0：在块中断中继续传输 1：在块中断中暂停传输
3:0	保留	-	-	-

12.3.6 DMA 配置寄存器（DMA_CH_CFGHx）

Address offset:0x1c+x*0x58（x 为通道编号）

Reset value:0x0000 0004

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:11	DST_PER	RW	3’h0	目标硬件接口选择 分配请求到对应的通道，即 3’h0：将目标请求 0 分配给通道 x 3’h1：将目标请求 1 分配给通道 x 3’h2：将目标请求 2 分配给通道 x 3’h3：将目标请求 3 分配给通道 x 3’h4：将目标请求 4 分配给通道 x 3’h5：将目标请求 5 分配给通道 x other：禁止
10	保留	-	-	-
9:7	SRC_PER	RW	3’h0	源硬件接口选择 分配请求到对应的通道，即 3’h0：将源请求 0 分配给通道 x 3’h1：将源请求 1 分配给通道 x 3’h2：将源请求 2 分配给通道 x 3’h3：将源请求 3 分配给通道 x 3’h4：将源请求 4 分配给通道 x 3’h5：将源请求 5 分配给通道 x other：禁止
6:2	保留	-	-	-
1	FIFO_MODE	RW	0	fifo 模式选择 0：可用空间为指定传输宽度的单 AHB 传输 1：对于目标传输，可用数据大于或等于 FIFO 深度的一半；对于源传输，可用空间大于 FIFO

				深度的一半。在突发事务请求结束或块传输结束时会有例外。
0	保留	-	-	-

12.3.7 DMA 通道 x 源地址寄存器（DMA_SAx）

Address offset:0x20+x*0x58（x 为通道编号）

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[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**
SA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	SA[31:0]	RW	32’h0	DMA 传输的当前源地址

12.3.8 DMA 通道 x 目标地址寄存器（DMA_DAx）

Address offset:0x28+x*0x58（x 为通道编号）

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[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**
DA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	DA[31:0]	RW	32’h0	DMA 传输的当前目标地址

12.3.9 DMA 半块传输中断寄存器（DMA_STATUSHALFBLOCK）

Address offset:0x2a8

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6’h0	半块传输完成中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0） （MASKHALFBLOCK=1 有效） 0：未激活的状态 1：激活的状态

12.3.10 DMA 半块传输中断屏蔽寄存器（DMA_MASKHALFBLOCK）

Address offset:0x2b0

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6’h0	半块传输完成中断屏蔽的写使能 （每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6’h0	半块传输完成中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应，只有当对应的 INT_MASK_WE 等于 1 时该寄存器才能被写入

12.3.11 DMA 半块传输完成中断清除寄存器（DMA_CLEARHALFBLOCK）

Address offset:0x2b8

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除半块传输完成中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0） 0：无作用 1：清除中断

12.3.12 DMA 传输完成中断寄存器（DMA_STATUSTFR）

Address offset:0x2e8

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6'h0	传输完成中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0）（MASKTFR=1 有效） 0：未激活的状态 1：激活的状态

12.3.13 DMA 块传输完成中断寄存器（DMA_STATUSBLOCK）

Address offset:0x2f0

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6'h0	块传输完成中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0）（MASKBLOCK=1 有效） 0：未激活的状态 1：激活的状态

12.3.14 DMA 源事务完成中断寄存器（DMA_STATUSSRCTRAN）

Address offset:0x2f8

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6'h0	源事务完成中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0） （MASKSRCTRAN=1 有效） 0：未激活的状态 1：激活的状态

12.3.15 DMA 目标事务完成中断寄存器（DMA_STATUSDSTTRAN）

Address offset:0x300

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6’h0	目标事务完成中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0） （MASKDSTTRAN=1 有效） 0：未激活的状态 1：激活的状态

12.3.16 DMA 传输错误中断寄存器（DMA_STATUSERR）

Address offset:0x308

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	STATUS	R	6’h0	传输错误中断的状态（每一 bit 对应一个通道，例如 STATUS [0]对应 CH0）（MASKERR=1 有效） 0：未激活的状态 1：激活的状态

12.3.17 DMA 传输完成中断屏蔽寄存器（DMA_MASKTFR）

Address offset:0x310

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6’h0	传输完成中断屏蔽的写使能 （每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6’h0	传输完成中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应， INT_MASK 与 INT_MASK_WE 必须同时写入， 且仅当 INT_MASK_WE 等于 1 时 INT_MASK 才能写入

12.3.18 DMA 块传输完成中断屏蔽寄存器（DMA_MASKBLOCK）

Address offset:0x318

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6’h0	块传输完成中断屏蔽的写使能（每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6’h0	块传输完成中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应， INT_MASK 与 INT_MASK_WE 必须同时写入， 且仅当 INT_MASK_WE 等于 1 时 INT_MASK 才能写入

12.3.19 DMA 源事务完成中断屏蔽寄存器（DMA_MASKSRCTRAN）

Address offset:0x320

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6'h0	源事务完成中断屏蔽的写使能（每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6'h0	源事务完成中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应， INT_MASK 与 INT_MASK_WE 必须同时写入，且仅当 INT_MASK_WE 等于 1 时 INT_MASK 才能写入

12.3.20 DMA 目标事务完成中断屏蔽寄存器（DMA_MASKDSTTRAN）

Address offset:0x328

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6'h0	目标事务完成中断屏蔽的写使能（每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6'h0	目标事务完成中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应， INT_MASK 与 INT_MASK_WE 必须同时写入，

				且仅当 INT_MASK_WE 等于 1 时 INT_MASK 才能写入

12.3.21 DMA 传输错误中断屏蔽寄存器（DMA_MASKERR）

Address offset:0x330

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	INT_MASK_WE	W	6’h0	传输错误中断屏蔽的写使能（每一 bit 对应一个通道，例如 INT_MASK_WE [0]对应 CH0） 0：中断屏蔽写入禁止 1：中断屏蔽写入使能
7:6	保留	-	-	-
5:0	INT_MASK	RW	6’h0	传输错误中断屏蔽（每一 bit 对应一个通道，例如 INT_MASK [0]对应 CH0） 0：屏蔽中断 1：不屏蔽中断 INT_MASK 与 INT_MASK_WE 一一对应， INT_MASK 与 INT_MASK_WE 必须同时写入， 且仅当 INT_MASK_WE 等于 1 时 INT_MASK 才能写入

12.3.22 DMA 传输完成中断清除寄存器（DMA_CLEARTFR）

Address offset:0x338

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除传输完成中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0） 0：无作用 1：清除中断

12.3.23DMA 块传输完成中断清除寄存器（DMA_CLEARBLOCK）

Address offset:0x340

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除块传输完成中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0） 0：无作用 1：清除中断

12.3.24DMA 源事务完成中断清除寄存器（DMA_CLEARSRCTRAN）

Address offset:0x348

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除源事务完成中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0） 0：无作用 1：清除中断

12.3.25DMA 目标事务中断清除寄存器（DMA_CLEARDSTTRAN）

Address offset:0x350

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除目标事务完成中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0）

				0：无作用 1：清除中断

12.3.26 DMA 传输错误中断清除寄存器（DMA_CLEARERR）

Address offset:0x358

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:0	CLEAR	W	6’h0	清除传输错误中断（每一 bit 对应一个通道，例如 CLEAR [0]对应 CH0） 0：无作用 1：清除中断

12.3.27 DMA 通道中断状态寄存器（DMA_STATUSINT）

Address offset:0x360

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.										HALFBLOCK	ERR	DSTT	SRCT	BLOCK	TFR
										R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5	HALFBLOCK	R	0	所有通道中是否存在半块传输完成中断 0：所有通道中不存在半块传输完成中断 1：所有通道中存在半块传输完成中断
4	ERR	R	0	所有通道中是否存在传输错误中断 0：所有通道中不存在传输错误中断 1：所有通道中存在传输错误中断
3	DSTT	R	0	所有通道中是否存在目标传输完成中断 0：所有通道中不存在目标传输完成中断 1：所有通道中存在目标传输完成中断
2	SRCT	R	0	所有通道中是否存在源传输完成中断 0：所有通道中不存在源传输完成中断 1：所有通道中存在源传输完成中断
1	BLOCK	R	0	所有通道中是否存在块传输完成中断 0：所有通道中不存在块传输完成中断 1：所有通道中存在块传输完成中断
0	TFR	R	0	所有通道中是否存在传输完成中断

				0：所有通道中不存在传输完成中断 1：所有通道中存在传输完成中断

12.3.28 DMA 源事务软件握手寄存器（DMA_REQSRC）

Address offset:0x368

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	SRC_REQ_WE	W	6’h0	源事务软件请求的写使能（每一 bit 对应一个通道，例如 SRC_REQ_WE [0]对应 CH0） 0：源请求写入禁止 1：源请求写入使能
7:6	保留	-	-	-
5:0	SRC_REQ	RW	6’h0	源事务软件请求（每一 bit 对应一个通道，例如 SRC_REQ[0]对应 CH0） 0：源请求未激活 1：源请求激活 CH_EN 为 0 时该寄存器硬件清零； SRC_REQ 与 SRC_REQ_WE 一一对应，只有当对应的 SRC_REQ_WE 等于 1 时该寄存器才能被写入

12.3.29 DMA 目标事务软件握手寄存器（DMA_REQDST）

Address offset:0x370

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	DST_REQ_WE	W	6’h0	目标事务软件请求的写使能（每一 bit 对应一个通道，例如 DST_REQ_WE [0]对应 CH0） 0：目标请求写入禁止 1：目标请求写入使能
7:6	保留	-	-	-
5:0	DST_REQ	RW	6’h0	目标事务软件请求（每一 bit 对应一个通道，例如 DST_REQ[0]对应 CH0）

				0：目标请求未激活 1：目标请求激活 CH_EN 为 0 时该寄存器硬件清零； DST_REQ 与 DST_REQ_WE 一一对应，只有当对应的 DST_REQ_WE 等于 1 时该寄存器才能被写入

12.3.30 DMA 源单事务软件握手寄存器（DMA_SGLRQSRC）

Address offset:0x378

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:8	SRC_SGLREQ_WE	W	6’h0	源单事务软件请求的写使能（每一 bit 对应一个通道，例如 SRC_SGLREQ_WE [0]对应 CH0） 0：源请求写入禁止 1：源请求写入使能
7:6	保留	-	-	-
5:0	SRC_SGLREQ	RW	6’h0	源单事务软件请求（每一 bit 对应一个通道，例如 SRC_SGLREQ [0]对应 CH0） 0：源请求未激活 1：源请求激活 CH_EN 为 0 时该寄存器硬件清零； SRC_SGLREQ 与 SRC_SGLREQ_WE 一一对应，只有当对应的 SRC_SGLREQ_WE 等于 1 时该寄存器才能被写入

12.3.31 DMA 目标单事务软件握手寄存器（DMA_SGLRQDST）

Address offset:0x380

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-

13:8	DST_SGLREQ_WE	W	6’h0	目标单事务软件请求的写使能（每一 bit 对应一个通道，例如 DST_SGLREQ_WE [0]对应 CH0） 0：目标请求写入禁止 1：目标请求写入使能
7:6	保留	-	-	-
5:0	DST_SGLREQ	RW	6’h0	目标单事务软件请求（每一 bit 对应一个通道，例如 DST_SGLREQ[0]对应 CH0） 0：目标请求未激活 1：目标请求激活 CH_EN 为 0 时该寄存器硬件清零； DST_SGLREQ 与 DST_SGLREQ_WE 一一对应，只有当对应的 DST_SGLREQ_WE 等于 1 时该寄存器才能被写入

13. 中断和事件

13.1 嵌套向量中断控制器(NVIC)

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内核异常等中断。更多关于异常和 NVIC 编程的说明请参考 Cortex-M4F 编程手册。

13.1.1 主要特性

- 86 个可屏蔽中断通道(不包含 16 个 Cortex™-M4F 的中断线);
- 8 个可编程的优先等级(使用了 3 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;

13.1.2 系统滴答（SysTick）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。

13.1.3 中断和异常向量表

向量编号	优先级	优先级类型	名称	地址	说明
-	-	-	-	0x0000_0000	保留
-	-3	固定	Reset	0x0000_0004	复位
-	-2	NMI	RCC HSE 时钟安全（HSE CSS）/FLASH 的 ECC 错误/SRAM 的 CCM 错误	0x0000_0008	不可屏蔽中断
-	-1	固定	硬件失效(HardFault)	0x0000_000C	所有类型的失效
-	0	可编程	存储管理(MemManage)	0x0000_0010	存储器管理
-	1	可编程	总线错误(BusFault)	0x0000_0014	预取指失败，存储器访问失败
-	2	可编程	错误应用(UsageFault)	0x0000_0018	未定义的指令或非法状态
-	-	-	-	0x0000_001C	保留
-	-	-	-	0x0000_0020	保留
-	-	-	-	0x0000_0024	保留
-	-	-	-	0x0000_0028	保留
-	3	可编程	SVCall	0x0000_002C	通过 SWI 指令的系统服务调用
-	4	可编程	调试监控(DebugMonitor)	0x0000_0030	调试监控器
-	-	-	-	0x0000_0034	保留
-	5	可编程	PendSV	0x0000_0038	可挂起的系统服务
-	6	可编程	SysTick	0x0000_003C	系统嘀嗒定时器

0	7	可编程	WWDG	0x0000_0040	窗口看门狗定时器中断
1	8	可编程	PVD	0x0000_0044	连到 EXTI 的电源电压检测(PVD)中断
2	9	可编程	TAMPER	0x0000_0048	侵入检测中断
3	10	可编程	RTC	0x0000_004C	实时时钟(RTC)全局中断
4	11	可编程	FMC	0x0000_0050	闪存 FMC 全局中断
5	12	可编程	RCC 和 CTC	0x0000_0054	复位和时钟控制(RCC)和 CTC 中断
6	13	可编程	EXTI0	0x0000_0058	EXTI 线 0 中断
7	14	可编程	EXTI1	0x0000_005C	EXTI 线 1 中断
8	15	可编程	EXTI2	0x0000_0060	EXTI 线 2 中断
9	16	可编程	EXTI3	0x0000_0064	EXTI 线 3 中断
10	17	可编程	EXTI4	0x0000_0068	EXTI 线 4 中断
11	18	可编程	DMA1 通道 1	0x0000_006C	DMA1 通道 1 全局中断
12	19	可编程	DMA1 通道 2	0x0000_0070	DMA1 通道 2 全局中断
13	20	可编程	DMA1 通道 3	0x0000_0074	DMA1 通道 3 全局中断
14	21	可编程	DMA1 通道 4	0x0000_0078	DMA1 通道 4 全局中断
15	22	可编程	DMA1 通道 5	0x0000_007C	DMA1 通道 5 全局中断
16	23	可编程	DMA1 通道 6	0x0000_0080	DMA1 通道 6 全局中断
17	24	可编程	DMA2 通道 6	0x0000_0084	DMA2 通道 6 全局中断
18	25	可编程	ADC1_2	0x0000_0088	ADC1 和 ADC2 全局中断
19	26	可编程	OTG1_FS-WKUP	0x0000_008C	USB1 OTG FS 唤醒通过 EXTI 中断
20	27	可编程	CAN1 int0	0x0000_0090	CAN1 中断 0
21	28	可编程	OTG1_FS	0x0000_0094	USB1 OTG 全局中断
22	29	可编程	TIM18	0x0000_0098	TIM18 全局中断
23	30	可编程	EXTI9_5	0x0000_009C	EXTI 线[9:5]中断
24	31	可编程	TIM1_BRK_TIM9	0x0000_00A0	TIMER1 中止中断和 TIMER9 全局中断
25	32	可编程	TIM1_UP_TIM10	0x0000_00A4	TIMER1 更新中断和 TIMER10 全局中断
26	33	可编程	TIM1_TRG_COM_TIM11	0x0000_00A8	TIMER1 触发和通信中断及 TIMER11 全局中断
27	34	可编程	TIM1_CC	0x0000_00AC	TIMER1 捕获比较中断
28	35	可编程	TIM2	0x0000_00B0	TIMER2 全局中断
29	36	可编程	TIM3	0x0000_00B4	TIMER3 全局中断
30	37	可编程	TIM4	0x0000_00B8	TIMER4 全局中断
31	38	可编程	I2C1_EV	0x0000_00BC	I2C1 事件中断
32	39	可编程	I2C1_ER	0x0000_00C0	I2C1 错误中断
33	40	可编程	I2C2_EV	0x0000_00C4	I2C2 事件中断
34	41	可编程	I2C2_ER	0x0000_00C8	I2C2 错误中断
35	42	可编程	SPI1	0x0000_00CC	SPI1 全局中断
36	43	可编程	SPI2	0x0000_00D0	SPI2 全局中断
37	44	可编程	USART1	0x0000_00D4	USART1 全局中断
38	45	可编程	USART2	0x0000_00D8	USART2 全局中断
39	46	可编程	USART3	0x0000_00DC	USART3 全局中断
40	47	可编程	EXTI15_10	0x0000_00E0	EXTI 线[15:10]中断
41	48	可编程	RTCAlarm	0x0000_00E4	连到 EXTI 的 RTC 闹钟中断
42	49	可编程	TIM19	0x0000_00E8	TIM19 全局中断

43	50	可编程	TIM8_BRK_TIM12	0x0000_00EC	TIMER8 中止中断和 TIMER12 全局中断
44	51	可编程	TIM8_UP_TIM13	0x0000_00F0	TIMER8 更新中断和 TIMER13 全局中断
45	52	可编程	TIM8_TRG_COM_TIM14	0x0000_00F4	TIMER8 触发和通信中断 TIMER14 全局中断
46	53	可编程	TIM8_CC	0x0000_00F8	TIMER8 捕获比较中断
47	54	可编程	ADC3	0x0000_00FC	ADC3 全局中断
48	55	可编程	ESMC	0x0000_0100	ESMC 全局中断
49	56	可编程	SDIO	0x0000_0104	SDIO 全局中断
50	57	可编程	TIM5	0x0000_0108	TIM5 全局中断
51	58	可编程	SPI3	0x0000_010C	SPI3 全局中断
52	59	可编程	UART1	0x0000_0110	UART 全局中断
53	60	可编程	LPUART	0x0000_0114	LPUART 全局和唤醒中断
54	61	可编程	TIM6	0x0000_0118	TIM6 全局中断
55	62	可编程	TIM7	0x0000_011C	TIM7 全局中断
56	63	可编程	DMA2 通道 1	0x0000_0120	DMA2 通道 1 全局中断
57	64	可编程	DMA2 通道 2	0x0000_0124	DMA2 通道 2 全局中断
58	65	可编程	DMA2 通道 3	0x0000_0128	DMA2 通道 3 全局中断
59	66	可编程	DMA2 通道 4_5	0x0000_012C	DMA2 通道 4 和通道 5 全局中断
60	67	可编程	TIM15	0x0000_0130	TIMER15 全局中断
61	68	可编程	TIM16	0x0000_0134	TIMER16 全局中断
62	69	可编程	TIM17	0x0000_0138	TIMER17 全局中断
63	70	可编程	I2C3_EV	0x0000_013C	I2C3 事件中断
64	71	可编程	I2C3_ER	0x0000_0140	I2C3 错误中断
65	72	可编程	I2C4_EV	0x0000_0144	I2C4 事件中断
66	73	可编程	I2C4_ER	0x0000_0148	I2C4 错误中断
67	74	可编程	ETH	0x0000_014C	以太网全局中断
68	75	可编程	ETH_WKUP	0x0000_0150	以太网唤醒中断
69	76	可编程	RNG	0x0000_0154	RNG 全局中断
70	77	可编程	AES	0x0000_0158	AES 全局中断
71	78	可编程	CORDIC	0x0000_015C	CORDIC 全局中断
72	79	可编程	DAC	0x0000_0160	DAC 全局中断
73	80	可编程	COMP1	0x0000_0164	COMP1 全局中断
74	81	可编程	COMP2	0x0000_0168	COMP2 全局中断
75	82	可编程	COMP3	0x0000_016C	COMP3 全局中断
76	83	可编程	COMP4	0x0000_0170	COMP4 全局中断
77	84	可编程	UART2	0x0000_0174	UART2 全局中断
78	85	可编程	UART3	0x0000_0178	UART3 全局中断
79	86	可编程	保留	0x0000_017C	
80	87	可编程	LPTIM	0x0000_0180	LPTIM 全局中断
81	88	可编程	OTG2_FS-WKUP	0x0000_0084	USB2 OTG FS 唤醒通过 EXTI 中断
82	89	可编程	OTG2_FS	0x0000_0088	USB2 OTG 全局中断
83	90	可编程	CAN1 int1	0x0000_008C	CAN1 中断 1
84	91	可编程	CAN2 int0	0x0000_0090	CAN2 中断 0

85	92	可编程	CAN2 int1	0x0000_0094	CAN2 中断 1

13.2 外部扩展中断/事件控制器（EXTI）

扩展中断和事件控制器，通过可配置线和直接输入线，管理着 CPU 和系统唤醒功能，并输出下述请求信号：

- 事件请求，送给 CPU 的事件输入
- 唤醒请求，送给功耗管理控制模块

EXTI 唤醒请求允许系统从停止模式唤醒，事件请求也可以在运行模式使用。

EXTI 允许管理多达 31 个可配置线和直接输入线（22 个可配置线和 9 个直接输入线）。

有 31 个能产生事件/中断请求的边沿检测器。其中 22 个可配置线，每个输入线可以独立地配置输入触发方式(上升沿或下降沿或者双边沿触发)。31 个可配置线和直接输入线，每个输入线都可以独立地被屏蔽。挂起寄存器保持着可配置线的中断请求。

13.2.1 主要特性

EXTI 控制器的主要特性如下：

- 可通过 GPIO 和指定模块(如 PVD/RTC 等）唤醒低功耗模式
- 可配置线
  - 可选有效触发沿(上升沿/下降沿)
  - 中断挂起标志位
  - 独立中断和事件屏蔽位
  - 可软件触发
- 直接输入线
  - 固定的上升沿触发
  - 无中断挂起标志位
  - 无独立中断和事件屏蔽位
  - 无软件触发

13.2.2 唤醒事件管理

本产品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：

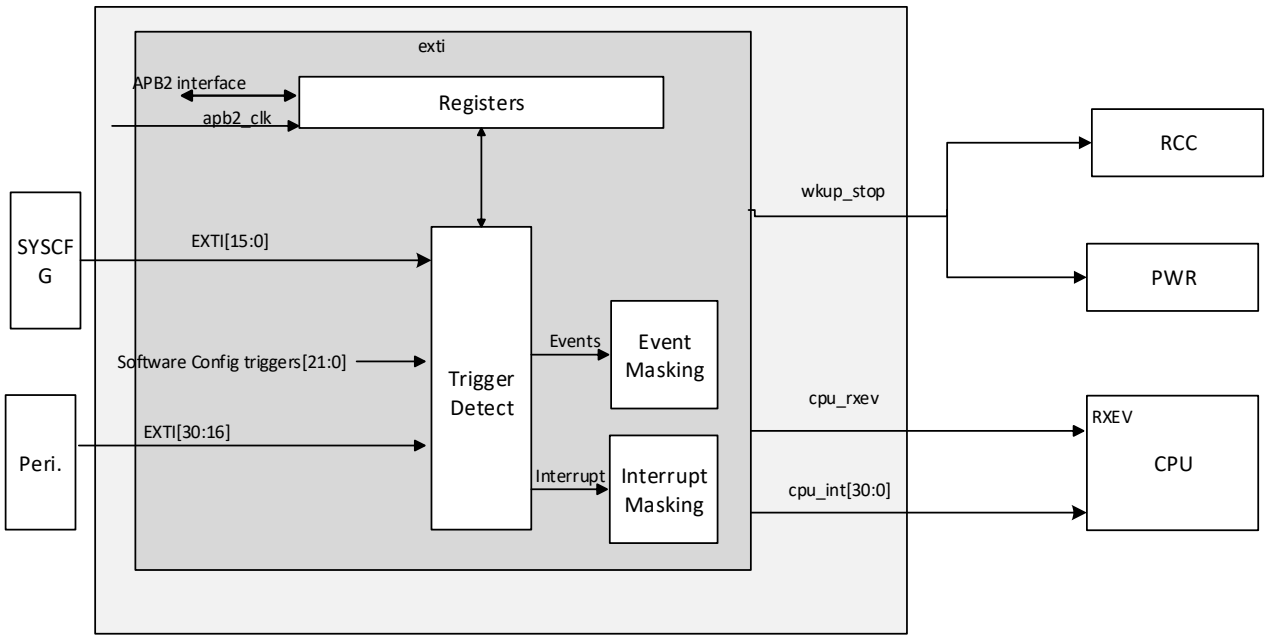
- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 Cortex-M4F 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)。
- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

13.2.3 功能说明

要产生中断，必须先配置好并使能中断线。22 个可配置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。

13.2.3.1 EXTI 框图



13.2.3.2 硬件中断选择

- 配置中断线的屏蔽位(EXTI_IMRx(x=1,2))
- 配置所选中断线的触发选择位(EXTI_RTSR 和 EXTI_FTSR);

13.2.3.3 硬件事件选择

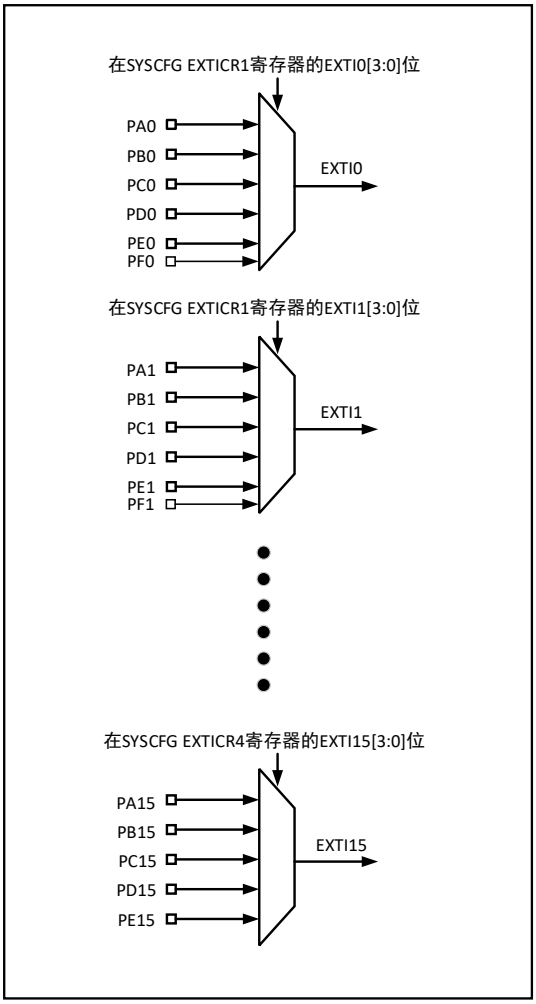
- 配置事件线的屏蔽位(EXTI_EMRx(x=1,2))
- 配置事件线的触发选择位(EXTI_RTSR 和 EXTI_FTSR)

13.2.3.4 软件中断/事件的选择

- 配置中断/事件线屏蔽位(EXTI_IMRx, EXTI_EMRx;(x=1,2))
- 设置软件中断寄存器的请求位(EXTI_SWIER)

13.2.4 外部中断/事件线映射

86 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：



注：通过 SYSCFG_EXTICRx 配置 GPIO 线上的外部中断/事件，必须先使能 SYSCFG 时钟。

表 13-1 EXTI 线连接

EXTI	Line 源	Line 类型
0-15	GPIO	可配置线
16	PVD	可配置线
17	RTC alarm 事件	可配置线
18	COMP1 输出	可配置线
19	COMP2 输出	可配置线
20	COMP3 输出	可配置线
21	COMP4 输出	可配置线
22	I2C1 ev 唤醒	直接输入线
23	I2C2 ev 唤醒	直接输入线
24	I2C3 ev 唤醒	直接输入线
25	I2C4 ev 唤醒	直接输入线
26	保留	-
27	保留	-
28	保留	-
29	保留	-
30	LPTIM1 唤醒	直接输入线
31	ETH 唤醒事件	直接输入线

32	LPUART 唤醒	直接输入线
33	USB1 OTG 唤醒事件	直接输入线
34	USB2 OTG 唤醒事件	直接输入线

13.3 寄存器描述

必须以 32 位的方式对这些寄存器进行访问。

13.3.1 中断屏蔽寄存器（0x00：EXTI_IMR1）

Address offset:0x00

Reset value:0xC3C0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMR31	IMR30	Res				IMR25	IMR24	IMR23	IMR22	IMR21	IMR20	IMR19	IMR18	IMR17	IMR16
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**
IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	IMRx(x=31,30)	RW	2’h3	**IMRx（x=31,30）**：线 x 上的中断屏蔽 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
29:26	保留	-	-	**-**
25:0	IMRx(x=0-25)	RW	26’h3C00000	**IMRx(x=0-25)**：线 x 上的中断屏蔽 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

13.3.2 事件屏蔽寄存器（0x04：EXTI_EMR1）

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMR31	EMR30	Res				EMR25	EMR24	EMR23	EMR22	EMR21	EMR20	EMR19	EMR18	EMR17	EMR16
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**
EMR15	EMR14	EMR13	EMR12	EMR11	EMR10	EMR9	EMR8	EMR7	EMR6	EMR5	EMR4	EMR3	EMR2	EMR1	EMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	EMRx(x=31,30)	RW	2’h0	EMRx(x=31,30)：线 x 上的事件屏蔽 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。
29:26	保留	-	-	**-**
25:0	EMRx(x=0-25)	RW	26’h0	EMRx(x=0-25)：线 x 上的事件屏蔽

				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。

13.3.3 上升沿触发选择寄存器（0x08：EXTI_RTSR）

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										RTSR 21	RTSR 20	RTSR 19	RTSR 18	RTSR 17	RTSR 16
										RW	RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTS R15	RTSR 14	RTSR 13	RTSR 12	RTSR 11	RTSR 10	RTS R9	RTS R8	RTS R7	RTS R6	RTSR 5	RTSR 4	RTSR 3	RTSR 2	RTSR 1	RTSR 0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21:0	RTSRx(x=0-21)	RW	22’h0	RTSRx(x=0-21)：线 x 上的上升沿触发事件配置位 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)

13.3.4 下降沿触发选择寄存器（0x0C：EXTI_FTSR）

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										FTSR 21	FTS R 20	FTS R 19	FTS R 18	FTS R 17	FTS R 16
										RW	RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FTSR 15	FTSR 14	FTFR 13	FTSR 12	FTSR 11	FTSR 10	FTS R9	FTS R8	FTS R7	FTS R6	FTSR 5	FTS R4	FTS R3	FTS R2	FTS R1	FTS R0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21:0	FTSRx	RW	22’h0	**RTSRx(x=0-21)**：线 x 上的下降沿触发事件配置位 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)

13.3.5 软件中断事件寄存器（0x10：EXTI_SWIER）

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										SWIE R21	SWIE R20	SWIE R19	SWIE R18	SWIE R17	SWIE R16
										RW	RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWIE R15	SWIE R14	SWIE R13	SWIE R12	SWIE R11	SWIE R10	SWI ER9	SWI ER8	SWI ER7	SWI ER6	SWIE R5	SWIE R4	SWIE R3	SWIE R2	SWIE R1	SWIE R0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21:0	SWIERx	RW	22'h0	**SWIERx(x=0-21)**：线 x 上的软件中断 当该位为'0'时，写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断，则此时将产生一个中断。 注：通过清除 EXTI_PR 的对应位(写入'1')，可以清除该位为'0'。

13.3.6 挂起寄存器（0x14：EXTI_PR）

Address offset:0x14

Reset value:0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										PR21	PR20	PR19	PR18	PR17	PR16
										RC_W1					
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
RC_W1															

Bit	Name	R/W	Reset Value	Function
31：22	保留	-	-	-
21：0	PRx	RC_W1	22'hX	**PRx(x=0-21)**：挂起位 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，也可以通过改变边沿检测的极性清除。

13.3.7 中断屏蔽寄存器（0x18：EXTI_IMR2）

Address offset:0x18

Reset value:0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res													IMR34	IMR33	IMR32
													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2:0	IMRx	RW	3'h7	**IMRx(x=34-32)**：线 x 上的中断屏蔽 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

13.3.8 事件屏蔽寄存器（0x1C：EXTI_EMR2）

Address offset:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													EMR34.	EMR33	EMR32
													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	**-**
2:0	EMRx	RW	3'h0	**EMRx(x=34-32)**：线 x 上的事件屏蔽 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。

14. 模拟/数字转换（ADC）

14.1 简介

12 位 ADC 是一种采用逐次逼近方式的模拟数字转换器。它有 20 个多路复用通道，可以转换来自多个外部通道和多个内部通道的模拟信号。模拟看门狗允许应用程序来检测输入电压是否超出用户设定的高低阈值。各个通道的 A/D 转换可以配置成单次、连续、扫描或间断转换模式。ADC 转换的结果可以按照左对齐或右对齐的方式存储在 16 位数据寄存器中。

14.2 主要特征

- 高性能:
  - 可配置的 12 位、10 位、8 位和 6 位分辨率
  - 最大 ADC 采样率：4 M_{SPS}
  - 自校准
  - 可编程采样时间
  - 数据寄存器可配置数据对齐方式
  - 支持规则通道数据转换的 DMA 请求
  - 双 ADC 模式
  - 可配置的单端或差分输入
- 过采样器
  - 16 bits 数据寄存器
  - 过采样率 2 - 256 可调
  - 可编程数据移位可达 8 bits
- 数据预处理
  - 增益补偿
  - Offset 补偿
- 动态低功耗特性:
  - 自动延迟转换模式：防止以低频 PCLK 运行产生溢出
- 模拟输入通道:
  - ADC1 连接 14 个外部通道，5 个内部通道和 1 个内部 V_{SSA} 通道
  - ADC2 连接 16 个外部通道，3 个内部通道和 1 个内部 V_{SSA} 通道
  - ADC3 连接 15 个外部通道，4 个内部通道和 1 个内部 V_{SSA} 通道
- 启动转换方式:
  - 软件启动
  - 硬件触发
- 转换模式:
  - 单次模式，每次触发转换一次选择的输入通道;
  - 扫描模式，可以扫描一系列通道;
  - 连续模式，连续转换所选择的输入通道;
  - 间断模式，每次触发连续转换子序列通道，多次触发直到转换所有通道

— 同步模式（适用于具有两个 ADC 的设备）。

- 模拟看门狗
- 中断的产生：
  - 采样结束
  - 转换结束(规则/注入通道)
  - 规则序列或注入序列转换结束
  - 模拟看门狗 1，2，3 事件
- ADC 输入范围(选择外部参考电压)：$V_{REF-} \leq V_{IN} \leq V_{REF+}$

14.3 功能说明

14.3.1 ADC 模块框图

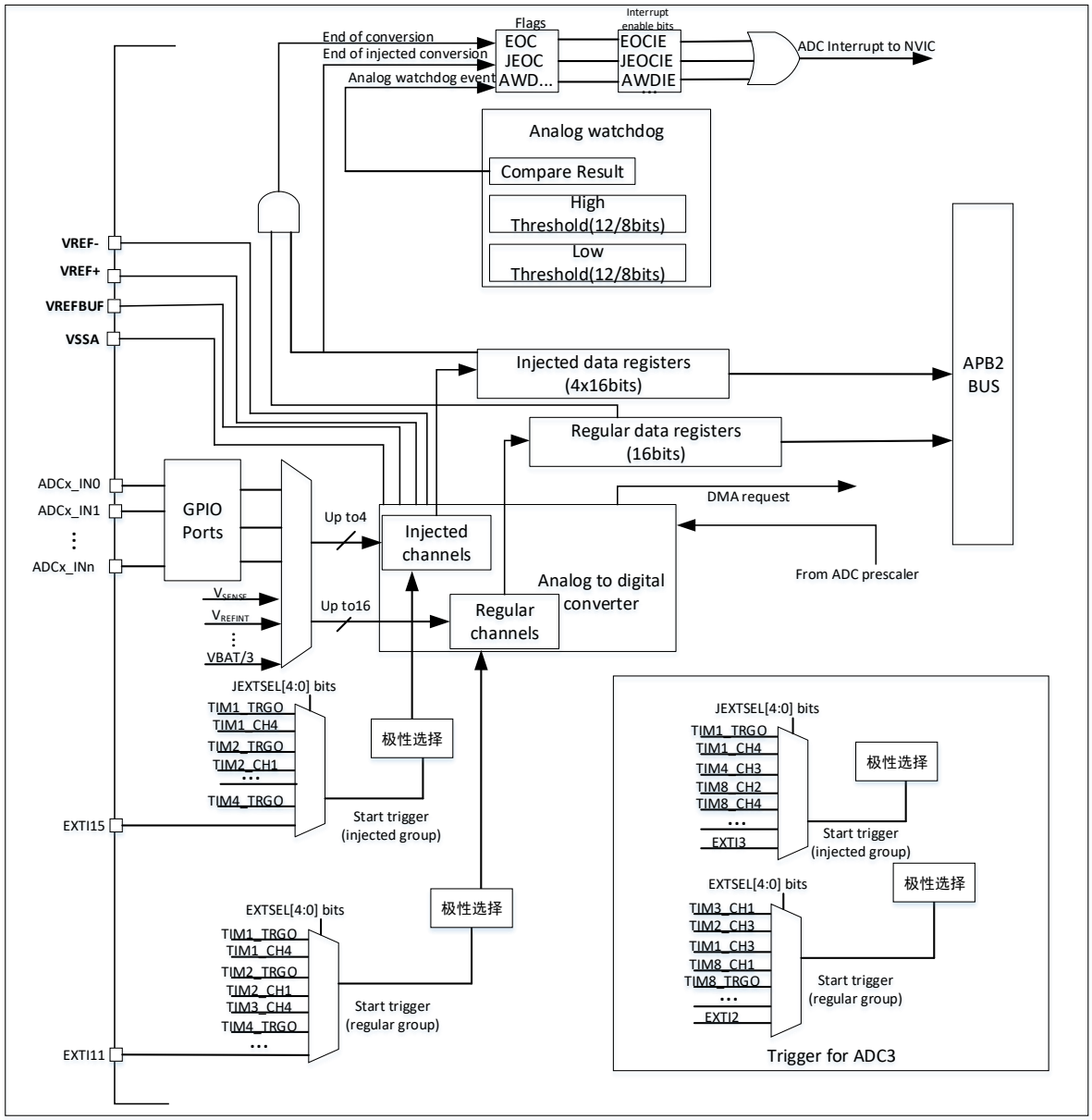


图 14-1 ADC 模块框图

14.3.2 ADC1/2/3 连接关系

ADC1、ADC2 和 ADC3 的外部通道共享一些 GPIO，（通道配置通过 ADC_SQRx 或 ADC_JSQR 设置），如下图所示。

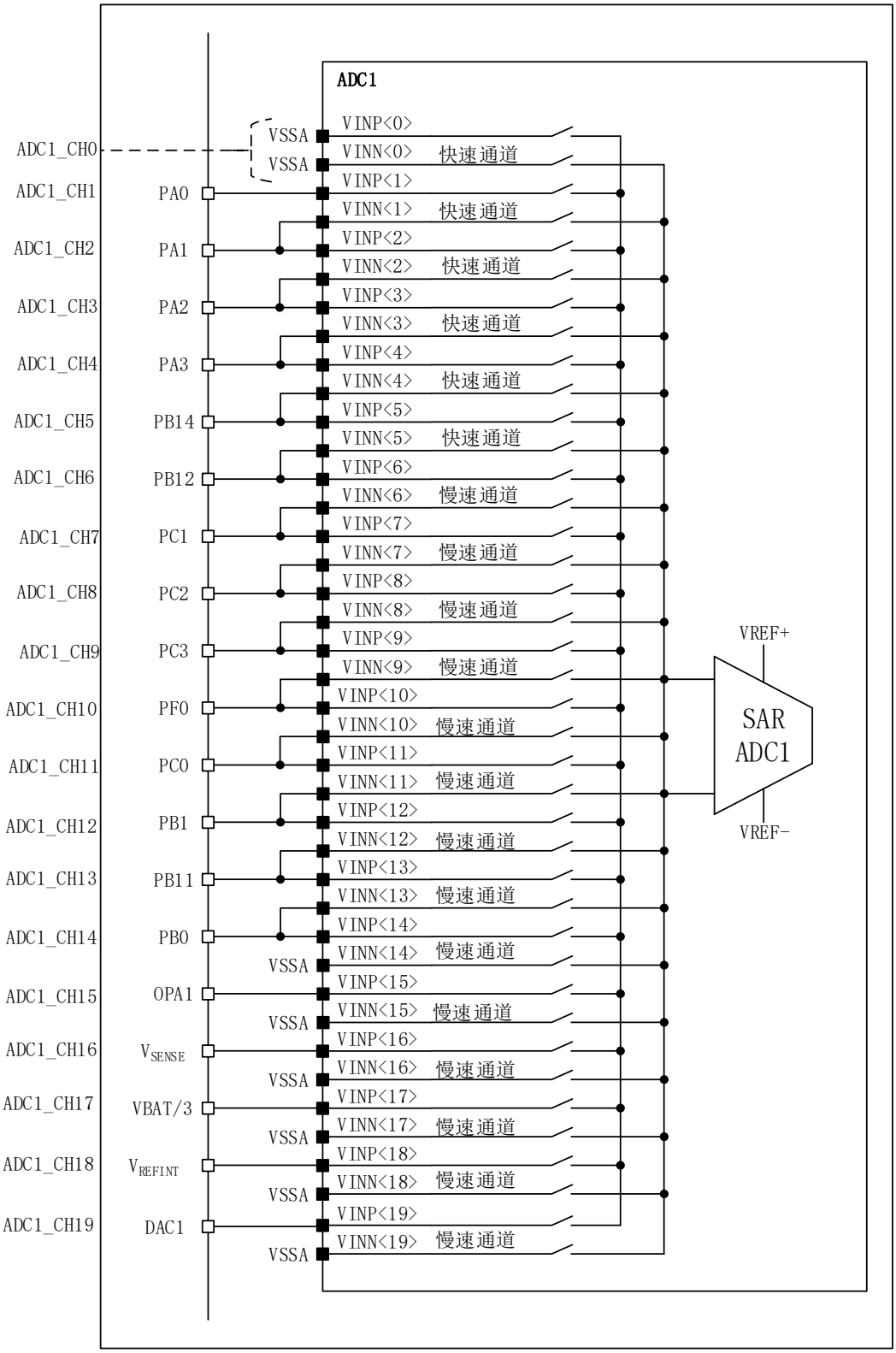


图 14-2 ADC1 通道连接关系

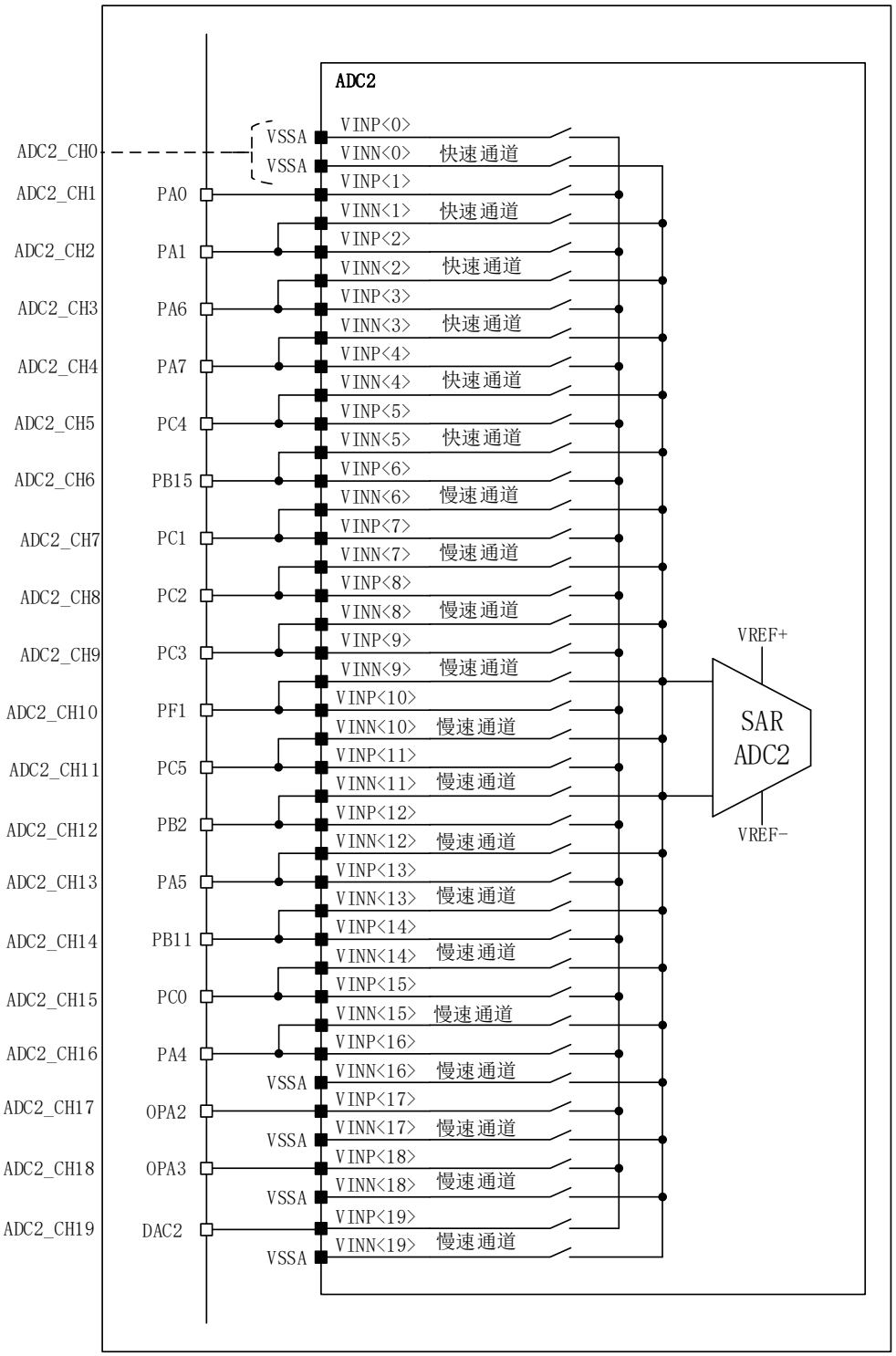


图 14-3 ADC2 通道连接关系

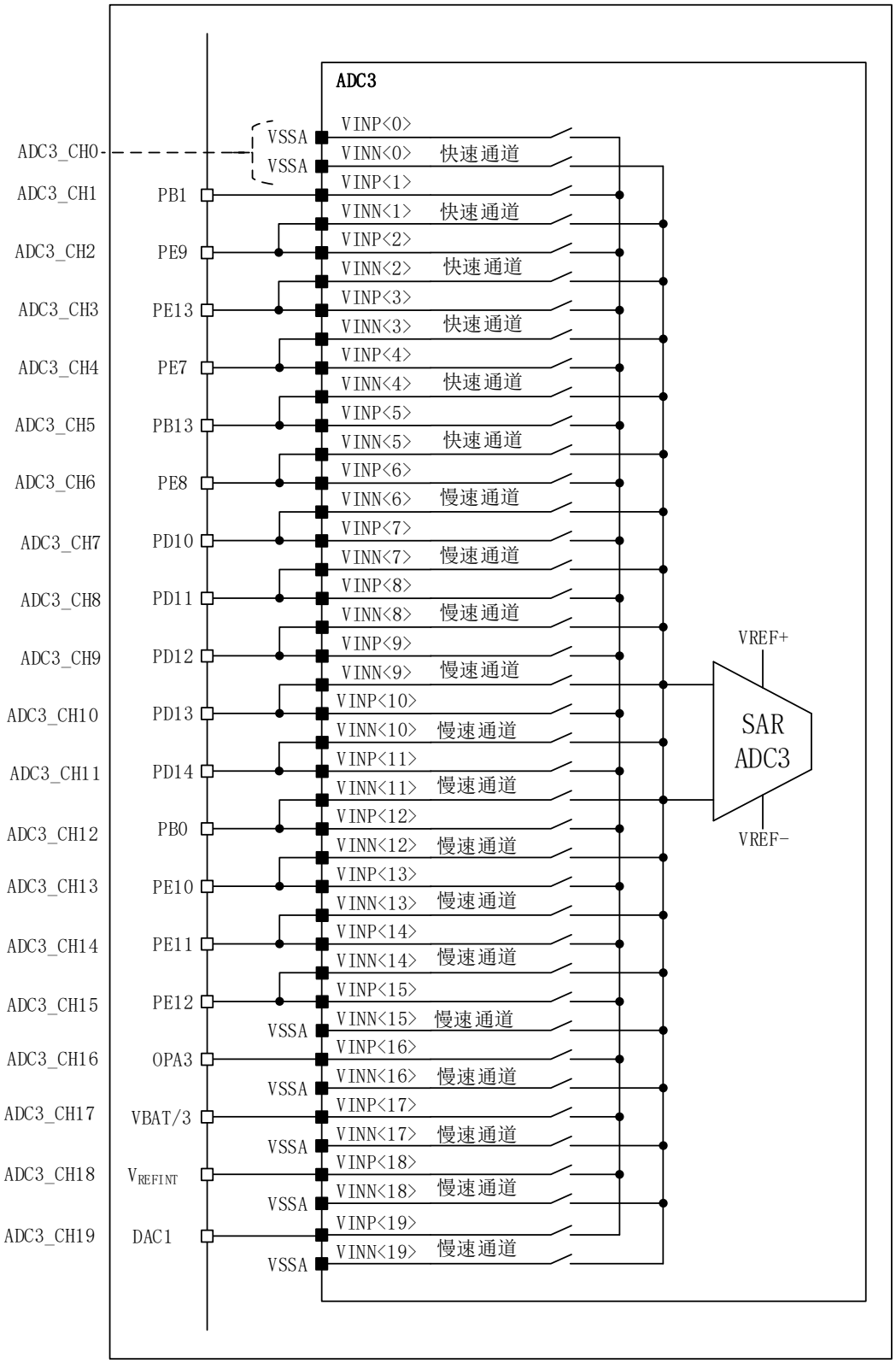


图 14-4 ADC3 通道连接关系

14.3.3 ADC1/2/3 使用限制

1. 单 ADC 和双 ADC 模式：间断模式(DISCEN)，不支持注入打断规则。
2. 单 ADC 和双 ADC 模式：自动延迟转换模式(AUTDLY)，不支持注入打断规则。

3. 单 ADC 和双 ADC 模式：自动注入(JAUTO)不支持自动延迟转换模式(AUTDLY)模式
4. 单 ADC 和双 ADC 模式：注入期间内的规则触发被忽略。
5. 单 ADC 和双 ADC 模式：单次转换模式时，软件必须将规则序列长度设置为 0。
6. 双 ADC 模式(ADC3 无双 ADC 模式)：
   a) Slave ADC 的 JEXTEN/JEXTSEL/EXTEN/EXTSEL/JADSTART/ADSTART 必须保持复位值
   b) DUAL=7,8,9 时不支持注入打断规则
   c) DUAL=8 时,采样时间只能选择 2.5,3.5 和 6.5 ADC 时钟周期,其他采样时间不能配置。

14.3.4 单端和差分输入通道

ADC 通道可配置为单端输入或差分输入，通过 ADC_DIFSEL 寄存器中 DIFSEL 配置。当 DIFSEL[x](x=0-19)为 1 时，选择转换的通道为差分输入模式，否则为单端输入模式。

必须当 ADC 被禁用时（ADEN=0）配置。请注意 DIFSEL 对应的单端通道始终保持为 0。

在单端模式下，通道 i(i=0-19)的模拟电压为 ADCy_INx（VINP[i]）和 V_{REF-}之差。

在差分模式下，通道 i(i=0-19)的模拟电压为 ADCy_INx（VINP[i]）和 ADCy_INx+1（VINN[i]）的电压差。

差分模式下，输入电压范围为 V_{REF-} 到 V_{REF+}，达到 2xV_{REF+}的全刻度。当 VINP[i]等于 V_{REF-},VINN[i]等于 V_{REF+} 则最大负端输入差分电压(V_{REF-})对应 0x000 输出。当 VINP[i]等于 V_{REF+},VINN[i]等于 V_{REF-} 则最大正端输入差分电压(V_{REF+})对应 0xFFF 输出。当 VINP[i]和 VINN[i]连在一起，零输入差分电压对应 0x800 输出。

差分模式下的 ADC 灵敏度比单端模式下小两倍。内部通道(如 V_{SENSE} 和 V_{REFINT})只采用单端模式。

完整的每个 ADC 输入通道连接描述，见相关章节描述。

警告：

在差分输入模式下配置通道“i”时，其负端输入电压 VINN[i]已连接到另一个通道，因此，此通道不再可用于单端模式或差分模式，必须不能配置转换。ADC1/ADC2/ADC3 共享一些通道：这可能导致另一个 ADC 上的通道不可用。唯一的例外是当主 ADC 和从 ADC 在交叉模式下运行。

14.3.5 ADC 校准

该 ADC 具有校准功能，用户可通过软件使能进行校准。在校准期间，ADC 计算一个用于 ADC 内部的校准因子（ADC 断电后丢失）。在 ADC 校准期间、未完成校准前，应用不能使用 ADC 模块。在使用 ADC 转换前，建议用户进行校准操作。校准用于消除芯片和芯片之间由于工艺，温度变化引起的失调误差和失配误差。

14.3.5.1 ADC 软件校准

软件设置 ADCAL=1 可启动校准，校准只能在 ADC 未打开时（ADEN=0)启动，且仅支持选择系统时钟作为 ADC 的时钟校准时钟频率是正常转换时钟频率的 1/4。当校准完成后，ADCAL 被硬件清 0。当 ADC 的工作条件发生改变时（V_{CC} 改变是失调误差和失配误差的主要因素，温度改变次之），推荐进行再次校准操作。

通过设置 ADC_CFGR2 寄存器中 CALNUM[2:0]来配置校准过程的重复次数，并对结果进行平均以得到更精确的校准结果。(注意:校准平均结果补偿到 AD 转换时，CALNUM 不能为 0)

内部校准寄存器可通过设置 ADC_CR 的 RSTCAL 复位，该位由硬件清除置 0。在校准寄存器被初始化后(即 RSTCAL 置 1 后)，该位即被清除。

校准的软件操作过程：

1. 确认 ADEN=0
2. 设置 RSTCAL（该步骤是可选的）；
3. 设置校准次数 CALNUM（该步骤是可选的）；
4. 设置 ADCAL=1
5. ADC 内部自动延迟 192 个 ADC CLK 等待 ADC 稳定；
6. 等待到 ADCAL=0

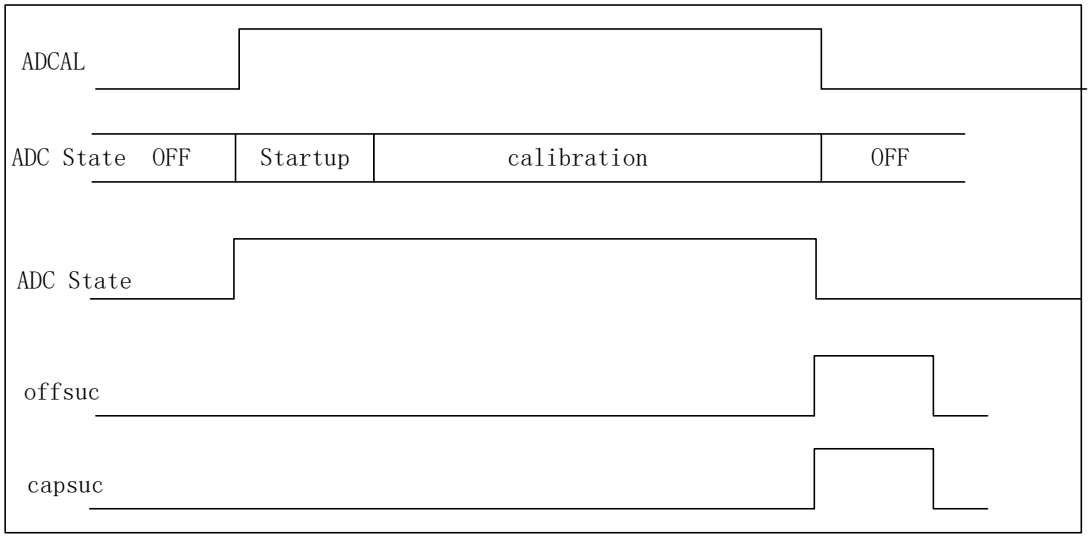


图 14-5 ADC 校准

14.3.6 ADC 开-关控制

ADC_CR 寄存器的 ADEN 是 ADC 模块的使能开关。为了省电，当 ADEN 为 0 时，ADC 模拟电路将会进入掉电模式。

注意：ADEN 位设置为 1 后，硬件会等待不少于 3 us 的延时（t_{STAB}）后进行转换。

通过清除 ADEN 位可以停止转换并复位 ADC 状态，并将 ADC 模拟电路置于断电模式。

14.3.7 写 ADC 控制位的限制

仅当 ADC 被禁用（ADEN= 0）且无校准(ADCAL=0)时，软件写 RCC 启用 ADC 时钟（参见 RCC 章节），软件可配置：

- ADC_CFGR2 的 CALNUM
- ADC_CR 寄存器中的 RSTCAL，ADCAL 和 ADEN。
- ADC_CALFACT

对于除了上述寄存器以外所有寄存器：

- ADC 没有正在进行的规则和注入转换时，才允许软件写入这些控制位。

对于 ADC_CCR 寄存器，仅当 ADC 被禁用（ADEN= 0）时，才允许软件写入这些控制位。

注意：如未按照本节所述配置 ADC 寄存器，ADC 行为可能处于未知状态。要从这种情况中恢复，必须禁用 ADC（清零 ADEN）。

14.3.8 ADC 时钟

由时钟控制器提供的 ADC CLK 和 PCLK2(APB2 时钟)同步。RCC 控制器(CLK 控制器)为 ADC 时钟提供一个专用的可编程预分频器。

14.3.9 ADC 通道选择

ADC1 有 14 个外部通道和 5 个内部通道和 1 个内部 V_{SSA} 通道，其中内部通道为：

- ADC1_CH15 连接 OPA1
- ADC1_CH16 连接温度传感器 V_{SENSE}，
- ADC1_CH17 连接 V_{BAT}/3,
- ADC1_CH18 连接 V_{REFINT}，
- ADC1_CH19 连接 DAC1。

ADC2 有 16 个外部通道和 3 个内部通道和 1 个内部 V_{SSA} 通道，其中内部通道为：

- ADC2_CH17 连接 OPA2，
- ADC2_CH18 连接 OPA3，
- ADC2_CH19 连接 DAC2。

ADC3 有 15 个外部通道和 4 个内部通道和 1 个内部 V_{SSA} 通道，其中内部通道为：

- ADC3_CH16 连接 OPA3,
- ADC3_CH17 连接 V_{BAT}/3,
- ADC3_CH18 连接 V_{REFINT}，
- ADC3_CH19 连接 DAC1。

ADC 通道选择可通过 ADC_SQRx 或 ADC_JSQR 配置，配置值和通道对应关系见上述章节表述。

ADC 将转换分为两组：规则转换和注入转换。每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：ADC_IN3、ADC_IN8、ADC_IN2、ADC_IN2、ADC_IN0、ADC_IN2、ADC_IN2、ADC_IN15。所有通道都可以按注入或规则通道组进行转换。

- 一个规则转换组最多由 16 个转换构成。转换序列的规则通道及其转换顺序在 ADC_SQRx 寄存器中选择。规则转换组中的序列长度必须写入 L[3:0] 。
- 一个注入转换组最多由 4 个转换构成。注入通道及其转换顺序在 ADC_JSQR 寄存器中选择。注入转换组中的序列长度必须写入 JL[1:0] 位。(注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。)

14.3.10强制停止 ADC

软件设置 ADC_CR 寄存器中的 ADSTP=1 可以停止当前正在进行的转换，让 ADC 进入空闲状态，为下次转换作准备。

当 ADSTP 由软件设置为 1，任何当前的转换中止且转换结果丢弃(ADC_DR 寄存器不用当前的转换值进行更新)。一旦结束转换过程，ADSTP 被硬件清零。

14.3.11动态低功耗特性

14.3.11.1 自动延迟转换模式(AUTDLY)

自动延迟转换模式可用于在低速运行时简化软件以及优化应用程序的性能，当然在这种模式下不容易产生 ADC 溢出的情况。这是一种自适应 ADC 速度和自适应系统读取 ADC 数据速度的方法。

自动延迟转换(AUTDLY)模式是指规则通道一次转换结束后需等待 CPU/DMA 读走数据寄存器的数据才可以开始下个通道的采样转换。如果 CPU/DMA 没有读走上次转换的数据，整个采样转换周期将不会切到下个通道，而是一直处于等待状态，直到读走上次转换数据后，继续进行转换。若在 AUTDLY 期间发生规则通道触发事件，该触发事件会被忽略。

AUTDLY 机制支持规则通道的所有转换模式，当使能 AUTDLY 标志位时，表明 ADC 转换周期需等待数据寄存器被读走才可以进行。

注意：AUTDLY 模式不支持注入触发打断规则转换。

14.3.12转换模式

14.3.12.1 单次转换模式

单次转换模式时，ADC 对单个通道执行一次转换，该模式能够运行在规则组和注入组。SQ1[4:0]位规定了 ADC 规则组单次转换通道，应用应保持 L 为 0；注入通道单次转换时，应用应保持 JL 为 0，（注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。）ADC_JSQR 寄存器的 JSQ4 [4:0]位配置了注入组单次转换通道。

当 ADEN 位为 1，CONT、SCAN 和 DISCEN/JDISCEN 皆 0 时，一旦相应外部触发发生，ADC 就会采样和转换一个通道。(注意:单次模式和连续模式不能同时使能，即单次模式时 CONT 为 0。)

- 如果一个规则通道被转换：

采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

- 转换数据被储存在 16 位 ADC_DR 寄存器中
- 转换结束后，EOC(转换结束)标志被设置
- 如果设置了 EOCIE，则产生中断。

- 如果一个注入通道被转换：

采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

- 转换数据被储存在 16 位的 ADC_JDRx(x=1-4)寄存器中
- 转换结束后，JEOC(注入转换结束)标志被设置
- 如果设置了 JEOCIE 位，则产生中断。

规则序列完成后：

- 设置 EOS（规则序列结束）标志
- 如果设置了 EOSIE 位，则会产生中断

注入序列完成后：

- 设置 JEOS（注入序列结束）标志
- 如果设置了 JEOSIE 位，则会产生中断

14.3.12.2 连续转换模式

连续转换模式时，ADC 对选择的通道进行连续转换，该模式能够运行在规则组。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

- 规则通道组：连续转换的通道由 SQ1[4:0]位设置。
- 注入通道组：只执行单次转换，转换的通道 SQx[4:0](x=1-4)设置。注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始

当 ADEN 位为 1，CONT 为 1 时，一旦相应外部触发发生，ADC 采样和转换选择的通道。

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位的 ADC_DR 寄存器中
  - 转换结束后，EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，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_JDRx(x=1-4)寄存器中
  - 转换结束后，JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

转换序列完成后：

- 设置 EOS（序列结束）标志
- 如果设置了 EOSIE 位，则会产生中断

然后，新的序列立即重新开始，并且 ADC 连续地重复转换序列。

注入序列完成后：

- 设置 JEOS（注入序列结束）标志
- 如果设置了 JEOSIE 位，则会产生中断

注意：不能同时启用间断转换模式和连续模式：禁止同时设置 JDISCEN/DISCEN=1 和 CONT=1。注入的通道不支持连续转换模式。

14.3.12.3 扫描模式

扫描转换模式时，ADC 对选择的一个序列所有通道连续转换，该模式能够运行在规则组和注入组。ADC_SQRx 规定了一个规则序列长度以及所有转换通道，其中 L[3:0]位规定了该序列长度。ADC_JSQR 寄存器规定了一个注入组序列长度以及所有转换通道，其中 JL[1:0]位规定了该序列长度（注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始）。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

当 ADEN 位为 1，SCAN 为 1 时，一旦相应外部触发发生，ADC 采样和转换一个序列。

- 如果一个规则序列被转换：
  - 转换数据被储存在 16 位的 ADC_DR 寄存器中
  - 每个通道转换结束后，EOC(转换结束)标志被设置，整个序列转换结束后，EOS 标志被设置。
  - 如果设置了 EOCIE，则产生中断，如果设置了 EOSIE，则产生中断。
- 如果一个注入序列被转换：
  - 转换数据被储存在 16 位的 ADC_JDRx(x=1-4)寄存器中

– 每个通道转换结束后，JEOC(注入转换结束)标志被设置，整个序列转换结束后，JEOS 标志被设置。

– 如果设置了 JEOCIE 位，则产生中断。如果设置了 JEOSIE，则产生中断。

14.3.12.4 间断转换模式

间断转换模式时，ADC 将选择的一个序列分成子序列(序列长度 1-8)，通过多次外部触发子序列，完成整个序列转换，该模式能够运行在规则组和注入组。 DISCNUM 规定了规则通道子序列长度，ADC_SQRx(x=1-4)规定了一个规则序列所有转换通道，其中 L[3:0]位规定了该序列长度。ADC_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 JL[1:0]位规定了该序列长度(注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始)。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

注意：注入通道组的子序列为 1.该模式下禁止 DISCEN 与 JDISCEN 同时使能

■ 规则组

一个外部触发信号启动 ADC_SQRx(x=1-4)描述通道 n(n<=8)次转换，直到此序列所有转换完成为止。总序列长度由 L[3:0]定义。

例如：

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，每个通道转换结束产生 EOC 事件。

第二次触发：转换的序列为 3，6，7，每个通道转换结束产生 EOC 事件。

第三次触发：转换的序列为 9，10，每个通道转换结束产生 EOC 事件。序列结束产生 EOS 事件。

第四次触发：转换的序列 0，1，2，每个通道转换结束产生 EOC 事件。

注：当一规则组以间断模式转换时，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

一个外部触发信号启动 ADC_JSQR 描述通道 1 次转换，直到序列中所有转换完成为止。总序列长度 JL[1:0]位定义。

例如：

n=1，被转换的通道 = 1，2，3

第一次触发：通道 1 被转换，每个通道转换结束产生 JEOC 事件。

第二次触发：通道 2 被转换，每个通道转换结束产生 JEOC 事件。

第三次触发：通道 3 被转换，每个通道转换结束产生 JEOC 事件。序列结束产生 JEOS 事件。

第四次触发：通道 1 被转换，每个通道转换结束产生 JEOC 事件。

注：

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同时设置间断模式。间断模式只能作用于一组转换。

14.3.13注入通道管理

注入通道外部触发优先级高于规则通道外部触发，即注入通道外部触发可以中断正在进行中的规则通道转换。注入通道有两种中断方式：触发注入和自动注入。

14.3.13.1 触发注入

ADC 在执行一个规则序列过程中，检测到注入通道外部触发，则中断当前规则序列正在转换的通道，开始执行注入序列转换，当注入序列全部转换结束后，ADC 恢复被中断的规则通道，完成规则序列转换。

ADC_SQRx (x=1-4) 规定了一个规则序列所有转换通道，其中 L[3:0]位规定了该序列长度。ADC_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 JL[1:0]位规定了该序列长度。注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。

注意：触发注入时，JAUTO 必须为 0。注入通道转换期间发生规则触发事件，该事件会在注入通道完成后进行转换，即从头开始规则序列而不是恢复被中断的规则通道。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

(注意：间断模式不支持触发注入；触发注入时不支持 AUTDLY 功能；当规则触发和注入触发同时发生时，规则触发被丢弃，只执行注入触发；注入期间内的规则触发被忽略。)

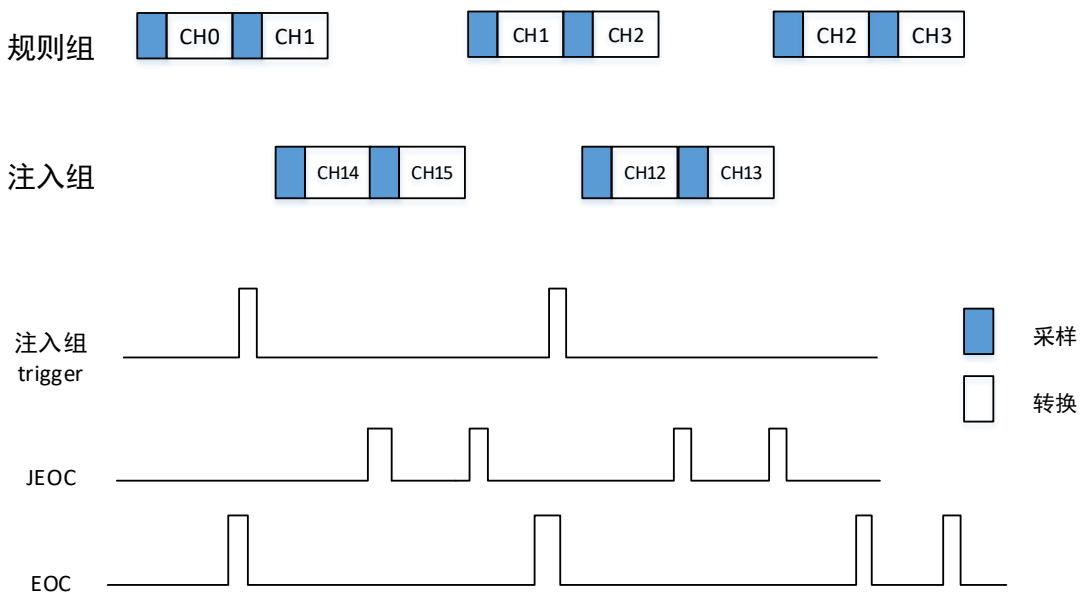


图 14-6 ADC 触发注入(扫描模式)

14.3.13.2 自动注入

自动注入模式时，ADC 完成一个规则序列后，自动执行注入序列转换。JAUTO 为自动注入模式使能位。ADC_SQRx (x=1-4) 规定了一个规则序列所有转换通道，其中 L[3:0]位规定了该序列长度。ADC_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 JL[1:0]位规定了该序列长度。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

非过采样模式的自动注入，注入序列结束后，JEOS,EOS 置位。

过采样模式的自动注入，规则序列结束后，EOS 置位，注入序列结束后，JEOS 置位。

（注意此模式必须禁止注入通道的外部触发；不能同时使用自动注入和间断模式；自动注入不支持 AUTDLY 模式）

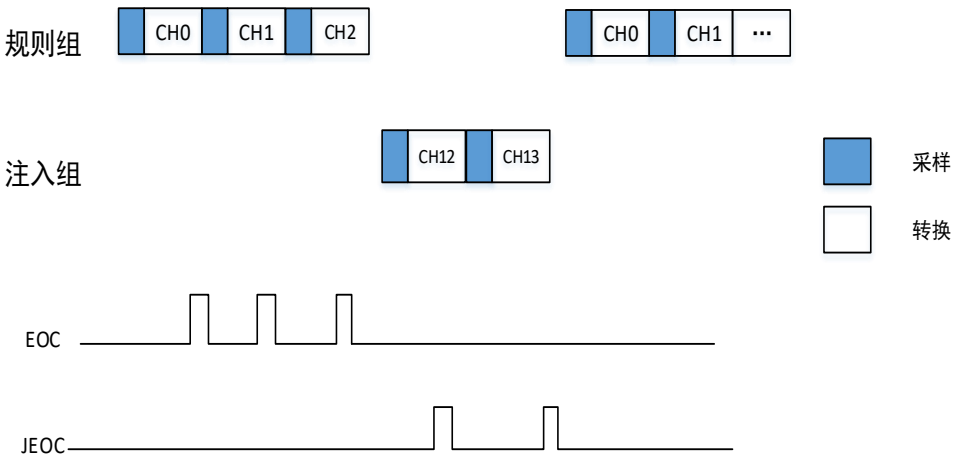


图 14-7 ADC 自动注入(CONT+SCAN+JAUTO)

14.3.14模拟看门狗

三个 AWD 模拟看门狗监测 ADC 通道是否保持在配置的电压范围内（窗口）

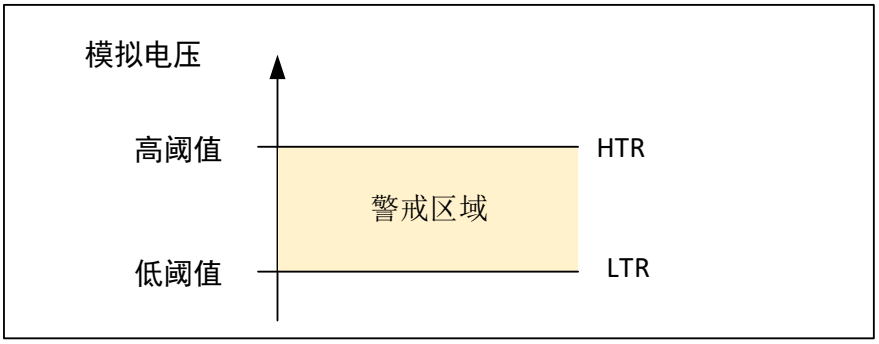


图 14-8 模拟看门狗保护区域

14.3.14.1 AWDx 标志和中断

通过在 ADC_IER 寄存器中设置 AWDxIE（x=1,2,3），可以为 3 个模拟看门狗中的每一个启用中断。AWDx（x=1,2,3）标志由软件写 1 来清除。在数据对齐之前将 ADC 转换结果与低阈值，高阈值进行比较。

14.3.14.2 模拟看门狗 1 描述

通过设置的 JAWD1EN/AWD1EN 位启用 AWD 模拟看门狗 1。该看门狗监视一个选定的注入/规则通道或所有启用的注入和规则通道是否保持在配置的电压范围（窗口）内。

下表显示了应如何配置 ADC_CFGR 寄存器以启用一个或多个通道上的模拟看门狗。

表 14-1 模拟看门狗通道选择

模拟看门狗保护通道	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有规则和注入通道	0	1	1
单个注入通道	1	0	1
单个规则通道	1	1	0
单个规则/注入通道	1	1	1

1. 通过 AWD1CH[4:0]选择 AWD1SGL 时看门狗保护通道，被保护通道应以规则或注入序列进行转换。

如果 ADC 转换的模拟电压低于低阈值或高于高阈值，则设置 AWD1 模拟看门狗状态位。这些阈值被写在模拟看门狗 1 的 ADC_TR1 寄存器的位 HT1[11:0]和 LT1[11:0]中。当转换分辨率小于 12 位的数据时（根据位 RES[1:0]），阈值的 LSB 必须保持为 0，因为内部比较总是在完整的 12 位原始转换数据上执行（左对齐）。

下表描述了如何针对模拟看门狗 1 的所有可能的分辨率执行比较。

表 14-2 模拟看门狗 1 比较

分辨率(RES[1:0])	模拟看门狗比较:		描述
	原始转换左对齐	阈值	
00：12bit	DATA[11:0]	LT1[11:0]和 HT1[11:0]	-
01：10bit	DATA[11:2]，00	LT1[11:0]和 HT1[11:0]	用户必须将 LT1[1:0]和 HT1[1:0]配置为 00
10：8bit	DATA[11:4]，0000	LT1[11:0]和 HT1[11:0]	用户必须将 LT1[3:0]和 HT1[3:0]配置为 00
11：6bit	DATA[11:6]，000000	LT1[11:0]和 HT1[11:0]	用户必须将 LT1[5:0]和 HT1[5:0]配置为 00

1. 有关模拟看门狗比较的更多详细信息，请参阅“增益补偿”一节。

14.3.14.3 看门狗 1 的模拟看门狗滤波器

当 ADC 仅配置一个输入通道（不允许在扫描模式下选择多个通道）时，可以通过 ADC_TR1 寄存器配置有效的 ADC 转换数据间隔：

- 当转换结果超出所配置的高/低阈值，但在 ADC_TR1 中定义的间隔内时，会每个转换结果会生成 DMA 请求(若使能 DMA)。
- 否则，不会发出 DMA 请求(即连续多个转换结果超出配置的阈值，且超过间隔，超出间隔的该次转换结果不会产生 DMA 请求(若使能 DMA))。RDATA 寄存器在每次转换时都会更新。如果数据超出范围，比 ADC_TR1 的 AWDFILT 位中指定的值高出若干倍，则设置 AWDx 标志，并产生相应的中断。

14.3.14.4 看门狗 2 和 3 的描述

第 2 和第 3 模拟看门狗更灵活，可以通过对 AWDxCH[19:0]（x=2，3）中的相应位进行编程来保护几个选定的通道。当 AWDxCH[19:0]（x=2,3）的任何位被设置时，相应的看门狗被启用。它们被限制为 8 位的分辨率，并且只有阈值的 8 个 MSB 可以被编程为 HTx[7:0]和 LTx[7:0]（x=2-3）。下表描述了如何针对所有可能的分辨率执行比较。

表 14-3 模拟看门狗 2 和 3 比较

分辨率(RES[1:0])	模拟看门狗比较：		描述
	原始转换左对齐	阈值	
00：12bit	DATA[11:4]	LTx[7:0]和 HTx[7:0]	数据[3:0]与比较无关
01：10bit	DATA[11:4]	LTx[7:0]和 HTx[7:0]	数据[3:2]与比较无关
10：8bit	DATA[11:4]	LTx[7:0]和 HTx[7:0]	-
11：6bit	DATA[11:6]，00	LTx[7:0]和 HTx[7:0]	用户必须将 LTx[1:0]和 HTx[1:0]配置为 00

14.3.14.5 ADCy_AWDx_OUT 信号输出产生

每个模拟看门狗都与内部硬件信号 ADCy_AWDx_OUT（y=1,2,3;x=1,2,3）相关联，该信号直接连接到一些片上定时器的 ETR 输入（外部触发）。请参阅片上定时器部分，了解如何选择 ADCy_AWDx_OUT 信号作为 ETR。

启用相关的模拟看门狗时，ADCy_AWDx_OUT 被激活：

- 当被保护转换超出编程阈值时，设置 ADCy_AWDx_OUT。
- ADCy_AWDx_OUT 在阈值内的下一个保护转换结束后复位 (如果下一个模拟看门狗通道转换仍在编程阈值外，则保持为 1)。
- 当禁用 ADC 时（当设置 ADEN=0 时），ADCy_AWDx_OUT 被复位。请注意，停止规则或注入的转换（设置 ADSTP=1）对 ADCy_AWDx_OUT 的生成没有影响。

注意：AWDx 标志由硬件设置，由软件复位：AWDx 标志对 ADCy_AWDx_OUT 的生成没有影响（例如：如果软件未清除 AWDx，AWDx 标志保持为 1，ADCy_AWDx_OUT 可以翻转）。

表 14-4 ADCy_AWDx_OUT 连接关系

ADC1_AWDx_OUT			ADC2_AWDx_OUT			ADC3_AWDx_OUT		
AWD1	AWD2	AWD3	AWD1	AWD2	AWD3	AWD1	AWD2	AWD3
tim1_etr5	tim1_etr6	tim1_etr7	tim3_etr8	tim3_etr9	tim3_etr10	tim8_etr8	tim8_etr9	tim8_etr10
-	-	-	tim8_etr5	tim8_etr6	tim8_etr7	-	-	-

14.3.14.6 带增益和偏移补偿的模拟看门狗

当增益和偏移补偿被启用时，模拟看门狗在补偿数据之后比较阈值。

注意：启用偏移补偿时（ADC_OFRy 寄存器中的 OFFSETy_EN 设置为 1），数据上溢或下溢可能导致错误的看门狗结果。启用溢出保护时（在 ADC_OFRy 中将 SATEN 设置为 1），看门狗会提供正确的结果。但是这会阻止使用有符号数据格式。

14.3.15数据处理

14.3.15.1 数据对齐

ADC_CFGR 寄存器中的 ALIGN 位用于选择转换后存储的数据的对齐方式，可选择左对齐和右对齐两种方式。转换的数据值已经减去了 ADC_OFRy 寄存器中自定义的偏移量，因此结果可以是一个负值。SEXT 位表示扩展的符号值。

特殊情况：左对齐时，除分辨率为 6 位外，其他分辨率数据以半字为基础对齐。分辨率 6 位时，数据以字节为基础进行对齐。

注意：在过采样模式下不支持左对齐。当 ROVSE 或 JOVSE 位被设置时，ALIGN 位值被忽略，ADC 仅提供右对齐数据。

14.3.15.2 offset

通过设置 ADC_OFRy 寄存器中 OFFSETy_EN = 1，可以将偏移 y（y = 1、2、3、4）施加到通道。应用可通过 ADC_OFRy 寄存器的位 OFFSETy_CH[4:0]选择偏移的通道。在这种情况下，转换结果减去 OFFSETy[11:0]中的用户定义的偏移量，结果可能是负值，因此读取的数据是有符号的，并且 SEXT 位表示扩展的符号值。

注意：过采样模式下不支持 offset。设置 ROVSE 或 JOVSE 位时，ADC_OFRy 寄存器中 OFFSETy_EN 位的值被忽略（视为复位）。

下表描述了如何针对 offset 的所有可能的分辨率执行比较。

表 14-5 Offset 计算值 vs 数据分辨率

分辨率(RES[1:0])	原始转换结果和 offset 减法		Result	描述
	原始转换结果 左对齐	Offset		
00:12bit	DATA[11:0]	OFFSET[11:0]	有符号 12bit 数	-
01:10bit	{DATA[11:2],00}	OFFSET[11:0]	有符号 10bit 数	用户必须配置 OFFSET[1:0]=00
10:8bit	{DATA[11:4],0000}	OFFSET[11:0]	有符号 8bit 数	用户必须配置 OFFSET[3:0]=0000
11:6bit	{DATA[11:6],000000}	OFFSET[11:0]	有符号 6bit 数	用户必须配置 OFFSET[5:0]=000000

从与通道“i”对应的 ADC_DR（规则通道）或 ADC_JDRy（注入通道，y=1,2,3,4）读数据时：

- 如果对应通道的偏移之一被使能（位 OFFSETy_EN = 1），则读数据时有符号的。
- 如果此通道的四个偏移均未启用，则读数据无符号。

下图所示为有符号数和无符号数数据对齐。

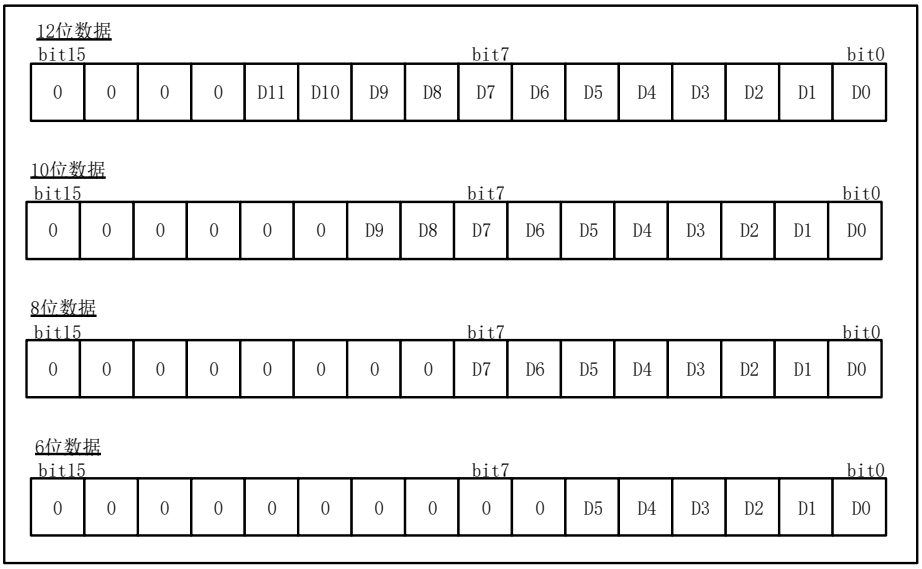


图 14-9 右对齐（offset 未使能，无符号数）

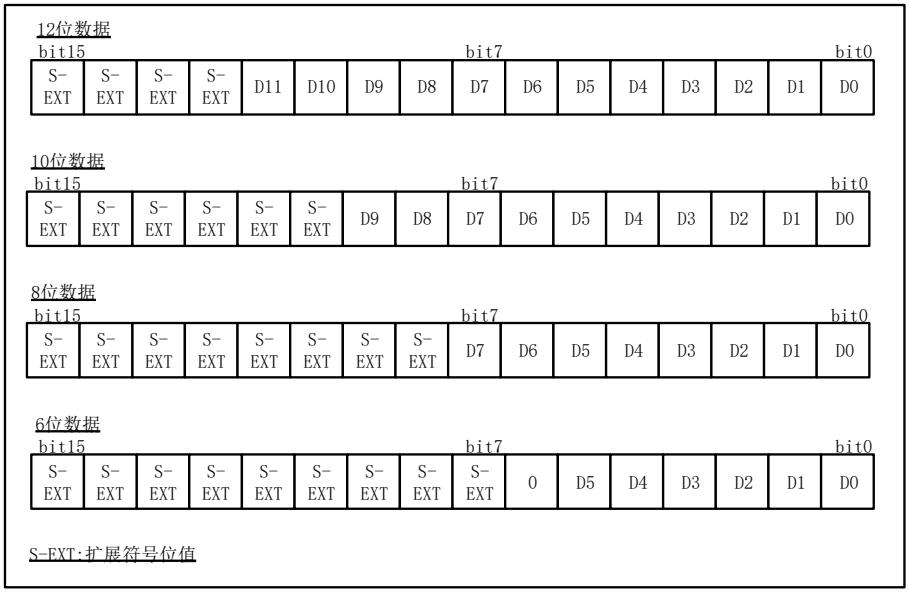


图 14-10 右对齐（offset 使能，有符号数）

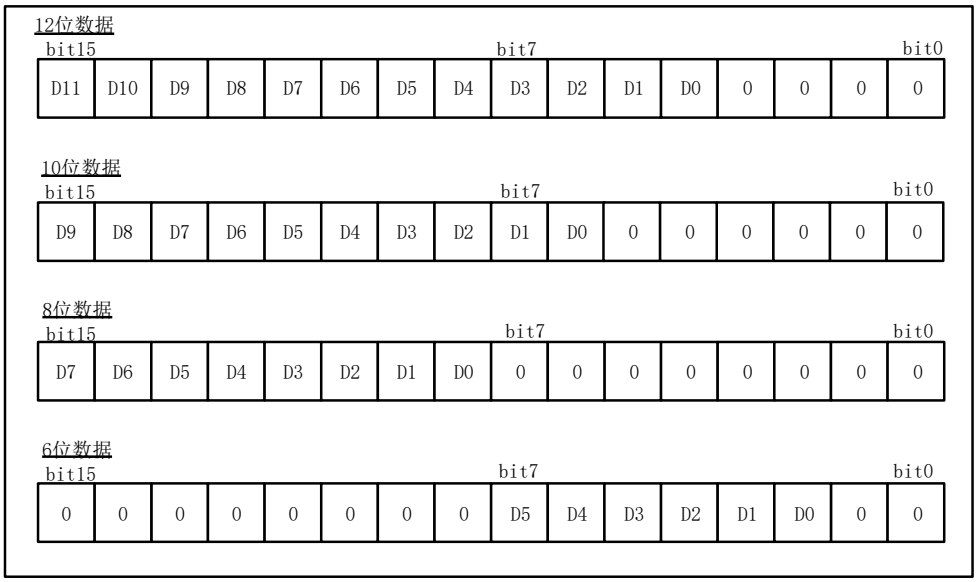


图 14-11 左对齐（offset 未使能，无符号数）

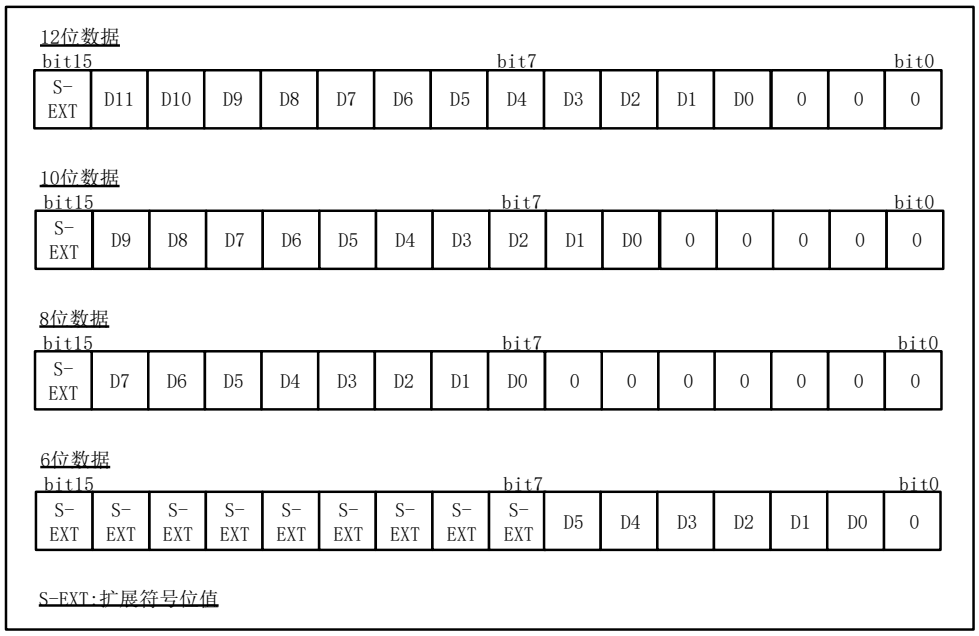


图 14-12 左对齐（offset 使能，有符号数）

14.3.15.3 增益补偿

当设置 ADC_CFGR2 寄存器 GCOMP 位时，对所有转换后的数据进行增益补偿。每次转换后，数据将使用以下公式进行计算。

DATA = DATA(adc result）x(GCOMPCOEFF)/4096

由于 GCOMPCOEFF 范围为 0-16383，因此实际增益补偿因子范围 0-3.999756。

在将结果数据存储在 RDATA 或 JDATAx 寄存器中之前，将计算 LSB−1 值四舍五入，并将误差降至最低。

增益补偿对于过采样也是有效的。当增益补偿用于过采样模式时，在累加和右移操作之后执行增益计算，以最小化功耗（增益计算仅进行一次，而不是每次转换）。

14.3.15.4 Offset 补偿

当使能偏移同时在 ADC_OFRy 寄存器中设置 SATEN 位时，数据是无符号的。所有偏移数据在 0x000 处饱和（在 12 位分辨率）。当 OFFSETPOS 位被设置时，偏移方向是正的，并且数据在 0xFFF 饱和（在 12 位分辨率）。在 8 位分辨率中，数据分别在 0x00 和 0xFF 处饱和。

在偏移和增益补偿之后，对无符号数执行模拟看门狗比较。为了正确的看门狗操作，偏移补偿后的数据必须为无符号格式（ADC_OFRy 寄存器中的 SATEN 位设置为 1）。

14.3.16过采样器

过采样单元执行数据预处理以卸载 CPU 计算负荷。它能够处理多个转换，并将它们平均为单个数据(增大数据宽度，最高可达 16 位)。

过采样器提供了以下公式，其中 N 和 M 可以调整：

$$Result = \frac{1}{M} \times \sum_{n=0}^{n=N-1} Conversion(t_n)$$

过采样器允许通过硬件执行以下功能：平均、降低数据速率、提高信噪比、基本滤波。

过采样率 N 是通过 ADC_CFGR2 寄存器中的 OVSR[2:0]位定义的，可在 2x 到 256x 的范围内调节。除法系数 M 由右移实现，最多 8 位。并且使用 ADC_CFGR2 寄存器中的 OVSS[3:0]位来定义。

求和单元可以产生高达 20 位的结果 (256x 12 位结果) , 该结果首先右移。然后，在最终传输到 ADC_DR 数据寄存器之前被截断为 16 个最低有效位，将移位留下的最低有效位四舍五入到最接近的值。

注意：如果移位后的中间结果超过 16 位，则该结果被原样截断而不饱和。

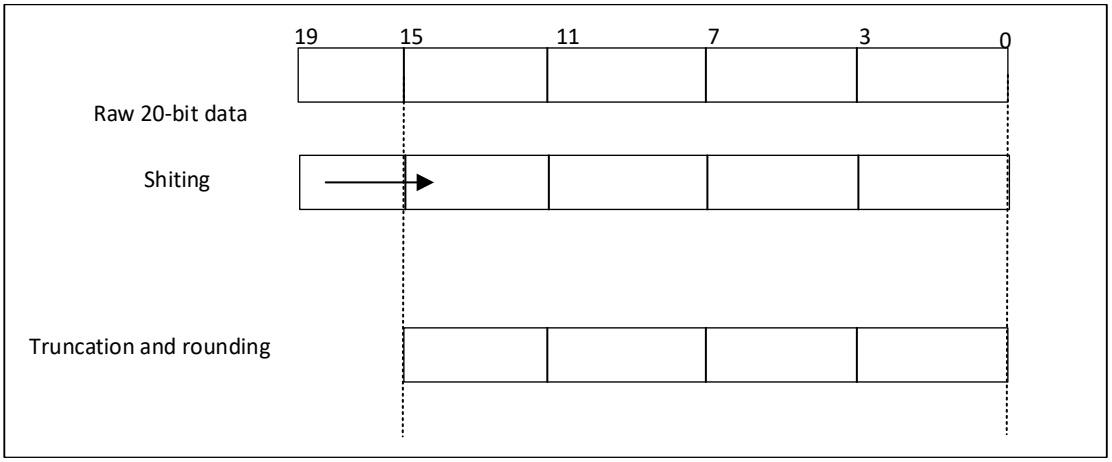


图 14-13 20 bits ~16 bits 结果截断

下图给出了处理的数字示例，从原始的 20 位累加数据到最终的 16 位结果。

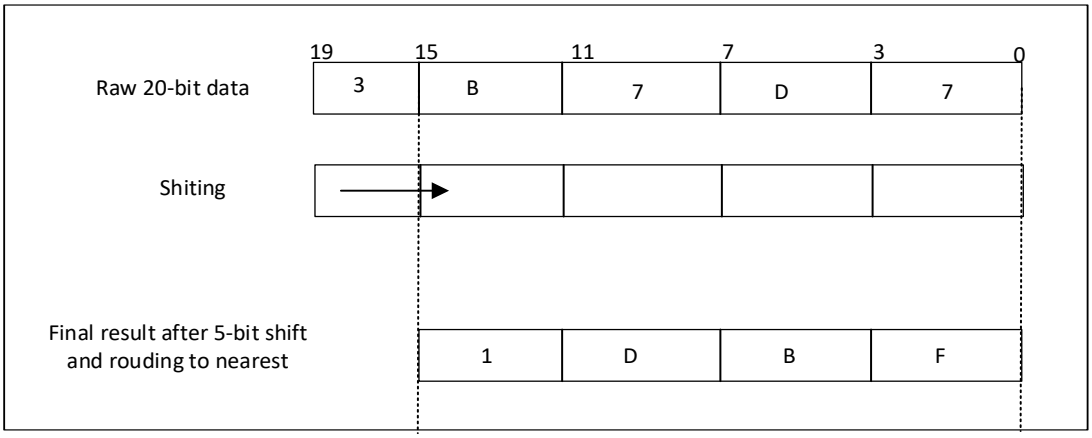


图 14-14 具有 5 位移位和舍入的数值示例

下表给出了原始转换数据为 0xFFF 的各种 N 和 M 组合的数据格式。

表 14-6 最大输出结果与 N 和 M 的关系（灰色单元格表示截断）

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800

256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

过采样模式下的转换时序没有变化：在整个过采样序列中，采样时间保持相等。每 N 个转换提供一个新数据，通过 N x T_{CONV} = N x（t_{SMPL}+t_{SAR}）等效延迟。标志设置如下：

- 采样阶段结束（EOSMP）设置在每个采样阶段之后
- 当过采样结果可用时，每 N 次转换发生一次转换结束（EOC）
- 序列结束（EOS）发生在过采样数据序列完成之后（即在 N x 总序列长度转换之后）

14.3.16.1 过采样时不支持的 ADC 操作模式（单个 ADC 模式和双 ADC 模式）

在过采样模式下，不支持注入打断规则。

14.3.16.2 过采样时支持的 ADC 操作模式（单个 ADC 模式）

在过采样模式下，大多数 ADC 操作模式都保持不变：

- Single 模式/连续模式转换
- 通过软件或外部触发启动 ADC 转换
- 转换期间停止 ADC
- 通过 CPU/DMA 读数据
- 低功耗模式(AUTDLY)
- 可编程分辨率：非 12 bits 分辨率转换结果（根据 ADC_CFGR 寄存器中的 RES [1:0]位）与 12 位转换相同的方式进行累加、截断、取整和移位。

注意：使用过采样数据时，数据对齐不可用。ADC_CFGR 中的 ALIGN 位被忽略，并且数据总是以右对齐的方式提供。过采样模式下不支持偏移校正。设置 ROVSE 或 JOVSE 位时，ADC_OFRy 寄存器中 OFFSETy_EN 位的值被忽略（视为重置）。

14.3.16.3 模拟看门狗

保持模拟看门狗功能（AWD1SGL，AWD1EN，JAWD1EN），但有以下区别：

- 忽略 RES[1:0]位，始终使用完整的 12 位值 HT[11:0]和 LT[11:0]进行比较
- 对 16 位过采样结果 ADC_DR[15:4]中的最高有效 12 位执行比较

注意：使用高移位值时必须小心，这会减小比较范围。例如，如果过采样结果被移位 4 位，从而产生 12 位的数据右对齐，则有效的模拟看门狗比较只能在 8 位上执行。在 ADC_DR[11:4]和 HT[0:7]/LT[0:7]之间进行比较，并且 HT[11:8]/LT[11:8]必须复位。

14.3.16.4 触发模式

平均器也可以用于基本的过滤。虽然不是一个非常强大的滤波器（缓慢的滚降和有限的阻带衰减），但它可以用作陷波滤波器来抑制恒定的寄生频率（通常来自电源或开关模式电源）。为此，可以使用 ADC_CFGR2 中的 TROVS 位启用特定的间断模式，以便能够具有由用户定义的过采样频率，并且独立于转换时间本身。

下面的图显示了在间断模式下如何响应触发启动转换。如果设置了 TROVS 位，则忽略 DISCEN 位的内容并将其视为 1。

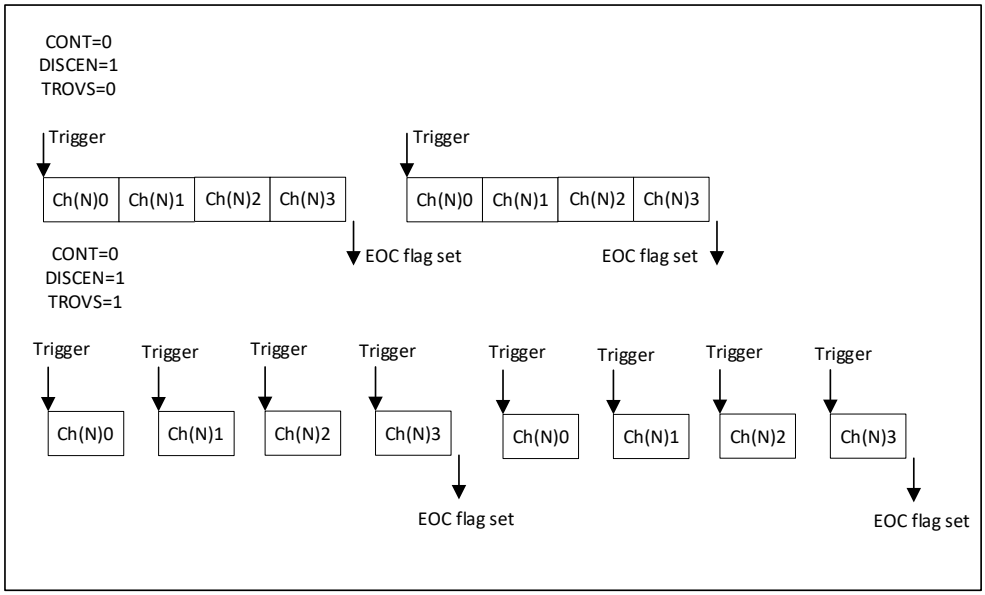


图 14-15 规则的过采样模式触发（TROVS 位=1）

14.3.16.5 仅对规则通道进行过采样

规则过采样模式位 ROVSM 定义了如果规则过采样序列被注入的转换中断，则如何恢复该序列：

- 在连续模式中，累加从最后一个规则通道有效数据重新开始（注入的触发而导致之前转换中止）。这确保了无论注入频率如何都将完成过采样（提供至少完成一次规则转换在触发之间）；
- 在恢复模式下，累加从 0 重新开始（忽略以前的转换结果）。该模式允许保证用于过采样的所有数据在单个时隙内背靠背地转换。必须注意使注入触发周期高于过采样周期长度。如果不遵守此条件，则过采样将无法完成，规则序列将被阻塞。

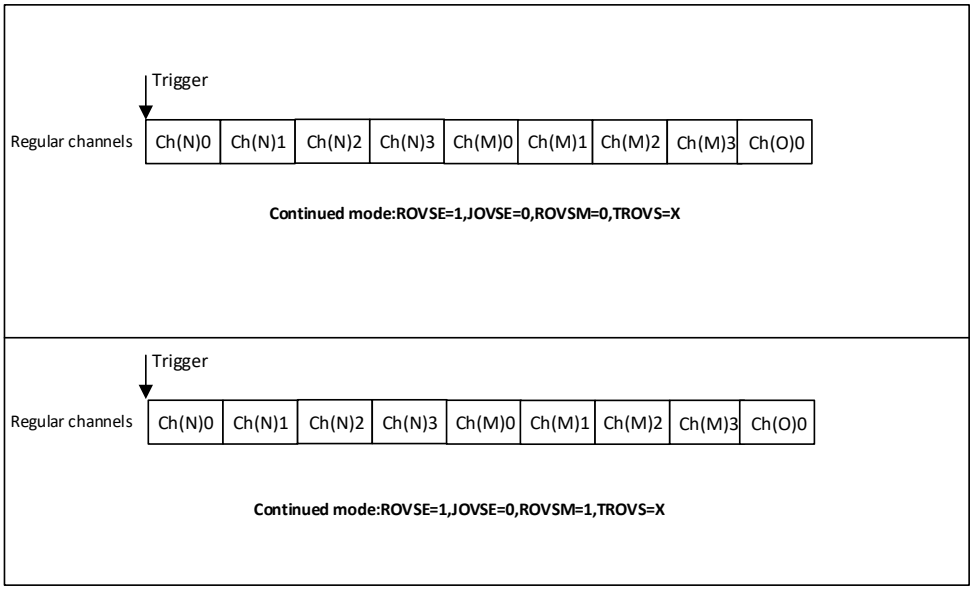


图 14-16 4x 过采样率的例子

14.3.16.6 仅对注入通道进行过采样

注入过采样模式位 JOVSE 仅为注入序列中的转换启用过采样。

14.3.16.7 自动注入模式

可以对自动注入的序列进行过采样，并将所有转换结果存储在寄存器中，以节省 DMA 资源。此模式仅在规则和注入过采样均使能时可用：JAUTO=1、ROVSE=1 和 JOVSE=1，不支持其他组合。ROVSM 位在自动注入模式下被忽略。下图显示了转换的顺序。

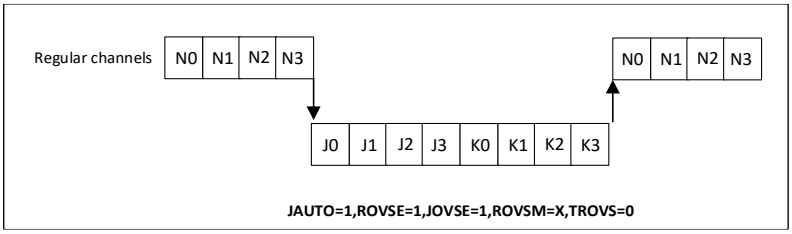


图 14-17 自动注入模式过采样

14.3.16.8 过采样时支持双 ADC 模式

对于同步注入模式和同步规则模式，在双 ADC 配置时可以启用过采样。在这种情况下，两个 ADC 必须使用完全相同的设置（包括过采样）。

当启用规则或注入过采样（ROVSE=1 或 JOVSE=1）时，不支持所有其他双 ADC 模式。

14.3.16.9 模式组合小结

下表总结了所有组合，包括不支持的模式。

表 14-7 过采样器模式小结

规则过采样 ROVSE	注入过采样 JOVSE	过采样模式 ROVSM;0=continued 1=resumed	触发规则模式 TROVS	描述
1	0	0	0	规则连续模式
1	0	0	1	不支持
1	0	1	0	规则恢复模式
1	0	1	1	触发规则恢复模式
1	1	0	X	不支持
1	1	1	0	注入和规则恢复模式
1	1	1	1	不支持
0	1	X	X	注入过采样

14.3.17 可编程采样时间

ADC 会在若干个 ADC CLK 周期内对输入电压进行采样，采样周期数通过 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMP[2:0] 位配置。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

t_{conv} = 采样时间 + 12.5 个周期

示例：

f_{ADC} = 16 MHz 且采样时间 = 3.5 个周期时：

t_{conv} = 3.5 + 12.5 = 16 个周期 = 1 μs

14.3.18外部触发转换

转换单个通道或转换一个序列可以由软件或外部事件（例如定时器捕获、输入引脚）触发。如果 EXTEN[1:0]（用于规则转换）或 JEXTEN[1:0]（用于注入转换）不等于 00，则外部事件能够触发具有所选极性的转换。

一旦软件设置了位 ADSTART = 1，规则软件触发选择就有效，而一旦软件设置位 JADSTART = 1，注入的软件触发选择就生效。在转换过程中发生的任何软硬件触发都将被忽略。

注意：在 STRT/JSTRT 有效期间，发生规则触发/注入触发，将被忽略。

下表提供了 EXTEN[1:0]和 JEXTEN[1:0]值与触发极性之间的对应关系。

表 14-8 规则外部触发配置触发极性

EXTEN[1:0]	Source
00	硬件触发检测禁用，软件触发检测启用(上升沿)
01	上升沿检测的硬件触发
10	下降沿检测的硬件触发
11	硬件触发，在上升沿和下降沿都有检测

注意：规则和注入触发的极性不能随时更改

表 14-9 注入外部触发配置触发极性

JEXTEN[1:0]	Source
00	硬件触发检测禁用，软件触发检测启用(上升沿)
01	上升沿检测的硬件触发
10	下降沿检测的硬件触发
11	硬件触发，在上升沿和下降沿都有检测

表 14-10 外部触发事件选择

ADC trigger selection EXTSEL[4:0] or JEXTSEL[4:0]	ADC triggers signals assignment			
	ADC1/2		ADC3	
	Regular	Injected	Regular	Injected
00000	tim1_cc1	tim1_trgo	tim3_cc1	tim1_trgo
00001	tim1_cc2	tim1_cc4	tim2_cc3	tim1_cc4
00010	tim1_cc3	tim2_trgo	tim1_cc3	tim2_trgo
00011	tim2_cc2	tim2_cc1	tim8_cc1	tim8_cc2
00100	tim3_trgo	tim3_cc4	tim3_trgo	tim4_cc3
00101	tim4_cc4	tim4_trgo	exti2	tim4_trgo
00110	exti11	exti15	tim4_cc1	tim4_cc4
00111	tim8_trgo	tim8_cc4	tim8_trgo	tim8_cc4
01000	tim8_trgo2	tim1_trgo2	tim8_trgo2	tim1_trgo2
01001	tim1_trgo	tim8_trgo	tim1_trgo	tim8_trgo
01010	tim1_trgo2	tim8_trgo2	tim1_trgo2	tim8_trgo2
01011	tim2_trgo	tim3_cc3	tim2_trgo	tim1_cc3
01100	tim4_trgo	tim3_trgo	tim4_trgo	tim3_trgo
01101	tim6_trgo	tim3_cc1	tim6_trgo	exti3
01110	tim15_trgo	tim6_trgo	tim15_trgo	tim6_trgo
01111	tim3_cc4	tim15_trgo	tim2_cc1	tim15_trgo
10000	tim9_cc1	tim2_cc3	tim18_cc3	tim18_cc4
10001	tim10_cc1	tim9_cc2	tim18_trgo	tim18_trgo
10010	tim11_cc1	tim13_cc1	tim18_cc1	tim9_cc1
10011	tim12_cc1	tim14_cc1	tim10_cc1	tim14_cc1

10100	tim18_trgo	tim12_cc2	tim13_cc1	tim18_cc2
10101	tim18_cc2	tim18_trgo	tim18_cc1	tim7_trgo
10110	tim19_cc1	tim18_cc1	tim7_trgo	lptim_out
10111	tim7_trgo	tim16_cc1	lptim_out	RES
11000	lptim_out	tim7_trgo	RES	Res
11001	RES	lptim_out	Res	Res
11010	Res	RES	Res	Res
11011	Res	Res	Res	Res
11100	Res	Res	Res	Res
11101	Res	Res	Res	Res
11110	Res	Res	Res	Res
11111	Res	Res	Res	Res

14.3.19 可配置分辨率

可通过降低 ADC 分辨率来执行快速转换。ADC_CFGR 寄存器的 RES 位用于选择数据寄存器中可用的位数。每种分辨率的最小转换时间如下：

- 12 位：3.5 + 12.5 = 16 f_{ADC} 周期
- 10 位：3.5 + 10.5 = 14 f_{ADC} 周期
- 8 位： 3.5 + 8.5 = 12 f_{ADC} 周期
- 6 位： 3.5 + 6.5 = 10 f_{ADC} 周期

14.3.20 DMA 请求

因为规则通道转换的值储存在一个唯一的数据寄存器中，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_DR 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_DR 寄存器传输到用户指定的目的地址。双 ADC 模式时，由 ADC2 转换的数据利用 ADC1 的 DMA 功能传输。

14.3.21 双 ADC 模式

在有 2 个 ADC 的器件中，可以使用双 ADC 模式。在双 ADC 模式里，根据 DUAL[2:0]位所选的模式，转换由 ADC1 软件/外部触触发启动。

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 种基本转换模式：

- 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

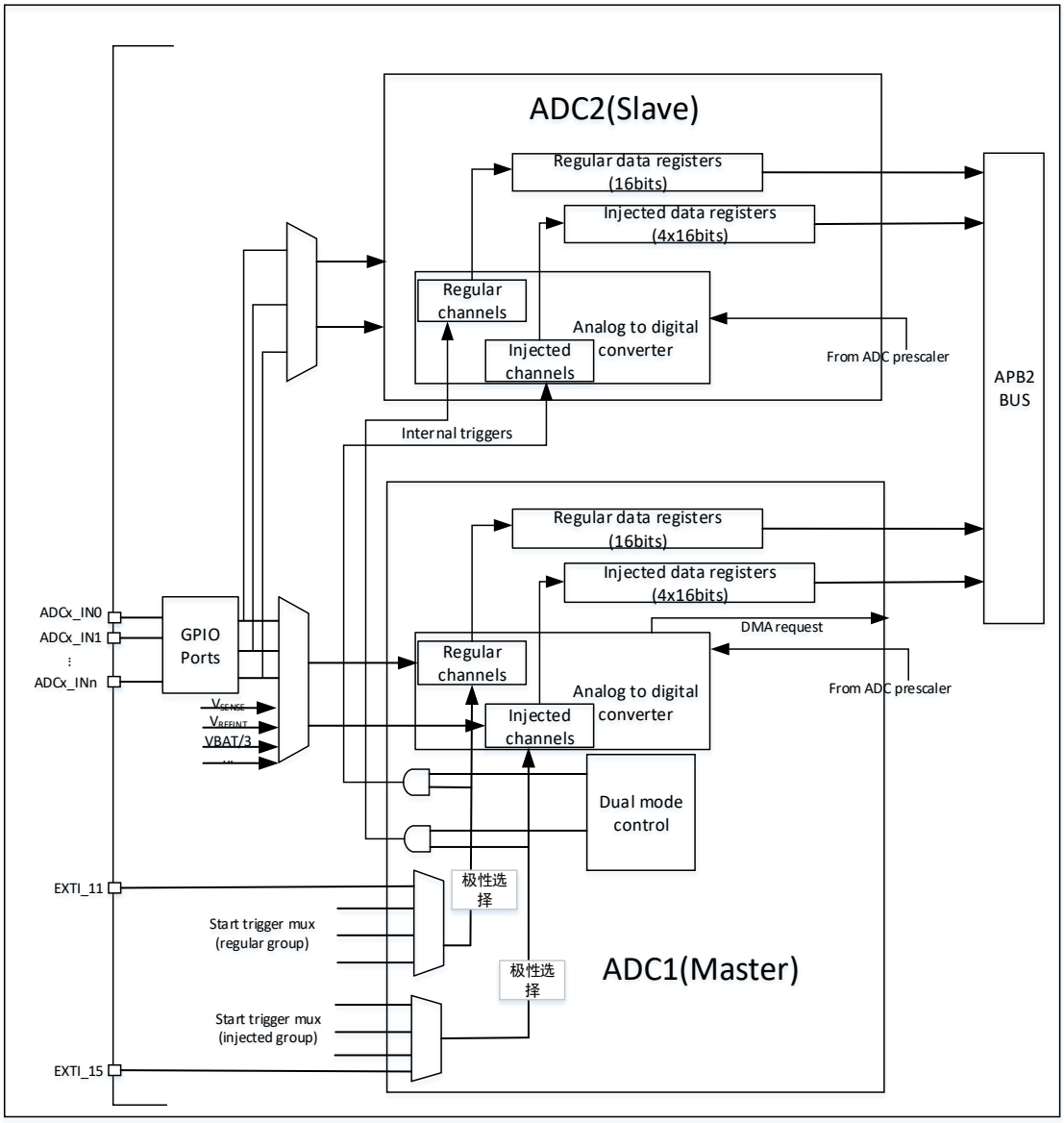


图 14-18 双 ADC 模块图

1.外部触发信号作用于 ADC2，但在本图中没有显示。

2.在双 ADC 模式中，在完整的 ADC1 数据寄存器(ADC1_DR)中包含了 ADC1 和 ADC2 的规则转换数据。

14.3.21.1 同步注入模式

同步注入模式时，ADC1 和 ADC2 同步转换注入通道组。ADC 1 和 ADC 2 的 ADC_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 ADC_JSQR 寄存器的 JL[1:0]位规定了该序列序列长度。

当 ADEN 位为 1，一旦 ADC1 外部触发(由 JEXTSEL[4:0]选择)发生，ADC1 和 ADC2 同步就会转换一个注入序列。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

注意:

1. 不要在 2 个 ADC 上转换相同的通道即两个 ADC 在同一个通道上的采样时间不能重叠
2. ADC1 和 ADC2 同步转换的通道采样时间要保持一致

在 ADC1 或 ADC2 的转换结束时:

- 转换的数据存储在每个 ADC 的 ADC_JDRx（x=1-4)寄存器中。
- 当 ADC1/ADC2 注入通道都被转换时，每个通道转换结束产生 JEOC 事件，序列结束产生 JEOS

事件。

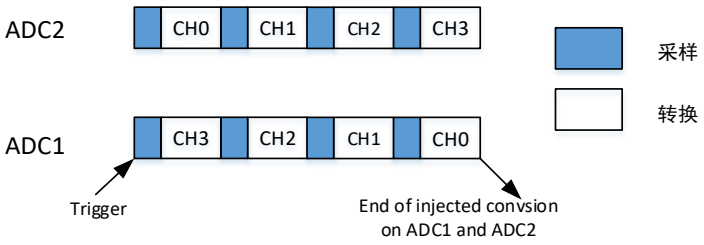


图 14-19 4 个通道上同步注入模式

14.3.21.2 同步规则模式

同步规则模式时，ADC1 和 ADC2 同步转换规则通道组。ADC 1 和 ADC 2 的 ADC_SQRx（x=1-4）寄存器规定了一个规则序列所有转换通道，其中 L[1:0]位规定了该序列长度。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

当 ADEN 位为 1，一旦 ADC1 外部触发(由 EXTSEL[4:0]选择)发生，ADC1 和 ADC2 同步就会转换一个规则序列。(注意:用户不要在 2 个 ADC 上转换相同的通道即两个 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 DMA 位)，传输到 SRAM 的 32 位 ADC1_DR 寄存器的高 16 位包含 ADC2 的转换数据，低 16 位包含 ADC1 的转换数据。
- 当 ADC1/ADC2 规则通道都被转换完时，每个通道转换结束产生 EOC 事件，序列结束产生 EOS 事件。

注：

1. 在同步规则模式中，必须转换具有相同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 转换会被重启。
2. ADC1 和 ADC2 同步转换的通道采样时间要保持一致。

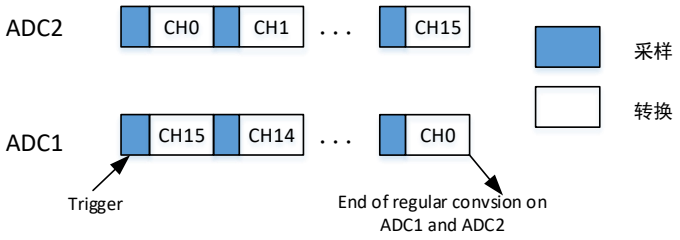


图 14-20 在 16 个通道上的同步规则模式

14.3.21.3 快速交叉模式

快速交叉模式，ADC1 和 ADC2 交替转换一个通道。此模式只应用于规则通道组（通常一个通道），ADC 1 和 ADC 2 的 SQ1 定义了所转换的通道。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

当 ADEN 位为 1 时，一旦 ADC1 外部触发(由 EXTSEL[4:0]选择)发生，ADC1 和 ADC2 就会交替转换一个规则通道。

外部触发源来自 ADC1 的规则通道多路器。外部触发产生后：

- ADC2 立即启动
- 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 CONT 位，所选的两个 ADC 规则通道将被连续地转换。ADC1 转换结束产生 EOC 标志，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的高 16 位包含 ADC2 的转换数据，低 16 位包含 ADC1 的转换数据。每个通道转换结束产生 EOC 事件，序列结束产生 EOS 事件。

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。(连续模式最大采样时间<4 个 ADCCLK 周期)

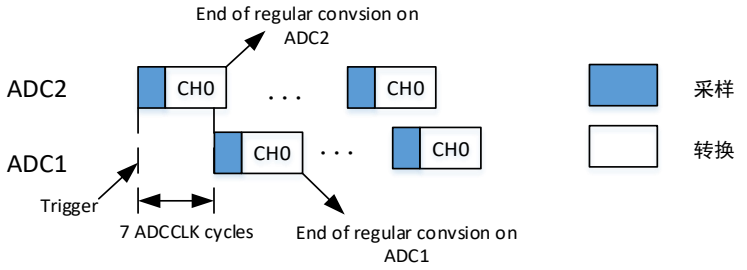


图 14-21 在一个通道上连续转换模式下的快速交叉模式

14.3.21.4 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)。外部触发源来自 ADC1 的规则通道多路器。外部触发产生后：

- ADC2 立即启动
- ADC1 在延迟 14 个 ADC 时钟周期后启动
- 在延迟第二个 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 转换结束后产生 EOC 标志，产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位)，ADC1_DR 寄存器的 32 位数据被传输到 SRAM，ADC1_DR 的高 16 位包含 ADC2 的转换数据，低 16 位包含 ADC1 的转换数据。每个通道转换结束产生 EOC 事件，序列结束产生 EOS 事件。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

在这个模式里不能设置 CONT 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。

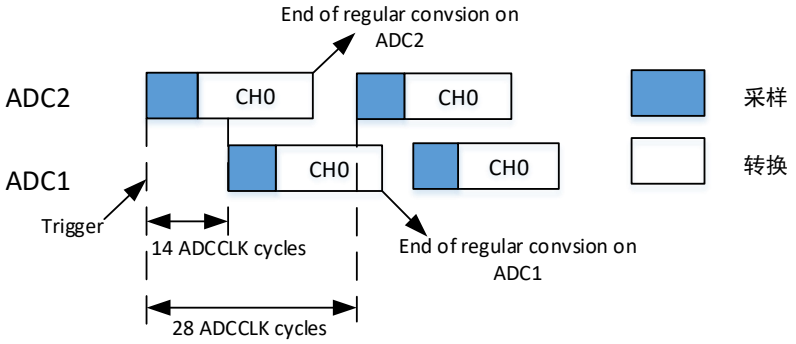


图 14-22 在一个通道上的慢速交叉模式

14.3.21.5 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 的注入通道多路器。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

如果 ADC1 和 ADC2 上同时使用注入扫描模式：

- 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 JEOS 中断，在所有 ADC1 注入组通道转换后产生一个 JEOS 中断。每个通道转换结束产生 JEOC 事件，序列结束产生 JEOS 事件。

如果允许产生 JEOS 中断，在所有 ADC2 注入组通道转换后产生一个 JEOS 中断。每个通道转换结束产生 JEOC 事件，序列结束产生 JEOS 事件。

当所有注入组通道都转换完后如果产生另一个外部触发，ADC1 注入组通道重新开始转换。

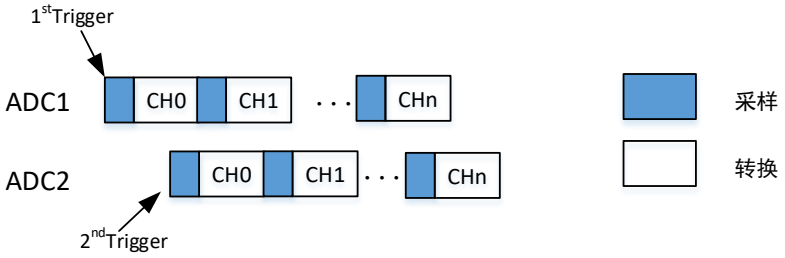


图 14-23 交替触发-每个 ADC 的注入通道组

如果 ADC1 和 ADC2 上同时使用注入间断模式：

- 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 JEOS 中断，在所有 ADC1 注入组通道转换后产生一个 JEOS 中断。每个通道转换结束产生 JEOC 事件，序列结束产生 JEOS 事件。

如果允许产生 JEOS 中断，在所有 ADC2 注入组通道转换后产生一个 JEOS 中断。每个通道转换结束产生 JEOC 事件，序列结束产生 JEOS 事件。

当所有注入组通道都转换完后，如果产生另一个外部触发，重新开始交替触发过程。

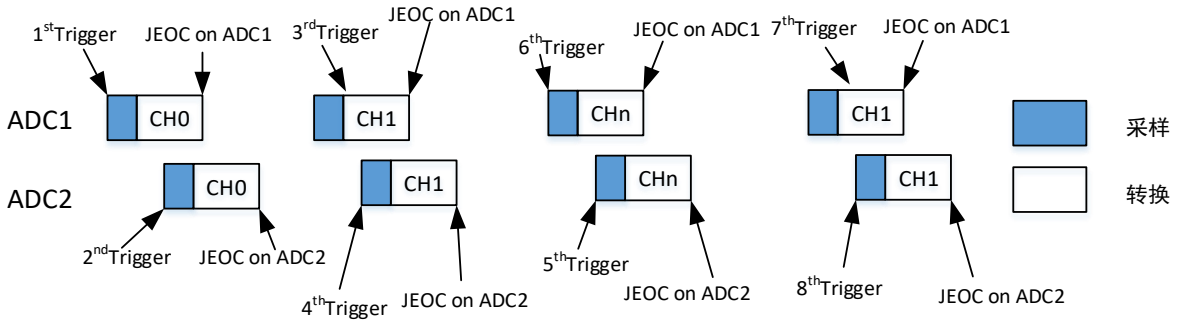


图 14-24 交替触发-在间断模式下每个 ADC 上的 4 个注入通道

14.3.21.6 独立模式

此模式，每个 ADC 独立工作，相互无关联。

14.3.21.7 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。采样结束，则产生 EOSMP，如设置了 EOSMPIE，则产生中断。

注：

1. 在混合的规则/注入同步模式中，必须转换具有相同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。
2. ADC1 和 ADC2 同步转换的通道采样时间保持一致

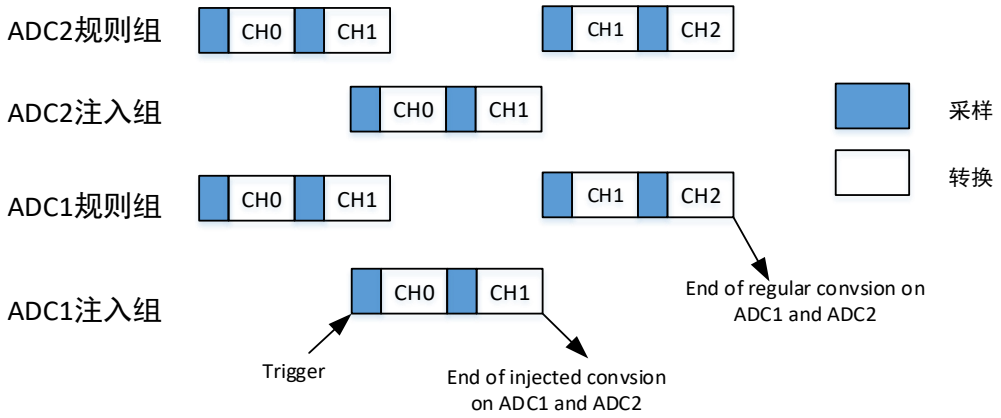


图 14-25 混合的规则/注入同步模式-scan 模式

14.3.22温度传感器和内部参考电压

温度传感器可用于测量器件节点温度（TJ)。温度传感器内部连接到 ADC1_CH16 通道，此通道把传感器输出的电压转换成数字值。不使用时，可将传感器置于掉电模式。

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。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。

注意：必须设置 VSENSESEL，VREFEN 位激活内部通道：ADC1_CH16(温度传感器)和 ADC1_CH18(VREFINT)的转换。

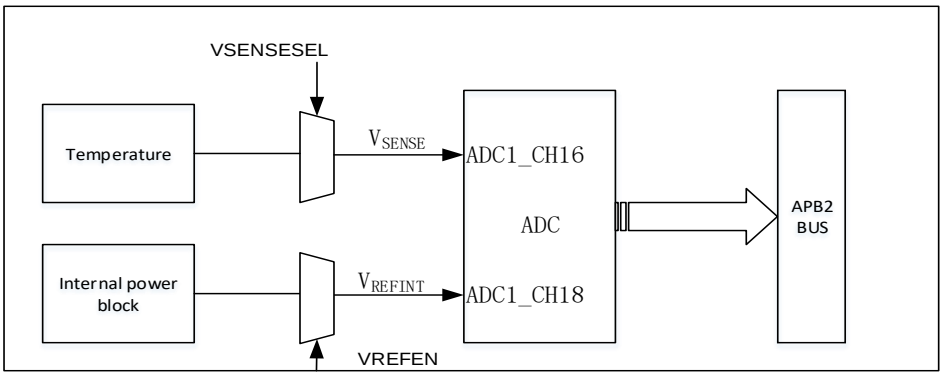


图 14-26 温度传感器通道框图

主要特性

读温度

要使用传感器，请执行以下操作：

1. 选择 ADC1_CH16
2. 选择一个采样时间，该采样时间要大于数据手册中所指定的最低采样时间。
3. 在 ADC1_CCR 寄存器中 VSENSESEL 位置 1，以便将温度传感器从掉电模式中唤醒。
4. 通过将 ADEN 位置 1，并使能外部触发，开始 ADC 转换
5. 读取 ADC 数据寄存器中生成的 V_{SENSE} 数据
6. 使用以下公式计算温度：

$$温度（°C）= \{(V_{SENSE} – V_{30}) / Avg_Slope\} + 30\ °C$$

其中：

V_{30} = 30 °C 时的 V_{SENSE} 值

Avg_Slope = 温度与 V_{SENSE} 曲线的平均斜率（单位为 mV/°C 或 μV/°C ）

（有关 V_{30} 和 Avg_Slope 实际值的相关信息，请参见数据手册中的电气特性一节。）

注意：传感器从掉电模式中唤醒需要一个建立时间，启动时间过后输出正确的 V_{SENSE}。ADC 在上电后同样需要一个建立时间，因此为了缩短延时，应同时将 ADEN 和 VSENSESEL 位置 1。

14.3.23电池监视

由于 V_{BAT} 电压可能变化，因此 V_{BAT} 引脚需要内部连接到桥接分配器，以确保 ADC 正确运行。

桥接器自动使能 V_{BAT}/3 连接到 ADC1/3_CH17 输入通道。

14.3.24 ADC 中断

对于每个 ADC，产生以下中断：

- 规则组的任何转换结束时（标志 EOC）
- 一个规则组序列结束时（标志 EOS）
- 注入组的任何转换结束时（标志 JEOC）
- 一个注入组序列结束时（标志 JEOS）
- 当发生模拟看门狗检测时（标志 AWD1、AWD2 和 AWD3）
- 当采样阶段结束时(标志 EOSMP)

独立的中断使能位具有灵活性。

表 14-11 每个 ADC 的中断

中断事件	事件 flag	使能控制位
一个规则转换结束	EOC	EOCIE
一个规则序列结束	EOS	EOSIE
一个注入转换结束	JEOC	JEOCIE
一个注入序列结束	JEOS	JEOSIE
模拟看门狗 1 状态位置位	AWD1	AWD1IE
模拟看门狗 2 状态位置位	AWD2	AWD2IE
模拟看门狗 3 状态位置位	AWD3	AWD3IE
采样阶段结束	EOSMP	EOSMPIE

14.4 寄存器描述（对于每个 ADC)

ADC1 寄存器基地址：0x4001_2400

ADC2 寄存器基地址：0x4001_2800

ADC3 寄存器基地址：0x4001_3C00

14.4.1 ADC 中断和状态寄存器（0x00:ADC_ISR)

Address offset：0x00

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	STRT	JSTRT	Res	AWD3	AWD2	AWD1	JEOS	JEOC	Res	EOS	EOC	EOSMP	Res
-			R	R	-	RC_W1					-	RC_W1	RC_W1	RC_W1	-

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	STRT	R	0	规则通道开始状态位 该位由硬件在开始转换时置位，转换结束后清零。 0：规则通道转换未开始

				1：规则通道转换已开始
11	JSTRT	R	0	注入通道开始状态位 该位由硬件在开始转换时置位，转换结束后清零。 0：注入通道转换未开始 1：注入通道转换已开始
10	保留	-	-	-
9	AWD3	RC_W1	0	模拟看门狗 3 标志。当转换的电压超过/低于 ADC_TR3 寄存器的字段 HT3[7:0]和 LT3[7:0]值时，该位由硬件设置。它通过软件写入 1 而被清除。 0：未发生模拟看门狗 3 事件（或软件已确认并清除标志事件） 1：发生模拟看门狗 3 事件
8	AWD2	RC_W1	0	模拟看门狗 2 标志。当转换的电压超过/低于 ADC_TR2 寄存器的字段 HT2[7:0]和 LT2[7:0]值时，该位由硬件设置。它通过软件写入 1 而被清除。 0：未发生模拟看门狗 2 事件（或软件已确认并清除标志事件） 1：发生模拟看门狗 2 事件
7	AWD1	RC_W1	0	模拟看门狗 1 标志。当转换的电压超过/低于 ADC_TR1 寄存器的字段 HT1[7:0]和 LT1[7:0]值时，该位由硬件设置。它通过软件写入 1 而被清除。 0：未发生模拟看门狗 1 事件（或软件已确认并清除标志事件） 1：发生模拟看门狗 1 事件
6	JEOS	RC_W1	0	注入序列转换结束标志。当注入序列的所有通道转换结束时，该位由硬件设置。它通过软件写入 1 而被清除。 0：注入序列转换未完成（或标志事件已被软件确认并清除） 1：注入序列转换已完成
5	JEOC	RC_W1	0	注入通道转换结束标志。当每次注入通道转换结束时，新数据可通过相应的 ADC_JDRy 寄存器中访问时，该位由硬件设置。通过软件写入 1 或通过读取相应的 ADC_JDRy 寄存器来清除。 0：注入通道转换未完成（或标志事件已被软件确认并清除） 1：注入通道转换已完成
4	保留	-	-	-
3	EOS	RC_W1	0	规则序列结束标志 规则序列转换结束时硬件置位该位。软件写 1 清零 0：转换序列没有完成（或者软件已经应答和清除该标志） 1：转换序列完成

2	EOC	RC_W1	0	转换结束标志 当每个规则通道每次转换结束后，新的数据结果可以从 ADC_DR 寄存器读到时，硬件置位该位。软件写 1 清零或读 ADC_DR 寄存器清零 0：通道转换没有完成（或者软件已经应答和清除该标志） 1：通道转换已完成
1	EOSMP	RC_W1	0	采样结束标志，在每次规则通道转换的采样阶段结束时，硬件置位该位，软件写 1 清零 0：不处在采样阶段结束时（或者软件已经应答和清除该标志） 1：采样阶段结束
0	保留	-	-	-

14.4.2 ADC 中断使能寄存器（0x04:ADC_IER)

Address offset：0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res						AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	Res	EOSIE	EOCIE	EOSMPIE	Res
-						RW	RW	RW	RW	RW	-	RW	RW	RW	-

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	AWD3IE	RW	0	模拟看门狗 3 标志中断使能。由软件设置和清除以使能/禁止模拟看门狗 3 标志中断。 0：禁止模拟看门狗 3 事件中断 1：开启模拟看门狗 3 事件中断
8	AWD2IE	RW	0	模拟看门狗 2 标志中断使能。由软件设置和清除以使能/禁止模拟看门狗 2 标志中断。 0：禁止模拟看门狗 2 事件中断 1：开启模拟看门狗 2 事件中断
7	AWD1IE	RW	0	模拟看门狗 1 标志中断使能。由软件设置和清除以使能/禁止模拟看门狗 1 标志中断。 0：禁止模拟看门狗 1 事件中断 1：开启模拟看门狗 1 事件中断
6	JEOSIE	RW	0	注入序列转换结束中断使能。由软件设置和清除以使能/禁止注入序列转换结束中断。 0：禁止 JEOS 中断 1：使能 JEOS 中断。当 JEOS 置位时，产生一个中断。
5	JEOCIE	RW	0	注入通道转换结束中断使能。该 bit 由软件设置和清除以使能/禁止注入通道转换结束中断。

				0：禁止 JEOC 中断 1：使能 JEOC 中断。当 JEOC 置位时，产生一个中断。 注意：只有当 JADSTART = 0 时，软件才被允许写入该位（这确保了没有注入转换正在进行）。
4	保留	-	-	-
3	EOSIE	RW	0	规则序列结束中断使能 该 bit 由软件设置和清除以使能/禁止规则序列结束中断 0：禁止 EOS 中断 1：使能 EOS 中断。当 EOS 置位时，产生一个中断。 注意：只有当 ADSTART = 0 时，软件才被允许写入该位（这确保了没有规则转换正在进行）。
2	EOCIE	RW	0	规则转换结束中断使能 该 bit 由软件设置和清除以使能/禁止规则转换结束中断 0：禁止 EOC 中断 1：使能 EOC 中断。当 EOC 置位时，产生一个中断。 注意：只有当 ADSTART = 0 时，软件才被允许写入该位（这确保了没有规则转换正在进行）。
1	EOSMPIE	RW	0	采样结束中断使能 该 bit 由软件设置和清除以使能/禁止采样结束中断 0：禁止 EOSMP 中断 1：使能 EOSMP 中断。当 EOSMP 置位时，产生一个中断。 注意：只有当 ADSTART = 0 时，软件才被允许写入该位（这确保了没有规则转换正在进行）。
0	保留	-	-	-

14.4.3 ADC 控制寄存器（0x08:ADC_CR)

Address offset：0x08

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res	Res	Res	RSTCAL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
RS	-			RS	-										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ADSTP	JADSTART	ADSTART	Res	ADEN
-											RS	RS	RS	-	RW

Bit	Name	R/W	Reset Value	Function
31	ADCAL	RS	0	ADC 校准启动。

				软件设置启动 ADC 校准，校正完成后硬件自动清零。在校准失败或者校准成功时由硬件清除 0：校准完成 1：写 1 校正 ADC，读为 1 表明校准正在进行 注意：仅当 ADEN=0 时，才允许软件通过设置 ADCAL 来启动校准。仅当当 ADEN 无效且无外部触发时（启用 ADC 且无转换正在进行），才允许软件通过写入 ADC_CALFACT 来更新校准因子
30:28	保留	-	-	-
27	RSTCAL	RS	0	校准复位使能位 该位由软件写 1 置 1，由硬件清除置 0。在校准寄存器被初始化后(即 RSTCAL 置 1 后)，该位即被清除。 0：校准寄存器已初始化 1：初始化校准寄存器 注：当正在进行转换时，如果设置 RSTCAL，清除校准寄存器需要额外的周期。
26：5	保留	-	-	-
4	ADSTP	RS	0	ADC 停止转换命令 该位由软件设置，用于停止并放弃正在进行的转换（ADSTP 命令）。当转换被有效地丢弃时，被硬件清除。停止注入转换后，ADC 序列和触发可以被重新配置，ADC 准备接受新的转换（外部触发） 0：无 ADC 停止转换命令执行 1：写 1 停止正在进行的转换。读取 1 表示正在执行 ADSTP 命令。
3	JADSTART	RS	0	ADC 软件触发启动注入转换 该位由软件设置，用于启动注入通道的 ADC 转换。通过软件写 1 置 1 启动转换，启动转换后立即被硬件清除置 0。 0：复位状态 1：开始转换注入通道
2	ADSTART	RS	0	ADC 软件触发启动规则转换 该位由软件设置，用于启动规则通道的 ADC 转换。 通过软件写 1 置 1 启动转换，启动转换后立即被硬件清除置 0。 0：复位状态 1：开始转换规则通道
1	保留	-	-	-
0	ADEN	RW	0	开/关 A/D 转换器 ADC 转换器工作使能，当置 1 后 ADC 转换器唤醒，在转换器上电到开始转换有一个延迟 tSTAB。当置 0 后 ADC 转换器处于掉电状态。 0：禁止 ADC 转换，ADC 转换器进入断电模式 1：使能 ADC 转换器。

14.4.4 ADC 配置寄存器（0x0C:ADC_CFGR)

Address offset：0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res	JDISCEN	DISCNUM[2:0]			DISCEN
-	RW					RW	RW	RW	RW	-	RW	RW			RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ALIGN	AUTDLY	CONT	Res	EXTEN[1:0]	EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]	SCAN	Res	DMAEN
RW	RW	RW	-	RW	RW	RW	RW	RW	RW	RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:26	AWD1CH[4:0]	RW	5'h0	模拟看门狗 1 通道选择位 该位由软件写设置，用于选择模拟看门狗的输入通道。 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 10010：ADC 模拟输入通道 18 10011：ADC 模拟输入通道 19 保留所有其他数值。 注： 模拟看门狗 1 通道选择对应的 ADC 模拟输入通道与 ADC 通道选择一致，详细对应关系详见相关章节。
25	JAUTO	RW	0	自动注入使能 该位由软件写 1 置 1 和写 0 置 0，用于开启或关闭规则通道组转换结束后自动进行注入通道组转换 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
24	JAWD1EN	RW	0	注入通道模拟看门狗 1 使能。在注入通道上开启模拟看门狗。 该位由软件写 1 置 1 和写 0 置 0。 0：在注入通道上禁用模拟看门狗 1 1：在注入通道上使用模拟看门狗 1
23	AWD1EN	RW	0	规则通道模拟看门狗 1 使能。在规则通道上开启模拟看门狗。 该位由软件写 1 置 1 和写 0 置 0。 0：在规则通道上禁用模拟看门狗 1 1：在规则通道上使用模拟看门狗 1
22	AWD1SGL	RW	0	单一通道看门狗 1 使能。 该位由软件写 1 置 1 和写 0 置 0，用于开启或关闭由 AWD1CH[4:0]定义的通道上的模拟看门狗 1。 0：在所有的通道上使用模拟看门狗 1 1：在单一通道上使用模拟看门狗 1
21	保留	-	-	-
20	JDISCEN	RW	0	注入通道间断模式使能。 该位由软件写 1 置 1 和写 0 置 0，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
19:17	DISCNUM[2:0]	RW	3'h0	间断模式通道计数 这些位是由软件写入，用于定义在接收到外部触发后，在间断模式下要转换的规则通道的数量。 000:1 通道

				001:2 通道 … 111:8 通道
16	DISCEN	RW	0	规则通道的间断模式使能 该位由软件写 1 置 1 和写 0 置 0，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
15	ALIGN	RW	0	数据对齐控制位 该位由软件写 1 置 1 和写 0 置 0 清除。 0：右对齐 1：左对齐
14	AUTDLY	RW	0	自动延迟转换模式 此位由软件设置和清除，以启用/禁用自动延迟转换模式。 0：自动延迟转换模式关闭 1：自动延迟转换模式打开
13	CONT	RW	0	连续转换使能 该位由软件写 1 置 1 和写 0 置 0 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
12	保留	-	-	-
11:10	EXTEN[1:0]	RW	2’h0	规则通道的外部触发启用和极性选择 这些位由软件设置和清除，以选择外部触发极性并启用规则组的触发。 00：禁用硬件触发检测（可以通过软件启动转换） 01：上升沿的硬件触发检测 10：下降沿的硬件触发器检测 11：上升沿和下降沿都有硬件触发检测 注意：只有确保没有规则转换正在进行，软件才允许写入这些位。
9:5	EXTSEL[4:0]	RW	5’h0	规则通道外部触发事件选择位。选择启动规则通道组转换的外部事件 ADC1,ADC2 外部触发事件如下：（详情见 4.16） 00000：事件 1 00001：事件 2 00010：事件 3 00011：事件 4 … 11111：预留 ADC3 外部触发配置如下: 00000：事件 1 00001：事件 2 00010：事件 3 00011：事件 4 … 11111：预留

4:3	RES[1:0]	RW	2'h0	分辨率（Resolution)控制位。通过软件写入这些位可选择转换的分辨率。 00：12 位 01：10 位 10：8 位 11：6 位
2	SCAN	RW	0	扫描模式使能 该位由软件写 1 置 1 和写 0 置 0，用于开启或关闭扫描模式。在扫描模式中，由 ADC_SQRx 或 ADC_JSQRx 寄存器选中的通道被转换。 0：关闭扫描模式 1：使用扫描模式
1	保留	-	-	-
0	DMAEN	RW	0	DMA 使能位 该位由软件写 1 置 1 和写 0 置 0 清除。 0：禁止 DMA 模式 1：使能 DMA 模式 注：双 ADC 模式中，只有 ADC1 能产生 DMA 请求。

14.4.5 ADC 配置寄存器 2（0x10:ADC_CFGR2)

Address offset：0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	CALNUM[2:0]			Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	GCOMP
-	RW	RW	RW	-											RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
-					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:28	CALNUM[2:0]	RW	3'h0	校准平均次数, default 000 000：1 次校准和平均 001：2 次校准和平均 010：4 次校准和平均 011：8 次校准和平均 100：16 次校准和平均 101：32 次校准和平均 Others：保留
27:17	保留	-	-	-
16	GCOMP	RW	0	增益补偿模式 此位由软件设置和清除，以启用增益补偿模式。 0：增益补偿未启用在 ADC 工作模式 1：增益补偿已启用并应用于所有通道 注意：只有当没有转换正在进行时，软件才允许写入此位

15:11	保留	-	-	-
10	ROVSM	RW	0	规则过采样模式 该位由软件设置和清除，以选择规则过采样模式。 0：继续模式：触发注入转换时，过采样暂时停止，并在注入序列后继续（注入序列期间保持过采样缓冲） 1：恢复模式：触发注射转换时，当前过采样被中止，并在注入序列后从开始恢复（过采样缓冲区在注入序列开始转换时清零） 注意：只有当没有转换正在进行时，软件才允许写入此位
9	TROVS	RW	0	触发规则过采样 此位由软件设置并清除，以启用触发过采样 0：通道的所有过采样转换在触发后连续完成 1：通道的每个过采样转换都需要一个新的触发 注意：只有当没有转换正在进行时，软件才允许写入此位
8:5	OVSS[3:0]	RW	4’h0	过采样移位 该位字段由软件设置和清除，以定义应用于原始过采样结果的右移。 0000：无移位 0001：移位 1 位 0010：移位 2 位 0011：移位 3 位 0100：移位 4 位 0101：移位 5 位 0110：移位 6 位 0111：移位 7 位 1000：移位 8 位 其他：预留 注意：只有当没有转换正在进行时，软件才允许写入此位
4：2	OVSR[2:0]	RW	3’h0	过采样率 此位字段由软件设置并清除，以定义过采样率。 000:2x 001:4x 010:8x 011:16x 100:32x 101:64x 110:128x 111:256x 注意：只有当没有转换正在进行时，软件才允许写入此位
1	JOVSE	RW	0	注入过采样使能 该位由软件设置和清除，以启用注入过采样。 0：禁用注入过采样 1：启用注入过采样

				注意：只有当没有转换正在进行时，软件才允许写入此位
0	ROVSE	RW	0	规则过采样使能 启用此位由软件设置并清除，以启用规则过采样。 0：禁用规则过采样 1：启用规则过采样 注意：只有当没有转换正在进行时，软件才允许写入此位

14.4.6 ADC 采样时间寄存器 1(0x14:ADC_SMPR1)

Address offset：0x14

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPPLUS	Res	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]	
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**
SMP5[2:0]		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

Bit	Name	R/W	Reset Value	Function
31	SMPPLUS	RW	0	采样时间添加一个时钟周期 1:2.5 ADC 时钟周期采样时间变为 3.5 ADC 时钟周期对于 ADC_SMPR1 和 ADC_SMPR2 寄存器。 0：采样时间保持设置为 2.5 ADC 时钟周期。 确保没有转换正在进行，才允许软件写入此位。
30	保留	-	-	-
29:0	SMP[9:0][2:0]	RW	30'h0	通道 x 采样时间选择 这些位由软件编写，用于为每个通道单独选择采样时间。 在采样周期期间，通道选择位必须保持不变。 000:2.5 ADC 时钟周期 001:6.5 ADC 时钟周期 010:12.5 ADC 时钟周期 011:24.5 ADC 时钟周期 100:47.5 ADC 时钟周期 101:92.5 ADC 时钟周期 110:247.5 ADC 时钟周期 111:640.5 ADC 时钟周期

14.4.7 ADC 采样时间寄存器 2(0x18:ADC_SMPR2)

Address offset：0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]	
-		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**
SMP15[2:0]		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

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29:0	SMP[19:10][2:0]	RW	30'h0	通道 x 采样时间选择 这些位由软件编写，用于为每个通道单独选择采样时间。 在采样周期期间，通道选择位必须保持不变。 000:2.5 ADC 时钟周期 001:6.5 ADC 时钟周期 010:12.5 ADC 时钟周期 011:24.5 ADC 时钟周期 100:47.5 ADC 时钟周期 101:92.5 ADC 时钟周期 110:247.5 ADC 时钟周期 111:640.5 ADC 时钟周期

14.4.8 ADC 看门狗阈值寄存器 1（0x20:ADC_TR1)

Address offset：0x20

Reset value：0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	HT1[11:0]											
-				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
Res	AWDFILT[2:0]			LT1[11:0]											
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:16	HT1[11:0]	RW	12'hFFF	模拟看门狗 1 高阈值 软件可配，定义模拟看门狗 1 高阈值 注意：只有当没有转换正在进行时，软件才允许写入此位
15	保留	-	-	-
14:12	AWDFILT[2:0]	RW	3'h0	模拟看门狗滤波参数 该位由软件设置和清除。 000：无过滤 001：连续两次检测生成 AWDx 标志或中断 … 111：八个连续检测生成 AWDx 标志或中断 注意：只有当没有转换正在进行时，软件才允许写入此位
11:0	LT1[11:0]	RW	12'h0	模拟看门狗 1 低阈值 软件可配，定义模拟看门狗 1 低阈值 注意：只有当没有转换正在进行时，软件才允许写入此位

14.4.9 ADC 看门狗阈值寄存器 2（0x24:ADC_TR2)

Address offset：0x24

Reset value：0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	HT2[7:0]							
-								RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Res	Res	Res	Res	Res	Res	LT2[7:0]							
-						RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23:16	HT2[7:0]	RW	8'hFF	模拟看门狗 2 高阈值 软件可配，定义模拟看门狗 2 高阈值 注意：只有当没有转换正在进行时，软件才允许写入此位
15:8	保留	-	-	-
7:0	LT2[7:0]	RW	8'h0	模拟看门狗 2 低阈值 软件可配，定义模拟看门狗 2 低阈值 注意：只有当没有转换正在进行时，软件才允许写入此位

14.4.10 ADC 看门狗阈值寄存器 3（0x28:ADC_TR3)

Address offset：0x28

Reset value：0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	HT3[7:0]							
-								RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res			Res	Res	Res	Res	LT3[7:0]							
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23:16	HT3[8:0]	RW	8'hFF	模拟看门狗 3 高阈值 软件可配，定义模拟看门狗 3 高阈值 注意：只有当没有转换正在进行时，软件才允许写入此位
15:8	保留	-	-	-
7:0	LT3[8:0]	RW	8'h0	模拟看门狗 3 低阈值 软件可配，定义模拟看门狗 3 低阈值 注意：只有当没有转换正在进行时，软件才允许写入此位

14.4.11 ADC 规则序列寄存器 1(0x30:ADC_SQR1)

Address offset:0x30

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			SQ4[4:0]					Res	SQ3[4:0]					Res	SQ2[4]
-			RW					-	RW					-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res	SQ1[4:0]					Res	Res	L[3:0]			
RW				-	RW					-		RW			

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28:24	SQ4[4:0]	RW	5'h0	规则序列第 4 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 4 次。

23	保留	-	-	-
22:18	SQ3[4:0]	RW	5'h0	规则序列第 3 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 3 次。
17	保留	-	-	-
16:12	SQ2[4:0]	RW	5'h0	规则序列第 2 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 2 次。
11	保留	-	-	-
10:6	SQ1[4:0]	RW	5'h0	规则序列第 1 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 1 次。
5:4	保留	-	-	-
3:0	L[3:0]	RW	4'h0	规则通道序列长度 软件配置这些位的值。这些位定义了在规则通道转换序列中通道数目。 0000：1 个转换 0001：2 个转换 …… 1111：16 个转换

某些通道未进行物理连接，因此不得选择进行转换。

14.4.12 ADC 规则序列寄存器 2(0x34:ADC_SQR2)

Address offset:0x34

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			SQ9[4:0]					Res	SQ8[4:0]					Res	SQ7[4]
-			RW					-	RW					-	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ7[3:0]				Res	SQ6[4:0]					Res	SQ5[4:0]				
RW				-	RW					-	RW				

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28:24	SQ9[4:0]	RW	5'h0	规则序列第 9 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 9 次。
23	保留	-	-	-
22:18	SQ8[4:0]	RW	5'h0	规则序列第 8 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 8 次。
17	保留	-	-	-
16:12	SQ7[4:0]	RW	5'h0	规则序列第 7 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 7 次。
11	保留	-	-	-

10:6	SQ6[4:0]	RW	5'h0	规则序列第 6 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 6 次。
5	保留	-	-	-
4:0	SQ5[4:0]	RW	5'h0	规则序列第 5 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 5 次。

某些通道未进行物理连接，因此不得选择进行转换。

14.4.13 ADC 规则序列寄存器 3(0x38:ADC_SQR3)

Address offset:0x38

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			SQ14[4:0]					Res	SQ13[4:0]					Res	SQ12[4]
-			RW					-	RW					-	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ12[3:0]				Res	SQ11[4:0]					Res	SQ10[4:0]				
RW				-	RW					-	RW				

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28:24	SQ14[4:0]	RW	5'h0	规则序列第 14 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 14 次。
23	保留	-	-	-
22:18	SQ13[4:0]	RW	5'h0	规则序列第 13 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 13 次。
17	保留	-	-	-
16:12	SQ12[4:0]	RW	5'h0	规则序列第 12 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 12 次。
11	保留	-	-	-
10:6	SQ11[4:0]	RW	5'h0	规则序列第 11 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 11 次。
5	保留	-	-	-
4:0	SQ10[4:0]	RW	5'h0	规则序列第 10 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 10 次。

某些通道未进行物理连接，因此不得选择进行转换。

14.4.14 ADC 规则序列寄存器 4(0x3C:ADC_SQR4)

Address offset:0x3C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					SQ16[4:0]					Res	SQ15[4:0]				
-					RW					-	RW				

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	-
10:6	SQ16[4:0]	RW	5'h0	规则序列第 16 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 16 次。
5	保留	-	-	-
4:0	SQ15[4:0]	RW	5'h0	规则序列第 15 次转换 这些位由软件写入,通道（0 至 19）被分配为规则转换序列中的第 15 次。

某些通道未进行物理连接，因此不得选择进行转换。

14.4.15 ADC 规则数据寄存器（0x40:ADC_DR)

Address offset：0x40

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[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
RDATA[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31：16	ADC2DATA[15:0]	R	16'h0	ADC2 转换的数据 软件可读这些位的值。 - 在 ADC1 中：双模式下，这些位包含了 ADC2 转换的规则通道数据。 - 在 ADC2 中：保留位。
15:0	RDATA[15:0]	R	16'h0	规则通道转换结果 软件可读这些位的值。这些位为只读，包含了规则通道的转换结果。数据是左或右对齐

14.4.16 ADC 注入序列寄存器(0x4C:ADC_JSQR)

Address offset:0x4C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res	JSQ3[4:0]					Res	JSQ2[4:1]			
RW					-	RW					-	RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
RW	-	RW					RW		RW					RW	

Bit	Name	R/W	Reset Value	Function
31:27	JSQ4[4:0]	RW	5'h0	注入序列第 4 次转换 这些位由软件写入,通道（0 至 19）被分配为注入转换序列中的第 4 次。软件配置这些位的值。这些位定义了转换序列中的第 4 个转换通道的编号(0~19)。 注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。
26	保留	-	-	-
25:21	JSQ3[4:0]	RW	5'h0	注入序列第 3 次转换 这些位由软件写入,通道（0 至 19）被分配为注入转换序列中的第 3 次。
20	保留	-	-	-
19:15	JSQ2[4:0]	RW	5'h0	注入序列第 2 次转换 这些位由软件写入,通道（0 至 19）被分配为注入转换序列中的第 2 次。
14	保留	-	-	-
13:9	JSQ1[4:0]	RW	5'h0	注入序列第 1 次转换 这些位由软件写入,通道（0 至 19）被分配为注入转换序列中的第 1 次。
8:7	JEXTEN[1:0]	RW	2'h0	注入通道的外部触发启用和极性选择 这些位由软件设置和清除，以选择外部触发极性并启用注入组的触发。 00:禁用硬件触发检测（可以通过软件启动转换） 01:上升沿的硬件触发检测 10:下降沿的软件触发检测 11:上升沿和下降沿都有硬件触发检测 注意：只有当没有注入的转换正在进行时，软件才允许写入这些位。
6:2	JEXTSEL[4:0]	RW	5'h0	注入通道组外部触发事件选择位 ADC1 和 ADC2 的外部触发事件如下：（详情见 4.16） 00000：事件 1 00001：事件 2 00010：事件 3 00011：事件 4 … 11111：预留 ADC3 的触发设置如下: 00000：事件 1 00001：事件 2 00010：事件 3 00011：事件 4 … 11111：预留
1:0	JL[1:0]	RW	2'h0	注入通道序列长度 这些位由软件编写，用于定义注入通道转换序列中的转换总数。 00：1 个转换 01：2 个转换 10：3 个转换

				11：4 个转换

某些通道未进行物理连接，因此不得选择进行转换。

14.4.17 ADC offset y 寄存器(0x60+0x04(y-1):ADC_OFRy)

Address offset：0x60+0x04(y-1)，(y=1 to 4)

Reset value:0x0000_0000

(SATEN、OFFSETPOS 只有 ADC_OFR1 有，并且 ADC_OFR1-4 共用)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSETy_EN	OFFSETy_CH[4:0]					SATEN	OFFSETPOS	Res							
RW	RW					RW	RW	-							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res				OFFSETy[11:0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31	OFFSETy_EN	RW	0	OFFSET y 使能 该位由软件写入，以启用或禁用 offset 值 OFFSETy[11:0]。 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。
30:26	OFF-SETy_CH[4:0]	RW	5'h0	数据偏移 y 的通道选择 这些位由软件写入，以定义 OFFSETy[11:0]的偏移将应用的通道。 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。 某些通道没有物理连接，因此不能为数据偏移量 y 选择。
25	SATEN	RW	0	饱和使能 此位由软件设置和清除，以使能偏移在 0x000 和 0xFFF 饱和。 0：无饱和控制，偏移结果可以有符号 1：启用饱和，偏移结果无符号且在 0x000 和 0xFFF 处饱和 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。
24	OFFSETPOS	RW	0	正偏移 此位由软件设置和清除，以使能正偏移。 0：负偏移 1：正偏移 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。
23:12	保留	-	-	-
11:0	OFFSETy[11:0]	RW	12'h0	OFFSETy_CH[4:0]通道的数据偏移 y。 这些位由软件写入，以定义在转换通道（可以是规则的或注入的）时从原始转换数据中减去的偏移 y。应用数据偏移 y 的通道必须在位 OFFSETy_CH[4:0]中编程。转换结果可以从 ADC_DR（规则转换）或 ADC_JDRyi 寄存器（注入转换）中读取。 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。 如果多个偏移（OFFSETy）指向同一通道，则仅考虑具有最低 x 值的偏移进行减法。 例如：如果 OFFSET1_CH[4:0]=4 且 OFFSET2_CH[4:0]=4，则这是在转换通道 4 时减去的 OFFSET1[11:0]。

14.4.18 ADC 注入数据寄存器（0x80+0x04(y-1):ADC_JDRy)

Address offset：0x80+0x04(y-1)，（y=1 to 4）

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15:0	JDATA[15:0]	R	16’h0	注入转换数据 这些 bit 只读。注入通道 y 的转换结果放于此寄存器。数据是左对齐或者右对齐的。

14.4.19 ADC 模拟看门狗 2 配置寄存器（0xA0:ADC_AWD2CR)

Address offset：0xA0

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	AWD2CH[19:16]			
-												RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：20	保留	-	-	-
19:0	AWD2CH[19:0]	RW	20’h0	模拟看门狗 2 通道选择 这些位由软件设置和清除。它们启用并选择要由模拟看门狗 2 保护的输入通道。 AWD2CH[i]=0：AWD2 未监控 ADC 模拟输入通道 i AWD2CH[i]=1：AWD2 监控 ADC 模拟输出通道 i 当 AWD2CH[19:0]=000..0 时，模拟看门狗 2 被禁用 注意：AWD2CH 选择的通道也必须选择到 SQRi 或 JSQRi 寄存器中。 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。 某些通道未进行物理连接，不得为模拟看门狗选择。

14.4.20 ADC 模拟看门狗 3 配置寄存器（0xA4:ADC_AWD3CR)

Address offset：0xA4

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	AWD3CH[19:16]			
-												RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															

RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：20	保留	-	-	-
19:0	AWD3CH[19:0]	RW	20'h0	模拟看门狗 3 通道选择 这些位由软件设置和清除。它们启用并选择要由模拟看门狗 3 保护的输入通道。 AWD3CH[i]=0：AWD3 未监控 ADC 模拟输入通道 i AWD3CH[i]=1：AWD3 监控 ADC 模拟输出通道 i 当 AWD3CH[19:0]=000..0 时，模拟看门狗 3 被禁用 注意：AWD3CH 选择的通道也必须选择到 SQRi 或 JSQRi 寄存器中。 注意：只有当确保没有正在进行的转换时，软件才允许写入这些位。 某些通道未进行物理连接，不得为模拟看门狗选择。

14.4.21 ADC 差分模式选择寄存器（0xB0:ADC_DIFSEL)

Address offset：0xB0

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：20	保留	-	-	-
19:0	DIFSEL[19:0]	RW	20'h0	通道 19 至 0 的差分模式。 这些位由软件设置和清除。它们允许选择通道配置为单端模式还是差分模式。 DIFSEL[i]=0：模拟输入通道 i 配置为单端模式 DIFSEL[i]=1：模拟输入通道 i 配置为差分模式 注意：DIFSEL 对应通道或连接到单端 i/O 端口或连接到内部通道必须保持复位值（单端输入模式）。 只有当 ADC 被禁用时，软件才允许写入这些位。

14.4.22 ADC 校准因子寄存器（0xB4:ADC_CALFACT)

Address offset：0xB4

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res									CALON	CAPSUC	OFFSUC	Res		Res	Res
-									R	RC_W1	RC_W1	-			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res					Res								
-															

Bit	Name	R/W	Reset Value	Function
31：23	保留	-	-	-
22	CALON	R	0	校准状态位。 1：ADC 正在进行校准； 0：ADC 校准已结束或未启动。
21	CAPSUC	RC_W1	0	电容校准状态位。 表示 ADC 电容校准是否成功。硬件置 1；软件写 1 置 0；
20	OFFSUC	RC_W1	0	Offset 校准状态位。 表示 ADC offset 校准是否成功。硬件置 1；软件写 1 置 0；
19:0	保留	-	-	-

14.4.23 ADC 增益补偿寄存器（0xC0:ADC_GCOMP)

Address offset：0xC0

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:0	GCOMPCOEFF[13:0]	RW	14’h0	增益补偿系数 这些位由软件设置和清除，以编程增益补偿系数。 00 1000 0000 0000：增益因子为 0.5 … 01 0000 0000 0000：增益因子为 1 10 0000 0000 0000：增益因子为 2 11 0000 0000 0000：增益因子为 3 … 该系数除以 4096，得到范围从 0 到 3.999756 的增益因子。 注意：此增益补偿仅在 ADC_CFGR2 寄存器的 GCOMP 位为 1 时应用。

14.5 寄存器描述(对于公共 ADC)

14.5.1 ADC 公共控制寄存器（0x308:ADC_CCR)

Address offset：0x308(只有 ADC1 有该寄存器)

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	VREFSEL	PWRMODE			Res	VSENSESEL	VREFEN	Res					
-			RW	RW			-	RW	RW	-					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res								VREFBUFSEL[1:0]		VREFBUFEN	DUAL[4:0]				
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28	VREFSEL	RW	0	ADC 参考电压选择, 0：V_{REF+} 作为参考电压（V_{REF+} 连到 V_{CCA}） 1：V_{REFBUF} 作为参考电压
27:25	PWRMODE	RW	3'h0	ADC 电流模式，用于 ADC 模拟电路比较器 000：6uA（default) 001：7uA 010：8uA 011：9uA 100：2uA 101：3uA 110：4uA 111：5uA
24	保留	-	-	-
23	VSENSESEL	RW	0	V_{SENSE} 选择 此位由软件控制 V_{SENSE} 的设置和禁用。 0:温度传感器通道禁用 1:温度传感器通道启用
22	VREFEN	RW	0	V_{REFINT} 使能 该位由软件设置和清除，以启用/禁用 V_{REFINT} 通道。 0:V_{REFINT} 通道禁用 1:V_{REFINT} 通道启用
21:8	保留	-	-	-
7:6	VREFBUFSEL	RW	2'h0	V_{REFBUF} 输出电压选择 00：1.08 V 01：2.048 V 10：2.5 V 11：2.9 V 注：（1.08V V_{REFBUF} 精准值存放地址：0x1FFF 7E28 2.048V V_{REFBUF} 精准值存放地址：0x1FFF 7E30 2.5V V_{REFBUF} 精准值存放地址：0x1FFF 7E34 2.9V V_{REFBUF} 精准值存放地址：0x1FFF 7E38 例如：从地址 0x1FFF 7E28 读到值是 0x1079，表示 V_{REFBUF} 精准值 1.079 V)
5	VREFBUFEN	RW	0	V_{REFBUF} 使能，当 VREFBUFEN==0 时，电压输出高阻 软件写 0 置 0，写 1 置 1， 0：禁用 V_{REFBUF} 1：使能 V_{REFBUF}
4:0	DUAL[4:0]	RW	5'h0	双 ADC 模式选择 这些位由软件写入以选择双 ADC 模式。 所有 ADC 独立： 00000：独立模式 00001 至 01001：双 ADC 模式，主从 ADC 协同工作 00001：组合的同步规则+同步注入模式

				00010：预留 00011：预留 00100：预留 00101：仅同步注入模式 00110：仅同步规则模式 00111：快速交叉模式 01000：慢速交叉模式 01001：仅交替触发模式 所有其他组合是保留的，不得编程 注意：只有当 ADC 被禁用时，软件才允许写入这些位。

15. 数字/模拟转换（DAC）

15.1 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 个输出通道，每个通道都有单独的转换器。在双 DAC 模式下，2 个通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。

15.1.1 主要特征

- 2 个 DAC 转换器：每个转换器对应 1 个输出通道
- 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
- 每个通道都有 DMA 功能
- 支持 DMA 下溢错误检测
- 外部触发转换
- 输入参考电压 V_{REF+}

15.1.2 模块框图

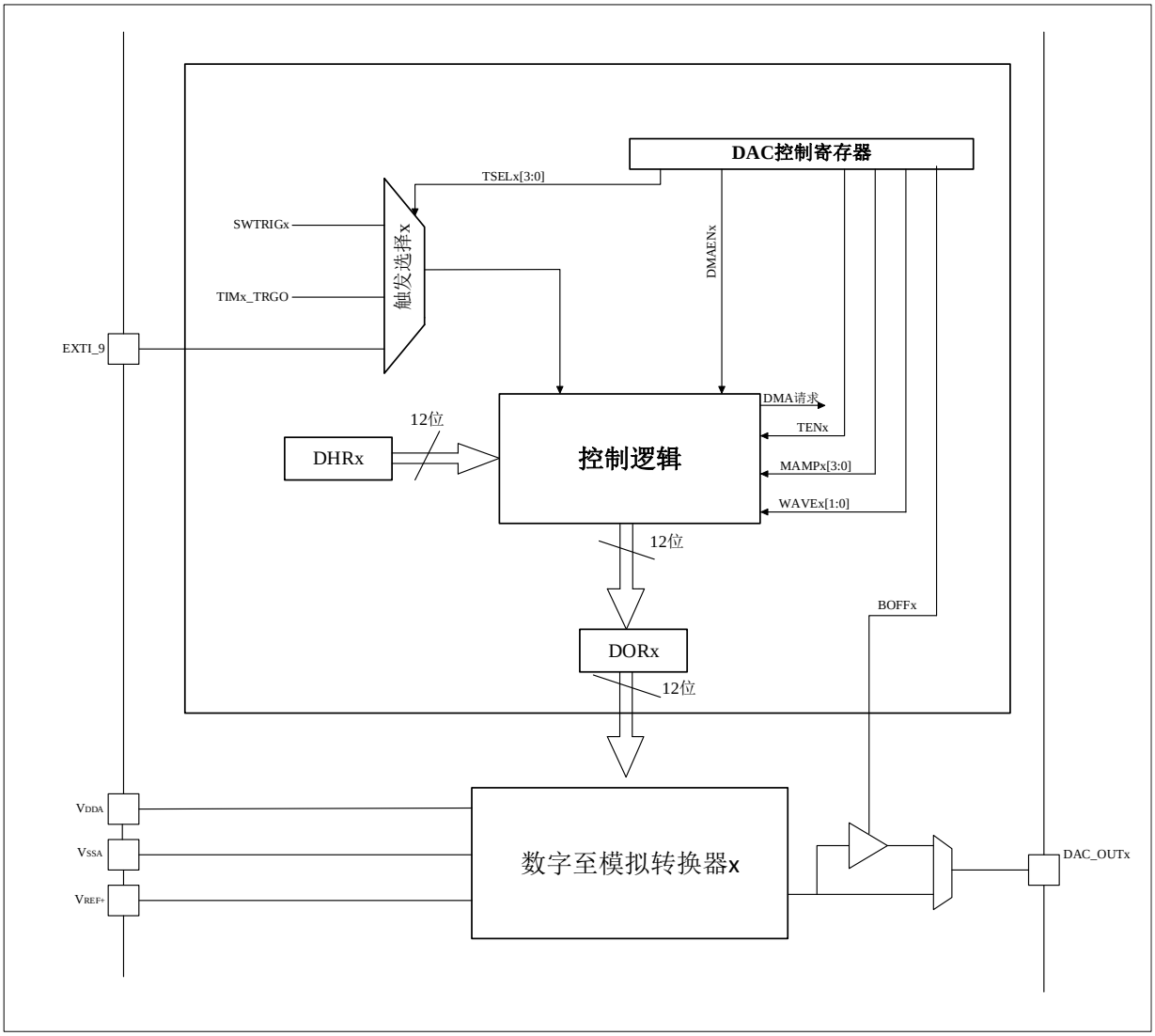


图 15-1 DAC 模块框图

名称	型号类型	描述
V_{CCA}	输入，模拟电源	模拟电源
V_{SSA}	输入，模拟电源地	模拟电源地
V_{REF+}	输入，模拟正参考电压	DAC 正参考电压， 1.6 V ≤ V_{REF+} ≤ V_{CCA}
DAC_OUTx	模拟输出信号	DACx 模拟输出

注意：在使能 DAC 模块前，GPIO 口(PA4 对应 DAC1，PA5 对应 DAC2)应配置位模拟模式。

15.2 DAC 功能描述

15.2.1 使能 DAC 通道

将 DAC_CR 寄存器的 ENx 位置‘1’即可打开对 DAC 通道 x 的供电。经过一段启动时间(t_{WAKEUP})，DAC 通道 x 即被使能。

注意：ENx 位只会使能 DAC 通道 x 的模拟部分，即便该位被置‘0’，DAC 通道 x 的数字部分仍然工作。

15.2.2 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_CR 寄存器的 BOFFx 位来使能或者关闭。

15.2.3 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_DHR8Rx[7:0]位(实际是存入寄存器 DAC_DHRx[11:4]位)
- 12 位数据左对齐：用户须将数据写入寄存器 DAC_DHR12Lx[15:4]位(实际是存入寄存器 DAC_DHRx[11:0]位)
- 12 位数据右对齐：用户须将数据写入寄存器 DAC_DHR12Rx[11:0]位(实际是存入寄存器 DAC_DHRx[11:0]位)

根据对 DAC_DHRyyyx 寄存器的操作(yyy 指不同的数据格式和对齐方式)，经过相应的移位后，写入的数据被转存到 DAC_DHRx 寄存器中(DHRx 是内部的数据保存寄存器 x)。随后，DAC_DHRx 寄存器的内容通过软件触发或外部触发自动地传送到 DAC_DORx 寄存器。

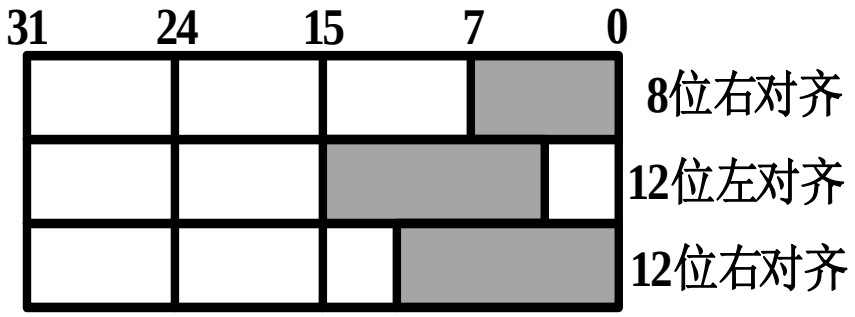


图 15-2 单 DAC 通道模式的数据寄存器

双 DAC 通道，有 3 种情况：

- 8 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR8RD[7:0]位（实际是存入寄存器 DAC_DHR1[11:4]位），将 DAC 通道 2 数据写入寄存器 DAC_DHR8RD[15:8]位（实际是存入寄存器 DAC_DHR2[11:4]位）
- 12 位数据左对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12LD[15:4]位（实际是存入寄存器 DAC_DHR1[11:0]位），将 DAC 通道 2 数据写入寄存器 DAC_DHR12LD[31:20]位（实际是存

入寄存器 DAC_DHR2[11:0]位)

- 12 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12RD[11:0]位(实际是存入寄存器 DAC_DHR1[11:0]位)，将 DAC 通道 2 数据写入寄存器 DAC_DHR12RD[27:16]位(实际是存入寄存器 DAC_DHR2[11:0]位)

根据对 DAC_DHRyyyD(yyy 指不同的数据格式和对齐方式)寄存器的操作，经过相应的移位后，写入的数据被转存到 DAC_DHR1 和 DAC_DHR2 寄存器中(DAC_DHR1 和 DAC_DHR2 是内部的数据保存寄存器)。随后，DAC_DHR1 和 DAC_DHR2 的内容通过软件触发或外部事件触发自动地传送到 DAC_DORx 寄存器。

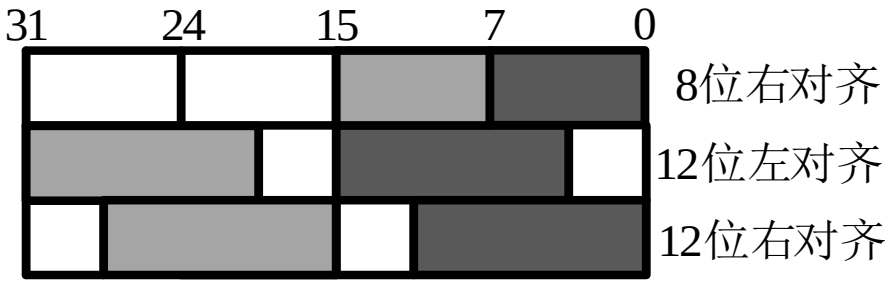


图 15-3 双 DAC 通道模式的数据寄存器

15.2.4 DAC 转换

不能直接对寄存器 DAC_DORx 写入数据，任何输出到 DAC 通道 x 的数据都必须写入 DAC_DHRx 寄存器(数据实际写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12LD、或者 DAC_DHR12RD 寄存器)。

如果没有选中硬件触发(寄存器 DAC_CR 的 TENx 位置'0')，存入寄存器 DAC_DHRx 的数据会在 1 个 APB1 时钟周期后自动传到寄存器 DAC_DORx 中。如果选中硬件触发(寄存器 DAC_CR 的 TENx 位置'1')，数据传输在触发发生以后 3 个 APB1 时钟周期完成。

一旦数据从 DAC_DHRx 寄存器装入 DAC_DORx 寄存器，在经过建立时间($t_{SETTLING}$)之后，输出即有效，这段时间的长短随着电源电压和模拟输出负载的不同会有所变化。

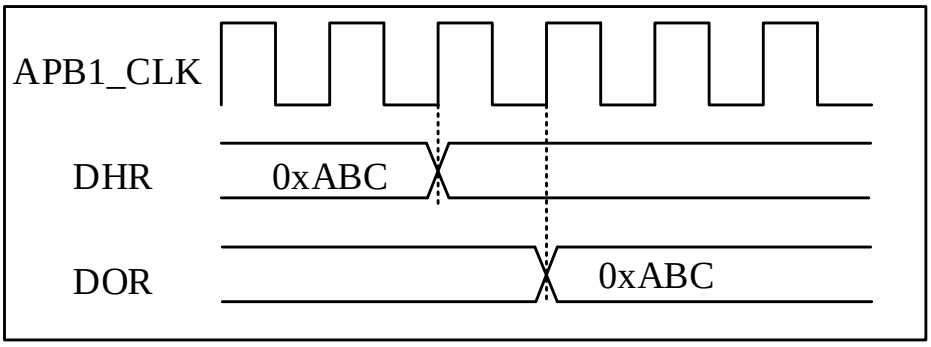


图 15-4 TEN=0 时转换时序图

15.2.5 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 通道引脚上的输出电压满足下面的关系：

DAC 输出 = V_{REF+} x（DOR / 4096)。

15.2.6 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(定时计数器、外部中断线和软件操作)。配置控制位 TSELx[3:0]可以选择 12 个触发事件之一触发 DAC 转换。

表 15-1 外部触发

TSELx[3:0]	DAC 触发信号选择	
	DAC1	DAC2
0	sw	sw
1	tim8_trgo	tim8_trgo
2	tim7_trgo	tim7_trgo
3	tim15_trgo	tim15_trgo
4	tim2_trgo	tim2_trgo
5	tim4_trgo	tim4_trgo
6	exti9	exti9
7	tim6_trgo	tim6_trgo
8	tim3_trgo	tim3_trgo
9	tim18_trgo	tim18_trgo
10	tim1_trgo	tim1_trgo
11	tim5_trgo	tim5_trgo

每次 DAC 侦测到来自选中的定时器 TRGO 或者外部中断线 9（事件）的上升沿输出时，最近存放在寄存器 DAC_DHRx 中的数据会被传送到寄存器 DAC_DORx 中。在 3 个 APB1 时钟周期后，寄存器 DAC_DORx 更新为新值。

如果选择软件触发，一旦 SWTRIG 位置‘1’，转换即开始。在数据从 DAC_DHRx 寄存器传送到 DAC_DORx 寄存器后，SWTRIG 位由硬件自动清‘0’。

注意：

1、不能再 ENx 为‘1’时改变 TSELx[3:0]位。

2、如果选择软件触发，数据从寄存器 DAC_DHRx 传送到寄存器 DAC_DORx 只需要 1 个 APB1 时钟周期。

3、使能输出缓存时(DAC_CR.BOFFx=0)，触发的频率不能超过 1 M；关闭输出缓存时(DAC_CR.BOFFx=1)，触发的频率不能超过 15 M。

15.2.7 DMA 功能

15.2.7.1 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 DAC_CR.DMAENx 位置‘1’，一旦有外部触发(而不是软件触发)发生，则会在 3 个时钟周期后产生一个 DMA 请求，然后 DAC_DHRx 寄存器的数据被传送到 DAC_DORx 寄存器中。

在双 DAC 模式下(即使用 DAC_DHR12RD 或 DAC_DHR12LD 或 DAC_DHR8RD 寄存器)，DAC_CR.DMAENx 中仅一位应被置位。

15.2.7.2 DMA 下溢检测

DAC 的 DMA 请求没有缓冲队列，因此如果第二个外部触发在接收到对第一个外部触发的应答(第一个请求)之前到达，则不会发出新的 DMA 请求，并将 DAC_SR 寄存器中的 DMA 下溢标志 DMAUDRx 将置“1”，报告错误。然后禁用 DMA 数据传输，不再处理任何 DMA 请求。DAC 通道继续转换原数据。软件应通过对寄存器 DMAUDRx 写入“1”来清除标志，将所用 DMA 通道的 DMAEN 位置零，并重新初始化 DMA 和 DAC 通道，以便正确地重新开始 DMA 传输。同时软件应修改 DAC 触发转换频率来减轻 DMA 工作负载，以避免再次发生 DMA 下溢情况。

对于各 DAC 通道，如果使能 DAC_CR 寄存器中相应的 DMAUDRIEx 位，还将产生相应的中断。

15.2.8 噪声生成

可以利用线性反馈移位寄存器(Linear Feedback Shift Register LFSR)产生幅度变化的伪噪声。设置 DAC_CR.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。

设置 DAC_CR.MAMPx[3:0]位可以屏蔽部分或者全部 LFSR 的数据，这样得到的 LFSR 的值与 DAC_DHRx 的数值相加，如果得到的数值有溢出，则将寄存器所能保留的最大值写入 DAC_DORx 中，如果得到的数值没有溢出，则将该值写入 DAC_DORx 中。

如果寄存器 LFSR 值为 0x000，则会注入‘1’(防锁定机制)。

将 DAC_CR.WAVEx[1:0]位置‘0’可以复位 LFSR 波形的生成算法。

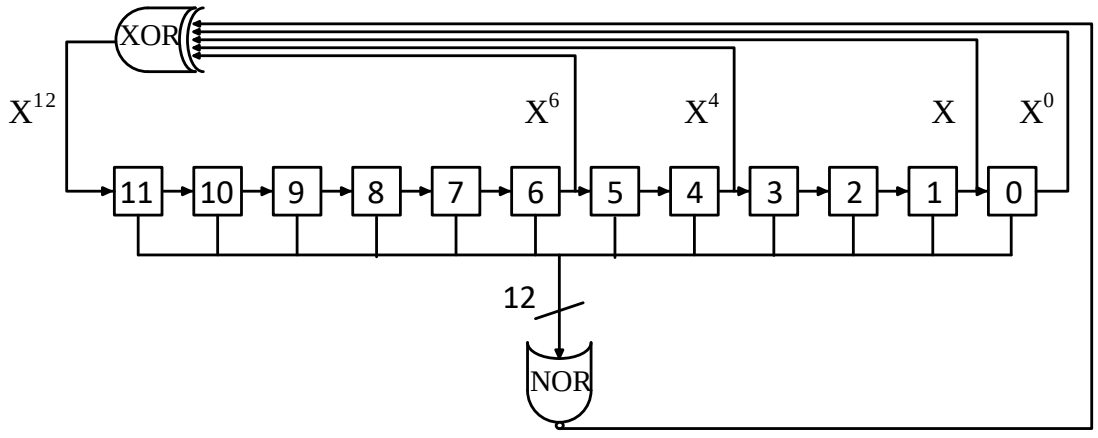


图 15-5 DAC LFSR 寄存器算法

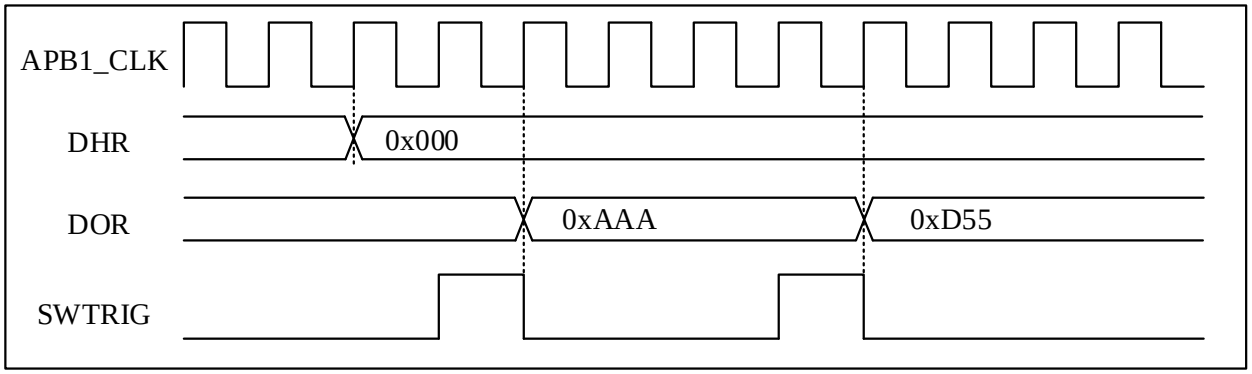


图 15-6 带 LFSR 波形生成的 DAC 转换(软件触发)

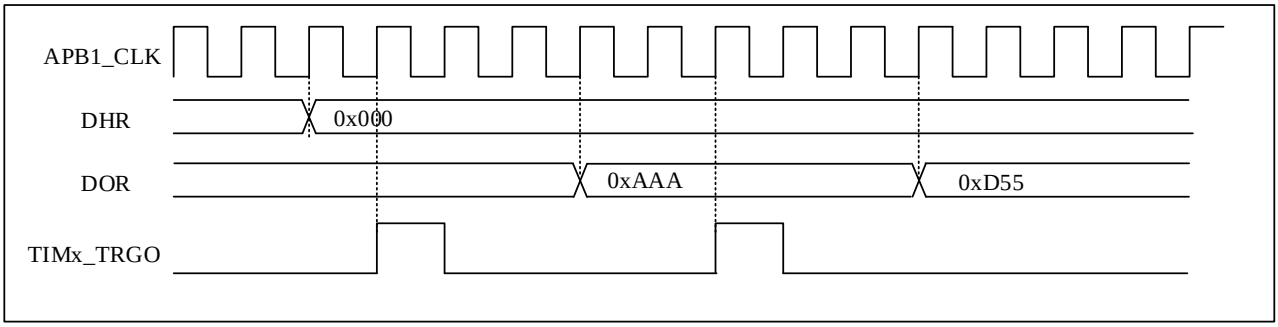


图 15-7 带 LFSR 波形生成的 DAC 转换(硬件触发)

注意：

1. 为了产生噪声，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为‘1’；
2. 若 DAC_DHR 的值发生了改变，则 LFSR 寄存器的值会复位为 0xAAA；
3. 若 DAC_CR.TENx 从“1”变为“0”，则 LFSR 寄存器的值也会复位为 0xAAA。

15.2.9 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 DAC_CR.WAVEx[1:0]位为“10”或“11”选择 DAC 的三角波生成功能。设置 DAC_CR.MAMPx[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 时钟周期累加 1。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 DAC_CR.WAVEx[1:0]位置‘0’可以复位三角波的生成。

注意：当基值为寄存器 DAC_DORx 的最大值时，不论 MAMPx 为何值，DAC_DORx 的输出均为其最大值。

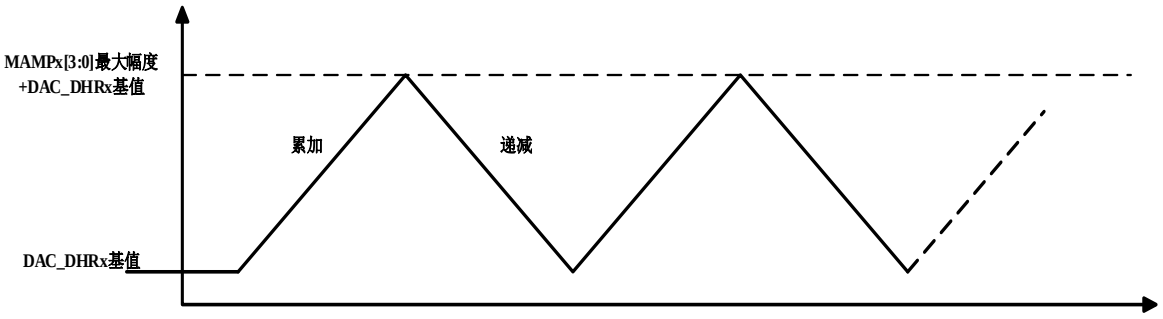


图 15-8 DAC 三角波生成

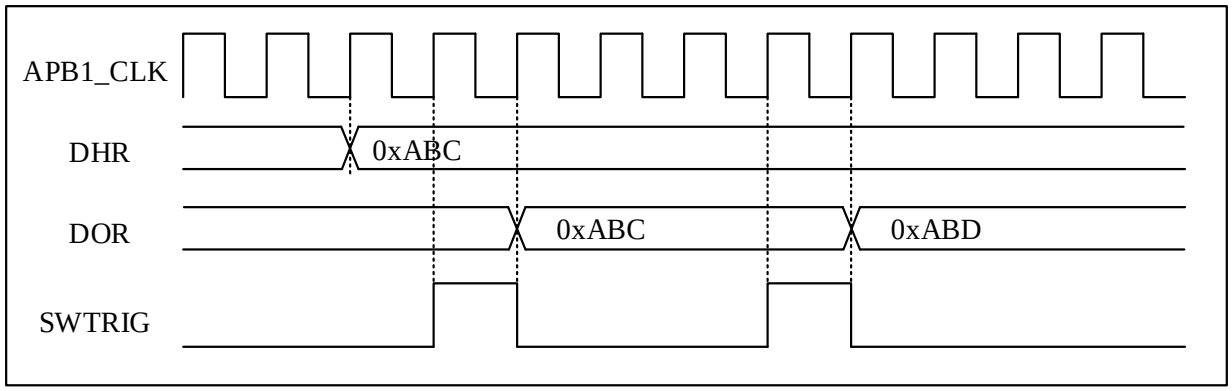


图 15-9 带三角生成的 DAC 转换(软件触发)

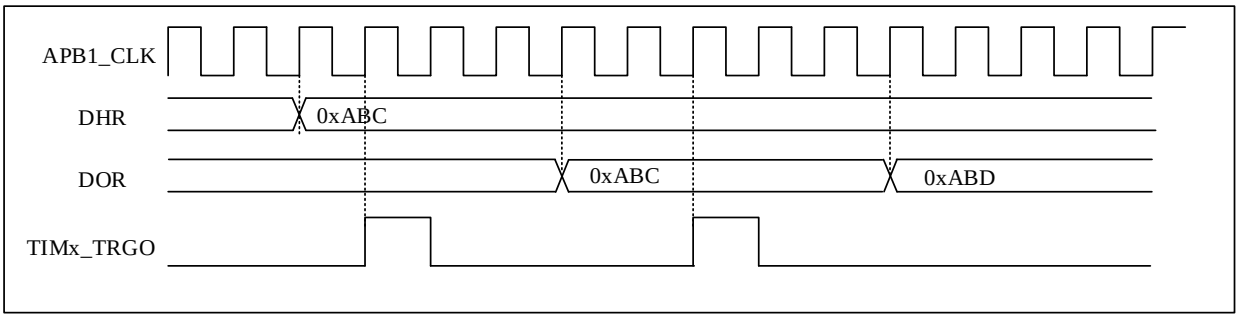


图 15-10 带三角生成的 DAC 转换(硬件触发)

注意：

1. 为了产生三角波，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为‘1’；
2. DAC_CR.MAMP[3:0]位必须在使能 DAC 之前设置，否则其值不能修改；
3.改变 DAC_DHR 的值，三角波计数器复位为 0；
4.若 DAC_CR.TENx 从“1”变为“0”，三角波计数器也复位为 0。

15.2.10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：DHR8RD、DHR12RD 和 DHR12LD，只需要访问一个寄存器即可同时完成驱动 2 个 DAC 通道的操作。

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 DAC_DHRx 寄存器操作。

所有模式详述于以下章节。

15.2.10.1 不使用波形发生器的独立触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

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 DHR1 的值传入寄存器 DAC_DOR1 中。

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 DHR2 的值传入寄存器 DAC_DOR2 中。

15.2.10.2 使用相同 LFSR 的独立触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“01”，并设置 DAC_CR.MAMPx[3:0]为相同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

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 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，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 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 LFSR2 计数器。

15.2.10.3 使用不同 LFSR 的独立触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“01”，并设 DAC_CR.MAMPx[3:0]为不同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或者 DHR8RD)。

当发生 DAC 通道 1 触发事件时，按照 DAC_CR.MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，然后更新 LFSR1 计数器。

当发生 DAC 通道 2 触发事件时，按照 DAC_CR.MAMP2[3:0]所设屏蔽的 LFSR2 计数器值与 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 LFSR2 计数器。

15.2.10.4 产生相同三角波的独立触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“1x”，并设 DAC_CR.MAMPx[3:0]为相同的三角波幅值；

- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，相同的三角波幅值加上 DAC_DHR1 寄存器值，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，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 DAC_DHR2 寄存器值，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 DAC 通道 2 三角波计数器。

15.2.10.5 产生不同三角波的独立触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“1x”，并设 DAC_CR.MAMPx[3:0]为不同的三角波幅值。
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，DAC_CR.MAMP1[3:0]所设的三角波幅值加上 DAC_DHR1 寄存器值，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，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，DAC_CR.MAMP2[3:0]所设的三角波幅值加上 DAC_DHR2 寄存器值，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 DAC 通道 2 三角波计数器。

15.2.10.6 同时非硬件转换

按照下列过程设置 DAC 转换模式：

- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

在此配置下，一个 APB1 时钟周期后，DAC_DHR1 和 DAC_DHR2 寄存器的数值即被分别传入 DAC_DOR1 和 DAC_DOR2 寄存器中。

15.2.10.7 不使用波形发生器的同时触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，延迟 3 个 APB1 时钟周期后，DAC_DHR1 和 DAC_DHR2 寄存器的数值分别传入 DAC_DOR1 和 DAC_DOR2 寄存器。

15.2.10.8 使用相同 LFSR 的同时触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；

- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“01”，并设 DAC_CR.MAMPx[3:0]为相同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)；

当发生触发事件时，DAC_CR.MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入 DAC_DOR1 寄存器中，然后更新 LFSR1 计数器。

同样，DAC_CR.MAMP1[3:0]所设屏蔽的 LFSR2 计数器值与 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 LFSR2 计数器。

15.2.10.9 使用不同 LFSR 的同时触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’；
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“01”，并设 DAC_CR.MAMPx[3:0]为不同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，然后更新 LFSR1 计数器。

同时，具有相同屏蔽的 LFSR2 计数器值与 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新 LFSR2 计数器。

15.2.10.10 使用相同三角波发生器的同时触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’：
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“1x”，并设 DAC_CR.MAMPx[3:0]为相同的三角波幅值。
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，相同的三角波幅值与 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，然后更新三角波计数器。

同时，相同的三角波幅值与 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新三角波计数器。

15.2.10.11 使用不同三角波发生器的同时触发

按照下列顺序设置 DAC 转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC_CR.TEN1 和 DAC_CR.TEN2 为‘1’：
- 通过设置 DAC_CR.TSEL1[3:0]和 DAC_CR.TSEL2[3:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
- 设置 2 个 DAC 通道的 DAC_CR.WAVEx[1:0]位为“1x”，并设 DAC_CR.MAMPx[3:0]为不同的三角波

幅值。

- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，DAC_CR.MAMP1[3:0]所设的三角波幅值与 DAC_DHR1 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR1 中，然后更新三角波计数器。

同时，DAC_CR.MAMP2[3:0]所设的三角波幅值与 DAC_DHR2 寄存器值相加，延迟 3 个 APB1 时钟周期后，结果传入寄存器 DAC_DOR2 中，然后更新三角波计数器。

15.2.11 DAC 输出时钟

DAC 输出时钟(DAC_CLK)为模拟端采样数据时钟，该时钟只有在 DAC_CR.ENx 为“1”时有效。在选择硬件触发时，检测到硬件触发 TRGO 上升沿后的 5 个 APB1 周期置“1”，在保持 3 个 APB1 周期后清 0；在选择软件触发时，检测到软件触发上升沿后的 3 个 APB1 周期置“1”，在保持 3 个 APB1 周期后清 0；在选择不触发时，检测到数据变化的 2 个 APB1 周期置“1”，在保持 3 个 APB1 周期后清 0。

注意：当 TENx 从“1”变为“0”的 1 个 APB1 周期后，DAC_DOR 的数据会更新为 DAC_DHR 寄存器中的数据。当 DAC_DOR 更新为新值时，会产生一个 DAC_CLK；反之，则不会产生 DAC_CLK。

15.3 寄存器描述

必须以字(32 位)的方式操作这些外设寄存器。

15.3.1 DAC 控制寄存器(DAC_CR)

Address offset：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3:0]				TEN2	BOFF2	EN2
	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**
Res	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:0]				TEN1	BOFF1	EN1
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30	DMAUDRIE2	RW	0	**DMAUDRIE2：**DAC 通道 2 DMA 下溢中断使能 0：屏蔽 DAC 通道 2 DMA 下溢中断； 1：使能 DAC 通道 2 DMA 下溢中断。
29	DMAEN2	RW	0	**DMAEN2**：DAC 通道 2 DMA 使能(DAC channel2 DMA enable) 0：屏蔽 DAC 通道 2 DMA 模式； 1：使能 DAC 通道 2 DMA 模式。
28：25	MAMP2	RW	4’h0	**MAMP2[3：0]：**DAC 通道 2 屏蔽/增幅选择器(DAC channel2 mask/amplitude selector) 0000：不屏蔽 LSFR 位 0/三角波增幅值等于 1； 0001：不屏蔽 LSFR 位[1：0]/三角波增幅值等于 3； 0010：不屏蔽 LSFR 位[2：0]/三角波增幅值等于 7； 0011：不屏蔽 LSFR 位[3：0]/三角波增幅值等于 15；

				0100：不屏蔽 LSFR 位[4：0]/三角波增幅值等于 31； 0101：不屏蔽 LSFR 位[5：0]/三角波增幅值等于 63； 0110：不屏蔽 LSFR 位[6：0]/三角波增幅值等于 127； 0111：不屏蔽 LSFR 位[7：0]/三角波增幅值等于 255； 1000：不屏蔽 LSFR 位[8：0]/三角波增幅值等于 511； 1001：不屏蔽 LSFR 位[9：0]/三角波增幅值等于 1023； 1010：不屏蔽 LSFR 位[10：0]/三角波增幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11：0]/三角波增幅值等于 4095；
24：23	WAVE2	RW	2’h0	**WAVE2[1：0]：**DAC 通道 2 噪声/三角波生成使能(DAC channel2 noise/triangle wave generation enable) 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。
22：19	TSEL2	RW	4’h0	**TSEL2[3：0]：**DAC 通道 2 出发选择(DAC channel2 trigger selection) 0000：sw 0001：tim8_trgo 0010：tim7_trgo 0011：tim15_trgo 0100：tim2_trgo 0101：tim4_trgo 0110：exti9 0111：tim6_trgo 1000：tim3_trgo 1001：tim18_trgo 1010：tim1_trgo 1011：tim5_trgo 其他：保留
18	TEN2	RW	0	**TEN2**：DAC 通道 2 触发使能(DAC channel2 trigger enable) 0：关闭 DAC 通道 2 触发 1：使能 DAC 通道 2 触发
17	BOFF2	RW	0	**BOFF2：**关闭 DAC 通道 2 输出缓存(DAC channel2 output buffer disable） 0：使能 DAC 通道 2 输出缓存； 1：关闭 DAC 通道 2 输出缓存。
16	EN2	RW	0	**EN2**：DAC 通道 2 使能(DAC channel2 enable) 0：关闭 DAC 通道 2； 1：使能 DAC 通道 2；
15	保留	-	-	-
14	DMAUDRIE1	RW	0	**DMAUDRIE1：**DAC 通道 1 DMA 下溢中断使能 0：屏蔽 DAC 通道 1 DMA 下溢中断； 1：使能 DAC 通道 1 DMA 下溢中断。
13	DMAEN1	RW	0	**DMAEN1**：DAC 通道 1 DMA 使能(DAC channel1 DMA enable) 0：屏蔽 DAC 通道 1 DMA 模式； 1：使能 DAC 通道 1 DMA 模式。

12：9	MAMP1	RW	4’h0	**MAMP1[3：0]：**DAC 通道 1 屏蔽/增幅选择器(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增幅值。 0000：不屏蔽 LSFR 位 0/三角波增幅值等于 1； 0001：不屏蔽 LSFR 位[1：0]/三角波增幅值等于 3； 0010：不屏蔽 LSFR 位[2：0]/三角波增幅值等于 7； 0011：不屏蔽 LSFR 位[3：0]/三角波增幅值等于 15； 0100：不屏蔽 LSFR 位[4：0]/三角波增幅值等于 31； 0101：不屏蔽 LSFR 位[5：0]/三角波增幅值等于 63； 0110：不屏蔽 LSFR 位[6：0]/三角波增幅值等于 127； 0111：不屏蔽 LSFR 位[7：0]/三角波增幅值等于 255； 1000：不屏蔽 LSFR 位[8：0]/三角波增幅值等于 511； 1001：不屏蔽 LSFR 位[9：0]/三角波增幅值等于 1023； 1010：不屏蔽 LSFR 位[10：0]/三角波增幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11：0]/三角波增幅值等于 4095；
8：7	WAVE1	RW	2’h0	**WAVE1[1：0]：**DAC 通道 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：关闭波形发生器； 01：使能噪声波形发生器； 1x：使能三角波发生器。
6：3	TSEL1	RW	4’h0	**TSEL1[3：0]：**DAC 通道 1 出发选择(DAC channel1 trigger selection) 0000：sw 0001：tim8_trgo 0010：tim7_trgo 0011：tim15_trgo 0100：tim2_trgo 0101：tim4_trgo 0110：exti9 0111：tim6_trgo 1000：tim3_trgo 1001：tim18_trgo 1010：tim1_trgo 1011：tim5_trgo 其他：保留
2	TEN1	RW	0	**TEN1**：DAC 通道 1 触发使能(DAC channel1 trigger enable) 0：关闭 DAC 通道 1 触发 1：使能 DAC 通道 1 触发
1	BOFF1	RW	0	**BOFF1：**关闭 DAC 通道 1 输出缓存(DAC channel1 output buffer disable) 0：使能 DAC 通道 2 输出缓存 1：关闭 DAC 通道 2 输出缓存
0	EN1	RW	0	**EN1**：DAC 通道 1 使能(DAC channel1 enable) 0：关闭 DAC 通道 1；

				1：使能 DAC 通道 1；

15.3.2 DAC 软件触发寄存器(DAC_ SWTRIGR)

Address offset：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														SWTRIG2	SWTRIG1
														W	W

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	SWTRIG2	W	0	**SWTRIG2**：DAC 通道 2 软件触发(DAC channel2 software trigger) 0：关闭 DAC 通道 2 软件触发； 1：使能 DAC 通道 2 软件触发。
0	SWTRIG1	W	0	**SWTRIG1**：DAC 通道 1 软件触发(DAC channel1 software trigger) 0：关闭 DAC 通道 1 软件触发； 1：使能 DAC 通道 1 软件触发。

15.3.3 DAC 通道 1 的 12 位右对齐数据保持寄存器(DAC_DHR12R1)

Address offset：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
Res														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Res				DACC1DHR[11:0]										
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	DACC1DHR	RW	12’h0	**DACC1DHR[11:0]**：DAC 通道 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 通道 1 的 12 位数据。

15.3.4 DAC 通道 1 的 12 位左对齐数据保持寄存器(DAC_DHR12L1)

Address offset：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:4	DACC1DHR	RW	12'h0	**DACC1DHR[11:0]**：DAC 通道 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 通道 1 的 12 位数据。
3:0	保留	-	-	-

15.3.5 DAC 通道 1 的 8 位右对齐数据保持寄存器(DAC_DHR8R1)

Address offset：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								DACC1DHR[7:0]							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	DACC1DHR	RW	8'h0	**DACC1DHR[7:0]**：DAC 通道 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 通道 1 的 8 位数据。

15.3.6 DAC 通道 2 的 12 位右对齐数据保持寄存器(DAC_DHR12R2)

Address offset：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
Res														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Res				DACC2DHR[11:0]										
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	DACC2DHR	RW	12'h0	**DACC2DHR[11:0]**：DAC 通道 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 通道 2 的 12 位数据。

15.3.7 DAC 通道 2 的 12 位左对齐数据保持寄存器(DAC_DHR12L2)

Address offset：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:4	DACC2DHR	RW	8'h0	**DACC2DHR[11:0]**：DAC 通道 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 通道 2 的 12 位数据。
3:0	保留	-	-	-

15.3.8 DAC 通道 2 的 8 位右对齐数据保持寄存器(DAC_DHR8R2)

Address offset：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res								DACC2DHR[7:0]							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	DACC2DHR	RW	8'h0	**DACC2DHR[7:0]**：DAC 通道 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 通道 2 的 8 位数据。

15.3.9 双 DAC 的 12 位右对齐数据保持寄存器(DAC_DHR12RD)

Address offset：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				DACC2DHR[11:0]											
				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**
Res				DACC1DHR[11:0]											
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:16	DACC2DHR	RW	12'h0	**DACC2DHR[11:0]**：DAC 通道 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 通道 2 的 12 位数据。
15:12	保留	-	-	-
11:0	DACC1DHR	RW	12'h0	**DACC1DHR[11:0]**：DAC 通道 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 通道 1 的 12 位数据。

15.3.10双 DAC 的 12 位左对齐数据保持寄存器(DAC_DHR12LD)

Address offset：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[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**
DACC1DHR[11:0]												Res			

RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:20	DACC2DHR	RW	12'h0	**DACC2DHR[11:0]**：DAC 通道 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 通道 2 的 12 位数据。
19:16	保留	-	-	-
15:4	DACC1DHR	RW	12'h0	**DACC1DHR[11:0]**：DAC 通道 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 通道 1 的 12 位数据。
3:0	保留	-	-	-

15.3.11双 DAC 的 8 位右对齐数据保持寄存器(DAC_DHR8RD)

Address offset：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:8	DACC2DHR	RW	8'h0	**DACC2DHR[7:0]**：DAC 通道 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 通道 2 的 8 位数据。
7:0	DACC1DHR	RW	8'h0	**DACC1DHR[7:0]**：DAC 通道 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 通道 1 的 8 位数据。

15.3.12DAC 通道 1 数据输出寄存器(DAC_DOR1)

Address offset：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
Res														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Res				DACC1DOR[11:0]										
				R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	DACC1DOR	R	12'h0	**DACC1DOR[11:0]**：DAC 通道 1 输出数据(DAC channel1 data output) 该位只读，表示 DAC 通道 1 的输出数据。

15.3.13DAC 通道 2 数据输出寄存器(DAC_DOR2)

Address offset：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				DACC2DOR[11:0]											
				R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	DACC2DOR	R	12'h0	**DACC2DOR[11:0]**：DAC 通道 2 输出数据(DAC channel2 data output) 该位只读，表示 DAC 通道 2 的输出数据。

15.3.14DAC 状态寄存器(DAC_SR)

Address offset：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		DMAUDR2 RC_W1	Res												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		DMAUDR1 RC_W1	Res												

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	DMAUDR2	RC_W1	0	**DMAUDR2**：DAC 通道 2DMA underrun flag 该位由硬件置 1，软件写 1 清 0 0：DAC 通道 2 没有 DMA 下溢情况发生； 1：DAC 通道 2 有 DMA 下溢情况发生。 注意：只有在 DMAUDR2 置“1”后，软件才能写“1”清“0”
28:14	保留	-	-	-
13	DMAUDR1	RC_W1	0	**DMAUDR1**：DAC 通道 1DMA underrun flag 该位由硬件置 1，软件写 1 清 0 0：DAC 通道 1 没有 DMA 下溢情况发生； 1：DAC 通道 1 有 DMA 下溢情况发生。 注意：只有在 DMAUDR1 置“1”后，软件才能写“1”清“0”
12:0	保留	-	-	-

15.3.15DAC 输出选择寄存器(DAC_OC)

Address offset：0x38

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			Res												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			Res											INTEREN2	INTEREN1

		RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	INTEREN2	RW	0	**INTEREN2：**DAC2 电压输出到片上外设/GPIO 选择信号。 该 bit 由软件控制。 0：DAC2 电压输出到 GPIO 1：DAC2 电压输出到片上外设(ADC, COMP)
0	INTEREN1	RW	0	**INTEREN1：**DAC1 电压输出到片上外设/GPIO 选择信号。 该 bit 由软件控制。 0：DAC1 电压输出到 GPIO 1：DAC1 电压输出到片上外设(ADC, COMP)

16. 电压基准缓冲器(V_{REFBUF})

16.1 简介

内嵌的 V_{REFBUF} 被用作 ADC 参考电压，同时也可以通过 V_{REF+}输出该电压。

16.2 功能描述

内嵌的参考电压支持四挡电压，可通过 ADC_CCR 的 VREFBUFSEL 配置:

- VREFBUFSEL=00：1.08 V
- VREFBUFSEL=01：2.048 V
- VREFBUFSEL=10：2.5 V
- VREFBUFSEL=11：2.9 V

通过置位 ADC_CCR 的 $V_{REFBUFEN}$ 使能 V_{REFBUF}，用户必须设置了 $V_{REFBUFSEL}$，V_{REFBUF} 输出即达到期望值。

17. 比较器(COMP)

17.1 简介

芯片内集成 4 个通用比较器（General purpose comparators）COMP，分别是 COMP1、COMP2、COMP3 和 COMP4。这 4 个模块可以作为单独的模块，也可以与 timer 组合在一起使用。

用作比较器模拟功能的 I/O，必须在 GPIO 寄存器中被配置成模拟模式，且需要使能 SYSCFG.P*_ANA2EN（*=A、B、C、F）寄存器中比较器映射的 PAD。

比较器可以使用在：

- 低功耗模式唤醒
- 模拟信号调节
- 和来自 timer 的 PWM 输出连接时，Cycle by cycle 的电流控制

17.1.1 主要特性

- 支持电压比较功能，每个比较器有可配置的正极或者负极输入，以实现灵活的电压选择
  - 多路 I/O pin
  - V_{CC}/V_{REFCMP} 的 64 档分压
  - 温度传感器输出
  - DAC 输出
- 可编程速度和功耗
- 可编程迟滞功能
- 配置寄存器写保护(LOCK 功能)
- 输出可以被连接到 I/O 或者 timer 的输入
  - OCREF_CLR 事件（cycle by cycle 的电流控制）
  - 为快速 PWM shutdown 的刹车
- 每个 COMP 具有中断产生能力，用作芯片从低功耗模式（睡眠/低功耗睡眠/停止模式）唤醒（通过 EXTI）
- 提供软件可配置数字滤波时间以增强芯片抗干扰能力
- 支持输出消隐以降低开关噪声
- 支持窗口比较功能

17.1.2 模块框图

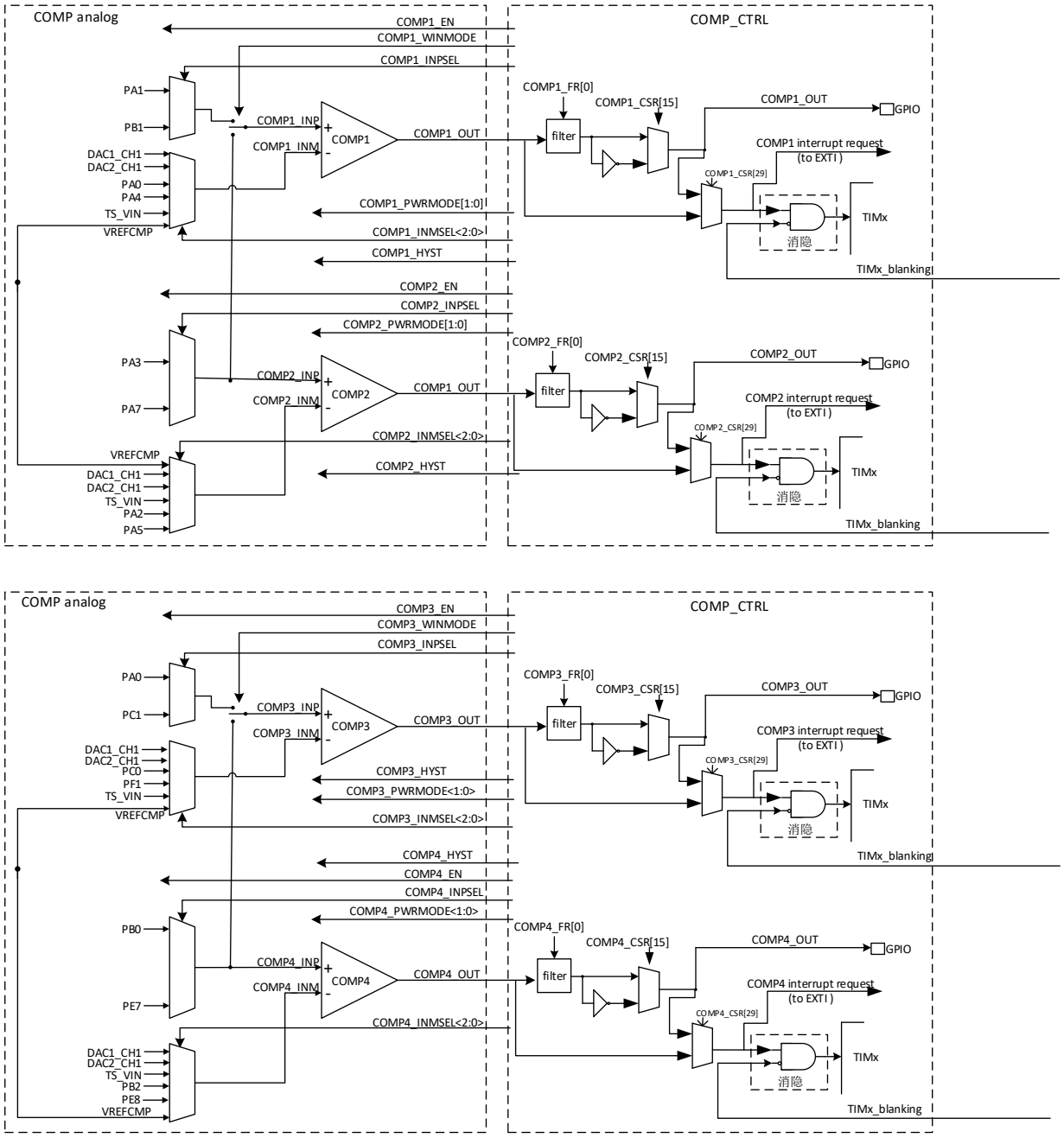


图 17-1 COMP 结构框图

17.2 COMP 功能描述

17.2.1 COMP 的输入和输出

比较器输出可以通过在 GPIO 的可选功能（alternate function）连接到 I/O 引脚。

输出也可以在内部连接到各种 timer 的输入，达到以下目的：

- 连接刹车输入时，PWM 信号的紧急 shut-down 刹车
- 使用 OCREF_CLR 输入的 Cycle-by-cycle 电流控制

- 时序测量的输入捕获

比较器输出到其他模块的连接关系，见下表。

表 17-1COMP 模块数字输出

COMP1_OUT	COMP2_OUT	COMP3_OUT	COMP4_OUT
PA0	PB9	PB15	PB14
PA6	PA2	PC2	PB1
PA11	PA7	PB7	PB6
PB8	PA12	EXTI20	EXTI21
EXTI18	EXTI19	lptim1_ext_trig8	lptim1_ext_trig9
lptim1_ext_trig6	lptim1_ext_trig7	tim1_bk2	tim1_bk2
tim1_bk2	tim1_bk2	tim8_bk2	tim8_bk2
tim8_bk2	tim8_bk2	tim20_bk2	tim20_bk2
tim20_bk2	tim20_bk2	tim1_bk	tim1_bk
tim1_bk	tim1_bk	tim8_bk	tim8_bk
tim8_bk	tim8_bk	tim15_bk	tim15_bk
tim15_bk	tim15_bk	tim16_bk	tim16_bk
tim16_bk	tim16_bk	tim17_bk	tim17_bk
tim17_bk	tim17_bk	tim3_ti3_in1	tim2_ti3_in1
tim2_ti4_in1	tim2_ti4_in2	tim2_ti2_in3	tim2_ti2_in4
tim2_ti2_in1	tim2_ti2_in2	tim3_ti2_in3	tim3_ti2_in4
tim3_ti2_in1	tim3_ti2_in2	tim4_ti2_in3	tim4_ti2_in4
tim4_ti2_in1	tim4_ti2_in2	tim5_ti2_in3	tim5_ti2_in4
tim5_ti2_in1	tim5_ti2_in2	tim18_ti2_in3	tim18_ti2_in4
tim18_ti2_in1	tim18_ti2_in2	tim15_ti2_in2	tim1_ti1_in4
tim1_ti1_in1	tim15_ti2_in1	tim1_ti1_in3	tim2_ti1_in4
tim2_ti1_in1	tim1_ti1_in2	tim2_ti1_in3	tim3_ti1_in4
tim3_ti1_in1	tim2_ti1_in2	tim3_ti1_in3	tim4_ti1_in4
tim4_ti1_in1	tim3_ti1_in2	tim4_ti1_in3	tim8_ti1_in4
tim8_ti1_in1	tim4_ti1_in2	tim8_ti1_in3	tim18_ti1_in4
tim18_ti1_in1	tim8_ti1_in2	tim18_ti1_in3	tim5_ti1_in7
tim15_ti1_in2	tim18_ti1_in2	tim5_ti1_in6	tim1_ocref_clr3
tim5_ti1_in4	tim15_ti1_in3	tim1_ocref_lr2	tim2_ocref_clr3
tim1_ocref_clr0	tim5_ti1_in5	tim2_ocref_lr2	tim3_ocref_clr3
tim2_ocref_clr0	tim1_ocref_clr1	tim3_ocref_lr2	tim4_ocref_clr3
tim3_ocref_clr0	tim2_ocref_clr1	tim4_ocref_lr2	tim5_ocref_clr3
tim4_ocref_clr0	tim3_ocref_clr1	tim5_ocref_lr2	tim8_ocref_clr3
tim5_ocref_clr0	tim4_ocref_clr1	tim8_ocref_lr2	tim15_ocref_clr3
tim8_ocref_clr0	tim5_ocref_clr1	tim15_ocref_lr2	tim16_ocref_clr3
tim15_ocref_clr0	tim8_ocref_clr1	tim16_ocref_lr2	tim17_ocref_clr3
tim16_ocref_clr0	tim15_ocref_clr1	tim17_ocref_lr2	tim18_ocref_clr3
tim17_ocref_clr0	tim16_ocref_clr1	tim18_ocref_lr2	tim1_etr4
tim18_ocref_clr0	tim17_ocref_clr1	tim1_etr3	tim2_etr4
tim1_etr1	tim18_ocref_clr1	tim2_etr3	tim3_etr4
tim2_etr1	tim1_etr2	tim3_etr3	tim4_etr4
tim3_etr1	tim2_etr2	tim4_etr3	tim5_etr4
tim4_etr1	tim3_etr2	tim5_etr3	tim8_etr4
tim5_etr1	tim4_etr2	tim8_etr3	tim18_etr4
tim8_etr1	tim5_etr2	tim18_etr3	-
tim18_etr1	tim8_etr2	-	-
-	tim18_etr2	-	-

17.2.2 COMP 的时钟和复位

COMP 模块有两个时钟源：

1. PCLK（APB clock），用于配置寄存器

2. COMP 时钟，用于模拟比较器输出后的电路（模拟输出的锁存电路、滤毛刺电路等）的时钟，可选择为 PCLK、LSE 或者 LSI。当需要在 stop 模式下工作时，选择 LSE 或者 LSI(通过 RCC_CCIPR.COMPxSEL 进行选择)。

注意：

1. PCLK 和 COMP CLK 由 RCC_APB2ENR.COMPEN 位同时使能，该使能关闭则 PCLK 和 COMP CLK 关闭，若使能则 PCLK 和 COMP CLK 同时开启。

2．进入 Stop 模式前，建议先配置 COMP CLK 为 LSI 或 LSE；然后进入 stop 模式。

COMP 模块的复位信号包含 APB 复位源和 COMP 模块软件复位源

1．APB 复位，用于 COMP 寄存器的复位

2．COMP 软件复位，用于模拟比较器输出后电路（模拟输出的锁存电路、滤毛刺电路等）的复位

注意：当 RCC_APB2RSTR 中的复位信号使能后，COMP 整个模块都会被复位，包括寄存器和用于模拟比较器输出后的电路。

17.2.3 COMP 的滤波功能

如果芯片的工作环境恶劣，比较器的输出会出现噪声信号。使能数字滤波模块，则比较器的输出波形中脉宽小于 COMPx_FR.FLTCNT[15:0]设定时间的噪声信号都可以被滤除。禁止数字滤波模块，则数字滤波模块的输出信号与输入信号相同。

注意：设置 COMP 滤波时间，开启滤波使能应在 COMPx_CSR.EN 使能前完成。

滤波示意如下所示：

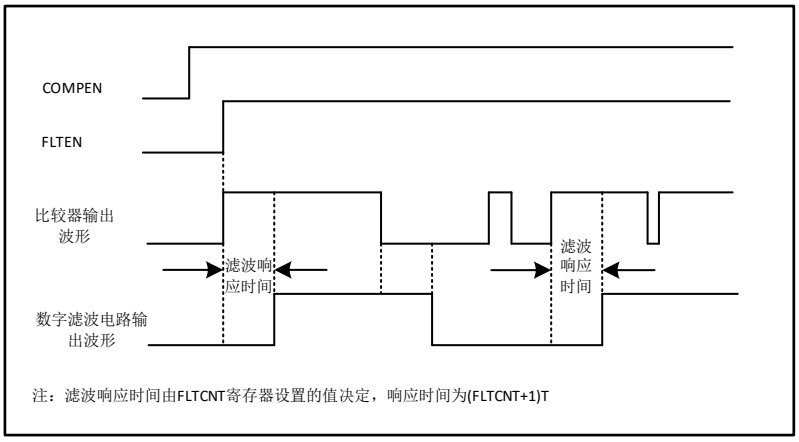


图 17-2 COMP 滤波示意图

17.2.4 COMP 的锁定机制

比较器可以用在安全方面，例如过流和温度保护。对于有特定功能安全需求的应用，需要确保在假的寄存器访问和 PC（program counter）混乱时，比较器的寄存器不能被改写。

由此，比较器控制和状态寄存器可以被写保护（只读）。

如果寄存器的写完成，COMPx_CSR.LOCK 位要被置成 1，这使得整个寄存器变成只读，包括 COMPx_CSR.LOCK 位。

写保护仅能够被芯片的系统复位复位掉。

17.2.5 COMP 的输出消隐

消隐功能的目的是防止电流在调节 PWM 周期开始时因短电流尖峰而跳闸（通常是电源开关反并联二极管中的恢复电流）。通过设置定时器的输出比较信号来实现消隐功能。每个比较器通道的消隐源由软件通过相应 COMPx_CSR 寄存器的 BLANKSEL[2:0]位单独选择。反相消隐信号与比较器输出进行逻辑“与”运算，以产生比较器通道的输出。

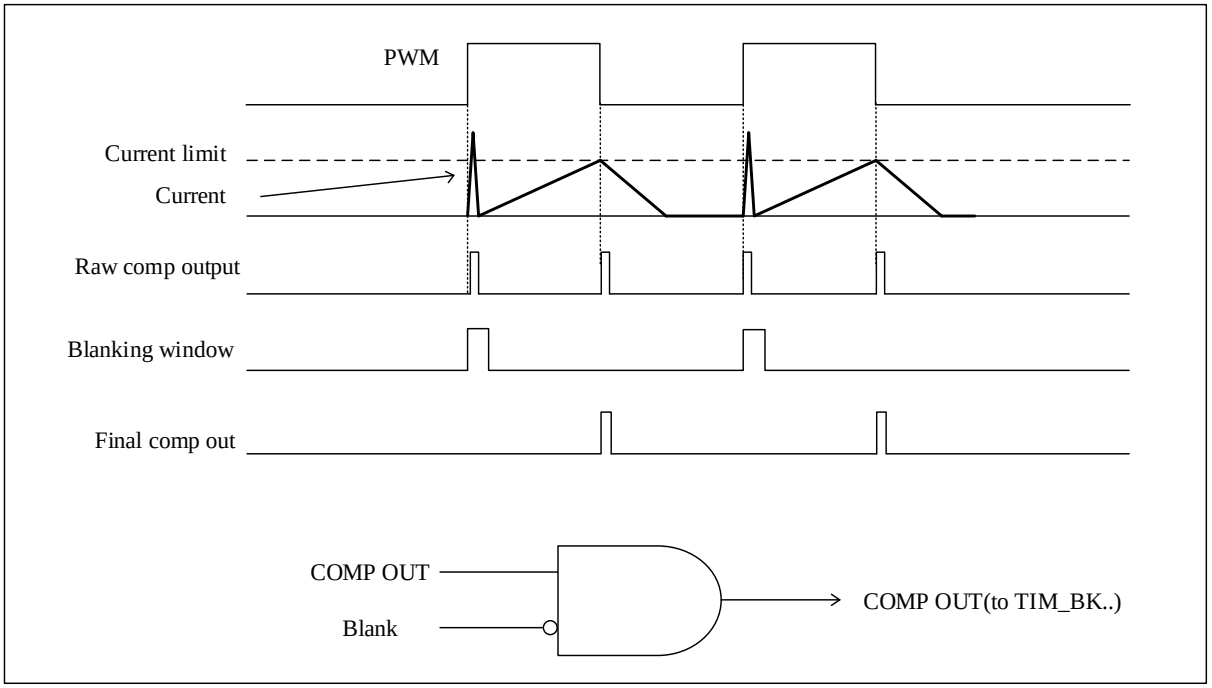


图 17-3 比较器消隐

17.2.6 COMP 的窗口模式

处于窗口模式的比较器简称为窗口比较器，窗口比较器的作用是监测模拟电压是否在低和高阈值范围内。可以使用 COMP1 和 COMP 2 创建窗口比较器，COMP3 和 COMP4 创建窗口比较器。被监测的模拟电压同时连接到两个比较器的正输入端，高阈值和低阈值分别连接到两个比较器的负输入端。

通过使能 COMP1/3_CSR.WINMODE 位，可以将 COMP1 和 COMP2，或 COMP3 和 COMP4 比较器的正输入端连接到一起，起到节省一个 I/O 引脚的作用。

例如当比较器 1 的 WINMODE 被置 1 时，比较器的正输入端将接入 COMP2_INP，当 WINMODE 被清 0 时，比较器的正输入端将接入 COMP1_INP。这样做可以省掉一个 IO 的引脚。

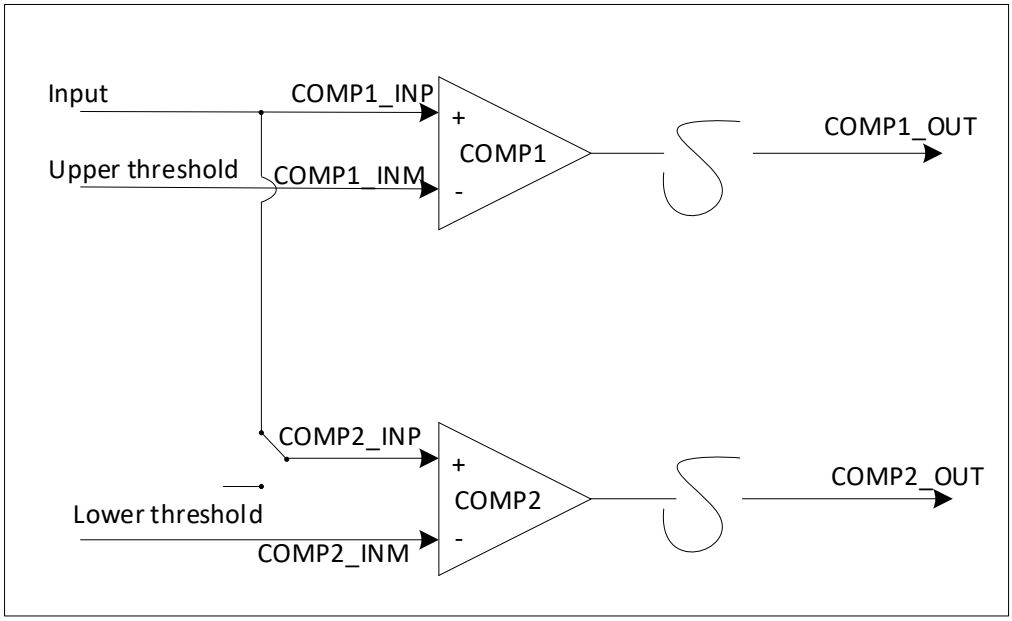


图 17-4 窗口比较器

17.2.7 COMP 的迟滞功能

比较器具有可编程的迟滞，可避免在有噪声信号的情况下出现虚假的输出跳变。通过使能 COMP1_CSR、COMP2_CSR、COMP3_CSR 和 COMP4_CSR 的 HYST 位，可分别打开 COMP1、COMP2、COMP3 和 COMP4 的迟滞功能。

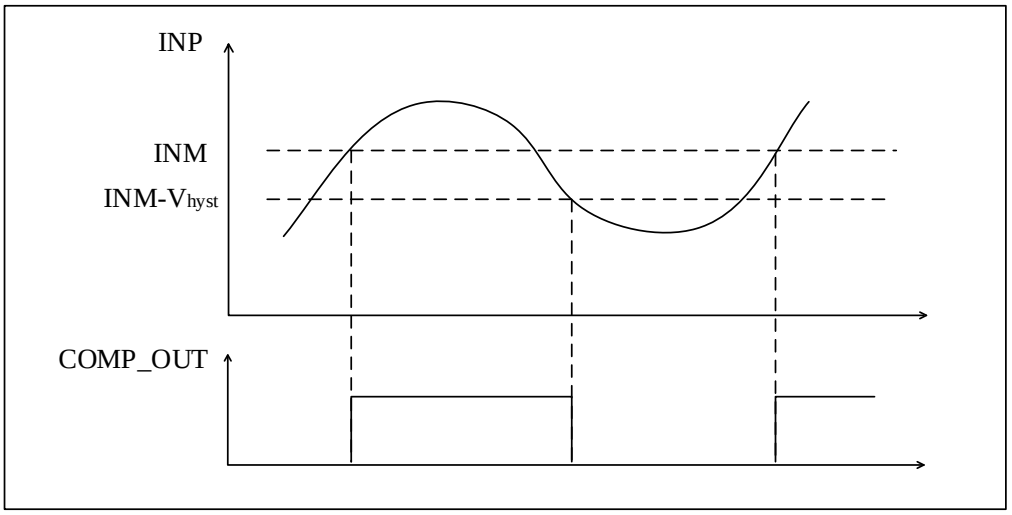


图 17-5 比较器的迟滞示意图

17.2.8 COMP 的功耗模式

比较器的功耗和传输延迟是可以被调整的，以实现特定的应用。该功能是通过 COMPx_CSR 寄存器的 PWRMODE[1:0]位来选择的。

17.2.9 COMP 的低功耗模式

模式	描述
睡眠模式	对 COMP 无影响。 比较器中断可以使设备退出睡眠模式
低功耗运行模式	无影响
低功耗睡眠模式	无影响。 比较器中断可以使设备退出低功耗睡眠模式
停止模式	对 COMP 无影响。 比较器中断可以使设备退出停止模式
待机模式	COMP 寄存器掉电，在退出待机模式后必须重新初始化

17.2.10 COMP 的中断

比较器的输出在芯片内部连接到 EXTI 控制器（Extended interrupts and events）。每个比较器有它自己的 EXTI 线，并能够产生中断或者事件。相同的机制被用作低功耗唤醒。

寄存器描述

17.2.11 COMP1 控制和状态寄存器(COMP1_CSR)

Address offset:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP1_OUT	TIM_EXTI_OUT_SEL	Res.	VCSEL	VCDIV_EN	VCDIV						PWRMODE		Res.	HYST
RW	R	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**
POLARITY	BLANKSEL			WINMODE	Res.	INPSEL	Res			INNSEL			COMP_VSEL		EN
RW	RW			RW		RW				RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP1_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP1_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP1_OUT	R	0	COMP1 输出状态 该位只读，它反映了 COMP1 在经过极性选择的输出电平。
29	TIM_EXTI_OUT_SEL	RW	0	COMP1 输出到 TIM/EXTI 选择。 0：不经过滤波，直接输出到 TIM/EXTI 1：经过滤波，输出到 TIM/EXTI
28	保留	-	-	-
27	VCSEL	RW	0	电压源选择信号 0：V_{REFCMP}

				1：V_{CC}
26	VCDIV_EN	RW	0	VREFCMP 使能信号 0:关闭 V_{REFCMP} 1:使能 V_{REFCMP}
25:20	VCDIV	RW	6’h0	VREFCMP 输出电压选择信号 6’h0：1/64 的分压源(V_{REFCMP}) 6’h1：2/64 的分压源(V_{REFCMP}) 6’h2：3/64 的分压源(V_{REFCMP}) … 6’h3E：63/64 的分压源(V_{REFCMP}) 6’h3F：64/64 的分压源(V_{REFCMP})
19:18	PWRMODE	RW	2’h0	COMP1 功耗模式选择 选择了功耗和由此而来的 COMP1 的速度 2’h0：高速模式(250 uA) 2’h1：中速模式(5 uA) 2’h2：保留 2’h3：保留
17	保留	-	-	-
16	HYST	RW	0	COMP1 迟滞功能使能控制 0：迟滞功能关闭 1：迟滞功能使能
15	POLARITY	RW	0	COMP1 输出极性选择 0：不反向 1：反向
14：12	BLANKSEL	RW	3’h0	比较器 1 消隐信号选择 3’h0：无消隐信号 3’h1：tim1_oc5 3’h2：tim2_oc3 3’h3：tim3_oc3 3’h4：tim8_oc5 3’h5：tim18_oc3 3’h6：tim15_oc1 3’h7：tim4_oc3
11	WINMODE	RW	0	COMP1 窗口模式使能 0：窗口模式不使能, COMP1 的正端输入为 COMP1_INP 1：窗口模式使能, COMP1 的正端输入与 COMP2 的正端输入相同
10	保留	-	-	-
9	INPSEL	RW	0	COMP1 非反向输入的信号选择 0：PA1 1：PB1
8:6	保留	-	-	-
5:3	INNSEL	RW	3’h0	COMP1 反向输入的信号选择 3’h0：PA0 3’h1：PA4 3’h2：DAC_CH1

				3’h3：DAC_CH2 3’h4：VREFCMP 3’h5：TS_VIN 3’h6：保留
2:1	COMP_VSEL	RW	2’h0	VREFCMP 分压源电压选择信号 2’h0：输出的分压源为 0.6 V 2’h1：输出的分压源为 1.5 V 2’h2：输出的分压源为 2.048 V 2’h3：输出的分压源为 2.5 V 注：（0.6 V V_{REFCMP} 精准值存放地址：0x1FFF 7E40 1.5 V V_{REFCMP} 精准值存放地址：0x1FFF 7E44 2.048 V V_{REFCMP} 精准值存放地址：0x1FFF 7E48 2.5 V V_{REFCMP} 精准值存放地址：0x1FFF 7E4C 例如：从地址 0x1FFF 7E40 读到值是 0x0599，表示 VrefBuf 精准值 0.599 V）
0	EN	RW	0	COMP1 使能位 0：关闭 COMP1 1：使能 COMP1

17.2.12 COMP1 滤波寄存器（COMP1_FR)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1															
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
Res.															FLTEN1
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT1	RW	16’h0	比较器 1 采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	保留	-	-	-
0	FLTEN1	RW	0	比较器 1 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 注：该位必须在 COMP1 的 EN 为 0 时置位

17.2.13 COMP2 控制和状态寄存器(COMP2_CSR)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP2_OUT	TIM_EXTI_OUT_SEL.	Res.									PWRMODE		Res.	
RW	R	RW										RW	RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	BLANKSEL			res		INPSEL	res			INNSEL			res	HYST	EN
RW	RW	RW	RW			RW				RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP2_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP2_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP2_OUT	R	0	COMP2 输出状态 该位只读，它反映了 COMP2 在经过极性选择的输出电平。
29	TIM_EXTI_OUT_SEL	RW	0	COMP2 输出到 TIM/EXTI 选择。 0：不经过滤波，ana_out（可以消隐）直接输出到 TIM/EXTI 1：选择输出到 PAD 上的 comp_out（可以滤波，可以消隐）输出到 TIM/EXTI
28：20	保留	-	-	-
19：18	PWRMODE	RW	2’h0	COMP2 功耗模式选择 选择了功耗和由此而来的 COMP2 的速度 2’h0：高速模式(250 uA) 2’h1：中速模式(5 uA) 2’h2：保留 2’h3：保留
17：16	保留	-	-	-
15	POLARITY	RW	0	COMP2 极性选择 0：不反向 1：反向
14:12	BLANKSEL	RW	3’h0	COMP2 消隐信号选择 3’h0：无消隐信号 3’h1：tim1_oc5 3’h2：tim2_oc3 3’h3：tim3_oc3 3’h4：tim8_oc5 3’h5：tim18_oc3 3’h6：tim15_oc1 3’h7：tim4_oc3
11:10	保留	-	-	-
9	INPSEL	RW	0	COMP2 非反向输入的信号选择 0：PA3 1：PA7
8:6	保留	-	-	-
5:3	INNSEL	RW	3’h0	COMP2 反向输入的信号选择 3’h0：PA2

				3’h1：PA5 3’h2：DAC_CH1 3’h3：DAC_CH2 3’h4：V_{REFCMP} 3’h5：TS_VIN 3’h6：保留
2	保留	-	-	-
1	HYST	RW	0	COMP2 迟滞功能使能控制 0：迟滞功能关闭 1：迟滞功能使能
0	EN	RW	0	COMP2 使能位 0：关闭 COMP2 1：使能 COMP2

17.2.14COMP2 滤波寄存器(COMP2_FR)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT2[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**
Res.															FLTEN2
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT2	RW	16’h0	比较器 2 采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	保留	-	-	-
0	FLTEN2	RW	0	比较器 2 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 Note：该位必须在 COMP2 的 EN 为 0 时置位

17.2.15COMP3 控制和状态寄存器(COMP3_CSR)

Address offset:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	COMP3_OUT	TIM_EXTI_OUT_SEL.	Res.									PWRMODE		Res.	
RW	R	RW										RW	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POLARITY	BLANKSEL[2:0]			WINMODE	Res.	INPSEL	Res			INNSEL			Res	HYST	EN
RW	RW	RW	RW	RW		RW				RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP3_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP3_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP3_OUT	R		COMP3 输出状态 该位只读，它反映了 COMP3 在经过极性选择的输出电平。
29	TIM_EXTI_OUT_SEL	RW	0	COMP3 输出到 TIM/EXTI 选择。 0：不经过滤波，ana_out（可以消隐）直接输出到 TIM/EXTI 1：选择输出到 PAD 上的 comp_out（可以滤波，可以消隐）输出到 TIM/EXTI
28：20	保留	-	-	-
19：18	PWRMODE	RW	2’h0	COMP3 功耗模式选择 选择了功耗和由此而来的 COMP3 的速度 2’h0：高速模式(250 uA) 2’h1：中速模式(5 uA) 2’h2：保留 2’h3：保留
17：16	保留	-	-	-
15	POLARITY	RW		COMP3 极性选择 软件可读可写 0：不反向 1：反向
14:12	BLANKSEL	RW	3’h0	COMP3 消隐信号选择 3’h0：无消隐信号 3’h1：tim1_oc5 3’h2：tim3_oc3 3’h3：tim2_oc4 3’h4：tim8_oc5 3’h5：tim18_oc4 3’h6：tim15_oc1 3’h7：tim4_oc3
11	WINMODE	RW		COMP3 窗口模式使能 0：窗口模式不使能, COMP3 的正端输入为 COMP3_INP 1：窗口模式使能, COMP3 的正端输入与 COMP4 的正端输入相同
10	保留	-	-	-
9	INPSEL	RW	0	COMP3 非反向输入的信号选择 0：PA0 1：PC1
8:6	保留	-	-	-
5：3	INNSEL	RW	3’h0	COMP3 不反向输入的信号选择 3’h0：PC0

				3’h1：PF1 3’h2：DAC_CH1 3’h3：DAC_CH2 3’h4：V_{REFCMP} 3’h5：TS_VIN 3’h6：保留
2	保留	-	-	-
1	HYST	RW	0	COMP3 迟滞功能使能控制 0：迟滞功能关闭 1：迟滞功能使能
0	EN	RW		COMP3 使能位 0：关闭 COMP3 1：使能 COMP3

17.2.16 COMP3 滤波寄存器(COMP3_FR)

Address offset:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT3[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**
Res.															FLTEN3
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT3	RW	16’h0	比较器 3 采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	保留	-	-	-
0	FLTEN3	RW	0	比较器 3 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 Note：该位必须在 COMP3 的 EN 为 0 时置位

17.2.17 COMP4 控制和状态寄存器(COMP4_CSR)

Address offset:0x30

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK.	COMP4_OUT	TIM_EXTI_OUT_SEL.	Res.									PWRMODE		Res.	
RW	R	RW										RW	RW		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POLARITY	BLANKSEL[2:0]			Res		INPSEL	Res			INNSEL			Res	HYST	EN
RW	RW	RW	RW			RW				RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	COMP4_CSR 寄存器 lock 软件置位，系统复位清零。当被置位，则会锁定 COMP4_CSR 寄存器的所有 32 位 0：未锁定，可读写整个寄存器 1：锁定，整个寄存器只读
30	COMP4_OUT	R	0	COMP4 输出状态 该位只读，它反映了 COMP4 在经过极性选择的输出电平。
29	TIM_EXTI_OUT_SEL	RW	0	COMP4 输出到 TIM/EXTI 选择。 0：不经过滤波，ana_out（可以消隐）直接输出到 TIM/EXTI 1：选择输出到 PAD 上的 comp_out（可以滤波，可以消隐）输出到 TIM/EXTI
28：20	保留	-	-	-
19：18	PWRMODE	RW	0	COMP4 功耗模式选择 选择了功耗和由此而来的 COMP3 的速度 2’h0：高速模式(250 uA) 2’h1：中速模式(5 uA) 2’h2：保留 2’h3：保留
17：16	保留	-	-	-
15	POLARITY	RW	0	COMP4 极性选择 软件可读可写 0：不反向 1：反向
14:12	BLANKSEL	RW	3’h0	比较器 4 消隐信号选择 3’h0：无消隐信号 3’h1：tim1_oc4 3’h2：tim8_oc5 3’h3：tim15_oc2 3’h4：tim1_oc5 3’h5：tim18_oc4 3’h6：tim15_oc1 3’h7：tim4_oc3
11:10	保留	-	-	-
9	INPSEL	RW	0	COMP4 不反向输入的信号选择，软件可读可写 0：PB0 1：PE7
8:6	保留	-	-	-
5：3	INNSEL	RW	3’h0	COMP4 不反向输入的信号选择 3’h0：PB2 3’h1：PE8 3’h2：DAC_CH1 3’h3：DAC_CH2 3’h4：V_{REFCMP}

				3’h5：TS_VIN 3’h6：保留
2	保留	-	-	-
1	HYST	RW	0	COMP4 迟滞功能使能控制 0：迟滞功能关闭 1：迟滞功能使能
0	EN	RW	0	COMP4 使能位 软件可读可写 0：关闭 COMP4 1：使能 COMP4

17.2.18 COMP4 滤波寄存器(COMP4_FR)

Address offset:0x34

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT4[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
Res.															FLTEN4
															RW

Bit	Name	R/W	Reset Value	Function
31:16	FLTCNT4	RW	16’h0	比较器 4 采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15:0]
15:1	保留	-	-	-
0	FLTEN4	RW	0	比较器 4 数字滤波功能配置 0：禁止数字滤波功能 1：使能数字滤波功能 Note：该位必须在 COMP4 的 EN 为 0 时置位

18. 运算放大器(OPA)

18.1 概述

内嵌 3 个运算放大器，每个运算放大器有两个输入和一个输出。三个 I/O 可以连接到外部引脚，从而实现任何类型的外部互连。运算放大器可以在内部配置成可编程的增益放大器，作为一个放大器正向增益范围为 2 至 32，反相增益范围从-1 至-31。

正输入可以连接到内部 DAC。输出连接到内部 ADC。

18.1.1 主要特性

- 轨对轨输入和输出电压范围
- 低输入偏置电流
- 低输入偏移电压
- 高频增益带宽
- 高速模式可实现更好的转换率

18.1.2 模块框图

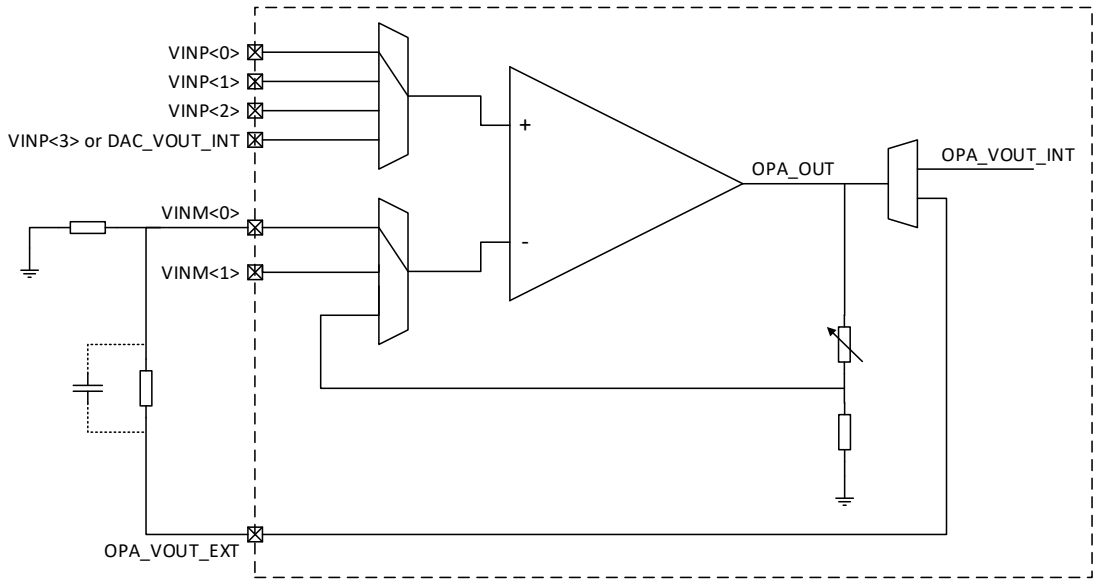


图 18-1 OPA 模块框图

18.2 OPA 功能描述

OPA 有多种模式。每个 OPA 可以单独使能，禁用时输出为高阻抗。

18.2.1 OPA 输出重定向到内部 ADC 通道

通过设置 OPAx_CSR 寄存器中的 OPAINTOEN 位，运算放大器输出可以在内部重定向到 ADC 通道。在这种情况下，映射到 OPAx_VOUT 的 GPIO 被释放可以用于其他目的。ADC 可以在内部测量 OPAx_VOUT 电压。

18.2.2 OPA 复位和时钟

RCC 提供的 OPA 时钟与 PCLK2（APB2 时钟）同步。在 RCC 控制器中提供时钟使能控制位。OPAx_CSR.OPAEN 位用来启用和禁用 OPA 模块。如果需要更改 OPA 寄存器配置，必须在使能 OPAEN 位之前，以避免对输出产生影响。当不再需要运算放大器的输出时，可以禁用 OPA 以节省功耗。禁用 OPA 时，将保留先前设置的所有配置。

18.2.3 初始配置

在默认配置条件下，OPA 会处于使用标准性能的正常工作模式，设置 OPAx_CSR.OPAHSM 将会使 OPA 切换到高速模式并获得更好的转换率。默认配置条件下，OPA 的三个输入/输出信号会连接到外部引脚。一旦设置了 OPAx_CSR 寄存器中的 OPAEN 位，OPA 即可工作。两个输入引脚和输出引脚按照下表进行连接，默认连接设置也可以更改。

注意：输入和输出引脚必须在相应的 GPIOx_MODER 寄存器配置为模拟模式（默认状态）。

18.2.4 信号连接

运算放大器引脚的连接由 OPAx_CSR 寄存器决定。下表中描述了 3 个运算放大器（OPAx，x=1…3）的连接。

表 18-1 运算放大器可能的连接

Signal	IO pin	Internal	Comment
OPA1_VINM	PA3(VINM<0>) PC5(VINM<1>)	OPA1_OUT or PGA	由 OPA1_VMSEL 和 OPA1_PGA_GAIN 控制
OPA1_VINP	PA1(VINP<0>) PA3(VINP<1>) PA7(VINP<2>)	DAC1_VOUT_INT	由 OPA1_VPSEL 控制
OPA1 output	PA2	ADC1_IN13	由 OPA1_INTOEN 控制
OPA2_VINM	PA5(VINM<0>) PC5(VINM<1>)	OPA2_OUT or PGA	由 OPA2_VMSEL 和 OPA2_PGA_GAIN 控制
OPA2_VINP	PA7(VINP<0>) PB14(VINP<1>) PB0(VINP<2>) PD14(VINP<3>)	-	由 OPA2_VPSEL 控制
OPA2 output	PA6	ADC2_IN16	由 OPA2_INTOEN 控制
OPA3_VINM	PB2(VINM<0>) PB10(VINM<1>)	OPA3_OUT or PGA	由 OPA3_VMSEL 和 OPA3_PGA_GAIN 控制
OPA3_VINP	PB0(VINP<0>) PB13(VINP<1>) PA1(VINP<2>)	DAC2_VOUT_INT	由 OPA3_VPSEL 控制

OPA3 output	PB1	ADC2_IN18,ADC3_IN13	由 OPA3_INTOEN 控制

18.3 OPA 模式

运算放大器的输入和输出都可以通过 GPIO 访问。运算放大器可用于多种配置环境：

- 独立模式(外部增益设置模式)
- PGA 模式(可编程增益放大器模式)

18.3.1 独立模式(外部增益设置模式)

首先从 OPAx_CSR 默认值和 GPIOx_MODER 默认状态开始，一旦设置了 OPAx_CSR.OPAEN 位，两个输入引脚和输出引脚就连接到运算放大器。

此默认配置使 OPA 在正常模式下运行。OPA 可通过如下流程修改配置：

- 通过设置 OPAx_CSR.OPAHSM，将 OPA 设置为运算放大器高速模式，以获得高转换率。

OPA 独立模式配置流程：

1) 配置 OPAx_CSR.VP_SEL/OPAx_CSR.VM_SEL 选择 VINP/VINM
2) 配置 OPAx_CSR.INTOEN 位来使能 OPAx_VOUT_EXT
3) 只要 OPAx_CSR.OPAEN 位被配置，电压就输出到 OPAx_VOUT_EXT

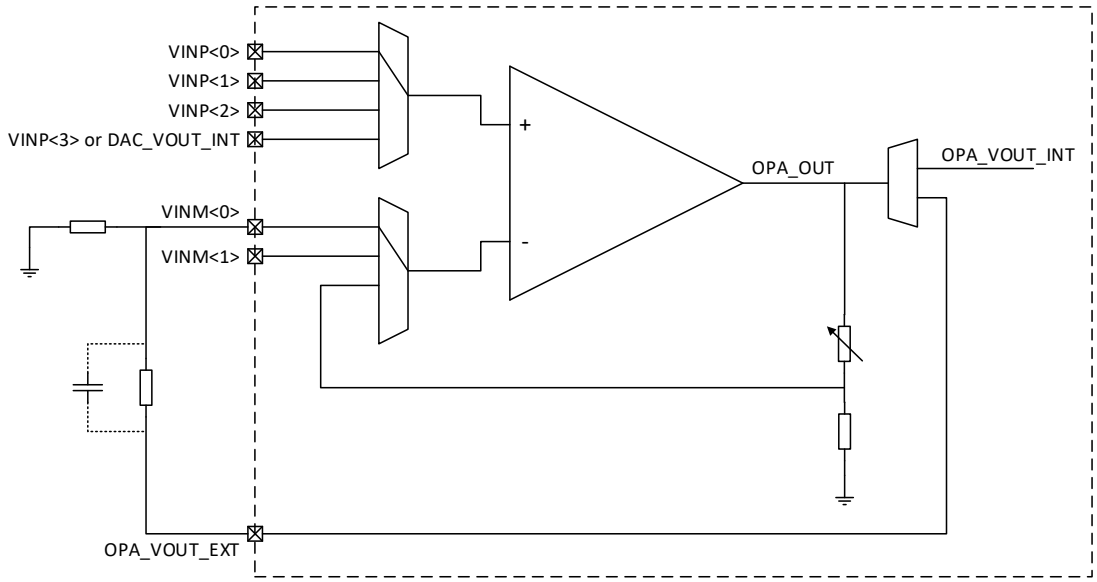


图 18-2 独立模式（外部增益设置模式）

18.3.2 可编程增益放大器模式

OPA 可编程增益放大器模式配置流程：

1) 配置 OPAx_CSR.VM_SEL 位为 2’b10，将反馈电阻连接到 OPAx_VINM 的输入端
2) 配置 OPAx_CSR.PGA_GAIN 位为 5’b00000~5’b00101，使得内部增益为 2, 4, 8, 16 或 32
3) 配置 OPAx_CSR.VP_SEL 位为 2’b00，将相应的 GPIO 的输入连接至 OPAx_VINP 端
4) 配置 OPAx_CSR.INTOEN 位来使能 OPAx_VOUT_EXT 或 OPAx_VOUT_INT
5) 只要 OPAx_CSR.OPAEN 位被设置，OPAx_VINP 引脚上的电压就会根据所选的增益进行放大并在 OPAx_VOUT_EXT 或 OPAx_VOUT_INT 引脚上可见。

注意：为了避免饱和，输入电压必须保持低于 V_{CCA} 除以所选增益。

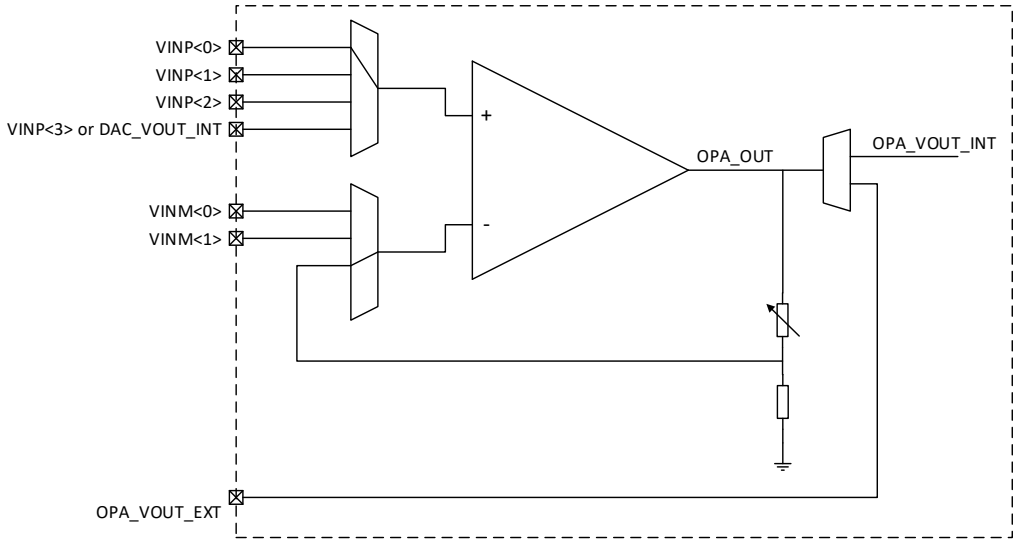


图 18-3 PGA 模式，内部增益设置(x2/x4/x8/x16/x32）反向输入未使用

18.3.3 带外部滤波的可编程增益放大器模式

OPA 带外部滤波的可编程增益放大器模式配置流程：

1) 配置 OPAx_CSR.VM_SEL 位为 2’b10，使得反馈电阻连接到 OPAx_VINM 的输入端
2) 配置 OPAx_CSR.PGA_GAIN 位为 5’b10000~5’b10101，使得带滤波功能的 VINM<0>引脚的内部增益为 2, 4, 8, 16 或 32
3) 配置 OPAx_CSR.VP_SEL 位，将相应的 GPIO 的输入端连接至 OPAx_VINP 端
4) 配置 OPAx_CSR.INTOEN 位来使能 OPAx_VOUT_EXT
5) 只要 OPAx_CSR.OPAEN 位被配置，OPAx_VINP 引脚上的电压就会根据所选的增益进行放大并在 OPAx_VOUT_EXT 引脚上可见。

VINM 上的任何外部连接都可以与内部 PGA 并联使用，例如，可以在 OPA_OUT 和 VINM 之间连接一个电容用于滤波目的。

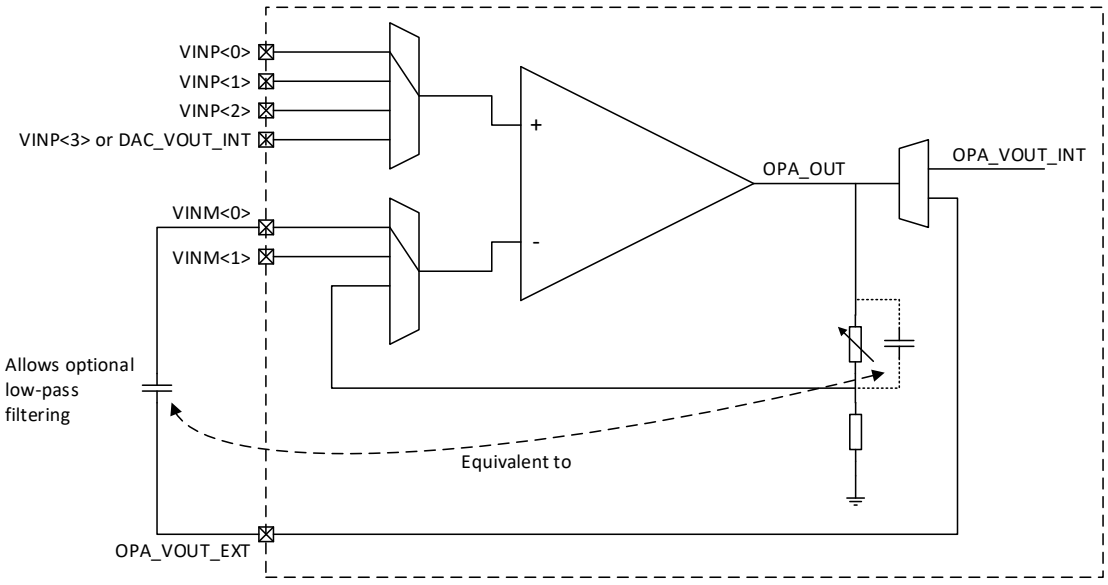


图 18-4 PGA 模式，内部增益设置(x2/x4/x8/x16/x32）反向输入用于滤波

18.3.4 可编程增益放大器，带外部偏置的正向或反相模式

OPA 带外部偏置的正向或反相模式配置流程：

1) 配置 OPAx_CSR.VM_SEL 位为 2’b10，使得反馈电阻连接到 OPAx_VINM 的输入端
2) 配置 OPAx_CSR.PGA_GAIN 位为 5’b01000~5’b01101，使得在 VINM<0>端的反向增益为-1，-3，-7, -15, -31 / 非反向增益为 2, 4, 8, 16, 32
3) 配置 OPAx_CSR.VP_SEL 位，将相应的 GPIO 的输入端连接至 OPAx_VINP 端
4) 配置 OPAx_CSR.INTOEN 位来使能 OPAx_VOUT_EXT 或 OPAx_VOUT_INT
5) 只要 OPAx_CSR.OPAEN 位被设置，OPAx_VINP 引脚上的电压就会根据所选的增益进行放大并在 OPAx_VOUT_EXT 或 OPAx_VOUT_INT 引脚上可见。

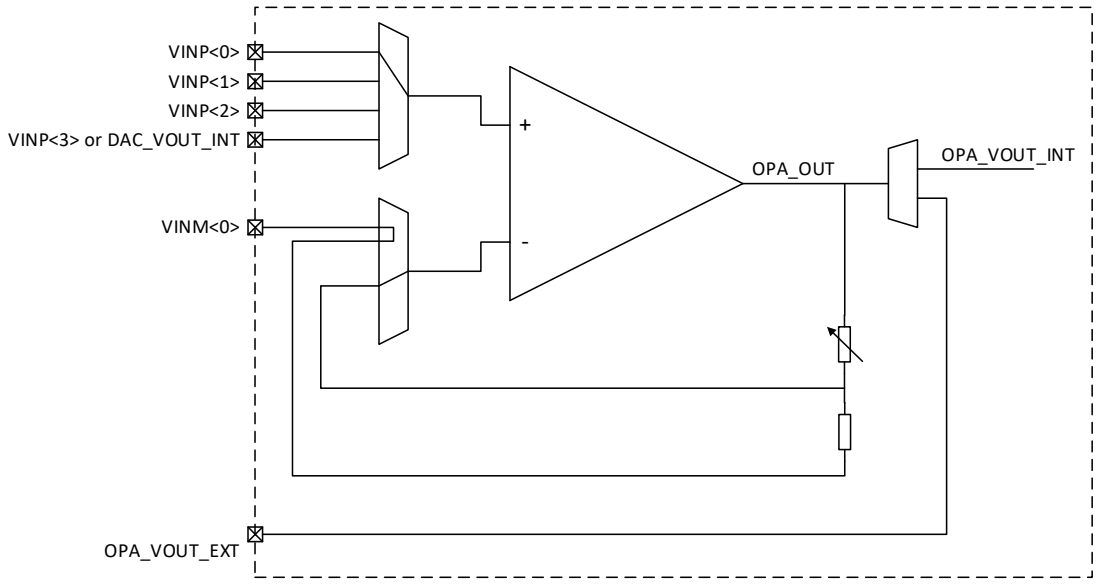


图 18-5 PGA 模式,非反向增益设置(x2/x4/x8/x16/x32）或反向增益设置(x-1/x-3/x-7/x-15/x-31)

18.3.5 可编程增益放大器，带外部偏置的正向或带滤波的反相模式

OPA 带外部偏置的正向或带滤波的反相模式配置流程：

1) 配置 OPAx_CSR.VM_SEL 位为 2’b10，将反馈电阻连接到 OPAx_VINM 的输入端
2) 配置 OPAx_CSR.PGA_GAIN 位为 5’b11000~5’b11101，使得带滤波功能的 VINM<0>和 VINM<1>引脚上的反向增益为-1, -3, -7, -15, -31 / 非反向增益为 2, 4, 8, 16, 32
3) 配置 OPAx_CSR.VP_SEL 位来将相应的 GPIO 输入端连接到 OPAx_VINP 端
4) 配置 OPAx_CSR.INTOEN 位来使能 OPAx_VOUT_EXT
5) 只要 OPAx__CSROPAEN 位被配置，OPAx_VINP 引脚上的电压就会根据所选的增益进行放大并在 OPAx_VOUT_EXT 引脚上可见。

VINM<1>上的任何外部连接都可以与内部 PGA 并联使用，例如，为了滤波目的，可以在 OPA_OUT 和 VINM<1>之间连接一个电容。

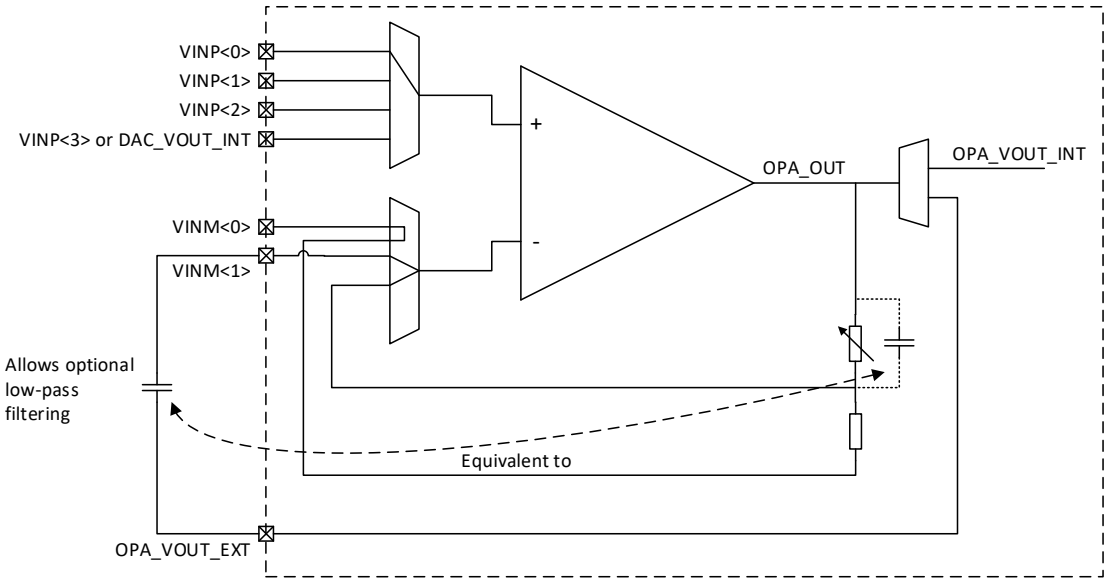


图 18-6 PGA 模式,用于滤波的非反向增益设置(x2/x4/x8/x16/x32）或反向增益设置(x-1/x-3/x-7/x-15/x-31)

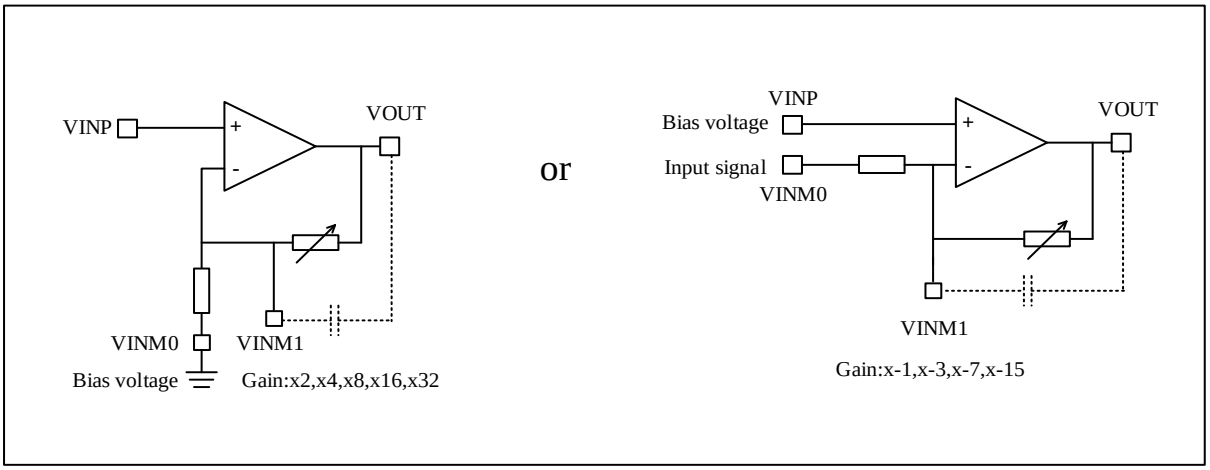


图 18-7 配置举例

18.3.6 电流采样模式

在电机应用中，可以选择共模电压内置，内置电压可以选择为 $V_{CC}/2$、$V_{CC}/5$。也可以使用外接电压。但是，共模电压外接与内置模式不能同时存在。

OPAx_CSR 寄存器中的 OPAx_VBSEL=0 时选择外接电压。

通过配置 OPAx_CSR 寄存器中的 OPAx_VBSEL=1 来选择内置电压；

通过配置 OPA1_CSR 寄存器中的 OPA_VBIAS 来控制 OPAx 的内置电压选择 $V_{CC}/2$ 或 $V_{CC}/5$。

注意 ： 电流采样模式中 OPA 输入输出需使用外部输入/输出，即不能使用内部信号做 OPA 输入。

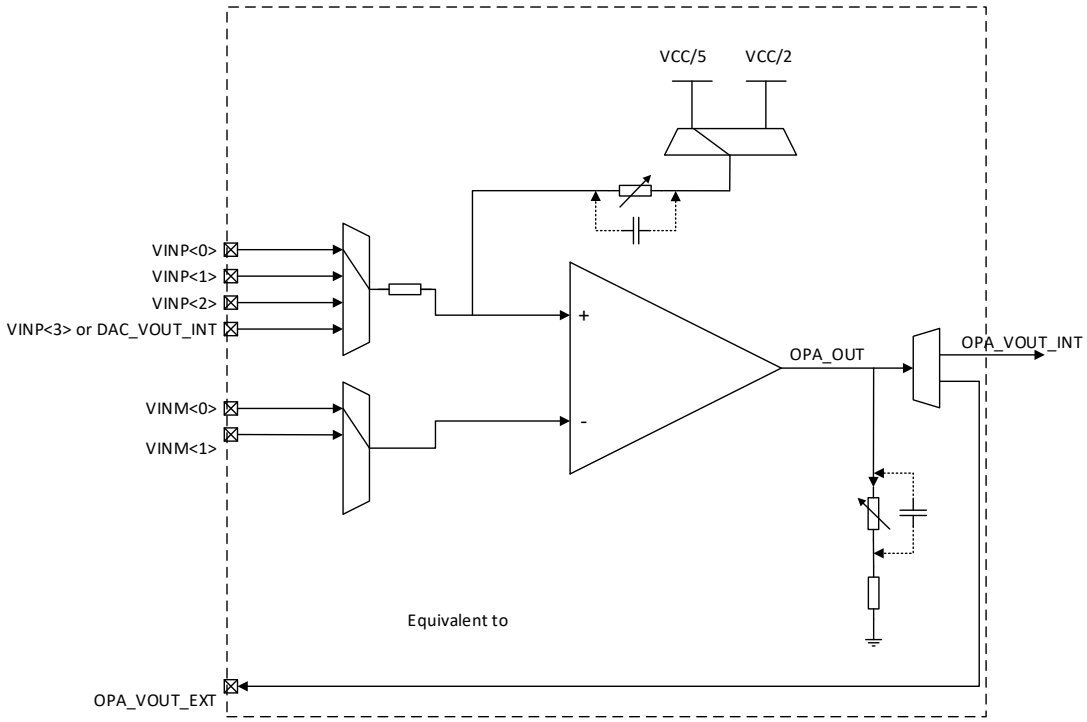


图 18-8 电流采样电路

18.4 OPA PGA 增益

当 OPA 配置为 PGA 模式时，在正向配置中，增益可编程为 x2、x4、x8、x16、x32，在反相配置中可编程为 x-1、x-3、x-7、x-15、x-31。

当 OPA 配置为正向模式时，增益仅取决于内部电阻分压器。当配置为反相模式时，增益因子不仅定义了片上反馈电阻，还定义了信号源输出阻抗。如果信号源输出阻抗与 PGA 的输入反馈电阻相比不可忽略，则会产生增益误差。

18.5 定时器控制 OPA VINP/VINM 选择模式

OPA 反相和非反相输入的选择可以自动完成。该自动切换由到达 OPA 输入多路复用器的 TIM1 CC6 或 TIM8 CC6 输出触发。这对于需要同时测量第一电机和第二电机上的三相电流的双电机控制非常有用。

通过设置 OPAx_TCMR.TxCM_EN 位(x=1,8)，启用自动开关。反相和非反相输入选择通过 OPAx_TCMR.VPS_SEL 和 VMS_SEL 配置。如果 TxCM_EN 位被清除，则使用 OPAx_CSR 中的 VP_SEL 和 VM_SEL 选择完成。

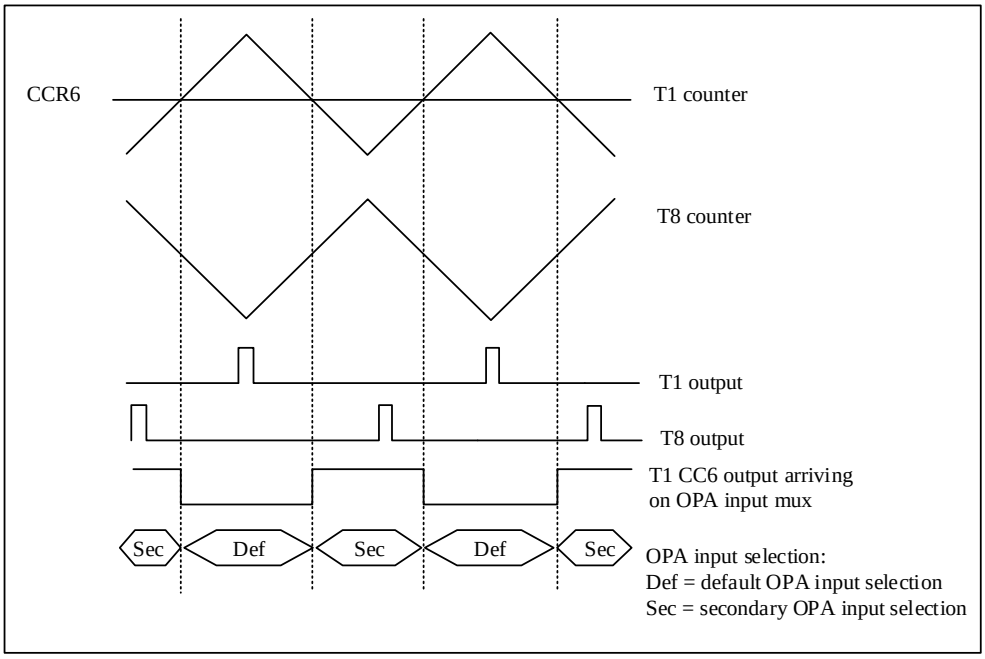


图 18-9 timer 控制多路复用器模式

18.6 OPA 低功耗模式

表 18-2 低功耗模式对 OPA 的影响

模式	描述
Sleep	无影响
Low-power run	无影响
Low-power sleep	无影响
Stop0/Stop1	OPA 寄存器值保持,OPA 不工作
Standby	OPA 寄存器掉电，在退出 Standby 模式后必须重新初始化。

18.7 寄存器描述

18.7.1 OPA1 控制/状态寄存器（OPA1_CSR)

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
LOCK	Res.	OPA1_VBSEL	Res										PGA_GAIN		
RW		RW											RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN		Res.			VBIAS		OPAINTOEN	OPAHSM	VM_SEL		Res	VP_SEL		Res.	OPAEN
RW	RW				RW	RW	RW	RW	RW	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。 此位用于将 OPA1_CSR 寄存器配置为只读。

				0:OPA1_CSR 可读写 1:OPA1_CSR 只读
30	保留	-	-	-
29	OPA1_VBSEL	RW	0	OPA1 共模电压选择 0：共模电压选择外接电压 1：共模电压选择内置电压
28:19	保留	-	-	-
18:14	PGA_GAIN	RW	5'h0	OPA 可配置放大器增益值 5'b00000：非反向内部增益=2 5'b00001：非反向内部增益=4 5'b00010：非反向内部增益=8 5'b00011：非反向内部增益=16 5'b00100：非反向内部增益=32 5'b00101：未使用 5'b00110：未使用 5'b00111：未使用 5'b01000：用于输入或偏置的 VINM0 引脚的反相增益=-1/正向增益=2 5'b01001：用于输入或偏置的 VINM0 引脚的反相增益=-3/正向增益=4 5'b01010：用于输入或偏置的 VINM0 引脚的反相增益=-7/正向增益=8 5'b01011：用于输入或偏置的 VINM0 引脚的反相增益=-15/正向增益=16 5'b01100：用于输入或偏置的 VINM0 引脚的反相增益=-31/正向增益=32 5'b01101：未使用 5'b01110：未使用 5'b01111：未使用 5'b10000：具有滤波功能的 VINM0 引脚上的非反向增益=2 5'b10001：具有滤波功能的 VINM0 引脚上的非反向增益=4 5'b10010：具有滤波功能的 VINM0 引脚上的非反向增益=8 5'b10011：具有滤波功能的 VINM0 引脚上的非反向增益=16 5'b10100：具有滤波功能的 VINM0 引脚上的非反向增益=32 5'b10101：未使用 5'b10110：未使用 5'b10111：未使用 5'b11000：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-1/非反向增益=2 5'b11001：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-3/正向增益=4 5'b11010：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-7/正向增益=8

				5’b11011：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-15/正向增益=16 5’b11100：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-31/正向增益=32 5’b11101：未使用 5’b11110：未使用 5’b11111：未使用
13:11	保留	-	-	-
10:9	OPA_VBIAS	RW	2’h0	OPA1 共模电压内置电压选择 2’h0：保留 2’h1：V_{CC}/2 2’h2：V_{CC}/5 2’h3：保留
8	OPAINTOEN	RW	0	OPA 内部输出使能 。 0：OPA 输出连接到输出 pin 1：OPA 输出连接到一个 ADC 通道，同时不连接到输出 pin。
7	OPAHSM	RW	0	OPA 高速模式。必须禁用运算放大器才能更改此配置。 0：OPA 工作在正常模式 1：OPA 工作在高速模式
6:5	VM_SEL	RW	2’h0	反向输入选择 2’h0：VINM0 引脚连接到 OPA1 VINM 输入 2’h1：VINM1 引脚连接到 OPA1 VINM 输入 2’h2：反馈电阻器连接到 OPA1 VINM 输入（PGA 模式），反相输入选择取决于 PGA_GAIN 设置 2’h3：保留
4	保留	-	-	-
3:2	VP_SEL	RW	2’h0	非反向输入选择 2’h0：VINP0 引脚连接到 OPA1 VINP 输入 2’h1：VINP1 引脚连接到 OPA1 VINP 输入 2’h2：VINP2 引脚连接到 OPA1 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA1 VINP 输入
1	保留	-	-	-
0	OPAEN	RW	0	OPA 使能 0：禁止 OPA 1：使能 OPA

18.7.2 OPA2 控制/状态寄存器（OPA2_CSR）

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
LOCK	Res.	OPA2_VBSEL	Res										PGA_GAIN		
RW		RW											RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PGA_GAIN		Res.					OPAINTOEN	OPAHSM	VM_SEL		Res	VP_SEL		Res.	OPAEN
RW	RW						RW	RW	RW	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。 此位用于将 OPA1_CSR 寄存器配置为只读。 0:OPA2_CSR 可读写 1:OPA2_CSR 只读
30	保留	-	-	-
29	OPA2_VBSEL	RW	0	OPA2 共模电压选择 0：共模电压选择外接电压 1：共模电压选择内置电压
28:19	保留	-	-	-
18:14	PGA_GAIN	RW	5'h0	OPA 可配置放大器增益值 5'b00000：非反向内部增益=2 5'b00001：非反向内部增益=4 5'b00010：非反向内部增益=8 5'b00011：非反向内部增益=16 5'b00100：非反向内部增益=32 5'b00101：未使用 5'b00110：未使用 5'b00111：未使用 5'b01000：用于输入或偏置的 VINM0 引脚的反相增益=-1/正向增益=2 5'b01001：用于输入或偏置的 VINM0 引脚的反相增益=-3/正向增益=4 5'b01010：用于输入或偏置的 VINM0 引脚的反相增益=-7/正向增益=8 5'b01011：用于输入或偏置的 VINM0 引脚的反相增益=-15/正向增益=16 5'b01100：用于输入或偏置的 VINM0 引脚的反相增益=-31/正向增益=32 5'b01101：未使用 5'b01110：未使用 5'b01111：未使用 5'b10000：具有滤波功能的 VINM0 引脚上的非反向增益=2 5'b10001：具有滤波功能的 VINM0 引脚上的非反向增益=4 5'b10010：具有滤波功能的 VINM0 引脚上的非反向增益=8 5'b10011：具有滤波功能的 VINM0 引脚上的非反向增益=16 5'b10100：具有滤波功能的 VINM0 引脚上的非反向增益=32 5'b10101：未使用 5'b10110：未使用 5'b10111：未使用 5'b11000：用于输入或偏置的 VINM0 引脚和用于滤波的

				VINM1 引脚的反相增益=-1/非反向增益=2 5’b11001：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-3/正向增益=4 5’b11010：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-7/正向增益=8 5’b11011：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-15/正向增益=16 5’b11100：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-31/正向增益=32 5’b11101：未使用 5’b11110：未使用 5’b11111：未使用
13:9	保留	-	-	-
8	OPAINTOEN	RW	0	OPA 内部输出使能 。 0：OPA 输出连接到输出 pin 1：OPA 输出连接到一个 ADC 通道，同时不连接到输出 pin。
7	OPAHSM	RW	0	OPA 高速模式。必须禁用运算放大器才能更改此配置。 0：OPA 工作在正常模式 1：OPA 工作在高速模式
6:5	VM_SEL	RW	2’h0	反向输入选择 2’h0：VINM0 引脚连接到 OPA2 VINM 输入 2’h1：VINM1 引脚连接到 OPA2 VINM 输入 2’h2：反馈电阻器连接到 OPA2 VINM 输入（PGA 模式），反相输入选择取决于 PGA_GAIN 设置 2’h3：保留
4	保留	-	-	-
3:2	VP_SEL	RW	2’h0	非反向输入选择 2’h0：VINP0 引脚连接到 OPA2 VINP 输入 2’h1：VINP1 引脚连接到 OPA2 VINP 输入 2’h2：VINP2 引脚连接到 OPA2 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA2 VINP 输入
1	保留	-	-	-
0	OPAEN	RW	0	OPA 使能 0：禁止 OPA 1：使能 OPA

18.7.3 3 控制/状态寄存器（OPA3_CSR）

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
LOCK	Res.	OPA3_VBSEL	Res										PGA_GAIN		
RW		RW											RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PGA_GAIN		Res.					OPAINTOEN	OPAHSM	VM_SEL		Res	VP_SEL		Res.	OPAEN
RW	RW						RW	RW	RW	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。 此位用于将 OPA1_CSR 寄存器配置为只读。 0:OPA3_CSR 可读写 1:OPA3_CSR 只读
30	保留	-	-	-
29	OPA3_VBSEL	RW	0	OPA3 共模电压选择 0：共模电压选择外接电压 1：共模电压选择内置电压
28:19	保留	-	-	-
18:14	PGA_GAIN	RW	5'h0	OPA 可配置放大器增益值 5'b00000：非反向内部增益=2 5'b00001：非反向内部增益=4 5'b00010：非反向内部增益=8 5'b00011：非反向内部增益=16 5'b00100：非反向内部增益=32 5'b00101：未使用 5'b00110：未使用 5'b00111：未使用 5'b01000：用于输入或偏置的 VINM0 引脚的反相增益=-1/正向增益=2 5'b01001：用于输入或偏置的 VINM0 引脚的反相增益=-3/正向增益=4 5'b01010：用于输入或偏置的 VINM0 引脚的反相增益=-7/正向增益=8 5'b01011：用于输入或偏置的 VINM0 引脚的反相增益=-15/正向增益=16 5'b01100：用于输入或偏置的 VINM0 引脚的反相增益=-31/正向增益=32 5'b01101：未使用 5'b01110：未使用 5'b01111：未使用 5'b10000：具有滤波功能的 VINM0 引脚上的非反向增益=2 5'b10001：具有滤波功能的 VINM0 引脚上的非反向增益=4 5'b10010：具有滤波功能的 VINM0 引脚上的非反向增益=8 5'b10011：具有滤波功能的 VINM0 引脚上的非反向增益=16 5'b10100：具有滤波功能的 VINM0 引脚上的非反向增益=32 5'b10101：未使用 5'b10110：未使用 5'b10111：未使用 5'b11000：用于输入或偏置的 VINM0 引脚和用于滤波的

				VINM1 引脚的反相增益=-1/非反向增益=2 5’b11001：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-3/正向增益=4 5’b11010：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-7/正向增益=8 5’b11011：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-15/正向增益=16 5’b11100：用于输入或偏置的 VINM0 引脚和用于滤波的 VINM1 引脚的反相增益=-31/正向增益=32 5’b11101：未使用 5’b11110：未使用 5’b11111：未使用
13:9	保留	-	-	-
8	OPAINTOEN	RW	0	OPA 内部输出使能 。 0：OPA 输出连接到输出 pin 1：OPA 输出连接到一个 ADC 通道，同时不连接到输出 pin。
7	OPAHSM	RW	0	OPA 高速模式。必须禁用运算放大器才能更改此配置。 0：OPA 工作在正常模式 1：OPA 工作在高速模式
6:5	VM_SEL	RW	2’h0	反向输入选择 2’h0：VINM0 引脚连接到 OPA3 VINM 输入 2’h1：VINM1 引脚连接到 OPA3 VINM 输入 2’h2：反馈电阻器连接到 OPA3 VINM 输入（PGA 模式），反相输入选择取决于 PGA_GAIN 设置 2’h3：保留
4	保留	-	-	-
3:2	VP_SEL	RW	2’h0	非反向输入选择 2’h0：VINP0 引脚连接到 OPA3 VINP 输入 2’h1：VINP1 引脚连接到 OPA3 VINP 输入 2’h2：VINP2 引脚连接到 OPA3 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA3 VINP 输入
1	保留	-	-	-
0	OPAEN	RW	0	OPA 使能 0：禁止 OPA 1：使能 OPA

18.7.4 OPA1 timer 控制模式寄存器（OPA1_TCMR）

偏移地址 0x0C

复位值 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res														
RW															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res											T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
											RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	OPA1_TCMR lock。该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。此位用于将 OPAx_TCMR 寄存器配置为只读。 0：OPA1_TCMR 可读写 1：OPA1_TCMR 只读
30:5	保留	-	-	-
4	T8CM_EN	RW	0	TIM8 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM8CC6 输出触发。 0：自动切换关闭 1：自动切换使能
3	T1CM_EN	RW	0	TIM1 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM1 CC6 输出触发。 0：自动切换关闭 1：自动切换使能
2:1	VPS_SEL	RW	2’h0	OPAx 非反相输入二次选择。这些位由软件设置和清除。它们用于在受控多路复用模式被启用时（T1CM_EN=1 或 T8CM_EN=1）选择 OPAx 非反相输入 2’h0：VINP0 引脚连接到 OPA1 VINP 输入 2’h1：VINP1 引脚连接到 OPA1 VINP 输入 2’h2：VINP2 引脚连接到 OPA1 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA1 VINP 输入
0	VMS_SEL	RW	0	OPA1 反相输入二次选择。该位由软件设置和清除。当启用受控多路复用器模式时（T1CM_EN=1 或 T8CM_EN=1），当使用独立模式时（即 VM_SEL=“2’h0”或“2’h1”），它用于选择 OPAx 反相输入： 0：输入来自 VINM0 1：输入来自 VINM1 当使用 PGA（VM_SEL=“2’h2”）时：该位只能置 0。

18.7.5 OPA2 timer 控制模式寄存器（OPA2_TCMR）

偏移地址 0x10

复位值 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res														
RW															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res											T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
											RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	OPA2_TCMR lock。该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。此位用于将 OPAx_TCMR 寄存器配置为只读。 0：OPA2_TCMR 可读写 1：OPA2_TCMR 只读
30:5	保留	-	-	-
4	T8CM_EN	RW	0	TIM8 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM8CC6 输出触发。 0：自动切换关闭 1：自动切换使能
3	T1CM_EN	RW	0	TIM1 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM1 CC6 输出触发。 0：自动切换关闭 1：自动切换使能
2:1	VPS_SEL	RW	2’h0	OPAx 非反相输入二次选择。这些位由软件设置和清除。它们用于在受控多路复用模式被启用时（T1CM_EN=1 或 T8CM_EN=1）选择 OPAx 非反相输入 2’h0：VINP0 引脚连接到 OPA1 VINP 输入 2’h1：VINP1 引脚连接到 OPA1 VINP 输入 2’h2：VINP2 引脚连接到 OPA1 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA1 VINP 输入
0	VMS_SEL	RW	0	OPA1 反相输入二次选择。该位由软件设置和清除。当启用受控多路复用器模式时（T1CM_EN=1 或 T8CM_EN=1），当使用独立模式时（即 VM_SEL=“2’h0”或“2’h1”），它用于选择 OPAx 反相输入： 0：输入来自 VINM0 1：输入来自 VINM1 当使用 PGA（VM_SEL=“2’h2”）时：该位只能置 0。

18.7.6 OPA3 timer 控制模式寄存器（OPA3_TCMR）

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
LOCK	Res														
RW															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res											T8CM_EN	T1CM_EN	VPS_SEL		VMS_SEL
											RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	LOCK	RW	0	OPA3_TCMR lock。该 bit 写一次。它是由软件设置的。它只能通过系统复位或模块软复位来清除。此位用于将 OPAx_TCMR 寄存器配置为只读。 0：OPA3_TCMR 可读写 1：OPA3_TCMR 只读
30:5	保留	-	-	-
4	T8CM_EN	RW	0	TIM8 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM8CC6 输出触发。 0：自动切换关闭 1：自动切换使能
3	T1CM_EN	RW	0	TIM1 控制多路复用模式启用。该位由软件设置和清除。它用于自动控制反相和非反相输入的默认选择（VP_SEL 和 VM_SEL）和二次选择（VPS_SEL 和 VMS_SEL）之间的切换。该自动开关由到达 OPAx 输入多路复用器的 TIM1 CC6 输出触发。 0：自动切换关闭 1：自动切换使能
2:1	VPS_SEL	RW	2’h0	OPAx 非反相输入二次选择。这些位由软件设置和清除。它们用于在受控多路复用模式被启用时（T1CM_EN=1 或 T8CM_EN=1）选择 OPAx 非反相输入 2’h0：VINP0 引脚连接到 OPA1 VINP 输入 2’h1：VINP1 引脚连接到 OPA1 VINP 输入 2’h2：VINP2 引脚连接到 OPA1 VINP 输入 2’h3：DAC3_CH1 引脚连接到 OPA1 VINP 输入
0	VMS_SEL	RW	0	OPA1 反相输入二次选择。该位由软件设置和清除。当启用受控多路复用器模式时（T1CM_EN=1 或 T8CM_EN=1），当使用独立模式时（即 VM_SEL=“2’h0”或“2’h1”），它用于选择 OPAx 反相输入： 0：输入来自 VINM0 1：输入来自 VINM1 当使用 PGA（VM_SEL=“2’h2”）时：该位只能置 0。

19. 高级定时器（TIM1 和 TIM8）

19.1 简介

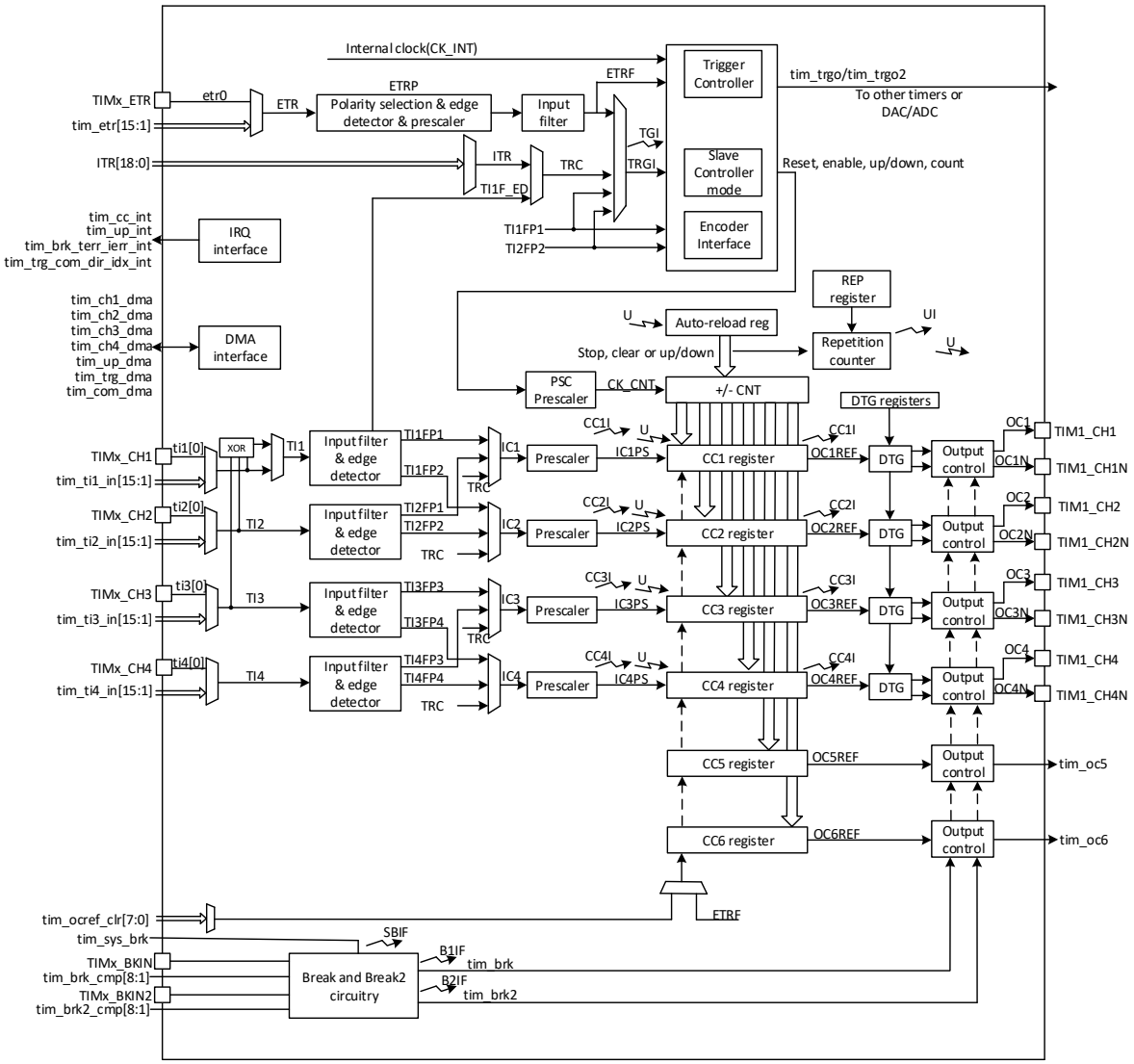
高级控制定时器(TIM1、TIM8)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。高级控制定时器(TIM1)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作。

19.1.1 TIM1/TIM8 主要特征

TIM1 定时器的功能包括：

- 16 位向上、向下、向上/下的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 多达 6 个独立通道：
  - 输入捕获（5/6 通道不支持）
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 带有可编程死区时间的互补输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 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
- 2 路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
- 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

19.1.2 模块框图



19.2 TIM1/TIM8 功能描述

19.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC)
- 自动装载寄存器（TIMx_ARR)
- 重复次数寄存器（TIMx_RCR)

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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(上溢或者下溢)并当 TIMx_CR1 寄存器中的 UDIS

位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下面几张图给出了在预分频器运行时，更改计数器参数的例子。

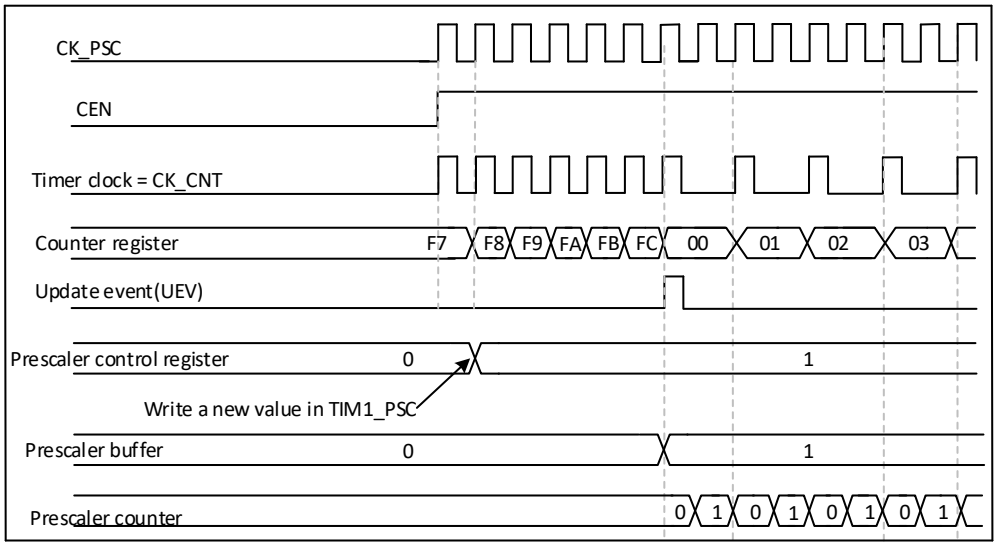


图 19-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

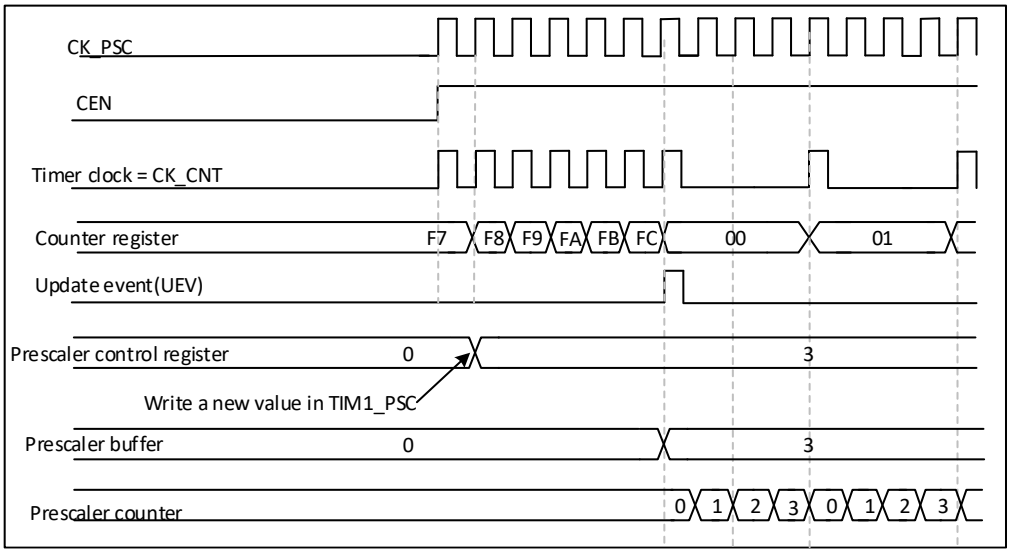


图 19-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

19.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

如果使用了重复计数器，那么更新事件(UEV)需要在上溢次数达到所配置的重复计数寄存器的值加一（即 TIMx_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx_RCR=0），那么每次计数上溢都会产生更新事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断或 DMA 请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 重复计数器被重新加载为 TIMx_RCR 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(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

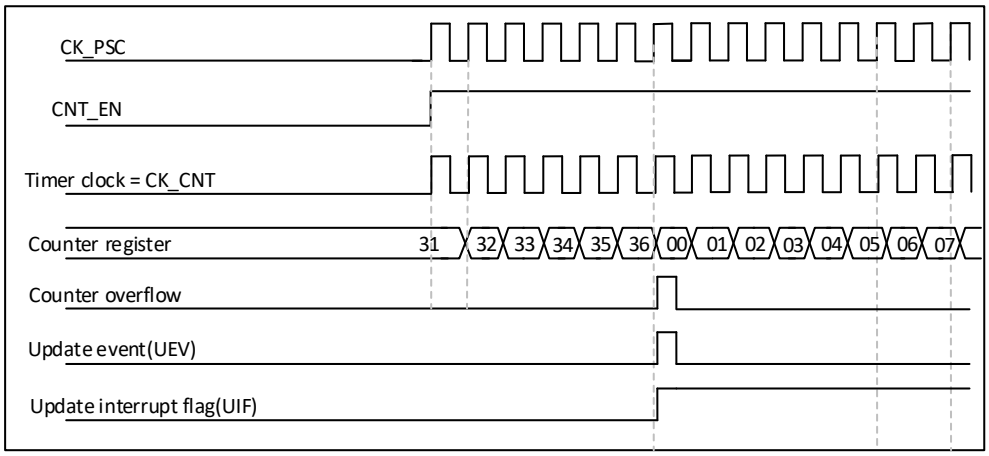


图 19-3 计数器时序图，内部时钟分频因子为 1

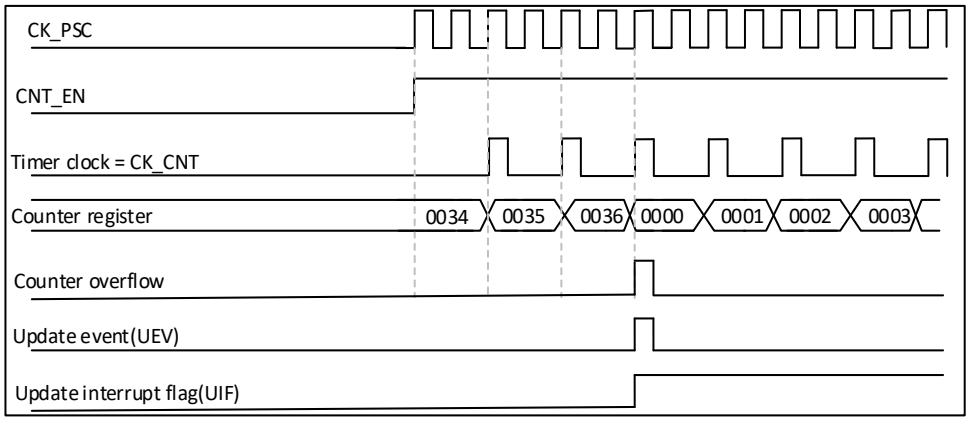


图 19-4 计数器时序图，内部时钟分频因子为 2

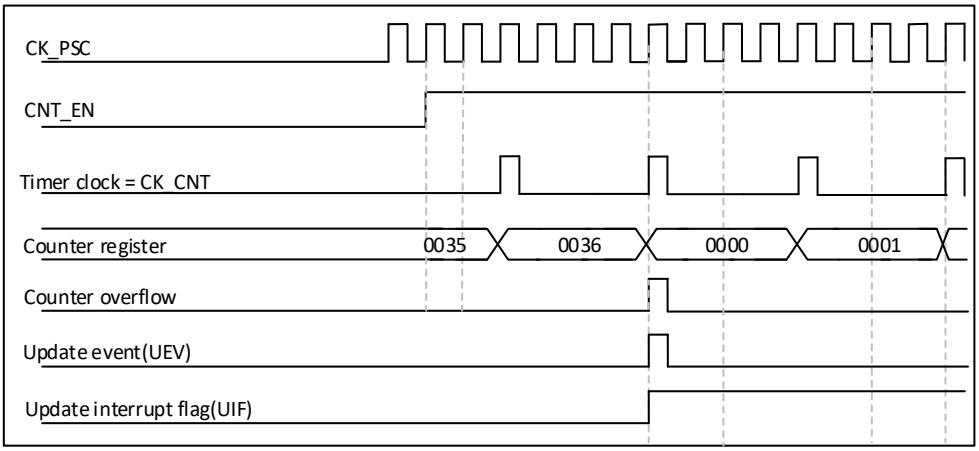


图 19-5 计数器时序图，内部时钟分频因子为 4

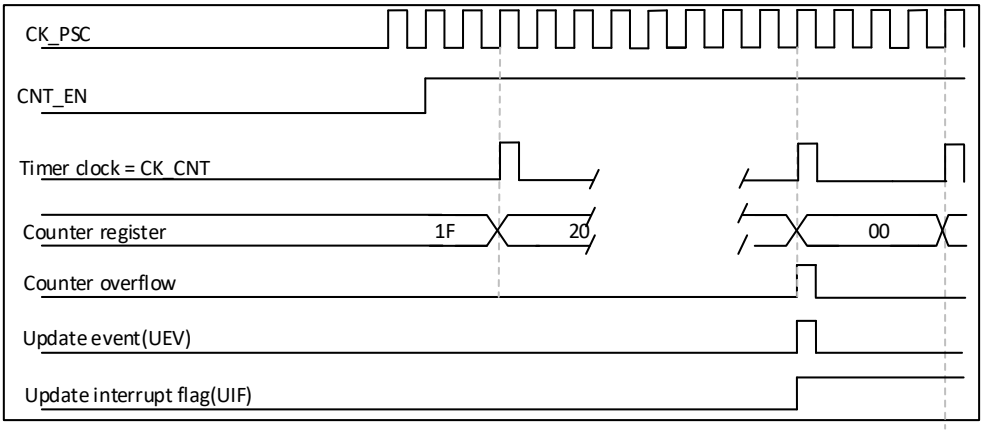


图 19-6 计数器时序图，内部时钟分频因子为 N

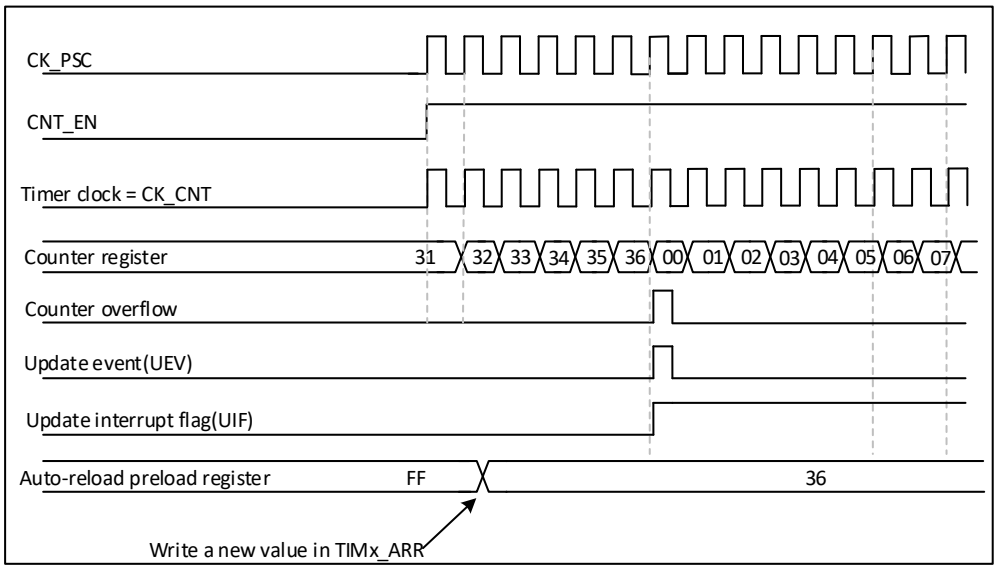


图 19-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

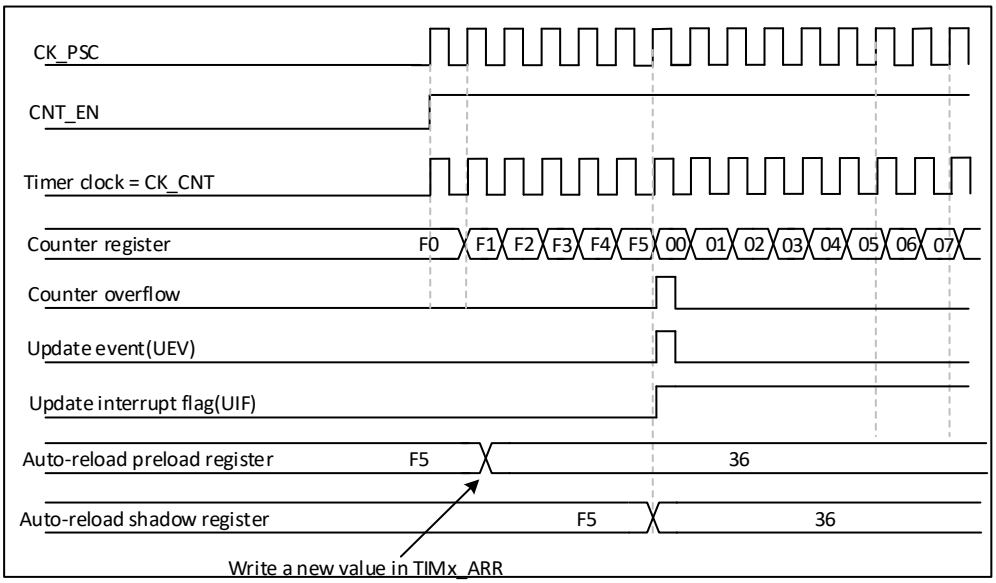


图 19-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

向下计数模式

在向下计数模式中，计数器从自动加载值(TIMx_ARR 的内容)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下溢事件。

如果使用了重复计数器，那么更新事件(UEV)需要在下溢次数达到所配置的重复计数寄存器的值加一（即 TIMx_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx_RCR=0），那么每次计数下溢都会产生更新事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清‘0’(但预分频系数不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源），通过设置 UG 位可以产生一个更新事件 UEV，但不置起 UIF 标志位(即不会产生中断或 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时,所有以下的寄存器都被更新,硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位):

- 重复计数器被重新加载为 TIMx_RCR 寄存器中的内容。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：自动加载值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 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

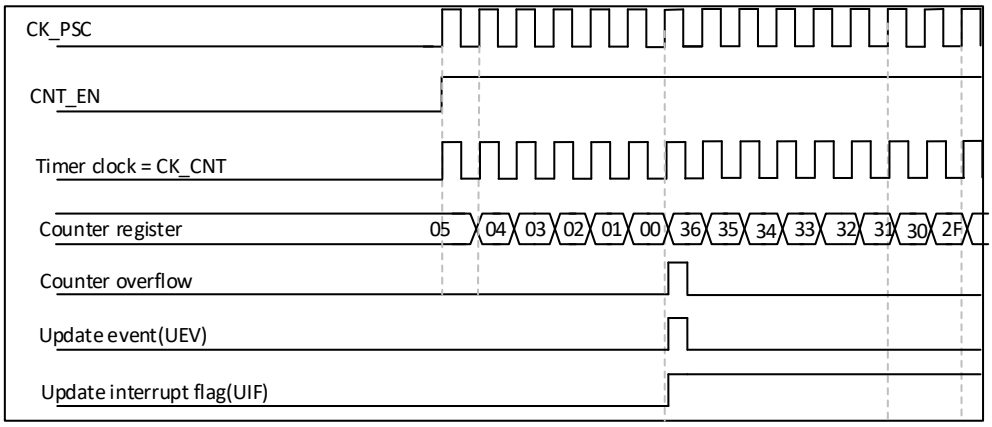


图 19-9 计数器时序图，内部时钟分频因子为 1

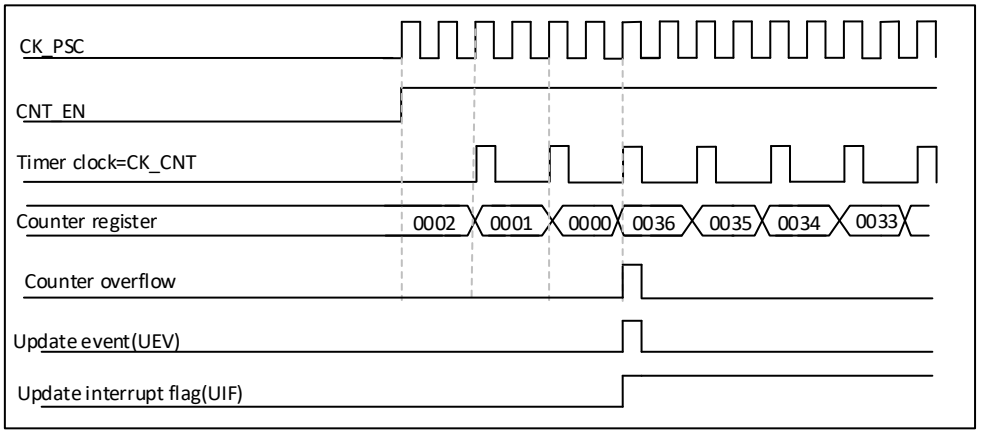


图 19-10 计数器时序图，内部时钟分频因子为 2

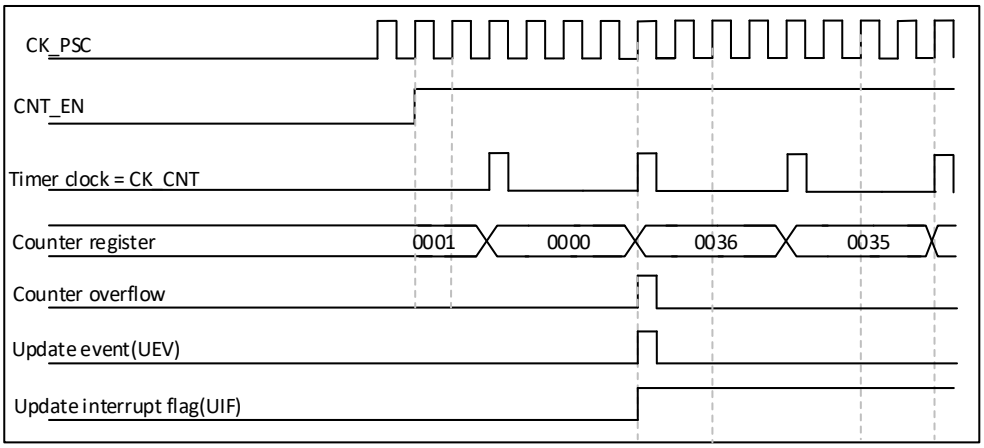


图 19-11 计数器时序图，内部时钟分频因子为 4

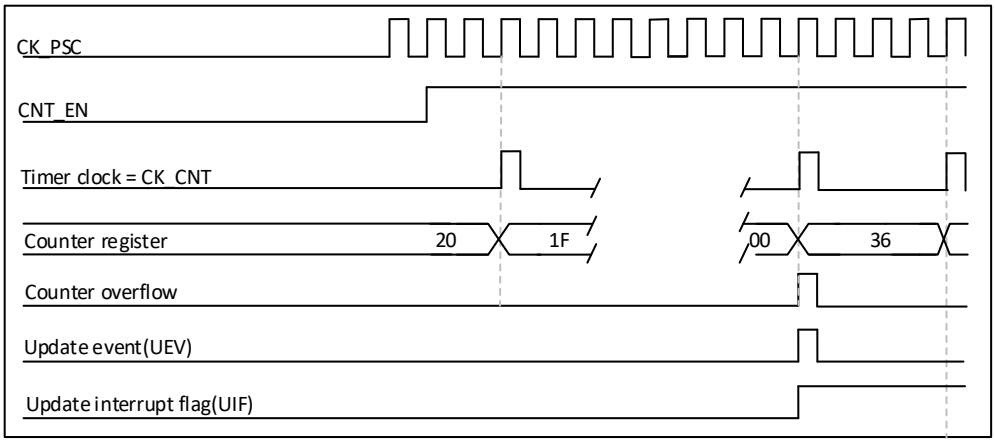


图 19-12 计数器时序图，内部时钟分频因子为 N

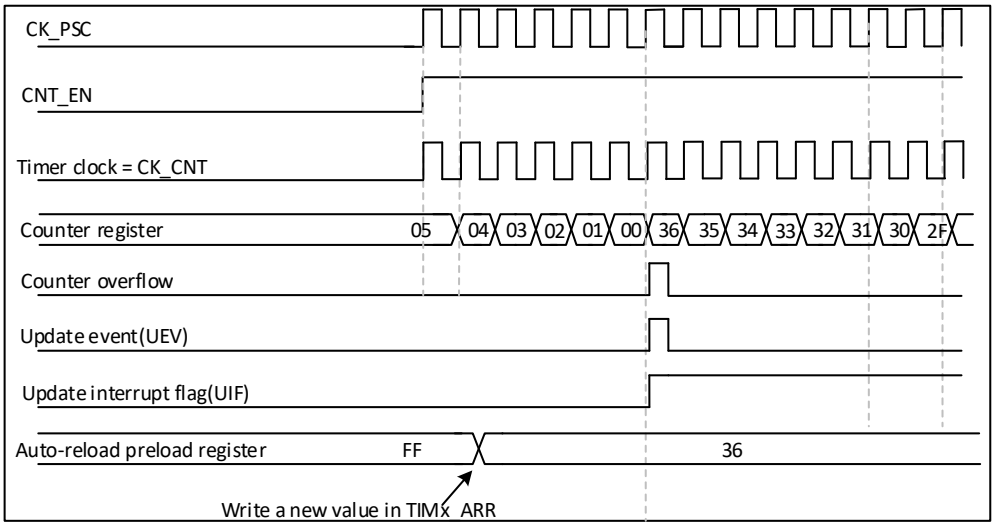


图 19-13 计数器时序图，当没有使用重复计数器时的更新事件

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(TIMx_ARR 寄存器)减 1，产生一个计数器上溢事件，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。

通过配置 TIMx_CR1 寄存器中的 CMS 位不为‘00’可以得到中央对齐模式。在通道配置为输出模式下的输出比较标志位在以下几种计数过程中会被置起：向下计数时（中央对齐模式 1，CMS=‘01’）、向上计数时（中央对齐模式 2，CMS=‘10’）、向上和向下计数时（中央对齐模式 3，CMS=‘11’）。

在此模式下，不能写入 TIMx_CR1 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 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器内部计数器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 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。虽然 UDIS 位被清为 0 之前不会产生更新事件，但是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 TIMx_CR1 寄存器中的 URS 位(选择更新请求源），通过设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 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时清除计数器，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 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置：

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。注：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：

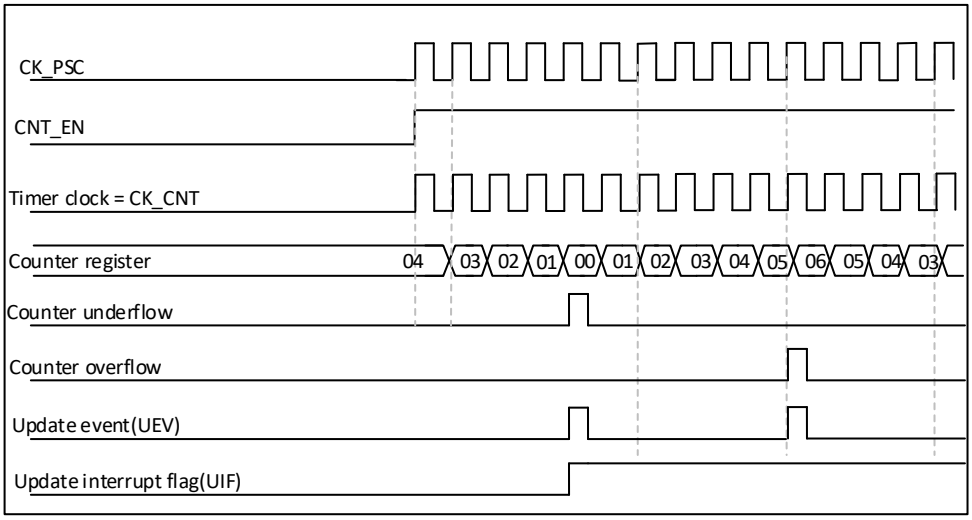


图 19-14 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

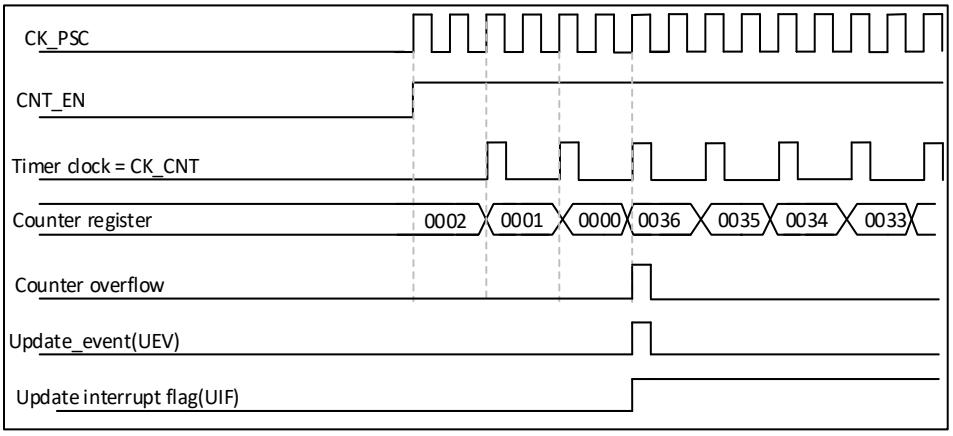


图 19-15 计数器时序图，内部时钟分频因子为 2

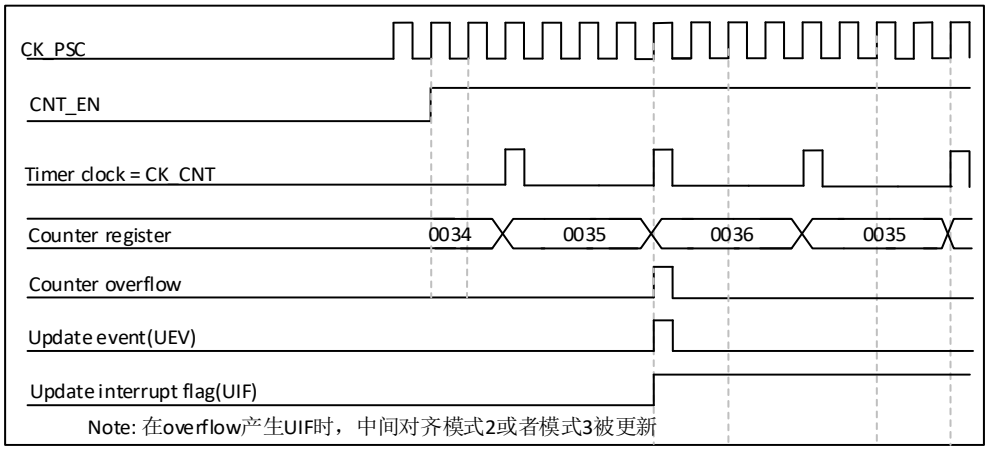


图 19-16 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

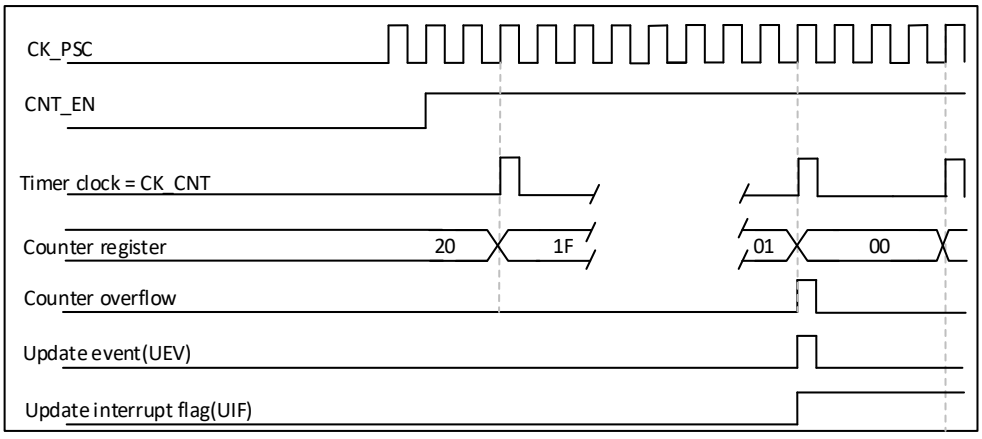


图 19-17 计数器时序图，内部时钟分频因子为 N

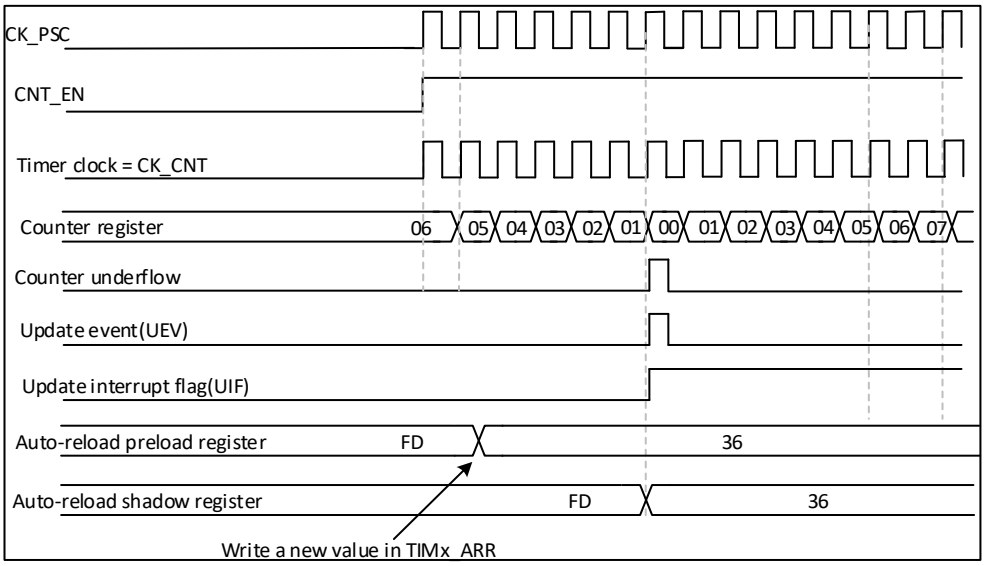


图 19-18 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

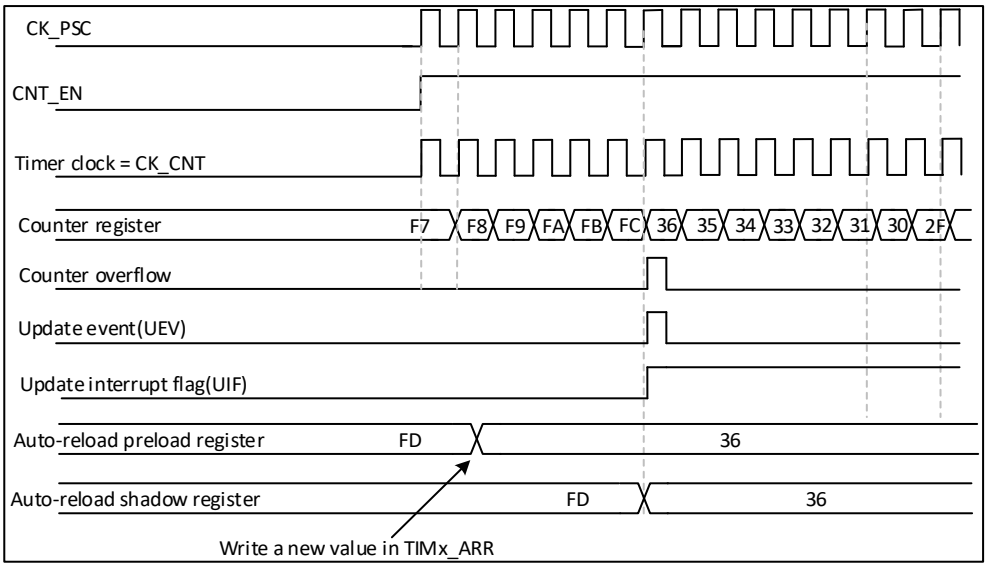


图 19-19 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

19.2.3 重复计数器

“时基单元”解释了计数器上溢/下溢时更新事件(UEV)是如何产生的，事实上它只能在重复计数器计数达到 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+1 个计数上溢或下溢时，数据才会从预装载寄存器传输到影子寄存器(TIMx_ARR 自动重载寄存器，TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx)，N 是 TIMx_RCR 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上溢时，
- 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。

中央对齐模式限制了 PWM 的最大循环周期为 128 (TIMx_RCR 为 8 位计数器) , 但它能够在每个 PWM 周期 2 次更新占空比。

重复计数器是自动重载的，重复速率由 TIMx_RCR 寄存器的值定义。当更新事件由软件产生(通过设置 TIMx_EGR 中的 UG 位)或者通过硬件的从模式控制器产生，无论重复计数器的值是多少，立即发生更新事件，并且 TIMx_RCR 寄存器中的内容被重载入到重复计数器。

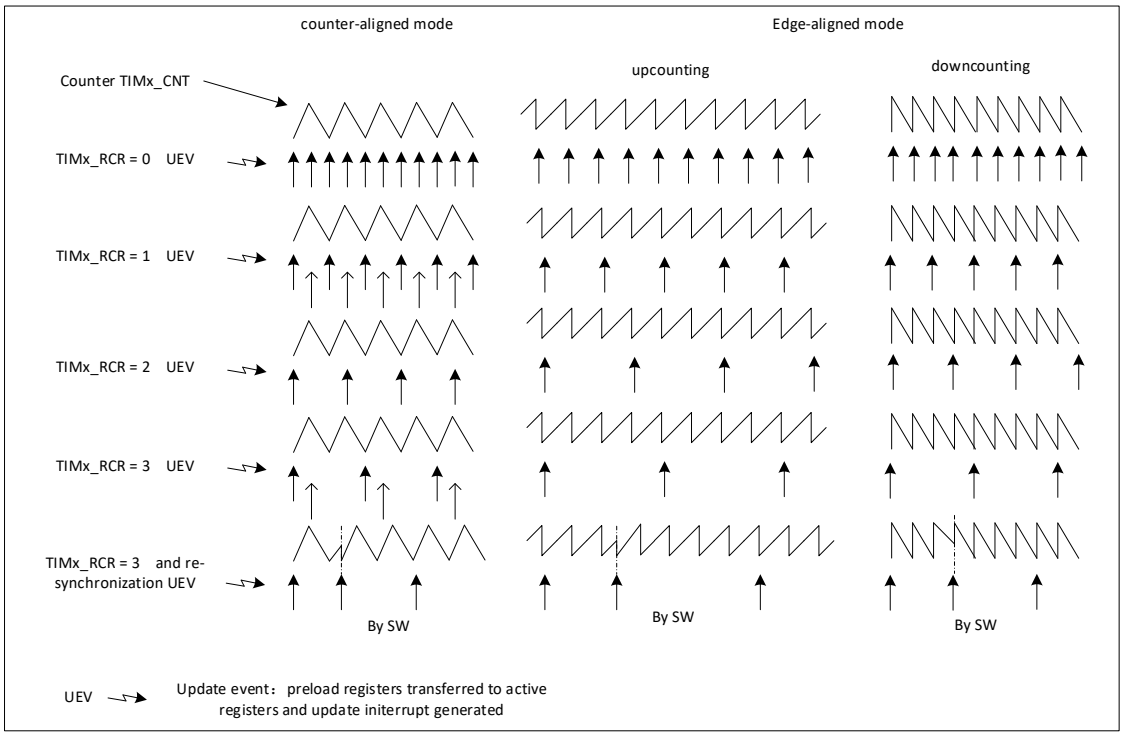


图 19-20 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

19.2.4 外部触发输入

定时器外部触发输入 tim_etr，可用于:

- 外部时钟
- 从模式的触发
- 在周期电流调节中作为 PWM 复位输入

19.2.5 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(TI1 和 TI2)
- 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。
- 编码器模式

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=0000)，则 CEN、DIR(TIMx_CR1 寄存器)和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 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。

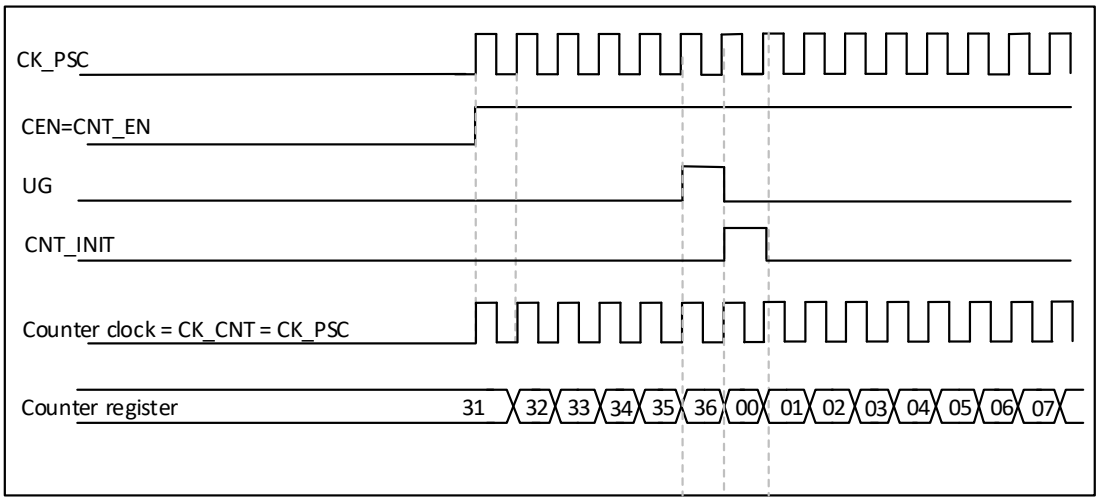


图 19-21 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=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。

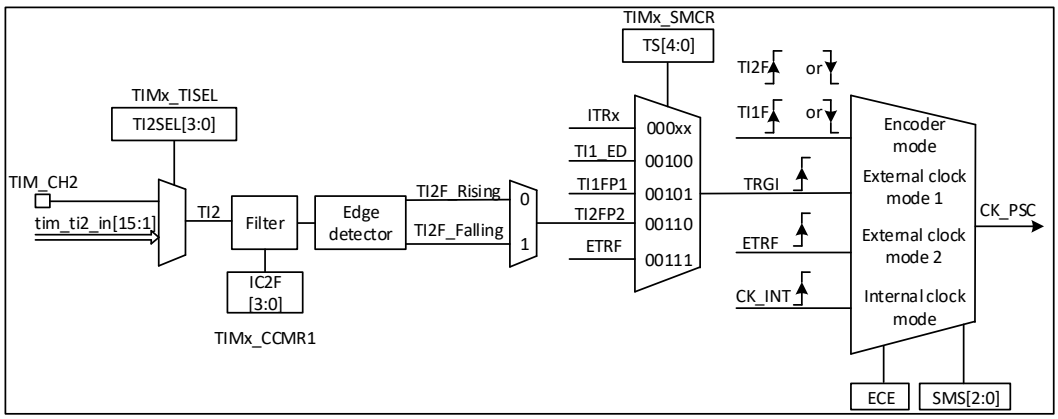


图 19-22 TI2 外部时钟连接例子

例如，要配置计数器在 TI2 输入端的上升沿向上计数，使用下列步骤：

1.配置 TIMx_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；

2.配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)；

3.配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性；

4.配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；

5.配置 TIMx_SMCR 寄存器中的 TS=00110，选定 TI2 作为触发输入源；

6.设置 TIMx_CR1 寄存器的 CEN=1，启动计数器。

注：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 TIF 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 输入端的同步电路。

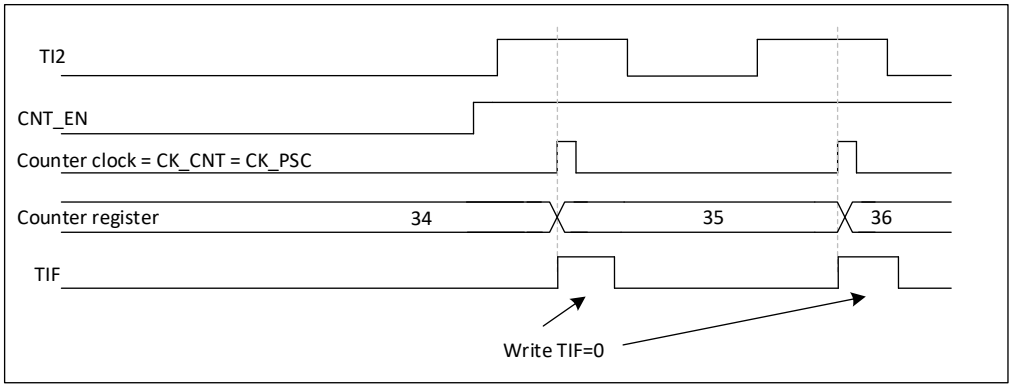


图 19-23 外部时钟模式 1 下的控制电路

外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=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：

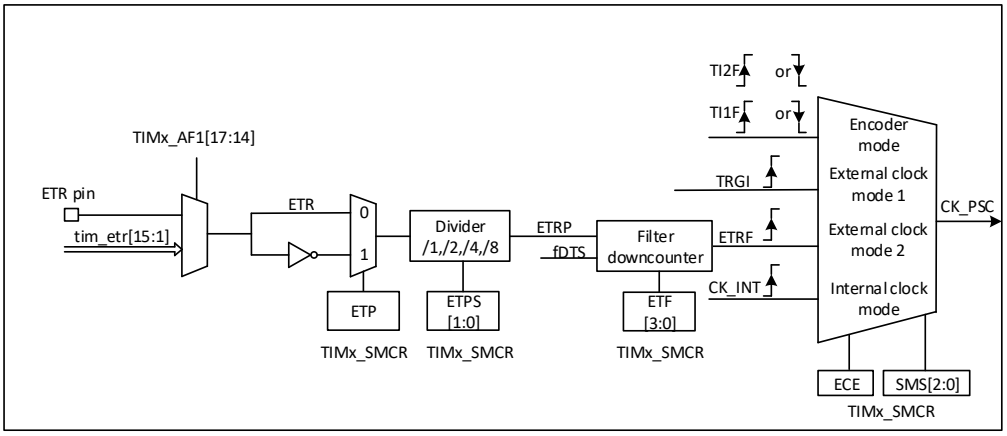


图 19-24 外部触发输入框图

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 TIMx_SMCR 寄存器中的 ETF[3:0]=0000；

2.设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01；

3.选择 ETR 输入端的上升沿，置 TIMx_SMCR 寄存器中的 ETP=0；

4.开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1；

5.启动计数器，写 TIMx_CR1 寄存器中的 CEN=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 ETRP 信号的同步电路。

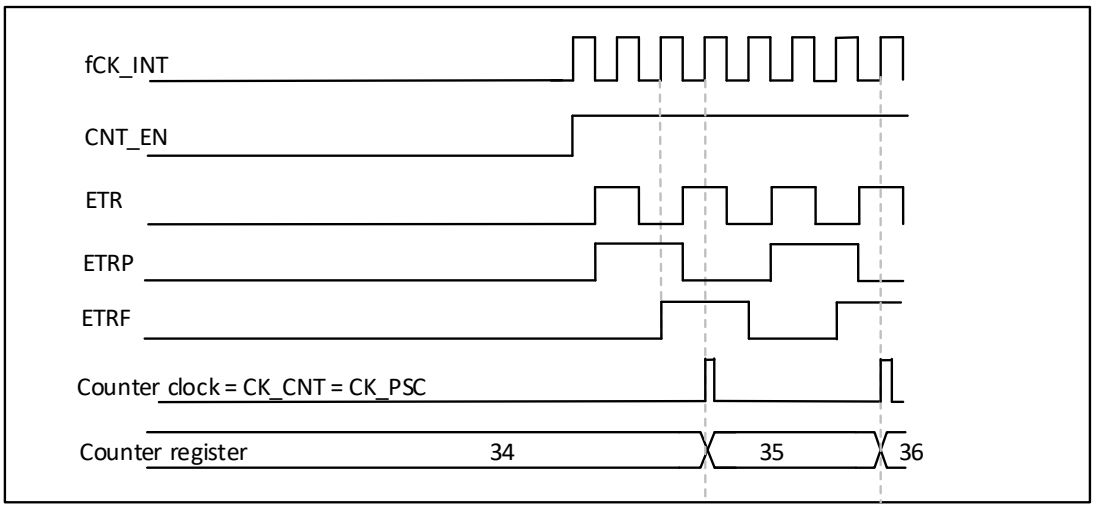


图 19-25 外部时钟模式 2 下的控制电路

19.2.6 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，通道 5 和 6 除外)，和输出部分(比较器和输出控制)。

下面几张图是捕获/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)。

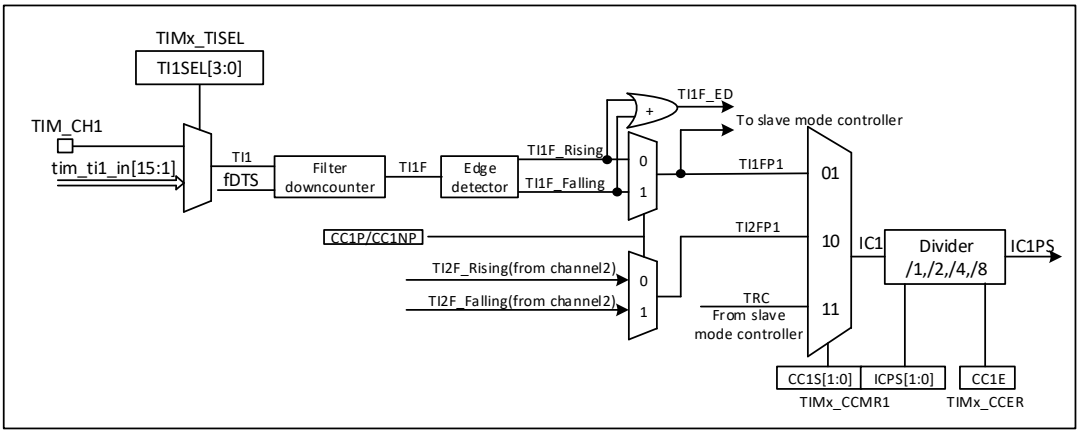


图 19-26 捕获/比较通道(如：通道 1 输入部分)

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。

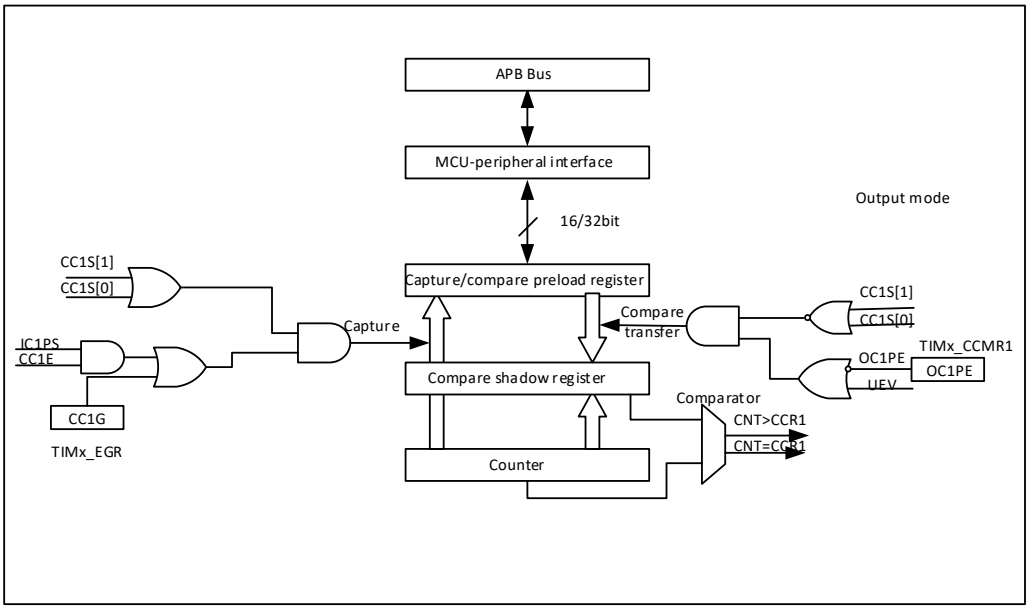


图 19-27 捕获/比较通道 1 的主电路

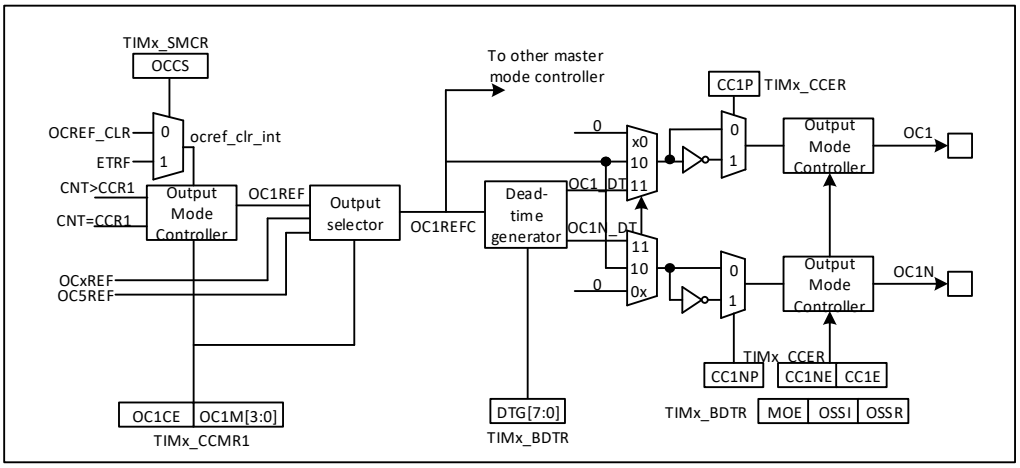


图 19-28 捕获/比较通道的输出部分(通道 1 至 4)

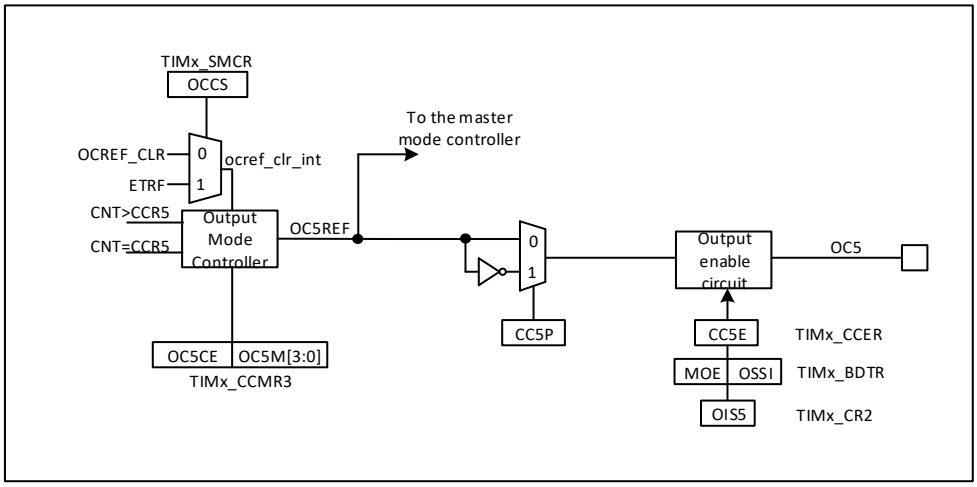


图 19-29 捕获/比较通道的输出部分(通道 5,6)

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。

19.2.7 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(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1，如果使能中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 ‘00’，通道就被配置为输入，并且 TIMx_CCR1 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 TIMx_CCMRx 寄存器中的 ICxF 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 fDTS 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 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 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 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，也可通过设置 TIMx_DIER 寄存器中的 CC1DE 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 TIMx_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

19.2.8 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，其余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，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。

- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

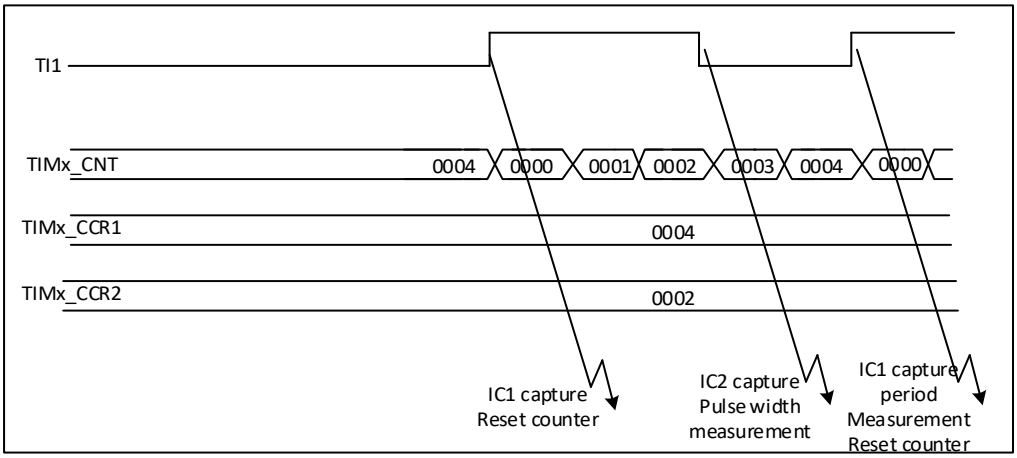


图 19-30 PWM 输入模式时序

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 TIMx_CH1/TIMx_CH2 信号。

19.2.9 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=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 TIMx_CCMRx 寄存器中相应的 OCxM=0101，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 CCxP 极性相反的信号。例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=0100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 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。

19.2.10 输出比较模式

此项功能是用来控制一个输出波形，或者表明一段给定的的时间已经到时。通道 1 到 4 可以输出，但通道 5 和 6 仅在 MCU 内部使用（例如用于产生混合波形或用于 ADC 触发）。

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(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 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(OCxM=0000)、被设置成有效电平(OCxM=0001)、被设置成无效电平(OCxM=0010)或进行翻转(OCxM=0011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

可以通过配置 TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 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 TIMx_ARR 和 TIMx_CCRx 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 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=0011

— 置 OCxPE = 0 禁用预装载寄存器

— 置 CCxP = 0 选择极性为高电平有效

— 置 CCxE = 1 使能输出

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 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(OCxPE='0'，否则 TIMx_CCRx 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。

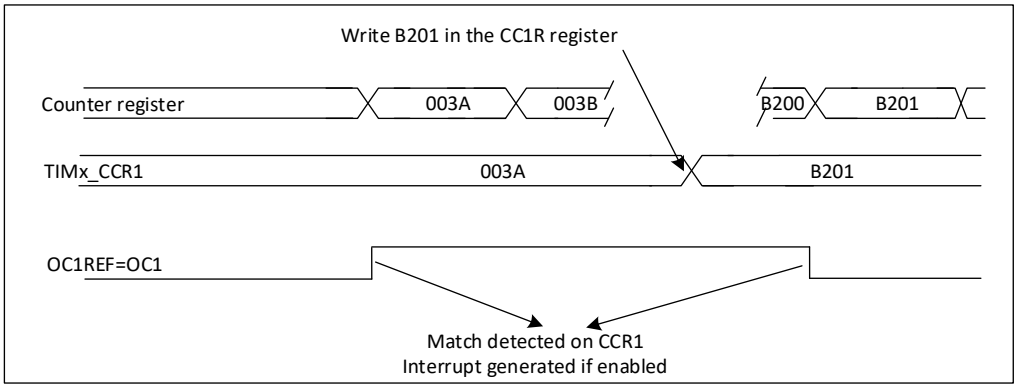


图 19-31 输出比较模式，翻转 OC1

19.2.11 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'0110'（PWM 模式 1)或'0111'(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 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 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，用户必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 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(TIMx_CCER 和 TIMx_BDTR 寄存器中)CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

根据 TIMx_CR1 寄存器中 CMS 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 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

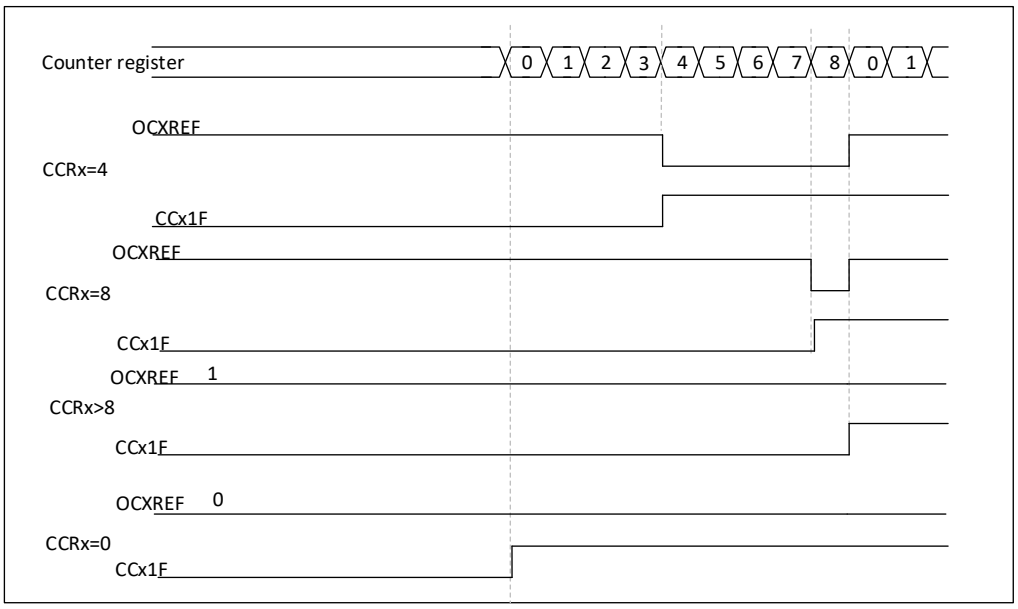


图 19-32 边沿对齐的 PWM 波形(ARR=8)

- 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 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 TIMx_CR1 寄存器中的 CMS 位不为‘00’时为中央对齐模式(CMS 位的所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 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。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器的 CMS=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。

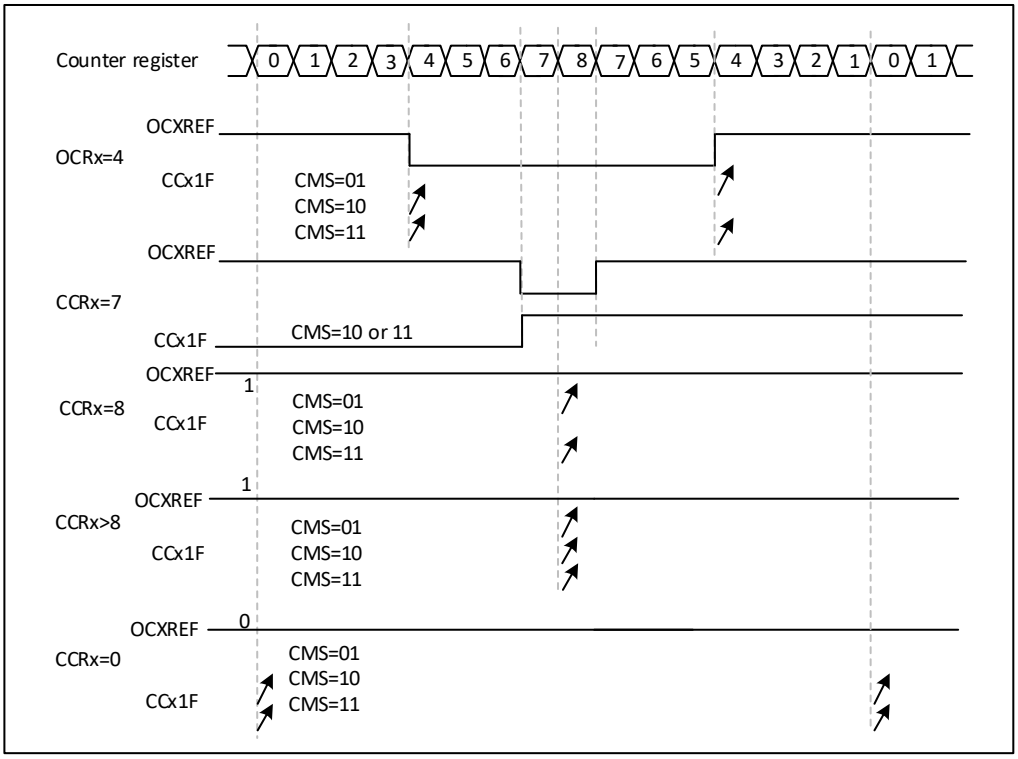


图 19-33 中央对齐的 PWM 波形(APR=8)

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 TIMx_CR1 寄存器中 DIR 位的当前值。此外，不要通过软件同时修改 DIR 和 CMS 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(TIMx_CNT>TIMx_ARR)，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 TIMx_ARR 的值写入计数器，方向被更新，但不会产生更新事件 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 TIMx_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

抖动模式

可以使能 TIMx_CR1 寄存器的 DITHEN 位来打开抖动模式提高 PWM 模式的有效分辨率。可以应用于 CCR（提高占空比分辨率）和 ARR（提高 PWM 频率的分辨率）

操作原理是使实际的 CCR（或 ARR）的值在 16 个连续的 PWM 周期内进行轻微的变化（增加或不增加一个定时器周期），如何变化可以通过预定义设置。计算平均占空比时就可以实现分辨率提高 16 倍。

下图为 4 个连续 PWM 周期抖动原理

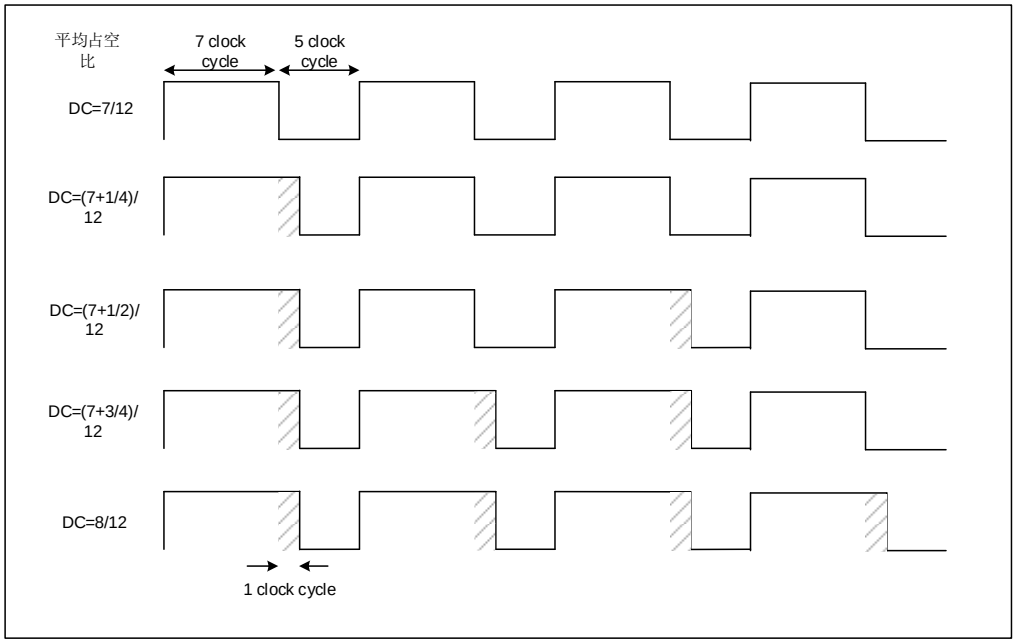


图 19-34 抖动原理

当抖动模式使能后，寄存器的值会自动如下变化：

- 最低的 4 bits 是用于提高分辨率的编码(小数部分)
- 高位被左移到 19:4 作为基值进行编码（整数部分）

注：当抖动模式打开或关闭时，ARR 和 CCR 的值会自动更新（例如，DITHEN=0 时 ARR=0x05，那么 DITHEN=1 时 ARR=0x50），复位 DITHEN 位时必须遵循以下步骤

1. CEN 和 ARPE 位必须复位
2. ARR[3:0]位必须复位
3. DITHEN 位必须复位
4. CCIF 标志位必须清除
5. CEN 位可以置位（可设置 ARPE=1）

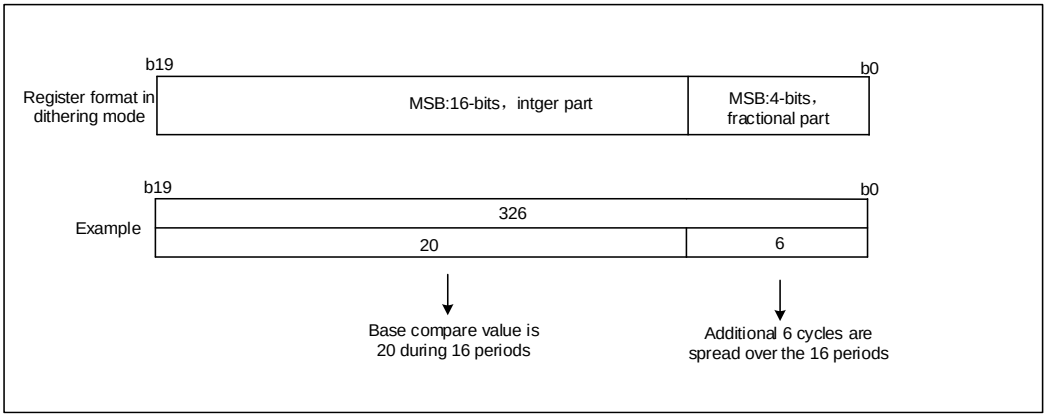


图 19-35 抖动模式中数据映射与寄存器编码

最小的频率计算公式如下：

$$\text{根据分辨率} = \frac{\text{定时器频率}}{\text{PWM 频率}} \qquad \text{推出：最小 PWM 频率} = \frac{\text{定时器频率}}{\text{最大分辨率}}$$

$$\text{无抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65536}$$

$$\text{有抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65535+\frac{15}{16}}$$

注：抖动模式里 TIMx_ARR 和 TIMxCCRy 的最大值被限定到 0xFFFEF（对应于整数部分的 65534 和抖动部分的 15），超出 0xFFFEF 会存在溢出情况，即 arr=0XFFFF+1=0。

下图显示的是抖动模式下，无论 PWM 频率多少都可以提高 PWM 的分辨率

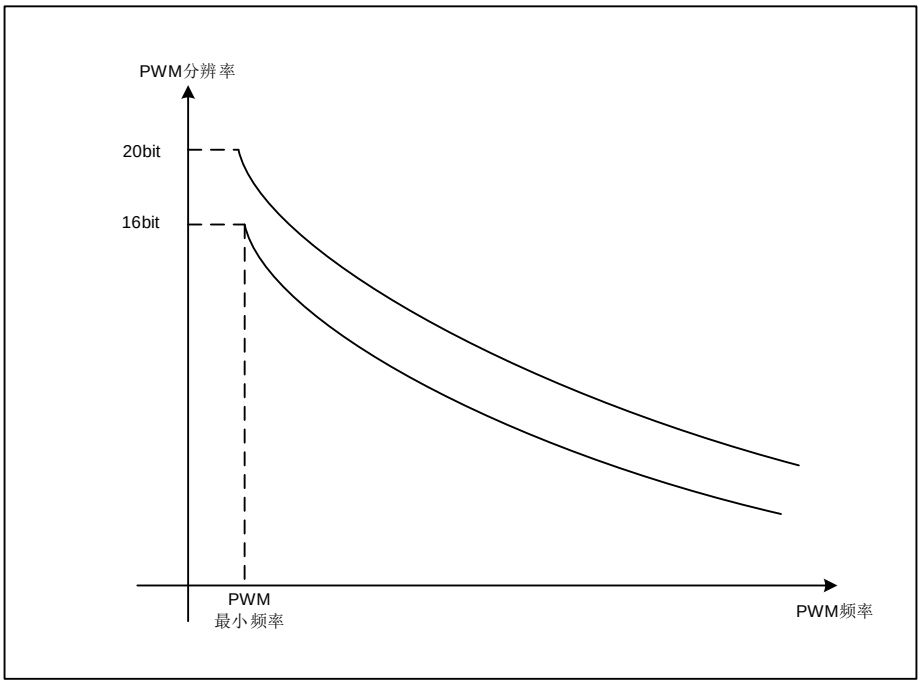


图 19-36 PWM 分辨率 VS 频率

连续 16 个周期内占空比和周期变化情况如下图：

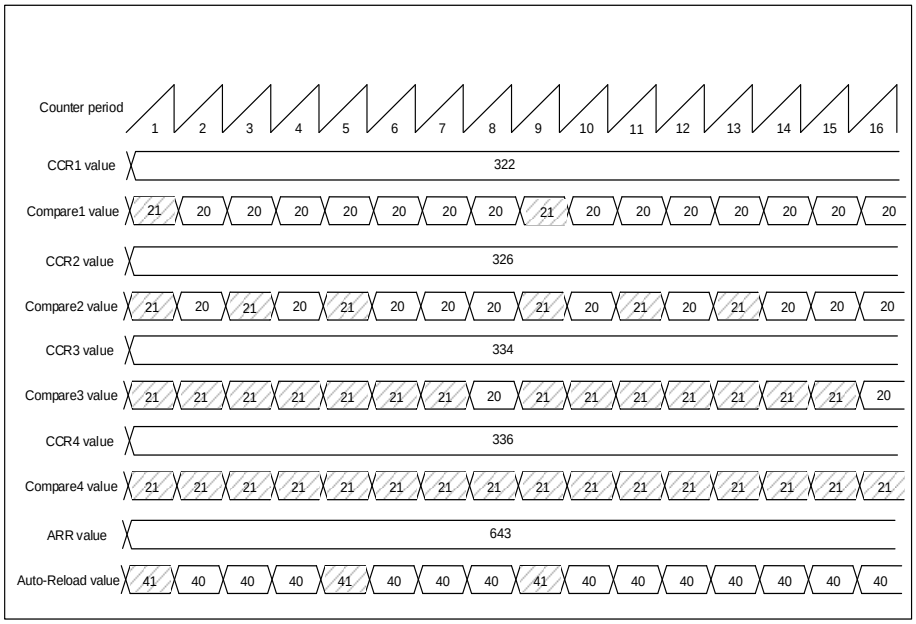


图 19-37 PWM 抖动模式

自动重新加载和比较值增量按照如下表描述的特定模式分布，抖动序列的增量分布尽可能地均匀，最大限度减少整体波纹

表 19-1 CCR 和 ARR 寄存器抖动变化

LSB value	PWM 周期															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000																
0001	+1															
0010	+1								+1							
0011	+1				+1				+1							
0100	+1				+1				+1				+1			
0101	+1		+1		+1				+1				+1			
0110	+1		+1		+1				+1		+1		+1			
0111	+1		+1		+1		+1		+1		+1		+1			
1000	+1		+1		+1		+1		+1		+1		+1		+1	
1001	+1	+1	+1		+1		+1		+1		+1		+1		+1	
1010	+1	+1	+1		+1		+1		+1	+1	+1		+1		+1	
1011	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1		+1	
1100	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1	+1	+1	
1101	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1		+1	+1	+1	
1110	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1	+1	+1	+1	+1	
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	

抖动模式也可以应用在中间对齐 PWM 模式中(CMS!=00)，下图是考虑了向上向下计数时，抖动模式应用于连续 8 个周期的情况。

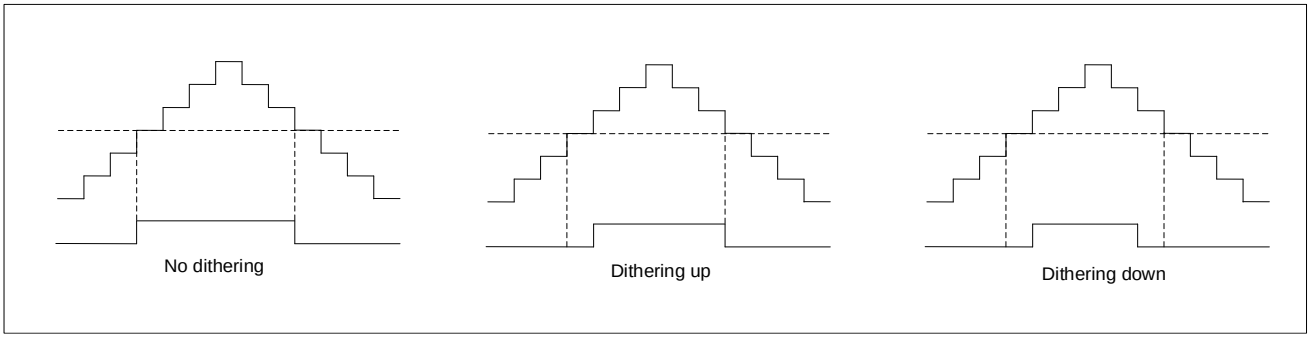


图 19-38 中间对齐 PWM 模式中抖动对占空比的影响

表 19-1 中间对齐 PWM 模式中 CCR 寄存器抖动变化

LSB value	PWM 周期															
	1		2		3		4		5		6		7		8	
	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN
0000																
0001	+1															
0010	+1								+1							
0011	+1				+1				+1							
0100	+1				+1				+1				+1			
0101	+1		+1		+1				+1				+1			
0110	+1		+1		+1				+1		+1		+1			
0111	+1		+1		+1		+1		+1		+1		+1			
1000	+1		+1		+1		+1		+1		+1		+1		+1	
1001	+1	+1	+1		+1		+1		+1		+1		+1		+1	
1010	+1	+1	+1		+1		+1		+1	+1	+1		+1		+1	
1011	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1		+1	
1100	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1	+1	+1	
1101	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1		+1	+1	+1	
1110	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1	+1	+1	+1	+1	
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	

19.2.12非对称 PWM 模式

非对称模式允许两个中央对齐的 PWM 信号产生可编程的相移。当频率由 TIMx_ARR 决定、占空比和相移由一对 TIMx_CCRx 寄存器决定时。不对称模式允许使用可编程的相移生成两路中心对齐的 PWM 信号。一个 TIMx_CCRx 寄存器在向上计数阶段控制，另一个在向下计数阶段控制，以便每半个 PWM 周期都可调整 PWM

- OC1REFC（or OC2REFC）由 TIMx_CCR1 和 TIMx_CCR2 控制
- OC3REFC（or OC4REFC）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过向 OCxM 写 1110(不对称 PWM 模式 1）和 OCxM 写 1111(不对称 PWM 模式 2)，可以在双通道（每对 CCR 寄存器控制一个 OCx 输出）上独立选择不对称的 PWM 模式

注意：出于兼容的原因，OCxM[3:0]字段分为两部分，最高位和低 3 位不连续。

当给定的通道被用作不对称通道时,其相邻通道(通道 1 与通道 2 相邻)也可使用。例如,如果 OC1REFC 由通道 1（不对称 PWM 模式 1）产生，则通道 2 也可以输出 OC2REF，或者通过不对称 PWM 模式 1 输出 OC2REFC

下图表示使用不对称 PWM 模式生成信号的示例（通道 1 到 4 被配置为不对称 PWM 模式 2），与死区时间生成器一起，控制全桥 DC-DC 相移逆变器。

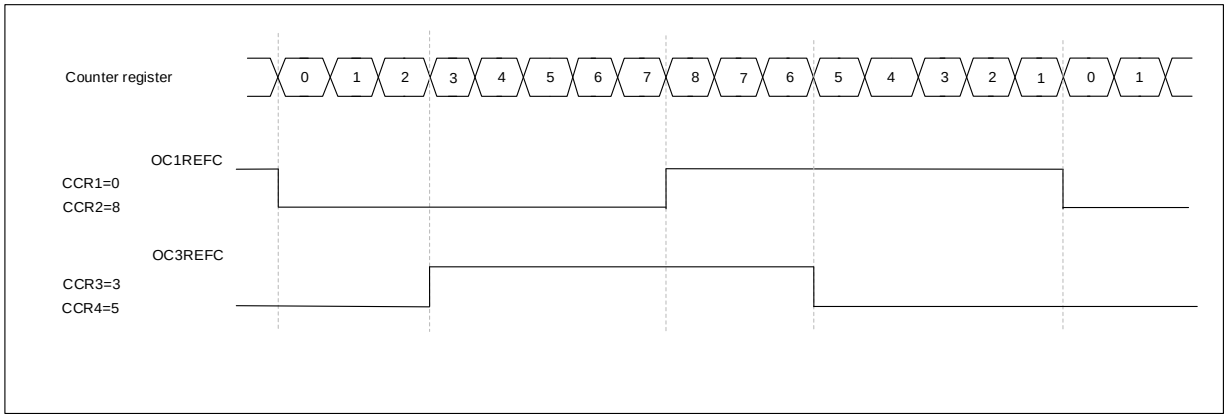


图 19-39 2 路 50%占空比相移 PWM

19.2.13组合 PWM 模式

组合 PWM 模式允许通过对各脉冲间进行可编程的延迟和相移来产生边沿或中央对齐 PWM 信号。频率由 ARR 决定，占空比和延迟由两个 CCR 决定。产生的信号 OCxREFC 由两个参考 PWM 间进行与/或逻辑产生

- OC1REFC（or OC2REFC）由 TIMx_CCR1 和 TIMx_CCR2 控制
- OC3REFC（or OC4REFC）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过向 OCxM 写 1100(组合 PWM 模式 1，逻辑或输出）和 OCxM 写 1101(组合 PWM 模式 2，逻辑与输出)，可以在双通道（每对 CCR 寄存器控制一个 OCx 输出）上独立选择不对称的 PWM 模式

当给定的通道被用作组合 PWM 通道，其互补通道必须被配置为相反的 PWM 模式(例如，一路为组合 PWM 模式 1、一路为组合 PWM 模式 2)

下图为组合模式生成信号示例，包括

- 通道 1 为组合 PWM 模式 2
- 通道 2 为 PWM 模式 1

- 通道 3 为组合 PWM 模式 2
- 通道 4 为 PWM 模式 1

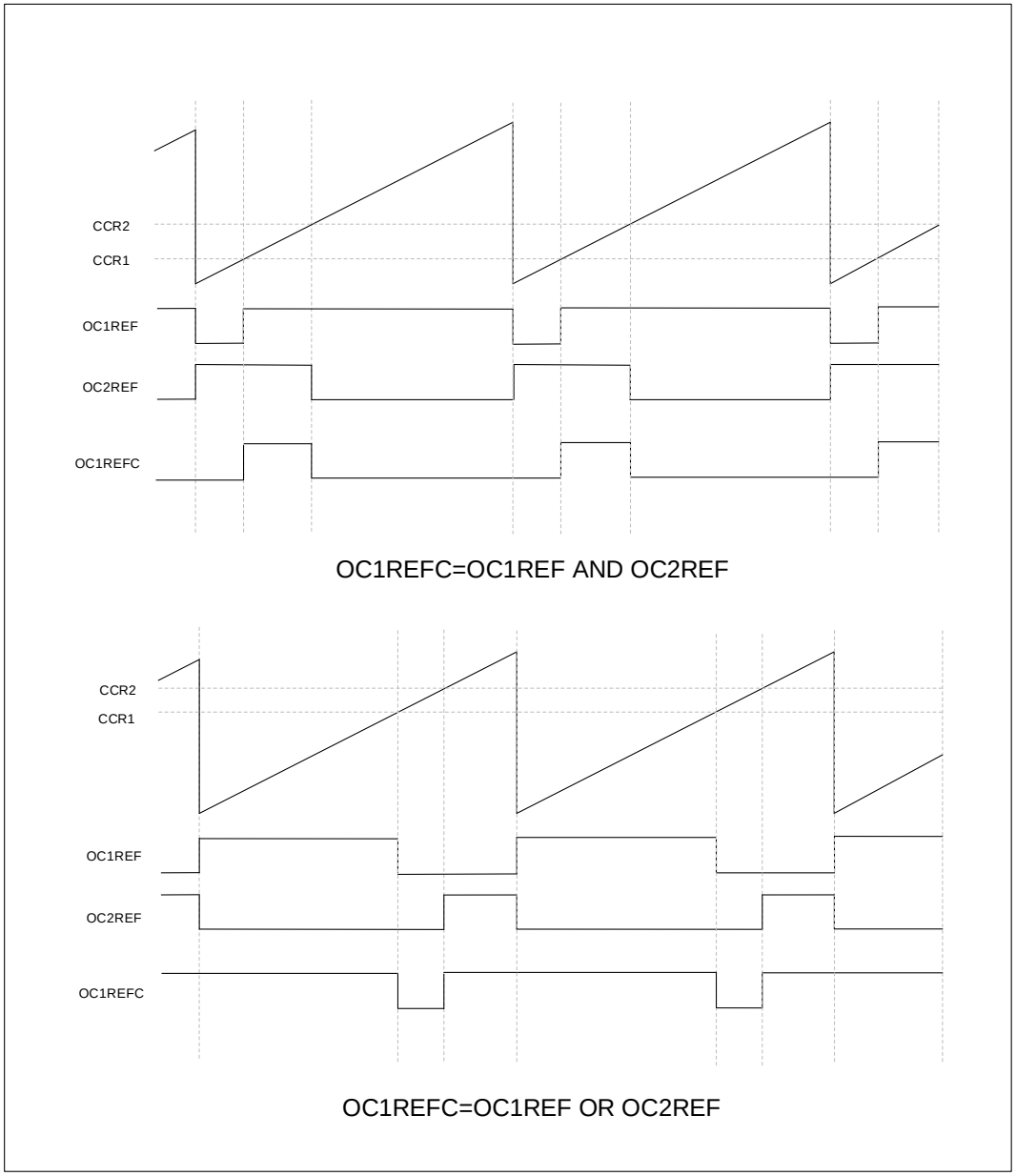


图 19-40 通道 1 和通道 3 的组合 PWM 模式

19.2.14三相 PWM 组合模式

三相组合 PWM 模式允许在脉冲中间通过单个编程信号与逻辑生成三路中央对齐的 PWM 信号。OC5REF 用于定义得到的组合信号。TIM_CCR5 的 GC5C1/ GC5C2/ GC5C3 用来选择 OC5REF 通过哪个参考信号进行组合。最终的输出信号 OCxREFC 由两个参考的 PWM 信号进行与逻辑产生

- GC5C1=1，OC1REFC 由 CCR1 和 CCR5 控制
- GC5C2=1，OC2REFC 由 CCR2 和 CCR5 控制
- GC5C3=1，OC3REFC 由 CCR3 和 CCR5 控制
- 三相组合 PWM 模式可以通过 GC5C1/ GC5C2/ GC5C3 独立选择通道 1 到 3

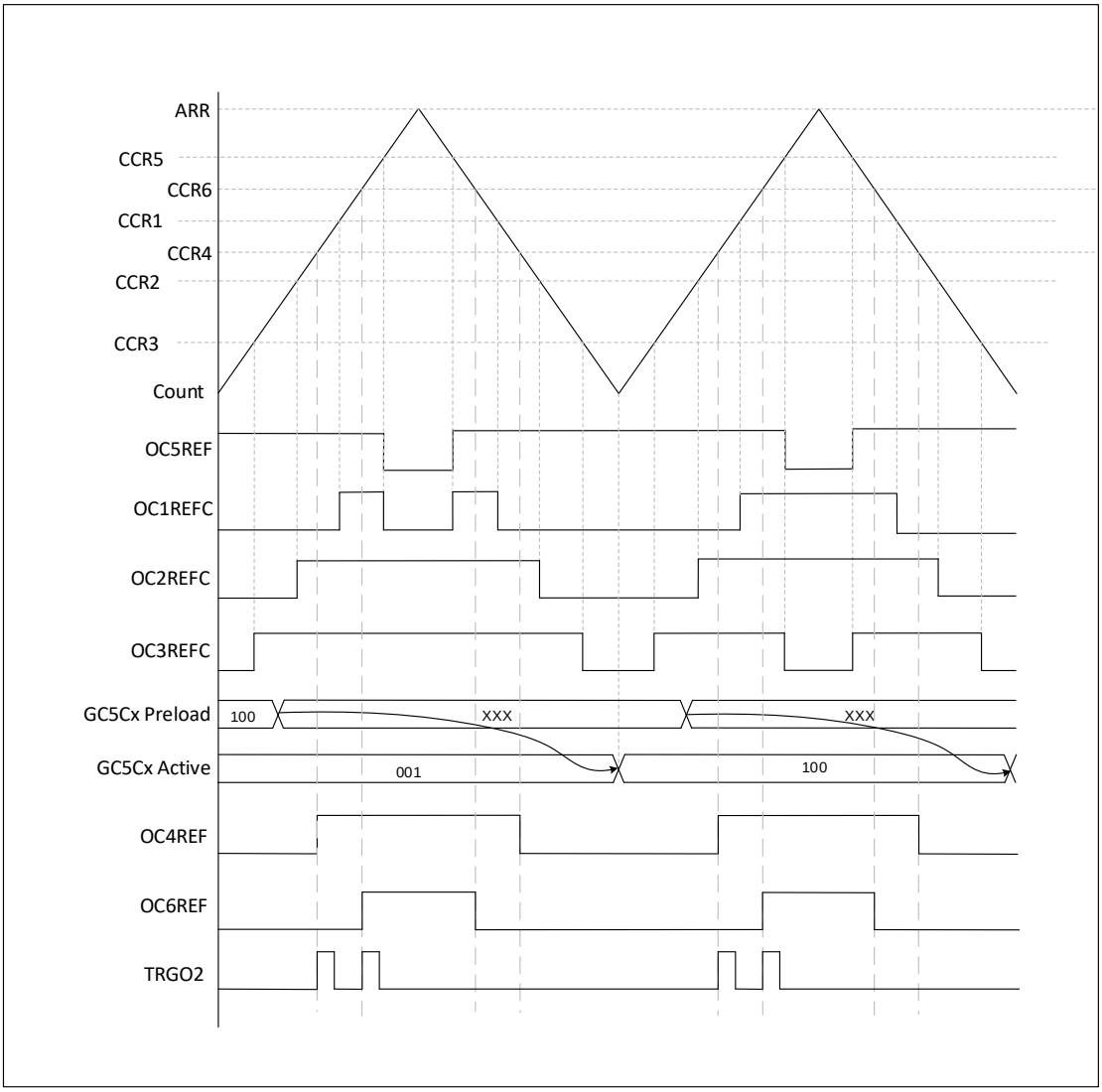


图 19-41 三相 PWM 组合模式

Trgo2 波形显示了在给定的三相 PWM 信号中 ADC 如何同步的

19.2.15互补输出和死区插入

高级控制定时器(TIM1)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 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 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：TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见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 IDLE 状态时(MOE 下降到 0)死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。通过配置 TIMx_BDTR 寄存器中的 DTG[7:0]位，可以控制所有通道的死区发生器。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输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

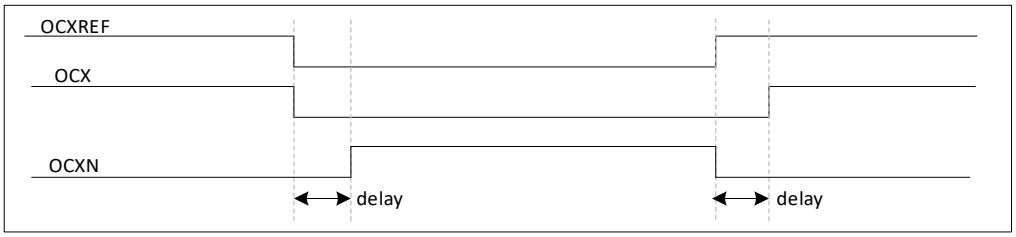


图 19-42 带死区插入的互补输出

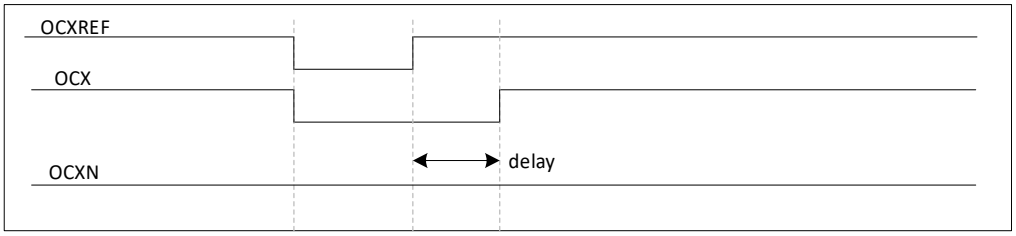


图 19-43 死区波形延迟大于负脉冲

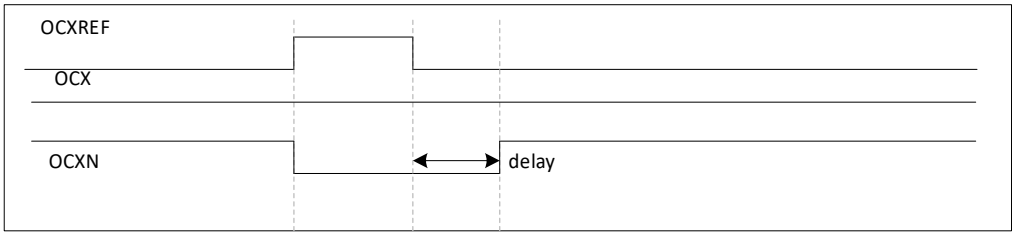


图 19-44 死区波形延迟大于正脉冲

当 DTAE=1，上升沿死区时间由 TIMx_BDTR 的 DTG[7:0]位配置，下降沿死区时间由 TIMx_DTR2 的 DTGF[7:0]位配置。DTAE 的写入需要在计数器使能前，当 CEN=1 时，不允许修改 DTAE。

可以通过预加载的机制在 PWM 工作过程中修改死区时间，当 TIMX_DTR2 的 DTPE 被置位时死区时间配置寄存器 DTG[7:0]和 DTGF[7:0]被预加载，在下一个更新事件时预加载的数据被加载到缓存寄存器中。

注:如果在计数器被使能时，DTPE 位被置位，则自最后一次更新以来的写的新数据都会被丢弃，而使用之前的值

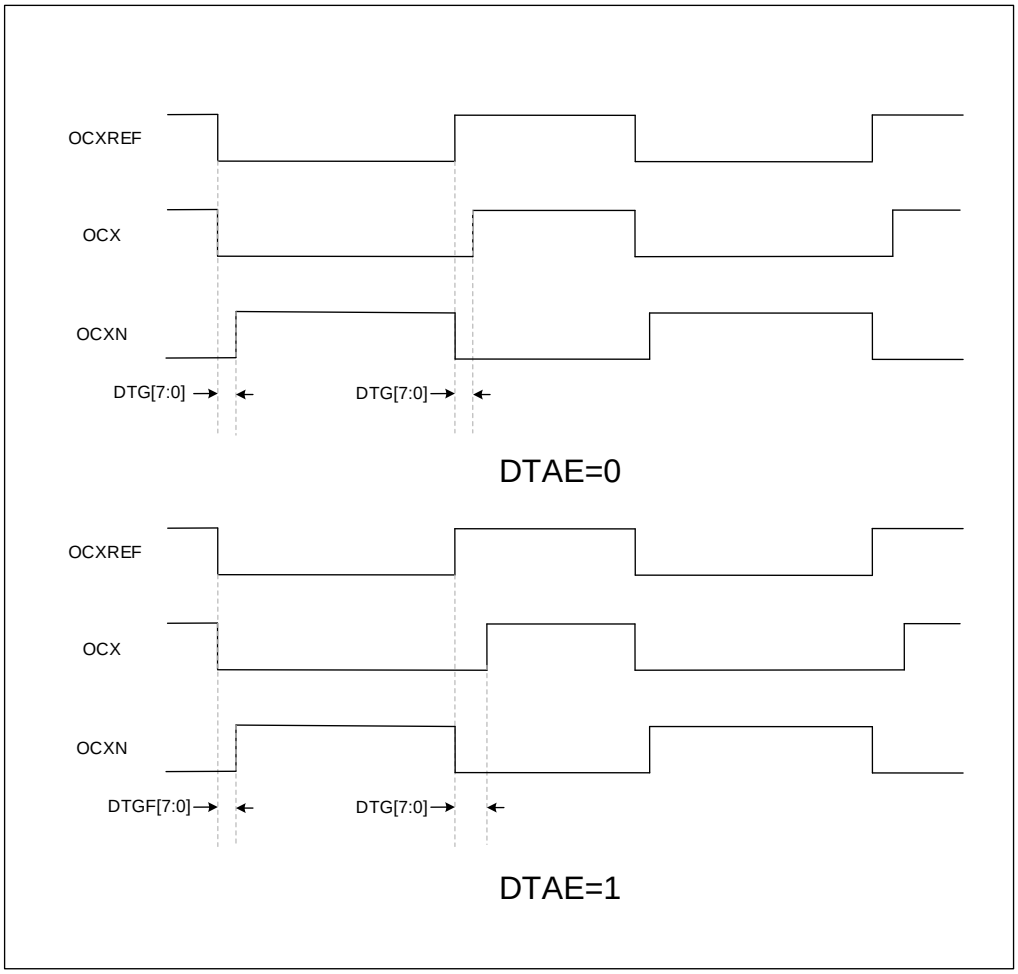


图 19-45 非对称死区时间

19.2.15.1 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 TIMx_CCER 寄存器的 CCxE 和 CCxNE 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：当只使能 OCxN(CCxE=0, CCxNE=1)时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0,则 OCxN=OCxREF。另一方面,当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1),当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

19.2.16 使用刹车功能

刹车功能的目的是保护由 TIMER 输出的 PWM 驱动的电源开关。两路刹车的输入通常是连接到三相逆变器和功率级芯片的故障输出。当刹车信号有效时，会立即关断 PWM 的所有输出，并将它们强制到预先设置好的一个安全的电平状态。部分 MCU 内部的事件也可以作为触发信号来关闭输出。

刹车事件有两路通道，通道 1 包括了系统级的故障(clock failure、ECC、奇偶校验等)和应用层的故障（通过输入引脚或者内置的比较器），并且可以在一个死区区间后强制输出到预先设置好的电平（无论工作还是空闲）。通道 2 仅包含一些应用层的故障，并且强制输出为无效状态。

在刹车过程中输出使能信号和输出电平由以下多个 bit 控制

- TIMx_BDTR 的 MOE 位可以通过软件或 2 路刹车事件复位来实现使能和关闭输出
- TIMx_BDTR 的 OSSI 位用于定义当处于空闲状态时 timer 是否还控制输出，或者释放对 GPIO 控制器的控制（高阻状态）
- TIMx_CR2 的 OISx 和 OISxN 位用于无论工作还是空闲状态下都将输出设置为无效电平，OCx 和 OCxN 输出在同一时刻不能同时设置为有效电平，无论 OISx 和 OISxN 为何值。

当从复位中退出时，刹车功能被关闭并且 MOE 为 0。刹车功能的开启可通过使能 TIMxBDTR 的 BKE 和 BK2E 位。刹车输入的极性可通过 BKP 和 BK2P 配置。BKEx 和 BKPx 可同时修改。当写入 BKE 和 BKP 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 MOE 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 TIMx_BDTR 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 MOE=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。

刹车通道 1 的源；

- 外部源可连接到其中一个 TIMx_BKIN 引脚上（选择 GPIO 和配置寄存器），再加上极性选择和滤波
- 内部源包含 2 部分
  - 来自刹车比较器信号（tim_brk_cmpx）
  - 来自系统刹车请求

刹车通道 2 的源；

- 外部源可连接到其中一个 TIMx_BKIN2 引脚上（选择 GPIO 和配置寄存器），极性选择和滤波
- 内部源来自刹车比较器信号（tim_brk2_cmpx）

刹车事件也可以由软件生成，可置位 TIMx_EGR 的 BG 和 B2G 位，所有刹车源均或起来作为 tim_brk 和 tim_brk2 的刹车输入。如下图：

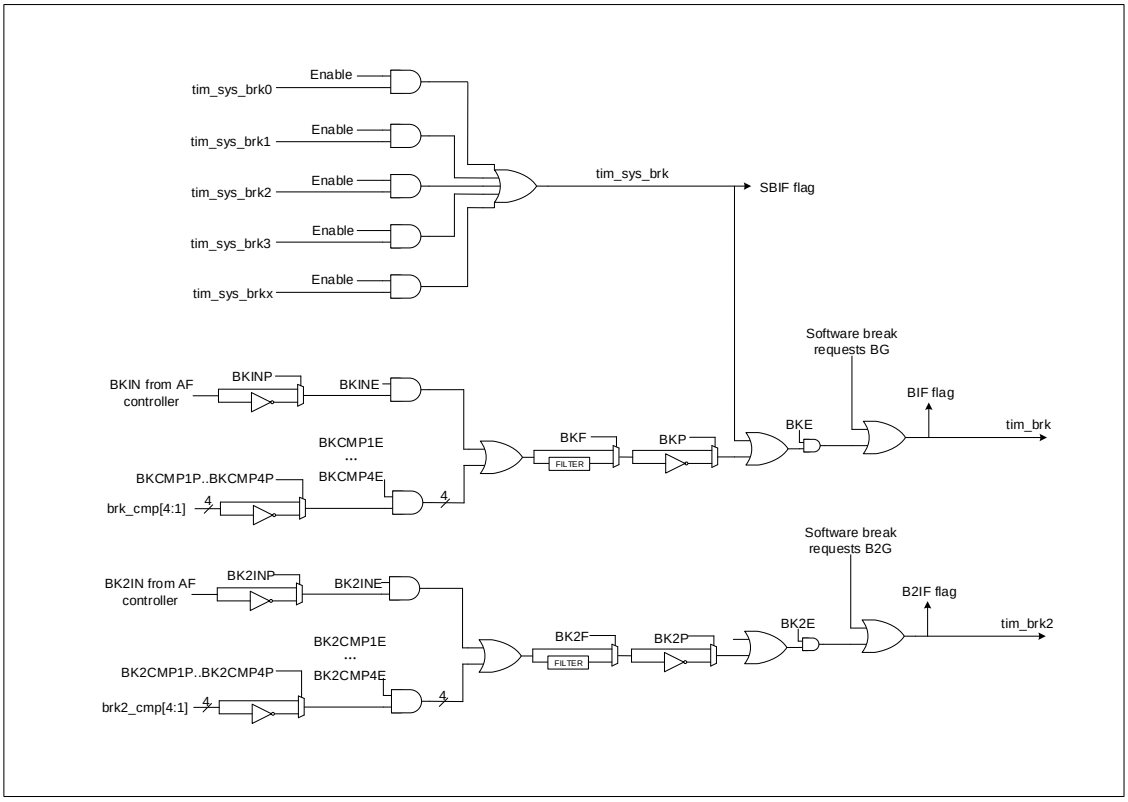


图 19-46 刹车和刹车 2 电路

注意：只有没有滤波的时候才能保证是异步的操作，如果滤波开启了，必须使用一个故障保护时钟模式来保证刹车中断事件得到处理（例如使用内部 PLL 或 CSS，无滤波）。

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者释放对 GPIO 控制器的控制(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放输出控制，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 OISx 和 OISxN 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，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个时钟周期)。

– 如果 OSSI=0，定时器释放输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。

- 刹车状态标志会被拉高（TIMx_SR 的 SBIF,BIF,B2IF）。如果设置了 TIMx_DIER 寄存器中的 BIE 位，当刹车状态标志(TIMx_SR 寄存器中的 BIF 位)为 ‘1’ 时，则产生一个中断。如果设置了 TIMx_DIER 寄存器中的 TDE 位，则产生一个 DMA 请求。
- 如果设置了 TIMx_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 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：当 AOE 为 1 时如果 MOE 被 CPU 复位了，输出会进入空闲状态，并被强制到无效电平或者高阻（取决于 OSSI 的值），如果 MOE 和 AOE 均被 CPU 复位，输出将进入无效状态并且由 OISx 位来驱动输出电平。

注：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 MOE。同时，状态标志 BIF 和 B2IF 不能被清除。

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(死区持续时间，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性)。用户可以通过设置 TIMx_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，参看刹车和死区寄存器(TIMx_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

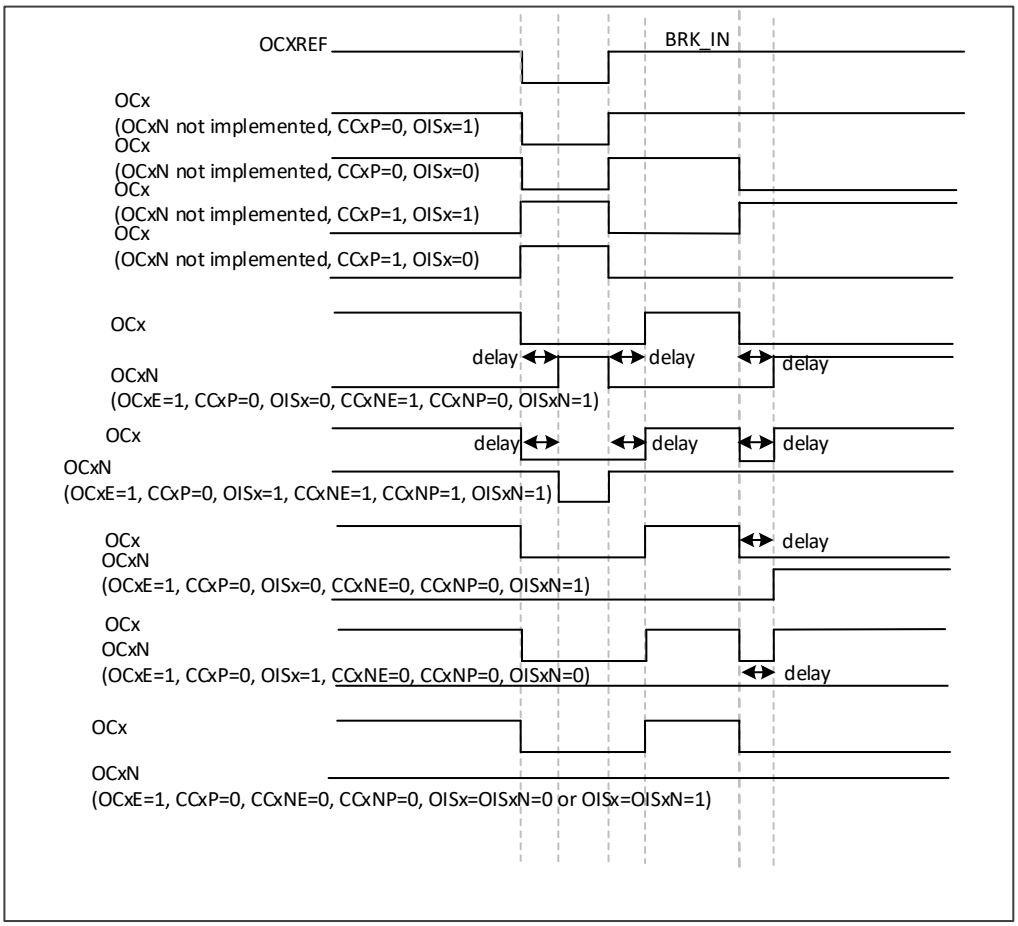


图 19-47 响应刹车的输出

两路刹车输入信号在 timer 的输出端具有不同的行为

tim_brk 既可以将输出置为无效状态，也可以将输出强制到预先配置的安全状态

tim_brk2 只能将输出置为无效状态。

tim_brk 的优先级比 tim_brk2 更高，由下图描述：

注：tim_brk2 仅在 OSSR=OSSI=1 时使用（输出不是高阻）

表 19-2 刹车和刹车 2 输入时输出行为

BRK	BRK2	TIM 输出状态	典型情况	
			OCxN 输出（低侧开关）	OCx 输出（高侧开关）
Active	X	-输出无效且强制输出状态（死区后） -输出禁止如果 OSSI=0	死区插入后开启	OFF
Inactive	Active	无效	OFF	OFF

下图描述的是当 tim_brk 和 tim_brk2 有效时 OCx 和 OCxN 的输出行为。极性都为高有效（TIMx_CCER 寄存器的 CCxP=CCxNP=0）

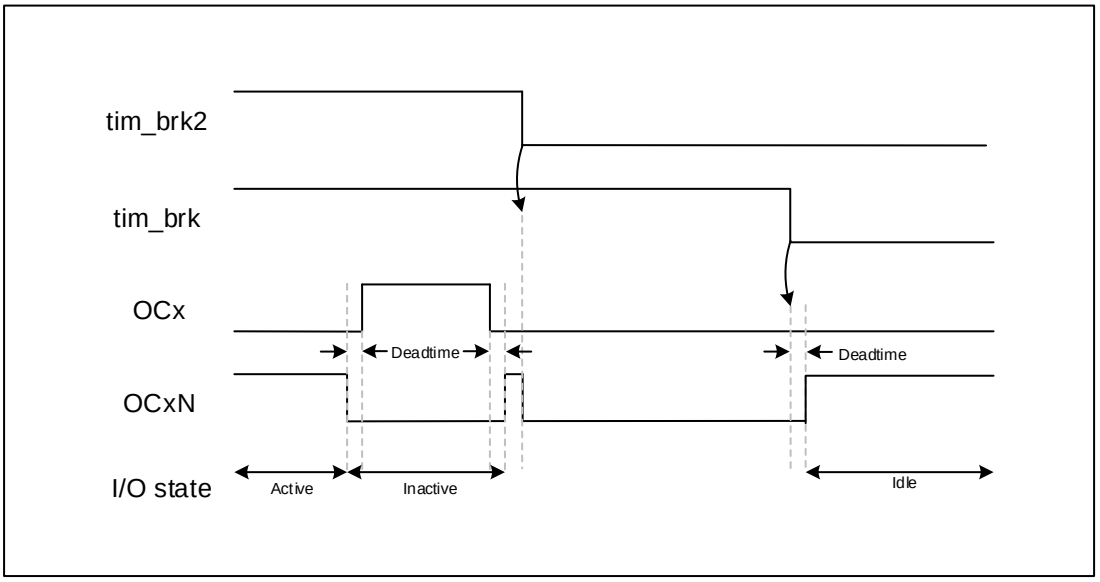


图 19-48 发生刹车和刹车 2 后 PWM 输出状态（OSSI=1）

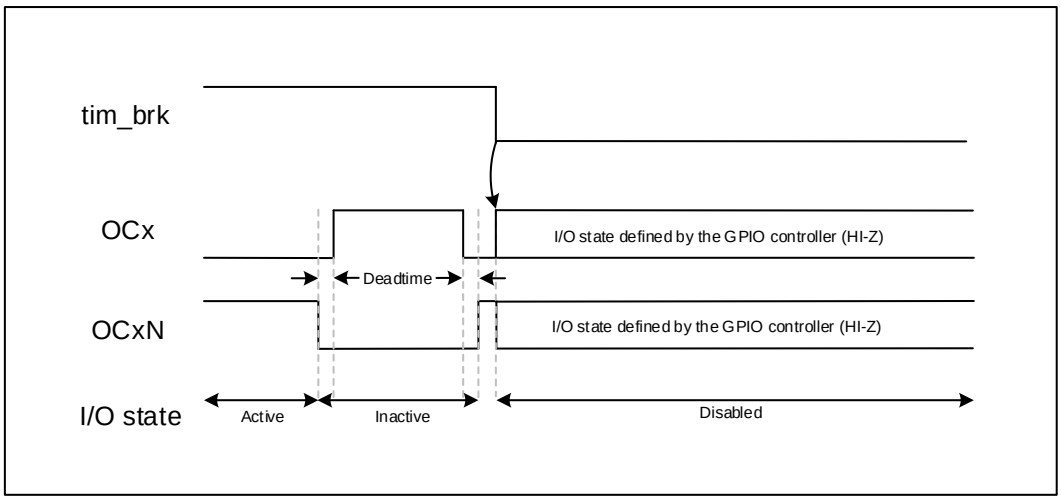


图 19-49 发生刹车和刹车 2 后 PWM 输出状态（OSSI=0）

19.2.17双向刹车输入

TIM1 具备双向的刹车 I/O，如下图所示：

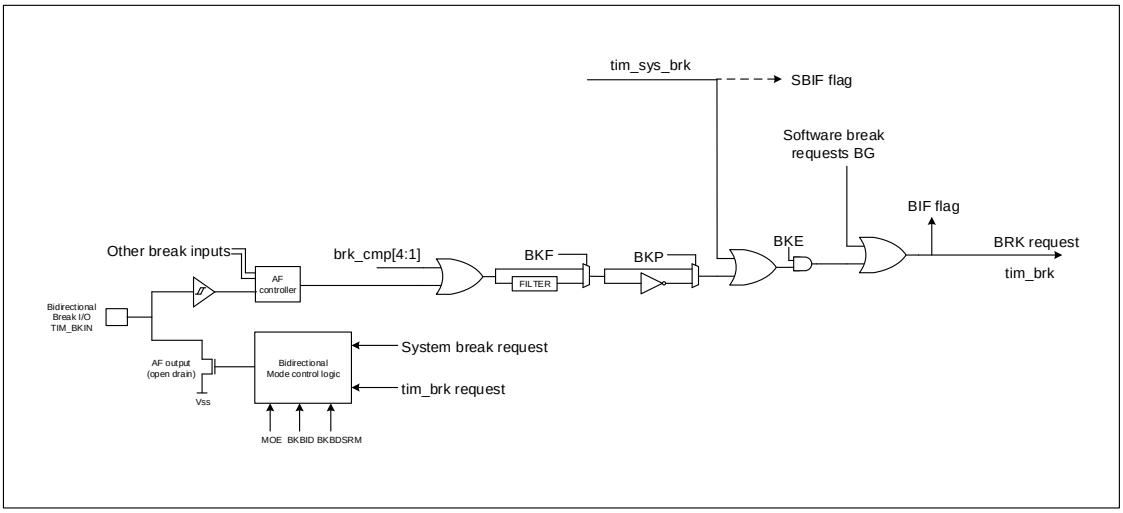


图 19-50 输出重定向（刹车 2 无请求时）

I/O 支持特性：

- 提供一种板级的全局刹车信号，通过唯一的既可做输入也可做输出的引脚，向外部 MCU 和栅极驱动器发送故障信号
- 当需要合并多个内部和外部的刹车源时，内部的刹车源和多个外部的开漏源进行或运算后作为刹车事件的的唯一触发信号

通过 TIMxBDTR 的 BKBID 和 BK2BID 位控制 tim_brk 和 tim_brk2 的输入为双向模式。BKBID 和 BK2BID 可以使用 TIMxBDTR 的 LOCK 位（LOCK level 1）设置为 1，实现被锁定为只读模式。

tim_brk 和 tim_brk2 输入都可使用双向模式，并且要求 I/O 配置成开漏模式且极性为低有效(通过 BKINP、BKP、BK2INP、BK2P 控制位）。无论来自系统的（例如 CCS）、还是板级外设的、还是刹车输入强制一个低电平表示刹车输入故障事件，都可以作为刹车请求。但出于安全的考虑，当极性位未有效配置时（配置成高电平有效时），双向模式会被禁止。

软件刹车事件 (BG、B2G) 也会引起刹车 I/O 被拉低，用于向外部设备显示本 timer 进入到了刹车状态。当然当 BKE 或 B2KE 为 1 时才有可以。当 BKE 或 B2KE 为 0，发生软件刹车事件，输出会被置为安全的状态并产生刹车标志信号，但是对 TIMx_BKIN 和 TIMx_BKIN2 I/O 没有影响

一种安全的解除机制可以防止系统被限制性锁死（在刹车输入口的低电平触发了一个刹车，该刹车又将输入口强制成低电平）

当 BKDSRM（BK2DSRM）位被置为 1 时，可以释放刹车输出控制，清除故障标志信号并提供重新配置系统的可能性。

任何情况下都不能关闭刹车保护电路：

- 刹车输入路径一直有效：即使 BKDSRM（BK2DSRM）位被置 1 和开漏控制被释放，刹车事件还是有效的。这防止在还处于刹车状态时 PWM 输出可以重启
- 当输出被使能的时候（MOE=1）,BKDSRM（BK2DSRM）位无法解除刹车保护

表 19-3 刹车保护状态

MOE	BKBID（BK2BID）	BKDSRM（BK2DSRM）	刹车保护状态
0	0	X	启用
0	1	0	启用
0	1	1	解除
1	X	X	启用

19.2.17.1 对刹车电路进行启用和重启用

默认情况下，刹车电路（在输入或双向模式下）处于启用状态

在发生刹车（或刹车 2）后需要遵循下面的程序来重启动保护：

- BKDSRM(BK2DSRM)位必须置 1 用于释放输出控制
- 软件必须要等到系统刹车状态消失然后清除 SBIF 状态标志（或者在重启用前系统地清除它）
- 软件必须轮询 BKDSRM(BK2DSRM)位直到它被硬件清除（当应用刹车状态消失）

通过这种方式，刹车电路被激活，并且 MOE 位可以被重新使能 PWM 输出

19.2.18在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为‘1’，能够用 OCREF_CLR_INT 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计数溢出所产生的更新事件 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。

而 OCREF_CLR_INT 可以通过配置 TIMx_SMCR 寄存器中的 OCCS 位，在 OCREF_CLR 和 ETRF(ETR 滤波后)之间选择。

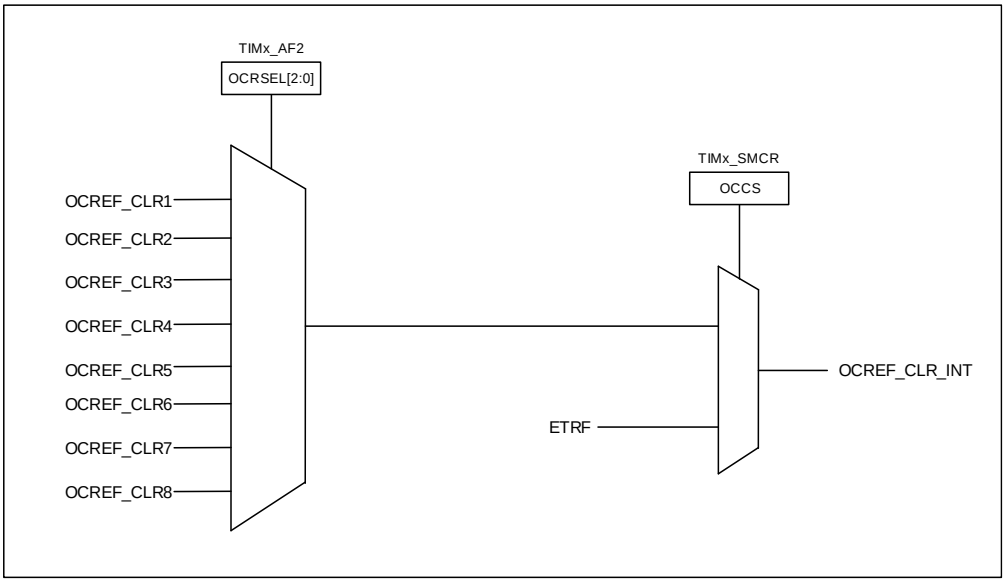


图 19-51 OCREF_CLR 输入选择

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：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)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 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM1 模式。

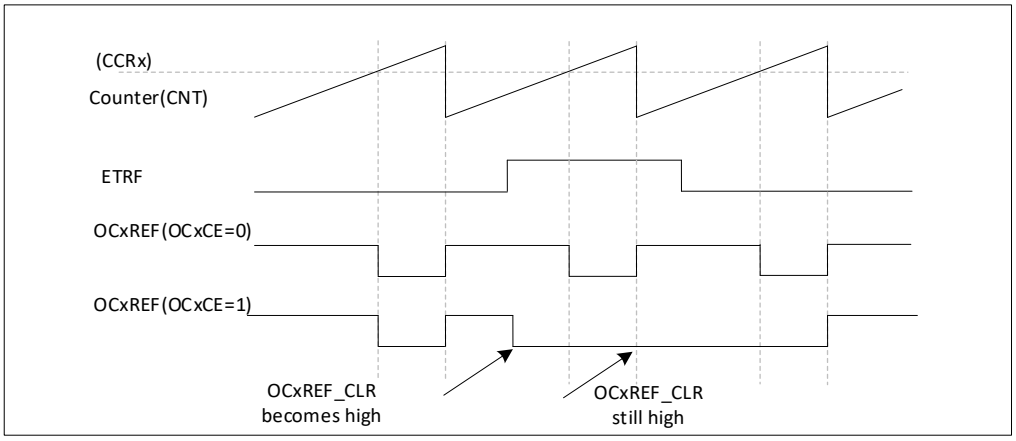


图 19-52 清除 TIMx 的 OCxREF

19.2.19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 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器。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更改所有通道的配置。COM 可以通过设置 TIMx_EGR 寄存器的 COMG 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位(TIMx_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIMx_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

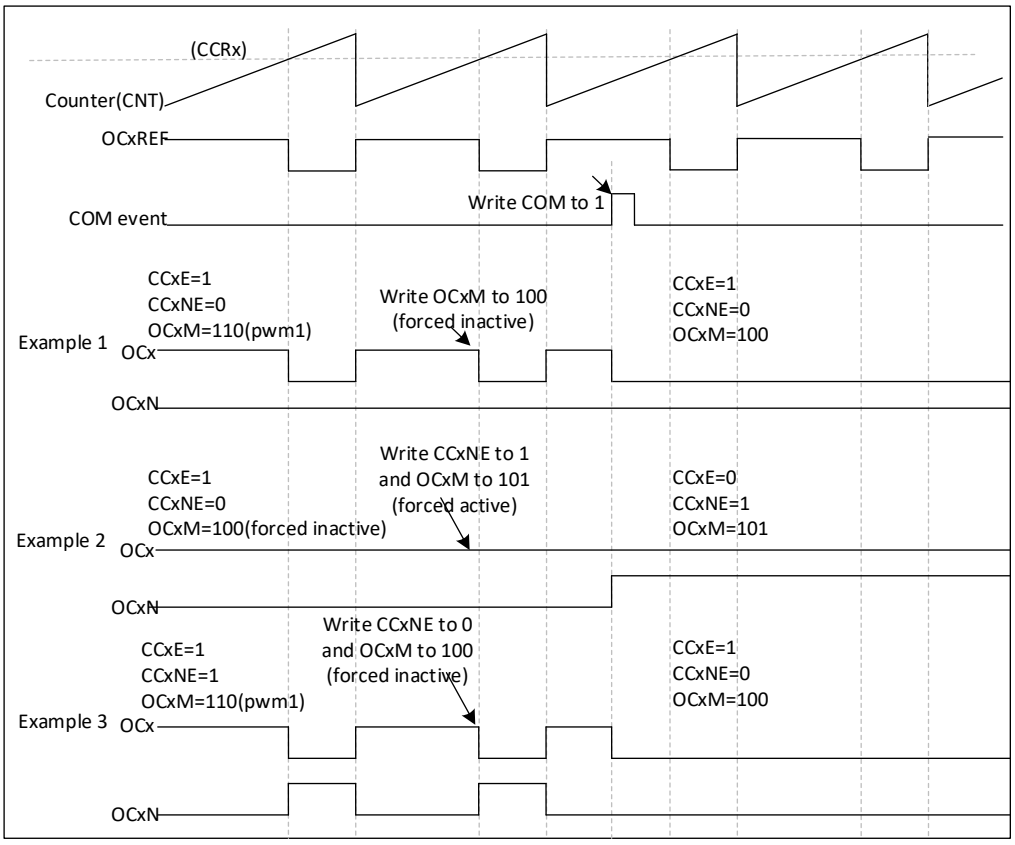


图 19-53 产生六步 PWM，使用 COM 的例子(OSSR=1)

19.2.20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 TIMx_CR1 寄存器中的 OPM 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 < CCRx ≤ ARR（特别地, 0 < CCRx)，
- 向下计数方式：计数器 CNT > CCRx。

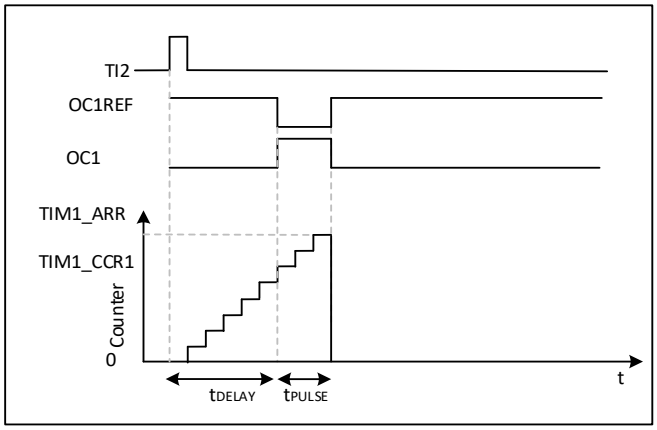


图 19-54 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 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)

- tDELAY 由 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 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 0 到 1 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 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，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=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 OPM=0 时，重复模式被选中。

19.2.20.1 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 CEN 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 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 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。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

19.2.21 可重触发的单脉冲模式

该模式允许计数器响应激励启动并产生长度可编程的脉冲，与上节所述的不可再触发单脉冲模式有以下区别：

- 触发一发生，脉冲就开始（无可编程延迟）。
- 如果在前一个触发所产生的脉冲完成之前发生新的触发，则延长脉冲。

要使用可重触发的单脉冲模式，计时器必须处于从模式，TIMx_SMCR 寄存器中的位 SMS[3:0]=“1000”（组合模式-复位+触发），OCxM[3:0]位设置为“1000”或“1001”（可重新触发 OPM 模式 1 或 2）。

如果此时计时器配置为递增计数模式，则相应的 CCRx 必须设置为 0（ARR 寄存器设置脉冲长度）。

如果计时器配置为递减计数模式，CCRx 必须大于或等于 ARR。

注：出于兼容性原因，OCxM[3:0]和 SMS[3:0]位字段被分成两部分，最高有效位与 3 个最低有效位位置不连续。

该模式不得与中央对齐计数模式一起使用。TIMx_CR1 中必须为 CMS[1:0]=00。

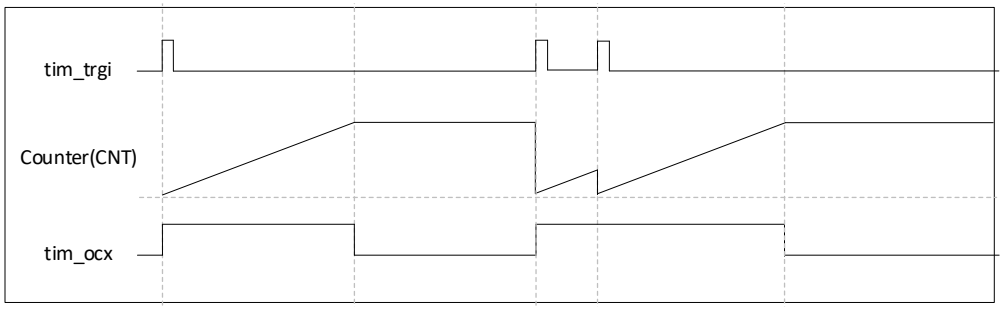


图 19-55 可重触发的单脉冲模式的例子

19.2.22 比较模式生成脉冲

脉冲可以通过比较匹配事件产生。当计数器值等于给定的比较值时，一个可编程宽度的脉冲信号就产生了，用于调试和同步的用途。

该模式可用于任何从模式，包括编码器模式。在边沿和中央对齐计数模式中它仅适用于 3/4 通道。脉冲生成器是唯一的并且由这两个通道共享，如下图所示：

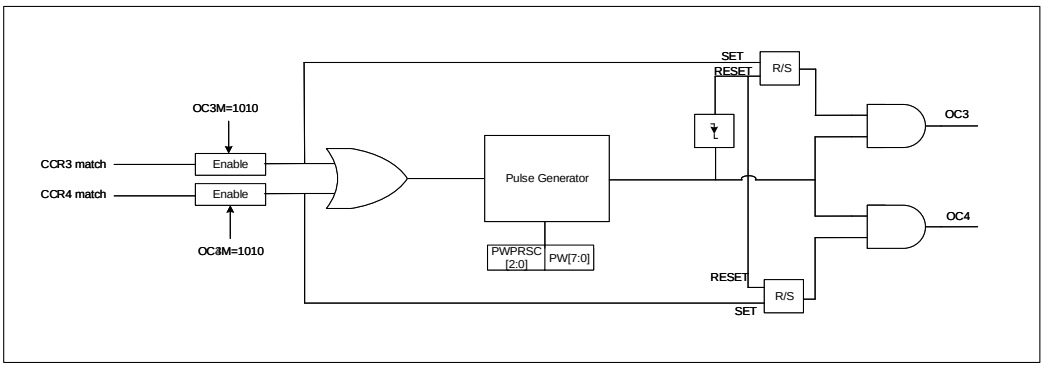


图 19-56 脉冲生成器电路

下图表示在边沿计数和编码器模式中脉冲是如何生成的：

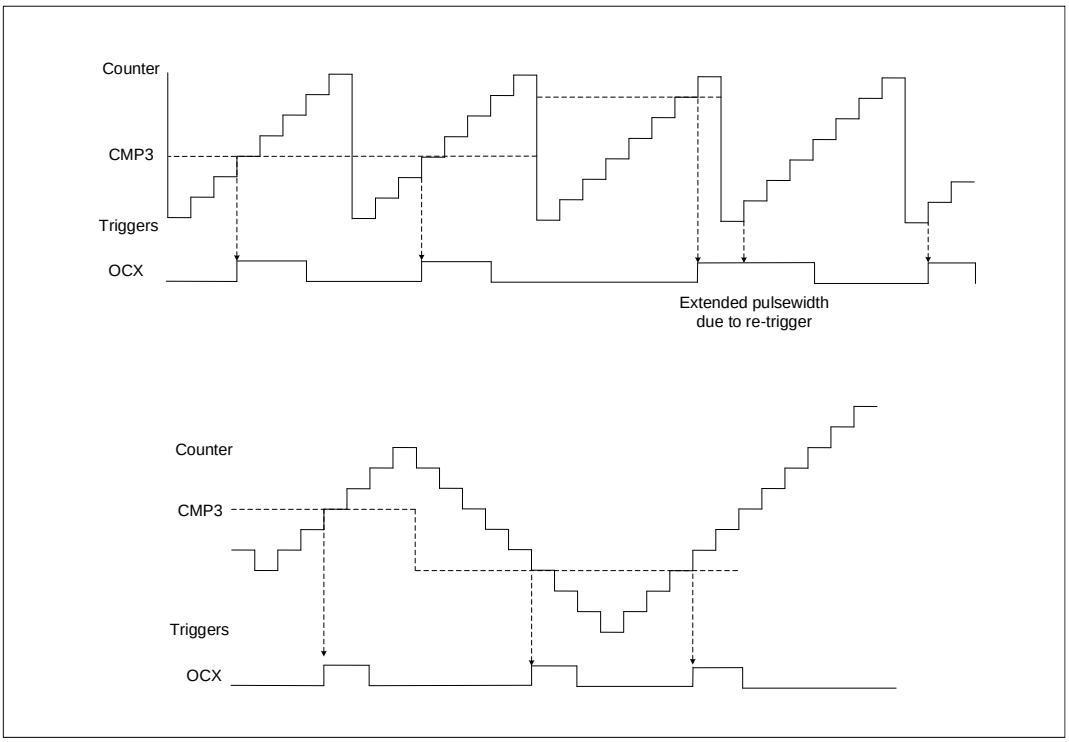


图 19-57 边沿对齐和中央对齐模式中通过比较事件生成脉冲

输出比较模式是通过 TIMx_CCMR2 中的 OC3M[3:0] 和 OC4M[3:0]来选择的

脉冲宽度是通过 PW[7:0]寄存器编程的，通过 PWPRSC[2:0]使用时钟预分频产生的时钟

t_{PW}=PW[7:0] x t_{PWG}

t_{PWG}= $2^{(PWPRSC[2:0])}$ x CK_INT

分辨率和最大值取决于预分频器的值

脉冲再触发：当脉冲正常输出时来了新的触发，会导致脉冲被延长

注：如果两个通道同时被使能，只要一个通道上的触发不与另一个通道产生的脉冲重叠，输出的脉冲就是独立的。如果两路触发是重叠的，触发 1 产生脉冲将被延长(会被再触发)，而最后到来的触发产生脉冲宽度是正确的。

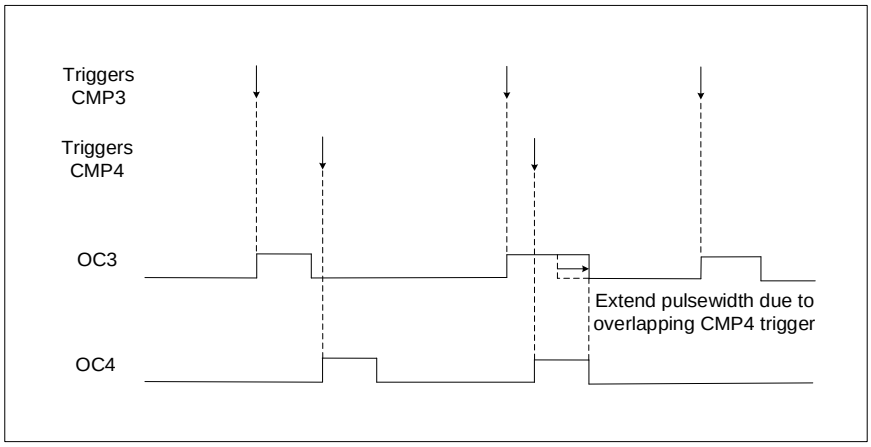


图 19-58 边沿对齐和中央对齐模式中通过比较事件生成脉冲

19.2.23编码器接口模式

19.2.23.1 正交编码器

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 TIMx_SMCR 寄存器中的 SMS=0001；如果只在 TI2 边沿计数，则置 SMS=0010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=0011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 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。参看下表，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=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 TIMx_CR1 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 TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；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 19-4 计数方向与编码器信号的关系（CC1P=CC2P=0）

有效沿	SMS[3:0]	反向信号电平 (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 信号		TI2FP2 信号	
			Rising	Falling	Rising	Falling
仅在 TI1 单沿计数 x1 模式	1110	高	减	加	-	-
		低	-	-	-	-
仅在 TI2 单沿计数	1111	高	-	-	加	减

x1 模式		低	-	-	-	-
仅在 TI1 双沿计数 x2 模式	0001	高	减	加	-	-
		低	加	减	-	-
仅在 TI2 双沿计数 x2 模式	0010	高	-	-	加	减
		低	-	-	减	加
在 TI1 和 TI2 双沿 计数 x4 模式	0011	高	减	加	加	减
		低	加	减	减	加

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：

- CC1S= ‘01’ (TIMx_CCMR1 寄存器，TI1FP1 映射到 IC1)
- CC2S= ‘01’ (TIMx_CCMR2 寄存器，TI2FP2 映射到 IC2)
- CC1P= ‘0’ (TIMx_CCER 寄存器，TI1FP1 不反相，TI1FP1=TI1)
- CC2P= ‘0’ (TIMx_CCER 寄存器，TI2FP2 不反相，TI2FP2=TI2)
- SMS= ‘011’ (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN= ‘1’ (TIMx_CR1 寄存器，计数器使能)

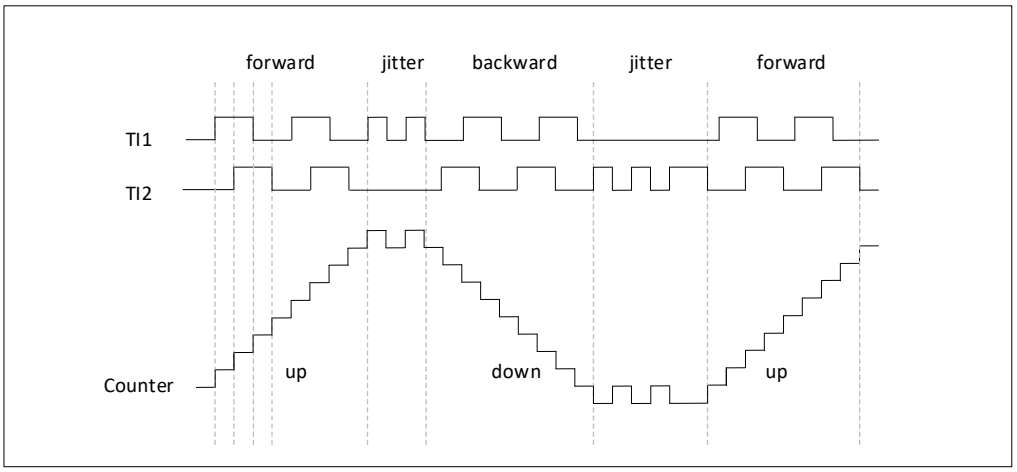


图 19-59 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P=‘1’，其他配置与上例相同)

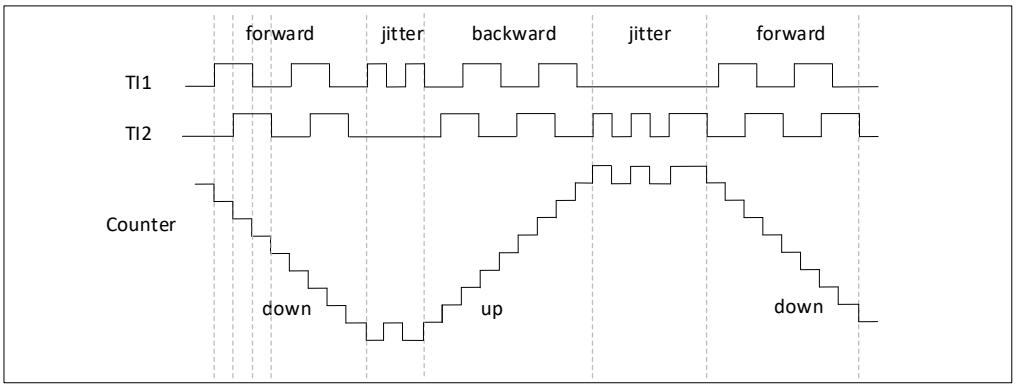


图 19-60 IC1FP1 反相的编码器接口模式实例

下图表示的是不同模式下当转向翻转时的计数值情况：

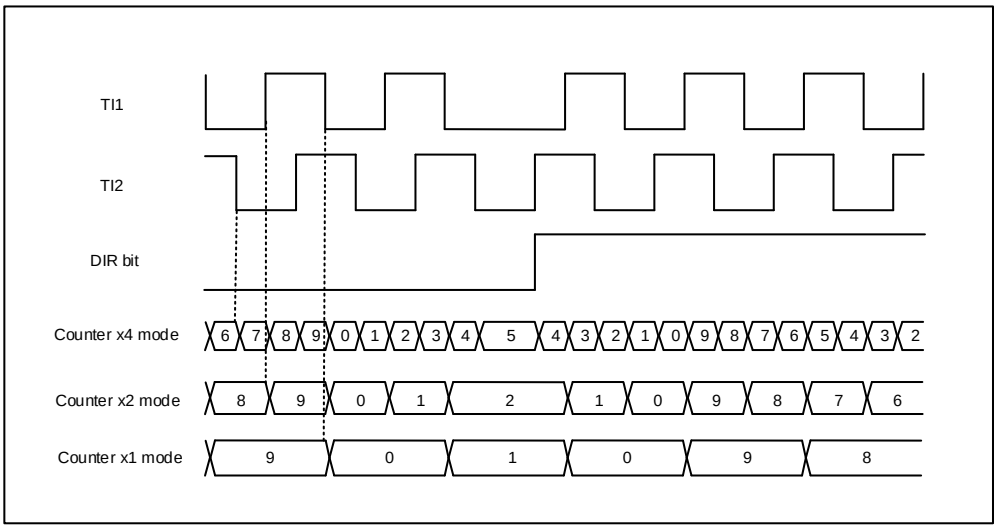


图 19-61 正交编码器计数模式

当定时器配置成编码器接口模式时，可以提供传感器当前位置的信息。通过将第二个定时器配置在捕获模式，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。

TIMxCR1 寄存器的 UIFREMAP 位强制将更新中断标志（UIF）连续复制到定时器计数器的最高位（TIMxCNT[31]）。这样允许计数值和一个由 UIFCPY 标志产生的状态可读出。这样可以通过避免引起竞态条件来简化角速度的计算。例如，通过共享后台任务（计数器读出）和中断之间（中断更新）的处理

UIF 和 UIFCPY 标志断言之间没有延迟

在 32 位计数器设计中，当 UIFREMAP 位被置位，当进行读取时计数器的第 31 位会被 UIFCPY 标志覆盖（计数器的最高有效位只在写模式下可以访问）

19.2.23.2 时钟加方向编码器模式

除正交编码器外，定时器还支持其他不同类型的编码器模式

时钟加方向模式如图所示，时钟由 TI2 提供，方向由 TI1 输入，该模式通过配置 TIMx_SMCR 寄存器的 SMS[3:0]来使能，如下所示：

- 1010：x2 模式，计数器在时钟的上升沿和下降沿更新
- 1011：x1 模式，计数器在单沿更新，CC2P=0 时上升沿更新，CC2P=1 时下降沿更新

TI1 输入的方向信号的极性由 CC1P 位决定，CC1P=0：正极性（当 TI1 为高时向上计数，当 TI1 为低时向下计数）；CC1P=1：负极性（当 TI1 为高时向下计数，当 TI1 为低时向上计数）

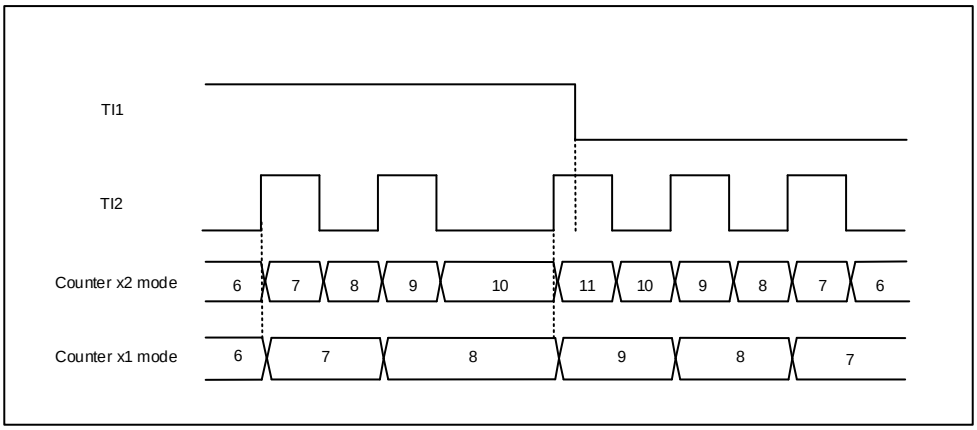


图 19-62 方向加时钟编码器模式

19.2.23.3 定向时钟编码器模式

在定向时钟模式中（如下图所示），时钟由两路信号提供，根据方向一次提供一条，从而具有一条向上计数时钟信号（TI2）和一条向下计数时钟信号(TI1)。该模式通过配置 TIMx_SMCR 寄存器的 SMS[3:0]来使能，如下所示：

- 1100：x2 模式，无论哪条时钟线，计数器都是在时钟的上升沿和下降沿都更新。可以通过 CC1P 和 CC2P 位来配置时钟初始状态。CCxP=0 代表高电平初始状态，CCxP=1 代表低电平初始状态
- 1101：x1 模式，计数器在单沿更新，取决于 CCx1P 和 CCx2P 的值。CCxP=0 表示下降沿计数并且初始状态为高电平。CCxP=1 表示上升沿计数并且初始状态为低电平。

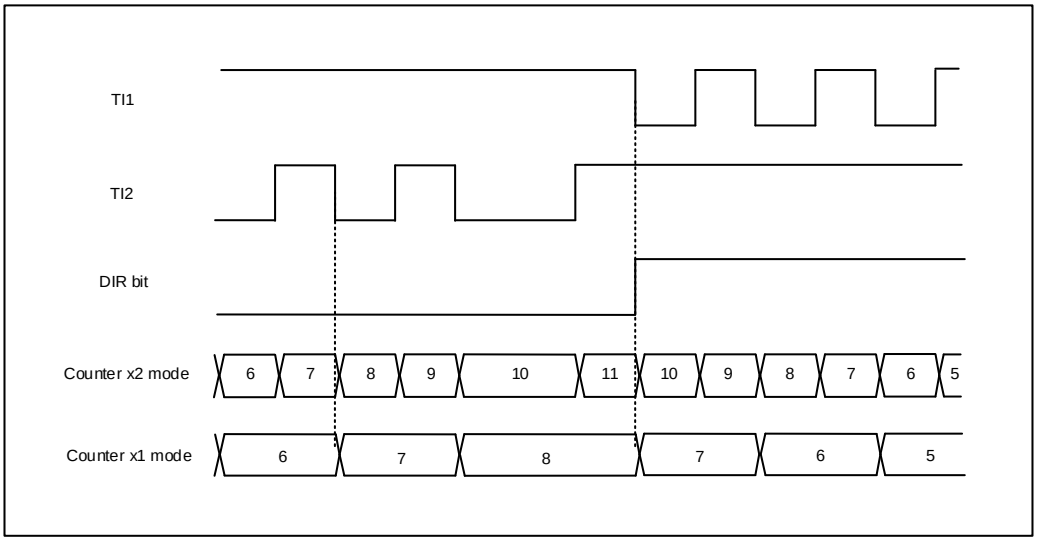


图 19-63 定向时钟编码器模式（CC1P=CC2P=0）

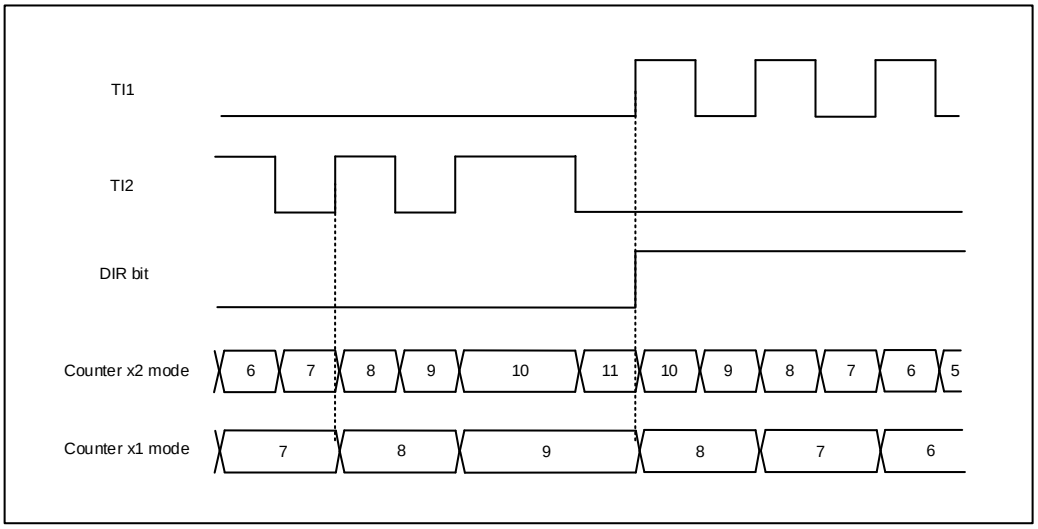


图 19-64 定向时钟编码器模式（CC1P=CC2P=1）

下表详细说明了对于任何输入转换，定向时钟模式是如何操作的：

表 19-5 计数方向与编码器信号极性的关系

有效沿	SMS[3:0]	反向信号电平 (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 信号		TI2FP2 信号	
			Rising	Falling	Rising	Falling
x2 模式 CCxP=0	1100	高	减	减	加	加
		低	-	-	-	-
x2 模式 CCxP=1	1100	高	-	-	-	-
		低	减	减	加	加
x1 模式 CCxP=0	1101	高	-	减	-	加
		低	-	-	-	-
x1 模式 CCxP=1	1101	高	-	-	-	-
		低	减	-	加	-

19.2.23.4 索引输入

计数器可以被编码器输出的索引信号复位，用于指示绝对参考位置。索引信号必须连接到 tim_etr 输入端。可通过数字滤波器来滤波。

TIMx_ECR 寄存器的 IE 位作为索引功能的使能，IE 位只能在编码器模式中使能，即当 SMS[3:0]配置成如下值时：0001、0010、0011、1010、1011、1100、1101、1110、1111

如下图所示，商用编码器有几种用于索引脉冲调节的选项

- A 与 B 的门控：脉冲宽度为一个通道的 1/4，与 A 和 B 的边沿对齐
- A 的门控（B 的门控）：脉冲宽度为一个通道的 1/2，与 A 的边沿对齐
- 非门控：脉冲宽度高达一个通道的周期，不与通道边沿对齐

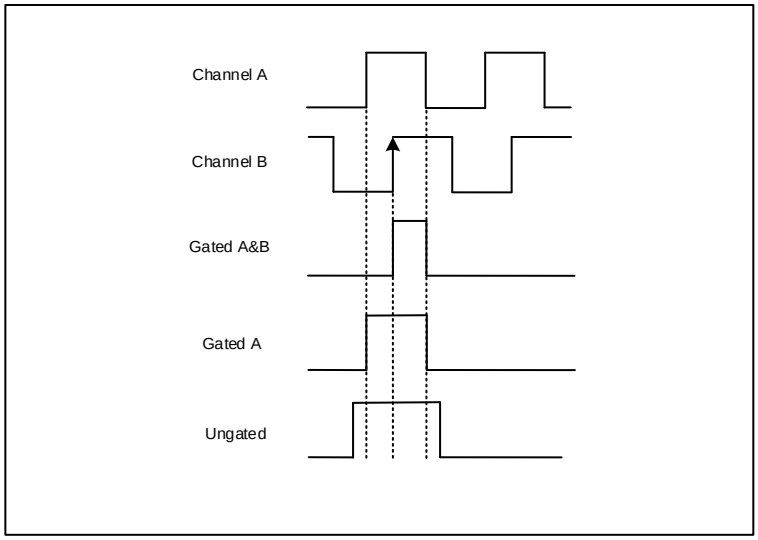


图 19-65 索引门控选择

下图表示的是，任意门控模式下，电路可容忍索引信号的抖动。在非门控模式中，索引信号脉宽需要严格小于 2 个编码器周期。如果大于或者等于 2 个周期，计数器会被复位多次。

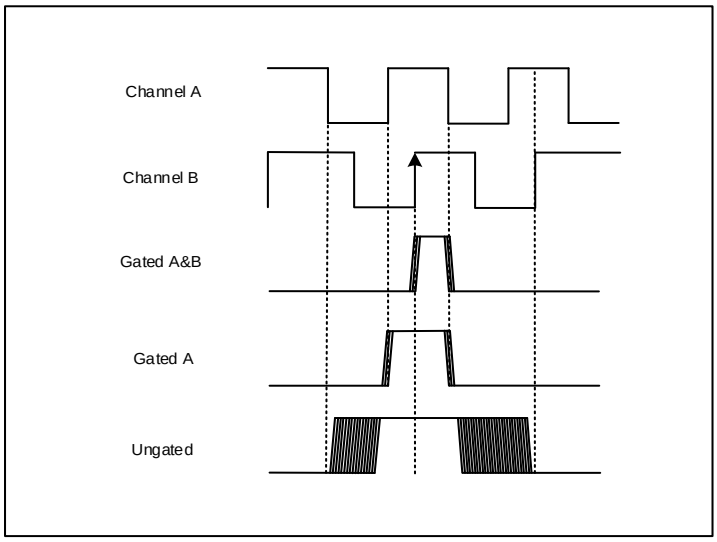


图 19-66 抖动的索引信号

定时器完全支持 3 种门控模式，不需要特别的编程，只需要定义编码器为何状态时（通道 A 和通道 B 状态的组合）索引信号需要被同步，通过配置 TIMx_ECR 寄存器的 IPOS[1:0]。

索引信号的检测事件根据计数方向的不同而不同，以确保旋转方向翻转期间的对称操作。

- 计数器在向上计数过程中被复位(DIR=0)
- 向下计数时，计数器的值设置为 TIMx_ARR

这使得无论计数方向如何，索引都可以在相同的机械角度位置上生成，下图展示了哪些位置索引生成，举个简单的例子（一个编码器提供 4 个沿每个标准机械旋转）

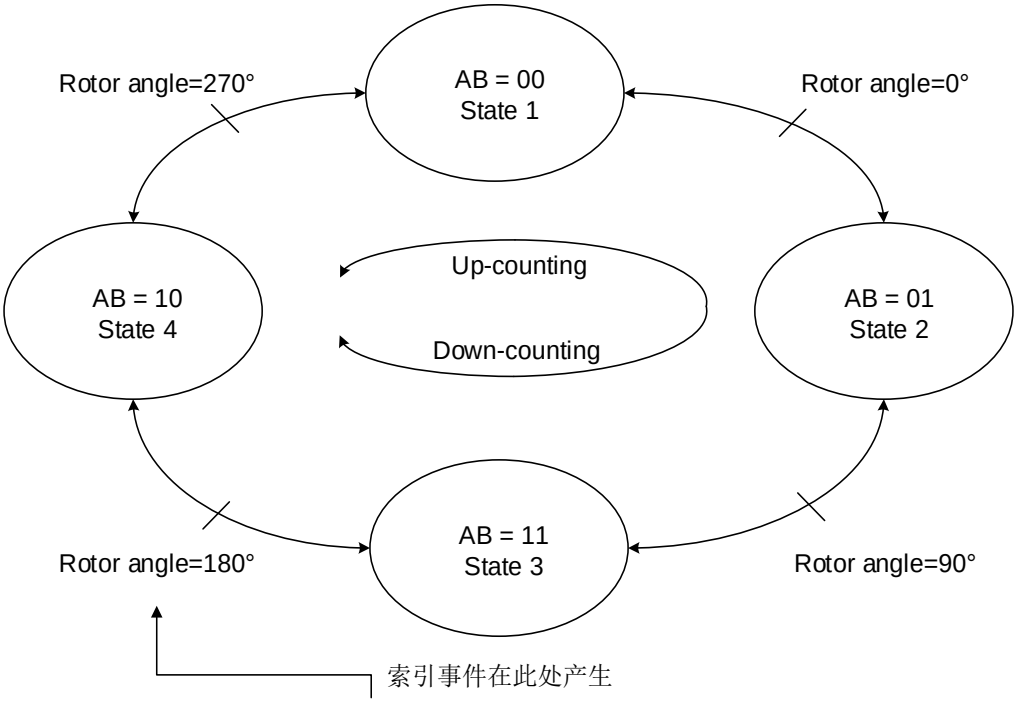


图 19-67 IPOS[1:0]=11 时索引生成

下图显示了 IPOS[1:0]=11 时的波形和相应值，它表明计数器值被强制的时刻是根据计数方向自动调整的

- 计数器清零当编码器进入 11 状态（通道 A=1，通道 B=1），向上计数时（DIR=0）
- 计数器值被设为 TIMx_ARR 当离开 11 状态，向下计数时（DIR=1）

索引检测事件可产生中断

箭头表示哪个转换过程中索引事件中断产生

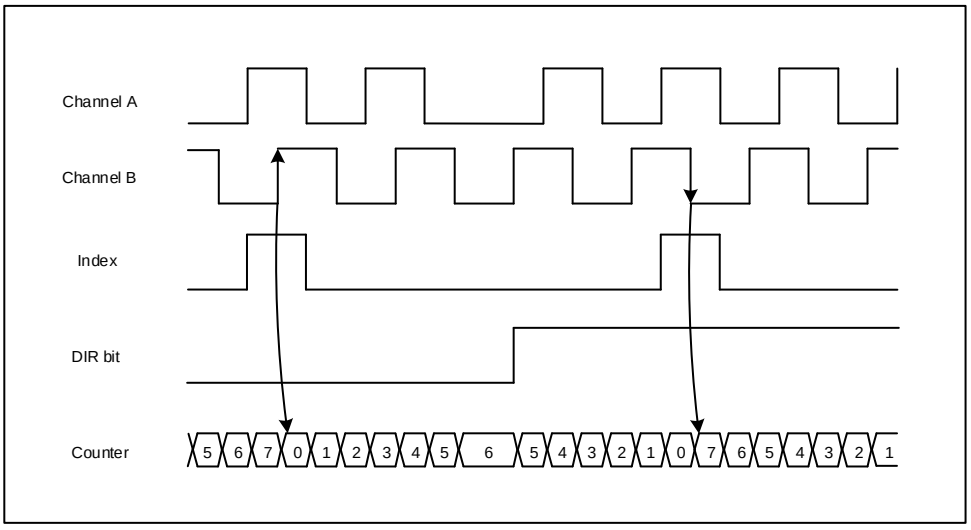


图 19-68 IPOS[1:0]=11 时索引与通道 A 门控模式时的计数值

下图显示了非门控模式时的波形和相应值，箭头表示哪个转换过程中索引事件中断产生

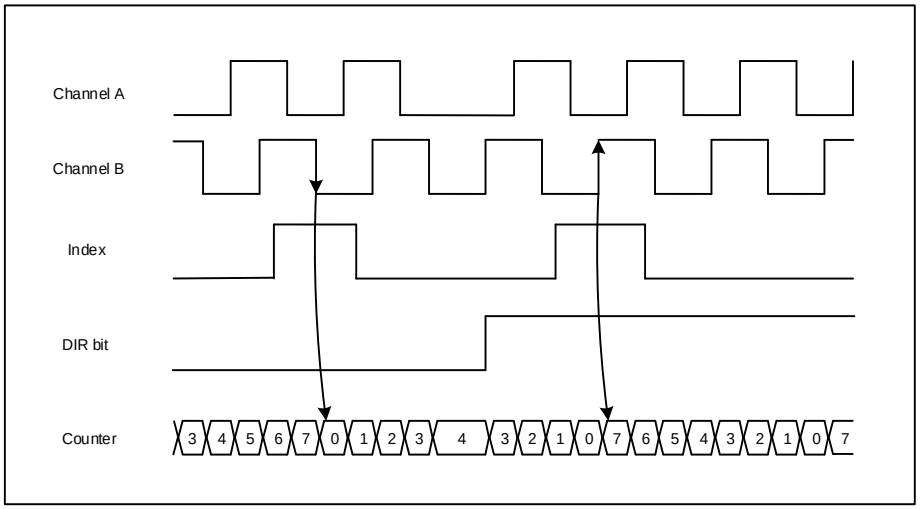


图 19-69 IPOS[1:0]=00 时无门控索引模式时的计数值

下图显示了 A 与 B 门控模式时的波形和相应值，箭头表示哪个转换过程中索引事件中断产生

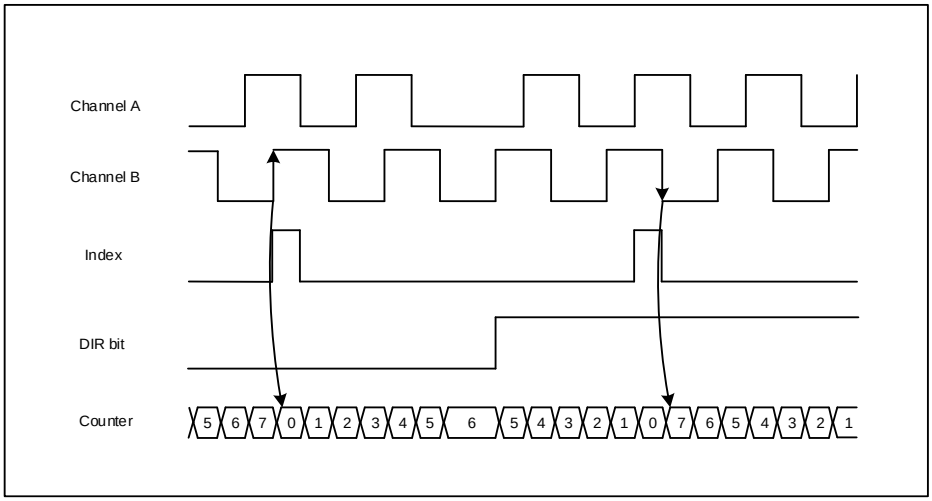


图 19-70 索引与通道 A 和 B 门控模式时的计数值

下面两张图详细说明了索引脉冲宽度小于编码器周期的 1/4 的情况

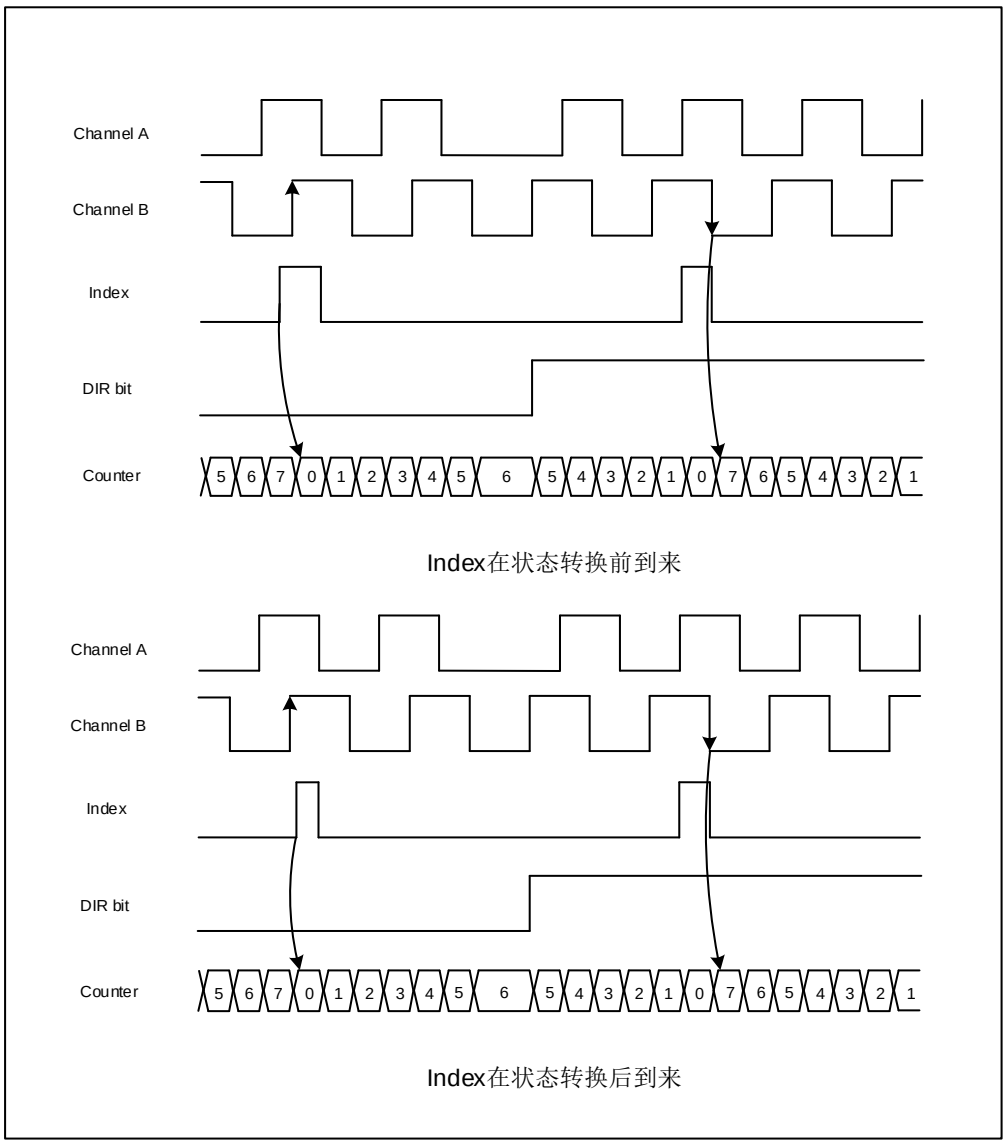


图 19-71 IPOS[1:0]=11 时窄索引脉冲时编码器模式的运行

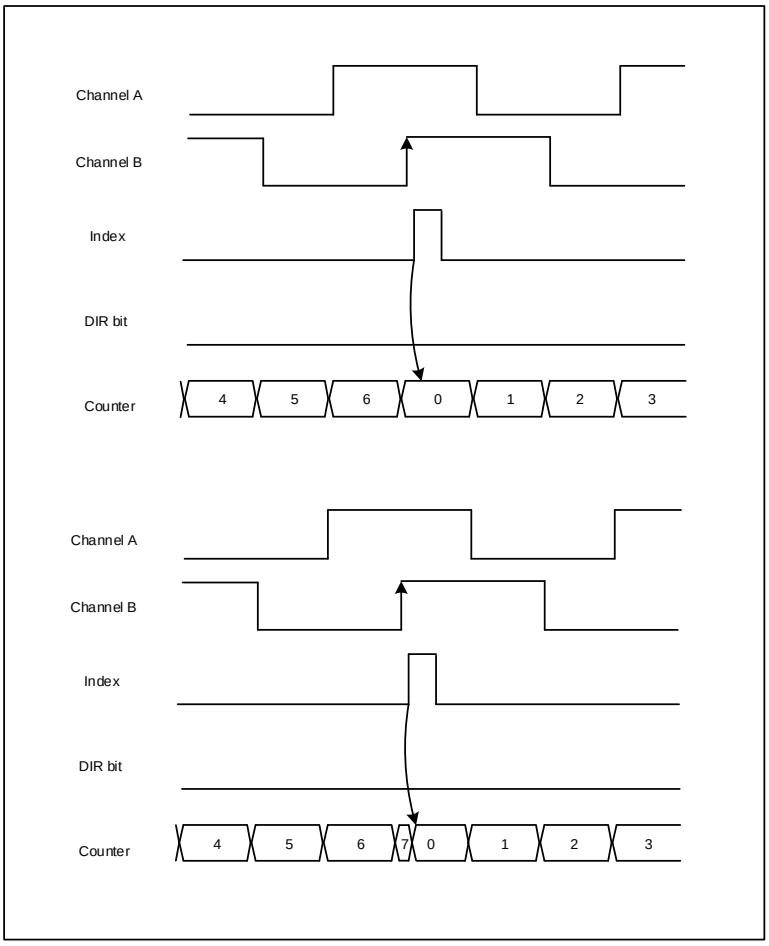


图 19-72 ARR=0x07 时 时窄索引脉冲复位计数器

下图详细显示了在 x1 和 x2 模式中如何管理索引

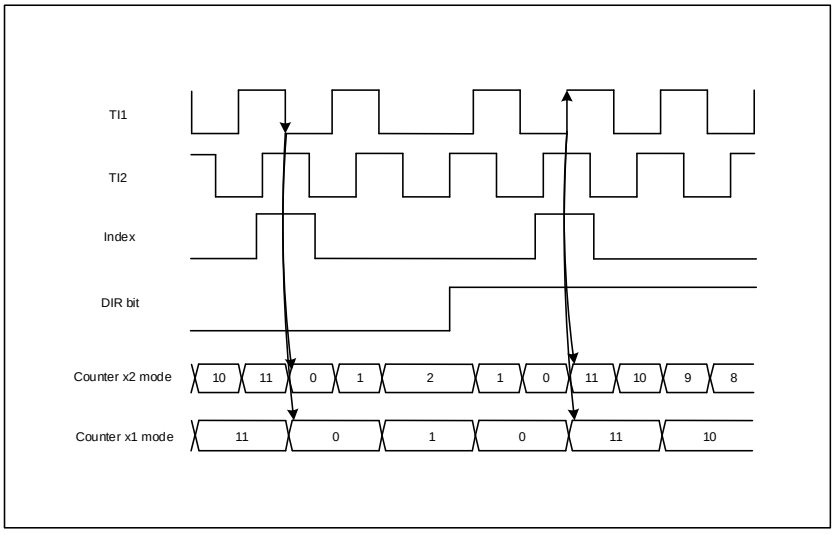


图 19-73 IPOS[1:0]=01 时在 x1 和 x2 模式中

■ 方向索引敏感性

可以通过配置 TIMx_ECR 寄存器的 IDIR[1:0]来允许索引仅在选定的计数方向上有效。

下图显示了基于 IDIR[1:0]的值，索引和计数复位事件之间的关系。

注：IDIR 必须在 IE 位被复位时才能被写入（索引模式关闭）

注：方向索引灵敏性不支持时钟+方向模式，当 SMS=1010 或 1011 时，IDIR 必须设置为 00

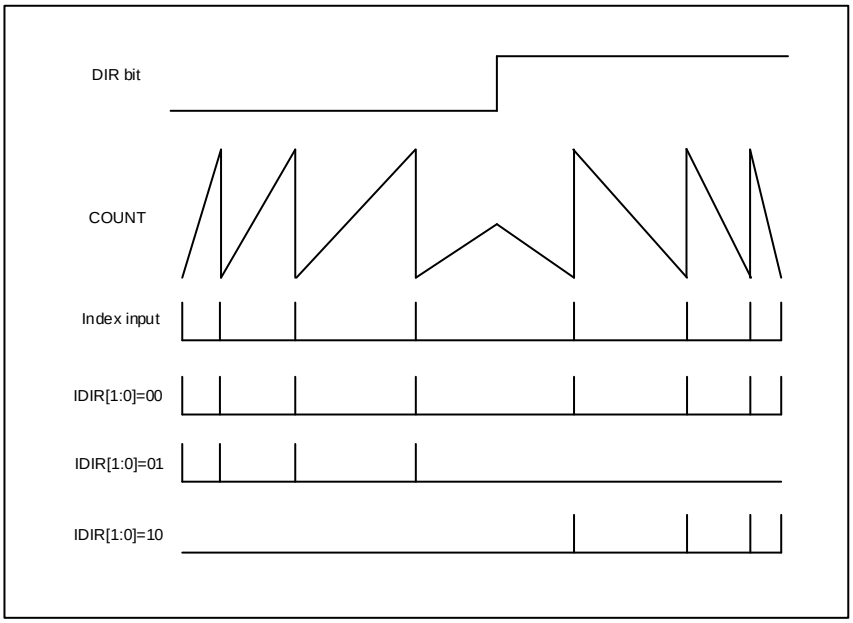


图 19-74 定向的索引敏感性

■ 特殊的第一次索引事件管理

TIMx_ECR 寄存器的 FIDX 位允许索引只进行一次，如下图所示。当第一次索引到来时，后续的索引都会忽略。如果需要，可以通过将 FIDX 位写 0 再写 1 来重启用电路。

注：当 FIDX=1 时，如果方向在位置 0(索引激活)发生改变，则可能发出两次索引（IDXF 标志置起）

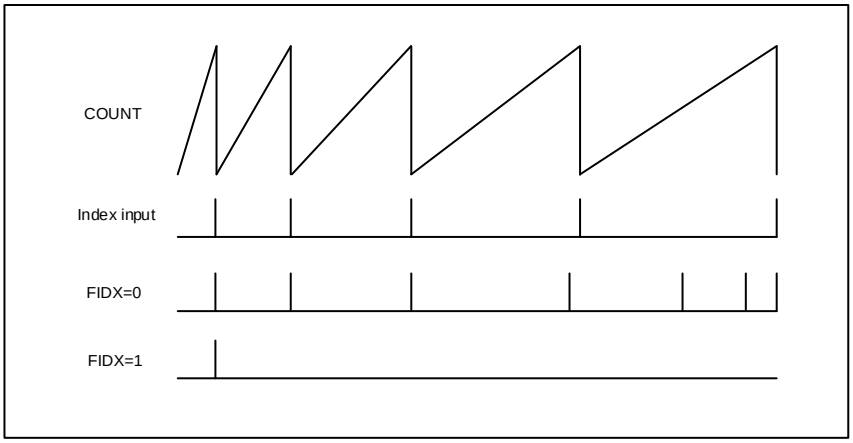


图 19-75 当 FIDX 位被置位时，计数器复位情况

■ 在非正交模式下的索引管理

下面两张图详细描述了在定向时钟模式和时钟+方向模式中索引是如何管理的。即 SMS=1010 或 1011 在这些模式中，索引的敏感性由 IPOS 决定

IPOS[0]=0:在时钟的低电平检测索引

IPOS[0]=1:在时钟的高电平检测索引

IPOS[1]没有作用。

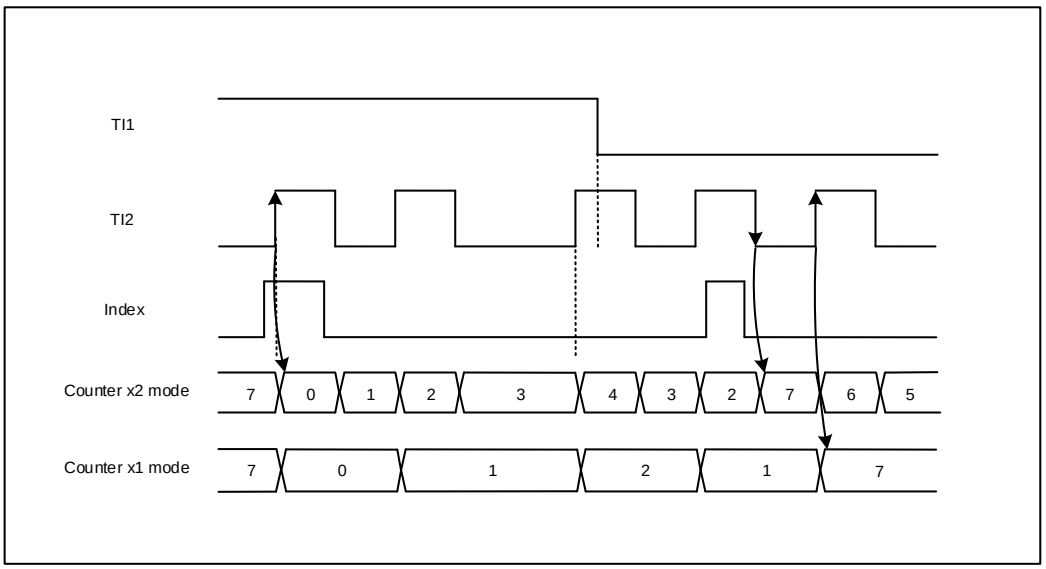


图 19-76 时钟加方向模式索引行为情况（IPOS[0]=1）

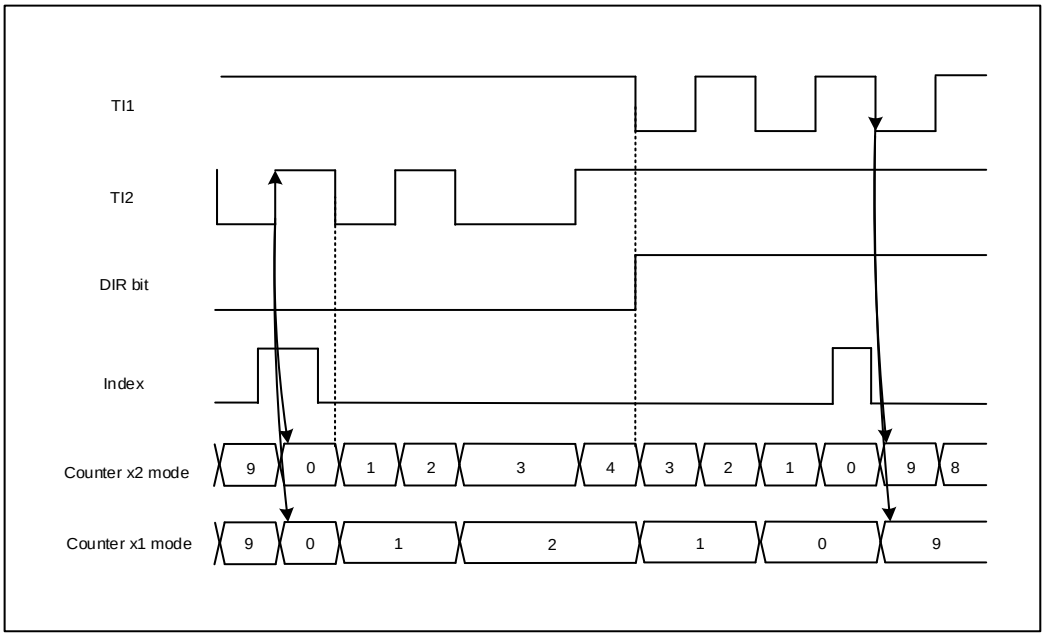


图 19-77 定向时钟模式索引行为情况（IPOS[0]=1）

■ 编码器错误管理

编码器配置中有 2 路正交信号是有效的，有必要检测转换错误。2 个输入上的读数对应 2 位格雷码来表示状态图，如图所示。每次只能改变 1 位。错误的转换将 TIMx_SR 状态寄存器中的中断标志 TERRF 拉高。若 TIMx_DIER 寄存器的 TERRIE 位置 1，则会产生转换错误中断。

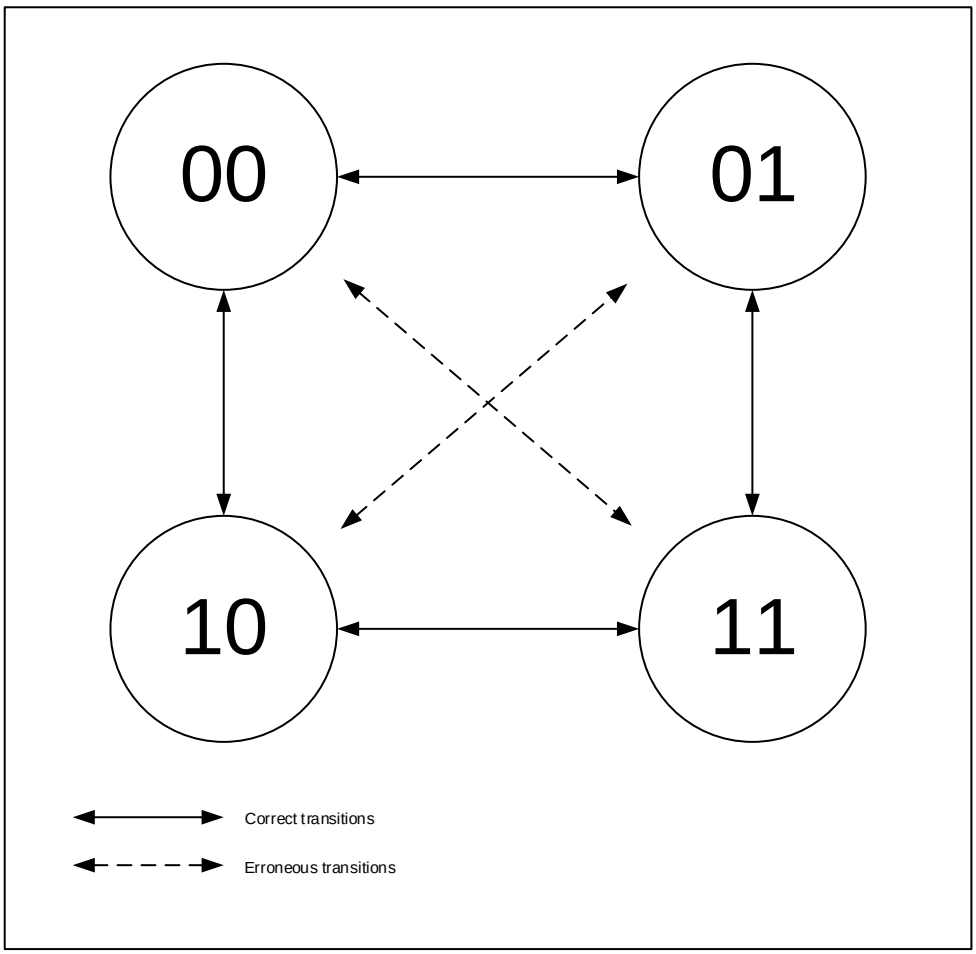


图 19-78 正交编码信号状态示意图

编码器有了索引信号后，就可以检测出导致每次旋转过程中脉冲超额的异常操作。编码器每旋转一次会提供 N 个脉冲，并会产生 4*N 的计数。索引信号每经过 4*N 时钟周期就会复位一次计数器。

如果计数器值从 TIMx_ARR 到 0 或者从 0 到 TIMx_ARR 递变而没有索引事件的话，这会被报告为索引位置错误。

溢出阈值由 TIMx_ARR 寄存器配置。1000 行编码器的结果计数值介于 0 到 3999 之间（4x 读模式中）。溢出检测阈值必须设置为 TIMx_ARR=3999+1=4000

向上计数时，错误会在延迟到 0 到 1 的转换时断言，下图表示在 A 与 B 门控模式中，对窄索引信号的处理。

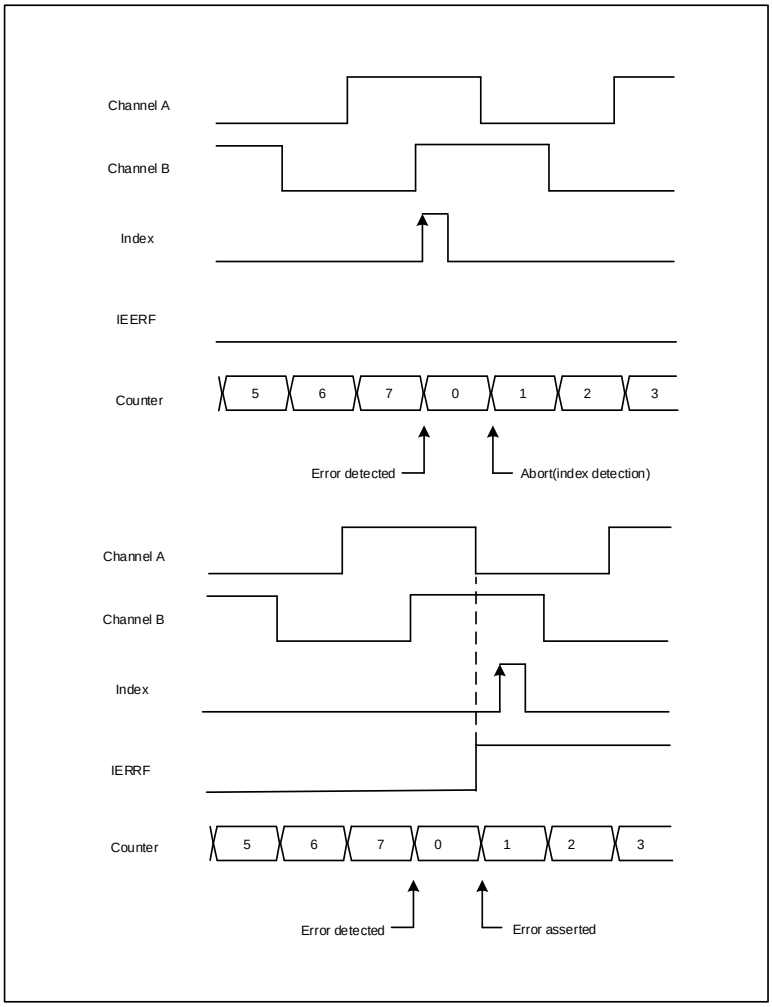


图 19-79 向上编码时错误检测

向下计数时，检测必须是以先前是从 1 到 0 的转换作为前提条件，下图表示在 A 与 B 门控模式中，对窄索引信号的处理。为了避免索引检测后编码器立即在 TIMx_ARR 和 0 两个值之间抖动而误产生假的错误认定。

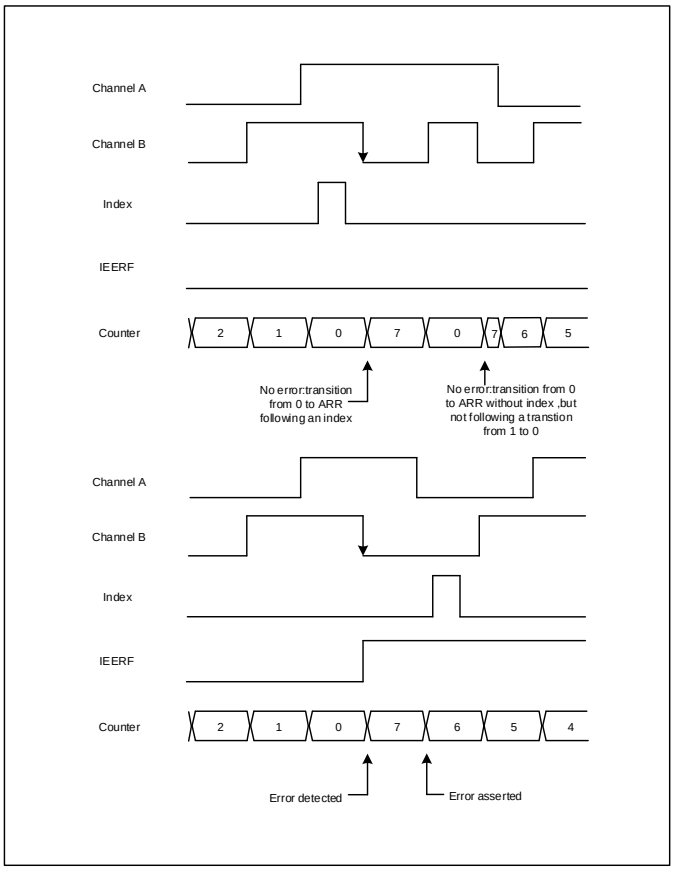


图 19-80 向下编码时错误检测

索引错误会将 TIMx_SR 状态寄存器的 IERRF 中断标志位拉高。若 TIMx_DIER 寄存器的 TERRIE 位置 1，则会产生转换错误中断。

■ 功能性的编码器中断

下面中断均在编码器模式中有效

— 方向改变：在编码器模式中，计数方向的任何改变都会引起 TIMx_CR1 的 DIR 位翻转，方向的改变会拉高 TIMx_SR 状态寄存器的 DIRF 中断标志位。如果 TIMx_DIER 寄存器的 DIRIE 位使能，还会产生方向改变的中断

— 索引事件:索引事件会拉高 TIMx_SR 状态寄存器的 IDXF 中断标志位。如果 TIMx_DIER 寄存器的 IDXIE 位使能，还会产生索引中断

■ 运行时从模式预加载功能更新编码器模式

从一个编码器模式切换到另一个运行模式是很有必要的，这通常在高速下完成的，以降低更新中断的速率。下图表示的是从 x4 模式切换到 x2 模式再切换到 x1 模式。

基于这个目的，SMS[3:0]可以预加载。使能位由 TIM_SMCR 寄存器的 SMSPE 位控制。从 SMS[3:0]预加载到激活值的触发信号选择可以使用 TIMx_SMCR 寄存器的 SMSPS 位控制。

— SMSPS=0:传输由 UEV 触发，该模式必须关闭索引功能(IE=0)

— SMSPS=1:传输由索引事件触发

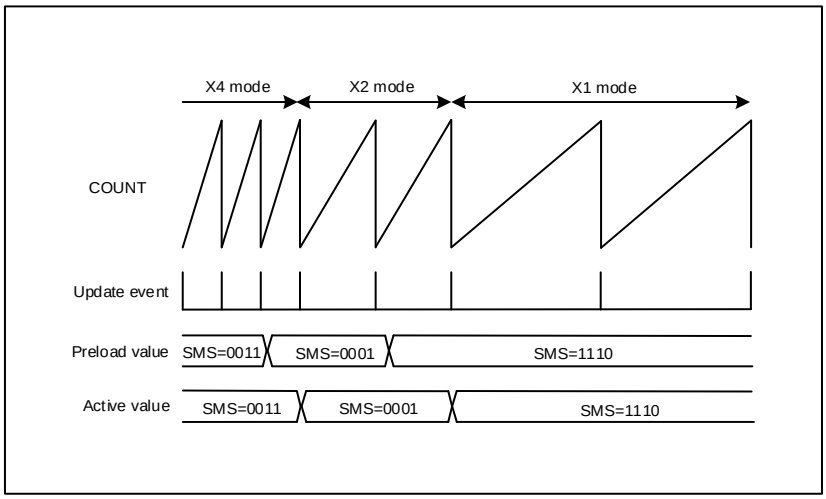


图 19-81 事件更新时编码器模式通过预加载更新

■ 编码器时钟输出

编码器的工作原理并不完全适用于高分辨率快速测量。在低速时，它需要相当长的整合时间来才能具有足够数量的时钟边沿和精确的测量。

在低速时，最好的方法是直接测量边沿到边沿的时钟周期。这时可能需要使用一个从定时器。本定时器可以通过 tim_trgo 输出编码器的时钟信号。从定时器可以执行周期测量并为每个编码器时钟边沿提供速度信息。

该模式通过设置 TIMx_CR2 寄存器的 MMS[3:0]=1000 来使能。并且仅当 SMS[3:0]配置为 0001,0010,0011,1010,1011,1100,1101,1110,1111 才能使用。其他的 SMS[3:0]值不被允许，可能会产生预期之外的行为。

19.2.24方向位输出

定时器的方向位也是有必要输出的，通过配置 TIMx_CCMR2 寄存器的 OC3M[3:0]和 OC4M[3:0]位段为 1011 实现从 OC3N 和 OC4 通道输出（复制 TIMx_CR1 寄存器的 DIR 位）。

该特性用于监测编码器的计数方向(或旋转方向)，或者能提供一个中间对齐 PWM 模式中上/下阶段的指示信号。

19.2.25UIF 位重映射

TIMx_CR1 寄存器的 UIFREMAP 位能够强制连续地将 UIF 复制到定时器计数器的第 31 位(TIMxCNT[31])。这样可以允许计数器的值和由 UIFCPY 标志信号展现的一种潜在的滚动状态都可以读出。在特殊情况下，这样可以通过避免竞争条件来简化计算，例如由同时处理计数器读出和更新中断。UIF 和 UIFCPY 标志的产生没有延迟。

19.2.26定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 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 TIMx_CH1、TIMx_CH2 和 TIMx_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。下节给出了此特性用于连接霍尔传感器的例子。

19.2.27 与霍尔传感器的接口

使用高级控制定时器(TIM1 或 TIM8)产生 PWM 信号驱动马达时，可以用另一个通用 TIMx(TIM2、TIM3、TIM4 或 TIM5)定时器作为“接口定时器”来连接霍尔传感器，见下图，3 个定时器输入引脚(TI1、TI2、TI3)通过一个异或门连接到 TI1 输入通道(通过设置 TIMx_CR2 寄存器中的 TI1S 位来选择)，此时“接口定时器”被用来捕获这个信号。

从模式控制器被配置成复位模式，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(如下图)。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 COM 事件)用于改变高级定时器 TIM1 或 TIM8 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 TIM1 或 TIM8。

举例：霍尔输入连接到 TIMx 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 TIMx 的 PWM 配置。

- 置 TIMx_CR2 寄存器的 TI1S 位为 ‘1’，配置三个定时器输入逻辑异或到 TI1 输入；
- 时基编程：置 TIMx_ARR 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 TIMx_CCMR1 寄存器中 CC1S=01，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 TIMx_CCMR1 寄存器中的 OC2M=111 和 CC2S=00；
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1 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(TIMx_CR2 寄存器中 CCPC=1)，同时触发输入控制 COM 事件(TIMx_CR2 寄存器中 CCUS=1)。在一次 COM 事件后，写入下一步的 PWM 控制位(CCxE、OCxM)，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

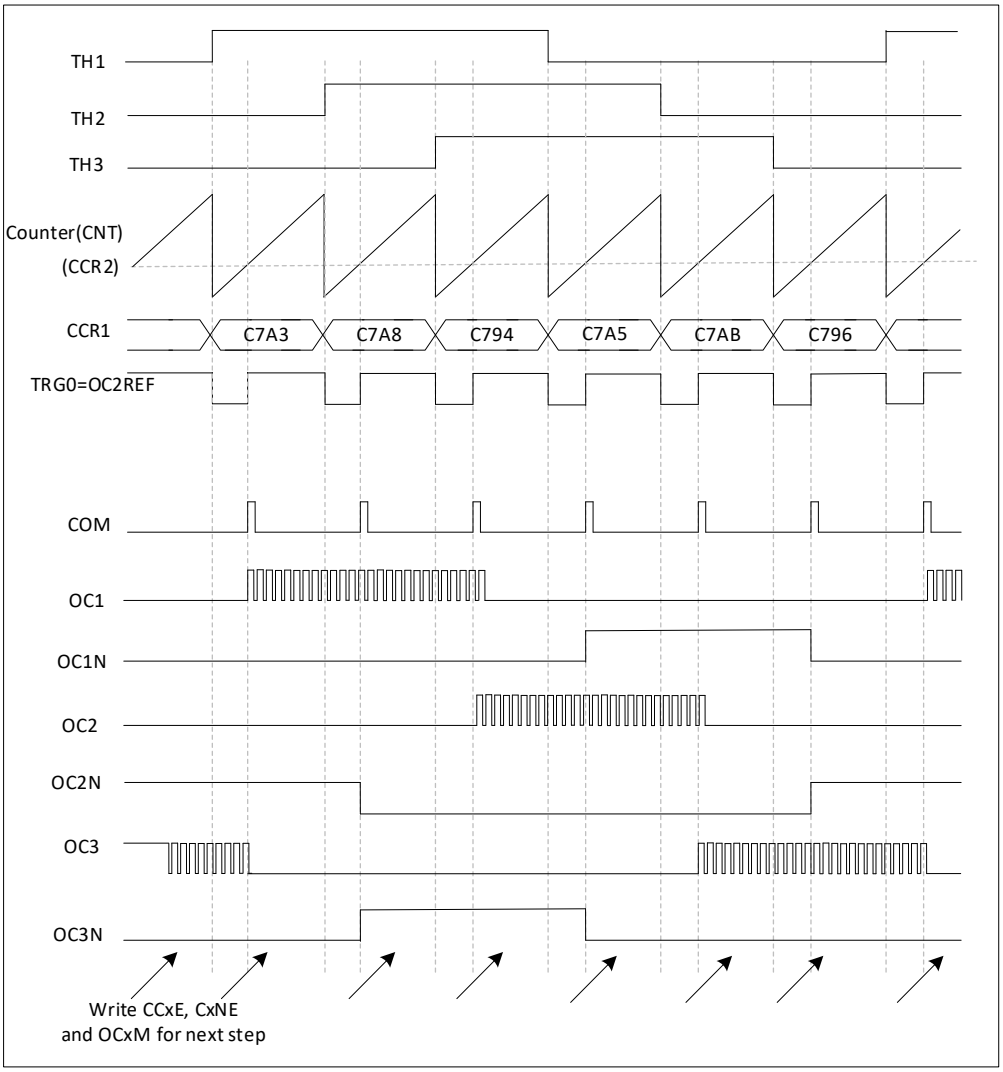


图 19-82 霍尔传感器接口的实例

19.2.28 TIMx 定时器和外部触发的同步

TIMx 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、触发、复位+触发和门控+复位模式。

19.2.28.1 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 TIMx_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=0100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=00101，选择 TI1 作为输入源。

- 置 TIMx_CR1 寄存器中 CEN=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(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(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 TIMx_ARR=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。

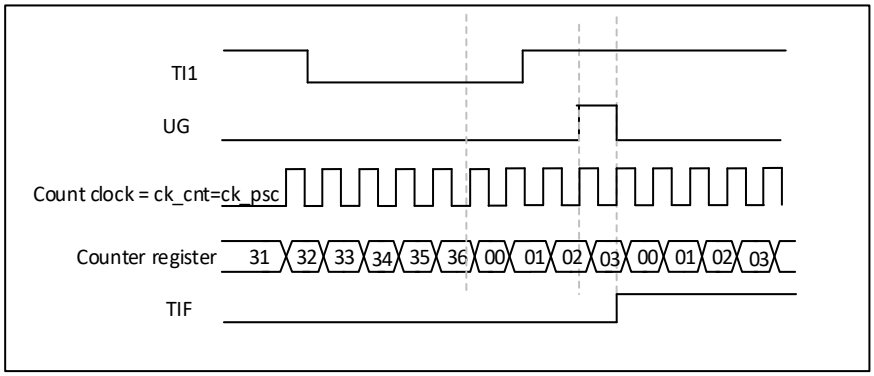


图 19-83 复位模式下的控制电路

19.2.28.2 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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=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 TIMx_SR 中的 TIF 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。

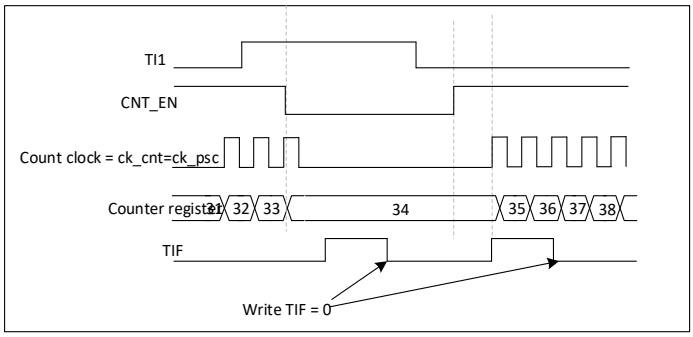


图 19-84 门控模式下的控制电路

19.2.28.3 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 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=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 TIF 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。

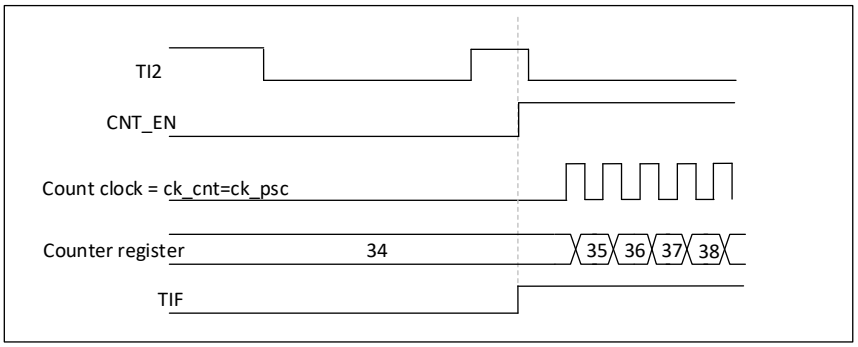


图 19-85 触发器模式下的控制电路

19.2.28.4 从模式：复位加触发组合模式

在该模式下，触发输入的上升沿会初始化计数器，并产生寄存器的更新信号，开启计数

该模式用于单脉冲模式

19.2.28.5 从模式：门控加复位组合模式

当触发输入为高时计数器的时钟才使能。当触发信号拉低时计数器停止。计数器的开始和停止都可以控制

该模式允许检测超出范围的 PWM 信号（占空比超过最大预期值）

19.2.28.6 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 TIMx_SMCR 寄存器的 TS 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 TIMx_SMCR 寄存器配置外部触发输入电路：

— ETF=0000：没有滤波

— ETPS=00：不用预分频器

— ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。

2. 按如下配置通道 1，检测 TI1 的上升沿：

— IC1F=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 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源

— 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)

3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 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。

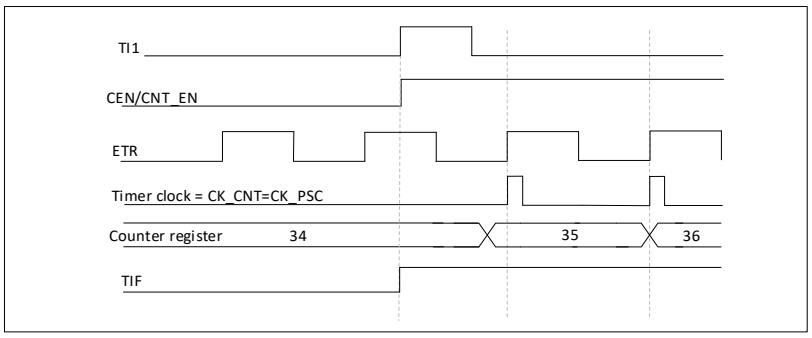


图 19-86 外部时钟模式 2 + 触发模式下的控制电路

19.2.29 DMA 突发模式

TIMx 定时器当发生单个事件时有生成多个 DMA 请求的能力。主要目的是能够多次重新编程定时器的部分功能而无需软件的开销。但是也可以在固定的间隔时间内读取一行中的几个寄存器。

DMA 控制器的目的地是唯一的并且必须指向虚拟寄存器 TIMx_DMAR.当给定的定时器事件发生，定时器发起一系列 DMA 请求。每个写入 TIMx_DMAR 寄存器的操作实际上是重定向一个定时器寄存器的。

TIMx_DCR 寄存器中的 DBL[4:0]位设置 DMA 突发长度。当对 TIMx_DMR 地址进行读取或写入访问时，定时器会识别突发传输，即传输次数（以半个字或字节为单位）

TIMx_DCR 的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址(当通过 TIMx_DMAR 地址完成读写操作)。DBA 定义为从 TIMx_CR1 寄存器地址开始的偏移量。

例子：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发特性是用于发生更新事件时更新 CCRx 寄存器中的内容，通过 DMA 传输半字到 CCRx。

通过以下步骤来完成：

如下配置对应的 DMA 通道：

DMA 通道外设地址是 DMAR 寄存器的地址

DMA 通道存储地址是 RAM 中的 BUFF 地址，包含了 DMA 传给 CCRx 的数据

传输的数据个数=3（见注释）

轮询模式关闭

通过配置 DBA 和 DBL 实现配置 DCR 寄存器：DBL=3，DBA=0XE

使能定时器更新 DMA 请求（UDE=1）

使能定时器

使能 DMA 通道

这此示例适用于每个 CCRx 寄存器更新一次的情况。如果每个 CCRx 都更新两次，数据的传输个数就需要设置为 6。举例假设 RAM 的 buffer 中包含了 data1, data2, data3, data4, data5 and data6。数据依次传输到 CCRx 中：第一次更新 DMA 请求，data1 传输给 CCR2，data2 传输给 CCR3，data3 传输给 CCR4；第二次更新 DMA 请求，data4 传输给 CCR2，data5 传输给 CCR3，data6 传输给 CCR4。

注：空值可以写入到保留寄存器

19.2.30 TIM1/TIM8 DMA 请求

TIM1/TIM8 可以生成 DMA 请求，如下表所示：

表 19-6 DMA 请求

DMA 请求源	DMA 使能控制位
更新	UDE
比较/捕获 1	CC1DE
比较/捕获 2	CC2DE
比较/捕获 3	CC3DE
比较/捕获 4	CC4DE
COM	COMDE
触发	TDE

19.2.31 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。详见第 20.2.22 节：定时器同步。

19.2.32 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_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

调试模式下的行为可以使用调试支持模块（DBG）中每个定时器的专用配置进行编程

为了安全的考虑，当计数器停止时，输出被禁止（如果 MOE 被复位了）。输出即可被强制到一个无效电平（OSSI=1），或者由 GPIO 控制器接管（OSSI=0），典型的是强制为高阻。

详见 DBG 模块描述

19.2.33 TIM1/TIM8 中断

中断事件	事件标志	中断使能控制位
更新	UIF	UIE
比较/捕获 1	CC1IF	CC1IE
比较/捕获 2	CC2IF	CC2IE
比较/捕获 3	CC3IF	CC3IE
比较/捕获 4	CC4IF	CC4IE

COM	COM	COMIE
触发	TIF	TIE
编码器索引	IDXF	IDXIE
编码器方向	DIRF	DIRIE
刹车	BIF	BIE
刹车 2	B2IF	
系统刹车	SBIF	
编码器索引错误	IERRF	IERRIE
编码器传输错误	TERRF	TERRIE

19.3 寄存器描述

19.3.1 TIM1 和 TIM8 控制寄存器 1（TIMx_CR1）

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
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**
Res			DITHEN	UIFREMAP	Res	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:13	保留	-	-	-
12	DITHEN	RW	0	抖动使能（Dithering enbale） 0：抖动关闭 1：抖动打开 注：该位只能在 CEN 位复位时才能修改
11	UIFREMAP	RW	0	UIF 状态位重映射使能（UIF status bit remapping） 0：无重映射，UIF 状态位不复制到 TIMx_CNT 寄存器的第 31 位 1：重映射，UIF 状态位复制到 TIMx_CNT 寄存器的第 31 位
10	保留	-	-	-
9:8	CKD	RW	2’h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间的分频比和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:5	CMS	RW	2’h0	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(TIMx_CCMRx 寄存器中 CCxS=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(TIMx_CCMRx 寄存器中 CCxS=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(TIMx_CCMRx 寄存器中 CCxS=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(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	RW	0	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。
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

19.3.2 TIM1 和 TIM8 控制寄存器 2（TIMx_CR2）

Address offset:0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						MMS[3]	Res	MMS2 [3:0]				Res	OIS6	Res	OIS5
						RW		RW	RW	RW	RW		RW		RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res	CCPC
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	-
25	MMS[3]	RW	0	详见 MMS[2:0]描述
24	保留	-	-	-
23:20	MMS2[3:0]	RW	4'h0	主模式选择 2（Master mode selection 2） 这四位用于选择同步信息（TRGO2）发送给 ADC，可能的组合如 0000：复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO2)。如果触发输入(复位模式下的从模式控制器)产生复位，则 TRGO 上的信号相对实际的复位 会有一个延迟。 0001：使能 –计数器使能信号 CNT_EN 可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 CEN 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 TIMx_SMCR 寄存器中 MSM 位的描述)。 0010：更新– 更新事件被选为触发输出(TRGO2)。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。 0011：比较脉冲 – 一旦发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即是它已经为高)，触发输出送出一个正脉冲(TRGO2)。 0100：比较 - OC1REFC 信号被用于作为触发输出(TRGO2)。 0101：比较 – OC2REFC 信号被用于作为触发输出(TRGO2)。 0110：比较 – OC3REFC 信号被用于作为触发输出(TRGO2)。 0111：比较 – OC4REFC 信号被用于作为触发输出(TRGO2)。 1000：比较 – OC5REFC 信号被用于作为触发输出(TRGO2)。 1001：比较 – OC6REFC 信号被用于作为触发输出(TRGO2)。 1010：比较脉冲 - OC4REFC 信号上升沿或下降沿被用于作为触发输出(TRGO2)。 1011：比较脉冲 – OC6REFC 信号上升沿或下降沿被用于作为触发输出(TRGO2) 1100：比较脉冲 - OC4REFC 信号上升沿或 OC6REFC 信号上升沿用于作为触发输出(TRGO2) 1101：比较脉冲 – OC4REFC 信号上升沿或 OC6REFC 信号下降沿用于作为触发输出(TRGO2) 1110：比较脉冲 – OC5REFC 信号上升沿或 OC6REFC 信号上升沿用于作为触发输出(TRGO2) 1111：比较脉冲 – OC5REFC 信号上升沿或 OC6REFC 信号下降沿用于作为触发输出(TRGO2) 注：从定时器或 ADC 的时钟必须在接收主定时器事件之前就打开，并在接收时不要改变
19	保留	-	-	-
18	OIS6	RW	0	输出空闲状态 6(OC6 输出)。参见 OIS1 位。

17	保留	-	-	-
16	OIS5	RW	0	输出空闲状态 5(OC5 输出)。参见 OIS1 位。
15	OIS4N	RW	0	输出空闲状态 4(OC4N 输出)。参见 OIS1N 位。
14	OIS4	RW	0	输出空闲状态 4(OC4 输出)。参见 OIS1 位。
13	OIS3N	RW	0	输出空闲状态 3(OC3N 输出)。参见 OIS1N 位。
12	OIS3	RW	0	输出空闲状态 3(OC3 输出)。参见 OIS1 位。
11	OIS2N	RW	0	输出空闲状态 2(OC2N 输出)。参见 OIS1N 位。
10	OIS2	RW	0	输出空闲状态 2(OC2 输出)。参见 OIS1 位。
9	OIS1N	RW	0	输出空闲状态 1(OC1N 输出)（Output Idle state 1) 0：当 MOE=0 时，死区后 OC1N=0； 1：当 MOE=0 时，死区后 OC1N=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
8	OIS1	RW	0	输出空闲状态 1(OC1 输出)（Output Idle state 1) 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0； 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
7	TI1S	RW	0	TI1 选择（TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
6:4	MMS[2:0]	RW	3’h0	主模式选择（Master mode selection) 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： 0000：复位 – TIMx_EGR 寄存器的 UG 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。 0001：使能 – 计数器使能信号 CNT_EN 可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 CEN 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 TIMx_SMCR 寄存器中 MSM 位的描述)。 0010：更新 – 更新事件被选为触发输出(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。 0011: 比较脉冲 – 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 0100：比较 – OC1REFC 信号被用于作为触发输出(TRGO)。 0101：比较 – OC2REFC 信号被用于作为触发输出(TRGO)。 0110：比较 – OC3REFC 信号被用于作为触发输出(TRGO)。 0111：比较 – OC4REFC 信号被用于作为触发输出(TRGO)。 1000：编码器时钟输出-编码器时钟信号作为触发输出（TRGO）。该编码值仅在 SMS[3:0]的值为 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111 时有效。其他的 SMS 值可能导致预期之外的效果。 1001-1111：保留 注意： 1.从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。

				2.若主从定时器不在同一总线上，主模式应该配置为能被从定时器采到的宽度。
3	CCDS	RW	0	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。
2	CCUS	RW	0	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(CCPC=1)，只能通过设置 COM 位更新它们； 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 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。
1	保留	-	-	-
0	CCPC	RW	0	捕获/比较预装载控制位（Capture/compare preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新（COMG 位设置或 tim_trgi 上检测到的上升沿，具体取决于 CCUS 位）。 注：该位只对具有互补输出的通道起作用。

19.3.3 TIM1 和 TIM8 从模式控制寄存器（TIMx_SMCR)

Address offset:0x08

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						SMSPS	SMSPE	Res		TS[4:3]		Res			SMS[3]
						RW	RW			RW	RW				RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	-
25	SMSPS	RW	0	SMS 预加载源（SMS preload source） 该位选择事件是否可以触发 SMS[3:0] 0：传输由定时器的更新事件触发 1：传输由索引事件触发
24	SMSPE	RW	0	SMS 预加载使能（SMS preload enable） 该位选择 SMS[3:0]是否可以预加载 0：SMS[3:0]可以被预加载 1：SMS[3:0]不可以被预加载
23:22	保留	-	-	-
21:20	TS[4:3]	RW	2’h0	详见 TS[2:0]描述
19:17	保留	-	-	-
16	SMS[3]	RW	0	详见 SMS[2:0]描述
15	ETP	RW	0	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。

14	ECE	RW	0	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 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=00111)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(TS 位不能是‘00111’)。 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。
13:12	ETPS	RW	2’h0	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 TIMxCLK 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。
11:8	ETF	RW	4’h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
7	MSM	RW	0	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。
6:4	TS	RW	3’h0	触发选择（Trigger selection) 选择用于同步计数器的触发输入。 00000：内部触发 0(ITR0) 00001：内部触发 1(ITR1)

				00010：内部触发 2(ITR2) 00011：内部触发 3(ITR3) 00100：TI1 边沿检测（TI1F_ED） 00101：滤波后的定时器输入 1（TI1FP1） 00110：滤波后的定时器输入 2（TI1FP2） 00111：外部触发输入（ETRF） 01000：内部触发 4(ITR4) 01001：内部触发 5(ITR5) 01010：内部触发 6(ITR6) 01011：内部触发 7(ITR7) 01100：内部触发 8(ITR8) 01101：内部触发 9(ITR9) 01110：内部触发 10(ITR10) 01111：内部触发 11(ITR11) 10000：内部触发 12(ITR12) 10001：内部触发 13(ITR13) 10010：内部触发 14(ITR14) 10011：内部触发 15(ITR15) 10100：内部触发 16(ITR16) 10101：内部触发 17(ITR17) 10110：内部触发 18(ITR18) 其它：保留 更多有关 ITRx 的细节，参见系统级联表 注：这些位只能在未用到(如 SMS=0000)时被改变，以避免在改变时产生错误的边沿检测。
3	OCCS	RW	0	OCREF 清除选择位（OCREF clear selection） 这位选择了 OCREF 的清除源。 0：OCREF_CLR_INT 连接 OCREF_CLR 输入 1：OCREF_CLR_INT 连接 ETRF
2:0	SMS	RW	3’h0	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) 0000：关闭从模式 – 如果 CEN=1，则预分频器直接由内部时钟驱动。 0001: 正交编码器模式 1，x2 模式– 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 0010: 正交编码器模式 2，x2 模式– 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 0011：正交编码器模式 3，x4 模式–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。 0100：复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 0101：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。 0110: 触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。

				0111：外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 1000：复位+触发组合模式：-触发输入（TRGI）的上升沿初始化计数器，生成寄存器更新并启动计数器 1001：门控+复位组合模式：-触发输入（TRGI）为高时计数器时钟使能。一旦触发信号拉低时计数器就停止并复位。计数器的开始和停止均可控。 1010：编码器模式：时钟+方向，x2 模式 1011：编码器模式：时钟+方向，x1 模式，TI2FP2 的敏感性由 CC2P 设置 1100：编码器模式：定向时钟，x2 模式 1101：编码器模式：定向时钟，x1 模式，TI1FP1，TI2FP2 的敏感性由 CC1P 和 CC2P 设置 1110：正交编码器模式：x1 模式。仅在 TI1FP1 的边沿计数，沿敏感性由 CC1P 设置 1111：正交编码器模式：x1 模式。仅在 TI2FP2 的边沿计数，沿敏感性由 CC2P 设置 注：如果 TI1F_ED 被选为触发输入(TS=00100)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：必须在接收主定时器事件之前，使能从设备（定时器，ADC）来接收 TRGO 或 TRGO2，并且时钟频率不能再接收过程中改变 注：在编码器模式下，不要使用 UEV 作为 tim_trgo 输出信号，（即 mms 不能配置为 010）

19.3.4 TIM1 和 TIM8 DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TERRIE	IERRIE	DIRIE	IDXIE	Res			
								RW	RW	RW	RW				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	TERRIE	RW	0	传输错误中断使能（Transition error interrupt enable） 0：传输错误中断不使能 1：传输错误中断使能
22	IERRIE	RW	0	索引错误中断使能（Index error interrupt enable） 0：索引错误中断不使能 1：索引错误中断使能
21	DIRIE	RW	0	方向改变中断使能（Direction change interrupt enable） 0：方向改变中断不使能 1：方向改变中断使能
20	IDXIE	RW	0	索引中断使能（Index interrupt enable） 0：索引中断不使能

				1：索引中断使能
19:15	保留	-	-	-
14	TDE	RW	0	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。
13	COMDE	RW	0	允许 COM 的 DMA 请求（COM DMA request enable) 0：禁止 COM 的 DMA 请求; 1：允许 COM 的 DMA 请求。
12	CC4DE	RW	0	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。
11	CC3DE	RW	0	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。。
10	CC2DE	RW	0	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。
9	CC1DE	RW	0	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。。
8	UDE	RW	0	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。
7	BIE	RW	0	允许刹车中断（Break interrupt enable) 0：禁止刹车中断; 1：允许刹车中断。
6	TIE	RW	0	触发中断使能（Trigger interrupt enable) 0：禁止触发中断; 1：使能触发中断。
5	COMIE	RW	0	允许 COM 中断（COM interrupt enable) 0：禁止 COM 中断; 1：允许 COM 中断。
4	CC4IE	RW	0	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。
3	CC3IE	RW	0	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。
2	CC2IE	RW	0	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。
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断;

				1：允许更新中断。

19.3.5 TIM1 和 TIM8 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		IC4IF	IC3IF	IC2IF	IC1IF	IC4IR	IC3IR	TERRF	IERRF	DIRF	IDXF	IC2IR	IC1IR	CC6IF	CC5IF
		RC_W0													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		RC_W0													

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	IC4IF	RC_W0	0	下降沿捕获 4 标志 参见 IC1IF 描述。
28	IC3IF	RC_W0	0	下降沿捕获 3 标志 参见 IC1IF 描述。
27	IC2IF	RC_W0	0	下降沿捕获 2 标志 参见 IC1IF 描述。
26	IC1IF	RC_W0	0	下降沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生下降沿捕获事件。
25	IC4IR	RC_W0	0	上升沿捕获 4 标志 参见 IC1IR 描述。
24	IC3IR	RC_W0	0	上升沿捕获 3 标志 参见 IC1IR 描述。
23	TERRF	RC_W0	0	传输错误中断标志（Transition error interrupt flag） 当编码器模式中出现传输错误，该标志由硬件置位。可通过软件清零或者写清零 0：未检测到编码器传输错误 1：检测到编码器传输错误
22	IERRF	RC_W0	0	索引错误中断标志（Index error interrupt flag） 当检测到索引出现错误，该标志由硬件置位。可通过软件清零或者写清零
21	DIRF	RC_W0	0	方向改变中断标志（Direction change interrupt flag） 当编码器模式中出现方向改变（TIMx_CR 中的 DIR 位改变），该标志由硬件置位。可通过软件清零或者写清零 0：未检测到方向改变 1：检测到方向改变
20	IDXF	RC_W0	0	索引中断标志（Index interrupt flag） 当检测到索引事件，该标志由硬件置位。可通过软件清零或者写清零 0：无索引事件发生 1：有索引事件发生
19	IC2IR	RC_W0	0	上升沿捕获 2 标志 参见 IC1IR 描述。

18	IC1IR	RC_W0	0	上升沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生上升沿捕获事件。
17	CC6IF	RC_W0	0	捕获/比较 6 中断标记（Capture/Compare 6 interrupt flag) 参考 CC1IF 描述。 注：通道 6 只能配置为输出
16	CC5IF	RC_W0	0	捕获/比较 5 中断标记（Capture/Compare 5 interrupt flag) 参考 CC1IF 描述。 注：通道 5 只能配置为输出
15:14	保留	-	-	-
13	SBIF	RC_W0	0	系统刹车中断标志（System break interrupt flag） 当系统刹车输入一旦有效，该标志信号会硬件置位。当系统刹车输入无效时可通过软件清零 该标志必须复位来重启 PWM 操作 0：无刹车事件发生 1：系统刹车输入处于有效。如果 TIMx_DIER 寄存器的 BIE 位=1 则会产生中断
12	CC4OF	RC_W0	0	捕获/比较 4 重复捕获标记（Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	RC_W0	0	捕获/比较 3 重复捕获标记（Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	RC_W0	0	捕获/比较 2 重复捕获标记（Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	RC_W0	0	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 TIMx_CCR1 寄存器时，CC1IF 的状态已经为‘1’。
8	B2IF	RC_W0	0	刹车 2 中断标记（Break2 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。如果 TIM_DIER 的 BIE=1 则产生中断
7	BIF	RC_W0	0	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。如果 TIM_DIER 的 BIE=1 则产生中断
6	TIF	RC_W0	0	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。

5	COMIF	RC_W0	0	COM 中断标记（COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新)该位由硬件置‘1’。它由软件清‘0’。 0：无 COM 事件产生； 1：COM 中断等待响应。
4	CC4IF	RC_W0	0	捕获/比较 4 中断标记（Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	RC_W0	0	捕获/比较 3 中断标记（Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记（Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	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： 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 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，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=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 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

19.3.6 TIM1 和 TIM8 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	B2G	W	0	产生刹车 2 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 2 事件。此时 MOE=0、B2IF=1
7	BG	W	0	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 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
6	TG	W	0	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：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5	COMG	W	0	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 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 注：该位只对拥有互补输出的通道有效。
4	CC4G	W	0	产生捕获/比较 4 事件（Capture/Compare 4 generation) 参考 CC1G 描述。
3	CC3G	W	0	产生捕获/比较 3 事件（Capture/Compare 3 generation) 参考 CC1G 描述。
2	CC2G	W	0	产生捕获/比较 2 事件（Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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 TIMx_ARR 的值。

19.3.7 TIM1 和 TIM8 捕获/比较模式控制寄存器 1（TIMx_CCMR1）

Address offset:0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							OC2M[3]	Res.							OC1M[3]

							RW								RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

19.3.7.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC2M[3]	RW	0	详见 OC2M[2:0]描述
23:17	保留	-	-	-
16	OC1M[3]	RW	0	详见 OC1M[2:0]描述
15	OC2CE	RW	0	输出比较 2 清零使能（Output Compare 2 clear enable)
14:12	OC2M	RW	3'h0	输出比较 2 模式（Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能（Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能（Output Compare 2 fast enable)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	输出比较 1 模式（OutputCompare1mode) 这些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 CC1P、CC1NP 位。 0000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用; 0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 0011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 0100：强制为无效电平。强制 OC1REF 为低。 0101：强制为有效电平。强制 OC1REF 为高。 0110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 0111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。

				1000：可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下，通道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为无效 1001：可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 1010：保留 1011：保留 1100：组合 PWM 模式 1–OC1REF 有与 PWM 模式 1 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 或逻辑组成 1101：组合 PWM 模式 2–OC1REF 有与 PWM 模式 2 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 与逻辑组成 1110：非对称 PWM 模式 1：OC1REF 有与 PWM 模式 1 中相同的行为，向上计数时 OC1REFC 输出 OC1REF，向下计数时 OC1REFC 输出 OC2REF 1111：非对称 PWM 模式 2：OC1REF 有与 PWM 模式 2 中相同的行为，向上计数时 OC1REFC 输出 OC1REF，向下计数时 OC1REFC 输出 OC2REF 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式中，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。 注 3：当通道有互补输出时，该位段可以预加载。如果 TIMx_CR2 的 CCPC 位置 1，然后仅当 COM 事件生成时 OC1M 有效位从预加载中更新新的值
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 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(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=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(TIMx_CR1 寄存器的 OPM=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。
2	OC1FE	RW	0	输出比较 1 快速使能（Output Compare 1 fast enable) 该位用于加快输出对触发输入事件的响应。必须在单脉冲模式（TIMx_CR1 寄存器的 OPM 位置 1）中使用，当触发到来时实现脉冲快速输出。 0：根据计数器与 CCR1 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象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。仅能用于 PWM 模式 1 和 PWM 模式 2
1:0	CC1S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

19.3.7.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC2F	RW	4'h0	输入捕获 2 滤波器（Input capture 2 filter)
11:10	IC2PSC	RW	2'h0	输入/捕获 2 预分频器（Input capture 2 prescaler)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F	RW	4'h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。

1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

19.3.8 TIM1 和 TIM8 捕获/比较模式控制寄存器 2（TIMx_CCMR2）

Address offset:0x1C

Reset value：0x0000 0000

参看以上 CCMR1 寄存器的描述

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							OC4M[3]	Res.							OC3M[3]
							RW								RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

19.3.8.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC4M[3]	RW	0	详见 OC4M[2:0]描述
23:17	保留	-	-	-
16	OC3M[3]	RW	0	详见 OC3M[2:0]描述
15	OC4CE	RW	0	输出比较 4 清零使能（Output Compare 4 clear enable)
14:12	OC4M	RW	3'h0	输出比较 4 模式（Output Compare 4 mode)
11	OC4PE	RW	0	输出比较 4 预装载使能（Output Compare 4 preload enable)
10	OC4FE	RW	0	输出比较 4 快速使能（Output Compare 4 fast enable)
9:8	CC4S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7	OC3CE	RW	0	输出比较 3 清'0'使能（Output Compare 1clear enable)
6:4	OC3M	RW	3'h0	输出比较 3 模式（OutputCompare3mode) 这些位定义了输出参考信号 OC3REF 的动作,而 OC3REF 决定了 OC3、OC3N 的值。OC3REF 是高电平有效，而 OC3、OC3N 的有效电平取决于 CC3P、CC3NP 位。

				0000：冻结。输出比较寄存器 TIMx_CCR3 与计数器 TIMx_CNT 间的比较对 OC3REF 不起作用； 0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR3)相同时，强制 OC3REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR3)相同时，强制 OC3REF 为低。 0011：翻转。当 TIMx_CCR3=TIMx_CNT 时，翻转 OC3REF 的电平。 0100：强制为无效电平。强制 OC3REF 为低。 0101：强制为有效电平。强制 OC3REF 为高。 0110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR3 时通道 3 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR3 时通道 3 为无效电平(OC3REF=0)，否则为有效电平(OC3REF=1)。 0111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR3 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR3 时通道 1 为有效电平，否则为无效电平。 1000：可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下，通道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为不活动 1001：可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 1010：比较脉冲：当 CCR3 匹配事件发生时在 OC3REF 上产生脉冲，脉冲可由 TIMx_ECR 的 PWPRSC[2:0]和 PW[7:0]配置 1011：方向输出：OC3REF 信号由 DIR 位覆盖 1100：组合 PWM 模式 1–OC3REF 有与 PWM 模式 1 中相同的行为，OC3REFC 是由 OC3REF 和 OC4REF 或逻辑组成 1101：组合 PWM 模式 2–OC3REF 有与 PWM 模式 2 中相同的行为，OC3REFC 是由 OC3REF 和 OC4REF 与逻辑组成 1110：非对称 PWM 模式 1：OC3REF 有与 PWM 模式 1 中相同的行为，向上计数时 OC3REFC 输出 OC3REF，向下计数时 OC3REFC 输出 OC4REF 1111：非对称 PWM 模式 2：OC3REF 有与 PWM 模式 2 中相同的行为，向上计数时 OC3REFC 输出 OC3REF，向下计数时 OC3REFC 输出 OC4REF 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC3S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式中，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，OC3REF 电平才改变。

				注 3：当通道有互补输出时，该位段可以预加载。如果 TIMx_CR2 的 CCPC 位置 1，然后仅当 COM 事件生成时 OC3M 有效位从预加载中更新的值 注 4：使用可重复 OPM 模式时，计数模式不要配置为中央计数模式
3	OC3PE	RW	0	输出比较 3 预装载使能（Output Compare 3 preload enable)
2	OC3FE	RW	0	输出比较 3 快速使能（Output Compare 3 fast enable)
1:0	CC3S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

19.3.8.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC4F	RW	4'h0	输入捕获 4 滤波器（Input capture 4 filter)
11:10	IC4PSC	RW	2'h0	输入/捕获 4 预分频器（Input capture 4 prescaler)
9:8	CC4S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7:4	IC3F	RW	4'h0	输入捕获 3 滤波器（Input capture 3 filter)
3:2	IC3PSC	RW	2'h0	输入/捕获 3 预分频器（Input capture 3 prescaler)
1:0	CC3S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

19.3.9 TIM1 和 TIM8 捕获/比较使能寄存器（TIMx_CCER）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										CC6P	CC6E	Res		CC5P	CC5E
										RW	RW			RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21	CC6P	RW	0	输入/捕获 6 输出极性（Capture/Compare 4 output polarity) 参考 CC1P 的描述。
20	CC6E	RW	0	输入/捕获 6 输出使能（Capture/Compare 4 output enable) 参考 CC1E 的描述。
19:18	保留	-	-	-
17	CC5P	RW	0	输入/捕获 5 输出极性（Capture/Compare 4 output polarity) 参考 CC1P 的描述。
16	CC5E	RW	0	输入/捕获 5 输出使能（Capture/Compare 4 output enable) 参考 CC1E 的描述。
15	CC4NP	RW	0	输入/捕获 4 互补输出极性（Capture/Compare 3 output polarity) 参考 CC1NP 的描述。
14	CC4NE	RW	0	输入/捕获 4 互补输出使能（Capture/Compare 3 output enable) 参考 CC1NE 的描述。
13	CC4P	RW	0	输入/捕获 4 输出极性（Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	RW	0	输入/捕获 4 输出使能（Capture/Compare 4 output enable) 参考 CC1E 的描述。
11	CC3NP	RW	0	输入/捕获 3 互补输出极性（Capture/Compare 3 output polarity) 参考 CC1NP 的描述。
10	CC3NE	RW	0	输入/捕获 3 互补输出使能（Capture/Compare 3 output enable) 参考 CC1NE 的描述。
9	CC3P	RW	0	输入/捕获 3 输出极性（Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	CC3E	RW	0	输入/捕获 3 输出使能（Capture/Compare 3 output enable) 参考 CC1E 的描述。
7	CC2NP	RW	0	输入/捕获 2 互补输出极性（Capture/Compare 2 output polarity) 参考 CC1NP 的描述。
6	CC2NE	RW	0	输入/捕获 2 互补输出使能（Capture/Compare 2 output enable) 参考 CC1NE 的描述。
5	CC2P	RW	0	输入/捕获 2 输出极性（Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能（Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性（Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效; 1：OC1N 低电平有效。

				注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出)则该位不能被修改。
2	CC1NE	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿： TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。 注： 1.对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。 2.一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2，则该位不能被修改。
0	CC1E	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 TIMx_CCR1 寄存器。 0：捕获禁止; 1：捕获使能。 注: 对于互补输出通道,这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表 19-7 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	
MOE 位	OSSI 位	OSSR 位	CCxE 位	CCxNE 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=OCREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF+极性， OCx=OCREF xor CCxP, 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（取反）+极性+死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开） OCx=CCxP, OCx=0 OCx_EN=0	输出禁止（与定时器断开） OCxN=CCxNP, OCx=0 OCxN_EN=0
		1	0	1	关闭状态（输出使能且为无效电平） OCx=CCxP, OCx_EN=1	OCxREF+极性， OCxN=OCREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+极性， OCx=OCREF xor CCxP, OCx_EN=1	关闭状态（输出使能且为无效电平） OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF+极性+死区，， OCx_EN=1	OCxREF（取反）+极性+死区， OCxN_EN=1
0	0	x	x	x	输出禁止（与定时器断开）	
	1		0	0		
	1		0	1	关闭状态（输出使能但为无效电平） 异步的: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCx_EN=1; (如果刹车或刹车 2 触发了) 然后若时钟存在（仅刹车 1 触发时有效，不需要刹车 2）：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平 注：刹车 2 仅用于 OSSI=OSSR=1	
	1		1	0		
	1		1	1		

尤其当死区有效时，输出总是按照死区内输出的规则输出，尽管此时 moe 可能已经被软件或硬件的方式置为 1。如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx，OISxN，CCxP 和 CCxNP 都必须清零。

19.3.10 TIM1 和 TIM8 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

CNT															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	UIFCPY	R	0	该位为只读寄存器，将 TIMx_ISR 寄存器的 UIF 位复制过来。如果 TIMx_CR1 中的 UIF 重映射位被复位了，该 bit 位保留并读出来为 0
30:16	保留	-	-	-
15:0	CNT	RW	16'h0	计数器的值（Counter value)

19.3.11 TIM1 和 TIM8 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16'h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

19.3.12 TIM1 和 TIM8 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	ARR	RW	20'hFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。 无抖动模式（DITHEN=0）：该寄存器保持自动加载值 抖动模式（DITHEN=1）： 该寄存器保持整数部分 ARR[19:4]，ARR[3:0]包含抖动部分

19.3.13 TIM1 和 TIM8 重复计数寄存器（TIMx_RCR）

Address offset:0x30

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	REP	RW	16'h0	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。 每次向下计数器达到 0，会产生一个更新事件并且重复计数器重新从 REP 值开始计数。对 TIMx_RCR 寄存器写入的新值只在下次更新事件发生时才起作用。 这意味着在 PWM 模式中，（REP+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;

19.3.14 TIM1 和 TIM8 捕获/比较寄存器 1（TIMx_CCR1）

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR1[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR1	RW/RO	20'h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出: CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 无抖动模式（DITHEN=0）: 该寄存器保持 CCR1[15:0]的比较值，CCR1[19:16]位被复位 抖动模式（DITHEN=1）: 该寄存器保持整数部分 CCR1[19:4]，CCR1[3:0]包含抖动部分 若 CC1 通道配置为输入: CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读 无抖动模式（DITHEN=0）:

				该寄存器保存捕获值到 CCR1[15:0]，CCR1[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR1[19:4]，CCR1[3:0]位被复位

19.3.15 TIM1 和 TIM8 捕获/比较寄存器 2（TIMx_CCR2）

Address offset:0x38

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR2[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR2	RW/RO	20’h0	捕获/比较通道 2 的值（Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC2PE 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 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR2[15:0]的比较值，CCR2[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR2[19:4]，CCR2[3:0]包含抖动部分 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。TIMx_CCR2 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR2[15:0]，CCR2[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR2[19:4]，CCR2[3:0]位被复位

19.3.16 TIM1 和 TIM8 捕获/比较寄存器 3（TIMx_CCR3）

Address offset:0x3C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR3[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR3	RW/RO	20'h0	捕获/比较通道 3 的值（Capture/Compare 3 value) 若 CC3 通道配置为输出: CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC3PE 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 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR3[15:0]的比较值，CCR3[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR3[19:4]，CCR3[3:0]包含抖动部分 若 CC3 通道配置为输入: CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。 TIMx_CCR3 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR3[15:0]，CCR3[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR3[19:4]，CCR3[3:0]位被复位

19.3.17 TIM1 和 TIM8 捕获/比较寄存器 4（TIMx_CCR4)

Address offset:0x40

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR4[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR4	RW/RO	20'h0	捕获/比较通道 4 的值（Capture/Compare 4 value) 若 CC4 通道配置为输出: CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC4PE 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 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR4[15:0]的比较值，CCR4[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR4[19:4]，CCR4[3:0]包含抖动部分 若 CC4 通道配置为输入:

				CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。 TIMx_CCR4 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR4[15:0]，CCR4[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR4[19:4]，CCR4[3:0]位被复位

19.3.18 TIM1 和 TIM8 刹车和死区寄存器（TIMx_BDTR）

Address offset:0x44

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

注释：根据锁定设置，BKBID/BK2BID/BK2P/BK2E/BK2F/BKF/AOE/BKP/BKE/OSSI、OSSR 和 DTG[7:0]位均可被写保护，有必要在第一次写入 TIMx_BDTR 寄存器时对它们进行配置。

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	BK2BID	RW	0	刹车 2 双向性（Break2 bidirectional） 参考 BKBID 描述
28	BKBID	RW	0	刹车双向性（Break bidirectional） 0：刹车输入 BRK 在输入模式中 1：刹车输入 BRK 在双向模式中 在双向模式中（BKBID 设置为 1），刹车输入被既配置为输入模式也是开漏输出模式。任何有效的刹车事件会将刹车输入电平拉低，向外部设备指示内部发生了刹车事件 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
27	BK2DSRM	RW	0	刹车 2 解除（Break2 disarm） 参考 BKDSRM 描述
26	BKDSRM	RW	0	刹车解除（Break disarm） 0：刹车输入 BRK 启用 1：刹车输入 BRK 解除 当没有刹车源时该位会被硬件清零 BKDSRM 位必须由软件置位来释放双向输出控制（开漏输出高阻状态），然后轮询它直到被硬件复位，表示故障状态已经消失。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
25	BK2P	RW	0	刹车 2 极性（Break 2 polarity） 0：刹车输入 BRK2 有效电平为低电平 1：刹车输入 BRK2 有效电平为高电平 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。

24	BK2E	RW	0	刹车 2 使能（Break 2 enable） 该位使能刹车 2 的保护 0：刹车 2 功能关闭 1：刹车功能开启 注：刹车 2 必须在 OSSR=OSSI=1 时才能使用 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
23:20	BK2F[3:0]	RW	4’h0	刹车 2 滤波（Break 2 filter） 这些位定义了对 BRK2 信号采样的频率和对 BRK2 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，BRK2 是异步的 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。
19:16	BKF[3:0]	RW	4’h0	刹车滤波（Break filter） 这些位定义了对 BRK 信号采样的频率和对 BRK 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，BRK2 是异步的 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6

				1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。
15	MOE	RW	0	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 AOE 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(TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 TIMx 捕获/比较使能寄存(TIMx_CCER)。
14	AOE	RW	0	自动输出使能（Automatic output enable) 0：MOE 只能被软件置'1'； 1：MOE 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(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。
13	BKP	RW	0	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(TIMx_BDTR 寄存器中的 LOCK 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。
12	BKE	RW	0	刹车功能使能（Break enable) 0：禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1：开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，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。
11	OSSR	RW	0	运行模式下"关闭状态"选择（Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下"关闭状态"选择（Off-state selection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOCK	RW	2'h0	锁定设置（Lock 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 TIMx_BDTR 寄存器的 DTG、BKBID、BK2BID、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 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 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 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 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。
7:0	DTG	RW	8'h0	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： DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg，Tdtg = TDTS； DTG[7:5]=10x => DT=(64+DTG[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTG[7:5]=110 => DT=(32+DTG[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTG[7:5]=111 => DT=(32+DTG[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位。

19.3.19 TIM1 和 TIM8 捕获/比较寄存器 5（TIMx_CCR5)

Address offset:0x48

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res									CCR5[19:16]			
RW	RW	RW										RW/RO			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR5[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31	GC5C3	RW	0	分组通道 5 和通道 3（Group channel 5 and channel 3） 通道 3 输出变形 0：OC5REF 对 OC3REFC 无影响 1：OC3REFC 由 OC5REF 和 OC3REF 与逻辑组成

				该位可以立即产生影响或者预加载，当更新事件发生时才更新（如果 TIMxCCMR2 中选择了预加载特性） 注：也可以将这种变形应用在组合 PWM 信号
30	GC5C2	RW	0	分组通道 5 和通道 2（Group channel 5 and channel 2） 通道 2 输出变形 0：OC5REF 对 OC2REFC 无影响 1：OC2REFC 由 OC5REF 和 OC2REF 与逻辑组成 该位可以立即产生影响或者预加载，当更新事件发生时才更新（如果 TIMxCCMR1 中选择了预加载特性） 注：也可以将这种变形应用在组合 PWM 信号
29	GC5C1	RW	0	分组通道 5 和通道 1（Group channel 5 and channel 1） 通道 1 输出变形 0：OC5REF 对 OC1REFC 无影响 1：OC1REFC 由 OC5REF 和 OC1REF 与逻辑组成 该位可以立即产生影响或者预加载，当更新事件发生时才更新（如果 TIMxCCMR1 中选择了预加载特性，有使能） 注：也可以将这种变形应用在组合 PWM 信号
28:20	保留	-	-	-
19:0	CCR5	RW/RO	20’h0	捕获/比较通道 5 的值(Capture/Compare 5 value) 若 CC5 通道配置为输出： CCR5 包含了装入当前捕获/比较 5 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC5PE 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 5 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC5 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR5[15:0]的比较值，CCR5[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR5[19:4]，CCR5[3:0]包含抖动部分

19.3.20 TIM1 和 TIM8 捕获/比较寄存器 6（TIMx_CCR6）

Address offset:0x4C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR6[19:16]			
												RW/RO			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR6[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR6	RW/RO	20’h0	捕获/比较通道 6 的值(Capture/Compare 6 value) 若 CC6 通道配置为输出： CCR6 包含了装入当前捕获/比较 6 寄存器的值(预装载值)。

				如果在 TIMx_CCMR3 寄存器(OC6PE 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 6 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC6 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR6[15:0]的比较值，CCR6[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR6[19:4]，CCR6[3:0]包含抖动部分

19.3.21 TIM1 和 TIM8 捕获/比较模式控制寄存器 3（TIMx_CCMR3）

Address offset:0x50

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							OC6M[3]	Res							OC5M[3]
							RW								RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res		OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res	
RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC6M[3]	RW	0	详见 OC6M[2:0]描述
23:17	保留	-	-	-
16	OC5M[3]	RW	0	详见 OC5M[2:0]描述
15	OC6CE	RW	0	输出比较 6 清零使能（Output Compare 6 clear enable)
14:12	OC6M	RW	3’h0	输出比较 6 模式（Output Compare 6 mode) 参照 OC5M[3:0]描述
11	OC6PE	RW	0	输出比较 6 预装载使能（Output Compare 6 preload enable)
10	OC6FE	RW	0	输出比较 6 快速使能（Output Compare 6 fast enable)
9:8	保留	-	-	-
7	OC5CE	RW	0	输出比较 5 清’0’使能（Output Compare 5 clear enable)
6:4	OC5M	RW	3’h0	输出比较 5 模式（Output Compare 5 mode) 这些位定义了输出参考信号 OC5REF 的动作，而 OC5REF 决定了 OC5、OC5N 的值。OC5REF 是高电平有效，而 OC5、OC5N 的有效电平取决于 CC5P、CC5NP 位。 0000：冻结。输出比较寄存器 TIMx_CCR5 与计数器 TIMx_CNT 间的比较对 OC5REF 不起作用； 0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR5)相同时，强制 OC5REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR5)相同时，强制 OC5REF 为低。 0011：翻转。当 TIMx_CCR5=TIMx_CNT 时，翻转 OC5REF 的电平。 0100：强制为无效电平。强制 OC5REF 为低。 0101：强制为有效电平。强制 OC5REF 为高。

				0110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR5 时通道 5 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR5 时通道 5 为无效电平(OC5REF=0),否则为有效电平(OC5REF=1)。 0111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR5 时通道 5 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR5 时通道 5 为有效电平，否则为无效电平。 1000: 可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（TRGI 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下,通道处于无效状态，直到检测到触发事件（TRGI 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为不活动 1001: 可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（TRGI 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（TRGI 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 其它：保留 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式中，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，OC5REF 电平才改变。 注 3: 当通道有互补输出时，该位段可以预加载。如果 TIMx_CR2 的 CCPC 位置 1，然后仅当 COM 事件生成时 OC5M 有效位从预加载中更新新的值 注 4：使用可重复 OPM 模式时，计数模式不要配置为中央计数模式
3	OC5PE	RW	0	输出比较 5 预装载使能（Output Compare 5 preload enable)
2	OC5FE	RW	0	输出比较 5 快速使能（Output Compare 5 fast enable)
1:0	保留	-	-	-

19.3.22 TIM1 和 TIM8 死区时间寄存器 2（TIMx_DTR2）

Address offset:0x54

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	DTPE	RW	0	死区时间预加载使能（Deadtime preload enable） 0：死区时间的值不可以预加载 1：死区时间的值可以预加载 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
16	DTAE	RW	0	死区时间不对称使能（Deadtime asymmetric enable）

				0：上升沿和下降沿的死区时间相同，都由 DTG[7:0]配置 1：上升沿死区时间由 DTG[7:0]配置，下降沿死区时间由 DTGF[7:0]配置 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
15:8	保留	-	-	-
7:0	DTGF[7:0]	RW	8'h0	下降沿死区时间生成器（Dead-time falling edge generator setup） 该位段定义了插入互补输出时下降沿的死区时间，假设 DT 表示其持续时间： DTGF[7:5]=0xx => DTF=DTGF[7:0] × Tdtg，Tdtg = TDTS； DTGF[7:5]=10x => DTF=(64+DTGF[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTGF[7:5]=110 => DTF=(32+DTGF[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTGF[7:5]=111 => DTF=(32+DTGF[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位

19.3.23 TIM1 和 TIM8 编码器控制寄存器（TIMx_ECR）

Address offset:0x58

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					PWPRSC[2:0]			PW[7:0]							
					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
Res								IPOS		FIDX	Res		IDIR		IE
								RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	-
26:24	PWPRSC	RW	3'h0	脉冲宽度预分频（Pulse width prescaler） 该位设置脉冲生成器的时钟预分频，公式如下： t_{PWG}= $2^{(PWPRSC[2:0])}$ x tCK_INT
23:16	PW	RW	8'h0	脉冲宽度（Pulse width） 该位定义了脉冲持续时间，公式如下： t_{PW}=PW[7:0] x t_{PWG}
15:8	保留	-	-	-
7:6	IPOS	RW	2'h0	索引定位（Index positioning） 在正交编码器模式中（SMS[3:0] = 0001, 0010, 0011, 1110, 1111），该位用于指示 AB 输入在哪种配置下索引事件会复位计数器 00：AB=00 时，索引会复位计数器 01：AB=01 时，索引会复位计数器 10：AB=10 时，索引会复位计数器 11：AB=11 时，索引会复位计数器

				在定向时钟模式或时钟+方向模式中(SMS[3:0] = 1010, 1011, 1100, 1101)，该位段用于指示哪种电平索引会复位计数器。在定向时钟模式中，应用于两路的时钟输入。 x0：当时钟为 0 时，索引复位计数器 x1：当时钟为 1 时，索引复位计数器 注：IPOS[1]位无效
5	FIDX	RW	0	第一次索引（First index） 该位指示是否只考虑第一次索引 0：索引一直有效 1：仅第一次索引可以复位计数器
4:3	保留	-	-	-
1	IDIR	RW	0	索引方向（Index direction） 该位指示索引事件在哪个计数方向时复位计数器 00：无论哪个计数方向，索引均可以复位计数器 01：仅在向上计数时，索引可以复位计数器 10：仅在向下计数时，索引可以复位计数器 11：保留
0	IE	RW	0	索引使能（Index enable） 该位指示索引是否复位计数器 0：索引关闭 1：索引使能

19.3.24 TIM1 和 TIM8 输入选择寄存器（TIMx_TISEL）

Address offset:0x5C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				TI4SEL[3:0]				Res				TI3SEL[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res				TI2SEL[3:0]				Res				TI1SEL[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:24	TI4SEL	RW	4’h0	TI4 输入选择（tim_ti4[15:0] input） 0000：TIMx_CH4 其他：保留
23:20	保留	-	-	-
19:16	TI3SEL	RW	4’h0	TI3 输入选择（tim_ti3[15:0] input） 0000：TIMx_CH3 其他：保留
15:12	保留	-	-	-
11:8	TI2SEL	RW	4’h0	TI2 输入选择（tim_ti2[15:0] input） 0000：TIMx_CH2 其他：保留
7:4	保留	-	-	-
3:0	TI1SEL	RW	4’h0	TI1 输入选择（tim_ti1[15:0] input）

				0000：TIMx_CH1 0001：tim_ti1_in1 0010：tim_ti1_in2 0011：tim_ti1_in3 0100：tim_ti1_in4 其他：保留

19.3.25 TIM1 和 TIM8 备用功能选项寄存器 1（TIMx_AF1）

Address offset:0x60

Reset value:0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res														ETRSEL[3:2]	
														RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRSEL [1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	Res				BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:14	ETRSEL	RW	4’h0	外部触发源选择（etr_in source selection） 该位段用于选择 ETR 输入源 0000：TIMx_ETR 0001：tim_etr1 0010：tim_ etr2 0011：tim_ etr 3 0100：tim_ etr 4 0101：tim_ etr 5 0110：tim_ etr 6 0111：tim_ etr 7 1000：tim_ etr 8 1001：tim_ etr 9 1010：tim_ etr 10 1011：tim_ etr 11 1100：tim_ etr 12 1101：tim_ etr 13 1110：tim_ etr 14 1111：tim_ etr 15 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
13	BKCMP4P	RW	0	tim_brk_cmp4 输入极性 该位选择 tim_brk_cmp4 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp4 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：tim_brk_cmp4 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效）

				注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
12	BKCMP3P	RW	0	tim_brk_cmp3 输入极性 该位选择 tim_brk_cmp3 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp3 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp3 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
11	BKCMP2P	RW	0	tim_brk_cmp2 输入极性 该位选择 brk_cmp2 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp2 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp2 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
10	BKCMP1P	RW	0	tim_brk_cmp1 输入极性 该位选择 brk_cmp1 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp1 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp1 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
9	BKINP	RW	0	BKIN 输入极性（TIMx_BKIN input polarity） 该位选择 BKIN 输入极性的备用功能，必须与 BKP 极性位同时配置 0：BKIN 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：BKIN 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
8:5	保留	-	-	-
4	BKCMP4E	RW	0	tim_brk_cmp4 输入使能（tim_brk_cmp4 enable） 该位用于刹车输入时使能 tim_brk_cmp4。tim_brk_cmp4 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp4 输入关闭 1：tim_brk_cmp4 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
3	BKCMP3E	RW	0	tim_brk_cmp3 输入使能（tim_brk_cmp3 enable） 该位用于刹车输入时使能 tim_brk_cmp3。tim_brk_cmp3 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp3 输入关闭 1：tim_brk_cmp3 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
2	BKCMP2E	RW	0	tim_brk_cmp2 输入使能（tim_brk_cmp2 enable）

				该位用于刹车输入时使能 tim_brk_cmp2。tim_brk_cmp2 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp2 输入关闭 1：tim_brk_cmp2 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
1	BKCMP1E	RW	0	tim_brk_cmp1 输入使能（tim_brk_cmp1 enable） 该位用于刹车输入时使能 tim_brk_cmp1。tim_brk_cmp1 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp1 输入关闭 1：tim_brk_cmp1 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
0	BKINE	RW	1	BKIN 输入使能（TIMx_BKIN input enable） 该位用于刹车输入时使能 BKIN 的备用功能。BKIN 输入与其它刹车源进行或逻辑作为刹车输入。 0：BKIN 输入关闭 1：BKIN 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改。

19.3.26 TIM1 和 TIM8 备用功能选项寄存器 2（TIMx_AF2）

Address offset:0x64

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													OCRSEL[2:0]		
													RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		BK2 CMP4P	BK2 CMP3P	BK2 CMP2P	BK2 CMP1P	BK2INP	Res				BK2 CMP4E	BK2 CMP3E	BK2 CMP2E	BK CMP1E	BK2INE
		RW	RW	RW	RW	RW					RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	-
18:16	OCRSEL	RW	3’h0	OCREF 复位源选择（ocref_clr source selection） 该位段用于选择 ocref_clr 输入源 0000：ocref_clr0 0001：ocref_clr1 0010：ocref_clr2 0011：ocref_clr3 其它：保留 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
15:14	保留	-	-	-
13	BK2CMP4P	RW	0	tim_brk2_cmp4 输入极性 该位选择 tim_brk2_cmp4 输入极性，必须与 BK2P 极性位同时配置

				0：tim_brk2_cmp4 输入极性不翻转（BK2P=0 时低有效，BK2P=1 时高有效） 1：tim_brk2_cmp4 输入极性翻转（BK2P=0 时高有效，BK2P=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
12	BK2CMP3P	RW	0	tim_brk2_cmp3 输入极性 该位选择 tim_brk2_cmp3 输入极性，必须与 BK2P 极性位同时配置 0：tim_brk2_cmp3 输入极性不翻转（BK2P=0 时低有效，BK2P=1 时高有效） 1：tim_brk2_cmp3 输入极性翻转（BK2P=0 时高有效，BK2P=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
11	BK2CMP2P	RW	0	tim_brk2_cmp2 输入极性 该位选择 tim_brk2_cmp2 输入极性，必须与 BK2P 极性位同时配置 0：tim_brk2_cmp2 输入极性不翻转（BK2P=0 时低有效，BK2P=1 时高有效） 1：tim_brk2_cmp2 输入极性翻转（BK2P=0 时高有效，BK2P=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
10	BK2CMP1P	RW	0	tim_brk2_cmp1 输入极性 该位选择 tim_brk2_cmp1 输入极性，必须与 BK2P 极性位同时配置 0：tim_brk2_cmp1 输入极性不翻转（BK2P=0 时低有效，BK2P=1 时高有效） 1：tim_brk2_cmp1 输入极性翻转（BK2P=0 时高有效，BK2P=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
9	BK2INP	RW	0	BKIN2 输入极性（TIMx_BKIN2 input polarity） 该位选择 BKIN 输入极性的备用功能，必须与 BK2P 极性位同时配置 0：BKIN2 输入极性不翻转（BK2P=0 时低有效，BK2P=1 时高有效） 1：BKIN2 输入极性翻转（BK2P=0 时高有效，BK2P=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
8:5	保留	-	-	-
4	BK2CMP4E	RW	0	tim_brk2_cmp4 输入使能（tim_brk2_cmp4 enable） 该位用于刹车输入时使能 tim_brk2_cmp4。tim_brk2_cmp4 输入与其它刹车源进行或逻辑作为刹车输入。 0：brk2_cmp4 输入关闭 1：brk2_cmp4 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
3	BK2CMP3E	RW	0	tim_brk2_cmp3 输入使能（tim_brk2_cmp3 enable）

				该位用于刹车输入时使能 tim_brk2_cmp3。tim_brk2_cmp3 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk2_cmp3 输入关闭 1：tim_brk2_cmp3 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
2	BK2CMP2E	RW	0	tim_brk2_cmp2 输入使能（tim_brk2_cmp2 enable） 该位用于刹车输入时使能 tim_brk2_cmp2。tim_brk2_cmp2 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk2_cmp2 输入关闭 1：tim_brk2_cmp2 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
1	BK2CMP1E	RW	0	tim_brk2_cmp1 输入使能（tim_brk2_cmp1 enable） 该位用于刹车输入时使能 tim_brk2_cmp1。tim_brk2_cmp1 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk2_cmp1 输入关闭 1：tim_brk2_cmp1 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
0	BK2INE	RW	0	BKIN2 输入使能（TIMx_BKIN input enable） 该位用于刹车输入时使能 BKIN2 的备用功能。BKIN2 输入与其它刹车源进行或逻辑作为刹车输入。 0：BKIN2 输入关闭 1：BKIN2 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改。

19.3.27 TIM1 和 TIM8 DMA 控制寄存器（TIMx_DCR）

Address offset:0x3DC

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12:8	DBL	RW	5’h0	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 TIMx_DMAR 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 .....

				…… 11001：26 次传输 例：我们考虑这样的传输：DBL=7 字节，DBA=TIMx_CR1 - 如果 DBL=7 字节，DBA=TIMx_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址）+ DBA +（DMA 索引)，其中 DMA 索引 = DBL 其中(TIMx_CR1 的地址）+ DBA 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(TIMx_CR1 的地址）+ DBA 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。
7:5	保留	-	-	-
4:0	DBA	RW	5’h0	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 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， …….

19.3.28 TIM1 和 TIM8 连续模式的 DMA 地址（TIMx_DMAR）

Address offset:0x3E0

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[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**
DMAB[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DMAB	RW	0	DMA 连续传送寄存器（DMA register for burst accesses) 对 TIMx_DMAR 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： TIMx_CR1 地址 +（DBA + DMA 索引)x4 其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 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 TIMx_DCR 寄存器中定义的 DBL。

20. 通用定时器（TIM2 到 TIM5/TIM18）

20.1 简介

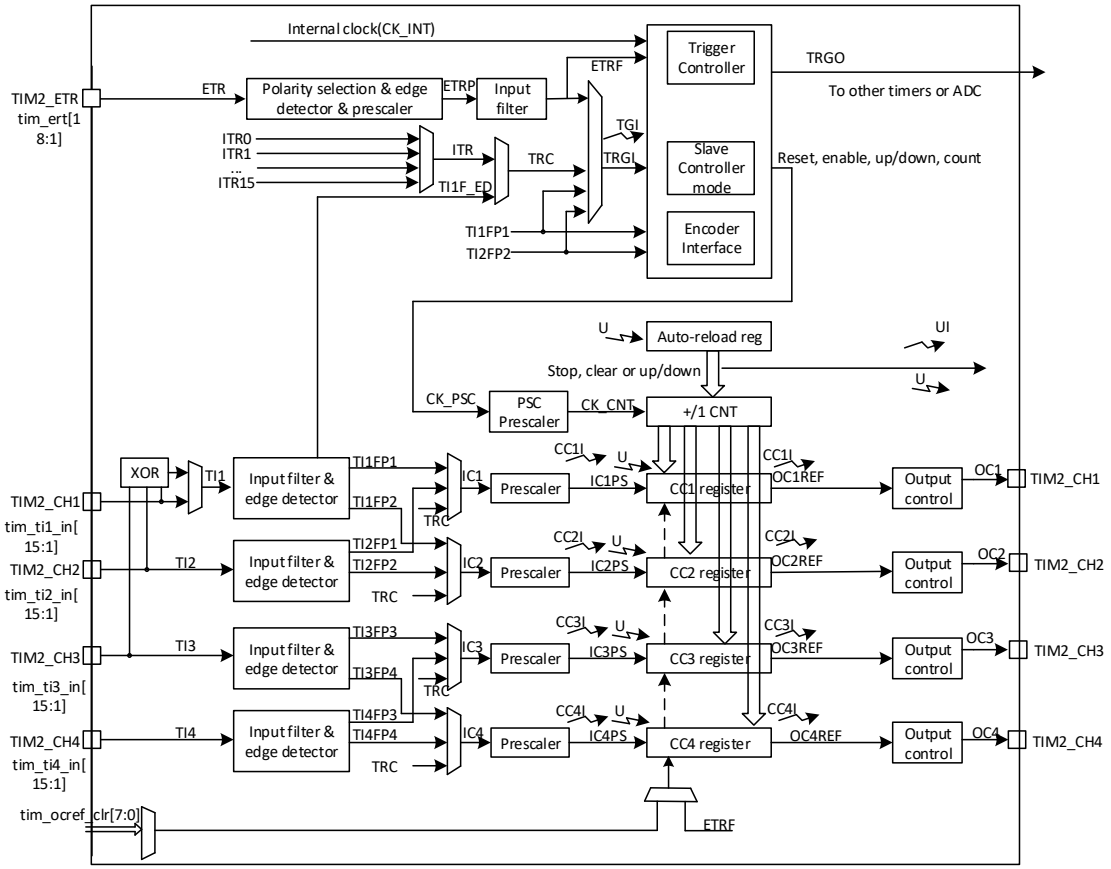
通用控制定时器(TIMx)由一个 32 位或 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。通用控制定时器(TIMx)和高级定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作。

20.1.1 TIM2/3/4/5/18 主要特征

主要功能包括：

- 32 位（TIM2）或 16 位（TIM3/4/5/18）向上、向下、向上/下的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，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(边缘或中央对齐模式)
  - 单脉冲模式输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 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
- 触发输入作为外部时钟或者按周期的电流管理

20.1.2 模块框图



20.2 TIM2/3/4/5/18 功能描述

20.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC)
- 自动装载寄存器（TIMx_ARR)

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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(上溢或者下溢)并当 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下边几张图给出了在预分频器运行时，更改计数器参数的例子。

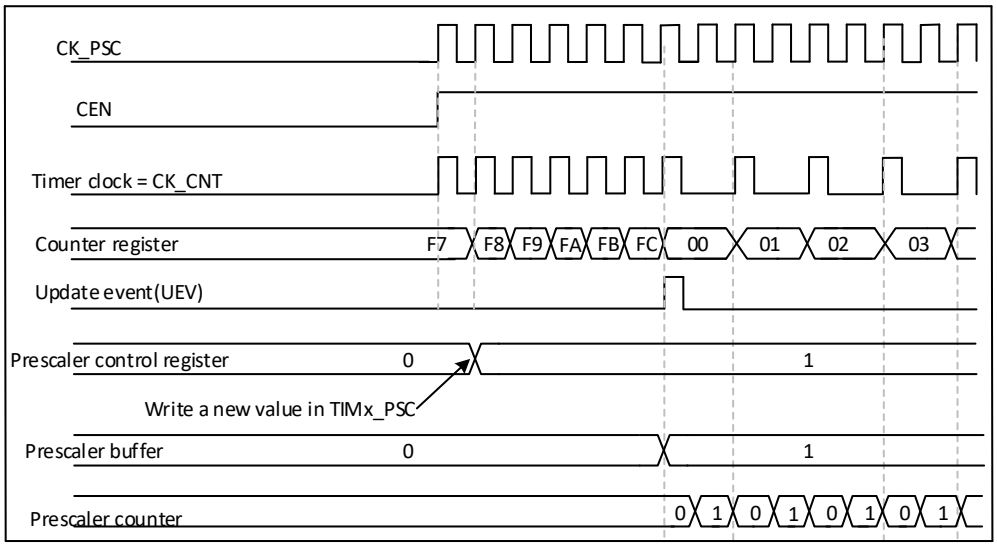


图 20-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

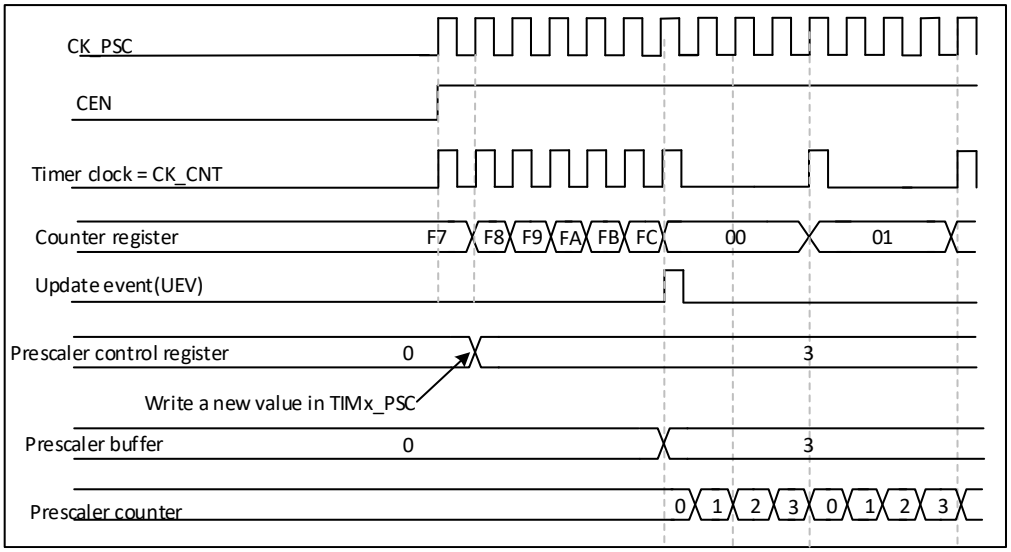


图 20-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

20.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断或 DMA 请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

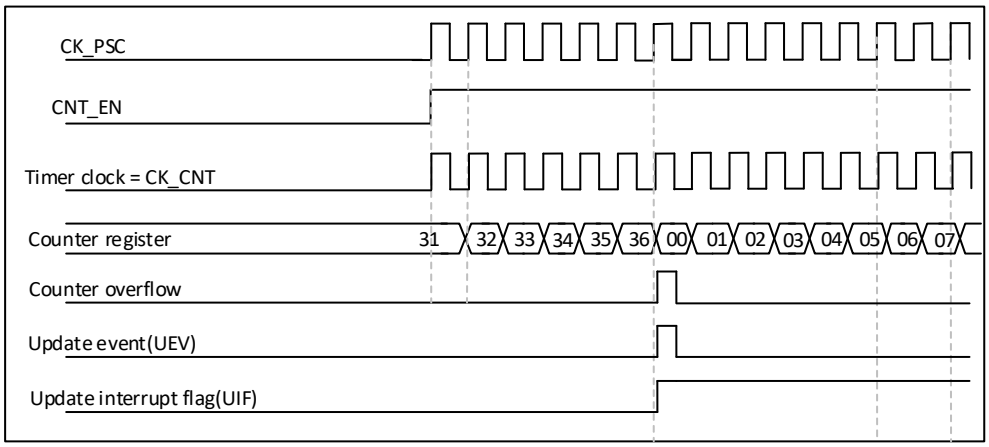


图 20-3 计数器时序图，内部时钟分频因子为 1

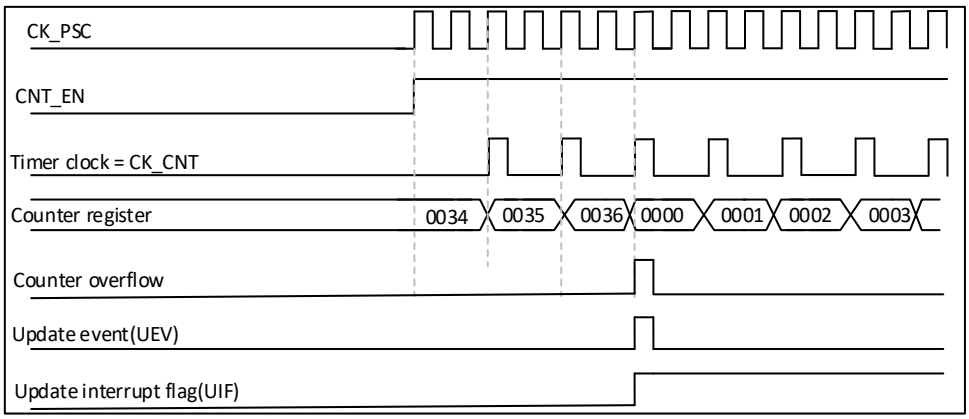


图 20-4 计数器时序图，内部时钟分频因子为 2

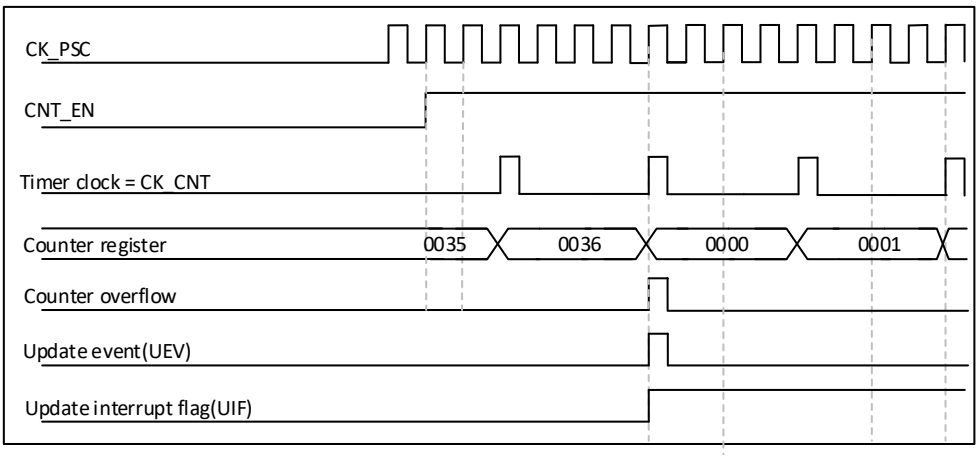


图 20-5 计数器时序图，内部时钟分频因子为 4

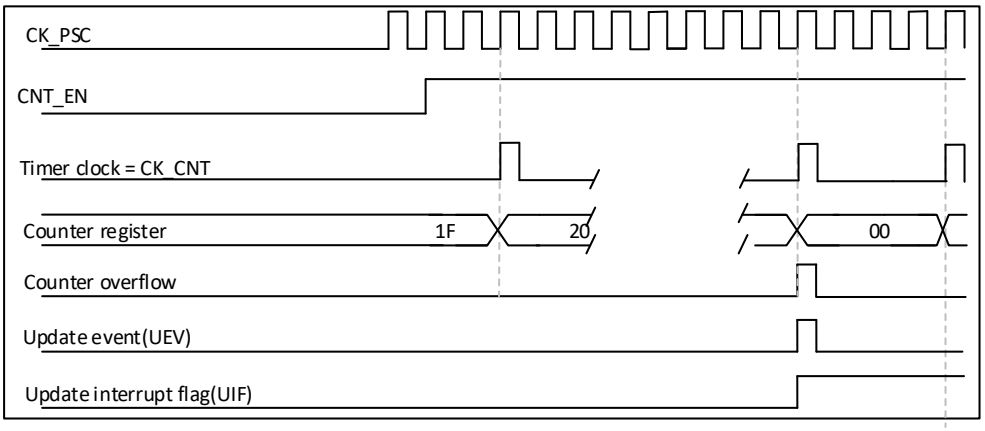


图 20-6 计数器时序图，内部时钟分频因子为 N

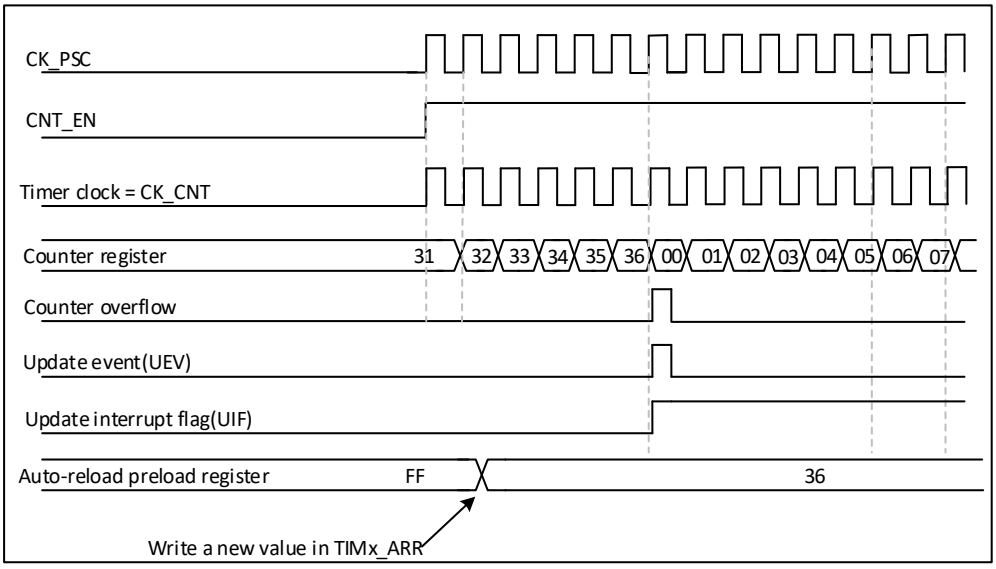


图 20-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

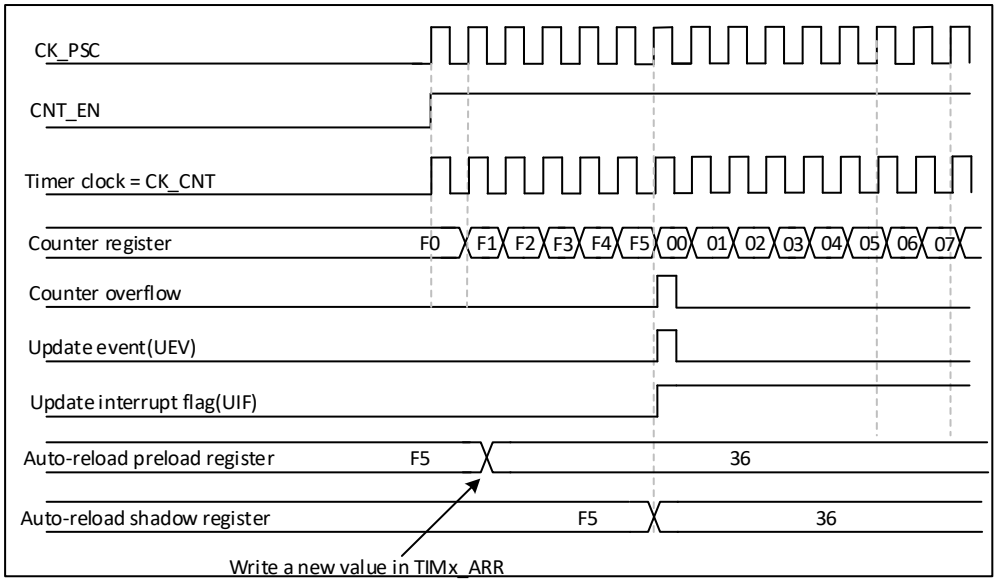


图 20-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

向下计数模式

在向下计数模式中，计数器从自动加载值(TIMx_ARR 的内容)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下溢事件。而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清'0'之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清'0'(但预分频系数不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源），通过设置 UG 位可以产生一个更新事件 UEV，但不置起 UIF 标志位(即不会产生中断或 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：自动加载值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 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

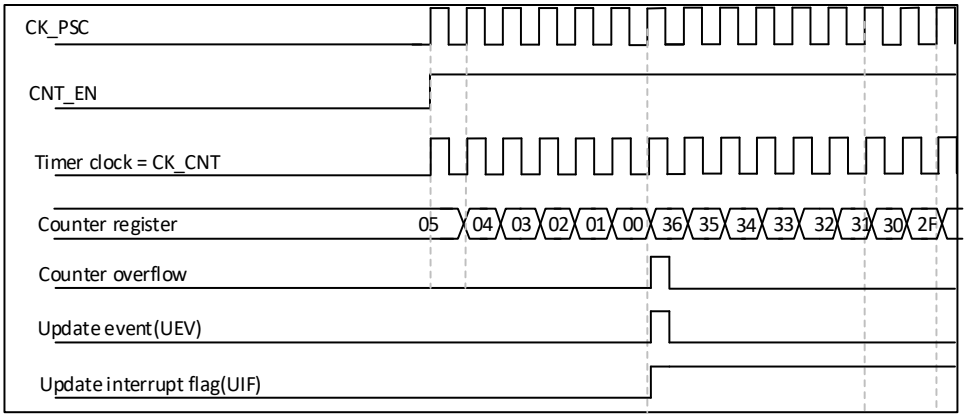


图 20-9 计数器时序图，内部时钟分频因子为 1

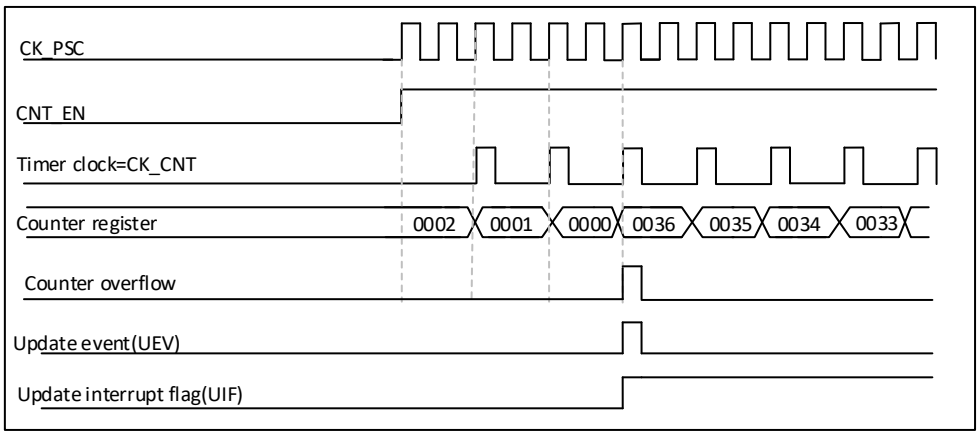


图 20-10 计数器时序图，内部时钟分频因子为 2

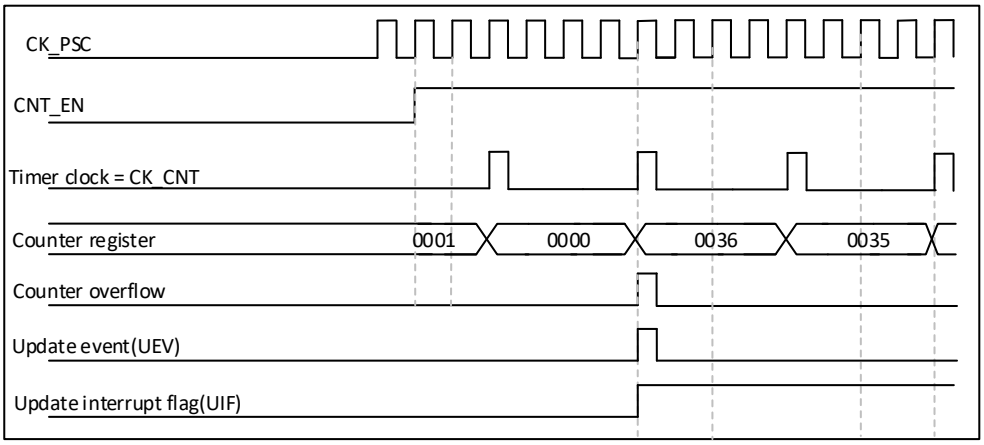


图 20-11 计数器时序图，内部时钟分频因子为 4

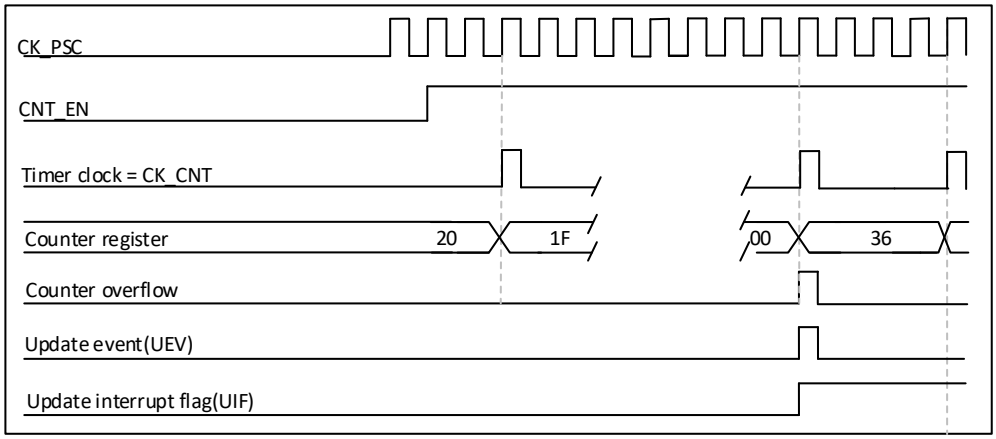


图 20-12 计数器时序图，内部时钟分频因子为 N

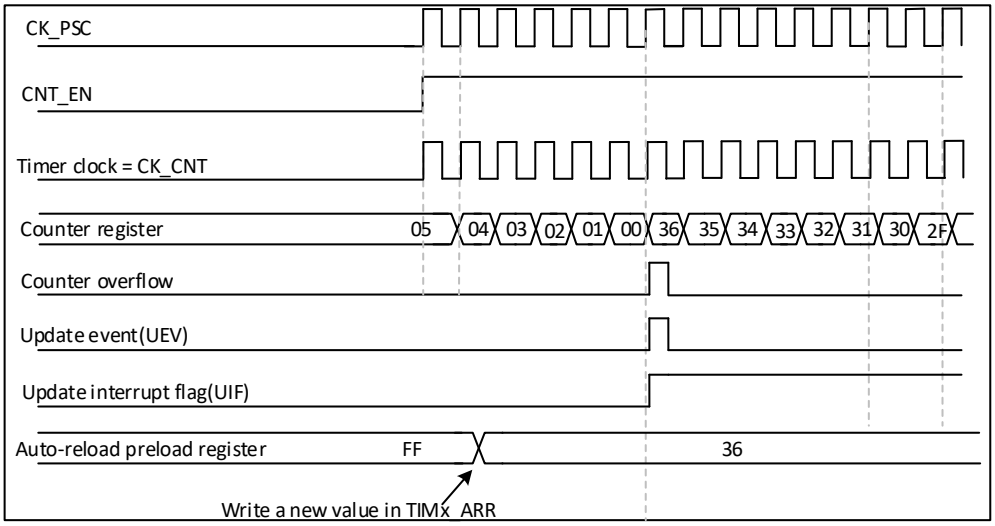


图 20-13 计数器时序图，当没有使用重复计数器时的更新事件

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(TIMx_ARR 寄存器)减 1，产生一个计数器上溢事件，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。

通过配置 TIMx_CR1 寄存器中的 CMS 位不为‘00’可以得到中央对齐模式。在通道配置为输出模式下的输出比较标志位在以下几种计数过程中会被置起：向下计数时（中央对齐模式 1，CMS=‘01’）、向上计数时（中央对齐模式 2，CMS=‘10’）、向上和向下计数时（中央对齐模式 3，CMS=‘11’）。

在此模式下，不能写入 TIMx_CR1 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 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器内部计数器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 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。虽然 UDIS 位被清为 0 之前不会产生更新事件，但是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 TIMx_CR1 寄存器中的 URS 位(选择更新请求源），通过设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 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时清除计数器，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 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置：

- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。注：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：

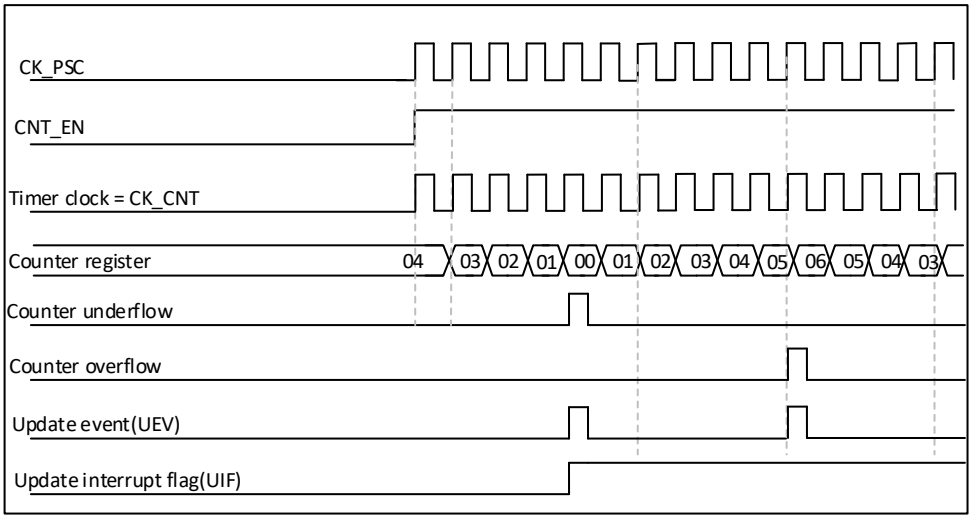


图 20-14 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

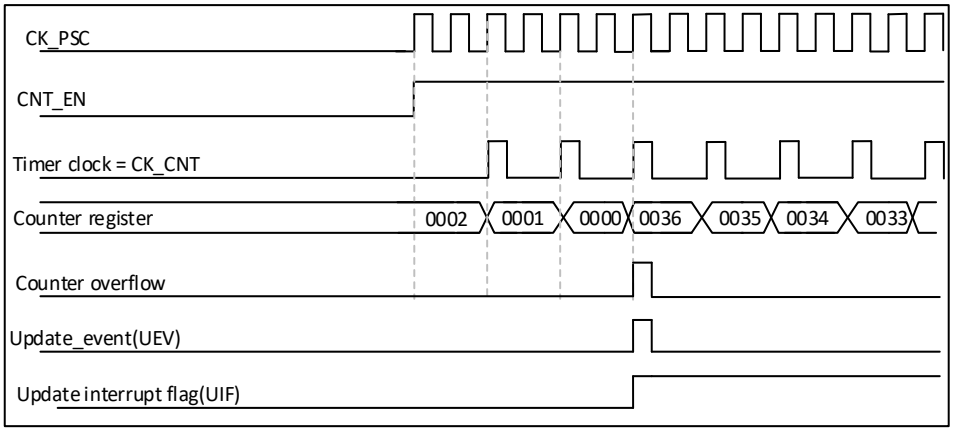


图 20-15 计数器时序图，内部时钟分频因子为 2

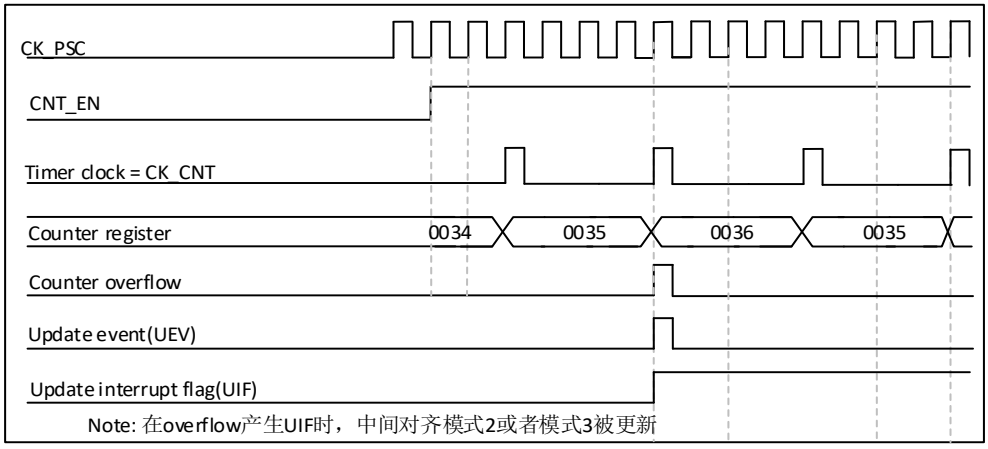


图 20-16 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

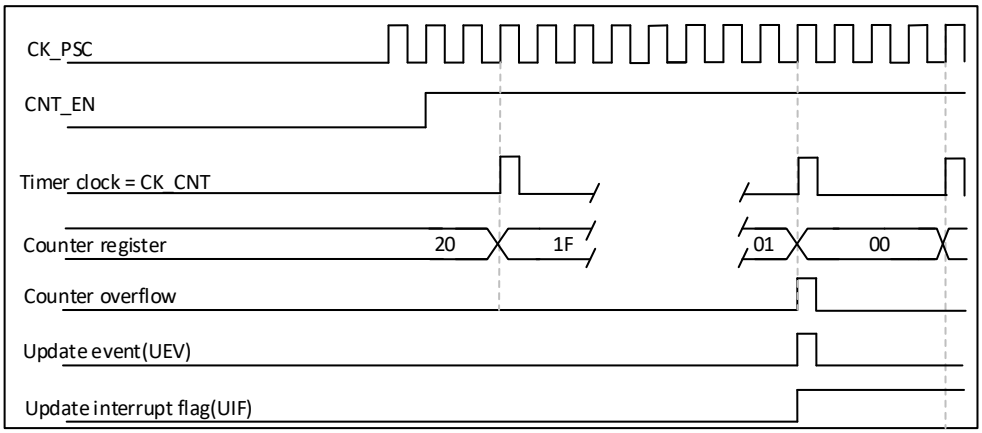


图 20-17 计数器时序图，内部时钟分频因子为 N

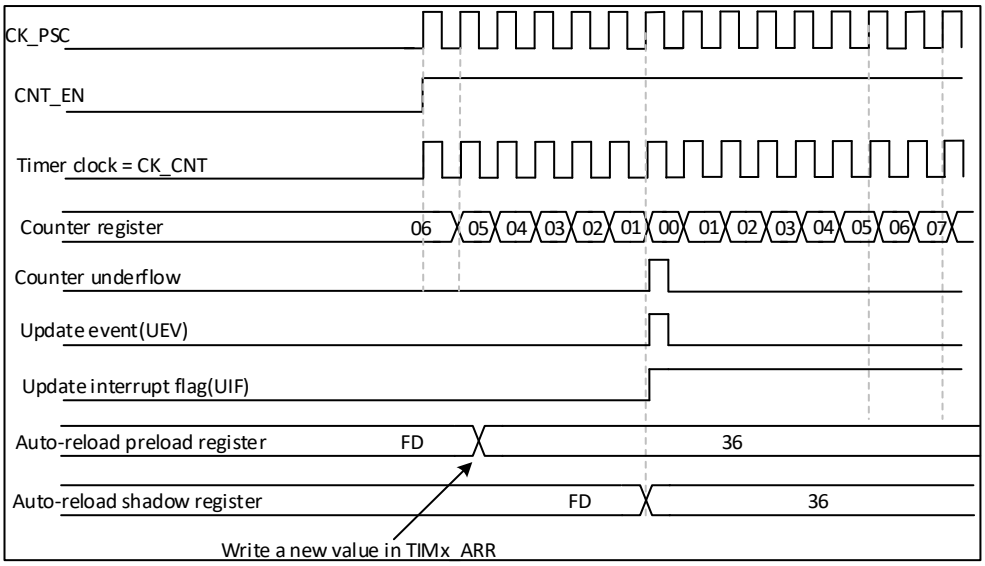


图 20-18 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

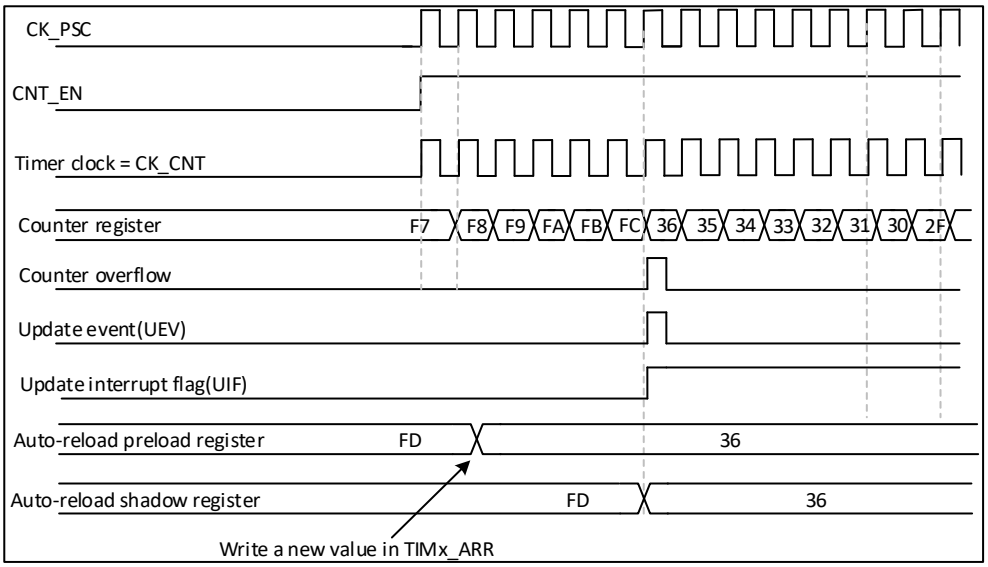


图 20-19 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

20.2.3 外部触发输入

定时器外部触发输入 tim_etr，可用于：

- 外部时钟
- 从模式的触发
- 在周期电流调节中作为 PWM 复位输入

20.2.4 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(TI1 和 TI2)
- 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。
- 编码器模式

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=0000)，则 CEN、DIR(TIMx_CR1 寄存器)和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 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。

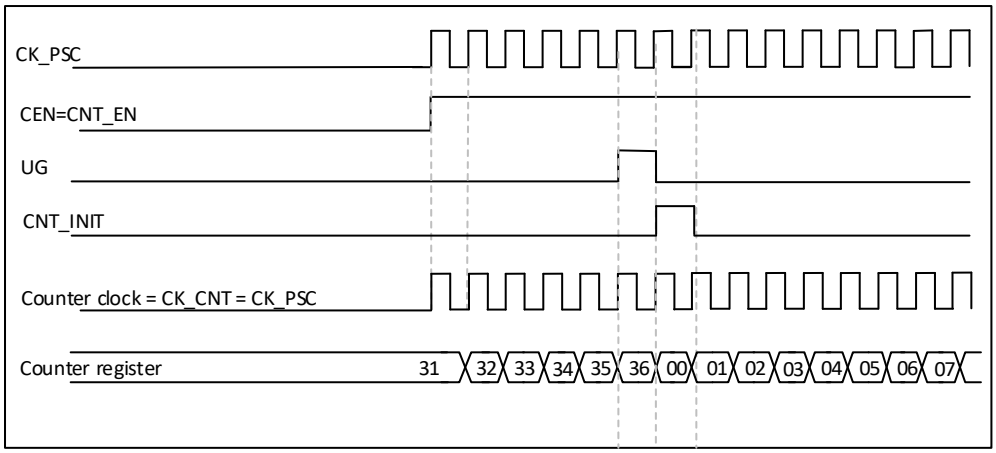


图 20-20 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=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。

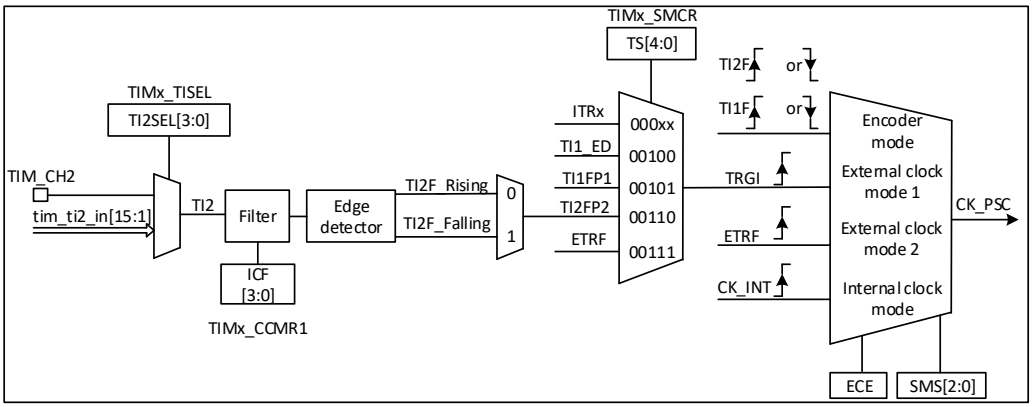


图 20-21 TI2 外部时钟连接例子

例如，要配置计数器在 TI2 输入端的上升沿向上计数，使用下列步骤：

1.配置 TIMx_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；

2.配置 TIMx_CCMR1 寄存器的 IC2F[3:0],选择输入滤波器带宽(如果不需要滤波器,保持 IC2F=0000)；

3.配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性；

4.配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；

5.配置 TIMx_SMCR 寄存器中的 TS=00110，选定 TI2 作为触发输入源；

6.设置 TIMx_CR1 寄存器的 CEN=1，启动计数器。

注：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 TIF 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 输入端的同步电路。

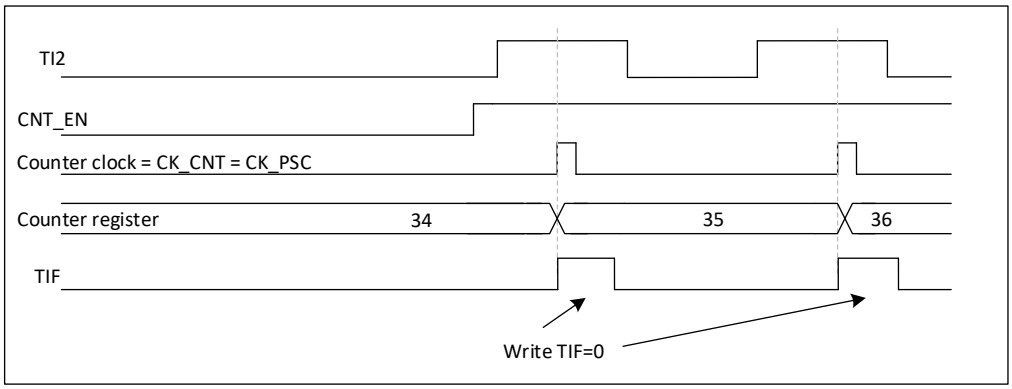


图 20-22 外部时钟模式 1 下的控制电路

外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=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：

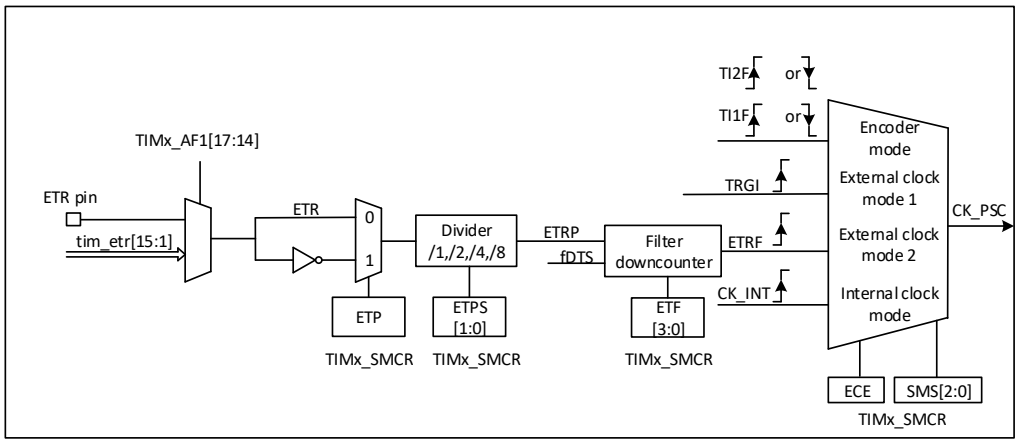


图 20-23 外部触发输入框图

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 TIMx_SMCR 寄存器中的 ETF[3:0]=0000；

2.设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01；

3.选择 ETR 输入端的上升沿，置 TIMx_SMCR 寄存器中的 ETP=0；

4.开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1；

5.启动计数器，写 TIMx_CR1 寄存器中的 CEN=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 ETRP 信号的同步电路。

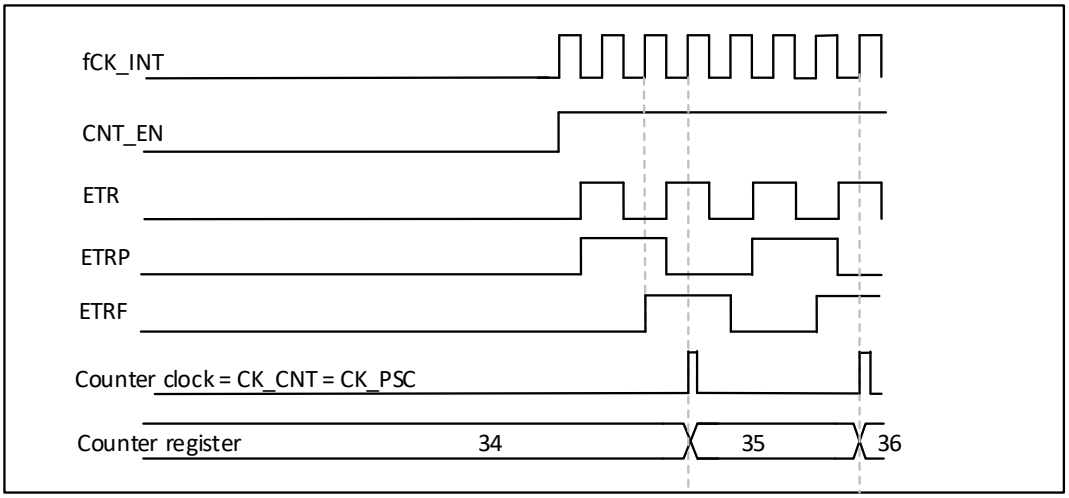


图 20-24 外部时钟模式 2 下的控制电路

20.2.5 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)。

下面几张图是捕获/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)。

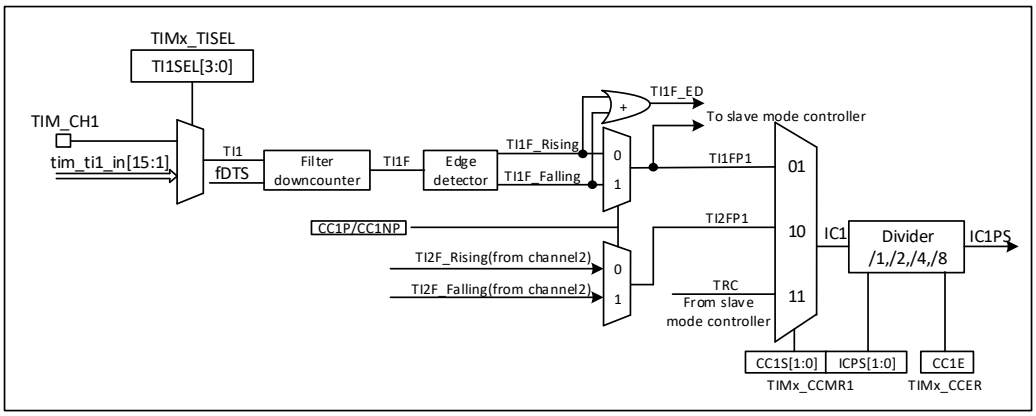


图 20-25 捕获/比较通道(如：通道 1 输入部分)

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。

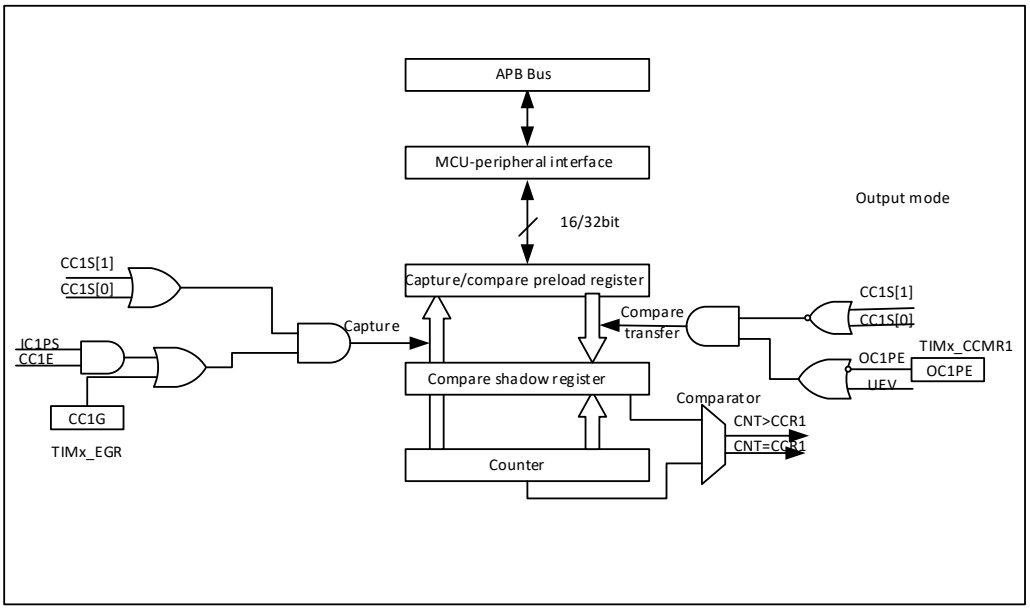


图 20-26 捕获/比较通道 1 的主电路

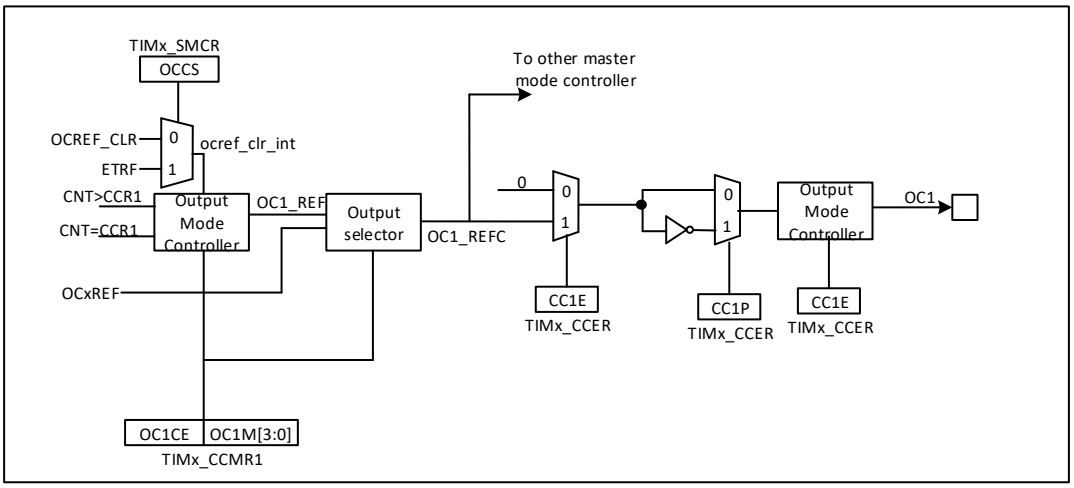


图 20-27 捕获/比较通道的输出部分(通道 1 至 4)

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。

20.2.6 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(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1，如果使能中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 ‘00’，通道就被配置为输入，并且 TIMx_CCR1 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 TIMx_CCMRx 寄存器中的 ICxF 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 fDTS 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 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 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 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，也可通过设置 TIMx_DIER 寄存器中的 CC1DE 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 TIMx_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

20.2.7 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，其余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，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。

- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

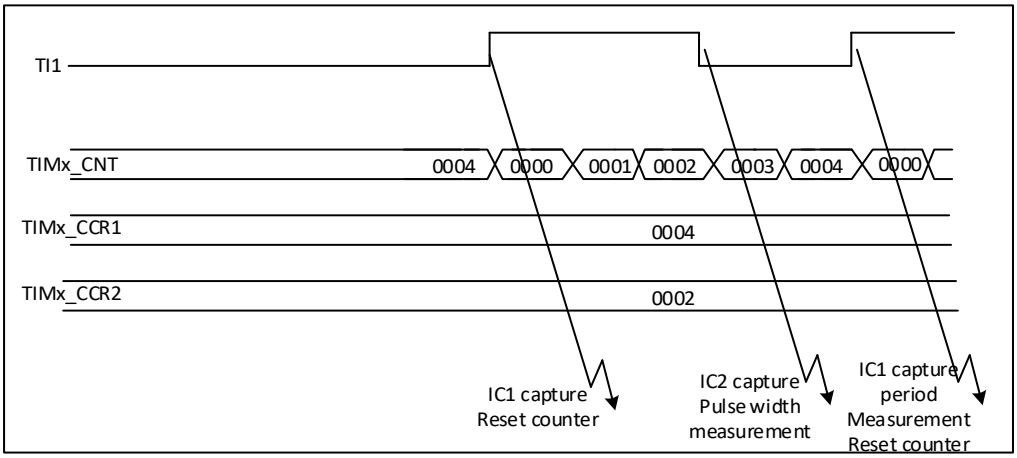


图 20-28 PWM 输入模式时序

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 TIMx_CH1/TIMx_CH2 信号。

20.2.8 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=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 TIMx_CCMRx 寄存器中相应的 OCxM=0101，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 CCxP 极性相反的信号。例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=0100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 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。

20.2.9 输出比较模式

此项功能是用来控制一个输出波形，或者表明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(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 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(OCxM=0000)、被设置成有效电平(OCxM=0001)、被设置成无效电平(OCxM=0010)或进行翻转(OCxM=0011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

可以通过配置 TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 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 TIMx_ARR 和 TIMx_CCRx 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 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=0011
— 置 OCxPE = 0 禁用预装载寄存器
— 置 CCxP = 0 选择极性为高电平有效
— 置 CCxE = 1 使能输出

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 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(OCxPE='0'，否则 TIMx_CCRx 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。

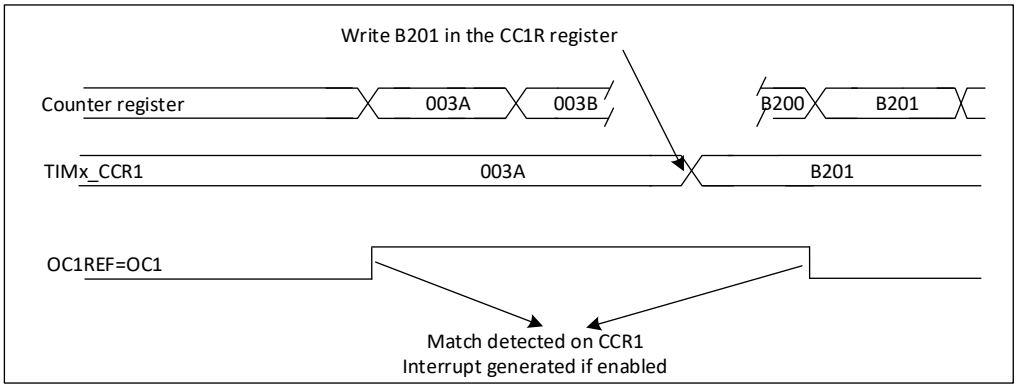


图 20-29 输出比较模式，翻转 OC1

20.2.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'0110'（PWM 模式 1)或'0111'(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 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 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，用户必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 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(TIMx_CCER)CCxE、CCxNE 控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

根据 TIMx_CR1 寄存器中 CMS 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 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

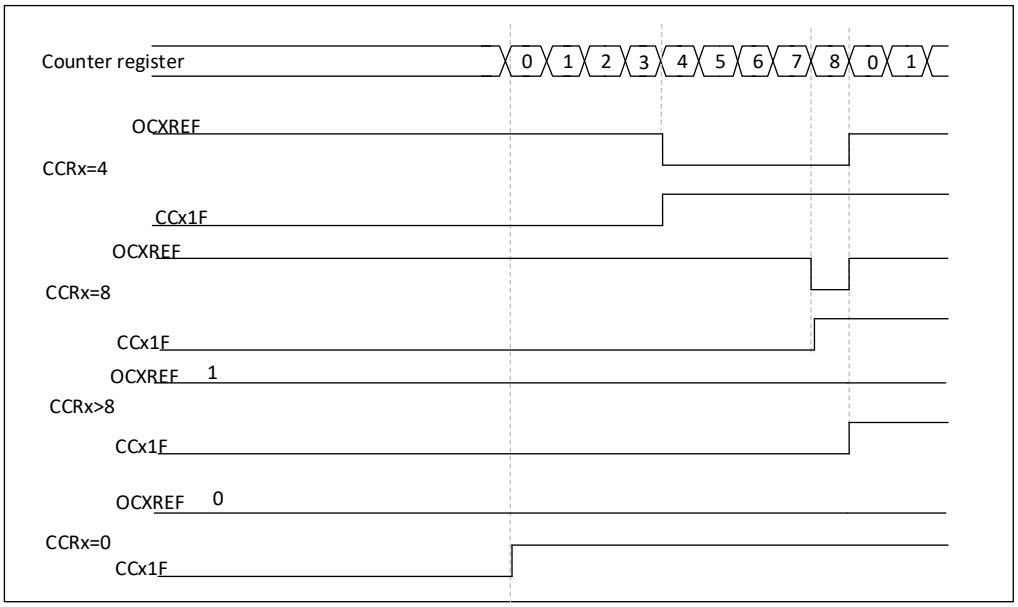


图 20-30 边沿对齐的 PWM 波形(ARR=8)

- 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 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 TIMx_CR1 寄存器中的 CMS 位不为‘00’时为中央对齐模式(CMS 位的所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 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。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器的 CMS=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。

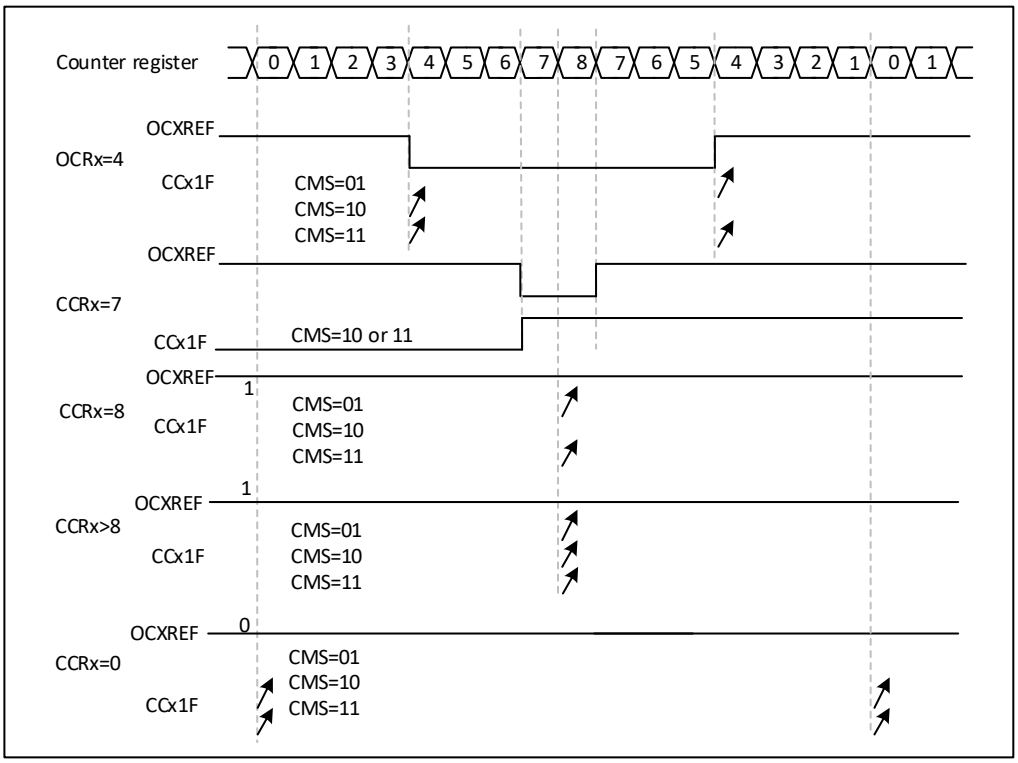


图 20-31 中央对齐的 PWM 波形(APR=8)

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 TIMx_CR1 寄存器中 DIR 位的当前值。此外，不要通过软件同时修改 DIR 和 CMS 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(TIMx_CNT>TIMx_ARR)，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 TIMx_ARR 的值写入计数器，方向被更新，但不会产生更新事件 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 TIMx_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

抖动模式

可以使能 TIMx_CR1 寄存器的 DITHEN 位来打开抖动模式提高 PWM 模式的有效分辨率。可以应用于 CCR（提高占空比分辨率）和 ARR（提高 PWM 频率的分辨率）

操作原理是使实际的 CCR（或 ARR）的值在 16 个连续的 PWM 周期内进行轻微的变化（增加或不增加一个定时器周期），如何变化可以通过预定义设置。计算平均占空比时就可以实现分辨率提高 16 倍。

下图为 4 个连续 PWM 周期抖动原理

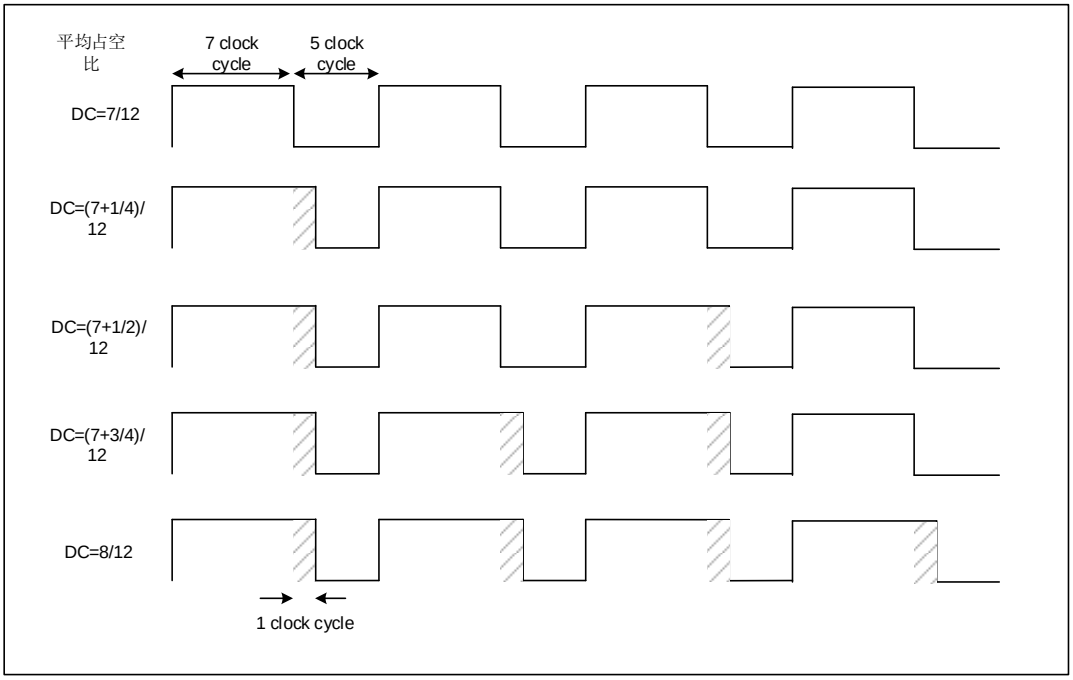


图 20-32 抖动原理

当抖动模式使能后，寄存器的值会自动如下变化：

- 最低的 4 bits 是用于提高分辨率的编码(小数部分)
- 高位被左移到 19:4 作为基值进行编码（整数部分）

注：当抖动模式打开或关闭时，ARR 和 CCR 的值会自动更新（例如，DITHEN=0 时 ARR=0x05，那么 DITHEN=1 时 ARR=0x50），复位 DITHEN 位时必须遵循以下步骤

1. CEN 和 ARPE 位必须复位
2. ARR[3:0]位必须复位
3. DITHEN 位必须复位
4. CCIF 标志位必须清除
5. CEN 位可以置位（可设置 ARPE=1）

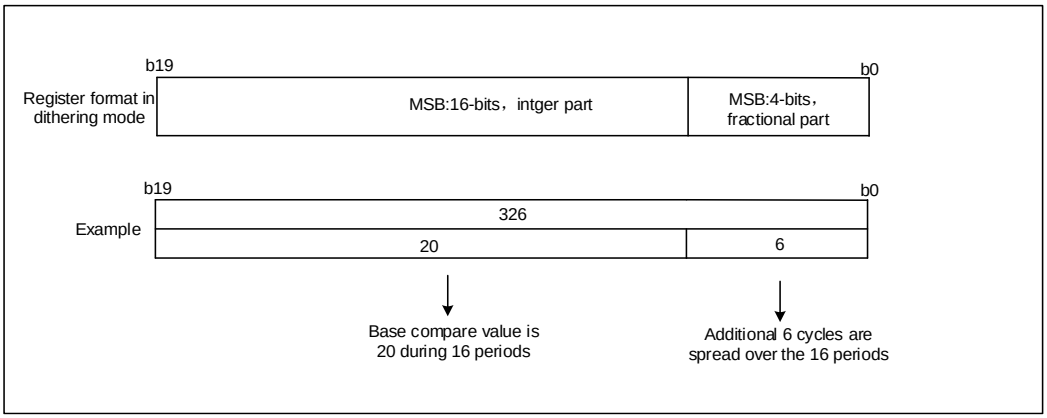


图 20-33 抖动模式中数据映射与寄存器编码

最小的频率计算公式如下：

$$\text{根据分辨率} = \frac{\text{定时器频率}}{\text{PWM 频率}} \qquad \text{推出：最小 PWM 频率} = \frac{\text{定时器频率}}{\text{最大分辨率}}$$

$$\text{无抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65536}$$

$$\text{有抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65535+\frac{15}{16}}$$

注：抖动模式里 TIMx_ARR 和 TIMxCCRy 的最大值被限定到 0xFFFEF（对应于整数部分的 65534 和抖动部分的 15），超出 0xFFFEF 会存在溢出情况，即 arr=0XFFFF+1=0。

下图显示的是抖动模式下，无论 PWM 频率多少都可以提高 PWM 的分辨率

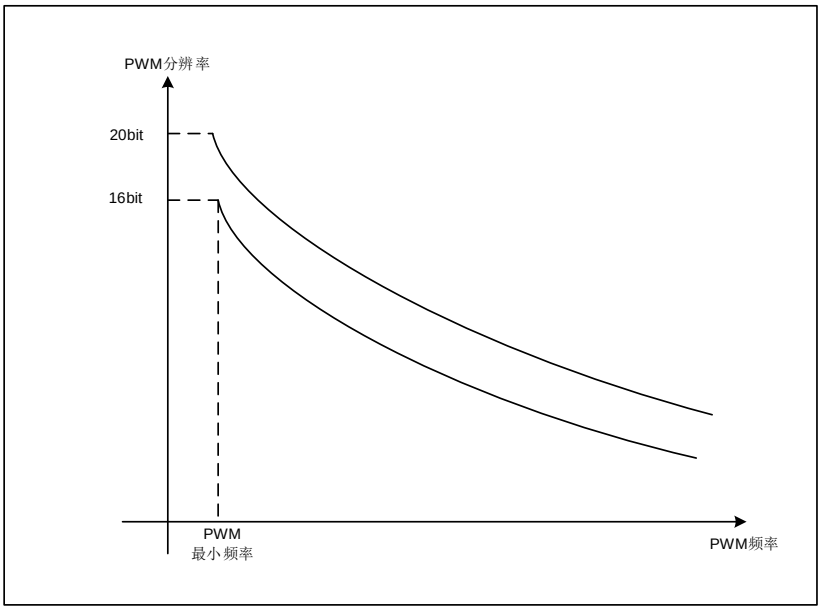


图 20-34 PWM 分辨率 vs 频率

连续 16 个周期内占空比和周期变化情况如下图：

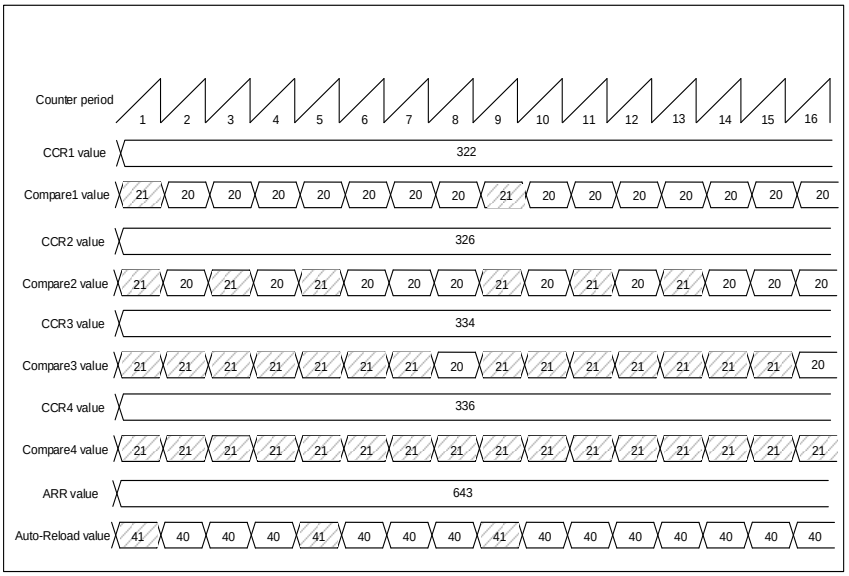


图 20-35 PWM 抖动模式

自动重新加载和比较值增量按照如下表描述的特定模式分布，抖动序列的增量分布尽可能地均匀，最大限度减少整体波纹。

表 20-1 CCR 和 ARR 寄存器抖动变化

LSB value	PWM 周期															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000																
0001	+1															
0010	+1								+1							
0011	+1				+1				+1							
0100	+1				+1				+1				+1			
0101	+1		+1		+1				+1				+1			
0110	+1		+1		+1				+1		+1		+1			
0111	+1		+1		+1		+1		+1		+1		+1			
1000	+1		+1		+1		+1		+1		+1		+1		+1	
1001	+1	+1	+1		+1		+1		+1		+1		+1		+1	
1010	+1	+1	+1		+1		+1		+1	+1	+1		+1		+1	
1011	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1		+1	
1100	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1	+1	+1	
1101	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1		+1	+1	+1	
1110	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1	+1	+1	+1	+1	
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	

抖动模式也可以应用在中间对齐 PWM 模式中(CMS!=00)，下图是考虑了向上向下计数时，抖动模式应用于连续 8 个周期的情况。

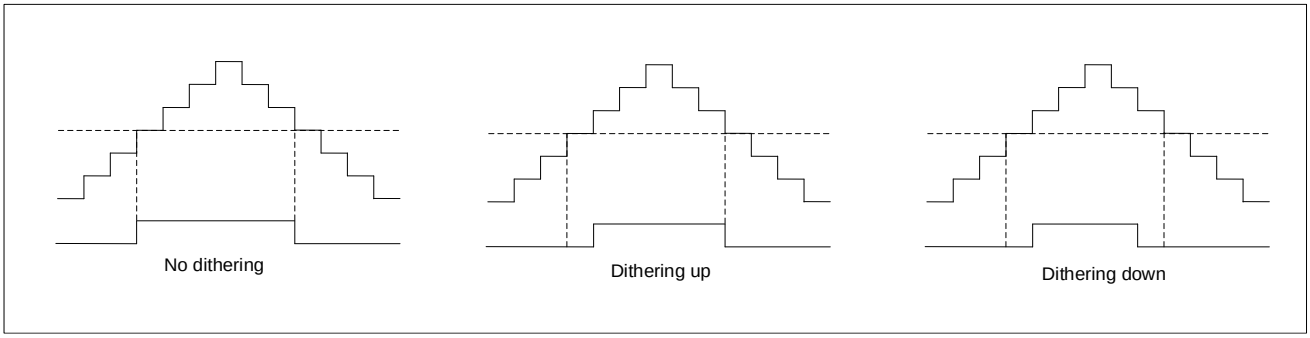


图 20-36 中间对齐 PWM 模式中抖动对占空比的影响

表 20-2 中间对齐 PWM 模式中 CCR 寄存器抖动变化

LSB value	PWM 周期															
	1		2		3		4		5		6		7		8	
	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN	UP	DN
0000																
0001	+1															
0010	+1								+1							
0011	+1				+1				+1							
0100	+1				+1				+1				+1			
0101	+1		+1		+1				+1				+1			
0110	+1		+1		+1				+1		+1		+1			
0111	+1		+1		+1		+1		+1		+1		+1			
1000	+1		+1		+1		+1		+1		+1		+1		+1	
1001	+1	+1	+1		+1		+1		+1		+1		+1		+1	
1010	+1	+1	+1		+1		+1		+1	+1	+1		+1		+1	
1011	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1		+1	
1100	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1	+1	+1	
1101	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1		+1	+1	+1	
1110	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1	+1	+1	+1	+1	
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	

20.2.11非对称 PWM 模式

非对称模式允许两个中央对齐的 PWM 信号产生可编程的相移。当频率由 TIMx_ARR 决定、占空比和相移由一对 TIMx_CCRx 寄存器决定时。不对称模式允许使用可编程的相移生成两路中心对齐的 PWM

信号。一个 TIMx_CCRx 寄存器在向上计数阶段控制，另一个在向下计数阶段控制，以便每半个 PWM 周期都可调整 PWM

- OC1REFC（or OC2REFC）由 TIMx_CCR1 和 TIMx_CCR2 控制
- OC3REFC（or OC4REFC）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过向 OCxM 写 1110(不对称 PWM 模式 1）和 OCxM 写 1111(不对称 PWM 模式 2)，可以在双通道（每对 CCR 寄存器控制一个 OCx 输出）上独立选择不对称的 PWM 模式

注意：出于兼容的原因，OCxM[3:0]字段分为两部分，最高位和低 3 位不连续。

当给定的通道被用作不对称通道时，其相邻通道(通道 1 与通道 2 相邻)也可使用。例如，如果 OC1REFC 由通道 1（不对称 PWM 模式 1）产生，则通道 2 也可以输出 OC2REF，或者通过不对称 PWM 模式 1 输出 OC2REFC

下图表示使用不对称 PWM 模式生成信号的示例（通道 1 到 4 被配置为不对称 PWM 模式 2），与死区时间生成器一起，控制全桥 DC-DC 相移逆变器。

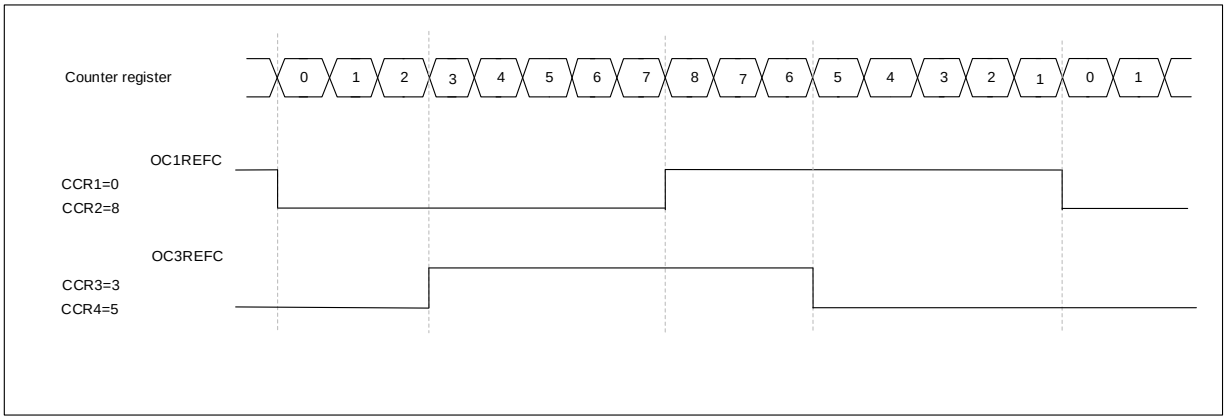


图 20-37 2 路 50%占空比相移 PWM

20.2.12 组合 PWM 模式

组合 PWM 模式允许通过对各脉冲间进行可编程的延迟和相移来产生边沿或中央对齐 PWM 信号。频率由 ARR 决定，占空比和延迟由两个 CCR 决定。产生的信号 OCxREFC 由两个参考 PWM 间进行与/或逻辑产生

- OC1REFC（or OC2REFC）由 TIMx_CCR1 和 TIMx_CCR2 控制
- OC3REFC（or OC4REFC）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过向 OCxM 写 1100(组合 PWM 模式 1，逻辑或输出）和 OCxM 写 1101(组合 PWM 模式 2，逻辑与输出)，可以在双通道（每对 CCR 寄存器控制一个 OCx 输出）上独立选择不对称的 PWM 模式

当给定的通道被用作组合 PWM 通道，其互补通道必须被配置为相反的 PWM 模式(例如，一路为组合 PWM 模式 1、一路为组合 PWM 模式 2)

下图为组合模式生成信号示例，包括

- 通道 1 为组合 PWM 模式 2
- 通道 2 为 PWM 模式 1
- 通道 3 为组合 PWM 模式 2
- 通道 4 为 PWM 模式 1

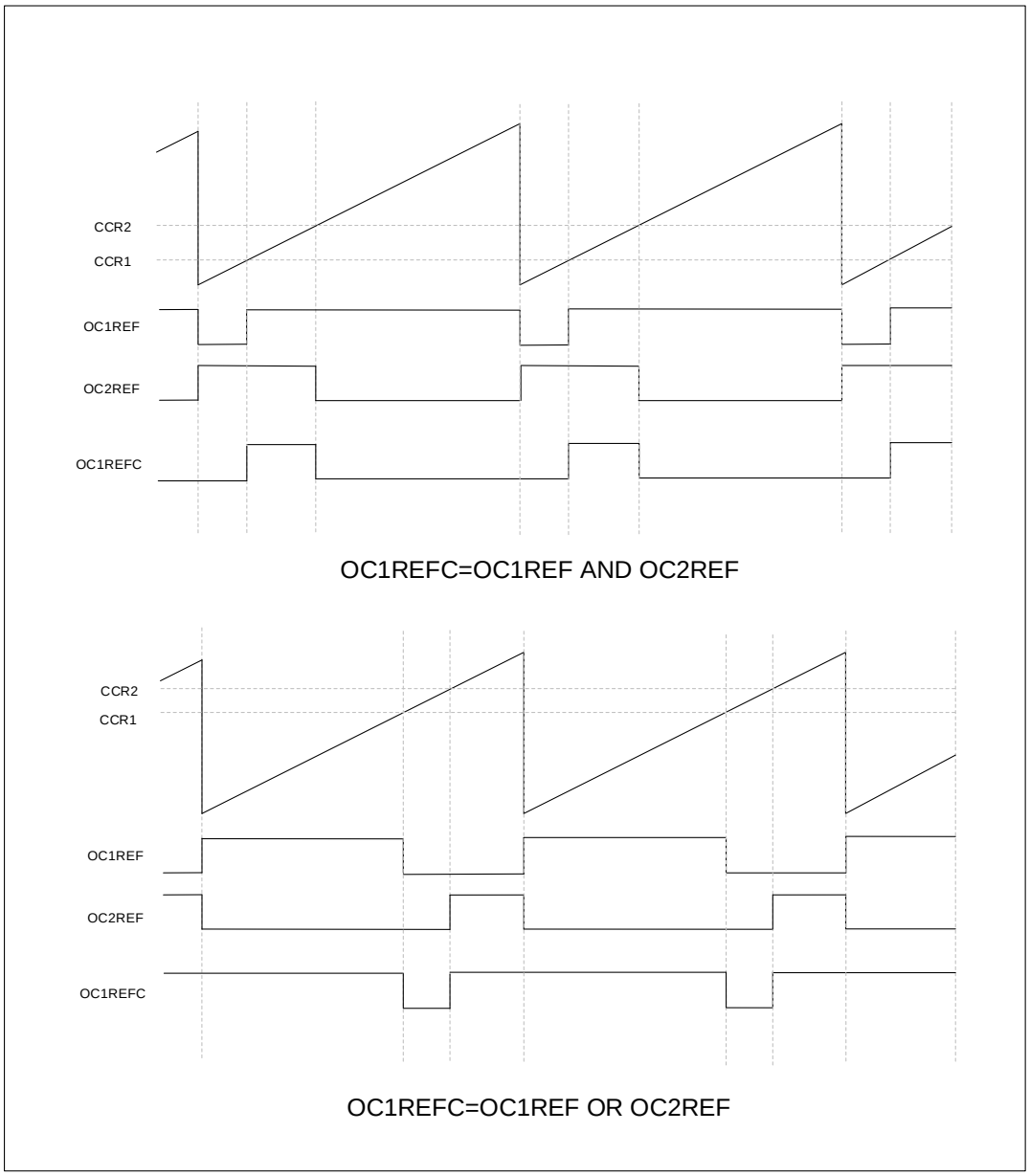


图 20-38 通道 1 和通道 3 的组合 PWM 模式

20.2.13在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为‘1’，能够用 OCREF_CLR_INT 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计数溢出所产生的更新事件 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。

而 OCREF_CLR_INT 可以通过配置 TIMx_SMCR 寄存器中的 OCCS 位，在 OCREF_CLR 和 ETRF(ETR 滤波后)之间选择。

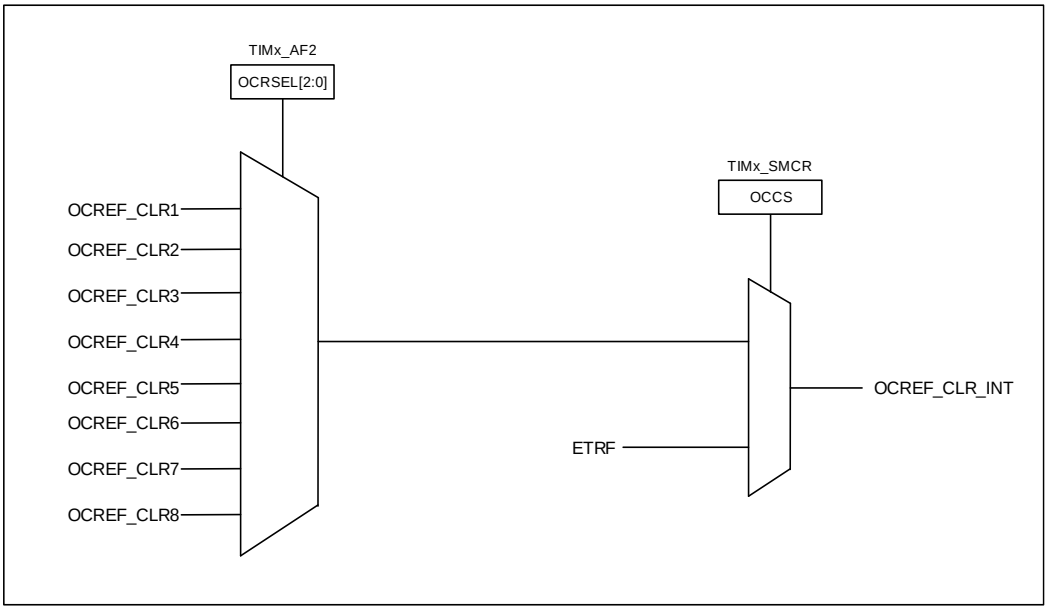


图 20-39 OCREF_CLR 输入选择

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：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)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 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM1 模式。

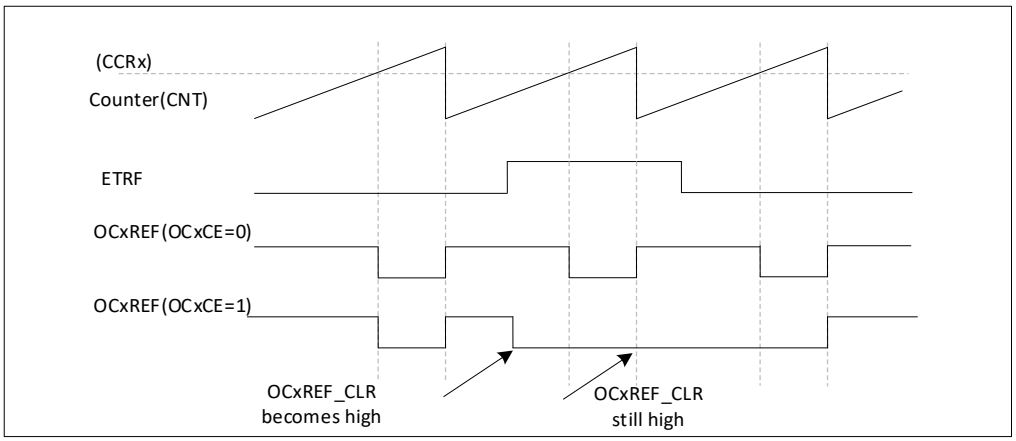


图 20-40 清除 TIMx 的 OCxREF

20.2.14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 TIMx_CR1 寄存器中的 OPM 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 < CCRx ≤ ARR（特别地, 0 < CCRx)，

- 向下计数方式：计数器 CNT > CCRx。

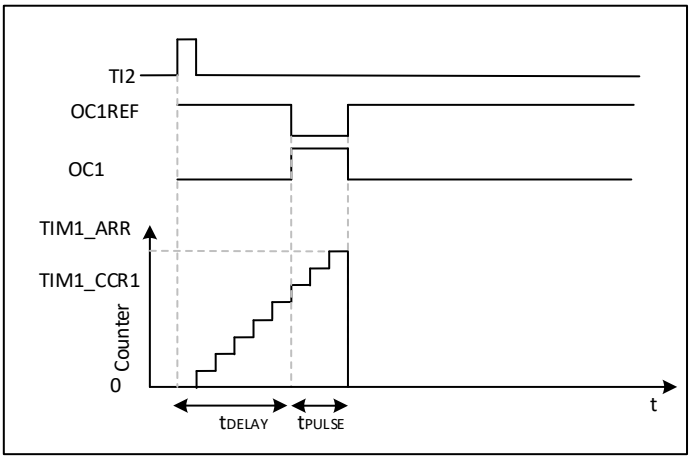


图 20-41 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 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)

- tDELAY 由 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 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 0 到 1 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 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，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=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 OPM=0 时，重复模式被选中。

20.2.14.1 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 CEN 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 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 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。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

20.2.15可重触发的单脉冲模式

该模式允许计数器响应激励启动并产生长度可编程的脉冲，与上节所述的不可再触发单脉冲模式有以下区别：

- 触发一发生，脉冲就开始（无可编程延迟）。
- 如果在前一个触发所产生的脉冲完成之前发生新的触发，则延长脉冲。

要使用可重触发的单脉冲模式，计时器必须处于从模式，TIMx_SMCR 寄存器中的位 SMS[3:0]=“1000”（组合模式-复位+触发），OCxM[3:0]位设置为“1000”或“1001”（可重新触发 OPM 模式 1 或 2）。
如果此时计时器配置为递增计数模式，则相应的 CCRx 必须设置为 0（ARR 寄存器设置脉冲长度）。
如果计时器配置为递减计数模式，CCRx 必须大于或等于 ARR。
注：出于兼容性原因，OCxM[3:0]和 SMS[3:0]位字段被分成两部分，最高有效位与 3 个最低有效位位置不连续。

该模式不得与中央对齐计数模式一起使用。TIMx_CR1 中必须为 CMS[1:0]=00。

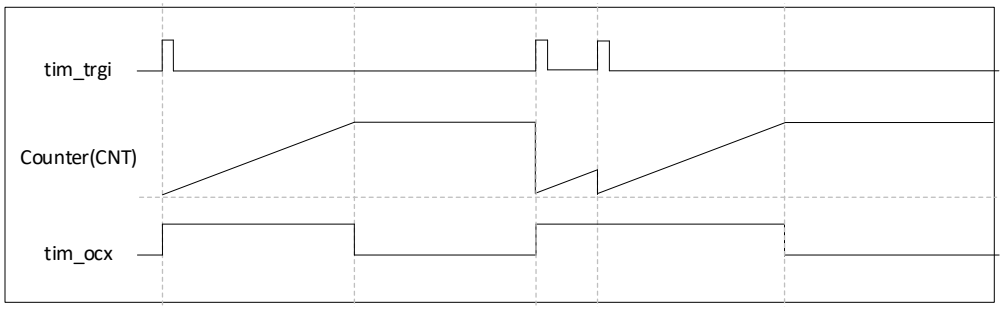


图 20-42 可重触发的单脉冲模式的例子

20.2.16比较模式生成脉冲

脉冲可以通过比较匹配事件产生。当计数器值等于给定的比较值时，一个可编程宽度的脉冲信号就产生了，用于调试和同步的用途。
该模式可用于任何从模式，包括编码器模式。在边沿和中央对齐计数模式中它仅适用于 3/4 通道。脉冲生成器是唯一的并且由这两个通道共享，如下图所示：

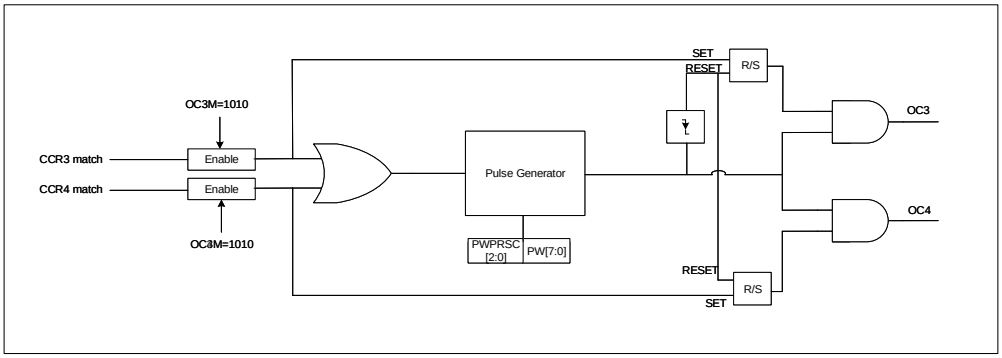


图 20-43 脉冲生成器电路

下图表示在边沿计数和编码器模式中脉冲是如何生成的：

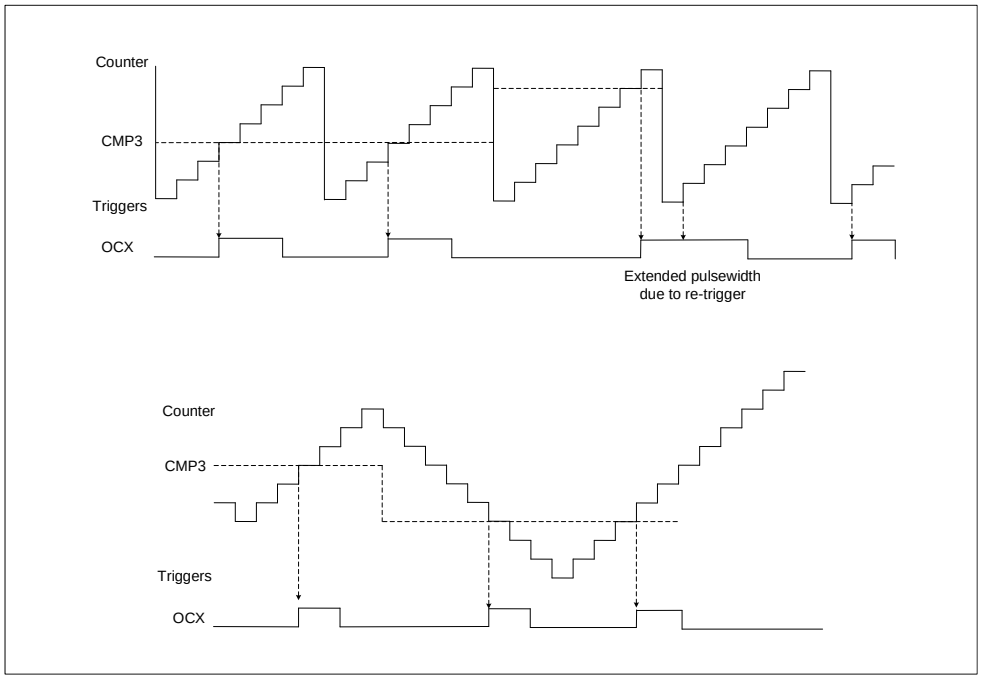


图 20-44 边沿对齐和中央对齐模式中通过比较事件生成脉冲

输出比较模式是通过 TIMx_CCMR2 中的 OC3M[3:0] 和 OC4M[3:0]来选择的

脉冲宽度是通过 PW[7:0]寄存器编程的，通过 PWPRSC[2:0]使用时钟预分频产生的时钟

t_{PW}=PW[7:0] x t_{PWG}

t_{PWG}= $2^{(PWPRSC[2:0])}$ x CK_INT

分辨率和最大值取决于预分频器的值

脉冲再触发：当脉冲正常输出时来了新的触发，会导致脉冲被延长

注：如果两个通道同时被使能，只要一个通道上的触发不与另一个通道产生的脉冲重叠，输出的脉冲就是独立的。如果两路触发是重叠的，触发 1 产生脉冲将被延长(会被再触发)，而最后到来的触发产生脉冲宽度是正确的。

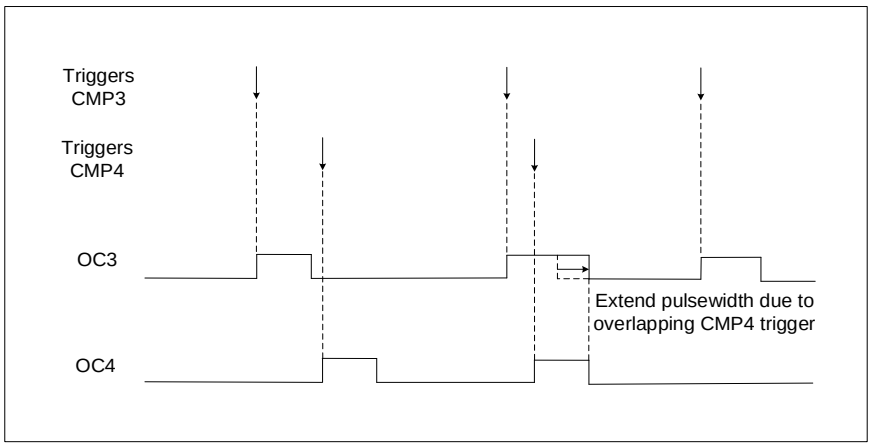


图 20-45 边沿对齐和中央对齐模式中通过比较事件生成脉冲

20.2.17编码器接口模式

20.2.17.1 正交编码器

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 TIMx_SMCR 寄存器中的 SMS=0001；如果只在 TI2 边沿计数，则置 SMS=0010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=0011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 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。参看下表，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=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 TIMx_CR1 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 TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；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 20-3 计数方向与编码器信号的关系（CC1P=CC2P=0）

有效沿	SMS[3:0]	反向信号电平 (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 信号		TI2FP2 信号	
			Rising	Falling	Rising	Falling
仅在 TI1 单沿计数 x1 模式	1110	高	减	加	-	-
		低	-	-	-	-
仅在 TI2 单沿计数 x1 模式	1111	高	-	-	加	减
		低	-	-	-	-
仅在 TI1 双沿计数 x2 模式	0001	高	减	加	-	-
		低	加	减	-	-
仅在 TI2 双沿计数 x2 模式	0010	高	-	-	加	减
		低	-	-	减	加
在 TI1 和 TI2 双沿计数 x4 模式	0011	高	减	加	加	减
		低	加	减	减	加

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：

- CC1S= ‘01’ (TIMx_CCMR1 寄存器，TI1FP1 映射到 IC1)
- CC2S= ‘01’ (TIMx_CCMR2 寄存器，TI2FP2 映射到 IC2)
- CC1P= ‘0’ (TIMx_CCER 寄存器，TI1FP1 不反相，TI1FP1=TI1)
- CC2P= ‘0’ (TIMx_CCER 寄存器，TI2FP2 不反相，TI2FP2=TI2)
- SMS= ‘011’ (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN= ‘1’ (TIMx_CR1 寄存器，计数器使能)

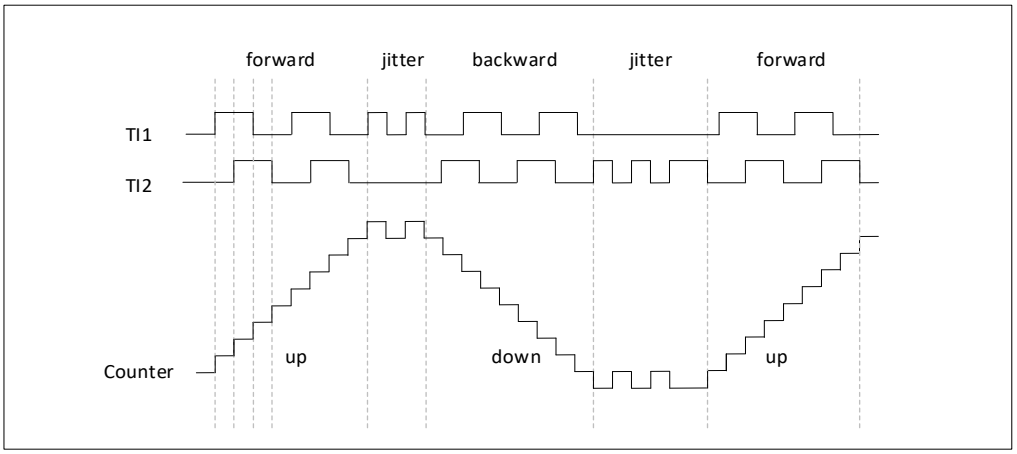


图 20-46 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P=‘1’，其他配置与上例相同)

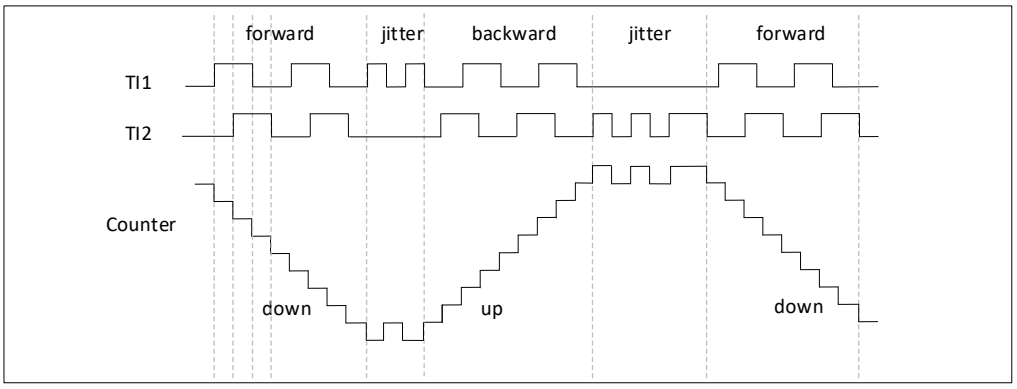


图 20-47 IC1FP1 反相的编码器接口模式实例

下图表示的是不同模式下当转向翻转时的计数值情况：

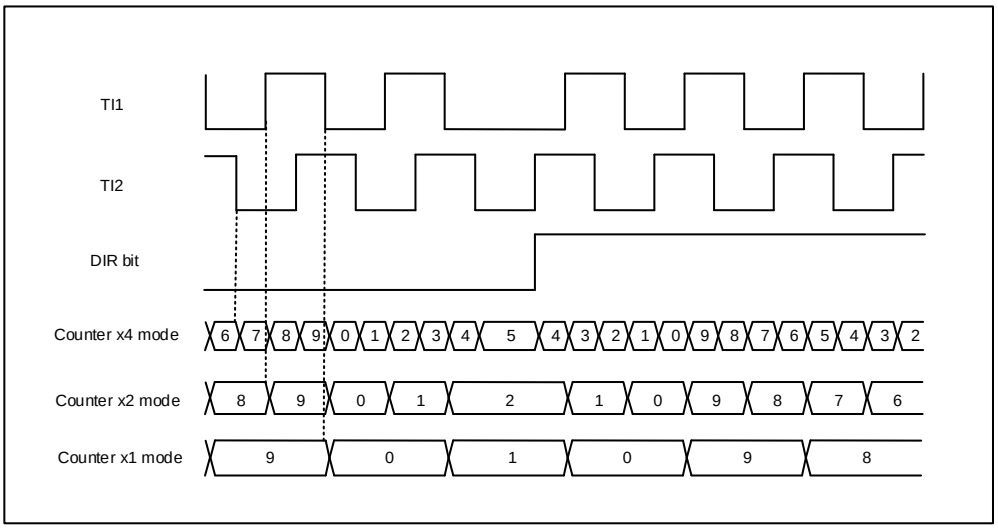


图 20-48 正交编码器计数模式

当定时器配置成编码器接口模式时，可以提供传感器当前位置的信息。通过将第二个定时器配置在捕获模式，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。

TIMxCR1 寄存器的 UIFREMAP 位强制将更新中断标志（UIF）连续复制到定时器计数器的最高位（TIMxCNT[31]）。这样允许计数值和一个由 UIFCPY 标志产生的状态可读出。这样可以通过避免引起竞态条件来简化角速度的计算。例如，通过共享后台任务（计数器读出）和中断之间（中断更新）的处理

UIF 和 UIFCPY 标志断言之间没有延迟

在 32 位计数器设计中，当 UIFREMAP 位被置位，当进行读取时计数器的第 31 位会被 UIFCPY 标志覆盖（计数器的最高有效位只在写模式下可以访问）

20.2.17.2 时钟加方向编码器模式

除正交编码器外，定时器还支持其他不同类型的编码器模式

时钟加方向模式如图所示，时钟由 TI2 提供，方向由 TI1 输入，该模式通过配置 TIMx_SMCR 寄存器的 SMS[3:0]来使能，如下所示：

- 1010：x2 模式，计数器在时钟的上升沿和下降沿更新
- 1011：x1 模式，计数器在单沿更新，CC2P=0 时上升沿更新，CC2P=1 时下降沿更新

TI1 输入的方向信号的极性由 CC1P 位决定，CC1P=0：正极性（当 TI1 为高时向上计数，当 TI1 为低时向下计数）；CC1P=1：负极性（当 TI1 为高时向下计数，当 TI1 为低时向上计数）

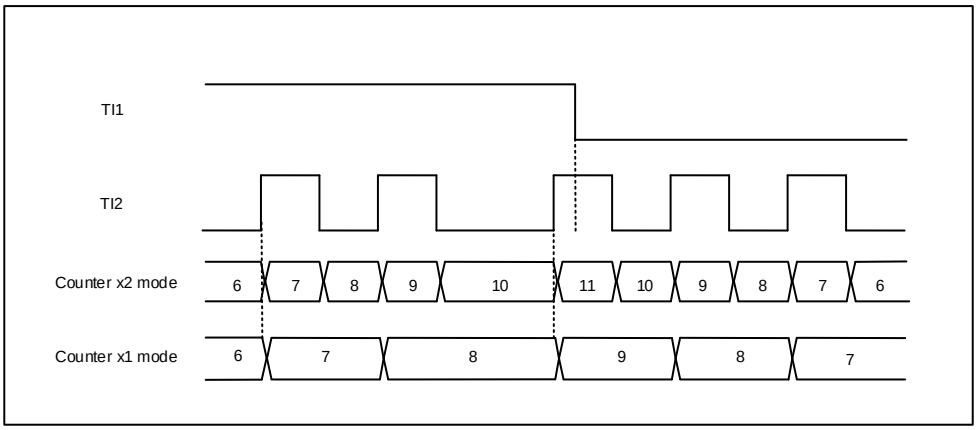


图 20-49 方向加时钟编码器模式

20.2.17.3 定向时钟编码器模式

在定向时钟模式中（如下图所示），时钟由两路信号提供，根据方向一次提供一条，从而具有一条向上计数时钟信号（TI2）和一条向下计数时钟信号(TI1)。该模式通过配置 TIMx_SMCR 寄存器的 SMS[3:0]来使能，如下所示：

- 1100：x2 模式，无论哪条时钟线，计数器都是在时钟的上升沿和下降沿都更新。可以通过 CC1P 和 CC2P 位来配置时钟初始状态。CCxP=0 代表高电平初始状态，CCxP=1 代表低电平初始状态
- 1101：x1 模式，计数器在单沿更新，取决于 CCx1P 和 CCx2P 的值。CCxP=0 表示下降沿计数并且初始状态为高电平。CCxP=1 表示上升沿计数并且初始状态为低电平。

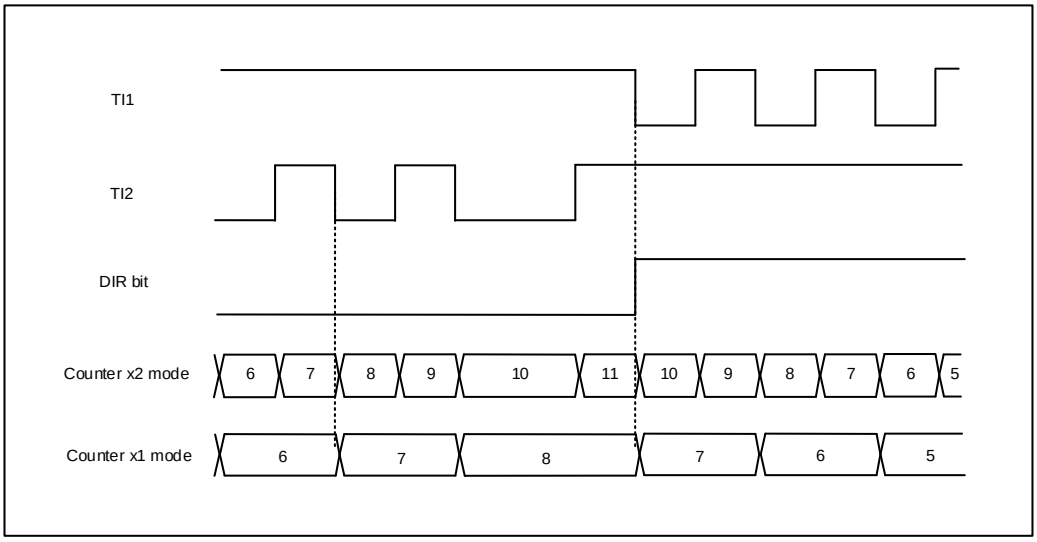


图 20-50 定向时钟编码器模式（CC1P=CC2P=0）

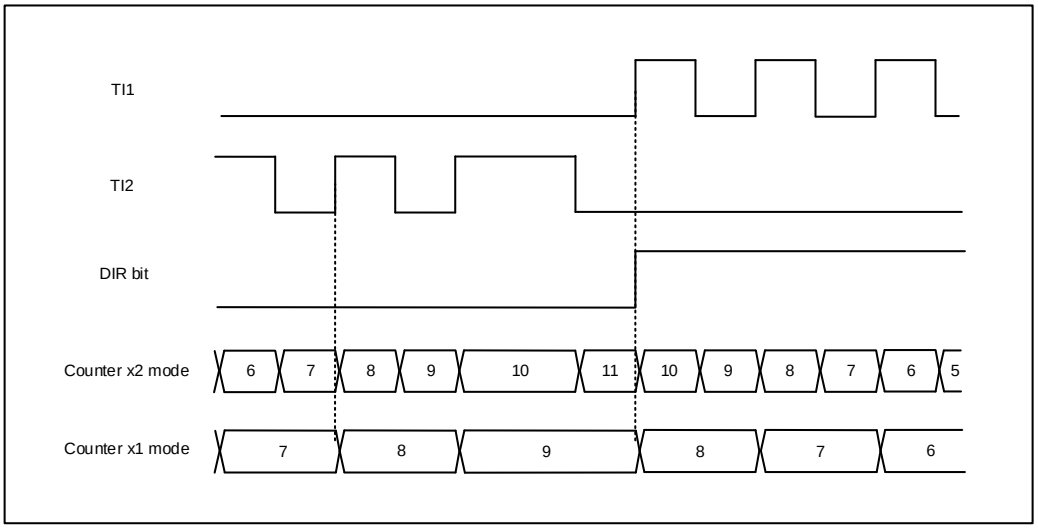


图 20-51 定向时钟编码器模式（CC1P=CC2P=1）

下表详细说明了对于任何输入转换，定向时钟模式是如何操作的：

表 20-4 计数方向与编码器信号极性的关系

有效沿	SMS[3:0]	反向信号电平 (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 信号		TI2FP2 信号	
			Rising	Falling	Rising	Falling
x2 模式 CCxP=0	1100	高	减	减	加	加
		低	-	-	-	-
x2 模式 CCxP=1	1100	高	-	-	-	-
		低	减	减	加	加
x1 模式 CCxP=0	1101	高	-	减	-	加
		低	-	-	-	-
x1 模式 CCxP=1	1101	高	-	-	-	-
		低	减	-	加	-

20.2.17.4 索引输入

计数器可以被编码器输出的索引信号复位，用于指示绝对参考位置。索引信号必须连接到 tim_etr 输入端。可通过数字滤波器来滤波。

TIMx_ECR 寄存器的 IE 位作为索引功能的使能，IE 位只能在编码器模式中使能，即当 SMS[3:0]配置成如下值时：0001、0010、0011、1010、1011、1100、1101、1110、1111

如下图所示，商用编码器有几种用于索引脉冲调节的选项

- A 与 B 的门控：脉冲宽度为一个通道的 1/4，与 A 和 B 的边沿对齐
- A 的门控（B 的门控）：脉冲宽度为一个通道的 1/2，与 A 的边沿对齐
- 非门控：脉冲宽度高达一个通道的周期，不与通道边沿对齐

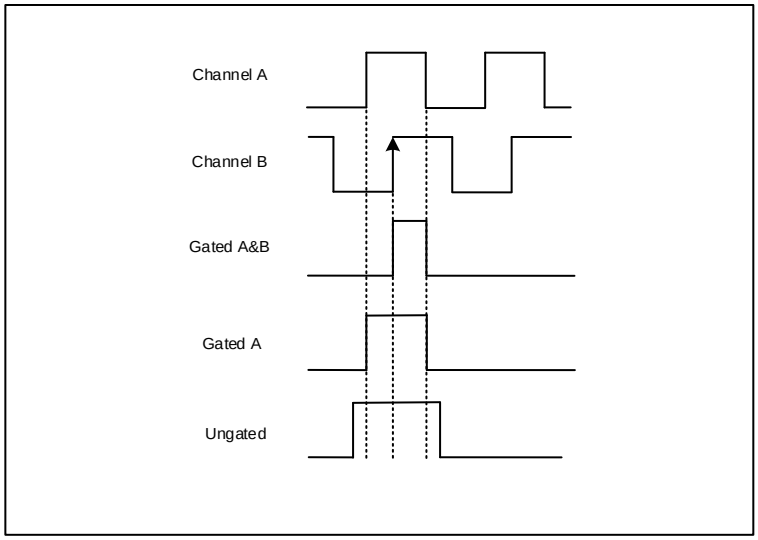


图 20-52 索引门控选择

下图表示的是，任意门控模式下，电路可容忍索引信号的抖动。在非门控模式中，索引信号脉宽需要严格小于 2 个编码器周期。如果大于或者等于 2 个周期，计数器会被复位多次。

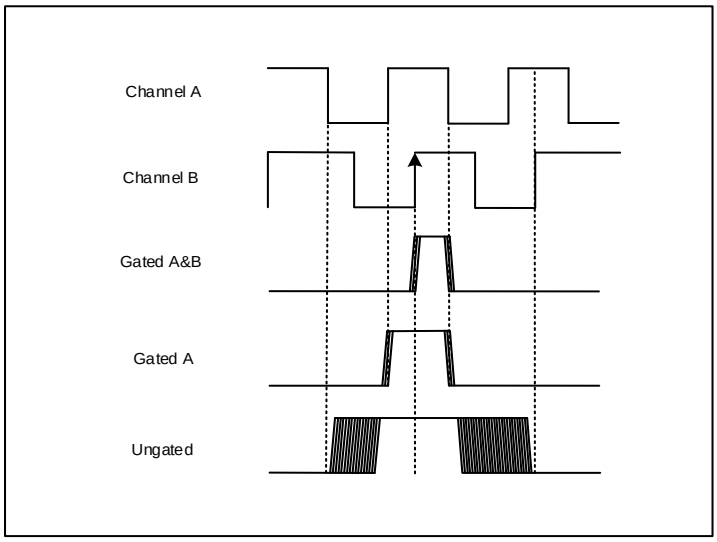


图 20-53 抖动的索引信号

定时器完全支持 3 种门控模式，不需要特别的编程，只需要定义编码器为何状态时（通道 A 和通道 B 状态的组合）索引信号需要被同步，通过配置 TIMx_ECR 寄存器的 IPOS[1:0]。

索引信号的检测事件根据计数方向的不同而不同，以确保旋转方向翻转期间的对称操作。

- 计数器在向上计数过程中被复位(DIR=0)
- 向下计数时，计数器的值设置为 TIMx_ARR

这使得无论计数方向如何，索引都可以在相同的机械角度位置上生成，下图展示了哪些位置索引生成，举个简单的例子（一个编码器提供 4 个沿每个标准机械旋转）

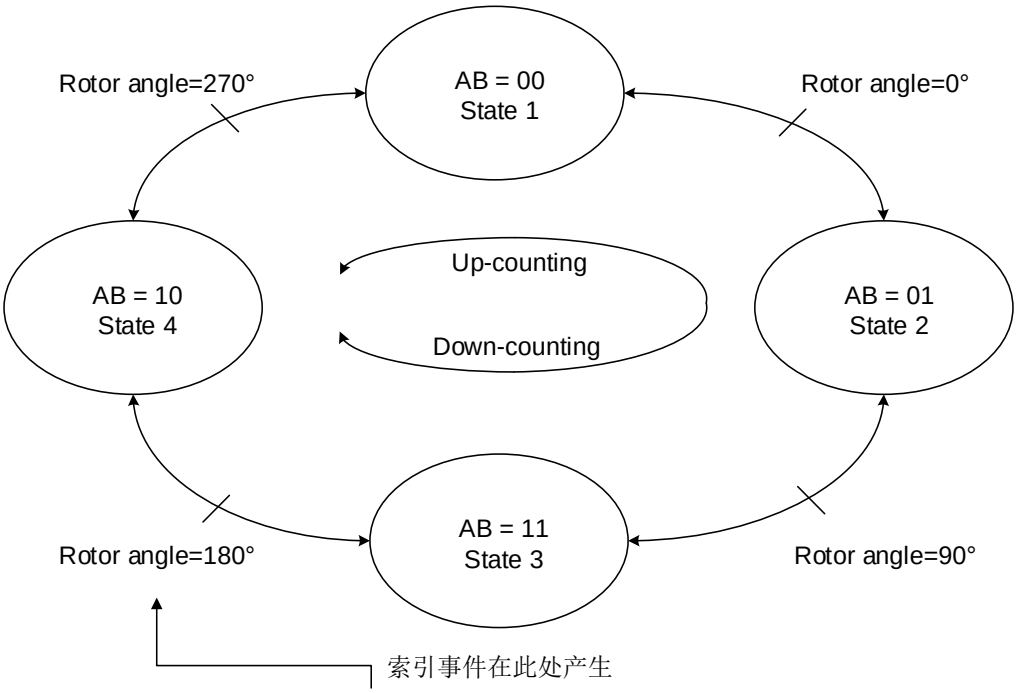


图 20-54 IPOS[1:0]=11 时索引生成

下图显示了 IPOS[1:0]=11 时的波形和相应值，它表明计数器值被强制的时刻是根据计数方向自动调整的

- 计数器清零当编码器进入 11 状态（通道 A=1，通道 B=1），向上计数时（DIR=0）
- 计数器值被设为 TIMx_ARR 当离开 11 状态，向下计数时（DIR=1）

索引检测事件可产生中断

箭头表示哪个转换过程中索引事件中断产生

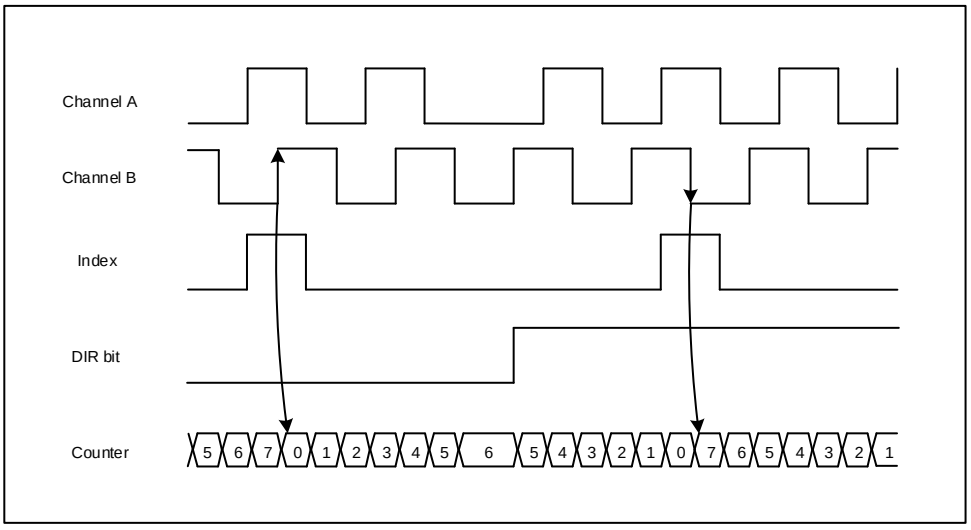


图 20-55 IPOS[1:0]=11 时索引与通道 A 门控模式时的计数值

下图显示了非门控模式时的波形和相应值，箭头表示哪个转换过程中索引事件中断产生

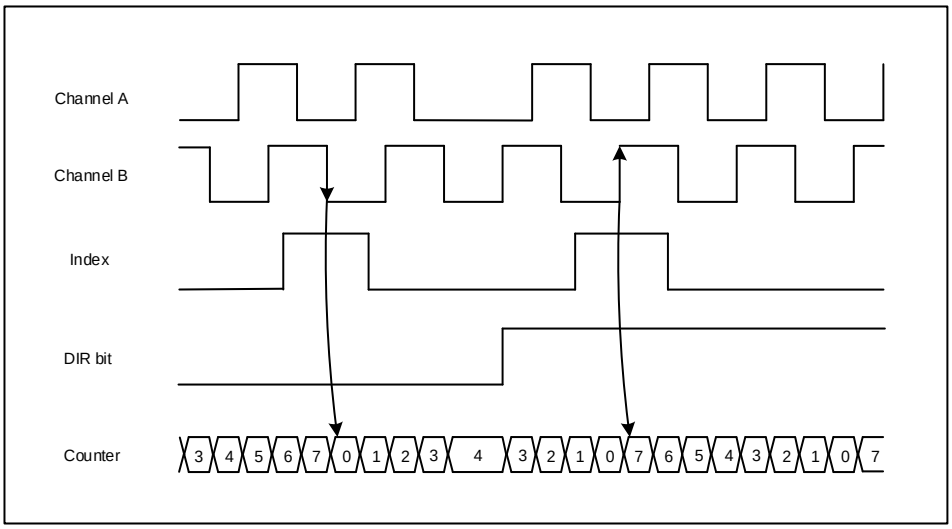


图 20-56 IPOS[1:0]=00 时无门控索引模式时的计数值

下图显示了 A 与 B 门控模式时的波形和相应值，箭头表示哪个转换过程中索引事件中断产生

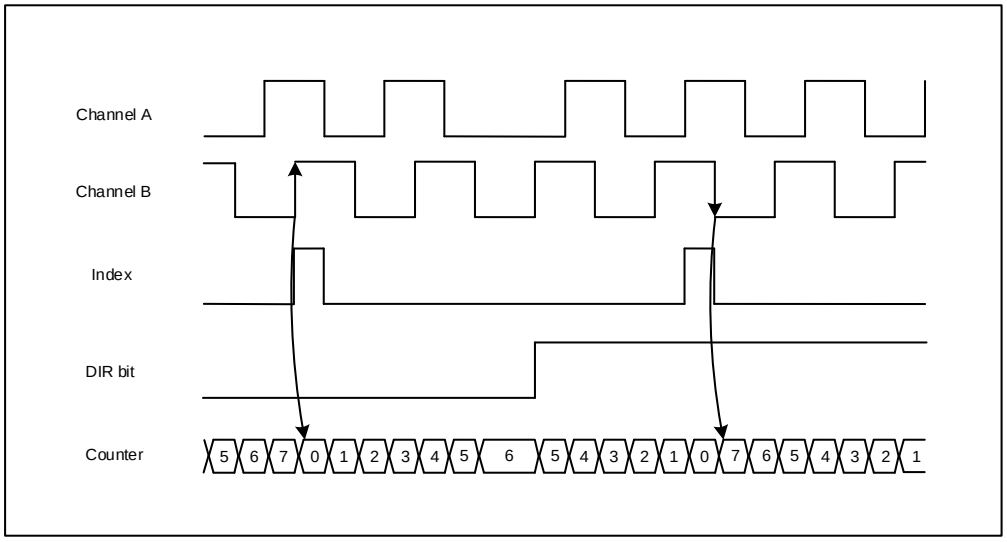


图 20-57 索引与通道 A 和 B 门控模式时的计数值

下面两张图详细说明了索引脉冲宽度小于编码器周期的 1/4 的情况

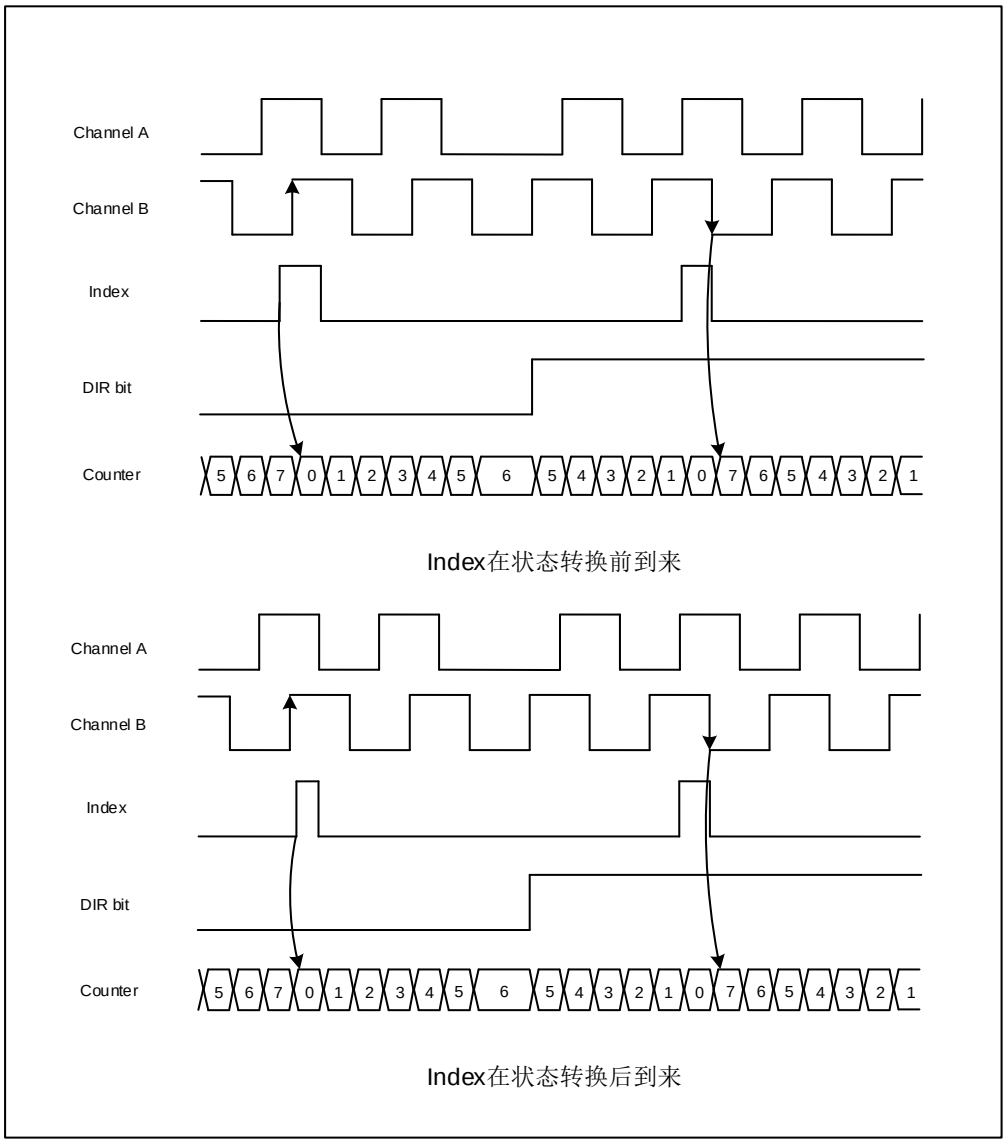


图 20-58 IPOS[1:0]=11 时窄索引脉冲时编码器模式的运行

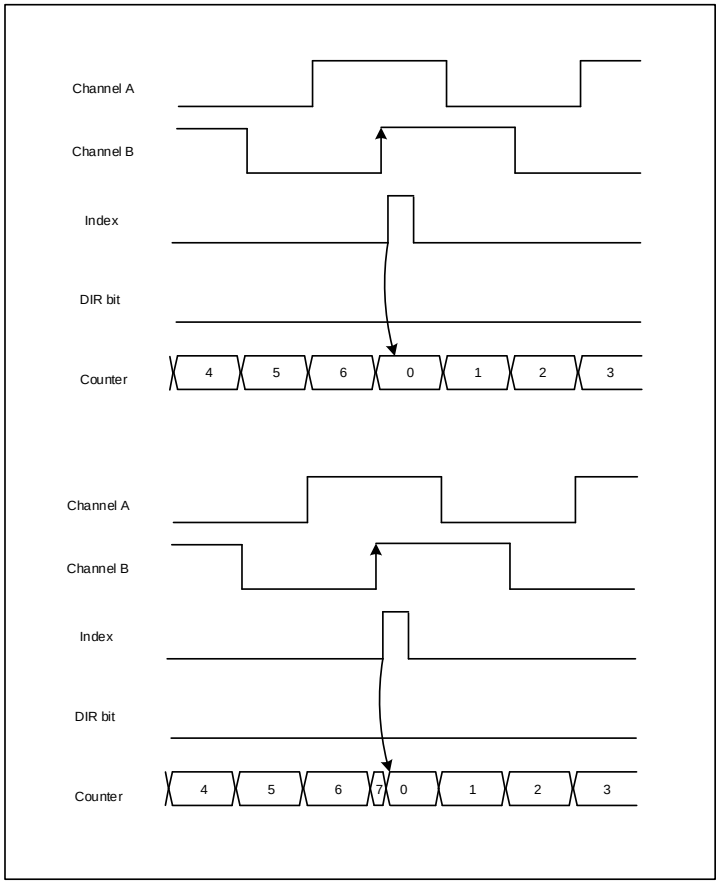


图 20-59 ARR=0x07 时 时窄索引脉冲复位计数器

下图详细显示了在 x1 和 x2 模式中如何管理索引

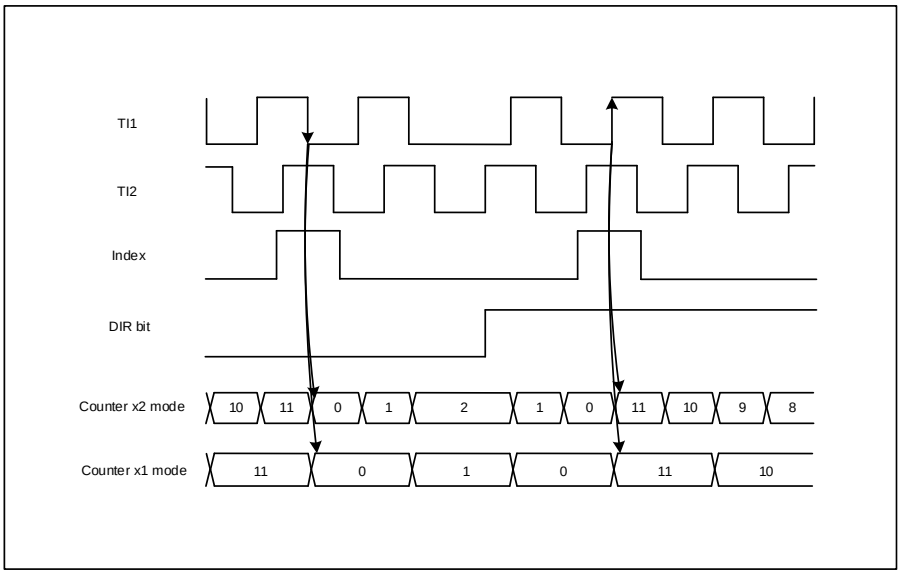


图 20-60 IPOS[1:0]=01 时在 x1 和 x2 模式中

■ 方向索引敏感性

可以通过配置 TIMx_ECR 寄存器的 IDIR[1:0]来允许索引仅在选定的计数方向上有效。

下图显示了基于 IDIR[1:0]的值，索引和计数复位事件之间的关系。

注：IDIR 必须在 IE 位被复位时才能被写入（索引模式关闭）

注：方向索引灵敏性不支持时钟+方向模式，当 SMS=1010 或 1011 时，IDIR 必须设置为 00

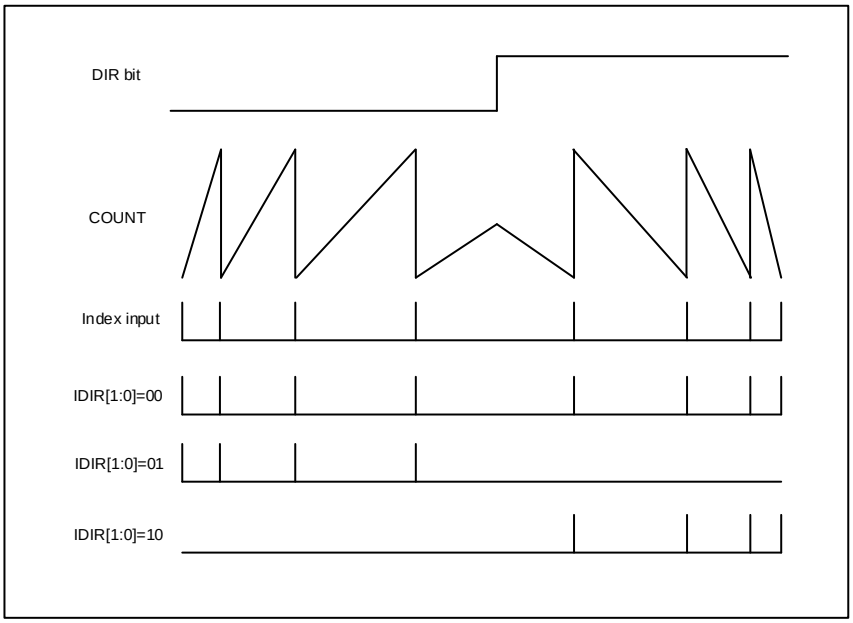


图 20-61 定向的索引敏感性

■ 特殊的第一次索引事件管理

TIMx_ECR 寄存器的 FIDX 位允许索引只进行一次，如下图所示。当第一次索引到来时，后续的索引都会忽略。如果需要，可以通过将 FIDX 位写 0 再写 1 来重启用电路。

注：当 FIDX=1 时，如果方向在位置 0(索引激活)发生改变，则可能发出两次索引（IDXF 标志置起）

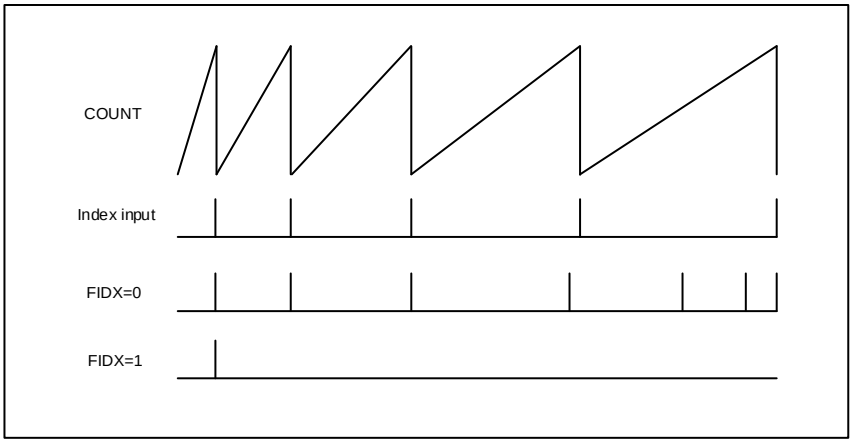


图 20-62 当 FIDX 位被置位时，计数器复位情况

■ 在非正交模式下的索引管理

下面两张图详细描述了在定向时钟模式和时钟+方向模式中索引是如何管理的。即 SMS=1010 或 1011 在这些模式中，索引的敏感性由 IPOS 决定

IPOS[0]=0:在时钟的低电平检测索引

IPOS[0]=1:在时钟的高电平检测索引

IPOS[1]没有作用。

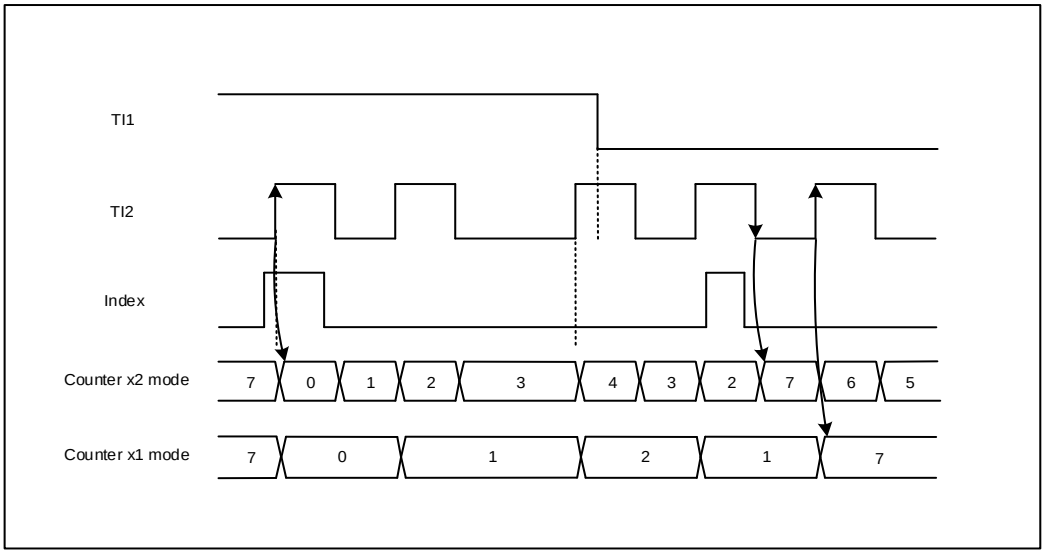


图 20-63 时钟加方向模式索引行为情况（IPOS[0]=1）

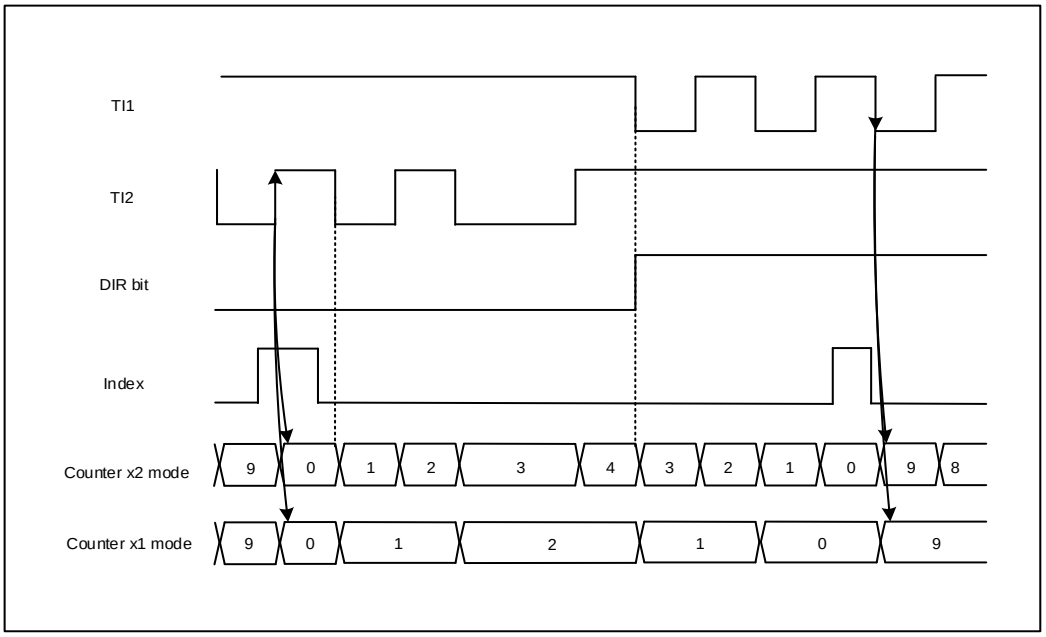


图 20-64 定向时钟模式索引行为情况（IPOS[0]=1）

■ 编码器错误管理

编码器配置中有 2 路正交信号是有效的，有必要检测转换错误。2 个输入上的读数对应 2 位格雷码来表示状态图，如图所示。每次只能改变 1 位。错误的转换将 TIMx_SR 状态寄存器中的中断标志 TERRF 拉高。若 TIMx_DIER 寄存器的 TERRIE 位置 1，则会产生转换错误中断。

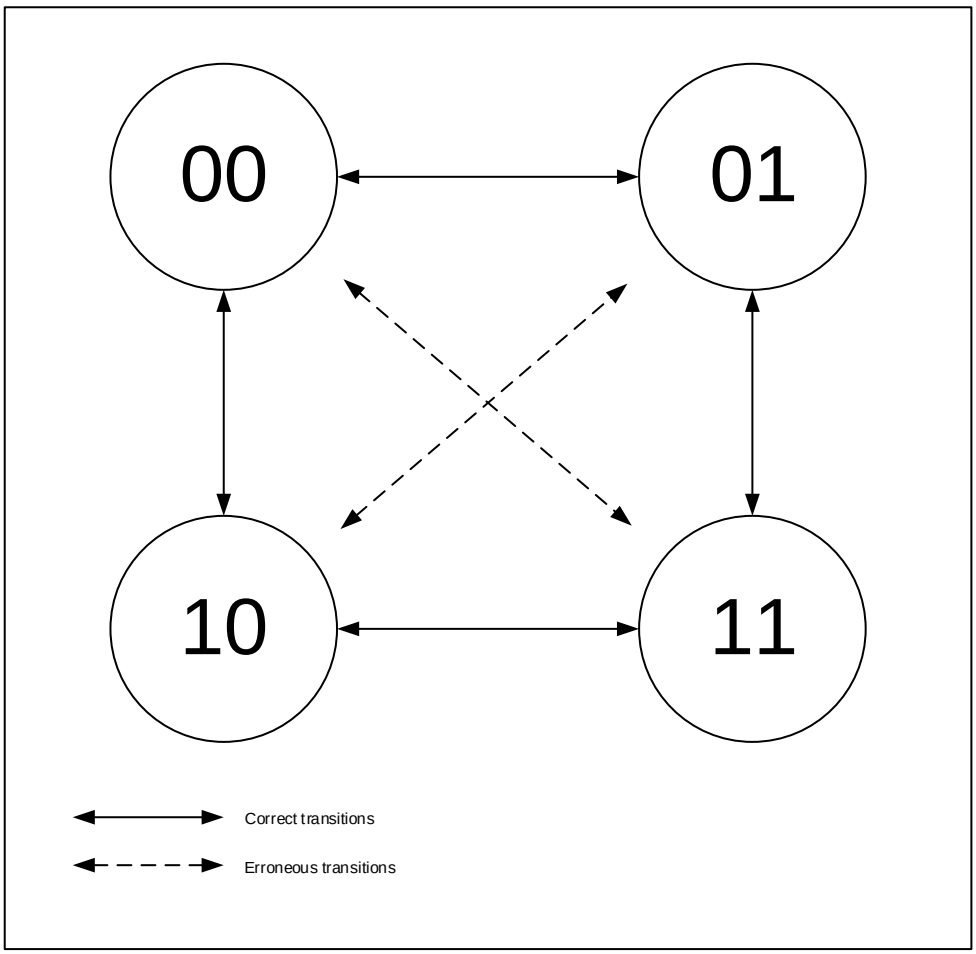


图 20-65 正交编码信号状态示意图

编码器有了索引信号后，就可以检测出导致每次旋转过程中脉冲超额的异常操作。编码器每旋转一次会提供 N 个脉冲，并会产生 4*N 的计数。索引信号每经过 4*N 时钟周期就会复位一次计数器。

如果计数器值从 TIMx_ARR 到 0 或者从 0 到 TIMx_ARR 递变而没有索引事件的话，这会被报告为索引位置错误。

溢出阈值由 TIMx_ARR 寄存器配置。1000 行编码器的结果计数值介于 0 到 3999 之间（4x 读模式中）。溢出检测阈值必须设置为 TIMx_ARR=3999+1=4000

向上计数时，错误会在延迟到 0 到 1 的转换时断言，下图表示在 A 与 B 门控模式中，对窄索引信号的处理。

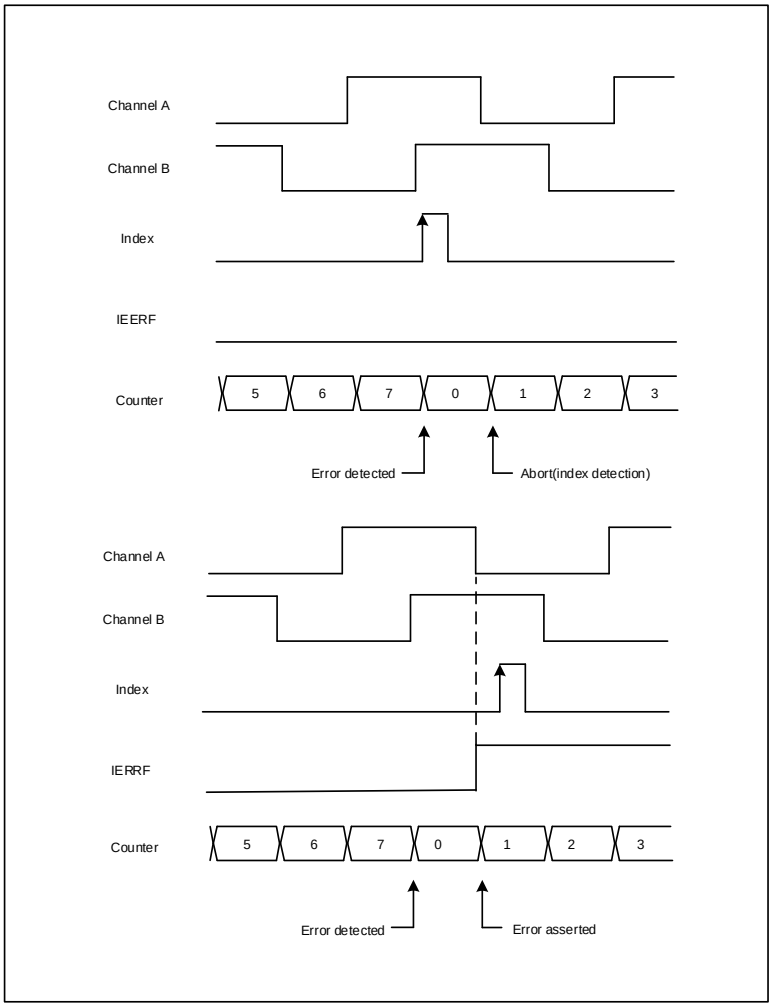


图 20-66 向上编码时错误检测

向下计数时，检测必须是以先前是从 1 到 0 的转换作为前提条件，下图表示在 A 与 B 门控模式中，对窄索引信号的处理。为了避免索引检测后编码器立即在 TIMx_ARR 和 0 两个值之间抖动而误产生假的错误认定。

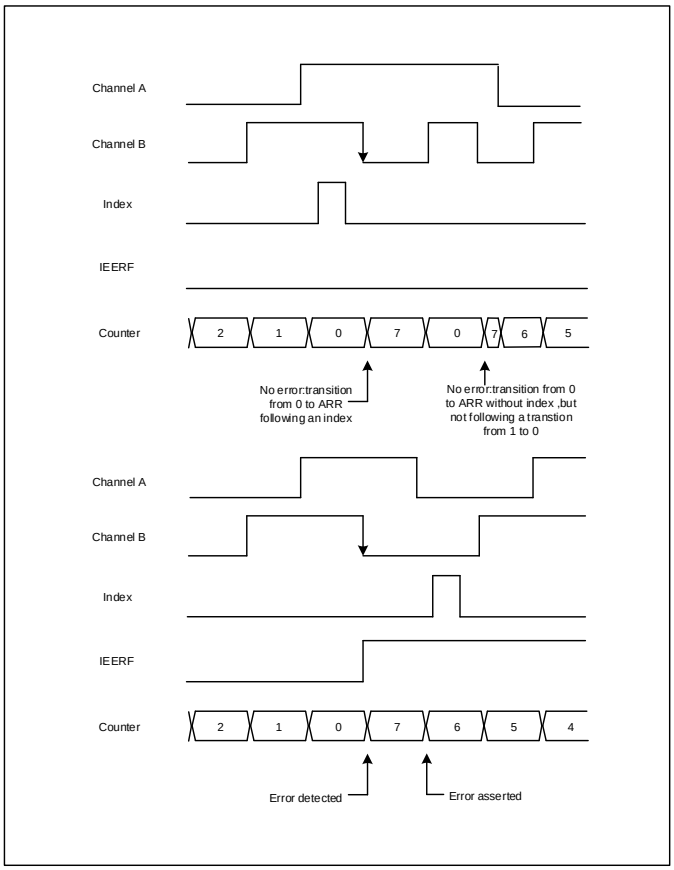


图 20-67 向下编码时错误检测

索引错误会将 TIMx_SR 状态寄存器的 IERRF 中断标志位拉高。若 TIMx_DIER 寄存器的 TERRIE 位置 1，则会产生转换错误中断。

- 功能性的编码器中断

下面中断均在编码器模式中有效

- 方向改变：在编码器模式中，计数方向的任何改变都会引起 TIMx_CR1 的 DIR 位翻转，方向的改变会拉高 TIMx_SR 状态寄存器的 DIRF 中断标志位。如果 TIMx_DIER 寄存器的 DIRIE 位使能，还会产生方向改变的中断
- 索引事件：索引事件会拉高 TIMx_SR 状态寄存器的 IDXF 中断标志位。如果 TIMx_DIER 寄存器的 IDXIE 位使能，还会产生索引中断
- 运行时从模式预加载功能更新编码器模式

从一个编码器模式切换到另一个运行模式是很有必要的，这通常在高速下完成的，以降低更新中断的速率。下图表示的是从 x4 模式切换到 x2 模式再切换到 x1 模式。

基于这个目的，SMS[3:0]可以预加载。使能位由 TIM_SMCR 寄存器的 SMSPE 位控制。从 SMS[3:0]预加载到激活值的触发信号选择可以使用 TIMx_SMCR 寄存器的 SMSPS 位控制。

- SMSPS=0:传输由 UEV 触发，该模式必须关闭索引功能(IE=0)
- SMSPS=1:传输由索引事件触发

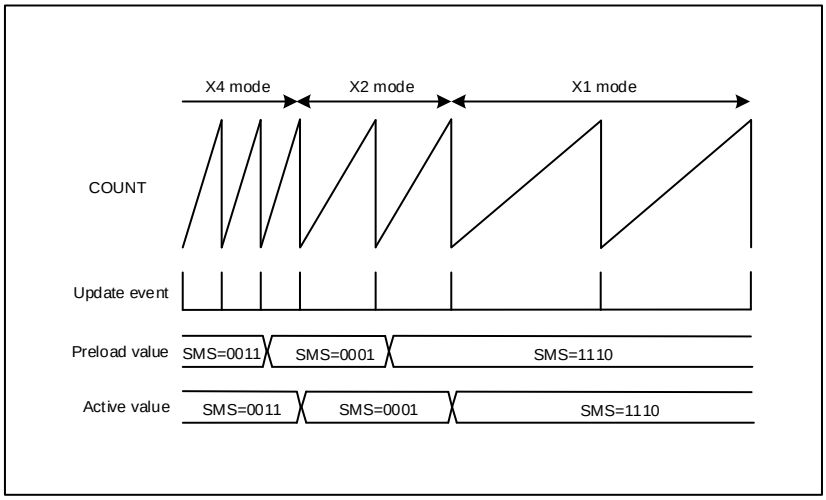


图 20-68 事件更新时编码器模式通过预加载更新

■ 编码器时钟输出

编码器的工作原理并不完全适用于高分辨率快速测量。在低速时，它需要相当长的整合时间来才能具有足够数量的时钟边沿和精确的测量。

在低速时，最好的方法是直接测量边沿到边沿的时钟周期。这时可能需要使用一个从定时器。本定时器可以通过 tim_trgo 输出编码器的时钟信号。从定时器可以执行周期测量并为每个编码器时钟边沿提供速度信息。

该模式通过设置 TIMx_CR2 寄存器的 MMS[3:0]=1000 来使能。并且仅当 SMS[3:0]配置为 0001,0010,0011,1010,1011,1100,1101,1110,1111 才能使用。其他的 SMS[3:0]值不被允许，可能会产生预期之外的行为。

20.2.18方向位输出

定时器的方向位也是有必要输出的，通过配置 TIMx_CCMR2 寄存器的 OC3M[3:0]和 OC4M[3:0]位段为 1011 实现从 OC3 和 OC4 通道输出（复制 TIMx_CR1 寄存器的 DIR 位）。

该特性用于监测编码器的计数方向(或旋转方向)，或者能提供一个中间对齐 PWM 模式中上/下阶段的指示信号。

20.2.19UIF 位重映射

TIMx_CR1 寄存器的 UIFREMAP 位能够强制连续地将 UIF 复制到定时器计数器的第 31 位(TIMxCNT[31])。这样可以允许计数器的值和由 UIFCPY 标志信号展现的一种潜在的滚动状态都可以读出。在特殊情况下，这样可以通过避免竞争条件来简化计算，例如由同时处理计数器读出和更新中断。UIF 和 UIFCPY 标志的产生没有延迟。

在 32 位定时器实现中，当设置 UIFREMAP 位时，计数器的 31 位在读取访问时被 UIFCPY 标志覆盖（计数器的最高有效位仅在写入模式下可访问）。

20.2.20定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 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 TIMx_CH1、TIMx_CH2 和 TIMx_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。

20.2.21 TIMx 定时器和外部触发的同步

TIMx 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、触发、复位+触发和门控+复位模式。

20.2.21.1 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 TIMx_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=0100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=00101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=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(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(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 TIMx_ARR=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。

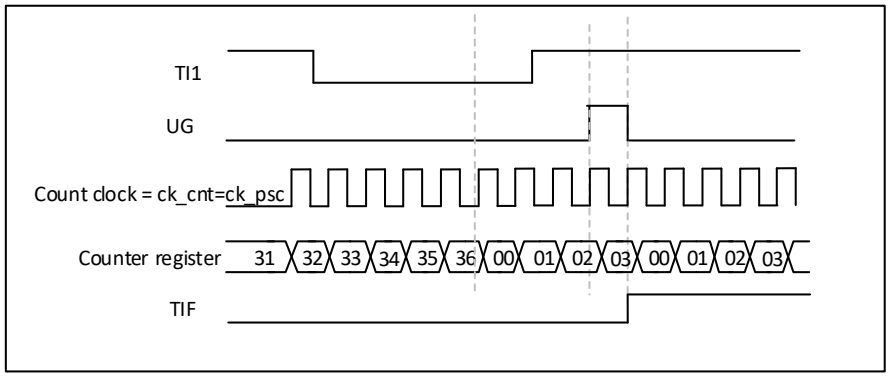


图 20-69 复位模式下的控制电路

20.2.21.2 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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，

选择 TI1 作为输入源。

- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=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 TIMx_SR 中的 TIF 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。

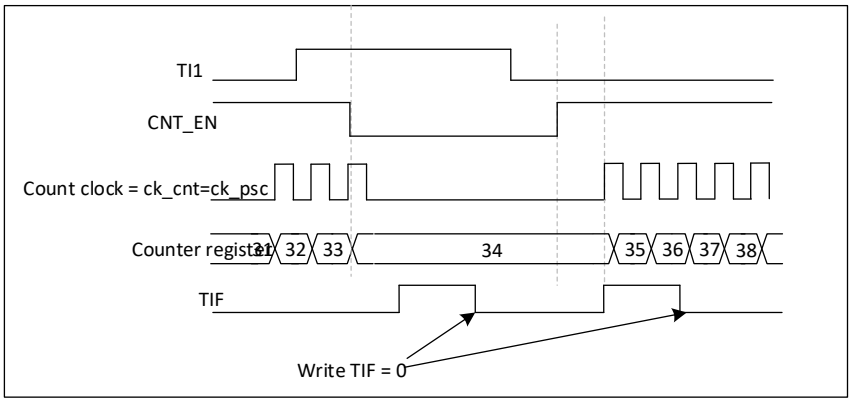


图 20-70 门控模式下的控制电路

20.2.21.3 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 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=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 TIF 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。

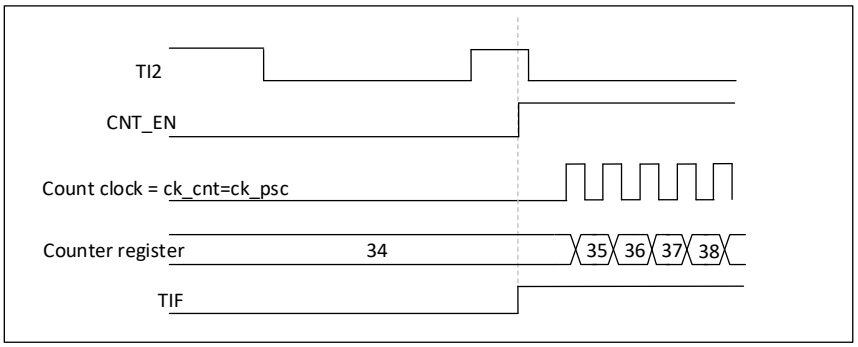


图 20-71 触发器模式下的控制电路

20.2.21.4 从模式：复位加触发组合模式

在该模式下，触发输入的上升沿会初始化计数器，并产生寄存器的更新信号，开启计数

该模式用于单脉冲模式

20.2.21.5 从模式：门控加复位组合模式

当触发输入为高时计数器的时钟才使能。当触发信号拉低时计数器停止。计数器的开始和停止都可以控制

该模式允许检测超出范围的 PWM 信号（占空比超过最大预期值）

20.2.21.6 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 TIMx_SMCR 寄存器的 TS 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 TIMx_SMCR 寄存器配置外部触发输入电路：

— ETF=0000：没有滤波

— ETPS=00：不用预分频器

— ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。

2. 按如下配置通道 1，检测 TI1 的上升沿：

— IC1F=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 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源

— 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)

3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 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。

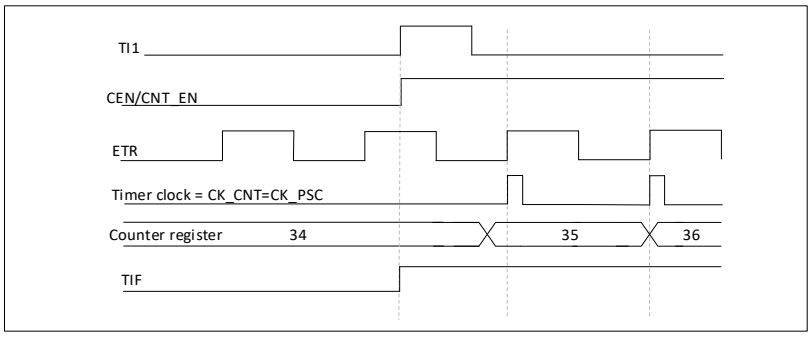


图 20-72 外部时钟模式 2 + 触发模式下的控制电路

20.2.22定时器同步

所有 TIMx 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。

20.2.22.1 使用一个定时器作为另一个定时器的预分频器

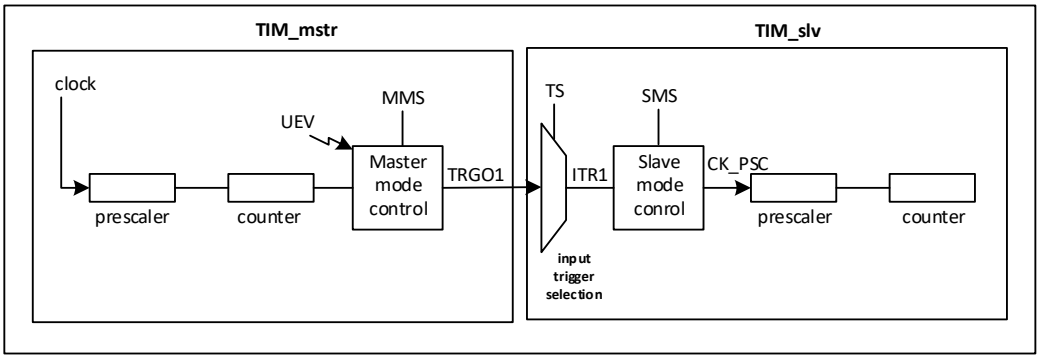


图 20-73 主/从定时器的例子

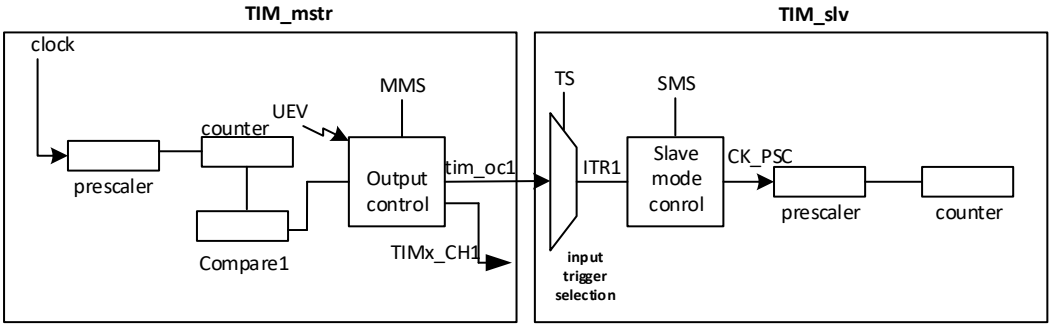


图 20-74 仅有 1 个通道定时器的主从连接示例

仅具有一个通道的计时器（见上图）不具有主模式。但是，tim_oc1 输出信号可以作为从定时器的触发器。tim_oc1 信号脉冲宽度必须编程为目标定时器的至少 2 个时钟周期，以确保从属定时器检测到触发。例如，如果目标定时器时钟比源定时器慢 4 倍，则 OC1 脉冲宽度必须为 8 个时钟周期。

如：可以配置 TIM_mstr 作为 TIM_slv 的预分频器。参考图 20-73，进行下述操作：

- 配置 TIM_mstr 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 TIM_mst_CR2 寄存器的 MMS= ‘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 TIM_mstr 的 TRGO1 输出至 TIM_slv，设置 TIM_slv_SMCR 寄存器的 TS= ‘00000’，配置 TIM_slv 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(TIM_slv_SMCR 寄存器的 SMS=111)；这样 TIM_slv 即可由 TIM_mstr 周期性的上升沿(即 TIM_mstr 的计数器溢出)信号驱动。
- 最后，必须设置相应 CEN 位分别启动两个定时器。

注：如果 OCx 已被选中为 TIM_mstr 的触发输出(MMS=1xx)，它的上升沿用于驱动 TIM_slv 的计数器。

20.2.22.2 使用一个定时器使能另一个定时器

在这个例子中，TIM_slv 的使能由 TIM_mstr 的输出比较控制。只当 TIM_mstr 的 OC1REF 为高时，TIM_slv 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(fCK_CNT=fCK_INT/3)得到。

- 配置 TIM_mstr 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM_mstr_CR2 寄存器的 MMS=100)
- 配置 TIM_mstr 的 OC1REF 波形(TIM_mst_CCMR1 寄存器)
- 配置 TIM_slv 从 TIM_mstr 获得输入触发(TIM_slv_SMCR 寄存器的 TS=00000)
- 配置 TIM_slv 为门控模式(TIM_slv_SMCR 寄存器的 SMS=101)
- 置 TIM_slv_CR1 寄存器的 CEN=1 以使能 TIM_slv
- 置 TIM_mst_CR1 寄存器的 CEN=1 以启动 TIM_mstr

注：TIM_slv 的时钟不与 TIM_mstr 的时钟同步，这个模式只影响 TIM_slv 计数器的使能信号。

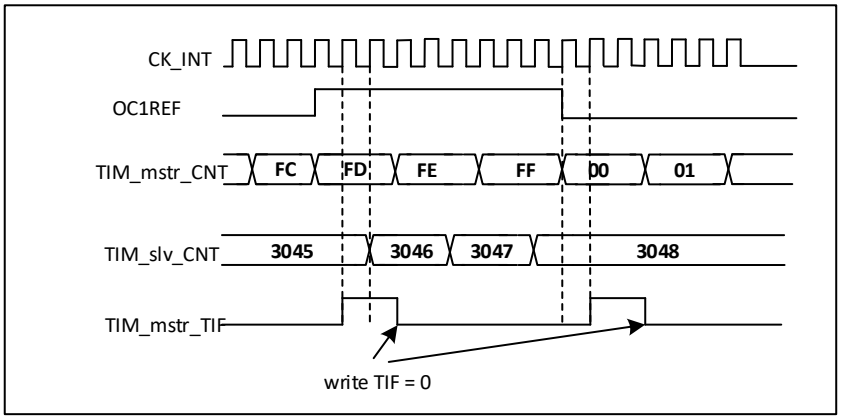


图 20-75 TIM_mstr 的 OC1REF 控制 TIM_slv

在上图的例子中，在 TIM_slv 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 TIM_mstr 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 TIMx_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步 TIM_mstr 和 TIM_slv。TIM_mstr 是主模式并从 0 开始，TIM_slv 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 TIM_mstr_CR1 的 CEN 位将禁止 TIM_mstr，TIM_slv 随即停止。

- 配置 TIM_mstr 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM_mst_CR2 寄存器的 MMS=100)。
- 配置 TIM_mstr 的 OC1REF 波形(TIM_mstr_CCMR1 寄存器)。
- 配置 TIM_slv 从 TIM_mstr 获得输入触发(TIM_slv_SMCR 寄存器的 TS=00000)
- 配置 TIM_slv 为门控模式(TIM_slv_SMCR 寄存器的 SMS=101)
- 置 TIM_mstr_EGR 寄存器的 UG= ‘1’，复位 TIM_mstr。
- 置 TIM_slv_EGR 寄存器的 UG= ‘1’，复位 TIM_slv。
- 写 ‘0XE7’ 至 TIM_slv 的计数器(TIM_slv_CNT)，初始化它为 0xE7。
- 置 TIM_slv_CR1 寄存器的 CEN= ‘1’ 以使能 TIM_slv。
- 置 TIM_mstr_CR1 寄存器的 CEN= ‘1’ 以启动 TIM_mstr。
- 置 TIM_mstr_CR1 寄存器的 CEN= ‘0’ 以停止 TIM_mstr。

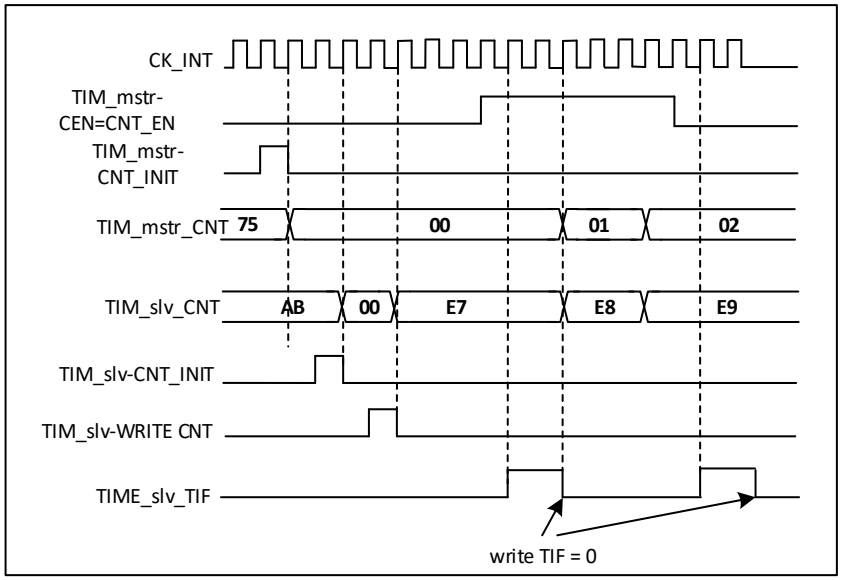


图 20-76 通过使能 TIM_mstr 可以控制 TIM_slv

20.2.22.3 使用一个定时器去启动另一个定时器

在这个例子中，使用 TIM_mstr 的更新事件使能 TIM_slv。一旦 TIM_mstr 产生更新事件，TIM_slv 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，TIM_slv 的 CEN 位被自动地置‘1’，同时计数器开始计数直到写‘0’到 TIM_slv_CR1 寄存器的 CEN 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 TIM_mstr 为主模式，送出它的更新事件(UEV)做为触发输出(TIM_mstr_CR2 寄存器的 MMS=010)。
- 配置 TIM_mstr 的周期(TIM_mstr_ARR 寄存器)。
- 配置 TIM_slv 从 TIM_mstr 获得输入触发(TIM_slv_SMCR 寄存器的 TS=00000)
- 配置 TIM_slv 为触发模式(TIM_slv_SMCR 寄存器的 SMS=110)
- 置 TIM_mstr_CR1 寄存器的 CEN=1 以启动 TIM_mstr。

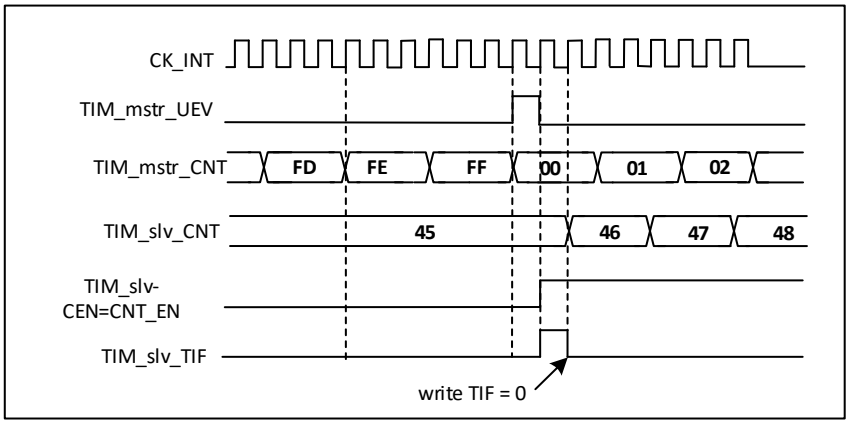


图 20-77 使用 TIM_mstr 的更新触发 TIM_slv

在上一个例子中，可以在启动计数之前初始化两个计数器。下图显示在与 0 相同配置情况下，使用触发模式而不是门控模式(TIM_slv_SMCR 寄存器的 SMS=110)的动作。

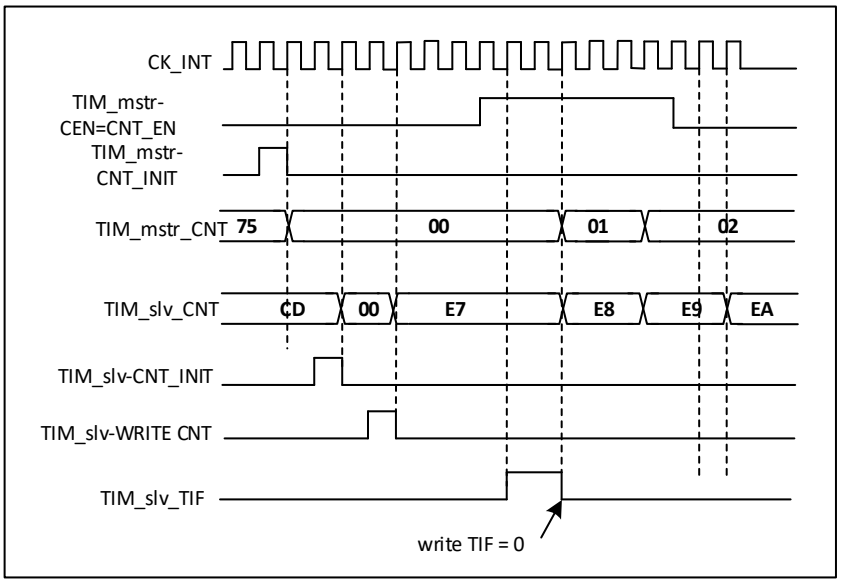


图 20-78 利用 TIM_mstr 的使能触发 TIM_slv

20.2.22.4 使用一个外部触发同步地启动 2 个定时器

这个例子中当 TIM_mstr 的 TI1 输入上升时使能 TIM_mstr，使能 TIM_mstr 的同时使能 TIM_slv。保证计数器的对齐，TIM_mstr 必须配置为主/从模式(对应 TI1 为从，对应 TIM_slv 为主)：

- 配置 TIM_mstr 为主模式，送出它的使能做为触发输出(TIM_mstr_CR2 寄存器的 MMS=001)。
- 配置 TIM_mstr 为从模式，从 TI1 获得输入触发(TIM_mstr_SMCR 寄存器的 TS=00100)。
- 配置 TIM_mstr 为触发模式(TIM_mstr_SMCR 寄存器的 SMS=110)。
- 配置 TIM_mstr 为主/从模式，TIM_mstr_SMCR 寄存器的 MSM=1。
- 配置 TIM_slv 从 TIM_mstr 获得输入触发(TIM_slv_SMCR 寄存器的 TS=00000)
- 配置 TIM_slv 为触发模式(TIM_slv_SMCR 寄存器的 SMS=110)。

当 TIM_mstr 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 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 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(TIMx_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在 TIM_mstr 的 CNT_EN 和 CK_PSC 之间有个延迟。

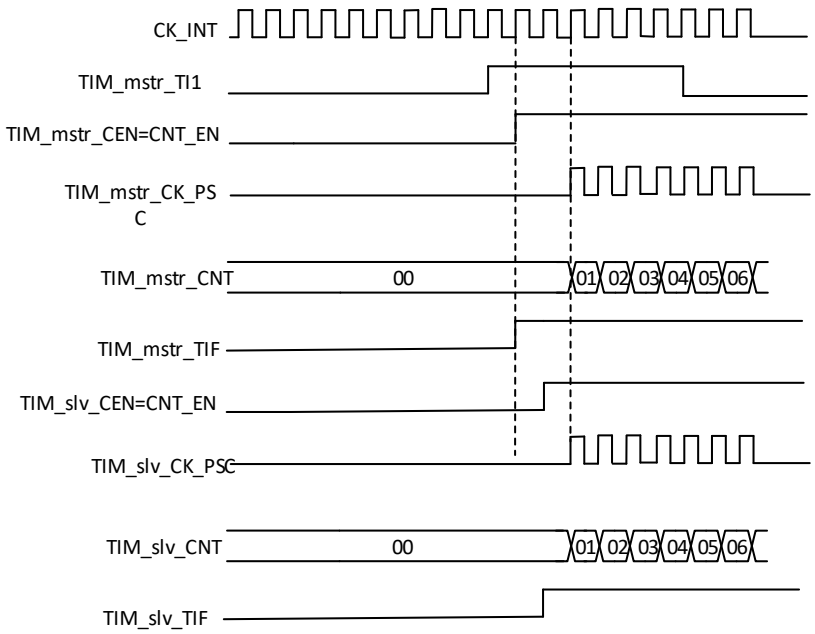


图 20-79 使用 TIM_mstr 的 TI1 输入触发 TIM_mstr 和 TIM_slv

20.2.23DMA 突发模式

TIMx 定时器当发生单个事件时有生成多个 DMA 请求的能力。主要目的是能够多次重新编程定时器的部分功能而无需软件的开销。但是也可以在固定的间隔时间内读取一行中的几个寄存器。

DMA 控制器的目的地是唯一的并且必须指向虚拟寄存器 TIMx_DMAR.当给定的定时器事件发生，定时器发起一系列 DMA 请求。每个写入 TIMx_DMAR 寄存器的操作实际上是重定向一个定时器寄存器的。

TIMx_DCR 寄存器中的 DBL[4:0]位设置 DMA 突发长度。当对 TIMx_DMR 地址进行读取或写入访问时，定时器会识别突发传输，即传输次数（以半个字或字节为单位）

TIMx_DCR 的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址(当通过 TIMx_DMAR 地址完成读写操作)。DBA 定义为从 TIMx_CR1 寄存器地址开始的偏移量。

例子：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发特性是用于发生更新事件时更新 CCRx 寄存器中的内容，通过 DMA 传输半字到 CCRx。

通过以下步骤来完成：

1. 如下配置对应的 DMA 通道：
    1) DMA 通道外设地址是 DMAR 寄存器的地址
    2) DMA 通道存储地址是 RAM 中的 BUFF 地址，包含了 DMA 传给 CCRx 的数据
    3) 传输的数据个数=3
    4) 轮询模式关闭
2. 通过配置 DBA 和 DBL 实现配置 DCR 寄存器：DBL=3，DBA=0XE
3. 使能定时器更新 DMA 请求（UDE=1）

4. 使能定时器
5. 使能 DMA 通道

这个例子此示例适用于每个 CCRx 寄存器更新一次的情况。如果每个 CCRx 都更新两次，数据的传输个数就需要设置为 6。举例假设 RAM 的 buffer 中包含了 data1, data2, data3, data4, data5 and data6。数据依次传输到 CCRx 中：第一次更新 DMA 请求，data1 传输给 CCR2，data2 传输给 CCR3，data3 传输给 CCR4;第二次更新 DMA 请求,data4 传输给 CCR2,data5 传输给 CCR3,data6 传输给 CCR4。

注：空值可以写入到保留寄存器

20.2.24 TIM2/3/4/5/18 DMA 请求

TIM2/3/4/5/18 可以生成 DMA 请求，如下表所示:

表 20-4 DMA 请求

DMA 请求源	DMA 使能控制位
更新	UDE
比较/捕获 1	CC1DE
比较/捕获 2	CC2DE
比较/捕获 3	CC3DE
比较/捕获 4	CC4DE
触发	TDE

20.2.25 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_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

调试模式下的行为可以使用调试支持模块（DBG）中每个定时器的专用配置进行编程

为了安全的考虑，当计数器停止时，输出被禁止（如果 MOE 被复位了）。输出即可被强制到一个无效电平（OSSI=1），或者由 GPIO 控制器接管（OSSI=0），典型的是强制为高阻。

更多的细节，见 DBG 模块描述

20.2.26 TIM2/3/4/5/18 低功耗模式

模式	说明
Sleep	无效果，外围设备处于活动状态。中断可能导致设备退出睡眠状态
Stop	定时器操作停止，寄存器内容保留。不能产生中断
Standby	定时器已断电，退出待机模式后必须重新初始化。

20.2.27 TIM2/3/4/5/18 中断

中断事件	事件标志	中断使能控制位
更新	UIF	UIE
比较/捕获 1	CC1IF	CC1IE
比较/捕获 2	CC2IF	CC2IE
比较/捕获 3	CC3IF	CC3IE

比较/捕获 4	CC4IF	CC4IE
COM	COM	COMIE
触发	TIF	TIE
编码器索引	IDXF	IDXIE
编码器方向	DIRF	DIRIE
编码器索引错误	IERRF	IERRIE
编码器传输错误	TERRF	TERRIE

20.3 寄存器描述

20.3.1 TIMx 控制寄存器 1（TIMx_CR1）（x=2/3/4/5/18）

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DITHEN	UIFREMAP	Res	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:13	保留	-	-	-
12	DITHEN	RW	0	抖动使能（Dithering enbale） 0：抖动关闭 1：抖动打开 注：该位只能在 CEN 位复位时才能修改
11	UIFREMAP	RW	0	UIF 状态位重映射使能（UIF status bit remapping） 0：无重映射，UIF 状态位不复制到 TIMx_CNT 寄存器的第 31 位 1：重映射，UIF 状态位复制到 TIMx_CNT 寄存器的第 31 位
10	保留	-	-	-
9:8	CKD	RW	2’h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:5	CMS	RW	2’h0	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(TIMx_CCMRx 寄存器中 CCxS=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(TIMx_CCMRx 寄存器中 CCxS=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(TIMx_CCMRx 寄存器中 CCxS=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(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	RW	0	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。
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

20.3.2 TIMx 控制寄存器 2（TIMx_CR2）（x=2/3/4/5/18)

Address offset:0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						MMS[3]	Res								
						RW									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								TI1S	MMS[2:0]			CCDS	Res		
								RW	RW	RW	RW	RW			

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	-
25	MMS[3]	RW	0	详见 MMS[2:0]描述
24:8	保留	-	-	-
7	TI1S	RW	0	TI1 选择（TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
6:4	MMS	RW	3’h0	主模式选择（Master mode selection) 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: 0000：复位 – TIMx_EGR 寄存器的 UG 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。 0001: 使能 – 计数器使能信号 CNT_EN 可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 CEN 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 TIMx_SMCR 寄存器中 MSM 位的描述)。 0010: 更新 – 更新事件被选为触发输出(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。 0011：比较脉冲 – 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 0100：比较 – OC1REFC 信号被用于作为触发输出(TRGO)。 0101：比较 – OC2REFC 信号被用于作为触发输出(TRGO)。 0110：比较 – OC3REFC 信号被用于作为触发输出(TRGO)。 0111：比较 – OC4REFC 信号被用于作为触发输出(TRGO)。 1000: 编码器时钟输出-编码器时钟信号作为触发输出（TRGO）。该编码值仅在 SMS[3:0]的值为 0001、0010、0011、1010、1011、1100、1101、1110、1111 时有效。其他的 SMS 值可能导致预期之外的效果。 1001-1111：保留 注意: 1.从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。 2.若主从定时器不在同一总线上，主模式应该配置为能被从定时器采到的宽度。
3	CCDS	RW	0	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。
2:0	保留	-	-	-

20.3.3 TIMx 从模式控制寄存器（TIMx_SMCR） (x=2/3/4/5/18)

Address offset:0x08

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						SMSPS	SMSPE	Res		TS[4:3]		Res			SMS[3]
						RW	RW			RW	RW				RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	-	-
25	SMSPS	RW	0	SMS 预加载源（SMS preload source） 该位选择事件是否可以触发 SMS[3:0] 0：传输由定时器的更新事件触发 1：传输由索引事件触发
24	SMSPE	RW	0	SMS 预加载使能（SMS preload enable） 该位选择 SMS[3:0]是否可以预加载 0：SMS[3:0]可以被预加载 1：SMS[3:0]不可以被预加载
23:22	保留	-	-	-
21:20	TS[4:3]	RW	2’h0	详见 TS[2:0]描述
19:17	保留	-	-	-
16	SMS[3]	RW	0	详见 SMS[2:0]描述
15	ETP	RW	0	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。
14	ECE	RW	0	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 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=00111)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(TS 位不能是‘00111’)。 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。
13:12	ETPS	RW	2’h0	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 TIMxCLK 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。
11:8	ETF	RW	4’h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
7	MSM	RW	0	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。
6:4	TS	RW	3’h0	触发选择（Trigger selection) 选择用于同步计数器的触发输入。 00000：内部触发 0(ITR0) 00001：内部触发 1(ITR1) 00010：内部触发 2(ITR2) 00011：内部触发 3(ITR3) 00100：TI1 边沿检测（TI1F_ED） 00101：滤波后的定时器输入 1（TI1FP1） 00110：滤波后的定时器输入 2（TI1FP2） 00111：外部触发输入（ETRF） 01000：内部触发 4(ITR4) 01001：内部触发 5(ITR5) 01010：内部触发 6(ITR6) 01011：内部触发 7(ITR7) 01100：内部触发 8(ITR8) 01101：内部触发 9(ITR9) 01110：内部触发 10(ITR10) 01111：内部触发 11(ITR11) 10000：内部触发 12(ITR12) 10001：内部触发 13(ITR13) 10010：内部触发 14(ITR14) 10011：内部触发 15(ITR15) 10100：内部触发 16(ITR16) 10101：内部触发 17(ITR17) 10110：内部触发 18(ITR18)

				其它：保留 更多有关 ITRx 的细节，参见系统级联表 注：这些位只能在未用到(如 SMS=0000)时被改变，以避免在改变时产生错误的边沿检测。
3	OCCS	RW	0	OCREF 清除选择位（OCREF clear selection） 这位选择了 OCREF 的清除源。 0：OCREF_CLR_INT 连接 OCREF_CLR 输入 1：OCREF_CLR_INT 连接 ETRF
2:0	SMS	RW	3'h0	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) 0000：关闭从模式 – 如果 CEN=1，则预分频器直接由内部时钟驱动。 0001: 正交编码器模式 1，x2 模式– 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 0010: 正交编码器模式 2，x2 模式– 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 0011：正交编码器模式 3，x4 模式–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。 0100: 复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 0101：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。 0110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 0111：外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 1000：复位+触发组合模式：-触发输入（TRGI）的上升沿初始化计数器，生成寄存器更新并启动计数器 1001：门控+复位组合模式：-触发输入（TRGI）为高时计数器时钟使能。一旦触发信号拉低时计数器就停止并复位。计数器的开始和停止均可控。 1010：编码器模式：时钟+方向，x2 模式 1011：编码器模式：时钟+方向，x1 模式，TI2FP2 的敏感性由 CC2P 设置 1100：编码器模式：定向时钟，x2 模式 1101: 编码器模式: 定向时钟，x1 模式，TI1FP1，TI2FP2 的敏感性由 CC1P 和 CC2P 设置 1110：正交编码器模式：x1 模式。仅在 TI1FP1 的边沿计数，沿敏感性由 CC1P 设置 1111：正交编码器模式：x1 模式。仅在 TI2FP2 的边沿计数，沿敏感性由 CC2P 设置 注：如果 TI1F_ED 被选为触发输入(TS=00100)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：在编码器模式下，不要使用 UEV 作为 tim_trgo 输出信号，（即 mms 不能配置为 010。

20.3.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x=2/3/4/5/18）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TERRIE	IERRIE	DIRIE	IDXIE	Res			
								RW	RW	RW	RW				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	RW		RW	RW	RW	RW	RW	RW	RW		RW	RW			

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	TERRIE	RW	0	传输错误中断使能（Transition error interrupt enable） 0：传输错误中断不使能 1：传输错误中断使能
22	IERRIE	RW	0	索引错误中断使能（Index error interrupt enable） 0：索引错误中断不使能 1：索引错误中断使能
21	DIRIE	RW	0	方向改变中断使能（Direction change interrupt enable） 0：方向改变中断不使能 1：方向改变中断使能
20	IDXIE	RW	0	索引中断使能（Index interrupt enable） 0：索引中断不使能 1：索引中断使能
19:15	保留	-	-	-
14	TDE	RW	0	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。
13	保留	-	-	-
12	CC4DE	RW	0	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。
11	CC3DE	RW	0	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。。
10	CC2DE	RW	0	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。
9	CC1DE	RW	0	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。。
8	UDE	RW	0	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。
7	BIE	RW	0	允许刹车中断（Break interrupt enable) 0：禁止刹车中断;

				1：允许刹车中断。
6	TIE	RW	0	触发中断使能（Trigger interrupt enable) 0：禁止触发中断; 1：使能触发中断。
5	保留	-	-	-
4	CC4IE	RW	0	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。
3	CC3IE	RW	0	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。
2	CC2IE	RW	0	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。
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断; 1：允许更新中断。

20.3.5 TIMx 状态寄存器（TIMx_SR）（x=2/3/4/5/18）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TERRF	IERRF	DIRF	IDXF	Res			
								RC_W0	RC_W0	RC_W0	RC_W0				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res			CC4OF	CC3OF	CC2OF	CC1OF	Res		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			RC_W0	RC_W0	RC_W0	RC_W0			RC_W0		RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	TERRF	RC_W0	0	传输错误中断标志（Transition error interrupt flag） 当编码器模式中出现传输错误，该标志由硬件置位。可通过软件清零或者写清零 0：未检测到编码器传输错误 1：检测到编码器传输错误
22	IERRF	RC_W0	0	索引错误中断标志（Index error interrupt flag） 当检测到索引出现错误，该标志由硬件置位。可通过软件清零或者写清零
21	DIRF	RC_W0	0	方向改变中断标志（Direction change interrupt flag） 当编码器模式中出现方向改变（TIMx_CR 中的 DIR 位改变），该标志由硬件置位。可通过软件清零或者写清零 0：未检测到方向改变

				1：检测到方向改变
20	IDXF	RC_W0	0	索引中断标志（Index interrupt flag） 当检测到索引事件，该标志由硬件置位。可通过软件清零或者写清零 0：无索引事件发生 1：有索引事件发生
19:13	保留	-	-	-
12	CC4OF	RC_W0	0	捕获/比较 4 重复捕获标记（Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	RC_W0	0	捕获/比较 3 重复捕获标记（Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	RC_W0	0	捕获/比较 2 重复捕获标记（Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	RC_W0	0	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 TIMx_CCR1 寄存器时，CC1IF 的状态已经为‘1’。
8:7	保留	-	-	-
6	TIF	RC_W0	0	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。
5	保留	-	-	-
4	CC4IF	RC_W0	0	捕获/比较 4 中断标记（Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	RC_W0	0	捕获/比较 3 中断标记（Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记（Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	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： 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 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，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1: 计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=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 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR)）。

20.3.6 TIMx 事件产生寄存器（TIMx_EGR）（x=2/3/4/5/18）

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									TG	Res	CC4G	CC3G	CC2G	CC1G	UG
									W		W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	TG	W	0	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：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5	保留	-	-	-
4	CC4G	W	0	产生捕获/比较 4 事件（Capture/Compare 4 generation) 参考 CC1G 描述。
3	CC3G	W	0	产生捕获/比较 3 事件（Capture/Compare 3 generation) 参考 CC1G 描述。
2	CC2G	W	0	产生捕获/比较 2 事件（Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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 TIMx_ARR 的值。

20.3.7 TIMx 捕获/比较模式控制寄存器 1（TIMx_CCMR1）（x=2/3/4/5/18）

Address offset:0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							OC2M[3]	Res.							OC1M[3]
							RW								RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

20.3.7.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC2M[3]	RW	0	详见 OC2M[2:0]描述
23:17	保留	-	-	-
16	OC1M[3]	RW	0	详见 OC1M[2:0]描述
15	OC2CE	RW	0	输出比较 2 清零使能（Output Compare 2 clear enable)
14:12	OC2M	RW	3'h0	输出比较 2 模式（Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能（Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能（Output Compare 2 fast enable)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	输出比较 1 模式（Output Compare 1 mode) 这些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 CC1P、CC1NP 位。 0000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。

				0011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 0100：强制为无效电平。强制 OC1REF 为低。 0101：强制为有效电平。强制 OC1REF 为高。 0110: PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 0111: PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 1000: 可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下，通道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为无效 1001: 可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 1010：保留 1011：保留 1100：组合 PWM 模式 1 –OC1REF 有与 PWM 模式 1 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 或逻辑组成 1101：组合 PWM 模式 2 –OC1REF 有与 PWM 模式 2 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 与逻辑组成 1110：非对称 PWM 模式 1：OC1REF 有与 PWM 模式 1 中相同的行为，向上计数时 OC1REFC 输出 OC1REF，向下计数时 OC1REFC 输出 OC2REF 1111：非对称 PWM 模式 2：OC1REF 有与 PWM 模式 2 中相同的行为，向上计数时 OC1REFC 输出 OC1REF，向下计数时 OC1REFC 输出 OC2REF
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。
2	OC1FE	RW	0	输出比较 1 快速使能（Output Compare 1 fast enable) 该位用于加快输出对触发输入事件的响应。必须在单脉冲模式（TIMx_CR1 寄存器的 OPM 位置 1）中使用，当触发到来时实现脉冲快速输出。 0: 根据计数器与 CCR1 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象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。仅能用于 PWM 模式 1 和 PWM 模式 2
1:0	CC1S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

20.3.7.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC2F	RW	4'h0	输入捕获 2 滤波器（Input capture 2 filter)
11:10	IC2PSC	RW	2'h0	输入/捕获 2 预分频器（Input capture 2 prescaler)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F	RW	4'h0	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 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。

1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

20.3.8 TIMx 捕获/比较模式控制寄存器 2（TIMx_CCMR2）（x=2/3/4/5/18）

Address offset:0x1C

Reset value：0x0000 0000

参看以上 CCMR1 寄存器的描述

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							OC4M[3]	Res.							OC3M[3]
							RW								RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

20.3.8.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC4M[3]	RW	0	详见 OC4M[2:0]描述
23:17	保留	-	-	-
16	OC3M[3]	RW	0	详见 OC3M[2:0]描述
15	OC4CE	RW	0	输出比较 4 清零使能（Output Compare 4 clear enable)
14:12	OC4M	RW	3'h0	输出比较 4 模式（Output Compare 4 mode)
11	OC4PE	RW	0	输出比较 4 预装载使能（Output Compare 4 preload enable)
10	OC4FE	RW	0	输出比较 4 快速使能（Output Compare 4 fast enable)
9:8	CC4S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7	OC3CE	RW	0	输出比较 3 清'0'使能（Output Compare 1clear enable)
6:4	OC3M	RW	3'h0	输出比较 3 模式（Output Compare 3 mode) 这些位定义了输出参考信号 OC3REF 的动作，而 OC3REF 决定了 OC3、OC3N 的值。OC3REF 是高电平有效，而 OC3、OC3N 的有效电平取决于 CC3P、CC3NP 位。 0000：冻结。输出比较寄存器 TIMx_CCR3 与计数器 TIMx_CNT 间的比较对 OC3REF 不起作用；

				0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR3)相同时，强制 OC3REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR3)相同时，强制 OC3REF 为低。 0011：翻转。当 TIMx_CCR3=TIMx_CNT 时，翻转 OC3REF 的电平。 0100：强制为无效电平。强制 OC3REF 为低。 0101：强制为有效电平。强制 OC3REF 为高。 0110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR3 时通道 3 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR3 时通道 3 为无效电平(OC3REF=0)，否则为有效电平(OC3REF=1)。 0111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR3 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR3 时通道 1 为有效电平，否则为无效电平。 1000：可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下，通道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为不活动 1001：可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 1010：比较脉冲：当 CCR3 匹配事件发生时在 OC3REF 上产生脉冲，脉冲可由 TIMx_ECR 的 PWPRSC[2:0]和 PW[7:0]配置 1011：方向输出：OC3REF 信号由 DIR 位覆盖 1100：组合 PWM 模式 1 –OC3REF 有与 PWM 模式 1 中相同的行为，OC3REFC 是由 OC3REF 和 OC4REF 或逻辑组成 1101：组合 PWM 模式 2 –OC3REF 有与 PWM 模式 2 中相同的行为，OC3REFC 是由 OC3REF 和 OC4REF 与逻辑组成 1110：非对称 PWM 模式 1：OC3REF 有与 PWM 模式 1 中相同的行为，向上计数时 OC3REFC 输出 OC3REF，向下计数时 OC3REFC 输出 OC4REF 1111：非对称 PWM 模式 2：OC3REF 有与 PWM 模式 2 中相同的行为，向上计数时 OC3REFC 输出 OC3REF，向下计数时 OC3REFC 输出 OC4REF
3	OC3PE	RW	0	输出比较 3 预装载使能（Output Compare 3 preload enable)
2	OC3FE	RW	0	输出比较 3 快速使能（Output Compare 3 fast enable)
1:0	CC3S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

20.3.8.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC4F	RW	4'h0	输入捕获 4 滤波器（Input capture 4 filter)
11:10	IC4PSC	RW	2'h0	输入/捕获 4 预分频器（Input capture 4 prescaler)
9:8	CC4S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7:4	IC3F	RW	4'h0	输入捕获 3 滤波器（Input capture 3 filter)
3:2	IC3PSC	RW	2'h0	输入/捕获 3 预分频器（Input capture 3 prescaler)
1:0	CC3S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

20.3.9 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x=2/3/4/5/18）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res	CC4P	CC4E	CC3NP	Res	CC3P	CC3E	CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
RW		RW	RW	RW		RW	RW	RW		RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	CC4NP	RW	0	输入/捕获 4 互补输出极性（Capture/Compare 3 output polarity) 参考 CC1NP 的描述。
14	保留	-	-	-
13	CC4P	RW	0	输入/捕获 4 输出极性（Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	RW	0	输入/捕获 4 输出使能（Capture/Compare 4 output enable) 参考 CC1E 的描述。
11	CC3NP	RW	0	输入/捕获 3 互补输出极性（Capture/Compare 3 output polarity) 参考 CC1NP 的描述。
10	保留	-	-	-

9	CC3P	RW	0	输入/捕获 3 输出极性（Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	CC3E	RW	0	输入/捕获 3 输出使能（Capture/Compare 3 output enable) 参考 CC1E 的描述。
7	CC2NP	RW	0	输入/捕获 2 互补输出极性（Capture/Compare 2 output polarity) 参考 CC1NP 的描述。
6	保留	-	-	-
5	CC2P	RW	0	输入/捕获 2 输出极性（Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能（Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性（Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效； 1：OC1N 低电平有效。
2	保留	-	-	-
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿： TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。
0	CC1E	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 TIMx_CCR1 寄存器。 0：捕获禁止； 1：捕获使能。 注： 对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表 20-5 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

CCxE 位	OCx 输出状态
0	输出禁止（与定时器断开）OCx=0,OCx_EN=0
1	OCxREF+极性，OCx=OCREF xor CCxP,OCx_EN=1

20.3.10 TIMx 计数器（TIMx_CNT）（x=2）

Address offset:0x24

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	UIFCPY	RW	0	如果 UIFREMAP=0 CNT[31]：计数器值的最高有效位 如果 UIFREMAP=1 UIFCPY:UIF Copy 此位是 TIMx_ISR 寄存器 UIF 位的只读副本
30:0	CNT	RW	31'h0	计数器的值（Counter value)

20.3.11 TIMx 计数器（TIMx_CNT）（x=3/4/5/18）

Address offset:0x24

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	UIFCPY	RW	0	如果 UIFREMAP=0 CNT[31]：计数器值的最高有效位 如果 UIFREMAP=1 UIFCPY:UIF Copy 此位是 TIMx_ISR 寄存器 UIF 位的只读副本
30:16	保留	-	-	-
15:0	CNT	RW	16'h0	计数器的值（Counter value)

20.3.12 TIMx 预分频器（TIMx_PSC）（x=2/3/4/5/18）

Address offset:0x28

Reset value：0x0000

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

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16'h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

20.3.13 TIMx 自动重装载寄存器（TIMx_ARR）（x=2）

Address offset:0x2C

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[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
ARR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	ARR	RW	32'hFFFFFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。 无抖动模式（DITHEN=0）： 该寄存器保持自动加载值 抖动模式（DITHEN=1）： 该寄存器保持整数部分 ARR[31:4]，ARR[3:0]包含抖动部分

20.3.14 TIMx 自动重装载寄存器（TIMx_ARR）（x=3/4/5/18）

Address offset:0x2C

Reset value:0x0FFFF

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

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	ARR	RW	20'h0FFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。 无抖动模式（DITHEN=0）： 该寄存器保持自动加载值 抖动模式（DITHEN=1）： 该寄存器保持整数部分 ARR[19:4]，ARR[3:0]包含抖动部分

20.3.15 TIMx 捕获/比较寄存器 1（TIMx_CCR1） (x=2)

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:0	CCR1	RW/RO	32’h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR1[31:4]，CCR1[3:0]包含抖动部分 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR1[31:4]，CCR1[3:0]位被复位

20.3.16 TIMx 捕获/比较寄存器 1（TIMx_CCR1） (x=3/4/5/18)

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR1[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR1	RW/RO	20’h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR1[19:4]，CCR1[3:0]包含抖动部分 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR1[19:4]，CCR1[3:0]位被复位

20.3.17 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x=2）

Address offset:0x38

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR2[15:0]															
RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO	RW/RO

Bit	Name	R/W	Reset Value	Function
31:0	CCR2	RW/RO	32'h0	捕获/比较通道 2 的值（Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 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 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR2[31:4]，CCR2[3:0]包含抖动部分 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。TIMx_CCR2 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR2[31:4]，CCR2[3:0]位被复位

20.3.18 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x=3/4/5/18）

Address offset:0x38

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR2[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR2	RW/RO	20'h0	捕获/比较通道 2 的值（Capture/Compare 2 value) 若 CC2 通道配置为输出: CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 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 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR2[19:4]，CCR2[3:0]包含抖动部分 若 CC2 通道配置为输入: CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。TIMx_CCR2 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR2[19:4]，CCR2[3:0]位被复位

20.3.19 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x=2）

Address offset:0x3C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:0	CCR3	RW/RO	32'h0	捕获/比较通道 3 的值（Capture/Compare 3 value) 若 CC3 通道配置为输出: CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 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 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR3[31:4]，CCR3[3:0]包含抖动部分

				若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。TIMx_CCR3 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR3[31:4]，CCR3[3:0]位被复位

20.3.20 TIMx 捕获/比较寄存器 3（TIMx_CCR3） (x=3/4/5/18)

Address offset:0x3C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR3[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR3	RW/RO	20’h0	捕获/比较通道 3 的值（Capture/Compare 3 value) 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 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 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持比较值。 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR3[19:4]，CCR3[3:0]包含抖动部分 若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。TIMx_CCR3 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR3[19:4]，CCR3[3:0]位被复位

20.3.21 TIMx 捕获/比较寄存器 4（TIMx_CCR4） (x=2)

Address offset:0x40

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:0	CCR4	RW/RO	32'h0	捕获/比较通道 4 的值（Capture/Compare 4 value) 若 CC4 通道配置为输出: CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 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 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 无抖动模式（DITHEN=0）: 该寄存器保持比较值。 抖动模式（DITHEN=1）: 该寄存器保持整数部分 CCR4[31:4]，CCR4[3:0]包含抖动部分 若 CC4 通道配置为输入: CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。TIMx_CCR4 寄存器只读 无抖动模式（DITHEN=0）: 该寄存器保存捕获值 抖动模式（DITHEN=1）: 该寄存器保存捕获值到 CCR4[31:4]，CCR4[3:0]位被复位

20.3.22 TIMx 捕获/比较寄存器 4（TIMx_CCR4） (x=3/4/5/18)

Address offset:0x40

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR4[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR4	RW/RO	20'h0	捕获/比较通道 4 的值（Capture/Compare 4 value) 若 CC4 通道配置为输出: CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 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 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 无抖动模式（DITHEN=0）：该寄存器保持比较值。 抖动模式（DITHEN=1）：

				该寄存器保持整数部分 CCR4[19:4]，CCR4[3:0]包含抖动部分 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。TIMx_CCR4 寄存器只读 无抖动模式（DITHEN=0）：该寄存器保存捕获值 抖动模式（DITHEN=1）：该寄存器保存捕获值到 CCR4[19:4]，CCR4[3:0]位被复位

20.3.23 TIMx 编码器控制寄存器（TIMx_ECR）（x=2/3/4/5/18）

Address offset:0x58

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					PWPRSC[2:0]			PW[7:0]							
					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
Res								IPOS		FIDX	Res		IDIR		IE
								RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	-
26:24	PWPRSC	RW	3'h0	脉冲宽度预分频（Pulse width prescaler） 该位设置脉冲生成器的时钟预分频，公式如下： t_{PWG}= $2^{(PWPRSC[2:0])}$ x tCK_INT
23:16	PW	RW	8'h0	脉冲宽度（Pulse width） 该位定义了脉冲持续时间，公式如下： t_{PW}=PW[7:0] x t_{PWG}
15:8	保留	-	-	-
7:6	IPOS	RW	2'h0	索引定位（Index positioning） 在正交编码器模式中（SMS[3:0] = 0001, 0010, 0011, 1110, 1111），该位用于指示 AB 输入在哪种配置下索引事件会复位计数器 00：AB=00 时，索引会复位计数器 01：AB=01 时，索引会复位计数器 10：AB=10 时，索引会复位计数器 11：AB=11 时，索引会复位计数器 在定向时钟模式或时钟+方向模式中(SMS[3:0] = 1010, 1011, 1100, 1101),该位段用于指示哪种电平索引会复位计数器。在定向时钟模式中，应用于两路的时钟输入。 x0：当时钟为 0 时，索引复位计数器 x1：当时钟为 1 时，索引复位计数器 注：IPOS[1]位无效
5	FIDX	RW	0	第一次索引（First index） 该位指示是否只考虑第一次索引 0：索引一直有效 1：仅第一次索引可以复位计数器
4:3	保留	-	-	-
1	IDIR	RW	0	索引方向（Index direction） 该位指示索引事件在哪个计数方向时复位计数器 00：无论哪个计数方向，索引均可以复位计数器 01：仅在向上计数时，索引可以复位计数器

				10：仅在向下计数时，索引可以复位计数器 11：保留
0	IE	RW	0	索引使能（Index enable） 该位指示索引是否复位计数器 0：索引关闭 1：索引使能

20.3.24 TIMx 输入选择寄存器（TIMx_TISEL）（x=2/3/4/5/18）

Address offset:0x5C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				TI4SEL[3:0]				Res				TI3SEL[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				TI2SEL[3:0]				Res				TI1SEL[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:24	TI4SEL	RW	4’h0	TI4 输入选择（tim_ti4[15:0] input） 0000：TIMx_CH4 0001：tim_ti4_in1 0010：tim_ti4_in2 其他：保留
23:20	保留	-	-	-
19:16	TI3SEL	RW	4’h0	TI3 输入选择（tim_ti3[15:0] input） 0000：TIMx_CH3 0001：tim_ti3_in1 其他：保留
15:12	保留	-	-	-
11:8	TI2SEL	RW	4’h0	TI2 输入选择（tim_ti2[15:0] input） 0000：TIMx_CH2 0001：tim_ti2_in1 0010：tim_ti2_in2 0011：tim_ti2_in3 0100：tim_ti2_in4 其他：保留
7:4	保留	-	-	-
3:0	TI1SEL	RW	4’h0	TI1 输入选择（tim_ti1[15:0] input） 0000：TIMx_CH1 0001：tim_ti1_in1 0010：tim_ti1_in2 0011：tim_ti1_in3 0100：tim_ti1_in4 其他：保留

20.3.25TIMx 备用功能选项寄存器 1（TIMx_AF1）（x=2/3/4/5/18）

Address offset:0x60

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:14	ETRSEL	RW	4’h0	外部触发源选择（etr_in source selection） 该位段用于选择 ETR 输入源 0000：TIMx_ETR 0001：tim_etr1 0010：tim_ etr2 0011：tim_ etr 3 0100：tim_ etr 4 0101：tim_ etr 5 0110：tim_ etr 6 0111：tim_ etr 7 1000：tim_ etr 8 1001：tim_ etr 9 1010：tim_ etr 10 1011：tim_ etr 11 1100：tim_ etr 12 1101：tim_ etr 13 1110：tim_ etr 14 1111：tim_ etr 15
13:0	保留	-	-	-

20.3.26TIMx 备用功能选项寄存器 2（TIMx_AF2）（x=2/3/4/5/18）

Address offset:0x64

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	-
18:16	OCRSEL	RW	3’h0	OCREF 复位源选择（ocref_clr source selection） 该位段用于选择 ocref_clr 输入源

				0000：ocref_clr0 0001：ocref_clr1 0010：ocref_clr2 0011：ocref_clr3 其它：保留
15:0	保留	-	-	-

20.3.27 TIMx DMA 控制寄存器（TIMx_DCR）（x=2/3/4/5/18）

Address offset:0x3DC

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12:8	DBL	RW	5'h0	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 TIMx_DMAR 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 .... 11001：26 次传输 例：我们考虑这样的传输：DBL=7 字节，DBA=TIMx_CR1 - 如果 DBL=7 字节，DBA=TIMx_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址）+ DBA +（DMA 索引)，其中 DMA 索引 = DBL 其中(TIMx_CR1 的地址）+ DBA 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(TIMx_CR1 的地址）+ DBA 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。
7:5	保留	-	-	-
4:0	DBA	RW	5'h0	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 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR，

				…….

20.3.28 TIMx 连续模式的 DMA 地址（TIMx_DMAR）（x=2/3/4/5/18）

Address offset:0x3E0

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[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**
DMAB[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DMAB	RW	0	DMA 连续传送寄存器（DMA register for burst accesses) 对 TIMx_DMAR 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: TIMx_CR1 地址 +（DBA + DMA 索引)x4 其中: “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址; “DBA”是 TIMx_DCR 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 TIMx_DCR 寄存器中定义的 DBL。

21. 通用定时器（TIM9/TIM12）

21.1 简介

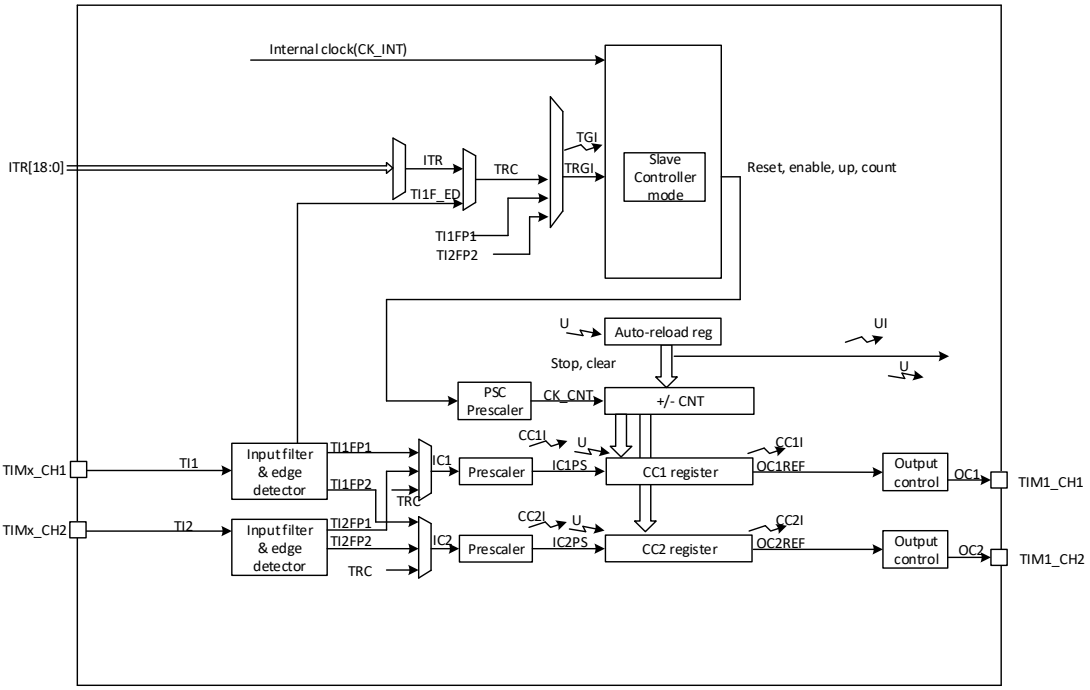
通用定时器(TIM9 和 TIM12)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。通用定时器 TIM9 和 TIM12 是完全独立的，它们不共享任何资源。它们可以同步操作。

21.1.1 TIM9 和 TIM12 主要特征

TIM9 和 TIM12 定时器的功能包括：

- 16 位向上自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 多达 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
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 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)
  - 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

21.1.2 模块框图



21.2 TIM9 和 TIM12 功能描述

21.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC)
- 自动装载寄存器（TIMx_ARR)

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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(上溢或者下溢)并当 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下面几张图给出了在预分频器运行时，更改计数器参数的例子。

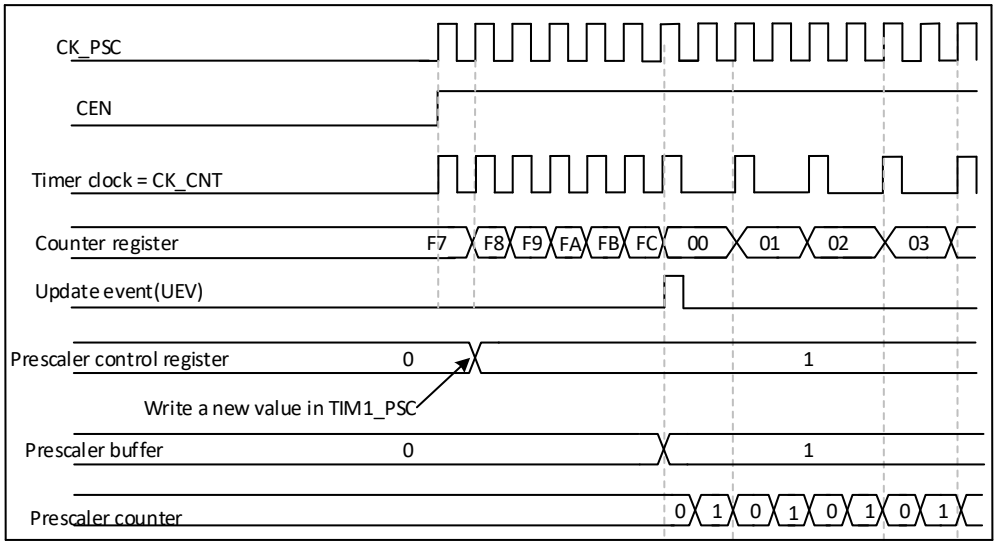


图 21-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

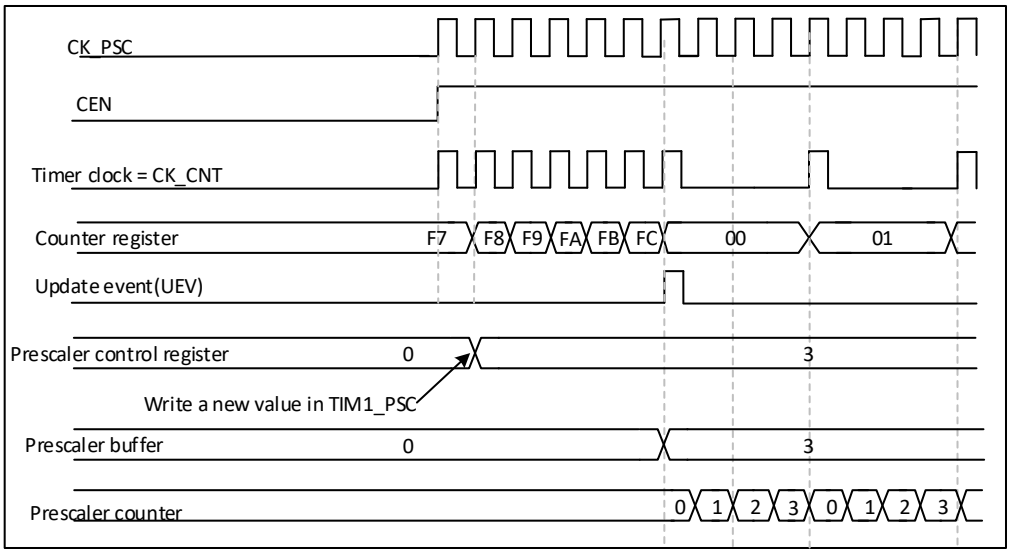


图 21-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

21.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

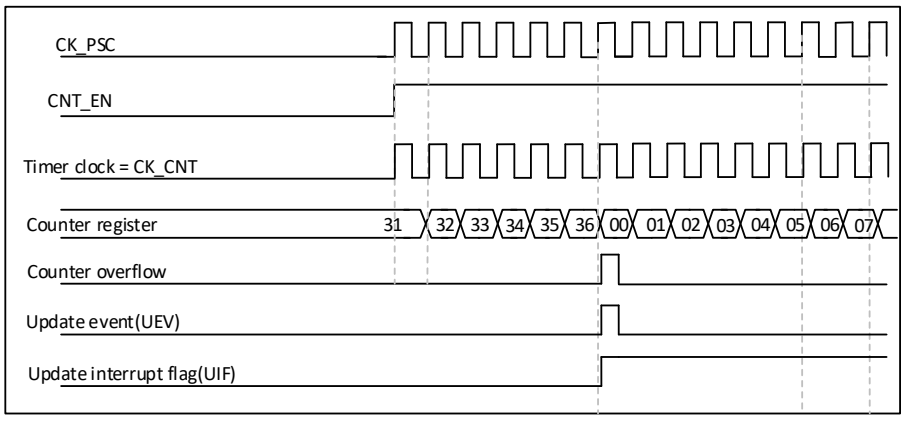


图 21-3 计数器时序图，内部时钟分频因子为 1

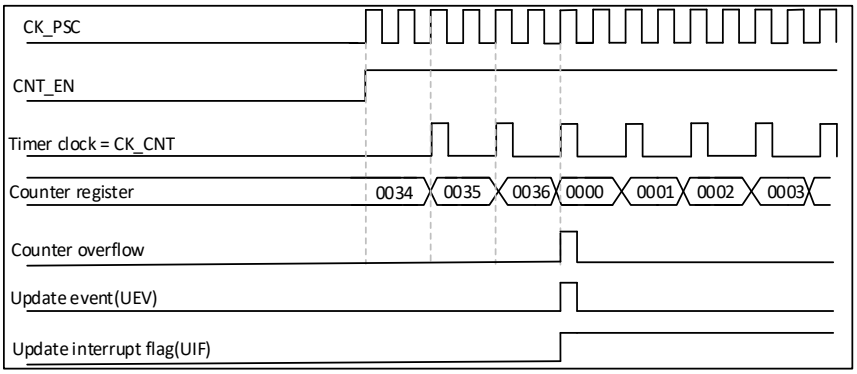


图 21-4 计数器时序图，内部时钟分频因子为 2

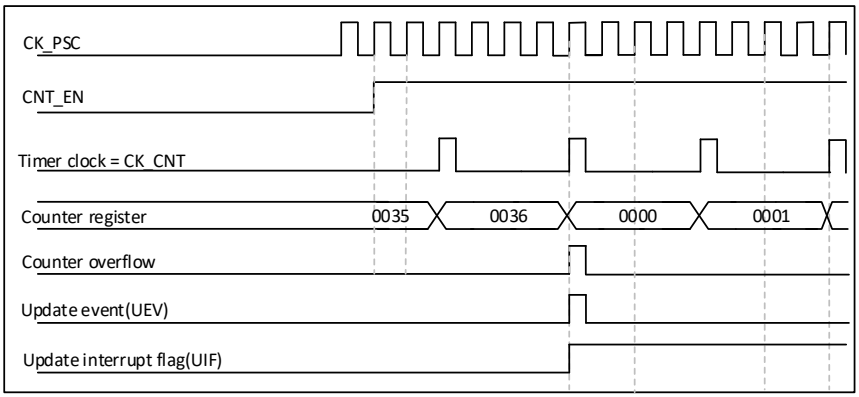


图 21-5 计数器时序图，内部时钟分频因子为 4

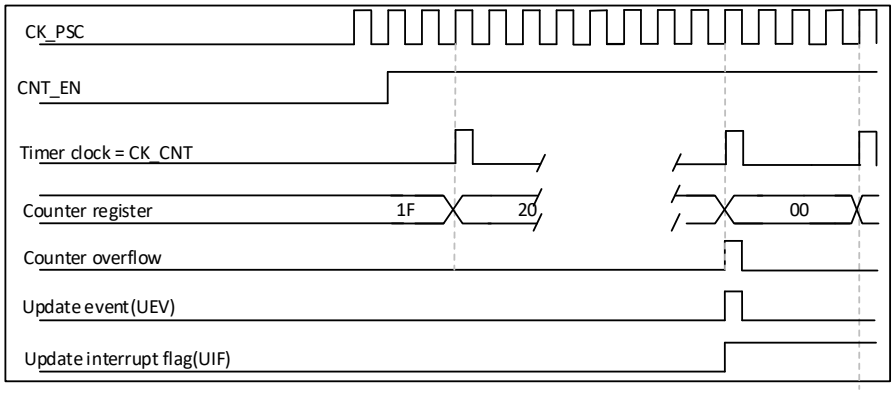


图 21-6 计数器时序图，内部时钟分频因子为 N

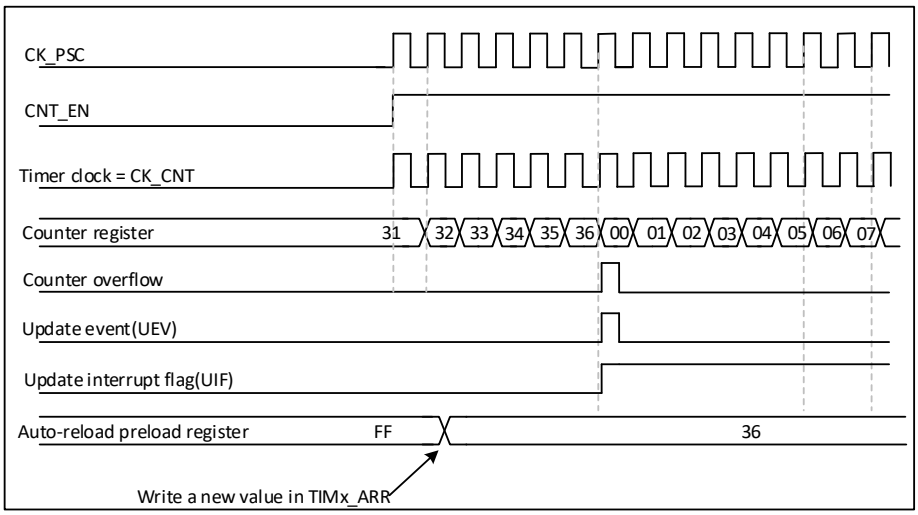


图 21-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

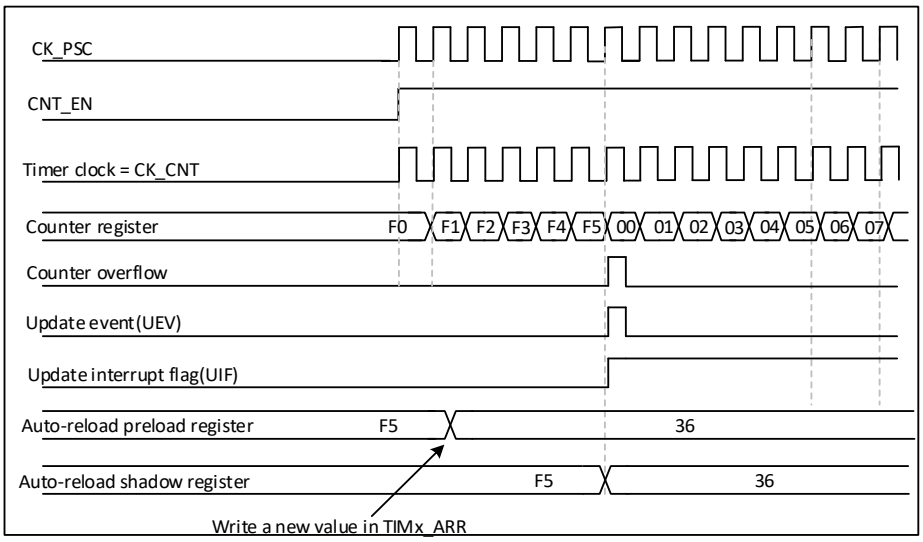


图 21-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

21.2.3 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：外部输入引脚（TIx）
- 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。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 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。

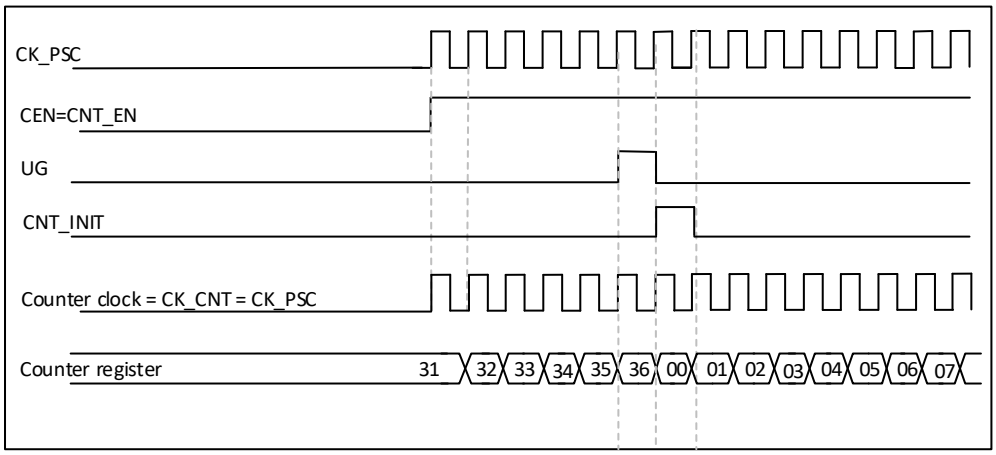


图 21-9 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=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。

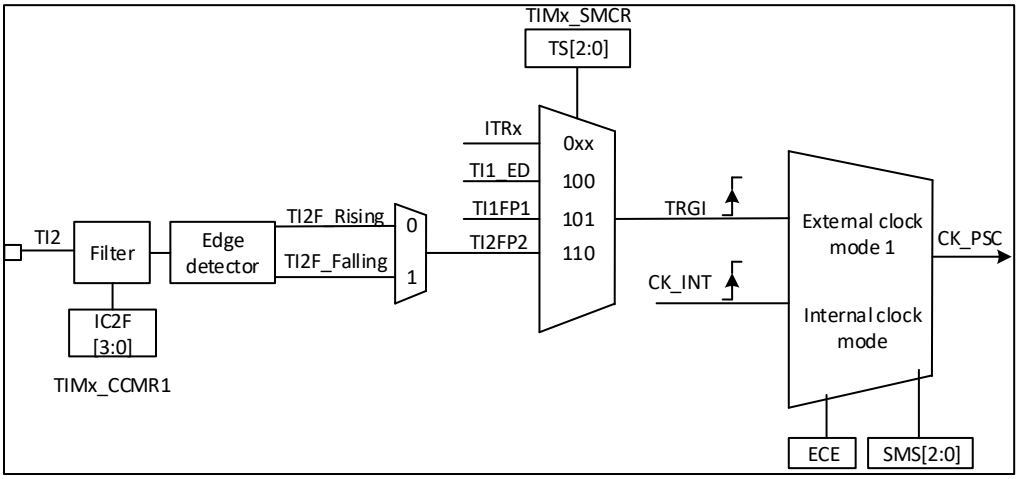


图 21-10 TI2 外部时钟连接例子

例如，要配置计数器在 TI2 输入端的上升沿向上计数，使用下列步骤：

1.配置 TIMx_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；

2.配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)；

3.配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性；

4.配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；

5.配置 TIMx_SMCR 寄存器中的 TS=00110，选定 TI2 作为触发输入源；

6.设置 TIMx_CR1 寄存器的 CEN=1，启动计数器。

注：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 TIF 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 输入端的同步电路。

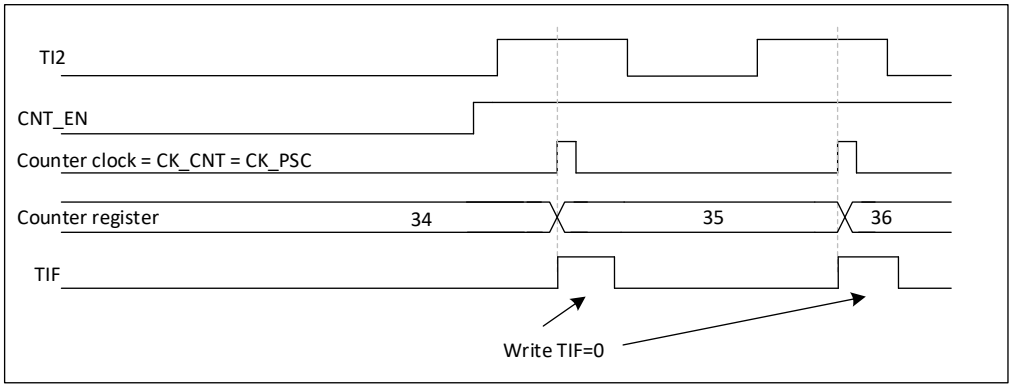


图 21-11 外部时钟模式 1 下的控制电路

21.2.4 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)。

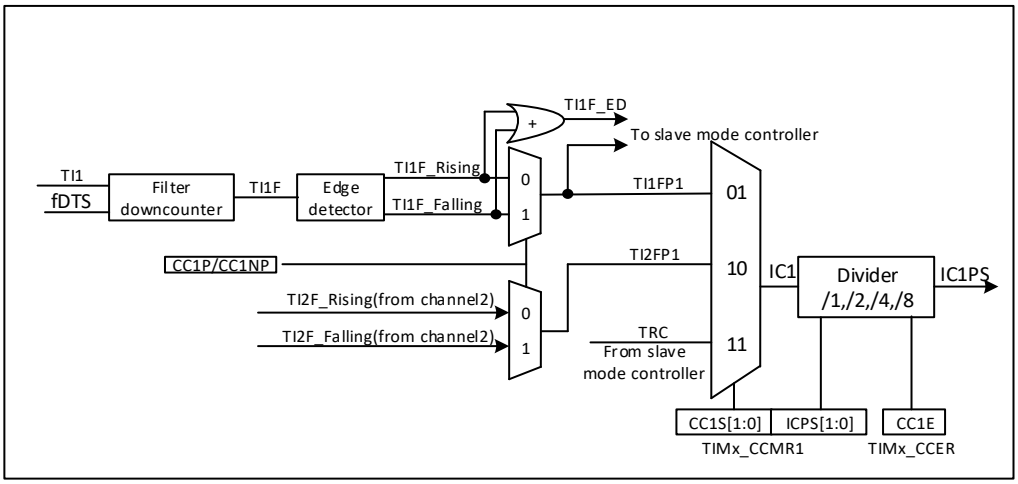


图 21-12 捕获/比较通道(如：通道 1 输入部分)

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。

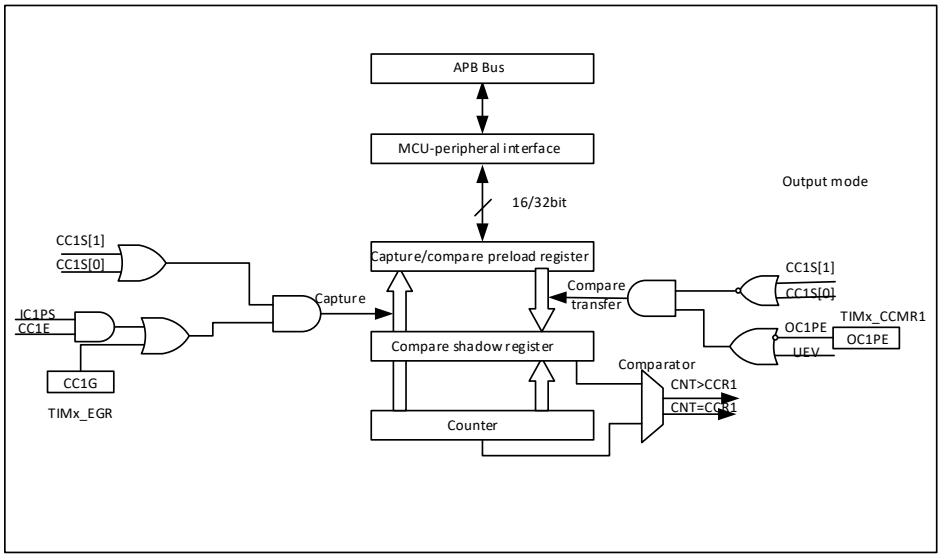


图 21-13 捕获/比较通道 1 的主电路

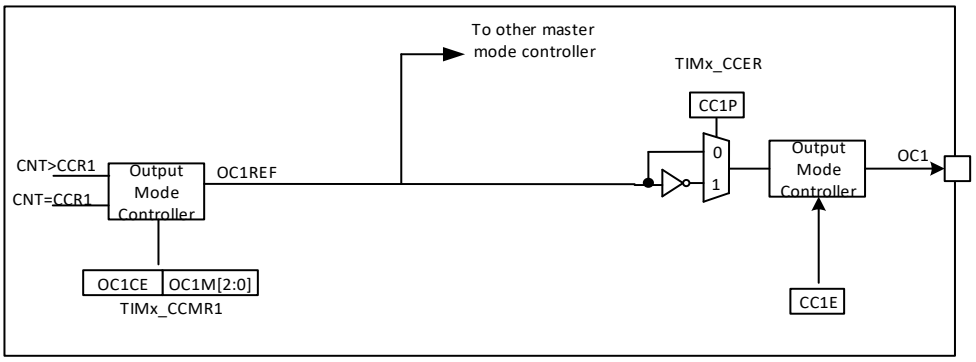


图 21-14 捕获/比较通道的输出部分

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。

21.2.5 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(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 ‘00’，通道就被配置为输入，并且 TIMx_CCR1 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 TIMx_CCMRx 寄存器中的 ICxF 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 fDTS 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 TIMx_CCMR1 寄存器中写入 IC1F=0011。

- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 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 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx_DIER 寄存器中的 CC1IE 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 TIMx_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断请求。

21.2.6 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，其余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，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

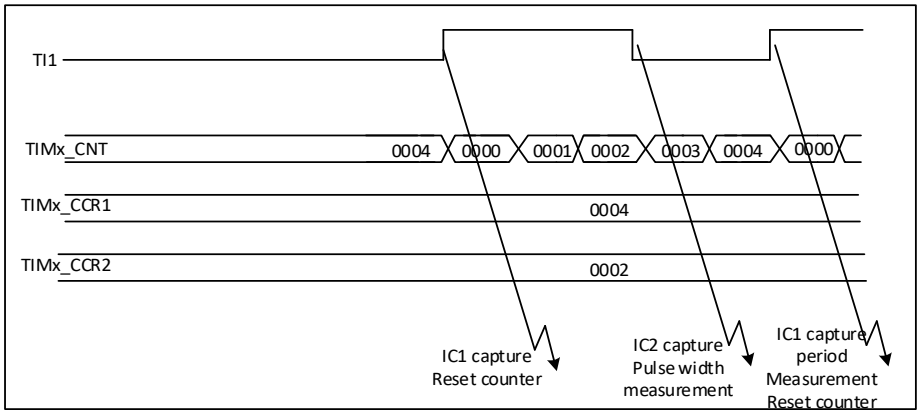


图 21-15 PWM 输入模式时序

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 TIMx_CH1/TIMx_CH2 信号。

21.2.7 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=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 TIMx_CCMRx 寄存器中相应的 OCxM=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 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 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请求。这将会在下面的输出比较模式一节中介绍。

21.2.8 输出比较模式

此项功能是用来控制一个输出波形，或者表明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(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 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(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。

可以通过配置 TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 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 TIMx_ARR 和 TIMx_CCRx 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 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011

— 置 OCxPE = 0 禁用预装载寄存器

— 置 CCxP = 0 选择极性为高电平有效

— 置 CCxE = 1 使能输出

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 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(OCxPE='0'，否则 TIMx_CCRx 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。

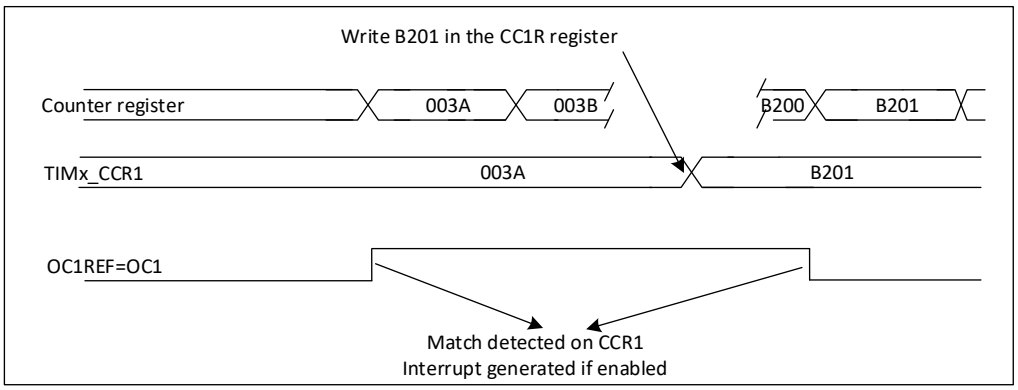


图 21-16 输出比较模式，翻转 OC1

21.2.9 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 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 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位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，用户必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 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(TIMx_CCER 中)CCxE 控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

PWM 边沿对齐模式

- 向上计数配置

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

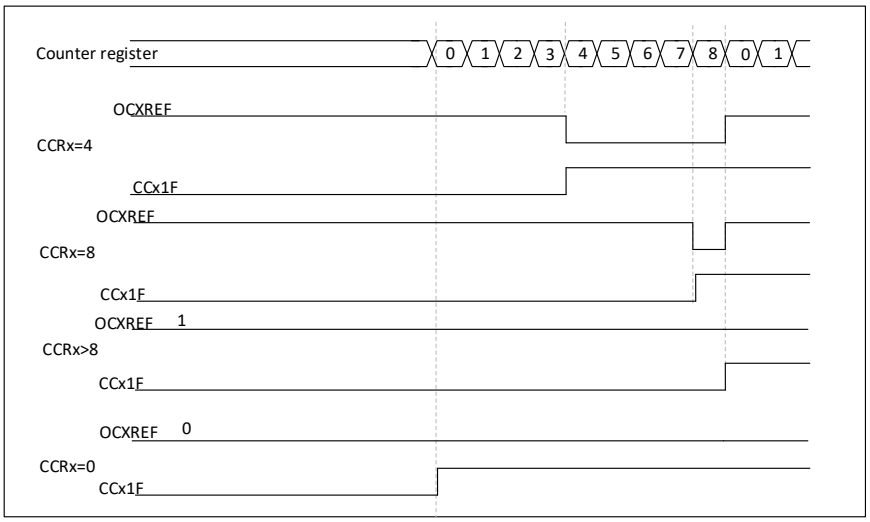


图 21-17 边沿对齐的 PWM 波形(ARR=8)

21.2.10 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 TIMx_CR1 寄存器中的 OPM 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 < CCRx ≤ ARR（特别地, 0 < CCRx)，

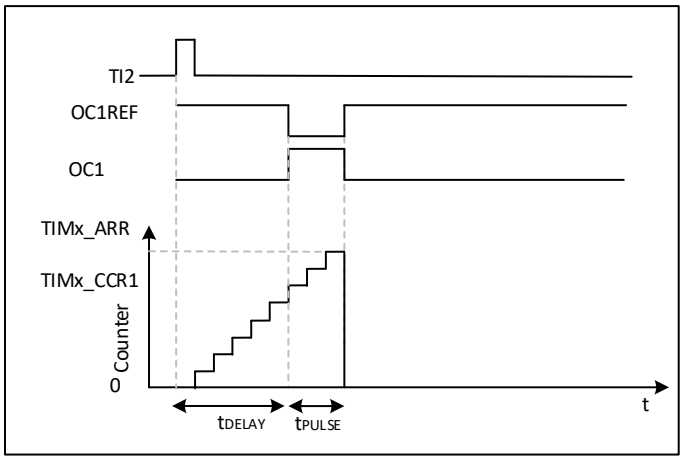


图 21-18 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 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 TIMx_CCR1 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(TIMx_ARR - TIMx_CCR1)。
- 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 0 到 1 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 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，CC1P=0。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=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。

21.2.10.1 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 CEN 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 TIMx_CCMRx 寄存器中的 OCxFE 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。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

21.2.11 TIMx 定时器和外部触发的同步

TIMx 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、触发模式。

21.2.11.1 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 TIMx_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=00101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=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(TIMx_SR 寄存器中的 TIF 位)被设置，下图显示当自动重装载寄存器 TIMx_ARR=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。

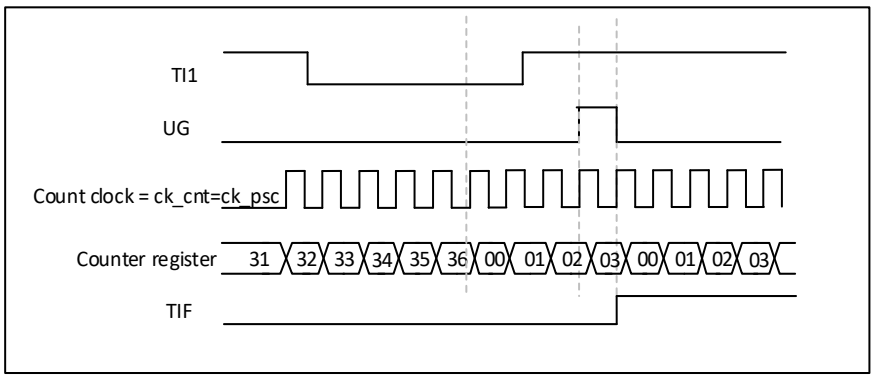


图 21-19 复位模式下的控制电路

21.2.11.2 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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=00101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=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 TIMx_SR 中的 TIF 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。

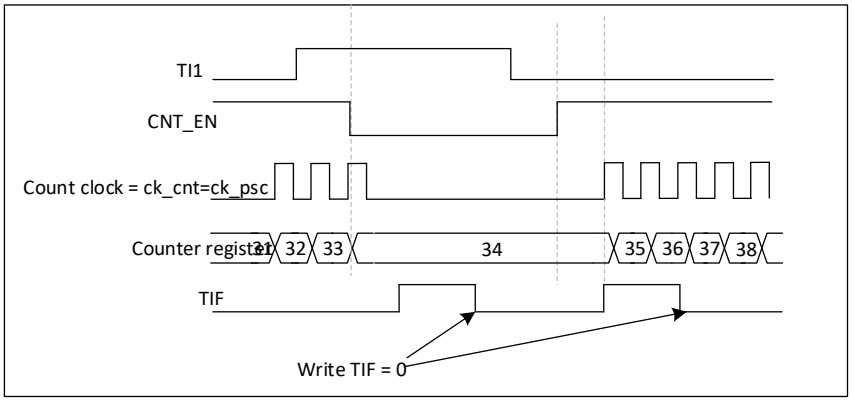


图 21-20 门控模式下的控制电路

21.2.11.3 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 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1

寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。

- 置 TIMx_SMCR 寄存器中 SMS=110,配置定时器为触发模式;置 TIMx_SMCR 寄存器中 TS=00110,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 TIF 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。

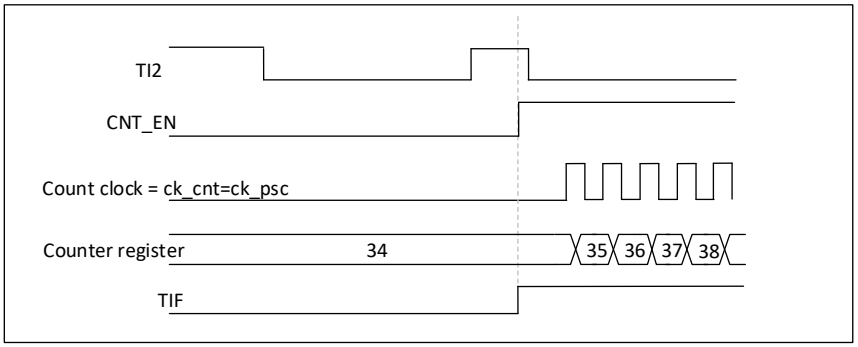


图 21-21 触发器模式下的控制电路

21.2.12定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。详见第 20.2.22 节：定时器同步。

21.2.13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_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

21.3 寄存器描述

21.3.1 TIM9 和 TIM12 控制寄存器 1（TIMx_CR1)

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
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
Res						CKD[1:0]		ARPE	Res			OPM	URS	UDIS	CEN
						RW	RW	RW				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9:8	CKD	RW	2'h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable)

				0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	保留	-	-	-
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

21.3.2 TIM9 和 TIM12 从模式控制寄存器（TIMx_SMCR）

Address offset:0x08

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8:4	TS	RW	5’h0	触发选择（Trigger selection) 这 5 位选择用于同步计数器的触发输入。 00000：内部触发 0(ITR0) 00001：内部触发 1(ITR1)

				00010：内部触发 2(ITR2) 00011：内部触发 3(ITR3) 00100：TI1 边沿检测（TI1F_ED） 00101：滤波后的定时器输入 1（TI1FP1） 00110：滤波后的定时器输入 2（TI1FP2） 01000：内部触发 4(ITR4) 01001：内部触发 5(ITR5) 01010：内部触发 6(ITR6) 01011：内部触发 7(ITR7) 01100：内部触发 8(ITR8) 01101：内部触发 9(ITR9) 01110：内部触发 10(ITR10) 01111：内部触发 11(ITR11) 10000：内部触发 12(ITR12) 10001：内部触发 13(ITR13) 10010：内部触发 14(ITR14) 10011：内部触发 15(ITR15) 10100：内部触发 16(ITR16) 10101：内部触发 17(ITR17) 10110：内部触发 18(ITR18) 其它：保留 更多有关 ITRx 的细节，参见系统级联表 注：这些位只能在未用到(如 SMS=00000)时被改变，以避免在改变时产生错误的边沿检测。
3	保留	-	-	-
2:0	SMS	RW	3'h0	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 CEN=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。

21.3.3 TIM9 和 TIM12 DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									TIE	Res			CC2IE	CC1IE	UIE
									RW				RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	TIE	RW	0	触发中断使能（Trigger interrupt enable) 0：禁止触发中断; 1：使能触发中断。
5:3	保留	-	-	-
2	CC2IE	RW	0	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。
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断; 1：允许更新中断。

21.3.4 TIM9 和 TIM12 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									TIF	Res			CC2IF	CC1IF	UIF
									RC_W0				RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	TIF	RC_W0	0	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。
5:3	保留	-	-	-
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记（Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	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: 0：无匹配发生; 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。

				当 TIMx_CCR1 的内容大于 TIMx_APR 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，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=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 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

21.3.5 TIM9 和 TIM12 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									TG	Res			CC2G	CC1G	UG
									W				W	W	W

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	TG	W	0	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：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5:3	保留	-	-	-
2	CC2G	W	0	产生捕获/比较 2 事件（Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 1 配置为输入：

				当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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)。

21.3.6 TIM9 和 TIM12 捕获/比较模式控制寄存器 1（TIMx_CCMR1)

Address offset:0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

21.3.6.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	OC2CE	RW	0	输出比较 2 清零使能（Output Compare 2 clear enable)
14:12	OC2M	RW	3'h0	输出比较 2 模式（Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能（Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能（Output Compare 2 fast enable)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	输出比较 1 模式（Output Compare 1 mode) 这些位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_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 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 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 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。
2	OC1FE	RW	0	输出比较 1 快速使能（Output Compare 1 fast enable) 该位用于加快输出对触发输入事件的响应。必须在单脉冲模式(TIMx_CR1 寄存器的 OPM 位置 1）中使用，当触发到来时实现脉冲快速输出。 0：根据计数器与 CCR1 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象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。仅能用于 PWM 模式 1 和 PWM 模式 2
1:0	CC1S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

21.3.6.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC2F	RW	4’h0	输入捕获 2 滤波器（Input capture 2 filter)
11:10	IC2PSC	RW	2’h0	输入/捕获 2 预分频器（Input capture 2 prescaler)
9:8	CC2S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F	RW	4'h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。
1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

21.3.7 TIM9 和 TIM12 捕获/比较使能寄存器（TIMx_CCER）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
								RW		RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	CC2NP	RW	0	输入/捕获 2 互补输出极性（Capture/Compare 2 output polarity) 参考 CC1NP 的描述。
6	保留	-	-	-
5	CC2P	RW	0	输入/捕获 2 输出极性（Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能（Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性(Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效; 1：OC1N 低电平有效。
2	保留	-	-	-
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿: TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）; TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿: TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）; TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效(捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。
0	CC1E	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 TIMx_CCR1 寄存器。 0：捕获禁止; 1：捕获使能。 注: 对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表 21-1 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

CCxE 位	OCx 输出状态
0	输出禁止（与定时器断开）OCx=0，OCx_EN=0
1	OCx=OCxREF+Polarity，OCx_EN=1

21.3.8 TIM9 和 TIM12 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CNT	RW	16’h0	计数器的值（Counter value)

21.3.9 TIM9 和 TIM12 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16’h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清’0’或被工作在复位模式的从控制器清’0’。

21.3.10 TIM9 和 TIM12 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	ARR	RW	16’hFFFF	自动重装载的值（Prescaler value)

				自动重装载的值为空时，计数器不工作。

21.3.11 TIM9 和 TIM12 捕获/比较寄存器 1（TIMx_CCR1）

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CCR1	RW/RO	16'h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_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： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读

21.3.12 TIM9 和 TIM12 捕获/比较寄存器 2（TIMx_CCR2）

Address offset:0x38

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CCR2	RW/RO	16'h0	捕获/比较通道 2 的值（Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC2PE 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 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_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： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。TIMx_CCR2 寄存器只读

22. 通用定时器（TIM10/TIM11/TIM13/TIM14/TIM19）

22.1 简介

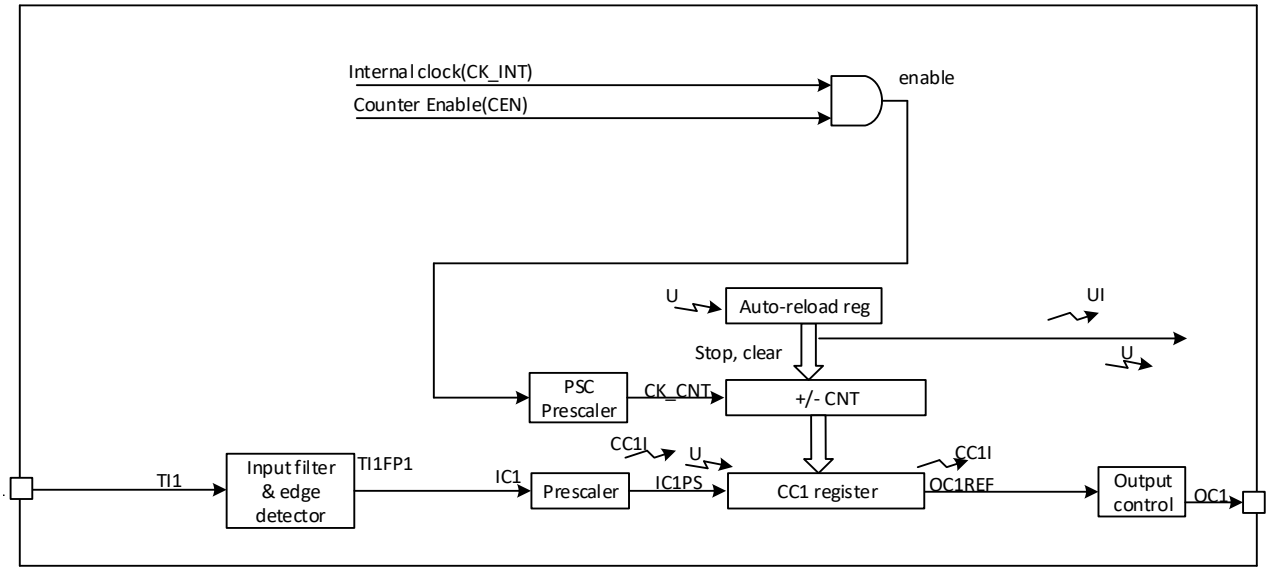
通用定时器(TIM10/TIM11/TIM13/TIM14/TIM19，以下简称 TIMx)是由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作。

22.1.1 TIMx 主要特征

TIMx 定时器的功能包括：

- 16 位向上自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，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(边缘模式)
  - 单脉冲模式输出
- 如下事件发生时产生中断：
  - 计数器向上溢出，计数器初始化(通过软件)
  - 输入捕获
  - 输出比较

22.1.2 模块框图



22.2 TIMx 功能描述

22.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC)
- 自动装载寄存器（TIMx_ARR)

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下面几张图给出了在预分频器运行时，更改计数器参数的例子。

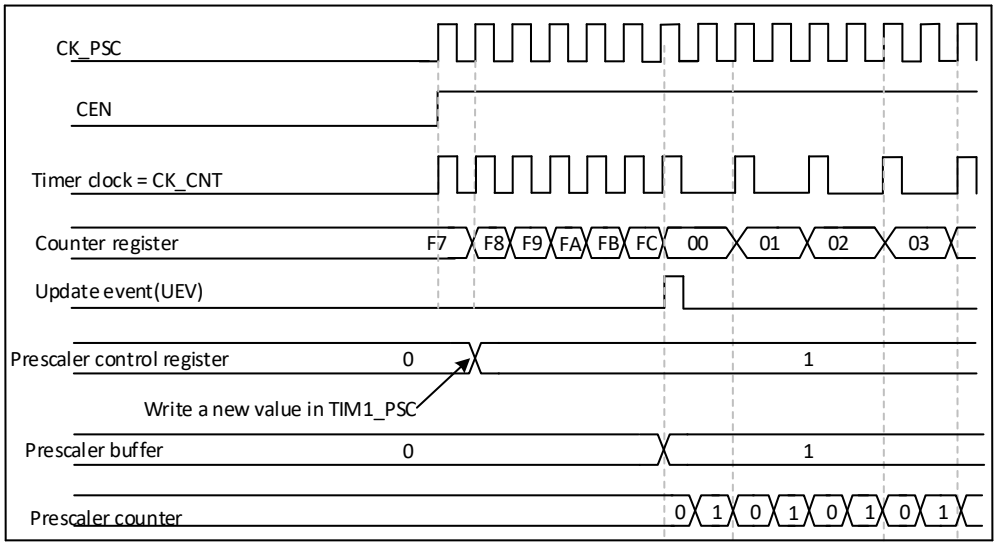


图 22-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

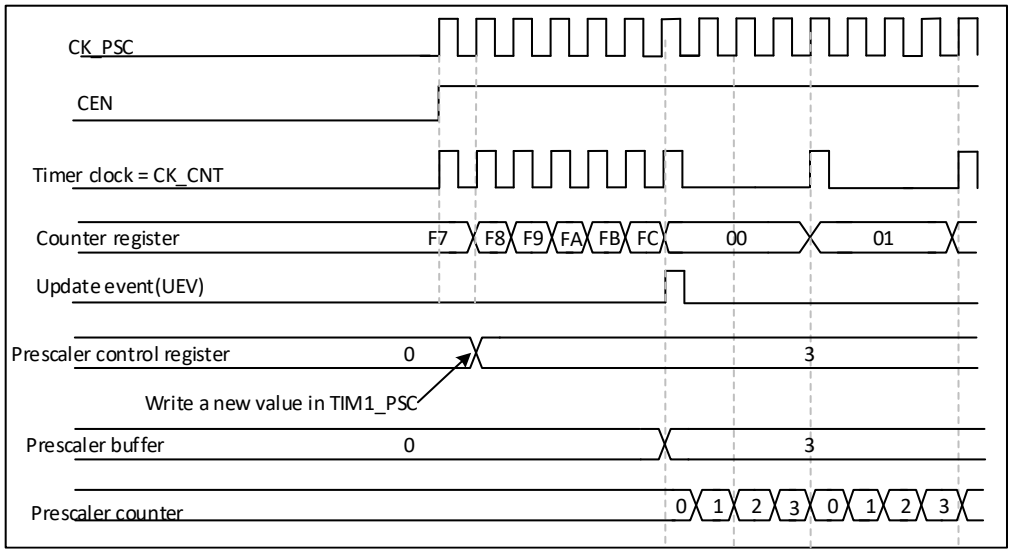


图 22-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

22.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

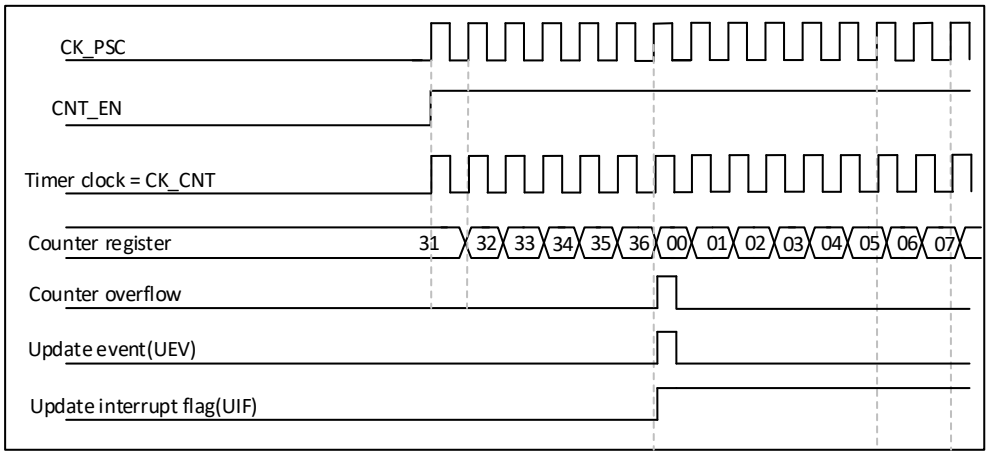


图 22-3 计数器时序图，内部时钟分频因子为 1

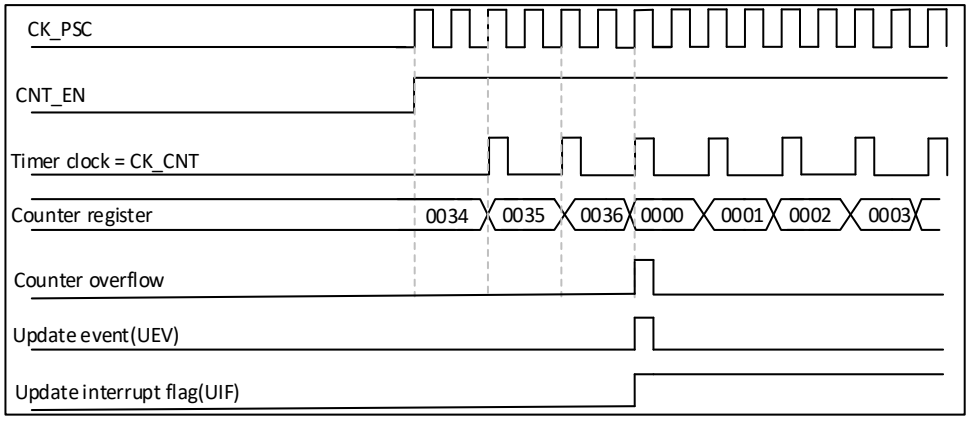


图 22-4 计数器时序图，内部时钟分频因子为 2

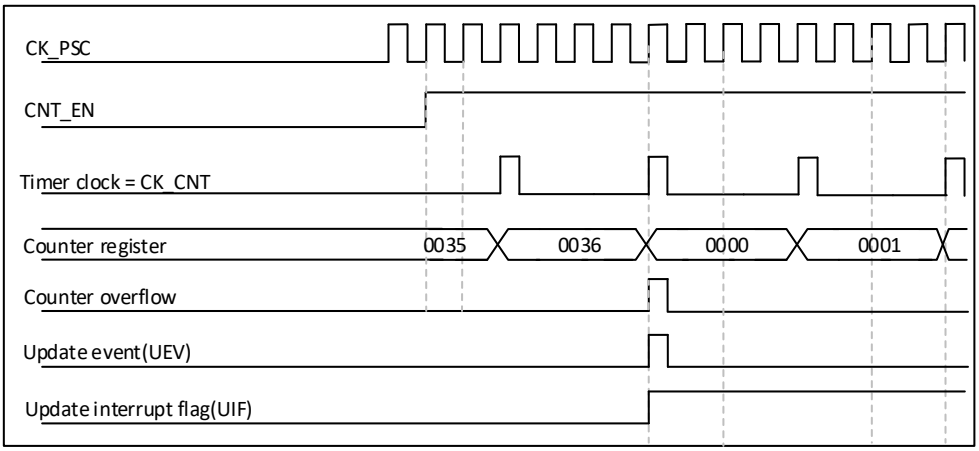


图 22-5 计数器时序图，内部时钟分频因子为 4

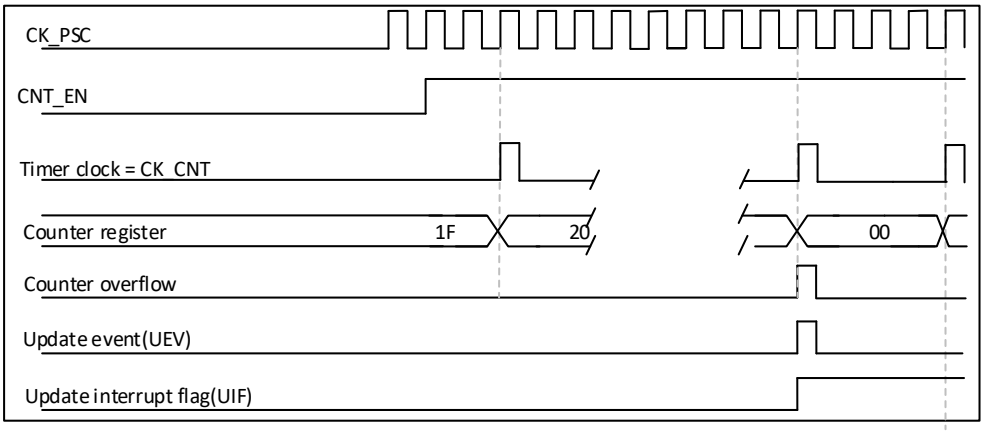


图 22-6 计数器时序图，内部时钟分频因子为 N

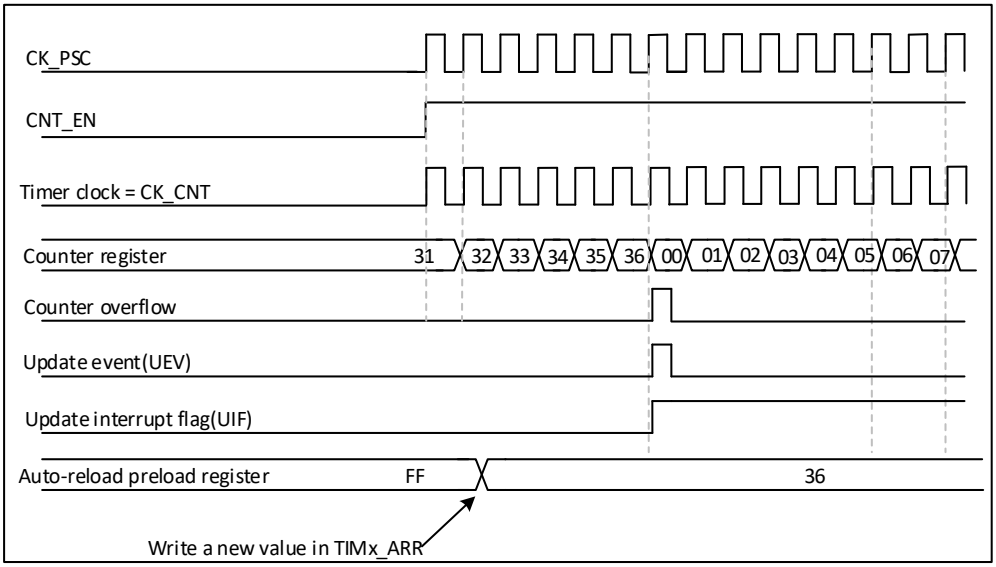


图 22-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

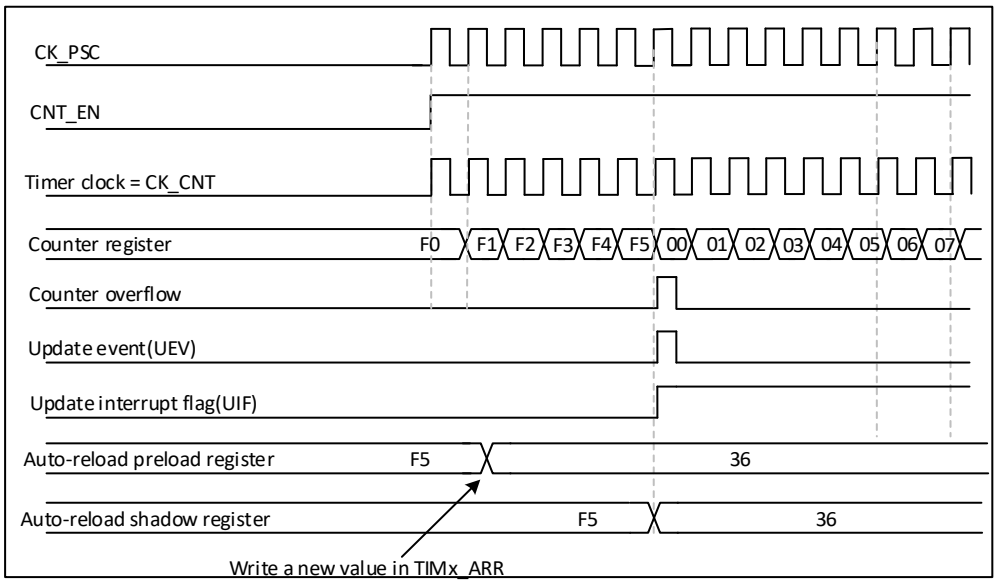


图 22-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

22.2.3 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK_INT)

内部时钟源(CK_INT)

对于 TIM10/TIM11/TIM13/TIM14/TIM19，内部时钟源时默认时钟，则 CEN 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 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。

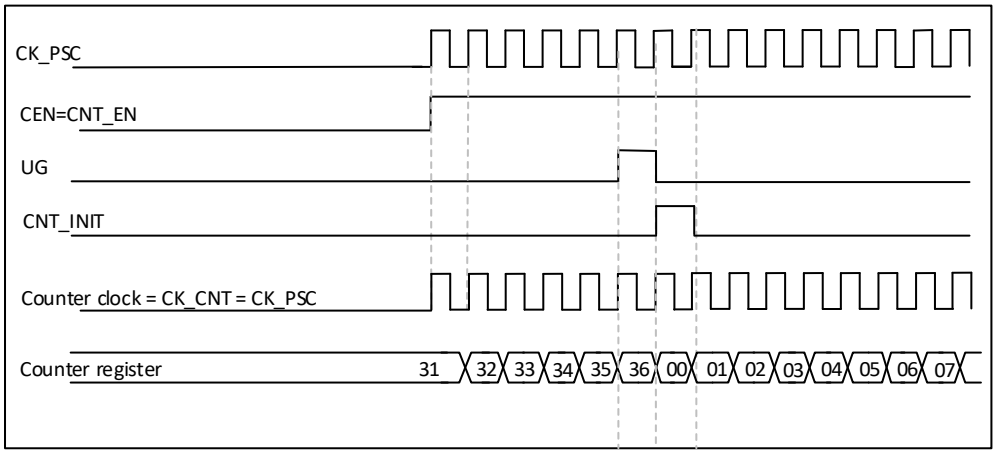


图 22-9 一般模式下的控制电路，内部时钟分频因子为 1

22.2.4 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)，它可以作为捕获控制。该信号通过预分频进入捕获寄存器(ICxPS)。

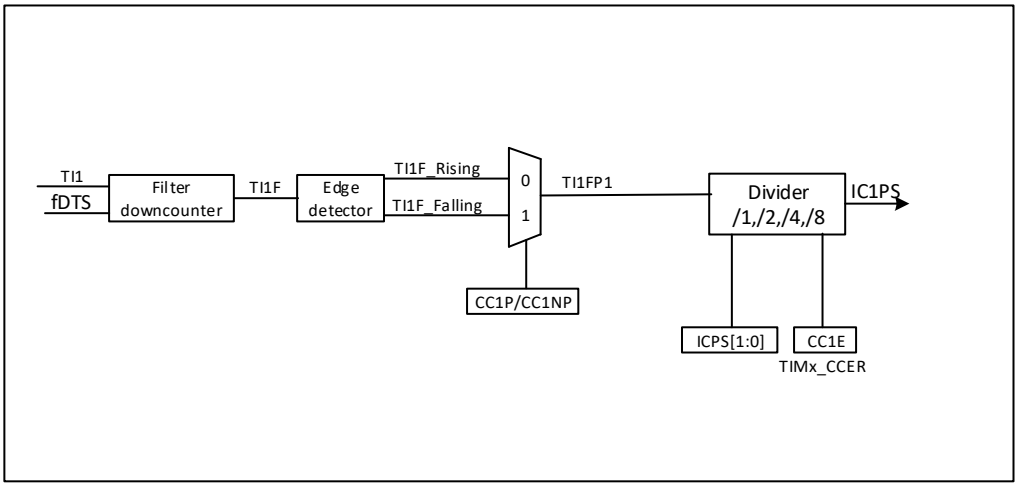


图 22-10 捕获通道

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。

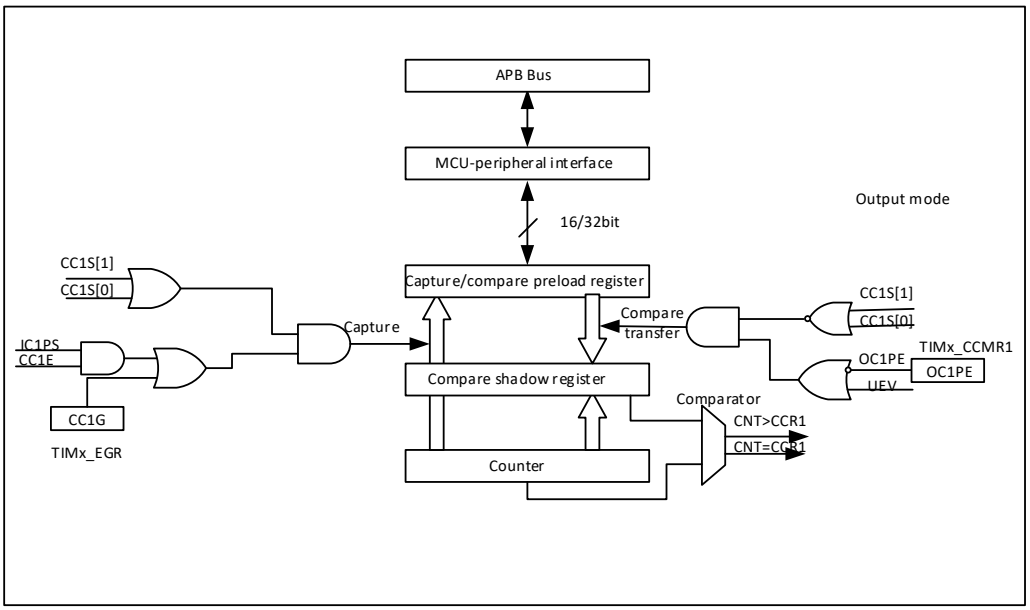


图 22-11 捕获/比较通道 1 的主电路

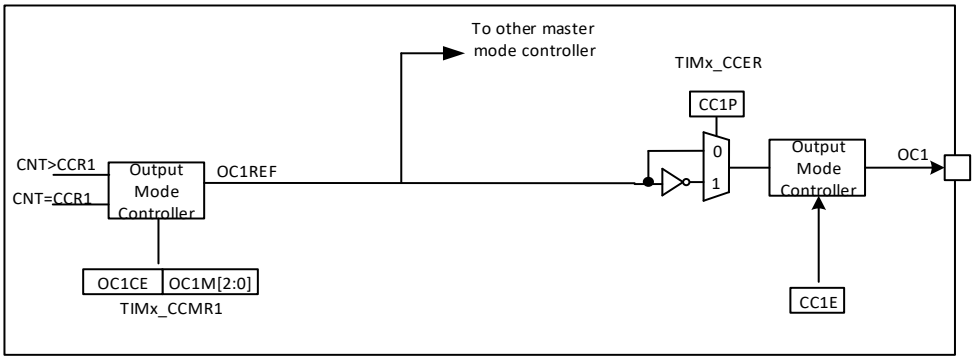


图 22-12 捕获/比较通道的输出部分

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。

22.2.5 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(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 ‘00’，通道就被配置为输入，并且 TIMx_CCR1 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 TIMx_CCMRx 寄存器中的 ICxF 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 fDTS 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 TIMx_CCMR1 寄存器中写入 IC1F=0011。

- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 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 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx_DIER 寄存器中的 CC1IE 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 TIMx_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断请求。

22.2.6 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=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 TIMx_CCMRx 寄存器中相应的 OCxM=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 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 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请求。这将会在下面的输出比较模式一节中介绍。

22.2.7 输出比较模式

此项功能是用来控制一个输出波形，或者表明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(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 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(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。

可以通过配置 TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 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，预分频器)。
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 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 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
— 置 OCxPE = 0 禁用预装载寄存器
— 置 CCxP = 0 选择极性为高电平有效
— 置 CCxE = 1 使能输出

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 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(OCxPE='0'，否则 TIMx_CCRx 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。

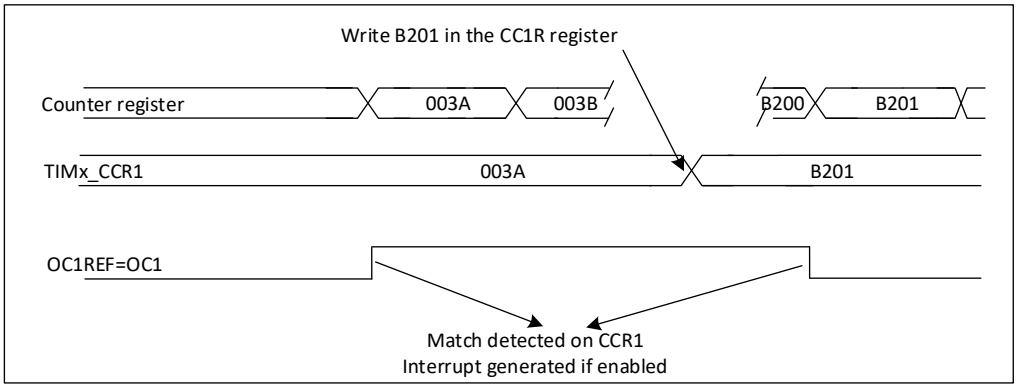


图 22-13 输出比较模式，翻转 OC1

22.2.8 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 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 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，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，用户必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 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(TIMx_CCER 中)CCxE 位的控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

PWM 边沿对齐模式

■ 向上计数配置

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

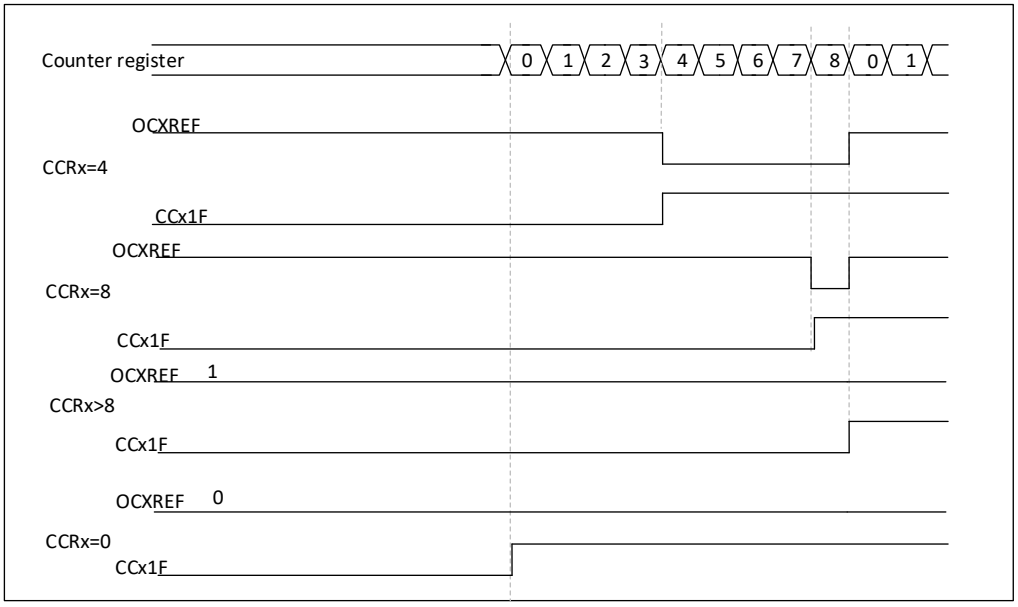


图 22-14 边沿对齐的 PWM 波形(ARR=8)

22.2.9 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。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=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。

22.2.10定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。详见第 20.2.22 节：定时器同步。

22.2.11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_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

22.3 寄存器描述

22.3.1 TIMX 控制寄存器 1（TIMx_CR1）

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
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
Res						CKD[1:0]		ARPE	Res			OPM	URS	UDIS	CEN
						RW	RW	RW				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9:8	CKD	RW	2'h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	保留	-	-	-
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器)

				1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

22.3.2 TIMX DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														CC1IE	UIE
														RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

22.3.3 TIMX 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res							CC1IF	UIF
						RC_W0								RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	CC1OF	RC_W0	0	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 TIMx_CCR1 寄存器时，CC1IF 的状态已经为‘1’。
8:3	保留	-	-	-
1	CC1IF	RC_W0	0	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： 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 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，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=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 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

22.3.4 TIMX 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														CC1G	UG
														W	W

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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)。

22.3.5 TIMX 捕获/比较模式控制寄存器 1（TIMx_CCMR1）

Address offset:0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								OC1CE	OC1M[2:0]			OC1PE	Res	CC1S[1:0]	
								ICIF[3:0]				ICIPSC[1:0]			
								RW	RW	RW	RW	RW	RW	RW	RW

22.3.5.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	输出比较 1 模式（Output Compare 1 mode) 这些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 CC1P、CC1NP 位。 000: 冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_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 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 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 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。
2	保留	-	-	-

1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

22.3.5.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:4	IC1F	RW	4'h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。
1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

22.3.6 TIMX 捕获/比较使能寄存器（TIMx_CCER）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												CC1NP	Res	CC1P	CC1E
												RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3	CC1NP	RW	0	输入/捕获 1 互补输出极性（Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效; 1：OC1N 低电平有效。
2	保留	-	-	-
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿: TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）; TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿: TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）; TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效（捕获、复位模式下触发、外部时钟或触发模式下）; TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。
0	CC1E	RW	0	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 TIMx_CCR1 寄存器。 0：捕获禁止; 1：捕获使能。

表 22-1 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

CCxE 位	OCx 输出状态
0	输出禁止（与定时器断开）OCx=0, OCx_EN=0
1	OCx=OCxREF+Polarity, OCx_EN=1

22.3.7 TIMX 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CNT	RW	16'h0	计数器的值（Counter value)

22.3.8 TIMX 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16'h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

22.3.9 TIMX 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	ARR	RW	16'hFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。

22.3.10 TIMX 捕获/比较寄存器 1（TIMx_CCR1）

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO	RW/ RO

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CCR1	RW/ RO	16'h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_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： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读

23. 通用定时器（TIM15/TIM16/TIM17）

23.1 简介

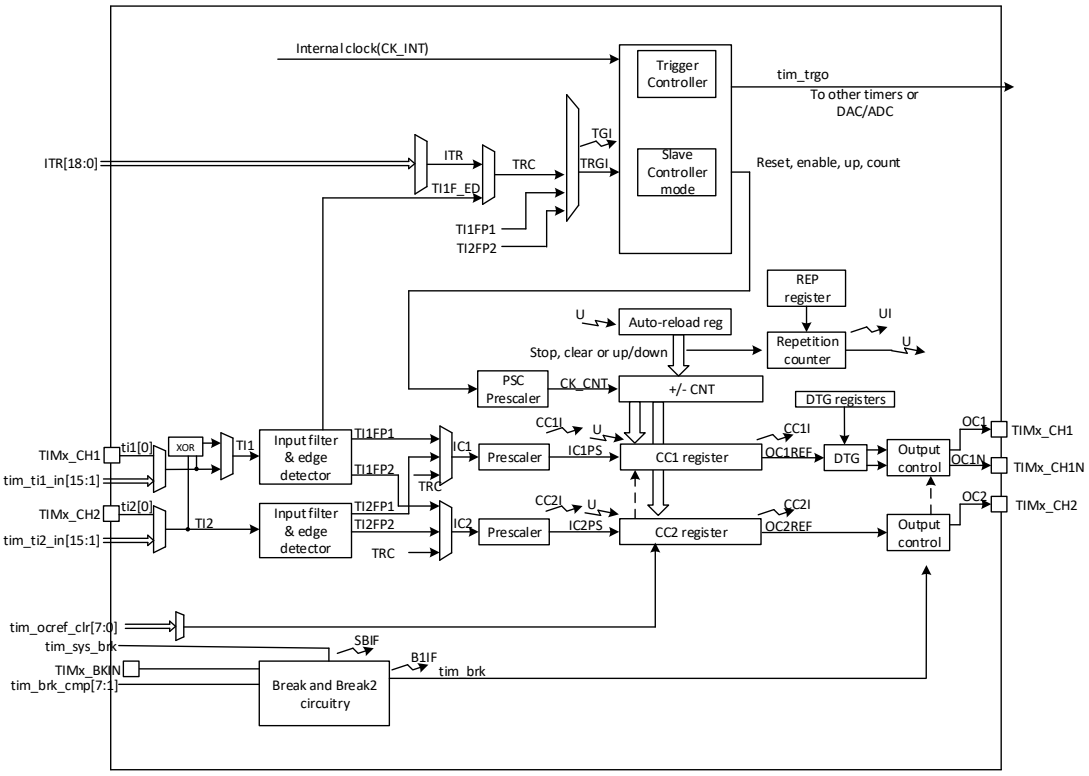
通用控制定时器(TIM15_16_17)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。通用控制定时器(TIM15_16_17)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作。

23.1.1 TIM15 主要特征

TIM15 定时器的功能包括：

- 16 位向上的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 多达 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
- 带有可编程死区时间的互补输出(只有通道 1)
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 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，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

23.1.2 模块框图

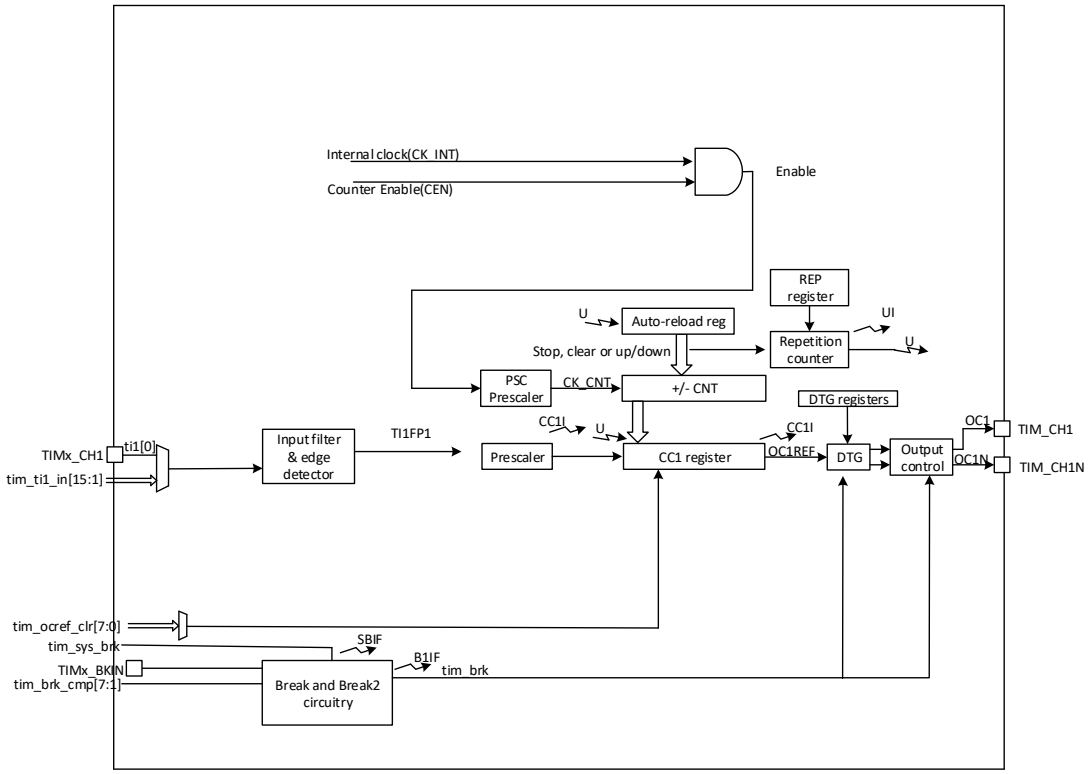


23.1.3 TIM16_17 主要特征

TIM16_17 定时器的功能包括：

- 16 位向上、向下、向上/下的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，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
- 带有可编程死区时间的互补输出
- 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：
  - 计数器向上溢出
  - 输入捕获
  - 输出比较
  - 刹车信号输入

23.1.4 模块框图



23.2 TIM15_16_17 功能描述

23.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC)
- 自动装载寄存器（TIMx_ARR)
- 重复次数寄存器（TIMx_RCR)

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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(上溢或者下溢)并当 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下面几张图给出了在预分频器运行时，更改计数器参数的例子。

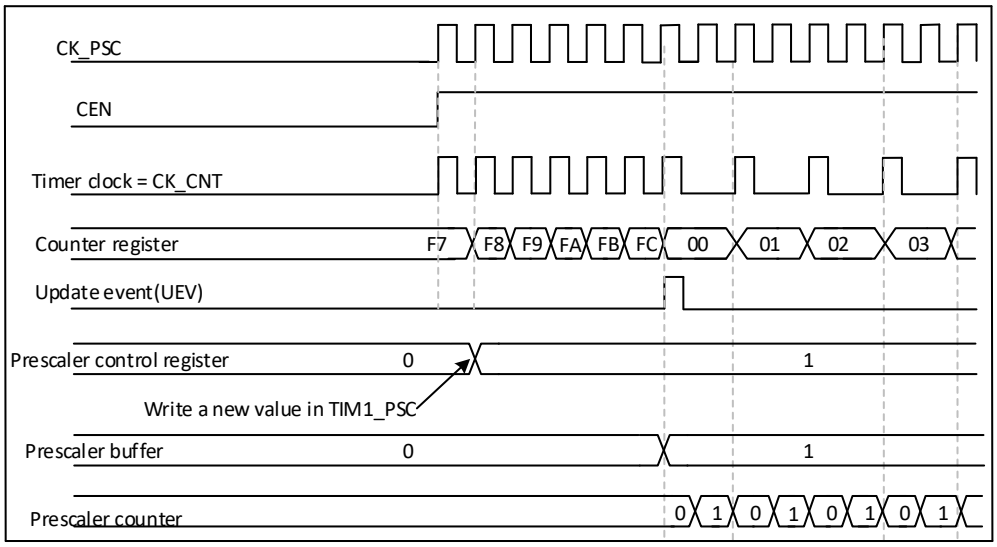


图 23-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

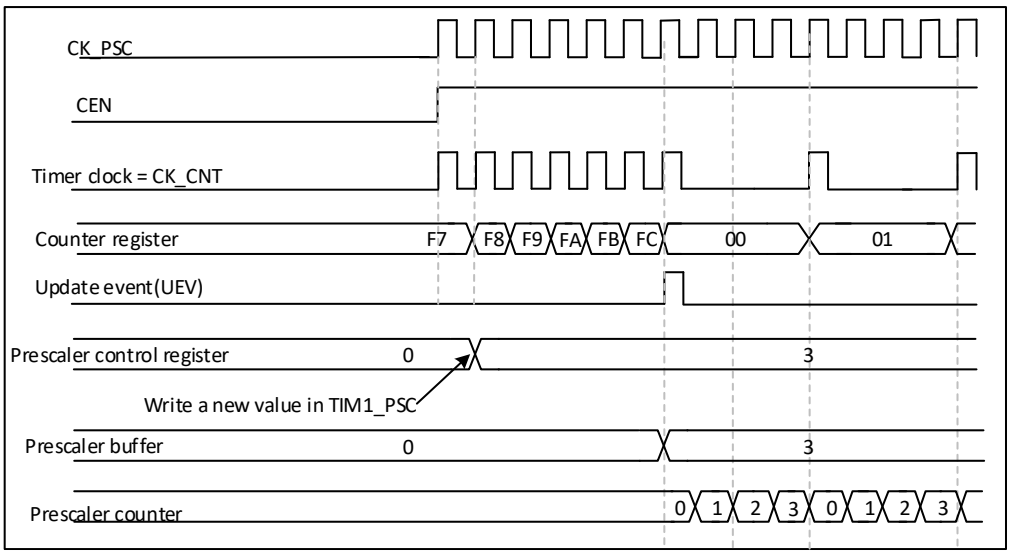


图 23-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

23.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

如果使用了重复计数器，那么更新事件(UEV)需要在上溢次数达到所配置的重复计数寄存器的值加一（即 TIMx_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx_RCR=0），那么每次计数上溢都会产生更新事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断或 DMA 请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 重复计数器被重新加载为 TIMx_RCR 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(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

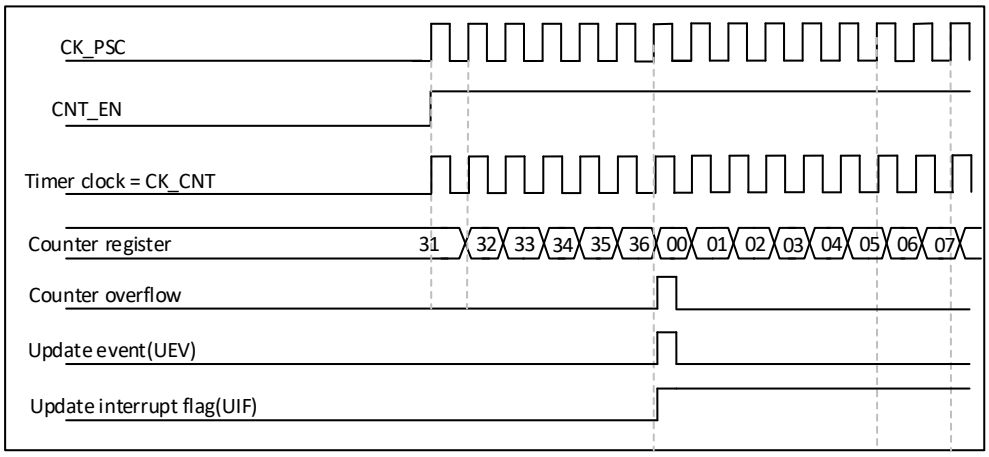


图 23-3 计数器时序图，内部时钟分频因子为 1

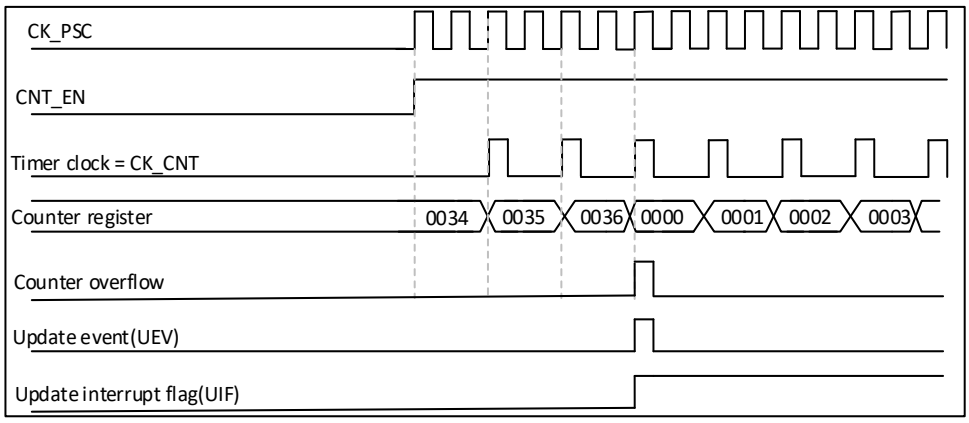


图 23-4 计数器时序图，内部时钟分频因子为 2

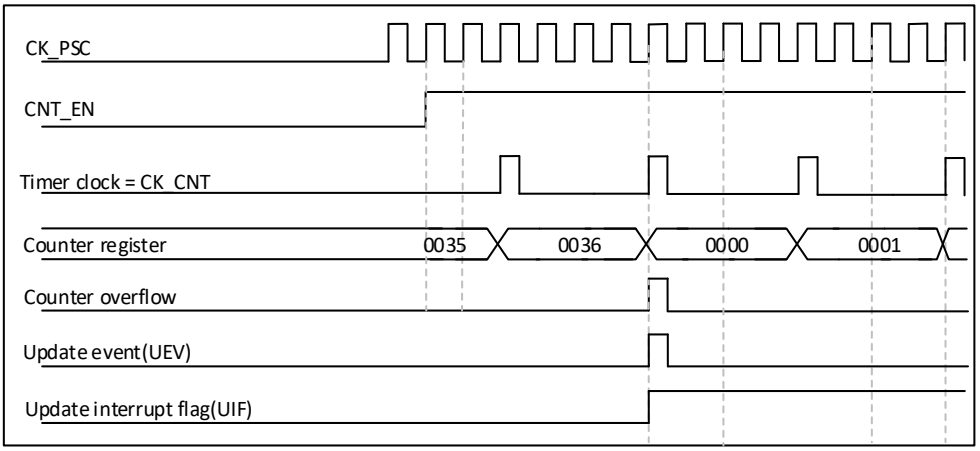


图 23-5 计数器时序图，内部时钟分频因子为 4

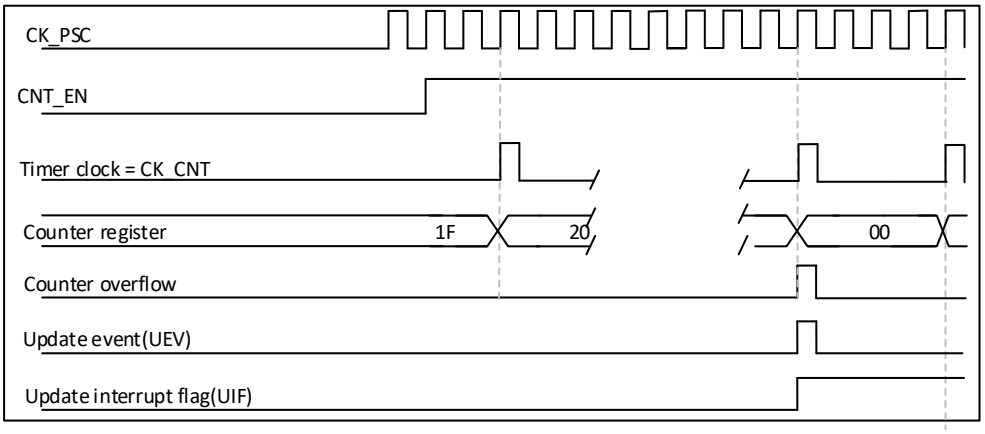


图 23-6 计数器时序图，内部时钟分频因子为 N

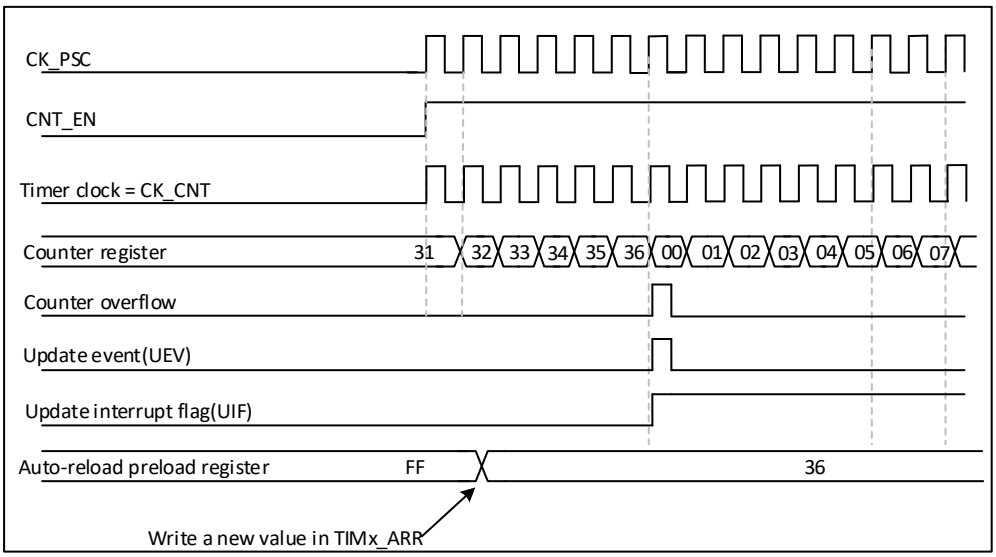


图 23-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

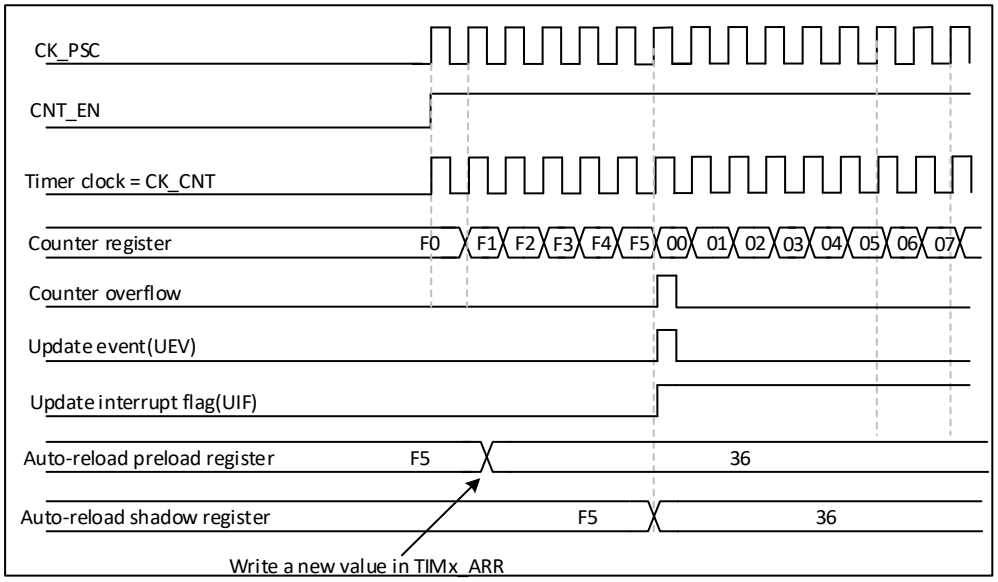


图 23-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

23.2.3 重复计数器

“时基单元”解释了计数器溢出时更新事件(UEV)是如何产生的，事实上它只能在重复计数器计数达到 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+1 个计数溢出时，数据才会从预装载寄存器传输到影子寄存器(TIMx_ARR 自动重载寄存器，TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx)，N 是 TIMx_RCR 重复计数寄存器中的值。

重复计数器在计数器溢出时递减。

重复计数器是自动重载的，重复速率由 TIMx_RCR 寄存器的值定义。当更新事件由软件产生(通过设置 TIMx_EGR 中的 UG 位)或者通过硬件的从模式控制器产生，无论重复计数器的值是多少，立即发生更新事件，并且 TIMx_RCR 寄存器中的内容被重载入到重复计数器。

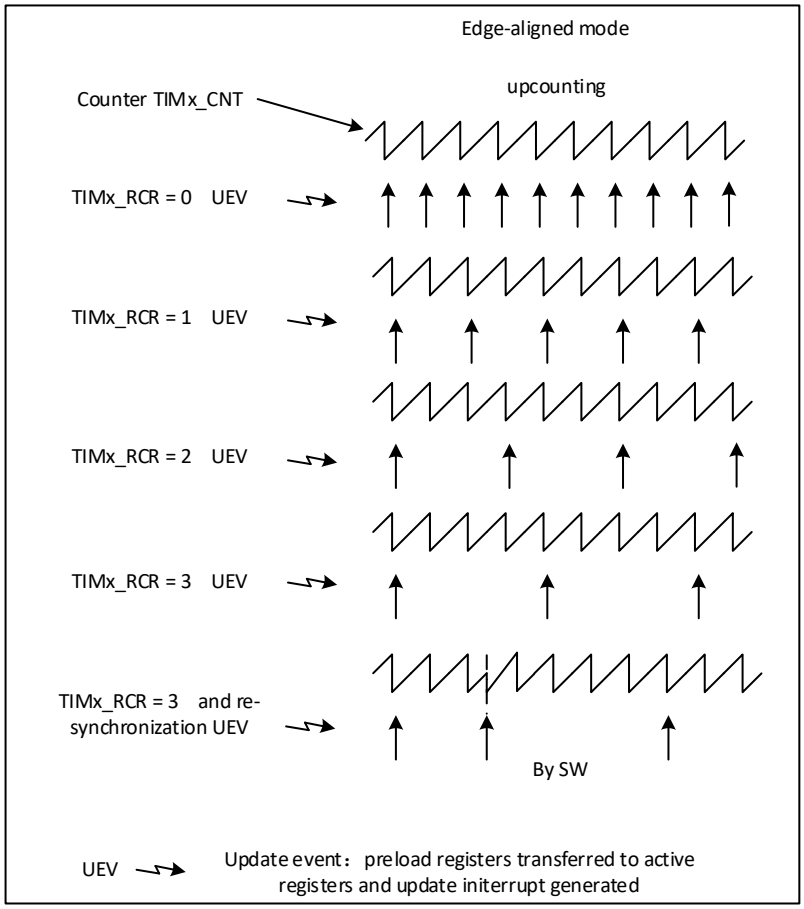


图 23-9 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

23.2.4 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(TI1 和 TI2)
- 内部触发输入(ITRx)(仅 tim15)：使用一个定时器作为另一个定时器的预分频器。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=0000)，则 CEN 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 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。

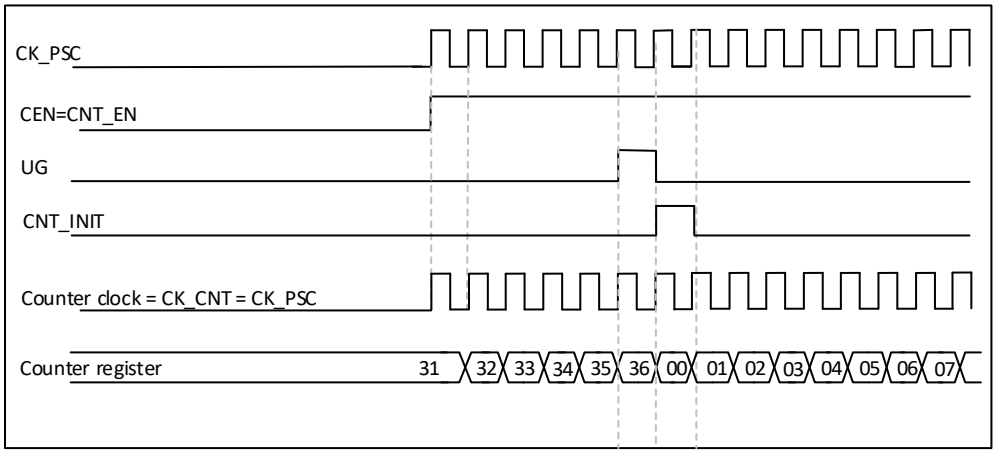


图 23-10 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=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。

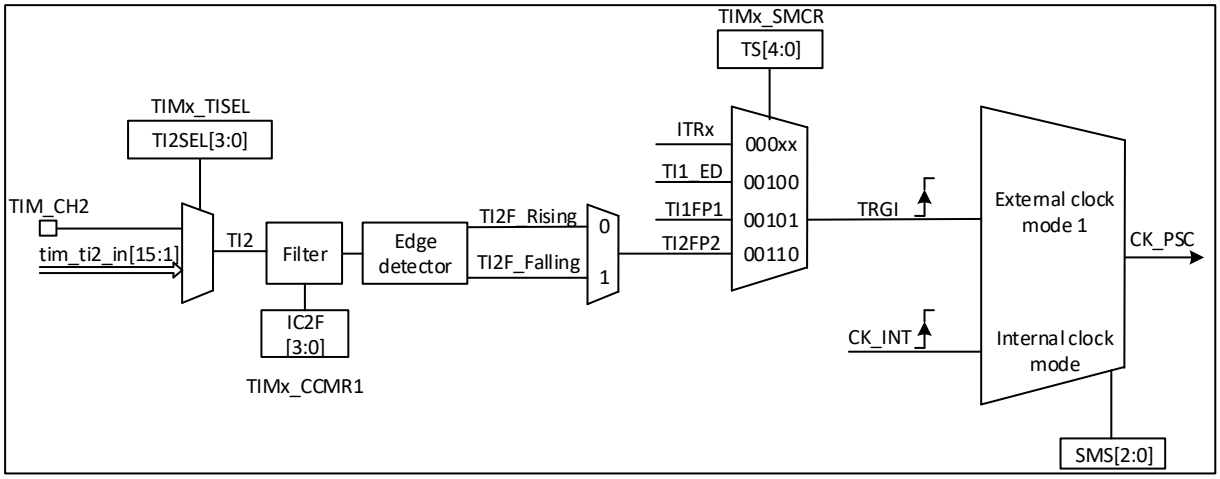


图 23-11 TI2 外部时钟连接例子

例如，要配置计数器在 TI2 输入端的上升沿向上计数，使用下列步骤：

1.配置 TIMx_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；

2.配置 TIMx_CCMR1 寄存器的 IC2F[3:0],选择输入滤波器带宽(如果不需要滤波器,保持 IC2F=0000)；

3.配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性；

4.配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；

5.配置 TIMx_SMCR 寄存器中的 TS=00110，选定 TI2 作为触发输入源；

6.设置 TIMx_CR1 寄存器的 CEN=1，启动计数器。

注：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 TIF 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 输入端的同步电路。

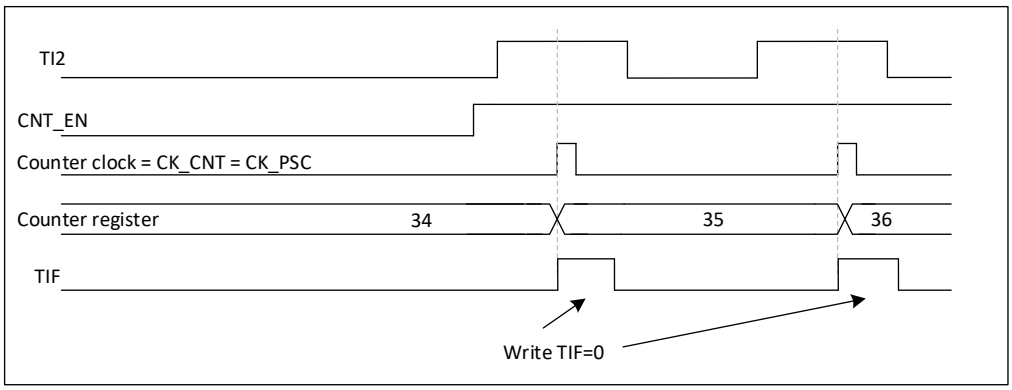


图 23-12 外部时钟模式 1 下的控制电路

23.2.5 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)。

下边几张图是一个捕获/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)。

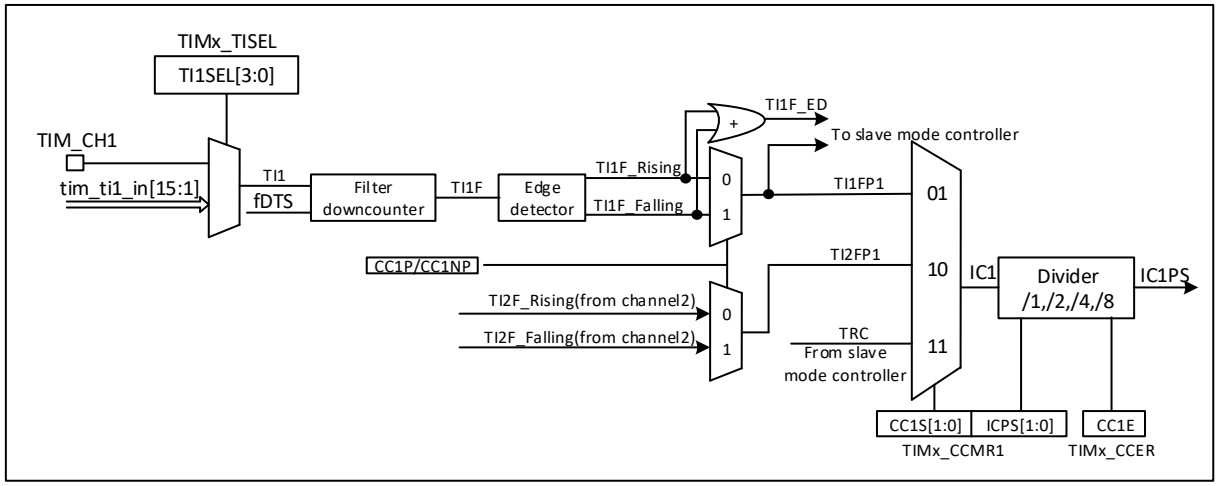


图 23-13 捕获/比较通道(如：通道 1 输入部分)

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。

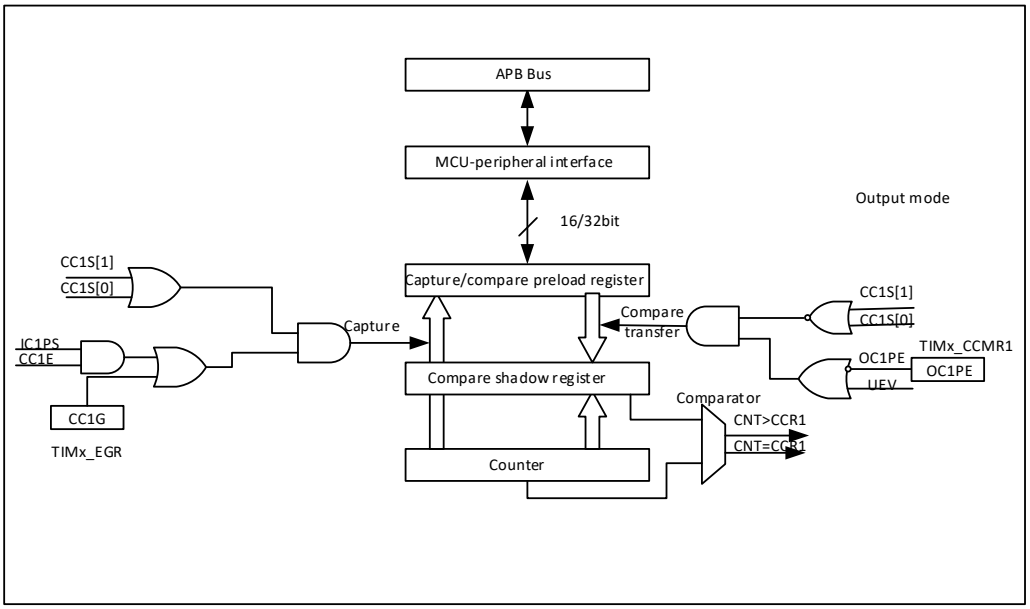


图 23-14 捕获/比较通道 1 的主电路

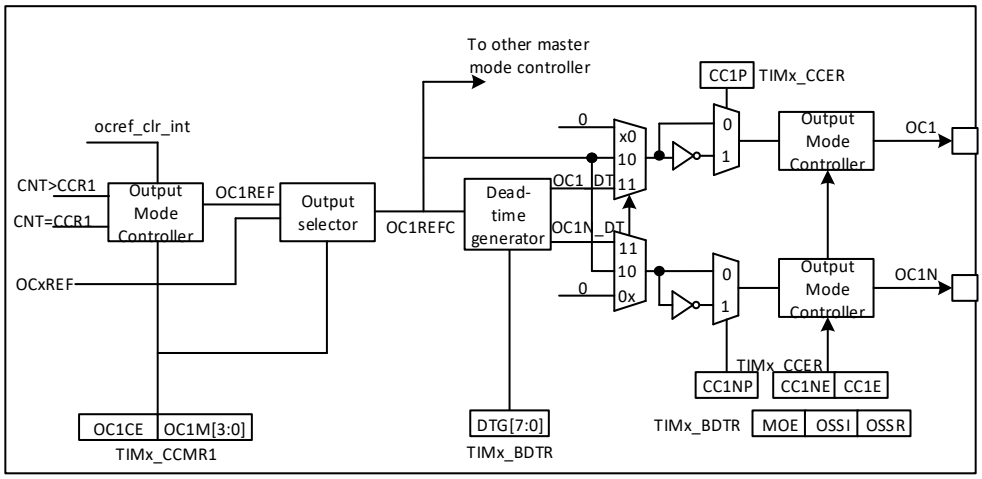


图 23-15 捕获/比较通道的输出部分（通道 1）

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。

23.2.6 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(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1，如果使能中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

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 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCMR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 ‘00’，通道就被配置为输入，并且 TIMx_CCR1 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 TIMx_CCMRx 寄存器中的 ICxF 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 fDTS 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 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 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 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，也可通过设置 TIMx_DIER 寄存器中的 CC1DE 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 TIMx_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

23.2.7 PWM 输入模式（仅 tim15）

该模式是输入捕获模式的一个特例，除下列区别外，其余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，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

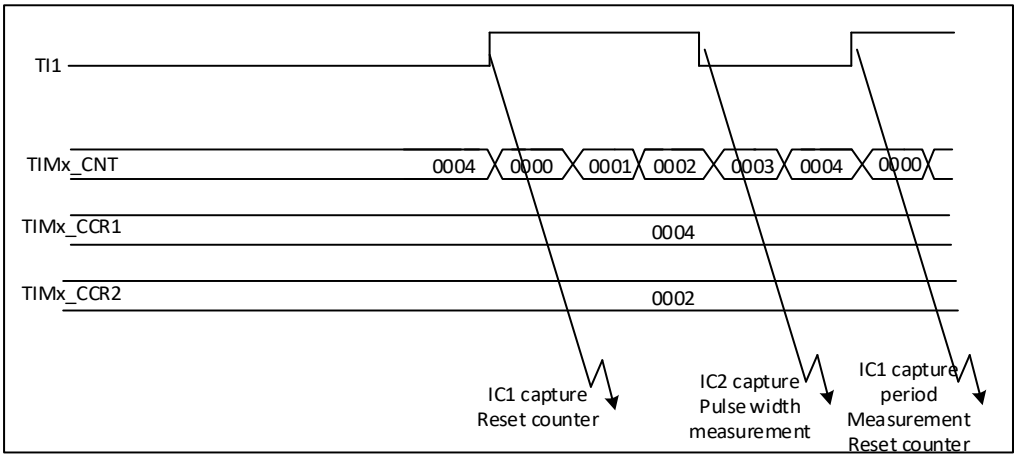


图 23-16 PWM 输入模式时序

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 TIMx_CH1/TIMx_CH2 信号。

23.2.8 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=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 TIMx_CCMRx 寄存器中相应的 OCxM=0101，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 CCxP 极性相反的信号。例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=0100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 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。

23.2.9 输出比较模式

此项功能是用来控制一个输出波形，或者表明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(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 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(OCxM=0000)、被设置成有效电平(OCxM=0001)、被设置成无效电平(OCxM=0010)或进行翻转(OCxM=0011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

可以通过配置 TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 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 TIMx_ARR 和 TIMx_CCRx 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 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=0011
— 置 OCxPE = 0 禁用预装载寄存器
— 置 CCxP = 0 选择极性为高电平有效
— 置 CCxE = 1 使能输出

5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 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(OCxPE='0'，否则 TIMx_CCRx 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。

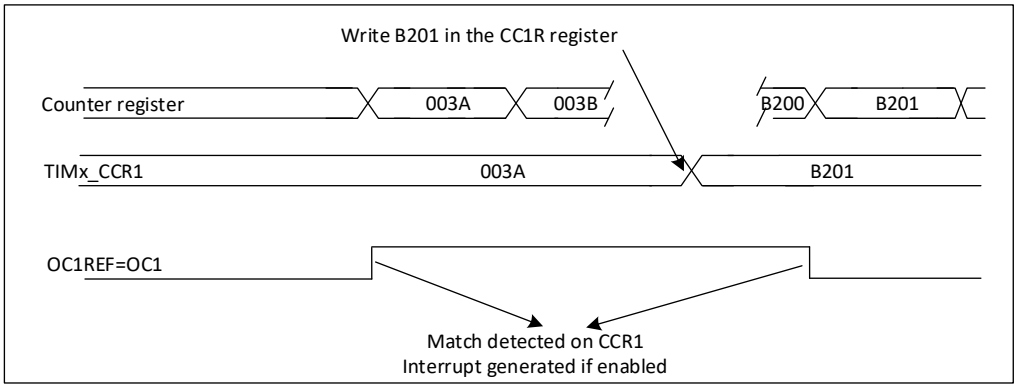


图 23-17 输出比较模式，翻转 OC1

23.2.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'0110'（PWM 模式 1)或'0111'(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 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，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，用户必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 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(TIMx_CCER 和 TIMx_BDTR 寄存器中)CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

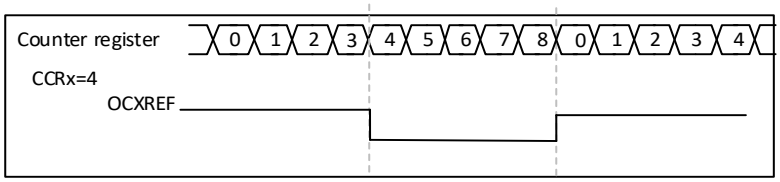


图 23-18 边沿对齐的 PWM 波形(APR=8)

抖动模式

可以使能 TIMx_CR1 寄存器的 DITHEN 位来打开抖动模式提高 PWM 模式的有效分辨率。可以应用于 CCR（提高占空比分辨率）和 ARR（提高 PWM 频率的分辨率）

操作原理是使实际的 CCR（或 ARR）的值在 16 个连续的 PWM 周期内进行轻微的变化（增加或不增加一个定时器周期），如何变化可以通过预定义设置。计算平均占空比时就可以实现分辨率提高 16 倍。

下图为 4 个连续 PWM 周期抖动原理

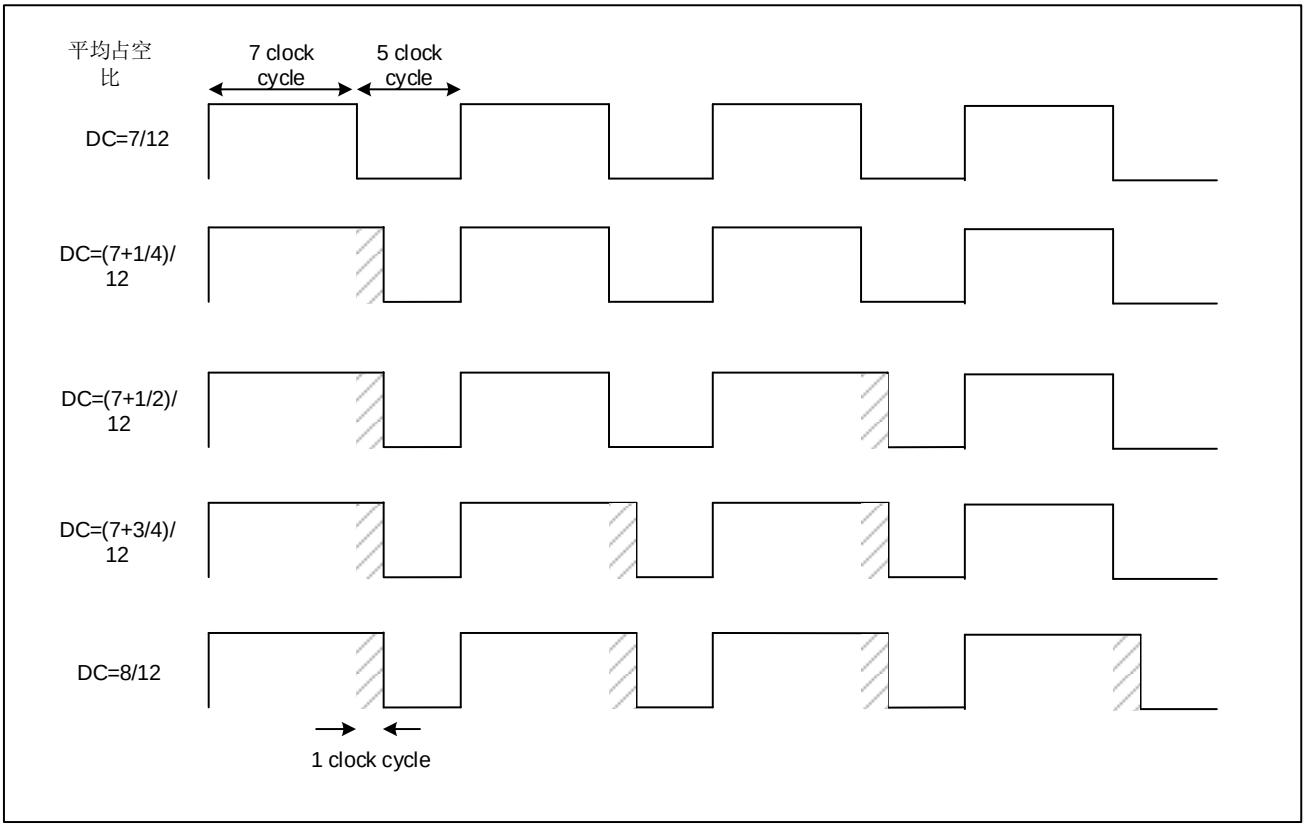


图 23-19 抖动原理

当抖动模式使能后，寄存器的值会自动如下变化：

- 最低的 4 bits 是用于提高分辨率的编码(小数部分)
- 高位被左移到 19:4 作为基值进行编码（整数部分）

注：当抖动模式打开或关闭时，ARR 和 CCR 的值会自动更新（例如，DITHEN=0 时 ARR=0x05，那么 DITHEN=1 时 ARR=0x50），复位 DITHEN 位时必须遵循以下步骤

1. CEN 和 ARPE 位必须复位
2. ARR[3:0]位必须复位
3. DITHEN 位必须复位

4. CCIF 标志位必须清除
5. CEN 位可以置位（可设置 ARPE=1）

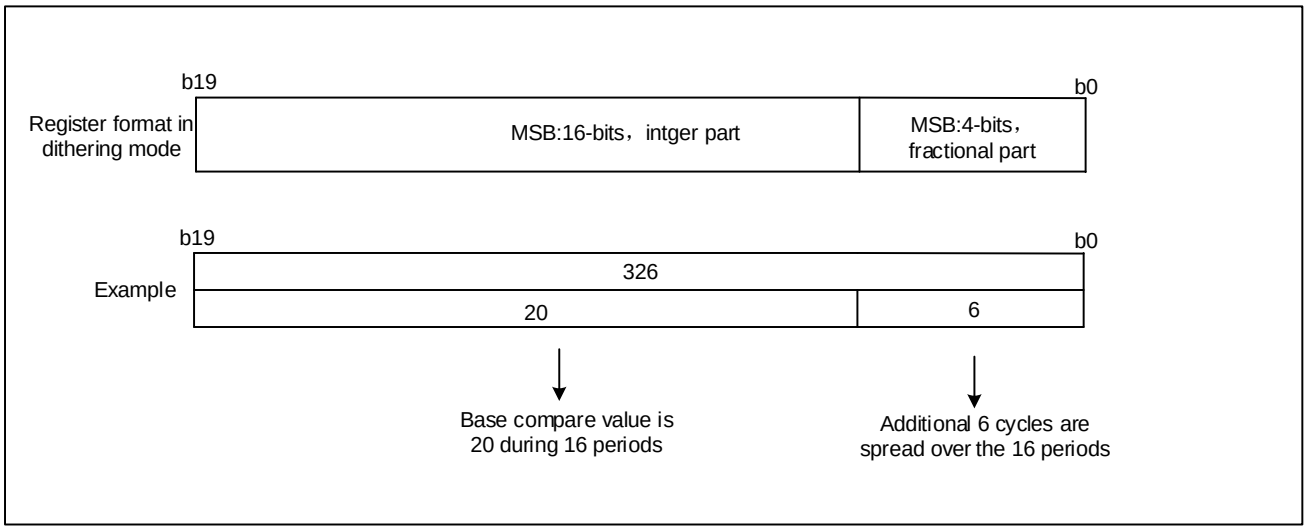


图 23-20 抖动模式中数据映射与寄存器编码

最小的频率计算公式如下：

$$\text{根据分辨率} = \frac{\text{定时器频率}}{\text{PWM 频率}} \qquad \text{推出：最小 PWM 频率} = \frac{\text{定时器频率}}{\text{最大分辨率}}$$

$$\text{无抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65536}$$

$$\text{有抖动模式时：最小 PWM 频率} = \frac{\text{定时器频率}}{65535+\frac{15}{16}}$$

注：抖动模式里 TIMx_ARR 和 TIMxCCRy 的最大值被限定到 0xFFFEF（对应于整数部分的 65534 和抖动部分的 15），超出 0xFFFEF 会存在溢出情况，即 arr=0XFFFF+1=0。

下图显示的是抖动模式下，无论 PWM 频率多少都可以提高 PWM 的分辨率

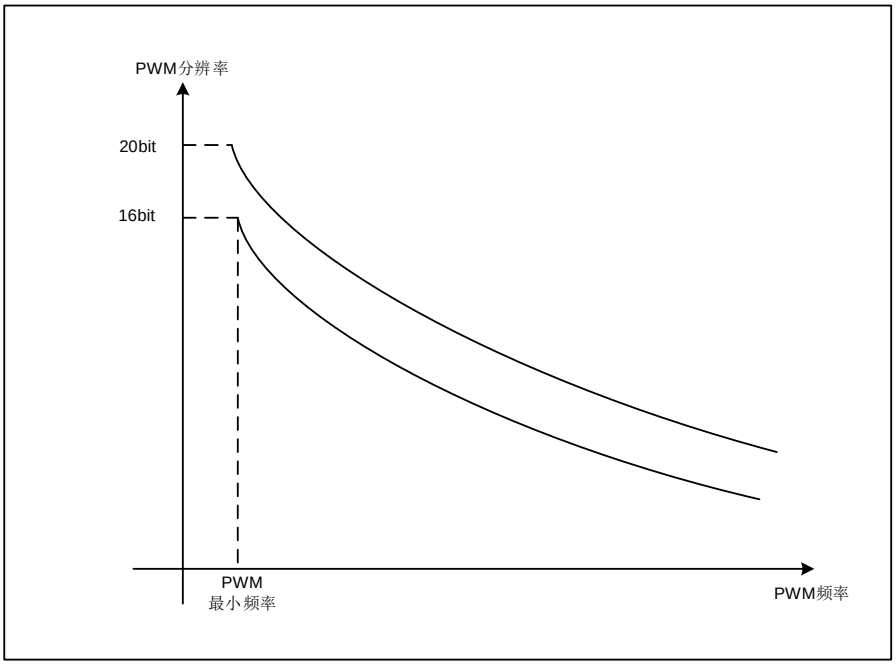


图 23-21 PWM 分辨率 VS 频率

连续 16 个周期内占空比和周期变化情况如下图：

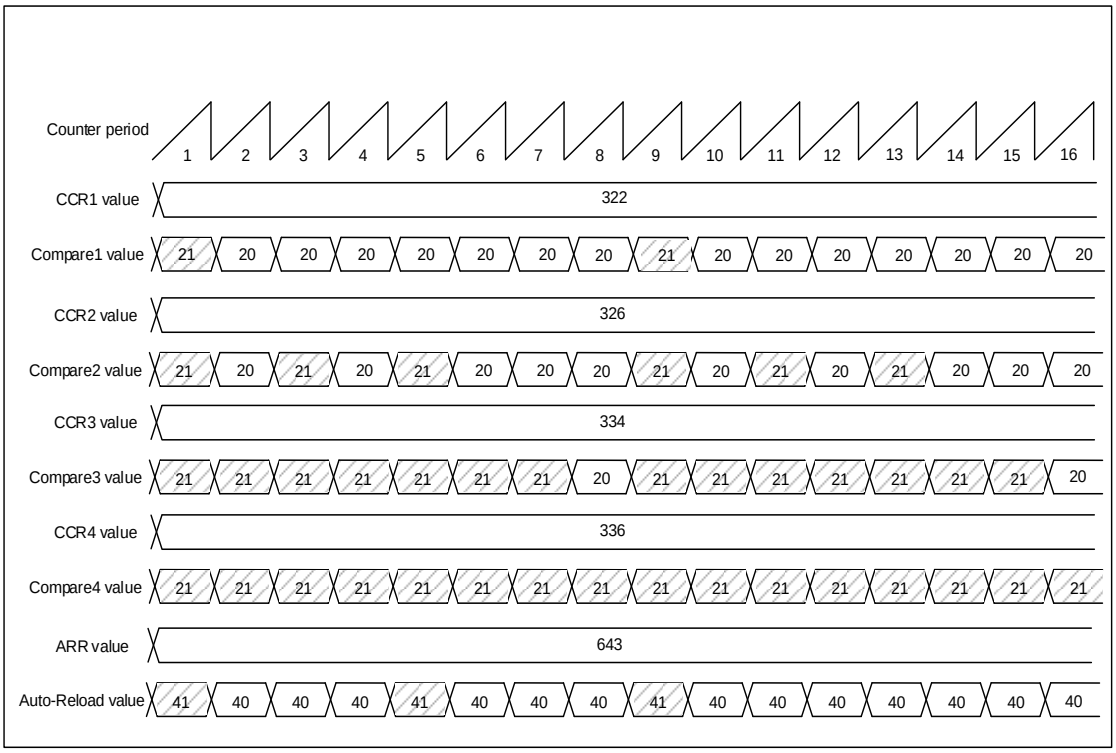


图 23-22 PWM 抖动模式

自动重新加载和比较值增量按照如下表描述的特定模式分布，抖动序列的增量分布尽可能地均匀，最大限度减少整体波纹

表 23-1 CCR 和 ARR 寄存器抖动变化

LSB value	PWM 周期															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000																
0001	+1															
0010	+1								+1							
0011	+1				+1				+1							
0100	+1				+1				+1				+1			
0101	+1		+1		+1				+1				+1			
0110	+1		+1		+1				+1		+1		+1			
0111	+1		+1		+1		+1		+1		+1		+1			
1000	+1		+1		+1		+1		+1		+1		+1		+1	
1001	+1	+1	+1		+1		+1		+1		+1		+1		+1	
1010	+1	+1	+1		+1		+1		+1	+1	+1		+1		+1	
1011	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1		+1	
1100	+1	+1	+1		+1	+1	+1		+1	+1	+1		+1	+1	+1	
1101	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1		+1	+1	+1	
1110	+1	+1	+1	+1	+1	+1	+1		+1	+1	+1	+1	+1	+1	+1	
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	

23.2.11 组合 PWM 模式（仅 tim15）

组合 PWM 模式允许通过对各脉冲间进行可编程的延迟和相移来产生 PWM 信号。频率由 ARR 决定，占空比和延迟由两个 CCR 决定。产生的信号 ocxrefc 由两个参考 PWM 间进行与/或逻辑产生

- OC1REFC（or OC2REFC）由 TIMx_CCR1 和 TIMx_CCR2 控制

通过向 OCxM 写 1100(组合 PWM 模式 1，逻辑或输出）和 OCxM 写 1101(组合 PWM 模式 2，逻辑与输出)，可以在双通道（每对 CCR 寄存器控制一个 OCx 输出）上独立选择不对称的 PWM 模式

当给定的通道被用作组合 PWM 通道，其互补通道必须被配置为相反的 PWM 模式(例如，一路为组合 PWM 模式 1、一路为组合 PWM 模式 2)

下图为组合模式生成信号示例，包括

- 通道 1 为组合 PWM 模式 2
- 通道 2 为 PWM 模式 1

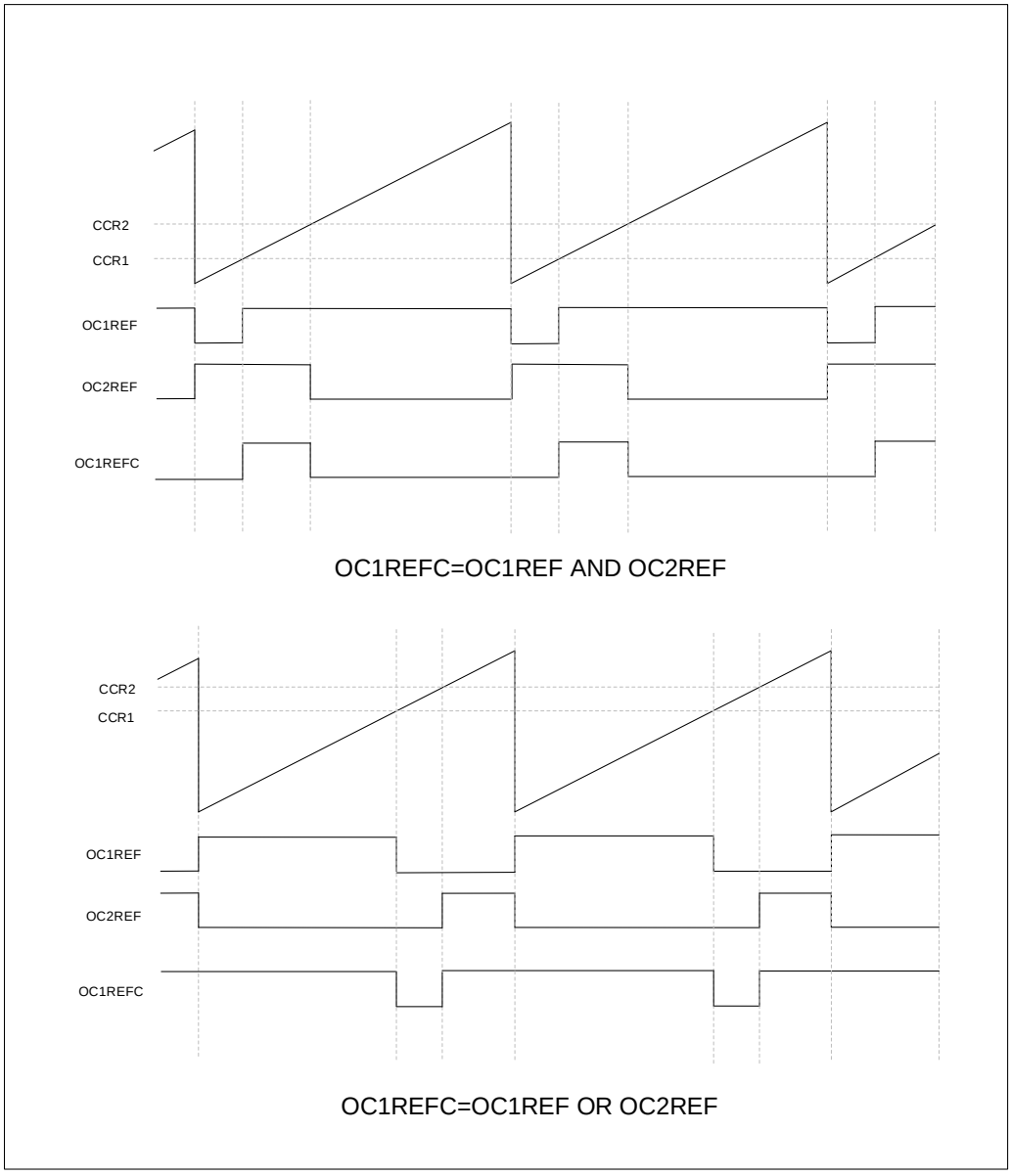


图 23-23 通道 1 和通道 2 的组合 PWM 模式

23.2.12 互补输出和死区插入

通用控制定时器(TIM15_16_17)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 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 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：TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见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 IDLE 状态时(MOE 下降到 0)死区被激活。同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。通过配置 TIMx_BDTR 寄存器中的 DTG[7:0]位，可以控制所有通道的死区发生器。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输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

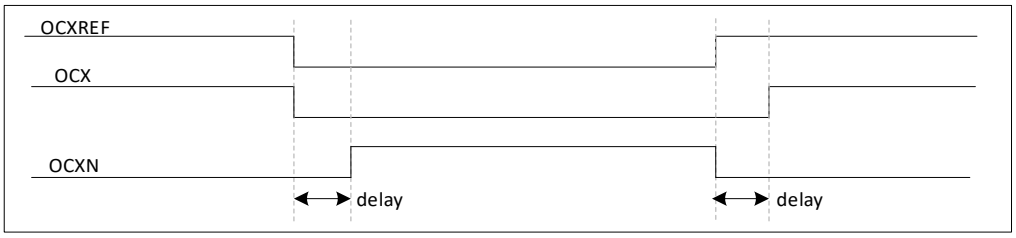


图 23-24 带死区插入的互补输出

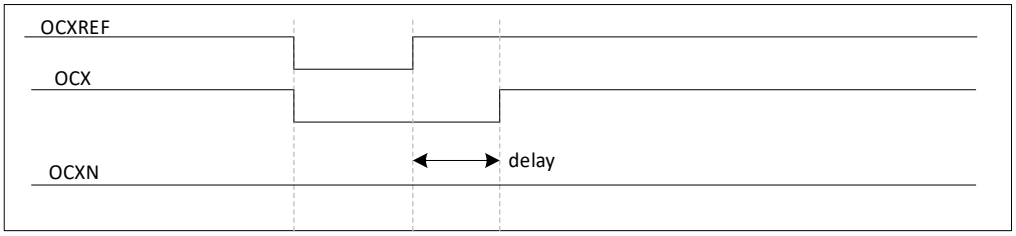


图 23-25 死区波形延迟大于负脉冲

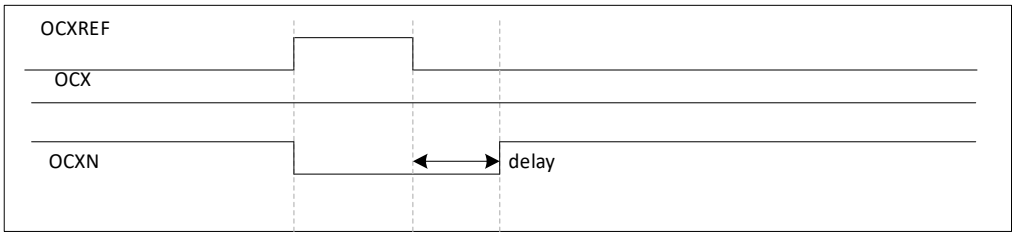


图 23-26 死区波形延迟大于正脉冲

当 DTAE=1，上升沿死区时间由 TIMx_BDTR 的 DTG[7:0]位配置，下降沿死区时间由 TIMx_DTR2 的 DTGF[7:0]位配置。DTAE 的写入需要在计数器使能前，当 CEN=1 时，不允许修改 DTAE。

可以通过预加载的机制在 PWM 工作过程中修改死区时间，当 TIMX_DTR2 的 DTPE 被置位时死区时间配置寄存器 DTG[7:0]和 DTGF[7:0]被预加载，在下一个更新事件时预加载的数据被加载到缓存寄存器中。

注:如果在计数器被使能时，DTPE 位被置位，则自最后一次更新以来的写的新数据都会被丢弃，而使用之前的值

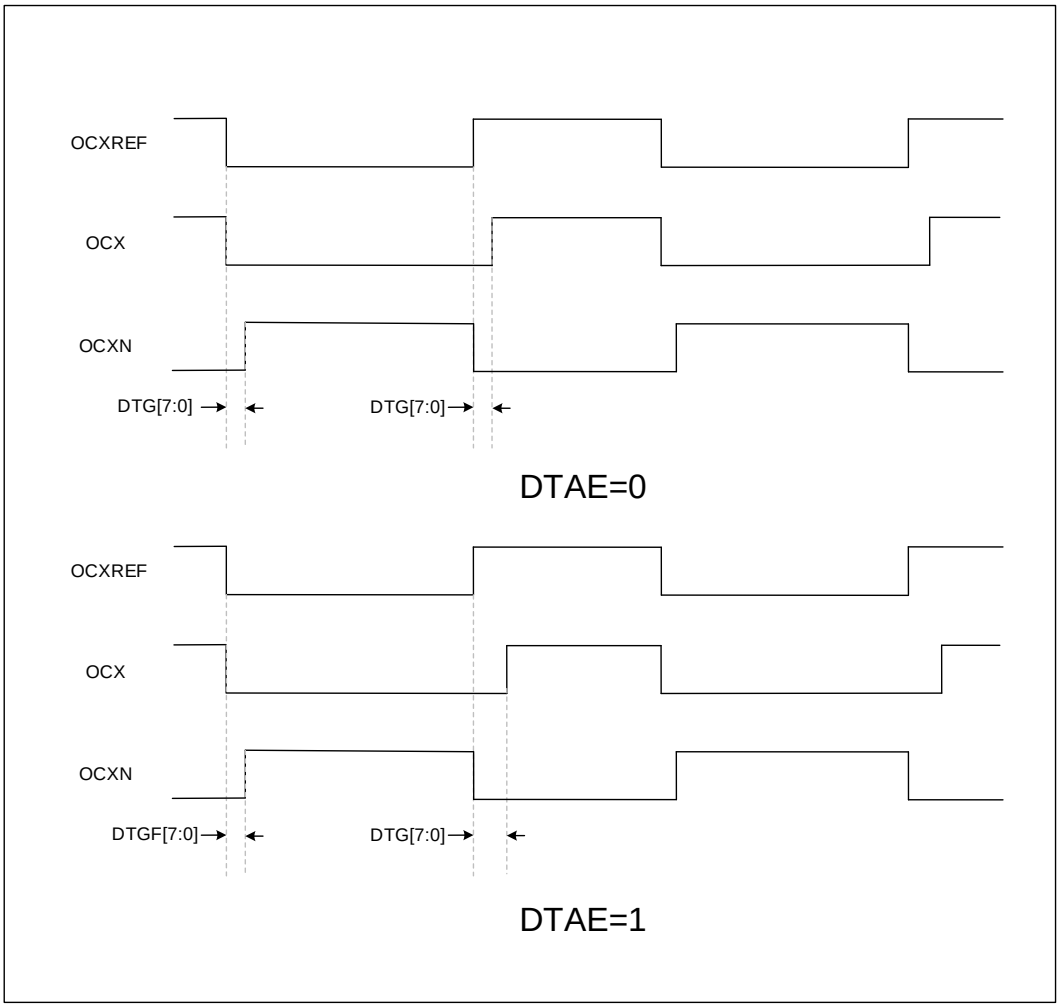


图 23-27 非对称死区时间

23.2.12.1 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 TIMx_CCER 寄存器的 CCxE 和 CCxNE 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：当只使能 OCxN(CCxE=0, CCxNE=1)时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0,则 OCxN=OCxREF。另一方面,当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1),当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

23.2.13使用刹车功能

刹车功能的目的是保护由 TIMER 输出的 PWM 驱动的电源开关。两路刹车的输入通常是连接到三相逆变器和功率级芯片的故障输出。当刹车信号有效时，会立即关断 PWM 的所有输出，并将它们强制到预先设置好的一个安全的电平状态。部分 MCU 内部的事件也可以作为触发信号来关闭输出。

刹车事件有两路通道，通道 1 包括了系统级的故障(clock failure、ECC、奇偶校验等)和应用层的故障（通过输入引脚或者内置的比较器），并且可以在一个死区区间后强制输出到预先设置好的电平（无论工作还是空闲）。通道 2 仅包含一些应用层的故障，并且强制输出为无效状态。

在刹车过程中输出使能信号和输出电平由以下多个 bit 控制

- TIMx_BDTR 的 MOE 位可以通过软件或 2 路刹车事件复位来实现使能和关闭输出
- TIMx_BDTR 的 OSSI 位用于定义当处于空闲状态时 timer 是否还控制输出，或者释放对 GPIO 控制器的控制（高阻状态）
- TIMx_CR2 的 OISx 和 OISxN 位用于无论工作还是空闲状态下都将输出设置为无效电平，OCx 和 OCxN 输出在同一时刻不能同时设置为有效电平，无论 OISx 和 OISxN 为何值。

当从复位中退出时，刹车功能被关闭并且 MOE 为 0。刹车功能的开启可通过使能 TIMxBDTR 的 BKE 位。刹车输入的极性可通过 BKP 配置。BKEx 和 BKPx 可同时修改。当写入 BKE 和 BKP 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 MOE 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 TIMx_BDTR 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 MOE=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。

刹车通道的源；

- 外部源可连接到其中一个 TIMx_BKIN 引脚上（选择 GPIO 和配置寄存器），再加上极性选择和滤波
- 内部源包含 2 部分
- 来自刹车比较器信号（tim_brk_cmpx）
- 来自系统刹车请求

刹车事件也可以由软件生成，可置位 TIMx_EGR 的 BG 位，所有刹车源均或起来作为 tim_brk 刹车输入。如下图：

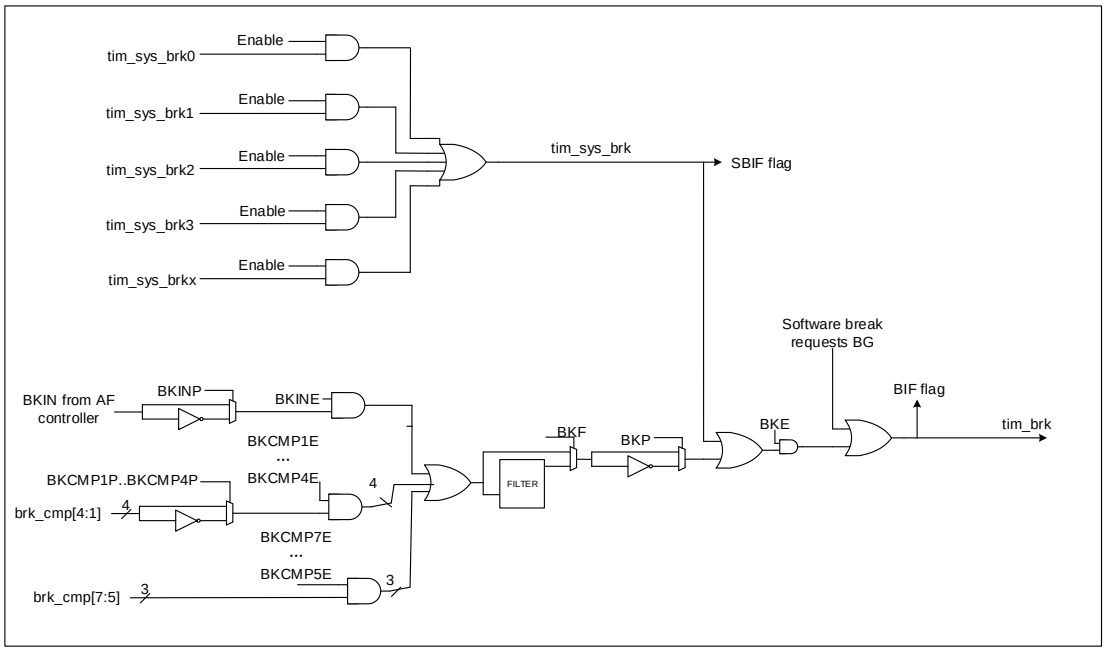


图 23-28 刹车电路

注意：只有没有滤波的时候才能保证是异步的操作，如果滤波开启了，必须使用一个故障保护时钟模式来保证刹车中断事件得到处理（例如使用内部 PLL 或 CSS，无滤波）。

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者释放对 GPIO 控制器的控制(由 OSSI

位选择)。这个特性在 MCU 的振荡器关闭时依然有效。

- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放输出控制，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 OISx 和 OISxN 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，因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个时钟周期)。

  – 如果 OSSI=0，定时器释放输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 刹车状态标志会被拉高（TIMx_SR 的 BIF）。如果设置了 TIMx_DIER 寄存器中的 BIE 位，当刹车状态标志(TIMx_SR 寄存器中的 BIF 位)为 ‘1’ 时，则产生一个中断。如果设置了 TIMx_DIER 寄存器中的 TDE 位，则产生一个 DMA 请求。
- 如果设置了 TIMx_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 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：当 AOE 为 1 时如果 MOE 被 CPU 复位了，输出会进入空闲状态，并被强制到无效电平或者高阻（取决于 OSSI 的值），如果 MOE 和 AOE 均被 CPU 复位，输出将进入无效状态并且由 OISx 位来驱动输出电平。

注：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 MOE。同时，状态标志 BIF 不能被清除。

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(死区持续时间，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性)。用户可以通过设置 TIMx_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，参看刹车和死区寄存器(TIMx_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

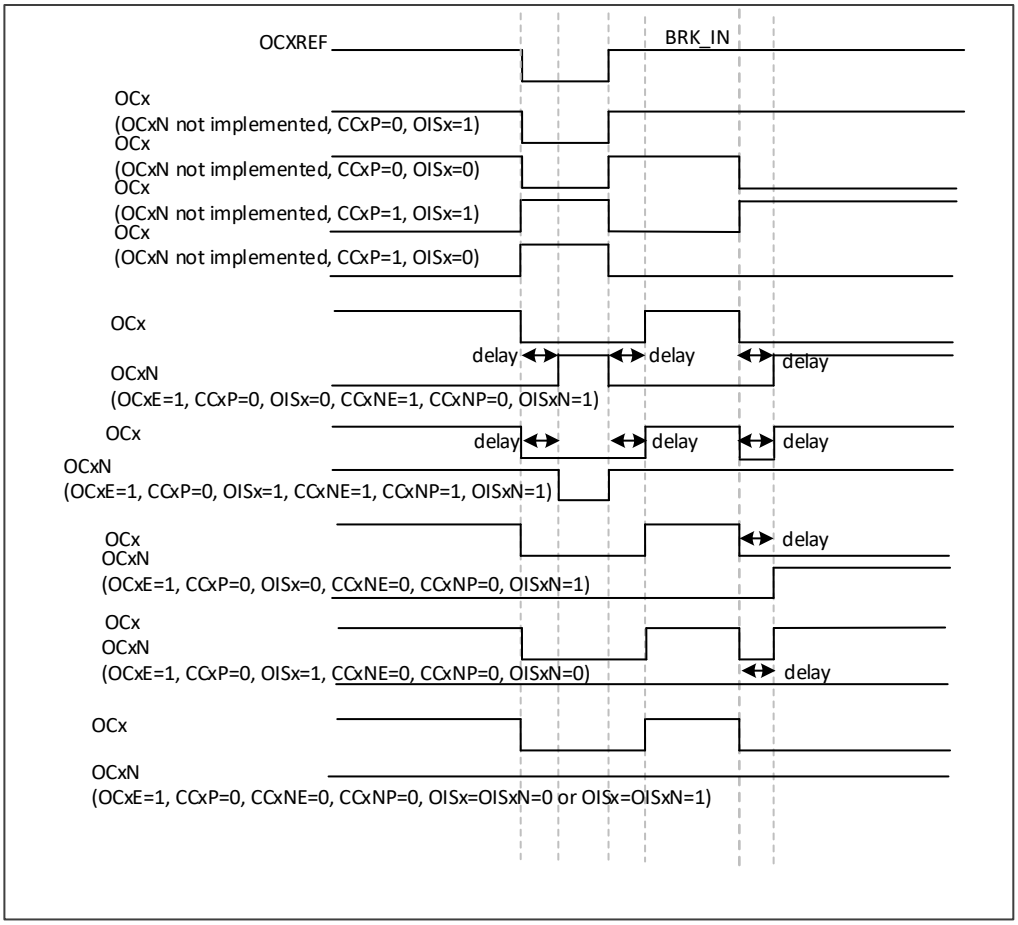


图 23-29 响应刹车的输出

23.2.14双向刹车输入

TIM15/16/17 具备双向的刹车 I/O，如下图所示：

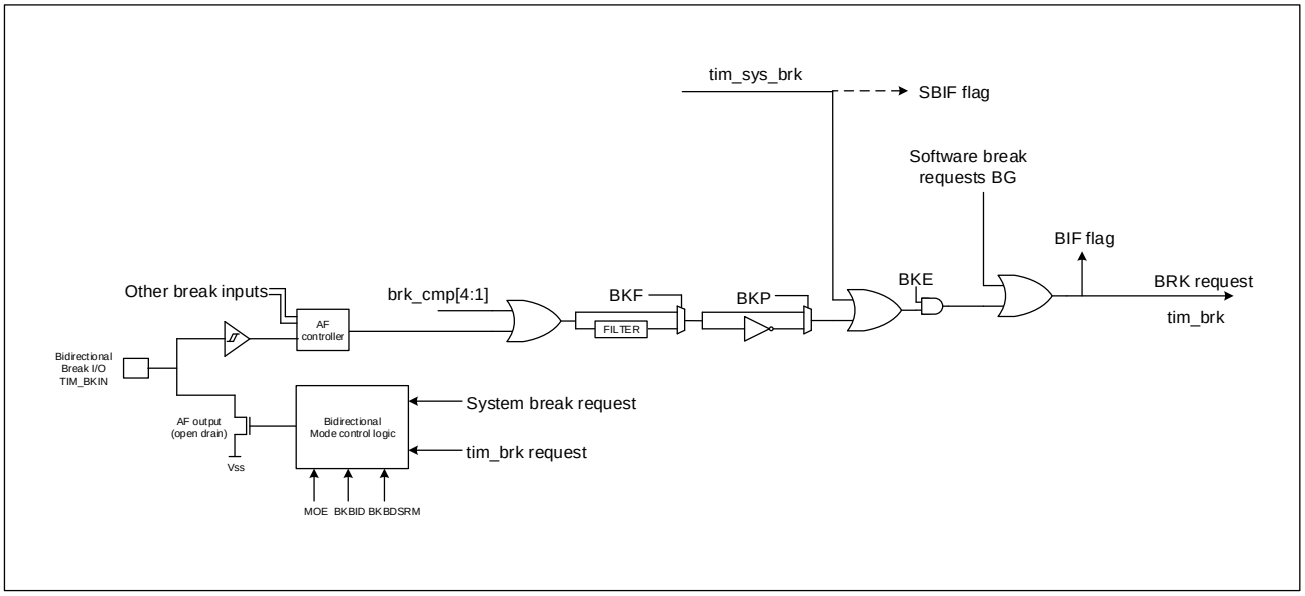


图 23-30 输出重定向

I/O 支持特性：

- 提供一种板级的全局刹车信号，通过唯一的既可做输入也可做输出的引脚，向外部 MCU 和栅极驱动器发送故障信号
- 当需要合并多个内部和外部的刹车源时，内部的刹车源和多个外部的开漏源进行或运算后作为刹车事件的的唯一触发信号

通过 TIMxBDTR 的 BKBID 位控制 tim_brk 的输入为双向模式。BKBID 可以使用 TIMxBDTR 的 LOCK 位（LOCK level 1）设置为 1，实现被锁定为只读模式。

tim_brk 输入可使用双向模式，并且要求 I/O 配置成开漏模式且极性为低有效（通过 BKINP、BKP 控制位）。无论来自系统的（例如 CCS）、还是板级外设的、还是刹车输入强制一个低电平表示刹车输入故障事件，都可以作为刹车请求。但出于安全的考虑，当极性位未有效配置时（配置成高电平有效时），双向模式会被禁止。

软件刹车事件（BG）也会引起刹车 I/O 被拉低，用于向外部设备显示本 timer 进入到了刹车状态。当然当 BKE 为 1 时才有可以。当 BKE 为 0，发生软件刹车事件，输出会被置为安全的状态并产生刹车标志信号，但是对 TIMx_BKIN I/O 没有影响

一种安全的解除机制可以防止系统被限制性锁死（在刹车输入口的低电平触发了一个刹车，该刹车又将输入口强制成低电平）

当 BKDSRM 位被置为 1 时,可以释放刹车输出控制,清除故障标志信号并提供重新配置系统的可能性。

任何情况下都不能关闭刹车保护电路：

- 刹车输入路径一直有效：即使 BKDSRM 位被置 1 和开漏控制被释放，刹车事件还是有效的。这防止在还处于刹车状态时 PWM 输出可以重启
- 当输出被使能的时候（MOE=1）,BKDSRM 位无法解除刹车保护

表 23-2 刹车保护状态

MOE	BKBID	BKDSRM	刹车保护状态
0	0	X	启用
0	1	0	启用
0	1	1	解除
1	X	X	启用

23.2.14.1 对刹车电路进行启用和重启用

默认情况下，刹车电路（在输入或双向模式下）处于启用状态

在发生刹车后需要遵循下面的程序来重启动保护：

- BKDSRM 位必须置 1 用于释放输出控制
- 软件必须要等到系统刹车状态消失然后清除 SBIF 状态标志（或者在重启用前系统地清除它）
- 软件必须轮询 BKDSRM 位直到它被硬件清除（当应用刹车状态消失）

通过这种方式，刹车电路被激活，并且 MOE 位可以被重新使能 PWM 输出

23.2.15在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为‘1’，能够用 OCREF_CLR_INT 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计数溢出所产生的更

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。可以在几个输入中选择 OCREF_CLR_INT 输入，如下图所示。

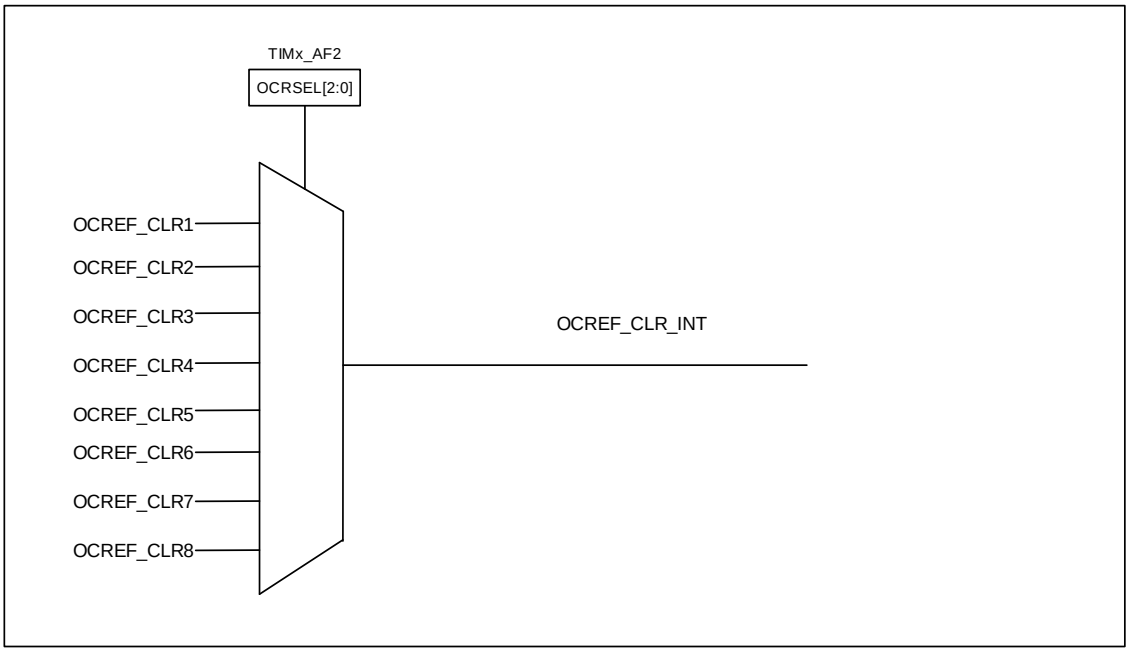


图 23-31 OCREF_CLR 输入选择

23.2.16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 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器。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更改所有通道的配置。COM 可以通过设置 TIMx_EGR 寄存器的 COMG 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位(TIMx_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIMx_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

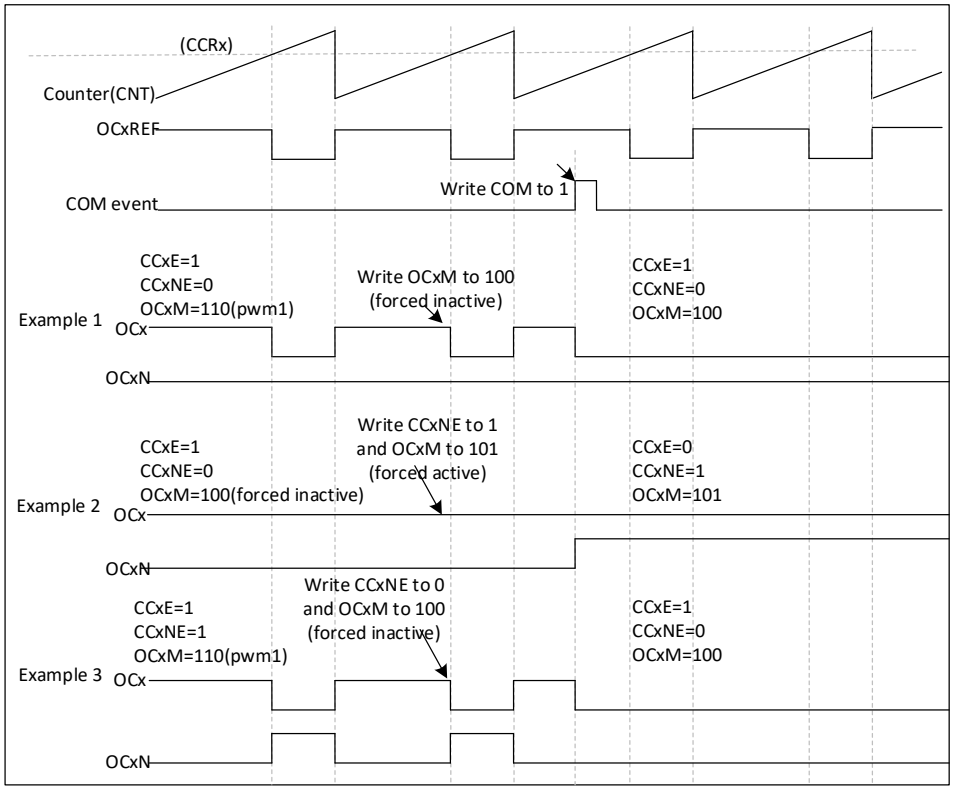


图 23-32 产生六步 PWM，使用 COM 的例子(OSSR=1)

23.2.17 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 TIMx_CR1 寄存器中的 OPM 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 < CCRx ≤ ARR（特别地, 0 < CCRx)，
- 向下计数方式：计数器 CNT > CCRx。

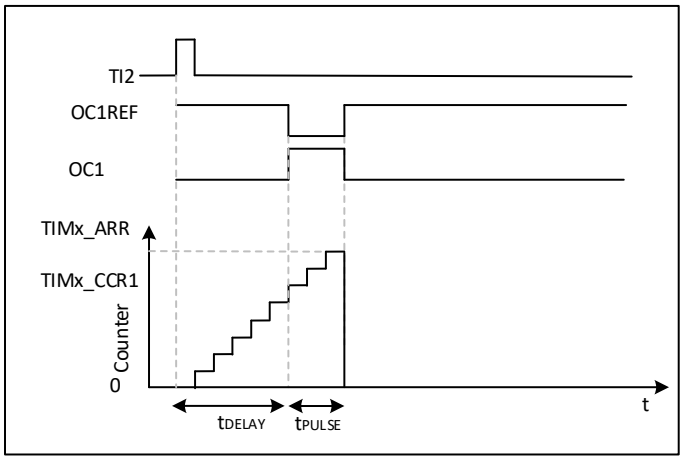


图 23-33 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 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)

- tDELAY 由 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 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 0 到 1 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 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，CC1P=0。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=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 OPM=0 时，重复模式被选中。

23.2.17.1 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 CEN 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 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 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。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

23.2.18 可重触发的单脉冲模式(仅 tim15)

该模式允许计数器响应激励启动并产生长度可编程的脉冲，与上节所述的不可再触发单脉冲模式有以下区别：

- 触发一发生，脉冲就开始（无可编程延迟）。
- 如果在前一个触发所产生的脉冲完成之前发生新的触发，则延长脉冲。

要使用可重触发的单脉冲模式，计时器必须处于从模式，TIMx_SMCR 寄存器中的位 SMS[3:0]=“1000”（组合模式-复位+触发），OCxM[3:0]位设置为“1000”或“1001”（可重新触发 OPM 模式 1 或 2）。

注：出于兼容性原因，OCxM[3:0]和 SMS[3:0]位字段被分成两部分，最高有效位与 3 个最低有效位位置不连续。

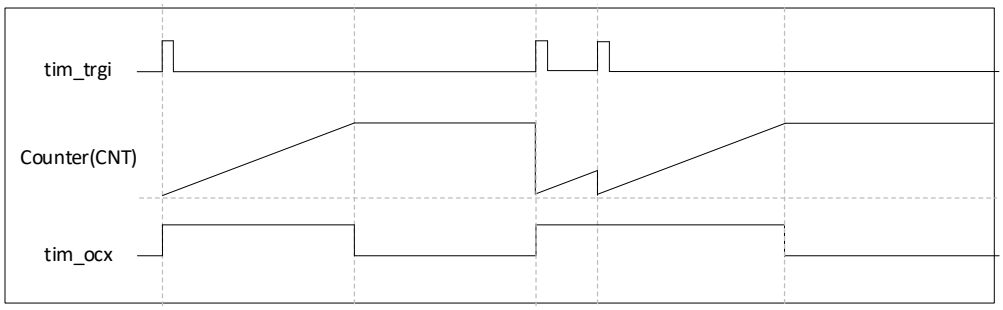


图 23-34 可重触发的单脉冲模式的例子

23.2.19 UIF 位重映射

TIMx_CR1 寄存器的 UIFREMAP 位能够强制连续地将 UIF 复制到定时器计数器的第 31 位(TIMxCNT[31])。这样可以允许计数器的值和由 UIFCPY 标志信号展现的一种潜在的滚动状态都可以读出。在特殊情况下，这样可以通过避免竞争条件来简化计算，例如由同时处理计数器读出和更新中断。UIF 和 UIFCPY 标志的产生没有延迟。

23.2.20 定时器输入异或功能（仅 tim15）

TIMx_CR2 寄存器中的 TI1S 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 2 个输入端为 TIMx_CH1、TIMx_CH2。

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。对于测量两个输入信号的边缘之间的间隔是有用的。

23.2.21 TIMx 定时器和外部触发的同步

TIMx 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、触发、复位+触发和门控+复位模式。

23.2.21.1 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 TIMx_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR，TIMx_CCRx)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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=0100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=00101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=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(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(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 TIMx_ARR=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。

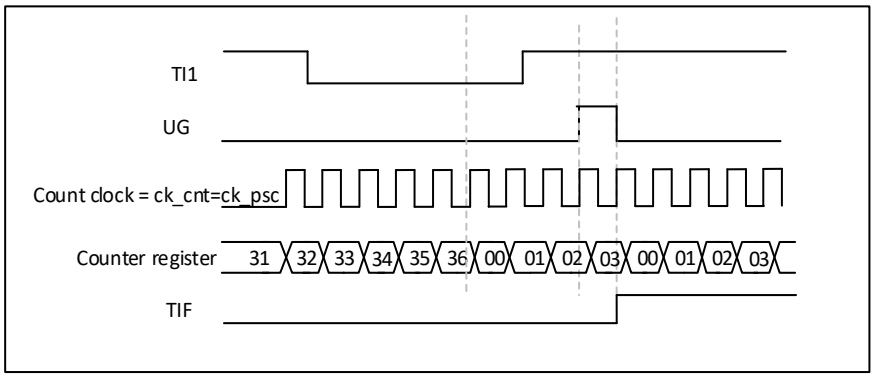


图 23-35 复位模式下的控制电路

23.2.21.2 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 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=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 TIMx_SR 中的 TIF 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。

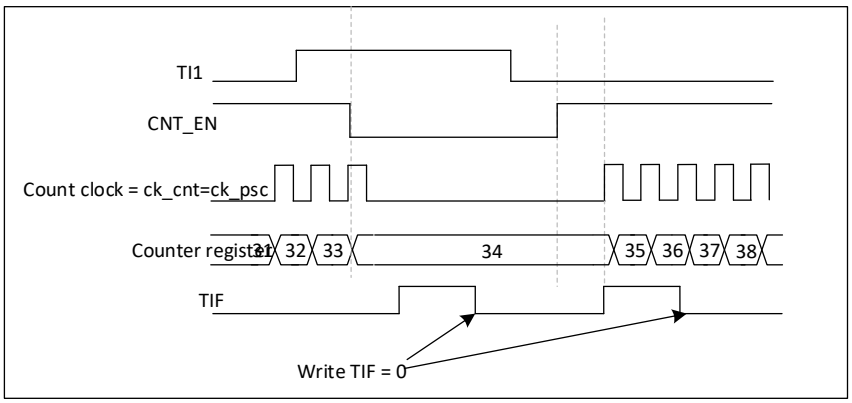


图 23-36 门控模式下的控制电路

23.2.21.3 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 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=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 TIF 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。

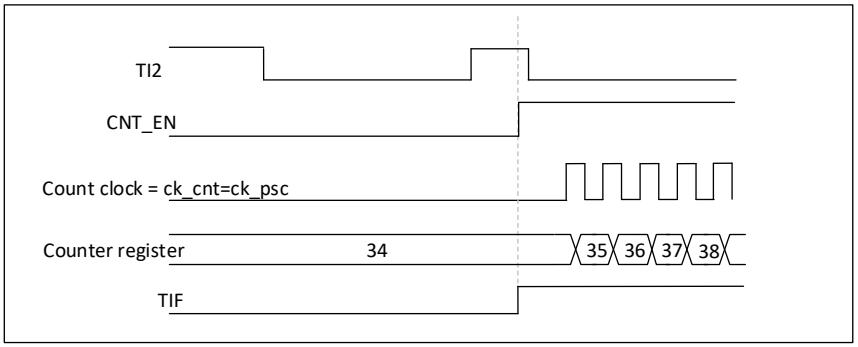


图 23-37 触发器模式下的控制电路

23.2.21.4 从模式：复位加触发组合模式（仅 tim15）

在该模式下，触发输入的上升沿会初始化计数器，并产生寄存器的更新信号，开启计数

该模式用于单脉冲模式

23.2.21.5 从模式：门控加复位组合模式（仅 tim15）

当触发输入为高时计数器的时钟才使能。当触发信号拉低时计数器停止。计数器的开始和停止都可以控制

该模式允许检测超出范围的 PWM 信号（占空比超过最大预期值）

23.2.22定时器同步（仅 tim15）

所有 TIMx 定时器在内部相连，用于定时器同步或链接。详见第 20.2.22 节：定时器同步。

23.2.23DMA 突发模式

TIMx 定时器当发生单个事件时有生成多个 DMA 请求的能力。主要目的是能够多次重新编程定时器的部分功能而无需软件的开销。但是也可以在固定的间隔时间内读取一行中的几个寄存器。

DMA 控制器的目的地是唯一的并且必须指向虚拟寄存器 TIMx_DMAR.当给定的定时器事件发生，定时器发起一系列 DMA 请求。每个写入 TIMx_DMAR 寄存器的操作实际上是重定向一个定时器寄存器的。TIMx_DCR 寄存器中的 DBL[4:0]位设置 DMA 突发长度。当对 TIMx_DMR 地址进行读取或写入访问时，定时器会识别突发传输，即传输次数（以半个字或字节为单位）

TIMx_DCR 的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址(当通过 TIMx_DMAR 地址完成读写操作)。DBA 定义为从 TIMx_CR1 寄存器地址开始的偏移量。

例子：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发特性是用于发生更新事件时更新 CCRx 寄存器中的内容，通过 DMA 传输半字到 CCRx。

通过以下步骤来完成：

1. 如下配置对应的 DMA 通道：
   a) DMA 通道外设地址是 DMAR 寄存器的地址
   b) DMA 通道存储地址是 RAM 中的 BUFF 地址，包含了 DMA 传给 CCRx 的数据
   c) 传输的数据个数=3（见注释）
   d) 轮询模式关闭
2. 通过配置 DBA 和 DBL 实现配置 DCR 寄存器：DBL=3，DBA=0XE
3. 使能定时器更新 DMA 请求（UDE=1）
4. 使能定时器
5. 使能 DMA 通道

这个例子此示例适用于每个 CCRx 寄存器更新一次的情况。如果每个 CCRx 都更新两次，数据的传输个数就需要设置为 4。举例假设 RAM 的 buffer 中包含了 data1，data2，data3, data4。数据依次传输到 CCRx 中：第一次更新 DMA 请求，data1 传输给 CCR1，data2 传输给 CCR2；第二次更新 DMA 请求，data3 传输给 CCR1，data4 传输给 CCR2。

注：空值可以写入到保留寄存器

23.2.24 DMA 请求

TIM15/TIM16/TIM17 可以生成 DMA 请求，如下表所示：

表 23-3 DMA 请求

DMA 请求源	DMA 使能控制位
更新	UDE
比较/捕获 1	CC1DE
COM(1)	COMDE
触发(1)	TDE

其中 1 仅用于 TIM15。

23.2.25 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_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

23.2.26 TIM15/TIM16/TIM17 中断

中断事件	事件标志	中断使能控制位
更新	UIF	UIE
比较/捕获 1	CC1IF	CC1IE

比较/捕获 2	CC2IF	CC2IE
COM	COMIF	COMIE
触发	TIF	TIE
刹车	BIF	BIE

23.3 TIM15 寄存器描述

23.3.1 TIM15 控制寄存器 1（TIMx_CR1）

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DITHEN	UIFREMAP	Res	CKD[1:0]		ARPE	Res			OPM	URS	UDIS	CEN
			RW	RW		RW	RW	RW				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:13	保留	-	-	-
12	DITHEN	RW	0	抖动使能（Dithering enbale） 0：抖动关闭 1：抖动打开 注：该位只能在 CEN 位复位时才能修改
11	UIFREMAP	RW	0	UIF 状态位重映射使能（UIF status bit remapping） 0：无重映射，UIF 状态位不复制到 TIMx_CNT 寄存器的第 31 位 1：重映射，UIF 状态位复制到 TIMx_CNT 寄存器的第 31 位
10	保留	-	-	-
9:8	CKD	RW	2’h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT 11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	保留	-	--	-
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

23.3.2 TIM15 控制寄存器 2（TIMx_CR2）

Address offset:0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res	CCPC
					RW	RW	RW	RW	RW	RW	RW	RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	-
10	OIS2	RW	0	输出空闲状态 2(OC2 输出)。参见 OIS1 位。
9	OIS1N	RW	0	输出空闲状态 1(OC1N 输出)（Output Idle state 1) 0：当 MOE=0 时，死区后 OC1N=0； 1：当 MOE=0 时，死区后 OC1N=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
8	OIS1	RW	0	输出空闲状态 1(OC1 输出)（Output Idle state 1) 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0； 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
7	TI1S	RW	0	TI1 选择（TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
6:4	MMS	RW	3’h0	主模式选择（Master mode selection)

				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 – TIMx_EGR 寄存器的 UG 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 可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 CEN 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 TIMx_SMCR 寄存器中 MSM 位的描述)。 010: 更新 – 更新事件被选为触发输出(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 CC1IF 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 – OC1REFC 信号被用于作为触发输出(TRGO)。 101：比较 – OC2REFC 信号被用于作为触发输出(TRGO)。 其他：保留 注意： 1.从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。 2.若主从定时器不在同一总线上，主模式应该配置为能被从定时器采到的宽度。
3	CCDS	RW	0	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。
2	CCUS	RW	0	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(CCPC=1)，只能通过设置 COM 位更新它们； 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 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。
1	保留	-	-	-
0	CCPC	RW	0	捕获/比较预装载控制位（Capture/compare preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新（COMG 位设置或 tim_trgi 上检测到的上升沿，具体取决于 CCUS 位）。 注：该位只对具有互补输出的通道起作用。

23.3.3 TIM15 从模式控制寄存器（TIMx_SMCR）

Address offset:0x08

Reset value：0x0000 0000

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

	RW	RW	RW	RW		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21:20	TS[4:3]	RW	2'h0	详见 TS[2:0]描述
19:17	保留	-	-	-
16	SMS[3]	RW	0	详见 SMS[2:0]描述
15:8	保留	-	-	-
7	MSM	RW	0	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。
6:4	TS	RW	3'h0	触发选择（Trigger selection) 选择用于同步计数器的触发输入。 00000：内部触发 0(ITR0) 00001：内部触发 1(ITR1) 00010：内部触发 2(ITR2) 00011：内部触发 3(ITR3) 00100：TI1 边沿检测（TI1F_ED） 00101：滤波后的定时器输入 1（TI1FP1） 00110：滤波后的定时器输入 2（TI1FP2） 00111：外部触发输入（ETRF） 01000：内部触发 4(ITR4) 01001：内部触发 5(ITR5) 01010：内部触发 6(ITR6) 01011：内部触发 7(ITR7) 01100：内部触发 8(ITR8) 01101：内部触发 9(ITR9) 01110：内部触发 10(ITR10) 01111：内部触发 11(ITR11) 10000：内部触发 12(ITR12) 10001：内部触发 13(ITR13) 10010：内部触发 14(ITR14) 10011：内部触发 15(ITR15) 10100：内部触发 16(ITR16) 10101：内部触发 17(ITR17) 10110：内部触发 18(ITR18) 其它：保留 更多有关 ITRx 的细节，参见系统级联表 注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
3	保留	-	-	-
2:0	SMS	RW	3'h0	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) 0000：关闭从模式 – 如果 CEN=1，则预分频器直接由内部时钟驱动。

				0001：保留。 0010：保留。 0011：保留。 0100: 复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 0101：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。 0110：触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 0111：外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 1000：复位+触发组合模式：-触发输入（TRGI）的上升沿初始化计数器，生成寄存器更新并启动计数器 1001：门控+复位组合模式：-触发输入（TRGI）为高时计数器时钟使能。一旦触发信号拉低时计数器就停止并复位。计数器的开始和停止均可控。 其他：保留 注：如果 TI1F_ED 被选为触发输入(TS=00100)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: 必须在接收主定时器事件之前，使能从设备（定时器，ADC）来接收 TRGO，并且时钟频率不能再接收过程中改变

23.3.4 TIM15 DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	COMDE	Res			CC1DE	UDE	BIE	TIE	COMIE	Res		CC2IE	CC1IE	UIE
	RW	RW				RW	RW	RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:15	保留	-	-	-
14	TDE	RW	0	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。
13	COMDE	RW	0	允许 COM 的 DMA 请求（COM DMA request enable) 0：禁止 COM 的 DMA 请求; 1：允许 COM 的 DMA 请求。
12:10	保留	-	-	-
9	CC1DE	RW	0	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。。
8	UDE	RW	0	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。

7	BIE	RW	0	允许刹车中断（Break interrupt enable) 0：禁止刹车中断; 1：允许刹车中断。
6	TIE	RW	0	触发中断使能（Trigger interrupt enable) 0：禁止触发中断; 1：使能触发中断。
5	COMIE	RW	0	允许 COM 中断（COM interrupt enable) 0：禁止 COM 中断; 1：允许 COM 中断。
4:3	保留	-	-	-
2	CC2IE	RW	0	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。
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断; 1：允许更新中断。

23.3.5 TIM15 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										IC2IF	IC1IF	Res		IC2IR	IC1IR
										RC_W0	RC_W0			RC_W0	RC_W0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res					CC2OF	CC1OF	Res	BIF	TIF	COMIF	Res		CC2IF	CC1IF	UIF
					RC_W0	RC_W0		RC_W0	RC_W0	RC_W0			RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21	IC2IF	RC_W0	0	下降沿捕获 2 标志 参见 IC1IF 描述。
20	IC1IF	RC_W0	0	下降沿捕获 1 标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生; 1：发生下降沿捕获事件。
19:18	保留	-	-	-
17	IC2IR	RC_W0	0	上升沿捕获 2 标志 参见 IC1IR 描述。
16	IC1IR	RC_W0	0	上升沿捕获 1 标志

				仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置 1。它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0：无重复捕获产生； 1：发生上升沿捕获事件。
15:11	保留	-	-	-
10	CC2OF	RC_W0	0	捕获/比较 2 重复捕获标记（Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	RC_W0	0	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 TIMx_CCR1 寄存器时，CC1IF 的状态已经为‘1’。
8	保留	-	-	-
7	BIF	RC_W0	0	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。如果 TIM_DIER 的 BIE=1 则产生中断
6	TIF	RC_W0	0	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。
5	COMIF	RC_W0	0	COM 中断标记（COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新)该位由硬件置‘1’。它由软件清‘0’。 0：无 COM 事件产生； 1：COM 中断等待响应。
4:3	保留	-	-	-
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记（Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	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： 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 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，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1: 计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。

0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=0，重复计数器=0 时产生更新事件。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

23.3.6 TIM15 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								BG	TG	COMG	Res		CC2G	CC1G	UG
								W	W	W			W	W	W

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	BG	W	0	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 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
6	TG	W	0	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：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5	COMG	W	0	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 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 注：该位只对拥有互补输出的通道有效。
4:3	保留	-	-	-
2	CC2G	W	0	产生捕获/比较 2 事件（Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。

				若通道 1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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 TIMx_ARR 的值。

23.3.7 TIM15 捕获/比较模式控制寄存器 1（TIMx_CCMR1）

Address offset:0x18

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							OC2M[3]	Res.							OC1M[3]
							RW								RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

23.3.7.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	OC2M[3]	RW	0	详见 OC2M[2:0]描述
23:17	保留	-	-	-
16	OC1M[3]	RW	0	详见 OC1M[2:0]描述
15	OC2CE	RW	0	输出比较 2 清零使能（Output Compare 2 clear enable)
14:12	OC2M	RW	3'h0	输出比较 2 模式（Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能（Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能（Output Compare 2 fast enable)
9:8	CC2S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	输出比较 1 模式（OutputCompare1mode)

				这些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 CC1P、CC1NP 位。 0000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 0001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 0010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 0011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 0100：强制为无效电平。强制 OC1REF 为低。 0101：强制为有效电平。强制 OC1REF 为高。 0110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平,否则为无效电平; 在向下计数时,一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 0111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平,否则为有效电平; 在向下计数时,一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 1000：可重触发 OPM 模式 1-在递增计数模式下，通道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且通道在下一次更新时再次有效。在递减计数模式下，通道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 1 中那样执行比较，并且信道在下一次更新时再次变为无效 1001：可重触发 OPM 模式 2-在递增计数模式下，信道处于无效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且通道在下一次更新时再次变为无效。在递减计数模式下，信道处于有效状态，直到检测到触发事件（tim_trgi 信号）。然后，如在 PWM 模式 2 中那样执行比较，并且信道在下一次更新时再次变为有效 1010：保留 1011：保留 1100：组合 PWM 模式 1–OC1REF 有与 PWM 模式 1 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 或逻辑组成 1101：组合 PWM 模式 2–OC1REF 有与 PWM 模式 2 中相同的行为，OC1REFC 是由 OC1REF 和 OC2REF 与逻辑组成 1110：保留 1111：保留 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式中，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。 注 3：当通道有互补输出时，该位段可以预加载。如果 TIMx_CR2 的 CCPC 位置 1，然后仅当 COM 事件生成时 OC1M 有效位从预加载中更新新的值
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0: 禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。

				1: 开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 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(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=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(TIMx_CR1 寄存器的 OPM=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。
2	OC1FE	RW	0	输出比较 1 快速使能（Output Compare 1 fast enable) 该位用于加快输出对触发输入事件的响应。必须在单脉冲模式 (TIMx_CR1 寄存器的 OPM 位置 1）中使用，当触发到来时实现脉冲快速输出。 0：根据计数器与 CCR1 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象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。仅能用于 PWM 模式 1 和 PWM 模式 2
1:0	CC1S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

23.3.7.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:12	IC2F	RW	4’h0	输入捕获 2 滤波器（Input capture 2 filter)
11:10	IC2PSC	RW	2’h0	输入/捕获 2 预分频器（Input capture 2 prescaler)
9:8	CC2S	RW	2’h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F	RW	4’h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6

				0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。
1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

23.3.8 TIM15 捕获/比较使能寄存器（TIMx_CCER）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								CC2NP	Res	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								RW		RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	CC2NP	RW	0	输入/捕获 2 互补输出极性（Capture/Compare 2 output polarity) 参考 CC1NP 的描述。
6	保留	-	-	-
5	CC2P	RW	0	输入/捕获 2 输出极性（Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能（Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性(Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效； 1：OC1N 低电平有效。

				注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出)则该位不能被修改。
2	CC1NE	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式）。 01：反相/下降沿： TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 反相（门控模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效(捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1 不反相（门控模式）。 注： 1.对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。 2.一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2，则该位不能被修改。
0	CC1E	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 TIMx_CCR1 寄存器。 0：捕获禁止； 1：捕获使能。 注： 对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表 23-4 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	
MOE 位	**OSSI 位**	**OSSR 位**	**CCxE 位**	**CCxNE 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=OCREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF+极性， OCx=OCREF xor CCxP, 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（取反）+极性+死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开） OCx=CCxP, OCx=0 OCx_EN=0	输出禁止（与定时器断开） OCxN=CCxNP, OCx=0 OCxN_EN=0
		1	0	1	关闭状态(输出使能且为无效电平) OCx=CCxP, OCx_EN=1	OCxREF+极性， OCxN=OCREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+极性， OCx=OCREF xor CCxP, OCx_EN=1	关闭状态（输出使能且为无效电平） OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF+极性+死区，， OCx_EN=1	OCxREF（取反）+极性+死区， OCxN_EN=1
0	0	x	x	x	输出禁止（与定时器断开）	
	1		0	0		
	1		0	1	关闭状态（输出使能但为无效电平） 异步的：OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCx_EN=1;（如果刹车或刹车 2 触发了） 然后若时钟存在（仅刹车 1 触发时有效，不需要刹车 2）：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平 注：刹车 2 仅用于 OSSI=OSSR=1	
	1		1	0		
	1		1	1		

特别的，当死区有效时，输出总是按照死区内输出的规则输出，尽管此时 moe 可能已经被软件或硬件的方式置为 1。

如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

23.3.9 TIM15 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

R															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	UIFCPY	R	0	该位为只读寄存器，将 TIMx_ISR 寄存器的 UIF 位复制过来。如果 TIMx_CR1 中的 UIF 重映射位被复位了，该 bit 位保留并读出来为 0
30:16	保留	-	-	-
15:0	CNT	RW	16'h0	计数器的值（Counter value)

23.3.10 TIM15 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16'h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

23.3.11 TIM15 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	ARR	RW	20'hFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。 无抖动模式（DITHEN=0）： 该寄存器保持自动加载值 抖动模式（DITHEN=1）： 该寄存器保持整数部分 ARR[19:4]，ARR[3:0]包含抖动部分

23.3.12 TIM15 重复计数寄存器（TIMx_RCR）

Address offset:0x30

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	REP	RW	8'h0	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。 每次向下计数器达到 0，会产生一个更新事件并且重复计数器重新从 REP 值开始计数。对 TIMx_RCR 寄存器写入的新值只在下次更新事件发生时才起作用。

23.3.13 TIM15 捕获/比较寄存器 1（TIMx_CCR1）

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR1[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR1	RW/RO	20'h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR1[15:0]的比较值，CCR1[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR1[19:4]，CCR1[3:0]包含抖动部分 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。TIMx_CCR1 寄存器只读

				无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR1[15:0]，CCR1[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR1[19:4]，CCR1[3:0]位被复位

23.3.14 TIM15 捕获/比较寄存器 2（TIMx_CCR2）

Address offset:0x38

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR2[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR2	RW/RO	20’h0	捕获/比较通道 2 的值（Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC2PE 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 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR2[15:0]的比较值，CCR2[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR2[19:4]，CCR2[3:0]包含抖动部分 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。 TIMx_CCR2 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR2[15:0]，CCR2[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR2[19:4]，CCR2[3:0]位被复位

23.3.15 TIM15 刹车和死区寄存器（TIMx_BDTR）

Address offset:0x44

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			BKBID	Res	BKDSRM	Res						BKF[3:0]			
			RW		RW							RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28	BKBID	RW	0	刹车双向性（Break bidirectional） 0：刹车输入 BRK 在输入模式中 1：刹车输入 BRK 在双向模式中 在双向模式中（BKBID 设置为 1），刹车输入被既配置为输入模式也是开漏输出模式。任何有效的刹车事件会将刹车输入电平拉低，向外部设备指示内部发生了刹车事件 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
27	保留	-	-	-
26	BKDSRM	RW	0	刹车解除（Break disarm） 0：刹车输入 BRK 启用 1：刹车输入 BRK 解除 当没有刹车源时该位会被硬件清零 BKDSRM 位必须由软件置位来释放双向输出控制（开漏输出高阻状态），然后轮询它直到被硬件复位，表示故障状态已经消失。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
25:20	保留	-	-	-
19:16	BKF[3:0]	RW	4’h0	刹车滤波（Break filter） 这些位定义了对 BRK 信号采样的频率和对 BRK 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，BRK2 是异步的 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。
15	MOE	RW	0	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 AOE 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(TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 TIMx 捕获/比较使能寄存(TIMx_CCER)。
14	AOE	RW	0	自动输出使能（Automatic output enable) 0：MOE 只能被软件置’1’； 1：MOE 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(TIMx_BDTR 寄存器中的 LOCK 位)设为’1’，则该位不能被修改。
13	BKP	RW	0	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(TIMx_BDTR 寄存器中的 LOCK 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。
12	BKE	RW	0	刹车功能使能（Break enable) 0：禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1：开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，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。
11	OSSR	RW	0	运行模式下“关闭状态”选择（Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下“关闭状态”选择（Off-state selection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOCK	RW	2’h0	锁定设置（Lock 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 TIMx_BDTR 寄存器的 DTG、BKBID、BK2BID、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 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 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 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 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。
7:0	DTG	RW	8'h0	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: DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg，Tdtg = TDTS； DTG[7:5]=10x => DT=(64+DTG[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTG[7:5]=110 => DT=(32+DTG[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTG[7:5]=111 => DT=(32+DTG[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位。

23.3.16 TIM15 死区时间寄存器 2（TIMx_DTR2）

Address offset:0x54

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	DTPE	RW	0	死区时间预加载使能（Deadtime preload enable） 0：死区时间的值不可以预加载 1：死区时间的值可以预加载 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
16	DTAE	RW	0	死区时间不对称使能（Deadtime asymmetric enable） 0：上升沿和下降沿的死区时间相同，都由 DTG[7:0]配置 1：上升沿死区时间由 DTG[7:0]配置，下降沿死区时间由 DTGF[7:0]配置 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
15:8	保留	-	-	-
7:0	DTGF[7:0]	RW	8'h0	下降沿死区时间生成器（Dead-time falling edge generator setup） 该位段定义了插入互补输出时下降沿的死区时间，假设 DT 表示其持续时间: DTGF[7:5]=0xx => DTF=DTGF[7:0] × Tdtg，Tdtg = TDTS；

				DTGF[7:5]=10x => DTF=(64+DTGF[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTGF[7:5]=110 => DTF=(32+DTGF[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTGF[7:5]=111 => DTF=(32+DTGF[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位

23.3.17 TIM15 输入选择寄存器（TIMx_TISEL）

Address offset:0x5C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				TI2SEL[3:0]				Res				TI1SEL[3:0]			
				RW	RW	RW	RW					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:8	TI2SEL	RW	4’h0	TI2 输入选择（tim_ti2[15:0] input） 0000：TIMx_CH2 0001：ti_in1 0010：ti_in2 其他：保留
7:4	保留	-	-	-
3:0	TI1SEL	RW	4’h0	TI1 输入选择（tim_ti1[15:0] input） 0000：TIMx_CH1 0001：tim_ti1_in1 0010：tim_ti1_in2 0011：tim_ti1_in3 其他：保留

23.3.18 TIM15 备用功能选项寄存器 1（TIMx_AF1）

Address offset:0x60

Reset value:0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	Res				BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		RW	RW	RW	RW	RW					RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13	BKCMP4P	RW	0	tim_brk_cmp4 输入极性 该位选择 tim_brk_cmp4 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp4 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp4 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
12	BKCMP3P	RW	0	tim_brk_cmp3 输入极性 该位选择 tim_brk_cmp3 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp3 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp3 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
11	BKCMP2P	RW	0	tim_brk_cmp2 输入极性 该位选择 brk_cmp2 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp2 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp2 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
10	BKCMP1P	RW	0	tim_brk_cmp1 输入极性 该位选择 brk_cmp1 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp1 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效) 1：tim_brk_cmp1 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
9	BKINP	RW	0	BKIN 输入极性（TIMx_BKIN input polarity） 该位选择 BKIN 输入极性的备用功能，必须与 BKP 极性位同时配置 0：BKIN 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：BKIN 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
8:5	保留	-	-	-
4	BKCMP4E	RW	0	tim_brk_cmp4 输入使能（tim_brk_cmp4 enable） 该位用于刹车输入时使能 tim_brk_cmp4。tim_brk_cmp4 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp4 输入关闭 1：tim_brk_cmp4 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
3	BKCMP3E	RW	0	tim_brk_cmp3 输入使能（tim_brk_cmp3 enable）

				该位用于刹车输入时使能 tim_brk_cmp3。tim_brk_cmp3 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp3 输入关闭 1：tim_brk_cmp3 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
2	BKCMP2E	RW	0	tim_brk_cmp2 输入使能（tim_brk_cmp2 enable） 该位用于刹车输入时使能 tim_brk_cmp2。tim_brk_cmp2 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp2 输入关闭 1：tim_brk_cmp2 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
1	BKCMP1E	RW	0	tim_brk_cmp1 输入使能（tim_brk_cmp1 enable） 该位用于刹车输入时使能 tim_brk_cmp1。tim_brk_cmp1 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp1 输入关闭 1：tim_brk_cmp1 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
0	BKINE	RW	1	BKIN 输入使能（TIMx_BKIN input enable） 该位用于刹车输入时使能 BKIN 的备用功能。BKIN 输入与其它刹车源进行或逻辑作为刹车输入。 0：BKIN 输入关闭 1：BKIN 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改。

23.3.19 TIM15 备用功能选项寄存器 2（TIMx_AF2）

Address offset:0x64

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	-
18:16	OCRSEL	RW	3’h0	OCREF 复位源选择（ocref_clr source selection） 该位段用于选择 ocref_clr 输入源 0000：ocref_clr0 0001：ocref_clr1 0010：ocref_clr2 0011：ocref_clr3

				其它：保留 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
15:0	保留	-	-	-

23.3.20 TIM15 DMA 控制寄存器（TIMx_DCR）

Address offset:0x3DC

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12:8	DBL	RW	5’h0	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 TIMx_DMAR 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 .... 11001：26 次传输 例：我们考虑这样的传输：DBL=7 字节，DBA=TIMx_CR1 - 如果 DBL=7 字节，DBA=TIMx_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址）+ DBA +（DMA 索引)，其中 DMA 索引 = DBL 其中(TIMx_CR1 的地址）+ DBA 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(TIMx_CR1 的地址）+ DBA 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。
7:5	保留	-	-	-
4:0	DBA	RW	5’h0	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 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， …….

23.3.21 TIM15 连续模式的 DMA 地址（TIMx_DMAR）

Address offset:0x3E0

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[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
DMAB[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DMAB	RW	0	DMA 连续传送寄存器（DMA register for burst accesses) 对 TIMx_DMAR 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： TIMx_CR1 地址 +（DBA + DMA 索引)x4 其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 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 TIMx_DCR 寄存器中定义的 DBL。

23.4 TIM16_TIM17 寄存器描述

23.4.1 TIM16_TIM17 控制寄存器 1（TIMx_CR1）

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DITHEN	UIFREMAP	Res	CKD[1:0]		ARPE	Res			OPM	URS	UDIS	CEN
			RW	RW		RW	RW	RW				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:13	保留	-	-	-
12	DITHEN	RW	0	抖动使能（Dithering enbale） 0：抖动关闭 1：抖动打开 注：该位只能在 CEN 位复位时才能修改
11	UIFREMAP	RW	0	UIF 状态位重映射使能（UIF status bit remapping） 0：无重映射，UIF 状态位不复制到 TIMx_CNT 寄存器的第 31 位 1：重映射，UIF 状态位复制到 TIMx_CNT 寄存器的第 31 位
10	保留	-	-	-
9:8	CKD	RW	2’h0	时钟分频因子（Clock division) 这 2 位定义在定时器时钟(CK_INT)频率和死区时间和死区发生器与数字滤波器(ETR,TIx)所用的采样时钟（tDTS）之间的分频比例。 00：tDTS = tCK_INT 01：tDTS = 2 x tCK_INT 10：tDTS = 4 x tCK_INT

				11：保留，不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	保留	-	-	-
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

23.4.2 TIM16_TIM17 控制寄存器 2（TIMx_CR2）

Address offset:0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						OIS1N	OIS1	Res				CCDS	CCUS	Res	CCPC
						RW	RW					RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	OIS1N	RW	0	输出空闲状态 1(OC1N 输出）(Output Idle state 1) 0：当 MOE=0 时，死区后 OC1N=0；

				1：当 MOE=0 时，死区后 OC1N=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
8	OIS1	RW	0	输出空闲状态 1(OC1 输出)（Output Idle state 1) 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0； 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
7:4	保留	-	-	-
3	CCDS	RW	0	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。
2	CCUS	RW	0	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(CCPC=1)，只能通过设置 COM 位更新它们； 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 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。
1	保留	-	-	-
0	CCPC	RW	0	捕获/比较预装载控制位（Capture/compare preloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新（COMG 位设置或 tim_trgi 上检测到的上升沿，具体取决于 CCUS 位）。 注：该位只对具有互补输出的通道起作用。

23.4.3 TIM16_TIM17 DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1DE	UDE	BIE	Res	COMIE	Res			CC1IE	UIE
						RW	RW	RW		RW				RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	CC1DE	RW	0	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。。
8	UDE	RW	0	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。
7	BIE	RW	0	允许刹车中断（Break interrupt enable) 0：禁止刹车中断； 1：允许刹车中断。

6	保留	-	-	-
5	COMIE	RW	0	允许 COM 中断（COM interrupt enable) 0：禁止 COM 中断; 1：允许 COM 中断。
4:2	保留	-	-	-
1	CC1IE	RW	0	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
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断; 1：允许更新中断。

23.4.4 TIM16_TIM17 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res	BIF	Res	COMIF	Res			CC1IF	UIF
						RC_W0		RC_W0		RC_W0				RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	CC1OF	RC_W0	0	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 TIMx_CCR1 寄存器时，CC1IF 的状态已经为‘1’。
8	保留	-	-	-
7	BIF	RC_W0	0	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。如果 TIM_DIER 的 BIE=1 则产生中断
6	保留	-	-	-
5	COMIF	RC_W0	0	COM 中断标记（COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新)该位由硬件置‘1’。它由软件清‘0’。 0：无 COM 事件产生; 1：COM 中断等待响应。
4:2	保留	-	-	-
1	CC1IF	RC_W0	0	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： 0：无匹配发生; 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。

				当 TIMx_CCR1 的内容大于 TIMx_APR 的内容时，在溢出时，CC1IF 位变高 如果通道 TI1 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 TIMx_CCR1 清‘0’。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=0，当重复计数器=0 时产生更新事件)。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

23.4.5 TIM16_TIM17 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								BG	Res	COMG	Res			CC1G	UG
								W		W				W	W

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	BG	W	0	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 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
6	保留	-	-	-
5	COMG	W	0	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 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 注：该位只对拥有互补输出的通道有效。
4:2	保留	-	-	-
1	CC1G	W	0	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 1 上产生一个捕获/比较事件： 若通道 1 配置为输出：

				设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	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 TIMx_ARR 的值。

23.4.6 TIM16_TIM17 捕获/比较模式控制寄存器 1（TIMx_CCMR1）

Address offset:0x18

Reset value：0x0000 000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															OC1M[3]
															RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								ICIF[3:0]				ICIPSC[1:0]			
								RW	RW	RW	RW	RW	RW	RW	RW

23.4.6.1 输出比较模式

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	OC1M[3]	RW	0	详见 OC1M[2:0]描述
15:8	保留	-	-	-
7	OC1CE	RW	0	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。
6:4	OC1M	RW	3'h0	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 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)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 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_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 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平,否则为无效电平; 在向下计数时,一旦 TIMx_CNT>TIMx_CCR1 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 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平,否则为有效电平; 在向下计数时,一旦 TIMx_CNT>TIMx_CCR1 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(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=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。
3	OC1PE	RW	0	输出比较 1 预装载使能（Output Compare 1 preload enable) 0: 禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1: 开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 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(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=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(TIMx_CR1 寄存器的 OPM=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。
2	OC1FE	RW	0	输出比较 1 快速使能（Output Compare 1 fast enable) 该位用于加快输出对触发输入事件的响应。必须在单脉冲模式 (TIMx_CR1 寄存器的 OPM 位置 1）中使用，当触发到来时实现脉冲快速输出。 0：根据计数器与 CCR1 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象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。仅能用于 PWM 模式 1 和 PWM 模式 2
1:0	CC1S	RW	2’h0	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； 其他：保留 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

23.4.6.2 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:4	IC1F	RW	4’h0	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 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8

				0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC	RW	2'h0	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 CC1S=00，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。
1:0	CC1S	RW	2'h0	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 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

23.4.7 TIM16_TIM17 捕获/比较使能寄存器（TIMx_CCER）

Address offset:0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												CC1NP	CC1NE	CC1P	CC1E
												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3	CC1NP	RW	0	输入/捕获 1 互补输出极性（Capture/Compare 1 complementary output polarity) 0：OC1N 高电平有效； 1：OC1N 低电平有效。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出)则该位不能被修改。
2	CC1NE	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。
1	CC1P	RW	0	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 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 不反相（门控模式、编码器模式）。 01：反相/下降沿： TIxFP1 下降沿有效（捕获、复位模式下触发、外部时钟或触发模式下）； TIxFP1 反相（门控模式、编码器模式）。 10：保留，不要使用这个配置。 11：不反相/双沿 TIxFP1 上升和下降沿都有效 (捕获、复位模式下触发、外部时钟或触发模式下)； TIxFP1 不反相（门控模式）。这个配置不能应用于编码器模式下。 注： 1.对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。 2.一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2，则该位不能被修改。
0	CC1E	RW	0	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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 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 TIMx_CCR1 寄存器。 0：捕获禁止； 1：捕获使能。 注： 对于互补输出通道，这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。

表 23-5 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	
MOE 位	**OSSI 位**	**OSSR 位**	**CCxE 位**	**CCxNE 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=OCREF xor CCxNP, OCxN_EN=1

		0	1	0	OCxREF+极性， OCx=OCREF xor CCxP, 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（取反）+极性+死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开） OCx=CCxP, OCx=0 OCx_EN=0	输出禁止（与定时器断开） OCxN=CCxNP, OCx=0 OCxN_EN=0
		1	0	1	关闭状态(输出使能且为无效电平) OCx=CCxP, OCx_EN=1	OCxREF+极性， OCxN=OCREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+极性， OCx=OCREF xor CCxP, OCx_EN=1	关闭状态（输出使能且为无效电平） OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF+极性+死区，， OCx_EN=1	OCxREF（取反）+极性+死区， OCxN_EN=1
0	0	x	x	x	输出禁止（与定时器断开）	
	1		0	0		
	1		0	1	关闭状态（输出使能但为无效电平） 异步的：OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCx_EN=1;（如果刹车或刹车 2 触发了） 然后若时钟存在（仅刹车 1 触发时有效，不需要刹车 2）：经过一个死区时间后 OCx=OISx, OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平 注：刹车 2 仅用于 OSSI=OSSR=1	
	1		1	0		
	1		1	1		

特别的，当死区有效时，输出总是按照死区内输出的规则输出，尽管此时 moe 可能已经被软件或硬件的方式置为 1。

如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

23.4.8 TIM16_TIM17 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	UIFCPY	R	0	该位为只读寄存器，将 TIMx_ISR 寄存器的 UIF 位复制过来。如果 TIMx_CR1 中的 UIF 重映射位被复位了，该 bit 位保留并读出来为 0
30:16	保留	-	-	-
15:0	CNT	RW	16'h0	计数器的值（Counter value)

23.4.9 TIM16_TIM17 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16’h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清’0’或被工作在复位模式的从控制器清’0’。

23.4.10 TIM16_TIM17 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	ARR	RW	20’hFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。 无抖动模式（DITHEN=0）： 该寄存器保持自动加载值 抖动模式（DITHEN=1）： 该寄存器保持整数部分 ARR[19:4]，ARR[3:0]包含抖动部分

23.4.11 TIM16_TIM17 重复计数寄存器（TIMx_RCR）

Address offset:0x30

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	REP	RW	8’h0	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。 每次向下计数器达到 0，会产生一个更新事件并且重复计数器重新从 REP 值开始计数。对 TIMx_RCR 寄存器写入的新值只在下次更新事件发生时才起作用。

23.4.12 TIM16_TIM17 捕获/比较寄存器 1（TIMx_CCR1）

Address offset:0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CCR1[19:16]			
												RW/RO			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	CCR1	RW/RO	20'h0	捕获/比较通道 1 的值（Capture/Compare 1 value) 若 CC1 通道配置为输出: CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 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 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 无抖动模式（DITHEN=0）： 该寄存器保持 CCR1[15:0]的比较值，CCR1[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保持整数部分 CCR1[19:4]，CCR1[3:0]包含抖动部分 若 CC1 通道配置为输入: CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。 TIMx_CCR1 寄存器只读 无抖动模式（DITHEN=0）： 该寄存器保存捕获值到 CCR1[15:0]，CCR1[19:16]位被复位 抖动模式（DITHEN=1）： 该寄存器保存捕获值到 CCR1[19:4]，CCR1[3:0]位被复位

23.4.13 TIM16_TIM17 死区时间寄存器 2（TIMx_DTR2）

Address offset:0x54

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	DTPE	RW	0	死区时间预加载使能（Deadtime preload enable） 0：死区时间的值不可以预加载 1：死区时间的值可以预加载 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
16	DTAE	RW	0	死区时间不对称使能（Deadtime asymmetric enable） 0：上升沿和下降沿的死区时间相同，都由 DTG[7:0]配置 1：上升沿死区时间由 DTG[7:0]配置，下降沿死区时间由 DTGF[7:0]配置 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，2，3，则该位不能被修改。
15:8	保留	-	-	-
7:0	DTGF[7:0]	RW	8'h0	下降沿死区时间生成器（Dead-time falling edge generator setup） 该位段定义了插入互补输出时下降沿的死区时间，假设 DT 表示其持续时间： DTGF[7:5]=0xx => DTF=DTGF[7:0] × Tdtg，Tdtg = TDTS； DTGF[7:5]=10x => DTF=(64+DTGF[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTGF[7:5]=110 => DTF=(32+DTGF[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTGF[7:5]=111 => DTF=(32+DTGF[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位

23.4.14 TIM16_TIM17 刹车和死区寄存器（TIMx_BDTR）

Address offset:0x44

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			BKBID	Res	BKDSRM	Res						BKF[3:0]			
			RW		RW							RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28	BKBID	RW	0	刹车双向性（Break bidirectional） 0：刹车输入 BRK 在输入模式中 1：刹车输入 BRK 在双向模式中

				在双向模式中（BKBID 设置为 1），刹车输入被既配置为输入模式也是开漏输出模式。任何有效的刹车事件会将刹车输入电平拉低，向外部设备指示内部发生了刹车事件 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
27	保留	-	-	-
26	BKDSRM	RW	0	刹车解除（Break disarm） 0：刹车输入 BRK 启用 1：刹车输入 BRK 解除 当没有刹车源时该位会被硬件清零 BKDSRM 位必须由软件置位来释放双向输出控制（开漏输出高阻状态），然后轮询它直到被硬件复位，表示故障状态已经消失。 注：任何对该位的写操作，都会延迟一个 APB 时钟生效。
25:20	保留	-	-	-
19:16	BKF[3:0]	RW	4’h0	刹车滤波（Break filter） 这些位定义了对 BRK 信号采样的频率和对 BRK 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，BRK2 是异步的 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2， N=6 0101：采样频率 fSAMPLING=fDTS/2， N=8 0110：采样频率 fSAMPLING=fDTS/4， N=6 0111：采样频率 fSAMPLING=fDTS/4， N=8 1000：采样频率 fSAMPLING=fDTS/8， N=6 1001：采样频率 fSAMPLING=fDTS/8， N=8 1010：采样频率 fSAMPLING=fDTS/16， N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8 注：一旦 LOCK 级别 1(TIMx_BDTR 寄存器中的 LOCK 位)被设置，则该位不能被修改。
15	MOE	RW	0	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 AOE 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(TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 TIMx 捕获/比较使能寄存(TIMx_CCER)。
14	AOE	RW	0	自动输出使能（Automatic output enable) 0：MOE 只能被软件置’1’； 1：MOE 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(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。
13	BKP	RW	0	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(TIMx_BDTR 寄存器中的 LOCK 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。
12	BKE	RW	0	刹车功能使能（Break enable) 0：禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1：开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，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。
11	OSSR	RW	0	运行模式下"关闭状态"选择（Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下"关闭状态"选择（Off-state selection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(TIMx 捕获/比较使能寄存器(TIMx_CCER))。 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 CCxE=1 或 CCxNE=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(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOCK	RW	2'h0	锁定设置（Lock 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 TIMx_BDTR 寄存器的 DTG、BKBID、BK2BID、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 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 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 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 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。
7:0	DTG	RW	8'h0	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:

				DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg，Tdtg = TDTS； DTG[7:5]=10x => DT=(64+DTG[5:0]）× Tdtg，Tdtg = 2 × TDTS； DTG[7:5]=110 => DT=(32+DTG[4:0]）× Tdtg，Tdtg = 8 × TDTS； DTG[7:5]=111 => DT=(32+DTG[4:0])× Tdtg，Tdtg = 16 × TDTS； 例：若 TDTS = 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(TIMx_BDTR 寄存器中的 LOCK 位)设为 1、2 或 3，则不能修改这些位。

23.4.15 TIM16_TIM17 输入选择寄存器（TIMx_TISEL）

Address offset:0x5C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												TI1SEL[3:0]			
												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3:0	TI1SEL	RW	4'h0	TI1 输入选择（tim_ti1[15:0] input） 0000：TIMx_CH1 0001：保留 0010：tim_ti1_in2 0011：tim_ti1_in3 0100：tim_ti1_in4 0101：tim_ti1_in5 其他：保留

23.4.16 TIM16_TIM17 备用功能选项寄存器 1（TIMx_AF1）

Address offset:0x60

Reset value:0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	Res				BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		RW	RW	RW	RW	RW					RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13	BKCMP4P	RW	0	tim_brk_cmp4 输入极性 该位选择 tim_brk_cmp4 输入极性，必须与 BKP 极性位同时配置

				0：tim_brk_cmp4 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：tim_brk_cmp4 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
12	BKCMP3P	RW	0	tim_brk_cmp3 输入极性 该位选择 tim_brk_cmp3 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp3 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：tim_brk_cmp3 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
11	BKCMP2P	RW	0	tim_brk_cmp2 输入极性 该位选择 brk_cmp2 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp2 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：tim_brk_cmp2 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
10	BKCMP1P	RW	0	tim_brk_cmp1 输入极性 该位选择 brk_cmp1 输入极性，必须与 BKP 极性位同时配置 0：tim_brk_cmp1 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：tim_brk_cmp1 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
9	BKINP	RW	0	BKIN 输入极性（TIMx_BKIN input polarity） 该位选择 BKIN 输入极性的备用功能，必须与 BKP 极性位同时配置 0：BKIN 输入极性不翻转（BKP=0 时低有效，BKP=1 时高有效） 1：BKIN 输入极性翻转（BKP=0 时高有效，BKP=1 时低有效） 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
8:5	保留	-	-	-
4	BKCMP4E	RW	0	tim_brk_cmp4 输入使能（tim_brk_cmp4 enable） 该位用于刹车输入时使能 tim_brk_cmp4。tim_brk_cmp4 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp4 输入关闭 1：tim_brk_cmp4 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
3	BKCMP3E	RW	0	tim_brk_cmp3 输入使能（tim_brk_cmp3 enable）

				该位用于刹车输入时使能 tim_brk_cmp3。tim_brk_cmp3 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp3 输入关闭 1：tim_brk_cmp3 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
2	BKCMP2E	RW	0	tim_brk_cmp2 输入使能（tim_brk_cmp2 enable） 该位用于刹车输入时使能 tim_brk_cmp2。tim_brk_cmp2 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp2 输入关闭 1：tim_brk_cmp2 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
1	BKCMP1E	RW	0	tim_brk_cmp1 输入使能（tim_brk_cmp1 enable） 该位用于刹车输入时使能 tim_brk_cmp1。tim_brk_cmp1 输入与其它刹车源进行或逻辑作为刹车输入。 0：tim_brk_cmp1 输入关闭 1：tim_brk_cmp1 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
0	BKINE	RW	1	BKIN 输入使能（TIMx_BKIN input enable） 该位用于刹车输入时使能 BKIN 的备用功能。BKIN 输入与其它刹车源进行或逻辑作为刹车输入。 0：BKIN 输入关闭 1：BKIN 输入开启 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改。

23.4.17 TIM16_TIM17 备用功能选项寄存器 2（TIMx_AF2）

Address offset:0x64

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	-
18:16	OCRSEL	RW	3'h0	OCREF 复位源选择（ocref_clr source selection） 该位段用于选择 ocref_clr 输入源 0000：ocref_clr0 0001：ocref_clr1 0010：ocref_clr2 0011：ocref_clr3 其它：保留

				注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)被设置为 1，则该位不能被修改
15:0	保留	-	-	-

23.4.18 TIM16_TIM17 option 寄存器（TIMx_OR1）

Address offset:0x68

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															HSE32EN
															RW

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	HSE32EN	RW	0	HSE32 分频连接 tim_ti1_in3 使能 0：不使能 1：使能

23.4.19 TIM16_TIM17 DMA 控制寄存器（TIMx_DCR）

Address offset:0x3DC

Reset value:0x0000 000

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12:8	DBL	RW	5'h0	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 TIMx_DMAR 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 .... 11010：27 次传输 例：我们考虑这样的传输：DBL=7 字节，DBA=TIMx_CR1 - 如果 DBL=7 字节，DBA=TIMx_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址）+ DBA +（DMA 索引)，其中 DMA 索引 = DBL 其中(TIMx_CR1 的地址) + DBA 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(TIMx_CR1 的地址）+ DBA 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。
7:5	保留	-	-	-
4:0	DBA	RW	5'h0	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 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， …….

23.4.20 TIM16_TIM17 连续模式的 DMA 地址（TIMx_DMAR）

Address offset:0x3E0

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[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**
DMAB[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DMAB	RW	0	DMA 连续传送寄存器（DMA register for burst accesses) 对 TIMx_DMAR 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 TIMx_CR1 地址 +（DBA + DMA 索引)x4 其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 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 TIMx_DCR 寄存器中定义的 DBL。

24. 基本定时器（TIM6 和 TIM7）

24.1 简介

基本定时器 TIM6 和 TIM7 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。

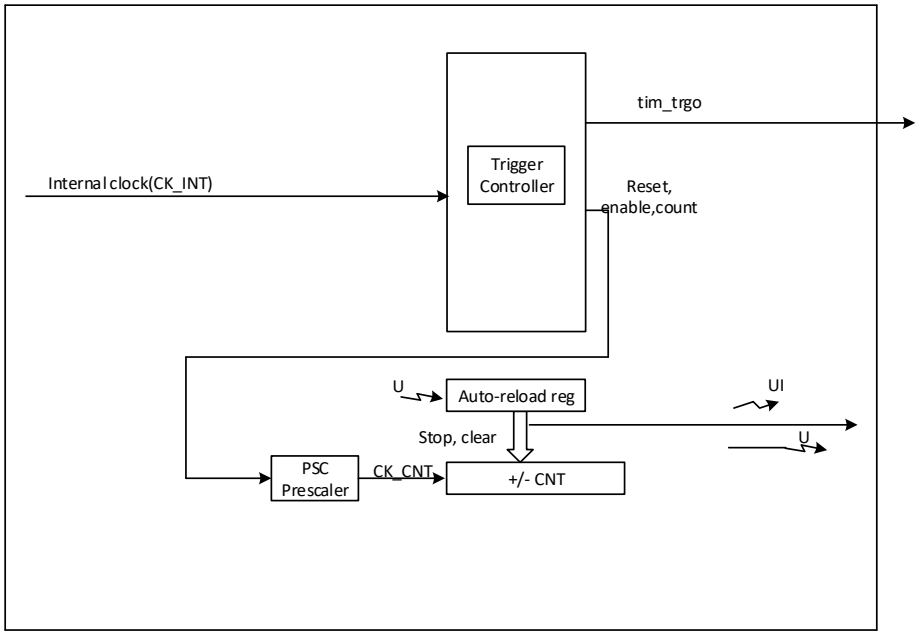
TIM6 和 TIM7 是完全独立的，它们不共享任何资源。

24.1.1 TIM6 和 TIM7 主要特征

TIM6 和 TIM7 定时器的主要功能包括：

- 16 位自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1～65535 之间的任意数值
- 触发 DAC 的同步电路
- 在更新事件（计数器溢出）发生时产生中断/DMA

24.1.2 模块框图



24.2 TIMx 功能描述

24.2.1 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。计数器的时钟可以被预分频器分频。

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(TIMx_CNT)
- 预分频器寄存器（TIMx_PSC）
- 自动装载寄存器（TIMx_ARR）

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它的预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)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 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK_CNT 驱动，仅当设置了 TIMx_CR1 寄存器中的计数器使能位(CEN)，CK_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx_CR 寄存器的 CEN 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 TIMx_PSC 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。新的预分频参数将在下一次更新事件到来时被采用。

下面几张图给出了在预分频器运行时，更改计数器参数的例子。

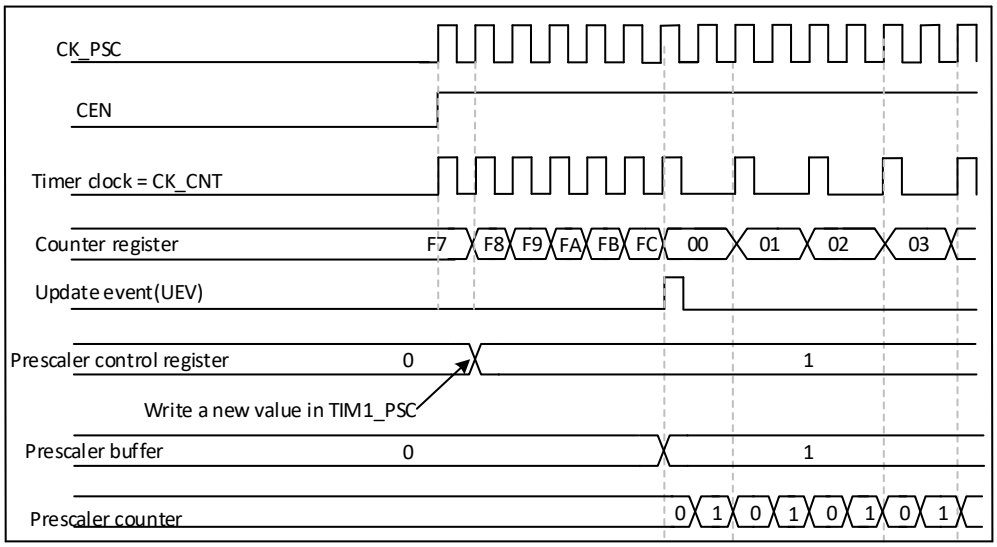


图 24-1 当预分频器的参数从 1 变到 2 时，计数器的时序图

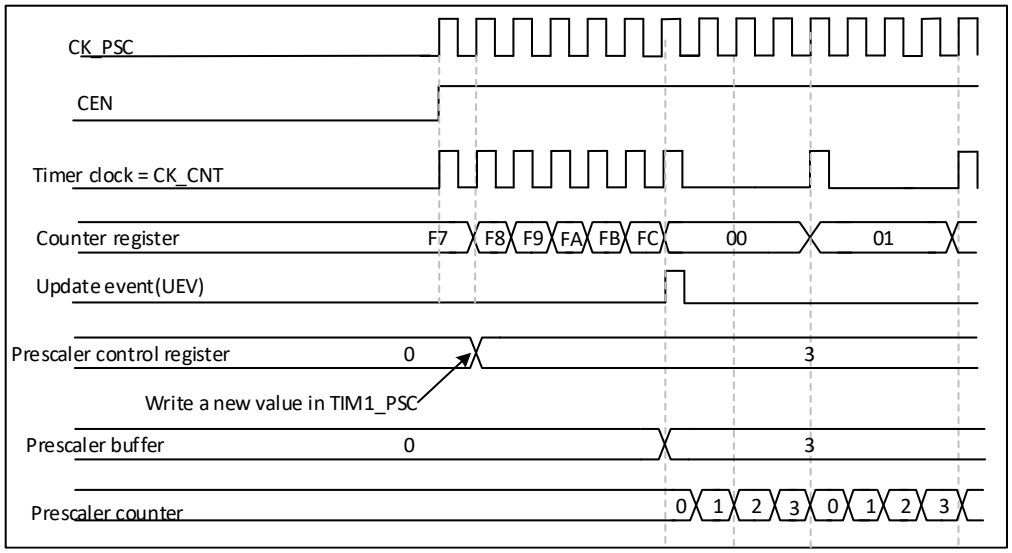


图 24-2 当预分频器的参数从 1 变到 4 时，计数器的时序图

24.2.2 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(TIMx_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

而在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx_CR1 寄存器中的 UDIS 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 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

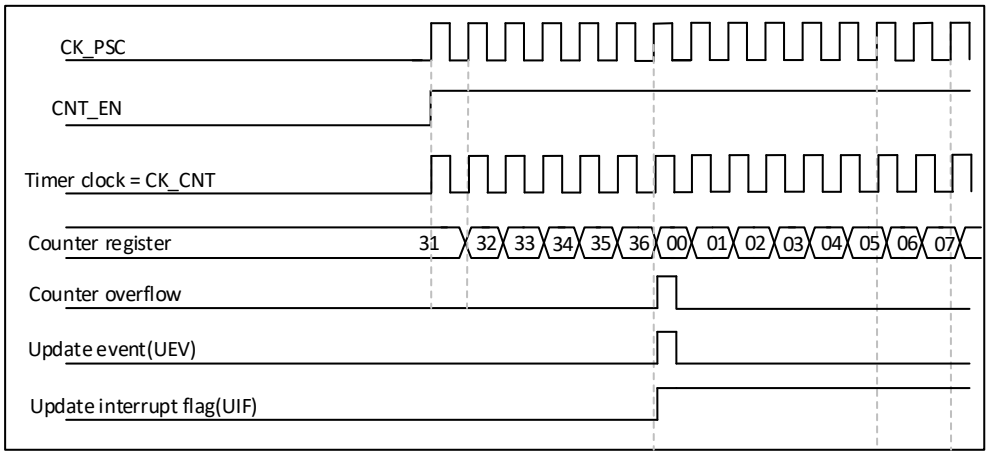


图 24-3 计数器时序图，内部时钟分频因子为 1

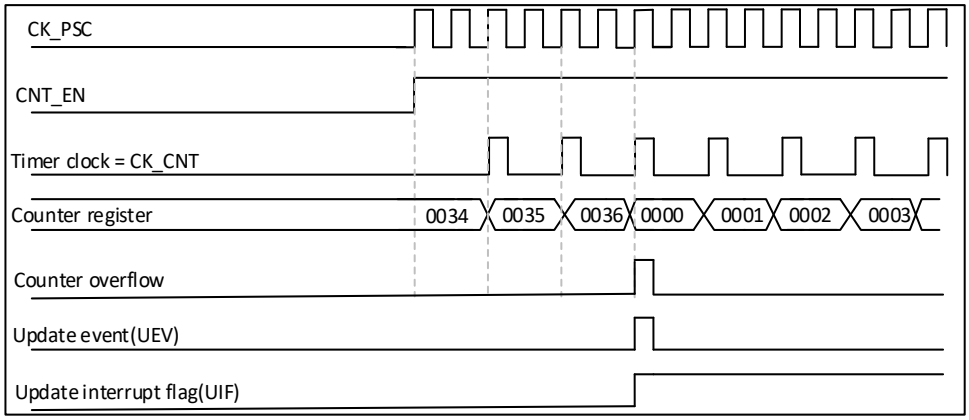


图 24-4 计数器时序图，内部时钟分频因子为 2

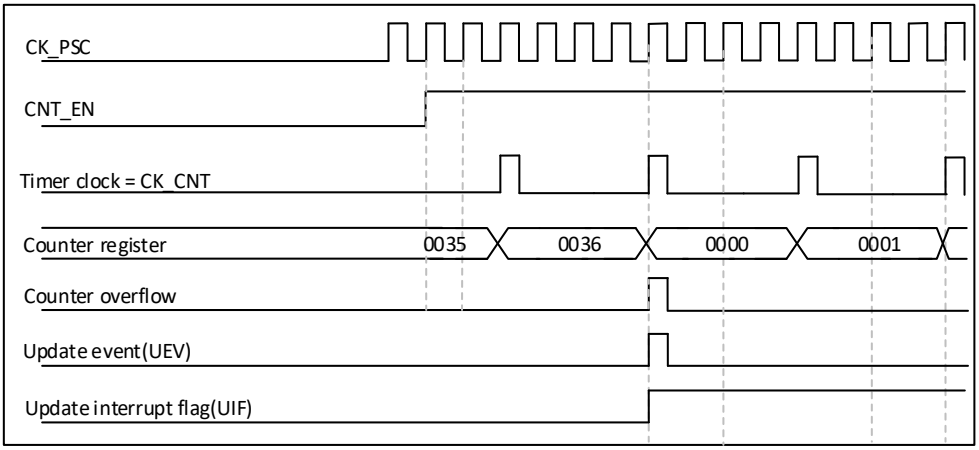


图 24-5 计数器时序图，内部时钟分频因子为 4

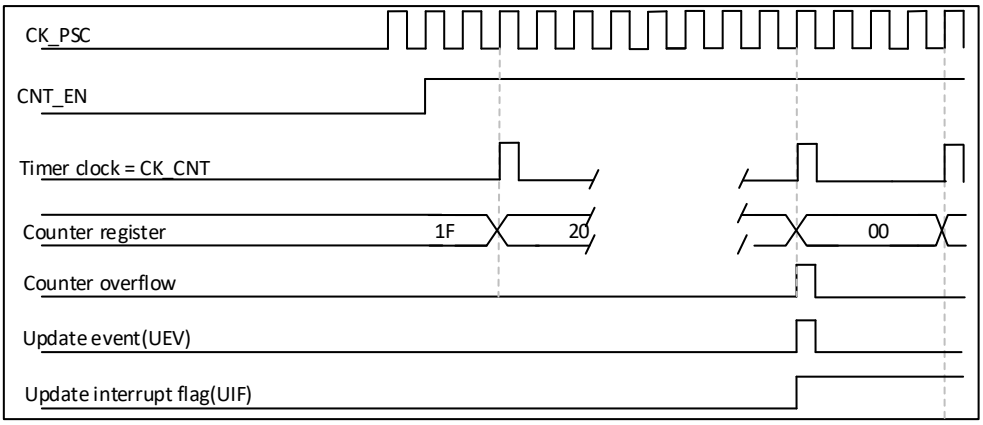


图 24-6 计数器时序图，内部时钟分频因子为 N

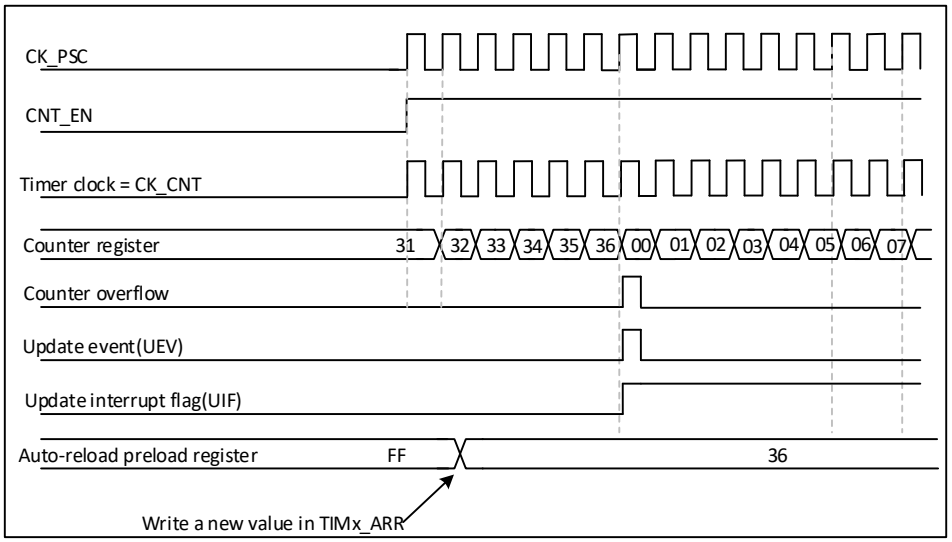


图 24-7 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx_ARR)

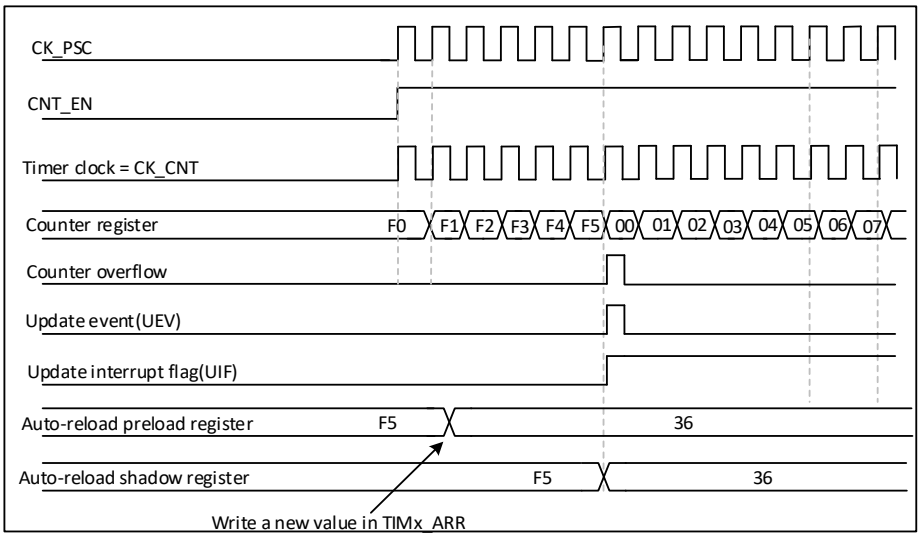


图 24-8 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx_ARR)

24.2.3 调试模式

当微控制器进入调试模式时(Cortex-M4F 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 DBG 节。

24.3 寄存器描述

24.3.1 TIM6 和 TIM7 控制寄存器 1（TIMx_CR1）

Address offset:0x00

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	ARPE	RW	0	自动重装载预装载允许位（Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	保留	-	-	-
3	OPM	RW	0	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 CEN 位)时，计数器停止。
2	URS	RW	0	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 UG 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。
1	UDIS	RW	0	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 UG 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。（译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 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。
0	CEN	RW	0	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 CEN 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 CEN 位。

24.3.2 TIM6 和 TIM7 控制寄存器 2（TIMx_CR2）

Address offset:0x04

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6:4	MMS	RW	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 – TIMx_EGR 寄存器的 UG 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 CEN 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 TIMx_SMCR 寄存器中 MSM 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。 注意：从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。
3:0	保留	-	-	-

24.3.3 TIM6 和 TIM7DMA/中断使能寄存器（TIMx_DIER）

Address offset:0x0C

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	UDE	RW	0	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。
7:1	保留	-	-	-
0	UIE	RW	0	允许更新中断（Update interrupt enable) 0：禁止更新中断; 1：允许更新中断。

24.3.4 TIM6 和 TIM7 状态寄存器（TIMx_SR）

Address offset:0x10

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															UIF
															RC_W0

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	UIF	RC_W0	0	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 TIMx_CR1 寄存器的 UDIS=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 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 − 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考从模式控制寄存器(TIMx_SMCR))。

24.3.5 TIM6 和 TIM7 事件产生寄存器（TIMx_EGR）

Address offset:0x14

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	UG	W	0	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)。

24.3.6 TIM6 和 TIM7 计数器（TIMx_CNT）

Address offset:0x24

Reset value：0x0000 0000

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

RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CNT	RW	16’h0	计数器的值（Counter value)

24.3.7 TIM6 和 TIM7 预分频器（TIMx_PSC）

Address offset:0x28

Reset value：0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PSC	RW	16’h0	预分频器的值（Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 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 TIM_EGR 的 UG 位清’0’或被工作在复位模式的从控制器清’0’。

24.3.8 TIM6 和 TIM7 自动重装载寄存器（TIMx_ARR）

Address offset:0x2C

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	ARR	RW	16’hFFFF	自动重装载的值（Prescaler value) 自动重装载的值为空时，计数器不工作。

25. 低功耗定时器(LPTIM)

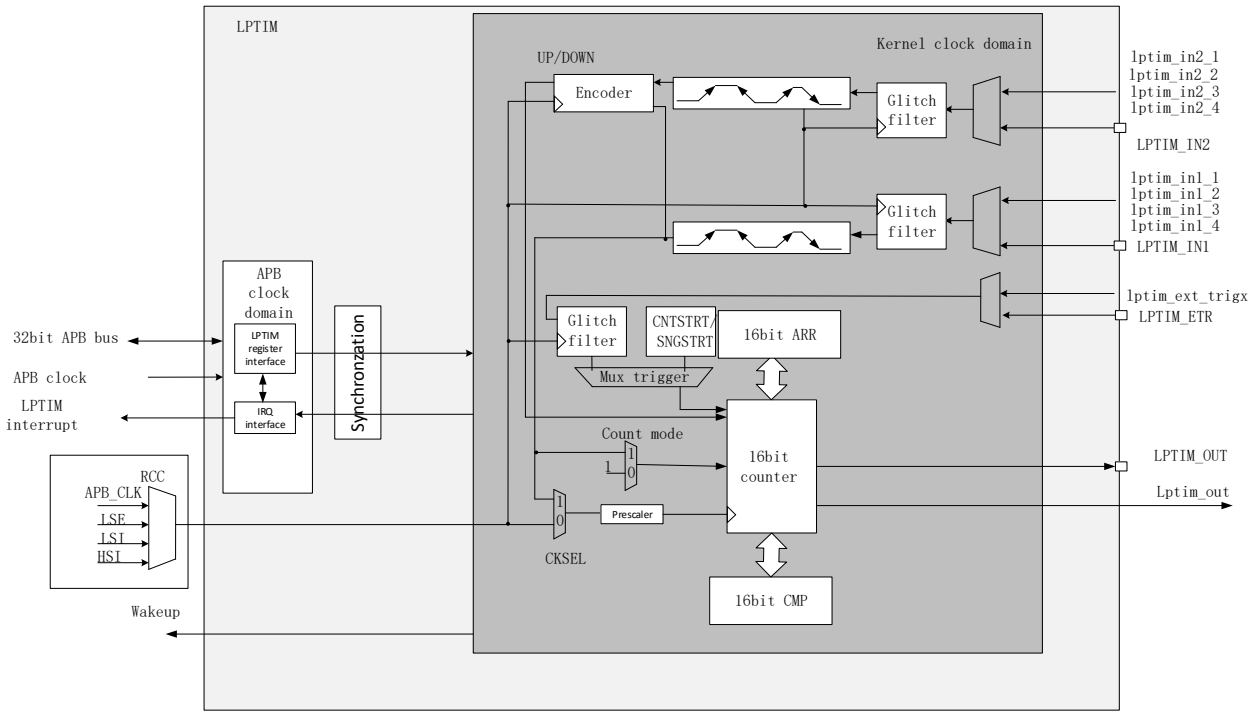
25.1 简介

PTIM 是一个 16 位定时器。由于其时钟源的多样性，LPTIM 能够在除 standby 模式之外的所有功耗模式下保持运行。考虑到 LPTIM 即使没有内部时钟源也能运行，可以用作“脉冲计数器”，这在某些应用中很有用。LPTIM 引入了一种灵活的时钟方案，可提供所需的功能和性能，同时将功耗降至最低。

25.1.1 主要特性

- 16 位向上计数器
- 3 位预分频器，具有 8 个可能的分频因子（1，2，4，8，16，32，64，128）
- 可选时钟
  - 内部时钟源：APB 时钟或其他任何振荡器（见 RCC 章节）
  - 通过 LPTIM 输入的外部时钟源（在没有运行嵌入式振荡器的情况下工作，由脉冲计数器应用程序使用）。
- 16 位 ARR 可重载寄存器
- 16 位比较寄存器
- 连续/单次模式
- 可选软件/硬件输入触发
- 可配置数字毛刺滤波器
- 可配置输出：Pulse、 PWM
- 可配置 IO 极性
- 编码器模式

25.1.2 模块框图



25.2 外部信号说明

25.2.1 LPTIM 输入和触发映射

表 25-1 LPTIM 外部触发连接

TRIGSEL	外部触发
lptim_ext_trig0	GPIO
lptim_ext_trig1	RTC_ALARM
lptim_ext_trig2	COMP1_OUT
lptim_ext_trig3	COMP2_OUT
lptim_ext_trig4	COMP3_OUT
lptim_ext_trig5	COMP4_OUT

表 25-2 LPTIM 输入 1 连接

lptim_in1_mux	LPTIM 输入 1 连接
lptim_in1_0	GPIO
lptim_in1_1	COMP1
lptim_in1_2	COMP3

表 25-3 LPTIM 输入 2 连接

lptim_in2_mux	LPTIM 输入 1 连接
lptim_in2_0	GPIO
lptim_in2_1	COMP2
lptim_in2_2	COMP4

25.3 LPTIM 功能描述

25.3.1 复位和时钟

通过 RCC 模块，可以使用内部时钟信号对 LPTIM 进行时钟控制。可以使用多个时钟源为 LPTIM 提供时钟。它可以使用内部时钟信号进行计时，该信号可以是 PCLK（APB 时钟）或任何其他可通过 RCC 选择的振荡器。此外，可以通过外部时钟源 Input1 为 LPTIM 提供时钟。当使用外部时钟源计数时，LPTIM 可能以下列两种配置之一运行：

- 第一种配置是 LPTIM 由外部信号计时，但同时内部时钟信号从 PCLK 或任何其他振荡器提供给 LPTIM。
- 第二种配置是当 LPTIM 通过其外部 Input1 由外部时钟源单独提供时钟。该配置用于在进入低功耗模式后关闭所有振荡器时实现超时功能或脉冲计数器功能。

CKSEL 位进行编程可以控制 LPTIM 是使用外部时钟源还是内部时钟源。当配置为使用外部时钟源时，CKPOL 位用于选择外部时钟信号有效边沿。

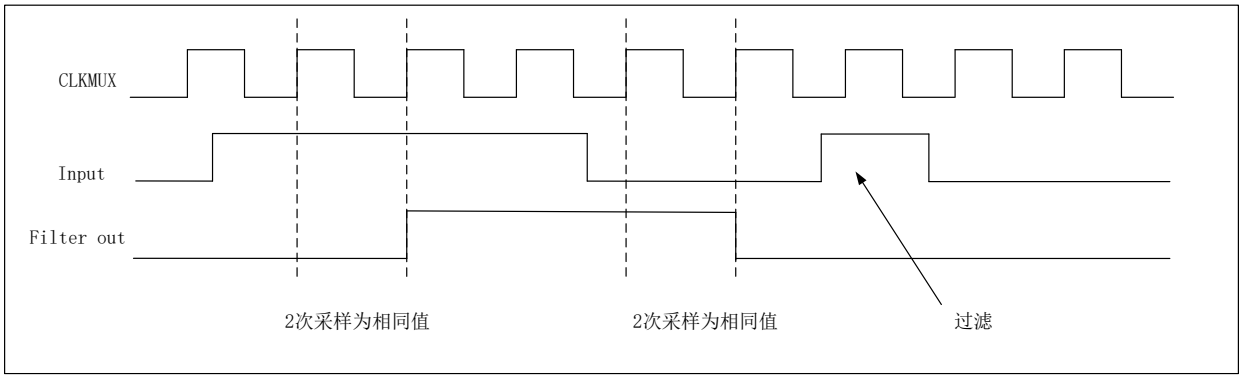
25.3.2 毛刺滤波器

LPTIM 输入，无论是外部（映射到 GPIO）还是内部（在芯片级映射到其他嵌入式外围设备），都受到数字滤波器的保护，可防止任何毛刺和噪声扰动在 LPTIM 内部传播。这是为了防止虚假计数或触发。在激活数字滤波器之前，应首先向 LPTIM 提供一个内部时钟源。这是保证过滤器正常运行所必需的。数字滤波器分为两组：

- 第一组数字滤波器保护 LPTIM 外部输入。数字滤波器灵敏度由 CKFLT 位控制
- 第二组数字滤波器保护 LPTIM 内部触发输入。数字滤波器灵敏度由 TRGFLT 位控制。

注意：数字滤波器灵敏度由组控制。不可能在同一组内单独配置每个数字滤波器灵敏度。

滤波器灵敏度作用于 LPTIM 某一输入上检测到的连续相等样本的数量，以将信号电平变化视为有效转换。下图显示了在编程 2 个连续样本的情况下毛刺滤波器行为的示例。



如果没有提供内部时钟信号，则必须通过将 CKFLT 和 TRGFLT 位设置为“0”来停用数字滤波器。在这种情况下，可以使用外部模拟滤波器来保护 LPTIM 外部输入免受毛刺干扰。

25.3.3 预分频

LPTIM 16 位计数器，由一个可配置的 2 次方预分频器控制驱动。预分频器分频比由 LPTIM_CFGR.PRESC[2:0]控制。

下表列出了所有情况：

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

25.3.4 触发选择

LPTIM 计数器可以通过软件启动，也可以在检测到 6 个（可选最大输入个数）触发输入之一的有效边沿后启动。

TRIGEN[1:0]用于确定 LPTIM 触发源：

当 TRIGEN[1:0]等于“00”时，一旦 CNTSTRT 或 SNGSTRT 位之一由软件设置，LPTIM 计数器就会启动。TRIGEN[1:0]的三个剩余可能值用于配置触发输入使用的有效边沿。一旦检测到有效边沿，LPTIM 计数器就会启动。

当 TRIGEN[1:0]不是‘00’时，TRIGSEL[2:0]用于选择触发输入中的哪一个用于启动计数器。

外部触发器被视为 LPTIM 的异步信号。因此，在触发检测之后，由于同步，在定时器开始运行之前需要两个计数器时钟周期的延迟。

如果在定时器已经启动时发生新的触发事件，它将被忽略（除非启用了 timeout 功能）。

在设置 SNGSTRT/CNTSTRT 位之前必须使能定时器。当定时器被禁用时，对这些位的任何写入都将被硬件丢弃。

当通过软件启动计数器时（TRIGEN[1:0] = 00），LPTIM_CR 寄存器更新（设置 SNGSTRT 或 CNTSTRT 位之一）和计数器有效启动之间有 3 个计数时钟周期的延迟。

25.3.5 操作模式

LPTIM 具有两种如下工作模式：

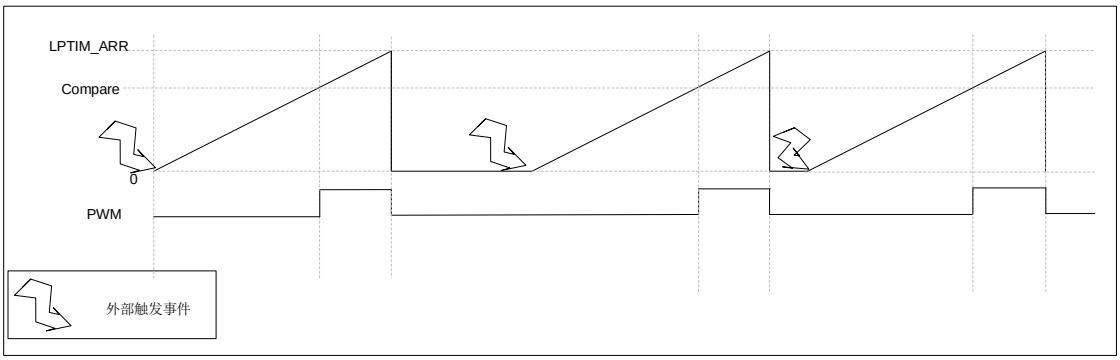
- 连续计数模式：计时器自由运行，从触发事件开始运行，直到计时器被 disable 掉才停止。
- 单次计数模式：定时器从一个触发事件开始，当达到 ARR 值时停止。

25.3.5.1 单次计数模式

要使能单次计数，LPTIM_CR.SNGSTRT 位必须置 1。

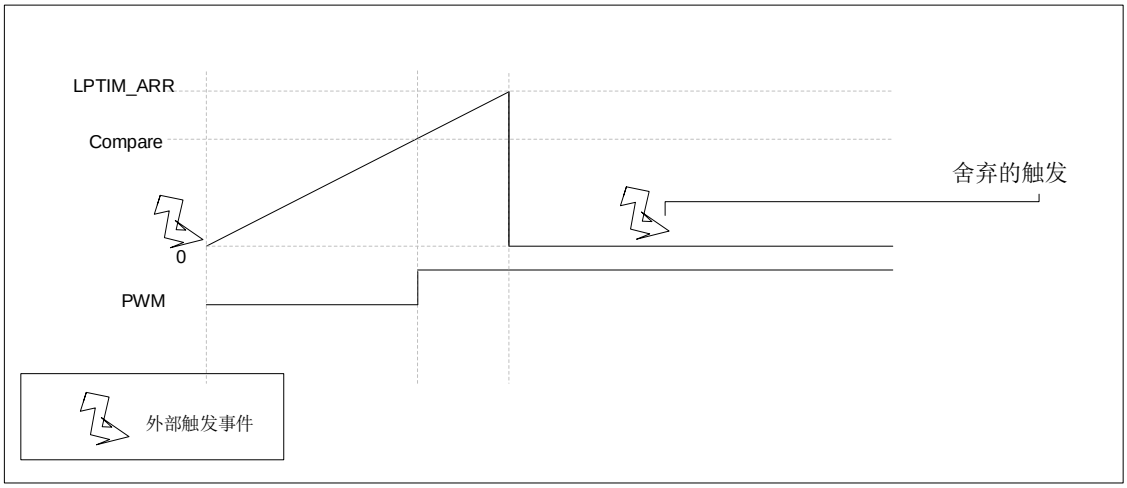
如果选择外部触发，则在设置 SNGSTRT 位和计数器停止（包含零值）后到达的外部事件，将启动计数器进行新的单次计数周期，如下图所示。

一个新的触发事件将重新启动计时器。在计数器启动之后，到达 ARR 之前，任何触发事件都将被忽略。



-激活一次设置模式

当 LPTIM_CFGR.WAVE 位置 1 时，一次设置模式被激活。这种情况下，计数器仅在第一次触发后启动一次，随后的任何触发事件均被忽略。如下图所示：



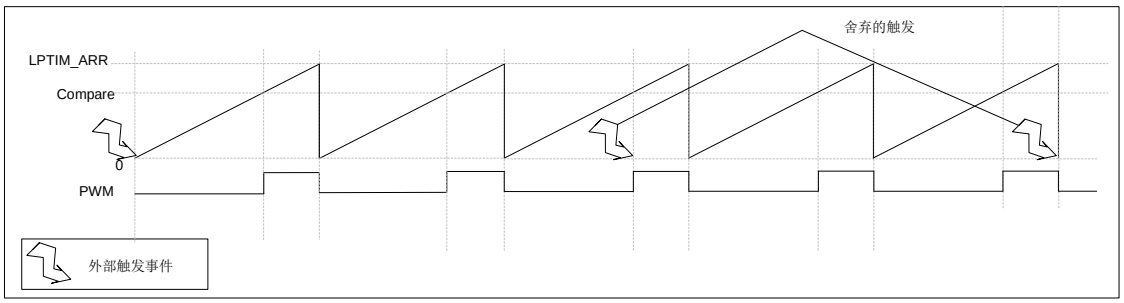
在软件启动(TRIGEN[1:0]=‘00’)的情况下，SNGSTRT 置位将启动计数器进行单次计数。

25.3.5.2 连续计数模式

要使能连续计数，LPTIM_CR.CNTSTRT 位必须置 1。

如果选择外部触发，则在 CNTSTRT 置位后到达的外部触发事件将启动计数器进行连续计数。任何后续的外部触发事件都将被丢弃，如下图所示。

在软件启动(TRIGEN[1:0]=‘00’)的情况下，设置 CNTSTRT 将启动计数器以进行连续计数。



LPTIM_CR.SNGSTRT 和 LPTIM_CR.CNTSTRT 位只能在定时器使能时置位 (LPTIM_CR.ENABLE 为“1”）。可以快速的从单次模式转变为连续模式：

- 如果之前选择了连续模式，则置位 LPTIM_CR.SNGSTRT 会将 LPTIM 切换到单次模式。计数器（如果开始计数）一到达 ARR 就会停止。

- 如果先前选择了单次模式，则置位 LPTIM_CR.CNTSTRT 会将 LPTIM 切换到连续模式。计数器（如果开始计数）一到达 ARR 就会重新启动。

25.3.5.3 波形产生

两个 16 位寄存器，LPTIM_ARR（自动重载寄存器）和 LPTIM_CMP（比较寄存器），用于在 LPTIM 输出上生成几种不同的波形。

定时器可以生成以下波形：

- PWM 模式：LPTIM 输出置位条件为 LPTIM_CNT 中的计数器值超过了 LPTIM_CMP 中的比较值。一旦 LPTIM_ARR 和 LPTIM_CNT 寄存器之间发生匹配，LPTIM 输出就会复位。
- 单脉冲模式：输出波形类似于第一个脉冲的 PWM 模式，然后输出永久复位。
- 一次设置模式：输出波形类似于单脉冲模式，不同点在于输出会保持在最后一个信号电平（取决于输出配置的极性）。

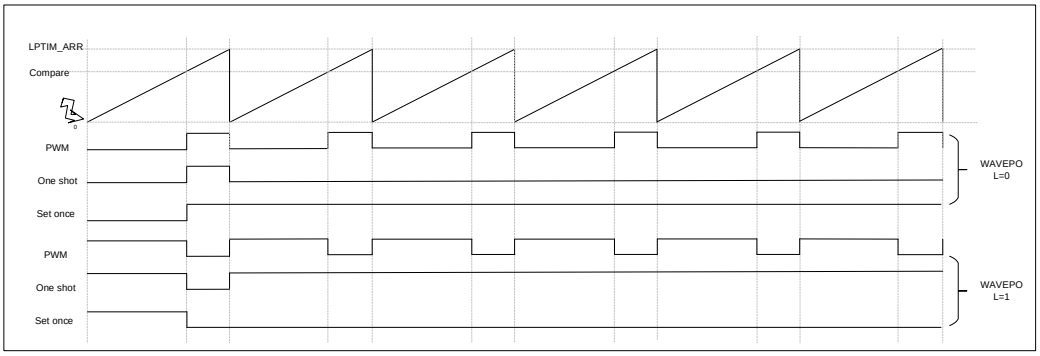
上述模式要求 LPTIM_ARR 寄存器值严格大于 LPTIM_CMP 寄存器值。

LPTIM 输出波形可通过 WAVE 位进行如下配置：

- 将 WAVE 位重置为“0”会强制 LPTIM 生成 PWM 波形或单脉冲波形，具体取决于设置的位：CNTSTRT 或 SNGSTRT。
- 将 WAVE 位设置为“1”会强制 LPTIM 生成一次设置模式波形。

WAVPOL 位控制 LPTIM 输出极性。更改立即生效，因此，在重新配置极性后，甚至在启用定时器之前，输出默认值将立即改变。

可以产生频率高达 LPTIM 时钟频率除以 2 的信号。下图显示了可在 LPTIM 输出上生成的三种可能波形。此外，它还显示了使用 WAVPOL 位改变极性的效果。



25.3.6 寄存器更新

LPTIM_ARR 寄存器和 LPTIM_CMP 寄存器在 APB 总线写操作后立即更新，或者如果定时器已经启动，则在当前计数周期结束时更新。PRELOAD 位控制 LPTIM_ARR 寄存器的更新方式：

当 PRELOAD 位重置为“0”时，LPTIM_ARR 寄存器在任何写访问后立即更新。

当 PRELOAD 位设置为“1”时，如果定时器已经启动，LPTIM_ARR 寄存器将在当前计数周期结束时更新。

LPTIM APB 接口和 LPTIM 内核逻辑使用不同的时钟，因此在 APB 写入的时刻和这些值可用于计数器比较器的时刻之间存在一些延迟。在此延迟期内，必须避免对这些寄存器进行任何额外的写入。

LPTIM_ISR 寄存器中的 ARROK 标志和 CMPOK 标志分别指示对 LPTIM_ARR 寄存器和 LPTIM_CMP 寄存器的写操作完成标志。

在写入 LPTIM_ARR 寄存器或 LPTIM_CMP 寄存器后，只有在前一个写入操作完成后，才能对同一寄存器执行新的写入操作。在 ARROK 或 CMPOK 标志位置位之前的任何连续写入都将导致不可预知的结果。

25.3.7 计数器模式

LPTIM 计数器可用于对 LPTIM Input1 上的外部事件进行计数，也可用于对内部时钟周期进行计数。CKSEL 位控制哪个源将用于更新计数器。

如果 LPTIM 配置为对 Input1 上的外部事件进行计数，则可以根据写入 CKPOL[1:0]位的值在上升沿、下降沿或双沿更新计数器。

根据 CKSEL 的值，可选择以下计数模式：

CKSEL=0：LPTIM 由内部时钟源计时

LPTIM 时钟由内部时钟源提供，LPTIM 计数器配置为在每个内部时钟脉冲后更新。

CKSEL=1：LPTIM 由外部时钟源提供时钟。

在此配置中，LPTIM 不需要内部时钟源（此模式不支持时钟毛刺滤波）。在 LPTIM 外部 Input1 上注入的信号用作 LPTIM 的时钟。此配置适用于未启用嵌入式振荡器的操作模式。

对于此配置，LPTIM 计数器可以在 input1 时钟信号的上升沿或下降沿更新，但不能同时在上升沿和下降沿更新。由于在 LPTIM 外部 Input1 上注入的信号也用于为 LPTIM 内核逻辑提供时钟，因此在计数器递增之前存在一些初始延迟（在启用 LPTIM 之后）。

25.3.8 定时器使能

LPTIM_CR.ENABLE 位用于使能/禁用 LPTIM 内核逻辑。置位 ENABLE 位后，需要延迟两个计数器时钟才能使能 LPTIM。

仅当 LPTIM 禁用时，才能修改 LPTIM_CFGR 和 LPTIM_IER 寄存器。

25.3.8.1 编码器模式

该模式允许处理来自用于检测旋转元件角位置的正交编码器的信号。编码器接口模式简单地充当具有方向选择的外部时钟。这意味着计数器只是在 0 和 ARR 值之间连续计数（0 到 ARR 或 ARR 到 0，具体取决于方向）。因此必须在启动前配置 LPTIM_ARR。从两个外部输入信号 Input1 和 Input2 生成一个时钟信号来为 LPTIM 计数器提供时钟。这两个信号之间的相位决定了计数方向。

编码器模式仅在 LPTIM 由内部时钟源计数时可用。Input1 和 Input2 输入上的信号频率不得超过 LPTIM 内部时钟的 1/4。这是强制性要求以保证 LPTIM 的正常运行。

方向更改由 LPTIM_ISR 寄存器中的两个向下和向上标志指示。此外，如果启用 DOWNIE/UPIE 位，则可以为两个方向更改事件生成中断。

要激活编码器模式，必须将 ENC 位设置为“1”。LPTIM 必须首先配置为连续模式。

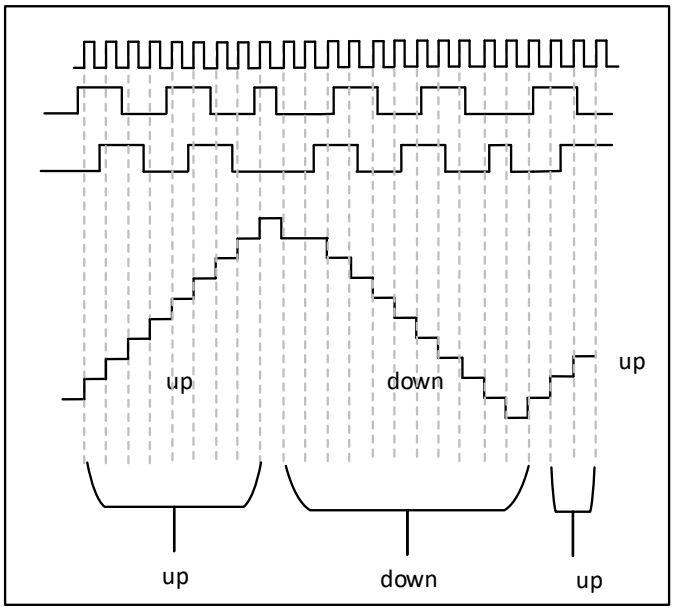
当编码器模式处于活动状态时，LPTIM 计数器会根据增量编码器的速度和方向自动修改。因此，它的内容总是代表编码器的位置。由 UP 和 DOWN 标志指示的计数方向对应于编码器转子的旋转方向。

根据 CKPOL[1:0]位配置的边沿灵敏度，可能会出现不同的计数情况。下表总结了可能的组合，假设 Input1 和 Input2 不同时切换。

有效沿	反向信号电平 (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
上升沿	高	减	-	加	-
	低	加	-	减	-
下降沿	高	-	加	-	减
	低	-	减	-	加
双沿	高	减	加	加	减
	低	加	减	减	加

下图显示了配置双沿灵敏度的编码器模式的计数序列。

在此模式下，LPTIM 必须由内部时钟源提供时钟，因此 CKSEL 位必须保持其复位值 0。此外，预分频器分频比必须等于其复位值 1（PRESC[2:0]位必须为“000”）。



25.3.9 Debug 模式

当芯片进入 debug 模式，取决于 DBG 模块的 DBG_LPTIM_STOP 位的设定，LPTIM 或者继续正常工作，或者停止工作。

25.3.10 LPTIM 低功耗模式

模式	描述
Sleep	无影响。LPTIM 中断（使能后）会退出 sleep 模式.
Low-power run	无影响。
Low-power sleep	无影响。LPTIM 中断导致设备退出低功耗睡眠模式。
Stop0/Stop1	当 LPTIM 由 LSE 或 LSI 计时时没有影响。LPTIM 中断导致设备退出 Stop 0 和 Stop 1。
Standby Shutdown	LPTIM 外设断电，在退出待机或关机模式后必须重新初始化。

注：在使用 lptim 唤醒时，需要确保两次唤醒的间隔时间大于 6 个 PCLK 周期，否则将导致不可预知的结果。

25.3.11 LPTIM 中断

如果下列事件在 LPTIM_IER 寄存器内使能，则这些事件将生成中断/唤醒事件：

- 比较匹配
- 自动重新加载匹配（如果是编码器模式，则无论方向如何）
- 外部触发事件
- 自动重载寄存器写入完成
- 比较寄存器写入完成
- 方向改变（编码器模式），可编程（上/下/两者）。

注意:如果在 LPTIM_ISR 寄存器（状态寄存器）中的相应标志置 1 后，LPTIM_IER 寄存器（中断使能寄存器）中的相应位才被置 1，则不会产生中断。

中断事件	描述
比较匹配	当计数器寄存器（LPTIM_CNT）的内容与比较寄存器（LPTIM_CMP）的内容匹配时，会产生中断标志。
自动重载匹配	当计数器寄存器的内容(LPTIM_CNT)与自动重新加载寄存器的内容匹配(LPTIM_ARR)，中断标志置位
外部触发事件	当检测到外部触发事件时，中断标志被置位
自动重载寄存器写入完成	当对 LPTIM_ARR 寄存器的写操作完成时，会产生中断标志。
比较寄存器写入完成	当对 LPTIM_CMP 寄存器的写操作完成时，会产生中断标志。
方向改变	在编码器模式下使用。嵌入了两个中断标志以指示方向改变： — UP 标志指示递增计数方向改变 — DOWN 标志表示减计数方向改变

25.4 寄存器描述

25.4.1 LPTIM 中断和状态寄存器，Address=0x00(LPTIM_ISR)

地址偏移：0x000

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
									R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	DOWN	R	0	计数器由向上计数变为向下计数 在编码器模式下，DOWN 位由硬件设置用于通知应用程序计数器已从向上计数变为向下计数。可以通过向 LPTIM_ICR 寄存器中的 DOWNCF 位写入 1 来清除 DOWN 标志。 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
5	UP	R	0	计数器由向下计数变为向上计数

				在编码器模式下，UP 位由硬件设置用于通知应用程序计数器已向下计数变为向上计数。可以通过向 LPTIM_ICR 寄存器中的 UPCF 位写入 1 来清除 UP 标志。注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
4	ARROK	R	0	自动重载寄存器更新 OK。 ARROK 由硬件置位，用于通知应用程序 APB 总线对 LPTIM_ARR 的写操作已成功完成。向 LPTIM_ICR.ARROKCF 写入 1 可清除 ARROK 标志.
3	CMPOK	R	0	比较寄存器更新 OK CMPOK 由硬件置位，用于通知应用程序对 LPTIM_CMP 寄存器的 APB 总线写操作已成功完成。
2	EXTTRIG	R	0	外部触发边沿事件 EXTTRIG 由硬件置位，用于通知应用程序所选外部触发输入上已出现有效边沿。如果因为计时器已经启动而忽略了触发器，则不会设置此标志。可以通过向 LPTIM_ICR 寄存器中的 EXTTRIGCF 位写入 1 来清除 EXTTRIG 标志。
1	ARRM	R	0	自动重载匹配。 ARRM 由硬件置位，用于通知应用程序 LPTIM_CNT 寄存器值匹配 LPTIM_ARR 寄存器的值。向 LPTIM_ICR.ARRMCF 位写入 1 可清除 ARRM 标志。
0	CMPM	R	0	比较匹配 CMPM 位由硬件置位，用于通知应用程序 LPTIM_CNT 寄存器值达到 LPTIM_CMP 寄存器的值。向 LPTIM_ICR.CMPMCF 位写入 1 可清除 ARRM 标志。

25.4.2 LPTIM 中断清零寄存器，Address=0x04(LPTIM_ICR)

地址偏移：0x004

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res									DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
									W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31：7	保留	-	-	-
6	DOWNCF	W	0	计数方向改变为向下清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 DOWN 标志。 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
5	UPCF	W	0	方向改变为向上清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 UP 标志。 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
4	ARROKCF	W	0	自动重载寄存器更新 OK 清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 ARROK 标志
3	CMPOKCF	W	0	比较寄存器更新 OK 清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 CMPOK 标志

2	EXTTRIGCF	W	0	外部触发有效沿清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 EXTTRIG 标志
1	ARRMCF	W	0	自动重载匹配清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 ARRM 标志
0	CMPMCF	W	0	比较匹配清除标志 向该位写入 1 会清除 LPTIM_ISR 寄存器中的 CMPM 标志

25.4.3 LPTIM 中断使能寄存器，Address=0x08(LPTIM_IER)

地址偏移：0x008

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res									DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
									RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：7	保留	-	-	-
6	DOWNIE	RW	0	计数器由向上计数变为向下计数中断使能 0：禁用 DOWN 中断 1：使能 DOWN 中断 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
5	UPIE	RW	0	计数器由向下计数变为向上计数中断使能 0：禁用 UP 中断 1：使能 UP 中断 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
4	ARROKIE	RW	0	自动重载寄存器更新 OK 中断使能 0：禁用 ARROK 中断 1：使能 ARROK 中断
3	CMPOKIE	RW	0	比较寄存器更新 OK 中断使能 0：禁用 CMPOK 中断 1：使能 CMPOK 中断
2	EXTTRIGIE	RW	0	外部触发有效边沿中断使能 0：禁用 EXTTRIG 中断 1：使能 EXTTRIG 中断
1	ARRMIE	RW	0	自动重载匹配中断使能 0：禁用 ARRM 中断 1：使能 ARRM 中断
0	CMPMIE	RW	0	比较匹配中断使能 0：禁用 CMPM 中断 1：使能 CMPM 中断

25.4.4 LPTIM 配置寄存器，Address=0x0C(LPTIM_CFGR)

地址偏移：0x00C

复位值：0x00000000

配置该寄存器时，需要一次设置所需寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							ENC	Res	PRELOAD	WAVPOL	WAVE	Res	TRIGEN[1:0]		Res.
							RW		RW	RW	RW		RW	RW	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
RW	RW	RW		RW	RW	RW		RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-
24	ENC	RW	0	编码器模式使能 ENC 位控制编码器模式 0：禁用编码器模式 1：启用编码器模式 注意：如果 LPTIM 不支持编码器模式功能，则该位保留。
23	保留	-	-	-
22	PRELOAD	RW	0	寄存器更新模式。 预加载位控制 LPTIM_ARR 寄存器更新模式。 0:每次 APB 总线写访问后更新寄存器； 1:寄存器在当前 LPTIM 计数周期结束时更新；
21	WAVPOL	RW	0	波形极性 WAVPOL 位控制输出极性 0：LPTIM 输出反映了 LPTIM_CNT 和 LPTIM_CMP 寄存器之间的比较结果 1：LPTIM 输出反映了 LPTIM_CNT 和 LPTIM_CMP 寄存器之间比较结果的反
20	WAVE	RW	0	波形形状 WAVE 位控制输出形状 0：停用一次设置模式，PWM 还是单脉冲波形具体取决于定时器的启动方式，用于 PWM 的 CNTSTRT 或用于单脉冲波形的 SNGSTRT。 1：激活一次设置模式
19	保留	-	-	-
18:17	TRIGEN[1:0]	RW	2’h0	触发启用和极性 TRIGEN 位控制 LPTIM 计数器是否由外部触发启动。如果选择了外部触发选项，则触发有效边沿可以采用三种配置： 00：软件触发（计数启动由软件发起） 01：上升沿为有效沿 10：下降沿为有效沿 111：保留
16	保留	-	-	-
15:13	TRIGSEL[2:0]	RW	3’h0	触发选择器

				TRIGSEL 位在以下 6 个可用源中选择将作为 LPTIM 触发事件的触发源： 000：lptim_ext_trig0 001：lptim_ext_trig1 010：lptim_ext_trig2 011：lptim_ext_trig3 100：lptim_ext_trig4 101：lptim_ext_trig5
12	保留	-	-	-
11:9	PRESC[2:0]	RW	3'h0	时钟预分频器。 PRESC 位配置预分频器分频系数。它可以是下列分部中的一个因素: 000:/1； 001:/2； 010:/4； 011:/8； 100:/16； 101:/32； 110:/64； 111:/128；
8	保留	-	-	-
7:6	TRGFLT[1:0]	RW	2'h0	触发的可配置数字滤波器 TRGFLT 值设置当内部触发器上发生电平变化时应检测到的连续相等样本数，然后才将其视为有效电平转换。必须存在内部时钟源才能使用此功能 00：任何触发活动电平变化都被视为有效触发 01：触发有效电平变化必须稳定至少 2 个时钟周期才被认为是有效触发。 10：触发有效电平变化必须稳定至少 4 个时钟周期才被认为是有效触发。 11：触发有效电平变化必须稳定至少 8 个时钟周期才被认为是有效触发。
5	保留	-	-	-
4:3	CKFLT[1:0]	RW	2'h0	外部时钟的可配置数字滤波器 CKFLT 值设置在外部时钟信号上发生电平变化时应检测到的连续相等采样数，然后才将其视为有效电平转换。必须存在内部时钟源才能使用此功能 00：任何外部时钟信号电平变化都被视为有效跳变 01：外部时钟信号电平变化必须至少稳定 2 个时钟周期才被认为是有效跳变。 10：外部时钟信号电平变化必须至少稳定 4 个时钟周期才被认为是有效跳变。 11：外部时钟信号电平变化必须至少稳定 8 个时钟周期才被认为是有效跳变。
2:1	CKPOL[1:0]	RW	2'h0	时钟极性 如果 LPTIM 由外部时钟源提供时钟：

				当 LPTIM 由外部时钟源提供时钟时，CKPOL 位用于配置计数器使用的一个或多个有效边沿： 00：上升沿为计数有效沿 如果 LPTIM 配置为编码器模式(置位 ENC 位)，则编码器子模式 1 处于活动状态。 01：下降沿为计数有效沿 如果 LPTIM 配置为编码器模式(置位 ENC 位)，则编码器子模式 2 处于活动状态。 10：两个沿都是有效沿。当两个外部时钟信号沿都被视为有效边沿时，LPTIM 还必须由内部时钟源提供时钟，其频率至少等于外部时钟频率的四倍。 如果 LPTIM 配置为编码器模式(置位 ENC 位)，则编码器子模式 3 处于活动状态。 11：不允许
0	CKSEL	RW	0	时钟选择器 CKSEL 位选择 LPTIM 将使用的时钟源： 0：LPTIM 由内部时钟源（APB 时钟或任何嵌入式振荡器）提供时钟 1：LPTIM 由外部时钟源通过 LPTIM Input1 提供时钟

25.4.5 LPTIM 控制寄存器，Address=0x10(LPTIM_CR)

地址偏移：0x010

复位值：0x00000000

配置 RCC 模块的 LPTIM1EN/LPTIM2EN 后，紧接着编程 LPTIM_CFGR.TRIGEN，然后等待 5 个 kernal 时钟后才能配置该寄存器的 SNGSTRT 或者 CNTSTRT。

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

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	CNTSTRT	RW	0	定时器启动连续模式。 该位由软件置位，由硬件清零在软件启动(TRIGEN[1:0]=‘00’)的情况下，该位置 1 将启动连续模式下的 LPTIM。 如果禁用软件启动(TRIGEN[1:0])，设置此位，一旦检测到外部触发，将采用连续模式启动定时器。 如果在进行单次计数模式计数时该位被置 1，则定时器不会在下一个 LPTIM_ARR 和 LPTIM_CNT 寄存器匹配时停止。LPTIM 计数器保持在连续模式下计数。 注：仅当 LPTIM 使能时，此位才能置 1。它将由硬件自动清零。
1	SNGSTRT	RW	0	LPTIM 启动单次模式。 该位由软件置位，由硬件清零。该位置 1 将以单脉冲模式启动 LPTIM。 注：仅当 LPTIM 使能时，此位才能置 1。它将由硬件自动复位。

0	ENABLE	RW	0	LPTIM 使能。 使能位由软件置 1 和清零。 0:LPTIM 禁用。 1:LPTIM 使能。

25.4.6 LPTIM 比较寄存器，Address=0x14(LPTIM_CMP)

地址偏移：0x014

复位值：0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15：0	CMP	RW	16’h0	自动重新加载值。 ARR 是 LPTIM 的比较值。 注：当 LPTIM 使能后才能更新该寄存器。

25.4.7 LPTIM 自动重载寄存器，Address=0x18(LPTIM_ARR)

地址偏移：0x018

复位值：0x00000001

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15：0	ARR	RW	16’h1	自动重新加载值。 ARR 是 LPTIM 的自动重载值。 此值必须严格大于 CMP[15:0]值。 注：当 LPTIM 使能后才能更新该寄存器。

25.4.8 LPTIM 计数寄存器，Address=0x1C(LPTIM_CNT)

地址偏移：0x01C

复位值：0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15：0	CNT	R	16’h0	计数器值。

				当 LPTIM 以异步时钟运行时，读取 LPTIM_CNT 寄存器可能返回不可靠的值。因此在这种情况下，有必要执行两次连续的读访问并验证返回的两个值是否相同。 需要注意的是，对于可靠的 LPTIM_CNT 寄存器读访问，需要执行两次连续的读操作并比较，连续读取访问的值相等时可以认为读取访问是可靠的。

25.4.9 LPTIM 选择寄存器，Address=0x20(LPTIM_OR)

地址偏移：0x020

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										IN2[2:1]		IN1[2:1]		IN2[0]	IN1[0]
										RW		RW		RW	

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5:4	IN2[2:1]	RW	2'h0	LPTIM 输入 2 重映射扩展 00：COMP2 01：COMP4
3:2	IN1[2:1]	RW	2'h0	LPTIM 输入 1 重映射扩展 00：COMP1 01：COMP3
1	IN2[0]	RW	0	LPTIM 输入 2 重映射 1：连接到 IN2[2:1]指示的 COMP 输出 0：连接到 GPIO
0	IN1[0]	RW	0	LPTIM 输入 1 重映射 1：连接到 IN1[2:1]指示的 COMP 输出 0：连接到 GPIO

26. 红外接口(IRTIM)

芯片内集成为遥控而用的红外接口（IRTIM）。它可以与一个红外 LED 一起实现遥控的功能。它使用 TIM16 和 TIM17 的内部连接，如下图所示。

为产生红外遥控信号，必须打开 Infrared interface（红外接口），并且 TIM16 的 channel 1(TIM16_OC1)和 TIM17 channel 1（TIM17_OC1）要被适当的配置以产生正确的波形。

红外接收器可以很容易通过一个基本输入捕获模式实现。

所有标准的红外脉冲调制模式可以通过对两个定时器的输出比较通道进行编程而获得。

TIM17 被用来产生高频载波信号，而 TIM16 可以产生调制包络。

红外功能输出到 IR_OUT pin，这个功能的激活是通过使能 GPIO_AFRx 寄存器的相关复用功能位实现的。

LED 需要大的灌电流驱动能力（仅可以在 PB9 / PA13 引脚），这可以通过 SYSCFG_CFGR1 寄存器的 I2C1_FMP 位被激活，这样就足够支撑直接控制红外 LED 大的灌电流而用。

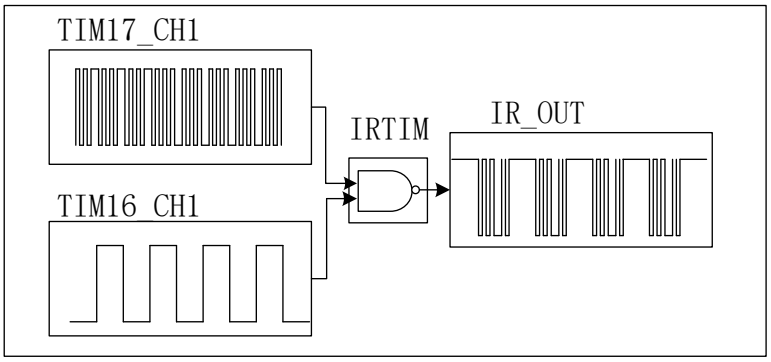


图 26-1 TRTIM 内部连接图

27. 实时时钟（RTC）

27.1 概述

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_BDCR 寄存器)处于备份域，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，对备份寄存器和 RTC 的访问被禁止，这是为了防止对备份域(BKP)的意外写操作。执行以下操作将使能对备份寄存器和 RTC 的访问：

- 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位，使能电源和备份接口时钟
- 设置寄存器 PWR_CR 的 DBP 位，使能对备份寄存器和 RTC 的访问。

27.2 主要特性

主要功能如下：

- 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)。
- 可以选择以下三种 RTC 的时钟源：
  - HSE 时钟的 128 分频；
  - LSE 振荡器时钟；
  - LSI 振荡器时钟
- 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)只能由备份域复位。
- 2 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。
  - RTC 全局中断，包含秒中断、闹钟中断、溢出中断。

27.3 功能描述

27.3.1 概述

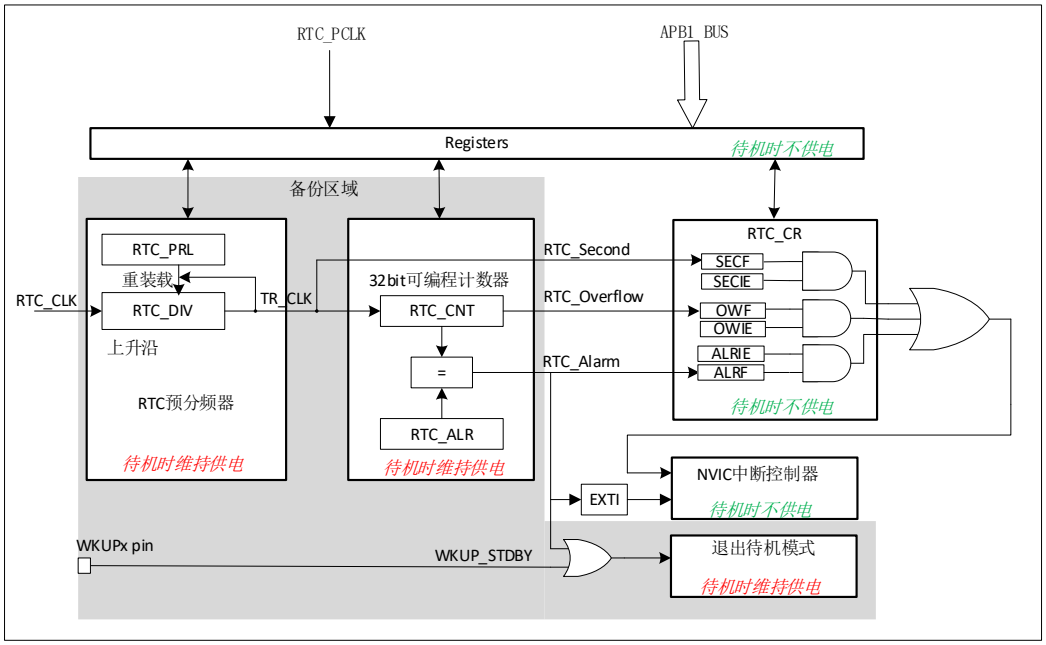


图 27-1 RTC 功能框图

RTC 由两个主要部分组成（参见上图）。第一部分（APB1 接口）用来和 APB1 总线相连。此单元还包含一组 32 位寄存器（RTC_CR，实际分散在两个地址的寄存器），可通过 APB1 总线对其进行读写操作。APB1 接口由 APB1 总线时钟驱动，用来连接 APB1 总线。

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 的预分频器模块，它可编程产生最长为 1 秒的 RTC 时间基准 TR_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_CR 寄存器中设置了相应的允许位，则在每个 TR_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 TR_CLK 周期累加并与存储在 RTC_ALR 寄存器中的可编程时间相比较，如果 RTC_CR 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。

闹钟中断作为待机模式和停止模式的唤醒信号。上图为待机模式的情况，从待机模式唤醒后。在停止模式，RTC 闹钟作为唤醒需要配置 EXTI line17 对应寄存器。

27.3.2 寄存器复位

RTC_PRL，RTC_ALR，RTC_CNT 和 RTC_DIV 寄存器仅能通过备份域复位信号复位。其他系统寄存器（RTC_CR）由系统复位或电源复位进行异步复位。

27.3.3 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 时钟的上升沿同步到 RTC APB1 时钟后的信号有效时更新。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

系统刚从停机模式唤醒（这种情况计数值只是不会更新，因为 CPU 不工作，系统时钟停止，但 RTC 正常计数，计数值不会同步到 V_{DDD} 区）

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_CR 寄存器中的 RSF 位（寄存器同步标志）被硬件置'1'。

说明：

1）CPU 可读的寄存器包括 RTC_CR，RTC_CNT，RTC_ALR 和 RTC_DIV；
2）RTC_CR 寄存器为 RTC_PCLK 域，CPU 任何时候读能读到稳定值；
3）RTC_CNT 和 RTC_DIV 来源于 RTC_CLK 域。在 RTC 工作后，RTC_DIV 寄存器在每个 RTC_CLK 的上升沿更新；RTC_CNT 和来源于 RTC_CLK 时钟域的标志位同样采用跟 RTC_DIV 寄存器同样的更新信号，虽然这样不是每次更新时 RTC_CNT 的值都改变；
4）RSF 实现在 RTC_PCLK 域，在 RTC_CLK 同步到 RTC_PCLK 后的脉冲信号有效时置位；
5）RSF 仅控制 RTC_CNT 和 RTC_DIV 的读取时机（硬件不会控制）

27.3.4 配置 RTC 寄存器

必须设置 RTC_CR 寄存器中的 CNF 位，使 RTC 进入配置模式后，才能写入 RTC_PRL、RTC_CNT、RTC_ALR 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_CR 寄存器中的 RTOFF 状态位，判断 RTC 寄存器是否处于更新中。仅当 RTOFF 状态位是'1'时，才可以写入 RTC 寄存器。

配置过程：

- 查询 RTOFF 位，直到 RTOFF 的值变为' 1'；（表明前面配置已经完成）
- 置 CNF 值为 1，进入配置模式；（此时 RTOFF 位仍为 1，保证 CPU 在写寄存器时 RTOFF=1）
- 对一个或多个 RTC 寄存器进行写操作；（RTOFF 在此过程仍为 1，RTC_CLK 域寄存器此步骤操作写入 buffer 寄存器）

- 清除 CNF 标志位，退出配置模式；(硬件检测到 CNF 清零后，开始执行上一步骤中的写寄存器操作，开始写入 RTC_CLK 域的寄存器；同时 RTOFF 被清零)
- 查询 RTOFF，直至 RTOFF 位变为’1’以确认写操作已经完成。(buffer 寄存器写入 RTC_CLK 域后置位 RTOFF)

注：仅当 CNF 标志位被清除时，写操作才能进行，这个过程至少需要 3 个 RTC_CLK 周期。(在清除 CNF 标志位后 3 个 RTC_CLK 不能重新启动配置，否则会出现配置错误（此时通过 RTOFF=0 控制）)

说明：

1）在此过程中，CPU 写寄存器时 RTOFF=1；
2）CPU 的写周期为从 CNF=1 到 CNF=0，配置其他寄存器在这两个操作之间；
3）先将 CNF 写 1 后将 CNF 清零，这个操作清零 RTOFF；只写 CNF=0 或者写 CNF=1 后不清零不会清零 RTOFF；
4）先将 CNF 写 1 后将 CNF 清零，这个操作启动 buffer 寄存器写 RTC_CLK 域寄存器的过程；
5）RTOFF 实现在 RTC_PCLK 域。

27.3.5 RTC 标志设置

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（SECF）。

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（OWF）。（达到 0 之前即为检测到 RTC_CNT 的最大值置位 OWF)

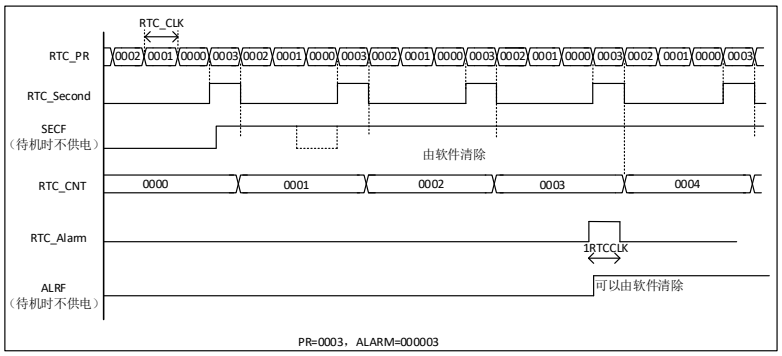
在计数器的值到达闹钟寄存器的值加 1（RTC_ALR+1）之前的 RTC 时钟周期中，设置 RTC_Alarm 和 RTC 闹钟标志(ALRF)。(检测到 RTC_CNT 计数到 RTC_ALR，且在最后一个 RTC_CLK 置位 ALRF)

对 RTC 闹钟寄存器（RTC_ALR）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 闹钟寄存器（RTC_ALR）和/或 RTC 计数器寄存器（RTC_CNT)。
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟寄存器（RTC_ALR）和/或 RTC 计数器寄存器（RTC_CNT)。

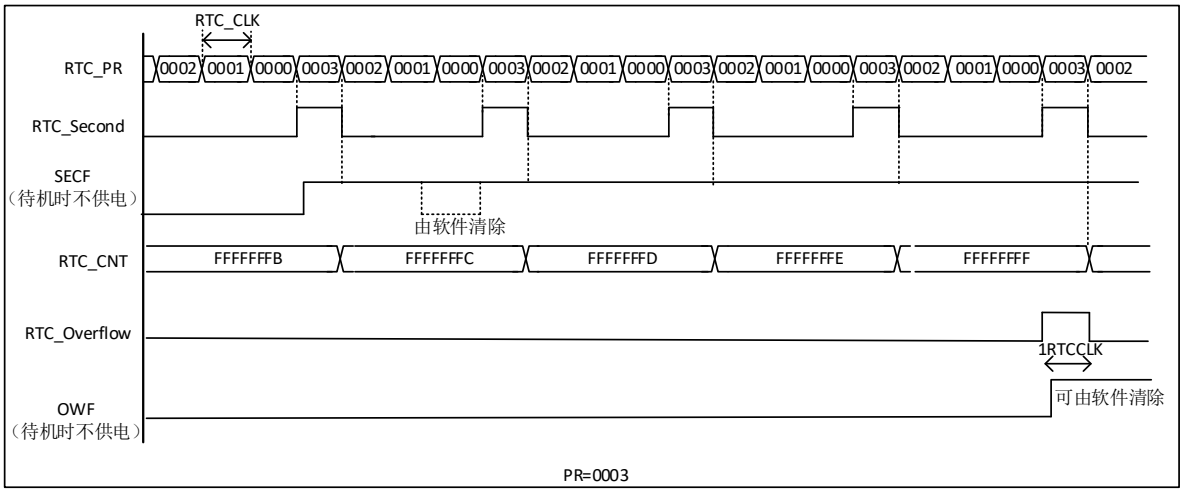
27.3.6 RTC 时序

(1) RTC 秒和闹钟时序



SECF 和 ALRF 为 RTC_PCLK 域信号，分别采样 RTC_CLK 域 RTC_Sectond 和 RTC_Alarm 信号产生。

(2) RTC 溢出时序



SECF 和 OWF 为 RTC_PCLK 域信号，分别采样 RTC_CLK 域 RTC_Sectond 和 RTC_Overflow 号产生。

27.4 寄存器描述

寄存器宽度为 32 位，只允许字操作。

27.4.1 RTC 控制寄存器高位(RTC_CRH)(0x00)

Address offset:0x00

Reset value:0x0000 0000

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

该寄存器由系统复位复位。

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	OWIE	RW	0	溢出中断允许位 0：不允许溢出中断 1：允许溢出中断
1	ALRIE	RW	0	闹钟中断允许位 0：不允许闹钟中断 1：允许闹钟中断
0	SECIE	RW	0	秒中断允许位 0：不允许秒中断 1：允许秒中断

这些位用于屏蔽中断请求。注意：在复位后，所有中断是未使能的，所以在初始化后，写 RTC 寄存器以确保没有正在挂起的中断请求是可能的。但是当外设正在完成前一次的写操作（RTOFF=0）时，是不能写 RTC_CRH 寄存器的。

该控制寄存器控制着 RTC 的功能。某些位必须使用专门的配置流程才能进行写操作。

27.4.2 RTC 控制寄存器低位(RTC_CRL)(0x04)

Address offset:0x04

Reset value:0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTOFF	CNF	RSF	OWF	ALRF	SECF
-										R	RW	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5	RTOFF	R	1	RTC 操作关闭（RTC operation OFF)，该位只读。 RTC 模块利用该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。 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
4	CNF	RW	0	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 RTC_CNT、RTC_ALR 或 RTC_PRL 寄存器写入新值。 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
3	RSF	RC_W0	0	寄存器同步标志（Registers synchronized flag).该寄存器由硬件置位，软件清零。当 RTC_CNT 寄存器和 RTC_DIV 寄存器更新后，硬件置'1'该位。 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，用户程序必须等待该位被硬件置'1'，以确保 RTC_CNT、RTC_ALR 或 RTC_PRL 已经被同步。 0：寄存器尚未被同步 1：寄存器已经被同步
2	OWF	RC_W0	0	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_CRH 寄存器中 OWIE=1，则产生中断。 此位只能由软件清'0'，写'1'无效。 0：无溢出； 1：32 位可编程计数器溢出
1	ALRF	RC_W0	0	闹钟标志（Alarm flag)。 当 32 位可编程计数器达到 RTC_ALR 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清'0'，写'1'无效。 0：无闹钟； 1：有闹钟。
0	SECF	RC_W0	0	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_CRH 寄存器中 SECIE=1，则产生中断。此位只能由软件清除，写'1'无效。 0：秒标志条件不成立 1：秒标志条件成立

RTC 的功能是被该控制寄存器控制的。当外设正在继续上一次写操作时（RTOFF=0），是不能写 RTC_CRL 寄存器的。

注：

- 任何标志位都将保持挂起状态，直到适当的 RTC_CR 请求位被软件复位，表示所请求的中断已经被接收。
- 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
- 当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新（无法同步）。
- OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零。
- 若 ALRF=1 且 ALRIE=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。
- 若 ALRF=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)。

27.4.3 RTC 预分频装载寄存器高位(RTC_PRLH)(0x08)

PRL 寄存器用来保存 RTC 预分频器周期性的计数值。该寄存器是被 RTC_CR 寄存器的 RTOFF 位写保护的，只有 RTOFF=1，才允许 CPU 进行写操作。

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

Address offset:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3:0	PRL[19:16]	W	4'h0	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： $f_{TR_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位

27.4.4 RTC 预分频装载寄存器低位(RTC_PRLL)(0x0C)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

Address offset:0x0C

Reset value:0x0000 8000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	PRL[15:0]	W	16'h8000	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： fTR_CLK = fRTCCLK/(PRL[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。

27.4.5 RTC 预分频余数寄存器高位(RTC_DIVH)(0x10)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

在每个 TR_CLK 周期，RTC_PRL 寄存器的值被重装载到 RTC 预分频计数器里。用户可以通过读取 RTC_DIV 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。

该寄存器只读属性，当 RTC_PRL 或者 RTC_CNT 寄存器的值发生任何变化，该寄存器值将由硬件重新装载。

Address offset:0x10

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3:0	DIV[19:16]	R	4'h0	RTC 时钟分频器余数高位。

27.4.6 RTC 预分频余数寄存器低位(RTC_DIVL)(0x14)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

Address offset:0x14

Reset value:0x0000 8000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	DIV[15:0]	R	16'h8000	RTC 时钟分频器

27.4.7 RTC 计数寄存器高位(RTC_CNTH)(0x18)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

RTC 模块有个 32 位可编程的计数器,该寄存器通过两个寄存器访问,计数基于预分频器产生的 TR_CLK 时间基准为参考进行计数。

RTC_CNT 寄存器用来存放计数器的计数值。寄存器是被写保护的，仅当 RTOFF=1 时 CPU 才能进行写操作。对高 16 位的 RTC_CNTH 或者低 16 位的 RTC_CNTL 寄存器进行写操作，直接装载到相应的可编程计数器里，并重装载 RTC 预分频器。当读操作发生，返回计数器的当前值（系统时间）。

Address offset:0x18

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CNT[31:16]	RW	16'h0000	RTC core counter 的高 16 位. 当读 RTC_CNTH 寄存器时，返回 RTC 计数器寄存器的当前值的高 16 位。只有进入配置模式才能对该寄存器进行写操作。

27.4.8 RTC 计数寄存器低位(RTC_CNTL)(0x1C)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

Address offset:0x1C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	CNT[15:0]	RW	16'h0000	RTC core counter 低 16 位 当读 RTC_CNTL 寄存器时，返回 RTC 计数器寄存器当前值的低 16 位。只有进入配置模式才能对该寄存器进行写操作。

27.4.9 RTC 闹钟寄存器高位(RTC_ALRH)(0x20)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

当可编程计数器（计数）达到存储在 RTC_ALR 寄存器的 32 位值时，并产生 alarm 中断请求。该寄存器是被 RTOFF 位写保护的，只有 RTOFF=1，才允许写访问。

Address offset:0x20

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	ALR[31:16]	RW	16'hFFFF	RTC 闹钟值高 16 位. 该寄存器用来保存由软件写入的闹钟时间的高 16 位。写该寄存器必须进入配置模式。

27.4.10 RTC 闹钟寄存器低位(RTC_ALRL)(0x24)

该寄存器位于 V_{BAT} 域，微控制器掉电后数据不改变。

Address offset:0x24

Reset value:0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	ALR[15:0]	RW	16'hFFFF	RTC 闹钟值低 16 位. 该寄存器用来保存由软件写入的闹钟时间的高 16 位。写该寄存器必须进入配置模式。

28. 独立看门狗（IWDG）

28.1 简介

独立看门狗寄存器（简称 IWDG），该模块具有高安全级别、时序精确及灵活使用的特点。IWDG 可用来检测和解决由软件错误引起的故障，并在计数器达到指定的超时值时（TIMEOUT）触发系统复位。

独立看门狗(IWDG)由专用的低速内部时钟(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。

28.2 主要特性

- 自由运行的递减计数器
- 时钟由独立的 RC 振荡器提供(可在停止模式和待机模式下工作)
- 看门狗被激活后，则在计数器递减至 0x000 时产生复位
- 具有低功耗冻结功能，能在停止模式和待机模式下保持工作状态，计数器保持递减计数

28.3 功能说明

28.3.1 模块框图

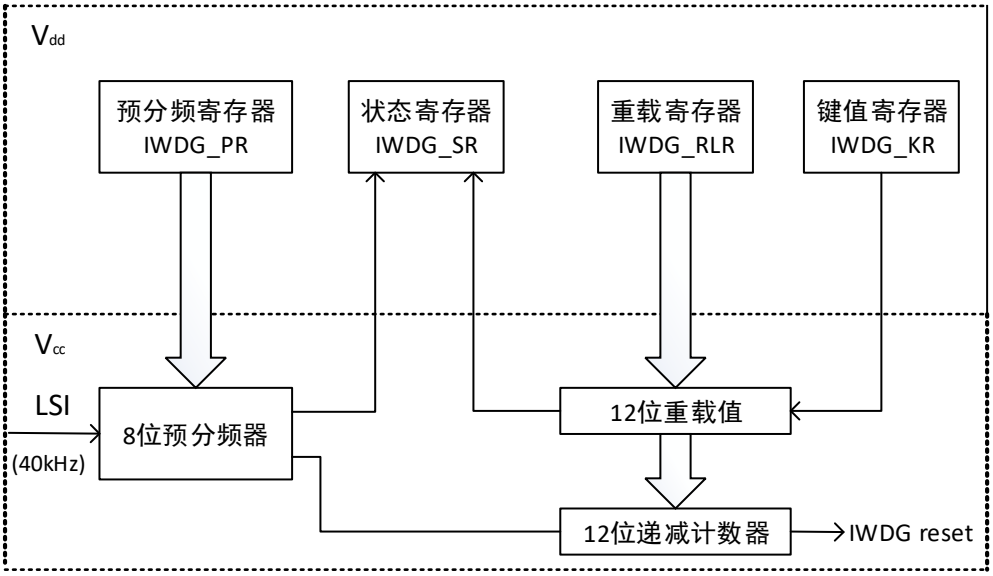


图 28-1 IWDG 模块框图

注：看门狗功能处于 Vcc 供电区，即在停机和待机模式时仍能正常工作。

在键寄存器(IWDG_KR)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，只要键寄存器 IWDG_KR 中被写入 0xAAAA，IWDG_RLR 中的值就会被重新加载到计数器中从而避免产生 IWDG 复位 。

表 28-1 看门狗超时时间 40 kHz 的输入时钟(LSI)

预分频系数	PR[2:0]位	最短时间(ms) PR[11:0] = 0x000	最长时间(ms) PR[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	6.4	26214.4

注：这些时间是按照 40 kHz 时钟给出。实际上，微控制器内部的 RC 频率会在 30 kHz~ 60 kHz 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。

28.3.2 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

28.3.3 寄存器保护

对 IWDG_PR 和 IWDG_RLR 寄存器的写访问是被保护的，要修改这两个寄存器的值，IWDG_KR 必须先被写入 0x5555。对这些寄存器的写其他数将重新开启写保护，比如写入 0xAAAA，IWDG_PR 和 IWDG_RLR 将被再次保护。

IWDG_SR 指示 IWDG_PR 或 IWDG_RLR 的值是否正在更新。

28.3.4 调试模式

如果 CPU 进入调试模式，根据调试模块中的 DBG 模块中 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器继续工作或停止

28.3.5 低功耗冻结

取决于选项字节里的 IWDG_STOP 和 IWDG_STBY 位，决定了 IWDG 停止模式或待机模式下是否继续保持计数。如果 IWDG 在停止或待机模式下保持工作，IWDG 能从当前的低功耗模式下唤醒微控制器。

28.4 寄存器描述

只能字的方式操作以下寄存器。

28.4.1 键值寄存器（IWDG_KR）

Address offset:0x00

Reset value:0x0000 0000（由待机模式复位）

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	KEY[15:0]	W	16'h0	键值。 软件必须以一定的时间间隔向该寄存器写入 0xAAAA，否则，当计数器计数到 0 时，看门狗会产生复位。 写入 0x5555：允许访问 IWDG_PR 和 IWDG_RLR； 写入 0xCCCC：激活 IWDG（如果选择了硬件看门狗则不受此命令控制）。

28.4.2 预分频寄存器（IWDG_PR）

Address offset:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2:0	PR[2:0]	RW	3'h0	预分频值。 通过配置该寄存器选择计数器时钟的预分频值。 要改变该寄存器，IWDG_SR 寄存器的 PVU 必须为 0。 此寄存器受写保护。向 IWDG_KR 写入 0x5555 可解除写保护。 000：4 分频； 001：8 分频； 010：16 分频； 011：32 分频； 100：64 分频； 101：128 分频； 110：256 分频； 111：256 分频；

28.4.3 重装载寄存器（IWDG_RLR）

Address offset:0x08

Reset value:0x0000 0FFF（由待机模式复位）

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

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	RL[11:0]	RW	12'h0	IWDG 计数器重装载值。

				当向 IWDG_KR 寄存器写入 0xAAAA 时，RL 值会传送到计数器中。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 RL 值和时钟预分频值来计算。 此寄存器受写保护。向 IWDG_KR 写入 0x5555 可解除写保护。只有当 IWDG_SR.RVU=0 时，才能对寄存器进行修改。

28.4.4 状态寄存器（IWDG_SR）

Address offset:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	RVU	R	0	看门狗计数器重装值更新。 此位由硬件置 1，表明重装载值正在更新。当重装载值更新结束后，此位由硬件清零。
0	PVU	R	0	看门狗预分频值更新。 此位由硬件置 1，表明预分频值正在更新。当预分频值更新结束后，此位由硬件清零。

注：在更新 IWDG_PR 和 IWDG_SR.RLR 前，要分别等待 IWDG_PVU 和 IWDG_SR.RVU 为 0.但在更新 IWDG_PR、IWDG_RLR 后，不必再等待 IWDG_SR.PVU、IWDG_SR.RVU 为 0，可继续执行后续的代码。

29. 窗口看门狗（WWDG）

29.1 简介

窗口看门狗通常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跑飞而产生的软件故障。除非递减计数器的值在 T[6]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。

29.2 主要特性

- 可编程的自由运行递减计数器
- 若看门狗被启动，满足以下任一条件时，产生系统复位:
  - 递减计数器的值小于 0x40
  - 递减计数器在窗口外被重新装载
- 如果启动了看门狗并且开启早期唤醒中断(EWI)，则当递减计数器等于 0x40 时产生 EWI，软件可在 EWI 中断程序内重装载计数器以避免 WWDG 复位。

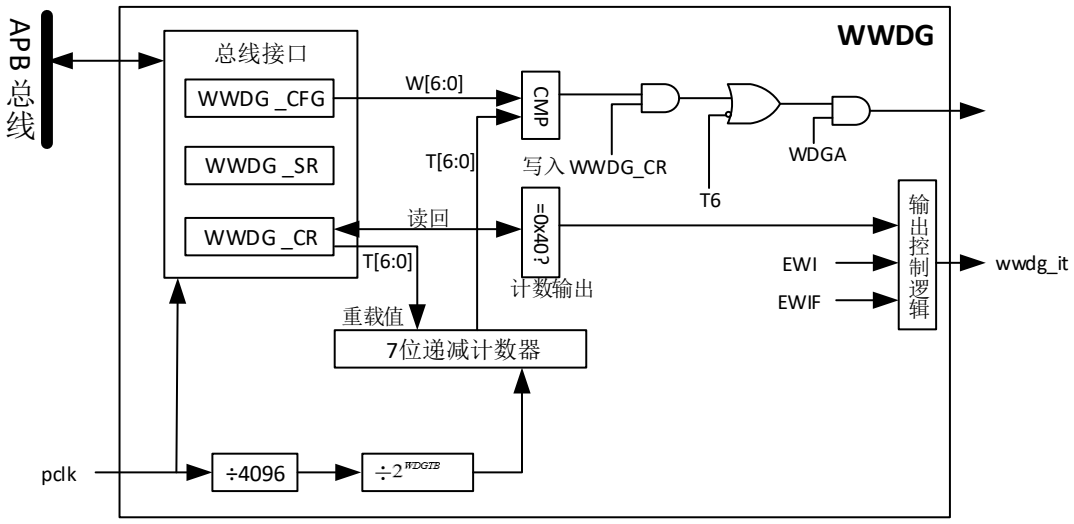


图 29-1 WWDG 模块框图

29.3 WWDG 功能描述

如果看门狗被启动(WWDG_CR 寄存器中的 WDGA 位被置”1”)，并且当 7 位(T[6:0])递减计数器从 0x40 翻转到 0x3F(T[6]位清零)时，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。

软件在正常运行过程中必须定期地写入 WWDG_CR 寄存器以防止微控制器发生复位。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_CR 寄存器中的数值必须在 0xFF 和 0xC0 之间:

- 启动看门狗

当选项字节里的 WWDG_SW 选择了软件启动 WWDG 时（WWDG_SW=1），系统复位之后 WWDG 处于关闭状态，将 WWDG_CR 里的 WDGA 为设为 1 即可使能 WWDG。WWDG 一旦被软件启动就无法被关闭，除非发生了系统复位。

当选项字节里的 WWDG_SW 选择了硬件启动 WWDG 时（WWDG_SW=0），系统复位之后 WWDG 时钟自动启动，且 WWDG 处于开启状态，且无法被关闭。

■ 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，T[6]位必须被置 1，以防止立即产生一个复位。

T[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_CR 寄存器时，预分频值是未知的。

配置寄存器(WWDG_CFR) 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 4-1 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(EWI)。设置 WWDG_CFR 寄存器中的 EWI 位开启该中断。当递减计数器到达 0x40 时产生此中断，相应的中断服务程序(ISR)可以用来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写'0'可以清除该中断。

注：可以用 T[6]位产生一个软件复位(设置 WDGA 位为"1"，T[6]位为"0")。

29.4 如何编写看门狗超时程序

可以使用下图中提供的公式计算窗口看门狗的超时时间。

警告：当写入 WWDG_CR 寄存器时，始终置 T[6] 位为'1'以避免立即产生一个复位。

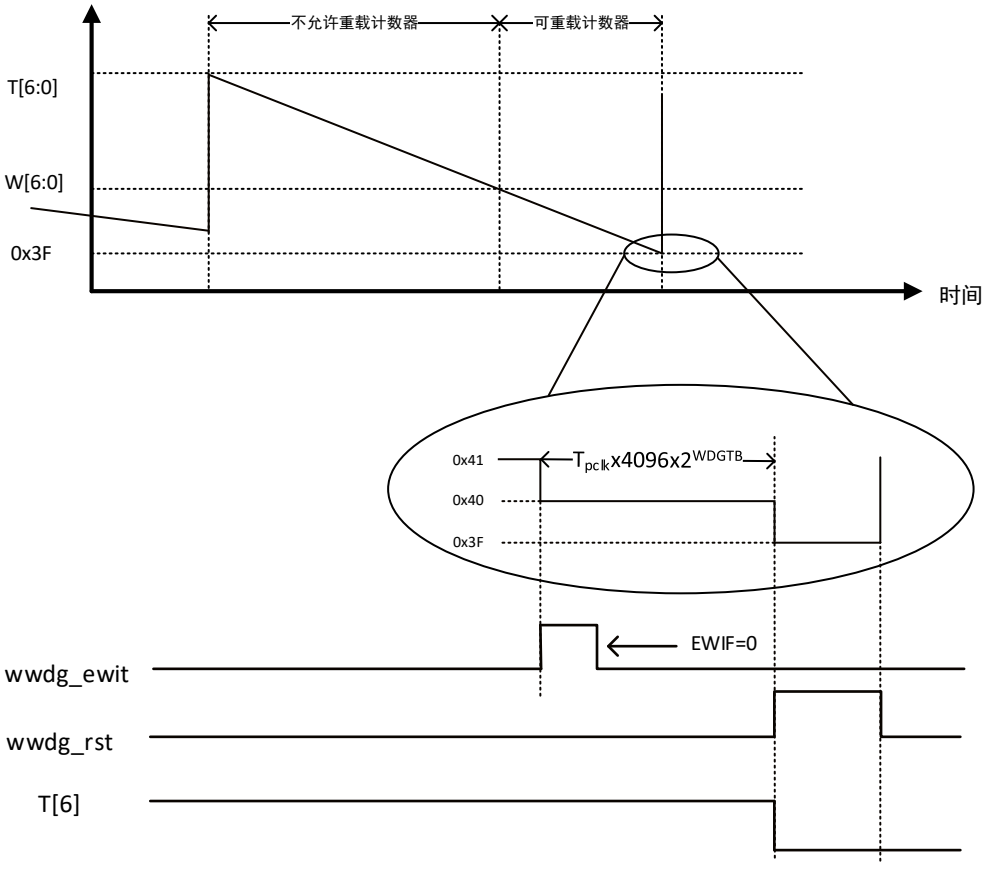


图 29-2 窗口看门狗时序图

计算 WWDG timeout 值的公式如下：

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WWDGTB[1:0]} \times (T[5:0] + 1) \quad (ms)$$

29.5 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，决定 WWDG 的计数器继续工作或停止。详见调试模式章节。

29.6 寄存器描述

29.6.1 控制寄存器（WWDG_CR）

Address offset:0x00

Reset value:0x0000 007F

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	WDGA	RS	0	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 WDGA=1 时，看门狗可以产生复位。 0：禁止看门狗 1：启用看门狗
6:0	T[6:0]	RW	7’h7F	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^WDGTB)个 PCLK1 周期减 1。当计数器值从 0x40 变 为 0x3F 时(T[6]变成 0)，产生看门狗复位。

29.6.2 配置寄存器（WWDG_CFR）

Address offset:0x04

Reset value:0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		W[6:0]						
-						RS	RW		RW						

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
9	EWI	RS	0	提前唤醒中断（Early wakeup interrupt) 此位若置’1’，则当计数器值达到 0x40，即产生中断。 此中断只能由硬件在复位后清除。

8:7	WDGTB[1:0]	RW	2’h0	时基（Timer base) 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
6:0	W[6:0]	RW	7’h7F	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。

29.6.3 状态寄存器（WWDG_SR）

Address offset:0x08

Reset value:0x0000 0000

21	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															EWIF
-															RC_W0

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	EWIF	RC_W0	0	提前唤醒中断标志（Early wakeup interrupt flag) 当计数器值达到 0x40 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”。

30. 外部串行存储控制器（ESMC）

30.1 简介

ESMC（External Serial Memory Controller）是一种专用通信接口，用于单（Single SPI）、双（Dual SPI）、四（Quad SPI）和八（Octal SPI）通道 SPI 接口存储器（NOR Flash，PSRAM 等）。它可以在以下两种模式中的任何一种模式下运行：

- 间接模式：所有操作均使用 ESMC 寄存器执行
- 内存映射(XIP)模式：外部闪存映射到设备地址空间，系统将其视为内部存储器

使用双存储器模式，同时访问两个 Qual SPI 存储器，类似于单个 Octal SPI 存储器，将吞吐量和容量加倍。

30.2 主要特征

- 支持两种功能模式：间接和内存映射
- 可同时发送/接收 8 位
  - 双闪存模式，通过并行访问两个闪存，可同时发送/接收 8 位
  - Octal SPI
- 支持 SDR 和 DDR
- 具有接收和发送 FIFO
- 允许 8 位、16 位和 32 位数据访问（间接模式仅支持 32 位读）
- 间接模式下，支持 DMA 操作
- 表示 FIFO 状态和操作完成的中断

30.3 ESMC 功能描述

30.3.1 ESMC 外部存储器连接图

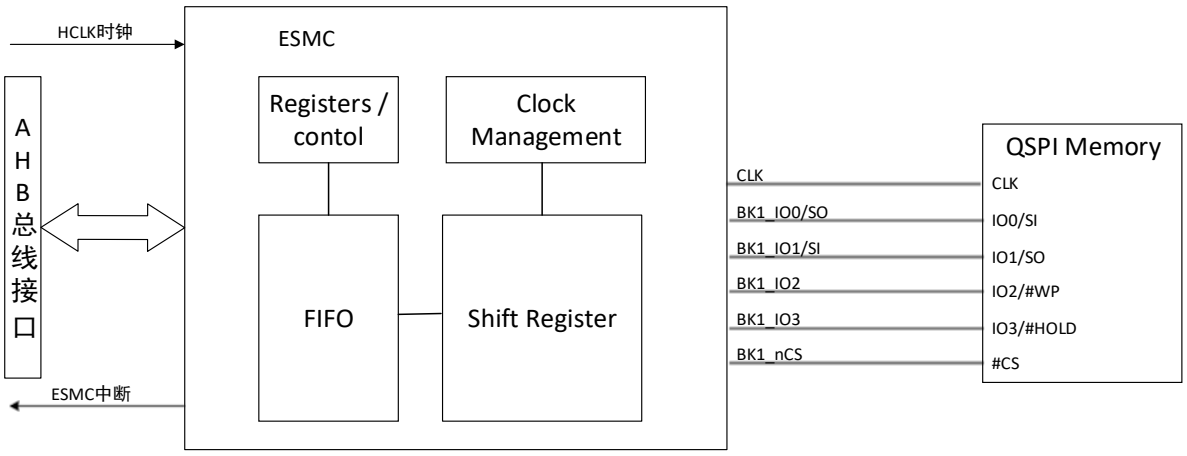


图 30-1 ESMC 连接图（Dual QSPI Memory 模式禁止时）

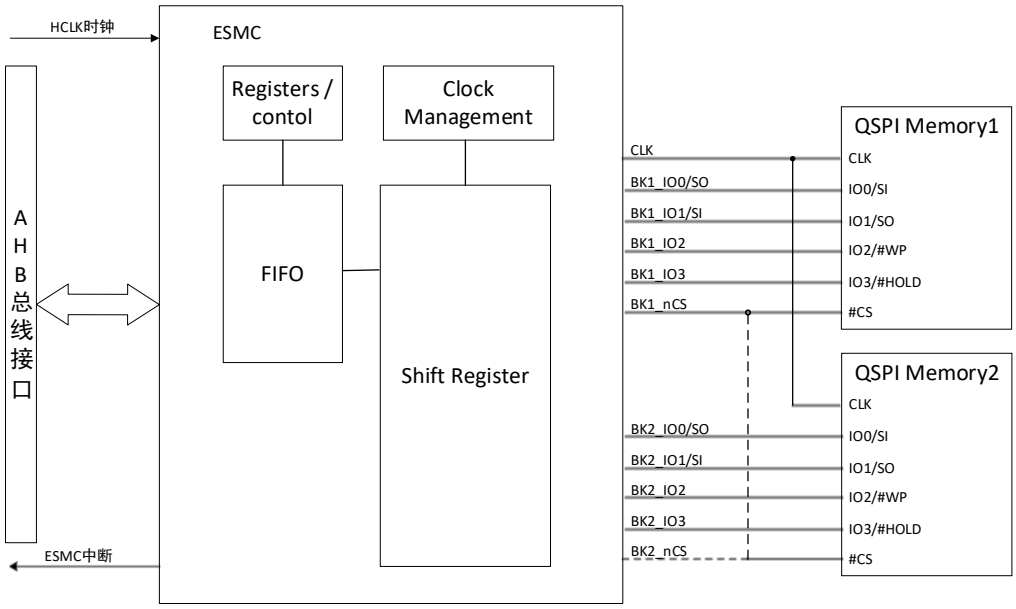


图 30-2 ESMC 连接图（Dual QSPI Memory 模式设置时）

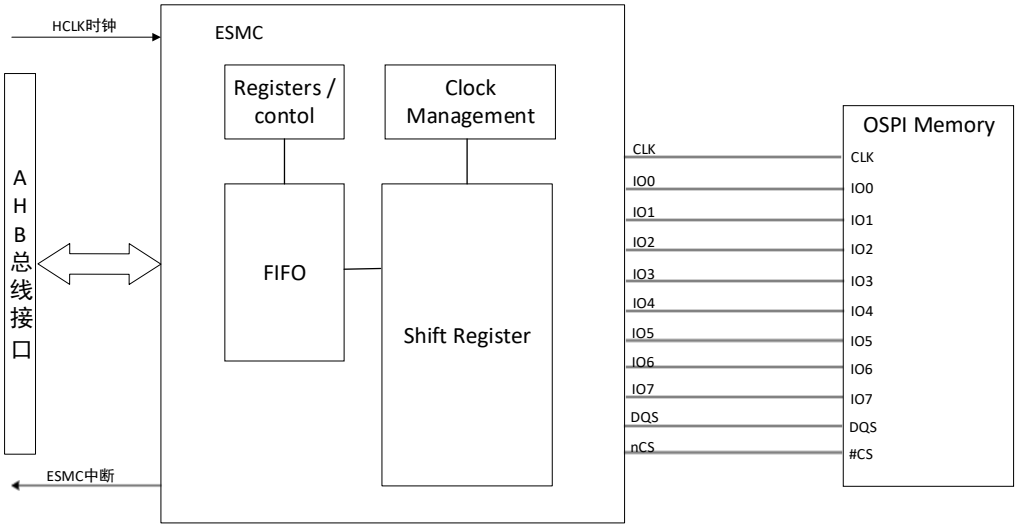


图 30-3 ESMC 连接图（OSPI Memory 模式设置时）

30.3.2 ESMC 命令序列

ESMC 使用命令与外部存储器通信。每个命令可以包括 5 个阶段：指令、地址、备用字节、Dummy、数据。可以跳过这些阶段中的任何一个配置，但必须至少存在指令、地址、备用字节或数据阶段中的一个。

nCS 在每个命令开始前拉低，在每个命令完成后再次拉高。

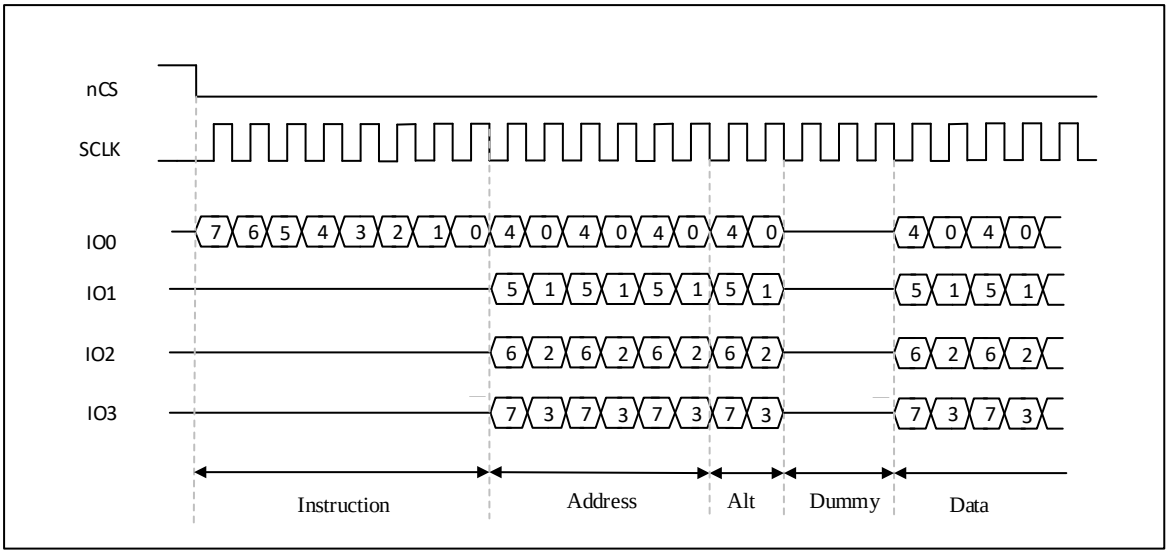


图 30-4 QSPI 指令示例

指令阶段

在此阶段，在 ESMC_SFCR[7:0]或 ESMC_ADDR24[7:0]寄存器的指令字段中配置的 8 位指令被发送到闪存，指定要执行的操作类型。

尽管大多数闪存每次(每个 cycle)只能从 IO0 引脚（单 SPI 模式）接收一位指令，但指令阶段可以选择一次发送 2 位（双 SPI 模式下 IO0/IO1）、一次发送 4 位（四 SPI 模式下 IO0/IO1/IO2/IO3）或一次发送 8 位（八 SPI 模式下 IO0 / IO1 /IO2 /IO3 /IO4 /IO5 / IO6/ IO7）。

指令的长度可以设置成无指令长度，1 字节指令和 2 字节指令。指令长度的大小通过以下寄存器实现：

指令长度的大小	配置的寄存器
无指令代码	使能位：ESMC_DCR/ESMC_XDCR/ESMC_XDCR_WE.NO_CMD = 1
1 字节指令代码	使能位：无，默认情况下位 1 字节指令 第 1byte 指令：ESMC_ ADDR24.INS1 [7:0] or ESMC_SFCR（只发送指令）
2 字节指令代码	使能位：ESMC_CR3/ESMC_XCR3/ESMC_XCR3_WE.CMD_EXT = 1 第 1byte 指令：ESMC_ ADDR24.INS1 [7:0] 第 2byte 指令：写入位 ESMC_ADDR32.MREG[7:0] or ESMC_XMODE or ESMC_XMODE_WE

地址阶段

在地址阶段，1-4 个字节被发送到闪存，以指示操作的地址。要发送的地址字节数在 ESMC_CR3 寄存器的 32BIT_ADDR，16BIT_ADDR，8BIT_ADDR 位配置（默认是 24bit 的地址）。在间接模式下，要发送的地址字节在 ESMC_ADDR 寄存器的地址[31:0]字段中指定，而在内存映射模式下，地址直接通过 AHB 总线给出。

地址可以一次发送 1 位 (在单 SPI 模式下通过 IO0)，一次发送 2 位 (在双 SPI 模式下通过 IO0/IO1)，一次发送 4 位（在四 SPI 模式下通过 IO0/IO1/IO2/IO3）或一次发送 8 位（八 SPI 模式下 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7）。

当 ESMC_SOCR 寄存器 NO_ADDR 位设置时，将跳过地址阶段，命令序列将直接进入下一阶段。

ESMC 包含 4 个 8 位地址寄存器 ADDR3、ADDR2、ADDR1、ADDR0，允许寻址支持 24 位和 32 位地址字段的闪存。默认情况下，ESMC 配置为 24 位地址。

在访问外部 SPI 闪存期间从 ESMC 发送 32 位地址。默认情况下，只发送地址寄存器的 24 个 LSB 位。设置 ADDR32BIT（CR3[2])后，32 位地址将从 ESMC 传输出去。地址寄存器值以当前选择的 SPI 模式(扩展 SPI、单 SPI、双 SPI、四 SPI 或八 SPI)通道发送。地址寄存器应在 ESMC 加载 SFCR 寄存器之前写入。

备用字节阶段

在备用字节阶段，1 个字节被发送到闪存，通常用于控制操作模式。要发送的备用字节数据在 ESMC_ADDR32[23:16]寄存器的 MREG[7:0]字段中配置。

备用字节阶段可以一次发送 2 位（在双 SPI 模式下通过 IO0/IO1），或一次发送 4 位（在四 SPI 模式下通过 IO0/IO1/IO2/IO3）。

当 ESMC_SOCR.SEND_M=0 时，将跳过备用字节阶段，命令序列将直接进入下一阶段。

Dummy 阶段

在 DUMMY 阶段，在不发送或接收任何数据的情况下给出 1-63 个周期，以便闪存在使用更高时钟频率时有时间准备数据。该阶段给出的周期数在 ESMC_DCR 寄存器的 DUMMY[5:0]字段中指定。在 SDR 和 DDR 模式中，持续时间都指定为完整 SCLK 周期数。

当 DUMMY 为零时，跳过 DUMMY 周期阶段，命令序列直接进入数据阶段。

数据阶段

在数据阶段，可以向闪存发送或从闪存接收任意数量的字节。

在间接模式下，发送/接收的字节数在 ESMC_BCR 寄存器中指定。

在间接写入模式下，发送至闪存的数据必须先写入 ESMC_DATA 寄存器，而在间接读取模式下，从闪存接收的数据通过读取 ESMC_DATA 寄存器获得。

在内存映射模式下，读取/写入的数据直接通过 AHB 总线进行。

数据阶段一次可以发送/接收 1 位(在单 SPI 模式下通过 IO0/IO1)、2 位(在双 SPI 模式下通过 IO0/IO1)、4 位(在四 SPI 模式下通过 IO0/IO1/IO2/IO3)或八位(八 SPI 模式下 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7)。

30.3.3 ESMC 接口协议模式

Single SPI 模式

传统 SPI 模式只允许串行发送/接收单个位。在此模式下，数据通过 SO 引脚（该 I/O 与 IO0 共享）发送到闪存，通过 SI 引脚(该 I/O 与 IO1 共享）从闪存接收。

通过将 ESMC_SOCR[1:0]设置为 00，以使用单 SPI 模式。

Dual SPI 模式

在双 SPI 模式下，通过 IO0/IO1 引脚同时发送/接收两位数据。

通过将 ESMC_SOCR[1:0]设置为 01，以使用双 SPI 模式。

Quad SPI 模式

在四 SPI 模式下，通过 IO0/IO1/IO2/IO3 引脚同时发送/接收四位数据。

通过将 ESMC_SOCR[1:0]设置为 10，以使用四 SPI 模式。

Dual Flash 模式

当 2QUAD 位（ESMC_CR）为 1，并且将 ESMC_SOCR[1:0]设置为 11 时，ESMC 处于双闪存模式，其中使用两个外部 QUAD SPI 闪存（闪存 1 和闪存 2），以便每个周期发送/接收 8 位数据（或 DDR 模式下的 16 位数据），将吞吐量和容量加倍。

每个闪存使用相同的 SCLK 引脚和可选的相同 nCS 引脚，但每个都有单独的 IO0、IO1、IO2 和 IO3 引脚。

双闪存模式可与 SDR 或 DDR 模式结合使用。

每个 byte 的[7:4]位数据存放在闪存 1，每个 byte [3:0]位数据存放在闪存 2。

在双闪存模式下读取 Flash 状态寄存器时，与在单 Flash 模式下读取相同字节相比，应读取两倍的字节。这意味着，如果每个 Flash 的状态位为 8 位，则必须将 QUAD SPI 配置为 2 字节，并且 QUADSPI 从每个 Flash 接收一个字节。如果每个 Flash 的状态为 16 位，则必须将 QUAD SPI 配置为 4 个字节，并且 QUADSPI 从每个 Flash 接收两个字节。

数据寄存器的第一个字节是 Flash1 第一字节，而第二个字节是 Flash2 第一字节。然后，数据寄存器的第三个字节是 Flash1 第二字节，而第四个字节是 Flash2 第二字节（在 Flash 具有 16 位状态寄存器的情况下）。

在双闪存模式下，必须始终访问偶数个字节。因此，ESMC_BCR 需要设置为偶数。

注：双闪存模式下，写给外部存储器的地址是实际地址的一半，比如，写的地址是 0x8，实际发给外部存储器的地址是 0x4。读取的地址时和实际的地址是一致的，比如读的地址是 0x8，实际发给外部存储器的地址也是 0x8。

Octal SPI 模式

在八 SPI 模式下，通过 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7 引脚同时发送/接收八位数据。

通过将 ESMC_SOCR [1：0]设置为 11，以使用八 SPI 模式。

SDR 模式

默认情况下，DDRDATA，DDRADDR,DDRCMD 位（ESMC_SOCR）为 0，ESMC 以单数据速率（SDR）模式运行。

在 SDR 模式下，当 ESMC 驱动 IO0、IO1、IO2、IO3 引脚时，这些引脚仅随 SCLK 下降沿转换。

在 SDR 模式下接收数据时，假设闪存使用 SCLK 的下降沿发送数据，ESMC 使用 SCLK 的上降沿对信号进行采样。

DDR 模式

当 DDR 位被设置时（可以通过设置 ESMC_SOCR 寄存器的 DDRDATA，DDRADDR,DDRCMD 位来分别设置数据，地址，指令），ESMC 以双数据速率(DDR)模式运行。

在 DDR 模式下，当 ESMC 在地址/备用字节/数据阶段驱动 IO0、IO1、IO2、IO3、IO4、IO5、IO6、IO7 引脚时，在 SCLK 的每个下降沿和上升沿上发送一位数据。

在 DDR 模式下接收数据时，假定闪存也使用上升和下降 SCLK 边沿发送数据，ESMC 使用 SCLK 周期的一半之后（在下一个相反的边沿）对信号进行采样。

DQS（/RWDS）的用法

DQS 引脚可用于读取数据选通。

将 八 SPI 模式下，配置寄存器（ESMC_CR3）的 XSPI_RWDS 使能置 1，可使能 DQS 管理。

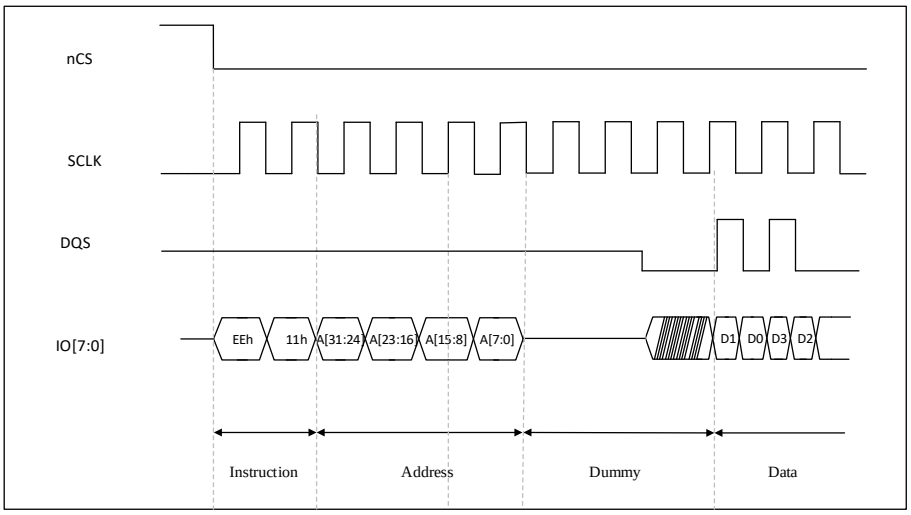


图 30-5 八线模式下 DQS 的 DDR 读时序

表 30-1 ESMC 能支持的存储器类型如下

器件类型	接口协议模式		数据方向
NOR Flash	1 线	SDR	接收
			发送
		DDR	接收
	2 线	SDR	接收
			发送
		DDR	接收
	4 线	SDR	接收
			发送
		DDR	接收
	双 4 线	SDR	接收
			发送
		DDR	接收
	8 线	SDR	发送
			接收
		DDR	发送
			接收
PSRAM	1 线	SDR	接收
			发送
	4 线	SDR	接收
			发送
	双 4 线	SDR	接收
			发送

30.3.4 ESMC 的工作模式

ESMC 有两种工作模式：

- 间接模式（寄存器控制）
- 内存映射模式

工作模式通过 XIP_EN 位（ESMC_CR 的第 6 位）设置(本模块支持 XIP 读和 XIP 写功能，但 XIP 写功能仅支持除 8 线以外的 PSRAM 使用)：
设置为 1 时，为内存映射模式。
设置为 0 时，为间接模式。

30.3.4.1 ESMC 间接模式

在间接模式下，命令通过写入相关寄存器来启动，数据通过写入或读取数据寄存器来传输。
当 REC =0(ESMC_DCR 的第 7 位)时,ESMC 处于间接写入模式,通过写入数据寄存器(ESMC_DATA)在数据阶段将字节发送到 Flash。
当 REC =1 时，ESMC 处于间接读取模式，通过读取 ESMC_DATA 在数据阶段从 Flash 接收字节。

30.3.4.2 ESMC 内存映射模式

在内存映射模式下，ESMC 允许从 SPI Flash 直接读取或写入数据。在这种模式下，ESMC 作为 SPI flash 和 CPU 之间的串并转换器进行操作。不同于间接模式，内存映射（XIP）模式有专用的寄存器，通用寄存器为 XSTRR，XSSOCR 和 XSSOCR_WE；读操作包括 XMODE，XSFCR，XSOCR，XDCR 和 XCR3；写操作寄存器为 XMODE_WE，XSFCR_WE，XSOCR_WE，XDCR_WE 和 XCR3_WE。
XIP 访问期间发送的指令保存在 ESMC XIP 指令寄存器(XSFCR 和 XSFCR_WE)。这两个寄存器的功能与 SFCR 寄存器类似，但仅在 XIP 模式下使用，写入寄存器不会启动指令代码传输。指令的传输在选定的 SPI 模式下完成。
要在 XIP 模式下操作，需要首先配置 SPI Flash。一旦配置正确，ESMC 的 XIP 操作在通电后即可就绪，无需进一步操作。
ESMC 控制器允许在系统复位后，使用预设的 READ Command 进行 FLASH 读取。其他的传输配置，需要通过软件配置 XIP 相关寄存器实现。
在 XIP 模式下，CPU 通过 AHB 总线，根据 ESMC 的 XSFCR 配置的命令开始读取操作，或者根据 ESMC 的 XSFCR_WE 配置的命令开始发送操作，地址来自于 AHB 总线。用户在使用 XIP 的过程中，如果要发送新的 FLASH 命令，不需要禁止 XIP，只需在操作过程中避免对 XIP 内存空间的任何访问，否则操作将终止并开始 XIP 读取。
要禁用 ESMC 中的 XIP 模式，需要在软件中清除 XIP_EN 位（ESMC_CR[6]）。一旦该位为 0，XIP 总线的访问不会对 ESMC 产生作用。XIP 操作在默认情况下是启用的，因此 READ 可以在系统复位后立即启动。通过清除 XIP 启用位(ESMC_CR[6]),可以在软件中禁用它。如果 XIP_EN 位(ESMC_CR[6])被清除，XIP 总线将设置为忙，用以指示访问错误。
注：XIP 写在 mode0 的情况下，需要设置为二分频以上才能工作。

30.3.5 ESMC 的可编程选项

30.3.5.1 字节计数

ESMC 使用一个 32 位的寄存器(BCR 寄存器)来控制数据的传输，包括读操作和写操作；
ESMC 支持定长的数据传输,默认情况下,只要 RXFIFO/TXFIFO 至少有一个空闲空间/至少一个数据，ESMC 就会从外部存储器读取/写入数据。一旦 RXFIFO/TXFIFO 的存储空间满/空,ESMC 将进入 WAIT

状态，等待主机从 RXFIFO/TXFIFO 读取数据或者写入数据。在开始下次传输之前，RXFIFO 必须完全为空，或者将 TXFIFO 写满。

此模式仅在间接模式可用。读取指定数量的字节后，SSO 引脚被停用，ESMC 进入 IDLE 状态，等待执行下一个命令。

表 30-2 IP 中不同模式下可以支持的读写字节数如下

大小	间接模式读	间接模式写	XIP 模式读	XIP 模式写	DMA 模式读	DMA 模式写
字节	x	√	√	√	x	√
半字	x	√	√	√	x	√
字	√	√	√	√	√	√

30.3.5.2 极性和相位

ESMC 仅支持模式 0 和模式 3 的 SPI 协议操作。

这是通过配置 ESMC_CR2[1:0]寄存器实现的。

如果 ESMC_CR2.CPOL 和 ESMC_CR2.CPHA 都被设为 0 时，表示为模式 0。如果 ESMC_CR2.CPOL 和 ESMC_CR2.CPHA 都被设为 1 时，表示为 SPI 模式 3。

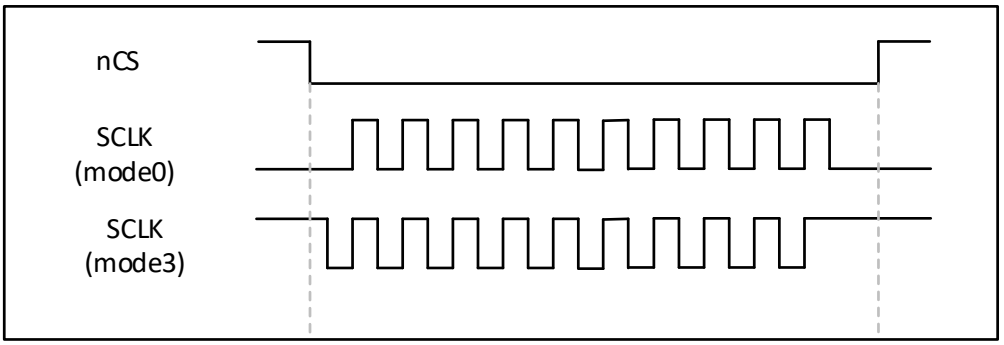


图 30-6. SDR 模式下模式 0 和模式 3 的时序

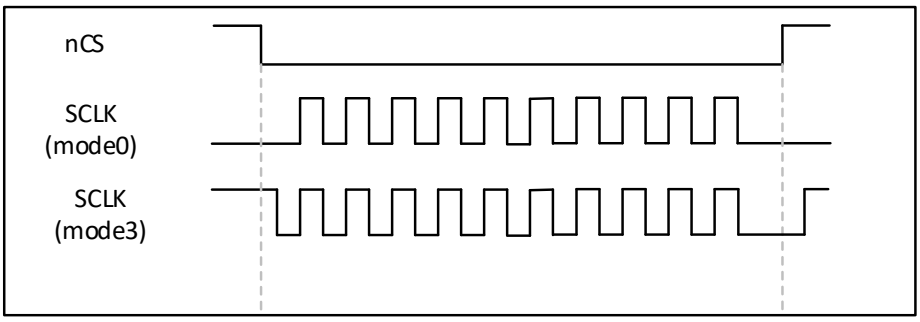


图 30-7. DDR 模式下模式 0 和模式 3 的时序

30.3.5.3 波特率

波特率寄存器 BAUD（0x3）用于定义 ESMC 的 SCLK 频率，基于系统时钟分频。BAUD 设置为 0x0 和 0x1 时，为二分频；即 SCLK 时钟最快为系统时钟的二分频。其他值计算公式为：

SCLK = HCLK / BAUD;

30.3.5.4 时序控制

时序控制寄存器 TCR(0x2)用于定义和控制时序：

- 设置在传输开始前的 nCS 引脚输出的 SETUP 时间。
- 设置在传输结束后的 nCS 引脚输出的 HOLD 时间。
- 设置在连续命令之间 Ncs 引脚输出的 HIGH 时间。

其中 SETUP 和 HOLD 时间在 ESMC_TCR.HOLD[3:0] 寄存器中配置，HIGH 时间在 ESMC_TCR.HIGH[3:0]寄存器中配置。SETUP 和 HOLD 的时间是以 HCLK 为单位，插入相应的 HCLK 的周期数。HIGH 的时间以 SCLK 为单位，设置值表示 nCS 引脚为高需要保持的 SCL 周期数。

注：如果 nCS 引脚的高电平时间超过了设置的 HIGH 时间数，没有新的指令写入，nCS 引脚也不会拉低。

30.3.5.5 ESMC 命令发送

默认情况下，命令在每个序列的最开始按照 SINGLE 模式发送。ESMC 也可以按照 DUAL、QUAD 或 OCTAL 模式发送特定的命令。

ESMC 有两个命令寄存器 SFCR0 和 SFCR1，命令的使用差别如下：

1. SFCR0 用于单字节 Flash 命令传输，如：RESET ENABLE（66h）、RESET MEMORY（99h）等。将命令写到 SFCR0 寄存器中，命令将根据当前 SPI 模式设置发出，一旦命令传输完成，nCS 引脚将变为非活动状态，以便执行下一个命令。
2. SFCR1 用于非单字节的 Flash 命令传输(包括 ADDRESS/M[7:0]/DUMMY/DATA 等序列）。

30.3.5.6 ESMC 命令序列配置

通过配置 ESMC_SOCR 寄存器可以发送特定的 ESMC 序列。下图为 ESMC 支持的几种 Flash 读/写操作及其格式。如下所述，通过配置控制寄存器允许实现闪存命令的任意组合，包括多 SPI 的命令、地址和数据，SDR/DDR 以及任意数量的 DUMMY 周期数。

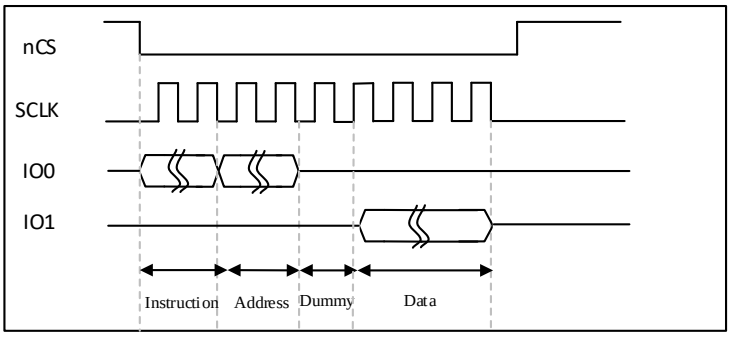


图 30-8 单线 SPI 读命令：MULTI ADDR =0; MULTI COMM =0;SPI_MODE=2’b00; DUMMY !=0

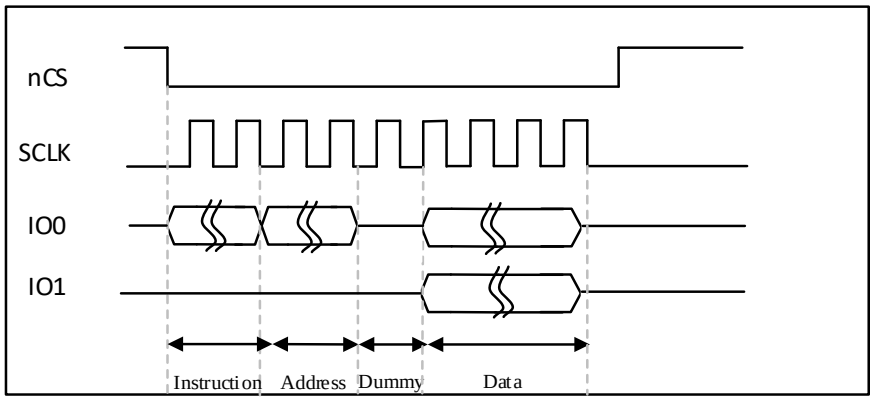


图 30-9 双线 SPI 读命令： MULTI ADDR =0; MULTI COMM =0; SPI_MODE=2’b01； DUMMY !=0

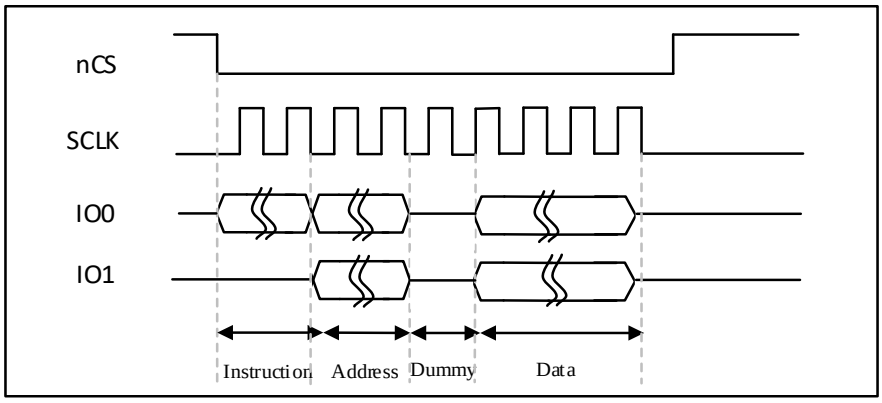


图 30-10 2IO读命令：MULTI COMM = 0; MULTI ADDR =1; SPI_MODE = 2’b01 ; DUMMY != 0

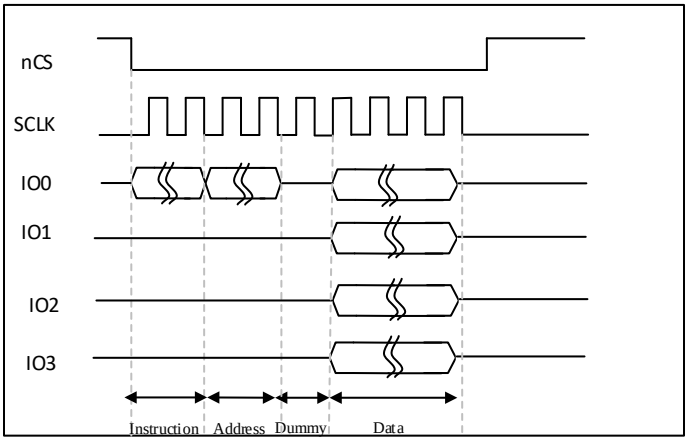


图 30-11 四线读命令：MULTI ADDR =0; MULTI COMM =0; SPI_MODE=2’b10

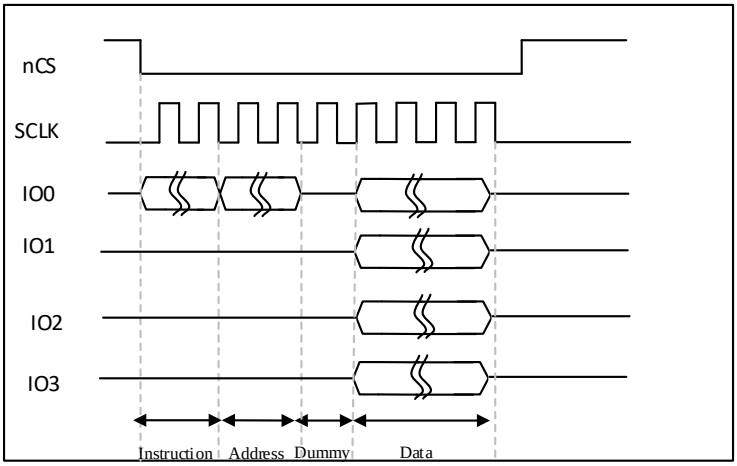


图 30-12 4IO 读命令： MULTI COMM =0; MULTI ADDR =1; SPI_MODE=2’b10

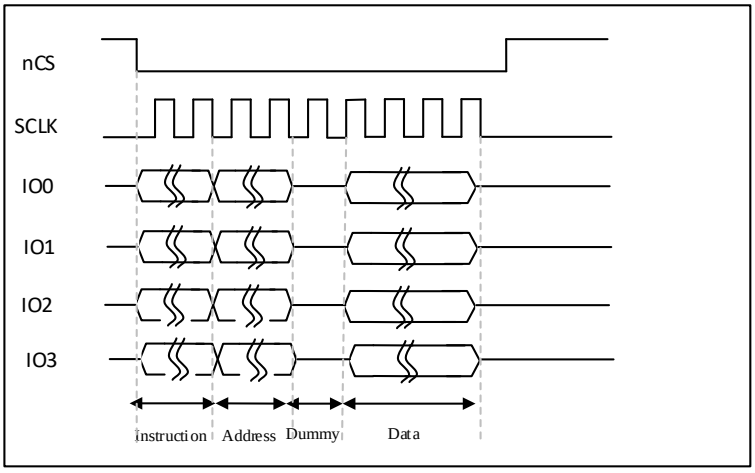


图 30-13 4IO QPI 模式读命令： MULTI COMM =1; MULTI ADDR =1; SPI_MODE=2’b10

8 线模式和 DDR 模式与以上的区别就是在更多的 IO 线上传输和双边沿采样，其他序列内容与上述命令类似。

30.3.5.7 Dummy 周期控制

通过ESMC_DCR.DUMMY[5:0]设置Dummy周期数(位[5:0])，允许设置的最大值为63。在发送Dummy周期期间，IO 引脚被设置为高阻抗状态，并且没有数据存入 RX FIFO 中，也没有数据被读出。

REC 位 (DCR 的第 7 位) 用于指定操作类型。当 REC=1 时，ESMC 对 FLASH 进行读操作。当 REC=0 时，ESMC 对 FLASH 进行写操作。如果 DATA 已被接收（或发送），则 ESMC 进入 WAIT 状态，等待 DATA 从 FIFO 中读出或写入。

NO_CMD（DCR 的第 6 位）用于使能/禁用向 SPI Flash 发送指令字节，NO_ADDR(CR3 的第 5 位)用于启用/禁止发送地址。设置 NO_CMD 位时，ESMC 仅发送地址和设置的 Dummy cycle，然后发送数据字节。

另外，Dummy 周期设置的值还与 XSPI_DUAL_LAT_OFF 位（CR2 的第 7 位）相关。当 XSPI_DUAL_LAT_0FF=1 时，ESMC 关闭双倍延迟。当 XSPI_DUAL_LAT_OFF =0 时，启用双倍延迟时。该功能只有在 xSPI（设置 CR3[4]= 1, SOCR[1:0]=2’h3, SOCR[7]=1）模式下才能使用。

表 30-3 不同模式下 DUMMY 周期数

DCR[5:0]	DUMMY 周期数		
	non-xSPI (CR3[4]=0)	xSPI OCTAL (CR2[7]=1，SOCR[7]=0，CR3[5]= 1，SOCR[1:0]=3)	xSPI OCTAL 双倍延迟 (CR2[7])=0，SOCR[7]=0，CR3[5]= 1，SOCR[1:0]=3)
N	N	N-1	2N-1
0	0	0	0
1	1	-	-
2	2	1	3
3	3	2	5
4	4	3	7
5	5	4	9
63	63	62	125

30.3.5.8 从机选择输出控制

SSOCR 控制寄存器可以在任何时候读取或写入，它用于传输期间选择驱动哪个从机。SSOCR 寄存器的内容自动分配给 SS5O-SS0O。该寄存器允许使用多达 6 个不同的 SPI 闪存进行 SPI 操作。

30.3.5.9 FIFO 的大小

ESMC 有 3 个 FIFO：用于间接模式下发送数据的 TXFIFO，用于间接模式下接收数据的 RXFIFO，用于 XIP 模式下接收数据的 XIPFIFO，深度分别是 16 个字节，16 个字节，8 个字节。

30.3.5.10 采样延伸

通过 ESMC_STR 和 ESMC_XSTR 寄存器，可配置数据阶段 ESMC 的采样点相对于 SCLK 时钟边沿延伸的 HCLK 数量。默认情况下，这两个寄存器的值为 0，采样点和 SCLK 之间不存在延迟。

该功能使用时有以下限制：

1. 该功能只能在 SDR 模式下使用，DDR 模式下不能使用
2. 采样延伸设置的最大值需要小于等于 SCLK 周期的一半。比如，当 ESMC_BAUD 设置为 2 时，ESMC_STR 和 ESMC_XSTR 寄存器最大值只能设置为 0x1。

30.3.6 DMA 模式

通过使能 DMA_EN（ESMC_CR[3]）可进行 DMA 传输。启用后 ESMC 的硬件会自动产生 DMA 发送请求或 DMA 接收请求信号。在进行 DMA 操作之前，首先应该对 ESMC 进行初始化，配置好相应的寄存器。在 DMA 模式下，ESMC 会生成 DMA 发送/接收请求信号，表示 ESMC 期望进行数据传输，一旦 ESMC 将 nCS 引脚拉低并且将指令和地址传输到 SPI Flash，就会开始数据传输。

ESMC 支持 DMA burst 读功能（不支持 DMA burst 写功能），ESMC 需要先设置 ESMC_IER[2]=1，设置以后 ESMC 只会在 RXFIFO 满了以后，发送 DMA 请求。

另外, ESMC 使用 DMA 模式时还需要注意以下几点：

1. DMA 模式下，需要先将 DMA 模块配置好，才能开始进行传输。
2. DMA 模式下，必须要在写入指令之前将 DMA_EN（ESMC_CR[3]）配置成 1。

3. DMA 接收模式下，只能接受字读，不支持字节和半字读。DMA 发送模式下，支持字节，半字，字写。

4. DMA 操作只能在寄存器模式下操作使用(不能在 XIP 模式下)。

5. DMA burst 读模式下，传输的数据量必须配置为单次 burst 数据的整数倍。

30.3.7 ESMC 中断

ESMC 有一个中断启用 GIE（ESMC_CR[4]），它使能/禁用 ESMC 中的所有中断源。

ESMC 有四个与中断相关的控制寄存器 ESMC_IER1，ESMC_IER2，ESMC_IFR1 和 ESMC_IFR2。

不管全局 IE 位如何，ESMC 的几个中断源都有自己的屏蔽位。IE 屏蔽位都具有相应的标志位。当特定事件发生时，无论是否设置了相应的 IE 位，标志位都会被自动设置。如果想要生成中断请求，必须同时设置全局 GIE（ESMC_CR[4]）位和对应的 IE（ESMC_IER1 和 ESMC_IER2）屏蔽位。

可在以下事件中产生中断：

Interrupt Event	Event Flag	Enable Control Bit
ADDR DONE	ADDRDONEIF	ADDRDONEIE
RX FIFO Word	RXFIFOWIF	RXFIFOWIE
DATA WAIT	DATAWAITIF	DATAWAITIE
IDLE	IDLE_DONEIF	IDLE_DONEIE
RX FIFO FULL	RXFIFOFIF	RXFIFOFIE
RX FIFO HALF FULL	RXFIFOHFIF	RXFIFOHFIE
FIFO not Empty	RXFIFONEIF	RXFIFONEIE
COMMAND DONE	COMMANDDONEIF	COMMANDDONEIE
RXTIMEOUT	RXTOUTIF	-
XIP WE Instr set	XIPWEInstrsetIF	-
TX WORD OVER	TXWORDOVIF	TXWORDOVIE
TX FIFO OVER	TXFIFOOVIF	TXFIFOOVIE
TX FIFO full	TXFIFOFIF	TXFIFOFIE
TX FIFO HALF FULL	TXFIFOHFIF	TXFIFOHFIE
TX FIFO Empty	TXFIFOEIF	TXFIFOEIE
XIP Rd Instr Set	XIPRDInstrSetIF	-

注：以下是 ESMC_SR 和 ESMC_IFR1/2 中具有类似功能寄存器的区别。

ESMC_SR. IDLE_STATE 和 ESMC_IFR.IDLE_DONE 的区别

ESMC_SR.IDLE_STATE 状态寄存器表示 ESMC 的状态机进入了 IDLE 状态。而 ESMC_IFR.IDLE_DONE 表示空闲中断请求，是指从非空闲到空闲状态的转换。比如，在高分频时，ESMC 在空闲状态，如果写入新的命令以后，ESMC_IFR.IDLE_DONE 会立刻从 1 变成 0。而 ESMC_SR.IDLE_STATE 需要在 nCS 引脚拉高以后(ESMC 不是空闲状态以后)从 1 变成 0。

在高分频时使用 ESMC_SR.IDLE_STATE 时，需要在发送完指令以后，先读 ESMC_IFR.CMDIF 标志位，在 ESMC_IFR.CMDIF 置位以后再读该寄存器。

ESMC_IFR2.TXFIFOOV 和 ESMC_SR.TXFIFOOV 具有相同的功能。ESMC_IFR.DATAWAIT 和 ESMC_SR. DATAWAIT 具有相同的功能。

这两组信号的区别是：ESMC_IFR.DATAWAIT 和 ESMC_IFR2. ESMC_IFR2.TXFIFOOV 的是中断请求信号，ESMC_SR. DATAWAIT 和 ESMC_SR. TXFIFOOV 是状态位信号。

30.4 ESMC 寄存器描述(只支持字节访问)

ESMC 寄存器基地址：0xA000 1000

30.4.1 配置寄存器（ESMC_CR）

Address offset：0x00

Reset value： 0xD2

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
CR	**0x00**	SPIEN	XIPEN	Res	GIE	DMAEN	2xQUAD	Res	SOFTRST	0xD2

Bit	Name	R/W	Reset Value	Function
7	SPIEN	RW	1	SPI 使能 0：SPI 禁止，所有内部寄存器和计数器停止 1：SPI 使能
6	XIPEN	RW	1	启用/禁用 XIP 使能 此位的任何更改，会终止 SPI 操作，清除 FIFO，并将 ESMC 驱动到空闲状态。清除 XIPEN 位后，只能通过间接模式编程命令和操作 0：禁止 XIP 功能 1：使能 XIP 功能
5	保留	-	0	-
4	GIE	RW	1	全局中断使能 0：关闭全局中断使能 1：打开全局中断使能(打开各个位的中断使能的同时需要将该位置 1)
3	DMAEN	RW	0	DMA 使能, 0：DMA 禁止 1：DMA 使能
2	2xQUAD	RW	0	双 Flash 模式 0：单 Flash 模式 1：双 Flash 模式
1	保留	-	1	-
0	SOFTRST	RW	0	软件复位 0：关闭软件复位 1：启用软件复位功能 注：软件复位时，ESMC_DATA 寄存器是清除指针（数据指向 RXFIFO 中的第一个数据），其余寄存器会被复位成默认值。

30.4.2 配置寄存器 2（ESMC_CR2)

Address offset：0x01

Reset value：0x40

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
CR2	**0x01**	XSPI_DUAL LAT_OFF	Res.	Res.	RXFIFO_CLR	TXFIFO_CLR	Res	CPOL	CPHA	0x40

Bit	Name	R/W	Reset Value	Function
7	XSPI_DUAL LAT_ OFF	RW	0	关闭 RWDSI 的 xSPI 模式 DUMMY 的双倍延迟。 1：禁用双倍延迟功能，xSPI 模式下，ESMC 发送的 DUMMY CYCLE 数是由 DCR[5:0]设置，周期数为 N-1 个 0：启用双倍延迟功能，xSPI 模式下，ESMC 发送的 DUMMY CYCLE 数为 2N-1 个
6	保留	-	1	-

5	保留	-	0	-
4	RXFIFO_CLR	RW	0	清除的 RX FIFO 指针
3	TXFIFO_CLR	RW	0	清除的 TX FIFO 指针
2	保留	-	0	-
1	CPOL	RW	0	时钟极性 0：空闲状态时，SCLK 保持低电平 1：空闲状态时，SCLK 保持高电平 注：CPOL 和 CPHA 需要配合使用，目前只支持 0x0 和 0x3 启用模式 3 时，建议先使能 CPOL 和 CPHA 寄存器，在配置波特率。
0	CPHA	RW	0	时钟相位 0：从第一个时钟边沿开始采样数据 1：从第二个时钟边沿开始采样数据 注：CPOL 和 CPHA 需要配合使用，目前只支持 0x0 和 0x3 启用模式 3 时，建议先使能 CPOL 和 CPHA 寄存器，在配置波特率。

30.4.3 时序控制寄存器（ESMC_TCR）

Address offset：0x02

Reset value： 0x11

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
TCR	**0x02**	HIGH[3:0]				HOLD[3:0]				0x11

Bit	Name	R/W	Reset Value	Function
7:4	HIGH[3:0]	RW	4’h1	HIGH 时间的周期数（以 SCLK 周期数为单位）： 1111 ： 周期数为 15 1110：周期数为 14 …… 0010：周期数为 2 0001：周期数为 1 0000：周期数为 1
3:0	HOLD[3:0]	RW	4’h1	建立和保持时间（以 HCLK 周期数为单位）： 建立时间： STR 模式 Mode0，DDR 模式 Mode0 和 Mode3： 1111 ： 插入 16 个系统时钟周期 1110：插入 15 个系统时钟周期 …… 0010：插入 3 个系统时钟周期 0001：插入 2 个系统时钟周期 0000：禁用 STR 模式 Mode3： 1111 ： 插入 15 个系统时钟周期 1110：插入 14 个系统时钟周期 …… 0010：插入 2 个系统时钟周期 0001：插入 1 个系统时钟周期 0000：禁用

				保持时间：(不管是 STR，DDR，Mode0，Mode3) 1111： 插入 16 个系统时钟周期 1110：插入 15 个系统时钟周期 …… 0010：插入 3 个系统时钟周期 0001：插入 2 个系统时钟周期 0000：禁用

30.4.4 波特率寄存器(ESMC_BAUD)

Address offset：0x03

Reset value： 0x04

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
BAUD	**0x03**	BAUD[7:0]								0x04

Bit	Name	R/W	Reset Value	Function
7:0	BAUD[7:0]	RW	8’h04	波特率寄存器用于定义 ESMC 的 SCLK 频率，基于系统时钟分频。 BAUD 设置为 0x0 和 0x1 时，为二分频；即 SCLK 时钟最快为系统时钟的二分频。 其他值计算公式为：SCLK = HCLK / BAUD；

30.4.5 SPI Flash 命令寄存器(ESMC_SFCR)

Address offset：0x04

Reset value： 0x0B

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
SFCR0 COMMANDONLY	**0x04**	SFCR0[7:0]								0x0B

Bit	Name	R/W	Reset Value	Function
7:0	SFCR0[7:0]	RW	8’h0B	SPI 指令（只发送指令） 指令格式只包含指令字节。地址，备用，Dummy，数据字节都不发送。

30.4.6 SPI 输出控制寄存器(ESMC_SOCR)

Address offset：0x05

Reset value： 0x02

Regis-ter	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Re-set
SOCR	**0x05**	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02

Bit	Name	R/W	Reset Value	Function
7	DDRDATA	RW	0	在数据接收/发送期间启用双数据速率操作。 设置此位后，数据在两个 SCLK 时钟边沿上移入/移出。
6	DDRADDR	RW	0	在闪存地址字节发送期间启用双数据速率操作。 设置此位后，地址在两个 SCLK 时钟边沿上移出。
5	DDRCMD	RW	0	在闪存指令字节发送期间启用双数据速率操作。

				设置此位后，指令在两个 SCLK 时钟边沿上移出。
4	MULTIADDR	RW	0	启用多位地址 设置此位后，地址字节和数据字节以相同的格式发送。 其中地址位分别以双、四或八进制格式发送，如果 MULTIADDR 清除，则表示地址以单 SPI 模式发送。
3	MULTICMD	RW	0	启用多位指令 设置此位后，指令字节和数据字节以相同的格式发送。 其中指令位分别以双、四或八进制格式发送，如果 MULTICMD 清除，则表示指令以单 SPI 模式发送。
2	SENM	RW	0	在 ADDR(8 位、24 位或 32 位)之后发送额外的 M[7:0]。
1:0	SPIMODE[1:0]	RW	2'h2	设置 SPI 传输格式: 2'b00 SINGLE SPI 2'b01 DUAL SPI 2'b10 QUAD SPI 2'b11 OCTAL SPI

30.4.7 Dummy 控制寄存器(ESMC_DCR)

Address offset：0x06

Reset value：0x80

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
DCR	**0x06**	REC	NO_CMD	DUMMY[5:0]						0x80

Bit	Name	R/W	Reset Value	Function
7	REC	RW	1	连续接收使能 设置后，ESMC 应执行闪存读取命令，在发送命令、地址和 DUMMY 字节后，ESMC 开始接收数据字节。 如果设置了这个位，ESMC 接收数据(读模式)，否则 ESMC 发送数据(写模式)。
6	NO_CMD	RW	0	设置此位后，将会省略指令，而以地址开始传输。
5:0	DUMMY[5:0]	RW	6'h0	定义地址传输后生成的 DUMMY 周期数。 DUMMY CYCLE 的设置还有部分寄存器（ESMC_CR2[7] , ESMC_SOCR[7], ESMC_CR3[5], ESMC_SOCR[1:0]）的设置相关，会有非 xSPI 模式，xSPI 模式关闭双倍延迟和 xSPI 启用双倍延迟这三种情况。

30.4.8 配置寄存器 3(ESMC_CR3)

Address offset：0x07

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
CR3	**0x07**	NO_DATA	Res	CMD_EXT	xSPI_RWDS	NO_ADDR	ADDR32BIT	ADDR16BIT	ADDR8BIT	0x00

Bit	Name	R/W	Reset Value	Function
7	NO_DATA	RW	0	当设置使能后，地址发送后不会进入数据等待状态。 这个位可以用于擦除命令，只发送指令和地址。
6	保留	-	0	-
5	CMD_EXT	RW	0	当设置使能后，ESMC 在发送命令后发送额外字节（M 寄存器的内容）。
4	xSPI_RWDS	RW	0	当设置使能后，支持 xSPI 模式 RWDS 引脚。

3	NO_ADDR	RW	0	当设置使能后，ESMC 不发送地址字节，指令字节后面跟着 DUMMY 周期（如果有）和数据字节。
2	ADDR32BIT	RW	0	启用 32 位地址
1	ADDR16BIT	RW	0	启用 16 位地址
0	ADDR8BIT	RW	0	启用 8 位地址

30.4.9 状态寄存器(ESMC_SR)

Address offset：0x14

Reset value： 0x10

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
SR	**0x14**	SPIF	DATAWAIT	TXFIFOOV	IDLE_STATE	XIPBUSY	TXBUSY	RXBUSY	SSACTIVE	0x10

Bit	Name	R/W	Reset Value	Function
7	SPIF	R	0	SPI 中断请求标志 0：没有中断请求挂起 1：至少有一个中断请求挂起。 该标志将一直置高，直到清除所有中断请求。
6	DATAWAIT	R	0	ESMC 状态机处于等待状态，如果是接收状态则是有数据要读取，如果是发送状态则是有数据等待发送 0：ESMC 状态机处于非等待状态 1：ESMC 状态机处于等待状态
5	TXFIFOOV	R	0	TX_FIFO 上溢标志位
4	IDLE_STATE	R	1	ESMC 处于 IDLE 状态标志位。
3	XIPBUSY	R	0	XIP 操作 BUSY 标志位
2	TXBUSY	R	0	间接模式写操作 BUSY 标志位
1	RXBUSY	R	0	间接模式读操作 BUSY 标志位
0	SSACTIVE	R	0	从机选中有效标志 如果任何 nCS 输出是有效的话，此位置为 1

30.4.10状态寄存器 2(ESMC_SR2)

Address offset：0x15

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
SR1	**0x15**	Res.	Res.	Res.	Res.	Res.	Res.	xSPI DUAL LATENCY	RWDSI_ND	0x00

Bit	Name	R/W	Reset Value	Function
7:2	保留	-	-	-
1	XSPI DUAL LATENCY	RC_W1	0	这个位表示启用了 xSPI 的双倍迟延时。 可以通过向该 bit 写 1 来清除该位。
0	RWDSI_ND	R	0	表示未检测到 Rwdsi 时钟。 如果在 xSPI 读操作中没有检测到 rwdsi 时钟，或者在超过 1 个 SCLK 周期的时间内没有检测 rwdsi，则置位此标志。

30.4.11 RX_FIFO 状态寄存器(ESMC_RXSTAT)

Address offset：0x16

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
RXSTAT	**0x16**	Res.	Res.	Res.	Res.	RXSTAT[3:0]				0x00

Bit	Name	R/W	Reset Value	Function
7：4	保留	-	-	-
0：3	RXSTAT	R	0	该寄存器显示 RX FIFO 中存储的数据字节数 1111：RXFIFO 中有 15 个数可供读取 1110：RXFIFO 中有 14 个数可供读取 ..... 0002：RXFIFO 中有 2 个数可供读取 0001：RXFIFO 中有 1 个数可供读取 0000：RXFIFO 中没有数据可供读取

30.4.12 中断标志寄存器(ESMC_IFR1)

Address offset：0x18

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
IFR	**0x18**	ADDRDONEIF	RXFIFOWIF	DATAWAITIF	IDLEDONEIF	RXFIFOFIF	RXFIFOHFIF	RXFIFONEIF	CMDIF	0x00

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIF	RC_W1	0	地址发送完成标志。当地址字段发出后该标志位置位。通过将新命令写入到 SFCR 寄存器或写入 1，该标志位自动清除。
6	RXFIFOWIF	R	0	该位表示 RX FIFO 包含至少 1 个字的数据。这意味着 RX FIFO 中至少存储有 4 个字节的数据。 注意：用户应避免在 RXFIFO 中少于一个字的时候读取数据。这会导致读取的数据不正确。
5	DATAWAITIF	R	0	此位表示 ESMC 当前执行闪存读写操作，且 ESMC 状态机处于等待状态，这意味着： 在写操作期间，ESMC 正在等待发送下一部分数据字节，FIFO 为空。 在读操作期间，ESMC 正在等待接收下一部分数据字节，FIFO 为满。 FIFO 写入或清空后，该位被清除，以便 ESMC 恢复当前闪存读/写操作。 0：未进入数据等待阶段，当前传输未结束 1：进入数据等待阶段，当前传输已经结束
4	IDLEDONEIF	R	0	空闲中断标志。 0：正在进行指令、地址、Dummy 或数据传输。 1：ESMC 完成操作，进入 IDLE 状态。
3	RXFIFOFIF	R	0	RX_FIFO 满

				0：RXFIFO 中至少有 1 个空间可供写入。 1：RXFIFO 完全满时，FIFO 内存中没有可用空间。
2	RXFIFOHFF	R	0	RX_FIFO 半满 RX FIFO 半满时（RXFIFO 的数据为 8），就会触发该中断。
1	RXFIFONEIF	R	0	FIFO 不为空 0：RXFIFO 中没有数据 1：RXFIFO 中包含至少 1 个字节的数据
0	CMDIF	RC_W1	0	指令发送完成标志 0：在 SFCR 寄存器载入新的指令或对该位写 1 时清除 1：指令字节发送完成时置位。

30.4.13中断使能寄存器(ESMC_IER1)

Address offset：0x19

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
IER	**0x19**	ADDRDONEIE	RXFIFOWORDIE	DATAWAITIE	IDLEDONEIE	RXFIFOFIE	RXFIFOHFIE	RXFIFONEIE	CMDIE	0x00

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIE	RW	0	地址发送完成中断使能 0：中断禁止 1：中断使能
6	RXFIFOWIE	RW	0	RX FIFO 接收字单位数据中断使能 0：中断禁止 1：中断使能
5	DATAWAITIE	RW	0	DATAWAIT 中断使能 0：中断禁止 1：中断使能
4	IDLEDONEIE	RW	0	空闲中断使能 0：中断禁止 1：中断使能
3	RXFIFOFIE	RW	0	RX FIFO 满中断使能 0：中断禁止 1：中断使能
2	RXFIFOHFIE	RW	0	RX FIFO 半满中断使能 0：中断禁止 1：中断使能
1	RXFIFONEIE	RW	0	FIFO 非空中断使能 0：中断禁止 1：中断使能
0	CMDIE	RW	0	指令发送完成中断使能 0：中断禁止 1：中断使能

30.4.14中断标志寄存器(ESMC_IFR2)

Address offset：0x1A

Reset value： 0x06

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
IFR2	**0x1A**	RXTOUTIF	XIPWEINSTRSETIF	TXWORDOVIF	TXFIFOOVIF	TXFIFOFULLIF	TXFIFOHEIF	TXFIFOEIF	XIPRDINSTRSETIF	0x06

Bit	Name	R/W	Reset Value	Function
7	RXTOUTIF	R	0	接收数据超时标志 如果 RXFIFOF 或 RXFIFOHF 中断被启用，RX TIMEOUT 中断也被启用。如果 RXFIFO 不为空，并且在 RXTOUT 寄存器中指定的时间内，没有新数据写入 RXFIFO，也没有字被读取，则产生超时中断。
6	XIPWEINSTRSETIF	R	0	表示总线正在进行 XIP 写访问。
5	TXWORDOVIF	RC_W1	0	此位表示要写入 TX FIFO 的数据没有按照字对齐。 当软件试图将超过 4 个字节的数据写入到 4 个字节的内存空间时，该位就会被置位。（例如，先写入单个字节，然后写入 4 字节的数据。相当于将 5 个字节写到 4 个字节的内存空间） 注：1. 加载到 TX FIFO 的数据应该始终与 word 对齐。 2. 不支持字和字节交叉写入的方法。 3. 该位仅在 TXFIFO 未满时设置。可以通过向该位写入 1 来清除改位。
4	TXFIFOOVIF	RC_W1	0	TX FIFO 溢出标志 当软件试图将一个新数据写入满的 TX FIFO 内存时，该位被置 1。 当该位被置 1 时，表示 TXFIFO 已经溢出，可以通过向该位写入 1 来清除该位。
3	TXFIFOFIF	R	0	TXFIFO 满标志 0：FIFO 中至少有 1 个空的空间。 1：FIFO 内存中没有可用空间。
2	TXFIFOHEIF	R	1	TXFIFO 半空标志 TX FIFO 半空时（TXFIFO 的数据为 8 时），就会触发该中断。
1	TXFIFOEIF	R	1	TXFIFO 空 0：TXFIFO 中至少有一个字节的数据 1：TXFIFO 完全为空。可以写入数据填充 TXFIFO。
0	XIPRDINSTRSETIF	R	0	表示总线正在进行 XIP 读访问。

30.4.15中断使能寄存器(ESMC_IER2)

Address offset：0x1B

Reset value： 0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
IER2	**0x1B**	Res	Res	TXWORDOVIE	TXFIFOOVIE	TXFIFOFIE	TXFIFOHEIE	TXFIFOEIE	Res	0x00

Bit	Name	R/W	Reset Value	Function
7:6	保留	-	-	-
5	TXWORDOVIE	RW	0	发送字溢出中断使能 0：中断禁止 1：中断使能
4	TXFIFOOVIE	RW	0	TX FIFO 溢出中断使能 0：中断禁止 1：中断使能
3	TXFIFOFIE	RW	0	TX FIFO 满中断使能 0：中断禁止 1：中断使能
2	TXFIFOHEIE	RW	0	TX FIFO 半空中断使能 0：中断禁止 1：中断使能
1	TXFIFOEIE	RW	0	TX FIFO 空中断使能 0：中断禁止 1：中断使能
0	保留	-	-	-

30.4.16采样延伸寄存器（ESMC_STRR）

Address offset：0x1C

Reset value：0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
STR	**0x1C**	Res	Res	SMP[5:0]						0x00

Bit	Name	R/W	Reset Value	Function
7:6	保留	-	-	-
5:0	SMP[5:0]	RW	6’h0	该位定义了可配置数据阶段 ESMC 的采样点相对于 SCLK 时钟边沿延伸的 HCLK 数量。 111111：周期数为 63 111110：周期数为 62 …… 011111：周期数为 31 …… 000010：周期数为 2 000001：周期数为 1 000000：无采样点延迟。在 SCLK 时钟边沿对数据进行采样

30.4.17接收超时寄存器(ESMC_RXTOUT)

Address offset：0x30

Reset value：0x3F

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
RXTOUT	**0x30**	RXTOUT[7:0]								0x3F

Bit	Name	R/W	Reset Value	Function
7：0	RXTOUT	RW	8’h3F	这个寄存器用来设置 RX TIMEOUT 的时间周期（以 SCLK 周期为单位）。

30.5 XIP 模式控制寄存器–读操作

30.5.1 XIP SPI Flash 命令寄存器(ESMC_XSFCR)

Address offset：0x20

Reset value：0x0B

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XSFCR	**0x20**	XSFCR[7:0]								0x0B

Bit	Name	R/W	Reset Value	Function
7:0	XINS[7:0]	RW	8’hB	XIP 模式下指令

30.5.2 XIP SPI 输出控制寄存器(ESMC_XSOCR)

Address offset：0x21

Reset value： 0x02

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Re-set
XSOCR	**0x21**	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02

Bit	Name	R/W	Reset Value	Function
7	DDRDATA	RW	0	在数据接收/发送期间启用双数据速率操作。 设置此位后，数据在两个 SCLK 时钟边沿上移入/移出。
6	DDRADDR	RW	0	在闪存地址字节发送期间启用双数据速率操作。 设置此位后，地址在两个 SCLK 时钟边沿上移出。
5	DDRCMD	RW	0	在闪存指令字节发送期间启用双数据速率操作。 设置此位后，指令在两个 SCLK 时钟边沿上移出。
4	MULTIADDR	RW	0	启用多位地址 设置此位后，地址字节和数据字节以相同的格式发送。 其中地址位分别以双、四或八进制格式发送，如果 MULTIADDR 清除，则表示地址以单 SPI 模式发送。
3	MULTICMD	RW	0	启用多位指令 设置此位后，指令字节和数据字节以相同的格式发送。 其中指令位分别以双、四或八进制格式发送，如果 MULTICMD 清除，则表示指令以单 SPI 模式发送。
2	SENM	RW	0	在 ADDR(8 位、24 位或 32 位)之后发送额外的 M[7:0]。
1:0	SPIMODE[1:0]	RW	2’h2	设置 SPI 传输格式: 2’b00 SINGLE SPI 2’b01 DUAL SPI

				2’b10 QUAD SPI 2’b11 OCTAL SPI

30.5.3 XIP 读 Dummy 控制寄存器(ESMC_XDCR)

Address offset：0x22

Reset value： 0x80

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XDCR	**0x22**	RES	NO_CMD	DUMMY[5:0]						0x80

Bit	Name	R/W	Reset Value	Function
7	保留	-	1	-
6	NO_CMD	RW	0	设置此位后，将会省略指令，而以地址开始传输。
5:0	DUMMY[5:0]	RW	6’h0	定义地址传输后生成的 DUMMY 周期数。 DUMMY CYCLE 的设置还有部分寄存器（ESMC_XCR2[7] , ESMC_XSOCR[7], ESMC_XCR3[5], ESMC_XSOCR[1:0]）的设置相关，会有非 xSPI 模式，xSPI 模式关闭双倍延迟和 xSPI 启用双倍延迟这三种情况。

30.5.4 XIP 配置寄存器 3(ESMC_XCR3)

Address offset：0x23

Reset value： 0x00

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XCR3	**0x23**	NO_DATA	Res	CMD_EXT	xSPI_RWDS	NO_ADDR	ADDR32BIT	ADDR16BIT	ADDR8BIT	0x00

Bit	Name	R/W	Reset Value	Function
7	NO_DATA	RW	0	当设置使能后，地址发送后不会进入数据等待状态。 这个位可以用于擦除命令，只发送指令和地址。
6	保留	-	0	-
5	CMD_EXT	RW	0	当设置使能后，ESMC 在发送命令后发送额外字节（M 寄存器的内容）。
4	xSPI_RWDS	RW	0	当设置使能后，支持 xSPI 模式 RWDS 引脚。
3	NO_ADDR	RW	0	当设置使能后，ESMC 不发送地址字节，指令字节后面跟着 DUMMY 周期（如果有）和数据字节。
2	ADDR32BIT	RW	0	启用 32 位地址
1	ADDR16BIT	RW	0	启用 16 位地址
0	ADDR8BIT	RW	0	启用 8 位地址

30.6 XIP 模式控制通用寄存器

30.6.1 XIP 模式寄存器（ESMC_XMODE）

Address offset：0x24

Reset value：0x00

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XMODE	**0x24**	M[7:0]								0x00

Bit	Name	R/W	Reset Value	Function
7：0	M[7:0]	RW	8'h0	只有在 ESMC_XCR3 寄存器中设置了 SEND_M 位，该寄存器的内容才会在地址字节之后发送。

30.6.2 XIP 写模式寄存器（ESMC_XMODE_WE）

Address offset：0x25

Reset value： 0x00

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Re-set
XMODE_WE	**0x25**	M[7:0]								0x00

Bit	Name	R/W	Reset Value	Function
7：0	M[7:0]	RW	8'h0	只有在 ESMC_XCR3_WE 寄存器中设置了 SEND_M 位，该寄存器的内容才会在地址字节之后发送。

30.6.3 XIP 从选择输出控制寄存器(ESMC_XSSOCR)

Address offset：0x26

Reset value： 0x01

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XSSOCR	**0x26**	Res	SS_CLR_RQ	SS[5:0]						0x01

Bit	Name	R/W	Reset Value	Function
7	保留	-	-	-
6	SS_CLR_RQ	RW	0	清除从机选择输出请求 向该位写 1 后，会将从机选择输出变为非活动的高电平状态。当前的操作会被立刻终止，ESMC 会进入 IDLE 状态,并且 nCS 引脚会被停用。操作完成以后该位会被自动清零。
5:0	SS[5:0]	RW	6'h1	从机选择启用/禁止 0：禁用相应的 SPI 从机，nCS 引脚被强制为非活动状态。 1：启用相应的 SPI 从机，进行 SPI 操作，在 SPI 传输期间激活 nCS 引脚

30.6.4 XIP 从选择输出控制寄存器(ESMC_XSSOCR_WE)

Address offset：0x27

Reset value： 0x01

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XSSOCR	**0x27**	Res	SS_CLR_RQ	SS[5:0]						0x01

Bit	Name	R/W	Reset Value	Function
7	保留	-	-	-
6	SS_CLR_RQ	RW	0	清除从机选择输出请求 向该位写 1 后，会将从机选择输出变为非活动的高电平状态。当前的操作会被立刻终止，ESMC 会进入 IDLE 状态,并且 nCS 引脚会被停用。操作完成以后该位会被自动清零。
5:0	SS[5:0]	RW	6'h1	从机选择启用/禁止 0：禁用相应的 SPI 从机，nCS 引脚被强制为非活动状态。 1：启用相应的 SPI 从机，进行 SPI 操作，在 SPI 传输期间激活 nCS 引脚

30.6.5 XIP 模式采样拉伸寄存器(ESMC_XSTRR)

Address offset：0x31

Reset value： 0x00

本寄存器支持 soft_rst 复位

Address offset：0x1C

Reset value：0x00

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XSTRR	**0x31**	Res	Res	SMP[5:0]						0x00

Bit	Name	R/W	Reset Value	Function
7:6	保留	-	-	-
5:0	SMP[5:0]	RW	6'h0	该位定义了可配置数据阶段 ESMC 的采样点相对于 SCLK 时钟边沿延伸的 HCLK 数量。 111111：周期数为 63 111110：周期数为 62 …… 011111：周期数为 31 …… 000010：周期数为 2 000001：周期数为 1 000000：无采样点延迟。在 SCLK 时钟边沿对数据进行采样

30.7 XIP 模式控制寄存器–写入操作

30.7.1 XIP 模式写入 SPI flash 命令寄存器(ESMC_XSFCR_WE)

Address offset：0x28

Reset value：0x32

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XSFCR_WE	**0x28**	XSFCR[7:0]								0x32

Bit	Name	R/W	Reset Value	Function
7：0	XSFCR_WE[7:0]	RW	8'h32	XIP 读模式下指令

30.7.2 XIP 模式读取 SPI 输出控制寄存器（ESMC_XSOCR_WE)

Address offset：0x29

Reset value：0x02

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Re-set
XSOCR_WE	**0x29**	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02

Bit	Name	R/W	Reset Value	Function
7	DDRDATA	RW	0	在数据接收/发送期间启用双数据速率操作。 设置此位后，数据在两个 SCLK 时钟边沿上移入/移出。
6	DDRADDR	RW	0	在闪存地址字节发送期间启用双数据速率操作。 设置此位后，地址在两个 SCLK 时钟边沿上移出。
5	DDRCMD	RW	0	在闪存指令字节发送期间启用双数据速率操作。 设置此位后，指令在两个 SCLK 时钟边沿上移出。
4	MULTIADDR	RW	0	启用多位地址 设置此位后，地址字节和数据字节以相同的格式发送。 其中地址位分别以双、四或八进制格式发送，如果 MULTIADDR 清除，则表示地址以单 SPI 模式发送。
3	MULTICMD	RW	0	启用多位指令 设置此位后，指令字节和数据字节以相同的格式发送。 其中指令位分别以双、四或八进制格式发送，如果 MULTICMD 清除，则表示指令以单 SPI 模式发送。
2	SENM	RW	0	在 ADDR(8 位、24 位或 32 位)之后发送额外的 M[7:0]。
1:0	SPIMODE[1:0]	RW	2'h2	设置 SPI 传输格式: 2'b00 SINGLE SPI 2'b01 DUAL SPI 2'b10 QUAD SPI 2'b11 OCTAL SPI

30.7.3 XIP 写 Dummy 周期控制寄存器(ESMC_XDCR_WE)

Address offset：0x2A

Reset value： 0x00

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XDCR_WE	**0x2A**	RES	NO_CMD	DUMMY[5:0]						0x00

Bit	Name	R/W	Reset Value	Function
7	保留	-	0	-
6	NO_CMD	RW	0	设置此位后，将会省略指令，而以地址开始传输。
5:0	DUMMY[5:0]	RW	6'h0	定义地址传输后生成的 DUMMY 周期数。 DUMMY CYCLE 的设置还有部分寄存器（ESMC_XCR2_WE[7] , ESMC_XSOCR_WE[7], ESMC_XCR3_WE[5],

				ESMC_XSOCR_WE[1:0]）的设置相关，会有非 xSPI 模式，xSPI 模式关闭双倍延迟和 xSPI 启用双倍延迟这三种情况。

30.7.4 XIP 写入配置寄存器 3(ESMC_XCR3_WE)

Address offset：0x2B

Reset value： 0x00

本寄存器支持 soft_rst 复位

Register	ADDR	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
XCR3_WE	**0x2B**	NO_DATA	Res	CMD_EXT	xSPI_RWDS	NO_ADDR	ADDR32BIT	ADDR16BIT	ADDR8BIT	0x00

Bit	Name	R/W	Reset Value	Function
7	NO_DATA	RW	0	当设置使能后，地址发送后不会进入数据等待状态。 这个位可以用于擦除命令，只发送指令和地址。
6	Res		0	
5	CMD_EXT	RW	0	当设置使能后，ESMC 在发送命令后发送额外字节（M 寄存器的内容）。
4	xSPI_RWDS	RW	0	当设置使能后，支持 xSPI 模式 RWDS 引脚。
3	NO_ADDR	RW	0	当设置使能后，ESMC 不发送地址字节，指令字节后面跟着 DUMMY 周期（如果有）和数据字节。
2	ADDR32BIT	RW	0	启用 32 位地址
1	ADDR16BIT	RW	0	启用 16 位地址
0	ADDR8BIT	RW	0	启用 8 位地址

30.8 ESMC 寄存器描述

ESMC 寄存器基地址：0xA000 1000

30.8.1 24 位地址寄存器（ESMC_ADDR24)

Address offset:0x08

Reset value:0x0000000B

该寄存器只能按字（32 位）访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR [23:8]															
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**
ADDR [7:0]								SFCR [7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	ADDR[23:0]	RW	24’h0	24 位地址 ESMC_CR3[2] = 0（默认 24 位地址模式）时，使用 24 位地址地址寄存器值以当前选择的 SPI 模式(扩展 SPI、单 SPI、双 SPI、四 SPI 或八 SPI)通道发送。地址寄存器应在 ESMC 加载 ESMC_SFCR 寄存器之前写入。
7:0	SFCR[7:0]	RW	8’h0B	SPI 指令设置

30.8.2 32 位地址寄存器（ESMC_ADDR32）

Address offset： 0x0C

Reset value： 0x0001 0000

该寄存器只能按字（32 位）访问

SS0~1 用于配置 SPI 主传输期间应驱动哪个从选择输出。寄存器的内容自动分配给 SS1O-SS0O。该寄存器允许使用多达 2 个不同的 SPI 闪存轻松进行 SPI 操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.								SSSET	SS CLR	SS[5:0]					
-								RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MREG[7:0]								ADDR[31:24]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	SSSET	RW	0	从机选择输出设置请求 向该位写 1 后，会使 SS[5:0]选择的从机选择输出(nCS 引脚)被驱动。一旦 nCS 引脚输出激活，该位将自动清除。
22	SSCLR	RW	0	清除从机选择输出请求 向该位写 1 后，会将从机选择输出变为非活动的高电平状态。当前的操作会被立刻终止，ESMC 会进入 IDLE 状态,并且 nCS 引脚会被停用。操作完成以后该位会被自动清零。
21:16	SS[5:0]	RW	6’h1	从机选择启用/禁止 0：禁用相应的 SPI 从机，nCS 引脚被强制为非活动状态。 1：启用相应的 SPI 从机，进行 SPI 操作，在 SPI 传输期间激活 nCS 引脚
15:8	MREG	RW	8’h0	只有在 CR3 寄存器中设置了 SEND_M 位时，该寄存器的内容才会在地址字节之后发送。
7:0	ADDR[31:24]	RW	8’h0	32 位地址 ESMC_CR3[2] = 1（32 位地址使能）时，使用 32 位地址

30.8.3 数据寄存器（ESMC_DATA）

Address offset： 0x10

Reset value： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA [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
DATA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DATA[31:0]	RW	32’h0	ESMC 要接收/发送的数据 闪存写入操作期间用于数据传输的数据缓冲区，读取命令期间用于数据存储的数据缓冲区。这些数据可用于两个方向。当数据缓冲存储器启用时，这些寄存器在读写方向上都指向 FIFO 顶部。

30.8.4 传输计数寄存器(ESMC_BCR)

Address offset： 0x2C

Reset value： 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCR[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**
BCR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	BCR[31:0]	RW	32’h0	读写数据个数 ESMC 支持定长的数据传输，默认情况下，只要 RXFIFO/TXFIFO 具有至少一个空闲空间/至少一个数据，ESMC 就会从外部存储器读取/写入数据。一旦 RXFIFO/TXFIFO 的存储空间满/空，ESMC 将进入 WAIT 状态，等待主机从 RXFIFO/TXFIFO 读取数据或者写入数据。在恢复下次传输之前，RXFIFO 必须完全为空，或者将新数据写到 TXFIFO 中。 在 ESMC 读取指定数量的字节后，nCS 引脚被停用，ESMC 进入空闲状态，等待下一个命令执行。 该寄存器允许用户命令接收 2^{32}-1 字节的数据。

31. SDIO 接口（SDIO）

31.1 简介

SD/SDIO MMC 卡主机模块在 AHB 外设总线和多媒体卡(MMC)、SD 存储卡、SDIO 卡和 CE-ATA 设备间提供了操作接口。

MMC 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获得。

CE-ATA 系统规格书可以在 CE-ATA 工作组的网站上获得。

支持以下功能：

- 支持 SD 卡 2.0 版本
- 支持 SD I/O 卡 2.0 版本
- 支持 MMC4.2 版本
- 支持 CE-ATA 1.1 版本
- 支持发送命令和向主机处理器申请中断

注：

SDIO 不支持 SPI 模式的通信模式

只支持 I/O 模式的 SD 卡或复合卡中的 I/O 部分,不支持 SD 存储设备中所有需要的命令。这里有些命令在 SD I/O 卡中不起作用，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。

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。

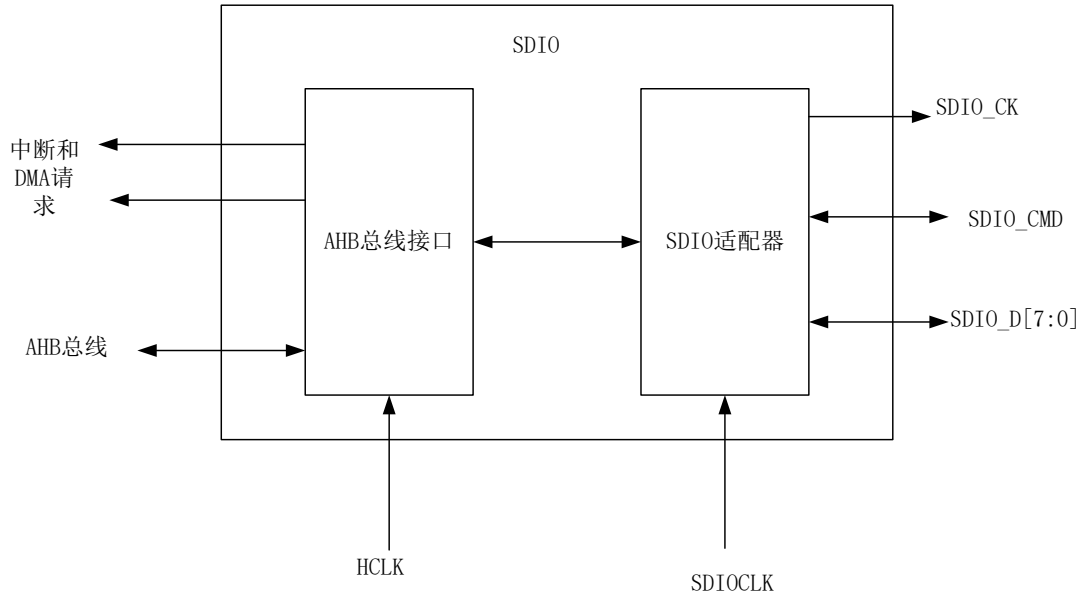


图 31-1 SDIO 框图

复位后默认情况下 SDIO_D[0]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 MMC 接到了总线上，则 SDIO_D[0]、SDIO_D[3:0]或 SDIO_D[7:0]可以用于数据传输。

如果一个 SD 或 SD I/O 卡接到总线，可通过主机配置数据传输使用 SDIO_D[0] 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)。

下表适用于 MMC/SD/SD I/O 卡总线：

表 31-1 SDIO 引脚定义

引脚	方向	说明
SDIO_CK	输出	MMC/SD/SDIO 卡时钟。这是从主机到卡的时钟线
SDIO_CMD	双向	MMC/SD/SDIO 卡命令。这是双向的命令/响应信号线
SDIO_D[7:0]	双向	MMC/SD/SDIO 卡数据。这些是双向的数据总线。

31.2 SDIO 适配器

SDIO 适配器包括控制单元、命令单元和数据单元，可以向卡生成信号。这些信号的具体描述如下：

SDIO_CLK：SDIO 控制器提供给卡的时钟。每个时钟周期在命令线(SDIO_CMD)和所有的数据线(SDIO_D)上直接发送一位命令或数据。对于 MMC 卡 V3.31 版本，SDIO_CLK 频率可以在 0 MHz 到 20 MHz 之间，对于 MMC 卡 V4.2 版本可以在 0 MHz 到 48MHz 之间，对于 SD 或 SD I/O 卡可以在 0 MHz 到 25MHz

SDIO_CMD：该信号是双向命令通道，用于卡的初始化和命令的传输。命令从 SDIO 控制器发送到卡，响应从卡发送到主机。

SDIO_D[7:0]：这些信号线都是双向数据通道。数据信号线操作在推挽模式。每次只有卡或者主机会驱动这些信号。默认情况下，上电或者复位后仅 D0 用于数据传输。SDIO 适配器可以配置更宽的数据总线用于数据传输，使用 D0-D3 或者 D0-D7(仅适用于 MMCV4.2)。SDIO 对数据信号线 D1-D7 有内部上拉。在进入 4 位模式后，卡断开 D1 和 D2 的内部上拉（D3 内部上拉保持不变是由于 SPI 模式下 CS 片选的使用）。相应地，在进入 8 位模式后，断开 D1,D2 和 D4-D7 的内部上拉（D3 内部上拉保持不变是由于 SPI 模式下 CS 片选的使用）。

SDIO 适配器是 SD/SD I/O /MMC/CE-ATA 的接口，它由 3 个子单元组成：

31.2.1 控制单元

控制单元包含电源管理功能和时钟管理功能用于输出时钟。电源管理是由 SDIO_PWRCTL 寄存器控制的，实现电源的掉电和上电。通过设置 SDIO_CLKCR 的 PWRSAV 位来配置省电模式，实现当总线空闲时，关闭 SDIO_CLK。时钟管理向卡生成 SDIO_CLK 时钟信号。

31.2.2 命令单元

命令单元实现向卡发送和接收命令。数据传输流由命令状态机（CSM）控 制。对 SDIO_CMD 寄存器进行一次写操作并设置该寄存器的 start_cmd 位为 1 后，命令传输开始。首先向卡发送一个命令，这个命令包含 48 位，通过 SDIO_CMD 线发出，每个 SDIO_CLK 发送一个比特数据。这 48 位命令包含 1 位起始位、1 位传输位、6 位命令索引（由 SDIO_CMD 寄存器的 cmd_index 位定义）、32 位参数（由 SDIO_CMDARG 定义）、7 位 CRC 和 1 位停止位。然后接收来自卡的响应，响应分为 48 位的短响应和 136 位的长响应，响应都存在 SDIO_RESP0 - SDIO_RESP3 寄存器中。

■ 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(见下图)。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。

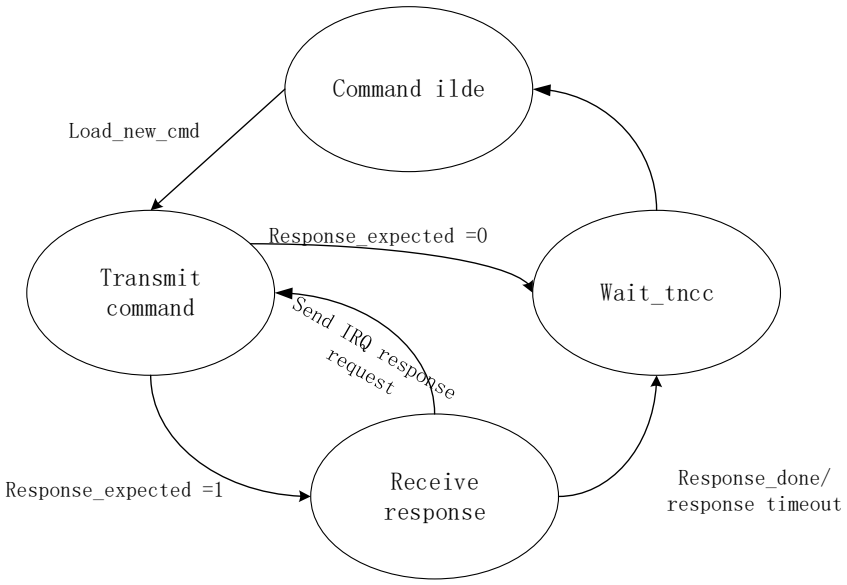


图 31-2 SD_MMC 命令通道状态机（CPSM）

命令通道状态机根据 CMD 寄存器位值执行以下功能：

1. send_initialization – 在发送命令之前发送 80 个时钟的初始化序列。

2. response_expected – 对命令的预期响应。命令发出后，命令通道状态机接收 48 位或 136 位响应，发送给 SDIO 适配器。如果在超时寄存器中编程的时钟数内未收到卡响应的起始位，则在中断状态寄存器中设置响应超时和命令完成位。如果未设置响应预期位，则命令发出后，在中断状态寄存器中设置命令完成位。

3. response_length – 如果设置了该位，则接收到 136 位的响应；如果未设置，则接收 48 位响应。

4. check_response_crc – 如果设置了该位，命令通道将在响应中接收到的 CRC7 与内部生成的 CRC7 进行比较。如果两者不匹配，则响应 CRC 错误信号发送给总线接口;在中断状态寄存器中设置响应 CRC 错误位。

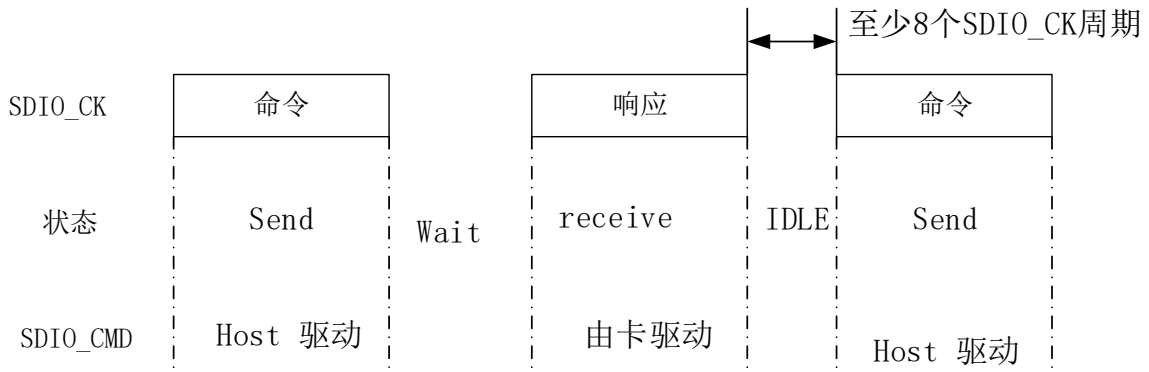


图 31-3 SDIO 命令传输

■ 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 MMC V3.31 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 MMC、SD 存储卡和 SDIO 卡上一般的命令格式。

CE-ATA 命令是 MMC V4.2 命令的扩充，所以具有相同的格式。

命令通道为半双工模式，这样命令和响应可以发送和接收。如果 CPSM 不处在发送状态，SDIO_CMD 输出处于高阻状态。SDIO_CMD 上的数据与 SDIO_CK 的上升沿同步。

表 31-2 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:0]	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。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

如果响应不包含 CRC(如 CMD1 的响应)，设备驱动应该忽略 CRC 错误状态。

表 31-3 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:0]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

表 31-4 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：48 位还是 136 位。命令通道中的状态标志见下表:

表 31-5 命令通道状态标志

标志	说明
CMDREND	响应的 CRC 正确
CCRCFAIL	响应的 CRC 错误
CMDSENT	命令(不需要响应的命令)已经送出
CTIMEOUT	响应超时
CMDACT	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 校验值是一个 7 位的数值。

31.2.3 数据单元

数据单元实现主机与卡之间的数据传输。当数据宽度为 8 位时，数据传输使用 SDIO_D[7:0]信号线；当数据宽度为 4 位时，数据传输使用 SDIO_D[3:0]信号线；当数据宽度为 1 位时，数据传输使用 SDIO_D[0]信号线。数据传输流由数据状态机(DSM)控制。

- 数据发送状态机(DTSM)

DTSM 工作在 SDIO_CK 频率，卡总线信号与 SDIO_CK 的上升沿同步。

如图所示,数据发送状态机在收到数据写入命令的响应后两个时钟开始数据发送；即使命令通道状态机检测到错误或响应 CRC 错误,也会发生这种情况。如果由于响应超时而未从卡接收到响应,则不传输数据。

根据命令寄存器中 transfer_mode 位的值，数据发送状态机将数据以流或块的形式放在卡数据总线上。

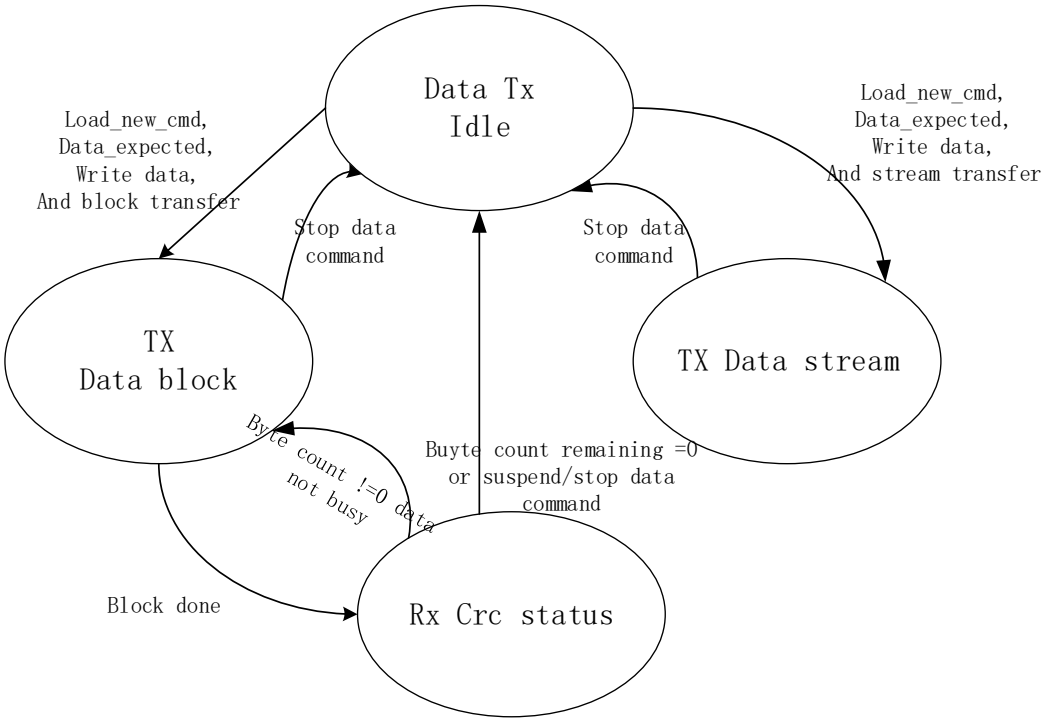


图 31-4 数据发送状态机（DTSM）

■ **数据接收状态机(DRSM)**

如图所示，数据接收状态机在数据读取命令的结束位后两个时钟周期接收数据，即使命令通道检测到响应错误或响应 CRC 错误也是如此。如果因为响应超时而没有收到卡的响应，则 SDIO 适配器不会收到数据传输完成的信号； 如果发送的命令是对卡的非法操作，则会发生这种情况，这会阻止卡开始读取数据传输。如果在数据超时之前未收到数据，则数据接收状态机向 SDIO 适配器发出数据超时信号并结束数据传输。根据命令寄存器中 transfer_mode 位的值，数据接收状态机以流或块的形式从卡数据总线获取数据。

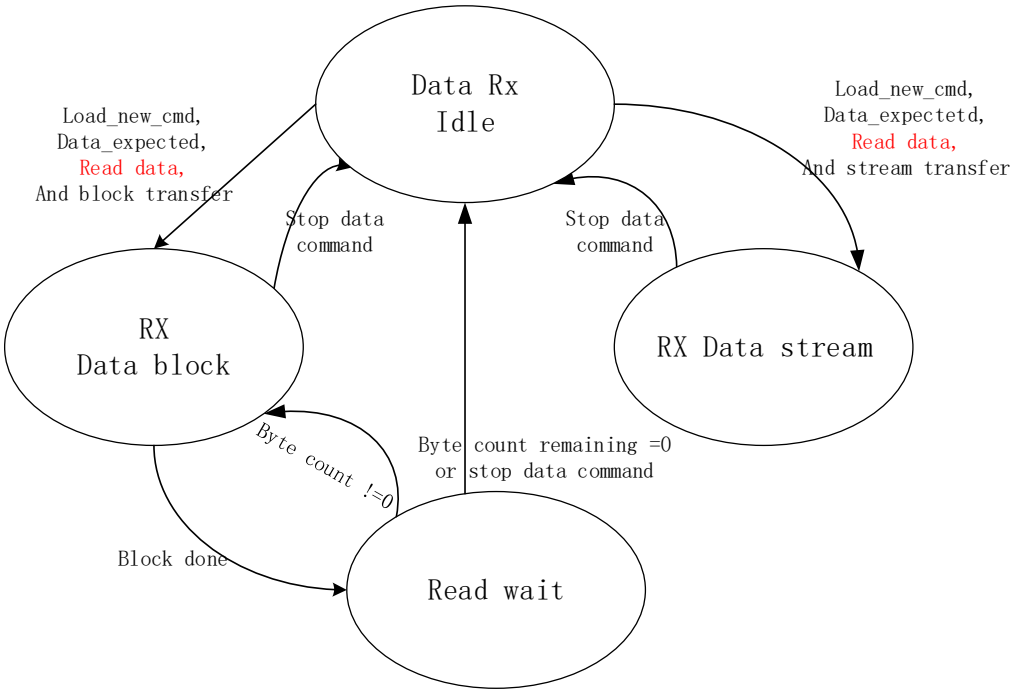


图 31-5 数据接收状态机（DRSM）

31.2.4 SDIO 时钟控制

SDIO 主要是有两个时钟控制，一个系统时钟 clk，另一个是 clk_in。具体的时钟描述如下：

1. clk，上升沿有效：系统时钟，
2. cclk_in 上升沿有效：卡时钟，这需要满足“clk >= cclk_in/10”
3. cclk_in_drv：延迟的 clk_in 驱动卡的时钟，是控制 data_out 和 cmd_out 时用的。
4. cclk_in_sample:延迟的 clk_in 采样卡时钟的输入，这需要考虑在 PAD 上过来的延迟，用于控制 ccmd_in 和 cdata_in 的采样
5. cclk_in 下降沿有效：clk_in 的下降沿，驱动 stop_clk_neg，fifo 读写的时钟

还有三个寄存器位可以用来进行时钟处理,即 clk 的 CKSEL,SMPEN,SMPCLKSEL 位：

1. CKSEL：用来控制输出命令，数据选择
2. SMPEN:预采样使能，
3. SMPCLKSEL:时钟相位选择，可以选择时钟的具体相位

31.3 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 FIFO。它包含一个数据通道、寄存器译码器、中断和 DMA 控制逻辑。

31.3.1 SDIO 中断

当至少有一个状态标志为高时，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。

31.3.2 SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程

SDIO/DMA：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 FIFO 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_FIFO 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)；WaitRest 置为 1(SDIO 卡主机等待响应)；start_cmd 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 CD（命令结束 ）中断，然后设置 SDIO 数据寄存器：DTEN 置为 1(使能 SDIO 卡主机发送数据)；DTDIR 置为 0(控制器至卡方向)； DTMODE 置为 0(块数据传送)；DMAEN 置为 1(使能 DMA)；DBLOCKSIZE 置为 9(512 字节)；其它域不用设置。
   e) 等待 SDIO_STA[10]=1。
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。

31.4 卡功能描述

31.4.1 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)。

31.4.2 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 SD I/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)。

31.4.3 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 V_{CC} 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 V_{CC} 条件下传送这些信息。当 SDIO 卡主机模块与卡的 V_{CC} 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 V_{CC} 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(SD I/O 卡的 CMD5)。SDIO 卡主机在执行这几个命令时会产生需要的 V_{CC} 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。

31.4.4 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 Fod 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 Fod 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。

31.4.5 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 命令接收新的数据时，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收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。

31.4.6 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。

31.4.7 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。

31.4.8 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。

31.4.9 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

31.4.10 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 129 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 31-6 卡状态

位	名称	类型	数值	说明	清除条件
31	ADDRESS_OUT_OF_RANGE	E R X	‘0’= 无错误 ‘1’= 错误	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	E X	‘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	S R	‘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_解锁_FAILED	E X	‘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	E R	‘0’= 无错误 ‘1’= 错误	之前的命令中 CRC 校验错误。	B
22	ILLEGAL_COMMAND	E R	‘0’= 无错误 ‘1’= 错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	E X	‘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	E R	‘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	E X	‘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	C
18:17	保留	-	-	-	-
16	CID/CSD_OVERWRITE	E X	‘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	E X	‘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	S X	‘0’= 不允许 ‘1’= 允许	执行命令时没有使用内部的 ECC。	A

13	ERASE_RESET		‘0’=未保护 ‘1’=已保护	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	S R	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 B 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	S R	‘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	E X	‘0’= 无错误 ‘1’= 错误	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	S R	‘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	E R	‘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				

31.4.11 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。

X 下表定义了不同的 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 31-7 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 ‘0’= 未处于保密模式 ‘1’= 处于保密模式	A
508:496	保留				
495:480	SD_CARD_TYPE	SR	00xxh’= 在物理规范版本 1.01 A ~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'= SD ROM 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_TIMEOUT	SR	擦除 UNIT_OF_ERASE_A 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 PW/2 的数值(PW 是写的性能)。

表 31-8 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 31-9 移动性能代码

SPEED_CLASS	数值定义
00h	未定义
01h	1 MB/S
02h	2 MB/S
……	……
Feh	254 MB/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 31-10 AU_SIZE 代码

SPEED_CLASS	数值定义
00h	未定义
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB
09h	4 MB
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 31-11 最大的 AU 长度

容量	16 – 64 MB	128 - 256 MB	512 MB	1 – 32 GB
最大的 AU 长度	512 KB	1MB	2 MB	4 MB

ERASE_SIZE

这个 16 位域给出了 NERASE，当 NERASE 个 AU 被擦除时，ERASE_TIMEOUT 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 31-12 ERASE_SIZE 代码

ERASE_CLASS	数值定义
0000h	不支持擦除的超时运算
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
……	……
FFFFh	65535 个 AU

ERASE_TIMEOUT

这 6 位给出了 TERASE, ERASE_SIZE 指示的多个 AU 被擦除, 这个数值给出从偏移量算起擦除超时。ERASE_TIMEOUT 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_TIMEOUT 的组 合，先确定 ERASE_TIMEOUT 再确定 ERASE_SIZE。

表 31-13 擦除超时代码

ERASE_TIMEOUT	数值定义
00	不支持擦除的超时运算
01	1 秒
02	2 秒
03	3 秒
.....	.....
63	63 秒

ERASE_OFFSET

这 2 位给出了 TOFFSET，当 ERASE_SIZE 和 ERASE_TIMEOUT 同为 0 时这个数值没有意义。

表 31-14 擦除偏移代码

ERASE_OFFSET	数值定义
0	0 秒
1	1 秒
2	2 秒
3	3 秒

31.4.12 SD 的 I/O 模式

SD 的 I/O 中断

为了让 SD I/O 卡能够中断 MMC/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 MMC/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 MMC/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 MMC/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，MMC/SD 模块所有数据线(SDIO/D[3:0])上提供上拉电阻。MMC/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

SD I/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

31.4.13 命令和响应

31.4.13.1 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 MMC 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：有一个 SD_STATUS(ACMD13)应用相关命令，如果在紧随 APP_CMD(CMD55)后收到 CMD13，它被解释为 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 MMC 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 MMC 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。

31.4.13.2 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。

31.4.13.3 命令格式

见命令格式表

MMC/SD 卡模块的命令

表 31-15 基于块传输的写命令

CMD 索引	类型	参数	响应格式	编写	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_BLOCKLEN 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。
CMD28	ac	[31:0] = 数据地址	R1b	SET_WRITE_PROT	如果卡有写保护功能，该命令设置指定组的写保护位。写保护特性设置在卡的特殊数据区(WP_GRP_SIZE)。
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PROT	如果卡有写保护功能，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	如果卡有写保护功能，该命令要求卡发送写保护位的状态
CMD31	保留				

表 31-16 基于块传输的写保护命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD28	ac	[31:0] =数据块址	R1b	SET_WRITE_PORT	定义在随后的多块读或写命令中需要传输块的数目
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PORT	按照 SET_BLOCKLEN 命令选择的长度写一个块
CMD30	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态
CMD31	保留				

表 31-17 擦除命令

CMD 索引	类型	参数	响应格式	编写	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	R1b	CLR_WRITE_PORT	按照 SET_BLOCKLEN 命令选择的长度写一个块
CMD36	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态
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]=填充物	R1	ERASE	擦除之前选择的数据块

表 31-18 I/O 模式命令

CMD 索引	类型	参数	响应格式	编写	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 MMC 标准中定义的与应用相关的寄存器。
CMD40	bcr	[31:0] = 写保护数据地址	R5	GO_IRO_STATE	置系统于中断模式。
CMD41	保留				

表 31-19 上锁命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD42	adtc	[31:0] = 填充位	R1b	LOCK_解锁	设置/清除密码或对卡上锁/解锁。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD43 …… CMD54	保留				

表 31-20 用相关命令

CMD 索引	类型	参数	响应格式	编写	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标准命令
CMD56	adtc	[31:1] = 填充位 [0] = RD/WR			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_BLOCKLEN 命令设置。
CMD57 … CMD59	保留				
CMD60 …… CMD63	保留给生产厂商				

31.5 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 7 种响应类型，它们的格式定义如下：

31.5.1 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 31-21 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

31.5.2 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。

31.5.3 R2(CSD、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 31-22 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

31.5.4 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 31-23 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	‘111111’	保留
0	1	1	结束位

31.5.5 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 31-24 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	‘111111’	保留
[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	‘111111’	CRC7
0		1	1	结束位

31.5.6 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 31-25 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 ORC
[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。

31.5.7 R5（中断请求）

仅适用于 MMC。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 31-26 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	‘111111’	CMD40
[39:8]参数域	[31:16]	16	X	成功的卡或主机的
	[15:0]	16	X	未定义。可以作为中
[7:1]		7	X	CRC7
0		1	1	结束位

31.5.8 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 31-27 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	‘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

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

31.6 编程序列

31.6.1 卡识别

主机复位后进入卡识别模式，寻找总线上的新卡。在卡识别模式下，主机复位所有的卡，验证工作电压范围，识别卡并询问每个卡的相对卡地址（RCA）。这个操作是在每个卡自己的命令信号线 CMD 上分别完成的。在卡识别模式中的所有数据通信只使用命令信号线（CMD）。

在卡识别过程中，卡应该工作在时钟频率为时钟速率 FOD（400 kHz)的情况下。

31.6.1.1 卡复位

命令 GO_IDLE_STATE（CMD0）是软件复位命令，并设置 MMC 和 SD 存储卡进入空闲状态（Idle State），不管当前卡的状态是什么。复位命令（CMD0）仅用于存储器或组合卡的存储器部分。为了重置只有 I/O 卡或组合卡的 I/O 部分，使用 CMD52 写 1 到 CCCR 的 RES 位。在非激活状态（Inactive State）的卡不受此命令的影响。

主机上电后，所有的卡都处于空闲状态（Idle State），包括之前已在非激活状态（Inactive State）的卡。上电或 CMD0 后，所有卡的 CMD 线处于输入模式，等待下一个命令的起始位。这些卡都是用默认的相对卡地址（RCA）初始化，并用默认 400 kHz 的时钟频率驱动器。

31.6.1.2 工作电压范围验证

在主机和卡之间开始通信时，主机可能不知道卡支持的电压，并且卡可能不知道主机能否提供其支持的电压。为了验证电压，下面的命令都在相关规范中定义。在协议规范中定义的命令包括：SEND_OP_COND（CMD1 用于 MMC), SD_SEND_OP_COND（ACMD41 用于 SD 存储卡)，IO_SEND_OP_COND（CMD5 用于 SD I/O 卡)，这些命令提供给主机一种机制去识别和拒绝那些不匹配主机所需的 V_{CC} 范围的卡。这是由主机发送所需的 V_{CC} 电压窗口作为此命令的操作数来实现的。如果卡不能在指定的范围内进行数据传输，必须从总线断开并进入非激活状态（Inactive State）。否则，该卡将响应返回它的 V_{CC} 范围。

如果该卡可以工作在所提供的电压下，响应将返回供电电压和在命令参数中设置的检查模式。如果该卡不能在提供的电压下工作，它不返回响应，并保持在空闲状态。初始化 SDHC 卡时强制性的在 ACMD41 命令之前发送 CMD8。收到 CMD8 是让该卡知道主机支持物理层 2.00 协议及卡支持高版本的功能。

31.6.1.3 卡识别过程

对于不同的卡，卡的识别过程不同。这些卡包括 MMC、CE-ATA、SD，或 SD I/O 卡。支持所有类型的 SD I/O 卡，即 SDIO_IO_ONLY 卡、SDIO_MEM_ONLY 卡和 SDIO COMBO 卡。卡识别过程步骤如下：

1. 检测卡是否连接。

2. 识别卡的类型：SD 卡、MMC(CE-ATA)或 SD I/O 卡。

– 发送 CMD5 命令。如果主机接收到响应，则是 SD I/O 卡；

– 如果没有响应，发送 ACMD41。如果主机接收到响应，则是 SD 卡；

– 否则，是 MMC 或者 CE-ATA 设备。

3. 根据卡的类型初始化卡。

使用 FOD（400 KHz)为时钟源，并按照下列命令顺序发送命令：

– SD 卡 - 发送 CMD0，ACMD41，CMD2，CMD3；

– SDHC 卡 - 发送 CMD0，CMD8，ACMD41，CMD2，CMD3；

– SD I/O 卡 - 如果卡没有存储器端口，发送 CMD52，CMD0，CMD5，CMD3；否则，发送 CMD52，CMD0，CMD5，ACMD41，CMD11（可选)，CMD2，CMD3；

– MMC/CE-ATA - 发送 CMD0，CMD1，CMD2，CMD3。

识别 MMC/CE-ATA 设备。

– CPU 应该通过发送 CMD8 查询 EXT_CSD 寄存器的 504 字节（S_CMD_SET）。如果第 4 位被设置为 1，则该设备支持 ATA 模式；

– 如果支持 ATA 模式, CPU 应通过设置 EXT_CSD 寄存器的 191 字节(CMD_SET)的(第 4 位) ATA 位选择 ATA 模式，以激活使用 ATA 命令集。CPU 使用 SWITCH（CMD6）命令选择命令集；

– 如果 CE-ATA 设备存在，FAST_IO（CMD39）和 RW_MULTIPLE_REGISTER（CMD60）命令将会成功，并且返回的数据将会是 CE-ATA 复位签名。

31.6.2 无数据命令

发送任何无数据命令时，软件需要用适当的参数设置 SDIO_CMD 寄存器和 SDIO_ARG 寄存器。通过这两个寄存器，主机形成命令，并将其发送到命令总线上。主机通过 SDIO_STA 寄存器的错误标志来反映命令响应的错误。

当接收到响应时，主机设置 SDIO_INTSTS 寄存器 RCRC(响应错误)位为 1。短响应被复制到 SDIO_RESP0，而长响应被复制到所有四个响应寄存器。SDIO_RESP3 寄存器的第 31 位代表的长响应的最高位，而 SDIO_RESP0 寄存器的第 0 位表示长响应最低位

31.6.3 写数据命令

单块或多块写操作步骤为：

1. 在 SDIO_DLEN 寄存器中设置数据大小（以字节为单位）。

2. 在 SDIO_BLKSIZE 寄存器中设置数据块大小；主机每次发送 BLKSIZE 大小的数据块。

3. 在 SDIO_CMDARG 寄存器中设置数据应该被写入的地址。

4. 设置 SDIO_CMD 寄存器。对于 SD 存储卡和 MMC 卡，使用 CMD24 命令为单块写和 CMD25 命令为多块写。对于 SD I/O 卡，使用 CMD53 命令来进行单块和多块传输。对于 CEATA，先用 CMD60

写 ATA 任务文件，然后使用 CMD61 命令写入数据。在写 CMD 寄存器之后，主机开始执行一个命令，当该命令被发送到总线时，CD(命令完成）标志被设置。

5. 将数据写入 SDIO_FIFO

6. 软件应查询数据错误中断。如果需要，软件可以通过发送停止命令（CMD12）终止数据传输。

7. 当收到 DLEN 中断时，数据传送结束。对于开放式的块传输，如果字节计数为 0，则软件必须发送 STOP 命令。如果字节计数不为 0，则在给定的字节数传送结束时，主机应该发送停止命令

31.6.4 单个数据块或多个数据块读

单块或多块读操作步骤为：

1. 在 SDIO_DLEN 寄存器中设置数据大小（以字节为单位）。

2. 在 SDIO_BLKSIZE 寄存器中设置数据块大小；主机每次发送 BLKSIZE 大小的数据块。

3. 在 SDIO_CMDARG 寄存器中设置需要读取数据的开始地址。

4. 设置 SDIO_CMD 寄存器。对于 SD 和 MMC 卡，使用 CMD17 用于单块读取和 CMD18 为多块读取。对于 SD I/O 卡，使用 CMD53 用于单块和多块传输。对于 CE-ATA，先用 CMD60 写 ATA 任务文件，然后使用 CMD61 来读取数据。设置 CMD 寄存器之后，主机开始执行该命令，当该命令被发送到总线时，CD 标志被设置.

5. 软件应查询数据错误中断。如果需要，软件可以通过发送停止命令（CMD12）终止数据传输。

6. 软件应从 FIFO 中读数据，并腾出 FIFO 的空间用于接收更多的数据。

7. 当收到 DTEND 中断时，软件应读出 FIFO 中剩余的数据。

31.7 寄存器描述

31.7.1 SDIO 电源控制寄存器（SDIO_POWER）

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PWRCTRL
-															RW

Bit	Name	R/W	Reset Value	Function
31：1	保留	-	-	-
0	PWRCTRL	RW	0	PWRCTRL:电源控制位 用于定义卡时钟的当前功能状态： 0：电源关闭，卡的时钟停止。 1：保留的上电状态。

注：写数据后的 7 个 HCLK 时钟周期内，不能写入这个寄存器。

31.7.2 SDIO 时钟控制寄存器（SDIO_CLKCR）

地址偏移：0x04

复位值：0x0000 7000

SDIO_CLKCR 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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CKSEL	SMPEN	SMPCLKSEL	WIDBUS[1:0]		PWRSAV	CLKEN	CLKDIV[7:0]							
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	保留	-	-	-
14	CKSEL	RW	1	CMD & DAT 输出时钟选择: 0：SD_CLK 时钟偏移 90 度输出; 1：SD_CLK 时钟偏移 180 度输出;
13	SMPEN	RW	1	预采样使能: 0：不使用预采样时钟; 1：使用预采样时钟;
12	SMPCLKSEL	RW	1	CMD & DAT 预采样时钟选择（都是上升沿）： 1：SD_CLK 时钟; 0：SD_CLK 时钟偏移 270 度;
11：10	WIDBUS	RW	2’h0	WIDBUS：宽总线模式使能位 00：默认总线模式，使用 SDIO_D[0]。 01：4 位总线模式，使用 SDIO_D[3:0]。 10/11: 8 位总线模式, 使用 SDIO_D[7:0]。
9	PWRSAV	RW	0	PWRSAV：省电配置位 为了省电，当总线为空闲时，设置 PWRSAV 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。
8	CLKEN	RW	0	CLKEN：时钟使能位 0：SDIO_CK 关闭。 1：SDIO_CK 使能。
7:0	CLKDIV	RW	8’h0	CLKDIV：时钟分频系数,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/[CLKDIV*2]

注意：

1 当 SD/SDIO 卡或 MMC 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_CLKCR 寄存器不控制 SDIO_CK。

31.7.3 SDIO 参数寄存器（SDIO_ARG）

地址偏移：0x08

复位值：0x0000 0000

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
CMDARG															
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**
CMDARG															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	CMDARG	RW	32’h0	CMDARG：命令参数 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。

31.7.4 SDIO 命令寄存器（SDIO_CMD）

地址偏移：0x0C

复位值：0x2000 0000

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
START CMD	Res			BOOTM ODE	BOOT DIS	BOOT ACK	BOOTEN	IEN	ATAC MD	REGS YNC	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**
AUTIIN IT	ABORT CMD	WAITP END	AUTOS TOP	DTMO ODE	DIR	DEXP ECT	CHECKRE SPCRC	RESP LEN	WAITR ESP	CMDINDEX[5:0]					
RW															

Bit	Name	R/W	Reset Value	Function
31	STARTCMD	RW	0	开始命令。 一旦命令被 CIU 接收，位就会被清除。 当该位被设置后，主机不能够写入命令寄存器。如果试图写，硬件锁错误将在中断状态寄存器中设置。 一旦发送命令,并从 SD_MMC_CEATA 卡收到响应，命令完成位在中断状态寄存器中设置。
30:28	保留	-	-	-
27	BOOTMODE	RW	0	启动模式 0 ： 强制启动操作 1 ： 备用启动操作

26	BOOTDIS	RW	0	Disable_boot 禁用启动。“BOOTDIS”和“BOOTEN”不能同时设置。
25	BOOTACK	RW	0	预期启动响应。 当该位和 BOOTEN 一起设置时，会给 CIU 一个预期响应。
24	BOOTEN	RW	0	启用启动 该位仅应设置为强制启动模式。 当软件一起设置该位和 start_cmd，CIU 通过判断对应卡的 boot 线是否拉低来启动。“BOOTDIS”和“BOOTEN”不能同时设置。
23	IEN	RW	0	中断使能 0：CE-ATA 设备中没有启用中断 1：CE-ATA 设备上启用中断(nIEN = 0)，
22	ATACMD	RW	0	ATACMD 0：主机对 CE-ATA 设备没有执行读访问 1：主机对 CE-ATA 设备执行读访问。
21	REGSYNC	RW	0	0 ： 正常的命令序列 1 ： 不发送命令，只是更新时钟寄存器值到卡时钟域。改变时钟频率或停止时钟，而无需发送命令到卡
20:16	保留	-	0	-
15	AUTOINIT	RW	0	0 ： 在发送此命令之前，不发送初始化序列(80 个时钟) 1 ： 在发送此命令之前发送初始化序列
14	ABORTCMD	RW	0	0 ： 不结束当前正在进行数据传输的命令。 1 ： 结束当前正在进行数据传输的命令。
13	WAITPEND	RW	0	如果该位置位，命令状态机开始发送命令前需要等待数据传输结束。 0：无影响 1：等待数据传输结束
12	AUTOSTOP	RW	0	0 ： 在数据传输结束时不发送停止命令 1 ： 在数据传输结束时发送停止命令
11	DTMODE	RW	0	0 ： 数据块传输命令 1 ： 数据流传输命令
10	DIR	RW	0	0：读卡 1：写卡
9	DEXPECT	RW	0	0 ： 没有预期数据传输(读/写) 1 ： 有预期数据传输(读/写)
8	CHECKRESPCRC	RW	0	0 ： 不检查响应 CRC 1 ： 检查响应 CRC 一些命令响应不返回有效的 CRC 位。 软件应该禁用这些命令的 CRC 检查，以禁用控制器的 CRC 检查。
7	RESPLEN	RW	0	0:短响应 1:长响应
6	WAITRESP	RW	0	0：不等待响应 1：等待响应

5:0	CMDINDEX	RW	6'h0	命令索引

注：1：写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

2：MMC 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。CE-ATA 设备只发送短响应。

31.7.5 SDIO 命令响应寄存器（SDIO_RESPCMD）

地址偏移：0x10

复位值：0x0000 0000

SDIO_RESPCMD 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 RESPCMD 域的内容未知。

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

Bit	Name	R/W	Reset Value	Function
31：6	保留	-	-	-
5:0	RESPCMD	R	6'h0	RESPCMD：响应的命令索引 最后收到的命令响应中的命令索引。

31.7.6 SDIO 命令响应寄存器（SDIO_RESPX）

地址偏移：0x14+4*(x-1),其中 x = 1..4

复位值：0x0000 0000

SDIO_RESP1/2/3/4 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
CARDSTATUSx[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**
CARDSTATUSx[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31：0	CARDSTATUSx	R	32'h0	CARDSTATUSx：见下表

根据响应状态，卡的状态长度是 32 位或 127 位。

表 31-28 响应类型和 SDIO_RESPx 寄存器

寄存器	短响应	长响应
SDIO_RESP1	卡状态[31:0]	卡状态[127:96]
SDIO_RESP2	不用	卡状态[95:64]
SDIO_RESP3	不用	卡状态[63:32]
SDIO_RESP4	不用	卡状态[31:1]

总是先收到卡状态的最高位，SDIO_RESP3 寄存器的最低位始终为 0。

31.7.7 SDIO 数据定时器寄存器（SDIO_TMOUT）

地址偏移：0x24

复位值：0xFFFFFF40

寄存器包含以卡总线时钟周期为单位的数据超时时间。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[23:8]															
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**
DATATIME[7:0]								RESPTIME[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

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。

Bit	Name	R/W	Reset Value	Function
31：8	DATATIME[23:0]	RW	24'hFFFFFF	DATATIME：数据超时时间 以卡总线时钟周期为单位的数据超时时间。
7:0	RESPTIME[7:0]	RW	8'h40	RESPTIME：命令超时时间 以卡总线时钟周期为单位的命令超时时间。

注：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。

31.7.8 SDIO 数据块长度寄存器（SDIO_BLKSIZE）

地址偏移：0x28

复位值：0x0000 0200

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

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15:0	DBLOCKSIZE	RW	16'h200	DBLOCKSIZE：数据块长度 当选择了块数据传输模式，该域定义数据块长度

31.7.9 SDIO 数据长度寄存器（SDIO_DLEN）

地址偏移：0x2C

复位值：0x0000 0200

SDIO_DLEN 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
DATALENGTH[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**
DATALENGTH[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	DATALENGTH	RW	32'h200	数据长度

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(SDIO_BLKSIZ)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。

31.7.10 SDIO 控制寄存器（SDIO_CTRL）

地址偏移：0x30

复位值：0x01000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							ODPUEN	Res							
-							RW	-							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CEATAINTEN	AUTOSTOPCCSD	CCSDEN	ABORTRD	AUTOIRQRESP	READWAIT	DMAEN	INTEN	Res.		FIFORST	SDIORST
-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	-		RW	RW

Bit	Name	R/W	Reset Value	Function
31：25	保留	-	-	-
24	ODPUEN	RW	1	CMD 线上拉: 0：不上拉 1：上拉
23：12	保留	-	-	-
11	CEATAINTEN	RW	0	0：在 CE-ATA 设备中，中断没有使能; 1：在 CE-ATA 设备中，中断使能;
10	AUTOSTOPCCSD	RW	0	0: 在向 CE-ATA 发送 CCSD 后，不发送 STOP; 1: 在向 CE-ATA 发送 CCSD 后,自动发送 STOP; 发送结束后，该 bit 自动清零;
9	CCSDEN	RW	0	0：不向 CE-ATA 发送 CCSD; 1：向 CE-ATA 发送 CCSD;
8	ABORTRD	RW	0	0 ： 没有变化 1 ：在读取数据期间发出挂起命令，重置数据状态机，等待下一个数据块。
7	AUTOIRQRESP	RW	0	0 :没有变化 1 :发送 IRQ 响应 一旦响应发送，该位自动清除。
6	READWAIT	RW	0	0 ： 无读等待 1 ： 有读等待 用于向 SDIO 卡发送读等待。
5	DMAEN	RW	0	0 ： 禁用 DMA 传输模式 1 ： 启用 DMA 传输模式

4	INTEN	RW	0	全局中断启用/禁用位: 0：禁用中断 1：启用中断 只有当该位为 1 且有一个或多个未屏蔽中断位设置时，全局中断端口才为 1
3：2	保留	-	-	-
1	FIFORST	RW	0	0 ： 没有变化 1 ： 复位 FIFO 要复位 FIFO，应该将该位设置为 1。在完成复位操作之后会自动清除
0	SDIORST	RW	0	0 :没有变化 1:重置控制器 要复位控制器，应该设置位为 1。这个位是自动清除的。

31.7.11 SDIO 状态寄存器（SDIO_STA）

地址偏移：0x34

复位值：0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		FIFOCNT[12:0]													Res
-		RW													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						CARDBSY	CARDPRESENT	CMDFSM[3:0]				FIFOF	FIFOE	TXWMARK	RXWMARK
-						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29:17	FIFOCNT	R	13’h0	FIFO 计数
16:10	保留	-	-	-
9	CARDBSY	R	0	卡状态 0：空闲 1：忙
8	CARDPRESENT	R	0	检查卡片是否存在 0 ： 卡片不存在 1 :卡片存在

7:4	CMDFSM	R	4'h0	命令 FSM 状态: 0 :空闲 1 ： 发送 init 序列 2 ： Tx cmd 起始位 3 ： Tx cmd Tx 位 4 ： Tx cmd 索引+参数 5 ： Tx cmd crc7 6 ： Tx cmd 结束位 7 ： Rx 响应起始位 8 ： Rx 响应 IRQ 响应 9 ： Rx 响应 tx 位 10 ： Rx 响应 cmd index 11 ： Rx 响应数据 12：Rx 对应 crc7 13 ： Rx 响应结束位 14 ： Cmd 路径等待 NCC 15 ： 等待
3	FIFOF	R	0	FIFO 满
2	FIFOE	R	1	FIFO 空
1	TXWMARK	R	1	TXFIFO 达到输出水位线
0	RXWMARK	R	0	RXFIFO 达到接收水位线

31.7.12 SDIO 中断状态寄存器（SDIO_INTSTS）

地址偏移：0x38

复位值：0x0000 0000

在对应位置'1'，SDIO_INTSTS 的该位清零.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															SDIOINT
-															RC_W1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EBE	ACD	SBE	HLE	FRUN	HTO	DRTO_BDS	RTO_BAR	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	CAD
RC_W1															

Bit	Name	R/W	Reset Value	Function
31:17	保留	--	-	-
16	SDIOINT	RC_W1	0	0 ： 没有来自卡的 SDIO 中断 1 ： 有来自卡的 SDIO 中断
15	EBE	RC_W1	0	结束位错误/无 CRC 错误（EBE）
14	ACD	RC_W1	0	自动命令完成(ACD)
13	SBE	RC_W1	0	起始位错误(SBE)
12	HLE	RC_W1	0	硬件锁定写错误(HLE)
11	FRUN	RC_W1	0	FIFO 溢出错误(FRUN)
10	HTO	RC_W1	0	主机超时错误(HTO)

9	DRTO_BDS	RC_W1	0	数据读超时(DRTO_BDS)
8	RTO_BAR	RC_W1	0	响应超时(RTO_BAR)
7	DCRC	RC_W1	0	数据 CRC 错误(DCRC)
6	RCRC	RC_W1	0	响应 CRC 错误(RCRC)
5	RXDR	RC_W1	0	接收 FIFO 数据请求(RXDR)
4	TXDR	RC_W1	0	发送 FIFO 数据请求(TXDR)
3	DTO	RC_W1	0	数据传输(DTO)
2	CD	RC_W1	0	命令完成(CD)
1	RE	RC_W1	0	响应错误(RE)
0	CAD	RC_W1	0	卡检测(CAD)

31.7.13 SDIO 中断屏蔽寄存器（SDIO_INTMASK）

地址偏移：0x3C

复位值：0x0000 0000

在对应位置‘1’，中断屏蔽寄存器决定哪一个状态位中断能够产生。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SDIOINTIE
-															RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EBEIE	ACDIE	SBEIE	HLEIE	FRUNIE	HTOIE	DRTOIE	RTOIE	DCRCIE	RCRCIE	RXDRIE	TXDRIE	DTOIE	CDIE	REIE	CADIE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	SDIOINTIE	RW	0	SDIO 中断使能 0：不使能 1：使能
15	EBEIE	RW	0	结束位错误/无 CRC 错误（EBE）中断使能 0：不使能 1：使能
14	ACDIE	RW	0	自动命令完成(ACD）中断使能 0：不使能 1：使能
13	SBEIE	RW	0	起始位错误(SBE）中断使能 0：不使能 1：使能
12	HLEIE	RW	0	硬件锁定写错误(HLE）中断使能 0：不使能 1：使能
11	FRUNIE	RW	0	FIFO 溢出错误(FRUN）中断使能 0：不使能 1：使能
10	HTOIE	RW	0	主机超时错误(HTO）中断使能

				0：不使能 1：使能
9	DRTO_BDSIE	RW	0	数据读超时(DRTO_BDS）中断使能 0：不使能 1：使能
8	RTO_BARIE	RW	0	响应超时(RTO_BAR）中断使能 0：不使能 1：使能
7	DCRCIE	RW	0	数据 CRC 错误(DCRC）中断使能 0：不使能 1：使能
6	RCRCIE	RW	0	响应 CRC 错误(RCRC）中断使能 0：不使能 1：使能
5	RXDRIE	RW	0	接收 FIFO 数据请求(RXDR）中断使能 0：不使能 1：使能
4	TXDRIE	RW	0	发送 FIFO 数据请求(TXDR）中断使能 0：不使能 1：使能
3	DTOIE	RW	0	数据传输(DTO）中断使能 0：不使能 1：使能
2	CDIE	RW	0	命令完成(CD）中断使能 0：不使能 1：使能
1	REIE	RW	0	响应错误(RE）中断使能 0：不使能 1：使能
0	CADIE	RW	0	卡检测(CAD）中断使能 0：不使能 1：使能

31.7.14 SDIO FIFO 阈值寄存器（SDIO_FIFOTH)

地址偏移：0x40

复位值：0x000F 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				RXWMARK[11:0]											
-				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
Res				TXWMARK[11:0]											
-				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-

27:16	RXWMARK[11:0]	RW	12'hf	接收 FIFO 的水位线设定。 当 FIFO 数据计数超过这个数时，产生读数据请求。当数据包接收结束时，不管水位线设定，都会产生请求，以完成剩余的数据读取。
15:12	保留	-	-	-
11:0	TXWMARK[11:0]	RW	12'hf	发送 FIFO 的水位线设定。 当 FIFO 数据计数小于或等于这个数字时，发出写数据请求。

31.7.15 SDIO 发送到卡计数器（SDIO_TCBCNT）

地址偏移：0x44

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCBCNT[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**
TCBCNT[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	TCBCNT[31:0]	R	32'h0	传输到卡上的字节数。

31.7.16 SDIO 发送到 FIFO 计数器(SDIO_TBBCNT)

地址偏移：0x48

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBBCNT[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**
TBBCNT[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	TCBCNT[31:0]	R	0	接收或者发送 FIFO 的字节数。

31.7.17 SDIO 数据 FIFO 寄存器（SDIO_FIFODATA）

地址偏移：0x200

复位值：0x0000 0000

接收和发送 FIFO 是 32 位宽读或写的一组寄存器，它在连续的 32 个地址上包含 32 个寄存器。

注：数据 fifo 寄存器包括发送和接收 fifo，因此要将这 32 个地址分为 16 个一组，发送接收各占一半。在每次读写的时候，最多读取接收 FIFO 或者写入发送 FIFO 的大小的数据。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[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
FIFODATA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	FIFODATA[31:0]	RW	32’h0	FIFODATA：接收或发送 FIFO 数据 FIFO 不支持从一个端口进行同时读写。

32. USB OTG 全速（OTG_FS）

32.1 概述

OTG_FS 是一款双角色设备（DRD）控制器，同时支持设备功能和主机功能，完全符合 USB 2.0 规范的 On-The-Go 补充标准。此外，该控制器也可配置为“仅主机”模式或“仅设备”模式，完全符合 USB 2.0 规范。在主机模式下，OTG_FS 支持全速（FS，12 Mb/s）和低速（LS，1.5 Mb/s）收发器，而设备模式下则仅支持全速（ FS，12 Mb/s）收发器。OTG_FS 同时支持 HNP(主机协商协议)和 SRP(会话请求协议)。主机模式下需要的唯一外部设备是提供 VBUS 的电荷泵。

32.1.1 主要特性

以下将从通用功能、主机模式功能和设备模式功能三方面来介绍 OTG_FS 控制器的特性。

32.1.1.1 通用功能

- 由 USB-IF 认证，符合 Universal Serial Bus Specification，Revision2.0 标准
- 完全支持在(OTG_FS 控制器的物理层，即 PHY)USB On-The-Go Supplement，Revision1.3 规范中定义的 OTG 协议。
  - 对插入的 A - B 类设备的辨认(ID 线)
  - 支持主机协商协议(HNP)和会话请求协议(SRP)
  - 在 OTG 应用中，允许主机关闭 VBUS 以节省耗电
  - OTG 控制器使用内部比较器监视 VBUS 电平
  - 可以动态的切换主机/设备角色
- 可以通过软件配置为以下角色：
  - 支持 SRP 协议的 USB 全速设备(B 类设备)
  - 支持 SRP 协议的 USB 全速/低速主机(A 类设备)
  - USB OTG 全速双角色设备
- 支持全速通信的 SOF 信号和低速通信的保持有效信号
  - SOF 的脉冲可以输出到引脚
  - SOF 在内部连接到定时器 2(TIM2)
  - 可配置的帧周期
  - 可配置的帧结束中断
- 具有省电功能，例如在 USB 挂起期间停止系统、关闭数字模块时钟、对 PHY 和 DFIFO 电源加以管理。
- 具有采用高级 FIFO 控制的 1.25 KB 专用 RAM
  - 通过软件为不同的 FIFO 配置不同的 RAM 区域，以便灵活有效的使用 RAM
  - 每个 FIFO 可以存储多个数据包
  - 允许动态的分配存储区
  - 不限定 FIFO 的长度一定是 2 的幂次方值，以便灵活有效地使用 RAM
- 不需要系统的介入就可以保证一个帧(1 ms)的最大数据流量

32.1.1.2 主机模式功能

OTG_FS 接口在主机模式下具有以下主要特性和要求：

- 需要一个外置的电荷泵为 VBUS 供电
- 支持最多 8 个主机通道，每个通道都可以动态的配置为任意一种传输类型
- 内置硬件调度控制器：
  - 在周期性硬件传输请求队列中支持多达 8 个中断和同步传输请求
  - 在非周期性硬件传输请求队列中支持多达 8 个控制和批量传输请求。
- 为有效地使用 RAM 空间，USB 的数据 RAM 区划分为一个共享的接收 FIFO、一个周期性发送 FIFO 和一个非周期性发送 FIFO。

32.1.1.3 设备模式功能

OTG_FS 接口在设备模式下具有以下特性：

- 1 个双向的控制端点 0
- 3 个 IN 端点，支持批量、中断或同步传输
- 3 个 OUT 端点，支持批量、中断或同步传输
- 管理一个共享的接收 FIFO 和一个发送 OUT FIFO，以高效使用 USB 数据 RAM
- 管理多达 4 个专用的发送 IN FIFO(为每个 IN 端点配置一个 FIFO)，以便降低应用程序的负荷
- 支持软件断开功能。

32.1.2 结构框图

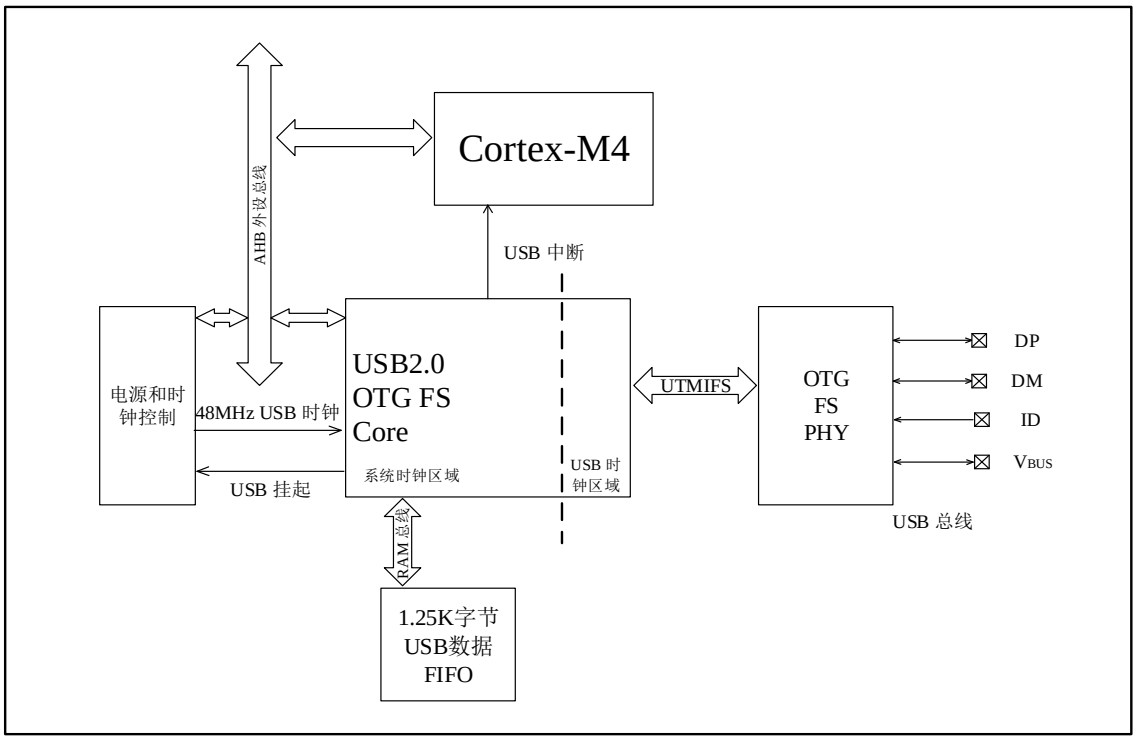


图 32-1 OTG_FS 结构框图

32.2 功能描述

32.2.1 OTG 全速控制器

USB OTG 全速控制器从复位和时钟控制模块(RCC)获得 48MHz±0.25％的时钟，这个时钟用于驱动 USB 全速模块(12Mbit/s)的 48MHz 控制时钟，必须在配置 OTG 全速控制器之前使能这个控制时钟。

微控制器内核(CPU)通过 AHB 外设总线来访问 OTG 全速控制器的寄存器，USB 事件由单独的 USB OTG 中断控制线管理。

微控制器以向 OTG_FS 专用地址(PUSH 寄存器)写 32 位数据的方式向 USB 控制器传输数据，这些数据将自动存入 USB 配置好的发送 FIFO 中。每个发送 FIFO 都配置了这样一个 PUSH 寄存器，为每个 IN 端点(设备模式)或 OUT 通道(主机模式)服务。

微控制器通过读 OTG_FS 的专用地址(POP 寄存器)获得来自 USB 总线的 32 位数据，这些数据将自动从共享的接收 FIFO 中载入。接收 FIFO 位于总共 1.25K 字节的 USB 数据 RAM 区。每一个 OUT 端点或 IN 通道都有这样一个 POP 寄存器为之服务。

USB 协议层由串行接口引擎(SIE)驱动，并连接到由内置物理层(PHY)支持的 USB 全速/低速收发模块。

32.2.2 全速 OTG PHY(物理接口)

内置的全速 OTG PHY 由 OTG 全速控制器控制，并通过 UTMI+总线(UTMIFS)的全速子集来收发控制和数据信号。PHY 为 USB 通信提供了物理层的支持。

全速的 OTG PHY 包括以下部分：

- 在主机和设备模式下使用的全速/低速收发模块。此模块直接在单端的 USB 线上驱动发送和接收。
- 内置了 ID 线的上拉电阻，用于区分 A/B 类设备。
- DP/DM 线内置了上拉和下拉电阻，由 OTG_FS 控制器控制以满足当前设备类型的需求。在设备模式下，当 VBUS 线上出现了有效的电平(B 类有效)，控制器使能 DP 线的上拉电阻，向主机通告接入一个 USB 全速设备。在主机模式下，控制器同时使能 DP 和 DM 线的下拉电阻。上拉和下拉电阻可以在控制器通过主机协商协议(HNP)切换角色类型时动态的切换。
- 上拉/下拉电阻的 ECN 电路。DP 线有两个由 OTG_FS 控制器分开控制的上拉电阻，符合 USB2.0 版本的 ECN 要求(Enginering Change Notice)。支持动态调整 DP 线的上拉强度，能更好的对抗噪声，并提高发送和接收信号质量。
- 内置带迟滞功能的 VBUS 探测比较器，用于检测 VBUS 的有效电平、A-B 会话有效电平和会话结束电平。这些电平用于驱动会话请求协议(SRP)、检测有效的会话开始和结束条件，并在 USB 通信中持续的检测 VBUS 线状态。
- VBUS 脉冲电路用于在 SRP 期间通过电阻对 VBUS 进行充电/放电操作(弱驱动)。

注意：为了确保 USB OTG FS 模块正常工作，AHB 的频率应该大于等于 30 MHz。

32.2.3 OTG 双角色设备(DRD)

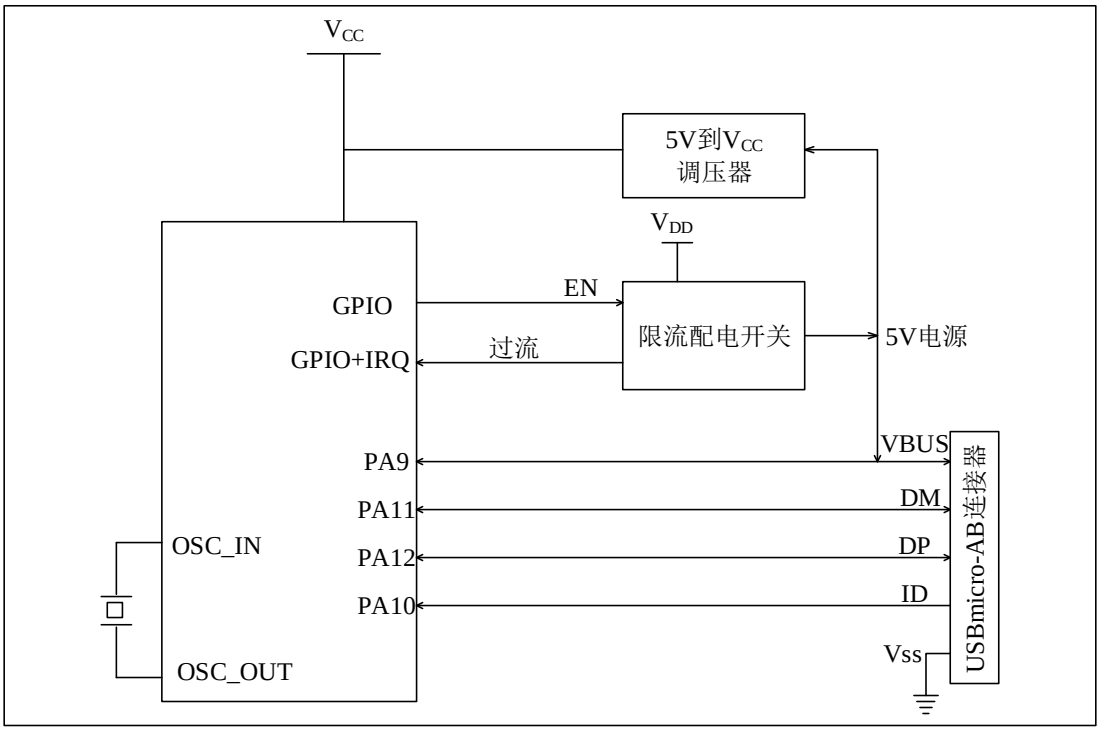


图 32-2 OTG_FS 的 A-B 设备连接

32.2.3.1 ID 信号检测

主机和设备(默认)的角色由 ID 线的状态定义。插入 USB 时即可根据哪一端 USB 电缆连接到 micro-AB 插座来确定 ID 线状态。

- 如果插入的是 USB 电缆的 B 端，并且 ID 线浮空，内置的上拉电阻将检测到 ID 线的高电平，此时控制器处于默认的设备模式下。在这种模式下，OTG_FS 控制器遵守 USB2.0 规范的补充 OTG1.3 规范的第 6.8.2 节：OTG B 类设备对标准 FSM 的描述。
- 如果插入的是 USB 电缆的 A 端，并且 ID 线接地。此时，OTG_FS 控制器将执行一个 ID 线状态改变中断(OTG_FS_GINTSTS 寄存器的 CIDSCHG 位)，自动切换到主机模式，并需要软件初始化主机模式。在这种模式下，OTG_FS 控制器遵守 USB2.0 规范的补充 OTG1.3 规范的的第 6.8.1 节：OTG A 类设备对标准 FSM 的描述。

32.2.3.2 HNP 双角色设备

USB 全局配置寄存器的 HNP 使能位(OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)允许 OTG_FS 控制器在 A 类主机和 A 类设备，以及 B 类主机和 B 类设备之间通过主机协商协议(HNP)动态地切换。当前设备的状态可以通过读 OTG 全局控制和状态寄存器的 ID 状态位(OTG_FS_GOTGCTL 寄存器的 CIDSTS 位)和全局中断和状态寄存器的当前操作模式位(OTG_FS_GINTSTS 寄存器的 CMOD 位)来综合获得。

32.2.3.3 SRP 双角色设备

USB 全局配置寄存器的 SRP 使能位(OTG_FS_GUSBCFG 寄存器的 SRPCAP 位)允许 OTG_FS 控制器关闭 VBUS 供电，为 A 类设备节省电能。注意：A 类设备无论处于主机模式还是设备模式下都需要控制管理 VBUS 的供电。

32.2.4 USB 设备模式

本章介绍 OTG_FS 控制器在 USB 设备模式时的功能。

OTG_FS 控制器在以下情况时工作在设备模式下：

- OTG B 类设备
  - 插入的是 USB 电缆的 B 端时，默认的是 OTG B 类设备模式
- OTG A 类设备
  - OTG A 类设备通过 HNP 协议切换到设备角色
- B 类设备
  - 在 ID 线存在时，插入的是 USB 电缆的 B 端，同时 USB 全局配置寄存器的 HNP 位
- (OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)为 0 时(请参考 OTG 1.3 版的第 6.8.3 节)
- 仅作为 USB 设备
  - USB 全局配置寄存器的强制设备模式位(OTG_FS_GUSBCFG 寄存器的 FDMOD 位)为 1，将强制使 OTG_FS 控制器工作于 USB 设备模式(请参考 OTG 1.3 版的第 6.8.3 节)。在这种情况下 ID 线无论是否存在，其状态都将被忽略。

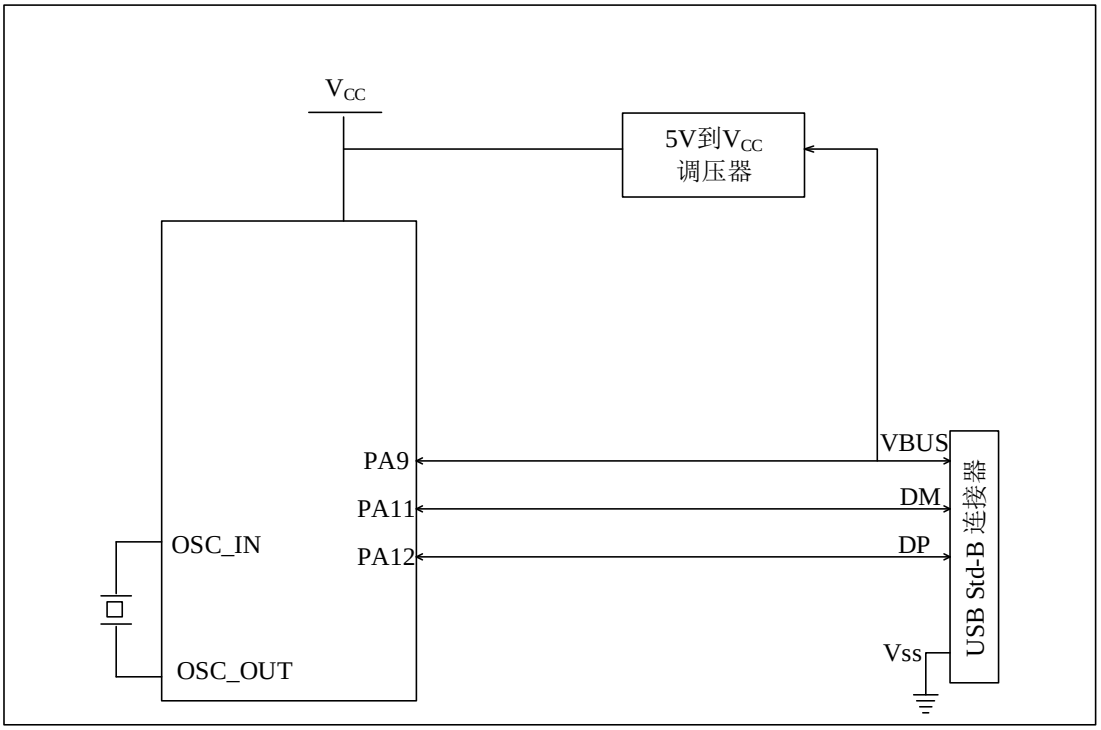


图 32-3 USB 仅作外设的连接

32.2.4.1 支持 SRP 功能的设备

USB 全局配置寄存器的 SRP 使能位(OTG_FS_GUSBCFG 寄存器的 SRPCAP 位)允许 OTG_FS 控制器实现会话请求协议(SRP)。在这种情况下，A 类设备可以在 USB 会话挂起时停止对 VBUS 的供电。

32.2.4.2 设备状态

上电状态

VBUS 线上检测到 B 类设备有效的电平后，USB 设备进入上电状态。OTG_FS 控制器将自动使能 DP 线上的上拉电阻以告知主机接入了一个全速设备，并产生一个会话请求中断(OTG_FS_GINTSTS 寄存器的 SRQINT 位)，标志进入上电状态。

主机在整个 USB 通信阶段都需要提供有效的 VBUS 电平。如果检测到 VBUS 线上的电平低于 B 类有效电平(例如发生了供电扰动，或者主机方关闭了供电)，OTG_FS 控制器将自动断开 USB 连接，并产生会话结束中断(OTG_FS_GOTGINT 寄存器的 SEDET 位)，标志控制器退出上电状态。

上电状态时，OTG_FS 控制器期望从主机接收复位信号，其他 USB 操作都是无效的。当接收到复位信号后，会产生复位中断(OTG_FS_GINTSTS 寄存器的 USBRST 位)。复位完成后，会产生枚举中断(OTG_FS_GINTSTS 寄存器的 ENUMDNE 位)，标志 OTG_FS 控制器进入默认状态。

软件断开

可以使用软件来配置退出上电状态。设置设备控制寄存器的软件断开位(OTG_FS_DCTL 寄存器的 SDIS 位)可以移除 DP 线上的上拉电阻，从而使主机即使在 USB 电缆依然连接时，识别到一个设备断开事件。

默认状态

在默认状态下，OTG_FS 控制器期望接收到 SET_ADDRESS 命令，此时其他操作都是无效的。当接收到一个有效的 SET_ADDRESS 命令后，应用程序需要将相关的地址写入设备配置寄存器的设备地址位(OTG_FS_DCFG 寄存器的 DAD 位)中。OTG_FS 控制器进入地址状态，并准备好响应主机发向这个地址的传输。

挂起状态

OTG_FS 控制器持续的检测 USB 线上的活动。在检测到 3ms 的 USB 空闲状态后，将产生早期挂起中断(OTG_FS_GINTSTS 寄存器的 ESUSP 位)，在 3ms 后产生挂起中断(OTG_FS_GINTSTS 寄存器的 USBSUSP 位)确认挂起。设备状态寄存器的设备挂起位(OTG_FS_DSTS 寄存器的 SUSPSTS 位)将自动置位，OTG_FS 控制器进入挂起状态。

设备可以自发的退出挂起状态，通过置位设备控制寄存器的远程唤醒信号位(OTG_FS_DCTL 寄存器的 WKUPINT 位)，并在 1ms 到 15ms 之间清除该位来实现。

当检测到主机发出恢复信号，将产生恢复中断(OTG_FS_GINTSTS 寄存器的 RWUSIG 位)，同时设备的挂起位被自动清除。

32.2.4.3 设备端点

OTG_FS 控制器支持以下 USB 端点：

- **控制端点 0**
  - 双向端点，仅用于处理控制信息
  - 对于 IN 和 OUT 两个方向，各有一套独立的寄存器来控制
  - 包括控制(OTG_FS_DIEPCTL0/ OTG_FS_DOEPCTL0)寄存器，传输配置(OTG_FS_DIEPTSIZ0/ OTG_FS_DOEPTSIZ0)寄存器，和状态中断(OTG_FS_DIEPINT0/ OTG_FS_DOEPINT0)寄存器。控制和传输长度寄存器中的某些位与其他端点的寄存器不同。
- **3 个 IN 端点**
  - 每个端点都可以配置为同步传输、批量传输或中断传输
  - 每个端点都有控制(OTG_FS_DIEPCTLx)寄存器，传输配置(OTG_FS_DIEPTSIZx)寄存器和状态中断(OTG_FS_DIEPINTx)寄存器
  - 设备 IN 端点通用中断屏蔽寄存器(OTG_FS_DIEPMSK)用于使能/禁止所有 IN 端点(包括端点 0)上任一类型的端点中断源
  - 支持未完成的同步 IN 传输中断(OTG_FS_GINTSTS 寄存器的 IISOIXFR 位)，在当前帧有至少一个同步 IN 传输未完成时，会产生该中断。此中断随同周期帧中断产生(OTG_FS_GINTSTS 寄存器的 EOPF)
- **3 个 OUT 端点**
  - 每个端点都可以配置为同步传输、批量传输或中断传输
  - 每个端点都有控制(OTG_FS_DOEPCTLx)寄存器，传输配置(OTG_FS_DOEPTSIZx)寄存器和状态中断(OTG_FS_DOEPINTx)寄存器
  - 设备 OUT 端点通用中断屏蔽寄存器(OTG_FS_DOEPMSK)用于使能/禁止所有 OUT 端点(包括端点 0)上任一类型的端点中断源
  - 支持未完成的同步 OUT 传输中断(OTG_FS_GINTSTS 寄存器的 INCOMPISOOUT 位)，在当前帧有至少一个同步 OUT 传输未完成时，会产生该中断。此中断随同周期帧中断产生(OTG_FS_GINTSTS 寄存器的 EOPF)

端点配置

- 以下对端点的配置，都通过设置端点 IN/OUT 控制寄存器实现（OTG_FS_DIEPCTLx/ OTG_FS_DOEPCTLx)：
  - 端点使能/禁止
  - 在当前配置中激活端点
  - 配置 USB 传输类型(同步，批量，中断)
  - 配置 USB 数据包长度
  - 配置与 IN 端点相关的发送 FIFO 编号
  - 配置期望收到的或将发送的 PID 包 DATA0/DATA1(仅对批量传输和中断传输有效)
  - 配置发送或接收事务时所对应的奇/偶帧(仅对同步传输有效)
  - 可选的配置：设置 NAK 位，无论 FIFO 的状态如何都回应主机 NAK
  - 可选的配置：设置 STALL 位，对主机发送的任何命令都回应 STALL
  - 可选的配置：设置 OUT 端点为侦听(SNOOP)模式，对于接收到的数据不检验 CRC

端点传输

设备端点传输长度寄存器(OTG_FS_DIEPTSIZx/ OTG_FS_DOEPTSIZx)用于配置传输长度和读取传输状态。对该寄存器的配置必需在设置端口控制寄存器的使能位之前。一旦端点被使能，该寄存器便处于只读状态，只有 OTG_FS 控制器可以更新该寄存器的传输状态位。

- 需要配置的传输参数如下：
  - 以字节为单位的单次传输的长度
  - 完成整个传输的 USB 数据包的数目

端点状态/中断

设备端点中断寄存器(OTG_FS_DIEPINTx/ OTG_FS_DOEPINTx)指示了端点与 USB 和 AHB 相关的事件状态。当设置了控制器中断寄存器的 OUT 端点中断位或者 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 位和 IEPINT 位)时，应用程序需要先读取设备所有端点中断寄存器(OTG_FS_DAINT)，确定发生中断的端点号，然后读取设备端点中断寄存器，获得中断的详细信息。应用程序必需随即清除寄存器的相应中断位，使之清除 OTG_FS_DAINT 寄存器和 OTG_FS_GINTSTS 寄存器的相应中断位。

- 设备控制器提供以下的状态位和中断事件：
  - 传输完成中断，指示了在 AHB 和 USB 方面的数据传输都完成了。
  - SETUP 阶段完成(仅对控制端点有效)
  - 相关的传输 FIFO 已经半空或者全空(IN 端点)
  - 已发送 NAK 响应至主机(仅对同步 IN 传输有效)
  - 当发送 FIFO 为空时，主机发出了 IN 请求(仅对批量 IN 传输和中断 IN 传输有效)
  - 在端点没有使能时，主机发出了 OUT 请求
  - 检测到串扰错误
  - 应用程序关闭了端点
  - 应用程序将端点状态设为 NAK(仅对同步 IN 传输有效)
  - 收到超过 3 个连续的 SETUP 包(仅对控制 OUT 传输有效)
  - 检测到超时错误(仅对控制 IN 传输有效)
  - 同步传输类型的数据包未产生中断而丢失

32.2.5 USB 主机

本章介绍了 OTG_FS 控制器工作在 USB 主机模式时的功能。OTG_FS 控制器在以下条件时工作在主机模式：

- OTG A 类主机
  - 插入的是 USB 电缆的 A 端，默认的是 OTG A 类主机模式
- OTG B 类主机
  - OTG B 类设备通过 HNP 协议切换到主机角色
- A 类设备
  - 在 ID 线存在时，插入的是 USB 电缆的 A 端，同时 USB 全局配置寄存器的 HNP 位为 0
- (OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)。此时，内置的 DP/DM 线上的上拉电阻自动有效。
- 仅作为主机

— USB 全局配置寄存器的强制主机模式位(OTG_FS_GUSBCFG 寄存器的 FHMOD 位)能强制 OTG_FS 控制器工作的 USB 主机模式。此时即使 USB 连接器上的 ID 线存在，也依然无效。内置的 DP/DM 线上的上拉电阻自动有效。

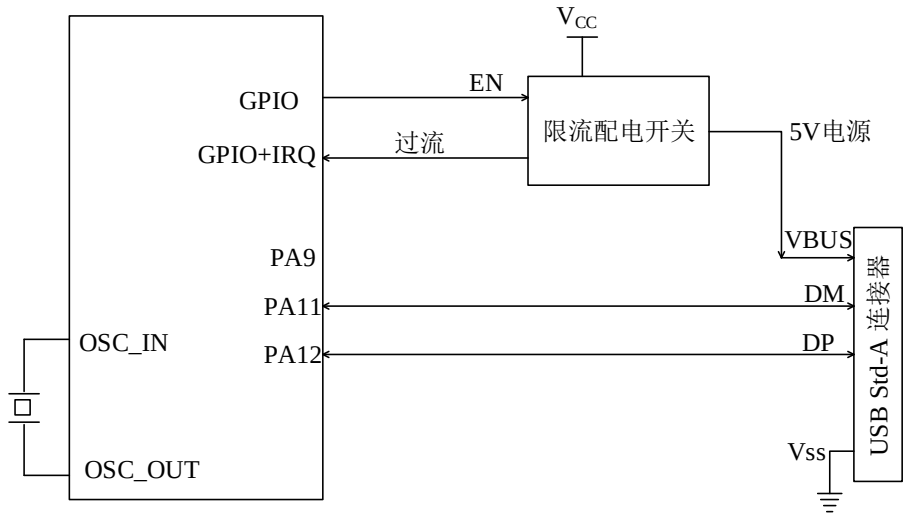


图 32-4 USB 仅作主机的连接

32.2.5.1 具备 SRP 功能的主机

通过配置 USB 全局配置寄存器的 SRP 使能位(OTG_FS_GUSBCFG 寄存器的 SRPCAP 位)，能够支持 SRP 功能。使能了 SRP 功能后，主机可以在 USB 会话挂起时，关闭 VBUS 的供电以节省耗电。

32.2.5.2 USB 主机状态

主机端口供电

控制器内部不提供 VBUS 的 5 V 供电，因此需要使用外接的电荷泵来给 VBUS 线供电，如果应用板上能提供 5 V 供电，也可以使用普通的电源开关。这个外接的电荷泵可以通过普通的 I/O 口来控制。当应用程序需要设置这个普通 I/O 口来给 VBUS 线供电时，也需要设置主机端口控制和状态寄存器的供电位(OTG_FS_HPRT 寄存器的 PPWR 位)。

VBUS 使能

在 USB 整个通信过程中，VBUS 线需要一直保证为有效的电平。任何导致 VBUS 线的电平下降到门限(4.25 V)以下的事件都将导致由会话中止位触发的 OTG 中断(OTG_FS_GOTGINT 寄存器的 SEDET 位)。此时，应用程序需要关闭对 VBUS 的供电，并清除端口供电位。电荷泵的过流信号也能用来防止电气损害，将此过流信号连接到设置为中断输入的 I/O 口上，一旦产生了过流事件，应用程序需要在中断中立即关闭 VBUS 的供电，并清除端口供电位。

主机对于设备接入的监测

如果 VBUS 线始终没有有效的电平(4.75 V)，无论何时插入 USB 设备或者 B 类设备，OTG_FS 控制器都不能监测到设备的插入。

当 VBUS 线处于有效电平范围内，一旦有 B 类设备插入，OTG_FS 控制器就会产生由设备插入位(OTG_FS_HPRT 寄存器的 PCDET 位)触发的主机端口中断。

主机对于设备断开的监测

断开插入的设备，将产生由设备断开事件触发的断开中断(OTG_FS_GINTSTS 寄存器的 DISCINT 位)。

主机枚举

一旦主机检测到有设备插入，需要进入枚举过程，就会向新插入的设备发送 USB 复位和配置命令。由于插入的设备在 DP(全速设备)或者 DM(低速设备)线上有上拉电阻，因此会导致总线的不稳定。当总线再次回到稳定状态时，会触发反跳结束的 OTG 中断(OTG_FS_GOTGINT 寄存器的 DBCDNE 位)，此时主机才能向设备发送 USB 复位命令。

应用程序通过置位主机端口控制和状态寄存器的端口复位位(OTG_FS_HPRT 寄存器的 PRST 位)向 USB 总线发出 USB 复位信号(单端 0)。应用程序需要保证该复位信号维持在 10ms 到 20ms 之间，并及时清除端口复位位。

一旦 USB 复位序列完成，会产生由端口使能/禁止改变位触发的主机端口中断(OTG_FS_HPRT 寄存器的 PENCHNG 位)。此中断通知应用程序可以从主机端口控制和状态寄存器获得被枚举的设备的速度信息(OTG_FS_HPRT 寄存器的 PSPD 位)，随之，主机将发送 SOF(全速设备)信号或保持激活(低速状态)。此时，主机可以向设备发送配置命令，完成设备枚举。

主机挂起

应用程序可以通过置位主机控制和状态寄存器的端口挂起位(OTG_FS_HPRT 寄存器的 PSUSP 位)来挂起 USB 活动。此时，OTG_FS 控制器将停止发送 SOF 信号，并进入挂起状态。

控制器可选择支持由远程设备来唤醒主机退出挂起状态。此时，控制器在检测到远程唤醒信号后，将产生远程唤醒中断(OTG_FS_GINTSTS 寄存器的 WKUPINT 位)，随之，主机端口控制和状态寄存器的唤醒位(OTG_FS_HPRT 寄存器的 PRES 位)将自动置位，控制器将自动向 USB 总线发送唤醒信号。应用程序需要控制唤醒信号的维持时间，并及时地清除端口唤醒位来退出挂起状态并重新开始发送 SOF 信号。

如果需要由主机自发退出挂起状态，应用程序可以设置端口唤醒位，此时唤醒信号将发送到 USB 总线，应用程序需要控制此唤醒信号维持的时间，并及时清除端口唤醒位。

32.2.5.3 主机通道

OTG_FS 控制器提供 8 个主机通道，每个通道对应一个 USB 主机传输(USB PIPE)。主机不支持同时发起超过 8 个的传输请求。如果应用程序未处理的传输请求超过了 8 个，主机控制器驱动器(HCD)必需在通道重新有效后，也就是收到传输完成和通道中止的中断后，为其重新安排通道。

每个通道都可以配置为输入/输出模式，也可以配置为任意的周期/非周期性传输类型。每个通道都有控制寄存器(OTG_FS_HCCHARx)，传输配置寄存器(OTG_FS_HCTSIZx)和状态/中断寄存器(OTG_FS_HCINTx)以及相应的屏蔽寄存器(OTG_FS_HCINTMSKx)。

主机通道控制

- 通道控制寄存器(OTG_FS_HCCHARx)可以提供以下操作：
  - 通道使能/禁止
  - 为连接上的 USB 设备配置传输速度：全速/低速
  - 为连接上的 USB 设备配置地址
  - 为连接上的 USB 设备配置端点号
  - 配置传输方向：输入/输出
  - 配置传输类型：控制，批量，中断，同步
  - 配置最大包长度(MPS)
  - 配置要进行周期性传输的奇数/偶数帧

主机通道传输

应用程序通过主机通道传输长度寄存器(OTG_FS_HCTSIZx)配置传输长度，并读回传输状态。必需在使能一个传输通道前，配置传输长度。一旦通道被使能，传输长度寄存器即变为只读，只有 OTG_FS 控制器可以更具当前传输状态更新该寄存器。

- 可以对以下传输参数进行配置：
  - 以字节为单位的单个包的传输长度
  - 整个传输过程所有的包个数
  - 起始的数据 PID

主机通道状态/中断

主机通道 x 中断寄存器(OTG_FS_HCINTx)指示了通道与 USB 和 AHB 相关的事件。当主机控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)被置位，应用程序需要先读主机所有通道中断寄存器(OTG_FS_HCAINT)，获得产生主机通道中断的通道号，然后根据通道号读主机通道 x 中断寄存器(OTG_FS_HCINTx)，获得中断的信息。应用程序需要清除主机通道 x 中断寄存器(OTG_FS_HCINTx)的相应位，以便清除 OTG_FS_HAINT 和 OTG_FS_GINTSTS 寄存器的相应位。可以通过 OTG_FS_HCINTMSKx 寄存器来屏蔽每个通道的中断源。

- 主机控制器提供以下的状态查询和中断：
  - 传输完成中断，指示应用程序端(AHB)和 USB 端的数据传输都已经完成。
  - 由于传输完成、USB 通信错误或者应用程序的禁止命令导致的通道中止

- 相应的传输 FIFO 已经半空或者全空(仅对 IN 通道有效)
- 收到 ACK 响应
- 收到 NAK 响应
- 收到 STALL 响应
- 由 CRC 错误、超时、位填充错误或者错误的 EOP 导致的 USB 传输错误
- 串扰错误
- 帧溢出
- 数据 PID(DATA0/DATA1)翻转错误

32.2.5.4 主机调度器

主机控制器内置一个硬件的调度器，用于自发的管理和驱动应用程序发起的传输请求。在每个帧开始时，主机将先处理周期性(同步和中断传输)的数据传输，再处理非周期性(控制和批量传输)的数据传输，以便符合 USB 规范对同步和中断传输的高优先级保障。

主机通过请求队列(一个周期性队列和一个非周期性队列)来管理 USB 传输通道。每个请求队列都能保存 8 个请求，而每个请求都代表应用程序提出的一个未处理的传输。每个请求队列中的请求都带有 IN/OUT 方向，通道号以及其他的执行一个 USB 传输的信息。USB 事务请求在队列中的写入顺序决定了 USB 总线上的执行顺序。在每个帧首，主机将先处理周期性请求队列，再处理非周期性请求队列。如果在一个帧结束的时候，仍然有未处理的同步或者中断传输请求，将产生未完成周期性传输中断(OTG_FS_GINTSTS 寄存器的 IPXFR 位)。

周期性队列和非周期性队列的管理完全由 OTG_FS 控制器完成。应用程序可以通过只读的寄存器来获得每个请求队列的状态信息。

- 周期性传输 FIFO 和队列状态寄存器(OTG_FS_HPTXSTS)与非周期性传输 FIFO 和队列状态寄存器(OTG_FS_GNPTXSTS)包括：
  - 周期和非周期性队列中还可以插入的请求数(最大为 8)
  - 周期性(非周期性)发送 FIFO(对于 OUT 传输)中剩余的可用空间
  - IN/OUT 命令，主机通道数和其他状态信息

由于每个请求队列可以保存至多 8 个请求，因此，应用程序可以尽可能的利用这一特性，最多提交 8 个周期性传输请求和 8 个非周期传输请求，以提高传输效率。例如，对于块传输 OUT/IN 数据，应用程序可以配置传输高达 64(每包的最大字节数)×8(最大请求数) = 512 字节的数据，实现全速设备的最高数据传输速率。这些数据将由主机自动调度而不需要应用程序插入管理。

- 应用程序通过以下步骤向主机调度发起一个周期性(非周期性)的 OUT 传输请求：
  - 配置主机通道的传输参数
  - 使能配置的通道
  - 读 OTG_FS_GNPTXSTS 寄存器的 HPTXSTS 位，确保周期性(非周期性)队列还能容纳至少 1 个请求。
  - 读 OTG_FS_HPTXSTS(OTG_FS_GNPTXSTS)寄存器，确保周期性(非周期性)的发送 FIFO 中剩余足够的空间。如果应用程序在收到周期性(非周期性)发送 FIFO 半空或全空中断后才提交传输请求的话，此步骤可省略。
  - 将数据载入相应的 FIFO 中(PUSH 寄存器)。每个通道都会配置一个 PUSH 寄存器。数据将根

据 OTG_FS_HCCHARx 寄存器的 EPTYPE 位，自动载入相应的周期性或非周期性发送 FIFO 中。当最后一个 32 位的数据被写入 FIFO 中，一个请求将被插入请求队列的末端，等待调度执行。

- 应用程序通过以下步骤，向主机调度发起一个周期性(非周期性)的 IN 传输请求：
  - 配置主机通道的传输参数
  - 设置主机通道控制寄存器的通道使能位(OTG_FS_HCCHARx 寄存器的 CHENA 位)来使能配置好的通道。此操作将在周期性(非周期性)请求队列末端插入一个传输请求，并等待调度执行。

32.2.6 SOF 触发

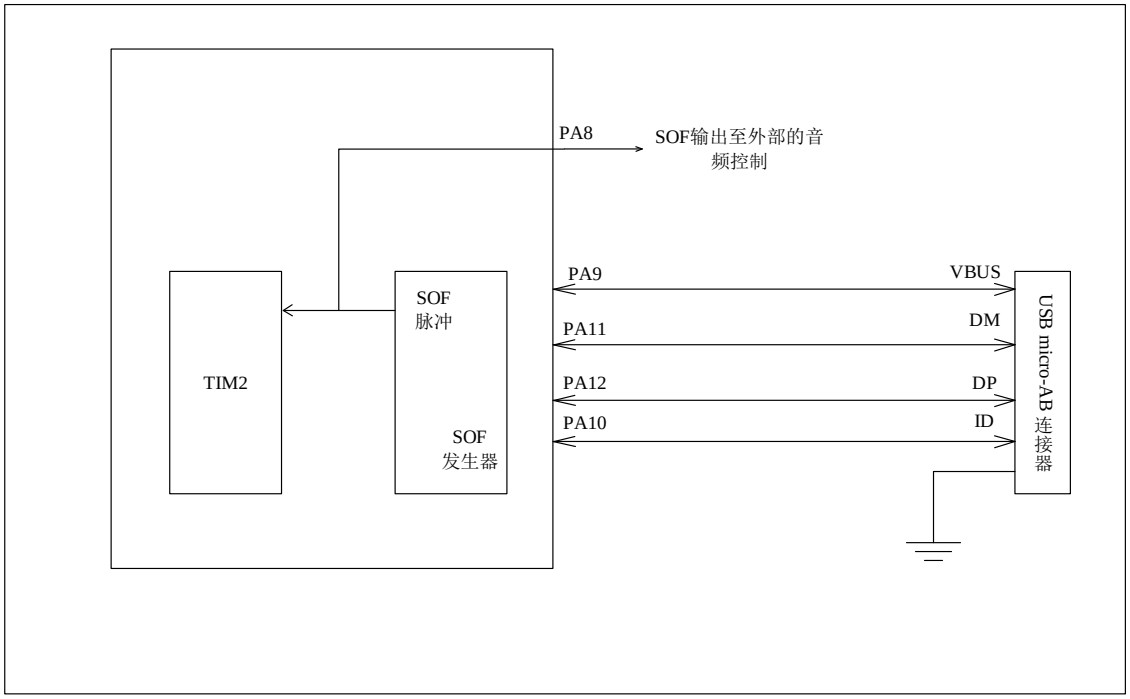


图 32-5 SOF 连接

OTG_FS 控制器提供：

- 对 SOF 的监控，跟踪和配置
- SOF 脉冲输出

由于音频设备需要与 PC 的数据流同步，或者主机需要根据音频设备的需求调整帧率，因此这些功能对于音频应用的时钟输出非常有用

32.2.6.1 主机 SOF

在主机模式下，可以通过配置主机帧间隔寄存器(OTG_FS_HFIR)来设置两个连续的 SOF(全速设备)或者保持有效(低速设备)之间时钟数目，因此，应用程序可以用它来控制 SOF 帧周期。在每个帧首都可以产生中断(OTG_FS_GINTSTS 寄存器的 SOF 位)。当前的帧号和时间都将保持不变直到在主机帧号寄存器(OTG_FS_HFNUM)中出现了下一个帧号。

每个 SOF 帧首都将产生一个宽度为 12 个系统时钟周期的 SOF 脉冲信号，通过全局控制和配置寄存器的 SOFOEN 位，可以配置在 SOF 引脚上输出这个脉冲。SOF 脉冲在内部也连接到定时器 2(TIM2)的触发输入端，因此输入捕获，输出比较和定时器都可以被 SOF 脉冲触发。

32.2.6.2 设备 SOF

在设备模式下，每次在 USB 线上收到 SOF 时，都将产生帧首中断(OTG_FS_GINTSTS 寄存器的 SOF 位)。可以从设备状态寄存器中读出当前帧号(OTG_FS_DSTS 寄存器的 FNSOF 位)。此时，会产生一个宽度为 12 个系统时钟周期的 SOF 脉冲信号，通过使能全局控制和配置寄存器的 SOF 输出使能位(OTG_FS_GCCFG 寄存器的 SOFOEN 位)可以在 SOF 引脚上输出这个脉冲信号。SOF 脉冲信号同样在内部连接到定时器 2(TIM2)的触发输入端，此时输入捕获，输出比较和定时器都可以被 SOF 脉冲触发。周期性帧结束中断(OTG_FS_GINTSTS 寄存器的 EOPF 位)用来告知应用程序，80%，85%，90%或 95%的帧已经完成，此间隔取决于设备配置寄存器的帧间隔位(OTG_FS_DCFG 寄存器的 PFIVL 位)。此功能用于判断一个帧内所有的同步传输是否都已经完成。

32.2.7 供电选项

OTG PHY 的耗电由通用控制器配置寄存器的以下 3 位决定：

- PHY 供电位(OTG_FS_GCCFG 寄存器的 PWRON 位)
  - 切换 PHY 全速收发器模块的开/关。必需预先设置才能允许 USB 通信。
- A 类 VBUS 监控使能位(OTG_FS_GCCFG 寄存器的 VBUSACEN 位)
  - 切换 A 类设备 VBUS 比较器的开/关。在 A 类设备模式时(USB 主机)，在 HNP 阶段必需使
  - 能该位。
- B 类 VBUS 监控使能位(OTG_FS_GCCFG 寄存器的 VBUSBCEN 位)
  - 切换 B 类设备 VBUS 比较器的开关。在 B 类设备模式时(USB 设备)，在 HNP 阶段必需使
  - 能该位。

当 USB 无通信或无 USB 设备连接时，暂停 USB，可以节省供电。

- 停止 PHY 时钟(OTG_FS_PCGCCTL 寄存器的 STPPCLK 位)
  - 当设置时钟门控控制寄存器的停止 PHY 时钟位时，大部分内部连接到 OTG 全速控制器的 48MHz 时钟通过门控被关闭，此时，即使应用程序仍然使能 48MHz 的输入时钟，但由 USB 时钟造成的动态功耗会大大下降。
  - 大部分的收发器会被禁止，但用于检测异步复位信号和远程唤醒事件的电路仍然处于
  - 激活状态。
- 门控 HCLK(OTG_FS_PCGCCTL 寄存器的 GATEHCLK 位)
  - 当设置时钟门控寄存器的门控 HCLK 位时，大部分内部连接到 OTG_FS 控制器的系统时钟会被门控电路关闭。只有寄存器的访问接口仍然保持有效。此时，即使应用程序仍然使能系统时钟，但由 USB 时钟造成的动态功耗会大大下降。
- USB 系统停止
  - 当 OTG_FS 控制器处于 USB 挂起状态时，应用程序需要完全关闭所有的系统时钟源来彻底降低所有功耗。先置位 PHY 时钟停止位，再设置供电控制系统模块(PWR)进入系统深度睡眠模式，将停止 USB 系统。

— 在检测到远程唤醒(作为主机)或来自 USB 总线的唤醒信号(作为设备)时，OTG_FS 控制器将自动恢复系统活动和 USB 的时钟。

为了降低功耗，只有在 OTG_FS 控制器需要访问 FIFO 时，才向 USB 数据 FIFO 提供时钟。

32.2.8 USB 数据 FIFO

下图示出 OTG_FS 控制器的框图和各个部分的功能。

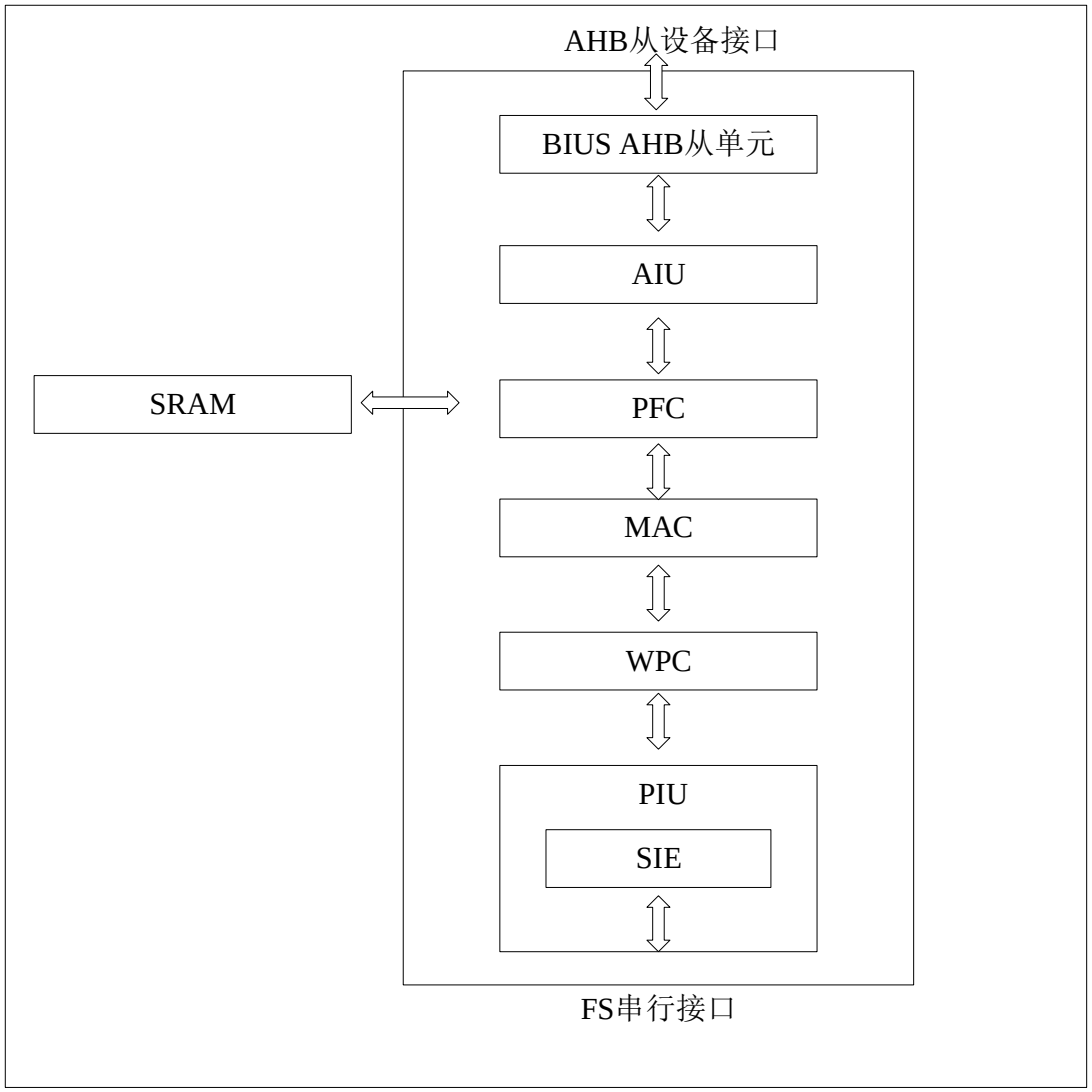


图 32-6 OTG_FS 控制器框图

BIUS：总线接口单元； AIU：应用程序接口单元； PFC：包 FIFO 控制器； MAC：媒体访问控制器； WPC：唤醒和供电控制器； PIU：PHY 接口单元； SIE：串行接口控制器

USB 系统规划了 1.25K 字节的专用 RAM 用于 FIFO 的管理。OTG_FS 控制器的包 FIFO 控制器(PFC)模块将此 RAM 划分为发送 FIFO 区域和接收 FIFO 区域，应用程序将数据暂时缓存到发送 FIFO 中，等待发送，而从 USB 总线上收到的数据则暂时缓存在接收 FIFO 中，等待应用程序读取。实际的 FIFO 数量及如何在专用 RAM 中规划这些 FIFO 都由实际的应用决定。比如在设备模式下，每个 IN 端点都会配置一个发送 FIFO，而这些 FIFO 的大小都可以由软件指定，这样，专用的 RAM 可以最优化的满足应用的需求。

32.2.9 设备模式下的 FIFO 结构

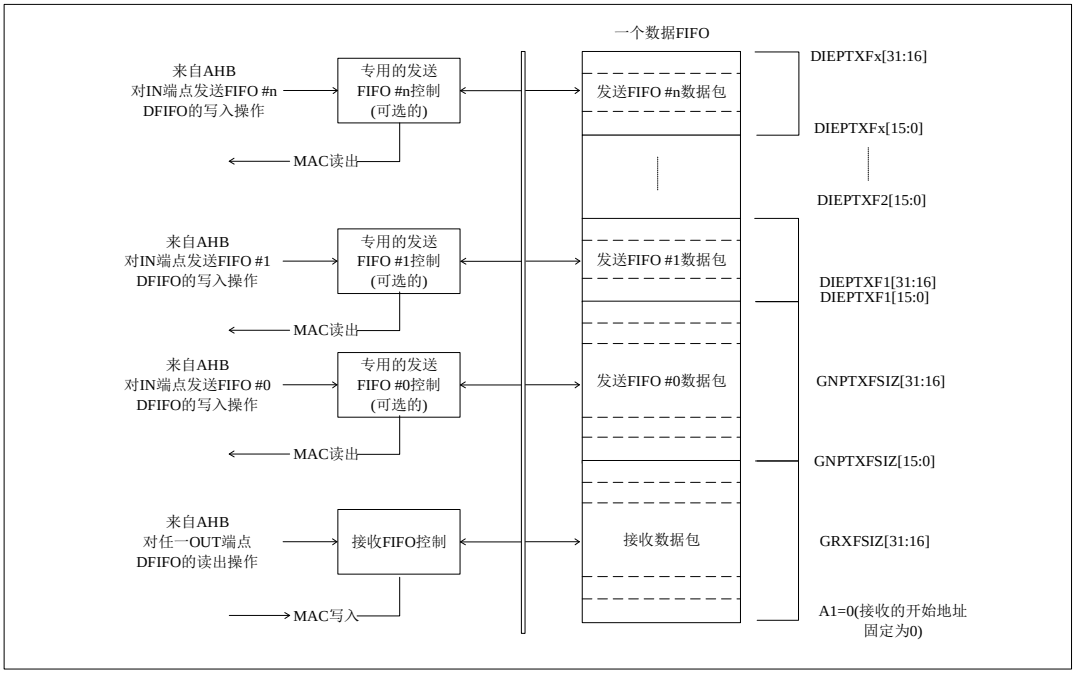


图 32-7 设备模式下的 FIFO 地址映像和 AHB 的 FIFO 操作映像

32.2.9.1 设备模式下的接收 FIFO

FS_OTG 控制器在设备模式下，使用一个单独的 FIFO 缓存所有 OUT 端点的数据。收到的数据包依次缓存在 RX FIFO 的可用空间中。收到的数据包的状态(包括 OUT 端点号，字节数，数据 PID 号和收到数据的有效性)由 PFC 模块写在收到的数据前。如果接收 FIFO 没有剩余空间，控制器将以 NACK 来回应主机，同时产生相应端点的中断。接收 FIFO 的大小可由接收 FIFO 长度寄存器(OTG_FS_GRXFSIZ)配置。

使用单个 RX FIFO 的架构，使得 USB 设备能更有效的利用整个接收 RAM 空间。

- 所有的 OUT 端点共享整个 RAM 空间(共享的 FIFO)
- OTG_FS 控制器可以最大限度的按照主机发送 OUT 数据的次序来利用接收 FIFO

只要接收 FIFO 还有至少一个数据包等待应用程序获取，应用程序就会不断收到接收 FIFO 非空中断(OTG_FS_GINTSTS 寄存器的 RXFLVL 位)。应用程序通过读接收状态和弹出寄存器(OTG_FS_GRXSTSP)从而获得数据包信息，并通过读相应端点的 POP 寄存器可以从接收 FIFO 中获取数据包。

32.2.9.2 设备模式下的发送 FIFO

共享 FIFO 的模式不适用于 IN 传输，只有提前知道主机请求的次序或者能预测进程的处理过程才能将所有端点的数据包都按次序传送到同一个发送 FIFO 中。所以在设备模式下，控制器为每个 IN 端点都配置了一个专用的 FIFO 。应用程序可以通过非周期性传输 FIFO 长度寄存器(OTG_FS_GNPTXFSIZ)来配置 IN 端点 0 的 FIFO 长度，并通过设备 IN 端点传输 FIFO 寄存器(OTG_FS_DIEPTXFx)来配置 IN 端点 x 的 FIFO 长度。

专用的 FIFO 的架构非常灵活，大大减少应用程序的负荷。这些 FIFO 没有请求序列，也没有必要在 USB 主机访问非周期性端点的时候预测访问的顺序。

据根在 AHB 配置寄存器中配置的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)的值，OTG_FS 控制器会产生发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)，提示对应 IN 端点的发送 FIFO 已经半空或者全空。应用程序先通过读设备所有端点中断寄存器(OTG_FS_DAINT)获得需要服务的 IN 端点号，然后通过读设备 IN 端点 x 的发送 FIFO 状态寄存器(OTG_FS_DTXFSTSx)检查 FIFO 是否有足够的剩余空间，最后通过写相应端点的 PUSH 寄存器来向发送 FIFOx 传送数据。

32.2.10 主机模式下的 FIFO 结构

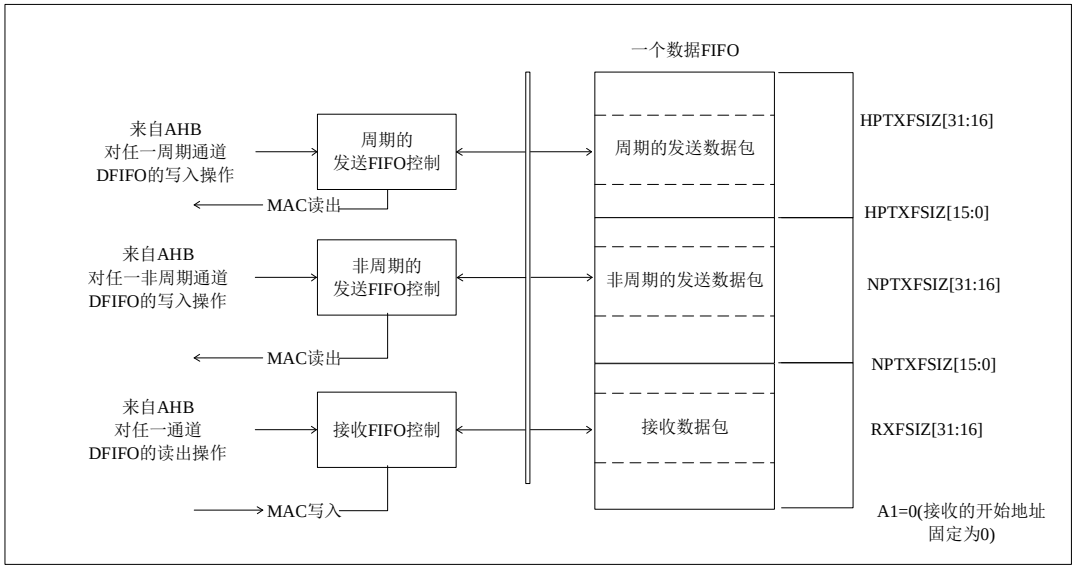


图 32-8 主机模式 FIFO 地址映像和 AHB 的 FIFO 操作映像

32.2.10.1 主机模式下的接收 FIFO

在主机模式下使用一个接收 FIFO 管理所有的周期性和非周期性的传输，此 FIFO 用于暂存从 USB 总线上收到但还未传输到系统存储区的数据(收到的数据包)。来自任一远程 IN 端点的数据包都将按次序暂存在 FIFO 中。收到数据包的状态（包括主机通道号，字节数，数据 PID 和收到数据的有效性）也储存在 FIFO 中。应用程序可以通过接收 FIFO 长度寄存器(OTG_FS_GRXFSIZ)来配置这个接收 FIFO 的长度。

使用单个接收 FIFO 的架构，使得 USB 主机能更有效的利用整个接收 RAM 空间。

- 所有配置好的 IN 通道共享整个 RAM 空间(共享的 FIFO)
- OTG_FS 控制器可以最大限度的按照主机发送 IN 命令的序列来利用接收 FIFO

只要接收 FIFO 还有至少一个数据包等待应用程序获取，应用程序就会收到接收 FIFO 非空中断。应用程序可以通过读接收状态和弹出寄存器获得数据包信息，并通过读相应端点的 POP 寄存器从接收 FIFO 中获取数据包。

32.2.10.2 主机模式下的发送 FIFO

在主机模式下，控制器使用一个发送 FIFO 来管理所有的非周期性(控制和块传输)OUT 传输，使用另一个发送 FIFO 来管理所有的周期性(同步和中断)OUT 传输。这两个 FIFO 用于暂存需要发送到 USB 总线的数据(发送的数据包)。可以通过主机周期性(非周期性)传输 FIFO 长度寄存器(OTG_FS_HPTXFSIZ/ OTG_FS_GNPTXFSIZ)来配置周期性(非周期性)发送 FIFO 的长度。

使用两个发送 FIFO 是因为要保证一个 USB 帧里周期性传输的高优先级。在每个帧开始的时候，内置的主机调度器先处理周期性请求队列再处理非周期性请求队列。

使用两个发送 FIFO 使 USB 主机能方便的分开优化管理周期性和非周期性的数据缓存。

- 所有用于周期性(非周期性)OUT 传输的主机通道共享同一块 RAM 缓存区(共享的 FIFO)
- OTG_FS 控制器可以根据主机软件对 OUT 命令的调度最大限度的利用周期性(非周期性)的发送 FIFO。

根据在 AHB 配置寄存器中配置的周期性发送 FIFO 空标志位(OTG_FS_GAHBCFG 寄存器的 PTXFELVL 位)的值，OTG_FS 控制器会产生周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 PTXFE 位)，提示周期性发送 FIFO 已经半空或者全空。应用程序需要先读周期性发送 FIFO 和队列状态寄存器(OTG_FS_HPTXSTS)来获得周期性发送 FIFO 和周期性请求队列是否有剩余空间的信息，只有在发送 FIFO 和请求队列都有剩余空间的情况下，应用程序才能将数据写入 FIFO 中。

根据在 AHB 配置寄存器中配置的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)的值，OTG_FS 控制器会产生非周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)，提示非周期性发送 FIFO 已经半空或者全空。应用程序需要先读非周期性发送 FIFO 和队列状态寄存器(OTG_FS_GNPTXSTS)来获得非周期性发送 FIFO 和非周期性请求队列是否有剩余空间的信息，只有在发送 FIFO 和请求队列都有剩余空间的情况下，应用程序才能将数据写入 FIFO 中。

32.2.11 USB 系统性能

通过配置大容量的 RAM 缓存区、高自由度的 FIFO 长度配置、AHB PUSH/POP 寄存器的 32 位快速访问、和高级的 FIFO 控制机制，使得 OTG_FS 控制器能最大限度的利用 RAM 空间而不必顾虑 USB 数据的次序，USB 系统达到了最优的性能。

这些优势如下：

- 应用程序的负荷减少，不用干预 USB 传输，优化了 CPU 带宽的使用率

– 当数据通过 USB 系统有效率的传送时，应用程序可以提前准备好大量的传输数据。

– 应用程序获得了大量的时间从一个单一的接收 FIFO 中获取数据

- USB 模块能够保持全速工作状态，也就是提供最大的全速带宽（尽量多的硬件自动运行，尽量少的软件参与）

– USB 控制器管理一个大型的数据缓存区，可以自主的管理数据在 USB 总线上的传输

– USB 控制器管理一个大型的数据接收缓存区，可以自主的从 USB 总线上接收数据并填入缓存区

由于 OTG_FS 控制器可以非常有效的使用 1.25K 字节的 RAM 缓存区，并且对于一个全速的帧来说 1.25K 字节已经足够用于管理发送/接收的数据，因此 USB 系统能在每个 USB 帧内(1ms)提供最大的全速数据传输率。

32.2.12 OTG_FS 中断

无论 OTG_FS 控制器工作在设备模式还是主机模式，应用程序都不能访问另一个模式下的寄存器组。如果应用程序有非法的访问，会产生模式不匹配中断并影响控制器中断寄存器的相应位(OTG_FS_GINTSTS 寄存器的 MMIS 位)。当控制器从一种模式切换到另一种模式时，新模式下的寄存器组都需要和上电复位时一样的重新初始化。

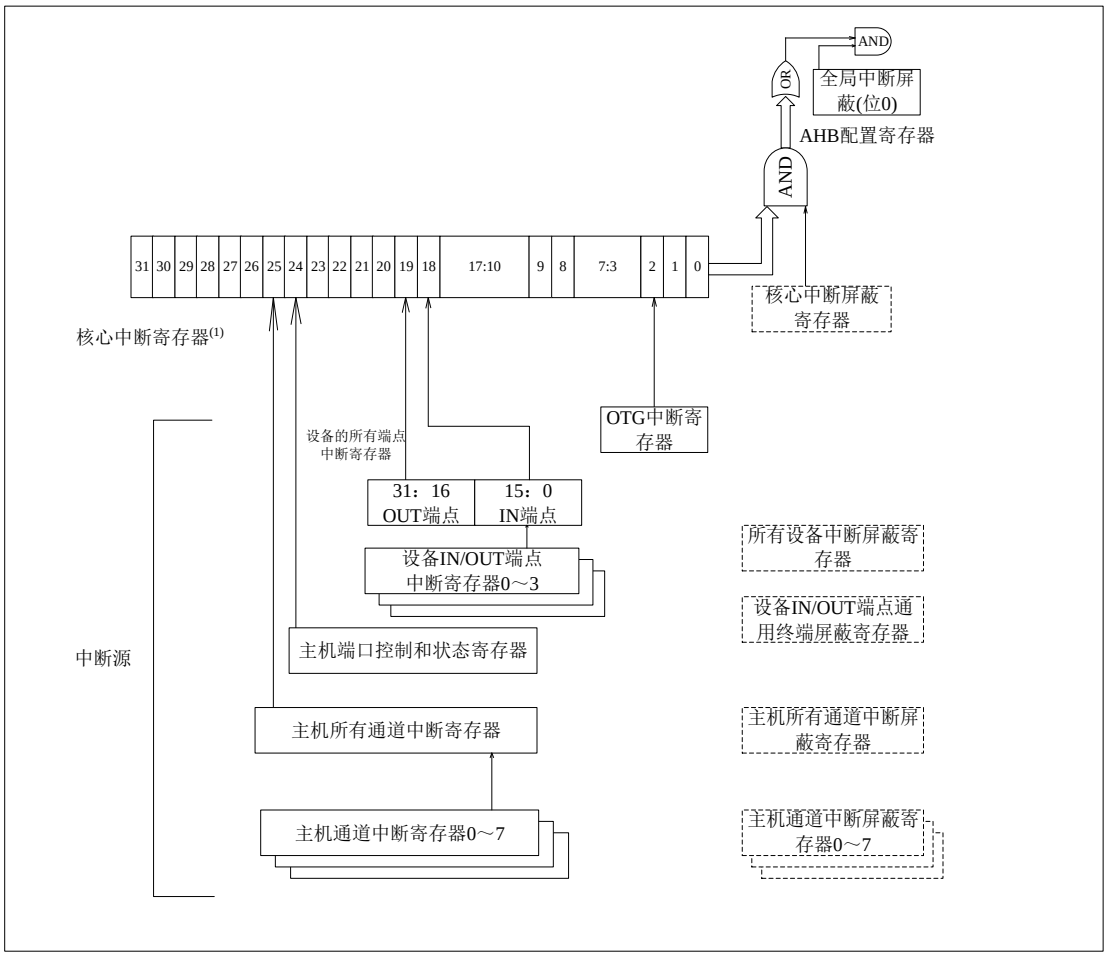


图 32-9 中断架构

(1) 请参考控制器中断寄存器的相关位

32.3 USBFS 寄存器

寄存器宽度为 32 位。

应用程序通过 AHB 从接口来读写控制和状态寄存器(CSRx)，从而实现对 OTG_FS 控制器的控制。这些寄存器都是 32 位访问的，并且 32 位地址对齐。包括以下寄存器组：

- 控制器全局寄存器组
- 主机模式寄存器组
- 主机全局寄存器组
- 主机端口 CSR 寄存器组
- 主机通道相关寄存器组
- 设备模式寄存器组
- 设备全局寄存器组
- 设备端点相关寄存器组
- 供电和时钟控制寄存器组
- 数据 FIFO(DFIFO)访问寄存器组

只有控制器全局寄存器，供电和时钟控制寄存器，数据 FIFO 访问寄存器以及主机端口控制和状态寄存器在主机模式和设备模式下都有效。无论 OTG_FS 控制器工作在主机模式还是设备模式下，应用程序都不能访问另一种模式下的寄存器组。如果应用程序发生了非法访问，会产生模式不匹配中断并影响控制器中断寄存器的相应位(OTG_FS_GINTSTS 寄存器的 MMIS 位)。当控制器从一种模式切换到另一种模式时，新模式下的寄存器组都需要和上电复位时一样的重新初始化。

必须以字(32 位)的方式操作这些外设寄存器。

32.3.1 CSR 存储器映像

主机模式寄存器和设备模式寄存器占据不同的地址。所有的寄存器都由 AHB 时钟驱动。



图 32-10 CSR 存储器映像

在设备模式下 x 为 3，在主机模式下 x 为 7

全局 CSR 地址映像

这些寄存器在主机模式和设备模式下都有效。

表 32-1 控制器全局控制和状态寄存器(CSRs)

寄存器代号	偏移地址	寄存器名
OTG_FS_OTGCTL	0x000	OTG_FS 控制和状态寄存器
OTG_FS_GOTGINT	0x004	OTG_FS 中断寄存器
OTG_FS_GAHBCFG	0x008	OTG_FS AHB 配置寄存器
OTG_FS_GUSBCFG	0x00C	OTG_FS USB 配置寄存器

OTG_FS_GRSTCTL	0x010	OTG_FS 复位寄存器
OTG_FS_GINTSTS	0x014	OTG_FS 控制器中断寄存器
OTG_FS_GINTMSK	0x018	OTG_FS 中断屏蔽寄存器
OTG_FS_GRXSTSR	0x01C	OTG_FS 接收状态调试读/OTG 状态读和弹出寄存器
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	OTG_FS 接收 FIFO 长度寄存器
OTG_FS_GNPTXFSIZ	0x028	OTG_FS 非周期性发送 FIFO 长度寄存器
OTG_FS_GNPTXSTS	0x02C	OTG_FS 非周期性发送 FIFO/队列状态寄存器
OTG_FS_GCCFG	0x038	OTG_FS 通用控制配置寄存器
OTG_FS_CID	0x03C	OTG_FS 控制器 ID 寄存器
OTG_FS_HPTXFSIZ	0x100	OTG_FS 主机模式周期性发送 FIFO 长度寄存器
OTG_FS_DIEPTXFx	0x104 0x124 … 0x13C	OTG_FS 设备模式 IN 端点 TX FIFO 长度寄存器(x = 1…3，指明 FIFO 的编号)

主机模式 CSR 地址映像

这些寄存器在每次切换到主机模式时都需要配置。

表 32-2 主机模式下的控制和状态寄存器(CSRs)

寄存器代号	**偏移地址**	**寄存器名**
OTG_FS_HCFG	0x400	OTG_FS 主机模式配置寄存器
OTG_FS_HFIR	0x404	OTG_FS 主机模式帧间隔寄存器
OTG_FS_HFNUM	0x408	OTG_FS 主机模式帧号码/剩余帧时间寄存器
OTG_FS_HPTXSTS	0x410	OTG_FS 主机模式周期性 TX FIFO/队列状态寄存器
OTG_FS_HAINT	0x414	OTG_FS 主机模式所有通道中断寄存器
OTG_FS_HAINTMSK	0x418	OTG_FS 主机模式所有通道中断屏蔽寄存器
OTG_FS_HPRT	0x440	OTG_FS 主机模式端口控制和状态寄存器
OTG_FS_HCCHARx	0x500 0x520 … 0x6E0	OTG_FS 主机模式通道 x 特性寄存器(x = 0 … 7，x 指示通道编号)
OTG_FS_HCINTx	0x508	OTG_FS 主机模式通道 x 中断寄存器(x = 0 … 7，x 指示通道编号)
OTG_FS_HCINTMSKx	0x50C	OTG_FS 主机模式通道 x 中断屏蔽寄存器(x = 0 … 7，x 指示通道编号)
OTG_FS_HCTSIZx	0x510	OTG_FS 主机模式通道 x 传输长度寄存器(x = 0 … 7，x 指示通道编号)

设备模式 CSR 地址映像

这些寄存器在每次切换到设备模式时都必须配置。

表 32-3 设备模式控制和状态寄存器

寄存器代号	**偏移地址**	**寄存器名**
OTG_FS_DCFG	0x800	OTG_FS 设备模式配置寄存器
OTG_FS_DCTL	0x804	OTG_FS 设备模式控制寄存器
OTG_FS_DSTS	0x808	OTG_FS 设备模式状态寄存器
OTG_FS_DIEPMSK	0x810	OTG_FS 设备模式 IN 端点共有中断屏蔽寄存器
OTG_FS_DOEPMSK	0x814	OTG_FS 设备模式 OUT 端点共有中断屏蔽寄存器
OTG_FS_DAINT	0x818	OTG_FS 设备模式所有端点中断寄存器
OTG_FS_DAINTMSK	0x81C	OTG_FS 设备模式所有端点中断屏蔽寄存器
OTG_FS_DVBUSDIS	0x828	OTG_FS 设备模式 VBUS 下电时间寄存器
OTG_FS_DVBUSPULSE	0x82C	OTG_FS 设备模式 VBUS 脉冲时间寄存器

OTG_FS_DIEPEMPMSK	0x834	OTG_FS 设备模式 IN 端点 FIFO 空中断屏蔽寄存器
OTG_FS_DIEPCTL0	0x900	OTG_FS 设备模式 IN 控制端点 0 控制寄存器
OTG_FS_DIEPCTLx	0x920 0x940 … 0xAE0	OTG_FS 设备模式 IN 端点 x 控制寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DIEPINTx	0x908	OTG_FS 设备模式 IN 端点 x 中断寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DIEPTSIZ0	0x910	OTG_FS 设备模式 IN 端点 0 传输长度寄存器
OTG_FS_DTXFSTSx	0x918	OTG_FS 设备模式 IN 端点 TX FIFO 状态寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DIEPTSIZx	0x930 0x950 … 0xAF0	OTG_FS 设备模式 IN 端点 x 传输长度寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DOEPCTL0	0xB00	OTG_FS 设备模式 OUT 控制端点 0 控制寄存器
OTG_FS_DOEPCTLx	0xB20 0xB40 … 0xCC0 0xCE0 0xCFD	OTG_FS 设备模式 OUT 端点 x 控制寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DOEPINTx	0xB08	OTG_FS 设备模式 OUT 端点 x 中断寄存器(x = 1…3，x 指示端点编号)
OTG_FS_DOEPTSIZx	0xB10	OTG_FS 设备模式 OUT 端点 x 传输长度寄存器(x = 1…3，x 指示端点编号)

数据 FIFO(DFIFO)访问寄存器址映射

这组寄存器列在主机模式和设备模式下都有效，用于读写指定方向的特殊端点或通道的 FIFO。如果一个主机模式下的通道是 IN 类型的，相对应的 FIFO 只能进行读操作。同样地，如果一个主机模式下的通道是 OUT 类型的，相对应的 FIFO 只能进行写操作。

表 32-4 数据 FIFO(DFIFO)访问寄存器图

数据 FIFO(DFIFO)访问寄存器段	**地址范围**	**访问方式**
设备模式下 IN 端点 0/主机模式下 OUT 通道 0：DFIFO 只写	写操作	0x1000~0x1FFC
设备模式下 OUT 端点 0/主机模式下 IN 通道 0：DFIFO 只读	读操作	
设备模式下 IN 端点 1/主机模式下 OUT 通道 1：DFIFO 只写	写操作	0x2000~0x2FFC
设备模式下 OUT 端点 1/主机模式下 IN 通道 1：DFIFO 只读	读操作	
……	……	……
设备模式下 IN 端点 x/主机模式下 OUT 通道 x：DFIFO 只写	写操作	0xX000~0xXFFC
设备模式下 OUT 端点 x/主机模式下 IN 通道 x：DFIFO 只读	读操作	

表中的 x 在设备模式下为 3，在主机模式下为 7

供电和时钟控制 CSR 寄存器映像

只有一个寄存器用来控制供电和时钟控制，此寄存器在设备模式下和主机模式下都有效。

表 32-5 供电和门控控制和状态寄存器

寄存器名	**缩写**	**偏移地址**
供电和门控时钟控制寄存器	PCGCR	0xE00~0xE04
保留	0xE05~0xFFF	

32.4 OTG_FS 全局寄存器

这些寄存器在设备模式和主机模式下都有效，并且在模式切换时不需要重新初始化。如果没有特别注明，寄存器中每位的值都由二进制数表示。

32.4.1 OTG_FS 控制和状态寄存器(OTG_FS_GOTGCTL)

Address:0x0000

Reset value:0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.												BSVLD	ASVLD	DBCT	CIDSTS
												R	R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Res.						SRQ	SRQSCS
				RW	RW	RW	R							RW	R

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19	BSVLD	R	0	BSVLD：B 类会话有效（B-session valid) 指示设备模式收发器的状态 0：B 类会话无效； 1：B 类会话有效。 在 OTG 模式，用户可以用此位来判断设备是否连入主机 注意：仅在设备模式下有效。
18	ASVLD	R	0	ASVLD：A 类会话有效（A-session valid) 指示主机模式收发器的状态 0：A 类会话无效； 1：A 类会话有效。 注意：仅在主机模式下有效。
17	DBCT	R	0	DBCT：长/短反射时间（Long/short debounce time) 指示检测到的连接的反射时间 0：长的反射时间，用于物理连接(100ms + 2.5us)； 1：短的反射时间，用于软件连接(2.5us)。 注意：仅在主机模式下有效
16	CIDSTS	R	1	CIDSTS：连接的 ID 线状态（Connector ID status) 指示连接器上的 ID 线状态 0：OTG_FS 处于 A 类设备模式； 1：OTG_FS 处于 B 类设备模式。 注意：在设备模式和主机模式下都有效
15:12	保留	-	-	-
11	DHNPEN	RW	0	DHNPEN：设备模式 HNP 使能（Device HNP enabled) 应用程序在成功收到一个来自 USB 主机的 SetFeature.SetHNPEnable 命令时设置此位。 0：禁止设备模式 HNP； 1：使能设备模式 HNP。 注意：仅在设备模式下有效。
10	HSHNPEN	RW	0	HSHNPEN：主机模式 HNP 使能（Host set HNP enable) 应用程序在成功使能连接上的设备的 HNP 时(通过 SetFeature.SetHNPEnable 命令)设置此位。 0：禁止主机模式 HNP； 1：使能主机模式 HNP。 注意：仅在主机模式下有效
9	HNPRQ	RW	0	HNPRQ：HNP 请求（HNP request)

				应用程序在需要向 USB 主机发送一个 HNP 请求时设置此位。当 OTG 中断寄存器的主机协商成功状态转换位(OTG_FS_GOTGINT 寄存器的 HNSSCHG 位)被置位时，应用程序可以通过写'0'清除此位。控制器在清除 HNSSCHG 位时清除此位。 0：无 NHP 请求； 1：HNP 请求。 注意：仅在设备模式下有效。
8	HNGSCS	R	0	HNGSCS：主机协商成功（Host negotiation success) 当主机协商成功时，控制器会设置此位。当应用程序设置 HNP 请求位(HNPRQ)时，控制器会清除此位。 0：主机协商失败； 1：主机协商成功。 注意：仅在设备模式下有效
7：2	保留	-	-	-
1	SRQ	RW	0	SRQ：会话请求（Session request) 应用程序通过设置此位在 USB 总线上发起一个会话请求。当 OTG 中断寄存器的主机协商成功状态改变位(OTG_FS_GOTGINT 寄存器的 HNSSCHG 位)被置位时，应用程序可以通过写'0'来清除此位。控制器在清除 HNSSCHG 位时清除此位。 如果用户使用 USB1.1 全速串行收发器接口来发起一个会话请求，应用程序必需在 B 类会话有效位(OTG_FS_GOTGCTL 寄存器的 BSVLD 位)被清除后，等待 VBUS 线降到 0.2V。不同的 PHY 使用不同的等待时间，由 PHY 的供应商掌控。 0：无会话请求； 1：会话请求。 注意：仅在设备模式下有效。
0	SRQSCS	R	0	SRQSCS：会话请求成功（Session request success) 控制器在会话请求成功后设置此位。 0：会话请求失败； 1：会话请求成功。 注意：仅在设备模式下有效。

32.4.2 OTG_FS 中断寄存器(OTG_FS_GOTGINT)

Address:0x0004

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.												DBCDNE RC_W1	ADTOCHG RC_W1	HNGDET RC_W1	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.						HNSSCHG RC_W1	SRSSCHG RC_W1	Res.					SEDET RC_W1	Res.	

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19	DBCDNE	RC_W1	0	DBCDNE：反射完成（Debounce done) 在设备连接线路反射结束后，控制器会设置此位。应用程序可以在检测到此中断后发起 USB 复位

				信号。此位仅在控制器 USB 配置寄存器的 HNP 有效或者 SRP 有效位(OTG_FS_GUSBCFG 寄存器 的 HNPCAP 位或 SRPCAP 位)置位时才有效。 注意：仅在主机模式下有效。
18	ADTOCHG	RC_W1	0	ADTOCHG：A 类设备超时（A-device timeout change) 控制器在 A 类设备等待 B 类设备插入超时时设置此位。 注意：在主机模式和设备模式下都有效
17	HNGDET	RC_W1	0	HNGDET：主机协商检测（Host negotiation detected) 控制器在检测到 USB 总线上的主机协商请求时设置此位。 注意：在主机模式和设备模式下都有效。
16:10	保留	-	-	-
9	HNSSCHG	RC_W1	0	HNSSCHG: 主机协商成功状态改变 (Host negotiation success status change) 控制器在 USB 主机协商请求成功或者失败时设置此位。应用程序需要通过读 OTG 控制和状态寄存 器的主机协商成功位(OTG_FS_GOTGCTL 寄存器的 HNGSCS 位)来获得成功或者失败的信息。 注意：在主机模式和设备模式下都有效。
8	SRSSCHG	RC_W1	0	SRSSCHG: 会话请求成功状态改变 (Session request success status change) 控制器在会话请求成功或者失败时设置此位。应用程序需要通过读 OTG 控制和状态寄存器的会话 请求成功位(OTG_FS_GOTGCTL 寄存器的 SRQSCS 位)来获得成功或者失败的信息。 注意：在主机模式和设备模式下都有效。
7:3	保留	-	-	-
2	SEDET	RC_W1	0	SEDET：检测到会话结束（Session end detected) 控制器在检测到 VBUS<0.8V 时，设置此位，指示 B 类设备的 VBUS 线的电平无效。
1:0	保留	-	-	-

32.4.3 OTG_FS AHB 配置寄存器(OTG_FS_GAHBCFG)

Address:0x0008

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	PTXFELVL	RW	0	PTXFELVL：周期性发送 FIFO 空级别（Periodic TxFIFO empty level) 指示在何种情况下需要触发控制器中断寄存器的周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 PTXFE 位)：

				0：PTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示周期性发送 FIFO 已经半空； 1：PTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示周期性发送 FIFO 已经全空。 注意：仅在主机模式下有效。
7	TXFELVL	RW	0	TXFELVL：发送 FIFO 空级别（TxFIFO empty level) 在设备模式下,此位指示在何种情况下需要触发 IN 端点发送 FIFO 空中断(OTG_FS_DIEPINTx 寄存器的 TXFE 位)： 0：TXFE(在 OTG_FS_DIEPINTx 寄存器中)中断指示 IN 端点 TX FIFO 已经半空； 1：TXFE(在 OTG_FS_DIEPINTx 寄存器中)中断指示 IN 端点 TX FIFO 已经全空。 在主机模式下，此位指示在何种情况下需要触发非周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)： 0：NPTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示非周期性发送 FIFO 已经半空； 1：NPTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示非周期性发送 FIFO 已经全空。
6:1	保留	-	-	-
0	GINT	RW	0	GINT：全局中断屏蔽（Global interrupt mask) 应用程序可以通过此位选择屏蔽或者不屏蔽中断。但无论是否设置此位，控制器仍然会更新中断状态寄存器。 0：屏蔽中断； 1：不屏蔽中断。 注意：在设备模式和主机模式下都有效

32.4.4 OTG_FS 配置寄存器(OTG_FS_GUSBCFG)

Address:0x000C

Reset value:0x10001440

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FDMOD	FHMOD	Res.												
	RW	RW													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.		TRDT [3:0]				HNPCAP	SRPCAP	Res.					TOCAL [2:0]		
		RW				R/RW	R/RW						RW		

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30	FDMOD	RW	0	FDMOD：强制设备模式（Force device mode) 不管 OTG_FS_ID 输入引脚的状态如何，设置此位为'1'将强制控制器进入设备模式。 0：普通模式； 1：强制设备模式。

				设置此位后，应用程序需要等待至少 25ms，以便设置生效。 注意：在主机模式和设备模式下都有效。
29	FHMOD	RW	0	FHMOD：强制主机模式（Force host mode) 不管 OTG_FS_ID 输入引脚的状态如何，设置此位为'1'将强制控制器进入主机模式。 0：普通模式 1：强制主机模式 设置此位后，应用程序需要等待至少 25ms，以便设置生效。 注意：在主机模式和设备模式下都有效。
28:14	保留	-	-	-
13:10	TRDT [3:0]	RW	4'h5	TRDT：USB 总线的周转时间（USB turnaround time) 以 PHY 层的时钟周期为单位设置周转时间。 设置一个 MAC 请求到包 FIFO 控制器(PFC)要求从 DFIFO(SPRAM)获取数据的响应时间。 这些位必需被设置为： 4'h5：当 MAC 接口是 16 位 UTMIFS； 4'h9：当 MAC 接口是 8 位 UTMIFS。 注意：仅在设备模式下有效。
9	HNPCAP	R/RW	0	HNPCAP：HNP 使能（HNP-capable) 应用程序通过此位来使能 OTG_FS 控制器的 HNP 功能。 0：禁止 HNP 功能； 1：使能 HNP 功能。 注意：在设备模式和主机模式下都有效。
8	SRPCAP	R/RW	0	SRPCAP： SRP 使能（SRP-capable)应用程序通过此位来使能 OTG_FS 控制器的 SRP 功能。如果控制器工作在无 SRP 功能的 B 类设备模式下，就不能要求连接的 A 类设备(主机)使能 VBUS 线并发起一个会话。0：禁止 SRP 功能；1：使能 SRP 功能。注意：在设备模式和主机模式下都有效。
7：3	保留	-	-	-
2:0	TOCAL [2:0]	RW	3'h0	TOCAL：FS 超时校准（FS timeout calibration)考虑到由 PHY 带来的额外的时延，应用程序可以调整控制器的全速包超时判断时延，这些位以 PHY 时钟的数目给出了调整时延的长度。不同的 PHY 在控制总线状态时，带来的额外时延是不同的，通过这些位可以适应不同的总线时延。USB 标准规定的全速包的超时判断是 16 到 18 个 BIT 时间。应用程序需要根据枚举的速度来配置此位。每个 PHY 时钟增加的 BIT 时间是 0.25 个 BIT 时间。

32.4.5 OTG_FS 复位寄存器(OTG_FS_GRSTCTL)

Address:0x0010

Reset value:0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBIDL	Res	CSftRstDone	Res												
R		RC_W1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					TXFNUM[4:0]					TXFFLSH	RXFFLSH	Res	FCRST	HSRST	CSRST
					RW					RS	RS		RS	RS	RS

Bit	Name	R/W	Reset Value	Function
31	AHBIDL	R	1	AHBIDL：AHB 主控模块空闲（AHB master idle) 指示 AHB 主控模块是否在空闲状态。 注意：在设备模式和主机模式下都有效。
30	保留	-	-	-
29	CSftRstDone	RC_W1	0	Core 软件复位结束 当所有必要的逻辑在 Core 中复位时时，Core 设置这个位。该位和 USBFS_GRSTCTL 中的 CSRST(bit0）被应用程序清除 0：没有 reset 1：Core 软件复位结束 注意：在设备模式和主机模式下都有效。
28:11	保留	-	-	-
10:6	TXFNUM[4:0]	RW	5’h0	TXFNUM：发送 FIFO 编号（TxFIFO number) 此位指示哪个 FIFO 需要通过发送 FIFO 刷新位来刷新。只有在控制器清除了发送 FIFO 刷新位时， 应用程序才能修改此位。 ● 5’b00000： 主机模式下表示非周期性发送 FIFO 设备模式下表示发送 FIFO 0 ● 5’b 00001： 主机模式下表示周期性发送 FIFO 设备模式下表示发送 FIFO 1 ● 5’b 00010： 设备模式下表示发送 FIFO 2 …… ● 5’b 00101： 设备模式下表示发送 FIFO 5 ● 5’b 10000： 在主机模式或设备模式下刷新所有发送 FIFO 注意：在主机模式和设备模式下都有效。
5	TXFFLSH	RS	0	TXFFLSH：发送 FIFO 刷新（TxFIFO flush)此位用于刷新单独的或所有的发送 FIFO，当控制器正在进行传输时，不能设置此位。应用程序只有在检测了控制器既没有向发送 FIFO 写数据也没有从发送 FIFO 读数据时，才能设置此位。可以通过以下方式来检测：读：NAK 有效中断，确保控制器没有从 FIFO 读数据。写：OTG_FS_GRSTCTL 寄存器的 AHBIDL 位，确保控制器没有在写 FIFO。注意：在设备模式和主机模式下都有效。
4	RXFFLSH	RS	0	RXFFLSH：接收 FIFO 刷新（RxFIFO flush)应用程序可以通过此位来刷新整个接收 FIFO，但必需确保控制器没有在传输数据。应用程序只有在检测了控制器既没有写接收 FIFO 也没有读接收 FIFO 时，才能设置此位。应用程序只有在等待此位重新被控制器清除后，才能进行其他的操作。此位需要 8 个时钟周期(以 PHY 或 AHB 时钟中较慢的那个为准)才能清除。注意：在主机模式和设备模式下都有效。
3	保留	-	-	-

2	FCRST	RS	0	FCRST：主机帧计数器复位（Host frame counter reset)应用程序通过写此位来复位控制器内部的帧计数器。在帧计数器复位后，控制器发送的首个 SOF 的帧号为 0。注意：仅在主机模式下有效。
1	HSRST	RS	0	HSRST：HCLK 软件复位（HCLK soft reset)应用程序可以使用此位来刷新 AHB 时钟域的控制逻辑。仅由 AHB 时钟域驱动的模块进行复位。FIFO 不会受此位影响。根据协议，所有的由 AHB 时钟域驱动的状态机都会在完成 AHB 当前传输后复位到空闲状态。清零 AHB 时钟域驱动的状态机中的 CSR 控制位。为了清除此中断，用于控制中断状态及由 AHB 时钟域驱动的状态机产生的状态屏蔽位将被清除。由于中断状态位并没有被清除，因此应用程序仍然可以获得在设置此位后产生的任何控制器事件的状态。控制器会自动在所有必要的逻辑模块复位后清除此位。这一操作将维持若干个时钟周期，具体由控制器当前所处的状态决定。注意：在设备模式和主机模式下都有效。
0	CSRST	RW	0	CSRST：控制器软件复位（Core soft reset)复位 HCLK 和 PCLK 驱动域：清除除了以下位之外的所有中断和 CSR 寄存器：OTG_FS_PCGCCTL 寄存器的 RSTPDMODL 位 OTG_FS_PCGCCTL 寄存器的 GAYEHCLK 位 OTG_FS_PCGCCTL 寄存器的 PWRCLMP 位 OTG_FS_PCGCCTL 寄存器的 STPPCLK 位 OTG_FS_HCFG 寄存器的 FSLSPCS 位 OTG_FS_DCFG 寄存器的 DSPD 位所有模块的状态机(除了 AHB 从单元)都将复位到空闲状态，所有的发送 FIFO 和接收 FIFO 都被刷新。所有在 AHB 主模块上的数据传输都在完成 AHB 传输的最后一个数据阶段后立即结束。所有在 USB 上的传输都立即结束。应用程序可以在任何需要复位控制器的时侯设置此位。控制器将在所有必需的逻辑模块复位之后自动清除此位，这段操作将维持若干个时钟周期，具体由控制器当前所处的状态决定。一旦此位被清除，软件必需在激活 PHY 域之前等待至少 3 个 PHY 时钟周期(同步操作的时延)。应用程序在开始其他操作之前必需确保本寄存器的位 31 为'1'(即 AHB 主控模块空闲)。通常，只有在软件开发阶段和在上面列出的 USB 配置寄存器中动态修改 PHY 选择位时才需要进入软件复位操作。如果用户改变了 PHY 的配置，与之相应的时钟会被选中并应用到 PHY 域。一旦选中了新的时钟，PHY 域需要进行复位才能进一步的操作。注意: 在设备模式和主机模式下都有效

32.4.6 OTG_FS 控制器中断寄存器(OTG_FS_GINTSTS)

Address:0x0014

Reset value:0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPINT	SRQINT	DISCINT	CIDSCHG	Res.	PTXFE	HCINT	HPRTINT	Res.		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	
RC_W1					R	R	R			RC_W1	RC_W1	R	R		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD

RC_W1		R	R	R	R	RC_W1	R	RC_W1	R

Bit	Name	R/W	Reset Value	Function
31	WKUPINT	RC_W1	0	WKUPINT：检测到唤醒/远程唤醒中断（Resume/remote wakeup detected interrupt)在设备模式下，当检测到 USB 总线上的唤醒信号即产生此中断；在主机模式下，在检测到 USB 总线上的一个远程唤醒信号时即产生此中断。注意：在设备模式和主机模式下都有效。
30	SRQINT	RC_W1	0	SRQINT：会话请求/检测到新会话中断（Session request/new session detected interrupt) 在主机模式下，当检测到一个来自设备的会话请求时即产生此中断； 在设备模式下，当 VBUS 在 B 类设备的有效电平范围内时即产生此中断。 注意：在设备模式和主机模式下都有效。
29	DISCINT	RC_W1	0	DISCINT: 检测到断开事件中断 (Disconnect detected interrupt)当检测到设备断开时即产生此中断。注意: 仅在主机模式下有效。
28	CIDSCHG	RC_W1	0	CIDSCHG: 连接上的 ID 线状态改变(Connector ID status change) 控制器在检测到连接上的 ID 线状态改变即设置此位。 注意：在主机模式和设备模式下都有效
27	保留	-	-	-
26	PTXFE	R	1	PTXFE：周期性发送 FIFO 空（Periodic TxFIFO empty)当周期性发送 FIFO 半空或者全空，并且周期性请求队列里至少能写入一个请求时，产生此中断。半空还是全空的状态由控制器 AHB 配置寄存器的周期性发送 FIFO 空级别位决定(OTG_FS_GAHBCFG 寄存器的 PTXFELVL 位)。注意：仅在主机模式下有效。
25	HCINT	R	0	HCINT：主机通道中断（Host channels interrupt)控制器设置此位，指示有一个未处理的主机通道事件(在主机模式下)。应用程序需要读主机所有通道中断寄存器(OTG_FS_HAINT 寄存器)来获得产生中断的通道号，然后读相应的主机通道 x 中断寄存器(OTG_FS_HCINTx 寄存器）获得中断的具体信息。应用程序需要通过清除 OTG_FS_HCINTx 寄存器的相应位来清除此位。注意: 仅在主机模式下有效。
24	HPRTINT	R	0	HPRTINT：主机端口中断（Host port interrupt)控制器设置此位指示 OTG_FS 控制器的某个端口状态发生改变。应用程序需要通过读主机端口控制和状态寄存器(OTG_FS_HPRT)来获得导致此中断的具体信息。应用程序必需通过清除主机端口控制和状态寄存器的相应位来清除此位。注意：仅在主机模式下有效。
23：22	保留	-	-	-
21	IPXFR	RC_W1	0	IPXFR: 未完成的周期性传输 (Incomplete periodic transfer)在主机模式下，控制器在帧结束时仍有属于当前帧的

				未完成的周期性传输时，产生此中断。注意：仅在主机模式下有效。
	INCOMPISOOUT			INCOMPISOOUT: 未完成的同步 OUT 传输 (Incomplete isochronous OUT transfer)在设备模式下，控制器在帧结束时仍有至少一个属于当前帧的未完成的同步 OUT 传输时，产生此中断。此中断随同本寄存器中的周期性帧结束中断(EOPF)产生。注意：仅在设备模式下有效。
20	IISOIXFR	RC_W1	0	IISOIXFR：未完成的同步 IN 传输（Incomplete isochronous IN transfer)在设备模式下，控制器在帧结束时仍有至少一个属于当前帧的未完成的同步 IN 传输时，产生此中断。此中断随同本寄存器中的周期性帧结束中断(EOPF)产生。注意：仅在设备模式下有效。
19	OEPINT	R	0	OEPINT：OUT 端点中断（OUT endpoint interrupt)在设备模式下，控制器设置此位指示有一个控制器未处理的 OUT 端点事件。应用程序需要读设备所有端点中断寄存器(OTG_FS_DAINT)来获得产生中断事件的端点号，然后读相应的设备 OUT 端点 x 中断寄存器(OTG_FS_DOEPINTx)来获得产生中断的具体信息。应用程序需要通过清除OTG_FS_DOEPINTx寄存器的相应位来清除此位。注意：仅在设备模式下有效。
18	IEPINT	R	0	IEPINT：IN 端点中断（IN endpoint interrupt)在设备模式下，控制器设置此位指示有一个控制器未处理的 IN 端点事件。应用程序需要读设备所有端点中断寄存器(OTG_FS_DAINT)来获得产生中断事件的端点号，然后读相应的设备 IN 端点 x 中断寄存器 (OTG_FS_DIEPINTx)来获得产生中断的具体信息。应用程序需要通过清除OTG_FS_DIEPINTx 寄存器的相应位来清除此位。注意:仅在设备模式下有效。
17：16	保留	-	-	-
15	EOPF	RC_W1	0	EOPF：周期性帧结束中断（End of periodic frame interrupt)此中断指示在当前帧中，在设备配置寄存器中设置的周期性帧间隔时间(OTG_FS_DCFG 寄存器的 PFIVL 位)到达。注意：仅在设备模式下有效。
14	ISOODRP	RC_W1	0	ISOODRP：同步 OUT 包丢失中断（Isochronous OUT packet dropped interrupt)当接收 FIFO 没有足够空间为一个同步 OUT 端点存放一个最大长度的数据包时，将导致控制器写同步 OUT 包失败，并产生此中断。注意：仅在设备模式下有效。
13	ENUMDNE	RC_W1	0	ENUMDNE: 枚举完成 (Enumerationdone)控制器设置此位指示已经获得速度枚举的信息。应用程序通过读设备状态寄存器(OTG_FS_DSTS)来获得进行枚举的速度信息。注意：仅在设备模式下有效。
12	USBRST	RC_W1	0	USBRST：USB 复位（USB reset)控制器在检测到一个 USB 复位信号时设置此位。注意：仅在设备模式下有效。
11	USBSUSP	RC_W1	0	USBSUSP：USB 挂起（USB suspend)控制器在检测到 USB 线上的挂起信号时设置此位。控制器会在数据线超

				过 3ms 没有活动的情况下进入挂起状态。注意：仅在设备模式下有效。
10	ESUSP	RC_W1	0	ESUSP: 早期挂起 (Early suspend)控制器在检测到 USB 总线有 3ms 处于空闲状态时设置此位。注意：仅在设备模式下有效。
9:8	保留	-	-	-
7	GONAKEFF	R	0	GONAKEFF：全局 OUT NAK 状态（Global OUT NAK effective)指示应用程序，设备控制寄存器的全局 OUT NAK 位(OTG_FS_DCTL 寄存器的 SGONAK 位)已生效。应用程序通过写设备控制寄存器的清除全局 OUT NAK 位(OTG_FS_DCTL 寄存器的 CGONAK 位)来清除此位。注意：仅在设备模式下有效。
6	GINAKEFF	R	0	GINAKEFF：全局非周期性 IN NAK 状态（Global IN non-periodic NAK effective)指示应用程序，设备控制寄存器的设置全局非周期性 IN NAK 位(OTG_FS_DCTL 寄存器的 SGINAK 位)已生效。也就是说，控制器已经相应了应用程序设置全局 IN NAK 位的操作。应用程序通过清除设备控制寄存器的清除全局非周期性 IN NAK 位(OTG_FS_DCTL 寄存器的 CGINAK 位)来清除此位。此中断并不意味着向 USB 总线发送了 NAK 握手信号，STALL 位的优先级高于 NAK 位。注意：仅在设备模式下有效。
5	NPTXFE	R	1	NPTXFE：非周期性发送 FIFO 空（Non-periodic TxFIFO empty)当非周期性发送 FIFO 全空或者半空，并且非周期性请求队列里至少能写入一个请求时，控制器会产生此中断。FIFO 半空或全空的状态取决于控制器 AHB 配置寄存器的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)。注意：仅在主机模式下有效。
4	RXFLVL	R	0	RXFLVL：接收 FIFO 非空（RxFIFO non-empty)指示接收 FIFO 中至少有个一个数据包等待处理。注意：在设备模式和主机模式下都有效。
3	SOF	RC_W1	0	SOF：帧首（Start of frame)在主机模式下，控制器设置此位指示有一个 SOF(全速设备)或保持有效(低速设备)信号已发向 USB 总线。应用程序必需向此位写'1'来清除此中断。在设备模式下，控制器设置此位表示在 USB 总线上检测到一个 SOF。应用程序需要读设备状态寄存器来获得帧编号。此中断仅当控制器处于全速状态时才会产生。注意：在设备模式和主机模式下都有效。
2	OTGINT	R	0	OTGINT: OTG 中断 (OTGinterrupt)控制器设置此位指示发生一个 OTG 协议事件。应用程序必需通过读 OTG 中断寄存器(OTG_FS_GOTGINT）来获得产生此中断的具体信息。应用程序必需通过清除 OTG_FS_GOTGINT 寄存器的相应位来清除此位。注意：在设备模式和主机模式下都有效
1	MMIS	RC_W1	0	MMIS：模式不匹配中断（Mode mismatch interrupt)控制器在应用程序试图执行以下操作时设置此位: 当控制器运

				行在设备模式下时，试图访问主机模式下的寄存器。当控制器运行在主机模式下时，试图访问设备模式下的寄存器。操作寄存器时，在 AHB 端会收到正确的响应，但控制器内部会忽视此类操作，并不对控制器的操作产生任何影响。注意：在主机模式和设备模式下都有效
0	CMOD	R	0	CMOD：当前模式（Current mode of operation)指示当前模式。0：设备模式；1：主机模式。注意：在设备模式和主机模式下都有效

32.4.7 OTG_FS 中断屏蔽寄存器(OTG_FS_GINTMSK)

Address:0x0018

Reset value:0x0000 0000

此寄存器与控制器中断寄存器配合使用，产生中断。当一个中断位被屏蔽，就不会产生相应的中断，然而控制器中断寄存器(OTG_FS_GINTSTS)的相应位仍然会被置起。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	Res.	PTXFEM	HCIM	PRTIM	Res.		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	
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
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	WUIM	RW	0	WUIM：检测到唤醒/远程唤醒中断屏蔽（Resume/remote wakeup detected interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：在设备模式和主机模式下都有效
30	SRQIM	RW	0	SRQIM：会话请求/检测到新会话中断屏蔽（Session request/new session detected interrupt mask) 0：屏蔽该中断；1：不屏蔽该中断。注意：在设备模式和主机模式下都有效。
29	DISCINT	RW	0	DISCINT：检测到断开事件中断屏蔽（Disconnect detected interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效。
28	CIDSCHGM	RW	0	CIDSCHGM：连接上的 ID 线状态改变屏蔽（Connector ID status change mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在主机模式和设备模式下都有效
27	保留	-	-	-
26	PTXFEM	RW	0	PTXFEM: 周期性发送 FIFO 空屏蔽 (Periodic TxFIFO empty mask) 0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效
25	HCIM	RW	0	HCIM：主机通道中断屏蔽（Host channels interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效。
24	PRTIM	RW	0	PRTIM：主机端口中断屏蔽（Host port interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效

23:22	保留	-	-	-
21	IPXFRM	RW	0	IPXFRM: 未完成的周期性传输中断屏蔽 (Incomplete periodic transfer mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效。
	IISOOXFRM			IISOOXFRM: 未完成的同步 OUT 传输中断屏蔽 (Incomplete isochronous OUT transfer mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
20	IISOIXFRM	RW	0	IISOIXFRM：未完成的同步 IN 传输中断屏蔽（Incomplete isochronous IN transfer mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
19	OEPINT	RW	0	OEPINT：OUT 端点中断屏蔽（OUT endpoints interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效。
18	IEPINT	RW	0	IEPINT：IN 端点中断屏蔽（IN endpoints interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
17:16	保留	-	-	-
15	EOPFM	RW	0	EOPFM：周期性帧结束中断屏蔽（End of periodic frame interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
14	ISOODRPM	RW	0	ISOODRPM: 同步 OUT 包丢失中断屏蔽 (Isochronous OUT packet dropped interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
13	ENUMDNEM	RW	0	ENUMDNEM：枚举完成中断屏蔽（Enumeration done mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
12	USBRST	RW	0	USBRST：USB 复位中断屏蔽（USB reset mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
11	USBSUSPM	RW	0	USBSUSPM：USB 挂起中断屏蔽（USB suspend mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效。
10	ESUSPM	RW	0	ESUSPM：早期挂起中断屏蔽（Early suspend mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
9:8	保留	-	-	-
7	GONAKEFFM	RW	0	GONAKEFFM：全局 OUT NAK 状态中断屏蔽（Global OUT NAK effective mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效
6	GINAKEFFM	RW	0	GINAKEFFM：全局非周期性 IN NAK 状态中断屏蔽（Global non-periodic IN NAK effective mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在设备模式下有效。
5	NPTXFEM	RW	0	NPTXFEM: 非周期性发送 FIFO 空中断屏蔽（Non-periodic TxFIFO empty mask）0：屏蔽该中断；1：不屏蔽该中断。注意：仅在主机模式下有效。
4	RXFLVLM	RW	0	RXFLVLM：接收 FIFO 非空中断屏蔽（Receive FIFO non-empty mask）0：屏蔽该中断；1：不屏蔽该中断。注意：在设备模式和主机模式下都有效
3	SOFM	RW	0	SOFM：帧首中断屏蔽（Start of frame mask）0：屏蔽该中断；1：不屏蔽该中断。注意：在设备模式和主机模式下都有效。
2	OTGINT	RW	0	OTGINT：OTG 中断屏蔽（OTG interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：在设备模式和主机模式下都有效

1	MMISM	RW	0	MMISM：模式不匹配中断屏蔽（Mode mismatch interrupt mask）0：屏蔽该中断；1：不屏蔽该中断。注意：在主机模式和设备模式下都有效。
0	保留	-	-	-

32.4.8 OTG_FS 接收状态调试读/OTG 状态读和 POP 寄存器(OTG_FS_GRXSTSR /OTG_FS_GRXSTSP)

读地址偏移:0x001C

弹出地址偏移：0x0020

Reset value:0x0000 0000

对接收状态调试读寄存器的读操作，将返回接收 FIFO 中顶部的数据。对接收状态调试读寄存器和 POP 寄存器的读操作，将会把接收 FIFO 中顶部的数据项顶出。在主机模式和设备模式下，对于接收状态数据的解释并不相同。如果接收 FIFO 为空，控制器将忽略对接收状态的读/POP 操作，并返回 0x0000 0000。应用程序只有在控制器中断寄存器的接收 FIFO 非空位(OTG_FS_GINTSTS 寄存器的 RXFLVL 位)为'1'时才能 POP 出接收状态 FIFO。

主机模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.											PKTSTS				DPID
											R				R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DPID	BCNT[10:0]											CHNUM[3:0]			
R	R											R			

Bit	Name	R/W	Reset Value	Function
31:21	保留	-	-	-
20:17	PKTSTS [3:0]	R	4'h0	PKTSTS：包状态（Packet status)指示接收到的数据包的状态 4'b0010：接收到 IN 数据包； 4'b0011：IN 传输完成(触发中断)； 4'b0101：数据翻转位出错(触发中断)； 4'b0111：通道中止(触发中断)； 其他：保留。
16:15	DPID[1:0]	R	2'h0	DPID：数据 PID（Data PID)指示接收到的数据包的数据 PID 2'b00：DATA0； 2'b10：DATA1； 2'b01：保留； 2'b11：保留。
14:4	BCNT[10:0]	R	11'h0	BCNT：字节数（Byte count)指示接收到的数据包的字节数
3:0	CHNUM [3:0]	R	4'h0	CHNUM：通道号（Channel number)指示当前收到的数据包属于哪个通道

设备模式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.							FRMNUM[3:0]				PKTSTS [3:0]				DPID
							R				R				R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DPID	BCNT[10:0]											EPNUM[3:0]			
R	R											R			

Bit	Name	R/W	Reset Value	Function
31:25	保留	-	-	-

24:21	FRMNUM[3:0]	R	4’h0	FRMNUM：帧编号（Frame number)此 4 位是从 USB 接收到的数据包所属的帧号的末 4 位，仅在同步 OUT 传输时有效
20:17	PKTSTS [3:0]	R	4’h0	PKTSTS: 包状态 (Packet status)指示接收到的数据包的状态 4’b0001：全局的 OUT NAK(触发中断)； 4’b0010：接收到 OUT 数据包； 4’b0011：OUT 传输完成(触发中断)； 4’b0100：SETUP 传输完成(触发中断)； 4’b0110：接收到 SETUP 数据包； 其他：保留。
16:15	DPID[1:0]	R	2’h0	DPID：数据 PID（Data PID)指示接收到 OUT 数据包的 PID 2’b00：DATA0； 2’b10：DATA1； 2’b01：保留； 2’b11：保留。
14:4	BCNT[10:0]	R	11’h0	BCNT：字节数（Byte count)指示接收到的数据包的字节数
3:0	EPNUM[3:0]	R	4’h0	EPNUM：端点号指示当前接收到的数据包所属的的端点号

32.4.9 OTG_FS 接收 FIFO 长度寄存器(OTG_FS_GRXFSIZ)

Address:0x0024

Reset value:0x0000 0100

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

Bit	Name	R/W	Reset Value	Function
31：11	保留	-	-	-
10：0	RXFD[10:0]	RW	12’h100	RXFD: 接收 FIFO 的深度 (RxFIFO depth)这个数值的单位是 32 位的字 最小值为 16 最大值为 256 上电复位的值是接收 FIFO 的最大深度值。

32.4.10 OTG_FS 非周期性 TX FIFO 长度寄存器(OTG_FS_GNPTXFSIZ)

Address:0x0028

Reset value:0x0100 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					NPTXFD/TX0FD										
					RW										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res					NPTXFSA/TX0FSA										
					RW										

主机模式

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	-
26:16	NPTXFD	RW	11’h100	NPTXFD: 非周期性发送 FIFO 深度 (Non-periodic TxFIFO depth)这个数值的单位是 32 位的字最小值是 16 最大值是 256
15:11	保留	-	-	-
10:0	NPTXFSA	RW	11’h100	NPTXFSA: 非周期性接收 FIFO 在 RAM 里的起始地址 (Non-periodic transmit RAM start address) 这些位的值表示非周期性接收 FIFO 在 RAM 里的起始地址

设备模式

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	-
26:16	TX0FD	RW	11’h100	TX0FD：EP0 TxFIFO 深度（Endpoint 0 TxFIFO depth)这个数值的单位是 32 位的字最小值是 16 最大值是 256
15:11	保留	-	-	-
10:0	TX0FSA	RW	11’h100	TX0FSA: EP0 传输 RAM 的起始地址 (Non-periodic transmit RAM start address) 这些位的值表示 EP0 传输 FIFO RAM 的起始地址

32.4.11 OTG_FS 非周期性 TX FIFO/请求队列状态寄存器(OTG_FS_GNPTXSTS)

Address:0x002C

Reset value:0x0008 0100

注意：在设备模式下，此寄存器无效。

此寄存器为只读寄存器，储存非周期性发送 FIFO 和非周期性传输请求队列的剩余空间信息

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

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:24	NPTXQTOP [6:0]	R	7’h0	NPTXQTOP: 非周期性传输请求队列的顶部 (Top of the non-periodic transmit request queue)正在由 MAC 模块处理的非周期性发送请求： 位[30:27]：通道/端点号； 位[26:25]： 2’b00：IN/OUT 命令； 2’b01：0 长度的传输包(设备 IN/主机 OUT)； 2’b11：通道中止命令； 位 24：结束(选中通道/端点的最后一个请求)。

23:16	NPTQXSAV [7:0]	R	8'h8	NPTQXSAV：非周期性传输请求队列的剩余空间（Non-periodic transmit request queue space available) 指示非周期性传输请求队列的剩余空间。在主机模式下，此队列既保存 IN 的传输请求又保存 OUT 的传输请求，在设备模式下，仅保存 IN 的传输请求。 8'h0：非周期性传输请求队列满； 8'h1：剩余 1 个请求空间； 8'h2：剩余 2 个请求空间； 8'hn：剩余 n 个请求空间(0 ≤ n ≤ 8)； 其他：保留。
15:0	NPTXFSAV [15:0]	R	16'h100	NPTXFSAV：非周期性发送 FIFO 的剩余空间（Non-periodic TxFIFO space available)指示非周期性发送 FIFO 的剩余空间，此值为 32 位。 16'h0：非周期性发送 FIFO 满； 16'h1：剩余 1 个字的空间； 16'h2：剩余 2 个字的空间； 16'hn：剩余 n 个字的空间(0 ≤ n ≤ 256）； 其他：保留。

32.4.12 OTG_FS 通用控制器配置寄存器(OTG_FS_GCCFG)

Address:0x0038

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21	VBUSIG	RW	0	VBUS 忽略(VBUS ignore) 当该控制位被置位，USBFS 并不监测 VBUS 引脚电压，并且认为在主机和设备模式下，VBUS 电压一直有效，然后可释放 VBUS 引脚作为其他用途。 0：VBUS 不被忽略 1：VBUS 被忽略，并认为 VBUS 电压一直有效
20	SOFOEN	RW	0	SOFOEN：SOF 输出使能（SOF output enable） 0：不输出 SOF 脉冲； 1：输出 SOF 脉冲到引脚上。
19	VBUSBCEN	RW	0	VBUSBSEN：VBUS 对于 B 类有效电平的监控使能（Enable the VBUS compare "B" device） 0：VBUS 不监控 B 类有效电平； 1：VBUS 监控 B 类有效电平。
18	VBUSACEN	RW	0	VBUSASEN：VBUS 对于 A 类有效电平的监控使能（Enable the VBUS compare "A" device） 0：VBUS 不监控 A 类有效电平； 1：VBUS 监控 A 类有效电平。

17	保留	-	-	-
16	PWRON	RW	0	PWRON: 上电 (Power on)用于在发送和接收时激活收发器 0：使能掉电； 1：禁止掉电(收发器激活)
15:0	保留	-	-	-

32.4.13 OTG_FS 控制器 ID 寄存器(OTG_FS_CID)

Address:0x003C

Reset value:0x0000 1000

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

Bit	Name	R/W	Reset Value	Function
31：0	PRODUCT_ID	RW	32’h1000	PRODUCT_ID：产品 ID（Product ID field)应用程序可以编写此 ID 位。

32.4.14 OTG_FS 主机周期性发送 FIFO 长度寄存器(OTG_FS_HPTXFSIZ)

Address:0x0100

Reset value:0x0100 0200

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

Bit	Name	R/W	Reset Value	Function
31:16	PTXFSIZ [15:0]	R/RW	16’h100	PTXFSIZ：主机周期性发送 FIFO 深度（Host periodic TxFIFO depth)此值为 32 位的字最小值为 16 最大值为 512
15:0	PTXSA [15:0]	R/RW	16’h200	PTXSA：主机周期性发送 FIFO 起始地址（Host periodic TxFIFO start address)此寄存器的复位值是最大接收 FIFO 深度和最大非周期发送 FIFO 深度之和。

32.4.15 OTG_FS 设备 IN 端点发送 FIFO 长度寄存器(OTG_FS_DIEPTXFx)(其中 x 是 FIFO 的编号，x=1…3)

Address:0x0104 +（FIFO_number -1)*0x04

Reset value:0x0100 0200

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

RW

Bit	Name	R/W	Reset Value	Function
31:16	INEPTXFD [15:0]	RW	16'h100	INEPTXFD：IN 端点发送 FIFO 的深度（IN endpoint TxFIFO depth) 此值为 32 位的字 最小值为 16 最大值为 512 复位值是最大可能的 IN 端点发送 FIFO 的深度
15:0	INEPTXSA [15:0]	RW	16'h200	INEPTXSA：IN 端点发送 FIFO 在 RAM 中的起始地址（IN endpoint FIFOx transmit RAM start-address) 此值为 IN 端点发送 FIFO 在 RAM 中的起始地址。

32.5 主机模式下的寄存器

如果没有特殊说明，寄存器中的值都以二进制形式表达。

主机模式下的寄存器仅在主机模式下生效，不能在设备模式下访问，非法访问的结果是未定义的。主机模式下的寄存器如下：

32.5.1 OTG_FS 主机模式配置寄存器(OTG_FS_HCFG)

Address:0x0400

Reset value:0x0000 0200

此寄存器在上电后配置控制器的操作，不要在初始化后再修改此寄存器。

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

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	FSLSS	R	0	FSLSS：支持全速和低速设备（FS- and LS-only support) 应用程序通过此位来配置控制器对设备进行枚举的速度。应用程序可以通过此位使控制器用全速 模式来枚举一个支持高速通信的设备。在初始化后不要修改此值。 1：仅支持全速/低速设备，即使插入的设备支持高速通信(只读)
1:0	FSLSPCS	RW	2'h0	FSLSPCS：全速/低速 PHY 时钟选择（FS/LS PHY clock select) 当控制器处于全速主机模式时： 2'b01：PHY 时钟运行在 48MHz； 其他值：保留。 当控制器处于低速主机模式时： 2'b00：保留； 2'b01：PHY 时钟运行在 48MHz； 2'b10：PHY 时钟运行在 6MHz，根据 USB1.1 全速模式定义，当 UTMIFS PHY 低功耗模式选中

				时，如果 PHY 支持 6MHz 时钟，就在低速模式下使用 6MHz 时钟。 用户一旦在低速模式 下选中了 6MHz 时钟，需要执行一个软件复位操作； 2’b11：保留。

32.5.2 OTG_FS 主机帧间隔寄存器(OTG_FS_HFIR)

Address:0x0404

Reset value:0x0000 EA60

此寄存器为 OTG_FS 控制器在枚举时选中的速度设置帧间隔时间。

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

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15：0	FRIVL [15:0]	RW	16’hEA60	FRIVL：帧间隔（Frame interval) 应用程序通过此位来配置两个连续的 SOF(全速)或保持有效(低速)之间的时间间隔。此寄存器用 PHY 时钟个数来表示帧间隔。应用程序只有在设置了主机端口控制和状态寄存器的端口使能位(OTG_FS_HPRT 寄存器的 PENA 位)之后才能设置此寄存器。如果没有设置此寄存器，控制器将按照主机配置寄存器的全速/低速 PHY 时钟选择位(OTG_FS_HCFG 寄存器的 FSLSPCS 位)定义的 PHY 时钟来计算值。不要在初始化后再修改此寄存器。 1ms ×（全速/低速的 PHY 时钟频率)

32.5.3 OTG_FS 主机帧号/帧时间剩余寄存器(OTG_FS_HFNUM)

Address:0x0408

Reset value:0x0000 03FFF

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

Bit	Name	R/W	Reset Value	Function
31：16	FTREM [15:0]	R	16’h0	FTREM：帧时间剩余（Frame time remaining) 指示当前帧还剩余多少时间，用 PHY 时钟数表达。每个 PHY 时钟，此值都会自减 1。当此值自减为 0，定义在帧间隔寄存器中的数值将自动载入此寄存器，并向 USB 总线发送一个新的 SOF。
15:0	FRNUM [15:0]	R	16’h3FFF	FRNUM：帧号（Frame number) 每向 USB 总线发送一个 SOF 信号，此域自动加 1。达到 0x3FFF 时则自动归零，重新开始累加。

32.5.4 主 OTG_FS 主机周期性发送 FIFO/请求队列寄存器(OTG_FS_HPTXSTS)

Address:0x0410

Reset value:0x0008 0100

此寄存器为只读寄存器，保存周期性发送 FIFO 和请求队列的剩余空间信息。

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

Bit	Name	R/W	Reset Value	Function
31:23	PTXQTOP [8:0]	R	9'h0	PTXQTOP: 周期性传输请求队列顶部 (Top of the periodic transmit request queue) 指示 MAC 模块正在处理的周期性发送请求项。 此寄存器仅用于模块调试。 位 31：奇数/偶数帧 0：偶数帧发送; 1：奇数帧发送。 位[30:27]：通道/端点号; 位[26:25]：类型 2'b00：IN/OUT; 2'b01：零长度数据包; 2'b11：中止通道命令。 位 24：结束(选中通道/端点的最后一个请求) 位 23：终止(所选通道/端点的最后一个条目)
22:16	PTXQSAV [6:0]	R	7'h8	PTXQSAV：周期性传输请求队列的剩余空间（Periodic transmit request queue space available) 指示周期性传输请求队列的剩余空间，此请求队列包括 IN 和 OUT 的请求。 7'h0：周期性传输请求队列满; 7'h1：剩余 1 个请求位置; 7'h2：剩余 2 个请求位置; 7'hn：剩余 n 个请求位置(0≤ n ≤8); 其他：保留
15:0	PTXFSAVL [15:0]	R	16'h100	PTXFSAVL: 周期性发送 FIFO 剩余空间 (Periodic transmit data FIFO space available) 指示周期性发送 FIFO 的剩余空间 此值为 32 位的字。 16'h0：周期性发送 FIFO 满; 16'h1：剩余 1 个字; 16'h2：剩余 2 个字; 16'hn：剩余 n 个字(0≤ n ≤256); 16'h100：剩余 256 个字; 其他：保留。

32.5.5 OTG_FS 主机所有通道中断寄存器(OTG_FS_HAINT)

Address:0x0414

Reset value:0x0000 0000

当某个通道产生了标志性的事件,主机所有通道中断会通过控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)打断应用程序。每个通道都有相对应的通道中断位，一共有 8 个控制位。应用程序通过设置和清除相应的主机通道 x 中断寄存器的相应位来设置和清除此位。

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

Bit	Name	R/W	Reset Value	Function
31：8	保留	-	-	-
7:0	HAINT [7:0]	R	8’h0	HAINT：通道中断（Channel interrupts)每一位表示一个通道：位 0 代表通道 0，位 7 表示通道 7 0：未产生主机通道中断 1：产生主机通道中断

32.5.6 OTG_FS 主机所有通道中断屏蔽寄存器(OTG_FS_HAINTMSK)

Address:0x0418

Reset value:0x0000 0000

主机所有通道中断屏蔽寄存器与主机所有通道中断寄存器配置使用，用于在产生事件时打断应

用程序。每个通道都有一个相对应的中断屏蔽位，共有 8 位。

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

Bit	Name	R/W	Reset Value	Function
31：8	保留	-	-	-
7:0	HAINTM	RW	8’h0	HAINTM：通道中断屏蔽（Channel interrupt mask) 0：禁用通道 n 中断 1：使能通道 n 中断 每一位表示一个通道：位 0 代表通道 0，位 7 代表通道 7

32.5.7 OTG_FS 主机端口控制和状态寄存器(OTG_FS_HPRT)

Address:0x0440

Reset value:0x0000 0000

此寄存器仅在主机模式下有效。一个 OTG 主机控制器仅支持一个端口。

此寄存器保存与主机端口相关的信息，包括每个端口的 USB 复位、使能、挂起、唤醒、连接状

态和测试模式信息等。标明 rc_w1 的位可以通过主机中断寄存器的主机端口
中断位(OTG_FS_GINTSTS 寄存器的 HPRTINT 位)来触发中断，打断应用程序。在端口中断服
务程序中，应用程序必需读此寄存器并清除导致中断的位。对于标明 rc_w1 的位，应用程序需要
通过写'1'来清除中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.													PSPD [1:0]		PTCTL
													R		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[3:0]			PPWR	PLSTS [1:0]		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
			RW	R			RW	RS	RW	RC_W1	R	RC_W1	RC_W1	RC_W1	R

Bit	Name	R/W	Reset Value	Function
31:19	保留	-	-	-
18:17	PSPD	R	2'h0	PSPD：端口速度（Port speed) 指示连上端口的设备速度 2'b01：全速设备； 2'b10：低速设备； 其他：保留。
16:13	PTCTL	RW	4'h0	PTCTL：端口测试控制（Port test control) 应用程序通过写非零值到此位来进入测试模式，在端口会出现相应的信号。 4'b0000：非测试模式； 4'b0001：Test_J 模式； 4'b0010：Test_K 模式； 4'b0011：Test_SE0_NAK 模式； 4'b0100：Test_Packet 模式； 4'b0101：Test_Force_Enable； 其他：保留
12	PPWR	RW	0	PPWR：端口供电（Port power) 应用程序通过设置此位来向端口供电，控制器在发生电流溢出事件时清除此位。 0：不供电； 1：供电。
11:10	PLSTS	R	2'b0	PLSTS：端口状态（Port line status) 指示 USB 数据线的当前逻辑状态 位 10：OTG_FS_FS_DP 线的逻辑状态； 位 11：OTG_FS_FS_DM 线的逻辑状态; 其他：保留。
9	保留	-	-	-
8	PRST	RW	0	PRST：端口复位（Port reset) 当应用程序设置此位时，会在端口上产生一个复位序列。应用程序需要控制复位的时间，并在复 位完成后清除此位。 0：端口不复位； 1：端口复位。

				应用程序需要保持此位的’1’状态至少 10ms 以便启动端口的复位序列。应用程序也可以在清除此 位前增加 10ms 的’1’状态，USB 规范没有对复位信号的最长持续时间做出定义。
7	PSUSP	RS	0	PSUSP：端口挂起（Port suspend) 应用程序设置此位，使端口进入挂起状态。在这种状态下控制器仅停止发送 SOF 信号。应用程序 需要通过设置端口时钟停止位来停止 PHY 时钟。 对此位的读操作将返回当前端口的挂起状态，当应用程序设置此寄存器的端口复位位或端口唤醒 位，或控制器检测到一个远程唤醒信号，或控制器中断寄存器的唤醒/远程唤醒检测中断位或端口 检测中断位(OTG_FS_GINTSTS寄存器的WKUINT位或DISCINT位)被设置，控制器将清除此 位。 0：端口不处于挂起模式； 1：端口处于挂起模式。
6	PRES	RW	0	PRES：端口唤醒（Port resume) 应用程序设置此位驱动端口发出唤醒信号。控制器将输出驱动唤醒信号直到应用程序清除此位。 当控制器检测到 USB 远程唤醒序列，按照控制器中断寄存器的端口唤醒/远程唤醒检测中断位的指 示，控制器会自动输出唤醒序列而不需要应用程序干预。如果控制器检测到设备断开，会自动清 除此位。对此位的读操作将返回控制器是否正在输出唤醒信号的信息。 0：无唤醒； 1：输出唤醒信号。
5	POCCHNG	RC_W1	0	POCCHNG：端口过流状态改变（Port overcurrent change) 控制器在此寄存器的端口过流位(位 4)发生变化时设置此位
4	POCA	R	0	POCA：端口过流位（Port overcurrent active) 指示端口是否发生了过流。 0：没有过流； 1：过流
3	PENCHNG	RC_W1	0	PENCHNG: 端口使能/禁止状态改变(Port enable/disable change) 控制器在本寄存器的端口使能位(位 2)发生变化时设置此位
2	PENA	RC_W1	0	PENA：端口使能（Port enable) 端口只有在复位序列后才能被控制器使能，在发生过流、设备断开获控制器清除此位时禁止端 口。应该程序不能通过寄存器写操作设置此位。此位不会触发中断。 0：端口禁止； 1：端口使能
1	PCDET	RC_W1	0	PCDET：端口连接检测（Port connect detected) 当控制器检测到一个设备连接到端口时，会触发控制器中断寄存器的主机端口中断位

				(OTG_FS_GINTSTS 寄存器的 HPRTINT 位)来产生中断，并设置此位。应用程序必需通过写'1'来 清除中断。
0	PCSTS	R	0	PCSTS：端口连接状态（Port connect status) 0：设备没有连接到该端口 1：设备连接到该端口

32.5.8 OTG_FS 主机通道 x 特性寄存器(OTG_FS_HCCHARx)(此处 x 代码通道号，x = 0...7)

Address:0x0500 +（通道号*0x20)

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODDFRM	DAD [6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
RS	RS	RW	RW							RW		RW		RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPDIR	EPNUM[3:0]				MPSIZ [10:0]										
RW	RW				RW										

Bit	Name	R/W	Reset Value	Function
31	CHENA	RS	0	CHENA：通道使能（Channel enable) 此位由应用程序设置，由 FS_OTG 主机控制器清除。 0：通道禁止； 1：通道使能
30	CHDIS	RS	0	CHDIS：通道禁止（Channel disable) 应用程序通过设置此位，可以立即停止该通道上的数据收/发，即使该通道数据的传输还没有完 成。应用程序只有在等到通道禁止中断产生时，才能认为该通道已被禁止。
29	ODDFRM	RW	0	ODDFRM：奇数帧（Odd frame) 应用程序通过设置/清除此位来告知 OTG 主机控制器应该在奇数/偶数帧时进行传输。此位仅对周期性传输(同步或中断)有效。 0：偶数帧； 1：奇数帧。
28:22	DAD	RW	7'h0	DAD：设备地址（Device address)应用程序通过此地址选择需要的设备。
21:20	MCNT	RW	2'h0	MCNT：设备地址（Multicount) 此域指示主机在这个周期性端点上每帧必须执行的传输数目。非周期性传输不使用这个参数。 2'b00：保留。设置这个数值将产生未定义的结果。 2'b01：1 次传输。 2'b10：在这个端点上，每帧需要产生 2 次传输。 2'b11：在这个端点上，每帧需要产生 3 次传输。 注：此域的数值至少应该设置为'01'
19:18	EPTYP	RW	2'b0	EPTYP：端点类型 指示选中的传输类型 2'b00：控制传输 2'b01：同步传输

				2'b10：块传输 2'b11：中断传输
17	LSDEV	RW	0	LSDEV：低速设备（Low speed device) 应用程序设置此位指示与之通信的设备是低速设备。
16	保留	-	-	-
15	EPDIR	RW	0	EPDIR：端点方向（Endpoint direction) 指示传输是 IN 还是 OUT 方向。 0：OUT 1：IN
14:11	EPNUM	RW	4'h0	EPNUM：端点号（Endpoint number) 指示与之通信的端点号
10:0	MPSIZ	RW	11'h0	MPSIZ：最大包长度（Maximum packet size) 指示选中端点的最大包长度

32.5.9 OTG_FS 主机通道 x 中断寄存器(OTG_FS_HCINTx)(其中 x 代表通道号，x=0...7)

Address:0x0508+(通道号*0x20)

Reset value:0x0000 0000

此寄存器指示通道与 USB 和 AHB 相关时间的状态。当控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)置起时，应用程序需要先读主机所有通道中断寄存器(OTG_FS_HAINT)，获得发生主机通道中断的通道号，再读取相应通道的中断寄存器，获得中断的详细信息。应用程序通过清除此寄存器的相应位，清除 OTG_FS_HAINT 寄存器和 OTG_FS_GINTSTS 寄存器的相应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
					RC_W1	RC_W1	RC_W1	RC_W1		RC_W1	RC_W1	RC_W1		RC_W1	RC_W1

Bit	Name	R/W	Reset Value	Function
31：11	保留	-	-	-
10	DTERR	RC_W1	0	DTERR：数据 PID 错误（Data toggle error)
9	FRMOR	RC_W1	0	FRMOR：帧溢出（Frame overrun)
8	BBERR	RC_W1	0	BBERR：串扰错误（Babble error)
7	TXERR	RC_W1	0	TXERR：传输错误（Transaction error) 指示发生了以下错误： CRC 校验失败 超时 位填充错误 EOP 失败
6	保留	-	-	-
5	ACK	RC_W1	0	ACK：ACK 已收到/已发送中断（ACK response received/transmitted interrupt)
4	NAK	RC_W1	0	NAK：NAK 已收到中断（NAK response received interrupt)
3	STALL	RC_W1	0	STALL：STALL 已收到中断 (STALL response received interrupt)

2	保留	-	-	-
1	CHH	RC_W1	0	CHH：通道中止（Channel halted) 指示由于 USB 传输错误，或者应用程序中止请求导致的传输异常结束。
0	XFRC	RC_W1	0	XFRC：传输完成（Transfer completed) 传输正常完成，没有出错。

32.5.10 OTG_FS 主机通道 x 中断屏蔽寄存器(OTG_FS_HCINTMSKx)(其中 x 为通道号，x=0...7)

Address:0x050C+(通道号*0x20)

Reset value:0x0000 0000

该寄存器包含 USBFS_HCHxINTF 寄存器内中断标志位的中断使能位。如果该寄存器的某位被软件置位，USBFS_HCHxINTF 寄存器内的相应位能够触发一个通道中断。该寄存器内的位可由软件置位和清除。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					DTERRM	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALML	Res.	CHHM	XFRCM
					RW	RW	RW	RW		RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31：11	保留	-	-	-
10	DTERRM	RW	0	DTERRM：数据 PID 错误中断屏蔽（Data toggle error mask) 0：屏蔽中断; 1：不屏蔽中断。
9	FRMORM	RW	0	FRMORM：帧溢出中断屏蔽（Frame overrun mask) 0：屏蔽中断; 1：不屏蔽中断
8	BBERRM	RW	0	BBERRM：串扰错误中断屏蔽（Babble error mask) 0：屏蔽中断; 1：不屏蔽中断
7	TXERRM	RW	0	TXERRM：传输错误中断屏蔽（Transaction error mask) 0：屏蔽中断; 1：不屏蔽中断。
6	保留	-	-	-
5	ACKM	RW	0	ACKM：ACK 已收到/已发送中断屏蔽（ACK response received/transmitted interrupt mask) 0：屏蔽中断; 1：不屏蔽中断
4	NAKM	RW	0	NAKM：NAK 已收到中断屏蔽（NAK response received interrupt mask) 0：屏蔽中断; 1：不屏蔽中断
3	STALML	RW	0	STALML：STALL 已收到中断屏蔽（STALL response received interrupt mask) 0：屏蔽中断; 1：不屏蔽中断
2	保留	-	-	-

1	CHHM	RW	0	CHHM：通道中止中断屏蔽（Channel halted mask) 0：屏蔽中断； 1：不屏蔽中断
0	XFRCM	RW	0	XFRCM：传输完成中断屏蔽（Transfer completed mask) 0：屏蔽中断； 1：不屏蔽中断

32.5.11 OTG_FS 主机通道 x 传输长度寄存器(OTG_FS_HCTSIZx)(其中 x 为通道号，x=0...7)

Address:0x0510+(通道号*0x20)

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:29	DPID[1:0]	RW	2’h0	DPID：数据 PID（Data PID) 应用程序通过此位告知控制器首个传输所使用的 PID 类型。控制器将会自动控制后续传输的 PID 类型。 2’b00：DATA0； 2’b01：保留； 2’b10：DATA1； 2’b11：SETUP(控制)
28:19	PKTCNT [9:0]	RW	10’h0	PKTCNT：包数目（Packet count) 应用程序通过此位告知控制器期望收到(IN)的包数目或期望发送(OUT)的包数目。 控制器在每次成功地发送或接收 OUT/IN 包之后，将自动地递减这个参数，一旦该域为 0，应用程序将收到中断，指示传输正常结束。
18:0	XFRSIZ [18:0]	RW	19’h0	XFRSIZ：传输长度（Transfer size) 对于 OUT 传输，这些位指示主机在传输期间发送的数据字节数目。 对于 IN 传输，这些位指示应用程序预留的缓冲区长度。对于 IN 传输(周期性和非周期性)，应用程序需要按照最大数据包的整数倍来预定义这个参数。

32.6 设备模式下的寄存器

32.6.1 OTG_FS 设备配置寄存器(OTG_FS_DCFG)

Address:0x0800

Reset value:0x0820 0000

在复位、或特殊控制命令、或枚举后，此寄存器用于配置控制器在设备模式下的操作特性。在

初始化后，不要再修改此寄存器。

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

Bit	Name	R/W	Reset Value	Function
31：13	保留	-	-	-
12:11	PFIVL [1:0]	RW	2’h0	PFIVL：周期性帧间隔（Periodic frame interval) 指示产生周期性帧结束中断的时间占整个帧的百分比。应用程序可以通过此中断判断一个帧内的同步传输是否都已传输完成。 2’h00：80%的帧时间； 2’h01：85%的帧时间； 2’h10：90%的帧时间； 2’h11：95%的帧时间。
10:4	DAD [6:0]	RW	7’h0	DAD：设备地址（Device address) 应用程序在收到 SetAddress 的控制命令后，按照参数填充此位
3	保留	-	-	-
2	NZLSOHSK	RW	0	NZLSOHSK：非零长度的状态 OUT 握手信号（Non-zero-length status OUT handshake) 在控制传输的状态阶段，如果收到了一个非零长度的数据包，应用程序可以通过此位选择发送一个握手信号。 1: 向非零长度的状态 OUT 传输发送 STALL 握手，并且不向应用程序传送收到的 OUT 包。 0：根据设备端口控制寄存器的 NAK 位和 STALL 位状态，选择发送握手信号，并将收到的 OUT 包传送给应用程序(零长度和非零长度的)。
1:0	DSPD [1:0]	RW	2’h0	DSPD：设备速度（Device speed) 指示应用程序需要控制器进行枚举操作时的速度，或者是应用程序能支持的最大速度。然而，实际的总线速度只有在整个序列完成后，才能根据所连接的 USB 主机的速度决定。 2’b00：保留； 2’b01：保留； 2’b10：保留； 2’b11：全速(USB 1.1 收发器，时钟为 48 MHz)

32.6.2 OTG_FS 设备控制寄存器(OTG_FS_DCTL)

Address:0x0804

Reset value:0x0000 0002

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL[2:0]			GONSTS	GINSTS	SDIS	RWUSIG
				RW	W	W	W	W	RW			R	R	RW	RW

Bit	Name	R/W	Reset Value	Function
31：12	保留	-	-	-
11	POPRGDNE	RW	0	POPRGDNE：上电配置结束（Power-on programming done) 应用程序通过此位指示从调电状态唤醒后，对寄存器的配置已经完成。
10	CGONAK	W	0	CGONAK：清除全局 OUT NAK（Clear global OUT NAK) 对此位的写操作将清除全局 OUT NAK 状态。
9	SGONAK	W	0	SGONAK：设置全局 OUT NAK（Set global OUT NAK) 对此位的写操作将设置全局 OUT NAK 状态。 应用程序通过此位告知控制器对所有的 OUT 端点都发送 NAK 信号。 程序在设置此位前，必需确认控制器中断寄存器的全局 OUT NAK 有效位(OTG-FS_GINTSTS 寄存器的 GONAKEFF 位)已被清除。
8	CGINAK	W	0	CGINAK：清除全局 IN NAK（(Clear global IN NAK)) 写此位将清除全局 IN NAK 状态
7	SGINAK	W	0	SGINAK：设置全局 IN NAK（Set global IN NAK) 对此位的写操作将设置全局非周期性 IN 的 NAK 状态。程序通过此位告知控制器对所有非周期性 IN 端点都发送 NAK 握手信号。 程序在设置此位前，必需确认控制器中断寄存器的全局 IN NAK 有效位(OTG_FS_GINTSTS 寄存器的 GINAKEFF 位)已被清除
6:4	TCTL	RW	3’h0	TCTL：测试控制（Test control) 3’b000：测试模式禁止； 3’b001：Test_J 模式； 3’b010：Test_K 模式； 3’b011：Test_SE0_NAK 模式； 3’b100：Test_Packet 模式； 3’b101：Test_Force_Enable； 其他：保留。
3	GONSTS	R	0	GONSTS：全局 OUT NAK 状态（Global OUT NAK status) 0：根据 FIFO 的状态和 NAK 位及 STALL 位的状态，发送握手信号。 1：不管存储区是否为空，都不写入数据到接收 FIFO。向除了 SETUP 外的所有传输都发送 NAK 握手信号，丢弃所有的同步 OUT 包。。
2	GINSTS	R	0	GINSTS：全局 IN NAK 状态（Global IN NAK status) 0：根据发送 FIFO 中数据的状态发送握手信号。 1: 不管发送 FIFO 中数据的状态，对所有非周期性 IN 端点都发送 NAK 握手信号
1	SDIS	RW	1	SDIS：软件断开（Soft disconnect) 应用程序通过此位告知 OTG 控制器执行软件断开操作。当设置此位后，主机将看到设备已经断开，设备方将不会从 USB 总线上收到任何信号。控制器将保持在断开状态，直到程序清除此位。 0：普通操作。当此位在设备软件断开后清除，控制器将发送一个设备连接事件到主机，主机将重新执行枚举操作。 1：控制器执行设备软件断开操作

0	RWUSIG	RW	0	RWUSIG：远程唤醒信号（Remote wakeup signaling) 当应用程序设置此位，控制器将发送远程唤醒信号，唤醒 USB 主机。应用程序必需设置此位来使控制器退出挂起状态。根据 USB2.0 规范，程序必需在设置此位后的 1~15ms 之间再次清除它

为了使 USB 主机能识别到设备断开的操作，软件断开(SDIS)位需要保持一段时间，下表列出了最小的持续时间(根据设备的不同状态)。为了适应时钟的抖动，建议应用程序在定义的最短时间外再额外保留一段时间的延迟。

软件断开的最短持续时间

操作速度	设备状态	最短持续时间
全速	挂起	1ms + 2.5us
全速	空闲	2.5us
全速	不空闲或挂起(正在执行传输操作)	2.5us

32.6.3 OTG_FS 设备状态寄存器(OTG_FS_DSTS)

Address:0x0808

Reset value:0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.										FNSOF [13:8]					
										R					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FNSOF [7:0]								Res.				EERR	ENUMSPD [1:0]		SUSPSTS
R													R		R

Bit	Name	R/W	Reset Value	Function
31：22	保留	-	-	-
21:8	FNSOF [13:0]	R	14’h0	接收到的 SOF 的帧号（Frame number of the received SOF)。
7:4	保留	-	-	-
3	EERR	R	0	EERR：奇怪的错误（Erratic error) 控制器在发生一些奇怪的错误时设置此位。 如果发生了奇怪的错误，OTG_FS 控制器将进入挂起状态，并设置控制器中断寄存器的早期挂起位(OTG_FS_GINTSTS 寄存器的 ESUSP 位)随之产生中断。如果早期挂起中断是由于此位引起的，应用程序只能通过执行软件断开来解决。
2:1	ENUMSPD [1:0]	R	2’h1	ENUMSPD：枚举速度（Enumerated speed) 指示 OTG_FS 控制器通过序列选定的执行速度。 2’b01：保留； 2’b10：保留； 2’b11：全速(PHY 时钟运行在 48MHz)； 其他：保留
0	SUSPSTS	R	0	SUSPSTS：挂起状态（Suspend status) 在设备模式下，如果在 USB 总线上检测到了挂起条件，将设置此位。控制器将在检测到 USB 数据线在 3ms 内没有活动的情况下进入挂起状态。控制器在以下条件时退出挂起模式：

				当 USB 数据线上出现了活动 当应用程序设置设备控制寄存器的远程唤醒信号位(OTG_FS_DCTL 寄存器的 RWUSIG 位)

32.6.4 OTG_FS 设备 IN 端点通用中断屏蔽寄存器(OTG_FS_DIEPMSK)

Address:0x0810

Reset value:0x0000 0000

此寄存器配合设备 IN 端点中断寄存器（OTG_FS_DIEPINTx）来产生 IN 端点中断。对应于 OTG_FS_DIEPINTx 寄存器的相应 IN 端点中断可以通过配置此寄存器来屏蔽。所有的中断在默认状态下都为屏蔽状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
									RWE	RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31：7	保留	-	-	-
6	INEPNEM	RW	0	INEPNEM：IN 端点 NAK 状态有效中断屏蔽（IN endpoint NAK effective mask) 0：中断屏蔽; 1：中断不屏蔽
5	INEPNMM	RW	0	INEPNMM: 端点收到 IN 命令不匹配中断屏蔽 (IN token received with EP mismatch mask) 0：中断屏蔽; 1：中断不屏蔽
4	ITTXFEMSK	RW	0	ITTXFEMSK: 当发送 FIFO 空时收到 IN 命令中断屏蔽（IN token received when TxFIFO empty mask) 0：中断屏蔽; 1：中断不屏蔽
3	TOM	RW	0	TOM：超时检测中断屏蔽(非同步端点)（Timeout condition mask（Non-isochronous endpoints)) 0：中断屏蔽; 1：中断不屏蔽
2	保留	-	-	-
1	EPDM	RW	0	EPDM：端点被禁止中断屏蔽（Endpoint disabled interrupt mask) 0：中断屏蔽; 1：中断不屏蔽。
0	XFRCM	RW	0	XFRCM: 传输结束中断屏蔽（Transfer completed interrupt mask) 0：中断屏蔽; 1：中断不屏蔽

32.6.5 OTG_FS 设备 OUT 端点通用中断屏蔽寄存器(OTG_FS_DOEPMSK)

Address:0x0814

Reset value:0x0000 0000

此寄存器配合设备 OUT 端点中断寄存器(OTG_FS_DOEPINTx)使用，产生 OUT 端点中断。

OTG_FS_DOEPINTx 寄存器的每一位都可以通过写此寄存器的相应位来屏蔽。在默认状态下，所有中断都屏蔽。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									B2BSTUP	Res.	OTEPDM	STUPM	Res.	EPDM	XFRCM
									RW		RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31：7	保留	-	-	-
6	B2BSTUP	RW	0	B2BSTUP：收到连续的 SETUP 包中断屏蔽（Back-to-back SETUP packets received mask) 0：中断屏蔽; 1：中断不屏蔽
5	保留	-	-	-
4	OTEPDM	RW	0	OTEPDM: 当端点被禁止时收到 OUT 命令中断屏蔽 (OUT token received when endpoint disabled mask) 0：中断屏蔽; 1：中断不屏蔽
3	STUPM	RW	0	STUPM：SETUP 阶段完成中断屏蔽（SETUP phase done mask) 仅对控制端点有效 0：中断屏蔽; 1：中断不屏蔽
2	保留	-	-	-
1	EPDM	RW	0	EPDM：端点被禁止中断屏蔽（Endpoint disabled interrupt mask) 0：中断屏蔽; 1：中断不屏蔽。
0	XFRCM	RW	0	XFRCM: 传输结束中断屏蔽（Transfer completed interrupt mask) 0：中断屏蔽; 1：中断不屏蔽

32.6.6 OTG_FS 设备所有端点中断寄存器(OTG_FS_DAINT)

Address:0x0818

Reset value:0x0000 0000

当每个端点发生了事件时，设备所有端点中断寄存器分别通过控制器中断寄存器的设备 OUT 端点中断位或设备 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 或 IEPINT 位)来产生中断打断应用程序。每个端点都有相对应的一位，OUT 端点有 16 位，IN 端点也有 16 位。对于双向端点，同时使用 IN 和 OUT 位。此寄存器的相应位，在应用程序设置和清除相应的设备端点 x 中断寄存器(OTG_FS_DIEPINTx 和 OTG_FS_DOEPINTx 寄存器)的相应位时自动设置和清除。。

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

Bit	Name	R/W	Reset Value	Function
31：20	保留	-	-	-
19:16	OEPINT [3:0]	R	4’h0	OEPINT：OUT 端点中断位（OUT endpoint interrupt bits) 每个位对应一个 OUT 端点。 位 16 对应 OUT 端点 0，位 18 对应 OUT 端点 3。
15:4	保留	-	-	-
3:0	IEPINT [3:0]	R	4’h0	IEPINT: IN 端点中断位(IN endpoint interrupt bits) 每个位对应一个 IN 端点。 位 0 对应 IN 端点 0，位 3 对应 IN 端点 3

32.6.7 OTG_FS 所有端点中断屏蔽寄存器(OTG_FS_DAINTMSK)

Address:0x081C

Reset value:0x0000 0000

设备所有端点中断屏蔽寄存器与设备端点中断寄存器配合使用，在发生设备端点事件时，产生中断打断应用程序。然而，设备所有端点中断寄存器(OTG_FS_DAINT)的相应位在屏蔽时仍然置位。

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

Bit	Name	R/W	Reset Value	Function
31：20	保留	-	-	-
19:16	OEPM [3:0]	RW	4’h0	OEPM：OUT 端点中断屏蔽寄存器（OUT EP interrupt mask bits) 每个位对应一个 OUT 端点。 位 16 对应 OUT 端点 0，位 19 对应 OUT 端点 3。 0：屏蔽中断； 1：不屏蔽中断
15:4	保留	-	-	-
3:0	IEPM [3:0]	RW	4’h0	IEPM：IN 端点中断屏蔽位（IN EP interrupt mask bits) 每个位对应一个 IN 端点。 位 0 对应 IN 端点 0，位 3 对应 IN 端点 3。 0：屏蔽中断； 1：不屏蔽中断

32.6.8 OTG_FS 设备 VBUS 放电时间寄存器(OTG_FS_DVBUSDIS)

Address:0x0828

Reset value:0x0000 17D7

此寄存器定义 SRP 期间，在 VBUS 脉冲后的 VBUS 放电时间

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	VBUSDT [15:0]	RW	16'h17D7	VBUSDT：设备 VBUS 放电时间（Device VBUS discharge time) 定义在 SRP 期间，VBUS 脉冲后的 VBUS 放电时间。这个数值等于以 PHY 时钟计算的 VBUS 放电时间/1024。 根据不同的 VBUS 负载，这个数值可以适当调整

32.6.9 OTG_FS 设备 VBUS 脉冲时间寄存器(OTG_FS_DVBUSPULSE)

Address:0x082C

Reset value:0x0000 05B8

此寄存器定义 SRP 期间 VBUS 脉冲的时间

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

Bit	Name	R/W	Reset Value	Function
31:12	保留	-	-	-
11:0	DVBUSP [11:0]	RW	12'h5B8	DVBUSP：设备 VBUS 脉冲时间（Device VBUS pulsing time) 定义在 SRP 期间，VBUS 脉冲时间。这个数值等于以 PHY 时钟计算的 VBUS 脉冲时间/1024。

32.6.10 OTG_FS 设备 IN 端点 FIFO 空中断屏蔽寄存器(OTG_FS_DIEPEMPMSK)

Address:0x0834

Reset value:0x0000 0000

此寄存器配合 IN 端点 FIFO 空中断寄存器(TXFE_OTG_FS_DIEPINTx)产生中断

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

Bit	Name	R/W	Reset Value	Function
31：4	保留	-	-	-
3:0	INEPTXFEM [3:0]	RW	4’h0	INEPTXFEM：IN 端点发送 FIFO 空中断屏蔽位（IN EP Tx FIFO empty interrupt mask bits) 此位用于屏蔽 OTG_FS_DIEPINTx 寄存器的相应位。 TXFE 中断的一位对应一个 IN 端点：位 0 对应 IN 端点 0，位 3 对应 IN 端点 3。 0：屏蔽中断； 1：不屏蔽中断

32.6.11 OTG_FS 设备控制 IN 端点 0 控制寄存器(OTG_FS_DIEPCTL0)

Address:0x0900

Reset value:0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.		SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAKSTS	Res.
RS	RS			W	W	RW				RS		R		R	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBAEP	Res.													MPSIZ [1:0]	
R														RW	

Bit	Name	R/W	Reset Value	Function
31	EPENA	RS	0	EPENA：端点使能（Endpoint enable) 应用程序设置此位，以启动在端点 0 上的数据传输。 控制器在产生以下端点中断前会先清除此位： 端点禁止； 传输完成。
30	EPDIS	RS	0	EPDIS：端点禁止（Endpoint disable) 应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到 端点禁止中断才能确为此端点已禁止。控制器在设置端点中断中断时会清除此位。应用程序只有 在端点已经使能时才能设置此位。
29:28	保留	-	-	-
27	SNAK	W	0	SNAK：设置 NAK（Set NAK) 写此位会设置端点的 NAK 状态 通过设置此位，程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包时设置此位。
26	CNAK	W	0	CNAK：清除 NAK（Clear NAK) 写此位会清除端点的 NAK 状态
25:22	TXFNUM[3:0]	RW	4’h0	TXFNUM：发送 FIFO 的编号（TxFIFO number) 此位设置分配给 IN 端点 0 的 FIFO 编号
21	STALL	RS	0	STALL：STALL 握手（STALL handshake) 应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或 全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有高优先级

20	保留	-	-	-
19:18	EPTYP[1:0]	R	2’h0	EPTYP：端点类型（Endpoint type) 对于控制端点，由硬件置为’00’。
17	NAKSTS	R	0	NAKSTS：NAK 状态（NAK status) 指示以下： 0：根据 FIFO 状态，控制器发送非 NAK 握手信号； 1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置此位还是控制器设置此位，控制器都将停止数据传输，而 不管发送 FIFO 中的数据是否有效。不管此位设置如何，控制器永远发送 ACK 握手响应 SETUP 数据包
16	保留	-	-	-
15	USBAEP	R	1	USBAEP：USB 活跃端点（USB active endpoint) 此位永远为’1’，指示控制端点 0 在任何配置情况下都是活跃的
14:2	保留	-	-	-
1:0	MPSIZ [1:0]	RW	2’h0	MPSIZ：最大包长度（Maximum packet size) 应用程序通过此位配置该端点的最大数据包长度 2’b00：64 字节； 2’b01：32 字节； 2’b10：16 字节； 2’b11：8 字节。

32.6.12 OTG 设备端点 x 控制寄存器(OTG_FS_DIEPCTLx)(其中 x 为端点号，x=1…3)

Address:0x0900+(x*0x20)

Reset value:0x0000 0000

应用程序通过这些寄存器控制非 0 端点的操作特性。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAKSTS	EONUM/DPID
RS	RS	W	W	W	W	RW				RW/RS		RW		R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBAEP	Res.				MPSIZ [10:0]										
RW					RW										

Bit	Name	R/W	Reset Value	Function
31	EPENA	RS	0	EPENA：端点使能（Endpoint enable) 应用程序设置此位，以启动端点上的数据传输。 控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成 端点禁止 传输完成
30	EPDIS	RS	0	EPDIS：端点禁止（Endpoint disable)

				应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到端点禁止中断才能确为此端点已禁止。控制器在设置端点中断中断时会清除此位。应用程序只有在端点已经使能时才能设置此位
29	SODDFRM	W	0	SODDFRM：设置奇数帧（Set odd frame) 仅对同步的 IN 和 OUT 端点有效。 写此位在 EONUM 中选择奇数帧
28	SD0PID	W	0	SD0PID：设置 DATA0 PID（Set DATA0 PID) 仅对中断/块传输 IN 端点有效。 写此位将设置数据 PID 位为 DATA0
	SEVNFRM	W	0	SEVNFRM：设置偶数帧 仅对同步 IN 端点有效。 写此位在 EONUM 中选择偶数帧
27	SNAK	W	0	SNAK：设置 NAK（Set NAK) 写此位会设置端点的 NAK 状态 通过设置此位，程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或 OUT 端点传输结束中断时设置此位
26	CNAK	W	0	CNAK：清除 NAK（Clear NAK) 写此位会清除端点的 NAK 状态
25:22	TXFNUM[3:0]	RW	4’h0	TXFNUM：发送 FIFO 的编号（TxFIFO number) 此位设置分配给该端点的 FIFO 编号，每个有效的 IN 端点都必需分配一个独立的 FIFO 编号。此位仅对 IN 端点有效。
21	STALL	RS	0	STALL：STALL 握手（STALL handshake) 仅对非控制、非同步 IN 端点有效(操作类型为 rw) 应用程序设置此位，该端点会以 STALL 来响应所有的主机命令。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有最高优先级。只有应用程序能清除此位，控制器不能。 仅对控制端点有效(操作类型是 rs) 应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有高优先级。然而控制器将永远用 ACK 握手来响应 SETUP 数据包
20	保留	-	-	-
19:18	EPTYP[1:0]	RW	2’h0	EPTYP：端点类型（Endpoint type) 此位指示端点的传输类型 2’b00：控制； 2’b01：同步； 2’b10：块传输； 2’b11：中断；
17	NAKSTS	R	0	NAKSTS：NAK 状态（NAK status)

				指示以下状态： 0：根据 FIFO 状态，控制器发送非 NAK 握手； 1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置还是控制器设置： 对于非同步 IN 端点，控制器都将停止数据传输，而不管发送 FIFO 中的数据是否有效。 对于同步 IN 端点，即使发送 FIFO 中的数据有效，控制器都将发送一个零长度的数据包。 不管此位设置如何，控制器永远发送 ACK 握手以响应 SETUP 数据包
16	EONUM	R	0	EONUM：偶数/奇数帧（Even/odd frame) 仅对同步 IN 端点有效： 指示该端点进行同步数据收发的帧号。程序必需适用本寄存器中 SEVNFRM 和 SODDFRM 位的状态来，设置希望收发帧的偶数/奇数帧号。 0：偶数帧； 1：奇数帧
	DPID	R	0	DPID：端点数据 PID（Endpoint data PID) 仅对中断/块传输 IN 端点有效： 此位保存通过此端点来传输的数据包的 PID。应用程序必需在使能端点后，配置此位，选择首个发送或接收的数据包的 PID。应用程序通过 SD0PID 寄存器位来配置 DATA0 或 DATA1。 0：DATA0； 1：DATA1
15	USBAEP	RW	0	USBAEP：USB 活跃端点（USB active endpoint) 指示在当前配置下，此端点是否为活跃端点。控制器在检测到 USB 复位后会清除所有端点的此位(除了端点 0)，在收到 SetConfiguration 和 SetInterface 命令后，应用程序必需根据需要设置此位。
14:11	保留	-	-	-
10:0	MPSIZ [10:0]	RW	11’h0	MPSIZ：最大包长度（Maximum packet size) 应用程序通过此位配置该端点的最大数据包长度此位的值以字节为单位

32.6.13 OTG_FS 设备控制 OUT 端点 0 控制寄存器(OTG_FS_DOEPCTL0)

Address:0x0B00

Reset value:0x0000 8000

本节描述了设备控制 OUT 端点 0 的控制寄存器。非 0 控制端点使用相应的端点 1-3 控制寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.		SNAK	CNAK	Res.				STALL	SNPM	EPTYP[1:0]		NAKSTS	Res.
RS	R			W	W					RS	RW	R		R	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBAEP	Res.													MPSIZ	
R														R	

Bit	Name	R/W	Reset Value	Function
31	EPENA	RS	0	EPENA：端点使能（Endpoint enable) 应用程序设置此位，以启动端点 0 上的数据传输。控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成； 端点禁止； 传输完成
30	EPDIS	R	0	EPDIS：端点禁止（Endpoint disable) 应用程序不可以禁止控制 OUT 端点 0
29:28	保留	-	-	-
27	SNAK	W	0	SNAK：设置 NAK（Set NAK) 写此位会设置端点的 NAK 状态。 通过设置此位，应用程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或传输结束中断时设置此位
26	CNAK	W	0	CNAK：清除 NAK（Clear NAK) 设置此位会清除端点的 NAK 状态
25:22	保留	-	-	-
21	STALL	RS	0	STALL：STALL 握手（STALL handshake) 应用程序只能设置此位；当收到一个 SETUP 命令时，由控制器清除。即使同时设置了 NAK 位、或全局 OUT NAK 位，STALL 位仍有高优先级。然而无论此位如何设置，控制器总是以 ACK 握手来响应 SETUP 数据包
20	SNPM	RW	0	SNPM：监听模式（Snoop mode) 设置此位将使端点进入监听模式，在监听模式下，控制器在将数据传送入应用程序缓存区之前将不会检测 OUT 包的正确性。
19:18	EPTYP [1:0]	R	2’h0	EPTYP：端点类型（Endpoint type) 由硬件设置为’00’。
17	NAKSTS	R	0	NAKSTS：NAK 状态（NAK status) 指示以下状态： 0: 根据 FIFO 状态，控制器发送非 NAK 握手信号。 1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置还是控制器设置： 控制器都将停止数据传输，而不管接收 FIFO 中的数据是否有效。不管此位设置如何，控制器永远发送 ACK 握手信号响应 SETUP 数据包
16	保留	-	-	-
15	EPACT	R	1	USBAEP：USB 活跃端点（USB active endpoint) 总是为’1’，表示控制端点 0 在任何配置下都是活跃的。
14:2	保留	-	-	-
1:0	MPSIZ [1:0]	R	2’h0	MPSIZ：最大包长度（Maximum packet size)

				控制 OUT 端点 0 的最大数据包长度必须与控制 IN 端点 0 的最大数据包长度一致。 2'b00：64 字节； 2'b01：32 字节； 2'b10：16 字节； 2'b11：8 字节。

32.6.14 OTG_FS 设备 OUT 端点 x 控制寄存器(OTG_FS_DOEPCTLx)(其中 x 为端点号，x=1…3)

Address:0x0B00+(x*0x20)

Reset value:0x0000 0000

应用程序通过此寄存器来控制除了端点 0 的其他端点的操作特性

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID / SEVNFRM	SNAK	CNAK	Res.				STALL	SNPM.	EPTYP		NAKSTS	EONUM /DPID
RS	RS	W	W	W	W					RW/RS	RW	RW		R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBAEP	Res.				MPSIZ [10:0]										
RW					RW										

Bit	Name	R/W	Reset Value	Function
31	EPENA	RS	0	EPENA：端点使能（Endpoint enable) 对于 IN 和 OUT 端点都有效。 应用程序设置此位，以启动端点上的数据传输。 控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成； 端点禁止； 传输完成
30	EPDIS	RS	0	EPDIS：端点禁止（Endpoint disable) 应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到端点禁止中断才能确为此端点已被禁止。控制器在设置端点中断时会清除此位。应用程序只有在端点已经使能时才能设置此位。。
29	SD1PID	W	0	SD1PID:设置 DATA1 PID(Set DATA1 PID) 仅对中断/块传输 IN/OUT 端点有效。 写此位将设置数据 PID 位为 DATA1
	SODDFRM	W	0	SODDFRM：设置奇数帧（Set odd frame) 仅对同步 OUT 端点有效 设置此位，则在奇偶帧域(EONUM)中选择奇数帧
28	SD0PID	W	0	SD0PID：设置 DATA0 PID（Set DATA0 PID) 仅对中断/块传输 OUT 端点有效。 写此位将设置数据 PID 位为 DATA0
	SEVNFRM	W	0	SEVNFRM：设置偶数帧 仅对同步 OUT 端点有效。

				设置此位，则在奇偶帧域(EONUM)中选择偶数帧
27	SNAK	W	0	SNAK：设置 NAK（Set NAK) 写此位会设置端点的 NAK 状态。 通过设置此位，应用程序能控制控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或 OUT 端点传输结束中断时设置此位
26	CNAK	W	0	CNAK：清除 NAK（Clear NAK) 写此位会清除端点的 NAK 状态
25:22	保留	-	-	-
21	STALL	RS	0	STALL：STALL 握手（STALL handshake) 仅对非控制，非同步 OUT 端点有效(操作类型为 rw) 应用程序设置此位，该端点会以 STALL 来响应所有来自主机的命令。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有最高优先级。只有应用程序能清除此位，控制器不能。 仅对控制端点有效(操作类型是 rs) 应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有高优先级。然而控制器将永远用 ACK 握手来响应 SETUP 数据包
20	SNPM	RW	0	SNPM：监听模式（Snoop mode) 设置此位将使端点进入监听模式。在监听模式下，控制器在将 OUT 数据包写入应用程序缓存区前不检查数据的正确性
19:18	EPTYP [1:0]	RW	2'h0	EPTYP：端点类型（Endpoint type) 此位指示端点的传输类型： 2'b00：控制 2'b01：同步 2'b10：块传输 2'b11：中断
17	NAKSTS	R	0	NAKSTS：NAK 状态（NAK status) 指示以下状态： 0: 根据 FIFO 状态, 控制器发送非 NAK 握手信号。 1：控制器发送 NAK 握手信号。 只要设置了此位, 不管是应用程序设置还是控制器设置： 控制器都将停止数据传输，而不管接收 FIFO 是否能接收数据。 不管此位设置如何，控制器永远发送 ACK 握手响应 SETUP 数据包
16	EONUM	R	0	EONUM：偶数/奇数帧（Even/odd frame) 仅对同步 OUT 端点有效：

				指示该端点进行同步数据收发的帧号。程序必需根据本寄存器中 SEVNFRM 和 SODDFRM 位的状态来设置偶数/奇数帧号。 0：偶数帧； 1：奇数帧
	DPID	R	0	DPID：端点数据 PID（Endpoint data PID） 仅对中断/块传输 OUT 端点有效： 此位保存通过此端点来传输的数据包的 PID。程序必需在使能端点后配置此位，选择首个发送或接收的数据包的 PID。应用程序通过 SD0PID 寄存器位来配置 DATA0 或 DATA1。 0：DATA0； 1：DATA1
15	USBAEP	RW	0	USBAEP：USB 活跃端点（USB active endpoint） 指示在当前配置下，此端点是否为活跃端点。控制器在检测到 USB 复位后会清除所有端点的这一位(除了端点 0)，在收到 SetConfiguration 和 SetInterface 命令后，程序必需根据需要设置此位
14:11	保留	-	-	-
10:0	MPSIZ [10:0]	RW	11'h0	MPSIZ：最大包长度（Maximum packet size） 应用程序通过此位配置该端点的最大数据包长度 长度数值以字节为单位

32.6.15 OTG_FS 设备端点 x 中断寄存器(OTG_FS_DIEPINTx)(其中 x 为端点号，x=0…3)

Address:0x0908+(x*0x20)

Reset value:0x0000 0080

此寄存器指示相应端点与 USB 和 AHB 相关的事件状态。当控制器中断寄存器的 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 IEPINT 位)为'1'时，程序必需先读设备所有端点中断寄存器(OTG_FS_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTG_FS_DAINT 和 OTG_FS_GINTST 寄存器的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								TXFE	INEPNE	Res.	ITTXFE	TOC	Res.	EPDISD	XFRC
								R	RC_W1		RC_W1	RC_W1		RC_W1	RC_W1

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7	TXFE	R	1	TXFE：发送 FIFO 空（Transmit FIFO empty） 当与此端点相对应的发送 FIFO 为空或者半空时，会产生此中断。控制器 AHB 配置寄存器的 TxFIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)将决定发送 FIFO 在空还是半空状态时产生中断。
6	INEPNE	RC_W1	0	INEPNE：IN 端点 NAK 有效（IN endpoint NAK effective）

				写 OTG_FS_DIEPCTLx 寄存器的 CNAK 位将清除 IN 端点的 NAK 状态，同时也清除此位。 此中断意味着控制器检测到 NAK 已经被设置(可以由应用程序设置也可以由控制器设置)，这个中断指示由应用程序设置的 NAK 位已经生效。 这个中断不能保证已经在 USB 线上发送了 NAK 信号。STALL 位的优先级高于 NAK 位。
5	保留	-	-	-
4	ITTXFE	RC_W1	0	ITTXFE：当发送 FIFO 空时收到 IN 命令（IN token received when TxFIFO is empty) 仅对于非周期性的 IN 端点有效。 指示与此端点相关的发送 FIFO(周期性/非周期性)为空时，收到了 IN 的命令。此中断仅在收到 IN 命令的端点产生。
3	TOC	RC_W1	0	TOC：超时（Timeout condition) 仅对控制 IN 端点有效。 指示自该端点收到最后一个 IN 命令以来，控制器监测到了超时状态
2	保留	-	-	-
1	EPDISD	RC_W1	0	EPDISD: 端点禁止中断 (Endpoint disabled interrupt) 此中断指示根据应用程序的请求，此端点已经被禁止
0	XFRC	RC_W1	0	XFRC：传输完成中断（Transfer completed interrupt) 此中断指示该端点上已配置好的传输，无论在 AHB 方面还是 USB 方面，都已传输完毕

32.6.16 OTG_FS 设备端点 x 中断寄存器(OTG_FS_DOEPINTx)(其中 x 为端点号，x=0…3)

Address:0x0B08+(x*0x20)

Reset value:0x0000 0000

此寄存器指示相应端点与 USB 和 AHB 相关的事件状态。当控制器中断寄存器的 OUT 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 位)为'1'时，应用程序必需先读设备所有端点中断寄存器(OTG_FS_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTG_FS_DAINT 和 OTG_FS_GINTST 寄存器的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.									B2BSTUP	Res	OTEPDIS	STUP	Res.	EPDISD	XFRC
									RC_W1		RC_W1	RC_W1		RC_W1	RC_W1

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	B2BSTUP	RC_W1	0	B2BSTUP：收到连续的 SETUP 包（Back-to-back SETUP packets received) 仅对控制 OUT 端点有效。 指示该端点收到了超过 3 个连续的 SETUP 包
5	保留	-	-	-

4	OTEPDIS	RC_W1	0	OTEPDIS：当端点禁止时收到了 OUT 命令（OUT token received when endpoint disabled) 仅对控制 OUT 端点有效。 指示端点禁止时收到了 OUT 命令，此中断仅在收到 OUT 命令的端点上产生。
3	STUP	RC_W1	0	STUP：SETUP 阶段完成（SETUP phase done) 仅对控制 OUT 端点有效。 指示与此控制端点相关的 SETUP 阶段已经完成，当前传输不会再有更多的连续的 SETUP 包。产生此中断后，应用程序可以开始处理收到的 SETUP 数据包。
2	保留	-	-	-
1	EPDISD	RC_W1	0	EPDISD：端点禁止中断（Endpoint disabled interrupt) 此中断指示根据应用程序的请求，此端点已经被禁止
0	XFRC	RC_W1	0	XFRC：传输完成中断（Transfer completed interrupt) 此中断指示该端点上已配置好的传输，无论在 AHB 端还是 USB 端，都已传输完毕

32.6.17 OTG_FS 设备 IN 端点 0 传输长度寄存器(OTG_FS_DIEPTSIZ0)

Address:0x0910

Reset value:0x0000 0000

应用程序必需在使能端点 0 之前配置此寄存器。一旦通过设备控制端点 0 控制寄存器的端点使能位(OTG_FS_DIEPCTL0 寄存器的 EPENA 位)使能了端点 0，就只有控制器可以修改此寄存器。

当端点使能位被清除之前，应用程序只能读此寄存器。

非 0 端点使用端点 1 至端点 15 的寄存器

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

Bit	Name	R/W	Reset Value	Function
31:21	保留	-	-	-
20:19	PKTCNT [1:0]	RW	2'h0	PKTCNT：包数目(Packet count) 指示在端点 0 上要传输"传输长度"数据，所需要的 USB 数据包的数量。 控制器每从发送 FIFO 中读出一个数据包(最大长度的数据包和短包)，此寄存器的值会自动减 1
18:7	保留	-	-	-
6:0	XFRSIZ [6:0]	RW	7'h0	XFRSIZ：传输长度（Transfer size) 指示端点 0 上要传输的字节数。当传输字节数减为 0 时，控制器会产生中断并通知应用程序。可以在每个包结束后，将此寄存器值设置为端点的最大传输长度。 控制器会在从存储区向发送 FIFO 写数据的时候自动递减此域。

32.6.18 OTG_FS 设备 OUT 端点 0 传输长度寄存器(OTG_FS_DOEPTSIZ0)

Address:0x0B10

Reset value:0x0000 0000

应用程序必需在使能端点 0 前配置好此寄存器。一旦端点 0 通过设备控制端点 0 控制寄存器的端点使能位(OTG_FS_DOEPCTL0 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。控制器在清除端点使能位之前，应用程序只能读此寄存器。

非 0 的端点使用针对端点 1 至端点 3 的寄存器。

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

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:29	STUPCNT	RW	2’h0	STUPCNT：SETUP 包数目（SETUP packet count) 此位指示端点能够接收的连续的 SETUP 包数目 2’b01：1 个包; 2’b10：2 个包; 2’b11：3 个包
28:20	保留	-	-	-
19	PKTCNT	RW	0	PKTCNT：包数目（Packet count) 当一个包被写入接收 FIFO 后，此位递减，直至减为 0。
18:7	保留	-	-	-
6:0	XFRSIZ	RW	7’h0	XFRSIZ：传输长度（Transfer size) 指示端点 0 上的传输字节数。控制器在传输长度为 0 时产生中断并通知应用程序。可以在每个包结束时，设置此寄存器值为端点的最大传输长度，并在每个数据包结束时产生中断。 控制器在数据包从接收 FIFO 中写入存储区时会自动递减此域

32.6.19 OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DIEPTSIZx)(其中 x 为端点号，x=1…3)

Address:0x0910+(x*0x20)

Reset value:0x0000 0000

应用程序必需在使能端点前配置此寄存器。一旦端点通过设备端点 x 控制寄存器的端点使能位(OTG_FS_DIEPCTLx 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。控制器清除了端点使能位之前，应用程序只能读此寄存器。

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

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:29	MCNT [1:0]	RW	2’h0	MCNT：帧内包数目（Multi count)

				对于周期性 IN 端点，此位指示在 USB 一个帧期间要传输的数据包数量。控制器根据此位来计算同步 IN 端点发送的数据 PID 号。 01：1 个包 10：2 个包 11：3 个包 其他：保留
28:19	PKTCNT [9:0]	RW	10'h0	PKTCNT：包数目（Packet count) 指示端点上总共要发送的数据包数。 每从发送 FIFO 中读出一个数据包，自动递减此域
18:0	XFRSIZ [18:0]	RW	19'h0	XFRSIZ：传输长度 该位指示当前端点的传输长度。控制器会在此位为 0 时产生中断并通知应用程序。可以配置此寄存器为端点的最大传输长度，并在每个数据包结束产生中断。 控制器会在每次从存储区向发送 FIFO 写数据时，自动递减此域

32.6.20 OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DOEPTSIZx)(其中 x 为端点号，x=1…3)

Address:0x0B10+(x*0x20)

Reset value:0x0000 0000

应用程序必需在使能端点前，配置此寄存器。一旦端点通过设备端点 x 控制寄存器的端点使能位(OTG_FS_DOEPCTLx 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。当控制器清除端点使能位之前，应用程序只能读此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID / STUPCNT		PKTCNT [9:0]										XFRSIZ [18:16]		
	R/RW		RW										RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ [15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30:29	RXDPID	R/RW	2'h0	RXDPID：收到的数据 PID（Received data PID) 此位仅对同步 OUT 端点有效。 指示收到的最后一个数据包的数据 PID： 00：DATA0； 01：DATA2； 10：DATA1； 11：MDATA。
	STUPCNT	R/RW	2'h0	STUPCNT：SETUP 包数目（SETUP packet count) 此位仅对控制 OUT 端点有效。 指示端点收到的连续的 SETUP 包的数量： 01：1 个包； 10：2 个包； 11：3 个包
28:19	PKTCNT [9:0]	RW	10'h0	PKTCNT：包数目（Packet count) 指示该端点上传输的 USB 包数目。 控制器在每次从存储区向接收 FIFO 中写数据包时，自动递减此域。
18:0	XFRSIZ [18:0]	RW	19'h0	XFRSIZ：传输长度（Transfer size)

				此位指示该端点要传输的字节数。控制器在此域为 0 时会产生中断通知应用程序。可以配置此域为端点的最大传输长度，并在每个数据包结束产生中断。 控制器在每次从接收 FIFO 向存储区写数据时，自动递减此数值。

32.6.21 OTG_FS 设备 IN 端点传输 FIFO 状态寄存器(OTG-FS_DTXFSTSx)(其中 x 为端点号，x=0…3)

Address:0x0918+(x*0x20)

Reset value:0x0000 0100

此寄存器为只读寄存器，保存设备 IN 端点的发送 FIFO 的剩余空间信息。

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

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15:0	INEPTFSAV [15:0]	R	16’h100	INEPTFSAV: IN 端点发送 FIFO 剩余空间(IN end-point TxFIFO space avail) 指示 IN 端点的发送 FIFO 的剩余空间信息，寄存器值以 32 位的字为单位。 16’h0：发送 FIFO 满； 16’h1：剩余 1 个字； 16’h2：剩余 2 个字； 16’hn：剩余 n 个字(其中 0 < n < 256)； 16’h100：剩余 256 个字； 其他：保留。

32.7 OTG_FS 电源和时钟门控寄存器(OTG_FS_PCGCCTL)

Address:0x0E00

Reset value:0x0000 0000

此寄存器在设备模式和主机模式下都有效。

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

Bit	Name	R/W	Reset Value	Function
31：2	保留	-	-	-
1	GATEHCLK	RW	0	GATEHCLK：HCLK 门控（Gate HCLK) 应用程序通过设置此位可以控制 HCLK，在 USB 挂起或会话无效时唤醒逻辑模块。应用程序在 USB 唤醒或新的会话发起时清除此位。
0	STPPCLK	RW	0	STPPCLK：停止 PHY 时钟（Stop PHY clock)

				当 USB 挂起、或会话无效、或者设备断开时，应用程序可以通过设置此位来停止 PHY 的时钟驱动。当 USB 唤醒或新的会话发起时，应用程序可以清除此位

33. 以太网（ENET）

33.1 以太网简介

以太网外设可根据 IEEE 802.3-2002 标准实现以太网的传输和数据接收

以太网提供了可配置、灵活的外围设备以满足各种应用程序和客户的需求。它支持到外部物理层的两个行业标准接口：规范中定义的介质独立接口（MII）、精简介质独立接口（RMII）。它可以用于交换机、网络接口卡等应用。

以太网符合以下标准：

- IEEE 802.3-2002 标准的以太网 MAC
- IEEE 1588-2002 标准的精确网络时钟同步
- AMBA 2.0 for AHB 主/从端口
- RMII 协会定义的 RMII 规范

33.2 以太网特征

以太网外设包含以下的特性：

- 通过外部 PHY 接口支持 10/100 Mbit/s 数据传输速率
- 与外部快速以太网 PHY 接口通信的 IEEE 802.3 兼容的 MII 接口
- 支持全双工、半双工工作
  - 支持 CSMA/CD 协议的半双工工作
  - 支持 IEEE 802.3x 流程的全双工工作
  - 全双工工作模式中，可选择性地将接收的暂停控制帧发给用户应用程序
  - 支持反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
- 在发送包中插入前导码和起始帧数据（SFD），在接收包中删除这些域
- 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
  - 最多 4 个 48 位（DA）地址过滤器，每个字节都有掩码
  - 做多 3 个 48 位 SA 地址比较检测，每个字节都有掩码
  - 用于多播和单播（DA）地址的 64bit 哈希滤波器
  - 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
- 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
- 接收功能：以太网帧框架下的 IPv4 和 TCP 接收包的校验和以及减载
- 高级接收功能：在 IPv4 和 IPv6 数据报中的 IPv4 标头校验和和 TCP,UDP 或 ICMP 校验和检查
- 支持 IEEE 1588-2002 中描述的以太网帧时间戳。每个帧发送或接收状态使用 64 位时间戳
- 2 个 FIFO：具有可编程深度的 2 KB 发送 FIFO，具有可编程深度的 2KB 接收 FIFO
- 接收时 EOF 发送后接收状态会插入到接收 FIFO 中，在接收 FIFO 中使能多帧存储，而不需要另一个 FIFO 来存储这些帧的接收状态
- 接收时可选对错误帧进行过滤，并在存储转发模式下不将其转发到应用程序
- 可选的转发尺寸不够的有效帧
- 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
- 支持在发送时使用存储转发机制
- 根据 FIFO 填充（阈值可配）级别，自动生成暂停帧控制或反压信号给 MAC 控制器
- 发送时处理冲突帧的自动重发
- 在迟到冲突、过度冲突、过度延迟和欠载条件下丢弃帧
- 软件控制清空发送 FIFO
- 计算 IPv4 标头校验和以及 TCP，UDP 或 ICMP 校验和并将其插入以存储和转发模式的传输帧中
- 支持内部的回环 MII 接口进行调试

33.2.1 DMA 特征

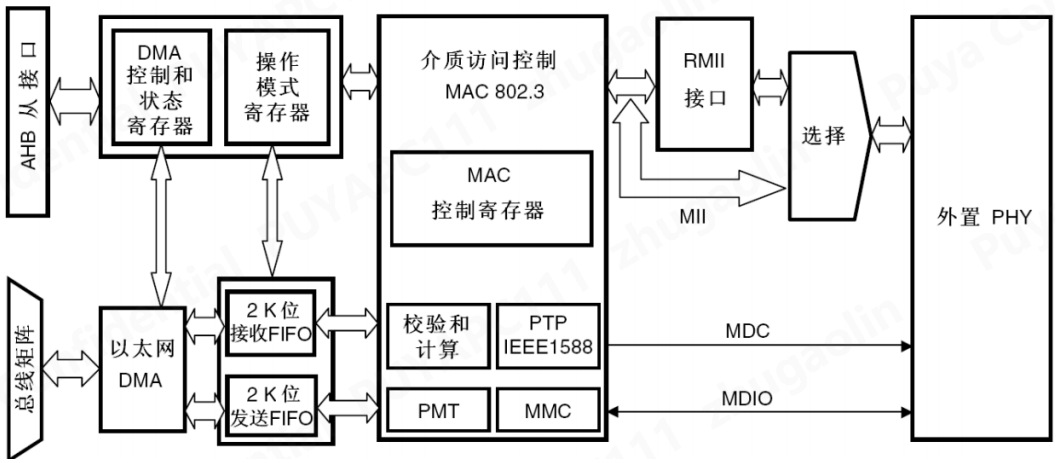
- 在 AHB 从接口下，支持所有类型的 AHB 突发传输
- 在 AHB 主接口下，软件可以选择 AHB 突发传输的类型(固定的或者不固定长度的突发)
- 从 AHB 主端口可选的地址对齐突发
- 使用帧分隔符优化面向数据包的 DMA 传输
- 支持字节对齐方式对数据缓存区寻址
- 双缓存区（环）和链表（链）描述符列表
- 描述符架构，允许以最小的 CPU 干预进行大数据块传输
- 每个描述符最多可传输 2KB 的数据
- 全面的状态报告，用于正常操作和错误的传输
- 发送和接收 DMA 的单独可编程突发大小，优化总线使用
- 适用于不同操作条件的可编程的中断选项
- 每帧发送/接收完的中断控制
- 接收和发送引擎二点循环或固定优先级仲裁
- 开始/停止模式
- 当前 Tx/Rx 缓存区指针作为状态寄存器
- 当前 Tx/Rx 描述符指针作为状态寄存器

33.2.2 PTP 特征

- 接收和发送帧的时间戳
- 粗略和精细校正方法
- 当系统时间大于目标时间时触发中断

- 每秒脉冲输出（产品替代功能输出）

33.2.3 模块框图



33.3 以太网功能描述：SMI、MII 和 RMII

以太网外设由 MAC 802.3（介质访问控制）和专用 DMA 控制器组成。支持默认的介质独立接口（MII）和简化介质独立接口（RMII），两者可通过 1 位控制位选择。

DMA 控制器通过 AHB 主从接口访问 MAC 控制器和存储器。AHB 主接口控制数据的传输，而 AHB 从接口用于访问控制和状态寄存器（CSR）空间。

发送 FIFO（TX FIFO）在 MAC 控制器发送之前通过 DMA 缓存系统内存的数据。类似的是，接收 FIFO（RX FIFO）存储从线路上接收的以太网帧，直到它们通过 DMA 被传送到系统内存。

以太网外设还包含 SMI 接口来与外部的 PHY 芯片进行通信，用户可以通过一组配置寄存器来配置 MAC 和 DMA 控制器以选择期望的工作模式和特性。

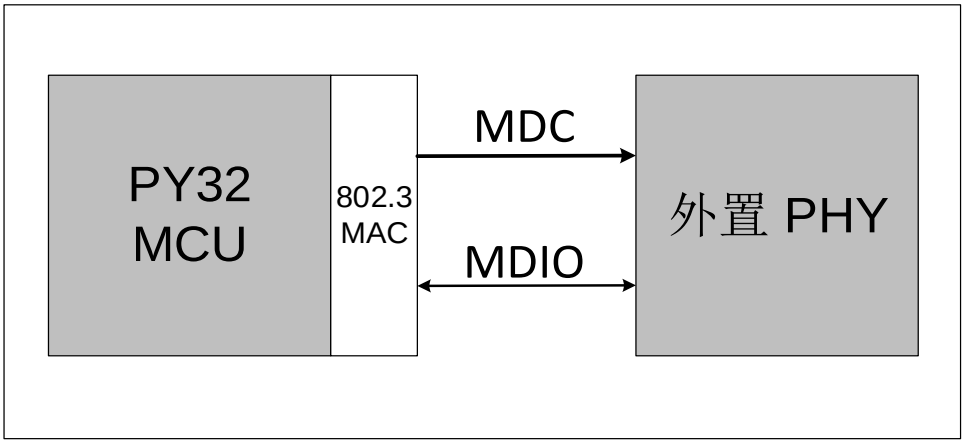
注：当使用以太网时，AHB 时钟频率必须大于等于 25Mhz

33.3.1 状态管理接口 SMI

SMI 允许通过 2 线时钟和数据线实现应用层来访问 PHY 寄存器。该接口最多支持访问 32 个 PHY。应用层可以选择 32 个 HY 中的任意一个 PHY，以及该 PHY 中 32 个寄存器中的任意一个寄存器来进行控制数据的发送和状态信号的读取。在任何给定时间，一个 PHY 中只能寻址一个寄存器

MDC 时钟线和 MDIO 数据线都被实现为 MCU 中的复用功能的 I/O

- MDC：周期性的时钟为数据传输提供时基，最大 2.5Mhz。高电平和低电平最小都为 160ns，周期最小为 400ns。在空闲状态时，SMI 管理接口驱动 MDC 时钟为低电平。
- MDIO：数据输入和输出线，MDC 时钟信号同步地将状态信息发给 PHY 或者从 PHY 中读取



33.3.1.1 SMI 帧格式

读写操作地帧格式 如下表：

	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。

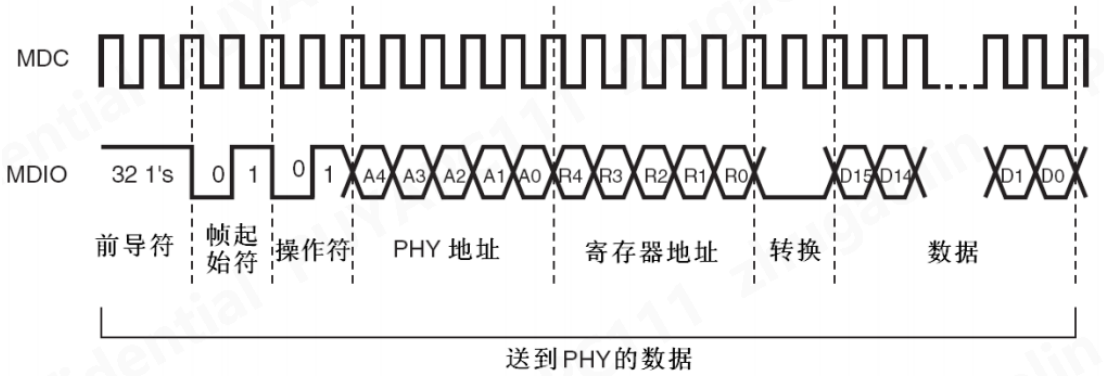
●TA：2 位的转向符，插在 RADDR 和数据(DATA)之间，用于避免读操作时发生冲突。读操作时，在 TA 的这 2 位时间内，MAC 控制器保持 MDIO 线的高阻状态，PHY 设备则先保持 1 位的高阻状态，在第 2 位时输出’0’信号。写操作时，在 TA 的这 2 位时间内，MAC 控制器驱动 MDIO 线输出’10’信号，而 PHY 设置则保持高阻状态。

● DATA：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’。

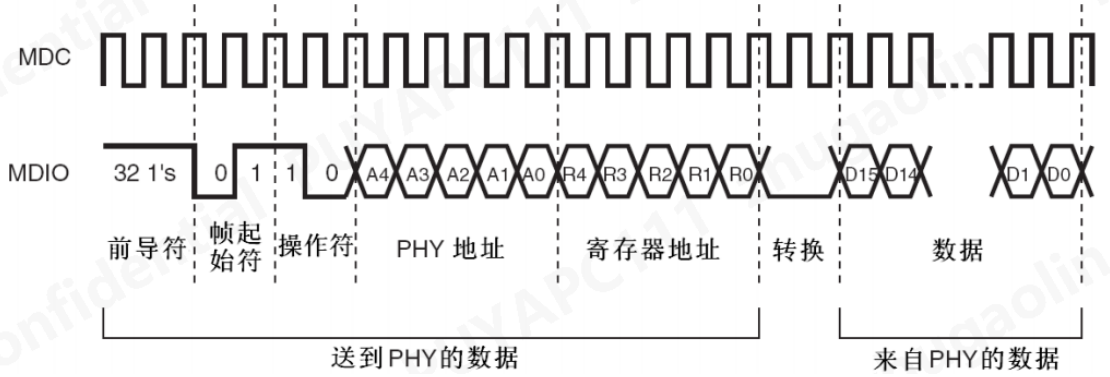
33.3.1.2 SMI 写操作

当应用程序设置 MW 和 MB 位时（ETH_MACMIIAR），SMI 会通过 PHY 的地址对 PHY 寄存器进行一个写操作，并且将 ETH_MACMIIDR 寄存器中的数据写进去。在操作过程中不能修改 MII 地址和数据寄存器，即使修改了也会被忽略（MB 位一直为高），并且发送完也不会报错。当写操作完后，SMI 接口将清除 MB 位。



33.3.1.3 SMI 读操作

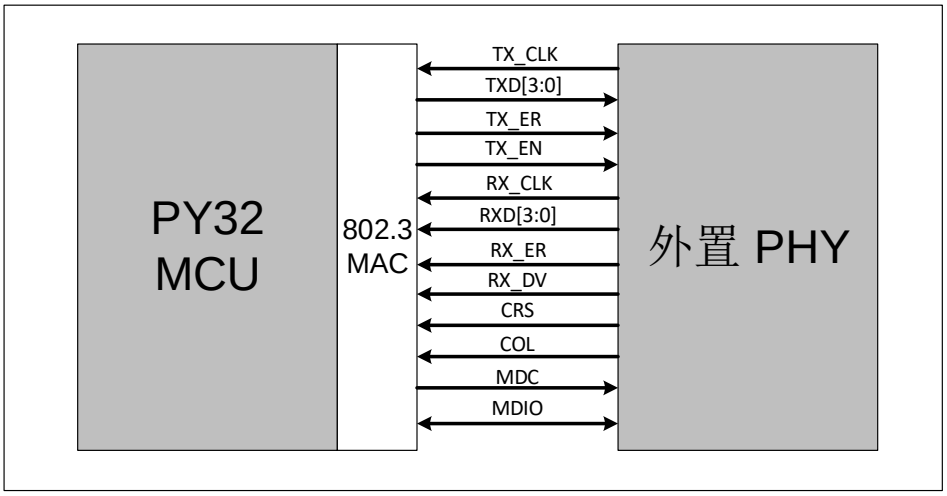
当 ETH_MACMIIAR 寄存器的 MW 位为 0 时，用户设置 MB 位，就会进行一次读操作。在操作过程中不能修改 MII 地址和数据寄存器，即使修改了也会被忽略（MB 位一直为高），并且发送完也不会报错。当读操作完后，MB 拉低，并更新 MII 数据寄存器。



33.3.1.4 SMI 时钟选择

SMI 接口的时钟是由应用时钟（AHB）分频而来，分频系数的选择取决于时钟的范围。具体分频系数见 ETH_MACMIIAR 寄存器的 bit[5:2]。

33.3.2 独立介质接口 MII



- MII_TX_CLK：发送数据使用的时钟信号，对于 10Mbit/s 的数据传输，此时钟为 2.5MHz，对于 100M bit/s 的数据传输，此时钟为 25MHz。
- MII_RX_CLK：接收数据使用的时钟信号，对于 10Mbit/s 的数据传输，此时钟为 2.5MHz，对于 100M bit/s 的数据传输，此时钟为 25MHz。
- MII_TX_EN：发送使能信号，，当数据前导码的起始位出现时，该信号必须有效，并且需要在传输完毕前保持有效。
- MII_TXD[3:0]：发送数据线，每次传输 4 位数据，数据在 MII_TX_EN 信号有效时有效。当 MII_TX_EN 信号无效时，PHY 将忽略传输的数据。
- MII_CRS：载波侦听信号，仅工作在半双工模式下，由 PHY 控制。该信号不需要与 MII_TX_CLK 和 MII_RX_CLK 保持同步。当它处于有效状态时，意味着发送或接收介质不处于空闲状态。MII_CRS 信号一直保持有效，直到发送和接收介质都处于空闲状态。
- MII_COL：冲突检测信号，仅工作在半双工模式下，由 PHY 控制。该信号不需要与 MII_TX_CLK 和 MII_RX_CLK 保持同步。当检测到介质发生冲突时，此信号有效，并且在整个冲突的持续时间内，保持此信号有效。
- MII_RXD[3:0]：接收数据线，每次接收 4 位数据，数据在 MII_RX_DV 信号有效时有效。根据 MII_RX_DV 和 MII_RX_ER 信号的状态，MII_RXD[3:0]数据值可被用来传达一些特定信息
- MII_RX_DV：接收数据使能信号，由 PHY 控制，当 PHY 准备好数据供 MAC 接收时，该信号有效。
- 当数据帧的第一个 4 位出现时，该信号必须有效，并且需要在传输完毕前保持有效。在传送最后 4 位数据后的第一个时钟之前，此信号必须变为无效状态。为确保正确地接收帧，MII_RX_DV 信号应该在 SFD 字段出现之前有效。
- MII_RX_ER：接收出错信号，为了表明 MAC 在接收过程中检测到错误，MII_RX_ER 信号必须在一个或多个时钟周期（MII_RX_CLK）内保持有效。具体错误原因需结合 MII_RX_DV 的状态及 MII_RXD[3:0]的数据值，详见下表：接收接口信号编码

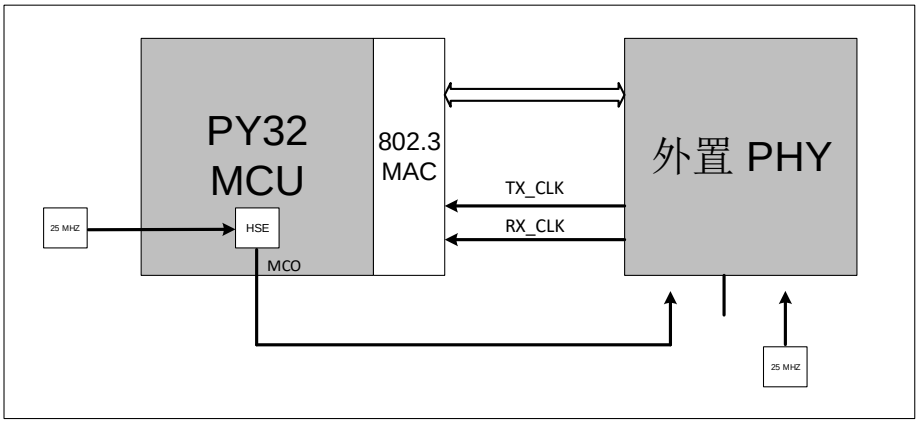
表 33-1 发送接口信号编码

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 33-2 接收接口信号编码

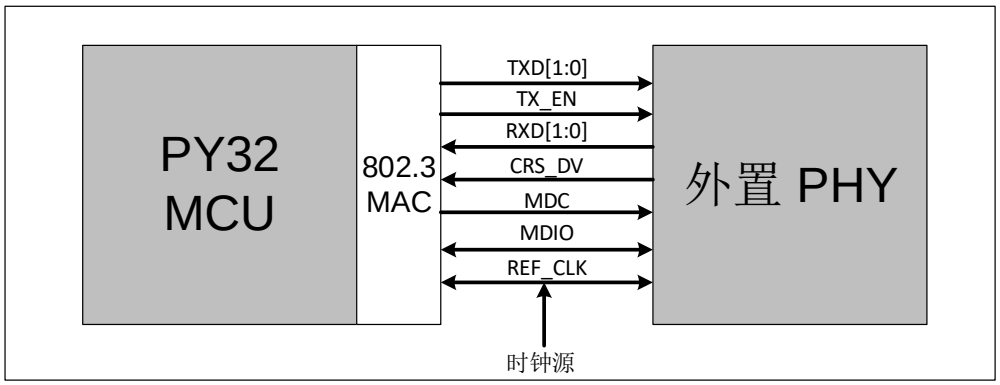
MII_RX_DV	MII_RX_ERR	MII_RXD	描述
0	0	0000 到 1111	正常的帧间隔
0	1	0000	正常的帧间隔
0	1	0001 到 1101	保留
0	1	1110	失败的载波指示
0	1	1111	保留
1	0	0000 到 1111	正常的数据接收
1	1	0000 到 1111	数据接收出错

33.3.2.1 MII 时钟源



用户需要给外部 PHY 提供一个外部的 25 MHz 时钟来产生 TX_CLK 和 RX_CLK 时钟信号。该时钟不需要与 MAC 时钟相同。可以使用外部的 25 MHz 晶振或者微控制器的时钟输出引脚 MCO 提供这一时钟。当时钟来源为 MCO 引脚时需配置合适的 PLL，保证 MCO 引脚输出的时钟为 25 MHz。

33.3.3 简化的独立介质接口 RMII

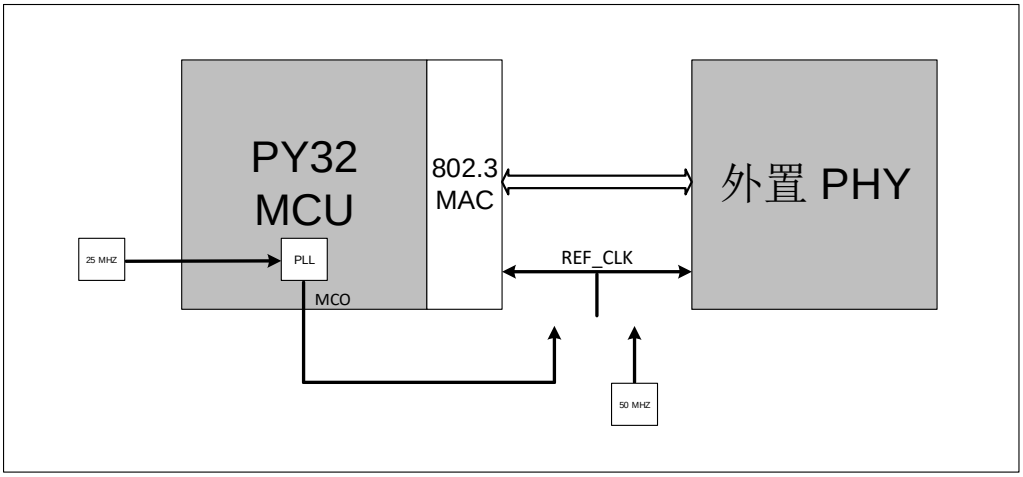


精简媒体独立接口（RMII）规范减少了以太网通信所需要的引脚数。根据 IEEE 802.3 标准，MII 接口需要 16 个引脚用于数据和控制信号，而 RMII 标准则将引脚数减少到了 7 个。

RMII 特性：

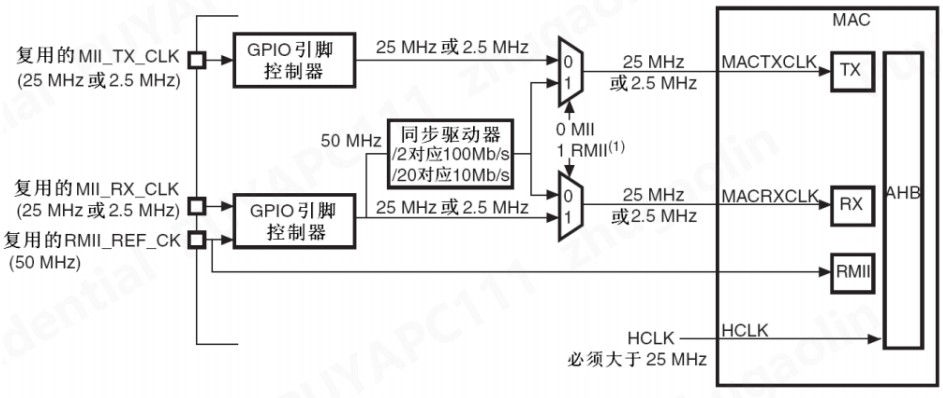
- 支持 10M 和 100Mbit/s 的速率
- 只有一个时钟信号，且该时钟信号需要提高到 50MHz；
- MAC 和外部的以太网 PHY 需要使用同样的时钟源；
- 使用 2 位宽度的数据收发。

33.3.3.1 RMII 时钟源



通过将相同的时钟源接到 MAC 和以太网 PHY 的 REF_CLK 引脚保证两者时钟源的同步。可以通过外部的 50MHz 信号或者微控制器的 MCO 引脚提供这一时钟。当时钟来源 MCO 引脚时需配置合适的 PLL，保证 MCO 引脚输出的时钟为 50MHz。

33.3.4 MII/RMII 选择



MII 和 RMII 的选择是通过 SYSCFG 中的偏移地址 0x0000 寄存器的 bit2。

33.4 以太网功能描述：MAC 802.3

IEEE 802.3 局域网（LAN）国际标准采用 CSMA/CD（具有冲突检测的载波侦听多路访问）作为访问方法。

以太网外设由 MAC802.3 控制器、介质独立接口和 DMA 控制器组成

MAC 块实现以下系统的 LAN CSMA/CD 子层：基带和宽度系统的 10Mbit/s 和 100Mbit/s 数据速率；全双工和半双工工作模式；冲突检测访问方式仅用于半双工操作模式；支持 MAC 控制帧子层

MAC 子层执行与数据链路控制过程相关联的以下功能：

- 数据封装（发送和接收）
  - 生成帧（帧边界定界，帧同步）
  - 生成地址（处理源地址和目的地址）
  - 错误检测
- 介质访问管理
  - 介质分配（避免冲突）
  - 冲突解决（处理冲突）

MAC 子层包含了两种工作模式

- 半双工模式：站点间通过 CSMA/CD 算法争夺物理介质的使用权
- 全双工模式：当以下状态发送时能同时发送和接收而不需要竞争解决
  - 支持同时发送和接收的物理介质能力
  - 2 个站点连接到 LAN
  - 2 个站点同时配置为全双工模式

33.4.1 MAC 802.3 帧格式

通过 CSMA/CDMAC 定义了两种帧格式来进行数据传输：

- 基础 MAC 帧格式
- 标签 MAC 帧格式（基础 MAC 帧格式的扩展）
- 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

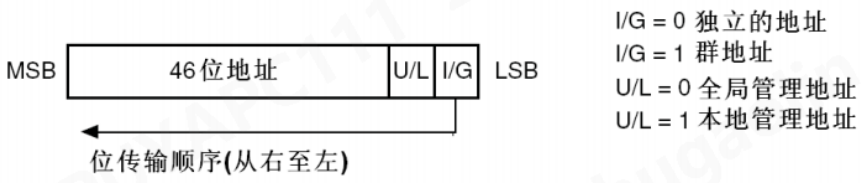
  二进制：01010101 01010101 01010101 01010101 01010101 01010101 01010101（从右至左的传输)
- 帧首符(SFD)：1 个字节，用于指示帧首。

  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：
  - 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。

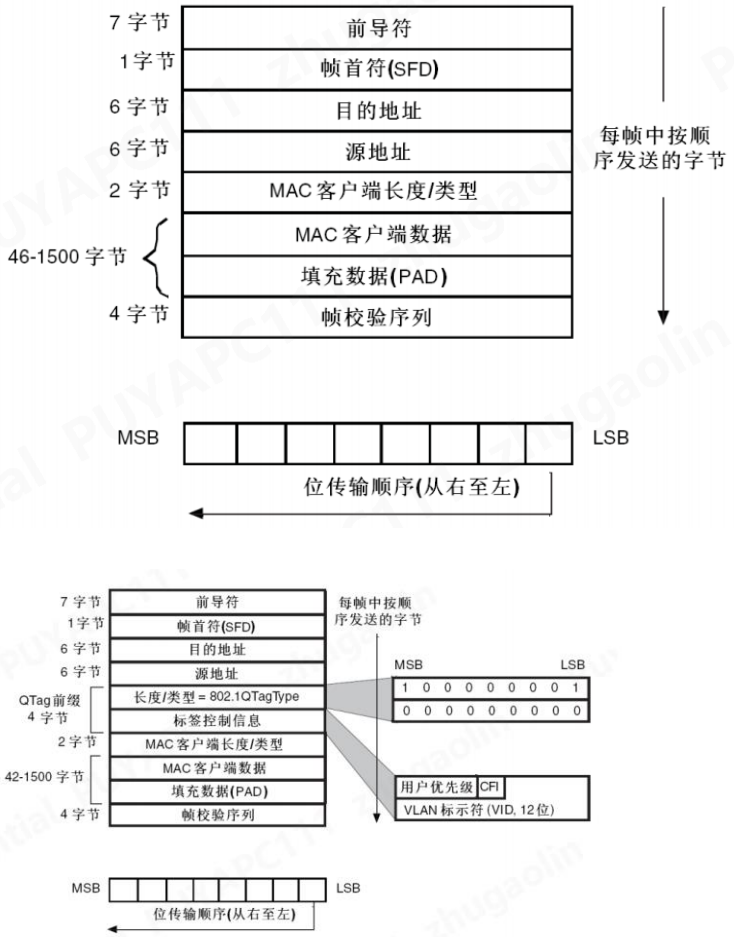
— 广播地址：是一个唯一的、预先约定好的多发地址(目的地址位全为 1)，总是指向给定 LAN 中的所有站点。



- 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(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 子层接收也可能不被接收。
- 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：
  $G(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$
  一个帧的 CRC 值计算如下：
  不带标签的 MAC 帧的最短长度为 46 字节。
  — 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 xn-1 项的系数，数据域的末位作为 x0 项的系数。
  — M(x)多项式乘以 x32，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。x32 的系数最先传输，x0 的系数最后传输。



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。

33.4.2 MAC 帧发送

DMA 控制发送路径所有的事务，通过将以太网帧从系统内存中取出再存到 FIFO 中。然后这些帧被发送到 MAC 控制器。当 EOF（end of frame）发送完后，表示传输帧结束时，传输状态从 MAC 控制器获取，并传输回 DMA。发送 FIFO 的深度为 2 Kbyte。FIFO 的填充水平用来指示给 DMA 以便于它可以使用 AHB 接口从系统内存中提取所需的突发数据。

当 SOF 被检测到，MAC 接收数据并且发送给 MII。应用层开启发送到帧数据到 MII 接口的时间是可变化的，取决于类似于 IFG 的延迟，传输前同步码/界定码的的时间，以及半双工模式中的退避（发生冲突后强制性重传）延迟。当 EOF 被发送到 MAC 控制器，控制器会完成常规的发送然后把发送状态传

给 DMA。如果一个常规冲突（半双工模式中）发生在发送阶段，MAC 控制器会让发送状态有效，然后接收并且丢弃多余的数据直到下一个 SOF 被接收到。在观察到来自 MAC 的重试请求时，应从 SOF 重新发送相同的帧。如果在传输期间没有连续提供数据，则 MAC 会发出下溢状态。在帧的正常传输期间，如果 MAC 在没有获得前一帧的 EOF 的情况下接收 SOF，则 SOF 被忽略，并且新帧被认为是上一帧的延续。

向 MAC 控制器发数据有两种操作模式：

- 阈值模式：当 FIFO 中的字节超过配置的阈值点时（或在超过阈值前写入 EOF），数据接准备转发给 MAC 控制器。阈值点可通过 ETH_DMABMR 的 TTC 位配置。
- 存储转发模式：仅当一个完整帧的存储在 FIFO 后，开始转发到 MAC 控制器。如果 TX FIFO 的深度小于发送帧的长度，那么当 FIFO 快满时该帧数据也会开始转发给 MAC 控制器

◆ 自动的CRC和字节填充生成

当从应用层接收的字节数量小于 60（目的地址+源地址+长度/类型+数据），会在发送帧的数据后面补 0 来保证数据长度能达到 46 字节，以满足 IEEE802.3 要求最小的数据长度。MAC 也可以配置不增加填充域。计算帧校验序列 (FCS) 字段的循环冗余校验 (CRC)，并将其附加到正在传输的数据中。当 MAC 被配置不在帧尾添加 CRC 值，计算的 CRC 就不会被传输。此规则有一个例外，即如果 MAC 被配置附加填充字节，CRC 将会在帧尾添加。

- 发送协议

MAC 控制发送以太网帧的工作过程：

1. 生成前导码和界定符
2. 在半双工模式中生成阻塞 01 序列（jam pattern）
3. 控制 jabber 定时器
4. 控制半双工模式的流量（反压）
5. 生成发送帧状态
6. 包含时间戳拍照逻辑

当一个新的帧发送被请求时，MAC 发送前导码和界定符，然后是数据。冲突窗口被定义为 1 个间隙时间（512bit 对应的时间），阻塞模式仅用于半双工模式。

在 MII 模式中，如果一个冲突在一个帧的任意位置发送，MAC 会发送 32 位的阻塞序列 (0x55555555) 给到 MII 来通知所有的其它站点发生了一个冲突。如果冲突发生在前导码的发送期间，MAC 会完成前导码和界定符的发送后，再发阻塞序列。

jabber 定时器用于当发送超过 2048 字节（默认）时切断发送过程。在半双工模式中 MAC 使用延迟机制来进行流量管控 。当应用层请求停止接收帧时，只要启用了流量管控，当 MAC 感受到帧的接收时就会发送 32 字节的阻塞序列。这会导致一个冲突并且远程工作站会后退。应用通过 ETH_MACFCR 寄存器的 BPA 位来请求流量控制。如果应用请求发送帧，则即使激活被压时也会对其进行调度和传输。注意，如果反压保持激活很长一段时间（并且发生超过 16 个连续的冲突事件），则远程站由于过度碰撞而中止其传输。如果为发送帧启用 IEEE 1588 时间戳，则当界定符被置于发送 MII 总线上时，该块会获取系统时间的快照

- 发送调度程序

MAC 负责在 MII 接口上进行帧发送的调度。它保持两个发送帧之间的间隙，并遵循半双工模式的截断二进制指数退避算法。MAC 在满足 IFG 和退避延迟后开始发送。它保持在两个任意发送帧之间，配置一段空闲的周期（ETH-MACCR 寄存器中的 IFG 位）。如果帧在 IFG 时间内就到达时，MII 会等待 MAC 发出的使能信号才会开始发送该帧。一旦 MII 的载波信号变为非活动状态，MAC 就启动其 IFG 计数器。当配置 IFG 的值后，在全双工模式下 MAC 就使能发送。在半双工模式中，当 IFG 被配置为 96bit 对应的时间，MAC 遵循 IEEE 802.3 规范种 4.2.3.2.1 章节的规则。如果在在 IFG 间隔的前三分之二检测到载波，MAC 会复位 IFG 计数器。如果是在最后的三分之一检测到的，MAC 会让 IG 计数器继续计数，并且在 IFG 间隔时间走完后使能发送器。MAC 在半双工模式下会实现截断的二进制指数退避算法。

■ 发送流量控制

当发送流量控制使能（ETH_MACFCR 的 TFE 位）位被置 1 后，在全双工模式中，MAC 会生成暂停帧并发送。Pause 帧会附加计算出的 CRC，然后发送。暂停帧生成可以通过两种方式启动：1.当 ETH_MACFCR 的 FCB 位置 1　2.当接收 FIFO 满了

— 如果应用层通过将 ETH_MACFCR 的 FCB 位置 1 来请求流量控制，MAC 生成并发送单个暂停帧。暂停帧中暂停时间的值是由 ETH_MACFCR 寄存器来配置的。如果要延迟或结束先前传输的暂停帧中设定的时间，应用层需要在设置新的暂停时间值（ERH_MACFCR 的 PT）后请求另一个暂停帧的发送 。

— 如果接收 FIFO 满，的时候，应用层请求了流量控制，MAC 也会生成并发送暂停帧。如果在该暂停时间用完之前，接收 FIFO 在可配置的时隙时间内依旧保持满，则发送第二个暂停帧。只要 FIFO 一直为满，该过程会一直重复。如果该状态不再满足了，MAC 会发送零暂停时间的暂停帧向远端指示接收 buffer 准备接收新的数据了。

■ 单个包发送操作

常规的发送序列：

— 如果系统有数据需要发送，DMA 控制器通过 AHB 主接口从内存中将他们取出来并且开始转发到 FIFO 中。FIFO 持续的接收数据，直到接收到完整的帧。

— 当超过阈值或者一个完整的数据包被 FIFO 接收了，帧数据会被发送到 MAC 控制器。DMA 持续地从 FIFO 中发送数据到 MAC。当完成一个帧地传输，DMA 控制器会接收到 MAC 的状态通知。

■ 两个包发送操作

— 因为 DMA 在将描述符释放到主机之前必须更新描述符状态，那么可能最多有两个帧发送到 FIFO。仅当 OSF 位置 1，DMA 会将第二个帧取出来放入到 FIFO。如果 OSF 未置 1，只有在 MAC 已经完全处理了一个帧并且 DMA 已经释放了描述符之后，才从存储器中提取下一帧

— 如果 OSF 位置 1，DMA 在完成第一个帧的发送到 FIFO 后立即开始取第二个帧。它不会等待状态的更新。与此同时，当第一个帧被发送时，第二个帧会被 FIFO 接收。一旦第一个帧发送完并且 MAC 返回了状态信息，会被推入到 DMA。如果此时 DMA 已经完成向 FIFO 发送第二个数据包，第二次的帧数据发送必须等待第一次帧发送的状态。

■ 冲突时的重发机制

在半双工模式下，当一个帧被发送到 MAC，MAC 接口可能会产生一个冲突事件。MAC 甚至在接收到帧结束之前就给出指示重发的状态。然后重新发送被使能，帧从 FIFO 中弹出。当超过 96 字节被转发到 MAC 控制器时，FIFO 控制器会释放该空间并使其可用于 DMA 以推送更多数据。这意味着当发送超过 96 字节后或者当 MAC 控制器指示了延迟冲突事件时，就无法再重发了。

■ 发送FIFO清空操作

可以使用工作模式寄存器的 bit20 进行软件清空发送 FIFO。清空操作是立即行为，即使 Tx FIFO 正在发送帧到 MAC 控制器，也会被情况，并且对应的指针被复位到初始位置。这会导致 MAC 发送器发生下溢事件，并且终止发送当前帧，该帧的状态信息会包含下溢和发生帧清空事件（TDES0 的 bit13 和 bit1）。在清空过程中，DMA 不会给 FIFO 发数据。包含清空帧数的发送状态字会被传给应用程序。被完全清除的帧会帧清除状态位置 1（TDES0）。当应用程序（DMA）接收到清除帧所有的状态字后，清除操作完成。发送 FIFO 清空控制寄存器清零。从此时开始，应用层发出的新的帧被接收。在清除操作之后，所有传输过程中的数据将被丢弃，除非是以 SOF 开始的。

■ 发送状态字

在发送给 MAC 控制器的帧结尾和控制器完成了帧的发送时，发送状态会返回给应用程序。状态的详细的描述与 TDES0 的 bit[17:0]一致。如果 IEEE 1588 时间戳被使能了，发送状态和 64 位的时间戳也都会被返回。

■ 发送校验和减载

例如 TCP 和 UDP 的通信协议都有校验和字段，可以确定通过网络传输的数据的完整性。因为以太网最广泛的用途是通过 IP 字段封装 TCP 和 UDP。以太网控制器有发送校验和减载的特性，用于支持校验和计算并插入到发送中去，在接收时可以检测错误。该章节介绍发送帧的校验和减载特性

注：TCP、UDP 和 ICMP 校验和的计算都是基于完整的帧，然后把计算结果插入到对应的头部区域。基于这种特性要求，仅当发送 FIFO 被配置为存储转发（ETH_DMAOMR 寄存器的 TSF 位为 1）的模式时该功能才有效。如果配置为阈值模式，发送校验和减载会被旁路。

必须要确保 FIFO 的深度能够存储一个完整的帧。如果 FIFO 深度不够，有效载荷（TCP/UDP/ICMP）的校验和插入功能会被旁路。即使在存储转发模式中，仅帧的 IPv4 头校验和可以修改。

发送校验和减载支持两种类型的校验和计算以及插入，可以通过设置 CIC 位来控制每帧的校验和（发送描述符 TDES1 的 bit28:27）

IPv4、TCP、UDP、ICMP、IPv6 和 ICMPv6 数据包报头规范分别参见 IETF 规范 RFC 791、RFC 793、RFC 768、RFC 792、RFC 2460 和 RFC 4443

— IP 报头校验和

在 IPv4 字段中，报头字段的完整性由 16 位报头校验和字段（IPv4 数据报的第十一和第十二字节）表示。当以太网帧类型的值为 0x0800 并且 IP 字段版本为 0x4，校验和减载就会检测 IPv4 字段。输入帧的校验和字段在计算过程中被忽略，并由计算值替换。IPv6 报头没有校验和字段，因此校验和减载不会修改 IPv6 报头字段。IP 报头校验和计算结果通过发送状态中的 IP 报头错误状态位（bit16）来指示，当以太网帧类型字段和 IP 报头版本字段的值不一致时，或者当以太网帧没有足够的数据时（IP 报头长度字段来显示的），该状态位就会被置位。换一句话来说，即在以下情况时 IP 报头错误被断言，该位会置 1

1. IPv4 字段
   a) 接收的以太网类型为 0x0800，但是 IP 报头版本字段不等于 0x4
   b) IPv4 报头长度字段显示的值小于 0x5（20 字节）
   c) 帧总长度小于 IPv4 报头长度字段的值
2. IPv6 字段

a) 以太网类型为 0x86dd，但 IP 报头版本字段不等于 0x6

b) 在完全接收到 IPv6 报头 (40 字节) 和扩展报头 (扩展报头中对应的报头长度字段所示) 之前，帧结束了。如果以太网类型字段显示是 IPv4 的有效载荷，即使校验和减载检测到 IP 报头错误，它也会插入 IPv4 报头校验和。

— TCP/UDP/ICMP 校验和

TCP/UDP/ICMP 校验和处理 IPv4 和 IPv6 报头（包括扩展报头）并确定封装的有效载荷是否是 TCP、UDP 或 ICMP

注意：

1. 对于非 TCP/UDP/ICMP/ICMP v6 有效载荷，校验和会被旁路并且在帧中不会进行修改
2. 分段的 IP 帧（IPv4 或 IPv6），具有安全功能（例如身份验证报头或封装的安全有效载荷）的 IP 帧以及具有路由标头的 IPv6 帧被会旁路，不由校验和处理。

校验和会计算 TCP/UDP/ICMP 有效载荷并插入到报头中对应的字段，工作有两种模式：

1. 第一种模式：TCP、UDP 或 ICMP v6 伪报头不包含在校验和计算中，并且假定存在于输入帧的校验和字段中。校验和字段被包含在校验和计算中，被最终的校验和计算值替代。
2. 第二种模式：校验和字段被忽略，TCP、UDP 或 ICMP v6 伪报头数据被会包含在校验和计算中，校验和字段会被最终的计算值覆盖。

要注意的是，对于对于 ICMP-over-IPv4 的数据包，ICMP 数据包中的校验和字段必须在这两种模式下总是等于 0x0000，因为此类数据包中未定义伪报头。如果不等于 0x0000，则可能会在数据包中插入不正确的校验和。

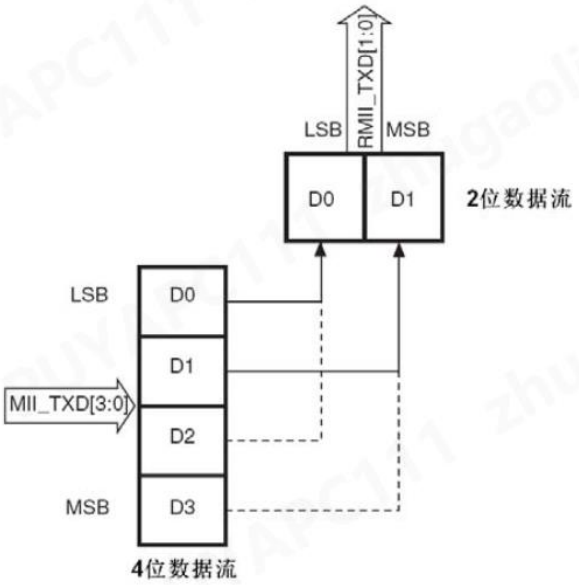
校验和结果通过有效载荷校验和错误状态位来展示，在发送状态向量的 bit12。当以下任意情况发送时置位：

1. 在存储转发模式中，帧已开始转发给 MAC 发送器，而帧的结尾未写入到 FIFO
2. 在接收到的 IP 报头中，在有效载荷长度字段所对应的字节长度前，包已结束。

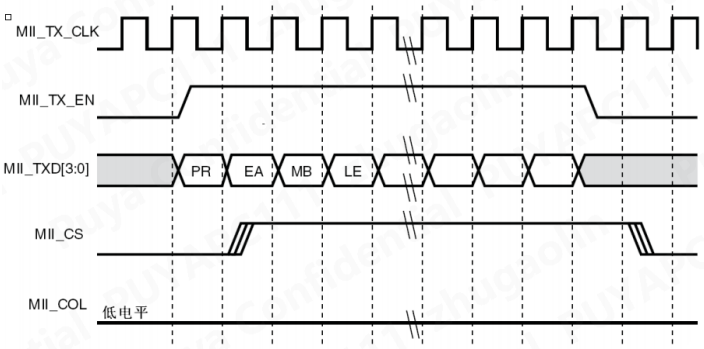
当包超过有效载荷长度字段所对应的字节长度，字节被忽略为填充字节，并且不会报错。当第一种类型的错误被检测到，报头不会被修改。对于第二种错误类型，校验和计算结果仍然会被插入到报头的对应区域。

■ MII/RMII发送bit顺序

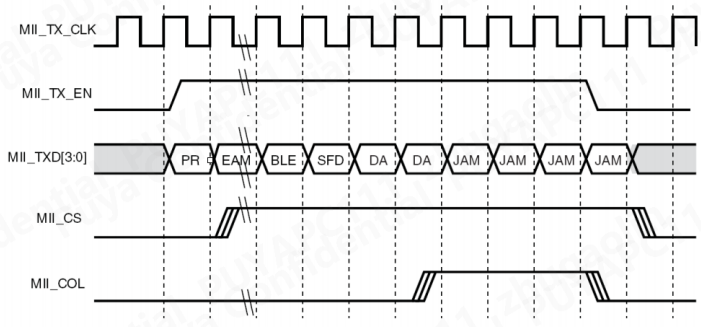
先发送低位，再发送高位



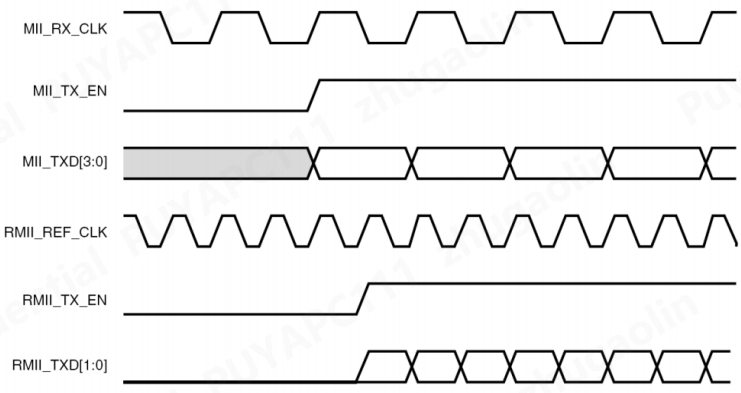
无冲突时的传输：



有冲突时的传输：



MII 和 RMII 模式下的帧传输：



33.4.3 MAC 帧接收

MAC 接收帧后再推入到 Rx FIFO 中，一旦 FIFO 超过配置的接收阈值（ETH_DMAOMR 寄存器的 RTC 位），就会向 DMA 发出状态（填充水平）。以便于 DMA 可以向 AHB 接口发起预先配置的突发传输。在默认的直通模式中，当 64 字节（ETH_DMAOMR 寄存器的 RTC 位）或一个完整的包数据被 FIFO 接收后，数据会弹出并且 DMA 会被通知其可用性。当 DMA 已初始化 AHB 接口传输，该数据就会从 FIFO 中持续地传输直到一个完整的包被发送。一旦 EOF 帧被发送完，状态字被弹出并被发送到 DMA 控制器。

在 Rx FIFO 存储转发模式中（ETH_DMAOMR 寄存器的 RSF 位），仅在帧被完全写入到接收 FIFO 中才可读出来。在该模式下，所有的错误帧被丢弃（如果配置了），仅有效的帧可被读出来并转发给应用程序。而在直通模式中，部分错误帧不会被丢弃，因为在帧尾时错误状态才会被接收，此时帧的开始已经被 FIFO 读取。

当 MAC 在 MII 接口上检测到界定符时，接收操作就会被初始化。MAC 控制器在处理帧之前会剥离前导码和界定符。报头字段会被检查进行过滤并且其 FCS 字段会进行 CRC 校验。如果地址过滤失败的话帧会被控制器丢弃。

◆ 接收协议

被接收帧的前导码和界定符均会被剥离，一旦界定符被检测到，MAC 开始发送以太网帧给 Rx FIFO，最开始的是界定符后的字节（目的地址）。如果 IEEE 1588 时间戳被使能了，检测到界定符时会对系统时间进行一次拍照。除非 MAC 过滤并丢弃了该帧，否则时间戳就会被发送到应用程序。

如果接收帧的长度/类型字段小于 0x600，并且如果 MAC 被配置为自动 CRC/PAD。MAC 将帧的数据发送到 RxFIFO，直到长度/类型字段中指定的计数，然后开始丢弃字节（包括界定符字段）。如果长度/类型字段大于或等于 0x600，NAC 会发送所有接收到的帧数据给 Rx FIFO，无论是否有配置了自动 CRC 功能。MAC 看门狗默认使能，帧数据大于 2048 字节（目的地址+源地址+长度/类型+数据+填充字节+帧校验序列）的部分均切断。当关闭看门狗功能时该特性被关闭。然后即使关闭了看门狗功能，大小大于 16KB 的帧也会被切断并且给出看门狗超时状态。

◆ 接收CRC:自动填充和CRC剥离

MAC 会对接收帧进行 CRC 错误检查。它计算 32 位的 CRC，计算范围为目的地址到帧校验序列的字段部分。编码算法：

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

无论是否剥离自动填充和 CRC，MAC 都会对接收的完整帧进行 CRC 检查计算。

◆ 接收校验和减载

在以太网接收帧中，IPv4 和 IPv6 帧均会被检测，并且针对数据完整性进行处理。可以通过设置 ETH_MACCR 寄存器的 IPCO 位来使能接收校验和减载。MAC 接收器通过判断帧类型字段的值是否等于 0x0800 和 0x86DD 来识别 IPv4 和 IPv6 帧。该方法也可用于识别带 VLAN 标签的帧。接收校验和减载会计算 IPv4 的报头并检查其是否匹配接收到的 IPv4 报头校验和。如果有效载荷类型（以太网类型字段）和 IP 报头版本不匹配，或者接收帧没有足够的字节，由 IPv4 报头的长度字段指示（或者当 IPv4 或

IPv6 报头中可用的字节少于 20 个字节）不匹配，IP 报头错误都会置位。接收校验和减载也会用于在 IP 数据段中识别 TCP、UDP 和 ICMP 有效载荷并计算其校验和。它包括用于校验和计算的 TCP/UDP/ICMPv6 伪标头字节，并检查接收到的校验和字段是否与计算值匹配。匹配结果会通过接收状态字中的有效载荷校验和错误位

指示出来。如果 TCP/UDP/ICMP 有效载荷长度与 IP 报头给出的期望有效载荷长度不匹配时，错误位也会置位。如 TCP/UDP/ICMP 校验章节中提到的，接收校验和减载会绕过分段 IP 数据报的有效载荷，具有安全功能的 IP 数据报、IPv6 路由标头以及 TCP、UDP 或 ICMP 以外的有效载荷。这些信息会在接收状态中体现，具体见 RDES0（接收描述符字 0）章节。在这种配置中，控制器不会向接收到的以太网帧附加任何有效负载校验和字节。

Bit 18: Ethernet frame	Bit 27: Header checksum error	Bit 28: Payload checksum error	Frame status
0	0	0	The frame is an IEEE 802.3 frame (Length field value is less than 0x0600).
1	0	0	IPv4/IPv6 Type frame in which no checksum error is detected.
1	0	1	IPv4/IPv6 Type frame in which a payload checksum error (as described for PCE) is detected
1	1	0	IPv4/IPv6 Type frame in which IP header checksum error (as described for IPCO HCE) is detected.
1	1	1	IPv4/IPv6 Type frame in which both PCE and IPCO HCE are detected.
0	0	1	IPv4/IPv6 Type frame in which there is no IP HCE and the payload check is bypassed due to unsupported payload.
0	1	1	Type frame which is neither IPv4 or IPv6 (checksum offload bypasses the checksum check completely)
0	1	0	Reserved

◆ 接收帧控制器

如果 MAC CSR 帧过滤寄存器中的 RA 位被复位了，MAC 会基于目的地址和源地址进行帧过滤（如果应用程序决定不接收任何坏帧（如 runt、CRC 错误帧等），则它仍然需要执行另一级别的过滤。）。当检测到过滤失败，该帧会被丢弃不 i 会发送到应用程序。当过滤地址寄存器动态的改变时，如果过滤失败（DA-SA），帧的其余部分被丢弃，并且 Rx 状态字被立即更新（零帧长度、CRC 错误和短小错误位置位），用于显示过滤失败。在以太网低功耗模式下，所有帧均被丢弃，也不会转发给应用程序。

◆ 接收流量控制

MAC 会检测接收暂停帧并按照暂停帧中规定的延迟时间暂停帧的传输（仅全双工模式）。可以通过 ETH_MACFCR 寄存器中的 RFCE 位使能暂停帧检测功能。当接收流量控制使能后，接收帧的目的地址会一直被监测是否与控制帧的多播地址相匹配（0x0180 C200 0001）.如果检测到匹配了（接收帧的目的地址与保留的控制帧目的地址匹配），MAC 会通过 ETH_MACFFR 的 PCF 位选择是否发送该接收的控制帧给应用程序。

MAC 还会解码接收控制帧的类型、操作码、暂停定时器时间、如果状态的字节计数器显示为 64 字节，并且如果没有 CRC 错误，MAC 发送器会在解码得到的暂停时间（暂停值乘上 slot 时间，1 slot 时间=64

字节对应的传输时间）内暂停任何数据发送。与此同时，如果另一个暂停帧被检测到，并且是一个零暂停时间，MAC 重置暂停时间并管理此新的暂停请求。

如果接收控制帧的类型不是 0x8808、操作码不是 0x00001、字节长度不是 64 字节，即使没有 CRC 错误，MAC 也不会生成暂停。

如果暂停帧是通过多播目的地址发送的，MAC 会通过地址匹配来过滤该帧。

如果暂停帧是通过单播目的地址发送的，MAC 会根据目的地址是否匹配 MAC 地址 0 寄存器以及 ETH_MACFCR 的 UPDF 为是否置 1（检测暂停帧即使是通过单播目的地址传输）来选择过滤该帧。

PCF 寄存器位（ETH_MACFFR 中的位[7:6]）除了地址过滤之外，还控制控制帧的过滤。

◆ 接收操作多帧处理

因为当数据发送完后，状态立即会指示，FIFO 可以存储任意数量的帧只要它不是满的。

◆ 错误处理

如果 Rx FIFO 在从 MAC 中接收到帧尾数据前就满了时，就会产生上溢，并且整个帧都会被丢掉，ETH_DMAMFBOCR 寄存器中的上溢计数器会累加。由于上溢了，状态会指示出这是一个部分帧。如果使能了 ETH_DMAOMR 的 FEF 和 FUGF 位，RX FIFO 可以过滤错误和小尺寸的帧。

如果接收 FIFO 被配置位存储转发模式，所有的错误帧都会被过滤且丢弃。

在直通模式下，如果从 Rx FIFO 读取帧的 SOF 时帧的状态和长度可用，则可以丢弃整个错误帧。DMA 可以通过使能接收帧的清空位，实现清除从 FIFO 读取的错误帧。发送给应用的数据会被停止，帧的其余部分在内部读取并丢弃 MAC 中断。然后可以开始下一帧传输（如果可用）

◆ 接收状态字

以太网帧接收的结束时，MAC 输出接收状态给应用程序（DMA）。接收状态的详细描述见 RDES0

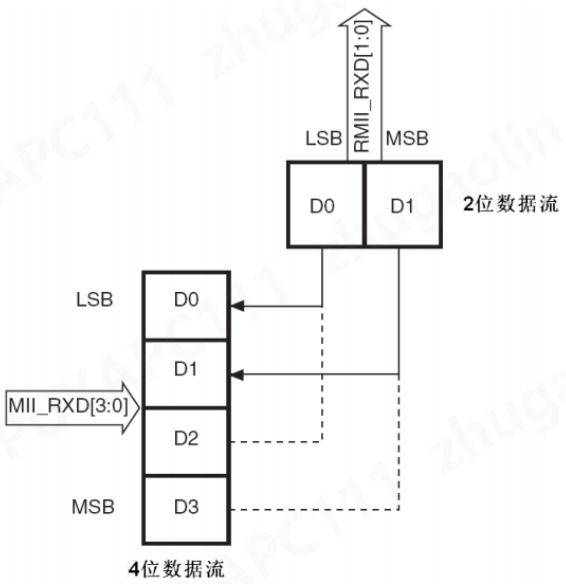
◆ 帧长度接口

在交换机应用程序的情况下，应用程序和 MAC 之间的数据传输和接收以完整的帧传输发生。应用层应该知道从入口端口接收的帧的长度，以便将帧传输到出口端口。MAC 控制器在每个帧接收的结尾将帧长度信息放置在状态里。

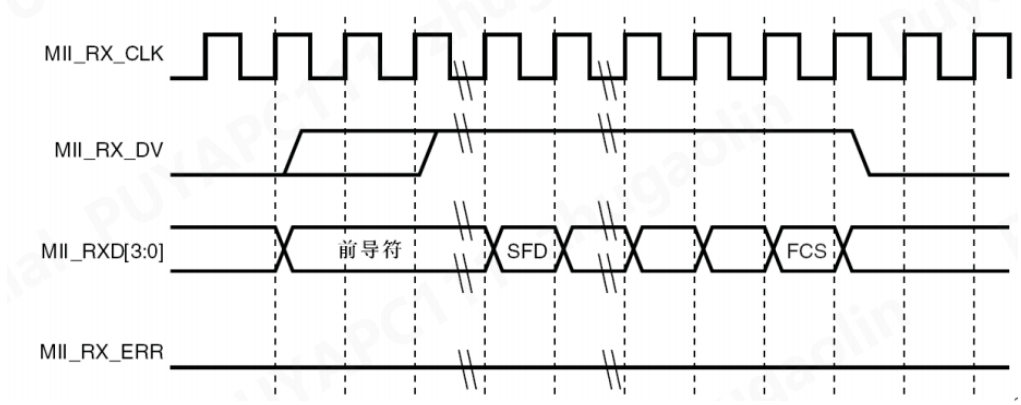
注意：由于溢出，写入 Rx FIFO 的部分帧的帧长度值为 0

◆ MII/RMII接收位顺序

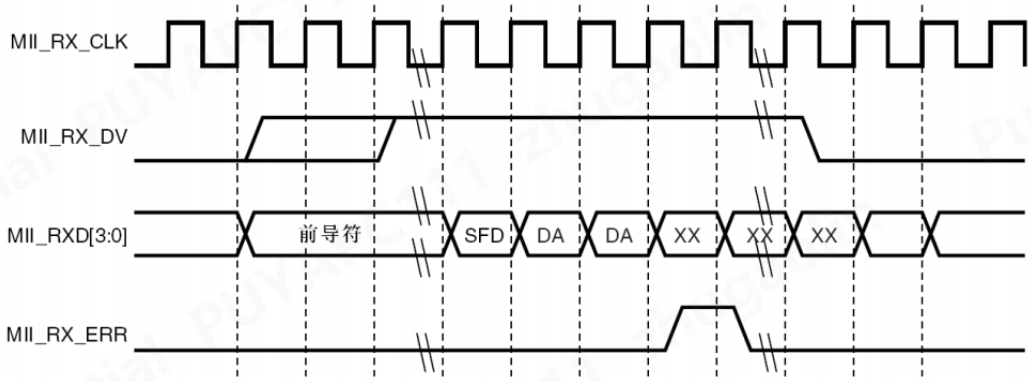
先接收低位，再接收高位



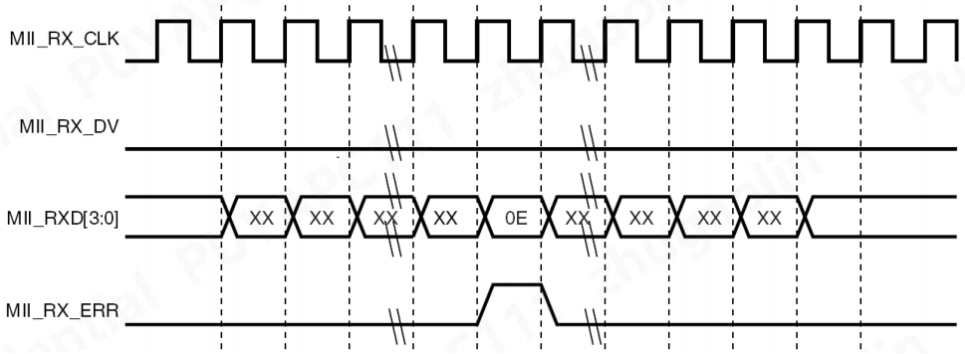
接收时无错误：



接收时有错误：



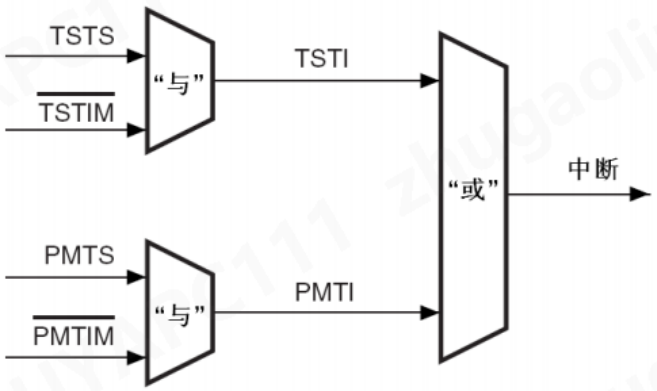
接收时有假载波：



33.4.4 MAC 中断

由于各种事件，MAC 核心可能会产生中断。ETH_MACSR 寄存器描述了哪些时间会引起 MAC 中断。可以通过在中断屏蔽寄存器中设置相应的屏蔽位来防止每个事件断言中断。

中断寄存器位仅指示事件报告的集合。需要读取对应的状态寄存器以及其它寄存器来清除中断。例如，中断寄存器的 bit3 置 1 了，表示在低功耗模式下接收到了魔法包和唤醒帧。需要读取 ETH_MACPMTCSR 寄存器来清除中断事件



33.4.5 MAC 过滤

◆ 地址过滤

地址过滤在所有的接收帧中检查目的地址和源地址，并且报告地址过滤状态。地址检查是基于应用层选择的不同的地址参数（帧过滤寄存器）。过滤后的帧也可以被识别：多播或广播帧。

地址过滤通过站点的物理地址和多播哈希表来进行地址检查

◆ 单播目的地址过滤

MAC 支持最多 4 个 MAC 地址的单播完美过滤，如果完美过滤被选择(帧过滤寄存器的 HU 位被复位)，MAC 会比较 48 位接收到的单播地址和配置的 MAC 地址是否匹配。默认 MAC Addr0 是使能的，其它的地址 MAC Addr1-3 可以分别通过一个使能位控制。MAC Addr1-3 的每个字节在进行目的地址比较时都可以被屏蔽比较，只需要设定对应的字节屏蔽控制位即可。这有助于目的地址的组地址过滤。在哈希过滤模式中 (HU 置位)，MAC 会使用 64 位的哈希表对单播地址进行非完美过滤。对于哈希过滤，MAC 使用接收到的目的地地址的 6 位的高位 CRC（参见下面的注释 1）来索引哈希表的内容。值为 000000

的选择哈希表寄存器中的 bit0，值为 111111 的选择哈希表寄存器中的 bit63。如果对应的位（由 6 位 CRC 决定）被置为 1 了，表示单播帧通过了哈希过滤，否则失败。

注释：CRC 是 32 位值的算法

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

◆ 多播目的地址过滤

MAC 可以通过配置帧过滤寄存器中的 PAM 位通过所有多播帧，如果 PAM 没置位，则 MAC 会基于 HM 位对多播地址进行过滤。在完美过滤模式中，多播帧的地址会与配置过的 MAC 目的地址 1-3 寄存器进行匹配。组地址过滤也是支持的。在哈希过滤模式中，MAC 使用 64 位哈希表进行非完美过滤。对于哈希过滤，MAC 使用接收到的目的地地址的 6 位的高位 CRC (参见下面的注释 1) 来索引哈希表的内容。值为 000000 的选择哈希表寄存器中的 bit0，值为 111111 的选择哈希表寄存器中的 bit63。如果对应的位（由 6 位 CRC 决定）被置为 1 了，表示单播帧通过了哈希过滤，否则失败。

◆ 哈希或完美地址过滤

通过设置帧过滤寄存器中的 HPF，当目的地址通过哈希过滤或者完美过滤即可认为帧通过了。如果 HPF=0，那么则支持一种过滤。设置对应的 HU 和 HM 位来实现在单播和多播帧的过滤中选择哪种过滤方式。

◆ 广播地址过滤

MAC 默认情况下不会对广播地址进行过滤，然而，如果 MAC 通过 BFD 位配置为拒绝接收广播帧，那么所有的广播帧均会被丢掉。

◆ 单播源地址过滤

MAC 也会对接收帧的源地址进行完美过滤。默认情况下，MAC 对源地址和源地址寄存器的值进行比较。MAC 地址寄存器 1-3 可以配置用于源地址比较，而不是目的地址比较，这是通过各寄存器的 bit30 来进行选择的。源地址的组过滤也同样支持。当帧过滤寄存器中的 SAF 位置 1 时，如果接收帧未通过源地址过滤则会被丢弃。否则，源地址过滤的结果会产生一个状态位放置在接收状态字中（参考接收描述符字 0，RDES0）

当 SAD 置 1，源地址和目的地址的过滤结果会与逻辑在一起，然后决定是否进行帧转发。这意味着只要有一个失败了，帧就被丢弃了。两个地址都过滤成功则会将帧转发给应用程序。

◆ 反向过滤操作

对于目的地址和源地址的过滤。有一种选择是对过滤匹配结果进行反向的操作再输出。这个操作是通过配置帧过滤寄存器的 DAIF 和 SAIF 位。DAIF 位适用于单播或多播帧的目的地址过滤反向。SAIF 适用于单播帧的源地址过滤反向。

下图为不同寄存器配置时，源地址和目的地址过滤总结。

Frame type	PM	HPF	HU	DAIF	HM	PAM	DB	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all frames
	0	X	0	0	X	X	X	Pass on perfect/group filter match
	0	X	0	1	X	X	X	Fail on perfect/Group filter match
	0	0	1	0	X	X	X	Pass on hash filter match
	0	0	1	1	X	X	X	Fail on hash filter match
	0	1	1	0	X	X	X	Pass on hash or perfect/Group filter match
	0	1	1	1	X	X	X	Fail on hash or perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all frames
	X	X	X	X	X	1	X	Pass all frames
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x
	0	0	X	0	1	0	X	Pass on hash filter match and drop PAUSE control frames if PCF = 0x
	0	1	X	0	1	0	X	Pass on hash or perfect/Group filter match and drop PAUSE control frames if PCF = 0x
	0	X	X	1	0	0	X	Fail on perfect/Group filter match and drop PAUSE control frames if PCF = 0x
	0	0	X	1	1	0	X	Fail on hash filter match and drop PAUSE control frames if PCF = 0x
	0	1	X	1	1	0	X	Fail on hash or perfect/Group filter match and drop PAUSE control frames if PCF = 0x

Frame type	PM	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all frames
	0	0	0	Pass status on perfect/Group filter match but do not drop frames that fail
	0	1	0	Fail status on perfect/group filter match but do not drop frame
	0	0	1	Pass on perfect/group filter match and drop frames that fail
	0	1	1	Fail on perfect/group filter match and drop frames that fail

33.4.6 MAC 回环模式

MAC 支持发送帧到自己的接收器的回环模式。默认该模式不使能，可通过 ETH_MACCR 寄存器的回环控制位使能。

33.4.7 MAC 管理计数器：MMC

MAC 管理计数器（MMC）包含了一组寄存器来收集关于接收和发送帧的的统计信息。包括用于控制寄存器行为的控制寄存器，2 个 32 位的寄存器包含产生的中断，2 个 32 位的寄存器包含对应的中断屏蔽。这些寄存器可以通过应用层访问，均为 32 位宽度

当帧通过地址过滤后接收端 MMC 计数器会更新，丢弃的帧统计不会更新，除非丢弃的帧是小于 6 个字节的短帧（未完全接收到 DA 字节）

- 好的发送帧和接收帧

如果发送器发送成功则发送的帧被认为是好的，换而言之，如果帧传输过程中如果没有因为以下错误而中止，则被认为是好帧：

- JABBER 定时器溢出
- 无载波/载波丢失
- 延迟冲突
- 短帧
- 过度延迟
- 过度冲突

如果以下错误未发送，则接收帧可认为为好帧

- CRC 错误
- 短帧（小于 64 字节）
- 对齐错误（仅在 10/100Mbit，帧最后不是一个完整字节）
- 长度错误（仅非典型帧）
- 超出范围（仅非典型帧，超过最大宽度）
- MII_RXER 输入错误

最大帧宽度取决于帧类型：

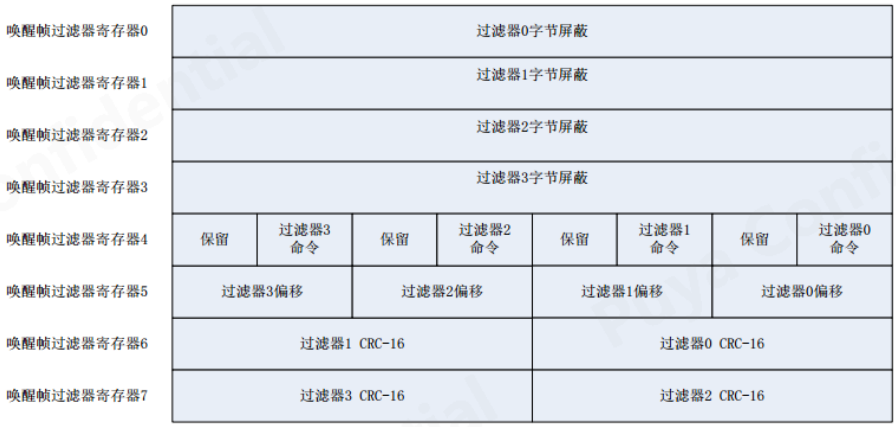
- 非标签帧最大尺寸=1518
- 标签帧最大尺寸=1522

33.4.8 电源管理：PMT

本章节描述的是 MAC 支持的电源管理机制 PMT。PMT 支持接收远程网络的唤醒帧和魔法包帧。PMT 当接收这两个帧时会生成中断。当远程唤醒帧使能且魔法包使能时，PMT 模块使能，使能控制位为 ETH_MACPMTCSR 寄存器中的 WFE 和 MPE。当进入到低功耗模式时，所有的接收帧均被 MAC 丢弃，也不会转发到应用程序。当检测到远程网络的唤醒帧或魔法包时，退出低功耗模式。

◆ 远程唤醒帧过滤寄存器

唤醒帧过滤器寄存器一共有 8 个，但这些寄存器共用一个相同的偏移地址。在完成对某个过滤器寄存器的读或写的时候，内部的指针会自动指到下一个过滤器寄存器。不论是读还是写操作，需连续进行 8 次操作。也就是说，对其设置时需要将设置的值分为 8 次逐一写入唤醒帧过滤器寄存器地址，读取的时候也是需要连续读 8 次唤醒帧过滤器寄存器，才能将所有值读出。



■ 过滤器 n 字节屏蔽

该寄存器定义了过滤器 n（n=0，1，2，3）使用帧的哪些字节来检查判断是否为唤醒帧。其第 31 位必须为'0',位[30:0]是字节屏蔽位。如果过滤器 n（n=0，1，2，3）的第 m 位（m=0~30）为'1'，则唤醒帧检测的 CRC 模块会处理输入帧的第[过滤器 n 偏移+m]字节，否则忽略之。

■ 过滤器 n 命令

共 4 位控制过滤器 n 的工作模式。最高位位 3 为地址类型选择，如果该位为'1'，则只检测多播帧；如果该位为'0'，则只检测单播帧。位 2 和位 1 必须保持为 0。位 0 是过滤器 n 的使能位，置位时使能过滤器 n，反之禁能过滤器 n。

■ 过滤器 n 偏移

与过滤器 n 字节屏蔽配合使用。该寄存器定义了过滤器 n 要检查的首字节在帧内的偏移量。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 n CRC-16

该寄存器包含了预先写入的 CRC-16 码，用于与帧数据计算的 CRC-16 值进行比较。

◆ 远程唤醒帧过滤检测

当 MAC 在深度睡眠状态并且 ETH_MACPMTCSR 寄存器的远程唤醒位被使能，在接收到远程唤醒帧之后恢复正常操作。应用程序会写 8 个唤醒过滤寄存器，通过连续对唤醒帧过滤寄存器地址写 8 次。应用程序通过将 ETH_MACPMTCSR 寄存器的 bit2 写 1 使能远程唤醒功能。PMT 支持四个可编程过滤器，它们提供不同的接收帧序列模式，如果接收帧通过了地址过滤，并且如果 CRC-16 与传入的检测序列一致，那么唤醒帧将被接收。过滤器偏移（最小值 12，表示帧的第 13 个字节）决定了被检测帧的偏移量。过滤字节屏蔽决定了帧的哪个字节必须被检测。字节屏蔽的第 31 位必须为 0。唤醒帧只会检查其长度错误，FCS 错误，bit 丢失错误，MII 错误，冲突，并且需要确保不是短小帧。即使唤醒帧超过了 512 字节长度，如果帧具有有效的 CRC 值，那么该帧还是被认为是有效的。在每个远程唤醒帧的接收中，唤醒帧的检测结果都会在 ETH_MACPMTCSR 寄存器中更新。如果使能了，PMT 的中断就会产生来表示接收了一个远程唤醒帧。

◆ 魔法包检测

另一种唤醒方法是检测 Magic Packet 唤醒帧（见 AMD 公司的"Magic Packet 技术"）。MAC 接收一个特殊的包叫做魔法包，对网络节点进行寻址。只有寻址到设备或广播地址的魔术包才会被检查，以确定它们是否满足唤醒要求。检查通过地址过滤（单播或广播）的魔术数据包，以确定它们是否符合所有字节中的 6 个字节的远程唤醒 LAN 数据格式，然后出现 16 次 MAC 地址。通过向 ETH_MACPMTCSR 的 bit1 写 1 来使能检测。PMT 模块持续地监测帧是否有魔法包序列，在目的地和源地址字段之后检查每个接收帧的 0xFFFF FFFF FFFF 序列。PMT 块然后在没有任何中断或中断的情况下针对 MAC 地址的 16 次重复来检查帧。如果地址的 16 次重复中出现中断，则在传入帧中再次扫描 0xFFFF FFFF FFFF 序列。16 次重复可以在帧中的任何位置，但必须在同步流之前。只要检测到 MAC 地址出现 16 次重复，设备也会接收多播帧。例如，MAC 的节点地址是 0x0011 2233 4455，那么 MAC 接下来就会去检测如下的数据序列：

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

当接收到魔法包时，魔法包的检测结果会在 ETH_MACPMTSCR 中更新，如果使能了，PMT 中断会产生来表示接收到了魔法包。

■ 低功耗期间系统的注意事项

在 MCU 处于深度睡眠模式时，若使能外部中断线 19，则以太网的 PMT 模块仍能够检测帧。

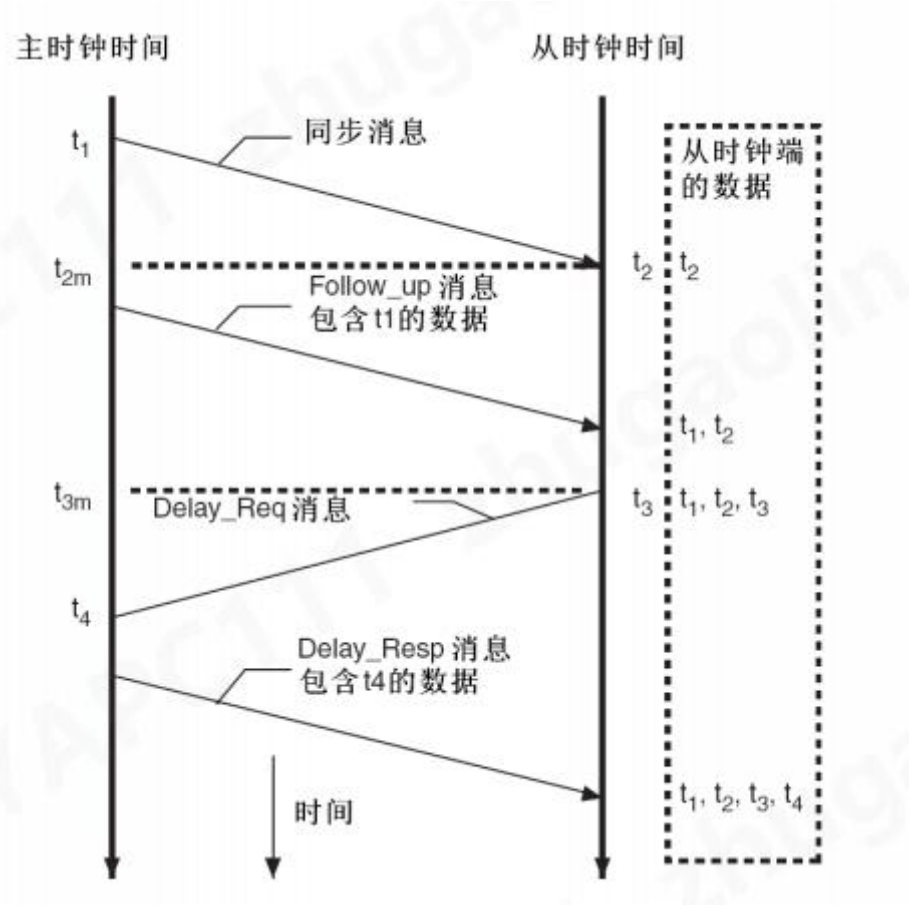
MAC 接收器的状态机在低功耗模式下需要一直保持使能，这意味着 ETH_MACCR 寄存器的 RE 位一直要置位，因为它涉及到魔法包/唤醒帧的检测。但是，在低功耗模式期间，应通过清除 ETH_MACCR 寄存器中的 TE 位来关闭发送状态机。并且 DMA 也需要关闭，因为没有必要将魔法包/唤醒帧的数据发给 SRAM。为了关闭 DMA，需要复位 ETH_DMAOMR 寄存器中的 ST 位和 SR 位

推荐的进入低功耗和唤醒步骤如下：

1. 关闭发送 DMA 然后等待之前的帧完全传输完成，可以通过 ETH_DMASR 的 bit[0]来判断；
2. 通过清除 ETH_MACCR 寄存器中的 RE 和 TE 位关闭 MAC 发送器和接收器；
3. 等待接收 DMA 读出 Rx FIFO 中的所有帧，然后关闭接收 DMA；
4. 配置使能外部中断线 19 产生事件或者中断，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 的中断标志位；
5. 置位 ETH_MACPMTCSR 中的 MFE/WFE 位使能魔法包/唤醒帧的检测；
6. 置位 ETH_MACPMTCSR 中的 PD 位使能进入 MAC 低功耗模式；
7. 置位 ETH_MACCR 中的 RE 位使能 MAC 接收器；
8. 进入系统 STOP 模式：
9. 当接收到有效的唤醒帧，以太网外设退出低功耗模式；
10. 读 ETH_MACPMTCSR 来清除电源管理事件标准，使能 MAC 发送状态机，使能接收和发送 DMA
11. 配置系统时钟：使能 HSE 并且设置时钟

33.4.9 精确时间协议（IEEE1588 PTP）

IEEE 1588 标准定义了一种协议，该协议允许在使用网络通信、本地计算和分布式对象等技术时，在测量和系统控制中可以使用精确的时钟同步。该协议适用于通过支持多播信息的局域网进行通信的系统，包括（但不限于）以太网。该协议用于同步异构系统，包括具有不同固有精度、分辨率和稳定性的时钟。该协议支持亚微秒范围内的全系统同步精度，网络和本地时钟计算资料最少。这种基于消息的协议称为时间精度协议。系统和网络被分类为主节点和从节点，用于分发定时/时钟信息。通过交换 PTP 信息来同步主从节点的技术如下图：



1. 主节点广播 PTP 同步消息给所有的节点。同步消息里包含了主设备的时间信息，当该消息离开主系统时的时间为 t1，在以太网端口中，这个时间可以被 MII 捕获到。
2. 从设备接到同步消息也会捕获到精确的时间，t2，通过其时间基准
3. 然后主设备发送 Follow_up 消息，里面包含了 t1 信息供后面使用
4. 从设备发送 Delay_Req 信息给主设备，当帧离开 MII 时记下准确时间 t3
5. 当信息进入到系统时，主设备接收到该消息并捕获到准确的时间 t4
6. 主设备发送 Delay_Resp 消息包含了 t4 信息给从设备
7. 从设备通过四个时间值 t1、t2、t3 和 t4 来同步其本地的时间基准到主设备的时间基准

该协议大多数实现在 UDP 层之上的软件中，然后，如上所述，需要硬件支持来捕捉特定 PTP 数据包在进入和离开 MII 接口时的准确时间。为了正确、高精度地实现 PTP，必须捕获该时间信息并将其返回给软件

33.4.9.1 参考时钟源

为了获取时间的快照，控制器需要 IEEE 1588 规范中定义的 64 位格式的参考时间（分为 2 个 32 位通道，高 32 位以秒为单位提供时间，低 32 位以 ns 为单位指示时间）。PTP 参考时钟输入用于内部生成一个参考时间（也可称为系统时间）并用于捕获时间戳。参考时钟的频率必须大于或者等于时间戳计数器的分辨率。在主节点和从节点之间的同步精度目标为 100ns

系统时间的生成、更新和修改见下一节

精度取决于 PTP 参考时钟的输入周期、OSC 的特性和同步过程的频率。

由于从 Tx 和 Rx 时钟输入域到 PTP 参考时钟域的同步，时间戳的 uncertainty 值被锁存值为 1 个参考时钟周期。如果我们算上分辨率的 uncertainty 值，我们将加上时间戳的一半周期时间

33.4.9.2 具有 PTP 特性发送帧

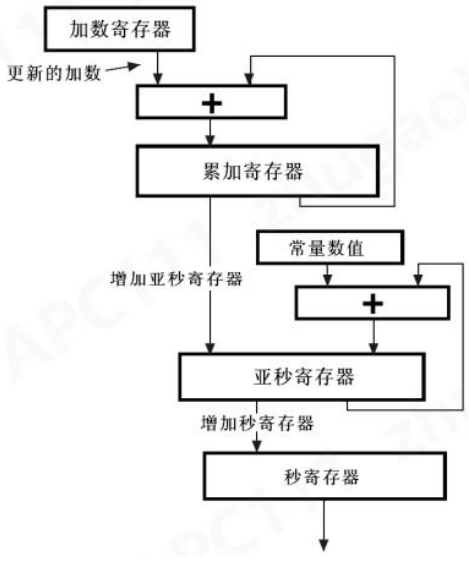
当 MII 接口输出帧的 SFD 时，一个时间戳被捕获。需要被捕获时间戳的帧是可控制的。换句话来说，每个发送帧都可以被标记来指示是否一定需要被捕获时间戳。发送帧不会被处理来识别 PTP 帧。帧的控制是通过发送描述符中的控制位来执行的。捕获的时间戳将以与帧提供状态相同的方式返回给应用程序。时间戳与帧的发送状态一起发送回响应的发送描述符中，从而将时间戳自动地连接到特定 PTP 帧中。64 位时间戳信息被写回到 TDES2 和 TDES3 中，其中 TDES2 保持时间戳的低 32bit 有效位

33.4.9.3 具有 PTP 特性接收帧

当 IEEE 1588 时间戳特性被使能后，以太网 MAC 捕获所有从 MII 接收的帧的时间戳。当帧接收完后，MAC 就提供时间戳。捕获的时间戳将以与帧提供状态相同的方式返回给应用程序。时间戳与帧的接收状态一起发送回响应的接收描述符中。64 位时间戳信息被写回到 RDES2 和 RDES3 中，其中 RDES2 保持时间戳的低 32bit 有效位

33.4.9.4 系统时间校正方法

64 位 PTP 时间是通过 PTP 输入参考时钟（HCLK）进行更新的。PTP 时间用作 MII 接收和发送帧的拍照（对时间戳）源。系统时间计数可通过粗略或精细的校正方式来进行初始化或者校正
在粗略校正方法中，初始化值或者偏移值会被写入到时间戳更新寄存器中。对于初始化，在时间戳更新寄存器中，系统时间计数器会被写入值。而对于系统时间校正，偏移值会与系统时间相加或相减。
在精细校正方法中，通过一段时间内校正从设备时钟（参考时钟）与主设备的频率漂移。与粗略校正方法不同的是，该校正方法不是在单周期完成的。更长的校正时间有助于保持线性时间，并且不会给 PTP 同步信息间隔之间的参考时间带来剧烈抖动（或大的抖动）。在该方法中，一个累加器会将 ADDend 寄存器中的值进行累加如下图，累加器产生的算术结果会被用作系统时间计数器的递增脉冲。该累加器和 ADDend 都是 32 位的寄存器。在这里，累加器充当高精度倍频器和分频器。



系统时间更新逻辑要求 50Mhz 时钟频率来达到 20ns 的精度，分频是参考时钟域要求时钟的比率。因此如果参考时钟（HCLK）是 66Mhz 的话，那么比率就是 66Mhz/50Mhz=1.32。因此。默认的 addend 值应该设置为 2^32/1.32=0xC1F0 7C1

如果参考时钟偏移更小，假如是 65Mhz，那么比率 65/50=1.3，addend 寄存器中的值设置为 2^32/1.3=0xC4EC 4EC4。如果时钟偏移更大，假如是 67Mhz，addend 寄存器中的值应设置为 0xBF0 B7672。如果偏移量为 0，addend 的默认值还是应该被设置为 0xC1F0 7C1（2^32/1.32）

在上图中，用于 Subsecond 寄存器递增的常值为 0d43。这使得系统时间的精度为 20ns（换句话说，它以 20ns 的步长递增）。

软件需要通过同步消息来计算频率的偏移量，然后更新 Addend 寄存器。初始时，在 Addend 中设置从设备时钟为一个频率无补偿值，该值=2^32/ 分频

如果主设备到从设备的延迟假定对于连续的同步消息的是相同的，那么需要应用以下描述的算法。在几个同步周期后，频率值被锁住。从设备时钟可以决定一个主设备到从设备的延迟值，并使用新的值与主设备进行重新同步。

算法如下：

- 在 MasterSyncTime 时，主设备发送一个同步消息给从设备。从设备接收到消息时本地时钟还是 SlaveClockTime，计算 MasterClockTime = MasterSyncTime（n）+ MasterToSlaveDelay（n）
- 主设备时钟计数当前同步周期，MasterClockCount = MasterClockTime（n）– MasterClockTime（n – 1）（假设 MasterToSlaveDelay）与同步周期 n 和 n-1 相同）
- 从设备计数当前同步周期，SlaveClockCount（n）= SlaveClockTime（n）– SlaveClockTime（n – 1）
- 当前同步周期，主设备和从设备时钟计数值的差值为 ClockDiffCount（n）= MasterClockCount（n）– SlaveClockCount（n）
- 从设备时钟的频率比例因子 FreqScaleFactor（n）=（MasterClockCount（n）+ ClockDiffCount（n））/SlaveClockCount（n）
- 那么 Addend 寄存器的频率补偿值 FreqCompensationValue（n）= FreqScaleFactor（n）× FreqCompensationValue（n – 1）

理论上，该算法在一个同步周期内实现锁定；然而，由于网络传播延迟和操作条件的变化，它可能延迟几个周期。

该算法是自校正的：如果从设备时钟相较于主设备来说，初始设置的值不正确的话，该算法会以更多同步周期为代价来校正它。

33.4.9.5 初始化生成系统时间的编程步骤

时间戳特性可以通过配置 ETH_PTPTSCR 寄存器的 bit0 来使能，设置该位后，需要初始化时间戳计数器。合适的步骤如下：

1. 置位 MACIMR 寄存器的 bit9 来屏蔽时间戳触发的中断；
2. 使能时间戳；
3. 基于 PTP 的时钟频率计算出值，再配置亚秒累加寄存器；
4. 如果使用精细校正方法，请对时间戳的加数寄存器进行配置，并配置时间戳控制寄存器的 bit5（加数寄存器更新）；
5. 轮询时间戳控制寄存器直到 bit5 被清除；
6. 置位时间戳控制寄存器的 bit1 选择精细校正方法；
7. 置位时间戳控制寄存器的 bit2（时间戳时间初始化）；
8. 初始化后时间戳计数器开始工作

9. 使能 MAC 接收器和发送器

注意:当清除了 ETH_PTPTSCR 的 bit0 关闭时间戳操作时,上述操作需要重新来一遍重启时间戳操作。

33.4.9.6 粗调校正方法中系统时间的更新步骤

1. 在时间戳更新高寄存器和低寄存器中写入偏移值（正或负）
2. 置位时间戳控制寄存器的 bit3（TSSTU）
3. 当 TSSTU 位被清除时，时间戳更新寄存器中的值与系统时间相加或相减

33.4.9.7 细调校正方法中系统时间的更新步骤

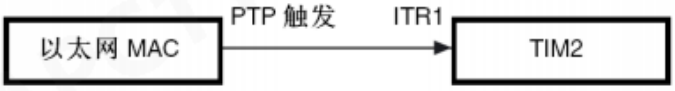
1. 利用前述”系统时间校准方法”介绍的算法，计算出期望的系统时钟频率所对应的加数寄存器的值;
2. 更新时间戳
3. 等待加法寄存器中的期望的新值生效，可以通过当系统到达目标值时触发时间戳中断;
4. 在目标时间高寄存器和低寄存器中配置所需的时间，通过清除 ETH_MACIMR 寄存器中的 bit9 来取消时间戳中断的屏蔽;
5. 置位时间戳控制寄存器的 bit4（TSARU）;
6. 当触发了中断，读取 ETH_MACSR 寄存器;
7. 用旧值重新编程时间戳加数寄存器，并再次设置 ETH_PTSCR 的 bit5

33.4.9.8 PTP 触发内部连接到 TIM2

当系统时间比目标时间快时，MAC 提供一个触发中断。使用中断会引入已知的延迟和命令执行时间的不确定性。

为了避免这种不确定性，当系统时间比目标时间快时，PTP 会输出触发信号。在内部连接到 TIM2 的输入触发上。通过 timer 的输入捕获特性、输出比较特性和 timer 的波形进行分析。如果 timer(PCLK1:TIM2 APB1 时钟）和 PTP 的参考时钟是同步的，则没有不确定的时间了。

PTP 触发信号可通过软件连接到 TIM2 的 ITR1 输入端，通过 AFIO_MAPR 寄存器的 bit29 SYSCFG 中的寄存器控制来使能该连接



33.4.9.9 PTP PPS 输出信号

PTP 脉冲输出用于检查网络中不同节点的同步性，为了测试本地从设备时钟和主参考时钟的差异，可以把主从设备的 PPS（秒脉冲）输出都连接到示波器，以测量 2 个时钟之间的差别。PPS 输出的使能由 SYSCFG 中的寄存器控制

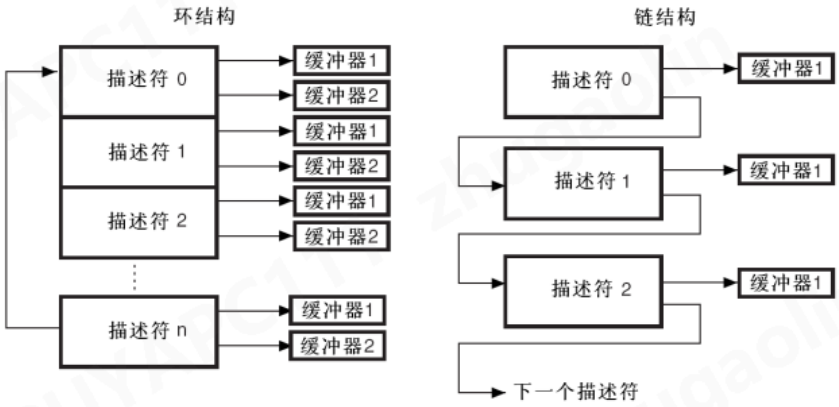


33.5 DMA 控制器

DMA 有独立的发送和接收引擎，以及 CSR 寄存器空间。发送引擎从系统内存中发送数据到 Tx FIFO，接收引擎则从 Rx FIFO 中发送数据到系统内存。控制器利用描述符的方式以最小的 CPU 干预将数据从源高效地移动到目的地 。DMA 是为面向数据包的数据传输而设计的，例如以太网中的帧。控制器可以被编程来实现当帧发送和接收完成时以及其他正常/错误条件的情况下中断 CPU。DMA 与 MCU 控制器的通讯通过两种数据结构

- 控制和状态寄存器（CSR）
- 描述符列表和数据缓存

DMA 将接收的帧数据发送到系统内存的接收缓存中，也可以将接收缓存中的数据发出来。系统内存的描述符是以指向缓存的指针的形式存放的，有两种描述符列表：一个用于接收，一个用于发送。每个列表的基地址会写入在 ETH_DMATDLAR 寄存器和 ETH_DMARDLAR 寄存器中。描述符列表是前向链接的（隐式或显式）。最后一个描述符可以指向第一条目以创建环形结构。描述符的显式链接是通过配置在接收和发送描述符中链接的第二地址来实现的（RDES1[24] 和 TDES1[24]）。描述符列表位于系统的物理内存空间中。每个描述符最多可以指向两个缓存区。这允许使用两个物理寻址的缓存区，而不是内存中的两个连续缓存区。数据缓存区位于系统的物理内存空间中，由整个帧或部分帧组成，但不能超过单个帧。缓存区中只包含数据。缓存区的状态会在描述符中显示。但是，单个描述符不能跨越多个帧，当检测到帧结束时，DMA 跳到下一个帧缓存区。数据链可以使能也可以禁止。环形和链形的描述符结构如下：



33.5.1 使用 DMA 传输的初始化过程

初始化 MAC 过程如下：

1. 向 ETH_DMABMR 寄存器配置系统访问的参数；
2. 向 ETH_DMAIER 寄存器配置来屏蔽必须要的中断；
3. 软件驱动创建发送和接收描述符列表，然后将每个列表的起始地址写入到 ETH_DMARDLAR 和 ETH_DMATDLAR 寄存器中以提供给 DMA；
4. 配置 MAC 寄存器 1、2、3 来选择期望的过滤属性；
5. 配置 MAC ETH_MACCR 寄存器使能接收和发送工作模式，根据自动协商结果（从 PHY 读取）设置 PS 和 DM 位；
6. 写 ETH_DMAOMR 寄存器的 bit13 和 bit1 为 1，开始发送和接收；

7. 发送和接收引擎状态机开始运转，先尝试从各个描述符列表中获取描述符，然后开始进行发送和接收操作。发送和接收过程彼此独立，并且可以分别启动或停止

33.5.2 主机突发访问

如果配置 ETH_DMABMR 的 FB 位，则 DMA 在 AHB 主机接口尝试执行固定长度的突发传输。最大的突发长度由 PBL 决定。对于要读取的 16 个字节，总是以最大可能的突发大小（受 PBL 限制）访问接收和发送描述符。

只有当发送 FIFO 中有足够的空间容纳配置的突发或字节数时，发送 DMA 才会启动数据传输，直到帧结束（当它小于配置的突发长度时）。DMA 指示 AHB 主接口所需的起始地址和传输次数。当 AHB 接口被配置位固定长度的突发，其会以最优的 INCR4，INCR8,INCR16 和 SINGLE 组合方式进行传输。否则（无固定的长度突发），使用 INCR 和 SINGLE 方式传输数据。

只有当接收 FIFO 中有足够的数据用于配置的突发时，或者当在接收 FIFO 中检测到帧结束时（当它小于配置的突发长度时），接收 DMA 才会启动数据传输。DMA 指示 AHB 主接口所需的起始地址和传输次数。当 AHB 接口被配置位固定长度的突发，其会以最优的 INCR4，INCR8,INCR16 和 SINGLE 组合方式进行传输。否则（无固定的长度突发），使用 INCR 和 SINGLE 方式传输数据。当 AHB 接口被配置用于地址对齐节拍时，两个 DMA 引擎都确保 AHB 启动的第一个突发传输小于或等于配置的 PBL 的大小。因此，所有后续节拍都在与配置的 PBL 对齐的地址处开始。DMA 只能将节拍的地址对齐到大小 16（对于 PBL>16），因为 AHB 接口不支持超过 INCR16。

33.5.3 主机数据缓存区对齐

发送和接收数据缓存对起始地址对齐没有任何限制。在 32 位的系统内存中，缓存的起始地址可以对其到 4 个字节中任意一个。但是，DMA 总是在地址对齐的情况下启动传输，不需要字节通道的带有伪数据的总线宽度。这通常发生在以太网帧的开始或结束的传输过程中。

- 读缓存区的例子：

如果发送缓存的地址为 0x0000 0FF2，并且有 15 个字节需要传输，那么 DMA 会从地址 0x0000 0FF0 开始读取完整的 5 个字数据（20 个字节），但是当发送数据给 Tx FIFO 时，额外的字节（开始的 2 个字节）会被丢弃。同样的是，最后 3 个字节也会被丢弃。DMA 需要确保传输到 Tx FIFO 的是 32 位数据，除非是帧尾的数据。

- 写缓存区的例子：

如果接收缓存的地址为 0x0000 0FF2，并且 16 个字节的数据需要接收，那么 DMA 会从地址 0x0000 0FF0 地址开始写 5 个完整的 32 位数据。但是第一次传输的前 2 个字节和第三次传输的最后 2 个字节传输将具有伪数据。

33.5.4 缓存区大小计算

DMA 不更新发送和接收描述符中的大小字段。DMA 只会更新其状态字段。驱动需要区计算大小。传输 DMA 向 MAC 核心传输确切数量的字节（由 TDES1 中的缓冲区大小字段指示）。如果描述符 TDES1 中的 FS 位被置 1，则 DMA 将来自缓冲器的第一次传输视为帧的开始。如果描述符 TDES1 中的 LS 位被置 1，则 DMA 将来自缓冲器的这一次传输视为帧的结束。接收 DMA 会一直发送数据直到缓存区满了或者一帧数据发完。如果描述符没有标记为最后一个（RDES0 中的 LS 位），但是当 FS 位置 1 时，则与描述符相对应的缓冲区已满并且缓存区中的有效数据就等于大小字段的值减去缓存指针指向的偏

移量。当数据缓存指针与数据位宽对齐时，偏移量为 0。如果描述符被标记为最后，缓存区可能不是满的（如 RDES1 中的缓存区大小所示）。要计算该最终缓冲区中的有效数据量，驱动器必须读取帧长度（RDES0[29:16]中的 FL 位），并减去该帧中先前缓存区的缓存大小之和。通常接收 DMA 会在新的描述时发送下一个帧的开始。

注意：即使当接收缓存的起始地址与系统数据位宽不对齐，系统应该分配一个大小与系统总线宽度一致的接收缓存区。举个例子：如果系统分配一个 1024 字节的接收缓存区，起始地址为 0x1000，软件可以在接收描述符中编程起始地址为 0x1002 的偏移。接收 DMA 会在将帧数据写入到缓存区时，在最初的 2 个位置补充伪数据（0x1000 和 0x1001）。实际的帧是从 0x1002 开始写的。因此，由于起始地址的偏移，在缓存区中真正有效的空间为 1022 个字节，即使缓存区大小被配置成 1024 字节。

33.5.5 DMA 仲裁

DMA 内部的仲裁器负责对 AHB 主接口的发送和接收信道访问之间的仲裁。支持两种类型的仲裁方式：循环方式和固定优先级方式。当选择了循环方式（ETH_DMABMR 寄存器的 DA 位为 0），当发送和接收 DMA 同时请求访问时，通过设置 PM 位仲裁器按比例分配数据总线。当 DA 位为 1，对于数据访问，接收 DMA 总是优先于发送 DMA。

33.5.6 DMA 错误状态

对于 DMA 通道启动的任何数据传输，如果从设备回复错误响应，则该 DMA 将停止所有操作，并更新状态寄存器（ETH_DMASR 寄存器）中的错误位和致命总线错误位。该 DMA 控制器只能在软或硬重置外围设备并重新初始化 DMA 之后才能恢复操作。

33.5.7 Tx DMA 的配置

33.5.7.1 非 OSF 模式（默认模式）

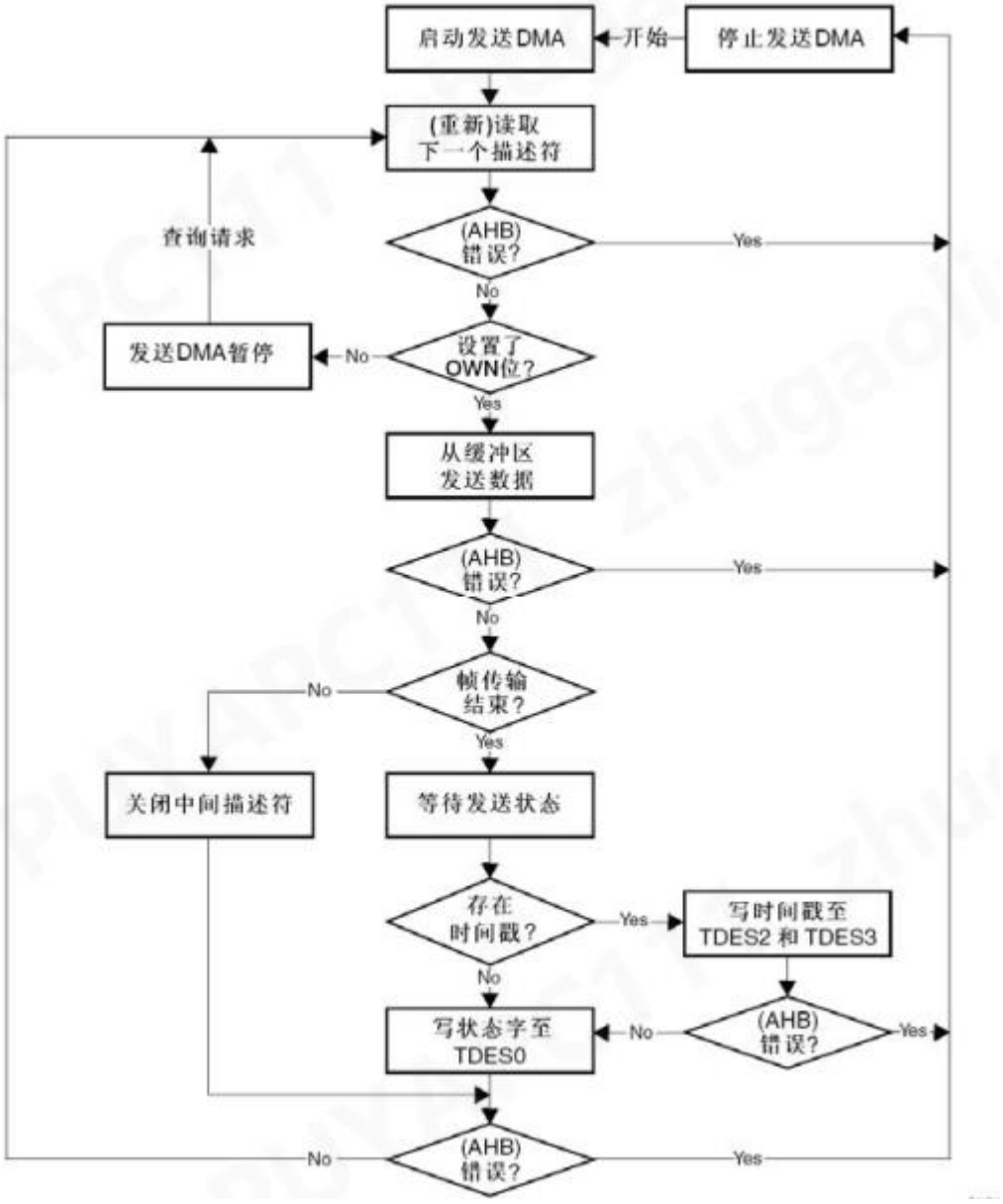
发送 DMA 引擎的默认配置：

设置发送描述符（TDES0-TDES3），在用以太网帧数据建立相应的数据缓存区之后置位 TDES0 中的 OWN 位；

1. 当 ETH_DMAOMR 寄存器的 ST 位置 1，DMA 进入运行状态；
2. 当进入到运行状态时，DMA 轮询发送描述符列表。当轮询完成后，它以顺序描述符环顺序或链式顺序继续；
3. 如果 DMA 检测到一个被标记为 CPU 拥有的描述符，或者发生了一个错误状态，发送缓存区（ETH_DMASR 寄存器的 bit2）会不可用并且常规中断统计位被置位（ETH_DMASR 寄存器的 bit16），发送引擎流程会跳转到步骤 9；
4. 如果获取的描述符被标记为 DMA 拥有（TDES0[31]）被置 1），DMA 会从获得的描述符中解码发送数据缓存地址；
5. DMA 从存储器中取出传输数据并传输数据；
6. 如果以太网帧存储在多个描述符中的数据缓冲区上，DMA 关闭中间描述符并获取下一个描述符。步骤 3、4、5 会重复直到帧尾被传输；

7. 当帧发送完成，如果 IEEE 1588 时间戳功能被使能，那么时间戳的值会被写入包含帧尾缓存的发送描述符 (TDES2 和 TDES3) 中。状态信息会被写入到发送描述符 TDES0。由于在该步骤中 OWN 位被清除，CPU 现在开始占用描述符。如果时间戳未使能，DMA 就不会修改 TDES2 和 TDES3；
8. 传输中断 (ETH_DMASR 寄存器[0]) 是在完成在其最后描述符中设置了 “完成中断”（TDES1[31]）的帧的传输之后设置的。DMA 引擎会回到步骤 3
9. 在挂起状态下，当 DMA 接收到发送轮询请求时，DMA 尝试重新获取描述符（然后返回步骤 3），并且下溢中断状态位被清除。

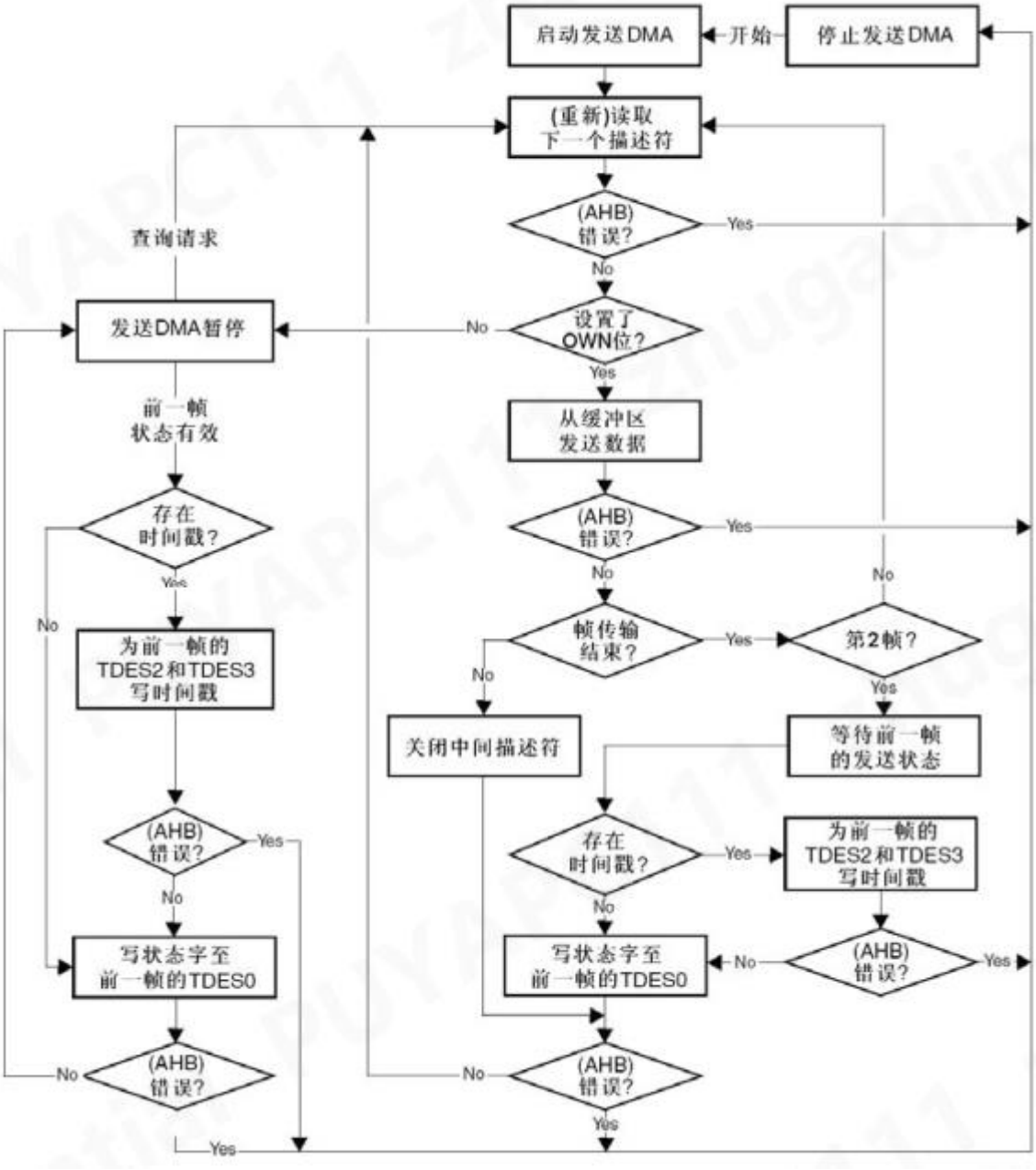
默认模式下 Tx DMA 发送流程：



33.5.7.2 OSF 模式

当在运行状态时，发送过程可以同时获取两个帧而不关闭第一个帧的状态描述符（如果置位了 ETH_DMAOMR 中的 OSF 位）。当完成第一帧的发送，它立即轮询第二帧的传输描述符列表。如果第二帧是有效的，发送流程会在写入第一帧的状态信息之前发送该帧。在 OSF 模式中，运行状态下 DMA 操作如下：

1. DMA 操作与 TxDMA（默认模式）的 1-6 步骤相同；
2. 不需要关闭先前帧的最后一个描述符，DMA 会直接取下一个描述符；
3. 如果 DMA 占用了所获取的描述符，DMA 会解码发送缓存的地址。如果 DMA 没有占用描述符的权利，DMA 会进入到挂起的状态，并且跳转到步骤 7；
4. DMA 从内存中取帧数据进行发送，直接帧尾被发完。如果该帧被拆分为多个描述符，则关闭中间描述符；
5. DMA 等待前一帧的发送状态和时钟戳，当状态变得有些时，如果捕获到了时间戳，DMA 将时间戳值写入到 TDES2 和 TDES3。MA 然后将状态（已清除的 OWN 位）写入相应的 TDES0，从而关闭描述符。如果前一帧中时间戳没有被使能，DMA 不会修改 TDES2 和 TDES3；
6. 如果发送中断被使能并置起，DMA 会获取下一个描述符，然后执行步骤 3(如果状态是正常的)。如果先前帧的状态是一个下溢错误，DMA 会进入挂起模式（步骤 7）；
7. 在挂起模式中，如果挂起状态和时钟戳被 DMA 接收了，DMA 会将时间戳写入到 TDES2 和 TDES3，然后将状态写入对应的 TDES0 中，然后设置相关的中断并返回到挂起状态。
8. 仅当接收到发送轮询命令（ETH_DMATPDR 寄存器）时，DMA 会离开挂起状态然后进入到运行状态（进入步骤 1 和步骤 2）



33.5.7.3 发送帧的进程

发送 DMA 期望数据缓存能包含完整的以太网帧，除前导码、填充字段和 FCS 字段以外。目的地址、源地址和类型/长度字段包含有些的数据。如果发送描述符显示 MAC 控制器必须关闭 CRC 和填充插入，则缓冲器必须具有完整的以太网帧（不包括前导码），包括 CRC 字节。帧可以是数据链，并跨越多个缓冲区。帧必须由第一缓存描述符（TDES1[29]）和最后缓存描述符（TDMES1[30]）分隔。当发送开始时，TDES1[29]需要在第一个描述符中设置，当它置 1 时，帧数据就开始从缓存区向 Tx FIFO 发送。如果当前帧的最后一个描述符 TDMES0[29]被清除，发送流程会尝试获取下一个描述符。发送进程期望在此描述符中 TDES1[29]被清除。如果 TDES1[30]被清除，则表明存在中间缓冲区。如果 TDES1[30]被置位，表示帧的最后一个缓存区。当帧最后一个缓存区被发送，DMA 会将最后的状态信息写入到发送描述符 0（TDES0）的字中，该字在传输描述符 0 中设置了最后一个段（TDES1[30]）。与此同时，如果表示完成的中断（TDES1[31]）置位了, ETH_DMASR 寄存器中发送中断标志（bit0）会置 1，下一

个描述符会被取走，进行重复。实际帧的发送在发送 FIFO 达到可编程发送阈值（ETH_DMAOMR 寄存器[16:14]）之后开始，或者一个完整的帧被存到 FIFO 时开始。当然还有存储转发模式。当 DMA 完成一个帧的发送后，会释放占用（TDES0[31]的 OWN 位清零）

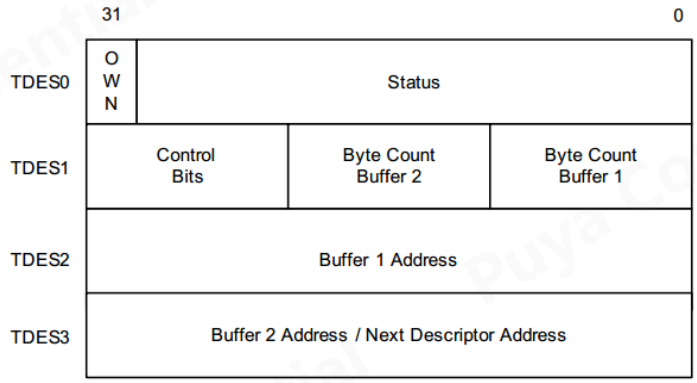
33.5.7.4 发送轮询被暂停

发送轮询进程会被如下情况暂停：

- DMA 检测到描述符被 CPU 占用，并且发送缓存区的不可用标志被置 1 了（ETH_DMASR 寄存器的 bit[2]），如果要继续，驱动程序必须将描述符所有权授给 DMA，然后发出轮询请求命令；
- 当检测到下溢导致的传输错误时，帧传输中止。适当的传输描述符 0（TDES0）位会被置起。如果第二种情况发送，总异常中断（在 ETH_DMASR 寄存器[15]中）和发送下溢位（在 ETH_DMASR 寄存器[5]中）都会置 1，并且该信息会写入描述符 0，引起暂停和挂起。如果 DMA 因为第一种情况进入到挂起状态，则总正常中断位（ETH_DMASR 寄存器[16]）和发送缓存区不可用位（ETH-DMASR 寄存器[2]）都会置 1。在这两种情况下，发送列表的位置会被保存，保存的位置即是被 DMA 关闭的最后一个描述符的下一个描述符。当纠正暂停之后驱动必须发出一个发送轮询需求命令。

33.5.7.5 发送 DMA 描述符

每次帧传输时至少需要有一个描述符



- TDES0：发送描述符字 0

TDES0 包含了发送帧的状态和描述符的 owner 属性，状态位仅在 LS 位置 1 时有效，因此当一个帧通过多个描述符进行发送时，仅在最后一个描述符时，状态位才有效。

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'。
30 :18	保留
17	**TTSS：**发送时间戳状态(Transmit time stamp status） **该状态位表示在发送帧中捕获到了时间戳** 置'1'时：TDES2 和 TDES3 中包含了捕获的时间戳值该位仅在 LS 位为 1 时才有效

16	**IHE**：IP 报头错误（IP header error） 置’1’时，表示 校验和减载功能检测到了 IP 报头错误，该位仅在发送校验和减载功能使能时有效。否则当检测到 IP 报头错误时，若以太网类型字段显示为 IPv4 payload，则仍然会插入 IPv4 报头校验和
15	**ES**：错误汇总（Error Summary) TDES0[16]：IP 报头错误 TDES0[14]：Jabber 定时器溢出 TDES0[13]：帧清空 TDES0[12]：payload 校验和错误 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]：下溢错误
14	**JT**：Jabber 定时器溢出（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 JD 位不为’1’时才会被置’1’。
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。
12	**PCE：**payload 校验和错误（Payload Checksum Error） 置’1’时，表示校验和减载功能检测到错误，并且不会将校验和插入到 TCP、UDP 和 ICMP 有效数据中。该错误的产生情况有两种：1.实际的有效数据字节数小于 IP 报头的有效数据长度字段的值；2 在存储转发模式中，当校验和还没被计算出来，传输层就开始转发帧数据到 MAC 发送器。仅当发送 FIFO 的深度小于以发送的太网帧的长度时才会发生第二种情况。
11	**LOC**：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。
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。
9	**LC**：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。
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。
7	**VF**：VLAN 帧（VLAN frame) 置’1’时，表示发送的帧是 VLAN 帧
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
2	**ED**：顺延过多（Excessive deferral) 置’1’时，表示当 MAC 设置寄存器的顺延检查位为’1’时，由于顺延超过 24288 位的时间而结束发送
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_DMASR 寄存器位 5)和发送状态位(ETH_DMASR 寄存器位 0)置’1’
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

TDES1 包含了缓存的大小以及发送帧时其它控制描述符链或者环的控制位

31	**IC**：完成时的中断（Interrupt on Completion) 置'1'时：当当前帧被发送后，会置位发送中断（ETH_DMASR） 注意：仅当 LS 位置 1 时该位有效
30	**LS：**最后的分块(Last Segment) 置'1'时：表示缓存区包含了帧的最后一个分块，此时 TBS1 和 TBS2 位段应该是各非零值
29	**FS：**最开始分块(First Segment) 置'1'时：表示缓存区包含了帧的最开始的一个分块
28：27	**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：插入 IPv4 报头校验和，当帧封装 IPv4 数据报时，使用此值插入 IPv4 标头校验和 10：插入 TCP/UDP/ICMP 校验和，校验和仅在 TCP、UDP 或 ICMP 段上计算，并且假定 TCP、UDP 和 ICMP 伪标头校验和存在于相应输入帧的校验和字段中。如果封装的数据报符合 IPv4，则还插入 IPv4 报头校验和。 11：插入完全计算的 TCP/UDP/ICMP 校验和，即伪报头也会进行计算，并且输入帧的相应校验和字段具有全零值。如果封装的数据报符合 IPv4，则还插入 IPv4 报头校验和。 校验和引擎检测 TCP、UDP 或 ICMP 段是否封装在 IPv4 或 IPv6 中，并相应地处理其数据。 该位仅当 FS 位置 1 时有效
26	**DC**：禁止 CRC（Disable CRC) 置'1'时，MAC 不会再发送帧的帧尾附加 CRC 该位仅当 FS 位置 1 时有效
25	**TER**：发送描述符列表的最后（Transmit End of Ring) 置'1'时，表示描述符列表到达了最后一个描述符，并且会返回到列表的初始地址以创建描述符环
24	**TCH：**下一个描述符链的地址（Second Address Chained） 置'1'时，表示描述符中的下一个地址的位段表示的是下一个描述符的地址而不是下一个缓存的地址。当该位为 1 时，TBS2 中的值即没有作用。TER 的功能优先于本位
23	**DP**：关闭 padding（Disable Padding) 置'1'时，当帧小于 64 字节时，MAC 不会自动增加填充字段。当该位复位后，DMA 会自动增加填充和 CRC。 该位仅当 FS 位为 1 时有效
22	**TTSE**：发送描述符使能(Transmit Timestamp Enable) 置'1'时，使能硬件时间戳功能 仅当 FS 位为 1 时有效
21：11	**TBS2**：发送缓存 2 的大小（Transmit Buffer 2 Size) 表示第二个缓存区的大小，当 TCH 为 1 时此位无效
10：0	**TBS1**：发送缓存 1 的大小（Transmit Buffer 1 Size) 表示第一个缓存区的大小 ，如果值为 0，DMA 会忽略该缓存区并且使用缓存区 2 或下一个描述符（取决于 TCH）

■ TDES2：发送描述符字 2

TDES2 包含了描述符的第一个缓存区的地址

31 ： 0	缓存区 1 地址指针（Buffer 1 Address Pointer) 该位段表示的是缓存区 1 的物理地址，缓存区地址的对齐没有限制

- TDES3：发送描述符字 3

TDES3 包含了描述符的第二个缓存区或下一个描述符的地址

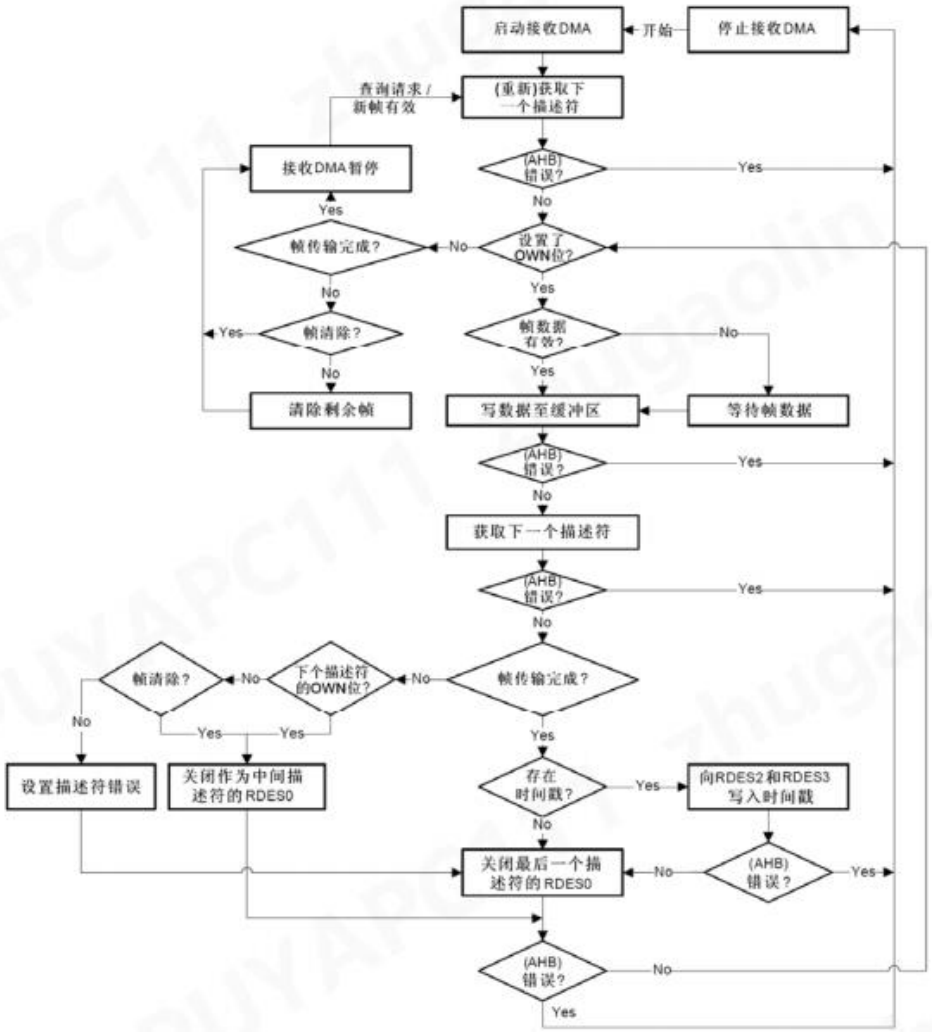
31 ： 0	缓存区 2 地址指针/下一个描述符地址（Buffer 2 Address Pointer/Next Descriptor Address) 当环形描述符架构被使用，则该位段表示的是缓存区 2 的物理地址 当描述符字 1 中的 TCH 置 1，则该位段表示的是下一个描述符物理地址，此时缓存地址必须与总线宽度对齐。

33.5.8 Rx DMA 的配置

接收 DMA 引擎接收步骤如下所示：

1. CPU 设置接收描述符（RDES0-RDES3），并且设置 OWN 位；
2. 一旦 ETH_DMAOMR 寄存器的 SR 位置 1，DMA 会进入运行模式。在运行模式中，DMA 会轮询接收描述符列表，尝试获取空闲描述符。如果被取走的描述符不是空闲的（CPU 占用），DMA 会进入挂起状态然后跳转到步骤 9；
3. 通过获取的描述符 DMA 解码接收数据缓存区的地址；
4. 接收帧被处理后放置在描述符的数据缓存区中；
5. 当缓存区满了或者一帧数据被传送完了，接收引擎会换取下一个描述符；
6. 如果当前帧发送完了，DMA 会进行步骤 7。如果 DMA 没有占用下一个获取的描述符的并且帧发送还未完成，DMA 会将 RDES0 中设置描述符错误位置 1(除非禁止清空)。DMA 会关闭当前的描述符（通过清除 OWN 位）并且通过清除 RDES1 值中的 LS 位将其标记为中间的描述符（如果未禁用清空，则将其标记最后的描述符），然后执行步骤 8.如果 DMA 占用下一个描述符但是当前的帧还没完成发送，DMA 会关闭当前的描述符而认作中间描述符，并且返回到步骤 4；
7. 如果 IEEE 1588 时间戳被使能，DMA 会将时间戳数据写入到当前描述符的 RDES2 和 RDES3。然后将接收帧的状态信息写入到状态字（RDES0）中，将 OWN 位清除，LS 位置 1。
8. 接收引擎会检查最新的描述符的 OWN 位。如果 CPU 拥有描述符（OWN 位为 0），则设置接收缓存区不可用位（在 ETH_DMASR 寄存器[7]中），DMA 接收引擎进入挂起状态（步骤 9）。如果 DMA 拥有描述符，会返回到步骤 4 等待下一个帧
9. 在接收引擎进入到挂起状态时，会清除接收 FIFO 中的部分帧（可设置清空控制位）；
10. 当接收轮询请求被给出或下一帧的开始可从接收 FIFO 获得时，接收 DMA 退出挂起状态。引擎回到步骤 2 并重新获取下一个描述符

DMA 在完成时间戳回写并准备好执行对描述符的状态回写之前，不会确认接收状态。如果软件使能了时间戳功能，当一个有效的时间戳值对帧无效时（例如，在时间戳被写入时接收 FIFO 已满了），DMA 将所有 1 写入 RDES2 和 RDES3。否则（时间戳未使能），RDES2 和 RDES3 不会改变。



33.5.8.1 获取接收描述符

接收引擎会一直尝试对到来的帧进行额外的描述符获取。当以下任一条件满足时尝试获取描述符：

- 当 DMA 处于运行状态后，立即置位接收开始/接收位（ETH_DMAOMR bit[1]）
- 在接收帧尾前当前描述符的数据缓存区已满
- 帧已完成接收，但当前接收的描述符未关闭
- 由于 CPU 占用缓存区或者一个新的帧被接收导致接收进程被挂起
- 发出接收轮询请求

33.5.8.2 接收帧进程

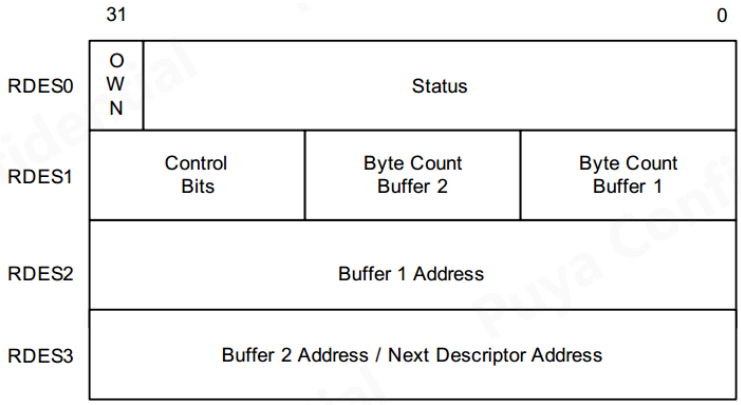
仅当帧通过地址过滤后，并且帧的大小大于等于 F 配置的 FIFO 阈值，MAC 才会发送和接收帧，或者使用存储转发模式。如果帧地址过滤失败，会直接丢弃（除非设置了 ETH_MACFFR[31] 接收所有的帧）。由于冲突或提前终止，短于 64 字节（可配置）的帧会从接收 FIFO 中被清除。DMA 块开始将帧数据传输到当前描述符所指向的接收缓存区。DMA 在 AHB 接口可以接收数据传输后后设置第一个描述符（RDES0[9]）。当 OWN（RDES0[31]）位被重置为 0 时，描述符被释放，无论是在数据缓存区填满时，还是在帧的最后一段被传输到接收缓存区时。如果一个帧被包含在一个描述符，那么第一个描述符（RDES0[9]）和最后一个描述符（RDES0[8]）标识位均被置 1。DMA 在先前的描述符中置 1 最后一个描述符并且释放 RDES0 状态位后，开始获取下一个描述符。然后 DMA 将接收中断位（ETH_DMASR

寄存器的 bit[6]）置 1。除非 DMA 遇到被标记为由 CPU 拥有的描述符，否则重复相同的过程。如果发送了这种情况，接收进程会将接收缓存无效位（ETH_DMASR bit[7]）置 1 并且进入到挂起状态。接收列表中的位置将保留。

33.5.8.3 接收进程挂起

如果一个新的接收帧在接收进程被挂起时到达，DMA 会重新获取当前内存中的描述符。如果该描述符被 DMA 拥有，接收进程会重新进入运行状态开始帧的接收。如果描述符依旧被 CPU 拥有，默认情况下 DMA 丢弃 Rx FIFO 顶部的当前帧，并递增丢失帧计数器。如果 Rx FIFO 中超过 1 个完成帧，此进程会重复。可以通过设置 DMA 操作模式寄存器位 24（DFRF）来避免在 Rx FIFO 顶部丢弃或刷新帧。在这种情况下，接收进程设置接收缓存区不可用状态位并返回到挂起状态。

33.5.8.4 DMA 接收描述符



■ RDES0：接收描述符字 0

RDES0 包含了接收帧的状态、帧长以及描述符的归属等信息。这些状态位仅当 LS 位置 1 时有效。

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 在接收完整个帧或者这个缓存里的数据全部填满以后把该位清'0'。
30	**AFM**：目的地址过滤失败（Destination Address Filter Fail） 置'1'时：MAC 的目的地址过滤失败
29：16	**FL**：帧长度（Frame Length） 该位段表示接收帧的字节长度被发送到系统内存中。仅当 LS（RDES0[8]）置 1 时、描述符错误位（RDES0[14]）或上溢错误位为 0 时有效。当启用 IP 校验和计算并且接收到的帧不是 MAC 控制帧时，帧长度还包括附加到以太网帧的两个字节。 当最后描述符位没有置 1，该位段指示当前帧已传输的累计字节数。当 ETH_MACCR 寄存器的 bit7 和 bit25 置 1 时，帧长还包含 CRC 的长度信息。
15	**ES**：错误汇总（Error Summary) RDES0[0]：Payload 校验和错误 RDES0[1]：CRC 错误 RDES0[3]：接收错误 TDES0[4]：看门狗溢出 TDES0[6]：迟到冲突 TDES0[7]：IPC 校验和错误

	TDES0[11]：上溢错误 TDES0[14]：描述符错误 该位仅当最后描述符置 1 时有效
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。
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。
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。
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。
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。
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。
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。
7	IPC 校验和错误(IPC 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。
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。
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。
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
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
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
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
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'

Bit 5: Frame Type	**Bit 7: IPC Checksum Error**	**Bit 0: Payload Checksum Error**	**Frame Status**
0	0	0	IEEE802.3 类型帧(长度域值小于 1536)

1	0	0	ETH_MACCR 的 bit10 为 1：IPv4/IPv6 类型帧，未检测到校验和错误 ETH_MACCR 的 bit10 为 0：IEEE802.3 类型帧（帧长大于等于 1536）
1	0	1	IPv4/IPv6 类型帧，检测到 payload 校验和错误
1	1	0	IPv4/IPv6 类型帧，检测到 IP 报头校验和错误
1	1	1	IPv4/IPv6 类型帧，检测到 payload 校验和错误和 IP 报头校验和错误
0	0	1	IPv4/IPv6 类型帧，不检测 payload 校验和错误，无 IP 报头校验和错误
0	1	1	非 IPv4/IPv6 类型帧
0	1	0	保留

■ RDES1：接收描述符字 1

RDES1 包含了缓存大小和其它控制环形或链形描述符的控制位

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_DMASR 寄存器的 RS 位(位 6)。这样也就不会发生 RS 位触发的中断 该位仅当最后描述符位（RDES0[8]）置 1 时有效
30：26	保留
25	**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
24	**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])。
23：22	**保留**
21：11	**RBS2**：发送缓存 2 的大小（Transmit Buffer 2 Size) 表示第二个缓存区的大小，缓存大小必须是 4 的倍数。当 RCH 为 1 时此位无效
10：0	**RBS1**：发送缓存 1 的大小（Transmit Buffer 1 Size) 表示第一个缓存区的大小，缓存大小必须是 4 的倍数。如果值为 0，DMA 会忽略该缓存区并且使用缓存区 2 或下一个描述符（取决于 RCH）

■ RDES2：接收描述符字 2

RDES2 包含了描述符的第一个缓存区的地址

31 ： 0	缓存区 1 地址指针（Buffer 1 Address Pointer) 该位段表示的是缓存区 1 的物理地址，缓存区地址的对齐没有限制 。当 RDES2 值用于存储帧的开始时，DMA 使用配置的值来生成其地址。在帧开始的传输过程中，DMA 以 RDES2[1:0]位作为 0 执行写入操作，但帧数据根据实际的缓冲区地址指针进行移位。

■ RDES3：发送描述符字 3

RDES3 包含了描述符的第二个缓存区或下一个描述符的地址

31 ： 0	缓存区 2 地址指针/下一个描述符地址（Buffer 2 Address Pointer/Next Descriptor Address) 当环形描述符架构被使用，则该位段表示的是缓存区 2 的物理地址 当描述符字 1 中的 RCH 置 1，则该位段表示的是下一个描述符物理地址，此时缓存地址必须与总线宽度对齐。

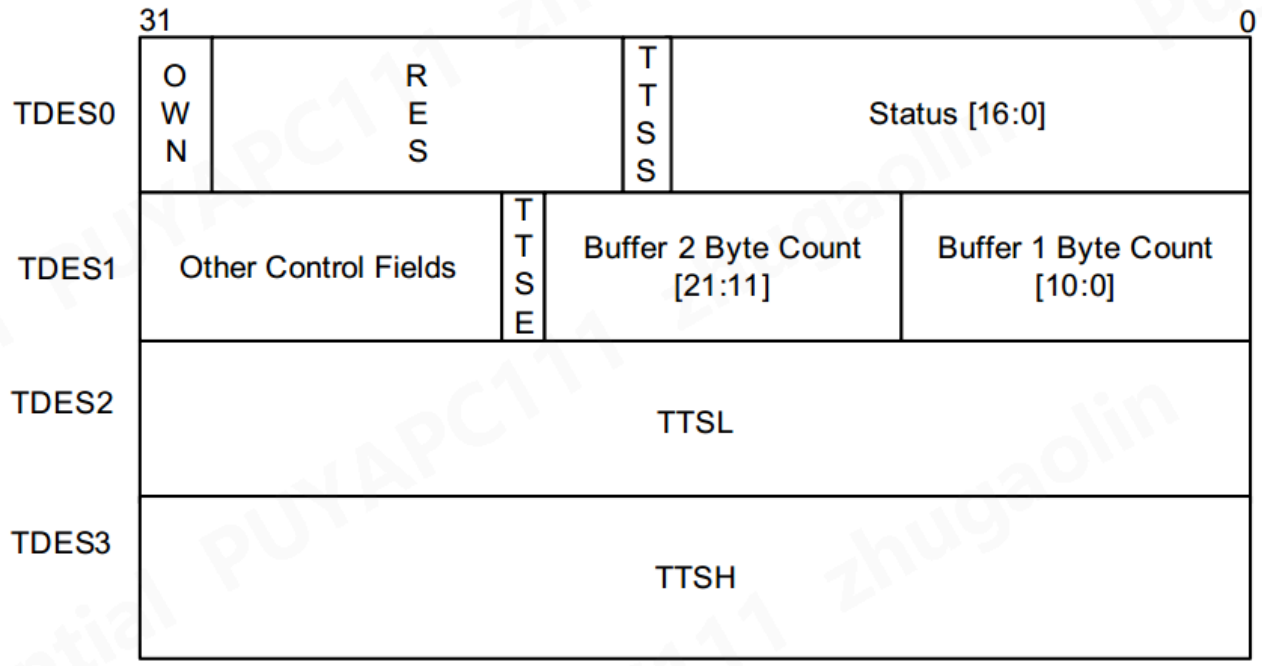
	当 RCH 置 0 时，RDES3 的值就没有限制了，除了以下情况：当 RDES3 值用于存储帧的开始时，DMA 使用配置的值来生成其地址

33.5.9 使能 IEEE1588-2005 时间戳的描述符格式

如果软件使能了 IEEE1588-2005 功能，当 DMA 关闭描述符时（清除了 DES0 的 OWN 位）DES2 和 DES3 描述符区域的内容具有不同的含义。在 DMA 清除了 DES0 的 OWN 位前会更新时间戳到 DES2 和 DES3.。

33.5.9.1 发送描述符

发送描述符存在更多的控制（TTSE 和 TTSS）来进行时间戳的发送，如下图，当软件置位 TTSE（OWN 为 1 时），表示会生成时间戳用于以太网帧的发送。DMA 会在描述符被关闭前将时间戳更新到 TDES2 和 TDES3。



- TDES0：

17	TTSS:发送时间戳状态（Transmit Timestamp Status) 此字段是一个状态位，指示已为相应的传输帧捕获了时间戳。当设置此位时，TDES2 和 TDES3 都具有为传输帧捕获的时间戳值。 仅当设置了最后一段控制位（描述符中的 TDES1[30]）时，此字段才有效。

- TDES1：

22	TTSE：发送时间戳使能（Transmit Timestamp Enable) 设置后，此字段将为描述符描述的发送帧启用 IEEE1588 硬件时间戳。 仅当设置了第一段控制位（描述符中的 TDES1[29]）时，此字段才有效。

传输描述符格式和字段描述在由软件创建时保持不变（OWM 为 1），然而，当 DMA 关闭最后的描述符并且 IEEE1588 功能被使能时，TDES2 和 TDES3 内容会被时间戳更新。

- TDES2：

31:0	TTSL：发送帧时间戳低位（Transmit Frame Timestamp Low) DMA 使用为相应传输帧捕获的时间戳的最低有效 32 位来更新此字段。仅当描述符中的最后段控制位（LS）被设置时，此字段才具有时间戳

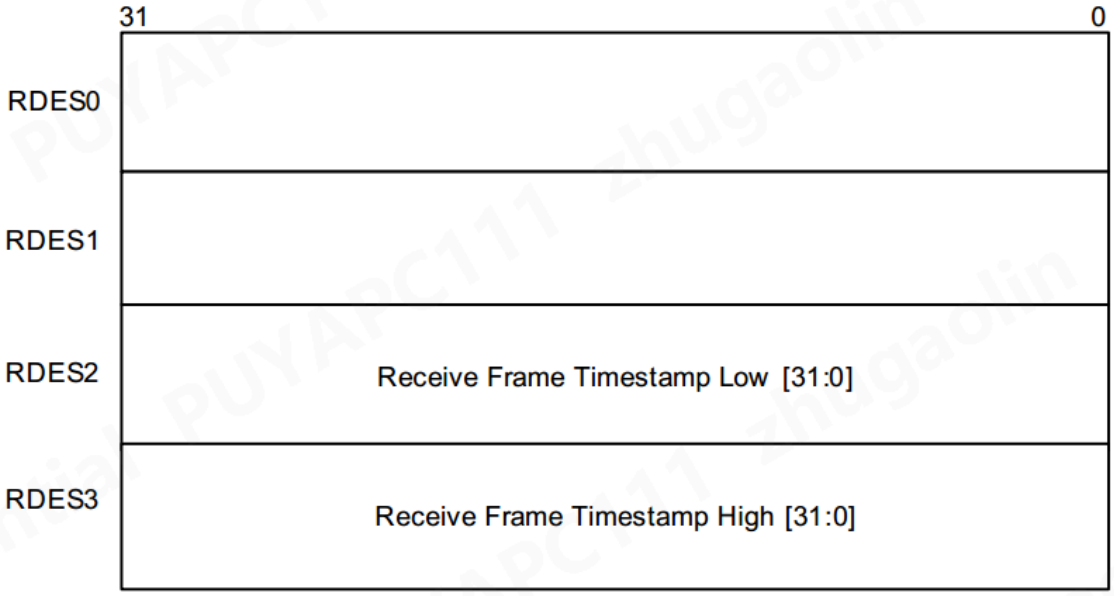
- TDES3：

31:0	TTSL：发送帧时间戳低位（Transmit Frame Timestamp High) DMA 使用为相应传输帧捕获的时间戳的最高有效 32 位来更新此字段。仅当描述符中的最后段控制位（LS）被设置时，此字段才具有时间戳

33.5.9.2 接收描述符

当时间戳被使能时，接收描述符格式如下：

当接收描述符关闭并且启用时间戳时，接收描述符的 RDES2 和 RDES3 具有不同含义的字段。



- RDES2：

31:0	RTSL：接收帧时间戳低位（Receive Frame Timestamp Low) DMA 使用为相应接收帧捕获的时间戳的最低有效 32 位来更新此字段。DMA 仅针对由 last descriptor 状态位（RDES0[8]）指示的接收帧的最后一个描述符更新此字段。当 RDES3 中的此字段和 RTSH 字段显示全 1 值时，时间戳必须被视为已损坏

- RDES3：

31:0	RTSH：接收帧时间戳高位（Receive Frame Timestamp High) DMA 使用为相应接收帧捕获的时间戳的最高有效 32 位来更新此字段。DMA 仅针对由 last descriptor 状态位（RDES0[8]）指示的接收帧的最后一个描述符更新此字段。当 RDES3 中的此字段和 RTSH 字段显示全 1 值时，时间戳必须被视为已损坏

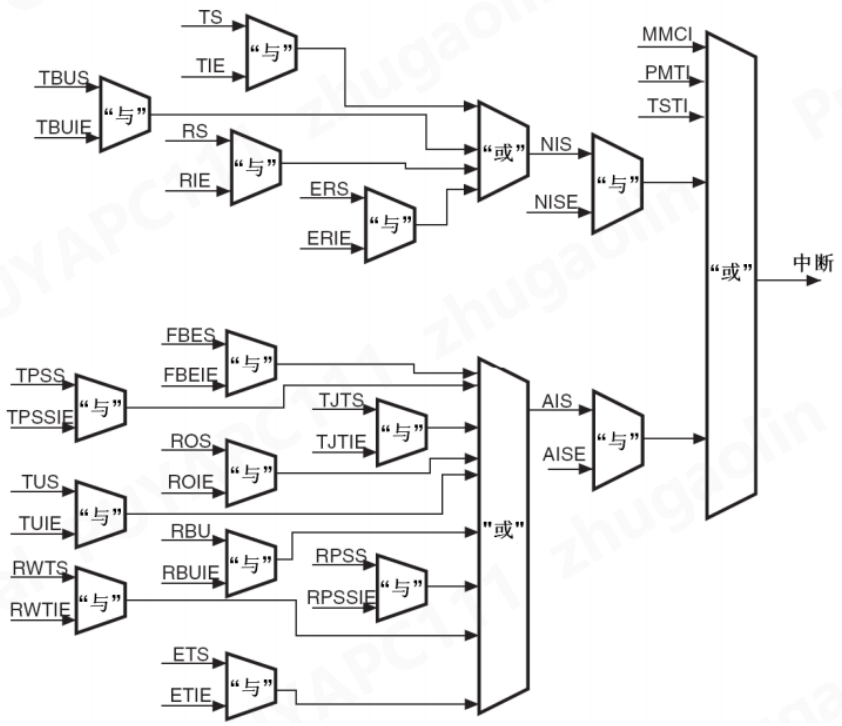
33.5.9.3 DMA 中断

有多种事件可以产生中断。ETH_DMASR 包含了所有能引发中断的位，而 ETH_DMAIER 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_DMASR 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_DMASR 的 TSTS 位或者 PMTS 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_DMASR[6])表示一个或者多个帧已经接收到缓存里，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_DMASR 寄存器来判断中断的来源。在正确地清除掉 ETH_DMASR 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_DMASR[6]为'1')，中断处理程序读出 ETH_DMASR 寄存器后，又发生了接收缓存不可用(ETH_DMASR[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



33.6 以太网中断

以太网控制器有两种中断向量：一种是常规操作引起的中断，一种是用于以太网唤醒时间的中断（映射在 EXTI 线 31）

第一种是为 MAC 和 DMA 生成的中断，见 MAC 中断和 DMA 中断的章节描述。第二种是由 PMT 生成的用于唤醒事件。唤醒事件用于退出低功耗模式而生成的一个中断。当映射在 EXTI 31 的唤醒事件发生时，如果开启了使能，会检测其上升沿用于退出低功耗模式。

在以太网唤醒事件映射到 EXTI 线 31 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 31 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 RS 位(ETH_DMASR[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 RS 位为'1'。在看门狗定时器递减为 0 时，才设置 RS 位为'1'，如果使能了相应的 RIE 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，RS 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。

33.7 寄存器描述

33.7.1 以太网 MAC 配置寄存器（ETH_MACCR)

Address offset:0x0000

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WD	JD.	Res.	Res.	IFG			CSD
-	-	-	-	-	-	-	-	RW	RW	-	-	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FES	ROD	LM.	DM.	IPCO	RD	Res	APCS	BL		DC	TE	RE	Res.	Res.
-	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	RW	RW	RW	-	-

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	-
23	WD	RW	0	禁止 Watchdog 定时器（Watchdog disable） 1：关闭定时器，最多能发送 16384 个字节的帧数据 0：当发送数据超过 2048 个字节后切断后面传输的数据
22	JD	RW	0	禁止 Jabber 定时器（Jabber disable） 1：关闭定时器，最多能发送 16384 个字节的帧数据 0：当发送数据超过 2048 个字节后切断后面传输的数据
21:20	保留	-	-	-
19:17	IFG	RW	0	帧间间隙（Interframe gap) 控制帧发送过程中，两个帧的最小间隙 000：96bit 对应的时间 000：88bit 对应的时间

				000：80bit 对应的时间 … 000：40bit 对应的时间 注意：在半双工模式中，最小的 IFG 仅可以配置为 64bit 的时间（IFG=100）
16	CSD	RW	0	禁止载波侦听(Carrier sense disable) 1：MAC 发送器在半双工模式时不响应 MII CRS 的信号 ，无载波或者载波丢失时不会生成错误。 0：响应，并可以根据载波侦听产生发送错误，甚至中止发送
15	保留	-	-	-
14	FES	RW	0	以太网速率(Fast Ethernet speed) 1:100Mbit/s 0:10Mbit/s
13	ROD	RW	0	禁止接收（Receive own disable) 1：在半双工模式中 MAC 不接收帧 0：在半双工模式中接收所有 PHY 发过来的帧 注意：仅适用于半双工模式
12	LM	RW	0	回环模式（Loopback 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 CEN 位)时，计数器停止。
11	DM	RW	0	全双工模式(Duplex mode) 1：使能 0：不使能
10	IPCO	RW	0	IPv4 校验和减载（IPv4 checksum offload) 对接收帧的有效载荷的 TCP/UDP/ICMP 报头进行 IPv4 的校验和检查（发送方会对所有的字段进行计算出校验和放在报头） 1：使能 0：不使能
9	RD	RW	0	禁止重新发送（Retry disable) 传输重试功能 1：MAC 只尝试一次传输，当 MII 上发生冲突时，MAC 忽略当前帧传输，并报告在传输帧状态中报告错误。 0：多次输出，取决于 BL 寄存器， 注意：仅适用于半双工模式。
8	保留	-	-	-
7	APCS	RW	0	自动填充/CRC 剥离（Automatic pad/CRC stripping) 自动去除填充和 FCS 字段功能 1：帧当数据小于等于 1500 字节时，MAC 去除填充和校验字段，大于等于 1501 时无操作。 0：接收所有的帧数据而不修改
6:5	BL	RW	0	后退限制（Back-off limit)

				在冲突之后的重试期间，MAC 需要等待随机的整数倍的 slot 时间（512 位对应的时间）才能重新尝试发送。 注意：仅用于半双工模式 00:k=min(n,10) 01:k=min(n,8) 10:k=min(n,4) 11：k = min（n, 1） n=发送尝试，随机整数倍的值的区间为[0,2^k]
4	DC	RW	0	延迟检查（Deferral check) 1: 使能延迟检测功能，当发送状态机推迟发送超过 24288bit 的时间时。MAC 发出帧中止状态，并且拉高发送帧中的过度延迟错误标志位。当数据准备发送时就会开启延迟计数，如果 CRS 信号拉高了会中断计数。延迟时间不会累计。如果发送器延迟了 10000 位的时间，然后发送、冲突、退回，然后在回退完成后必须再次延迟，延迟时间会清零并重新计数 仅适用于半双工模式 0：关闭延迟检查功能，MAC 延迟直到 CRS 信号变为无效。 注意：仅用于半双工模式
3	TE	RW	0	发送器使能（Transmitter enable) 1：使能 MAC 发送器状态机 0：在完成当前帧的发送后禁止 MAC 发送器状态机
2	RE	RW	0	接收器使能（Receiver enable) 1：使能 MAC 接收器状态机 0：在完成当前帧的接收后禁止 MAC 接收器状态机
1:0	保留	-	-	-

33.7.2 以太网 MAC 帧过滤寄存器（ETH_MACFFR）

Address offset:0x0004

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			VTFE
RW	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res	Res	Res	Res	HPF	SAF	SAIF	PCF		BFD	PAM	DAIF	HM	HU.	PM
-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	RA	RW	0	接收所有帧（Receive all） 1.接收所有的 MAC 帧，不管是否通过地址过滤。SA/DA 地址过滤结果在接收状态字中会进行更新（通过或失败） 0： 仅接收通过了 SA/DA 地址过滤的帧
30:17	保留	-	-	-
16	VTFE			VLAN 标签过滤使能(VLAN Tag Filter Enable) 1：使能 MAC 能够丢弃与 VLAN 标记比较不匹配的带有 VLAN 标记的帧 0：MAC 转发所有帧，而与 VLAN 标签的匹配状态无关

15:11	保留	-	-	-
10	HPF	RW	0	哈希或完成过滤（Hash or perfect filter) 1：如果 HM 和 HU 位都为 1 时，表示通过地址筛选通过的帧匹配哈希滤波或者完美滤波器 0：如果 HM 和 HU 位都为 1 时，表示帧仅匹配哈希滤波器
9	SAF	RW	0	源地址过滤(Source address filter) 源地址过滤 MAC 比较接收帧的 SA 字段和 SA 寄存器的编程值。如果匹配，接收状态字中的 SAMatch 位拉高 1：如果源地址匹配失败，MAC 丢掉该帧。 0：接收帧，但还会进行地址比较判断，并更新转发 SAMatch
8	SAIF	RW	0	源地址反向过滤(Source address inverse filtering) 1：源地址匹配的帧认为是过滤失败 0：源地址不匹配的帧认为是失败
7:6	PCF	RW	0	通过控制帧（Pass control frames) 控制所有控制帧（单播或多播暂停帧）的转发 00 或 01：不转发控制帧到应用 10：转发所有的控制帧，即使未通过地址滤波 11：转发通过地址滤波的控制帧
5	BFD	RW	0	禁止广播帧（Broadcast frames disable) 1：地址过滤器过滤所有的广播帧 0：通过接收广播帧
4	PAM	RW	0	通过所有的多播帧(Pass all multicast) 1：通过所有的广播帧 0：过滤广播帧，过滤方式取决于 HM 位
3	DAIF	RW	0	目的地址反向过滤（Destination address inverse filtering) 1：地址检查模块会对单播和多播帧进行 DA 地址进行反向过滤 0：无滤波动作
2	HM	RW	0	多播帧哈希过滤(Hash multicast) 1：MAC 会对接收的多播帧通过哈希列表进行目标地址滤波 0: MAC 会对接收的多播帧进行完美地址滤波 ，通过接收帧的 DA 字段和 DA 寄存器中的配置值进行比较
1	HU	RW	0	单播帧哈希过滤(Hash unicast) 1：MAC 会对接收的单播帧通过哈希列表进行目标地址滤波 0: MAC 会对接收的单播帧进行完美地址滤波 ，通过接收帧的 DA 字段和 DA 寄存器中的配置值进行比较
0	PM	RW	0	混合模式(Promiscuous mode) 1：无论输入帧的目标地址和源地址是什么，均接收，SA/DA 滤波失败的标志位会一直被清除

33.7.3 以太网 MAC 哈希表高位寄存器（ETH_MACHTHR）

Address offset:0x0008

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTH[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
HTH[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	HTH	RW	0	哈希表高位（Hash table high） 包含了多播哈希表的高 32 位

33.7.4 以太网 MAC 哈希表低位寄存器（ETH_MACHTLR）

Address offset:0x000C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTL[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
HTL[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	HTL	RW	0	哈希表低位（Hash table low） 包含了多播哈希表的低 32 位

33.7.5 以太网 MAC MII 地址滤寄存器（ETH_MACMIIAR）

Address offset:0x0010

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			Res.
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA					MR					Res.	CR			MW.	MB
RW	RW	-	RW	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:11	PA	-	-	PHY 地址(PHY address) 选择 PHY 设备，最多支持 32 个 PHY 地址设备
10:6	MR			MII 接口寄存器(MII register) 选择 PHY 设备的 MII 寄存器
5:2	CR	RW	0	时钟范围（Clock range) 选择基于不同 HCLK 频率下的 MDC 时钟 bit[5]=0 时： 0000：60-100Mhz HCLK/42 0001：100-150Mhz HCLK/62 0010：20-35Mhz HCLK/16 0011：35-60Mhz HCLK/26 0100：150-250Mhz HCLK/102 0101：250-300Mhz HCLK/104

				0101、0111：保留 bit[5]=1 时，可以配置大于 2.5Mhz 的 MAC 时钟频率。 1000：HCLK/4 1001：HCLK/6 1010：HCLK/8 1011：HCLK/10 1100：HCLK/12 1101：HCLK/14 1110：HCLK/16 1111：HCLK/18
1	MW.	RW	0	MII 写(MII write) 1：告诉 PHY，将会通过 MII 数据寄存器进行写操作 0：将会进行读操作，将数据存到 MII 数据寄存器中
0	MB	RW	0	MII 忙(MII busy) 在写 ETH_MACMIIAR 和 ETH_MACMIIDR 寄存器之前该位必须读出为 0，在写 ETH_MACMIIAR 过程中，该位也必须要被复位为 0.在 PHY 寄存器访问期间，该位可以置 1 来显示正在进行读写操作。在写操作中，ETH_MACMIIDR 中的数据必须保持有效直到该位被 MAC 清除。而在读的过程中，ETH_MACMIIDR 中的数据直到该位被 MAC 清除后才有效。在清除该位前 ETH_MACMIIAR 不能被写入。

33.7.6 以太网 MAC MII 数据寄存器（ETH_MACMIIDR）

Address offset:0x0014

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	
15:0	MD	RW	0	MII 数据（MII data） MII 数据寄存器存储写入和读出 PHY 的 16 位数据

33.7.7 以太网 MAC 流控寄存器（ETH_MACFCR）

Address offset:0x0018

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res	Res	Res	Res	Res	Res	Res	ZQPD	Res	PLT		UPFD	RFCE	TFCE	FCB/BPA
-	-	-	-	-	-	-	-	RW	-	RW	RW	RW	RW	RW	RW_W1/RW

Bit	Name	R/W	Reset Value	Function
31:16	PT	RW	0	暂停时间(Pause time) 规定发送的控制帧的暂停时间字段所对应的值
15:8	保留	-	-	-
7	ZQPD	RW		禁止零量子暂停(Zero-quanta pause disable) 1：禁止在 FIFO 层的流量控制信号解除断言时自动生成零量子暂停控制帧 0：使能自动生成零量子暂停控制帧
6	保留	-	-	-
5:4	PLT	RW	0	暂停低阈值（Pause low threshold) 用于配置暂停帧中的暂停定时器的最低阈值，在该阈值时间内暂停帧将自动重新传输。阈值的值需要比 PT 中配置的值小。 例如 PT=0x100（256 个 SLOT 时间值），PLT=01，则在 228slot（256-28）时间时会重发暂停帧 00 ： 暂停时间减去 4 个 slot 时间 01 ： 暂停时间减 28 个 slot 时间 10 ： 暂停时间减 144 个 slot 时间 11 ： 暂停时间减 256 个 slot 时间
3	UPFD	RW	0	单播暂停帧检测(Unicast pause frame detect) 1: MAC 检测 ETH_MACA0HR 和 ETH_MACA0LR 寄存器中定义的单播地址的暂停帧，或者特殊多播地址的暂停帧 0：仅检测 802.3x 标准中规定的特殊多播地址的暂停帧
2	RFCE	RW	0	接收流量控制使能(Receive flow control enable) 1：MAC 对接收的暂停帧进行解码并在一段规定的时间暂停接收器（PT） 0：不使能暂停帧的解码
1	TFCE	RW	0	发送流量控制使能(Transmit flow control enable) 1：在全双工模式中，MAC 在发送暂停帧时使能流量控制 ；在半双工模式中使能反压模式 0: 在全双工模式中，不使能流量控制，不发暂停帧; 在半双工模式中不使能反压模式
0	FCB/BPA	RW	0	流量控制忙/反压有效(Flow control busy/back pressure activate) 该位在全双工模式中初始化一个暂停帧 ；在半双工模式中如果 TFCE 为 1 时激活反压功能。 在全双工模式中，该位在写流量控制寄存器前应该读为 0，为了产生一个暂停帧，该位需要被写为 1。在发送控制帧时，该位保持为 1 来指示正在发送帧。当完成暂停控制帧的发送，MAC 复位该位到 0。在该位被清除之前，不能写流量控制寄存器 在半双工模式中，当该位为 1 时，反压功能有效，在反压中，当 MAC 接收新的帧，发送器开始发送 JAM 序列来产生一个冲突。 当 MAC 被配置为全双工模式，BPA 自动禁止

33.7.8 以太网 MAC VLAN 标签寄存器（ETH_MACVLANTR）

Address offset:0x001C

Reset value：0x0000 0000

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

VLANTI															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

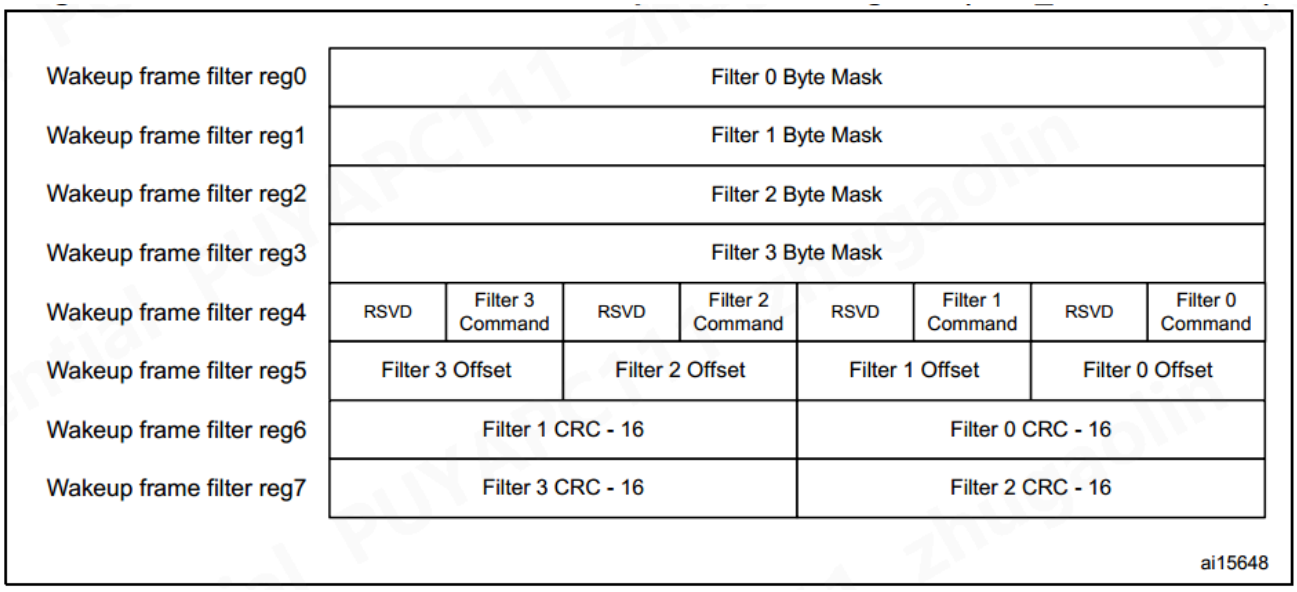
Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
16	VLANTC	RW		12 位 VLAN 标签比较(12-bit VLAN tag comparison) 1：使用 12 位 VLAN 标识符进行比较和筛选 ，而不是 16 位 VLAN 标识符 bits[11:0]与接收到的 VLAN 标签帧中的对应字段进行比较 0：VLAN 帧中的第 15 和第 16 字节的 16 位数据被用于做比较
15:0	VLANTI	RW	0	VLAN 标签标识符（VLAN tag identifier) 它包含了 802.1Q VLAN 标签来识别 VLAN 帧，并与接收的 VLAN 帧的第 15 和 16 字节进行比较。Bit[15:13]是用户优先级，bit[12]是规范格式指示符（CFI），bit[11:0]是 VLAN 标签的 VLAN 标识符字段。当设置 VLANTC 位时，只有 VID（bit[11：0]）被用于比较

33.7.9 以太网 MAC 远程唤醒帧过滤寄存器（ETH_MACRWUFFR）

Address offset:0x0028

Reset value：0x0000 0000

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（使用同一个偏移地址）。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



33.7.10以太网 MAC 控制和状态寄存器（ETH_MACPMTCSR）

Address offset:0x002C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WFFRPR.	Res.	Res.	RWKPTR					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-	-	-	RO	RO	RO	RO	RO	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	GU	Res.	Res.	WFR	WPR	Res.	Res.	WFE	MPE	PD

-	-	-	-	-	-	RW	-	-	RW	RW	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	WFFRPR.	RW	0	唤醒帧过滤寄存器指针复位(Wakeup frame filter register pointer reset) 1：复位远程唤醒帧过滤寄存器的指针到 0b000，在一个周期后自动拉低
30:29	保留	-	-	-
28:24	RWKPTR	RO		远程唤醒 FIFO 指针 表示远程唤醒帧过滤寄存器的指针的当前值
23:10	保留	-	-	-
9	GU			全局单播 bal unicast) 1：使能通过 MAC 地址过滤的单播包成为唤醒帧
8:7	保留	-	-	-
6	WFR	RW	0	唤醒帧被接收（Wakeup frame received) 1：表示电源管理事件是由接到到的唤醒帧产生的，可读清
5	WPR	RW	0	单播暂停帧检测(Magic packet received) 1：表示电源管理事件是由接到到的魔法包产生的，可读清
4:3	保留	-	-	-
2	WFE	RW	0	唤醒帧使能（Wakeup frame enable) 1：使能通过唤醒帧的接收产生的 PMT 事件
1	MPE	RW	0	魔法包使能（Magic Packet enable) 1：使能通过魔法包的接收产生的 PMT 事件
0	PD	RW	0	断电模式（Power down) 1：所有接收的帧都被丢掉，当接收到魔法帧和唤醒帧时自动清零，并且断电模式关闭。在该位被清除后接收到的帧发送给上层应用。仅当魔法包使能或唤醒帧使能时，才必须置位该位

33.7.11以太网 MAC 中断状态寄存器（ETH_MACSR）

Address offset:0x0038

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSTS	Res.	Res.	MMCTS	MMCRS	MMCS	PMTS	Res.	Res.	Res.
-	-	-	-	-	-	RC_F	-	-	R	R	R	R	-	-	-

Bit	Name	R/W	Reset Value	Function
30:10	保留	-	-	-
9	TSTS	RW	0	时间戳触发状态(Time stamp trigger status) 当系统时间等于或者大于目标时间寄存器（高/低）中规定的时间时 ，该位被置 1。该位可通过读 ETH_PTPTSSR 寄存器清零
8:7	保留	-	-	-
6	MMCTS	RW	0	MMC 发送状态（MMC transmit status) ETH_MMCTIR 寄存器发生任意中断时该位会拉高，当中断寄存器 ETH_MMCTIR 中所有位都清除时，该位被清除
5	MMCRS	RW	0	MMC 接收状态(MMC receive status)

				ETH_MMCRIR 寄存器发生任意中断时该位会拉高，当中断寄存器 ETH_MMCRIR 中所有位都清除时，该位被清除
4	MMCS	RW	0	MMC 状态(MMC status) 当 MMCTS 或 MMCRS 任意一位为高时置 1，当两位都为低时为 0
3	PMTS	RW	0	PMT 状态（PMT status) 当在断电模式下接收到魔法包或者唤醒 LAN 帧时，该位拉高。当 ETH_MACPMTCSR 寄存器的 bit5 和 6 被读清后，该位被清零
2:0	保留	-	-	-

33.7.12以太网 MAC 中断屏蔽寄存器（ETH_MACIMR）

Address offset:0x003C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
30:10	保留	-	-	-
9	TSTIM	RW	0	时间戳触发中断屏蔽(Time stamp trigger interrupt mask) 如果此位置 1，则会禁止生成时间戳中断
8:7	保留	-	-	-
3	PMTIM	RW	0	PMT 中断屏蔽（PMT interrupt mask) 如果此位置 1，则会禁止将 ETH_MACSR 中的 PMT 状态位置 1 二触发中断信号。
2:0	保留	-	-	-

33.7.13以太网 MAC 地址 0 高位寄存器（ETH_MACA0HR）

Address offset:0x0040

Reset value:0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MO	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			Res.
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA0H															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	MO			始终为 1
30:16	保留	-	-	-
15:0	MACA0H	RW	FFFF	MAC 地址 0 的高 16 位（MAC address0 high [47:32]) 包含 6 字节 MAC 地址 0 的高 16 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制（暂停）帧时插入 MAC 地址

33.7.14以太网 MAC 地址 0 低位寄存器（ETH_MACA0LR）

Address offset:0x0040

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACA0L[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
MACA0L[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	MACA0L[31:0]	RW	FFFFFFFF	MAC 地址 0 的低 32 位（MAC address0 low [31:0]) 包含 6 字节 MAC 地址 0 的低 32 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制帧时插入 MAC 地址

33.7.15以太网 MAC 地址 1 高位寄存器（ETH_MACA1HR）

Address offset:0x0048

Reset value:0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA.	MBC.						Res.	Res.	Res.	Res.	Res.			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
MACA1H															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	AE	RW	0	地址使能（Address enable) 1：通过 MAC 地址 1 进行地址完美过滤 0：地址过滤器不进行地址过滤
30	SA	RW	0	源地址（Source address) 1：MAC 地址 1[47:0]用于接收帧的 SA 字段比较， 0：MAC 地址 1[47:0]用于接收帧的 DA 字段比较
29:24	MBC	RW	0	屏蔽字节控制（Mask byte control) 每个 MAC 地址字节的比较时的屏蔽控制，置 1 时 MAC 不比较地址 1 中的 DA/SA 字段 位 29：屏蔽 ETH_MACA1HR [15:8]字段 位 28：屏蔽 ETH_MACA1HR [15:8]字段 位 27：屏蔽 ETH_MACA1LR [31:24]字段 … 位 24：屏蔽 ETH_MACA1LR [7:0]字段
23:16	保留	-	-	-
15:0	MACA1H	RW	FFFF	MAC 地址 1 的高 16 位（MAC address1 high [47:32]) 包含 6 字节 MAC 地址 1 的高 16 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制（暂停）帧时插入 MAC 地址

33.7.16以太网 MAC 地址 1 低位寄存器（ETH_MACA1LR）

Address offset:0x004C

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACA1L[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
MACA1L[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	MACA1L	RW	FFFFFFFF	MAC 地址 1 的低 32 位（MAC address1 low [31:0]) 包含 6 字节 MAC 地址 1 的低 32 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制帧时插入 MAC 地址

33.7.17以太网 MAC 地址 2 高位寄存器（ETH_MACA2HR）

Address offset:0x0050

Reset value:0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA.	MBC.						Res.	Res.	Res.	Res.	Res.			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
MACA2H															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	AE	RW	0	地址使能（Address enable) 1：通过 MAC 地址 2 进行地址完美过滤 0：地址过滤器不进行地址过滤
30	SA	RW	0	源地址（Source address) 1：MAC 地址 2[47:0]用于接收帧的 SA 字段比较， 0：MAC 地址 2[47:0]用于接收帧的 DA 字段比较
29:24	MBC	RW	0	屏蔽字节控制（Mask byte control) 每个 MAC 地址字节的比较时的屏蔽控制，置 1 时 MAC 不比较地址 2 中的 DA/SA 字段 位 29：屏蔽 ETH_MACA2HR [15:8]字段 位 28：屏蔽 ETH_MACA2HR [15:8]字段 位 27：屏蔽 ETH_MACA2LR [31:24]字段 … 位 24：屏蔽 ETH_MACA2LR [7:0]字段
23:16	保留	-	-	-
15:0	MACA2H	RW	FFFF	MAC 地址 2 的高 16 位（MAC address2 high [47:32]) 包含 6 字节 MAC 地址 2 的高 16 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制（暂停）帧时插入 MAC 地址

33.7.18以太网 MAC 地址 2 低位寄存器（ETH_MACA2LR）

Address offset:0x0054

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACA2L[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
MACA2L[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	MACA2L	RW	FFFFFFFF	MAC 地址 2 的低 32 位（MAC address2 low [31:0]) 包含 6 字节 MAC 地址 2 的低 32 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制帧时插入 MAC 地址

33.7.19以太网 MAC 地址 3 高位寄存器（ETH_MACA3HR）

Address offset:0x0058

Reset value:0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA.	MBC.						Res.	Res.	Res.	Res.	Res.			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
MACA3H															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	AE	RW	0	地址使能（Address enable) 1：通过 MAC 地址 3 进行地址完美过滤 0：地址过滤器不进行地址过滤
30	SA	RW	0	源地址（Source address) 1：MAC 地址 3[47:0]用于接收帧的 SA 字段比较， 0：MAC 地址 3[47:0]用于接收帧的 DA 字段比较
29:24	MBC	RW	0	屏蔽字节控制（Mask byte control) 每个 MAC 地址字节的比较时的屏蔽控制，置 1 时 MAC 不比较地址 3 中的 DA/SA 字段 位 29：屏蔽 ETH_MACA3HR [15:8]字段 位 28：屏蔽 ETH_MACA3HR [15:8]字段 位 27：屏蔽 ETH_MACA3LR [31:24]字段 … 位 24：屏蔽 ETH_MACA3LR [7:0]字段
23:16	保留	-	-	-
15:0	MACA3H	RW	FFFF	MAC 地址 3 的高 16 位（MAC address3 high [47:32]) 包含 6 字节 MAC 地址 3 的高 16 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制（暂停）帧时插入 MAC 地址

33.7.20以太网 MAC 地址 3 低位寄存器（ETH_MACA3LR）

Address offset:0x005C

Reset value:0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACA3L[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
MACA3L[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	MACA3L	RW	FFFFFFFF	MAC 地址 3 的低 32 位（MAC address3 low [31:0]) 包含 6 字节 MAC 地址 3 的低 32 位，用于 MAC 对接收帧进行地址过滤以及在发送流量控制帧时插入 MAC 地址

33.7.21以太网 MMC 控制寄存器（ETH_MMCCR）

Address offset:0x0100

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res			Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTL	CNTPRST.	MCF	ROR	CSR.	CR.
-	-	-	-	-	-	-	RW	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	UCDBC	RW	0	计数 MMC 丢弃的广播帧 1：更新计数器 0：不更新计数器
7:6	保留	-	-	-
5	CNTPRSTL	RW	0	满./半满预置 1：当 CNTPRST 为 1，帧计数器预置为 0xFFFF_FFF0 0：当 CNTPRST 为 1，帧计数器预置为 0x7FFF_FFF0
4	CNTPRST.	RW	0	计数器预置 置 1 时所有计数器被初始化或预置为满/半满的值，取决于 CNTPRSTLVL，一个周期后自清
3	MCF	RW	0	MMC 计数器冻结（MMC counter freeze） 1：冻结所有的 MMC 计数器并保持当前值。（当该位未被清除时，计数器不会随着接收帧和发送帧而更新。如果在 ROR 位为 1 时读任何 MMC 计数器，该计数器会被清除）
2	ROR	RW	0	读清除（ Reset on read） 1: MMC 计数器在读操作后会被清零，当最低有效字节[7:0]被读时计数器被清零
1	CSR	RW	0	计数器停止翻转（ Counter stop rollover） 1：计数器在到达最大值后不会翻转到 0

0	CR.	RW	0	计数器复位（Counter reset） 1：所有计数器被复位，一个周期后该位自清

33.7.22以太网 MMC 接收中断寄存器（ETH_MMCRIR）

Address offset:0x0104

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	RGUFS	RC_R	0	接收的好的单播帧状态（ Received Good Unicast Frames Status） 当接收好的单播帧时，计数达到最大值的一半时，该位拉高
16:7	保留	-	-	-
6	RFAES	RC_R	0	接收的对齐错误帧状态（ Received frames alignment error status） 当接收对齐错误帧时，计数达到最大值的一半时，该位拉高
5	RFCES	RC_R	0	接收的 CRC 错误帧状态（ Received frames CRC error status） 当接收 CRC 错误帧时，计数达到最大值的一半时，该位拉高
4:0	保留	-	-	-

33.7.23以太网 MMC 发送中断寄存器（ETH_MMCTIR）

Address offset:0x0108

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21	TGFS	RC_R	0	发送的好帧状态（ Transmitted good frames status） 当发送的是好帧时，计数达到最大值的一半时，该位拉高
20:16	保留	-	-	-
15	TGFMSCS.	RC_R	0	多个冲突后发送好帧的状态（ Transmitted good frames more single collision status） 当在多个冲突之后发送好帧，计数达到最大值的一半时，该位拉高
14	TGFSCS	RC_R	0	单个冲突后发送好帧的状态（ Transmitted good frames single collision status） 当在单个冲突之后发送好帧，计数达到最大值的一半时，该位拉高

13:0	保留	-	-	-

33.7.24以太网 MMC 接收中断屏蔽寄存器（ETH_MMCRIMR）

Address offset:0x010C

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	RGUFM	RW	0	接收的好的单播帧屏蔽（ Received Good Unicast Frames mask） 1：当接收好的单播帧时，计数达到最大值的一半时，不产生中断
16:7	保留	-	-	-
6	RFAEM	RW	0	接收的对齐错误帧屏蔽（ Received frames alignment error mask） 1：当接收对齐错误帧时，计数达到最大值的一半时，不产生中断
5	RFCEM	RW	0	接收的 CRC 错误帧屏蔽（ Received frames CRC error mask） 1：当接收 CRC 错误帧时，计数达到最大值的一半时，不产生中断
4:0	保留	-	-	-

33.7.25以太网 MMC 发送中断屏蔽寄存器（ETH_MMCTIMR）

Address offset:0x0110

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	-	-
21	TGFM	RW	0	发送的好帧屏蔽（ Transmitted good frames mask） 1：当发送的是好帧时，计数达到最大值的一半时，不产生中断
20:16	保留	-	-	-
15	TGFMSCM.	RW	0	多个冲突后发送好帧的屏蔽（ Transmitted good frames more single collision mask） 1：当在多个冲突之后发送好帧，计数达到最大值的一半时，不产生中断
14	TGFSCM	RW	0	单个冲突后发送好帧的屏蔽（ Transmitted good frames single collision mask） 1：当在单个冲突之后发送好帧，计数达到最大值的一半时，不产生中断
13:0	保留	-	-	-

33.7.26以太网 MMC 单次冲突后发送好帧计存器（ETH_MMCTGFSCCR）

Address offset:0x014C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFSCC[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
TGFSCC[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TGFSCC	RW	0	在单个冲突后发送好帧计数（ Transmitted good frames single collision counter/Transmitted good frames after a single collision counter）

33.7.27以太网 MMC 多次冲突后发送好帧计数寄存器（ETH_MMCTGFMSCCR）

Address offset:0x0150

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFMSCC[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
TGFMSCC[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TGFMSCC	RW	0	在多个冲突后发送好帧计数

33.7.28以太网 MMC 发送好帧计数寄存器（ETH_MMCTGFR）

Address offset:0x0168

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TGFC[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
TGFC[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TGFC	RW	0	发送好帧计数（Transmitted good frames counterr）

33.7.29以太网 MMC CRC 错误接收帧计数寄存器（ETH_MMCRFCECR）

Address offset:0x0168

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFCEC[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
RFCEC [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	RFCEC	RW	0	接收的 CRC 错误帧计数（Received frames CRC error counter Received frames with CRC error counter

33.7.30以太网 MMC 对齐错误接收帧计数寄存器（ETH_MMCRFAECR）

Address offset:0x0198

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFAEC[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
RFAEC [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	RFAEC	RW	0	接收的对齐错误帧计数（Received frames alignment error counter Received frames with alignment error counter）

33.7.31以太网 MMC 对齐错误接收帧计数寄存器（ETH_MMCRFAECR）

Address offset:0x01C4

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RGUFC[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
RGUFC[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	RGUFC	RW	0	接收的好的单播帧计数（Received good unicast frames counter）

33.7.32以太网 PTP 时间戳控制寄存器（ETH_PTPTSCR）

Address offset:0x0700

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res			Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
-	-	-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	-	-
5	TSARU	RW	0	时间戳假发寄存器更新（Time stamp addend register update）

				1：时间戳 addend 寄存器内的数据被更新到 PTP 模块中进行精细校正。当更新完成后该位会被清除。在设置该寄存器前必须读取为 0。
4	TSITE	RW	0	时间戳中断触发使能（Time stamp interrupt trigger enable） 1：当系统时间比目标时间寄存器中的值大时，产生时间戳中断。当生成时间戳触发中断时，该位被清除
3	TSSTU	RW	0	时间戳系统时间更新（Time stamp system time update） 1：系统时间被更新为时间戳高更新和低更新寄存器中的值。在设置该寄存器前必须读取为 0。当硬件更新完成后，该位会清除
2	TSSTI	RW	0	时间戳系统时间初始化（Time stamp system time initialize） 1：系统时间被初始化为时间戳高更新和低更新寄存器中的值。在设置该寄存器前必须读取为 0。当完初始化后，该位会清除。
1	TSFCU	RW	0	时间戳精细和粗略更新（Time stamp fine or coarse update） 1：系统时间戳通过精细方法更新 0：系统时间戳通过粗略方法更新
0	TSE	RW	0	时间戳使能（Time stamp enable） 1：在发送帧和接收帧中使能时间戳 0：时间戳功能被挂起并且在发送和接收帧中时间戳不会累加。因为被保持住的系统时间已暂停，在设置该位为 1 后，你需要初始化时间戳特性（系统时间）

33.7.33以太网 PTP 亚秒递增寄存器（ETH_PTPSSIR）

Address offset:0x0700

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	STSSI	RW	0	系统时间亚秒级递增（System time subsecond increment） 编程该寄存器用于每个更新周期时将亚秒级的值加到系统时间上。例如: 为了达到 20ns 的精度，该值=20/0.467 约等于 43（0x2A）

33.7.34以太网 PTP 时间戳高位寄存器（ETH_PTPTSHR）

Address offset:0x0708

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STS[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
STS[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	STS	RW	0	系统时间秒（System time second）

				该寄存器值显示当前由控制器保存的系统时钟秒值

33.7.35以太网 PTP 时间戳低位寄存器（ETH_PTPTSLR）

Address offset:0x070C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STPNS	STSS[30:16]														
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
STSS[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	STPNS	RW	0	系统正时间或负时间指示（System time positive or negative sign） 1：时间为负值 0：时间为正值。因为时间应该总是正的，所以该位通常为 0
30:0	STSS	RW	0	系统时间亚秒（System time subseconds） 该字段值表示亚秒时间，精度为 0.46 ns

33.7.36以太网 PTP 时间戳高位更新寄存器（ETH_PTPTSHUR）

Address offset:0x0710

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSUS[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
TSUS[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TSUS	RW	0	时间戳更新秒（Time stamp update second） 该字段的值表示系统被初始化或更新的秒级时间

33.7.37以太网 PTP 时间戳低位更新寄存器（ETH_ PTPTSLUR）

Address offset:0x0710

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSUPNS	TSUSS[30:16]														
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
TSUSS[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	TSUPNS	RW	0	时间戳更新正值和负值的指示（Time stamp update positive or negative sign） 1：时间为负

				0：时间为正 当 TSSTI 被置 1，该位应该被清零，如果此时该位被置 1，那么系统时间会减去时间戳更新寄存器的值，反之，系统时间会加上时间戳更新寄存器的值
30:0	TSUSS	RW	0	时间戳更新亚秒（ Time stamp update subseconds） 该字段的值表示系统被初始化或更新的亚秒级时间。该值精度为 0.46ns（0x0000_0001 即 0.46ns）

33.7.38以太网 PTP 时间戳加数寄存器（ETH_PTPTSAR）

Address offset:0x0718

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSA[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
TSA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TSA	RW	0	时间戳加数（Time stamp addend） 该寄存器表示 32 位的值被加到计数寄存器中来进行事件的同步

33.7.39以太网 PTP 时间戳目标时间高位寄存器（ETH_PTPTTHR）

Address offset:0x071C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSH[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
TTSH[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TTSH	RW	0	目标时间戳高位（Target time stamp high） 该寄存器存储了秒级事件值。当时间戳的值等于或者超过目标事件值时，MAC 会产生中断

33.7.40以太网 PTP 时间戳目标时间低位寄存器（ETH_PTPTTLR）

Address offset:0x0720

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSL[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
TTSL[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TTSL	RW	0	目标时间戳低位（Target time stamp low）

				该寄存器存储了亚秒级事件值。当时间戳的值等于或者超过目标事件值时，MAC 会产生中断

33.7.41以太网 DMA 总线模式寄存器（ETH_DMABMR）

Address offset:0x1000

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RIB	Res.	Res.	Res.	TXPR	MB	AAB.	FPM	USP	RDP						FB
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
PM		PBL						Res.	DSL					DA	SR
RW	RW	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	RIB	RW	0	INCRx Burst 的重建 1：当 AHB 主接口得到 EBT（Retry，Split，Losing bus grant。AHB 主接口可重建由 INCRx 启动的 burst 传输的 pending beats，AHB 主接口使用 INCRx 和 SINGLE 的指定 burst 组合重建 beats，默认情况下，AHB 主接口重建具有未指定（INCR）突发的 EBT 的 pending beats
30:28	保留	-	-	-
27	TXPR	RW	0	发送器优先级 1：当设置时，此位表示在系统侧总线的仲裁期间，发送 DMA 的优先级高于接收 DMA
26	MB	RW	0	混合突发 1：当 FB 为 0 时，AHB 主接口以 INCR（未定义突发）开始长度大于 16 的所有 burst。而对于 16 及以下的 burst 长度，它恢复到固定 burst 传输（INCRx 和 SINGLE）
25	AAB.	RW	0	地址对齐（Address-aligned beats） 1：当 FB 等于 1 时，AHB 接口会生成所有的突发与 LS 位的起始地址对齐。如果 FB 等于 0，第一个突发是不对齐的，后续的突发与地址对齐
24	FPM	RW	0	4 倍 PBL 模式（4xPBL mode） 1：该位将编程的 PBL 值（位[22:17]和位[13:8]）乘以四倍，因此，DMA 传输的数据最多为 4、8、16、32、64 和 128 拍取决于 PBL 值。
23	USP	RW	0	使用独立的 PBL 值（Use separate PBL） 1：配置 RxDMA 使用 RDP 的值作为 PBL，而实际 PBL 的配置值仅用于 TxDMA。 0：PBL 的值用于 RxDMA 和 TxDMA
22:17	RDP	RW	0	RxDMA 的突发长度（ Rx DMA PBL） 这些位表示在一个 RxDMA 事务中要传输的最大节拍数。这是单个读/写操作中使用的最大值。每次在主机总线上启动突发传输时，RxDMA 总是尝试按照 RDP 中指定的方式进行突发。RDP 可以使用允许值 1、2、4、8、16 和 32 进行编程，其它的值是无效的。 注意：仅当 USP 为 1 时有效
16	FB	RW	0	固定的突发（ Fixed burst） 1：AHB 接口使用固定的突发传输，包括 SINGLE, INCR4, INCR8 or INCR16 0：AHB 接口使用 SINGLE, INCR 突发模式

15:14	PM	RW	0	Rx Tx 优先级比例（Rx Tx priority ratio） 00：1:1 01：2:1 10：3:1 11：4:1 注意：仅当 DA 位清零时有效
13:8	PBL	RW	0	突发长度（Programmable burst length） 这些位表示在一次 DMA 事务中要传输的最大节拍数。这是单个读/写操作中使用的最大值。每次在主机总线上启动突发传输时，RxDMA 总是尝试按照 RDP 中指定的方式进行突发。RDP 可以使用允许值 1、2、4、8、16 和 32 进行编程，其它的值是无效的。 注意：仅当 USP 为 1 时有效 PBL 值的初始化需要注意以下几点： 1.PBL 的最大值根据 TxFIFO 和 RxFIFO 的大小 2.FIFO 有一个限制，即支持的最大节拍是 FIFO 深度的一半 3.如果 PBL 用于接收和发送 DMA，最小的 FIFO 深度需要考虑 4.不要编程超出范围的 PBL 值，否则系统可能无法正常运行
7	保留	-	-	-
6:2	DSL	RW	0	描述符跳过长度（ Descriptor skip length） 该位段定义了在两个非链式的描述符之间跳过的字的数量。地址跳过从当前描述符的末尾开始到下一个描述符的开头。如果 DSL 的值等于 0，在环形模式中，描述符表被 DMA 视为连
1	DA	RW	0	DMA 仲裁（ DMA Arbitration） 0：Rx:Tx 优先级以位[15:14]为单位的循环 1：RX 比 Tx 优先级高
0	SR	RW	0	软复位（Software reset） 1：MAC 的 DMA 控制器复位所有的 MAC 子层的内部寄存器和逻辑。当完成复位后自动清零。在对任何寄存器重新编程之前，此位中值读出为 0。

33.7.42以太网 DMA 发送查询请求寄存器（ETH_DMATPDR）

Address offset:0x1004

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TPD[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
TPD[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	TPD	RW	0	发送轮询请求（Transmit poll demand） 当该寄存器写入任何值时，DMA 会读取当前描述符指向的地址（ETH_DMACHTDR 寄存器）。如果描述符由 CPU 占用，发送器会返回一个挂起的状态并且 ETH_DMASR 寄存器的 bit2 会置 1.如果描述符不由 CPU 占用，发送恢复

33.7.43以太网 DMA 接收查询请求寄存器（ETH_DMARPDR）

Address offset:0x1008

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPD[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
RPD[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	RPD	RW	0	接收轮询请求（Receive poll demand） 当该寄存器写入任何值时，DMA 会读取当前描述符指向的地址（ETH_DMACHRDR 寄存器）。如果描述符由 CPU 占用，接收器会返回一个挂起的状态并且 ETH_DMASR 寄存器的 bit7 会置 1.如果描述符不由 CPU 占用，接收恢复

33.7.44以太网 DMA 接收描述符列表地址寄存器（ETH_DMARDLAR）

Address offset:0x100C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRL[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
SRL[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	SRL	RW	0	接收列表的起始（Start of receive list） 该寄存器包含了在接收描述符列表中第一个描述符的基地址。其 LSB（bit [1/2/3:0] 对应 32/64/128-bit 总线宽度）在内部会被忽略，并被 DMA 认为都是 0。因此 LSB 是只读的。

33.7.45以太网 DMA 发送描述符列表地址寄存器（ETH_DMATDLAR）

Address offset:0x1010

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STL[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
STL[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	STL	RW	0	发送列表的起始（Start of transmit list）

				该寄存器包含了在发送描述符列表中第一个描述符的基地址。其 LSB（bit [1/2/3:0] 对应 32/64/128-bit 总线宽度）在内部会被忽略，并被 DMA 认为都是 0。因此 LSB 是只读的。

33.7.46以太网 DMA 状态寄存器（ETH_DMASR）

Address offset:0x1014

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TSTS	PMTS	MMCS	Res.	EBS			TPS			RPS			NIS
-	-	R	R	R	-	R	R	R	R	R	R	R	R	R	RC_W1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIS.	ERS	FBES	Res.	Res.	ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS.	TS.
RC_W1	RC_W1	RC_W1	-	-	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	-	-
29	TSTS	R	0	时间戳触发状态（Time stamp trigger status） 该位表示 MAC 控制器的时间戳生成模块产生了中断。软件需要读 MAC 控制器状态寄存器来清除其源（bit9），然后来清零该位。
28	PMTS	R	0	PMT 中断（PMT Interrupt） 该位表示 MAC 控制器的 PMT 模块产生了中断。软件必须读取 MAC 控制器中的相应寄存器以获得中断的确切原因，并清除其源以将该位重置为 0。
27	MMCS	R	0	MMC 中断（ MMC Interrupt） 该位表示 MAC 控制器的 MMC 模块产生了中断。软件必须读取 MAC 控制器中的相应寄存器以获得中断的确切原因，并清除其源以将该位重置为 0。
26	保留	-	-	-
25:23	EBS	R	0	错误状态（ Error bits status） 该位表示会导致总线错误的错误类型。当致命总线错误位 ETH_DMASR register [13]）置 1 时有效。该位不会产生中断。 Bit23： 1：TxDMA 发送数据时出错 0：RxDMA 接收数据时出错 Bit24： 1：读操作时出错 0：写操作时出错 Bit25： 1：访问描述符时出错 0：访问数据缓存区时出错
22:20	TPS	R	0	发送进程状态（ Transmit process state） 该位段表示发送 DMA 的状态机，不生成中断 000：stopped 状态，发出复位或停止发送命令 001：ruuning 状态，获取发送描述符 010：running 状态，等待状态信息

				011：running 状态，从内存中读数据并将其排队传输到缓冲区（Tx FIFO） 100,101：保留 110：suspended 状态，发送描述符无效或者发送缓存区下溢 111：running 状态：关闭发送描述符
19:17	RPS	R	0	接收进程状态（ Receive process state） 该位段表示接收 DMA 的状态机，不生成中断 000：stopped 状态，发出复位或停止接收命令 001：ruuning 状态，获取接收描述符 010：保留 011：running 状态，等待接收数据包 100：suspended 状态，接收描述符无效 101：running 状态：关闭发送描述符 110：保留 111：running 状态，从 Rx FIFO 中发数据到系统内存
16	NIS	RC_W1	0	常规中断和（ Normal interrupt summary） 该位段表示接收 DMA 的状态机，不生成中断 该位为下列中断的或逻辑： – ETH_DMASR [0]：发送中断 – ETH_DMASR [2]：发送缓存无效 – ETH_DMASR [6]：接收中断 – ETH_DMASR [14]：早接收中断 仅未屏蔽的中断可影响常规中断和 这是一个粘性位，每次清除导致 NIS 设置的相应位时，都必须清除它（通过向该位写入 1）
15	AIS.	RC_W1	0	非常规中断和（ Abnormal interrupt summary） 该位段表示接收 DMA 的状态机，不生成中断 该位为下列中断的或逻辑： – ETH_DMASR [1]:发送进程停止 – ETH_DMASR [3]:发送 jabber 定时器溢出 – ETH_DMASR [4]：接收 FIFO 上溢 – ETH_DMASR [5]：发送下溢 – ETH_DMASR [7]：接收缓存无效 – ETH_DMASR [8]：接收进程停止 – ETH_DMASR [9]：接 WDT 溢出 – ETH_DMASR [10]：早发送中断 – ETH_DMASR [13]：致命总线错误 仅未屏蔽的中断可影响常规中断和 这是一个粘性位，每次清除导致 NIS 设置的相应位时，都必须清除它（通过向该位写入 1）
14	ERS	RC_W1	0	早接收状态（Early receive status） 该位表示 DMA 已经填充了数据包的第一个数据缓存区。接收中断 ETH_DMASR[6]自动清除此位）
13	FBES	RC_W1	0	致命总线错误状态（ Fatal bus error status） 该位表示发生了致命总线错误，置 1 时，对应的 DMA 引擎会禁止所有的总线访问
12:11	保留	-	-	-

10	ETS	RC_W1	0	早发送状态（Early transmit status） 此位表示要发送的帧已完全传输到发送 FIFO。
9	RWTS	RC_W1	0	接收看门狗定时器溢出（Receive watchdog timeout status） 此位表示接收的帧长度大于 2048 个字节
8	RPSS	RC_W1	0	接收看门狗定时器溢出（Receive watchdog timeout status） 此位表示接收的帧长度大于 2048 个字节
7	RBUS	RC_W1	0	接收缓存区无效状态（Receive buffer unavailable status） 此位表示接收列表中下一个描述符被 CPU 占用，并且无法被 DMA 请求。接收进程被挂起。为了恢复处理接收描述符，CPU 需要改变描述符的拥有权并且发出接收轮询要求的命令。如果没有轮询命令发出，当接收到下一个识别的传入帧时，接收进程恢复。仅当先前的接收描述符是由 DMA 占用时，该位置 1
6	RS	RC_W1	0	接收状态（Receive status） 该位表示完成了帧接收。特定的帧状态信息已发布在描述符中。接收器仍处于运行状态
5	TUS	RC_W1	0	发送下溢状态（ Transmit underflow status） 该位表示在发送期间出现了发送缓存下溢。发送器会进入挂起状态并且 TDES0[1] 会置 1
4	ROS	RC_W1	0	接收上溢状态（ Receive overflow status） 该位表示在接收期间出现了接收缓存上溢。如果将部分帧传输到应用程序，则在 RDES0[11]中设置溢出状态。
3	TJTS	RC_W1	0	发送 jabber 定时器溢出状态（Transmit jabber timeout status） 该位表示发送 jabber 定时器溢出，这意味着发射器过于活跃。发送进程会被中止并且进入到 stopped 状态。TDES0[14] 会置 1
2	TBUS	RC_W1	0	发送缓存区无效状态（ Transmit buffer unavailable status） 此位表示发送列表中下一个描述符被 CPU 占用，并且无法被 DMA 请求。接收进程被挂起。为了恢复处理接收描述符，CPU 需要改变描述符的拥有权并且发出发送轮询要求的命令
1	TPSS.	RC_W1	0	发送进程停止状态（ Transmit process stopped status） 此位表示发送进程进入 stopped
0	TS	RC_W1	0	发送状态（ Transmit status） 此位表示帧发送完成并且第一个描述符中的 TDES1[31]置 1

33.7.47以太网 DMA 工作模式寄存器（ETH_DMAOMR）

Address offset:0x1018

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DTCEDF.	RSF	DFRF	Res.	Res.	TSF	FTF	Res.	Res.	Res.	TTC[2]
-	-	-	-	-	RW	RW	RW	-	-	RW	RW	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTC[1:0]		ST	Res.	Res.	Res.	Res.	Res.	FEF	FUGF	Res.	RTC		OSF	SR	Res.
RW	RW	RW	-	-	-	-	-	RW	RW	-	RW	RW	RW	RW	-

Bit	Name	R/W	Reset Value	Function
31:27	保留	-	-	-

26	DTCEDF	RW	0	禁止丢弃 TCP/IP 校验和错误帧（Dropping of TCP/IP checksum error frames disable） 1: 控制器不会丢弃只有接收校验和减载引擎检测到错误的帧。这种帧在 MAC 接收的以太网帧中没有任何错误（包括 FCS 错误），但仅在封装的有效载荷中有错误。 0：如果 FEF 为 0 所有错误帧都被丢弃
25	RSF	RW	0	存储转发模式（Receive store and forward） 1：当帧被完全写入 Rx FIFO 后开始读取帧，忽略 RTC 位。 0：Rx FIFO 工作在直通模式，受制于由 RTC 位指定的阈值。
24	DFRF	RW	0	禁止清空接收帧（ Disable flushing of received frames） 1：由于接收描述符/缓冲器不可用，RxDMA 不会像正常情况下清除该位时那样清除任何帧
23:22	保留	-	-	-
21	TSF	RW	0	发送存储转发模式（Transmit store and forward） 1：当帧被完全写入 Tx FIFO 后开始发送帧，忽略 TTC 位。 0：Tx FIFO 工作在直通模式，受制于由 RTTC 位指定的阈值。 注意：仅当发送停止时可以改变该位
20	FTF	RW	0	清空发送帧（Flush transmit FIFO） 1：发送 FIFO 控制器逻辑被复位到默认值并且所有在 Tx FIFO 中的数据被清空。当清空动作完成后该位自动清 0。在该位未被清零时不能写入。
19:17	保留	-	-	-
16:14	TTC	RW	0	发送阈值控制（Transmit threshold control） 该位段控制发送 FIFO 的阈值点。当在 Tx FIFO 中缓存帧的大小大于阈值时，开始发送进程。另外，如果是一个完整的帧，即使大小小于阈值也会开启发送。当 TSF 被清零时，该位段有效 000：64 001：128 010：192 011：256 100：40 101：32 101：24 101：16
13	ST	RW	0	开始/停止发送（Start/stop transmission） 1：发送进入 running 状态，并且 DMA 检查当前位置的发送列表中要发送的帧。在当前列表的位置 (发送列表的基地址由 ETH_DMATDLAR 寄存器设置) 或者从先前停止发送时保持的位置进行描述符获取的尝试。如果没有描述符被 DMA 拥有，发送会进入挂起状态并且发送缓存无效位置 1（ETH_DMASR [2]）。开始发送命令仅在发送停止时有效。如果在设置 DMA ETH_DMATDLAR 寄存器之前就发出命令，DNA 的行为是不可预知的。 0：在完成当前帧的发送后发送进程进入 stopped 状态。发送列表中的下一个描述符位置被保存，并且在重新开始传输时变为当前位置。仅当当前帧被发送完或发送进入挂起状态时停止发送命令才有效
12:8	保留	-	-	-
7	FEF	RW	0	转发错误帧（Forward error frames） 1：除了短小错误帧以外的所有帧都被转发给 DMA

				0：Rx FIFO 丢弃错误状态（CRC 错误，冲突错误，巨大帧，看门狗溢出，缓存上溢）对应的帧。然而，如果帧的起始字节指针已经被发送到读控制器侧（阈值模式中），那么该帧不会被丢弃。当帧起始字节未发送到 ARI 总线时，错误帧会被丢弃
6	FUGF	RW	0	转发小尺寸的好帧（Forward undersized good frames） 1：Rx FIFO 转发小尺寸（没有错误但长度小于 64 字节，包括填充和 crc）的帧 0：丢弃小于 64 字节的帧，除非因为接收阈值低的原因，这种帧已经被发送。
5	保留	-	-	-
4:3	RTC	RW	0	接收阈值控制（ Receive threshold control） 该位段控制接收 FIFO 的阈值点。当在 Rx FIFO 中缓存帧的大小大于阈值时，开始发送帧到内存。另外，如果是一个完整的帧，即使大小小于阈值也会开启发送。当 RSF 被清零时，该位段有效 00：64 01：32 10：96 11：128
2	OSF	RW	0	第二帧连续模式（Operate on second frame） 1：在接收到第一个帧的状态信息之前，DMA 传输第二个帧
1	SR	RW	0	开始/停止接收（Start/stop receive） 1：接收器进入 running 状态，并且 DMA 检查当前位置的发送列表中要发送的帧。在当前列表的位置（接收列表的基地址 ETH_DMATDLAR 寄存器设置）或者从先前停止发送时保持的位置进行描述符获取的尝试。如果没有描述符被 DMA 拥有，接收会进入挂起状态并且接收缓存无效位被置 1（ETH_DMASR[7]）。开始接收命令仅在接收停止时有效。如果在设置 DMA ETH_DMATDLAR 寄存器之前就发出命令，DNA 的行为是不可预知的。 0：在完成当前帧的接收后发送进程进入 stopped 状态。接收列表中的下一个描述符位置被保存，并且在重新开始接收时变为当前位置。仅当接收进程进入 suspened 或 running 状态时停止接收命令才有效
0	保留	-	-	-

33.7.48以太网 DMA 中断使能寄存器（ETH_DMAIER）

Address offset:0x101C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISE
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AISE	ERIE	FBEIE	.		ETIE.	RWTIE.	RPSIE	RBUIE	RIE	TUIE.	ROIE	TJTIE	TBUIE	TPSIE	TIE.
RW	RW	RW	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	保留	-	-	-
16	NISE	RW	0	常规中断和使能（Normal interrupt summary enable） 1：常规中断被使能 0：常规中断被禁止

				– ETH_DMASR [0]：发送中断 – ETH_DMASR [2]：发送缓存区无效 – ETH_DMASR [6]：接收中断 – ETH_DMASR [14]：早接收中断
15	AISE	RW	-	非常规中断和使能（Abnormal interrupt summary enable） 1：非常规中断被使能 0：非常规中断被禁止 – ETH_DMASR [1]：发送进程停止 – ETH_DMASR [3]：发送 jabber 定时器溢出 – ETH_DMASR [4]：接收上溢 – ETH_DMASR [5]：发送下溢 – ETH_DMASR [7]：接收缓存区无效 – ETH_DMASR [8]：接收进程停止 – ETH_DMASR [9]：接收看门狗溢出 – ETH_DMASR [10]：早发送中断 – ETH_DMASR [13]：致命总线错误
14	ERIE	RW	0	早接收中断使能（ Early receive interrupt enable） 1：当非常规中断和被使能时，早接收中断被使能 0：早接收中断被禁止
13	FBEIE	RW	0	致命总线错误中断使能（Fatal bus error interrupt enable） 1：当常规中断和被使能时，致命总线错误中断被使能 0：致命总线错误中断被禁止
12:11	保留	-	-	-
10	ETIE.	RW	0	早发送中断使能（Early transmit interrupt enable） 1：当非常规中断和被使能时，早发送中断被使能 0：早发送中断被禁止
9	RWTIE	RW	0	接收看门狗溢出中断中断使能（ receive watchdog timeout interrupt enable） 1：当非常规中断和被使能时，接收看门狗溢出中断被使能 0：接收看门狗溢出中断被禁止
8	RPSIE	RW	0	接收进程停止中断中断使能（Receive process stopped interrupt enable） 1：当非常规中断和被使能时，接收进程停止中断被使能 0：接收进程停止中断被禁止
7	RBUIE	RW	0	接收缓存无效中断中断使能（Receive buffer unavailable interrupt enable） 1：当非常规中断和被使能时，接收缓存无效中断被使能 0：接收缓存无效中断被禁止
6	RIE	RW	0	接收中断使能（Receive interrupt enable） 1：当非常规中断和被使能时，接收中断被使能 0：接收中断被禁止
5	TUIE	RW	0	下溢中断中断使能（Underflow interrupt enable） 1：当非常规中断和被使能时，下溢中断被使能 0：下溢中断被禁止
4	ROIE	RW	0	上溢中断中断使能（Overflow interrupt enable） 1：当非常规中断和被使能时，上溢中断被使能 0：上溢中断被禁止
3	TJTIE	RW	0	发送 jabber 溢出中断中断使能（Transmit jabber timeout interrupt enable） 1：当非常规中断和被使能时，发送 jabber 溢出中断被使能

				0：发送 jabber 溢出中断被禁止
2	TBUIE	RW	0	发送缓存无效中断使能（Transmit buffer unavailable interrupt enable） 1：当常规中断和被使能时，发送缓存无效中断被使能 0：发送缓存无效中断被禁止
1	TPSIE	RW	0	发送进程停止中断使能（Transmit process stopped interrupt enable） 1：当常规中断和被使能时，发送进程停止中断被使能 0：发送进程停止中断被禁止
0	TIE	RW	0	发送中断使能（Transmit interrupt enable） 1：当非常规中断和被使能时，发送中断被使能 0：发送中断被禁止

33.7.49以太网 DMA 丢失帧和缓存溢出计数寄存器（ETH_DMAMFBOCR）

Address offset:0x1020

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	-	-
28	OFOC	RC_R	0	FIFO 上溢计数器的上溢位（Overflow bit for FIFO overflow counter）
27:17	MFA	RC_R	0	应用程序错过的帧计数器（Missed frames by the application）
16	OMFC	RC_R	-	错过的帧计数器的上溢位（Overflow bit for missed frame counter）
15:0	MFC	RC_R		控制器错过的帧计数器（ Missed frames by the controller） 计数由于 CPU 接收缓存区不可用导致的被控制器错过的帧。该计数器在 DMA 每丢弃一个到来帧时进行递增。

33.7.50以太网 DMA 当前发送描述符寄存器（ETH_DMACHTDR）

Address offset:0x1048

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	HTDAP	RO	0	主设备发送描述符地址指针（Host transmit descriptor address pointer） 复位时清零，该指针在工作中由 DMA 更新

33.7.51以太网 DMA 当前接收描述符寄存器（ETH_DMACHRDR）

Address offset:0x104C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	HRDAP	RO	0	主设备接收描述符地址指针（Host receive descriptor address pointer） 复位时清零，该指针在工作中由 DMA 更新

33.7.52以太网 DMA 当前发送缓存地址寄存器（ETH_DMACHTBAR）

Address offset:0x1050

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	HTBAP	RO	0	主设备发送缓存区地址指针（Host transmit buffer address pointer） 复位时清零，该指针在工作中由 DMA 更新

33.7.53以太网 DMA 当前接收缓存地址寄存器（ETH_DMACHRBAR）

Address offset:0x1054

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	HRBAP	RO	0	主设备接收缓存区地址指针（Host receive buffer address pointer） 复位时清零，该指针在工作中由 DMA 更新

34. CAN 总线控制器

34.1 简介

控制器局域网络（CAN）子系统由两个 CAN 模块、一个共享消息 RAM 存储器组成。关于这三个组成部分的基地址，请参考章节【存储器映射】。

CAN 模块符合 ISO 11898-1：2015（CAN 协议规范第 2.0 版 A、B 部分）和 CAN FD 协议规范第 1.0 版（CAN with Flexible Data-Rate Specification Version 1.0）。

FDCAN1 遵循 CAN 总线 CAN2.0(2.0A、CAN2.0B）协议。

FDCAN2 遵循 CAN 总线 CAN2.0(2.0A、CAN2.0B）和 CAN FD 协议。

1.6KB 的消息 RAM 存储器可实现过滤器（Rx Filter）、接收 FIFO（Rx FIFO）、接收缓冲区（Rx Buffer）、发送事件 FIFO（Tx Event FIFO）、发送缓冲区（Tx Buffer）功能。该消息 RAM 在 FDCAN1 和 FDCAN2 模块之间共用。

34.1.1 主要特性

- 适用 ISO 11898-1:2015 标准
- CAN FD 最多支持收发 64 个数据字节
- 支持 CAN 错误日志
- 支持 AUTOSAR 和 SAE J1939
- 支持接收过滤功能
- 两个可配置的接收 FIFO
- 接收到高优先级消息时单独发出信号指示
- 最多 3 个专用接收缓冲区
- 最多 3 个专用发送缓冲区
- 可配置的发送 FIFO 或队列
- 可配置的发送事件 FIFO
- 主机 CPU 可直接访问消息 RAM
- FDCAN1 和 FDCAN2 两个模块共享同一个消息 RAM
- 支持可编程回环测试模式
- 支持可屏蔽模块中断
- 两个时钟域：CAN 内核时钟和 APB 总线接口时钟

34.1.2 模块框图

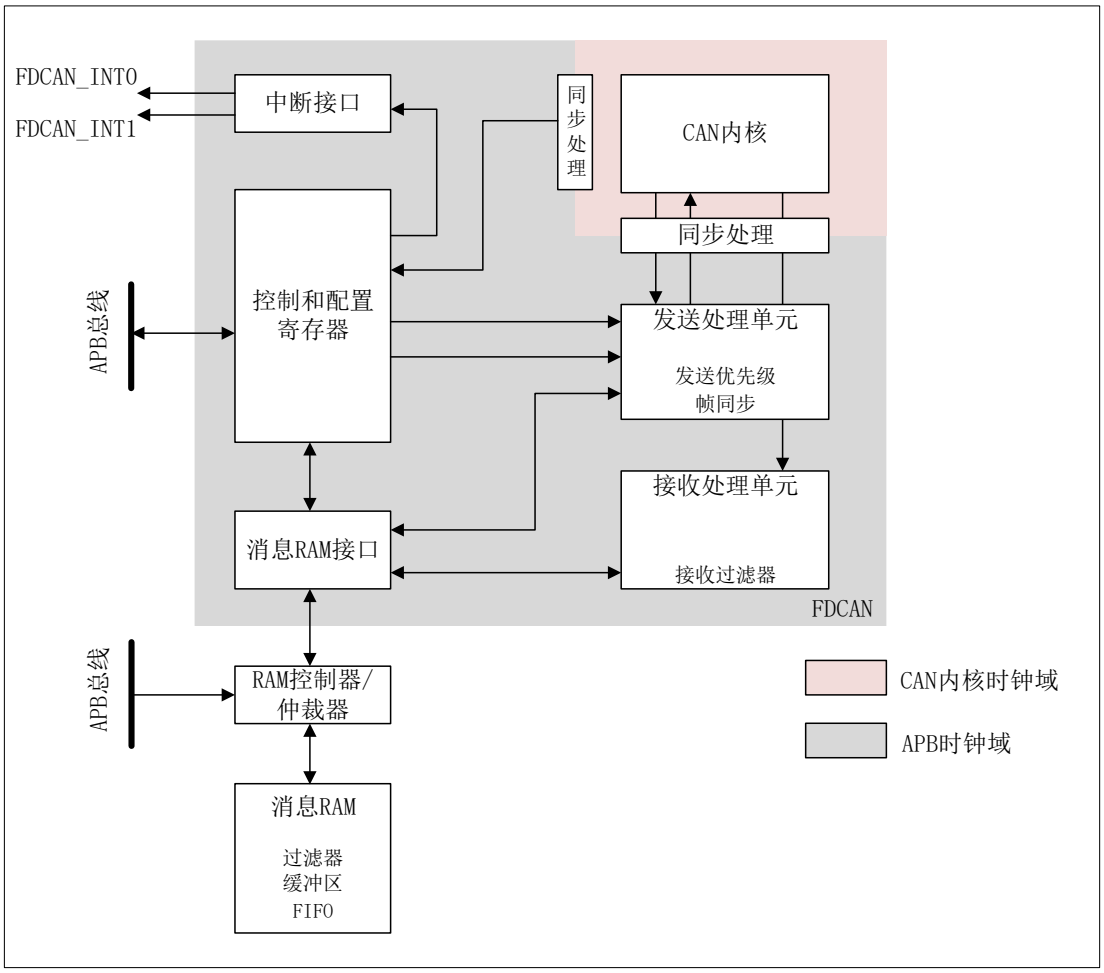


图 34-1 模块框图

CAN 内核

CAN 内核包含协议控制器、接收与发送移位寄存器。它可处理 ISO 11898-1:2015 的所有协议功能，并支持 11 位标准 ID 和 29 位扩展 ID。

同步

同步模块将来自 APB 时钟域的信号同步到 CAN 内核时钟域，以及将来自 CAN 内核时钟域的信号同步到 APB 时钟域。

时钟

APB 时钟域的时钟为 FDCAN 控制逻辑时钟，CAN 内核时钟域的时钟为 FDCAN 通信时钟，请参考章节【RCC】 。为了保证 FDCAN 通信的稳定性，控制逻辑时钟频率必须大于等于通信时钟频率。

发送处理单元

该处理单元控制消息从外部消息 RAM 到 CAN 内核的传输。最多可配置 32 个发送缓冲区进行发送。发送缓冲区可用作专用发送缓冲区、发送 FIFO（或队列），或二者的组合。发送事件 FIFO 将发送时间戳与对应的消息 ID 存储在一起。另外还支持取消发送。

接收处理单元

该处理单元控制接收到的消息从 CAN 内核到外部消息 RAM 的传输。接收处理单元支持两个接收 FIFO（每个 FIFO 的大小均可配置）和最多 64 个专用接收缓冲区，用于存储所有通过接收过滤的消息。与接收 FIFO 有所不同，专用接收缓冲区仅用于存储特定 ID 的消息。接收时间戳与其消息存储在一起。对于 11 位标准 ID，最多可定义 128 个过滤器；对于 29 位扩展 ID，最多可定义 64 个过滤器。

APB 接口

将 FDCAN 连接至 APB 总线。

消息 RAM

接口通过 RAM 控制器和仲裁器将 FDCAN 访问连接到外部 2KB 消息 RAM。

双中断线

FDCAN 提供 FDCAN_INT0 和 FDCAN_INT1 两条中断线。对 FDCAN_ILE.EINT0 和 FDCAN_ILE.EINT1 位进行编程，可分别使能或禁止这两条中断线。

34.2 FDCAN 功能描述

34.2.1 工作模式

34.2.1.1 软件初始化

软件初始化是通过置位 FDCAN_CCCR.INIT 开始的。软件复位、硬件复位或进入 Bus_Off 会置位 FDCAN_CCCR.INIT。当 FDCAN_CCCR.INIT 被置位时，CAN 总线上传入和传出的消息都将停止，FDCAN_TX 引脚的状态变为隐性（高电平）。错误逻辑管理单元（EML）的计数器保持不变。将 FDCAN_CCCR.INIT 置位不会改变任何配置寄存器。将 FDCAN_CCCR.INIT 清零可以完成软件初始化，随后，位流处理单元（BSP）等待总线上出现 11 个连续隐性位的序列（Bus_Idle），以此将其自身与 CAN 总线上的数据传输进行同步，然后才能参与总线活动并开始消息传输。

仅当 FDCAN_CCCR.INIT 和 FDCAN_CCCR.CCE 均置位时，才能访问 FDCAN 配置寄存器。FDCAN 配置寄存器如下表。

表 34-1 FDCAN 配置寄存器列表

寄存器	说明
FDCAN_DBTP	FDCAN 数据位时间和预分频寄存器
FDCAN_TEST	FDCAN 测试寄存器
FDCAN_RWD	FDCAN RAM 看门狗寄存器
FDCAN_CCCR	FDCAN CC 控制寄存器

FDCAN_NBTP	FDCAN 标称位时间和预分频寄存器
FDCAN_TSCC	FDCAN 时间戳计数器配置寄存器
FDCAN_TOCC	FDCAN 超时计数器配置寄存器
FDCAN_TDCR	FDCAN 发送器延迟补偿寄存器
FDCAN_GFC	FDCAN 全局过滤器配置寄存器
FDCAN_SIDFC	FDCAN 标准 ID 过滤器配置寄存器
FDCAN_XIDFC	FDCAN 扩展 ID 过滤器配置寄存器
FDCAN_XIDAM	FDCAN 扩展 ID 与掩码寄存器
FDCAN_RXF0C	FDCAN 接收 FIFO0 配置寄存器
FDCAN_RXBC	FDCAN 接收缓冲区配置寄存器
FDCAN_RXF1C	FDCAN 接收 FIFO1 配置寄存器
FDCAN_RXESC	FDCAN 接收缓冲区和 FIFO 元素大小配置寄存器
FDCAN_TXBC	FDCAN 发送缓冲区配置寄存器
FDCAN_TXESC	FDCAN 发送缓冲区元素大小配置寄存器
FDCAN_TXEFC	FDCAN 发送事件 FIFO 配置寄存器

FDCAN_CCCR.CCE 仅在 FDCAN_CCCR.INIT 置位时才能被置位或清零，在 FDCAN_CCCR.INIT 清零时自动清零。

FDCAN_CCCR.CCE 置位后，以下寄存器会被复位：

– FDCAN_HPMS——高优先级消息状态

– FDCAN_RXF0S——接收 FIFO 0 状态

– FDCAN_RXF1S——接收 FIFO 1 状态

– FDCAN_TXFQS——发送 FIFO/队列状态

– FDCAN_TXBRP——发送缓冲区请求挂起

– FDCAN_TXBTO——发送缓冲区发送发生

– FDCAN_TXBCF——发送缓冲区取消完成

– FDCAN_TXEFS——发送事件 FIFO 状态

– FDCAN_TXBAR——发送缓冲区添加请求

– FDCAN_TXBCR——发送缓冲区取消请求

当 CCCR.CCE 置位时，超时计数器值 TOCV.TOC 预设为 TOCC.TOP 配置的值。

以下寄存器仅在 CCCR.CCE = ‘0’ 时可写

• TXBAR - Tx 缓冲区添加请求

• TXBCR - Tx 缓冲区取消请求

仅当 FDCAN_CCCR.INIT 和 FDCAN_CCCR.CCE 均已置位时，主机才能将 FDCAN_CCCR.TEST 和 FDCAN_CCCR.MON 置位，主机可随时复位 FDCAN_CCCR.TEST 和 FDCAN_CCCR.MON。仅当 FDCAN_CCCR.INIT 和 FDCAN_CCCR.CCE 均已置位时，才能将 FDCAN_CCCR.DAR 置位或清零。

34.2.1.2 正常工作

FDCAN 完成初始化且 FDCAN_CCCR.INIT 清零后，FDCAN 会将其自身与 CAN 总线同步并准备好进行通信。接收消息时，通过接收过滤器接收到的消息（包含消息 ID 和 DLC）会存储到专用的接收缓冲区，或接收 FIF0，或接收 FIF1。发送消息时，先初始化或更新专用发送缓冲区和（或）发送 FIFO（队列）。不支持在接收到远程帧后自动发送。

34.2.1.3 CAN FD 操作

CANFD 帧传输有两种类型，第一种是不带比特率切换的 CANFD 帧，第二种是带比特率切换的 CANFD 帧（控制段、数据段和 CRC 段的传输比特率比帧开始和结束时的高）。
经典 CAN（ClassicalCAN）标准帧中的 r0 位（IDE 位之后，标准帧格式中的保留位）、扩展帧中的 r1 位（RTR 位之后，扩展帧格式中的第一个保留位），在 CANFD 帧格式中解码为 FDF 位。FDF 位为隐性（逻辑 1）表示 CANFD 帧，为显性（逻辑 0）表示经典 CAN 帧。在 CANFD 帧中，FDF 位之后的 res 和 BRS（BitRateSwitch，比特率切换）两个位，决定是否切换 CANFD 帧中的比特率，并通过 res 显性和 BRS 隐性指示已切换比特率。res 位隐性的编码保留，供将来协议扩展使用。如果 FDCAN 接收到的帧中 FDF 位和 res 位都是隐性，则会通过将 FDCAN_PSR.PXE 置位的方式发出协议异常事件。如果使能协议异常处理（FDCAN_CCCR.PXHD=0），FDCAN 将在下一采样点将工作状态从接收（FDCAN_PSR.ACT=0b10）变为同步（FDCAN_PSR.ACT=0b00）。如果禁止协议异常处理（FDCAN_CCCR.PXHD=1），FDCAN 会将隐性 res 位当作格式错误处理，并将以错误帧进行响应。

CAN FD 模式通过编程 FDCAN_CCCR.FDOE 来使能。如果 FDCAN_CCCR.FDOE=1，则会使能 CAN FD 帧的发送和接收。经典 CAN 帧始终可进行发送和接收。是发送 CAN FD 帧还是经典 CAN 帧，可通过相应发送缓冲区元素中的 FDF 位配置。如果 FDCAN_CCCR.FDOE=0，接收到的帧会被当作经典 CAN 帧，从而会在接收到 CAN FD 帧时发送错误帧。如果 CAN FD 模式已禁止，即使发送缓冲区元素的 FDF 位置 1，也不会发送 CAN FD 帧。仅当 FDCAN_CCCR.INIT 和 FDCAN_CCCR.CCE 均置位时，才能更改 FDCAN_CCCR.FDOE 和 FDCAN_CCCR.BRSE。

如果 FDCAN_CCCR.FDOE=0，则会忽略 FDF 和 BRS 位的设置，帧以经典型 CAN 格式发送。
如果FDCAN_CCCR.FDOE=1,且 FDCAN_CCCR.BRSE=0,则仅会评估发送缓冲区元素的 FDF 位。
如果 FDCAN_CCCR.FDOE=1,且 FDCAN_CCCR.BRSE=1,则会使能比特率切换发送 CAN FD 帧。所有 FDF 和 BRS 位置位的发送缓冲区元素均会以比特率切换的 CAN FD 格式发送。

建议仅当以下条件满足时，才在 CAN 操作期间切换模式：
– CAN FD 数据阶段的故障率显著高于 CAN FD 仲裁阶段的故障率。在这种情况下，应禁止发送的 CAN FD 比特率切换。
– 系统启动期间，所有节点在被验证能够以 CAN FD 格式进行通信之前，都将发送经典 CAN 消息。如果通过验证，所有节点会切换为 CAN FD 操作。
– CAN 局部网络（CAN Partial Networking）中的唤醒消息必须以经典 CAN 格式发送。
– EOL 编程（End-of-line programming）时，并非所有节点都支持 CANFD，非 CANFD 的节点会保持在静默模式直到编程结束。随后，所有节点切换回经典 CAN 通信。

在 CANFD 格式中，DLC 代码 0 到 8 的编码 (对应的数据字节长度) 与经典 CAN 相同，代码 9 到 15 的编码与经典 CAN 不同，如下表所示

表 34-2 FDCAN 中的 DLC 编码

DLC 编码		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数据字节数	经典 CAN	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8
	CAN FD	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

在 CAN FD 帧中，如果 BRS 位为隐性，则位时长将在 BRS（比特率切换）位后，在帧内切换。在 BRS 位之前，在 CAN FD 仲裁段，位时长按照寄存器 FDCAN_NBTP（Nominal Bit Timing & Prescaler Register，标称位时长和预分频寄存器）的定义执行。在接下来的 CAN FD 数据段，位时长按照寄存器 FDCAN_DBTP（Data Bit Timing & Prescaler Register，数据位时长和预分频寄存器）的定义执行。位时长在 CRC 分隔符之后，或检测到错误时从数据段位时长切换回来，以先发生的事件为准。

CAN FD 数据段的最大可配置比特率取决于 CAN 通信时钟频率。举例来说，如果 CAN 通信时钟频率为 20MHz，最短可配置的位时间为 4 个时间片（tq），则数据阶段的比特率为 5Mbps。

在带比特率切换和不带比特率切换的 CAN FD 两种数据帧格式中，位 ESI（Error Status Indicator，错误状态指示符）的值，由发送开始时发送节点的错误状态决定。如果发送节点为错误被动状态，则 ESI 以隐性位发送，否则以显性位发送。

34.2.1.4 收发器延迟补偿

在 CAN FD 传输的数据段，只有一个节点进行发送，其他所有节点都是接收节点。总线的长度没有影响。当通过引脚 FDCAN_TX 发送时，FDCAN 通过引脚 FDCAN_RX 从其本地 CAN 收发器接收发送的数据。接收数据因发送器延迟而延迟，如果该延迟大于 TSEG1（采样点之前的时间段），则会检测到位错误。为了使数据段位时间可以比发送器延迟更短，引入了延迟补偿。如果没有发送器延迟补偿，CAN FD 帧数据段的比特率将受到发送器延迟的限制。

FDCAN 的协议单元实现了延迟补偿机制来补偿发送器延迟，从而实现在 CAN FD 数据段以更高的比特率进行传输，而不受特定 CAN 收发器延迟的影响。

为了检查发送节点的数据段是否存在位错误，会将延迟的发送数据与第二采样点 SSP 处的接收数据进行比较。如果检测到位错误，发送节点将在下一个常规采样点对该位错误作出反应。在仲裁段，延迟补偿始终被禁用。

发送器延迟补偿支持数据位时间短于发射器延迟的配置，ISO 11898-1:2015 对此进行了详细描述，它通过置位 FDCAN_DBTP.TDC 使能。

在第二采样点 SSP 将接收到的位与发送的位进行比较。SSP 的位置定义为，从 FDCAN 发送输出引脚 FDCAN_TX 通过收发器到接收输入引脚 FDCAN_RX 测得的延迟，与由 FDCAN_TDCR.TDCO 配置的发送器延迟补偿偏移之和。发送器延迟补偿偏移，用于调整 SSP 在已接收到的位中的位置（例如数据段的位时间的一半）。第二采样点的位置向下取整到下一个整数 mtq（Minimum time quantum，即最小时间片，为一个 FDCAN 通信时钟周期）。

FDCAN_PSR.TDCV 显示实际的发送器延迟补偿值。当 FDCAN_CCCR.INIT 置位时，FDCAN_PSR.TDCV 被清零；当 FDCAN_DBTP.TDC 置位时，FDCAN_PSR.TDCV 在每次发送 FD 帧时都会更新。

对于在 FDCAN 中实现的发送器延迟补偿，必须考虑以下边界条件：

– 测得的 FDCAN_TX 到 FDCAN_RX 的延迟与配置的发送器延迟补偿偏移 FDCAN_TDCR.TDCO 之和，必须小于数据段的 6 个位时间。

– 测得的 FDCAN_TX 到 FDCAN_RX 的延迟与配置的发送器延迟补偿偏移 FDCAN_TDCR.TDCO 之和，必须小于或等于 127mtq。如果此和大于 127mtq，则发送器延迟补偿使用最大值 127mtq。

– 数据段在 CRC 分隔符的采样点结束，将停止在 SSP 检查接收位。

如果通过编程 FDCAN_DBTP.TDC=1 使能发送器延迟补偿，则会在每个 CAN FD 帧的 FDF 位到 res 位的下降沿处开始测量。当在发送节点的 FDCAN_RX 引脚上检测到该边沿时，停止测量。该测量的分辨率为一个 mtq。

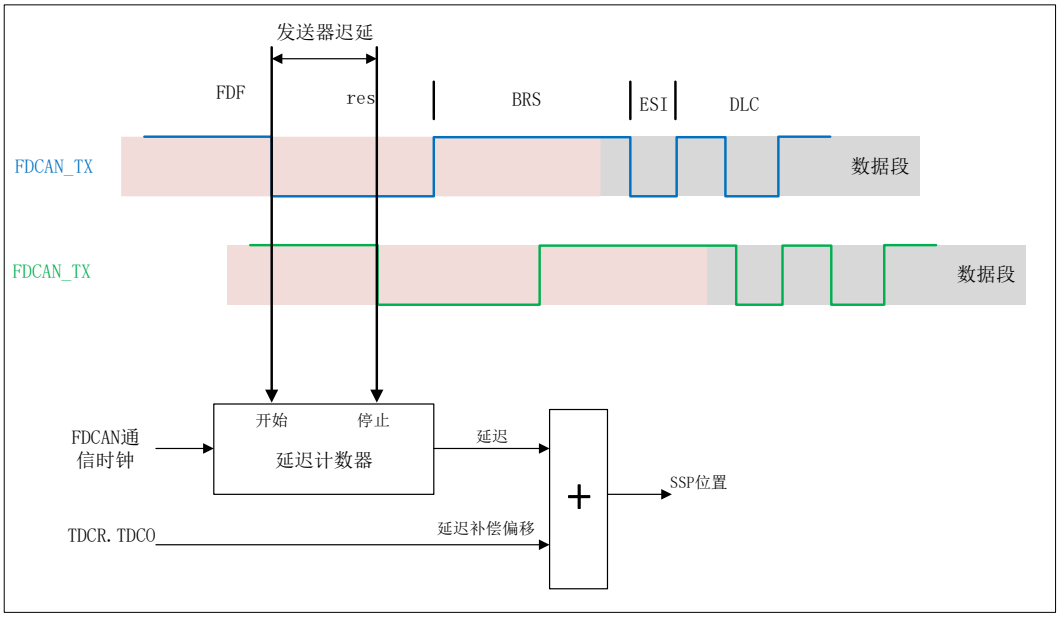


图 34-2 收发器延迟测量

为了避免接收到的 FDF 位中的显性毛刺，在接收到的 res 位的下降沿之前结束延迟补偿测量，从而导致 SSP 位置提前，可通过编程 FDCAN_TDCR.TDCF 来使能发送器延迟补偿滤波窗口，从而定义 SSP 位置的最小值。FDCAN_RX 引脚上的显性边缘会导致 SSP 位置更加提前，但对于发射器延迟测量而言这将被忽略。当 SSP 位置至少为 FDCAN_TDCR.TDCF 且 FDCAN_RX 引脚为低电平时，测量停止。

34.2.1.5 受限工作模式

在受限工作模式下，节点能够接收数据帧和远程帧，并能对有效帧进行确认，但不会发送数据帧、远程帧、主动错误帧或过载帧。在出现错误或过载情况时，不会发送显性位，而是等待出现总线空闲条件，以重新同步到 CAN 通信。错误记录（FDCAN_TDCR.CEL）动作时，错误计数器（FDCAN_TDCR.REC、FDCAN_TDCR.TEC）被冻结。主机可通过置位 FDCAN_CCCR.ASM 将 FDCAN 设置为受限工作模式，仅当 FDCAN_CCCR.CCE 和 FDCAN_CCCR.INIT 均设为 1 时，主机才能将 FDCAN_CCCR.ASM 置位。FDCAN_CCCR.ASM 可随时通过主机清零。

当发送处理单元无法及时从消息 RAM 中读取数据时，会自动进入受限工作模式。要退出受限工作模式，主机 CPU 必须清零 FDCAN_CCCR.ASM。

受限工作模式可用于适应不同 CAN 比特率的应用。在这种情况下，应用程序会测试不同的比特率，并在收到有效帧后退出受限操作模式。

注：受限工作模式不得与回环模式（内部或外部）结合使用。

34.2.1.6 总线监听模式

配置 FDCAN_CCCR.MON 为 1，将 FDCAN 设置为总线监听模式。在总线监听模式下（见 ISO 11898-1:2015,10.14 Bus monitoring），FDCAN 能够接收有效数据帧和有效远程帧，但不能启动发送。在该模式下，FDCAN 仅在 CAN 总线上发送隐性位，如果 FDCAN 需要发送一个显性位（ACK 位、过载标志、主动错误标志），该位将在内部被改道发送，以便 FDCAN 可以监听该显性位，但 CAN 总线可以保持隐性状态。在总线监听模式下，寄存器 FDCAN_TXBRP 保持复位状态。

总线监听模式可用于分析 CAN 总线上的流量，同时又不会因发送显性位而对总线造成影响。图 34-3 显示了总线监听模式下 FDCAN_TX 和 FDCAN_RX 信号与 FDCAN 的连接。

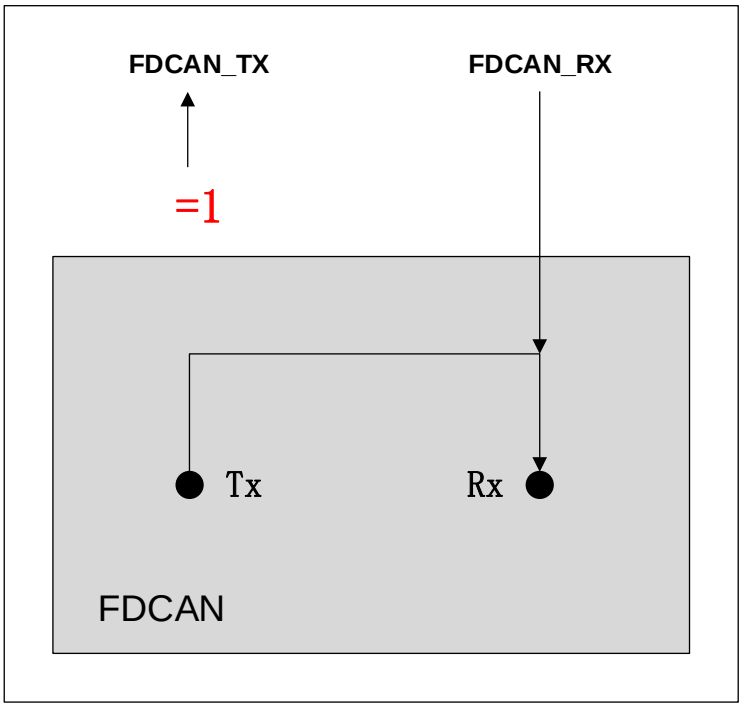


图 34-3 总线监听模式下的引脚控制

34.2.1.7 禁止自动重发（DAR）模式

根据 CAN 规范（见 ISO 11898-1:2015, 8.3.4 Recovery Management），FDCAN 可自动重发仲裁失败或在发送期间被错误干扰的帧。默认使能自动重发。

在 DAR 模式下，所有发送在 CAN 总线上启动后都会自动取消。发送缓冲区发送请求挂起位 FDCAN_TXBRP.TRPx 在成功发送后复位，如果取消时发送尚未开始、发送已因仲裁失败而中止、或帧发送期间发生错误，该位也会复位。

- 成功发送：
  - 对应的发送缓冲区发送已发生位 FDCAN_TXBTO.TOx 置位
  - 对应的发送缓冲区取消已完成位 FDCAN_TXBCF.CFx 不置位
- 即使已取消但也成功发送：
  - 对应的发送缓冲区发送已发生位 FDCAN_TXBTO.TOx 置位
  - 对应的发送缓冲区取消已完成位 FDCAN_TXBCF.CFx 置位
- 仲裁失败或帧发送受到干扰：
  - 对应的发送缓冲区发送已发生位 FDCAN_TXBTO.TOx 不置位

– 对应的发送缓冲区取消已完成位 FDCAN_TXBCF.CFx 置位

如果帧发送成功，并且使能了发送事件的存储，则会写入发送事件 FIFO 元素，其中事件类型 ET=0b10（即使已取消也会进行发送）。

34.2.1.8 掉电（休眠模式）

FDCAN 可通过 CC 控制寄存器 FDCAN_CCCR.CSR 设置为掉电模式。只要时钟停止请求有效，FDCAN_CCCR.CSR 位就读为 1。

当所有挂起的发送请求都完成后，FDCAN 会一直等待直到检测到总线空闲状态。检测到总线空闲状态后，FDCAN 将 FDCAN_CCCR.INIT 设置为 1 以防止任何进一步的 CAN 传输。然后，FDCAN 通过将 FDCAN_CCCR.CSA 置 1 确认其已准备好进入掉电状态。在模块时钟（控制逻辑时钟和通信时钟）关闭之前，可继续访问寄存器，但 FDCAN_CCCR.INIT 保持为 1。

注：

在 CAN 总线受到严重干扰的情况下，可能永远不会达到空闲状态，因此 FDCAN 不会置位 FDCAN_CCCR.INIT。这种情况可通过轮询 FDCAN_PSR.ACT 来检测。如果 FDCAN 没有进入空闲状态，软件可以写 CCCR.INIT=1，这将立即停止 FDCAN 的 CAN 通信，无论是否正在进行发送或接收。

要退出掉电模式，应用程序必须在复位 CC 控制寄存器标志 FDCAN_CCCR.CSR 之前开启模块时钟。FDCAN 将通过复位 FDCAN_CCCR.CSA 确认时钟已开启。之后，应用程序可通过复位 FDCAN_CCCR.INIT 重新开始 CAN 通信。

34.2.1.9 测试模式

要使能对 FDCAN 测试寄存器 FDCAN_TEST（请参见章节[FDCAN 测试寄存器（FDCAN_TEST）]）的写访问，必须将 FDCAN_CCCR.TEST 置 1，从而能够配置测试模式和测试功能。

通过编程 FDCAN_TEST.TX，发送引脚 FDCAN_TX 可实现四种输出功能。除了串行数据输出这一默认功能之外，还可驱动 CAN 采样点信号来监测 FDCAN 的位时长，并可输出恒定的显性或隐性电平。从 FDCAN_TEST.RX 可读取引脚 FDCAN_RX 的实际值。这两种功能（FDCAN_TEST.TX 和 FDCAN_TEST.RX）均可用于检查 CAN 总线的物理层。

由于 CAN 通信时钟与控制逻辑时钟之间的同步机制，从配置 FDCAN_TEST.TX 到引脚 FDCAN_TX 实现指定功能之间，可能有几个控制逻辑时钟周期的延迟。这一点还适用于通过 FDCAN_TEST.RX 读取引脚 FDCAN_RX 的情况。

注：

测试模式仅限用于生产测试和自检。对 FDCAN_TX 引脚进行软件控制会干扰所有 CAN 协议功能。不建议实际应用时使用测试模式。

34.2.1.10 外部回环模式

配置 FDCAN_TEST.LBCK 为 1，将 FDCAN 设置为外部回环模式。在回环模式下，FDCAN 将其自身发送的消息作为接收的消息来处理，并将消息存储（如果这些消息通过了接收过滤）到接收缓冲区或接收 FIFO 中。下图显示了外部回环模式下 FDCAN_TX 和 FDCAN_RX 引脚信号与 FDCAN 的连接。

该模式用于硬件自检。为了不受外部影响，FDCAN 在回环模式下忽略 ACK 错误（在数据帧或远程帧的 ACK 域采样到隐性位）。在此模式下，FDCAN_TX 输出的信号在内部直接反馈到接收处理单元，FDCAN_RX 输入引脚的实际值被忽略。从引脚 FDCAN_TX 可以监听到发送的消息。

34.2.1.11 内部回环模式

配置 FDCAN_TEST.LBCK 和 FDCAN_CCCR.MON 为 1，将 FDCAN 设置为内部环回模式。该模式可用于“热自检（Hot Selftest）”，也就是说，FDCAN 可以进行检测，同时又不会影响接入的正在运行的 CAN 系统。在此模式下，引脚 FDCAN_RX 与 FDCAN 断开连接，引脚 FDCAN_TX 保持隐性。下图显示了内部环回模式下 FDCAN_TX 和 FDCAN_RX 引脚信号与 FDCAN 的连接。

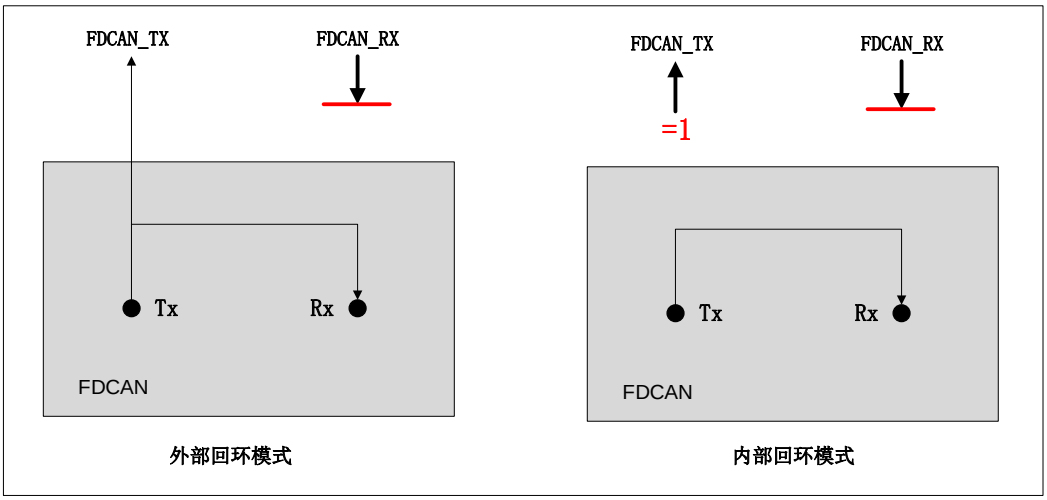


图 34-4 回环模式的引脚控制

34.2.2 时间戳生成

FDCAN 提供了一个 16 位循环计数器来生成内部时间戳。计数时间单位可通过预分频器 FDCAN_TSCC.TCP 配置为 CAN 位时间 1~16 倍。计数器值可通过 FDCAN_TSCV.TCV 读取。对寄存器 FDCAN_TSCV 进行写访问会将计数器复位为 0。时间戳计数器循环时，中断标志 FDCAN_IR.TSW 会置位。

在接收到或发送 SOF（Start Of Frame，帧起始）时，计数值被捕获并存储到接收缓冲区或接收 FIFO（RXTS[15:0]）或发送事件 FIFO（TXTS[15:0]）元素的时间戳部分。

34.2.3 超时计数器

FDCAN 提供了一个 16 位超时计数器来指示接收 FIF0、接收 FIF1 和发送事件 FIFO 的超时条件。它作为递减计数器工作，并且与时间戳计数器一样，使用 FDCAN_TSCC.TCP 控制的预分频器。超时计数器通过寄存器 FDCAN_TOCC 配置。实际计数器值可从 FDCAN_TOCV.TOC 读取。仅当 FDCAN_CCCR.INIT=0 时，才会启动超时计数器。当 FDCAN_CCCR.INIT=1 时，超时计数器停止计数，例如当 CAN 进入 Bus_Off 状态时。

工作模式通过 FDCAN_TOCC.TOS 选择。在连续模式下工作时，计数器在 FDCAN_CCCR.INIT 复位时启动。对 FDCAN_TOCV 进行写操作，会将计数器预设为 FDCAN_TOCC.TOP 配置的值，并继续递减计数。

当超时计数器由其中一个 FIFO 控制时，空 FIFO 会将计数器预设为 FDCAN_TOCC.TOP 配置的值。存储了第一个 FIFO 元素后，开始递减计数。对 FDCAN_TOCV 执行写入操作不起作用。

计数器达到 0 时，中断标志 FDCAN_IR.TOO 置位。在连续模式下，计数器在达到 FDCAN_TOCC.TOP 后立即重启。

注：

超时计数器的时钟信号来自 CAN 内核采样点信号。因此，由于 CAN 内核同步（重同步）机制的原因，超时计数器递减的时间点可能有所不同。如果在 CAN FD 中使用了比特率切换功能，则超时计数器在仲裁字段和数据字段中使用不同的时钟计数。

34.2.4 接收处理

接收处理单元控制着接收过滤、已接收消息到接收缓冲区或其中一个接收 FIFO（共两个）的传输以及接收 FIFO 放入和获取索引。

34.2.4.1 接收过滤

FDCAN 能够配置两组接收过滤器，一组供标准 ID 使用，另一组供扩展 ID 使用。这两组过滤器可分配给接收缓冲区、接收 FIFO0 或接收 FIFO1。对于接收过滤，每个过滤器列表从第 0 个元素开始执行比较，直到第一个匹配的元素。接收过滤在第一个匹配的元素处停止。不会为此消息评估剩下的过滤器元素。

- 主要特性有：
  - 每个过滤器元素可配置为
    - ID 范围过滤器（起止范围）
    - 一个或两个专用 ID 的过滤器
    - ID 位屏蔽的经典过滤器
  - 每个过滤器元素均可配置为实现接收过滤或拒绝过滤
  - 每个过滤器元素均可单独使能/禁止
  - 按顺序比较过滤器元素，到第一个匹配的过滤器元素后停止执行
- 相关配置寄存器位：
  - 全局过滤器配置（FDCAN_GFC）
  - 标准 ID 过滤器配置（FDCAN_SIDFC）
  - 扩展 ID 过滤器配置（FDCAN_XIDFC）
  - 扩展 ID 与运算掩码（FDCAN_XIDAM）
- 根据过滤器元素的配置（SFEC/EFEC），发生匹配时会触发下列操作之一：
  - 将接收到的帧存储到接收 FIFO0 或 FIFO1 中
  - 将接收到的帧存储到接收缓冲区中
  - 拒绝接收到的帧
  - 将高优先级消息中断标志 FDCAN_IR.HPM 置 1
  - 将高优先级消息中断标志 FDCAN_IR.HPM 置 1，并将接收到的帧存储到接收 FIF0 或 FIF1

中接收到完整的 ID 后，开始接收过滤。接收过滤完成后，如果找到匹配的接收缓冲区或接收 FIFO，消息处理单元便会开始将接收到的消息数据以 32 位的形式写入匹配的接收缓冲区或接收 FIFO 中。如果 CAN 协议控制器检测到错误条件（例如 CRC 错误），则会丢弃此消息，丢弃后的影响如下：

接收缓冲区

匹配接收缓冲区的新数据标志不会置 1，但接收缓冲区 (部分) 会被接收到的数据覆盖。有关错误类型，请参见 FDCAN_PSR.LEC 和 FDCAN_PSR.DLEC。

接收 FIFO

匹配接收 FIFO 的放入索引不会更新，但相关接收 FIFO 元素（部分）会被接收到的数据覆盖。有关错误类型，请参见 FDCAN_PSR.LEC 和 FDCAN_PSR.DLEC。如果匹配的接收 FIFO 在覆盖模式下工作，必须考虑接收 FIFO 覆盖模式中介绍的边界条件。

注：

当接收的消息写入两个接收 FIFO 之一或写入接收缓冲区中时，未匹配的已接收 ID 会独立于所用过滤器之外进行存储。接收过滤的过程很大程度上取决于所配置的过滤器元素的顺序。

34.2.4.2 ID 范围过滤器

该过滤器会将所有接收到的帧与 SF1ID/SF2ID 和 EF1ID/EF2ID 定义的范围内的消息 ID 进行匹配检查。范围过滤与扩展帧一起使用时，有两种可能的情况：

- EFT=0b00: 应用范围过滤器之前，会将已接收帧的消息 ID 与扩展 ID 与运算掩码 (FDCAN_XIDAM) 进行与运算
- EFT=0b11：不会为范围过滤使用扩展 ID 与运算掩码（FDCAN_XIDAM）

34.2.4.3 专用 ID 过滤器

可将过滤器元素配置为过滤一个或两个特定的消息 ID。要过滤一个特定的消息 ID，过滤器元素必须配置为 SF1ID=SF2ID 和 EF1ID=EF2ID。

34.2.4.4 ID 位屏蔽过滤器

ID 位屏蔽过滤用于通过屏蔽已接收消息 ID 的某些位来过滤消息 ID 组。进行 ID 位屏蔽过滤时，SF1ID/EF1ID 用作消息 ID 过滤器，SF2ID/EF2ID 用作过滤器 ID 掩码（ID Mask）。

ID 掩码中为 0 的位将屏蔽掉已配置 ID 过滤器的对应位，即，已接收消息 ID 在该位位置的值不与接收过滤器对应位的值进行比较，只有对应掩码位为 1 的已接收消息 ID 的位才进行比较。

如果所有掩码位均为 1，则仅当接收到的消息 ID 和消息 ID 过滤器完全相同时，才会出现匹配。如果所有掩码位均为 0，则所有消息 ID 都匹配。

34.2.4.5 标准消息 ID 过滤

图 34-5 介绍了标准消息 ID（11 位 ID）的过滤流程。章节【标准消息 ID 过滤器元素】介绍了标准消息 ID 过滤器元素。

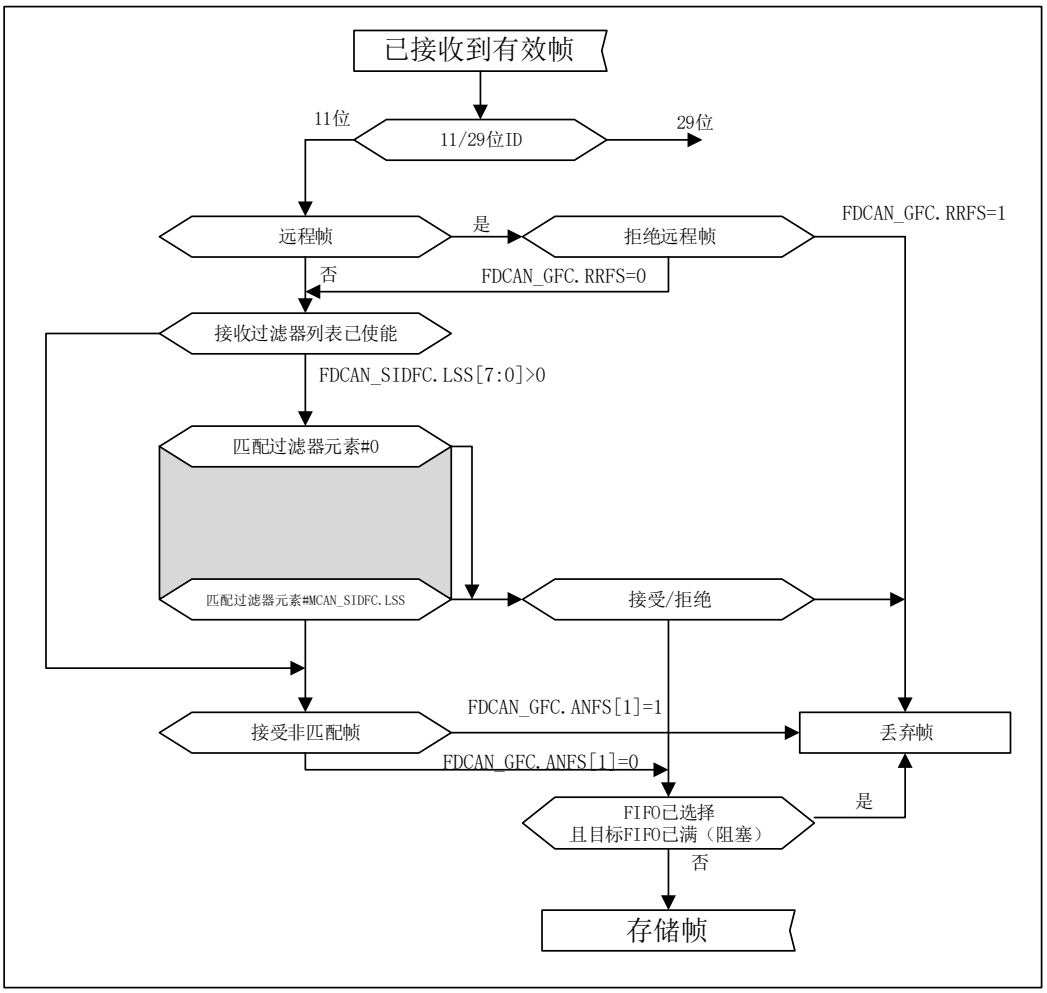


图 34-5 标准消息 ID 过滤器路径

在全局过滤器配置寄存器（FDCAN_GFC）和标准 ID 过滤器配置寄存器（FDCAN_SIDFC）控制下，接收消息的 ID、远程传输请求位（RTR）和 ID 扩展位（IDE）将与配置的过滤器元素列表进行比较。

34.2.4.6 扩展消息 ID 过滤

图 34-6 介绍了扩展消息 ID（29 位 ID）的过滤流程。章节【扩展消息 ID 过滤器元素】介绍了扩展消息 ID 过滤器元素。

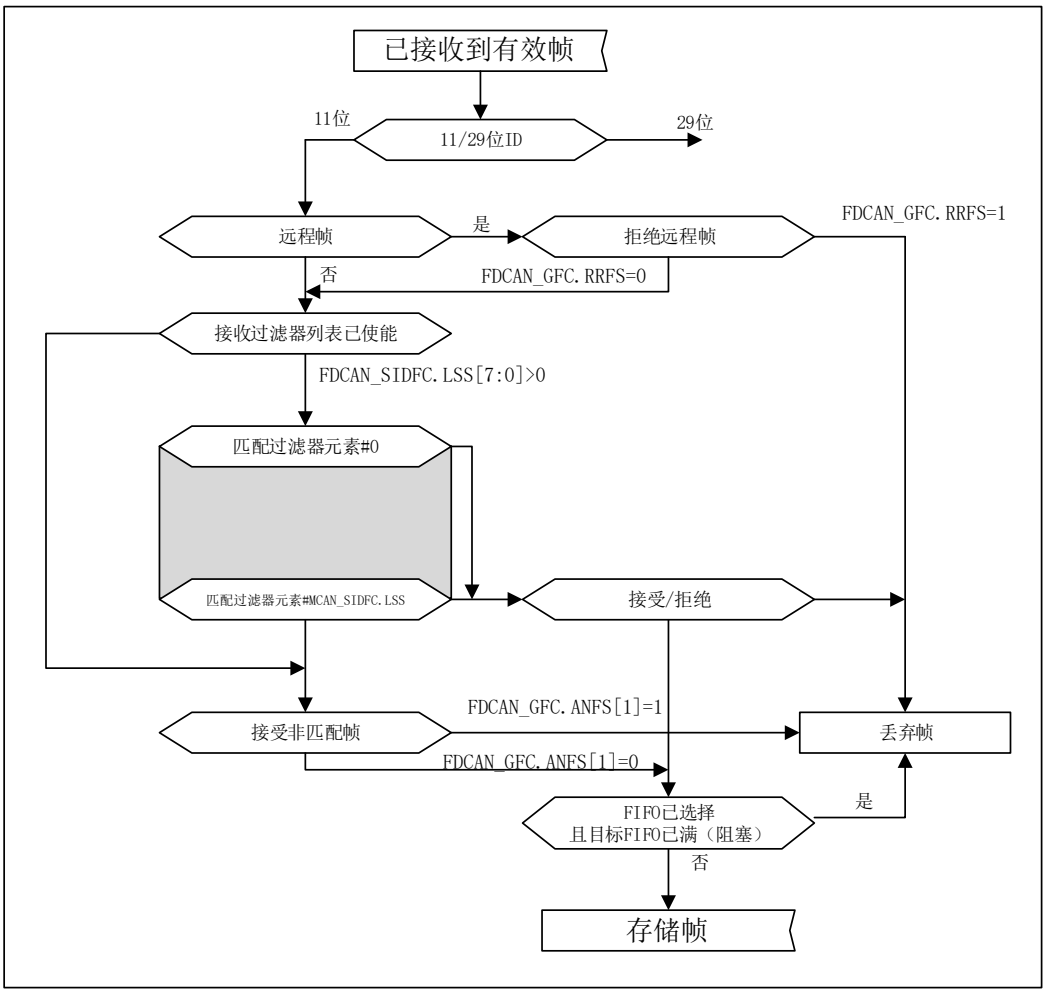


图 34-6 扩展消息 ID 过滤器路径

在全局过滤器配置寄存器（FDCAN_GFC）和扩展 ID 过滤器配置寄存器（FDCAN_XIDFC）控制下，接收消息的 ID、远程传输请求位（RTR）和 ID 扩展位（IDE）将与配置的过滤器元素列表进行比较。扩展 ID 与掩码（FDCAN_XIDAM）会在执行过滤器列表之前与接收到的 ID 进行与运算。

34.2.4.7 接收 FIFO

接收 FIFO0 和接收 FIFO1 分别最多可保存 64 个元素（实际元素个数与消息 RAM 容量有关）。两个接收 FIFO 的配置通过寄存器 FDCAN_RXF0C 和 FDCAN_RXF1C 完成。

通过接收过滤的消息会传输到匹配过滤器元素配置的接收 FIFO 中。有关可用于接收 FIFO0 和接收 FIFO1 的过滤器机制的说明，请参见章节【接收过滤】 。章节【接收缓冲区和 FIFO 元素】介绍了接收缓冲区和 FIFO 元素。

为了避免接收 FIFO 溢出，可使用接收 FIFO 水位线（Watermark）。当接收 FIFO 填充级别达到 FDCAN_RXFnC.FnWM 配置的接收 FIFO 水位线时，中断标志 FDCAN_IR.RFnW 置位。当接收 FIFO 的放入索引和获取索引相遇时，接收 FIFO 满的条件产生，标志位 FDCAN_RXFnS.FnF 置位，中断标志位 FDCAN_IR.RFnF 置位。

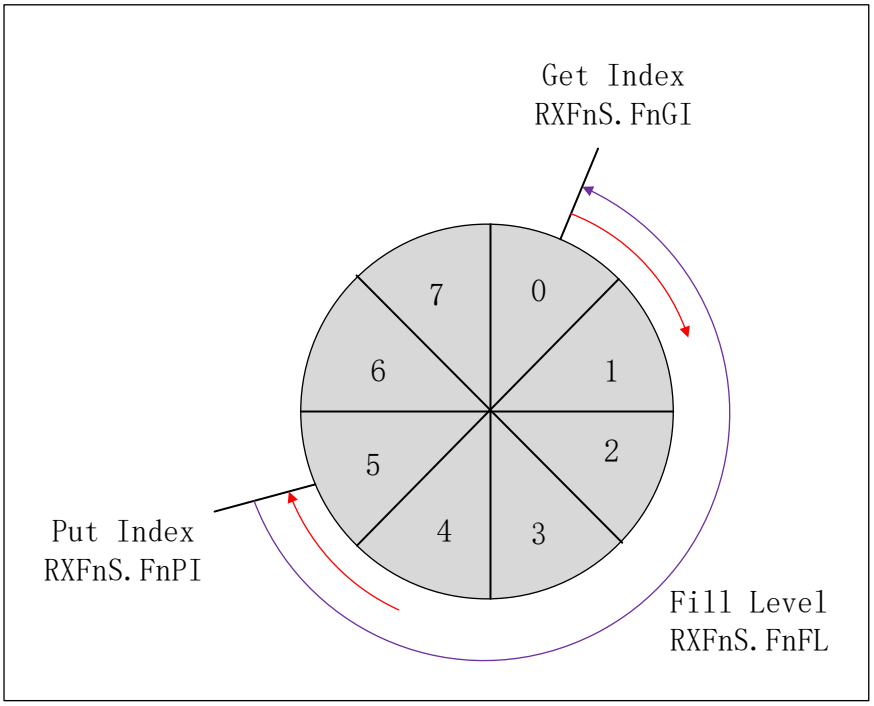


图 34-7 接收 FIFO 状态

当从接收 FIFO 中读取消息时，接收 FIFO 获取索引 FDCAN_RXFnS.FnGI 乘以 FIFO 元素大小后，必须再加上起始地址 FDCAN_RXFnC.FnSA。

表 34-3 接收缓冲器和 FIFO 元素大小

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	数据段（字节）	元素大小（字）
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

34.2.4.8 接收 FIFO 阻塞模式

接收 FIFO 阻塞模式是通过 FDCAN_RXFnC.FnOM=0 配置的，该模式是接收 FIFO 的默认工作模式。达到接收 FIFO 已满条件时(FDCAN_RXFnS.FnPI=FDCAN_RXFnS.FnGI)，标志位 RXFnS.FnF=1，中断标志 FDCAN_IR.RFnF=1。此时，FDCAN 不会继续向相应的接收 FIFO 写入消息，除非从相应的接收 FIFO 读取消息且其获取索引已递增。

如果在相应的接收 FIFO 已满时收到消息，此消息会被丢弃，并会通过 FDCAN_RXFnS.RFnL=1 指示消息丢失。此外，中断标志 FDCAN_IR.RFnL 也会置位。

34.2.4.9 接收 FIFO 覆盖模式

接收 FIFO 覆盖模式是通过 FDCAN_RXFnC.FnOM=1 配置的。

达到接收 FIFO 已满条件时(FDCAN_RXFnS.FnPI=FDCAN_RXFnS.FnGI)，标志位 RXFnS.FnF=1。此时，FIFO 接收的下一条消息将覆盖最早收到的消息，放入索引和获取索引都会加 1。

如果接收 FIFO 在覆盖模式下工作，并且接收 FIFO 已满，则应至少从获取索引加 1 处开始读取接收 FIFO 元素。这是因为，当 CPU 正在从消息 RAM 读取消息时，恰好有接收到的消息正在被写入消息 RAM，这可能导致 CPU 读到的数据与实际不一致，从接收 FIFO 读取数据时向获取索引中添加偏移量可避免此问题。偏移量取决于 CPU 访问接收 FIFO 的速度。图 34-8 显示了读取接收 FIFO 时获取索引的偏移量为 2。在这种情况下，存储在元素 1 和 2 中的两条消息会丢失。

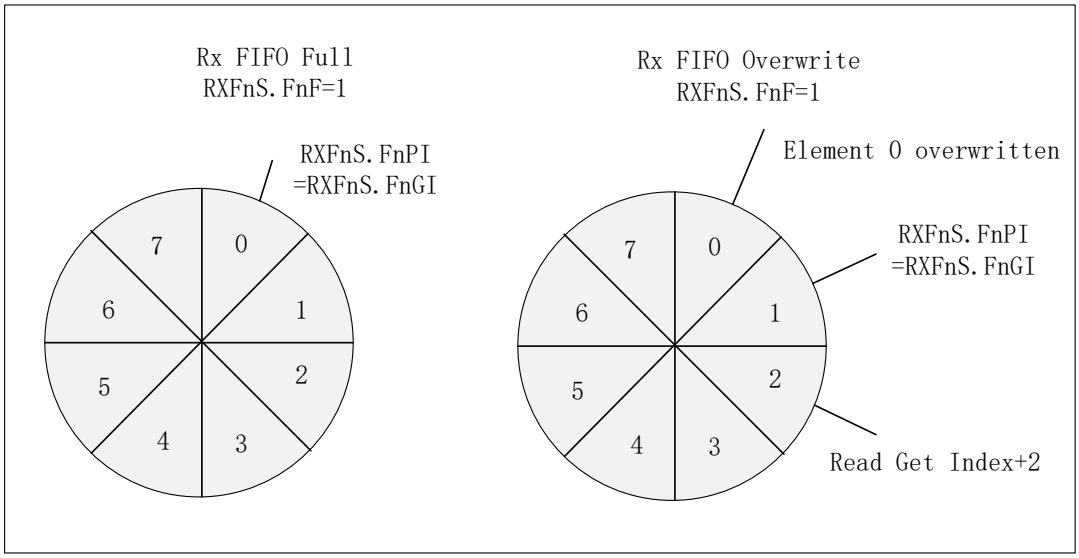


图 34-8 接收 FIFO 溢出处理

从接收 FIFO 读取数据后，必须将读取的最后一个元素的编号写入接收 FIFO 确认索引寄存器 FDCAN_RXFnA.FnA 中，使增加后的获取索引指向下一个元素。如果放入索引未增加到该接收 FIFO 元素编号，接收 FIFO 已满标志会复位（FDCAN_RXFnS.FnF=0）。

34.2.4.10 专用接收缓冲区

FDCAN 支持多达 64 个专用接收缓冲区。专用接收缓冲区的起始地址是通过 RXBC.RBSA 配置的。要使用专用接收缓冲区，对于标准 ID 消息，必须配置标准消息 ID 过滤器元素的 SFEC 位为 0b111、SFID2 位为 0b00（请参见章节【标准消息 ID 过滤器元素】）；对于扩展 ID 消息，必须配置扩展消息 ID 过滤器元素的 EFEC 位为 0b111、EFID2 位为 0b00(请参见章节【扩展消息 ID 过滤器元素】)。下表为专用接收缓冲区的过滤器配置示例。

接收到的消息被过滤器元素接受后，会存储在过滤器元素引用的消息 RAM 中的接收缓冲区中。存储格式与接收 FIFO 元素的格式相同。此外，中断寄存器中的标志 FDCAN_IR.DRX（专用接收缓冲区中存储的消息）也会置 1。

表 34-4 接收缓冲区过滤器配置示例

过滤器元素	**SFID1[10:0] EFID1[28:0]**	**SFID2[10:9] EFID2[10:9]**	**SFID2[5:0] EFID2[5:0]**
0	ID 消息 1	00	00 0000
1	ID 消息 2	00	00 0001
2	ID 消息 3	00	00 0010

匹配的接收到的消息最后一个字写入消息 RAM 后，寄存器 FDCAN_NDAT1/FDCAN_NDAT2 中对应的新数据标志会置位。只要新数据标志保持置位的状态，相应的接收缓冲区就会被锁定，以免更新为新的匹配消息。新数据标志必须由主机向相应的位写 1 进行复位。

接收缓冲区新数据标志置位时，其对应的消息 ID 过滤器元素将不进行接收过滤，其它的过滤器元素会继续进行接收过滤。其它的消息 ID 过滤器元素可能导致接收到的消息存储到另一接收缓冲区或接收 FIFO 中，或者被拒绝，具体取决于过滤器配置。

34.2.4.11 接收缓冲区处理

- 复位中断标志 FDCAN_IR.DRX
- 读取新数据寄存器
- 从消息 RAM 中读取消息
- 复位已处理消息的新数据标志

34.2.5 发送处理

发送处理单元处理专用发送缓冲区、发送 FIFO 和发送队列的发送请求。它控制着发送消息到 CAN 内核的传输、放入和获取索引以及发送事件 FIFO。最多可为消息发送设置 32 个发送缓冲区，每个发送缓冲区可以独立配置发送模式（经典 CAN 模式或 CAN FD 模式）。发送缓冲区的描述见章节【发送缓冲区元素】。下表描述了发送帧的可能配置。

表 34-5 发送帧的可能配置

FDCAN_CCCR		**发送缓冲区元素**		**发送帧**
BRSE	**FDOE**	**FDF**	**BRS**	
忽略	0	忽略	忽略	经典 CAN
0	1	0	忽略	经典 CAN
0	1	1	忽略	不切换比特率的 FD
1	1	0	忽略	经典 CAN
1	1	1	0	不切换比特率的 FD
1	1	1	1	切换比特率的 FD

注：

AUTOSAR 至少需要三个发送队列缓冲区，并支持发送取消。

当发送缓冲区请求挂起寄存器 FDCAN_TXBRP 更新时，发送处理单元会启动发送扫描来检查优先级最高的挂起的发送请求（消息 ID 最小的发送缓冲区）。FDCAN_TXBRP 更新的条件是，一个发送缓冲区发送完成，或主机通过写 FDCAN_TXBAR 添加发送请求，或主机通过写 FDCAN_TXBCR 取消已挂起的发送。

34.2.5.1 发送暂停

发送暂停功能用于 CAN 消息 ID（永久性地）被指定为特定值、并且无法轻易更改的 CAN 系统中。这些消息 ID 的 CAN 仲裁优先级可能高于其他已定义的消息，而在特定的应用中，其相对仲裁优先级应反转。这样可能会导致如下情况: 一个 ECU 突发多条较高仲裁优先级的 CAN 消息造成另一个仲裁优先级较低的 ECU CAN 消息被延迟。

举例来说，如果 CAN ECU-1 启用了此功能，并且其应用软件请求发送四条消息，成功发送第一条消息后，将等待两个总线空闲的 CAN 位时间，然后才允许开始发送下一条请求的消息。如果其他 ECU 有挂起的消息，则会在空闲时间开始发送这些消息，不需要与 ECU-1 的下一条消息进行仲裁。接收到消息后，ECU-1 可在接收到的消息释放 CAN 总线后立即开始进行下一次发送。

此功能是通过 FDCAN_CCCR.TXP 控制的。如果此位置位，则每次成功发送消息后，FDCAN 都将暂停两个 CAN 位时间，然后再开始进行下一次发送。这样，即使网络中其他节点的 ID 优先级较低，也能发送消息。默认禁止此功能（FDCAN_CCCR.TXP=0）。此功能可使来自单个节点突发的多条消息变得稀疏一些，避免发送较低优先级消息的 ECU 出现应用程序错误地认为自己请求过多传输。

34.2.5.2 专用发送缓冲区

专用发送缓冲区可完全在主机 CPU 的控制下发送消息。每个专用发送缓冲区都配置了特定的消息 ID。如果多个发送缓冲区配置为相同消息 ID，则会先发送编号最小的发送缓冲区中的消息，这些发送缓冲区应按升序请求发送，即先请求发送缓冲区编号最小的，或者，通过单次写入 FDCAN_TXBAR 同时请求发送。

如果缓冲区中的消息有更新，通过置位 FDCAN_TXBAR.ARn 请求发送。。已请求发送的消息，会根据消息 ID 先在内部与发送 FIFO 或发送队列中的消息进行仲裁，然后在外部与 CAN 总线上的消息进行仲裁，最后根据仲裁结果发送出去。

消息 RAM 中分配的专用发送缓冲区元素大小为若干个 32 位字（4 字节）。因此，消息 RAM 中专用发送缓冲区的起始地址，是通过将发送缓冲区索引（0~31）乘以元素大小（字），再乘以 4（转换为字节），然后与 FDCAN_TXBC.TBSA 相加计算得到。

表 34-6 发送缓冲器、FIFO（队列）元素大小

TXESC.TBDS2[2:0]	数据段（字节）	元素大小（字）
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

34.2.5.3 发送 FIFO

发送 FIFO 模式是通过将 FDCAN_TXBC.TFQM 配置为 0 实现的。存储在发送 FIFO 中的消息，是从获取索引 FDCAN_TXFQS.TFGI 指向的消息开始发送的。每次发送后，获取索引会循环递增，直至发送 FIFO 为空。发送 FIFO 会按消息写入的顺序，发送来自不同发送缓冲区但 ID 相同的消息。FDCAN 通过计算获取索引与放入索引之差，得到可用（空闲）的发送 FIFO 元素个数（TXFQS.TFFL）。新消息必须写入以放入索引 FDCAN_TXFQS.TFQPI 指向的发送缓冲区开始的发送 FIFO 中。添加发送请求会使放入索引递增，指向下一空闲的发送 FIFO 元素。当放入索引与获取索引相遇时，会指示发送 FIFO 已满（FDCAN_TXFQS.TFQF=1）。这种情况下，在下一条消息已发送且获取索引递增之前，不应继续向发送 FIFO 写入消息。

如果有单条消息添加到发送 FIFO，通过对发送 FIFO 放入索引对应的 FDCAN_TXBAR 位写 1 来请求发送。

如果有多条（假设有 n 条）消息添加到发送 FIFO，则会写入以放入索引开始的 n 个连续发送缓冲区中。随后通过 FDCAN_TXBAR 请求发送，放入索引循环递增 n。请求发送的发送缓冲区数不应超过发送 FIFO 空闲级别（FDCAN_TXFQS.TFFL）指示的空闲发送缓冲区数。

如果获取索引指向的发送缓冲区的发送请求已取消，那么获取索引会递增到下一个有发送请求挂起的发送缓冲区，并重新计算发送 FIFO 空闲级别（FDCAN_TXFQS.TFFL）。如果取消其他任何发送缓冲区的发送，获取索引和 FIFO 空闲级别保持不变。

消息 RAM 中分配的发送 FIFO 元素大小为若干个 32 位字（4 字节）。因此，下一个可用（空闲）的发送 FIFO 的起始地址，是通过将放入索引 FDCAN_TXFQS.TFQPI（0~31）乘以元素大小（字），再乘以 4，然后与 FDCAN_TXBC.TBSA 相加计算得到。

34.2.5.4 发送队列

发送队列模式是通过将 FDCAN_TXBC.TFQM 配置为 1 实现的。存储在发送队列中的消息，是先从消息 ID 最小（优先级最高）的开始发送的。如果多个发送队列配置为相同消息 ID，如果多个发送队列配置为相同消息 ID，由于发送顺序与存储消息的发送队列的编号有关，而这些发送队列的编号依赖于当前放入索引的状态，因此无法预测发送顺序。

新消息必须写入放入索引 FDCAN_TXFQS.TFQPI 指向的发送缓冲区中。放入索引一直指向发送队列中编号最小的空闲缓冲区。如果发送队列已满（FDCAN_TXFQS.TFQF=1），则放入索引无效，并且在至少有一个请求的消息已发出或挂起的发送请求已取消之前，不应继续向发送队列写入消息。

应用程序可使用寄存器 FDCAN_TXBRP 来代替放入索引，并可将消息放入任何没有挂起发送请求的发送缓冲区中。

消息 RAM 中分配的发送队列缓冲区元素大小为若干个 32 位字（4 字节）。因此，下一个可用（空闲）的发送队列缓冲区的起始地址，是通过将放入索引 FDCAN_TXFQS.TFQPI（0~31）乘以元素大小（字），再乘以 4，然后与 FDCAN_TXBC.TBSA 相加计算得到。

34.2.5.5 混合使用专用发送缓冲区和发送 FIFO

在这种情况下，消息 RAM 中的发送缓冲区会被划分为一组专用发送缓冲区和一个发送 FIFO。专用发送缓冲区的数量是通过 FDCAN_TXBC.NDTB 配置的。分配给发送 FIFO 的发送缓冲区数量是通过 FDCAN_TXBC.TFQS 配置的。如果 FDCAN_TXBC.TFQS 配置为 0，则仅会使用专用发送缓冲区。

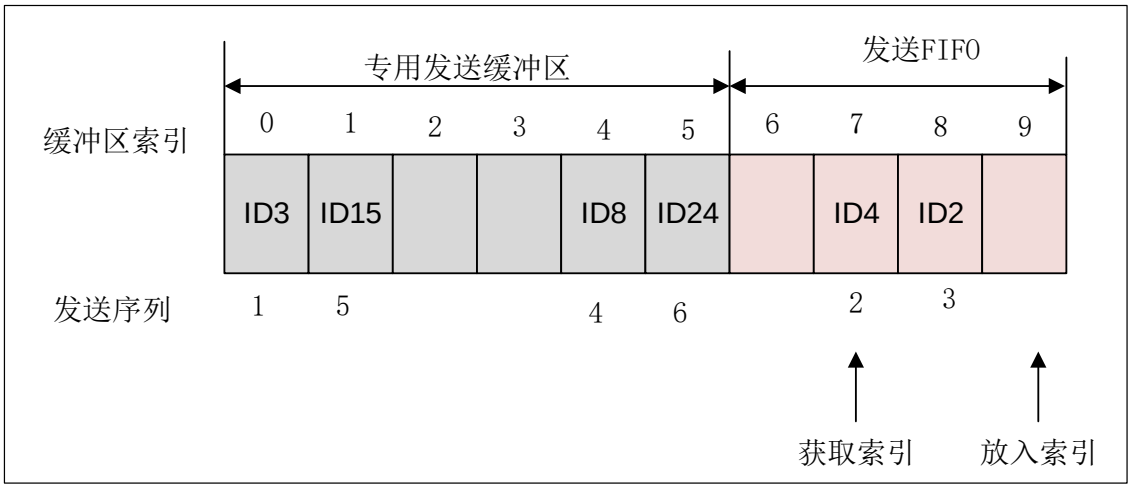


图 34-9 混合配置专用发送缓冲区和发送 FIFO 示例

发送优先级：

–扫描所有激活了发送请求的专用发送缓冲区和最早挂起发送请求的发送 FIFO 缓冲区（FDCAN_TXEFS.TFGI 指向的缓冲区）

–消息 ID 最小的缓冲区优先级最高，下次将发送该缓冲区的消息

34.2.5.6 混合使用专用发送缓冲区和发送队列

在这种情况下，消息 RAM 中的发送缓冲区会被划分为一组专用发送缓冲区和一个发送队列。专用发送缓冲区的数量是通过 FDCAN_TXBC.NDTB 配置的。发送队列缓冲区的数量是通过 FDCAN_TXBC.TFQS 配置的。如果 FDCAN_TXBC.TFQS 配置为 0，则仅会使用专用发送缓冲区。

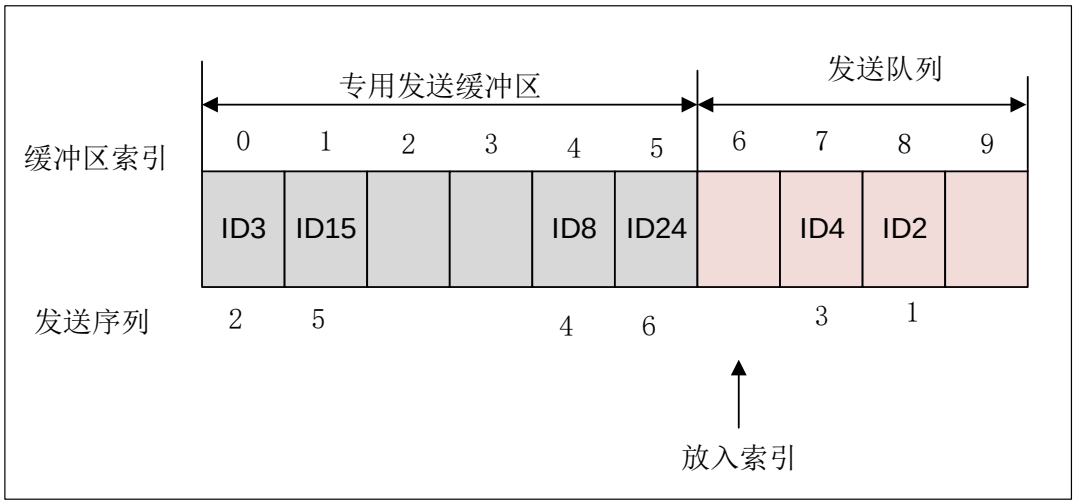


图 34-10 混合配置专用发送缓冲区和发送队列示例

发送优先级：

– 扫描所有激活了发送请求的专用发送缓冲区和发送队列缓冲区

– 消息 ID 最小的发送缓冲区优先级最高，下次将发送该缓冲区的消息

34.2.5.7 发送取消

FDCAN 支持发送取消功能。这个功能特别适用于网关应用和基于 AUTOSAR 的应用。要取消专用发送缓冲区或发送队列缓冲区的发送请求，主机必须向寄存器 FDCAN_TXBCR 中相应的位（发送缓冲区索引对应的位）写 1。发送取消不适用于发送 FIFO 模式。

成功取消后，寄存器 FDCAN_TXBCF 的相应位会置 1。

如果在缓冲区发送过程中请求取消发送，则只要发送正在进行中，FDCAN_TXBRP 相应的位就会保持置 1 状态。如果发送成功，FDCAN_TXBTO 和 FDCAN_TXBCF 相应的位会置 1。如果发送不成功，则不会重复发送，只会将 FDCAN_TXBCF 相应的位置位。

注：

如果一个挂起的发送在开始发送之前被立即取消，即使该节点中有另一消息挂起，取消后也会有一个短暂的窗口期，在此期间不会开始任何发送。这样，另一节点便可以发送优先级可能比该节点中第二条消息低的消息。

34.2.5.8 发送事件处理

FDCAN 用发送事件 FIFO 来支持发送事件的处理。FDCAN 在 CAN 总线上发送消息后，消息 ID 和时间戳会存储在发送事件 FIFO 元素中。为了将发送事件关联到发送事件 FIFO 元素，已发送的发送缓冲区中的消息标记会被复制到发送事件 FIFO 元素中。

发送事件 FIFO 最多可配置 32 个元素，章节【发送事件 FIFO 元素】描述了发送事件 FIFO 元素。

发送事件 FIFO 的用途是将处理发送状态信息与处理发送消息分开，也就是让发送缓冲区仅保存要发送的消息，而将发送状态单独存储在发送事件 FIFO 中。这样做是有优势的，尤其是在处理动态管理的发送队列时，发送缓冲区可在发送成功后立即用于新消息。重写发送缓冲区之前，不需要保存该发送缓冲区的发送状态信息。

当 FDCAN_IR.TEFF 指示发送事件 FIFO 已满时，在至少已读取一个元素且发送事件 FIFO 获取索引递增之前，不会继续向发送事件 FIFO 写入元素。如果在发送事件 FIFO 已满时发生发送事件，此事件会被丢弃，中断标志 FDCAN_IR.TEFL 置位。

为了避免发送事件 FIFO 溢出，可使用发送事件 FIFO 水位线。当发送事件 FIFO 填充数量达到由 TXEFC.EFWM 配置的发送事件 FIFO 水位线时，中断标志 FDCAN_IR.TEFW 置位。

从发送事件 FIFO 读取数据时，必须将发送事件 FIFO 获取索引 TXEFS.EFGI 乘以 2（发送事件 FIFO 元素大小为两个字），再乘以 4，然后与发送事件 FIFO 起始地址 TXEFC.EFSA 相加，以得到目标数据所在的地址。

34.2.6 FIFO 确认处理

接收 FIF0、接收 FIF1 和发送事件 FIFO 的获取索引，是通过对相应 FIFO 确认索引寄存器进行写操作来控制的，请参见章节【FDCAN 接收 FIFO0 确认寄存器（FDCAN_RXF0A）】、章节【FDCAN 接收 FIFO1 确认寄存器(FDCAN_RXF1A)】和章节【FDCAN 发送事件 FIFO 确认寄存器(FDCAN_TXEFA)】。对 FIFO 确认索引进行写操作会将 FIFO 获取索引设置为 FIFO 确认索引加 1，进而会更新 FIFO 填充级别。有两种用例：

1.如果只从 FIFO 读取了一个元素（获取索引指向的元素），则应将获取索引值写入 FIFO 确认索引寄存器中。

2.如果已从 FIFO 中读取了一个序列的元素，则仅需在该序列读取结束时，将最后一个元素的索引值写入 FIFO 确认索引寄存器，即可更新 FIFO 获取索引。

由于 CPU 可自由访问 FDCAN 的消息 RAM，在以任意顺序（不考虑获取索引）读取 FIFO 元素时需要特别注意，特别是在从两个接收 FIFO 之一读取高优先级消息时。在这种情况下，不应对 FIFO 确认索引寄存器进行写操作，否则会导致获取索引指向错误的位置，还会改变 FIFO 填充级别，一些较早的 FIFO 元素可能会丢失。

注：

应用程序必须确保写入到 FIFO 确认索引寄存器中的值是正确的，FDCAN 不会检查数值是否正确。

34.2.7 消息 RAM

34.2.7.1 消息 RAM 配置

消息 RAM 的宽度为 32 位。FDCAN 模块最多可在消息 RAM 中分配 212 个字，如下图所示。实际应用中不要求对所有部分都进行配置，各部分之间的顺序也没有限制。

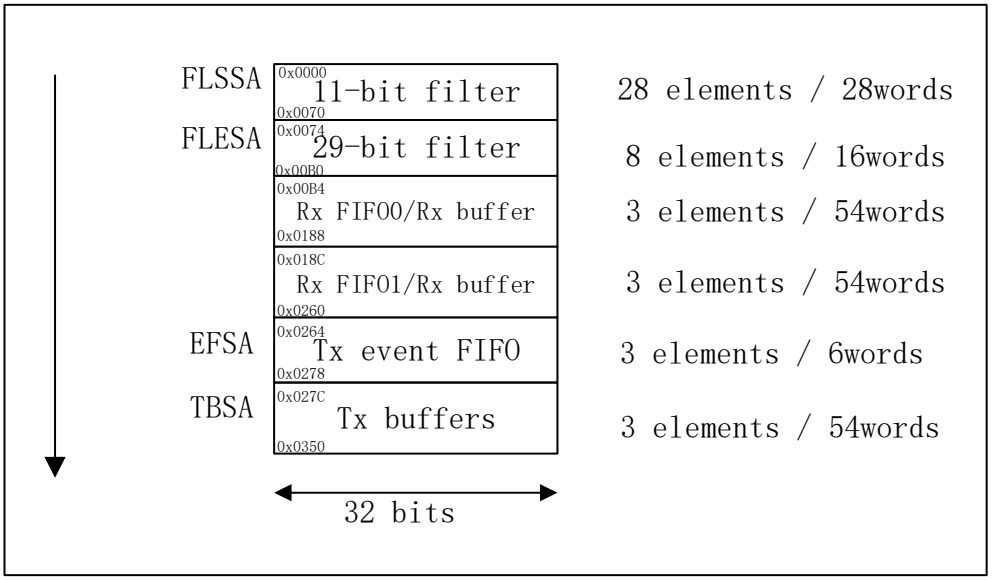


图 34-11 消息 RAM 配置

FDCAN 访问消息 RAM 时需要按 32 位对齐的地址进行寻址，而不是字节对齐的地址。可配置的起始地址是 32 位字地址，即仅评估地址的 15 到 2 位，忽略两个最低有效位。

注：

FDCAN 不会检查消息 RAM 的配置是否存在错误，因此配置不同部分的起始地址以及各部分元素的数量时必须多加留意，以免造成数据入侵或丢失。

34.2.7.2 接收缓冲区和 FIFO 元素

最多可在消息 RAM 中配置 2 个接收 FIFO/接收缓冲区。每个接收 FIFO/接收缓冲区均可配置为最多存储 3 个消息。接收缓冲区和接收 FIFO 元素的结构如下表所示，相关说明请见下表。可通过寄存器 FDCAN_RXESC 配置元素大小，CAN FD 消息的数据段最多可达 64 字节。

表 34-7 接收缓冲区和 FIFO 元素

位	31	24	23	16	15	8	7	0
R0	ESI	XTD RTR ID[28:0]						
R1	ANMF	FIDX[6:0]	Res FDF BRS DLC[3:0]		RXTS[15:0]			
R2	DB3[7:0]		DB2[7:0]		DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]		DB6[7:0]		DB5[7:0]		DB4[7:0]	
...	...		...		...		...	
Rn	DBm[7:0]		DBm-1[7:0]		DBm-2[7:0]		DBm-3[7:0]	

表 34-8 接收缓冲区和 FIFO 元素说明

位域	标记	说明
R0[31]	ESI	错误状态指示符（Error State Indicator） 0：发送节点为错误主动状态 1：发送节点为错误被动状态
R0[30]	XTD	扩展 ID 标志位（Extended Identifier） 向主机指示接收到的帧的 ID 是标准 ID 还是扩展 ID。 0：11 位标准 ID 1：29 位扩展 ID
R0[29]	RTR	远程发送请求（Remote Transmission Request）

		向主机指示接收到的是于数据帧还是远程帧。 0：接收到的帧是数据帧 1：接收到的帧是远程帧 注： CAN FD 没有远程帧。在 CAN FD 帧中（FDF=1），显性位 RRS（Remote Request Substitution）替代了 RTR 位。
R0[28:0]	ID[28:0]	ID（Identifier） 标准 ID 或扩展 ID 取决于 XTD 位，标准 ID 存储在 ID[28:18]中。
R1[31]	ANMF	接收非匹配帧（Accepted Non-matching Frame） 可通过 FDCAN_GFC.ANFS 和 FDCAN_GFC.ANFE 使能接收非匹配帧。 0：接收到的帧与过滤器索引 FIDX 指向的过滤器元素匹配 1：接收到的帧与任何接收过滤器元素都不匹配
R1[30:24]	FIDX[6:0]	过滤器索引（Filter Index） 0~127：匹配的接收过滤器元素的索引（ANMF =1 时无效） 范围为 0 到 FDCAN_SIDFC.LSS]减 1 或 FDCAN_XIDFC.LSE 减 1。
R1[21]	FDF	FD 格式（FD Format） 0：经典 CAN 帧格式 1：CAN FD 帧格式（新的 DLC 编码和 CRC）
R1[20]	BRS	比特率切换（Bit Rate Switch） 0：接收帧没有比特率切换 1：接收帧有比特率切换
R1[19:16]	DLC[3:0]	数据长度代码 （Data Length Code） 0~8：经典 CAN 和 CAN FD：接收到的帧包含 0~8 个数据字节 9~15：CAN：接收到的帧包含 8 个数据字节 9~15：CAN FD：接收到的帧包含 12/16/20/24/32/48/64 个数据字节
R1[15:0]	RXTS[15:0]	接收时间戳（Rx Timestamp） 接收到 SOF（Start Of Frame）时捕获的时间戳计数器值。 分辨率取决于时间戳计数器预分频器 FDCAN_TSCC.TCP 的配置。
R2[31:24]	DB3[7:0]	数据字节 3（Data Byte 3）
R2[23:16]	DB2[7:0]	数据字节 2（Data Byte 2）
R2[15:8]	DB1[7:0]	数据字节 3（Data Byte 1）
R2[7:0]	DB0[7:0]	数据字节 2（Data Byte 0）
R3[31:24]	DB7[7:0]	数据字节 7（Data Byte 7）
R3[23:16]	DB6[7:0]	数据字节 6（Data Byte 6）
R3[15:8]	DB5[7:0]	数据字节 5（Data Byte 5）
R3[7:0]	DB4[7:0]	数据字节 4（Data Byte 4）
...	...	...
Rn[31:24]	DBm[7:0]	数据字节 m（Data Byte m）
Rn[23:16]	DBm-1[7:0]	数据字节 m-1（Data Byte m-1）
Rn[15:8]	DBm-2[7:0]	数据字节 m-2（Data Byte m-2）
Rn[7:0]	DBm-3[7:0]	数据字节 m-3（Data Byte m-3）

34.2.7.3 发送缓冲区元素

发送缓冲区可配置为专用发送缓冲区和发送 FIFO（或队列）。如果发送缓冲区由专用发送缓冲区和发送 FIFO（或队列）共享，则专用发送缓冲区在发送缓冲区的起始处，其后是发送 FIFO（或队列）。发送处理单元会评估发送缓冲区配置 FDCAN_TXBC.TFQS 和 FDCAN_TXBC.NDTB，以区分专用发送缓冲区与发送 FIFO（队列）。发送缓冲区元素的结构如下表所示，相关说明请见下表。可通过寄存器 FDCAN_TXESC 配置元素大小，CAN FD 消息的数据段最多可达 64 字节。

表 34-9 发送缓冲区元素

位	31			24	23					16	15		8	7	0
T0	ESI	XTD	RTR		ID[28:0]										
T1	MM[7:0]				EFC	Res	FDF	BRS	DLC[3:0]		MM[15:8]			Res	
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]			DB0[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]			DB4[7:0]	
...	...				...						...			...	
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]			DBm-3[7:0]	

表 34-10 发送缓冲区元素说明

位域	标记	说明
T0[31]	ESI	错误状态指示符（Error State Indicator） 0：CAN FD 格式的 ESI 位仅取决于错误被动标志 1：CAN FD 格式的 ESI 位以隐性位发送 注： 发送缓冲区的 ESI 位会与错误被动标志进行或运算，以确定将发送的 FD 帧 ESI 位的值。 按照 CAN FD 协议规范的要求，错误主动节点可选择发送隐性 ESI 位，但错误被动节点将始终发送隐性 ESI 位。
T0[30]	XTD	扩展 ID 标志位（Extended Identifier） 0：11 位标准 ID 1：29 位扩展 ID
T0[29]	RTR	远程发送请求（Remote Transmission Request） 0：发送数据帧 1：发送远程帧 注： 当 RTR = 1 时，即使 FDCAN_CCCR.FDOE 使能以 CAN FD 格式进行发送，FDCAN 也会按照 ISO 11898-1:2015 的规定发送远程帧。
T0[28:0]	ID[28:0]	ID（Identifier） 标准 ID 或扩展 ID，取决于 XTD 位，标准 ID 必须写在 ID[28:18]中。
T1[31:24]	MM[7:0]	消息标记（Message Marker） 在发送缓冲区配置期间由 CPU 写入。复制到发送事件 FIFO 元素中，用于标识发送消息状态。
T1[23]	EFC	事件 FIFO 控制（Event FIFO Control） 0：不存储发送事件 1：存储发送事件

T1[21]	FDF	FD 格式（FD Format） 0：以经典 CAN 格式发送帧 1：以 CAN FD 格式发送帧
T1[20]	BRS	比特率切换（Bit Rate Switch） 0：CAN FD 帧发送时不切换比特率 1：CAN FD 帧发送时切换比特率 注： 仅当 CAN FD 使能（FDCAN_CCCR.FDOE=1）时，才会评估 ESI、FDF 和 BRS 位。仅当额外设置 FDCAN_CCCR.BRSE =1 时，才会评估 BRS 位。
T1[19:16]	DLC[3:0]	数据长度代码（Data Length Code） 0~8：经典 CAN 和 CAN FD：发送帧包含 0~8 个数据字节 9~15：CAN：发送帧包含 8 个数据字节 9~15：CAN FD：发送帧包含 12/16/20/24/32/48/64 个数据字节
T1[15:8]	MM[15:8]	宽消息标记的高字节（High byte of Wide Message Marker） 在发送缓冲区配置期间由 CPU 写入。复制到发送事件 FIFO 元素中，用于标识发送消息状态。仅当 FDCAN_CCCR.WMM=1 时有效。
T2[31:24]	DB3[7:0]	数据字节 3（Data Byte 3）
T2[23:16]	DB2[7:0]	数据字节 2（Data Byte 2）
T2[15:8]	DB1[7:0]	数据字节 3（Data Byte 1）
T2[7:0]	DB0[7:0]	数据字节 2（Data Byte 0）
T3[31:24]	DB7[7:0]	数据字节 7（Data Byte 7）
T3[23:16]	DB6[7:0]	数据字节 6（Data Byte 6）
T3[15:8]	DB5[7:0]	数据字节 5（Data Byte 5）
T3[7:0]	DB4[7:0]	数据字节 4（Data Byte 4）
...	...	...
Tn[31:24]	DBm[7:0]	数据字节 m（Data Byte m）
Tn[23:16]	DBm-1[7:0]	数据字节 m-1（Data Byte m-1）
Tn[15:8]	DBm-2[7:0]	数据字节 m-2（Data Byte m-2）
Tn[7:0]	DBm-3[7:0]	数据字节 m-3（Data Byte m-3）

34.2.7.4 发送事件 FIFO 元素

每个元素存储已发送消息相关的信息。通过读取发送事件 FIFO，主机 CPU 按消息的发送顺序获取这些信息。关于发送事件 FIFO 的状态信息，可从寄存器 FDCAN_TXEFS 中获取。发送事件 FIFO 元素的结构如下表所示，相关说明请见下表。

E1A: 当 FDCAN_CCCR.WMM=0 时，E1A.TXTS[15:0]保存由 FDCAN 的内部时间戳逻辑生成的 16 位时间戳。

E1B: 当使能 16 位宽消息标记（FDCAN_CCCR.WMM=1）时，E1B.MM[15:8]为宽消息标记的高 8 位。

表 34-11 发送事件 FIFO 元素

位	**31**		**24**	**23**		**16**	**15**	**8**	**7**	**0**
E0	ESI	XTD	RTR	ID[28:0]						
E1A	MM[7:0]			EF[1:0]	FDF	BRS	DLC[3:0]	TXTS[15:0]		

E1B	MM[7:0]	EF[1:0]	FDF	BRS	DLC[3:0]	MM[15:8]	Res

表 34-12 发送事件 FIFO 元素说明

位域	标记	说明
E0[31]	ESI	错误状态指示符（Error State Indicator） 0：发送节点为错误主动状态 1：发送节点为错误被动状态
E0[30]	XTD	扩展 ID 标志位（Extended Identifier） 0：11 位标准 ID 1：29 位扩展 ID
E0[29]	RTR	远程发送请求（Remote Transmission Request） 0：发送帧为数据帧 1：发送帧为远程帧
E0[28:0]	ID[28:0]	标识符（Identifier） 标准 ID 或扩展 ID，取决于 XTD 位。标准 ID 存储在 ID[28:18]中。
E1A/B[31:24]	MM[7:0]	消息标记（Message Marker） 从发送缓冲区复制到发送事件元素，用于标识发送消息状态。
E1A/B[23:22]	EF[1:0]	事件类型（Event Type） 00：保留 01：发送事件 10：取消后仍发送（在 DAR 模式下发送时始终为该值） 11：保留
E1A/B[21]	FDF	FD 格式（FD Format） 0：经典 CAN 帧格式 1：CAN FD 帧格式（新的 DLC 编码和 CRC）
E1A/B[20]	BRS	比特率切换（Bit Rate Switch） 0：CAN FD 帧发送时没有切换比特率 1：CAN FD 帧发送时有切换比特率
E1A/B[19:16]	DLC[3:0]	数据长度代码（Data Length Code） 0~8：经典 CAN 和 CAN FD：发送帧包含 0~8 个数据字节 9~15：经典 CAN：发送帧包含 8 个数据字节 9~15：CAN FD：发送帧包含 12/16/20/24/32/48/64 个数据字节
E1A[15:0]	TXTS[15:0]	发送时间戳（Tx Timestamp） 发送 SOF（Start Of Frame）时捕获的时间戳计数器值。分辨率取决于时间戳计数器预 分频器 FDCAN_TSCC.TCP 的配置。
E1B[15:8]	MM[15:8]	宽消息标记的高字节（High byte of Wide Message Marker） 在发送缓冲区配置期间由 CPU 写入。复制到发送事件 FIFO 元素中，用于标识发送消息状态。

34.2.7.5 标准消息 ID 过滤器元素

最多可为 11 位标准 ID 配置 128 个过滤器元素（每个元素大小为 1 个字）。其地址为过滤器元素索引（0~127）乘以 4（转换成字节），加上起始地址 FDCAN_SIDFC.FLSSA。标准消息 ID 元素的结构如下表所示，相关说明请见下表。

表 34-13 标准消息 ID 过滤器元素

位	31	24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]		Res	SFID2[10:0]		

表 34-14 标准消息 ID 过滤器元素说明

位域	标记	说明
S0[31:30]	SFT[1:0]	标准过滤器类型（Standard Filter Type） 00：从 SFID1 到 SFID2 的范围过滤器（SFID2 ≥ SFID1） 01：SFID1 或 SFID2 定义的双 ID 过滤器 10：经典过滤器：SFID1=ID，SFID2=掩码 11：禁止该过滤器元素 注：当 ***SFT=0b11*** 时，该过滤器元素被禁止，接收过滤继续进行（与 ***SFEC=0b000*** 时的行为相同）。
S0[29:27]	SFEC[2:0]	标准过滤器元素配置（Standard Filter Element Configuration） 所有使能的过滤器元素都用于标准帧的接收过滤。当匹配到一个已使能的过滤器元素， 或到达过滤器列表结尾处时，停止接收过滤。如果 SFEC=0b100、0b101 或 0b110， 出现匹配时中断标志 FDCAN_IR.HPM 置位，并产生中断（如果使能）。在这种情况下， 寄存器 FDCAN_HPMS 会更新为优先级匹配的状态。 000：禁止该过滤器元素 001：如果匹配，则存储到接收 FIFO0 中 010：如果匹配，则存储到接收 FIFO1 中 011：如果匹配，则拒绝 ID 100：如果匹配，则设置优先级，不存储 101：如果匹配，则设置优先级并存储到 FIFO0 中 110：如果匹配，则设置优先级并存储到 FIFO1 中 111：存储到接收缓冲区中或作为调试消息，忽略 SFT[1:0]的配置
S0[26:16]	SFID[10:0]	标准过滤器 ID1（Standard Filter ID 1） 标准 ID 过滤器元素的第一个 ID。当为接收缓冲区或调试消息过滤时，此位域定义要存储 的标准消息的 ID。接收的 ID 必须完全匹配，不使用掩码机制。
S0[10:0]	SFID2[10:0]	标准过滤器 ID2（Standard Filter ID 2） 该位域具有不同含义，具体视 SFEC 的配置而定： – SFEC=0b001~0b110 时：标准 ID 过滤器元素的第二个 ID

		– SFEC=0b111 时：用于接收缓冲区或调试消息的过滤器
	SFID2[10:9]	确定接收的消息存储到接收缓冲区中，还是作为调试消息序列中的消息 A、B 或 C 进行 处理。 00：将消息存储到接收缓冲区中 01：调试消息 A 10：调试消息 B 11：调试消息 C
	SFID2[5:0]	定义存储匹配消息的接收缓冲区到起始地址 RXBC.RBSA 的偏移（缓冲区索引）。

34.2.7.6 扩展消息 ID 过滤器元素

最多可为 29 位扩展 ID 配置 64 个过滤器元素（每个元素大小为 2 个字）。访问扩展消息 ID 过滤器元素时，其地址为过滤器元素索引（0~63）乘以 2，再乘以 4（转换成字节），然后加上起始地址 FDCAN_XIDFC.FLESA。扩展消息 ID 元素的结构如下表所示，相关说明请见下表。

表 34-15 扩展消息 ID 过滤器元素

位	**31**	**24**	**23**	**16**	**15**	**8**	**7**	**0**
F0	EFEC[2:0]		EFID1[28:0]					
F1	EFT[1:0]	Res	EFID2[28:0]					

表 34-16 扩展消息 ID 过滤器元素说明

位域	**标记**	**说明**
F0[31:29]	EFEC[2:0]	扩展过滤器元素配置 （Extended Filter Element Configuration） 所有使能的过滤器元素都用于扩展帧的接收过滤。当匹配到一个已使能的过滤器元素，或到达过滤器列表结尾处时，停止接收过滤。如果 EFEC=0b100、0b101 或 0b110，出现匹配时中断标志 FDCAN_IR.HPM 置位，并产生中断（若使能）。在这种情况下，寄存器 FDCAN_HPMS 会更新为优先级匹配的状态。 000：禁止该过滤器元素 001：如果匹配，则存储到接收 FIFO0 中 010：如果匹配，则存储到接收 FIFO1 中 011：如果匹配，则拒绝 ID 100：如果匹配，则设置优先级 101：如果匹配，则设置优先级并存储到 FIFO0 中 110：如果匹配，则设置优先级并存储到 FIFO1 中 111：存储到接收缓冲区中或作为调试消息，忽略 EFT[1:0]的配置
F0[28:0]	EFID[28:0]	扩展过滤器 ID1（Extended Filter ID 1） 扩展 ID 过滤器元素的第一个 ID。当为接收缓冲区或调试消息过滤时，此位域定义要存储的扩展消息的 ID。接收的 ID 必须完全匹配，仅使用 FDCAN_XIDAM 掩码机制。
F1[31:30]	EFT[1:0]	扩展过滤器类型（Extended Filter Type） 00：从 EFID1 到 EFID2 的范围过滤器(EFID2 ≥ EFID1) 01：EFID1 或 EFID2 定义的双 ID 过滤器 10：经典过滤器：EFID1=ID，EFID2=掩码

		11：从 EFID1 到 EFID2 的范围过滤器（EFID2 ≥ EFID1），不使用 FDCAN_XIDAM 掩码
F1[28:0]	EFID2[28:0]	扩展过滤器 ID2（Extended Filter ID 2） 该位域具有不同含义，具体视 EFEC 的配置而定： – SFEC =0b001~0b110 时：扩展 ID 过滤器元素的第二个 ID – SFEC =0b111 时：用于接收缓冲区或调试消息的过滤器
	SFID2[10:9]	确定接收的消息存储到接收缓冲区中，还是作为调试消息序列中的消息 A、B 或 C 进行处理。 00：将消息存储到接收缓冲区中 01：调试消息 A 10：调试消息 B 11：调试消息 C
	SFID2[5:0]	定义存储匹配消息的接收缓冲区到起始地址 RXBC.RBSA 的偏移(缓冲区索引)。

34.3 寄存器描述

FDCAN 寄存器支持字节写操作。

表 34-17 FDCAN 基地址

名称	基地址	描述
FDCAN1	0x4000CC00	FDCAN1 基地址
FDCAN2	0x4000E000	FDCAN2 基地址

34.3.1 FDCAN 字节序寄存器（FDCAN_ENDN），Address=0x04

地址偏移：0x004

复位值：0x87654321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	ETV	R	0x87	字节序测试值

34.3.2 FDCAN 数据位时间和预分频寄存器（FDCAN_DBTP），Address=0x0C

地址偏移：0x00C

复位值：0x00000A33

仅当 FDCAN_CCCR.CCE 和 FDCAN_CCCR.INIT 位均被置位时，才能对该寄存器进行写访问。CAN 位时间的可编程范围为 4 到 49 个时间片（Time quanta，tq）。CAN 时间片的可编程范围为 1 到 32 个 FDCAN 通信时钟周期（Minimum time quantum，mtq）。tq =（DBRP + 1)mtq。

DTSEG1 为 Prop_Seg 与 Phase_Seg1 之和。DTSEG2 为 Phase_Seg2。

因此，位时间的长度为（编程值）[DTSEG1 + DTSEG2 + 3] tq 或（功能值）[Sync_Seg + Prop_Seg +Phase_Seg1 + Phase_Seg2] tq。

信息处理时间（IPT）为零，这意味着下一位数据在采样点之后的第一个时钟边沿可用。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.								TDC	Res.		DBRP[4:0]				
-								RW	-		RW	RW	RW	RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.			DTSEG1[4:0]					DTSEG2[3:0]			DSJW[3:0]				
-			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	0	-
23	TDC	RW	0	发送器延迟补偿 0：禁止发送器延迟补偿 1：使能发送器延迟补偿
22:21	保留	-	0	-
20:16	DBRP[4:0]	RW	0	数据比特率预分频 生成位时间片的 FDCAN 通信时钟预分频值，位时间为该时间片的倍数。预分频有效值为 0 到 31，当 TDC=1 时，范围被限制在 0 到 1，硬件实际使用值为编 程值加 1。
15:13	保留	-	0	-
12:8	DTSEG1[4:0]	RW	0	采样点之前的数据时间 有效值为 0 到 31，硬件实际使用值为编程值加 1。
7:4	DTSEG2[3:0]	RW	0	采样点之后的数据时间 有效值为 1 到 15，硬件实际使用值为编程值加 1。
3:0	DSJW[3:0]	RW	0	数据（重新）同步跳转宽度 有效值为 0 到 15，硬件实际使用值为编程值加 1。

注：当 FDCAN 通信时钟为 8MHz 时，复位值 0x0000 0A33 会将 FDCAN 数据段的比特率配置为 500Kbps。通过 DBTP 配置的 CAN FD 数据段比特率不能小于通过 NBTP 配置的仲裁段比特率。

34.3.3 FDCAN 测试寄存器（FDCAN_TEST），Address=0x10

地址偏移：0x010

复位值：0x000000X0（位 7 的复位值由 FDCAN_RX 引脚的实际电平决定）

仅当 FDCAN_CCCR.TEST 位被置 1 后，才能对该测试寄存器进行写访问。当 FDCAN_CCCR.TEST 位被清零时，所有测试寄存器功能都会设为其复位值。

回环模式和软件控制 FDCAN_TX 引脚属于硬件测试模式。TX 配置为非 0b00 时可能会干扰 CAN 总线上的消息传输。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.										SVAL	TXBNS[4:0]				
-										R	R	R	R	R	R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.		PVAL	TXBNP[4:0]					RX	TX[1:0]		LBCK	Res.			
-		R	R	R	R	R	R	R	RW		RW	-			

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	0	-
21	SVAL	R	0	已开始有效

				0：TXBNS 的值无效 1：TXBNS 的值有效
20:16	TXBNS[4:0]	R	0	已开始发送的缓冲区编号 最后开始发送的发送缓冲区编号。 当 SVAL 位被置 1 时有效，有效范围为 0~31。
15:14	保留	-	0	-
13	PVAL	R	0	已准备有效 0：TXBNP 的值无效 1：TXBNP 的值有效
12:8	TXBNP[4:0]	R	0	已准备好发送的缓冲区编号 准备好发送的发送缓冲区编号。 当 PVAL 位被置 1 时有效，有效范围为 0~31。
7	RX	R	0	接收引脚 监测到的 FDCAN_RX 引脚的实际值 0：CAN 总线为显性（FDCAN_RX=0） 1：CAN 总线为隐性（FDCAN_RX=1）
6:5	TX[1:0]	RW	0	发送引脚控制 00：复位值，FDCAN_TX 引脚由 CAN 内核控制，在 CAN 位时间结束时更新 01：可以在 FDCAN_TX 引脚监测采样点 10：FDCAN_TX 引脚输出显性电平（0） 11：FDCAN_TX 引脚输出隐性电平（1）
4	LBCK	RW		回环模式 0：复位值，禁止回环模式 1：使能回环模式（详见【外部回环模式】 和【内部回环模式】章节）
3:0	保留	-	0	-

34.3.4 FDCAN RAM 看门狗寄存器（FDCAN_RWD），Address=0x14

地址偏移：0x014

复位值：0x00000000

RAM 看门狗会监视消息 RAM 的 READY 信号输出。对消息 RAM 进行访问时，会启动消息 RAM 看门狗计数器，该计数器的起始值由 FDCAN_RWD.WDC 位进行配置。

当消息 RAM 通过激活其 READY 信号输出，指示访问成功完成时，该计数器会重新载入 FDCAN_RWD.WDC 的值。如果计数器递减为 0 之前消息 RAM 没有响应，计数器将停止计数，并将中断标志 FDCAN_IR.WDI 位置 1。RAM 看门狗计数器由 FDCAN 控制逻辑时钟计数。

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	
15:8	WDV[7:0]	R	0	看门狗计数值 消息 RAM 看门狗计数器的实际值
7:0	WDC[7:0]	R	0	看门狗配置 消息 RAM 看门狗计数器的起始值。如果使用初始值 0x00，则计数器被禁止。

34.3.5 FDCAN CC 控制寄存器（FDCAN_CCCR），Address=0x18

地址偏移：0x018

复位值：0x00000001

寄存器各个位的设置条件请参见【软件初始化】章节。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	WMM	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
RW	RW	RW	RW	RW	-	RW	RW	RW	RW	RW	RW	R	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	-
15	NISO	RW	0	非 ISO 模式选择 0：使用 ISO 11898-1:2015 规定的 CAN FD 帧格式 1：使用 Bosch CAN FD 规范 V1.0 规定的 CAN FD 帧格式
14	TXP	RW	0	传输暂停 若该位被置为 1，则 FDCAN 在成功传输完一帧数据后会暂停两个位时间，再进行 下一帧数据的传输。 0：禁止传输暂停 1：使能传输暂停

13	EFBI	RW	0	总线同步期间的边沿滤波 0：禁止边沿滤波 1：需要两个连续的显性 tq 才能检测硬同步边沿
12	PXHD	RW	0	协议异常处理禁止 0：使能协议异常处理 1：禁止协议异常处理 注： 当协议异常处理被禁止时，FDCAN 将在检测到协议异常条件时传输错误帧。
11	WMM	RW	0	宽消息标记 当使用 16 位宽消息标记（WMM=1）时，发送事件 FIFO 中的 16 位内部时间戳是无效的。 0：使用 8 位消息标记 1：使用 16 位消息标记，替换发送事件 FIFO 中的 16 位时间戳
10	保留	-	0	-
9	BRSE	RW	0	比特率切换 0：禁止发送时进行比特率切换 1：使能发送时进行比特率切换 注： 当 CAN FD 模式无效（FDOE=0）时，BRSE 的设置不起作用。
8	FDOE	RW	0	FD 模式使能 0：禁止 FD 模式 1：使能 FD 模式
7	TEST	RW	0	测试模式使能 0：正常模式，测试寄存器 FDCAN_TEST 保持复位值 1：测试模式，使能写访问测试寄存器 FDCAN_TEST
6	DAR	RW	0	禁止自动重发 0：使能允许自动重发未成功发送的消息 1：禁止自动重发
5	MON	RW	0	总线监听模式 仅当 CCE 和 INIT 位均置 1 时，主机才能将 MON 位置 1。此位可随时通过主机复位。 0：禁止总线监听模式 1：使能总线监听模式
4	CSR	RW	0	时钟停止请求 当已请求时钟停止时，在所有挂起的传输请求均已完成且 CAN 总线达到空闲状 态之后，INIT 位和 CSA 位将会依次被置 1。 0：未请求时钟停止 1：已请求时钟停止
3	CSA	RW	0	时钟停止请求 当已请求时钟停止时，在所有挂起的传输请求均已完成且 CAN 总线达到空闲状 态之后，INIT 位和 CSA 位将会依次被置 1。 0：未请求时钟停止 1：已请求时钟停止

2	ASM	RW	0	传输暂停 若该位被置为 1，则 FDCAN 在成功传输完一帧数据后会暂停两个位时间，再进行 下一帧数据的传输。 0：禁止传输暂停 1：使能传输暂停
1	CCE	RW	0	配置更改使能 0：CPU 对受保护的配置寄存器无写访问权限 1：CPU 对受保护的配置寄存器有写访问权限（当 INIT=1 时）
0	INIT	RW	0	0：正常工作 1：启动初始化 注： 由于两个时钟域之间存在同步机制，INIT 的值在写入后可能会有一定延迟才能被正确读取。因此，在重新设定 INIT 值之前，请确认之前的设定值确实已被写入。

34.3.6 FDCAN 标称位时间和预分频寄存器（FDCAN_NBTP），Address=0x1C

地址偏移：0x01C

复位值：0x0600 0A03

仅当 FDCAN_CCCR.CCE 和 FDCAN_CCR.INIT 位均被置 1 时，才能对该寄存器进行写访问。CAN 标称位时间的可编程范围为 4 到 385 个时间片（Time quanta，tq）。CAN 时间片的可编程范围为 1 到 512 个 FDCAN 通信时钟周期（Minimum time quantum，mtq）。tq =（NBRP + 1)mtq。

NTSEG1 为 Prop_Seg 与 Phase_Seg1 之和。NTSEG2 为 Phase_Seg2。

因此，位时间的长度为（编程值）[NTSEG1 + NTSEG2 + 3] tq 或（功能值）[Sync_Seg + Prop_Seg +Phase_Seg1 + Phase_Seg2] tq。

信息处理时间（IPT）为零，这意味着下一位数据在采样点之后的第一个时钟边沿可用。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
RW							RW								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
RW								-	RW						

Bit	Name	R/W	Reset Value	Function
31:25	NSJW[6:0]	RW	0x6	标称（重）同步跳转宽度 有效值为 0 到 127，硬件实际使用值为编程值加 1。
24:16	NBRP[8:0]	RW	0	标称比特率预分频 生成位时间片的 FDCAN 通信时钟预分频值，位时间为该时间片的倍数。预分 频有效值为 0 到 511，硬件实际使用值为编程值加 1。
15:8	NTSEG1[7:0]	RW	0xA	采样点之前的标称时间段 有效值为 1 到 255，硬件实际使用值为编程值加 1。
7	保留	-	0	-
6:0	NTSEG2[6:0]	RW	0x3	采样点之后的标称时间段

				有效值为 1 到 127，硬件实际使用值为编程值加 1。

34.3.7 FDCAN 时间戳计数器配置寄存器（FDCAN_TSCC），Address=0x20

地址偏移：0x020

复位值：0x0000 0000

该寄存器用于配置内部 16 位时间戳计数器。内部时间戳处理的介绍请参见章节【时间戳生成】。

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

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	0	
19:16	TCP[3:0]	RW	0	时间戳计数器预分频 将时间戳和超时计数器时间单位配置为 CAN 位时间的 1~16 倍。 有效设定值为 0~15，硬件实际使用值为编程值加 1。
15:2	保留	-	0	
1:0	TSS[1:0]	R	0	时间戳选择 00：时间戳计数器的值一直为 0x0000 01：时间戳计数器的值根据 TCP 递增 10：禁止设定 11：时间戳计数器的值一直为 0x0000 （等同于设定值“00”）

34.3.8 FDCAN 时间戳计数器值寄存器（FDCAN_TSCV），Address=0x24

地址偏移：0x024

复位值：0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	-
15:0	TSC[15:0]	RW	0	时间戳计数器 在接收到或发送 SOF（Start Of Frame）时捕获的内部时间戳计数器值。 当 FDCAN_TSCC.TSS=0b01 时，时间戳计数器以 CAN 位时间的 1~16 倍进行递增，具体倍数由 FDCAN_TSCC.TCP 的设定值决定。 计数回卷时，中断标志 FDCAN_IR.TSW 会被置 1。写操作会将计数器清零。 注： 1.“回卷”是指时间戳计数器的值从非零变为零，并不是由于 FDCAN_TSCV 的写操作引起的清零。

				2. 字节访问时，对该寄存器的任何一个字节进行写操作都会导致时间戳寄存器被清零。

34.3.9 FDCAN 超时计数器配置寄存器（FDCAN_TOCC），Address=0x28

地址偏移：0x028

复位值：0xFFFF 0000

超时计数器的介绍请参见章节【超时计数器】章节。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[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
Res.													TOS[1:0]		ETOC
-													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	TOP[15:0]	RW	0	超时周期 超时计数器（递减计数器）的起始值，配置超时周期。
15:3	保留	-	0	-
2:1	TOS[1:0]	RW	0	超时选择 在连续模式下操作时，对 FDCAN_TOCV 进行写访问会将计数器预设为 FDCAN_TOCC.TOP 配置的值，并继续向下递减。当超时计数器由其中一个 FIFO 控制时，空 FIFO 会将计数器预设为 FDCAN_TOCC.TOP 配置的值。当第一个 FIFO 元素被存储后，计数器开始递减计数。 00：连续模式 01：由发送事件 FIFO 控制超时 10：由接收 FIFO0 控制超时 11：由接收 FIFO1 控制超时
0	ETOC	RW	0	超时计数器使能 0：禁止超时计数器 1：使能超时计数器

34.3.10 FDCAN 超时计数器值寄存器（FDCAN_TOCV），Address=0x2C

地址偏移：0x02C

复位值：0x0000 FFFF

超时计数器的介绍请参见章节【超时计数器】章节。

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	
15:0	TOC[15:0]	RW	0xFFFF	超时计数器

				超时计数器以 CAN 位时间的 1~16 倍进行递减，具体倍数由 FDCAN_TSCC.TCP 的设定值决定。当计数器递减到零时，中断标志位 FDCAN_IR.TOO 会被置 1，计数器停止计数。可通过 FDCAN_TOCC.TOS 来设置启动和复位（或重启）条件。 注： 字节访问时，当 FDCAN_TOCC.TOS=0b00 时，对该寄存器的任何一个字节进行写操作都会将超时计数器预设为 FDCAN_TOCC.TOP 配置的值。

34.3.11 FDCAN 错误计数器寄存器（FDCAN_ECR），Address=0x40

地址偏移：0x040

复位值：0x0000 0000

超时计数器的介绍请参见章节【超时计数器】章节。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.								CEL[7:0]							
-								R							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RP	REC[6:0]							TEC[7:0]							
R	R							R							

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	0	
23:16	CEL[7:0]	R	0	CAN 错误记录 当 CAN 协议错误导致发送错误计数器 TEC 或接收错误计数器 REC 递增时，该错误记录计数器 CEL 都会递增。计数器值达到 0xFF 时，会停止计数。只有 RP 位置位但 REC 没有变化时，CEL 不会递增。CEL 在 REC 和 TEC 增加后递增。 对 CEL 进行读访问会复位计数器。计数器值达到 0xFF 时，停止计数，TEC 或 REC 下一次增加时中断标志 FDCAN_IR.ELO 置位。 注： 字节访问时，读取寄存器的字节 2 会将 CEL 清零，读取字节 3、1 或 0 没有影响。
15	RP	R	0	接收错误被动 0：接收错误计数器低于错误被动级别 128 1：接收错误计数器已达到错误被动级别 128
14:8	REC[6:0]	R	0	接收错误计数器 接收错误计数器的实际状态，计数值范围为 0~127。
7:0	TEC[6:0]	R	0	发送错误计数器 发送错误计数器的实际状态，计数值范围为 0~255。

注: 如果 FDCAN_CCCR.ASM 被置为 1，则检测到 CAN 协议错误时，CAN 协议控制器不会使 TEC 和 REC 递增，但 CEL 仍会递增。

34.3.12 FDCAN 协议状态寄存器（FDCAN_PSR），Address=0x44

地址偏移：0x044

复位值：0x0000 0707

超时计数器的介绍请参见章节【超时计数器】章节。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.									TDCV[6:0]						
-									R						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	PXE	RFDF	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
-	R	R	R	R	R			R	R	R	R		R		

Bit	Name	R/W	Reset Value	Function
31:23	保留	-	0	-
22:16	TDCV[6:0]	R	0	发送器延迟补偿值 第二采样点（Secondary Sample Point，SSP）的位置，由 FDCAN_TX 引脚到 FDCAN_RX 引脚之间测得的迟延值与 FDCAN_TDCR.TDCO 之和决定。SSP 位置在数据段。TDCV 为已发送位的起点与第二采样点之间的 mtq 数，有效值为 0~127 个 mtq。 字节访问时，读取寄存器的字节 2 会将 CEL 清零，读取字节 3、1 或 0 没有影响。
15	保留	-	0	-
14	PXE	R	0	协议异常事件 0：自上次读访问后未发生协议异常事件 1：已发生协议异常事件 访问类型为 RX：进行读访问时复位。 注： 字节访问时，读取寄存器的字节 0 会将 PXE 清零，读取字节 3/2/1 没有影响。
13	RFDF	R	0	收到 CAN FD 消息 此位的置位与接收过滤无关。 0：自从上次被 CPU 清零后未收到 CAN FD 格式的消息 1：收到 FDF 标志为 1 的 CAN FD 格式的消息 注： 字节访问时，读取寄存器的字节 0 会将 RFDF 清零，读取字节 3/2/1 没有影响。
12	RBRS	R	0	上次收到的 CAN FD 消息的 BRS 标志 该位与 RFDF 一起置位，与接收过滤无关。 0：上次收到的 CAN FD 消息的 BRS 标志未置位 1：上次收到的 CAN FD 消息的 BRS 标志已置位 注： 字节访问时，读取寄存器的字节 0 会将 RBRS 清零，读取字节 3/2/1 没有影响。
11	RESI	R	0	上次收到的 CAN FD 消息的 ESI 标志 该位与 RFDF 一起被设置，与验收过滤器无关。 0：上次收到的 CAN FD 消息的 ESI 标志未置位 1：上次收到的 CAN FD 消息的 ESI 标志已置位 注： 字节访问时，读取寄存器的字节 0 会将 RESI 清零，读取字节 3/2/1 没有影响。
10:8	DLEC[2:0]	R	0	数据段最后一次错误码

				在设置了 BRS 标志的 CAN FD 帧的数据段中发生的最后一个错误的类型。错误码与 LEC 相同。当设置了 BRS 标志的 CAN FD 帧传输（接收或发送）没有错误时，DLEC 将被清零。 注： 字节访问时，读取寄存器的字节 0 会将 DLEC 设置为 0b111，读取字节 3/2/1 没有影响。
7	BO	R	0	Buss_Off 状态 0：FDCAN 未处于 Bus_Off 状态 1：FDCAN 处于 Bus_Off 状态
6	EW	R	0	警告状态 0：两个错误计数器均小于错误警告（Error_Warning）上限 96 1：至少有一个错误计数器已达到错误警告（Error_Warning）上限 96
5	EP	R	0	错误被动 0：FDCAN 处于错误主动（Error_Active）状态。FDCAN 正常参与总线通信，并在检测到错误时发送主动错误标志。 1：FDCAN 处于错误被动（Error_Passive）状态
4:3	ACT[1:0]	R	0	通信状态 监测 FDCAN 的通信状态。 00：同步中，节点正在进行 CAN 通信同步 01：空闲，节点既不是接收器也不是发送器 10：接收器，节点作为接收器工作 11：发送器，节点作为发送器工作 注： 协议异常事件会将 ACT 设为 0b00
2:0	LEC[2:0]	R	0	最后一次错误码 0：没有错误（No Error）。自成功接收或发送使 LEC 复位之后，没有错误发生。 1：填充错误（Stuff Error）。在已接收到的消息中，某一部分出现了 5 个以上的相同 位，这种情况是不允许的。 2：格式错误（Form Error）。已接收帧的固定格式部分有格式错误。 3：ACK 错误（ACK Error）。FDCAN 发送的消息没有被其他节点确认。 4：Bit1 错误（Bit1 Error）。在消息的发送过程中（仲裁段除外），设备想发送一个隐 性电平（逻辑值 1），但是监测到的总线电平为显性（逻辑值 0）。 5：Bit0 错误（Bit0 Error）。在消息的发送过程中（包括 ACK 位、主动错误标志、过载 标志），设备想发送一个显性电平（数据或 ID 的逻辑值 0），但是监测到的总线电平为 隐性（逻辑值 1）。在总线 Bus_Off 恢复期间，每次监测到 11 位隐性位序列时，都会设 置为该状态。这样，CPU 就能够监测总线恢复序列的进程（表明总线未卡在显性状态 或持续受到干扰）。

				6：CRC 错误（CRC Error）。已接收消息的 CRC 有错误。收到的 CRC 和用接收到的数据 计算出的 CRC 不匹配。 7：无变化（No Change）。对协议状态寄存器的任何读访问会将 LEC 重新初始化为 7。 当 LEC 显示值为 7 时，表示自上次 CPU 对协议状态寄存器进行读访问之后，没有检测到 CAN 总线事件。 注： 1.当设置了 BRS 标志的 CAN FD 帧已经到达数据段时，下一个 CAN 事件（错误或有效帧）将显示在 DLEC 中而不是 LEC 中。CAN FD CRC 序列中的固定填充位的错误将显示为格式错误（Form Error）而不是填充错误（Stuff Error）。 2.Bus_Off 恢复序列（请参考 ISO11898-1:2015）不能通过置位或清零 FDCAN_CCCR.INIT 来缩短。如果设备进入 Bus_Off 状态，它将自动置位 FDCAN_CCCR.INIT，停止所有总线活动。一旦 FDCAN_CCCR.INIT 被 CPU 清零，设备在恢复正常操作之前，将等待 129 个总线空闲（129*11 个连续的隐性位）。在 Bus_Off 恢复序列结束时，错误管理计数器将被复位。在清零 FDCAN_CCCR.INIT 后的等待期间，每次监测到一个 11 位隐性序列时，就向 FDCAN_PSR.LEC 写入一个 Bit0 错误码，使 CPU 能够随时检查 CAN 总线是否处于显性状态或持续被干扰，并监测 Bus_Off 恢复序列。FDCAN_ECR.REC 用于对这些序列进行计数。 3.字节操作时，读取寄存器的字节 0 会将 LEC 设置为 0b111，读取字节 3/2/1 没有影响。

34.3.13 FDCAN 发送器延迟补偿寄存器（FDCAN_TDCR），Address=0x48

地址偏移：0x048

复位值：0x0000 0000

超时计数器的介绍请参见章节【超时计数器】章节。

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

Bit	Name	R/W	Reset Value	Function
31:15	保留	-	0	-
14:8	TDCO[6:0]	RW	0	发送器延迟补偿 SSP 偏移 该偏移值是指，从 FDCAN_TX 引脚到 FDCAN_RX 引脚测得的延迟与第二采样点（SSP）之间的距离。有效设定值为 0~127mtq。
7	保留	-	0	-
6:0	TDCF[6:0]	RW	0	发送器延迟补偿滤波窗口宽度 定义 SSP 位置的最小值，对于发送器延迟的测量，将忽略 FDCAN_RX 引脚上会导致 SSP 位置提前的显性边沿。当 TDCF 设定值大于 TDCO 时，该功能被启用。有效设定值为 0~127mtq。

34.3.14 FDCAN 中断寄存器（FDCAN_IR），Address=0x50

地址偏移：0x050

复位值：0x0000 0000

当如下列表中的一个条件被检测到时，对应标志位会置位。在主机将标志位清零之前，保持置位状态。所有标志位写 1 清零，写 0 无影响。硬复位会将寄存器清零。是否产生中断，由寄存器 FDCAN_IE 控制；哪条中断线产生中断，由寄存器 FDCAN_ILS 控制。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.		DRX	TOO	MRAF	TSW
-		RW								-		RW			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
RW															

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	0	-
29	ARA	RW	0	访问保留地址（Access to Reserved Address) 0：未对保留地址进行访问 1：已对保留地址进行访问
28	PED	RW	0	数据段协议错误（使用数据位时间）(Protocol Error in Data Phase（Data Bit Time isused)) 0：未检测到数据段协议错误 1：检测到数据段协议错误（FDCAN_PSR.DLEC≠0/7）
27	PEA	RW	0	仲裁段协议错误（使用标称位时间）（Protocol Error in Arbitration Phase（Nominal Bit Time is used)) 0：未检测到仲裁段协议错误 1：检测到仲裁段协议错误（FDCAN_PSR.LEC≠0/7）
26	WDI	RW	0	看门狗中断（Watchdog Interrupt) 0：未发生消息 RAM 看门狗事件 1：因 READY 信号缺失而发生消息 RAM 看门狗事件
25	BO	RW	0	Bus_Off 状态（Bus_Off Status) 0：Bus_Off 状态未改变 1：Bus_Off 状态已改变
24	EW	RW	0	警告状态（Warning Status) 0：错误警告（Error_Warning）状态未改变 1：错误警告（Error_Warning）状态已改变
23	EP	RW	0	错误被动（Error Passive) 0：错误被动（Error_Passive）状态未改变 1：错误被动（Error_Passive）状态已改变
22	ELO	RW	0	错误记录溢出（Error Logging Overflow) 0：CAN 错误记录计数器未溢出 1：CAN 错误记录计数器已溢出
21:20	保留	-	0	-
19	DRX	RW	0	消息存储到专用接收缓冲区（Message stored to Dedicated Rx Buffer) 每当收到的消息已存储到专用接收缓冲区时，都会置位该标志。 0：接收缓存区未更新 1：至少有一条消息已存储到接收缓冲区

18	TOO	RW	0	发生超时（Timeout Occurred) 0：未发生超时 1：已发生超时
17	MRAF	RW	0	消息 RAM 访问失败（Message RAM Access Failure) 接收处理中置位条件： 1）在接收到后续消息的仲裁段之前，尚未完成对已接收消息的接收过滤或存储。在这种情况下，接收过滤或消息存储被终止，接收处理单元将开始处理后续消息。 2）无法向消息 RAM 写入消息，消息存储将终止。 在这两种情况下，FIFO 放入索引都不会更新，专用接收缓冲区的新数据标志不会置位，当下一条消息存储到该位置时，已部分存储的消息会被覆盖。 发送处理中置位条件： 1）发送处理单元无法及时从消息 RAM 读取消息。在这种情况下，消息发送将终止。 在发送处理单元访问失败的情况下，FDCAN 被切换到受限工作模式（详见章节【受限工作模式】）。要退出受限工作模式，主机 CPU 必须复位 FDCAN_CCCR.ASM。 0：未发生消息 RAM 访问失败 1：发生了消息 RAM 访问失败
16	TSW	RW	0	时间戳回卷（Timestamp Wraparound) 0：时间戳计数器未回卷 1：时间戳计数器已回卷
15	TEFL	RW	0	发送事件 FIFO 元素丢失（Tx Event FIFO Element Lost) 0：发送事件 FIFO 元素未丢失 1：发送事件 FIFO 元素有丢失；对空间大小为 0 的发送事件 FIFO 进行写操作时，该位也会置位
14	TEFF	RW	0	发送事件 FIFO 已满（Tx Event FIFO Full) 0：发送事件 FIFO 未满 1：发送事件 FIFO 已满
13	TEFW	RW	0	发送事件 FIFO 到达水位线（Tx Event FIFO Watermark Reached) 0：发送事件 FIFO 的填充级别低于水位线 1：发送事件 FIFO 的填充级别达到水位线
12	TEFN	RW	0	发送事件 FIFO 新条目（Tx Event FIFO New Entry) 0：发送事件 FIFO 无变化 1：发送处理单元写入了发送事件 FIFO 元素
11	TFE	RW	0	发送 FIFO 为空（Tx FIFO Empty) 0：发送 FIFO 非空 1：发送 FIFO 为空
10	TCF	RW	0	发送取消已完成（**Transmission Cancellation Finished)** **0**：发送取消未完成 **1**：发送取消已完成
9	TC	RW	0	发送完成（Transmission Completed) 0：发送未完成 1：发送已完成
8	HPM	RW	0	高优先级消息（High Priority Message) 0：未收到高优先级消息

				1：已收到高优先级消息
7	RF1L	RW	0	接收 FIFO1 消息丢失（Rx FIFO 1 Message Lost) 0：接收 FIFO1 消息未丢失 1：接收 FIFO1 消息有丢失；对空间大小为 0 的接收 FIFO1 进行写操作时，该位也会置位
6	RF1F	RW	0	接收 FIFO1 已满（Rx FIFO 1 Full) 0：接收 FIFO1 未满 1：接收 FIFO1 已满
5	RF1W	RW	0	接收 FIFO1 到达水位线（Rx FIFO 1 Watermark Reached) 0：接收 FIFO1 的填充级别低于水位线 1：接收 FIFO1 的填充级别达到水位线
4	RF1N	RW	0	接收 FIFO1 新消息（Rx FIFO 1 New Message) 0：没有新消息写入接收 FIFO1 1：有新消息写入接收 FIFO1
3	RF0L	RW	0	接收 FIFO0 消息丢失（Rx FIFO 0 Message Lost) 0：接收 FIFO0 消息未丢失 1：接收 FIFO0 消息有丢失；对空间大小为 0 的接收 FIFO0 进行写操作时，该位也会置位
2	RF0F	RW	0	接收 FIFO0 已满（Rx FIFO 0 Full) 0：接收 FIFO0 未满 1：接收 FIFO0 已满
1	RF0W	RW	0	接收 FIFO0 到达水位线（Rx FIFO 0 Watermark Reached) 0：接收 FIFO0 的填充级别低于水位线 1：接收 FIFO0 的填充级别达到水位线
0	RF0N	RW	0	接收 FIFO0 新消息（Rx FIFO 0 New Message) 0：没有新消息写入接收 FIFO0 1：有新消息写入接收 FIFO0

34.3.15 FDCAN 中断使能寄存器（FDCAN_IE），Address=0x54

地址偏移：0x054

复位值：0x0000 0000

中断使能寄存器中的设置，决定中断寄存器（FDCAN_IR）中的哪些状态变化会指示在中断线上。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res		DRXE	TOOE	MRAFE	TSWE
-		RW								-		RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
RW															

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	0	-
29	ARAE	RW	0	访问保留地址使能（Access to Reserved Address Enable) 0：禁止中断 1：使能中断
28	PEDE	RW	0	数据阶段的协议错误使能（Protocol Error in Data Phase Enable) 0：禁止中断 1：使能中断

27	PEAE	RW	0	仲裁阶段中的协议错误使能（Protocol Error in Arbitration Phase Enable) 0：禁止中断 1：使能中断
26	WDIE	RW	0	看门狗中断使能（Watchdog Interrupt Enable) 0：禁止中断 1：使能中断
25	BOE	RW	0	Bus_Off 状态中断（Bus_Off Status Interrupt Enable) 0：禁止中断 1：使能中断
24	EWE	RW	0	警告状态中断使能（Warning Status Interrupt Enable) 0：禁止中断 1：使能中断
23	EPE	RW	0	错误被动中断使能（Error Passive Interrupt Enable) 0：禁止中断 1：使能中断
22	ELOE	RW	0	错误记录溢出中断使能（Error Logging Overflow Interrupt Enable) 0：禁止中断 1：使能中断
21:20	保留	-	0	-
19	DRXE	RW	0	消息存储到专用接收缓冲区中断使能（Message stored to Dedicated Rx Buffer Interrupt Enable) 0：禁止中断 1：使能中断
18	TOOE	RW	0	发生超时中断使能（Timeout Occurred Interrupt Enable) 0：禁止中断 1：使能中断
17	MRAFE	RW	0	消息 RAM 访问失败中断使能（Message RAM Access Failure Interrupt Enable) 0：禁止中断 1：使能中断
16	TSWE	RW	0	时间戳回卷中断使能（Timestamp Wraparound Interrupt Enable) 0：禁止中断 1：使能中断
15	TEFLE	RW	0	发送事件 FIFO 元素丢失中断使能（Tx Event FIFO Element Lost Interrupt Enable) 0：禁止中断 1：使能中断
14	TEFFE	RW	0	发送事件 FIFO 满中断使能（Tx Event FIFO Full Interrupt Enable) 0：禁止中断 1：使能中断
13	TEFWE	RW	0	发送事件 FIFO 到达水位线中断使能(Tx Event FIFO Watermark Reached Interrupt Enable) 0：禁止中断 1：使能中断
12	TEFNE	RW	0	发送事件 FIFO 新条目中断使能（Tx Event FIFO New Entry Interrupt Enable) 0：禁止中断

				1：使能中断
11	TFEE	RW	0	发送 FIFO 空中断使能（Tx FIFO Empty Interrupt Enable) 0：禁止中断 1：使能中断
10	TCFE	RW	0	发送取消已完成中断使能（Transmission Cancellation Finished Interrupt Enable) 0：禁止中断 1：使能中断
9	TCE	RW	0	发送完成中断使能（Transmission Completed Interrupt Enable) 0：禁止中断 1：使能中断
8	HPME	RW	0	高优先级消息中断使能（High Priority Message Interrupt Enable) 0：禁止中断 1：使能中断
7	RF1LE	RW	0	接收 FIFO1 消息丢失中断使能（Rx FIFO 1 Message Lost Interrupt Enable) 0：禁止中断 1：使能中断
6	RF1FE	RW	0	接收 FIFO1 满中断使能（Rx FIFO 1 Full Interrupt Enable) 0：禁止中断 1：使能中断
5	RF1WE	RW	0	接收 FIFO1 到达水位线中断使能（Rx FIFO 1 Watermark Reached Interrupt Enable) 0：禁止中断 1：使能中断
4	RF1NE	RW	0	接收 FIFO1 新消息中断使能（Rx FIFO 1 New Message Interrupt Enable) 0：禁止中断 1：使能中断
3	RF0LE	RW	0	接收 FIFO0 消息丢失中断使能（Rx FIFO 0 Message Lost Interrupt Enable) 0：禁止中断 1：使能中断
2	RF0FE	RW	0	接收 FIFO0 满中断使能（Rx FIFO 0 Full Interrupt Enable) 0：禁止中断 1：使能中断
1	RF0WE	RW	0	接收 FIFO0 到达水位线中断使能（Rx FIFO 0 Watermark Reached Interrupt Enable) 0：禁止中断 1：使能中断
0	RF0NE	RW	0	接收 FIFO0 新消息中断使能（Rx FIFO 0 New Message Interrupt Enable) 0：禁止中断 1：使能中断

34.3.16 FDCAN 中断线选择寄存器（FDCAN_ILS），Address=0x58

地址偏移：0x058

复位值：0x0000 0000

中断线选择寄存器，将中断寄存器（FDCAN_IR）中特定中断标志生成的中断分配到两条中断线之一。要产生中断，必须通过 FDCAN_ILE.EINT0 和 FDCAN_ILE.EINT1 使能相应的中断线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.		DRXL	TOOL	MRAFL	TSWL
-		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**
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	0	-
29	ARAL	RW	0	访问保留地址中断线（Access to Reserved Address Line) 0：选择中断线 0 1：选择中断线 1
28	PEDL	RW	0	数据段协议错误中断线（Protocol Error in Data Phase Line) 0：选择中断线 0 1：选择中断线 1
27	PEAL	RW	0	仲裁段协议错误中断线（Protocol Error in Arbitration Phase Line) 0：选择中断线 0 1：选择中断线 1
26	WDIL	RW	0	看门狗中断线（Watchdog Interrupt Line) 0：选择中断线 0 1：选择中断线 1
25	BOL	RW	0	Bus_Off 状态中断线（Bus_Off Status Interrupt Line) 0：选择中断线 0 1：选择中断线 1
24	EWL	RW	0	警告状态中断线（Warning Status Interrupt Line) 0：选择中断线 0 1：选择中断线 1
23	EPL	RW	0	错误被动中断线（Error Passive Interrupt Line) 0：选择中断线 0 1：选择中断线 1
22	ELOL	RW	0	错误记录溢出中断线（Error Logging Overflow Interrupt Line) 0：选择中断线 0 1：选择中断线 1
21:20	保留	-	0	-
19	DRXL	RW	0	消息存储到专用接收缓冲区中断线（Message stored to Dedicated Rx Buffer Interrupt Line) 0：选择中断线 0 1：选择中断线 1
18	TOOL	RW	0	发生超时中断线（Timeout Occurred Interrupt Line) 0：选择中断线 0 1：选择中断线 1
17	MRAFL	RW	0	消息 RAM 访问失败中断线（Message RAM Access Failure Interrupt Line) 0：选择中断线 0 1：选择中断线 1

16	TSWL	RW	0	时间戳回卷中断线（Timestamp Wraparound Interrupt Line) 0：选择中断线 0 1：选择中断线 1
15	TEFLL	RW	0	发送事件 FIFO 元素丢失中断线（Tx Event FIFO Element Lost Interrupt Line) 0：选择中断线 0 1：选择中断线 1
14	TEFFL	RW	0	发送事件 FIFO 已满中断线（Tx Event FIFO Full Interrupt Line) 0：选择中断线 0 1：选择中断线 1
13	TEFWL	RW	0	发送事件 FIFO 到达水位线中断线（Tx Event FIFO Watermark Reached Interrupt Line) 0：选择中断线 0 1：选择中断线 1
12	TEFNL	RW	0	发送事件 FIFO 新条目中断线（Tx Event FIFO New Entry Interrupt Line) 0：选择中断线 0 1：选择中断线 1
11	TFEL	RW	0	发送 FIFO 为空中断线（Tx FIFO Empty Interrupt Line) 0：选择中断线 0 1：选择中断线 1
10	TCFL	RW	0	发送取消已完成中断线（Transmission Cancellation Finished Interrupt Line) 0：选择中断线 0 1：选择中断线 1
9	TCL	RW	0	发送完成中断线（Transmission Completed Interrupt Line) 0：选择中断线 0 1：选择中断线 1
8	HPML	RW	0	高优先级消息中断线（High Priority Message Interrupt Line) 0：选择中断线 0 1：选择中断线 1
7	RF1LL	RW	0	接收 FIFO1 消息丢失中断线（Rx FIFO 1 Message Lost Interrupt Line) 0：选择中断线 0 1：选择中断线 1
6	RF1FL	RW	0	接收 FIFO1 已满中断线（Rx FIFO 1 Full Interrupt Line) 0：选择中断线 0 1：选择中断线 1
5	RF1WL	RW	0	接收 FIFO1 到达水位线中断线（Rx FIFO 1 Watermark Reached Interrupt Line) 0：选择中断线 0 1：选择中断线 1
4	RF1NL	RW	0	接收 FIFO1 新消息中断线（Rx FIFO 1 New Message Interrupt Line) 0：选择中断线 0 1：选择中断线 1
3	RF0LL	RW	0	接收 FIFO0 消息丢失中断线（Rx FIFO 0 Message Lost Interrupt Line) 0：选择中断线 0 1：选择中断线 1
2	RF0FL	RW	0	接收 FIFO0 已已满中断线（Rx FIFO 0 Full Interrupt Line) 0：选择中断线 0

				1：选择中断线 1
1	RF0WL	RW	0	接收 FIFO0 到达水位线中断线（Rx FIFO 0 Watermark Reached Interrupt Line) 0：选择中断线 0 1：选择中断线 1
0	RF0NL	RW	0	接收 FIFO0 新消息中断线（Rx FIFO 0 New Message Interrupt Line) 0：选择中断线 0 1：选择中断线 1

34.3.17 FDCAN 中断线使能寄存器（FDCAN_ILE），Address=0x5C

地址偏移：0x05C

复位值：0x0000 0000

连接到 CPU 的两条中断线中的每一条，都可以通过设置 EINT0 和 EINT1 分别使能或禁用。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
-														RW	RW

Bit	Name	R/W	ResetValue	Function
31:2	保留	-	0	-
1	EINT1	RW	0	中断线 1 使能 0：中断线 1 禁止 1：中断线 1 使能
0	EINT0	RW	0	中断线 0 使能 0：中断线 0 禁止 1：中断线 0 使能

34.3.18 FDCAN 全局过滤器配置寄存器（FDCAN_GFC），Address=0x80

地址偏移：0x080

复位值：0x0000 0000

用于消息 ID 过滤的全局设置。全局过滤器配置，控制标准消息和扩展消息的过滤路径，详见图 34-5 和图 34-6。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
-										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	0	-
5:4	ANFS[1:0]	RW	0	接受不匹配的标准帧 定义如何处理，已接收的与过滤器列表任何元素都不匹配的 11 位标准 ID 消息。 00：在接收 FIFO0 中接受不匹配的标准帧

				01：在接收 FIFO1 中接受不匹配的标准帧 10：拒绝不匹配的标准帧 11：拒绝不匹配的标准帧
3:2	ANFE[1:0]	RW	0	接受不匹配的扩展帧 定义如何处理，已接收的与过滤器列表任何元素都不匹配的 29 位扩展 ID 消息。 00：在接收 FIFO0 中接受不匹配的扩展帧 01：在接收 FIFO1 中接受不匹配的扩展帧 10：拒绝不匹配的扩展帧 11：拒绝不匹配的扩展帧
1	RRFS	RW	0	拒绝远程标准帧 0：过滤 11 位标准 ID 的远程帧 1：拒绝所有 11 位标准 ID 的远程帧
0	RRFE	RW	0	拒绝远程扩展帧 0：过滤 29 位扩展 ID 的远程帧 1：拒绝所有 29 位扩展 ID 的远程帧

34.3.19 FDCAN 标准 ID 过滤器配置寄存器（FDCAN_SIDFC），Address=0x84

地址偏移：0x084

复位值：0x0000 0000

用于 11 位标准 ID 过滤器的设置。标准 ID 过滤器配置，控制标准消息的过滤路径，见图 34-5。

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

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	0	-
23:16	LSS[7:0]	RW	0	标准 ID 过滤器列表大小 0：无标准 ID 过滤器 1~128：标准 ID 过滤器元素数量 ＞128：大于 128 的设定值被解析为 128
15:2	FLSSA[15:2]	RW	0	标准 ID 过滤器列表起始地址 设置标准 ID 过滤器列表的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	-

34.3.20 FDCAN 扩展 ID 过滤器配置寄存器（FDCAN_XIDFC），Address=0x88

地址偏移：0x088

复位值：0x0000 0000

用于 29 位扩展 ID 过滤器的设置。扩展 ID 过滤器配置，控制扩展消息的过滤路径，见图 34-6。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[6:0]						

-									Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLESA[15:2]														Res.	Res.
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	

Bit	Name	R/W	Reset Value	Function
31:23	保留	-	0	-
22:16	LSE[7:0]	RW	0	扩展 ID 过滤器列表大小 0：无扩展 ID 过滤器 1~64：扩展 ID 过滤器元素数量 ＞64：大于 64 的设定值被解析为 64
15:2	FLESA[15:2]	RW	0	扩展 ID 过滤器列表起始地址 扩展 ID 过滤器列表的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	-

34.3.21 FDCAN 扩展 ID 与掩码寄存器（FDCAN_XIDAM），Address=0x90

地址偏移：0x090

复位值：0x1FFF FFFF

用于 29 位扩展 ID 过滤器的设置。扩展 ID 过滤器配置，控制扩展消息的过滤路径，见图 34-6。

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

Bit	Name	R/W	Reset Value	Function
31:29	保留	-	0	
28:0	EIDM[28:0]	RW	0	扩展 ID 掩码 对于扩展帧的接收过滤，会将 EIDM 与已接收帧的 ID 进行与运算。用于屏蔽 SAE J1939 中的 29 位 ID。所有位的复位值都被设置为 1，掩码无效。

34.3.22 FDCAN 高优先级消息状态寄存器（FDCAN_HPMS），Address=0x94

地址偏移：0x094

复位值：0x0000 0000

配置为生成优先级事件的消息 ID 过滤器元素，每次匹配时都会更新此寄存器。这可用于监测收到的高优先级消息的状态，并可对这些消息进行快速访问。

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]					
R	R							R		R					

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	-

15	FLST	R	0	过滤器列表 指示匹配的过滤器列表。 0：标准过滤器列表 1：扩展过滤器列表
14:8	FIDX[6:0]	R	0	过滤器索引 匹配的过滤器元素的索引。范围为 0 到 FDCAN_SIDFC.LSS 减 1 或 FDCAN_XIDFC.LSE 减 1。
7:6	MSI[1:0]	R	0	消息存储器指示符 00：未选择 FIFO 01：FIFO 消息丢失 10：消息存储在 FIFO0 中 11：消息存储在 FIFO1 中
5:0	BIDX[5:0]	R	0	缓存区索引 存储消息的接收 FIFO 元素的索引。仅当 MSI[1]=1 时有效。

34.3.23 FDCAN 新数据 1 寄存器（FDCAN_NDAT1），Address=0x98

地址偏移：0x098

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND[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**
ND[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	ND[31:0]	RW	0	新数据 该寄存器保存接收缓冲区 0 到 31 的新数据标志。当接收帧更新接收缓冲区时，对应的标志位会置 1。在主机将标志位清零之前，标志位保持置 1 状态。通过向对应位写 1 将标志位清零，写入 0 不起任何作用。硬复位会将寄存器清零。 0：接收缓冲区未更新 1：接收缓冲区通过新消息更新

34.3.24 FDCAN 新数据 2 寄存器（FDCAN_NDAT2），Address=0x9C

地址偏移：0x09C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND[63:48]															
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**
ND[47:32]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	ND[63:32]	RW	0	新数据

				该寄存器保存接收缓冲区 32 到 63 的新数据标志。当接收帧更新接收缓冲区时，对应的标志位会置 1。在主机将标志位清零之前，标志位保持置 1 状态。通过向对应位写 1 将标志位清零，写入 0 不起任何作用。硬复位会将寄存器清零。 0：接收缓冲区未更新 1：接收缓冲区通过新消息更新

34.3.25 FDCAN 接收 FIFO0 配置寄存器（FDCAN_RXF0C），Address=0xA0

地址偏移：0x0A0

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM[6:0]							Res.	F0S[6:0]						
RW	-							-	-						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA[15:2]														Res.	
-														-	

Bit	Name	R/W	Reset Value	Function
31	F0OM	RW	0	接收 FIFO0 工作模式 FIFO0 可工作在阻塞模式（Blocking mode）或覆盖模式（Overwrite mode） 0：接收 FIFO0 为阻塞模式 1：接收 FIFO0 为覆盖模式
30:24	F0WM[6:0]	RW	0	接收 FIFO0 水位线 0：禁止接收 FIFO0 水位线中断 1~64：接收 FIFO0 水位线中断（FDCAN_IR.RF0W）的级别 ＞64：禁止接收 FIFO0 水位线中断
23	保留	-	0	
22:16	F0S[6:0]	RW	0	接收 FIFO0 大小 0：无接收 FIFO0 1~64：接收 FIFO0 元素数量 ＞64：大于 64 的设定值被解析为 64 接收 FIFO0 元素的索引为 0 到 F0S 减 1
15:2	F0SA[15:2]	RW	0	接收 FIFO0 起始地址 消息 RAM 中接收 FIFO0 的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	

34.3.26 FDCAN 接收 FIFO0 状态寄存器（FDCAN_RXF0S），Address=0xA4

地址偏移：0x0A4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	F0PI[5:0]					
-						R	.R			R					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
-		R						-	R						

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	0	-
25	RF0L	R	0	接收 FIFO0 消息丢失 该位是中断标志 FDCAN_IR.RF0L 的副本。当 FDCAN_IR.RF0L 被复位时，该位也被复位 0：接收 FIFO0 无消息丢失 1：接收 FIFO0 有消息丢失；对空间大小为 0 的接收 FIFO0 进行写操作时，该位也会置位 注： 当 FDCAN_RXF0C.F0OM=1 时，覆盖最早的消息，不会置位该标志位。
24	F0F	R	0	接收 FIFO0 满 0：接收 FIFO0 未满 1：接收 FIFO0 已满
23:22	保留	-	0	-
21:16	F0PI[5:0]	R	0	接收 FIFO0 放入索引 接收 FIFO0 写入索引指针，范围 0~63
15:14	保留	-	0	-
13:8	F0GI[5:0]	R	0	接收 FIFO0 获取索引 接收 FIFO0 读取索引指针，范围 0~63
7	保留	-	0	-
6:0	F0FL[6:0]	R	0	接收 FIFO0 填充级别 接收 FIFO0 中存储的元素数量，范围 0~64

34.3.27 FDCAN 接收 FIFO0 确认寄存器（FDCAN_RXF0A），Address=0xA8

地址偏移：0x0A8

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
-										RW					

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	0	-
5:0	F0AI[5:0]	RW	0	接收 FIFO0 确认索引 主机从接收 FIFO0 读取消息或消息序列后，必须将从接收 FIFO0 中读取的最后一个元素的缓冲区索引写入 F0AI。此操作会将接收 FIFO0 获取索引 FDCAN_RXF0S.F0GI 设为 F0AI 加 1，并会更新 FIFO0 填充级别 FDCAN_RXF0S.F0FL

34.3.28 FDCAN 接收缓冲区配置寄存器（FDCAN_RXBC），Address=0xAC

地址偏移：0x0AC

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	0	-
15:2	RBSA[15:2]	RW	0	接收缓冲区的起始地址 配置消息 RAM 中接收缓冲区的起始地址（32 位地址），也用于引用调试消息 A、B、C
1:0	保留	-	0	-

34.3.29 FDCAN 接收 FIFO1 配置寄存器（FDCAN_RXF1C），Address=0xB0

地址偏移：0x0B0

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F1OM	F1WM[6:0]							Res.	F1S[6:0]						
RW	RW							-	RW						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F1SA[15:2]														Res.	
RW														-	

Bit	Name	R/W	Reset Value	Function
31	F1OM	RW	0	接收 FIFO1 操作模式 FIFO1 可工作在阻塞模式（Blocking mode）或覆盖模式（Overwrite mode） 0：接收 FIFO1 为阻塞模式 1：接收 FIFO1 为覆盖模式
30:24	F1WM[6:0]	RW	0	接收 FIFO1 水印 0：禁止接收 FIFO1 水位线中断 1~64：接收 FIFO1 水位线中断（FDCAN_IR.RF1W）的级别 ＞64：禁止接收 FIFO1 水位线中断
23	保留	-	0	-
22:16	F1S[6:0]	RW	0	接收 FIFO1 大小 0：无接收 FIFO1 1~64：接收 FIFO1 元素数量 ＞64：大于 64 的设定值被解析为 64 接收 FIFO1 元素的索引为 0 到 F1S 减 1
15:2	F1SA[15:2]	RW	0	接收 FIFO1 起始地址 消息 RAM 中接收 FIFO1 的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	-

34.3.30 FDCAN 接收 FIFO1 状态寄存器（FDCAN_RXF1S），Address=0xB4

地址偏移：0x0B4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	F1PI[5:0]					
R		-				R	R	-		RW					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
-		R						-	R						

Bit	Name	R/W	Reset Value	Function
31:30	DMS[1:0]	R	0	调试消息状态 00：空闲状态，等待接收调试消息 01：已接收调试消息 A 10：已接收调试消息 A、B 11：已接收调试消息 A、B、C
29:26	保留	-	0	-
25	RF1L	R	0	接收 FIFO1 消息丢失 0：接收 FIFO1 无元素丢失 1：接收 FIFO1 有元素丢失；对空间大小为 0 的接收 FIFO1 进行写操作时，该位也会置位 注： 当 FDCAN_RXF1C.F1OM=1 时，覆盖最早的消息，不会置位该标志位。
24	F1F	R	0	接收 FIFO1 满 0：接收 FIFO1 未满 1：接收 FIFO1 已满
23:22	保留	-	0	-
21:16	F1PI[5:0]	RW	0	接收 FIFO1 放入索引 接收 FIFO1 写入索引指针，范围 0~63
15:14	保留	-	0	-
13:8	F1GI[5:0]	R	0	接收 FIFO1 获取索引 接收 FIFO1 读取索引指针，范围 0~63
7	保留	-	0	-
6:0	F1FL[6:0]	R	0	接收 FIFO1 填充级别 接收 FIFO1 中存储的元素数量，范围 0~64

34.3.31 FDCAN 接收 FIFO1 确认寄存器（FDCAN_RXF1A），Address=0xB8

地址偏移：0x0B8

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
-										RW					

Bit	Name	R/W	Reset Value	Function
31:6	保留	-	0	-
5:0	F1AI	RW	0	接收 FIFO1 确认索引 主机从接收 FIFO1 读取消息或消息序列后，必须将从接收 FIFO1 中读取的最后一个元素的缓冲区索引写入 F1AI。此操作会将接收 FIFO1 获取索引 FDCAN_RXF1S.F1GI 设为 F1AI 加 1，并会更新 FIFO1 填充级别 FDCAN_RXF1S.F1FL

34.3.32 FDCAN 接收缓冲区和 FIFO 元素大小配置寄存器（FDCAN_RXESC），Address=0xBC

地址偏移：0x0BC

复位值：0x0000 0000

配置接收缓冲区和接收 FIFO 元素的数据字节数。大于 8 字节的数据段仅适用于 CAN FD 模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RBDS[2:0]			Res.	F1DS[2:0]			Res.	F0DS[2:0]		
-					RW			-	RW			-	RW		

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	0	-
10:8	RBDS[2:0]	RW	0	接收缓冲区数据段大小 000：8 字节 001：12 字节 010：16 字节 011：20 字节 100：24 字节 101：32 字节 110：48 字节 111：64 字节
7	保留	-	0	-
6:4	F1DS[2:0]	RW	0	接收 FIFO1 数据段大小 000：8 字节 001：12 字节 010：16 字节 011：20 字节 100：24 字节 101：32 字节 110：48 字节 111：64 字节
3	保留	-	0	-
2:0	F0DS[2:0]	RW	0	接收 FIFO0 数据段大小 000：8 字节 001：12 字节 010：16 字节

				011：20 字节 100：24 字节 101：32 字节 110：48 字节 111：64 字节

34.3.33 FDCAN 发送缓冲区配置寄存器（FDCAN_TXBC），Address=0xC0

地址偏移：0x0C0

复位值：0x0000 0000

仅当 FDCAN_CCCR 寄存器的 CCE 位和 INIT 位均被置 1 时，才能对该寄存器进行写访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]					
-	RW	RW						-		RW					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TBSA[13:0]														Res.	Res.
RW														-	

Bit	Name	R/W	Reset Value	Function
31	保留	-	0	-
30	TFQM	RW	0	发送 FIFO/队列模式 0：发送 FIFO 模式 1：发送队列模式
29:24	TFQS[5:0]	RW	0	发送 FIFO/队列大小 0：无发送 FIFO 或队列 1-32：发送 FIFO 队列的缓冲区大小 ＞32：大于 32 的设定值被解析为 32
23:22	保留	-	0	-
21:16	NDTB[5:0]	RW	0	专用发送缓冲区大小 0：无专用发送缓冲区 1-32：专用发送缓冲区大小 ＞32：大于 32 的设定值被解析为 32
15:2	TBSA[15:2]	RW	0	发送缓冲区起始地址 消息 RAM 中发送缓冲区部分的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	-

注意：

TFQS 和 NDTB 之和不得大于 32。FDCAN 没有对该设置错误的监测。消息 RAM 中的发送缓冲区部分从专用发送缓冲区开始。

34.3.34 FDCAN 发送 FIFO/队列状态寄存器（FDCAN_TXFQS），Address=0xC4

地址偏移：0x0C4

复位值：0x0000 0000

发送 FIFO 或队列的状态与寄存器 FDCAN_TXBRP 中挂起的发送请求有关。因此，由于正在运行的发送扫描（TXBRP 尚未更新），添加或取消请求的效果可能会延迟。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]				
-										R	R				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
-			R					-		R					

Bit	Name	R/W	Reset Value	Function
31:22	保留	-	0	-
21	TFQF	R	0	发送 FIFO/队列满 0：发送 FIFO 或队列未满 1：发送 FIFO 或队列已满
20:16	TFQPI[4:0]	R	0	发送 FIFO/队列放入索引 发送 FIFO 或队列的放入索引指针，范围 0-31
15:13	保留	-	0	-
12:8	TFGI[4:0]	R	0	发送 FIFO 获取索引 发送 FIFO 的读取索引指针，范围 0-31。若已配置为发送队列模式（FDCAN_TXBC.TFQM=1），则该位读取值为 0
7:6	保留	-	0	-
5:0	TFFL[5:0]	R	0	发送 FIFO 空闲级别 从 TFGI 开始的连续空闲的发送 FIFO 元素数量，范围 0-32。若已配置为发送队列模式（FDCAN_TXBC.TFQM=1），则该位读取值为 0

注：在专用发送缓冲区与发送 FIFO 或发送队列组合配置的情况下，放入索引和获取索引是指，从第一个专用发送缓冲区开始的发送缓冲区编号。

例如：对于有 12 个专用发送缓冲区和 20 个发送 FIFO 缓冲区的组合配置，放入索引 15 指向发送 FIFO 的第 4 个缓冲区。

34.3.35 FDCAN 发送缓冲区元素大小配置寄存器（FDCAN_TXESC），Address=0xC8

地址偏移：0x0C8

复位值：0x0000 0000

配置发送缓冲区元素的数据字节数。大于 8 字节的数据段仅适用于 CAN FD 模式。

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

Bit	Name	R/W	ResetValue	Function
31:3	保留	-	0	-
2:0	TBDS[2:0]	RW	0	发送缓冲区数据段大小 000：8 字节 001：12 字节 010：16 字节 011：20 字节

				100：24 字节 101：32 字节 110：48 字节 111：64 字节

注：如果发送缓冲区元素的数据长度（由 DLC 配置），大于 TBDS 配置的数据长度，则发送缓冲区中未定义的数据字节将会以 0xCC（填充字节）发送。

34.3.36 FDCAN 发送缓冲区请求挂起寄存器（FDCAN_TXBRP），Address=0xCC

地址偏移：0x0CC

复位值：0x0000 0000

配置发送缓冲区元素的数据字节数。大于 8 字节的数据段仅适用于 CAN FD 模式。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP[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**
TRP[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	TRP[31:0]	R	0	发送请求挂起标志 每个发送缓冲区都有自己的发送请求挂起位。这些位通过寄存器 FDCAN_TXBAR 置位；在请求的发送已完成，或请求的发送已通过寄存器 FDCAN_TXBCR 取消后复位。 只有 FDCAN_TXBC 配置了的发送缓冲区对应的 TXBRP 位才会置位。 TXBRP 位置位后会启动发送扫描（见章节【发送处理】），以检查优先级最高的挂起的发送请求（即消息 ID 最小的发送缓冲区）。 取消发送请求会复位相应的 TXBRP 位。如果请求取消时发送已经开始，则无论发送是否成功，在发送结束时 TXBRP 位都会被复位。TXBRP 位复位后，对应的取消请求位也会立即复位。 请求取消后，以下情况通过寄存器 FDCAN_TXBCF 指示取消已完成： • 发送成功及相应的 FDCAN_TXBTO 位置位后 • 取消时发送尚未开始 • 发送因仲裁失败而中止 • 帧发送期间发生错误 在 DAR 模式下，所有发送在失败后都会自动取消。所有失败的发送，对应的 FDCAN_TXBCF 位都会置位。 0：无发送请求挂起 1：发送请求挂起 注：在发送扫描期间置位的 TXBRP 位，在本次发送扫描期间被忽略，在此期间，如果请求取消这些发送，则发送被立即取消，对应的 TXBRP 位复位。

34.3.37 FDCAN 发送缓冲区添加请求寄存器（FDCAN_TXBAR），Address=0xD0

地址偏移：0x0D0

复位值：0x0000 0000

配置发送缓冲区元素的数据字节数。大于 8 字节的数据段仅适用于 CAN FD 模式。

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

Bit	Name	R/W	Reset Value	Function
31:0	AR[31:0]	RW	0	添加请求 每个发送缓冲区都有自己的添加请求位，写 1 置位相应的添加请求位，写 0 无影响。主机可一次设置多个发送缓冲区的发送请求。只有 FDCAN_TXBC 配置了的发送缓冲区对应的 TXBAR 位才会置位。没有运行发送扫描时，这些位立即复位，否则将保持置位到发送扫描完成。 0：未添加发送请求 1：已添加发送请求 注： 如果添加请求的发送缓冲区已有发送请求挂起（FDCAN_TXBRP 寄存器相应位已置位），该添加请求将被忽略。

34.3.38 FDCAN 发送缓冲区取消请求寄存器（FDCAN_TXBCR），Address=0xD4

地址偏移：0x0D4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR[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
CR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	CR[31:0]	RW	0	取消请求 每个发送缓冲区都有自己的取消请求位，写 1 置位相应的取消请求位，写 0 无影响。主机可一次设置多个发送缓冲区的取消请求。只有 FDCAN_TXBC 配置了的发送缓冲区对应的 TXBCR 位才会置位。这些位保持置位直到对应的 FDCAN_TXBRP 位复位。 0：无取消请求挂起 1：有取消请求挂起

34.3.39 FDCAN 发送缓冲区发送已发生寄存器（FDCAN_TXBTO），Address=0xD8

地址偏移：0x0D8

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO[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
TO[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	TO[31:0]	R	0	发送已发生 每个发送缓冲区都有自己的发送已发生标志位。当发送缓冲区发送完成，FDCAN_TXBRP 相应位清零时，对应的发送已发生标志位置位。向 FDCAN_TXBAR 相应位写 1 请求新的发送时，对应的发送已发生标志位复位。 0：未进行发送 1：已进行发送

34.3.40 FDCAN 发送缓冲区取消已完成寄存器（FDCAN_TXBCF），Address=0xDC

地址偏移：0x0DC

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF[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**
CF[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	CF[31:0]	R	0	取消已完成 每个发送缓冲区都有自己的取消已完成标志位。当通过 FDCAN_TXBCR 请求的取消已完成，FDCAN_TXBRP 相应位清零时，对应的取消已完成标志位置位。如果 FDCAN_TXBRP 相应位在取消时没有置位，取消已完成标志位会立即置位。向 FDCAN_TXBAR 相应位写 1 请求新的发送时，对应的取消已完成标志位复位。 0：无发送缓冲区取消 1：发送缓冲区已取消

34.3.41 FDCAN 发送缓冲区发送中断使能寄存器（FDCAN_TXBTIE），Address=0xE0

地址偏移：0x0E0

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE[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
TIE[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	TIE[31:0]	RW	0	发送中断使能 每个发送缓冲区都有自己的发送中断使能位。 0：禁止发送中断 1：使能发送中断

34.3.42 FDCAN 发送缓冲区取消已完成中断使能寄存器（FDCAN_TXBCIE），Address=0xE4

地址偏移：0x0E4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE[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
CFIE[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:0	CFIE[31:0]	RW	0	取消已完成中断使能 每个发送缓冲区都有自己的取消已完成中断使能位。 0：禁止取消已完成中断 1：使能取消已完成中断

34.3.43 FDCAN 发送事件 FIFO 配置寄存器（FDCAN_TXEFC），Address=0xF0

地址偏移：0x0F0

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:30	保留	-	0	-
29:24	EFWM[5:0]	RW	0	事件 FIFO 水位线 0：禁止水位线中断 1~32：发送事件 FIFO 水位线中断（FDCAN_IR.TEFW）的级别 ＞32：禁止水位线中断
23:22	保留	-	0	-
21:16	EFS[5:0]	RW	0	事件 FIFO 大小 0：禁止发送事件 FIFO 1-32：发送事件 FIFO 的元素数量 ＞32：大于 32 的设定值被解析为 32 发送事件 FIFO 元素的索引为 0 到 EFS 减 1
15:2	EFSA[15:2]	RW	0	事件 FIFO 起始地址 消息 RAM 中事件 FIFO 的起始地址（32 位字地址，见图 34-11）
1:0	保留	-	0	-

34.3.44 FDCAN 发送事件 FIFO 状态寄存器（FDCAN_TXEFS），Address=0xF4

地址偏移：0x0F4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						TEFL	EFF	Res			EFPI[4:0]				

-						R	R	-			R				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	EFGI[4:0]					Res			EFFL[5:0]				
-			R					-			-				

Bit	Name	R/W	Reset Value	Function
31:26	保留	-	0	-
25	TEFL	R	0	发送事件 FIFO 元素丢失 该位是中断标志 FDCAN_IR. TEFL 的副本。当 FDCAN_IR. TEFL 被复位时，该位也被复位 0：发送事件 FIFO 无元素丢失 1：发送事件 FIFO 有元素丢失；对空间大小为 0 的发送事件 FIFO 进行写操作时，该位也会置位
24	EFF	R	0	事件 FIFO 已满 0：发送事件 FIFO 未满 1：发送事件 FIFO 已满
23:21	保留	-	0	-
20:16	EFPI[4:0]	R	0	事件 FIFO 放入索引 发送事件 FIFO 写入索引指针，范围 0~31
15:13	保留	-	0	-
12:8	EFGI[4:0]	R	0	事件 FIFO 获取索引 发送事件 FIFO 读取索引指针，范围 0~31
7:6	保留	-	0	-
5:0	EFFL[5:0]	R	0	事件 FIFO 填充级别 接收 FIFO0 中存储的元素个数，范围 0~32

34.3.45 FDCAN 发送事件 FIFO 确认寄存器（FDCAN_TXEFA），Address=0xF8

地址偏移：0x0F8

复位值：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	0	-
4:0	EFAI[4:0]	RW	0	事件 FIFO 确认索引 主机从发送事件 FIFO 读取元素或元素序列后，必须将从事件 FIFO 中读取的最后一个元素的索引写入 EFAI。此操作会将发送事件 FIFO 获取索引 FDCAN_TXEFS.EFGI 设为 EFAI 加 1，并会更新事件 FIFO 填充级别 FDCAN_TXEFS.EFFL。

35. 串行外设接口/片上音频接口(SPI/I²S)

35.1 概述

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。

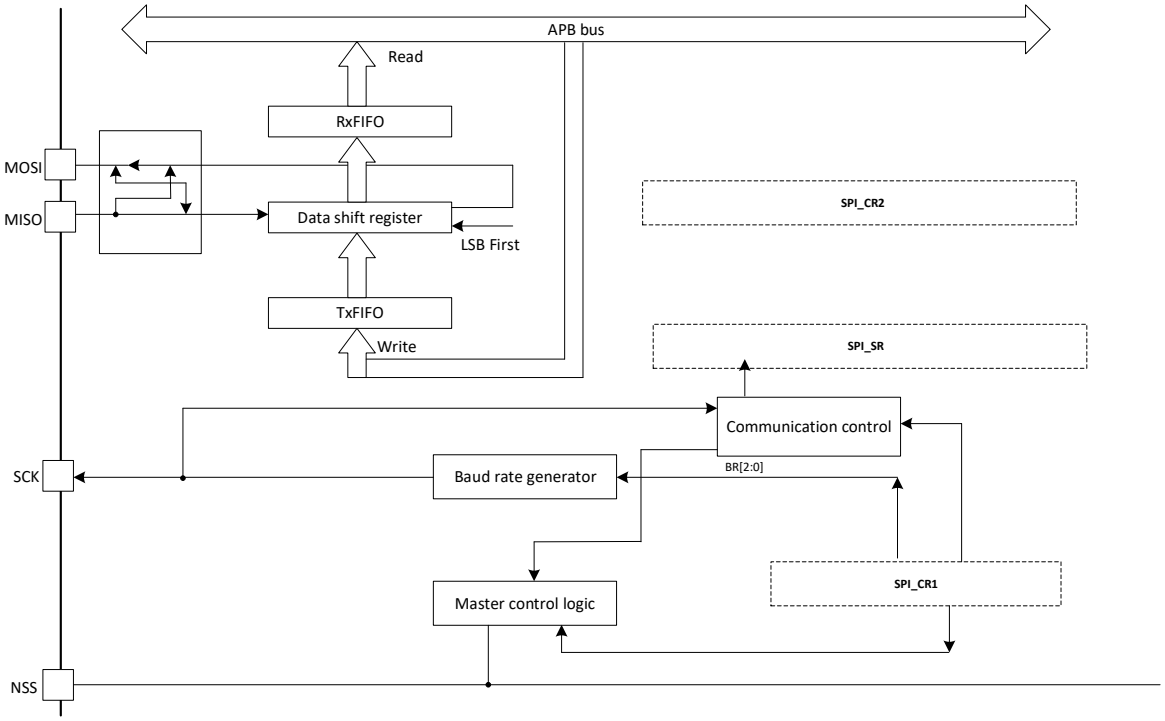


图 35-1 SPI 框图

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，还可使用经 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，通过接口为外部的从设备提供时钟信号。

35.1.1 SPI 特征

- 3 线全双工同步传输
- 2 线半双工同步传输（有双向数据线）
- 2 线单工同步传输（无双向数据线）
- 8 或 16 位传输帧格式选择
- 主或从操作
- 支持多主模式
- 8 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 $f_{PCLK}/2$)

- 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
- 两个具有 DMA 功能的深度为 4、宽度为 16bit（当数据帧设置为 8bit 时，宽度为 8bit）的嵌入式 Rx 和 Tx FIFO

35.1.2 I²S 特征

- 单工通信(仅发送或接收)
- 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(8 ~ 192 kHz)
- 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/从接收模式下的溢出标志位
- 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)

35.2 SPI 功能描述

35.2.1 单主机和单从机通信

针对不同的应用场景，SPI 可以使用几种不同的配置进行通讯。这些配置使用 2 线、3 线（软件 NSS 管理）或者 4 线（硬件 NSS 管理）。

35.2.1.1 全双工通信

SPI 被配置成全双工通讯的情况下，主机和从机的移位寄存器，在 MOSI 和 MISO 之间，使用两个单向的线连到一起。在 SPI 通讯期间，数据在主机提供的时钟沿同步移位。主机通过 MOSI 发送数据，从 MISO 接收来自从机的数据。当数据帧传输完成，结束在主机和从机之间的信息交互。

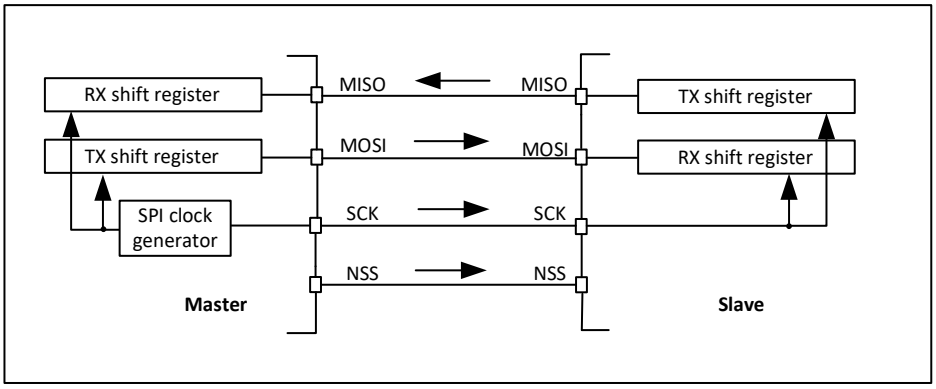


图 35-2 全双工单主机/单从机应用

35.2.1.2 半双工通信

通过设定 BIDIMODE（SPI_CR1 寄存器），SPI 可以工作在半双工模式。在这种配置下，用 1 根数据线完成主机和从机移位寄存器的连接。在通讯过程中，在 SCK 的时钟沿，数据在两个移位寄存器之间以 BIDIOE（SPI_CR1 寄存器）选择的方向同步移位。在该配置下，主机的 MISO 和从机的 MOSI 可被释放作为通用端口给其他应用使用。

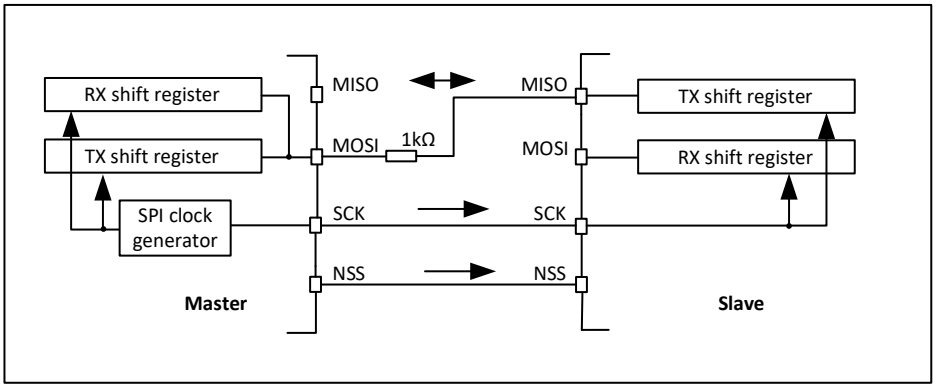


图 35-3 半双工单主机/单从机应用

NSS 可以被使用在主机和从机之间进行硬件控制。也可以使用软件控制 NSS 单工通信。

通过使用 RXONLY（SPI_CR1 寄存器），设定 SPI 在只发送模式或者只接收模式，使 SPI 工作在单线模式下。在这个配置下，在主机和从机的移位寄存器之间只使用 1 根线。在该配置下，主机的 MISO 和从机的 MOSI 可被释放作为通用端口给其他应用使用。

- 只接收模式（RXONLY=1）：通过置位 RXONLY，应用可以禁止 SPI 输出功能。在从机配置，MISO 输出被禁止，该端口被用作 GPIO。当从机 NSS 信号有效时，从机继续从 MOSI 接收数据。在主机配置下，MOSI 输出被禁止，该端口可以用作 GPIO。只要 SPI 被使用，时钟信号就被连续的产生。停止时钟的唯一方法是清零 RXONLY 或者 SPE，直到来自 MISO 的输入完成。

- 只发送模式（RXONLY=0）：配置与全双工相同。未使用的端口可以被用作标准的 GPIO。

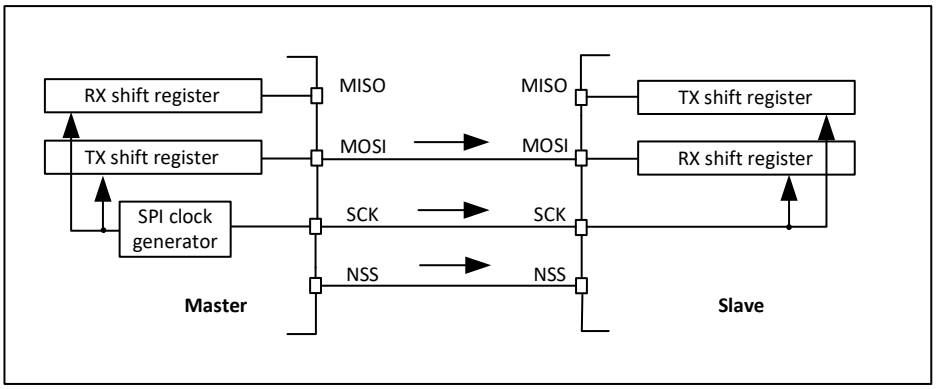


图 35-4 单工单从机/单主机应用

(主设备处于仅发送模式/从设备处于仅接收模式)

（1）在主机和从机之间可以使用 NSS 进行硬件控制。NSS 可选软件控制。

（2）在标准的只发模式下，所有与传输接收相关的事件都被必须被忽略。

（3）在该配置下，两边的 MISO pin 都可以被用作 GPIO。

注：设定 BIDIMODE 双向模式，任何单工通讯可以变更为半双工通讯。

35.2.2 多从机通信

在一个有两个或者更多独立从机的应用场景里，主机使用 GPIO 控制 NSS 来管理从机。主机必须通过拉低连接的从机 NSS，选择某个从机。当完成这个操作，就建立了主机和选定的从机通讯。

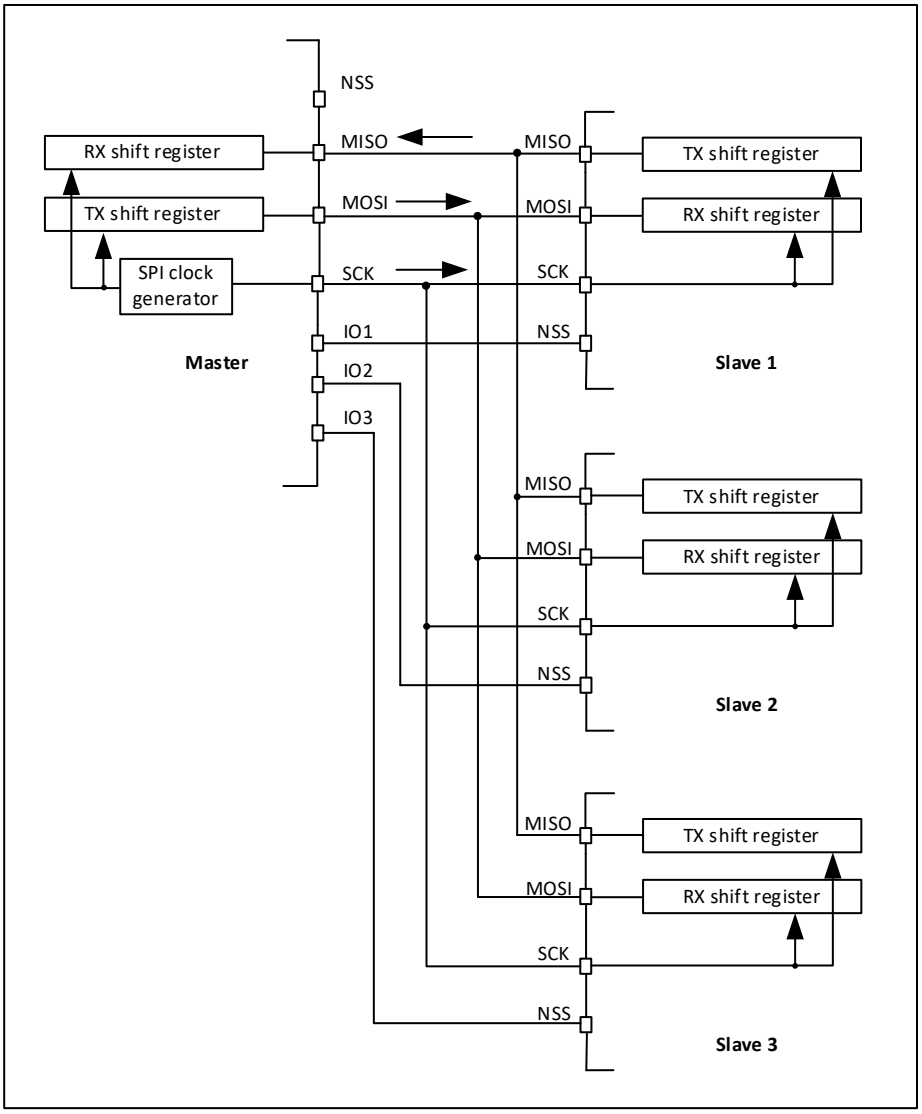


图 35-5 主机与三个独立的从机通信

NSS 在这种配置下在主机端未被使用。必须通过 SSM=1, SSI=1 来防止任何 MODF 错误。

由于从机的 MISO 连接到一起，所有从机必须把他们 MISO 的 GPIO 配置作为开漏模式。

35.2.3 多主机通信

SPI 总线被设计成具备多主机功能，用户可以使用其内嵌特性，该特性可以发现两个试图同时控制总线节点的潜在冲突。当需要用到这种检测时，要使用配置为硬件输入模式的 NSS 引脚。

在这种模式下有两个以上 SPI 节点的连接是不可能的，因为单次只有一个节点可以在公共数据线上传输。当节点无效，默认两个都保持从机模式。一旦节点要控制总线，切换成主机模式，并把有效的电平经过 GPIO 给予其余节点的 NSS 输入。在该进程完成后，有效的从机选择信号被释放，控制总线的节点暂时回到待机，并等待新的进程开始。

如果两个节点在同一时间都给出控制请求，总线冲突就会产生 MODF 事件。

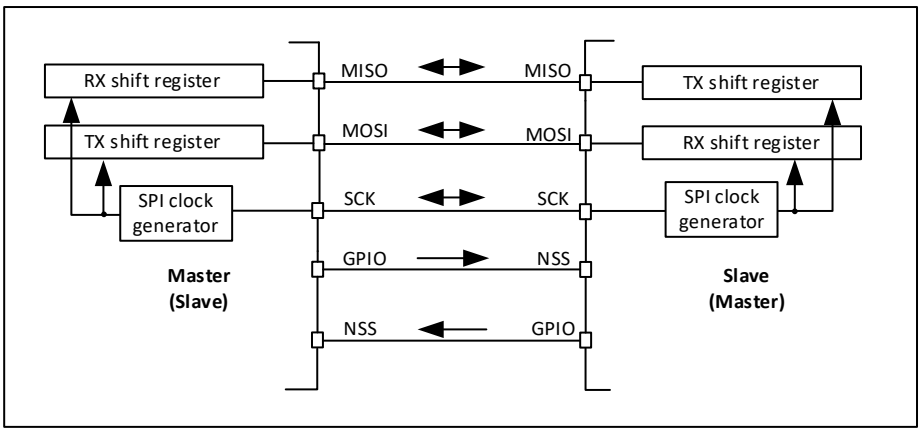


图 35-6 多主机应用

35.2.4 从机选择(NSS)引脚管理

在从机模式，NSS 作为标准的片选输入，使从机能与主机通讯。在主机模式，NSS 既可以作为输出又可以作为输入。当作为输入时，它可以防止多主机的总线冲突，当作为输出时，它可以驱动单个从机的从机选择引脚。

通过 SPI_CR1 寄存器的 SSM bit，可以选择硬件或者软件进行从机管理：

- 软件 NSS 管理（SSM=1）：在这个配置下，从机选择信号被内部的 SSI bit（SPI_CR1 寄存器）值驱动。外部 NSS 引脚被释放给其他应用使用。
- 硬件 NSS 管理（SSM=0）：在这个情况下，有几个可能的配置。
  1) NSS 输出使能（SSM=0, SSOE=1）：这个配置仅在作为主机时使用。硬件管理 NSS 引脚。当 SPI 在主机模式被使能（SPE=1），NSS 信号就被拉低并保持低电平，直到 SPI 被关闭（SPE=0）。在多主机应用中，SPI 不能进行这种 NSS 配置。
  2) NSS 输出禁止（SSM=0, SSOE=0）：如果 MCU 在总线上作为主机，这个配置允许进行多主机通讯。如果 NSS 引脚此时被拉低，SPI 进入主机模式默认状态，自动被配置为从机模式。在从机模式，NSS 引脚作为标准的片选输入，当 NSS 为低时，从机被选中。

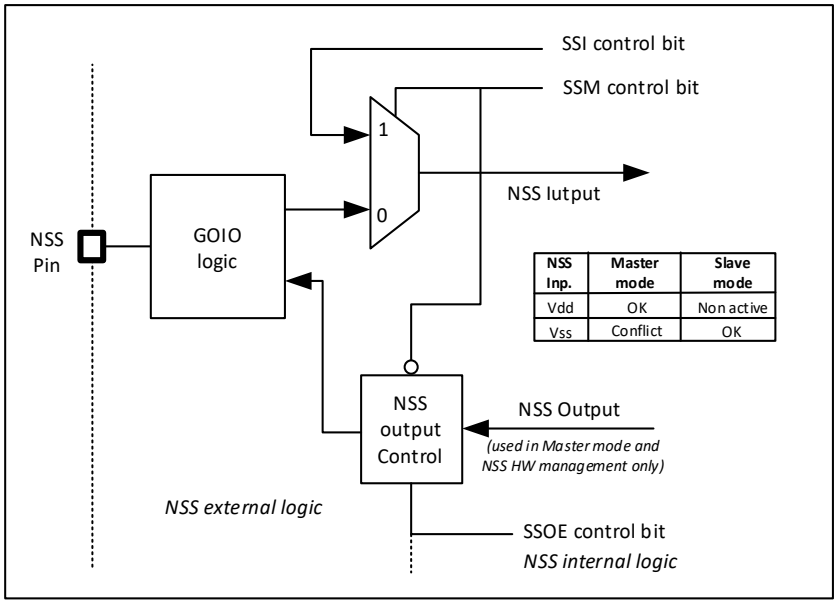


图 35-7 硬件/软件管理从机

35.2.5 通讯格式

在 SPI 通讯期间，接收和发送操作同时进行。SCK 将数据线上的数据移位和采样操作同步。通讯格式取决于时钟相位、时钟极性和数据帧格式。为了能够进行通讯，主机和从机必须遵循相同的通讯格式。

35.2.5.1 时钟极性和相位

通过 SPI_CR1 寄存器中的 CPOL 和 CPHA 位，可以用软件选择四种可能的时序关系。

CPOL（时钟极性）位控制不传任何数据时的时钟电平状态。此位对主器件和从器件都有作用。如果复位 CPOL，SCK 引脚在空闲状态处于低电平。如果将 CPOL 置 1，SCK 引脚在空闲状态处于高电平。

如果将 CPHA（时钟相位）位置 1，则 SCK 引脚上的第二个边沿（如果复位 CPOL 位，则为下降沿；如果将 CPOL 位置 1，则为上升沿）对 MSBit 采样。即，在第二个时钟边沿锁存数据。如果复位 CPHA 位，则 SCK 引脚上的第一个边沿（如果将 CPOL 位置 1，则为下降沿；如果复位 CPOL 位，则为上升沿）对 MSBit 采样。即，在第一个时钟边沿锁存数据。

CPOL 和 CPHA 的组合选择了数据采样时钟边沿。

在 CPOL/CPHA 改变之前，SPI 必须被关闭（SPE=0）。

SCK 的 IDLE 状态必须对应 SPI_CR1 寄存器选择的极性。

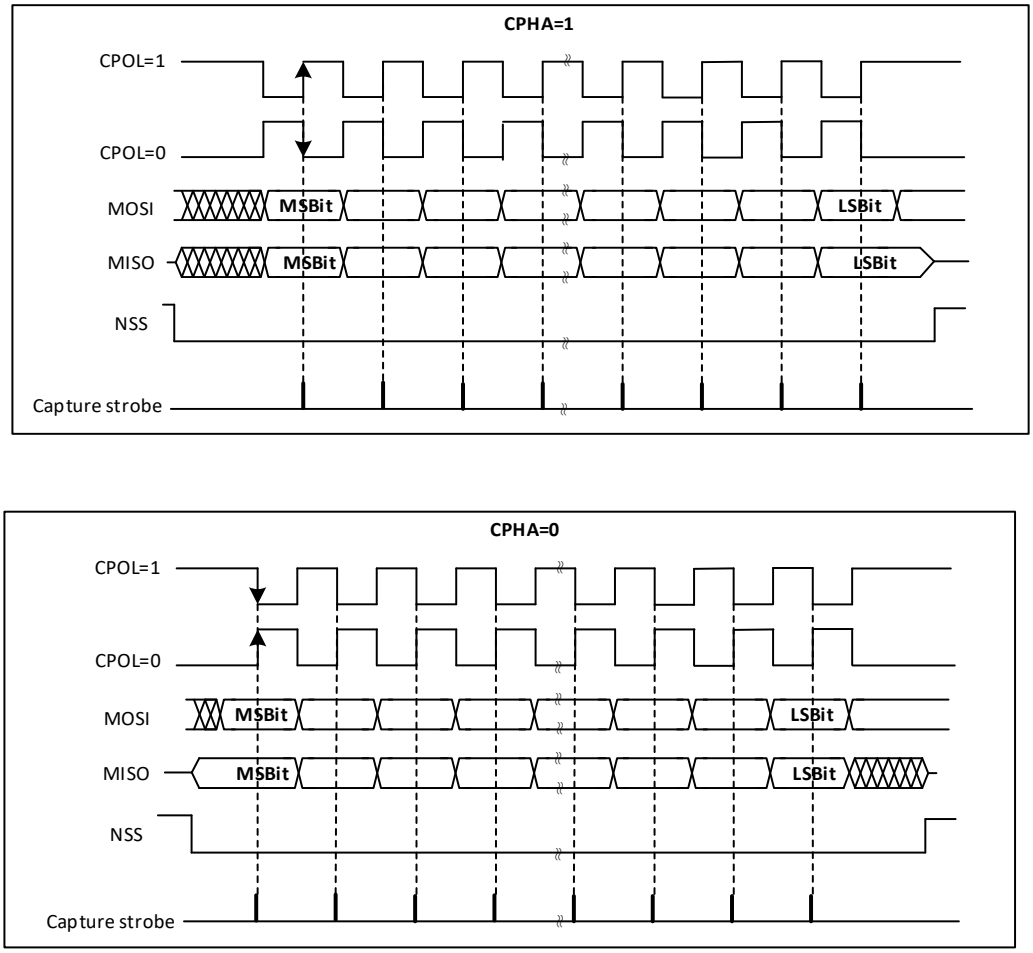


图 35-8 数据时钟时序图

数据位的顺序取决于 LSBFIRST(SPI_CR1 寄存器)的设定。

35.2.5.2 数据帧格式

通过 LSBFIRST(SPI_CR1 寄存器)，SPI 移位寄存器可以设定为 MSB-FIRST 或者 LSB-FIRST。通过使用 DFF（SPI_CR1 寄存器)，选择数据帧的位数，可选择为 8 位或者 16 位长度，该设置对于发送和接收都适用。

35.2.6 SPI 配置

对于主机和从机，SPI 的配置流程几乎一样。当进行标准的通讯，进行以下步骤：

1. 写 GPIO 寄存器：配置 MOSI、MISO 和 SCK 引脚
2. 写 SPI_CR1 寄存器
   1) 通过 BR[2:0]配置时钟波特率
   2) 配置 CPOL 和 CPHA
   3) 通过 RXONLY 或者 BIDIMODE 和 BIDIOE（RXONLY 和 BIDIMODE 不能同时有效），选择单工或者半双工模式
   4) 配置 LSBFIRST
   5) 配置 SSM 和 SSI

6) 配置 MSTR （在多主机 NSS 配置中，要避免 NSS 的冲突，主机配置 MSTR 来防止 MODF 错误）

3. 写 SPI_CR2 寄存器
   1) 配置 DS bit，选择数据帧位数
   2) 配置 SSOE（从机模式不需要）

35.2.7 SPI 使能流程

在主机发送时钟之前使能 SPI 从机。如果不这样处理，不期望的数据传输可能会发生。在开始与主机通讯之前，从机的数据寄存器必须写入要发送的数据。SCK 信号必须在 SPI 从机被使能之前，固定在 IDLE 状态电平。

全双工模式（或者只发），当 SPI 被使能并且 TXFIFO 不空，或者向 TXFIFO 进行下一个写，主机开始通讯。

在任何主机只接收模式（RXONLY=1，或者 BIDIMODE=1 且 BIDIOE=0），在 SPI 被使能后，主机开始通讯，立即提供时钟。

35.2.8 数据传输和接收流程

35.2.8.1 接收 FIFO 和发送 FIFO

SPI 所有数据通讯都通过 FIFO，深度为 4，宽度为 16bit（当数据帧设置为 8bit 时，宽度为 8bit）。该特性使 SPI 能够以连续数据流进行工作，并防止由于 CPU 来不及处理数据导致的通讯问题。发送和接收有独立的 FIFO，叫做 TXFIFO 和 RXFIFO。这些 FIFO 被用在所有的 SPI 模式。

FIFO 的处理取决于多种参数，包括：数据交换模式（全双工、半双工）、数据帧格式。

读 SPI_SR 寄存器会得到最早存放在 RXFIFO 中还未被读走的数据结果。写 SPI_DR 寄存器，会在 FIFO 发送队列的最后位置，存入被写的数据。读访问必须通常与 RXFIFO 阈值对齐。FTLVL 和 FRLVL 位显示了两个 FIFO 当前的占用情况。

对 SPI_DR 寄存器的访问必须通过 RXNE 事件管理。当数据存储器在 RXFIFO 并且达到阈值，该事件被触发。当 RXNE 被清零，RXFIFO 就被认为是空的。

写要发送的数据帧，通过 TXE 事件管理。当 TXFIFO 水位线小于或者等于总容量的一半时，该事件就会被触发。否则，TXE 被清零，并且 TXFIFO 被认为是满的。

用这样的方式，RXFIFO 可以存 2 个数据帧。

TXE 和 RXNE 事件都可以通过查询、中断方式处理。

当 RXFIFO 满时，如果下一个数据被接收，则溢出事件产生。溢出事件可以通过查询和中断的方式处理。

被置位的 BSY 位显示了当前数据帧的通讯正在进行。当时钟信号被连续的提供，在主机端的两个数据帧之间，BSY 标志保持置位。但在从机端的每个数据帧传输之间，BSY 会保持最小 1 个 SPI 时钟低电平。

35.2.8.2 处理序列

一些数据帧可以通过初始序列传递来完成一条信息。当发送被使能，当主机的 TXFIFO 里有任何数据，序列开始并继续进行。时钟信号被主机连续的提供，直到 TXFIFO 变空，然后停止等待额外的数据。

在只接收模式，即半双工（BIDIMODE=1, BIDIOE=0）或者单工模式（BIDIMODE=0，RXONLY=1），在 SPI 被使能并且只接收模式被激活的时候，主机就立即开始接收。主机一直会提供时钟并连续地接收数据，直到主机停止了 SPI 或者关闭了只接收模式。

当主机以连续的模式（SCK 信号是连续的）进行通讯，主机必须要考虑从机处理数据流的能力。当有必要时，主机必须降低通讯速度，并提供更慢的时钟，或者发送分开的帧。要注意的是，对于主机或者从机来说，没有下溢错误信号，来自于从机的数据被主机处理（即使从机不能及时发送正确的数据）。

每个序列都必须被 NSS 引脚管理，同时在多从机系统中选择要进行通讯的其中一个从机。在一个单从机系统，没有必要使用 NSS 去控制从机，但通常也最好提供 NSS，使通讯的每个数据序列开头同步。NSS 可以由软件和硬件两种方式管理。

当 BSY 被置位，它显示了正在进行的数据帧交互。当专门的帧交互被完成时，RXNE 标志置位。最后一位被采样，并且整个数据帧被存在 RXFIFO 中。

35.2.8.3 关闭 SPI 的流程

全双工或者单传输模式下，主机可以在提供要发送的数据结束后完成交互。在这种情况下，在最后的数据交互后，停止时钟。在这些模式下，SPI 关闭之前，用户必须使用标准的关闭流程。在主机发送时禁止 SPI，如果此时一个帧交互正在进行，或者下一个数据帧存在 TXFIFO 中，SPI 的功能是不能被保证的。

当主机处在任何只接收模式，停止连续时钟的唯一方法是关闭 SPI（SPE=0）。

当 SPI 被关闭，接收到未读走的数据存放在 RXFIFO 中，这些数据必须在下一次 SPI 使能要开始新的序列之前被处理掉。为防止有未读的数据，要确保当 SPI 被关闭时，RXFIFO 是空的。

标准的关闭流程是基于 BSY 状态，并查看 FTLVL，以确保传输彻底完成。也可以通过特定检查来鉴别正在进行交互的通讯，例如：

- 当 NSS 信号被软件管理，主机要向从机提供正确的 NSS 脉冲。
- 当完成来自 FIFO 的交互数据流时，此时最后的数据帧仍在传输过程中。

正确的关闭流程是（只接收模式除外）：

1. 等待 FTLVL= 0（没有数据要发送）
2. 等待 BSY=0（最后的数据被处理完成）
3. 关闭 SPI（SPE=0）
4. 读数据，直到 FRLVL=0（读所有接收到的数据）

对于特定只接收模式，正确的关闭流程是：

1. 在最后一个数据帧传输过程中，通过关闭 SPI（SPE=0），打断接收流程
2. 等待 BSY=0（最后的数据帧已被处理）
3. 读数据，直到 FRLVL=0（读所有接收到的数据）

35.2.8.4 数据包

当帧大小为 8，任何 16 位的读或者写访问时，都会自动使用数据包。在这种情况下，双数据帧会被并行处理。

下图提供了数据包处理过程。在发送方的单个 16 位访问后，两个数据帧被发送。在接收方，如果 RXFIFO 宽度是 16 位，则立即产生 RXNE 事件。作为对 RXNE 事件的响应，接收方通过一个 16 位读 SPI_DR 寄存器，访问了 2 个数据帧。在接收端，RxFIFO 阈值的设定和接下来的读访问必须保持对齐，否则数据会丢失。

在发送端，可以使用 8 位访问方式写奇数序列来完成通讯；在接收端，为了产生 RXNE 事件，在接收最后一个数据帧，接收方必须改变 Rx_FIFO 阈值。

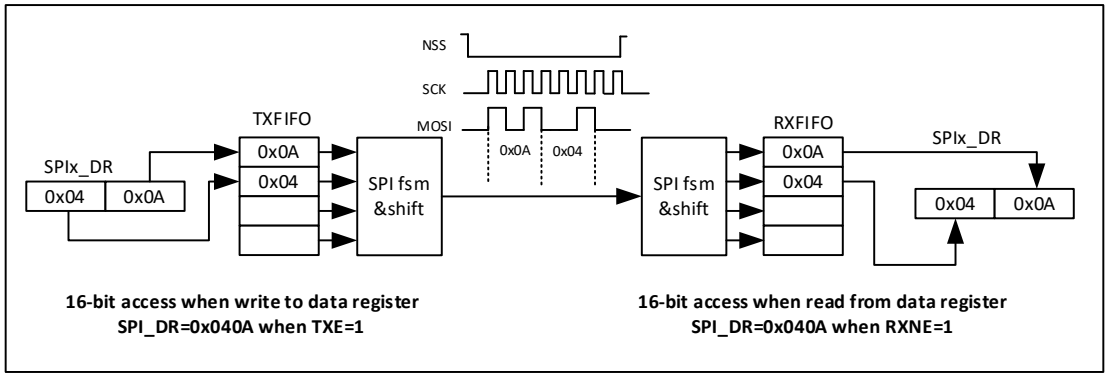


图 35-9 FIFO 中的数据包传输与接收

35.2.8.5 通信时序

本节介绍一些典型的时序，这些时序对于查询、中断都是有效的。为了简化，假定 LSBFIRST=0, CPOL=0, CPHA=1。

1. 当 NSS 有效，SPI 被使能，从机开始控制 MISO。NSS 被释放或者 SPI 关闭时；从机失去对 MISO 的控制。对于从机，在传输开始前必须提供充足的时间给主机，以便提前准备数据。在主机端，仅在 SPI 被使能时，SPI 外设会控制 MOSI 和 SCK 信号（也包括 NSS 信号）。如果 SPI 被关闭，SPI 外设就从 GPIO 断开，在这些线上的电平值取决于 GPIO 的设定。
2. 在主机端，如果通讯是连续的，则 BSY 在帧之间保持有效。在从机端，BSY 信号在数据帧之间通常变低至少一个时钟周期。
3. 只有当 TXFIFO 是满的，TXE 信号才被清零。
4. 在 TXEIE 被置位后，产生 TXE 中断。当 TXE 信号有效时，开始向 TXFIFO 写数据，直到 TXFIFO 变满。

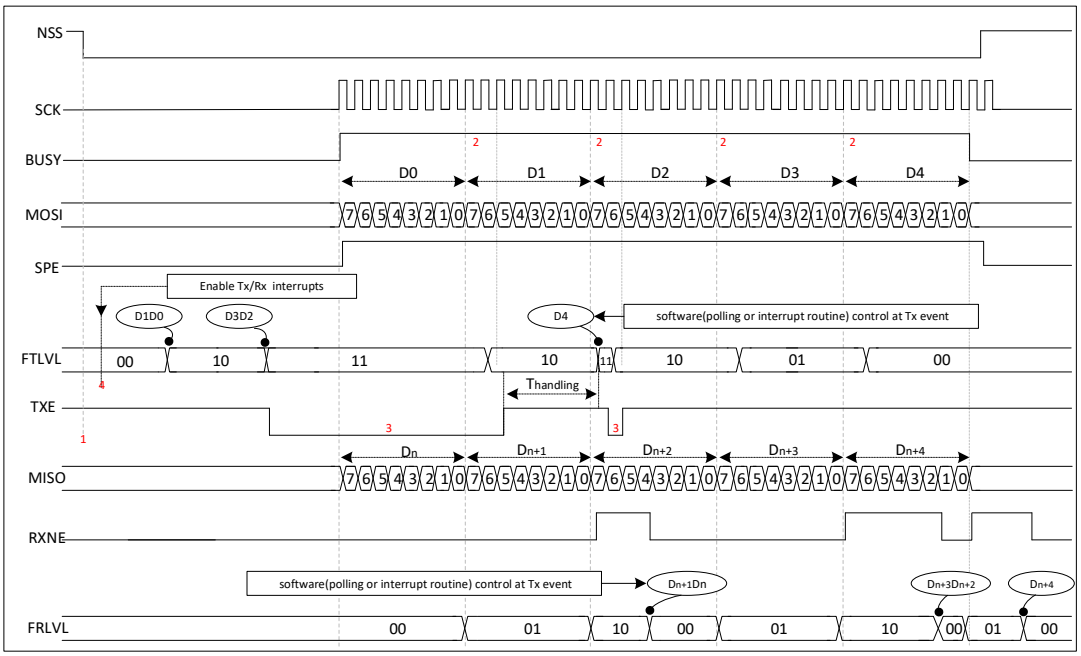


图 35-10 主机 full-duplex communication diagram(bit frame=8)

35.2.9 状态位

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

35.2.9.1 TX FIFO 空标志(TXE)

当 TXFIFO 有足够的空间存放要发送的数据时，TXE 标志位被置位。TXE 标志位与 TXFIFO 水位线有关。该标志位变高并保持高电平，直到 TXFIFO 水位线小于等于 1/2 FIFO 深度才会被硬件清零。如果 TXEIE（SPI_CR2）被置位，则会产生中断请求。

35.2.9.2 接收缓冲非空标志(RXNE)

如果 RXNEIE 位（SPI_CR2）被置位，则产生中断。

35.2.9.3 忙标志(BSY)

BSY 标志由硬件置位与清除(写入此位无效果)，此标志表明 SPI 通信状态。

当它被硬件置位为'1'时，表明 SPI 处于通信状态。

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。

以下情况该标志将被清除为'0'：

- 当 SPI 被正确的关闭掉
- 主机模式，当产生 MODF 标志
- 主机模式，当传输完成，不再有数据要发送
- 从机模式，在每个数据传输之间，BSY 标志置为 0，并保持至少一个 SPI 时钟周期

注：不要使用 BSY 标志处理每个数据发送和接收。使用 TXE 和 RXNE 更合适。

35.2.10 错误标志

35.2.10.1 主模式失效(MODF)

主模式失效（MODF）仅发生在：当 NSS 作为输入信号（SSOE=0），NSS 引脚硬件模式管理下，主设备的 NSS 引脚被拉低；或者在 NSS 引脚软件模式管理下，SSI 位被置为’0’时。此时，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；
- SPE 位被清为’0’。这将停止一切输出，并且关闭 SPI 接口；
- MSTR 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_SR 寄存器的读或写操作；
2. 然后写 SPI_CR1 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，SPE 和 MSTR 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 SPE 和 MSTR 位。

在从器件中，不能将 MODF 位置 1。但是，在多主模式配置中，器件可在 MODF 位置 1 时，处于从模式。在这种情况下，MODF 位指示系统控制可能存在多主模式冲突。可使用中断程序从此状态完全恢复，方法是执行复位或返回到默认状态。

35.2.10.2 溢出错误（OVR）

当数据被主机或者从机接收，并且 RXFIFO 没有足够的空间存接收到的数据时，产生溢出错误。如果软件没有足够的时间读走以前接收到的数据（RXFIFO 中存放），该错误就会发生。

当溢出错误发生，新收到的数据不会覆盖以前存放在 RXFIFO 的数据。接收到的新数据被忽略，并且所有接下来发送的数据丢失。

依次读出 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

35.2.11 SPI 中断

表 35-1 SPI 中断请求

中断事件	事件标志	使能控制位
TXFIFO 空	TXE	TXEIE
RXFIFO 非空	RXNE	RXNEIE
主模式失效事件	MODF	ERRIE
溢出错误	OVR	ERRIE

35.2.12 CRC 计算

为了检查发送和接收数据的可靠性，利用两个独立的 CRC 计算器。SPI 提供独立于帧数据长度的 CRC8 或 CRC16 计算，可固定为 8 位或 16 位。对于所有其他数据帧长度，没有可用的 CRC。

35.2.12.1 CRC 标准

在 SPI 启用（SPE = 1）之前，通过设置 SPI_CR1 寄存器中的 CRCEN 位来启用 CRC 计算。计算在由 SPI_CR1 寄存器中的 CPHA 和 CPOL 位定义的采样时钟边沿进行处理。计算出的 CRC 值在数据块结束时自动检查，并且由 CPU 或 DMA 管理传输。当接收数据内部计算的 CRC 与接收到的 CRC 不匹配时，设置 CRCERR 标志以指示数据错误。处理 CRC 计算的正确程序取决于 SPI 配置和选择的传输管理方式。

35.2.12.2 CPU 管理的 CRC 传输

在接收到最后一个数据帧传输结束之前必须设置 CRCNEXT 位，用于指示在当前处理的数据帧后开始传输 CRC 帧。在 CRC 传输期间停止 CRC 计算。

接收到的 CRC 数据以字节或字的形式存储在 RXFIFO 中。在 CRC 模式下，必须将接收缓冲区视为单个 16 位缓冲区，用于一次仅接收一个数据帧。CRC 格式的传输通常在数据传输结束时需要多出一个数据帧。当设置一个通过 16 位 CRC 校验的 8 位数据帧时，需要多两帧才能发送完整的 CRC 值。

当接收到最后一个 CRC 数据时，将执行自动校验，比较接收到的值和 SPI_RXCRC 寄存器。软件必须检查 SPI_SR 寄存器中的 CRCERR 标志以确定数据传输状态。软件通过向其写入“0”来清除 CRCERR 标志。CRC 接收后，CRC 值存储在 RXFIFO 中，必须读 SPI_DR 寄存器以清除 RXNE 标志

35.2.12.3 DMA 下的 CRC 传输

当使用带 CRC 的 DMA 模式启用 SPI 通信时，通信结束时 CRC 的发送和接收是自动完成的（在仅接收模式下读取 CRC 数据除外），CRCNEXT 位不必由软件处理。SPI 传输 DMA 通道的计数器必须设置为要传输的数据帧数，其中不包括 CRC 帧。在接收端，接收到的 CRC 值在传输结束时由 DMA 自动处理，但用户必须注意从 RXFIFO 中清除接收到的 CRC 信息，因为它总是存放到 RXFIFO 中。在全双工模式下，接收 DMA 通道的计数器可以设置为要接收的包括 CRC 在内的数据帧的数量。

在只接收模式下，DMA 接收通道计数器应该只包含传输的数据量，而不包括 CRC 帧。在数据和 CRC 传输结束时，如果在传输过程中出错，SPI_SR 寄存器中的 CRCERR 标志会被置位。

35.2.13 TI 模式

SPI 接口与 TI 协议兼容。可以使用 SPI_CR2 寄存器的 FRF 位来配置 SPI，以兼容此协议。

时钟极性和相位都被强制为遵循 TI 协议，和 SPI_CR1 中的设置无关。NSS 管理也特定于 TI 协议，在这种情况下，无法通过 SPI_CR1 和 SPI_CR2 寄存器（SSM、SSI 和 SSOE）来对 NSS 管理进行配置。

在从模式下，SPI 波特率预分频器用于控制在当前传输完成时 MISO 引脚切换为高阻态的时刻。可以使用任意波特率，因此可以非常灵活地确定此时刻。波特率通常设置为外部主时钟波特率。MISO 信号变为高阻态的延时（trelease）取决于内部重新同步以及通过 SPI_CR1 寄存器的 BR[2:0] 位设置的波特率值。具体公式如下：

$$\frac{t_{baud_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud_rate}}{2} + 6 \times t_{pclk}$$

如果从器件在数据帧传输期间检测到错位的 NSS 脉冲，TIFRE 标志将置 1。

35.3 I²S 功能描述

35.3.1 简介

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：

MCK：主时钟(独立映射)，在 I²S 配置为主模式，寄存器 SPI_I2SPR 的 MCKOE 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_I2SPR，另一个是 I²S 通用配置寄存器 SPI_I2SCFGR(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)。

通过将寄存器 SPI_I2SCFGR 的 I2SMOD 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。

在 I²S 模式下不使用寄存器 SPI_CR1 和所有的 CRC 寄存器。同样，I²S 模式下也不使用寄存器 SPI_CR2 的 SSOE 位，和寄存器 SPI_SR 的 MODF 位和 CRCERR 位。

I²S 使用与 SPI 相同的寄存器 SPI_DR 用作 16 位模式数据传输。

35.3.2 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_SR 的 CHSIDE 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(CHSIDE 位在 PCM 协议下无意义)。

有四种数据和帧格式组合，可采用下列格式发送数据：

- 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_DR 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_I2SCFGR 的 I2SSTD[1:0]位和 PCMSYNC 位来选择。

35.3.2.1 I²S 飞利浦协议

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。

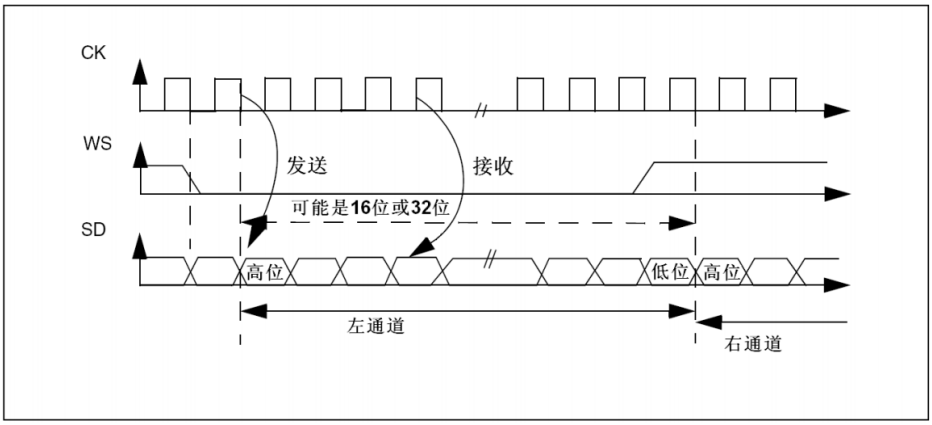


图 35-11 I²S 飞利浦协议波形(16/32 位全精度，CKPOL = 0)

当 CKPOL=0 时，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。当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据。

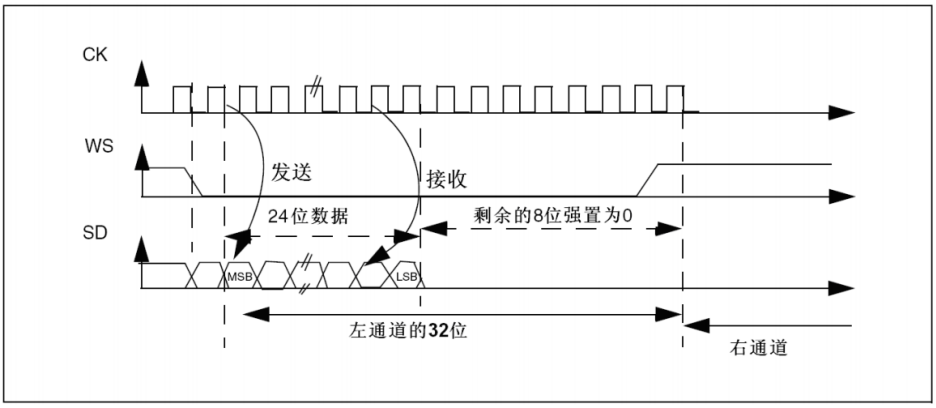


图 35-12 I²S 飞利浦协议标准波形(24 位帧，CKPOL = 0)

此模式需要对寄存器 SPI_DR 进行 2 次读或写操作。

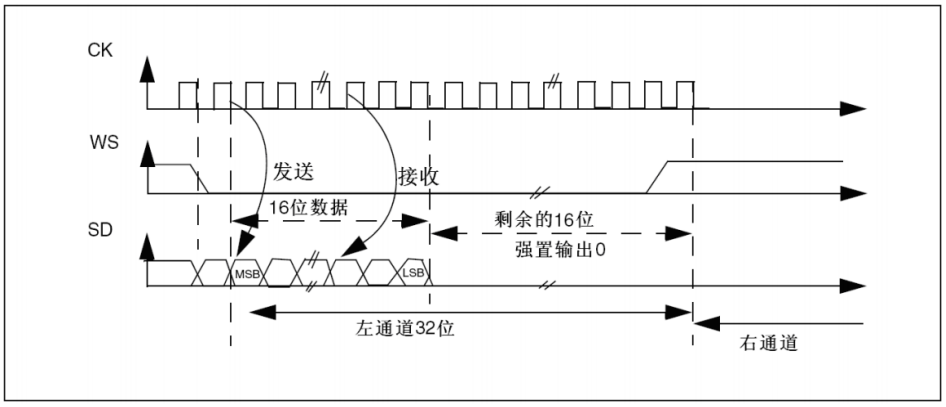


图 35-13 I²S 飞利浦协议标准波形(16 位扩展至 32 位包帧，CKPOL = 0)

当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据

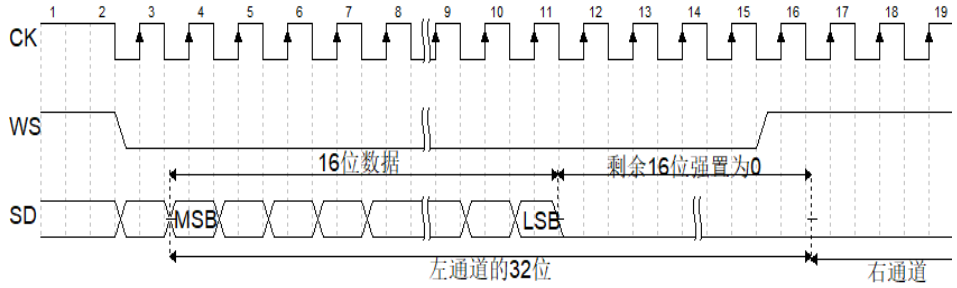


图 35-14 I²S 飞利浦协议标准波形(16 位扩展至 32 位包帧，CKPOL = 1)

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_DR。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 35-15 I²S 数据接收

在发送时需要将 MSB 写入寄存器 SPI_DR；标志位 TXE 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 TXE 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 RXNE 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。

35.3.2.2 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。

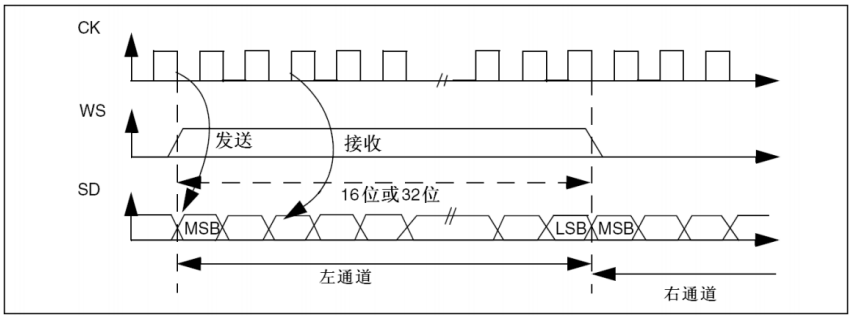


图 35-16 MSB 对齐 16 位或 32 位全精度，CPKOL = 0

当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据

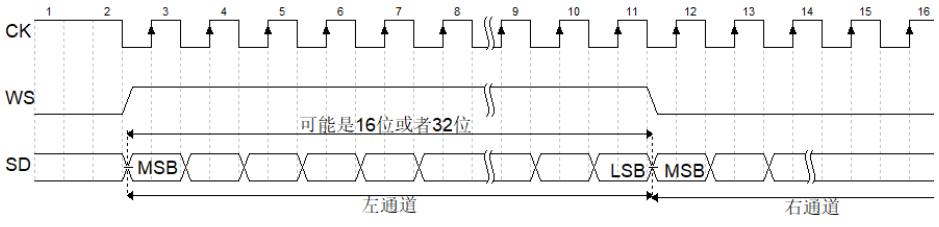


图 35-17 MSB 对齐 16 位或 32 位全精度，CPKOL =1

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。

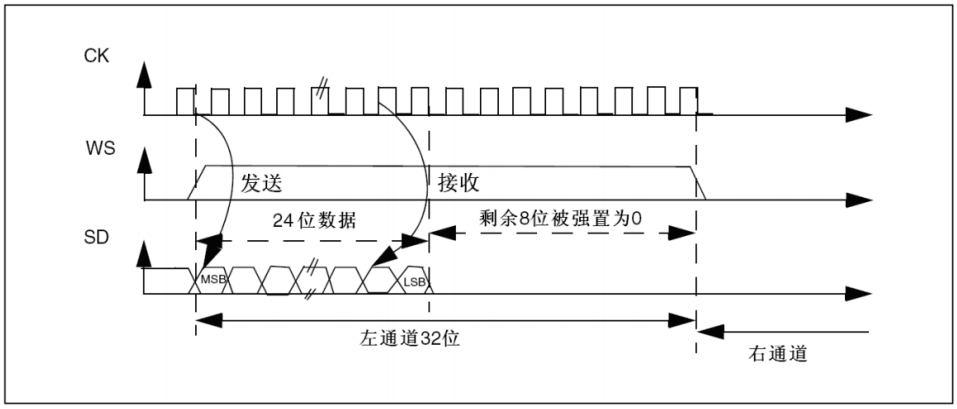


图 35-18 MSB 对齐 24 位数据，CKPOL = 0

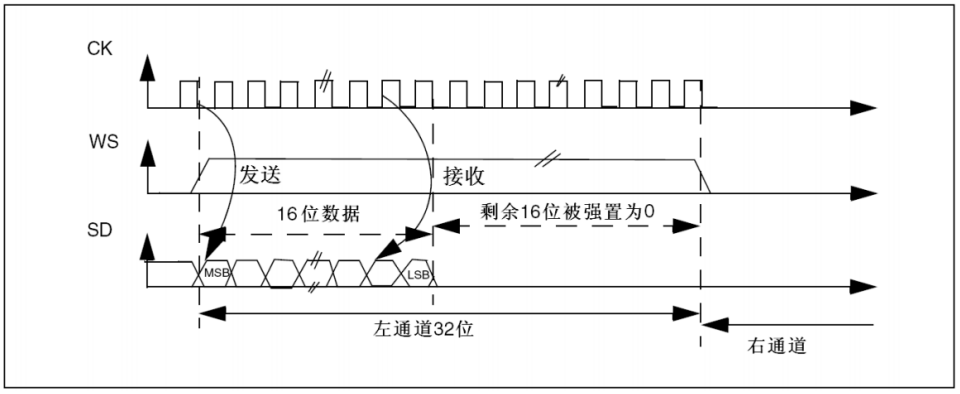


图 35-19 MSB 对齐 16 位数据扩展到 32 位包帧，CPKOL = 0

当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据。一旦有效数据开始从 SD 引脚送出，即发生下一次 TXE 事件。在接收时，一旦接收到有效数据(而不是 0x0000 部分)，即发生 RXNE 事件。

35.3.2.3 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)。当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据

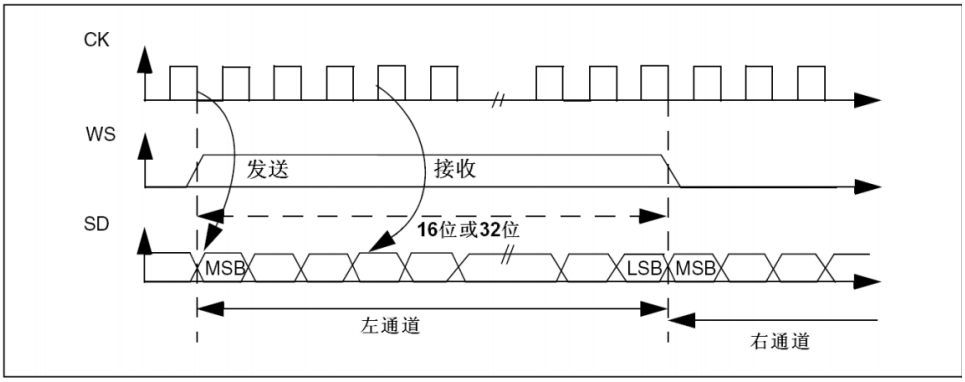


图 35-20 LSB 对齐 16 位或 32 位全精度，CKPOL = 0

- 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_DR 进行 2 次写操作。操作流程如下所示。

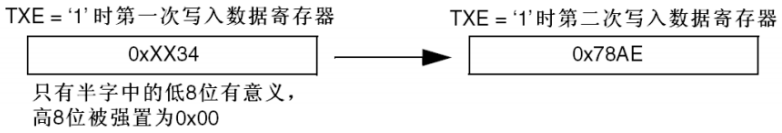


图 35-21 要求读取 0x3478AE 的操作

- 在接收模式下

如果要接收数据 0x3478AE，需要在 2 个连续的 RXNE 事件发生时，分别对寄存器 SPI_DR 进行 1 次读操作。

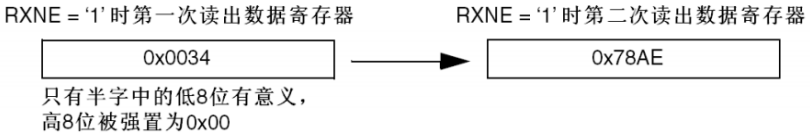


图 35-22 要求接收 0x3478AE 的操作

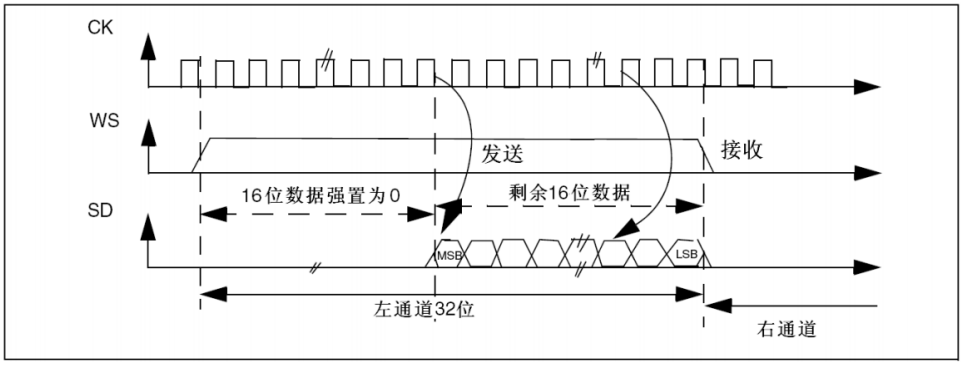


图 35-23 LSB 对齐 16 位数据扩展到 32 位包帧，CPOL = 0

当 CKPOL=1，发送方也在时钟信号(CK)的下降沿改变数据。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_DR。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 0x0000 76A3)，需要的操作：只需要操作一次 SPI_DR。

在发送时，如果 TXE 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 TXE 事件。在接收时，一旦接收到有效数据(而不是 0x0000 部分)，即发生 RXNE 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。

35.3.2.4 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_I2SCFGR 的 PCMSYNC 位来选择。

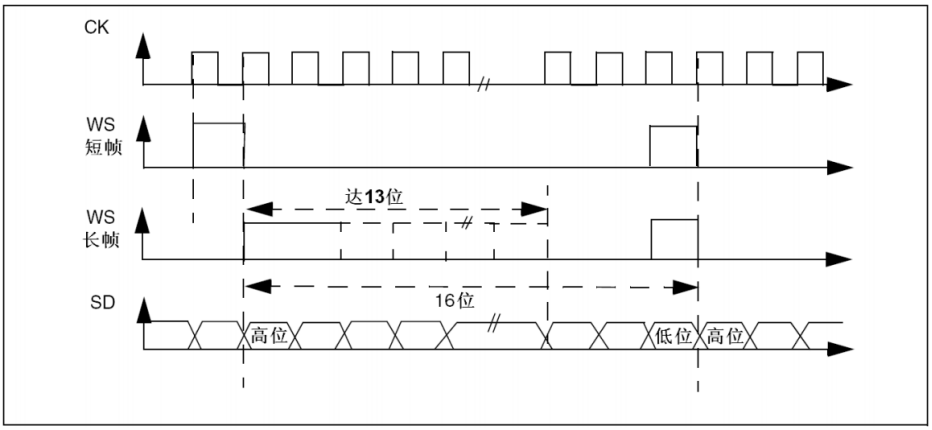


图 35-24 PCM 标准波形(16 位)

当 CKPOL=1 时，发送方也在时钟信号(CK)的上升沿改变数据。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。

当 CKPOL=1 时，发送方也在时钟信号(CK)的上升沿改变数据

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_I2SCFGR 寄存器的 DATLEN 位和 CHLEN 位来确定。

35.3.3 时钟

I^2S 的比特率即确定了在 I^2S 数据线上的数据流和 I^2S 的时钟信号频率。即

I^2S 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，I^2S 比特率计算如下：

I^2S 比特率 = 16×2×fs；

如果包长为 32 位，则有：

I^2S 比特率=32×2×fs

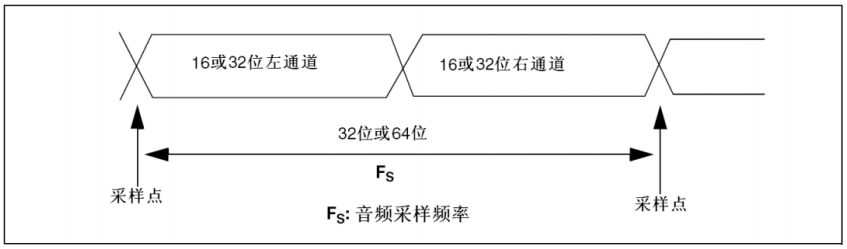


图 35-25 音频采样定义

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。音频的采样频率可以是 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而获得，当需要生成主时钟时(寄存器 SPI_I2SPR 的 MCKOE 位为 ‘1’)：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)*8]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)*4]

当关闭主时钟时(MCKOE 位为 ‘0’)：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)]

使用标准的 8MHz HSE 时钟得到精确的音频频率，见 EXCEL 表。

35.3.4 发送与接收

35.3.4.1 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_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟(MCK)。

1) 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 TXE 置‘1’，这时，要把对应右声道的数据写入发送缓存。标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。标志位 CHSIDE 的值在 TXE 为‘1’时更新，因此它在 TXE 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 TXE 置为‘1’，如果寄存器 SPI_CR2 的 TXEIE 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 5.2 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_DR 写入下一个要传输的数据。建议在要关闭 I²S 功能时，等待标志位 TXE=1 及 BSY=0，再将 I2SE 位清‘0’。

2) 接收流程

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 RXNE 置‘1’，如果寄存器 SPI_CR2 的 RXNEIE 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_DR 进行读操作即可清除 RXNE 标志位。每次接收以后即更新 CHSIDE。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。

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_CR2 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。

注：在传输期间 BSY 标志始终为低。

35.3.4.2 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。

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 TXE 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 CHSIDE 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，CHSIDE 取决于来自外部主 I²S 的 WS 信号。这意味着从 I²S 在接收到主机生成的时钟信号之前，就要准备好第一个要发送的数据。WS 信号为‘1’表示先发送左声道。

注：设置 I2SE 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 TXE 置‘1’，如果寄存器 SPI_CR2 的 TXEIE 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 TXE 为‘1’。

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_DR 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_DR，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_CR2 的 ERRIE 位为‘1’，在寄存器 SPI_SR 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。

建议在清除 I2SE 位关闭 I²S 之前，先等待 TXE=1 并且 BSY=0。

2) 接收流程

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 RXNE 置‘1’，如果寄存器 SPI_CR2 的 RXNEIE 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_DR 读出)以后即更新 CHSIDE，它对应 I²S 单元产生的 WS 信号。读取 SPI_DR 寄存器，将清除 RXNE 位。

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_CR2 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 RXNE=1 时将 I2SE 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。

35.3.5 标志位

35.3.5.1 状态标志

有 3 个状态标志位供用户监控 I²S 总线的状态。

1) 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(I2SCFG=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。

2) 发送缓存空标志位(TXE)

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(I2SE 位为‘0’)，该标志位也为‘0’。

3) 接收缓存非空标志位(RXNE)

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_DR 时，该位清‘0’。

4) 声道标志位(CHSIDE)

在发送模式下，该标志位在 TXE 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_DR 接收到数据时刷新，指示接收到的数据所在的声道。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_SR 的标志位 OVR 或 UDR 为‘1’，且寄存器 SPI_CR2 的 ERRIE 位为‘1’，则会产生中断，而后可以通过读寄存器 SPI_SR 来清除中断标志。

35.3.5.2 错误标志

I²S 单元有 2 个错误标志位。

1) 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_DR 寄存器，该标志位会被置‘1’。在寄存器 SPI_I2SCFGR 的 I2SMOD 位置‘1’后，该标志位才有效。如果寄存器 SPI_CR2 的 ERRIE 位为‘1’，就会产生中断。通过对寄存器 SPI_SR 进行读操作来清除该标志位。

2) 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_CR2 的 ERRIE 位为‘1’，则产生中断指示发生了错误。这时，接收缓存内容，不会刷新为从发送设备送来的新数据。对寄存器 SPI_DR 的读操作返回最后一个正确接收的数据。其他所有上溢发生后由发送设备发出的 16 位数据都会丢失。先读寄存器 SPI_SR 再读寄存器 SPI_DR，来清除该标志位。

35.3.6 DMA 功能

除了 CRC 功能外，同 SPI。因为在 I²S 模式下没有数据传输保护功能。

35.3.7 中断

表 35-2 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	TXE	TXEIE
接收缓冲器非空标志位	RXNE	RXNEIE
下溢标志位	OVR	ERRIE
上溢标志位	UDR	

35.4 寄存器描述

SPI1 寄存器基地址：0x4001 3000

SPI2 寄存器基地址：0x4000 3800

SPI3 寄存器基地址：0x4000 3C00

35.4.1 SPI_CR1 寄存器

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	双向数据模式使能（Bidirectional data mode enable) 0：选择“双线双向”模式； 1：选择“单线双向”模式。 注：I^2S 模式下不使用。
14	BIDIOE	RW	0	双向模式下的输出使能（Output enable in bidirectional mode) 和 BIDIMODE 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：I^2S 模式下不使用。
13	CRCEN	RW	0	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(SPE=0)，才能写该位，否则出错。 该位只能在全双工模式下使用。 注：I^2S 模式下不使用。
12	CRCNEXT	RW	0	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_DR 寄存器写入最后一个数据后应马上设置该位。 注：I^2S 模式下不使用。
11	DFF	RW	0	数据帧格式 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(SPE=0)时，才能写该位，否则出错。 注：I^2S 模式下不使用。
10	RXONLY	RW	0	只接收（Receive only) 该位和 BIDIMODE 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。
9	SSM	RW	0	软件从设备管理（Software slave management) 当 SSM 被置位时，NSS 引脚上的电平由 SSI 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。
8	SSI	RW	0	内部从设备选择（Internal slave select) 该位只在 SSM 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。
7	LSBFIRST	RW	0	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，若软件在通信进行时修改通信将发生错误。 注：I²S 模式下不使用。
6	SPE	RW	0	SPI 使能（SPI enable) 0：禁止 SPI 设备； 1：开启 SPI 设备。 注：I²S 模式下不使用。
5:3	BR[2:0]	RW	0	波特率控制（Baud rate control) 000：fPCLK/2 001：fPCLK/4 010：fPCLK/8 011：fPCLK/16 100：fPCLK/32 101：fPCLK/64 110：fPCLK/128 111：fPCLK/256 注: 当通信在进行时不能改变该位的值，若软件在通信进行时修改通信将发生错误。 注意：I²S 模式下不使用。
2	MSTR	RW	0	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：I²S 模式下不使用。
1	CPOL	RW	0	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: 当通信在进行时不能改变该位的值，若软件在通信进行时修改通信将发生错误。 注：I²S 模式下不使用。
0	CPHA	RW	0	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: 当通信在进行时不能改变该位的值，若软件在通信进行时修改通信将发生错误。 注：I²S 模式下不使用。

35.4.2 SPI_CR2 寄存器

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRF	Res		FRXTH	Res				TXEIE	RXNEIE	ERRIE	CLRTXFIFO	Res	SSOE	TXDMAEN	RXDMAEN
RW	-		RW	RW	RW	-		RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	FRF	RW	0	帧格式: 0：SPI Motorola 模式 1：SPI Ti 模式 只有在禁止 SPI(SPE=0)后才能对此位进行操作
14:13	保留	-	-	-
12	FRXTH	RW	0	FIFO 接收阈值 该位用来设定触发 RXNE 事件的 RXFIFO 阈值。 1：如果 FIFO level 大于或者等于 1/2，产生 RXNE 0：FIFO 中有数据
11:8	保留	-	-	-
7	TXEIE	RW	0	发送缓冲区空中断使能（Tx buffer empty interrupt enable) 0：禁止 TXE 中断; 1：允许 TXE 中断，当 TXE 标志置位为‘1’时产生中断请求。
6	RXNEIE	RW	0	接收缓冲区非空中断使能（RX buffer not empty interrupt enable) 0：禁止 RXNE 中断; 1：允许 RXNE 中断，当 RXNE 标志置位时产生中断请求。
5	ERRIE	RW	0	错误中断使能（Error interrupt enable) 当错误(CRCERR、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。
4	CLRTXFIFO	RW	0	清空 TXFIFO 软件置位，硬件复位 0：没作用 1：清空 TXFIFO 注：只有当 SPI 禁止(SPE=0)时，才能写该位，否则无效。
3	保留	-	-	-
2	SSOE	RW	0	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：I2S 模式下不使用。
1	TXDMAEN	RW	0	发送缓冲区 DMA 使能（Tx buffer DMA enable) 当该位被设置时，TXE 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。

0	RXDMAEN	RW	0	接收缓冲区 DMA 使能（Rx buffer DMA enable) 当该位被设置时，RXNE 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。

35.4.3 SPI_SR 寄存器

Address offset:0x08

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			FTLVL		FRLVL		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
-			R		R		R	R	R	R	RC_W0	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15:13	保留	-	-	-
12:11	FTLVL	R	0	FIFO 发送 level 这些位由硬件设置和清除。 00：FIFO 空 01：1/4 FIFO 10：1/2 FIFO 11：FIFO 已满（FIFO 阈值大于 1/2 时视为 FULL） 注意：该位在 I2S 模式下不使用。
10:9	FRLVL	R	0	FIFO 接收 level 这些位由硬件设置和清除。 00：FIFO 空 01：1/4 FIFO 10：1/2 FIFO 11：FIFO 已满 注意：这些位在 I²S 模式和 带 CRC 校验的 SPI 仅接收模式下不使用。
8	FRE	R	0	FRE：帧格式错误（Frame format error) 该标志在 TI 从模式下用于 SPI 此标志由硬件置 1，在读取 SPIx_SR 时由软件复位。 0：未发生帧格式错误 1：发生帧格式错误
7	BSY	R	0	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。
6	OVR	R	0	溢出标志（Overrun 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。

5	MODF	RW	0	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。 注：I2S 模式下不使用。
4	CRCERR	RCW0	0	CRC 错误标志（CRC error flag) 0：收到的 CRC 值和 SPI_RXCRCR 寄存器中的值匹配； 1：收到的 CRC 值和 SPI_RXCRCR 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：I2S 模式下不使用。
3	UDR	R	0	下溢标志位（Underrun 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’。 注：在 SPI 模式下不使用。
2	CHSIDE	R	0	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。
1	TXE	R	1	发送缓冲为空（Transmit buffer empty) 0：发送缓冲非空； 1：发送缓冲为空。
0	RXNE	R	0	接收缓冲非空（Receive buffer not empty) 0：接收缓冲为空； 1：接收缓冲非空。

35.4.4 SPI_DR 寄存器

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DR[15:0]	RW	0	数据寄存器（Data register) 用作 Rx 和 Tx FIFO 之间的接口。读取数据寄存器时，访问 RxFIFO，而写入数据寄存器访问 TxFIFO 注意：数据始终是右对齐的。写入寄存器时忽略未使用的位，读取寄存器时读取为零。Rx 阈值设置必须始终与当前使用的读取访问相对应。

35.4.5 SPI_CRCPR 寄存器

Address offset:0x10

Reset value:0x0007

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

Bit	Name	R/W	Reset Value	Function
15:0	CRCPOLY[15:0]	RW	0	CRC 多项式寄存器（CRC polynomial register) 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 I^2S 模式下不使用。 注：多项式值只能是奇数，不支持偶数值。

35.4.6 SPI_RXCRCR 寄存器

Address offset:0x14

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	RxCRC[15:0]	R	0	接收 CRC 寄存器 在启用 CRC 计算时，RXCRC[15:0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CR1 的 CRCEN 位写入‘1’时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 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 I^2S 模式下不使用。

35.4.7 SPI_TXCRCR 寄存器

Address offset:0x18

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	TxCRC[15:0]	R	0	发送 CRC 寄存器 在启用 CRC 计算时，TXCRC[15:0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPI_CR1 中的 CRCEN 位写入‘1’时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 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。

35.4.8 SPI_I2S_CFGR 寄存器

Address offset:0x1c

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
-				RW	RW	RW		RW	-	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
15:12	保留	-	-	-
11	I2SMOD	RW	0	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。
10	I2SE	RW	0	I²S 使能（I²S enable) 0:关闭 I2S； 1：I2S 使能。 注：在 SPI 模式下不使用。
9:8	I2SCFG	RW	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。
7	PCMSYNC	RW	0	PCM 帧同步（PCM frame synchronization) 0：短帧同步； 1：长帧同步。 注：该位只在 I2SSTD = 11（使用 PCM 标准)时有意义。 在 SPI 模式下不使用。
6	保留	-	-	-
5:4	I2SSTD	RW	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。 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。
3	CKPOL	RW	0	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。
2:1	DATLEN	RW	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。
0	CHLEN	RW	0	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 DATLEN = 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。

35.4.9 SPI_I2SPR 寄存器

Address offset:0x20

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						MCKOE	ODD	I2SDIV							
-						RW	RW	RW							

Bit	Name	R/W	Reset Value	Function
15:10	保留	-	0	-
9	MCKOE	RW	0	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。
8	ODD	RW	0	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。 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。
7:0	I2SDIV	RW	0	I²S 线性预分频（I²S linear prescaler) 禁止设置 I2SDIV [7:0] = 0 或者 I2SDIV [7:0] = 1

				注：为了正确操作，该位只有在关闭了 I^2S 时才能设置。仅在 I^2S 主设备模式下使用该位。 在 SPI 模式下不使用。

36. I²C 接口

36.1 简介

I²C(Inter-integrated circuit)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 总线特定的顺序、协议、仲裁和时序。支持标准（Sm）、快速（Fm）、快速增强模式（Fm+）。根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

36.2 主要特征

- 从机和主机模式
- 多主机功能：可以做主机，也可以做从机
- 作为主机
  - 产生时钟
  - 产生起始和停止信号
- 作为从机
  - 可编程的 I²C 地址检测
  - 可响应 2 个从地址的双地址能力(其中 1 个可配置掩码)
  - 检测停止位
- 支持 7/10 位寻址模式
- 支持广播呼叫功能
- 支持不同的通讯速度
  - 标准(高至 100 KHz)
  - 快速(高至 400 KHz)
  - 快速增强(高至 1 MHz)
- 状态标志位：
  - 发送/接收模式标志
  - 字节传输完成标志
  - I²C 总线忙标志
- 错误标志位
  - 主机模式仲裁丢失
  - 地址/数据传输后的应答(ACK)失败
  - 起始/停止条件错误
  - 禁止拉长时钟功能时的上溢或下溢
- 2 个中断向量
  - 1 个事件中断，用于地址/数据通讯成功
  - 1 个错误中断
- 可选的拉长时钟功能
- 支持 DMA 功能
- 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

- 支持 SMBus 2.0
  - 25 ms 时钟低电平超时延时
  - 10 ms 主设备累积时钟低电平扩展时间
  - 25 ms 从设备累积时钟低电平扩展时间
  - 带 ACK 控制的硬件 PEC 产生/校验
  - 支持地址分辨协议（ARP）
  - 超时和空闲条件检测
- 兼容 SMBus
- 支持地址匹配时从 Stop 模式唤醒
  - 唤醒过程中支持可配置的 timeout 时间

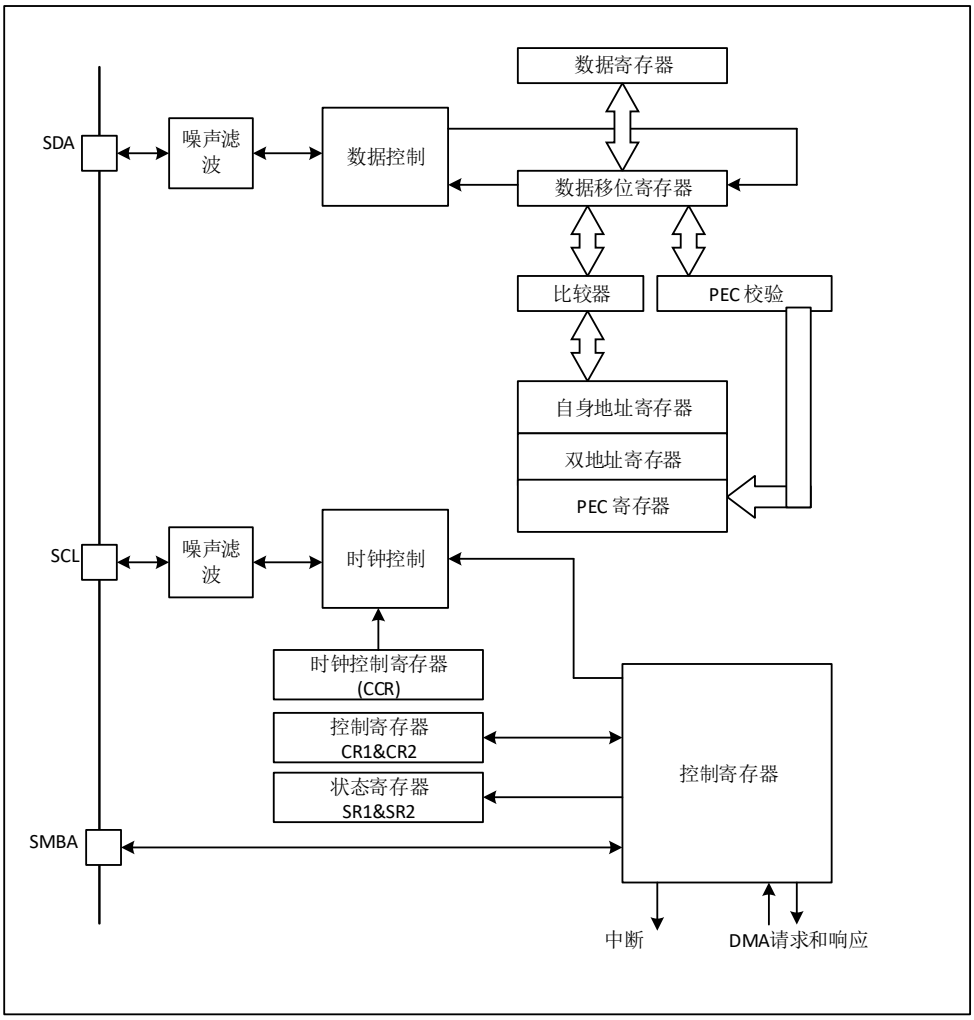


图 36-1 I²C 模块

36.3 I²C 功能描述

36.3.1 简介

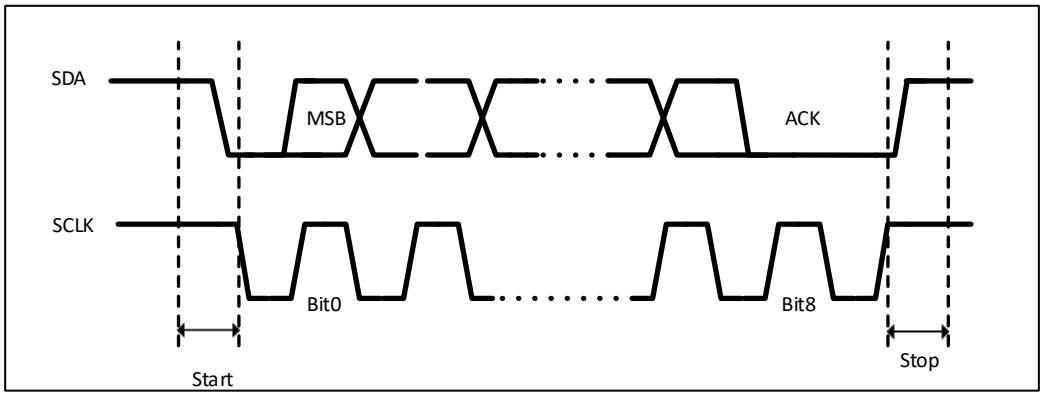


图 36-2 I²C 总线协议

I2C 支持以下四种模式：

- 从发送器模式 （Slave transmitter）
- 从接收器模式 （Slave receiver）
- 主发送器模式（Master transmitter）
- 主接收器模式（Master receiver）

默认情况下，I²C 接口总是工作在从模式。接口在生成起始条件后，自动地从从模式到主模式；在允许多主机功能的情况下，当仲裁丢失或产生停止信号时，则从主模式切换到从模式

作为主机，I²C 接口启动数据传输，并产生时钟信号。串行数据的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机模式下由软件控制产生。

作为从机，I²C 接口能识别自己的地址(7 位或 10 位)和广播呼叫地址。软件能够控制开启或禁止对广播呼叫地址的识别。

数据和地址按 8 位（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 master 模式发送。

在一个字节传输后的第一个始终时钟内，接收方必须回送一个应答位(ACK)给发送方。

36.3.2 时钟

I²C 模块的输入时钟是 APB 时钟。模块内部生成 SCL 时钟再输出到总线上给从机。

SCL 产生机制如下：有一个计数器根据 CCR 寄存器的值来计数高低电平（CCR 寄存器决定高低电平的长度）。此外，CCR 寄存器还控制标准模式，快速模式和快速增强模式的选择，以及 SCL 高低电平的占空比。

36.3.3 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，这个计算器使用一个可编程的多项式对每一位串行数据进行计算。

● PEC 计算由 I2C_CR1 寄存器的 ENPEC 位使能。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 TxE 事件时设置 I2C_CR1 寄存器中的 PEC 位，PEC 将在这个字节后被发送。

– 在接收时：在最后一个 RxNE 事件时设置 I2C_CR1 寄存器中的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器会发送一个 NACK。如果是主接收器，不管校对的结果如何，PEC 后都将发送 NACK。PEC 位必须在接收当前字节的 ACK 脉冲之前设置。

- 在 I2C_SR1 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(对应 DMA 传输字节数）信号时，它在最后一个字节后自动发送 PEC。

– 在接收时：当 I2C 接口从 DMA 处接收到一个 EOT_1(对应 DMA 传输字节数-1）信号时，它将自动把下一个字节作为 PEC，并且将检查它。在接收到 PEC 后产生一个 DMA 请求。

- 为了允许中间 PEC 传输，在 I2C_CR2 寄存器中有一个控制位(LAST 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。

36.3.4 I²C 的从模式

默认情况下，I²C 接口总是工作在 slave 模式。从 slave 模式切换到 master 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_CR2 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：4MHz

快速模式下为：8MHz

快速增强模式下为：16MHz

设置从机地址及地址屏蔽:

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，并与芯片的地址 OAR1 或者广播呼叫地址(如果 ENGC=1)相比较。如果启用了额外地址（ENDUAL=1）,地址会和芯片的 OAR2 地址进行比较，并且 OAR2 寄存器中的地址可以通过 OA2MSK[2:0]进行屏蔽，屏蔽从 OA2 高第 7 位开始，当 OA2MSK 配置为 1 到 6 时，将分别只有 OA2[7:2]、OA2[7:3]、OA2[7:4]、OA2[7:5]、OA2[7:6] 或 OA2[7] 与接收到的地址作比较。如果 OA2MSK=7，接收到的所有 7 位地址（保留地址除外）均得到应答。

地址不匹配（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。

地址匹配：I2C 接口产生以下时序：

- 如果 ACK 被软件置'1'，则产生一个应答脉冲
- 硬件置位 ADDR 位，如果设置了 ITEVTEN 位，则产生中断
- 如果 ENDUAL=1，软件必须读 DUALF 位，以确认响应了哪个从地址

在从模式下 TRA 位指示当前是处于接收器模式还是发送器模式

36.3.4.1 从机发送模式

在接收到地址并清除 ADDR 位后，（如果地址字节的最低位是 1）Slave 将数据（字节）从 DR 寄存器，经由内部移位寄存器发送到 SDA 上。

Slave 拉低 SCL，直到 ADDR 位被清除，并且待发送数据已写入 DR 寄存器（参考 EV1、EV3）。

当收到应答脉冲时：TxE 位被硬件置位，如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 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_DR 寄存器，则 BTF 位被置位。Slave 拉低 SCL，直到 BTF 位被软件清零（读 I2C_SR1 之后，再写入 I2C_DR 寄存器）。

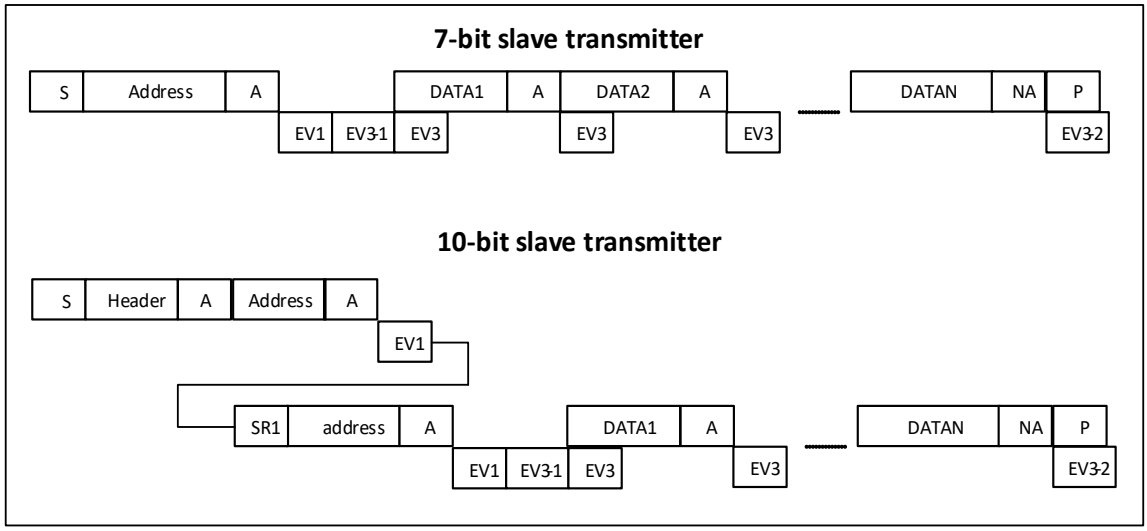


图 36-3 从发送器的传送序列图

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV1：ADDR=1，通过先读 SR1 寄存器，再读 SR2 寄存器，清零 ADDR 位

EV3-1：TxE=1，移位寄存器为空，数据寄存器为空，向 DR 寄存器写 Data1

EV3：TxE=1，移位寄存器不为空，数据寄存器为空，向 DR 寄存器写（Data2）清零 TxE

EV3-2：AF=1；软件向 AF 位写 0 清零该位

注:

1 ）EV1 和 EV3-1 事件拉低 SCL，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

36.3.4.2 从机接收模式

在接收到地址并清除 ADDR 后，（如果地址字节的最低位是 0）slave 将通过内部移位寄存器把从 SDA 线接收到的字节存进 DR 寄存器。I^2C 接口在接收到每个字节后都执行下列操作：

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 RxNE 被置位，并且在接收新的数据结束之前，DR 寄存器未被读出，则 BTF 位被置位，在清除 BTF（读出 I2C_SR1 之后再读 I2C_DR 寄存器）之前，slave 一直拉低 SCL。(见下图)。

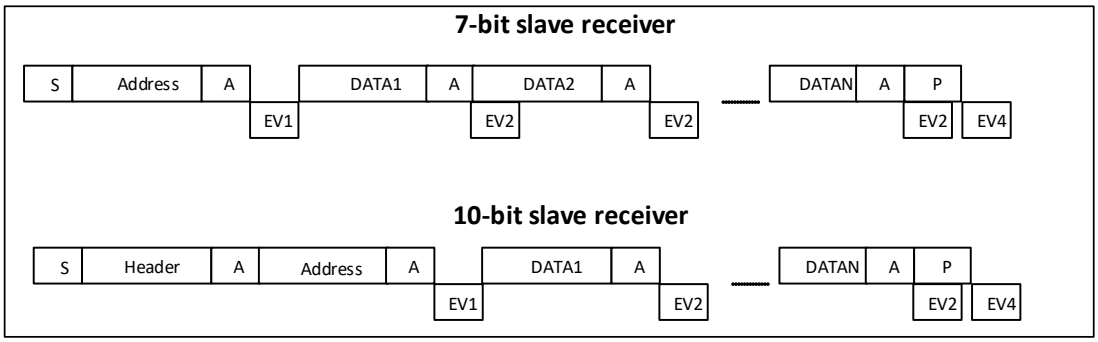


图 36-4 从接收器的传送序列图

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV1：ADDR=1，通过先读 SR1，后读 SR2 实现 ADDR 的清零

EV2：RxNE=1，读 DR 寄存器清零该位

EV4：STOPF=1，通过先读 SR1 寄存器，后写 CR1 寄存器实现对该位的清零。

注：

1） EV1 事件拉低 SCL，直到相应软件序列结束。

2） EV2 软件序列必须在当前字节传输完成之前完成。

3） 当用户检查 SR1 寄存器内容后，应该对每个置位的标志位，进行完整的清除序列。比如 ADDR 和 STOPF 标志位，需要用以下序列：

如果 ADDR=1，先读 SR1，再读 SR2；如果 STOPF=1，先读 SR1，再写 CR1。

这样做的目的是确保如果 ADDR 和 STOPF 都被置位，都能被发现并且清除掉。

36.3.4.3 关闭通信

在传输完最后一个数据字节后，主机产生一个停止条件，从机检测到该条件时：

- 硬件置位 STOPF，如果设置了 ITEVTEN 位，则产生一个中断。

通过先读 SR1，后写 CR1，实现对 STOPF 位的清零。

36.3.5 I²C 的主模式

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 START 位在总线上产生了起始条件，设备就进入了主机模式。

以下是主机模式所要求的操作顺序：

- 在 I2C_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置 I2C_CCR 寄存器
- 配置 I2C_TRISE 寄存器
- 配置 I2C_CR1 寄存器中的 PE 启动外设
- 置 I2C_CR1 寄存器中的 START 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：4MHz

- 快速模式下为：8MHz
- 快速增强模式下为：16MHz

36.3.5.1 主机产生时钟

CCR 寄存器以输入时钟上升沿计数，产生 SCL 的高低电平。由于从机可能拉长 SCL 信号，在 SCL 上升沿产生后，主机在 I2C_TRISE 寄存器所设置的时间到达时，检测来自总线的 SCL 信号。

— 如果 SCL 是低电平，意味着从机正在拉长 SCL 总线，高电平计数器停止计数，直到 SCL 被检测到高电平。这是为了确保 SCL 参数的最小高电平时间。

— 如果 SCL 是高电平，高电平计数器保持计数。

实际上，即使从机不拉长 SCL，从 SCL 上升沿产生，到 SCL 上升沿被发现，这样的反馈回路也是要花费些时间的。这个回路的时间与 SCL 的上升时间有关系，再加上 SCL 输入路径的模拟噪声滤波延时，以及芯片内部由于用 APB 时钟进行的 SCL 同步时间。反馈回路的最大时间在 TRISE 寄存器中设置，所以无论 SCL 上升时间如何，SCL 的频率保持稳定。

36.3.5.2 起始条件

当 BUSY=0 时，设置 START=1，I2C 接口将产生一个起始条件，并切换至主机模式(MSL 被置位)。

注：在主机模式下设置 START 位，将在当前字节传输完成后，由硬件产生一个重开始条件。

一旦发出起始条件：

● SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。

● 主机读 I2C_SR1 寄存器，再把从机地址写入 I2C_DR 寄存器。

36.3.5.3 从机地址发送

主机将从机的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：

– ADD10 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

– 然后主机读 SR1 寄存器，并且将第二个地址字节写入 DR 寄存器。

– ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

– 随后主机读 SR1 寄存器，然后读 SR2 寄存器。

● 在 7 位地址模式时，将送出一个地址字节。

一旦该地址字节被送出，

– ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。

– 随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器。

根据送出从机地址的最低位，主设备决定进入是发送器模式，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让最低位等于 0。

– 要进入接收器模式，主设备发送从地址时让最低位等于 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)，然后送最低位等于 0 的从地址。（头段字节中的 xx 是 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)，然后送最低位等于 1 的从地址。然后再重新发送一个起始条件，后面跟着头字节(11110xx1)。（头字节中的 xx 是 10 位地址中的最高 2 位）。

TRA 位指示主设备是在接收器模式还是发送器模式

36.3.5.4 主机发送模式

在发送了地址和清除了 ADDR 位后,主机通过内部移位寄存器将数据字节从 DR 寄存器发送到 SDA 线上。

主机等待，直到第一个数据字节被写入 DR 寄存器（参见 EV8_1）。

当收到 ACK 脉冲时，TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 被置位，且在上一次数据发送结束之前，没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位。在清除 BTF（读 I2C_SR1 之后，再写 I2C_DR 寄存器）之前，I2C 接口将保持 SCL 为低电平。

关闭通信

在 DR 寄存器中写入最后一个字节后，通过设置 STOP 位产生一个停止条件(见图的 EV8_2)，然后 I2C 接口将自动回到从模式(MSL 位清除)。

注：当 TxE 或 BTF 位置位时，在出现 EV8_2 事件时产生停止条件。

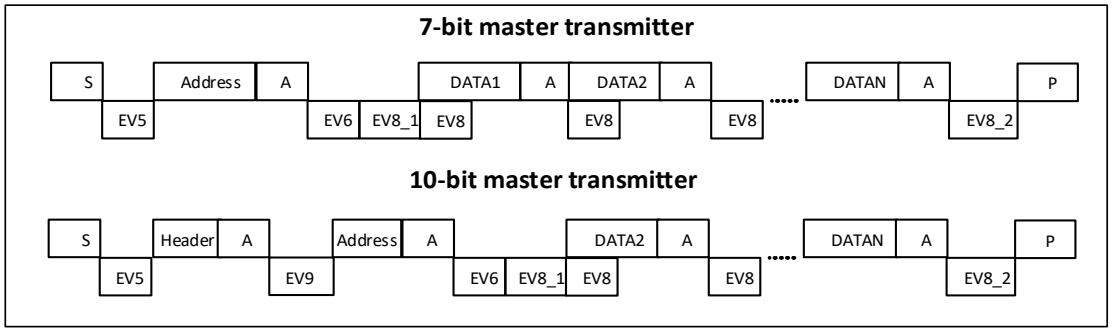


图 36-5 主发送器传送序列图

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV5：SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件

EV6：ADDR=1,读 SR1，再读 SR2，清除该事件

EV8_1：TXE=1，移位寄存器空

EV8：TXE=1，写 DR 寄存器清除该事件

EV8_2：TXE=1，BTF=1，产生停止条件时由硬件清除

EV9： ADDR10=1,读 SR1 然后写 DR 寄存器清除该事件

注：

1．EV5, EV6, EV8_1 和 EV8_2 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束

2．EV8 软件序列必须在当前字节发送完成前执行完毕。若 EV8 的软件序列不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 TxE

36.3.5.5 主机接收模式

在发送地址和清除 ADDR 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 DR 寄存器。在接收到每个字节后，I²C 接口依次执行以下操作：

- 如果 ACK 位被置位，则发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断。

如果 RxNE 位被置位，并且在接收新数据结束前，DR 寄存器中的数据没有被读走，硬件将设置 BTF=1，在清除 BTF 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再读出 I2C_DR 寄存器将清除 BTF 位。

关闭通信

方法 1：该方法的应用场景是，当 I²C 的中断被设成应用程序中最高优先级

主机在接收到从机最后一个字节后，发送一个 NACK。接收到 NACK 后，从机释放对 SCL 和 SDA 线的控制。主机就可以发送一个停止/重开始条件。

1) 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 RxNE 事件之后)必须清除 ACK 位。
2) 为了产生一个停止/重开始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置停止/开始位。
3) 当接收单个字节时，清除 ACK 和停止条件的产生位要刚好在 EV6 之后(EV6_1 时，清除 ADDR 之后)。在产生了停止条件后，I2C 接口自动回到从模式(MSL 位被清除)。

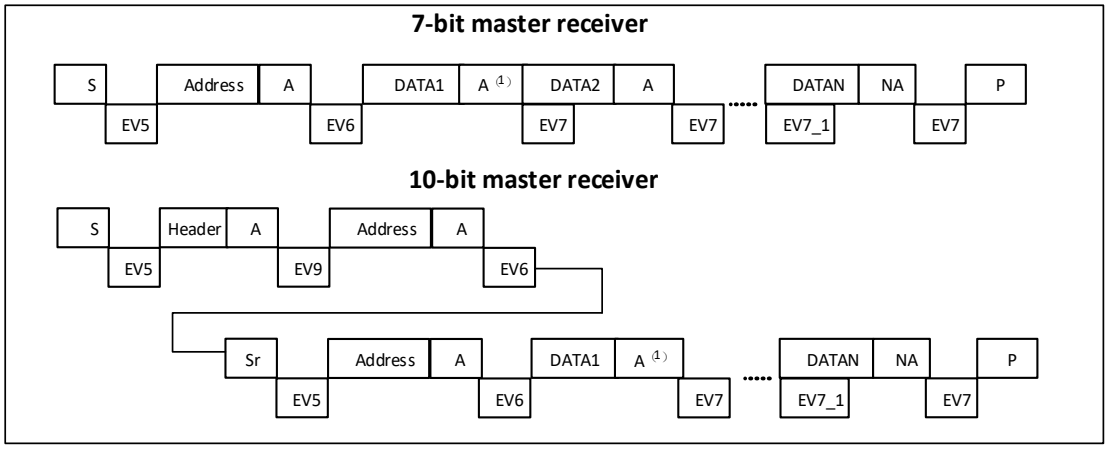


图 36-6 方法 1：主模式发送时的时序

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV5：SB=1，读 SR1，再写 DR 寄存器，该位被清零

EV6：ADDR=1，读 SR1，再读 SR2，该位被清零

EV6_1：无相关的标志事件，仅用作 1 个字节的接收。

EV7：RxNE=1，读 DR 寄存器，该位被清零

EV7_1：RxNE=1，读 DR 寄存器，写 ACK=0 并置位 STOP

EV9： ADD10=1,读 SR1 然后写 DR 寄存器清除该事件

注:

1) 如果是单个字节接收，则上述标注为（1）的地方会是 NA
2) EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束
3) EV7 软件序列必须在当前字节发送完成前执行完毕。在 EV7，软件序列不能在当前传输的字节传输完成前执行完毕，则推荐使用 BTF 代替 RXNE。
4) EV6_1 或者 EV7_1 的软件序列必须在当前字节传输的 ACK 位发送之前完成。

方法 2：这个方法的应用场景是，I2C 的中断在应用中不是最高优先级，或者使用查询方式

用这个方法，如果 DataN-2 没有被读，在 DataN-1 之后，通讯会被拉长（RxNE 和 BTF 都被置位）。然后，在读 DR 寄存器的 DataN-2 前，清 ACK 位，以确保 ACK 位在 DataN ACK 之前被清掉。在读 DataN-2 之后，置位 STOP/START 位，并读 DataN-1。在 RxNE 置位后，读 DataN。

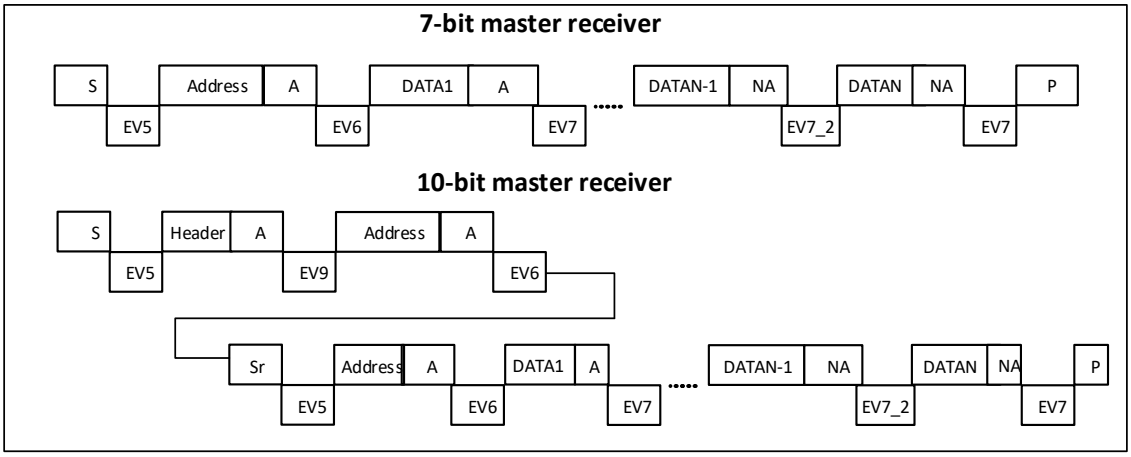


图 36-7 方法 2：N>2 时主模式发送时的时序

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV5：SB=1，先读 SR1 寄存器先读 SR2 寄存器，再写 DR 寄存器，清零该位

EV6：ADDR，先读 SR1，再读 SR2，清零该位

EV7：RxNE=1，读 DR 寄存器清零该位

EV7_2：BTF=1, DataN-2 存在 DR 寄存器中，DataN-1 存在移位寄存器中，写 ACK=0，读 DR 寄存器中的 DataN-2。置位 STOP，读 DataN-1

EV9： ADD10=1,读 SR1 然后写 DR 寄存器清除该事件

注:

1) EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束
2) EV7 软件序列必须在当前字节发送完成前执行完毕。在 EV7，软件序列不能在当前传输的字节传输完成前执行完毕，则推荐使用 BTF 代替 RXNE。

●当 3 个字节要被读走:

— RXNE=1 , DataN-2 没有读。
— DataN-1 接收
— BTF=1，移位和 DR 寄存器满。DR 寄存器存放了 DataN-2，移位寄存器存放了 DataN-1。此时，SCL 拉低，总线上没有其他要被接收的数据

— 清零 ACK 位
— 读 DR 寄存器中的 DataN-2，这将启动移位寄存器对 DataN 的接收
— DataN 接收完成（发送 NACK 条件）
— 写 START 或者 STOP 位
— 读 DataN-1
— RxNE=1
— 读 DataN

以上流程是针对 N > 2 的描述。1 个字节和 2 个字节的接收，要用不同的处理方式，参见以下描述：

● 2 个字节接收的情况
— 置位 POS 和 ACK 位
— 等待 ADDR 置位
— 清零 ADDR 位
— 清零 ACK 位
— 等待 BTF 被置位
— 写 STOP 位
— 读 DR 两次

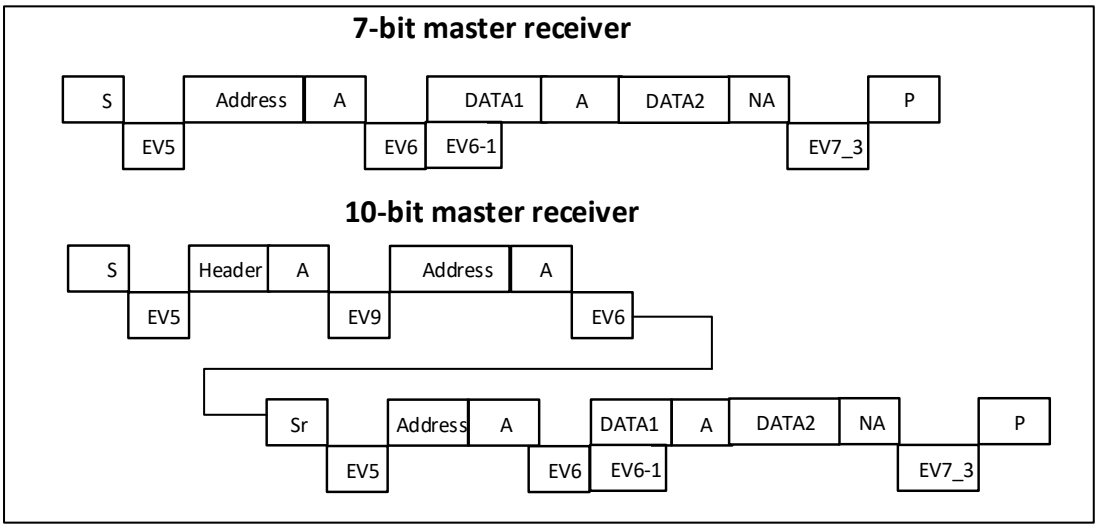


图 36-8 方法 2：N=2 时主模式发送时的时序

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV5：SB=1，先读 SR1 寄存器，再写 DR 寄存器，清零该位

EV6：ADDR=1，先读 SR1 寄存器，后读 SR2 寄存器，清零 ADDR 位

EV6_1：无相关的标志位事件。在 EV6 后，也就是地址被清零后，ACK 应该被清零

EV7_3：BTF=1，写 STOP=1，之后读两次 DR（Data1 和 Data2）

EV9： ADD10=1,读 SR1 然后写 DR 寄存器清除该事件

注:

1) EV5, EV6 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束
2) EV6_1 的软件序列必须在当前字节传输的 ACK 位发送之前完成。

- 单个字节接收的情况
  - 在 ADDR 事件里，清零 ACK 位
  - 清零 ADDR
  - 写 STOP 或者 START 位
  - 在 RxNE 标志置位后，读数据

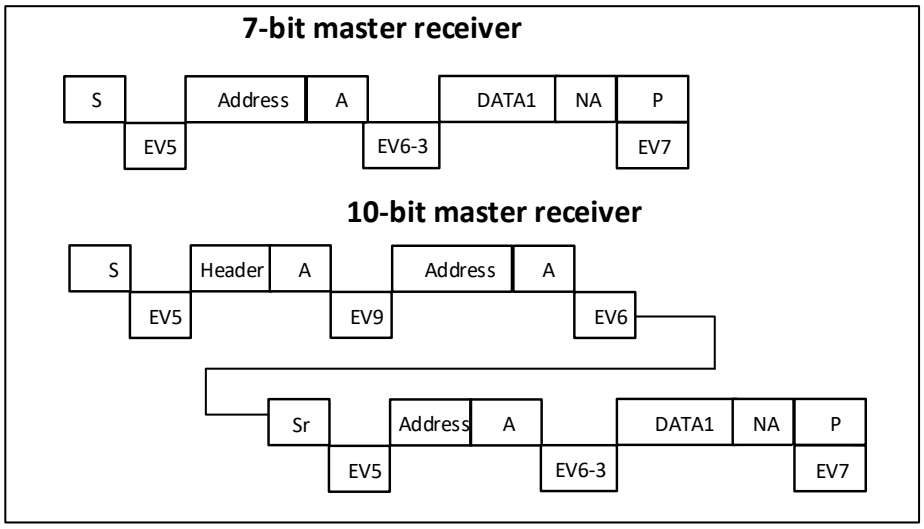


图 36-9 方法 2：N=1 时主模式发送时的时序

说明：S= Start（起始条件），SR= Repeated Start（重复的起始条件），P= Stop（停止条件），A= Acknowledge（响应），NA= Non-acknowledge（不响应），EVx= Event(ITEVFEN= 1 时产生中断)

EV5：SB=1，先读 SR1 寄存器，再写 DR 寄存器，清零该位

EV6_3：ADDR=1，写 ACK=0。先读 SR1 寄存器，后读 SR2 寄存器，清零 ADDR 位。在 ADDR 被清零后，写 STOP=1

EV7：RxNE=1，读 DR 寄存器清零该位

EV9： ADD10=1,读 SR1 然后写 DR 寄存器清除该事件

注:

EV5, EV6, EV8_1 和 EV8_2 事件，拉长 SCL 的低电平，直到相应的软件序列执行结束。

36.3.6 错误标志

总线错误（BERR）

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：

- BERR 位被置位为'1'；如果设置了 ITERREN 位，则产生一个中断；
- 在从机模式：数据被丢弃，硬件释放总线。
  - 如果是错误的起始条件，从机认为是一个重开始，并等待地址或停止条件
  - 如果是错误的停止条件，从机按正常的停止条件操作，同时硬件释放总线
- 在主机模式：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。

应答失败（AF)

当接口检测到一个无应答位时，产生应答错误。此时：

● AF 位被置位，如果设置了 ITERREN 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。

– 如果是处于从机模式，硬件释放总线

– 如果是处于主机模式，软件必须生成一个停止条件或者重开始

仲裁丢失（ARLO)

当 I2C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

● ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断

● I2C 接口自动回到从模式(MSL 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机发送重开始条件之后响应

● 硬件释放总线

过载/欠载错误（OVR)

在从机模式下，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(RxNE=1)，但在 DR 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 RxNE 位，发送器应该重新发送最后一次发送的字节

在从机模式下，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 DR 寄存器(TxE=1)，则发生欠载错误。此时：

● 在 DR 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运行错误时，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 DR 寄存器

在发送第一个字节时，必须在清除 ADDR 之后且在第一个 SCL 上升沿之前写入 DR 寄存器；如果不能做到这点，则接收方应该丢弃第一个数据。

36.3.7 DMA 功能

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 必须在当前字节传输结束之前被初始化和使能。DMAEN 位（I2C_CR2 寄存器中）必须在 ADDR 事件发生前使能。

在 master 或者 slave 模式，当时钟延长功能使能，DMAEN 可以在清零 ADDR 之前的 ADDR 事件期间置位。DMA 请求必须在当前字节传输完成之前响应。当 DMA 传输数据长度达到 DMA 设定的值时，DMA 控制器向 I^2C 发送 EOT（End of transfer），并产生传输完成中断（如果中断使能位有效）：

● 主机发送模式：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后，置位 STOP 条件。

●主机接受模式：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(对应 DMA 传输字节数 - 1)。如果在 I2C_CR2 寄存器中设置了 LAST 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。

36.3.8 支持从 STOP 模式唤醒

在 MCU 处于 STOP 模式下，可以通过 I²C 接收地址，若地址匹配则可以唤醒 MCU，反之则保持 STOP 状态。

1) 实现以上功能需要在进入 STOP 模式之前使能 I2C_CR1 的 PE 和 WUPEN，并且在 RCC 中使能 I2C 的时钟。
2) STOP 模式唤醒过程还支持可配置的 timeout 时间，可通过设置 I2C_CR1 的 WKUP_CNT 和 WKUP_DIV进行配置,若使能ITERREN,则在产生TIMEOUT之后唤醒MCU,若不使能ITERREN，则在产生 timeout 之后保持 STOP 状态。

36.3.9 中断

表 36-1 所有的 I²C 中断请求

中断事件	事件标志	开启控制位
起始位已发送(Master)	SB	ITEVTEN
地址已发送(Master) 或 地址匹配(Slave)	ADDR	
ADD10	10 位头段已发送(主)	
已收到停止(Slave)	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVTEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失(Master)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

36.3.10 系统管理总线 SMbus

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：用来发布命令，产生时钟和终止发送的设备。主机：是一种专用的主设备，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。在一个系统里只允许有一个主机。

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。本产品的 I²C 模块支持 SMbus/PMbus 功能。

表 36-2 SMbus 与 I²C 比较的不同点：

SMBus	I²C
最大传输速度 100 kHz	最大传输速度 400 kHz
最小传输速度 10 kHz	无最小传输速度
35 ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 V_{CC} 决定
不同的地址类型(保留、动态等)	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 和 I²C 之间的相似点：

- 两条线的总线协议(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

36.3.10.1 地址解析协议(ARP)

SMBus 的从地址冲突可以通过给每个从设备动态地分配一个新的唯一地址来解决。

ARP 有以下的属性：

- 地址分配利用标准 SMBus 物理层仲裁机制
- 当设备维持供电期间，分配的地址仍保持不变，允许设备在断电时保留其地址。
- 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。

36.3.10.2 SMBus 提醒模式（ALERT)

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 SMBALERT 发信号给主机表示它希望进行通信，这可通过设置 I2C_CR1 寄存器上的 ALERT 位实现。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_SR1 寄存器中的 SMBALERT 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。

36.3.10.3 超时错误（TIMEOUT)

在定时规范上 I²C 和 SMBus 之间有很多差别。

SMBus 定义一个时钟低超时，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_SR1 中的状态标志 TIMEOUT 错误表明了这个特征的状态。

36.3.10.4 使用 SMBus 模式的接口

将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 和 TEXTEN 位置 1 来使能超时检测。定时器必须按如下方式编程：即在 SMBus 规范规定的时间最大值之前检测出超时情况。

• $t_{TIMEOUT}$ 检查

要使能 $t_{TIMEOUT}$ 检查，必须将 12 位 TIMEOUTA[11:0]位编程为定时器重载值，以检查 $t_{TIMEOUT}$ 参数。必须将 TIDLE 位配置为“0”，以检测 SCL 低电平超时。

然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

如果 SCL 的低电平持续时间超过（TIMEOUTA+1）x 2048 x t_{PCLK}，I2C_SR1 寄存器的 TIMEOUT 标志将置 1。

（最大 $t_{TIMEOUT}$ = 25 ms）。

注：TIMEOUTEN 位置 1 时，不允许更改 TIMEOUTA[11:0] 位和 TIDLE 位的配置。

• $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 检查

必须根据外设配置为主设备还是从设备来配置 TIMEOUTB 定时器，以便为从设备校验 tLOW:SEXT，为主设备校验 $t_{LOW:MEXT}$。由于标准只规定了最大值，用户可以为这两个参数选择相同的值。

然后，通过将 I2C_TIMEOUTR 寄存器中的 TEXTEN 位置 1 来使能定时器。

如果 SMBus 外设延展 SCL 的累积时间超过（TIMEOUTB+1）x 2048 x t_{PCLK}，则 I2C_SR1 寄存器中的 TIMEOUT 标志将置 1。

请参见下：不同 PCLK 频率下的 TIMEOUTB 设置示例。

注：TEXTEN 位置 1 时，不允许更改 TIMEOUTB 配置。

36.3.10.5 总线空闲检测

要使能 t_{IDLE} 检查，必须将 12 位 TIMEOUTA[11:0] 字段编程为定时器重载值，以获取 t_{IDLE} 参数。必须将 TIDLE 位配置为“1”，以检测 SCL 和 SDA 高电平超时。

然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

如果 SCL 和 SDA 线的高电平持续时间超过（TIMEOUTA+1）x 4 x tPCLK，I2C_SR1 寄存器中的 TIMEOUT 标志将置 1。

请参见表:不同 PCLK 频率下的 TIMEOUTA 设置示例（最大 t_{IDLE} = 50 µs）。

注：TIMEOUTEN 置 1 时，不允许更改 TIMEOUTA 和 TIDLE 配置。

36.3.10.6 PMbus

PMbus 是基于 SMbus 而来的，传输逻辑与 SMbus 完全一致，区别在于 PMbus 定义了一些与电源管理相关的功能（由软件完成）。

36.3.10.7 SMBus：I2C_TIMEOUTR 寄存器配置示例

仅当支持 SMBus 功能时，才涉及本节内容。

- **将 $t_{TIMEOUT}$ 的最大持续时间配置为 25 ms：**

表 36-3 不同 PCLK 频率下的 TIMEOUTA 设置示例（最大 $t_{TIMEOUT}$ = 25 ms）

f_{PCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- **将 t_{LOW}:SEXT 和 t_{LOW}:MEXT 的最大持续时间配置为 8 ms：**

表 36-4 不同 PCLK 频率下的 TIMEOUTB 设置示例

f_{PCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- **将 t_{IDLE} 的最大持续时间配置为 50 µs：**

表 36-5 不同 PCLK 频率下的 TIMEOUTA 设置示例（最大 t_{IDLE} = 50 µs）

f_{PCLK}	TIMEOUTB[11:0]	TIDLE	TIMEOUTEN	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 us
16 MHz	0xC7	1	1	64 x 4 x 62.5 ns = 50 us
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 us

表 36-6 SMBus 超时规范

符号	参数	限值		单位
		最小值	最大值	
$t_{TIMEOUT}$	检测时钟低电平超时	25	35	ms
$t_{LOW:SEXT}$(1)	累积时钟低电平延长时间（从设备）	-	25	ms
$t_{LOW:MEXT}$(2)	累积时钟低电平延长时间（主设备）	-	10	ms
t_{IDLE}	SCL 和 SDA 高电平超时	-	50	us

1. $t_{LOW:SEXT}$ 是一段累积时间，即给定从设备在一条消息的最初起始到停止期间时钟信号可延展的时间。其他从设备或主设备也可能延长时钟，进而导致时钟低电平总延长时间超过 $t_{LOW:SEXT}$。因此，测量该参数时该器件应该是全速主设备寻址的唯一器件。
2. $t_{LOW:MEXT}$ 是一段累积时间，即主设备在消息的每个字节（定义为 START 到 ACK、ACK 到 ACK 或 ACK 到STOP）内时钟信号可延展的时间。从设备或其他主设备也可能延长时钟，进而导致时钟低电平总时间超过$t_{LOW:MEXT}$（针对给定字节）。因此，测量该参数时该全速主设备只寻址一个从设备。

36.4 寄存器描述

寄存器可以 half-word 或者 word 访问。

36.4.1 I²C 控制寄存器 1 (I2C_CR1)

Address offset:0x00

Reset value:0x0008 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.												WKUP_CNT		WKUP_DIV	
-												RW	RW	RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	WUPEN	SMBUS	PE
RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-

19:18	WKUP_CNT	RW	2'h2	低功耗唤醒 timeout 的计数个数 2'b00：2 2'b01：8 2'b10：32 2'b11：128 注：timout 时间的计算公式为 (WKUP_DIV/FREQ)*WKUP_CNT us 例：WKUP_DIV 选择 11 即 1024，FREQ 为 8M，CNT 选择 00 即 2，则 timeout 时间为 (1024/8)* 2 = 256 us 注：PCLK 为 72M 时，timeout 时间为 3.5us-1817.6us PCLK 为 4M 时，timeout 时间为 64 us - 32768 us
17:16	WKUP_DIV	RW	2'h0	PCLK 的分频系数，分频后的时钟用于计数 STOP 模式唤醒时的 timeout 时间 2'b00：128 2'b01：256 2'b10：512 2'b11：1024
15	SWRST	RW	0	软件复位。 当被置位时，I²C 处于复位状态。在复位释放前，要确保 I²C 的引脚被释放，总线是空闲状态。 0：I²C 模块不处于复位状态 1：I²C 模块处于复位状态 注：该位可以用于错误或锁住状态时重新初始化 I²C。如 BUSY 位为 1，在总线上又没有检测到停止条件时。
14	保留	-	-	-
13	ALERT	RW	0	SMBus 提醒。 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； PE=0 时，由硬件清除。
12	PEC	RW	0	数据包出错检查 软件可置位和清零该位，硬件可以在以下情况下清零该位：当传送 PEC 后，起始条件、或者停止条件、或者当 PE=0 时； 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。
11	POS	RW	0	ACK/PEC 位置（用于数据接收），软件可置位/清零该寄存器，或 PE=0 时由硬件清零。

				0：ACK 位控制当前移位寄存器内正在接收的字节的(N)ACK。PEC 位表明当前移位寄存器内的字节是 PEC 1：ACK 位控制在移位寄存器里接收的下一个字节的(N)ACK。PEC 位表明在移位寄存器里接收的下一个字节是 PEC 注：POS 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 ADDR 之后清除 ACK 位。为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，ADDR 延长事件时设置 PEC 位。
10	ACK	RW	0	应答使能。软件可置位/清零该寄存器，或 PE = 0 时由硬件清零。 0：无应答返回 1：在接收到一个字节后返回一个应答。（匹配地址或数据时）
9	STOP	RW	0	停止条件产生，软件可以置位/清零该寄存器，或者当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件置位。 在主模式下： 0：无停止条件产生 1：在当前字节传输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：在当前字节传输后释放 SCL 和 SDA 线
8	START	RW	0	起始条件产生。 软件可置位/清零该寄存器，或当起始条件发出后或 PE=0 时由硬件清零。 主模式： 0：无起始条件产生 1：重复产生起始条件 从模式： 0：无起始条件产生 1：当总线空闲时，产生起始条件（并由硬件自动切换到主机模式）
7	NOSTRETCH	RW	0	禁止时钟延长（从机）。 当 ADDR 或 BTF 标志被置位时，该位用于从机禁止时钟延长，直到被软件复位。 0：允许时钟延长 1：禁止时钟延长
6	ENGC	RW	0	广播呼叫使能。 0：禁止广播呼叫。以 NACK 响应地址 00h 1：允许广播呼叫。以 ACK 响应地址 00h
5	ENPEC	RW	0	PEC 使能。 0：禁止 PEC 计算

				1：开启 PEC 计算
4	ENARP	RW	0	ARP 使能。 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。
3	SMBTYPE	RW	0	SMBus 类型。 0：SMBus 设备； 1：SMBus 主机；
2	WUPEN	RW	0	从模式唤醒 STOP 模式使能 0：禁止从模式唤醒 STOP 模式 1：使能从模式唤醒 STOP 模式
1	SMBUS	RW	0	SMBus 模式使能 0：I²C 模式； 1：SMBus 模式；
0	PE	RW	0	I²C 模块使能。 0：禁止 1：I²C 使能 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 PE=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。

36.4.2 I²C 控制寄存器 2 (I2C_CR2)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res.	FREQ[6:0]						
-			RW	RW	RW	RW	RW	-	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	LAST	RW	0	DMA 最后一次传输。 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，使在最后一次接收数据时可以产生一个 NACK。
11	DMAEN	RW	0	DMA 请求使能 0：禁止 DMA 请求 1：使能 DMA 请求
10	ITBUFEN	RW	0	缓冲器中断使能。 0：当 TxE=1 或 RxNE=1 时，不产生中断

				1：当 TxE=1 或 RxNE=1 时，产生事件中断（不管 DMAEN 是何值）
9	ITEVTEN	RW	0	事件中断使能。 0：禁止 1：允许事件中断 在下列条件下，将产生该中断： ● SB=1（主模式）； ● ADDR=1（主/从模式） ● ADD10=1（主模式）； ● STOPF=1（从模式） ● BTF=1，但没有 TxE 或 RxNE 事件 ● 如果 ITBUFFEN=1，TxE 事件为 1 ● 如果 ITBUFEN=1，RxNE 事件为 1
8	ITERREN	RW	0	错误中断使能。 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： ● BERR=1 ● ARLO=1 ● AF=1 ● OVR=1 ● PECERR=1 ● TIMEOUT=1； ● SMBAlert=1.
7	保留	-	-	-
6:0	FREQ	RW	0	I^2C 模块时钟频率。 必须用 APB 时钟频率的值配置该寄存器，以产生与 I^2C 协议兼容的数据 setup 和 hold 时间。运行时钟为偶数时钟。 最小允许可设定的频率分别是 100k 模式下 4 MHz 及以上、400k 模式下大于 8MHz、1MHz 模式下大于 16MHz。 0000000：禁止 0000001：禁止 ..... 0000011：禁止 0000100：4MHz ….. …..（可配的最大频率详见 APB 的最大时钟频率）

36.4.3 I^2C 自身地址寄存器 1（I2C_OAR1)

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDMODE	Res.	Res.	Res.	Res.	Res.	ADD[9:8]		ADD[7:1]							ADD0
RW	-					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	ADDMODE	RW	0	寻址模式（从模式）。 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）；
14	保留	-	0	-
13:10	保留	-	0	-
9:8	ADD[9:8]	RW	2’h0	接口地址。 7 位地址模式该寄存器无关。 10 位地址模式位地址的 9~8 位。
7:1	ADD[7:1]	RW	7’h0	接口地址的 7~1 位。
0	ADD0	RW	0	接口地址。 7 位地址模式时该寄存器无效。 10 位地址模式时为地址的 0 位。

36.4.4 I²C 自身地址寄存器 2（I2C_OAR2）

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OA2MSK[2:0]			ADD2[7:1]							ENDUAL
-					RW			RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:11	保留	-	-	-
10:8	OA2MSK[2:0]	RW	3’h0	设备自身地址 2 屏蔽位 000：无屏蔽 001：OA2[1] 被屏蔽，为无关位。仅比较 OA2[7:2]。 010：OA2[2:1] 被屏蔽，为无关位。仅比较 OA2[7:3]。 011：OA2[3:1] 被屏蔽，为无关位。仅比较 OA2[7:4]。 100：OA2[4:1] 被屏蔽，为无关位。仅比较 OA2[7:5]。 101：OA2[5:1] 被屏蔽，为无关位。仅比较 OA2[7:6]。 110：OA2[6:1] 被屏蔽，为无关位。仅比较 OA2[7]。 111：OA2[7:1] 被屏蔽，为无关位。不进行比较，对接收到的全部 7 位地址（保留位除外）应答。
7:1	ADD2[7:1]	RW	7’h0	接口地址的 7~1 位。 双地址模式下地址的 7~1 位。
0	ENDUAL	RW	0	双地址模式使能位。 0：在 7 位地址模式下，只有 OAR1 被识别； 1：在 7 位地址模式下，OAR1 和 OAR2 都被识别。

36.4.5 I²C 数据寄存器(I2C_DR)

Address offset:0x10

Reset value:0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	DR[7:0]	RW	8'h0	8 位数据寄存器，芯片内部实际是两个独立的缓存共用一个地址，分别用于存放接收到的数据（RX_DR）、放置要发送到总线的数据（TX_DR）。 **发送器模式：** 当写一个字节至 DR 寄存器时（实际写到 TX_DR），自动启动数据传输。一旦传输开始（TxE=1），如果能及时把下一个需传输的数据写入 DR 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 DR 寄存器（实际是 RX_DR）（RxNE=1）。在接收到下一个字节（RxNE=1）之前读出数据寄存器，即可实现连续的数据接收。 注： 1) 在从机模式下，地址不会被拷贝进数据寄存器 2) 硬件不处理写冲突（如果 TxE=0，仍能写入数据寄存器） 3) 如果在处理 ACK 脉冲时发生 ARLO 事件，接收到的字节不会被拷贝到数据寄存器里，因此不能被读到

36.4.6 I²C 状态寄存器 1（I2C_SR1)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALERT	TIMEOUT	Res.	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
RC_W0		-	RC_W0					R	R	-	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	SMBALERT	RC_W0	0	SMBus 提醒状态。 主机模式： 0：无 SMBus 提醒； 1：在引脚产生 SMBAlert 提醒事件； 从机模式： 0：没有 SMBAlert 响应地址头序列； 1：收到 SMBAlert 响应地址头序列直到 SMBAlert 变低。 该位由软件写 0 或 PE=0 时由硬件清除。
14	TIMEOUT	RC_W0	0	超时或 T_{low} 错误。 0：无超时错误； 1：SCL 为低的持续时间已达到 25ms（超时）；或者主机低电平累计时钟扩展时间超过 10ms（$T_{low:\ next}$）；或从设备低电平

				累积时钟扩展时间超过 25ms（$T_{low:sext}$）；或者 T_{idle} 的时间超时；或者 I^2C 模式从 STOP 模式唤醒时可通过设置 I2C_CR1 设置 timeout 时间。 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，或当 PE=0 时由硬件清除。 注：该功能仅在 SMBUS 模式和 I^2C 模式下从 STOP 模式唤醒过程下有作用。
13	保留	-	-	-
12	PECERR	RC_W0	0	在接收时发生 PEC 错误。 0：无 PEC 错误，接收到 PEC 后返回 ACK（如果 ACK=1） 1：有 PEC 错误，接收到 PEC 后返回 NACK（不管 ACK 为何值） 该位由将软件写 0 清除，或当 PE=0 时由硬件清除。
11	OVR	RC_W0	0	过载/欠载标志。 0：无过载/欠载； 1：出现过载/欠载。 当 NOSTRETCH=1 时，在从模式下该位被硬件置位； 该位由软件写 0 清除，或当 PE=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。
10	AF	RC_W0	0	应答失败标志。 0：没有应答失败； 1：应答失败。 当没有返回应答时，硬件将置位该寄存器。 该位由软件写 0 清除，或当 PE=0 时由硬件清除。
9	ARLO	RC_W0	0	仲裁丢失（主模式）。 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，硬件将置位该寄存器。 该位由软件写 0 清除，或在 PE=0 时由硬件清除。 在 ARLO 事件之后，I^2C 接口自动切换回从模式（MSL=0）。 注：在 SMBUS 模式下，在从模式下的仲裁仅发生在数据解读，或应答传输区间（不包括地址的应答）。
8	BERR	RC_W0	0	总线出错标志。 0：无起始或者停止条件出错； 1：起始或者停止条件出错。 当接口检测到错误的起始或者停止条件，硬件将该位置 1。 该位由软件写 0 清除，或者在 PE=0 时由硬件清除。
7	TxE	R	0	数据寄存器为空（发送时）标志。 0：数据寄存器非空； 1：数据寄存器为空。 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 DR 寄存器可清除该位，或在发生一个起始或停止条件后，或当 PE=0 时由硬件自动清除。 如果收到一个 NACK，或下一个要发送的字节是 PEC（PEC=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 BTF 时写入数据，都不能清除 TxE 位，因为此时数据寄存器为空。
6	RxNE	R	0	数据寄存器非空（接收时）标志。 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，置位该寄存器。在接收地址阶段，该寄存器不置位。 软件对数据寄存器的读写操作会清除该寄存器，或当 PE=0 时由硬件清除。 注：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为此时数据寄存器仍为满。
5	保留	-	-	-
4	STOPF	R	0	停止条件检测位（从模式）。 0：没有检测到停止位； 1：检测到停止条件。 在一个应答之后（如果 ACK=1），当从设备在总线上检测到停止条件时，硬件将该位置 1。 将软件读取 I2C_SR1 寄存器后，对 I2C_CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 注：在收到 NACK 后，STOPF 位不会置位。
3	ADD10	R	0	10 位地址头序列已发送（主模式）。 0：没有 ADD10 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，当主设备将第一个字节发送出去时，硬件将该位置 1. 软件读取 I2C_SR1 寄存器后，对 I2C_DR 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 主：收到 NACK 后，该寄存器不被置位。
2	BTF	R	0	字节传输发送结束标志位。 0：字节传输发送未完成 1：字节传输发送成功结束 当 NOSTRETCH=0 时，在下列情况下硬件将置位该寄存器（从机模式，NOSTRETCH=0 时；主机模式，与 NOSTRETCH 无关）： — 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（RxNE=1）。 — 发送时，当一个新数据应该被发送，且数据寄存器还未被写入新的数据（TxE=1）。 软件读取 I2C_SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。 注：

				在收到一个 NACK 后，BTF 位不会被置位。 如果下一个要传输的字节是 PEC（I2C_SR1.TRA=1,I2C_CR1.PEC=1），BTF 位不会被置位。
1	ADDR	R	0	地址已被发送（主模式）/地址匹配（从模式）。 读 I2C_SR2 寄存器后，再读 I2C_SR1 寄存器将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。 **地址匹配（从机）：** 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 OAR 寄存器或广播呼叫地址，或 SMBus 设备默认地址、或 SMBus 主机识别出 SMBus 提醒地址匹配时，硬件将置位该位。 **注：**在从机模式下，推荐进行完整的清零序列，即在 ADDR 被置位后，先读 SR1 寄存器，再读 SR2 寄存器。 **地址已发送（主机）：** 0：地址发送没有结束； 1：地址发送结束。 ● 10 位地址时，当收到地址的第二个字节的 ACK 后置位； — 7 位地址时，当收到 ACK byte 后置位。 **注：**在收到 NACK 后，该寄存器不会被置位。
0	SB	R	0	起始位标志（主模式）。 0：未发送起始条件； 1：起始条件已发送； —当发送起始条件时，置位该寄存器。 —软件读取 I2C_SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或当 PE=0 时，由硬件清除。

36.4.7 I²C 状态寄存器 2（I2C_SR2)

Address offset:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Res.	TRA	BUSY	MSL
R	R	R	R	R	R	R	R	R	R	R	R	-	R	R	R

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:8	PEC	R	8'h0	数据包出错检测寄存器。 当 ENPEC=1 时，该寄存器存放内部的 PEC 的值。
7	DUALF	R	0	双地址标志（从模式）. 0：接收到的地址与 OAR1 匹配； 1：接收到的地址与 OAR2 匹配。

				当产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件清除该寄存器。
6	SMBHOST	R	0	接收到 SMBus 主机头序列（从模式）标志 0：未收到 SMBus 主机的地址； 1：当 SMBTYPE=1 且 ENARP=1 时，收到 SMBus 主机地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件清除该寄存器。
5	SMBDEFAULT	R	0	SMBus 从设备默认地址（从模式） 0：未收到 SMBus 设备的默认地址； 1：当 ENARP=1 时，收到 SMBus 设备的默认地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件清除该寄存器。
4	GENCALL	R	0	广播呼叫地址（从模式） 0：未收到广播呼叫地址； 1：当 ENGC=1 时，收到广播呼叫的地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件清除该寄存器。
3	保留	-	-	-
2	TRA	R	0	发送/接收标志 0：接收到数据 1：数据已发送 在整个地址传输阶段的结尾，该寄存器根据地址字节的 R/W 位来设定。 当检测到停止条件（STOPF=1），或者重复的起始条件、或者总线仲裁丢失（ARLO=1），或当 PE=0 时，硬件清除该寄存器。
1	BUSY	R	0	总线忙标志 0：在总线上无数据通讯 1：在总线上正在进行数据通讯 当检测到 SDA 或 SCL 为低电平时，硬件置位。 当检测到一个停止条件时，硬件清零。 该寄存器指示当前正在进行的总线通讯，当接口被禁用（PE=0）时该信息仍然被更新。
0	MSL	R	0	主从模式标志 0：从机模式 1：主机模式 —当接口处于主模式（SB=1）时，硬件置位； —当总线上检测到一个停止条件（STOPF=1）、仲裁丢失（ARLO=1）、或当 PE=0 时，硬件清零。

36.4.8 I²C 时钟控制寄存器(I2C_CCR)

Address offset:0x1C

Reset value:0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	F+	Res.	CCR[11:0]											
RW	RW	RW	-	RW											

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15	F/S	RW	0	I^2C 主模式选择。 0：标准模式 1：快速模式
14	DUTY	RW	0	快速模式时的占空比。 0：快速模式下：$T_{low}/T_{high}=2$ 1：快速模式下：$T_{low}/T_{high}=16/9$
13	F+	RW	0	I^2C 快速增强模式选择。 0：标准模式或快速模式，具体选择哪种由 bit15 决定； 1：快速增强模式；
12	保留	-	-	-
11:0	CCR[11:0]	RW	12'h0	快速/标准模式下的时钟控制分频系数（主模式）。 该分频系数用于设置主模式下的 SCL 时钟。 ● 标准模式或者 SMBus 模式： T_{high}=CCR x Tpclk T_{low} =CCR x Tpclk ● 快速模式： ✓ DUTY=0: T_{high}=CCR x Tpclk T_{low} =2 x CCR x Tpclk ✓ DUTY=1(为达到 400KHz): T_{high}=9 x CCR x Tpclk T_{low} =16 x CCR x Tpclk ● 快速增强版模式： ✓ DUTY=0: T_{high} =3 x CCR x Tpclk T_{low} =5 x CCR x Tpclk ✓ DUTY=1: T_{high} =2 x CCR x Tpclk T_{low} =3x CCR x Tpclk 注： 1. 允许设定的最小值为 0x04，在快速模式下允许的最小值为 0x01 2. $T_{high}=t_{r(SCL)}+t_{w(SCLH)}$ 3. $T_{low}=t_{r(SCL)}+t_{w(SCLL)}$ 4. 这些延时没有过滤器 5. 只有当 PE=0 时才能配置该寄存器；

36.4.9 I2C 上升时间寄存器（I2C_TRISE)

Address offset:0x20

Reset value:0x0000 0082

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	-	-	-
12	THOLDDATA_SEL	RW	0	数据保持时间选择 0：默认硬件计算数据保持时间 1：通过 THLDDATA 配置数据保持时间
11:7	THOLDDATA	RW	5'h1	在快速/标准/快速增强模式下的最大数据保持时间(发送模式) 这些位是用于数据发送模式下，保证数据保持时间的最小时间。 例如：标准模式允许的最大 SDA 下降时间为 300ns。如果 I2C_CR2.FREQ[6:0]的值等于 0x08，T_{pclk} =125ns，则 TRISE 中配置为 0x03 （300ns/125ns = 2.4 +1 =3.4） 如果结果不为整数，则将整数部分写入 THOLDDATA，以确保配置。
6:0	TRISE	RW	7'h2	在快速/标准模式下的最大上升时间（主模式）。 这些位应该提供在主机模式下，SCL 反馈回路的最大持续时间。这样做的目的是无论 SCL 上升沿持续时间多少，SCL 都能保持一个稳定的频率。 这些位必须设置为 I^2C 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 I2C_CR2.FREQ[5:0]的值等于 0x08，T_{pclk}=125ns，则 TRISE 中配置为 0x09（1000ns/125ns = 8 + 1 = 9）。 如果结果不为整数，则将整数部分写入 TRISE，以确保 t_{HIGH} 参数。 注：当 PE=0 时才能设置该寄存器。

36.4.10 I2C 超时寄存器(I2C_TIMEOUTR)

Address offset:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
				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**
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	TEXTEN	RW	0	时钟信号延展超时使能 0：禁止时钟信号延展超时检测。 1：使能时钟信号延展超时检测 当 I^2C 接口执行 SCL 延展的累积时间超过 $t_{LOW:EXT}$ 时，将检测到超时错误（ TIMEOUT = 1)。
30:28	保留	-	-	-
27:16	TIMEOUTB[11:0]	RW	12'h0	总线超时 B 该字段用于配置累积时钟延展超时： 在主模式下，将检测主设备的累积时钟低电平延展时间（$t_{LOW:MEXT}$）

				在从模式下，将检测从设备的累积时钟低电平延展时间（$t_{LOW:SEXT}$） $t_{LOW:EXT}$ =（TIMEOUTB+1）x 2048 x t_{PCLK} 注：仅可在 TEXTEN=0 时写入这些位
15	TIMEOUTEN	RW	0	时钟超时使能 0：禁止 SCL 超时检测。 1：使能 SCL 超时检测：当 SCL 的低电平时间超过 $t_{TIMEOUT}$（TIDLE=0），或 SCL 的高电平时间超过 t_{IDLE}（TIDLE=1）时，将检测到超时错误（TIMEOUT=1）。
14:13	保留	RES	-	-
12	TIDLE	RW	0	空闲时钟超时检测 0：TIMEOUTA 用于检测 SCL 低电平超时 1: TIMEOUTA 用于检测 SCL 和 SDA 高电平超时 (总线空闲条件) 注：仅可在 TIMOUTEN=0 时写入该位。
11：0	TIMEOUTA[11:0]	RW	12’h0	总线超时 A 该字段用于配置： – SCL 低电平超时条件 $t_{TIMEOUT}$（当 TIDLE=0 时） $t_{TIMEOUT}$=（TIMEOUTA+1）x 2048 x t_{PCLK} –总线空闲条件，即 SCL 和 SDA 高电平（当 TIDLE=1 时） t_{IDLE}=（TIMEOUTA+1）x 4 x t_{PCLK} 注：仅可在 TIMOUTEN=0 时写入这些位。

37. 通用同步收发器(USART)

37.1 简介

通用同步异步收发器（USART）能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。USART 利用分数波特率发生器提供宽范围的波特率选择。

支持同步单向通信和半双工单线通信，也支持 LIN(局部互联网)，智能卡协议和 IrDA(红外数据组织)规范，以及调制解调器（CTS/RTS）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。

37.2 主要特征

- 全双工异步通信
- 可配置 16 或 8 的过采样方法，以实现速度和时钟容忍度之间的最佳选择
- 两个内置的 fifo 用于发送和接收
- 两个 FIFO 都可以通过软件来使能和提供状态位
- 双时钟域：除了 PCLK 以外，还存在的一个 USART_CLK 时钟
- 自动波特率检测功能
- 可配置的数据字长度（7 位，8 位或者 9 位）
- 可配置的数据传输方向：MSB,LSB
- 可配置的停止位—支持 0.5,1,1.5 或 2 个停止位
- 提供同步模式的输入输出功能
- 使用 DMA（直接存储器访问）实现可配置的多缓冲区通信
- 单线半双工通信
- 单独的发送器和接收器使能位
- 可配置的 TX/RX pin 脚信号
- 支持的硬件流控：modem 和 RS485
- 通信控制和错误检测信号
- 奇偶校验控制
  - 发送校验位
  - 对接收数据进行校验
- 多处理器通信。如果地址不匹配，则进入静默模式
  - 静默模式中唤醒（通过空闲总线检测或地址标志检测）
  - 唤醒接收器的方式：地址位（MSB,第 9 位），总线空闲

37.2.1 USART 拓展功能

- LIN 主机发送同步断开帧的能力以及 LIN 从机检测断开帧的能力
  - 当 USART 硬件配置成 LIN 时，能够生成 13 位断开帧并检测断开帧
- IRDA SIR 编码器解码器
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3 标准里定义的异步智能卡协议

- 智能卡用到 0.5 和 1.5 个停止位

■ MODBUS 通信

- TIMOUT 功能
- CR/LF 字符识别功能

37.2.2 USART 功能框图

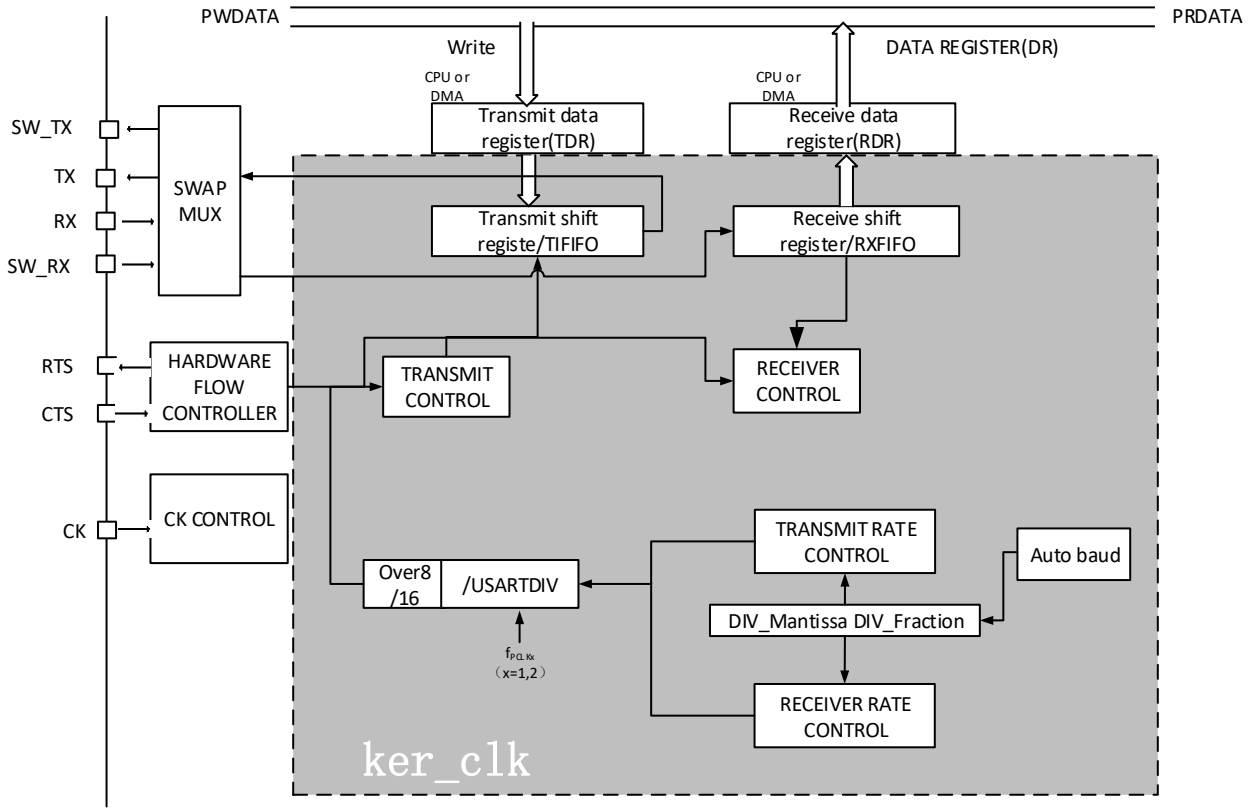


图 37-1 USART 框图

USART 有两个时钟域，一个是 PCLK,一个是 USART_CLK：USART_CLK 和 PCLK 的速度没有要求，唯一的要求是 要求软件管理通信的速度尽可能快于通信。

空闲符号被视为完全由‘1’组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位(‘1’的位数也包括了停止位的位数)。

断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 2 个停止位(‘1’)来应答起始位。

发送和接收由 1 个共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

37.3 USART 功能描述

37.3.1 传输 pin 脚

37.3.1.1 USART 特征

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：接收数据串行输入。通过过采样技术来区别数据和噪声，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器使能，但不发送数据时，TX 引脚处于高电平。在单线模式和智能卡模式里，此 I/O 口被同时用于数据的发送和接收。

37.3.1.2 USART 硬件流控

RS232 硬件流控

CTS：低电平开始发送，发送结束变为高电平。

RTS：低电平表明 USART 准备好接收数据。

RS485 硬件流控

DE:这个信号用控制外部接收器的传输

37.3.1.3 同步模式和智能卡模式

在同步主模式和智能卡模式下，需要使用以下引脚:

CK：发送器时钟输出，在同步主模式下，该引脚输出与 SPI 主模式对应的同步传输的发送器数据时钟(在起始位和停止位上没有时钟脉冲，并且在最后一个数据位上有一个发送时钟脉冲的软件选项)。在同步模式中，数据可以在 RX 引脚上同步接收。该机制可用于控制具有移位寄存器的外设(例如 LCD 驱动器)。时钟相位和极性是软件可编程的。

37.3.2 USART 特性描述

发送器根据 M 位的状态发送 7 位，8 位或 9 位的数据。当发送使能位(TE)被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

注:

1. 在 7 位数据模式，智能卡模式，LIN 主机模式不支持。
2. 在数据传输期间不能复位 TE 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。
3. TE 位被激活后将发送一个空闲帧。
4. 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。
5. USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

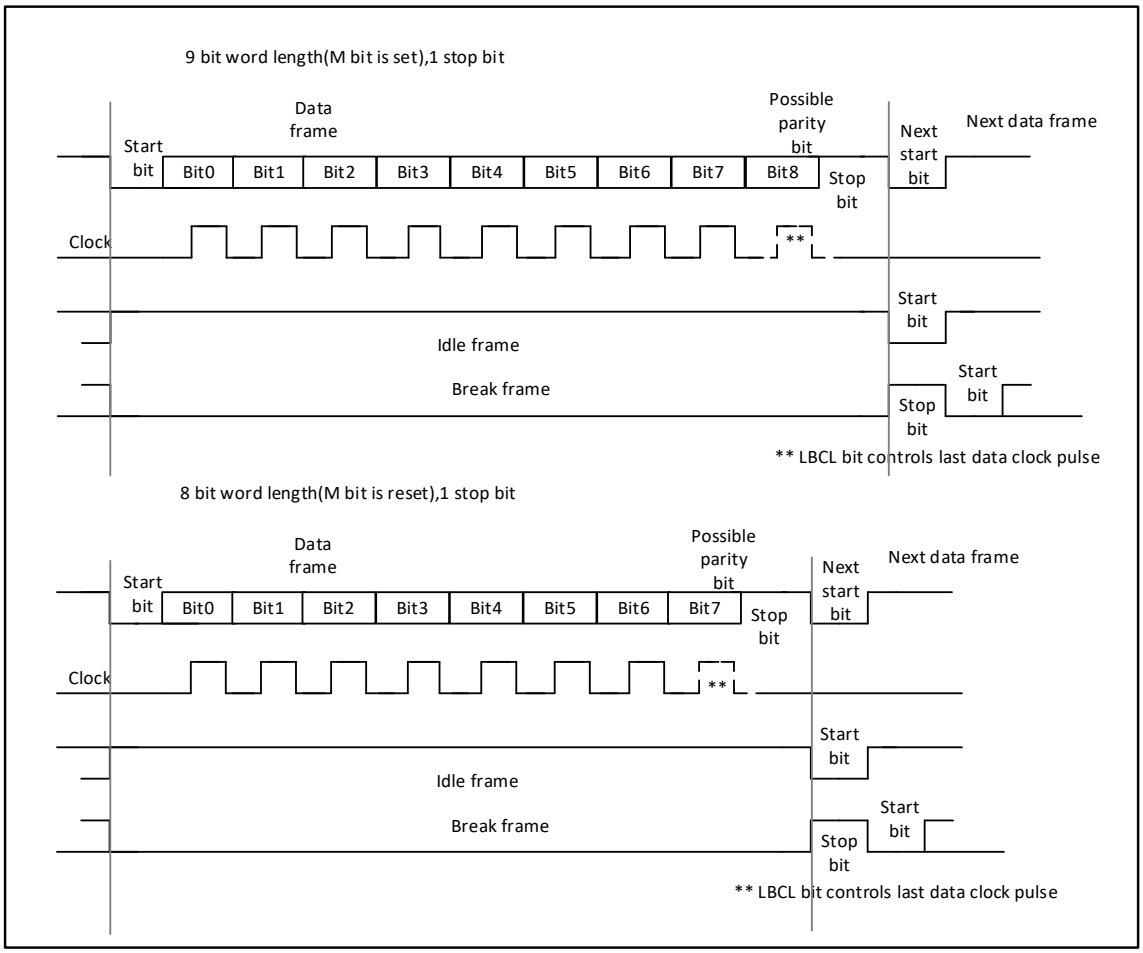


图 37-2 字长编程

37.3.3 USART FIFO 和阈值

USART 可以在 FIFO 模式下工作。USART 带有一个发送 FIFO（TXFIFO)和一个接收 FIFO（RXFIFO)。通过设置 USART_CR1.FIFOEN 来启用 FIFO 模式。此模式仅支持 UART、同步模式（发送）和智能卡模式 TX。

由于最大数据字长为 9 位，因此 TXFIFO 的宽度为 9 位。然而，RXFIFO 的默认宽度是 12 位。这是由于接收器不仅将数据存储在 FIFO 中，而且还存储与每个字符相关的错误标志(奇偶校验错误，噪声错误和帧错误标志)。

接收到的数据与相应的标志一起存储在 RXFIFO 中。但是，在读取 DR 时只读取数据。状态标志在 USART_SR 寄存器中可用。可以配置触发 TX 和 RX 中断的 TXFIFO 和 RXFIFO 级别。这些阈值通过 USART_CR1 中的 RXFTCFG 和 TXFTCFG 位进行编程。

在这种情况下:当 RXFIFO 接收的数据数量达到 RXFTCFG 位字段中编程的阈值时，在 USART_CR3 寄存器中设置 RXFT 标志，产生相应的中断(如果启用)。

当 TXFIFO 中的空位置数量达到 TXFTCFG 位字段中编程的阈值时，在 USART_CR3 寄存器中设置 TXFT 标志，产生相应的中断(如果启用)。

使能 FIFO 模式收到数据时，RXFNE 位用来表示 RXFIFO 不为空。读取 USART_DR 返回在 RXFIFO 中输入的最早的数据。当接收到数据时，它与相应的错误位一起存储在 RXFIFO 中。

•在多处理器通信模式下:当 FIFO 模式被禁用时，RXNE 标志在每个字节接收后被设置。当 DMA 读取接收数据寄存器时，它被清除。使能 FIFO 模式时，当 RXFIFO 不为空时，设置 RXFNE 标志。在每个 DMA 请求之后，从 RXFIFO 接收 1 个数据。当 RXFIFO 不为空时触发 DMA 请求(有数据要从 RXFIFO 读取)。

在单缓冲区模式下:

当 FIFO 模式被禁用时，清除 RXNE 标志是通过从 USART_DR 寄存器执行软件读取来完成的。RXNE 标志也可以通过在 USART_CR1 将 RXFRQ 位置为“1”来清除。RXNE 标志必须在接收下一个字符结束之前清除，以避免溢出错误。

使能 FIFO 模式时，当 RXFIFO 不为空时，设置 RXFNE。在每次从 USART_DR 读取操作之后，都会从 RXFIFO 检索一个数据。当 RXFIFO 为空时，RXFNE 标志被清除。当 RXFIFO 已满时，必须在接收下一个字符结束之前读取 RXFIFO 中的第一个数据，以避免溢出错误。如果设置了 RXFNEIE 位，RXFNE 标志会产生中断。

37.3.4 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位(TE)被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

37.3.4.1 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 USART_DR 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。

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：1 和 2 个停止位。

注:

在数据传输期间不能复位 TE 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。

TE 位被激活后将发送一个空闲帧。

37.3.4.2 配置停止位

随每个字符发送的停止位的位数可以通过控制寄存器 2 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。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位(当 m=0 时)；或者 11 位低电平，后跟停止位(m=1 时)。不可能传输更长的断开帧(长度大于 10 或者 11 位)。

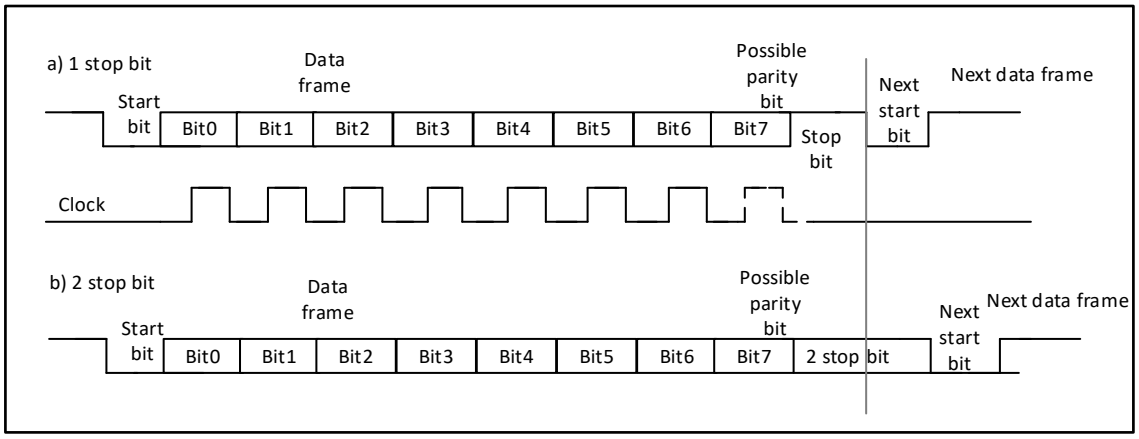


图 37-3 可配置停止位

配置步骤：

1）通过在 USART_CR1 寄存器上置位 UE 位来激活 USART

2）编程 USART_CR1 的 M 位来定义字长。

3）在 USART_CR2 中编程停止位的位数。

4）如果采用多缓冲器通信，配置 USART_CR3 中的 DMA 使能位(DMAT)。按多缓冲器通信中的描述配置 DMA 寄存器。

5）利用 USART_BRR 寄存器选择要求的波特率。

6）设置 USART_CR1 中的 TE 位，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_DR 寄存器(此动作清除 TXE 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_DR 寄存器中写入最后一个数据字后，要等待 TC=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，避免破坏最后一次传输。

37.3.4.3 单字节通讯

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 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_DR 寄存器而不会覆盖先前的数据

如果 TXEIE 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_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART_DR 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，TXE 位立即被置起。

当一帧发送完成时(停止位发送后)并且设置了 TXE 位，TC 位被置起，如果 USART_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

在 USART_DR 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(详见下图)之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1．读一次 USART_SR 寄存器；

2．写一次 USART_DR 寄存器。

注：TC 位也可以通过软件对它写’0’来清除。此清零方式只推荐在多缓冲器通信模式下使用。

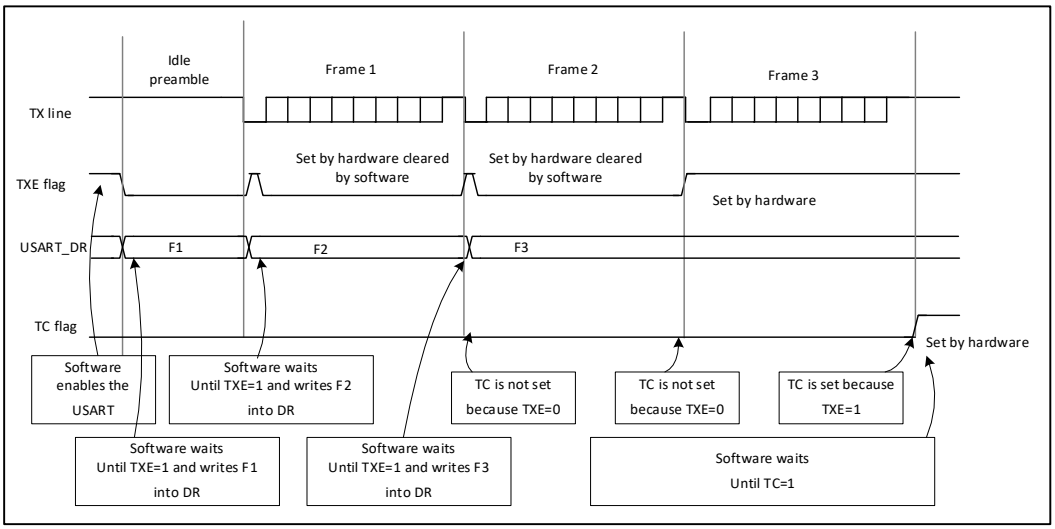


图 37-4 TC/TXE behavior when transmitting

37.3.4.4 中断字符

设置 SBK 可发送一个断开符号。断开帧长度取决 M 位。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK 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：如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置起。

37.3.4.5 空闲字符

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

37.3.5 接收器

USART 可以根据 USART_CR1 的 M 位接收 8 位或 9 位的数据字。

37.3.5.1 起始位检测

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：1 1 1 0 X 0 X 0 X 0 0 0 0

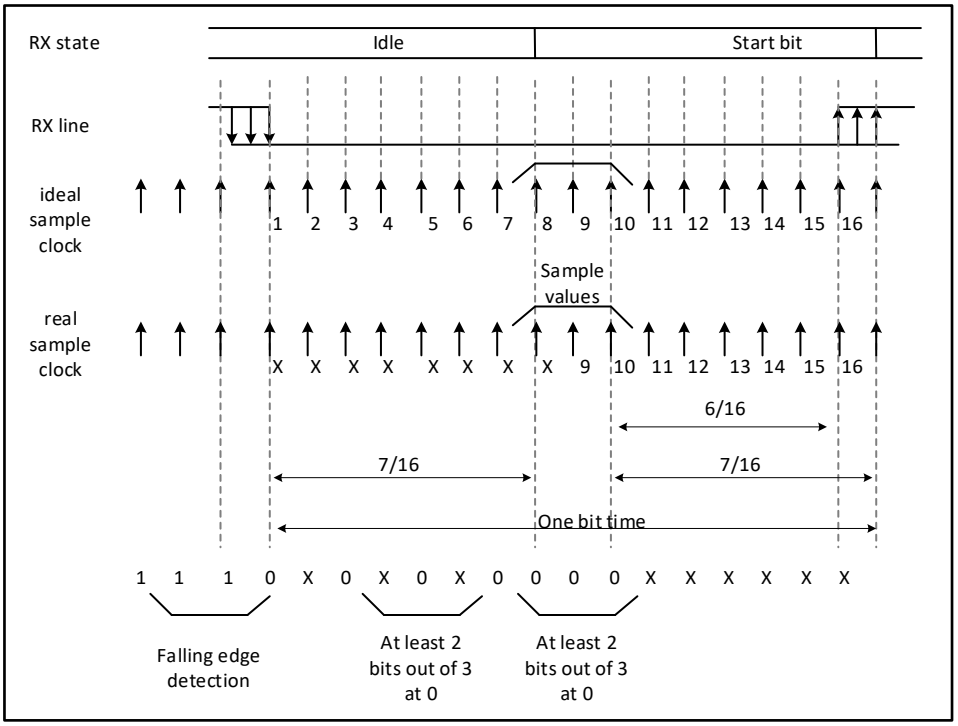


图 37-5 Start bit detection

如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)等待下降沿。如果 3 个采样点都为'0'(在第 3、5、7 位的第一次采样，和在第 8、9、10 的第二次采样都为'0')，则确认收到起始位，这时设置 RXNE 标志位，如果 RXNEIE=1，则产生中断。

如果两次 3 个采样点上仅有 2 个是'0'(第 3、5、7 位的采样点和第 8、9、10 位的采样点)，那么起始位仍然是有效的，但是会设置 NE 噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态(不设置标志位)。

如果有一次 3 个采样点上仅有 2 个是'0'(第 3、5、7 位的采样点或第 8、9、10 位的采样点)，那么起始位仍然是有效的，但是会设置 NE 噪声标志位。

37.3.5.2 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_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 USART_CR1 寄存器的 UE 置 1 来激活 USART。
2. 编程 USART_CR1 的 M 位定义字长
3. 在 USART_CR2 中编写停止位的个数
4. 如果需多缓冲器通信，选择 USART_CR3 中的 DMA 使能位(DMAR)。按多缓冲器通信所要求的配置 DMA 寄存器。
5. 利用波特率寄存器 USART_BRR 选择希望的波特率。
6. 设置 USART_CR1 的 RE 位。激活接收器，使它开始寻找起始位。

当一字符被接收到时：

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。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 RXNEIE 位被设置，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，RXNE 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_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

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

37.3.5.3 中断字符

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

37.3.5.4 空闲字符

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 IDLEIE 位被设置将产生一个中断。

37.3.5.5 上溢错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。RXNE 标记是接收到每个字节后被置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RXNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：

- ORE 位被置位。
- RDR 内容将不会丢失。读 USART_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 ORE 位

注意：当 ORE 位置位时，表明至少有 1 个数据已经丢失。有两种可能性：

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。
- 如果 RXNE=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_SR 寄存器读访问和 USART_DR 读访问之间)接收到新的数据，此种情况也可能发生。

37.3.5.6 噪声错误

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。

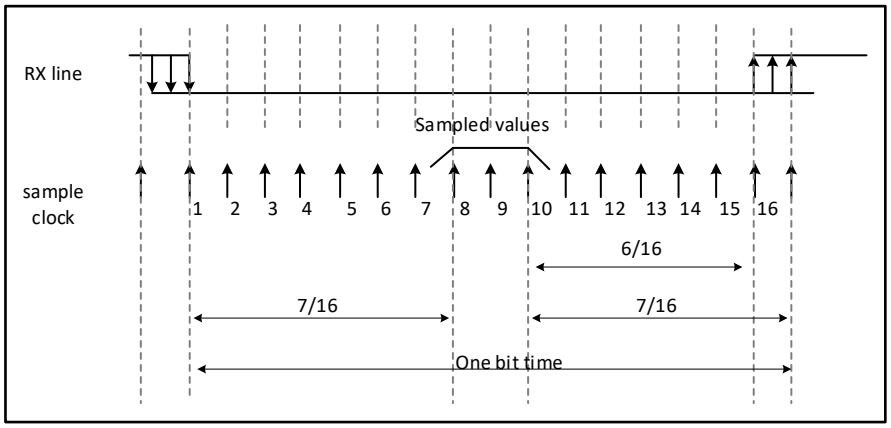


图 37-6 Data sampling for noise detection

表 37-1 Noise detection from sampled dat

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志。
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，因为 NE 标志位和 RXNE 标志位是同时被设置，RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USART_CR3 寄存器中 EIE 位，将产生一个中断。
- 先读出 USART_SR，再读出 USART_DR 寄存器，将清除 NE 标志位

37.3.5.7 帧错误

当以下情况发生时检测到帧错误：

- 由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。
- 当帧错误被检测到时：
- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 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_CR3 寄存器中 EIE 位被置位的话，将产生中断。
- 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 FE 位。

37.3.5.8 在接收时配置停止位

接收期间的可配置的停止位被接收的停止位的个数可以通过控制寄存器 2 的控制位来配置，在正常模式时，可以是 1 或 2 个。

- 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。
- 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 RXNE 标志将被设置。

特殊帧及错误处理：

断开字符：当接收到一个断开帧时，USART 按错误帧处理。
空闲字符：当检测到空闲帧时，按正常帧处理，如果 IDLEIE=1，则产生中断。

选择合适的时钟域进行：

时钟源的选择是通过 RCC 来完成控制的，在使能 USART 前，必须选择好时钟源。
时钟源的选择只要考虑两个因素：
1.是否需要 usart 在低功耗模式使用
2.通讯速度
可以在 RCC 中配置 usart_clk 时钟源，来做到双时钟。否则，usart_clk 时钟与 usart_pclk 时钟相同。
一些 usart_clk 源使 USART 能够在 MCU 处于低功耗模式时接收数据。根据接收到的数据和所选择的唤醒模式，USART 在需要时唤醒 MCU，以便通过对 USART_DR 寄存器执行软件读取或通过 DMA 来传输接收到的数据。
对于其他时钟源，系统必须处于工作状态才能使能 USART 通信. 接收机实现了不同的用户可配置的过采样技术(同步模式除外)，通过区分传入数据和噪声来进行数据恢复。这使得在最大通信速度和噪声/时钟误差抗扰度之间获得平衡。
噪音错误：

使用过采样技术(同步模式除外)，区别有效输入数据和噪音来进行数据恢复。

37.3.6 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应设置成相同。
1:16 位过采样模式

$$\text{Tx/Rx 波特率} = \frac{fck}{(16*USARTDIV)}$$

这里的 f_{CK} 是给外设的时钟。
注：在写入 USART_BRR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

表 37-2 设置波特率时的误差计算

波特率		f_{ck} = 8MHZ			f_{ck} = 36MHZ			f_{ck} = 100MHZ		
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%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.400	208.375	0.02%	2.400	937.5	0.00%	2.39998	2604.1875	0.00%
2	9.6	9.604	52.0625	0.04%	9.600	234.375	0.00%	9.59417	651.4375	0.06%
3	19.2	19.198	26.04375	0.01%	19.200	117.1875	0.00%	19.2012	325.5000	0.01%
4	57.6	57.554	8.6875	0.08%	57.600	39.0625	0.00%	57.6037	108.5000	0.01%
5	115.2	115.942	4.3125	0.64%	115.385	19.5	0.16%	115.207	54.2500	0.01%
6	230.4	228.571	2.1875	0.80%	230.769	9.75	0.16%	230.415	27.1250	0.01%
7	460.8	470.588	1.0625	2.08%	461.538	4.875	0.16%	460.829	13.5625	0.01%
8	921.6		不可能		923.077	2.4375	0.16%	925.926	6.7500	0.47%
9	2250		不可能		2250.000	1	0.00%	2272.73	2.7500	1.00%
10	4500		不可能	不可能	不可能	不可能	不可能	4545.45	1.3750	1.00%

注:

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。

2:8 位过采样模式

Tx/Rx 波特率 $=\frac{fck}{(8*USARTDIV)}$

3.不得使用小于 1 分频系数

37.3.7 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 接收器的容忍度为 97%-103%。

37.3.8 USART 自动波特率检测

- **自动波特率检测**

USART 能够基于接收到的字符自动检测波特率，并配置波特率寄存器 USART_BRR.

自动波特率检测适用于如下情况：

系统通讯速度预先未知；

系统使用精度较低的时钟源，这种机制允许在不测量时钟偏差的情况下获得正确的波特率；
时钟源频率必须与期望的通讯速度兼容:
1.16 位过采样的波特率必须在 fCK/65535 和 fCK/16 之间。
2.8 位过采样的波特率必须在 fCK/65535 和 fCK/8 之间。
通过 USART_CR2.ABRMOD 配置自动波特率检测模式。基于不同的字符模式有 2 种模式。在这些自动波特率模式中，在同步数据接收过程中测量波特率数次，并将每次测量结果与前一次进行比较。

- **自动波特率检测模式**

Mode0：此模式针对以 1 开头为起始的情况。这种情况下，是测量起始位到 1（下降沿到上升沿）。
Mode1：此模式针对以 10xx 位模式开头的任何字符。在这种情况下，USART 测量开始位和第一个数据位的持续时间。测量是沿下降沿到下降沿进行的，确保在慢信号倾斜的情况下有更好的精度。
并行数据模式：对 RX 线的每个中间转换执行检测。如果 RX 上的转换没有与接收器充分同步(接收器基于 0 位计算的波特率)，就会产生一个错误。
注：在自动波特率检测模式时，如果波特率时钟接近于 fck 时钟的一分频，会导致误差较大。

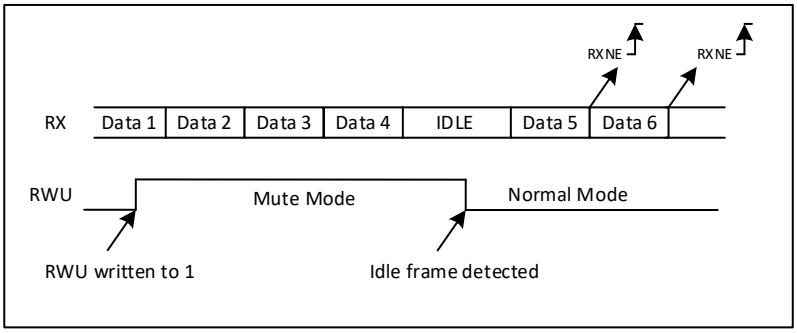
37.3.9 多处理器通信

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，我们通常希望只有被寻址的接收者才被激活，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。未被寻址的设备配置为静默模式。
通过静默功能，可以将非寻址设备置于静默模式。要使用静默模式特性，必须在 USART_CR1 寄存器中设置 RWU 位。
在静默模式里:
- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USART_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

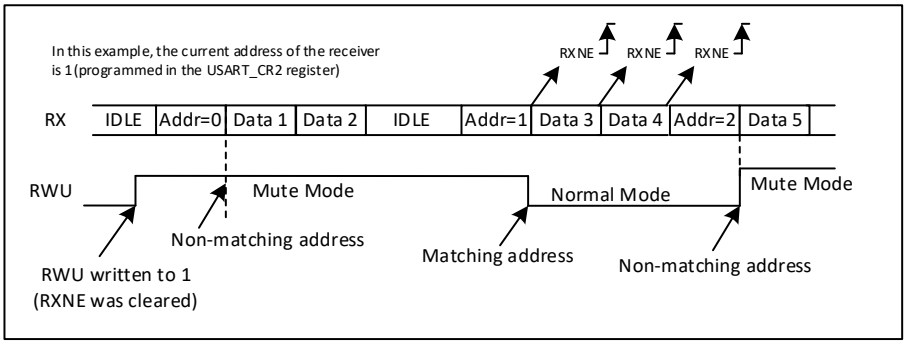
根据 USART_CR1 寄存器中的 WAKE 位状态，USART 可以用二种方法进入或退出静默模式。
- 如果 WAKE 位被设置为 0：进行空闲总线检测。
- 如果 WAKE 位被设置为 1：进行地址标记检测。

37.3.9.1 空闲总线检测(WAKE=0)

当 RWU 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。RWU 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



利用空闲总线检测的静默模式



37.3.9.2 地址标记(address mark)检测(WAKE=1)（4 位或者 7 位地址检测）

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 位或者 7 位 LSB 中。这个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_CR2 寄存器的 ADD。选择 7 位或 4 位地址检测是使用 ADDM7 位完成的。

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 RWU 位。接收该字节既不会设置 RXNE 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 RWU 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 RXNE 位，因为 RWU 位已被清零。当接收缓冲器不包含数据时(USART_SR 的 RXNE=0)，RWU 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。

注：在 7 位和 9 位数据模式下，分别对 6 位和 8 位地址(ADD[5:0]和 ADD[7:0])进行地址检测。这个地址字节不会设置 RXNE 标志，当 USART 进入静默模式时，不会发出中断或 DMA 请求。当启用 FIFO 管理时，软件在进入静默模式之前应确保 RXFIFO 中至少有一个空位置。当 RWU 写为 1 时，USART 也进入静默模式。在这种情况下，RWU 位也会自动设置。当接收到与编程地址匹配的地址字符时，USART 退出静默模式。清除 RWU 位，正常接收后续字节。当 RWU 位已被清除，地址字符可以检测 RXNE/RXFNE 位。

37.3.10 USART Modbus 通信

USART 为 Modbus/RTU 和 Modbus/ASCII 协议的实现提供基本支持。Modbus/RTU 是一种半双工的块传输协议。协议的控制部分(地址识别、块完整性控制和命令解释)必须在软件中实现。USART 提供了基本的支持：结束块检测，没有软件开销或其他资源。

Modbus/RTU

在这种模式下，一个块的结束被“沉默”(空闲行)识别超过 2 个字符。该功能通过可编程超时功能实现。超时功能和中断必须通过 USART_CR2 寄存器中的 RTOEN 位和 USART_CR1 寄存器中的 RTOIE 激活。对应于 2 个字符时间(例如 22 x 位时间)的超时值必须在 RTO 寄存器中编程。当接收线在这段时间内空闲时，在接收到最后一个停止位之后，产生一个中断，通知软件当前块接收完成。

Modbus/ASCII

在这种模式下，块的结束由特定的(CR/LF)字符序列识别。USART 使用字符匹配功能来管理这个机制。通过在 ADD[7:0]字段中编写对应字符，当收到对应字符时退出静默模式。

37.3.11 校验控制

设置 USART_CR1 寄存器上的 PCE 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)。

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_CR1 中的 PS = 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_CR1 中的 PS = 1)，校验位将是 1。

传输模式: 如果 USART_CR1 的 PCE 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_SR 寄存器中的 PE 标志被置 1，并且如果 USART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

37.3.12 LIN(局域互联网)模式

配置 USART_CR2.LINEN=1 选择 LIN 模式。在 LIN 模式下，下列位必须保持为 0：

1.USART_CR2 寄存器的 CLKEN;

2.USART_CR3 寄存器的 STOP/SCEN/HDSEL/IREN.

37.3.12.1 发送

与一般 USART 发送相比，LIN 发送有如下区别：

M=0，数据长度为 8 位；

需要配置 USART_CR2.LINEN=1。置位 SBK 后将发送 13 位 0 作为断开帧。

37.3.12.2 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发送某数据帧期间，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_CR1 的 RE=1)，电路检测 RX 上的起始信号。检测起始位的方法同检测断开帧或数据是一样的。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_CR2 的 LBDL = 0) 或 11 个(当 USART_CR2 的 LBDL = 1) 连续位都是 0，并且又跟着一个定界符，USART_SR 的 LBD 标志被置位。如果 LBDIE 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，接收器继续如正常 USART 那样工作，不需要考虑检测断开帧。

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)。

本情况：断开帧长度不够：需要舍弃该帧，不设置LBD

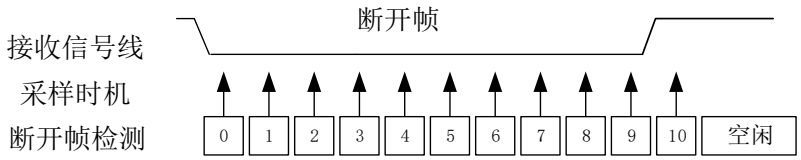
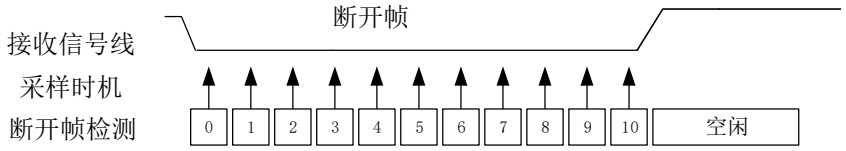


图 37-7 LIN 模式下的断开帧长度不够的情况

本情况：断开帧长度刚好，设置LBD



本情况：断开帧长度很长，设置LBD

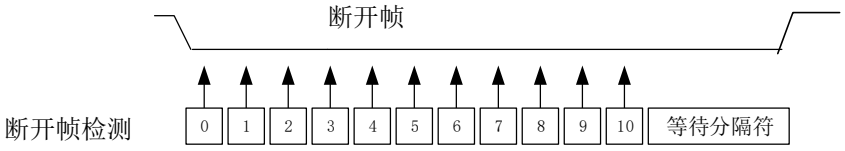
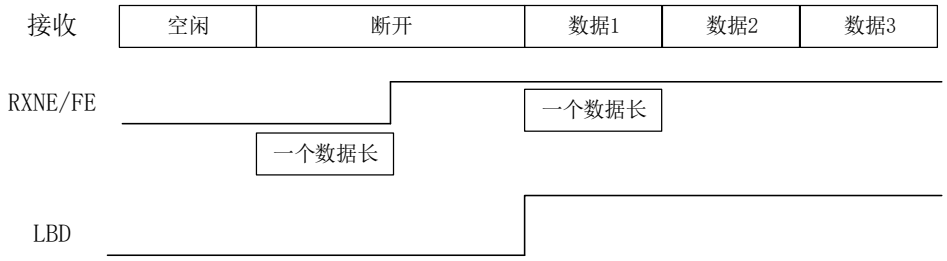


图 37-8 LIN 模式下的断开帧长度够的情况

断开发生在空闲后



断开发生在正在接收数据时

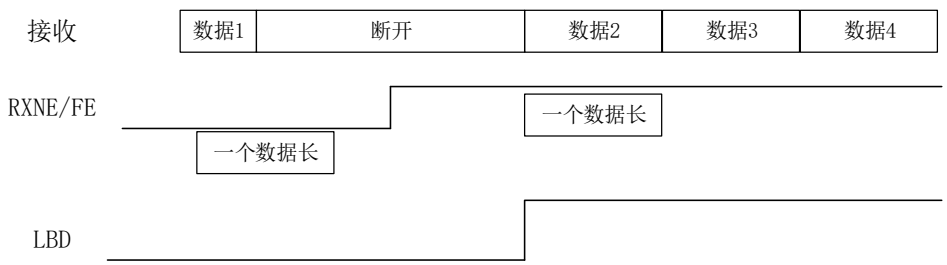


图 37-9 LIN 模式下的断开检测与帧错误的检测

37.3.13同步模式

通过在 USART_CR2 寄存器上写 CLKEN 位选择同步模式

37.3.13.1 主模式

在同步模式里，下列位必须保持清零状态：

- USART_CR2 寄存器中的 LINEN 位
- USART_CR3 寄存器中的 SCEN,HDSEL 和 IREN 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_CR2 寄存器中 LBCL 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_CR2 寄存器的 CPOL 位允许用户选择时钟极性，USART_CR2 寄存器上的 CPHA 位允许用户选择外部时钟的相位。

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 CPOL 和 CPHA)，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 RE=1，数据在 CK 上采样(根据 CPOL 和 CPHA 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保持时间(取决于波特率，1/16 位时间)。

注：

1． CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器(TE = 1)，并且发送数据时(写入数据至 USART_DR 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．应该在发送器和接收器都被禁止时，配置 LBCL,CPOL 和 CPHA；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 TE 和 RE，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)。

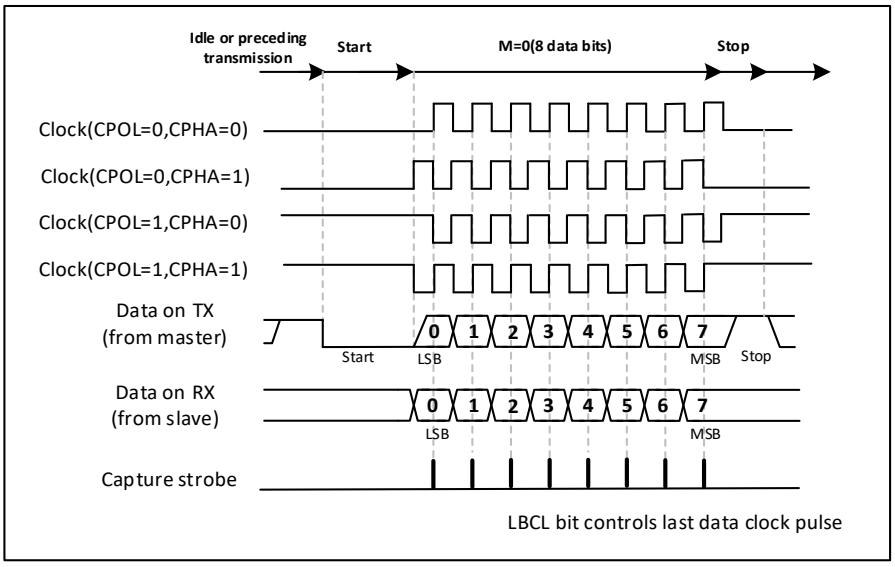


图 37-10 USART 数据时钟时序示例（M=0）

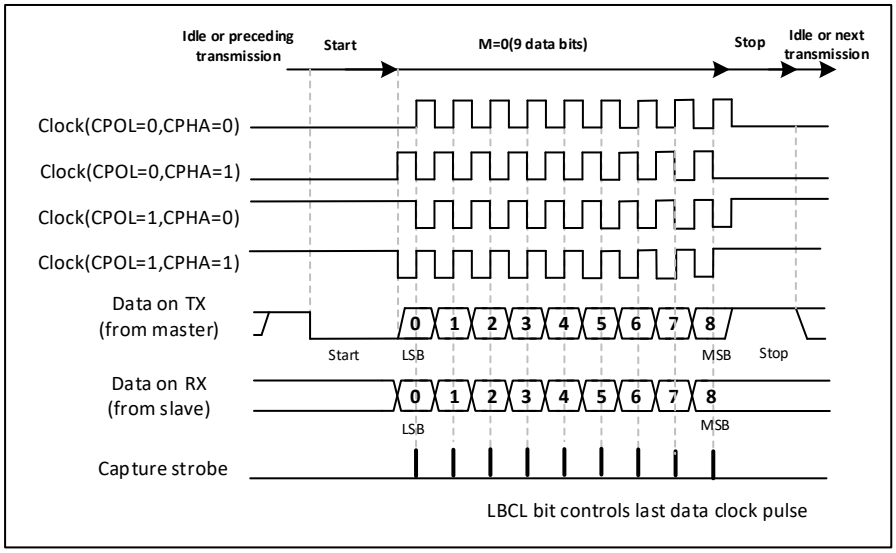


图 37-11 RX 数据采样/保持时间（M=1）

37.3.14 单线半双工通信

单线半双工模式通过设置 USART_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART_CR2 寄存器的 LINEN 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 IREN 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 USART_CR3 中的 HDSEL 位选择半双工和全双工通信。

当 HDSEL 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，必须配置成开漏。

除此以外，通信与正常 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。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

37.3.15 USART 接收超时

通过在 USART_CR2 控制寄存器中设置 RTOEN 位来启用接收器超时特性。

超时时间是使用 USART_RTOR 寄存器中的 RTO 位域进行设置的。

当超过设置的超时时间时，USART_SR 寄存器被设置为 1。如果设置了 CR1 的 RTOIE，会产生对应的中断。

37.3.16 智能卡

设置 USART_CR3 寄存器的 SCEN 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_CR2 寄存器的 LINEN 位
- USART_CR3 寄存器的 HDSEL 位和 IREN 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。T = 0（character mode）和 T = 1（block mode)都支持。

37.3.16.1 T = 0（character mode)

USART 应该被设置为：

● 8 位数据位加校验位：此时 USART_CR1 寄存器中 M=1、PCE=1

● 发送和接收时为 1.5 个停止位：即 USART_CR2 寄存器的 STOP=11

注：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使用 1.5 个停止位。

在 T = 0(字符)模式下，在保护时间段内每个字符的末尾表示奇偶校验错误。

下图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。

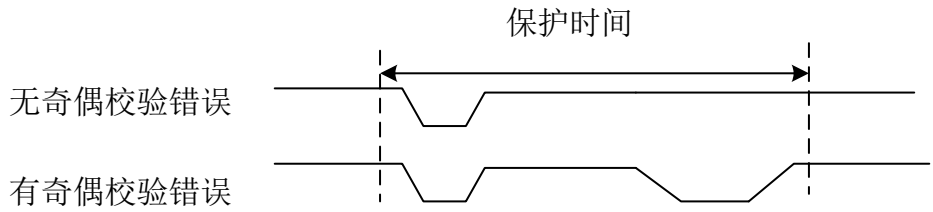


图 37-12 ISO7816-3 异步协议

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。为了做到这点，SW_RX 必须和 TX 连接到相同的 I/O 口。在发送起始位和数据字节期间，发送器的输出使能位 TX_EN 被置起，在发送停止位期间被释放(弱上拉)，因此在发现校验错误的情况下接收器可以将数据线拉低。如果 TX_EN 不被使用，在停止位期间 TX 被拉到高电平：这样的话，只要 TX 配置成开漏，接收器也可以驱动这根线。

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。在正常操作时，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。在智能卡模式里，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)。应用可以根据协议处理重新发送数据。如果设置了 NACK 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。TXE 位(在启用 FIFO 模式的情况下 TXFNF 位)可以使用寄存器中的 TXFRQ 位进行设置。如果接收到的字符错误，RXNE(在启用 FIFO 模式的情况下为 RXFNE)/接收 DMA 请求不被激活。

●智能卡传输中的自动重试:在 USART 检测 NACK 和重复字符的起始位之间插入 2.5 时钟周期的延迟。TC 位在接收到最后一个重复字符后立即设置(无保护时间)。如果软件想要再次重复它，它必须确保标准要求的最小 2 个时钟周期。

●在传输中，USART 在两个连续字符之间插入保护时间(按照保护时间寄存器)。由于保护时间是在前一个字符的停止位之后测量的，因此必须将 GT[7:0]寄存器编程为所需的 CGT(字符保护时间，由 7816-3 规范定义)减去 12(一个字符的持续时间)。

● TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC 被置起。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，TC 在这段时间被强制拉低，当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。

● TC 标志的撤销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 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是 1 或 2 个时钟周期。

● 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 0x00 数据将被当成数据而不是断开帧。
2. 当来回切换 TE 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。

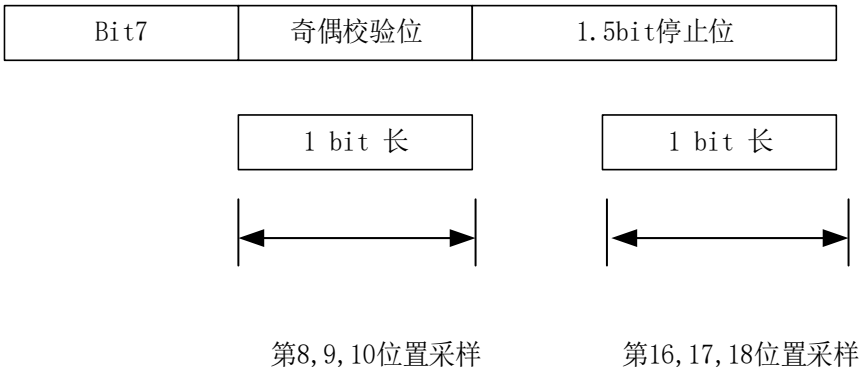


图 37-13 使用 1.5 停止位检测奇偶检验错

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 USART_GTPR 中配置。CK 频率可以从 fCK/2 到 fCK/62，这里的 fCK 是外设输入时钟。

37.3.16.2 T = 1 (block mode)

在块模式下，向智能卡发出读取请求时，软件必须将 RTOR 寄存器编程为 BWT(块等待时间)- 11 值。如果在此期限到期之前没有收到卡的答复，则会产生超时中断。如果第一个字符在期限到期之前被接收，RXNE/RXFNE 中断将发出信号

注: RXNE/RXFNE 中断必须启用，即使在 DMA 模式下使用 USART 以块模式从智能卡读取。同时，DMA 必须只在接收到第一个字节之后才启用。

在智能卡协议定义中，BWT/CWT 值应该从最后一个字符的开始(起始位)定义。考虑到最后一个字符本身的长度，RTO 寄存器必须分别编程为 BWT -11 或 CWT -11。

块长度计数器用于计算 USART 接收到的所有字符。当 USART 发送时，这个计数器被重置。块的长度由智能卡在块的第三个字节（序言字节）中进行通信。这个值必须编程到 USART_RTOR 寄存器的 BLEN 字段中。当使用 DMA 模式时，在块开始之前，必须将该寄存器字段编程为最小值。根据这个值，在接收到的第 4 个字符之后产生中断。软件必须读取 LEN 字段(第三个字节)，其值必须从接收缓冲区读取。

在中断驱动接收模式下，块的长度可以通过软件或编程 BLEN 值来检查。但是，在块开始之前，可以编程 BLEN 的最大值(0xFF)。真正的值是在接收到第三个字符后编程的。

如果块使用 LRC 纵向冗余检查(尾声字节)，则 BLEN = LEN。如果块使用 CRC 机制(2 字节)，则必须编程 BLEN = LEN+1。总块长度(包括序言、尾声和信息字节)等于 BLEN+4。块的结束通过 EOBF 标志和中断(当设置 EOBIE 位时)向软件发出信号。

37.3.17 IrDA SIR ENDEC 功能模块

通过设置 USART_CR3 寄存器的 IREN 位选择 IrDA 模式。在 IRDA 模式里，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN,STOP 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 HDSEL 位。

37.3.17.1 IrDA 正常模式

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'(见下图)。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 SIR ENDEC 最高只支持到 115.2Kbps 速率。在正常模式里，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)。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。脉冲的宽度规定为正常模式时位周期的 3/16。
- 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。接收器端的脉冲检测逻辑滤除宽度小于 2 个 PSC 周期的脉冲(PSC 是在 IrDA 低功耗波特率寄存器 USART_GTPR 中编程的预分频值)。宽度小于 1 个 PSC 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 PSC 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 PSC=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_CR2 寄存器上的 STOP 位必须配置成 1 个停止位。

37.3.17.2 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(1.42 MHz < PSC < 2.12 MHz)。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。

接收器：

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度小于 1 个 PSC 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(USART_GTPR 中的 PSC)的低电平信号才被接受为有效的信号。

注：

1. 宽度小于 2 个大于 1 个 PSC 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)。

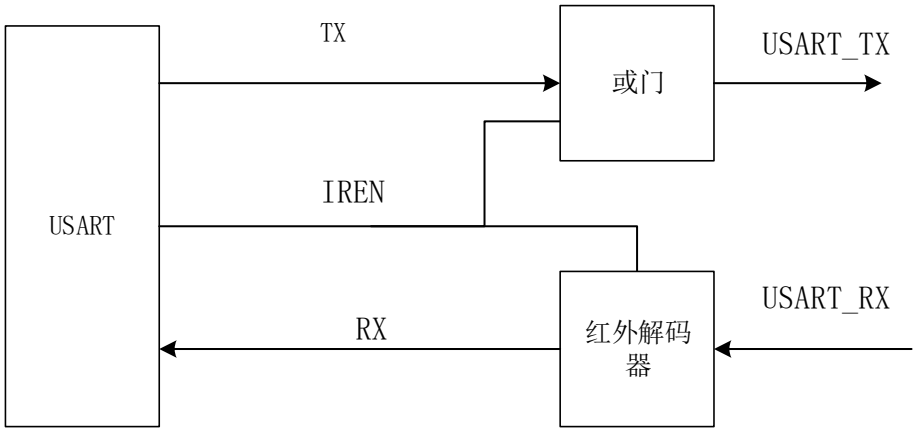


图 37-14 IrDA SIR ENDEC 框图

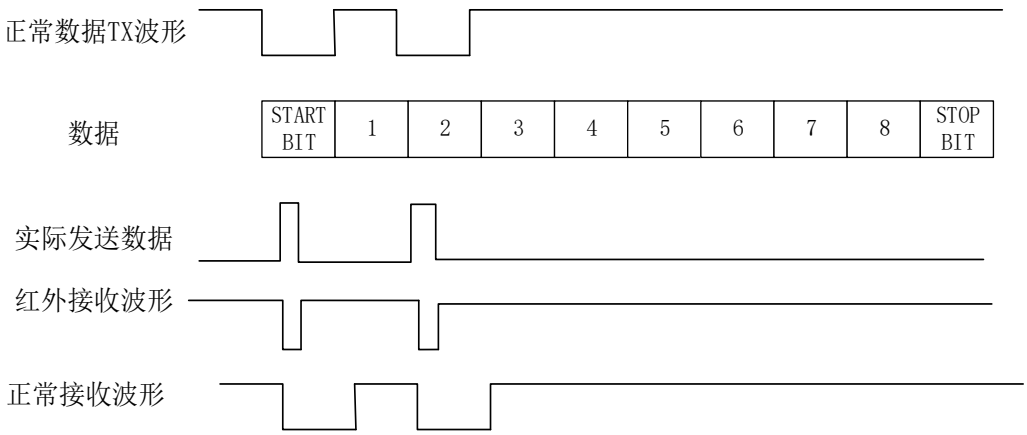


图 37-15 IrDA 数据调制（3/16）—普通模式

37.3.18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：在 USART2_SR 寄存器里，可以清零 TXE/RXNE 标志来实现连续通信。

利用 DMA 发送：

使用 DMA 进行发送，可以通过设置 USART_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为'1'时，DMA 就从指定的 SRAM 区传送数据到 USART_DR 寄存器.

1. 在 DMA 控制寄存器中写入 USART_DR 寄存器地址，将其配置为传输的目的地。在每次 TXE(如果启用 FIFO 模式 TXFNF)之后，数据也会从内存中移动到这个地址
2. 在 DMA 控制寄存器中写入内存地址，将其配置为传输源。在每次 TXE(或 TXFNF，如果启用 FIFO 模式)事件之后，数据从这个内存区域加载到 USART_DR 寄存器中
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 控制器指定的数据量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。检测 USART_SR 寄存器的 TC 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 TXE=1，再等待 TC=1。

利用 DMA 接收：

可以通过设置 USART_CR3 寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就把数据从 USART_DR 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_DR 寄存器地址配置成传输的源地址。在每个 RXNE 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 RXNE 事件后，数据将从 USART_DR 传输到此存储器。
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 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 通道的中断矢量上产生一中断。

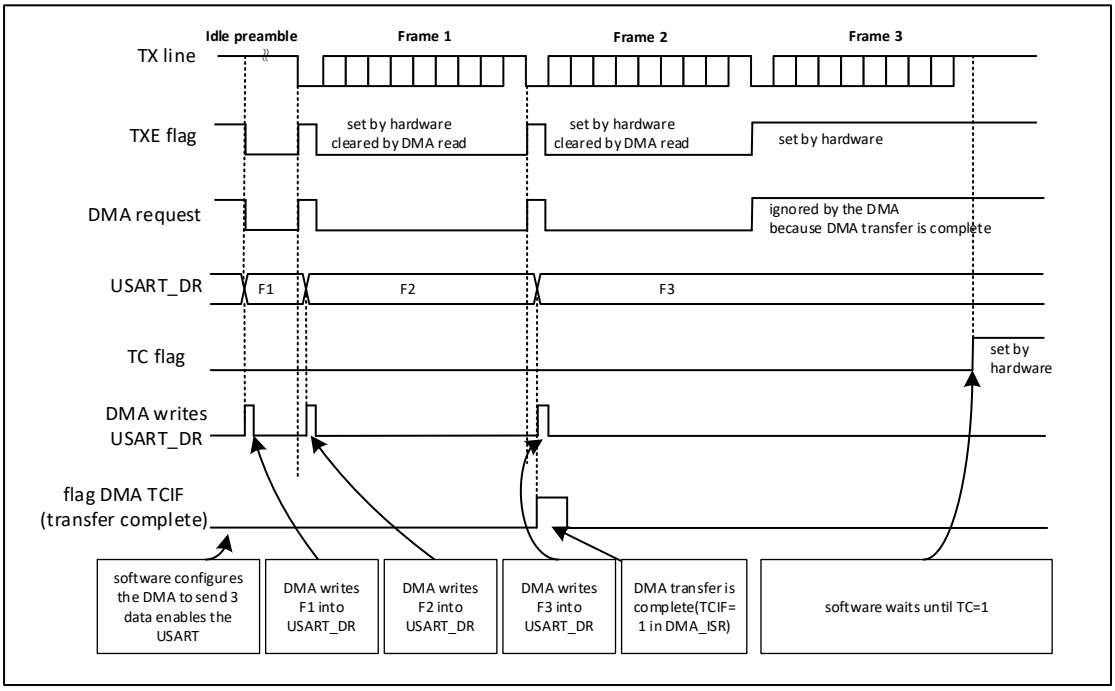


图 37-16 利用 DMA 发送

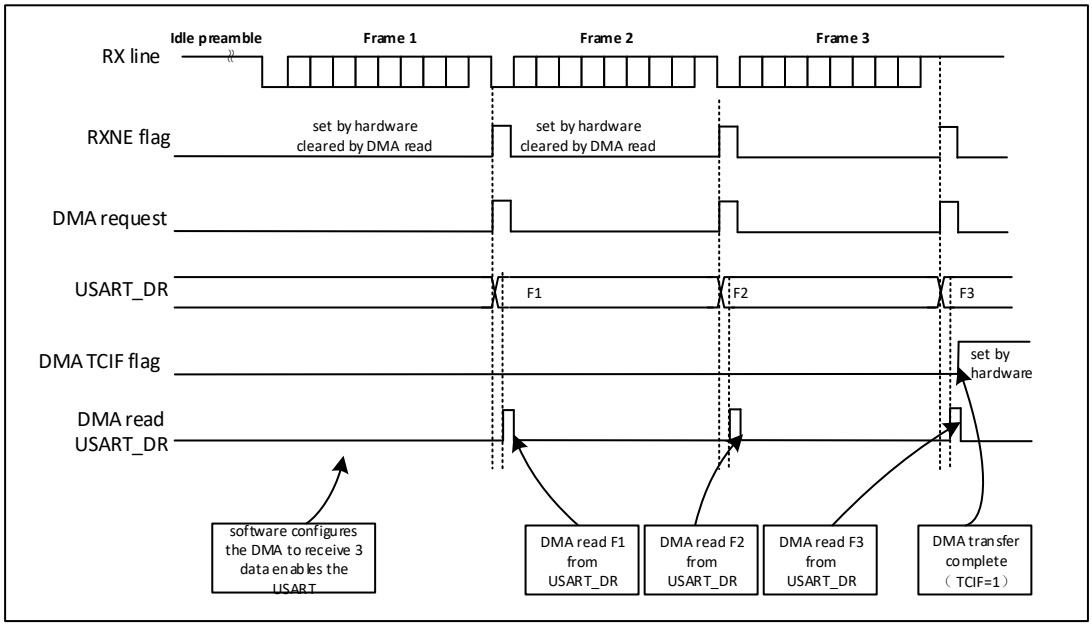


图 37-17 利用 DMA 接收

多缓冲区通信中的错误标志和中断产生

在多缓冲区通信的情况下，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 RXNE 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。

37.3.19硬件流控制

37.3.19.1 RS232 硬件流控

利用 nCTS 输入和 nRTS 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。

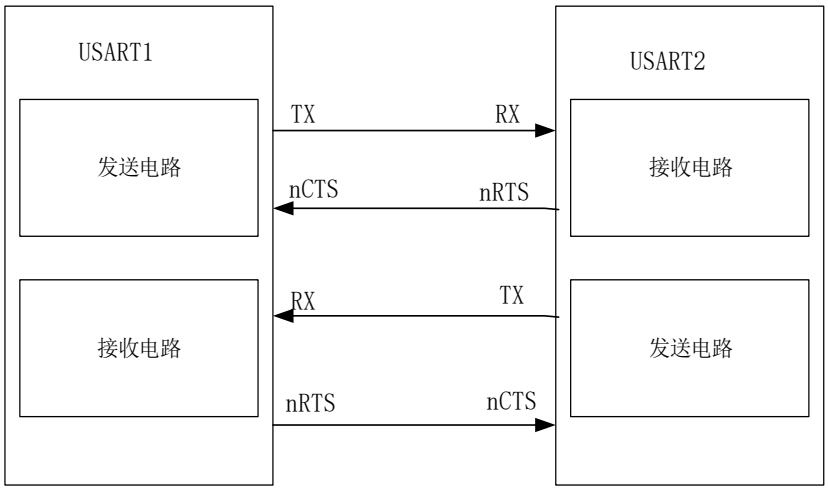


图 37-18 两个 USART 间的硬件流控制

通过将 USART_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

RTS 流控制：

如果 RTS 流控制被使能(RTSE=1)，只要 USART 接收器准备好接收新的数据，nRTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

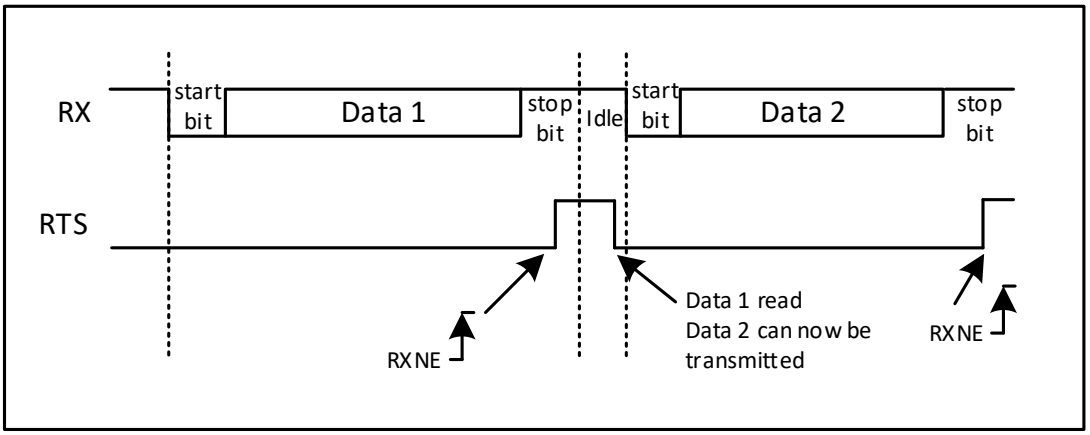


图 37-19 RTS 流控制

CTS 流控制：

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 nCTS 输入一变换状态，硬件就自动设置 CTSIF 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 CTSIE 位，则产生中断。下图是一个启用 CTS 流控制通信的例子。

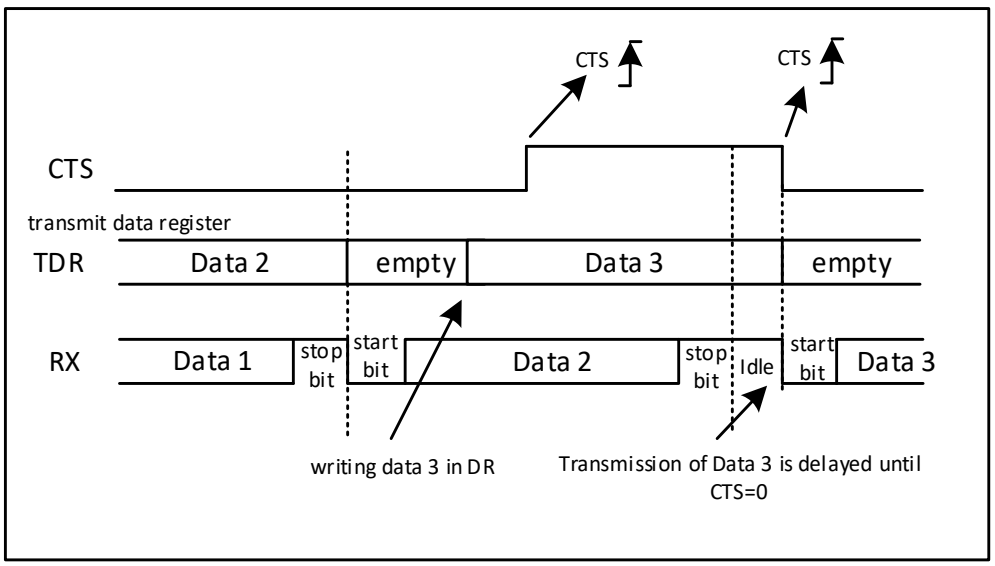


图 37-20 CTS 流控制

37.3.19.2 RS485 硬件流控

通过在 USART_CR3 控制寄存器中设置位 DEM 来启用，通过 DE(驱动使能)信号使用户能够激活外部收发器控制。断言时间是 DE 信号激活到起始位开始的时间。它是使用 USART_CR1 控制寄存器中的 DEAT[4:0]位域编程的。解除激活时间是传输数据中最后一个停止位结束到 DE 信号解除激活的时间。它是使用 USART_CR1 控制寄存器中的 DEDT[4:0]位域编程的。DE 信号的极性可以使用 USART_CR3 控制寄存器中的 DEP 位进行配置。

在 USART 中，DEAT 和 DEDT 以采样时间单位表示(1/8 或 1/16 位时间，取决于过采样率)。

37.3.20中断请求

表 37-3 USART 中断请求

中断向量	标志位	使能位
usart	txe	txeie
	cts	ctsie
	tc	tcie
	rxne	rxneie
	idle	idleie
	pe	peie
	lbd	lbdie
	ne/ore/fe	eie
	txft	txftie
	rxft	rxftie
	rxff	rxffie
	txfe	txfeie
	rtof	rtofie
	eobf	eobfie

37.4 寄存器描述

37.4.1 状态寄存器（SR）

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
Res												RTOF	EOBF	Res	TXFE
-												RC_W0	RC__W0	-	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFF	RXFT	TXFT	ABRRQ	ABRE	ABRF	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
R	R	R	W	R	R	RC_W0	RC__W0	R	RC_W0	RC_W0	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
19	RTOF	RC_W0	0	已经超过在 RTOR 寄存器中编程的超时值后，如果无任何通信，此位由硬件置 1。

				如果 USART_CR2 寄存器中 RTOIE=1，则会生成中断。 在智能卡模式下，该超时对应于 CWT 或 BWT 时间。 0：未达到超值值 1：已达到超时值，未接收到任何数据 注：如果 RTOR 寄存器中编程的时间值将 2 个字符隔开，则 RTOF 不置 1。如果此时间大于该值 +2 个采样时间 (2/16 或 2/8，具体取决于过采样方法)，则 RTOF 标志置 1。 即使 RE = 0，计数器仍会计数，但 RTOF 仅在 RE = 1 时置 1。如果 RE 置 1 时已经超时，则 RTOF 将置 1。 如果 USART 不支持接收器超时功能，该位保留且保持复位值。
18	EOBF	RC_W0	0	块结束标志 接收到完整块后，此位由硬件置 1（例如 T=1 智能卡模式）。接收到的字节数（从块的起始处，包括起始字段）等于或大于 BLEN + 4 时执行检测。 0：未达到块结束 1：已达到块结束（字符数） 注：如果不支持智能卡模式，该位保留且保持复位值。
17	保留	-	-	-
16	TXFE	R	0	当 TXFIFO 为空时，该位由硬件置 1。当 TXFIFO 包含至少一个数据时，该标志被清零。也可以通过向 TXFRQ 写入 1 将 TXFE 标志置 1。 0：TXFIFO 非空。 1：TXFIFO 为空。
15	RXFF	R	0	当接收到的数据量对应于 RXFIFO 大小 + 1（RXFIFO 已满 + USART_RDR 寄存器中的 1 个数据）时，该位由硬件置 1。
14	RXFT	R	0	RXFT：RXFIFO 阈值标志 达到在 RXFTCFG 中编程的阈值时，该位由硬件置 1。这意味着，接收 FIFO 中有（RXFTCFG - 1) 个数据，USART_RDR 寄存器中有一个数据。如果 USART_CR3 寄存器中的 RXFTIE 位 = 1（位 27），则会生成中断。 0：接收 FIFO 未达到编程的阈值。 1：接收 FIFO 已达到编程的阈值。
13	TXFT	R	0	TXFIFO 阈值标志

				当 TXFIFO 达到在 TXFTCFG 中编程的阈值时（即 TXFIFO 包含 TXFTCFG 个位置），该位由硬件置 1。 0：TXFIFO 未达到编程的阈值。 1：TXFIFO 已达到编程的阈值。
12	ABRRQ	W	0	自动波特率请求。 该位写 1 会复位 ABRF 标志位，并且请求下一帧的自动波特率检测。
11	ABRE	R	0	自动波特率错误标志。 当自动波特率检测出错（波特率超出范围或者字符比较错误）时，硬件置位该寄存器。 软件通过写 1 到 ABRRQ 寄存器清零该位。
10	ABRF	R	0	自动波特率检测标志。 当自动波特率设置（同时设置 RXNE=1，当中断使能后产生中断），或者自动波特率检测操作出错（ABRE=1，RXNE=1，FE=1）时该位由硬件置 1. 软件通过写 1 到 ABRRQ 位清零该位。
9	CTS	RC_W0	0	CTS：CTS 标志 如果设置了 CTSE 位，当 nCTS 输入变化状态时，该位被硬件置高。由软件将其清零。 如果 USART_CR3 中的 CTSIE 为’1’，则产生中断。 0：nCTS 状态线上没有变化； 1：nCTS 状态线上发生变化。
8	LBD	RC_W0	0	LBD：LIN 断开检测标志 当探测到 LIN 断开帧时，该位由硬件置‘1’，由软件清‘0’(向该位写 0)。如果 USART_CR3 中的 LBDIE = 1，则产生中断。 0：没有检测到 LIN 断开帧； 1：检测到 LIN 断开帧。 注意：若 LBDIE=1，当 LBD 为’1’时要产生中断
7	TXE	R	1	TXE:发送数据寄存器空(FIFO 禁用) TXE:TXFIFO 未满(FIFO 使能) 当 DR 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_CR1 寄存器中的 TXEIE 为 1，则产生中断。先读取 SR 再对 USART_DR 进行写操作，该位将清零。 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。
6	TC	RC_W0	1	发送完成标志 当包含有数据的一帧发送完成后，并且 TXE=1，由硬件将该位置为 1。如果 USART_CR1 寄存器中的 TXEIE 为 1，则产

				生中断。对 USART_DR 的写操作，将该位清零。 0：发送还未完成； 1：发送完成。
5	RXNE	RC_W0	0	读数据寄存器非空标志 当 RDR 移位寄存器中的数据被转移到 USART_DR 寄存器中，该位被硬件置位。如果 USART_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 USART_DR 的读操作可以将该位清零。RXNE 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方式。 0：数据也没有收到 1：收到数据，可以读出
4	IDLE	R	0	监测到总线空闲标志 当监测到总线空闲时，该位被硬件置位。如果 USART_CR1 中 IDLEIE 为 1，则产生中断。由于软件序列清除该位（先读 USART_SR;然后读 USART_DR）。 0：没有检测到空闲总线； 1：检测到空闲总线 注意：IDLE 位不会再次被置高直到 RXNE 位被置起(即又检测到一次空闲总线)
3	ORE	R	0	过载错误标志 当 RXNE=1 时，当前被接收在移位寄存器中的数据，需要传送到 DR 寄存器时，硬件将该位置位。如果 USART_CR1 中的 RXNEIE 则产生中断。由软件序列将其清零(先读 USART_SR，然后读 USART_DR)。 0：没有过载错误； 1：检测到过载错误。 注意: 该位被置位时，DR 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 EIE 位,在多缓冲区通信模式下，ORE 标志置位会产生中断。
2	NE	R	0	NE：噪声错误标志 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_SR，再读 USART_DR)。 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 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。 在多缓冲区通信模式下,如果设置了 EIE 位，则设置 NE 标志时会产生中断

1	FE	R	0	FE:帧错误标志 当检测到同步错位，过多的噪声或者检测到断开帧，该位被硬件置位。由软件序列将其清零(先读 USART_SR,再读 USART_DR)。 0:没有检测到帧错误； 1：检测到帧错误或者断开帧。 注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 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 ORE 标志位。 在多缓冲区通信模式下,如果设置了 EIE 位，则设置 FE 标志时会产生中断。
0	PE	R	0	PE:校验错误标志 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_SR 和 USART_DR）。在清除 PE 位前，软件必须等待 RXNE 标志位被指 1，如果 USART_CR1 中的 PEIE 为 1，则产生中断。 0：没有奇偶校验错误； 1：奇偶校验错误。

37.4.2 发送数据寄存器（DR）

地址偏移：0x04

复位值:0x0

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8:0	DR[8:0]	RW	0	包含要传输的数据字符。当启用奇偶校验(在 USART_CR1 寄存器中 PCE 位设置为 1)时，写在 MSB 中的值(根据数据长度，第 7 位或第 8 位)没有影响，因为它被奇偶校验所取代

37.4.3 波特率寄存器（BRR）

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Faction[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:4	DIV_Mantissa[11:0]	RW	12'h0	12bit 整数 这 12 个位用于定义 USART 除数（USARTDIV）的整数
3:0	DIV_Fraction[3:0]	RW	4'h0	4bit 小数 这 4 个位用于定义 USART 除数（USARTDIV）的小数。

37.4.4 控制寄存器 1（CR1）

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
FIFOEN	TXFRQ	RXFRQ	Res	DATAINV	TX_INV	RX_INV	RXFTCFG[2:0]			TXFTCFG[2:0]			SWAP	ADDM7	Res
RW	W	W	-	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
MSB	M1	UE	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
RW															

Bit	Name	R/W	Reset Value	Function
31	FIFOEN	RW	0	fifo 模式使能： 0：FIFO 模式关闭 1：FIFO 模式打开 FIFO 模式可用于标准 UART 通信，SPI 主模式和智能卡模式。它不能在 IrDA 和 LIN 模式下启用. 该位只能在 USART 未使能时配置（UE=0）
30	TXFRQ	W	0	TXFRQ：发送数据刷新请求 禁止 FIFO 模式时，向该位写入"1"会将 TXE 标志置 1。这可能会丢弃据。由于错误(NACK)而未发送数据且 USART_SR 寄存器中的 FE 标志有效时，只能在智能卡模式下使用此位。如果 USART 不支持智能卡模式，该位保留且必须保持复位值。使能 FIFO 时，TXFRQ 位置 1 以清空整个 FIFO。这会将 TXFE 标志置位。在 UART 模式和智能卡模式下都支持清空发送 FIFO。
29	RXFRQ	W	0	接收数据刷新请求 写 1 来清除整个接收的 FIFO。 这个使能是不读数据但是会丢弃所有 FIFO 接收数据，避免出现过载情况
28	保留	-	-	-
27	DATAINV	RW	0	2 进制数据翻转 0:1=H 0=L 1:1=L 0=H 奇偶校验位也是反向
26	TX_INV	RW	0	TX 引脚主动电平反转 该位由软件设置和清除。

				0：TX 引脚信号工作使用标准逻辑电平(V_{CC} =1/空闲，Gnd =0/标记) 1:TX 引脚信号值反转(V_{CC} =0/标记，Gnd =1/空闲)。 这使得在 TX 线路上使用外部逆变器成为可能。 该位只能在 USART 未使能时配置（UE=0）
25	RX_INV	RW	0	RXINV：RX 引脚主动电平反转该位由软件设置和清除。 0：RX 引脚信号使用标准逻辑电平工作(V_{CC} =1/空闲，Gnd =0/标记) 1:RX 引脚信号值反转(V_{CC} =0/标记，Gnd =1/空闲)。 这使得在 RX 线路上使用外部逆变器成为可能。 该位只能在 USART 未使能时配置（UE=0）。
22:24	RXFTCFG[2:0]	RW	3’b0	000:接收 FIFO 到达深度的 1/8 001:接收 FIFO 到达深度的 1/4 010:接收 FIFO 到达深度的 1/2 011:接收 FIFO 到达深度的 3/4 100:接收 FIFO 到达深度的 7/8 101:接收 FIFO 满剩余组合:保留
19:21	TXFTCFG[2:0]	RW	3’b0	XFTCFG[2:0]：TXFIFO 阈值配置 000:TXFIFO 达到其深度的 1/8 001:TXFIFO 达到其深度的 1/4 010:TXFIFO 达到其深度的 1/2 011:TXFIFO 达到其深度的 3/4 100:TXFIFO 达到其深度的 7/8 101:TXFIFO 变为空
18	SWAP	RW	1’b0	0:使用标准引脚中定义的 TX/RX 引脚 1:交换 TX 和 RX 引脚功能。这使得可以在与另一个 UART 交叉连接的情况下工作。 该位只能在 USART 未使能时配置（UE=0）
17	ADDM7	RW	1’b0	7 位地址检测/4 位地址检测 该位用于选择 4 位地址检测还是 7 位地址检测。 0：4 位地址检测 1:7 位地址检测(8 位数据模式下) 该位只能在 USART 未使能时配置（UE=0）
16	保留	-	-	-
15	MSB	RW	0	0:发送/接收数据时，数据位 0 在起始位之后。 1:数据首先通过 MSB(位 7/8)发送/接收，在起始位之后。 该位只能在 USART 未使能时配置（UE=0）
14	M1	RW	0	这个和第 12 位 M0 组成 M 决定数据大小, M： 00:8 位 01:9 位 10:7 位 该位只能在 USART 未使能时配置（UE=0） 在 7 位数据长度模式下，不支持智能卡模式、LIN 主模式。
13	UE	RW	0	UE:USART 使能

				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，以此减少功耗。该位由软件置位和清零。 0：USART 分频器和输出被禁止； 1：USART 模块使能。
12	M0	RW	0	M0：字长 该位定义了数据字的长度，由软件对其设置和清零。 该位只能在 USART 未使能时配置（UE=0）
11	WAKE	RW	0	Wake:唤醒的方法 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 0：被空闲总线唤醒； 1：被地址标记唤醒。 该位只能在 USART 未使能时配置（UE=0）
10	PCE	RW	0	PCE:校验控制使能 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 M=1,最高位就是第 9 位；如果 M=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。 该位只能在 USART 未使能时配置（UE=0）
9	PS	RW	0	PS:校验选择 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”或者清“0”。当前字节传输完成后，该选择生效。 0：偶校验； 1：奇校验。 该位只能在 USART 未使能时配置（UE=0）
8	PEIE	RW	0	PEIE:PE 中断使能 0：禁止产生中断 1：PE 中断使能
7	TXEIE	RW	0	TXE：TXE 中断使能 0：禁止； 1：TXE 中断使能
6	TCIE	RW	0	TCIE:发送完成中断使能 0:禁止产生中断。 1:当 USART_SR 中的 ORE 或者 RXNE 为 1 时，产生 USART 中断。
5	RXNEIE	RW	0	RXNEIE:接收缓冲区非空中断使能 0：禁止产生中断 1：当 USART_SR 中的 ORE 或者 RXNE 为“1”时，产生 USART 中断。
4	IDLEIE	RW	0	IDLE:IDLE 中断使能 0：禁止产生中断；

				1：当 USART_SR 中的 IDLE 为“1”时，产生 USART 中断
3	TE	RW	0	TE:发送使能 该位使能发送器。 0：禁止发送； 1：使能发送。 在数据传输过程中，除了在智能卡模式下，如果 TE 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”(空闲帧)。当 TE 被设置后，在真正发送开始之前，有一位时间的延迟。
2	RE	RW	0	RE:接收使能 0：禁止接收； 1：使能接收，并开始搜寻 RX 引脚的起始位。
1	RWU	RW	0	RWU:接收唤醒 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：接收器处于正常工作模式； 1：接收器处于静默模式。 在把 USART 置于静默模式(设置 RWU 位)之前，USART 已经先接收了一个数据字节。否则在静默模式下，不能被空闲总线检测唤醒。当配置成地址标记检测唤醒(WAKE 位=1)，在 RXNE 位被置位时，不能用软件修改 RWU 位
0	SBK	RW	0	SBK:发送断开帧 软件置位该寄存器，发送断开帧。断开帧的停止位发送后，硬件清零该寄存器。 0：不发送断开帧； 1：发送断开帧。

37.4.5 控制寄存器 2（CR2）

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
Res	Res	Res	Res	Res	Res	RTOEN	DEP	DEM	ADD[7:1]						
-						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
ADD[0]	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res	LBDIE	LBDL	Res				
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-				

Bit	Name	R/W	Reset Value	Function
31：26	保留	-	-	-
25	RTOEN	RW	1’b0	接收器超时使能 此位由软件置 1 和清零。 0：禁止接收器超时功能。 1：使能接收器超时功能。

				使能此功能后，如果 RX 线路在 RTOR（接收器超时寄存器）中编程的持续时间内处于空闲状态 （无接收），则 USART_SR 寄存器中的 RTOF 标志置 1。
24	DEP	RW	1’b0	驱动使能极性选择 0：DE 信号为高 1：DE 信号为低 该位只能在 USART 未使能时配置（UE=0）。
23	DEM	RW	1’b0	驱动使能模式： 0:DE 禁止 1:DE 使能 该位只能在 USART 未使能时配置（UE=0）
22:15	ADD[7:0]	RW	8’h0	USART 地址 地址标记检测时的地址。 在 4 位地址检测时，只有 ADD[3:0]被使用 该位只能在 USART 未使能时配置（UE=0）
14	LINEN	RW	0	LIN 模式使能。 0：LIN 模式关闭； 1：LIN 模式关闭； LIN 模式下，通过使能 SBK 位，发送 LIN 同步断开帧（13 位），以及检测断开帧。 该位只能在 USART 未使能时配置（UE=0）。
13：12	STOP[1:0]	RW	2’b0	Stop 位配置。 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； 该位只能在 USART 未使能时配置（UE=0）
11	CLKEN	RW	0	时钟使能，该位用来使能 CK 引脚。 0：CK 引脚禁止； 1：CK 引脚使能； 该位只能在 USART 未使能时配置（UE=0）
10	CPOL	RW	0	时钟极性 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。 该位只能在 USART 未使能时配置（UE=0）
9	CPHA	RW	0	时钟相位 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。 该位只能在 USART 未使能时配置（UE=0）
8	LBCL	RW	0	最后一位时钟脉冲 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字节（MSB）对应的时钟脉冲。 0：最后一位数据的时钟脉冲不在 CK pin 输出； 1：最后一位数据的时钟脉冲在 CK pin 输出。

				注：最后一位数据位就是第八个或者第九个发送的位（由 USART_CR1 中的 M 位决定） 该位只能在 USART 未使能时配置（UE=0）
7	保留	-	-	-
6	LBDIE	RW	0	LIN 断开帧中断使能。 0：禁止； 1：产生；
5	LBDL	RW	0	LIN 断开帧检测长度。 0：检测 10 位断开帧； 1：检测 11 位断开帧；
4:0	保留	-	-	-

37.4.6 控制寄存器 3（CR3）

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
Res	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					FXTFIE	RXFTIE	TXFEIE
-	RW														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFFIE	ABRMOD	Res	ABREN	OVER8	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
RW	RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30	EOBIE	RW	0	EOB 中断使能。 0：禁止； 1：中断使能；
29	RTOIE	RW	0	RTO 中断使能。 0：禁止； 1：中断使能；
28:24	DEAT[4:0]	RW	5’h0	起始位到 DE 拉高信号的时间。 以采样时间表示 (1/8 位或 1/16 位时间, 这个取决于过采样配置) 该位只能在 USART 未使能时配置（UE=0）
23:19	DEDT[4:0]	RW	5’h0	最后一个数据的停止位到 DE 的时间。 以采样时间表示 (1/8 位或 1/16 位时间, 这个取决于过采样配置) 该位只能在 USART 未使能时配置（UE=0）
18	TXFTIE	RW	1’b0	TXFIFO 阈值中断使能 0：禁止中断 1：使能中断
17	RXFTIE	RW	1’b0	RXFTIE：RXFIFO 阈值中断使能 0：禁止中断

				1：使能中断
16	TXFEIE	RW	1’b0	TXFEIE: TXFIFO 空中断使能 0：禁止中断 1：使能中断
15	RXFFIE	RW	1’b0	RXFFIE:RXFIFO 满中断使能 0：禁止中断 1：使能中断
14	ABRMOD	RW	0	自动波特率检测模式。 0：从起始位开始测量波特率； 1：下降沿到下降沿测量；
13	保留	-	-	-
12	ABREN	RW	0	自动波特率使能。 0：禁止； 1：自动波特率使能； 该位只能在 USART 未使能时配置（UE=0）
11	OVER8	RW	0	过采样模式。 0：16 位过采样； 1：8 位过采样； 该位只能在 USART 未使能时配置（UE=0）
10	CTSIE	RW	0	CTS 中断使能。 0：禁止中断 1：使能中断
9	CTSE	RW	0	CTS 使能。 0：CTS 硬件流控制禁止； 1：CTS 模式使能。 当 CTS 输入为 0 时，才会传输数据。此时，当数据写入数据寄存器后，要等待 CTS 有效后才会启动传输。 该位只能在 USART 未使能时配置（UE=0）
8	RTSE	RW	0	RTS 使能。 0：RTS 硬件流控制禁止； 1：RTS 输出使能， 只有当接收 buffer 未满时才会请求下一个数据。当前数据发送完成后，发送操作暂停。如果可以接收数据，将 RTS 置为有效（0）。 该位只能在 USART 未使能时配置（UE=0）
7	DMAT	RW	0	DMA 使能发送。 0：禁止发送 DMA； 1：使能发送 DMA；
6	DMAR	RW	0	DMA 使能接收。 0：禁止接收 DMA； 1：使能接收 DMA；
5	SCEN	RW	0	智能卡模式使能。 0：禁止；

				1：使能； 该位只能在 USART 未使能时配置（UE=0）
4	NACK	RW	0	智能卡 NACK 使能。 0：奇偶校验错误时发送 NACK 禁止； 1：奇偶校验错误时发送 NACK 使能；
3	HDSEL	RW	0	半双工选择。 0：非半双工模式； 1：半双工模式； 该位只能在 USART 未使能时配置（UE=0）
2	IRLP	RW	0	IrDA 低功耗。 0：正常模式； 1：低功耗模式； 该位只能在 USART 未使能时配置（UE=0）
1	IREN	RW	0	IrDA 模式使能。 软件使能和清零该寄存器。 0：IrDA 禁止； 1：IrDA 使能； 该位只能在 USART 未使能时配置（UE=0）
0	EIE	RW	0	错误中断使能。 0：禁止； 1: 帧错误(FE)、overrun 错误(ORE)、噪声(NE)中断使能;

37.4.7 保护时间和预分频（GTPR）

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	保留	-	-	-
15：8	GT[7:0]	RW	8’h0	保护时间值 以波特时钟为单位的保护时间。 在智能卡模式，需要此功能。 注：该位只能在 USART 未使能时配置（UE=0）
7：0	PSC[7:0]	RW	8’h0	预分频器值 在红外(IrDA)低功耗模式下： PSC[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(8 位有效)分频 00000000：Res； 00000001：对源时钟 1 分频； 00000010：对源时钟 2 分频；

				…… 1. 在红外(IrDA)正常模式下： PSC 只能设置为 00000001。 2. 在智能卡模式下： PSC[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：Res； 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：位[7:5]在智能卡模式下没有意义。 注：该位只能在 USART 未使能时配置（UE=0）

37.4.8 接收超时寄存器（RTOR）

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
BLEN[7:0]								RTO[23:16]							
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
RTO[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：24	BLEN[7:0]	RW	8’h0	块长度： 这是智能卡模式下 T=1 接收的块长度
23:0	RTO[23:0]	RW	24’h0	RX 接收超时时间 在标准模式下，在最后接收达到的字符后在设定时间内没有检测到起始位，RTOF 标志位置起。 在智能卡模式下，这个值用来执行 CWT 和 BWT。在这个情况下，CWT 和 BWT 是从最后接收到的字符起始位开始计算的

RTOR 可以动态编写。如果新值小于或等于计数器，则设置 RTOF 标志。

38. 通用异步收发器(UART)

38.1 简介

UART 是一种可编程通用异步收发器。

38.2 主要特征

- AMBA APB 接口
- 支持 5/6/7/8/9 位串行数据
- 支持 1/2 个停止位（5 位数据时：1/1.5 个停止位）
- 支持发送地址/数据
- 支持固定奇偶校验
- 支持断开帧
- 起始位错误检测
- 支持可编程分数波特率
- 支持 SWAP 功能
- 支持大小端切换 MSBFIRST 功能
- 支持 DMA 传输
- 支持 4 位小数波特率

38.3 UART 功能描述

38.3.1 UART（RS232）串行协议

因为 UART 和所选设备之间的串行通信是异步的，所以在串行数据中添加了额外的位（开始和停止）来指示开始和结束。利用这些比特可以使两个设备同步。这种由起始位和停止位组成的串行数据结构被称为一个字符，如下图所示。

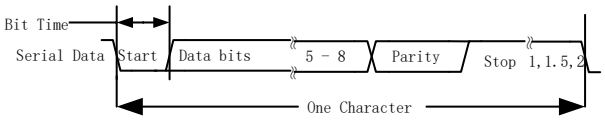


图 38-1 串行数据格式

一个额外的奇偶校验位可以添加到串行字符。该位出现在字符结构中的最后一个数据位之后和停止位之前，以便为 UART 提供对接收到的数据执行错误检查的能力。

UART 线路控制寄存器（ 寄存器说明”中的“CR1”部分）用于控制串行字符特性。数据字的各个位在起始位之后发送，从最低有效位（LSB）开始。它们后面是可选的奇偶校验位，后面是停止位，可以是 1、1.5 或 2。

注：UART 实现的停止位持续时间可能会更长，原因如下：

- 在某些配置的字符之间插入的空闲时间

传输中的所有比特都是在完全相同的持续时间内传输的；这方面的例外是当使用 1.5 个停止位时的半停止位。该持续时间被称为比特周期或比特时间；一位时间等于十六个波特时钟。

为了确保线路的稳定性，一旦检测到起始位，接收器就在位时间的大约中点对串行输入数据进行采样。



图 38-2 接收器串行数据采样点

作为 16550 标准的一部分，一个可选的波特时钟参考输出信号为需要它的接收设备提供定时信息。

UART 的波特率由单时钟实现中的串行时钟 sclk 或 pclk 以及分频锁存寄存器（BRR）控制。

下图显示了不同除数值的baudout_n输出的时序图。

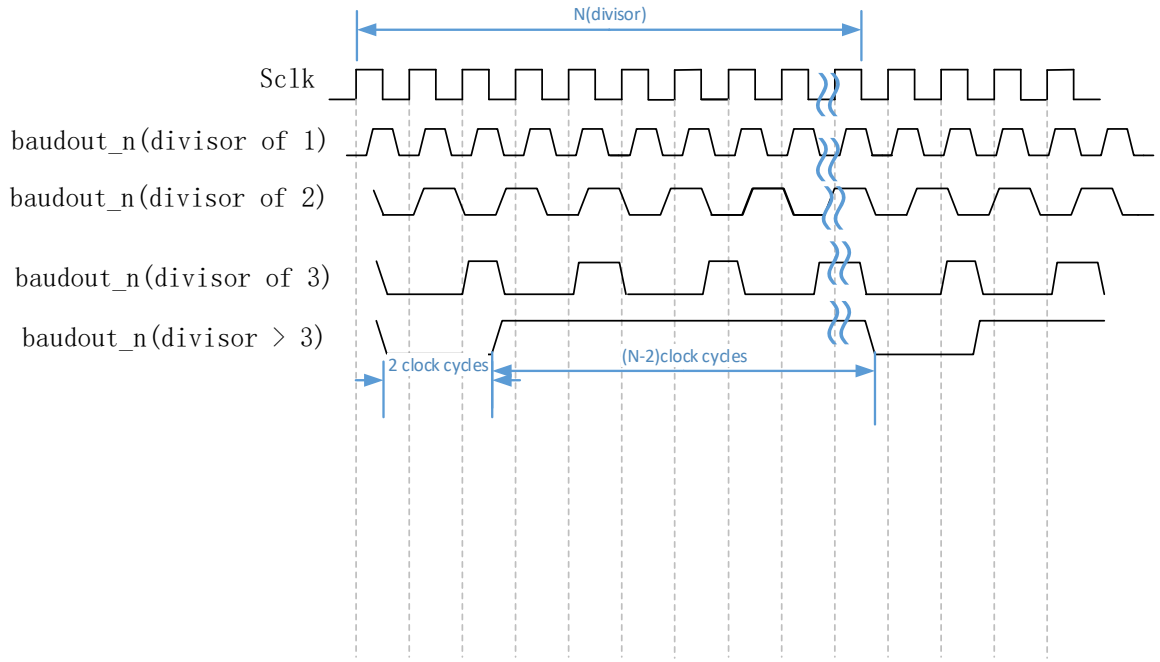


图 38-3 时序图

38.3.2 9 位数据传输

UART 在发送和接收模式下都可以被配置为具有 9 位数据传输。字符中的第 9 位出现在字符的第 8 位之后和奇偶校验位之前。下图显示了字符的串行传输，其中 D8 表示第 9 位，还显示了 9 位模式下的一般串行传输。（此为正常小端模式，如果是大端模式则调转 9 位数据位的顺序）

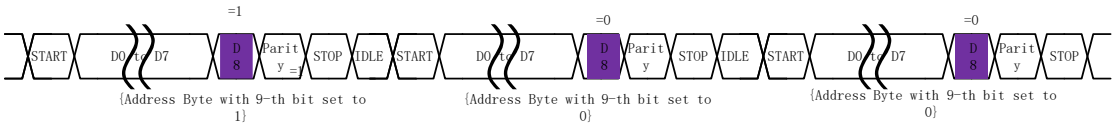


图 38-4 9位数据传输

通过启用 9 位数据传输模式，UART 可以用于多点系统，其中一个主设备连接到系统中的多个从设备。主机与其中一个从机通信。当主设备想要将数据块传输到从设备时，它首先发送一个地址字节来识别目标从设备。

地址/数据字节之间的区分是基于输入字符中的第 9 位来完成的。如果第 9 位设置为 0，则该字符表示一个数据字节。如果第 9 位设置为 1，则该字符表示地址字节。所有从系统将地址字节与它们自己的地

址进行比较，并且只有目标从系统（其中地址匹配）能够从主系统接收数据。主机开始向目标从机发送数据字节。未寻址的从系统忽略传入的数据，直到接收到新的地址字节。

请注意一个地址后面跟着 2 个数据字节。地址字节在第 9 位（D8）设置为 1 的情况下输出，而数据字节在第九位（D8）设置为 0 的情况下发出。奇偶校验位是一个可选字段。

用于 9 位数据传输的 UART 的配置执行以下操作：

- CR3[0]位用于启用或禁用 9 位数据传输。
- 在接收的情况下，CR1[1]位用于在基于硬件和软件的地址匹配之间进行选择。
- CR3[2]位用于在发送的情况下使能发送地址。
- CR3[3]位用于在基于硬件和软件的地址传输之间进行选择。
- TAR 和 RAR 寄存器分别用于发送地址和匹配接收到的地址。
- TDR、RDR 寄存器为 9 位寄存器，用于在 9 位模式下进行数据传输。
- SR[8]位用于指示地址接收中断。

发送模式

UART 支持两种类型的传输模式：

- 传输模式 0（当（CR3[3]）设置为 0 时）
- 传输模式 1（当（CR3[3]）设置为 1 时）

传输模式 0

在传输模式 0 中，地址被编程在传输地址寄存器（TAR）中，数据被写入传输保持寄存器（TDR）。TDR 寄存器的第 9 位在此模式中不适用。

下图为基于 SEND_ADDR（CR3[2]）、TDR 空条件的地址和数据传输。

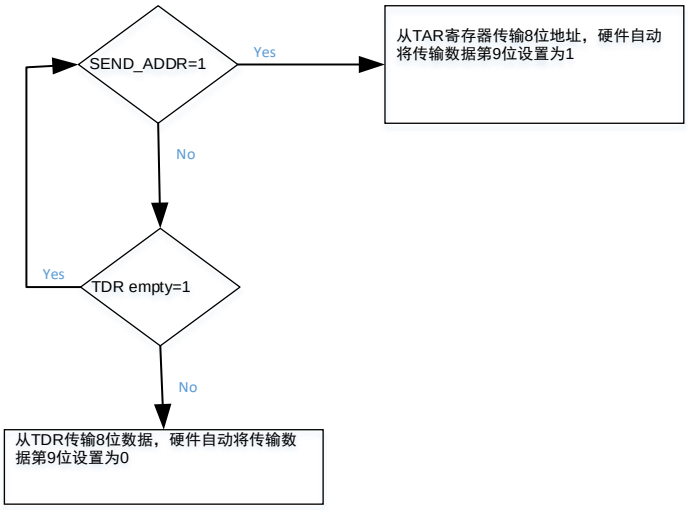


图 38-5 自动地址传输流程图

要向其传输数据的目标从机地址被编程在 TAR 寄存器中。必须启用 SEND_ADDR（CR3[2]）位来传输串行 UART 线上 TAR 寄存器中的目标从机地址，其中第 9 个数据位设置为 1，表示正在发送地址到从机。地址字符开始在 UART 线上传输后，UART 清除 SEND_ADDR 位。

传输到目标从机所需的数据通过传输保持寄存器（TDR）进行编程。数据在 UART 线上传输，第 9 个数据位设置为 0，表示正在发送数据到从机。

传输模式 1

在传输模式 1 中，TDR 寄存器为 9 位宽，地址和数据都通过 TDR 寄存器编程。UART 不区分地址和数据。SEND_ADDR（CR3[2]）位和传输地址寄存器（TAR）不适用于此模式。根据是否必须发送地址/数据，软件必须用 1/0 写入第 9 位。

表 38-1 传输关系表

M_E	TX_MODE	SEND_ADDR	使用场景
0	X	X	发 8 位 TDR 发数据
1	0	0	发“0+8 位 TDR 数据“
1	0	1	发“1+8 位 TAR 地址”
1	1	X	发 “9 位 TDR 数据”

注：该表说明作为发送方可以使用的几种发送场景和对应的配置。

接收模式

UART 支持两种接收模式：

- 硬件地址匹配接收模式（当 ADDR_Match（CR3[1]）设置为 1 时）
- 软件地址匹配接收模式（当 ADDR_Match（CR3[1]）设置为 0 时）

硬件地址匹配接收模式

在硬件地址匹配接收模式中，如果接收字符的第 9 位设置为 1，则 UART 将接收字符与在接收地址寄存器（RAR）中编程的地址进行匹配：

如果接收到的地址与 RAR 寄存器中的编程地址匹配成功，则随后接收数据字节。

如果地址匹配失败，则 UART 控制器丢弃数据字符，直到接收到匹配的地址为止。

下图为基于地址匹配功能的数据字节接收流程图。

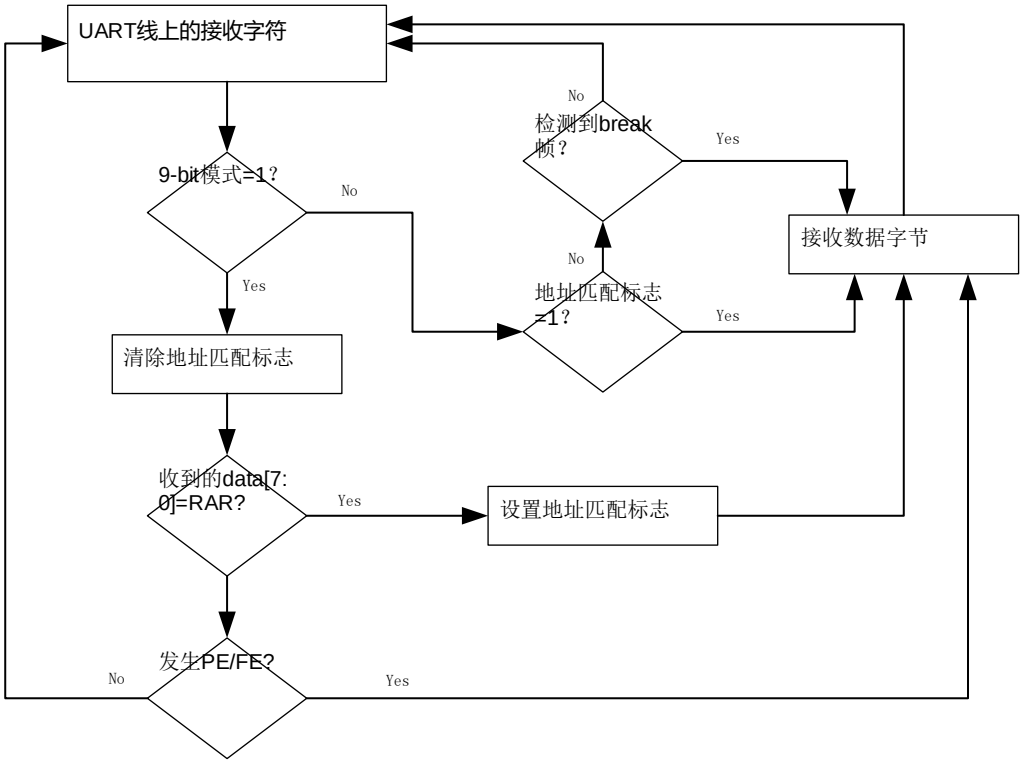


图 38-6 地址匹配功能的数据字节接收流程图

UART 接收字符，而不管第 9 位数据是否设置为 1。如果接收到的字符的第 9 位被设置为 1，则它清除内部地址匹配标志，然后将接收到的 8 位字符信息与 RAR 寄存器中编程的地址进行比较。

如果接收到的地址字符与 RAR 寄存器中编程的地址匹配成功，则地址匹配标志被设置为 1，并且接收到的字符推送到 RDR 寄存器，并且 SR 寄存器中的 ADDR_RCVD 位被设置为指示地址已被接收，随后的数据字节（接收字符的第 9 位设置为 0）被推送到 RDR。

如果地址与 RAR 寄存器匹配失败并且奇偶校验或者接收的地址字符中发现帧错误的情况下，则接收的地址字符串仍然被推送到 RDR 寄存器，ADDR_RCVD 和 PE/FE 错误位被设置为 1。

如果接收到任何中断字符，UART 将其视为一个特殊字符，并将其推送到 RDR 寄存器，而不考虑地址匹配标志。

软件地址匹配接收模式

在这种操作模式中，UART 不执行与 RAR 寄存器的接收地址字符 (第 9 位数据设置为 1) 的地址匹配。UART 始终接收 9 位数据，推进 RDR 寄存器。每当接收到地址字节并通过线路状态寄存器中的 ADDR_RCVD 位指示时，必须由用户自行比较地址。

38.3.3 波特率

UART 的波特率由 pclk 和分频锁存寄存器（BRR）和小数波特率寄存器 BRRF 控制。

波特率由以下因素决定：

- 串行时钟工作频率（pclk）
- 波特率生成器除数值 divisor（由 BRR 寄存器组成）
- 可接受的波特率误差

计算波特率的方程式如下：

波特率=串行时钟工作频率/（16* DIVISOR）——（1）

其中，DIVISOR–用于对 BRR 进行编程的数字（十六进制）。

串行时钟频率–UART pclk 引脚处的频率。

根据等式（1），DIVISOR 可以计算为：DIVISOR=串行时钟频率/（16*波特率）

（等式算出的小数部分放在 BRRF）

同样从等式（1）中，还可以示出：串行时钟频率= 波特率* 16* 除数

波特率和选定波特率定之间的误差如下所示：

百分比错误=（ |波特率-选定波特率| ）/波特率* 100%

表 38-2 设置波特率时的误差计算

波特率		fpclk = 4 MHZ			fpclk = 24 MHZ			fpclk = 48 MHZ		
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%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.404	104	0.16%	2.4	625	0.00%	2.4	1250	0.00%
2	9.6	9.615	26	0.16%	9.615	156	0.16%	9.615	312	0.16%
3	19.2	19.231	13	0.16%	19.231	78	0.16%	19.231	156	0.16%
4	57.6	62.5	4	8.51%	57.692	26	0.16%	57.692	52	0.16%
5	115.2	125	2	8.51%	115.385	13	0.16%	115.385	26	0.16%
6	230.4	250	1	8.51%	250	6	8.51%	230.769	13	0.16%
7	460.8	不可能	不可能	不可能	500	3	8.51%	500	6	8.51%
8	921.6	不可能	不可能	不可能	1500	1	62.76%	1000	3	8.51%
9	2250	不可能	不可能	不可能	不可能	不可能	不可能	3000	1	33.33%
10	4500	不可能	不可能	不可能	不可能	不可能	不可能	不可能	不可能	不可能

注:

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.在配置时钟的时候，由于波特率时钟是在系统时钟上进行分频的，但得到的时钟频率与实际时钟频率会有误差。配置的时钟误差需要在 2%以内，才可以正常工作。

38.3.4 UART 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 UART 异步接收器能够容忍的范围，UART 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 < UART 接收器的容忍度

对于正常接收数据，UART 接收器的容忍度等于最大能容忍的变化为 96%-105%。

38.3.5 利用 DMA 连续通讯

UART 可以利用 DMA 连续通讯。Rx 缓冲器和 Tx 缓冲器的 DMA 请求分别产生。

利用 DMA 发送

通过配置 UART_CR3.DMAT 使能利用 DMA 发送。当发送数据寄存器为空，DMA 就从指定的 SRAM 区传送数据到 UART_DR 寄存器。

DMA 发送的配置步骤如下：

- 配置 UART_TDR 寄存器地址作为 DMA 传输的目标地址。在每个发送数据寄存器为空事件后，数据将被传送到这个地址；
- 配置存储器（如 SRAM）地址作为 DMA 传输的源地址。在每次发送数据寄存器为空事件后，将从此存储器读数据加载到 UART_TDR 寄存器；
- 写 DMA 控制寄存器配置传输字节数；
- 写 DMA 寄存器配置通道优先级；
- 使能 DMA 通道；

当传输达到配置的传输字节数，DMA 控制器产生中断请求。

利用 DMA 接收

通过配置 UART_CR3.DMAR 使能利用 DMA 接收。每收到一个字节，DMA 就把数据从 UART_DR 寄存器传送到指定的 SRAM 区。

DMA 接收的配置步骤如下：

- 配置 UART_RDR 寄存器地址作为 DMA 传输的源地址。在每个接收数据寄存器非空事件后，数据将从此 UART_RDR 寄存器地址读出数据传输到存储器；
- 配置 SRAM 存储器地址作为 DMA 传输的目标地址。在每次接收数据寄存器非空事件后，数据将从 UART_RDR 寄存器传送到此地址；
- 写 DMA 控制寄存器配置传输字节数；

- 写 DMA 寄存器配置通道优先级；
- 使能 DMA 通道；

当传输达到配置的传输字节数，DMA 控制器产生中断请求。

38.3.6 中断

UART 中断输出信号（intr）的断言——只要几个优先中断类型中的一个被启用并激活，就会发生中断。当发生中断时，主机可访问 IIDR 寄存器判定中断类型。

以下中断类型可以通过 CR2 寄存器启用：

- 接收器错误
- 接收器数据可用
- 发送保持寄存器为空（在可编程 TDR 中断模式下）
- 忙检测指示

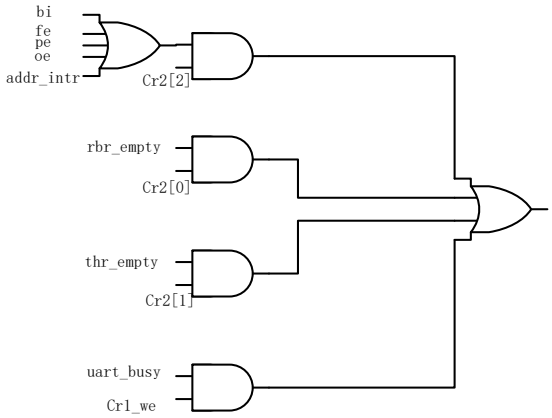


图 38-7 UART中断映射图

这些中断类型在下表中有详细说明。

表 38-3 中断控制功能

Interrupt ID				Interrupt Set and Reset Functions			
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	0	1	–	None	None	–
0	1	1	0	Highest	Receiver line status	溢出/奇偶校验/帧错误、中断或地址接收中断	对应位写 1 清零
0	1	0	0	Second	Received data available	接收器数据可用	读取接收器缓冲寄存器
0	0	1	0	Third	Transmit holding register empty	发送保持寄存器为空	写入 TDR
0	1	1	1	Fifth	Busy detect indication	UART 繁忙时（BUSY[0] 设置为 1），master 尝试写入 CR1 寄存器。	对应位写 1 清零

38.4 寄存器描述

寄存器可以 half-word 或者 word 访问。

术语：设置=设置为 1；清除=清除为 0；

38.4.1 UART DR, Address=0X00(DR)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	DR								
-							R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8:0	DR	RW	9'b0	接收/发送数据寄存器。一共是由两个寄存器组成(一个是发送的TDR, 一个是接收的 RDR），所以 DR 寄存器实现了读和写的两个功能。

38.4.2 UART BRR, Address=0X04(BRR)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:0	BRR	RW	16'b0	该寄存器构成 16 位 divisor，divisor 包含 UART 的波特率除数。 输出波特率等于串行时钟（pclk）频率除以波特率除数值的十六倍，如下所示：波特率=（串行时钟频率）/（16*divisor）。 请注意，当除数锁存寄存器（BRR）设置为零时，波特时钟被禁用，不会发生串行通信。 此外，一旦设置了 BRR，在发送或接收数据之前，应等待至少 8 个时钟周期通过。

38.4.3 UART SR, Address=0X10(SR)

地址偏移：0x10

复位值：0x00000060

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	BUSY_ERR	BUSY	ADDR_RCVD	Res	TXE	TDRE	BRI	FE	PE	ORE	RXNE
-					RC_W1	R	RC_W1	-	R	R	RC_W1				R

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
10	BUSY_ERR	RC_W1	1’b0	检测到 busy 时的误操作标志位。 错误标志：UART 繁忙时，尝试写入 CR1 寄存器。 0：无 busy 误操作错误。 1：UART 繁忙时，尝试写入 CR1 寄存器。 该位写 1 清零
9	BUSY	R	1'b0	UART 忙标志位。 这表示串行传输正在进行中，清除时表示 UART 处于空闲或非活动状态。 在以下任何一种情况下，此位都将设置为 1（忙）： -串行接口上正在进行传输 -串行接口上正在进行接收 -传输 TDR 中存在的数据，波特除数为非零（BRR 不等于 0） -接收 RDR 中存在数据 注意: 即使可能从另一个设备发送了新字符，UART 忙位也可能被清除。也就是说，如果 UART 在 TDR 和 RDR 中没有数据，并且没有正在进行的传输，并且新字符的起始位刚刚到达 UART。这是由于直到位周期的中间才看到有效的开始，并且该持续时间取决于已编程的波特除数。 0：UART 处于空闲或非活动状态 1：UART 忙
8	ADDR_RCVD	RC_W1	1'b0	地址接收。 如果启用 9 位数据模式，则该位用于指示接收数据的第 9 位被设置为 1。该位还可以用于指示输入字符是地址还是数据。 0：传入字符为数据 1：传入字符为地址
7	保留	-	-	-
6	TXE	R	1'b1	发送为空。 每当当前没有发送数据且 TDR 为空的时候，就会设置此位。 0：发送不为空 1：发送为空
5	TDRE	R	1'b1	发送保持寄存器空。 该位指示 TDR 为空。每当数据从 TDR 传输到发送器移位寄存器且没有新数据写入 TDR 时，该位被设置。如果启用了 TDR 中断，这也会导致发生 TDR 中断。 0：TDR 非空 1：TDR 为空
4	BRI	RC_W1	1'b0	break 中断位。 这用于指示在串行输入数据上检测到中断序列。如果在 UART 模式下，每当串行输入 sin 保持在逻辑“0”状态的时间超过起始时间+数据位+奇偶校验位+停止位的总和时，就会设置它。 0：未检测到 break 序列 1：检测到 break 序列 该位写 1 清零
3	FE	RC_W1	1'b0	帧错误位。

				这用于指示接收机中出现帧错误。当接收机在接收的数据中没有检测到有效的 STOP 位时，就会发生帧错误。应该注意的是，如果发生 break，也将设置帧错误位，如 break 中断位所示。之所以会发生这种情况，是因为 break 字符通过将 sin 输入保持到逻辑 0 的时间长于字符的持续时间而隐式地生成帧错误。 0：无帧错误 1：帧错误 该位写 1 清零
2	PE	RC_W1	1'b0	奇偶校验错误位。 如果设置了奇偶校验使能位，则这用于指示接收器中奇偶校验错误的发生。应该注意的是，如果发生了 break，在这种情况下，如果奇偶校验生成和检测被启用并且奇偶校验被设置为奇数，则 PE 也会被设置。 0：无奇偶校验错误 1：奇偶校验错误 该位写 1 清零
1	ORE	RC_W1	1'b0	溢出错误位。 这用于指示溢出错误的发生。如果在读取以前的数据之前接收到新的数据字符，则会发生这种情况。当新字符在从 RDR 读取前一个字符之前到达接收器时，设置 ORE 位。当这种情况发生时，RDR 中的数据将被覆盖。 0：无溢出错误 1：溢出错误 该位写 1 清零
0	RXNE	R	1'b0	数据就绪位。 这用于指示接收器在 RDR 中至少包含一个字符。读取 RDR 时，此位被清除。 0：数据未就绪 1：数据就绪

38.4.4 UART CR1, Address=0X14(CR1)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	MSBFIRST	SWAP	Res	SBK	SP	PS	PCE	STOP	M	
-						RW	RW	-	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	保留	-	-	-
9	MSBFIRST	RW	1'b0	MSB 优先。 0：start bit 后，收发第 0 位数据； 1：start bit 后，收发第 5/6/7/8/9 位数据； 在数据传输过程中，不能修改这个位。
8	SWAP	RW	1'b0	TX/RX 引脚互换。 0：TX/RX 引脚不互换； 1：TX/RX 引脚互换；

				此时用作 cross-wired 连接其他 UART 时。
7	保留	-	-	-
6	SBK	RW	1'b0	break 控制位。 这用于将 break 发送到接收设备。如果设置为 1，串行输出将强制进入间隔（逻辑 0）状态。sout 线被强制为低电平，直到 SBK 位被清除。 0：释放串行输出以进行数据传输 1：串行输出被迫进入间隔状态
5	SP	RW	1'b0	固定奇偶校验。 仅当 UART 不忙时可写。此位用于强制固定奇偶校验值。当 PCE、PS 和 SP 设置为 1 时，奇偶校验位被发送并检查是否为逻辑 0。如果 PCE 和 Stick Parity 被设置为 1，并且 PS 是逻辑 0，则奇偶校验位被发送并被检查是否为逻辑 1。如果此位设置为 0，则固定奇偶校验被禁用。 0：固定奇偶校验已禁用 1：固定奇偶校验已启用
4	PS	RW	1'b0	偶数奇偶校验选择。 仅当 UART 不忙时可写。当奇偶校验被启用时，这用于在偶数和奇数奇偶校验之间进行选择。如果设置为 1，则传输或检查偶数个逻辑“1”。如果设置为零，则传输或检查奇数个逻辑“1”。 0：奇校验 1：偶校验
3	PCE	RW	1'b0	奇偶校验启用。 仅当 UART 不忙时可写。该位用于分别启用和禁用发送和接收的串行字符中的奇偶校验生成和检测。 0：禁用奇偶校验 1：启用奇偶校验
2	STOP	RW	1'b0	停止位的数量。 仅当 UART 不忙时可写。这用于选择外围设备将发送和接收的每个字符的停止位数。如果设置为零，则在串行数据中传输 1 个停止位。如果设置为 1 并且数据位设置为 5，则发送 1.5 个停止位。否则，传输 2 个停止位。注意，无论所选择的停止位的数量如何，接收器将仅检查第一个停止位。注：由于某些配置的字符之间插入的空闲时间和传输方向上的波特时钟除数值，UART 实现的 STOP 位持续时间可能会更长； 0：1 个停止位 1：1.5/2 个停止位
1:0	M	RW	2'b0	数据长度选择。 仅当 UART 不忙时可写。当 CR3 中的 M_E 设置为 0 时，此寄存器用于选择外围设备将发送和接收的每个字符的数据位数。 00：每个字符 5 个数据位 01：每个字符 6 个数据位 10：每个字符 7 个数据位 11：每个字符 8 个数据位

38.4.5 UART CR2, Address=0X18(CR2)

地址偏移：0x18

复位值：0x00000008

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

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3	BUSYERRIE	RW	1’b1	启用 BUSYERR 状态中断。 这用于启用/禁用 BUSYERR 状态中断生成。UART 繁忙时，尝试写入 CR1 寄存器。优先级最低=4。 0：禁用 BUSYERR 状态中断 1：启用 BUSYERR 状态中断
2	LSIE	RW	1'b0	启用接收器线路状态中断。 这用于启用/禁用接收器线路状态中断的生成。这是最高优先级的中断。 该中断标志是 PE、FE、OE、BI、ADDR_RCV 中断标志的组合。 0：禁用接收器线路状态中断 1：启用接收器线路状态中断
1	TDREIE	RW	1'b0	启用传输保持寄存器空中断。 这用于启用/禁用传输保持寄存器空中断的生成。这是优先级第三高的中断。 0：禁用传输空中断 1：启用传输空中断
0	RXNEIE	RW	1'b0	启用接收数据可用中断。这用于启用/禁用接收数据可用中断。这些是优先级第二高的中断。 0：禁用接收数据中断 1：启用接收数据中断

38.4.6 UART CR3, Address=0X1C(CR3)

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
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	DMA_SOFT_ACK	DMAT	DMAR	TX_MODE	SEND_ADDR	ADDR_MATCH	M_E
-									RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:7	保留	-	-	-
6	DMA_SOFT_ACK	RW	1’b0	软件清除 dma 发送请求和 dma 接收请求。 0：不影响 1：清除 dma 发送请求和 dma 接收请求
5	DMAT	RW	1'b0	传送时使能 DMA。 0：禁止；

				1：传送时使能 DMA；
4	DMAR	RW	1'b0	接收时使能 DMA。 0：禁止； 1：接收时使能 DMA；
3	TX_MODE	RW	1'b0	传输模式控制位。该比特用于控制 9-bit 数据传输期间的传输模式的类型。 1：在这种操作模式下，传输保持寄存器为 9 位宽。用户需要确保 TDR 寄存器的地址/数据写入正确。 地址：第 9 位设置为 1 数据：第 9 位数设置为 0 注意：传输地址寄存器（TAR）不适用于此操作模式。 0：在此操作模式中，传输保持寄存器的宽度为 8 位。用户需要将地址编程到传输地址寄存器中，并将数据编程到 TDR 寄存器中。SEND_ADDR 位用作控制旋钮，指示 UART 何时发送地址。
2	SEND_ADDR	RW	1'b0	发送地址控制位。 此位用作控制旋钮，供用户在传输模式下确定何时发送地址。 注：1.在发出地址字符后，该位由硬件自动清除。用户不应将此位编程为 0。 2.此字段仅在 M_E 位设置为 1 且 TX_MODE 设置为 0 时适用。 0：9 位字符内容来自 TDR 寄存器 1：9 位字符内容：第 9 位设置为 1，其余 8 位将与“传输地址寄存器”中正在编程的内容相匹配。
1	ADDR_MATCH	RW	1'b0	地址匹配模式。 此位用于在接收期间启用地址匹配功能。 在地址匹配模式下，UART 将等待第 9 位设置为 1 的传入字符。并进一步检查地址是否与“接收地址匹配寄存器”中编程的地址匹配。如果匹配，则后续字符将被视为有效数据，UART 开始接收数据。 在正常模式下，UART 将开始接收数据，9 位字符将形成。用户负责读取数据并区分 b/n 地址和数据。注意: 仅当 M_E 设置为 1 时，此字段才适用。数值： 0：正常模式 1：地址匹配模式
0	M_E	RW	1'b0	使能 9bit 此位用于启用用于发送和接收传输的 9 位数据

38.4.7 UART RAR, Address=0X20(RAR)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	RAR							
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	RAR	RW	8'b0	这是接收模式下的地址匹配寄存器。 如果在输入字符中第 9 位被设置为 1，则将对照该寄存器值检查剩余的 8 位。如果匹配成功，则第 9 位设置为 0 的后续字符将被视为数据字节，直到接收到下一个地址字节。 注：-仅当“ADDR_MATCH”和“M_E”位设置为 1 时，此寄存器才适用。RAR 可以在任何时间点被编程。但是，当任何接收正在进行时，用户不得更改此寄存器值。

38.4.8 UART TAR, Address=0X24(TAR)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TAR							
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:8	保留	-	-	-
7:0	TAR	RW	8'b0	这是传输模式下的地址匹配寄存器。 如果 M_E 位被启用，并且“SEND_ADDR”位被设置为 1，则 UART 将发送第 9 位设置为 1 的 9 位字符，剩余的 8 位地址将从该寄存器发送。 注：此寄存器仅用于发送地址。正常数据应通过编程 TDR 寄存器发送。在 UART 串行通道上开始发送地址后，硬件将自动清除“SEND_ADDR”位

38.4.9 UART BRRF, Address=0X28(BRRF)

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BRRF			
-												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3:0	BRRF	RW	4'b0	波特率小数部分。 小数值由（BRRF）/（2^4）确定。

39. 低功耗通用异步收发器（LPUART）

LPUART 是一种 UART，允许在有限功耗下双向 UART 通信。仅需 32.768 kHz LSE 时钟即可进行高达 9600 波特/s 的 UART 通信。当 LPUART 由与 LSE 时钟不同的时钟源驱动时，可以达到更高的波特率。即使当微控制器处于低功耗模式，能耗极低时，LPUART 也会等待 UART 帧的到来。LPUART 包含所有必要的硬件支持，使在最小功耗下可以进行异步串行通信。

它支持半双工单线通信和调制解调器操作(CTS/RTS)，还支持多处理器通信。

DMA（直接存储器访问）可用于数据发送/接收。

39.1 LPUART 主要特性

- AMBA APB 接口
- 全双工异步通讯
- NRZ 标准模式（标记/空格）
- 波特率可编程
- 32.768 kHz 时钟，波特率范围 300~9600 波特/s.更高波特率需要更高时钟频率支持
- 双时钟域：PCLK 及专用内核时钟
- 数据字长度可配置（7 位/8 位/9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位数可配置（1/2 位）
- 单线半双工通讯
- 支持 DMA 连续传输
- 传送和接收独立使能
- 独立发送/接收信号极性控制
- Tx/Rx 引脚可以互换
- 支持硬件 RS-485/modem 流控制
- 发送检测标志
  - 忙标志
  - 传输结束
- 奇偶校验控制
  - 发送时产生奇偶校验位
  - 接收时奇偶校验
- 4 个错误标志
  - 上溢错误
  - 噪声检测
  - 帧结构错误
  - 奇偶校验错误
- 中断源标志
  - CTS 改变
  - 发送寄存器空
  - 发送完成

- 接收数据寄存器非空
- 检测到总线空闲
- 溢出错误
- 帧错误
- 噪音操作
- 校验错误
- 地址字节匹配

- 多处理器通讯：当地址不匹配时 LPUART 进入静默模式
- 支持静默模式唤醒（空闲帧/地址标记检测）

39.2 LPUART 功能描述

39.2.1 LPUART 框图

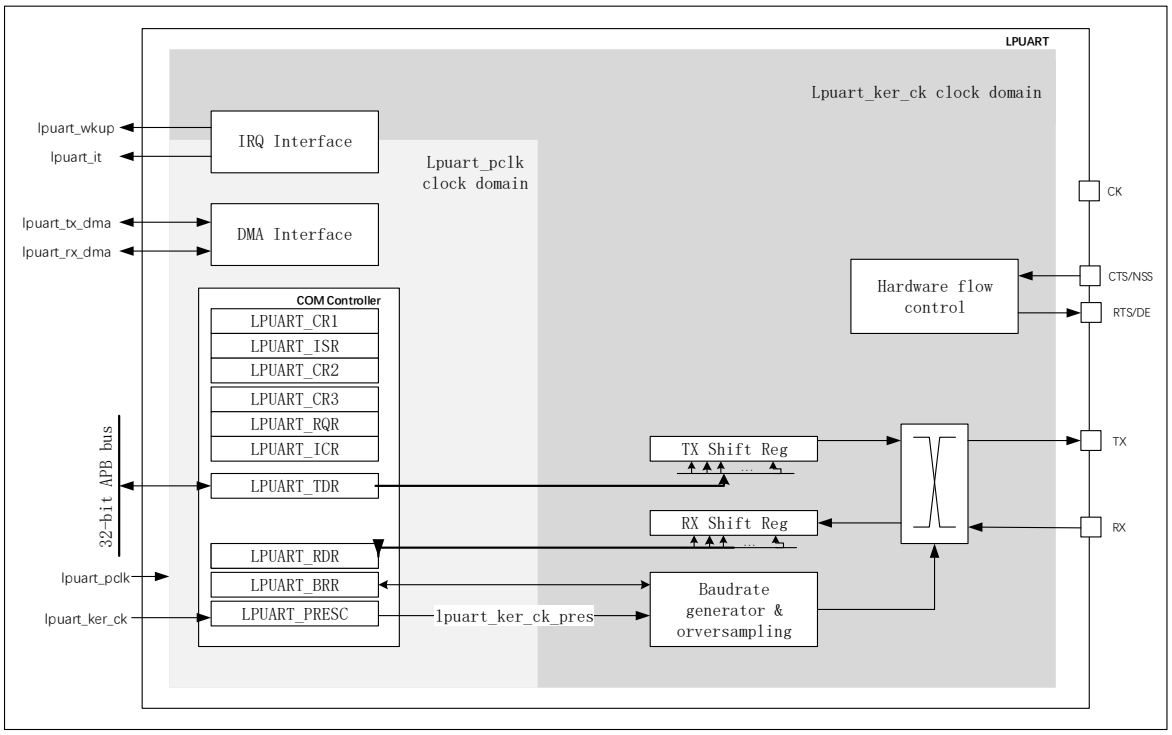


图 39-1 LPUART 框图

39.2.2 LPUART 信号

任何 LPUART 双向通信至少需要两个引脚：接收数据输入(RX)和发送数据输出(TX)。

- RX：接收数据串行输入。
- TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器使能，但不发送数据时，TX 引脚处于高电平。在单线模式里，此 I/O 口被同时用于数据的发送和接收。

硬件流控制模式

- CTS：低电平开始发送，发送结束变为高电平
- RTS：低电平表明 LPUART 准备好接收数据

RS485 硬件流控制模式

- DE：驱动器使能，激活外部收发器的发送

注：DE 和 RTS 复用同一引脚

39.2.3 LPUART 字符描述

根据{M1，M0}配置，分为 7 位/8 位/9 位三种数据长度。

M[1:0]= 2’b10，7 位；

M[1:0]= 2’b00：8 位；

M[1:0]= 2’b01：9 位；

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

空闲字符定义：在 7 位/8 位/9 位数据长度+停止位时间内为全 1；

断开字符定义：在 7 位/8 位/9 位数据长度时间内为全 0；断开帧结尾，传送器插入 2 个停止位。即 8 位模式，断开帧长度为 10 位 0；9 位模式，断开帧长度为 11 位 0.

发送和接收操作由通用波特率发生器驱动。发送器和接收器的使能位置 1 时，将分别生成发送时钟和接收时钟。

下面是各个不同长度帧的波形。

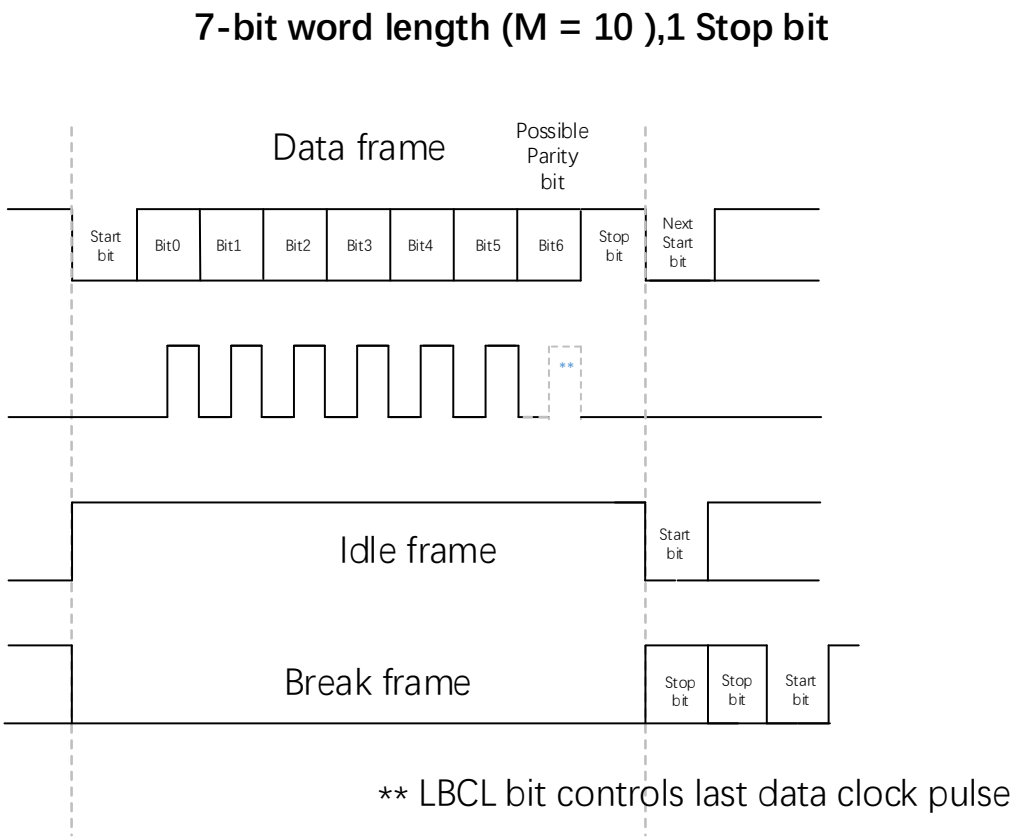


图 39-2 LPUART 字长编程

39.2.4 LPUART 发送

LPUART 发送步骤如下所述。

- 配置 LPUART_CR1.{M1,M0}，定义传输数据宽度（7/8/9 位）；

- 配置 LPUART_BRR 寄存器，选择波特率；
- 配置 LPUART_CR2.STOP 寄存器，定义停止位个数(1/2 位);
- 配置 LPUART_CR1.UE=1,使能 LPUART;
- 多处理器传输时，配置 LPUART_CR3.DMAT=1，使能 DMA;
- 配置 TE=1，硬件插入空闲帧时序;
- 写传输数据到 LPUART_TDR 寄存器（该操作清零 TXE 位）。该项操作可能重复多次。
- 写最后一个数据到 LPUART_TDR 后，软件等待 TC=1，表明最后一帧发送完成。
- 硬件产生发送时序：
  - 产生起始位，开始传送
  - 传送移位寄存器输出数据（默认 LSB）到 TX 引脚
  - 停止位（根据配置发送个数）
- 传送完最后 1 帧数据后，产生 TC=1（TXE=1 时）。当 TCIE=1，产生 TC 中断。

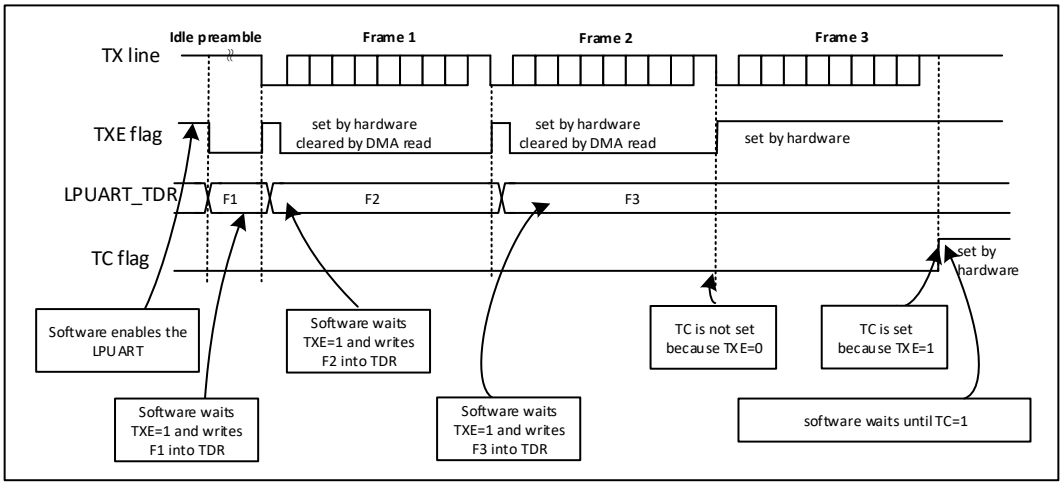


图 39-3 发送时的 TC/TXE 行为 TC/TXE 时序

特殊帧

断开帧

设置 SBKRQ=1 将会在当前传输完成后传送一个断开帧。断开帧的长度取决于{M1，M0}寄存器的配置。断开帧发送完毕后，硬件会清零 SBKRQ。

空闲帧

设置 TE=1，在第一个数据帧之前先发送一个空闲帧。

LPUART 接收

39.2.4.1 起始位检测

在 RX 上检测到下降沿后，硬件开始检测起始位。在该位的中间检测到 0，则判定该位为起始位。如果检测到的是 1，则置位噪声错误标志（NE），当前起始位无效，重新检测起始位。

39.2.4.2 数据字符接收

字符接收配置步骤：

- 配置 LPUART_CR1.{M1，M0}，选择接收长度；
- 配置 LPUART_BRR 寄存器，选择波特率；

- 配置 USAR_CR2.STOP，选择 1 位或者 2 位停止位；
- 配置 LPUART_CR1.UE=1，使能 LPUART；
- 多处理器传输时，配置 LPUART_CR3.DMAR=1，使能 DMA 接收;
- 配置 LPUART_CR1.RE，使能接收，开始检测起始位；

当接收到字符后：

- 非 FIFO 模式，置位 RXNE。表明移位寄存器的内容已被转移到 RDR，即数据可以被读出；
- 当 RXNEIE=1 时，产生中断；
- 在接收时，如果检测到帧错误、噪声错误或溢出错误，则设置相应错误标志位；
- 多处理器通讯：
  - 非 FIFO 模式，RXNE 在每个字节接收后被置位，并由 DMA 对接收数据寄存器的读操作而清零；
- 在单缓冲器模式：
  - 非 FIFO 模式，软件读 LPUART_RDR 寄存器对 RXNE 位清零。RXNE 标志也可以通过写 LPUART_RQR.RXFRQ=1 清零。RXNE 位必须在下一字符接收结束前被清零，以免溢出错误；

39.2.4.3 特殊帧及错误处理

-断开字符

当接收到一个断开帧时，LPUART 按错误帧处理。

-空闲字符

当检测到空闲帧时，按正常帧处理，如果 IDLEIE=1，则产生中断。

-上溢错误

如果 RXNE 没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。每接收到 1 个字节后置位 RXNE 标志。如果下一个数据已收到或先前 DMA 请求还没执行时，RXNE 标志仍为置起状态，溢出错误产生。

溢出错误产生时，执行如下错误处理：

- 置位 ORE；
- RDR 内容不会丢失。读 LPUART_DR 寄存器仍可正常读出先前数据；
- 移位寄存器内容被覆盖，随后接收到的数据丢失；
- 如果 RXNEIE=1，或者 EIE=1，则产生中断；
- 写 ORECF=1 清零 ORE 位；

ORE 置位，表明至少 1 个数据已经丢失。有如下两种情况：

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 中，可以被读出；
- 如果 RXNE=0，上一个有效数据已经被读走，RDR 无有效数据可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（被丢失）的数据时，此种情况可能发生。

39.2.4.4 时钟源选择

LPUART 的时钟源由 RCC_CCIPR.LPUARTxSEL 寄存器配置。在配置 LPUART_CR1.UE=1 前，必须配置时钟源。

时钟源的选择决定了通讯速度范围。

LPUART 双时钟域，一个为内核时钟（lpuart_ker_ck），一个为系统时钟（lpuart_pclk）。当用作从低功耗唤醒时，内核时钟源由 RCC_CCIPR.LPUARTxSEL 寄存器配置。

当不支持双时钟域时，内核时钟与系统时钟相同。

当 MCU 进入低功耗模式，内核时钟源选择低频时，LPUART 可以接收数据并唤醒 MCU，然后软件或者 DMA 可以读取 LPUART_RDR 中数据。

39.2.4.5 帧错误

接收时没有检测到停止位，从而出现同步失效或大量噪声时，会产生帧错误。

帧错误时做如下处理：

- 硬件设置 FE 位；
- 无效数据从移位寄存器传送到 LPUART_RDR 寄存器；
- 在单字节通讯情况下，不产生中断。但因为 NE 标志位和 RXNE 标志位时同时被设置，FE 位出现上升沿。在多处理器通讯情况下，如果 LPUART_CR3.EIE=1，将产生一个中断。
- 写 LPUART_ICR.FECF=1 可以清零 FE 标志。

39.2.4.6 配置停止位个数

可配置位 1/2 个停止位：

- 1 停止位：在第 8/9/10 采样点采样停止位；
- 2 停止位：在第一个停止位采样后采样第 2 个停止位。在检测到第 2 个停止位结束后设置 RXNE 和 FE 标志。第 1 个停止位不会检测帧错误。

39.2.5 LPUART 波特率产生

发送和接收配置相同波特率。

$$\text{Tx/Rx 波特率} = \frac{256\ X\ lpuartckpres}{LPUARTDIV}$$

LPUARTDIV 在 LPUART_BRR 寄存器配置。在通讯进行时不要改变波特率的值。LPUART_BRR 寄存器值不能低于 0x300.

fCK 的频率范围为 3 ~ 4096 倍*波特率。

LPUART 时钟源为 LSE 时，最大波特率为 9600 波特率。当 LPUART 由不同于 LSE 时钟的时钟源进行时钟时，可以达到更高的波特率。例如，LPUART 时钟源频率为 100MHz，可达到的最大波特率约为 33M 波特。

表 39-1 lpuart_ker_ck_pres = 32,768 KHz 时编程波特率的误差计算

Baud rate	**lpuart_ker_ck_pres = 32,768 KHz**		
期望值	**实际值**	**在波特率寄存器中编程的值**	**误差%**
0.3 KBps	0.3 KBps	0x6D3A	0
0.6 KBps	0.6 KBps	0x369D	0
1200 Bps	1200.087 Bps	0x1B4E	0.007
2400 Bps	2400.17 Bps	0xDA7	0.007
4800 Bps	4801.72 Bps	0x6D3	0.035
9600 Bps	9608.94 Bps	0x369	0.035

表 39-2 lpuart_ker_ck_pres = 100 MHz 时编程波特率的误差计算

Baud rate	**lpuart_ker_ck_pres = 100 MHz**		
期望值	**实际值**	**在波特率寄存器中编程的值**	**误差%**
38400 波特	38400.04 波特	A2C2A	0.0001
57600 波特	38400.04 波特	6C81C	0.0001
115200 波特	115200.12 波特	3640E	0.0001
230400 波特	230400.23 波特	1B207	0.0001
460800 波特	460804.61 波特	D903	0.001
921600 波特	921625.81 波特	6C81	0.0028
4000000 波特	4000000.00 波特	1900	0
10000000 波特	10000000.00 波特	A00	0
20000000 波特	20000000.00 波特	500	0
33000000 波特	33032258.06 波特	307	0.1

39.2.6 LPUART 接收容忍度

只有当整体的时钟系统地变化小于 LPUART 异步接收器能够容忍的范围，LPUART 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+DWU < LPUART 接收器的容忍度

其中 DWU 是使用低功耗模式唤醒时由于采样点偏差引起的误差。

表 39-3 LPUART 接收器可以在指定的最大容忍偏差下正确接收数据（理论值）

{M1,M0}位	**768 < BRR < 1024**	**1024 < BRR < 2048**	**2048 < BRR < 4096**	**4096 ≤ BRR**
8 位({M1,M0} =00), 1 停止位	1.82%	2.56%	3.90%	4.42%
9 位({M1,M0} =01), 1 停止位	1.69%	2.33%	2.53%	4.14%
7 位({M1,M0} =10), 1 停止位	2.08%	2.86%	4.35%	4.42%
8 位({M1,M0} =00), 2 停止位	2.08%	2.86%	4.35%	4.42%
9 位({M1,M0} =01), 2 停止位	1.82%	2.56%	3.90%	4.42%
7 位({M1,M0} =10), 2 停止位	2.34%	3.23%	4.92%	4.42%

注：当接收到的帧中包含恰好是 10/11/9 位的空闲帧（对应 M= ‘b00/11/10’ ）时，表中指定的数据在特殊情况下可能会略有不同。

39.2.7 LPUART 多处理器通信

通过 LPUART 可以实现多处理器通讯（将几个 LPUART 连在一个网络里）。例如某个 LPUART 设备可以是主，其 TX 输出和其他 LPUART 从机的 RX 输入相连接；LPUART 从机各自的 TX 输出逻辑与在一起，并且和主机的 RX 输入相连接。

未被寻址的设备可以配置 MME 寄存器置于静默模式。在静默模式：

- 不会设置任何接收状态位；
- 禁止所有接收中断；

- LPUART_ISR.RWU=1.通过配置 LPUART_RQR.MMRQ，硬件和软件可以控制 LPUART 进入静默模式；

根据 LPUART_CR1.WAKE 的值，LPUART 可以用两种方式进入或退出静默模式：

- WAKE=0：空闲总线检测；
- WAKE=1：地址标志检测；

39.2.7.1 空闲总线检测

当 MMRQ 写 1，RMU 自动置 1 后，LPUART 进入静默模式。

当检测到空闲帧时，LPUART 被唤醒。然后 RWU 被硬件清零，但是 LPUART_ISR.IDLE 并不置起。

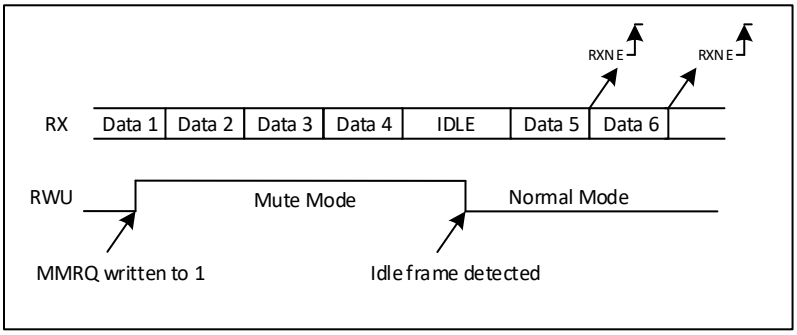


图 39-4 用总线帧检测的静默模式

注：

如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。

如果在线路处于空闲状态时激活 LPUART，在一个 IDLE 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

39.2.7.2 4 位/7 位地址标志检测

在这个模式，如果 MSB=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 个或者 7 个 LSB 中（由 ADDM7 寄存器选择几个）。这个 4 位/7 位地址被接收器同 LPUART_CR2 寄存器中 ADD 中配置的地址比较。

如果接收到的字节与配置地址不匹配，LPUART 进入静默模式。此时，硬件设置 RWU。接收该字节既不会设置 RXNE 标志也不会产生中断或发出 DMA 请求，因为 LPUART 已经在静默模式。

当 MMRQ 配置为 1 时，LPUART 也会进入静默模式，并设置 RWU=1.

如果接收到的字节与配置地址匹配，LPUART 退出静默模式。此时，清零 RWU，随后正常接收数据。因为 RWU 已被清零，此种情况会设置 RXNE/RXFNE 位。

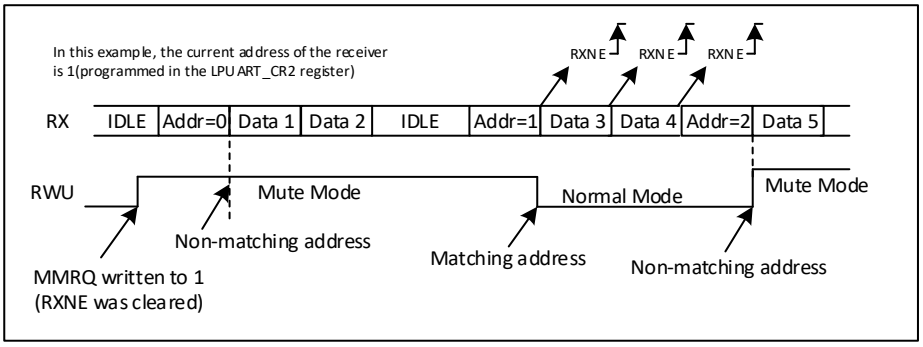


图 39-5 用地址标志检测的静默模式

39.2.8 LPUART 奇偶校验

设置 LPUART_CR1.PCE=1，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{M1,M0}位定义的帧长度，支持的 LPUART 帧格式如下表所示：

表 39-4 支持的 LPUART 帧格式

{M1,M0}位	PCE 位	LPUART 帧
00	0	| 开始位 | 8 位数据 | 停止位 |
00	1	|开始位| 7 位数据| 奇偶校验位 |停止位|
01	0	|开始位| 9 位数据|停止位|
01	1	|开始位| 8 位数据|奇偶校验位|停止位|
10	0	|开始位| 7 位数据|停止位|
10	1	|开始位| 6 位数据|奇偶校验位|停止位|

39.2.8.1 偶校验

偶校验位使得一帧中的 6 或 7 或 8 个 LSB 数据以及校验位中“1”的个数为偶数。

39.2.8.2 奇校验

奇校验位使得一帧中的 6 或 7 或 8 个 LSB 数据以及校验位中“1”的个数为奇数。

39.2.8.3 接收校验

奇偶校验错误时设置 PE=1，当 PEIE=1 时产生中断。软件写 PECF=1 清零 PE。

39.2.8.4 发送校验位

发送时 MSB 由奇偶校验位代替。

39.2.9 LPUART 单线半双工通信

当配置 LPUART_CR3.HDSEL=1 时，LPUART 进入单线半双工通讯模式。

在此模式下：

- TX 和 RX 在模块内部连接；
- RX 引脚不用；
- 当无数据传输时释放 TX 引脚作为标准 I/O

39.2.10 DMA 连续通信

LPUART 可以利用 DMA 连续通讯。接收缓冲器和发送缓冲器的 DMA 请求分别产生。

39.2.10.1 利用 DMA 发送

通过配置 LPUART_CR3.DMAT 使能利用 DMA 发送。当 TXE=1，DMA 就从指定的 SRAM 区传送数据到 LPUART_DR 寄存器。

DMA 发送的配置步骤如下：

- 配置 SYSCFG_CFGR3 或者 SYSCFG_CFGR4 选择 LPUART 使用的 DMA 通道

- 配置 LPUART_TDR 寄存器地址作为 DMA 传输的目标地址。在每个 TXE 事件后，数据将被传送到这个地址；
- 配置存储器（如 SRAM）地址作为 DMA 传输的源地址。在每次 TXE 事件后，将从此存储器读数据加载到 LPUART_TDR 寄存器；
- 写 DMA 控制寄存器配置传输字节数；
- 写 DMA 寄存器配置通道优先级；
- 根据应用，配置在传输完成一半还是全部完成时产生 DMA 中断；
- 写 TCCF=1 清零 LPUART_ISR.TC 标志；
- 使能 DMA 通道；

当传输达到配置的传输字节数，DMA 控制器产生中断请求。

当 DMA 写完所有要传输数据后，设置 TCIF=1.然后软件需要等待 TC=1，表明 LPUART 已经发送完最后一帧数据。

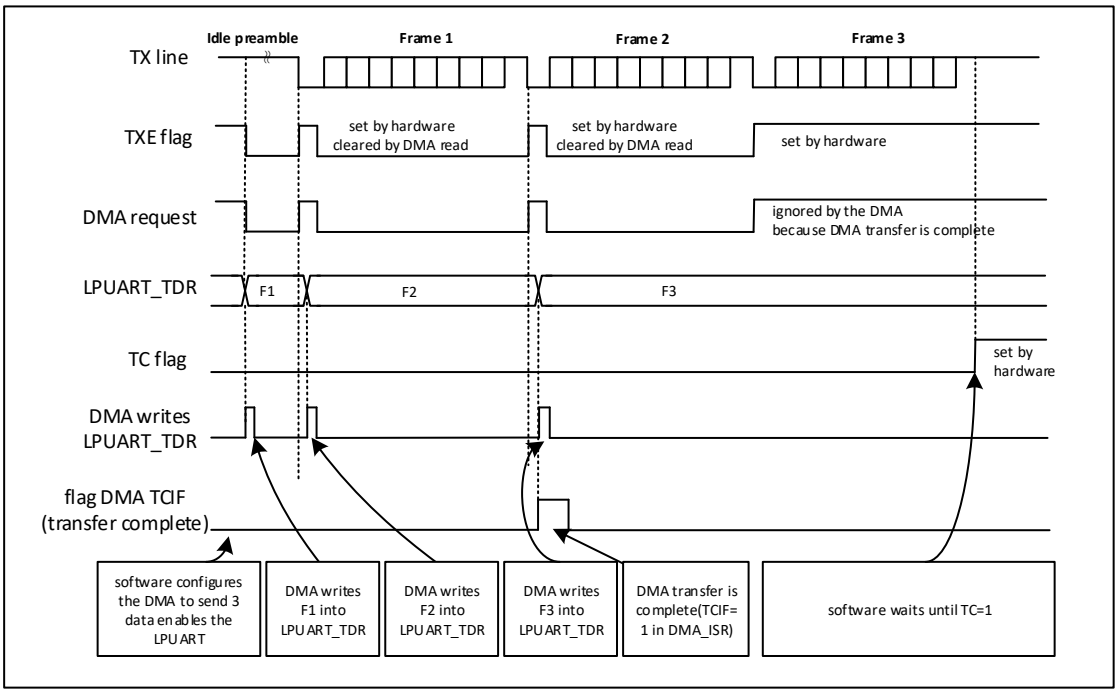


图 39-6 LPUART 用 DMA 发送

39.2.10.2 利用 DMA 接收

通过配置 LPUART_CR3.DMAR 使能利用 DMA 接收。每收到一个字节，DMA 就把数据从 LPUART_RDR 寄存器传送到指定的 SRAM 区。

DMA 接收的配置步骤如下：

- 配置 SYSCFG_CFGR3 或者 SYSCFG_CFGR4 选择 LPUART 使用的 DMA 通道
- 配置 LPUART_RDR 寄存器地址作为 DMA 传输的源地址。在每个 RXNE 事件后，数据将从此 LPUART_RDR 寄存器地址读出数据传输到存储器；
- 配置 SRAM 存储器地址作为 DMA 传输的目标地址。在每次 RXNE 事件后，数据将从 LPUART_RDR 寄存器传送到此地址；
- 写 DMA 控制寄存器配置传输字节数；

- 写 DMA 寄存器配置通道优先级；
- 根据应用，配置在传输完成一半还是全部完成时产生 DMA 中断；
- 使能 DMA 通道；

当传输达到配置的传输字节数，DMA 控制器产生中断请求。

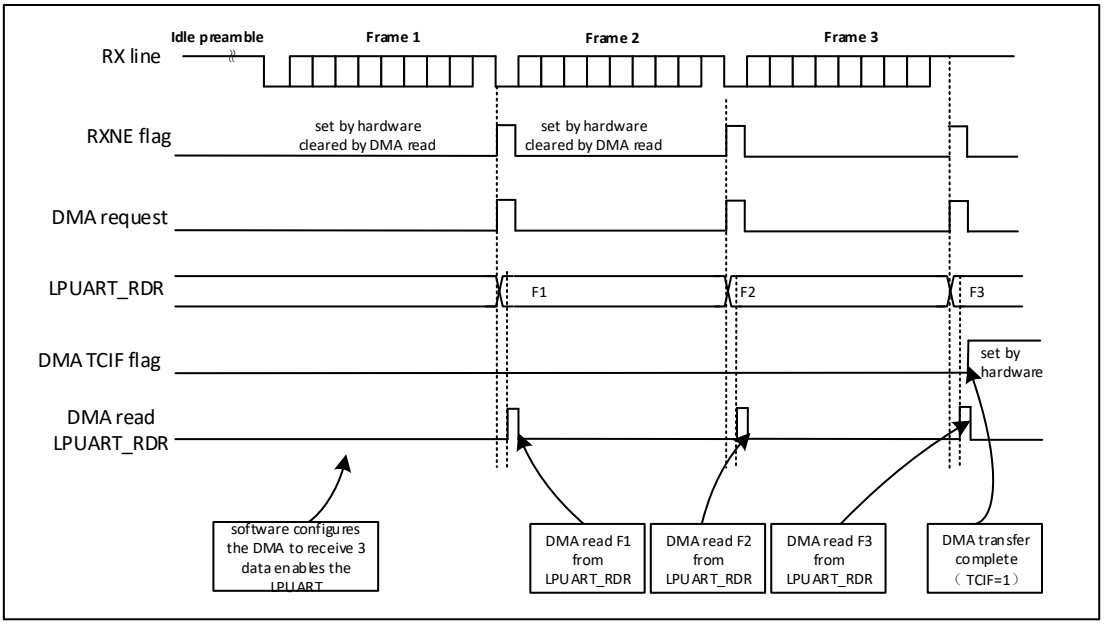


图 39-7 LPUART 用 DMA 接收

39.2.10.3 错误标志和中断

在多处理器通讯模式，在传输时产生任何错误都会在当前字节传送完成后产生错误标志，如果对应中断使能，则产生中断。

39.2.11 RS232 硬件流控制和 RS485 驱动器使能

通过 CTS 和 RTS 可以控制两个 LPUART 模块的串行数据流。

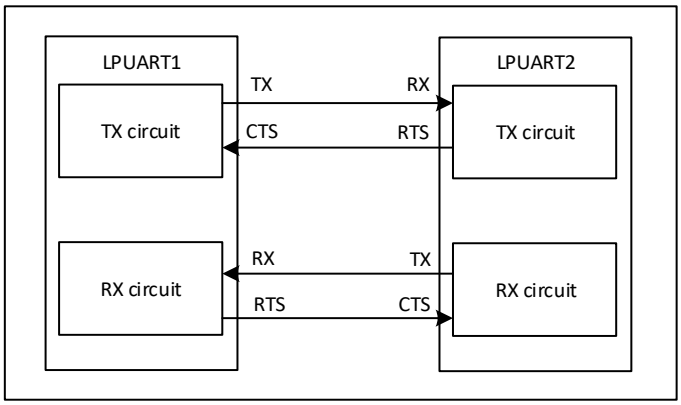


图 39-8 硬件流控制

39.2.11.1 RS232 RTS 流控制

通过配置 LPUART_CR3.RTSE=1，使能 RTS 流控制。当 LPUART 准备好接收数据时，将 RTS 拉低。当接收寄存器满时，RTS 拉高（无效），表明当前帧是最后一帧。

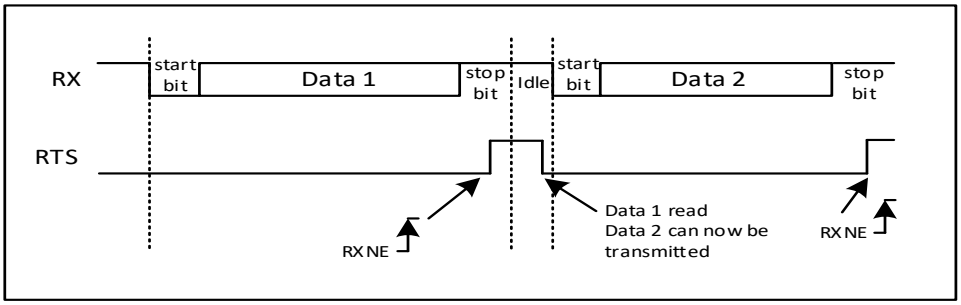


图 39-9 RTS 流控制

39.2.11.2 RS232 CTS 流控制

通过配置 LPUART_CR3.CTSE=1，使能 CTS 流控制。使能后，在传送下一帧前检测 CTS 信号。如果 CTS=0（有效），传送下一笔数据（假设有数据需要传输）。如果 CTS=1（无效），则当前传输完成（在 stop bit 位前完成）。

当 CTSE=1，只要 CTS 信号反转，硬件就置位 CTSIF。表明接收器准备好通讯或未准备好通讯。当 CTSIE=1 时，产生中断。

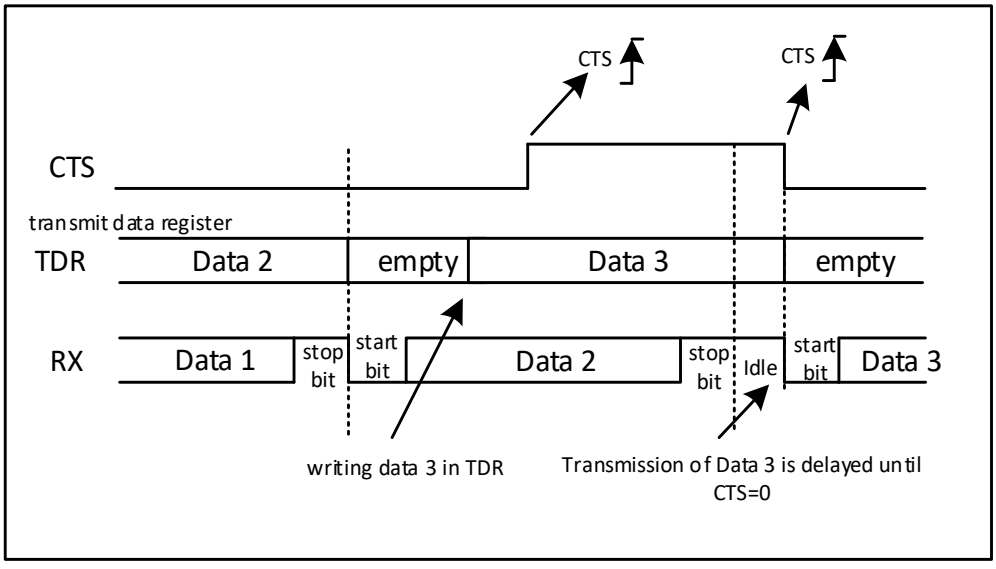


图 39-10 CTS 流控制

注：CTS 在当前帧结束前至少维持 3 个 LPUART 源时钟的高电平。CTSCF 也需要维持至少 2xPCLK 周期。

39.2.11.3 RS485 驱动使能

配置 LPUART_CR3.DEM=1，使能 DE 功能。通过 DE 信号，用户可以激活外部传输控制。DE 信号的有效极性通过 LPUART_CR3.DEP 寄存器配置。

有效时间由 LPUART_CR1.DEAT 配置，为 DE 信号有效沿与起始位开始之间的时间。

释放时间由 LPUART_CR1.DEDT 配置，为最后一个停止位与 DE 无效之间的时间。

具体时间计算如下所示，其中 P=BRR[20:11]:

- DE 插入时间：
  - （1+（DEAT x P））x f_{CK}，P≠0
  - （1+DEAT）x fCK，P=0
- DE 释放时间：
  - （1+（DEDT x P））x f_{CK}，P≠0
  - （1+DEDT）x fCK，P=0

39.2.12 LPUART 低功耗模式

LPUART 支持低功耗模式，使得在 LPUART PCLK 不存在时仍能传输数据。

当 UESM=1，LPUART 能在低功耗模式下唤醒 MCU。

39.2.12.1 根据 WUS 产生中断唤醒请求

当检测到唤醒事件，硬件产生 WUF 标志。如果 WUFIE=1 时，产生 WUF 中断。在此情况下，WUF 标志置 1 便足以将 MCU 从低功耗模式唤醒。

注：

- 进入低功耗模式前，保证没有 LPUART 传输；
- 不管 MCU 是在低功耗模式还是工作模式，只要检测到唤醒事件，就会置位 WUF 标志；
- 当初始化和使能 LPUART 接收后，必须要通过 REACK 位确认 LPUART 是否真正使能，然后再配置进入低功耗模式；
- 当使能 DMA 接收，在进入低功耗模式前必须关闭 DMA 接收；然后从低功耗唤醒后再重新使能 DMA；

39.2.12.2 从静默模式进入低功耗模式

如果 LPUART 在进入低功耗模式前处于静默模式：

- 不能用空闲帧唤醒静默模式，因为在低功耗模式下不支持空闲帧检测；
- 用地址匹配的方式唤醒静默模式，则从低功耗模式唤醒也要用地址匹配的方法。当进入低功耗时，如果 RXNE=1，则 LPUART 将不会进入低功耗模式，而维持在静默模式；

39.2.12.3 低功耗模式时内核时钟关闭

如果低功耗模式下，内核时钟也关闭，则 LPUART 在 RX 线上检测到下降沿时，会产生 lpuart_ker_ck_req 信号去使能内核时钟。

如果唤醒事件 LPUART_CR3.WUS 配置为地址匹配方式：

- 内核时钟使能后，接收完一帧后如果地址匹配，则 MCU 唤醒，开始正常数据接收；

下图为唤醒事件后地址验证匹配的情况：

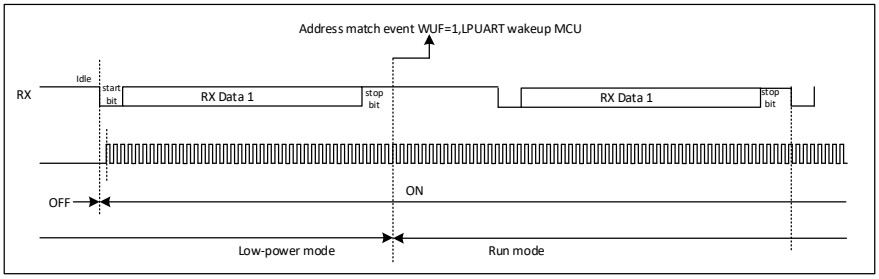


图 39-11 LPUART 地址匹配唤醒

如果唤醒事件后地址验证不匹配，则内核时钟再次关闭，MCU 不会被唤醒，系统仍在低功耗模式。

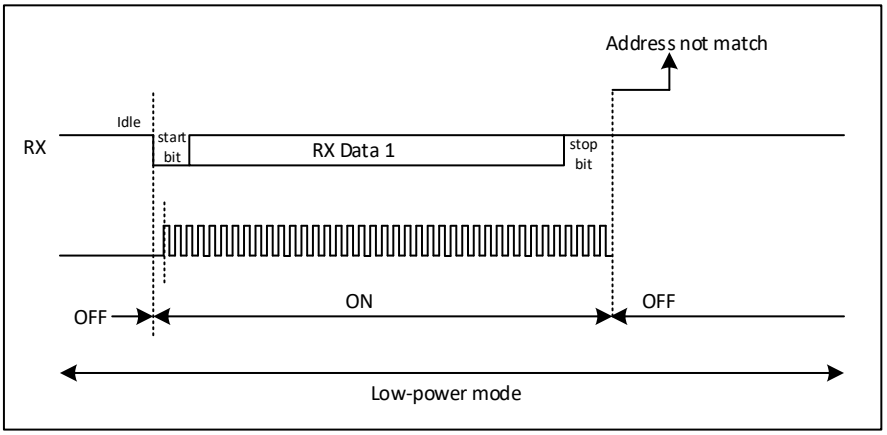


图 39-12 LPUART 地址不匹配无法唤醒

39.3 LPUART 中断

中断事件	事件标志	使能位	中断清零	中断信号是否有效	
				中断	唤醒
发送数据寄存器空	TXE	TXEIE	写数据到 TDR	YES	NO
CTS 中断	CTSIF	CTSIE	软件配置 CTSCF=1	YES	NO
传送完成	TC	TCIE	写数据到 TDR 或 TCCF=1	YES	NO
接收寄存器未空（读数据准备好）	RXNE（对应标志位 RXFNE）	RXNEIE（对应中断使能 RXFNEIE）	读 RDR 寄存器或 RXFRQ=1	YES	YES
溢出错误	ORE	RXNEIE/RXFNEIE	ORECF=1	YES	NO
空闲帧	IDLE	IDLEIE	IDLECF=1	YES	NO
奇偶校验错误	PE	PEIE	PECF=1	YES	NO
多处理器通讯时，噪声、overrun 和帧错误	NR/ORE/FE	EIE	NECF=1 清零 NE；ORECF=1 清零 ORE；	YES	NO

			FECF=1 清零 FE；		
地址字节匹配	CMF	CMIE	CMCF=1	YES	NO
从低功耗模式唤醒	WUF	WUFIE	WUCF=1	YES	YES

39.4 LPUART 寄存器

39.4.1 LPUART 控制寄存器 1（LPUART_CR1）

Address offset：0x00

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
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
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	RXFFIE	RW	0	RXFIFO 满中断使能。 0：中断禁止； 1：RXFF 中断使能； 只能在 FIFO 使能时（FIFOEN=1）才能写入该位
30	TXFEIE	RW	0	TXFIFO 空中断使能。 0：中断禁止； 1：TXFE 中断使能； 只能在 FIFO 使能时（FIFOEN=1）才能写入该位
29	FIFOEN	RW	0	FIFO 模式使能。 0：FIFO 模式禁止； 1：FIFO 模式使能； 只能在 LPUART 未使能时（UE=0）才能写入该位
28	M1	RW	0	{M1,M0}的值配置长度，由软件置位或者清零。 2’b00：1 起始位，8 数据位，n 停止位； 2’b01：1 起始位，9 数据位，n 停止位； 2’b10：1 起始位，7 数据位，n 停止位； 只能在 LPUART 未使能时（UE=0）才能写入该位。
27:26	保留	-	-	-
25:21	DEAT[4:0]	RW	5’h0	该寄存器配置驱动使能（DE）信号有效和起始位之间的时间。用采样时间作为单位。 如果不支持 DE 功能，则该位保留。 只能在 LPUART 未使能时（UE=0）才能写入该位
20:16	DEDT[4:0]	RW	5’h0	该寄存器配置发送帧停止位与 DE 信号无效之间的时间。用采样时间作为单位（1/8 或者 1/16 位周期，取决于过采样率）。如果 DEDT 时间内有写 LPUART_TDR 寄存器请求发送，则数据只有当 DEDT 和 DEAT 都结束后才发送。 如果不支持 DE 功能，则该位保留。 只能在 LPUART 未使能时（UE=0）才能写入该位
15	保留	-	-	-

14	CMIE	RW	0	字符匹配中断使能。该位由软件置位或者清零。 0：中断禁止； 1：CMF 中断使能；
13	MME	RW	0	静音模式使能。 0：接收器工作在工作模式； 1：接收器在工作模式和静默模式之间切换；
12	M0	RW	0	与 M1 结合配置字长度。 只能在 LPUART 未使能时（UE=0）才能写入该位
11	WAKE	RW	0	接收唤醒方式。 从静默模式唤醒方式。由软件置位或者清零。 0：空闲帧唤醒； 1：地址标志唤醒； 只能在 LPUART 未使能时（UE=0）才能写入该位
10	PCE	RW	0	奇偶校验控制。 0：奇偶校验禁止； 1：奇偶校验使能； 奇偶校验位：9bit 数据的第 9 位；8bit 数据的第 8 位；7bit 数据的第 7 位。 只能在 LPUART 未使能时（UE=0）才能写入该位
9	PS	RW	0	奇偶校验选择。由软件置位和清零。 0：偶校验； 1：奇校验； 只能在 LPUART 未使能时（UE=0）才能写入该位
8	PEIE	RW	0	PE 中断使能。由软件置位和清零。 0：禁止； 1：PE 中断使能；
7	TXEIE	RW	0	传输寄存器空中断使能。由软件置位和清零。 0：禁止； 1：TXE 中断使能；
6	TCIE	RW	0	传送结束中断使能。由软件置位和清零。 0：禁止； 1：TC 中断使能；
5	RXNEIE	RW	0	接收寄存器非空中断使能；由软件置位和清零。 0：禁止； 1：ORE 或者 RXNE/RXFNE 中断使能；
4	IDLEIE	RW	0	IDLE 中断使能。由软件置位和清零。 0：禁止； 1：IDLE 中断使能；
3	TE	RW	0	传送使能。 0：传送禁止； 1：传送使能；
2	RE	RW	0	接收使能。 0：接收禁止； 1：接收使能，开始检测 start 位；
1	UESM	RW	0	停止低功耗模式下 LPUART 唤醒使能。

				该寄存器为 1，如果 LPUART 时钟选择 LSI 或 LSE，LPUART 能唤醒 MCU。 0：LPUART 不能唤醒 MCU； 1：LPUART 模式唤醒 MCU。使能后，LPUART 时钟源选择 LSI 或 LSE。 从低功耗模式唤醒后，该位清零。
0	UE	RW	0	LPUART 使能。 当该位清零后，LPUART 模块会立即停止当前操作。LPUART 的配置会保持，但所有状态标志位，LPUART_ISR 寄存器值会变为默认值。该位由软件置位和清零。 0：LPUART 预分频器和输出禁止，低功耗模式； 1：LPUART 使能； 软件需要等待 LPUART_ISR.TC 置位后，才能清零 UE 位，进入低功耗模式； 同时，在清零 UE 位之前 DMA 通道需要禁止。 该位清零时，LPUART 配置寄存器值维持，但 LPUART_ISR 状态寄存器全部复位。

39.4.2 LPUART 控制寄存器 2（LPUART_CR2）

Address offset：0x04

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.			TXOE_ALWAYS_ON	MSBFIRST	DATAINV	TXINV	RXINV
RW								-			RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.				Res.			ADDM7	Res.	Res.	Res.	Res.
RW	-	RW	RW	-				-			RW	-			

Bit	Name	R/W	Reset Value	Function
31:24	ADD[7:0]	RW	8’b0	LPUART 地址。 该寄存器用于多处理器静默或者停止模式，用作地址唤醒时的地址。传送器发送的 MSB 位为 1. 在正常接收模式时，该寄存器用作字符检测。此时，比对接收内容与 ADD[7:0]相比较，如果匹配，则置位 CMF 标志位。 只有在 RE=0 或者 UE=0 时，该寄存器可以被写。
23:21	保留	-	-	-
20	TXOE_ALWAYS_ON	RW	1	该寄存器用于控制 LPUART 输出使能。 0：硬件自主控制输出使能开/关 1：软件控制输出使能常开 只能在 LPUART 未使能时（UE=0）才能写入该位 注：在半双工模式，需要将此位关闭。
19	MSBFIRST	RW	0	最高有效位在前。 0：起始位后，收发第 0 位数据； 1：起始位后，收发第 7/8/9 位数据；

				只能在 LPUART 未使能时（UE=0）才能写入该位
18	DATAINV	RW	0	二进制数据逆处理。 0: 以正/正向逻辑收发数据寄存器数据。（1=H，0=L） 1: 以反/反向逻辑收发数据寄存器数据。奇偶位也会逆处理。（1=L，0=H） 只能在 LPUART 未使能时（UE=0）才能写入该位
17	TXINV	RW	0	TX 引脚有效电平反相。 0：TX 引脚为标准逻辑电平（V_{CC} =1/空闲, Gnd=0/标记）； 1：TX 引脚信号值取反（V_{CC} =0/标记, Gnd=1/空闲)。 只能在 LPUART 未使能时（UE=0）才能写入该位
16	RXINV	RW	0	RX 引脚有效电平反相。 0：RX 引脚为标准逻辑电平（V_{CC} =1/空闲, Gnd=0/标记）； 1：RX 引脚信号值取反（V_{CC} =0/标记, Gnd=1/空闲)。 只能在 LPUART 未使能时（UE=0）才能写入该位
15	SWAP	RW	0	TX/RX 引脚互换。 0：TX/RX 引脚按照标准引脚排列定义； 1: TX/RX 引脚互换。此时用作交叉连接其他 LPUART 时。 只能在 LPUART 未使能时（UE=0）才能写入该位
14	保留	-	-	-
13:12	STOP[1:0]	RW	2’b0	停止位配置。 00：1 停止位； 01，11：保留； 10：2 停止位； 只能在 LPUART 未使能时（UE=0）才能写入该位
11:5	保留	-	-	-
4	ADDM7	RW	0	7 位/4 位地址检测配置。 0：4 位地址检测； 1：7 位地址检测（8 位数据模式）； 只能在 LPUART 未使能时（UE=0）才能写入该位
3:0	保留	-	-	-

39.4.3 LPUART 控制寄存器 3（LPUART_CR3)

Address offset：0x08

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXTFCFG			Res.	TXFTIE	WUFIE	WUS[1:0]		Res.	Res.	Res.	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
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
RW	RW	RW	RW	-	RW	RW	RW	RW	RW	-	-	RW	-		RW

Bit	Name	R/W	Reset Value	Function
31:29	TXFTCFG[2:0]	RW	0	TXFIFO 阈值配置。 000：TXFIFO 达到总空间 1/8；

				001：TXFIFO 达到总空间 1/4； 110：TXFIFO 达到总空间 1/2； 011：TXFIFO 达到总空间 3/4； 100：TXFIFO 达到总空间 7/8； 101：TXFIFO 空； 其他值：保留；
28	RXFTIE	RW	0	RXFIFO 阈值中断使能。 0：中断禁止； 1：FIFO 达到 RXFTCFG 中配置阈值中断使能。
27:25	RXFTCFG[2:0]	RW	0	RXFIFO 满阈值配置。 000：RXFIFO 达到总空间 1/8； 001：RXFIFO 达到总空间 1/4； 110：RXFIFO 达到总空间 1/2； 011：RXFIFO 达到总空间 3/4； 100：RXFIFO 达到总空间 7/8； 101：RXFIFO 满； 其他值：保留；
24	保留	-	-	-
23	TXFTIE	RW	0	TXFIFO 阈值中断使能。 0：中断禁止； 1：TXFIFO 达到 TXFTCFG 配置阈值中断使能；
22	WUFIE	RW	0	低功耗唤醒中断使能。 0：中断禁止； 1：WUF 中断使能；
21:20	WUS[1:0]	RW	0	低功耗唤醒选择。 00：地址匹配产生 WUF； 01：保留； 10：起始位检测产生 WUF； 11：RXNE 产生 WUF；
19:16	保留	-	-	-
15	DEP	RW	0	DE 极性选择。 0：DE 高电平有效； 1：DE 低电平有效； 只能在 LPUART 未使能时（UE=0）才能写入该位
14	DEM	RW	0	DE 模式使能。 0：禁止； 1：DE 模式使能，DE 信号在 RTS 引脚输出。 用户通过使能该位来控制外部传输。 只能在 LPUART 未使能时（UE=0）才能写入该位
13	DDRE	RW	0	接收错误时是否禁止 DMA。 0：接收错误时不禁止 DMA。相应出错标志置位，但 RXNE=0. 不会产生 DMA 请求，所以出错数据不会输出。 1：接收错误时禁止 DMA。RXNE 和出错标志位都会置位。在出错标志位清零前，屏蔽 DMA 请求。 只能在 LPUART 未使能时（UE=0）才能写入该位

12	OVRDIS	RW	0	溢出禁止。 0：当接收到新数据前已收数据没读时置位溢出出错标志（PRE）； 1: 溢出功能禁止。接收到新数据后，RXNE=1，ORE=0，新数据会覆盖 LPUART_RDR 寄存器中数据。 只能在 LPUART 未使能时（UE=0）才能写入该位
11	保留	-	-	-
10	CTSIE	RW	0	CTS 中断使能。 0：禁止； 1：CTSIF 中断使能；
9	CTSE	RW	0	CTS 使能。 0：CTS 硬件流控制禁止； 1：CTS 模式使能。当有当 CTS 输入为 0 时，才会传输数据。此时，当数据写入数据寄存器后，要等待 CTS 有效后才会启动传输。 只能在 LPUART 未使能时（UE=0）才能写入该位
8	RTSE	RW	0	RTS 使能。 0：RTS 硬件流控制禁止； 1：RTS 输出使能，只有当接收缓冲区未满时才会请求下一个数据。当前数据发送完成后，发送操作暂停。如果可以接收数据了，将 RTS 置为有效（0）。 只能在 LPUART 未使能时（UE=0）才能写入该位
7	DMAT	RW	0	传送时使能 DMA。 0：禁止； 1：传送时使能 DMA；
6	DMAR	RW	0	接收时使能 DMA。 0：禁止； 1：接收时使能 DMA；
5:4	保留	-	-	-
3	HDSEL	RW	0	半双工选择。 0：非半双工模式； 1：半双工模式选择； 只能在 LPUART 未使能时（UE=0）才能写入该位
2:1	保留	-	-	-
0	EIE	RW	0	错误中断使能。 0：禁止； 1：帧错误 FE、溢出错误 ORE、噪声 NF 中断产生；

39.4.4 LPUART 波特率寄存器（LPUART_BRR）

Address offset：0x0C

Reset value：0x0000_0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BBR[19:16]			
-												RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BBR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:20	保留	-	-	-
19:0	BBR[19:0]	RW	0x300	LPUART 波特率。 禁止在 LPUART_BRR 寄存器中写入小于 0x300 的值

39.4.5 LPUART 请求寄存器（LPUART_RQR）

Address offset：0x18

Reset value：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:5	保留	-	-	-
4	TXFRQ	W	0	传送数据刷新请求。 写 1 刷新整个 FIFO，同时会置位 TXFE（表明 TXFIFO 空）。
3	RXFRQ	W	0	接收数据刷新请求。 该位写 1 会清零 RXNE 标志。 通过该位能不读并且丢弃接受到的数据，避免 overrun 情况。
2	MMRQ	W	0	静音模式请求。 该位写 1 使 LPUART 进入静默模式，并置位 RWU 标志。
1	SBKRQ	W	0	发送中止请求。 该位写 1 会置位 SBKF，并发送 BREAK 请求。
0	保留	-	-	-

39.4.6 LPUART 中断和状态寄存器（LPUART_ISR）

Address offset：0x1C

Reset value：0x0080_00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res	TXFT	RXFT	Res	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
-				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
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
-					R	R	-	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:23	保留	-	-	-
27	TXFT	R	0	TXFIFO 阈值标志。 0：TXFIFO 未达到 TXFTCFG 阈值； 1：TXFIFO 达到 TXFTCFG 阈值；
26	RXFT	R	0	RCFIFO 阈值标志。 0：RXFIFO 未达到 RXFTCFG 阈值；

				1：RXFIFO 达到 RXFTCFG 阈值；
25	保留	-	-	-
24	RXFF	R	0	RXFIFO 满标志。表明接收到的数据 RXFIFO size+1。 0：RXFIFO 未满； 1：RXFIFO 满；
23	TXFE	R	1 开启 FIFO_EN 默认 1,否则默认 0	TXFIFO 空。 0：TXFIFO 未空； 1：TXFIFO 空； 注：FIFO_EN 关闭的时候该位为 0
22	REACK	R	0	接收使能确认标志。 硬件置位/复位该寄存器。表明硬件在进入低功耗模式前准备好接收。 如果 LPUART 不支持从停止低功耗模式唤醒，则该位保留。
21	TEACK	R	0	发送使能确认标志。 硬件置位/复位该寄存器。 当在 LPUART_CR1 寄存器中写入 TE=0 生成空闲帧请求，然后写入 TE=1 以满足 TE=0 的最小周期时，可以使用该寄存器。
20	WUF	R	0	从低功耗模式唤醒标志。 当检测到唤醒事件时硬件写该位为 1. 事件通过 WUS 位域定义。通过向 LPUART_ICR 寄存器中的 WUCF 写入 1，此位由软件清零。如果 LPUART_CR3 寄存器中 WUFIE=1，则会生成中断。 注：当 UESM 清零时，WUF 标志也清零。
19	RWU	R	0	接收到静默模式唤醒。 当接收到静默模式序列，该寄存器置位；如果接收到唤醒序列，该寄存器清零。具体哪种唤醒序列（地址标记或者空闲帧）由寄存器 LPUART_CR1.WAKE 位控制。 0：接收器为工作模式； 1：接收器为静默模式；
18	SBKF	R	0	发送中止标志。 该寄存器表明请求发送断开字符。由软件写 1 到 LPUART_RQR.SBKRQ 寄存器置位该寄存器。硬件在发送完中止字符的停止位后清零该寄存器。 0：不发送断开字符； 1：发送断开字符；
17	CMF	R	0	地址匹配标志。 当接收到 ADD[7:0]值匹配的字符时置位该寄存器。软件写 1 到 CMCF 寄存器会清零该位。
16	BUSY	R	0	忙标志。 当 RX 线接收数据时（正确接收到起始位），硬件该寄存器置位。接收结束，硬件清零该寄存器。 0：LPUART 空闲； 1：正在进行接收；
15:11	保留	-	-	-

10	CTS	R	0	CTS 标志。 0：CTS 线为 1； 1：CTS 线为 0；
9	CTSIF	R	0	CTS 中断标志。 0：CTS 状态线未改变； 1：CTS 状态线值改变；
8	保留	-	-	-
7	TXE	R	1	传输寄存器空标志，可以写数据到 LPUART_TDR。 0：数据寄存器满； 1：数据寄存器空；
6	TC	R	1	传送完成标志。 传送数据帧完成后，且 TXE=1，则硬件置位该寄存器。TCIE=1 时产生中断。 软件写 1 到 TCCF 寄存器清零该位。 0：传送未完成； 1：传送完成；
5	RXNE	R	0	读数据寄存器不空标志。 当移位寄存器值传送到 LPUART_RDR 寄存器，硬件置位该寄存器。 软件读 LPUART_RDR 寄存器则清零该位。 当 RXNEIE=1 时，产生中断。 0：未收到数据； 1：接收数据准备好读出；
4	IDLE	R	0	空闲标志。 检测到空闲帧，硬件置位该寄存器。当 IDLEIE=1 时产生中断。 软件写 1 到 IDLECF 清零该位。 0：未检测到 IDLE 帧； 1：检测到 IDLE 帧；
3	ORE	R	0	溢出错误标志。 当 RXNE=1 时，在移位寄存器中接收到的数据正准备转移到 RDR 寄存器时，硬件设置该位。 软件写 1 到 ORECF 寄存器清零该位。 当 RXNEIE=1 或者 EIE=1 时，产生中断。 注：该寄存器置位时，RDR 寄存器内容不会丢失，但移位寄存器内容被覆盖。 当 EIE=1 时，产生 ORE 中断。
2	NE	R	0	START 位噪声标志。 在数据帧接收到噪声时，硬件置位该寄存器。 软件写 NFCF=1 会清零该位。
1	FE	R	0	帧错误标志。 当检测到不同步、过多的噪声或中断字符时，由硬件设置此位。 软件写 FECF=1 清零该位。 当 EIE=1 时，产生中断。
0	PE	R	0	校验值错误。

				当接收时校验值错误时，硬件置位该寄存器。 软件写 PECF=1 清零该位。 当 PEIE=1 时，产生中断。

39.4.7 LPUART 中断标志清零寄存器（LPUART_ICR）

Address offset：0x20

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
-											W	-		W	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
-						W	-		W	-	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:21	保留	-	-	-
20	WUCF	W	0	低功耗唤醒唤醒标志清零。 软件写 1 会清零 WUF 寄存器。
19:18	保留	-	-	-
17	CMCF	W	0	地址匹配标志清零。 软件写 1 会清零 CMF 寄存器。
16:10	保留	-	-	-
9	CTSCF	W	0	CTS 标志清零。 软件写 1 清零 CTSIF 寄存器。
8:7	保留	-	-	-
6	TCCF	W	0	传送完成标志清零。 软件写 1 清零 TC 寄存器。
5	保留	-	-	-
4	IDLECF	W	0	空闲标志清零。 软件写 1 清零 IDLEF 寄存器。
3	ORECF	W	0	溢出错误标志清零。 软件写 1 清零 ORE 寄存器。
2	NECF	W	0	噪声标志清零。 软件写 1 会清零 NE 寄存器。
1	FECF	W	0	帧错误标志清零。 软件写 1 清零 FE 寄存器。
0	PECF	W	0	校验值错误标志清零。 软件写 1 清零 PE 寄存器。

39.4.8 LPUART 接收数据寄存器（LPUART_RDR）

Address offset：0x24

Reset value：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8:0	RDR[8:0]	R	0	接收数据寄存器。

39.4.9 LPUART 发送数据寄存器（LPUART_TDR）

Address offset：0x28

Reset value：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8:0	TDR[8:0]	RW	0	发送数据寄存器。 当校验使能时，该寄存器的 bit7 或者 bit8 在发送时被校验位代替。 注意：建议先开启 TE 和 UE 之后，再写 TDR 数据。

39.4.10 LPUART 预分频器寄存器（LPUART_PRESC）

Address offset：0x2C

Reset value：0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3:0	PRESCALER	RW	0	输入时钟预分频寄存器。 0000：不分频； 0001：2 分频； 0010：4 分频； 0011：6 分频； 0100：8 分频； 0101：10 分频； 0110：12 分频； 0111：16 分频； 1000：32 分频； 1001：64 分频； 1010：128 分频； 1011：256 分频； 其它：256 分频.

40. 时钟校准控制器（CTC）

40.1 简介

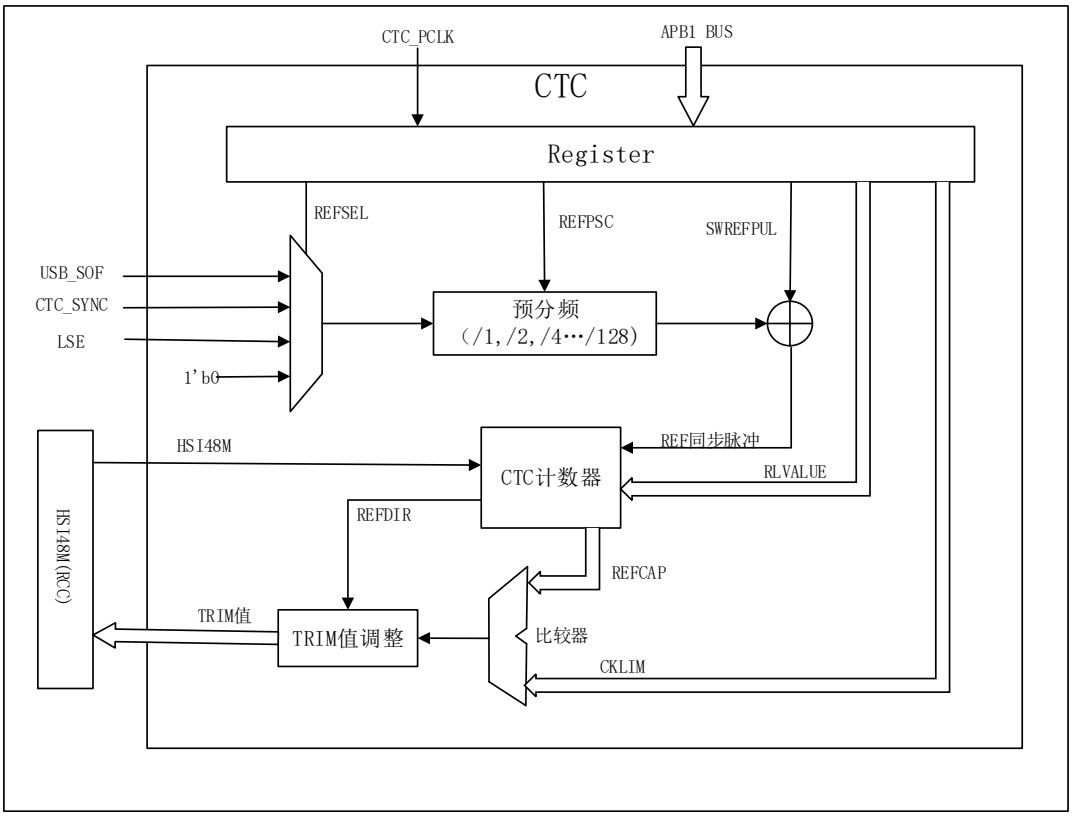
时钟校准控制器（CTC）采用硬件的方式，自动校准内部 48 MHz RC 晶振（HSI48）。当 OTG_FS 模块使用 HSI48 时钟作为时钟源时，必须要求 HSI48 时钟频率在 48 MHz±0.25%的范围内，但是没有经过校准的内部晶振无法满足这么高的精度。CTC 模块基于外部高精度的参考信号源来校准 HSI48 的时钟频率，通过自动的或手动的调整校准值，以得到一个精准的 HSI48 时钟。

40.2 主要特征

CTC 模块主要完成如下功能：

- 三个外部参考信号源：GPIO，LSE 时钟，来自 OTG_FS 模块的 USB_SOF 信号
- 提供软件参考同步脉冲；
- 硬件自动校准，无需软件操作；
- 具有参考信号源捕获和重载功能的 16 bits 校准计数器；
- 用于频率评估和自动校准的 8 bits 时钟校准基值；
- 标志位和中断，用于指示时钟校准的状态：校准成功状态（CKOKIF），警告状态（CKWARNIF）和错误状态（ERRIF）。

40.2.1 模块框图



40.3 功能描述

40.3.1 REF 同步脉冲发生器

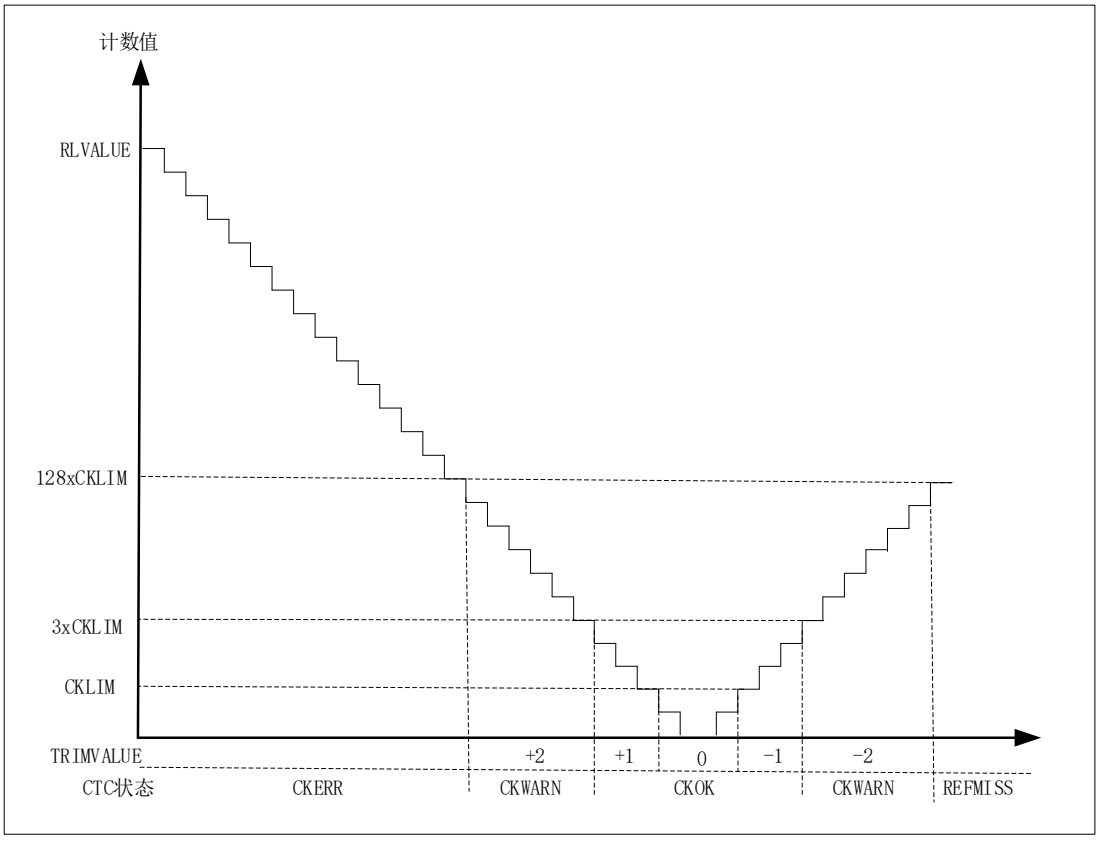
首先，通过设置 CTC_CTL1 寄存器（CTC 控制寄存器 1）中的 REFSEL 位来选择参考信号源：GPIO，LSE 时钟输出或者 USB_SOF。

然后，可以通过设置 CTC_CTL1 寄存器中的 REFPOL 位来配置参考信号源同步时的信号极性，通过设置 CTC_CTL1 寄存器中的 REFPSC 位来产生一个合适的同步时钟频率信号。

如果需要使用软件参考脉冲信号,则需要设置 CTC_CTL0 寄存器(CTC 控制寄存器 0)中的 SWREFPUL 位为 1。软件参考脉冲信号与外部参考脉冲信号最后进行逻辑'或'操作。

40.3.2 CTC 校准计数器

CTC 时钟校准计数器由 HSI48 提供时钟。在置位 CTC_CTL0 寄存器中的 CNTEN 位后，当检测到第一个参考同步脉冲信号，计数器开始从 RLVALUE 值（RLVALUE 在 CTC_CTL1 寄存器中定义）开始向下计数。每次检测到参考同步脉冲信号时，计数器重载 RLVALUE 值，同时重新开始向下计数。如果始终检测不到参考同步脉冲信号，计数器会向下计数到零，然后再向上计数到 128 x CKLIM（CKLIM 在 CTC_CTL1 中定义），最后停止，直到检测到下一个参考同步脉冲信号。一旦检测到参考同步脉冲信号，当前 CTC 校准计数器的计数值被捕获并存入 CTC_SR 寄存器中的 REFCAP 位中，同时，当前计数器的计数方向被存入 CTC_SR 寄存器中的 REFDIR 位。详细内容如下图所示。



40.3.3 频率评估和自动校准过程

当参考同步脉冲信号出现时，时钟频率评估功能开始执行。如果参考同步脉冲信号出现在计数器向下计数的过程中，说明当前时钟频率比期望时钟频率（频率为 48M）慢，需要增大 CTC_CTL0 中的 TRIMVALUE 值（时钟校准值）。如果参考同步脉冲信号出现在计数器向上计数的过程中，说明当前时钟频率比期望时钟频率快，需要减小 TRIMVALUE 值。CTC_SR 寄存器中的 CKOKIF 位，CKWARNIF 位，CKERR 位和 REFMISS 位反映了频率评估的状态。

如果 CTC_CTL0 中的 AUTOTRIM（硬件自动校准模式）位置 1，硬件使能自动校准模式。在这个模式中，如果参考同步脉冲信号出现在计数器向下计数的过程中，说明当前时钟频率比期望时钟频率慢，CTC_CTL0 中的 TRIMVALUE 值会自动增大，来提高当前的时钟频率。反之，如果参考同步脉冲信号出现在计数器向上计数的过程中，说明当前时钟频率比期望时钟频率快，TRIMVALUE 值会自动减小，从而减小当前的时钟频率。

- 计数器的值 < CKLIM 时，检测到参考同步脉冲信号：

CTC_SR 寄存器中的 CKOKIF 位（时钟校准成功标志位）被置位，同时，如果 CTC_CTL0 寄存器中的 CKOKIE 位（时钟校准完成中断使能位）置 1，将会产生一个中断。如果 CTC_CTL0 寄存器中的 AUTOTRIM 置 1，CTC_CTL0 寄存器中的 TRIMVALUE 值不变。

- CKLIM ≤ 计数器的值 < 3 x CKLIM 时，检测到参考同步脉冲信号：

CTC_SR 寄存器中的 CKOKIF 位被置位，同时，如果 CTC_CTL0 寄存器中的 CKOKIE 位置 1，将会产生一个中断。如果 CTC_CTL0 寄存器中的 AUTOTRIM 位置 1，在计数器向下计数过程中，CTC_CTL0 寄存器中的 TRIMVALUE 值将加 1，而在向上计数过程中将减 1。

- 3 x CKLIM ≤ 计数器的值 < 128 x CKLIM 时，检测到参考同步脉冲信号：

CTC_SR 寄存器中的 CKWARNIF 位（时钟校准警告中断位）被置位，同时，如果 CTC_CTL0 寄存器中的 CKWARNIE 位（时钟校准警告中断使能位）置 1，将会产生一个中断。如果 CTC_CTL0 寄存器中的 AUTOTRIM 位置 1，在计数器向下计数过程中，CTC_CTL0 寄存器中的 TRIMVALUE 值将加 2，而在向上计数过程中将减 2。

- 计数器的值 ≥ 128 x CKLIM，计数器在向下计数过程中，检测到参考同步脉冲信号：

CTC_SR 寄存器中的 CKERR 位 (时钟校准错误位) 被置位, 同时, 如果 CTC_CTL0 寄存器中的 ERRIE 位（错误中断使能位）置 1，将会产生一个中断。CTC_CTL0 寄存器中的 TRIMVALUE 值不变。

- 计数器的值 = 128 x CKLIM，计数器在向上计数过程中：

CTC_SR 寄存器中的 REFMISS 位（REF 同步脉冲丢失位）被置位，同时，如果 CTC_CTL0 寄存器中的 ERRIE 位置 1，将会产生一个中断。CTC_CTL0 寄存器中的 TRIMVALUE 值不变。

如果 CTC_CTL0 寄存器中的 TRIMVALUE 的校准值大于 127, 将会发生上溢事件, 同时, 若 TRIMVALUE 的校准值小于 0, 将会发生下溢事件。TRIMVALUE 的取值范围为 0~127 (上溢事件发生时, TRIMVALUE 值为 127；下溢事件发生时，TRIMVALUE 值为 0）。然后，CTC_SR 寄存器中的 TRIMERR 位（校准值错误位）将会被置位，如果 CTC_CTL0 寄存器中的 ERRIE 位置 1，将会产生一个中断。

40.3.4 软件编程指南

CTC_CTL1 寄存器中 RLVALUE 位和 CKLIM 位是时钟频率评估和硬件自动校准的关键。它们的数值由期望时钟的频率（HSI48：48 MHz）和参考同步脉冲信号的频率计算得到。理想状态是参考同步脉冲信号在 CTC 计数器计数到零时出现，所以 RLVALUE 的值为：

RLVALUE =（f_{clock} ÷ f_{REF}）– 1

CKLIM 的值由用户根据时钟的精度来设置，一般建议设为步长的一半，所以 CKLIM 的值为：

CKLIM =（f_{clock} ÷ f_{REF}）× 0.12% ÷ 2

典型的步长值是 0.12%，f_{clock} 是期望时钟的频率（48 MHz），f_{REF} 是参考同步脉冲信号的频率。

CTC_CTL0 寄存器中 TRIMVALUE 可以在 AUTOTRIM 位为 0 时通过软件写入，但修改 TRIMVALUE 会直接影响 HSI48M 时钟的频率，因此，不应该随意通过软件修改 TRIMVALUE。建议用户在两次参考信号中间，根据标志位判断修改；或者若已经存在可靠的值，用户可以直接修改 RCC_CFGR1 寄存器中 HSI48 TRIM 的值。

40.4 寄存器描述（基址 0x4000_C800）

40.4.1 CTC 控制寄存器 0(CTC_CTL0)

Address offset：0x00

Reset value：0x00004000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIMVALUE[6:0]							SWREFPUL	AUTOTRIM	CNTEN	Res.	EREFIE	ERRIE	CKWARNIE	CKOKIE
	RW							W	RW	RW		RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:15	保留	-	-	-
14:8	TRIMVALUE[6:0]	RW	7’h40	HSI48 校准值. 当 CTC_CTL0 中的 AUTOTRIM 值为 0 时，该位由软件置位和清除，该模式用于软件校准过程。 当 CTC_CTL0 中的 AUTOTRIM 值为 1 时，该位只读，由硬件自动修改，该模式用于硬件校准过程。 TRIMVALUE 的中间值是 64，当 TRIMVALUE 值加 1 时，HSI48 时钟频率增加大约 57 KHz。当 TRIMVALUE 值减 1 时，HSI48 时钟频率的减少大约 57KHz。
7	SWREFPUL	W	0	软件生成同步参考信号脉冲。 该位由软件置位，并为 CTC 计数器提供一个同步参考脉冲信号。该位由硬件自动清除，读操作时返回 0。 0：没有影响； 1：软件产生一个同步参考脉冲信号；

6	AUTOTRIM	RW	0	硬件自动校准模式。 该位由软件置位或清除。当该位置 1 时，硬件自动校准模式使能，通过硬件不断地自动修改 CTC_CTL0 中的 TRIMVALUE 值，直到 HSI48 的时钟频率达到 48MHz。 0：禁止硬件自动校准模式 1：使能硬件自动校准模式
5	CNTEN	RW	0	CTC 计数器使能。 该位由软件置位或清除，用于使能或禁止 CTC 计数器。当该位置 1 时，不能修改 CTC_CTL1 的值。 0：禁止 CTC 计数器 1：使能 CTC 计数器
4	保留	-	-	-
3	EREFIE	RW	0	期望参考信号中断使能。 0：禁止期望参考信号产生中断 1：使能期望参考信号产生中断
2	ERRIE	RW	0	错误中断使能。 0：禁止错误中断 1：使能错误中断
1	CKWARNIE	RW	0	时钟校准警告中断使能。 0：禁止时钟校准警告中断 1：使能时钟校准警告中断
0	CKOKIE	RW	0	时钟校准完成中断使能。 0：禁止时钟校准完成中断 1：使能时钟校准完成中断

40.4.2 CTC 控制寄存器 1(CTC_CTL1)

Address offset：0x04

Reset value：0x2022 BB7F

该寄存器只能按字（32 位）访问。当 CNTEN 为 1 时，不能修改该寄存器的值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFPOL	Res.	REFSEL[1:0]		Res.	REFPSC[2:0]			CKLIM[7:0]							
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
RLVALUE[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	REFPOL	RW	0	参考信号源极性。 该位由软件置位或清除，用于选择参考信号源的同步极性。 （如果参考信号源选择了 USB_SOF，则上升沿和下降沿将同时有效，不论该位为 0 或是 1） 0：选择上升沿 1：选择下降沿
30	保留	-	-	-

29:28	REFSEL[1:0]	RW	2’h2	参考信号源选择。 该位由软件置位或清除，用于选择参考信号源。 00：选择 GPIO 输入信号 01：选择 LSE 时钟 10：选择 USB_SOF 信号 11：保留，选择 0
27	保留	-	-	-
26:24	REFPSC[2:0]	RW	3’h0	参考信号源预分频。 该位由软件置位或清除。 000：参考信号不分频 001：参考信号 2 分频 010：参考信号 4 分频 011：参考信号 8 分频 100：参考信号 16 分频 101：参考信号 32 分频 110：参考信号 64 分频 111：参考信号 128 分频
23:16	CKLIM[7:0]	RW	8’h22	时钟校准时基限值。 该位由软件置位或清除，用于定义时钟校准时基限值。该位用于频率评估和自动校准过程。
15:0	RLVALUE[15:0]	RW	16’hBB7F	CTC 计数器重载值。 该位由软件置位或清除，用于定义 CTC 计数器的重载值，当检测到一个同步参考脉冲时，该值将重载到 CTC 校准计数器中。

40.4.3 CTC 状态寄存器（CTC_SR）

Address offset:0x08

Reset value:0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFCAP[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**
REFDIR	Res.				TRIMERR	REFMISS	CKERR	Res.				EREFIF	ERRIF	CKWARNIF	CKOKIF
R					R	R	R					R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:16	REFCAP[15:0]	R	16’h0	CTC 计数器捕获值。 当检测到一个同步参考脉冲信号时，CTC 校准计数器中的计数值被存入到 REFCAP 位中。
15	REFDIR	R	1	CTC 校准时钟计数方向。 当检测到一个同步参考脉冲信号时，CTC 校准计数器的计数方向被存入 REFDIR 位中。 0：向上计数 1：向下计数
14:11	保留	-	-	-
10	TRIMERR	R	0	校准值错误位。 当 CTC_CTL0 中的 TRIMVALUE 值发生上溢或下溢时，该位由硬件置位。若 CTC_CTL0 中的

				ERRIE 位置 1，则会产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 TRIMERR 位清零。 0：无校准值错误发生 1：发生校准值错误
9	REFMISS	R	0	同步参考脉冲信号丢失。 当同步参考脉冲信号丢失时，该位由硬件置位。当 CTC 校准计数器在增计数的过程中计数到 128 x CKLIM 都没有检测到同步参考脉冲信号时，REFMISS 位置位。说明当前时钟太快，无法校准到期望频率值，或者有其他错误产生。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 REFMISS 位清零。 0：无同步参考脉冲信号丢失 1：同步参考脉冲信号丢失
8	CKERR	R	0	时钟校准错误位。 当时钟校准错误产生时，该位由硬件置位。当 CTC 校准计数器计数值在减计数的过程中大于或等于 128 x CKLIM，并检测到同步参考脉冲信号时，CKERR 置位，说明当前时钟太慢，无法校准到期望频率值。当 CTC_CTL0 中的 ERRIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 CKERR 位清零。 0：无时钟校准错误发生 1：发生时钟校准错误
7:4	保留	-	-	-
3	EREFIF	R	0	期望参考中断标志位。 当 CTC 校准时钟计数器计数到 0 时，检测到参考信号，该位由硬件置位。当 CTC_CTL0 中的 EREFIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTC 中的 EREFIC 位，可以将 EREFIF 位清零。 0：无期望参考信号产生 1：期望参考信号产生
2	ERRIF	R	0	错误中断标志位。 当发生一个错误时，该位由硬件置位。只要有 TRIMERR，REFMISS 或者 CKERR 错误发生时，该位置位。当 CTC_CTL0 中的 ERRIE 置位时，产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 ERRIF 位清零。 0：无错误发生 1：发生错误
1	CKWARNIF	R	0	时钟校准警告中断标志位。 当时钟校准警告产生时，该位由硬件置位。当 CTC 校准计数器计数值大于或等于 3xCKLIM 且小于 128xCKLIM，并检测到同步参考脉冲信号时，CKWARNIF 置位。这说明当前时钟频率太慢或者

				太快，但可以通过校准达到期望频率值。当时钟校准警告产生时，TRIMVALUE 值加 2 或者减 2。当 CTC_CTL0 中的 CKWARNIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTC 中的 CKWARNIC 位，可以将 CKWARNIF 位清零。 0：无时钟校准警告发生 1：有时钟校准警告发生
0	CKOKIF	R	0	时钟校准成功中断标志位。 当时钟校准成功时，该位由硬件置位。若在 CTC 校准计数器计数值小于 3 x CKLIM 时，检测到同步参考脉冲信号，CKOKIF 置位。说明当前时钟频率正常，可以使用，不需要通过 TRIMVALUE 值进行时钟校准。当 CTC_CTL0 中的 CKOKIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTC 中的 CKOKIC 位，可以将 CKOKIF 位清零。 0：时钟校准未成功 1：时钟校准成功

40.4.4 CTC 中断清除寄存器（CTC_INTC）

Address offset:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:4	保留	-	-	-
3	EREFIC	W	0	EREFIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 EREFIF 位，写 0 没影响。
2	ERRIC	W	0	ERRIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 ERRIF 位，TRIMERR 位，REFMISS 位和 CKERR 位，写 0 没影响。
1	CKWARNIC	W	0	CKWARNIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 CKWARNIF 位，写 0 没影响。
0	CKOKIC	W	0	CKOKIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 CKOKIF 位，写 0 没影响

41. 随机数生成器(RNG)

41.1 概述

RNG 即随机数发生器。用户可以通过配置控制寄存器来使能和关闭 RNG 的随机源，随机数需要等待 DATA_RDY 位置位后才能被读取。

41.1.1 主要特性

主要功能如下：

- 拥有 2 个独立的随机源。
- 将随机源产生的随机数使用线性反馈移位寄存器(LFSR)进行后期处理，经过 32 个时钟周期后生成 32bit 的随机数。

41.1.2 顶层结构图

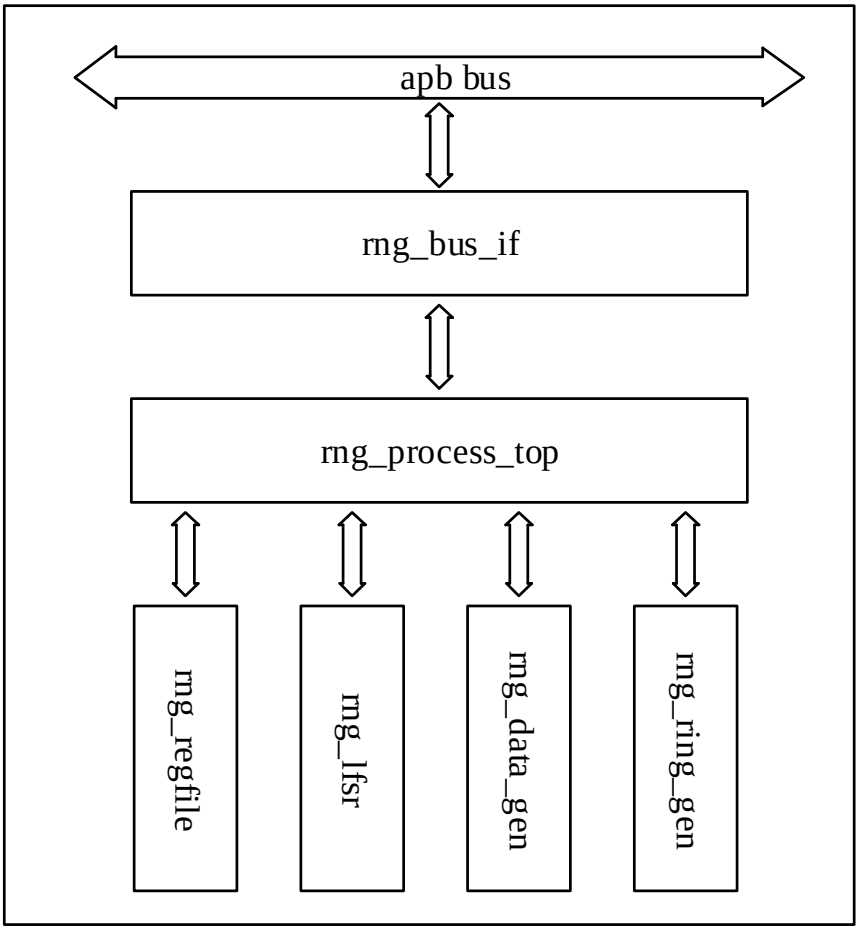


图 41-1 RNG 顶层模块框图

41.2 功能描述

41.2.1 随机数的产生

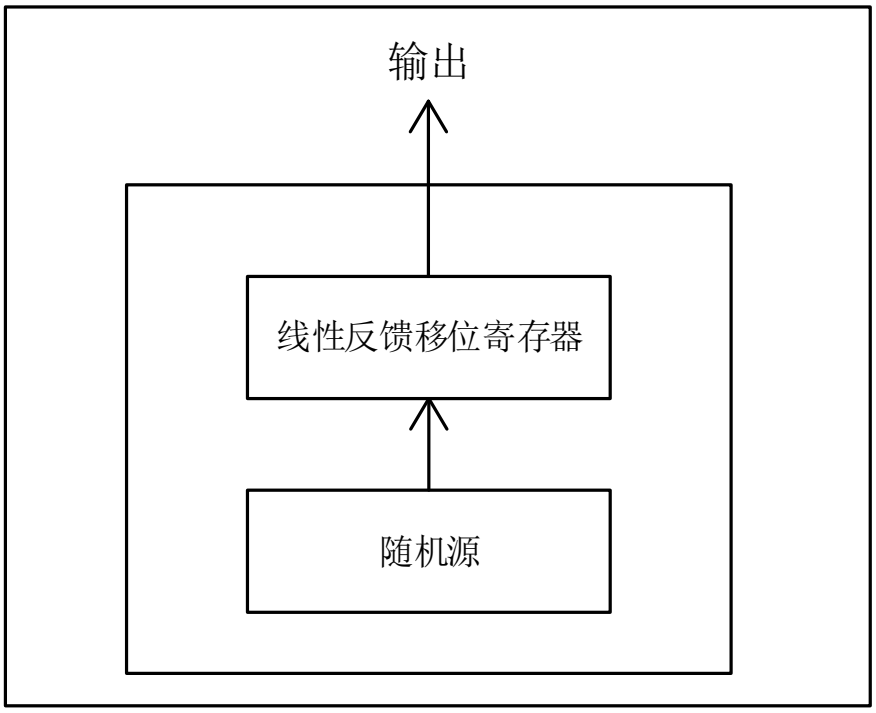


图 41-2 RNG 随机数产生模型框图

在使能 RNG 随机源后(RNG_CR = 0x1/2/3)，RNG 的随机源在每个时钟周期都会产生一个 1bit 的随机数，每个随机数都会经过线性反馈移位寄存器的处理，经过 32 个时钟周期后，产生一个 32bit 的随机数。

41.2.2 RNG 的软件操作

以只使能随机源 0 为例：

1. 打开 RNG 模块的时钟；
2. 设置控制寄存器 RNG_CR 为 0x01；
3. 等待 RNG_SR 寄存器中的 DATA_RDY 位置位；
4. 读取 RNG_DR 寄存器中的随机数。

41.2.3 RNG 的随机源

RNG 有两个随机源：

随机源 0 是由两组多个自由运行的与非门形成的环形振荡器的输出异或而产生的；

随机源 1 是由一组多个自由运行的同或门形成的环形振荡器的输出异或而产生的。

41.3 寄存器描述

寄存器宽度为 32bits。

41.3.1 RNG 控制寄存器（RNG_CR）

Address:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.													IE	RBG1_EN	RBG0_EN
													RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:3	保留	-	-	-
2	IE	RW	0	RNG 数据准备完成中断使能 1：使能中断。 0：禁止中断。
1	RBG1_EN	RW	0	随机源 1 使能位。 1：随机源 1 使能。 0：随机源 1 禁止。
0	RBG0_EN	RW	0	随机源 0 使能位。 1：随机源 0 使能。 0：随机源 0 禁止。

41.3.2 RNG 线性反馈移位寄存器(RNG_LFSR)

Address:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNG_LFSR[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
RNG_LFSR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	RNG_LFSR[31:0]	RW	32’h0	用于存放 RNG 线性反馈移位寄存器中的数据，用于查看当前线性反馈移位寄存器中的数据，也可以在随机源打开的情况下，将任意数据写入从而产生新的随机数。

41.3.3 RNG 状态寄存器(RNG_SR)

Address:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-

0	DATA_RDY	R	0	数据 ready 状态标志位 1：随机数据已经准备好。 0：随机数据还未准备好。 读取 RNG_DATA 后自动清零 如果 RNG_CR 中的 IE 位置 1 则会产生一个 RNG 中断

注意：1、DATA_RDY 位只有在随机源使能后才能被置 1；

2、一旦 RNG_DR 寄存器被读取，这个位返回 0，直到计算出一个新的有效值。

41.3.4 RNG 数据寄存器(RNG_DR)

Address:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	RNG_DATA[31:0]	R	32’h0	32 位随机数。

42. AES 硬件加速器(AES)

42.1 简介

AES 硬件加速器(AES)使用完全符合联邦信息处理标准(FIPS)第 197 号出版物中定义的高级加密标准(AES)的算法和实现，对数据进行加密或解密。

AES 支持 CTR、GCM、CMAC、CCM、ECB 和 CBC 链模式，密钥长度为 128 位或 256 位。

AES 是一种 AMBA AHB 从外设，只能通过 32 位单次访问。

外设支持传入和传出数据的 DMA 单次传输。

42.1.1 主要特征

- 支持 128 位和 256 位密钥长度
- 基于多种链模式的加密和解密
  - 密码块链 CBC
  - 电子密码本 ECB
  - 计数器 CTR
  - Galois 计数器 CCM、
  - Galois 消息认证码 CMAC
  - CBC-MAC 计数器 GCM 模式
- 数据为大端对齐
- 数据输入输出可以采用 DMA 进行操作

42.1.2 模块框图

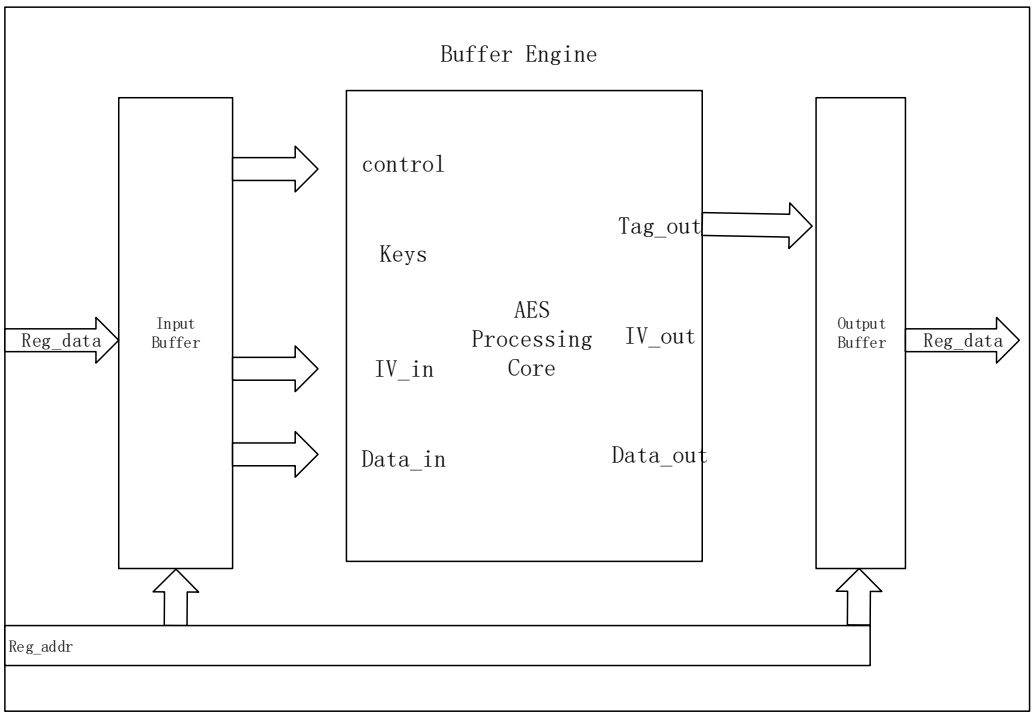
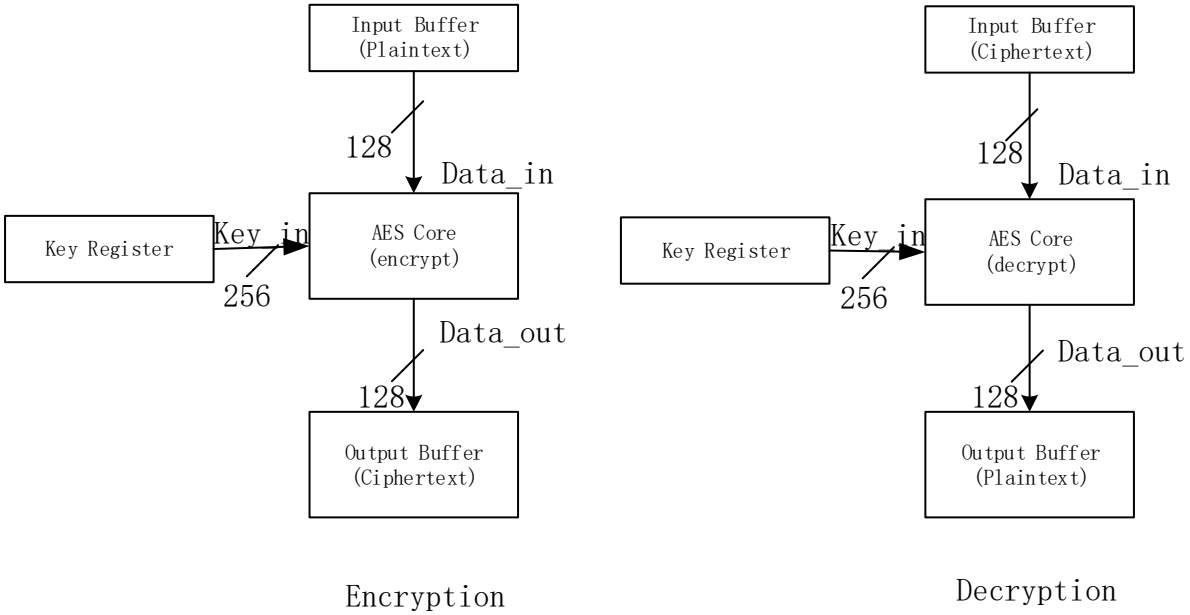


图 42-1 AES 模块框图

42.2 功能说明

42.2.1 ECB Mode

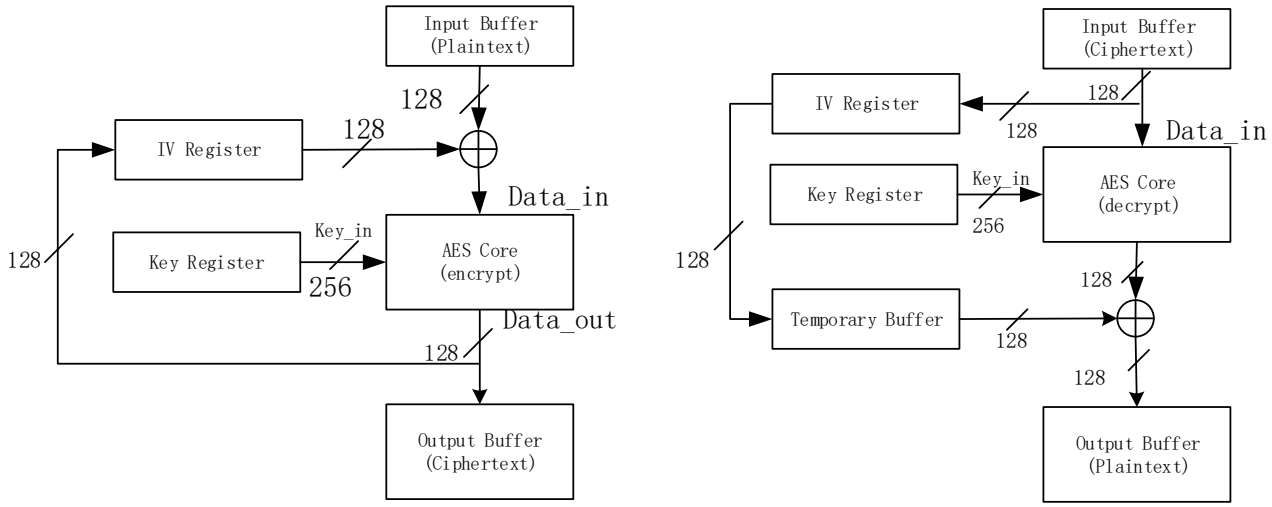


Encryption　　Decryption

基本的电子代码本(ECB)操作模式，其中输入数据直接传递给基本加密核心，加密核心的输出直接传递给输出缓冲区。

对于解密，加密核心反向操作。

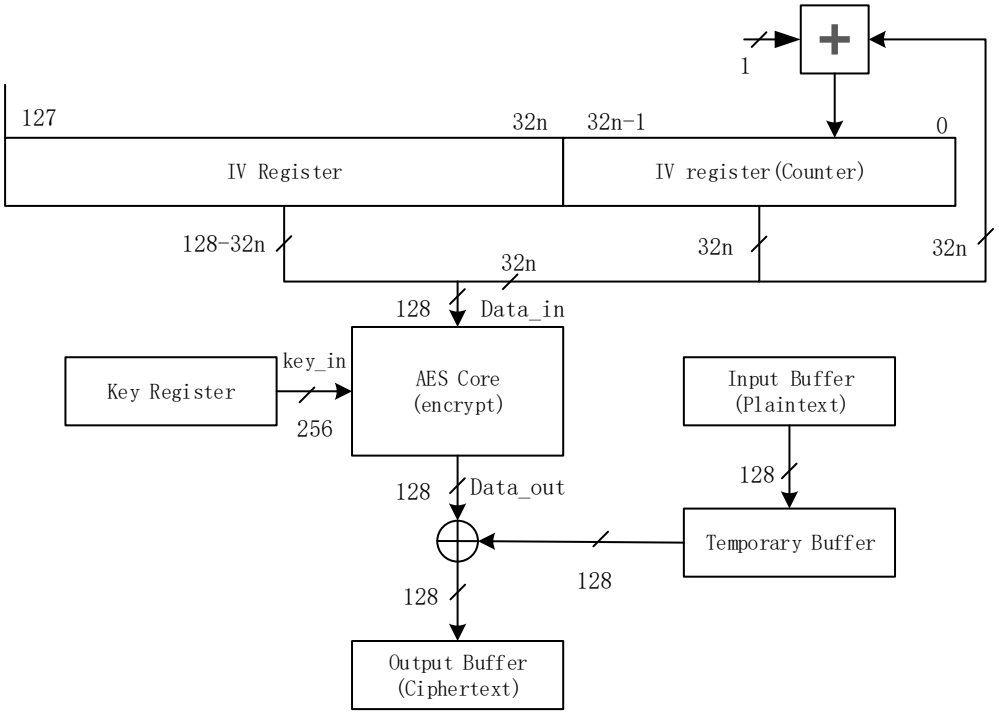
42.2.2 CBC Feedback Mode



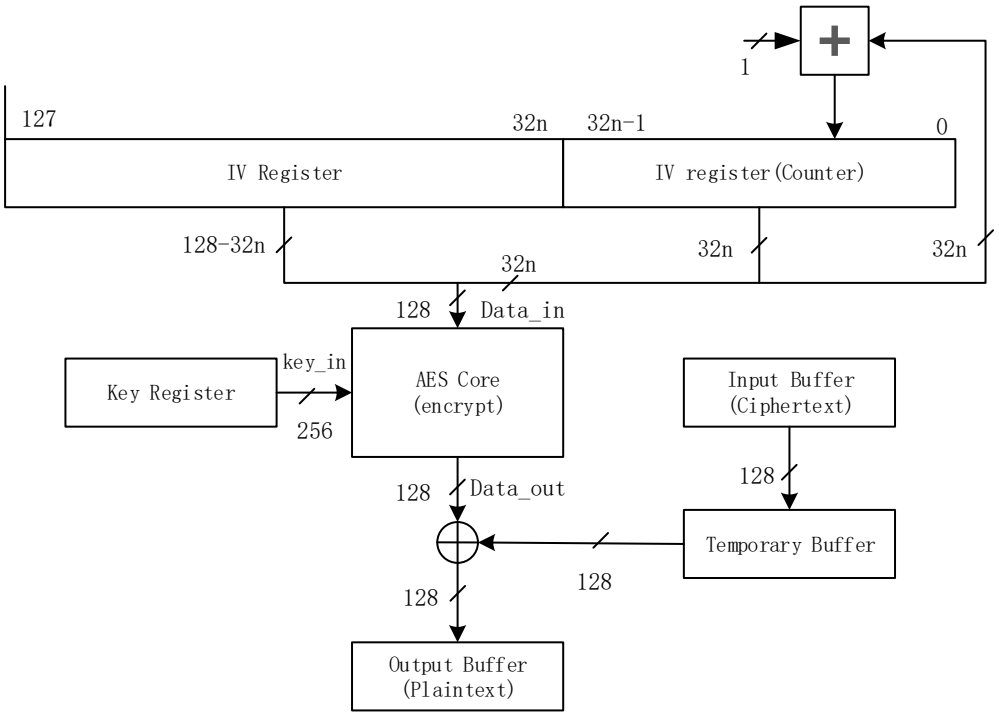
Encryption　　Decryption

密码块链(CBC)反馈操作模式，其中输入数据在传递到基本加密核心之前与初始化向量(IV)进行异或处理。加密核心的输出直接传递到输出缓冲区并成为下一个 IV。对于解密，逆转加密操作，和加密核心输出进行异或。当前操作的输入密文为下一操作的 IV。

42.2.3 CTR Mode



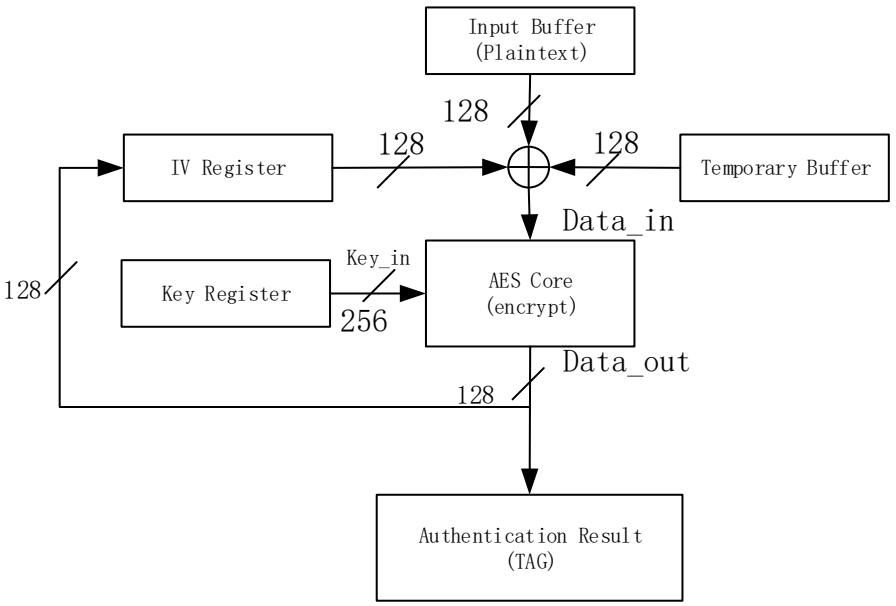
Encryption



Decryption

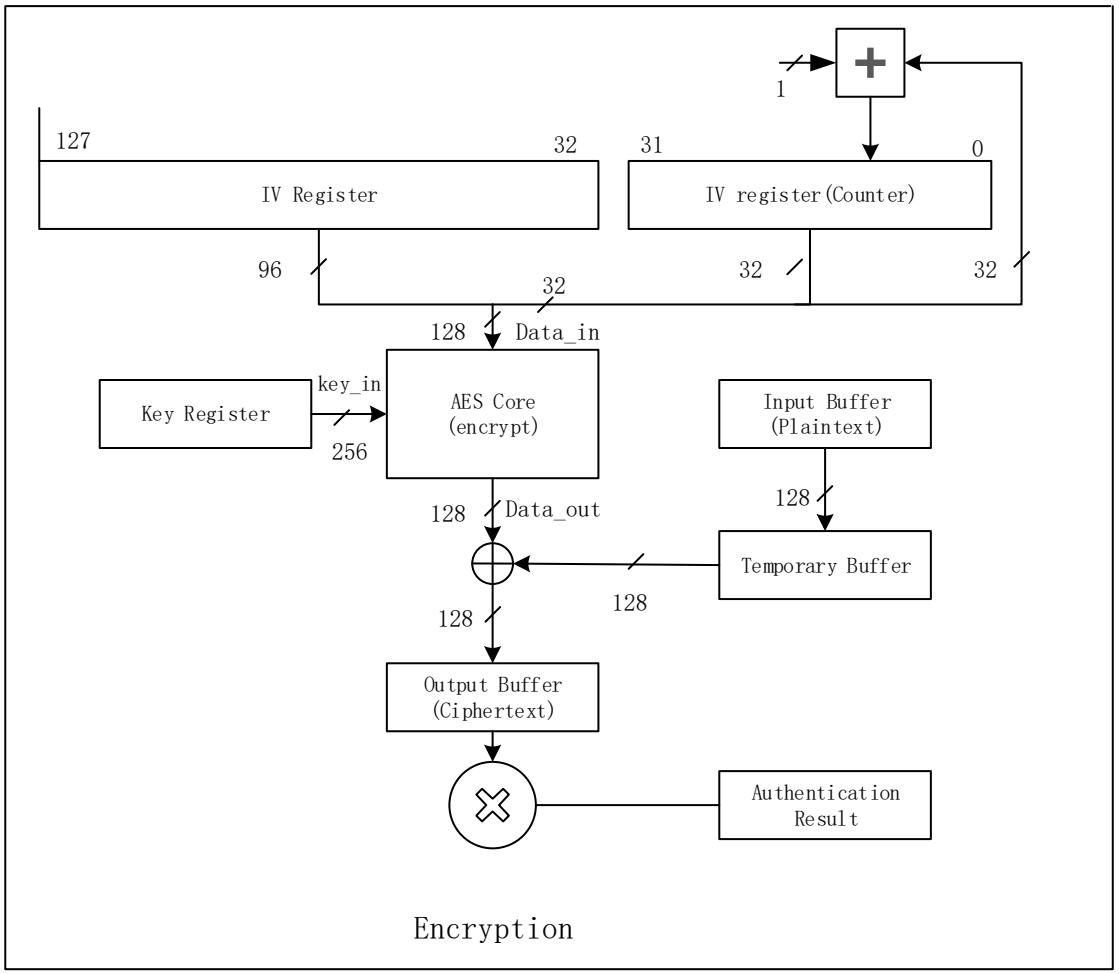
Counter（CTR)的操作模式。该操作将加密 IV。加密核心(加密 IV)的输出与数据进行异或运算，从而产生输出结果。IV 由两个部分组成，一个固定部分和一个计数器部分。计数器部分随着每个块的增加而增加。计数器宽度根据上下文而变化，可以是 16、32、64、96 或 128 位宽。请注意，在此模式下，加密和解密使用相同的加密操作。

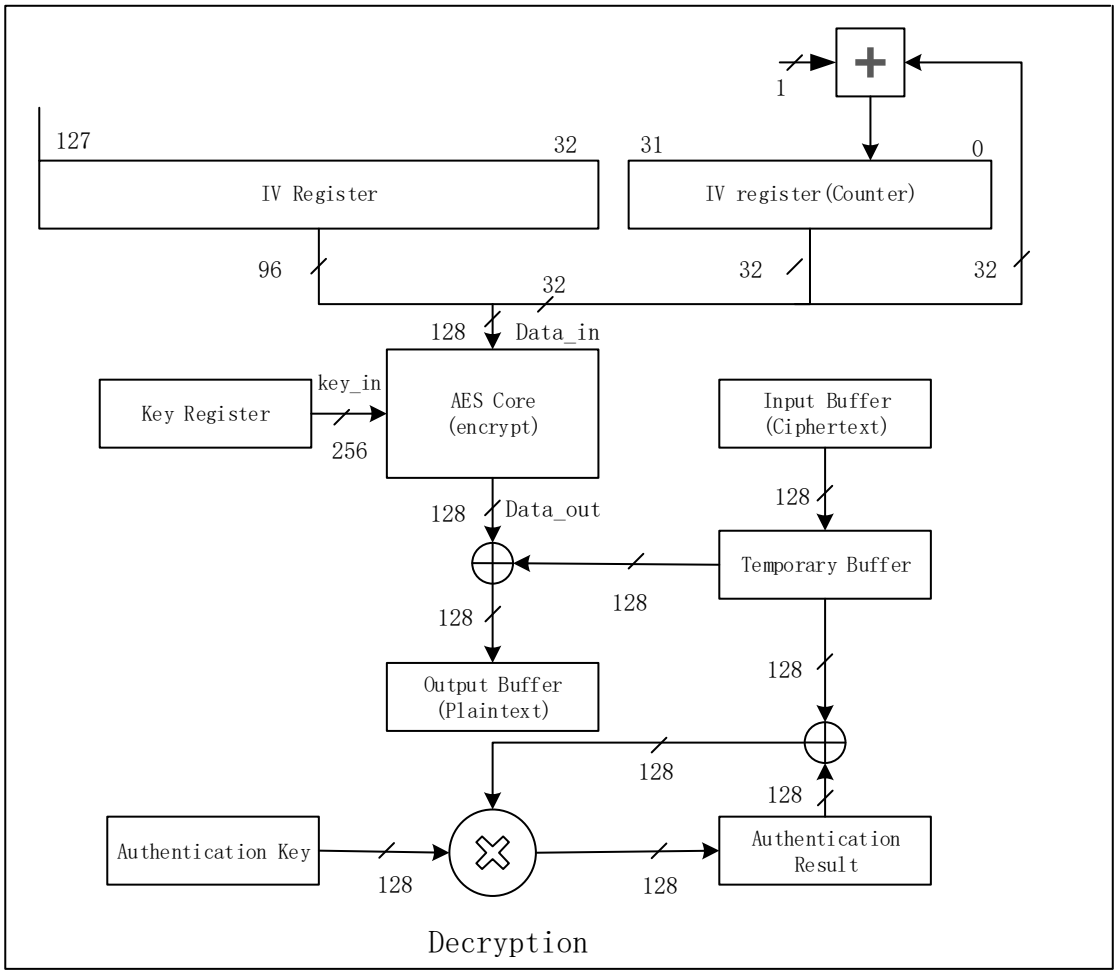
42.2.4 CMAC Authentication Mode



CMAC 认证模式，其中加密核心的输入与 IV 进行异或。然后，加密核心的输出被反馈为下一个区块的 IV。最后一个数据输入块和附加数据（存储在临时缓冲区）以及 IV 做异或处理，异或结果进行加密操作输出结果（TAG）。

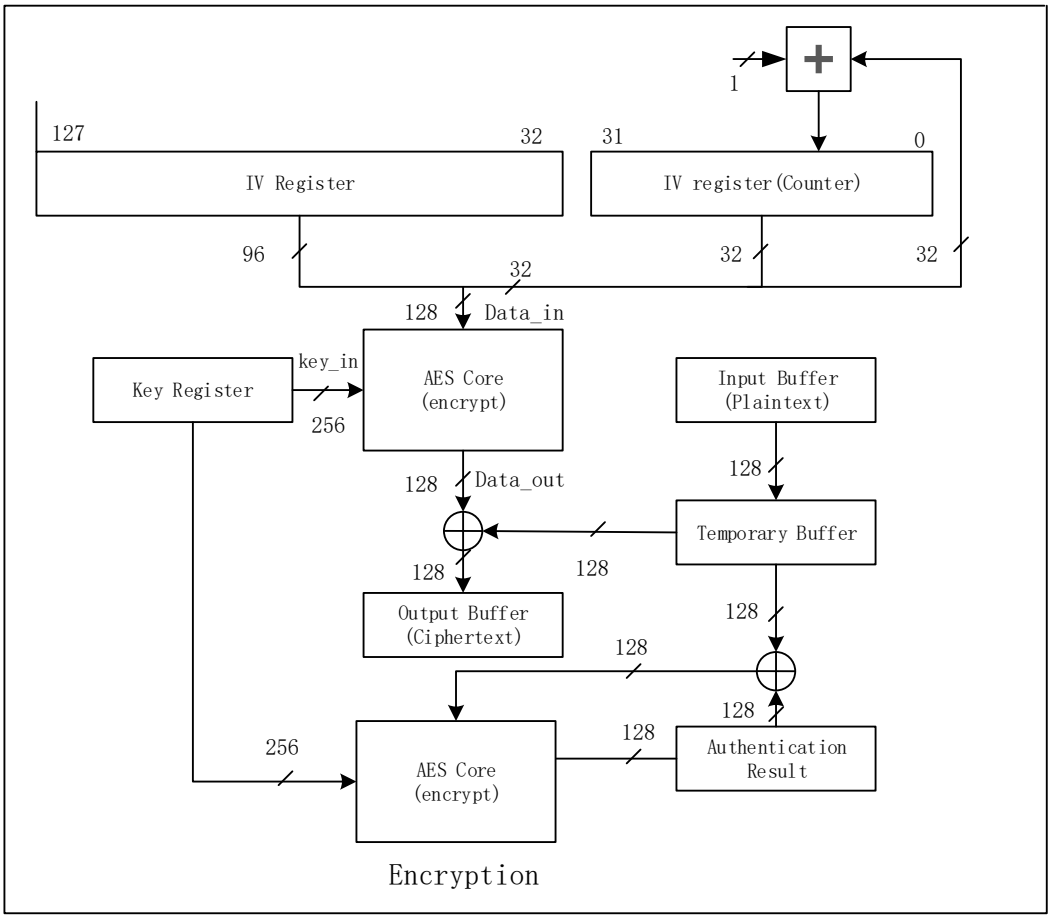
42.2.5 GCM Operation

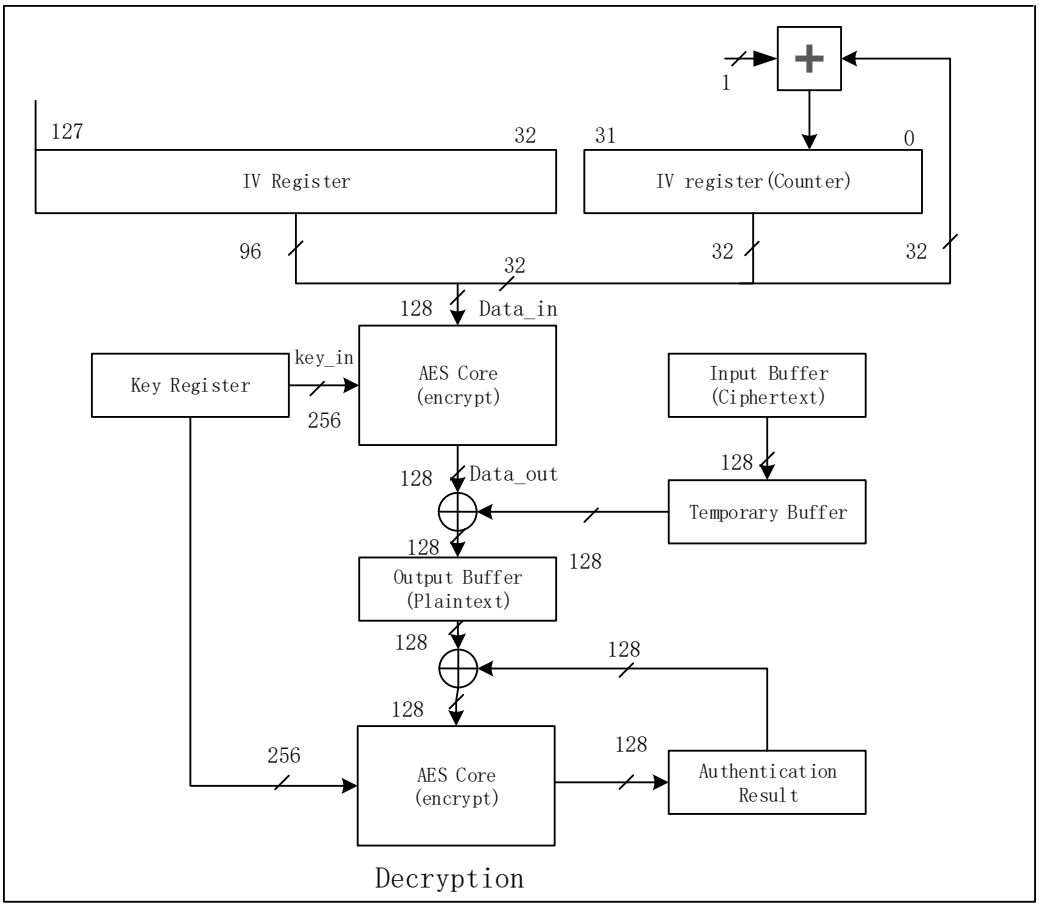




上图说明用于加密和解密的 GCM 操作的一个回合。一个 32 位计数器被用作 IV(就像它用于 CTR 模式一样)。数据以与 CTR 模式相同的方式加密，通过将加密核心输出与输入进行异或处理。加密/解密后，密文与中间认证结果进行异或。异或结果用作多项式乘法的输入，以创建下一个(中间)身份验证结果。

42.2.6 CCM Operation





上图显示用于加密和解密的 CCM(带有 CBC-MAC 的计数器)操作的一个回合。一个 32 位计数器被用作 IV(就像它用于 CTR 模式一样)。数据与 CTR 模式相同的方式加密，通过将加密核心输出与输入进行异或处理。明文直接在加密操作之后，与中间认证结果进行异或处理，异或结果用作第二个加密操作的输入，以计算下一个(中间)身份验证结果。

42.3 寄存器描述

AES 寄存器基地址：0x4800_2400

偏移地址	名称	描述
0x00	AES_MSG_CFG	信息控制寄存器
0x04	AES_CTX_CFG	密钥控制寄存器
0x08	AES_MSG_TOTAL_BYTES	信息长度寄存器
0x0C	AES_MSG_AAD_BYTES	附加信息长度寄存器
0x10	AES_GCM_AAD_INFO	GCM 模式附加信息长度寄存器
0x14	AES_CTX_KEY_SEL	密钥选择寄存器
0x20	AES_CTX_KEY0	密钥寄存器 0
0x24	AES_CTX_KEY1	密钥寄存器 1
0x28	AES_CTX_KEY2	密钥寄存器 2
0x2C	AES_CTX_KEY3	密钥寄存器 3
0x30	AES_CTX_KEY4	密钥寄存器 4
0x34	AES_CTX_KEY5	密钥寄存器 5
0x38	AES_CTX_KEY6	密钥寄存器 6
0x3C	AES_CTX_KEY7	密钥寄存器 7
0x40	AES_CTX_CBC_KEY0	CBC 模式密钥寄存器 0
0x44	AES_CTX_CBC_KEY1	CBC 模式密钥寄存器 1
0x48	AES_CTX_CBC_KEY2	CBC 模式密钥寄存器 2
0x4C	AES_CTX_CBC_KEY3	CBC 模式密钥寄存器 3
0x50	AES_CTX_CTR0	CTR 模式算子寄存器 0
0x54	AES_CTX_CTR1	CTR 模式算子寄存器 1
0x58	AES_CTX_CTR2	CTR 模式算子寄存器 2
0x5C	AES_CTX_CTR3	CTR 模式算子寄存器 3
0x60	AES_CTX_IV0	初始向量寄存器 0
0x64	AES_CTX_IV1	初始向量寄存器 1
0x68	AES_CTX_IV2	初始向量寄存器 2
0x6C	AES_CTX_IV3	初始向量寄存器 3
0x70	AES_CTX_MAC0	消息验证码寄存器 0
0x74	AES_CTX_MAC1	消息验证码寄存器 1
0x78	AES_CTX_MAC2	消息验证码寄存器 2
0x7C	AES_CTX_MAC3	消息验证码寄存器 3
0x80	AES_INGRESS_FIFO	输入 FIFO 寄存器
0x84	AES_ING_DBCFG	输入 FIFO 状态寄存器
0x88	AES_DMA_ENG_LEN	输出 FIFO 寄存器
0x8C	AES_ENG_DBCFG	输出 FIFO 状态寄存器

0x90	AES_DMA_ING_LEN	DMA 输入长度寄存器
0x94	AES_INGF_STATUS	DMA 输入爆发传输设置寄存器
0x98	AES_ENGRESS_FIFO	DMA 输出长度寄存器
0x9C	AES_ENGFIFO_STATUS	
0xA0	AES_ING_DMA_DONE	DMA 输出爆发传输设置寄存器
0xA4	AES_ENG_DMA_DONE	完成状态寄存器
0xA8	AES_DONE_STATUS	DMA 输入完成状态寄存器

42.3.1 信息控制寄存器(AES_MSG_CFG)

Address offset:0x00

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	KEY_SIZE		MAC_LEN				ALG_MODE				DIR	MSG_BEGIN	MSG_END	AES_GO
-	-	RW	RW	RW				RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	保留	-	-	-
13:12	KEY_SIZE[1:0]	RW	2'h0	设定密钥位宽： 0：128bit 1：Res 2：256bit
11:8	MAC_LEN[3:0]	RW	4'h0	设定消息验证码或初始向量长度 （仅用于 CCM、CMAC、XCBC） 在标准规范中，MAC 的长度为 4、6、8、10、12、14、16 字节 0： 16 bytes 1： 1 byte 2： 2 bytes 3： 3 bytes 4： 4 bytes 5： 5 bytes 6： 6 bytes 7： 7 bytes 8： 8 bytes 9： 9 bytes 10： 10 bytes 11： 11 bytes 12： 12 bytes 13： 13 bytes 14： 14 bytes 15： 15 bytes
7:4	ALG_MODE[3:0]	RW	4'h0	选择模式：

				0：ECB 1：CBC 2：CTR 3：CCM 4：CMAC 5：GCM 6~15：Res
3	DIR	RW	0	1：Encrypt 0：Decrypt
2	MSG_BEGIN	RW	0	指示当前操作包含信息的第一个字节。将会在操作之前自动取得需要的密钥。
1	MSG_END	RW	0	指示当前操作包含信息的最后一个字节。如果操作结果是消息验证码或初始向量，将会在操作完成后自动存储到密钥存储空间。
0	AES_GO	RW	0	启动加解密操作。开始操作后此位和其他控制位在 AES_DONE 置位前需要保持不变。

42.3.2 密钥控制寄存器(AES_CTX_CFG)

Address offset:0x04

Reset value:0x0000_0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTX_INDEX												INV_KEY_STR	INV_KEY_RET	CTX_STR	CTX_RET
RW												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	保留	-	-	-
15:4	CTX_INDEX[11:0]	RW	12’h0	上下文页编号
3	INV_KEY_STR	RW	0	存储上一轮密钥到密钥存储空间
2	INV_KEY_RET	RW	0	取出上一轮密钥
1	CTX_STR	RW	0	把当前密钥存储到密钥存储空间
0	CTX_RET	RW	1	取出当前密钥。

42.3.3 信息长度寄存器(AES_MSG_TOTAL_BYTES)

Address offset:0x08

Reset value:0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-

27:0	TOTAL_BYTES[27:0]	RW	28’h0	设定信息的长度，包括所有分组，不包括附加信息。 在 CCM 和 GCM 模式中需要此项。

42.3.4 附加信息长度寄存器（AES_MSG_AAD_BYTES）

Address offset:0x0C

Reset value:0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:16	ADD_LEN	RW	16’h0	设定当前操作中附加信息的长度。 在 CCM 和 GCM 模式中需要此项。
15:0	MSG_LEN	RW	16’h0	设定当前操作中信息的长度，包括信息和附加信息。所有输入到 AES 中的数据总数 = 所有写入到 In-gress FIFO 中的数据总数

42.3.5 GCM 模式附加信息长度寄存器（AES_GCM_AAD_INFO）

Address offset:0x10

Reset value:0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:28	保留	-	-	-
27:0	ADD_LEN_TOT[27:0]	RW	28’h0	设定附加信息的总长度，包括所有分组。在 GCM 模式中需要此项。

42.3.6 密钥寄存器 0（AES_CTX_KEY0）

Address offset：0x20

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_0															
W															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY_0															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_0	W	0	密钥[31:0]

42.3.7 密钥寄存器 1（AES_CTX_KEY1）

Address offset：0x24

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_1															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_1															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_1	W	32’h0	密钥[63:32]

42.3.8 密钥寄存器 2（AES_CTX_KEY2）

Address offset:0x28

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_2															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_2															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_2	W	32’h0	密钥[95:64]

42.3.9 密钥寄存器 3（AES_CTX_KEY3）

Address offset:0x02C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_3															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_3															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_3	W	32’h0	密钥[127:96]

42.3.10密钥寄存器 4（AES_CTX_KEY4）

Address offset:0x30

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_4															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_4															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_4	W	32’h0	密钥[159:128]

42.3.11密钥寄存器 5（AES_CTX_KEY5）

Address offset:0x34

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_5															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_5															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_5	W	32’h0	密钥[191:160]]

42.3.12密钥寄存器 6（AES_CTX_KEY6）

Address offset:0x38

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_6															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_6															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_6	W	32’h0	密钥[223:192]

42.3.13密钥寄存器 7（AES_CTX_KEY7）

Address offset:0x3C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY_7															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY_7															
W															

Bit	Name	R/W	Reset Value	Function
31:0	KEY_7	W	32’h0	密钥[255:224]

42.3.14CBC 模式密钥寄存器 0（AES_CTX_CBC_KEY0）

Address offset:0x40

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CBC_KEY_0															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBC_KEY_0															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CBC_KEY_0	W	32’h0	CBC 密钥[31:0]

42.3.15 CBC 模式密钥寄存器 1（AES_CTX_CBC_KEY1）

Address offset:0x44

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CBC_KEY_1															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBC_KEY_1															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CBC_KEY_1	W	32’h0	CBC 密钥[63:32]

42.3.16 CBC 模式密钥寄存器 2（AES_CTX_CBC_KEY2）

Address offset:0x48

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CBC_KEY_2															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBC_KEY_2															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CBC_KEY_2	W	32’h0	CBC 密钥[95:64]

42.3.17 CBC 模式密钥寄存器 3（AES_CTX_CBC_KEY3）

Address offset:0x4C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CBC_KEY_3															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBC_KEY_3															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CBC_KEY_3	W	32’h0	CBC 密钥[127:96]

42.3.18 CTR 模式算子寄存器 0（AES_CTX_CTR0）

Address offset:0x50

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCM_CTR_0															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCM_CTR_0															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CCM_CTR_0	W	32'h0	CTR/CCM 密钥[31:0]

42.3.19CTR 模式算子寄存器 1（AES_CTX_CTR1）

Address offset：0x54

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCM_CTR_1															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCM_CTR_1															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CCM_CTR_1	W	32'h0	CTR/CCM 密钥[63:32]

42.3.20CTR 模式算子寄存器 2（AES_CTX_CTR2）

Address offset：0x58

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCM_CTR_2															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCM_CTR_2															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CCM_CTR_2	W	32'h0	CTR/CCM 密钥[95:64]

42.3.21CTR 模式算子寄存器 3（AES_CTX_CTR3）

Address offset：0x5C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCM_CTR_3															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCM_CTR_3															
W															

Bit	Name	R/W	Reset Value	Function
31:0	CCM_CTR_3	W	32'h0	CTR/CCM 密钥[127:96]

42.3.22初始化向量寄存器 0（AES_CTX_IV0）

Address offset：0x60

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV0															
W															

Bit	Name	R/W	Reset Value	Function
31:0	IV0	W	32'h0	初始向量[31:0]

42.3.23初始化向量寄存器 1（AES_CTX_IV1）

Address offset：0x64

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV1															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV1															
W															

Bit	Name	R/W	Reset Value	Function
31:0	IV1	W	32’h0	初始向量[63:32]

42.3.24初始化向量寄存器 2（AES_CTX_IV2）

Address offset：0x68

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV2															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV2															
W															

Bit	Name	R/W	Reset Value	Function
31:0	IV2	W	32’h0	初始向量[95:64]

42.3.25初始化向量寄存器 3（AES_CTX_IV3）

Address offset：0x6C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV3															
W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV3															
W															

Bit	Name	R/W	Reset Value	Function
31:0	IV3	W	32’h0	初始向量[127:96]

42.3.26消息验证码寄存器 0（AES_CTX_MAC0）

Address offset：0x70

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAC_0															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_0															
R															

Bit	Name	R/W	Reset Value	Function
31:0	MAC_0	R	32’h0	初始向量[31:0]

42.3.27消息验证码寄存器 1（AES_CTX_MAC1）

Address offset：0x74

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAC_1															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_1															
R															

Bit	Name	R/W	Reset Value	Function
31:0	MAC_1	R	32’h0	初始向量[63:32]

42.3.28消息验证码寄存器 2（AES_CTX_MAC2）

Address offset：0x78

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAC_2															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_2															
R															

Bit	Name	R/W	Reset Value	Function
31:0	MAC_2	R	32’h0	初始向量[95:64]

42.3.29消息验证码寄存器 3（AES_CTX_MAC3）

Address offset：0x7C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAC_3															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_3															
R															

Bit	Name	R/W	Reset Value	Function
31:0	MAC_3	R	32’h0	初始向量[127:96]

42.3.30输入 FIFO（AES_INGRESS_FIFO）

Address offset：0x80

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	INGRESS_FIFO	W	32’h0	AES 输入数据 FIFO 入口

42.3.31 DMA 输入爆发传输设置寄存器（AES_ING_DBCFG）

Address offset：0x84

Reset value：0xC000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLAINTEXT_REQ.	DMA_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
R/W	R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DST_MSIZE			DST_TR_WIDTH		
-	-	-	-	-	-	-		-	-	R/W	R/W	R/W	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	PLAINTEXT_REQ	R/W	1	此位设定当前 AES 模式是否需要传入数据
30	DMA_EN	R/W	1	此位设定输入数据由 DMA 还是 CPU 传入 0：CPU 1：DMA
29:6	保留	-	-	-
5:3	DST_MSIZE	R/W	2’h2	DMA 的目标传输爆发传输长度： 0： 1 1：4 2：8 3：16 4：32 5：64 6：128
2:0	DST_TR_WIDTH	R/W	2’h 2	DMA 的目标传输宽度： 0：8bit 1：16bit 2：32bit

42.3.32 DMA 输出长度寄存器（AES_DMA_ENG_LEN）

Address offset：0x88

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	ENGRESS_DMA_DATA_LENGTH	R/W	32’h0	设定 DMA 向 AES 输入数据的总长度，用于生成握手信号。

42.3.33 DMA 输出爆发传输设置寄存器（AES_ENG_DBCFG）

Address offset：0x8C

Reset value：0xC000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIPERTEXT_REQ.	DMA_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
R/W	R/W	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRC_MSIZE			SRC_TR_WIDTH		
-	-	-	-	-	-	-		-	-	R/W	R/W	R/W	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	CIPERTEXT_REQ	R/W	1	此位设定当前 AES 模式是否需要传出数据
30	DMA_EN	R/W	1	此位设定输入数据由 DMA 还是 CPU 传入 0：CPU 1：DMA
29:6	保留	-	-	-
5:3	SRC_MSIZE	R/W	2'h2	DMA 的目标传输爆发传输长度： 0： 1 1：4 2：8 3：16 4：32 5：64 6：128
2:0	SRC_TR_WIDTH	R/W	2'h2	DMA 的目标传输宽度： 0：8bit 1：16bit 2：32bit

42.3.34DMA 输入长度寄存器（AES_DMA_ING_LEN）

Address offset：0x90

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:0	INGRESS_DMA_DATA_LENGTH	R/W	32'h0	设定 DMA 向 AES 输入数据的总长度，用于生成握手信号。 即使使用 CPU 而不是 DMA 来传输数据，此项依然需要。

42.3.35输入 FIFO 状态寄存器（AES_INGF_STATUS）

Address offset：0x94

Reset value：0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	ING_FULL	R	0	指示输入数据 FIFO 满
0	ING_EMPTY	R	1	指示输入数据 FIFO 空

42.3.36输出 FIFO（AES_ENGRESS_FIFO）

Address offset：0x98

Reset value：0xXXXX XXXX

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

Bit	Name	R/W	Reset Value	Function
31:0	INGRESS_FIFO	R	X	AES 输入数据 FIFO 入口

42.3.37输出 FIFO 状态寄存器（AES_ENGFIFO_STATUS）

Address offset：0x9C

Reset value：0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	ENG_FULL	R	0	指示输入数据 FIFO 满
0	ENG_EMPTY	R	1	指示输入数据 FIFO 空

42.3.38DMA 输入完成状态寄存器（AES_ING_DMA_DONE）

Address offset：0XA0

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	DMA_INGRESS_DONE	R	0	指示 DMA 输入已经完成传输。 读取清零

42.3.39DMA 输出完成状态寄存器（AES_ENG_DMA_DONE）

Address offset：0XA4

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:1	保留	-	-	-
0	DMA_NGRESS_DONE	R	0	指示 DMA 输出已经完成传输。 读取清零

42.3.40完成状态寄存器（AES_DONE_STATUS）

Address offset：0XA8

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:2	保留	-	-	-
1	MAC_VAILD	RC/W0	0	指示 MAC 验证是否成功 此位仅在 AES_DONE 置位后有效， AES_GO 置位后自动清零 写 0 可以清中断。
0	AES_DONE	RC/W0	0	指示操作已经完成 AES_GO 置位后自动清零 写 0 可以清中断。

43. CORDIC 协处理器(CORDIC)

43.1 概述

CORDIC 协处理器提供电机控制、计量、信号处理和许多其他应用中常用的数学函数（主要是三角函数、双曲函数）的硬件加速。

与软件实现相比，它加快了这些函数的计算速度，从而处理器可以使用较低的工作频率，或者不占用处理器资源以执行其他任务。

43.1.1 主要特性

主要特性如下：

- 24 位 CORDIC
- 圆周和双曲线系统
- 旋转和矢量模式
- 功能：正弦、余弦、sinh、cosh、atan、atan2、atanh、modulus、square root、natural
- 可编程精度
- 低延迟 AHB 接口
- 输入所需数据后，可直接读取结果（总线挂起），无需轮询或中断
- 还支持轮询、中断、DMA 等读取结果方式
- DMA 读写通道
- 通过 DMA 进行多次寄存器读/写

43.2 功能描述

43.2.1 概述

CORDIC 是一种用于计算三角函数和双曲函数的高效逐步逼近算法。

在三角（圆形）模式下，通过将单位向量[1, 0]通过逐渐减小的角度旋转，直到旋转角度的累积和等于输入角度θ，从而确定角度θ的正弦和余弦。旋转向量 x 和 y 分别对应于θ的余弦和正弦。反过来，向量[x, y]的角度就是$arctangent\left(y/x\right)$，通过将[x, y]通过逐渐减小的角度旋转以获得单位向量[1, 0]。旋转角度的累积和就是原始向量的角度。

CORDIC 算法也可以用于计算双曲函数（sinh，cosh，atanh），方法是将连续的圆形旋转替换为沿双曲线的旋转。

其他函数可以从上述基本函数派生出来。

43.2.2 CORDIC 函数

使用协处理器的第一步是通过相应地对 CORDIC_CSR 寄存器 FUNC 进行编程来选择所需的功能。

表 43-1 CORDIC 协处理器支持的功能列表

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	Angle θ	modulus m	m * cosθ	m * sinθ
Sine	Angle θ	modulus m	m * sinθ	m * cosθ
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1}$ x	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1}$ x	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

有几个函数需要两个输入参数（ARG1 和 ARG2），有些函数同时生成两个结果（RES1 和 RES2）。例如，在进行极坐标到直角坐标的转换时：$\sin\theta$ 也会生成$\cos\theta$ ，$\cos\theta$也会生成$\sin\theta$。同样，对于直角坐标到极坐标的转换（phase(x,y)，modulus(x,y)）和双曲函数（$\cosh\theta$ ，sinhθ）也是如此。

注意：指数函数，exp x，可以作为 sinh x 和 cosh x 之和。

此外，基数为 N 的对数，logN x，可以通过将 ln x 乘以常数 K 得到，其中 K = 1/ln N。

对于某些函数（atan，log，sqrt），可以应用缩放因子（参见第 43.2.4 节）来扩展函数的范围，扩展后的范围超出 q1.31 固定点格式支持的最大[-1，1]范围。对于所有其他的圆形函数，缩放因子必须设置为 0，对于双曲函数，缩放因子必须设置为 1。

43.2.2.1 Cosine

表 43-2 Cosine parameters

Parameter	Description	Range
ARG1	以弧度表示的角度 θ，并已除以 π。	[-1, 1]
ARG2	模数 m	[0, 1]
RES1	m * cosθ	[-1, 1]
RES2	m * sinθ	[-1, 1]
SCALE	不适用	0

此函数用于计算范围为-π 到π的角度的余弦值。它还可以用于执行从极坐标到直角坐标的转换。

主要参数是以弧度为单位的角度θ 。在编程 ARG1 之前，必须将其除以π 。

第二个参数是模数 m。如果 m 大于 1，则必须在软件中对其进行缩放，以使其适应 ARG2 的 q1.31 范围。

主要结果，RES1，是角度的余弦值，乘以模数 m。

次要结果，RES2，是角度的正弦值，乘以模数 m。

43.2.2.2 Sine

表 43-3 Sine parameters

Parameter	Description	Range
ARG1	以弧度表示的角度 θ 除以 π 后的值	[-1, 1]
ARG2	模数 m	[0, 1]
RES1	m * sinθ	[-1, 1]
RES2	m * cosθ	[-1, 1]

SCALE	不适用	0

此函数用于计算范围为-π 到π的角度的余弦值。它还可以用于执行从极坐标到直角坐标的转换。

主要参数是以弧度为单位的角度θ 。在编程 ARG1 之前，必须将其除以π 。

第二个参数是模数 m。如果 m 大于 1，则必须在软件中对其进行缩放，以使其适应 ARG2 的 q1.31 范围。

主要结果，RES1，是角度的正弦值，乘以模数 m。

次要结果，RES2，是角度的余弦值，乘以模数 m。

43.2.2.3 Phase

表 43-4 Phase parameters

Parameter	Description	Range
ARG1	x 坐标	[-1, 1]
ARG2	y 坐标	[-1, 1]
RES1	以弧度表示的相位角θ除以 π 后的值	[-1, 1]
RES2	模数m	[0, 1]
SCALE	不适用	0

43.2.2.4 Modulus

表 43-5 Modulus parameters

Parameter	Description	Range
ARG1	x 坐标	[-1, 1]
ARG2	y 坐标	[-1, 1]
RES1	模数m	[0, 1]
RES2	弧度表示的相位角θ除以 π 后的值	[-1, 1]
SCALE	不适用	0

这个函数计算的是向量 v = [x y]的大小，或者说模。它也可以用来执行直角坐标到极坐标的转换。

主要参数是 x 坐标，即向量在 x 轴方向上的大小。如果|x| > 1，必须在软件中应用缩放以适应 ARG1 的 q1.31 范围。

次要参数是 y 坐标，即向量在 y 轴方向上的大小。如果|y| > 1，必须在软件中应用缩放以适应 ARG2 的 q1.31 范围。

主要结果，RES1，是模数，由以下公式给出：$|v| = \sqrt{x^2 + y^2}$ 。RES1 被限制到 1。

次要结果，RES2，是向量 v 的相位角θ。RES2 必须乘以π才能得到以弧度表示的角度。请注意，由于相位角的循环性，接近π的值有时可能会绕回到-π。

43.2.2.5 Arctangent

表 43-6 Arctangent parameters

Parameter	Description	Range
ARG1	$x * 2^{-n}$	[-1, 1]
ARG2	不适用	-
RES1	以弧度表示的 $2^{-n} * \tan^{-1} x$ 除以 π 后的值	[-1, 1]
RES2	No	-

SCALE	n	[0 7]

这个函数计算的是输入参数 x 的反正切值。

主要参数 ARG1 是输入值 x=tanθ 。如果|x| > 1，必须在软件中应用 2^{-n} 的缩放因子，使得-1 < x · 2^{-n} < 1。缩放后的值 x · 2^{-n} 被编程到 ARG1 中，缩放因子 n 必须被编程到 SCALE 参数中。

请注意，允许的最大输入值是 tanθ=128，对应于角度θ= 89.55 度。对于|x| > 128，必须使用软件方法来找到 $\tan^{-1}$ x。

次要参数，ARG2，未使用。

主要结果，RES1，是角度θ= $\tan^{-1}$ x。RES1 必须乘以 2^{n} · π 才能得到以弧度表示的角度。

次要结果，RES2，未使用。

43.2.2.6 Hyperbolic cosine

表 43-7 Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	x * 2^{-n}	[-0.559 0.559]
ARG2	不适用	
RES1	2^{-n} * coshx	[0.5 0.846]
RES2	2^{-n} * sinhx	[-0.683 0.683]
SCALE	n	1

这个函数计算的是双曲角 x 的双曲余弦。它也可以用来计算指数函数 e^{x} = cosh x + sinh x，和 e^{-x} = cosh x - sinh x。

主要参数是双曲角 x。只支持在-1.118 到+1.118 范围内的 x 值。由于 cosh x 的最小值是 1，超出了 q1.31 格式的范围，必须在软件中应用 2^{-n} 的缩放因子。缩放因子 n = 1 必须被编程到 SCALE 参数中。

次要参数未使用。

主要结果，RES1，是双曲余弦，cosh x。RES1 必须乘以 2 才能得到正确的结果。

次要结果，RES2，是双曲正弦，sinh x。RES2 必须乘以 2 才能得到正确的结果。

43.2.2.7 Hyperbolic sine

表 43-8 Hyperbolic sine parameters

Parameter	Description	Range
ARG1	x * 2^{-n}	[-0.559, 0.559]
ARG2	不适用	-
RES1	2^{-n} * sinhx	[-0.683, 0.683]
RES2	2^{-n} * coshx	[0.5, 0.846]
SCALE	n	1

这个函数计算的是双曲角 x 的双曲余弦。它也可以用来计算指数函数 $e^x = \cosh x + \sinh x$，和 $e^{-x} = \cosh x - \sinh x$。

主要参数是双曲角 x。只支持在-1.118 到+1.118 范围内的 x 值。由于 cosh x 的最小值是 1，超出了 q1.31 格式的范围，必须在软件中应用 2^{-n} 的缩放因子。缩放因子 n = 1 必须被编程到 SCALE 参数中。

次要参数未使用。

主要结果，RES1，是双曲余弦，sinh x。RES1 必须乘以 2 才能得到正确的结果。

次要结果，RES2，是双曲正弦，cosh x。RES2 必须乘以 2 才能得到正确的结果。

43.2.2.8 Hyperbolic arctangent

表 43-9 Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x * 2^{-n}$	[-0.403 0.403]
ARG2	不适用	-
RES1	$2^{-n} *$ atanh x	[-0.559 0.559]
RES2	不适用	-
SCALE	n	1

这个函数计算的是输入参数 x 的双曲反正切。

主要参数是输入值 x。只支持在-0.806 到+0.806 范围内的 x 值。x 的值必须通过一个 2^{-n} 的因子进行缩放，其中 n = 1。缩放后的值 x * 0.5 被编程到 ARG1 中，缩放因子 n = 1 必须被编程到 SCALE 参数中。

次要参数未使用。

主要结果是双曲反正切值 atanh x。RES1 必须乘以 2 才能得到正确的值。

次要结果未使用。

43.2.2.9 Natural logarithm

表 43-10 Natural logarithm parameters

Parameter	Description	Range
ARG1	$x * 2^{-n}$	[0.054 0.875]
ARG2	不适用	-
RES1	$2^{-(n+1)} *$ ln x	[-0.279 0.137]
RES2	不适用	-
SCALE	n	[1 4]

这个函数计算的是输入参数 x 的自然对数。

主要参数是输入值 x。只支持在 0.107 到 9.35 范围内的 x 值。x 的值必须通过一个 2^{-n} 的因子进行缩放，使得 $x * 2^{-n} < 1 - 2^{-n}$。缩放后的值 $x * 2^{-n}$ 被编程到 ARG1 中，缩放因子 n 必须被编程到 SCALE 参数中。

下表列出了有效的缩放因子 n，以及对应的 x 和 ARG1 的范围。

表 43-11 自然对数中缩放因子及其对应的 ARG1 和 x 范围

n	x range	ARG1 range
1	0.107 ≤ x < 1	0.0535 ≤ ARG1 < 0.5
2	1 ≤ x < 3	0.25 ≤ ARG1 < 0.75
3	3 ≤ x < 7	0.375 ≤ ARG1 < 0.875

4	7 ≤ x ≤ 9.35	0.4375 ≤ ARG1 ≤ 0.584

次要参数未使用。

主要结果是自然对数，ln x。RES1 必须乘以 $2^{-(n+1)}$才能得到正确的值。

次要结果未使用。

43.2.2.10 Square root

表 43-12 Square root parameters

Parameter	Description	Range
ARG1	$x * 2^{-n}$	[0.027 0.875]
ARG2	不适用	-
RES1	$\sqrt{x} * 2^{-n}$	[0.04 1]
RES2	不适用	-
SCALE	n	[0 2]

这个函数计算的是输入参数 x 的平方根。

主要参数是输入值 x。只支持在 0.027 到 2.34 范围内的 x 值。x 的值必须通过一个 2^{-n} 的因子进行缩放，使得 $x \cdot 2^{-n} < (1-2^{(-n-2)})$。

缩放后的值 $x \cdot 2^{-n}$ 被编程到 ARG1 中，缩放因子 n 必须被编程到 SCALE 参数中。

表 43-13 平方根中缩放因子及其对应的 ARG1 和 x 范围

n	x range	ARG1 range
0	0.027 ≤ x < 0.75	0.027 ≤ ARG1 < 0.75
1	0.75 ≤ x < 1.75	0.375 ≤ ARG1 < 0.875
2	1.75 ≤ x ≤ 2.341	0.4375 ≤ ARG1 ≤ 0.585

次要参数未使用。

主要结果是 x 的平方根。RES1 必须乘以 2^{-n} 才能得到正确的值。

次要结果未使用。

43.2.3 定点表示法

CORDIC 以定点有符号整数格式进行运算。输入和输出值可以是 q1.31 或 q1.15 格式。

在 q1.31 格式中，数字由一个符号位和 31 个小数位（二进制小数位）表示。因此，数值范围是-1（0x80000000）到 $1 - 2^{-31}$（0x7FFFFFFF)。

在 q1.15 格式中，数值范围是-1（0x8000）到 $1 - 2^{-15}$（0x7FFF)。这种格式的优点是两个 q1.15 输入参数可以打包成一个 32 位的写入，两个结果可以在一个 32 位的读取中获取。

43.2.4 缩放因子

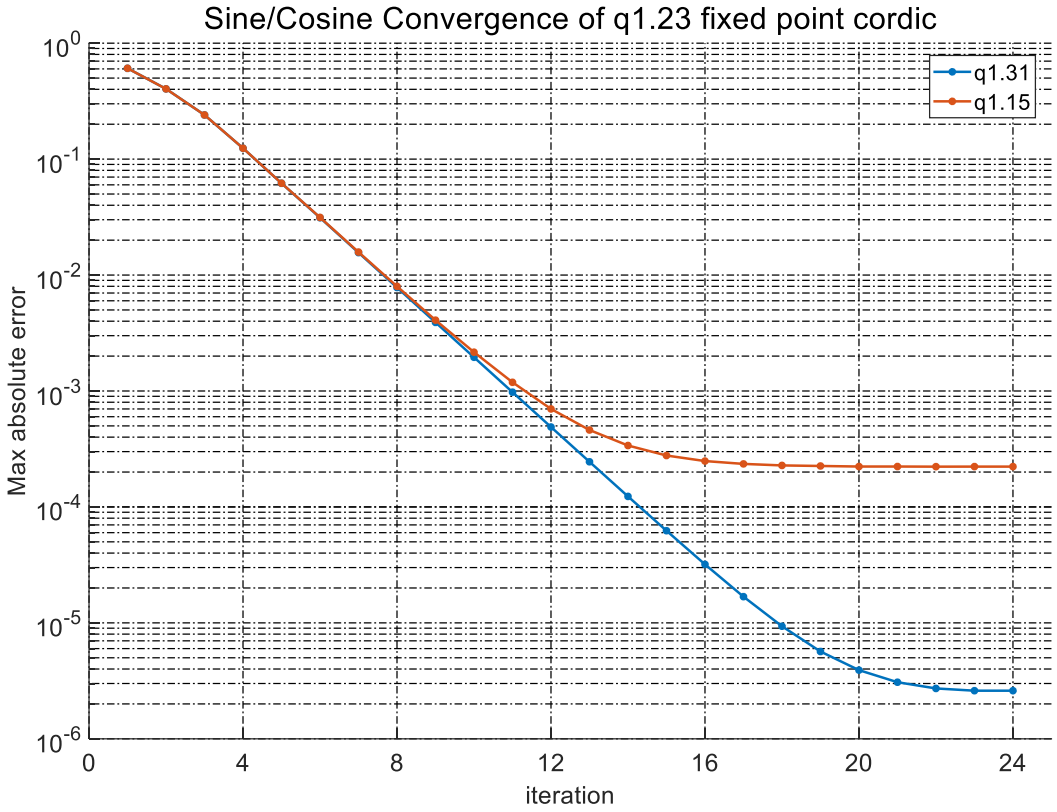
在 CORDIC 函数章节列出的函数中有几个指定了一个缩放因子 SCALE。这允许函数的输入范围扩展到 CORDIC 支持的范围，而不会使输入、输出或内部寄存器溢出。如果需要缩放因子，必须在软件中计算并编程到 CORDIC_CSR 寄存器的 SCALE 字段中。在写入 CORDIC_WDATA 寄存器之前，必须相应地对输入参数进行缩放。从 CORDIC_RDATA 寄存器读取结果后，也必须对结果做缩放。

注意：缩放因子会导致缩放值精度损失。

43.2.5 精度

结果的精度取决于 CORDIC 迭代的次数。见下图。

CORDIC 可以在每个时钟周期执行两次迭代。对于每个函数，下表表示每四次迭代后剩余的最大误差，



以及达到该精度所需的时钟周期数。根据该表，可以在 CORDIC_CSR 寄存器的精度字段中确定和编程所需的周期数。

一旦完成配置的迭代次数，协处理器立即停止，并且可以立即读取结果。

表 43-14 精度与迭代次数

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}

Sqrt[4]	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. 最大残差误差是在给定迭代次数后与双精度浮点数执行相同计算的最大误差。可能会有额外的舍入误差，对于 q15 格式最高为 2^-16，对于 q31 格式最高为 2^-20。
2. 对于模数大于 0.5，可实现的精度随模数的大小成比例降低，因为量化误差变得显著。
3. SCALE = 1。如果使用更高的缩放因子，则可实现的精度成比例降低。
4. SCALE = 0。如果使用更高的缩放因子，则可实现的精度成比例降低。

43.2.6 Zero-overhead mode

使用协处理器的最快方式是预先编程 CORDIC_CSR 寄存器，设置要执行的函数（FUNC）、所需的时钟周期数（PRECISION）、输入和输出值的大小（ARGSIZE, RESSIZE）、输入参数的数量（NARGS）和/或结果的数量（NRES），以及如果需要的话，设置缩放因子（SCALE）。然后，通过将输入参数写入 CORDIC_WDATA 寄存器来触发计算。一旦写入了正确数量的输入参数（并且任何正在进行的计算已经完成），就会使用这些输入参数和当前的 CORDIC_CSR 设置启动新的计算。如果没有变化，就不需要重新编程 CORDIC_CSR 寄存器。

如果需要双 32 位输入参数（ARGSIZE = 0, NARGS = 1），则必须先写入主输入参数 ARG1，然后写入次输入参数 ARG2。如果在一系列计算中，次参数保持不变，那么可以通过在第一次计算开始后，将参数数量重新编程为一（NARGS = 0）来避免第二次写入。只要 FUNC 没有改变，次参数就会保持其编程值。

注：复位后，ARG2 设置为 +1（0x7FFFFFFF)。

如果使用两个 16 位参数（ARGSIZE = 1)，则必须将它们打包到 32 位字中，其中 ARG1 位于最低有效半字中，ARG2 位于最高有效半字中。然后将打包的 32 位字写入 CORDIC_WDATA 寄存器。在这种情况下只需要一次写入（NARGS = 0)。

对于仅采用一个输入参数 ARG1 的函数，建议设置 NARGS = 0。

如果 NARGS = 1，则必须对 CORDIC_WDATA 执行第二次写入才能触发计算。在这种情况下不使用 ARG2 数据。

一旦计算开始，任何读取 CORDIC_RDATA 寄存器的尝试都会插入总线等待状态，直到计算完成，然后返回结果。因此，软件可以写入输入并立即读取结果，而无需轮询以查看其是否有效。或者，处理器可以在读取结果之前等待适当数量的时钟周期。如果需要，这段时间可用于对 CORDIC_CSR 寄存器进行编程以进行下一次计算并准备下一个输入数据。CORDIC_CSR 寄存器可以在计算过程中重新编程，而不会影响正在进行的计算的结果。以同样的方式，一旦配置了前面的参数，就可以使用下一个参数更新 CORDIC_WDATA 寄存器。下一个参数和设置直到上一个计算完成后生效。

计算完成后，可以从 CORDIC_RDATA 寄存器读取结果。

如果需要两个 32 位结果（NRES = 1，RESSIZE = 0），则首先读取主要结果（RES1)，然后读取辅助结果(RES2)。如果仅期望一个 32 位结果(NRES = 0, RESSIZE = 0)，则只需读取主要结果(RES1)。如果需要 16 位结果（RESSIZE = 1），则对 CORDIC_RDATA 的单次读取会获取打包到 32 位字中的两个结果。RES1 位于下半字, RES2 位于上半字。在这种情况下，建议编程 NRES = 0。如果 NRES = 1，则必须执行 CORDIC_RDATA 的第二次读取，以便释放 CORDIC 以进行下一个操作。第二次读取的数据必须被丢弃。

如果已写入预期数量的参数，则在读取预期数量的结果后开始下一次计算。这意味着在任何时候，都可能有一项计算正在进行中，或者正在等待读取结果，以及一项待处理的操作。在有操作处于待处理状态时，如果进一步访问 CORDIC_WDATA，将会取消该待处理操作并覆盖相关数据。

以下序列总结了 CORDIC 在零开销模式下的使用：

1. 设置 CORDIC_CSR 寄存器。
2. 在 CORDIC_WDATA 寄存器中对第一次计算的参数进行配置。这将启动第一次计算。
3. 如果需要，更新 CORDIC_CSR 寄存器设置下一次计算的配置。
4. 从 CORDIC_RDATA 寄存器读取结果。
5. 在 CORDIC_WDATA 寄存器中对下一次计算的参数进行配置。
6. 转到步骤 3。

43.2.7 Polling mode

当 CORDIC_RDATA 寄存器中有新结果时，CORDIC_CSR 寄存器中的 RRDY 标志会被置位。可以通过读取寄存器来轮询该标志。通过读取 CORDIC_RDATA 寄存器（一次或两次，具体取决于 CORDIC_CSR 寄存器的 NRES 字段）来重置它。

当结果不在可用时立即读取，轮询 RRDY 标志比直接读取 CORDIC_RDATA 寄存器花费的时间稍长。

43.2.8 Interrupt mode

通过设置 CORDIC_CSR 寄存器中的中断使能（IE）位，只要产生 RRDY 标志，就会生成中断。当标志复位时，中断被清除。

该模式允许在中断服务程序下读取计算结果，因此相对于其他任务具有优先级。然而，由于中断处理延迟，它比直接读取结果或轮询标志慢。

43.2.9 DMA mode

如果 CORDIC_CSR 寄存器中的 DMA 写使能（DMAWEN）位被置位，并且没有待处理的操作，则会生成 DMA 写请求。DMA 控制器可以将主输入参数（ARG1）传输到 CORDIC_WDATA 寄存器。

如果 CORDIC_CSR 寄存器中的 NARGS = 1，则会生成第二个 DMA 写入请求，以将辅助输入参数（ARG2）传输到 CORDIC_WDATA 寄存器中。当所有输入参数都已写入，并且任何正在进行的计算已完成（通过读取结果）时，将开始新的计算并生成另一个 DMA 写入请求。

如果 CORDIC_CSR 寄存器中的 DMA 读使能（DMAREN）位被置位，则 RRDY 标志置位时会生成 DMA 读请求。然后，DMA 控制器可以将主要结果（RES1）从 CORDIC_RDATA 寄存器读出。如果 CORDIC_CSR 寄存器中的 NRES = 1，则会生成第二个 DMA 请求以读出辅助结果（RES2)。当所有结果都被读取后，RRDY 标志被清除。

注: 由于 DMA 执行对 CORDIC_WDATA 或 CORDIC_RDATA 寄存器的访问，每个 DMA 请求都必须得到确认。因此，当 DMA 读使能时，必须避免 CPU 访问 CORDIC_RDATA 寄存器。同样，当 DMA 写入使能时，处理器必须避免访问 CORDIC_WDATA 寄存器

43.3 寄存器描述

寄存器宽度为 32bits。

43.3.1 CORDIC 控制/状态寄存器（CORDIC_CSR)

Address:0x00

Reset value:0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY.	Res								ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN
R									RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					RW			RW				RW			

Bit	Name	R/W	Reset Value	Function
31	RRDY	R	1'b0	结果就绪标志： 0：输出寄存器中无新数据 。 1：CORDIC_RDATA 寄存器包含新数据。 当 CORDIC 操作完成时，该位由硬件置位。当 CORDIC_RDATA 寄存器被读取（NRES+1）次时，它会被硬件复位。
30:23	保留	-	-	必须保持复位值
22	ARGSIZE	RW	1'b0	输入数据的宽度: 0：32 位 1：16 位 ARGSIZE 选择用于表示输入数据的位数。 如果选择 32 位数据，CORDIC_WDATA 寄存器需要 q1.31 格式的参数。 如果选择 16 位数据，CORDIC_WDATA 寄存器需要 q1.15 格式的参数。 主要参数（ARG1）写入最低有效半字，次要参数（ARG2）写入最高有效半字。
21	RESSIZE	RW	1'b0	输出数据的宽度 ： 0：32 位 1：16 位 RESSIZE 选择用于表示输出数据的位数。如果选择 32 位数据，CORDIC_RDATA 寄存器包含 q1.31 格式的结果。如果选择 16 位数据，则 CORDIC_RDATA 的最低有效半字包含 q1.15 格式的主要结果（RES1），最高有效半字包含同样为 q1.15 格式的次要结果（RES2）。
20	NARGS	RW	1'b0	CORDIC_WDATA 寄存器期望的参数数量: 0：下一次计算只需要一次 32 位写入（如果 ARGSIZE = 1，则需要两个 16 位值）。 1：必须将两个 32 位值写入 CORDIC_WDATA 寄存器才能触发下一次计算。 读取返回该位的当前状态。
19	NRES	RW	1'b0	CORDIC_RDATA 寄存器中的结果数量 ：

				0：下一次计算完成时，仅将一个 32 位值（如果 RESSIZE = 1，则为两个 16 位值）传输到 CORDIC_RDATA 寄存器。从 CORDIC_RDATA 读取一次会重置 RRDY 标志。 1：下一次计算完成时，两个 32 位值将传输到 CORDIC_RDATA 寄存器。需要从 CORDIC_RDATA 读取两次才能重置 RRDY 标志。 读取返回该位的当前状态。
18	DMAWEN	RW	1'b0	启用 DMA 写通道 ： 0：禁用。不生成 DMA 写请求。 1：启用。只要没有挂起的操作，就会在 DMA 写入通道上生成请求。 该位由软件置位和清零。读取返回该位的当前状态。
17	DMAREN	RW	1'b0	启用 DMA 读取通道: 0：禁用。不生成 DMA 读取请求。 1：启用。只要 RRDY 标志置位，就会在 DMA 读取通道上生成请求。 该位由软件置位和清零。读取返回该位的当前状态。
16	IEN	RW	1'b0	启用中断: 0：禁用。不产生中断请求。 1：启用。每当 RRDY 标志置位时，就会产生中断请求。 该位由软件置位和清零。读取返回该位的当前状态。
15:11	保留	-	-	必须保持复位值
10:8	SCALE[2:0]	RW	3'b0	缩放因子 该字段的值为应用于参数和/或结果的缩放因子。值 n 意味着参数已乘以因子 2^n，并且结果需要乘以 2n。请参阅相关章节了解每个函数的比例因子的适用性以及适当的范围
7:4	PRECISION[3:0]	RW	4'b0101	所需精度（迭代次数）： 1 到 6：(迭代次数)/4 。 则迭代次数为 PRECISION 乘以 4。 要确定给定精度所需的迭代次数，请参阅相关表格。 请注意，对于大多数函数，该字段的建议范围是 3 到 6 另外当 PRECISION=0 或者大于 6 时，迭代次数为 24 次
3:0	FUNC[3:0]	RW	4'b0	功能: 0：Cosine 1：Sine 2：Phase 3：Modulus 4：Arctangent 5：Hyperbolic cosine 6：Hyperbolic sine 7：Arctanh 8：Natural logarithm 9：Square Root 10 to 15：保留

43.3.2 CORDIC 参数寄存器（CORDIC_WDATA)

Address:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	ARG[31:16]	W	16h'0	函数输入参数: 该寄存器使用 CORDIC_CSR 寄存器 FUNC 字段中所选函数的输入参数进行编程。 如果选择 32 位格式（CORDIC_CSR.ARGSIZE = 0）并且需要两个输入参数（CORDIC_CSR.NARGS = 1），则需要对该寄存器进行两次连续写入。 第一个写入主要参数（ARG1），第二个写入次要参数（ARG2)。 如果选择 32 位格式并且只需要一个输入参数（NARGS = 0），则只需要对此寄存器进行一次写入，其中包含主参数（ARG1)。 如果选择 16 位格式（CORDIC_CSR.ARGSIZE = 1），则对该寄存器的一次写入包含两个参数。主要参数（ARG1）位于下半部分 ARG[15:0]，次要参数（ARG2）位于上半部分 ARG[31:16]。在这种情况下，NARGS 必须设置为 0。 请参阅 CORDIC 函数 章节了解每个函数所需的参数及其允许的范围。 写入所需数量的参数后，CORDIC 将使用提供的输入参数评估 CORDIC_CSR.FUNC 指定的函数，前提是之前的计算已完成。如果计算正在进行，则 ARG1 和 ARG 2 值将保持待定状态，直到计算完成并读取结果。在此期间，对寄存器的写入会取消挂起的操作并覆盖参数数据。
15:0	ARG[15:0]	W	16h'0	

43.3.3 CORDIC 结果寄存器（CORDIC_RDATA）

Address:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	RES[31:16]	R	16h’0	函数结果 如果选择 32 位格式（CORDIC_CSR.RESSIZE = 0）并且需要两个输出值（CORDIC_CSR.NRES = 1），则当 RRDY 标志置位时，必须读取该寄存器两次。第一次读取获取主要结果（RES1）。第二次读取获取辅助结果（RES2）并重置 RRDY。 如果选择 32 位格式并且只需要一个输出值（NRES = 0），则只需读取该寄存器一次即可获取主要结果（RES1）并复位 RRDY 标志。 如果选择 16 位格式（CORDIC_CSR.RESSIZE = 1），则该寄存器包含下半部分的主要结果（RES1）RES[15:0]，以及上半部分的辅助结果（RES2）RES[31:16]。在这种情况下，NRES 必须设置为 0，并且只执行一次读取。 从此寄存器读取数据会重置 CORDIC_CSR 寄存器中的 RRDY 标志
15:0	RES[15:0]	R	16h’0	

44. 6800/8080 控制器（LCDC）

44.1 概述

LCDC 用于驱动 8080/6800 协议的 LCD 显示屏。

LCDC 模块可以灵活地写指令/数据到 LCD 显示屏，也可以灵活地从 LCD 显示屏读取数据。

LCDC 模块能对读/写操作分别配置时序（读/写驱动脉冲的时长可分别配置），以适应不同的 LCD 显示屏。

44.2 主要特性

- 可编程的读写脉冲宽度
- 支持数据位宽小于或等于 18 位的 8080/6800 LCD 显示屏，包括 8/9/16/18 等常见数据位宽
- 最多支持驱动 4 个 LCD 显示屏（以分时复用的方式）,4 个显示屏的时序参数，协议可分别配置
- 对数据寄存器的读写就等同于对 LCD 的读写
- 支持利用 DMA 向 LCD 大量地写入数据，而不占用 CPU 资源

44.3 功能描述

44.3.1 使能 LCDC

LCDC_CSR 的 EN0~3 位被写 1，即为使能 LCD0~3；LCDC_CSR 的 EN0~3 位被写 0，即为失能 LCD0~3。

如果 LCDC 处于失能状态，往 LCDx_DR/LCDx_CMDR(x=0~3)寄存器写入数据不会触发对 LCD 的写数据/指令操作，从 LCDx_DR 寄存器读取的数据是无效值，且不会触发对 LCD 的读数据操作。

CS 信号线 (片选线) 的控制逻辑: 当 LCDx 工作时, CSx 处于低电平 (此时, 其他 CS 线处于高电平)，LCDx 不工作时，CSx 处于高电平。

44.3.2 工作模式

LCD_CSR 的 MODEx 位为 1 的时候，LCDx 工作在 6800 模式，反之则为 8080 模式。复位后，LCDx 默认工作在 8080 模式。

44.3.3 发送 LCD 数据

如果 LCDx 处于使能状态，只需要写 LCDx_DR 寄存器即可触发 LCDx 数据发送，引脚 LCDC_RS 为高电平，D[17:0]会被识别为数据。

读写 LCDx_DR，时序参数将会依照 LCDx_WCFGR(写操作)和 LCDx_RCFGR(读操作)进行配置。

当 LCD 使能且 DMA 模式使能时，如果 LCDC 没有传输数据，就会向 DMA 发送写请求，如果 DMA 响应了写请求，则 LCDC 会将 DMA 发送来的数据作为 LCD 写数据，并发送到引脚。当 LCDC 正在传输数据的时候，LCDC 会停止发送写请求。

44.3.4 发送 LCD 指令

如果 LCDx 处于使能状态，只需要写 LCDx_CMDR 寄存器即可触发 LCDx 指令发送，引脚 LCDC_RS 为低电平，D[17:0]会被识别为指令。

读写 LCDx_CMDR，时序参数将会依照 LCDx_WCFGR(写操作)和 LCDx_RCFGR(读操作) 进行配置。

如果 LCDx 的处于使能状态，只需要读 LCDx_DR 寄存器即可读取 LCD 显示屏。

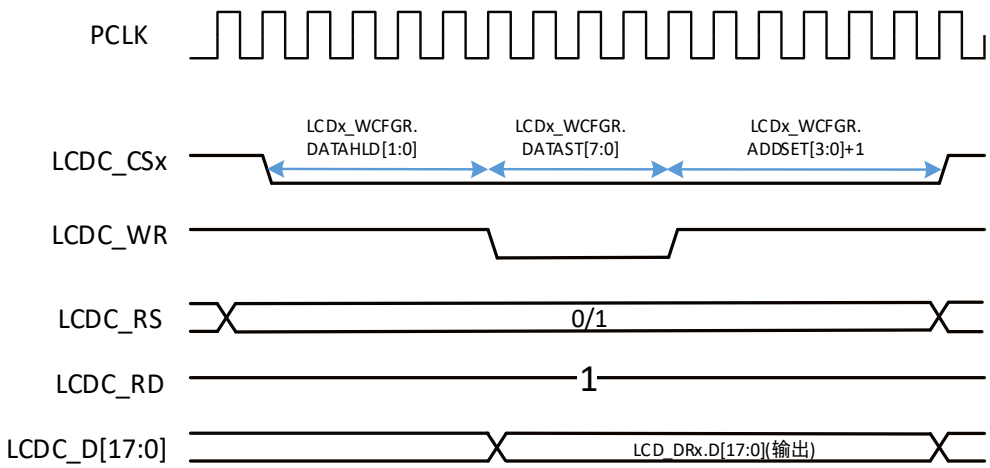
44.3.5 从 LCD 读取指令

LCDC 支持读指令操作，当 LCDx 处于使能状态，只需读取 LCDx_CMDR 寄存器即可触发 LCDx 指令读取，与数据读取不同的时，读取指令时，引脚 LCDC_RS 低电平。

注意：某些型号的 LCD 可直接用此方式读取设备 ID 或状态寄存器，而无需提前写入指令。

44.3.6 LCDC 时序

(1) LCDx 8080 写数据/指令时序

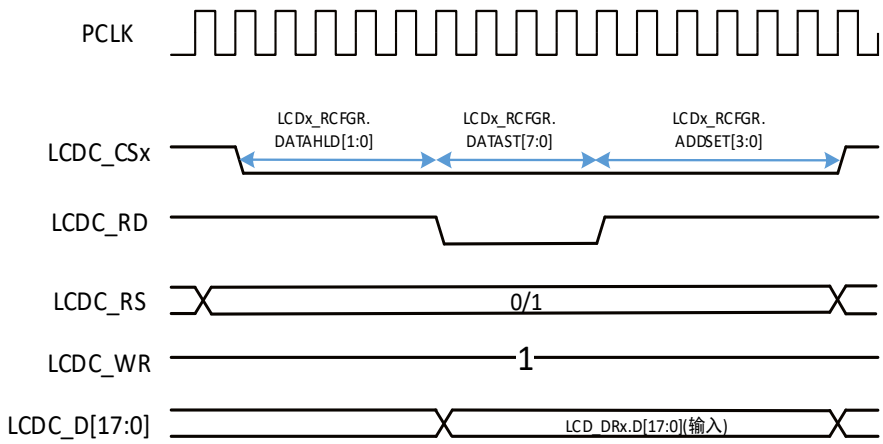


当 LCDx _DR 被写时，触发一次 8080/6800 写数据操作；当 LCDx_CMDR 被写时，触发一次 8080/6800 写命令操作。如果 LCD_CSR 寄存器 MODEx=0，则触发的是 8080 时序，反之则为 6800 时序。

当 LCDC 模块正在向 LCD 写数据/命令时，busy=1，LCDC 的任何寄存器不可在此期间被写，否则写数据会被忽略。读取 LCDC_CSR[17]可获知 busy 状态。

上图的 LCDC_WR 即为微处理器的 LCDC_8WR_6E 引脚；LCDC_RD 即为微处理器的 LCDC_8RD_6RW 引脚。

(2) LCDC 8080 读数据/指令时序

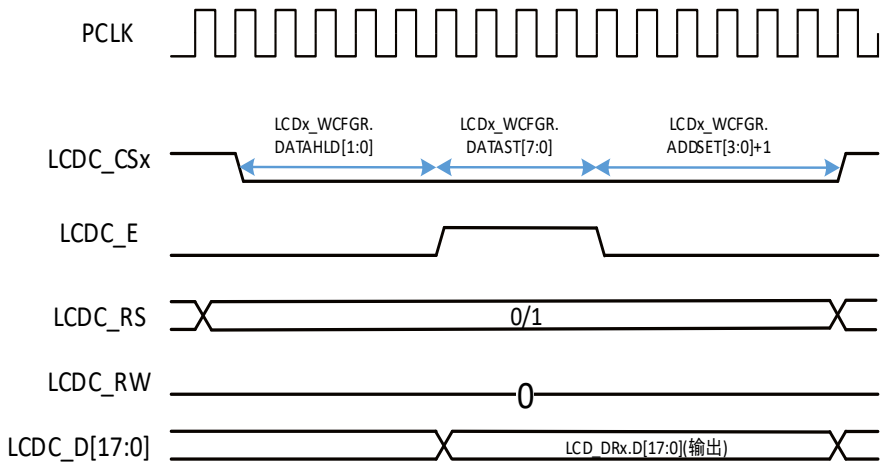


当 LCDx_DR 被读时，触发一次 8080/6800 读数据操作；当 LCDx_CMDR 被读时，触发一次 8080/6800 读命令操作。如果 LCD_CSR 寄存器 MODEx=0，则触发的是 8080 时序，反之则为 6800 时序。

当 LCDC 模块在从 LCD 读数据前，APB 总线无法访问 LCDC 的任何寄存器。

上图的 LCDC_RD 即为微处理器的 LCDC_8RD_6RW 引脚；LCDC_WR 即为微处理器的 LCDC_8WR_6E 引脚。

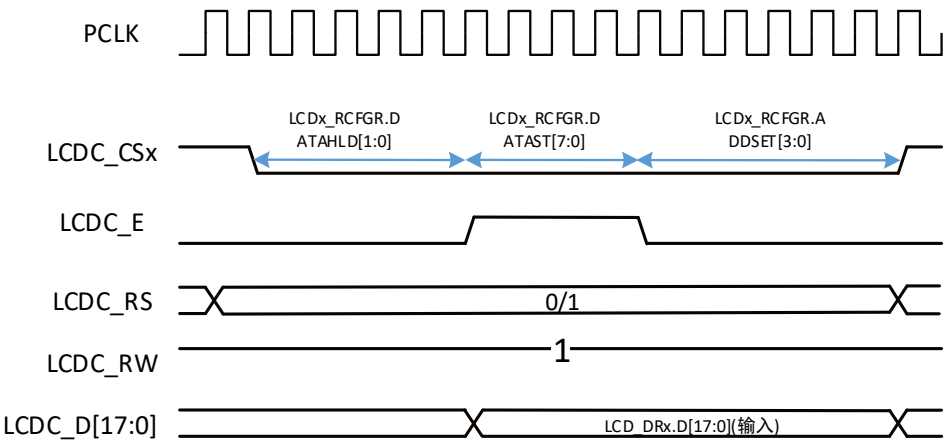
(3) LCDC 6800 写数据/指令时序



触发方式与 8080 一致，需要设置 LCD_CSR 寄存器 MODEx=1。

上图的 LCDC_E 即为微处理器的 LCDC_8WR_6E 引脚；LCDC_RW 即为微处理器的 LCDC_8RD_6RW 引脚。

(4) LCDC 6800 读数据/指令时序



触发方式与 8080 一致，需要设置 LCD_CSR 寄存器 MODEx=1。

上图的 LCDC_E 即为微处理器的 LCDC_8WR_6E 引脚；LCDC_RW 即为微处理器的 LCDC_8RD_6RW 引脚。

44.3.7 DMA 请求

DMA 响应 LCDC 传输请求后会将写数据发送到 LCDC 寄存器，LCDC 开始发送数据。

(1) 外设地址

DMA 功能仅用于 LCD0_DR~LCD3_DR 的写数据。发送指令，指令参数，时序参数必须用 CPU。

(2) 数据宽度

LCDC 模块支持字节，半字，字的 DMA 传输。LCDC 寄存器里，LCD0_DR~LCD3_DR 支持字节，半字，字写，其余寄存器仅支持字写。

当 DMA 或 CPU 将半字数据写到 LCDx_DR 时，引脚 LCDC_D[17:16]会发送低电平；当 DMA 或 CPU 将字节数据写到 LCDx_DR 时，引脚 LCDC_D [17:8]会发送低电平。

(3) 适用范围

LCDC 的 DMA 仅有写请求，没有读请求。读数据时，将较长时间地占用 CPU 资源。

当 DMA 发送未完成时，CPU 不能访问 LCDC 的任何寄存器，不能保证 LCDC 动作。

LCDC 仅支持 1 路 DMA 传输。当 LCDC 开启 DMA 传输后，不支持多个 DMA 通道写入 LCDC，也不支持 DMA 写入多个 LCDC 寄存器。

44.3.8 LCDC 忙机制

当 LCD 正在传输数据的时候（读或写数据/指令），LCDC 将处于忙状态，LCD_CSR[17]=1（busy 位等于 1）。

当 LCD 处于 busy 状态时，LCDC 任何寄存器不允许写操作，否则写数据会被忽略。

如果 CPU 访问 LCDx_DR/LCDx_CMDR，则当 LCD 处于 busy 状态时，数据寄存器 LCDx_DR 不允许读，否则读出数据固定为 0，但允许读 LCDC_CSR，LCDx_WCFGR，LCDx_RCFGR 寄存器。

44.4 寄存器描述

注意：仅 LCD0_DR~LCD3_DR 支持字节，半字，字写，其余寄存器仅支持字写。所有寄存器仅支持字读。

44.4.1 LCDC 控制与状态寄存器(LCDC_CSR)

Address offset:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.														BUSY	DMAEN
-														R	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.								MODE3	MODE2	MODE1	MODE0	EN3	EN2	EN1	EN0
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17	BUSY	R	0	LCDC 忙标志 0：LCDC 空闲 1：LCDC 忙 参考章节 LCDC 忙机制
16	DMAEN	RW	0	LCDC DMA 使能 0：LCDC DMA 关闭 1：LCDC DMA 开启
15:8	保留	-	-	-
7	MODE3	RW	0	LCD3 接口时序 0：LCD3 接口时序是 8080 时序 1：LCD3 接口时序是 6800 时序
6	MODE2	RW	0	LCD2 接口时序 0：LCD2 接口时序是 8080 时序 1：LCD2 接口时序是 6800 时序
5	MODE1	RW	0	LCD1 接口时序 0：LCD1 接口时序是 8080 时序 1：LCD1 接口时序是 6800 时序
4	MODE0	RW	0	LCD0 接口时序 0：LCD0 接口时序是 8080 时序 1：LCD0 接口时序是 6800 时序
3	EN3	RW	0	LCD3 使能 0：LCD3 不使能 1：LCD3 使能
2	EN2	RW	0	LCD2 使能 0：LCD2 不使能 1：LCD2 使能
1	EN1	RW	0	LCD1 使能 0：LCD1 不使能 1：LCD1 使能
0	EN0	RW	0	LCD0 使能 0：LCD0 不使能

				1：LCD0 使能

44.4.2 LCD0 数据寄存器(LCD0_DR)

Address offset:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	D[17:0]	RW	18'h0	LCD0 数据寄存器

44.4.3 LCD0 命令寄存器(LCD0_CMDR)

Address offset:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	CMD[17:0]	RW	18'h0	LCD0 指令寄存器

44.4.4 LCD0 写时序配置寄存器(LCD0_WCFGR)

Address offset:0x0C

Reset value:0x0000 FF0F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		Res.													
RW	RW	-													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								Res.				ADDSET[3:0]			
RW	RW	RW	RW	rwr	RW	RW	RW	-				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD0 写数据保持时间 00：写数据保持时间是 1 个 APB 周期 01：写数据保持时间是 2 个 APB 周期 10：写数据保持时间是 3 个 APB 周期 11：写数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD0 写数据持续时间 0000 0000：写数据持续时间是 1 个 APB 周期 0000 0001：写数据持续时间是 2 个 APB 周期 0000 0010：写数据持续时间是 3 个 APB 周期 … 1111 1111：写数据持续时间是 256 个 APB 周期

7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD0 写地址建立持续时间 0000：写地址建立时间是 1 个 APB 周期 0001：写地址建立时间是 2 个 APB 周期 … 1111：写地址建立时间是 16 个 APB 周期

44.4.5 LCD0 读时序配置寄存器(LCD0_RCFGR)

Address offset:0x10

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD0 读数据保持时间 00：读数据保持时间是 1 个 APB 周期 01：读数据保持时间是 2 个 APB 周期 10：读数据保持时间是 3 个 APB 周期 11：读数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD0 读数据持续时间 0000 0000：读数据持续时间是 1 个 APB 周期 0000 0001：读数据持续时间是 2 个 APB 周期 0000 0010：读数据持续时间是 3 个 APB 周期 … 1111 1111：读数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD0 读地址建立持续时间 0000：读地址建立时间是 1 个 APB 周期 0001：读地址建立时间是 2 个 APB 周期 … 1111：读地址建立时间是 16 个 APB 周期

44.4.6 LCD1 数据寄存器(LCD1_DR)

Address offset:0x14

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-

17:0	D[17:0]	RW	18'h0	LCD1 数据寄存器 APB 总线写本寄存器时，LCDC 模块会把 D[17:0]视作 LCD 数据，按照特定的时序（8080 或 6800，取决于 LCDC_CSR 的 MODE1 位）发送到 LCD。 APB 总线读取本寄存器时，LCDC 模块会按照特定的时序读取 LCD 的数据，并发送到 APB 总线。 本寄存器支持 8，16，32 位写，但仅支持 32 位读 由读写本寄存器所产生的 8080/6800 时序里，只有 CS1 片选线工作（低电平有效）,其余片选线不工作（维持高电平）。

44.4.7 LCD1 命令寄存器(LCD1_CMDR)

Address offset:0x18

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	CMD[17:0]	RW	18'h0	LCD1 指令寄存器

44.4.8 LCD1 写时序配置寄存器(LCD1_WCFGR)

Address offset:0x1C

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD1 写数据保持时间 00：写数据保持时间是 1 个 APB 周期 01：写数据保持时间是 2 个 APB 周期 10：写数据保持时间是 3 个 APB 周期 11：写数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD1 写数据持续时间 0000 0000：写数据持续时间是 1 个 APB 周期 0000 0001：写数据持续时间是 2 个 APB 周期 0000 0010：写数据持续时间是 3 个 APB 周期 … 1111 1111：写数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD1 写地址建立持续时间 0000：写地址建立时间是 1 个 APB 周期

				0001：写地址建立时间是 2 个 APB 周期 … 1111：写地址建立时间是 16 个 APB 周期

44.4.9 LCD1 读时序配置寄存器(LCD1_RCFGR)

Address offset:0x20

Reset value:0x0000 FF0F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		Res.													
RW	RW	-													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								Res.				ADDSET[3:0]			
RW	RW	RW	RW	rwr	RW	RW	RW	-				RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2’h0	LCD1 读数据保持时间 00：读数据保持时间是 1 个 APB 周期 01：读数据保持时间是 2 个 APB 周期 10：读数据保持时间是 3 个 APB 周期 11：读数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8’hff	LCD1 读数据持续时间 0000 0000：读数据持续时间是 1 个 APB 周期 0000 0001：读数据持续时间是 2 个 APB 周期 0000 0010：读数据持续时间是 3 个 APB 周期 … 1111 1111：读数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4’hf	LCD1 读地址建立持续时间 0000：读地址建立时间是 1 个 APB 周期 0001：读地址建立时间是 2 个 APB 周期 … 1111：读地址建立时间是 16 个 APB 周期

44.4.10 LCD2 数据寄存器(LCD2_DR)

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	D[17:0]	RW	18’h0	LCD2 数据寄存器

44.4.11 LCD2 命令寄存器(LCD2_CMDR)

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	CMD[17:0]	RW	18'h0	LCD2 指令寄存器

44.4.12 LCD2 写时序配置寄存器(LCD2_WCFGR)

Address offset:0x2C

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD2 写数据保持时间 00：写数据保持时间是 1 个 APB 周期 01：写数据保持时间是 2 个 APB 周期 10：写数据保持时间是 3 个 APB 周期 11：写数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD2 写数据持续时间 0000 0000：写数据持续时间是 1 个 APB 周期 0000 0001：写数据持续时间是 2 个 APB 周期 0000 0010：写数据持续时间是 3 个 APB 周期 … 1111 1111：写数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD2 写地址建立持续时间 0000：写地址建立时间是 1 个 APB 周期 0001：写地址建立时间是 2 个 APB 周期 … 1111：写地址建立时间是 16 个 APB 周期

44.4.13 LCD2 读时序配置寄存器(LCD2_RCFGR)

Address offset:0x30

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD2 读数据保持时间 00：读数据保持时间是 1 个 APB 周期 01：读数据保持时间是 2 个 APB 周期 10：读数据保持时间是 3 个 APB 周期 11：读数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD2 读数据持续时间 0000 0000：读数据持续时间是 1 个 APB 周期 0000 0001：读数据持续时间是 2 个 APB 周期 0000 0010：读数据持续时间是 3 个 APB 周期 … 1111 1111：读数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD2 读地址建立持续时间 0000：读地址建立时间是 1 个 APB 周期 0001：读地址建立时间是 2 个 APB 周期 … 1111：读地址建立时间是 16 个 APB 周期

44.4.14 LCD3 数据寄存器(LCD3_DR)

Address offset:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	D[17:0]	RW	18'h0	LCD3 数据寄存器

44.4.15 LCD2 命令寄存器(LCD3_CMDR)

Address offset:0x38

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:18	保留	-	-	-
17:0	CMD[17:0]	RW	18'h0	LCD3 指令寄存器.

44.4.16 LCD3 写时序配置寄存器(LCD3_WCFGR)

Address offset:0x3C

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD3 写数据保持时间 00：写数据保持时间是 1 个 APB 周期 01：写数据保持时间是 2 个 APB 周期 10：写数据保持时间是 3 个 APB 周期 11：写数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD3 写数据持续时间 0000 0000：写数据持续时间是 1 个 APB 周期 0000 0001：写数据持续时间是 2 个 APB 周期 0000 0010：写数据持续时间是 3 个 APB 周期 … 1111 1111：写数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD3 写地址建立持续时间 0000：写地址建立时间是 1 个 APB 周期 0001：写地址建立时间是 2 个 APB 周期 … 1111：写地址建立时间是 16 个 APB 周期

44.4.17 LCD3 读时序配置寄存器(LCD3_RCFGR)

Address offset:0x40

Reset value:0x0000 FF0F

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

Bit	Name	R/W	Reset Value	Function
31:30	DATAHLD[1:0]	RW	2'h0	LCD3 读数据保持时间 00：读数据保持时间是 1 个 APB 周期 01：读数据保持时间是 2 个 APB 周期 10：读数据保持时间是 3 个 APB 周期 11：读数据保持时间是 4 个 APB 周期
29:16	保留	-	-	-
15:8	DATAST[7:0]	RW	8'hff	LCD3 读数据持续时间 0000 0000：读数据持续时间是 1 个 APB 周期 0000 0001：读数据持续时间是 2 个 APB 周期 0000 0010：读数据持续时间是 3 个 APB 周期 … 1111 1111：读数据持续时间是 256 个 APB 周期
7:4	保留	-	-	-
3:0	ADDSET[3:0]	RW	4'hf	LCD3 读地址建立持续时间 0000：读地址建立时间是 1 个 APB 周期 0001：读地址建立时间是 2 个 APB 周期 … 1111：读地址建立时间是 16 个 APB 周期

45. MCU 调试接口

45.1 简介

MCUDBG 模块给调试者提供以下支持：

- 低功耗模式
- 定时器、看门狗、I²C 和 CAN 的时钟控制
- MCUDBG 寄存器还提供芯片 ID 的编码。使用 JTAG 或者 SW 调试接口，或者用户程序都可以访问此 ID 编码。

45.2 主要特性

- 支持睡眠模式，停止模式和待机模式的调试
- CPU 进入 HALT 时，控制定时器、看门狗停止计数或者继续计数
- CPU 进入 HALT 时，阻止 I²C 的 SMBUS 超时
- 启用 CAN 的读取时禁止修改功能

45.3 功能描述

45.3.1 支持调试低功耗模式

用户可以通过执行 WFI 和 WFE 指令进入低功耗模式。

MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的功耗。

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 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK 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 HSI16 时钟，在停止模式下为 FCLK 和 HCLK 提供时钟。
- 在待机模式下，调试器必须先置位 DBG_STDBY 位。这将激活 HSI16 时钟，在待机模式下为 FCLK 和 HCLK 提供时钟。

45.4 寄存器描述

MCUDBG 寄存器映射在外部 PPB 总线。

45.4.1 ID 编码（DBGMCU_IDCODE）

该寄存器由上电复位，系统复位不会复位该寄存器。当内核处于复位状态下时，调试器可以访问该寄存器。

Address offset：0x00

Reset value：0x4444BBBB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[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**
REV_ID[15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:0	REV_ID	R	32’h4444BBBB	MCU 产品 ID。该值保存在 FLASH information 区。

45.4.2 调试配置寄存器（DBGMCU_CR1）

该寄存器由上电复位，系统复位不会复位该寄存器。当内核处于复位状态下时，调试器可以访问该寄存器。

Address offset：0x04

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	Res				DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT
-	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**
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	Res					DBG_STDBY	DBG_STOP	DBG_SLEEP
RW	RW	RW	RW	RW	RW	RW	RW	-					RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	-
30	DBG_TIM11_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM11 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
29	DBG_TIM10_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM10 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
28	DBG_TIM9_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM9 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数；

				1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
27	DBG_TIM14_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM14 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
26	DBG_TIM13_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM13 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
25	DBG_TIM12_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM12 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
24:21	保留	-	-	-
20	DBG_TIM7_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM7 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
19	DBG_TIM6_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM6 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
18	DBG_TIM5_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM5 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
17	DBG_TIM8_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM8 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
16	DBG_I2C2_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C2 SMBUS 超时模式停止。 0：CPU halt 时，I2C2 与正常模式相同； 1：CPU halt 时，I2C2 SMBUS 超时功能禁止；
15	DBG_I2C1_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C1 SMBUS 超时模式停止。 0：CPU halt 时，I2C1 与正常模式相同； 1：CPU halt 时，I2C21 SMBUS 超时功能禁止；
14	DBG_CAN1_STOP	RW	0	当 CPU 核处于 halt 状态时，启动 CAN1 模块的“读取时禁止修改功能”。 0：CPU halt 时，CAN1 与正常模式相同； 1：CPU halt 时，会启用 CAN1 的“读时禁止修改功能”。 读时禁止修改功能描述如下：ECR.CEL[7:0]，PSR.PXE, PSR.RFDF, PSR.RBRS, PSR.RESI 不会

				被复位, PSR.LEC 和 PSR.DLEC 信号不会被置位。并且这些寄存器被重复读取后，也不会被复位或者置位。
13	DBG_TIM4_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM4 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
12	DBG_TIM3_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM3 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
11	DBG_TIM2_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM2 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
10	DBG_TIM1_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM1 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
9	DBG_WWDG_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 WWDG 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
8	DBG_IWDG_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 IWDG 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
7:3	保留	-	-	-
2	DBG_STDBY	RW	0	调试待机模式。 0：（FCLK 关，HCLK 关）正常待机模式处理； 1：（FCLK 开，HCLK 开）系统时钟由 HSI16 提供。MCU 自动产生系统复位退出待机模式，与上电复位一样。
1	DBG_STOP	RW	0	调试停止模式。 0：（FCLK 关，HCLK 关）在停止模式，HCLK 和 FCLK 都会关闭。当从停止模式退出时，时钟配置与复位后相同（系统时钟为 HSI16）。随后，软件需要重新配置时钟控制器以使能 PLL 和 HSE 等。 1：（FCLK 开，HCLK 开）。当进入停止模式，系统内部时钟源不会关闭，FCLK 和 HCLK 存在。当退出停止模式，如果需要改变时钟控制，软件需要重新配置。
0	DBG_SLEEP	RW	0	调试睡眠模式。 0：（FCLK 开，HCLK 关）。在睡眠模式，FCLK 由原先配置好的系统时钟提供，HCLK 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后，软件不需要重新配置时钟。 1：（FCLK 开，HCLK 开）。在睡眠模式，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

45.4.3 调试配置寄存器（DBGMCU_CR2）

该寄存器由上电复位，系统复位不会复位该寄存器。当内核处于复位状态下时，调试器可以访问该寄存器。

Address offset：0x08

Reset value：0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							DBG_LPTIM_STOP	DBG_TIM19_STOP	DBG_TIM18_STOP	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	DBG_CAN2_STOP	DBG_I2C4_SMBUS_TIMEOUT	DBG_I2C3_SMBUS_TIMEOUT
-							RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	保留	-	-	-
8	DBG_LPTIM_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 LPTIM 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
7	DBG_TIM19_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM19 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数；
6	DBG_TIM18_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM18 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
5	DBG_TIM17_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM17 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
4	DBG_TIM16_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM16 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
3	DBG_TIM15_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM15 计数停止。 0：CPU halt 时，计数器时钟使能，正常计数； 1：CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
2	DBG_CAN2_STOP	RW	0	当 CPU 核处于 halt 状态时，启动 CAN2 模块的“读取时禁止修改功能”。 0：CPU halt 时，CAN2 与正常模式相同； 1：CPU halt 时，会启用 CAN2 的“读时禁止修改功能”。 读时禁止修改功能描述如下：ECR.CEL[7:0], PSR.PXE, PSR.RFDF, PSR.RBRS, PSR.RESI 不会被复位, PSR.LEC 和 PSR.DLEC 信号不会被置位。并

				且这些寄存器被重复读取后，也不会被复位或者置位。
1	DBG_I2C4_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C4 SMBUS 超时模式停止。 0：CPU halt 时，I2C4 与正常模式相同； 1：CPU halt 时，I2C4 SMBUS 超时功能禁止；
0	DBG_I2C3_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C3 SMBUS 超时模式停止。 0：CPU halt 时，I2C3 与正常模式相同； 1：CPU halt 时，I2C3 SMBUS 超时功能禁止；

46. 更新历史

版本	日期	更新记录
V0.5	2025.04.30	初始版本
V0.6	2025.05.06	更新系统架构框图

Puya Semiconductor Co., Ltd.

声　　明

普冉半导体(上海)股份有限公司（以下简称：“Puya”）保留更改、纠正、增强、修改 Puya 产品和/或本文档的权利，恕不另行通知。用户可在下单前获取产品的最新相关信息。

Puya 产品是依据订单时的销售条款和条件进行销售的。

用户对 Puya 产品的选择和使用承担全责，同时若用于其自己或指定第三方产品上的，Puya 不提供服务支持且不对此类产品承担任何责任。

Puya 在此不授予任何知识产权的明示或暗示方式许可。

Puya 产品的转售，若其条款与此处规定不一致，Puya 对此类产品的任何保修承诺无效。

任何带有 Puya 或 Puya 标识的图形或字样是普冉的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代并替换先前版本中的信息。

普冉半导体(上海)股份有限公司 - 保留所有权利