# RX32G410 参考手册

文档编号：RM00006

基于带 FPU 的 32 位 RISC 内核的专业电机微控制器

版本：V2.1

# 目录

1 简介 ..... 41
2 文档约定 ..... 42
2.1 寄存器缩写列表 ..... 42
2.2 词汇表 ..... 42
3 系统和存储器概述 ..... 43
3.1 系统架构 ..... 43
3.2 存储器结构 ..... 43
3.2.1 介绍 ..... 43
3.2.2 存储器映像和寄存器边界地址 ..... 44
3.3 内置 SRAM ..... 47
3.3.1 奇偶校验 ..... 47
3.3.2 CCM SRAM 读保护 ..... 47
3.3.3 CCM SRAM 擦除 ..... 47
3.4 Flash 概述 ..... 47
3.5 Boot 配置 ..... 48
4 内置 Flash 存储器（FLASH） ..... 49
4.1 介绍 ..... 49
4.2 Flash 主要特征 ..... 49
4.3 Flash 功能描述 ..... 49
4.3.1 Flash 结构 ..... 49
4.3.2 双存储区支持同时操作 ..... 50
4.3.3 误码矫正功能（ECC） ..... 50
4.3.4 读取等待周期 ..... 50
4.3.5 指令预取 ..... 51
4.3.6 Flash 擦除和编程 ..... 51
4.3.7 解锁 Flash ..... 51
4.3.8 Flash 擦除流程 ..... 52
4.3.9 FLASH 写入流程 ..... 52
4.4 保护 ..... 52
4.4.1 读保护 ..... 52
4.4.2 写保护 ..... 53
4.5 选项字节 ..... 53
4.5.1 选项字节描述 ..... 53
4.5.2 选项字节擦除 ..... 53
4.5.3 选项字节写入 ..... 53
4.5.4 选项字节组织 ..... 54
4.5.5 用户和读保护选项字节 ..... 54
4.5.6 写保护 A 区地址选项字节 ..... 56
4.5.7 写保护 B 区地址选项字节 ..... 56
4.6 Flash 寄存器描述 ..... 57
4.6.1 Flash 访问控制寄存器（FLASH_ACR） ..... 57

4.6.2 Flash 键寄存器（FLASH_KEYR） 58
4.6.3 Flash OPTKEY 寄存器（FLASH_OPTKEYR） 58
4.6.4 Flash 状态寄存器（FLASH_SR） 58
4.6.5 Flash 控制寄存器（FLASH_CR） 60
4.6.6 Flash ECC 寄存器（FLASH_ECCR） 61
4.6.7 Flash 选项字节寄存器（FLASH_OPTR） 62
4.6.8 Flash 写保护 A 区地址寄存器（FLASH_WRP1AR） 64
4.6.9 Flash 写保护 B 区地址寄存器（FLASH_WRP1BR） 64
5 循环冗余校验计算单元 (CRC) 65
5.1 CRC 简介 65
5.2 CRC 主要特性 65
5.3 CRC 功能说明 65
5.4 CRC 寄存器 66
5.4.1 数据寄存器（CRC_DR） 66
5.4.2 独立数据寄存器（CRC_IDR） 66
5.4.3 控制寄存器（CRC_CR） 66
6 电源控制（PWR） 67
6.1 概述 67
6.2 电源管理器 67
6.2.1 上电复位（POR）和掉电复位（PDR） 67
6.2.1 可编程电压检测器（PVD） 68
6.3 低功耗模式 69
6.3.1 运行模式 70
6.3.2 睡眠模式 70
6.3.3 停止 0 模式 70
6.3.4 停止 1 模式 70
6.3.5 待机模式 70
6.3.6 低功耗模式下的自动唤醒（AWU） 71
6.4 PWR 寄存器描述 72
6.4.1 电源控制寄存器（PWR_CR1） 72
6.4.2 电源控制寄存器（PWR_CR2） 72
6.4.3 电源控制寄存器（PWR_CR3） 73
6.4.4 电源控制寄存器（PWR_CR4） 74
6.4.5 电源状态寄存器 1（PWR_SR1） 75
6.4.6 电源状态寄存器 2（PWR_SR2） 76
6.4.7 电源状态清除寄存器（PWR_SCR） 76
7 备份寄存器（BKP） 77
7.1 BKP 简介 77
7.2 BKP 特性 77
7.3 BKP 功能描述 77
7.3.1 侵入检测 77
7.3.2 RTC 校准 77
7.4 BKP 寄存器描述 78

7.4.1 RTC 时钟校准寄存器（BKP_RTCCR） 78
7.4.2 备份控制寄存器（BKP_CR） 78
7.4.3 备份控制/状态寄存器（BKP_CSR） 79
8 复位和时钟控制（RCC） 80
8.1 复位 80
8.1.1 电源复位 80
8.1.2 系统复位 80
8.1.3 RTC 域复位 81
8.2 时钟 81
8.2.1 HSE 时钟 84
8.2.2 HSI16 时钟 85
8.2.3 PLL 内部锁相环 86
8.2.4 LSI 时钟 86
8.2.5 系统时钟（SYSCLK）选择 86
8.2.6 时钟安全系统（CSS） 87
8.2.7 ADC 时钟 87
8.2.8 RTC 时钟 87
8.2.9 定时器时钟 88
8.2.10 看门狗时钟 88
8.2.11 时钟输出 88
8.2.12 使用 TIM5/TIM15/TIM16/TIM17 进行内部/外部时钟测量 88
8.2.13 外设时钟使能寄存器（RCC_AHBxENR, RCC_APBxENRy） 91
8.3 低功耗模式 91
8.4 RCC 寄存器 92
8.4.1 时钟控制寄存器（RCC_CR） 92
8.4.2 内部时钟源校准寄存器（RCC_ICSCR） 94
8.4.3 时钟配置寄存器（RCC_CFGR） 94
8.4.4 PLL 配置寄存器（RCC_PLLCFGR） 96
8.4.5 定时器时钟配置寄存器（RCC_CFGR1） 98
8.4.6 时钟中断使能寄存器（RCC_CIER） 99
8.4.7 时钟中断标志寄存器（RCC_CIFR） 100
8.4.8 时钟中断清除寄存器（RCC_CICR） 101
8.4.9 AHB1 外设复位寄存器（RCC_AHB1RSTR） 102
8.4.10 AHB2 外设复位寄存器（RCC_AHB2RSTR） 103
8.4.11 APB1 外设复位寄存器 1（RCC_APB1RSTR1） 104
8.4.12 APB2 外设复位寄存器（RCC_APB2RSTR） 106
8.4.13 AHB1 外设时钟使能寄存器（RCC_AHB1ENR） 108
8.4.14 AHB2 外设时钟使能寄存器（RCC_AHB2ENR） 109
8.4.15 APB1 外设时钟使能寄存器 1（RCC_APB1ENR1） 110
8.4.16 APB2 外设时钟使能寄存器（RCC_APB2ENR） 113
8.4.17 独立外设时钟配置寄存器（RCC_CCIPR） 114
8.4.18 RTC 域控制寄存器（RCC_BDCR） 115
8.4.19 控制/状态寄存器（RCC_CSR） 116

9 通用 I/O（GPIO） ..... 118
9.1 简介 ..... 118
9.2 GPIO 主要特征 ..... 118
9.3 GPIO 功能描述 ..... 118
9.3.1 通用 IO（GPIO） ..... 121
9.3.2 I/O 引脚复用功能复用器和映射 ..... 121
9.3.3 I/O 端口控制寄存器 ..... 122
9.3.4 I/O 端口数据寄存器 ..... 122
9.3.5 I/O 数据位操作 ..... 122
9.3.6 I/O 锁定机制 ..... 122
9.3.7 I/O 复用功能输入/输出 ..... 123
9.3.8 外部中断/唤醒线 ..... 123
9.3.9 输入配置 ..... 123
9.3.10 输出模式配置 ..... 124
9.3.11 复用功能配置 ..... 124
9.3.12 模拟配置 ..... 125
9.3.13 将 HSE 振荡器引脚用作 GPIO ..... 126
9.3.14 使用 PB8 作为 GPIO ..... 126
9.4 GPIO 寄存器 ..... 127
9.4.1 GPIO 端口模式寄存器（GPIOx_MODER）（x = A...F） ..... 127
9.4.2 GPIO 端口输出类型寄存器（GPIOx_OTYPER）（x = A...F） ..... 127
9.4.3 GPIO 端口输出速度寄存器（GPIOx_OSPEEDR）（x = A...F） ..... 128
9.4.4 GPIO 端口上拉/下拉寄存器（GPIOx_PUPDR）（x = A...F） ..... 128
9.4.5 GPIO 端口输入数据寄存器（GPIOx_IDR）（x = A...F） ..... 129
9.4.6 GPIO 端口输出数据寄存器（GPIOx_ODR）（x = A...F） ..... 129
9.4.7 GPIO 端口置位/复位寄存器（GPIOx_BSRR）（x = A...F） ..... 129
9.4.8 GPIO 端口配置锁定寄存器（GPIOx_LCKR）（x = A...F） ..... 130
9.4.9 GPIO 复用功能低位寄存器（GPIOx_AFRL）（x = A...F） ..... 131
9.4.10 GPIO 复用功能高位寄存器（GPIOx_AFRH）（x = A...F） ..... 131
9.4.11 GPIO 端口位复位寄存器（GPIOx_BRR）（x = A...F） ..... 132
10 系统配置控制器（SYSCFG） ..... 133
10.1 SYSCFG 主要特征 ..... 133
10.2 SYSCFG 寄存器 ..... 133
10.2.1 SYSCFG 外部中断配置寄存器 1（SYSCFG_EXTICR1） ..... 133
10.2.2 SYSCFG 外部中断配置寄存器 2（SYSCFG_EXTICR2） ..... 133
10.2.3 SYSCFG 外部中断配置寄存器 3（SYSCFG_EXTICR3） ..... 134
10.2.4 SYSCFG 外部中断配置寄存器 4（SYSCFG_EXTICR4） ..... 134
10.2.5 SYSCFG CCM SRAM 控制和状态寄存器（SYSCFG_SCSR） ..... 135
10.2.6 SYSCFG 配置寄存器 2（SYSCFG_CFGR2） ..... 135
10.2.7 SYSCFG CCM SRAM 密钥寄存器 （SYSCFG_SKR） ..... 136
11 外部中断和事件控制器（EXTI） ..... 137
11.1 简介 ..... 137
11.2 EXTI 主要特性 ..... 137

11.3 EXTI 功能介绍 137
11.3.1 EXTI 框图 138
11.3.2 外设异步中断 138
11.3.3 硬件中断选择 139
11.3.4 硬件事件选择 139
11.3.5 软件中断/事件选择 139
11.4 EXTI 中断/事件线映射 139
11.5 EXTI 寄存器描述 142
11.5.1 中断屏蔽寄存器（EXTI_IMR） 142
11.5.2 事件屏蔽寄存器（EXTI_EMR） 142
11.5.3 上升沿触发选择寄存器（EXTI_RTSR） 143
11.5.4 下降沿触发选择寄存器（EXTI_FTSR） 144
11.5.5 软件中断事件寄存器（EXTI_SWIER） 144
11.5.6 挂起寄存器（EXTI_PR） 145
12 可配置逻辑单元（CLC） 146
12.1 CLC 简介 146
12.2 CLC 引脚和内部信号 149
12.3 CLC 设置 149
12.4 输入提供 150
12.4.1 源多路开关 150
12.4.2 逻辑输入门 150
12.4.3 逻辑功能 151
12.4.4 软件输入 151
12.5 输出 151
12.6 应用逻辑 151
12.6.1 组合逻辑 151
12.6.2 状态逻辑 152
12.7 CLC 中断 152
12.8 睡眠模式操作 152
12.9 复位 152
12.10 CLC 寄存器 153
12.10.1 可配置逻辑单元 x 控制寄存器（CLCxCON）（x = 1~4） 153
12.10.2 可配置逻辑单元 x 输入多路开关选择寄存器（CLCxSEL）（x = 1~4） 155
12.10.3 可配置逻辑单元 x 源使能寄存器（CLCxGLS）（x = 1~4） 155
13 CORDIC 协同处理器（CORDIC） 157
13.1 CORDIC 简介 157
13.2 CORDIC 主要特征 157
13.3 CORDIC 功能描述 157
13.3.1 通用描述 157
13.3.2 CORDIC 函数 157
13.3.3 定点表示法 163
13.3.4 缩放系数 163
13.3.5 精度 164

13.3.6 零开销模式 167
13.3.7 轮询方式 168
13.3.8 中断模式 168
13.3.9 DMA 模式 168
13.4 CORDIC 寄存器描述 170
13.4.1 CORDIC 控制/状态寄存器（CORDIC_CSR） 170
13.4.2 CORDIC 参数寄存器（CORDIC_WDATA） 171
13.4.3 CORDIC 结果寄存器（CORDIC_RDATA） 172
14 FMAC 滤波加速器（FMAC） 173
14.1 FMAC 简介 173
14.2 FMAC 主要特征 173
14.3 FMAC 功能描述 174
14.3.1 通用描述 174
14.3.2 本地存储与缓存 175
14.3.3 输入缓存 175
14.3.4 输出缓存 178
14.3.5 初始化函数 180
14.3.6 滤波函数 181
14.3.7 固定点表示法 185
14.3.8 使用 FMAC 实现 FIR 滤波器 185
14.3.9 使用 FMAC 实现 IIR 滤波器 187
14.3.10 滤波器初始化示例 189
14.3.11 滤波器操作示例 190
14.3.12 滤波器设计提示 192
14.4 FMAC 寄存器描述 193
14.4.1 FMAC X1 缓冲区配置寄存器（FMAC_X1BUFCFG） 193
14.4.2 FMAC X2 缓冲区配置寄存器（FMAC_X2BUFCFG） 193
14.4.3 FMAC_Y buffer 配置寄存器（FMAC_YBUFCFG） 194
14.4.4 FMAC 参数寄存器（FMAC_PARAM） 195
14.4.5 FMAC 控制寄存器（FMAC_CR） 196
14.4.6 FMAC 状态寄存器（FMAC_SR） 197
14.4.7 FMAC 写数据寄存器（FMAC_WDATA） 198
14.4.8 FMAC 读数据寄存器（FMAC_RDATA） 199
15 模数转换器（ADC） 200
15.1 ADC 简介 200
15.2 ADC 主要特征 200
15.3 ADC 功能描述 201
15.3.1 ADC 框图 201
15.3.2 ADC 引脚和内部信号 202
15.3.3 ADC 时钟 203
15.3.4 ADC 通道连接 204
15.3.5 校准 205
15.3.6 ADC 开关控制 206

15.3.7 通道选择 206
15.3.8 单次转换模式 206
15.3.9 连续转换模式 206
15.3.10 时序图 207
15.3.11 模拟看门狗 207
15.3.12 扫描模式 208
15.3.13 注入通道管理 208
15.3.14 间断模式 209
15.3.15 数据对齐 210
15.3.16 可编程的通道采样时间 211
15.3.17 外部触发转换 211
15.3.18 DMA 请求 214
15.4 双 ADC 模式 215
15.4.1 同步注入模式 217
15.4.2 同步规则模式 217
15.4.3 快速交叉模式 218
15.4.4 慢速交叉模式 218
15.4.5 交替触发模式 219
15.4.6 独立模式 220
15.4.7 混合的规则/注入同步模式 220
15.4.8 混合的同步规则+交替触发模式 220
15.4.9 混合同步注入+交叉模式 221
15.5 温度传感器（TPS） 221
15.6 ADC 中断 222
15.7 ADC 寄存器描述 223
15.7.1 ADC 状态寄存器（ADC_SR） 223
15.7.2 ADC 控制寄存器 1（ADC_CR1） 224
15.7.3 ADC 控制寄存器 2（ADC_CR2） 226
15.7.4 ADC 采样时间寄存器 1（ADC_SMPR1） 229
15.7.5 ADC 采样时间寄存器 2（ADC_SMPR2） 229
15.7.6 ADC 注入通道数据偏移寄存器 x（ADC_JOFRx）（x = 1~4） 230
15.7.7 ADC 看门狗高阈值寄存器（ADC_HTR） 230
15.7.8 ADC 看门狗低阈值寄存器（ADC_LTR） 230
15.7.9 ADC 规则序列寄存器 1（ADC_SQR1） 231
15.7.10 ADC 规则序列寄存器 2（ADC_SQR2） 231
15.7.11 ADC 规则序列寄存器 3（ADC_SQR3） 231
15.7.12 ADC 注入序列寄存器（ADC_JSQR） 232
15.7.13 ADC 注入数据寄存器 x（ADC_JDRx）(x = 1~4) 232
15.7.14 ADC 规则数据寄存器（ADC_DR） 233
15.7.15 ADC 控制寄存器 3（ADC_CR3） 233
15.7.16 ADC 校准寄存器（ADC_CAL） 233
15.7.17 ADC 控制寄存器 4（ADC_CR4） 234
15.7.18 ADC TPS 校准寄存器（ADC_TPSOFFSET） 234

15.7.19 ADC 规则数据寄存器 x（ADC_DRx）(x = 1~8) ..... 234
16 数模转换器（DAC） ..... 235
16.1 DAC 简介 ..... 235
16.2 DAC 主要特征 ..... 235
16.3 DAC 实现 ..... 236
16.4 DAC 功能描述 ..... 237
16.4.1 DAC 框图 ..... 237
16.4.2 DAC 引脚和内部信号 ..... 238
16.4.3 DAC 通道使能 ..... 242
16.4.4 DAC 数据格式 ..... 242
16.4.5 DAC 转换 ..... 243
16.4.6 DAC 输出电压 ..... 244
16.4.7 DAC 触发器选择 ..... 244
16.4.8 DMA 请求 ..... 245
16.4.9 DAC 生成噪音 ..... 245
16.4.10 DAC 生成三角波 ..... 246
16.4.11 DAC 生成锯齿波 ..... 247
16.4.12 DAC 通道缓冲器校准 ..... 249
16.4.13 DAC 双通道转换模式（如果双通道可用） ..... 250
16.5 DAC 低功耗模式 ..... 254
16.6 DAC 中断 ..... 254
16.7 DAC 寄存器描述 ..... 255
16.7.1 DAC 控制寄存器（DAC_CR） ..... 255
16.7.2 DAC 软件触发寄存器（DAC_SWTRIGR） ..... 258
16.7.3 DAC 通道 1 12 位右对齐数据保持寄存器（DAC_DHR12R1） ..... 259
16.7.4 DAC 通道 1 12 位左对齐数据保持寄存器（DAC_DHR12L1） ..... 259
16.7.5 DAC 通道 1 8 位右对齐数据保持寄存器（DAC_DHR8R1） ..... 260
16.7.6 DAC 通道 2 12 位右对齐数据保持寄存器（DAC_DHR12R2） ..... 260
16.7.7 DAC 通道 2 12 位左对齐数据保持寄存器（DAC_DHR12L2） ..... 260
16.7.8 DAC 通道 2 8 位右对齐数据保持寄存器（DAC_DHR8R2） ..... 261
16.7.9 双 DAC 12 位右对齐数据保持寄存器（DAC_DHR12RD） ..... 261
16.7.10 双 DAC 12 位左对齐数据保持寄存器（DAC_DHR12LD） ..... 262
16.7.11 双 DAC 8 位右对齐数据保持寄存器（DAC_DHR8RD） ..... 262
16.7.12 DAC 通道 1 数据输出寄存器（DAC_DOR1） ..... 262
16.7.13 DAC 通道 2 数据输出寄存器（DAC_DOR2） ..... 263
16.7.14 DAC 状态寄存器（DAC_SR） ..... 263
16.7.15 DAC 校准寄存器（DAC_CCR） ..... 264
16.7.16 DAC 模式控制寄存器（DAC_MCR） ..... 264
16.7.17 DAC 通道 1 锯齿波寄存器（DAC_STR1） ..... 265
16.7.18 DAC 通道 2 锯齿寄存器（DAC_STR2） ..... 265
16.7.19 DAC 锯齿波模式寄存器（DAC_STMODR） ..... 266
17 直接存储器访问控制器（DMA） ..... 267
17.1 简介 ..... 267

17.2 DMA 主要特征 267
17.3 DMA 实现 267
17.3.1 DMA1 和 DMA2 267
17.3.2 DMA 请求映射 267
17.4 DMA 功能描述 268
17.4.1 DMA 框图 268
17.4.2 DMA 引脚和内部信号 269
17.4.3 DMA 传输 269
17.4.4 DMA 仲裁器 270
17.4.5 DMA 通道 270
17.4.6 DMA 数据宽度、对齐方式和数据大小端 273
17.4.7 DMA 错误管理 274
17.4.8 DMA 中断 274
17.5 DMA 寄存器描述 275
17.5.1 DMA 中断状态寄存器（DMA_ISR） 275
17.5.2 DMA 中断寄存器标志清除寄存器（DMA_IFCR） 276
17.5.3 DMA 通道 x 配置寄存器（DMA_CCRx） 277
17.5.4 DMA 通道 x 传输数量寄存器（DMA_CNDTRx） 280
17.5.5 DMA 通道 x 外设地址寄存器（DMA_CPARx） 281
17.5.6 DMA 通道 x 外设存储地址寄存器（DMA_CMARx） 281
18 DMA 请求复用器（DMAMUX） 282
18.1 简介 282
18.2 主要特性 282
18.3 DMAMUX 实现 283
18.3.1 DMAMUX 实例化 283
18.3.2 DMAMUX 映射 283
18.4 DMAMUX 功能描述 286
18.4.1 DMAMUX 框图 286
18.4.2 DMAMUX 信号 287
18.4.3 DMAMUX 通道 287
18.4.4 DMAMUX 请求线复用器 287
18.4.5 DMAMUX 请求发生器 290
18.5 DMAMUX 中断 291
18.6 DMAMUX 寄存器描述 292
18.6.1 DMAMUX 请求线复用器通道 x 配置寄存器（DMAMUX_CxCR） 292
18.6.2 DMAMUX 请求线复用器中断通道状态寄存器（DMAMUX_CSR） 293
18.6.3 DMAMUX 请求线复用器中断清除标志寄存器（DMAMUX_CFR） 293
18.6.4 DMAMUX 请求发生器通道 x 配置寄存器（DMAMUX_RGxCR）（x=0...3） 293
18.6.5 DMAMUX 请求发生器中断状态寄存器（DMAMUX_RGSR） 294
18.6.6 DMAMUX 请求发生器中断清除标志寄存器（DMAMUX_RGCFR） 294
19 比较器（COMP） 295
19.1 简介 295
19.2 主要特性 295

19.3 功能描述 ..... 295
19.3.1 COMP 框图 ..... 295
19.3.2 COMP 引脚和内部信号 ..... 296
19.3.3 COMP 复位和时钟 ..... 296
19.3.4 COMP 锁定机制 ..... 296
19.3.5 COMP 防误触发功能 ..... 297
19.3.6 COMP 迟滞 ..... 298
19.3.7 COMP 输出遮蔽功能 ..... 298
19.3.8 COMP 低功耗模式 ..... 299
19.3.9 COMP 中断 ..... 299
19.4 COMP 寄存器描述 ..... 300
19.4.1 比较器 x 控制和状态寄存器（COMP_CxCR）（x=1/2/3/4） ..... 300
19.4.2 比较器 x 校正寄存器（COMP_RGxCAL）（x=1/2/3/4） ..... 303
20 运算放大器（OPAMP） ..... 304
20.1 简介 ..... 304
20.2 OPAMP 主要特性 ..... 304
20.3 OPAMP 功能描述 ..... 304
20.3.1 OPAMP 输出重定向至内部 ADC 通道 ..... 304
20.3.2 OPAMP 复位和时钟 ..... 304
20.3.3 初始化配置 ..... 304
20.3.4 信号路由 ..... 305
20.3.5 OPAMP 模式 ..... 306
20.3.6 校准 ..... 312
20.3.7 OPAMP 低功耗模式 ..... 313
20.4 OPAMP 寄存器描述 ..... 314
20.4.1 OPAMPx 控制状态寄存器（OPAMPx_CSR）（x=1...4） ..... 314
20.4.2 OPAMP1 控制寄存器 2（OPAMP1_CR2） ..... 316
20.4.3 OPAMP2 控制寄存器 2（OPAMP2_CR2） ..... 317
20.4.4 OPAMP3 控制寄存器 2（OPAMP3_CR2） ..... 318
20.4.5 OPAMP4 控制寄存器 2（OPAMP4_CR2） ..... 319
21 高分辨率定时器（HRTIM） ..... 320
21.1 HRTIM 简介 ..... 320
21.2 HRTIM 主要特征 ..... 320
21.3 HRTIM 功能描述 ..... 321
21.3.1 综合描述 ..... 321
21.3.2 HRTIM 引脚和内部信号 ..... 323
21.3.3 时钟 ..... 327
21.3.4 定时器 A..F 定时单元 ..... 330
21.3.5 主定时器 ..... 351
21.3.6 上下计数模式 ..... 353
21.3.7 置位/复位事件优先级和窄脉冲管理 ..... 360
21.3.8 外部事件全局条件 ..... 363
21.3.9 定时单元中的外部事件过滤 ..... 368

21.3.10 延迟保护 374
21.3.11 寄存器预加载和更新管理 380
21.3.12 带有“大于”比较的 PWM 模式 384
21.3.13 事件在多个定时器之间的传播 385
21.3.14 输出管理 390
21.3.15 突发模式控制器 392
21.3.16 斩波 400
21.3.17 故障保护 401
21.3.18 辅助输出 406
21.3.19 将 HRTIM 与其他定时器或 HRTIM 实例同步 408
21.3.20 ADC 触发 411
21.3.21 DAC 触发 415
21.3.22 中断 419
21.3.23 DMA 421
21.3.24 HRTIM 初始化 425
21.3.25 Debug 调试 426
21.4 应用案例 427
21.4.1 降压转换器 427
21.4.2 具有同步整流功能的降压转换器 428
21.4.3 多相转换器 429
21.4.4 过渡模式功率因数校正 430
21.5 HRTIM 寄存器 432
21.5.1 HRTIM 主定时器控制寄存器 (HRTIM_MCR) 432
21.5.2 HRTIM 主定时器中断状态寄存器（HRTIM_MISR） 435
21.5.3 HRTIM 主定时器中断清除寄存器（HRTIM_MICR） 436
21.5.4 HRTIM 主定时器 DMA/中断使能寄存器（HRTIM_MDIER） 437
21.5.5 HRTIM 主定时器计数器寄存器（HRTIM_MCNTR） 439
21.5.6 HRTIM 主定时器周期寄存器（HRTIM_MPER） 439
21.5.7 HRTIM 主定时器重复寄存器（HRTIM_MREP） 440
21.5.8 HRTIM 主定时器比较 1 寄存器（HRTIM_MCMP1R） 440
21.5.9 HRTIM 主定时器比较 2 寄存器（HRTIM_MCMP2R） 441
21.5.10 HRTIM 主定时器比较 3 寄存器（HRTIM_MCMP3R） 441
21.5.11 HRTIM 主定时器比较 4 寄存器（HRTIM_MCMP4R） 442
21.5.12 HRTIM 定时器 x 控制寄存器 (HRTIM_TIMxCR) (x = A~F) 443
21.5.13 HRTIM 定时器 x 中断状态寄存器 (HRTIM_TIMxISR) (x = A~F) 448
21.5.14 HRTIM 定时器 x 中断清除寄存器（HRTIM_TIMxICR）(x = A~F) 450
21.5.15 HRTIM 定时器 x DMA 中断使能寄存器 (HRTIM_TIMxDIER) (x = A~F) 451
21.5.16 HRTIM 定时器 x 计数器寄存器（HRTIM_CNTxR）(x = A~F) 455
21.5.17 HRTIM 定时器 x 周期寄存器（HRTIM_PERxR）(x = A~F) 455
21.5.18 HRTIM 定时器 x 重复寄存器 (HRTIM_REPxR) (x = A~F) 456
21.5.19 HRTIM 定时器 x 比较 1 寄存器 (HRTIM_CMP1xR) (x = A~F) 456
21.5.20 HRTIM 定时器 x 比较 1 复合寄存器 (HRTIM_CMP1CxR) (x = A~F) 457
21.5.21 HRTIM 定时器 x 比较 2 寄存器 (HRTIM_CMP2xR) (x = A~F) 457

21.5.22 HRTIM 定时器 x 比较 3 寄存器（HRTIM_CMP3xR) (x = A~F)....458
21.5.23 HRTIM 定时器 x 比较 4 寄存器（HRTIM_CMP4xR) (x = A~F)....458
21.5.24 HRTIM 定时器 x 捕获 1 寄存器（HRTIM_CPT1xR）（x = A~F）....459
21.5.25 HRTIM 定时器 x 捕获 2 寄存器（HRTIM_CPT2xR）（x = A~F）....460
21.5.26 HRTIM 定时器 x 死区时间寄存器（HRTIM_DTxR）（x = A~F）....461
21.5.27 HRTIM 定时器 x 输出 1 置位寄存器（HRTIM_SETx1R）（x = A~F）....463
21.5.28 HRTIM 定时器 x 输出 1 复位寄存器（HRTIM_RSTx1R）（x = A~F）....466
21.5.29 HRTIM 定时器 x 输出 2 置位寄存器（HRTIM_SETx2R）（x = A~F）....466
21.5.30 HRTIM 定时器 x 输出 2 复位寄存器（HRTIM_RSTx2R）（x = A~F）....467
21.5.31 HRTIM 定时器 x 外部事件过滤寄存器 1 (HRTIM_EEFxR1) (x = A~F)....468
21.5.32 HRTIM 定时器 x 外部事件过滤寄存器 2 (HRTIM_EEFxR2) (x = A~F)....470
21.5.33 HRTIM 定时器 A 复位寄存器（HRTIM_RSTAR)....471
21.5.34 HRTIM 定时器 B 复位寄存器（HRTIM_RSTBR)....473
21.5.35 HRTIM 定时器 C 复位寄存器（HRTIM_RSTCR)....475
21.5.36 HRTIM 定时器 D 复位寄存器（HRTIM_RSTDR)....477
21.5.37 HRTIM 定时器 E 复位寄存器（HRTIM_RSTER)....479
21.5.38 HRTIM 定时器 F 复位寄存器（HRTIM_RSTFR)....481
21.5.39 HRTIM 定时器 x 斩波寄存器（HRTIM_CHPxR）（x = A~F）....483
21.5.40 HRTIM 定时器 A 捕获 1 控制寄存器（HRTIM_CPT1ACR）....484
21.5.41 HRTIM 定时器 B 捕获 1 控制寄存器（HRTIM_CPT1BCR）....484
21.5.42 HRTIM 定时器 C 捕获 1 控制寄存器（HRTIM_CPT1CCR）....484
21.5.43 HRTIM 定时器 D 捕获 1 控制寄存器（HRTIM_CPT1DCR）....485
21.5.44 HRTIM 定时器 E 捕获 1 控制寄存器（HRTIM_CPT1ECR）....485
21.5.45 HRTIM 定时器 F 捕获 1 控制寄存器（HRTIM_CPT1FCR）....485
21.5.46 HRTIM 定时器 x 捕获 2 控制寄存器（HRTIM_CPT2xCR）（x = A~F）....486
21.5.47 HRTIM 定时器 x 输出寄存器（HRTIM_OUTxR) (x = A~F)....492
21.5.48 HRTIM 定时器 x 故障寄存器（HRTIM_FLTxR）（x = A~F）....495
21.5.49 HRTIM 定时器 x 控制寄存器 2（HRTIM_TIMxCR2）（x = A~F）....496
21.5.50 HRTIM 定时器 x 外部事件滤波寄存器 3 (HRTIM_TIMxEEFR3)（x = A~F）....499
21.5.51 HRTIM 控制寄存器 1 (HRTIM_CR1)....500
21.5.52 HRTIM 控制寄存器 2 (HRTIM_CR2)....502
21.5.53 HRTIM 中断状态寄存器 （HRTIM_ISR)....504
21.5.54 HRTIM 中断清除寄存器（HRTIM_ICR）....505
21.5.55 HRTIM 中断使能寄存器（HRTIM_IER）....506
21.5.56 HRTIM 输出使能寄存器（HRTIM_OENR）....507
21.5.57 HRTIM 输出禁用寄存器（HRTIM_ODISR）....508
21.5.58 HRTIM 输出禁用状态寄存器（HRTIM_ODSR）....509
21.5.59 HRTIM 突发模式控制寄存器（HRTIM_BMCR）....510
21.5.60 HRTIM 突发模式触发寄存器（HRTIM_BMTRGR)....512
21.5.61 HRTIM 突发模式比较寄存器（HRTIM_BMCMPR)....514
21.5.62 HRTIM 突发模式周期寄存器（HRTIM_BMPER)....514
21.5.63 HRTIM 定时器外部事件控制寄存器 1 (HRTIM_EECR1)....515
21.5.64 HRTIM 定时器外部事件控制寄存器 2 (HRTIM_EECR2)....517

21.5.65 HRTIM 定时器外部事件控制寄存器 3 (HRTIM_EECR3) ..... 518
21.5.66 HRTIM ADC 触发 1 寄存器（HRTIM_ADC1R） ..... 519
21.5.67 HRTIM ADC 触发 2 寄存器（HRTIM_ADC2R） ..... 520
21.5.68 HRTIM ADC 触发 3 寄存器（HRTIM_ADC3R） ..... 521
21.5.69 HRTIM ADC 触发 4 寄存器（HRTIM_ADC4R) ..... 523
21.5.70 HRTIM DLL 控制寄存器（HRTIM_DLLCR） ..... 525
21.5.71 HRTIM 故障输入寄存器 1 (HRTIM_FLTINR1) ..... 526
21.5.72 HRTIM 故障输入寄存器 2 (HRTIM_FLTINR2) ..... 528
21.5.73 HRTIM 突发 DMA 主定时器更新寄存器（HRTIM_BDMUPR） ..... 530
21.5.74 HRTIM 突发 DMA 定时器 x 更新寄存器（HRTIM_BDTxUPR）（x = A~F） ..... 531
21.5.75 HRTIM 突发 DMA 数据寄存器（HRTIM_BDMADR） ..... 532
21.5.76 HRTIM ADC 扩展触发寄存器（HRTIM_ADCER） ..... 533
21.5.77 HRTIM ADC 触发更新寄存器（HRTIM_ADCUR） ..... 536
21.5.78 HRTIM ADC 后缩放器寄存器 1 (HRTIM_ADCPS1) ..... 537
21.5.79 HRTIM ADC 后缩放器寄存器 2 (HRTIM_ADCPS2) ..... 538
21.5.80 HRTIM 故障输入寄存器 3 (HRTIM_FLTINR3) ..... 539
21.5.81 HRTIM 故障输入寄存器 4（HRTIM_FLTINR4） ..... 541
21.5.82 HRTIM 选项寄存器（HRTIM_OPT） ..... 542
22 高级控制定时器（TIM1/TIM8） ..... 543
22.1 TIM1/TIM8 简介 ..... 543
22.2 TIM1/TIM8 主要特征 ..... 543
22.3 TIM1/TIM8 功能描述 ..... 544
22.3.1 框图 ..... 544
22.3.2 TIM1/TIM8 引脚和内部信号 ..... 545
22.3.3 时基单元 ..... 550
22.3.4 计数模式 ..... 552
22.3.5 重复计数器 ..... 562
22.3.6 外部触发输入 ..... 563
22.3.7 时钟选择 ..... 564
22.3.8 捕获比较通道 ..... 567
22.3.9 输入捕获模式 ..... 570
22.3.10 PWM 输入模式 ..... 570
22.3.11 强制输出模式 ..... 571
22.3.12 输出比较模式 ..... 571
22.3.13 PWM 模式 ..... 573
22.3.14 非对称 PWM 模式 ..... 581
22.3.15 组合 PWM 模式 ..... 581
22.3.16 三相组合 PWM 模式 ..... 582
22.3.17 互补输出和死区插入 ..... 583
22.3.18 使用刹车功能 ..... 586
22.3.19 双向刹车输入 ..... 590
22.3.20 在外部事件时清除 tim_ocxref 信号 ..... 592
22.3.21 生成 6 步 PWM ..... 593

22.3.22 单脉冲模式 595
22.3.23 可重触发单脉冲模式 596
22.3.24 在比较模式下的脉冲 597
22.3.25 编码器接口模式 598
22.3.26 方向位输出 616
22.3.27 UIF 位重映射 617
22.3.28 定时器输入异或功能 617
22.3.29 与霍尔传感器接口 617
22.3.30 定时器同步 620
22.3.31 ADC 触发器 624
22.3.32 DMA 突发模式 624
22.3.33 TIM1/TIM8 DMA 请求 625
22.3.34 调试模式 625
22.4 TIM1/TIM8 低功耗模式 625
22.5 TIM1/TIM8 中断 625
22.6 TIM1/TIM8 寄存器描述 627
22.6.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 1，8） 627
22.6.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 1，8） 628
22.6.3 TIMx 从模式控制寄存器（TIMx_SMCR）（x = 1，8） 632
22.6.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 1，8） 636
22.6.5 TIMx 状态寄存器（TIMx_SR）（x = 1，8） 637
22.6.6 TIMx 事件产生寄存器（TIMx_EGR）（x = 1，8） 640
22.6.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 1，8） 641
22.6.8 TIMx 捕获/比较模式寄存器 1[复用]（TIMx_CCMR1）（x = 1，8） 643
22.6.9 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 1，8） 646
22.6.10 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 1，8） 647
22.6.11 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 1，8） 650
22.6.12 TIMx 计数器（TIMx_CNT）（x = 1，8） 654
22.6.13 TIMx 预分频器（TIMx_PSC）（x = 1，8） 654
22.6.14 TIMx 自动重装载寄存器（TIMx_ARR）（x = 1，8） 655
22.6.15 TIMx 重复计数寄存器（TIMx_RCR）（x = 1，8） 655
22.6.16 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 1，8） 656
22.6.17 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 1，8） 656
22.6.18 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 1，8） 657
22.6.19 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 1，8） 658
22.6.20 TIMx 刹车和死区寄存器（TIMx_BDTR）（x = 1，8） 659
22.6.21 TIMx 捕获/比较寄存器 5（TIMx_CCR5）（x = 1，8） 664
22.6.22 TIMx 捕获/比较寄存器 6（TIMx_CCR6）（x = 1，8） 665
22.6.23 TIMx 捕获/比较模式寄存器 3（TIMx_CCMR3）（x = 1，8） 665
22.6.24 TIMx 定时器死区寄存器 2（TIMx_DTR2）（x = 1，8） 666
22.6.25 TIMx 定时器编码器控制寄存器（TIMx_ECR）（x = 1，8） 667
22.6.26 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x =1，8） 668
22.6.27 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 1，8） 669

22.6.28 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 1，8） ..... 671
22.6.29 TIMx 捕获/比较寄存器 1N（TIMx_CCR1N）（x = 1，8） ..... 673
22.6.30 TIMx 捕获/比较寄存器 2N（TIMx_CCR2N）（x = 1，8） ..... 673
22.6.31 TIMx 捕获/比较寄存器 3N（TIMx_CCR3N）（x = 1，8） ..... 674
22.6.32 TIMx 捕获/比较寄存器 4N（TIMx_CCR4N）（x = 1，8） ..... 674
22.6.33 TIMx 刹车输入控制选择（TIMx_BICS）（x = 1，8） ..... 675
22.6.34 TIMx DMA 控制寄存器（TIMx_DCR）（x = 1，8） ..... 676
22.6.35 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 1，8） ..... 677
23 通用定时器（TIM2/TIM3/TIM4/TIM5） ..... 678
23.1 TIM2/TIM3/TIM4/TIM5 简介 ..... 678
23.2 TIM2/TIM3/TIM4/TIM5 主要功能 ..... 678
23.3 TIM2/TIM3/TIM4/TIM5 实现 ..... 678
23.4 TIM2/TIM3/TIM4/TIM5 功能描述 ..... 679
23.4.1 框图 ..... 679
23.4.2 TIM2/TIM3/TIM4/TIM5 引脚和内部信号 ..... 680
23.4.3 时基单元 ..... 683
23.4.4 计数器模式 ..... 685
23.4.5 时钟选择 ..... 695
23.4.6 捕获/比较通道 ..... 698
23.4.7 输入捕获模式 ..... 700
23.4.8 PWM 输入模式 ..... 701
23.4.9 强制输出模式 ..... 702
23.4.10 输出比较模式 ..... 702
23.4.11 PWM 模式 ..... 703
23.4.12 非对称 PWM 模式 ..... 711
23.4.13 组合 PWM 模式 ..... 711
23.4.14 在外部事件时清除 tim_ocxref 信号 ..... 713
23.4.15 单脉冲模式 ..... 714
23.4.16 可重触发单脉冲模式 ..... 716
23.4.17 在比较模式下的脉冲 ..... 716
23.4.18 编码器接口模式 ..... 719
23.4.19 方向位输出 ..... 737
23.4.20 UIF 位重映射 ..... 737
23.4.21 定时器输入异或功能 ..... 737
23.4.22 定时器和外部触发的同步 ..... 738
23.4.23 定时器同步 ..... 742
23.4.24 ADC 触发 ..... 746
23.4.25 DMA 突发模式 ..... 746
23.4.26 TIM2/TIM3/TIM4/TIM5 DMA 请求 ..... 747
23.4.27 调试模式 ..... 748
23.5 TIM2/TIM3/TIM4/TIM5 低功耗模式 ..... 748
23.6 TIM2/TIM3/TIM4/TIM5 中断 ..... 748
23.7 TIM2/TIM3/TIM4/TIM5 寄存器描述 ..... 749

23.7.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 2~5） ..... 749
23.7.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 2~5） ..... 750
23.7.3 TIMx 从模式控制寄存器（TIMx_SMCR）（x = 2~5） ..... 752
23.7.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 2~5） ..... 756
23.7.5 TIMx 状态寄存器（TIMx_SR）（x = 2~5） ..... 757
23.7.6 TIMx 事件产生寄存器（TIMx_EGR）（x = 2~5） ..... 759
23.7.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 2~5） ..... 760
23.7.8 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 2~5） ..... 762
23.7.9 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 2~5） ..... 764
23.7.10 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 2~5） ..... 765
23.7.11 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 2~5） ..... 766
23.7.12 TIMx 计数器（TIMx_CNT）（x = 3，4） ..... 768
23.7.13 TIMx 计数器（TIMx_CNT）（x = 2，5） ..... 768
23.7.14 TIMx 预分频器（TIMx_PSC）（x = 2~5） ..... 769
23.7.15 TIMx 自动重装载寄存器（TIMx_ARR）（x = 3，4） ..... 769
23.7.16 TIMx 自动重装载寄存器（TIMx_ARR）（x = 2，5） ..... 769
23.7.17 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 3，4） ..... 770
23.7.18 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 2，5） ..... 770
23.7.19 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 3，4） ..... 771
23.7.20 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 2，5） ..... 772
23.7.21 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 3，4） ..... 773
23.7.22 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 2，5） ..... 774
23.7.23 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 3，4） ..... 775
23.7.24 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 2，5） ..... 776
23.7.25 TIMx 编码器控制寄存器（TIMx_ECR）（x = 2~5） ..... 777
23.7.26 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x = 2~5） ..... 778
23.7.27 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 2~5） ..... 779
23.7.28 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 2~5） ..... 780
23.7.29 TIMx DMA 控制寄存器（TIMx_DCR）（x = 2~5） ..... 781
23.7.30 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 2~5） ..... 782
24 通用定时器（TIM15/TIM16/TIM17） ..... 783
24.1 TIM15/TIM16/TIM17 简介 ..... 783
24.2 TIM15 主要功能 ..... 783
24.3 TIM16/TIM17 主要功能 ..... 784
24.4 TIM15/TIM16/TIM17 功能描述 ..... 785
24.4.1 框图 ..... 785
24.4.2 TIM15/TIM16/TIM17 引脚和内部信号 ..... 787
24.4.3 时基单元 ..... 790
24.4.4 计数器模式 ..... 792
24.4.5 重复计数器 ..... 795
24.4.6 时钟选择 ..... 796
24.4.7 捕获/比较通道 ..... 798
24.4.8 输入捕获模式 ..... 800

24.4.9 PWM 输入模式（仅 TIM15）.....801
24.4.10 强制输出模式.....802
24.4.11 输出比较模式.....802
24.4.12 PWM 模式.....803
24.4.13 组合 PWM 模式（仅 TIM15）.....808
24.4.14 互补输出和死区插入.....809
24.4.15 使用刹车功能.....812
24.4.16 双向刹车输入.....816
24.4.17 在外部事件时清除 tim_ocxref 信号.....817
24.4.18 单脉冲模式.....818
24.4.19 可重触发单脉冲模式（仅 TIM15）.....819
24.4.20 UIF 位重映射.....820
24.4.21 定时器输入异或功能（仅 TIM15）.....820
24.4.22 定时器和外部触发的同步（仅 TIM15）.....821
24.4.23 定时器同步（仅 TIM15）.....824
24.4.24 使用定时器输出作为其他定时器的触发器（仅 TIM16/TIM17）.....824
24.4.25 ADC 触发（仅 TIM15）.....824
24.4.26 DMA 突发模式.....824
24.4.27 TIM15/TIM16/TIM17 DMA 请求.....825
24.4.28 调试模式.....825
24.5 TIM15/TIM16/TIM17 低功耗模式.....825
24.6 TIM15/TIM16/TIM17 中断.....825
24.7 TIM15 寄存器描述.....827
24.7.1 TIM15 控制寄存器 1（TIM15_CR1）.....827
24.7.2 TIM15 控制寄存器 2（TIM15_CR2）.....828
24.7.3 TIM15 从模式控制寄存器（TIM15_SMCR）.....830
24.7.4 TIM15 DMA/中断使能寄存器（TIM15_DIER）.....831
24.7.5 TIM15 状态寄存器（TIM15_SR）.....832
24.7.6 TIM15 事件产生寄存器（TIM15_EGR）.....834
24.7.7 TIM15 捕获/比较模式寄存器 1 [复用]（TIM15_CCMR1）.....835
24.7.8 TIM15 捕获/比较模式寄存器 1 [复用]（TIM15_CCMR1）.....836
24.7.9 TIM15 捕获/比较使能寄存器（TIM15_CCER）.....839
24.7.10 TIM15 计数器（TIM15_CNT）.....842
24.7.11 TIM15 预分频器（TIM15_PSC）.....842
24.7.12 TIM15 自动重装载寄存器（TIM15_ARR）.....843
24.7.13 TIM15 重复计数寄存器（TIM15_RCR）.....843
24.7.14 TIM15 捕获/比较寄存器 1（TIM15_CCR1）.....844
24.7.15 TIM15 捕获/比较寄存器 2（TIM15_CCR2）.....845
24.7.16 TIM15 刹车和死区寄存器（TIM15_BDTR）.....845
24.7.17 TIM15 定时器死区寄存器 2（TIM15_DTR2）.....849
24.7.18 TIM15 定时器输入选择寄存器（TIM15_TISEL）.....849
24.7.19 TIM15 复用功能选择寄存器 1（TIM15_AF1）.....850
24.7.20 TIM15 复用功能选择寄存器 2（TIM15_AF2）.....852

24.7.21 TIM15 DMA 控制寄存器（TIM15_DCR） 852
24.7.22 TIM15 用于完整传输的 DMA 地址（TIM15_DMAR） 853
24.8 TIM16/TIM17 寄存器描述 854
24.8.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 16~17） 854
24.8.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 16~17） 855
24.8.3 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 16~17） 856
24.8.4 TIMx 状态寄存器（TIMx_SR）（x = 16~17） 857
24.8.5 TIMx 事件产生寄存器（TIMx_EGR）（x = 16~17） 858
24.8.6 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 16~17） 859
24.8.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 16~17） 860
24.8.8 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 16~17） 862
24.8.9 TIMx 计数器（TIMx_CNT）（x = 16~17） 865
24.8.10 TIMx 预分频器（TIMx_PSC）（x = 16~17） 865
24.8.11 TIMx 自动重装载寄存器（TIMx_ARR）（x = 16~17） 866
24.8.12 TIMx 重复计数寄存器（TIMx_RCR）（x = 16~17） 866
24.8.13 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 16~17） 867
24.8.14 TIMx 刹车和死区寄存器（TIMx_BDTR）（x = 16~17） 867
24.8.15 TIMx 定时器死区寄存器 2（TIMx_DTR2）（x = 16~17） 871
24.8.16 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x = 16~17） 872
24.8.17 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 16~17） 872
24.8.18 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 16~17） 874
24.8.19 TIMx 选项寄存器 1（TIMx_OR1）（x = 16~17） 874
24.8.20 TIMx DMA 控制寄存器（TIMx_DCR）（x = 16~17） 875
24.8.21 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 16~17） 875
25 基本定时器（TIM6/TIM7） 876
25.1 TIM6/TIM7 简介 876
25.2 TIM6/TIM7 主要功能 876
25.3 TIM6/TIM7 功能描述 876
25.3.1 框图 876
25.3.2 TIM6/TIM7 内部信号 877
25.3.3 TIM6/TIM7 时钟 877
25.3.4 时基单元 877
25.3.5 计数器模式 879
25.3.6 UIF 位重映射 883
25.3.7 ADC 触发 883
25.3.8 TIM6/TIM7 DMA 请求 884
25.3.9 调试模式 884
25.4 TIM6/TIM7 低功耗模式 884
25.5 TIM6/TIM7 中断 884
25.6 TIMx 寄存器描述 885
25.6.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 6/7） 885
25.6.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 6/7） 887
25.6.3 TIMx/DMA/中断使能寄存器（TIMx_DIER）（x = 6/7） 887

25.6.4 TIMx 状态寄存器（TIMx_SR）（x = 6/7） 888
25.6.5 TIMx 事件产生寄存器（TIMx_EGR）（x = 6/7） 888
25.6.6 TIMx 计数器（TIMx_CNT）（x = 6/7） 888
25.6.7 TIMx 预分频器（TIMx_PSC）（x = 6/7） 889
25.6.8 TIMx 自动重装载寄存器（TIMx_ARR）（x = 6/7） 889
26 参考电压缓冲器（VREFBUF） 890
26.1 VREFBUF 简介 890
26.2 VREFBUF 功能描述 890
26.3 VREFBUF 校准 891
26.4 VREFBUF 内部参考电压档位切换步骤 891
26.5 VREFBUF 切换外部参考电压步骤 891
26.6 VREFBUF 寄存器描述 892
26.6.1 VREFBUF 控制和状态寄存器（VREFBUF_CSR） 892
26.6.2 VREFBUF 校准控制寄存器（VREFBUF_CCR） 893
27 实时时钟（RTC） 894
27.1 RTC 简介 894
27.2 主要特性 894
27.3 功能描述 895
27.3.1 概述 895
27.3.2 复位过程 896
27.3.3 读 RTC 寄存器 896
27.3.4 配置 RTC 寄存器 896
27.3.5 RTC 标志的设置 897
27.4 RTC 寄存器描述 898
27.4.1 RTC 控制寄存器高位（RTC_CRH） 898
27.4.2 RTC 控制寄存器低位（RTC_CRL） 898
27.4.3 RTC 预分频装载寄存器（RTC_PRLH/RTC_PRLL） 900
27.4.4 RTC 预分频器余数寄存器（RTC_DIVH / RTC_DIVL） 901
27.4.5 RTC 计数器寄存器（RTC_CNTH/RTC_CNTL） 901
27.4.6 RTC 闹钟寄存器（RTC_ALRH/RTC_ALRL） 902
28 独立看门狗（IWDG） 903
28.1 简介 903
28.2 IWDG 主要特性 903
28.3 IWDG 功能描述 903
28.3.1 硬件看门狗 903
28.3.2 寄存器访问保护 903
28.3.3 调试模式 903
28.4 IWDG 寄存器描述 905
28.4.1 键寄存器（IWDG_KR） 905
28.4.2 预分频寄存器（IWDG_PR） 905
28.4.3 重装载寄存器（IWDG_RLR） 906
28.4.4 状态寄存器（IWDG_SR） 906
29 窗口看门狗（WWDG） 907

29.1 WWDG 简介 907
29.2 WWDG 主要特性 907
29.3 WWDG 功能描述 907
29.4 如何编写看门狗超时程序 909
29.5 调试模式 909
29.6 WWDG 寄存器描述 910
29.6.1 控制寄存器（WWDG_CR） 910
29.6.2 配置寄存器（WWDG_CFR） 910
29.6.3 状态寄存器（WWDG_SR） 911
30 USB 全速设备接口（USB） 912
30.1 USB 简介 912
30.2 USB 主要特征 912
30.3 USB 框图 913
30.3.1 USB 功能模块描述 914
30.4 编程中需要考虑的问题 915
30.4.1 通用 USB 设备编程 915
30.4.2 系统复位和上电复位 916
30.4.3 双缓冲端点 920
30.4.4 同步传输 922
30.4.5 挂起/恢复事件 922
30.5 USB 寄存器描述 924
30.5.1 通用寄存器 924
30.5.2 端点寄存器 931
30.5.3 缓冲区描述表 935
31 控制器局域网（bxCAN） 939
31.1 bxCAN 简介 939
31.2 bxCAN 主要特点 939
31.3 bxCAN 总体描述 940
31.3.1 CAN 2.0B 主动内核 940
31.3.2 控制、状态和配置寄存器 940
31.3.3 发送邮箱 940
31.3.4 接收过滤器 941
31.4 bxCAN 工作模式 941
31.4.1 初始化模式 942
31.4.2 正常模式 942
31.4.3 睡眠模式（低功耗） 942
31.5 测试模式 943
31.5.1 静默模式 943
31.5.2 环回模式 944
31.5.3 环回静默模式 945
31.6 处于调试模式时 945
31.7 bxCAN 功能描述 945
31.7.1 发送处理 945

31.7.2 时间触发通信模式 947
31.7.3 接收管理 947
31.7.4 标识符过滤 949
31.7.5 报文存储 953
31.7.6 出错管理 955
31.7.7 位时间特性 955
31.8 bxCAN 中断 958
31.9 寄存器访问保护 959
31.10 CAN 控制和状态寄存器描述 960
31.10.1 CAN 主控制寄存器（CAN_MCR） 960
31.10.2 CAN 主状态寄存器（CAN_MSR） 961
31.10.3 CAN 发送状态寄存器（CAN_TSR） 962
31.10.4 CAN 接收 FIFO 0 寄存器（CAN_RF0R） 965
31.10.5 CAN 接收 FIFO 1 寄存器（CAN_RF1R） 966
31.10.6 CAN 中断使能寄存器（CAN_IER） 967
31.10.7 CAN 错误状态寄存器（CAN_ESR） 969
31.10.8 CAN 位时序寄存器 (CAN_BTR) 970
31.11 CAN 邮箱寄存器描述 971
31.11.1 发送邮箱标识符寄存器（CAN_TIxR）（x=0..2） 972
31.11.2 发送邮箱数据长度和时间戳寄存器（CAN_TDTxR）（x=0..2） 973
31.11.3 发送邮箱低字节数据寄存器（CAN_TDLxR）（x=0..2） 974
31.11.4 发送邮箱高字节数据寄存器（CAN_TDHxR）（x=0..2） 974
31.11.5 接收 FIFO 邮箱标识符寄存器（CAN_RIxR）（x=0..1） 975
31.11.6 接收 FIFO 邮箱数据长度和时间戳寄存器（CAN_RDTxR）（x=0..1） 976
31.11.7 接收 FIFO 邮箱低字节数据寄存器（CAN_RDLxR）（x=0..1） 977
31.11.8 接收 FIFO 邮箱高字节数据寄存器（CAN_RDHxR）（x=0..1） 977
31.12 CAN 过滤器寄存器描述 978
31.12.1 CAN 过滤器主控寄存器（CAN_FMR） 978
31.12.2 CAN 过滤器模式寄存器（CAN_FM1R） 978
31.12.3 CAN 过滤器位宽寄存器（CAN_FS1R） 979
31.12.4 CAN 过滤器 FIFO 关联寄存器（CAN_FFA1R） 979
31.12.5 CAN 过滤器激活寄存器（CAN_FA1R） 980
31.12.6 CAN 过滤器组 i 的寄存器 x（CAN_FiRx）（i＝0..13；x=1..2） 980
32 串行外设接口（SPI） 981
32.1 SPI 简介 981
32.2 SPI 主要特性 981
32.3 SPI 功能描述 981
32.3.1 概述 981
32.3.2 配置 SPI 为从模式 986
32.3.3 配置 SPI 为主模式 987
32.3.4 配置 SPI 为单工通信 987
32.3.5 数据发送与接收过程 988
32.3.6 状态标志 993

32.3.7 关闭 SPI ..... 994
32.3.8 利用 DMA 的 SPI 通信 ..... 994
32.3.9 错误标志 ..... 996
32.3.10 SPI 中断 ..... 996
32.4 SPI 寄存器描述 ..... 997
32.4.1 SPI 控制寄存器 1（SPI_CR1） ..... 997
32.4.2 SPI 控制寄存器 2（SPI_CR2） ..... 999
32.4.3 SPI 状态寄存器（SPI_SR） ..... 1000
32.4.4 SPI 数据寄存器（SPI_DR） ..... 1001
33 I²C 接口（I2C） ..... 1002
33.1 I²C 简介 ..... 1002
33.2 I²C 主要特点 ..... 1002
33.3 I²C 功能描述 ..... 1003
33.3.1 模式选择 ..... 1003
33.3.2 I²C 从模式 ..... 1005
33.3.3 I²C 主模式 ..... 1007
33.3.4 错误条件 ..... 1013
33.3.5 SDA/SCL 线控制 ..... 1014
33.3.6 SMBus ..... 1015
33.3.7 DMA 请求 ..... 1017
33.3.8 包错误校验（PEC） ..... 1018
33.4 I²C 中断请求 ..... 1018
33.5 I²C 调试模式 ..... 1020
33.6 I²C 寄存器描述 ..... 1020
33.6.1 控制寄存器 1（I2C_CR1） ..... 1020
33.6.2 控制寄存器 2（I2C_CR2） ..... 1022
33.6.3 自身地址寄存器 1（I2C_OAR1） ..... 1024
33.6.4 自身地址寄存器 2（I2C_OAR2） ..... 1024
33.6.5 数据寄存器（I2C_DR） ..... 1025
33.6.6 状态寄存器 1（I2C_SR1） ..... 1025
33.6.7 状态寄存器 2（I2C_SR2） ..... 1028
33.6.8 时钟控制寄存器（I2C_CCR） ..... 1030
33.6.9 TRISE 寄存器（I2C_TRISE） ..... 1031
34 通用异步收发器（UART） ..... 1032
34.1 UART 介绍 ..... 1032
34.2 UART 主要特性 ..... 1032
34.3 UART 功能概述 ..... 1033
34.3.1 UART 特性描述 ..... 1034
34.3.2 发送器 ..... 1035
34.3.3 接收器 ..... 1038
34.3.4 分数波特率的产生 ..... 1041
34.3.5 UART 接收器容忍时钟的变化 ..... 1042
34.3.6 多处理器通信 ..... 1043

34.3.7 校验控制 ..... 1044
34.3.8 LIN（局域互联网）模式 ..... 1045
34.3.9 单线半双工通信 ..... 1047
34.3.10 利用 DMA 连续通信 ..... 1047
34.3.11 硬件流控制 ..... 1049
34.4 UART 中断请求 ..... 1051
34.5 UART 模式配置 ..... 1052
34.6 UART 寄存器描述 ..... 1053
34.6.1 状态寄存器（UART_SR） ..... 1053
34.6.2 数据寄存器（UART_DR） ..... 1055
34.6.3 波特比率寄存器（UART_BRR） ..... 1055
34.6.4 控制寄存器 1（UART_CR1） ..... 1056
34.6.5 控制寄存器 2（UART_CR2） ..... 1057
34.6.6 控制寄存器 3（UART_CR3） ..... 1059
35 调试支持（DBG） ..... 1060
35.1.1 对定时器、RTC、看门狗和 I2C 的调试支持 ..... 1060
35.1.2 MCU 调试配置写/器件 ID 寄存器（IDCODE_CRW） ..... 1060
35.1.3 MCU 调试配置写/器件 ID 寄存器（IDCODE_CRW） ..... 1061
35.1.4 MCU 调试配置读寄存器（DBGMCU_CRR） ..... 1062
35.1.5 MCU APB1 调试冻结寄存器（DBGMCU_APB1_FZR1） ..... 1062
35.1.6 MCU APB2 调试冻结寄存器（DBGMCU_APB2_FZR） ..... 1064
36 器件电子签名 ..... 1065
36.1 闪存容量寄存器 ..... 1065
36.2 产品唯一身份标识寄存器（96 位） ..... 1065
37 版本历史 ..... 1067

# 表目录

表 3.1 RX32G410 系列存储器映像和外设边界地址 45
表 3.2 Boot 模式 48
表 4.1 Flash 结构（未开启 ECC） 49
表 4.2 Flash 结构（开启 ECC） 50
表 4.3 等待周期和 CPU 时钟频率对应关系 50
表 4.4 选项字节格式 53
表 6.1 低功耗模式一览 69
表 9.1 端口位配置表 (1) 120
表 11.1 EXTI 线路连接 140
表 12.1 CLC 引脚复用功能表 149
表 12.2 CLC 输入源 149
表 12.3 逻辑功能示例 150
表 13.1 CORDIC 函数 158
表 13.2 余弦参数 158
表 13.3 正弦参数 159
表 13.4 相位参数 159
表 13.5 模参数 160
表 13.6 反正切参数 160
表 13.7 双曲余弦参数 161
表 13.8 双曲正弦参数 161
表 13.9 双曲反正切参数 162
表 13.10 自然对数参数 162
表 13.11 自然对数缩放系数及对应范围 162
表 13.12 平方根参数 163
表 13.13 平方根缩放系数及对应范围 163
表 13.14 精度与迭代次数 166
表 14.1 读取和写入方法的有效组合 186
表 15.1 ADC 内部输入/输出信号 202
表 15.2 ADC 输入/输出引脚 202
表 15.3 ADC1 通道 204
表 15.4 ADC2 通道 204
表 15.5 ADC3 通道 205
表 15.6 模拟看门狗通道选择 208
表 15.7 ADC1 规则通道的外部触发信号 211
表 15.8 ADC2 规则通道的外部触发信号 212
表 15.9 ADC3 规则通道的外部触发信号 212
表 15.10 ADC1 注入通道的外部触发信号 213
表 15.11 ADC2 注入通道的外部触发信号 213
表 15.12 ADC3 注入通道的外部触发信号 214
表 15.13 ADC 中断 222
表 16.1 DAC 特征 236

表 16.2 DAC 输入/输出引脚 238
表 16.3 DAC 输入/输出信号 238
表 16.4 DAC1 通道 1 互连 239
表 16.5 DAC1 通道 2 互连 239
表 16.6 DAC2 通道 1 互连 240
表 16.7 DAC2 通道 2 互连 240
表 16.8 DAC3 通道 1 互连 241
表 16.9 DAC3 通道 2 互连 241
表 16.10 HFSEL 描述 243
表 16.11 低功耗模式对 DAC 的影响 254
表 16.12 DAC 中断 254
表 17.1 DMA 实现 267
表 17.2 DMA 内部输入输出信号 269
表 17.3 可编程的数据传输宽度和大小端操作（当 PINC = MINC = 1） 273
表 17.4 DMA 中断请求 274
表 18.1 DMAMUX 实例化 283
表 18.2 DMAMUX：复用器输入到资源的分配 283
表 18.3 DMAMUX：触发输入到资源的分配 285
表 18.4 DMAMUX：同步输入到资源的分配 285
表 18.5 DMAMUX 信号 287
表 18.6 DMAMUX 中断 291
表 19.1 COMPx 正向输入分配 296
表 19.2 COMPx 反向输入分配 296
表 19.3 COMPx_P 端和 ADC 的 IO 对应关系 297
表 19.4 COMPx_N 端和 ADC 的 IO 对应关系 297
表 19.5 遮蔽源 298
表 19.6 比较器在低功耗模式下的行为 299
表 20.1 运算放大器的可能连接方式 305
表 20.2 工作模式及校正 312
表 20.3 低功耗模式对 OPAMP 的影响 313
表 21.1 HRTIM 输入/输出汇总 323
表 21.2 外部事件映射及相关功能 325
表 21.3 更新使能输入和源 325
表 21.4 来自通用定时器的突发模式时钟源 325
表 21.5 突发模式触发源 326
表 21.6 内部同步源 326
表 21.7 故障输入 326
表 21.8 HRTIM DAC 触发器连接 326
表 21.9 定时器分辨率和最小 PWM 频率（$f_{HRTIM}$ = 192MHz 时） 327
表 21.10 周期寄存器与比较寄存器的最小和最大值 331
表 21.11 定时器工作模式 331
表 21.12 定时器 A 到 F 之间的事件映射 337
表 21.13 交错模式选择 338

表 21.14 交错模式下比较 1 至 3 的值 339
表 21.15 死区分辨率和最大绝对值 350
表 21.16 翻转事件目标和模式编程 357
表 21.17 EExFLTR[3:0]代码取决于 UDM 位设置 359
表 21.18 外部事件特征 365
表 21.19 输出置位/复位延迟和抖动与外部事件工作模式的关系 366
表 21.20 每个定时器的过滤信号映射 369
表 21.21 每个定时器的窗口信号映射（EEFLTR[3:0] = 1111） 371
表 21.22 HRTIM 可预装载控制寄存器和关联的更新源 381
表 21.23 主定时器更新事件传播 386
表 21.24 TIMx 更新事件传播 386
表 21.25 能够生成更新事件的复位事件 388
表 21.26 用于定时器复位的更新事件传播 388
表 21.27 输出状态编程，x= A..F，y = 1 或 2 390
表 21.28 突发模式的定时器输出编程 393
表 21.29 采样速率和滤波器长度与 FLTFxF[3:0]和时钟设置的关系 403
表 21.30 故障输入消隐事件 404
表 21.31 故障 1 至 6 计数器复位源 405
表 21.32 同步事件的作用与定时器工作模式之间的关系 409
表 21.33 DAC 双通道触发示例 417
表 21.34 HRTIM 中断汇总 420
表 21.35 HRTIM DMA 请求总结 422
表 22.1 TIM 输入/输出引脚 545
表 22.2 TIM 内部输入/输出信号 545
表 22.3 连接到 tim_ti1 的输入多路复用器 546
表 22.4 连接到 tim_ti2 的输入多路复用器 546
表 22.5 连接到 tim_ti3 的输入多路复用器 546
表 22.6 连接到 tim_ti4 的输入多路复用器 547
表 22.7 TIMx 内部触发连接 547
表 22.8 连接到 tim_etr 的输入多路复用器 548
表 22.9 定时器刹车互连 548
表 22.10 定时器刹车 2 互连 548
表 22.11 系统中断互连 549
表 22.12 连接到 ocref_clr 的输入多路复用器 549
表 22.13 CCR 和 ARR 寄存器的变化抖动模式 579
表 22.14 CCR 寄存器在中心对齐 PWM 模式下的变化抖动模式 580
表 22.15 刹车保护解除条件 591
表 22.16 计数方向与编码器信号的关系（CC1P = CC2P = 0） 599
表 22.17 计数方向与编码器信号及极性设置的关系 603
表 22.18 DMA 请求 625
表 22.19 低功耗模式对 TIM1/TIM8 的影响 625
表 22.20 中断请求 625
表 22.21 带刹车功能的互补通道 tim_ocx 和 tim_ocxn 的输出控制位 653

表 23.1 RX32G410 系列通用定时器 678
表 23.2 TIM 输入/输出引脚 680
表 23.3 TIM 内部输入/输出信号 680
表 23.4 连接到 tim_ti1 的输入多路复用器 681
表 23.5 连接到 tim_ti2 的输入多路复用器 681
表 23.6 连接到 tim_ti3 的输入多路复用器 681
表 23.7 连接到 tim_ti4 的输入多路复用器 681
表 23.8 TIMx 内部触发连接 682
表 23.9 连接到 tim_etr 的输入多路复用器 682
表 23.10 连接到 ocref_clr 的输入多路复用器 683
表 23.11 CCR 和 ARR 寄存器的变化抖动模式 709
表 23.12 CCR 寄存器在中心对齐 PWM 模式下的变化抖动模式 710
表 23.13 计数方向与编码器信号的关系（CC1P = CC2P = 0） 720
表 23.14 计数方向与编码器信号及极性设置的关系 723
表 23.15 DMA 请求 747
表 23.16 低功耗模式对 TIM2/TIM3/TIM4/TIM5 的影响 748
表 23.17 中断请求 748
表 23.18 标准 tim_ocx 通道的输出控制位 767
表 24.1 TIM 输入/输出引脚 787
表 24.2 TIM 内部输入/输出信号 787
表 24.3 连接到 tim_ti1 的输入多路复用器 788
表 24.4 连接到 tim_ti2 的输入多路复用器 788
表 24.5 TIMx 内部触发连接 788
表 24.6 定时器刹车互连 789
表 24.7 系统中断互连 789
表 24.8 连接到 ocref_clr 的输入多路复用器 789
表 24.9 CCR 和 ARR 寄存器的变化抖动模式 808
表 24.10 刹车保护解除条件 816
表 24.11 DMA 请求 825
表 24.12 低功耗模式对 TIM15/TIM16/TIM17 的影响 825
表 24.13 中断请求 825
表 24.14 带刹车功能的互补输出通道 tim_oc1 和 tim_oc1n 的控制位（TIM15） 841
表 24.15 带刹车功能的互补输出通道 tim_oc1 和 tim_oc1n 的控制位（TIM16/TIM17） 864
表 25.1 TIM 内部输入/输出信号 877
表 25.2 DMA 请求 884
表 25.3 低功耗模式对 TIM6/TIM7 的影响 884
表 25.4 中断请求 884
表 26.1 VREF 缓冲器模式 890
表 28.1 看门狗超时时间（32kHz 的输入时钟（LSI））(1) 904
表 30.1 双缓冲批量端点缓冲区标识定义 920
表 30.2 双缓冲批量端点的缓冲区使用标识 921
表 30.3 同步端点的缓冲区使用标识 922
表 30.4 唤醒事件检测 923

表 30.5 接收状态编码 ..... 934
表 30.6 端点类型编码 ..... 934
表 30.7 端点特殊类型定义 ..... 934
表 30.8 发送状态编码 ..... 934
表 30.9 分组缓冲区大小的定义 ..... 938
表 31.1 发送邮箱寄存器列表 ..... 953
表 31.2 接收邮箱寄存器列表 ..... 953
表 32.1 SPI 中断请求 ..... 996
表 33.1 SMBus 与 I²C 的比较 ..... 1015
表 33.2 I²C 中断请求 ..... 1018
表 34.1 检测噪声的数据采样 ..... 1040
表 34.2 设置波特率时的误差计算 ..... 1041
表 34.3 当 DIV_Fraction=0 时，UART 接收器的容忍度 ..... 1042
表 34.4 当 DIV_Fraction!=0 时，UART 接收器的容忍度 ..... 1042
表 34.5 帧格式 ..... 1044
表 34.6 UART 中断请求 ..... 1051
表 34.7 UART 模式设置 (1) ..... 1052
表 37.1 版本历史 ..... 1067

# 图目录

图 3.1 存储器映像 ..... 44
图 5.1 CRC 计算单元框图 ..... 65
图 6.1 电源框图 ..... 67
图 6.2 上电复位和掉电复位波形图 ..... 68
图 6.3 PVD 门限 ..... 68
图 8.1 复位电路框图 ..... 80
图 8.2 时钟树 ..... 83
图 8.3 HSE 时钟源 ..... 84
图 8.4 使用 TIM15 捕获模式进行频率测量 ..... 89
图 8.5 使用 TIM16 捕获模式进行频率测量 ..... 89
图 8.6 使用 TIM17 捕获模式进行频率测量 ..... 90
图 8.7 使用 TIM5 捕获模式进行频率测量 ..... 90
图 9.1 I/O 端口位基本结构 ..... 119
图 9.2 5V 容忍 I/O 端口位的基本结构 ..... 119
图 9.3 输入浮空/上拉/下拉配置 ..... 123
图 9.4 输出配置 ..... 124
图 9.5 复用功能配置 ..... 125
图 9.6 高阻态模拟配置 ..... 125
图 11.1 可配置的中断/事件框图 ..... 138
图 11.2 外部中断/事件 GPIO 映射 ..... 140
图 12.1 可配置逻辑单元 ..... 146
图 12.2 CLC 输入源选择 ..... 147
图 12.3 逻辑功能组合选项 ..... 148
图 13.1 CORDIC 在三角函数中的收敛情况 ..... 164
图 13.2 CORDIC 在双曲函数中的收敛情况 ..... 165
图 13.3 CORDIC 在平方根函数中的收敛情况 ..... 166
图 14.1 FMAC 模块框图 ..... 174
图 14.2 输入缓冲区 ..... 176
图 14.3 循环输入缓冲区 ..... 177
图 14.4 循环输入缓冲区操作 ..... 178
图 14.5 循环输出缓冲区 ..... 179
图 14.6 循环输出缓冲区操作 ..... 180
图 14.7 FIR 滤波器结构 ..... 182
图 14.8 IIR 滤波器结构（直接 1 型） ..... 184
图 14.9 X1 缓冲区初始化 ..... 189
图 14.10 滤波示例 1 ..... 190
图 14.11 滤波示例 2 ..... 191
图 15.1 ADC 框图 ..... 201
图 15.2 ADC 时钟方案 ..... 203
图 15.3 校准时序图 ..... 205
图 15.4 ADC 转换时序图 ..... 207

图 15.5 模拟看门狗警戒区 208
图 15.6 注入转换延迟 209
图 15.7 数据右对齐 210
图 15.8 数据左对齐 210
图 15.9 双 ADC 框图 216
图 15.10 4 个通道的同步注入模式 217
图 15.11 16 个通道的同步规则模式 217
图 15.12 在 1 个通道上连续转换模式下的快速交叉模式 218
图 15.13 在 1 个通道上的慢速交叉模式 219
图 15.14 交替触发：每个 ADC1 的注入通道组 219
图 15.15 交替触发：在间断模式下每个 ADC 上的 4 个注入通道 220
图 15.16 交替+规则同步 220
图 15.17 触发事件发生在注入转换期间 221
图 15.18 交叉的单通道转换被注入序列 CH11 和 CH12 中断 221
图 15.19 温度传感器和 $V_{REFINT}$ 通道框图 222
图 16.1 双通道 DAC 框图 237
图 16.2 DAC 单通道模式下的数据寄存器 242
图 16.3 DAC 双通道模式下的数据寄存器 243
图 16.4 触发禁用 TEN=0 时的转换时序图 244
图 16.5 DAC LFSR 寄存器计算算法 245
图 16.6 LFSR 产生波形的 DAC 转换（使能软件触发） 246
图 16.7 生成 DAC 三角波 246
图 16.8 生成三角波波形的 DAC 转换（使能软件触发） 247
图 16.9 DAC 生成锯齿波（STDIRx = 0） 247
图 16.10 DAC 生成锯齿波（STDIRx = 1） 248
图 16.11 DAC 锯齿波 STINCTRIG 和 STRSTTRIG 优先级（STDIR = 0） 248
图 17.1 DMA 框图 268
图 18.1 DMAMUX 框图 286
图 18.2 DMAMUX 请求线复用器通道的同步模式 289
图 18.3 DMA 请求线复用器通道的事件生成 289
图 19.1 COMP 框图 296
图 19.2 比较器迟滞 298
图 19.3 比较器输出遮蔽图 299
图 20.1 独立模式：外部增益设置模式 306
图 20.2 电压跟随器模式 307
图 20.3 PGA 模式单端输入（P 端无偏置电压） 309
图 20.4 PGA 模式差分输入（P 端接偏置电压） 309
图 20.5 PGA 模式（VINM_N2 用于滤波） 311
图 21.1 高分辨率定时器框图 322
图 21.2 计数器和捕获寄存器格式与时钟预分频系数 328
图 21.3 定时器 A..F 框图 330
图 21.4 连续定时器工作模式 332
图 21.5 单发定时器工作模式 332

图 21.6 定时器复位重新同步（预分频比大于 32）.....334
图 21.7 连续模式下的重复率与 HRTIM_REPxR 内容.....335
图 21.8 单发模式下的重复计数器行为.....335
图 21.9 比较事件对输出的操作：发生比较 1 时置位，发生比较 2 时复位.....337
图 21.10 定时器 A 定时单元捕获电路.....341
图 21.11 自动延迟概述 (仅限比较 2 寄存器).....342
图 21.12 自动延迟比较.....343
图 21.13 半触发模式样例.....345
图 21.14 推挽模式框图.....346
图 21.15 推挽模式样例.....347
图 21.16 带死区的推挽.....348
图 21.17 已插入死区的互补输出.....349
图 21.18 死区插入与死区符号（1 表示负死区）.....349
图 21.19 低脉冲宽度的互补输出 (SDTRx = SDTFx = 0).....350
图 21.20 低脉冲宽度的互补输出 (SDTRx = SDTFx = 1).....350
图 21.21 低脉冲宽度的互补输出 (SDTRx = 0, SDTFx = 1).....351
图 21.22 低脉冲宽度的互补输出 (SDTRx = 1, SDTFx=0).....351
图 21.23 主定时器概览.....352
图 21.24 在上下计数模式时的基本对称波形.....353
图 21.25 在上下计数模式时的复杂对称波形.....354
图 21.26 在上下计数模式时的基本对称波形.....354
图 21.27 在上下计数模式时的外部事件管理.....355
图 21.28 纵横的上下计数操作.....355
图 21.29 交叉的上下计数操作.....356
图 21.30 推挽上下模式样例.....356
图 21.31 带“大于”比较功能的的上下计数模式.....357
图 21.32 输出在周期事件上设置的上下模式，OUTROM[1:0]=10.....358
图 21.33 在上下计数模式时重复计数器的表现.....359
图 21.34 窄脉冲生成的短距离置位/复位管理.....361
图 21.35 外部事件调节概述(代表 1 个通道).....364
图 21.36 外部事件下降沿的延迟（计数器复位和输出置位）.....367
图 21.37 外部事件的延迟（发生外部事件时，输出复位）.....367
图 21.38 事件消隐模式.....368
图 21.39 事件延迟模式.....368
图 21.40 采用边沿有效触发的外部触发消隐.....370
图 21.41 外部触发消隐，电平有效触发.....370
图 21.42 事件窗口模式.....371
图 21.43 采用边沿有效触发的外部触发窗口.....372
图 21.44 外部触发窗口，电平有效触发.....372
图 21.45 外部触事件计数器 – 通道 A.....373
图 21.46 外部事件计数器累积模式（EEVxRSTM = 1，EEVxCNT = 2）.....374
图 21.47 延迟空闲模式进入.....375
图 21.48 突发模式和延迟保护优先级（DIDL = 0）.....376

图 21.49 突发模式和延迟保护优先级（DIDL = 1） 377
图 21.50 均衡空闲保护示例 378
图 21.51 重同步定时器更新（在 HRTIM_TIMBCR 中 TAU=1） 383
图 21.52 在“大于”PWM 模式下提前开启和提前关闭的行为 385
图 21.53 输出管理概览 390
图 21.54 HRTIM 输出状态和转换 391
图 21.55 突发模式工作示例 392
图 21.56 发生外部事件时触发突发模式 395
图 21.57 使能死区且 IDLESx = 1 时的延迟突发模式进入 396
图 21.58 死区内的延迟突发模式进入 397
图 21.59 死区发生器使能时的突发模式退出 398
图 21.60 突发模式仿真示例 400
图 21.61 载波频率信号插入 400
图 21.62 已使能斩波模式的 HRTIM 输出 401
图 21.63 故障保护电路（完全表示了 FAULT1，部分显示了 FAULT2..6） 402
图 21.64 故障信号过滤（FLTxF[3:0]= 0010：$f_{SAMPLING} = f_{HRTIM}$，N = 4） 403
图 21.65 故障计数器累加模式（FLTxRSTM = 1，FLTxCNT[3:0] = 2） 404
图 21.66 辅助输出 406
图 21.67 突发模式期间的辅助输出和主输出（DIDLx = 0） 407
图 21.68 退出突发模式时辅助输出上的死区失真 407
图 21.69 同步启动模式下计数器的行为 411
图 21.70 ADC 触发选择概述 412
图 21.71 ADC 触发 413
图 21.72 向上计数模式下的 ADC 触发后缩放 414
图 21.73 向上/向下计数模式下的 ADC 触发后缩放 414
图 21.74 将多个更新合并到一路 hrtim_dac_trgx 输出 415
图 21.75 DAC 斜率补偿触发 418
图 21.76 DAC 触发器概述 419
图 21.77 DMA 突发概览 423
图 21.78 突发 DMA 工作流程图 424
图 21.79 DMA 突发传输后进行寄存器更新 425
图 21.80 降压转换器拓扑结构 427
图 21.81 双降压转换器管理 428
图 21.82 同步整流取决于输出电流 428
图 21.83 采用同步整流功能实现降压 429
图 21.84 三相交错降压转换器 429
图 21.85 三相交错降压转换器控制 430
图 21.86 过渡模式功率因数校正（PFC） 431
图 21.87 过渡模式功率因数校正（PFC）波形 431
图 22.1 高级控制定时器框图 544
图 22.2 当预分频器的参数从 1 变成 2 时，计数器的时序图 551
图 22.3 当预分频器的参数从 1 变成 4 时，计数器的时序图 551
图 22.4 计数器时序图，内部时钟分频系数为 1 552

图 22.5 计数器时序图，内部时钟分频系数为 2 ..... 553
图 22.6 计数器时序图，内部时钟分频系数为 4 ..... 553
图 22.7 计数器时序图，内部时钟分频系数为 N ..... 554
图 22.8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入） ..... 554
图 22.9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR） ..... 555
图 22.10 计数器时序图，内部时钟分频系数为 1 ..... 556
图 22.11 计数器时序图，内部时钟分频系数为 2 ..... 557
图 22.12 计数器时序图，内部时钟分频系数为 4 ..... 557
图 22.13 计数器时序图，内部时钟分频系数为 N ..... 557
图 22.14 计数器时序图，当没有使用重复计数器时的更新事件 ..... 558
图 22.15 计数器时序图，内部时钟分频系数为 1，TIMx_ARR=0x6 ..... 559
图 22.16 计数器时序图，内部时钟分频系数为 2 ..... 560
图 22.17 计数器时序图，内部时钟分频系数为 4，TIMx_ARR=0x36 ..... 560
图 22.18 计数器时序图，内部时钟分频系数为 N ..... 561
图 22.19 计数器时序图，ARPE=1 时的更新事件（计数器下溢） ..... 561
图 22.20 计数器时序图，ARPE=1 时的更新事件（计数器上溢） ..... 562
图 22.21 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置 ..... 563
图 22.22 外部触发输入框图 ..... 564
图 22.23 一般模式下的控制电路，内部时钟分频系数为 1 ..... 564
图 22.24 外部触发输入框图 ..... 565
图 22.25 外部时钟模式 1 下的控制电路 ..... 566
图 22.26 外部触发输入框图 ..... 566
图 22.27 外部时钟模式 2 下的控制电路 ..... 567
图 22.28 捕获/比较通道（如：通道 1 输入部分） ..... 568
图 22.29 捕获/比较通道 1 的主电路 ..... 568
图 22.30 捕获/比较通道的输出阶段（通道 1，通道 2、3 和 4 同上） ..... 569
图 22.31 捕获/比较通道的输出阶段（通道 5，通道 6 同上） ..... 569
图 22.32 PWM 输入模式时序 ..... 571
图 22.33 输出比较模式，翻转 OC1 ..... 572
图 22.34 边沿对齐的 PWM 波形（ARR=8） ..... 574
图 22.35 中央对齐的 PWM 波形（ARR=8） ..... 575
图 22.36 抖动原理 ..... 576
图 22.37 抖动模式下的数据格式和寄存器编码 ..... 577
图 22.38 PWM 分辨率 vs 频率 ..... 578
图 22.39 PWM 抖动模式 ..... 579
图 22.40 中心对齐 PWM 模式下抖动对占空比的影响 ..... 580
图 22.41 生成两个具有 50%占空比的相位偏移 PWM 信号 ..... 581
图 22.42 通道 1 和通道 3 的组合 PWM 模式 ..... 582
图 22.43 每周期具有多个触发脉冲的三相组合 PWM 信号 ..... 583
图 22.44 带死区插入的互补输出 ..... 584
图 22.45 非对称死区时间 ..... 585
图 22.46 死区波形延迟大于负脉冲 ..... 585
图 22.47 死区波形延迟大于正脉冲 ..... 585

图 22.48 刹车和刹车 2 电路概述 ..... 587
图 22.49 tim_brk 在发生刹车事件时的各种输出行为（OSSI=1） ..... 589
图 22.50 当 tim_brk/tim_brk2 有效（OSSI=0）时 PWM 输出状态 ..... 590
图 22.51 输出重定向（未表示 tim_brk2 请求） ..... 591
图 22.52 tim_ocref_clr 输入选择复用器 ..... 592
图 22.53 清除 TIMx 的 tim_ocxref ..... 593
图 22.54 COM 事件生成 6 步 PWM 的示例（OSSR=1） ..... 594
图 22.55 单脉冲模式的例子 ..... 595
图 22.56 可重触发单脉冲模式 ..... 596
图 22.57 脉冲发生器电路 ..... 597
图 22.58 在比较事件上的脉冲生成，适用于边沿对齐模式 ..... 597
图 22.59 在同时触发的情况下延长脉冲宽度 ..... 598
图 22.60 编码器接口模式下的计数器工作示例 ..... 600
图 22.61 tim_ti1fp1 极性反相时的编码器接口模式示例 ..... 600
图 22.62 正交编码器的计数模式 ..... 601
图 22.63 方向加时钟编码器模式 ..... 602
图 22.64 方向时钟编码器模式（CC1P=CC2P=0） ..... 602
图 22.65 方向时钟编码器模式（CC1P=CC2P=1） ..... 603
图 22.66 索引门控选项 ..... 604
图 22.67 抖动的索引信号 ..... 604
图 22.68 IPOS[1:0] = 11 时的索引生成 ..... 605
图 22.69 通道 A 上带索引门控的计数器读取（IPOS[1:0] = 11） ..... 606
图 22.70 非门控模式下索引的计数器读取（IPOS[1:0] = 00） ..... 606
图 22.71 通道 A 和 B 上带索引门控的计数器读取 ..... 607
图 22.72 窄索引脉冲情况下的编码器模式行为（IPOS[1:0] = 11） ..... 608
图 22.73 计数器重置窄索引脉冲（更近距离，ARR = 0x07） ..... 609
图 22.74 x1 和 x2 模式下的索引行为（IPOS[1:0] = 01） ..... 610
图 22.75 定向索引灵敏度 ..... 611
图 22.76 作为 FIDX 位设置的功能的计数器重置 ..... 611
图 22.77 时钟+方向模式下的索引行为，IPOS[0] = 1 ..... 612
图 22.78 方向时钟模式下的索引行为，IPOS[0] = 1 ..... 612
图 22.79 正交编码信号的状态图 ..... 613
图 22.80 递增计数编码器错误检测 ..... 614
图 22.81 递减计数编码器错误检测 ..... 615
图 22.82 更新时带有预加载转移的编码器模式更改（SMSPS = 0） ..... 616
图 22.83 测量 3 个信号边缘之间的时间间隔 ..... 617
图 22.84 霍尔传感器接口的实例 ..... 619
图 22.85 复位模式下的控制电路 ..... 620
图 22.86 门控模式下的控制电路 ..... 621
图 22.87 触发器模式下的控制电路 ..... 622
图 22.88 外部时钟模式 2+触发模式下的控制电路 ..... 623
图 23.1 通用定时器框图 ..... 679
图 23.2 当预分频器的参数从 1 变到 2 时，计数器的时序图 ..... 684

图 23.3 当预分频器的参数从 1 变到 4 时，计数器的时序图 684
图 23.4 计数器时序图，内部时钟分频系数为 1 685
图 23.5 计数器时序图，内部时钟分频系数为 2 686
图 23.6 计数器时序图，内部时钟分频系数为 4 686
图 23.7 计数器时序图，内部时钟分频系数为 N 687
图 23.8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入） 687
图 23.9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR） 688
图 23.10 计数器时序图，内部时钟分频系数为 1 689
图 23.11 计数器时序图，内部时钟分频系数为 2 689
图 23.12 计数器时序图，内部时钟分频系数为 4 690
图 23.13 计数器时序图，内部时钟分频系数为 N 690
图 23.14 计数器时序图，更新事件 691
图 23.15 计数器时序图，内部时钟分频系数为 1，TIMx_ARR=0x6 692
图 23.16 计数器时序图，内部时钟分频系数为 2 693
图 23.17 计数器时序图，内部时钟分频系数为 4，TIMx_ARR=0x36 693
图 23.18 计数器时序图，内部时钟分频系数为 N 694
图 23.19 计数器时序图，ARPE=1 时的更新事件（计数器下溢） 694
图 23.20 计数器时序图，ARPE=1 时的更新事件（计数器溢出） 695
图 23.21 一般模式下的控制电路，内部时钟分频系数为 1 696
图 23.22 tim_ti2 外部时钟连接例子 696
图 23.23 外部时钟模式 1 下的控制电路 697
图 23.24 外部触发输入框图 697
图 23.25 外部时钟模式 2 下的控制电路 698
图 23.26 捕获/比较通道（如：通道 1 输入部分） 699
图 23.27 捕获/比较通道 1 的主电路 699
图 23.28 捕获/比较通道的输出阶段（通道 1，通道 2、3 和 4 同上） 700
图 23.29 PWM 输入模式时序 702
图 23.30 输出比较模式，翻转 tim_oc1 703
图 23.31 边沿对齐的 PWM 波形（ARR=8） 704
图 23.32 中央对齐的 PWM 波形（APR=8） 705
图 23.33 抖动原理 706
图 23.34 抖动模式下的数据格式和寄存器编码 707
图 23.35 PWM 分辨率 vs 频率（16 位模式） 708
图 23.36 PWM 分辨率 vs 频率（28 位模式） 708
图 23.37 PWM 抖动模式 709
图 23.38 中心对齐 PWM 模式下抖动对占空比的影响 710
图 23.39 生成两个具有 50%占空比的相位偏移 PWM 信号 711
图 23.40 通道 1 和通道 3 的组合 PWM 模式 712
图 23.41 tim_ocref_clr 输入选择复用器 713
图 23.42 清除 TIMx 的 tim_ocxref 714
图 23.43 单脉冲模式的例子 715
图 23.44 可重触发单脉冲模式 716
图 23.45 脉冲发生器电路 717

图 23.46 在比较事件上的脉冲生成，适用于边沿对齐模式和编码器模式 718
图 23.47 在同时触发的情况下延长脉冲宽度 719
图 23.48 编码器接口模式下的计数器工作示例 720
图 23.49 tim_ti1fp1 极性反相时的编码器接口模式示例 721
图 23.50 正交编码器的计数模式 721
图 23.51 方向加时钟编码器模式 722
图 23.52 方向时钟编码器模式（CC1P=CC2P=0） 723
图 23.53 方向时钟编码器模式（CC1P=CC2P=1） 723
图 23.54 索引门控选项 724
图 23.55 抖动的索引信号 725
图 23.56 IPOS[1:0] = 11 时的索引生成 726
图 23.57 通道 A 上带索引门控的计数器读取（IPOS[1:0] = 11） 726
图 23.58 非门控模式下索引的计数器读取（IPOS[1:0] = 00） 727
图 23.59 通道 A 和 B 上带索引门控的计数器读取 727
图 23.60 窄索引脉冲情况下的编码器模式行为（IPOS[1:0] = 11） 728
图 23.61 计数器重置窄索引脉冲（更近距离，ARR = 0x07） 729
图 23.62 x1 和 x2 模式下的索引行为（IPOS[1:0] = 01） 730
图 23.63 定向索引灵敏度 731
图 23.64 作为 FIDX 位设置的功能的计数器重置 731
图 23.65 时钟+方向模式下的索引行为，IPOS[0] = 1 732
图 23.66 方向时钟模式下的索引行为，IPOS[0] = 1 732
图 23.67 正交编码信号的状态图 733
图 23.68 递增计数编码器错误检测 734
图 23.69 递减计数编码器错误检测 735
图 23.70 更新时带有预加载转移的编码器模式更改（SMSPS = 0） 736
图 23.71 复位模式下的控制电路 738
图 23.72 门控模式下的控制电路 739
图 23.73 触发器模式下的控制电路 740
图 23.74 外部时钟模式 2+触发模式下的控制电路 741
图 23.75 主/从定时器的例子 742
图 23.76 仅定时器通道 1 主/从连接示例 742
图 23.77 TIM_mstr 的 tim_oc1ref 控制 TIM_slv 743
图 23.78 通过使能 TIM_mstr 可以控制 TIM_slv 744
图 23.79 使用 TIM_mstr 的更新触发 TIM_slv 745
图 23.80 利用 TIM_mstr 的使能触发 TIM_slv 745
图 23.81 使用 TIM_mstr 的 tim_ti1 输入触发 TIM_mstr 和 TIM_slv 746
图 24.1 TIM15 框图 785
图 24.2 TIM16/TIM17 框图 786
图 24.3 当预分频器的参数从 1 变到 2 时，计数器的时序图 791
图 24.4 当预分频器的参数从 1 变到 4 时，计数器的时序图 791
图 24.5 计数器时序图，内部时钟分频系数为 1 792
图 24.6 计数器时序图，内部时钟分频系数为 2 793
图 24.7 计数器时序图，内部时钟分频系数为 4 793

图 24.8 计数器时序图，内部时钟分频系数为 N.....794
图 24.9 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）.....795
图 24.10 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）.....795
图 24.11 取决于模式和 TIMx_RCR 寄存器设置的更新频率示例.....796
图 24.12 一般模式下的控制电路，内部时钟分频系数为 1.....797
图 24.13 tim_ti2 外部时钟连接例子.....797
图 24.14 外部时钟模式 1 下的控制电路.....798
图 24.15 捕获/比较通道（如：通道 1 输入部分）.....798
图 24.16 捕获/比较通道 1 的主电路.....799
图 24.17 捕获/比较通道的输出阶段（通道 1）.....799
图 24.18 捕获比较通道的输出阶段（TIM15 的通道 2）.....800
图 24.19 PWM 输入模式时序.....802
图 24.20 输出比较模式，翻转 tim_oc1.....803
图 24.21 边沿对齐的 PWM 波形（ARR=8）.....804
图 24.22 抖动原理.....805
图 24.23 抖动模式下的数据格式和寄存器编码.....806
图 24.24 PWM 分辨率 vs 频率.....807
图 24.25 PWM 抖动模式.....807
图 24.26 通道 1 和通道 2 的组合 PWM 模式.....809
图 24.27 带死区插入的互补输出.....810
图 24.28 非对称死区时间.....811
图 24.29 死区波形延迟大于负脉冲.....811
图 24.30 死区波形延迟大于正脉冲.....811
图 24.31 刹车电路概述.....813
图 24.32 tim_brk 在发生刹车事件时的各种输出行为（OSSI=1）.....815
图 24.33 输出重定向.....817
图 24.34 tim_ocref_clr 输入选择复用器.....817
图 24.35 单脉冲模式的例子.....818
图 24.36 可重触发单脉冲模式.....819
图 24.37 测量两个信号边沿之间的间隔时间.....820
图 24.38 复位模式下的控制电路.....821
图 24.39 门控模式下的控制电路.....822
图 24.40 触发器模式下的控制电路.....823
图 25.1 基本定时器框图.....876
图 25.2 一般模式下的控制电路，内部时钟分频系数为 1.....877
图 25.3 当预分频器的参数从 1 变到 2 时，计数器的时序图.....878
图 25.4 当预分频器的参数从 1 变到 4 时，计数器的时序图.....879
图 25.5 计数器时序图，内部时钟分频系数为 1.....880
图 25.6 计数器时序图，内部时钟分频系数为 2.....880
图 25.7 计数器时序图，内部时钟分频系数为 4.....881
图 25.8 计数器时序图，内部时钟分频系数为 N.....881
图 25.9 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）.....882
图 25.10 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）.....883

图 27.1 简化的 RTC 框图 ..... 895
图 27.2 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004 ..... 897
图 27.3 RTC 溢出波形图示例，PR=0003 ..... 897
图 28.1 独立看门狗框图 ..... 904
图 29.1 窗口看门狗框图 ..... 907
图 29.2 窗口看门狗时序图 ..... 909
图 30.1 USB 框图 ..... 913
图 30.2 分组缓冲区对应的缓冲区描述表项定位 ..... 917
图 31.1 CAN 网拓扑结构 ..... 940
图 31.2 CAN 框图 ..... 941
图 31.3 bxCAN 工作模式 ..... 943
图 31.4 bxCAN 工作在静默模式 ..... 944
图 31.5 bxCAN 工作在环回模式 ..... 944
图 31.6 bxCAN 工作在环回静默模式 ..... 945
图 31.7 发送邮箱状态 ..... 947
图 31.8 接收 FIFO 状态 ..... 948
图 31.9 过滤器组位宽设置—寄存器组织 ..... 950
图 31.10 过滤器编号的例子 ..... 951
图 31.11 过滤器机制的例子 ..... 952
图 31.12 CAN 错误状态图 ..... 954
图 31.13 位时序 ..... 956
图 31.14 各种 CAN 帧 ..... 957
图 31.15 事件标志和中断产生 ..... 958
图 31.16 RX 和 TX 邮箱 ..... 971
图 32.1 SPI 框图 ..... 982
图 32.2 单主和单从应用 ..... 983
图 32.3 数据时钟时序图 ..... 985
图 32.4 主/全双工模式下（BIDIMODE=0 且 RXONLY=0）连续传输时，TXE/RXNE/BSY 行为 ..... 990
图 32.5 从/全双工模式下（BIDIMODE=0 且 RXONLY=0）连续传输时，TXE/RXNE/BSY 行为 ..... 990
图 32.6 主设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 行为 ..... 991
图 32.7 从设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 行为 ..... 991
图 32.8 只接收模式（BIDIMODE=0 并且 RXONLY=1）下连续传输时，RXNE 行为 ..... 992
图 32.9 非连续传输发送（BIDIMODE=0 并且 RXONLY=0）时，TXE/BSY 行为 ..... 993
图 32.10 使用 DMA 发送 ..... 995
图 32.11 使用 DMA 接收 ..... 995
图 33.1 I²C 总线协议 ..... 1003
图 33.2 I²C 的功能框图 ..... 1004
图 33.3 从发送器的传送序列图 ..... 1006
图 33.4 从接收器的传送序列图 ..... 1007
图 33.5 主发送器传送序列图 ..... 1009
图 33.6 方法 1：主接收器传送序列图 ..... 1010
图 33.7 方法 2：N>2 时主接收器传送序列图 ..... 1011
图 33.8 方法 2：N=2 时主接收器传送序列图 ..... 1012

图 33.9 方法 2：N=1 时主接收器传送序列图 ..... 1013
图 33.10 I²C 中断映射图 ..... 1019
图 34.1 UART 框图 ..... 1034
图 34.2 字长设置 ..... 1035
图 34.3 配置停止位 ..... 1036
图 34.4 发送时 TC/TXE 的变化情况 ..... 1037
图 34.5 起始位侦测 ..... 1038
图 34.6 检测噪声的数据采样 ..... 1040
图 34.7 利用地址标记检测的静默模式 ..... 1043
图 34.8 LIN 模式下的断开检测（11 位断开长度 – 设置了 LBDL 位） ..... 1046
图 34.9 LIN 模式下的断开检测与帧错误的检测 ..... 1046
图 34.10 利用 DMA 发送 ..... 1048
图 34.11 利用 DMA 接收 ..... 1049
图 34.12 两个 UART 间的硬件流控制 ..... 1049
图 34.13 RTS 流控制 ..... 1050
图 34.14 CTS 流控制 ..... 1050
图 34.15 UART 中断映像图 ..... 1052

# 1 简介

本参考手册用于帮助开发者理解并使用睿兴科技（南京）有限公司（后文简称“睿兴”或“RX”）的芯片。本文提供了 RX32G410 系列微控制器的存储和外设的完整信息。

相关封装和电气特性请参考相应的数据手册。

本参考手册和数据手册均可从睿兴官网 www.rxtek-icore.com 获得。

# 2 文档约定

## 2.1 寄存器缩写列表

| | |
|---|---|
| read/write（rw） | 软件可以读取和写入这个位。 |
| read-only（r） | 软件只能读取这个位。 |
| write-only（w） | 软件只能写入这个位。读取这个位将返回复位值。 |
| read/clear write0 (rc_w0) | 软件可以读取这个位，并且可以通过写入 0 来清零这个位。写入 1 对这个位没有影响。 |
| read/clear write1 (rc_w1) | 软件可以读取这个位，并且可以通过写入 1 来清零这个位。写入 0 对这个位没有影响。 |
| read/set (rs) | 软件可以读取和设置这个位。写入 0 对这个位没有影响。 |
| read/write once (rwo) | 软件只能写入这个位一次，但可以在任何时候读取它。只有复位操作才能将这个位返回到其复位值。 |
| Reserved (Res.) | 保留位，必须保持在复位值。 |

## 2.2 词汇表

Quad world：四字，128 位长度数据。
Double world：双字，64 位长度数据。
Word：字，32 位长度数据。
Half-word：半字，16 位长度数据。
Byte：字节，8 位长度数据。

# 3 系统和存储器概述

## 3.1 系统架构

RX32G410 系列集成了带 FPU 的 32 位 RISC 内核，是电机专用的高性能微控制器。

- 多个驱动单元
  - 32 位 RISC 内核总线
  - DMA1
  - DMA2
- 五个被动单元
  - 内置 Flash
  - 内置 SRAM
  - 内置 CCM SRAM
  - AHB1 外设（含 AHB 至 APB 的桥及连接 APB1/APB2 总线的 APB 外设）
  - AHB2 外设

## 3.2 存储器结构

### 3.2.1 介绍

程序存储器，数据存储器，寄存器和 I/O 口统一编址在 4G 的线性地址空间里。

字节在存储器中以小端格式编码。一个字中编号最低的字节被认为是该字的最低有效字节，最高地址字节是最高有效字节。

### 3.2.2 存储器映像和寄存器边界地址

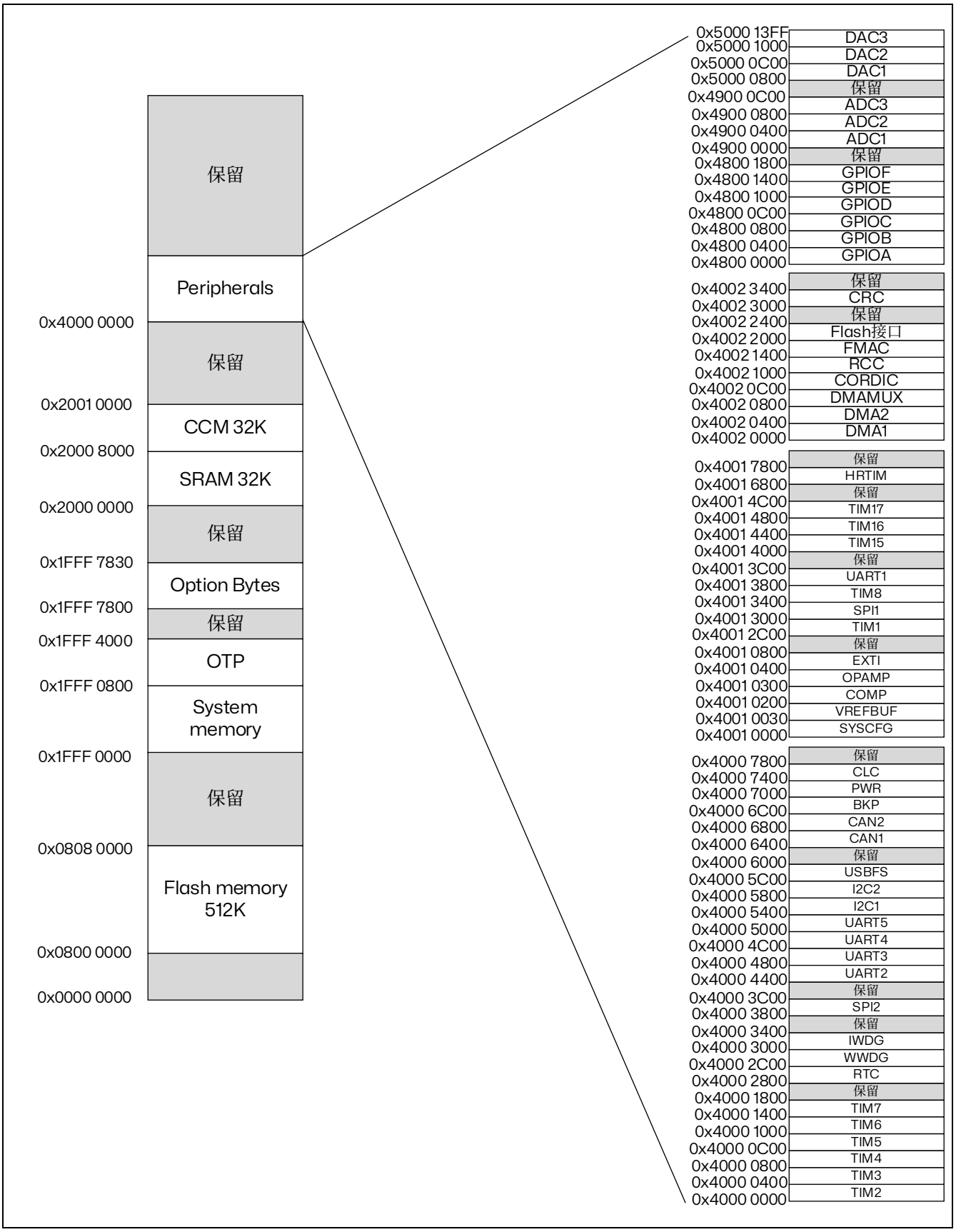



图 3.1 存储器映像

所有未分配给片上存储器和外围设备的内存映射区域都被配置为“Reserved”或者“保留”。有关可用内存和寄存器区域的详细映射，请参阅下表。

下表列出了设备中可用外设的边界地址。

表 3.1 RX32G410 系列存储器映像和外设边界地址

| 总线 | 边界地址 | 外设 |
|---|---|---|
| AHB2 | 0x5000 1000 - 0x5000 13FF | DAC3 |
| | 0x5000 0C00 - 0x5000 0FFF | DAC2 |
| | 0x5000 0800 - 0x5000 0BFF | DAC1 |
| | 0x5000 0C00 - 0x4900 07FF | 保留 |
| | 0x4900 0800 - 0x4900 0BFF | ADC3 |
| | 0x4900 0400 - 0x4900 07FF | ADC2 |
| | 0x4900 0000 - 0x4900 03FF | ADC1 |
| | 0x4800 1800 - 0x48FF FFFF | 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 - 0x47FF FFFF | 保留 |
| | 0x4002 3000 - 0x4002 33FF | CRC |
| | 0x4002 2400 - 0x4002 2FFF | 保留 |
| | 0x4002 2000 - 0x4002 23FF | Flash 接口 |
| | 0x4002 1400 - 0x4002 1FFF | FMAC |
| | 0x4002 1000 - 0x4002 13FF | RCC |
| | 0x4002 0C00 - 0x4002 0FFF | CORDIC |
| | 0x4002 0800 - 0x4002 0BFF | DMAMUX |
| | 0x4002 0400 - 0x4002 07FF | DMA2 |
| | 0x4002 0000 - 0x4002 03FF | DMA1 |
| APB2 | 0x4001 7800 - 0x4001 FFFF | 保留 |
| | 0x4001 6800 - 0x4001 77FF | HRTIM |
| | 0x4001 4C00 - 0x4001 67FF | 保留 |
| | 0x4001 4800 - 0x4001 4BFF | TIM17 |
| | 0x4001 4400 - 0x4001 47FF | TIM16 |
| | 0x4001 4000 - 0x4001 43FF | TIM15 |
| | 0x4001 3C00 - 0x4001 3FFF | 保留 |
| | 0x4001 3800 - 0x4001 3BFF | UART1 |
| | 0x4001 3400 - 0x4001 37FF | TIM8 |
| | 0x4001 3000 - 0x4001 33FF | SPI1 |
| | 0x4001 2C00 - 0x4001 2FFF | TIM1 |
| | 0x4001 0800 - 0x4001 2BFF | 保留 |

| 总线 | 边界地址 | 外设 |
|---|---|---|
| | 0x4001 0400 - 0x4001 07FF | EXTI |
| | 0x4001 0300 - 0x4001 03FF | OPAMP |
| | 0x4001 0200 - 0x4001 02FF | COMP |
| | 0x4001 0030 - 0x4001 01FF | VREFBUF |
| | 0x4001 0000 - 0x4001 0029 | SYSCFG |
| APB1 | 0x4000 7800 - 0x4000 FFFF | 保留 |
| | 0x4000 7400 - 0x4000 77FF | CLC |
| | 0x4000 7000 - 0x4000 73FF | PWR |
| | 0x4000 6C00 - 0x4000 6FFF | BKP |
| | 0x4000 6800 - 0x4000 6BFF | CAN2 |
| | 0x4000 6400 - 0x4000 67FF | CAN1 |
| | 0x4000 6000 - 0x4000 63FF | 保留 |
| | 0x4000 5C00 - 0x4000 5FFF | USBFS |
| | 0x4000 5800 - 0x4000 5BFF | I2C2 |
| | 0x4000 5400 - 0x4000 57FF | I2C1 |
| | 0x4000 5000 - 0x4000 53FF | UART5 |
| | 0x4000 4C00 - 0x4000 4FFF | UART4 |
| | 0x4000 4800 - 0x4000 4BFF | UART3 |
| | 0x4000 4400 - 0x4000 47FF | UART2 |
| | 0x4000 3C00 - 0x4000 43FF | 保留 |
| | 0x4000 3800 - 0x4000 3BFF | SPI2 |
| | 0x4000 3400 - 0x4000 37FF | 保留 |
| | 0x4000 3000 - 0x4000 33FF | IWDG |
| | 0x4000 2C00 - 0x4000 2FFF | WWDG |
| | 0x4000 2800 - 0x4000 2BFF | RTC |
| | 0x4000 1800 - 0x4000 27FF | 保留 |
| | 0x4000 1400 - 0x4000 17FF | TIM7 |
| | 0x4000 1000 - 0x4000 13FF | TIM6 |
| | 0x4000 0C00 - 0x4000 0FFF | TIM5 |
| | 0x4000 0800 - 0x4000 0BFF | TIM4 |
| | 0x4000 0400 - 0x4000 07FF | TIM3 |
| | 0x4000 0000 - 0x4000 03FF | TIM2 |

## 3.3 内置 SRAM

RX32G410 系列内置总计 64KB 的 SRAM。

- 32KB 的 SRAM（映射在 0x2000 0000）
- 32KB 的 CCM SRAM（映射在 0x1000 0000 和 SRAM 的结尾处）

这些 SRAM 可以以字节（8 位）、半字节（16 字节）或字（32 位）进行访问。这些存储器可以在没有等待周期的情况下由 CPU 或者 DMA 寻址，需要特别注意的是，映射在 0x1000 0000 的 CCM SRAM 不可以通过 DMA 寻址。

*注：不建议将代码和数据一起放在*CCM SRAM *中，这样会造成效率降低。*

### 3.3.1 奇偶校验

整个 CCM SRAM 区域具有奇偶校验功能，SRAM 区域不具备。

可通过 FLASH 选项字节寄存器（FLASH_OPTR）SRAM_PE 位来使用 CCM SRAM 奇偶校验功能。

在写入 CCM SRAM 时，将计算并存储奇偶校验位，在读取时会自动检查；当某一位发生错误时，则产生一个非屏蔽中断。CCM SRAM 的奇偶校验错误相关标志位参考 SYSCFG_CFGR2 寄存器的 CFGR2_SPF 和 CFGR2_SPF2。

*注：在使能*CCM SRAM *的奇偶校验功能之前，建议先将整个*CCM SRAM *区域进行软件初始化，避免在读取未初始化的区域时出现奇偶校验错误，或者保证在读取*CMM SRAM *的某块区域之前已经对该区域进行过擦除或者写入操作。*

### 3.3.2 CCM SRAM 读保护

CCM SRAM 的读保护通过设置 RDP 选项字节来激活的。

### 3.3.3 CCM SRAM 擦除

使用 FLASH_CR 的 CCM_RST 位，可以在系统复位的时候擦除 CCM SRAM。

也可以使用 SYSCFG_SKR 寄存器的 KEY[7:0]解除软件擦写保护，然后使用 SCSR_CCMER 启动擦除。

## 3.4 Flash 概述

Flash 有两个不同的物理区域组成：

- 主 Flash 块。用于存储应用代码和数据。
- 信息块，主要包含三个部分：
  - 选项字节：用于硬件和存储器保护。
  - 系统存储器：包含 RX 专用代码。
  - OTP：一次性编程区域

Flash 接口基于 AHB 协议进行指令和数据访问。Flash 寄存器具有 Flash 的操作（擦除/写入）功能。

## 3.5 Boot 配置

RX32G410 的 BOOT 配置可以通过 BOOT0 引脚和选项字节组合配置，如下表所示：

表 3.2 Boot 模式

| nBOOT1<br>FLASH_OPTR[23] | nBOOT0<br>FLASH_OPTR[27] | BOOT0<br>Pin PB8 | nSWBOOT0<br>FLASH_OPTR[26] | Boot 区域 |
|---|---|---|---|---|
| x | x | 0 | 1 | 主 Flash 块 |
| x | 1 | x | 0 | 主 Flash 块 |
| 0 | x | 1 | 1 | SRAM |
| 0 | 0 | x | 0 | SRAM |
| 1 | x | 1 | 1 | System 块 |
| 1 | 0 | x | 0 | System 块 |

**内置 Bootloader**

内置 Bootloader 代码位于系统存储器（映射在 0x1FFF 0000）中，用于以下串口接口重新编程 Flash：

- 引脚 PA9/PA10 上的 UART1。

# 4 内置 Flash 存储器（FLASH）

## 4.1 介绍

Flash 接口管理 CPU AHB ICode 和 Dcode 对 Flash 进行访问，具有 Flash 擦除和编程功能以及读写保护机制。

Flash 接口通过一个包含指令预取和缓存系统来加速代码的执行。

## 4.2 Flash 主要特征

- 总计 512 KB 的 Flash，128 bit 位宽，双 bank 支持同时读写操作
- 支持误码矫正（ECC）
- 支持字、半字和字节读取操作
- 支持 128 位进行 Flash 编程
- 指令预取：128 bit
- 指令缓存：64×128 bit 的 ICode 缓存线（1 KB RAM）
- 指令缓存：16×128 bit 的 DCode 缓存线（256 B RAM）
- 支持页擦、存储区擦除和全擦
- 读保护
- 写保护
- 选项字节加载器

## 4.3 Flash 功能描述

### 4.3.1 Flash 结构

Flash 包含 512 页（每页 1KB）的主存储块和信息块，如下表所示。

表 4.1 Flash 结构（未开启 ECC）

| Flash 区域 | | 名称 | 地址 | 大小(字节) |
|---|---|---|---|---|
| 主存储块 | 存储区 1<br>(Bank1：256 KB) | 页 0 | 0x0800 0000 - 0x0800 03FF | 1K |
| | | 页 1 | 0x0800 0400 - 0x0800 07FF | 1K |
| | | … | … | … |
| | | 页 255 | 0x0803 FC00 - 0x0803 FFFF | 1K |
| | 存储区 2<br>(Bank2：256 KB) | 页 256 | 0x0804 0000 - 0x0804 03FF | 1K |
| | | 页 257 | 0x0804 0400 - 0x0804 07FF | 1K |
| | | … | … | … |
| | | 页 511 | 0x0807 FC00 - 0x0807 FFFF | 1K |
| 信息块 | - | 系统存储器 | 0x1FFF 0000 - 0x1FFF 07FF | 2K |
| | - | OTP | 0x1FFF 0800 - 0x1FFF 3FFF | 14K |
| | - | 选项字节 | 0x1FFF 7800 - 0x1FFF 782F | 48 |

表 4.2 Flash 结构（开启 ECC）

| Flash 区域 | | 名称 | 地址 | 大小(字节) |
|---|---|---|---|---|
| 主存储块 | 存储区 1 (Bank1：128 KB) | 页 0 | 0x0800 0000 - 0x0800 01FF | 0.5K |
| | | 页 1 | 0x0800 0200 - 0x0800 03FF | 0.5K |
| | | … | … | … |
| | | 页 255 | 0x0801 FE00 - 0x0801 FFFF | 0.5K |
| | 存储区 2 (Bank2：128 KB) | 页 256 | 0x0802 0000 - 0x0802 01FF | 0.5K |
| | | 页 257 | 0x0802 0200 - 0x0802 03FF | 0.5K |
| | | … | … | … |
| | | 页 511 | 0x0803 FE00 - 0x0803 FFFF | 0.5K |
| 信息块 | - | 系统存储器 | 0x1FFF 0000 - 0x1FFF 07FF | 2K |
| | - | OTP | 0x1FFF 0800 - 0x1FFF 3FFF | 14K |
| | - | 选项字节 | 0x1FFF 7800 - 0x1FFF 782F | 48 |

### 4.3.2 双存储区支持同时操作

Flash 分为两个独立的存储区。对存储区 1 进行擦写操作的时候，可对存储区 2 进行读取操作。

### 4.3.3 误码矫正功能（ECC）

Flash 支持：

- 单错误校正
- 双重错误检测

检测到一个错误并校正后，FLASH_ECCR 寄存器中的 ECCC 会置 1。如果 ECCCIE 被置 1，将产生中断。

检测到两个错误后，FLASH_ECCR 寄存器中的 ECCD 会置 1，将会产生非屏蔽中断。

检测到错误后，发生错误地址将会被保存在 FLASH_ECCR 寄存器中的 ADDR_ECC[20:0]中。FLASH_ECCR 寄存器只会在 ECC 标志位（ECCC 和 ECCD）为 0 时更新。

### 4.3.4 读取等待周期

为了正确读取 Flash 的指令和数据，必须根据 CPU 的时钟（HCLK）在 Flash 访问控制寄存器（FLASH_ACR）中配置等待周期（Latency）。等待周期和 CPU 时钟频率的关系如下表所示。

表 4.3 等待周期和 CPU 时钟频率对应关系

| 等待周期（WS）（Latency） | HCLK(MHz) |
|---|---|
| 0 WS（1 CPU cycles） | ≤38 |
| 1 WS（2 CPU cycles） | ≤76 |
| 2 WS（3 CPU cycles） | ≤115 |
| 3 WS（4 CPU cycles） | ≤153 |
| 4 WS（5 CPU cycles） | ≤192 |

复位后，CPU 时钟频率是 16MHz，FLASH_ACR 中配置为 1 等待周期。

修改 CPU 频率时，必须执行以下步骤以保证足够的 Flash 访问等待周期：

**增加 CPU 频率：**

1. 修改 Flash 访问控制寄存器（FLASH_ACR）中 Latency 的等待周期。
2. 读取并确认 FLASH_ACR 寄存器中的等待周期。
3. 修改 CPU 频率。
4. 确认 CPU 频率。

**降低 CPU 频率：**

1. 修改 CPU 频率。
2. 确认 CPU 频率。
3. 修改 Flash 访问控制寄存器（FLASH_ACR）中 Latency 的等待周期。
4. 读取并确认 FLASH_ACR 寄存器中的等待周期。

### 4.3.5 指令预取

带 FPU 的 32 位 RISC 内核通过总线取指令和数据。预取缓冲区可以提高总线的访问效率。

预取缓冲区（128 bit）：CPU 每次取指最多为 32 位的字，取一条指令时，下一条指令已经在缓冲区中等待，这样 CPU 可以工作在更高的主频。

复位后预取缓冲区默认关闭。只有在 SYSCLK 小于 40MHz 且 AHB 没有预分频）（SYSCLK 必须等于 HCLK）时才可以开启或关闭预取缓冲区。通常情况下，建议在芯片初始化阶开启或关闭缓冲区，此时芯片运行在内部 16MHz 晶振上。

### 4.3.6 Flash 擦除和编程

Flash 支持在线编程和应用编程。

在线编程（in-circuit programming，ICP）是指使用 JTAG、串行调试接口或者 bootloader 更新 Flash 的内容。

应用编程（in-application programming，IAP）是指使用芯片支持的通讯协议更新 Flash 的内容。IAP 允许用户在芯片运行过程中更新 Flash 的内容。然而要实现 IAP，需要预先使用 ICP 将部分应用代码编写进 Flash 中。

如果在 Flash 操作期间复位芯片将无法保证 Flash 的内容正确。

### 4.3.7 解锁 Flash

复位后，Flash 控制寄存器（Flash control register，FLASH_CR）不可写入，这是为了防止 Flash 由于干扰产生误操作。通过以下流程可以解锁 FLASH_CR：

1. 在 Flash 密钥寄存器（Flash key register，FLASH_FKEYR）写入 KEY1 = 0x45670123。
2. 在 Flash 密钥寄存器（Flash key register，FLASH_FKEYR）写入 KEY2 = 0xCDEF89AB。

任何错误的流程将会导致 FLASH_CR 上锁直到系统复位。一旦执行了错误的写入密钥流程，将会产生一个总线错误和一个硬件错误中断。

FLASH_CR 可以通过写入其中的 LOCK 位来上锁。

*注：FLASH_CR 寄存器在 Flash 状态寄存器（Flash status register，FLASH_SR）的 BSY 位置起时不可被写入。在 BSY 位置起时，任何写入动作将会导致 AHB 总线挂起直到 BSY 位被清除。*

### 4.3.8 Flash 擦除流程

Flash 擦除可以分为页擦和全擦。全擦不会擦除信息区（系统存储器和选项字节）。

**页擦**

页擦流程如下：

1. 确认 FLASH_SR 寄存器的 BSY 位以保证没有正在执行的 Flash 操作。
2. 检查并清除之前的 Flash 操作造成的错误。
3. 将 FLASH_CR 的 PER 位置 1，配置 PNB 选择要擦除的页。
4. 将 FLASH_CR 的 STRT 位置 1。
5. 等 FLASH_SR 中 BSY 被清除。

**存储区全擦**

存储区 1 或存储区 2 全擦流程如下：

1. 确认 FLASH_SR 寄存器的 BSY 位以保证没有正在执行的 Flash 操作。
2. 检查并清除之前的 Flash 操作造成的错误。
3. 将 FLASH_CR 的 MER1 或 MER2 位置 1。
4. 将 FLASH_CR 的 STRT 位置 1。
5. 等 FLASH_SR 中 BSY 被清除。

**全擦**

全擦流程如下：

1. 确认 FLASH_SR 寄存器的 BSY 位以保证没有正在执行的 Flash 操作。
2. 检查并清除之前的 Flash 操作造成的错误。
3. 将 FLASH_CR 的 MER1 和 MER2 位置 1。
4. 将 FLASH_CR 的 STRT 位置 1。
5. 等 FLASH_SR 中 BSY 被清除。

### 4.3.9 FLASH 写入流程

主 FLASH 一次写入一个 128 bit 的数据。写入流程如下：

1. 确认 FLASH_SR 的 BSY 位以保证没有正在执行的 Flash 操作。
2. 将 FLASH_CR 的 PG 位置 1。
3. 向目标地址写入数据（128bit）。
4. 等待 FLASH_SR 的 BSY 位清零。
5. 读取并验证写入的数据。

## 4.4 保护

闪存中的用户代码区可以防止非法的读出，同样可以对闪存区的页加以保护，防止在程序跑飞的情况下不被意外地改变，写保护的基本单位是 1KB。

### 4.4.1 读保护

读保护是通过设置 RDP 选项字节，然后在系统重新复位加载了新的 RDP 选项字节后启动的。当保护字节被写入相应的值后，只允许 Flash 中的代码对 Flash 进行读操作。

**解除读保护**

1. 擦除整个选项字节区域，读保护码（RDP）将会变为 0xFF，此时读保护仍然有效。
2. 写入正确的 RDP 代码 0xAA 以解除 Flash 的保护，该操作会导致整块 Flash 被擦除。
3. 进行系统复位以重新加载选项字节，读保护被解除。

### 4.4.2 写保护

RX32G410 系列的写保护以 1KB 为单位。如果在已经开启了写保护的区域进行写入或者擦除操作，FLASH_SR 中的 WRPERR 将会置起。

写保护是通过设置 FLASH_WRP1AR、FLASH_WRP1BR，然后在系统重新复位加载了新的 FLASH_WRP1AR、FLASH_WRP1BR 选项字节后启动的。

**解除写保护**

1. 擦除整个选项字节区域。
2. 写入正确的 FLASH_WRP1AR、FLASH_WRP1BR 代码。
3. 进行系统复位以重新加载选项字节，写保护被解除。

*注：全擦选项字节会将* RDP *擦除为* 0xFF，*系统重启后会启动读保护，因此需要将* RDP *进行修改。*

## 4.5 选项字节

### 4.5.1 选项字节描述

选项字节一共有 48 字节，选项字节可以由终端用户根据应用情况进行配置。

选项字节中，一个 64 位的字被做如下划分。

表 4.4 选项字节格式

| 63-56 | 55-48 | 47-40 | 39-32 | 31-24 | 23-16 | 15-8 | 7-0 |
|---|---|---|---|---|---|---|---|
| 选项字节 3 | 选项字节 2 | 选项字节 1 | 选项字节 0 | 选项字节 3 | 选项字节 2 | 选项字节 1 | 选项字节 0 |

选项字节的内容可以通过直接访问选项字节所在地址获取，也可以通过访问选项字节寄存器（FLASH_OPTR）获取。

*注：选项字节的新配置在系统重启后才会被加载。*

### 4.5.2 选项字节擦除

选项字节擦除由 FPEC 在选项字节写入过程中自动执行。

### 4.5.3 选项字节写入

选项字节写入流程如下：

1. 对 FPEC 解锁。
2. 对 FLASH_OPTKEYR 写入 KEY1 = 0x0819 2A3B。
3. 对 FLASH_OPTKEYR 写入 KEY2 = 0x4C5D 6E7F。
4. 检查 FLASH_SR 的 BSY 位，以确认没有其他正在进行的 Flash 操作。
5. 写选项值至 FLASH_OPTR/FLASH_WRP1AR/ FLASH_WRP1BR。
6. 将 FLASH_CR 寄存器的 OPTSTRT 位置 1，启动选项字节编程。
7. 等待 FLASH_SR 的 BSY 位清零。
8. 读取并验证写入的半字。

### 4.5.4 选项字节组织

| Address | [63:56] | [55:48] | [47:40] | [39:32] | [31:24] | [23:16] | [15:8] | [7:0] |
|---|---|---|---|---|---|---|---|---|
| 1FFF7800 | 用户选项 | | | 读保护 | 用户选项 | | | 读保护 |
| 1FFF7808 | 保留 | | | | | | | |
| 1FFF7810 | 保留 | | | | | | | |
| 1FFF7818 | 保留 | 写保护 A 区末端偏移 | 保留 | 写保护 A 区开始偏移 | 保留 | 写保护 A 区末端偏移 | 保留 | 写保护 A 区开始偏移 |
| 1FFF7820 | 保留 | 写保护 B 区末端偏移 | 保留 | 写保护 B 区开始偏移 | 保留 | 写保护 B 区末端偏移 | 保留 | 写保护 B 区开始偏移 |

### 4.5.5 用户和读保护选项字节

复位值：0x184F 30AA

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | nBOOT0 | nSWBOOT0 | Reserved | | nBOOT1 | Res. | ECC | Res. | WWDG_SW | IWDG_STDBY | IWDG_STOP | IWDG_SW |
| | | | | rw | rw | | | rw | | rw | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | nRST_STDBY | nRST_STOP | Reserved | | | | RDP[7:0] | | | | | | | |
| | | rw | rw | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:28 保留，必须保持为复位值。

Bit 27 **nBOOT0**：BOOT0 选择位

0：Boot0=0

1：Boot0=1

Bit 26 **nSWBOOT0**：选择 BOOT0 模式

0：Boot0 取决于 nBOOT0 位

1：Boot0 取决于 PB8/BOOT0 引脚

Bit 25:24 保留，必须保持为复位值。

Bit 23 **nBOOT1**：引导配置

0：nBOOT1=0

1：nBOOT1=1

Bit 22 保留，必须保持为复位值。

Bit 21 **ECC**：主 Flash ECC

0：禁止 ECC

1：开启 ECC

Bit 20 保留，必须保持为复位值。

Bit 19 **WWDG_SW**：窗口看门狗选择

0：硬件窗口看门狗

1：软件窗口看门狗

Bit 18 **IWDG_STDBY**：待机模式时独立看门狗计数冻结

0：进入待机模式时独立看门狗计数冻结。

1：进入待机模式时独立看门狗计数运行。

Bit 17 **IWDG_STOP**：停机模式时独立看门狗计数冻结

0：进入停机模式时独立看门狗计数冻结。

1：进入停机模式时独立看门狗计数运行。

Bit 16 **IWDG_SW**：独立看门狗选择

0：硬件独立看门狗

1：软件独立看门狗

Bit 15:14 保留，必须保持为复位值。

Bit 13 **nRST_STDBY**：

0：当进入待机模式时产生复位

1：当进入待机模式时不复位

Bit 12 **nRST_STOP**：

0：当进入停机模式时产生复位

1：当进入停机模式时不复位

Bit 11:8 保留，必须保持为复位值。

Bit 7:0 **RDP[7:0]**：读保护级别

0xAA：级别为 0，读保护未激活。

0xCC：级别为 2，芯片读保护激活。

其他：级别为 1，存储器读保护激活。

### 4.5.6 写保护 A 区地址选项字节

地址：0x1FFF 7818

复位值：0x0000 01FF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1A_END[8:0] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | WRP1A_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |

Bit 31:25 保留，必须保持为复位值。

Bit 24:16 **WRP1A_END[8:0]**：WRP 第一区域“A”区末端地址偏移（单位：1K 字节）
包含 WRP 第一个区域的最后一个页面。

Bit 15:9 保留，必须保持为复位值。

Bit 8:0 **WRP1A_STRT[8:0]**：WRP 第一区域“A”区起始地址偏移（单位：1K 字节）
包含 WRP 第一个区域的第一个页面。

### 4.5.7 写保护 B 区地址选项字节

地址：0x1FFF 7820

复位值：0x0000 01FF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1B_END[8:0] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | WRP1B_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |

Bit 31:25 保留，必须保持为复位值。

Bit 24:16 **WRP1B_END[8:0]**：WRP 第二区域“B”区末端地址偏移（单位：1K 字节）
包含 WRP 第二个区域的最后一个页面。

Bit 15:9 保留，必须保持为复位值。

Bit 8:0 **WRP1B_STRT[8:0]**：WRP 第二区域“B”区起始地址偏移（单位：1K 字节）
包含 WRP 第二个区域的第一个页面。

## 4.6 Flash 寄存器描述

### 4.6.1 Flash 访问控制寄存器（FLASH_ACR）

地址偏移：0x00
复位值：0x0000 0601
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DCRST | ICRST | DCEN | ICEN | PRFTEN | Reserved | | | | LATENCY[3:0] | | | |
| | | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bit 31:13 保留，必须保持为复位值。

Bit 12 **DCRST**：数据缓存复位
0：不复位数据缓存
1：复位数据缓存
仅在 DCEN=0 时可写。

Bit 11 **ICRST**：指令缓存复位
0：不复位指令缓存
1：复位指令缓存
仅在 ICEN=0 时可写。

Bit 10 **DCEN**：数据缓存使能
0：关闭数据缓存
1：使能数据缓存

Bit 9 **ICEN**：指令缓存使能
0：关闭指令缓存
1：使能指令缓存

Bit 8 **PRFTEN**：预取缓冲区使能
0：关闭预取缓冲区
1：使能预取缓冲区

Bit 7:4 保留，必须保持为复位值。

Bit 3:0 **LATENCY[3:0]**：等待周期
该位表示 SYSCLK（系统时钟）周期与 Flash 访问时间的比例。
000：零等待周期
001：一个等待周期
010：两个等待周期
011：三个等待周期
100：四个等待周期
…

### 4.6.2 Flash 键寄存器（FLASH_KEYR）

地址偏移：0x08
复位值：0x0000 0000
访问：无等待，支持字访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEYR[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 |
| KEYR[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **KEYR**：Flash 键值

KEY1：0x4567 0123
KEY2：0xCDEF 89AB

### 4.6.3 Flash OPTKEY 寄存器（FLASH_OPTKEYR）

地址偏移：0x0C
复位值：0x0000 0000
访问：无等待，支持字访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPTKEYR[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 |
| OPTKEYR[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **OPTKEYR**：选项字节键值

KEY1：0x0819 2A3B
KEY2：0x4C5D 6E7F

### 4.6.4 Flash 状态寄存器（FLASH_SR）

地址偏移：0x10
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | BSY |
| | | | | | | | | | | | | | | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OPTV ERR | Reserved | | | | | | | PGS ERR | SIZE ERR | PGA ERR | WRP ERR | PROG ERR | Res | OPE RR | EOP |
| rc_w1 | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | | rc_w1 | rc_w1 |

Bit 31:17 保留，必须保持为 0。

Bit 16 **BSY**：忙碌

表示正在执行一个 Flash 操作。在 Flash 操作开始时该位由硬件置 1，当操作结束或者发生错误时该位由硬件清零。

Bit 15 **OPTVERR**：选项有效性错误

当选项字节没有被正确加载时，该位被硬件置位。

写 1 清零。

Bit 14:8 保留，必须保持为复位值。

Bit 7 **PGSERR**：编程顺序错误

写入闪存时发生顺序错误由硬件置位，或者当 PROGERR、SIZERR、PGAERR、WRPERR 因之前的写入错误而置位时该位由硬件置位。

写 1 清零。

Bit 6 **SIZERR**：字节大小错误

仅允许四字编程，否则该位由硬件置位。

写 1 清零。

Bit 5 **PGAERR**：写入对齐错误

待编程的数据不能被包含在同一个 128 位闪存行中，由硬件置位。

写 1 清零。

Bit 4 **WRPERR**：写保护错误

写入开启了写保护的 Flash 区域时由硬件置位。

写 1 清零。

Bit 3 **PROGERR**：写入错误

当一个四字地址被编程时，写入值不是“0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF”的 Flash 区域时由硬件置位。

写 1 清零。

Bit 2 保留，必须保持为 0。

Bit 1 **OPERR**：操作错误

Flash 区域操作（编程/擦除）未成功完成时，由硬件置位。

写 1 清零。

Bit 0 **EOP**：操作完成

一个或多个 Flash 操作（写入/擦除）成功完成时，由硬件置位。

写 1 清零。

*注：*EOP *只有在*Flash *成功写入或者擦除结束时生效。*

### 4.6.5 Flash 控制寄存器（FLASH_CR）

地址偏移：0x14

复位值：0xC000 0000

访问：当未执行 Flash 内存操作时，访问无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | OPT LOCK | Reserved | | OBL_LAUNCH | Res | ERRIE | EOPIE | Reserved | | | | | | OPT STRT | STRT |
| rs | rs | | | rc_w1 | | rw | rw | | | | | | | rs | rs |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MER2 | Reserved | | | BKER | PNB[7:0] | | | | | | | | MER1 | PER | PG |
| rw | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **LOCK**：锁存位

软件写 1 有效。当该位置 1 时，FPEC 和 FLASH_CR 被锁住。检测到正确的解锁序列后，硬件清零此位。

在一次解锁不成功后，下次系统复位前，该位一直保持置位。

Bit 30 **OPTLOCK**：选项字节锁存位

当该位置位时，FLASH_CR 寄存器中与选项字节有关的所有位和选项页都被锁定。硬件检测到解锁顺序后，清除该位。在对 OPTLOCK 位进行解锁顺序之前，必须先清除 LOCK 位。

在解锁操作不成功的情况下，该位始终保持设置，直到下一次重置。

Bit 29:28 保留，必须保持为复位值。

Bit 27 **OBL_LAUNCH**：强制加载选项字节

当设置 1 时，强制重新加载选项字节。只有选项字节加载完成，该位才被清除。

如果 OPTLOCK 被设置，则该位不能被写入。

0：选项字节加载完成

1：请求加载选项字节

Bit 26 保留，必须保持为复位值。

Bit 25 **ERRIE**：错误中断使能

该位允许在 FLASH_SR 中的 PGERR/WRPERR 被置位时产生中断。

0：关闭 ERR 中断

1：使能 ERR 中断

Bit 24 **EOPIE**：操作结束中断使能

该位允许在 FLASH_SR 中的 EOP 被置位时产生中断。

0：关闭 EOP 中断

1：使能 EOP 中断

Bit 23:18 保留，必须保持为复位值。

Bit 17 **OPTSTRT**：选项修改开始

该位允许修改选项。该位仅由软件设置，在 BSY 置 1 时由硬件清零。

Bit 16 **STRT**：开始

软件写 1 后触发一次擦除操作。在 BSY 置 1 时由硬件清零。

Bit 15 **MER2**：BANK2 全擦除操作

该位置 1 可使能对存储区 2 的全擦操作。

Bit 14:12 保留，必须保持为复位值。

Bit 11 **BKER**：BANK 擦除

该位选择进行页擦的存储区

0：存储区 1 参与页擦

1：存储区 2 参与页擦

Bit 10:3 **PNB[7:0]**：擦除页码选择

00000000：page 0

00000001：page 1

00000010：page 2

00000011：page 3

....

11111111：page 255

Bit 2 **MER1**：BANK1 全擦除操作

该位置 1 可使能对存储区 1 的全擦操作。

Bit 1 **PER**：页擦除操作

该位置 1 可使能对主 Flash 的页擦操作。

Bit 0 **PG**：编程操作

该位置 1 可使能对主 Flash 的编程操作。

### 4.6.6 Flash ECC 寄存器（FLASH_ECCR）

地址偏移：0x18

复位值：0x0000 0000

访问：当未执行 Flash 内存操作时，访问无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCD | ECCC | ECCD2 | ECCC2 | Reserved | | | ECCCIE | Reserved | | BK_ECC | ADDR_ECC[20:16] | | | | |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | | | | rw | | | r | r | 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 31 **ECCD**：ECC 检测

当在 64 bit LSB 检测到两个 ECC 错误时，由硬件置位。当该位置起时，将产生一个非屏蔽中断。

写 1 清除。

Bit 30 **ECCC**：ECC 校正

当在 64 bit LSB 检测到一个 ECC 错误并纠正后，有硬件置位。如果设置了 ECCCIE,则会产生中断。

写 1 清除。

Bit 29 **ECCD2**：ECC 检测 2

当在 64 bit MSB 检测到两个 ECC 错误时，由硬件置位。当该位置起时，将产生一个非屏蔽中断。

写 1 清除。

Bit 28 **ECCC2**：ECC2 校正

当在 64 bit MSB 检测到一个 ECC 错误并纠正后，有硬件置位。如果设置了 ECCCIE，则会产生中断。

写 1 清除。

Bit 27:25 保留，必须保持为复位值。

Bit 24 **ECCCIE**：ECC 纠错中断使能

0：关闭 ECCC 中断

1：开启 ECCC 中断

Bit 23:22 保留，必须保持为复位值。

Bit 21 **BK_ECC**：ECC 错误发生区域

该位表明 ECC 纠错或双 ECC 纠错所发生的 BANK.

0：BANK1

1：BANK2

Bit 20:0 **ADDR_ECC[20:0]**：ECC 错误地址

该位表明 Bank 中出现 ECC 错误或双 ECC 错误的地址。

### 4.6.7 Flash 选项字节寄存器（FLASH_OPTR）

地址偏移：0x20

复位值：0xFxxx xxxx 该寄存器从 Flash OBL 中加载。

访问：当未执行 Flash 内存操作时，访问无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | n BOOT0 | nSW BOOT0 | CCMSR AM_RST | SRAM_ PE | nBOOT1 | Res. | ECC | Res. | WWDG_ SW | IWDG_ STDBY | IWDG_ STOP | IWDG_ SW |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | nRST_ STDBY | nRST_ STOP | Reserved | | | | RDP[7:0] | | | | | | | |
| | | rw | rw | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:28 保留，必须保持为复位值。

Bit 27 **nBOOT0**：BOOT0 选择位

0：Boot0=0

1：Boot0=1

Bit 26 **nSWBOOT0**：选择 BOOT0 模式

0：Boot0 取决于 nBOOT0 位

1：Boot0 取决于 PB8/BOOT0 引脚

Bit 25 **CCMSRAM_RST**：当系统复位时擦除 CCM SRAM

0：当系统复位产生时 CCM SRAM 被擦除

1：当系统复位产生时 CCM SRAM 不会被擦除

Bit 24 **SRAM_PE**：CCM SRAM 校验使能

0：使能 CCM SRAM 校验

1：禁止 CCM SRAM 校验

Bit 23 **nBOOT1**：引导配置

0：nBOOT1=0

1：nBOOT1=1

Bit 22 保留，必须保持为复位值。

Bit 21 **ECC**：主 Flash ECC

0：禁止 ECC

1：开启 ECC

Bit 20 保留，必须保持为复位值。

Bit 19 **WWDG_SW**：窗口看门狗选择

0：硬件窗口看门狗

1：软件窗口看门狗

Bit 18 **IWDG_STDBY**：待机模式时独立看门狗计数冻结

0：进入待机模式时独立看门狗计数冻结。

1：进入待机模式时独立看门狗计数运行。

Bit 17 **IWDG_STOP**：停机模式时独立看门狗计数冻结

0：进入停机模式时独立看门狗计数冻结。

1：进入停机模式时独立看门狗计数运行。

Bit 16 **IWDG_SW**：独立看门狗选择

0：硬件独立看门狗

1：软件独立看门狗

Bit 15:14 保留，必须保持为复位值。

Bit 13 **nRST_STDBY**：

0：当进入待机模式时产生复位

1：当进入待机模式时不复位

Bit 12 **nRST_STOP**：

0：当进入停机模式时产生复位

1：当进入停机模式时不复位

Bit 11:8 保留，必须保持为复位值。

Bit 7:0 **RDP[7:0]**：读保护级别

0xAA：级别为 0，读保护未激活。

0xCC：级别为 2，芯片读保护激活。

其他：级别为 1，存储器读保护激活。

### 4.6.8 Flash 写保护 A 区地址寄存器（FLASH_WRP1AR）

地址偏移：0x2C
复位值：0x0000 01FF
寄存器位在选项字节加载（OBL）时从 Flash 内存中加载值
访问：当未执行 Flash 内存操作时，访问无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1A_END[8:0] | | | | | | | | |
| | | | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1A_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:25 保留，必须保持为复位值。

Bit 24:16 **WRP1A_END[8:0]：**WRP 第一区域“A”区末端地址偏移（单位：1K 字节）
包含 WRP 第一个区域的最后一个页面。

Bit 15:9 保留，必须保持为复位值。

Bit 8:0 **WRP1A_STRT[8:0]：**WRP 第一区域“A”区起始地址偏移（单位：1K 字节）
包含 WRP 第一个区域的第一个页面。

### 4.6.9 Flash 写保护 B 区地址寄存器（FLASH_WRP1BR）

地址偏移：0x30
复位值：0x0000 01FF
寄存器位在选项字节加载（OBL）时从 Flash 内存中加载值
访问：当未执行 Flash 内存操作时，访问无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1B_END[8:0] | | | | | | | | |
| | | | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | WRP1B_STRT[8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:25 保留，必须保持为复位值。

Bit 24:16 **WRP1B_END[8:0]：**WRP 第二区域“B”区末端地址偏移（单位：1K 字节）
包含 WRP 第二个区域的最后一个页面。

Bit 15:9 保留，必须保持为复位值。

Bit 8:0 **WRP1B_STRT[8:0]：**WRP 第二区域“B”区起始地址偏移（单位：1K 字节）
包含 WRP 第二个区域的第一个页面。

# 5 循环冗余校验计算单元（CRC）

## 5.1 CRC 简介

CRC（循环冗余校验）计算单元使用一个固定的多项式发生器从一个 32 位的数据字中产生 CRC 码。在众多的应用中，基于 CRC 的技术还常用来验证数据传输或存储的完整性。根据 EN/IEC 60335-1 标准的规定，这些技术提供了验证 Flash 完整性的方法。CRC 计算单元有助于在运行期间计算软件的签名，并将该签名与链接时生成并存储在指定存储单元的参考签名加以比较。

## 5.2 CRC 主要特性

- 使用 CRC-32（以太网）多项式：0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 单输入/输出 32 位数据寄存器
- CRC 计算在 4 个 AHB 时钟周期（HCLK）内完成
- 8 位通用寄存器（可用于临时存储）

下图为 CRC 计算单元框图

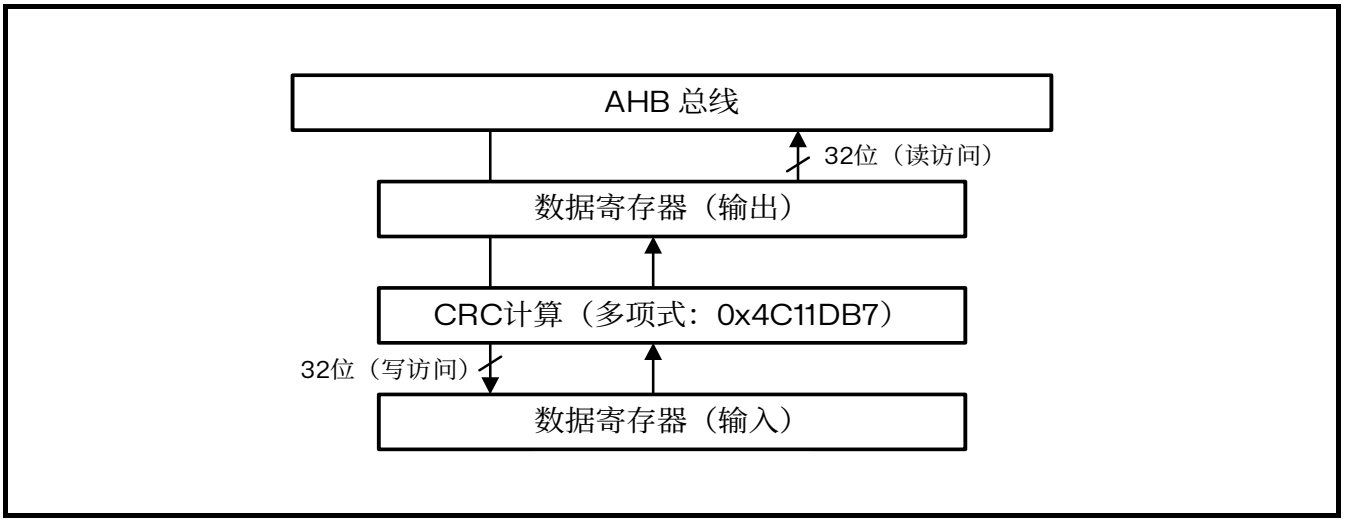



图 5.1 CRC 计算单元框图

## 5.3 CRC 功能说明

CRC 计算单元主要由单个 32 位数据寄存器组成，该寄存器：

- 用作输入寄存器，向 CRC 计算器中输入新数据（向寄存器写入数据时）
- 可保存之前的 CRC 计算结果（读取寄存器时）

对数据寄存器的每个写操作都会把当前新输入的数值和之前生成在数据寄存器中的 CRC 值再做一次 CRC 计算（CRC 计算针对整个 32 位数据字完成，而非逐字节进行）。

CRC 计算的时候，写操作被阻塞，因此允许执行背靠背写访问或连续的写读访问。

使用 CRC_CR 寄存器中的 RESET 控制位即可将 CRC 计算器复位为 0xFFFF FFFF。此操作不影响 CRC_IDR 寄存器的内容。

## 5.4 CRC 寄存器

CRC 计算单元包含两个数据寄存器和一个控制寄存器。CRC 寄存器必须按字（32 位）访问。

### 5.4.1 数据寄存器（CRC_DR）

偏移地址：0x00

复位值：0xFFFF FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DR[31:16] | | | | | | | | | | | | | | | |
| rw | 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 31:0 **DR[31:16]**：数据寄存器位
向 CRC 计算器写入新数据时用作输入寄存器。
读取寄存器时可读出之前的 CRC 计算结果。

### 5.4.2 独立数据寄存器（CRC_IDR）

地址偏移：0x04
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 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 31:8 保留，必须保持复位值。

Bit 15:0 **IDR[15:0]**：通用 16 位数据寄存器位
可用作一个字节的临时存储单元。
此寄存器不受 CRC_CR 寄存器中 RESET 位产生的 CRC 复位影响。

### 5.4.3 控制寄存器（CRC_CR）

地址偏移：0x08

复位值：0x0000 0000

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

Bit 31:1 保留，必须保持复位值。

Bit 0 **RESET**：
复位 CRC 计算单元并将数据寄存器设为 0xFFFF FFFF。
此位只能置 1，将由硬件自动进行清零。

# 6 电源控制（PWR）

## 6.1 概述

RX32G410 的工作电压（VDD/VDDA）为 2.5~3.6V。VDD 和 VDDA 必须保持一致。

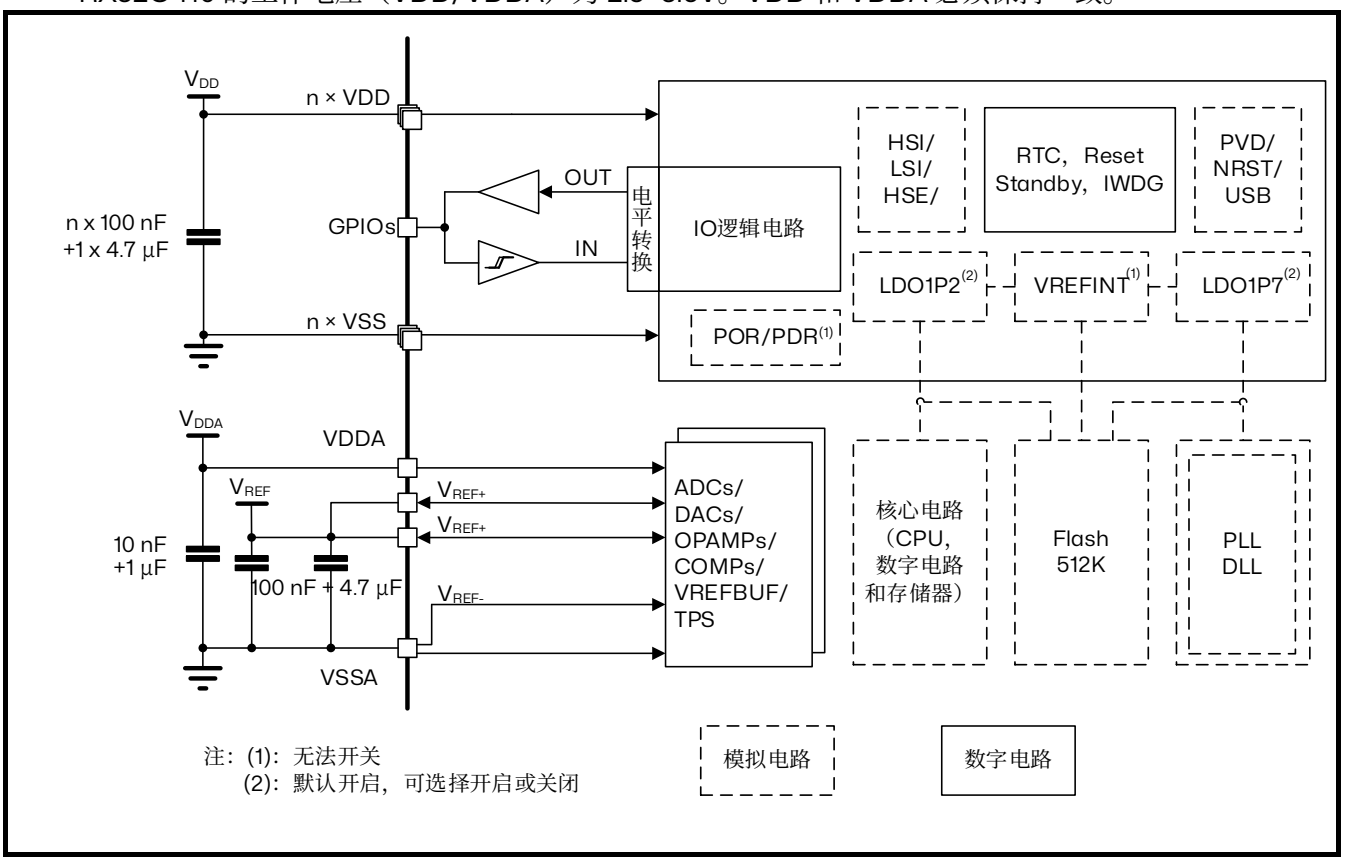



图 6.1 电源框图

## 6.2 电源管理器

### 6.2.1 上电复位（POR）和掉电复位（PDR）

RX32G410 内部有一个完整的上电复位（POR）和掉电复位（PDR）电路，当供电电压达到 2.5V 时，系统既能正常工作。

当 VDD/VDDA 低于指定的限位电压 $V_{POR}/V_{PDR}$ 时，系统保持为复位状态，无需外部复位电路。关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

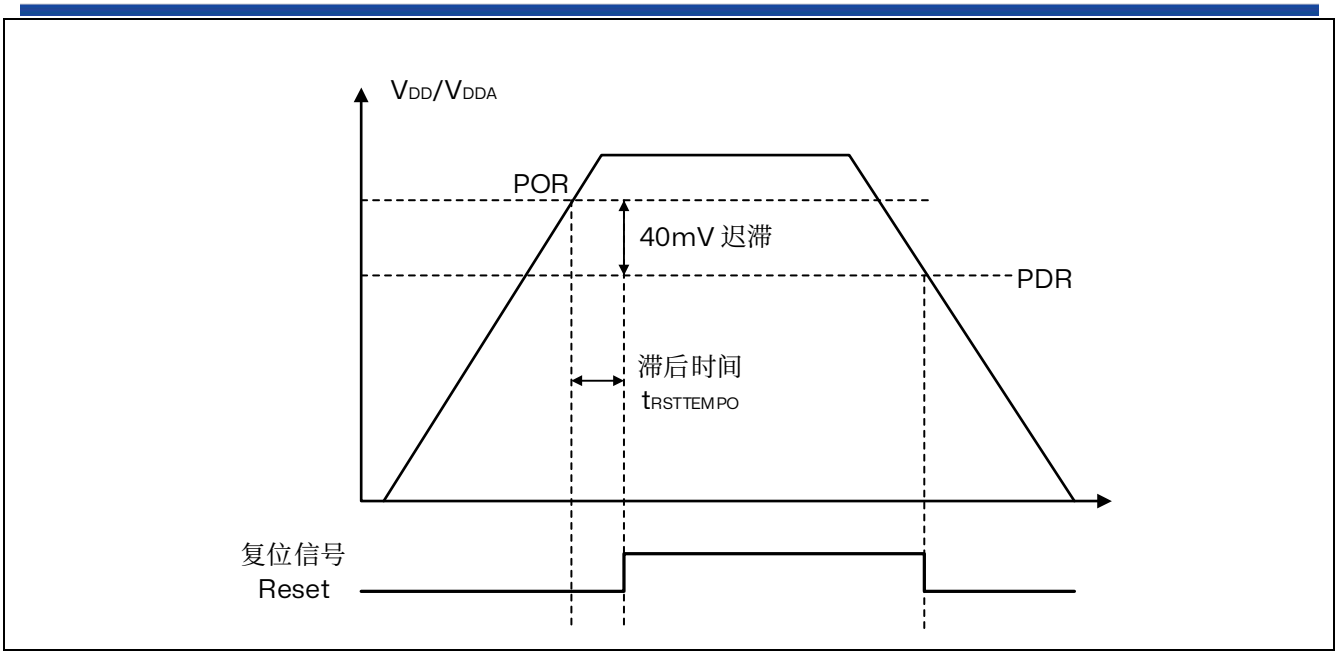



图 6.2 上电复位和掉电复位波形图

### 6.2.1 可编程电压检测器（PVD）

用户可以利用 PVD 对 VDD 电压与电源控制寄存器 2（PWR_CR2）中的 PLS[2:0]位进行比较来监控电源，这几位选择监控电压的阈值。通过设置 PVDE 位来使能 PVD。

电源状态寄存器 2（PWR_SR2）中的 PVDO 标志用来表明 VDD 是高于还是低于 PVD 的电压阈值。该事件在内部连接到外部中断的线 16，如果该中断在外部中断寄存器中是使能的，该事件就会产生中断。当 VDD 下降到 PVD 阈值以下和（或）当 VDD 上升到 PVD 阈值之上时，根据外部中断第 16 线的上升/下降边沿触发设置，就会产生 PVD 中断。

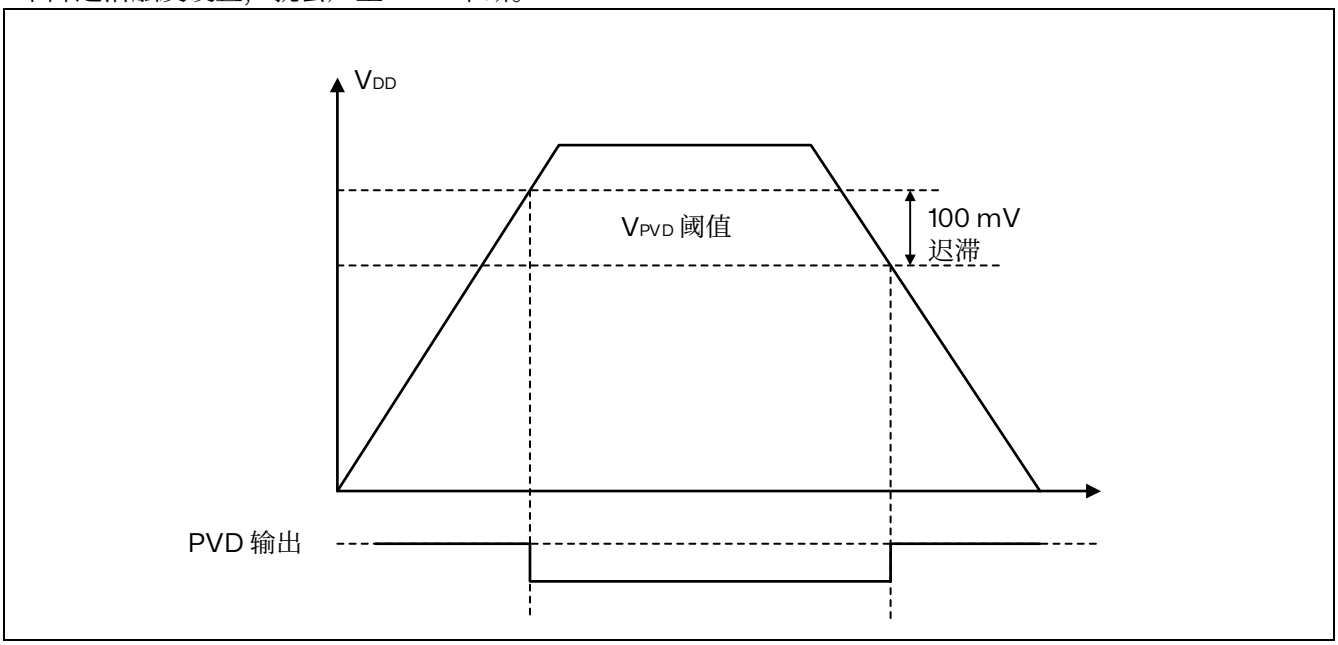



图 6.3 PVD 门限

## 6.3 低功耗模式

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

RX32G410 支持三种低功耗模式，用户可以在以下模式中进行选择：

- 睡眠模式：CPU 时钟被关闭。所有外设继续运行，并且 CPU 可以被中断或者事件唤醒。
- 停止 0 和停止 1 模式：保留 SRAM 和寄存器内容。关闭除 LSI 以外所有时钟。该系列可以被任一外部中断线从停止模式中唤醒。

  在停止 0 模式下，内部 LDO1P7 和 LDO1P2 配置为正常模式，这允许最快的唤醒时间但功耗更高。活动的外设和唤醒源与停止 1 模式相同。
- 待机模式：待机模式用于实现最低的功耗，关闭 LDO1P7 和 LDO1P2，关闭除 LSI 以外所有时钟。该系列可以被外部复位（NRST 引脚）、IWDG 复位、RTC 唤醒以及 WKUP 唤醒引脚的上升沿和下降沿从待机模式中唤醒。

*注：配置上升沿唤醒时，*WKUP *引脚不要外部上拉。配置下降沿唤醒时，*WKUP *引脚不要外部下拉。*

此外，可以通过以下方式降低运行模式的功耗：

- 降低系统时钟速度
- 当 APB 和 AHB 外设未使用时对它们进行时钟门控。

表 6.1 低功耗模式一览

| 模式 | 唤醒 | 唤醒后时钟 | 对时钟影响 | LDO | |
|---|---|---|---|---|---|
| | | | | 1P7 | 1P2 |
| 睡眠<br>(立即睡眠或<br>退出时睡眠) | 任一中断<br>唤醒事件 | 与进入睡眠模式前一致 | CPU 时钟关闭 | 正常 | 正常 |
| 停止 0 | 任意外部中断<br>（在 EXTI 寄存器中配置）<br>指定外设事件 | HSI | 关闭除 LSI 以外所有时钟 | 正常 | 正常 |
| 停止 1 | | | | 低功耗 | 低功耗 |
| 待机 | WKUP 引脚电平变化、<br>RTC 唤醒事件、<br>NRST 复位、<br>IWDG 复位 | HSI | 关闭除 LSI 以外所有时钟 | 关闭 | 关闭 |

**调试模式**

默认情况下，如果在进行调试微处理器时，使微处理器进入停止 0、停止 1 或待机模式，将失去调试连接。这是因为内核失去了时钟。

通过设置 DBGMCU_CR 寄存器中的对应配置位，可以在低功耗模式下调试软件。有关更多详细信息，请参阅章节：调试支持（DBG）。

### 6.3.1 运行模式

**降低系统时钟速度**

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（SYSCLK、HCLK、PCLK）速度。进入睡眠模式之前，也可以使用这些预分频器降低外设速度。

更多详细信息，请参阅章节：时钟配置寄存器（RCC_CFGR）。

**外设时钟门控**

在运行模式下，可随时停止各外设和存储器的 HCLK 和 PCLK 以降低功耗。

要进一步降低睡眠模式的功耗，可在进入睡眠模式之前禁止外设时钟。

外设时钟门控由 RCC_AHBENR 和 RCC_APBxENR 寄存器控制。

### 6.3.2 睡眠模式

**睡眠模式下的 I/O 状态**

睡眠模式下，所有 I/O 保持与运行模式相同的状态。

### 6.3.3 停止 0 模式

停止 0 模式基于带 FPU 的 32 位 RISC 内核深度睡眠模式基础上结合了外设的时钟控制机制。该模式下，内部 LDO 配置为正常模式。关闭除 LSI 以外所有时钟。内部 SRAM 和寄存器内容都将保留。

**停止 0 模式下的 I/O 状态**

在停止 0 模式下，所有 I/O 引脚保持与运行模式相同的状态。

### 6.3.4 停止 1 模式

停止 1 模式与停止 0 模式相同，但内部 LDO 进入低功耗模式。

### 6.3.5 待机模式

待机模式可以实现 MCU 的最低功耗。该模式是在带 FPU 的 32 位 RISC 内核深度睡眠模式时关闭内部线性稳压器 LDO1P7 和 LDO1P2。PLL、HSI 和 HSE 振荡器也被关闭。

SRAM 和寄存器内容会丢失，除了备份域和待机电路中的寄存器。

**待机模式下的 I/O 状态**

在待机模式下，I/O 可以保持在模拟状态。

PC13 上的 RTC 侵入在待机模式下是可用的。4 个唤醒引脚（WKUPx，x=1,2,3,4）也是可用的。

### 6.3.6 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微控制器（自动唤醒模式）。RTC 提供一个可编程的时间基数，用于周期性从停止（0 或 1）或待机模式下唤醒。RTC 时钟源选择 LSI 可以实现此功能。为此，可以通过在 RTC 域控制寄存器（RCC_BDCR）中编程 RTCSEL[1:0]位来选择 LSI 作为 RTC 时钟源：

- 低功耗内部 RC 振荡器（LSI）
  使用该时钟源，节省了一个 32.768kHz 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：

- 配置外部中断线 17 为上升沿触发。
- 配置 RTC 使其可产生 RTC 闹钟事件。

如果要从待机模式中唤醒，不必配置外部中断线 17。

## 6.4 PWR 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。

### 6.4.1 电源控制寄存器（PWR_CR1）

地址偏移：0x00

复位值：0x0000 0000

从待机模式唤醒后，此寄存器将被重置。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | DBP | Reserved | | | | | | LPMS[1:0] | |
| | | | | | | | rw | | | | | | | rw | rw |

Bit 31:9 保留，必须保持复位值。

Bit 8 **DBP**：取消备份区域的写保护

在复位后，RTC 和备份寄存器处于被保护状态以防意外写入。

设置这位允许写入这些寄存器。

0：禁止写入 RTC 和备份寄存器

1：允许写入 RTC 和备份寄存器

*注：如果 RTC 的时钟是 HSE/128，该位必须保持为 '1'。*

Bit 7:2 保留，必须保持复位值。

Bit 1:0 **LPMS[1:0]**：低功耗模式选择

00：停止 0 模式

01：停止 1 模式

10：保留

11：待机模式

### 6.4.2 电源控制寄存器（PWR_CR2）

地址偏移：0x04

复位值：0x0000 0008

从待机模式唤醒后，此寄存器将被重置。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PLS[2:0] | | | PVDE |
| | | | | | | | | | | | | rw | rw | rw | rw |

Bit 31:4 保留，必须保持复位值。

Bit 3:1 **PLS[2:0]**：PVD 电平选择

这些位用于选择电源电压监测器的电压阈值

000：2.2V　　100：2.6V

001：2.3V　　101：2.7V

010：2.4V　　110：2.8V

011：2.5V　　111：2.9V

*注：当 SYSCFG_CFGR2 寄存器的 PVDL 位被设置时此位被写保护。*
*此保护仅能通过系统复位来重置。*

Bit 0 **PVDE**：电源电压监测器(PVD)使能

0：禁止 PVD

1：开启 PVD

*注：当 SYSCFG_CFGR2 寄存器的 PVDL 位被设置时此位被写保护。*
*此保护仅能通过系统复位来重置。*

### 6.4.3 电源控制寄存器（PWR_CR3）

地址偏移：0x08

复位值：0x0000 0000

退出待机模式以及 RCC_APB1RSTR1 寄存器中的 PWRRST 位被设置时，该寄存器不会被重置。

访问：访问此寄存器需要比标准 APB 访问更多的 APB 周期（写操作需要 3 个周期，读操作需要 2 个周期）。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIWUL | Reserved | | | | | | | | | | | EWUP4 | EWUP3 | EWUP2 | EWUP1 |
| rw | | | | | | | | | | | | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15 **EIWUL**：使能内部唤醒线（RTC/IWDG）

0：禁止内部唤醒线唤醒

1：使能内部唤醒线

Bit 14:4 保留，必须保持为复位值。

Bit 3 **EWUP4**：使能 WKUP 引脚 4

0：WKUP 引脚 4 为通用 I/O。WKUP 引脚 4 上的事件不能将 CPU 从待机模式唤醒

1：WKUP 引脚 4 用于将 CPU 从待机模式唤醒，WKUP 引脚 4 上的上升沿/下降沿将系统从待机模式唤醒，有效边沿通过 PWR_CR4 的 WP4 设置。

Bit 2 **EWUP3**：使能 WKUP 引脚 3

0：WKUP 引脚 3 为通用 I/O。WKUP 引脚 3 上的事件不能将 CPU 从待机模式唤醒

1：WKUP 引脚 3 用于将 CPU 从待机模式唤醒，WKUP 引脚 3 上的上升沿/下降沿将系统从待机模式唤醒，有效边沿通过 PWR_CR4 的 WP3 设置。

Bit 1 **EWUP2**：使能 WKUP 引脚 2

0：WKUP 引脚 2 为通用 I/O。WKUP 引脚 2 上的事件不能将 CPU 从待机模式唤醒

1：WKUP 引脚 2 用于将 CPU 从待机模式唤醒，WKUP 引脚 2 上的上升沿/下降沿将系统从待机模式唤醒，有效边沿通过 PWR_CR4 的 WP2 设置。

Bit 0 **EWUP1**：使能 WKUP 引脚 1

0：WKUP 引脚 1 为通用 I/O。WKUP 引脚 1 上的事件不能将 CPU 从待机模式唤醒

1：WKUP 引脚 1 用于将 CPU 从待机模式唤醒，WKUP 引脚 1 上的上升沿/下降沿将系统从待机模式唤醒，有效边沿通过 PWR_CR4 的 WP1 设置。

### 6.4.4 电源控制寄存器（PWR_CR4）

地址偏移：0x0C
复位值：0x0000 0000
退出待机模式以及 RCC_APB1RSTR1 寄存器中的 PWRRST 位被设置时，该寄存器不会被重置。
访问：访问此寄存器需要比标准 APB 访问更多的 APB 周期（写操作需要 3 个周期，读操作需要 2 个周期）。

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

Bit 31:4 保留，必须保持为复位值。

Bit 3 **WP4**：WKUP4 引脚唤醒极性
此位定义了用于外部唤醒引脚上的事件检测极性，WKUP4
0：检测到高电平（上升沿）
1：检测到低电平（下升沿）

*注：配置上升沿唤醒时，*WKUP *引脚不要外部上拉。配置下降沿唤醒时，*WKUP *引脚不要外部下拉。*

Bit 2 **WP3**：WKUP3 引脚唤醒极性
此位定义了用于外部唤醒引脚上的事件检测极性，WKUP3
0：检测到高电平（上升沿）
1：检测到低电平（下升沿）

*注：配置上升沿唤醒时，*WKUP *引脚不要外部上拉。配置下降沿唤醒时，*WKUP *引脚不要外部下拉。*

Bit 1 **WP2**：WKUP2 引脚唤醒极性
此位定义了用于外部唤醒引脚上的事件检测极性，WKUP2
0：检测到高电平（上升沿）
1：检测到低电平（下升沿）

*注：配置上升沿唤醒时，*WKUP *引脚不要外部上拉。配置下降沿唤醒时，*WKUP *引脚不要外部下拉。*

Bit 0 **WP1**：WKUP1 引脚唤醒极性
此位定义了用于外部唤醒引脚上的事件检测极性，WKUP1
0：检测到高电平（上升沿）
1：检测到低电平（下升沿）

*注：配置上升沿唤醒时，*WKUP *引脚不要外部上拉。配置下降沿唤醒时，*WKUP *引脚不要外部下拉。*

### 6.4.5 电源状态寄存器 1（PWR_SR1）

地址偏移：0x10

复位值：0x0000 0000

退出待机模式以及 RCC_APB1RSTR1 寄存器中的 PWRRST 位被设置时，该寄存器不会被重置。

访问：与此标准 APB 读取相比，读取此寄存器需要额外的 2 个 APB 周期。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WUFI | Reserved | | | | | | SBF | Reserved | | | | WUF4 | WUF3 | WUF2 | WUF1 |
| r | | | | | | | r | | | | | r | r | r | r |

Bit 31:16 保留，必须保持复位值。

Bit 15 **WUFI**：内部唤醒标志（IWDG/RTC 唤醒）

当在内部唤醒线上检查到唤醒时，设置此位。当所有内部唤醒源被清除时该位被清除。

Bit 14:9 保留，必须保持复位值。

Bit 8 **SBF**：待机标志

该位由硬件设置，并只能由 POR/PDR（上电/掉电复位）或设置电源状态清除寄存器（PWR_SCR）的 CSBF 位清除。

0：系统不在待机模式

1：系统进入待机模式

Bit 7:4 保留，必须保持复位值。

Bit 3 **WUF4**：WKUP4 引脚唤醒标志

当在唤醒引脚 WKUP4 检测到一个唤醒事件时此位被设置，被清除通过向电源状态清除寄存器（PWR_SCR）的 CWUF4 位写“1”。

Bit 2 **WUF3**：WKUP3 引脚唤醒标志

当在唤醒引脚 WKUP3 检测到一个唤醒事件时此位被设置，被清除通过向电源状态清除寄存器（PWR_SCR）的 CWUF3 位写“1”。

Bit 1 **WUF2**：WKUP2 引脚唤醒标志

当在唤醒引脚 WKUP2 检测到一个唤醒事件时此位被设置，被清除通过向电源状态清除寄存器（PWR_SCR）的 CWUF2 位写“1”。

Bit 0 **WUF1**：WKUP1 引脚唤醒标志

当在唤醒引脚 WKUP1 检测到一个唤醒事件时此位被设置，被清除通过向电源状态清除寄存器（PWR_SCR）的 CWUF1 位写“1”。

### 6.4.6 电源状态寄存器 2（PWR_SR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | PVDO | Reserved | | | | | | | | | | |
| | | | | r | | | | | | | | | | | |

Bit 31:12 保留，必须保持复位值。

Bit 11 **PVDO**：PVD 输出

当 PVD 被 PVDE 位使能后该位才有效

0：$V_{DD}/V_{DDA}$ 高于由 PLS[2:0]选定的 PVD 阈值

1：$V_{DD}/V_{DDA}$ 低于由 PLS[2:0]选定的 PVD 阈值

*注：在待机模式下*PVD *被停止。因此，待机模式后或复位后，直到设置*PVDE *位之前，该位为*0。

Bit 10:0 保留，必须保持复位值。

### 6.4.7 电源状态清除寄存器（PWR_SCR）

地址偏移：0x18

复位值：0x0000 0000

访问：与此标准 APB 写入相比，写入此寄存器需要额外的 3 个 APB 周期。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CWUFI | Reserved | | | | | | CSBF | Reserved | | | | CWUF4 | CWUF3 | CWUF2 | CWUF1 |
| w | | | | | | | w | | | | | w | w | w | w |

Bit 31:16 保留，必须保持复位值。

Bit 15 **CWUFI**：清除内部唤醒标志

写 1 清除 PWR_SR1 寄存器的内部唤醒标志 WUFI

Bit 14:9 保留，必须保持复位值。

Bit 8 **CSBF**：清除待机标志

写 1 清除 PWR_SR1 寄存器的待机标志 SBF

Bit 7:4 保留，必须保持复位值。

Bit 3 **CWUF4**：清除唤醒标志 4

写 1 清除 PWR_SR1 寄存器的 WKUP4 引脚唤醒标志 WUF4

Bit 2 **CWUF3**：清除唤醒标志 3

写 1 清除 PWR_SR1 寄存器的 WKUP3 引脚唤醒标志 WUF3

Bit 1 **CWUF2**：清除唤醒标志 2

写 1 清除 PWR_SR1 寄存器的 WKUP2 引脚唤醒标志 WUF2

Bit 0 **CWUF1**：清除唤醒标志 1

写 1 清除 PWR_SR1 寄存器的 WKUP1 引脚唤醒标志 WUF1

# 7 备份寄存器（BKP）

## 7.1 BKP 简介

BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

复位后，对备份寄存器和 RTC 的访问被禁止，并且备份域被保护以防止可能存在的意外的写操作。执行以下操作可以使能对备份寄存器和 RTC 的访问。

- 通过设置寄存器 RCC_APB1ENR1 的 PWREN 和 BKPEN 位来打开电源和备份接口的时钟。
- 电源控制寄存器（PWR_CR1）的 DBP 位来使能对备份寄存器和 RTC 的访问。

## 7.2 BKP 特性

- 用来管理防侵入检测并具有中断功能的状态/控制寄存器
- 用来存储 RTC 校验值的校验寄存器
- 在 PC13 引脚（当该引脚不用于侵入检测时）上输出 RTC 校准时钟，RTC 闹钟脉冲或者秒脉冲

## 7.3 BKP 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。BKP_CSR 寄存器中的侵入事件标志位 TEF 位会被置起。

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。等待本次侵入事件处理完成后重新用 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产生。这相当于对侵入引脚 TAMPER 进行电平检测。

### 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 121ppm。

## 7.4 BKP 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器可以用半字（16 位）或字（32 位）进行访问。

### 7.4.1 RTC 时钟校准寄存器（BKP_RTCCR）

地址偏移：0x2C
复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | ASOS | ASOE | CCO | CAL[6:0] | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15:10 保留，必须保持为复位值。

Bit 9 **ASOS**：闹钟或秒输出选择
当设置了 ASOE 位，ASOS 位可用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲还是闹钟脉冲信号。
0：输出 RTC 闹钟脉冲
1：输出秒脉冲

*注：该位只能被备份区的复位所清除*

Bit 8 **ASOE**：闹钟或秒脉冲输出使能
根据 ASOS 位的设置，该位允许 RTC 闹钟或秒脉冲输出到 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：该位只能被备份区的复位所清除*

Bit 7 **CCO**：校准时钟输出
0：无影响
1：此位置 1 可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 CCO 位置 1 时，必须关闭侵入检测功能以避免检测到无用的侵入信号。

*注：当 VDD 供电断开时，该位被清除。*

Bit 6:0 **CAL[6:0]**：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 0~121ppm。

### 7.4.2 备份控制寄存器（BKP_CR）

地址偏移：0x30
复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | TPAL | TPE |
| | | | | | | | | | | | | | | rw | rw |

Bit 15:2 保留，必须保持为复位值。

Bit 1 **TPAL**：侵入检测 TAMPER 引脚有效电平
0：侵入检测 TAMPER 引脚上高电平有效（如果 TPE 位为 1）
1：侵入检测 TAMPER 引脚上低电平有效（如果 TPE 位为 1）

Bit 0 **TPE**：侵入检测 TAMPER 引脚使能

0：侵入检测 TAMPER 引脚作为通用 I/O 口使用

1：使能侵入检测引脚作为侵入检测使用

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

### 7.4.3 备份控制/状态寄存器（BKP_CSR）

地址偏移：0x34

复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | TIF | TEF | Reserved | | | | | TPIE | CTI | CTE |
| | | | | | | r | r | | | | | | rw | w | w |

Bit 15:10 保留，必须保持为复位值。

Bit 9 **TIF**：侵入中断标志

当检测到有侵入事件且 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（同时也清除了中断）。如果 TPIE 位被清除，则此位也会被清除。

0：无侵入中断

1：已产生侵入中断

*注：仅当系统复位或由待机模式唤醒后才复位该位。*

Bit 8 **TEF**：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：已检测到侵入事件

Bit 7:3 保留，必须保持为复位值。

Bit 2 **TPIE**：使能侵入 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）

*注：侵入中断无法将系统内核从低功耗模式唤醒。*

*注：仅当系统复位或由待机模式唤醒后才复位该位。*

Bit 1 **CTI**：清除侵入检测中断

此位只能写入，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

Bit 0 **CTE**：清除侵入检测事件

此位只能写入，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）

# 8 复位和时钟控制（RCC）

## 8.1 复位

有三种类型的复位，分别定义为系统复位、电源复位和 RTC 域复位。

### 8.1.1 电源复位

当发生以下任一事件时，会产生电源复位：

1. 上电复位（POR）或掉电复位（PDR 复位）
2. 退出待机模式时

上电或掉电复位（POR/PDR），会将所有寄存器设置为其复位值，RTC 域除外。

当退出待机模式时，$V_{CORE}$ 域中的所有寄存器都会被设置为其复位值。$V_{CORE}$ 域之外的寄存器（RTC、WKUP、IWDG 以及待机模式控制）不会受到影响。

### 8.1.2 系统复位

系统复位会将所有寄存器设置为其复位值，时钟控制/状态寄存器（RCC_CSR）中的复位标志和 RTC 域中的寄存器除外。

系统复位在发生以下任一事件时生成：

1. NRST 引脚出现低电平（外部复位）
2. 窗口看门狗事件（WWDG 复位）
3. 独立看门狗事件（IWDG 复位）
4. 软件复位（SW 复位）（见软件复位）
5. 低功耗模式安全复位（见低功耗模式安全复位）
6. 选项字节加载复位（见选项字节加载复位）

可以通过检查控制/状态寄存器 RCC_CSR 中的复位标志来识别复位源。

#### NRST 引脚（外部复位）

引脚上的任何有效复位信号都会传播到设备内部逻辑，并且所有内部复位源都会通过脉冲发生器外部驱动至该引脚。脉冲发生器保证每个内部复位源在 NRST 引脚上输出时具有至少 20μs 的最小复位脉冲持续时间。

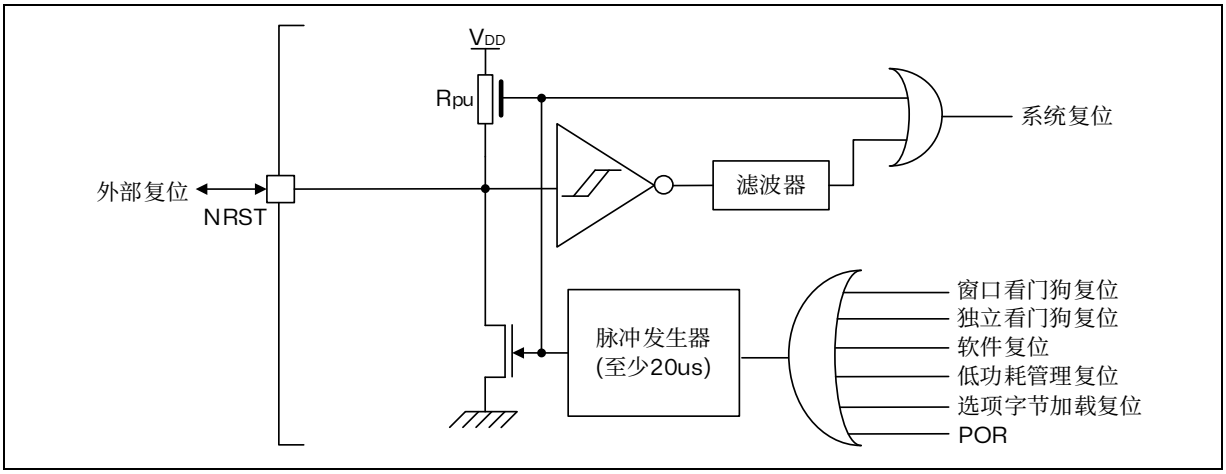



图 8.1 复位电路框图

**低功耗模式安全复位**

为防止关键应用错误地进入低功耗模式，提供了两种低功耗模式安全复位。如果在选项字节中使能，则在以下条件下生成复位：

1. 进入待机模式：通过清零用户选项字节中的 nRST_STDBY 位来使能此类型复位。这时，即使执行了进入待机模式的过程，系统将被复位而不是进入待机模式。
2. 进入停止模式：通过清零用户选项字节中的 nRST_STOP 位来使能此类型复位。这时，即使执行了进入停机模式的过程，系统将被复位而不是进入停机模式。

有关用户选项字节的更多信息，请参阅 4.5.1 节：选项字节描述。

**选项字节加载复位**

当 FLASH_CR 寄存器中的 OBL_LAUNCH 位（第 27 位）被置位时，会生成选项字节加载复位。该位用于通过软件启动选项字节加载。

### 8.1.3 RTC 域复位

RTC 域有两种特定的复位。

当发生以下任一事件时，会生成 RTC 域复位：

1. 软件复位，通过设置 RTC 域控制寄存器（RCC_BDCR）中的 BDRST 位来触发。
2. 在 $V_{DD}$ 掉电的前提下，$V_{DD}$ 上电将引发 RTC 域复位。

RTC 域复位仅影响 RTC、备份寄存器和 RCC RTC 域控制寄存器。

## 8.2 时钟

四种不同的时钟源可被用来驱动系统时钟（SYSCLK）：

- LSI（低速内部）32KHz RC 振荡器时钟
- HSI16（高速内部）16 MHz RC 振荡器时钟
- HSE 振荡器时钟，频率范围 16 至 24MHz
- PLL 时钟

复位后启动，HSI16 被用作系统时钟源。

同时设备的 LSI 还作为附加时钟源：

- 32 kHz 低速内部 RC（LSI RC），驱动独立看门狗，并可选择驱动用于从停止和待机模式自动唤醒的实时时钟（RTC）。

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

可使用多个预分频器来配置 AHB 频率、APB1 和 APB2 域。AHB、APB1 和 APB2 域的最大频率为 192MHz。

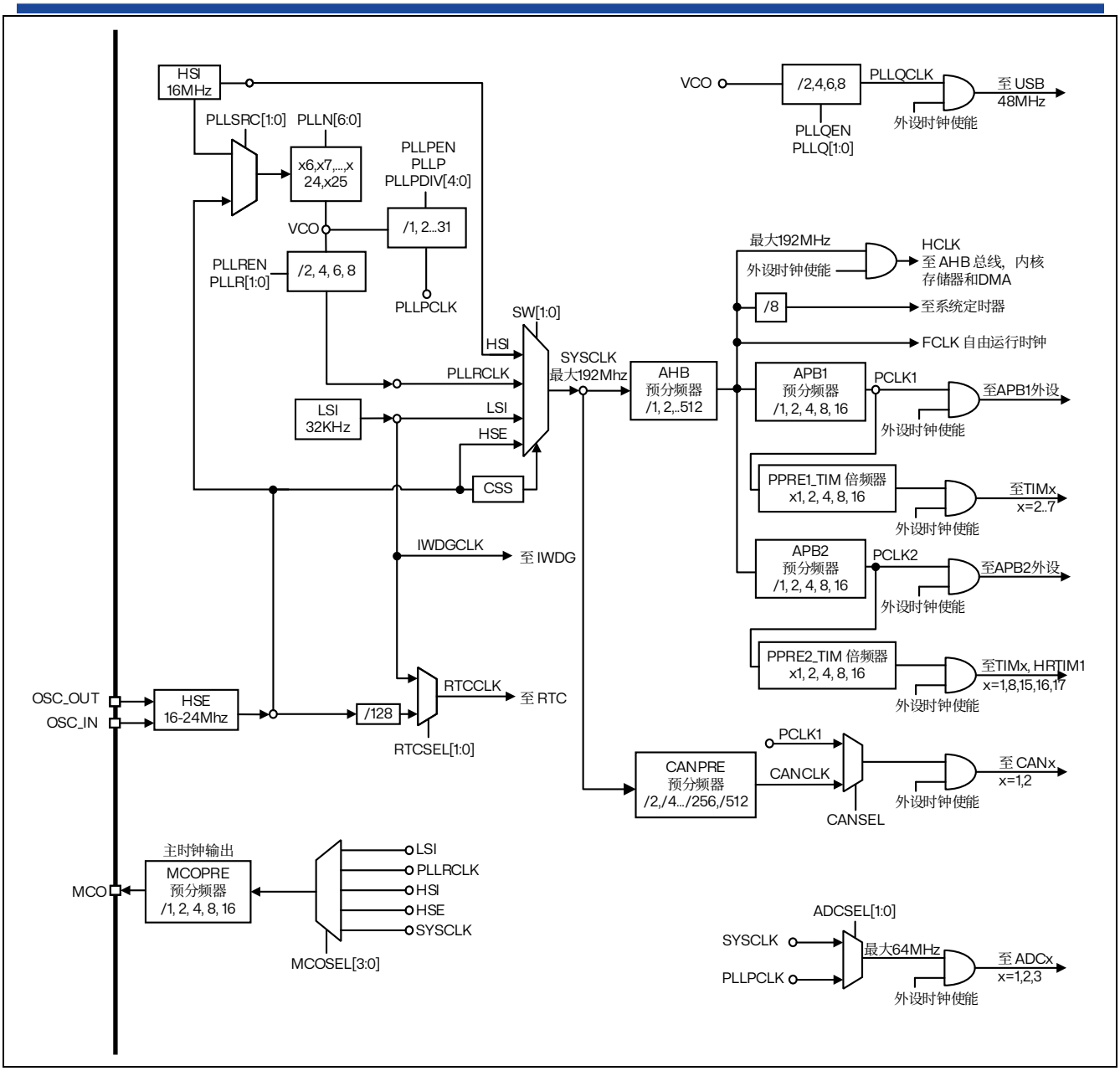
除以下情况外，所有外设时钟均源自其总线时钟（HCLK、PCLK1 或 PCLK2）：

- 用于 USB 设备 FS 的 48 MHz 时钟。此时钟可通过软件设置源来派生：
  - PLL“Q”时钟
- ADC 时钟，可通过软件从以下源中选择一个来派生：
  - 系统时钟（SYSCLK）
  - PLL“P”时钟
- UART 时钟，可通过软件从以下两个源中选择一个来派生：
  - APB1 或 APB2 时钟（根据 UART 映射到哪个 APB，选择 PCLK1 或 PCLK2）
- I2C 时钟，可通过软件从以下源中选择一个来派生：
  - APB1 时钟（PCLK1）
- RTC 时钟，可通过软件从以下两个源中选择一个来派生：
  - LSI 时钟
  - HSE 时钟除以 128

  仅当时钟为 LSI 时，才支持停止模式（包括唤醒）功能。
- IWDG 时钟始终为 LSI 时钟。
- CAN 时钟，可通过软件从以下两个源中选择一个来派生：
  - CANCLK 时钟（由系统时钟 SYSCLK 分频而来）
  - PCLK1 时钟

RCC 通过 AHB 时钟（HCLK）8 分频后作为内核系统定时器的外部时钟。

FCLK 作为带 FPU 的 32 位 RISC 内核的自由运行时钟。




1. 有关内部和外部时钟源特性的详细信息，请参阅设备数据手册中的“电气特性”部分。

图 8.2 时钟树

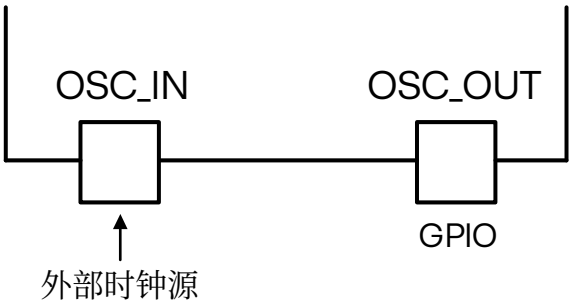
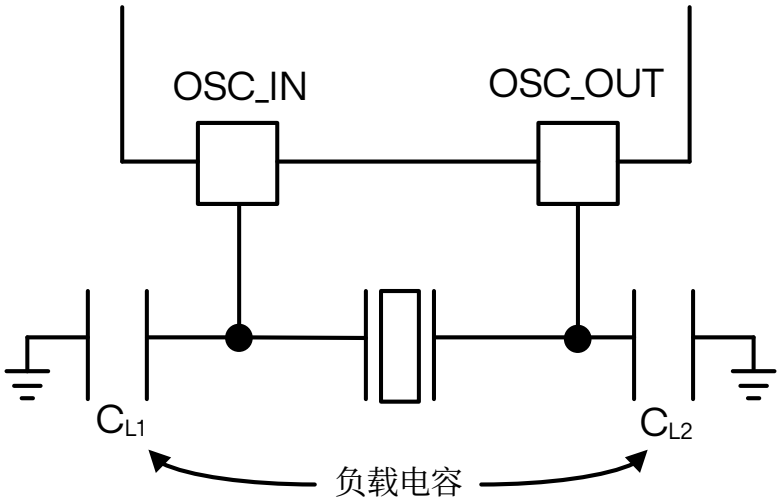
### 8.2.1 HSE 时钟

高速外部时钟信号（HSE）由以下两种时钟源产生：

- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

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

图 8.3 HSE 时钟源

| 时钟源 | 硬件配置 |
| --- | --- |
| 外部时钟 |  |
| 晶体/陶瓷谐振器 |  |

**外部晶体/陶瓷谐振器（HSE 晶体）**

16 至 24 MHz 的外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考上图，进一步信息可参考数据手册的电气特性部分。

时钟控制寄存器（RCC_CR）中的 HSERDY 标志指示 HSE 振荡器是否稳定。启动时，直到此位由硬件设置，时钟才会被释放。如果在时钟中断使能寄存器（RCC_CIER）中使能，则可以生成中断。

可以使用时钟控制寄存器（RCC_CR）中的 HSEON 位来打开和关闭 HSE 晶体。

**外部源（HSE 旁路）**

在此模式下，必须提供外部时钟源。其频率可达 24MHz。通过设置时钟控制寄存器（RCC_CR）中的 HSEBYP 和 HSEON 位来选择此模式。外部时钟信号（方波、正弦波或三角波）需具有约 40-60%的占空比（具体取决于频率，请参阅数据手册），并驱动 OSC_IN 引脚，而 OSC_OUT 引脚可用作 GPIO。见上图。

### 8.2.2 HSI16 时钟

HSI16 时钟信号由内部 16 MHz 的 RC 振荡器生成。

HSI16 RC 振荡器的优势在于以低成本（无需外部组件）提供时钟源。其启动时间也比 HSE 晶体振荡器快，然而，即使经过校准，其频率准确性仍低于外部晶体振荡器或陶瓷谐振器。

HSI16 时钟可在从停止模式（Stop 0、Stop 1）唤醒后选择为系统时钟。请参阅第 8.3 节：低功耗模式。如果 HSE 晶体振荡器出现故障，它也可以作为备用时钟源（辅助时钟）。请参阅第 8.2.7 节：时钟安全系统（CSS）。

**校准**

由于制造工艺的差异，RC 振荡器频率在不同芯片之间可能会有所不同，因此睿兴会在工厂对每颗芯片进行校准，以确保在 $T_A$=25°C 时具有 1%的准确性。

复位后，工厂校准值会被加载到内部时钟源校准寄存器（RCC_ICSCR）的 HSICAL[7:0]位中。

时钟控制寄存器（RCC_CR）中的 HSIRDY 标志指示 HSI16 振荡器是否稳定。启动时，直到此位由硬件设置，HSI16 输出时钟才会被释放。

可以使用时钟控制寄存器（RCC_CR）中的 HSION 位来打开和关闭 HSI16 振荡器。

如果 HSE 晶体振荡器出现故障，HSI16 信号也可以作为备用源（辅助时钟）。请参阅第 8.2.7 节：时钟安全系统（CSS）。

### 8.2.3 PLL 内部锁相环

内部锁相环 PLL 可用于将 HSI16 或 HSE 的输出时钟频率进行倍频。PLL 的输入频率必须在设备数据手册中定义的范围内。所选时钟源会通过可编程倍频因子 PLLN（6 至 25）进行倍频，以提供所需输入范围内的时钟频率。请参阅图：时钟树和 PLL 配置寄存器（RCC_PLLCFGR）。

必须在使能 PLL 之前完成 PLL 配置（包括输入时钟选择和倍频因子的设定）。一旦 PLL 被使能，这些参数便无法更改。如需修改 PLL 配置，请按照以下步骤操作：

1. 在时钟控制寄存器（RCC_CR）中将 PLLON 设置为 0，以禁用 PLL。
2. 等待 PLLRDY 被清除。此时 PLL 已完全停止。
3. 更改所需参数。
4. 在时钟控制寄存器（RCC_CR）中将 PLLON 设置为 1，以再次使能 PLL。
5. 在 PLL 配置寄存器（RCC_PLLCFGR）中配置 PLLPEN、PLLQEN、PLLREN，以使能所需的 PLL 输出。

如果在时钟中断使能寄存器（RCC_CIER）中使能，当 PLL 准备就绪时，可以生成中断。

PLL 的输出频率不得超过 192 MHz。

每个 PLL 输出时钟的使能位（PLLPEN、PLLQEN、PLLREN）可在不停止相应 PLL 的情况下随时修改。如果 PLLCLK 被用作系统时钟，则不能清除 PLLREN。

### 8.2.4 LSI 时钟

LSI RC 作为低功耗时钟源，可在停止和待机模式下为独立看门狗（IWDG）和实时时钟（RTC）持续运行。时钟频率为 32KHz。有关更多详细信息，请参阅数据手册中的电气特性部分。

LSI RC 可通过控制/状态寄存器（RCC_CSR）中的 LSION 位进行开启和关闭。

控制/状态寄存器（RCC_CSR）中的 LSIRDY 标志用于指示 LSI 振荡器是否稳定。在系统启动时，直到此位由硬件设置，时钟才会被释放。如果在时钟中断使能寄存器（RCC_CIER）中使能，可以生成中断。

### 8.2.5 系统时钟（SYSCLK）选择

系统时钟（SYSCLK）可由四种不同的时钟源驱动：

- LSI 振荡器
- HSI16 振荡器
- HSE 振荡器
- PLL 锁相环

系统时钟的最大频率为 192 MHz。系统复位后，默认选择 HSI16 振荡器作为系统时钟。当某个时钟源被直接用作或通过 PLL 用作系统时钟时，该时钟源无法停止。

仅当目标时钟源准备就绪（启动延迟后时钟稳定或 PLL 锁定）时，才能从一个时钟源切换到另一个时钟源。如果选择了尚未准备就绪的时钟源，则会在该时钟源准备就绪时进行切换。时钟控制寄存器（RCC_CR）和控制/状态寄存器（RCC_CSR）中的状态位指示哪些时钟已准备就绪。时钟配置寄存器（RCC_CFGR）中的 SWS[1:0]系统时钟切换状态指示当前正在使用哪个时钟作为系统时钟。

建议从低速系统时钟切换到高速系统时钟，或从高速切换到低速时，使用中等速度时钟作为过渡状态，持续至少 1μs。

时钟源切换条件：

- 从 HSE 或 HSI16 切换到 AHB 频率（HCLK）高于 80 MHz 的 PLL
- 从 HCLK 高于 80 MHz 的 PLL 切换到 HSE 或 HSI16

过渡状态：

- 将 AHB 预分频器 HPRE[3:0]位设置为将系统频率除以 2
- 将系统时钟切换到 PLL
- 等待至少 1μs，然后重新配置 AHB 预分频器位以获得所需的 HCLK 频率

### 8.2.6 时钟安全系统（CSS）

时钟安全系统可通过软件激活。在此情况下，时钟检测器在 HSE 振荡器启动延迟后使能，并在该振荡器停止时禁用。若检测到 HSE 时钟故障，HSE 振荡器将自动禁用，同时向高级控制定时器（TIM1/TIM8 及 TIM15/16/17）的中断输入和 hrtim_sys_flt 发送时钟故障事件，并生成中断以通知软件出现故障（时钟安全系统中断 CSSI），从而使微控制器（MCU）能够执行救援操作。CSSI 与带 FPU 的 32 位 RISC 内核的非屏蔽中断相关联。

*注：一旦使能 CSS 且 HSE 时钟出现故障，将触发 CSS 中断并自动生成非屏蔽中断。非屏蔽中断将无限执行，除非清除 CSS 中断挂起位。因此，在非屏蔽中断中，用户必须通过设置时钟中断清除寄存器（RCC_CICR）中的 CSSC 位来清除 CSS 中断。*

如果 HSE 振荡器被直接或间接地作为系统时钟，(间接的意思是：它被作为 PLL 输入时钟，并且 PLL 时钟被作为系统时钟)，时钟故障将导致系统时钟自动切换到 HSI16 振荡器，同时外部 HSE 振荡器被关闭。在时钟失效时，如果 HSE 振荡器时钟（被分频或未被分频）是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

### 8.2.7 ADC 时钟

ADC 时钟源自系统时钟或 PLL 的“P”输出。

有关 ADC 时钟的最大值，请参阅设备数据手册。

### 8.2.8 RTC 时钟

RTCCLK 时钟源可以是 HSE/128 或 LSI 时钟。通过编程 RTC 域控制寄存器（RCC_BDCR）中的 RTCSEL[1:0]位来选择时钟源。除非 RTC 域复位，此选择不能被改变。为确保 RTC 正常工作，系统必须始终配置为使得 PCLK 频率大于或等于 RTCCLK 频率。

HSE 和 LSI 时钟不在 RTC 域内。因此：

- 若选择 LSI 作为 RTC 时钟：
  - 若 $V_{DD}$ 供电关闭，则 RTC 状态无法保证。
- 若使用经过预分频的 HSE 时钟作为 RTC 时钟：
  - 若 $V_{DD}$ 供电关闭或内部电压调节器关闭（从 $V_{CORE}$ 域断电），则 RTC 状态无法保证。

当 RTC 时钟为 LSI 时，在系统复位期间，RTC 仍保持时钟供应并处于功能状态。

### 8.2.9 定时器时钟

定时器时钟频率由专用的定时器时钟配置寄存器（RCC_CFGR1）进行倍频。若 APB 预分频值为 1，则定时器时钟频率被设置为与 APB 域相同的频率；否则，根据 APB 预分频值，它们可被软件设置不同的倍频值，详见定时器时钟配置寄存器（RCC_CFGR1）。

### 8.2.10 看门狗时钟

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

### 8.2.11 时钟输出

- MCO
  微控制器时钟输出（MCO）功能允许将时钟信号输出到外部的 MCO 引脚。
  可从五个时钟信号中选择一个作为 MCO 时钟：
  - LSI
  - SYSCLK
  - HSI16
  - HSE
  - PLLCLK

  选择由时钟配置寄存器（RCC_CFGR）中的 MCOSEL[3:0]位域控制。所选时钟可通过时钟配置寄存器（RCC_CFGR）中的 MCOPRE[2:0]位域进行分频。

MCO 时钟输出需要 MCO 引脚上选择相应的复用功能。

### 8.2.12 使用 TIM5/TIM15/TIM16/TIM17 进行内部/外部时钟测量

可通过 TIM5、TIM15、TIM16 或 TIM17 的通道 1 输入捕获功能间接测量所有板上时钟源的频率，如下列图所示。

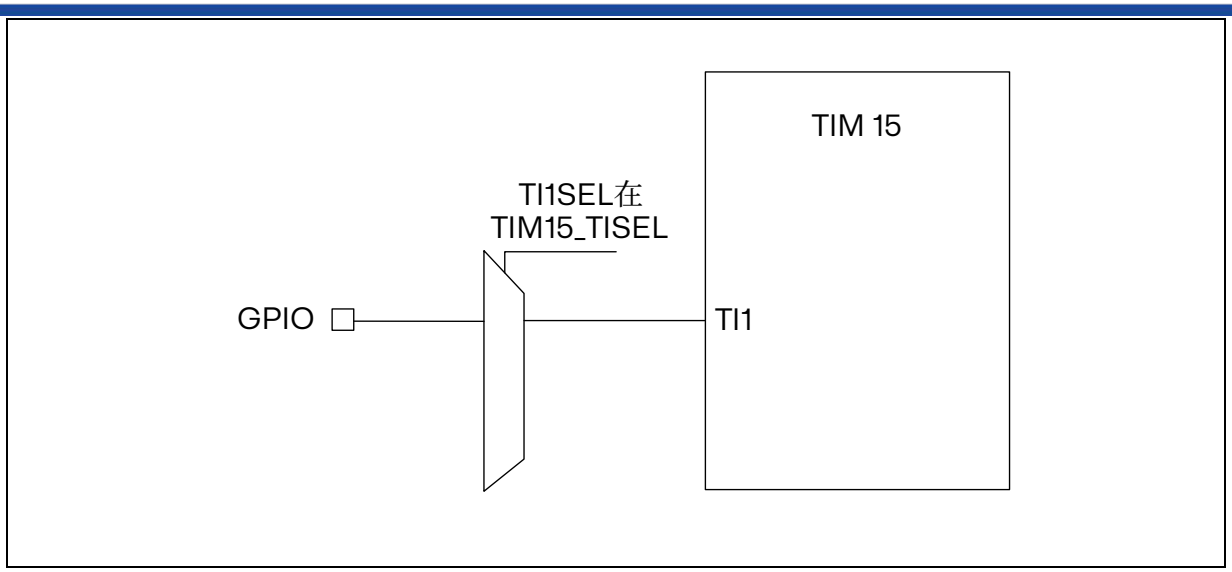



图 8.4 使用 TIM15 捕获模式进行频率测量

定时器 15 的输入捕获通道可以是一个 GPIO 引脚。如下:

- TIM15 通道 1 连接到 GPIO。请参考设备数据手册中的复用功能映射。

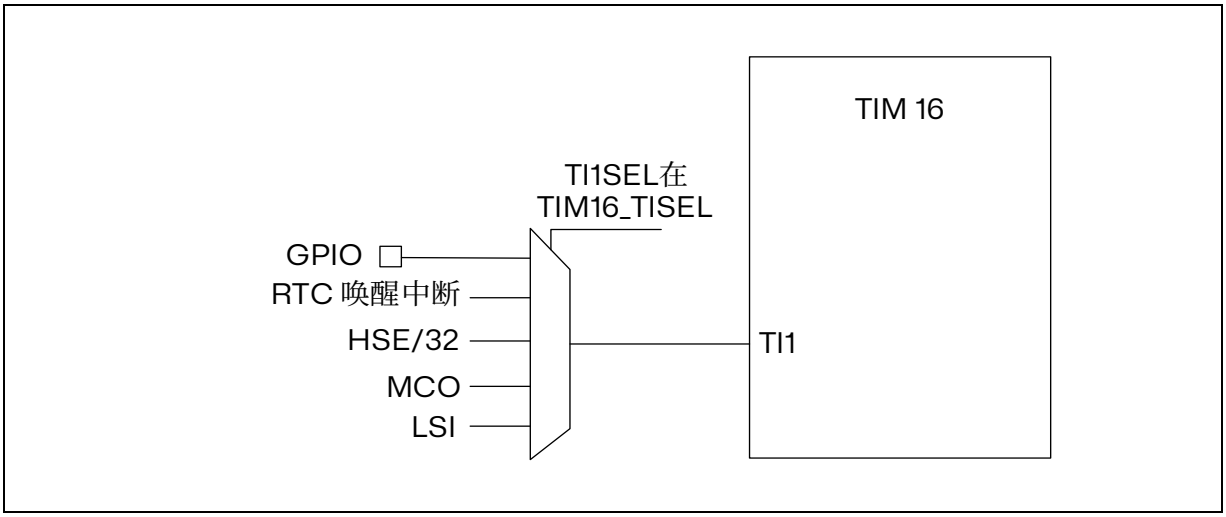



图 8.5 使用 TIM16 捕获模式进行频率测量

定时器 16 的输入捕获通道可以是一个 GPIO 引脚或 MCU 的内部时钟。可能性如下:

- TIM16 通道 1 连接到 GPIO。请参考设备数据手册中的复用功能映射。
- TIM16 通道 1 连接到 RTC 唤醒中断信号。在这种情况下，应使能 RTC 中断。
- TIM16 通道 1 连接到 HSE/32 时钟。
- TIM16 通道 1 连接到 MCO。
- TIM16 通道 1 连接到 LSI 时钟。

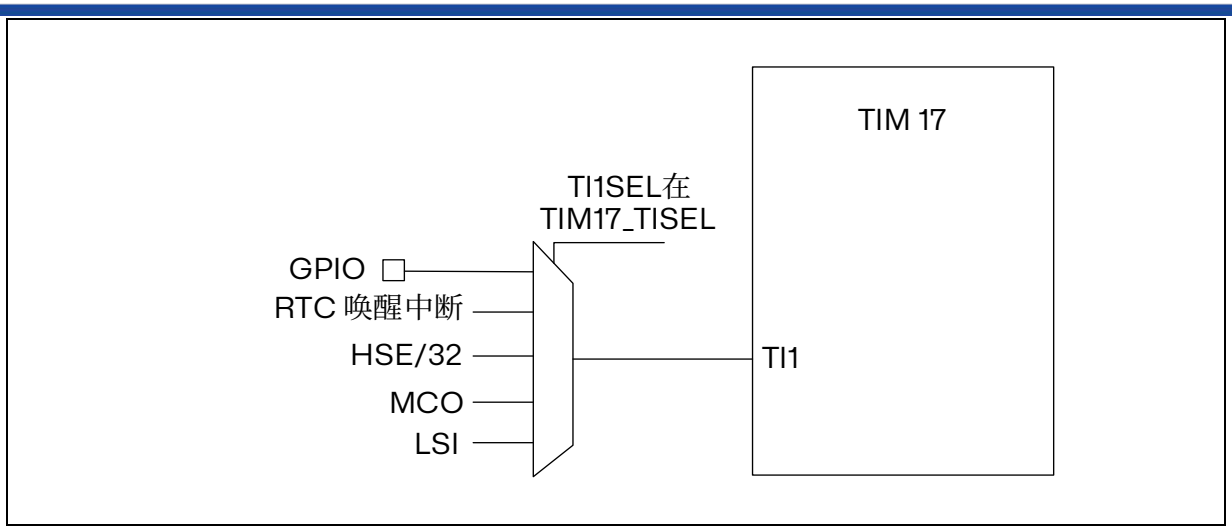



图 8.6 使用 TIM17 捕获模式进行频率测量

定时器 17 的输入捕获通道可以是一个 GPIO 引脚或 MCU 的内部时钟。可能性如下:

- TIM17 通道 1 连接到 GPIO。请参考设备数据手册中的复用功能映射。
- TIM17 通道 1 连接到 RTC 唤醒中断信号。在这种情况下，应使能 RTC 中断。
- TIM17 通道 1 连接到 HSE/32 时钟。
- TIM17 通道 1 连接到 MCO。
- TIM17 通道 1 连接到 LSI 时钟。

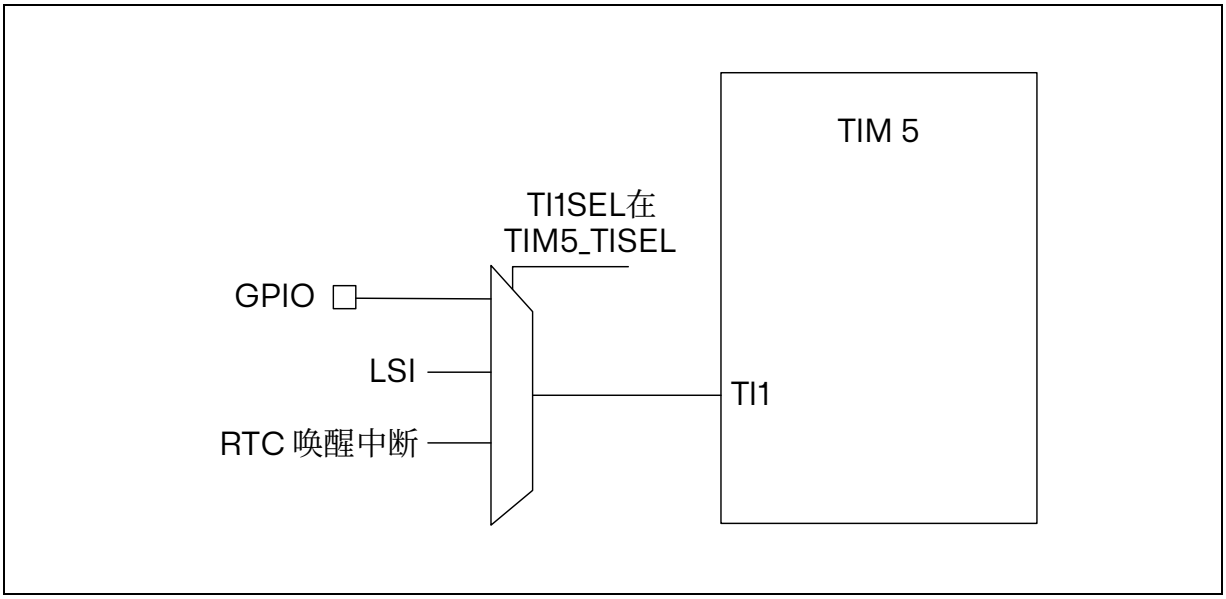



图 8.7 使用 TIM5 捕获模式进行频率测量

定时器 5 的输入捕获通道可以是一个 GPIO 引脚或 MCU 的内部时钟。可能性如下:

- TIM5 通道 1 连接到 GPIO。请参考设备数据手册中的复用功能映射。
- TIM5 通道 1 连接到 LSI 时钟。
- TIM5 通道 1 连接到 RTC 唤醒中断信号。在这种情况下，应使能 RTC 中断。

### 8.2.13 外设时钟使能寄存器（RCC_AHBxENR, RCC_APBxENRy）

每个外设时钟都可以通过 RCC_AHBxENR、RCC_APBxENRy 寄存器中的 xxxxEN 位来使能。

当外设时钟未激活时，不支持对外设寄存器的读或写访问。

使能位具有同步机制，以用来为外设创建无抖动时钟。设置使能位后，时钟在激活前有 2 个时钟周期的延迟。

**注：**在使能外设时钟后，软件必须等待一段延迟时间，然后才能访问外设寄存器。

## 8.3 低功耗模式

- AHB 和 APB 外设时钟（包括 DMA 时钟）可通过软件禁用。
- 睡眠模式会关闭 CPU 时钟。
- 停止模式（停止 0 和停止 1）会关闭除 LSI 以外所有时钟。
- 待机模式会关闭除 LSI 以外所有时钟。

可以通过设置 DBGMCU_CR 寄存器中的 DBG_STOP 或 DBG_STANDBY 位来覆盖 CPU 的深度睡眠模式，以便进行调试。

当退出停止模式（停止 0、停止 1 或待机）时，系统时钟为 HSI16。

如果正在进行 Flash 存储器编程操作，则停止模式、待机模式的进入会延迟，直到 Flash 存储器接口访问完成。如果正在访问 APB 域，则停止模式、待机模式的进入会延迟，直到 APB 访问完成。

## 8.4 RCC 寄存器

### 8.4.1 时钟控制寄存器（RCC_CR）

偏移地址：0x00
复位值：0x0000 0500
HSEBYP 不受复位影响
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | PLL RDY | PLLON | Reserved | | | HSE BIAS | CSSON | HSE BYP | HSE RDY | HSEON |
| | | | | | | r | rw | | | | rw | rs | rw | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | HSI RDY | Res | HSION | Reserved | | | | | | | |
| | | | | | r | | rw | | | | | | | | |

Bit 31:26 保留，必须保持复位值。

Bit 25 **PLLRDY**：主 PLL 时钟就绪标志

由硬件设置，表示主 PLL 已锁定。

0：PLL 未锁定

1：PLL 已锁定

Bit 24 **PLLON**：主 PLL 使能

由软件设置和清除以启用主 PLL。

在进入停止和待机模式时由硬件清除。如果 PLL 时钟用作系统时钟，则无法复位此位。

0：PLL 关闭

1：PLL 使能

Bit 23:21 保留，必须保持复位值。

Bit 20 **HSEBIAS**：HSE 起振电阻控制

0：断开 HSE 起振电阻（默认）

1：连接 HSE 起振电阻

*注：*HSEBIAS=1 *连接起振电阻必须在* HSEON=1 *外部高速晶振使能之前设置。*

Bit 19 **CSSON**：时钟安全系统使能

由软件设置以启用时钟安全系统。当 CSSON 设置时，如果 HSE 振荡器就绪，则硬件使能时钟检测器，如果检测到 HSE 时钟故障，则硬件禁用时钟检测器。此位仅可设置，复位时清除。

0：时钟安全系统关闭（时钟检测器关闭）

1：时钟安全系统使能（如果 HSE 振荡器稳定，则时钟检测器开启；否则关闭）

Bit 18 **HSEBYP**：HSE 晶体振荡器旁路

由软件设置和清除以使用外部时钟旁路振荡器。外部时钟必须在 HSEON 位设置后才能被设备使用。仅当 HSE 振荡器禁用时，才可以写入 HSEBYP 位。

0：HSE 晶体振荡器未旁路

1：HSE 晶体振荡器使用外部时钟旁路

Bit 17 **HSERDY**：HSE 时钟就绪标志

由硬件设置，表示 HSE 振荡器已稳定。

0：HSE 振荡器未就绪

1：HSE 振荡器就绪

*注：一旦* HSEON *位清除，* HSERDY *将在* 6 *个* HSE *时钟周期后变为低电平。*

Bit 16 **HSEON**：HSE 时钟使能

由软件设置和清除。

在进入停止和待机模式时由硬件清除以停止 HSE 振荡器。如果 HSE 振荡器直接或间接用作系统时钟，则无法复位此位。

0：HSE 振荡器关闭

1：HSE 振荡器使能

Bit 15:11 保留，必须保持复位值。

Bit 10 **HSIRDY**：HSI16 时钟就绪标志

由硬件设置，表示 HSI16 振荡器已稳定。此位仅在通过设置 HSION 由软件使能 HSI16 时设置。

0：HSI16 振荡器未就绪

1：HSI16 振荡器已就绪

*注：一旦* HSION *位清除，* HSIRDY *将在* 6 *个* HSI16 *时钟周期后变为低电平。*

Bit 9 保留，必须保持复位值。

Bit 8 **HSION**：HSI16 时钟使能

由软件设置和清除。

在进入停止和待机模式时由硬件清除以停止 HSI16 振荡器。

在离开停止模式或者 HSE 晶体振荡器发生故障，由硬件设置强制 HSI16 振荡器开启。

如果 HSI16 直接或间接用作系统时钟，则由硬件设置此位。

0：HSI16 振荡器关闭

1：HSI16 振荡器使能

Bit 7:0 保留，必须保持复位值。

### 8.4.2 内部时钟源校准寄存器（RCC_ICSCR）

偏移地址：0x04
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | HSICAL[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 |
| Reserved | | | | | | | | | | | | | | | |

Bit 31:24 保留，必须保持复位值。

Bit 23:16 **HSICAL[7:0]**：HSI16 时钟校准

这些位在启动时会初始化出厂编程设定的 HSI16 校准微调值。

Bit 15:0 保留，必须保持复位值。

### 8.4.3 时钟配置寄存器（RCC_CFGR）

偏移地址：0x08
复位值：0x0000 0005
访问：0≤等待周期≤2，支持字，半字和字节访问
仅在时钟源切换期间插入 1 或 2 个等待状态；
若访问发生在 APB 或 AHB 预分频器值更新过程中，则插入 0 至 15 个等待状态。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | MCOPRE[2:0] | | | MCOSEL[3:0] | | | | Reserved | | | | CANSEL | CANPRE[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 |
| Reserved | 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 31 保留，必须保持复位值。

Bit 30:28 **MCOPRE[2:0]**：微控制器时钟输出预分频

这些位由软件设置和清除。
强烈建议在使能 MCO 输出之前更改此预分频器的值。
000：MCO 不分频
001：MCO 分频为 2
010：MCO 分频为 4
011：MCO 分频为 8
100：MCO 分频为 16
其他值：不允许

Bit 27:24 **MCOSEL[3:0]**：微控制器时钟输出选择

这些位由软件设置和清除。
0000：MCO 输出禁用，MCO 无时钟
0001：选择 SYSCLK 系统时钟
0010：保留
0011：选择 HSI16 时钟

0100：选择 HSE 时钟
0101：选择 PLL 时钟
0110：选择 LSI 时钟
其他：保留

*注：在启动或* MCO *时钟源切换期间，此时钟输出可能会有一些截断周期。*

Bit 23:20 保留，必须保持复位值。

Bit 19 **CANSEL**：CAN 总线时钟来源选择
0：PCLK1
1：CANCLK

Bit 18:16 **CANPRE[2:0]**：CANCLK 时钟预分频
000：CAN 时钟 2 分频
001：CAN 时钟 4 分频
010：CAN 时钟 8 分频
011：CAN 时钟 16 分频
100：CAN 时钟 64 分频
101：CAN 时钟 128 分频
110：CAN 时钟 256 分频
111：CAN 时钟 512 分频

Bit 15:14 保留，必须保持复位值。

Bit 13:11 **PPRE2[2:0]**：APB2 预分频
由软件设置和清除，用于控制 APB2 时钟（PCLK2）的分频系数。
0xx：HCLK 不分频
100：HCLK 2 分频
101：HCLK 4 分频
110：HCLK 8 分频
111：HCLK 16 分频

Bit 10:8 **PPRE1[2:0]**：APB1 预分频
由软件设置和清除，用于控制 APB1 时钟（PCLK1）的分频系数。
0xx：HCLK 不分频
100：HCLK 2 分频
101：HCLK 4 分频
110：HCLK 8 分频
111：HCLK 16 分频

Bit 7:4 **HPRE[3:0]**：AHB 预分频
由软件设置和清除，用于控制 AHB 时钟的分频系数。
0xxx：SYSCLK 不分频
1000：SYSCLK 2 分频
1001：SYSCLK 4 分频
1010：SYSCLK 8 分频
1011：SYSCLK 16 分频
1100：SYSCLK 64 分频
1101：SYSCLK 128 分频

1110：SYSCLK 256 分频

1111：SYSCLK 512 分频

Bit 3:2 **SWS[1:0]**：系统时钟切换状态

由硬件设置和清除，以指示哪个时钟源被用作系统时钟。

00：LSI 用作系统时钟

01：HSI16 用作系统时钟

10：HSE 用作系统时钟

11：PLL 用作系统时钟

Bit 1:0 **SW[1:0]**：系统时钟切换

由软件设置和清除，以选择系统时钟源（SYSCLK）。

在退出停止和待机模式或 HSE 振荡器故障时，由硬件配置以强制选择 HSI16 振荡器。

00：选择 LSI 作为系统时钟

01：选择 HSI16 作为系统时钟

10：选择 HSE 作为系统时钟

11：选择 PLL 作为系统时钟

### 8.4.4 PLL 配置寄存器（RCC_PLLCFGR）

偏移地址：0x0C

复位值：0x0000 1000

访问：无等待，支持字，半字和字节访问

此寄存器用于根据以下公式配置 PLL 时钟输出：

- f(VCO 时钟) = f(PLL 时钟输入) × (PLLN)
- f(PLL_P) = f(VCO 时钟) / PLLP
- f(PLL_Q) = f(VCO 时钟) / PLLQ
- f(PLL_R) = f(VCO 时钟) / PLLR

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLPDIV[4:0] | | | | | PLLR[1:0] | | PLL REN | Res | PLLQ[1:0] | | PLL QEN | Reserved | | PLLP | PLL PEN |
| 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 | PLLN[6:0] | | | | | | | Reserved | | | | | | PLLSRC[1:0] | |
| | rw | rw | rw | rw | rw | rw | rw | | | | | | | rw | rw |

Bit 31:27 **PLLPDIV[4:0]**：主 PLL“P”分频系数

由软件设置和清除，用于控制 PLL“P”频率。PLL“P”输出时钟频率= VCO 频率/ PLLPDIV。

00000：PLL“P”时钟由位 PLLP 控制

00001：保留

00010：PLL“P”时钟 = VCO / 2

...

11111：PLL“P”时钟 = VCO / 31

Bit 26:25 **PLLR[1:0]**: 主 PLL“R”时钟分频系数（系统时钟）

由软件设置和清除，用于控制主 PLL 输出时钟 PLLCLK 的频率。该输出可选择为系统时钟。仅当 PLL 禁用时，这些位才可写。

PLL“R”输出时钟频率 = VCO 频率/ PLLR，其中 PLLR 可以是 2、4、6 或 8。

00：PLLR = 2

01：PLLR = 4

10：PLLR = 6

11：PLLR = 8

*注：设置这些位时，需确保此域的频率不超过*192MHz。

Bit 24 **PLLREN**：PLL“R”时钟输出使能

由软件设置和清除，用于使能 PLL 的 PLL“R”时钟输出（用作系统时钟）。

当 PLL 的 PLL“R”时钟输出用作系统时钟时，此位不可写。

为节省功耗，当不使用 PLL 的 PLL“R”时钟输出时，应将 PLLREN 设为 0。

0：PLL“R”时钟输出禁用

1：PLL“R”时钟输出使能

Bit 23 保留，必须保持复位值。

Bit 22:21 **PLLQ[1:0]**：主 PLL“Q”时钟分频系数

由软件设置和清除，用于控制主 PLL 输出时钟 PLL“Q”的频率。该输出可选择用于 USB（48 MHz 时钟）。仅当 PLL 禁用时，这些位才可写。

PLL“Q”输出时钟频率 = VCO 频率/ PLLQ，其中 PLLQ 可以是 2、4、6 或 8。

00：PLLQ = 2

01：PLLQ = 4

10：PLLQ = 6

11：PLLQ = 8

*注：设置这些位时，需确保此域的频率不超过*192MHz。

Bit 20 **PLLQEN**：主 PLL“Q”时钟输出使能

由软件设置和清除，用于使能 PLL 的 PLL“Q”时钟输出。

为节省功耗，当不使用 PLL 的 PLL“Q”时钟输出时，应将 PLLQEN 设为 0。

0：PLL“Q”时钟输出禁用

1：PLL“Q”时钟输出使能

Bit 19:18 保留，必须保持复位值。

Bit 17 **PLLP**：主 PLL“P”时钟分频系数

由软件设置和清除，用于控制主 PLL 输出时钟 PLL“P”的频率。仅当 PLL 禁用时，此位才可写。

当 PLLPDIV[4:0]设为“00000”时，PLL“P”输出时钟频率 = VCO 频率/ PLLP，其中 PLLP 可为 7 或 17

0：PLLP = 7

1：PLLP = 17

*注：设置这些位时，需确保此域的频率不超过*192MHz。

Bit 16 **PLLPEN**：主 PLL“P”时钟输出使能

由软件设置和清除，用于使能 PLL 的 PLL“P”时钟输出。

为节省功耗，当不使用 PLL 的 PLL“P”时钟输出时，应将 PLLPEN 设为 0。

0：PLL“P”时钟输出禁用

1：PLL“P”时钟输出使能

Bit 15 保留，必须保持复位值.

Bit 14:8 **PLLN[6:0]**：VCO 主 PLL 倍频系数

由软件设置和清除，用于控制 VCO 的倍频系数。仅当 PLL 禁用时，这些位才可写。

VCO 输出频率 = VCO 输入频率 × PLLN，其中 6 ≤ PLLN ≤ 25

0000110：6x

0000111：7x

0001000：8x

...

0010111：23x

0011000：24x

0011001：25x

其他：6x

*注：VCO 的输出频率不能超过* 400MHz。

Bit 7:2 保留，必须保持复位值.

Bit 1:0 **PLLSRC[1:0]**：主 PLL 输入时钟源

由软件设置和清除，用于选择 PLL 时钟源。仅当 PLL 禁用时，这些位才可写。

为节省功耗，当不使用任何 PLL 时，应将 PLLSRC 设为 00。

00：无时钟信号发送到 PLL

01：无时钟信号发送到 PLL

10：选择 HSI16 作为 PLL 时钟源

11：选择 HSE 作为 PLL 时钟源

### 8.4.5 定时器时钟配置寄存器（RCC_CFGR1）

偏移地址：0x10

复位值：0x0000 0000

访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | PPRE2_TIM | | Reserved | | PPRE1_TIM | |
| | | | | | | | | | | rw | rw | | | rw | rw |

Bit 5:4 **PPRE2_TIM[1:0]**：APB2_TIM 倍频（APB2_TIM）

由软件来控制 APB2_TIM 时钟的倍频系数。

00：PCLK2 1x（如果 PPRE2 不分频）或 PCKL2 2x（如果 PPRE2 ≥ 2 分频）

01：PCLK2 4x（需要 PPRE2 ≥ 4 分频，否则无效为 2x）

10：PCLK2 8x（需要 PPRE2 ≥ 8 分频，否则无效为 2x）

11：PCLK2 16x（需要 PPRE2 ≥ 16 分频，否则无效为 2x）

Bit 1:0 **PPRE1_TIM[1:0]**：APB1_TIM 倍频（APB1_TIM）

由软件来控制 APB1_TIM 时钟的倍频系数。

00：PCLK1 1x（如果 PPRE1 不分频）或 PCKL1 2x（如果 PPRE1 ≥ 2 分频）

01：PCLK1 4x（需要 PPRE1 ≥ 4 分频，否则无效为 2x）

10：PCLK1 8x（需要 PPRE1 ≥ 8 分频，否则无效为 2x）

11：PCLK1 16x（需要 PPRE1 ≥ 16 分频，否则无效为 2x）

### 8.4.6 时钟中断使能寄存器（RCC_CIER）

偏移地址：0x18

复位值：0x0000 0000

访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | PLL RDYIE | HSE RDYIE | HSI RDYIE | Reserved | | LSI RDYIE |
| | | | | | | | | | | rw | rw | rw | | | rw |

Bit 31:6 保留，必须保持为复位值。

Bit 5 **PLLRDYIE**：PLL 就绪中断使能

由软件设置和清除，以使能/禁用由 PLL 锁定引起的中断。

0：PLL 锁定中断禁用

1：PLL 锁定中断使能

Bit 4 **HSERDYIE**：HSE 就绪中断使能

由软件设置和清除，以使能/禁用由 HSE 振荡器稳定引起的中断。

0：HSE 就绪中断禁用

1：HSE 就绪中断使能

Bit 3 **HSIRDYIE**：HSI16 就绪中断使能

由软件设置和清除，以使能/禁用由 HSI16 振荡器稳定引起的中断。

0：HSI16 就绪中断禁用

1：HSI16 就绪中断使能

Bit 2:1 保留，必须保持为复位值。

Bit 0 **LSIRDYIE**：LSI 就绪中断使能

由软件设置和清除，以使能/禁用由 LSI 振荡器稳定引起的中断。

0：LSI 就绪中断禁用

1：LSI 就绪中断使能

### 8.4.7 时钟中断标志寄存器（RCC_CIFR）

偏移地址：0x1C
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | CSSF | Reserved | | PLL RDYF | HSE RDYF | HSI RDYF | Reserved | | LSI RDYF |
| | | | | | | | r | | | r | r | r | | | r |

Bit 31:9 保留，必须保持为复位值。

Bit 8 **CSSF**：时钟安全系统中断标志
当检测到 HSE 振荡器故障时，由硬件设置。
通过软件设置 CSSC 位来清除。
0：无 HSE 时钟故障引起的时钟安全中断
1：HSE 时钟故障引起的时钟安全中断

Bit 7:6 保留，必须保持为复位值。

Bit 5 **PLLRDYF**：PLL 就绪中断标志
当 PLL 锁定且 PLLRDYDIE 被设置时，由硬件设置。
通过软件设置 PLLRDYC 位来清除。
0：无 PLL 锁定引起的时钟就绪中断
1：PLL 锁定引起的时钟就绪中断

Bit 4 **HSERDYF**：HSE 就绪中断标志
当 HSE 时钟稳定且 HSERDYDIE 被设置时，由硬件设置。
通过软件设置 HSERDYC 位来清除。
0：无 HSE 振荡器引起的时钟就绪中断
1：HSE 振荡器引起的时钟就绪中断

Bit 3 **HSIRDYF**：HSI 就绪中断标志
当 HSI16 时钟稳定且 HSIRDYDIE 被设置时（响应 HSION 的设置，参见时钟控制寄存器（RCC_CR)），由硬件设置。若 HSION 未设置但 HSI16 振荡器通过外设的时钟请求被使能，则此位不被设置且不产生中断。
通过软件设置 HSIRDYC 位来清除。
0：无 HSI16 振荡器引起的时钟就绪中断
1：HSI16 振荡器引起的时钟就绪中断

Bit 2:1 保留，必须保持为复位值。

Bit 0 **LSIRDYF**：LSI 就绪中断标志
当 LSI 时钟稳定且 LSIRDYDIE 被设置时，由硬件设置。
通过软件设置 LSIRDYC 位来清除。
0：无 LSI 振荡器引起的时钟就绪中断
1：LSI 振荡器引起的时钟就绪中断

### 8.4.8 时钟中断清除寄存器（RCC_CICR）

偏移地址：0x20
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | CSSC | Reserved | | PLL RDYC | HSE RDYC | HSI RDYC | Reserved | | LSI RDYC |
| | | | | | | | w | | | w | w | w | | | w |

Bit 31:9 保留，必须保持为复位值。

Bit 8 **CSSC**：时钟安全系统中断清除
此位由软件设置以清除 CSSF 标志。
0：无影响
1：清除 CSSF 标志

Bit 7:6 保留，必须保持为复位值。

Bit 5 **PLLRDYC**：PLL 就绪中断清除
此位由软件设置以清除 PLLRDYF 标志。
0：无影响
1：清除 PLLRDYF 标志

Bit 4 **HSERDYC**: HSE 就绪中断清除
此位由软件设置以清除 HSERDYF 标志。
0：无影响
1：清除 HSERDYF 标志

Bit 3 **HSIRDYC**: HSI16 就绪中断清除
此位由软件设置以清除 HSIRDYF 标志。
0：无影响
1：清除 HSIRDYF 标志

Bit 2:1 保留，必须保持为复位值。

Bit 0 **LSIRDYC**：LSI 就绪中断清除
此位由软件设置以清除 LSIRDYF 标志。
0：无影响
1：清除 LSIRDYF 标志

### 8.4.9 AHB1 外设复位寄存器（RCC_AHB1RSTR）

偏移地址：0x28

复位值：0x0000 0000

访问：无等待，支持字，半字和字节访问

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CRC RST | Reserved | | | | | | | FMAC RST | CORDIC RST | DMAMUX1 RST | DMA2 RST | DMA1 RST |
| | | | rw | | | | | | | | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持为复位值。

Bit 12 **CRCRST**：CRC 复位

由软件设置和清除。

0：无影响

1：复位 CRC

Bit 11:5 保留，必须保持为复位值。

Bit 4 **FMACRST**：FMAC 复位

由软件设置和清除。

0：无影响

1：复位 FMAC

Bit 3 **CORDICRST**：CORDIC 复位

由软件设置和清除。

0：无影响

1：复位 CORDIC

Bit 2 **DMAMUX1RST**：DMAMUX1 复位

由软件设置和清除。

0：无影响

1：复位 DMAMUX1

Bit 1 **DMA2RST**：DMA2 复位

由软件设置和清除。

0：无影响

1：复位 DMA2

Bit 0 **DMA1RST**：DMA1 复位

由软件设置和清除。

0：无影响

1：复位 DMA1

### 8.4.10 AHB2 外设复位寄存器（RCC_AHB2RSTR）

偏移地址：0x2C
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | DAC3RST | DAC2RST | DAC1RST |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC3RST | ADC2RST | ADC1RST | Reserved | | | | | | | GPIOFRST | GPIOERST | GPIODRST | GPIOCRST | GPIOBRST | GPIOARST |
| rw | rw | rw | | | | | | | | rw | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持为复位值。

Bit 18 **DAC3RST**：DAC3 复位
由软件设置和清除。
0：无影响
1：复位 DAC3

Bit 17 **DAC2RST**：DAC2 复位
由软件设置和清除。
0：无影响
1：复位 DAC2

Bit 16 **DAC1RST**：DAC1 复位
由软件设置和清除。
0：无影响
1：复位 DAC1

Bit 15 **ADC3RST**：ADC3 复位
由软件设置和清除。
0：无影响
1：复位 ADC3

Bit 14 **ADC2RST**：ADC2 复位
由软件设置和清除。
0：无影响
1：复位 ADC2

Bit 13 **ADC1RST**：ADC1 复位
由软件设置和清除。
0：无影响
1：复位 ADC1

Bit 12:6 保留，必须保持为复位值。

Bit 5 **GPIOFRST**：GPIO 端口 F 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 F

Bit 4 **GPIOERST**：GPIO 端口 E 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 E

Bit 3 **GPIODRST**：GPIO 端口 D 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 D

Bit 2 **GPIOCRST**：GPIO 端口 C 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 C

Bit 1 **GPIOBRST**：GPIO 端口 B 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 B

Bit 0 **GPIOARST**：GPIO 端口 A 复位
由软件设置和清除。
0：无影响
1：复位 GPIO 端口 A

### 8.4.11 APB1 外设复位寄存器 1（RCC_APB1RSTR1）

偏移地址：0x38
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CLC RST | PWR RST | Reserved | | CAN RST | Res | USB RST | I2C2 RST | I2C1 RST | UART5 RST | UART4 RST | UART3 RST | UART2 RST | Res |
| | | 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 | SPI2 RST | Reserved | | | | | | | | TIM7 RST | TIM6 RST | TIM5 RST | TIM4 RST | TIM3 RST | TIM2 RST |
| | rw | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bit 31:30 保留，必须保持为复位值。

Bit 29 **CLCRST**：CLC 复位
由软件设置和清除。
0：无影响
1：复位 CLC

Bit 28 **PWRRST**：电源接口复位
由软件设置和清除。
0：无影响
1：复位 PWR

Bit 27:26 保留，必须保持为复位值。

Bit 25 **CANRST**：CAN 复位

由软件设置和清除。

0：无影响

1：复位 CAN

Bit 24 保留，必须保持为复位值。

Bit 23 **USBRST**：USB 复位

由软件设置和清除。

0：无影响

1：复位 USB

Bit 22 **I2C2RST**：I2C2 复位

由软件设置和清除。

0：无影响

1：复位 I2C2

Bit 21 **I2C1RST**：I2C1 复位

由软件设置和清除。

0：无影响

1：复位 I2C1

Bit 20 **UART5RST**：UART5 复位

由软件设置和清除。

0：无影响

1：复位 UART5

Bit 19 **UART4RST**：UART4 复位

由软件设置和清除。

0：无影响

1：复位 UART4

Bit 18 **UART3RST**：UART3 复位

由软件设置和清除。

0：无影响

1：复位 UART3

Bit 17 **UART2RST**：UART2 复位

由软件设置和清除。

0：无影响

1：复位 UART2

Bit 16:15 保留，必须保持为复位值。

Bit 14 **SPI2RST**：SPI2 复位

由软件设置和清除。

0：无影响

1：复位 SPI2

Bit 13:6 保留，必须保持为复位值。

Bit 5 **TIM7RST**：TIM7 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM7

Bit 4 **TIM6RST**：TIM6 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM6

Bit 3 **TIM5RST**：TIM5 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM5

Bit 2 **TIM4RST**：TIM4 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM4

Bit 1 **TIM3RST**：TIM3 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM3

Bit 0 **TIM2RST**：TIM2 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM2

### 8.4.12 APB2 外设复位寄存器（RCC_APB2RSTR）

偏移地址：0x40
复位值：0x0000 0000
访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | HRTIM1 RST | Reserved | | | | | | | TIM17 RST | TIM16 RST | TIM15 RST |
| | | | | | rw | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | UART1 RST | TIM8 RST | SPI1 RST | TIM1 RST | Reserved | | | | | | | | | | SYSCFG RST |
| | rw | rw | rw | rw | | | | | | | | | | | rw |

Bit 31:27 保留，必须保持为复位值。

Bit 26 **HRTIM1RST**：HRTIM1 复位
由软件设置和清除。
0：无影响
1：复位 HRTIM1

Bit 25:19 保留，必须保持为复位值。

Bit 18 **TIM17RST**：TIM17 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM17 定时器

Bit 17 **TIM16RST**：TIM16 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM16 定时器

Bit 16 **TIM15RST**：TIM15 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM15 定时器

Bit 15 保留，必须保持为复位值。

Bit 14 **UART1RST**：UART1 复位
由软件设置和清除。
0：无影响
1：复位 UART1

Bit 13 **TIM8RST**：TIM8 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM8 定时器

Bit 12 **SPI1RST**：SPI1 复位
由软件设置和清除。
0：无影响
1：复位 SPI1

Bit 11 **TIM1RST**：TIM1 定时器复位
由软件设置和清除。
0：无影响
1：复位 TIM1 定时器

Bit 10:1 保留，必须保持为复位值。

Bit 0 **SYSCFGRST**：SYSCFG + COMP + OPAMP + VREFBUF 复位
0：无影响
1：复位 SYSCFG、COMP、OPAMP 和 VREFBUF

### 8.4.13 AHB1 外设时钟使能寄存器（RCC_AHB1ENR）

偏移地址：0x48

复位值：0x0000 0100

访问：无等待，支持字，半字和字节访问

*注：当外设时钟未激活时，不支持对外设寄存器的读或写访问。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | CRC EN | Reserved | | | FLASH EN | Reserved | | | FMAC EN | CORDIC EN | DMAMUX1 EN | DMA2 EN | DMA1 EN |
| | | | rw | | | | rw | | | | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持为复位值。

Bit 12 **CRCEN**：CRC 时钟使能

由软件设置和清除。

0：CRC 时钟禁用

1：CRC 时钟使能

Bit 11:9 保留，必须保持为复位值。

Bit 8 **FLASHEN**：闪存接口时钟使能

由软件设置和清除。此位仅在 Flash 处于断电模式时可以被禁用。

0：闪存接口时钟禁用

1：闪存接口时钟使能

Bit 7:5 保留，必须保持为复位值。

Bit 4 **FMACEN**：FMAC 使能

由软件设置和清除。

0：FMAC 时钟禁用

1：FMAC 时钟使能

Bit 3 **CORDICEN**：CORDIC 时钟使能

由软件设置和清除。

0：CORDIC 时钟禁用

1：CORDIC 时钟使能

Bit 2 **DMAMUX1EN**：DMAMUX1 时钟使能

由软件设置和清除。

0：DMAMUX1 时钟禁用

1：DMAMUX1 时钟使能

Bit 1 **DMA2EN**：DMA2 时钟使能

由软件设置和清除。

0：DMA2 时钟禁用

1：DMA2 时钟使能

Bit 0 **DMA1EN**：DMA1 时钟使能

由软件设置和清除。

0：DMA1 时钟禁用

1：DMA1 时钟使能

### 8.4.14 AHB2 外设时钟使能寄存器（RCC_AHB2ENR）

偏移地址：0x4C

复位值：0x0000 0000

访问：无等待，支持字，半字和字节访问

*注：当外设时钟未激活时，不支持对外设寄存器的读或写访问。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | DAC3EN | DAC2EN | DAC1EN |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC3EN | ADC2EN | ADC1EN | Reserved | | | | | | | GPIOFEN | GPIOEEN | GPIODEN | GPIOCEN | GPIOBEN | GPIOAEN |
| rw | rw | rw | | | | | | | | rw | rw | rw | rw | rw | rw |

Bit 31:19 保留，必须保持为复位值。

Bit 18 **DAC3EN**：DAC3 时钟使能

由软件设置和清除。

0：DAC3 时钟禁用

1：DAC3 时钟使能

Bit 17 **DAC2EN**：DAC2 时钟使能

由软件设置和清除。

0：DAC2 时钟禁用

1：DAC2 时钟使能

Bit 16 **DAC1EN**：DAC1 时钟使能

由软件设置和清除。

0：DAC1 时钟禁用

1：DAC1 时钟使能

Bit 15 **ADC3EN**：ADC3 时钟使能

由软件设置和清除。

0：ADC3 时钟禁用

1：ADC3 时钟使能

Bit 14 **ADC2EN**：ADC2 时钟使能

由软件设置和清除。

0：ADC2 时钟禁用

1：ADC2 时钟使能

Bit 13 **ADC1EN**：ADC1 时钟使能

由软件设置和清除。

0：ADC1 时钟禁用

1：ADC1 时钟使能

Bit 12:6 保留，必须保持为复位值。

Bit 5 **GPIOFEN**：GPIO 端口 F 时钟使能
由软件设置和清除。
0：IO 端口 F 时钟禁用
1：IO 端口 F 时钟使能

Bit 4 **GPIOEEN**：GPIO 端口 E 时钟使能
由软件设置和清除。
0：IO 端口 E 时钟禁用
1：IO 端口 E 时钟使能

Bit 3 **GPIODEN**：GPIO 端口 D 时钟使能
由软件设置和清除。
0：IO 端口 D 时钟禁用
1：IO 端口 D 时钟使能

Bit 2 **GPIOCEN**：GPIO 端口 C 时钟使能
由软件设置和清除。
0：IO 端口 C 时钟禁用
1：IO 端口 C 时钟使能

Bit 1 **GPIOBEN**：GPIO 端口 B 时钟使能
由软件设置和清除。
0：IO 端口 B 时钟禁用
1：IO 端口 B 时钟使能

Bit 0 **GPIOAEN**：GPIO 端口 A 时钟使能
由软件设置和清除。
0：IO 端口 A 时钟禁用
1：IO 端口 A 时钟使能

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

偏移地址：0x58
复位值：0x0000 0400
访问：无等待，支持字，半字和字节访问

*注：当外设时钟未激活时，不支持对外设寄存器的读或写访问。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CLC EN | PWR EN | BKP EN | Res | CAN EN | Res | USB EN | I2C2 EN | I2C1 EN | UART5 EN | UART4 EN | UART3 EN | UART2 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | SPI2 EN | Reserved | | WWDG EN | Reserved | | | | | TIM7 EN | TIM6 EN | TIM5 EN | TIM4 EN | TIM3 EN | TIM2 EN |
| | rw | | | rs | | | | | | rw | rw | rw | rw | rw | rw |

Bit 31:30 保留，必须保持为复位值。

Bit 29 **CLCEN**：CLC 时钟使能
由软件设置和清除。
0：CLC 时钟禁用
1：CLC 时钟使能

Bit 28 **PWREN**：电源接口时钟使能
由软件设置和清除。
0：电源接口时钟禁用
1：电源接口时钟使能

Bit 27 **BKPEN**：备份接口时钟使能
由软件设置和清除。
0：备份接口时钟禁用
1：备份接口时钟使能

Bit 26 保留，必须保持为复位值。

Bit 25 **CANEN**：CAN 时钟使能
由软件设置和清除。
0：CAN 时钟禁用
1：CAN 时钟使能

Bit 24 保留，必须保持为复位值。

Bit 23 **USBEN**：USB 时钟使能
由软件设置和清除。
0：USB 时钟禁用
1：USB 时钟使能

Bit 22 **I2C2EN**：I2C2 时钟使能
由软件设置和清除。
0：I2C2 时钟禁用
1：I2C2 时钟使能

Bit 21 **I2C1EN**：I2C1 时钟使能
由软件设置和清除。
0：I2C1 时钟禁用
1：I2C1 时钟使能

Bit 20 **UART5EN**：UART5 时钟使能
由软件设置和清除。
0：UART5 时钟禁用
1：UART5 时钟使能

Bit 19 **UART4EN**：UART4 时钟使能
由软件设置和清除。
0：UART4 时钟禁用
1：UART4 时钟使能

Bit 18 **UART3EN**：UART3 时钟使能
由软件设置和清除。
0：UART3 时钟禁用

1：UART3 时钟使能

Bit 17 **UART2EN**：UART2 时钟使能

由软件设置和清除。

0：UART2 时钟禁用

1：UART2 时钟使能

Bit 16:15 保留，必须保持为复位值。

Bit 14 **SPI2EN**：SPI2 时钟使能

由软件设置和清除。

0：SPI2 时钟禁用

1：SPI2 时钟使能

Bit 13:12 保留，必须保持为复位值。

Bit 11 **WWDGEN**：窗口看门狗时钟使能

由软件设置以使能窗口看门狗时钟。由硬件系统复位清除。

如果 WWDG_SW 选项位被清除，该位也可以由硬件设置。

0：窗口看门狗时钟禁用

1：窗口看门狗时钟使能

Bit 10:6 保留，必须保持为复位值。

Bit 5 **TIM7EN**：TIM7 定时器时钟使能

由软件设置和清除。

0：TIM7 时钟禁用

1：TIM7 时钟使能

Bit 4 **TIM6EN**：TIM6 定时器时钟使能

由软件设置和清除。

0：TIM6 时钟禁用

1：TIM6 时钟使能

Bit 3 **TIM5EN**：TIM5 定时器时钟使能

由软件设置和清除。

0：TIM5 时钟禁用

1：TIM5 时钟使能

Bit 2 **TIM4EN**：TIM4 定时器时钟使能

由软件设置和清除。

0：TIM4 时钟禁用

1：TIM4 时钟使能

Bit 1 **TIM3EN**：TIM3 定时器时钟使能

由软件设置和清除。

0：TIM3 时钟禁用

1：TIM3 时钟使能

Bit 0 **TIM2EN**：TIM2 定时器时钟使能

由软件设置和清除。

0：TIM2 时钟禁用

1：TIM2 时钟使能

### 8.4.16 APB2 外设时钟使能寄存器（RCC_APB2ENR）

偏移地址：0x60
复位值：0x0000 0000
访问：支持字，半字和字节访问

*注：当外设时钟未激活时，不支持对外设寄存器的读或写访问。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | HRTIM1EN | Reserved | | | | | | | TIM17EN | TIM16EN | TIM15EN |
| | | | | | rw | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | UART1EN | TIM8EN | SPI1EN | TIM1EN | Reserved | | | | | | | | | | SYSCFGEN |
| | rw | rw | rw | rw | | | | | | | | | | | rw |

Bit 31:27 保留，必须保持为复位值。

Bit 26 **HRTIM1EN**：HRTIM1 时钟使能
由软件设置和清除。
0：HRTIM1 时钟禁用
1：HRTIM1 时钟使能

Bit 25:19 保留，必须保持为复位值。

Bit 18 **TIM17EN**：TIM17 定时器时钟使能
由软件设置和清除。
0：TIM17 定时器时钟禁用
1：TIM17 定时器时钟使能

Bit 17 **TIM16EN**：TIM16 定时器时钟使能
由软件设置和清除。
0：TIM16 定时器时钟禁用
1：TIM16 定时器时钟使能

Bit 16 **TIM15EN**：TIM15 定时器时钟使能
由软件设置和清除。
0：TIM15 定时器时钟禁用
1：TIM15 定时器时钟使能

Bit 15 保留，必须保持为复位值。

Bit 14 **UART1EN**：UART1 时钟使能
由软件设置和清除。
0：UART1 时钟禁用
1：UART1 时钟使能

Bit 13 **TIM8EN**：TIM8 定时器时钟使能
由软件设置和清除。
0：TIM8 定时器时钟禁用
1：TIM8 定时器时钟使能

Bit 12 **SPI1EN**：SPI1 时钟使能
由软件设置和清除。

0：SPI1 时钟禁用

1：SPI1 时钟使能

Bit 11 **TIM1EN**：TIM1 定时器时钟使能

由软件设置和清除。

0：TIM1 定时器时钟禁用

1：TIM1 定时器时钟使能

Bit 10:1 保留，必须保持为复位值。

Bit 0 **SYSCFGEN**：SYSCFG + COMP + VREFBUF + OPAMP 时钟使能

由软件设置和清除。

0：SYSCFG + COMP + VREFBUF + OPAMP 时钟禁用

1：SYSCFG + COMP + VREFBUF + OPAMP 时钟使能

### 8.4.17 独立外设时钟配置寄存器（RCC_CCIPR）

偏移地址：0x88

复位值：0x0000 0000

访问：无等待，支持字，半字和字节访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ADCSEL[1:0] | | Reserved | | | | | | | | | | | |
| | | rw | rw | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

Bit 31:30 保留，必须保持为复位值。

Bit 29:28 **ADCSEL[1:0]**：ADC1/2/3 时钟源选择

这些位由软件设置和清除，用于选择 ADC1/2/3 接口所使用的时钟源。

00：未选择时钟

01：选择 PLL“P”时钟作为 ADC1/2/3 的时钟

10：选择系统时钟作为 ADC1/2/3 的时钟

11：保留

Bit 27:0 保留，必须保持为复位值。

### 8.4.18 RTC 域控制寄存器（RCC_BDCR）

偏移地址：0x90
复位值：0x0000 0000
访问：0 ≤ 等待状态 ≤ 3，支持字、半字和字节访问。
对该寄存器的连续访问会插入等待状态。

*注：此寄存器的位不在 $V_{CORE}$ 域内。因此，复位后，这些位受写保护，必须先设置 PWR 电源控制寄存器 1（PWR_CR1）中的 DBP 位，才能对这些位进行修改。这些位（BDRST 除外）仅在 RTC 域复位后复位（见章节：RTC 域复位）。任何内部或外部复位对这些位均无效。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | BDRST |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTCEN | Reserved | | | | | RTCSEL[1:0] | | Reserved | | | | | | | |
| rw | | | | | | rw | rw | | | | | | | | |

Bit 31:17 保留，必须保持为复位值。

Bit 16 **BDRST**：RTC 域软件复位
由软件设置和清除。
0：未激活复位
1：复位整个 RTC 域

Bit 15 **RTCEN**：RTC 时钟使能
由软件设置和清除。
0：禁用 RTC 时钟
1：使能 RTC 时钟

Bit 14:10 保留，必须保持为复位值。

Bit 9:8 **RTCSEL[1:0]**：RTC 时钟源选择
由软件设置，以选择 RTC 的时钟源。一旦选择了 RTC 时钟源，除非 RTC 域被复位，否则无法更改。可以使用 BDRST 位来复位它们。
00：无时钟
01：无时钟
10：使用 LSI 振荡器时钟作为 RTC 时钟
11：使用分频 128 后的 HSE 振荡器时钟作为 RTC 时钟

Bit 7:0 保留，必须保持为复位值。

### 8.4.19 控制/状态寄存器（RCC_CSR）

偏移地址：0x94
复位值：0x0C00 0000
由系统复位重置，但仅由电源复位重置复位标志。
访问：0 ≤ 等待状态 ≤3，支持字、半字和字节访问。
对该寄存器的连续访问会插入等待状态。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPWR RSTF | WWDG RSTF | IWDG RSTF | SFT RSTF | POR RSTF | PIN RSTF | OBL RSTF | RMVF | Reserved | | | | | | | |
| r | r | r | r | r | r | r | rw | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | LSIRDY | LSION |
| | | | | | | | | | | | | | | r | rw |

Bit 31 **LPWRRSTF**：低功耗复位标志
当因非法进入停止和待机模式而发生复位时，由硬件设置。
通过写入 RMVF 位清除。
0：未发生非法模式复位
1：发生非法模式复位

Bit 30 **WWDGRSTF**：窗口看门狗复位标志
当发生窗口看门狗复位时，由硬件设置。
通过写入 RMVF 位清除。
0：未发生窗口看门狗复位
1：发生窗口看门狗复位

Bit 29 **IWDGRSTF**：独立窗口看门狗复位标志
当发生独立看门狗复位域时，由硬件设置。
通过写入 RMVF 位清除。
0：未发生独立看门狗复位
1：发生独立看门狗复位

Bit 28 **SFTRSTF**：软件复位标志
当发生软件复位时，由硬件设置。
通过写入 RMVF 位清除。
0：未发生软件复位
1：发生软件复位

Bit 27 **PORRSTF**：上电/掉电复位标志
当发生上电/掉电复位时，由硬件设置。
通过写入 RMVF 位清除。
0：未发生上电/掉电复位
1：发生上电/掉电复位

Bit 26 **PINRSTF**：引脚复位标志

当 NRST 引脚发生复位时，由硬件设置。

通过写入 RMVF 位清除。

0：NRST 引脚未发生复位

1：NRST 引脚发生复位

Bit 25 **OBLRSTF**：选项字节加载复位标志

当因选项字节加载而发生复位时，由硬件设置。

通过写入 RMVF 位清除。

0：未因选项字节加载而发生复位

1：因选项字节加载而发生复位

Bit 24 **RMVF**：移除复位标志

通过软件设置以清除复位标志。

0：无影响

1：清除复位标志

Bit 23:2 保留，必须保持为复位值。

Bit 1 **LSIRDY**：LSI 振荡器就绪

由硬件设置和清除，以指示 LSI 振荡器是否稳定。在 LSION 位被清除后，LSIRDY 将在 3 个 LSI 振荡器时钟周期后变为低电平。即使 LSION = 0，如果 LSI 被独立看门狗或 RTC 请求，此位也可以被设置。

0：LSI 振荡器未就绪

1：LSI 振荡器就绪

Bit 0 **LSION**：LSI 振荡器使能

由软件设置和清除。

0：LSI 振荡器关闭

1：LSI 振荡器开启

# 9 通用 I/O（GPIO）

## 9.1 简介

每个通用 I/O 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）和 1 个 16 位复位寄存器（GPIOx_BRR）。此外，所有 GPIO 都包括 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）。

## 9.2 GPIO 主要特征

- 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）输出数据
- 可为每个 I/O 选择不同的速度
- 输入状态：浮空、上拉/下拉、模拟
- 将数据输入到输入数据寄存器（GPIOx_IDR）或外设（复用功能输入）
- 置位和复位寄存器（GPIOx_BSRR），对 GPIOx_ODR 具有按位写权限
- 锁定机制（GPIOx_LCKR），可冻结 I/O 端口配置
- 模拟功能
- 复用功能选择寄存器
- 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 I/O 引脚用作 GPIO 或多种外设功能中的一种

## 9.3 GPIO 功能描述

根据数据手册中列出的每个 I/O 端口的特性，可通过软件将通用 I/O（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
- 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 I/O 端口位均可自由编程，但 I/O 端口寄存器必须按 32 位字、半字或字节进行访问。GPIOx_BSRR 寄存器和 GPIOx_BRR 寄存器旨在实现对 GPIOx_ODR 寄存器进行原子读取/修改访问。这样便可确保在读取和修改访问之间发生中断请求也不会有问题。

下两张图分别显示了标准和 5V 容忍 I/O 端口位的基本结构。下表给出了可能的端口位配置方案。

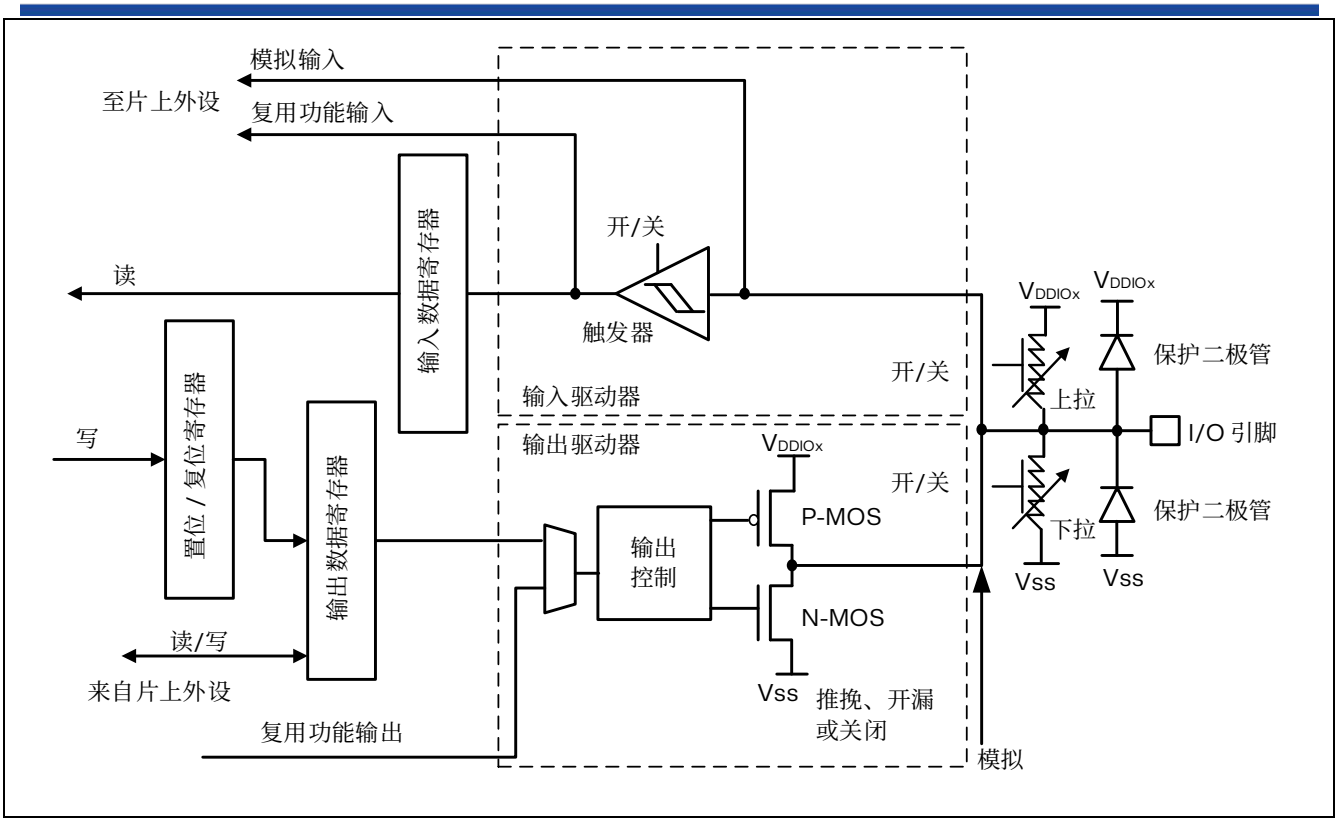
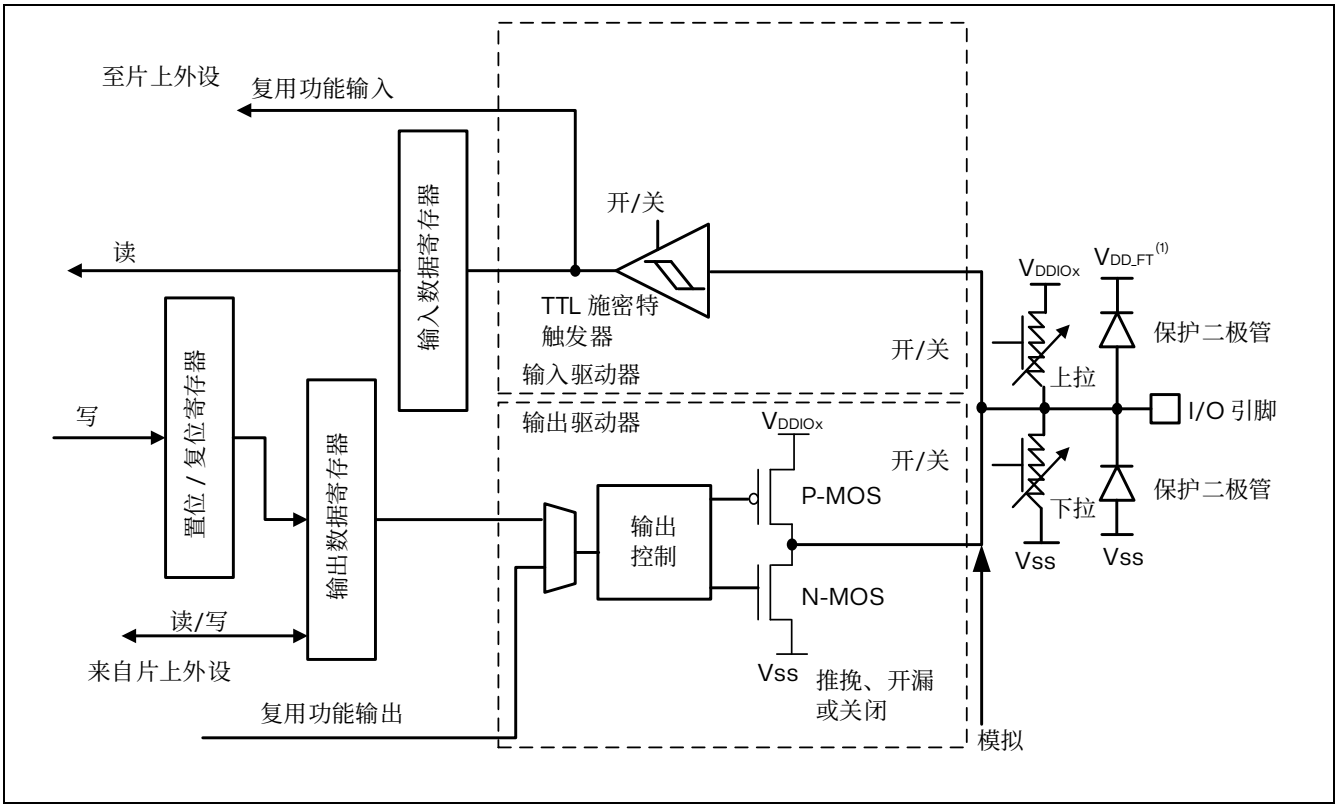



图 9.1 I/O 端口位基本结构




1. VDD_FT 是和 5V 容忍 I/O 相关的电位，与 VDD 不同。

图 9.2 5V 容忍 I/O 端口位的基本结构

表 9.1 端口位配置表 (1)

| MODE[1:0] | OTYPE | OSPEED[1:0] | | PUPD[1:0] | | I/O 配置 | |
|---|---|---|---|---|---|---|---|
| 01 | 0 | SPEED[1:0] | | 0 | 0 | GP 输出 | PP |
| | 0 | | | 0 | 1 | GP 输出 | PP + PU |
| | 0 | | | 1 | 0 | GP 输出 | PP + PD |
| | 0 | | | 1 | 1 | 保留 | |
| | 1 | | | 0 | 0 | GP 输出 | OD |
| | 1 | | | 0 | 1 | GP 输出 | OD + PU |
| | 1 | | | 1 | 0 | GP 输出 | OD + PD |
| | 1 | | | 1 | 1 | 保留（GP 输出 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 | 输入 | 浮空 |
| | x | x | x | 0 | 1 | 输入 | PU |
| | x | x | x | 1 | 0 | 输入 | PD |
| | x | x | x | 1 | 1 | 保留（输入浮空） | |
| 11 | x | x | x | 0 | 0 | 输入/输出 | 模拟 |
| | x | x | x | 0 | 1 | 保留 | |
| | x | x | x | 1 | 0 | | |
| | x | x | x | 1 | 1 | | |

1. GP =通用、PP =推挽、PU =上拉、PD =下拉、OD =开漏、AF =复用功能。

### 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，大多数 I/O 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：

- PA15：JTDI 处于上拉状态
- PA14：JTCK/DBGCLK 处于下拉状态
- PA13：JTMS/DBGIO 处于上拉状态
- PB4：NJTRST 处于上拉状态
- PB3：JTDO 处于浮空状态

当引脚配置为输出后，写入到输出数据寄存器（GPIOx_ODR）的值将在 I/O 引脚上输出。可以在推挽模式下或开漏模式下使用输出驱动器（仅驱动低电平，高电平为高阻态）。

输入数据寄存器（GPIOx_IDR）每个 AHB 时钟周期捕获一次 I/O 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 I/O 引脚复用功能复用器和映射

器件 I/O 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 I/O 引脚。这可以确保共用同一个 I/O 引脚的外设之间不会发生冲突。

每个 I/O 引脚都有一个复用器，该复用器采用多达 16 路复用功能输入（AF0 到 AF15），可通过 GPIOx_AFRL（针对引脚 0 到 7）和 GPIOx_AFRH（针对引脚 8 到 15）寄存器对这些输入进行配置：

- 复位后，复用器选择为复用功能 0（AF0）。在复用模式下通过 GPIOx_MODER 寄存器配置 I/O。
- 器件数据手册中详细说明了每个引脚的特定复用功能分配。

除了这种灵活的 I/O 复用架构之外，各外设还可以将复用功能映射到不同 I/O 引脚，这可以优化小型封装中可用外设的数量。

要在指定配置下使用 I/O，用户必须按照以下步骤操作：

- **调试功能：**每个器件复位后，立即将这些引脚分配为可由调试主机使用的复用功能引脚。
- **GPIO：**在 GPIOx_MODER 寄存器中将所需 I/O 配置为输出、输入或模拟通道。
- **外设复用功能：**
  - 在 GPIOx_AFRL 或 GPIOx_AFRH 寄存器中，将 I/O 连接到所需的 AFx。
  - 通过 GPIOx_OTYPER、GPIOx_PUPDR 和 GPIOx_OSPEEDER 寄存器，分别选择类型、上拉/下拉以及输出速度。
  - 在 GPIOx_MODER 寄存器中将所需 I/O 配置为复用功能。
- **模拟功能：**
  - 对于 ADC、DAC、OPAMP 和 COMP，在 GPIOx_MODER 寄存器中将所需 I/O 配置为模拟模式，并在 ADC、DAC、OPAMP 和 COMP 寄存器中配置所需功能。

    如上所述，对于附加功能（如 DAC 或 OPAMP），输出由相应的外设控制。在使能外设控制寄存器中的附加功能输出之前，必须谨慎选择 I/O 端口的模拟功能。
  - 对于 RTC、WKUPx 和振荡器的其它功能，在相关的 RTC、PWR 和 RCC 寄存器中配置所需功能。这些功能优先于标准 GPIO 寄存器中的配置。

有关复用功能 I/O 引脚映射的详细信息，请参见器件数据手册中的“复用功能映射”表。

### 9.3.3 I/O 端口控制寄存器

每个 GPIO 端口有 4 个 32 位存储器映射的控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR），可配置多达 16 个 I/O。GPIOx_MODER 寄存器用于选择 I/O 模式（输入、输出、复用、模拟）。GPIOx_OTYPER 和 GPIOx_OSPEEDR 寄存器用于选择输出类型（推挽或开漏）和速度。无论采用哪种 I/O 方向，GPIOx_PUPDR 寄存器都用于选择上拉/下拉。

### 9.3.4 I/O 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。通过 I/O 输入的数据存储到输入数据寄存器（GPIOx_IDR）中，它是一个只读寄存器。

有关寄存器说明的详细信息，请参见章节：GPIO 端口输入数据寄存器（GPIOx_IDR）（x = A...F）和 GPIO 端口输出数据寄存器（GPIOx_ODR）（x = A...F）。

### 9.3.5 I/O 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。置位复位寄存器的大小是 GPIOx_ODR 的二倍。

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**复位** 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复位操作，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。

### 9.3.6 I/O 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，必须应用特定的写/读序列。当正确的 LOCK 序列应用到此寄存器的第 16 位后，会使用 LCKR[15:0]的值来锁定 I/O 的配置（在写序列期间，LCKR[15:0]的值必须相同）。将 LOCK 序列应用到某个端口位后，在执行下一次 MCU 复位或外设复位之前，将无法对该端口位的值进行修改。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。

LOCK 序列（参见章节：GPIO 端口配置锁定寄存器（GPIOx_LCKR）（x = A...F）只能通过对 GPIOx_LCKR 寄存器进行字（32 位长）访问的方式来执行，因为 GPIOx_LCKR 的第 16 位必须与[15:0]位同时置位。有关详细信息，请参见章节：GPIO 端口配置锁定寄存器（GPIOx_LCKR）（x = A...F）中的 LCKR 寄存器说明。

### 9.3.7 I/O 复用功能输入/输出

有两个寄存器可用来从每个 I/O 可用的复用功能输入/输出中进行选择。借助这些寄存器，用户可根据应用程序的要求将某个复用功能连接到其它某个引脚。

这意味着可使用 GPIOx_AFRL 和 GPIOx_AFRH 复用功能寄存器在每个 GPIO 上复用多个可用的外设功能。这样一来，应用程序可为每个 I/O 选择任何一个可用功能。由于 AF 选择信号由复用功能输入和复用功能输出共用，所以只需为指定 I/O 的复用功能输入/输出选择一个通道即可。

要了解在每个 GPIO 引脚上复用了哪些功能，请参见器件数据手册。

### 9.3.8 外部中断/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。请参见章节：外部中断和事件控制器（EXTI）。

### 9.3.9 输入配置

当 I/O 端口被配置为输入时：

- 输出缓冲器被禁止
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

下图说明了 I/O 端口位的输入配置。

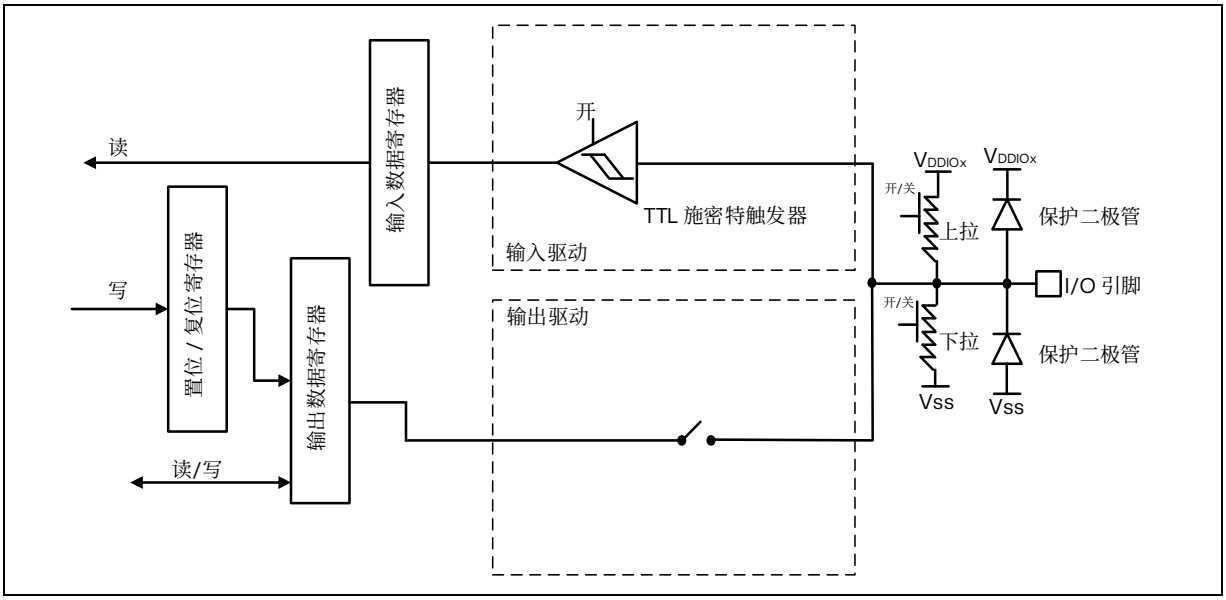



图 9.3 输入浮空/上拉/下拉配置

### 9.3.10 输出模式配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被打开：
  - 开漏模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”会使端口保持高阻态（Hi-Z）（P-MOS 始终不激活）
  - 推挽模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”可激活 P-MOS
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 对输入数据寄存器的读访问可获取 I/O 状态
- 对输出数据寄存器的读访问可获取最后的写入值

下图说明了 I/O 端口位的输出配置。

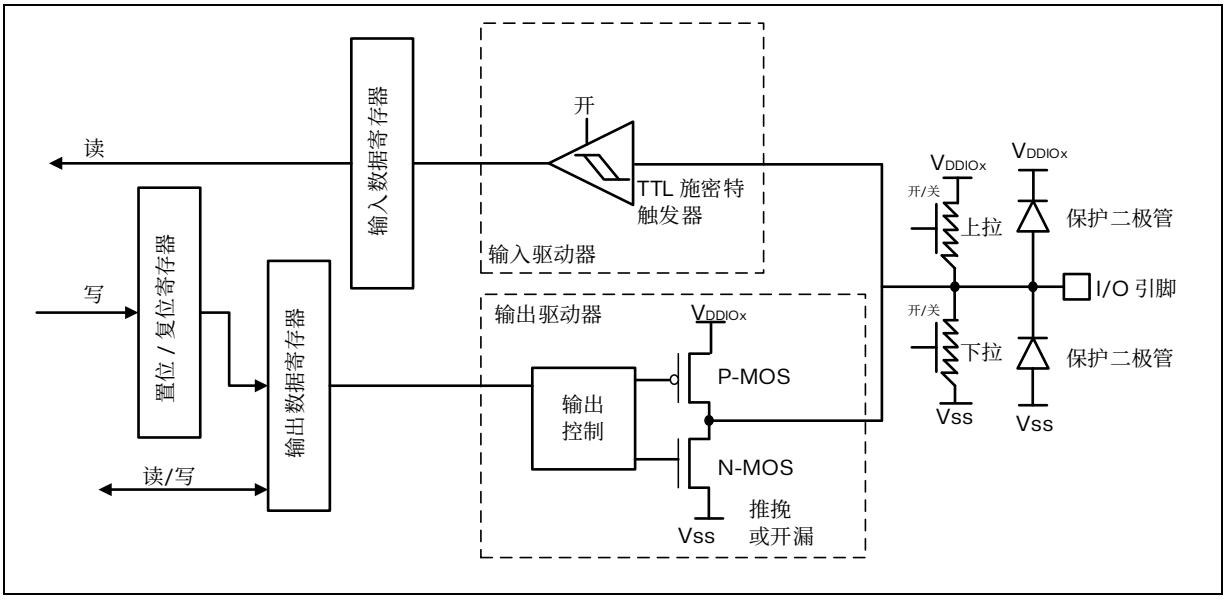



图 9.4 输出配置

### 9.3.11 复用功能配置

当 I/O 端口被配置为复用功能时：

- 可将输出缓冲器配置为开漏或推挽模式
- 输出缓冲器由来自外设的信号驱动（发送器使能和数据）
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

下图说明了 I/O 端口位的复用功能配置。

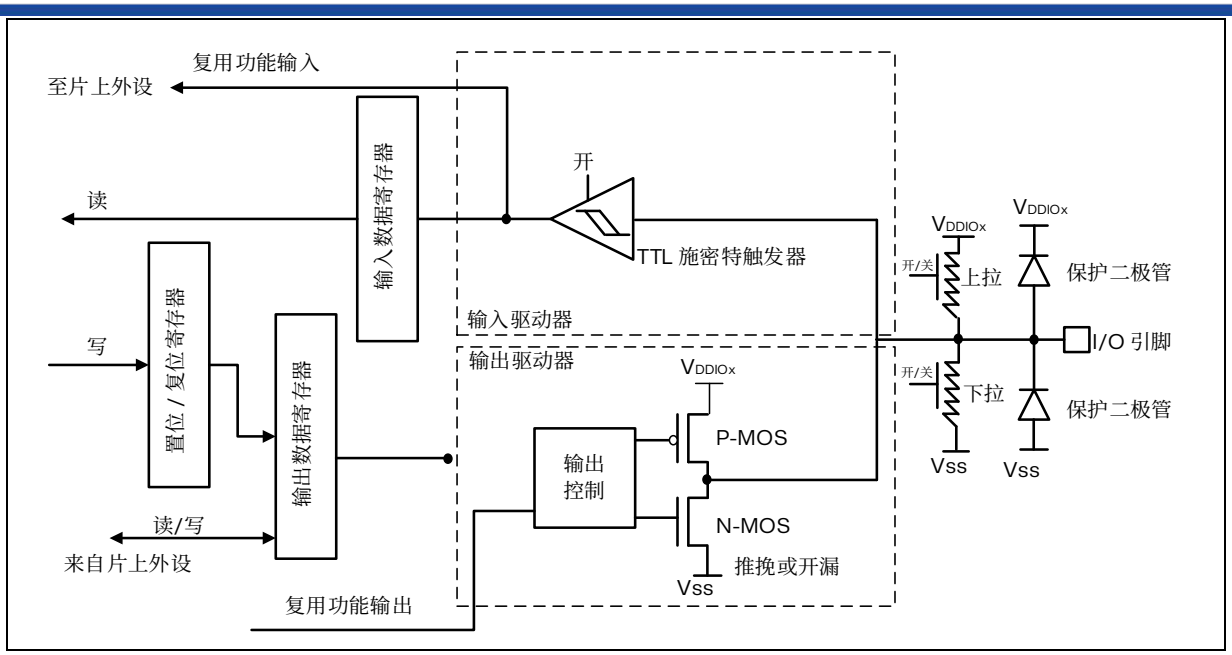



图 9.5 复用功能配置

### 9.3.12 模拟配置

当 I/O 端口配置为模拟时：

- 输出缓冲器被禁止
- 施密特触发器输入停用，I/O 引脚的每个模拟输入的功耗变为零。施密特触发器的输出被强制处理为恒定值（0）
- 弱上拉和下拉电阻被硬件关闭
- 对输入数据寄存器的读访问值为“0”

下图说明了 I/O 端口位的高阻态模拟输入配置。

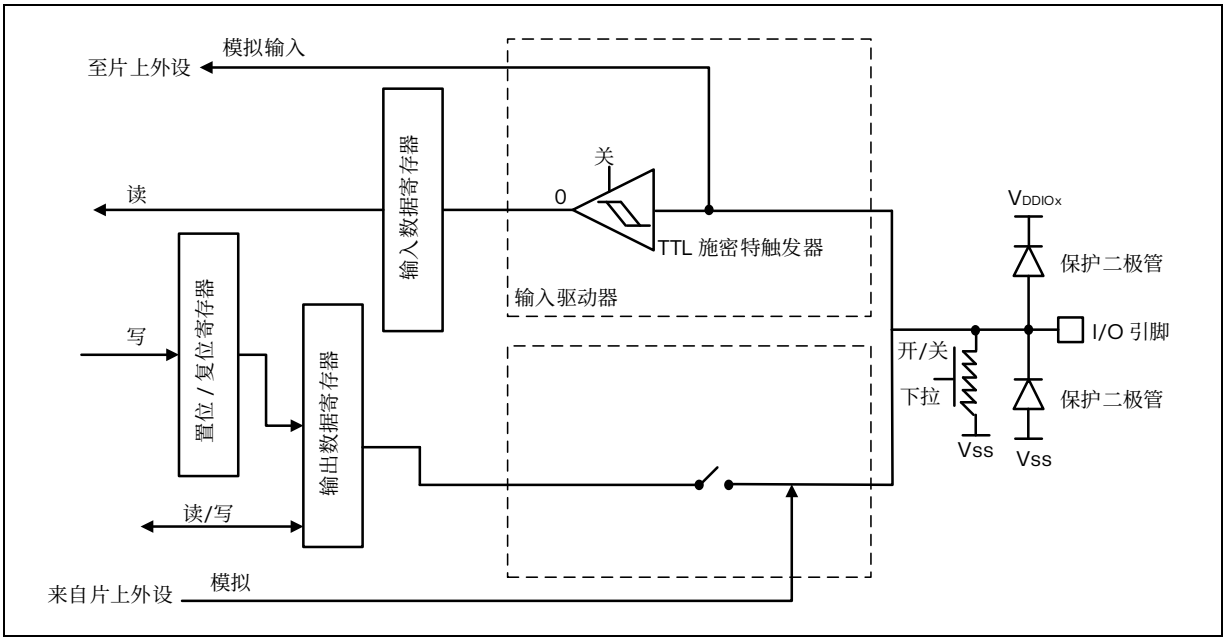



图 9.6 高阻态模拟配置

### 9.3.13 将 HSE 振荡器引脚用作 GPIO

当 HSE 振荡器关闭（复位后的默认状态）时，可将相关的振荡器引脚用作常规 GPIO。

当 HSE 振荡器打开（通过设置 RCC_CR 寄存器中的 HSEON 位）时，振荡器会控制与其相关联的引脚，这些引脚的 GPIO 配置不起作用。

将振荡器配置为用户外部时钟模式时，仅为时钟输入保留 OSC_IN 引脚，OSC_OUT 引脚仍可用作常规 GPIO。

### 9.3.14 使用 PB8 作为 GPIO

PB8 可用作启动引脚（BOOT0）或 GPIO。根据用户选项字节中的 nSWBOOT0 位，它将在输入模式和模拟输入模式之间切换：

- 如果在选项字节加载阶段后 nSWBOOT0 = 1。
- 如果在复位后 nSWBOOT0 = 0。

*注：如果引脚未连接，建议将 PB8 设置为非模拟模式以限制功耗。*

## 9.4 GPIO 寄存器

本节对 GPIO 寄存器进行了详细介绍。
可按字、半字或字节模式写入外设寄存器。

### 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（端口 C...F）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODE15[1:0] | | MODE14[1:0] | | MODE13[1:0] | | MODE12[1:0] | | MODE11[1:0] | | MODE10[1:0] | | MODE9[1:0] | | MODE8[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODE7[1:0] | | MODE6[1:0] | | MODE5[1:0] | | MODE4[1:0] | | MODE3[1:0] | | MODE2[1:0] | | MODE1[1:0] | | MODE0[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **MODE[15:0][1:0]：**端口 x 配置 I/O 引脚 y（y = 0...15）

这些位通过软件写入，用于配置 I/O 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **OT[15:0]：**端口 x 配置 I/O 引脚 y（y = 0...15）

这些位通过软件写入，用于配置 I/O 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 00C0（端口 B）
复位值：0x0000 0000（其他端口）

| 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 31:0 **OSPEED[15:0][1:0]：**端口 x 配置 I/O 引脚 y（y = 0...15）

这些位通过软件写入，用于配置 I/O 输出速度。

00：低速

01：中速

10：高速

11：超高速

### 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：0x0000 0100（端口 B）
复位值：0x0000 0000（其他端口）

| 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] | | PUPD10[1:0] | | PUPD9[1:0] | | PUPD8[1:0] | | PUPD7[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 31:0 **PUPD[15:0][1:0]：**端口 x 配置 I/O 引脚 y（y = 0...15）

这些位通过软件写入，用于配置 I/O 上拉或下拉。

00：无上拉或下拉（浮空）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 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 | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **ID[15:0]：**端口 x 输入数据 I/O 引脚 y（y = 0...15）

这些位为只读。只能在字模式下访问。它们包含相应 I/O 端口的输入值。

### 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **OD[15:0]：**端口 x 输出数据 I/O 引脚 y（y = 0...15）

这些位可通过软件读取和写入。

*注：对于原子置位/复位，通过写入* GPIOx_BSRR *或* GPIOx_BRR *寄存器，可分别置位和/或复位* ODR *位* (x = A...E)。

### 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 | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:16 **BR[15:0]：**端口 x 复位 I/O 引脚 y（y = 0...15）

这些位为只写。读取这些位可返回值 0x0000。

0：对应 ODx 位无操作

1：复位对应的 ODx 位

*注：如果同时对* BSx *和* BRx *置位，则* BSx *的优先级更高。*

Bit 15:0 **BS[15:0]：**端口 x 置位 I/O 引脚 y（y = 0...15）

这些位为只写。读取这些位可返回值 0x0000。

0：对应 ODx 位无操作

1：置位对应的 ODx 位

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

当正确的写序列应用到第 16 位（LCKK）时，此寄存器将用于锁定端口位的配置。bit[15:0]的值用于锁定 GPIO 的配置。在写序列期间，不能更改 LCKR[15:0]的值。将 LOCK 序列应用到某个端口位后，在执行下一次 MCU 复位或外设复位之前，将无法对该端口位的值进行修改。

*注：可使用特定的写序列对*GPIOx_LCKR *寄存器执行写操作。在此锁定序列期间只允许使用字访问（32 位长）。*

每个锁定位冻结一个特定的配置寄存器（控制寄存器和复用功能寄存器）。

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | 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 31:17 保留，必须保持为复位值。

Bit 16 **LCKK：**锁定键

可随时读取此位。可使用锁定键写序列对其进行修改。

0：端口配置锁定键未激活

1：端口配置锁定键已激活。到下一次 MCU 复位或外设复位前，GPIOx_LCKR 寄存器将一直处于锁定状态。

锁定键写序列：

WR LCKR[16] = “1” + LCKR[15:0]

WR LCKR[16] = “0” + LCKR[15:0]

WR LCKR[16] = “1” + LCKR[15:0]

RD LCKR

RD LCKR[16] = “1”（此读操作为可选操作，但它可确认锁定已激活）

*注：在锁定键写序列期间，不能更改*LCK[15:0]*的值。*

*锁定序列中的任何错误都将中止锁定操作。*

*在任一端口位上的第一个锁定序列之后，对*LCKK *位的任何读访问都将返回“1”，直到下一次*MCU *复位或外设复位为止*

Bit 15:0 **LCK[15:0]：**端口 x 锁定 I/O 引脚 y（y = 0...15）

这些位都是读/写位，但只能在 LCKK 位等于“0”时执行写操作。

0：端口配置未锁定

1：端口配置已锁定

### 9.4.9 GPIO 复用功能低位寄存器（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 31:0 **AFSEL[7:0][3:0]：**端口 x 引脚 y 的复用功能选择（y = 0...7）

这些位通过软件写入，用于配置复用功能 I/O。

0000：AF0

0001：AF1

0010：AF2

0011：AF3

0100：AF4

0101：AF5

0110：AF6

0111：AF7

1000：AF8

1001：AF9

1010：AF10

1011：AF11

1100：AF12

1101：AF13

1110：AF14

1111：AF15

### 9.4.10 GPIO 复用功能高位寄存器（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 31:0 **AFSEL[15:8][3:0]：**端口 x 引脚 y 的复用功能选择（y = 8...15）

这些位通过软件写入，用于配置复用功能 I/O。

0000：AF0

0001：AF1

0010：AF2

0011：AF3

0100：AF4

0101：AF5
0110：AF6
0111：AF7
1000：AF8
1001：AF9
1010：AF10
1011：AF11
1100：AF12
1101：AF13
1110：AF14
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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:16 保留，必须保持为复位值

Bit 15:0 **BR[15:0]：**端口 x 复位 I/O 引脚 y（y = 0...15）
这些位只能写。读取这些位将返回值 0x0000。
0：对应 ODx 位无操作
1：复位对应的 ODx 位

# 10 系统配置控制器（SYSCFG）

## 10.1 SYSCFG 主要特征

RX32G410 系列器件配备了一组配置寄存器。系统配置控制器的主要特征如下：

- 管理外部中断线与 GPIOs 的连接
- 管理健壮性功能
- 设置 CCM RAM 软件擦除

## 10.2 SYSCFG 寄存器

### 10.2.1 SYSCFG 外部中断配置寄存器 1（SYSCFG_EXTICR1）

地址偏移：0x08

复位值：0x0000 0000

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

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 0 到 3）

这些位通过软件写入，以选择外部中断/事件检测的 EXTI 输入的源输入。

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

### 10.2.2 SYSCFG 外部中断配置寄存器 2（SYSCFG_EXTICR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 4 到 7）

这些位通过软件写入，以选择外部中断/事件检测的 EXTI 输入的源输入。

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

### 10.2.3 SYSCFG 外部中断配置寄存器 3（SYSCFG_EXTICR3）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **EXTIx[3:0]：**EXTI x 配置（x = 8 到 11）
这些位通过软件写入，以选择外部中断/事件检测的 EXTI 输入的源输入。
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

### 10.2.4 SYSCFG 外部中断配置寄存器 4（SYSCFG_EXTICR4）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **EXTIx[3:0]：**EXTI x 配置（x = 12 到 15）
这些位通过软件写入，以选择外部中断/事件检测的 EXTI 输入的源输入。
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

### 10.2.5 SYSCFG CCM SRAM 控制和状态寄存器（SYSCFG_SCSR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | CCMBSY | CCMER |
| | | | | | | | | | | | | | | r | rw |

Bit 31:2 保留，必须保持为复位值。

Bit 1 **CCMBSY**：CCM SRAM 因擦除操作而繁忙

0：没有正在进行的 CCM SRAM 擦除操作

1：正在进行 CCM SRAM 擦除操作

Bit 0 **CCMER**：CCM SRAM 擦除

设置此位将启动硬件 CCM SRAM 擦除操作。

此位在 CCM SRAM 擦除操作结束时自动清除。

*注：此位具有写保护功能：只有在* SYSCFG_SKR *寄存器中写入正确的密钥序列后，才能设置此位。*

### 10.2.6 SYSCFG 配置寄存器 2（SYSCFG_CFGR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | SPF2 | SPF | Reserved | | | | ECCL | PVDL | SPL | CLL |
| | | | | | | rc_w1 | rc_w1 | | | | | rs | rs | rs | rs |

Bit 31:10 保留，必须保持为复位值。

Bit 9 **SPF2**：CCM SRAM 资料奇偶校验错误标志

当检测到 CCM SRAM 奇偶校验错误时，该位由硬件设置。

通过写入“1”可由软件清除。

0：未检测到奇偶校验错误

1：检测到奇偶校验错误

Bit 8 **SPF**：CCM SRAM 指令奇偶校验错误标志

当检测到 CCM SRAM 奇偶校验错误时，该位由硬件设置。

通过写入“1”可由软件清除。

0：未检测到奇偶校验错误

1：检测到奇偶校验错误

Bit 7:4 保留，必须保持为复位值。

Bit 3 **ECCL**：ECC 锁定

此位由软件设置，并且只能通过系统复位来清除。它可用于使能并锁定 Flash ECC 错

误连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。
0：ECC 错误与 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入断开连接。
1：ECC 错误连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。

Bit 2 **PVDL**：PVD 锁定使能位

此位由软件设置，并且只能通过系统复位来清除。它可用于使能并锁定 PVD 连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入，以及 PWR_CR2 寄存器中的 PVDE 和 PLS[2:0]。
0：PVD 中断与 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入断开连接。PVDE 和 PLS[2:0]位可由应用程序编程。
1：PVD 中断连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入，PVDE 和 PLS[2:0]位为只读。

Bit 1 **SPL**：CCM SRAM 奇偶校验锁定位

此位由软件设置，并且只能通过系统复位来清除。它可用于使能并锁定 CCM SRAM 奇偶校验错误信号连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。
0：CCM SRAM 奇偶校验错误信号与 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入断开连接。
1：CCM SRAM 奇偶校验错误信号连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。

Bit 0 **CLL**：内核锁住信号输出使能位

此位由软件设置，并且只能通过系统复位来清除。它可用于使能并将内核锁住信号输出连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。
0：内核锁住输出与 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入断开连接。
1：内核锁住输出连接到 TIM1/8/15/16/17 刹车输入和 HRTIM1 的 hrtim_sys_flt 输入。

### 10.2.7 SYSCFG CCM SRAM 密钥寄存器 （SYSCFG_SKR）

地址偏移：0x24
复位值：0x0000 0000

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

Bit 31:8 保留，必须保持为复位值。

Bit 7:0 **KEY[7:0]**：用于软件擦除的 CCM SRAM 写保护密钥

要解锁 SYSCFG_SCSR 寄存器中的 CCMER 位的写保护，需要执行以下步骤：
1、将“0xCA”写入 Key[7:0]
2、将“0x53”写入 Key[7:0]

*注：写入错误的密钥将重新激活写保护。*

# 11 外部中断和事件控制器（EXTI）

## 11.1 简介

EXTI 的主要特征包括：

- 可生成多达 22 个事件/中断请求
  - 21 个可配置线
  - 1 个直接线
- 每个事件/中断线具有独立屏蔽功能
- 可配置上升沿或下降沿触发（仅限可配置线）
- 专用状态位（仅限可配置线）
- 模拟事件/中断请求（仅限可配置线）

## 11.2 EXTI 主要特性

外部中断和事件控制器（EXTI）负责管理外部和内部异步事件/中断，并向 CPU/中断控制器生成事件请求，以及向电源控制器生成唤醒请求。

EXTI 能够管理多达 22 条可从停止模式唤醒的事件线。

这些事件线分为可配置型和直接型：

- 可配置型事件线：可独立选择激活边沿，并通过状态标志指示中断来源。此类事件线主要用于 I/O 的外部中断以及部分外设。
- 直接型事件线：由某些外设用于生成从停止状态唤醒的事件或中断。状态标志由外设提供。

每条线均可独立屏蔽，以用于中断或事件生成。

此外，该控制器还允许通过向专用寄存器写入数据，以软件方式模拟事件或中断，并与相应的硬件事件线进行多路复用。

## 11.3 EXTI 功能介绍

对于可配置的中断线，需要对其进行配置并使能才能生成中断。这通过编程两个触发寄存器以设置所需的边沿检测，并通过在中断屏蔽寄存器的相应位写入 1 来使能中断请求来实现。当中断线上出现所选边沿时，将生成一个中断请求，并且与中断线对应的挂起位也会被设置。通过向挂起寄存器中写入 1，可以清除此请求。

直接中断线其中断在中断屏蔽寄存器中默认是使能的，并且在挂起寄存器中没有与之对应的挂起位。

为了生成事件，需要配置并使能事件线。这同样通过编程两个触发寄存器以设置所需的边沿检测，并通过在事件屏蔽寄存器的相应位写入 1 来使能事件请求来实现。当事件线上出现所选边沿时，将生成一个事件脉冲，但与事件线对应的挂起位不会被设置。

对于可配置线路，还可以通过在软件中断/事件寄存器中写入 1 来由软件生成中断/事件请求。

*注：与直接线路相关联的中断或事件仅在系统处于停止模式时被触发。如果系统仍在运行，* EXTI *不会生成任何中断/ 事件。*

### 11.3.1 EXTI 框图

外部中断/事件框图如下图所示。

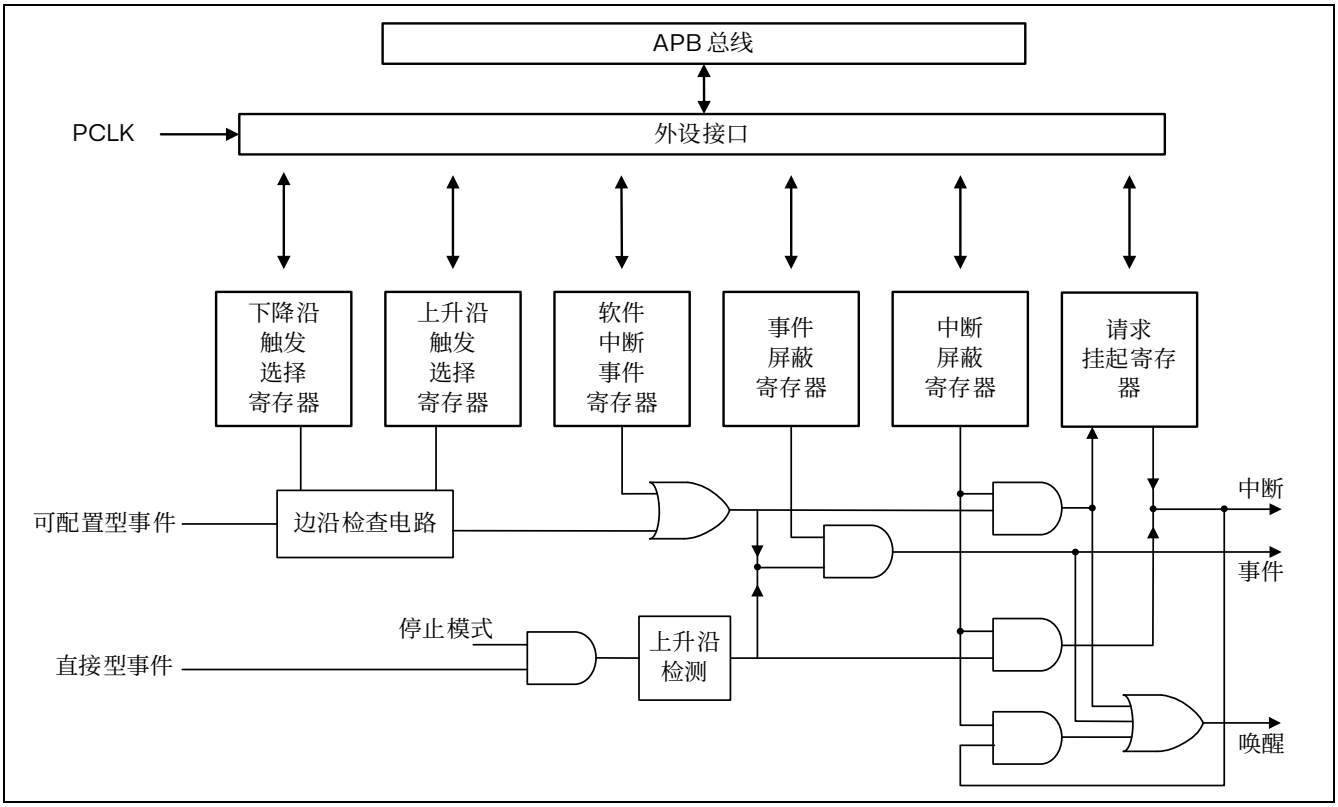



图 11.1 可配置的中断/事件框图

### 11.3.2 外设异步中断

某些外设能够在系统运行模式和停止模式下生成事件，从而允许从停止模式唤醒系统。为实现这一点，外设会生成一个与系统时钟（如 APB 时钟）同步的事件版本以及一个异步的事件版本。这个异步事件会连接到一个 EXTI 直接线路。

*注：少数具有从停止模式唤醒能力的外设会连接到一个* EXTI *可配置线路。在这种情况下，需要进行* EXTI *配置以允许从停止模式唤醒。*

### 11.3.3 硬件中断选择

要将某条线路配置为中断源，请按照以下步骤操作：

1. 配置 EXTI_IMR 寄存器中的相应屏蔽位。
2. 配置中断线路的触发选择位（EXTI_RTSR 和 EXTI_FTSR）。
3. 配置控制映射到 EXTI 的中断通道的使能和屏蔽位，以便能够正确响应来自 EXTI 线路的中断请求。

*注：直接线路不需要任何* EXTI *配置。*

### 11.3.4 硬件事件选择

要将某条线路配置为事件源，请按照以下步骤操作：

1. 配置 EXTI_EMR 寄存器中的相应屏蔽位。
2. 配置事件线路的触发选择位（EXTI_RTSR 和 EXTI_FTSR）。

### 11.3.5 软件中断/事件选择

任何可配置线路都可以配置为软件中断/事件线路。生成软件中断的步骤如下：

1. 配置相应的屏蔽位（EXTI_IMR，EXTI_EMR）。
2. 设置软件中断寄存器（EXTI_SWIER）中的所需位。

## 11.4 EXTI 中断/事件线映射

GPIO 连接到 16 条可配置的中断/事件线（参见下图）。

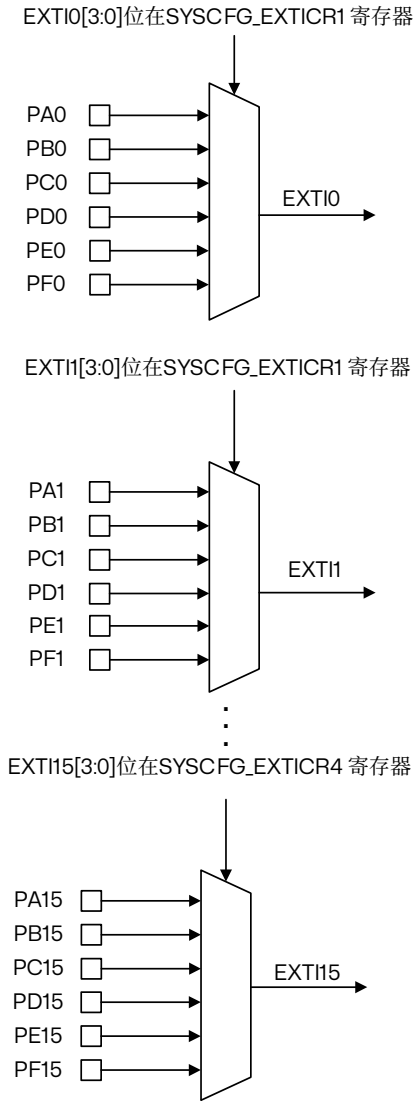



图 11.2 外部中断/事件 GPIO 映射

表 11.1 EXTI 线路连接

| EXTI 线 | 线路源 | 线路类型 |
|---|---|---|
| 0~15 | GPIO | 可配置 |
| 16 | PVD | 可配置 |
| 17 | RTC 闹钟事件 | 可配置 |
| 18 | USB 设备 FS 唤醒事件 | 直接 |
| 19 | 保留 | - |
| 20 | 保留 | - |
| 21 | COMP1 输出 | 可配置 |
| 22 | COMP2 输出 | 可配置 |
| 23~28 | 保留 | - |
| 29 | COMP3 输出 | 可配置 |
| 30 | COMP4 输出 | 可配置 |

## 11.5 EXTI 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器必须按字（32 位）进行访问。

### 11.5.1 中断屏蔽寄存器（EXTI_IMR）

偏移地址：0x00
复位值：0x0004 0000。直接线设为 1，其他线设为 0，参考表 12.1 EXTI 线路连接。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | IM30 | IM29 | Reserved | | | | | | IM22 | IM21 | Reserved | | IM18 | IM17 | IM16 |
| | rw | rw | | | | | | | rw | rw | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IM15 | IM14 | IM13 | IM12 | IM11 | IM10 | IM9 | IM8 | IM7 | IM6 | IM5 | IM4 | IM3 | IM2 | IM1 | IM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **IMx：**线 x 上的中断屏蔽（x = 30~29）
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

Bit 28:23 保留，必须保持为复位值。

Bit 22:21 **IMx：**线 x 上的中断屏蔽（x = 22~21）
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

Bit 20:19 保留，必须保持为复位值。

Bit 18:0 **IMx：**线 x 上的中断屏蔽（x = 18~0）
0：屏蔽来自线 x 上的中断请求
1：开放来自线 x 上的中断请求

*注：直接线的复位值被设置为1，以便默认使能中断。*

### 11.5.2 事件屏蔽寄存器（EXTI_EMR）

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 | EM30 | EM29 | Reserved | | | | | | EM22 | EM21 | Reserved | | EM18 | EM17 | EM16 |
| | rw | rw | | | | | | | rw | rw | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EM15 | EM14 | EM13 | EM12 | EM11 | EM10 | EM9 | EM8 | EM7 | EM6 | EM5 | EM4 | EM3 | EM2 | EM1 | EM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **EMx：**线 x 上的事件屏蔽（x = 30~29）
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

Bit 28:23 保留，必须保持为复位值。

Bit 22:21 **EMx：**线 x 上的事件屏蔽（x = 22~21）

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

Bit 20:19 保留，必须保持为复位值。

Bit 18:0 **EMx：**线 x 上的事件屏蔽（x = 18~0）

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

### 11.5.3 上升沿触发选择寄存器（EXTI_RTSR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | RT30 | RT29 | Reserved | | | | | | RT22 | RT21 | Reserved | | | RT17 | RT16 |
| | rw | rw | | | | | | | rw | rw | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RT15 | RT14 | RT13 | RT12 | RT11 | RT10 | RT9 | RT8 | RT7 | RT6 | RT5 | RT4 | RT3 | RT2 | RT1 | RT0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **RTx：**线 x 上的上升沿触发事件配置位（x = 30~29）

0：禁止输入线 x 上的上升沿触发（中断和事件）

1：使能输入线 x 上的上升沿触发（中断和事件）

Bit 28:23 保留，必须保持为复位值。

Bit 22:21 **RTx：**线 x 的上升触发事件配置位（x = 22~21）

0：禁止输入线 x 上的上升沿触发（中断和事件）

1：使能输入线 x 上的上升沿触发（中断和事件）

Bit 20:18 保留，必须保持为复位值。

Bit 17:0 **RTx：**线 x 的上升触发事件配置位（x = 17~0）

0：禁止输入线 x 上的上升沿触发（中断和事件）

1：使能输入线 x 上的上升沿触发（中断和事件）

*注：外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。在写 EXTI_RTSR 寄存器时，在外部中断线上的上升沿信号不能被识别，挂起位也不会被置位。*

*在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。*

### 11.5.4 下降沿触发选择寄存器（EXTI_FTSR）

偏移地址：0x0C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | FT30 | FT29 | Reserved | | | | | | FT22 | FT21 | Reserved | | | FT17 | FT16 |
| | rw | rw | | | | | | | rw | rw | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FT15 | FT14 | FT13 | FT12 | FT11 | FT10 | FT9 | FT8 | FT7 | FT6 | FT5 | FT4 | FT3 | FT2 | FT1 | FT0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **FTx：**线 x 上的下降沿触发事件配置位（x = 30~29）
0：禁止输入线 x 上的下升沿触发（中断和事件）
1：使能输入线 x 上的下升沿触发（中断和事件）

Bit 28:23 保留，必须保持为复位值。

Bit 22:21 **FTx：**线 x 上的下降沿触发事件配置位（x = 22~21）
0：禁止输入线 x 上的下升沿触发（中断和事件）
1：使能输入线 x 上的下升沿触发（中断和事件）

Bit 20:18 保留，必须保持为复位值。

Bit 17:0 **FTx：**线 x 上的下降沿触发事件配置位（x = 17~0）
0：禁止输入线 x 上的下升沿触发（中断和事件）
1：使能输入线 x 上的下升沿触发（中断和事件）

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

### 11.5.5 软件中断事件寄存器（EXTI_SWIER）

偏移地址：0x10
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | SWI30 | SWI29 | Reserved | | | | | | SWI22 | SWI21 | Reserved | | | SWI17 | SWI16 |
| | rw | rw | | | | | | | rw | rw | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWI15 | SWI14 | SWI13 | SWI12 | SWI11 | SWI10 | SWI9 | SWI8 | SWI7 | SWI6 | SWI5 | SWI4 | SWI3 | SWI2 | SWI1 | SWI0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **SWIx：**线 x 上的软件中断（x = 30~29）
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。
通过清除 EXTI_PR 的对应位（写入“1”），可以清除该位为“0”。

Bit 28:23 保留，必须保持复位值。

Bit 22:21 **SWIx：**线 x 上的软件中断（x = 22~21）

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。

通过清除 EXTI_PR 的对应位（写入“1”），可以清除该位为“0”。

Bit 20:18 保留，必须保持为复位值。

Bit 17:0 **SWIx：**线 x 上的软件中断（x = 17~0）

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。

通过清除 EXTI_PR 的对应位（写入“1”），可以清除该位为“0”。

### 11.5.6 挂起寄存器（EXTI_PR）

偏移地址：0x14

复位值： 0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | PIF30 | PIF29 | Reserved | | | | | | PIF22 | PIF21 | Reserved | | | PIF17 | PIF16 |
| | rc_w1 | rc_w1 | | | | | | | rc_w1 | rc_w1 | | | | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIF15 | PIF14 | PIF13 | PIF12 | PIF11 | PIF10 | PIF9 | PIF8 | PIF7 | PIF6 | PIF5 | PIF4 | PIF3 | PIF2 | PIF1 | PIF0 |
| 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 | rc_w1 | rc_w1 |

Bit 31 保留，必须保持为复位值。

Bit 30:29 **PIFx：**线 x 上挂起的中断标志（x = 30~29）

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”可以清除该位。

Bit 28:23 保留，必须保持复位值。

Bit 22:21 **PIFx：**线 x 上挂起的中断标志（x = 22~21）

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”可以清除该位。

Bit 20:18 保留，必须保持为复位值。

Bit 17:0 **PIFx：**线 x 上挂起的中断标志（x = 17~0）

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”可以清除该位。

# 12 可配置逻辑单元（CLC）

## 12.1 CLC 简介

可配置逻辑单元（Configurable Logic Cell，CLC）模块允许用户将一些信号的组合指定为逻辑功能的输入，并使用逻辑输出来控制其他外设或 I/O 引脚。由于 CLC 模块的操作不受软件执行限制，且支持大量输出设计，因此在嵌入式设计中提供了更大的灵活性和可能性。

CLC 包含 4 个主要部分，如图 13.1 所示。首先，输入数据选择多路开关将输入信号送到 4 个数据门，如图 13.2 所示。然后，4 个数据门中的每个门可以选择 32 个输入信号中的任意信号来传递给逻辑功能，如图 13.3 所示。之后，逻辑功能的输出被送至内部逻辑和外部引脚，并可以产生中断。

CLC 模块的输出可以被送至另一个 CLC 模块的输入，产生更复杂的逻辑功能。

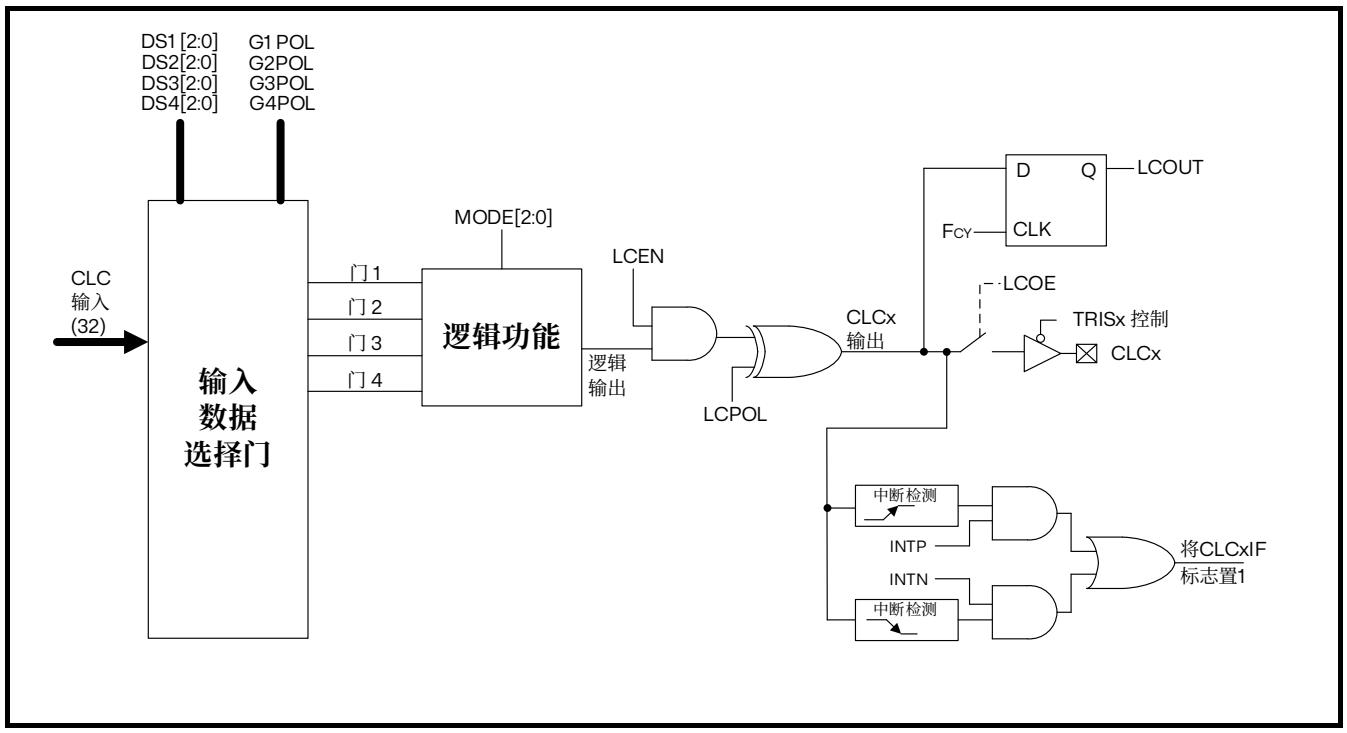
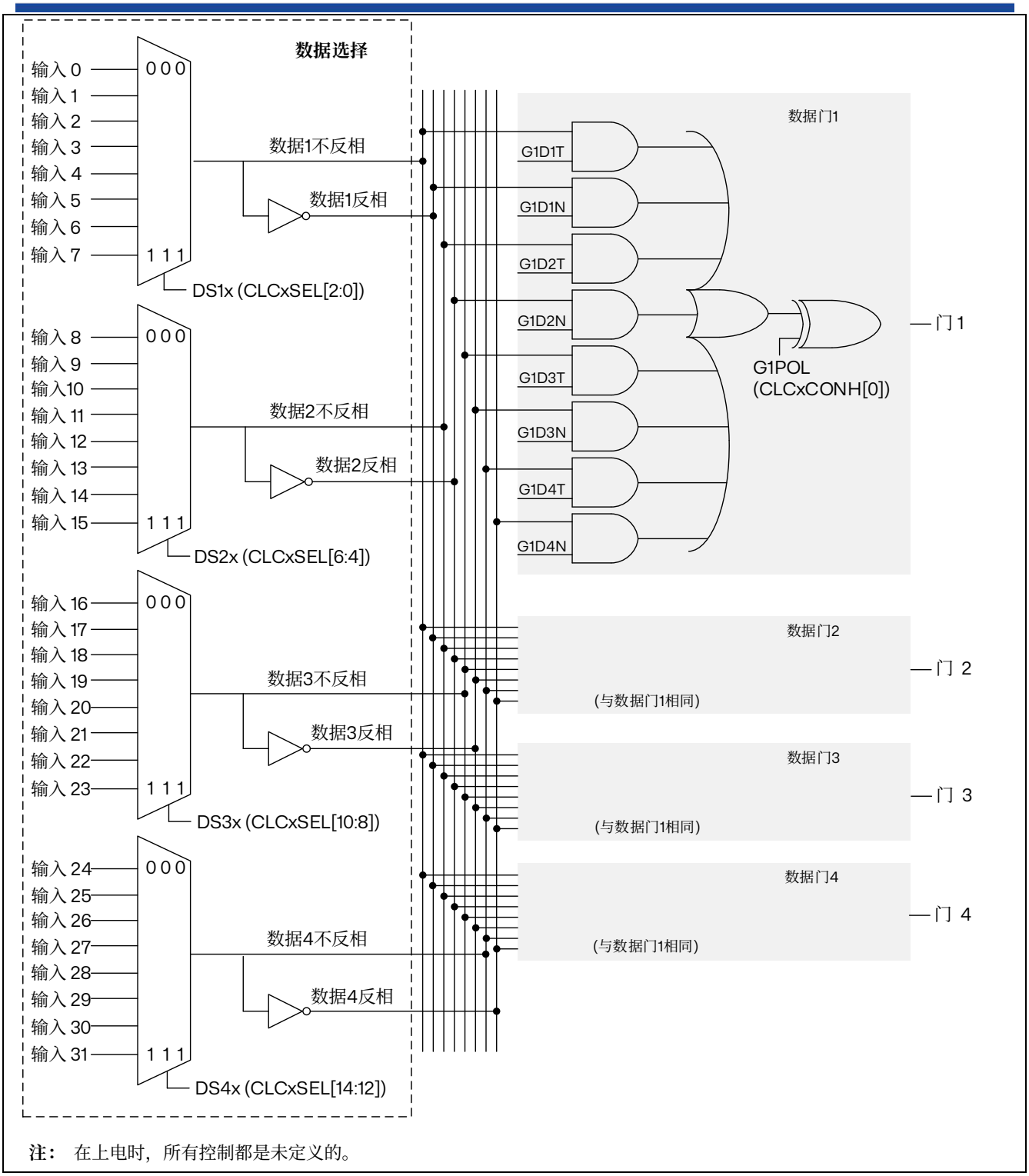


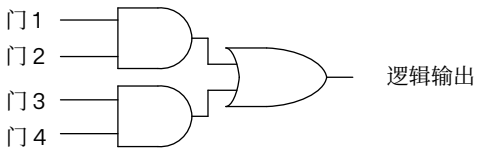

图 12.1 可配置逻辑单元




**注：** 在上电时，所有控制都是未定义的。

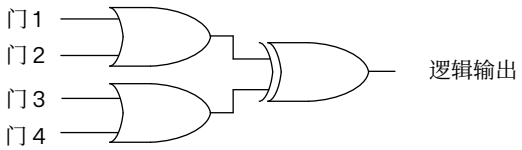
图 12.2 CLC 输入源选择

**AND - OR**



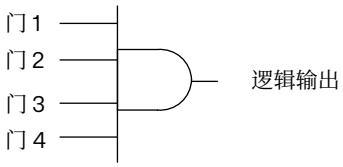

MODE[2:0] = 0 0 0

**OR - XOR**



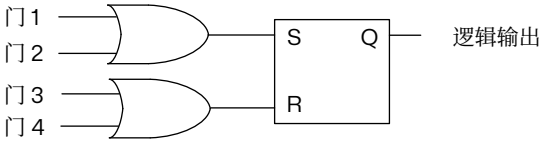

MODE[2:0] = 0 0 1

**4 输入 AND**



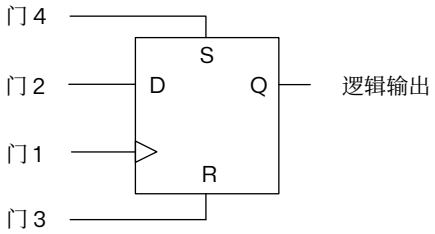

MODE[2:0] = 0 1 0

**S–R 锁存器**



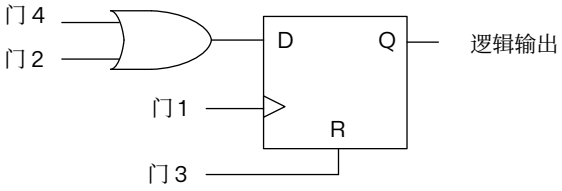

MODE[2:0] = 0 1 1

**带置 1 和复位功能的 1 输入 D 触发器**



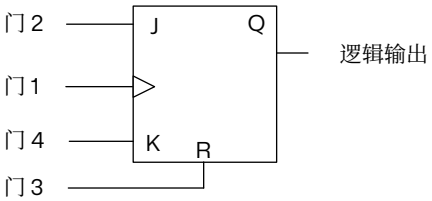

MODE[2:0] = 1 0 0

**带复位功能的 2 输入 D 触发器**



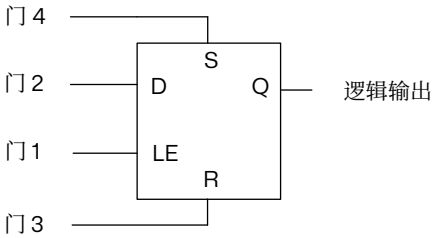

MODE[2:0] = 1 0 1

**带复位功能的 J-K 触发器**




MODE[2:0] = 1 1 0

**带置 1 和复位功能的 1 输入透明锁存器**




MODE[2:0] = 1 1 1

图 12.3 逻辑功能组合选项

## 12.2 CLC 引脚和内部信号

CLC 包含：

- 多达 32 个输入源用于多路开关选择（CLCxSEL 寄存器）
- 可通过 GPIO 引脚复用来选择对应的 CLC 输入和输出。

关于 CLC 引脚复用功能表和 CLC 输入源请参见下两张表。

表 12.1 CLC 引脚复用功能表

| 复用 | INA | INB | INC | IND | 1OUT | 2OUT | 3OUT | 4OUT |
|---|---|---|---|---|---|---|---|---|
| AF10 | PA7 | PB13 | PB14 | PB0 | PB1 | PB2 | PB10 | PB11 |
| | PB12 | PC4 | PC5 | PB15 | PD8 | PD9 | PD10 | PC6 |
| | PE9 | PE10 | PE11 | PE12 | PE14 | PE15 | PE7 | PE8 |

表 12.2 CLC 输入源

| DSx[2:0] | DS1 | DS2 | DS3 | DS4 |
|---|---|---|---|---|
| 000 | CLC_INA(1) | CLC_INB(1) | CLC_INC(1) | TIM15_TRGO |
| 001 | SYSCLK | HRTIM1_CHB1 | CLC1_OUT | CLC2_OUT |
| 010 | CLC3_OUT | COMP1_OUT | COMP2_OUT | COMP3_OUT |
| 011 | LSI | UART1_TX | SPI1_TX(2)(3) | SPI1_RX(2)(3) |
| 100 | MCO | COMP4_OUT | UART1_RX | HRTIM1_CHE1 |
| 101 | TIM1_OC5 | TIM8_OC5 | CLC4_OUT | CLC_IND(1) |
| 110 | HRTIM1_CHA1 | HRTIM1_CHC1 | HRTIM1_CHD1 | HRTIM1_CHF1 |
| 111 | TIM2_OC1 | TIM3_OC1 | TIM4_OC1 | TIM5_OC1 |

(1) 该输入源来自对应的 GPIO 引脚。关于复用功能选择配置，请参见 CLC 引脚复用功能表。
(2) SPI1 作为主机时，SPI1_MOSI 线上信号作为 TX 输入源，SPI1_MISO 线上信号作为 RX 输入源。
(3) SPI1 作为从机时，SPI1_MISO 线上信号作为 TX 输入源，SPI1_MOSI 线上信号作为 RX 输入源。

## 12.3 CLC 设置

CLCxCON[15:0]用于选择逻辑功能，并决定和控制 I/O 引脚。CLCxCON[19:16]用于控制输出信号极性。要使 CLC 工作，LCEN（CLCxCON[15]）必须置 1。在 LCEN 清零时，可以设定所有寄存器。

CLCxSEL 寄存器用于控制将哪些输入信号送至图 13.2 的输入总线。真值（T）和取反（N）值都会在数据总线上提供。

CLCxGLS 寄存器用于选择将数据总线上的哪些信号施加到输入 OR 门。真值和取反输入单独进行使能；不要对同一信号同时使能两者。

CLC 模块输出的最终极性由 LCPOL（CLCxCON[5]）控制。当 LCPOL = 1 时，输出反相，当 LCPOL = 0 时，输出不反相。GxPOL 位（CLCxCON[19:16]）用于控制逻辑功能输入的极性。

INTP 和 INTN 引脚（CLCxCON[11:10]）用于允许在 CLC 输出的上升沿和下降沿产生中断。INTPIF 和 INTNIF 位（CLCxCON[9:8]）用于配置 CLC 输出的上升沿和下降沿中断标志位。

LCOUT 位为只读位，它反映逻辑单元输出的状态。要将 CLCxOUT 信号输出到某个 I/O 引脚，需将 LCOE 位置 1 并将 I/O 配置为复用输出。CLCxOUT 信号通过 GPIO 引脚复用功能选择，需要进行相应配置。关于 GPIO 引脚复用功能表请参见器件数据手册。

### 12.4 输入提供

CLC 中的每个逻辑单元可接受 4 个输入，每个输入均来自 4 个数据门之一。每个数据门都连接到 8 个输入源。数据门允许在每个输入源的反相或不反相极性之间进行选择。可供 CLC 使用的输入源请参见章节：CLC 引脚和内部信号。

#### 12.4.1 源多路开关

该模块具有 4 个输入源多路开关。多路开关输入通过设置 CLCxSEL 寄存器中的控制位进行选择，以定义要通过 4 个数据选择多路开关中的每个多路开关选择的数据源。4 个数据选择多路开关的每个多路开关都送至 4 个逻辑功能输入门之一，如图 13.3 所示。该模块具有基于每个输入源多路开关的输出构造的内部数据总线（见图 13.2）。数据总线同时具有每个选定输入源的真值（T）和取反（N）版本。因此，内部数据总线上最多具有 8 个信号可连接到逻辑功能的输入门。

#### 12.4.2 逻辑输入门

4 个逻辑输入门用于将来自数据选择多路开关的输入源送至 4 个逻辑功能输入。每个输入源信号的真值和取反形式都可供每个逻辑门使用。输入信号源使用 CLCxGLS 寄存器来进行使能，供每个逻辑功能输入使用。最多可以使能 8 个信号，供每个逻辑功能输入使用。对于 4 个逻辑功能输入中的每个输入，可以使能 8 个信号源中任意数量的信号源。每个逻辑门均提供输入信号的逻辑或（OR）。选定（真值或取反）信号通过逻辑或来构成门输出数据。逻辑 NAND 通过使用 GxPOL 位更改输出极性而获得。如果需要逻辑与（AND），可以根据 DeMorgan 法则，选择取反输入并反转输出极性。如果所有输入均取反，并施加到 NOR 逻辑，结果将与逻辑与完全相同。写为代数形式：

C = A AND B
等同于：
C = NOT(NOT(A) OR NOT(B))。

下表总结了可以通过使用门控制位获得的基本功能。该表显示的是使用全部 4 个输入多路开关源的情形，但输入门可以配置为使用较少的源。如果未选择任何输入（CLCxGLS = 0x00），则输出将为 0 或 1，具体取决于 GxPOL 位。

表 12.3 逻辑功能示例

| CLCxGLS | GxPOL 位 | 功能 |
|---|---|---|
| 0xAAAA | 0 | OR（D1，D2，D3，D4） |
| 0xAAAA | 1 | NOR（D1，D2，D3，D4） |
| 0x5555 | 0 | NAND（D1，D2，D3，D4） |
| 0x5555 | 1 | AND（D1，D2，D3，D4） |
| 0x0000 | 0 | 逻辑 0 |
| 0x0000 | 1 | 逻辑 1 |

如果某个门的输出必须为 0 或 1，则建议的方法是将 CLCxGLS 中与该门相关的所有位均设置为 0，并使用门极性位 GxPOL 来设置所需的电平。

### 12.4.3 逻辑功能

有 8 种可用的逻辑功能，包括：

- AND-OR
- OR-XOR
- AND
- S-R 锁存器
- 带置 1 和复位功能的 D 触发器
- 带复位功能的 D 触发器
- 带复位功能的 J-K 触发器
- 带置 1 和复位功能的透明锁存器

这些逻辑功能如图 13.3 所示。每种逻辑功能具有 4 个输入和 1 个输出。MODE[2:0]位（CLCxCON[2:0]）用于设置逻辑单元的功能行为。共有 4 个组合选项和 4 个状态选项。3 个状态选项将输入门 1 定义为上升沿时钟，具有 D 和 J-K 触发器的传统含义。第 4 个状态选项（MODE[2:0]位= 111）为透明锁存器；当锁存器使能（LE）为真时，Q 跟随 D 变化；当 LE 为假时，Q 保持状态。对于同时具有 S（置 1）和 R（复位）输入的选项，当 S 或 R 为逻辑 1 时，输出将随时钟异步变化；R 将优先。

### 12.4.4 软件输入

逻辑功能的门数据输入可由软件直接控制，方法是将与逻辑门相关的所有 CLCxGLS 位均设置为 0，并写入相应的 GxPOL 位（见表 13.1）。门输出将等于 GxPOL 位的值。

## 12.5 输出

LCOUT（CLCxCON[6]）是逻辑单元输出，并被送至 I/O 端口引脚或器件内的其他模块。在所有情况下，信号值均在 LCPOL 反相器之后获取。要在某个 I/O 引脚上观察该输出，用户需要将 LCOE（CLCxCON[7]）置 1。

## 12.6 应用逻辑

CLC 同时提供了组合和状态（见图 13.3）逻辑功能选项。输入门的输出将施加到逻辑功能。如果 CLCxGLS = 0x00，当 GxPOL 位（CLCxCON[3:0]）清零时，功能将接收到逻辑 0，当 GxPOL 位置 1 时，将接收到逻辑 1。

### 12.6.1 组合逻辑

组合功能（MODE[2:0] = 010、001 和 000）基于输入门的 AND/OR 逻辑进行构造。4 输入 AND 逻辑可以提供 OR 逻辑功能，方法是根据 DeMorgan 法则对输入和输出进行反相。对 XOR 逻辑的输出进行反相等效于对其中一个输入（但不是同时对两个输入）进行反相。

当 LCEN（CLCxCON[15]）清零时，SR 功能（MODE[2:0] = 011）不会受影响，对于状态逻辑寄存器也是如此。锁存器为复位优先型，这意味着复位信号优先于可能存在的任何置 1 信号。

### 12.6.2 状态逻辑

状态功能同时包含了带异步置 1（S）和复位（R）功能的 D 触发器和 J-K 触发器。输入门 1 提供上升沿时钟。如果需要下降沿时钟，可以通过门逻辑（G1POL）对门 1 进行反相。输入门 2（有时还有门 4）向寄存器或锁存器输入提供数据。在透明锁存器模式（MODE[2:0] = 111）下工作时，如果 LE 为高电平，则输出 Q 跟随 D 变化，如果 LE 为低电平，则输出 Q 保持状态。

各种模式可能共用也可能不共用状态存储器，切换模式可能会也可能不会改变状态变量的状态。对于所有模式，寄存器均为复位优先。

## 12.7 CLC 中断

CLC 模块具有两种类型的可允许中断：上升沿中断事件和下降沿中断事件。这两种中断事件分别通过 INTP（CLCxCON[11]）和 INTN（CLCxCON[10]）控制位使能。

使能该模块（LCEN = 1）时，如果在 INTP = 1 时出现上升沿输出或在 INTN = 1 时出现下降沿事件，对应的 INTPIF（CLCxCON[9]）和 INTNIF（CLCxCON[8]）中断标志位将被置起。

如果 CLC 逻辑的初始输出状态为 1，并且 INTP = 1，则在 LCEN 设置为 1 时会产生中断。类似地，如果 CLC 的初始输出状态为 0，并且 INTN = 1，则会产生中断。这些条件必须在软件中进行检测并清除。类似地，如果在使能 CLC 模块时 INTP 或 INTN 置 1，则可能会产生假中断。

用户应确保清除在 CLC 模块初始化过程中可能出现的任何假中断事件。

如果 CLC 中断允许位 INTP 和 INTN 清零，则不会产生中断。但是，如果出现中断条件，对应的 INTPIF 和 INTNIF 中断标志位仍会置 1。用户可通过在中断服务程序（ISR）中清零 INTPIF 和 INTNIF 以清除该中断。

## 12.8 睡眠模式操作

CLC 模块不会受睡眠模式影响，因为它不依赖于系统时钟源工作。但是，一些输入源在睡眠期间可能会被禁止，所以功能可能被中断。如果时钟源继续工作，模块也将继续工作。

## 12.9 复位

向 LCEN 位写入 0 时，所有状态逻辑功能的输出将复位为 0。系统复位会将 CLCxCON、CLCxSEL、CLCxGLS 寄存器恢复为默认状态并禁止该模块。

执行器件复位会将模块寄存器中的所有位恢复为默认状态。在复位之后，所有逻辑功能的输出为 0；这同时包括锁存器和触发器功能。当器件复位置为有效（LCEN（CLCxCONL[15]）= 0）时，状态逻辑将复位，逻辑功能的输出将强制为低电平。

## 12.10 CLC 寄存器

CLC 模块由以下寄存器控制：

- CLCxCON
- CLCxSEL
- CLCxGLS

CLCx 控制寄存器（CLCxCON）用于使能模块和允许中断、控制输出使能位、选择输出极性和选择逻辑功能。此外，CLCx 控制寄存器不仅允许用户控制单元输出的逻辑极性，而且允许控制一些中间变量的逻辑极性。

CLCx 输入多路开关选择寄存器（CLCxSEL）允许用户为图 13.2 中虚线内所示的 4 个数据选择多路开关中的每个多路开关选择 8 个输入信号中的一个。4 个数据选择多路开关中每个多路开关的输出都连接到通过 MODE[2:0]位（CLCxCON[2:0]）选择的逻辑功能的输入，请参见图 13.3。

CLCx 源使能寄存器（CLCxGLS）允许用户基于通过 CLCxSEL 配置的 4 个输入数据源构造任意 4 个布尔变量表达式。对于 4 个信号中的每个信号（通过 CLCx 输入多路开关选择寄存器 CLCxSEL 选择），真值和反码值都可供图 13.2 中的数据门中所示的积和电路使用。

### 12.10.1 可配置逻辑单元 x 控制寄存器（CLCxCON）（x = 1~4）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | G4POL | G3POL | G2POL | G1POL |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LCEN | Reserved | | | INTP | INTN | INTPIF | INTNIF | LCOE | LCOUT | LCPOL | Reserved | | MODE[2:0] | | |
| rw | | | | rw | rw | rw | rw | rw | r | rw | | | rw | rw | rw |

Bit 31:20 保留，必须保持为复位值。

Bit 19 **G4POL**：门 4 极性控制位

0：门 4 逻辑的输出不反相

1：门 4 逻辑的输出在施加到逻辑单元时反相

Bit 18 **G3POL**：门 3 极性控制位

0：门 3 逻辑的输出不反相

1：门 3 逻辑的输出在施加到逻辑单元时反相

Bit 17 **G2POL**：门 2 极性控制位

0：门 2 逻辑的输出不反相

1：门 2 逻辑的输出在施加到逻辑单元时反相

Bit 16 **G1POL**：门 1 极性控制位

0：门 1 逻辑的输出不反相

1：门 1 逻辑的输出在施加到逻辑单元时反相

Bit 15 **LCEN**：CLCx 可配置逻辑单元使能

0：禁止 CLCx 可配置逻辑单元，并输出逻辑 0

1：使能 CLCx 可配置逻辑单元，并混合输入信号

Bit 14:12 保留，必须保持为复位值。

Bit 11 **INTP**：CLCx 可配置逻辑单元上升沿中断使能位
0：不产生中断
1：当 LCOUT 上出现上升沿时，将产生中断
Bit 10 **INTN**：CLCx 可配置逻辑单元下降沿中断使能位
0：不产生中断
1：当 LCOUT 上出现下降沿时，将产生中断
Bit 9 **INTPIF**：CLCx 可配置逻辑单元上升沿中断标志位
0：没有上升沿中断发生，写 0 清除此标志位
1：发生上升沿中断
Bit 8 **INTNIF**：CLCx 可配置逻辑单元下降沿中断标志位
0：没有下降沿中断发生，写 0 清除此标志位
1：发生下降沿中断
Bit 7 **LCOE**：CLCx 可配置逻辑单元端口使能
0：禁止端口引脚输出
1：使能端口引脚输出
Bit 6 **LCOUT**：CLCx 可配置逻辑单元数据输出状态位
0：输出低电平
1：输出高电平
Bit 5 **LCPOL**：CLCx 可配置逻辑单元输出极性控制位
0：模块的输出不反相
1：模块的输出反相

Bit 4:3 保留，必须保持为复位值。

Bit 2:0 **MODE[2:0]**：CLCx 可配置逻辑单元模式位
000：单元是 AND-OR 逻辑
001：单元是 OR-XOR 逻辑
010：单元是 4 输入 AND 逻辑
011：单元是 SR 锁存器
100：单元是带置 1 和复位功能的 1 输入 D 触发器
101：单元是带复位功能的 2 输入 D 触发器
110：单元是带复位功能的 JK 触发器
111：单元是带置 1 和复位功能的 1 输入透明锁存器

### 12.10.2 可配置逻辑单元 x 输入多路开关选择寄存器（CLCxSEL）（x = 1~4）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res | DS4[2:0] | | | Res | DS3[2:0] | | | Res | DS2[2:0] | | | Res | DS1[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

Bit 31:15 保留，必须保持为复位值。

Bit 14:12 **DS4[2:0]：**数据选择多路开关 4 信号选择位

Bit 11 保留，必须保持为复位值。

Bit 10:8 **DS3[2:0]：**数据选择多路开关 3 信号选择位

Bit 7 保留，必须保持为复位值。

Bit 6:4 **DS2[2:0]：**数据选择多路开关 2 信号选择位

Bit 3 保留，必须保持为复位值。

Bit 2:0 **DS1[2:0]：**数据选择多路开关 1 信号选择位

### 12.10.3 可配置逻辑单元 x 源使能寄存器（CLCxGLS）（x = 1~4）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G4D4T | G4D4N | G4D3T | G4D3N | G4D2T | G4D2N | G4D1T | G4D1N | G3D4T | G3D4N | G3D3T | G3D3N | G3D2T | G3D2N | G3D1T | G3D1N |
| 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 |
| G2D4T | G2D4N | G2D3T | G2D3N | G2D2T | G2D2N | G2D1T | G2D1N | G1D4T | G1D4N | G1D3T | G1D3N | G1D2T | G1D2N | G1D1T | G1D1N |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **G4D4T：**与 G1 相同
Bit 30 **G4D4N：**与 G1 相同
Bit 29 **G4D3T：**与 G1 相同
Bit 28 **G4D3N：**与 G1 相同
Bit 27 **G4D2T：**与 G1 相同
Bit 26 **G4D2N：**与 G1 相同
Bit 25 **G4D1T：**与 G1 相同
Bit 24 **G4D1N：**与 G1 相同
Bit 23 **G3D4T：**与 G1 相同
Bit 22 **G3D4N：**与 G1 相同
Bit 21 **G3D3T：**与 G1 相同
Bit 20 **G3D3N：**与 G1 相同
Bit 19 **G3D2T：**与 G1 相同
Bit 18 **G3D2N：**与 G1 相同
Bit 17 **G3D1T：**与 G1 相同

Bit 16 **G3D1N**：与 G1 相同
Bit 15 **G2D4T**：与 G1 相同
Bit 14 **G2D4N**：与 G1 相同
Bit 13 **G2D3T**：与 G1 相同
Bit 12 **G2D3N**：与 G1 相同
Bit 11 **G2D2T**：与 G1 相同
Bit 10 **G2D2N**：与 G1 相同
Bit 9 **G2D1T**：与 G1 相同
Bit 8 **G2D1N**：与 G1 相同
Bit 7 **G1D4T**：门 1 数据 4 真值使能位
0：对于门 1 禁止数据 4（不反相）信号
1：对于门 1 使能数据 4（不反相）信号
Bit 6 **G1D4N**：门 1 数据 4 取反使能位
0：对于门 1 禁止数据 4（反相）信号
1：对于门 1 使能数据 4（反相）信号
Bit 5 **G1D3T**：门 1 数据 3 真值使能位
0：对于门 1 禁止数据 3（不反相）信号
1：对于门 1 使能数据 3（不反相）信号
Bit 4 **G1D3N**：门 1 数据 3 取反使能位
0：对于门 1 禁止数据 3（反相）信号
1：对于门 1 使能数据 3（反相）信号
Bit 3 **G1D2T**：门 1 数据 2 真值使能位
0：对于门 1 禁止数据 2（不反相）信号
1：对于门 1 使能数据 2（不反相）信号
Bit 2 **G1D2N**：门 1 数据 2 取反使能位
0：对于门 1 禁止数据 2（反相）信号
1：对于门 1 使能数据 2（反相）信号
Bit 1 **G1D1T**：门 1 数据 1 真值使能位
0：对于门 1 禁止数据 1（不反相）信号
1：对于门 1 使能数据 1（不反相）信号
Bit 0 **G1D1N**：门 1 数据 2 取反使能位
0：对于门 1 禁止数据 1（反相）信号
1：对于门 1 使能数据 1（反相）信号

# 13 CORDIC 协同处理器（CORDIC）

## 13.1 CORDIC 简介

CORDIC 协处理器为电机控制、计量、信号处理以及许多其他应用中常用的数学函数（主要是三角函数）提供硬件加速。

与软件实现相比，它加快了这些函数的计算速度，使得可以使用更低的操作频率，或者释放处理器周期以执行其他任务。

## 13.2 CORDIC 主要特征

- 24 位 CORDIC 旋转引擎
- 圆形和双曲线模式
- 旋转和向量模式
- 函数：正弦 sin、余弦 cos、双曲正弦 sinh、双曲余弦 cosh、反正切 atan、反正切 2 atan2、反双曲正切 atanh、模 modulus、平方根 sqrt、自然对数 ln
- 可编程精度
- 低延迟 AHB 从接口
- 结果准备好后即可读取，无需轮询或中断
- DMA 读写通道
- DMA 多寄存器读/写

## 13.3 CORDIC 功能描述

### 13.3.1 通用描述

CORDIC 是求解三角函数和双曲函数的一种经济高效的连续逼近算法。

在三角函数（圆形）模式下，通过以递减的角度旋转单位向量[1, 0]，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。旋转后向量的 x 和 y 笛卡尔分量分别对应于 θ 的余弦和正弦。相反地，对应于反正切（y/x）的向量[x, y]的角度，是通过将[x, y]以递减的角度逐次旋转以获得单位向量[1,0] 来确定的。旋转角度的累积和给出了原始向量的角度。

CORDIC 算法还可以通过将连续的圆形旋转替换为沿双曲线的步骤，用于计算双曲函数（sinh、cosh、atanh）。

其他函数可以从上述基本函数中派生出来。

### 13.3.2 CORDIC 函数

使用协处理器的第一步是通过相应编程 CORDIC_CR 寄存器的 FUNC 位域来选择所需的函数。

下表列出了 CORDIC 协处理器支持的函数。

表 13.1 CORDIC 函数

| 函数 | 描述 | 主要参数（ARG1） | 次要参数（ARG2） | 主要结果（RES1） | 次要结果（RES2） |
|---|---|---|---|---|---|
| cos | 余弦 | 角度 θ | 模 m | m • cos θ | m • sin θ |
| sin | 正弦 | 角度 θ | 模 m | m • sin θ | m • cos θ |
| phase | 相位 | x | y | atan2(y,x) | $\sqrt{x^2+y^2}$ |
| modulus | 模 | x | y | $\sqrt{x^2+y^2}$ | atan2(y,x) |
| atan | 反正切 | x | 无 | $\tan^{-1}$ x | 无 |
| cosh | 双曲余弦 | x | 无 | cosh x | sinh x |
| sinh | 双曲正弦 | x | 无 | sinh x | cosh x |
| atanh | 双曲反正切 | x | 无 | $\tanh^{-1}$ x | 无 |
| ln | 自然对数 | x | 无 | ln x | 无 |
| sqrt | 平方根 | x | 无 | $\sqrt{x}$ | 无 |

一些函数接受两个输入参数（ARG1 和 ARG2），并且一些函数同时生成两个结果（RES1 和 RES2）。这是算法的一个附加功能，意味着只需一次操作即可获得两个值。例如，在执行极坐标到直角坐标转换时就是这种情况：sin θ 同时生成 cos θ，cos θ 同时生成 sin θ。类似地，对于直角坐标到极坐标转换（phase(x,y)，modulus(x,y)）和双曲函数（cosh θ，sinh θ）也是如此。

*注：指数函数* exp x *可以通过* sinh x *和* cosh x *的和来获得。此外，以* N *为底的对数* $\log_N$ x *可以通过将* lnx *乘以常数* K *来得到，其中* K = 1/ln N。

对于某些函数（atan, log, sqrt），可以应用一个缩放系数（见第 14.3.4 节）来扩展函数的范围，使其超出 q1.31 定点格式支持的最大范围[-1, 1]。对于所有其他圆形函数，缩放系数必须设置为 0；对于双曲函数，缩放系数必须设置为 1。

**cos 余弦**

表 13.2 余弦参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | θ 角度（以弧度为单位），除以 π | [-1, 1] |
| ARG2 | 模 m | [0, 1] |
| RES1 | m • cos θ | [-1, 1] |
| RES2 | m • sin θ | [-1, 1] |
| SCALE | 不适用 | 0 |

此函数计算范围在-π 到 π 之间的角度的余弦值。它也可以用于执行极坐标到直角坐标的转换。

主要参数是以弧度表示的角度θ。在编程 ARG1 之前，必须将其除以 π。

次要参数是模 m。如果 m 大于 1，则必须在软件中应用缩放以适应 ARG2 的 q1.31 范围。

主要结果 RES1 是角度的余弦值乘以模。

次要结果 RES2 是角度的正弦值乘以模。

### sin 正弦

表 13.3 正弦参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | θ 角度（以弧度为单位），除以 π | [-1, 1] |
| ARG2 | 模 m | [0, 1] |
| RES1 | m • sin θ | [-1, 1] |
| RES2 | m • cos θ | [-1, 1] |
| SCALE | 不适用 | 0 |

此函数计算范围在-π 到 π 之间的角度的正弦值。它也可以用于执行极坐标到直角坐标的转换。

主要参数是以弧度表示的角度θ。在编程 ARG1 之前，必须将其除以 π。

次要参数是模 m。如果 m 大于 1，则必须在软件中应用缩放以适应 ARG2 的 q1.31 范围。

主要结果 RES1 是角度的正弦值乘以模。

次要结果 RES2 是角度的余弦值乘以模。

### phase 相位

表 13.4 相位参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | x 坐标 | [-1, 1] |
| ARG2 | y 坐标 | [-1, 1] |
| RES1 | 相位角θ（以弧度为单位），除以 π | [-1, 1] |
| RES2 | m • cos θ | [0, 1] |
| SCALE | 不适用 | 0 |

此函数计算向量 **v** = [x y]在-π 到 π 范围内的相位角（也称为 atan2(y,x)）。它也可以用于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，则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，则必须在软件中应用缩放以适应 ARG2 的 q1.31 范围。

主要结果 RES1 是向量 **v** 的相位角 θ。为了获得以弧度表示的角度，RES1 必须乘以 π。请注意，由于相位角的循环性质，接近 π 的值有时会回绕到-π。

次要结果 RES2 是模，由下式给出：|v|=$\sqrt{x^2 + y^2}$ 。如果|**v**| > 1，则 RES2 中的结果将饱和到 1。

### modulus 模

表 13.5 模参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | x 坐标 | [-1, 1] |
| ARG2 | y 坐标 | [-1, 1] |
| RES1 | 模 m | [0, 1] |
| RES2 | 相位角θ | [-1, 1] |
| SCALE | 不适用 | 0 |

此函数计算向量 **v** = [x y]的大小（或称模）。它也可以用于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 轴方向上的分量大小。如果|x| > 1，则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 轴方向上的分量大小。如果|y| > 1，则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，由下式给出：$|v|=\sqrt{x^2+y^2}$。如果|**v**| > 1，则 RES1 的结果将饱和到 1。

次要结果 RES2 是向量 **v** 的相位角 θ。为了获得以弧度表示的角度，RES2 必须乘以 π。请注意，由于相位角的循环性质，接近 π 的值有时会回绕到-π。

### atan 反正切

表 13.6 反正切参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [-1, 1] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-n} \cdot \tan^{-1} x$（以弧度为单位），除以π | [-1, 1] |
| RES2 | 不适用 | - |
| SCALE | n | [0 7] |

此函数计算输入参数 x 的反正切值，即反正切（或逆正切）。

主要参数 ARG1 是输入值 x = tanθ。如果|x| > 1，则必须在软件中应用一个 $2^{-n}$ 的缩放系数，使得$-1 < x \cdot 2^{-n} < 1$。将缩放后的值 $x \cdot 2^{-n}$ 编程到 ARG1 中，并将缩放系数 n 编程到 SCALE 参数中。

请注意，允许的最大输入值是 tanθ = 128，这对应于角度 θ = 89.55 度。对于|x| > 128 的情况，必须使用软件方法来求解 $\tan^{-1} x$。

次要参数 ARG2 未使用。

主要结果 RES1 是角度 $\theta = \tan^{-1} x$。为了获得以弧度表示的角度，RES1 必须乘以 $2^n \cdot \pi$。

次要结果 RES2 未使用。

### cosh 双曲余弦

表 13.7 双曲余弦参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [-0.559, 0.559] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-n} \cdot \cosh x$ | [0.5, 0.846] |
| RES2 | $2^{-n} \cdot \sinh x$ | [-0.683, 0.683] |
| SCALE | n | 1 |

此函数计算双曲角度 x 的双曲余弦值。它也可以用于计算指数函数 $e^x = \cosh x + \sinh x$ 和 $e^{-x} = \cosh x - \sinh x$。

主要参数是双曲角度 x。仅支持范围在-1.118 到+1.118 之间的 x 值。由于 cosh x 的最小值为 1，超出了 q1.31 格式的范围，因此必须在软件中应用一个 $2^{-n}$ 的缩放系数。必须将因子 n = 1 编程到 SCALE 参数中。

次要参数未使用。

主要结果 RES1 是双曲余弦值，即 cosh x。为了获得正确结果，RES1 必须乘以 2。

次要结果 RES2 是双曲正弦值，即 sinh x。为了获得正确结果，RES2 必须乘以 2。

### sinh 双曲正弦

表 13.8 双曲正弦参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [-0.559, 0.559] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-n} \cdot \sinh x$ | [-0.683, 0.683] |
| RES2 | $2^{-n} \cdot \cosh x$ | [0.5, 0.846] |
| SCALE | n | 1 |

此函数计算双曲角度 x 的双曲正弦值。它也可以用于计算指数函数 $e^x = \cosh x + \sinh x$ 和 $e^{-x} = \cosh x - \sinh x$。

主要参数是双曲角度 x。仅支持范围在-1.118 到+1.118 之间的 x 值。对于所有输入值，必须在软件中应用一个 $2^{-n}$ 的缩放系数，其中 n = 1。将缩放后的值 $x \cdot 0.5$ 编程到 ARG1 中，并将因子 n = 1 编程到 SCALE 参数中。

次要参数未使用。

主要结果 RES1 是双曲正弦值，即 sinh x。为了获得正确结果，RES1 必须乘以 2。

次要结果 RES2 是双曲余弦值，即 cosh x。为了获得正确结果，RES2 必须乘以 2。

### atanh 双曲反正切

表 13.9 双曲反正切参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [-0.403 0.403] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-n} \cdot \text{atanh } x$ | [-0.559 0.559] |
| RES2 | 不适用 | - |
| SCALE | n | 1 |

此函数计算输入参数 x 的双曲反正切值。

主要参数是输入值 x。仅支持范围在-0.806 到+0.806 之间的 x 值。输入值 x 必须乘以一个 $2^{-n}$ 的缩放系数进行缩放，其中 n = 1。将缩放后的值 $x \cdot 0.5$ 编程到 ARG1 中，并将因子 n = 1 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，即 atanh x。为了获得正确值，RES1 必须乘以 2。

次要结果未使用。

### ln 自然对数

表 13.10 自然对数参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [0.054 0.875] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-(n+1)} \cdot \ln x$ | [-0.279 0.137] |
| RES2 | 不适用 | - |
| SCALE | n | [1 4] |

此函数计算输入参数 x 的自然对数。

主要参数是输入值 x。仅支持范围在 0.107 到 9.35 之间的 x 值。输入值 x 必须乘以一个 $2^{-n}$ 的缩放系数进行缩放，使得 $x \cdot 2^{-n} < 1\text{-}2^{-n}$。将缩放后的值 $x \cdot 2^{-n}$ 编程到 ARG1 中，并将因子 n 编程到 SCALE 参数中。

下表列出了有效的缩放系数 n 以及对应的 x 和 ARG1 的范围。

表 13.11 自然对数缩放系数及对应范围

| n | x 范围 | ARG1 范围 |
|---|---|---|
| 1 | $0.107 \le x < 1$ | $0.0535 \le \text{ARG1} < 0.5$ |
| 2 | $1 \le x < 3$ | $0.25 \le \text{ARG1} < 0.75$ |
| 3 | $3 \le x < 7$ | $0.375 \le \text{ARG1} < 0.875$ |
| 4 | $7 \le x \le 9.35$ | $0.4375 \le \text{ARG1} < 0.584$ |

次要参数未使用。

主要结果是自然对数，即 ln x。为了获得正确值，RES1 必须乘以 $2^{(n+1)}$。

次要结果未使用。

**sqrt 平方根**

表 13.12 平方根参数

| 参数 | 描述 | 范围 |
|---|---|---|
| ARG1 | $x \cdot 2^{-n}$ | [0.027 0.875] |
| ARG2 | 不适用 | - |
| RES1 | $2^{-n}\sqrt{x}$ | [0.04 1] |
| RES2 | 不适用 | - |
| SCALE | n | [0 2] |

此函数计算输入参数 x 的平方根。

主要参数是输入值 x。仅支持范围在 0.027 到 2.34 之间的 x 值。输入值 x 必须乘以一个 $2^{-n}$ 的缩放系数进行缩放，使得 $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$。

将缩放后的值 $x \cdot 2^{-n}$ 编程到 ARG1 中，并将因子 n 编程到 SCALE 参数中。

下表列出了有效的缩放系数 n 以及对应的 x 和 ARG1 的范围。

表 13.13 平方根缩放系数及对应范围

| n | x 范围 | ARG1 范围 |
|---|---|---|
| 1 | $0.027 \le x < 0.75$ | $0.027 \le ARG1 < 0.75$ |
| 2 | $0.75 \le x < 1.75$ | $0.375 \le ARG1 < 0.875$ |
| 3 | $1.75 \le x \le 2.341$ | $0.4375 \le ARG1 \le 0.585$ |

次要参数未使用。

主要结果是 x 的平方根。为了获得正确值，RES1 必须乘以 2n。

次要结果未使用。

### 13.3.3 定点表示法

CORDIC 采用定点有符号整数格式运算。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 1 个符号位和 31 个小数位（二进制小数位）表示。因此，数值范围为-1（0x80000000）至 $1 - 2^{-31}$（0x7FFFFFFF）。

在 q1.15 格式中，数值范围为 1（0x8000）至 $1 - 2^{-15}$（0x7FFF）。此格式的优点在于，两个输入参数可以打包到一个 32 位写操作中，且两个结果可以通过一次 32 位读取操作获取。

### 13.3.4 缩放系数

上述章节中列出的几个函数指定了一个缩放系数 SCALE。这使得函数输入范围得以扩展，以覆盖 CORDIC 支持的全值范围，而不会使输入、输出或内部寄存器饱和。如果需要使用缩放系数，则必须通过软件计算并将其编程到 CORDIC_CSR 寄存器的 SCALE 位域中。

在将缩放后的值编程到 CORDIC_WDATA 寄存器之前，必须相应地对输入参数进行缩放。同时，在从 CORDIC_RDATA 寄存器读取结果时，也必须对缩放进行还原。

*注：由于缩放值的截断，缩放系数会导致精度损失。*

### 13.3.5 精度

结果的精度取决于 CORDIC（坐标旋转数字计算）的迭代次数。对于三角函数（正弦、余弦、相位、模量），该算法以每次迭代一个二进制位的恒定速率收敛，见图 14.1。

对于双曲函数（双曲正弦、双曲余弦、自然对数），由于 CORDIC 算法的特殊性，其收敛速率较不恒定（见图 14.2）。平方根函数的收敛速度大约是双曲函数的两倍（见图 14.3）。

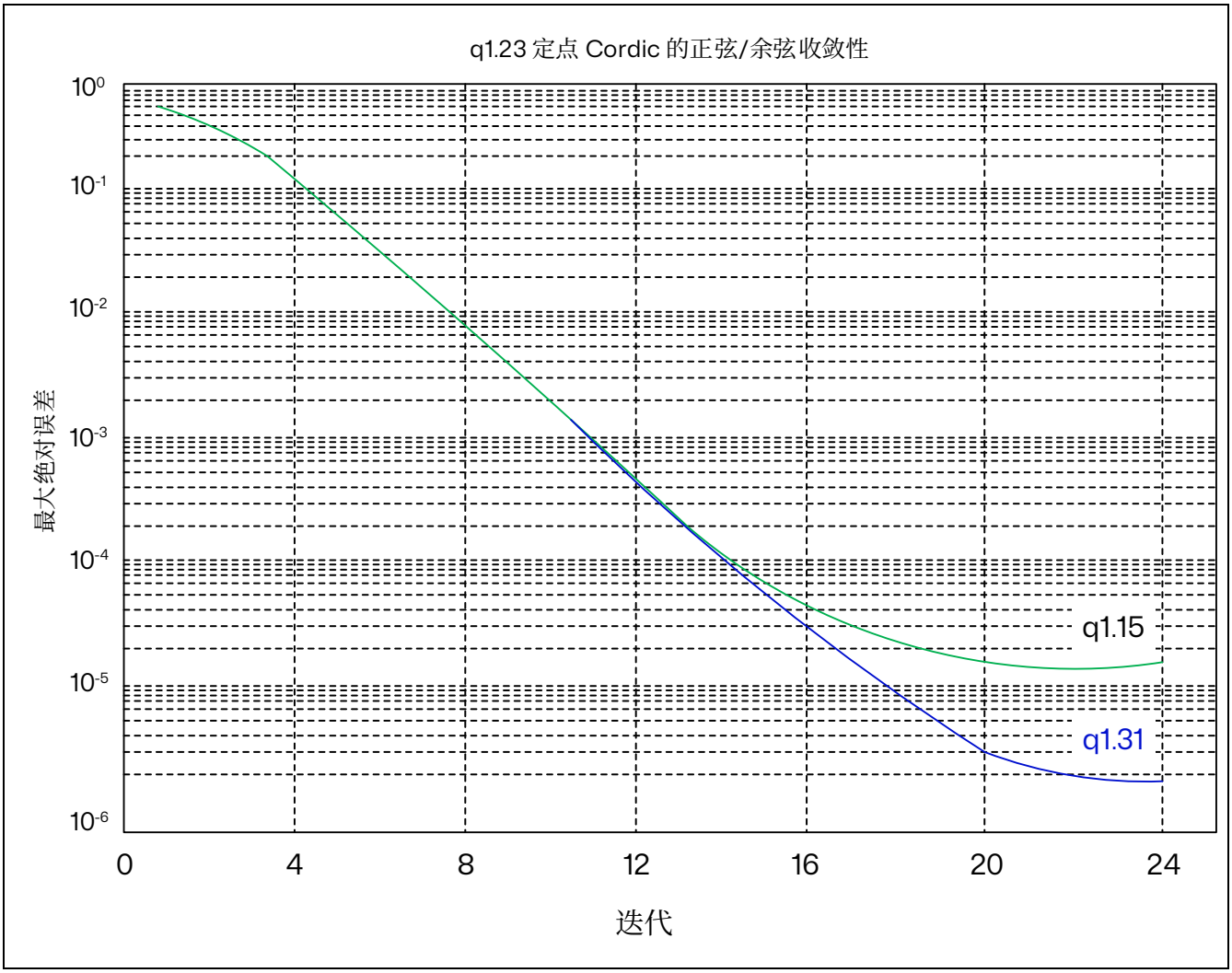



图 13.1 CORDIC 在三角函数中的收敛情况

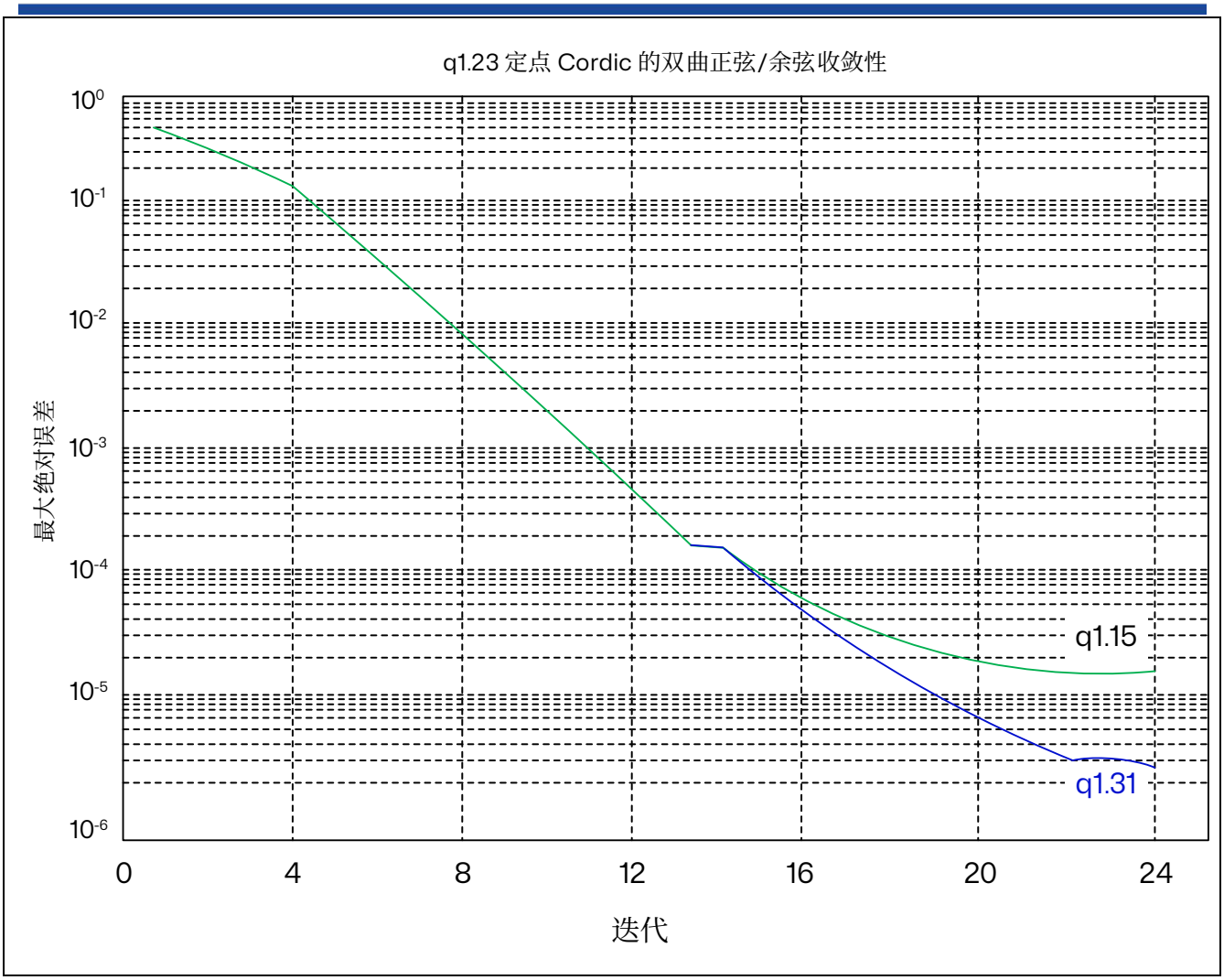



图 13.2 CORDIC 在双曲函数中的收敛情况

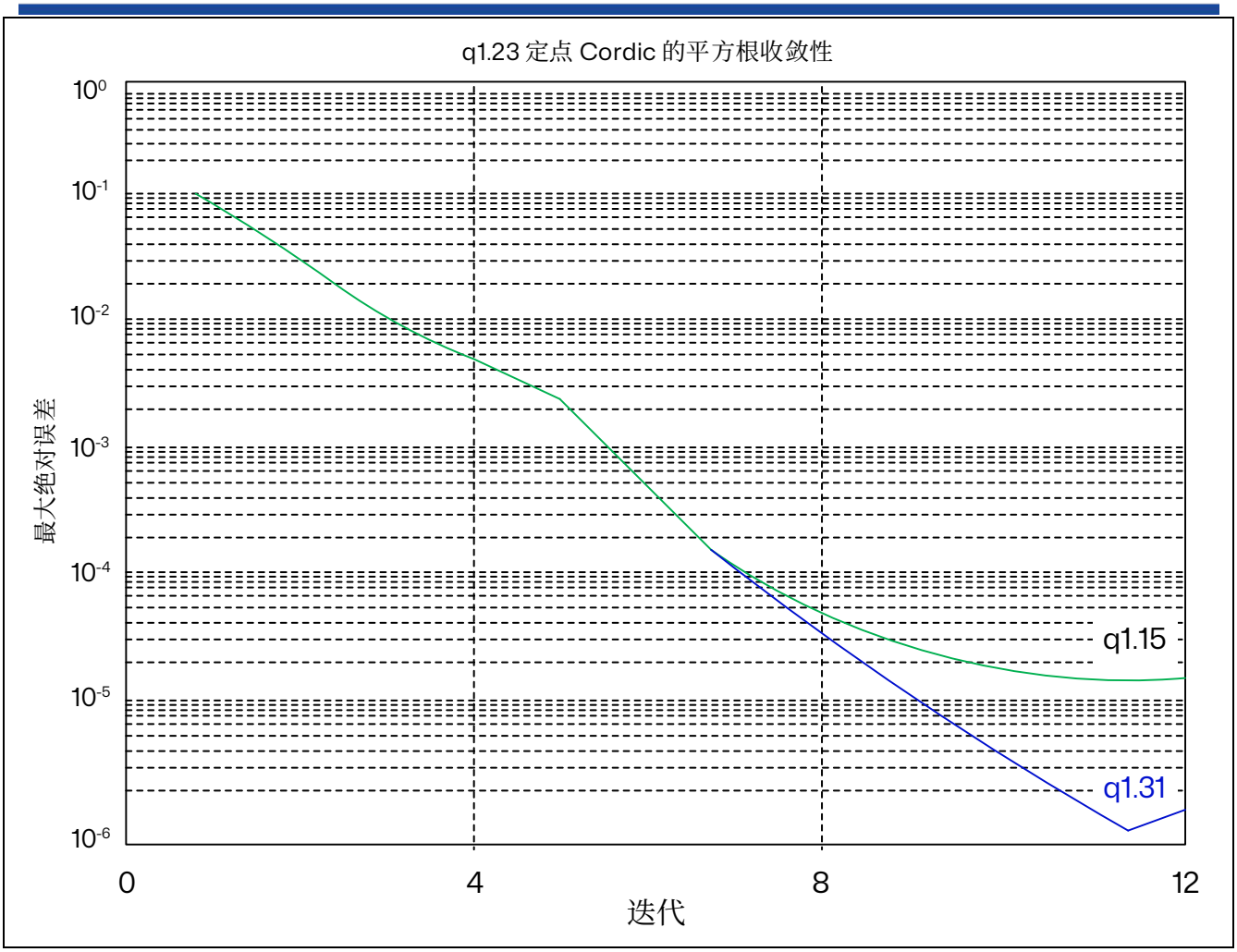



图 13.3 CORDIC 在平方根函数中的收敛情况

*注：当量化误差开始变得显著时，收敛速率会降低。*

CORDIC 每个时钟周期可执行三次迭代。对于每个函数，下表中列出了每三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。根据此表，可以确定所需的周期数，并将其编程到 CORDIC_CR 寄存器的 PRECISION 位域中。协处理器在完成编程的迭代次数后立即停止，并且可以立即读取结果。

表 13.14 精度与迭代次数

| 函数 | 迭代次数 | 循环次数 | 最大剩余误差(1) | |
|---|---|---|---|---|
| | | | q1.31 格式 | q1.15 格式 |
| Sin, Cos, Phase(2), Mod, Atan(4) | 3 | 1 | $2^{-2}$ | $2^{-2}$ |
| | 6 | 2 | $2^{-5}$ | $2^{-5}$ |
| | 9 | 3 | $2^{-8}$ | $2^{-8}$ |
| | 12 | 4 | $2^{-11}$ | $2^{-11}$ |
| | 15 | 5 | $2^{-14}$ | $2^{-14}$ |
| | 18 | 6 | $2^{-17}$ | $2^{-16}$ |
| Sinh, Cosh, Atanh, | 3 | 1 | $2^{-1}$ | $2^{-1}$ |
| | 6 | 2 | $2^{-4}$ | $2^{-4}$ |
| | 9 | 3 | $2^{-7}$ | $2^{-7}$ |

| 函数 | 迭代次数 | 循环次数 | 最大剩余误差(1) | |
|---|---|---|---|---|
| | | | q1.31 格式 | q1.15 格式 |
| Ln(3) | 12 | 4 | $2^{-10}$ | $2^{-10}$ |
| | 15 | 5 | $2^{-13}$ | $2^{-13}$ |
| | 18 | 6 | $2^{-16}$ | $2^{-15}$ |
| Sqrt(4) | 3 | 1 | $2^{-6}$ | $2^{-6}$ |
| | 6 | 2 | $2^{-12}$ | $2^{-12}$ |
| | 9 | 3 | $2^{-16}$ | $2^{-15}$ |

1. 最大剩余误差是指在给定迭代次数后剩余的最大误差，与以双精度浮点执行的相同计算相比。可能会产生额外的舍入误差，对于 q15 格式或 q31 格式最多为 $2^{-16}$ 或 $2^{-20}$。
2. 对于模量>0.5 的情况，随着量化误差变得显著，可实现的精度会按比例降低。
3. SCALE=1。如果使用更高的缩放系数，可实现的精度会按比例降低。
4. SCALE=0。如果使用更高的缩放系数，可实现的精度会按比例降低。

### 13.3.6 零开销模式

使用协处理器的最快方法是在 CORDIC_CSR 寄存器中预编程要执行的功能（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），以及适用的缩放系数（SCALE）。

计算通过将输入参数写入 CORDIC_WDATA 寄存器来触发。一旦写入了正确数量的输入参数（且任何正在进行的计算已完成），就会使用这些输入参数和当前的 CORDIC_CSR 设置启动新的计算。如果没有更改，则无需重新编程 CORDIC_CSR 寄存器。

如果需要双 32 位输入参数（ARGSIZE = 0, NARGS = 1），则必须先写入主要输入参数（ARG1），然后写入次要参数（ARG2）。如果在一系列计算中次要参数保持不变，则可以在第一次计算开始后，通过将参数数量重新编程为一个（NARGS = 0）来避免第二次写入。只要函数不更改，次要参数就会保持其编程值。

*注：复位后，*ARG2 *被设置为*+1 *（*0x7FFFFFFF*）。*

如果使用两个 16 位参数（ARGSIZE = 1），则必须将它们打包到一个 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）。

对于仅接受一个输入参数 ARG1 的函数，建议设置 NARGS = 0。如果 NARGS = 1，则必须执行对 CORDIC_WDATA 的第二次写入来触发计算。在这种情况下，ARG2 数据不被使用。

一旦计算开始，任何尝试读取 CORDIC_RDATA 寄存器的操作都会插入总线等待状态，直到计算完成并返回结果。然后，软件可以写入输入并立即读取结果，而无需轮询查看其是否有效。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。这段时间可以用于为下一次计算编程 CORDIC_CSR 寄存器，并在需要时准备下一个输入数据。在计算进行中，可以重新编程 CORDIC_CSR 寄存器，而不会影响正在进行的计算的结果。同样，一旦考虑到前一个参数，就可以使用下一个参数更新 CORDIC_WDATA 寄存器。下一个参数和设置将保持挂起状态，直到前一个计算完成。

当计算完成时，可以从 CORDIC_RDATA 寄存器读取结果。如果预期有两个 32 位结果（NRES = 1, RESSIZE = 0），则首先读出主要结果（RES1），然后是次要结果（RES2）。如果只预期一个 32 位结果（NRES = 0，RESSIZE = 0），则在第一次读取时输出 RES1。

如果预期有 16 位结果（RESSIZE = 1），则对 CORDIC_RDATA 的一次读取会获取打包到一个 32 位字中的两个结果。RES1 位于较低半字中，RES2 位于较高半字中。在这种情况下，建议编程 NRES = 0。

如果 NRES = 1，则必须执行对 CORDIC_RDATA 的第二次读取以释放 CORDIC 以便进行下一次操作。第二次读取的数据必须被丢弃。

当下一个计算所需的参数数量已被写入，且已读取预期数量的结果时，下一个计算就会开始。这意味着在任何时候，都可能有一个计算正在进行中，或等待读取结果，以及一个操作处于挂起状态。当操作处于挂起状态时，对 CORDIC_WDATA 的任何进一步访问都会取消它并覆盖数据。

以下序列总结了在零开销模式下使用 CORDIC_IP 的步骤：

1. 使用适当的设置编程 CORDIC_CSR 寄存器
2. 在 CORDIC_WDATA 寄存器中为第一次计算编程参数。这将启动第一次计算。
3. 如果需要，为下一次计算更新 CORDIC_CSR 寄存器设置。
4. 在 CORDIC_WDATA 寄存器中为下一次计算编程参数。
5. 从 CORDIC_RDATA 寄存器读取结果。这将触发下一次计算。
6. 转到步骤 3。

### 13.3.7 轮询方式

当 CORDIC_RDATA 寄存器中有新结果可用时，CORDIC_CSR 寄存器中会设置 RRDY 标志。通过读取该寄存器可以轮询此标志。读取 CORDIC_RDATA 寄存器（一次或两次，具体取决于 CORDIC_CSR 寄存器的 NRES 位域）可以复位该标志。

轮询 RRDY 标志的时间比直接读取 CORDIC_RDATA 寄存器稍长，因为在结果可用时不会立即读取。读取 CORDIC_CSR 寄存器时，处理器和总线接口不会停滞，因此如果无法接受处理器停滞（例如，必须处理低延迟中断），则此模式可能适用。

### 13.3.8 中断模式

通过在 CORDIC_CSR 寄存器中设置中断使能（IE）位，每当设置 RRDY 标志时，就会生成一个中断。当标志复位时，中断被清除。

此模式允许在中断服务例程中读取计算结果，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更慢。

### 13.3.9 DMA 模式

若 CORDIC_CSR 寄存器中设置了 DMA 写使能（DMAWEN）位，且没有待处理操作，则会生成一个 DMA 写通道请求。DMA 控制器可将主要输入参数（ARG1）从内存传输至 CORDIC_WDATA 寄存器。向该寄存器写入数据会取消 DMA 请求。若 CORDIC_CSR 寄存器中 NARGS=1，则会生成第二个 DMA 写通道请求，以将次要输入参数（ARG2）传输至 CORDIC_WDATA 寄存器。当所有输入参数均已写入，且任何正在进行的计算已完成（通过读取结果确认），则会开始新的计算，并生成另一个 DMA 写通道请求。

若 CORDIC_CSR 寄存器中设置了 DMA 读使能（DMAREN）位，则 RRDY 标志激活时会生成一个 DMA 读通道请求。此时，DMA 控制器可将主要结果（RES1）从 CORDIC_RDATA 寄存器传输至内存。读取该寄存器会取消 DMA 请求。若 CORDIC_CSR 寄存器中 NRES=1，则会生成第二个 DMA 请求以读取次要结果（RES2）。当所有结果均已读取时，RRDY 标志会被取消。

DMA 读和写通道可分别启用。若两个通道均启用，则 CORDIC 可在无需处理器干预的情况下，自主地对数据缓冲区执行重复计算。这使处理器能够执行其他任务。对于写通道，DMA 控制器以内存到外设模式运行；对于读通道，则以外设到内存模式运行。序列由处理器设置 DMAWEN 标志启动，DMA 读和

写请求的生成速度取决于 CORDIC 处理数据的速度。

在某些情况下，输入数据可能存储在内存中，而输出则定期传输至另一个外设，如数模转换器。在这种情况下，目标外设每次需要新数据时都会生成一个 DMA 请求。DMA 控制器可直接从 CORDIC_RDATA 寄存器中获取下一个样本（此时，尽管源为外设寄存器，但 DMA 控制器仍以内存到外设模式运行）。读取结果的行为使 CORDIC 能够开始新的计算，进而生成 DMA 写通道请求，DMA 控制器则将下一个输入值传输至 CORDIC_WDATA 寄存器。此时，DMA 写通道被启用（DMAWEN=1），但读通道不得启用。

类似地，来自另一个外设（如 ADC）的数据可直接传输至 CORDIC_WDATA 寄存器（以外设到内存模式）。此时，DMA 写通道不得启用。CORDIC 处理输入数据，并在完成后（若 DMAREN=1）生成 DMA 读请求。随后，DMA 控制器将结果从 CORDIC_RDATA 寄存器传输至内存（外设到内存模式）。

*注：不会对* CORDIC_CSR *寄存器生成* DMA *请求。因此，* DMA *模式仅在使用相同设置重复执行相同功能时才有用。在一系列* DMA *传输期间，比例因子不得更改。*

*注：每次* DMA *请求都必须得到确认，这是* DMA *访问* CORDIC_WDATA *或* CORDIC_RDATA *寄存器的结果。如果在此之前对相关寄存器进行了额外的访问，则确认会被提前断言，并可能阻塞* DMA *通道。因此，当启用* DMA *读通道时，必须避免* CPU *访问* CORDIC_RDATA *寄存器。同样，当启用* DMA *写通道时，处理器必须避免访问* CORDIC_WDATA *寄存器。*

## 13.4 CORDIC 寄存器描述

CORDIC 寄存器仅支持 32 位字格式访问

### 13.4.1 CORDIC 控制/状态寄存器（CORDIC_CSR）

偏移地址：0x00

复位值：0x0000 0050

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RRDY | Reserved | | | | | | | | ARG SIZE | RES SIZE | NARGS | NRES | DMA WEN | DMA REN | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | SCALE[2:0] | | | PRECISION[3:0] | | | | FUNC[3:0] | | | |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **RRDY**：结果就绪标志

0：输出寄存器无新数据

1：CORDIC_RDATA 寄存器有新数据

该位在 CORDIC 操作完成时由硬件设置，当 CORDIC_RDATA 寄存器被读取（NRES+1）次后由硬件清除。此位设置时，若 IEN 位也设置，则触发 CORDIC 中断；若 DMAREN 位设置，则生成 DMA 读通道请求。此位设置期间，不会启动新计算。

Bit 30:23 保留，必须保持为复位值。

Bit 22 **ARGSIZE**：输入数据宽度

0：32 位

1：16 位

ARGSIZE 选择输入数据的表示位数。32 位数据时，CORDIC_WDATA 寄存器期望 q1.31 格式参数；16 位数据时，期望 q1.15 格式参数。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。

Bit 21 **RESSIZE**：输出数据宽度

0：32 位

1：16 位

RESSIZE 选择输出数据的表示位数。32 位数据时，CORDIC_RDATA 寄存器包含 q1.31 格式结果；16 位数据时，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格式的次要结果（RES2）。

Bit 20 **NARGS**：CORDIC_WDATA 寄存器预期的参数数量

0：下一个计算仅需一次 32 位写入（或若 ARGSIZE=1，则为两次 16 位值写入）。

1：必须向 CORDIC_WDATA 寄存器写入两个 32 位值以触发下一个计算。

读取返回该位的当前状态。

Bit 19 **NRES**： CORDIC_RDATA 寄存器中的结果数量

0：下一个计算完成后，仅一个 32 位值（或若 RESSIZE=1，则为两个 16 位值）被传输到 CORDIC_RDATA 寄存器。从 CORDIC_RDATA 读取一次即复位 RRDY 标志。

1：下一个计算完成后，两个 32 位值被传输到 CORDIC_RDATA 寄存器。需从 CORDIC_RDATA 读取两次以复位 RRDY 标志。

读取返回该位的当前状态。

Bit 18 **DMAWEN**：使能 DMA 写通道

0：禁用。不生成 DMA 写请求。

1：使能。当无操作挂起时，在 DMA 写通道上生成请求。

此位由软件设置和清除。读取返回该位的当前状态。

Bit 17 **DMAREN**：使能 DMA 读通道

0：禁用。不生成 DMA 读请求。

1：使能。当 RRDY 标志设置时，在 DMA 读通道上生成请求。

此位由软件设置和清除。读取返回该位的当前状态。

Bit 16 **IEN**：使能中断。

0：禁用。不生成中断请求。

1：使能。当 RRDY 标志设置时，生成中断请求。

此位由软件设置和清除。读取返回该位的当前状态。

Bit 15:11 保留，必须保持为复位值。

Bit 10:8 **SCALE[2:0]**：缩放系数

此位域的值表示应用于参数和/或结果的缩放系数。值 n 意味着参数已乘以 $2^{-n}$，和/或结果需要乘以 $2^{n}$。有关每个功能的缩放系数适用性和适当范围，请参阅第 14.3.2 节。

Bit 7:4 **PRECISION[3:0]**：精度要求（迭代次数）

0：保留

1 至 6：（迭代次数）/3

要确定给定精度所需的迭代次数，请参阅表：精度与迭代次数。注意，对于大多数功能，此位域的推荐范围为 3 至 6。

Bit 3:0 **FUNC[3:0]**：功能

0：余弦 cos

1：正弦 sin

2：相位 phase

3：模 modulus

4：反正切 atan

5：双曲余弦 cosh

6：双曲正弦 sinh

7：反双曲正切 atanh

8：自然对数 ln

9：平方根 sqrt

10 至 15：保留

### 13.4.2 CORDIC 参数寄存器（CORDIC_WDATA）

偏移地址：0x04

复位值：0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARG[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 |
| ARG[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:0 **ARG[31:0]：**函数输入参数

此寄存器用于编程存储在 CORDIC_CSR 寄存器 FUNC 位域中选定函数的输入参数。

若选择 32 位格式（CORDIC_CSR.ARGSIZE = 0）且需要两个输入参数（CORDIC_CSR.NARGS = 1），则需对此寄存器进行两次连续写入。第一次写入主要参数（ARG1），第二次写入次要参数（ARG2）。

若选择 32 位格式且仅需一个输入参数（NARGS = 0），则只需对此寄存器进行一次写入，包含主要参数（ARG1）。

若选择 16 位格式（CORDIC_CSR.ARGSIZE = 1），则一次写入此寄存器即包含两个参数。主要参数（ARG1）位于低位半字（ARG[15:0]），次要参数（ARG2）位于高位半字（ARG[31:16]）。在此情况下，NARGS 必须设置为 0。

有关每个函数所需的参数及其允许范围，请参阅第章节：CORDIC 函数。

当所需数量的参数已写入后，若之前的计算已完成，CORDIC 将使用提供的输入参数评估 CORDIC_CSR.FUNC 指定的函数。如果计算正在进行中，ARG1 和 ARG2 的值将保持等待状态，直至计算完成并读取结果。在此期间，对此寄存器的写入将取消等待中的操作并覆盖参数数据。

### 13.4.3 CORDIC 结果寄存器（CORDIC_RDATA）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RES[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 |
| RES[15:0] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:0 **RES[31:0]：**函数的结果

若选择 32 位格式（CORDIC_CSR.RESSIZE = 0）且预期有两个输出结果（CORDIC_CSR.NRES = 1），则在 RRDY 标志置位时，必须对此寄存器进行两次读取。第一次读取获取主要结果（RES1），第二次读取获取次要结果（RES2）并复位 RRDY。

若选择 32 位格式且仅预期有一个输出结果（NRES = 0），则只需对此寄存器进行一次读取，以获取主要结果（RES1）并复位 RRDY 标志。

若选择 16 位格式（CORDIC_CSR.RESSIZE = 1），则此寄存器包含主要结果（RES1）在低位半字（RES[15:0]），次要结果（RES2）在高位半字（RES[31:16]）。在此情况下，NRES 必须设置为 0，且只需进行一次读取。

对此寄存器的读取会复位 CORDIC_CSR 寄存器中的 RRDY 标志。

# 14 FMAC 滤波加速器（FMAC）

## 14.1 FMAC 简介

滤波器数学加速器单元对向量执行算术运算。它包含一个乘/加（MAC）单元，以及地址生成逻辑，使其能够对存储在本地存储器中的向量元素进行索引。

该单元支持输入和输出的循环缓冲区，从而允许实现数字滤波器。既可实现有限冲激响应滤波器，也可实现无限冲激响应滤波器。

该单元允许将频繁或冗长的滤波操作从 CPU 中卸载，从而使处理器能够执行其他任务。在许多情况下，与软件实现相比，它可以加速此类计算，从而加快时间关键型任务的速度。

## 14.2 FMAC 主要特征

- 16×16 位乘法器
- 24+2 位累加器，支持加法和减法
- 16 位输入和输出数据
- 256×16 位本地存储器
- 可在存储器中定义最多 3 个区域用于数据缓冲区（两个输入，一个输出），由可编程基地址指针和相关大小寄存器定义
- 输入和输出缓冲区可为循环缓冲区
- 滤波器功能：FIR、IIR（直接 1 型）
- 向量功能：点积、卷积、相关
- AHB 从接口
- DMA 读写数据通道

## 14.3 FMAC 功能描述

### 14.3.1 通用描述

FMAC 滤波加速模块如下图所示。

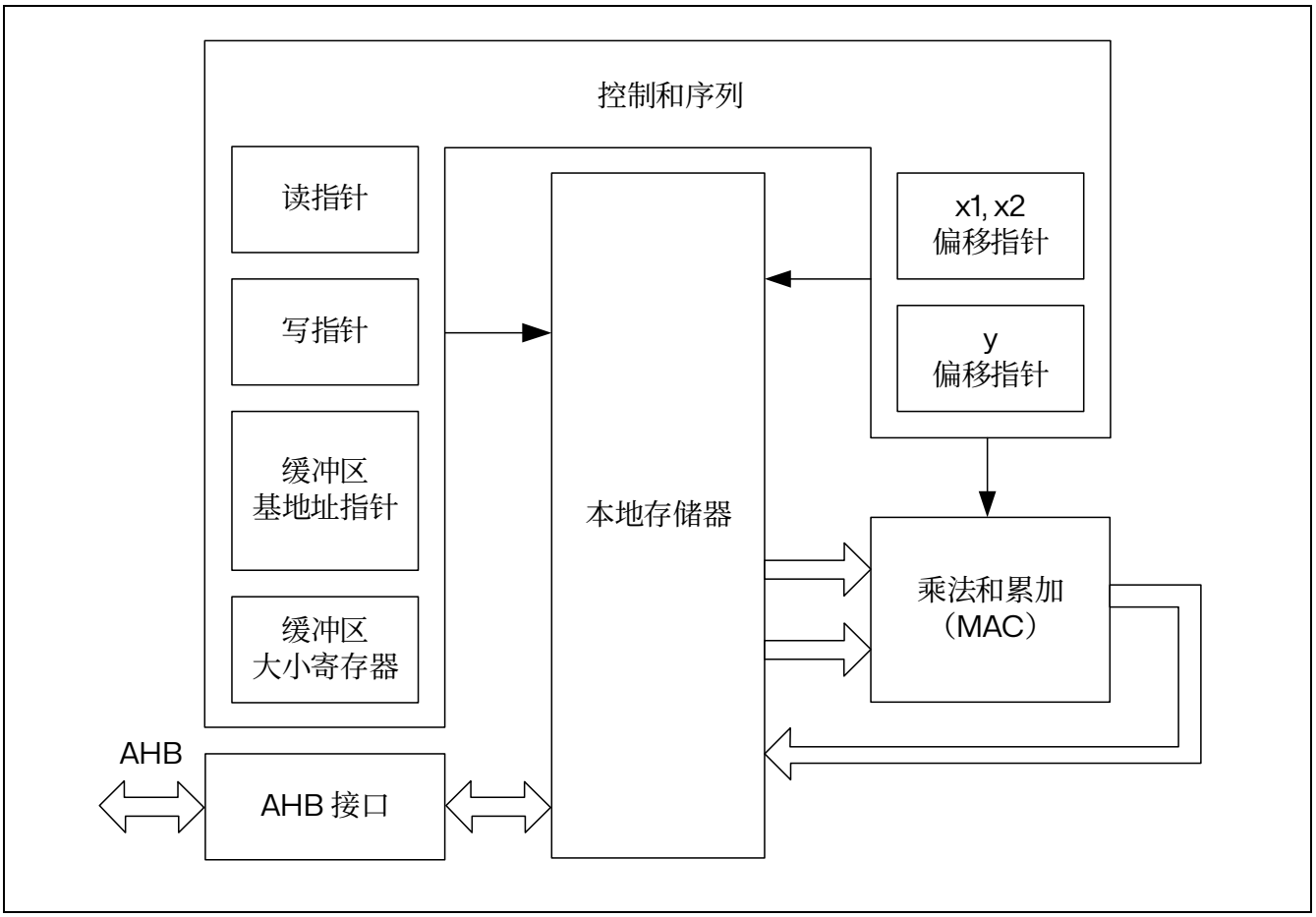



图 14.1 FMAC 模块框图

该单元围绕一个定点乘法器和累加器（MAC）构建。MAC 可以从存储器中取两个 16 位有符号输入值，将它们相乘，并将结果累加到累加器的内容中。存储器中输入值的地址由一组指针确定。这些指针可以由内部硬件进行加载、递增、递减或复位。指针和 MAC 操作由内置序列器控制，以执行所需的操作。

为了计算点积，两个输入向量由处理器或 DMA 控制器加载到本地存储器中，选择并启动所需的操作。每对输入向量元素从存储器中取出，相乘并累加。当所有向量元素处理完毕后，累加器的内容存储在本地存储器中，处理器或 DMA 可以从中读取。

有限冲激响应（FIR）滤波器操作（也称为卷积）包括重复计算系数向量和输入样本向量的点积，后者每次重复时移动一个样本延迟，丢弃最近的样本并添加一个新样本。

无限冲激响应（IIR）滤波器操作是反馈系数与先前输出样本的卷积，再加上 FIR 卷积的结果。

有关滤波器操作的更详细描述，请参见后续章节：滤波器功能。

### 14.3.2 本地存储与缓存

该单元包含一个 256 × 16 位的读写存储器，用于本地存储：

- 输入值（输入向量的元素）存储在两个缓冲区 X1 和 X2 中。
- 输出值（操作结果）存储在另一个缓冲区 Y 中。
- 缓冲区的位置和大小指定如下：
  - x1_base：X1 缓冲区的基地址
  - x2_base：X2 缓冲区的基地址
  - y_base：Y 缓冲区的基地址
  - x1_buf_size：分配给 X1 缓冲区的 16 位地址数量
  - x2_buf_size：分配给 X2 缓冲区的 16 位地址数量
  - y_buf_size：分配给 Y 缓冲区的 16 位地址数量。

这些参数在配置单元时编程到相应的寄存器中。

CPU（或 DMA 控制器）可以使用初始化函数（第 15.3.5 节：初始化函数）和写入数据寄存器来初始化每个缓冲区的内容。数据被传输到目标缓冲区内由写入指针指示的位置。每次新的写入后，写入指针递增。当写入指针到达分配的缓冲区空间的末尾时，它会回绕到基地址。此功能用于在操作之前加载向量的元素，或者初始化滤波器并加载滤波器系数。

#### 缓冲区配置

缓冲区大小和基地址偏移量必须在 X1、X2 和 Y 缓冲区配置寄存器中配置。对于每个函数，所需的缓冲区大小在第 15.3.6 节：滤波函数的函数描述中指定。基地址可以在内部存储器的任意位置选择，前提是所有缓冲区都适合在内部存储器地址范围内（0x00 至 0xFF），即基地址+缓冲区大小必须小于 256。

对缓冲区的大小和位置没有约束（它们可以重叠甚至完全重合）。对于滤波器函数，建议不要重叠缓冲区，因为这可能导致错误行为。

当需要循环缓冲区操作时，可以向缓冲区大小添加一个可选的“预留空间”d。此外，可以设置一个水位线来调节 CPU 或 DMA 活动。d 的值和水位线必须根据应用程序性能要求来选择。为了最大吞吐量，输入缓冲区绝不能为空，因此 d 必须略大于水位线，以允许任何中断或 DMA 延迟。另一方面，如果输入数据不能如单元处理它们那样快地提供，则可以允许缓冲区在等待写入下一个数据时变空，因此 d 可以等于水位线（以确保输入不会发生溢出）。

### 14.3.3 输入缓存

X1 和 X2 缓冲区用于存储 MAC 的输入数据。每次乘法都从 X1 缓冲区和 X2 缓冲区中取一个值并将它们相乘。控制单元中的一个指针为每个值生成读地址偏移量（相对于缓冲区基地址）。指针由硬件根据当前函数管理。

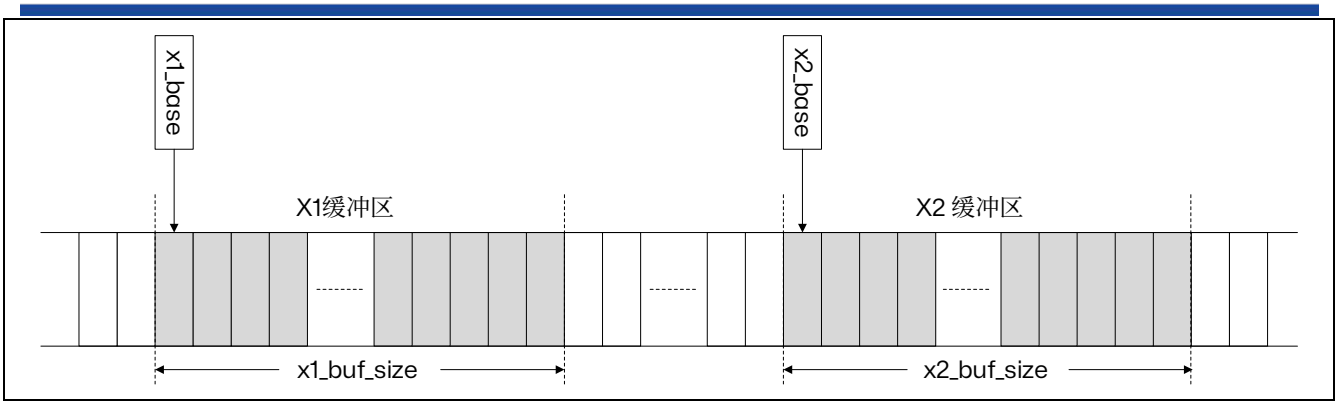



图 14.2 输入缓冲区

X1 缓冲区可用作循环缓冲区，在这种情况下，只要有可用空间，新数据就会不断传输到输入缓冲区中。对于数字滤波器而言，预加载此缓冲区是可选的，因为如果在操作开始时缓冲区中没有写入任何输入样本，则会将其标记为空，这会触发 CPU 或 DMA 加载新样本，直到有足够的样本开始操作。然而，在向量操作的情况下，预加载仍然是有用的，即输入数据已经存在于系统内存中，并且不需要循环操作。

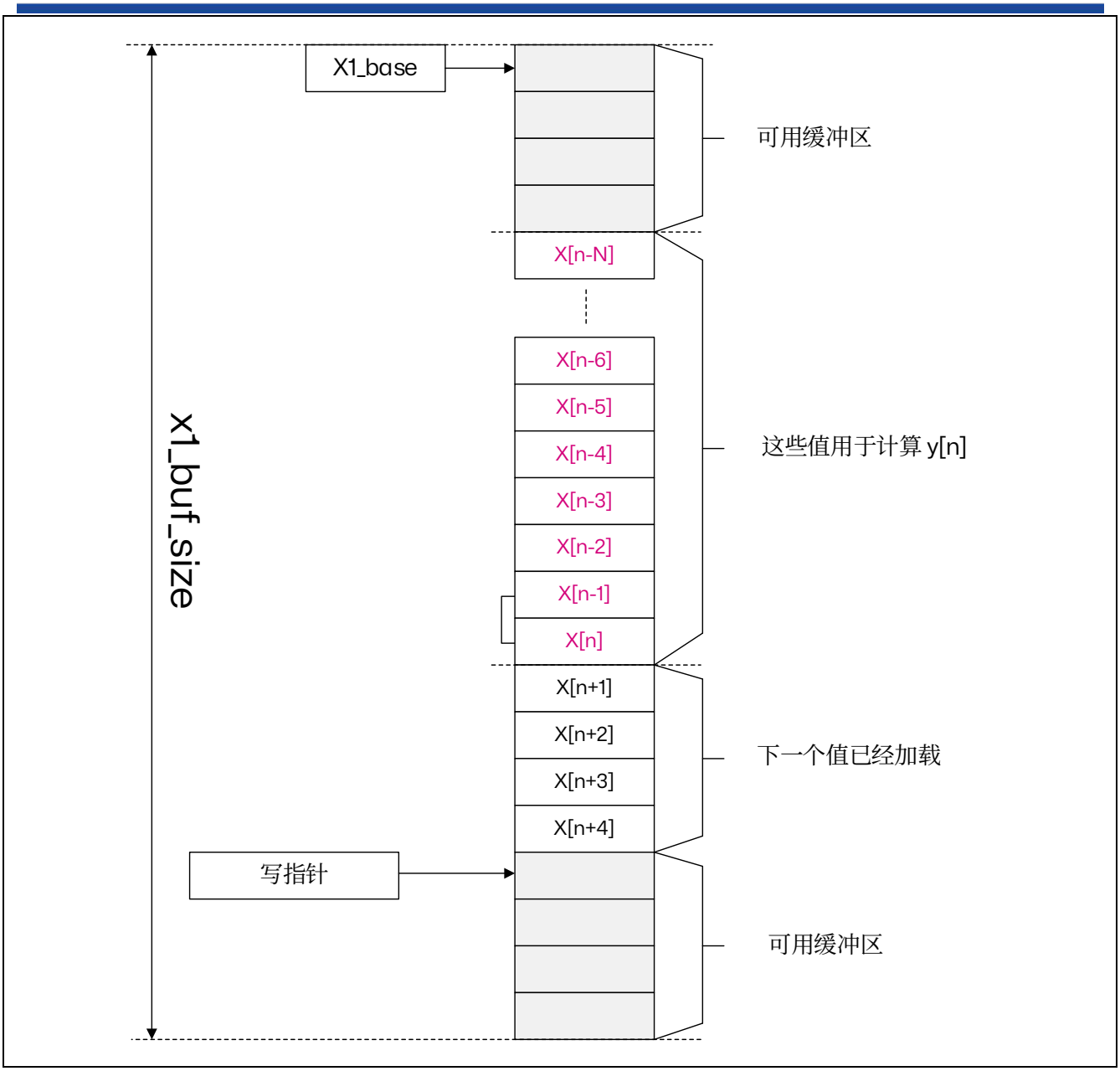



图 14.3 循环输入缓冲区

X2 缓冲区仅能在向量模式（即非循环模式）下使用，并且需要进行预加载，除非缓冲区内容在连续操作之间不发生变化。对于滤波功能，X2 缓冲区用于存储滤波器系数。

当作为循环缓冲区操作时，分配给缓冲区的空间（x1_buf_size）通常必须大于当前计算中使用的元素数量，以确保缓冲区中始终有新值可用。上图展示了滤波操作中缓冲区的布局。在计算输出样本 y[n]时，单元使用一组 N+1 个输入样本，即 x[n-N]到 x[n]。完成此操作后，单元开始计算 y[n+1]，使用输入样本组 x[n-N+1]到 x[n+1]。此时，最早的输入样本 x[n-N]从输入集中移除，并加入新样本 x[n+1]。

处理器或 DMA 控制器必须确保在需要时新样本 x[n+1]在缓冲区空间中可用。如果不可用，缓冲区将被标记为空，这将暂停单元的执行，直到添加新样本。X1 缓冲区不会发出下溢条件信号。

*注：如果样本流由定时器或其他外设（如 ADC）控制，则缓冲区会定期变空，因为滤波器处理每个新样本的速度比源提供样本的速度快。这是滤波器操作的一个基本特征。*

如果缓冲区中的空闲空间数量小于 FMAC_X1BUFCFG 寄存器的 FULL_WM 位域中编程设定的水位阈值，则缓冲区会被标记为满。只要未满标志未设置，如果使能，就会生成中断以请求为缓冲区提供更多数据。水位阈值允许在一次中断下传输多个数据，而不会发生溢出。然而，如果确实发生溢出，则会设置 OVFL 错误标志，并忽略写入数据。在发生溢出时，写入指针不会递增。

下图展示了滤波操作过程中 X1 缓冲区的操作。此示例显示了一个 8 阶 FIR 滤波器，其水位阈值设置为 4。

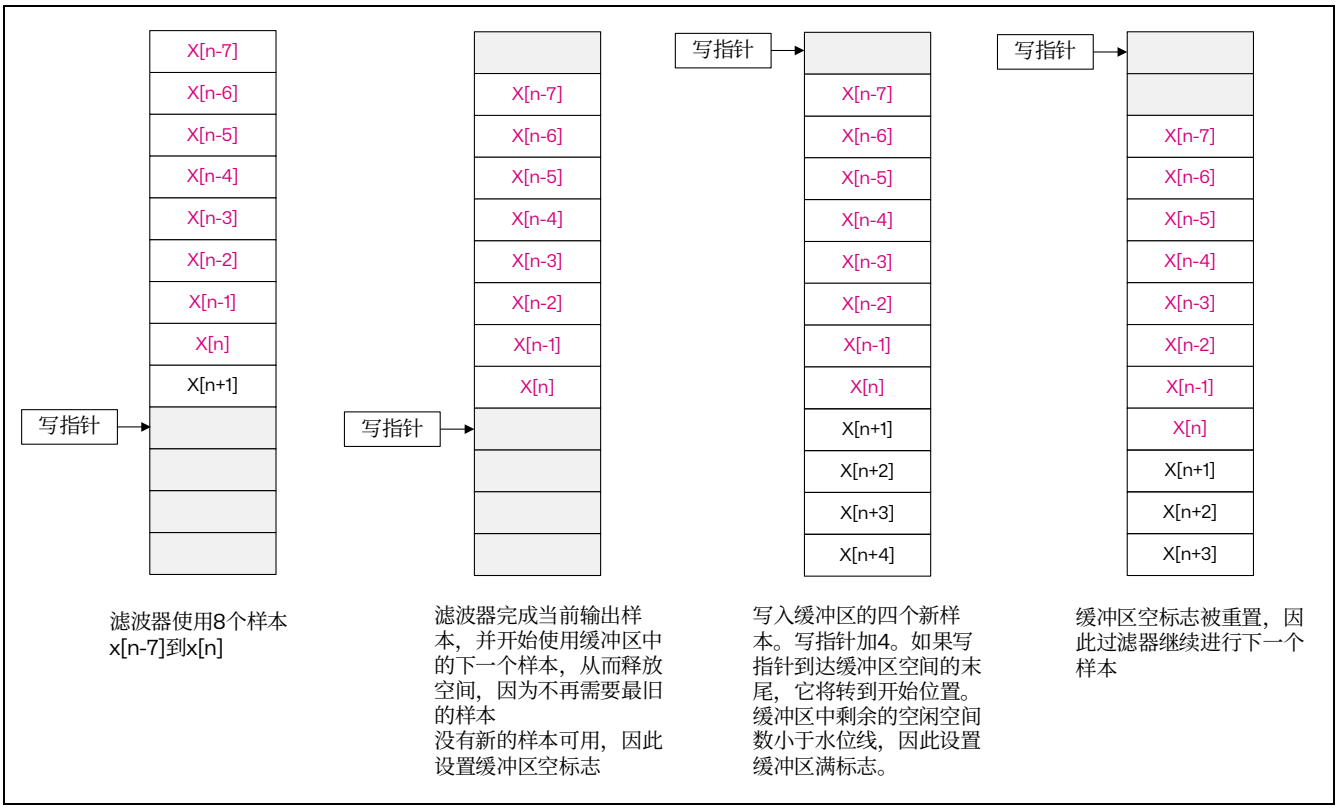



图 14.4 循环输入缓冲区操作

### 14.3.4 输出缓存

Y（输出）缓冲区用于存储累加的结果。每个新的输出值都存储在缓冲区中，直到被处理器或 DMA 控制器读取。每次对读取数据寄存器进行读访问时，都会从读指针指示的地址中获取读取数据。该指针在每次读取后递增，并在到达分配的 Y 缓冲区空间末尾时回绕到基地址。

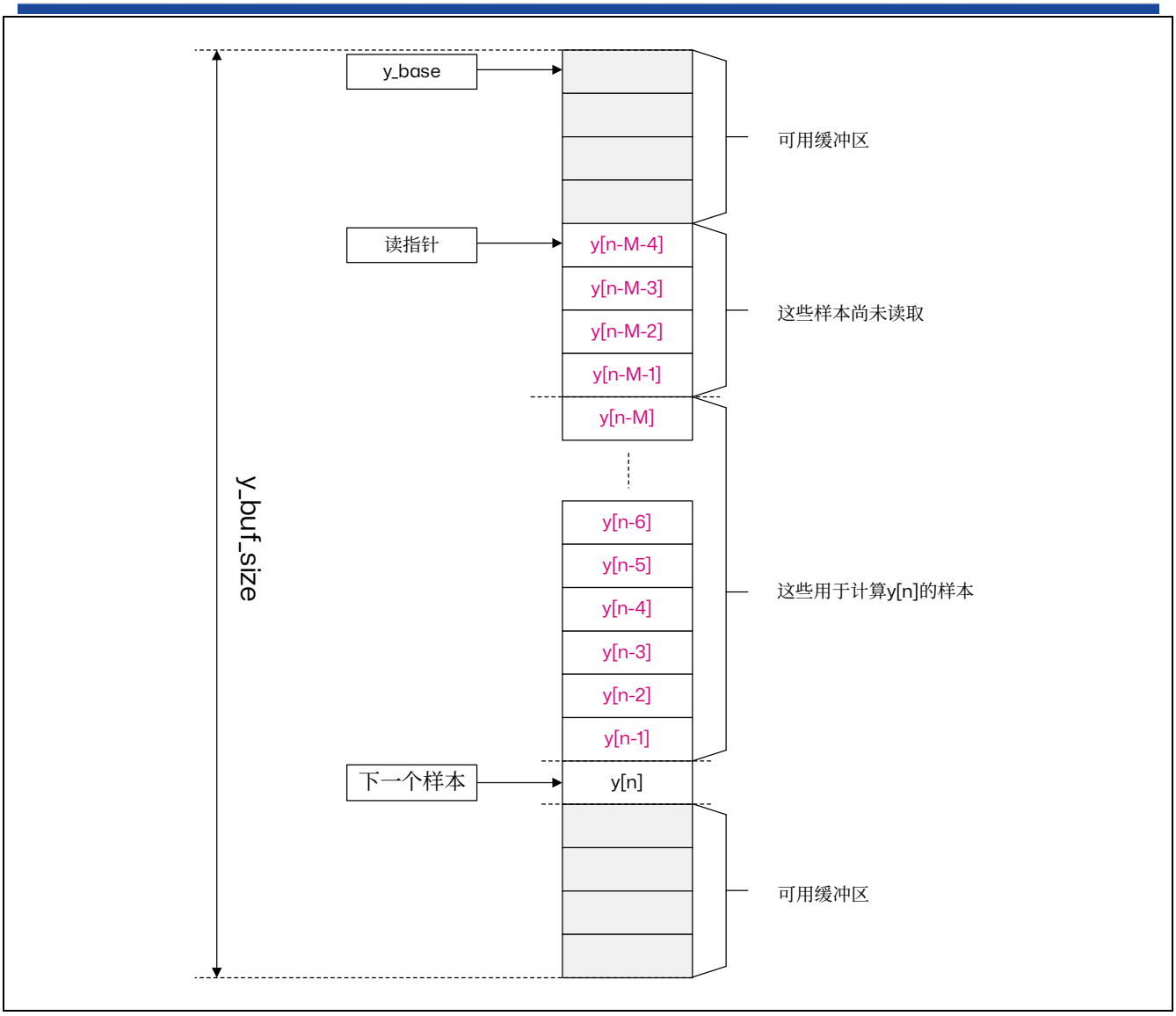



图 14.5 循环输出缓冲区

Y 缓冲区也可以作为循环缓冲区运行。如果下一个输出值的地址与读指针指示的地址相同（即存在一个未读样本），则缓冲区会被标记为满，并且执行将暂停，直到该样本被读取。

在 IIR 滤波器的情况下，Y 缓冲区用于存储用于计算下一个输出样本 y[n]的一组 M 个先前的输出样本，即 y[n-M]到 y[n-1]。每次向该组中添加新样本时，最早的样本 y[n-M]会被移除。

如果缓冲区中未读数据的数量小于 FMAC_YBUFCFG 寄存器的 EMPTY_WM 位域中编程设定的水位阈值，则缓冲区会被标记为空。只要未空标志未设置，如果使能，就会生成中断或 DMA 请求，以请求从缓冲区中读取数据。水位阈值允许在一次中断下传输多个数据，而不会发生下溢。然而，如果确实发生下溢，则会设置 UNFL 错误标志。在这种情况下，读指针不会递增，并且读操作将返回读指针地址处的内存内容。

下图展示了 Y 缓冲区在循环模式下的操作。此示例显示了一个 7 阶 IIR 滤波器，其水位阈值设置为 4。

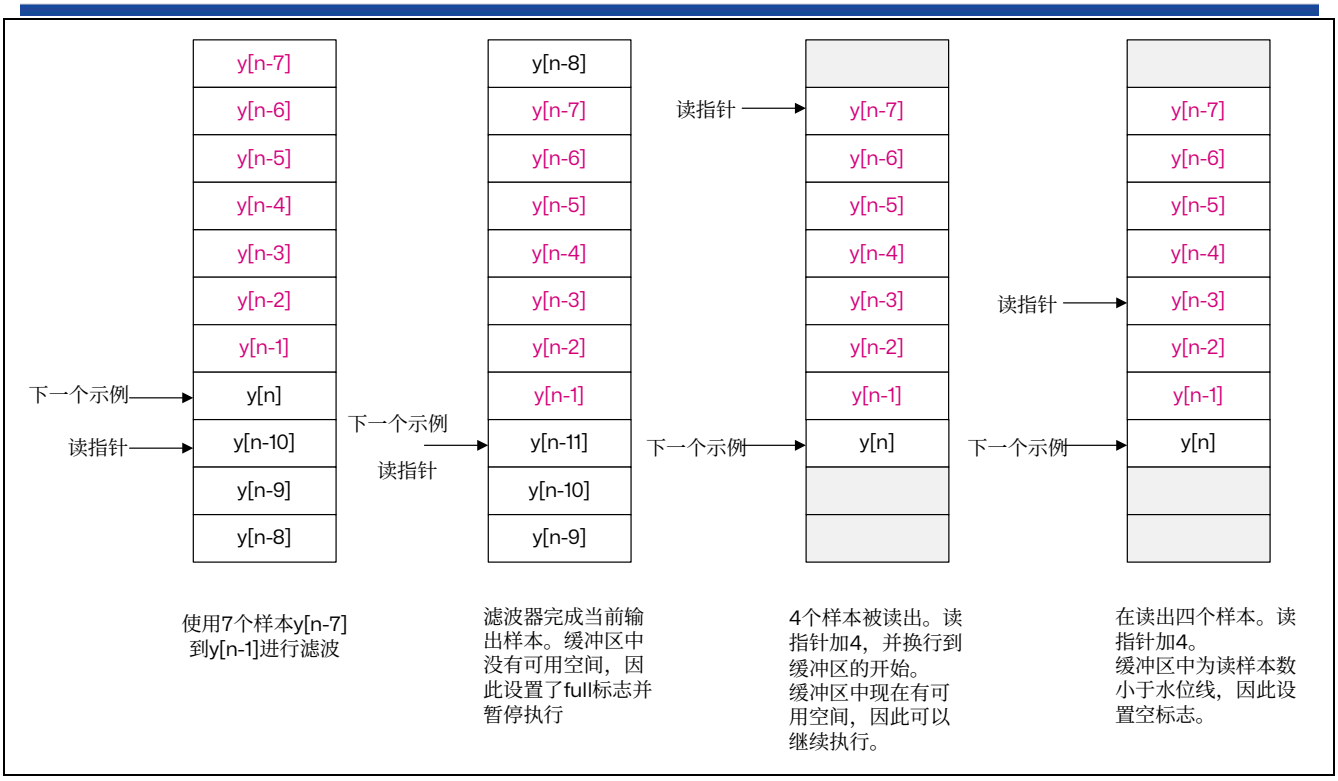



图 14.6 循环输出缓冲区操作

### 14.3.5 初始化函数

以下函数用于初始化 FMAC 单元。它们通过向 FMAC_PARAM 寄存器的 FUNC 位域写入适当的值并设置 START 位来触发。P 和 Q 位域也必须包含每个函数所需的适当参数值，如下所述。R 位域未使用。当函数完成时，START 位会由硬件自动复位。

在初始化期间，建议禁用 DMA 请求和中断。由于不需要流量控制，因此可以通过软件或内存到内存的 DMA 传输将数据传输到 FMAC 内存中。

#### 加载 X1 缓冲区

此函数从 X1_BASE 地址开始，预先将 N 个值加载到 X1 缓冲区中。连续写入 FMAC_WDATA 寄存器会将写入数据加载到 X1 缓冲区，并递增写入地址。当函数完成时，写入指针指向地址 X1_BASE + N。

该函数可用于预先将向量的元素加载到缓冲区中，或初始化滤波器的输入存储元素。

参数

- 参数 P 包含要加载到 X1 缓冲区中的值数 N。
- 参数 Q 和 R 未使用。

当向 FMAC_WDATA 寄存器执行了 N 次写入后，该函数即完成。

#### 加载 X2 缓冲区

此函数从 X2_BASE 地址开始，预先将 N + M 个值加载到 X2 缓冲区中。连续写入 FMAC_WDATA 寄存器会将写入数据加载到 X2 缓冲区，并递增写入地址。

该函数可用于预先将向量的元素或滤波器的系数加载到缓冲区中。在 IIR 滤波器的情况下，N 个前馈系数和 M 个反馈系数会被连接在一起，并共同加载到 X2 缓冲区中。系数的总数等于 N + M。对于 FIR 滤波器，没有反馈系数，因此 M = 0。

#### 参数

- 参数 P 包含从 X2_BASE 地址开始要加载到 X2 缓冲区中的值数 N。
- 参数 Q 包含从 X2_BASE + N 地址开始要加载到 X2 缓冲区中的值数 M。
- 参数 R 未使用。

当向 FMAC_WDATA 寄存器执行了 N + M 次写入后，该函数即完成。

#### 加载 Y 缓冲区

此函数从 Y_BASE 地址开始，预先将 N 个值加载到 Y 缓冲区中。连续写入 FMAC_WDATA 寄存器会将写入数据加载到 Y 缓冲区，并递增写入地址。当函数完成时，读指针指向地址 Y_BASE + N。

该函数可用于预先加载 IIR 滤波器的反馈存储元素。

#### 参数

- 参数 P 包含要加载到 Y 缓冲区中的值数。
- 参数 Q 和 R 未使用。

当向 FMAC_WDATA 寄存器执行了 N 次写入后，该函数即完成。

### 14.3.6 滤波函数

FMAC 单元支持以下滤波函数。这些函数通过向 FMAC_PARAM 寄存器的 FUNC 位域写入相应值并设置 START 位来触发。P、Q 和 R 位域也必须包含每个函数所需的适当参数值，如下所述。滤波函数将持续运行，直到 START 位被软件复位。

#### 卷积（FIR 滤波器）

$\underline{Y} = \underline{B} * \underline{X}$

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

此函数对长度为 N+1 的向量 $\underline{B}$ 和无限长度的向量 $\underline{X}$ 执行卷积运算。对于递增的 n 值，$\underline{Y}$ 的元素计算为点积，即 $y_n = \underline{B} \cdot \underline{X_n}$，其中 $\underline{X_n} = [x_{n-N},...,x_n]$ 由 $\underline{X}$ 中索引从 n-N 到 n 的 N+1 个元素组成。

此函数对应于有限冲激响应（FIR）滤波器，其中向量 B 包含滤波器系数，向量 X 包含采样数据。

滤波器的结构（直接形式）如下图所示。

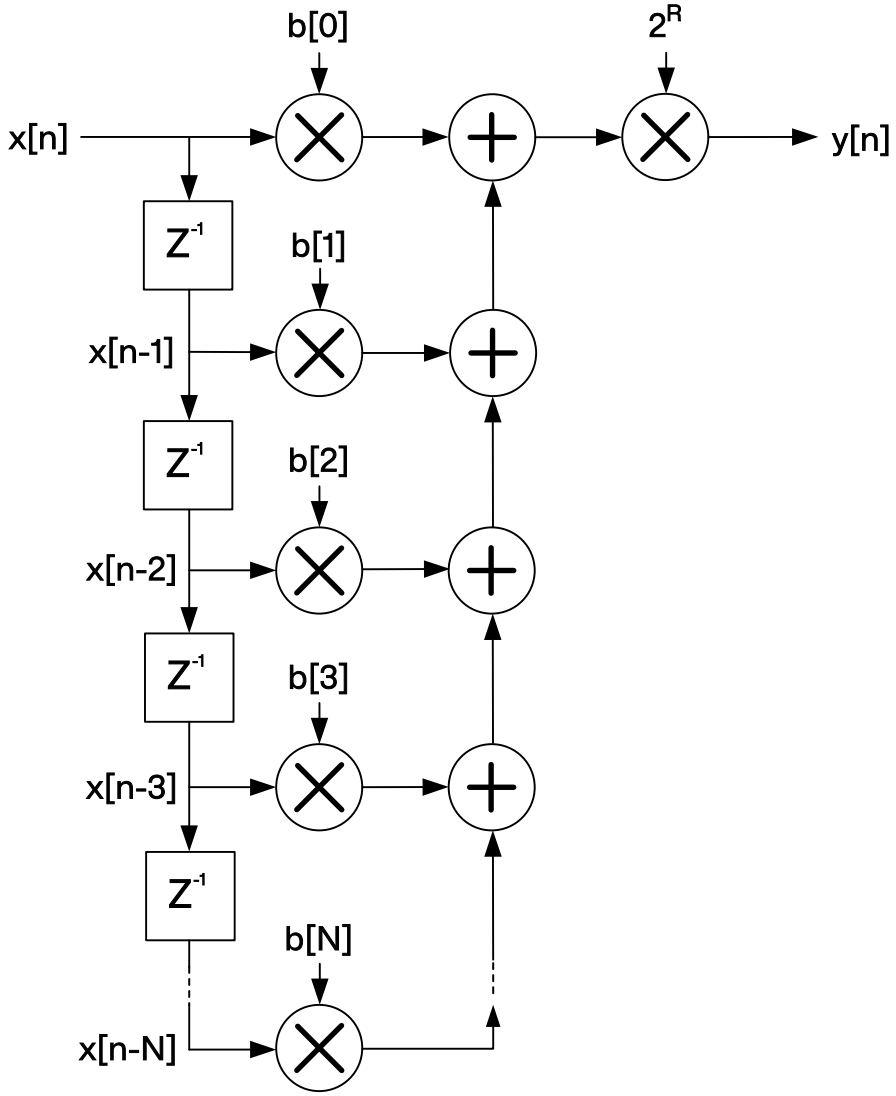



图 14.7 FIR 滤波器结构

请注意，通过颠倒系数向量 B 的顺序可以计算出互相关向量。

**输入：**

- X1 缓冲区包含向量 X 的元素。它是一个长度为 N + 1 + d 的环形缓冲区。
- X2 缓冲区包含向量 B 的元素。它是一个长度为 N + 1 的固定缓冲区。

**输出：**

- Y 缓冲区包含输出值 $y_n$。它是一个长度为 d 的环形缓冲区。

**参数：**

- 参数 P 包含系数向量 B 的长度 N+1，其取值范围为[2:127]。
- 参数 R 包含应用于累加器输出的增益。输出到 Y 缓冲区的值乘以 $2^R$，其中 R 的取值范围为[0:7]。
- 参数 Q 未使用。

当 FMAC_PARAM 寄存器中的 START 位被软件复位时，该函数完成。

### IIR 滤波器

Y = B*X+A*Y'

$$y_n = 2^R \cdot \left\{ \sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right\}$$

此函数实现了一个无限脉冲响应（IIR）滤波器。滤波器输出向量 Y 是长度为 N+1 的系数向量 B 与不定长向量 X 的卷积，加上延迟输出向量 Y'与长度为 M 的第二个系数向量 A 的卷积。对于递增的 n 值，Y 的元素计算为 $y_n$ = B.$X_n$ + A.$Y_{n-1}$，其中 Xn = [$x_{n-N}$,...,$x_n$]包含索引 n-N 到 n 处 X 的 N+1 个元素，而 $Y_{n-1}$ = [$y_{n-M}$,...,$y_{n-1}$]包含索引 n-M 到 n-1 处 Y 的 M 个元素。滤波器结构（直接形式 1）如下图所示。

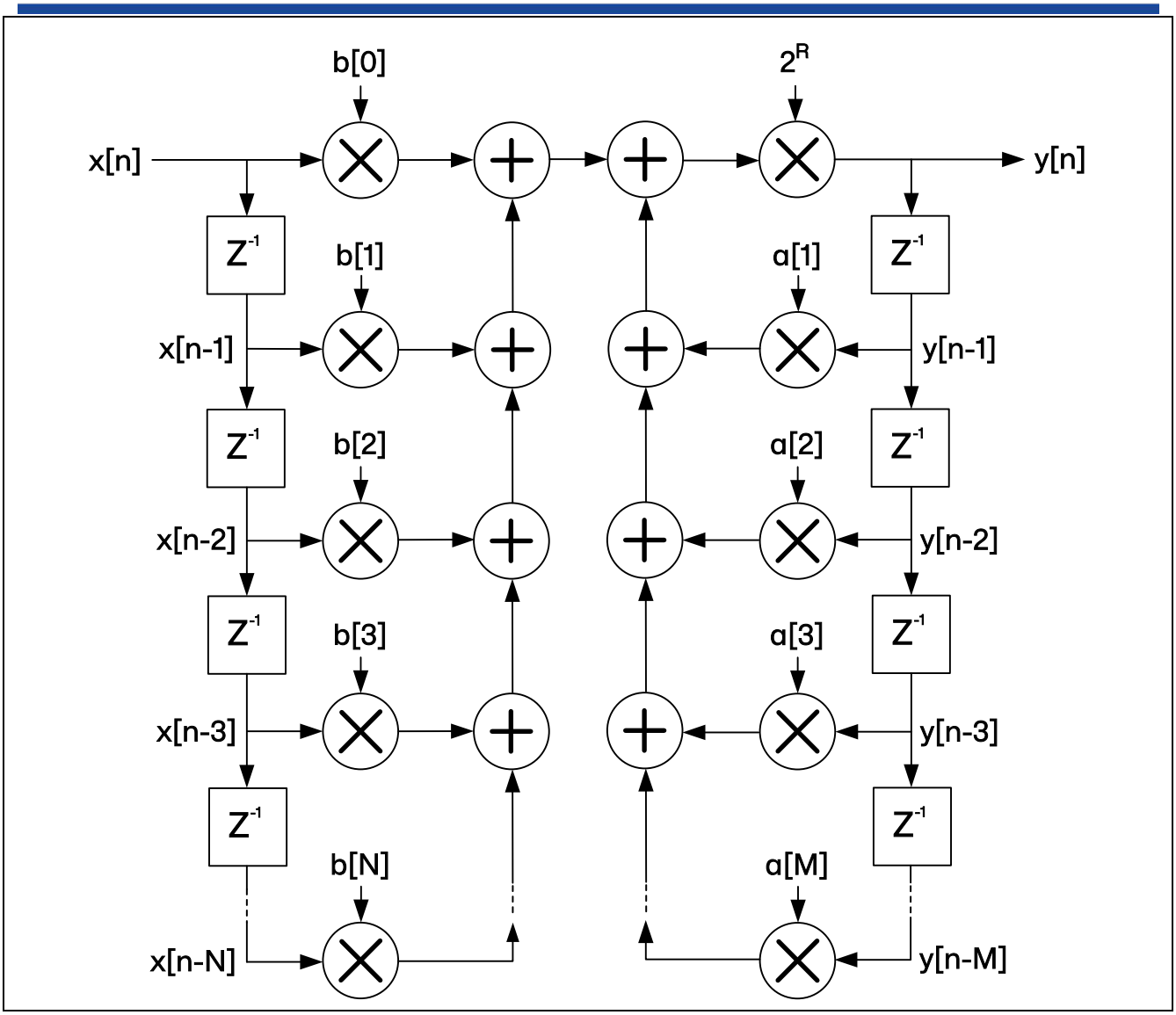



图 14.8 IIR 滤波器结构（直接 1 型）

**输入：**

- X1 缓冲区包含向量 X 的元素。它是一个长度为 N+1+d 的循环缓冲区。
- X2 缓冲区包含连接在一起的系数向量 B 和 A 的元素（$b_0$, $b_1$, $b_2$..., $b_N$, $a_1$, $a_2$, ..., $a_M$）。它是一个长度为 M+N+1 的固定缓冲区。

**输出：**

- Y 缓冲区包含输出值 $y_n$。它是一个长度为 M+d 的循环缓冲区。

**参数：**

- 参数 P 包含系数向量 B 的长度 N+1，其取值范围为[2:64]
- 参数 Q 包含系数向量 A 的长度 M，其取值范围为[1:63]。
- 参数 R 包含应用于累加器输出的增益。输出到 Y 缓冲区的值乘以 $2^R$，其中 R 的取值范围为[0:7]。

该函数在 FMAC_PARAM 寄存器的 START 位被软件复位时完成。

### 14.3.7 固定点表示法

FMAC 以固定点有符号整数格式运行。输入和输出值为 q1.15 格式。

在 q1.15 格式中，数字由一个符号位和 15 个分数位（二进制小数点后的位）表示。因此，数值范围为 -1（0x8000）到 $1 - 2^{-15}$（0x7FFF）。

累加器具有 26 位，其中 22 位为分数位，4 位为整数/符号位（q4.22）。这使其能够支持范围在-8（0x2000000）到+7.99999976（0x1FFFFFF）之间的部分累加和。可在累加器输出端应用以 6dB 为步长的 0dB 到 42dB 之间的可编程增益。

请注意，如果数值范围超出，累加器的内容不会饱和。值大于+7.99999976 或小于-8 的部分和会发生溢出，但只要后续的累加操作能够抵消这种溢出，就是无害的。然而，如果发生溢出，FMAC_SR 寄存器中的 SAT 标志会被设置，并且如果 FMAC_CR 寄存器中的 SATIEN 位被设置，则会生成中断。这有助于滤波器的调试。

累加器输出的数据在应用可编程增益后，可以通过设置 FMAC_CR 寄存器中的 CLIPEN 位来选择性地进行饱和处理。如果设置了此位，则任何超出 q1.15 输出数值范围的值，都会根据符号被设置为 $1 - 2^{-15}$ 或-1。如果未使能饱和处理，则在应用增益后，累加器中未使用的位将被简单截断。

### 14.3.8 使用 FMAC 实现 FIR 滤波器

FMAC 支持长度为 N 的 FIR 滤波器，其中 N 为阶数或系数的数量。长度为 N 的 FIR 滤波器所需的最小本地内存为 2N + 1:

- N 个系数
- N 个输入样本
- 1 个输出样本

由于本地内存大小为 256，因此 N 的最大值为 127。

如果需要最大吞吐量，可能需要分别为输入和输出样本缓冲区分配少量的额外空间 d1 和 d2，以确保滤波器永远不会因等待新的输入样本或等待输出样本被读取而停滞。在这种情况下，本地内存需求为 2N + d1 + d2。

缓冲区必须按以下方式配置:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2（如果不需要额外空间，则为 1）

缓冲区基地址可以在任意位置分配，但 X2 缓冲区不得与其他缓冲区重叠，否则系数将被覆盖。一个示例配置为:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

然而，如果内存空间有限，X1 和 Y 缓冲区区域可以重叠，这样每个输出样本就会取代最旧的、不再需要的输入样本：

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

在这种情况下，Y_BUF_SIZE = X1_BUF_SIZE = N + d1，以保持缓冲区同步。

*注：X1 缓冲区配置寄存器的*FULL_WM *位位域必须编程为一个小于或等于*$\log_2$(d1)*的值，否则在写入 N 个输入样本之前，缓冲区就会被标记为已满，并且不会再请求更多样本。类似地，Y 缓冲区配置寄存器的*EMPTY_WM *位位域必须小于或等于*$\log_2$(d2)。

滤波器系数**必须**使用加载 X2 缓冲区功能预先加载到 X2 缓冲区中。X1 缓冲区可以选择性地预先加载最多 N 个样本。预先加载 Y 缓冲区没有意义，因为对于 FIR 滤波器而言，没有反馈路径。

在配置和初始化缓冲区后，必须根据用于向 FMAC 内存写入和读取数据的方法对 FMAC_CR 寄存器进行编程。

支持三种方法：

- 轮询：不生成 DMA 请求或中断请求。软件必须检查 X1_FULL 标志是否为低电平，然后才能向 WDATA 写入，或者检查 Y_EMPTY 标志是否为低电平，然后才能从 RDATA 读取。
- 中断：在写入时，当 X1_FULL 标志为低电平时，或者在读取时，当 Y_EMPTY 标志为低电平时，发出中断请求。
- DMA：在 X1_FULL 标志为低电平时，DMA 请求在 DMA 写入通道上被断言；在 Y_EMPTY 标志为低电平时，在读取通道上被断言。

对于读取和写入可以使用不同的方法。但是，不建议对同一操作同时使用中断和 DMA 请求[a]。有效的组合列在下表中。

表 14.1 读取和写入方法的有效组合

| WIEN | RIEN | DMAWEN | DMAREN | 写入 | 读取 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 轮询 | 轮询 |
| 0 | 1 | 0 | 0 | 轮询 | 中断 |
| 1 | 0 | 0 | 0 | 中断 | 轮询 |
| 1 | 1 | 0 | 0 | 中断 | 中断 |
| 0 | 0 | 0 | 1 | 轮询 | DMA |
| 0 | 0 | 1 | 0 | DMA | 轮询 |
| 0 | 0 | 1 | 1 | DMA | DMA |
| 0 | 1 | 1 | 0 | DMA | 中断 |
| 1 | 0 | 0 | 1 | 中断 | DMA |

a. 如果同时使能了中断和 DMA 请求，则只有 DMA 可以执行传输。

通过向 FMAC_PARAM 寄存器写入以下位域值来启动滤波器：

- FUNC = 8（FIR 滤波器）；
- P = N（系数数量）；
- Q = “无关”；
- R = 增益；
- START = 1；

如果 X1 缓冲区中预加载的值少于 N + d - $2^{FULL_WM}$，则 X1FULL 标志保持低电平。如果 FMAC_CR 寄存器中设置了 WIEN 位，则会立即断言中断请求，要求处理器通过 FMAC_WDATA 寄存器向缓冲区写入 $2^{FULL_WM}$ 个额外样本。该断言一直保持到 FMAC_SR 寄存器中的 X1FULL 标志变为高电平。中断服务例程必须在每次向 FMAC_WDATA 寄存器写入 $2^{FULL_WM}$ 后检查 X1FULL 标志，并重复传输直到标志变为高电平。类似地，如果 FMAC_CR 寄存器中设置了 DMAWEN 位，则会生成 DMA 写入通道请求，直到 X1FULL 标志变为高电平。

当至少 N 个样本（包括任何预加载的样本）已写入 X1 缓冲区时，滤波器计算第一个输出样本。

当 $2^{EMPTY_WM}$ 个输出样本已写入 Y 缓冲区时，FMAC_SR 寄存器中的 YEMPTY 标志变为低电平。如果 FMAC_CR 寄存器中设置了 RIEN 位，则会断言中断请求，要求处理器通过 FMAC_RDATA 寄存器从缓冲区读取 $2^{EMPTY_WM}$ 个样本。该断言一直保持到 YEMPTY 标志变为高电平。中断服务例程必须在每次从 FMAC_RDATA 寄存器读取 $2^{EMPTY_WM}$ 后检查 YEMPTY 标志，并重复传输直到标志变为高电平。如果 FMAC_CR 中设置了 DMAREN 位，则会生成 DMA 读取通道请求，直到 YEMPTY 标志变为高电平。

滤波器以这种方式持续运行，直到软件通过重置 START 位来停止它。

### 14.3.9 使用 FMAC 实现 IIR 滤波器

FMAC 支持长度为 N 的 IIR 滤波器，其中 N 是前馈阶数或系数的数量。反馈系数的数量 M 可以是 1 到 N-1 之间的任何值。只能实现直接形式 1 的实现，因此为其他形式设计的滤波器需要进行转换。

具有 N 个前馈系数和 M 个反馈系数的 IIR 滤波器的最小内存需求为 2N + 2M：

- N + M 个系数
- N 个输入样本
- M 个输出样本

如果 M = N-1，则可以实现的最大滤波器长度为 N = 64。

与 FIR 类似，为了最大吞吐量，输入和输出缓冲区大小分别允许有少量额外空间 d1 和 d2，使得总内存需求为 2M + 2N + d1 + d2。

缓冲区必须按以下方式配置：

- X1_BUF_SIZE = N + d1；
- X2_BUF_SIZE = N + M；
- Y_BUF_SIZE = M + d2；

缓冲区基地址可以分配在任何位置，但不得重叠。以下是一个示例配置：

- X2_BASE = 0；
- X1_BASE = N + M；
- Y_BASE = 2N + M + d1；

*注：X1 缓冲区配置寄存器的*FULL_WM *位位域必须编程为一个小于或等于*$\log_2$(d1)*的值，否则在写入 N 个输入样本之前，缓冲区就会被标记为满，并且不会再请求更多样本。类似地，Y 缓冲区配置寄存器的* EMPTY_WM *位位域必须小于或等于*$\log_2$(d2)。

滤波器系数（N 个前馈系数后跟 M 个反馈系数）**必须**使用加载 X2 缓冲区功能预加载到 X2 缓冲区中。X1 缓冲区可以选择性地预加载最多 N 个样本。Y 缓冲区可以选择性地预加载最多 M 个值。这具有初始化反馈延迟线的效果。

配置完缓冲区后，必须按照与 FIR 滤波器相同的方式对 FMAC_CR 寄存器进行编程（见上一节：使用 FMAC 实现 FIR 滤波器）。

通过向 FMAC_PARAM 寄存器写入以下位位域值来启动滤波器：

- FUNC = 9（IIR 滤波器）；
- P = N（前馈系数数量）；
- Q = M（反馈系数数量）；
- R = 增益；
- START = 1；

如果 X1 缓冲区中预加载的值少于 $N + d - 2^{FULL_WM}$，则 X1FULL 标志保持低电平。如果 FMAC_CR 寄存器中设置了 WIEN 位，则会立即断言中断请求，要求处理器通过 FMAC_WDATA 寄存器向缓冲区写入 $2^{FULL_WM}$ 个额外样本。该断言一直保持到 FMAC_SR 寄存器中的 X1FULL 标志变为高电平。中断服务例程必须在每次向 FMAC_WDATA 寄存器写入 $2^{FULL_WM}$ 后检查 X1FULL 标志，并重复传输直到标志变为高电平。类似地，如果 FMAC_CR 寄存器中设置了 DMAWEN 位，则会生成 DMA 写入通道请求，直到 X1FULL 标志变为高电平。

当至少 N 个样本（包括任何预加载的样本）已写入 X1 缓冲区时，滤波器计算第一个输出样本。第一个样本是使用 X1 缓冲区中的前 N 个样本和 Y 缓冲区中的前 M 个样本（无论是否预加载）计算得出的。第一个输出样本被写入 Y 缓冲区的 Y_BASE + M 位置。

当 $2^{EMPTY_WM}$ 个新输出样本已写入 Y 缓冲区时，FMAC_SR 寄存器中的 YEMPTY 标志变为低电平。如果 FMAC_CR 寄存器中设置了 RIEN 位，则会断言中断请求，要求处理器通过 FMAC_RDATA 寄存器从缓冲区读取 $2^{EMPTY_WM}$ 个样本。该断言一直保持到 YEMPTY 标志变为高电平。中断服务例程必须在每次从 FMAC_RDATA 寄存器读取 $2^{EMPTY_WM}$ 后检查 YEMPTY 标志，并重复传输直到标志变为高电平。如果 FMAC_CR 中设置了 DMAREN 位，则会生成 DMA 读取通道请求，直到 YEMPTY 标志变为高电平。

滤波器以这种方式持续运行，直到软件通过重置 START 位来停止它。

### 14.3.10 滤波器初始化示例

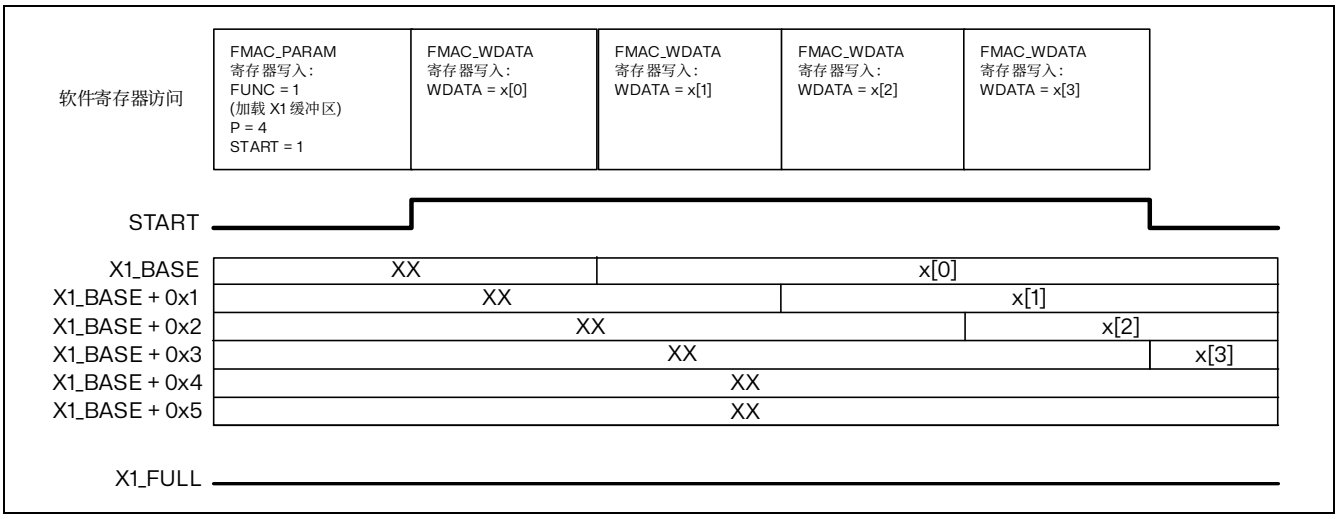



图 14.9 X1 缓冲区初始化

上图中的示例展示了 X1 缓冲区预加载四个样本（P=4）的情况。缓冲区大小为六（X1_BUF_SIZE=6）。初始化通过编程设置 FMAC_PARAM 寄存器的 START 位来启动。然后，将四个样本写入 FMAC_WDATA，并从 X1_BASE 开始传输到本地内存。写入第四个样本后，START 位复位。此时，X1 缓冲区按写入顺序包含这四个样本，且写指针（下一个空闲位置）位于 X1_BASE + 0x4。

### 14.3.11 滤波器操作示例

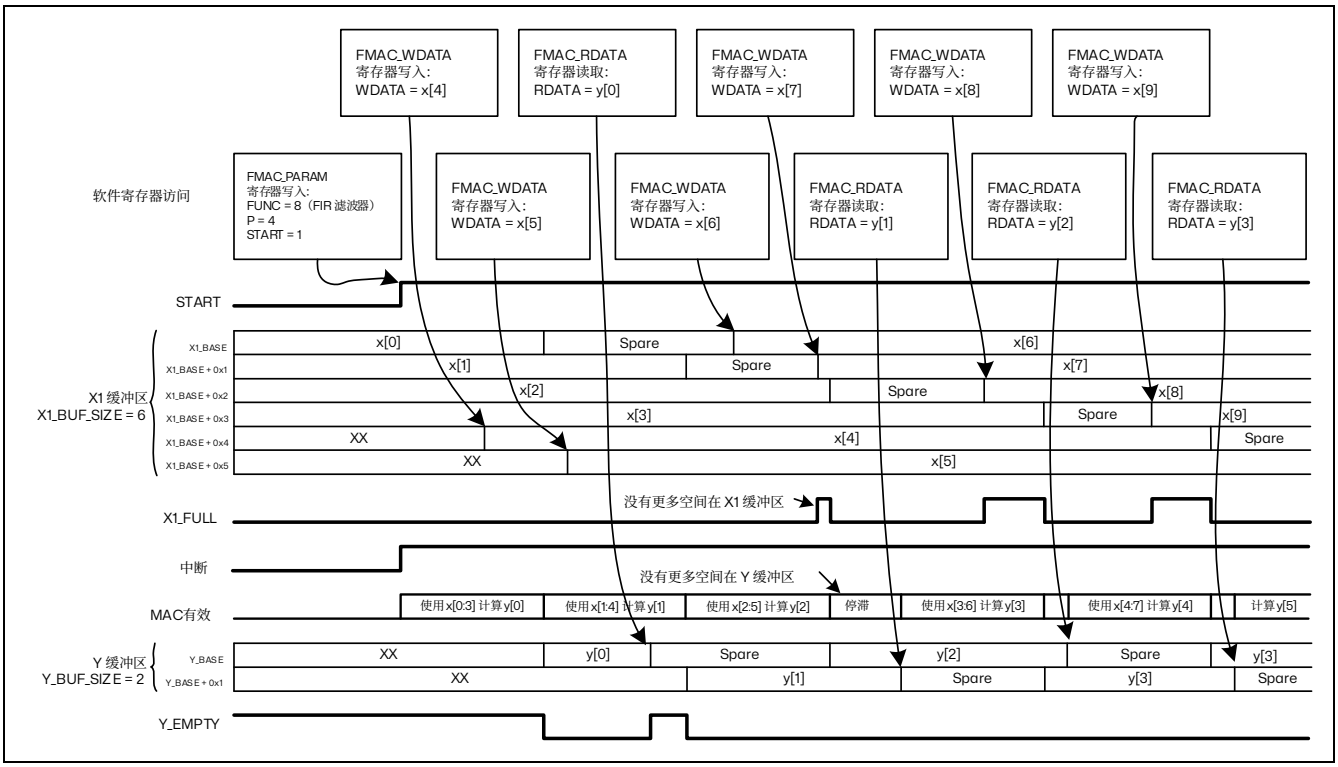



图 14.10 滤波示例 1

图 15.10 中的示例展示了滤波器操作的开始阶段。该滤波器有四阶（P=4）。X1 缓冲区大小为六，Y 缓冲区大小为二。FULL_WM 和 EMPTY_WM 位位域均设置为 0。在启动滤波器之前，X1 缓冲区已预加载了四个样本，即 x[0:3]（如图 15.9 所示）。因此，在设置 START 位之后，滤波器立即开始计算第一个输出样本 y[0]。由于 X1FULL 标志未设置（由于 X1 缓冲区中存在两个未初始化的空间），中断立即被触发，以请求新数据。处理器将两个新样本 x[4]和 x[5]写入 FMAC_WDATA 寄存器，这些样本随后被传输到 X1 缓冲区中的空闲位置。

与此同时，FMAC 完成第一个输出样本 y[0]的计算，并将其写入 Y 缓冲区，导致 Y_EMPTY 标志变为低电平。同时，样本 x[0]被丢弃，因为它不再需要，从而释放了其在内存中的位置（位于 X1_BASE）。由于 X1 缓冲区中存在所有所需的输入样本 x[1:5]，FMAC 可以立即开始处理第二个输出样本 y[1]。

由于 Y_EMPTY 标志为低电平，处理器在写完 x[5]后中断仍保持活动状态。处理器从 FMAC_RDATA 寄存器中读取 y[0]，从而释放了其在 Y 缓冲区中的位置。此时，输出缓冲区中没有样本，因为 y[1]仍在计算中，所以 Y_EMPTY 标志变为高电平。然而，中断仍然保持活动状态，因为 X1 缓冲区中仍有空闲空间，处理器接下来将用 x[6]填充该空间，依此类推。

*注：在此示例中，处理器填充输入缓冲区的速度比 FMAC 处理它们的速度快，因此 X1_full 标志会定期变为活动状态。然而，处理器读取 Y 缓冲区的速度不够快，因此 FMAC 会定期停滞，等待 Y 缓冲区中释放空间。这意味着滤波器并未以最大吞吐量运行。原因是，在此示例中，滤波器长度较小，而处理器相对较慢。因此，增加 Y 缓冲区大小并无助益。*

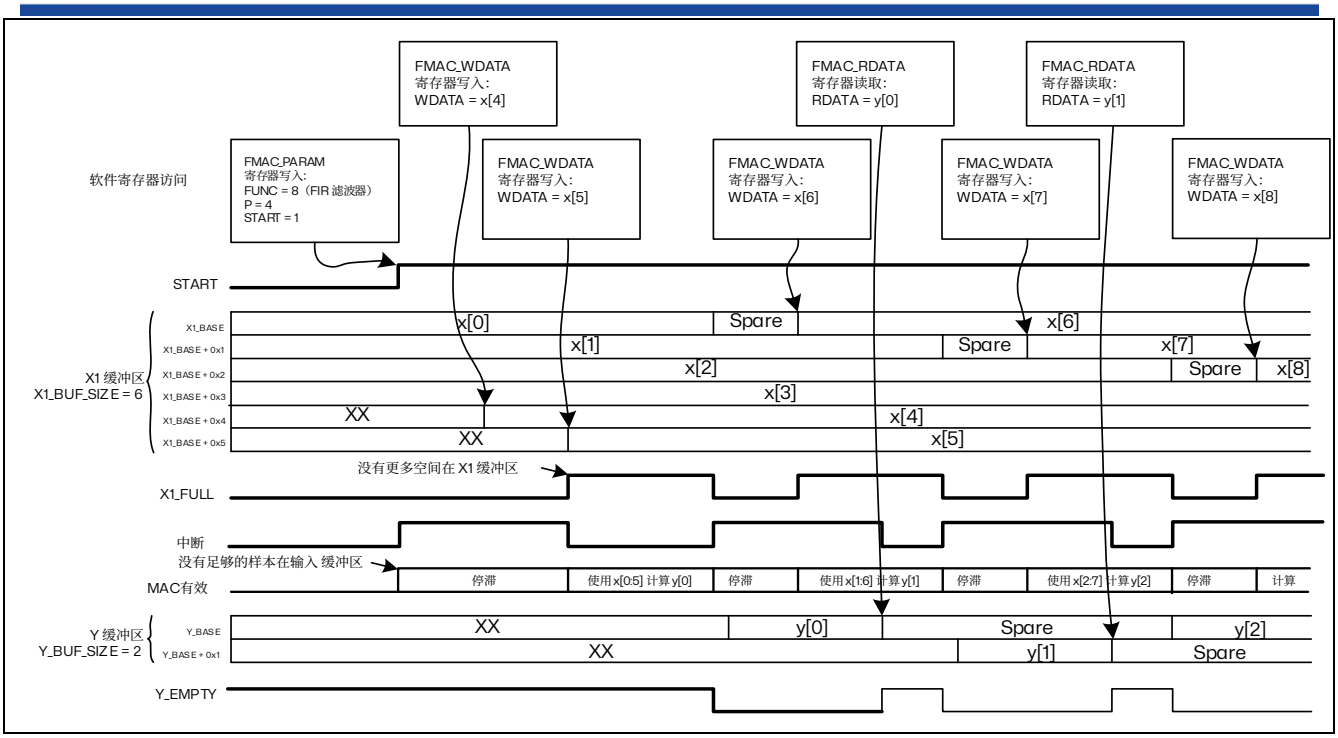



图 14.11 滤波示例 2

图 15.11 中的示例展示了相同滤波器操作的开始阶段，但此次滤波器有六阶（P=6）。X1 缓冲区大小为六，Y 缓冲区大小为二。FULL_WM 和 EMPTY_WM 位位域均设置为 0。在启动滤波器之前，X1 缓冲区已预加载了四个样本，即 x[0:3]（如 15.9 所示）。由于输入缓冲区中的样本数量不足，X1FULL 标志未设置，因此中断立即被触发，以请求新数据。此时，FMAC 处于停滞状态。

处理器将两个新样本 x[4]和 x[5]写入 FMAC_WDATA 寄存器，这些样本随后被传输到 X1 缓冲区中的空闲位置。一旦 X1 缓冲区中有六个未使用的样本（因为缓冲区大小为六），X1_FULL 标志就会变为活动状态，导致中断失效。FMAC 开始计算第一个输出样本 y[0]。由于这需要所有六个输入样本，X1 缓冲区中没有空闲空间，因此 X1_FULL 标志保持活动状态。只有当 FMAC 计算完 y[0]并将其写入 Y 缓冲区后，x[0]才能被丢弃，从而释放 X1 缓冲区中的一个空间，并使 X1_FULL 标志失效。同时，Y_EMPTY 标志也变为无效状态。这两个标志状态都会导致中断被触发，首先请求处理器写入一个新的输入样本，然后读取刚刚计算出的输出样本。在写入新的输入样本之前，FMAC 会保持停滞状态。

在此示例中，处理器必须等待 FMAC 完成当前输出样本的计算，然后才能写入新的输入样本。因此，X1 缓冲区经常会变空，导致 FMAC 停滞。这可以通过在输入缓冲区中留出一些额外空间来避免。

### 14.3.12 滤波器设计提示

FMAC 架构对数字滤波器的设计施加了一些约束，具体细节如下：

1. 直接形式 2 或转置形式的实现效率不高。为此类形式设计的滤波器必须转换为直接形式 1。
2. 级联滤波器必须要么合并为一个单级，要么作为单独的滤波器实现。在后一种情况下，可以将多组滤波器系数预加载到内存中，每级一组，并且仅通过更改 X2_BASE 地址来选择使用哪一组。实现多级滤波器最有效的方法是首先用输入样本预加载一个大的 X1 缓冲区，然后使用第一级系数对其运行 IIR 滤波器函数，并将输出样本存储回内存中。然后更改 X2_BASE 指针以指向第二级系数，并在再次运行 IIR 函数之前，用第一级的输出（如果需要，则加上增益）重新加载输入缓冲区。对所有级重复此过程。一旦最后一级的样本被传回系统内存，就可以将输入缓冲区加载下一组输入样本，并开始新一轮的计算。请注意，为了保持每轮之间的连续性，必须首先用上一轮的最后 N-1 个输入加上一个新样本预加载每级的 N 个样本输入缓冲区。同样，出于同样的原因，每级的输出缓冲区必须加载上一轮的最后 M 个样本。
3. 如果 IIR 设计使用直接形式 1，那么在输入出现大步长或某些滤波器系数的绝对值大于 1 的情况下，累加器中可能会产生大的正或负部分和。由于累加器仅限于 26 位，因此它在不溢出（改变符号）的情况下可以处理的最大值为正 0x1FFFFFF 或负 0x2000000。这对应于 q3.23 定点格式中的 3.99999988 和-4。如果在累加结束之前能够“撤销”溢出，那么溢出就不会成为问题。然而，在滤波器启动时并非总是如此，这可能会导致意外的结果。考虑预加载输出缓冲区以合适的值来避免这种情况。
4. IIR 滤波器具有前馈（分子）系数$[b_0, b_1, ..., b_{N-1}]$和反馈（分母）系数$[1, a_1, ..., a_M]$。许多 IIR 滤波器要求某些分母系数的绝对值大于 1，以实现频率响应中的陡峭滚降。鉴于系数采用 q1.15 定点格式编码，这是不可能的。然而，通过将分母系数按 $2^{-R}$ 因子缩放，使得 $2^{-R}.[1, a1, ..., aM]$均小于 1，可以实现此类滤波器。但是，必须在累加器的输出处应用 $2^R$ 的逆增益以补偿缩放。这对信噪比有不利影响。

## 14.4 FMAC 寄存器描述

### 14.4.1 FMAC X1 缓冲区配置寄存器（FMAC_X1BUFCFG）

偏移地址：0x00
复位值：0x0000 0000
访问：字访问
仅当 FMAC_PARAM 寄存器中的 START = 0 时，才能修改此寄存器。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | FULL_WM[1:0] | | Reserved | | | | | | | |
| | | | | | | rw | rw | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X1_BUF_SIZE[7:0] | | | | | | | | X1_BASE[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:26 保留，必须保持为复位值。

Bit 25:24 **FULL_WM[1:0]**：缓冲区满标志的水位线

在循环模式下操作时，定义设置 X1 缓冲区满标志的阈值。如果缓冲区中的空闲空间数量小于 $2^{FULL_WM}$，则设置该标志。

0：阈值 = 1

1：阈值 = 2

2：阈值 = 4

3：阈值 = 8

设置大于 1 的阈值允许在一次中断下将多个数据传输到缓冲区。

如果使能了 DMA 写请求（FMAC_CR 寄存器中的 DMAWEN = 1），则必须将阈值设置为 1。

Bit 23:16 保留，必须保持为复位值。

Bit 15:8 **X1_BUF_SIZE[7:0]**：X1 缓冲区以 16 位字为单位分配的大小

最小缓冲区大小为滤波器中的前馈阶数（+水位线阈值-1）。

Bit 7:0 **X1_BASE[7:0]**：X1 缓冲区的基地址

### 14.4.2 FMAC X2 缓冲区配置寄存器（FMAC_X2BUFCFG）

偏移地址：0x04
复位值：0x0000 0000
访问：字访问

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X2_BUF_SIZE[7:0] | | | | | | | | X2_BASE[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持为复位值。

Bit 15:8 **X2_BUF_SIZE[7:0]**：X2 缓冲区以 16 位字为单位分配的大小
当函数正在执行时（START = 1），不能修改此位域。

Bit 7:0 **X2_BASE[7:0]**： X2 缓冲区的基地址
当 START=1 时，可以修改 X2 缓冲区的基地址，例如更改系数值。在进行此操作时，滤波器必须停止工作，因为在计算过程中更改系数会影响结果。

### 14.4.3 FMAC_Y buffer 配置寄存器（FMAC_YBUFCFG）

偏移地址：0x08
复位值：0x0000 0000
访问：字访问
此寄存器仅当 FMAC_PARAM 寄存器中的 START = 0 时，才能进行修改。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | EMPTY_WM[1:0] | | Reserved | | | | | | | |
| | | | | | | rw | rw | | | | | | | | |

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

Bit 31:26 保留，必须保持为复位值。

Bit 25:24 **EMPTY_WM[1:0]**：缓冲区空标志的水位线
在循环模式下操作时，定义设置 Y 缓冲区空标志的阈值。如果缓冲区中的未读值数量小于 $2^{EMPTY_WM}$，则设置该标志。
0：阈值 = 1
1：阈值 = 2
2：阈值 = 4
3：阈值 = 8
设置大于 1 的阈值允许在一次中断下从缓冲区传输多个数据。
如果使能了 DMA 读请求（FMAC_CR 寄存器中的 DMAREN = 1），则必须将阈值设置为 1。

Bit 23:16 保留，必须保持为复位值。

Bit 15:8 **Y_BUF_SIZE[7:0]**：Y 缓冲区以 16 位字为单位分配的大小
对于 FIR 滤波器，最小缓冲区大小为 1（+水位线阈值）。对于 IIR 滤波器，最小缓冲区大小为滤波器中的前馈阶数（+水位线阈值）。

Bit 7:0 **Y_BASE[7:0]**：Y 缓冲区的基地址

### 14.4.4 FMAC 参数寄存器（FMAC_PARAM）

偏移地址：0x0C
复位值：0x0000 0000
访问：字访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| START | FUNC[6:0] | | | | | | | R[7:0] | | | | | | | |
| 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 |
| Q[7:0] | | | | | | | | P[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **START**：执行使能

0：停止执行

1：开始执行

设置此位将触发在 FUNC 位域中选择的函数的执行。通过软件复位此位可以停止任何正在进行的函数。对于初始化函数，此位由硬件复位。

Bit 30:24 **FUNC[6:0]**：函数

0：保留

1：加载 X1 缓冲区

2：加载 X2 缓冲区

3：加载 Y 缓冲区

4 至 7：保留

8：卷积（FIR 滤波器）

9：IIR 滤波器（直接 1 型）

10 至 127：保留

当函数正在执行时（START = 1），不能修改此位域。

Bit 23:16 **R[7:0]**：输入参数 R

此参数的值取决于函数。

当函数正在执行时（START = 1），不能修改此位域。

Bit 15:8 **Q[7:0]**：输入参数 Q

此参数的值取决于函数。

当函数正在执行时（START = 1），不能修改此位域。

Bit 7:0 **P[7:0]**：输入参数 P

此参数的值取决于函数。

当函数正在执行时（START = 1），不能修改此位域。

### 14.4.5 FMAC 控制寄存器（FMAC_CR）

偏移地址：0x10
复位值：0x0000 0000
访问：字访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | RESET |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLIP EN | Reserved | | | | | DMA WEN | DMA REN | Reserved | | | SAT IEN | UNFL IEN | OVFL IEN | WIEN | RIEN |
| rw | | | | | | rw | rw | | | | rw | rw | rw | rw | rw |

Bit 31:17 保留，必须保持为复位值。

Bit 16 **RESET**：复位 FMAC 模块

此位复位写指针和读指针、内部控制逻辑、FMAC_SR 寄存器和 FMAC_PARAM 寄存器，包括（如果处于活动状态）START 位。其他寄存器设置不受影响。此位由硬件复位。

0：复位无效

1：复位有效

Bit 15 **CLIPEN**：使能限幅

0：限幅禁用。累加器输出值超出 q1.15 范围时，将发生回滚。

1：限幅使能。累加器输出值超出 q1.15 范围时，将根据符号饱和到最大正值或负值（+1 或-1）。

Bit 14:10 保留，必须保持为复位值。

Bit 9 **DMAWEN**：使能 DMA 写通道请求

0：禁用。不生成 DMA 请求。

1：使能。当 X1 缓冲区未满时，生成 DMA 请求。

此位仅当 FMAC_PARAM 寄存器中的 START=0 时，才能进行修改。读取时返回此位的当前状态。

Bit 8 **DMAREN**：使能 DMA 读通道请求

0：禁用。不生成 DMA 请求。

1：使能。当 Y 缓冲区不为空时，生成 DMA 请求。

此位仅当 FMAC_PARAM 寄存器中的 START=0 时，才能进行修改。读取时返回此位的当前状态。

Bit 7:5 保留，必须保持为复位值。

Bit 4 **SATIEN**：饱和错误中断使能

0：禁用。检测到饱和时不生成中断。

1：使能。如果 SAT 标志设置，则生成中断请求。

此位由软件设置和清除。读取时返回此位的当前状态。

Bit 3 **UNFLIEN**：下溢错误中断使能

0：禁用。检测到下溢时不生成中断。

1：使能。如果 UNFL 标志设置，则生成中断请求。

此位由软件设置和清除。读取时返回此位的当前状态。

Bit 2 **OVFLIEN：上溢**错误中断使能

0：禁用。检测到溢出时不生成中断。

1：使能。如果 OVFL 标志设置，则生成中断请求。

此位由软件设置和清除。读取时返回此位的当前状态。

Bit 1 **WIEN：**写中断使能

0：禁用。不生成写中断请求。

1：使能。当 X1 缓冲区 FULL 标志未设置时，生成中断请求。

此位由软件设置和清除。读取时返回此位的当前状态。

Bit 0 **RIEN：**读中断使能

0：禁用。不生成读中断请求。

1：使能。当 Y 缓冲区 EMPTY 标志未设置时，生成中断请求。

此位由软件设置和清除。读取时返回此位的当前状态。

### 14.4.6 FMAC 状态寄存器（FMAC_SR）

偏移地址：0x14

复位值：0x0000 0001

访问：字访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | SAT | UNFL | OVFL | Reserved | | | | | | X1 FULL | Y EMPTY |
| | | | | | r | r | r | | | | | | | r | r |

Bit 31:11 保留，必须保持为复位值。

Bit 10 **SAT：**饱和错误标志

当累加结果超出累加器的数值范围时，会发生饱和。

0：未检测到饱和

1：检测到饱和。如果 SATIEN 位设置，则生成中断。

此标志通过复位模块来清除。

Bit 9 **UNFL：**下溢错误标志

当从 FMAC_RDATA 读取而 Y 缓冲区中没有有效数据时，会发生下溢。

0：未检测到下溢

1：检测到下溢。如果 UNFLIEN 位设置，则生成中断。

此标志通过复位模块来清除。

Bit 8 **OVFL：**溢出错误标志

当向 FMAC_WDATA 写入而 X1 缓冲区中没有可用空间时，会发生溢出。

0：未检测到溢出

1：检测到溢出。如果 OVFLIEN 位设置，则生成中断。

此标志通过复位模块来清除。

Bit 7:2 保留，必须保持为复位值。

Bit 1 **X1FULL**：X1 缓冲区满标志

如果可用空间数量少于 FULL_WM 阈值，则缓冲区被标记为满。可用空间数量是写指针与当前正在使用的最近样本之间的差值。

0：X1 缓冲区未满。如果 WIEN 位设置，则中断请求被断言直到标志被设置。如果 DMAWEN 设置，则在标志被设置之前生成 DMA 写通道请求。

1：X1 缓冲区满。

此标志由硬件或复位设置和清除。

*注：在 X1 缓冲区的最后一个可用空间被填满后，X1FULL 标志变高之前有 3 个时钟周期的延迟。为了避免溢出的风险，建议在写入 X1 缓冲区后插入软件延迟，然后再读取 FMAC_SR。或者，可以使用 FULL_WM 阈值为 2。*

Bit 0 **YEMPTY**：Y 缓冲区空标志

如果未读数据数量少于 EMPTY_WM 阈值，则缓冲区被标记为空。未读数据数量是读指针与当前输出目标地址之间的差值。

0：Y 缓冲区不为空。如果 RIEN 位设置，则中断请求被断言直到标志被设置。如果 DMAREN 设置，则在标志被设置之前生成 DMA 读通道请求。

1：Y 缓冲区空。

此标志由硬件或复位设置和清除。

*注：在从 Y 缓冲区读取最后一个样本后，YEMPTY 标志变高之前有 3 个时钟周期的延迟。为了避免下溢的风险，建议在从 Y 缓冲区读取后插入软件延迟，然后再读取 FMAC_SR。或者，可以使用 EMPTY_WM 阈值为 2。*

### 14.4.7 FMAC 写数据寄存器（FMAC_WDATA）

偏移地址：0x18

复位值：0x0000 0000

访问：字访问

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **WDATA[15:0]**：写数据

当对此寄存器进行写访问时，写数据被传输到写指针指示的地址偏移量处。每次写访问后，指针地址自动递增。

### 14.4.8 FMAC 读数据寄存器（FMAC_RDATA）

偏移地址：0x1C
复位值：0x0000 0000
访问：字访问

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **RDATA[15:0]：**读数据

当对此寄存器进行读访问时，读数据是 Y 输出缓冲区中由 READ 指针指示的偏移地址处的内容。每次读访问后，指针地址自动递增。

# 15 模数转换器（ADC）

## 15.1 ADC 简介

本部分介绍多达 3 个 ADC 的实现。

每个 ADC 由 12 位逐次逼近模数转换器组成。

每个 ADC 的复用通道最多达 16 条。各种不同通道的 A/D 转换可在单次、连续、扫描或间断采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

ADC 的输入时钟不得超过 64MHz，它是由 SYSCLK 或 PLLPCLK 产生。详见章节：ADC 时钟。

## 15.2 ADC 主要特征

- 高性能特性
  - 多达 3 个 12 位分辨率 ADC
    ADC1 连接至 11 个外部通道和 4 个内部通道
    ADC2 连接至 7 个外部通道和 6 个内部通道
    ADC3 连接至 10 个外部通道和 4 个内部通道
  - ADC 转换时间与 AHB 总线时钟频率无关
  - 自校准
  - 可独立设置各通道采样时间
  - 多达四条注入通道（对规则通道或注入通道的模拟输入分配完全可配置）
  - 数据对齐以保持内置数据一致性
  - 数据可由 DMA 管理，实现规则通道转换
  - 4 个专用注入数据寄存器供注入通道使用
  - 8 个专用规则数据寄存器供规则通道使用
- 每个 ADC 都有外部模拟输入通道
  - 专用 GPIO 焊盘有最多达 4 条快速通道
  - 专用 GPIO 焊盘有最多达 7 条慢速通道
- 此外，还有几条内部专用通道
  - 内部基准电压（$V_{REFINT}$），连接到 ADC3
  - 内部温度传感器（$V_{TPS}$），连接到 ADC2
  - 参考电压（$V_{REF}$）监测通道，连接到 ADC1
  - OPAMP1/2 内部输出，连接到 ADC1
  - OPAMP1/2/3/4 内部输出，连接到 ADC2
  - OPAMP3/4 内部输出，连接到 ADC3
- 可通过以下方式启动转换过程：
  - 通过软件启动常规转换和注入转换
  - 通过极性可配置的硬件触发器（内部定时器事件或 GPIO 输入事件）来启动常规转换和注入转换
- ADC1 和 ADC2 的双重 ADC 模式
- 转换模式
  - 每个 ADC 均可转换单条通道，也可扫描一系列通道

- 单次模式会在每次触发时对选定的输入执行一次转换
- 连续模式可连续转换选定的输入
- 间断采样模式

- 转换结束、注入转换结束和发生模拟看门狗事件时产生中断
- ADC 输入范围：$V_{SSA} \le V_{IN} \le V_{REF+}$

请参考设备数据手册中的 OPAMP 电气特性部分，了解在转换 OPAMP 输出电压时应应用的 ADC 采样时间值。

## 15.3 ADC 功能描述

### 15.3.1 ADC 框图

下图为 ADC 框图。

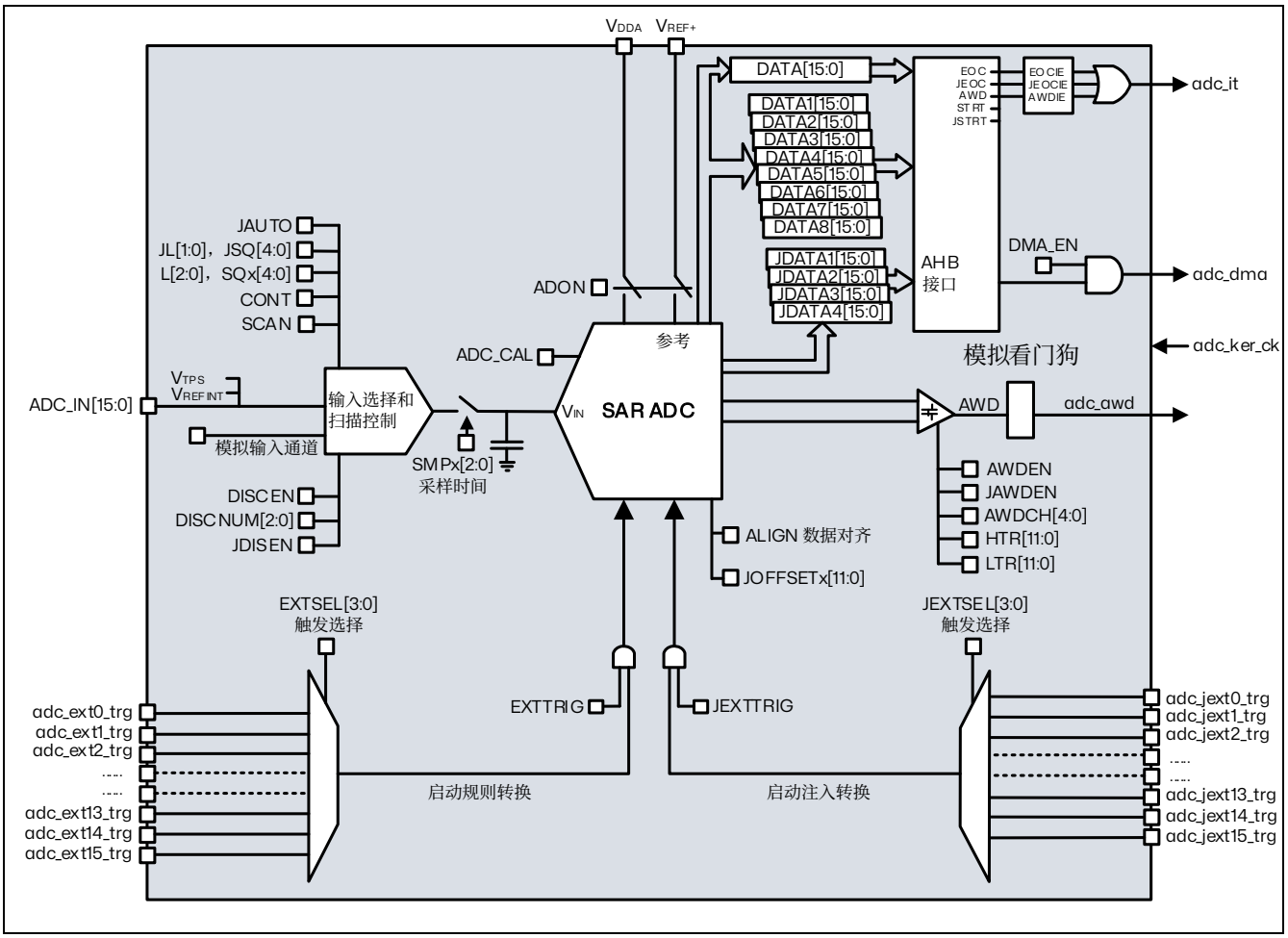



图 15.1 ADC 框图

### 15.3.2 ADC 引脚和内部信号

表 15.1 ADC 内部输入/输出信号

| 内部信号名称 | 信号类型 | 说明 |
|---|---|---|
| adc_ext_trg[15:0] | 输入 | 共有多达 16 个外部触发输入用于常规转换（可连接至片上定时器）。 |
| adc_jext_trg[15:0] | 输入 | 共有多达 16 个外部触发输入用于注入转换（可连接至片上定时器）。 |
| adc_ker_ck | 输入 | ADC 内核时钟 |
| adc_awd | 输出 | 内部模拟看门狗输出信号，连接至片上定时器。 |
| adc_it | 输出 | ADC 中断 |
| adc_dma | 输出 | ADC DMA 请求 |
| $V_{TPS}$ | 模拟输入 | 内部温度传感器的输出电压 |
| $V_{REFINT}$ | 模拟输入 | 内部基准电压的输出电压 |
| $V_{REF}$ | 模拟输入 | 参考电压 |

表 15.2 ADC 输入/输出引脚

| 引脚名称 | 信号类型 | 说明 |
|---|---|---|
| VREF+ | 输入，模拟参考正极 | ADC 使用的高端/正极参考电压 |
| VDDA | 输入，模拟电源 | 等效于 $V_{DD}$ 的模拟电源 |
| VSSA | 输入，模拟电源地 | 等效于 $V_{SS}$ 的模拟电源地 |
| ADCx_IN[15:0] | 模拟输入信号 | 16 个模拟输入通道 |

1. VDDA 和 VSSA 应该分别连接到 VDD 和 VSS。

### 15.3.3 ADC 时钟

ADC 时钟可以是名为 adc_ker_ck 的特定时钟源。

三个 ADC 的输入时钟相同，可从两个不同的时钟源中选择（请参见下图：ADC 时钟方案）：

– SYSCLK 系统时钟

– PLL“P”时钟

该时钟可在 RCC 中配置（有关如何生成 ADC 时钟（adc_ker_ck）专用时钟的更多信息，请参见 RCC 部分）。

*注：ADC 的输入时钟不得超过* 64MHz。

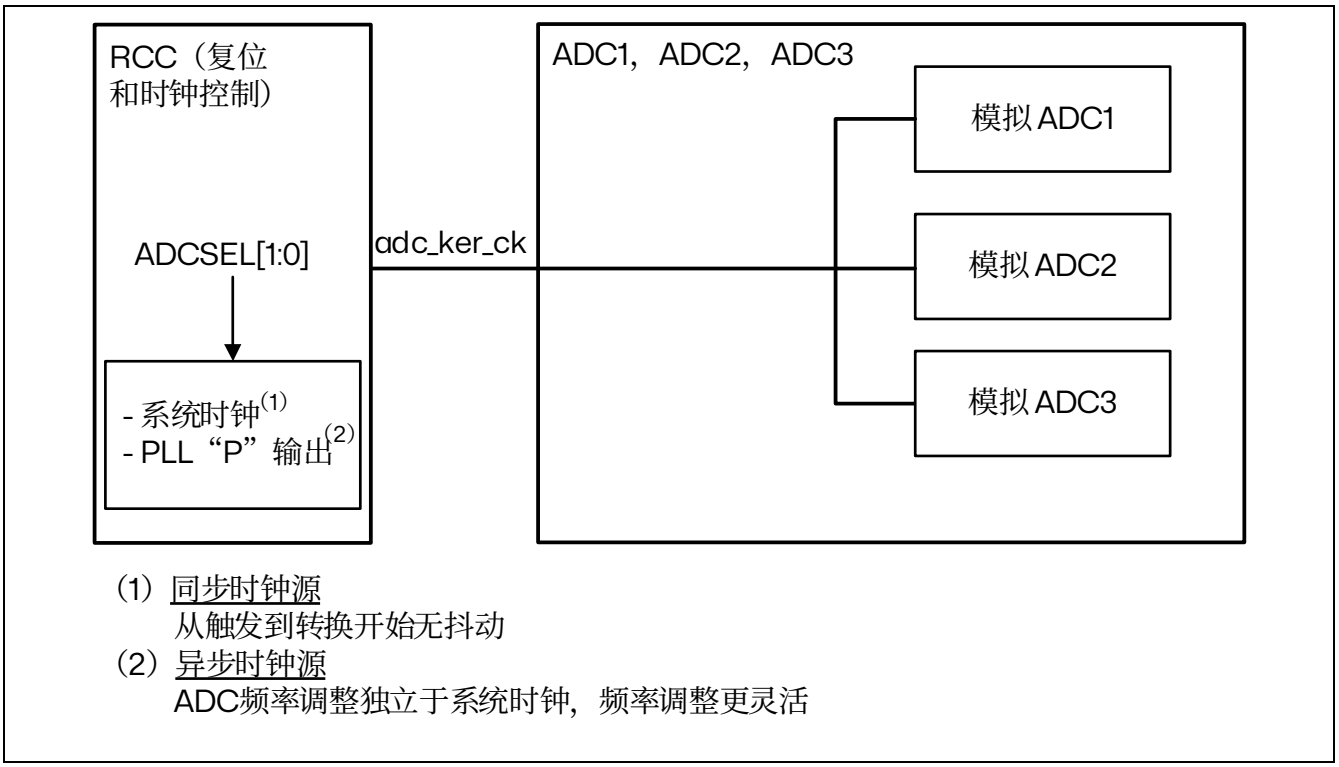



图 15.2 ADC 时钟方案

### 15.3.4 ADC 通道连接

ADC1、ADC2、ADC3 的通道如下列表所示。
通道编号，即通道 x 对应设备数据手册中定义的 ADCy_INx 引脚。

表 15.3 ADC1 通道

| 通道编号 | 模拟信号来源 | 通道速度 |
|---|---|---|
| 通道 0 | VSSA | FAST_4M |
| 通道 1 | PC14 | SLOW_2M |
| 通道 2 | PC15 | SLOW_2M |
| 通道 3 | PC0 | SLOW_2M |
| 通道 4 | PA0 | SLOW_2M |
| 通道 5 | PA1 | SLOW_2M |
| 通道 6 | PA3 | SLOW_2M |
| 通道 7 | PC1 | FAST_4M |
| 通道 8 | PC2 | FAST_4M |
| 通道 9 | PC3 | FAST_4M |
| 通道 10 | PA4 | FAST_4M |
| 通道 11 | OPAMP1_OUT | FAST_4M |
| 通道 12 | OPAMP2_OUT | FAST_4M |
| 通道 13 | PC13 | SLOW_2M |
| 通道 14 | 保留 | SLOW_2M |
| 通道 15 | VREF | SLOW_2M |

表 15.4 ADC2 通道

| 通道编号 | 模拟信号来源 | 通道速度 |
|---|---|---|
| 通道 0 | VSSA | FAST_4M |
| 通道 1 | PC4 | SLOW_2M |
| 通道 2 | PC5 | SLOW_2M |
| 通道 3 | PE9 | SLOW_2M |
| 通道 4 | PE10 | SLOW_2M |
| 通道 5 | PE13 | SLOW_2M |
| 通道 6 | 保留 | SLOW_2M |
| 通道 7 | PA7 | FAST_4M |
| 通道 8 | PB0 | FAST_4M |
| 通道 9 | OPAMP1_OUT | FAST_4M |
| 通道 10 | OPAMP2_OUT | FAST_4M |
| 通道 11 | OPAMP3_OUT | FAST_4M |
| 通道 12 | OPAMP4_OUT | FAST_4M |
| 通道 13 | 保留 | SLOW_2M |
| 通道 14 | 保留 | SLOW_2M |
| 通道 15 | TPS | SLOW_2M |

表 15.5 ADC3 通道

| 通道编号 | 模拟信号来源 | 通道速度 |
|---|---|---|
| 通道 0 | VSSA | FAST_4M |
| 通道 1 | PE14 | SLOW_2M |
| 通道 2 | PE15 | SLOW_2M |
| 通道 3 | PB10 | SLOW_2M |
| 通道 4 | PB11 | SLOW_2M |
| 通道 5 | PB13 | SLOW_2M |
| 通道 6 | PB14 | SLOW_2M |
| 通道 7 | PD8 | FAST_4M |
| 通道 8 | PD9 | FAST_4M |
| 通道 9 | PD10 | FAST_4M |
| 通道 10 | PB12 | FAST_4M |
| 通道 11 | OPAMP3_OUT | FAST_4M |
| 通道 12 | OPAMP4_OUT | FAST_4M |
| 通道 13 | VREFINT | SLOW_2M |

*注：1. 如果需要采样* VREFINT *通道，需要置位* ADC_CR2 *寄存器的* TSVREFE *位*

2. *如果需要采样* TPS *通道，采样时间要设定为* 640.5T *（选择最高档位）*

### 15.3.5 校准

ADC 有一个内置自校准模式。校准可大幅减小因内部电容器组的变化而造成的准精度误差。在校准期间，在每个电容器上都会计算出一个误差修正码（数字值），这个码用于消除在随后的转换中每个电容器上产生的误差。

通过设置 ADC_CR2 寄存器的 CAL 位启动校准。一旦校准结束，CAL 位被硬件复位，可以开始正常转换。建议在上电时执行一次 ADC 校准。校准阶段结束后，校准码储存在 ADC_CAL 中。

*注：建议在每次上电后执行一次校准。启动校准前，*ADC *必须处于上电状态（*ADON= *“*1*”*）超过至少两个*ADC *时钟周期。*

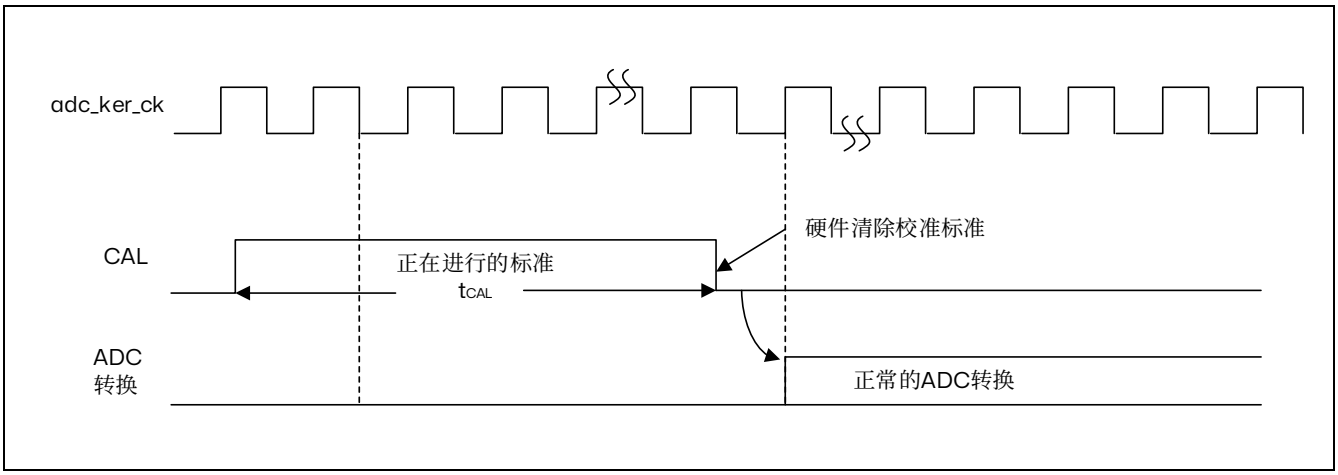



图 15.3 校准时序图

### 15.3.6 ADC 开关控制

可通过将 ADC_CR2 寄存器中的 ADON 位置 1 来为 ADC 供电。首次将 ADON 位置 1 时，会将 ADC 从掉电模式中唤醒。

ADC 供电延迟一段时间后（$t_{STAB}$），再次设置 ADON 位时开始进行转换。

可通过将 ADON 位清零来停止转换并使 ADC 进入掉电模式。在此模式下，ADC 几乎不耗电（只有几μA）。

### 15.3.7 通道选择

RX32G410 每个 ADC 有多路通道。可以将转换分为两组：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。

- **规则组：**由多达 8 个转换组成。规则通道和它们的转换顺序在 ADC_SQRx 寄存器中选择。规则组中转换的总数应写入 ADC_SQR1 寄存器的 L[2:0]位中。
- **注入组：**由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_JSQR 寄存器中选择。注入组里的转换总数目应写入 ADC_JSQR 寄存器的 JL[1:0]位中。

如果 ADC_SQRx 或 ADC_JSQR 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

### 15.3.8 单次转换模式

单次转换模式下，ADC 只执行一次转换。该模式既可通过设置 ADC_CR2 寄存器的 ADON 位（只适用于规则通道）启动也可通过外部触发启动（适用于规则通道或注入通道），这时 CONT 位为 0。一旦选择通道的转换完成：

一旦选择通道的转换完成：

- 如果一个规则通道被转换：
  - 转换数据被储存在对应的 16 位 ADC_DR 寄存器中
  - EOC（转换结束）标志被置位
  - 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_JDR1 寄存器中
  - EOC & JEOC（转换结束和注入转换结束）标志被设置

  *注：* ADC_CR4 *寄存器的* EOC_OPT *位可选择控制* EOC *（转换结束）标志*
  OPT = 0b*：规则或注入通道组转换结束时设置* ADC_SR *寄存器的* EOC
  OPT = 1b*：规则通道组转换结束时设置* ADC_SR *寄存器的* EOC

  - 如果设置了 JEOCIE 位则产生中断

然后 ADC 停止。

### 15.3.9 连续转换模式

在连续转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。此模式可通过外部触发启动或通过设置 ADC_CR2 寄存器上的 ADON 位启动，此时 CONT 位是 1。

每个转换后：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位 ADC_DR 寄存器中

- EOC（转换结束）标志被置位
- 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_JDR1 寄存器中
- EOC & JEOC（转换结束和注入转换结束）标志被设置

*注：*ADC_CR4 *寄存器的*EOC_OPT *可选择控制*EOC *（转换结束）标志*
OPT = 0b*：规则或注入通道组转换结束时设置*ADC_SR *寄存器的*EOC
OPT = 1b*：规则通道组转换结束时设置*ADC_SR *寄存器的*EOC

- 如果设置了 JEOCIE 位则产生中断

### 15.3.10 时序图

如下图所示，ADC 在开始精确转换前需要一个稳定时间 $t_{STAB}$。在开始 ADC 转换和 16 个时钟周期后，EOC 标志被设置，16 位 ADC 数据寄存器包含转换的结果。

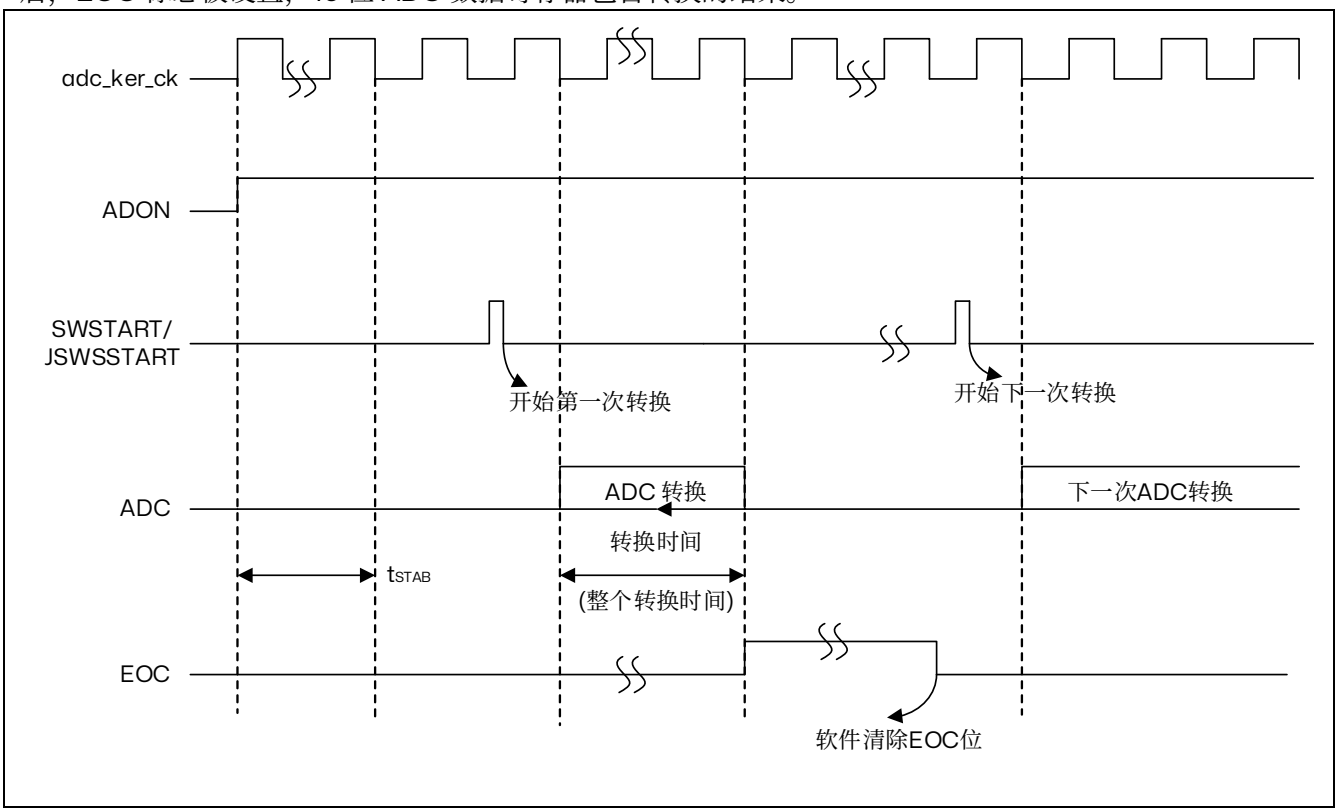



图 15.4 ADC 转换时序图

### 15.3.11 模拟看门狗

如果被 ADC 转换的模拟电压低于低阀值或高于高阀值，AWD 模拟看门狗状态位被设置。阀值位于 ADC_HTR 和 ADC_LTR 寄存器的最低 12 个有效位中。通过设置 ADC_CR1 寄存器的 AWDIE 位以允许产生相应中断。

阀值独立于由 ADC_CR2 寄存器上的 ALIGN 位选择的数据对齐模式。比较是在对齐之前完成的。

通过配置 ADC_CR1 寄存器，模拟看门狗可以作用于 1 个或多个通道。

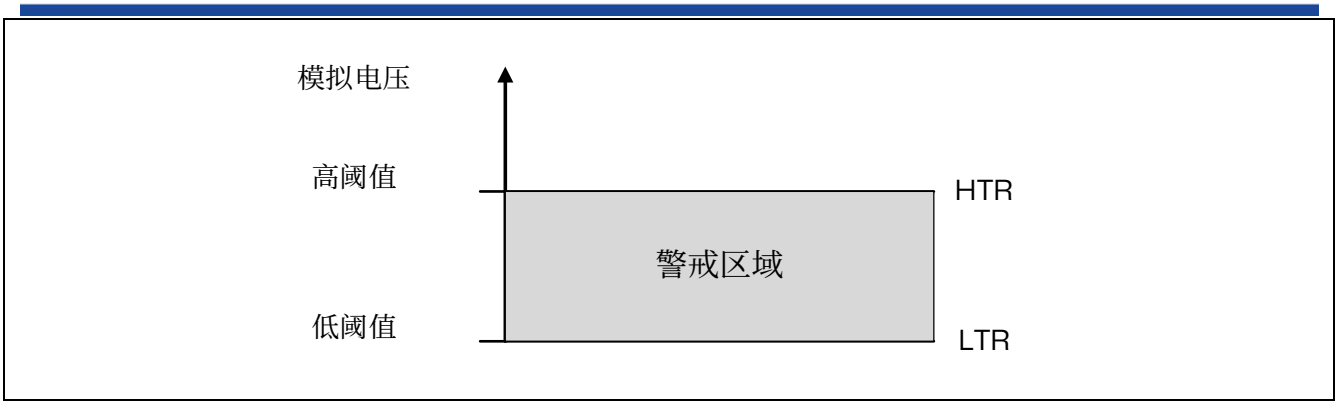



图 15.5 模拟看门狗警戒区

表 15.6 模拟看门狗通道选择

| 模拟看门狗警戒通道 | ADC_CR1 寄存器控制位 | | |
|---|---|---|---|
| | AWDSGL 位 | AWDEN 位 | JAWDEN 位 |
| 无 | 任意值 | 0 | 0 |
| 所有注入通道 | 0 | 0 | 1 |
| 所有规则通道 | 0 | 1 | 0 |
| 所有注入和规则通道 | 0 | 1 | 1 |
| 单一的 (1) 注入通道 | 1 | 0 | 1 |
| 单一的 (1) 规则通道 | 1 | 1 | 0 |
| 单一的 (1) 注入或规则通道 | 1 | 1 | 1 |

1. 由 AWDCH[4:0]位选择

### 15.3.12 扫描模式

此模式用来扫描一组模拟通道。

扫描模式可通过设置 ADC_CR1 寄存器的 SCAN 位来选择。一旦这个位被设置，ADC 扫描所有被 ADC_SQRx 寄存器（对规则通道）或 ADC_JSQR（对注入通道）选中的所有通道。在每个组的每个通道上执行单次转换。在每个转换结束时，同一组的下一个通道被自动转换。如果设置了 CONT 位，转换不会在选择组的最后一个通道上停止，而是再次从选择组的第一个通道继续转换。

规则通道转换的对应数据被存储在 16 位 ADC_DRx（x = 1~8）规则数据寄存器 x 中

注入通道转换的对应数据被存储在 16 位 ADC_JDRx（x = 1~4）注入数据寄存器 x 中

如果设置了 DMA 位，在每个转换后，DMA 控制器把规则组通道的转换数据传输到 SRAM 中。

### 15.3.13 注入通道管理

#### 触发注入

清除 ADC_CR1 寄存器的 JAUTO 位，并且设置 SCAN 位，即可使用触发注入功能。

1. 利用外部触发或通过设置 ADC_CR2 寄存器的 ADON 位，启动一组规则通道的转换。
2. 如果在规则通道转换期间产生一外部注入触发，当前转换被复位，注入通道序列被以单次扫描方式进行转换。
3. 然后，恢复上次被中断的规则组通道转换。如果在注入转换期间产生一规则事件（前一次的规则序列已经完成），注入转换不会被中断，但是规则序列将在注入序列结束后被执行。

*注：当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为32 个ADC 时钟周期（即2 个具有3.5 个时钟间隔采样时间的转换），触发之间最小的间隔必须是33 个ADC 时钟周期。*

**自动注入**

如果设置了 JAUTO 位，在规则组通道之后，注入组通道被自动转换。这可以用来转换在 ADC_SQRx 和 ADC_JSQR 寄存器中设置的多至 12 个转换序列。

在此模式里，必须禁止注入通道的外部触发。

如果除 JAUTO 位外还设置了 CONT 位，规则通道至注入通道的转换序列被连续执行。

当从规则转换切换到注入序列或从注入转换切换到规则序列时，会自动插入 2 个 ADC 时钟间隔的延迟。

*注：不可能同时使用自动注入和间断模式。*

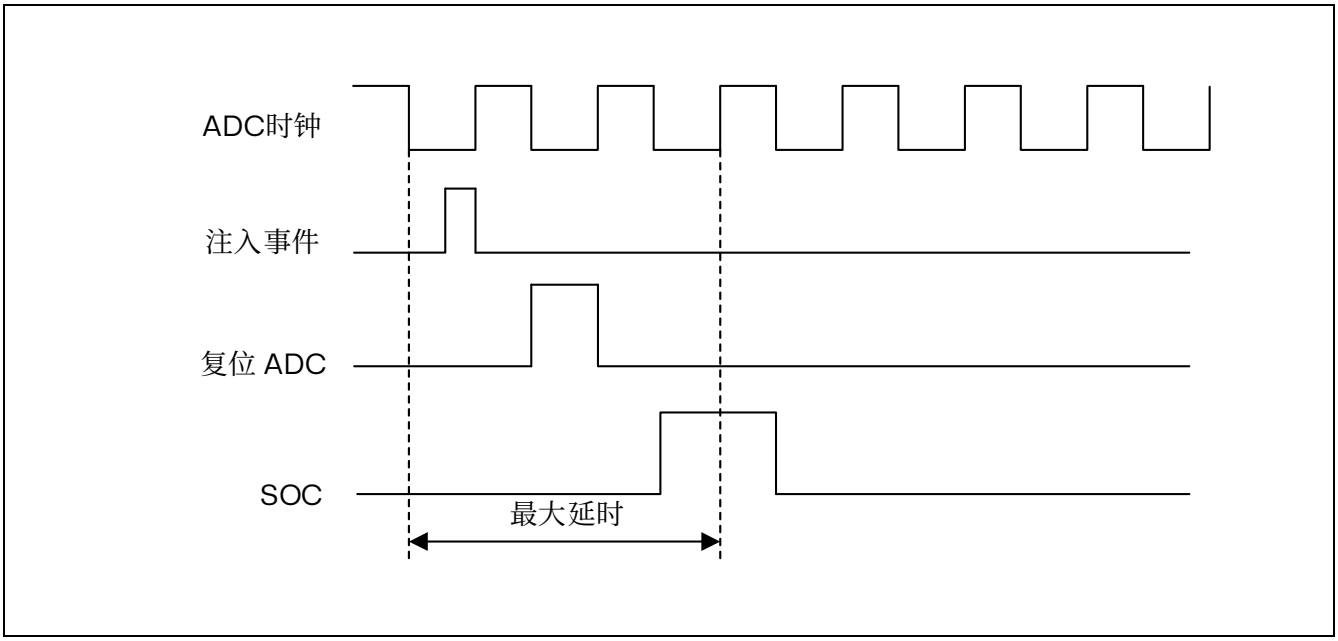



图 15.6 注入转换延迟

### 15.3.14 间断模式

**规则组**

此模式通过设置 ADC_CR1 寄存器上的 DISCEN 位激活。它可以用来执行一个短序列的 n 次转换（n<=8），此转换是 ADC_SQRx 寄存器所选择的转换序列的一部分。数值 n 由 ADC_CR1 寄存器的 DISCNUM[2:0]位给出。

一个外部触发信号可以启动 ADC_SQRx 寄存器中描述的下一轮 n 次转换，直到此序列所有的转换完成为止。总的序列长度由 ADC_SQR1 寄存器的 L[2:0]定义。

举例：

n = 3，被转换通道 = 0、1、2、3、6、7、9、10

第一次触发：转换的序列为 0、1、2

第二次触发：转换的序列为 3、6、7

第三次触发：转换的序列为 9、10，并产生 EOC 事件

第四次触发：转换的序列为 0、1、2

*注：当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。*
*当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0、1 和 2。*

**注入组**

此模式通过设置 ADC_CR1 寄存器的 JDISCEN 位激活。在一个外部触发事件后，该模式按通道顺序逐个转换 ADC_JSQR 寄存器中选择的序列。

一个外部触发信号可以启动 ADC_JSQR 寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由 ADC_JSQR 寄存器的 JL[1:0]位定义。

例子：

n=1，被转换的通道 = 1、2、3

第一次触发：通道 1 被转换

第二次触发：通道 2 被转换

第三次触发：通道 3 被转换，并且产生 EOC 和 JEOC 事件

第四次触发：通道 1 被转换

*注：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。*
*不能同时使用自动注入和间断模式。*
*不得同时为规则组和注入组设置间断模式。只能针对一个组使能间断模式。*

### 15.3.15 数据对齐

ADC_CR2 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐。

注入组通道转换的数据值已经减去了在 ADC_JOFRx 寄存器中定义的偏移量，因此结果可以是一个负值。SEXT 位是扩展的符号值。

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

注入组

| SEXT | SEXT | SEXT | SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

规则组

| 0 | 0 | 0 | 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

图 15.7 数据右对齐

注入组

| SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

规则组

| D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

图 15.8 数据左对齐

### 15.3.16 可编程的通道采样时间

ADC 使用若干个 adc_ker_ck 周期对输入电压采样，采样周期数目可以通过 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMP[2:0]位更改。每个通道可以分别用不同的时间采样。

总转换时间如下计算：

$T_{CONV}$ = 采样时间 + 12.5 个周期

例如：

当 adc_ker_ck = 64MHz，采样时间为 3.5 周期

$T_{CONV}$ = 3.5 + 12.5 = 16 个周期 = 250ns

### 15.3.17 外部触发转换

转换可以由外部事件触发（例如定时器捕获，EXTI 线）。如果设置了 EXTTRIG 控制位，则外部事件就能够触发转换。EXTSEL[3:0]和 JEXTSEL[3:0]控制位允许应用程序选择 16 个可能的事件中的某一个，可以触发规则和注入组的采样

*注：当外部触发信号被选为 ADC 规则或注入转换时，只有它的上升沿可以启动转换。*

表 15.7 ADC1 规则通道的外部触发信号

| 名称 | 触发源 | 连接类型 | EXTSEL[3:0] |
|---|---|---|---|
| adc_ext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_ext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_ext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_ext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_ext_trg4 | TIM2_TRGO 事件 | | 0100 |
| adc_ext_trg5 | TIM3_TRGO 事件 | | 0101 |
| adc_ext_trg6 | TIM4_TRGO 事件 | | 0110 |
| adc_ext_trg7 | SWSTART | 软件控制位 | 0111 |
| adc_ext_trg8 | EXTI 线 2 | 外部引脚 | 1000 |
| adc_ext_trg9 | TIM6_TRGO 事件 | 片上定时器的内部信号 | 1001 |
| adc_ext_trg10 | hrtim_adc_trg1 事件 | | 1010 |
| adc_ext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_ext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_ext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_ext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_ext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

表 15.8 ADC2 规则通道的外部触发信号

| 名称 | 触发源 | 连接类型 | EXTSEL[3:0] |
|---|---|---|---|
| adc_ext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_ext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_ext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_ext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_ext_trg4 | TIM2_TRGO 事件 | | 0100 |
| adc_ext_trg5 | TIM3_TRGO 事件 | | 0101 |
| adc_ext_trg6 | TIM4_TRGO 事件 | | 0110 |
| adc_ext_trg7 | SWSTART | 软件控制位 | 0111 |
| adc_ext_trg8 | EXTI 线 3 | 外部引脚 | 1000 |
| adc_ext_trg9 | TIM6_TRGO 事件 | 片上定时器的内部信号 | 1001 |
| adc_ext_trg10 | hrtim_adc_trg3 事件 | | 1010 |
| adc_ext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_ext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_ext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_ext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_ext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

表 15.9 ADC3 规则通道的外部触发信号

| 名称 | 触发源 | 连接类型 | EXTSEL[3:0] |
|---|---|---|---|
| adc_ext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_ext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_ext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_ext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_ext_trg4 | TIM2_TRGO 事件 | | 0100 |
| adc_ext_trg5 | TIM3_TRGO 事件 | | 0101 |
| adc_ext_trg6 | TIM4_TRGO 事件 | | 0110 |
| adc_ext_trg7 | SWSTART | 软件控制位 | 0111 |
| adc_ext_trg8 | EXTI 线 4 | 外部引脚 | 1000 |
| adc_ext_trg9 | TIM6_TRGO 事件 | 片上定时器的内部信号 | 1001 |
| adc_ext_trg10 | hrtim_adc_trg5 事件 | | 1010 |
| adc_ext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_ext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_ext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_ext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_ext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

表 15.10 ADC1 注入通道的外部触发信号

| 名称 | 触发源 | 连接类型 | JEXTSEL[3:0] |
|---|---|---|---|
| adc_jext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_jext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_jext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_jext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_jext_trg4 | TIM1_OC4/OC5 事件 | | 0100 |
| adc_jext_trg5 | TIM2_TRGO 事件 | | 0101 |
| adc_jext_trg6 | EXTI 线 13 | 外部引脚 | 0110 |
| adc_jext_trg7 | JSWSTART | 软件控制位 | 0111 |
| adc_jext_trg8 | TIM15_TRGO 事件 | 片上定时器的内部信号 | 1000 |
| adc_jext_trg9 | TIM7_TRGO 事件 | | 1001 |
| adc_jext_trg10 | hrtim_adc_trg2 事件 | | 1010 |
| adc_jext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_jext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_jext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_jext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_jext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

表 15.11 ADC2 注入通道的外部触发信号

| 名称 | 触发源 | 连接类型 | JEXTSEL[3:0] |
|---|---|---|---|
| adc_jext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_jext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_jext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_jext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_jext_trg4 | TIM8_OC4/OC5 事件 | | 0100 |
| adc_jext_trg5 | TIM3_TRGO 事件 | | 0101 |
| adc_jext_trg6 | EXTI 线 14 | 外部引脚 | 0110 |
| adc_jext_trg7 | JSWSTART | 软件控制位 | 0111 |
| adc_jext_trg8 | TIM15_TRGO 事件 | 片上定时器的内部信号 | 1000 |
| adc_jext_trg9 | TIM7_TRGO 事件 | | 1001 |
| adc_jext_trg10 | hrtim_adc_trg4 事件 | | 1010 |
| adc_jext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_jext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_jext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_jext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_jext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

表 15.12 ADC3 注入通道的外部触发信号

| 名称 | 触发源 | 连接类型 | JEXTSEL[3:0] |
|---|---|---|---|
| adc_jext_trg0 | TIM1_TRGO 事件 | 片上定时器的内部信号 | 0000 |
| adc_jext_trg1 | TIM1_TRGO2 事件 | | 0001 |
| adc_jext_trg2 | TIM8_TRGO 事件 | | 0010 |
| adc_jext_trg3 | TIM8_TRGO2 事件 | | 0011 |
| adc_jext_trg4 | TIM8_OC4/OC5 事件 | | 0100 |
| adc_jext_trg5 | TIM4_TRGO 事件 | | 0101 |
| adc_jext_trg6 | EXTI 线 15 | 外部引脚 | 0110 |
| adc_jext_trg7 | JSWSTART | 软件控制位 | 0111 |
| adc_jext_trg8 | TIM15_TRGO 事件 | 片上定时器的内部信号 | 1000 |
| adc_jext_trg9 | TIM7_TRGO 事件 | | 1001 |
| adc_jext_trg10 | hrtim_adc_trg5 事件 | | 1010 |
| adc_jext_trg11 | hrtim_adc_trg6 事件 | | 1011 |
| adc_jext_trg12 | hrtim_adc_trg7 事件 | | 1100 |
| adc_jext_trg13 | hrtim_adc_trg8 事件 | | 1101 |
| adc_jext_trg14 | hrtim_adc_trg9 事件 | | 1110 |
| adc_jext_trg15 | hrtim_adc_trg10 事件 | | 1111 |

软件触发事件可以通过对寄存器 ADC_CR2 的 SWSTART 或 JSWSTART 位置 1 产生。
规则组的转换可以被注入触发打断。

### 15.3.18 DMA 请求

因为规则通道转换的值储存在一个通用的数据寄存器 ADC_DR 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数据寄存器 ADC_DR 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。

此外，ADC 规则通道转换值会相应的存储在与转换序列匹配的 ADC_DRx（x=1...8）寄存器中。当某一个规则通道转换结束后产生 DMA 请求，可以使用 DMA 将转换数据从该 ADC_DRx（x=1...8）寄存器传输到用户指定的目的地址。

## 15.4 双 ADC 模式

在双 ADC 模式里，根据 ADC1_CR1 寄存器中 CR1_DUALMOD[3:0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同步触发。

*注：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 种模式：

- 同步注入模式
- 同步规则模式
- 快速交叉模式
- 慢速交叉模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同步注入模式 + 同步规则模式
- 同步规则模式 + 交替触发模式
- 同步注入模式 + 交叉模式

*注：在双 ADC 模式里，为了在主数据寄存器上读取从转换数据，必须使能 DMA 位，即使不使用 DMA 传输规则通道数据。*

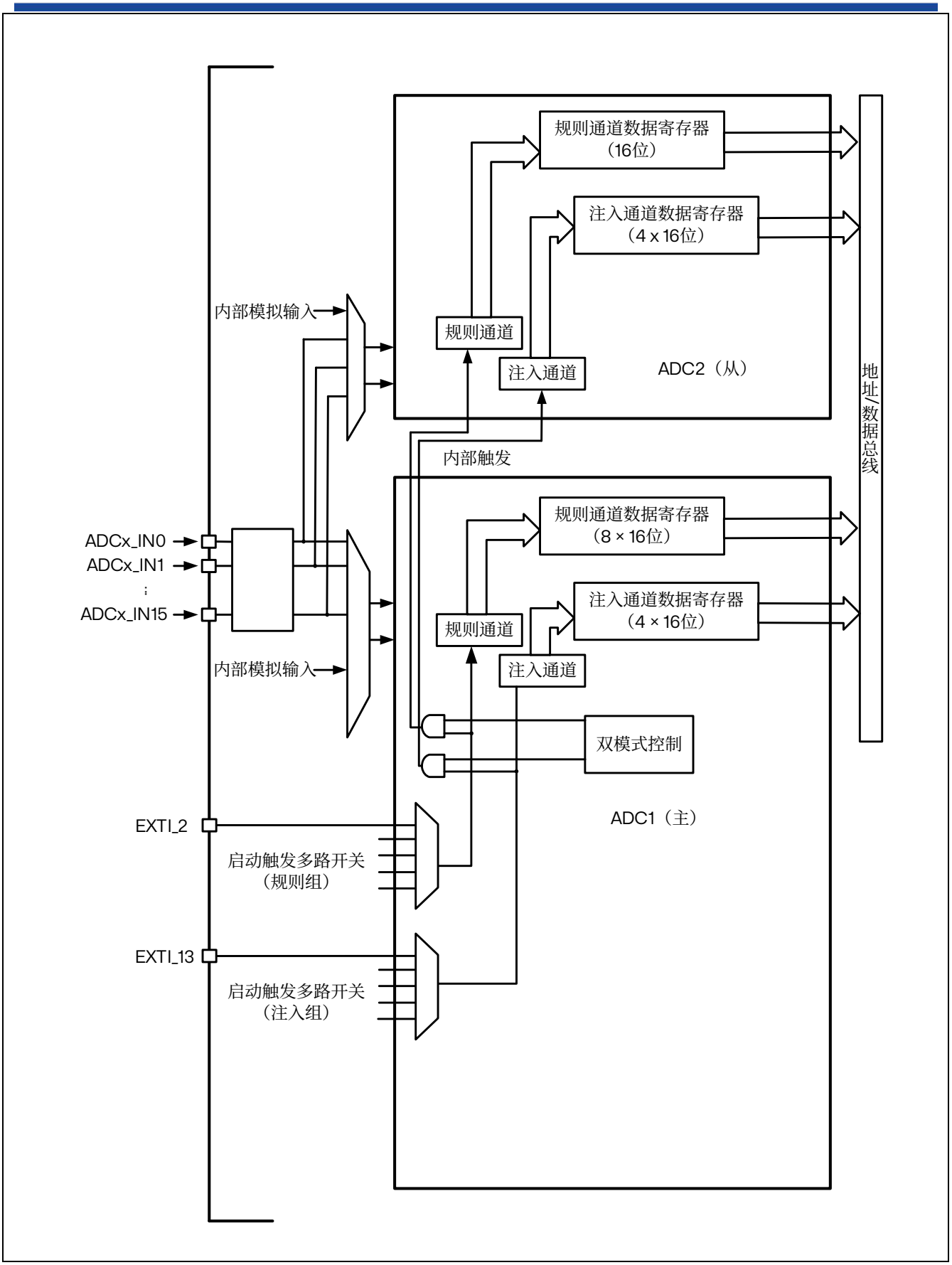

规则通道数据寄存器
（16位）
注入通道数据寄存器
（4 x 16位）
内部模拟输入
规则通道
注入通道
ADC2（从）
地址/数据总线
内部触发
规则通道数据寄存器
（8 × 16位）
ADCx_IN0
ADCx_IN1
ADCx_IN15
注入通道数据寄存器
（4 × 16位）
规则通道
内部模拟输入
注入通道
双模式控制
ADC1（主）
EXTI_2
启动触发多路开关
（规则组）
EXTI_13
启动触发多路开关
（注入组）

图 15.9 双 ADC 框图

### 15.4.1 同步注入模式

此模式转换一个注入通道组。外部触发来自 ADC1 的注入组多路开关（由 ADC1_CR2 寄存器的 JEXTSEL[3:0]选择），它同时给 ADC2 提供同步触发。

*注：不要在 2 个 ADC 上转换相同的通道（两个 ADC 在同一个通道上的采样时间不能重叠）。*

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个 ADC 接口的 ADC_JDRx 寄存器中。
- 当所有 ADC1/ADC2 注入通道都被转换时，产生 JEOC 中断（若任一 ADC 接口开放了中断）。

*注：在同步模式中，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。*

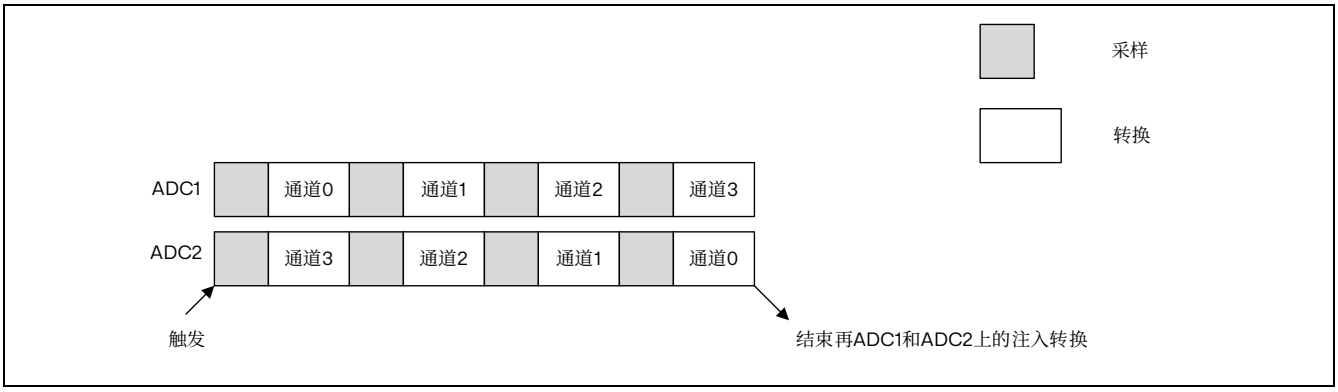



图 15.10 4 个通道的同步注入模式

### 15.4.2 同步规则模式

此模式在规则通道组上执行。外部触发来自 ADC1 的规则组多路开关（由 ADC1_CR2 寄存器的 EXTSEL[3:0]选择），它同时给 ADC2 提供同步触发。

*注：不要在 2 个 ADC 上转换相同的通道（两个 ADC 在同一个通道上的采样时间不能重叠）。*

在 ADC1 或 ADC2 的转换结束时：

- 产生一个 32 位 DMA 传输请求（如果设置了 DMA 位），32 位的 ADC1_DR 寄存器内容传输到 SRAM 中，它上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。
- 当所有 ADC1/ADC2 规则通道都被转换完时，产生 EOC 中断（若任一 ADC 接口开放了中断）。

*注：在同步规则模式中，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。*

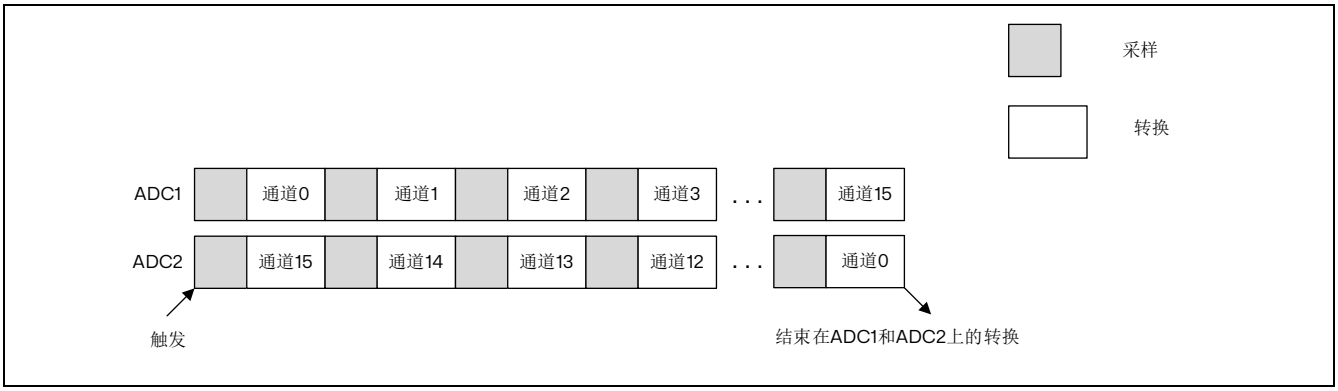



图 15.11 16 个通道的同步规则模式

### 15.4.3 快速交叉模式

此模式只适用于规则通道组。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动
- ADC1 在延迟 8 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 产生一个 EOC 中断后（由 EOCIE 使能），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 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

*注：最大允许采样时间<8 个adc_ker_ck 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。*

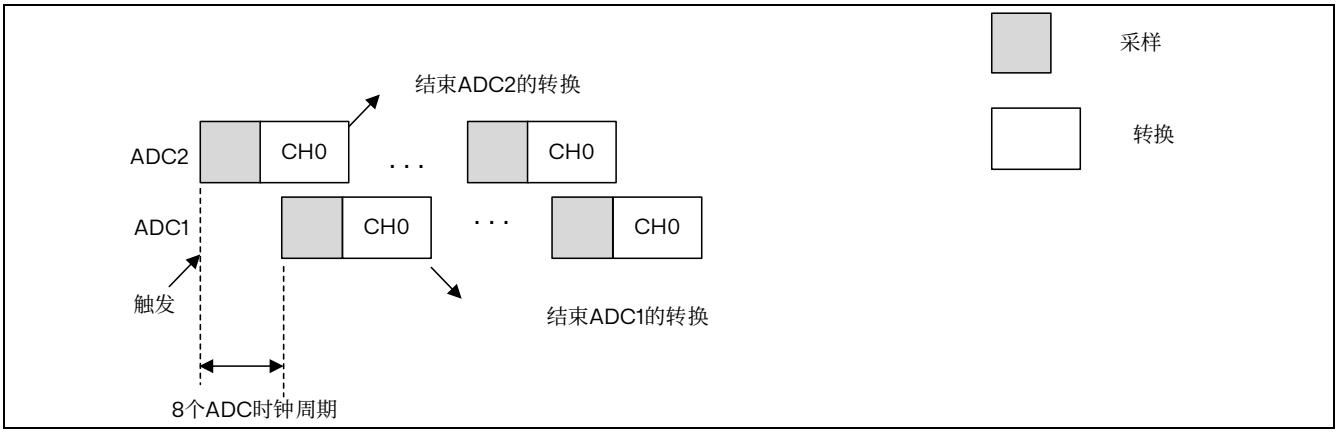



图 15.12 在 1 个通道上连续转换模式下的快速交叉模式

### 15.4.4 慢速交叉模式

此模式只适用于规则通道组。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动
- ADC1 在延迟 16 个 ADC 时钟周期后启动
- 在延迟第二次 16 个 ADC 周期后 ADC2 再次启动，如此循环。

*注：最大允许采样时间<16 个ADCCLK 周期，以避免和下个转换重叠。*

ADC1 产生一个 EOC 中断后（由 EOCIE 使能），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 的上半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

在 32 个 ADC 时钟周期后自动启动新的 ADC2 转换。

在这个模式下不能设置 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。*

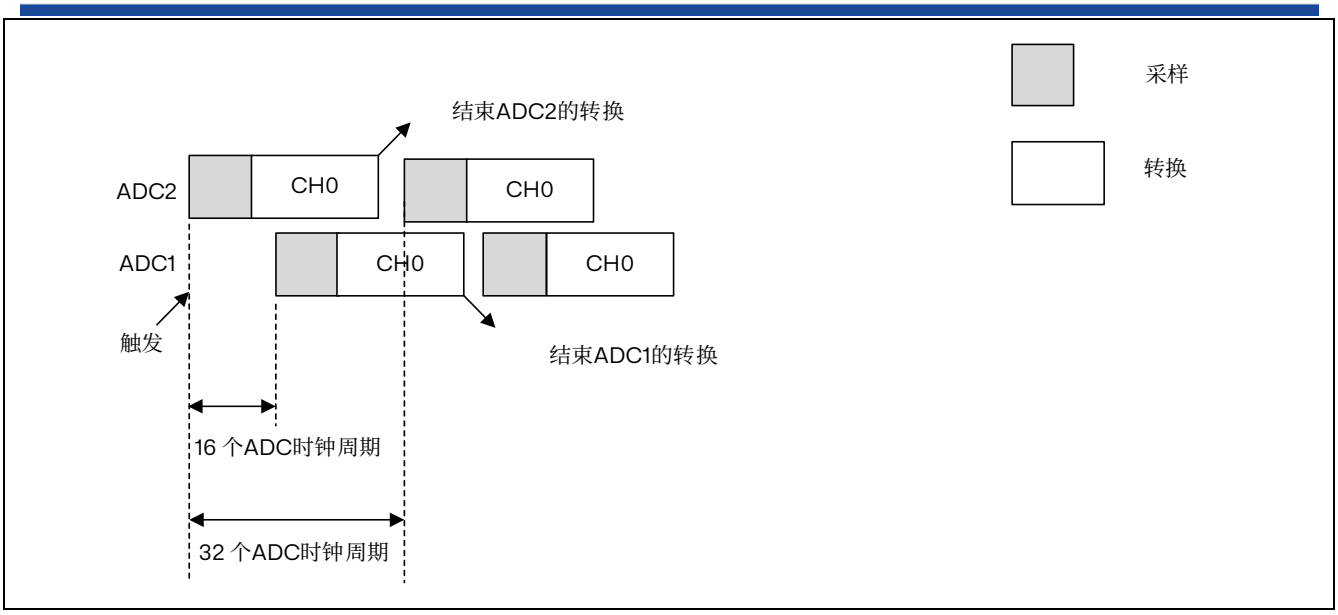



图 15.13 在 1 个通道上的慢速交叉模式

### 15.4.5 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路开关。

- 当第一个触发产生时，ADC1 上的所有注入组通道被转换。
- 当第二个触发到达时，ADC2 上的所有注入组通道被转换。
- 如此循环.....

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后，如果又有另一个外部触发，交替触发处理从转换 ADC1 注入组通道重新开始。

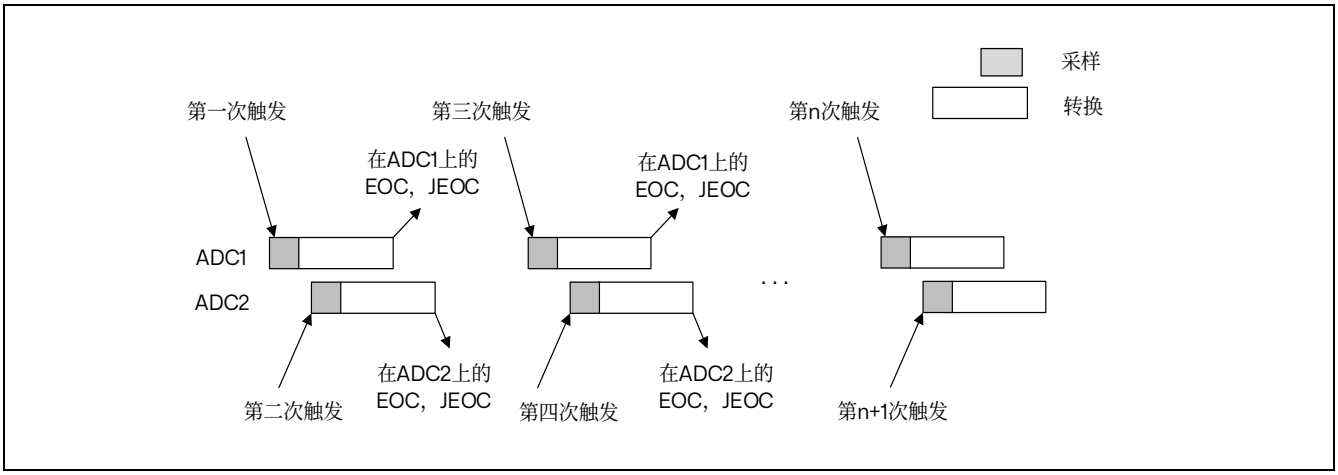



图 15.14 交替触发：每个 ADC1 的注入通道组

如果 ADC1 和 ADC2 上同时使用了注入间断模式：

- 当第一个触发产生时，ADC1 上的第一个注入通道被转换。
- 当第二个触发到达时，ADC2 上的第一个注入通道被转换。
- 如此循环.....

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。
当所有注入组通道都转换完后，如果又有另一个外部触发，则重新开始交替触发过程。

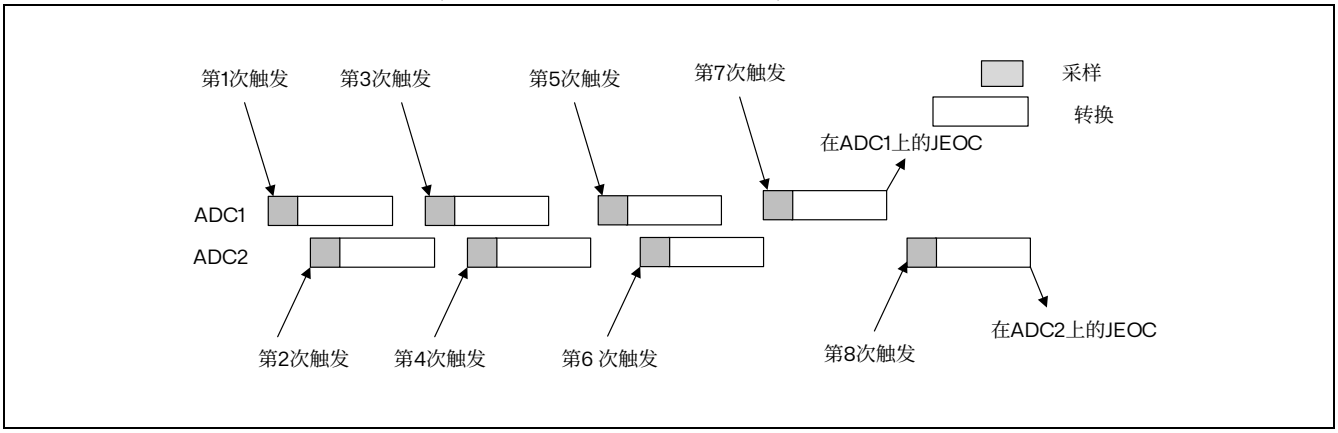



图 15.15 交替触发：在间断模式下每个 ADC 上的 4 个注入通道

### 15.4.6 独立模式

此模式里，双 ADC 同步不工作，每个 ADC 接口独立工作。

### 15.4.7 混合的规则/注入同步模式

规则组同步转换可以被中断，以启动注入组的同步转换。

*注：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。*

### 15.4.8 混合的同步规则+交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。下图显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换已经在运行，为了在注入转换后确保同步，所有的 ADC（主和从）的规则转换被停止，并在注入转换结束时同步恢复。

*注：在混合的同步规则+交替触发模式中，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。*

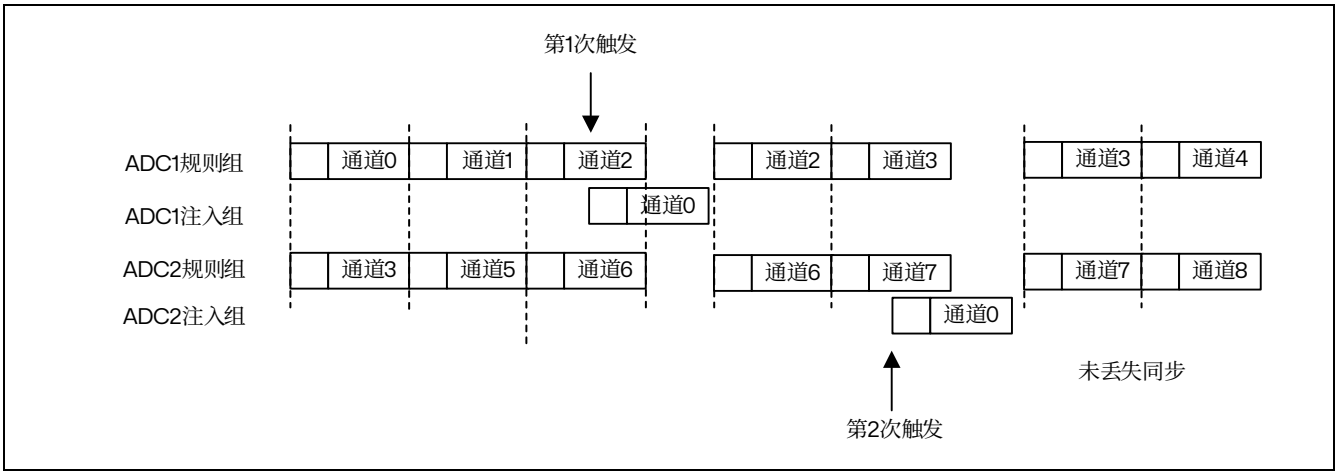



图 15.16 交替+规则同步

如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。下图示出了这种情况的操作（第 2 个触发被忽略）。

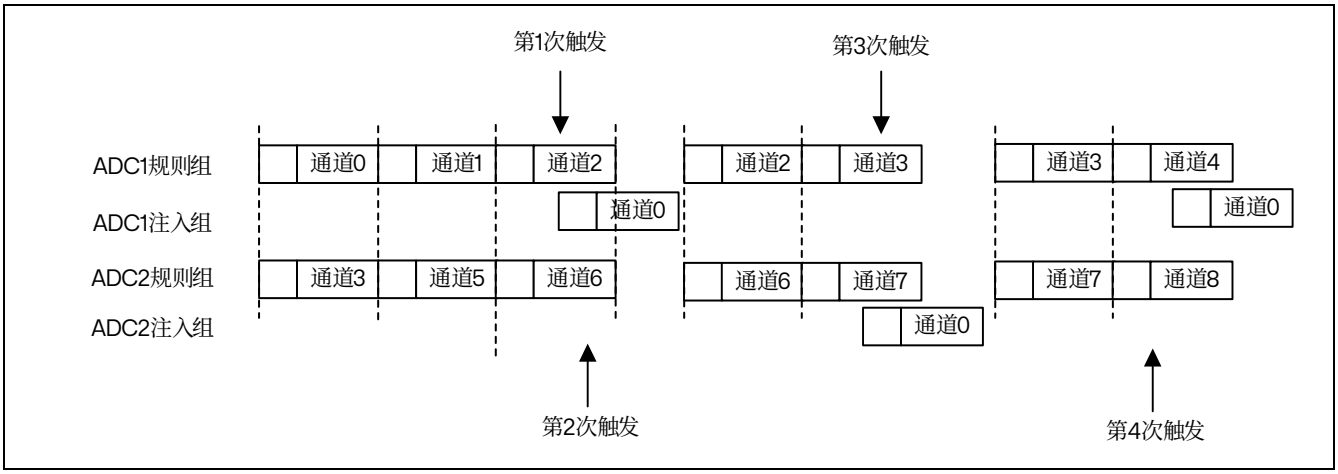



图 15.17 触发事件发生在注入转换期间

### 15.4.9 混合同步注入+交叉模式

一个注入事件可以中断一个交叉转换。这种情况下，交叉转换被中断，注入转换被启动，在注入序列转换结束时，交叉转换被恢复。下图是这种情况的一个例子。

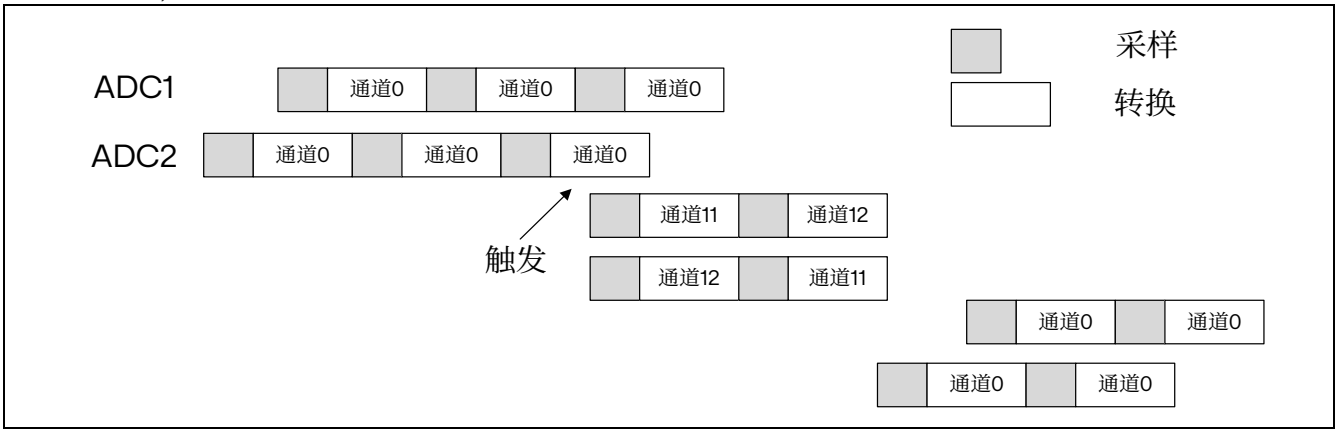



图 15.18 交叉的单通道转换被注入序列 CH11 和 CH12 中断

## 15.5 温度传感器（TPS）

温度传感器可以用来测量器件周围的温度（$T_A$）。

温度传感器在内部和 ADC2 的通道 15 相连，此通道把传感器输出的电压转换成数字值。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同（最多相差 45°C）。

*注：必须设置* TSVREFE *位激活内部通道：* ADC2_IN15 *（*TPS*）和* ADC3_IN13 *（*VREFINT*）的转换。*

内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

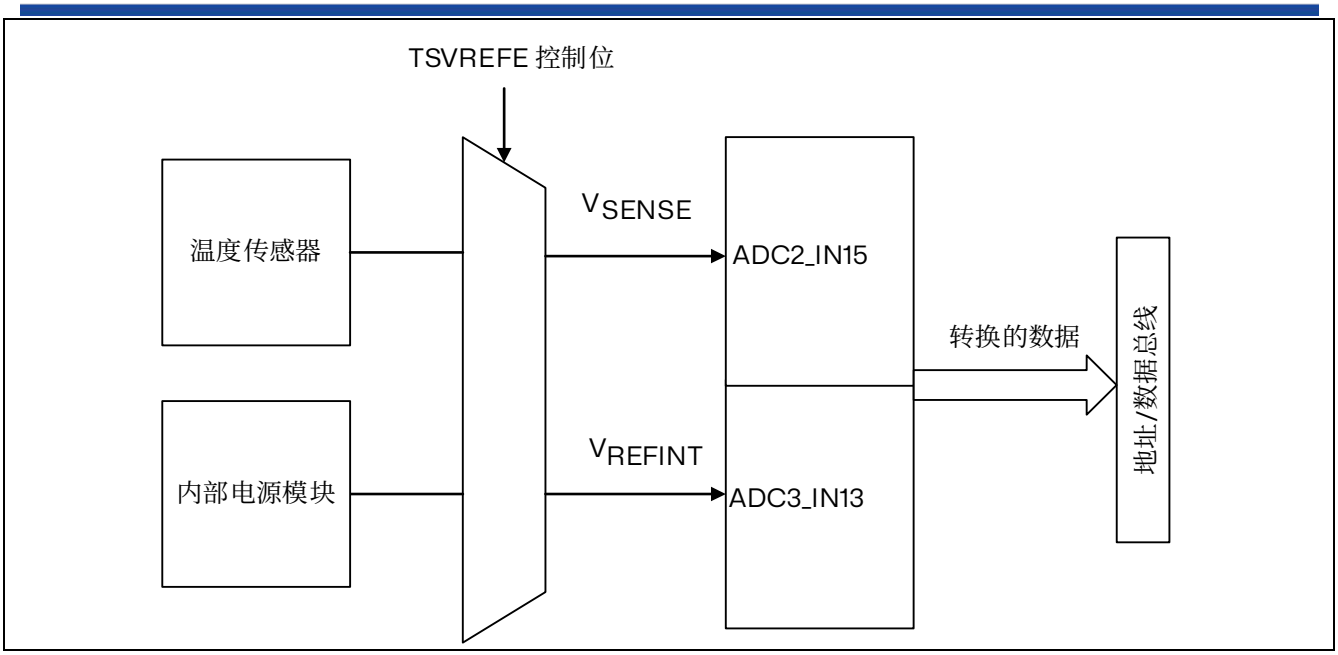



图 15.19 温度传感器和 $V_{REFINT}$ 通道框图

## 15.6 ADC 中断

规则和注入组转换结束时能产生中断，当模拟看门狗状态位被设置时也能产生中断。它们都有独立的中断使能位。

*注：ADC1 和 ADC2 的中断映射在同一个中断向量上，而 ADC3 的中断有自己的中断向量。*

ADC_SR 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTRT（注入组通道转换的启动）
- STRT（规则组通道转换的启动）

表 15.13 ADC 中断

| 中断事件 | 事件标志 | 使能控制位 |
|---|---|---|
| 规则组转换结束 | EOC | EOCIE |
| 注入组转换结束 | JEOC | JEOCIE |
| 设置了模拟看门狗状态位 | AWD | AWDIE |

## 15.7 ADC 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器必须按字（32 位）进行访问。

### 15.7.1 ADC 状态寄存器（ADC_SR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | STRT | JSTRT | JEOC | EOC | AWD |
| | | | | | | | | | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

Bit 31:5 保留，必须保持为复位值。

Bit 4 **STRT**：规则通道开始位

该位由硬件在规则通道转换开始时设置，由软件清除。

0：规则通道转换未开始

1：规则通道转换已开始

Bit 3 **JSTRT**：注入通道开始位

该位由硬件在注入通道转换开始时设置，由软件清除。

0：注入通道转换未开始

1：注入通道转换已开始

Bit 2 **JEOC**：注入通道转换结束位

该位由硬件在所有注入通道组转换结束时设置，由软件清除。

0：转换未完成

1：转换完成

Bit 1 **EOC**：转换结束位

该位由硬件在（规则或注入）通道组转换结束时设置，由软件清除或由读取 ADC_DR 时清除。

0：转换未完成

1：转换完成

Bit 0 **AWD**：模拟看门狗标志位

该位由硬件在转换的电压值超出了 ADC_LTR 和 ADC_HTR 寄存器定义的范围时设置，由软件清除。

0：没有发生模拟看门狗事件

1：发生模拟看门狗事件

### 15.7.2 ADC 控制寄存器 1（ADC_CR1）

地址偏移：0x04

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | AWDFILT[2:0] | | | AWDEN | JAWDEN | Reserved | | DUALMOD[3:0] | | | |
| | | | | | rc_w0 | rc_w0 | rc_w0 | rw | rw | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISCNUM[2:0] | | | JDISCEN | DISCEN | JAUTO | AWDSGL | SCAN | JEOCIE | AWDIE | EOCIE | AWDCH[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:27 保留，必须保持为复位值。

Bit 26:24 **AWDFILT[2:0]：**模拟看门狗滤波参数

当 AWDEN 和 JAWDEN 位都为 0 或监测通道改变时，AWDFILT 计数器清零。

000：无滤波

001：连续 2 次检测到超过模拟看门狗阈值才会置位 AWD 标志位

…

111：连续 8 次检测到超过模拟看门狗阈值才会置位 AWD 标志位

Bit 23 **AWDEN：**在规则通道上开启模拟看门狗

该位由软件设置和清除。

0：在规则通道上禁用模拟看门狗

1：在规则通道上使用模拟看门狗

Bit 22 **JAWDEN：**在注入通道上开启模拟看门狗

该位由软件设置和清除。

0：在注入通道上禁用模拟看门狗

1：在注入通道上使用模拟看门狗

Bit 21:20 保留，必须保持为复位值。

Bit 19:16 **DUALMOD[3:0]：**双模式选择

软件使用这些位选择操作模式。

0000：独立模式

0001：混合的同步规则+注入同步模式

0010：混合的同步规则+交替触发模式

0011：混合同步注入+快速交叉模式

0100：混合同步注入+慢速交叉模式

0101：注入同步模式

0110：规则同步模式

0111：快速交叉模式

1000：慢速交叉模式

1001：交替触发模式

*注：在 ADC2 和 ADC3 中这些位为保留位*

*在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。*

Bit 15:13 **DISCNUM[2:0]：**间断模式通道计数

软件通过这些位定义在间断模式下，收到外部触发后转换规则通道的数目。

000：1 个通道

001：2 个通道

.....

111：8 个通道

Bit 12 **JDISCEN：**注入通道上的间断模式

该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式。

0：注入通道组上禁用间断模式

1：注入通道组上使用间断模式

Bit 11 **DISCEN：**规则通道上的间断模式

该位由软件设置和清除，用于开启或关闭规则通道组上的间断模式。

0：规则通道组上禁用间断模式

1：规则通道组上使用间断模式

Bit 10 **JAUTO：**自动的注入通道组转换

该位由软件设置和清除，用于开启或关闭规则通道组转换结束后自动的注入通道组转换。

0：关闭自动的注入通道组转换

1：开启自动的注入通道组转换

Bit 9 **AWDSGL：**扫描模式中在一个单一的通道上使用看门狗

该位由软件设置和清除，用于开启或关闭由 AWDCH[4:0]位指定的通道上的模拟看门狗功能

0：在所有的通道上使用模拟看门狗

1：在单一通道上使用模拟看门狗

Bit 8 **SCAN：**扫描模式

该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，转换由 ADC_SQRx 或 ADC_JSQR 寄存器选中的通道。

0：关闭扫描模式

1：使用扫描模式

*注：如果分别设置了*EOCIE *或*JEOCIE *位，只在最后一个通道转换完毕后才会产生*EOC *或*JEOC *中断。*

Bit 7 **JEOCIE：**允许产生注入通道转换结束中断

该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。

0：禁止 JEOC 中断

1：允许 JEOC 中断。当硬件设置 JEOC 位时产生中断。

Bit 6 **AWDIE：**允许产生模拟看门狗中断

该位由软件设置和清除，用于禁止或允许模拟看门狗产生中断。在扫描模式下，如果看门狗检测到超范围的数值时，只有在设置了该位时扫描才会中止。

0：禁止模拟看门狗中断

1：允许模拟看门狗中断

Bit 5 **EOCIE：**允许产生 EOC 中断

该位由软件设置和清除，用于禁止或允许转换结束后产生中断。
0：禁止 EOC 中断
1：允许 EOC 中断。当硬件设置 EOC 位时产生中断。

Bit 4:0 **AWDCH[4:0]：**模拟看门狗通道选择位
这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。
0000：ADC 模拟输入通道 0
0001：ADC 模拟输入通道 1
.....
1011：ADC 模拟输入通道 11
1100：ADC 模拟输入通道 12
1101：ADC 模拟输入通道 13
1110：ADC 模拟输入通道 14
1111：ADC 模拟输入通道 15

### 15.7.3 ADC 控制寄存器 2（ADC_CR2）

地址偏移：0x08
复位值：0x0100 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TS VREFE | SW START | JSW START | EXT TRIG | EXTSEL[3:0] | | | |
| | | | | | | | | r | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JEXT TRIG | JEXTSEL[3:0] | | | | ALIGN | Res. | DMA | Reserved | | | | RST CAL | CAL | CONT | ADON |
| rw | rw | rw | rw | rw | rw | | rw | | | | | rw | rw | rw | rw |

Bit 31:24 保留，必须保持为复位值。

Bit 23 **TSVREFE：**温度传感器和 $V_{REFINT}$ 使能
该位由软件设置和清除，用于开启或禁止温度传感器和 $V_{REFINT}$ 通道。该位仅出现在 ADC1 中。
0：禁止温度传感器和 $V_{REFINT}$
1：使能温度传感器和 $V_{REFINT}$

Bit 22 **SWSTART：**开始转换规则通道
由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 EXTSEL[3:0]位中选择了 SWSTART 为触发事件，该位用于启动一组规则通道的转换，
0：复位状态；
1：开始转换规则通道。

Bit 21 **JSWSTART：**开始转换注入通道
由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 JEXTSEL[3:0]位中选择了 JSWSTART 为触发事件，该位用于启动一组注入通道的转换。
0：复位状态
1：开始转换注入通道

Bit 20 **EXTTRIG：**规则通道的外部触发转换模式

该位由软件设置和清除，用于开启或禁止可以启动规则通道组转换的外部触发事件。

0：不用外部事件启动转换

1：使用外部事件启动转换

Bit 19:16 **EXTSEL[3:0]：**选择启动规则通道组转换的外部事件

这些位选择用于启动规则通道组转换的外部事件。

ADC1/2/3 的触发配置如下：

0000：TIM1_TRGO

0001：TIM1_TRGO2

0010：TIM8_TRGO

0011：TIM8_TRGO2

0100：TIM2_TRGO

0101：TIM3_TRGO

0110：TIM4_TRGO

0111：SWSTART

1000：ADC1：EXTI2，ADC2：EXTI3，ADC3：EXTI4

1001：TIM6_TRGO

1010：ADC1：hrtim_adc_trg1，ADC2：hrtim_adc_trg3，ADC3：hrtim_adc_trg5

1011：hrtim_adc_trg6

1100：hrtim_adc_trg7

1101：hrtim_adc_trg8

1110：hrtim_adc_trg9

1111：hrtim_adc_trg10

Bit 15 **JEXTTRIG：**注入通道的外部触发转换模式

该位由软件设置和清除，用于开启或禁止可以启动注入通道组转换的外部触发事件。

0：禁止外部事件启动转换

1：使用外部事件启动转换

Bit 14:11 **JEXTSEL[3:0]：**选择启动注入通道组转换的外部事件

这些位选择用于启动注入通道组转换的外部事件。

ADC1/2/3 的触发配置如下：

0000：TIM1_TRGO

0001：TIM1_TRGO2

0010：TIM8_TRGO

0011：TIM8_TRGO2

0100：ADC1：TIM1_OC4/OC5，ADC2：TIM8_OC4/OC5，
ADC3：TIM8_OC4/OC5

0101：ADC1：TIM2_TRGO，ADC2：TIM3_TRGO，ADC3：TIM4_TRGO

0110：ADC1：EXTI13，ADC2：EXTI14，ADC3：EXTI15

0111：JSWSTART

1000：TIM15_TRGO

1001：TIM7_TRGO

1010：ADC1：hrtim_adc_trg2，ADC2：hrtim_adc_trg4，ADC3：hrtim_adc_trg5

1011：hrtim_adc_trg6
1100：hrtim_adc_trg7
1101：hrtim_adc_trg8
1110：hrtim_adc_trg9
1111：hrtim_adc_trg10

Bit 10 **ALIGN**：数据对齐
该位由软件设置和清除。详情请参阅参考章节：数据对齐。
0：右对齐
1：左对齐

Bit 9 保留，必须保持为复位值。

Bit 8 **DMA**：直接存储器访问模式
该位由软件设置和清除。详见 DMA 控制器章节。
0：不使用 DMA 模式
1：使用 DMA 模式

Bit 7:4 保留，必须保持为复位值。

Bit 3 **RSTCAL**：复位校准
该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。
0：校准寄存器已初始化
1：初始化校准寄存器

*注：如果正在进行转换时设置* RSTCAL，*清除校准寄存器需要额外的周期。*

Bit 2 **CAL**：A/D 校准
该位由软件设置以开始校准，并在校准结束时由硬件清除。
0：校准完成
1：开始校准

Bit 1 **CONT**：连续转换
该位由软件设置和清除。如果设置了此位，则规则组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

Bit 0 **ADON**：开/关 A/D 转换器
该位由软件设置和清除。当该位为 0 时，写入 1 将把 ADC 从断电模式下唤醒。当该位为 1 时，写入 1 将启动转换。应用程序需注意，在转换器上电至转换开始有一个延迟 $t_{STAB}$。
0：关闭 ADC 转换/校准，并进入断电模式
1：开启 ADC 并启动转换

*注：如果在这个寄存器中与* ADON *一起还有其他位被改变，则转换不被触发。这是为了防止触发错误的转换。*

### 15.7.4 ADC 采样时间寄存器 1（ADC_SMPR1）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SMP15[2:1] | |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP15[0] | SMP14[2:0] | | | SMP13[2:0] | | | SMP12[2:0] | | | SMP11[2:0] | | | SMP10[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:18 保留，必须保持为复位值。

Bit 17:0 **SMPx[2:0]：**通道 x 的采样时间（x = 15...10）

这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。

000：3.5 周期

001：7.5 周期

010：11.5 周期

011：13.5 周期

100：28.5 周期

101：55.5 周期

110：71.5 周期

111：640.5 周期

### 15.7.5 ADC 采样时间寄存器 2（ADC_SMPR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SMP9[2:0] | | | SMP8[2:0] | | | SMP7[2:0] | | | SMP6[2:0] | | | SMP5[2:1] | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP5[0] | SMP4[2:0] | | | SMP3[2:0] | | | SMP2[2:0] | | | SMP1[2:0] | | | SMP0[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:30 保留，必须保持为复位值。

Bit 29:0 **SMPx[2:0]：**通道 x 的采样时间（x = 9..0）

这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。

000：3.5 周期

001：7.5 周期

010：11.5 周期

011：13.5 周期

100：28.5 周期

101：55.5 周期

110：71.5 周期

111：640.5 周期

### 15.7.6 ADC 注入通道数据偏移寄存器 x（ADC_JOFRx）（x = 1~4）

地址偏移：0x14 - 0x20

复位值：0x0000 0000

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

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

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **JOFFSETx[11:0]：**注入通道 x 的数据偏移（x = 1...4）

当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。转换的结果可以在 ADC_JDRx 寄存器中读出。

### 15.7.7 ADC 看门狗高阈值寄存器（ADC_HTR）

地址偏移：0x24

复位值：0x0000 0FFF

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

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

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **HTR[11:0]：**模拟看门狗高阈值

这些位定义了模拟看门狗的阈值高限。

### 15.7.8 ADC 看门狗低阈值寄存器（ADC_LTR）

地址偏移：0x28

复位值：0x0000 0000

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

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

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **LTR[11:0]：**模拟看门狗低阈值

这些位定义了模拟看门狗的阈值低限。

### 15.7.9 ADC 规则序列寄存器 1（ADC_SQR1）

地址偏移：0x2C

复位值：0x0000 0000

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

Bit 31:23 保留，必须保持为复位值。

Bit 22:20 **L[2:0]：**规则通道序列长度

这些位由软件定义在规则通道转换序列中的通道数目。

000：1 个转换

001：2 个转换

.....

111：8 个转换

Bit 19:0 保留，必须保持为复位值。

### 15.7.10 ADC 规则序列寄存器 2（ADC_SQR2）

地址偏移：0x30

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | SQ8[4:0] | | | | | SQ7[4:0] | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:10 保留，必须保持为复位值。

Bit 9:0 **SQx[3:0]：**规则序列中的第 x 个转换（x = 8...7）

这些位由软件定义转换序列中的第 x 个转换通道的编号

### 15.7.11 ADC 规则序列寄存器 3（ADC_SQR3）

地址偏移：0x34

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SQ6[4:0] | | | | | SQ5[4:0] | | | | | SQ4[4:1] | | | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SQ4[0] | SQ3[4:0] | | | | | SQ2[4:0] | | | | | SQ1[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:30 保留，必须保持为复位值。

Bit 29:0 **SQx[3:0]：**规则序列中的第 x 个转换（x = 6...1）

这些位由软件定义转换序列中的第 x 个转换通道的编号

### 15.7.12 ADC 注入序列寄存器（ADC_JSQR）

地址偏移：0x38

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | JL[1:0] | | JSQ4[4:1] | | | |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JSQ4[0] | JSQ3[4:0] | | | | | JSQ2[4:0] | | | | | JSQ1[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:22 保留，必须保持为复位值。

Bit 21:20 **JL[1:0]：**注入通道序列长度

这些位由软件定义在规则通道转换序列中的通道数目。

00：1 个转换

01：2 个转换

10：3 个转换

11：4 个转换

Bit 19:0 **JSQx [4:0]：**注入序列中的第 x 个转换（x = 4...1）

这些位由软件定义转换序列中的第 x 个转换通道的编号

*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。例如：*ADC_JSQR[21:0] = 10 00011 00011 00111 00010*，意味着扫描转换将按下列通道顺序转换：*7、3、3，*而不是* 2、7、3。

### 15.7.13 ADC 注入数据寄存器 x（ADC_JDRx）(x = 1~4)

地址偏移：0x3C-0x48

复位值：0x0000 0000

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

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **JDATA[15:0]：**注入转换的数据

这些位为只读，包含了注入通道的转换结果。数据是左对齐或右对齐。详情请参阅参考章节：数据对齐。

### 15.7.14 ADC 规则数据寄存器（ADC_DR）

地址偏移：0x4C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| DATA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:16 **ADC2DATA[15:0]：**ADC2 转换的数据

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 和 ADC3 中：不使用这些位。

Bit 15:0 **DATA[15:0]：**规则转换的数据

这些位为只读，包含了规则通道的转换结果。数据是左对齐或右对齐。详情请参阅参考章节：数据对齐。

### 15.7.15 ADC 控制寄存器 3（ADC_CR3）

地址偏移：0x50

复位值：0x0000 0000

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

Bit 31:17 保留，必须保持为复位值。

Bit 16 **BASW：**ADC 采样遮蔽长度

0：遮蔽长度为 ADC 采样周期

1：遮蔽长度为 ADC 采样周期+12.5T

Bit 15:0 保留，必须保持为复位值。

### 15.7.16 ADC 校准寄存器（ADC_CAL）

地址偏移：0x54

复位值：0x0000 0040

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

Bit 31:7 保留，必须保持为复位值。

Bit 6:0 **DATA[6:0]：**ADC 自校准的值写入该寄存器

可以手动写入值，在 ADC 转换中生效

### 15.7.17 ADC 控制寄存器 4（ADC_CR4）

地址偏移：0x5C

复位值：0x0000 0000

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

Bit 31:6 保留，必须保持为复位值。

Bit 5 **EOC_OPT：**转换结束标志位 EOC 控制位

0：规则或注入通道组转换结束时设置 ADC_SR 寄存器的 EOC

1：规则通道组转换结束时设置 ADC_SR 寄存器的 EOC

Bit 4:0 保留，必须保持为复位值。

### 15.7.18 ADC TPS 校准寄存器（ADC_TPSOFFSET）

地址偏移：0x64

复位值：0x0000 0000

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

Bit 31:8 保留，必须保持为复位值。

Bit 7:0 **OFFSET[7:0]：**TPS 的校准值写入该寄存器

可以手动写入值，也可以自加载

转换 TPS 时生效，转换值+OFFSET

其中 OFFSET[7]为符号位

### 15.7.19 ADC 规则数据寄存器 x（ADC_DRx）(x = 1~8)

地址偏移：0x68-0x84

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DRx_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 |
| DRx_DATA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:16 **DRx_ADC2DATA[15:0]：** ADC2 转换的规则通道 x 数据

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 和 ADC3 中：不使用这些位。

Bit 15:0 **DRx_DATA[15:0]：**转换的规则通道 x 数据

这些位为只读，包含了规则通道的转换结果。数据是左对齐或右对齐。

# 16 数模转换器（DAC）

## 16.1 DAC 简介

DAC 模块是 12 位电压输出数模转换器。DAC 可以按 8 位或 12 位模式进行配置，并且可与 DMA 控制器配合使用。在 12 位模式下，数据可以采用左对齐或右对齐。DAC 有两个输出通道，每个通道各有一个转换器。在 DAC 双通道模式下，每个通道可以单独进行转换；当两个通道组合在一起同步执行更新操作时，也可以同时进行转换。可通过一个输入参考电压引脚 VREF+（与其他模拟外设共享）来提高分辨率。还可以在同一输入上设置内部参考。请参见参考电压缓冲器（VREFBUF）一节。

如果 DAC 输出与输出焊盘断开连接并连接到片上外设，DACx_OUTy 引脚可用作通用输入/输出（GPIO）。可选择使能 DAC 输出缓冲器，从而支持高驱动输出电流。可分别对每条 DAC 输出通道应用校准。

## 16.2 DAC 主要特征

DAC 的主要特性如下（请参见下图：双通道 DAC 框图）

- 6 个 12 位 DAC 通道
- 12 位模式下数据采用左对齐或右对齐
- 同步更新功能
- 噪声波和三角波生成
- 锯齿波生成
- DAC 双通道单独或同时转换
- 每条通道的 DMA 功能，包括 DMA 下溢错误检测
- 通过外部触发信号进行转换
- DAC 输出通道缓冲/非缓冲模式
- 缓冲器偏移校准
- 每路 DAC 输出均可与 DAC_OUTx 输出引脚断开连接
- DAC 输出可与片上外设连接
- 来自 $V_{REF+}$引脚的输入参考电压或内部 VREFBUF 参考电压

下图所示为 DAC 通道的框图，下表则给出了引脚说明。

## 16.3 DAC 实现

表 16.1 DAC 特征

| DAC 特征 | DAC1 | DAC2 | DAC3 |
|---|---|---|---|
| 双通道 | DAC1_CH1<br>DAC1_CH2 | DAC2_CH1<br>DAC2_CH2 | DAC3_CH1<br>DAC3_CH2 |
| 输出缓冲(1)(2) | DAC1_OUT1<br>DAC1_OUT2 | DAC2_OUT1 | 没有输出缓冲 |
| I/O 连接(1)(2) | DAC1_OUT1 在 PE9<br>DAC1_OUT2 在 PE10 | DAC2_OUT1 在 PE13 | 没有连接至 GPIO |
| 最大采样速率 | 1 Msps | | |
| $V_{REF+}$引脚 | 可作为参考电压 | | |

（1） 仅 DAC1_OUT1、DAC1_OUT2、DAC2_OUT1 有输出缓冲，可通过配置 DAC_CR 寄存器的 BOFF1 位和 BOFF2 位开启。

（2） DAC 输出缓冲器开启时只能对外输出至 IO。输出缓冲器关闭时只能走内部通道给 COMP，内外部通道无法同时使用。

## 16.4 DAC 功能描述

### 16.4.1 DAC 框图

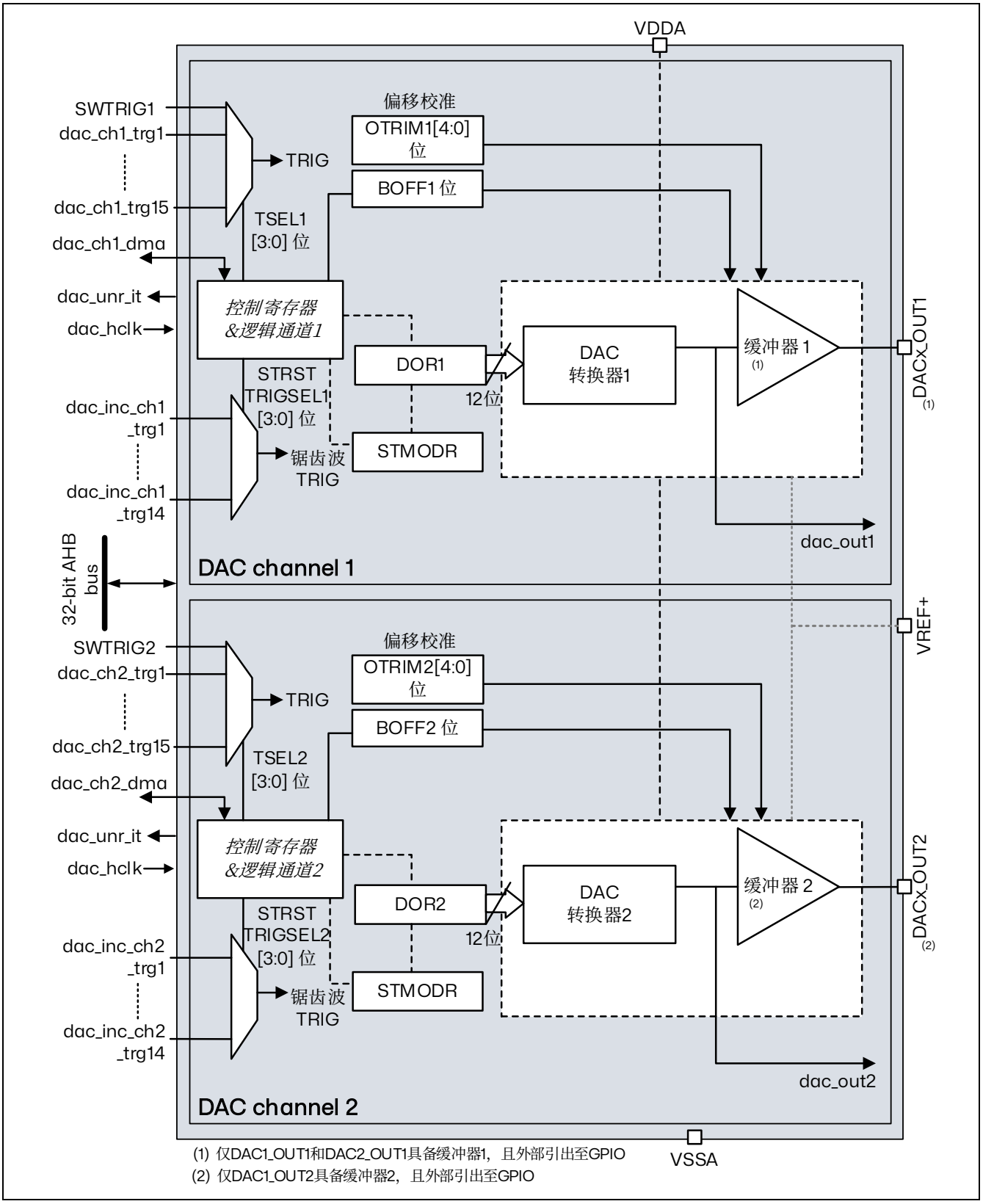



图 16.1 双通道 DAC 框图

### 16.4.2 DAC 引脚和内部信号

DAC 包含：

- 多达两条输出通道
- DACx_OUTy 可与输出引脚断开连接并用作普通 GPIO
- dac_outx 可使用与片上外设（比如比较器）的内部引脚连接。
- DAC 输出通道（缓冲/非缓冲）

DAC 包含多达两条独立的输出通道。每条输出通道均可连接到片上外设，比如 COMP。在这种情况下，DAC 输出通道可与 DACx_OUTy 输出引脚断开连接，相应的 GPIO 可用于其他用途。

当选择 DAC 通道内部连接至 COMP 反向输入端时，需要关闭 DAC 输出缓冲器。

当选择 DAC 通道外部连接至 GPIO 时，需要打开 DAC 输出缓冲器。仅 DAC1_OUT1、DAC1_OUT2、DAC2_OUT3 可缓冲。

表 16.2 DAC 输入/输出引脚

| 引脚名称 | 信号类型 | 注释 |
|---|---|---|
| VREF+ | 正模拟参考电压输入 | DAC 高/正参考电压，$V_{REF+} \le V_{DDAmax}$（请参见数据手册） |
| VDDA | 模拟电源输入 | 模拟电源 |
| VSSA | 模拟电源地输入 | 模拟电源地 |
| DACx_OUTy | 模拟输出信号 | DACx 通道 y 模拟输出，连接到 GPIO |

表 16.3 DAC 输入/输出信号

| 内部信号名称 | 信号类型 | 说明 |
|---|---|---|
| dac_ch1_dma | 双向 | DAC 通道 1 DMA 请求/确认 |
| dac_ch2_dma | 双向 | DAC 通道 2 DMA 请求/确认 |
| dac_ch1_trgx（x = 1...15） | 输入 | DAC 通道 1 触发输入 |
| dac_ch2_trgx（x = 1...15） | 输入 | DAC 通道 2 触发输入 |
| dac_ch1_inc_trgx（x = 1...15） | 输入 | DAC 通道 1 锯齿波增量触发输入 |
| dac_ch2_inc_trgx（x = 1...15） | 输入 | DAC 通道 2 锯齿波增量触发输入 |
| dac_unr_it | 输出 | DAC 下溢中断 |
| dac_hclk | 输入 | DAC 外设时钟 |
| dac_out1 | 模拟输出 | 片上外设的 DAC 通道 1 输出 |
| dac_out2 | 模拟输出 | 片上外设的 DAC 通道 2 输出 |

表 16.4 DAC1 通道 1 互连

| 信号名 | 源 | 源类型 | TSEL1[3:0] |
|---|---|---|---|
| SWTRIG1 | SWTRIG | 软件控制位 | 0000 |
| dac_ch1_trg1 | TIM8_TRGO | 片上定时器的内部信号 | 0001 |
| dac_ch1_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_ch1_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_ch1_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_ch1_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_ch1_trg6 | EXTI9 | 外部引脚 | 0110 |
| dac_ch1_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_ch1_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_ch1_trg9 | hrtim_dac_reset_trg1 | 片上定时器的内部信号 | 1001 |
| dac_ch1_trg10 | hrtim_dac_reset_trg2 | 片上定时器的内部信号 | 1010 |
| dac_ch1_trg11 | hrtim_dac_reset_trg3 | 片上定时器的内部信号 | 1011 |
| dac_ch1_trg12 | hrtim_dac_reset_trg4 | 片上定时器的内部信号 | 1100 |
| dac_ch1_trg13 | hrtim_dac_reset_trg5 | 片上定时器的内部信号 | 1101 |
| dac_ch1_trg14 | hrtim_dac_reset_trg6 | 片上定时器的内部信号 | 1110 |
| dac_ch1_trg15 | hrtim_dac_trg1 | 片上定时器的内部信号 | 1111 |

表 16.5 DAC1 通道 2 互连

| 信号名 | 源 | 源类型 | TSEL2[3:0] |
|---|---|---|---|
| SWTRIG2 | SWTRIG | 软件控制位 | 0000 |
| dac_inc_ch2_trg1 | TIM8_TRGO | 片上定时器的内部信号 | 0001 |
| dac_inc_ch2_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_inc_ch2_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_inc_ch2_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_inc_ch2_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_inc_ch2_trg6 | EXTI10 | 外部引脚 | 0110 |
| dac_inc_ch2_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_inc_ch2_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_inc_ch2_trg9 | hrtim_dac_step_trg1 | 片上定时器的内部信号 | 1001 |
| dac_inc_ch2_trg10 | hrtim_dac_step_trg2 | 片上定时器的内部信号 | 1010 |
| dac_inc_ch2_trg11 | hrtim_dac_step_trg3 | 片上定时器的内部信号 | 1011 |
| dac_inc_ch2_trg12 | hrtim_dac_step_trg4 | 片上定时器的内部信号 | 1100 |
| dac_inc_ch2_trg13 | hrtim_dac_step_trg5 | 片上定时器的内部信号 | 1101 |
| dac_inc_ch2_trg14 | hrtim_dac_step_trg6 | 片上定时器的内部信号 | 1110 |

表 16.6 DAC2 通道 1 互连

| 信号名 | 源 | 源类型 | TSEL1[3:0] |
|---|---|---|---|
| SWTRIG1 | SWTRIG | 软件控制位 | 0000 |
| dac_ch1_trg1 | TIM8_TRGO | 片上定时器的内部信号 | 0001 |
| dac_ch1_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_ch1_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_ch1_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_ch1_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_ch1_trg6 | EXTI9 | 外部引脚 | 0110 |
| dac_ch1_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_ch1_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_ch1_trg9 | hrtim_dac_reset_trg1 | 片上定时器的内部信号 | 1001 |
| dac_ch1_trg10 | hrtim_dac_reset_trg2 | 片上定时器的内部信号 | 1010 |
| dac_ch1_trg11 | hrtim_dac_reset_trg3 | 片上定时器的内部信号 | 1011 |
| dac_ch1_trg12 | hrtim_dac_reset_trg4 | 片上定时器的内部信号 | 1100 |
| dac_ch1_trg13 | hrtim_dac_reset_trg5 | 片上定时器的内部信号 | 1101 |
| dac_ch1_trg14 | hrtim_dac_reset_trg6 | 片上定时器的内部信号 | 1110 |
| dac_ch1_trg15 | hrtim_dac_trg2 | 片上定时器的内部信号 | 1111 |

表 16.7 DAC2 通道 2 互连

| 信号名 | 源 | 源类型 | TSEL2[3:0] |
|---|---|---|---|
| SWTRIG2 | SWTRIG | 软件控制位 | 0000 |
| dac_inc_ch2_trg1 | TIM8_TRGO | 片上定时器的内部信号 | 0001 |
| dac_inc_ch2_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_inc_ch2_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_inc_ch2_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_inc_ch2_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_inc_ch2_trg6 | EXTI10 | 外部引脚 | 0110 |
| dac_inc_ch2_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_inc_ch2_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_inc_ch2_trg9 | hrtim_dac_step_trg1 | 片上定时器的内部信号 | 1001 |
| dac_inc_ch2_trg10 | hrtim_dac_step_trg2 | 片上定时器的内部信号 | 1010 |
| dac_inc_ch2_trg11 | hrtim_dac_step_trg3 | 片上定时器的内部信号 | 1011 |
| dac_inc_ch2_trg12 | hrtim_dac_step_trg4 | 片上定时器的内部信号 | 1100 |
| dac_inc_ch2_trg13 | hrtim_dac_step_trg5 | 片上定时器的内部信号 | 1101 |
| dac_inc_ch2_trg14 | hrtim_dac_step_trg6 | 片上定时器的内部信号 | 1110 |

表 16.8 DAC3 通道 1 互连

| 信号名 | 源 | 源类型 | TSEL1[3:0] |
|---|---|---|---|
| SWTRIG1 | SWTRIG | 软件控制位 | 0000 |
| dac_ch1_trg1 | TIM1_TRGO | 片上定时器的内部信号 | 0001 |
| dac_ch1_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_ch1_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_ch1_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_ch1_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_ch1_trg6 | EXTI9 | 外部引脚 | 0110 |
| dac_ch1_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_ch1_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_ch1_trg9 | hrtim_dac_reset_trg1 | 片上定时器的内部信号 | 1001 |
| dac_ch1_trg10 | hrtim_dac_reset_trg2 | 片上定时器的内部信号 | 1010 |
| dac_ch1_trg11 | hrtim_dac_reset_trg3 | 片上定时器的内部信号 | 1011 |
| dac_ch1_trg12 | hrtim_dac_reset_trg4 | 片上定时器的内部信号 | 1100 |
| dac_ch1_trg13 | hrtim_dac_reset_trg5 | 片上定时器的内部信号 | 1101 |
| dac_ch1_trg14 | hrtim_dac_reset_trg6 | 片上定时器的内部信号 | 1110 |
| dac_ch1_trg15 | hrtim_dac_trg3 | 片上定时器的内部信号 | 1111 |

表 16.9 DAC3 通道 2 互连

| 信号名 | 源 | 源类型 | TSEL2[3:0] |
|---|---|---|---|
| SWTRIG2 | SWTRIG | 软件控制位 | 0000 |
| dac_inc_ch2_trg1 | TIM1_TRGO | 片上定时器的内部信号 | 0001 |
| dac_inc_ch2_trg2 | TIM7_TRGO | 片上定时器的内部信号 | 0010 |
| dac_inc_ch2_trg3 | TIM15_TRGO | 片上定时器的内部信号 | 0011 |
| dac_inc_ch2_trg4 | TIM2_TRGO | 片上定时器的内部信号 | 0100 |
| dac_inc_ch2_trg5 | TIM4_TRGO | 片上定时器的内部信号 | 0101 |
| dac_inc_ch2_trg6 | EXTI10 | 外部引脚 | 0110 |
| dac_inc_ch2_trg7 | TIM6_TRGO | 片上定时器的内部信号 | 0111 |
| dac_inc_ch2_trg8 | TIM3_TRGO | 片上定时器的内部信号 | 1000 |
| dac_inc_ch2_trg9 | hrtim_dac_step_trg1 | 片上定时器的内部信号 | 1001 |
| dac_inc_ch2_trg10 | hrtim_dac_step_trg2 | 片上定时器的内部信号 | 1010 |
| dac_inc_ch2_trg11 | hrtim_dac_step_trg3 | 片上定时器的内部信号 | 1011 |
| dac_inc_ch2_trg12 | hrtim_dac_step_trg4 | 片上定时器的内部信号 | 1100 |
| dac_inc_ch2_trg13 | hrtim_dac_step_trg5 | 片上定时器的内部信号 | 1101 |
| dac_inc_ch2_trg14 | hrtim_dac_step_trg6 | 片上定时器的内部信号 | 1110 |

### 16.4.3 DAC 通道使能

将 DACx_CR 寄存器中的相应 ENx 位置 1，即可使能对应 DAC 通道。经过一段启动时间 $t_{WAKEUP}$ 后，DAC 通道被真正使能。

*注：ENx 位只会使能模拟 DAC 通道 x。即使 ENx 位复位，DAC 通道 x 数字接口仍处于使能状态。*

### 16.4.4 DAC 数据格式

根据所选配置模式，数据必须按如下方式写入指定寄存器：

- 对于 DAC 单通道，有三种可能的方式：
  - 8 位右对齐：软件必须将数据加载到 DAC_DHR8Rx[7:0]位（存储到 DHRx[11:4]位）
  - 12 位左对齐：软件必须将数据加载到 DAC_DHR12Lx[15:4]位（存储到 DHRx[11:0]位）
  - 12 位右对齐：软件必须将数据加载到 DAC_DHR12Rx[11:0]位（存储到 DHRx[11:0]位）

根据加载的 DAC_DHRyyyx 寄存器，用户写入的数据将移位并存储到相应的 DHRx（数据保持寄存器 x，即内部非存储器映射寄存器）。之后，DHRx 寄存器将被自动加载，或者通过软件或外部事件触发加载到 DORx 寄存器。

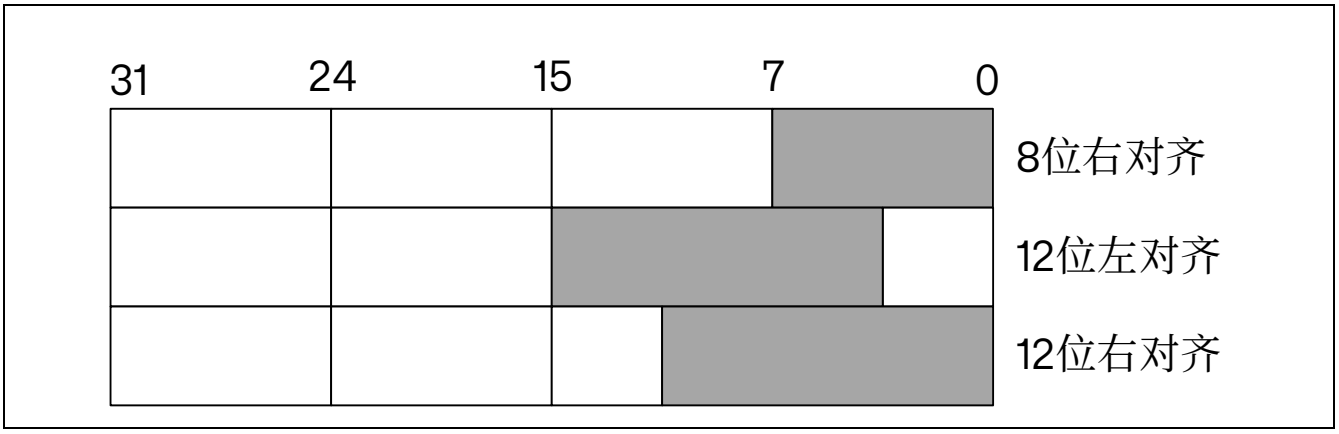



图 16.2 DAC 单通道模式下的数据寄存器

- 对于 DAC 双通道（可用时），有三种可能的方式：
  - 8 位右对齐：将 DAC 通道 1 的数据加载到 DAC_DHR8RD[7:0]位（存储到 DHR1[11:4]位），将 DAC 通道 2 的数据加载到 DAC_DHR8RD[15:8]位（存储到 DHR2[11:4]位）
  - 12 位左对齐：将 DAC 通道 1 的数据加载到 DAC_DHR12LD[15:4]位（存储到 DHR1[11:0]位），将 DAC 通道 2 的数据加载到 DAC_DHR12LD [31:20]位（存储到 DHR2[11:0]位）
  - 12 位右对齐：将 DAC 通道 1 的数据加载到 DAC_DHR12RD[11:0]位（存储到 DHR1[11:0]位），将 DAC 通道 2 的数据加载到 DAC_DHR12RD[27:16]位（存储到 DHR2[11:0]位）

根据加载的 DAC_DHRyyyD 寄存器，用户写入的数据将移位并存储到 DHR1 和 DHR2（数据保持寄存器，即内部非存储器映射寄存器）。之后，DHR1 和 DHR2 寄存器将被自动加载，或者通过软件或外部事件触发分别被加载到 DOR1 和 DOR2 寄存器。

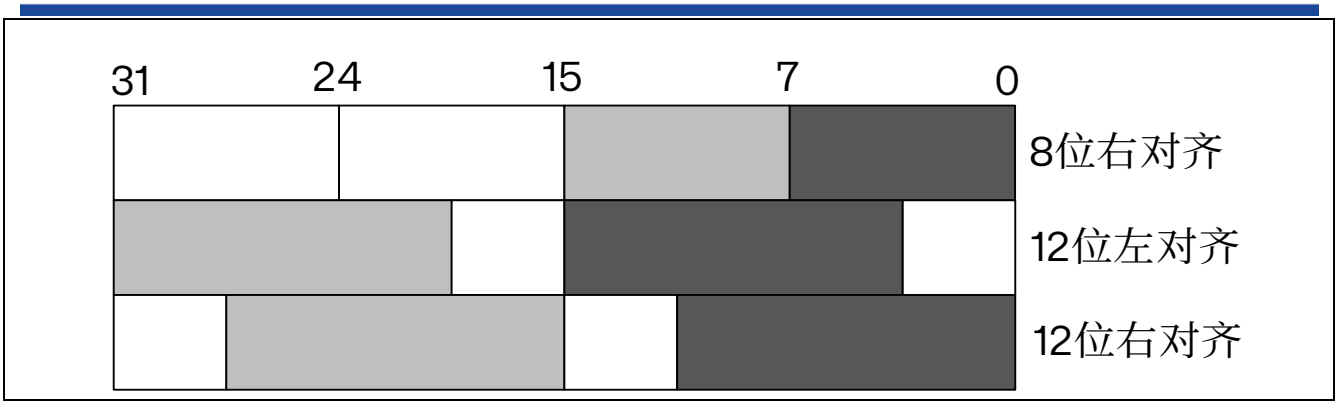



图 16.3 DAC 双通道模式下的数据寄存器

### 16.4.5 DAC 转换

DAC_DORx 无法直接写入，任何数据都必须通过加载 DAC_DHRx 寄存器（写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12RD 或 DAC_DHR12LD）才能传输到 DAC 通道 x。

如果未选择硬件触发（DAC_CR 寄存器中的 TENx 位复位），那么经过一个 dac_hclk 时钟周期后，DAC_DHRx 寄存器中存储的数据将自动转移到 DAC_DORx 寄存器。但是，如果选择硬件触发（置位 DAC_CR 寄存器中的 TENx 位）且触发条件到来，将在三个 dac_hclk 时钟周期后进行转移。

当 DAC_DORx 加载了 DAC_DHRx 内容时，模拟输出电压将在一段时间 $t_{SETTLING}$ 后可用，具体时间取决于电源电压和模拟输出负载。

当 dac_hclk 时钟速度超过 80MHz 时，必须设置 DAC_MCR 的 HFSEL 位。它增加了从 DAC_DHRx 寄存器到 DAC_DORx 寄存器传输的额外延迟。

请参考下面 HFSEL 的描述表，了解 DAC_DORx 更新速率限制与 HFSEL 位和 dac_hclk 时钟频率的关系。

如果在不允许的期间内更新数据或发生软件/硬件触发事件，则外设行为不可预测。

上述时序仅与 DAC 接口的限制有关。请参考产品数据表中的 $t_{SETTLING}$ 参数值。

表 16.10 HFSEL 描述

| HFSEL[1:0] | AHB 频率 | 功能 |
|---|---|---|
| 00 | < 80 MHz | DAC_DOR 更新速率为 3 个 AHB 时钟周期 |
| 01 | ≥ 80 MHz(1) | DAC_DOR 更新速率为 5 个 AHB 时钟周期 |
| 10 | ≥ 160 MHz | DAC_DOR 更新速率为 7 个 AHB 时钟周期 |
| 11 | 保留 | - |

1、请参考设备数据手册以获取最大 AHB 频率值。

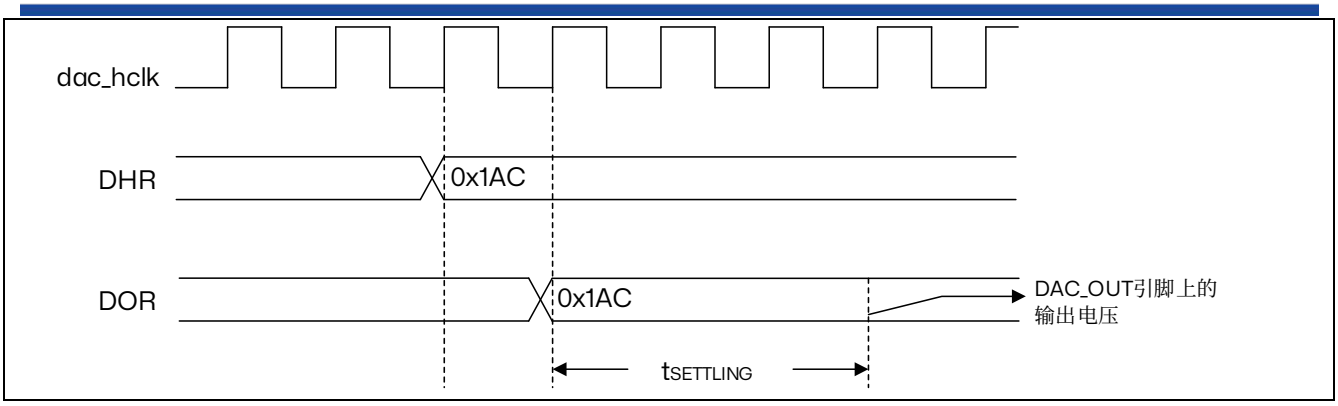



图 16.4 触发禁用 TEN=0 时的转换时序图

### 16.4.6 DAC 输出电压

经过线性转换后，数字输入会转换为 0 到 $V_{REF+}$之间的输出电压。

各 DAC 通道引脚的模拟输出电压通过以下公式确定：

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

### 16.4.7 DAC 触发器选择

如果 TENx 控制位置 1，可通过外部事件（定时计数器、外部中断线）触发转换。TSELx[3:0]控制位将决定通过 16 个可能事件中的哪一个来触发转换，如 DAC_CR 寄存器中的 TSELx[3:0]位所示。这些事件可以是软件触发或硬件触发。请参阅章节：DAC 引脚和内部信号。

每当 DAC 接口在所选触发源上检测到上升沿时（请参见下表），DAC_DHRx 寄存器中存储的最后一个数据即会传输到 DAC_DORx 寄存器中。发生触发后再经过三个 dac_hclk 周期，DAC_DORx 寄存器将会得到更新。

如果选择软件触发，一旦 SWTRIG 位置 1，转换即会开始。DAC_DHRx 寄存器内容加载到 DAC_DORx 寄存器中后，SWTRIG 即由硬件复位。

锯齿波生成的复位触发选择和增量触发选择分别通过 STRSTTRIGSELx 和 STINCTRIGSELx 控制位进行。STRSTTRIGSELx 的映射与 TSELx 类似。请参阅章节：DAC 引脚和内部信号，了解 TSELx、STRSTTRIGSELx 和 STINCTRIGSELx 的映射。

注：ENx 位置 1 时，无法更改 TSELx[3:0]位。

如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 dac_hclk 时钟周期即可转移到 DAC_DORx 寄存器。

### 16.4.8 DMA 请求

每个 DAC 通道都具有 DMA 功能。两个 DMA 通道用于处理 DAC 通道的 DMA 请求。

当 DMAENx 位置 1 时，如果发生外部触发（而不是软件触发），则传输结束后 DAC_DHRx 寄存器的值会转移到 DAC_DORx 寄存器，并且会产生 DMA 请求。

在双通道模式下，如果两个 DMAENx 位均置 1，则将产生两个 DMA 请求。如果只需要一个 DMA 请求，应仅将相应 DMAENx 位置 1。这样，应用程序可以在双通道模式下通过一个 DMA 请求和一个特定 DMA 通道来管理两个 DAC 通道。

由于 DAC_DHRx 已在 DMA 请求之前向 DAC_DORx 传输数据，必须在发生第一个触发事件之前将第一个数据写入 DAC_DHRx。

#### DMA 下溢

DAC 的 DMA 请求没有缓冲队列。这样，如果第二个外部触发到达时尚未收到第一个外部触发的应答，将不会发出新的请求，并且 DAC_SR 寄存器中的 DMA 通道 x 下溢标志 DMAUDRx 将置 1，以报告这一错误状况。DAC 通道 x 仍将继续转换旧有数据。

软件应通过写入 1 来将 DMAUDRx 标志清零，将所用 DMA 数据流的 DMAEN 位清零，并重新初始化 DMA 和 DAC 通道，以便正确地重新开始 DMA 传输。软件应修改 DAC 触发转换频率或减轻 DMA 工作负载，以避免再次发生 DMA 下溢。最后，可通过使能 DMA 数据传输和转换触发来继续完成 DAC 转换。

对于各 DAC 通道 x，如果使能 DAC_CR 寄存器中相应的 DMAUDRIEx 位，还将产生中断。

### 16.4.9 DAC 生成噪音

为了生成可变振幅的伪噪声，可使用 LFSR（线性反馈移位寄存器）。将 WAVEx[1:0]置为“01”即可选择生成噪声。LFSR 中的预加载值为 0xAAA。在每次发生触发事件后，经过三个 dac_hclk 时钟周期，该寄存器会依照特定的计算算法完成更新。

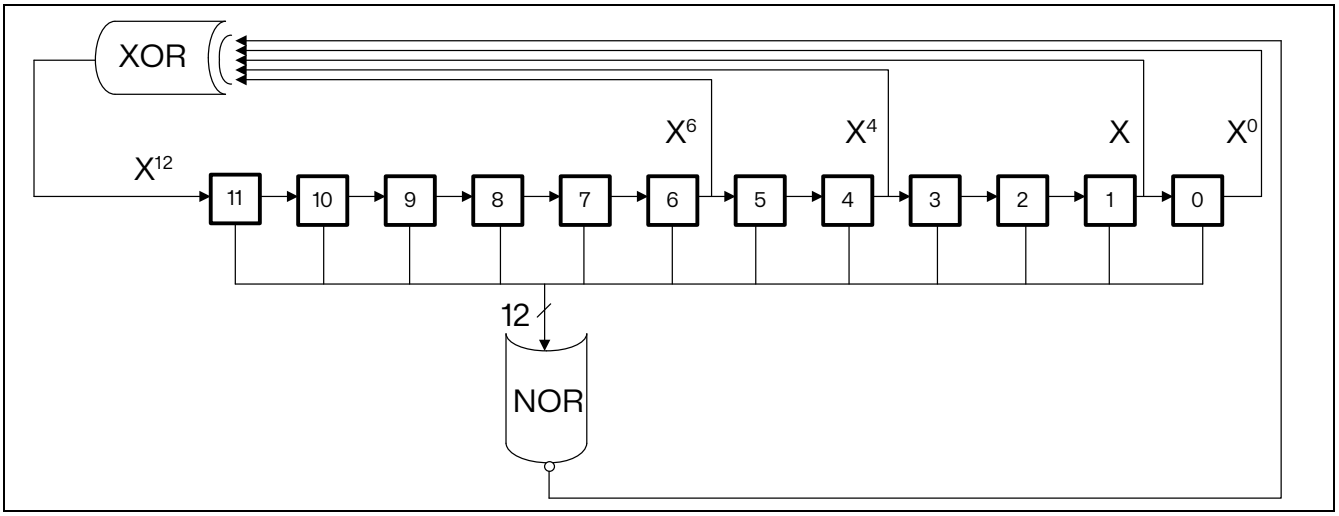



图 16.5 DAC LFSR 寄存器计算算法

LFSR 值可以通过 DAC_CR 寄存器中的 MAMPx[3:0]位来部分或完全屏蔽，在不发生溢出的情况下，该值将与 DAC_DHRx 的内容相加，然后转移到 DAC_DORx 寄存器中。

如果 LFSR 为 0x0000，将向其注入“1”（防锁定机制）。

可以通过复位 WAVEx[1:0]位来将 LFSR 波形产生功能关闭。

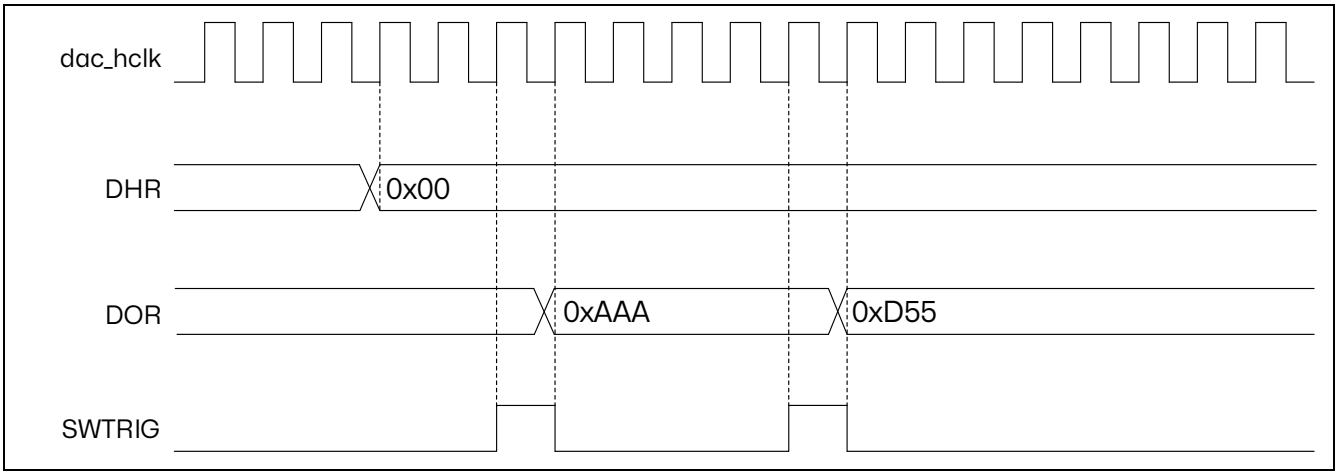



图 16.6 LFSR 产生波形的 DAC 转换（使能软件触发）

*注：要生成噪声，必须通过将* DAC_CR *寄存器中的* TENx *位置* 1 *来使能* DAC *触发。*

### 16.4.10 DAC 生成三角波

可以在直流电流或慢变信号上叠加一个小幅三角波。将 WAVEx[1:0]置为“10”即可选择 DAC 生成三角波。振幅通过 DAC_CR 寄存器中的 MAMPx[3:0]位进行配置。每次发生触发事件后，经过三个 dac_hclk 时钟周期，内部三角波计数器将会递增。在不发生溢出的情况下，该计数器的值将与 DAC_DHRx 寄存器内容相加，所得总和将传输到 DAC_DORx 寄存器中。只要小于 MAMPx[3:0]位定义的最大振幅，三角波计数器就会一直递增。一旦达到配置的振幅，计数器将递减至零，然后再递增，以此类推。

可以通过复位 WAVEx[1:0]位来将三角波产生功能关闭。

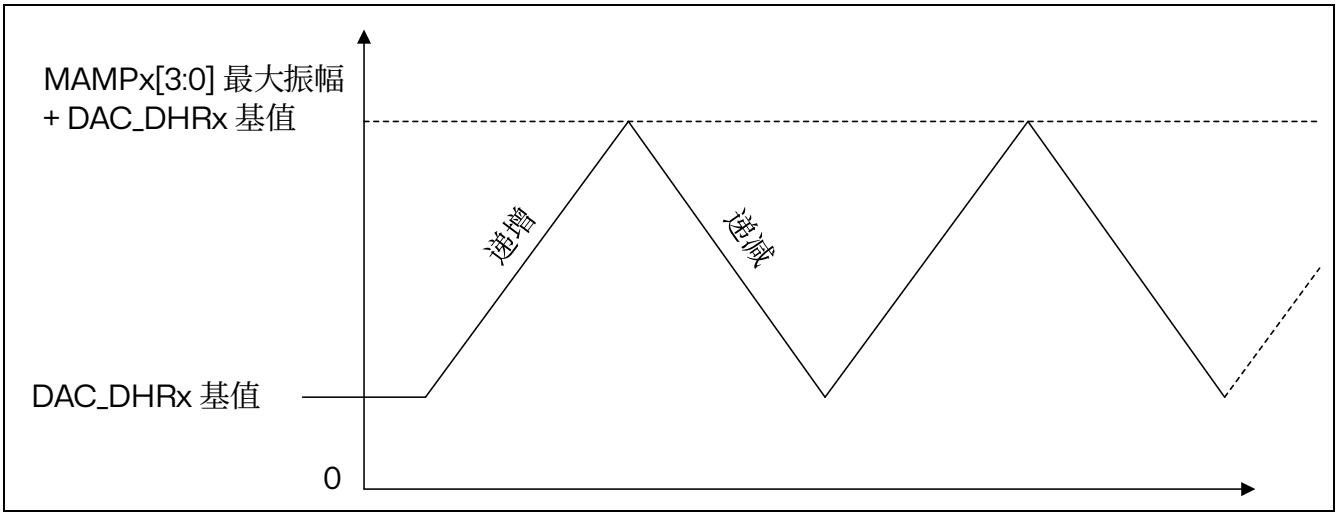



图 16.7 生成 DAC 三角波

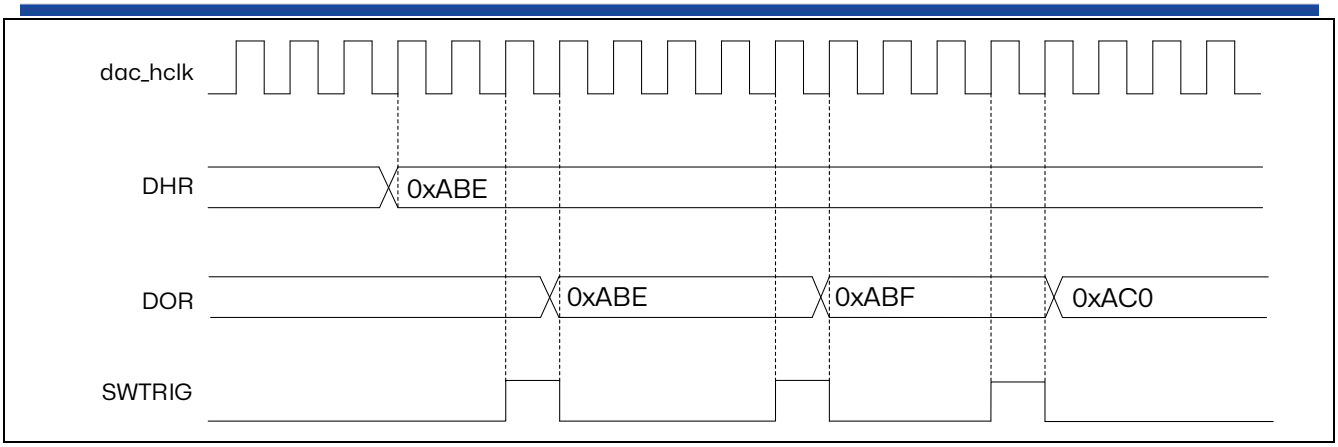



图 16.8 生成三角波波形的 DAC 转换（使能软件触发）

*注：*1. *要生成三角波，必须通过将*DAC_CR *寄存器中的*TENx *位置*1 *来使能* DAC *触发*

2. MAMPx[3:0]*位必须在使能* DAC *之前进行配置，否则将无法更改。*

### 16.4.11 DAC 生成锯齿波

DAC 可以生成锯齿波形。需要为初始值、增量值和方向控制设置特定的寄存器：

- 在 DAC_CR 寄存器中，将 WAVEx[1:0]设置为 11 以选择 DAC 生成锯齿波。
- 锯齿波计数器的初始值（重置值）通过 DAC_STRx 寄存器中的 STRSTDATAx[11:0]位进行配置。
- 增量值由 DAC_STRx 寄存器中的 STINCDATAx[15:0]位定义。
- 锯齿波方向由 DAC_STRx 寄存器中的 STDIRx 位定义。

锯齿波计数器从 STRSTDATAx[11:0]（第 12 位到第 15 位设置为 0000）开始，然后每个增量触发器都会增加（或减少）STINCDATAx[15:0]的值。

DAC 输出使用这些计数器值的 12 个最高有效位（MSB）。当计数器达到 0x0000 或 0xFFFF 时，该值会达到饱和。锯齿波重置触发器信号将计数器值初始化为 STRSTDATAx[11:0]（第 12 位到第 15 位设置为 0000）的值。

必须通过 STINCTRIGSELx[3:0]和 STRSTTRIGSELx[3:0]位来选择增量触发器和重置触发器。

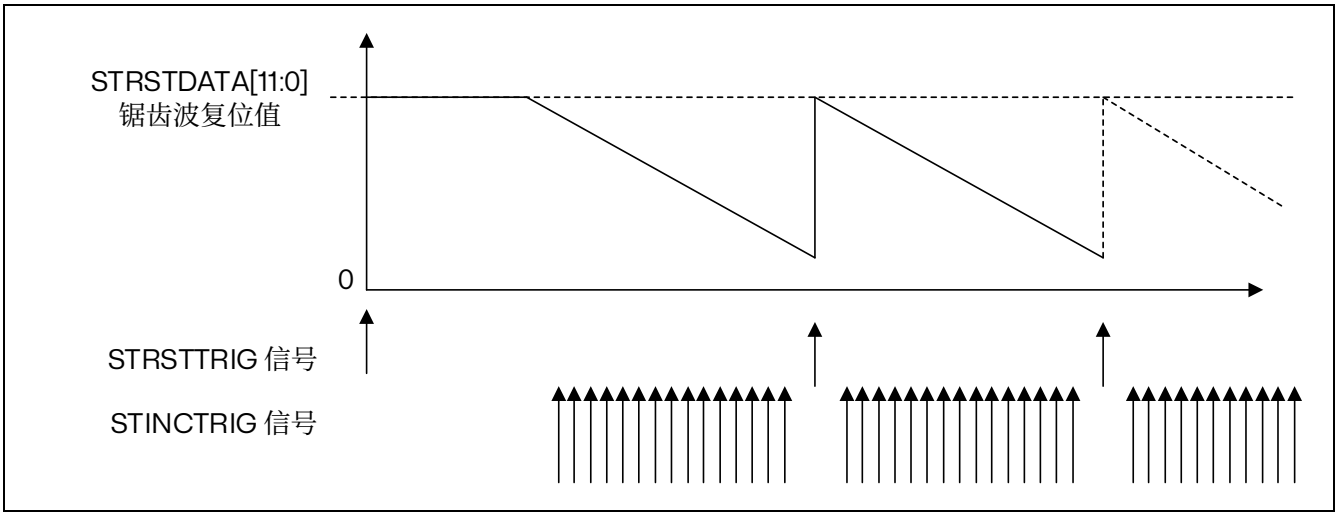



图 16.9 DAC 生成锯齿波（STDIRx = 0）

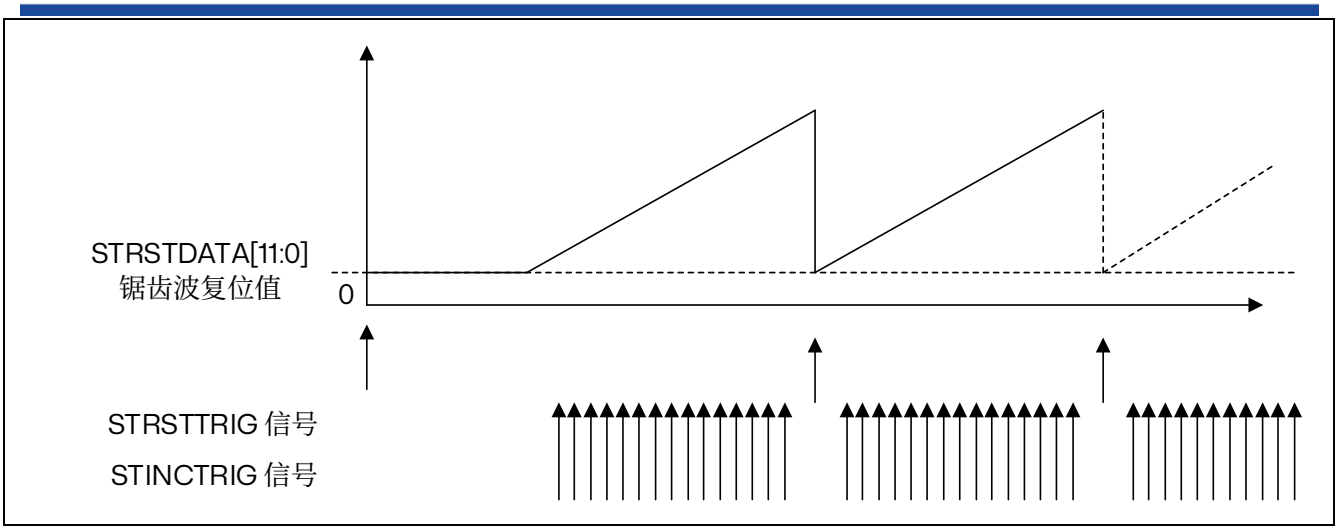



图 16.10 DAC 生成锯齿波（STDIRx = 1）

STRSTTRIG 信号的优先级高于 STINCTRIG。触发信号的速度不能超过表“HFSEL 描述”中定义的 DAC_DORx 更新速率。如果 STINCTRIG 的触发速度比允许的数据更新速率还快，则会忽略 STINCTRIG 触发。如果在 STINCTRIG 触发之后且在 DAC_DORx 更新速率限制之前应用 STRSTTRIG 信号，则 STRSTTRIG 信号将被挂起。然后，在数据递增之后立即应用以复位触发。

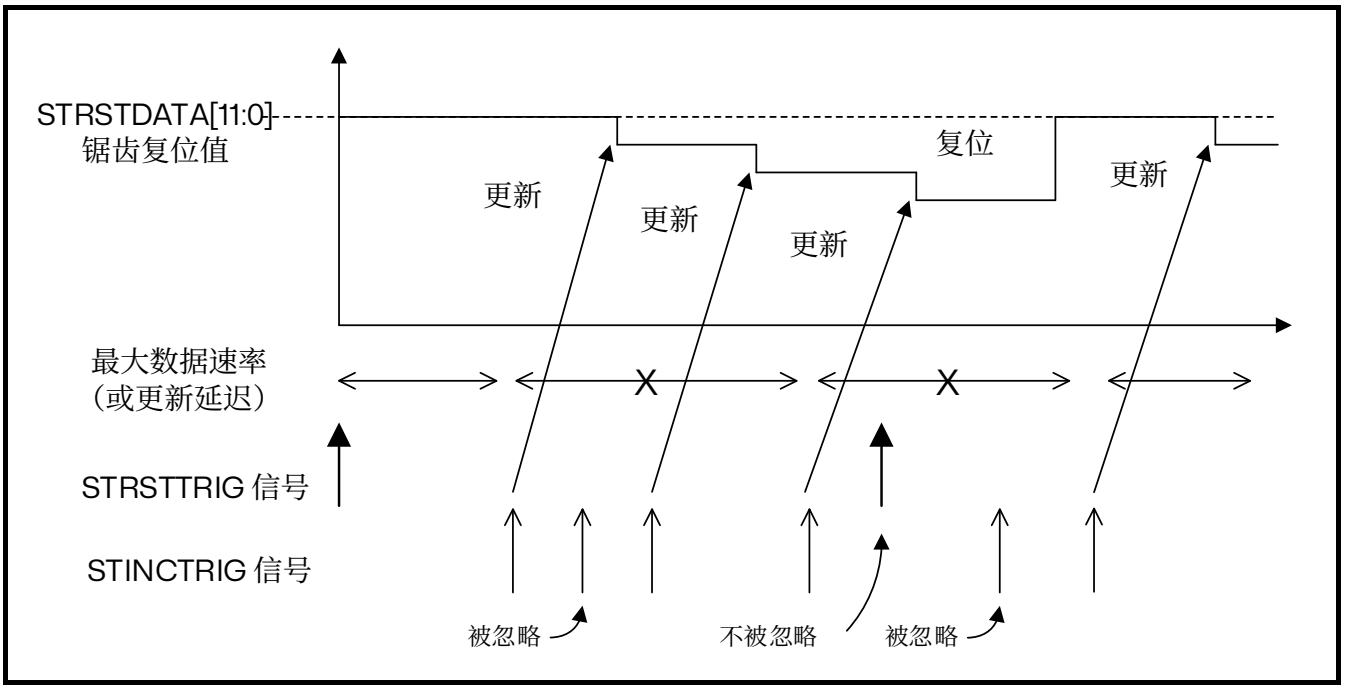



图 16.11 DAC 锯齿波 STINCTRIG 和 STRSTTRIG 优先级（STDIR = 0）

### 16.4.12 DAC 通道缓冲器校准

N 位数模转换器（DAC）的传递函数为：

$$V_{OUT} = ((D/2^N) \times G \times V_{ref}) + V_{OS}$$

其中，VOUT 为模拟输出，D 为数字输入，G 为增益，Vref 为标称满量程电压，Vos 为偏移电压。对于理想的 DAC 通道，G = 1 且 Vos = 0。

由于输出缓冲器所具有的特性，各部分的电压偏移可能有所不同，并会在模拟输出上引入绝对偏移误差。为补偿 Vos，需要通过调整技术进行校准。

仅当 DAC 通道 x 在输出缓冲器使能的条件下运行时，校准才有效（BOFFx = 0b）。如果在缓冲器关闭的条件下应用于其他模式，校准无效。校准过程中：

- 缓冲器输出将与引脚内部/外部连接断开，并会进入高阻态模式（HiZ）。
- 缓冲器将作为比较器来检测中间代码值 0x800，并通过内部桥将其与 VREF+/2 信号进行比较，然后根据比较结果（CAL_FLAGx 位）将其输出信号切换为 0 或 1。

提供以下两种校准技术：

- 出厂调整（始终使能）
  DAC 缓冲器偏移是在出厂时进行调整的。DAC_CCR 寄存器中 OTRIMx[4:0]位的默认值是出厂调整值，会在 DAC 数字接口复位时载入。
- 用户调整
  如果运行条件不同于标称出厂调整条件，特别是 $V_{DDA}$ 电压、温度、VREF+值发生变化时，可进行用户调整，并且用户可在应用过程中随时通过软件进行调整。

*注： 更多关于标称出厂调整条件的详细信息，请参见数据手册。*

另外，如果 $V_{DD}$ 移除（例如器件进入待机模式），则需要进行校准。执行用户调整校准的步骤如下：

1. 如果 DAC 通道激活，可向 DAC_CR 中的 ENx 位写入 0 来禁止通道。
2. 写入 DAC_CR 寄存器，使能 DAC 通道 x 输出缓冲器（BOFFx = 0b），
3. 将 DAC_CR 寄存器中的 CENx 位置 1，开始进行 DAC 通道 x 校准，
4. 应用调整算法：
   a) 向 OTRIMx[4:0]位写入以 00000b 开头的代码
   b) 等待 $t_{TRIM}$ 延时
   c) 检查 DAC_SR 中的 CAL_FLAGx 位是否已置 1
   d) 如果 CAL_FLAGx 已置 1，调整代码 OTRIMx[4:0]已找到，并将在运行过程中用来补偿输出值，否则会使 OTRIMx[4:0]递增并重复执行（a）到（d）的步骤。

软件算法可使用逐次逼近法或二分法较快地计算和设置 OTRIMx[4:0]位的内容，

CAL_FLAGx 位的换向/切换指示偏移已正确补偿，相应的调整代码必须保留在 DAC_CCR 寄存器的 OTRIMx[4:0]位中。

*注：为了获得正确的值，对 OTRIMx[4:0]位进行的写操作与对 DAC_SR 寄存器中的 CAL_FLAGx 位进行的读操作之间必须保留 $t_{TRIM}$ 延时。此参数在数据手册的电气特性部分进行了说明。*

*如果 $V_{DDA}$、$V_{REF+}$ 和温度条件在器件运行期间不发生更改，同时器件更为频繁地进入待机模式，软件可将第一次用户校准时找到的 OTRIMx[4:0]位存储在 Flash 中。然后在器件电源再次恢复时直接加载/写入这些位，从而无需等待新的校准时间。当 CENx 位置 1 时，不允许将 ENx 位置 1。*

### 16.4.13 DAC 双通道转换模式（如果双通道可用）

为了在同时需要两个 DAC 通道的应用中有效利用总线带宽，DAC 模块有三个双寄存器可供操作：DHR8RD、DHR12RD 和 DHR12LD。这样，只需一个寄存器访问即可同时驱动两个 DAC 通道。要生成相应波形，无需访问 DHRxxxD 寄存器。因此，可单独也可同时使用两个输出通道。

通过两个 DAC 通道和这三个双寄存器可以实现 15 种转换模式。但如果需要，所有这些转换模式也都可以通过单独的 DHRx 寄存器来实现。

下面几段内容将介绍所有这些模式。

**独立触发（不产生波形）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1。
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为不同的值，以配置不同的触发源。
3. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）。

DAC 通道 1 触发信号到达时，DHR1 寄存器内容转移到 DAC_DOR1（三个 dac_pclk 时钟周期后）。

DAC 通道 2 触发信号到达时，DHR2 寄存器内容转移到 DAC_DOR2（三个 dac_pclk 时钟周期后）。

**独立触发（生成单个 LFSR）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1。
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为不同的值，以配置不同的触发源。
3. 将两个 DAC 通道的 WAVEx[1:0]设置为 01，并在 MAMPx[3:0]位中配置相同的 LFSR 掩码值。
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）。

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。LFSR2 计数器随即更新。

**独立触发（生成不同 LFSR）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1。
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为不同的值，以配置不同的触发源。
3. 将两个 DAC 通道的 WAVEx[1:0]设置为 01，并在 MAMP1[3:0]和 MAMP2[3:0]位中设置不同的 LFSR 掩码值。
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）。

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0]配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用 MAMP2[3:0]配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。LFSR2 计数器随即更新。

**独立触发（生成单个三角波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1。
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为不同的值，以配置不同的触发源。
3. 将两个 DAC 通道的 WAVEx[1:0]设置为 1x，并在 MAMPx[3:0]位中配置相同的最大振幅值。
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）。

DAC 通道 1 触发信号到达时，DAC 通道 1 三角波计数器内容（使用相同的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。DAC 通道 1 三角波计数器随即更新。

DAC 通道 2 触发信号到达时，DAC 通道 2 三角波计数器内容（使用相同的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。DAC 通道 2 三角波计数器随即更新。

**独立触发（生成不同三角波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1。
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为不同的值，以配置不同的触发源。
3. 将两个 DAC 通道的 WAVEx[1:0]设置为 1x，并在 MAMP1[3:0]和 MAMP2[3:0]位中设置不同的最大振幅值。
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）。

DAC 通道 1 触发信号到达时，DAC 通道 1 三角波计数器内容（使用 MAMP1[3:0]配置的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。DAC 通道 1 三角波计数器随即更新。

DAC 通道 2 触发信号到达时，DAC 通道 2 三角波计数器内容（使用 MAMP2[3:0]配置的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。DAC 通道 2 三角波计数器随即更新。

**独立触发（生成单个锯齿波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将 STRSTTRIGSEL1[3:0]、STRSTTRIGSEL2[3:0]、STINCTRIGSEL1[3:0]和 STINCTRIGSEL2[3:0]设置为不同的值，以配置不同的触发源。
2. 将两个 DAC 通道的 WAVEx[1:0]设置为 11，并为每个寄存器设置相同的 STRSTDATAx[11:0]、STINCDATAx[15:0]和 STDIRx 值。

DAC 通道 1 触发信号到达时，DAC 通道 1 锯齿波计数器内容更新到 DHR1 寄存器，并将其转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。

DAC 通道 2 触发信号到达时，DAC 通道 2 锯齿波计数器内容更新到 DHR2 寄存器，并将其转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。

**独立触发（生成不同锯齿波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将 STRSTTRIGSEL1[3:0]、STRSTTRIGSEL2[3:0]、STINCTRIGSEL1[3:0]和 STINCTRIGSEL2[3:0]设置为不同的值，以配置不同的触发源。
2. 将两个 DAC 通道的 WAVEx[1:0]设置为 11，并为每个寄存器设置不同的 STRSTDATAx[11:0]、STINCDATAx[15:0]和 STDIRx 值。

DAC 通道 1 触发信号到达时，DAC 通道 1 锯齿波计数器内容更新到 DHR1 寄存器，并将其转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。

DAC 通道 2 触发信号到达时，DAC 通道 2 锯齿波计数器内容更新到 DHR2 寄存器，并将其转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。

**同步软件启动**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

在此配置中，DHR1 和 DHR2 寄存器内容会在一个 dac_hclk 时钟周期后分别转移到 DAC_DOR1 和 DAC_DOR2 中。

**同步触发（不产生波形）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[3:0]和 TSEL2[3:0]设置为相同的值，以便为两个 DAC 通道配置相同的触发源
3. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

当触发信号到达时，DHR1 和 DHR2 寄存器内容将分别转移到 DAC_DOR1 和 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。

**同步触发（生成单个 LFSR）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[3:0]和 TSEL2[3:0]设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0]设置为 01，并在 MAMPx[3:0]位中配置相同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。LFSR1 计数器随即更新。同时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。LFSR2 计数器随即更新。

**同步触发（生成不同 LFSR）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[3:0]和 TSEL2[3:0]设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0]设置为 01，并在 MAMP1[3:0]和 MAMP2[3:0]位中设置不同的 LFSR 掩码值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0]配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。LFSR1 计数器随即更新。

同时，LFSR2 计数器内容（使用 MAMP2[3:0]配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。LFSR2 计数器随即更新。

**同步触发（生成单个三角波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[3:0]和 TSEL2[3:0]设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0]设置为 1x，并在 MAMPx[3:0]位中配置相同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，DAC 通道 1 三角波计数器内容（使用相同的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 dac_hclk 时钟周期之后）。DAC 通道 1 三角波计数器随即更新。

同时，DAC 通道 2 三角波计数器内容（使用相同的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。DAC 通道 2 三角波计数器随即更新。

**同步触发（生成不同三角波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

- 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
- 将 TSEL1[3:0]和 TSEL2[3:0]设置为相同的值，以便为两个 DAC 通道配置相同的触发源
- 将两个 DAC 通道的 WAVEx[1:0]设置为 1x，并在 MAMP1[3:0]和 MAMP2[3:0]位中设置不同的最大振幅值
- 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，DAC 通道 1 三角波计数器内容（使用 MAMP1[3:0]配置的三角波振幅）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB 时钟周期之后）。DAC 通道 1 三角波计数器随即更新。

同时，DAC 通道 2 三角波计数器内容（使用 MAMP2[3:0]配置的三角波振幅）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。DAC 通道 2 三角波计数器随即更新。

**同步触发（生成单个锯齿波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将 STRSTTRIGSEL1[3:0]、STRSTTRIGSEL2[3:0]、STINCTRIGSEL1[3:0]和 STINCTRIGSEL2[3:0]设置为相同的值，为两个 DAC 通道配置相同的触发源。
2. 将两个 DAC 通道的 WAVEx[1:0]设置为 11，并为每个寄存器设置相同的 STRSTDATAx[11:0]、STINCDATAx[15:0]和 STDIRx 值。

触发信号到达时，DAC 通道 1/2 锯齿波计数器内容更新到 DHR1 和 DHR2 寄存器，并将其转移到 DAC_DOR1 和 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。

**同步触发（生成不同锯齿波）**

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将 STRSTTRIGSEL1[3:0]、STRSTTRIGSEL2[3:0]、STINCTRIGSEL1[3:0]和 STINCTRIGSEL2[3:0]设置为不同的值，为两个 DAC 通道配置相同的触发源。
2. 将两个 DAC 通道的 WAVEx[1:0]设置为 11，并为每个寄存器设置不同的 STRSTDATAx[11:0]、STINCDATAx[15:0]和 STDIRx 值。

触发信号到达时，DAC 通道 1/2 锯齿波计数器内容更新到 DHR1 和 DHR2 寄存器，并将其转移到 DAC_DOR1 和 DAC_DOR2 中（三个 dac_hclk 时钟周期之后）。

## 16.5 DAC 低功耗模式

表 16.11 低功耗模式对 DAC 的影响

| 模式 | 说明 |
|---|---|
| 睡眠 | 无影响，DAC 可与 DMA 结合使用。 |
| 停止 0 /停止 1 | DAC 工作被停止并保留其寄存器内容。 |
| 待机 | DAC 外设掉电，退出待机模式后必须重新初始化。 |

## 16.6 DAC 中断

表 16.12 DAC 中断

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机模式退出 | 从待机模式退出 |
|---|---|---|---|---|---|---|---|
| DAC | DMA 下溢 | DMAUDRx | DMAUDRIEx | DMAUDRIEx 写 1 | 是 | 否 | 否 |

## 16.7 DAC 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器必须按字（32 位）进行访问。

### 16.7.1 DAC 控制寄存器（DAC_CR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOFF2 | CEN2 | DMA UDRIE2 | DMA EN2 | MAMP2[3:0] | | | | WAVE2[1:0] | | TSEL2[3:0] | | | | TEN2 | EN2 |
| 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 |
| BOFF1 | CEN1 | DMA UDRIE1 | DMA EN1 | MAMP1[3:0] | | | | WAVE1[1:0] | | TSEL1[3:0] | | | | TEN1 | EN1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **BOFF2：**禁止 DAC 通道 2 输出缓冲器

此位由软件置 1 和清零，用于使能/禁止 DAC 通道 2 的输出缓冲器。

0：使能 DAC 通道 2 输出缓冲器

1：禁止 DAC 通道 2 输出缓冲器

*注：DAC2 通道 2 和 DAC3 通道 2 走内部，无输出缓冲器。即使使能此位，也不会生效。*

*注：输出缓冲器使能时只能对外输出至 IO。输出缓冲器禁止时只能走内部通道给 COMP，内外部通道无法同时使用。*

Bit 30 **CEN2：**DAC 通道 2 校准使能

此位由软件置 1 和清零，用于使能/禁止 DAC 通道 2 校准，仅当 DAC_CR 中的位 EN2=0 时才可写入（仅当 DAC 通道已禁止时才可以进入/退出校准模式），否则会忽略写操作。

0：DAC 通道 2 处于正常运行模式

1：DAC 通道 2 处于校准模式

*注：DAC2 通道 2 和 DAC3 通道 2 走内部，无校准。*

Bit 29 **DMAUDRIE2：**DAC 通道 2 DMA 下溢中断使能

此位由软件置 1 和清零。

0：禁止 DAC 通道 2 DMA 下溢中断

1：使能 DAC 通道 2 DMA 下溢中断

Bit 28 **DMAEN2：**DAC 通道 2 DMA 使能

此位由软件置 1 和清零。

0：关闭 DAC 通道 2 DMA 模式；

1：使能 DAC 通道 2 DMA 模式。

Bit 27:24 **MAMP2[3:0]：**DAC 通道 2 掩码/振幅选择器

这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。

0000：不屏蔽 LSFR 的位 0 /三角波振幅等于 1

0001：不屏蔽 LSFR 的位[1:0] /三角波振幅等于 3

0010：不屏蔽 LSFR 的位[2:0] /三角波振幅等于 7

0011：不屏蔽 LSFR 的位[3:0] /三角波振幅等于 15

0100：不屏蔽 LSFR 的位[4:0] /三角波振幅等于 31

0101：不屏蔽 LSFR 的位[5:0] /三角波振幅等于 63

0110：不屏蔽 LSFR 的位[6:0] /三角波振幅等于 127

0111：不屏蔽 LSFR 的位[7:0] /三角波振幅等于 255

1000：不屏蔽 LSFR 的位[8:0] /三角波振幅等于 511

1001：不屏蔽 LSFR 的位[9:0] /三角波振幅等于 1023

1010：不屏蔽 LSFR 的位[10:0] /三角波振幅等于 2047

≥1011：不屏蔽 LSFR 的位[11:0] /三角波振幅等于 4095

Bit 23:22 **WAVE2[1:0]：**DAC 通道 2 噪声/三角波/锯齿波生成使能

这些位由软件置 1 或清零。

00：禁止生成波

01：使能生成噪声波

10：使能生成三角波

11：使能生成锯齿波

*注：只在位* TEN2 = 1 *（使能* DAC *通道*2 *触发）时使用。*

Bit 21:18 **TSEL2[3:0]：**DAC 通道 2 触发器选择

这些位用于选择 DAC 通道 2 的外部触发事件

0000：SWTRIG2

0001：dac_inc_ch2_trg1

0010：dac_inc_ch2_trg2

…

1110：dac_inc_ch2_trg14

有关触发配置和映射的详细信息，请参见触发选择表。

*注：只在位* TEN2 = 1 *（使能* DAC *通道*2 *触发）时使用。*

Bit 17 **TEN2：**DAC 通道 2 触发使能

此位由软件置 1 和清零，以使能/禁止 DAC 通道 2 触发

0：禁止 DAC 通道 2 触发，写入 DAC_DHR2 寄存器的数据在一个 dac_hclk 时钟周期之后转移到 DAC_DOR2 寄存器

1：使能 DAC 通道 2 触发，DAC_DHR2 寄存器的数据在三个 dac_hclk 时钟周期之后转移到 DAC_DOR2 寄存器

*注：如果选择软件触发，*DAC_DHR2 *寄存器的内容只需一个*dac_hclk *时钟周期即可转移到* DAC_DOR2 *寄存器。*

Bit 16 **EN2：**DAC 通道 2 使能

此位由软件置 1 和清零，以使能/禁止 DAC 通道 2。

0：禁止 DAC 通道 2

1：使能 DAC 通道 2

Bit 15 **BOFF1：**DAC 通道 1 输出缓冲器禁止

此位由软件置 1 和清零，用于使能/禁止 DAC 通道 1 的输出缓冲器。

0：使能 DAC 通道 1 输出缓冲器

1：禁止 DAC 通道 1 输出缓冲器

*注：DAC3 通道1 走内部，无输出缓冲器。即使使能此位，也不会生效。*

*注：输出缓冲器使能时只能对外输出至IO。输出缓冲器禁止时只能走内部通道给COMP，内外部通道无法同时使用。*

Bit 14 **CEN1：**DAC 通道 1 校准使能

此位由软件置 1 和清零，用于使能/禁止 DAC 通道 1 校准，仅当 DAC_CR 中的位 EN1=0 时才可写入（仅当 DAC 通道已禁止时才可以进入/退出校准模式），否则会忽略写操作。

0：DAC 通道 1 处于正常运行模式

1：DAC 通道 1 处于校准模式

*注：（DAC3 通道1 走内部，无校准。）*

Bit 13 **DMAUDRIE1：**DAC 通道 1 DMA 下溢中断使能

此位由软件置 1 和清零。

0：DAC 通道 1 DMA 下溢中断使能

1：DAC 通道 1 DMA 下溢中断使能

Bit 12 **DMAEN1：**DAC 通道 1 DMA 使能

此位由软件置 1 和清零。

0：禁止 DAC 通道 1 DMA 模式

1：使能 DAC 通道 1 DMA 模式

Bit 11:8 **MAMP1[3:0]：**DAC 通道 1 掩码/振幅选择器

这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。

0000：不屏蔽 LSFR 的位 0 /三角波振幅等于 1

0001：不屏蔽 LSFR 的位[1:0] /三角波振幅等于 3

0010：不屏蔽 LSFR 的位[2:0] /三角波振幅等于 7

0011：不屏蔽 LSFR 的位[3:0] /三角波振幅等于 15

0100：不屏蔽 LSFR 的位[4:0] /三角波振幅等于 31

0101：不屏蔽 LSFR 的位[5:0] /三角波振幅等于 63

0110：不屏蔽 LSFR 的位[6:0] /三角波振幅等于 127

0111：不屏蔽 LSFR 的位[7:0] /三角波振幅等于 255

1000：不屏蔽 LSFR 的位[8:0] /三角波振幅等于 511

1001：不屏蔽 LSFR 的位[9:0] /三角波振幅等于 1023

1010：不屏蔽 LSFR 的位[10:0] /三角波振幅等于 2047

≥1011：不屏蔽 LSFR 的位[11:0] /三角波振幅等于 4095

Bit 7:6 **WAVE1[1:0]：**DAC 通道 1 噪声/三角波/锯齿波生成使能

这些位将由软件置 1 和清零。

00：禁止生成波

01：使能生成噪声波

10：使能生成三角波

11：使能生成锯齿波

*注：只在位* TEN1 = 1 *（使能* DAC *通道*1 *触发）时使用。*

Bit 5:2 **TSEL1[3:0]：** DAC 通道 1 触发器选择

这些位用于选择 DAC 通道 1 的外部触发事件

0000：SWTRIG1

0001：dac_ch1_trg1

0010：dac_ch1_trg2

…

1111：dac_ch1_trg15

有关触发配置和映射的详细信息，请参见触发选择表。

*注：只在位* TEN1 = 1 *（使能* DAC *通道*1 *触发）时使用。*

Bit 1 **TEN1：** DAC 通道 1 触发使能

此位由软件置 1 和清零，以使能/禁止 DAC 通道 1 触发

0：禁止 DAC 通道 1 触发，写入 DAC_DHR1 寄存器的数据在一个 dac_hclk 时钟周期之后转移到 DAC_DOR1 寄存器

1：使能 DAC 通道 1 触发，DAC_DHR1 寄存器的数据在三个 dac_hclk 时钟周期之后转移到 DAC_DOR1 寄存器

*注：如果选择软件触发，*DAC_DHR1 *寄存器的内容只需一个*dac_hclk *时钟周期即可转移到* DAC_DOR1 *寄存器。*

Bit 0 **EN1：** DAC 通道 1 使能

此位由软件置 1 和清零，以使能/禁止 DAC 通道 1。

0：禁止 DAC 通道 1

1：使能 DAC 通道 1

### 16.7.2 DAC 软件触发寄存器（DAC_SWTRIGR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SWTRIGB2 | SWTRIGB1 |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SWTRIG2 | SWTRIG1 |
| | | | | | | | | | | | | | | rw | rw |

Bit 31:18 保留，必须保持为复位值。

Bit 17 **SWTRIGB2：** DAC 通道 2 软件触发 B

该位由软件置 1，用于在软件触发模式（锯齿波生成）下触发 DAC。由硬件清除。

0：不触发

1：触发锯齿波增量

Bit 16 **SWTRIGB1：** DAC 通道 1 软件触发

该位由软件置 1，用于在软件触发模式（锯齿波生成）下触发 DAC。由硬件清除。

0：不触发
1：触发锯齿波增量

Bit 15:2 保留，必须保持为复位值。

Bit 1 **SWTRIG2**：DAC 通道 2 软件触发
该位由软件置 1，用于在软件触发模式下触发 DAC。
0：不触发
1：触发。
*注：一旦* DAC_DHR2 *寄存器值加载到* DAC_DOR2 *寄存器中，该位即会由硬件清零（一个* dac_hclk *时钟周期之后）。*

Bit 0 **SWTRIG1**：DAC 通道 1 软件触发
该位由软件置 1，用于在软件触发模式下触发 DAC。
0：不触发
1：触发。
*注：一旦* DAC_DHR1 *寄存器值加载到* DAC_DOR1 *寄存器中，该位即会由硬件清零（一个* dac_hclk *时钟周期之后）。*

### 16.7.3 DAC 通道 1 12 位右对齐数据保持寄存器（DAC_DHR12R1）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DACC1DHR[11:0] | | | | | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **DACC1DHR[11:0]**：DAC 通道 1 的 12 位右对齐数据
这些位通过软件写入，用于指定 DAC 通道 1 的 12 位数据。

### 16.7.4 DAC 通道 1 12 位左对齐数据保持寄存器（DAC_DHR12L1）

偏移地址：0x0C
复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DACC1DHR[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | |

Bit 31:16 保留，必须保持为复位值。

Bit 15:4 **DACC1DHR[11:0]**：DAC 通道 1 的 12 位左对齐数据
这些位通过软件写入，用于指定 DAC 通道 1 的 12 位数据。

Bit 3:0 保留，必须保持为复位值。

### 16.7.5 DAC 通道 1 8 位右对齐数据保持寄存器（DAC_DHR8R1）

偏移地址：0x10

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | DACC1DHR[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:8 保留，必须保持为复位值。

Bit 7:0 **DACC1DHR[7:0]：**DAC 通道 1 的 8 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 1 的 8 位数据。

### 16.7.6 DAC 通道 2 12 位右对齐数据保持寄存器（DAC_DHR12R2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DACC2DHR[11:0] | | | | | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **DACC2DHR[11:0]：**DAC 通道 2 的 12 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 12 位数据。

### 16.7.7 DAC 通道 2 12 位左对齐数据保持寄存器（DAC_DHR12L2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DACC2DHR[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | |

Bit 31:16 保留，必须保持为复位值。

Bit 15:4 **DACC2DHR[11:0]：**DAC 通道 2 的 12 位左对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 12 位数据。

Bit 3:0 保留，必须保持为复位值。

### 16.7.8 DAC 通道 2 8 位右对齐数据保持寄存器（DAC_DHR8R2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | DACC2DHR[7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:8 保留，必须保持为复位值。

Bit 7:0 **DACC2DHR[7:0]：**DAC 通道 2 的 8 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 8 位数据。

### 16.7.9 双 DAC 12 位右对齐数据保持寄存器（DAC_DHR12RD）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DACC1DHR[11:0] | | | | | | | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:28 保留，必须保持为复位值。

Bit 27:16 **DACC2DHR[11:0]：**DAC 通道 2 的 12 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 12 位数据。

Bit 15:12 保留，必须保持为复位值。

Bit 11:0 **DACC1DHR[11:0]：**DAC 通道 1 的 12 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 1 的 12 位数据。

### 16.7.10 双 DAC 12 位左对齐数据保持寄存器（DAC_DHR12LD）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DACC2DHR[11:0] | | | | | | | | | | | | Reserved | | | |
| 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] | | | | | | | | | | | | Reserved | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | |

Bit 31:20 **DACC2DHR[11:0]：**DAC 通道 2 的 12 位左对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 12 位数据。

Bit 19:16 保留，必须保持为复位值。

Bit 15:4 **DACC1DHR[11:0]：**DAC 通道 1 的 12 位左对齐数据

这些位通过软件写入，用于指定 DAC 通道 1 的 12 位数据。

Bit 3:0 保留，必须保持为复位值。

### 16.7.11 双 DAC 8 位右对齐数据保持寄存器（DAC_DHR8RD）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 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 31:16 保留，必须保持为复位值。

Bit 15:8 **DACC2DHR[7:0]：**DAC 通道 2 的 8 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 2 的 8 位数据。

Bit 7:0 **DACC1DHR[7:0]：**DAC 通道 1 的 8 位右对齐数据

这些位通过软件写入，用于指定 DAC 通道 1 的 8 位数据。

### 16.7.12 DAC 通道 1 数据输出寄存器（DAC_DOR1）

偏移地址：0x2C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | DACC1DOR[11:0] | | | | | | | | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **DACC1DOR[11:0]：**DAC 通道 1 数据输出

这些位为只读，其中包含 DAC 通道 1 的数据输出。

### 16.7.13 DAC 通道 2 数据输出寄存器（DAC_DOR2）

偏移地址：0x30

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DACC2DOR[11:0] | | | | | | | | | | | |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **DACC2DOR[11:0]：**DAC 通道 2 数据输出

这些位为只读，其中包含 DAC 通道 2 的数据输出。

### 16.7.14 DAC 状态寄存器（DAC_SR）

偏移地址：0x34

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | CAL_FLAG2 | DMAUDR2 | Reserved | | | | | | | | | | | | |
| | r | rc_w1 | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | CAL_FLAG1 | DMAUDR1 | Reserved | | | | | | | | | | | | |
| | r | rc_w1 | | | | | | | | | | | | | |

Bit 31 保留，必须保持为复位值。

Bit 30 **CAL_FLAG2：**DAC 通道 2 校准偏移状态

此位通过硬件置 1 和清零。

0：校准调整值小于偏移校正值

1：校准调整值等于或大于偏移校正值

*注：（DAC2 通道 2，DAC3 通道 2 走内部，无该标志）*

Bit 29 **DMAUDR2：**DAC 通道 2 DMA 下溢标志

此位由硬件置 1，由软件清零（写入 1）。

0：DAC 通道 2 未发生 DMA 下溢错误状况

1：DAC 通道 2 发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动）

Bit 28:15 保留，必须保持为复位值。

Bit 14 **CAL_FLAG1：**DAC 通道 1 校准偏移状态

此位通过硬件置 1 和清零。

0：校准调整值小于偏移校正值

1：校准调整值等于或大于偏移校正值

*注：（DAC3 通道一走内部，无该标志）*

Bit 13 **DMAUDR1：**DAC 通道 1 DMA 下溢标志

此位由硬件置 1，由软件清零（写入 1）。

0：DAC 通道 1 未发生 DMA 下溢错误状况
1：DAC 通道 1 发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动）

Bit 12:0 保留，必须保持为复位值。

### 16.7.15 DAC 校准寄存器（DAC_CCR）

偏移地址：0x38
复位值：0x0011 0011

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | OTRIM2[4:0] | | | | |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | OTRIM1[4:0] | | | | |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

Bit 31:21 保留，必须保持为复位值。

Bit 20:16 **OTRIM2[4:0]：**DAC 通道 2 偏移调整值
*注：（DAC2 通道 2，DAC3 通道 2 走内部，无该值）*

Bit 15:5 保留，必须保持为复位值。

Bit 4:0 **OTRIM1[4:0]：**DAC 通道 1 偏移调整值
*注：（DAC3 通道 1 走内部，无该值）*

### 16.7.16 DAC 模式控制寄存器（DAC_MCR）

偏移地址：0x3C
复位值：0x0000 0000

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

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:14 **HFSEL[1:0]：**高频接口模式选择
00：禁止高频接口模式，DAC_DOR 更新速率为 3 个 AHB 时钟周期
01：使能与 AHB >80 MHz 兼容的高频接口模式，使 DAC_DOR 更新速率达到 5 个 AHB 时钟周期
10：使能与 AHB >160 MHz 兼容的高频接口模式，使 DAC_DOR 更新速率达到 7 个 AHB 时钟周期
11：保留

Bit 13:0 保留，必须保持为复位值。

### 16.7.17 DAC 通道 1 锯齿波寄存器（DAC_STR1）

偏移地址：0x58

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STINCDATA1[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | STDIR1 | STRSTDATA1[11:0] | | | | | | | | | | | |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 **STINCDATA1[15:0]**：DAC 通道 1 锯齿波增量值

Bit 15:13 保留，必须保持为复位值。

Bit 12 **STDIR1**：DAC 通道 1 锯齿波方向设置

此位由软件写入，以选择锯齿波步进方向

0：递减

1：递增

Bit 11:0 **STRSTDATA1[11:0]**：DAC 通道 1 锯齿波复位值

### 16.7.18 DAC 通道 2 锯齿寄存器（DAC_STR2）

偏移地址：0x5C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STINCDATA2[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | STDIR2 | STRSTDATA2[11:0] | | | | | | | | | | | |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 **STINCDATA2[15:0]**：DAC 通道 2 锯齿波增量值

Bit 15:13 保留，必须保持为复位值。

Bit 12 **STDIR2**：DAC 通道 2 锯齿波方向设置

此位由软件写入，以选择锯齿波步进方向

0：递减

1：递增

Bit 11:0 **STRSTDATA2[11:0]**：DAC 通道 2 锯齿波复位值

### 16.7.19 DAC 锯齿波模式寄存器（DAC_STMODR）

偏移地址：0x60

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | STINCTRIGSEL2[3:0] | | | | Reserved | | | | STRSTTRIGSEL2[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | STINCTRIGSEL1[3:0] | | | | Reserved | | | | STRSTTRIGSEL1[3:0] | | | |
| | | | | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bit 31:28 保留，必须保持为复位值。

Bit 27:24 **STINCTRIGSEL2[3:0]：**DAC 通道 2 锯齿波增量触发选择

有关触发配置和映射的详细信息，请参见触发选择表。

0000：SWTRIGB2

0001：dac _inc_ch2_trg1

...

1111：dac _inc_ch2_trg15

Bit 23:20 保留，必须保持为复位值。

Bit 19:16 **STRSTTRIGSEL2[3:0]：**DAC 通道 2 锯齿波复位触发选择

有关触发配置和映射的详细信息，请参见触发选择表。

0000：SWTRIGB2

0001：dac_ch2_trg1

...

1111：dac_ch2_trg15

映射与 TSEL2[3:0]相同。

Bit 15:12 保留，必须保持为复位值。

Bit 11:8 **STINCTRIGSEL1[3:0]：**DAC 通道 1 锯齿波增量触发选择

有关触发配置和映射的详细信息，请参见触发选择表。

0000：SWTRIGB1

0001：dac_inc_ch1_trg1

...

1111：dac_inc_ch1_trg15

Bit 7:4 保留，必须保持为复位值。

Bit 3:0 **STRSTTRIGSEL1[3:0]：**DAC 通道 1 锯齿波复位触发选择

有关触发配置和映射的详细信息，请参见触发选择表。

0000：SWTRIGB1

0001：dac_ch1_trg1

...

1111：dac_ch1_trg15

映射与 TSEL1[3:0]相同。

# 17 直接存储器访问控制器（DMA）

## 17.1 简介

直接存储器访问（DMA）用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据。这样节省的 CPU 资源可供其它操作使用。

DMA 控制器采用单个 AHB 主架构。

DMA 有两个实例：DMA1 和 DMA2（请参阅下表：DMA1 和 DMA2 支持的通道数量的实现）。

每个通道都专门用于管理来自一个或多个外设的存储器访问请求。每个 DMA 都包含一个仲裁器，用于处理 DMA 请求之间的优先级。

## 17.2 DMA 主要特征

- 单 AHB 主设备
- 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数据传输
- 以源和目标身份访问片上存储器映射设备，如闪存、SRAM 以及 AHB 和 APB 外设
- 所有 DMA 通道均可独立配置：
  - 每个通道都与来自外设的 DMA 请求信号相关联，或与存储器到存储器传输中的软件触发相关联。此配置由软件完成。
  - 请求之间的优先级可通过软件编程（每个通道有四个级别：非常高、高、中、低），在优先级相等的情况下也可通过硬件确定（例如，对通道 1 的请求比对通道 2 的请求具有优先权）。
  - 源和目标传输大小独立（字节、半字、字），模拟打包和解包。源地址和目标地址必须根据数据大小对齐。
  - 支持使用循环缓冲区管理从/向外设到/从存储器的传输
  - 可编程的数据传输数量：0 到 $2^{16}$ - 1
- 每个通道生成中断请求。每个中断请求由以下三个 DMA 事件中的任意一个引起：传输完成、半传输或传输错误。

## 17.3 DMA 实现

### 17.3.1 DMA1 和 DMA2

DMA1 和 DMA2 的实现采用了下表所示的硬件配置参数。

表 17.1 DMA 实现

| 特征 | DMA1 | DMA2 |
|---|---|---|
| 数量通道 | 8 | 8 |

### 17.3.2 DMA 请求映射

DMA 控制器通过 DMAMUX 外设与来自 AHB/APB 外设的 DMA 请求相连。
有关不同请求的映射，请参阅 DMAMUX 部分。

## 17.4 DMA 功能描述

### 17.4.1 DMA 框图

DMA 框图如下图所示。

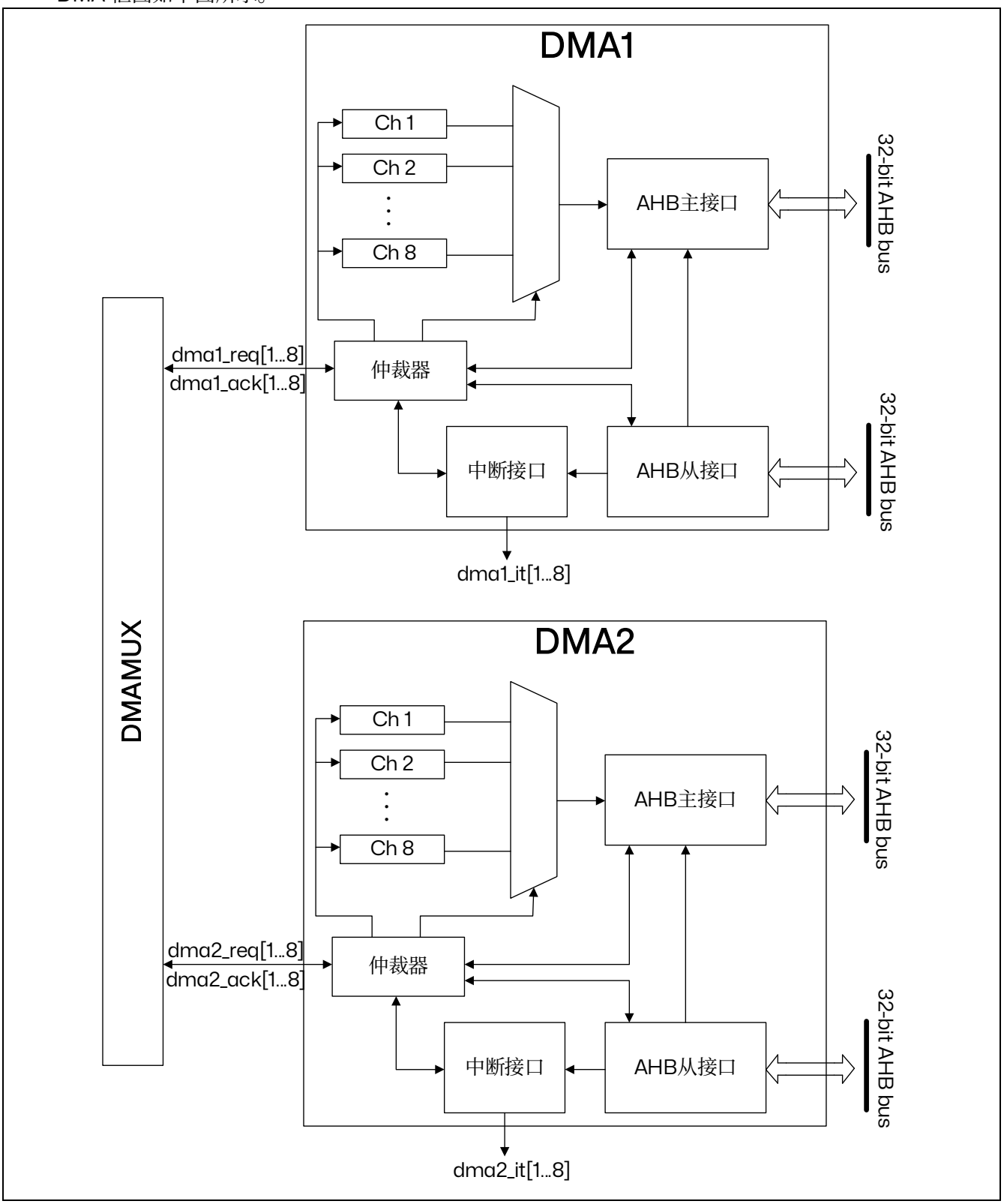



图 17.1 DMA 框图

*注：请参阅上图：DMA 框图，了解功能实现。*

DMA 控制器通过与其他系统主设备共享 AHB 系统总线来执行直接存储器传输。总线矩阵采用轮询调度。当 CPU 和 DMA 指向同一目标（存储器或外设）时，DMA 请求可能会阻止 CPU 访问系统总线几个总线周期。

根据通过 AHB 从接口的配置，DMA 控制器在 DMA 通道及其相关接收请求之间进行仲裁。DMA 控制器还通过单个 AHB 端口主设备调度 DMA 数据传输。

DMA 控制器为每个通道向中断控制器生成一个中断。

### 17.4.2 DMA 引脚和内部信号

表 17.2 DMA 内部输入输出信号

| 信号名称 | 信号类型 | 描述 |
|---|---|---|
| dma_req[x] | 输入 | 通道 x 请求 |
| dma_ack[x] | 输出 | 通道 x 应答 |
| dma_it[x] | 输出 | 通道 x 中断 |

### 17.4.3 DMA 传输

软件在通道级别配置 DMA 控制器，以执行由一系列 AHB 总线传输组成的块传输。

可以从外设请求 DMA 块传输，或者在内存到内存的传输情况下由软件触发。

在事件发生后，单个 DMA 传输的后续步骤如下：

1. 外设向 DMA 控制器发送单个 DMA 请求信号。
2. DMA 控制器根据与此外设请求相关联的通道的优先级来处理请求。
3. 一旦 DMA 控制器授予外设权限，DMA 控制器就会向外设发送确认信号。
4. 外设在从 DMA 控制器收到确认后，立即释放其请求。
5. 一旦外设取消请求，DMA 控制器就会释放确认信号。

外设可以发出另一个单请求并启动另一个 DMA 单次传输。

当外设是传输的源或目标时，会使用请求/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发送到 DMA 控制器来启动传输。然后，DMA 控制器从存储器中的某个位置（通过内部当前外设/存储器地址寄存器寻址）读取单个数据，并将该数据写入外设。

对于给定通道 x，DMA 块传输由以下重复序列组成：

- 单个 DMA 传输，在 DMA AHB 总线主设备上封装两个单个数据的 AHB 传输：
  - 从外设数据寄存器或存储器中的某个位置（通过内部当前外设/存储器地址寄存器寻址）读取单个数据（字节、半字或字）。
    用于第一次单次传输的起始地址是外设或存储器的基地址，并在 DMA_CPARx 或 DMA_CMARx 寄存器中编程。
  - 向外设数据寄存器或存储器中的某个位置（通过内部当前外设/存储器地址寄存器寻址）写入单个数据（字节、半字或字）。
    用于第一次传输的起始地址是外设或存储器的基地址，并在 DMA_CPARx 或 DMA_CMARx 寄存器中编程。
- 对已编程的 DMA_CNDTRx 寄存器进行后递减

- 该寄存器包含要传输的数据项剩余数量（AHB“读取后跟写入”传输的数量）。此序列会一直重复，直到 DMA_CNDTRx 为零。

*注：AHB 主总线源/目标地址必须与传输到源/目标的单个数据的编程大小对齐。*

### 17.4.4 DMA 仲裁器

DMA 仲裁器管理不同通道之间的优先级。

当仲裁器（硬件请求或软件触发）授予活动通道 x 时，会发出一次 DMA 传输（例如，AHB 的“先读后写”单数据传输）。然后，仲裁器会再次考虑活动通道集合，并选择优先级最高的通道。

优先级的管理分为两个阶段：

- 软件：在 DMA_CCRx 寄存器中，将每个通道的优先级配置为四个不同等级之一：
  - 最高
  - 高
  - 中
  - 低
- 硬件：如果两个请求的软件优先级相同，则编号最低的通道获得优先权。例如，通道 2 的优先级高于通道 4。

当通道 x 被编程为存储器到存储器模式的块传输时，在该通道 x 的每次 DMA 传输之间都会进行再次仲裁。每当有另一个并发活动的请求通道时，DMA 仲裁器会自动切换并授予另一个优先级最高的请求通道，该通道的优先级可能低于存储器到存储器通道。

### 17.4.5 DMA 通道

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

DMA 通道是在块传输级别上进行编程的。

**可编程的数据量**

通过 DMA_CCRx 寄存器的 PSIZE[1:0]和 MSIZE[1:0]位域，可以分别对外设和存储器中的单个数据（字节、半字或字）的传输数据量进行编程。

**指针增量**

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

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

*注：如果通道 x 被禁用，DMA 寄存器不会重置。DMA 通道寄存器（DMA_CCRx、DMA_CPARx 和 DMA_CMARx）保留在通道配置阶段编程的初始值。*

在**循环模式**下，最后一次传输结束时，DMA_CNDTRx 寄存器的内容会自动地被重新加载为其初始数值，内部的当前外设/存储器地址寄存器也被重新加载为 DMA_CPARx 和 DMA_CMARx 寄存器设定的初

始基地址。

**通道配置过程**

下面是配置 DMA 通道 x 的过程，需要按照以下顺序操作：

1. 在 DMA_CPARx 寄存器中设置外设寄存器的地址。
   发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在 DMA_CMARx 寄存器中设置数据存储器的地址。
   发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
3. 在 DMA_CNDTRx 寄存器中设置要传输的数据量。
   在每个数据传输后，这个数值递减。
4. 在 DMA_CCRx 寄存器中配置以下参数：
   – 通道优先级
   – 数据传输方向
   – 循环模式
   – 外设和存储器递增模式
   – 外设和存储器数据大小
   – 半传输和/或全传输和/或传输错误时的中断使能
5. 设置 DMA_CCRx 寄存器的 EN 位，启动该通道。

通道一旦使能，就可以处理连接到该通道的外设发出的任何 DMA 请求，或者可以启动存储器到存储器的块传输。

*注：通道配置程序的最后两个步骤可以合并成对* DMA_CCRx *寄存器的单次访问，以配置和使能通道。*

**通道状态和禁用通道**

处于活动状态的通道 x 是一个已使能的通道（读取 DMA_CCRx.EN = 1）。活动通道 x 必须是已由软件使能（DMA_CCRx.EN 设置为 1）且之后未发生传输错误（DMA_ISR.TEIFx = 0）的通道。如果发生传输错误，硬件会自动禁用该通道（DMA_CCRx.EN = 0）。

以下三种用例可能发生：

- 暂停和恢复通道
  这对应于以下两个操作：
  – 软件禁用活动通道（在 DMA_CCRx.EN = 1 时写入 DMA_CCRx.EN = 0）。
  – 软件在不重新配置其他通道寄存器（如 DMA_CNDTRx、DMA_CPARx 和 DMA_CMARx）的情况下再次使能通道（将 DMA_CCRx.EN 设置为 1）。
  DMA 硬件不支持此用例，不保证剩余的数据传输能正确执行。
- 停止和中止通道
  如果应用程序不再需要该通道，软件可以禁用此活动通道。通道会停止并中止，但 DMA_CNDTRx 寄存器的内容可能无法正确反映与已中止的源和目标缓冲区/寄存器相比的剩余数据传输。
- 中止并重新启动通道
  这对应于软件的以下步骤：禁用活动通道，然后重新配置通道并再次使能它。
  如果满足以下条件，硬件支持此操作：
  – 应用程序确保在软件禁用通道时，没有通过其主端口同时进行 DMA 数据传输。例如，应用程序可以先禁用外设的 DMA 模式，以确保没有来自该外设的挂起硬件 DMA 请求。

– 软件必须对相同的 DMA_CCRx 寄存器执行单独的写访问操作：首先禁用通道。其次，如果需要更改配置，则为下一个块传输（包括 DMA_CCRx）重新配置通道。当 DMA_CCRx.EN=1 时，存在只读 DMA_CCRx 寄存器位域。最后，再次使能通道。

当发生通道传输错误时，DMA_CCRx 寄存器的 EN 位由硬件清除。在 DMA_ISR 寄存器的 TEIFx 位被设置之前，软件无法再次设置此 EN 位以重新激活通道 x。

**循环模式（在存储器到外设/外设到存储器传输中）**

循环模式用于处理循环缓冲区和连续的数据传输（如 ADC 的扫描模式）。在 DMA_CCRx 寄存器中的 CIRC 位用于开启这一功能。

*注：在存储器到存储器模式下不得使用循环模式。在使能循环模式（CIRC = 1）的通道之前，软件必须清除 DMA_CCRx 寄存器的 MEM2MEM 位。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成配置通道时设置的初值，DMA 操作将会继续进行。*

为了停止循环传输，软件需要在禁用 DMA 通道之前停止外设生成 DMA 请求（如退出 ADC 扫描模式）。在开始/使能传输之前以及在停止循环传输之后，软件必须明确地为 DMA_CNDTRx 编程值。

**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_CCRx 寄存器中的 MEM2MEM 位之后，在软件设置了 DMA_CCRx 寄存器中的 EN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA_CNDTRx 寄存器变为 0 时，DMA 传输结束。

*注：在循环模式下不得使用存储器到存储器模式。在使能存储器到存储器模式（MEM2MEM = 1）的通道之前，软件必须清除 DMA_CCRx 寄存器的 CIRC 位。*

**外设到外设模式**

任何 DMA 通道都可以在外设到外设模式下运行：

- 当选择外设的硬件请求来触发 DMA 通道时
  此外设是 DMA 发起者，并控制从此外设到属于另一个存储器映射外设的寄存器的数据传输（该外设未配置为 DMA 模式）。
- 当没有选择并连接到 DMA 通道的外设请求时
  软件通过设置 DMA_CCRx 寄存器的 MEM2MEM 位来配置寄存器到寄存器的传输。

**编程传输方向，分配源/目标**

DMA_CCRx 寄存器的 DIR 位的值设置传输方向，并因此确定源和目标，而不考虑源/目标类型（外设或存储器）：

- **DIR = 1** 通常定义存储器到外设的传输。更一般地说，如果 DIR = 1：
  – **源**属性由 DMA_MARx 寄存器、DMA_CCRx 寄存器的 MSIZE[1:0]位域和 MINC 位定义。无论它们的通常命名如何，这些“存储器”寄存器、位域和位都用于在外设到外设模式下定义源外设。
  – **目标**属性由 DMA_PARx 寄存器、DMA_CCRx 寄存器的 PSIZE[1:0]位域和 PINC 位定义。无论它们的通常命名如何，这些“外设”寄存器、位域和位都用于在存储器到存储器模式下定义目标存储器。
- **DIR = 0** 通常定义外设到存储器的传输。更一般地说，如果 DIR = 0：
  – **源**属性由 DMA_PARx 寄存器、DMA_CCRx 寄存器的 PSIZE[1:0]位域和 PINC 位定义。无论

它们的通常命名如何，这些“外设”寄存器、位域和位都用于在存储器到存储器模式下定义源存储器。

– **目标**属性由 DMA_MARx 寄存器、DMA_CCRx 寄存器的 MSIZE[1:0]位域和 MINC 位定义。无论它们通常如何命名，这些“存储器”寄存器、位域和位都用于在外设到外设模式下定义目标外设。

### 17.4.6 DMA 数据宽度、对齐方式和数据大小端

当 PSIZE[1:0]和 MSIZE[1:0]不相等时，DMA 控制器会执行一些数据对齐操作，如下表所述。

表 17.3 可编程的数据传输宽度和大小端操作（当 PINC = MINC = 1）

| 源端宽度（MSIZE 如果 DIR=1，则为 PSIZE） | 目标宽度（PSIZE 如果 DIR=1，则为 MSIZE） | 传输数目（NDT） | 源：地址/数据（DMA_CMARx 如果 DIR=1，则为 DMA_CPARx） | 传输操作 | 目标：地址/数据 |
|---|---|---|---|---|---|
| 8 | 8 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1：在 0x0 读 B0[7:0]，在 0x0 写 B0[7:0]<br>2：在 0x1 读 B1[7:0]，在 0x1 写 B1[7:0]<br>3：在 0x2 读 B2[7:0]，在 0x2 写 B2[7:0]<br>4：在 0x3 读 B3[7:0]，在 0x3 写 B3[7:0] | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 |
| 8 | 16 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1：在 0x0 读 B0[7:0]，在 0x0 写 00B0[15:0]<br>2：在 0x1 读 B1[7:0]，在 0x2 写 00B1[15:0]<br>3：在 0x2 读 B2[7:0]，在 0x4 写 00B2[15:0]<br>4：在 0x3 读 B3[7:0]，在 0x6 写 00B3[15:0] | @0x0 / 00B0<br>@0x2 / 00B1<br>@0x4 / 00B2<br>@0x6 / 00B3 |
| 8 | 32 | 4 | @0x0 / B0<br>@0x1 / B1<br>@0x2 / B2<br>@0x3 / B3 | 1：在 0x0 读 B0[7:0]，在 0x0 写 000000B0[31:0]<br>2：在 0x1 读 B1[7:0]，在 0x4 写 000000B1[31:0]<br>3：在 0x2 读 B2[7:0]，在 0x8 写 000000B2[31:0]<br>4：在 0x3 读 B3[7:0]，在 0xC 写 000000B3[31:0] | @0x0 / 000000B0<br>@0x4 / 000000B1<br>@0x8 / 000000B2<br>@0xC / 000000B3 |
| 16 | 8 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1：在 0x0 读 B1B0[15:0]，在 0x0 写 B0[7:0]<br>2：在 0x2 读 B3B2[15:0]，在 0x1 写 B2[7:0]<br>3：在 0x4 读 B5B4[15:0]，在 0x2 写 B4[7:0]<br>4：在 0x6 读 B7B6[15:0]，在 0x3 写 B6[7:0] | @0x0 / B0<br>@0x1 / B2<br>@0x2 / B4<br>@0x3 / B6 |
| 16 | 16 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1：在 0x0 读 B1B0[15:0]，在 0x0 写 B1B0[15:0]<br>2：在 0x2 读 B3B2[15:0]，在 0x2 写 B3B2[15:0]<br>3：在 0x4 读 B5B4[15:0]，在 0x4 写 B5B4[15:0]<br>4：在 0x6 读 B7B6[15:0]，在 0x6 写 B7B6[15:0] | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 |
| 16 | 32 | 4 | @0x0 / B1B0<br>@0x2 / B3B2<br>@0x4 / B5B4<br>@0x6 / B7B6 | 1：在 0x0 读 B1B0[15:0]，在 0x0 写 0000B1B0[31:0]<br>2：在 0x2 读 B3B2[15:0]，在 0x4 写 0000B3B2[31:0]<br>3：在 0x4 读 B5B4[15:0]，在 0x8 写 0000B5B4[31:0]<br>4：在 0x6 读 B7B6[15:0]，在 0xC 写 0000B7B6[31:0] | @0x0 / 0000B1B0<br>@0x4 / 0000B3B2<br>@0x8 / 0000B5B4<br>@0xC / 0000B7B6 |
| 32 | 8 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0XC / BFBEBDBC | 1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B0[7:0]<br>2：在 0x4 读 B7B6B5B4[31:0]，在 0x1 写 B4[7:0]<br>3：在 0x8 读 BBBAB9B8[31:0]，在 0x2 写 B8[7:0]<br>4：在 0xC 读 BFBEBDBC[31:0]，在 0x3 写 BC[7:0] | @0x0 / B0<br>@0x1 / B4<br>@0x2 / B8<br>@0x3 / BC |
| 32 | 16 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC | 1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B1B0[15:0]<br>2：在 0x4 读 B7B6B5B4[31:0]，在 0x2 写 B5B4[15:0]<br>3：在 0x8 读 BBBAB9B8[31:0]，在 0x4 写 B9B8[15:0]<br>4：在 0xC 读 BFBEBDBC[31:0]，在 0x6 写 BDBC[15:0] | @0x0 / B1B0<br>@0x2 / B5B4<br>@0x4 / B9B8<br>@0x6 / BDBC |
| 32 | 32 | 4 | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC | 1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B3B2B1B0[31:0]<br>2：在 0x4 读 B7B6B5B4[15:0]，在 0x4 写 B7B6B5B4[31:0]<br>3：在 0x8 读 BBBAB9B8[31:0]，在 0x8 写 BBBAB9B8[31:0]<br>4：在 0xC 读 BFBEBDBC[31:0]，在 0xC 写 BFBEBDBC[31:0] | @0x0 / B3B2B1B0<br>@0x4 / B7B6B5B4<br>@0x8 / BBBAB9B8<br>@0xC / BFBEBDBC |

**操作一个不支持字节或半字写的 AHB 设备**

当 DMA 模块开始一个 AHB 的字节或半字写操作时，数据将在 HWDATA[31:0]总线中未使用的部分重复。

因此，如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时（即 HSIZE 不适于该模块），不会发生错误，DMA 将按照下面两个例子写入 32 位 HWDATA 数据：

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

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

- 一个 AHB 上对地址 0x0（或 0x1、0x2 或 0x3）的写字节数据“0xB0”操作，将转换到 APB 上对地址 0x0 的写字数据“0xB0B0B0B0”操作。
- 一个 AHB 上对地址 0x0（或 0x2）的写半字数据“0xB1B0”操作，将转换到 APB 上对地址 0x0 的写字数据“0xB1B0B1B0”操作。例如，如果要写入 APB 备份寄存器（与 32 位地址对齐的 16 位寄存器），需要配置存储器数据源宽度（MSIZE）为 16 位，外设目标数据宽度（PSIZE）为 32 位。

### 17.4.7 DMA 错误管理

读写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时，硬件会自动地清除发生错误的通道 x 所对应的通道配置寄存器 DMA_CCRx 的 EN 位，该通道操作被停止。

DMA_ISR 寄存器中对应该通道的传输错误中断标志位 TEIFx 将被置位，如果在 DMA_CCRx 寄存器中设置了传输错误中断允许位，则将产生中断。

在 DMA_ISR 寄存器的 TEIFx 位被清除（通过置位 DMA_IFCR 寄存器的 CTEIFx 位）之前，软件无法再次置位 DMA_CCRx 寄存器的 EN 位（通道 x 重新激活）。

当软件收到涉及外设的通道传输错误通知时，软件必须首先停止该外设的 DMA 模式，以禁用任何挂起或未来的 DMA 请求。然后，软件可以正常重新配置 DMA 和外设在 DMA 模式下进行新的传输。

### 17.4.8 DMA 中断

每个 DMA 通道 x 可以在半传输、传输完成或传输错误时生成中断。为提供灵活性，提供了单独的中断使能位。

表 17.4 DMA 中断请求

| 中断请求 | 中断事件 | 事件标志 | 中断使能位 |
|---|---|---|---|
| 通道 x 中断 | 通道 x 半传输 | HTIFx | HTIEx |
| | 通道 x 传输完成 | TCIFx | TCIEx |
| | 通道 x 传输错误 | TEIFx | TEIEx |
| | 通道 x 半传输或传输完成或传输错误 | GIFx | - |

## 17.5 DMA 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。

外设寄存器必须按字（32 位）进行访问。

*注：请参阅* DMA *框图以了解功能实现。*

### 17.5.1 DMA 中断状态寄存器（DMA_ISR）

偏移地址：0x00

复位值：0x0000 0000

当在 DMA_IFCR 寄存器中设置相应清除位或相应的全局清除位 CGIFx 时，硬件会清除每个状态位。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TEIF8 | HTIF8 | TCIF8 | GIF8 | TEIF7 | HTIF7 | TCIF7 | GIF7 | TEIF6 | HTIF6 | TCIF6 | GIF6 | TEIF5 | HTIF5 | TCIF5 | GIF5 |
| 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 |
| TEIF4 | HTIF4 | TCIF4 | GIF4 | TEIF3 | HTIF3 | TCIF3 | GIF3 | TEIF2 | HTIF2 | TCIF2 | GIF2 | TEIF1 | HTIF1 | TCIF1 | GIF1 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 4*(x-1)+3 **TEIFx：**通道 x 的传输错误标志（x = 1...8）

硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入 1 可以清除这里对应的标志位。

0：在通道 x 没有传输错误（TE）；

1：在通道 x 发生了传输错误（TE）。

Bit 4*(x-1)+2 **HTIFx：**通道 x 的半传输标志（x = 1...8）

硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入 1 可以清除这里对应的标志位。

0：在通道 x 没有半传输事件（HT）；

1：在通道 x 产生了半传输事件（HT）。

Bit 4*(x-1)+1 **TCIFx：**通道 x 的传输完成标志（x = 1...8）

硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入 1 可以清除这里对应的标志位。

0：在通道 x 没有传输完成事件（TC）；

1：在通道 x 产生了传输完成事件（TC）。

Bit 4*(x-1)+0 **GIFx：**通道 x 的全局中断标志（x = 1...8）

硬件设置这些位。在 DMA_IFCR 寄存器的相应位写入 1 可以清除这里对应的标志位。

0：在通道 x 没有 TE、HT 或 TC 事件；

1：在通道 x 产生了 TE、HT 或 TC 事件。

### 17.5.2 DMA 中断寄存器标志清除寄存器（DMA_IFCR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTEIF8 | CHTIF8 | CTCIF8 | CGIF8 | CTEIF7 | CHTIF7 | CTCIF7 | CGIF7 | CTEIF6 | CHTIF6 | CTCIF6 | CGIF6 | CTEIF5 | CHTIF5 | CTCIF5 | CGIF5 |
| 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 |
| CTEIF4 | CHTIF4 | CTCIF4 | CGIF4 | CTEIF3 | CHTIF3 | CTCIF3 | CGIF3 | CTEIF2 | CHTIF2 | CTCIF2 | CGIF2 | CTEIF1 | CHTIF1 | CTCIF1 | CGIF1 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 4*(x-1)+3 **CTEIFx：**清除通道 x 的传输错误标志（x = 1...8）

这些位由软件设置和清除。

0：无作用

1：清除 DMA_ISR 寄存器中的对应 TEIF 标志。

Bit 4*(x-1)+2 CHTIFx：清除通道 x 的半传输标志（x = 1...8）

这些位由软件设置和清除。

0：无作用

0：清除 DMA_ISR 寄存器中的对应 HTIF 标志。

Bit 4*(x-1)+1 CTCIFx：清除通道 x 的传输完成标志（x = 1...8）

这些位由软件设置和清除。

0：无作用

0：清除 DMA_ISR 寄存器中的对应 TCIF 标志。

Bit 4*(x-1)+0 CGIFx：清除通道 x 的传全局中断标志（x = 1...8）

这些位由软件设置和清除。

0：无作用

0：清除 DMA_ISR 寄存器中的对应 GIF、TEIF、HTIF、TCIF 标志。

### 17.5.3 DMA 通道 x 配置寄存器（DMA_CCRx）

偏移地址：0x08 + 20 * (x-1)，（x = 1 ~ 8）
复位值：0x0000
当 EN=1 时，MEM2MEM、PL[1:0]、MSIZE[1:0]、PSIZE[1:0]、MINC、PINC 和 DIR 是只读的。
MEM2MEM 和 CIRC 位的状态不能同时为高。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | MEM2MEM | PL[1:0] | | MSIZE[1:0] | | PSIZE[1:0] | | MINC | PINC | CIRC | DIR | TEIE | HTIE | TCIE | EN |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:15 保留，必须保持为复位值。

Bit 14 **MEM2MEM：**存储器到存储器模式

0：禁止

1：使能

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位是只读的。*

Bit 13:12 **PL[1:0]：**通道优先级

00：低

01：中

10：高

11：最高

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域是只读的。*

Bit 11:10 **MSIZE[1:0]：**存储器数据宽度

定义了每次 DMA 传输到指定存储器的数据宽度。

在存储器到存储器模式下如果 DIR=1，该位域表示存储器源；如果 DIR=0，则表示存储器目标。

在外设到外设模式下如果 DIR=1，该位域表示外设源；如果 DIR=0，则表示外设目标。

00：8 位

01：16 位

10：32 位

11：保留

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域是只读的。*

Bit 9:8 **PSIZE[1:0]：**外设数据宽度

这些位由软件设置和清除。

在存储器到存储器模式下如果 DIR=1，该位域表示存储器目标；如果 DIR=0，则表示存储器源。

在外设到外设模式下如果 DIR=1，该位域表示外设目标；如果 DIR=0，则表示外设源。

00：8 位

01：16 位

10：32 位

11：保留

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域是只读的。*

Bit 7 **MINC：**存储器递增模式

定义了每次 DMA 传输到指定存储器时的递增模式。

在存储器到存储器模式下如果 DIR=1，该位表示存储器源；如果 DIR=0，则表示存储器目标。

在外设到外设模式下如果 DIR=1，该位表示外设源；如果 DIR=0，则表示外设目标。

0：禁止

1：使能

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位是只读的。*

Bit 6 **PINC：**外设地址增量模式

该位由软件设置和清除。

在存储器到存储器模式下如果 DIR=1，该位表示存储器目标；如果 DIR=0，则表示存储器源。

在外设到外设模式下如果 DIR=1，该位表示外设目标；如果 DIR=0，则表示外设源。

0：禁止

1：使能

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位是只读的。*

Bit 5 **CIRC：**循环模式

0：禁止

1：使能

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位是只读的。*

Bit 4 **DIR**：数据传输方向

此位仅在存储器到外设和外设到存储器模式下设置。

0：从外设读取

– 源属性由 PSIZE 和 PINC 以及 DMA_CPARx 寄存器定义。
在存储器到存储器模式下仍然有效。

– 目标属性由 MSIZE 和 MINC 以及 DMA_CMARx 寄存器定义。
在外设到外设模式下仍然有效。

1：从存储器读取

– 目标属性由 PSIZE 和 PINC 以及 DMA_CPARx 寄存器定义。
在存储器到存储器模式下仍然有效。

– 源属性由 MSIZE 和 MINC 以及 DMA_CMARx 寄存器定义。
在外设到外设模式下仍然有效。

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位是只读的。*

Bit 3 **TEIE**：传输错误中断使能

该位由软件设置和清除。

0：禁止 TE 中断

0：使能 TE 中断

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位不是只读的。*

Bit 2 **HTIE**：半传输中断使能

该位由软件设置和清除。

0：禁止 HT 中断

0：使能 HT 中断

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位不是只读的。*

Bit 1 **TCIE**：传输完成中断使能

该位由软件设置和清除。

0：禁止 TC 中断

0：使能 TC 中断

*注：此位由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位不是只读的。*

Bit 0 **EN**：通道使能

当发生通道传输错误时，此位由硬件清除。在 DMA_ISR 寄存器的 TEIFx 位被清除（通过设置 DMA_IFCR 寄存器的 CTEIFx 位）之前，软件无法再次设置此位（重新激活通道 x）。

0：通道禁止

1：通道使能

*注：此位由软件设置和清除。*

### 17.5.4 DMA 通道 x 传输数量寄存器（DMA_CNDTRx）

偏移地址：0x0C + 20 * (x-1)，（x = 1 ~ 8）

复位值：0x0000

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **NDT[15:0]：**数据传输数量（0 至 65535）

当通道启用时，此位域由硬件更新：

– 每次 DMA“先读后写”传输后，该值递减，表示剩余要传输的数据项数量。

– 如果传输的编程数据量已达到，且通道不在循环模式（DMA_CCRx 寄存器中 CIRC = 0），则该值保持为零。

– 如果通道处于循环模式（CIRC = 1），则在传输完成后，该值会自动重新加载为之前编程的值。

如果此位域为零，则无论通道状态如何（使能或禁止），都无法进行传输。

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域是只读的。*

### 17.5.5 DMA 通道 x 外设地址寄存器（DMA_CPARx）

偏移地址：0x10 + 20 * (x-1)，（x = 1 ~ 8）

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA[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 |
| PA[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 **PA[31:0]：**外设地址

它包含从/向外设数据寄存器读取/写入数据的基地址。

当 PSIZE[1:0] = 01（16 位），PA[31:0]的位 0 被忽略。访问将自动对齐到半字地址。

当 PSIZE[1:0] = 10（32 位），PA[31:0]的位 1 和位 0 被忽略。访问将自动对齐到字地址。

在存储器到存储器模式下如果 DIR = 1，该位域表示存储器目标地址；如果 DIR = 0，则表示存储器源地址。

在外设到外设模式下如果 DIR = 1，该位域表示外设目标地址；如果 DIR = 0，则表示外设源地址。

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域不是只读的。*

### 17.5.6 DMA 通道 x 外设存储地址寄存器（DMA_CMARx）

偏移地址：0x14 + 20 * (x-1)，（x = 1 ~ 8）

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MA[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 |
| MA[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **MA[31:0]：**存储器地址

它包含从/向存储器读取/写入数据的基地址。

当 MSIZE[1:0] = 01（16 位），MA[31:0]的位 0 被忽略。访问将自动对齐到半字地址。

当 MSIZE[1:0] = 10（32 位），MA[31:0]的位 1 和位 0 被忽略。访问将自动对齐到字地址。

在存储器到存储器模式下如果 DIR = 1，该位域表示存储器源地址；如果 DIR = 0，则表示存储器目标地址。

在外设到外设模式下如果 DIR = 1，该位域表示外设源地址；如果 DIR = 0，则表示外设目标地址。

*注：此位域由软件设置和清除。在通道使能（EN=1）时，不得对其进行写入。在通道使能（EN=1）时，此位域不是只读的。*

# 18 DMA 请求复用器（DMAMUX）

## 18.1 简介

外设通过设置其 DMA 请求信号来指示存在 DMA 传输请求。DMA 控制器会处理 DMA 请求并生成 DMA 确认信号，而且相应的 DMA 请求信号也将变为无效，但在此之前 DMA 请求一直处于挂起状态。

在本文档中，DMA 请求/确认协议所需的控制信号组并未明确列出和说明，而是称作 DMA 请求线。

DMAMUX 请求复用器可在产品的外设和 DMA 控制器之间重新配置（路由）DMA 请求线。该路由功能通过可编程的多通道 DMA 请求线复用器来确保实现。每条通道可不受限制地选择一个 DMA 请求线，或者按照与来自其 DMAMUX 同步输入的事件相同步的方式选择一个 DMA 请求线。此外，DMAMUX 还可用作其输入触发信号的可编程事件的 DMA 请求发生器。

第 19.3.1 节规定了 DMAMUX 实例的数量及其主要特性。

DMAMUX 请求复用器输入到外设 DMA 请求线和到 DMAMUX 请求发生器输出的分配、DMAMUX 请求复用器输出到 DMA 控制器通道的分配以及 DMAMUX 同步和触发输入到内部和外部信号的分配取决于产品实现，具体在第 19.6 节中进行了介绍。

## 18.2 主要特性

- 多达 16 通道可编程 DMA 请求线复用器输出
- 多达 4 通道 DMA 请求发生器
- 多达 20 个 DMA 请求发生器的触发输入
- 多达 20 个同步输入
- 每个 DMA 请求发生器通道均具有：
  - DMA 请求触发输入选择器
  - DMA 请求计数器
  - 所选 DMA 请求触发输入的事件溢出标志
- 每个 DMA 请求线复用器通道输出均具有：
  - 多达 88 个来自外设的输入 DMA 请求线
  - 一个 DMA 请求线输出
  - 同步输入选择器
  - DMA 请求计数器
  - 所选同步输入的事件溢出标志
  - 一个事件输出，用于 DMA 请求链接

## 18.3 DMAMUX 实现

### 18.3.1 DMAMUX 实例化

DMAMUX 通过下表中列出的硬件配置参数进行实例化。

表 18.1 DMAMUX 实例化

| 特征 | DMAMUX |
|---|---|
| DMAMUX 输出请求通道数 | 16 |
| DMAMUX 请求发生器通道数 | 4 |
| DMAMUX 请求触发输入数 | 20 |
| DMAMUX 同步输入数 | 20 |
| DMAMUX 外设请求输入数 | 88 |

### 18.3.2 DMAMUX 映射

资源到 DMAMUX 的映射通过硬接线实现。

DMAMUX 与 DMA1 和 DMA2 配合使用：

- DMAMUX 通道 0 到 7 与 DMA1 通道 1 到 8 相连
- DMAMUX 通道 8 到 15 与 DMA2 通道 1 到 8 相连

表 18.2 DMAMUX：复用器输入到资源的分配

| DMA 请求 MUX 输入 | 资源 | DMA 请求 MUX 输入 | 资源 | DMA 请求 MUX 输入 | 资源 |
|---|---|---|---|---|---|
| 1 | DMAMUX_Req G0 | 44 | TIM1_CH3 | 87 | 保留 |
| 2 | DMAMUX_Req G1 | 45 | TIM1_CH4 | 88 | 保留 |
| 3 | DMAMUX_Req G2 | 46 | TIM1_UP | 89 | 保留 |
| 4 | DMAMUX_Req G3 | 47 | TIM1_TRIG | 90 | 保留 |
| 5 | ADC1 | 48 | TIM1_COM | 91 | 保留 |
| 6 | DAC1_CH1 | 49 | TIM8_CH1 | 92 | 保留 |
| 7 | DAC1_CH2 | 50 | TIM8_CH2 | 93 | 保留 |
| 8 | TIM6_UP | 51 | TIM8_CH3 | 94 | 保留 |
| 9 | TIM7_UP | 52 | TIM8_CH4 | 95 | HRTIM_MASTER（hrtim_dma1） |
| 10 | SPI1_RX | 53 | TIM8_UP | 96 | HRTIM_TIMA（hrtim_dma2） |
| 11 | SPI1_TX | 54 | TIM8_TRIG | 97 | HRTIM_TIMB（hrtim_dma3） |
| 12 | SPI2_RX | 55 | TIM8_COM | 98 | HRTIM_TIMC（hrtim_dma4） |
| 13 | SPI2_TX | 56 | TIM2_CH1 | 99 | HRTIM_TIMD（hrtim_dma5） |
| 14 | 保留 | 57 | TIM2_CH2 | 100 | HRTIM_TIME（hrtim_dma6） |
| 15 | 保留 | 58 | TIM2_CH3 | 101 | HRTIM_TIMF（hrtim_dma7） |
| 16 | I2C1_RX | 59 | TIM2_CH4 | 102 | DAC3_CH1 |

| DMA 请求 MUX 输入 | 资源 | DMA 请求 MUX 输入 | 资源 | DMA 请求 MUX 输入 | 资源 |
|---|---|---|---|---|---|
| 17 | I2C1_TX | 60 | TIM2_UP | 103 | DAC3_CH2 |
| 18 | I2C2_RX | 61 | TIM3_CH1 | 104 | 保留 |
| 19 | I2C2_TX | 62 | TIM3_CH2 | 105 | 保留 |
| 20 | 保留 | 63 | TIM3_CH3 | 106 | 保留 |
| 21 | 保留 | 64 | TIM3_CH4 | 107 | 保留 |
| 22 | 保留 | 65 | TIM3_UP | 108 | 保留 |
| 23 | 保留 | 66 | TIM3_TRIG | 109 | 保留 |
| 24 | UART1_RX | 67 | TIM4_CH1 | 110 | FMAC_Read |
| 25 | UART1_TX | 68 | TIM4_CH2 | 111 | FMAC_Write |
| 26 | UART2_RX | 69 | TIM4_CH3 | 112 | Cordic_Read |
| 27 | UART2_TX | 70 | TIM4_CH4 | 113 | Cordic_ Write |
| 28 | UART3_RX | 71 | TIM4_UP | 114 | 保留 |
| 29 | UART3_TX | 72 | TIM5_CH1 | 115 | 保留 |
| 30 | UART4_RX | 73 | TIM5_CH2 | 116 | DAC2_CH2 |
| 31 | UART4_TX | 74 | TIM5_CH3 | 117 | 保留 |
| 32 | UART5_RX | 75 | TIM5_CH4 | 118 | 保留 |
| 33 | UART5_TX | 76 | TIM5_UP | 119 | 保留 |
| 34 | 保留 | 77 | TIM5_TRIG | 120 | 保留 |
| 35 | 保留 | 78 | TIM15_CH1 | 121 | 保留 |
| 36 | ADC2 | 79 | TIM15_UP | 122 | 保留 |
| 37 | ADC3 | 80 | TIM15_TRIG | 123 | 保留 |
| 28 | 保留 | 81 | TIM15_COM | 124 | 保留 |
| 29 | 保留 | 82 | TIM16_CH1 | 125 | 保留 |
| 40 | 保留 | 83 | TIM16_UP | 126 | 保留 |
| 41 | DAC2_CH1 | 84 | TIM17_CH1 | 127 | 保留 |
| 42 | TIM1_CH1 | 85 | TIM17_UP | - | - |
| 43 | TIM1_CH2 | 86 | 保留 | - | - |

表 18.3 DMAMUX：触发输入到资源的分配

| 触发输入 | 资源 | 触发输入 | 资源 |
|---|---|---|---|
| 0 | EXTI LINE0 | 16 | DMAMUX1_ch0_event |
| 1 | EXTI LINE1 | 17 | DMAMUX1_ch1_event |
| 2 | EXTI LINE2 | 18 | DMAMUX1_ch2_event |
| 3 | EXTI LINE3 | 19 | DMAMUX1_ch3_event |
| 4 | EXTI LINE4 | 20 | 保留 |
| 5 | EXTI LINE5 | 21 | 保留 |
| 6 | EXTI LINE6 | 22 | 保留 |
| 7 | EXTI LINE7 | 23 | 保留 |
| 8 | EXTI LINE8 | 24 | 保留 |
| 9 | EXTI LINE9 | 25 | 保留 |
| 10 | EXTI LINE10 | 26 | 保留 |
| 11 | EXTI LINE11 | 27 | 保留 |
| 12 | EXTI LINE12 | 28 | 保留 |
| 13 | EXTI LINE13 | 29 | 保留 |
| 14 | EXTI LINE14 | 30 | 保留 |
| 15 | EXTI LINE15 | 31 | 保留 |

表 18.4 DMAMUX：同步输入到资源的分配

| 同步输入 | 资源 | 同步输入 | 资源 |
|---|---|---|---|
| 0 | EXTI LINE0 | 16 | DMAMUX1_ch0_event |
| 1 | EXTI LINE1 | 17 | DMAMUX1_ch1_event |
| 2 | EXTI LINE2 | 18 | DMAMUX1_ch2_event |
| 3 | EXTI LINE3 | 19 | DMAMUX1_ch3_event |
| 4 | EXTI LINE4 | 20 | 保留 |
| 5 | EXTI LINE5 | 21 | 保留 |
| 6 | EXTI LINE6 | 22 | 保留 |
| 7 | EXTI LINE7 | 23 | 保留 |
| 8 | EXTI LINE8 | 24 | 保留 |
| 9 | EXTI LINE9 | 25 | 保留 |
| 10 | EXTI LINE10 | 26 | 保留 |
| 11 | EXTI LINE11 | 27 | 保留 |
| 12 | EXTI LINE12 | 28 | 保留 |
| 13 | EXTI LINE13 | 29 | 保留 |
| 14 | EXTI LINE14 | 30 | 保留 |
| 15 | EXTI LINE15 | 31 | 保留 |

## 18.4 DMAMUX 功能描述

### 18.4.1 DMAMUX 框图

DMAMUX 框图如下图所示。

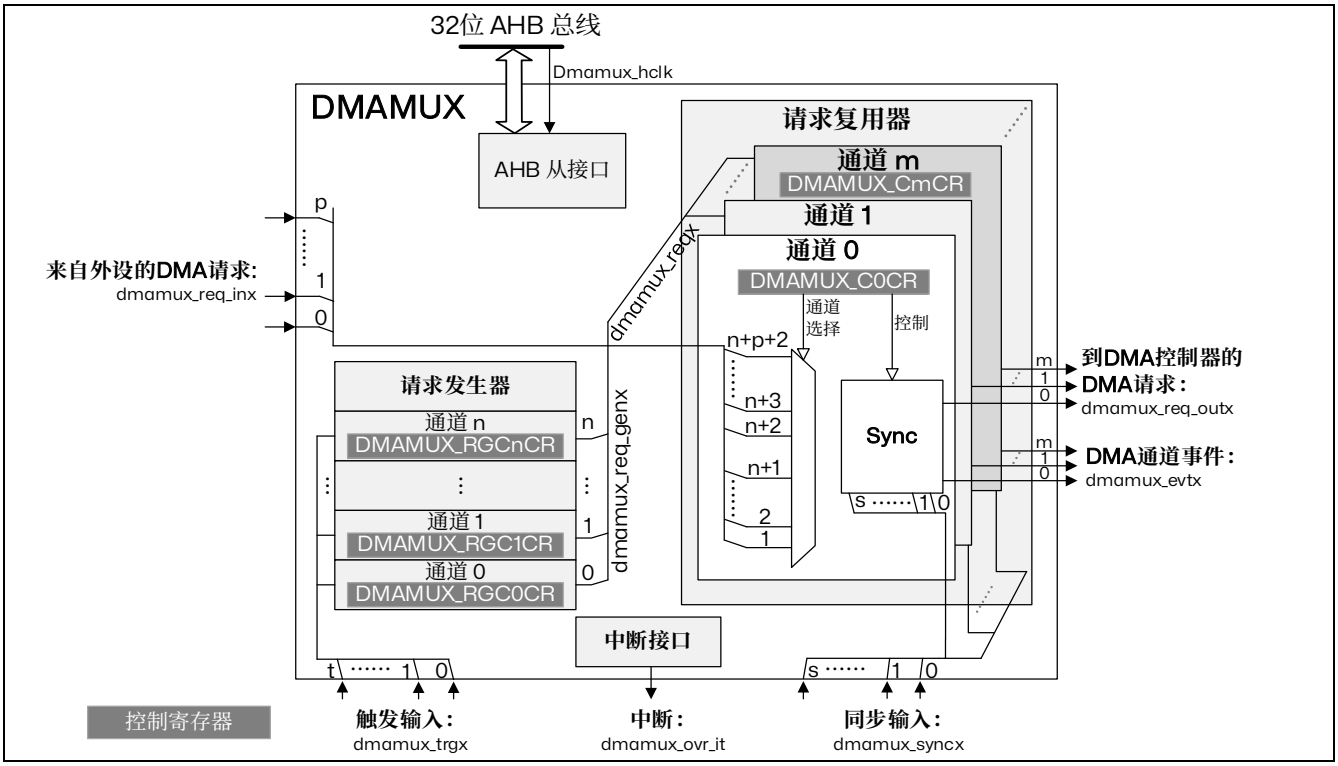



图 18.1 DMAMUX 框图

DMAMUX 具有两个主要子模块：请求线复用器和请求线发生器。

该实现可：

- 分配 DMAMUX 请求复用器子模块输入（dmamux_reqx），这些输入来自外设（dmamux_req_inx）和 DMAMUX 请求发生器子模块的通道（dmamux_req_genx）
- 将 DMAMUX 请求输出分配到 DMA 控制器的通道（dmamux_req_outx）
- 将内部或外部信号分配到 DMA 请求触发输入（dmamux_trgx）
- 将内部或外部信号分配到同步输入（dmamux_syncx）

### 18.4.2 DMAMUX 信号

下表列出了 DMAMUX 信号。

表 18.5 DMAMUX 信号

| 信号名称 | 说明 |
|---|---|
| dmamux_hclk | DMAMUX AHB 时钟 |
| dmamux_req_inx | DMAMUX DMA 请求线输入（来自外设） |
| dmamux_trgx | DMAMUX DMA 请求触发输入（请求生成器子模块） |
| dmamux_req_genx | DMAMUX 请求生成器子模块通道输出 |
| dmamux_reqx | DMAMUX 请求复用器子模块输入（来自外设请求和请求发生器通道） |
| dmamux_syncx | DMAMUX 同步输入（到请求复用子模块） |
| dmamux_req_outx | DMAMUX 请求输出（到 DMA 控制器） |
| dmamux_evtx | DMAMUX 事件中断 |
| dmamux_ovr_it | DMAMUX 溢出中断 |

### 18.4.3 DMAMUX 通道

DMAMUX 通道是一个可能包括额外的 DMAMUX 请求发生器通道的 DMAMUX 请求复用器通道，具体取决于所选的请求复用器输入。

DMAMUX 请求复用器通道专用于连接到 DMA 控制器的一个通道。

**通道配置流程**

请遵循以下顺序来配置 DMAMUX x 通道和相关的 DMA 通道 y:

1. 对 DMA 通道 y 进行完整的设置和配置，但不要使能通道 y。
2. 对相关 DMAMUX y 通道进行完整的设置和配置。
3. 最后，将 DMA y 通道寄存器中的 EN 位置 1 以激活 DMA 通道。

### 18.4.4 DMAMUX 请求线复用器

DMAMUX 请求复用器及其多条通道可确保 DMA 请求/确认控制信号（称为 DMA 请求线）的实际路由。

每个 DMA 请求线并行连接到 DMAMUX 请求线复用器的所有通道。

DMA 请求来自外设或 DMAMUX 请求发生器。

DMAMUX 请求线复用器通道 x 按照 DMAMUX_CxCR 寄存器中的 8 位 DMAREQ_ID 位域的配置来选择 DMA 请求线编号。

*注：位域* DMAREQ_ID *为空值表示未选择任何* DMA *请求线。*

**警告：** 不得为不同的 x 和 y DMAMUX 请求复用器通道编程相同的非空 DMA_REQ_ID 值（通过 DMAMUX_CxCR 和 DMAMUX CyCR）。不允许为 DMAMUX 请求线复用器的两个不同通道配置相同的非零 DMAREQ_ID。

除了 DMA 请求选择外，还可根据需要配置和使能同步模式和/或事件生成

**同步模式和通道事件生成**

每个 DMAMUX 请求线复用器通道 x 可单独同步，方法是将 DMAMUX_CxCR 寄存器中的同步使能（SE）位置 1。

DMAMUX 具有多个同步输入。同步输入并行连接到请求复用器的所有通道。

同步输入通过给定通道 x 的 DMAMUX_CxCR 寄存器中的 SYNC_ID 位域选择。

当某个通道处于此同步模式时，如果通过 DMAMUX_CxCR 寄存器的 SPOL[1:0]位域检测到所选输入同步信号上的可编程上升沿/下降沿，则所选输入 DMA 请求线信号将被传送到复用器通道输出。

此外，DMAMUX 请求复用器内部有一个可编程 DMA 请求计数器，可用于实现通道请求输出生成功能以及事件生成功能。通道 x 输出上的事件生成功能通过 DMAMUX_CxCR 寄存器的 EGE 位（事件生成使能）使能。

如以下两个图所示，检测到同步输入的边沿后，所选输入 DMA 请求线将连接到 DMAMUX 复用器通道 x 输出。

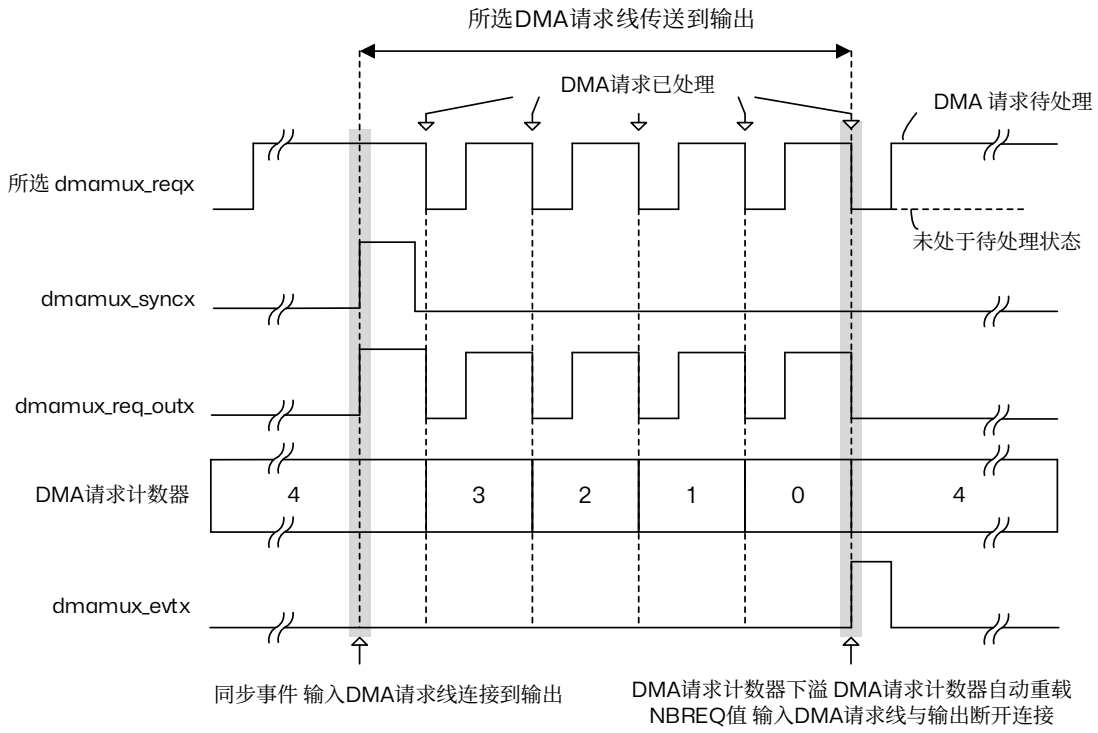
*注：如果没有待处理的选定输入 DMA 请求线时发生同步事件，则该事件将被丢弃。以下确认的输入请求线在再次发生同步事件之前不会连接到 DMAMUX 复用器通道输出。*

从这一点开始，每次 DMA 控制器处理连接的 DMAMUX 请求（即一个请求被处理且不再被确认）时，DMAMUX 请求计数器就会递减。

当 DMA 请求计数器计数不足时，它会自动加载 DMAMUX_CxCR 寄存器中 NBREQ 位域的值，并且输入 DMA 请求线与多路复用器通道 x 输出断开连接。

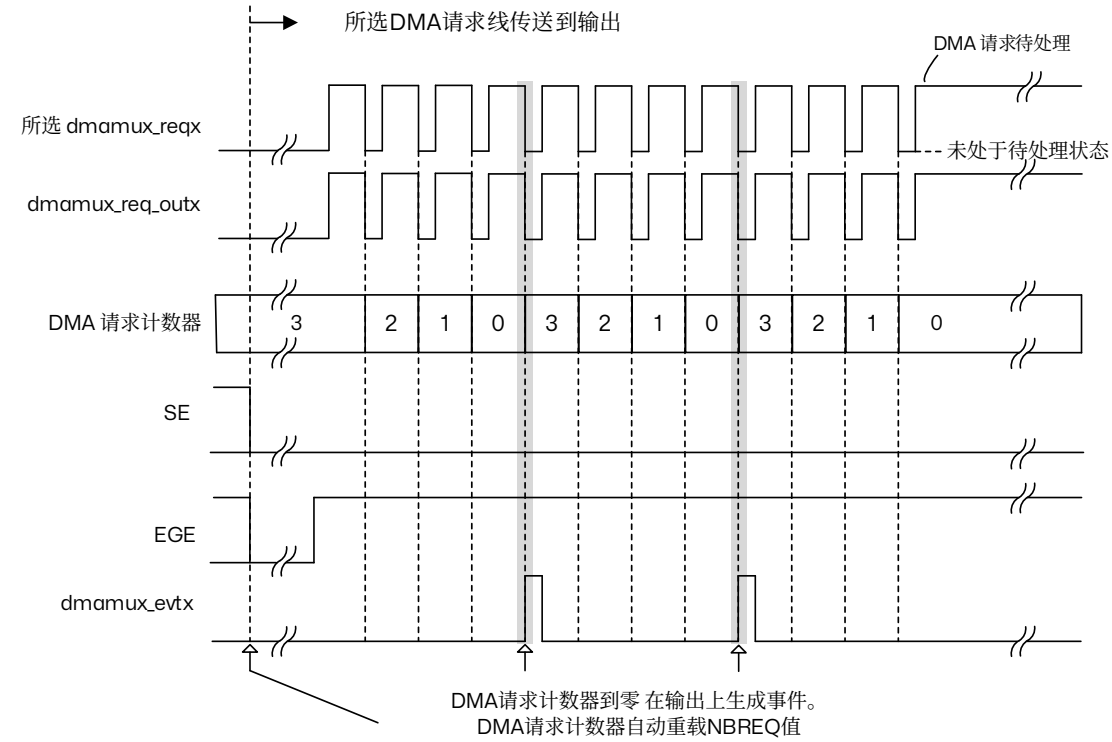
因此，检测到同步事件后传送到复用器通道 x 输出的 DMA 请求数等于 NBREQ 位域中的值加 1。

*注：只能通过软件在相应复用器通道 x 的同步使能位（SE）和事件生成使能位（EGE）均禁止时写入 NBREQ 位域的值。*




示例：DMAMUX_CCRx 配置：NBREQ=4，SE=1，EGE=1，SPOL=01（上升沿）

图 18.2 DMAMUX 请求线复用器通道的同步模式




示例：DMAMUX_CCRx 配置：NBREQ=3，SE=0，EGE=1

图 18.3 DMA 请求线复用器通道的事件生成

如果 EGE 使能，当复用器通道的 DMA 请求计数器自动重新装入已编程 NBREQ 位域的值时，复用器通道会生成一个通道事件（即，一个 AHB 时钟周期的脉冲）。如上两图所示。

*注：如果 EGE 被使能且 NBREQ=0，则每次 DMA 请求被处理后都会生成一个事件。*

*注：如果边沿后的状态保持稳定的时间持续 2 个 AHB 时钟周期以上，则会检测到同步事件（边沿）。*

*写入 DMAMUX_CxCR 寄存器后，同步事件将被屏蔽 3 个 AHB 时钟周期。*

**同步溢出和中断**

如果在 DMA 请求计数器的值小于复用器通道 x 已编程的 NBREQ 位域值时发生新的同步事件，则 DMAMUX_CSR 状态寄存器中的同步溢出标志位 SOFx 将置 1。

*注：DMA 控制器的相关通道使用结束后，应禁止请求复用器通道 x 同步 (DMAMUX_CxCR.SE=0)。否则，在新检测到的同步事件后，由于未从 DMA 控制器收到 DMA 确认（即无已处理请求），将会发生同步溢出。*

溢出标志 SOFx 的复位方式是将 DMAMUX_CFR 寄存器中的相关清除同步溢出标志位 CSOFx 置 1。

如果 DMAMUX_CxCR 寄存器中同步溢出中断使能位 SOIE 置 1，则将同步溢出标志置 1 会产生中断。

### 18.4.5 DMAMUX 请求发生器

DMAMUX 请求发生器会在其 DMA 请求触发输入上出现触发事件后产生 DMA 请求。

DMAMUX 请求发生器有多个通道。DMA 请求触发输入并行连接到所有通道。

DMAMUX 请求发生器通道的输出是 DMAMUX 请求线复用器的输入。

每个 DMAMUX 请求发生器通道 x 在相应的 DMAMUX_RGxCR 寄存器中都有一个使能位 GE（发生器使能）。

DMAMUX 请求发生器通道 x 的 DMA 请求触发输入通过相应 DMAMUX_RGxCR 寄存器中的 SIG_ID（触发信号 ID）位域选择。

DMA 请求触发输入上的触发事件可以是上升沿、下降沿或任一边沿。有效边沿通过相应 DMAMUX_RGxCR 寄存器中的 GPOL（发生器极性）位域选择。

触发事件后，相应发生器通道开始在其输出上生成 DMA 请求。每个已处理 DMA 请求（即，当请求信号取消激活时）会使 DMAMUX 请求发生器内置的 DMA 请求计数器减 1。下溢时，DMA 请求计数器将自动装入相应 DMAMUX_RGxCR 寄存器的 GNBREQ 位域中的值，请求发生器通道将停止生成 DMA 请求。

因此，触发事件后生成的 DMA 请求的数量为 GNBREQ+1。

*注：只能通过软件在相应发生器通道 x 的使能位 GE 被禁止时写入 GNBREQ 位域的值。*

*没有硬件写保护。*

*如果边沿后的状态保持稳定的时间持续 2 个 AHB 时钟周期以上，则会检测到触发事件（边沿）。*

*写入 DMAMUX_RGxCR 寄存器后，触发事件将被屏蔽 3 个 AHB 时钟周期。*

**触发溢出和中断**

如果在 DMAMUX 请求发生器计数器的值小于相应请求发生器通道 x 已编程的 GNBREQ 位域值时发

生新的 DMA 请求触发事件，并且请求发生器通道 x 通过 GE 使能，则状态 DMAMUX_RGSR 寄存器中的请求触发事件溢出标志位 OFx 将通过硬件置为有效。

*注：DMA 控制器的相关通道使用结束后，应禁止请求发生器通道 x（DMAMUX_RGxCR.GE=0）。否则，在新检测到的触发事件后，由于未从 DMA 收到确认（即无已处理请求），将会发生触发溢出。*

溢出标志 OFx 的复位方式是将 DMAMUX_RGCFR 寄存器中的相关清除溢出标志位 COFx 置 1。

如果 DMAMUX_RGxCR 寄存器中的 DMA 请求触发事件溢出中断使能位 OIE 置 1，则将 DMAMUX 请求触发溢出标志置 1 会产生中断。

## 18.5 DMAMUX 中断

发生以下事件时会产生中断：

- 每个 DMA 请求线复用器通道发生同步事件溢出
- 每个 DMA 请求发生器通道发生触发事件溢出

对任一事件，每个通道的中断使能、状态和清除标志寄存器位均可单独使用。

表 18.6 DMAMUX 中断

| 中断信号 | 中断事件 | 事件标志 | 清除位 | 使能位 |
|---|---|---|---|---|
| dmamuxovr_it | DMAMUX 请求线复用器的通道 x 上发生同步事件溢出 | SOFx | CSOFx | SOIE |
| | DMAMUX 请求发生器的通道 x 上发生触发事件溢出 | OFx | COFx | OIE |

## 18.6 DMAMUX 寄存器描述

请参考包含 DMAMUX 基地址寄存器边界地址的表格。
DMAMUX 寄存器可以按字节（8 位）、半字（16 位）或字（32 位）进行访问。
地址必须与数据大小对齐。

### 18.6.1 DMAMUX 请求线复用器通道 x 配置寄存器（DMAMUX_CxCR）

偏移地址：0x000 + 0x04*x，（x = 0 ~ 15）
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | SYNC_ID[4:0] | | | | | NBREQ[4:0] | | | | | SPOL[1:0] | | SE |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | EGE | SOIE | Res | DMAREQ_ID[6:0] | | | | | | |
| | | | | | | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持为复位值。

Bit 28:24 **SYNC_ID[4:0]：**同步识别符
选择同步输入。（请参见表 19.4 DMAMUX：同步输入到资源的分配）

Bit 23:19 **NBREQ[4:0]：**要转发的 DMA 请求数减 1
定义在同步事件后要转发到 DMA 控制器的 DMA 请求的数量，和/或生成输出事件前的 DMA 请求的数量。
只有当 SE 和 EGE 位均为低电平时，才能写入该位域。

Bit 18:17 **SPOL[1:0]：**同步极性
定义所选同步输入的边沿极性：
00：无事件。即，无同步和检测。
01：上升沿
10：下降沿
11：上升沿和下降沿

Bit 16 **SE：**同步使能
0：失能同步
1：使能同步

Bit 15:10 保留，必须保持为复位值。

Bit 9 **EGE：**事件生成使能
0：禁止事件生成
1：使能事件生成

Bit 8 **SOIE：**同步溢出中断使能
0：禁止中断
1：使能中断

Bit 7 保留，必须保持为复位值。

Bit 6:0 **DMAREQ_ID[6:0]：**DMA 请求标识
选择输入 DMA 请求。（请参见表 19.3 DMAMUX：触发输入到资源的分配）

### 18.6.2 DMAMUX 请求线复用器中断通道状态寄存器（DMAMUX_CSR）

偏移地址：0x080

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOF15 | SOF14 | SOF13 | SOF12 | SOF11 | SOF10 | SOF9 | SOF8 | SOF7 | SOF6 | SOF5 | SOF4 | SOF3 | SOF2 | SOF1 | SOF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **SOF[15:0]：**同步溢出事件标志

当 DMA 请求线复用器通道 x 上发生同步事件且 DMA 请求计数器的值小于 NBREQ 时，该标志将置 1。

该标志的清零方式是向 DMAMUX_CFR 寄存器中的相应 CSOFx 位写入 1。

### 18.6.3 DMAMUX 请求线复用器中断清除标志寄存器（DMAMUX_CFR）

偏移地址：0x084

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSOF15 | CSOF14 | CSOF13 | CSOF12 | CSOF11 | CSOF10 | CSOF9 | CSOF8 | CSOF7 | CSOF6 | CSOF5 | CSOF4 | CSOF3 | CSOF2 | CSOF1 | CSOF0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **CSOF[15:0]：**清除同步溢出事件标志

将 1 写入每个位时，DMAMUX_CSR 寄存器中相应的溢出标志 SOFx 将清零。

### 18.6.4 DMAMUX 请求发生器通道 x 配置寄存器（DMAMUX_RGxCR）（x=0...3）

偏移地址：0x100 + 0x04*x，（x = 0 ~ 3）

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | GNBREQ[4:0] | | | | | GPOL[1:0] | | GE |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OIE | Reserved | | | SIG_ID[4:0] | | | | |
| | | | | | | | rw | | | | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持为复位值。

Bit 23:19 **GNBREQ[4:0]：**要生成的 DMA 请求数（减 1）

定义触发事件后生成的 DMA 请求的数量。生成的 DMA 请求的实际数量为 GNBREQ+1。

*注：只有在* GE *位禁止时才能写入此位域。*

Bit 18:17 **GPOL[1:0]：**DMA 请求发生器触发极性

定义所选触发输入的边沿极性

00：无事件。即，无触发检测和生成。
01：上升沿
10：下降沿
11：上升沿和下降沿

Bit 16 **GE**：DMA 请求发生器通道 x 使能
0：禁止 DMA 请求发生器通道 x
1：使能 DMA 请求发生器通道 x

Bit 15:9 保留，必须保持为复位值。

Bit 8 **OIE**：触发溢出中断使能
0：禁止在出现触发溢出事件时产生中断。
1：使能在出现触发溢出事件时产生中断。

Bit 7:5 保留，必须保持为复位值。

Bit 4:0 **SIG_ID[4:0]**：信号标识
选择用于 DMA 请求发生器的通道 x 的 DMA 请求触发输入

### 18.6.5 DMAMUX 请求发生器中断状态寄存器（DMAMUX_RGSR）

偏移地址：0x140
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | OF3 | OF2 | OF1 | OF0 |
| | | | | | | | | | | | | r | r | r | r |

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **OF[3:0]**：触发溢出事件标志
当 DMA 请求发生器通道 x 上发生触发事件且 DMA 请求发生器计数器的值小于 GNBREQ 时，该标志将置 1。
该标志的清零方式是向 DMAMUX_RGCFR 寄存器中的相应 COFx 位写入 1。

### 18.6.6 DMAMUX 请求发生器中断清除标志寄存器（DMAMUX_RGCFR）

偏移地址：0x144
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | COF3 | COF2 | COF1 | COF0 |
| | | | | | | | | | | | | w | w | w | w |

Bit 31:16 保留，必须保持为复位值。

Bit 15:0 **COF[3:0]**：清除触发溢出事件标志
将 1 写入每个位时，DMAMUX_RGSR 寄存器中相应的溢出标志 OFx 将清零。

# 19 比较器（COMP）

## 19.1 简介

该设备嵌入了 4 个超高速模拟比较器。
这些比较器可用于各种功能，包括：

- 在模拟信号的触发下从低功耗模式唤醒
- 模拟信号调理
- 与定时器的 PWM 输出结合使用时，构成逐周期电流控制环路

## 19.2 主要特性

- 每个比较器具有可配置的正负输入，用于灵活的电压选择：
  - 多路 I/O 引脚
  - DAC 通道
  - 通过调节器（缓冲分压器）提供的内部参考电压和三个因数分压值（1/4、1/2、3/4）
- 可编程迟滞
- 输出重定向到 I/O 或定时器输入，用于触发快速 PWM 关断的刹车事件
- 输出遮蔽，以抵抗开关噪声
- 每个通道生成的中断，可用于从睡眠和停止模式唤醒

## 19.3 功能描述

### 19.3.1 COMP 框图

一个比较器 COMP 框图如下图所示：

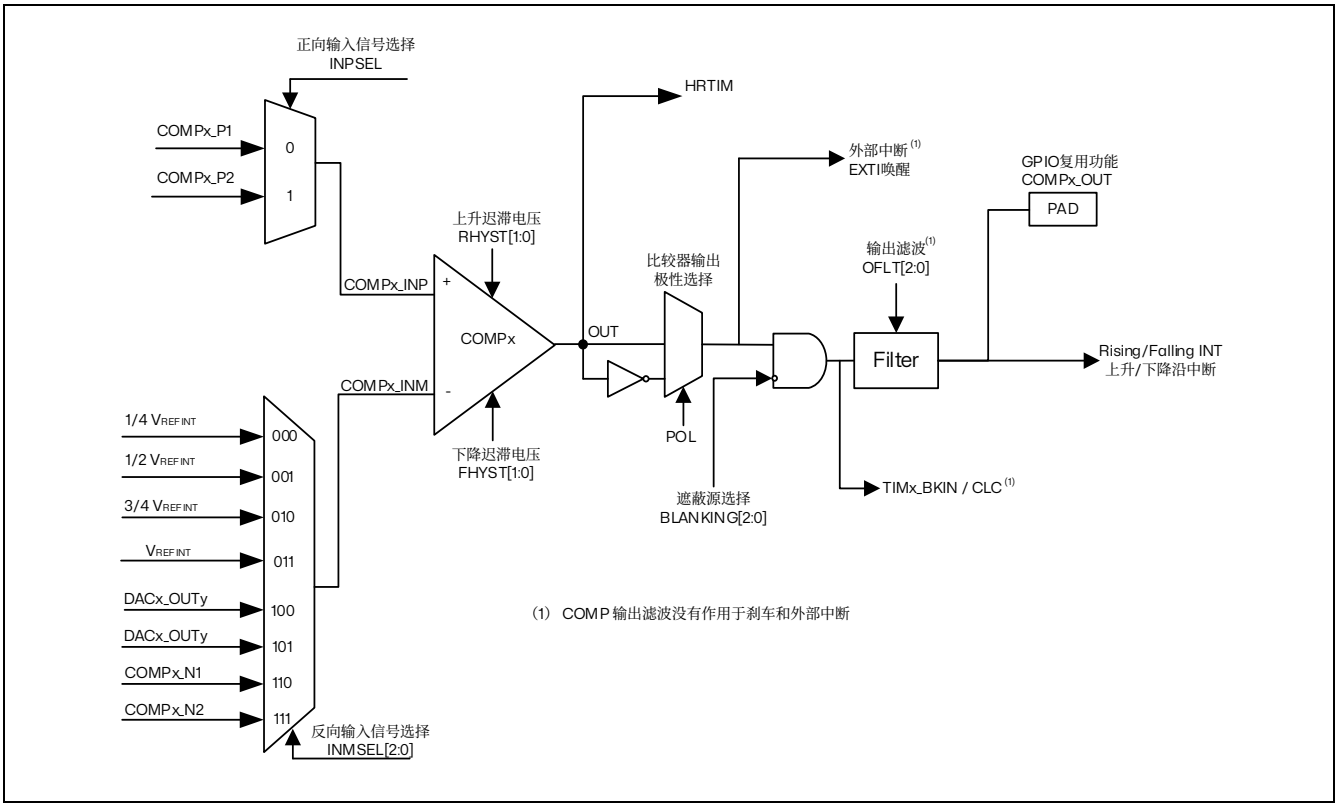


图 19.1 COMP 框图

### 19.3.2 COMP 引脚和内部信号

用作比较器输入的 I/O 必须在 GPIO 寄存器中配置为模拟模式。

比较器输出可以使用数据手册“复用功能映射”表中给出的复用功能通道连接到 I/O。

输出也可以在内部重定向到用于以下用途的各种定时器输入：

- 使用 BKIN 和 BKIN2 输入紧急关断 PWM 信号
- 使用 OCREF_CLR 输入进行逐周期电流控制
- 用于时序测量的输入捕捉

可以同时将比较器输出内部和外部重定向。

表 19.1 COMPx 正向输入分配

| INPSEL | COMP1_INP | COMP2_INP | COMP3_INP | COMP4_INP |
|---|---|---|---|---|
| 0 | PA0 | PA1 | PB13 | PB14 |
| 1 | PC15 | PC5 | PE8 | PE12 |

表 19.2 COMPx 反向输入分配

| INMSEL[2:0] | COMP1_INM | COMP2_INM | COMP3_INM | COMP4_INM |
|---|---|---|---|---|
| 000 | 1/4 $V_{REFINT}$ | | | |
| 001 | 1/2 $V_{REFINT}$ | | | |
| 010 | 3/4 $V_{REFINT}$ | | | |
| 011 | $V_{REFINT}$ | | | |
| 100 (1) | DAC3_CH1 | DAC3_CH2 | DAC3_CH1 | DAC3_CH2 |
| 101 (1) | DAC1_CH1 | DAC1_CH2 | DAC2_CH1 | DAC2_CH2 |
| 110 | PC0 | PC1 | PD9 | PD10 |
| 111 | PC14 | PC4 | PE7 | PE11 |

(1) 选择 DAC 通道内部连接至 COMP 反向输入端时，需要关闭 DAC 输出缓冲器。

### 19.3.3 COMP 复位和时钟

由时钟控制器提供的 COMP 时钟与 APB2 时钟同步。

RCC 控制器中没有专用于 COMP 时钟使能控制位。复位和时钟使能位是 COMP 和 SYSCFG 共用的。

*注：**重要：**极性选择逻辑和输出重定向到端口独立于 APB 时钟工作。这使得比较器即使在停止模式下也能工作。*

### 19.3.4 COMP 锁定机制

比较器能用于安全的用途，比如过流或者过热保护。在某些特定的安全需求的应用中，有必要保证比较器设置不能被无效寄存器访问或者程序计数器破坏所改变。

为此，比较器控制和状态寄存器可以是写保护的（只读）。

一旦编程完成，就可以设置 COMP_CxCR 寄存器的 LOCK 位。这会导致大部分寄存器变成只读的，具体位参阅寄存器章节。

只能通过 MCU 复位来移除写保护。

### 19.3.5 COMP 防误触发功能

**（1）当 COMP 和 ADC 同时使用时**

当 COMP 的 P 或 N 端与 ADC 采样通道是同一个 IO 时，当外部接 RC 电路或 ADC 在通道转换时造成压差，此时 COMP 可能被 ADC 在通道转换时产生的压差影响而导致误触发。

当 COMP_CxCR 寄存器的 BAS_EN=1 时，根据 COMP_CxCR 寄存器的 INPSEL 和 COMP_CxCR 寄存器的 INMSEL[2:0]（x=1/2/3/4）的配置，在对应的 ADC 采样通道采样时对 COMP 输出进行遮蔽，这样可以防止由 ADC 的转换压差和 RC 电路造成的 COMP 误触发。

下两张表分别为 COMPx 的 P 端/N 端和 ADC 的 IO 对应关系。

表 19.3 COMPx_P 端和 ADC 的 IO 对应关系

| CxCR_INPSEL | 值 | P 端输入 | ADC 采样通道 |
|---|---|---|---|
| C1CR_INPSEL | 0 | COMP1_P1 | ADC1_IN4 |
| | 1 | COMP1_P2 | ADC1_IN2 |
| C2CR_INPSEL | 0 | COMP2_P1 | ADC1_IN5 |
| | 1 | COMP2_P2 | ADC2_IN2 |
| C3CR_INPSEL | 0 | COMP3_P1 | ADC3_IN5 |
| | 1 | COMP3_P2 | - |
| C4CR_INPSEL | 0 | COMP4_P1 | ADC3_IN6 |
| | 1 | COMP4_P2 | - |

表 19.4 COMPx_N 端和 ADC 的 IO 对应关系

| CxCR_INMSEL | 值 | N 端输入 | ADC 采样通道 |
|---|---|---|---|
| C1CR_INMSEL | 110 | COMP1_N1 | ADC1_IN3 |
| | 111 | COMP1_N2 | ADC1_IN1 |
| C2CR_INMSEL | 110 | COMP2_N1 | ADC1_IN7 |
| | 111 | COMP2_N2 | ADC2_IN1 |
| C3CR_INMSEL | 110 | COMP3_N1 | ADC3_IN8 |
| | 111 | COMP3_N2 | - |
| C4CR_INMSEL | 110 | COMP4_N1 | ADC3_IN9 |
| | 111 | COMP4_N2 | - |

**（2）当 COMP 切换引脚时**

当 COMP 的 P 端或 N 端切换引脚会产生电压变化，此时 COMP 会被影响而导致误触发。

当需要切换引脚时，COMP 分别提供了硬件和软件两种遮蔽方式。

硬件遮蔽默认开启，软件遮蔽需要额外配置，具体操作如下：

**硬件遮蔽：**

- COMP 的通道切换默认硬件遮蔽。即 COMP_CxCR 寄存器的 CH_BLANK_EN 位默认为 1

**软件遮蔽：**

- 置位 COMP_CxCR 寄存器的 CH_DIS_BLANKING 位来打开遮蔽
- 置位 COMP_CxCR 寄存器的 CH_DIS 位来关闭 COMP 的 P 和 N 通道
- 切换 P 或 N 通道
- 清零 COMP_CxCR 寄存器的 CH_DIS 位来开启 COMP 的 P 和 N 通道
- 清零 COMP_CxCR 寄存器的 CH_DIS_BLANKING 位来关闭遮蔽

### 19.3.6 COMP 迟滞

该比较器具有可编程迟滞，可在有噪声信号时避免发生意外输出转换。它是对称的，既作用于比较器输出的下降沿，又作用于比较器输出的上升沿。迟滞可在不需要时（例如，退出低功耗模式时）禁止，以便使用外部组件强制迟滞值。

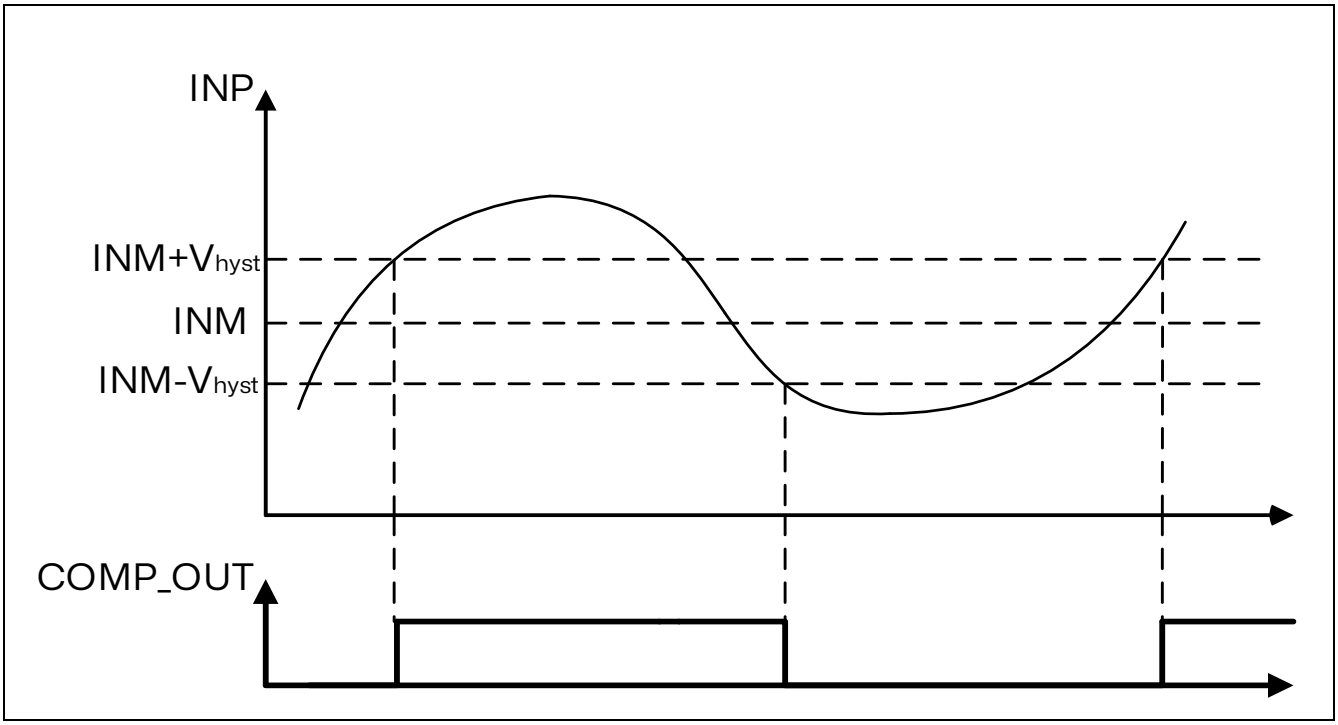



图 19.2 比较器迟滞

### 19.3.7 COMP 输出遮蔽功能

消除功能的目的是防止电流调节在 PWM 周期开始时（通常是功率开关反并联二极管中的恢复电流）出现短时尖峰时跳闸。 这里通过设置一个带有定时器输出比较信号的遮蔽窗来实现的。每个比较器通道有软件通过对应的 COMP_CxCR 寄存器的 BLANKING[2:0]位域单独选择遮蔽源，如下表：遮蔽源所示。

表 19.5 遮蔽源

| BLANKSEL[2:0] | COMP1 | COMP2 | COMP3 | COMP4 |
|---|---|---|---|---|
| 001 | TIM1_OC5 | TIM1_OC5 | TIM1_OC5 | TIM1_OC5 |
| 010 | TIM2_OC3 | TIM2_OC3 | TIM2_OC3 | TIM2_OC3 |
| 011 | TIM3_OC3 | TIM3_OC3 | TIM3_OC3 | TIM3_OC3 |
| 100 | TIM8_OC5 | TIM8_OC5 | TIM8_OC5 | TIM8_OC5 |
| 101 | TIM5_OC3 | TIM5_OC3 | TIM5_OC3 | TIM5_OC3 |
| 110 | TIM15_OC1 | TIM15_OC1 | TIM15_OC1 | TIM15_OC1 |
| 111 | TIM4_OC3 | TIM4_OC3 | TIM4_OC3 | TIM4_OC3 |

对遮蔽信号进行取反后与比较器级联输出进行逻辑与操作，以产生比较器通道 x 的输出。请参考下图中提供的示例。

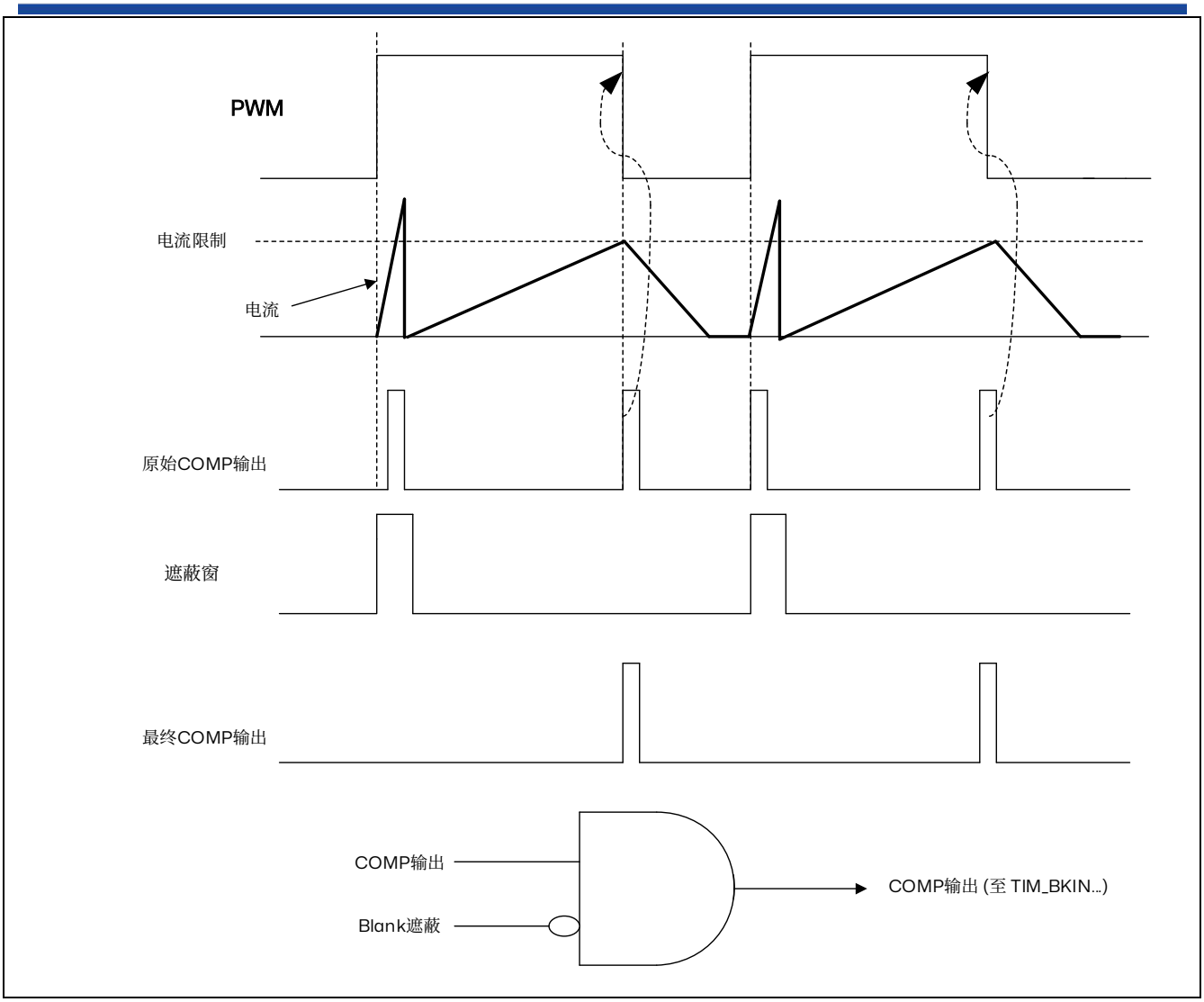



图 19.3 比较器输出遮蔽图

### 19.3.8 COMP 低功耗模式

表 19.6 比较器在低功耗模式下的行为

| 模式 | 描述 |
|---|---|
| Sleep | 对比较器没有影响。<br>比较器中断可使器件退出睡眠模式。 |
| Stop | 对比较器没有影响。<br>比较器中断可使器件退出停止模式。 |
| Standby | 导致比较器寄存器断电。<br>退出待机模式后必须重新初始化。 |

### 19.3.9 COMP 中断

比较器输出从内部连接到扩展中断和事件控制器。每个比较器都具有其各自的 EXTI 线，能够产生中断或事件，相同的机制用于退出睡眠和停止的低功耗模式。

要使能 COMPx 中断，需要遵循以下顺序：

1. 将 EXTI 线配置为中断模式，选择上升沿、下降沿或任一边沿有效，然后使能与 COMPx 输出事件对应的 EXTI 线。
2. 配置并使能映射到相应 EXTI 线的中断通道。
3. 使能 COMPx。

中断事件通过 COMP_CxCR 标志进行标记。

## 19.4 COMP 寄存器描述

### 19.4.1 比较器 x 控制和状态寄存器（COMP_CxCR）（x=1/2/3/4）

地址偏移：0x000+0x04*(x-1)

复位值：0x0000 0002

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | OUT | LIF | FIF | RIF | HIF | FHYST[1:0] | | LIE | BRGEN | BLANKING[2:0] | | | HIE | RHYST[1:0] | |
| rw | r | rc_w0 | rc_w0 | rc_w0 | rc_w0 | 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 | OFLT[2:0] | | | CH_DIS_BLANKING | CH_DIS | BAS_EN | INPSEL | Res | INMSEL[2:0] | | | FIE | RIE | CH_BLANK_EN | EN |
| rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **LOCK：**COMP_CxCR 寄存器锁定

该位由软件设置和清除。此位设置后，除了 CH_BALNK_EN、INPSEL、POL、CH_DIS、CH_DIS_BLANKING、RIF、FIF、HIF、LIF、RIE、FIE、HIE、LIE 以外，CxCR[31:0]的其他位只能读（对应的 CAL 寄存器也会锁上，只有 APB2RSTR_SYSCFGRST 置 1 可解除 lock ）

0：解锁，除了 OUT 只读以外，CxCR[31:0]的其他位能读/写

1：锁定

Bit 30 **OUT：**比较器 x 输出状态

只读标志反映了在极性选择器和遮蔽之前的比较器 x 的输出电平

0：低输出

1：高输出

Bit 29 **LIF：**低电平输出中断事件标志位

0：无低电平输出中断事件

1：低电平输出中断事件发生

Bit 28 **FIF：**下降沿中断事件标志位

0：无下降沿中断事件

1：下降沿中断事件发生

Bit 27 **RIF：**上升沿中断事件标志位

0：无上升沿中断事件

1：上升沿中断事件发生

Bit 26 **HIF：**高电平输出中断事件标志位

0：无高电平输出中断事件

1：高电平输出中断事件发生

Bit 25:24 **FHYST[1:0]：**下降迟滞电压

00：0mv

01：20mv

10：40mv
11：60mv

Bit 23 **LIE**：低电平输出中断使能
0：禁止低电平输出中断
1：使能低电平输出中断

Bit 22 **BRGEN**：$V_{REFINT}$ 标量电阻桥使能
此软件控制的位使 $V_{REFINT}$ 标量电阻桥能够工作。若要禁用电阻桥，必须将所有 COMP_CxCR 寄存器的 BRGEN 位设置为禁用状态。当电阻桥被禁用时，输入选择器的 1/4 $V_{REFINT}$、1/2 $V_{REFINT}$ 和 3/4 $V_{REFINT}$ 输入将不再获得 $V_{REFINT}$ 电阻桥分压后的电压，而是直接接收到 $V_{REFINT}$ 电压。
0：禁止
1：使能

Bit 21:19 **BLANKING[2:0]**：比较器 x 遮蔽源选择
此位域由软件控制，为比较器通道 x 选择遮蔽源，详见章节：COMP 输出遮蔽功能。
000：无遮蔽
001：TIM1 OC5 被选为遮蔽源
010：TIM2 OC3 被选为遮蔽源
011：TIM3 OC3 被选为遮蔽源
100：TIM8 OC5 被选为遮蔽源
101：TIM5 OC3 被选为遮蔽源
110：TIM15 OC1 被选为遮蔽源
111：TIM4 OC3 被选为遮蔽源

Bit 18 **HIE**：高电平输出中断使能
0：禁止高电平输出中断
1：使能高电平输出中断

Bit 17:16 **RHYST[1:0]**：上升迟滞电压
00：0mv
01：20mv
10：40mv
11：60mv

Bit 15 **POL**：比较器输出极性
此位由软件选择控制比较器 x 的输出极性
0：同向输出
1：反向输出

Bit 14:12 **OFLT[2:0]**：输出滤波，连续的 PCLK 时钟比较输出不变则认为有效，否则保持不变
000：无滤波　　100：128 个时钟周期
001：16 个时钟周期　　101：256 个时钟周期
010：32 个时钟周期　　110：512 个时钟周期
011：64 个时钟周期　　111：1024 个时钟周期
*注：*COMP *输出滤波没有作用于刹车和外部中断*，详情请参见 COMP 框图

Bit 11 **CH_DIS_BLANKING**：切换通道前进行 BLANKING 遮蔽
0：通道切换前不开启遮蔽
1：通道切换前开启遮蔽

Bit 10 **CH_DIS**：P 端或 N 端通道切换时软件主动关闭通道

0：通道切换时不主动关闭通道

1：通道切换时软件主动关闭通道，避免切换引脚造成 COMP 误触发

Bit 9 **BAS_EN**：ADC 采样的遮蔽使能

0：禁止遮蔽 ADC 采样

1：使能遮蔽 ADC 采样（参见章节：COMP 防误触发功能）

Bit 8 **INPSEL**：比较器 x 正向输入信号选择

此位由软件控制来选择比较器 x 的正向输入 COMPx_INP 的信号，详情参见表：

COMPx 正向输入分配

0：COMPx_P1

1： COMPx_P2

Bit 7 保留，必须保持为复位值。

Bit 6:4 **INMSEL[2:0]**：比较器 x 反向输入信号选择

此位域由软件控制来选择比较器 x 的反向输入 COMPx_INM 的信号，详情参见表：

COMPx 反向输入分配

000：1/4 $V_{REFINT}$

001：1/2 $V_{REFINT}$

010：3/4 $V_{REFINT}$

011：$V_{REFINT}$

100：DAC3_CH1(COMP1) DAC3_CH2(COMP2)
DAC3_CH1(COMP3) DAC3_CH2(COMP4)

101：DAC1_CH1(COMP1) DAC1_CH2(COMP2)
DAC2_CH1(COMP3) DAC2_CH2(COMP4)

110：COMPx_N1

111：COMPx_N2

*注：当选择* DAC *通道内部连接至* COMP *反向输入端时，需要关闭* DAC *输出缓冲器。*

Bit 3 **FIE**：下降沿中断使能

0：禁止下降沿中断

1：使能下降沿中断

Bit 2 **RIE**：上升沿中断使能

0：禁止上升沿中断

1：使能上升沿中断

Bit 1 **CH_BLANK_EN**：硬件切换 P 端和 N 端通道遮蔽

由硬件默认为 1 开启 P 端和 N 端通道切换遮蔽，可软件关闭。

具体参考章节：比较器防误触发功能。

0：关闭硬件遮蔽

1：使能硬件遮蔽

Bit 0 **EN**：比较器 x 使能

0：禁止

1：使能

*注：比较器的时钟由* RCC_APB2ENR *的* SYSCFGEN *位控制。*

### 19.4.2 比较器 x 校正寄存器（COMP_RGxCAL）（x=1/2/3/4）

地址偏移：0x040+0x04*(x-1)

复位值：0x0000 0808

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | NADJ[3:0] | | | | Reserved | | NEN | PEN | PADJ[3:0] | | | |
| | | | | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw |

Bit 31:12 保留，必须保持为复位值。

Bit 11:8 **NADJ[3:0]：**比较器 NMOS offset 偏置调节

Bit 7:6 保留，必须保持为复位值。

Bit 5 **NEN：**比较器 NMOS 校正使能位

0：COMP NMOS 校正禁止

1：COMP NMOS 校正使能

Bit 4 **PEN：**比较器 PMOS 校正使能位

0：COMP PMOS 校正禁止

1：COMP PMOS 校正使能

Bit 3:0 **PADJ[3:0]：**比较器 PMOS offset 偏置调节

# 20 运算放大器（OPAMP）

## 20.1 简介

该设备内置了四个运算放大器，每个放大器具有两个输入端和一个输出端。这三个输入/输出端（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为跟随器、非反相增益范围为 4 至 32 的放大器。

## 20.2 OPAMP 主要特性

- 轨对轨输入电压范围
- 低输入偏置电流
- 低输入偏置电压
- 高频增益带宽
- 高速模式，以实现更高的转换速率

## 20.3 OPAMP 功能描述

OPAMP 具有多种模式。
每个 OPAMP 可以单独使能，当禁用时，其输出为高阻抗状态。
使能后，OPAMP 可以处于校准模式，此时其所有输入和输出均断开连接；或者处于功能模式。
在功能模式下，OPAMP 的输入和输出按照章节：信号路由中的描述进行连接。

### 20.3.1 OPAMP 输出重定向至内部 ADC 通道

通过设置 OPAMPx_CSR 寄存器中的 TOINT_EN 位，可以将运算放大器的输出在内部重定向至一个 ADC 通道。在这种情况下，原本映射给特定 OPAMPx_OUT 输出的 GPIO 引脚将被释放，可以用于其他用途。ADC 可以在内部测量 OPAMPx_OUT 的电压。请参阅下表：运算放大器的可能连接方式，了解 OPAMPx 内部输出与 ADC 通道之间的分配关系。

### 20.3.2 OPAMP 复位和时钟

由时钟控制器提供的 OPAMP 时钟与 PCLK2（APB2 时钟）同步。在 RCC 控制器中没有提供时钟使能控制位。要使用 OPAMP 的时钟源，必须在 RCC 控制器中设置 SYSCFG 时钟使能控制位。

OPAEN 位用于使能和禁用 OPAMP 的操作。为了避免对输出产生伪影效应，必须在使能 OPAEN 位之前更改 OPAMP 的寄存器配置。

当不再需要 OPAMP 的输出时，可以禁用 OPAMP 以节省功率。在 OPAMP 被禁用期间，之前设置的所有配置（包括校准）都将保持不变。

### 20.3.3 初始化配置

OPAMP 的默认配置为功能模式，其中三个输入/输出端连接到外部引脚。在默认模式下，OPAMP 使用工厂校准的修调值进行偏移校准。修调值可以调整，详见章节：校准，以了解如何更改修调值。默认配置使用正常模式，提供标准性能。可以通过设置 OPAHSM 位，将 OPAMP 切换到高速模式，以获得更高的转换速率。正常模式和高速模式的特性均在数据手册中定义。

一旦设置 OPAMPx_CSR 寄存器中的 OPAEN 位，OPAMP 即开始工作。两个输入引脚和一个输出引脚的连接方式如章节：信号路由中所述，且默认连接设置可以更改。

*注：在对应的*GPIOx_MODER *寄存器中，输入和输出引脚必须配置为模拟模式（默认状态）。*

### 20.3.4 信号路由

运算放大器引脚的路由由 OPAMPx_CSR 和 OPAMPx_CR2 寄存器决定。
下表描述了四个运算放大器（OPAMPx，x = 1...4）的连接情况。

表 20.1 运算放大器的可能连接方式

| 信号 | 引脚 | 内部的 | 描述 |
|---|---|---|---|
| OPAMP1_VINM | PA3 (VINM_N1)<br>PC3 (VINM_N2)<br>PC1 (VINM_N3) | OPAMP1_OUT 或 PGA | 由位 GAIN 和 VM_SEL 控制 |
| OPAMP1_VINP | PA2 (VINP_P1)<br>PC0 (VINP_P2) | 外接偏置 $V_{REFINT}$/2 或 $V_{ref}$/2 或 GND | 引脚由位 VP_SEL 控制，偏置由位 VOL_SEL 和 VOL_EN 控制 |
| OPAMP1_OUT | PA4 | ADC1_IN11<br>ADC2_IN9(1) | 当 OPAMP 使能且 OPAMP 内部输出禁用时，该引脚被连接。<br>ADC 输入由 ADC 控制。 |
| OPAMP2_VINM | PA6 (VINM_N1)<br>PC4 (VINM_N2)<br>PE7 (VINM_N3) | OPAMP2_OUT 或 PGA | 由位 GAIN 和 VM_SEL 控制 |
| OPAMP2_VINP | PA5 (VINP_P1)<br>PE8 (VINP_P2) | 外接偏置 $V_{REFINT}$/2 或 $V_{ref}$/2 或 GND | 引脚由位 VP_SEL 控制，偏置由位 VOL_SEL 和 VOL_EN 控制 |
| OPAMP2_OUT | PA7 | ADC1_IN12<br>ADC2_IN10(1) | 当 OPAMP 使能且 OPAMP 内部输出禁用时，该引脚被连接。<br>ADC 输入由 ADC 控制。 |
| OPAMP3_VINM | PB1 (VINM_N1)<br>PC5 (VINM_N2)<br>PE9 (VINM_N3) | OPAMP3_OUT 或 PGA | 由位 GAIN 和 VM_SEL 控制 |
| OPAMP3_VINP | PB2 (VINP_P1)<br>PE10 (VINP_P2) | 外接偏置 $V_{REFINT}$/2 或 $V_{ref}$/2 或 GND | 引脚由位 VP_SEL 控制，偏置由位 VOL_SEL 和 VOL_EN 控制 |
| OPAMP3_OUT | PB0 | ADC2_IN11<br>ADC3_IN11(1) | 当 OPAMP 使能且 OPAMP 内部输出禁用时，该引脚被连接。<br>ADC 输入由 ADC 控制。 |
| OPAMP4_VINM | PB11 (VINM_N1)<br>PD8 (VINM_N2)<br>PE11 (VINM_N3) | OPAMP4_OUT 或 PGA | 由位 GAIN 和 VM_SEL 控制 |
| OPAMP4_VINP | PB10 (VINP_P1)<br>PE12 (VINP_P2) | 外接偏置 $V_{REFINT}$/2 或 $V_{ref}$/2 或 GND | 引脚由位 VP_SEL 控制，偏置由位 VOL_SEL 和 VOL_EN 控制 |
| OPAMP4_OUT | PB12 | ADC2_IN12<br>ADC3_IN12(1) | 当 OPAMP 使能且 OPAMP 内部输出禁用时，该引脚被连接。<br>ADC 输入由 ADC 控制。 |

1. 当设置 TOINT_EN 位时，此 ADC 通道在内部连接到 OPAMPx_OUT。在这种情况下，原本用于 OPAMPx_OUT 的 I/O 可以用于其他用途。

### 20.3.5 OPAMP 模式

运算放大器的输入和输出端均可通过引脚访问。这些放大器可在多种配置环境中使用：

- 独立模式（外部增益设置模式）
- 电压跟随器模式
- PGA 模式

*注：必须将信号阻抗保持在一定水平以下，以避免输入泄漏产生明显伪影（由于源端的电阻压降）。有关详细信息，请参阅数据手册中的电气特性部分。*

**独立模式（外部增益设置模式）**

以下是配置 OPAMP 作为独立模式使用的**必要步骤**：

1. 独立模式 P 端选择：
   - VINP_P1：OPAMPx_CSR 寄存器 VP_SEL=0
   - VINP_P2：OPAMPx_CSR 寄存器 VP_SEL=1
2. 独立模式 N 端选择：
   - VINM_N1：OPAMPx_CSR 寄存器 VM_SEL[4:0]=00001
   - VINM_N2：OPAMPx_CSR 寄存器 VM_SEL[4:0]=00010
   - VINM_N3：OPAMPx_CSR 寄存器 VM_SEL[4:0]=00100
3. OPAMP 输出选择：
   - 输出至外部 GPIO：OPAMPx_CSR 寄存器 TOEXT_EN=1
4. OPAMP 使能：
   - OPAMPx_CSR 寄存器 OPAEN=1

*注：可将*OPAHSM *设置为“运算放大器高速”模式，以获得高转换速率。*

*注：可设置*USERTRIM *以修改输入失调的校准值。*

*注：下图中示例的独立模式设置的外部电阻增益* = (R1+R2)/R1

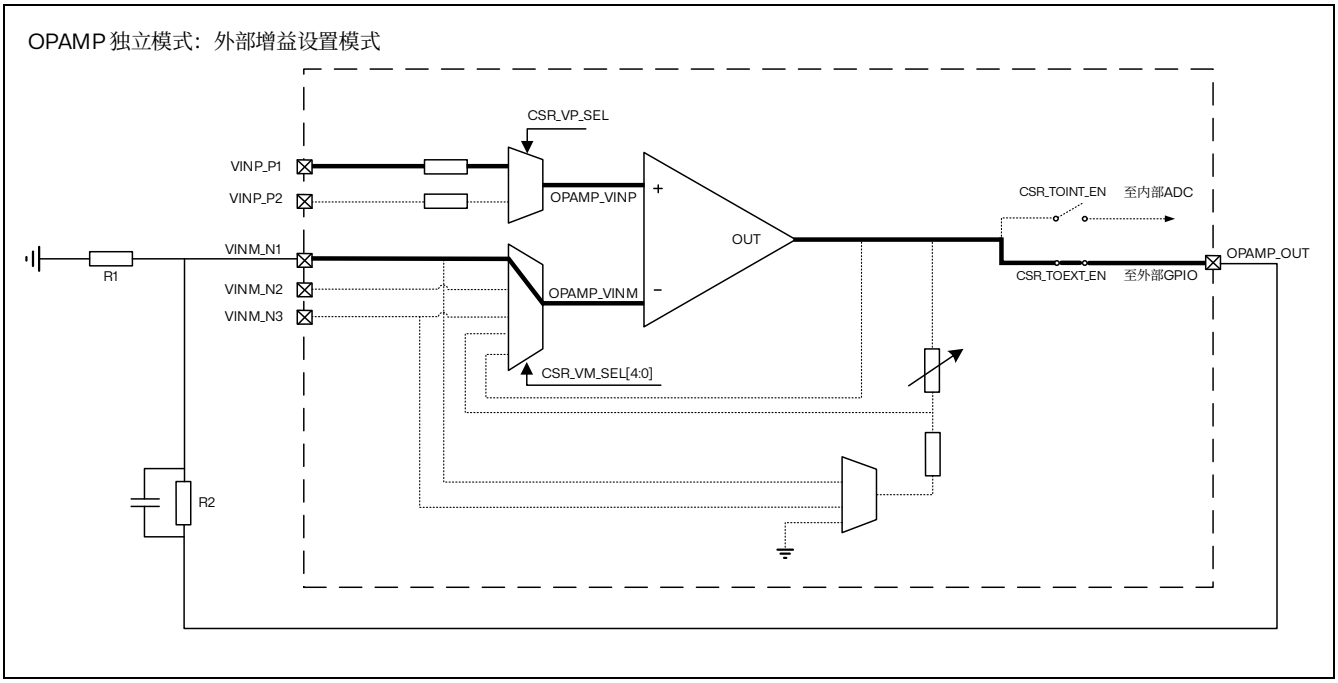



图 20.1 独立模式：外部增益设置模式

**电压跟随器模式配置方式**

以下是配置 OPAMP 作为电压跟随器模式使用的**必要步骤**：

1. OPAMPx 模式选择：
   - 电压跟随器模式：OPAMPx_CSR 寄存器 VM_SEL[4:0]=10000
2. 电压跟随器模式 P 端选择：
   - VINP_P1：OPAMPx_CSR 寄存器 VP_SEL=0
   - VINP_P2：OPAMPx_CSR 寄存器 VP_SEL=1

   *注：与*OPAMPx_VINM *对应的引脚可用于其他用途。*
3. OPAMP 输出选择：
   - 输出至内部 ADC：OPAMPx_CSR 寄存器 TOINT_EN=1
   - 输出至外部 GPIO：OPAMPx_CSR 寄存器 TOEXT_EN=1

   *注：若选择输出至内部*ADC *通道，外部映射有*OPAMPx_OUT *的*I/O *口空闲，可用于其他用途。*
4. OPAMP 使能：
   - OPAMPx_CSR 寄存器 OPAEN=1

   *注：一旦设置*OPAEN *位，*OPAMPx_VINP *引脚上的电压就会被缓冲至*OPAMPx_OUT *引脚。*

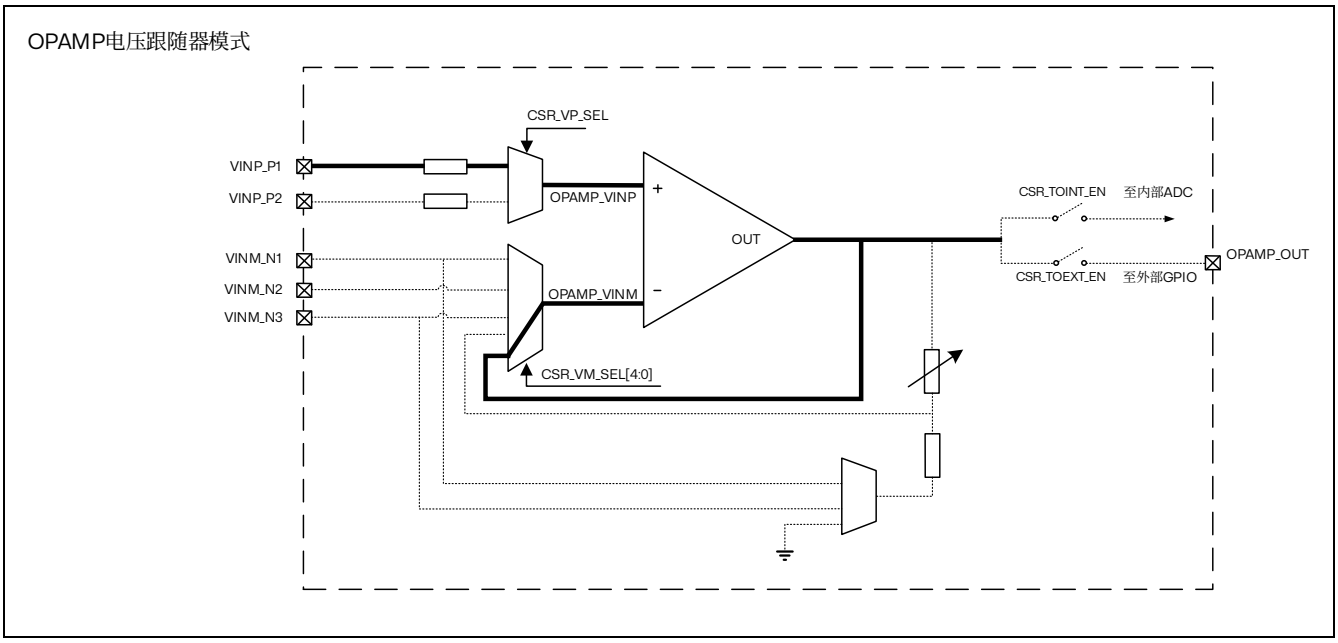



图 20.2 电压跟随器模式

### PGA 模式

以下是配置 OPAMP 作为 PGA 模式使用的**必要步骤**:

1. OPAMPx 模式选择:
   - PGA 模式：OPAMPx_CSR 寄存器 VM_SEL[4:0]=01000
2. PGA 模式 P 端选择:
   - VINP_P1：OPAMPx_CSR 寄存器 VP_SEL=0
   - VINP_P2：OPAMPx_CSR 寄存器 VP_SEL=1
3. PGA 模式 P 端内部偏置电压选择（不选默认偏置 GND 即无偏置电压）:
   - 偏置 $V_{REFINT}/2$：OPAMPx_CR2 寄存器 VOL_EN=1，VOL_SEL=0，VREFINT_EN=1
   - 偏置 $V_{ref}/2$：OPAMPx_CR2 寄存器 VOL_EN=1，VOL_SEL=1
   - 偏置 GND：OPAMPx_CR2 寄存器 VOL_EN=0

   *注：* $V_{REFINT}$ 为内部稳定的带隙 1V 电压输出。

   *注：* $V_{ref}/2$ 由 VREFBUF 配置，可选连接到外部 VREF+引脚，或连接内部 2.0V/2.5V/2.8V。

   *注：OPAMP1 和 OPAMP2 共用一个偏置电压，VOL_SEL 位在 OPAMP1_CR2 寄存器设置。*

   *注：OPAMP3 和 OPAMP4 共用一个偏置电压，VOL_SEL 位在 OPAMP3_CR2 寄存器设置。*
4. PGA 模式 N 端选择:
   - 内部接地（单端）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=01
   - VINM_N1（差分）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=10
   - VINM_N3（差分）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=11

   *注：VINM_N2 在 PGA 模式仅用于滤波，无法作为 PGA 模式 N 端的输入。*
5. PGA 模式增益放大倍数选择:
   - 4X：OPAMPx_CSR 寄存器 GAIN[2:0]=000
   - 8X：OPAMPx_CSR 寄存器 GAIN[2:0]=001
   - 12X：OPAMPx_CSR 寄存器 GAIN[2:0]=010
   - 16X：OPAMPx_CSR 寄存器 GAIN[2:0]=011
   - 24X：OPAMPx_CSR 寄存器 GAIN[2:0]=100
   - 32X：OPAMPx_CSR 寄存器 GAIN[2:0]=101

   *注：为避免饱和，输入电压必须保持在 VDDA 除以所选增益所得的电压值以下。*
6. OPAMP 输出选择:
   - 输出至内部 ADC：OPAMPx_CSR 寄存器 TOINT_EN=1
   - 输出至外部 GPIO：OPAMPx_CSR 寄存器 TOEXT_EN=1
7. OPAMP 使能:
   - OPAMPx_CSR 寄存器 OPAEN=1

   *注：一旦设置 OPAEN 位，OPAMPx_VINP 引脚上的电压就会按所选增益进行放大。*

   *注：PGA 模式输出 = （VINP - VINM）* GAIN + 偏置电压*

**例 1：PGA 模式下 N 端选择内部接地，此时 P 端作为单端输入，偏置 GND。**

– PGA 模式输出 = （VINP - 0）* GAIN + 0 = VINP * GAIN

**例 2：PGA 模式下 N 端选择 VINM_N1，此时 P 端 N 端作为差分输入，偏置 $V_{REFINT}/2$。**

– PGA 模式输出 = （VINP - VINM）* GAIN + $V_{REFINT}/2$

OPAMP PGA模式：**例1 N端选择内部接地，此时P端作为单端输入，偏置GND。**

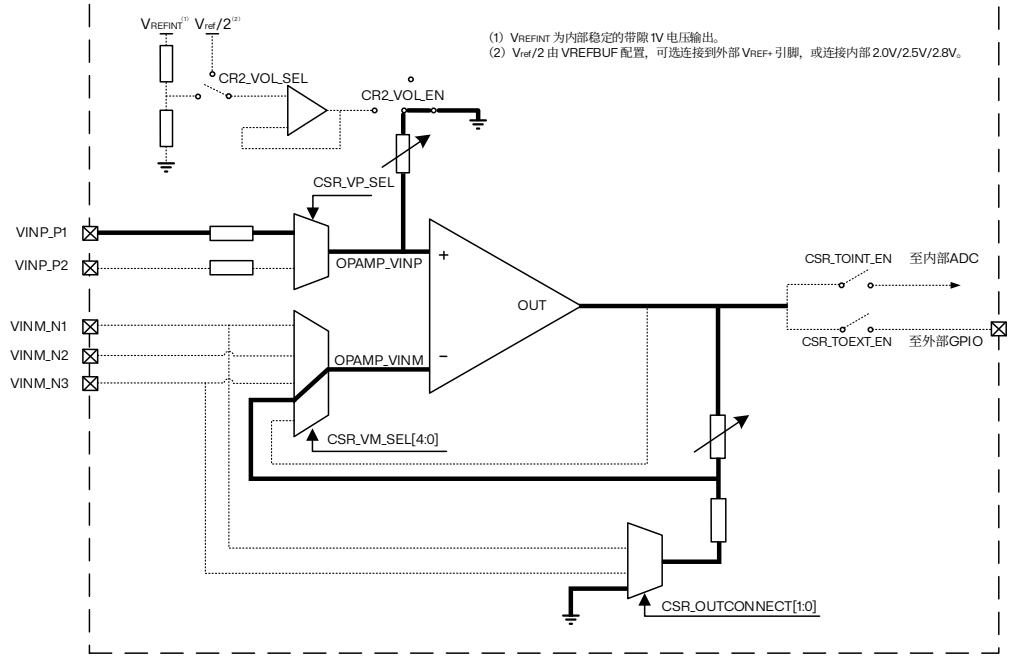



图 20.3 PGA 模式单端输入（P 端无偏置电压）

OPAMP PGA模式：**例2 N端选择VINM_N1，此时P端N端作为差分输入，偏置$V_{REFINT}/2$。**

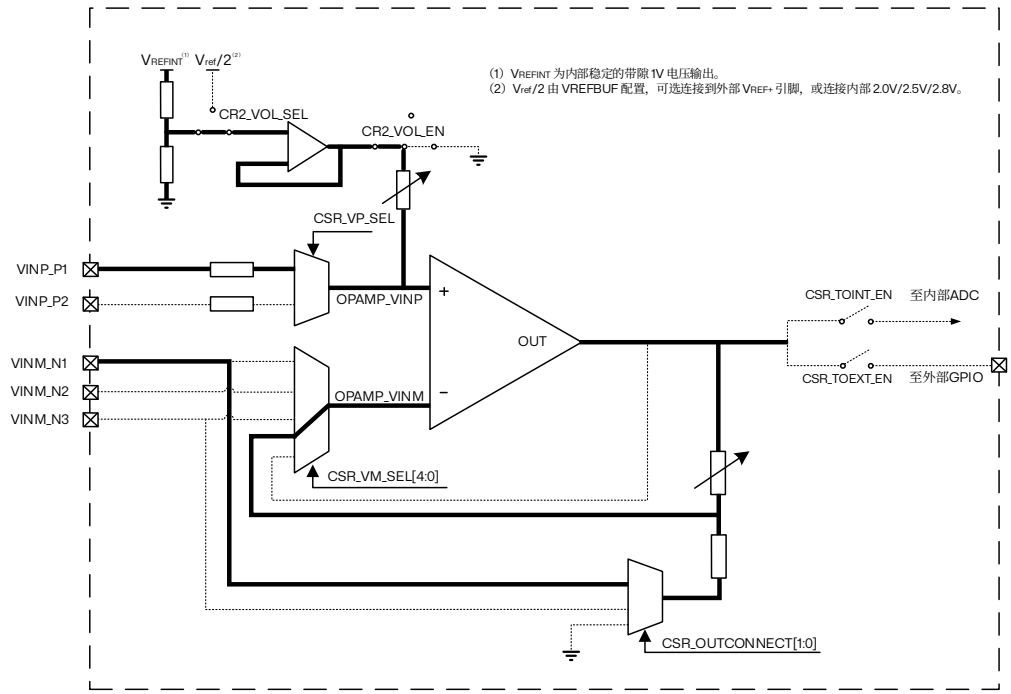



图 20.4 PGA 模式差分输入（P 端接偏置电压）

**带外部滤波的 PGA 模式**

以下是配置 OPAMP 作为 PGA 模式对输入信号幅度进行放大并采用外部滤波的**必要步骤**:

1. OPAMPx 模式选择（PGA 模式下 VINM_N2 连接外部滤波）:
   - 带外部滤波的 PGA 模式：OPAMPx_CSR 寄存器 VM_SEL[4:0]=01010
2. 带外部滤波的 PGA 模式 P 端选择:
   - VINP_P1：OPAMPx_CSR 寄存器 VP_SEL=0
   - VINP_P2：OPAMPx_CSR 寄存器 VP_SEL=1
3. 带外部滤波的 PGA 模式 P 端内部偏置电压选择（不选默认偏置 GND 即无偏置电压）:
   - 偏置 $V_{REFINT}/2$：OPAMPx_CR2 寄存器 VOL_EN=1，VOL_SEL=0，VREFINT_EN=1
   - 偏置 $V_{ref}/2$：OPAMPx_CR2 寄存器 VOL_EN=1，VOL_SEL=1
   - 偏置 GND：OPAMPx_CR2 寄存器 VOL_EN=0

   *注：* $V_{REFINT}$ 为内部稳定的带隙 1V 电压输出。

   *注：* $V_{ref}/2$ 由 VREFBUF 配置，可选连接到外部 VREF+引脚，或连接内部 2.0V/2.5V/2.8V。

   *注：OPAMP1 和 OPAMP2 共用一个偏置电压，VOL_SEL 位在 OPAMP1_CR2 寄存器设置。*

   *注：OPAMP3 和 OPAMP4 共用一个偏置电压，VOL_SEL 位在 OPAMP3_CR2 寄存器设置。*
4. PGA 模式 N 端选择:
   - 内部接地（单端）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=01
   - VINM_N1（差分）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=10
   - VINM_N3（差分）：OPAMPx_CSR 寄存器 OUTCONNECT[1:0]=11

   *注：VINM_N2 在 PGA 模式仅用于滤波，无法作为 PGA 模式 N 端的输入。*
5. PGA 模式增益放大倍数选择:
   - 4X：OPAMPx_CSR 寄存器 GAIN[2:0]=000
   - 8X：OPAMPx_CSR 寄存器 GAIN[2:0]=001
   - 12X：OPAMPx_CSR 寄存器 GAIN[2:0]=010
   - 16X：OPAMPx_CSR 寄存器 GAIN[2:0]=011
   - 24X：OPAMPx_CSR 寄存器 GAIN[2:0]=100
   - 32X：OPAMPx_CSR 寄存器 GAIN[2:0]=101

   *注：为避免饱和，输入电压必须保持在 VDDA 除以所选增益所得的电压值以下。*
6. OPAMP 输出配置:
   - 输出至内部 ADC 和外部 GPIO：OPAMPx_CSR 寄存器 TOINT_EN=1 且 TOEXT_EN=1
7. OPAMP 使能:
   - OPAMPx_CSR 寄存器 OPAEN=1

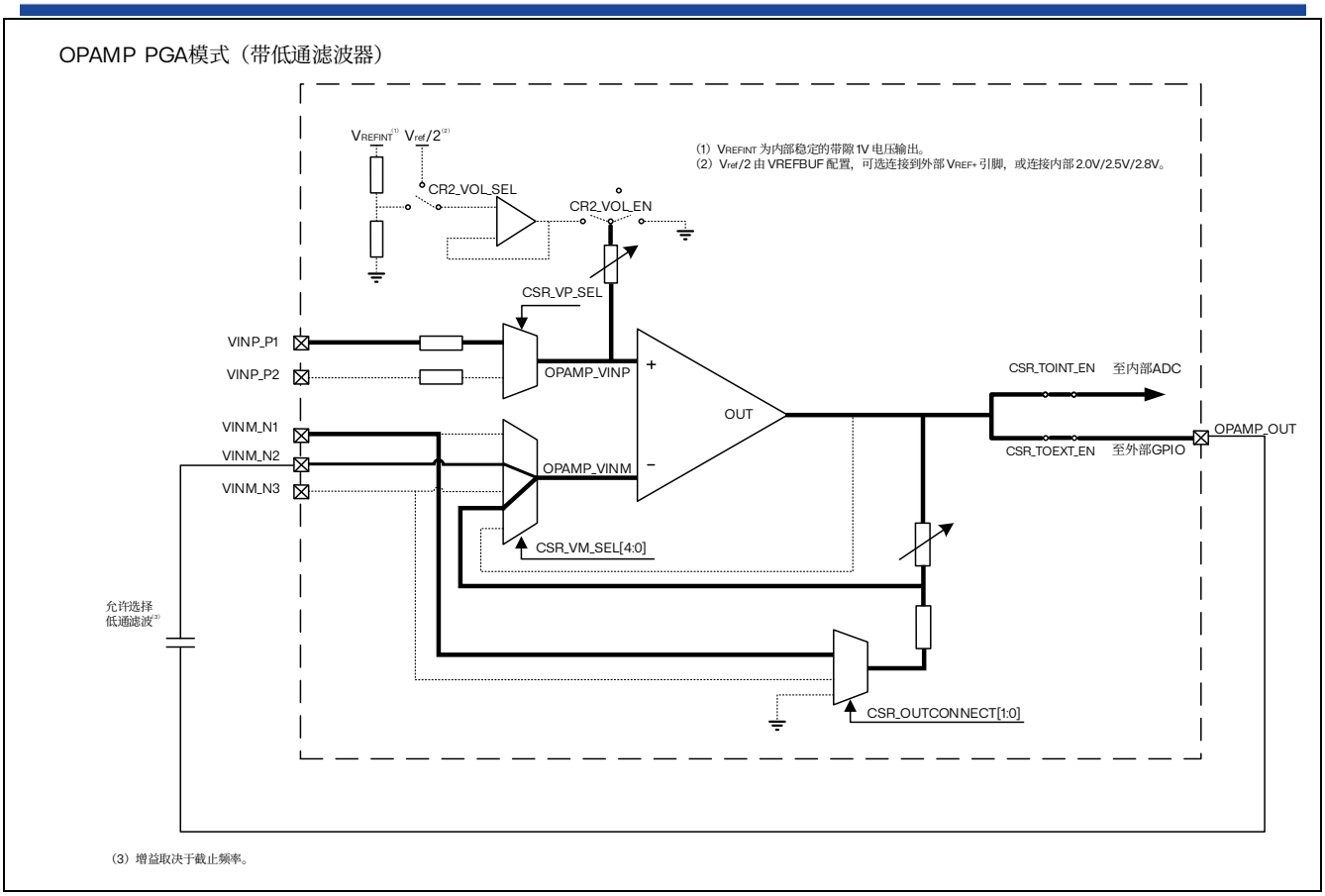



图 20.5 PGA 模式（VINM_N2 用于滤波）

*注：VINM_N2 上的任何外部连接均可与内部可编程增益放大器（PGA）并联使用，例如，可在 OPAMP_OUT 和 VINM_N2 之间连接一个电容以实现滤波目的（PGA 电阻网络中使用的电阻值请参见数据手册）。*

*注：根据目标截止频率 $f_c$ 和电阻 $R$ 可计算所需电容：$C = \frac{1}{2\pi R f_c}$*

*注：增益取决于截止频率。*

### 20.3.6 校准

OPAMP 的偏移值通过使用微调电路来最小化。在启动时，微调值会被初始化为预设的“工厂”微调值。如果 OPAMP 的使用条件与工厂微调条件不同，用户也可以对每个运算放大器进行微调。

每个运算放大器都可由用户进行微调。特定的寄存器允许为正常模式和高速模式设置不同的微调值。

校准的目的是尽可能地消除 OPAMP 输入偏置电压。校准电路允许在稳定的电压和温度条件下将输入失调电压降低。

对于每个运算放大器，需要微调两个值（OPAMPx_CSR 寄存器中的 TRIMOFFSETP 和 TRIMOFFSETN），一个用于 N 端，一个用于 P 端。

用户可以通过设置 OPAMPx_CSR 寄存器中的 USERTRIM 位，从“工厂”值切换到“用户”微调值。此位在启动时复位，因此“工厂”值会默认应用于 OPAMP 选项寄存器。

偏移微调位 TRIMOFFSETP 和 TRIMOFFSETN 通常在校准操作初始化（即将 CALON 位设置为 1）后进行配置。当 CALON=1 时，运算放大器的输入会与 I/O 断开连接。

- 将 OPAMPx_CSR 寄存器 CALSEL 设置为 01 初始化 P 端的偏移校准（使用低电压参考）。
- 将 OPAMPx_CSR 寄存器 CALSEL 设置为 11 初始化 N 端的偏移校准（使用高压参考）。

当 CALON=1 时，CALOUT 位反映了由 CALSEL 和 OPAHSM 选择的微调值的影响。软件必须在 OPAMP 控制寄存器中将 TRIMOFFSETN 位从 0x00 增加到第一个使 CALOUT 位从 1 变为 0 的值。如果 CALOUT 位被复位，则偏移已正确校准，并且必须存储相应的微调值。在微调值改变后，CALOUT 标志需要最多 2ms 才能稳定。

*注：微调值越接近最佳微调值，稳定所需的时间越长（在任何情况下，最大稳定时间都不会超过 2ms）。*

表 20.2 工作模式及校正

| 模式 | 控制位 | | | | 输出 | |
|---|---|---|---|---|---|---|
| | OPAEN | OPAHSM | CALON | CALSEL | Vout | CALOUT 标志 |
| 正常模式 | 1 | 0 | 0 | X | 模拟 | 0 |
| 高速模式 | 1 | 1 | 0 | X | 模拟 | 0 |
| 偏移校准 N | 1 | X | 1 | 11 | 模拟 | X |
| 偏移校准 P diff | 1 | X | 1 | 01 | 模拟 | X |

**校准程序**

以下是对任意一个运算放大器进行完整校准的步骤：

1. 将 OPAMPx_CSR 寄存器中的 OPAEN 位设置为 1，以使能运算放大器。
2. 将 OPAMPx_CSR 寄存器中的 USERTRIM 位设置为 1。
3. 选择校准模式（请参阅上表：工作模式和校准）。步骤 3 至 4 需要重复四次。在第一次迭代中，选择正常模式和 N 差分对。此校准模式对应于 OPAMPx_CSR 寄存器中的 OPAHSM = 0 和 CALSEL = 11。
4. 从 0b00000 开始，在 OPAMPx_CSR 中递增 TRIMOFFSETN[4:0]，直到 OPAMPx_CSR 中的 CALOUT 变为 0。

换向意味偏移已正确补偿，必须将相应的微调代码保存在 TRIMOFFSETP 和 TRIMOFFSETN 位中。

重复步骤 3 至 4，以进行以下校准：

- 正常模式和 P 差分对，CALSEL = 01
- 高速模式和 N 差分对
- 高速模式和 P 差分对

如果某种模式不使用，则无需执行相应的校准。

所有运算放大器可以同时进行校准。

*注：在整个校准阶段，运算放大器输出的外部连接不得拉高或拉低超过* 500 µA *的电流。*

如果 OPAMP 输出在内部连接到 ADC 通道，并且与输出引脚断开连接（OPAMPx_CSR 寄存器中的 TOINT_EN= 1，TOEXT_EN = 0），则偏移微调程序与运算放大器输出连接到输出引脚（TOINT_EN = 0，TOEXT_EN = 1）的情况不同。校准程序与上述相似，但步骤 4 中指示的 CALOUT 位变化检测无法使用。相反，必须使用 ADC 输出数据作为指示器来检测运算放大器输出的变化：CALOUT 从 1 变为 0 对应于 ADC 输出数据从接近最大 ADC 输出的值变为接近最小 ADC 输出的值（ADC 作为连接到 OPAMP 输出的比较器工作）。另一种解决方案是在 TOINT_EN = 0，TOEXT_EN = 1 的情况下进行校准，然后将 TOINT_EN 更改为 1，TOEXT_EN 更改为 0。在这种情况下，校准期间 OPAMP 输出 GPIO 会切换，必须注意该 GPIO 上没有冲突。

### 20.3.7 OPAMP 低功耗模式

表 20.3 低功耗模式对 OPAMP 的影响

| 模式 | 描述 |
|---|---|
| Sleep | 无影响。 |
| Stop 0 / Stop 1 | 无影响，OPAMP 寄存器内容保留。 |
| Standby | OPAMP 寄存器断电，必须在退出待机模式后重新初始化。 |

## 20.4 OPAMP 寄存器描述

### 20.4.1 OPAMPx 控制状态寄存器（OPAMPx_CSR）（x=1...4）

地址偏移：0x00 + 4*(x-1)

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCK | CAL OUT | TOINT_ EN | TRIMOFFSETP[4:0] | | | | | TRIMOFFSETN[4:0] | | | | | OUTCONNECT [1:0] | | GAIN[2] |
| rw | r | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GAIN[1:0] | | CALSEL[1:0] | | CALON | USER TRIM | VP_SEL | TOEXT_ EN | OPA HSM | VM_SEL[4:0] | | | | | FORCE_ VP | OPAEN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **LOCK**：OPAMPx_CSR 锁定

此位为一次性写入位。由软件设置。只能通过系统复位清除。

此位用于将 OPAMPx_CSR 寄存器配置为只读。

0：OPAMPx_CSR 可读写

1：OPAMPx_CSR 只读

Bit 30 **CALOUT**：OPAMPx 校准输出

校准模式期间，此信号翻转时，微调偏移。

Bit 29 **TOINT_EN**：OPAMPx 内部输出 ADC 使能

0：OPAMPx 输出至内部 ADC 禁止

1：OPAMPx 输出至内部 ADC 使能

Bit 28:24 **TRIMOFFSETP[4:0]**：校准 P 端

Bit 23:19 **TRIMOFFSETN[4:0]**：校准 N 端

在复位时，这些位由工厂微调值加载。仅当 USER_TRIM=1 时，它们才可以被修改。

Bit 18:17 **OUTCONNECT[1:0]**：PGA 模式下 OPAMPx 的输出连接

00：不连接

01：OPAMPx 的输出经过内部电阻接地

10：OPAMPx 的输出经过内部电阻接到 OPAMPx_N1

11：OPAMPx 的输出经过内部电阻接到 OPAMPx_N3

Bit 16:14 **GAIN[2:0]**：OPAMPx PGA 模式放大倍率选择

000：4x

001：8x

010：12x

011：16x

100：24x

101：32x

Bit 13:12 **CALSEL[1:0]**：OPAMPx 校准选择

00：0.033*VDDA 作为 OPAMPx 输入

01：0.1*VDDA 作为 OPAMPx 输入（作为 P 端校准）

10：0.5*VDDA 作为 OPAMPx 输入

11：0.9*VDDA 作为 OPAMPx 输入（作为 N 端校准）

Bit 11 **CALON**：OPAMPx 校准模式使能

0：正常模式

1：校准模式（通过硬件打开所有开关）

Bit 10 **USERTRIM**：用于从“出厂”OPAMPx 偏移微调值切换到“用户”OPAMPx 偏移微调值对于正常模式和低功耗模式，该位均有效。

0：使用“出厂”微调代码

1：使用“用户”微调代码

Bit 9 **VP_SEL**：非反相输入选择

0：OPAMPx_P1

1：OPAMPx_P2

Bit 8 **TOEXT_EN**：OPAMPx 输出至外部 IO 引脚使能

0：OPAMPx 输出至外部 IO 引脚禁止

1：OPAMPx 输出至外部 IO 引脚使能

Bit 7 **OPAHSM**：OPAMPx 高速模式使能

0：高速模式禁止

1：高速模式使能

Bit 6:2 **VM_SEL[4:0]**：反相输入选择

00001：OPAMPx_N1 引脚连接到 OPAMP VINM 输入（独立模式）

00010：OPAMPx_N2 引脚连接到 OPAMP VINM 输入（独立模式或 PGA 模式下反相输入用于滤波）

00100：OPAMPx_N3 引脚连接到 OPAMP VINM 输入（独立模式）

01000：反馈电阻连接到 OPAMP 的 VINM 输入（PGA 模式），PGA 模式放大倍率取决于 GAIN[2:0]的设置

10000：OPAMP_OUT 连接到 OPAMP VINM 输入（电压跟随器模式）

*注：PGA 模式时，OPAMPx 的反相输入选择由 OUTCONNECT[1:0]决定。*

Bit 1 **FORCE_VP**：强制将内部参考电压施加到运算放大器 VINP 输入（保留用于测试）

0：正常操作模式：非反相输入连接到输入。

1：校准验证模式：非反相输入连接到校准参考电压。

Bit 0 **OPAEN**：运算放大器使能

0：运算放大器禁止

1：运算放大器使能

### 20.4.2 OPAMP1 控制寄存器 2（OPAMP1_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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | VREFINT_EN | VOL_SEL | VOL_EN |
| | | | | | | | | | | | | | rw | rw | rw |

Bit 31:3 保留，必须保持为复位值。

Bit 2 **VREFINT_EN**：$V_{REFINT}$ 使能信号

0：禁止 $V_{REFINT}$

1：使能 $V_{REFINT}$

Bit 1 **VOL_SEL**：OPAMP1/2 P 端输入外接偏置电压选择

0：偏置电压选择为 $V_{REFINT}/2$

1：偏置电压选择为 $V_{ref}/2$

*注：仅 PGA 模式可用，使用时需要置位 OPAMP1/2_CR2 寄存器的 VOL_EN=1。*

*注：OPAMP1 和 OPAMP2 共用一个偏置电压。*

*注：当偏置电压选择为 $V_{REFINT}/2$，$V_{REFINT}$ 为内部稳定的带隙 1V 电压输出。需要置位 OPAMPx_CR2 寄存器的 VREFINT_EN 位来使能 $V_{REFINT}$ 信号。*

*注：当偏置电压选择为 $V_{ref}/2$ 时，$V_{ref}$ 电压由 VREFBUF 配置来选择，可选择连接到外部 $V_{REF+}$ 引脚，或选择连接内部参考电压缓冲器 2.0V/2.5V/2.8V。详见章节：参考电压缓冲器（VREFBUF）。*

Bit 0 **VOL_EN**：OPAMP1 内部 VOL 选择线

0：VOL 接地（AGND）

1：VOL 接偏置电压

*注：仅 PGA 模式可用，独立模式和电压跟随器模式下此位开关高阻。*

### 20.4.3 OPAMP2 控制寄存器 2（OPAMP2_CR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | VREFINT_EN | Res | VOL_EN |
| | | | | | | | | | | | | | rw | | rw |

Bit 31:3 保留，必须保持为复位值。

Bit 2 **VREFINT_EN：**$V_{REFINT}$ 使能信号

0：禁止 $V_{REFINT}$

1：使能 $V_{REFINT}$

Bit 1 保留，必须保持为复位值。

Bit 0 **VOL_EN：**OPAMP2 内部 VOL 选择线

0：VOL 接地（AGND）

1：VOL 接偏置电压

*注：仅 PGA 模式可用，独立模式和电压跟随器模式下此位开关高阻。*

### 20.4.4 OPAMP3 控制寄存器 2（OPAMP3_CR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | VREFINT_EN | VOL_SEL | VOL_EN |
| | | | | | | | | | | | | | rw | rw | rw |

Bit 31:3 保留，必须保持为复位值。

Bit 2 **VREFINT_EN**：$V_{REFINT}$ 使能信号

0：禁止 $V_{REFINT}$

1：使能 $V_{REFINT}$

Bit 1 **VOL_SEL**：OPAMP3/4 P 端输入外接偏置电压选择

0：偏置电压选择为 $V_{REFINT}/2$

1：偏置电压选择为 $V_{ref}/2$

*注：仅PGA 模式可用，使用时需要置位OPAMP3/4_CR2 寄存器的VOL_EN=1。*

*注：OPAMP3 和OPAMP4 共用一个偏置电压。*

*注：当偏置电压选择为$V_{REFINT}/2$，$V_{REFINT}$ 为内部稳定的带隙1V 电压输出。需要置位OPAMPx_CR2 寄存器的VREFINT_EN 位来使能 $V_{REFINT}$ 信号。*

*注：当偏置电压选择为$V_{ref}/2$ 时，$V_{ref}$ 电压由VREFBUF 配置来选择，可选择连接到外部 $V_{REF+}$ 引脚，或选择连接内部参考电压缓冲器2.0V/2.5V/2.8V。详见章节：参考电压缓冲器（VREFBUF）。*

Bit 0 **VOL_EN**：OPAMP3 内部 VOL 选择线

0：VOL 接地（AGND）

1：VOL 接偏置电压

*注：仅PGA 模式可用，独立模式和电压跟随器模式下此位开关高阻。*

### 20.4.5 OPAMP4 控制寄存器 2（OPAMP4_CR2）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | VREFINT_EN | Res | VOL_EN |
| | | | | | | | | | | | | | rw | | rw |

Bit 31:3 保留，必须保持为复位值。

Bit 2 **VREFINT_EN：**$V_{REFINT}$ 使能信号

0：禁止 $V_{REFINT}$

1：使能 $V_{REFINT}$

Bit 1 保留，必须保持为复位值。

Bit 0 **VOL_EN：**OPAMP4 内部 VOL 选择线

0：VOL 接地（AGND）

1：VOL 接偏置电压

*注：仅 PGA 模式可用，独立模式和电压跟随器模式下此位开关高阻。*

# 21 高分辨率定时器（HRTIM）

## 21.1 HRTIM 简介

高分辨率定时器可生成多达 12 路高度精确定时数字信号，主要用于驱动开关模式电源或照明系统等电源转换系统，但也可用于，一般对时间分辨率有极高要求的应用。

该定时器采用模块化架构，可生成独立波形或耦合波形。波形由独立式定时信号（使用计数器和比较单元）以及多种外部事件（如模拟或数字反馈以及同步信号）确定，因此可生成大量不同的控制信号（PWM、相移、恒定 Ton...），从而满足大部分转换拓扑的需求。

为实现控制和监测用途，该定时器还具有定时测量功能，并连接到内置的 ADC 和 DAC 转换器。此外，该定时器还具有轻载管理模式，能够处理各种故障机制，从而实现安全关断。

## 21.2 HRTIM 主要特征

- 高分辨率时序单元：
  - 163 ps 分辨率，可对电压和温度变化进行补偿
  - 所有输出均支持全分辨率，可在触发单脉冲模式下调整占空比、频率和脉宽
  - 6 个 16 位定时单元（每个定时单元包含 1 个独立计数器和 4 个比较单元）
  - 12 路输出可通过任何定时单元控制，每条通道多达 32 个置位/复位源
  - 模块化结构可满足多种配有 1 或 2 个开关的独立转换器的需求，也可满足少数大型多开关拓扑的需求
- 多达 10 个外部事件，可用于任何定时单元
  - 适用于峰值电流控制，零电压和零电流检测（ZVS 和 ZCS 操作）和计数器复位（变频操作）
  - 每个外部事件有 4 个源（数字输入、内置比较器输出、ADC 模拟看门狗和通用定时器的触发器器输出）
  - 可编程极性和边沿有效性
  - 5 个事件用于快速异步模式
  - 5 个事件用于可编程数字滤波器
  - 利用消隐和窗口模式实现伪事件过滤
- 多条通道可连接到内置模拟外设
  - 10 个用于 ADC 转换器的触发信号
  - 9 个用于 DAC 转换器的触发信号
  - 比较器输出（具有滤波和消隐功能）
- 丰富的保护机制
  - 适用于过流、短路和过压/欠压保护
  - 6 路故障输入可组合使用并关联到任何定时单元
  - 每个故障通道有 3 个源（数字输入、内置比较器输出和外部事件）
  - 可编程极性和边沿有效性，数字滤波器
  - 对谐振变换器配有专门的延时保护
- 多个 HRTIM 实例可与外部同步输入/输出同步
- 多功能输出级
  - 高分辨率死区时间插入（低至 651 ps）
  - 可编程输出极性

  - 斩波模式
- 突发模式控制器，可同时处理多个转换器上的轻载操作
- 8 个中断向量，每个向量最多具有 14 个源
- 7 个 DMA 请求，最多具有 14 个源，可通过突发模式实现多寄存器更新

## 21.3 HRTIM 功能描述

### 21.3.1 综合描述

HRTIM 可分为以下几个子实体：

- 主定时器
- 定时单元（定时器 A 到定时器 F）
- 输出级
- 突发模式控制器
- 外部事件和故障信号调节逻辑，由所有定时器共享
- 系统接口

主定时器基于 16 位递增计数器。它可通过 4 个比较单元置位/复位 12 路输出中的任何一路，并向 6 个定时器单元提供同步信号。其主要用途是使定时器单元受唯一的时钟源控制。交错降压转换器是一个典型的应用示例，主定时器在其中管理着多个单元之间的相移。

定时器单元既可以独立工作，也可以与其他定时器（包括主定时器）配合工作。每个定时器都可控制两路输出。输出置位/复位事件可以由定时单元比较寄存器触发，或者由来自主定时器、其他定时器的事件或外部事件触发。

输出级有多种用途：

- 在互补 PWM 模式下配置 2 路输出时添加死区
- 将载波频率添加到调制信号上
- 通过将输出异步置为预定义的安全电平来管理故障事件

在轻载运行的情况下，突发模式控制器可控制一个或多个定时器。突发长度和周期、以及输出的空闲状态可通过编程设定。

外部事件和故障信号调理逻辑包括：

- 输入选择 MUX（例如，为给定外部事件通道选择数字输入或片上时钟源）
- 极性和边沿有效性编程
- 数字滤波（对 12 条通道中的 5 条通道进行滤波）

系统接口允许 HRTIM 与 MCU 的其余部分进行交互：

- 向 CPU 发出中断请求
- 通过 DMA 控制器自动访问存储器，包括 HRTIM 特有的突发模式
- 触发 ADC 和 DAC 转换器

HRTIM 寄存器被分为 8 个组：

- 主定时器寄存器
- 定时器 A 到定时器 F 寄存器
- 通用寄存器，用于所有定时器单元共用的功能

*注：　按照书写惯例，文本和寄存器中提到的6 个定时单元用“x”字母表示，其中x 可以是A 到F 中的任何值。定时器的框图如下图所示。*

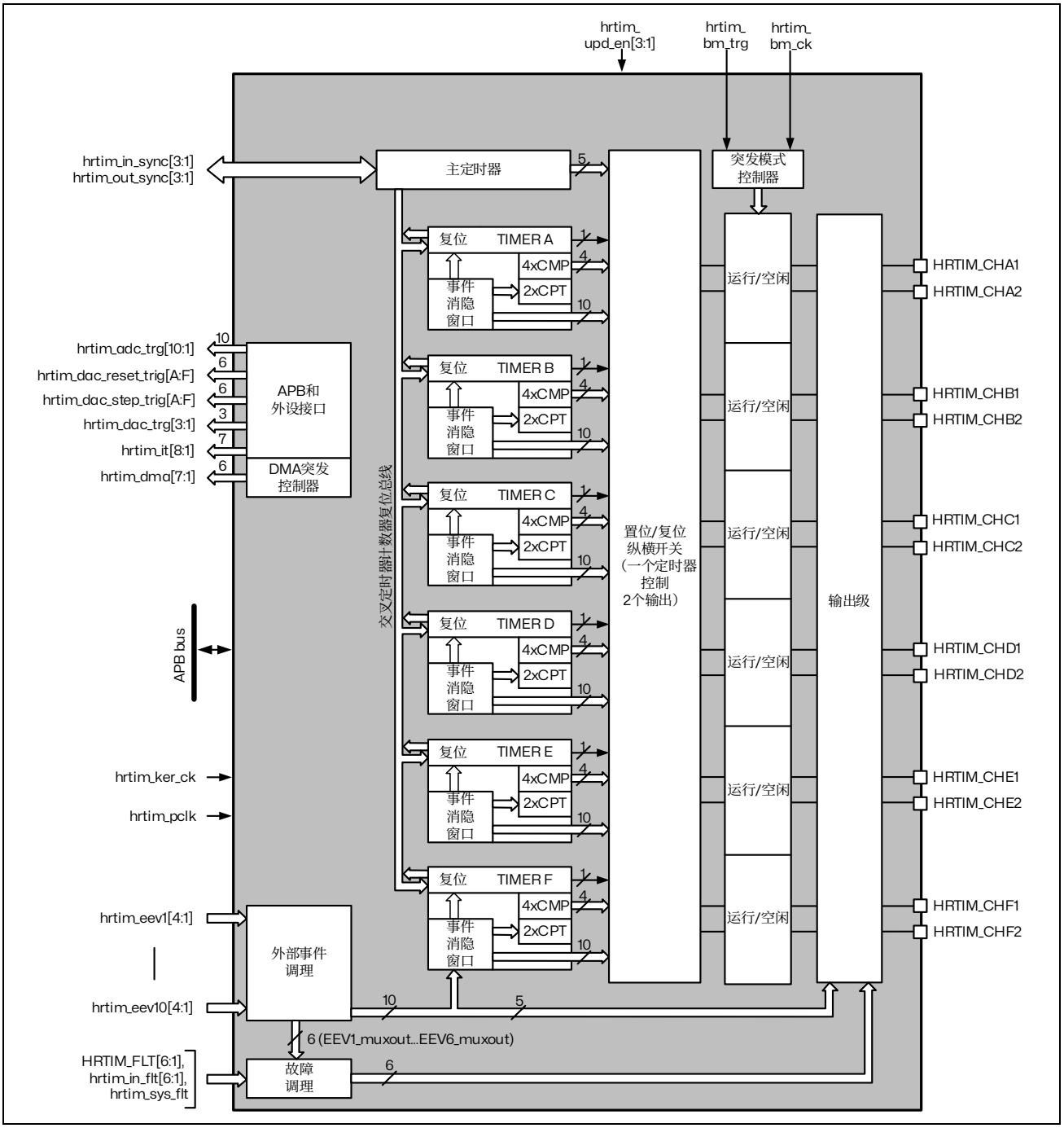



图 21.1 高分辨率定时器框图

### 21.3.2 HRTIM 引脚和内部信号

本节中的表格汇总了 HRTIM 的输入和输出，包括片上和片外的信号。

表 21.1 HRTIM 输入/输出汇总

| 信号名称 | 信号类型 | 描述 |
|---|---|---|
| HRTIM_CHA1,<br>HRTIM_CHA2,<br>HRTIM_CHB1,<br>HRTIM_CHB2,<br>HRTIM_CHC1,<br>HRTIM_CHC2,<br>HRTIM_CHD1,<br>HRTIM_CHD2,<br>HRTIM_CHE1,<br>HRTIM_CHE2,<br>HRTIM_CHF1,<br>HRTIM_CHF2 | 输出 | 主要的 HRTIM 定时器输出。<br>这些输出成对工作时（HRTIM_CHx1 & HRTIM_CHx2），可以插入死区或独立工作。 |
| hrtim_in_flt1[4:1]<br>hrtim_in_flt2[4:1]<br>hrtim_in_flt3[4:1]<br>hrtim_in_flt4[4:1]<br>hrtim_in_flt5[4:1]<br>hrtim_in_flt6[4:1] | 数字输入 | 故障输入：置为有效时立即禁止 HRTIM 输出（12 路片上输入和 6 路片外 HRTIM_FLTx 输入）。 |
| hrtim_sys_flt | 数字输入 | 涵盖 MCU 内部故障事件（时钟安全系统、SRAM 奇偶校验错误、内核锁住、PVD 输出）的系统故障。 |
| hrtim_in_sync[3:1] | 数字输入 | 将整个 HRTIM 与其他内部或外部定时器资源进行同步的同步输入：<br>hrtim_in_sync1：保留<br>hrtim_in_sync2：时钟源为 TIM1_TRGO 输出（详见表 22.6 内部同步源）<br>hrtim_in_sync3：时钟源为 HRTIM_SCIN 输入引脚 |
| hrtim_out_sync[2:1] | 数字输出 | 此输出用于级联或同步多个片上或片外 HRTIM 实例：<br>hrtim_out_sync1：保留<br>hrtim_out_sync2：目标是片外 HRTIM 或外设（通过 HRTIM_SCOUT 输出引脚） |

表 22.1 HRTIM 输入/输出汇总（继续）

| 信号名称 | 信号类型 | 描述 |
|---|---|---|
| hrtim_eev1[4:1] | 数字输入 | 外部事件：10 个事件均可在 4 个时钟源中选择，可选择片上时钟源（来自其他内置外设：比较器、ADC 模拟看门狗、TIMx 定时器、触发输出）或片外时钟源（HRTIM_EEVx 输入引脚）。 |
| hrtim_eev2[4:1] | | |
| hrtim_eev3[4:1] | | |
| hrtim_eev4[4:1] | | |
| hrtim_eev5[4:1] | | |
| hrtim_eev6[4:1] | | |
| hrtim_eev7[4:1] | | |
| hrtim_eev8[4:1] | | |
| hrtim_eev9[4:1] | | |
| hrtim_eev10[4:1] | | |
| hrtim_upd_en[3:1] | 数字输入 | HRTIM 寄存器更新使能输入 hrtim_upd_en[3:1]（片上互连）会触发从影子寄存器到活动寄存器的传输操作 |
| hrtim_bm_trg | 数字输入 | 在这个输入上的脉冲触发一个突发模式输入。这个输入连接到一个内部源（详见表 22.5 突发模式触发源） |
| hrtim_bm_ck[4:1] | 数字输入 | 突发模式时钟（片上互连） |
| hrtim_adc_trg[10:1] | 数字输出 | ADC 转换开始触发信号。hrtim_adc_trg1 .. hrtim_adc_trg10 输出在这里称为 ADC 触发 1 到 ADC 触发 10。 |
| hrtim_dac_trg[3:1] | 数字输出 | DAC 转换更新触发信号。 |
| hrtim_dac_reset_trg[6:1]<br>hrtim_dac_step_trg[6:1] | 数字输出 | 双通道 DAC 触发。 |
| hrtim_it[8:1] | 数字输出 | 中断请求。 |
| hrtim_dma[7:1] | 数字输出 | DMA 请求。 |
| hrtim_pclk | - | APB 时钟。 |
| hrtim_ker_ck | - | HRTIM 内核时钟（APB2 时钟）。 |

表 21.2 外部事件映射及相关功能

| 外部事件通道 | 快速模式 | 数字滤波 | 平衡故障定时器 A,B,C | 平衡故障定时器 D,E,F | 源（EExSRC[1:0]） | | | | 比较器和每个封装可用的输入源 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 源 1 (00) | 源 2 (01) | 源 3 (10) | 源 4 (11) | 64-pin 和 80-pin |
| hrtim_eev 1[4:1] | 是 | - | - | - | PC12 | COMP2 | TIM1_ TRGO | ADC1_ AWD | 比较器&输入 |
| hrtim_eev 2[4:1] | 是 | - | - | - | PC11 | COMP4 | TIM2_ TRGO | - | 比较器&输入 |
| hrtim_eev 3[4:1] | 是 | - | - | - | PB7 | - | TIM3_ TRGO | - | 比较器&输入 |
| hrtim_eev 4[4:1] | 是 | - | - | - | PB6 | COMP1 | - | ADC2_ AWD | 比较器&输入 |
| hrtim_eev 5[4:1] | 是 | - | - | - | PB9 | COMP3 | - | - | 比较器&输入 |
| hrtim_eev 6[4:1] | - | 是 | 是 | - | PB5 | COMP2 | COMP1 | - | 比较器&输入 |
| hrtim_eev 7[4:1] | - | 是 | 是 | - | PB4 | COMP4 | TIM7_ TRGO | ADC3_ AWD | 比较器&输入 |
| hrtim_eev 8[4:1] | - | 是 | - | 是 | PB8 | - | COMP3 | - | 比较器&输入 |
| hrtim_eev 9[4:1] | - | 是 | - | 是 | PB3 | - | TIM15_ TRGO | COMP4 | 比较器&输入 |
| hrtim_eev 10[4:1] | - | 是 | - | - | PC5/ PC6 | - | TIM6_ TRGO | - | 比较器&输入 |

表 21.3 更新使能输入和源

| hrtim_upd_en[3:1] | 更新源 |
|---|---|
| hrtim_upd_en1 | TIM16_OC |
| hrtim_upd_en2 | TIM17_OC |
| hrtim_upd_en3 | TIM6_TRGO |

表 21.4 来自通用定时器的突发模式时钟源

| hrtim_bm_ck[4:1] | BMCLK[3:0] | 时钟源 |
|---|---|---|
| hrtim_bm_ck1 | 0110 | TIM16_OC |
| hrtim_bm_ck2 | 0111 | TIM17_OC |
| hrtim_bm_ck3 | 1000 | TIM7_TRGO |
| hrtim_bm_ck4 | 1001 | 保留 |

表 21.5 突发模式触发源

| - | 触发源 |
|---|---|
| hrtim_bm_trg | TIM7_TRGO |

表 21.6 内部同步源

| - | 同步源 |
|---|---|
| hrtim_in_sync2 | TIM1_TRGO |

表 21.7 故障输入

| 故障通道 | 外部输入 FLTxSRC[1:0] =00 | 片上源 FLTxSRC[1:0] =01 | 外部输入 FLTxSRC[1:0] =10 | 片上源 FLTxSRC[1:0] =11 |
|---|---|---|---|---|
| hrtim_in_flt1[4:1] | PA12 | COMP2 | EEV1 | N/A |
| hrtim_in_flt2[4:1] | PA15 | COMP4 | EEV2 | N/A |
| hrtim_in_flt3[4:1] | PE6 | - | EEV3 | N/A |
| hrtim_in_flt4[4:1] | PC13 | COMP1 | EEV4 | N/A |
| hrtim_in_flt5[4:1] | PA11/PC7 | COMP3 | EEV5 | N/A |
| hrtim_in_flt6[4:1] | PC10 | - | EEV6 | N/A |

表 21.8 HRTIM DAC 触发器连接

| HRTIM DAC 触发器 | DAC1_CH1 DAC1_CH2 | DAC2_CH1 | DAC3_CH1 DAC3_CH2 |
|---|---|---|---|
| hrtim_dac_trg1 | 是 | - | - |
| hrtim_dac_trig2 | - | 是 | - |
| hrtim_dac_trig3 | - | - | 是 |
| hrtim_dac_reset_trg1 hrtim_dac_step_trg1 | 是 | 是 | 是 |
| hrtim_dac_reset_trg2 hrtim_dac_step_trg2 | 是 | 是 | 是 |
| hrtim_dac_reset_trg3 hrtim_dac_step_trg3 | 是 | 是 | 是 |
| hrtim_dac_reset_trg4 hrtim_dac_step_trg4 | 是 | 是 | 是 |
| hrtim_dac_reset_trg5 hrtim_dac_step_trg5 | 是 | 是 | 是 |
| hrtim_dac_reset_trg6 hrtim_dac_step_trg6 | 是 | 是 | 是 |

### 21.3.3 时钟

HRTIM 必须由 $t_{HRTIM}$ APB2 时钟提供，以实现全分辨率。$t_{HRTIM}$ 时钟周期通过使用边缘定位逻辑均匀分为最多 32 个中间步骤。HRTIM 中存在的所有时钟都源自这个参考时钟。

#### 术语定义

$f_{HRTIM}$：主 HRTIM 时钟（hrtim_ker_ck）。所有后续时钟均由该时钟源生成，并与该时钟源同步。

$f_{HRCK}$：高分辨率等效时钟。考虑到 $f_{HRTIM}$ 时钟周期被除以 32，它相当于 32 × $f_{HRTIM}$ 的频率。

$f_{DTG}$：死区发生器时钟。为了方便起见，本文档中仅使用 $t_{DTG}$ 周期（$t_{DTG}$ = 1/$f_{DTG}$）。

$f_{CHPFRQ}$：斩波级时钟源。

$f_{1STPW}$：定义斩波模式下初始脉冲长度的时钟源。为方便起见，本文档中仅使用 $t_{1STPW}$ 周期（$t_{1STPW}$ = 1/$f_{1STPW}$）。

$f_{BRST}$：突发模式控制器计数器时钟。

$f_{SAMPLING}$：对故障或外部事件输入进行采样时所需的时钟。

$f_{FLTS}$：由用作 $f_{SAMPLING}$ 时钟源的 $f_{HRTIM}$ 生成的时钟，用于对故障事件进行过滤。

$f_{EEVS}$：由用作 $f_{SAMPLING}$ 时钟源的 $f_{HRTIM}$ 生成的时钟，用于对外部事件进行过滤。

#### 定时器时钟和预分频器

HRTIM 中的每个定时器都有独立的时钟预分频器，以供用户调整定时器分辨率。（参见下表）。

表 21.9 定时器分辨率和最小 PWM 频率（$f_{HRTIM}$ = 192MHz 时）

| CKPSC[2:0] | 预分频比 | $f_{HRCK}$ 等效频率 | 分辨率 | 最小 PWM 频率 |
|---|---|---|---|---|
| 000 | 1 | 192 x 32 MHz = 6.144 GHz | 162.8 ps | 93.75 kHz |
| 001 | 2 | 192 x 16 MHz = 3.07 GHz | 326 ps | 46.87 kHz |
| 010 | 4 | 192 x 8 MHz = 1.54 GHz | 651 ps | 23.49 kHz |
| 011 | 8 | 192 x 4 MHz = 768 MHz | 1.30 ns | 11.75 kHz |
| 100 | 16 | 192 x 2 MHz = 384 MHz | 2.60 ns | 5.86 kHz |
| 101 | 32 | 192 MHz | 5.21 ns | 2.93 kHz |
| 110 | 64 | 192 /2 MHz = 96 MHz | 10.41 ns | 1.47 kHz |
| 111 | 128 | 192 /4 MHz = 48 MHz | 20.83 ns | 0.73 kHz |

高分辨率可用于边沿定位、PWM 周期调整以及外部触发的脉冲持续时间。

然而，以下功能并不支持高分辨率：

- 定时器计数器的读写访问
- 捕获单元

对于时钟预分频比小于 32（CKPSC[2:0] < 5）的情况，计数器和捕获寄存器的最低有效位（LSB）是不重要的。这些最低有效位无法写入（仅针对计数器寄存器），并且在读取时返回 0。

例如，如果 CKPSC[2:0] = 2（预分频比为 4），向计数器寄存器写入 0xFFFF 会得到一个有效的值 0xFFF8。相反，任何介于 0xFFFF 和 0xFFF8 之间的计数器值在读取时都会显示为 0xFFF8。

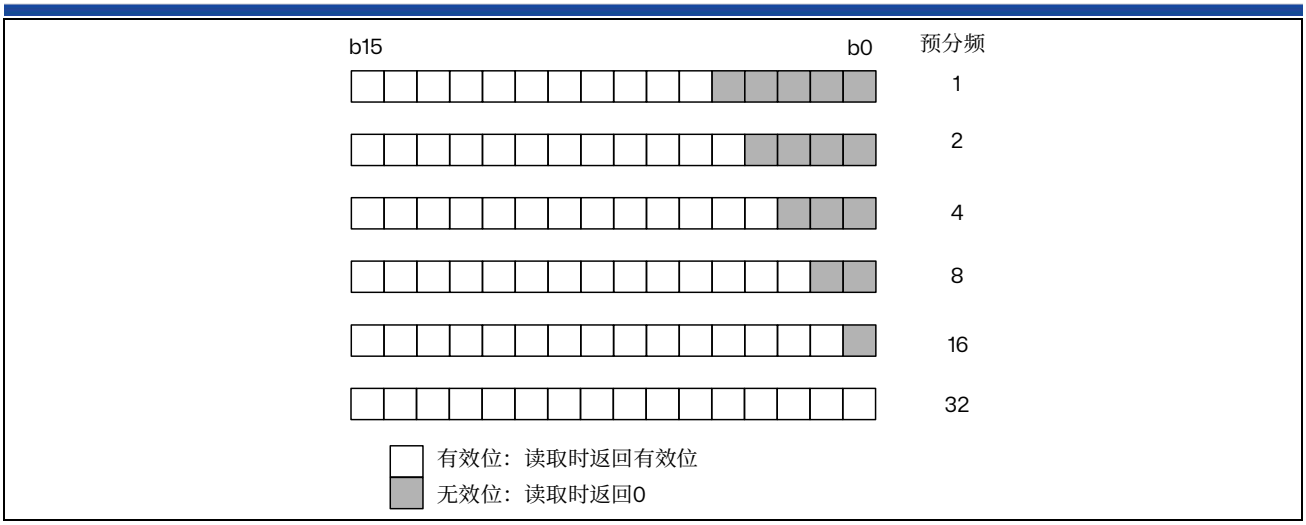



图 21.2 计数器和捕获寄存器格式与时钟预分频系数

### 初始化

启动时，务必先初始化预分频器位域，然后再写入比较和周期寄存器。定时器使能后（HRTIM_MCR 寄存器中的 MCEN 或 TxCEN 位已置 1），不能修改预分频器。

如果有多个定时器已使能，预分频器会与先启动的定时器的预分频器同步。

**警告：仅当计数器与输出行为与其他定时器的信息和信号无关时，主定时器和 TIMA..E 定时器才能使用不同的预分频比。如果以下某个事件从一个定时单元（或主定时器）传输到另一个定时单元，则务必要在这些定时器中配置相同的预分频比：输出置位/复位事件、计数器复位事件、更新事件、外部事件过滤或捕获触发。预分频系数不相等会导致结果不可预测。**

### 死区发生器时钟

死区时间的预分频器由 $(f_{HRTIM} \times 8) / 2^{(DTPRSC[2:0])}$ 提供，通过 HRTIM_DTxR 寄存器中 DTPRSC[2:0] 位编程。

当 $f_{HRTIM}$ = 192MHz 时，$t_{DTG}$ 的范围从 651ps 到 83.3ns。

### 斩波级时钟

斩波级时钟源 $f_{CHPFRQ}$ 由 $f_{HRTIM}$ 生成，其分频系数范围为 16 到 256，因此当 $f_{HRTIM}$ = 192 MHz 时，750 kHz ≤ $f_{CHPFRQ}$ ≤ 12 MHz。

$t_{1STPW}$ 是在斩波模式下初始脉冲的长度，它使用 HRTIM_CHPxR 寄存器中的 STRPW[3:0]位进行编程，具体如下：

$t_{1STPW}$ = (STRPW[3:0]+1) x 16 x $t_{HRTIM}$

它使用 $f_{HRTIM}$ / 16 作为时钟源（对于 $f_{HRTIM}$ = 192MHz，为 12MHz）。

### 突发模式预分频器

突发模式控制器计数器时钟 $f_{BRST}$ 可由多个时钟源提供，其中一个时钟源由 $f_{HRTIM}$ 生成。
在这种情况下，$f_{BRST}$ 的范围从 $f_{HRTIM}$ 到 $f_{HRTIM}$ / 32768（对于 $f_{HRTIM}$ = 192MHz，为 5.859kHz）。

### 故障输入采样时钟

故障输入噪声抑制滤波器具有一个由 $f_{SAMPLING}$ 定义的时间常数，它可以是 $f_{HRTIM}$ 或 $f_{FLTS}$。
$f_{FLTS}$ 是从 $f_{HRTIM}$ 派生而来，对于 $f_{HRTIM}$ = 192MHz，其范围从 192MHz 到 24MHz。

### 外部事件输入采样时钟

外部事件输入噪声抑制滤波器具有一个由 $f_{SAMPLING}$ 定义的时间常数，它可以是 $f_{HRTIM}$ 或 $f_{EEVS}$。
$f_{EEVS}$ 是从 $f_{HRTIM}$ 派生而来，对于 $f_{HRTIM}$ = 192MHz，其范围同样从 192MHz 到 24MHz。

### 21.3.4 定时器 A..F 定时单元

HRTIM 嵌入了 6 个完全相同的定时单元（这些定时单元由采用自动重载机制的 16 位递增计数器组成，用于定义计数周期）、4 个比较单元和 2 个捕获单元，如下图所示。

每个单元都包含对 2 路输出的所有控制功能，因此可作为独立定时器工作。

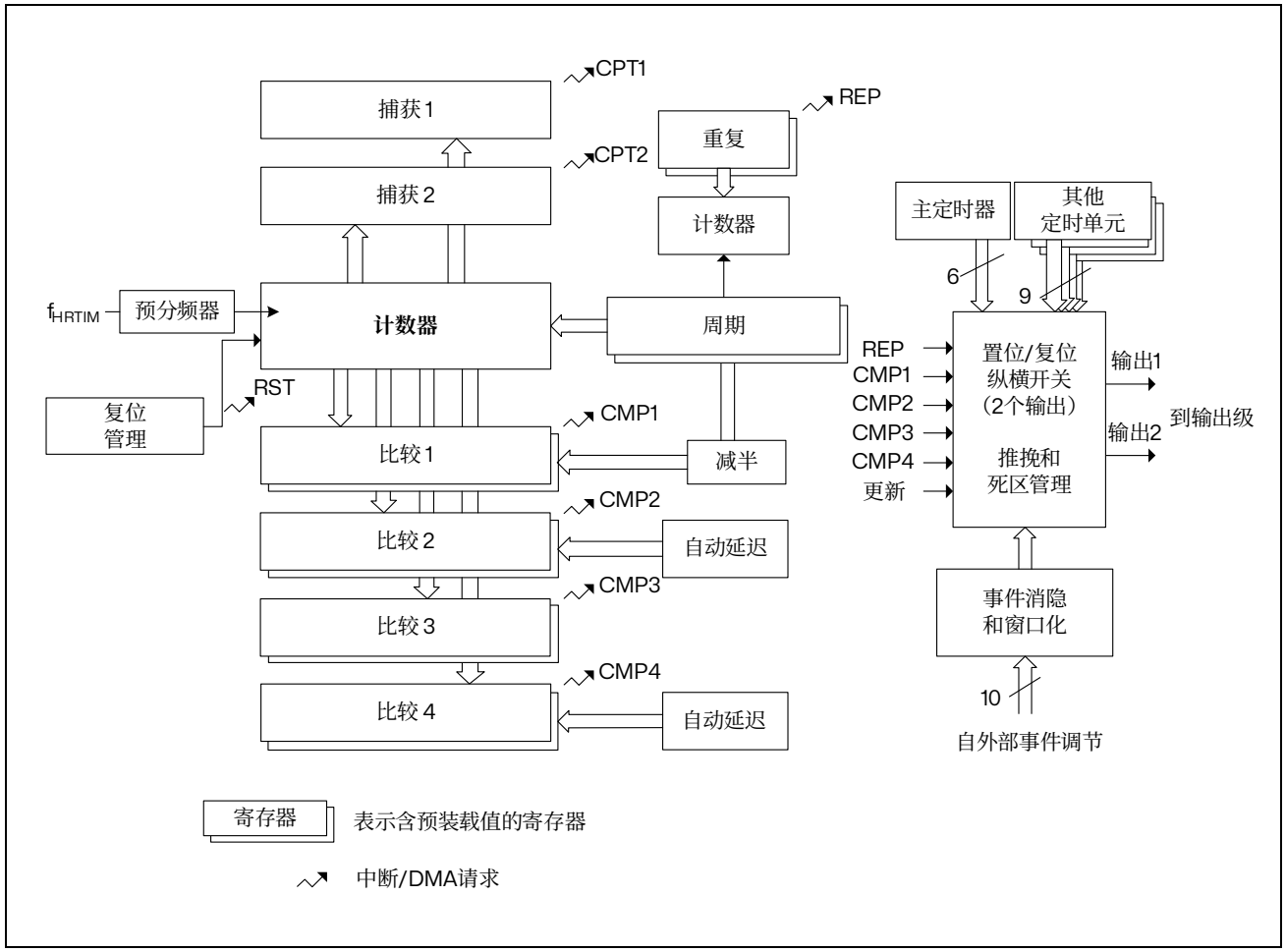



图 21.3 定时器 A..F 框图

周期和比较值必须在与高分辨率实现相关的下限和上限之间，这些限制在下表中列出：

然而，以下功能并不支持高分辨率：

- 最小值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期。值 0x0000 仅可以写入 CMP1 和 CMP3 寄存器中，以跳过 PWM 脉冲。有关详细信息，请参阅：“空占空比异常情况”一节。
- 最大值必须小于或等于 0xFFFF - 1 个 $f_{HRTIM}$ 时钟周期

表 21.10 周期寄存器与比较寄存器的最小和最大值

| CKPSC[2:0] 值 | 最小值 (1) | 最大值 |
|---|---|---|
| 0 | 0x0060 | 0xFFDF |
| 1 | 0x0030 | 0xFFEF |
| 2 | 0x0018 | 0xFFF7 |
| 3 | 0x000C | 0xFFFB |
| 4 | 0x0006 | 0xFFFD |
| ≥ 5 | 0x0003 | 0xFFFD |

1. 值 0x0000 仅可以写入 CMP1 和 CMP3 寄存器中，以跳过 PWM 脉冲。有关详细信息，请参阅“空占空比异常情况”一节。

*注： 比较值大于周期寄存器值不会产生比较匹配事件。*

### 计数器工作模式

定时器 A..F 可在连续（自由运行）模式下工作，也可以单发方式工作，此时会由复位事件触发开始计数，工作模式通过 HRTIM_TIMxCR 控制寄存器中的 CONT 位设置。附加的 RETRIG 位可用于选择单发操作是可再触发的或不可再触发的。工作模式的详细情况在下表以及下图中进行了总结。

表 21.11 定时器工作模式

| CONT | RETRIG | 工作模式 | 启动/停止条件<br>时钟和事件生成 |
|---|---|---|---|
| 0 | 0 | 单发<br>不可再触发 | 将 TxEN 位置 1 会使能定时器，但不会启动计数器。<br>第一个复位事件会触发计数器开始计数，但计数器在达到 PER 值之前会忽略任何后续复位事件。<br>随后会生成 PER 事件，计数器停止计数。<br>发生复位事件后，计数器会重新从 0x0000 开始计数。 |
| 0 | 1 | 单发<br>可再触发 | 将 TxEN 位置 1 会使能定时器，但不会启动计数器。<br>如果计数器停止计数，则复位事件会使计数器开始计数，否则会将计数器清零。计数器达到 PER 值后，会生成 PER 事件，计数器会停止计数。发生复位事件后，计数器会重新从 0x0000 开始计数。 |
| 1 | x | 连续模式 | 将 TxEN 位置 1 会使能定时器，同时会启动计数器。<br>计数器达到 PER 值后，会翻转到 0x0000 并重新开始计数。 可以随时复位计数器。 |

可以随时清零 TxEN 位，以禁止定时器并停止计数。

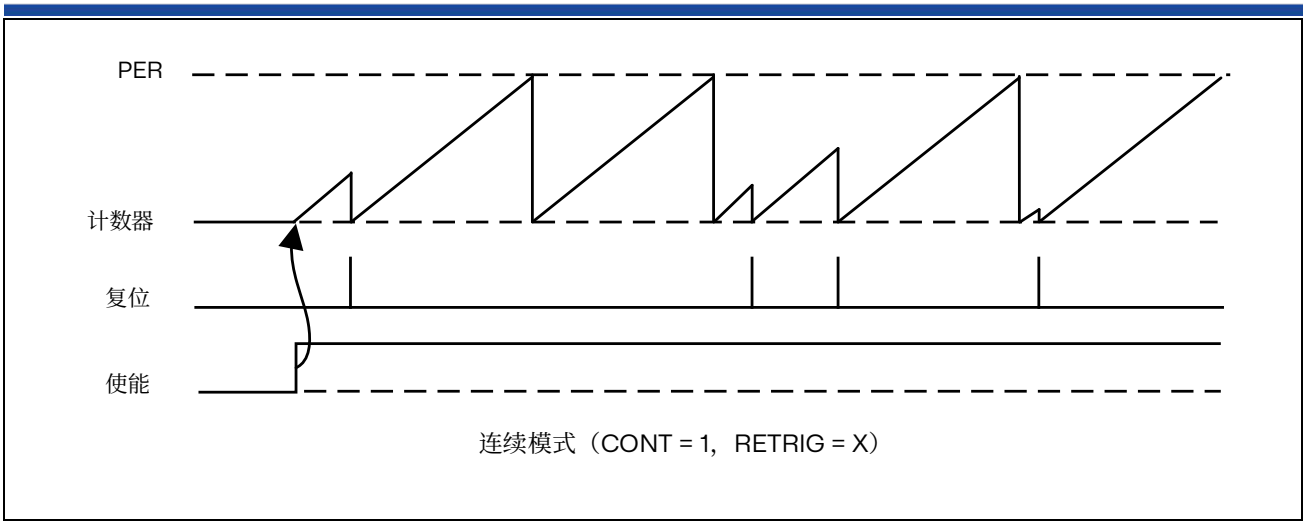



图 21.4 连续定时器工作模式

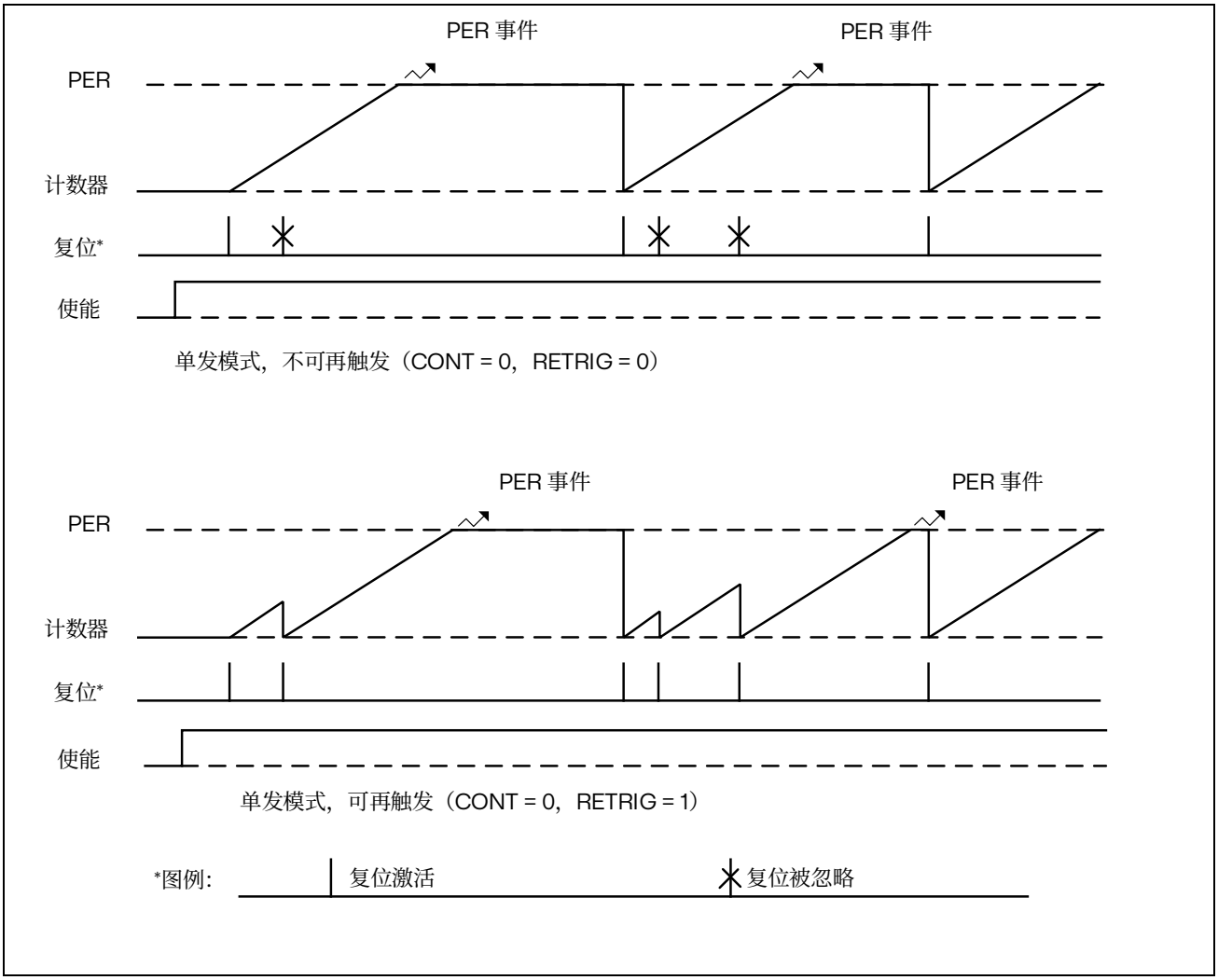



图 21.5 单发定时器工作模式

### 翻转事件

在连续模式下，如果计数器在达到 HRTIM_PERxR 寄存器中设置的周期值后恢复为 0，则会生成计数器翻转事件。

该事件在 HRTIM 中用于多种用途：

- 置位/复位输出
- 触发寄存器内容更新（从预装载寄存器传输到活动寄存器）
- 触发中断请求（IRQ）或直接内存访问（DMA）请求
- 作为突发模式时钟源或突发启动触发信号
- 作为 ADC 触发信号
- 使重复计数器递减

如果初始计数器值大于定时器启动时的周期值，或者在计数器已超过该值时设置了新周期，计数器不会复位：计数器将在达到最大周期值时溢出，并且重复计数器不会递减。

### 定时器复位

定时单元计数器的复位可通过多达 30 种事件触发，这些事件可同时在 HRTIM_RSTxR 寄存器中选择，具体包括以下复位源：

- 定时单元：比较 2、比较 4 和更新（3 个事件）
- 主定时器：复位和比较 1..4（5 个事件）
- 外部事件 EXTEVNT1..10（10 个事件）
- 所有其他定时单元（例如，对于定时器 A，有定时器 B..F）：比较 1、2 和 4（12 个事件）

可同时选择多个事件处理多个复位源。在这种情况下，会对多个复位请求进行或运算。如果在同一 $f_{HRTIM}$ 时钟周期内生成 2 个计数器复位事件，则会考虑后一个定时器复位事件。

此外，还可以使用 HRTIM_CR2 寄存器中的 TxRST 位对计数器执行软件复位。这些控制位分组到一个寄存器中，从而可同时复位多个计数器。

仅当相关计数器已使能后（TxCEN 位置 1），才会考虑复位请求。

当 $f_{HRTIM}$ 时钟的分频比大于 32（计数周期大于 $f_{HRTIM}$）时，计数器复位事件会延迟到预分频时钟的下一有效边沿。这样可确保在输出跳变同步到复位事件（通常是恒定 Ton 时间转换器）时生成的波形无抖动。

下图展示了当时钟分频比为 128（$f_{HRTIM}$ 除以 4）时的复位处理方式。

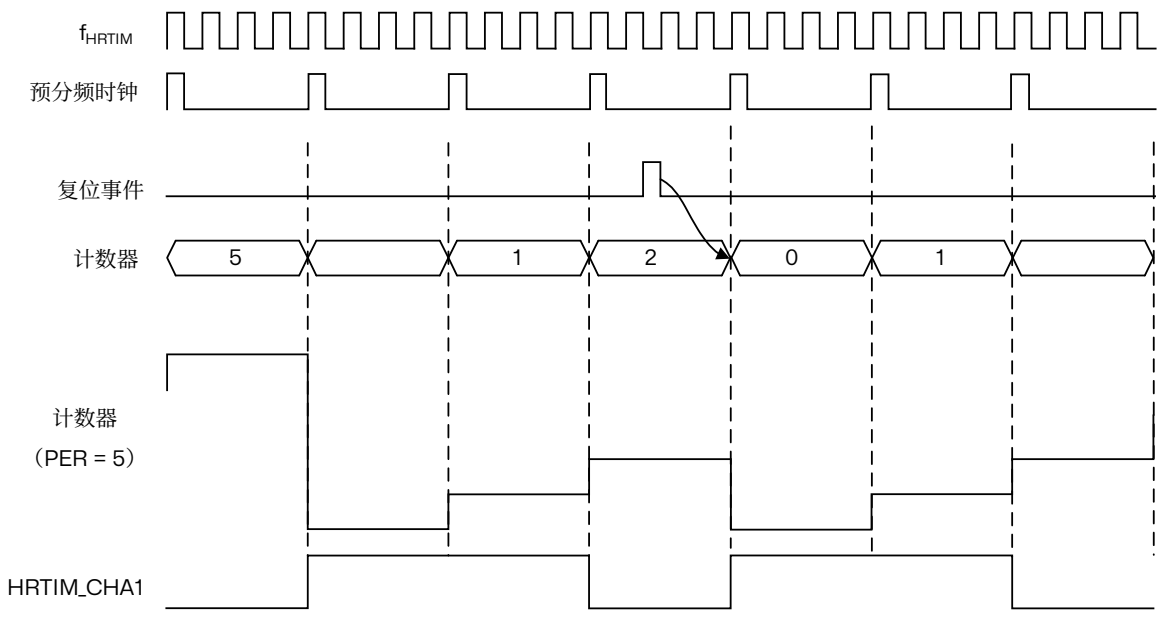



图 21.6 定时器复位重新同步（预分频比大于 32）

### 重复计数器

软件通常会在达到周期值时生成中断，从而在下一周期开始之前留出最长的时间用于处理。重复计数器的主要用途是通过分离开关频率和中断频率来调整周期中断率并分担 CPU 的负荷。

定时单元包含重复计数器。该计数器无法读取，但只可使用 HRTIM_REPxR 寄存器中的自动重载值进行编程。

定时器使能后（TXCEN 位置 1），重复计数器会初始化为 HRTIM_REPxR 寄存器的内容。定时器使能后，每次计数器由于复位事件或计数器翻转而清空时，重复计数器都会减 1。当重复计数器达到 0 后，会发出 REP 中断或 DMA 请求（若使能，使用 HRTIM_DIER 寄存器中的 REPIE 和 REPDE 位）。

如果 HRTIM_REPxR 寄存器设为 0，会在每个周期产生中断。如果值大于 0，则会(HRTIM_REPxR + 1)个周期后产生 REP 中断。下图显示了连续模式下重复计数器取不同值时的操作。

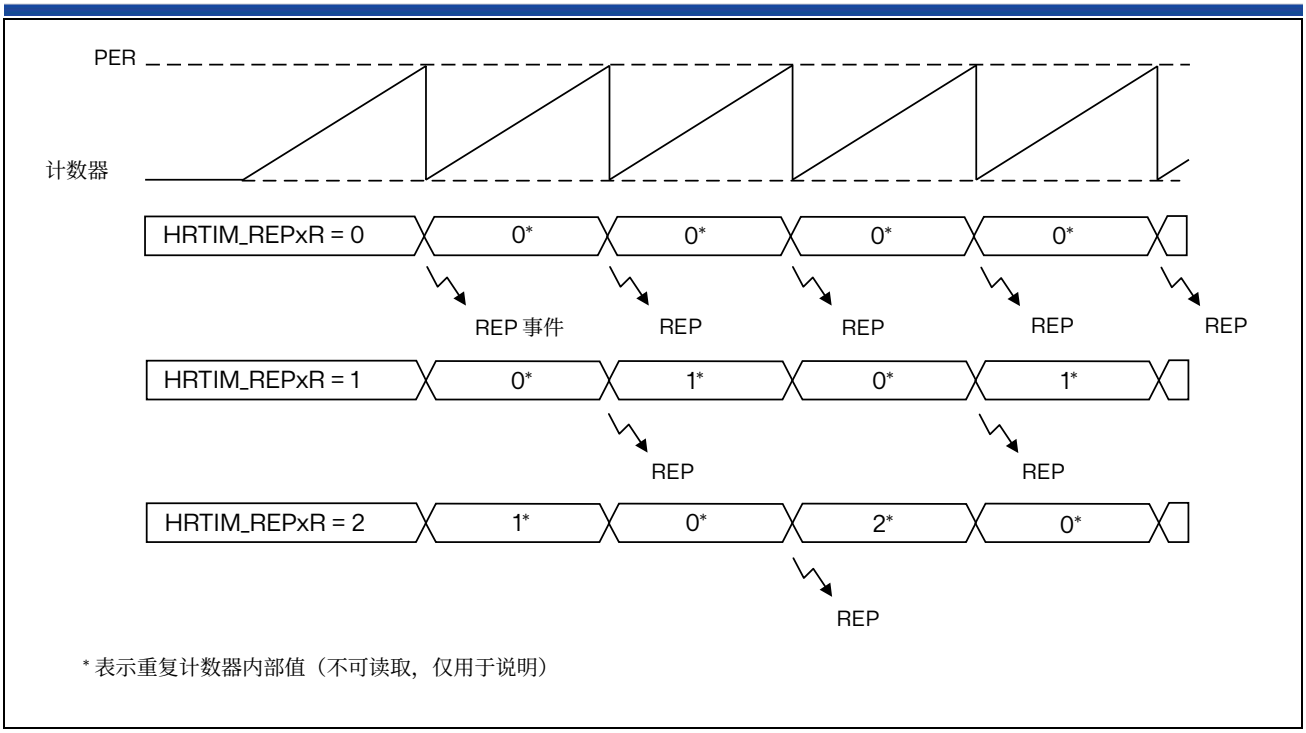



图 21.7 连续模式下的重复率与 HRTIM_REPxR 内容

无论在连续模式还是单发模式下，如果计数器在达到周期值（可变频率操作）之前复位，则也可使用重复计数器（如下图所示）。复位会使重复计数器在计数器使能后（TxCEN 位置 1）执行第一次启动时递减。

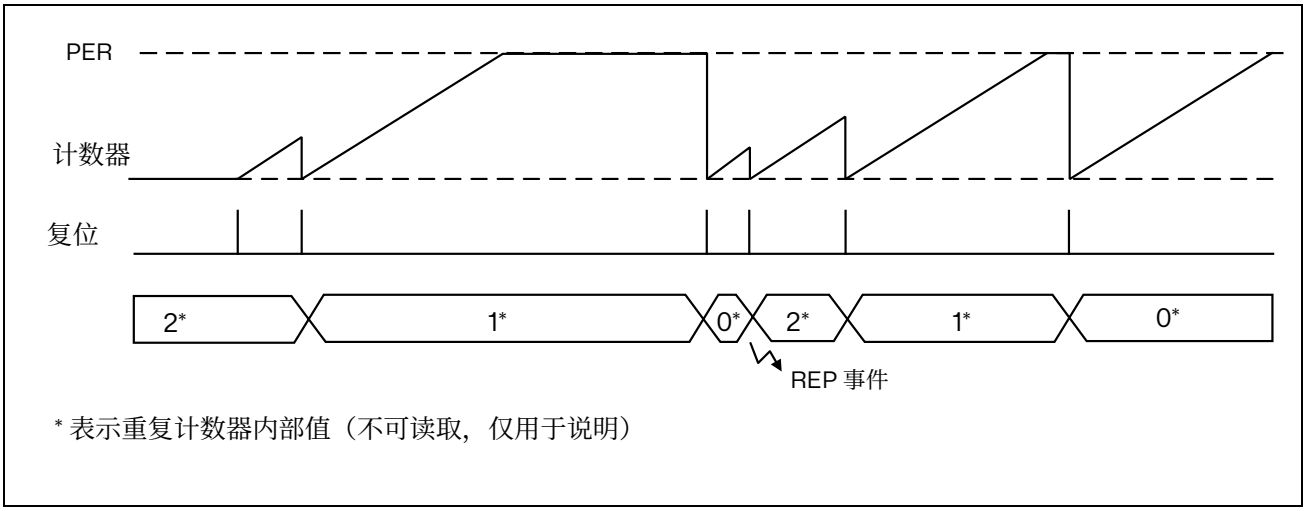



图 21.8 单发模式下的重复计数器行为

来自 HRTIM_SCIN[3:1]源的复位或启动事件会像其他任何复位事件一样使重复计数器递减。但在通过 SYNCIN 启动的单发模式下（HRTIM_TIMxCR 寄存器中的 SYNCSTRTx 位置 1），重复计数器仅会在周期后出现第一个复位事件时递减。任何后续的复位事件都不会更改重复计数器的值，直至计数器通过新的 HRTIM_SCIN[3:1]输入请求重启。

### 置位/复位纵横开关

“置位”事件相当于跳变为输出有效状态，而“复位”事件则相当于跳变为输出无效状态。

波形的极性在输出级中定义，以适应正逻辑或负逻辑外部组件：对于正极性（POLx = 0），有效电平对应于逻辑电平 1；对于负极性（POLx = 1），有效电平对应于逻辑电平 0。

每个定时单元都会控制两路输出的置位/复位纵横开关。这两路输出可通过多达 32 种事件置位、复位或切换，这些事件可从以下源中选择：

- 定时单元：周期、比较 1..4、寄存器更新（6 个事件）
- 主定时器：周期、比较 1..4、HRTIM 同步（6 个事件）
- 所有其他定时单元（例如，对于定时器 A，则为定时器 B..F）：TIMEVNT1..9（下表中描述的 9 个事件）
- 外部事件 EXTEVNT1..10（10 个事件）
- 软件强制（1 个事件）

*注：在递增/递减计数模式（*UDM *位置*1*）下，计数器的周期事件根据*OUTROM[1:0]*位设置来定义。*

事件源会进行或运算，可同时选择多个事件。

每路输出均由两个 32 位寄存器控制，一个寄存器包含置位编码（HRTIM_SETxyR），另一个寄存器包含复位编码（HRTIM_RSTxyR），其中 x 代表定时单元 A..F，y 代表输出 1 或 2（例如 HRTIM_SETA1R、HRTIM_RSTC2R...）。

如果为置位和复位选择了相同事件，则会切换输出状态。每个 HRTIM 周期输出状态的切换次数不能超过 1 次：如果同一周期中有两个连续的切换事件，则仅会考虑第一个切换事件。

仅当计数器使能后（TxCEN 位置 1），才会考虑置位和复位请求，但软件在定时器启动时强制请求允许预置输出的情况除外。

下表汇总了来自其他定时单元的可用于置位和复位输出的事件。编号对应于寄存器中列出的定时器事件（如 TIMEVNTx），空白位置表示不可用事件。

例如，定时器 A 输出可由以下事件置位或复位：定时器 B 比较 1、2，定时器 C 比较 2 和 3... 定时器 E 比较 3 将列为 HRTIM_SETA1R 中的 TIMEVNT7。

表 21.12 定时器 A 到 F 之间的事件映射

| 源 | | 定时器 A | | | | 定时器 B | | | | 定时器 C | | | | 定时器 D | | | | 定时器 E | | | | 定时器 F | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CMP1 | CMP2 | CMP3 | CMP4 | CMP1 | CMP2 | CMP3 | CMP4 | CMP1 | CMP2 | CMP3 | CMP4 | CMP1 | CMP2 | CMP3 | CMP4 | CMP1 | CMP2 | CMP3 | CMP4 | CMP1 | CMP2 | CMP3 | CMP4 |
| 目标 | TA | - | - | - | - | 1 | 2 | - | - | - | 3 | 4 | - | 5 | 6 | - | - | - | - | 7 | 8 | - | - | - | 9 |
| | TB | 1 | 2 | - | - | - | - | - | - | - | - | 3 | 4 | - | - | 5 | 6 | 7 | 8 | - | - | - | - | 9 | - |
| | TC | - | 1 | 2 | - | - | 3 | 4 | - | - | - | - | - | - | 5 | - | 6 | - | - | 7 | 8 | - | 9 | - | - |
| | TD | 1 | - | - | 2 | - | 3 | - | 4 | - | - | - | 5 | - | - | - | - | 6 | - | - | 7 | 8 | - | 9 | - |
| | TE | - | - | - | 1 | - | - | 2 | 3 | 4 | 5 | - | - | 6 | 7 | - | - | - | - | - | - | - | - | 8 | 9 |
| | TF | - | - | 1 | - | 2 | - | - | 3 | 4 | - | - | 5 | - | - | 6 | 7 | - | 8 | 9 | - | - | - | - | - |

下图展示了如何使用两个比较事件来生成 PWM（脉冲宽度调制）信号。

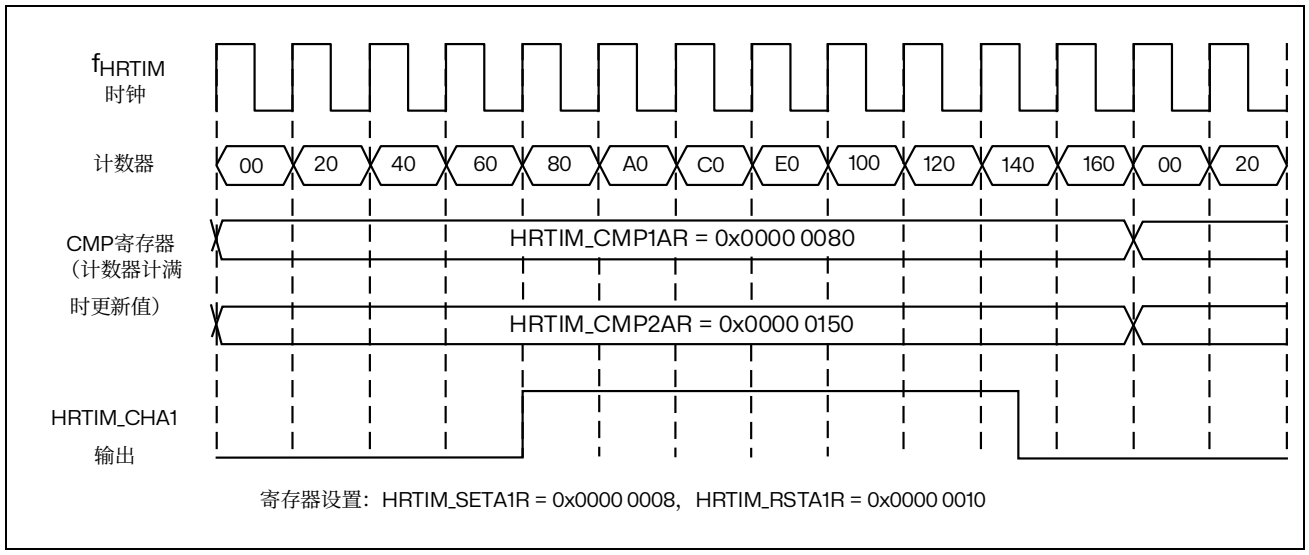



图 21.9 比较事件对输出的操作：发生比较 1 时置位，发生比较 2 时复位

### 发生更新事件时置位/复位

在更新时进行的置位或复位事件是以低分辨率进行的。当 CKPSC[2:0] < 5 时，高分辨率延迟被设置为最大值，因此与其他比较置位/复位事件相比，更新时的置位/复位事件总是滞后的，其抖动在 0 到 31/32 个 $f_{HRTIM}$ 时钟周期之间变化。

### 半占空比模式

此模式用于生成占空比固定为 50%、频率可变的方波信号（通常用于使用谐振拓扑的转换器），允许在设定新周期时自动将占空比强制设为周期值的一半。

要使能此模式，应向 HRTIM_TIMxCR 寄存器中的 HALF 位写入 1。HRTIM_PERxR 寄存器写入数值后，会自动将比较 1 值更新为 HRTIM_PERxR/2 值。

生成方波的输出必须编程为在发生 CMP1 事件时进行一次跳变，在发生周期事件时进行一次跳变，具体如下：

- HRTIM_SETxyR = 0x0000 0008，HRTIM_RSTxyR = 0x0000 0004，或者
- HRTIM_SETxyR = 0x0000 0004，HRTIM_RSTxyR = 0x0000 0008

HALF 模式会覆盖 HRTIM_CMP1xR 寄存器的内容。访问 HRTIM_PERxR 寄存器仅会更新比较 1 内部寄存器。用户可访问的 HRTIM_CMP1xR 寄存器不会更新为 HRTIM_PERxR / 2 的值。

当使能预装载（PREEN = 1、MUDIS、TxUDIS）时，则会在发生更新事件时刷新比较 1 活动寄存器。如果禁止预装载 (PREEN= 0)，则在 HRTIM_PERxR 写入数值后，比较 1 活动寄存器会立即更新。

如果使能了 HALF 模式，周期必须大于或等于 6 个 $f_{HRTIM}$ 时钟周期（如果 CKPSC[2:0] = 0，则为 0xC0；如果 CKPSC[2:0] = 1，则为 0x60；如果 CKPSC[2:0] = 2，则为 0x30；依此类推）。

### 交错模式

交错模式是对半周期模式的补充，有助于实现交错拓扑。

当更新 HRTIM_PERxR 的值时，该模式允许自动重新计算比较寄存器的内容。

选择交错模式是通过 HRTIM_MCR 和 HRTIM_TIMxCR 寄存器中的 HALF 位和 IL[1:0]位来完成的，如下表所示。

表 21.13 交错模式选择

| HALF 位 | INTLVD[1:0] 位 | 模式 |
|---|---|---|
| 0 | 00 | 失能 |
| 0 | 01 | 三重交错（120°） |
| 0 | 10 | 四重交错（90°） |
| 0 | 11 | 保留 |
| 1 | xx | 双重交错（180°） |

下表列出了三种可用模式的比较值。比较寄存器的内容将被覆盖。相应的比较事件可用于触发输出置位/复位或重置从属定时器。

表 21.14 交错模式下比较 1 至 3 的值

| 模式 | 双重交错<br>180° | 三重交错<br>120° | 四重交错<br>90° |
|---|---|---|---|
| CMP1 值 | PERxR/2 | PERxR/3 | PERxR/4 |
| CMP2 值 | 无影响 | 2x (PERxR/3) | PERxR/2 |
| CMP3 值 | 无影响 | 无影响 | 3x (PERxR/4) |

*注： 在半占空比模式和交错模式下，比较寄存器由硬件控制，直接写入这些寄存器不会产生影响。但是，写入的值会被存储在预加载寄存器中，并在退出这些模式后的更新事件上生效。*

*注： 三重和四重交错模式不应与其他使用 CMP2 的模式（如双通道 DAC 触发模式和半触发占空比模式）同时使用。*

### 零占空比异常情况

对于宽度小于 3 个 $t_{HRTIM}$ 时钟周期的脉冲，不支持高分辨率行为（参见章节：置位/复位事件优先级和窄脉冲管理），并且禁止在 HRTIM_TIMxCMPy 寄存器中写入任何严格小于 $f_{HRTIM}$ 时钟 3 个周期的值（如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18；依此类推）（参见章节：HRTIM 定时器 x 比较 1 寄存器（HRTIM_CMP1xR）（x = A 到 F））。

然而，要跳过输出脉冲并实现零占空比，只需在满足以下条件时，在以下两个寄存器中写入零值即可：HRTIM_TIMxCMP1 和 HRTIM_TIMxCMP3：

- 输出设置（SET）事件由周期（PERIOD）事件生成
- 输出复位（RESET）由比较 1（或比较 3）事件生成
- 比较 1（或比较 3）事件在定时器单元本身内是有效的，并且不用于其他定时单元

对于其他任何用例，可以通过使用完全相同的比较值（大于 $f_{HRTIM}$ 时钟的 3 个周期）来编程 SET 和 RESET 事件来实现。在这种情况下，输出将被强制复位（参见章节：置位/复位事件优先级和窄脉冲管理）。

### 交换模式

此模式允许通过单个位访问来交换两个输出：输出 1 信号连接到输出 2 引脚，而输出 2 信号连接到输出 1 引脚。输出的交换通过 HRTIM_CR2 寄存器中的 SWPx 位触发，并在下一个更新事件上生效。

交换发生在置位/复位纵横开关单元之前，如下所示：

- 如果 SWPx = 0，则 HRTIM_SETx1R 和 HRTIM_RSTx1R 编码为输出 1，HRTIM_SETx2R 和 HRTIM_RSTx2R 编码为输出 2
- 如果 SWPx = 1，则 HRTIM_SETx1R 和 HRTIM_RSTx1R 编码为输出 2，HRTIM_SETx2R 和 HRTIM_RSTx2R 编码为输出 1

交换模式仅影响预加载寄存器，而不影响有效寄存器。

*注：在使用交换模式时，必须使能预加载模式。*

因此，它不会修改与输出阶段相连的正规输出并行的辅助输出（有关详细信息，请参阅后续章节）。它们提供以下内部状态、事件和信号：

- O1CPY、O2CPY、SETxy 和 RSTxy 状态标志，以及相应的中断和 DMA 请求
- 输出设置/复位时的捕获触发器（TA2、TB2、TC2、TD2、TE2、TF2）
- 使用 Tx2 输出副本生成的外部事件滤波器

例如，当 SWP = 0 时，SETx1 标志与输出 1 相关；当 SWPx = 1 时，它与输出 2 相关。

同样，交换模式不会改变 HRTIM_OUTxR 寄存器中控制位的归属（DIDLx、CHPx、FAULTx[1:0]、IDLESx、POLx 位）。例如，POL1 位控制输出 1 的极性，无论 SWP 位的值如何。

*注：在推挽模式（在* HRTIM_TIMxCR *寄存器中* PSHPLL = 1*）下，* SWPx *位将被忽略。*

### 捕获

定时单元能够在内部和外部事件的触发下捕获计数器值。捕获的目的是：

- 测量事件到达时间或发生间隔。
- 在自动延迟模式下更新比较 2 和比较 4 的值（参见：自动延迟模式）。

捕获操作以 $f_{HRTIM}$ 的分辨率进行：对于时钟预分频比小于 32（CKPSC[2:0] < 5）的情况，寄存器的最低有效位是不重要的（读取为 0）。

定时器包含 2 个捕获寄存器：HRTIM_CPT1xR 和 HRTIM_CPT2xR。捕获触发事件在 HRTIM_CPT1xCR 和 HRTIM_CPT2xCR 寄存器中编程。

定时单元计数器的捕获可以由多达 28 个事件触发，这些事件可以同时在 HRTIM_CPT1xCR 和 HRTIM_CPT2xCR 寄存器中选择，来源包括：

- 外部事件，EXTEVNT1..10（10 个事件）
- 所有其他定时单元（例如，对于定时器 A，则为定时器 B..F）：比较 1、2 和输出 1 置位/复位事件（16 个事件）
- 定时单元：更新（1 个事件）
- 软件捕获（1 个事件）

可同时选择多个事件处理多个捕获触发信号。在这种情况下，会对多个并发触发请求进行或运算。如果 HRTIM_TIMxDIER 寄存器中的 CPTxIE 和 CPTxDE 位已置 1，捕获可生成中断或 DMA 请求。

电路未采用避免重复捕获的机制：即使前一个值未读取、或者捕获标志未清零，也会触发新捕获。

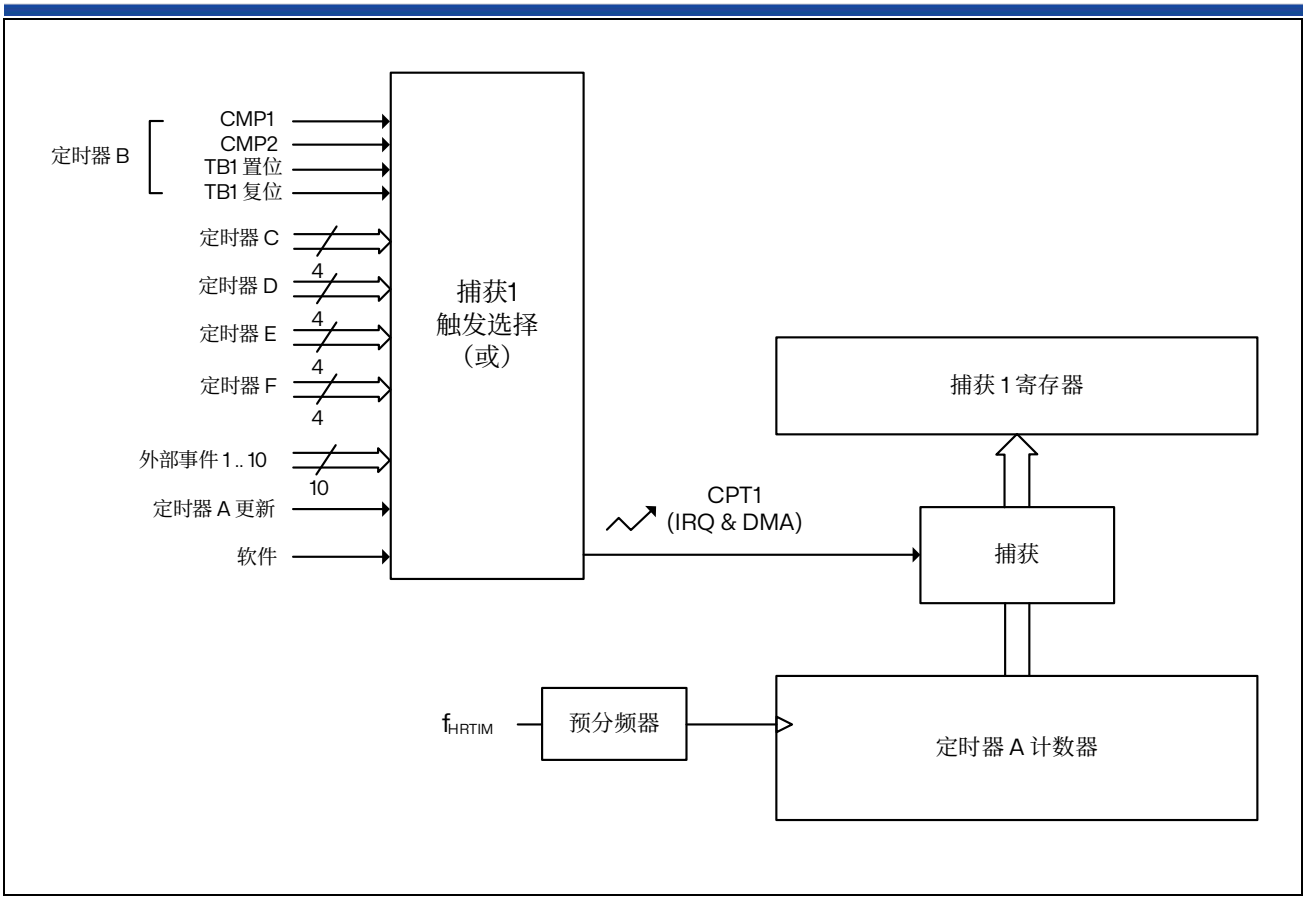



图 21.10 定时器 A 定时单元捕获电路

## 自动延迟模式

此模式可相对于捕获事件生成比较事件，这样一来，便可在捕获后的设定时间进行输出更改等操作。在这种情况下，会独立于定时器计数器值进行比较匹配。这样可生成时序与外部事件同步的波形，而无需进行软件计算和中断维护。

只要未触发捕获事件，便会忽略 HRTIM_CMPxR 寄存器的内容（计数器值与比较值相匹配的情况下，不会生成比较事件）。一旦触发了捕获事件，则会对在 HRTIM_CMPxR 中编程的比较值与 HRTIM_CPTxyR 中捕获的计数器值求和，并会用得出的结果更新内部自动延迟比较计数器，如下图所示。自动延迟比较寄存器属于定时单元内部的寄存器，不能读取。HRTIM_CMPxR 预装载寄存器的内容在计算后不会修改。

此特性仅适用于比较 2 和比较 4 寄存器。比较 2 寄存器与捕获 1 相关联，而比较 4 寄存器与捕获 2 相关联。与常规模式一样，HRTIM_CMP2xR 和 HRTIM_CMP4xR 比较寄存器不能编程为小于 3 个 $f_{HRTIM}$ 时钟周期的值。

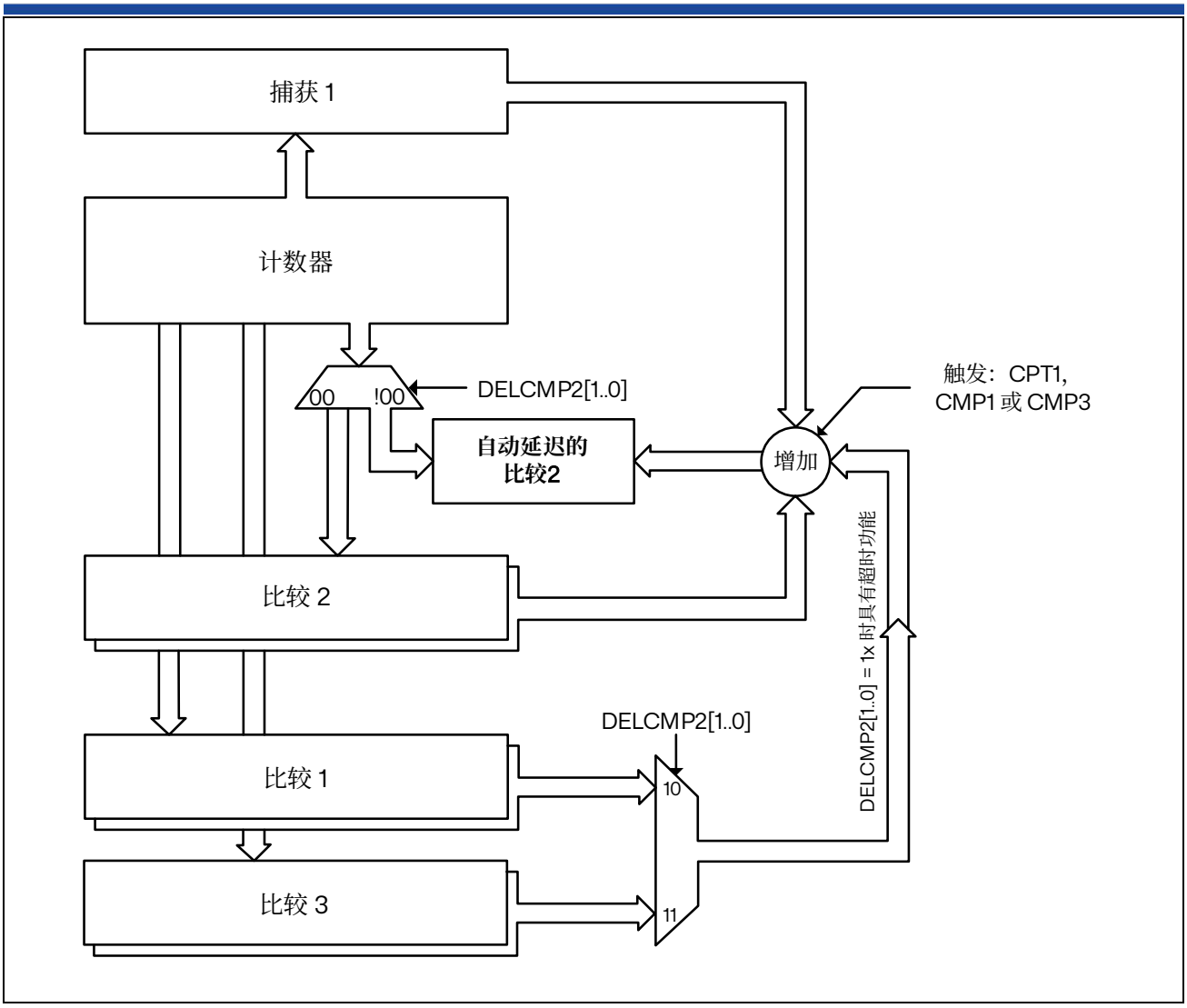



图 21.11 自动延迟概述（仅限比较 2 寄存器）

自动延迟比较寄存器的有效期从捕获开始，到周期事件为止：计数器达到周期值后，系统会重新设置，比较寄存器在发生捕获之前会处于禁用状态。

HRTIM_TIMxCR 寄存器中的 DELCMP2[1:0]和 DELCMP4[1:0]位可配置自动延迟模式，具体如下：

- 00
  常规比较模式：会直接将 HRTIM_CMP2xR 和 HRTIM_CMP4xR 寄存器的内容与计数器值进行比较。
- 01
  自动延迟模式：会重新计算比较 2 和比较 4 寄存器值，并在捕获 1/2 事件后用计算值与计数器值进行比较。
- 10 或 11
  具有超时的自动延迟模式：会重新计算比较 2 和比较 4 寄存器值，并在捕获 1/2 事件后用计算值与计数器值进行比较；如果缺少捕获 1/2 事件，则在比较 1 匹配（DELCMPx[1:0]= 10）或比较 3 匹配（DELCMPx[1:0]= 11）后用计算值与计数器值进行比较，以便实现超时功能。

进行捕获时，会与（HRTIM_CMP2/4xR + HRTIM_CPT1/2xR）值进行比较。如果周期内未触发捕获，行为将取决于 DELCMPx[1:0]值：

- DELCMPx[1:0] = 01：未生成比较事件
- DELCMPx[1:0] = 10 或 11：与 2 个比较寄存器值之和进行比较（例如 HRTIM_CMP2xR + HRTIM_CMP1xR）。如果捕获是在 CMPx + CMP1（或 CMPx + CMP3）后触发的，则不会考虑捕获。

下一 PWM 周期开始时会再次使能捕获。

如果自动延迟求和的结果大于 0xFFFF（溢出），则会忽略该值且新周期开始之前不会生成比较事件。

*注：如果从一个值重新编程为另一个值，以便正确地重新初始化自动延迟机制，则必须复位* DELCMPx[1:0] *位域，例如：*

- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

下图举例说明了以下信号是怎样生成的：

- 当计数器等于比较 1 值时，输出置位
- 给定外部事件下降沿 5 个周期后，输出复位

*注：为了简化图示，本例中未使用高分辨率（*CKPSC[2:0] = 101*），因此计数器以* $f_{HRTIM}$ *的速率递增。同样，图中所示的外部事件信号不含任何重新同步延迟：实际情况下，由于需要通过内部重新同步阶段来处理外部输入信号，因此下降沿与捕获事件之间会有*1 *到*2 *个* $f_{HRTIM}$ *时钟周期的延迟。*

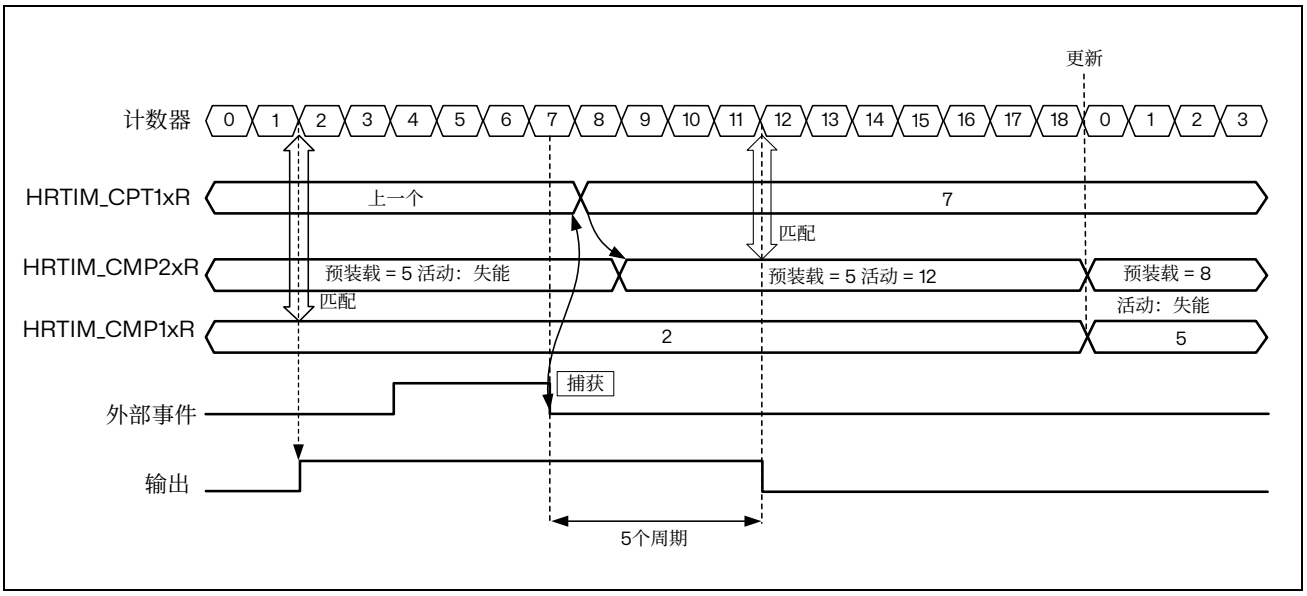



图 21.12 自动延迟比较

使用常规比较通道（例如比较 1）进行输出置位：计数器与比较寄存器的内容匹配后，输出会立即变为有效状态。

使用延迟比较进行输出复位：仅当发生了捕获事件时，才会生成比较事件。如果计数器与延迟比较值（计数值值=4）匹配，则不会生成事件。捕获事件由外部事件触发后，捕获寄存器的内容会立即与延迟比较值相加，得出新的比较值。示例中，自动延迟值 4 与的捕获值 7 相加，得出自动延迟比较寄存器中的值 11。从此时起，可生成比较事件，并会在计数器等于 11 时生成，比较事件会使输出复位。

自动延迟模式下的重复捕获管理

使能自动延迟模式时（DELCMPx[1:0]=01, 10, 11），将阻止重复捕获。

如果同一计数周期内出现多个捕获请求，只会考虑使用第一个捕获请求来计算自动延迟比较值。仅可在以下条件下进行新捕获：

- 自动延迟比较具有匹配的计数器值（比较事件）
- 计数器已翻转（周期）
- 定时器已复位

更改自动延迟比较值

如果已预装载自动延迟比较值（PREEN 位置 1），则无论比较寄存器是何时写入数值的、是否发生了捕获事件（请参见下图，其中，延迟是在计数器翻转时更改的），都会在发生下一更新事件（例如周期事件）时考虑新的比较值。

如果已禁止预装载（PREEN 位复位），则即使比较值在捕获事件发生后进行了修改，也会会立即考虑新的比较值，如下例所示：

1. 在 t1 时刻，DELCMP2 = 1。
2. 在 t2 时刻，CMP2_act = 0x40 => 比较禁止
3. 在 t3 时刻，发生捕获事件，捕获到值 CPTR1 = 0x20。=> 比较使能，比较值= 0x60
4. 在 t4 时刻，CMP2_act = 0x100（在计数器达到值 CPTR1 + 0x40 之前）=> 比较仍处于使能状态，新比较值= 0x120
5. 在 t5 时刻，计数器达到周期值=> 比较禁止，cmp2_act = 0x100

同样，如果 CMP1（CMP3）的值在 DELCMPx = 10 或 11 时发生了变化，并且已禁止预装载：

1. 在 t1 时刻，DELCMP2 = 2。
2. 在 t2 时刻，CMP2_act = 0x40 => 比较禁止
3. 在 t3 时刻，发生 CMP3 事件 - 发生捕获 1 事件之前 CMP3_act = 0x50 => 比较使能，比较值= 0x90
4. 在 t4 时刻，CMP3_act = 0x100（计数器达到 0x90 之前）=> 比较仍处于使能状态，将在 CMP3_act= 0x140 时发生比较 2 事件。

### 半触发模式

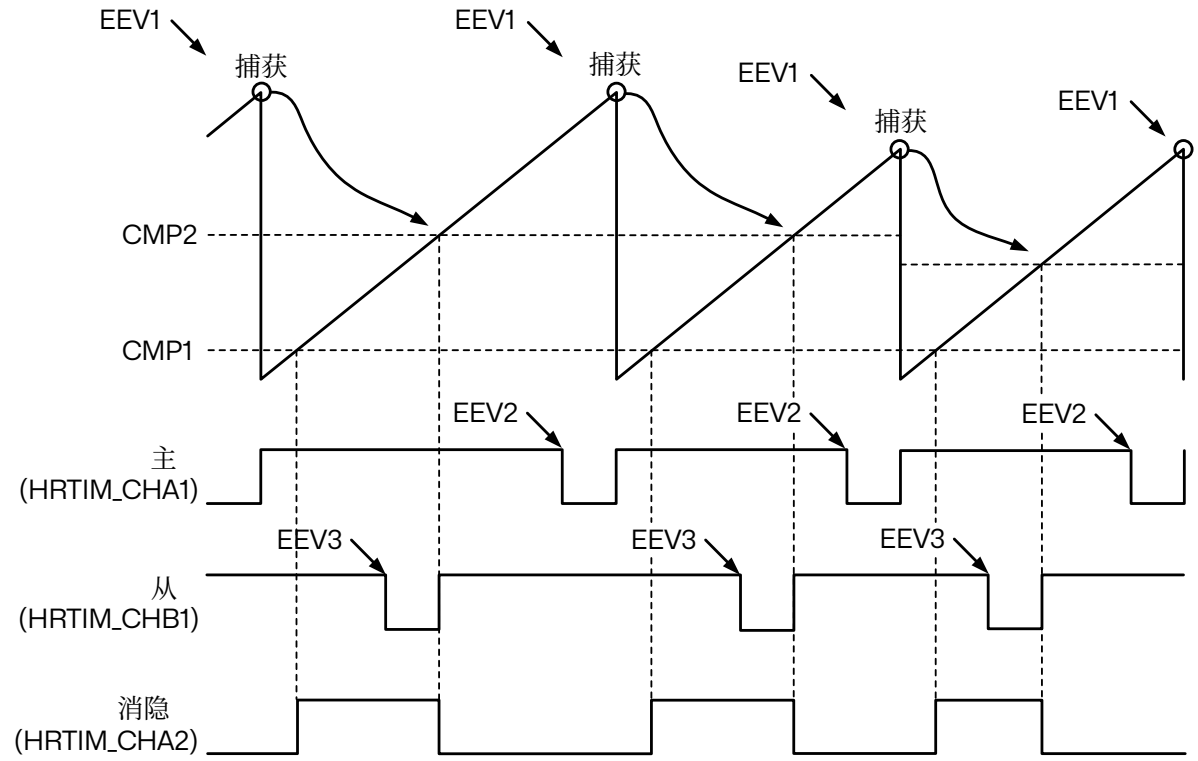
此模式的目的是允许同步两个具有可变频率操作且需要 180°相位偏移的交错转换器。基本原理是建立一个主从系统。

从转换器的同步是基于主转换器前一个开关周期进行连续调整的。这通过使用捕获单元来实现。主转换器的开关周期被捕获，除以 2，然后由硬件存储在比较 2 寄存器中。比较 2 寄存器包含一个值，该值等于捕获周期的一半，即主转换器的开关周期。然后，可以使用比较 2 事件来触发管理从转换器的第二个计时单元。

通过设置 HRTIM_TIMxCR2 寄存器中的 TRGHLF 位来使能此模式。一旦计时器开始运行（TxEN 位设置），该位就不能更改。

半触发模式不能与使用 CMP2 的其他模式（双通道 DAC 触发、交错和平衡空闲模式）同时使用。

用户可以写入 CMP2 的初始值，但一旦触发第一次捕获，该值就会被忽略。当 TRGHLF 位被复位时，CMP2 的预加载机制将被禁用。




HRTIM_CHA1
- 在EEV1时置位（EEV1触发捕获1）
- 在EEV2时复位
- EEV1在TIMA_CMP1时消隐（避免频率失控）

HRTIM_CHB1
- 在CMP2（触发半周期）或TIMA_TA2对EEV1进行消隐时置位
  （如果EEV1发生在CMP1和CMP2之间，则TB1置位，否则在CMP2时置位）
- 在EEV3时复位

HRTIM_CHA2
- 在TA_CMP1时置位
- 在TA_CMP2时复位

图 21.13 半触发模式样例

*注：在半触发模式下，比较2 寄存器由硬件控制，写入该寄存器不会产生任何效果。但是，写入的值会存储在预加载寄存器中，并在退出该模式后的更新事件上生效。*

### 推挽模式

该模式的主要目的是使用推挽拓扑驱动转换器。如果需要使用延迟空闲保护（通常用于谐振转换器），也需要使能该模式（请参考章节：延迟保护）。

通过将 HRTIM_TIMxCR 寄存器中的 PSHPLL 位置 1，即可使能推挽模式。

在该模式下，会按照周期将纵横开关生成的信号交替地施加到输出 1 和输出 2 上（如果信号施加到输出 1 上，则输出 2 保持其无效状态，反之亦然）。重定向速率（推挽频率）由定时器的周期事件定义，如下图所示。推挽周期是定时器计数周期的两倍。

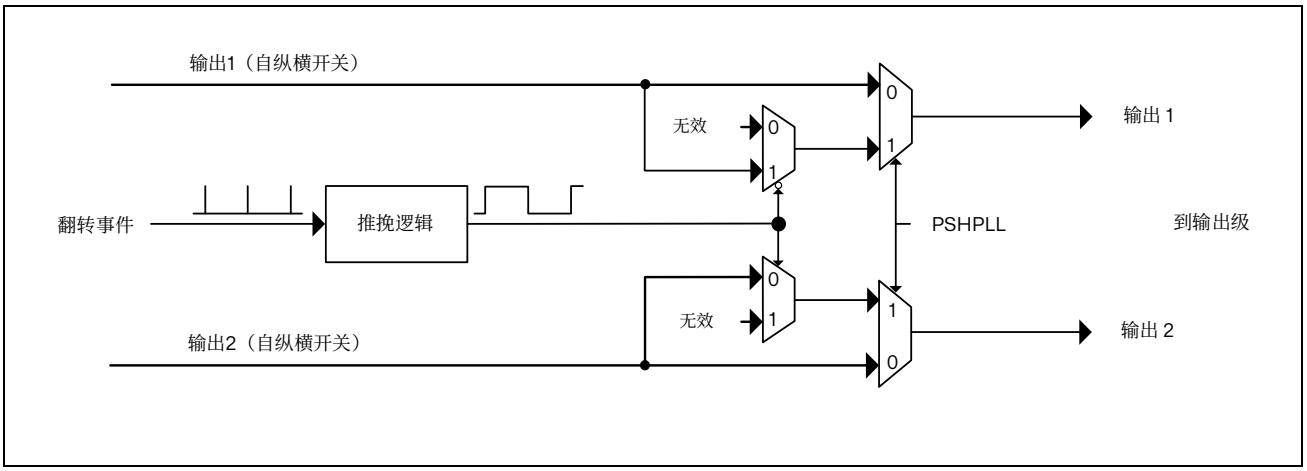



图 21.14 推挽模式框图

推挽模式在定时器以连续模式和单次触发模式运行时都可用。要停止推挽操作并在重新使能之前重置计数器，必须禁用定时器。

为了获得正确的行为，必须选择作为计数器复位源的事件来设置（或复位）输出。如果输出在周期时设置，则必须设置输出；否则，必须复位它。如果不这样做，输出从其无效周期切换到有效周期时可能会出现错误（可能会意外上升或保持低位）。

两路输出的信号波形由 HRTIM_SETxyR 和 HRTIM_RSTxyR 定义。需要 HRTIM_SETx1R = HRTIM_SETx2R 且 HRTIM_RSTx1R = HRTIM_RSTx2R，才能使两路输出的波形完全相同，并实现平衡的操作。不过，仍可对两路输出进行不同的编程，以实现其他用途。

HRTIM_TIMxISR 中的 CPPSAT 状态位指示目前哪一路输出上的信号有效。CPPSTAT 在推挽模式禁用时复位。

在下图中提供的示例中，定时器内部波形的定义如下：

- 发生周期事件时，输出置位
- 发生比较 1 匹配事件时，输出复位

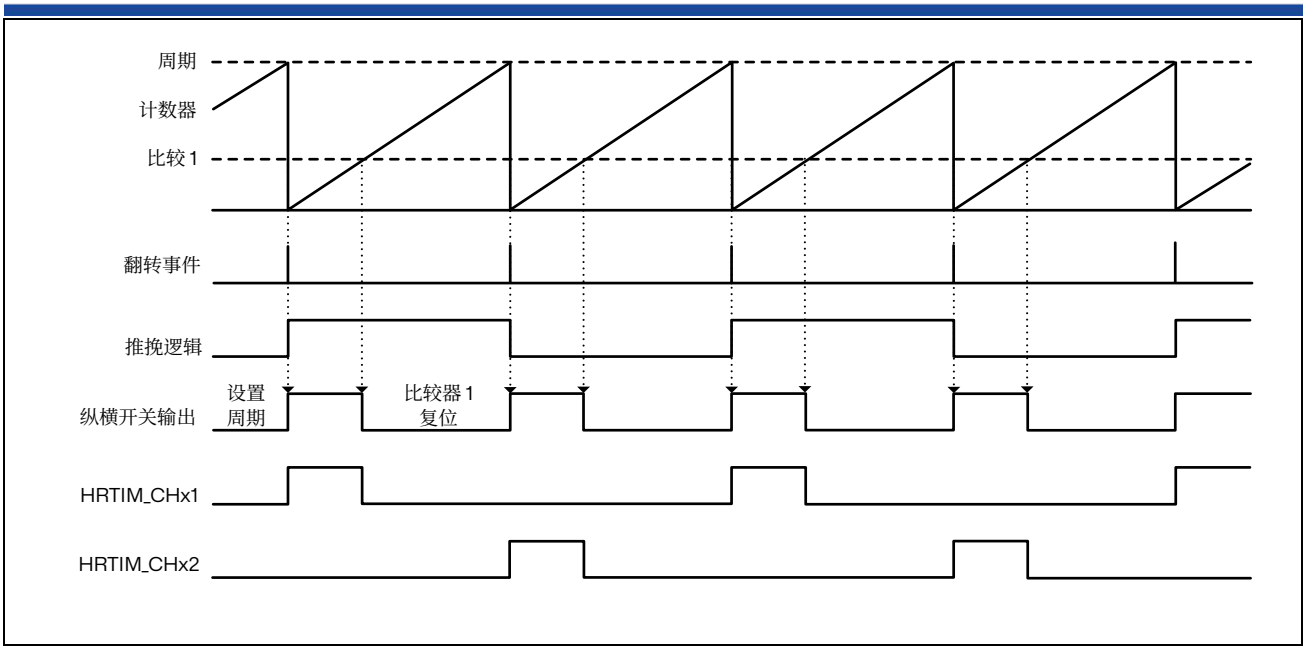



图 21.15 推挽模式样例

下图展示了在推挽模式下如何插入死区时间，包括正死区时间和负死区时间。在这种情况下，输出不再进行互补，死区时间分别应用于每个输出（纵横开关的输出 1 和输出 2 都被使用）。

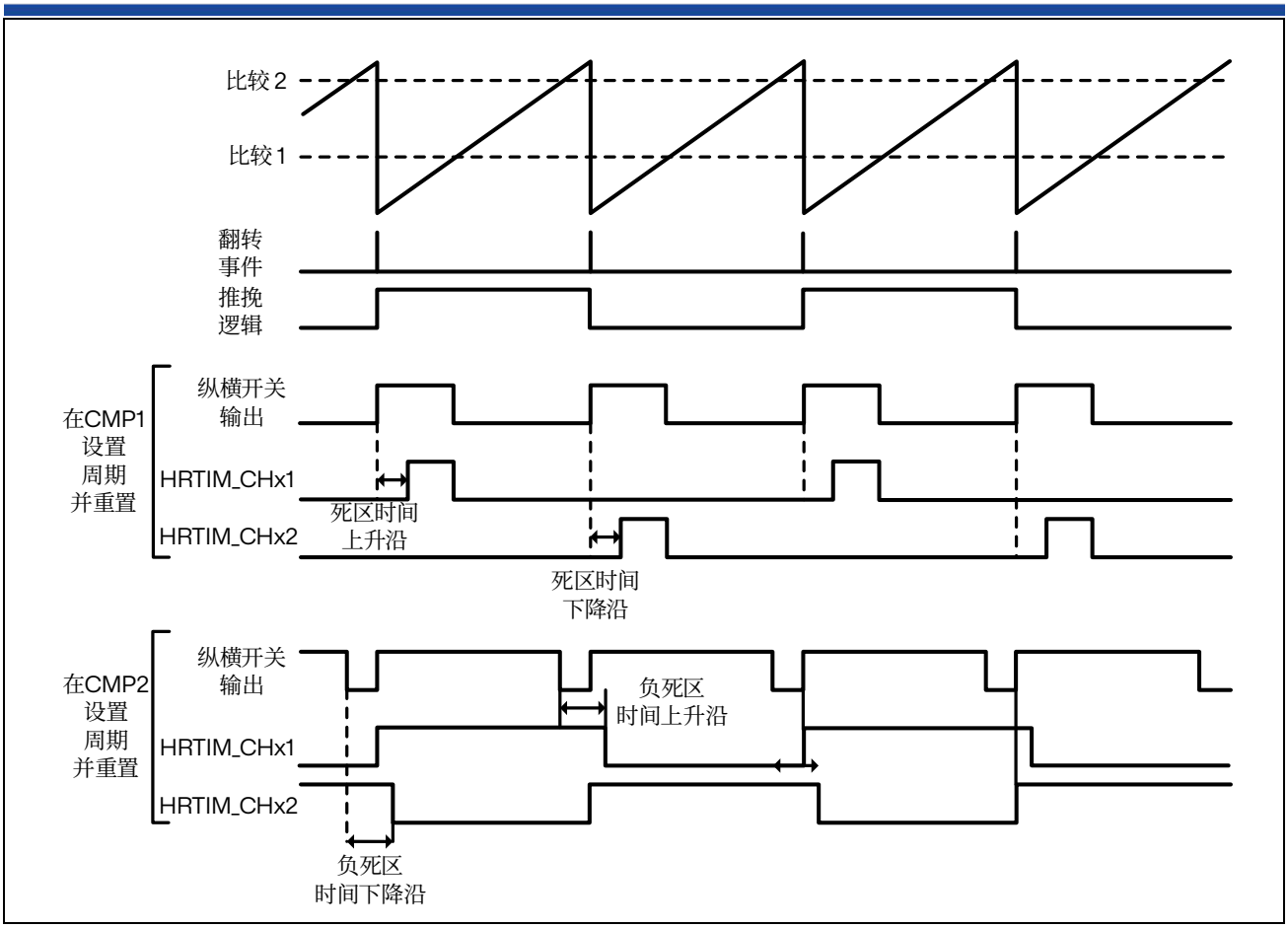



图 21.16 带死区的推挽

**死区时间**

死区插入单元可通过单个参考波形生成一对互补信号，并且有效状态跳变之间的延迟可编程。这通常用于使用半桥或全桥的拓扑，可简化软件操作流程：只需要对 1 个波形进行编程和控制即可驱动两路输出。

死区插入通过将 HRTIM_OUTxR 寄存器的 DTEN 位置 1 来使能。互补信号基于为输出 1 定义的参考波形构建而成，使用 HRTIM_SETx1R 和 HRTIM_RSTx1R 寄存器：如果 DTEN 位置 1，则 HRTIM_SETx2R 和 HRTIM_RSTx2R 寄存器无效。

可按照与参考波形上升沿与下降沿的关系定义两个死区，如下图所示。

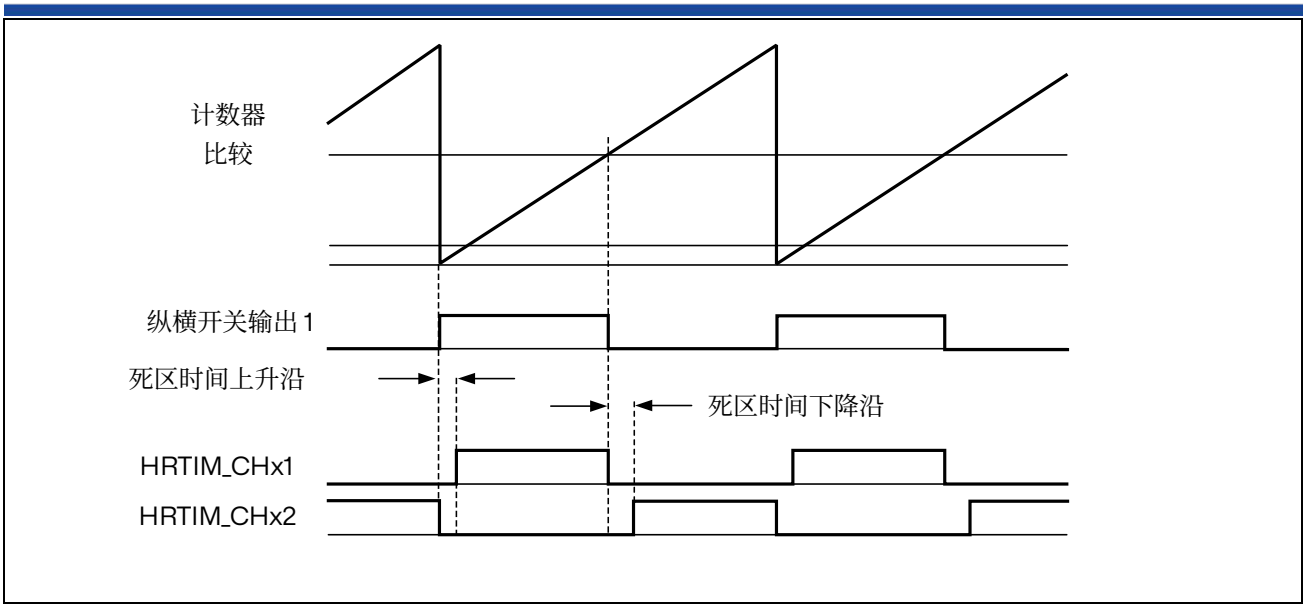



图 21.17 已插入死区的互补输出

如果需要使用一些控制叠加，可定义负死区值，此时要使用死区符号位（HRTIM_DTxR 寄存器中的 SDTFx 和 SDTRx 位）。下图显示了各符号对应的互补信号波形。

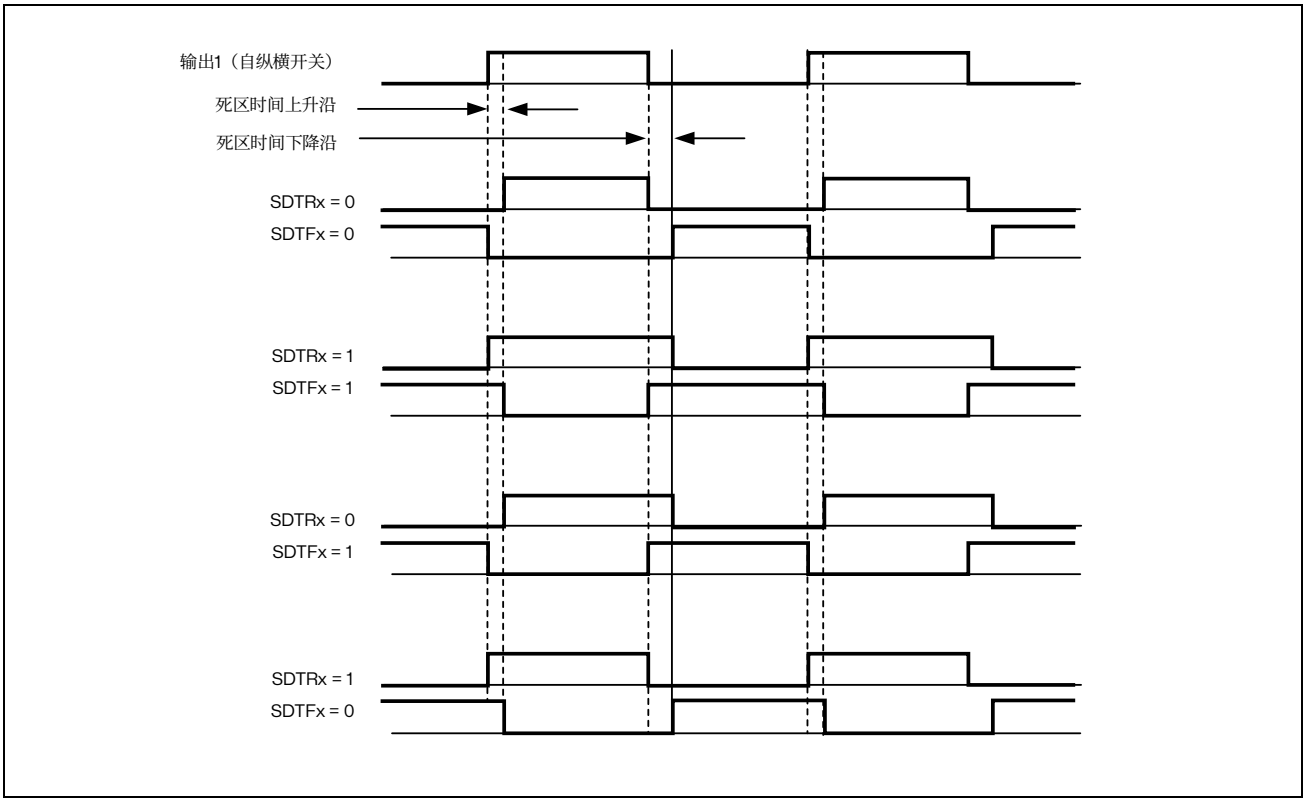



图 21.18 死区插入与死区符号（1 表示负死区）

死区值使用 DTFx[8:0] 和 DTRx[8:0] 位域定义，基于根据 DTPRSC[2:0] 位预分频的特定时钟，具体如下：

$t_{DTx}$ = +/- DTx[8:0] x $t_{DTG}$

其中 x 可以是 R 或 F，而 $t_{DTG}$ = ($2^{(DTPRSC[2:0])}$) x ($t_{HRTIM}$ / 8)。

下表给出了根据预分频值得出的分辨率和最大绝对值。

表 21.15 死区分辨率和最大绝对值

| DTPRSC[2:0] | $t_{DTG}$ | $t_{DTx}$ 最大值 | $f_{HRTIM}$= 192 MHz | |
|---|---|---|---|---|
| | | | $t_{DTG}$ (ns) | $\|t_{DTx}\|$最大值(µs) |
| 000 | $t_{HRTIM}$ / 8 | 511 * $t_{DTG}$ | 0.65 | 0.34 |
| 001 | $t_{HRTIM}$ / 4 | | 1.30 | 0.66 |
| 010 | $t_{HRTIM}$ / 2 | | 2.60 | 1.33 |
| 011 | $t_{HRTIM}$ | | 5.22 | 2.67 |
| 100 | 2 * $t_{HRTIM}$ | | 10.41 | 5.32 |
| 101 | 4 * $t_{HRTIM}$ | | 22.60 | 10.65 |
| 110 | 8 * $t_{HRTIM}$ | | 41.67 | 21.29 |
| 111 | 16 * $t_{HRTIM}$ | | 83.34 | 42.59 |

下两张图展示了在所有死区配置下，死区发生器对脉宽小于死区值的参考波形的处理方式。

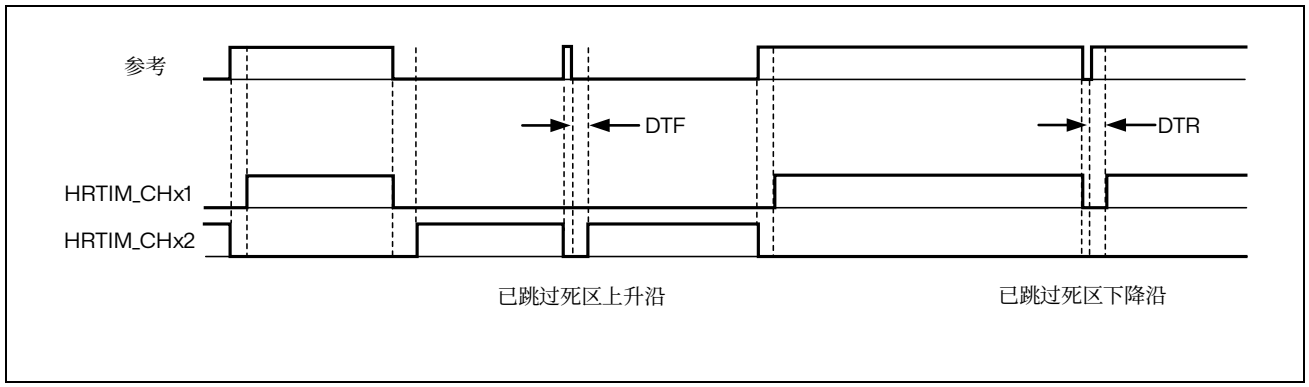



图 21.19 低脉冲宽度的互补输出 (SDTRx = SDTFx = 0)

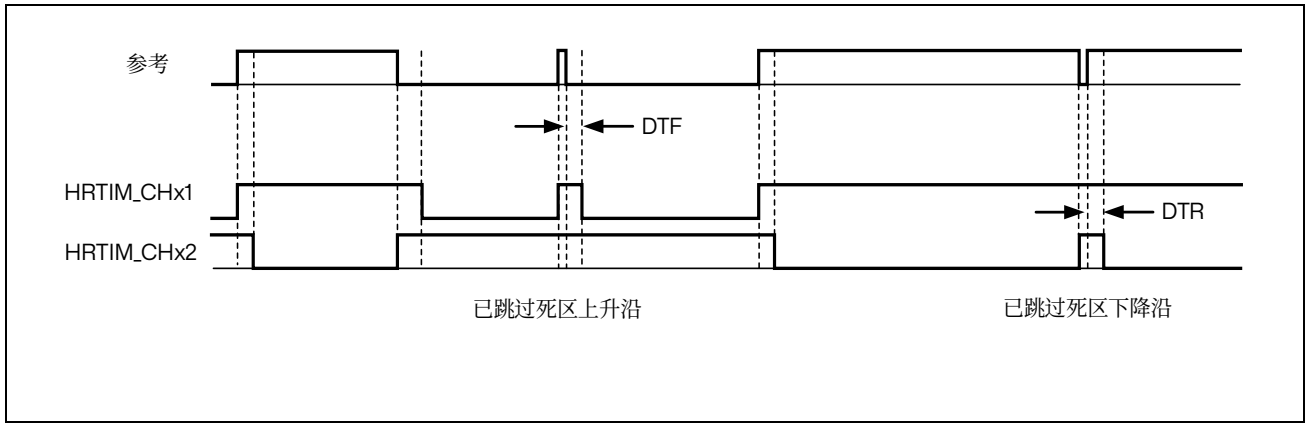



图 21.20 低脉冲宽度的互补输出 (SDTRx = SDTFx = 1)

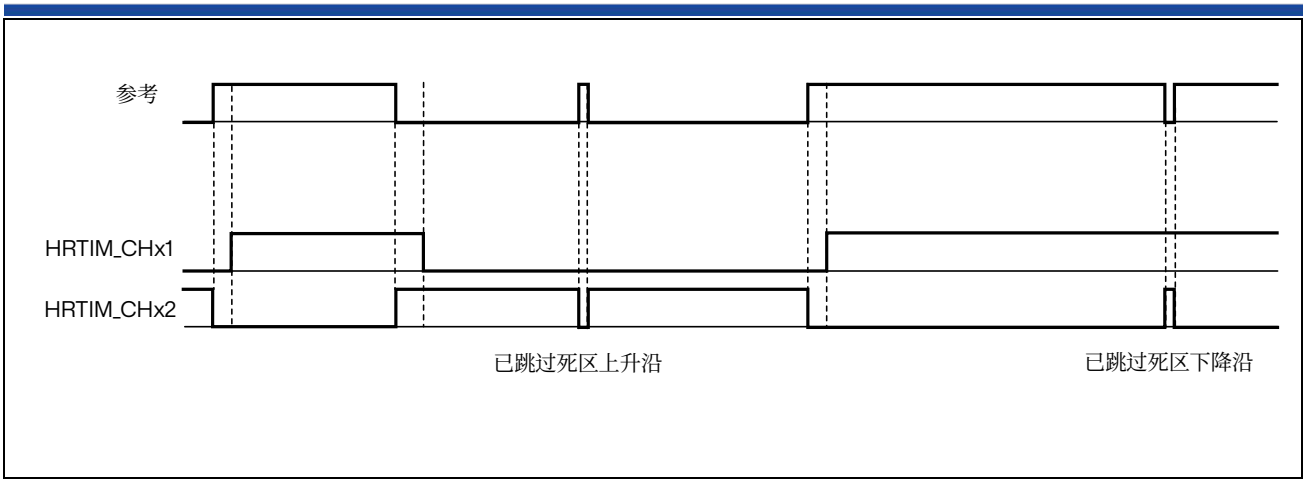



图 21.21 低脉冲宽度的互补输出（SDTRx = 0, SDTFx = 1)

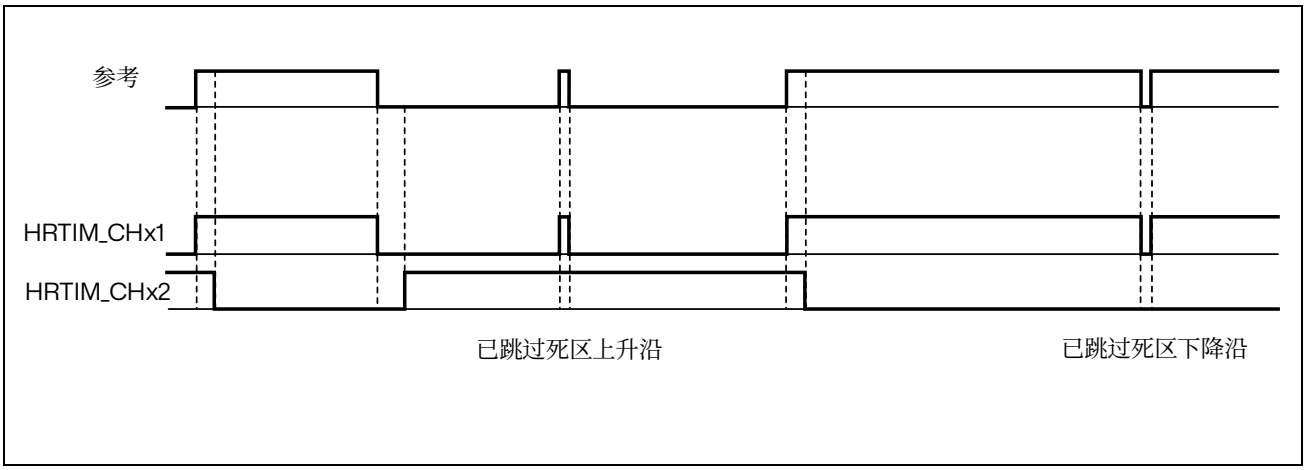



图 21.22 低脉冲宽度的互补输出（SDTRx = 1, SDTFx=0)

出于安全方面的考虑，可使用 DTFLKx、DTRLKx、DTFSLKx 和 DTRSLKx 锁定死区的符号和/或数值，避免对死区寄存器进行误写操作。一旦这些位置 1，在下一次系统复位之前，关联位和位域将变为只读状态。

**警告：**在以下情况下，不得更改 DTEN 位：

– 当定时器被使能时（TxEN 位被置 1）

– 当定时器的输出被另一个定时器置位/复位时（TxEN 复位时）

否则，会导致不可预测的行为。

因此，必须禁用定时器（TxCEN 位复位），并确保相应的输出也被禁用。

对于 DTEN 必须在突发模式使能的同时置 1 以确保进入突发模式后就有死区的这一特殊情况（BME = 1、DIDL = 1、IDLEM = 1），需要在将 DTEN 位置 1 之前通过软件命令将两路输出强制设为其 IDLES 状态（SST、RST 位）。这样做是为了避免在死区尚未使能之前就进入突发模式时会造成的不良后果。

### 21.3.5 主定时器

主定时器的主要用途是向 6 个定时单元提供公用信号，以便进行同步或置位/复位输出。主定时器不会直接控制任何输出，但仍可间接地供置位/复位纵横开关使用。

下图提供了主定时器的概览。

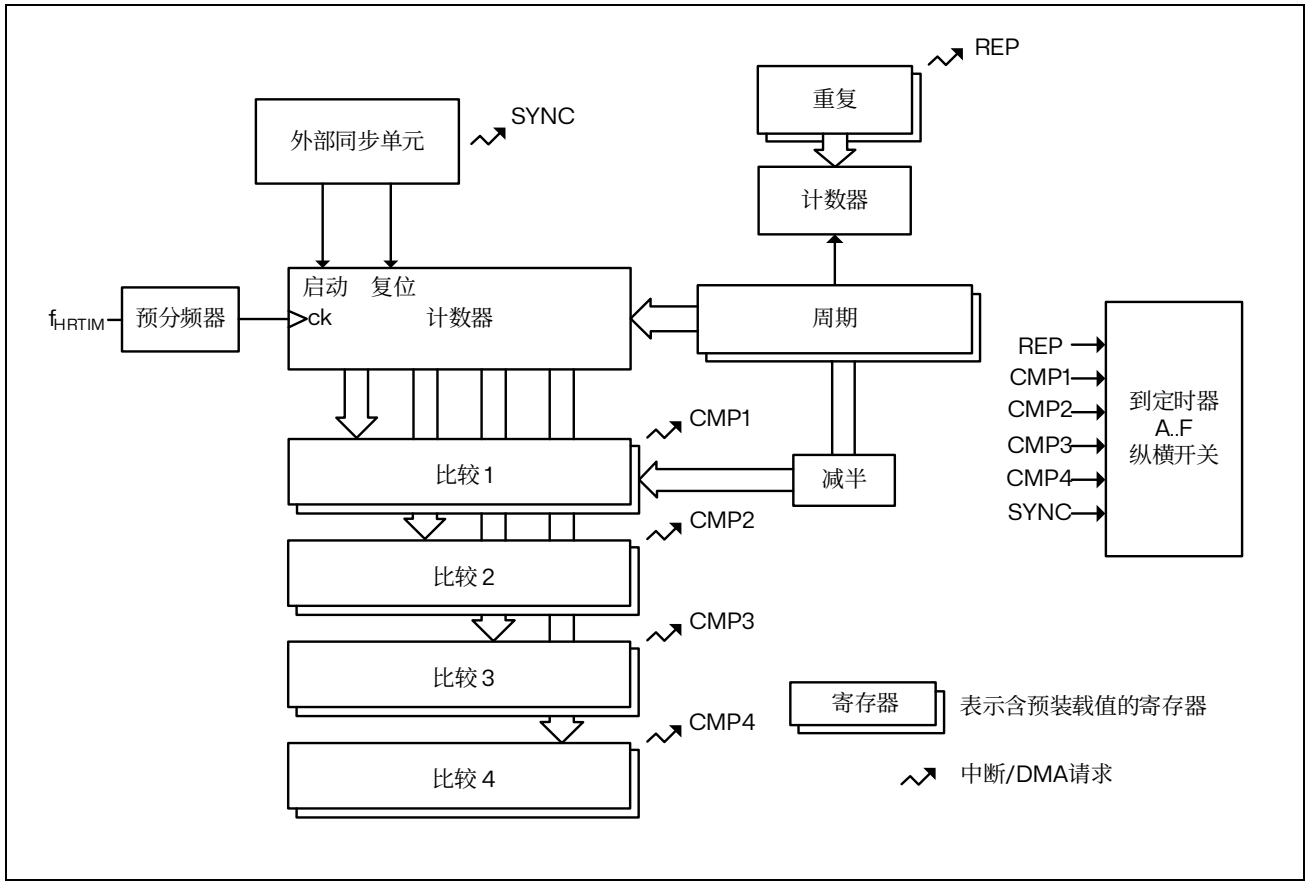



图 21.23 主定时器概览

主定时器采用的架构与定时单元非常相似，二者的不同之处如下：

- 主定时器未关联输出，也没有输出相关的控制
- 主定时器没有自己的纵横开关单元，也没有推挽或死区模式
- 主定时器只能通过外部同步电路复位
- 主定时器没有捕获单元，也没有自动延迟模式
- 主定时器不含外部事件消隐和窗口电路
- 主定时器的中断/DMA 请求数量有限：比较 1..4、重复、寄存器更新和外部同步事件

主定时器控制寄存器包含主定时器和 A..F 定时单元的所有定时器使能位。这样可通过单次写访问同时启动所有定时器。

主定时器还会利用 MCU 内部和外部（输入/输出）资源处理整个 HRTIM 定时器的外部同步（请参见章节：将 HRTIM 与其他定时器或 HRTIM 实例同步）

主定时器控制寄存器的映射偏移与定时单元寄存器的偏移相同。

### 21.3.6 上下计数模式

HRTIM 原生设计为向上计数器。然而，它也提供了一种带有上下计数器的操作模式，也称为中心对齐模式。

这种模式通过 HRTIM_TIMxCR2 寄存器中的 UDM 位来使能。一旦定时器开始运行(TxEN 位被设置)，就不应更改此位。此模式仅适用于 TIMA..F。主定时器仅工作在向上计数模式。

并非所有 HRTIM 功能都支持上下计数。本节将详细介绍与向上计数模式相比的功能差异。

在上下计数模式中，HRTIM_PERxR 中的周期必须预加载（或静态）。它只能在周期事件或计数器复位时更新。

设置/复位纵横开关的编程方式如下：

来自定时单元的事件将根据计数器的向上/向下方向来设置/复位输出：

- 如果事件在 HRTIM_SETxyR 寄存器中使能，则它在向上计数期间设置输出，在向下计数期间复位输出。
- 如果事件在 HRTIM_RSTxyR 寄存器中使能，则它在向上计数期间复位输出，在向下计数期间设置输出。
- 如果事件同时在 HRTIM_SETxyR 和 HRTIM_RSTxyR 寄存器中使能，则输出会切换。

这适用于：

- 定时单元：周期、比较 1..4、寄存器更新（6 个事件）
- 主定时器：周期、比较 1..4、HRTIM 同步（6 个事件）
- 所有其他定时单元（例如，对于定时器 A 的定时器 B..F）：TIMEVNT1..9（表 218 中描述的 9 个事件）

下图展示了如何生成基本波形。

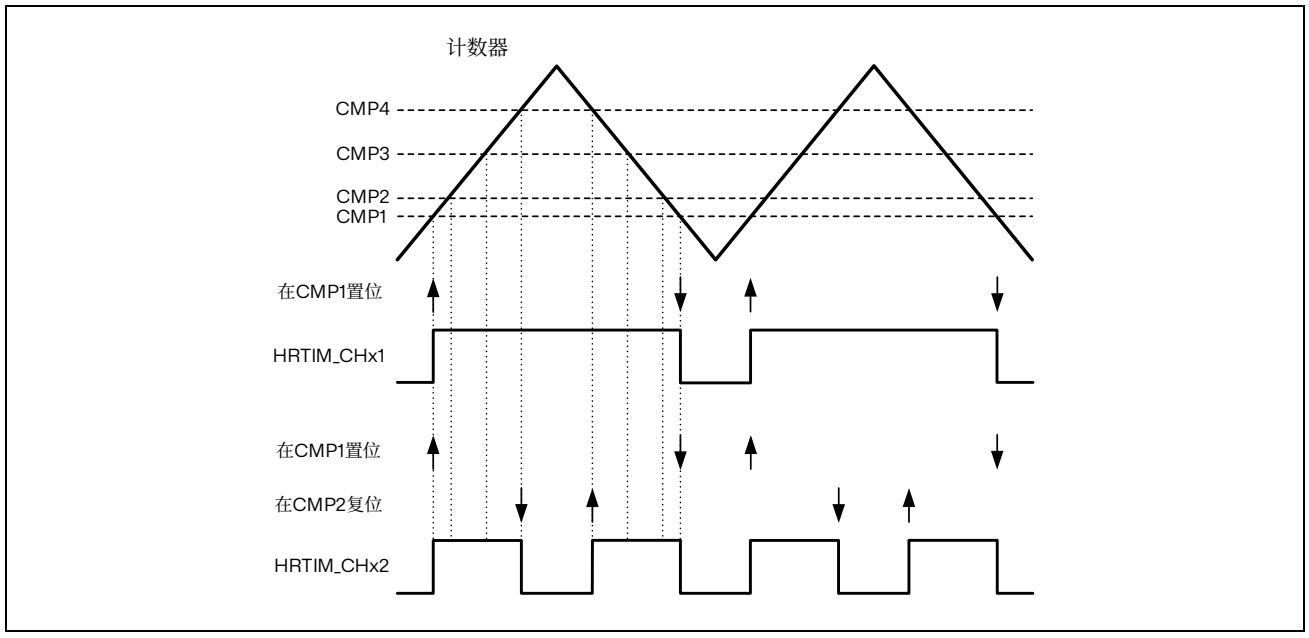



图 21.24 在上下计数模式时的基本对称波形

下图展示了如何使用四个可用的比较单元和切换模式来生成一些更复杂的波形。

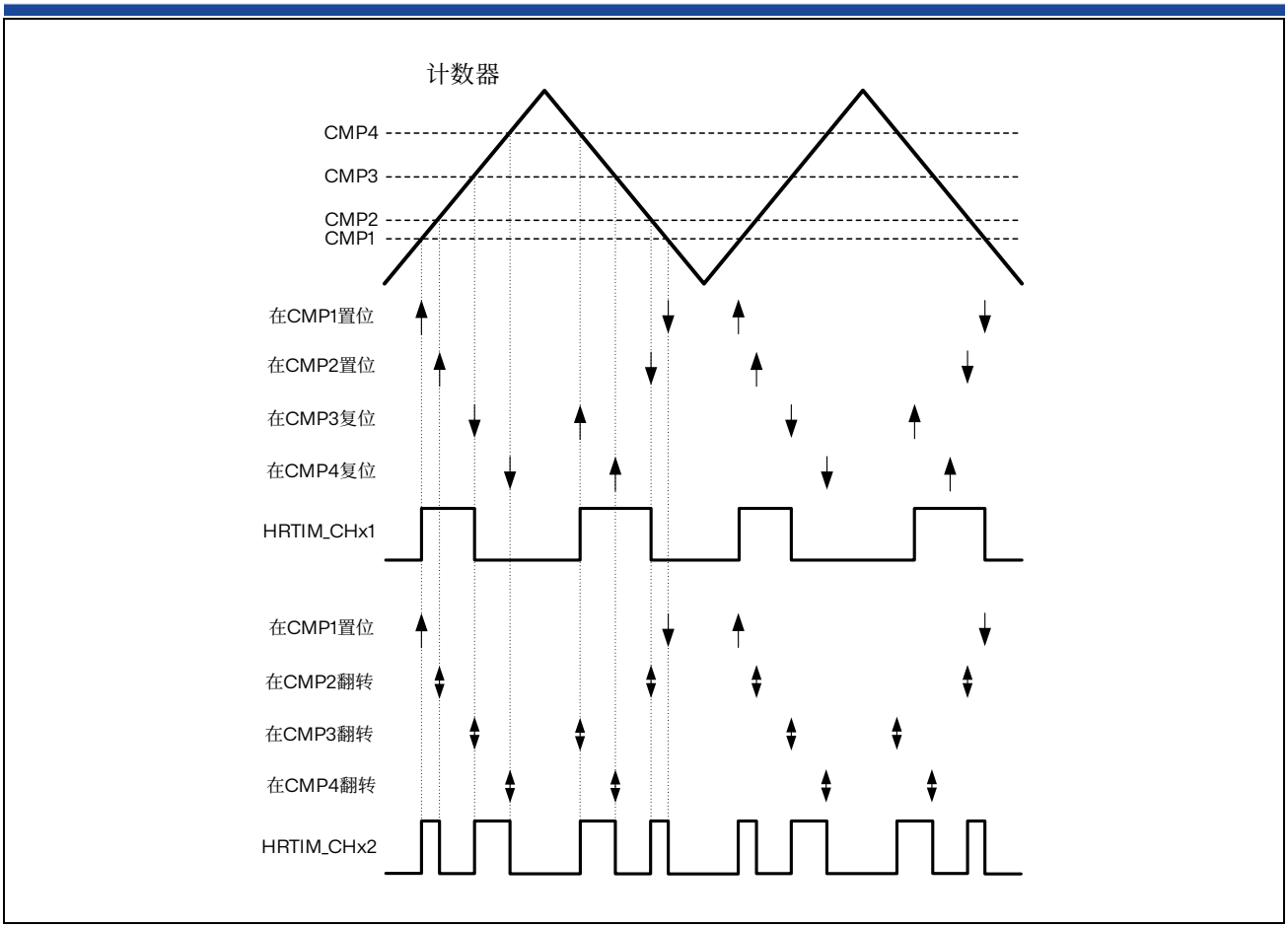



图 21.25 在上下计数模式时的复杂对称波形

下图示了如何生成一个不对称波形。在这种情况下，需要注意的是，为了使波形不对称，比较单元 2 的值必须大于比较单元 1 的值。

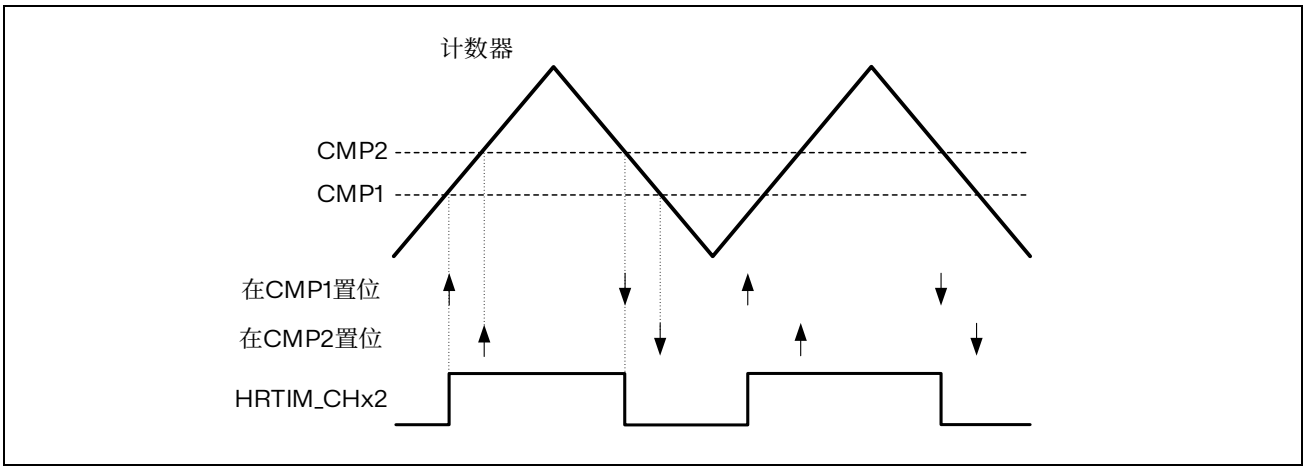



图 21.26 在上下计数模式时的基本对称波形

*注：对于不对称操作，需要确保*CMP2 > CMP1。

软件强制位和外部事件 EXTEVNT1..10 的行为在仅上升计数模式和上下计数模式中是一致的。下图展示了如何响应外部事件来缩短脉冲宽度。

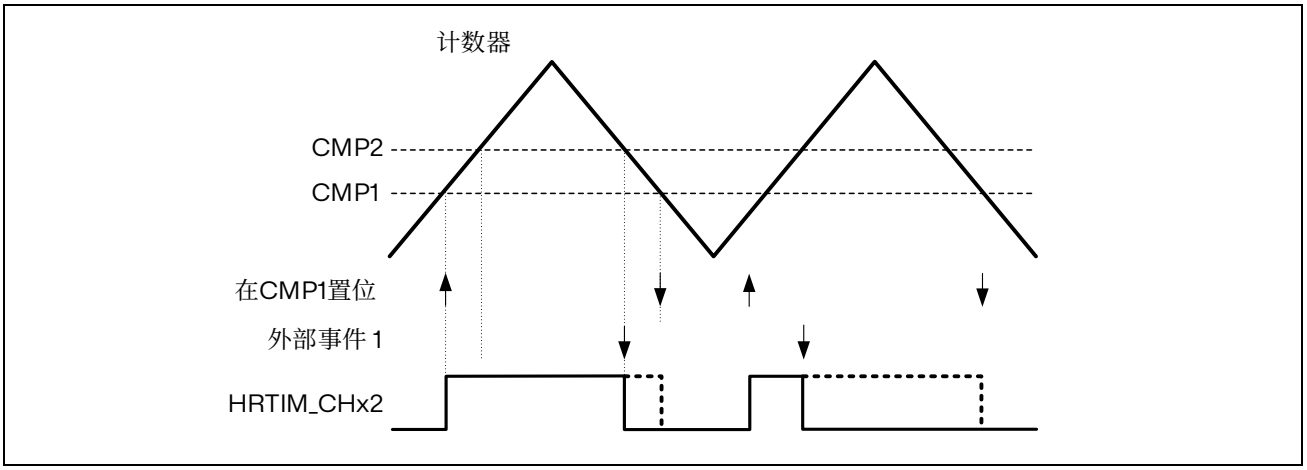



图 21.27 在上下计数模式时的外部事件管理

上下计数模式既可用于连续操作模式，也可用于单次触发（可重触发和非重触发）操作模式。在重置时，计数器会重新从 0 开始。下图展示了在单次可重触发模式下，定时器 B 的计数器行为。

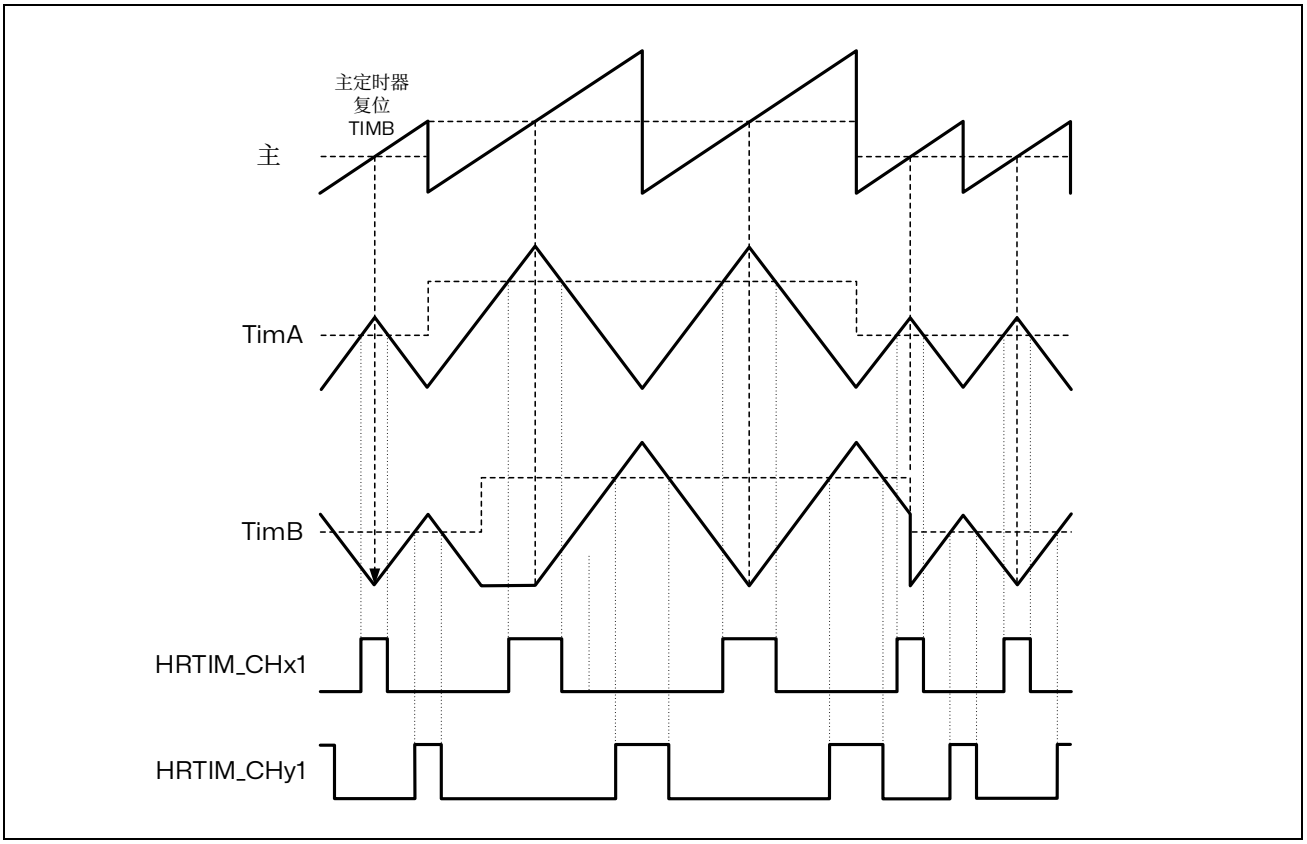



图 21.28 纵横的上下计数操作

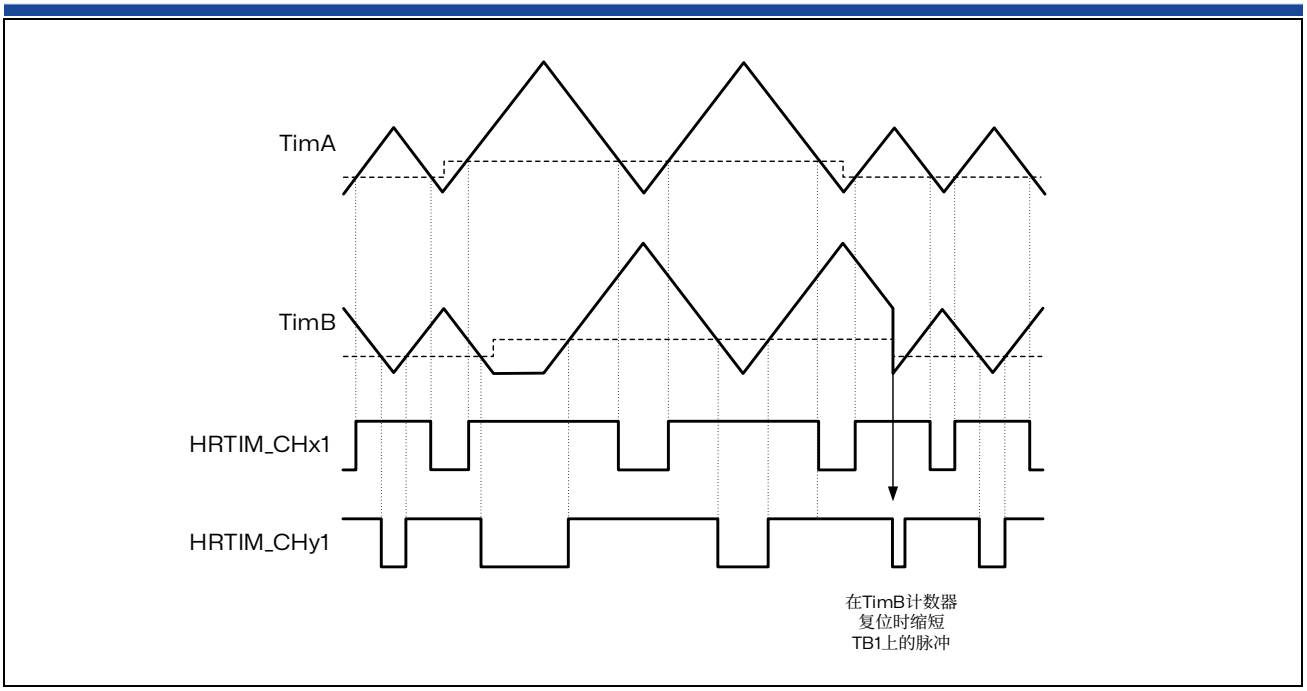



图 21.29 交叉的上下计数操作

*注：在上下计数模式下，比较值必须低于周期值（*TIMx_PER - 0xC0 *如果* CKPSC[2:0] = 0，TIMx_PER - 0x60 *如果* CKPSC[2:0] = 1，TIMx_PER - 0x30 *如果* CKPSC[2:0] = 2，*…），且至少为* $f_{HRTIM}$ *时钟的*3 *个周期。这适用于在定时单元内部生成的比较事件。对于在其他定时单元中生成的比较事件，必须避免在计数器方向改变（计数器为*0、*周期事件或计数器重置）的* $f_{HRTIM}$ *时钟的*1 *个周期之内发生任何事件。*

上下计数模式支持以下功能：

- 半模式
- 死区时间插入
- 推挽模式，当计数器=0 时交替进行推挽操作（见下图）。
- 延迟空闲
- 突发模式
- 使用“大于”比较的 PWM 模式（见下图）。

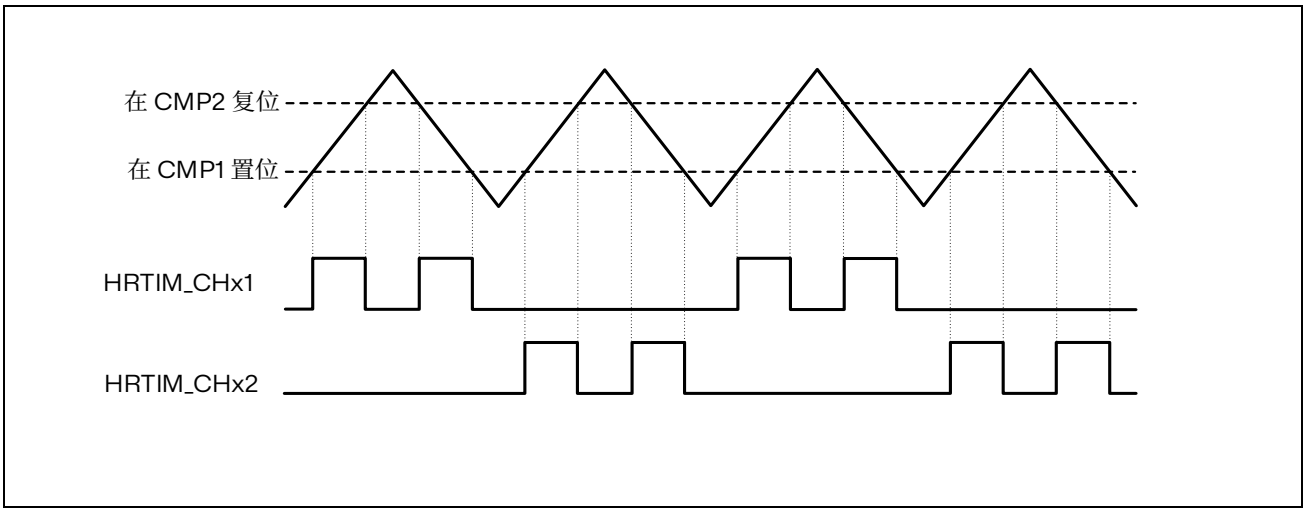



图 21.30 推挽上下模式样例

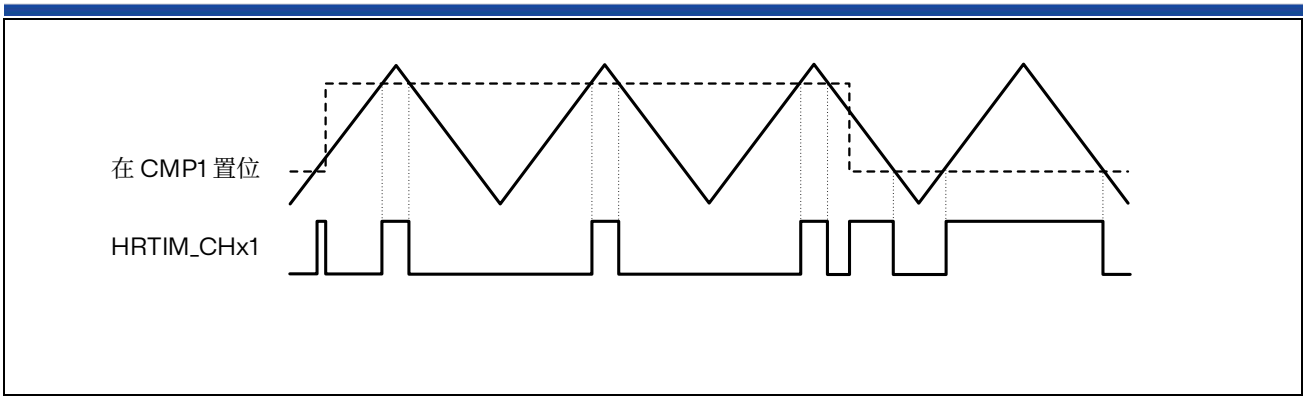



图 21.31 带“大于”比较功能的的上下计数模式

**警告：**上下计数模式不支持以下功能：

- 自动延迟模式
- 平衡空闲
- 半触发模式

捕获功能在以下方面有所差异：

- 在上升计数时，捕获值是从计数开始计算的
- 在下降计数时，捕获值是以 PER 事件为参考的
- 捕获寄存器的第 16 位保存计数方向状态

在上下计数模式下，计数器翻转事件的定义不同，以支持各种操作条件。它可以在以下情况下生成：

- 当计数器等于 0 时（“谷值”模式）
- 当计数器达到在 HRTIM_PERxR 中设置的周期值时（“峰值”模式）
- 当满足上述两个条件中的任何一个时（即 0 或 HRTIM_PERxR 的值）

在 HRTIM 中，此事件用于多种目的。生成模式（谷值、峰值或两者）可以根据目标单独编程。下表总结了使用案例以及 HRTIM_TIMxCR2 寄存器中与之相关的翻转模式（xxROM[1:0]）编程位。

*注：对于同时考虑重置和翻转的事件（*IRQ/DMA *和*TxRSTU*），*ROM[1:0]*仅影响翻转的生成。无论*ROM[1:0]*的值如何，重置事件始终会被考虑。*

表 21.16 翻转事件目标和模式编程

| 使用翻转事件 | 编程位 |
|---|---|
| 输出设置/重置 | OUTROM[1:0] |
| 注册内容更新触发器(从预加载转移到活动) | ROM[1:0] |
| IRQ 和/或 DMA 请求触发器 | ROM[1:0] |
| 突发模式时钟源和/或突发启动触发器 | BMROM[1:0] |
| ADC 触发器(详情请参阅 ADC 后缩放器部分) | ADROM[1:0] |
| 外部事件过滤 | ROM[1:0] |
| 重复计数减量 | ROM[1:0] |
| 故障和事件计数器 | FEROM[1:0] |

翻转事件的生成定义如下，根据 xxROM[1:0]位域的设置：

- xxROM[1:0] = 00：当满足两个条件中的任何一个时（即 0 或 HRTIM_PERxR 的值）生成事件
- xxROM[1:0] = 01：当计数器等于 0 时（“谷值”模式）或计数器被重置时生成事件
- xxROM[1:0] = 10：当计数器达到在 HRTIM_PERxR 中设置的周期值时（“峰值”模式）生成事件

下图展示了具有周期事件设置和 OUTROM[1:0] = 10 的推挽上下模式。

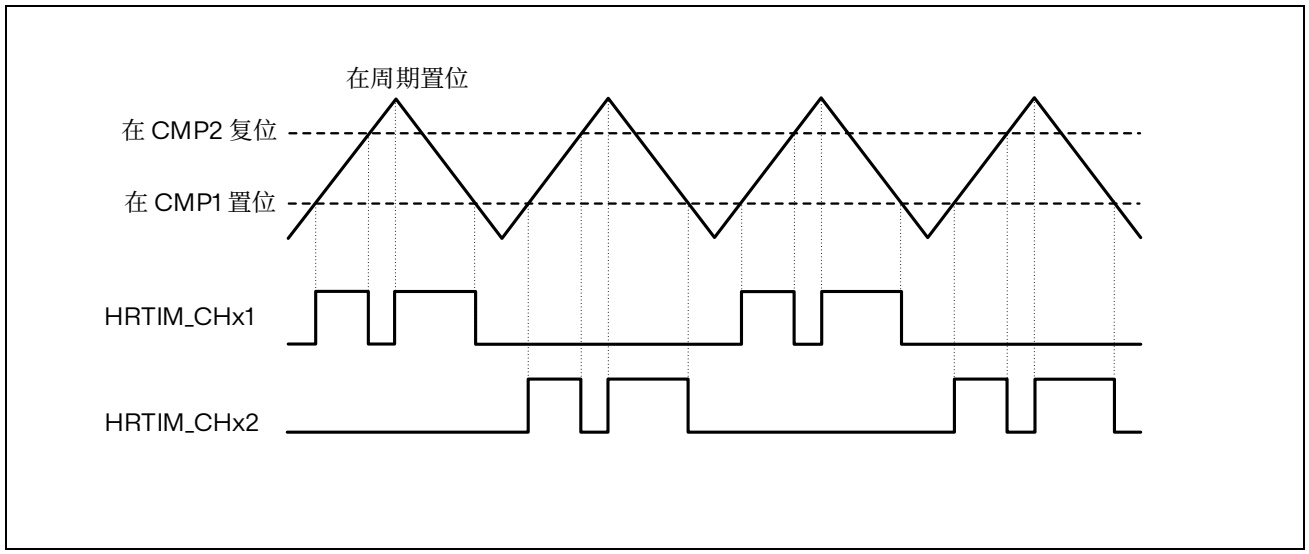



图 21.32 输出在周期事件上设置的上下模式，OUTROM[1:0]=10

下图展示了在上下计数模式下重复计数器是如何递减的。

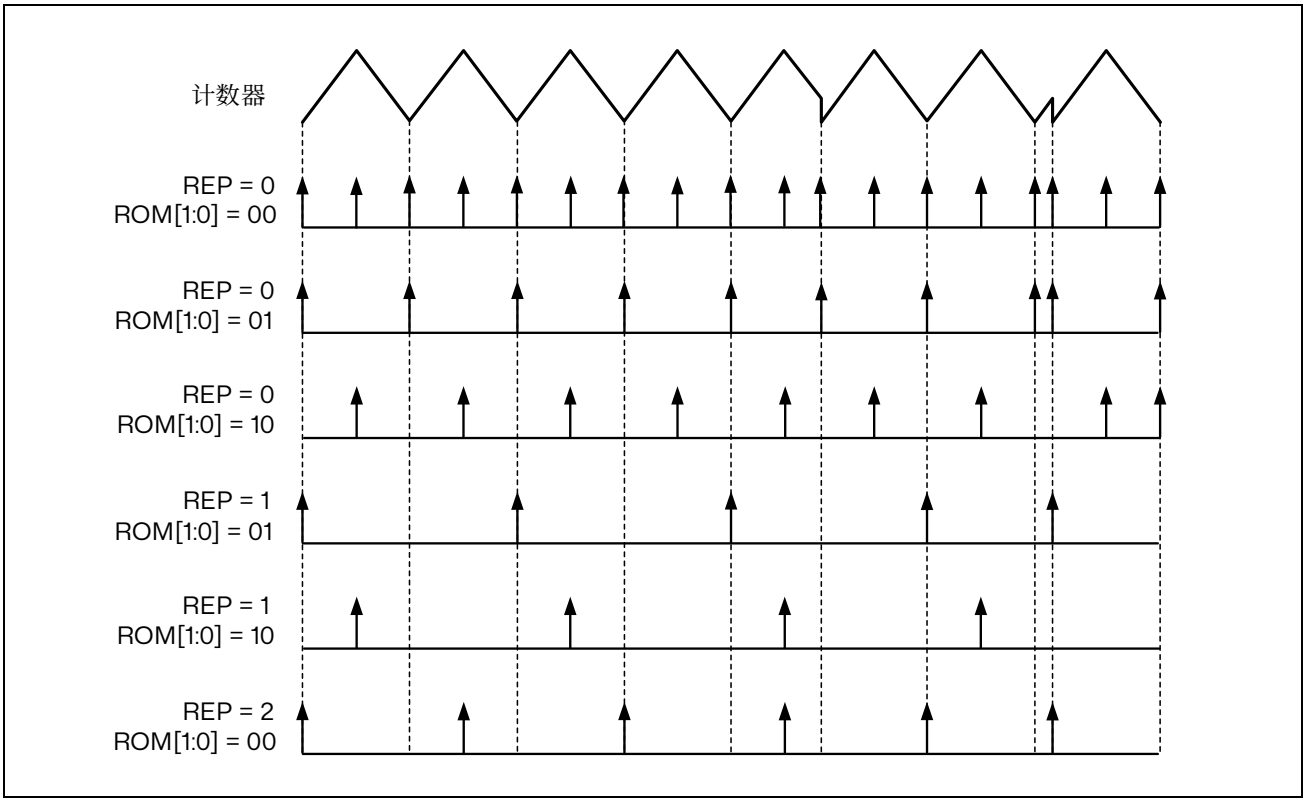



图 21.33 在上下计数模式时重复计数器的表现

双通道 DAC 触发器在上升计数模式下的工作方式相同。

事件消隐和窗口化功能有所不同，以便在输出脉冲内的可编程时间内执行消隐或窗口化操作。EExFLTR[3:0]编码取决于 UDM 位的设置，如下表所示。每当使用翻转事件进行消隐或窗口化时，ROM[1:0]编程适用于定义其生成时间。

表 21.17 EExFLTR[3:0]代码取决于 UDM 位设置

| EE1FLTR[3:0] | 向上计数模式<br>(UDM = 0) | 向上向下计数模式<br>(UDM = 1) |
|---|---|---|
| 0010 | 消隐从计数器复位/翻转<br>到比较 2 | 消隐从比较 1 到比较 2，<br>仅在向上计数阶段 |
| 0100 | 消隐从计数器复位/翻转<br>到比较 4 | 消隐从比较 3 到比较 4，<br>仅在向上计数阶段 |
| 1101 | 窗口化从计数器复位/翻转<br>到比较 2 | 窗口化从比较 2 到比较 3，<br>仅在向上计数阶段 |
| 1110 | 窗口化从计数器复位/翻转<br>到比较 3 | 窗口化从比较 2 到比较 3，<br>仅在向下计数阶段 |
| 1111 | 来自另一个定时单位的窗口化：<br>TIMWIN 源(详见后续表格) | 窗口化从向上计数阶段的<br>比较 2 到向下计数阶段的比较 3 |

### 21.3.7 置位/复位事件优先级和窄脉冲管理

本节介绍了在 3 个连续 $t_{HRTIM}$ 周期内出现多个置位和/或复位请求时，输出波形是如何产生的。

#### 情况 1：时钟预分频器 CKPSC[2:0] < 5

在每个 $t_{HRTIM}$ 周期内，会执行一个仲裁过程，分三个步骤：

1. 对于每个有效事件，会确定所需输出跳变（置位、复位或切换）。
2. 在有效事件之间执行预定义的仲裁（从最高到最低优先级 CMP4 → CMP3 → CMP2 → CMP1 → PER，参见章节：并发置位请求/并发复位请求）。
3. 在低分辨率事件和已经赢得预定义仲裁的事件之间，执行基于高分辨率延迟的仲裁，其中复位具有最高优先级。

如果两个不同源发出的置位和复位请求是同时的，则复位操作的优先级最高。如果设置和重置请求之间的间隔小于 2 个 $f_{HRTIM}$ 周期，则行为取决于时间间隔以及与 $f_{HRTIM}$ 时钟的对齐情况，如下图所示。

*注：如果置位和复位请求是同时发生的，并且来自同一个定时单元，则应用* CMPx *优先级，如上述步骤* 2 *所示。例如，假设* CMP2 = CMP4*：*

- *如果* CMP2 *执行置位操作而* CMP4 *执行复位操作，则输出被复位。*
- *如果* CMP2 *执行复位操作而* CMP4 *执行置位操作，则输出被置位。*

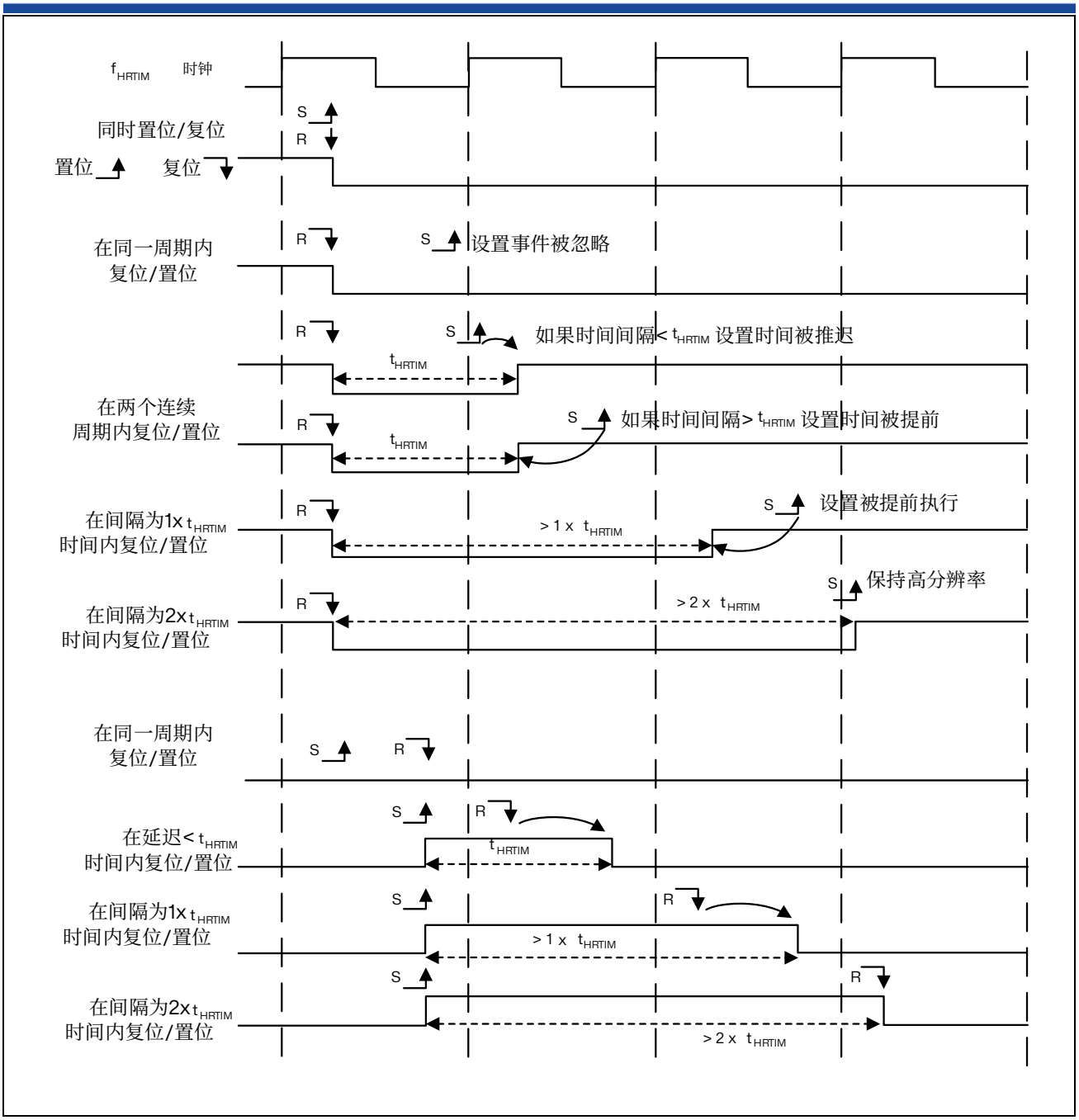



图 21.34 窄脉冲生成的短距离置位/复位管理

如果置位和复位事件是在同一个 $t_{HRTIM}$ 周期内生成的，那么复位事件具有最高优先级，置位事件将被忽略。

如果置位和复位事件是在两个周期之间生成的，且间隔小于 $t_{HRTIM}$ 周期，那么将生成一个 1 个 $t_{HRTIM}$ 周期的脉冲。

如果置位和复位事件是在小于 2 个 $t_{HRTIM}$ 周期的间隔内生成的，那么将生成一个 2 个 $t_{HRTIM}$ 周期的脉冲。

如果置位和复位事件是在 2 到 3 个 $t_{HRTIM}$ 周期之间的间隔内生成的，且间隔超过 2 个完整的 $t_{HRTIM}$ 周期，那么可以使用高分辨率功能。

如果置位和复位事件是在大于 3 个 $t_{HRTIM}$ 周期的间隔内生成的，那么总是可以使用高分辨率。

并发置位请求/并发复位请求

如果为置位事件选择了多个源，则在同一 $f_{HRTIM}$ 时钟周期内出现多个置位请求时，会执行仲裁。

如果相邻定时器 (TIMEVNT1..9) 发出多个请求，则会考虑最先发出的请求。仲裁分 2 步执行，具体取决于：

1. 源（CMP4 → CMP3 → CMP2 → CMP1），
2. 延迟。

如果主定时器在同一 $f_{HRTIM}$ 时钟周期内发出多个请求，则会应用预定义的仲裁，并会考虑单个请求（优先级从高到低）：

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

*注：建议避免从主定时器向给定的定时器生成多个置位（复位）请求，其间隔应小于* 3x $t_{HRTIM}$，*以保持高分辨率。*

如果在同一 $f_{HRTIM}$ 时钟周期内出现多个定时器内部请求，则会应用预定义的仲裁，并会按照以下优先级顺序考虑请求，而不考虑有效定时（优先级从高到低）

CMP4 → CMP3 → CMP2 → CMP1 → PER

*注：实际上，仅当同时使用自动延迟比较* 2 *和比较* 4 *时（也就是由于有效置位/ 复位与外部事件相关联，无法通过优先级确定的情况下），才要首先考虑这一点。在这种情况下，优先级最高的信号必须与* CMP4 *事件相关联*

最后，最高优先级会分配给非定时相关的事件：EXTEVNT1..10、RESYNC（来自 SYNC 事件（SYNCRSTx 或 SYNCSTRTx 置 1 时）或软件复位）、更新和软件置位 (SST)。更新事件被视为具有最大延迟（如果 PSC = 0，则为 0x1F）。

总之，在事件紧密相邻（在同一 $f_{HRTIM}$ 时钟周期内发生）的情况下，有效的置位（复位）事件将在以下事件之间进行仲裁：

- 任何 TIMEVNT1..9 事件
- 来自主定时器的单个源（根据上面给出的固定仲裁）
- 来自定时器的单个源
- “低分辨率事件”

仲裁原则同样适用于并发复位请求。在这种情况下，复位请求的优先级最高。

### 情况 2：时钟预分频器 CKPSC[2:0] ≥ 5

当高分辨率无效时，窄脉冲管理得到简化。

在预分频器时钟周期内发生的置位或复位事件会延迟到预分频时钟的下一有效边沿（针对计数器复位），即使仍在每个 $t_{HRTIM}$ 周期进行仲裁的情况也不例外。

如果同一预分频器时钟周期内发生复位事件后又发生置位事件，将考虑最新事件。

### 21.3.8 外部事件全局条件

HRTIM 定时器可处理并非定时器中生成的事件，此类事件被称为“外部事件”。外部事件来自多个片上或片下源：

- 内置比较器
- 数字输入引脚（通常连接到片下比较器和过零检测器）
- 其他外设的片上事件（ADC 的模拟看门狗和通用定时器触发输出）。

外部事件调节电路可为给定通道选择信号源（使用 4:1 多路复用器），并可将信号源转换为可由纵横开关单元处理的信息（例如，通过在外部事件通道上检测到的下降沿来触发输出复位）。

最多可调节 10 个外部事件通道，这些外部事件通道可同时用于 6 个定时器中的任何一个。由于这种调节通常取决于外部组件（比如过零检测器）和环境条件（滤波器设置通常与应用噪声级和信号有关），因此对于所有定时器是通用的。下图显示了单条通道的调节逻辑概览。

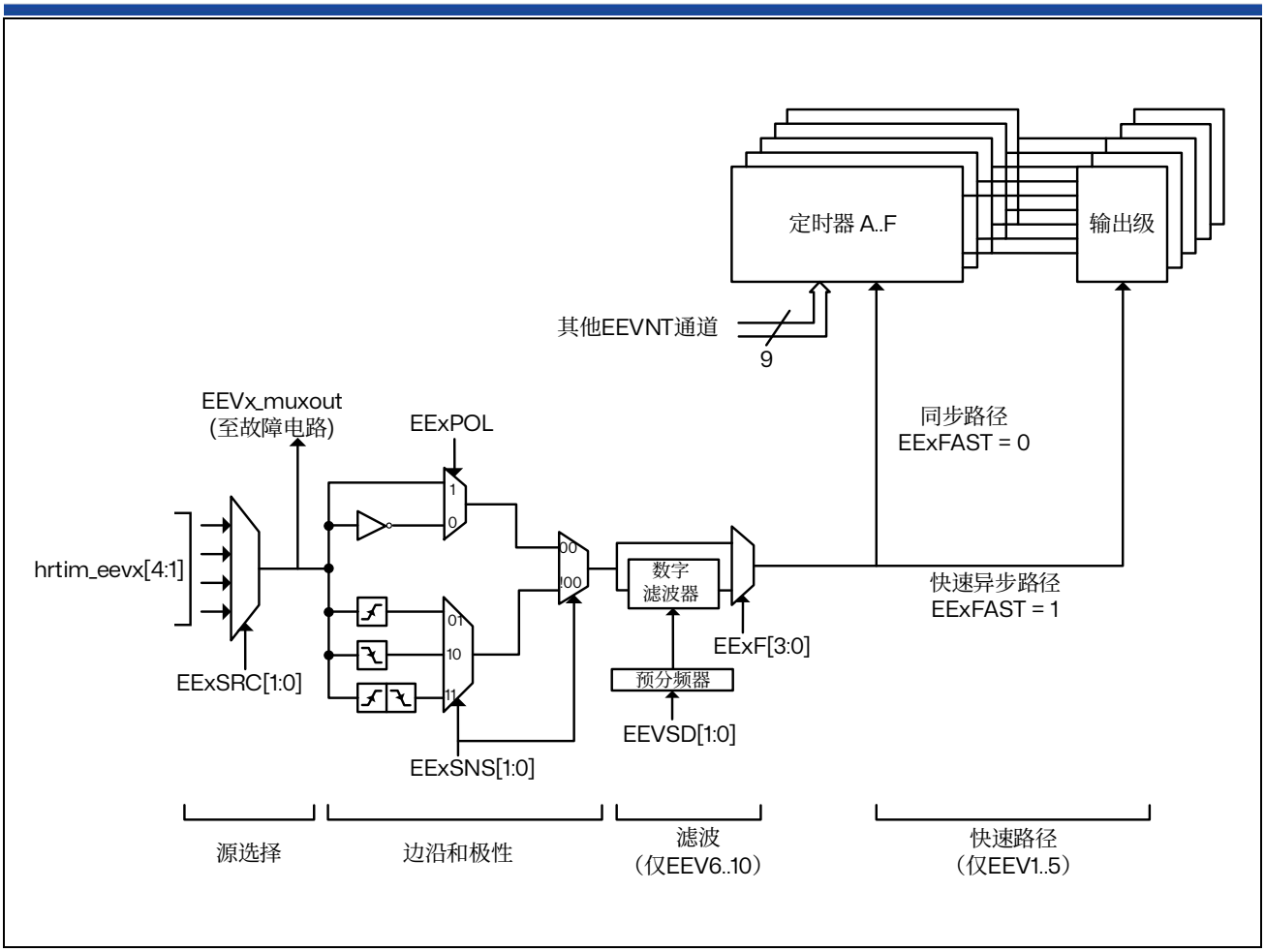



图 21.35 外部事件调节概述(代表 1 个通道)

10 个外部事件是通过 HRTIM_EECR1 和 HRTIM_EECR2 寄存器进行初始化的：

- 使用 EExSRC[1:0]位来选择最多 4 个源
- 使用 EExSNS[1:0]位选择采用电平有效还是边沿有效（上升沿、下降沿或两种边沿）
- 如果选择采用电平有效，则使用 EExPOL 位选择极性
- 使用 EExFAST 位为外部事件 1 到 5 使用低延迟模式（见“外部事件延迟”一节）

*注：即使*EESNS *位已复位（选择电平有效），用作复位、捕获、突发模式、*ADC *触发和延迟保护的触发信号的外部事件也是边沿有效：如果* POL = 0，*触发在外部事件上升沿有效；如果* POL = 1，*触发在外部事件下降沿有效。*

计数器禁止后（TxCEN 位复位），会立即丢弃外部事件，以防止任何输出状态更改和计数器复位，但用作 ADC 触发信号的外部事件除外。

此外，还可以使用 HRTIM_EECR3 寄存器中的 EExF[3:0]位为外部事件 6 到 10 使能数字噪声滤波器。

数字滤波器由计数器组成，需要使用 N 个有效采样来验证输出跳变。如果输入值在计数器达到 N 值之前发生变化，计数器会复位，跳变会被丢弃（视为伪事件）。如果计数器达到 N，跳变被视为有效，并会作为正确的外部事件进行传输。

因此，数字滤波器会向正在进行滤波的外部事件添加延迟，延迟时长视采样时钟和滤波器长度（预期的有效采样数）而定。

采样时钟为 $f_{HRTIM}$ 时钟或由 $f_{HRTIM}$ 预分频而生成的特定时钟 $f_{EEVS}$，通过 HRTIM_EECR3 寄存器中的 EEVSD[1:0]位定义。

下表总结了与 10 个外部事件通道相关联的可用源和特性。

表 21.18 外部事件特征

| 外部事件通道 | 快速模式 | 数字滤波 | 平衡故障<br>定时器 A,B,C | 平衡故障<br>定时器 D,E,F |
|---|---|---|---|---|
| hrtim_eev1[4:1] | 是 | - | - | - |
| hrtim_eev2[4:1] | 是 | - | - | - |
| hrtim_eev3[4:1] | 是 | - | - | - |
| hrtim_eev4[4:1] | 是 | - | - | - |
| hrtim_eev5[4:1] | 是 | - | - | - |
| hrtim_eev6[4:1] | - | 是 | 是 | - |
| hrtim_eev7[4:1] | - | 是 | 是 | - |
| hrtim_eev8[4:1] | - | 是 | - | 是 |
| hrtim_eev9[4:1] | - | 是 | - | 是 |
| hrtim_eev10[4:1] | - | 是 | - | - |

### 外部事件延迟

外部事件调节能够根据性能预期调整外部事件处理时间（以及关联的延迟）：

- 常规工作模式，在该模式下，会在对输出纵横开关进行操作之前，使用时钟对外部事件进行重新采样。此过程会增加一些延迟，但可以访问所有纵横开关功能，从而生成外部触发的高分辨率脉冲。
- 快速工作模式，在该模式下，外部事件与输出操作之间的延迟得到最大限度地缩短。该模式便于实现超快速过流保护等功能。

HRTIM_EECR1 寄存器中的 EExFAST 位可用于定义通道 1 到 5 的工作模式。这会影响输出脉冲上存在的延迟和抖动，具体见下表。

表 21.19 输出置位/复位延迟和抖动与外部事件工作模式的关系

| EExFAST | 响应时间延迟 | 响应时间抖动 | 输出脉冲上的抖动<br>(计数器通过外部事件复位) |
|---|---|---|---|
| 0 | 5 到 6 个<br>$f_{HRTIM}$ 时钟周期 | 1 周期的<br>$f_{HRTIM}$ 时钟 | 无抖动，脉冲宽度保持在高分辨率 |
| 1 | 最小延迟<br>(取决于比较器还是数字输入) | 最短延迟 | 1 个 $f_{HRTIM}$ 时钟周期的抖动，<br>脉冲宽度分辨率可低至 $t_{HRTIM}$ |

当电平检测敏感时，设置（EExSNS[1:0] = 00），才能使用 EExFAST 模式；不能用于边沿检测敏感的设置。

可以对外部事件应用事件过滤（消隐和窗口，EExFLTR[3:0] != 0000）。在这种情况下，EExLTCHx 位必须复位：不支持延迟模式，也不支持窗口超时功能。

*注：相关* EExFAST *位置*1 *后，不得修改外部事件配置（源和极性）。*

快速外部事件不能用于切换输出：必须在 HRTIM_SETxyR 或 HRTIM_RSTxyR 寄存器中使能，而不能在两个寄存器中同时使能。

如果置位事件和复位事件（来自 2 个独立的快速外部事件）同时发生，则复位事件在纵横开关中的优先级最高，输出变为无效状态。

如果 EExFAST 位置 1，则在外部事件发生后的 11 个 $f_{HRTIM}$ 时钟周期内，输出都不能更改。

下两图给出了进行输出置位/复位和计数器复位时对外部事件的实际响应时间示例。

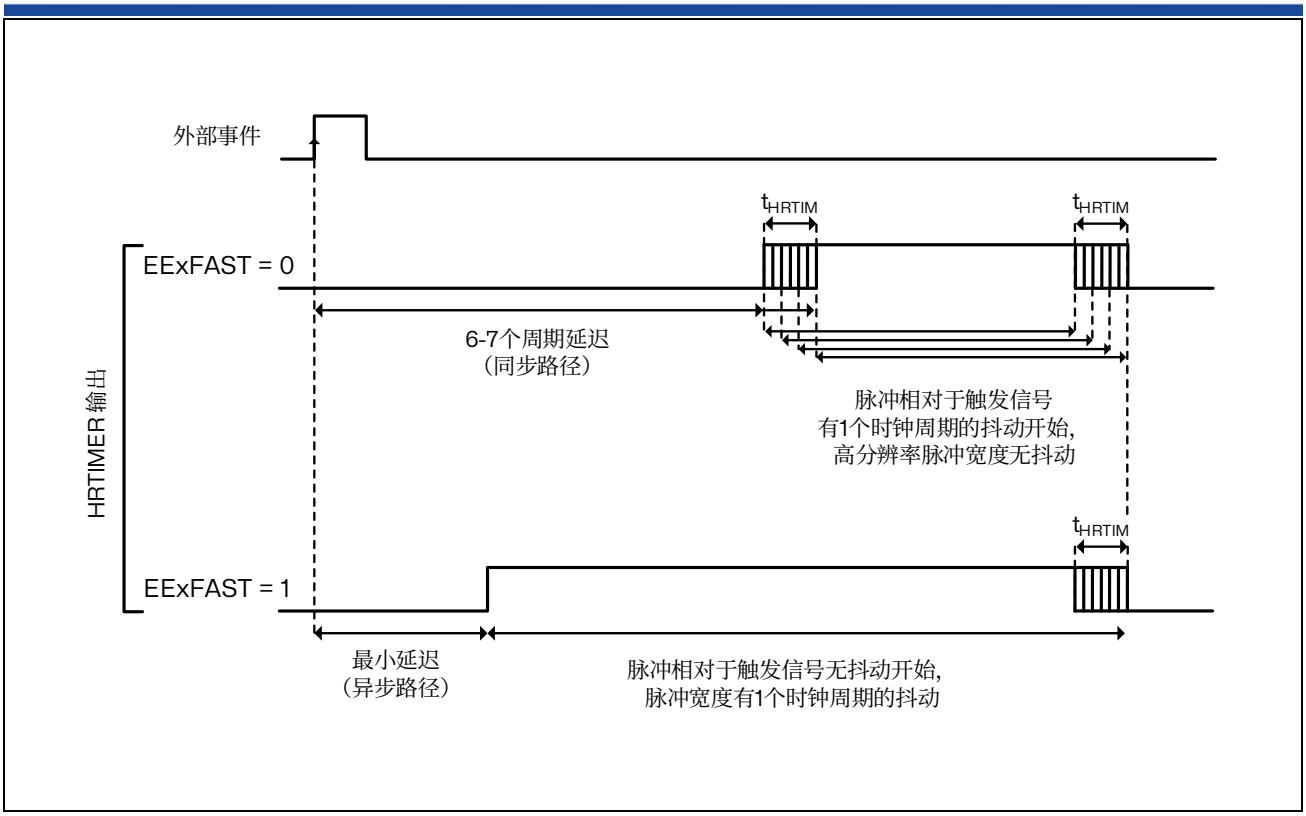



图 21.36 外部事件下降沿的延迟（计数器复位和输出置位）

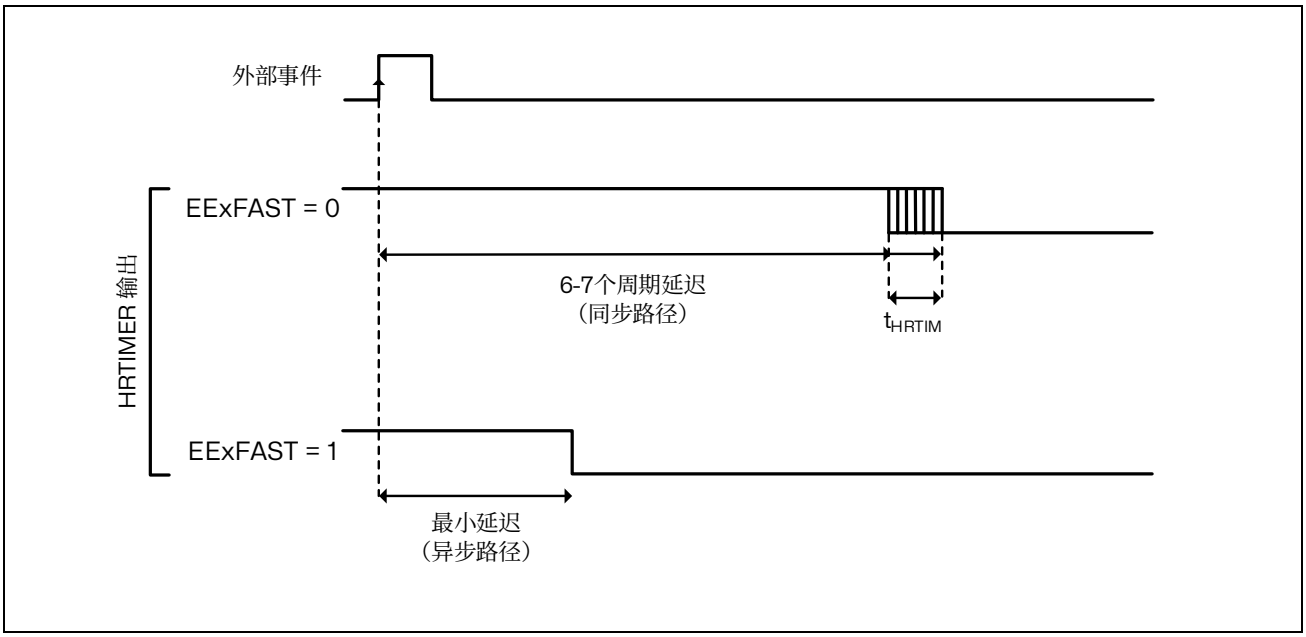



图 21.37 外部事件的延迟（发生外部事件时，输出复位）

### 21.3.9 定时单元中的外部事件过滤

经过调节后，有 10 个外部事件可供所有定时单元使用。
这些事件可直接使用，且在定时单元计数器使能后（TxCEN 位置 1）立即生效。
此外，还可对这些事件进行过滤，以便在限制时间段内执行操作，该时间段通常与计数周期有关。可执行两种操作：

- 消隐模式，在定义的时间段内屏蔽外部事件
- 窗口模式，仅在定义的时间段内使能外部事件

这些模式通过 HRTIM_EEFxR1 和 HRTIM_EEFxR2 寄存器中的 HRTIM_EExFLTR[3:0]位来使能。TimerA..F 这 6 个定时单元都具有针对这 10 个外部事件的可编程过滤器设置。

#### 消隐模式

在事件消隐模式下（如下图所示），如果外部事件发生在给定的消隐周期内，则会被忽略。举例来说，为了避免 PWM 周期开始时电流限制因开关噪声而失效，可使用此模式。当 EExFLTR[3:0]位域的值在 0001 到 1100 范围内时，该模式有效。

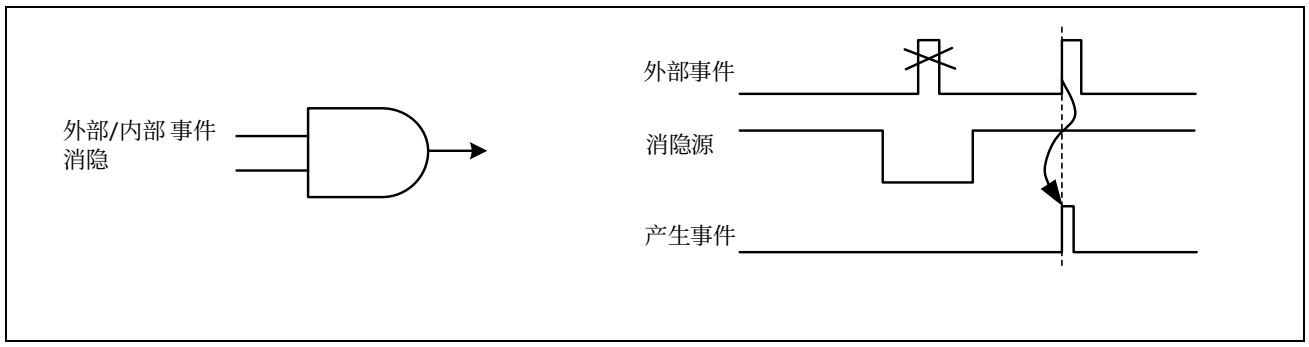



图 21.38 事件消隐模式

在事件延迟模式下，不会立即考虑外部事件，而是会将其保存（锁存）下来，并在消隐周期结束后立即生成外部事件，如下图所示。可将 HRTIM_EEFxR1 和 HRTIM_EEFxR2 寄存器中的 EExLTCH 位置 1 来使能此模式。

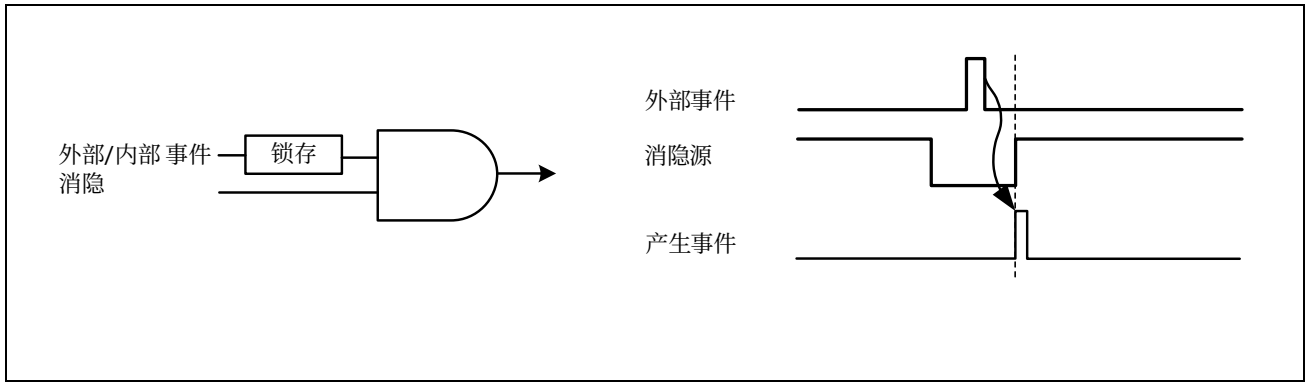



图 21.39 事件延迟模式

消隐信号来自多个源：

- 定时器本身：消隐持续时间从计数器复位开始，到比较匹配为止（对于比较1到比较4，EExFLTR[3:0] = 0001 到 0100）。在上下计数模式（UDM 位设置为 1）下，计数器复位事件根据 ROM[1:0]位的设置来确定。
- 来自其他定时单元（EExFLTR[3:0] = 0101 到 1100）：消隐持续时间从选定的定时单元计数器复位开始，到其中一个比较匹配为止，或者可完全编程为 Tx2 输出上的波形。在这种情况下，只要 Tx2 信号无效，事件就会被屏蔽（不需要使能输出，而且会在输出级之前获取信号）。

EEXFLTR[3:0]配置 0101 到 1100 在位说明中被称为 TIMFLTR1 到 TIMFLTR8，各定时单元的配置在位说明中的含义有所不同。下表给出了每个定时器的 8 个可用选项：CMPx 是指计数器复位到比较匹配期间进行消隐，Tx2 是指通过 HRTIM_SETx2 和 HRTIM_RSTx2 寄存器定义的定时单元 TIMx 输出 2 波形。举例来说，定时器 B（TIMFLTR3）是定时器 A 输出 2 波形。

表 21.20 每个定时器的过滤信号映射

| 源 | | 定时器 A | | | | 定时器 B | | | | 定时器 C | | | | 定时器 D | | | | 定时器 E | | | | 定时器 F | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CMP1 | CMP2 | CMP4 | TA2 | CMP1 | CMP2 | CMP4 | TB2 | CMP1 | CMP2 | CMP4 | TC2 | CMP1 | CMP2 | CMP4 | TD2 | CMP1 | CMP2 | CMP4 | TE2 | CMP1 | CMP2 | CMP4 | TF2 |
| 目标 | TA | - | | | | 1 | - | 2 | 3 | 4 | - | 5 | - | 7 | - | - | - | - | 8 | - | - | 6 | - | - | - |
| | TB | 1 | - | 2 | 3 | - | | | | 4 | 5 | - | - | - | 7 | - | - | 8 | - | - | - | - | 6 | - | - |
| | TC | - | 1 | - | - | 2 | - | 3 | - | - | | | | 5 | - | 6 | 7 | - | - | 8 | - | 4 | - | - | - |
| | TD | 1 | - | - | - | - | 2 | - | - | 3 | 4 | - | 5 | - | | | | 6 | - | 7 | - | - | - | 8 | - |
| | TE | - | 1 | - | - | 2 | - | - | - | 3 | - | - | - | 6 | - | 7 | 8 | - | | | | - | - | 4 | 5 |
| | TF | - | - | 1 | - | - | 2 | - | - | - | - | 3 | - | - | 4 | 5 | - | 6 | - | 7 | 8 | - | | | |

下两图举例说明了常规模式和延迟模式下所有边沿有效和电平有效的外部事件消隐。

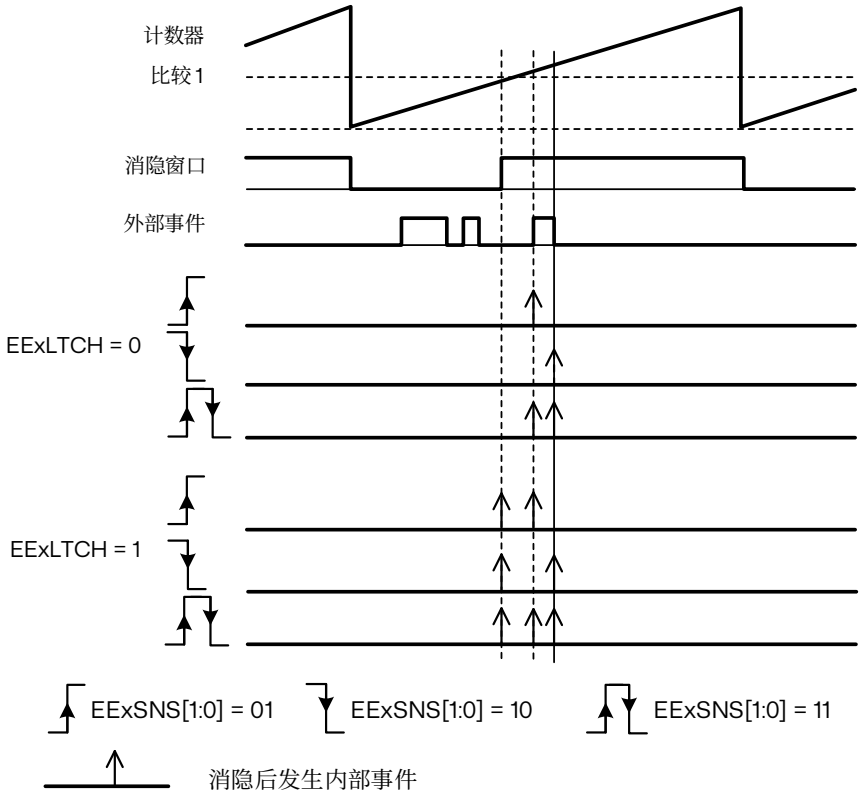



图 21.40 采用边沿有效触发的外部触发消隐

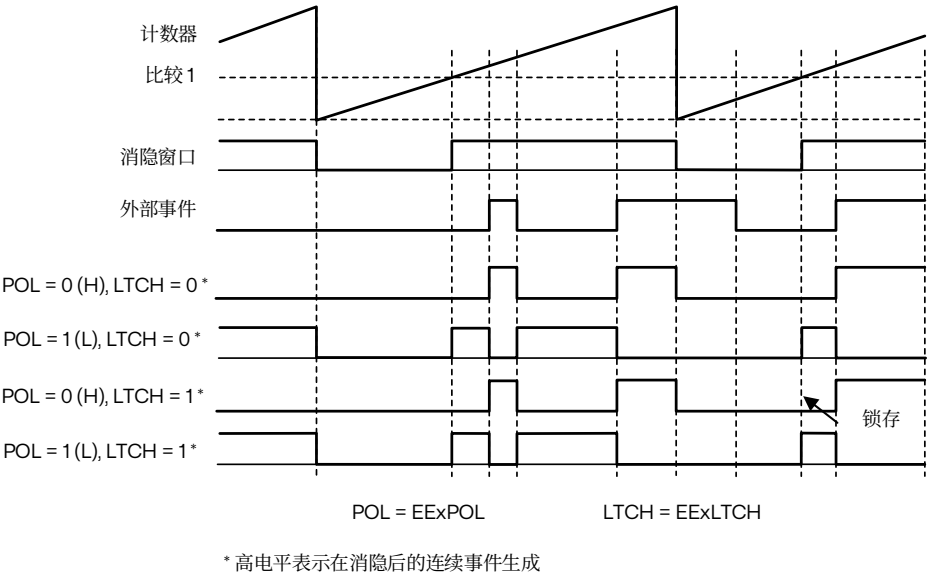



图 21.41 外部触发消隐，电平有效触发

### 窗口模式

在事件窗口模式下，仅当事件在给定事件窗口内发生时才会被考虑，否则会被忽略。当 EExFLTR[3:0]的值在 1101 到 1111 范围内时，该模式有效。

图 21.42 事件窗口模式

EEFxR1 和 EEFxR2 寄存器中的 EExLTCH 可锁存信号，如果该位置 1：在这种情况下，如果事件在窗口期间发生，但在窗口结束时延迟，也会被接受。

- 如果 EExLTCH 位复位，且信号在窗口期间发生，则信号会直接通过。
- 如果 EExLTCH 位复位，并且无信号发生，则会在窗口结束时生成超时事件。

窗口模式可用于过滤同步信号。当缺少预期的同步事件时（例如在转换器启动期间），可通过超时生成功能强制生成默认同步事件。

每个外部事件窗口有 3 个源，其编码如下：

- 1101 和 1110：窗口持续时间从计数器复位开始，到比较匹配为止（分别是比较 2 和比较 3）。在上下计数模式（UDM 位设置为 1）下，计数器复位事件根据 ROM[1:0]位的设置来确定。
- 1111：窗口与另一定时单元相关，持续时间从计数器复位开始，到其比较 2 匹配为止。源被称为 TIMWIN 的位中说明，请参见下表。举例来说，定时器 B 中的外部事件可通过从定时器 A 计数器复位开始、到定时器 A 比较 2 为止的窗口进行过滤。

表 21.21 每个定时器的窗口信号映射（EEFLTR[3:0] = 1111）

| 目标 | 定时器 A | 定时器 B | 定时器 C | 定时器 D | 定时器 E | 定时器 F |
|---|---|---|---|---|---|---|
| TIMWIN（源） | 定时器 B CMP2 | 定时器 A CMP2 | 定时器 D CMP2 | 定时器 C CMP2 | 定时器 F CMP2 | 定时器 E CMP2 |

*注：如果外部事件是在快速模式下编程的，则不支持生成超时事件*

下两图介绍了如何根据 EExLTCH 位的设置生成事件，以实现各种边沿有效和电平有效触发。为了便于理解，专门针对超时事件进行了说明。

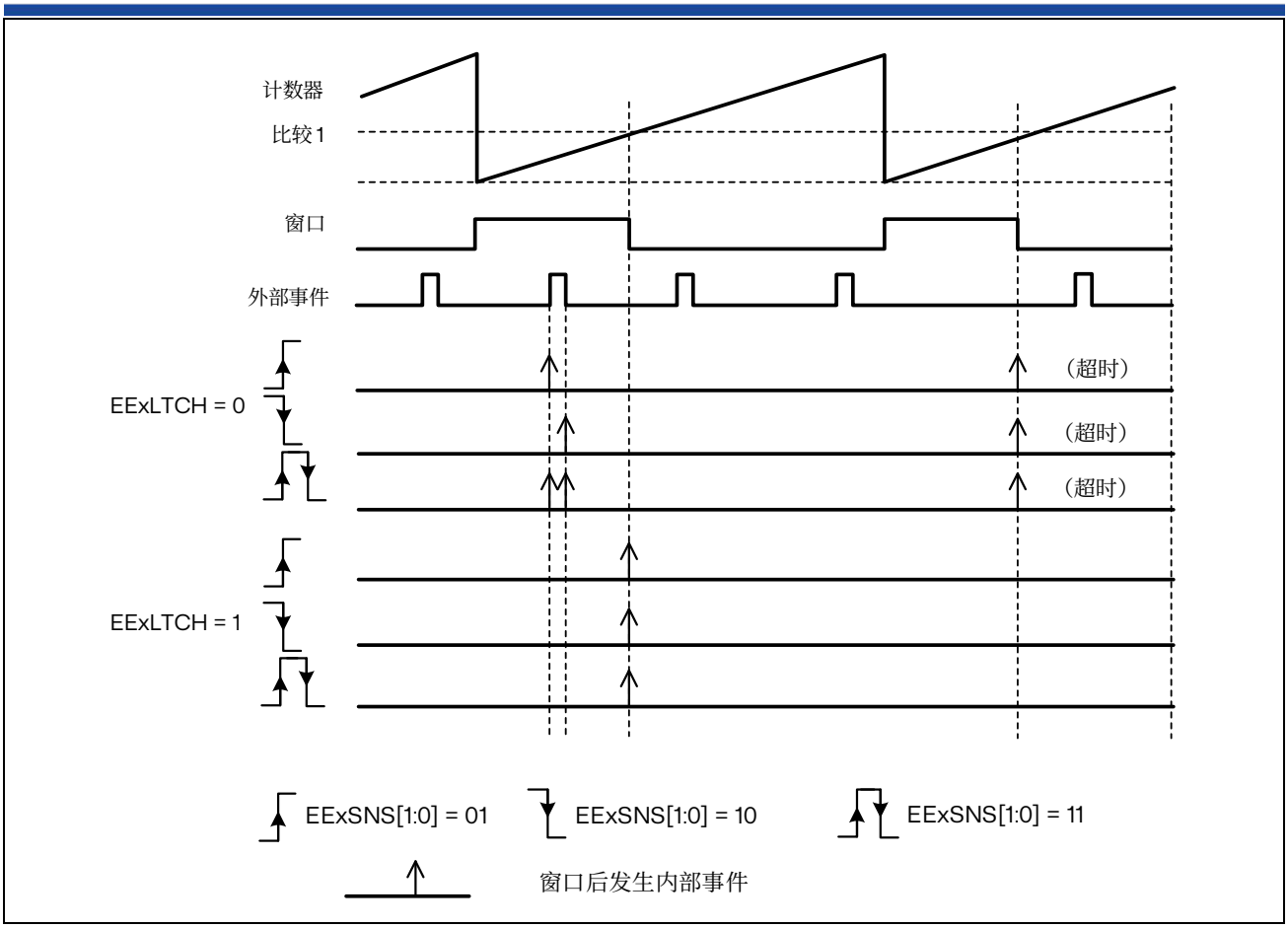



图 21.43 采用边沿有效触发的外部触发窗口

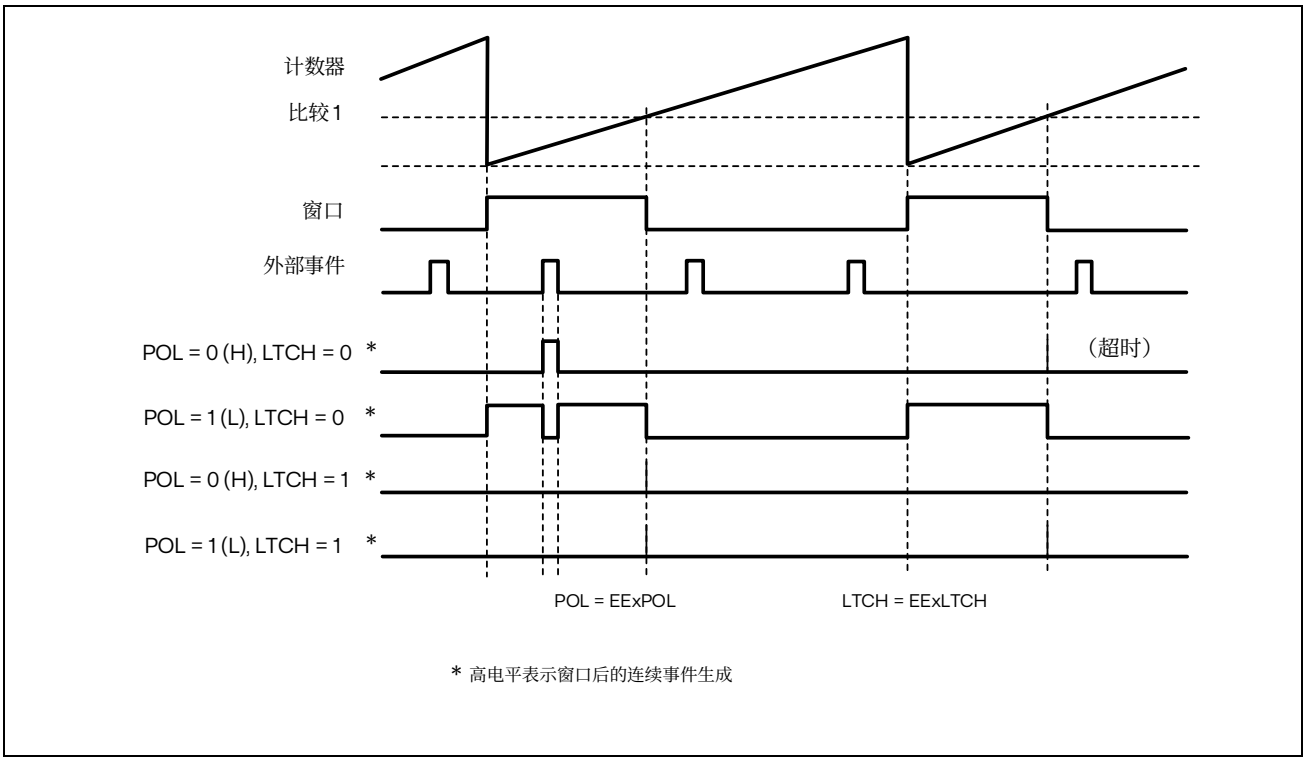



图 21.44 外部触发窗口，电平有效触发

### 外部事件计数器

每个定时单元还在滤波单元后面配备了一个外部事件计数器，通常用于实现谷底跳过（Valley Skipping）功能。

该电路允许对经过滤波的 10 个外部事件中的任何一个进行进一步滤波，如下图所示。

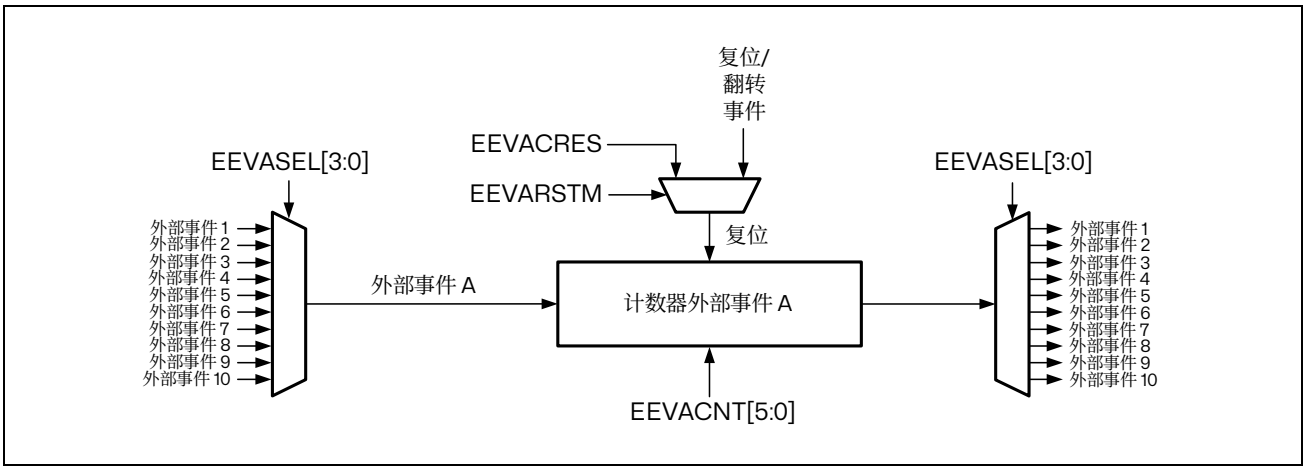



图 21.45 外部触事件计数器 – 通道 A

计数器通过 HRTIM_EEFxR3 寄存器中的 EEVACE 位使能。此模式仅对边沿敏感的外部事件有效（EEASNS[1:0]位=01、10 或 11）。

只有在有效边沿的数量大于或等于（EEVACNT[5:0]+1）中编程的值时，外部事件才会传播到定时器。

有两种操作模式可供选择：

- 当 EEVARSTM 位被重置时，外部事件计数器在每个复位/溢出事件上重置：外部事件仅在给定 PWM 周期内出现多次时才有效。
- 当 EEVARSTM 位被设置时，外部事件计数器仅在事件在最后一个 PWM 周期内没有出现时重置。这是一种累积模式，事件必须在多个 PWM 周期内至少出现一次，如下图所示。

在编程计数器值之后，必须使能外部事件计数器（在设置 EEVACNT[5:0]位之后，必须设置 EEVACE 位）。

一旦计数器被使能，EEVACNT[5:0]位就可以在任何时候实时更改。新值将根据 EEVARSTM 位的编程在下一个复位/溢出事件时生效，或者在软件复位（EEVACRES 位设置）后生效。

一旦设置了 EEVACE 位，就不应再修改 EEVASEL[3:0]位。

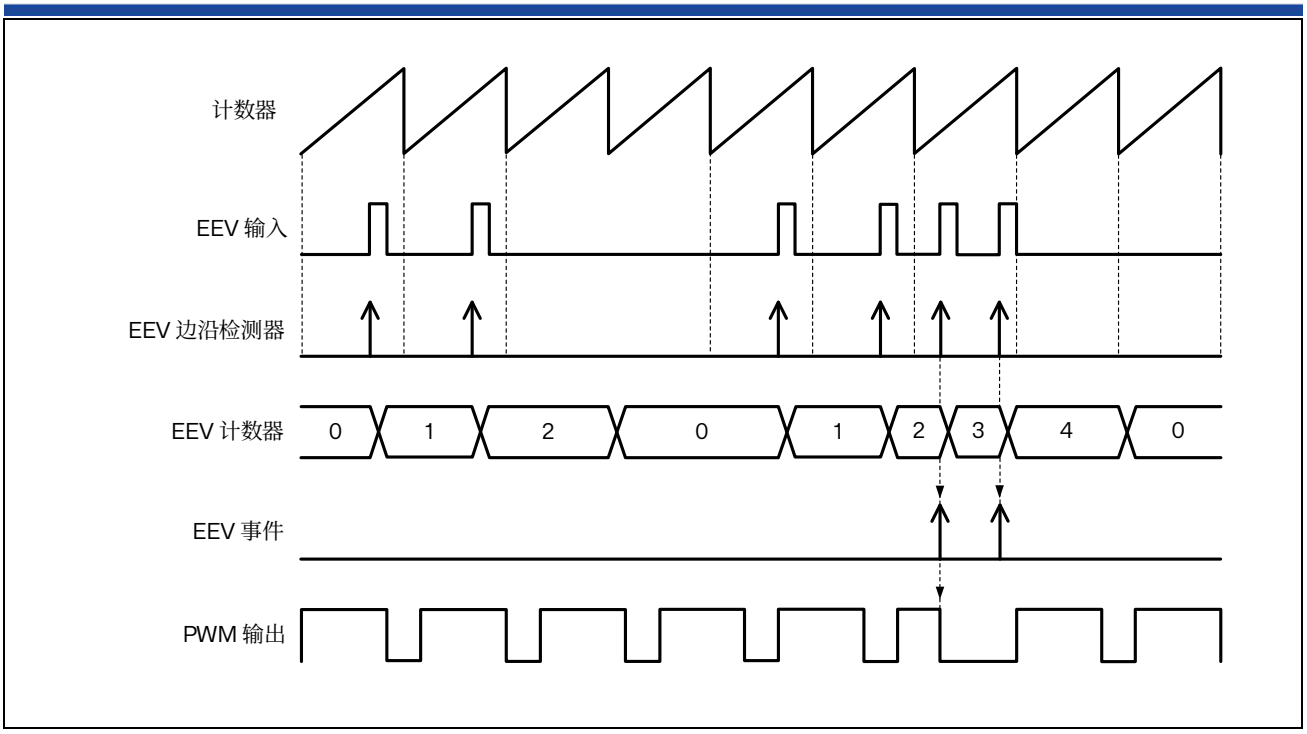



图 21.46 外部事件计数器累积模式（EEVxRSTM = 1，EEVxCNT = 2）

### 21.3.10 延迟保护

如果需要在有效脉冲结束后或推挽周期结束后以延迟的方式关断 PWM 输出，HRTIM 通常会为谐振转换器提供特定的保护机制。这些功能通过 HRTIM_OUTxR 寄存器中的 DLYPRTEN 位使能，并将使用特定的外部事件通道。

#### 延迟空闲模式

在该模式下，有效电平会在保护激活之前结束。所选外部事件会使输出在有效脉冲结束时进入空闲模式（由 HRTIM_RSTx1R 或 HRTIM_RSTx2R 中的输出复位事件定义）

一旦保护被触发，会永久保持空闲模式，但计数器会继续工作，直至输出重新使能。Tx1OEN 和 Tx2OEN 位不受延迟空闲模式进入的影响。要退出延迟空闲模式并恢复操作，需要将 Tx1OEN 和 Tx2OEN 位重写为 1。输出状态将在发出输出使能命令后第一次跳变为有效状态时更改。

*注：在有效脉冲结束之前，进入了延迟空闲模式便不能立即退出：务必先确保输出处于空闲状态，然后再重新开始运行模式。具体做法可以是等到下一周期，或者轮询* TIMxISR *寄存器中 的* O1CPY *和/ 或* O2CPY *状态位。*

延迟空闲模式可应用于单路输出（DLYPRT[2:0] = x00 或 x01），也可应用于两路输出(DLYPRT[2:0] = x10)。

进入响应延迟空闲模式后，可通过生成中断或 DMA 请求以作出响应。外部事件到达后，会立即将 HRTIM_TIMxISR 中的 DLYPRT 标志置 1，和输出上的有效脉冲是否结束无关。

延迟空闲模式触发后，可通过 HRTIM_TIMxISR 中的 O1STAT 和 O2STAT 确定输出状态。即使延迟空闲模式应用于单路输出，也会更新这两个状态位。如果使能了推挽模式，HRTIM_TIMxISR 中的 IPPSTAT 标志会指示延迟保护请求在哪一周期发生。

无论定时器采用何种工作模式（常规、推挽、死区），均可使用此模式。此模式仅支持 2 个外部事件：

- hrtim_evt6 和 hrtim_evt7（对于定时器 A、B 和 C）
- hrtim_evt8 和 hrtim_evt9（对于定时器 D、E 和 F）

仅当计数器使能后（TxCEN 位置 1），才能触发延迟保护模式。即使 TxEN 位已复位，在 TxyOEN 位置 1 之前，延迟保护模式仍保持有效状态。

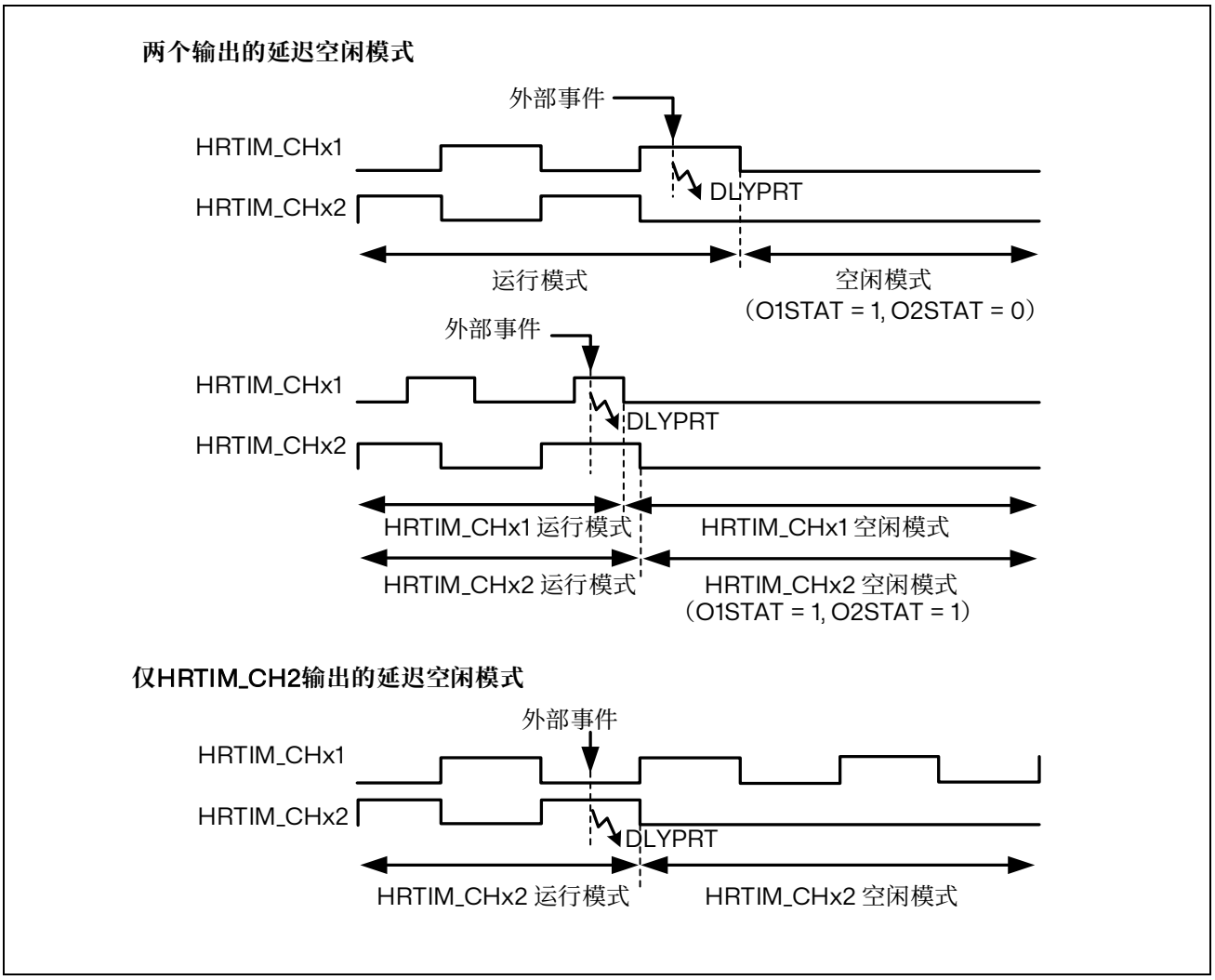



图 21.47 延迟空闲模式进入

延迟空闲模式的优先级要高于突发模式：一旦触发了延迟空闲保护，就会丢弃任何突发模式退出请求。相反，如果在突发模式有效时退出延迟保护，突发模式将正常恢复，输出将保持为空闲状态，直至退出突发模式。下图给出了这些不同情况的概览。

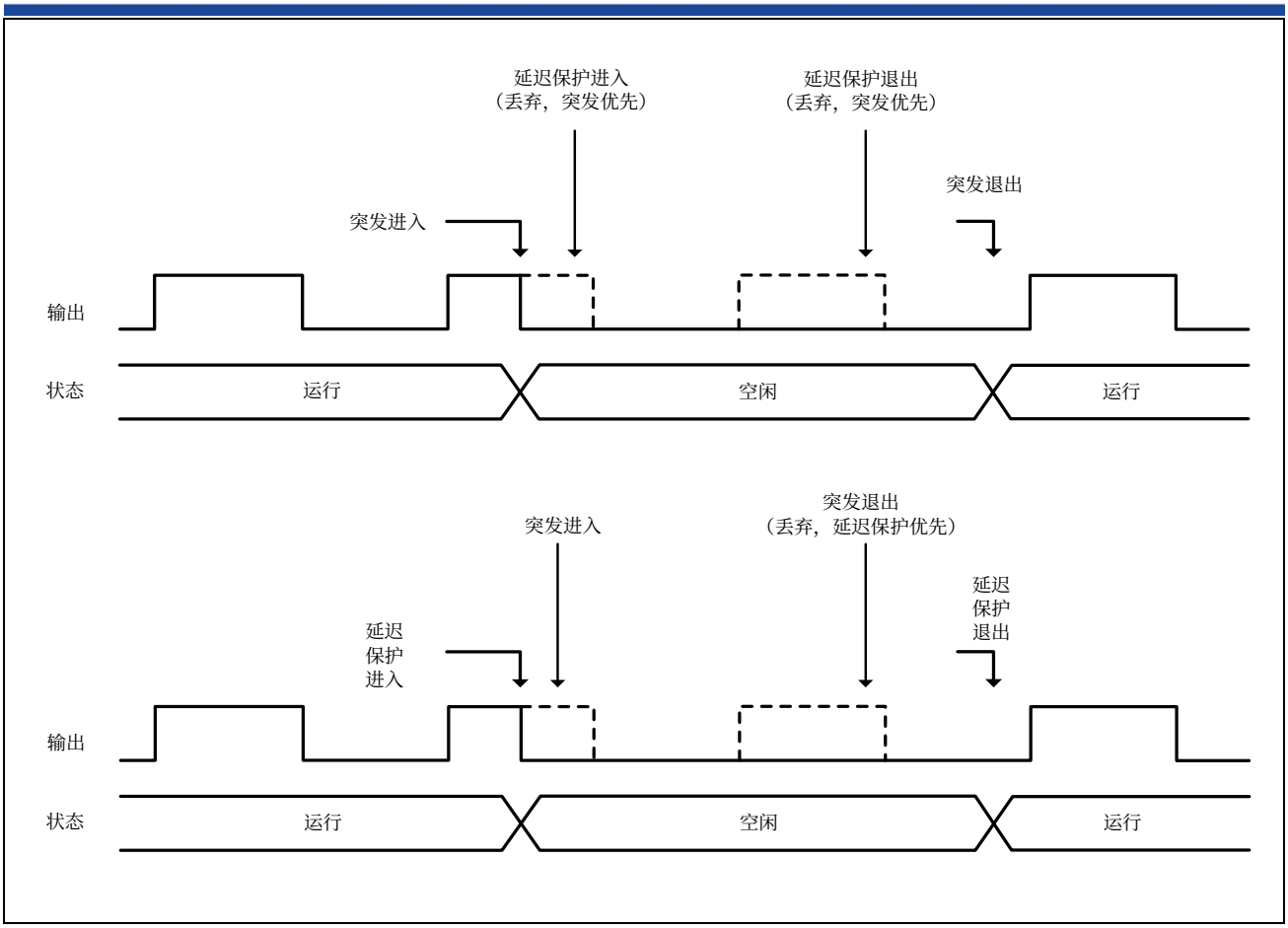



图 21.48 突发模式和延迟保护优先级（DIDL = 0）

如果使能了延迟突发模式进入（DIDL 位置 1），则会应用相同的优先级，如下图所示。

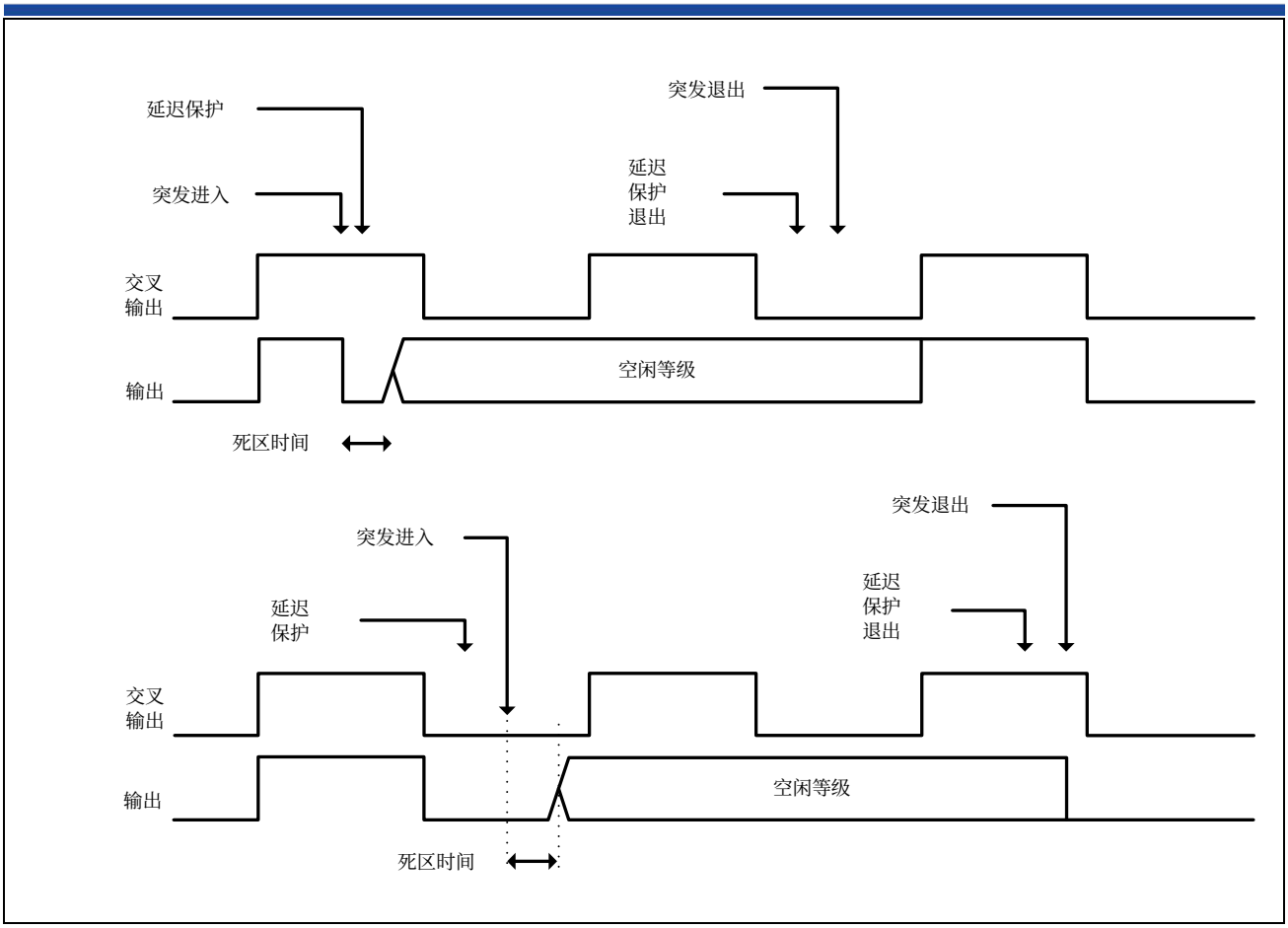



图 21.49 突发模式和延迟保护优先级（DIDL = 1）

### 均衡空闲

仅在推挽模式下可用，在有效脉冲因保护方面的原因而缩短的情况下，可在两路输出上实现均衡脉宽。终止时间早于设定时间的脉宽会复制到另一路输出上，两路输出随后会进入空闲状态，直至通过软件恢复正常工作。此模式通过向 HRTIM_OUTxR 中的 DLYPRT[2:0]位域写入 x11 来使能。

此模式仅支持 2 个外部事件：

- hrtim_evt6 和 hrtim_evt7（对于定时器 A、B 和 C）
- hrtim_evt8 和 hrtim_evt9（对于定时器 D、E 和 F）

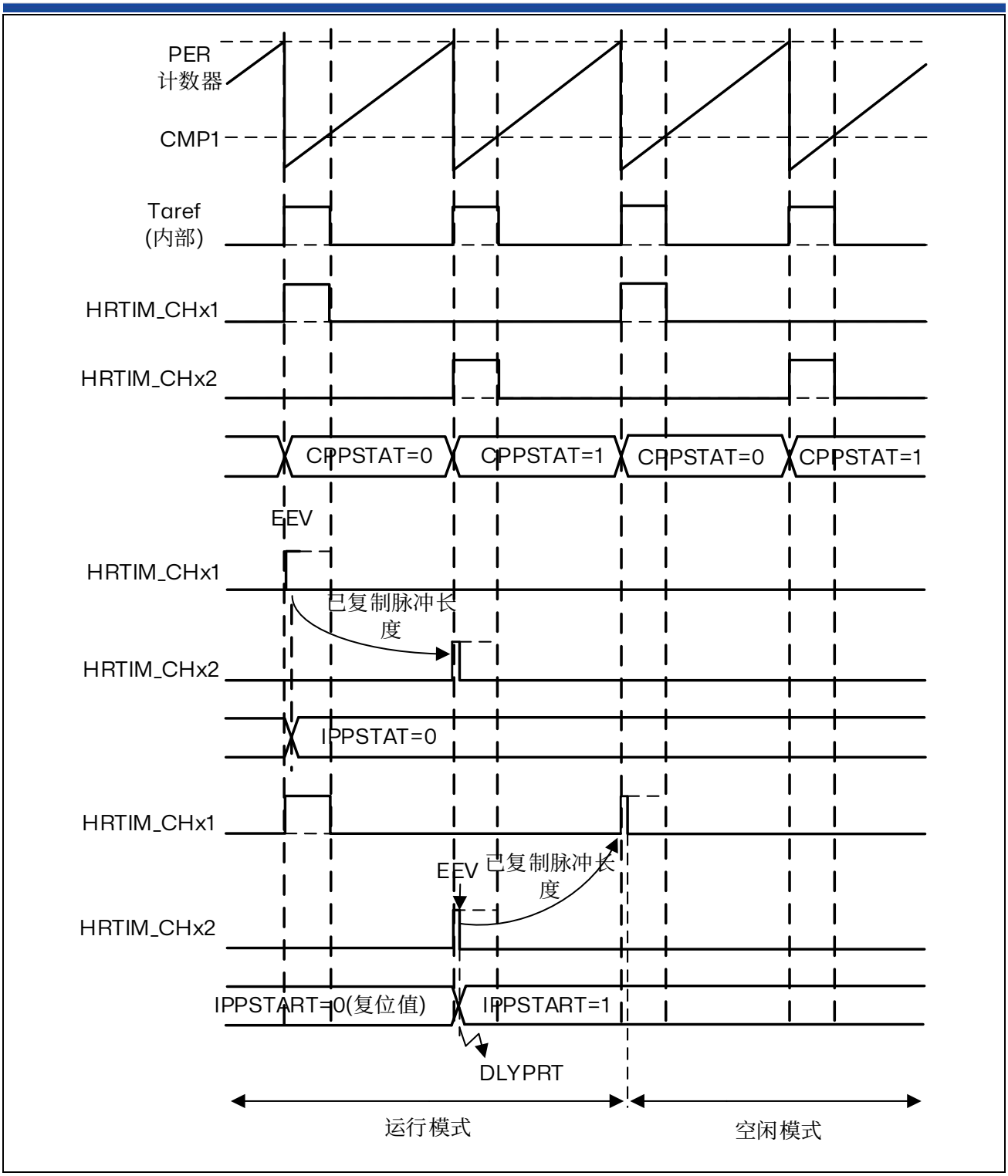



图 21.50 均衡空闲保护示例

如果均衡空闲模式已使能，所选外部事件会触发将计数器值捕获到比较 4 活动寄存器（用户不可访问该值）。推挽模式会额外保持一个周期，以重复缩短的脉冲：保持常规输出置位事件的同时，会生成新的输出复位事件。

随后会进入空闲模式，输出会采用由 HRTIM_OUTxR 寄存器中的 IDLESx 位定义的电平。均衡空闲模式进入由 DLYPRT 标志指示，IPPSTAT 标志则指示外部事件在哪一周期发生，可用于确定缩短脉冲的顺序（先是 A1，然后是 A2，反之亦然）。

定时器操作不会中断（计数器继续运行）。

要使能均衡空闲模式，需要执行下列初始化操作：

- 定时器在连续模式下工作（CONT = 1）
- 使能推挽模式
- HRTIM_CMP4xR 必须设为 0，并且内容要传输到活动寄存器（例如，强制进行软件更新）
- DELCMP4[1:0]位域必须设为 00（禁止自动延迟模式）
- DLYPRT[2:0] = x11（使能延迟保护）

*注：均衡空闲工作期间，不得对* HRTIM_CMP4xR *寄存器执行写操作。保留* CMP4 *事件，不得用于其他用途。*

*在均衡空闲模式下，建议避免使用多个外部事件或基于软件的复位事件，后者会导致输出复位。如果同一周期内此类事件先于均衡空闲请求到达，则会导致输出脉冲不均衡（第*1 *个脉冲长度由外部事件或软件复位定义，而第*2 *个脉冲长度则由均衡空闲模式进入定义）。*

均衡空闲模式下可用的最小脉宽为 4 个 $f_{HRTIM}$ 时钟周期（当 CKPSC[2:0] = 0 时为 0x80，CKPSC[2:0] = 1 时为 0x40，CKPSC[2:0] = 2 时为 0x20，依此类推）。如果在计数器达到其最小值之前进行捕获，会先将当前脉冲最多延长为 4 个 $f_{HRTIM}$ 时钟周期，然后再将其复制到次级输出。在任何情况下，脉宽始终都是均衡的。

Tx1OEN 和 Tx2OEN 位不受均衡空闲模式进入的影响。要退出均衡空闲模式并恢复操作，需要同时将 Tx1OEN 和 Tx2OEN 位重写为 1。输出状态将在输出使能后的第一次有效跳变时更改。

可采用与延迟空闲模式进入相似的方法恢复操作。例如，如果外部事件到达时输出 1 激活（延迟空闲模式在输出 2 脉冲后生效），可先为输出 1 启动重启序列。为此，需要轮询 HRTIM_TIMxISR 寄存器中的 CPPSTAT 位。使用上例（IPPSTAT 标志等于 0），操作将在 CPPSTAT 位为 0 零时恢复。

为了获得特定的重启序列，可轮序 CPPSTAT 了解哪一输出将先激活。这样一来，便可使用与空闲模式进入序列相同的序列进行重启：如果 EEV 在输出 1 激活期间到达，则将在输出 1 激活时（CPPSTAT = 0）发起重启序列。

*注：正在执行脉冲均衡序列时，不得禁止均衡空闲模式。需要等到* CMP4 *标志置*1*（指示序列已结束）才能复位* DLYPRTEN *位。*

仅当计数器使能后（TxCEN 位置 1），才能触发均衡空闲保护模式。即使 TxCEN 位已复位，在 TxyOEN 位置 1 之前，延迟保护模式仍保持有效状态。

下列情况下，均衡空闲模式可与突发模式一起使用：

- TxBM 位必须被重置（突发期间保持计数器时钟）
- 输出处于突发空闲状态时，不得触发均衡空闲保护。

均衡空闲模式的优先级要高于突发模式：一旦触发了均衡空闲保护，就会丢弃任何突发模式退出请求。相反，如果在突发模式有效时退出延迟保护，突发模式将正常恢复。

*注：虽然输出状态在空闲模式下会冻结，但在延迟保护后的空闲时间段内，仍会在辅助输出上生成一些事件*

- *输出置位/复位中断或*DMA *请求*
- *基于输出信号的外部事件过滤*
- *由置位/复位触发的捕获事件*

#### 均衡空闲自动恢复

均衡空闲模式可以配置为在触发后自动恢复操作。

一旦缩短的脉冲被复制到备用输出，脉冲宽度将重置为其原始值，定时器恢复操作：两个输出将继续处于运行模式。

通过在 HRTIM_OUTxR 寄存器中设置 BIAR 位来使能此功能。

此模式仅当 HRTIM_PERxR 中的周期大于 $f_{HRTIM}$ 时钟的 6 个周期时才应使用，即如果 CKPSC[2:0] = 0，则为 0xC0；如果 CKPSC[2:0] = 1，则为 0x60；如果 CKPSC[2:0] = 2，则为 0x30，以此类推。

*注：此位仅在* DLYPRT[2:0] = 011 *或*111 *时才有意义，否则将被忽略。*

*注：在均衡空闲自动恢复模式下，必须将*IDLES *状态设置为无效状态。*

### 21.3.11 寄存器预加载和更新管理

大部分 HRTIM 寄存器均已缓冲，可根据需要进行预装载。这样做通常可避免波形被与有效事件（置位/复位）不同步的寄存器更新更改。

使能预装载模式后，受访问的寄存器变为影子寄存器。收到软件发出的更新请求或与事件同步的更新请求后，影子寄存器的内容会传输到活动寄存器。

默认情况下，HRTIM_MCR 和 HRTIM_TIMxCR 寄存器中的 PREEN 位会复位，寄存器不会预装载：任何写操作均会直接更新活动寄存器。如果 PREEN 位在定时器运行且预装载已使能时复位，预装载寄存器的内容会直接传输到活动寄存器中。

每个定时单元和主定时器都有自己的 PREEN 位。如果 PRREN 已置 1，仅当发生更新事件时，预装载寄存器才会使能，其内容才会传输到活动寄存器。

如果需要使用预装载功能，可选择两种方法初始化定时器：

- 刚好在定时器初始化结束时使能 PREEN 位，以在定时器使能之前将预装载寄存器内容传输到活动寄存器（通过将 MCEN 和 TxCEN 位置 1）。
- 在初始化过程中的任何时间使能 PREEN 位，并恰好在启动之前强制进行软件更新。

下表列出了可预装载的寄存器，并总结了可用更新事件。

表 21.22 HRTIM 可预装载控制寄存器和关联的更新源

| 定时器 | 可预装载寄存器 | 预装载使能 | 更新源 |
|---|---|---|---|
| 主定时器 | HRTIM_DIER<br>HRTIM_MPER<br>HRTIM_MREP<br>HRTIM_MCMP1R<br>HRTIM_MCMP2R<br>HRTIM_MCMP3R<br>HRTIM_MCMP4R | HRTIM_MCR 中的 PREEN 位 | 软件<br>重复事件<br>突发 DMA 事件<br>突发 DMA 事件后的重复事件 |
| 定时器 x<br>x = A..F | HRTIM_TIMxDIER<br>HRTIM_TIMxPER<br>HRTIM_TIMxREP<br>HRTIM_TIMxCMP1R<br>HRTIM_TIMxCMP1CR<br>HRTIM_TIMxCMP2R<br>HRTIM_TIMxCMP3R<br>HRTIM_TIMxCMP4R<br>HRTIM_DTxR<br>HRTIM_SETx1R<br>HRTIM_RSTx1R<br>HRTIM_SETx2R<br>HRTIM_RSTx2R<br>HRTIM_RSTxR | HRTIM_TIMxCR 中的 PREEN 位 | 软件<br>TIMx 重复事件<br>TIMx 复位事件<br>突发 DMA 事件<br>来自其他定时器（TIMy，主定时器）的更新事件<br>突发 DMA 事件后的更新事件<br>更新使能输入 hrtim_upd_en[3:1]<br>更新使能输入 hrtim_upd_en[3:1]后的更新事件 |
| HRTIM 通用寄存器 | HRTIM_ADC1R<br>HRTIM_ADC2R<br>HRTIM_ADC3R<br>HRTIM_ADC4R | TIMx 或主定时器更新，取决于 HRTIM_CR1 中的 ADxUSRC[2:0]位（如果所选定时器中的 PREEN = 1） | |

主定时器有 4 个更新选项：

1. 软件：将 1 写入 HRTIM_CR2 中的 MSWU 位会立即强制更新寄存器。在这种情况下，会取消任何待定的硬件更新请求。
2. 当主计数器翻转且主重复计数器等于 0 时，会进行更新。当 HRTIM_MCR 中的 MREPU 位置 1 时，会使能此操作。
3. 突发 DMA 结束后进行更新。当 HRTIM_MCR 中的 BRSTDMA[1:0] = 01 时，会使能此操作。可以同时设置 MREPU = 1 和 BRSTDMA = 01。
   *注：如果 SWU 位置 1，可在突发序列结束后立即进行更新（即强制更新模式）。如果 SWU 位复位，将在突发序列结束后发生下一个更新事件时进行更新。*
4. 突发 DMA 结束后主计数器翻转时，会进行更新。当 HRTIM_MCR 中的 BRSTDMA[1:0] = 10 时，会使能此操作。

中断或 DMA 请求可通过主定时器更新事件生成。

各个定时器（TIMA..F）还可以通过以下方式进行更新：

- 软件：将 1 写入 HRTIM_CR2 中的 TxSWU 位会立即强制更新寄存器。在这种情况下，会取消任何挂起的硬件更新请求。
- 当计数器翻转且重复计数器等于 0 时，会进行更新。当 HRTIM_TIMxCR 中的 TxREPU 位置 1 时，会使能此操作。
- 当计数器复位或在连续模式下翻转时，会进行更新。当 HRTIM_TIMxCR 中的 TxRSTU 位置 1 时，会使能此操作。此操作用于在单发模式下工作的定时器等。
- 突发 DMA 结束后立即进行更新。当 HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0001 时，会使能此操作。
- 在突发 DMA 结束后发生更新事件时会进行更新（更新事件可通过 TxREPU、MSTU 或 TxU 使能）。当 HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0010 时，会使能此操作。
- 更新使能输入 hrtim_upd_en[3:1]上接收到请求时，会进行更新。当 HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0011、0100、0101 时，会使能此操作。
- 在更新使能输入 hrtim_upd_en[3:1]上接收到请求后发生更新事件时会进行更新（更新事件可通过 TxREPU、MSTU 或 TxU 使能）。当 HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0110、0111、1000 时，会使能此操作。
- 与其他任何定时器或主定时器更新同步进行更新（例如，TIMA 可与 TIMB 同步更新）。此操作通过将 HRTIM_TIMxCR 寄存器中的 MSTU 和 TxU 位置 1 来使能，用于需要使用多个定时器的转换器。

更新使能输入 hrtim_upd_en[3:1]可使更新事件与来自通用定时器的片上事件同步。这些输入为上升沿有效。

表 22.3 列出了更新使能输入与片上源之间的连接关系。

这样可将低频更新请求与高频信号同步（例如，在不得不用 100Hz 的速率更新 100KHz 的翻转计数器时）。

*注：当 CKPSC[2:0] > 5 时，更新事件会同步到预分频器时钟。*

来自相邻定时器（当 MSTU、TAU、TBU、TCU、TDU、TEU、TFU 位被设置时）或来自软件更新（TxSWU 位）的更新可以立即被考虑，也可以与定时器的复位/翻转事件重新同步。这通过 HRTIM_TIMxCR 寄存器中的 RSYNCU 位来实现，如下图所示：

- RSYNCU = 0：来自相邻定时器的更新立即被考虑。
- RSYNCU = 1：来自相邻定时器的更新将在下一个复位/翻转事件上被考虑。

只有当 UPDGAT[3:0] = 0000 时，RSYNCU 位才有意义，否则它将被忽略。

Timx 更新事件可以生成中断或 DMA 请求。

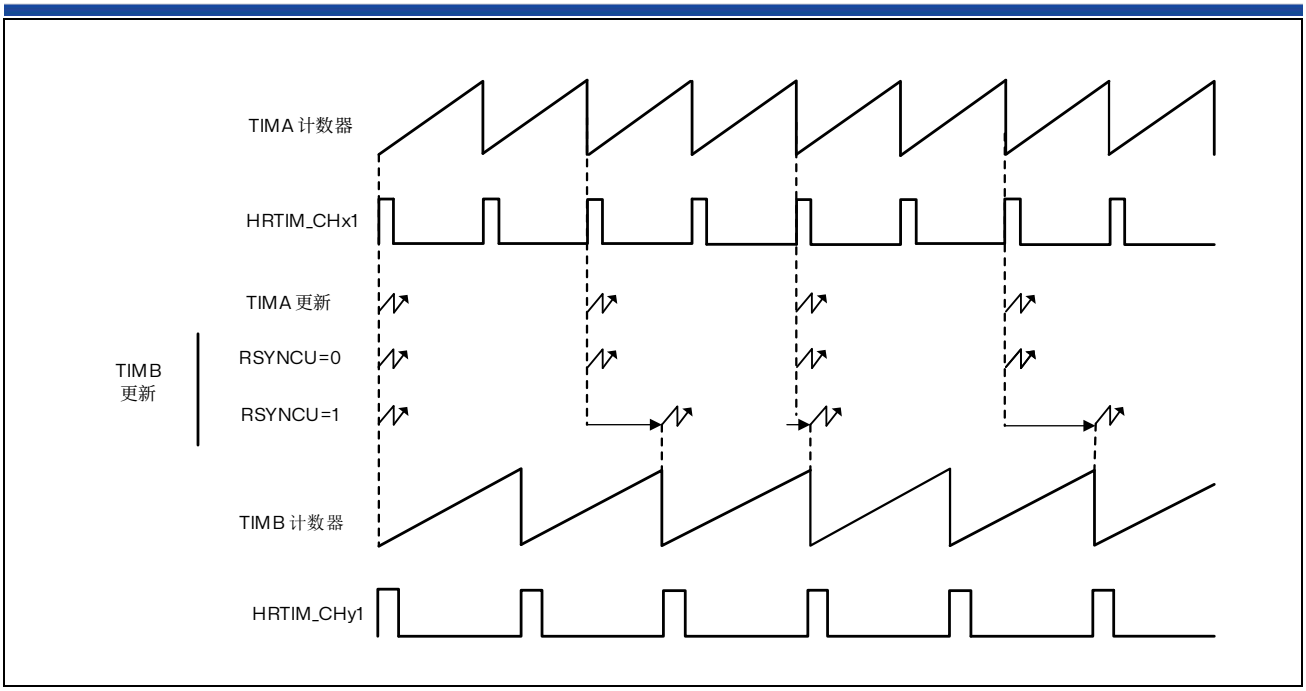



图 21.51 重同步定时器更新（在 HRTIM_TIMBCR 中 TAU=1）

无论选择哪一种更新事件，HRTIM_CR1 寄存器中的 MUDIS 和 TxUDIS 位都可以暂时禁止将预装载寄存器的内容传输至活动寄存器。这样便可修改并联的定时器中的多个寄存器。这些位复位后，会立即执行常规更新事件。

MUDIS 和 TxUDIS 位全部分组到同一寄存器中。这样便可同时禁止和恢复更新并联工作的定时器（不需要同步）。下例为实际用例。第一个电源转换器通过主定时器、TIMB 和 TIMC 控制。TIMB 和 TIMC 必须同时通过主定时器重复事件更新。第二个转换器与 TIMA、TIMD 和 TIME 并行运行，TIMD、TIME 必须通过 TIMA 重复事件更新。

第一个转换器：

在 HRTIM_MCR 中，MREPU 位已置 1：将在主定时器计数器重复周期结束时进行更新。在 HRTIM_TIMBCR 和 HRTIM_TIMCCR 中，MSTU 位必须置 1 才能通过主定时器同时更新 TIMB 和 TIMC 定时器。

如果必须通过软件调整电源转换器的设定值，则必须在对寄存器执行写访问更新数值（例如比较值）之前将 HRTIM_CR 寄存器的 MUDIS、TBUDIS 和 TCUDIS 位置 1。从此刻起，会忽略任何硬件更新请求，并可无风险地访问预装载寄存器，将其中的内容传输到活动寄存器中。软件处理结束后，必须将 MUDIS、TBUDIS 和 TCUDIS 位复位。发生主定时器重复事件后，会立即将预装载寄存器的内容传输到活动寄存器。

第二个转换器：

在 HRTIM_TIMACR 中，TAREPU 位已置 1：将在定时器 A 计数器重复周期结束时进行更新。在 HRTIM_TIMDCR 和 HRTIM_TIMECR 中，TAU 位必须置 1 才能通过定时器 A 同时更新 TIMD 和 TIME 定时器。

如果必须通过软件调整电源转换器的设定值，则必须在对寄存器执行写访问更新数值（例如比较值）之前将 HRTIM_CR 寄存器的 TAUDIS、TDUDIS 和 TEUDIS 位置 1。从此刻起，会忽略任何硬件更新请求，并可无风险地访问预装载寄存器，将其中的内容传输到活动寄存器中。软件处理结束后，可将 TAUDIS、TDUDIS 和 TEUDIS 位复位：发生定时器 A 重复事件后，会立即将预装载寄存器的内容传输到活动寄存器。

### 21.3.12 带有“大于”比较的 PWM 模式

CMP1 和 CMP3 寄存器生成的 PWM 信号具有一种特定的无延迟更新模式。它允许在 PWM 周期内尽快应用新的占空比值，而无需等待当前 PWM 周期完成。这减少了软件控制循环中的整体延迟时间。如以下图所示，这最终可以实现：

- 如果新的比较值低于当前计数器值，且当前比较值高于计数器值（在写入新值时），输出将提前关闭。
- 如果新的比较值高于计数器值，且当前比较值也高于计数器值（在写入新值时），输出将提前打开，重新使能输出。

当新的比较值和当前比较值都低于计数器时，输出信号保持不变。

此功能仅适用于 CMP1 或 CMP3 的复位事件，并使用 HRTIM_TIMxCR2 寄存器中的 GTCMP1 和 GTCMP3 使能位来使能。

当相应的 GTCMPx 位被设置时，预加载机制对于比较寄存器是不起作用的，无论 PREEN 位的值如何。此模式旨在在写入新值后尽快考虑新的比较值，而无需等待预加载到有效寄存器的传输。

这些位定义了比较 1 和比较 3 的操作模式，如下：

- GTCMPx = 0：当计数器等于比较值时，生成比较 x 事件（比较匹配模式）。如果比较值在运行时被更改，比较事件可能不会被生成。
- GTCMPx = 1：当计数器大于比较值时，生成比较 x 事件。如果比较值在运行时被更改，新的比较值将与当前计数器值进行比较，并可以生成输出置位或复位。

“大于”比较模式会根据比较结果的不同，导致纵横开关的行为有所区别。让我们考虑 CMP1 事件执行输出复位的情况。当写入新的比较值时，会考虑以下两种情况：

- 如果新的比较值低于计数器值，将发出复位事件，这可能会导致输出提前关闭。
- 如果新的比较值高于计数器值，将生成一个置位事件，以便在计数器实际超过比较值并触发复位之前，重新激活输出值（提前打开）。

“大于”比较模式同时支持置位和复位操作。

“大于”比较模式仅应用于以下配置：

1. 在固定频率配置中，周期事件必须触发输出置位，而“大于”比较则触发输出复位（或者，如果“大于”比较触发置位，那么周期事件必须触发复位）。
2. 对于可变频率配置，选作计数器复位源的事件也必须选作定时器输出的置位或复位源（与“大于”比较事件的方向相反）。

注：当 CMP1 和/或 CMP3 模式在半模式和交错模式下由硬件控制时，不应使用“大于”模式。

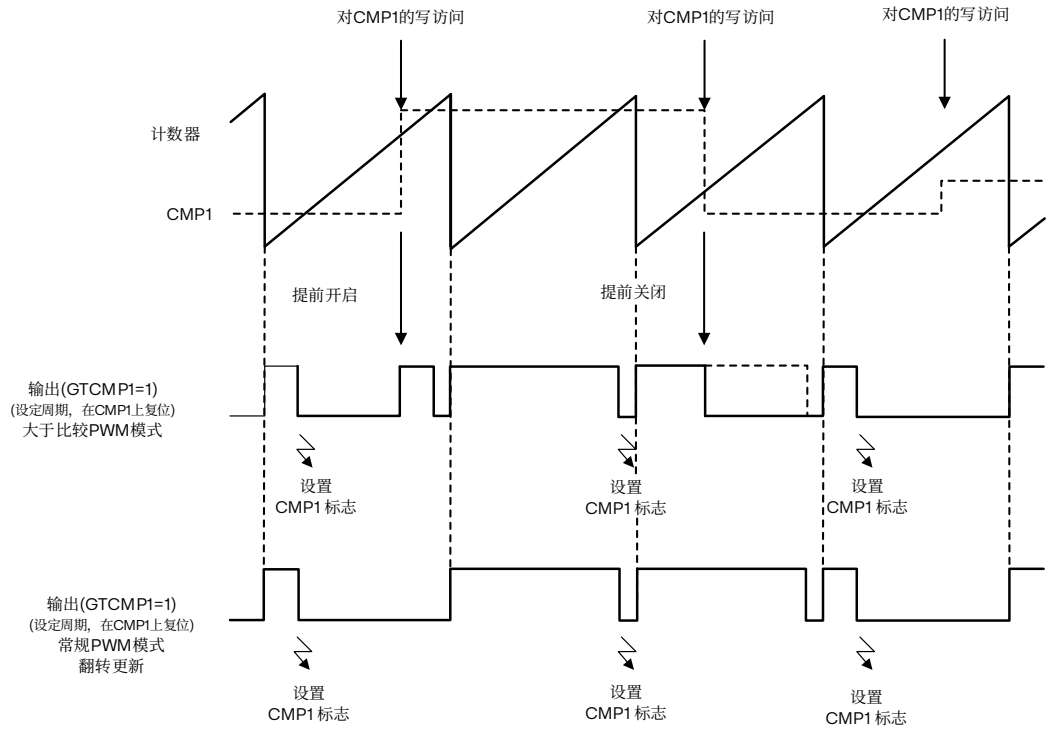



图 21.52 在“大于”PWM 模式下提前开启和提前关闭的行为

立即更新模式意味着在写入寄存器的同时，预加载寄存器的内容会被传输到有效寄存器中。当 GTCMP1 和/或 GTCMP3 位被设置时，它们各自的预加载机制将被禁用（对于 HRTIM_TIMxCMP1 和/或 HRTIM_TIMxCMP3 寄存器），无论 PREEN 位的值如何。

*注：在上图所示的情况下，对于延迟开启和提前关闭，不会生成比较中断标志（HRTIM_TIMxISR 中的 CMP1 和 CMP3）。*

*注：“大于”比较不能在同一个输出的 CMP1 和 CMP3 上都进行（GTCMP1 和 GTCMP3 位不能同时被设置）。*

### 21.3.13 事件在多个定时器之间的传播

HRTIM 提供多种方式在多个定时单元（包括主定时器）之间逐级传输事件或共享事件，以充分利用其模块化架构的优势。这些是需要使用多路同步输出的转换器的主要特性。

本节总结了各种选项，并详细说明了事件是否会在 HRTIM 中传播以及怎样传播。

**主计时器更新触发的 TIMx 更新**

下表中列出的源将生成主定时器更新。该表指出了是否可使用源事件在任何 TIMx 定时单元中触发同步更新。

工作条件：HRTIM_TIMxCR 寄存器中的 MSTU 位置 1。

表 21.23 主定时器更新事件传播

| 源 | 条件 | 传播 | 注释 |
|---|---|---|---|
| 突发 DMA 结束 | BRSTDMA[1:0] = 01 | 无 | 必须在 TIMxCR 中完成（UPDGAT[3:0] = 0001） |
| 突发 DMA 结束后的翻转事件 | BRSTDMA[1:0] = 10 | 有 | - |
| 由计数器翻转引起的重复事件 | MREPU = 1 | 有 | - |
| 由计数器复位（通过 HRTIM_SCIN 或软件）引起的重复事件 | | 无 | - |
| 软件更新 | MSWU = 1 | 无 | 所有软件更新位（TxSWU）被分组在 HRTIM_CR2 寄存器中，可用于同时更新 |

## 由 TIMy 更新触发的 TIMx 更新

下表中列出的源将生成 TIMy 更新。该表指出了是否可使用给定事件在另一定时器或多个 TIMx 定时器中触发同步更新。

工作条件：HRTIM_TIMxCR 寄存器中的 TyU 位置 1（源 = TIMy，目标 = TIMx）。

表 21.24 TIMx 更新事件传播

| 源 | 条件 | 传播 | 注释 |
|---|---|---|---|
| 突发 DMA 结束 | UPDGAT[3:0] = 0001 | 无 | 必须直接在 HRTIM_TIMxCR 中进行（UPDGAT[3:0] = 0001） |
| 由更新使能输入 hrtim_upd_en[3:1] 引发的更新 | UPDGAT[3:0] = 0011, 0100, 0101 | 无 | 必须直接在 HRTIM_TIMxCR 中进行（UPDGAT[3:0] = 0011、0100、0101） |
| 主定时器更新 | HRTIM_TIMyCR 中的 MSTU = 1 | 无 | 必须在 HRTIM_TIMxCR 中的 MSTU = 1 时进行 |
| 另一个 TIMx 更新（TIMz>TIMy>TIMx） | HRTIM_TIMyCR 中的 TzU = 1 TIMxCR 中的 TyU=1 | 无 | 必须在 HRTIM_TIMxCR 中的 TzU = 1 时进行 HRTIM_TIMyCR 中的 TzU = 1 |
| 由计数器翻转引发的重复事件 | TyREPU = 1 | 有 | - |
| 由计数器复位引发的重复事件 | TyREPU = 1 | - | 请参见下文中的计数器复位情况 |
| 计数器翻转 | TyRSTU = 1 | 有 | - |

| 源 | 条件 | 传播 | 注释 |
| --- | --- | --- | --- |
| 计数器软件复位 | HRTIM_CR2 中的 TyRST = 1 | 无 | 可与 HRTIM_CR2 寄存器中的更新同时进行与更新同时完成 |
| TIMz 比较引发的计数器复位 | HRTIM_RSTyR 中的 TIMzCMPn | 有 | - |
| 外部事件引发的计数器复位 | HRTIM_RSTyR 中的 EXTEVNTn | 有 | - |
| 主定时器比较或主定时器周期引发的计数器复位 | HRTIM_RSTyR 中的 MSTCMPn 或 MSTPER | 有 | - |
| TIMy 比较引发的计数器复位 | HRTIM_RSTyR 中的 CMPn | 有 | - |
| 更新引起的计数器复位 | HRTIM_RSTyR 中的 UPDT | 无 | 传播会导致进入锁定状态（更新会引发复位，进而引发更新） |
| HRTIM_SCIN 引发的计数器复位 | HRTIM_TIMyCR 中的 SYNCRSTy | 无 | - |
| 软件更新 | TySWU = 1 | 无 | 所有软件更新位（TxSWU）均分组到 HRTIM_CR2 寄存器中，可用于同步更新 |

### 引发 TIMx 更新的 TIMx 计数器复位

下表列出了计数器复位源，并指出了这些源是否可用于生成更新事件。
工作条件：HRTIM_TIMxCR 寄存器中的 TxRSTU 位。

表 21.25 能够生成更新事件的复位事件

| 源 | 条件 | 传播 | 注释 |
|---|---|---|---|
| 计数器翻转 | - | 有 | - |
| 更新事件 | HRTIM_RSTxR 中的 UPDT | 无 | 传播会导致进入锁定状态（更新会引发复位，进而引发更新） |
| 外部事件 | HRTIM_RSTxR 中的 EXTEVNTn | 有 | - |
| TIMy 比较 | HRTIM_RSTxR 中的 TIMyCMPn | 有 | - |
| 主定时器比较 | HRTIM_RSTxR 中的 MSTCMPn | 有 | - |
| 主定时器周期 | HRTIM_RSTxR 中的 MSTPER | 有 | - |
| 比较 2 和比较 4 | HRTIM_RSTxR 中的 CMPn | 有 | - |
| 软件 | in HRTIM_CR2 中 TxRST=1 | 有 | - |
| HRTIM_SCIN | HRTIM_TIMxCR 中的 SYNCRSTx | 有 | - |

### 引发 TIMx 计数器复位的 TIMx 更新

下表列出了更新事件源，并指出了这些源是否可用于生成计数器复位事件。
工作条件：HRTIM_RSTxR 寄存器中的 UPDT 位置 1。

表 21.26 用于定时器复位的更新事件传播

| 源 | 条件 | 传播 | 注释 |
|---|---|---|---|
| 突发 DMA 结束 | UPDGAT[3:0] = 0001 | 有 | - |
| 由更新使能输入 hrtim_upd_en[3:1] 引发的更新 | UPDGAT[3:0] = 0011, 0100, 0101 | 有 | - |
| 突发 DMA 后的翻转引发的主定时器更新 | HRTIM_TIMxCR 中 MSTU = 1 HRTIM_MCR 中 BRSTDMA[1:0] = 10 | 有 | - |
| 翻转后的重复事件引发的主定时器更新 | HRTIM_TIMxCR 中 MSTU = 1 | 有 | - |

| | | | |
|---|---|---|---|
| 计数器复位（通过软件或 HRTIM_SCIN 复位）后的重复事件引发的主定时器更新 | HRTIM_MCR 中 MREPU = 1 | 无 | - |
| 软件触发的主定时器更新 | HRTIM_TIMxCR 中 MSTU = 1 HRTIM_CR2 中 MSWU = 1 in | 无 | 所有软件更新位（TxSWU）均分组到 HRTIM_CR2 寄存器中，可用于同步更新 |
| TIMy 计数器翻转引发的 TIMy 更新 | HRTIM_TIMxCR 中 TyU = 1 HRTIM_TIMyCR 中 TyRSTU = 1 | 有 | - |
| TIMy 重复事件引发的 TIMy 更新 | HRTIM_TIMxCR 中 TyU = 1 HRTIM_TIMyCR 中 TyREPU = 1 | 有 | - |
| 外部事件或 TIMy 比较（通过 TIMy 复位）引发的 TIMy 更新 | HRTIM_TIMxCR 中 TyU = 1 HRTIM_TIMyCR 中 TyRSTU = 1 HRTIM_RSTyCR 中 EXTEVNTn 或 CMP4/2 | 有 | - |
| 上面列出的源以外的源引发的 TIMy 更新 | HRTIM_TIMxCR 中 TyU = 1 | 无 | - |
| 翻转后的重复事件 | HRTIM_TIMxCR 中 TxREPU = 1 | 有 | - |
| 计数器复位后的重复事件 | | 无 | - |
| 定时器复位 | HRTIM_TIMxCR 中 TxRSTU = 1 | 无 | 传播会导致进入锁定状态（复位会引发更新，进而引发复位） |
| 软件 | HRTIM_CR2 中 TxSWU | 无 | - |

### 21.3.14 输出管理

每个定时单元都控制着一对输出。输出有三种工作状态：

- RUN：此状态为主要工作状态，在该状态下，输出会获取在纵横开关单元中设定的有效电平或无效电平。
- IDLE：当输出通过软件禁用或处于突发模式工作期间（此时，正常工作模式下会暂时禁止输出，更多详细信息请参考第 15 小节），该状态是 HRTIM 复位后的默认工作状态。该状态永久有效或无效。
- FAULT：此状态为安全状态，FAULTx 输入上存在关断请求时会进入此状态。该状态可以是永久有效、无效或高阻态（Hi-Z）。

输出状态由 HRTIM_OENR 寄存器中的 TxyOEN 位以及 HRTIM_ODSR 寄存器中的 TxyODS 位指示，如下表所示。

表 21.27 输出状态编程，x= A..F，y = 1 或 2

| TxyOEN（控制/状态）（由软件置 1，硬件清 0） | TxyODS（状态） | 输出工作状态 |
|---|---|---|
| 1 | x | RUN |
| 0 | 0 | IDLE |
| 0 | 1 | FAULT |

TxyOEN 位既是控制位，也是状态位：该位必须通过软件置 1，才能使输出处于 RUN 模式。输出恢复 IDLE 或 FAULT 模式时，该位会通过硬件清零。TxyOEN 位清零后，TxyODS 位会指示输出处于 IDLE 状态还是 FAULT 状态。HRTIM_ODISR 寄存器中的第三位可通过软件禁止输出。

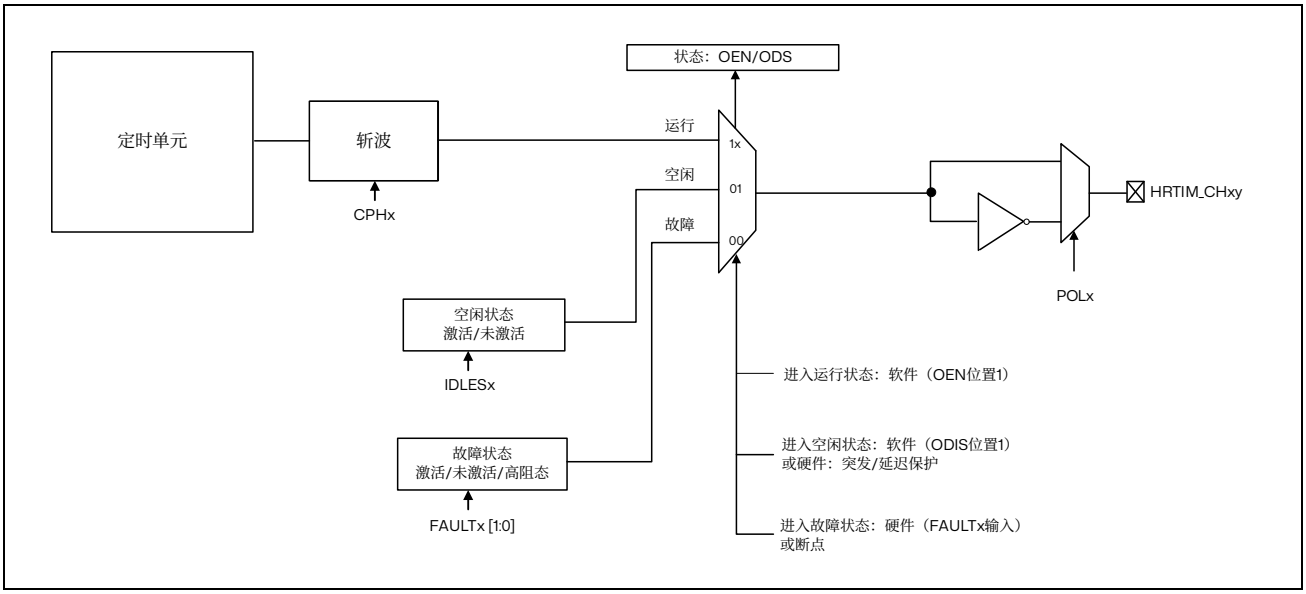



图 21.53 输出管理概览

下图总结了三种状态对应的位值以及转换的触发方式。任何外部或内部故障源均可触发故障。

当突发模式或延迟保护有效时，可进入空闲状态。

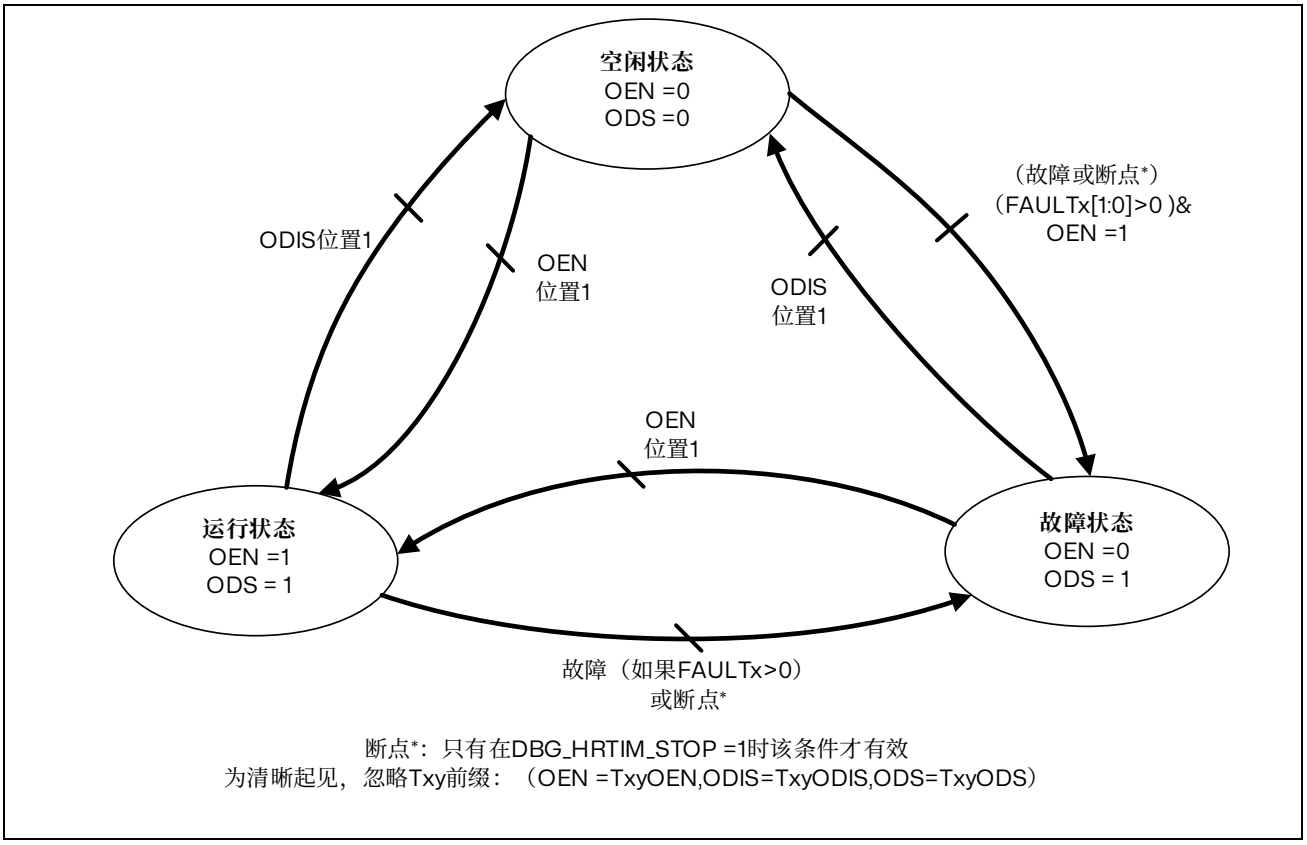



图 21.54 HRTIM 输出状态和转换

FAULT 和 IDLE 电平定义为有效或无效。有效（或无效）是指定时器输出上可引发电源开关闭合（对于无效状态，则为电源开关断开）的电平。

IDLE 状态的优先级最高：即使 FAULT 条件仍有效，也可以实现由 ODIS 位置 1 触发的 FAULT→IDLE 转换。

FAULT 状态的优先级要高于 RUN 状态：如果 TxyOEN 位置 1 的同时发生了故障事件，则将进入 FAULT 状态。IDLE→FAULT 转换中给出了相应的条件（如上图所示）：需要使能故障保护（FAULTx[1:0] 位 = 01、10、11），且 Txy OEN 位在故障有效时置 1（如果 DBG_HRTIM_STOP = 1，则在断点期间置 1）。

输出极性通过 HRTIM_OUTxR 中的 POLx 位编程。如果 POLx = 0，极性为正（输出高电平有效）；如果 POLx = 1，极性为负（输出低电平有效）。实际上，极性的定义取决于要驱动的电源开关（PMOS 与 NMOS）或栅极驱动器极性。

每路输出 FAULT 状态下的输出电平通过 HRTIM_OUTxR 中的 FAULTx[1:0]位配置，具体如下：

- 00：输出永不进入故障状态，停留在 RUN 或 IDLE 状态
- 01：处于 FAULT 状态时，输出为有效电平
- 10：处于 FAULT 状态时，输出为无效电平
- 11：处于 FAULT 状态时，输出为三态。必须通过诸如上拉或下拉电阻在外部强制进入安全状态。

*注：只要输出处于* FAULT *状态，就不应更改* FAULTx[1:0] *位。*

IDLE 状态下的输出电平通过 HRTIM_OUTxR 中的 IDLESx 位配置，具体如下：

- 0：处于 IDLE 状态时，输出为无效电平
- 1：处于 IDLE 状态时，输出为有效电平

当 TxyOEN 位置 1 进入 RUN 状态时，输出会立即连接到纵横开关输出。如果定时器时钟停止，电平将为无效（HRTIM 复位后）或相当于 RUN 电平（定时器停止、且输出禁止时）。

在 HRTIM 初始化过程中，在使输出进入 RUN 模式之前，可在 HRTIM_SETx1R 和 HRTIM_RSTx1R 寄存器中使用软件强制输出置位和复位预先设定输出电平。

### 21.3.15 突发模式控制器

突发模式控制器可通过硬件使输出交替地进入 IDLE 状态和 RUN 状态，以便通过可编程周期性和占空比跳过一些开关周期。

轻载运行时，突发模式工作常用于电源转换器中。该模式减少了输出跳变次数以及关联的开关损耗，从而显著提高转换器效率。

在突发模式下运行时，会在空闲周期（等于几个计数周期）后输出一个或几个脉冲，空闲周期内通常不会生成任何输出脉冲，如下图中的示例所示。

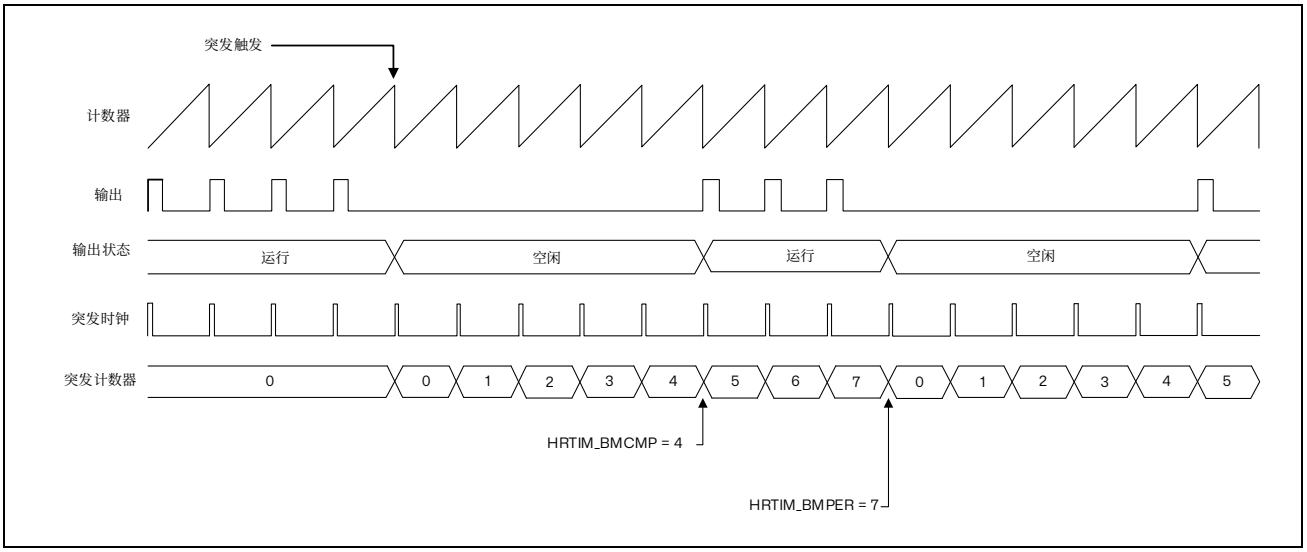



图 21.55 突发模式工作示例

突发模式控制器包括：

- 可由各种源提供时钟的计数器（HRTIM 之内或之外，通常是 PWM 周期结束时）。
- 定义空闲周期数的比较寄存器：HRTIM_BMCMP。
- 定义突发重复率的周期寄存器（相当于空闲周期与运行周期之和）：HRTIM_BMPER。

突发模式控制器能够接管对 10 路 PWM 输出中的任何一路的控制。突发模式工作期间，每路输出的状态通过 HRTIM_OUTxR 寄存器中的 IDLESx 和 IDLEMx 位编程，如下表所述。

表 21.28 突发模式的定时器输出编程

| IDLEMx | IDLESx | 突发模式时的输出状态 |
|---|---|---|
| 0 | X | 无操作：输出不受突发模式操作的影响 |
| 1 | 0 | 在突发期间输出无效 |
| 1 | 1 | 在突发期间输出有效 |

*注：突发模式有效时，不得更改* IDLEMx *位。*

突发模式控制器仅会作用于输出级。空闲周期内仍会生成一些事件：

- 输出置位/复位中断或 DMA 请求
- 基于 Tx2 输出信号的外部事件过滤
- 由输出置位/复位触发的捕获事件

突发模式期间，即使 TxBM 位已置 1，HRTIM_SCOUT 输出上也不会生成启动事件或复位事件。

### 工作模式

在给定的定时单元上使用突发模式之前，必须使能计数器（TxCEN 位置 1）。突发模式通过 HRTIM_BMCR 寄存器中的 BME 位使能。

通过 HRTIM_BMCR 寄存器中的 BMOM 位，定时单元可在连续模式或单发模式下工作。BMOM = 1 时，使能连续模式。在 HRTIM_BMCR 中的 BMSTAT 位复位终止突发工作之前，都会保持突发工作状态。

在单发模式下（BMOM = 0），会执行一次空闲序列，随后会触发突发模式，之后会立即恢复正常定时器操作。

空闲周期和运行周期的持续时间通过突发模式计数器和 2 个寄存器定义。HRTIM_BMCMPR 寄存器以计数定义所选定时器处于空闲状态的持续时间（空闲周期）。HRTIM_BMPER 定义总突发模式周期（空闲周期与运行周期之和）。触发初始突发模式后，空闲周期长度为 HRTIM_BMCMPR+1，总突发周期为 HRTIM_BMPER+1。

*注：突发模式周期不得小于或等于通过* DTRx[8:0] *和* DTFx[8:0] *位域定义的死区持续时间。*

突发模式工作期间，定时单元和主定时器的计数器可停止和复位。HRTIM_BMCR 保留了 6 个控制位来实现此目的：MTBM（主定时器）和 TABM..TEBM（用于定时器 A..E）。

当 MTBM 或 TxBM 位复位时，会保留计数器时钟。举例来说，这样可在多相位系统中保持与其他定时器的相位关系。

如果 MTBM 或 TxBM 位已置 1，则在突发空闲周期期间，相应的计数器会停止，并会保持在复位状态下。这样便可使定时器在退出空闲状态时重新开始一个完整的周期。如果 SYNCSRC[1:0] = 00 或 10（主定时器启动或定时器 A 启动时的同步输出），退出突发模式时，会在 HRTIM_SCOUT 输出上发送一个脉冲。

*注：当均衡空闲模式激活时* (DLYPRT[1:0] = 0x11)*，不得将* TxBM *位置* 1。

### 突发模式时钟

突发模式控制器计数器可由多个时钟源提供时钟，具体通过 HRTIM_BMCR 寄存器中的 BMCLK[3:0]位来选择：

- BMCLK[3:0] = 0000 至 0101：主定时器和 TIMA..E 的复位/翻转事件。这样可使突发模式空闲周期和运行周期与定时单元计数周期对齐（均处于自由运行和计数器复位模式）。
- BMCLK[3:0] = 0110 至 1001：时钟由 hrtim_bm_ck[4:1]输入提供，这些输入连接到通用定时器，如下表所示。在这种情况下，突发模式空闲周期和运行周期不必与定时单元计数周期对齐（输出上的脉冲可能中断，从而导致波形的占空比被修改）。
- BMCLK[3:0] = 1010：$f_{HRTIM}$ 时钟由通过 HRTIM_BMCR 寄存器中的 BMPRSC[3:0]位定义的系数进行预分频。在这种情况下，突发模式空闲周期和运行周期不必与定时单元计数周期对齐（输出上的脉冲可能中断，从而导致波形的占空比被修改）。

TIMx OC 输出上的脉冲宽度必须至少为 N 个 $f_{HRTIM}$ 时钟周期，才能被 HRTIM 突发模式控制器检测到。

### 突发模式触发

要触发突发模式工作，可使用 32 个源，这些源通过 HRTIM_BMTRGR 寄存器选择：

- 软件触发（通过软件置 1，通过硬件复位）
- 6 个主定时器事件：重复、复位/翻转、比较 1 到 4
- 来自定时器 A..E 的 5 x 4 个事件：重复、复位/翻转、比较 1 到 2
- 外部事件 7（包括 TIMA 事件过滤）和 8（包括 TIMD 事件过滤）
- 外部事件 7 之后的定时器 A 周期（包括 TIMA 事件过滤）
- 外部事件 8 之后的定时器 D 周期（包括 TIMD 事件过滤）
- hrtim_bm_trg 输入上的片上事件（连接到通用定时器的 TRGO 输出），具体细节见下表。

这些源可结合起来使用，实现多源并发触发。

突发模式不可再次触发。在连续模式下，突发模式终止之前，会忽略新的触发；而在单发模式下，包含运行周期的当前突发结束之前（HRTIM_BMPER+1 个周期），会忽略触发。这一点同样适用于软件触发（即使软件位已丢弃，也会通过硬件复位）。

下图显示了突发模式是如何响应外部事件而启动的（立即启动，或在事件后的定时器周期启动）。

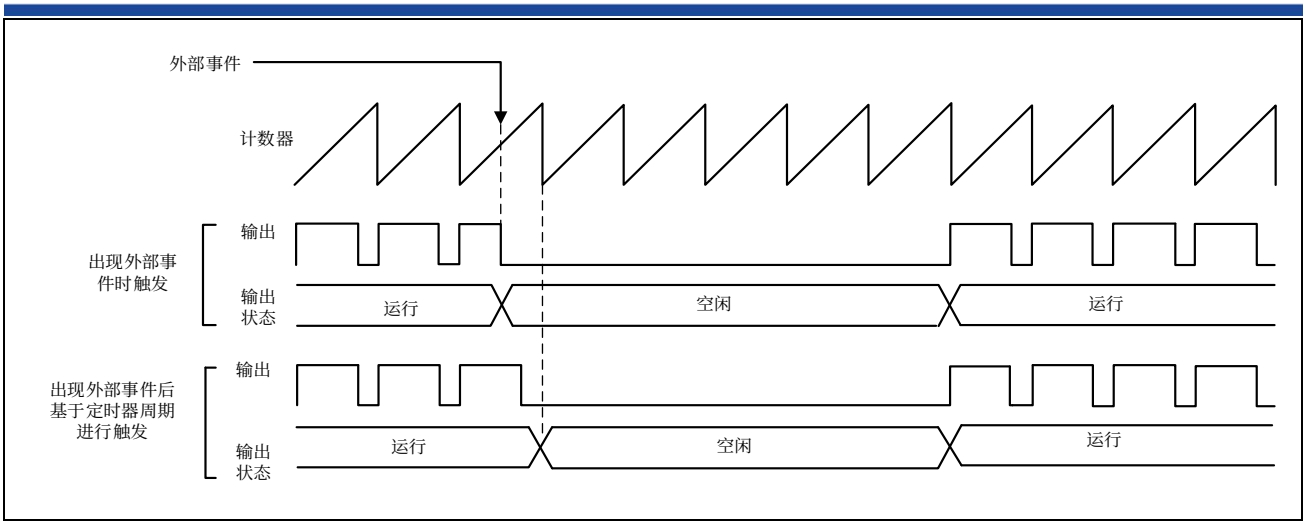



图 21.56 发生外部事件时触发突发模式

对于 TAEEV7 和 TDEEV8 组合触发（在外部事件后的定时器周期触发），无论采用何种突发模式编程以及正在进行何种突发操作，外部事件检测始终有效：

- 如果在外部事件与定时器周期事件之间突发模式已使能（BME=1）或触发已使能（BMTRG 寄存器中的 TAEEV7 或 TDEEV8 位置 1），则会触发突发模式。
- 即使外部事件在突发结束之前发生，也会再次触发单发突发模式（只要突发之后发生相应的周期事件）。

*注：TAEEV7 和 TDEEV8 触发仅在周期事件后有效。如果计数器在周期事件之前复位，则会丢弃挂起的 hrtim_evt7 和 hrtim_evt8 事件。*

### 突发模式延迟进入

默认情况下，输出将在突发模式触发后立即获取其空闲电平（空闲电平取决于 IDLES1 和 IDLES2 设置）。

还可以延迟进入突发模式，并在输出获取其空闲状态之前的可编程周期内将输出强制为无效状态。驱动两路互补输出（其中一路输出具有有效空闲状态）时，可利用此特性避免超出死区，如下图所示。这样可避免半桥中出现击穿电流的风险，但会导致对突发模式进入的延迟响应。

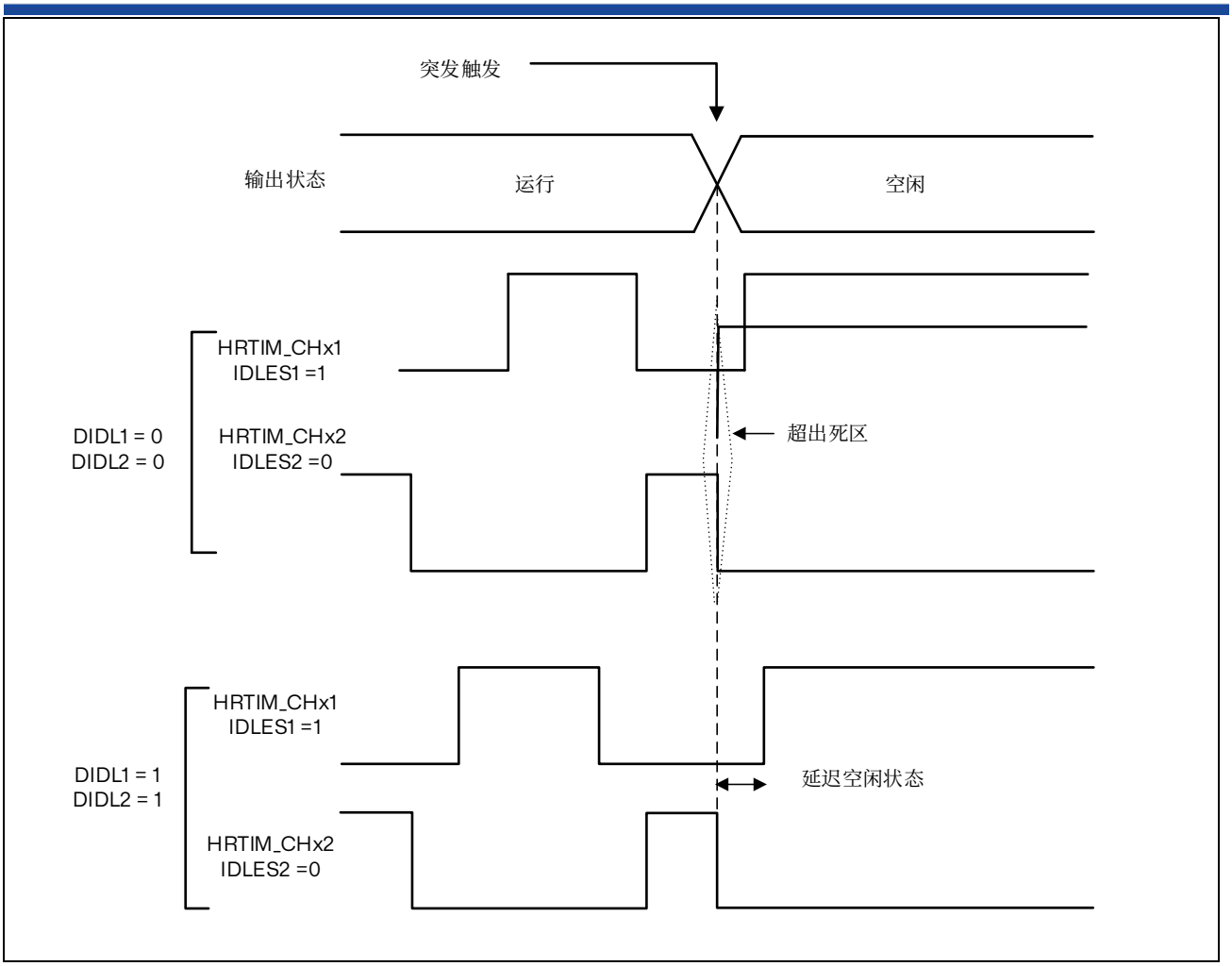



图 21.57 使能死区且 IDLESx = 1 时的延迟突发模式进入

延迟突发进入模式通过 HRTIM_OUTxR 寄存器中的 DIDLx 位使能（每路输出一个使能位）。该模式会在输出获取其空闲状态之前强制插入死区。每路 TIMx 输出都有自己的死区值：

- 当 DIDL1 = 1 时，为输出 1 上的 DTRx[8:0]
- 当 DIDL2 = 1 时，为输出 2 上的 DTFx[8:0]

仅当其中一路输出在突发模式期间具有有效空闲电平（IDLES=1）、且使用的死区为正值（SDTR/SDTF 设为 0）时，才可将 DIDLx 位置 1。

*注：延迟突发进入模式使用死区发生器资源。因此，如果 2 个 DIDLx 位中有任何一位置 1，且相应的定时单元使用死区插入（HRTIM_OUTxR 中的 DTEN 位置 1），则无法将 timerx 输出 2 用作外部事件过滤器（Tx2 过滤信号不可用）。*

如果由 DTRx[8:0]和 DTFx[8:0]定义的持续时间小于 3 个 $f_{HRTIM}$ 时钟循环周期，则必须应用第 7 小节中列出的与窄脉冲管理相关的限制。

如果突发模式进入在常规死区内到达，死区会中止，并会重新开始新的对应于无效周期的死区，如下图所示。

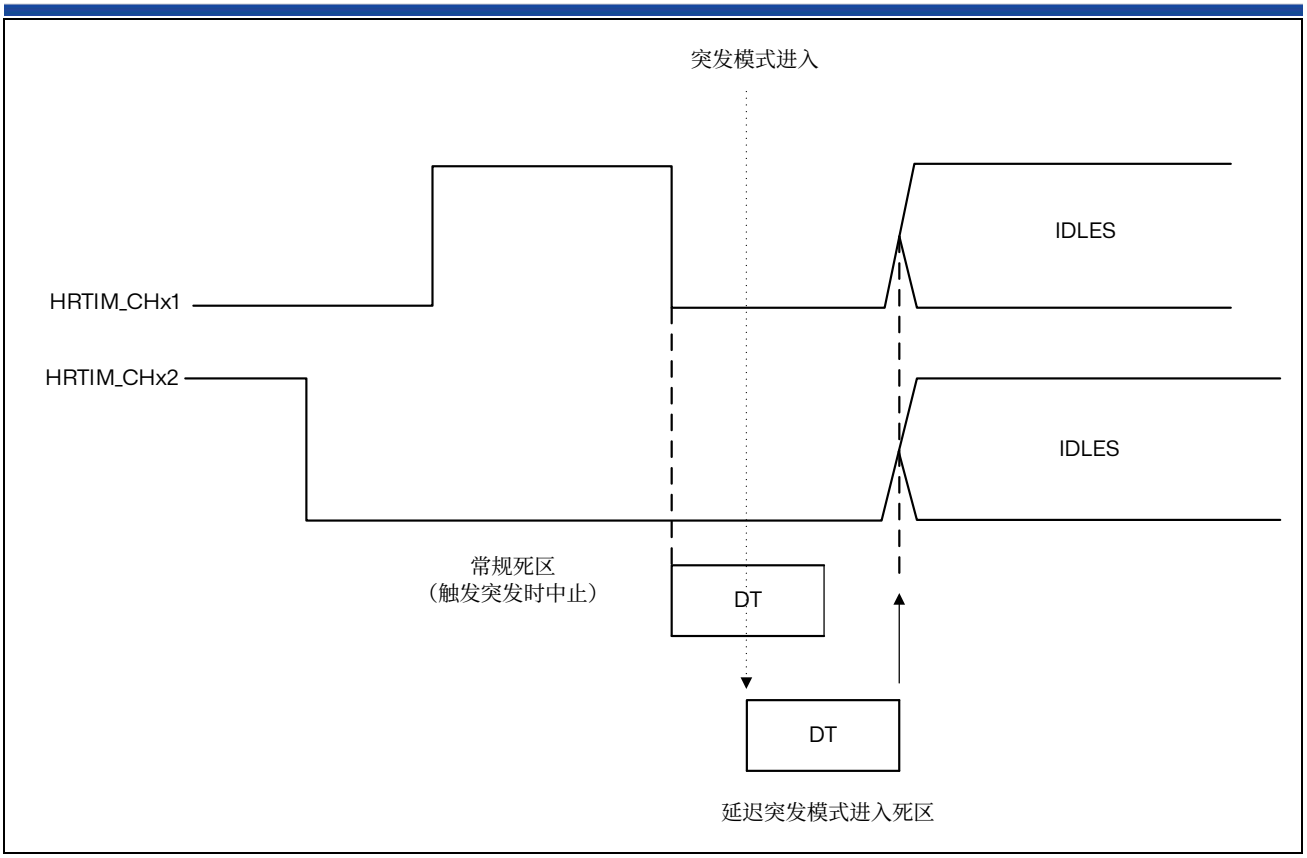



图 21.58 死区内的延迟突发模式进入

### 突发模式退出

突发模式退出通过软件强制进行（连续模式下），或在经过空闲周期后进行（单发模式下）。两种情况下，计数器都会立即重启（如果计数器通过 MTBM 或 TxBM 位= 1 保持在复位状态），但输出状态仅会在编程的置位/复位事件后才会从空闲模式有效跳变为工作模式

如果 HRTIM_IER 寄存器中的 BMPERIE 使能位置 1，则会在单发模式和连续模式下生成突发周期中断。该中断可用于在连续突发模式下将突发模式退出与突发周期进行同步。

下图显示了死区使能后是如何恢复正常工作的。虽然突发模式退出是立即进行的，但仅对任何互补输出上的第一个置位事件有效。

图中显示了两种不同情况：

1. 纵横开关输出波形上的信号无效时，突发模式结束。Tx1 输出上发生置位事件时会恢复 Tx1 和 Tx2 的有效状态，Tx2 输出不会在退出突发模式时获取互补电平。
2. 突发模式在纵横开关输出波形有效时结束：Tx2 输出上发生置位事件时会恢复工作，Tx1 不会在退出突发模式时立即获取有效电平。

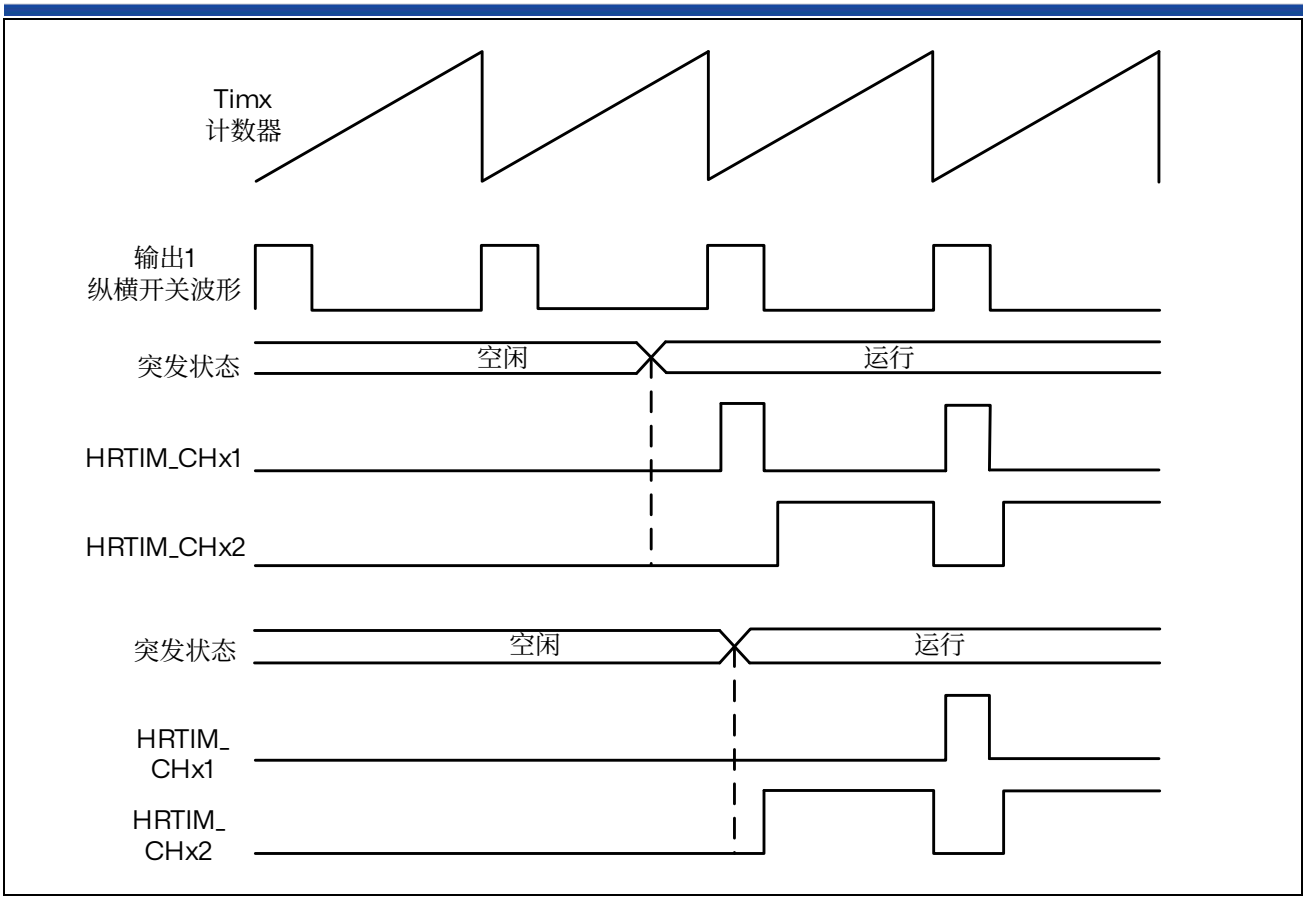



图 21.59 死区发生器使能时的突发模式退出

如果使能了推挽模式，上述行为会略有不同。如果输出无效，推挽模式会在周期开始使强制进行输出复位，如果前一周期内输出为高电平，则会对称地将输出强设置为有效电平。

因此，即使未明确编程跳变，也可在退出突发模式时复位空闲状态有效的输出。出于对称性的原因，即使未明确编程跳变，只要输出进入空闲状态时为有效电平，即可在退出突发模式时将输出置 1。

### 突发模式寄存器预装载和更新

当 BMPREN 位（突发模式预装载使能）可进行突发模式比较并预装载周期寄存器（HRTIM_BMCMP 和 HRTIM_BMPER）。

当 BMPREN 置 1 时，预装载寄存器的内容会传输到活动寄存器：

- 当突发模式被使能时（BME = 1），
- 在突发模式周期结束时。

如果对 HRTIM_BMPER 周期寄存器进行写操作，则会暂时禁止更新，直到向 HRTIM_BMCMP 比较寄存器进行写操作为止，这样可确保两个寄存器在进行修改时是一致的。

如果只需要更改比较寄存器，则只需要进行一次写操作。如果只需要更改周期，还需要重新写入比较寄存器，以便将新值纳入考虑范围。

当 BMPREN 位复位时，对 BMCMPR 和 BMPER 进行写访问会直接更新活动寄存器。此时，对于以下 2 种情况，需要考虑何时在总突发周期内进行更新：

a）比较寄存器更新

如果新的比较值大于当前突发模式计数器值，则会在当前周期内考虑新的比较值。

如果新的比较值小于当前突发模式计数器值，在连续模式下，会在下一个周期内考虑新的比较值；在单发模式下，则会忽略新的比较值（不会出现比较匹配的情况，并将持续处于空闲状态，直至空闲周期结束）。

b）周期寄存器更新

如果新的周期值大于当前突发模式计数器值，则会在当前周期内考虑此更改。

*注：如果新的周期值小于当前突发模式计数器值，则不会考虑新的周期值，突发模式计数器将溢出（达到* 0xFFFF *时溢出），更改将在下一周期生效。在单发模式下，计数器将在达到* 0xFFFF *时翻转，突发模式将重新启动并再持续一个周期，直到达到新的编程值。*

**使用复合寄存器模拟突发模式**

突发模式控制器仅会控制单个转换器的一个定时器或一组定时器。需要为多个独立定时器使用突发模式时，可以使用 DMA 和 HRTIM_CMP1CxR 复合寄存器仿真简单的突发模式控制器，复合寄存器会保存重复寄存器和比较 1 寄存器的别名。

此方法适用于仅需要使用简单 PWM 来更新占空比的转换器（通常是降压转换器）。在这种情况下，使用 CMP1 寄存器复位输出（并定义占空比），而输出会在发生周期事件时置 1。

此时，对 CMP1CxR 进行一次 32 位写访问便足以定义占空比（使用 CMP1 值）以及保持该占空比的周期数（使用重复值）。要实现突发模式，只需要在连续模式下通过 DMA（在发生重复事件时）传输两个 32 位数据，数据结构如下：

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

举例来说，数值：
{0x0003 0000}：CMP1 = 0，持续 3 个周期
{0x0001 0800}：CMP1 = 0x0800，持续 1 个周期
提供的突发模式每 6 个 PWM 周期会有 2 个周期有效，如下图所示。

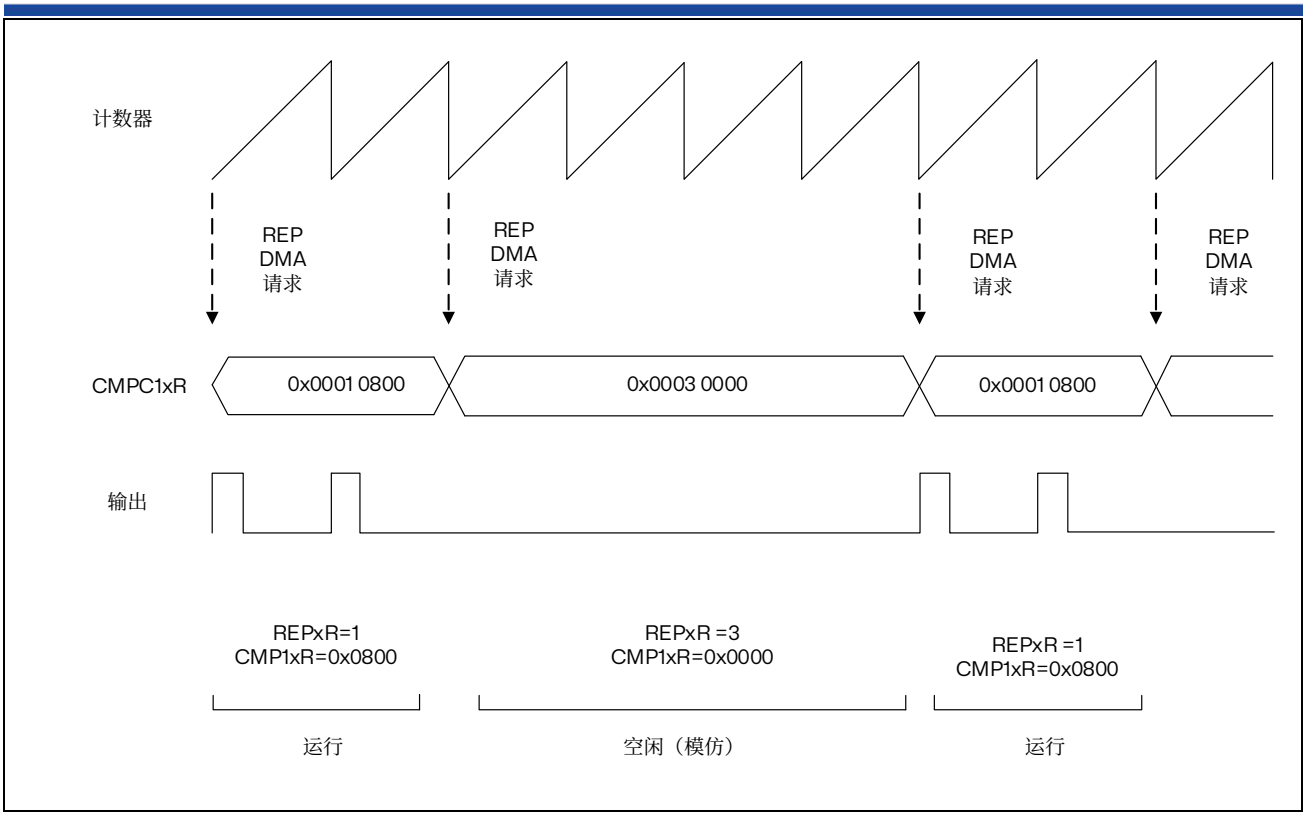



图 21.60 突发模式仿真示例

### 21.3.16 斩波

可在定时单元输出信号上添加高频载波以驱动隔离变压器。此操作是在插入极性之前对输出级执行的（如下图所示），会使用 HRTIM_OUTxR 寄存器中的 CHP1 和 CHP2 位分别在输出 1 和输出 2 上使能斩波。

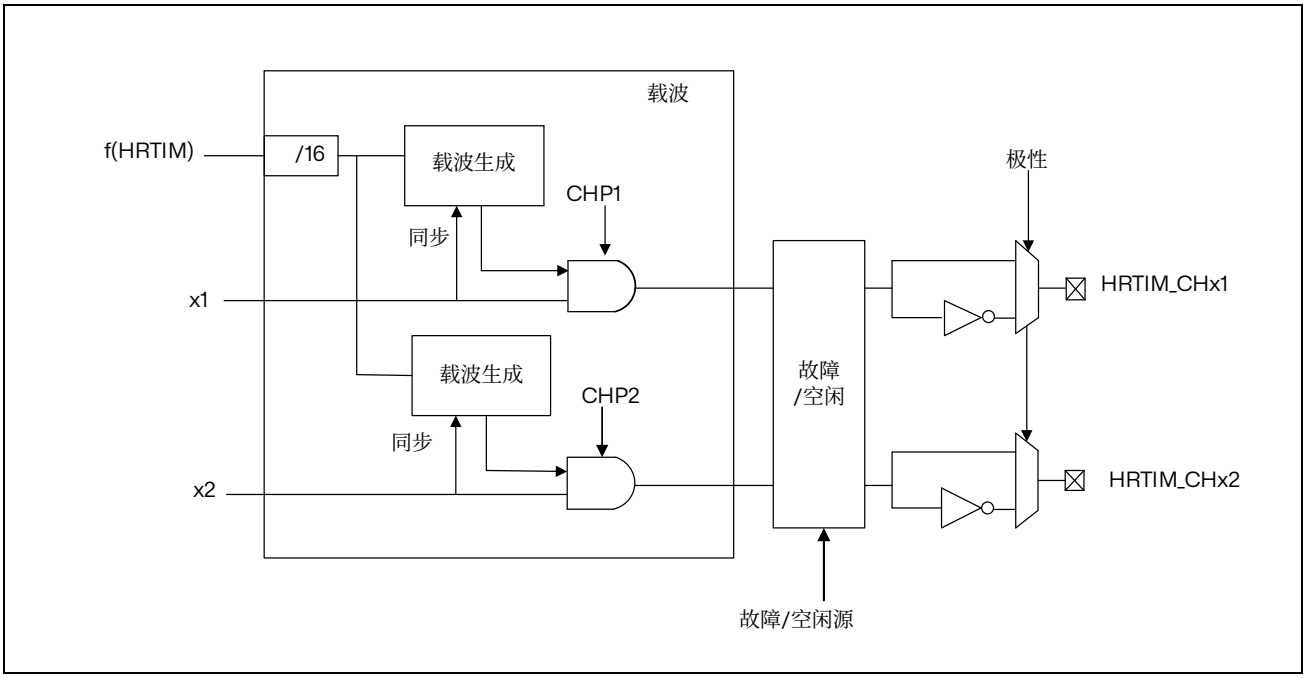



图 21.61 载波频率信号插入

可通过 HRIM_CHPxR 寄存器调整斩波参数，以在脉冲开始处定义一个特定脉宽，后跟频率和占空比可编程的载波频率，如下图所示。

CARFRQ[3:0]位按照公式 $F_{CHPFRQ}$ = $f_{HRTIM}$ / (16 x (CARFRQ[3:0]+1))定义频率，范围从 12 MHz 到 750 KHz（对于 $f_{HRTIM}$ = 192 MHz)。

可通过 CARDTY[2:0]调整占空比（步长为 1/8），占空比范围为 0/8 到 7/8。CARDTY[2:0] = 000 时（占空比 = 0/8），输出波形仅包含参考波形上升沿之后的启动脉冲，不会添加任何载波。

初始脉冲的脉宽是通过 STRPW[3:0]位域定义的，具体如下：

t1STPW = (STRPW[3:0]+1) x 16 x $t_{HRTIM}$。脉宽范围为 83.3 ns 到 1.33 µs（对于 $f_{HRTIM}$=192 MHz)。

载波频率参数是根据 $f_{HRTIM}$ 频率定义的，与 CKPSC[2:0]设置无关。

在斩波模式下，载波频率和初始脉宽会通过逻辑与函数与参考波形相结合。初始脉冲结束时会执行同步，以获得重复的信号波形。

斩波信号会在输出波形有效状态结束时停止，不会等到当前载波周期结束。因此，斩波信号包含的脉冲可能比编程的脉冲短。

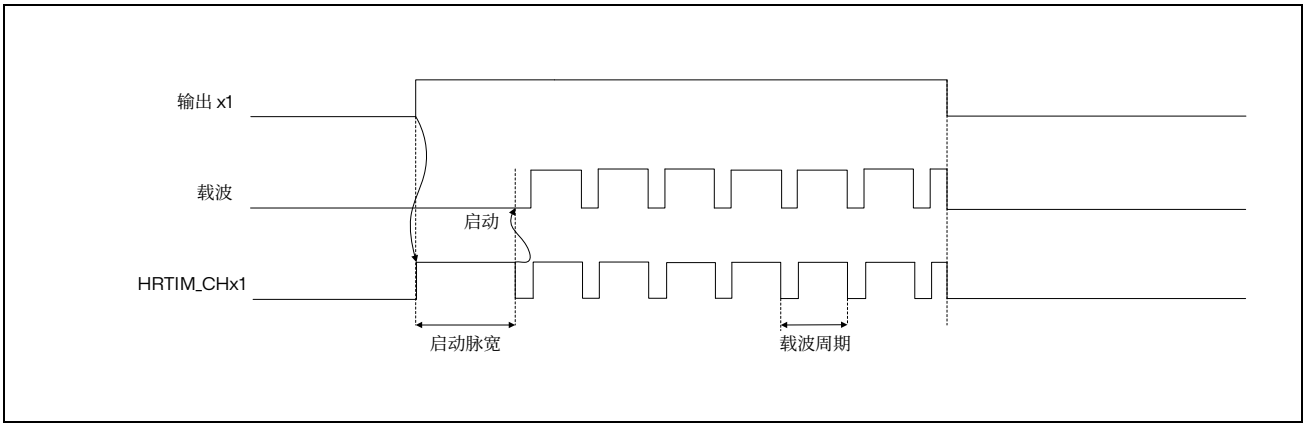



图 21.62 已使能斩波模式的 HRTIM 输出

*注：必须在通过* HRTIM_OENR *寄存器中的* TxyOEN *位使能输出之前将* CHP1 *和* CHP2 *位置*1*。斩波模式激活时（两个* CHPx *位中至少有一个置*1*），不能修改* CARFRQ[2:0]、CARDTY[2:0] *和* STRPW[3:0] *位域。*

### 21.3.17 故障保护

HRTIMER 具有通用的故障保护电路，可在发生异常操作时禁止输出。一旦触发故障，输出便会进入预定义的安全状态。输出通过软件重新使能之前，都会保持此状态。如果是永久故障请求，即使软件尝试重新使能输出，输出也将保持其故障状态，直至故障源消失。

HRTIM 具有 6 个 FAULT 输入通道；所有通道均可用，并可结合起来用于 6 个定时单元中的每一个，如下图所示。

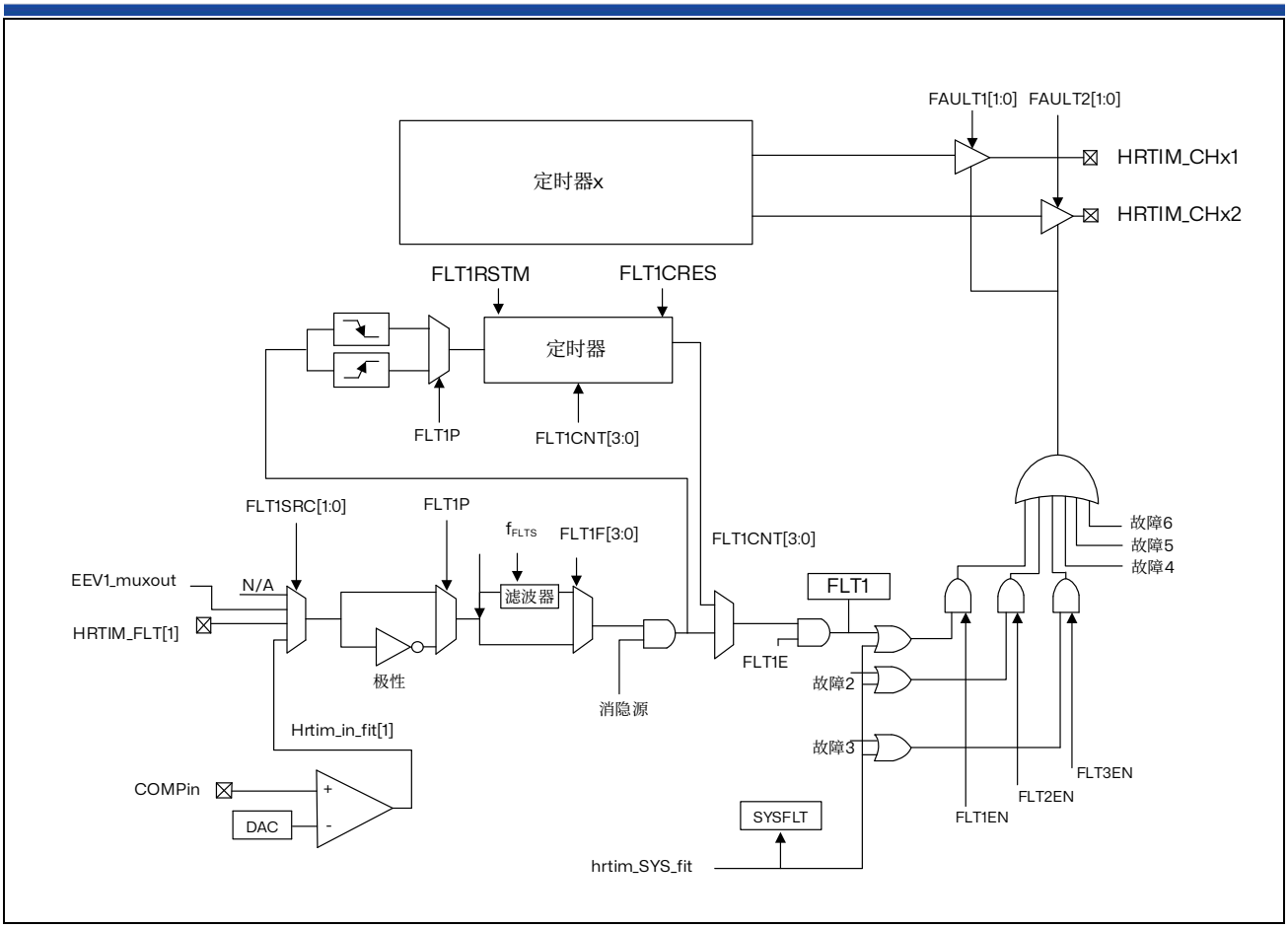



图 21.63 故障保护电路（完全表示了 FAULT1，部分显示了 FAULT2..6）

在连接到定时单元之前，每条故障通道均可完全通过 HRTIM_FLTINR1 和 HRTIM_FLTINR2 寄存器进行配置。FLTxSRC 和 FLTxSRC[1:0]位用于选择故障信号的来源，可以是数字输入或内部事件（内置比较器输出）。

表 22.7 故障输入总结了 6 个故障通道的每个通道可用的信号来源：

上图中提到的 EEVx_muxout 事件是在 hrtim_eevx[4:1]输入复用器之后产生的，该复用器由 EExSRC[1:0]位控制。

可通过 HRTIM_FLTINRx 寄存器中的 FLTxP 极性位选择信号的极性，以定义有效电平。如果 FLTxP= 0，信号低电平有效；如果 FLTxP = 1，信号高电平有效。

可在极性设置后过滤故障信息。如果 FLTxF[3:0]位域设为 0000，则不会对信号进行过滤，信号将独立于 $f_{HRTIM}$ 时钟异步操作。对于所有其他 FLTxF[3:0]位域值，会对信号进行数字过滤。数字滤波器由计数器组成，需要使用 N 个有效样本来验证输出跳变。如果输入值在计数器达到 N 值之前发生变化，计数器会复位，跳变会被丢弃（视为伪事件）。如果计数器达到 N，跳变被视为有效，并会作为正确的外部事件进行传输。

因此，数字滤波器会向正在进行滤波的外部事件添加延迟，延迟时长视采样时钟和滤波器长度（预期的有效样本数）而定。下图显示了伪故障信号的过滤方式。

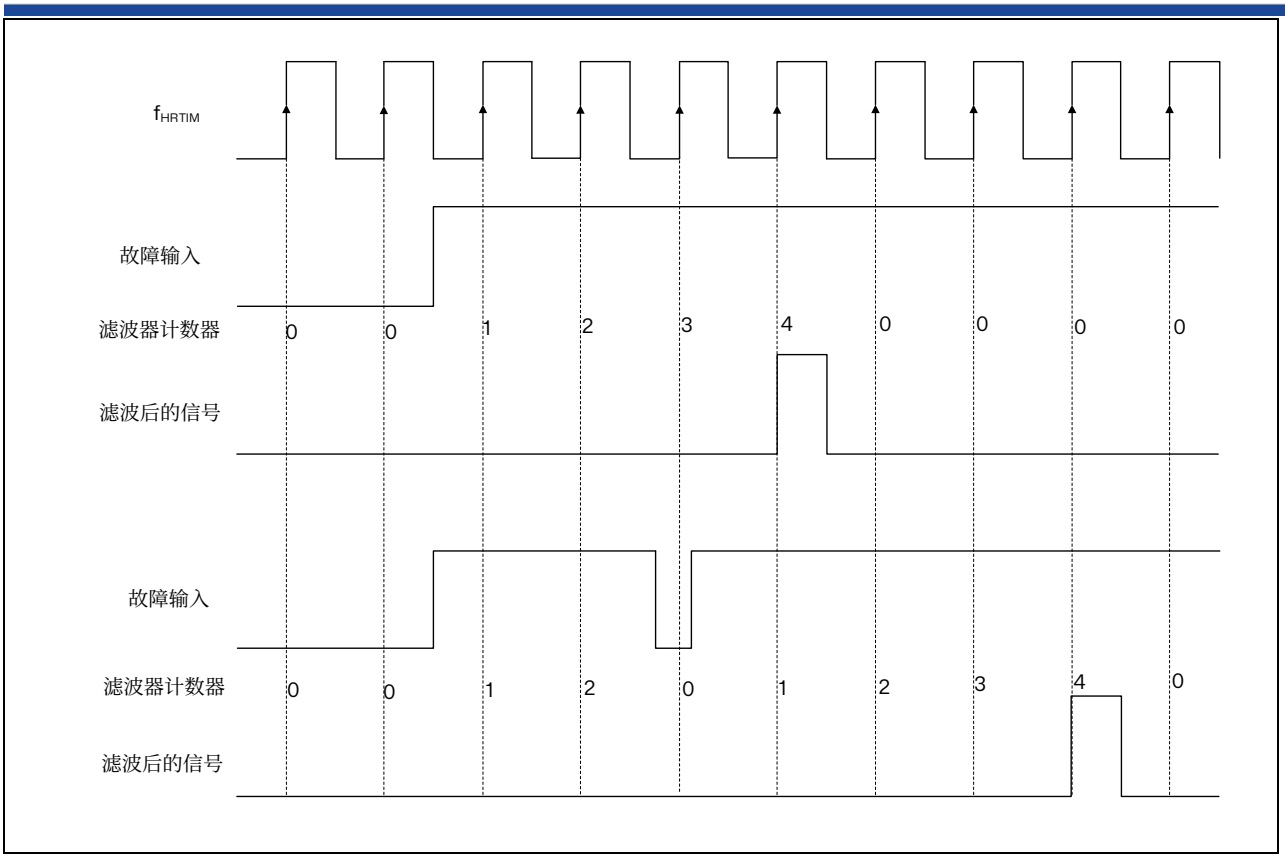



图 21.64 故障信号过滤（FLTxF[3:0]= 0010：$f_{SAMPLING} = f_{HRTIM}$， N = 4）

过滤周期从 2 个 $f_{HRTIM}$ 时钟周期到 8 个 32 分频的 $f_{FLTS}$ 时钟周期。$f_{FLTS}$ 通过 HRTIM_FLTINR2 寄存器中的 FLTSD[1:0]位定义。

下表总结了采样速率和滤波器长度。必须将滤波器长度减去 1 个采样时钟周期的抖动，以考虑因采样造成的不确定性，从而实现有效过滤。

表 21.29 采样速率和滤波器长度与 FLTFxF[3:0]和时钟设置的关系

| - | $f_{FLTS}$ 与 FLTSD[1:0] | | | | 当 $f_{HRTIM}$=192 MHz 时的滤波器长度 | |
|---|---|---|---|---|---|---|
| FLTFxF[3:0] | 00 | 01 | 10 | 11 | 最小 | 最大 |
| 0001,0010,0011 | $f_{HRTIM}$ | $f_{HRTIM}$ | $f_{HRTIM}$ | $f_{HRTIM}$ | $f_{HRTIM}$, N =2<br>10.4 ns | $f_{HRTIM}$, N =8<br>41.7 ns |
| 0100, 0101 | $f_{HRTIM}$ /2 | $f_{HRTIM}$ /4 | $f_{HRTIM}$ /8 | $f_{HRTIM}$ /16 | $f_{HRTIM}$ /2, N =6<br>62.5 ns | $f_{HRTIM}$ /16, N =8<br>667 ns |
| 0110, 0111 | $f_{HRTIM}$ /4 | $f_{HRTIM}$ /8 | $f_{HRTIM}$ /16 | $f_{HRTIM}$ /32 | $f_{HRTIM}$ /4, N =6<br>125 ns | $f_{HRTIM}$ /32, N =8<br>1.33 µs |
| 1000, 1001 | $f_{HRTIM}$ /8 | $f_{HRTIM}$ /16 | $f_{HRTIM}$ /32 | $f_{HRTIM}$ /64 | $f_{HRTIM}$ /8, N =6<br>250 ns | $f_{HRTIM}$ /64, N =8<br>2.67 µs |
| 1010, 1011, 1100 | $f_{HRTIM}$ /16 | $f_{HRTIM}$ /32 | $f_{HRTIM}$ /64 | $f_{HRTIM}$ /128 | $f_{HRTIM}$ /16, N =5<br>417 ns | $f_{HRTIM}$ /128, N =8<br>5.33 µs |
| 1101, 1110, 1111 | $f_{HRTIM}$ /32 | $f_{HRTIM}$ /64 | $f_{HRTIM}$ /128 | $f_{HRTIM}$ /256 | $f_{HRTIM}$ /32, N =5<br>833 ns | $f_{HRTIM}$ /256, N =8<br>10.67 µs |

### 故障消隐和事件计数

故障输入可以暂时禁用，以消隐伪故障事件。消隐源列在以下表中。

表 21.30 故障输入消隐事件

| - | FLTxBLKS = 0，重置对齐窗口 | | FLTxBLKS = 1 移动窗口 | |
|---|---|---|---|---|
| 故障输入 | 消隐窗口开始 | 消隐窗口结束 | 消隐窗口开始 | 消隐窗口结束 |
| hrtim_flt1[4:1] | Timer A 重置/翻转 | Timer A CMP3 事件 | Timer A CMP4 事件 | Timer A CMP3 事件 |
| hrtim_flt2[4:1] | Timer B 重置/翻转 | Timer B CMP3 事件 | Timer B CMP4 事件 | Timer B CMP3 事件 |
| hrtim_flt3[4:1] | Timer C 重置/翻转 | Timer C CMP3 事件 | Timer C CMP4 事件 | Timer C CMP3 事件 |
| hrtim_flt4[4:1] | Timer D 重置/翻转 | Timer D CMP3 事件 | Timer D CMP4 事件 | Timer D CMP3 事件 |
| hrtim_flt5[4:1] | Timer E 重置/翻转 | Timer E CMP3 事件 | Timer E CMP4 事件 | Timer E CMP3 事件 |
| hrtim_flt6[4:1] | Timer F 重置/翻转 | Timer F CMP3 事件 | Timer F CMP4 事件 | Timer F CMP3 事件 |

故障计数器还允许忽略多个伪故障事件并定义接受标准。

FLTxCNT[3:0]位域选择 FAULTx 计数器的阈值。当事件数量等于 FLTxCNT[3:0]的值时，故障被视为有效。

FLTxRSTM 选择 FAULTx 计数器的复位模式：

- 0：在复位/溢出事件时，故障计数器硬件复位，如下表所示。
- 1：仅在每个复位/溢出事件发生时，如果在上一个计数周期内没有事件发生，故障计数器才会复位，如下图所示。

故障计数器可以在任何时候通过软件使用 FLTxCRES 位进行复位。

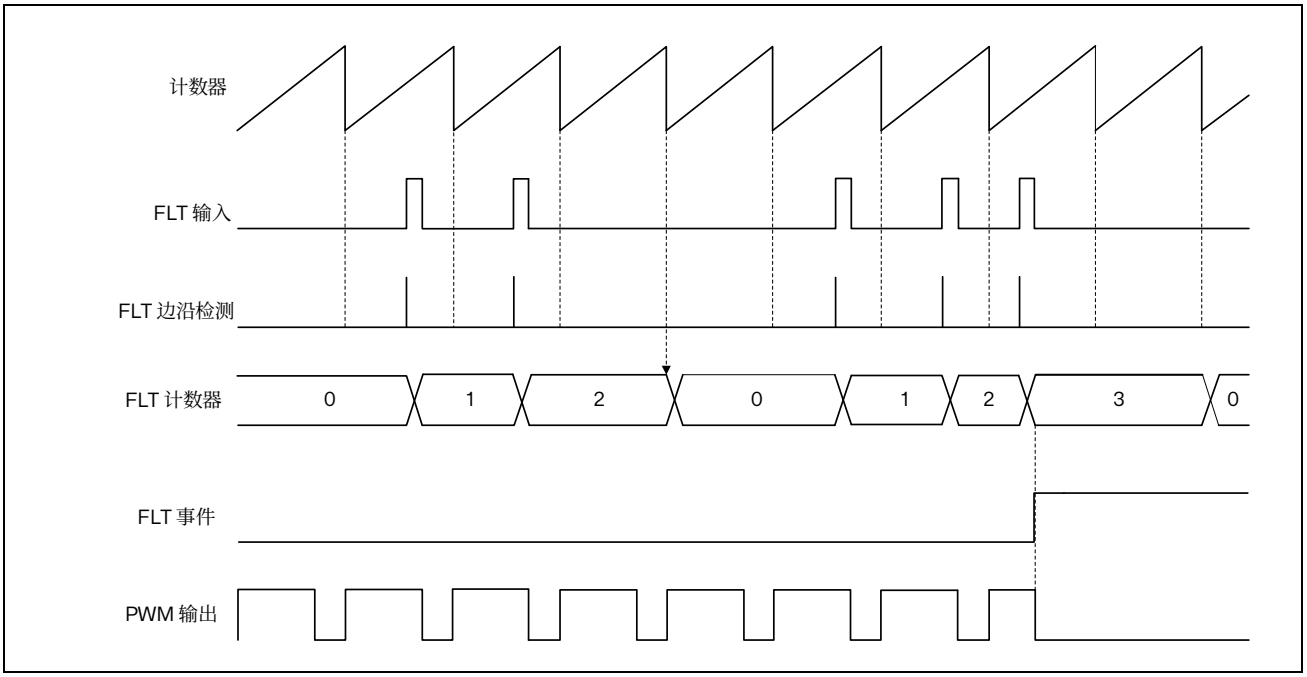



图 21.65 故障计数器累加模式（FLTxRSTM = 1，FLTxCNT[3:0] = 2）

给定的 FLTx 输入计数器可以由单个源进行复位。下表指示了与特定故障关联的定时器单元。这并不意味着故障线不能由多个定时器共享（例如，FLT1 带有使能的事件计数器，可以同时作用于定时器 A、定时器 B 和定时器 C）。

表 21.31 故障 1 至 6 计数器复位源

| 故障输入 | 故障计数复位源 |
|---|---|
| hrtim_flt1[4:1] | Timer A 复位/翻转 |
| hrtim_flt2[4:1] | Timer B 复位/翻转 |
| hrtim_flt3[4:1] | Timer C 复位/翻转 |
| hrtim_flt4[4:1] | Timer D 复位/翻转 |
| hrtim_flt5[4:1] | Timer E 复位/翻转 |
| hrtim_flt6[4:1] | Timer F 复位/翻转 |

**系统故障输入（hrtim_sys_flt）**

此故障由 MCU 的 B 类电路提供（有关详细信息，请参见系统配置控制器（SYSCFG）部分），对应于来自以下源的系统故障：

- 时钟安全系统
- SRAM 奇偶校验器
- 内核锁住信号
- PVD 检测器
- Flash ECC 双错误检测

此输入会覆盖 FAULT 输入，并禁用所有 FAULTy[1:0] = 01、10、11 的输出。

对于每条 FAULT 通道，可通过 HRTIM_FLTxR 寄存器中仅可写入一次的 FLTxLCK 位锁定 FLTxE、FLTxP、FLTxSRC、FLTxF[3:0]位（使其成为只读位），以实现功能安全。使能后，故障调节设置会冻结，直到下一次 HRTIM 复位或系统复位。

按上述方法对故障信号进行调节后，信号会发送到定时单元。对于任一定时单元，都会使用 HRTIM_FLTxR 寄存器中的 FLT1EN 到 FLT6EN 位使能 6 条故障通道，并且可同时选择 6 条通道（只要输出受故障机制保护，便会自动使能 sysfault）。这样便可实现：

- 一条故障通道同时禁止多个定时单元
- 多条故障通道进行或运算来禁止单个定时单元

HRTIM_FLTxR 寄存器中仅可写入一次的 FLTLCK 位可在下一次复位之前锁定 FLTxEN 位（使其成为只读位），以实现功能安全。使能后，定时单元故障相关设置会冻结，直到下一次 HRTIM 复位或系统复位。

对于每个定时器，故障期间的输出状态是通过 HRTIM_OUTxR 寄存器中的 FAULT1[1:0]和 FAULT2[1:0]位定义的。

### 21.3.18 辅助输出

定时器 A 和 E 所具有的辅助输出与进入输出级的常规输出并行工作。辅助输出可提供以下内部状态、事件和信号：

- SETxy 和 RSTxy 状态标志，以及相应的中断和 DMA 请求
- 输出置位/复位后的捕获触发事件
- Tx2 输出复制后的外部事件过滤信号

辅助输出会在突发模式控制器之前或之后获取，具体视 HRTIM 工作模式而定。下图给出了概览。

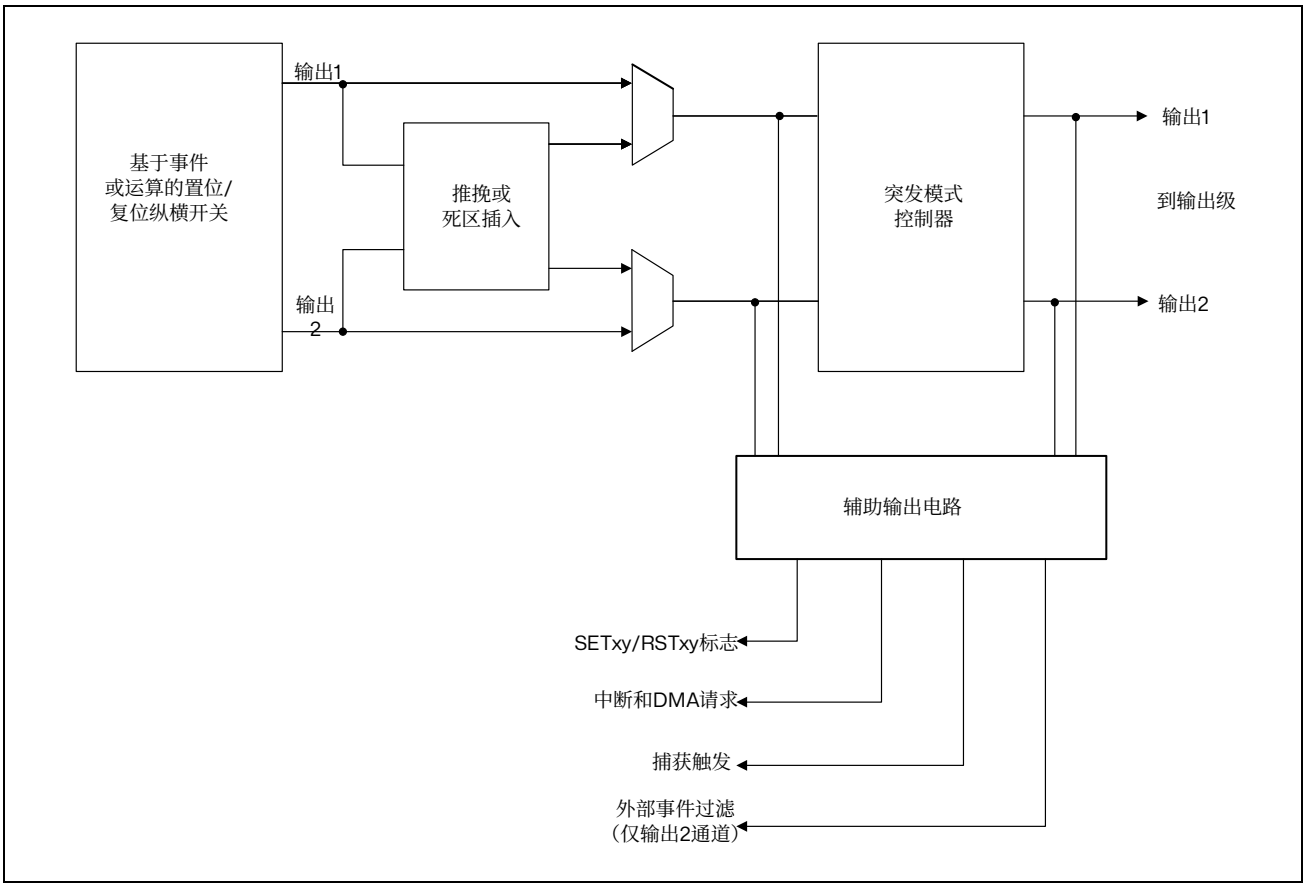



图 21.66 辅助输出

默认情况下，辅助输出是输出 Tx1 和 Tx2 的副本。例外情况包括：

- 禁止死区时（DTEN = 0）的延迟空闲和均衡空闲保护。触发保护后，辅助输出会保持不变，并会遵循纵横开关发出的信号。相反，如果使能了死区（DTEN = 1），主输出和辅助输出都会强制设为无效电平。
- 突发模式（TCEN=1，IDLEMx=1）；有两种情况：
  a) 如果 DTEN=0 或 DIDLx=0，辅助输出将不受突发模式进入的影响，而会继续遵循纵横开关发出的参考信号（请参见下图）。
  b) 如果同时使能了死区（DTEN=1）和延迟突发模式进入（DIDLx=1），则辅助输出与主输出的行为相同。它们会在死区持续时间结束后强制设为 IDLES 电平，随后会在所有突发周期内保持这一电平。突发模式终止后，会保持 IDLES 电平，直至跳变到相反电平（与主输出相似）。

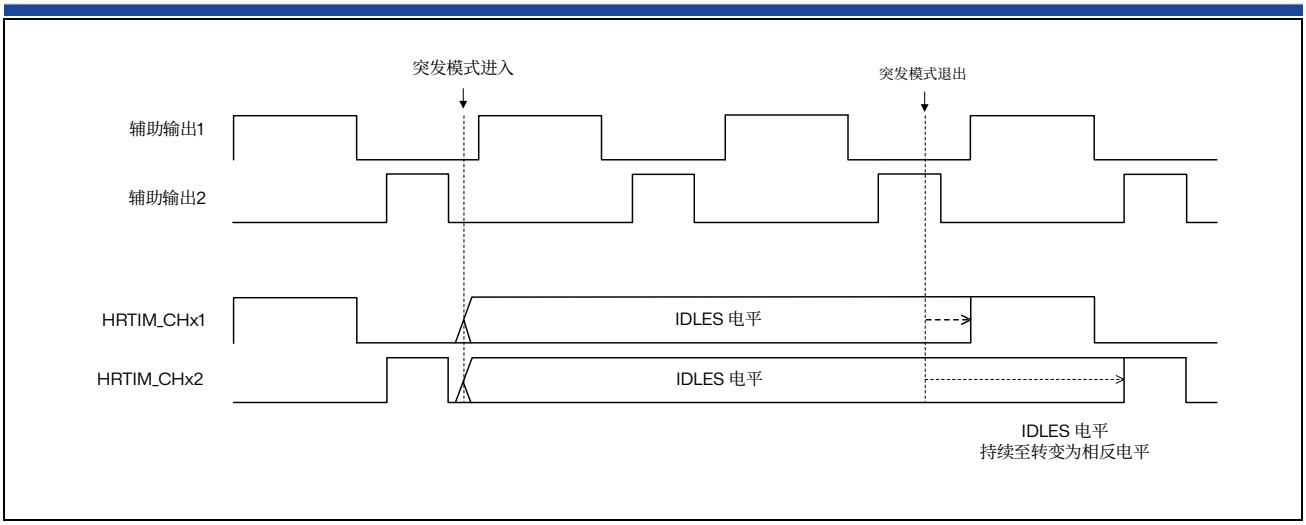



图 21.67 突发模式期间的辅助输出和主输出（DIDLx = 0）

如果在死区内发生从突发模式退出或者在延迟保护后重新使能输出，则辅助输出上的信号会略有失真。这种情况下，应用到辅助输出的死区会延长，以便能够符合主输出上的死区要求。下图给出了一些例子。

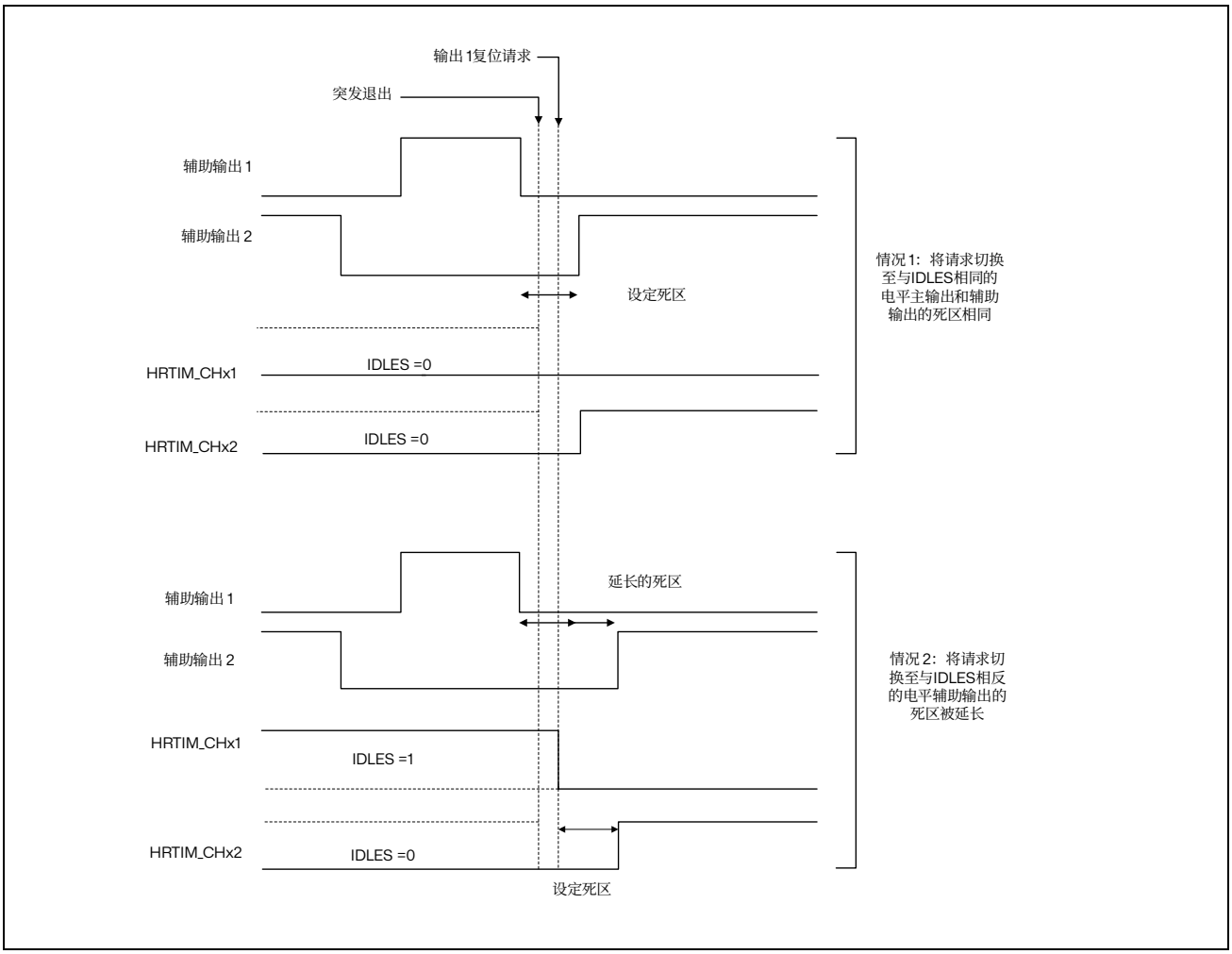



图 21.68 退出突发模式时辅助输出上的死区失真

### 21.3.19 将 HRTIM 与其他定时器或 HRTIM 实例同步

HRTIM 可作为主单元（生成同步信号）或从单元（等待触发被同步）来同步多个 HRTIM 实例。此功能也可用于将 HRTIM 与其他外部或片上定时器进行同步。同步电路在主定时器内进行控制。

#### 同步输出

本节介绍了必须对 HRTIM 进行何种配置才能同步外部资源并充当主定时器单元。

可选择四种事件作为要发送到同步输出的源，方法是使用 HRTIM_MCR 寄存器中的 SYNCSRC[1:0]位，具体如下：

- 00：主定时器启动
  当 MCEN 位置 1，或者定时器在单发模式下达到周期值后重新启动时，会生成该事件。
  如果计数期间发生复位（CONT 或 RETRIG 位置 1），也会生成该事件。
- 01：主定时器比较 1 事件
- 10：定时器 A 启动
  当 TACEN 位置 1，或者计数器复位并重新开始计数（响应此次复位）时，会生成该事件。以下计数器复位事件不会传播到同步输出：连续模式下的计数器翻转、单发不可再触发模式下被丢弃的复位请求。仅当计数期间发生复位时（CONT 或 RETRIG 位置 1），才会考虑复位。
- 11：定时器 A 比较 1 事件

HRTIM_MCR 寄存器中的 SYNCOUT[1:0]位指定了同步事件的生成方式。

如果 SYNCOUT[1:0] = 1x，则会在 HRTIM_SCOUT 输出引脚上生成同步脉冲。SYNCOUT[0]位指定了同步信号的极性。如果 SYNCOUT[0] = 0，HRTIM_SCOUT 引脚具有低空闲电平，并会发出长度为 16 个 $f_{HRTIM}$ 时钟周期的正脉冲进行同步。如果 SYNCOUT[0] = 1，空闲电平为高电平，并会生成负脉冲。

*注：同步脉冲后会有 16 个 $f_{HRTIM}$ 时钟周期的空闲电平，在此期间，会丢弃任何新的同步请求。因此，最大同步频率为 $f_{HRTIM}/32$。*

SYNCOUT[1:0]位使能后（也就是位域值不是 00 后），会立即在 HRTIM_SCOUT 引脚施加空闲电平。

必须在配置 MCU 输出和计数器使能之前执行同步输出初始化步骤，具体执行步骤如下：

1. 配置 HRTIM_MCR 中的 SYNCOUT[1:0]和 SYNCSRC[1:0]位域
2. 配置 HRTIM_SCOUT 引脚（请参见通用 I/O 部分）
3. 使能主定时器或定时器 A 计数器（MCEN 或 TACEN 位置 1）

使用同步输入模式启动计数器（使用 SYNCSTRTM/SYNCSTRTx 位）时，使用同步输出且同步源选择主定时器启动或者 TIMA 启动/复位事件（SYNCSRC[1:0] = 00 或 10），仅当计数器在启动或运行过程中复位时，才会生成输出脉冲。如果复位请求将计数器清零而不启动计数器，则不会影响同步输出。

### 同步输入

HRTIM 可通过外部源进行同步，具体通过对 HRTIM_MCR 寄存器中的 SYNCIN[1:0] 位进行编程来实现：

- 00：禁止同步输入
- 01：保留配置
- 10：片上定时器 TRGO 输出连接到 hrtim_in_sync[2]输入。
- 11：HRTIM_SCIN 输入引脚（hrtim_in_sync[3]）上的正脉冲

目标定时器（主定时器或定时单元）使能后（MCEN 和/或 TxCEN 位置 1），不能更改此位域。

HRTIM_SCIN 输入为上升沿有效。定时器行为是通过 HRTIM_MCR 和 HRTIM_TIMxCR 寄存器中的下列位定义的（有关详细信息，请参见下表）：

- 同步启动：传入信号会启动定时器的计数器（SYNCSTRTM 和/或 SYNCSTRTx 位置 1）。TxCEN（MCEN） 位必须置 1 才能使能定时器并使计数器准备好启动。在连续模式下，计数器在收到同步信号之前不会启动。
- 同步复位：传入信号会复位计数器（SYNCRSTM 和/或 SYNCRSTx 位置 1）。该事件会像其他任何复位事件一样使重复计数器递减。

仅当相关计数器使能后（MCEN 或 TxCEN 位置 1），才考虑同步事件。同步请求会触发 SYNC 中断。

*注：如果当前计数器值大于有效周期值，同步启动事件会复位计数器。*

同步事件的作用取决于定时器工作模式，请参见下表中总结的内容。

表 21.32 同步事件的作用与定时器工作模式之间的关系

| 工作模式 | SYNC RSTx | SYNC STRTx | SYNC 复位或启动事件后的行为 |
|---|---|---|---|
| 单次不可再触发 | 0 | 1 | 当计数器停止并且满足以下条件时，会考虑启动事件：<br>- 当 MCEN 或 TxCEN 位被置 1 时<br>- 已达到周期时间。<br>如果在计数器因达到周期值而停止时发生启动事件，则会将计数器复位。复位请求会将计数器清零，但不会启动计数器（计数器仅可通过同步事件重启）。任何在计数期间发生的复位事件都会被忽略（与在常规不可再触发模式下时一样）。 |
| | 1 | X | 复位事件将启动定时器计数操作。仅当计数器停止并且满足以下条件时，才会考虑复位事件：<br>- 当 MCEN 或 TxCEN 位被置 1 时<br>- 已达到周期时间。<br>如果选择了多个复位请求（来自 HRTIM_SCIN 和内部事件），则仅会考虑第一个到达的请求。 |

| 工作模式 | SYNC RSTx | SYNC STRTx | SYNC 复位或启动事件后的行为 |
|---|---|---|---|
| 单发可再触发 | 0 | 1 | 仅当计数器未启动或周期结束时，计数器启动才有效。任何在计数器启动后发生的同步事件都无效。<br>如果在计数器因达到周期值而停止时发生启动事件，则会将计数器复位。复位请求会将计数器清零，但不会启动计数器（计数器仅可通过同步事件启动）。计数过程中发生的复位事件会被考虑（与在常规可再触发模式下时一样）。 |
| | 1 | X | HRTIM_SCIN 发出的复位请求会被当作来自内部事件的 HRTIM 计数器复位请求进行考虑，并将启动或重新启动定时器计数操作。如果选择了多个复位请求，则会考虑第一个到达的请求。 |
| 连续模式 | 0 | 1 | 定时器已使能（MCEN 或 TxCEN 位置 1），并正在等待同步事件启动计数器。计数器启动后发生的任何同步事件均不起作用（计数器仅可通过同步事件启动）。复位请求会将计数器清零，但不会启动计数器。 |
| | 1 | X | HRTIM_SCIN 发出的复位请求会被当作来自内部事件的 HRTIM 计数器复位请求进行考虑，并将启动或重新启动定时器计数操作。如果选择了多个复位请求，则会考虑第一个到达的请求。 |

*当同步复位事件与周期事件发生在同一个 $f_{HRTIM}$ 时钟周期内时，该复位操作会被推迟到下一个编程设定的周期事件（因为这两个事件都会导致计数器翻转）。这仅在使能高分辨率模式时适用（即* CKPSC[2:0] < 5*）。*

下图显示了单发模式下如何进行同步启动。

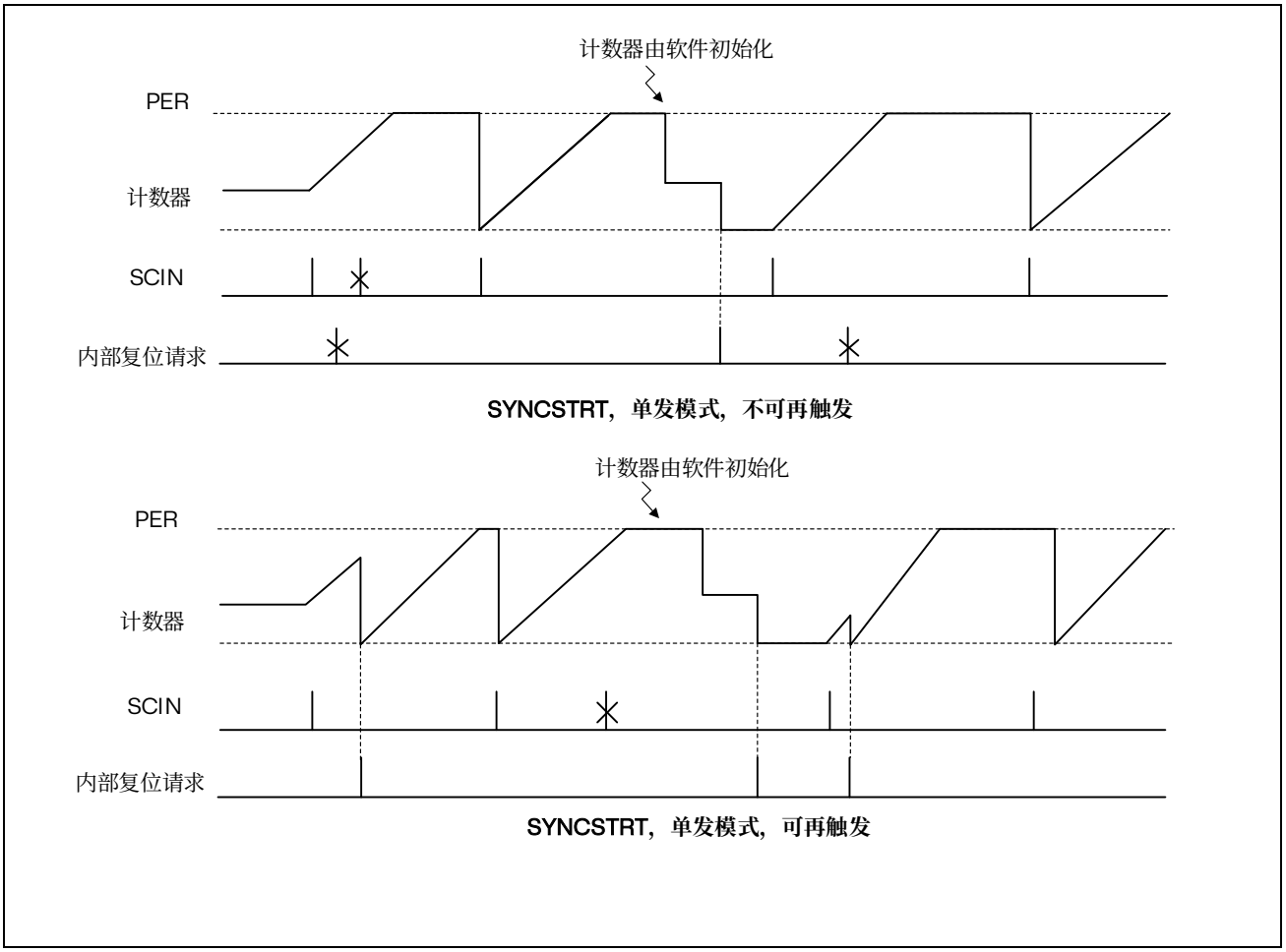



图 21.69 同步启动模式下计数器的行为

### 21.3.20 ADC 触发

ADC 可以由主定时器和 6 个定时单元触发。

ADC 的规则组通道和注入组通道各有 10 个独立的触发源。外部事件可用作触发信号。这类触发信号会在 HRTIM_EECRx 寄存器中定义的调节结束后立即获取，并且与 EEFxR1 和 EEFxR2 寄存器设置无关。

在 HRTIM_ADC1R 到 HRTIM_ADC4R 寄存器中，最多可以将 32 个事件组合（逻辑或运算）用于 ADC 触发 1 到 4，如下图所示。ADC 触发 1/3 和 2/4 使用一组相同的源。可通过同时选择多个源的方式在单个开关周期内进行多次触发。典型用例是用于非重叠多相转换器，可通过单个 ADC 触发输出对所有相位进行连续采样。

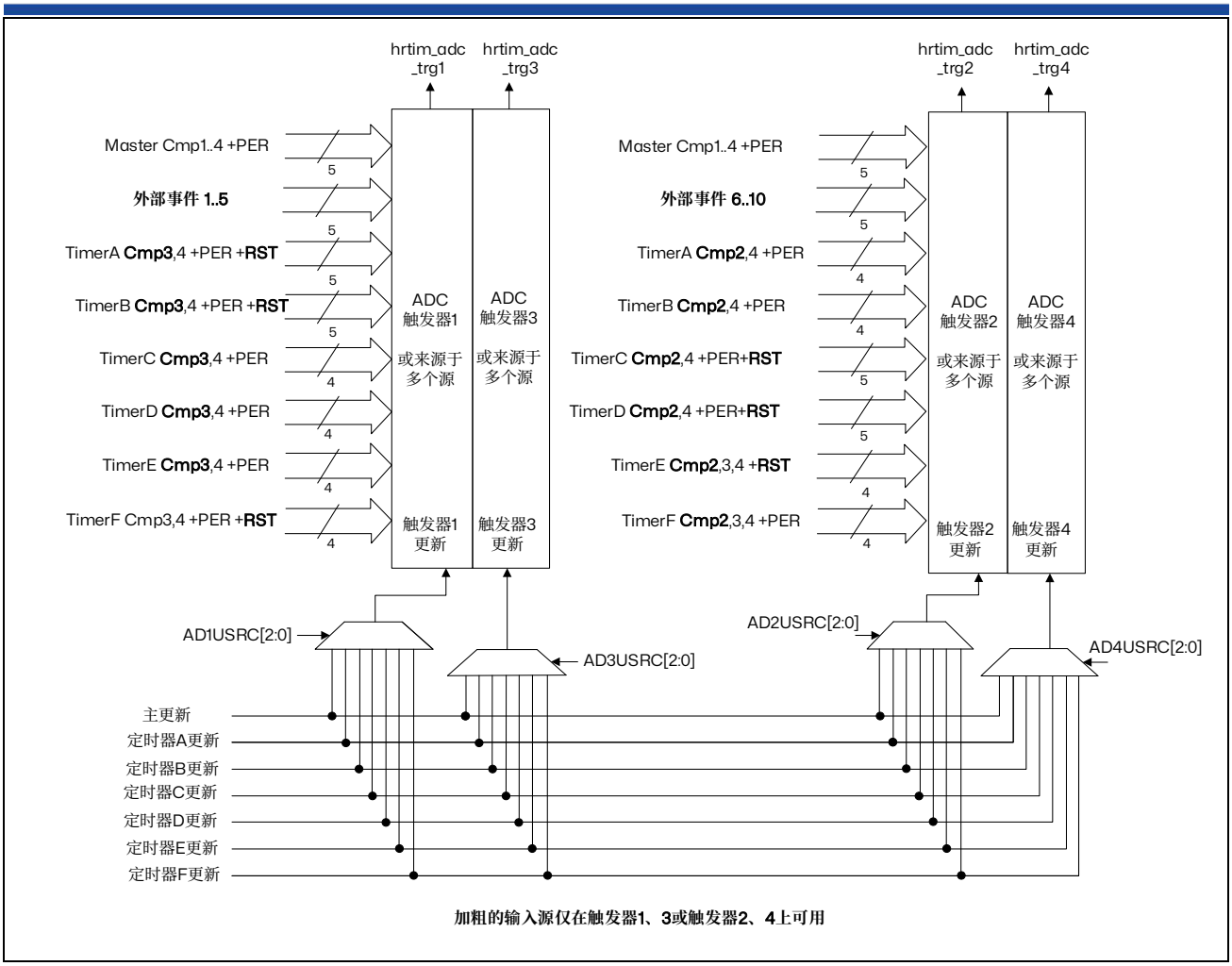



图 21.70 ADC 触发选择概述

ADC 触发 5 到 10 在 HRTIM_ADCER 寄存器中进行配置，如下图所示。ADC 触发 5/7/9 和 6/8/10 使用一组相同的源。

对于这些触发，一次只能选择一个源（从 32 个可能的事件中选择一个）。

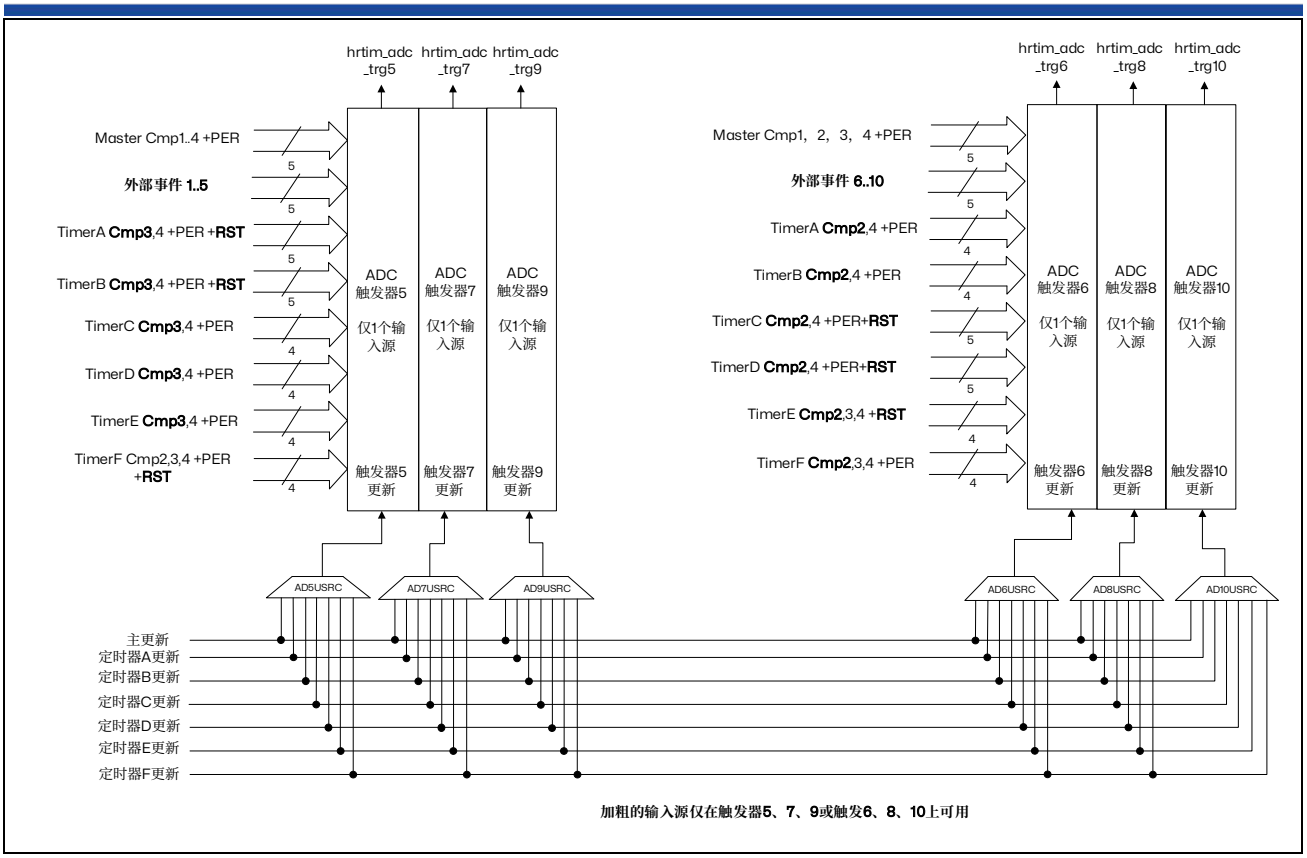



图 21.71 ADC 触发

HRTIM_ADC1R 至 HRTIM_ADC4R 和 HRTIM_ADCER 寄存器会预装载，并可与相关定时器同步更新。更新源是通过 HRTIM_CR1 和 HRTIM_ADCUR 寄存器中的 ADxUSRC[2:0]位进行定义。

例如，如果 ADC 触发 1 输出定时器 A 的 CMP2 事件（HRTIM_ADC1R = 0x0000 0400），则 HRTIM_ADC1R 通常与定时器 A（AD1USRC[2:0] = 001）同时更新。

如果源定时器中禁止预装载（PREEN 位复位），HRTIM_ADCxR 寄存器也不会预装载：写访问会立即更新触发源。

### ADC 后缩放器

后缩放单元允许降低 ADC 触发率，如下图所示。

每个 ADC 触发率都可以使用 HRTIM_ADCxPS1 和 HRTIM_ADCxPS2 寄存器中的 ADCxPSC[4:0]位进行单独调整。

在中心对齐模式下，ADC 触发率还依赖于源定时器中编程的 ADROM[1:0]位域，如图 21.73 所示。ADROM[1:0]位域用于编码可以触发 ADC 的任何事件：复位、翻转（周期）和比较事件：

- ADROM[1:0] = 00：在向上和向下计数阶段都生成事件
- ADROM[1:0] = 01：在向下计数阶段生成事件
- ADROM[1:0] = 10：在向上计数阶段生成事件

ADC 后缩放器的编程寄存器是预加载的，并且可以在不停止定时器的情况下实时更新。

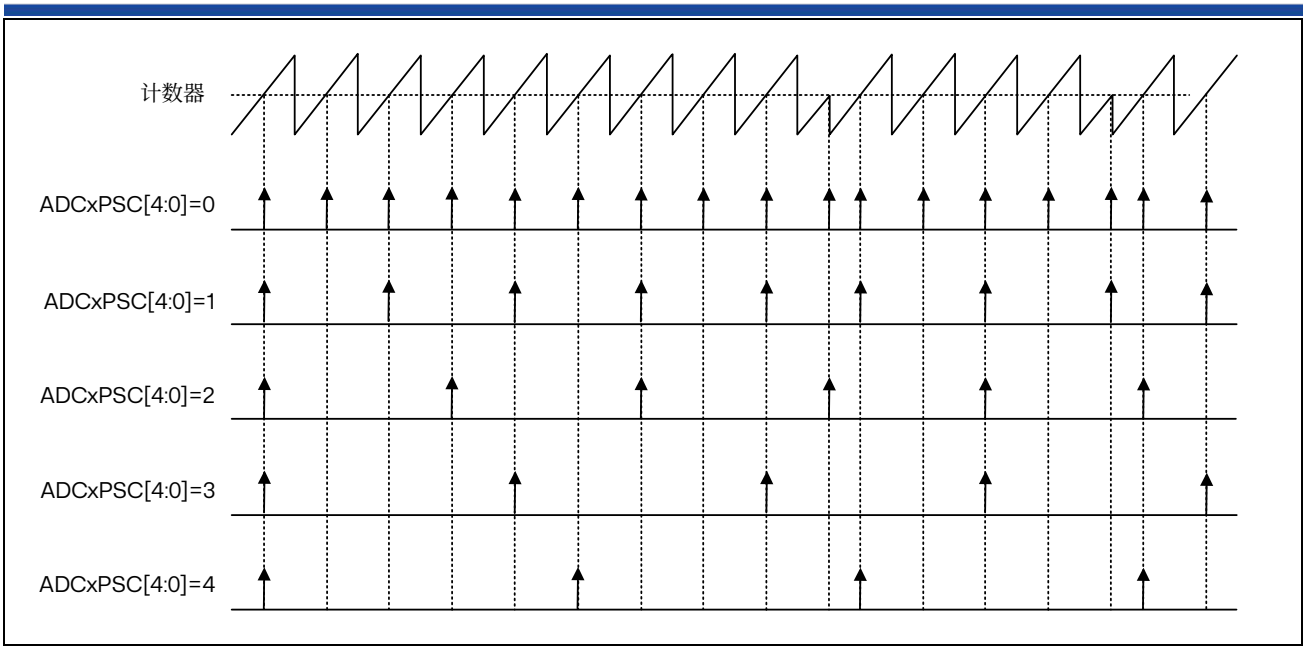



图 21.72 向上计数模式下的 ADC 触发后缩放

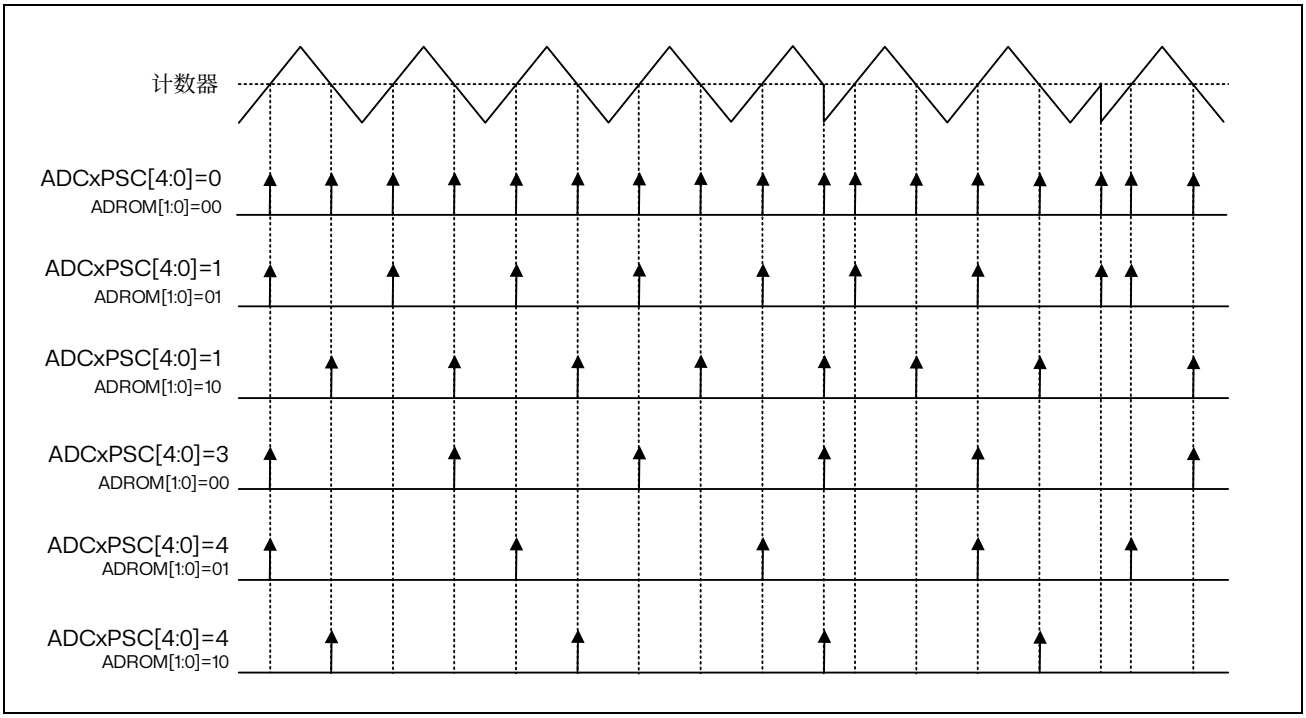



图 21.73 向上/向下计数模式下的 ADC 触发后缩放

### 21.3.21 DAC 触发

HRTIM 可使嵌入式 DAC 与定时器同步更新。

主定时器和定时器单元发出的更新事件可在 3 路 hrtim_dac_trgx 输出的任意一路上生成 DAC 更新触发信号。

*注：每个定时器都有自己的DAC 相关控制寄存器*

HRTIM_MCR 和 HRTIM_TIMxCR 寄存器 DACSYNC[1:0]位的编程值含义如下：

- 00：未生成更新
- 01：在 hrtim_dac_trg1 上生成更新
- 10：在 hrtim_dac_trg2 上生成更新
- 11：在 hrtim_dac_trg3 上生成更新

会在 hrtim_dac_trgx 输出上生成 1 个 $f_{HRTIM}$ 时钟周期的输出脉冲。

如果多个定时器中使能了 DACSYNC[1:0]位，hrtim_dac_trgx 输出将由所有定时器更新事件的或运算结果决定。例如，如果定时器 A 和定时器 B 中 DACSYNC = 1，那么定时器 A 中的更新事件将与定时器 B 中的更新事件进行或运算，在相应的 hrtim_dac_trgx 输出上生成 DAC 更新触发信号，如下图所示。

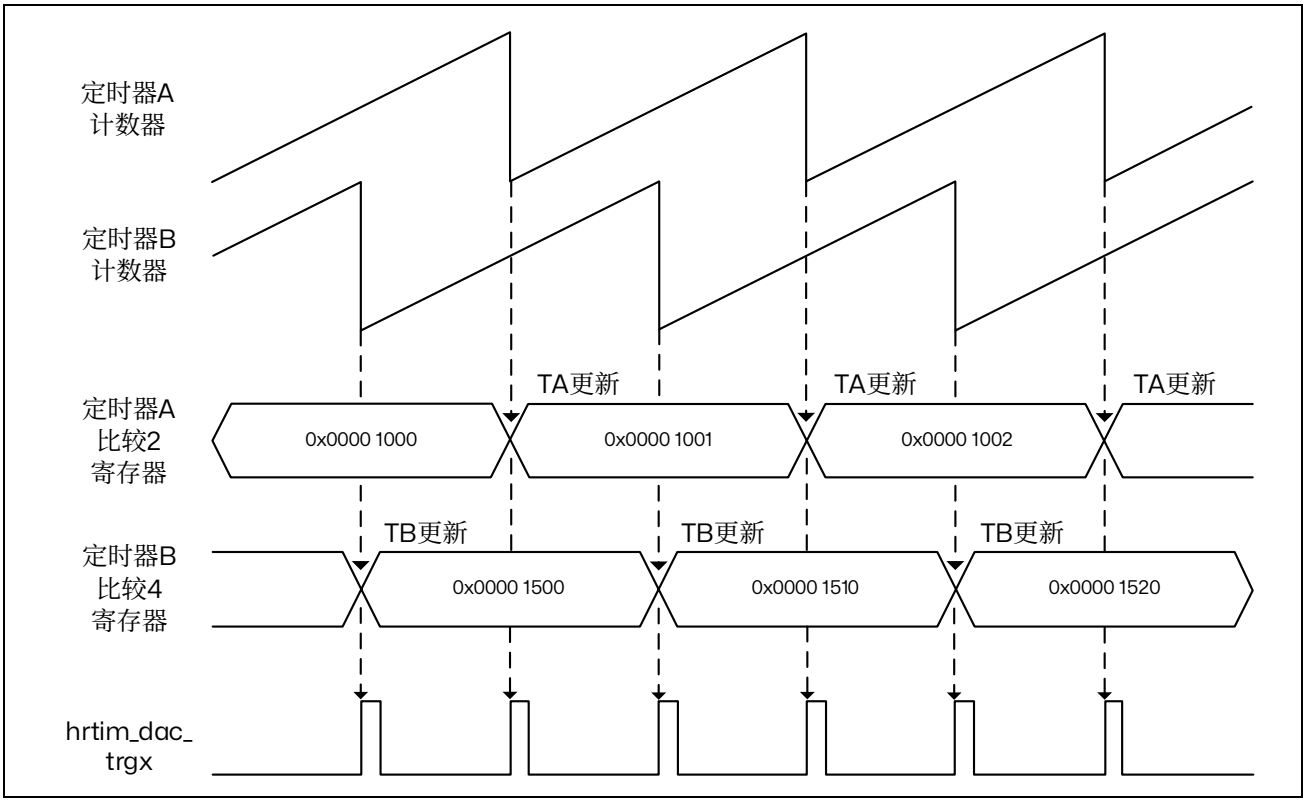



图 21.74 将多个更新合并到一路 hrtim_dac_trgx 输出

关于 DAC 的连接，请参考表 22.9：HRTIM DAC 触发器连接。

**双通道 DAC 触发**

斜率补偿技术和滞后控制可以使用 HRTIM 内置功能和 DAC 锯齿波发生器轻松实现。其原理是，让 DAC 生成一个与 PWM 周期同步的递减锯齿波，或者一个与 PWM 信号同步的方波。

这种模式通过 TIMxCR2 寄存器中的 DCDE 位来使能。一旦定时器开始运行（TxEN 位设置），该位就不能更改。

它使用两个触发输出，如下图所示：

– hrtim_dac_reset_trgx 生成 DAC 复位/更新事件
– hrtim_dac_step_trgx 生成 DAC 值增量变化的请求

TIMxCR2 寄存器中的 DCDR 位定义了 hrtim_dac_reset_trgx 触发器的生成时机：

- DCDR = 0：触发器在计数器复位或翻转事件上生成
- DCDR = 1：触发器在输出 1 设置事件上生成

*注：当* DCDE *位复位（双通道* DAC *触发禁用）时，* DCDR *位没有意义。*

TIMxCR2 寄存器中的 DCDS 位定义了 hrtim_dac_step_trgx 触发器的生成时机：

- DCDS = 0：触发器在比较事件 2 上生成
- DCDS = 1：触发器在输出 1 复位事件上生成

DCDR 和 DCDS 位允许覆盖以下使用场景：

– 边沿对齐斜率补偿（DCDR = DCDS = 0）：DAC 的锯齿波在 PWM 周期开始时启动，并在周期内生成多个触发器
– 中心对齐斜率补偿（DCDR = 1, DCDS = 0）：DAC 的锯齿波在输出设置事件上启动，并在周期内生成多个触发器
– 滞后控制器：当输出状态改变时，每个周期内必须改变 DAC 值两次。每个 PWM 周期生成两个触发器。在边沿对齐模式（DCDR=0, DCDS=1）下，触发器在计数器复位或翻转时生成。在中心对齐模式（DCDR=1, DCDS=1）下，触发器在输出设置时生成。

当 DCDE 设置且 DCDS 位复位时，比较器 2 具有特定的工作模式。一旦发生比较匹配，有效比较值将自动更新，以便触发器可以周期性重复，周期等于 CMP2 值，如下图所示。

使用 DCDS 位复位的双通道 DAC 触发（使用比较事件 2）不应与使用 CMP2 的模式（三重/四重交错和半触发周期模式）同时使用。

*注：*CMP2 *值可以实时更改。新值将在下一个比较匹配时生效。*

*注：当* DCDS *位复位时，* CMP2 *值不应被其他机制修改：交错、半触发周期和平衡空闲模式必须禁用。*

下表给出了一个示例，用于在一个 PWM 周期内生成 6 个触发器。它表明有必要将除法结果向上取整。

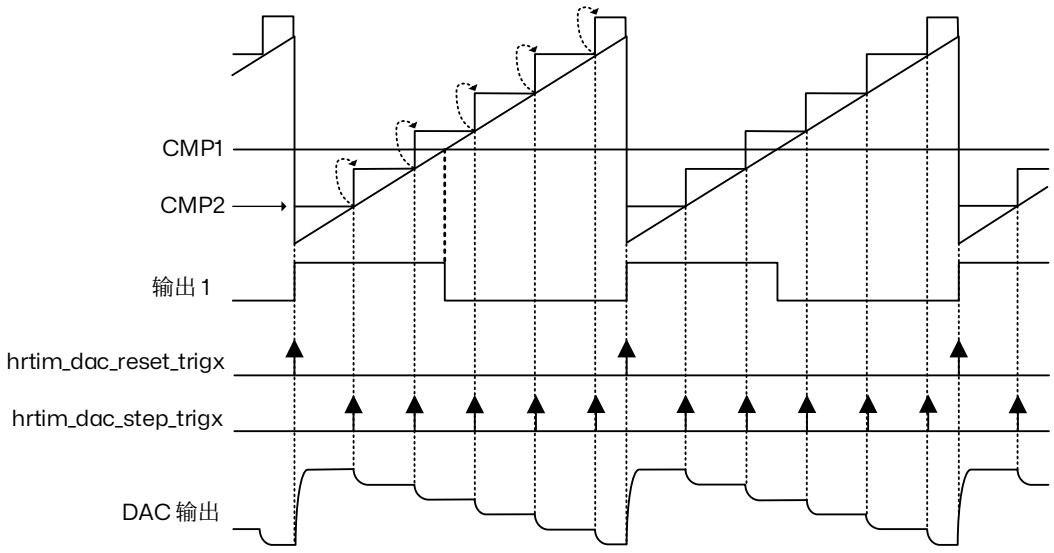
考虑一个计数器周期 TIMxPER = 8192。将 8192 除以 6 得到 1365.33。

- x 向下取整的值：1365：会生成 7 个触发器，第 6 个和第 7 个非常接近（分别在计数器=8190 和 8192 时）
- x 向上取整的值：1366：会生成 6 个触发器。第 6 个触发器 dac_step_trg（在计数器=8192 时）会因计数器从 8192 翻转到 0 而被中止。

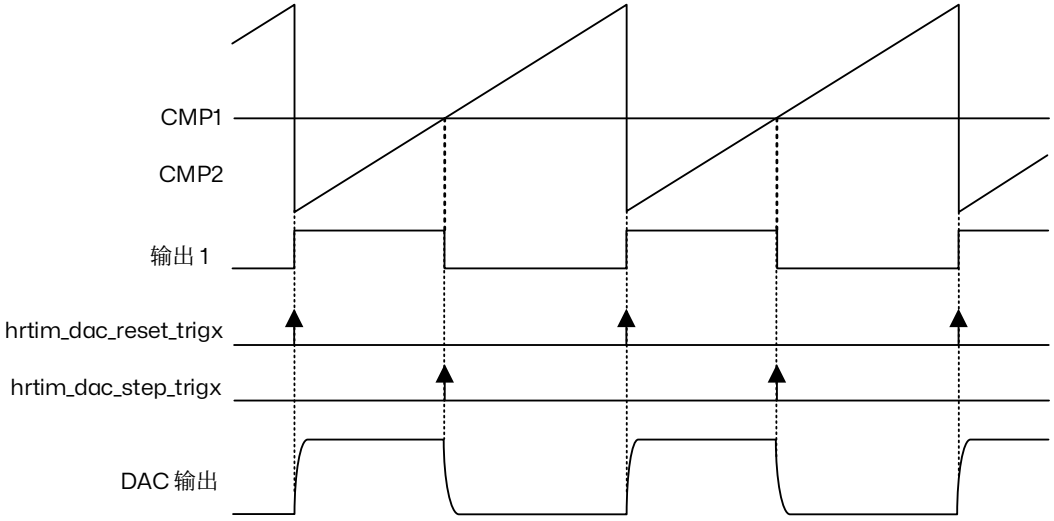
表 21.33 DAC 双通道触发示例

| - | CMP2 = 1365 | dac_trg | dac_step_trg | CMP2 = 1366 | dac_trg | dac_step_trg |
|---|---|---|---|---|---|---|
| 计数值 | 1365 | - | 1 | 1366 | - | 1 |
| | 2730 | - | 2 | 2732 | - | 2 |
| | 4095 | - | 3 | 4098 | - | 3 |
| | 5460 | - | 4 | 5464 | - | 4 |
| | 6825 | - | 5 | 6830 | - | 5 |
| | 8190 | - | 6 | 8192 | 6 | - |
| | 8192 | 7 | - | 1366 | - | 1 |
| | 1365 | - | 1 | 2732 | - | 2 |
| | ... | - | - | ... | - | - |

*注：在中心对齐模式下，每个切换周期必须有偶数个触发器，以避免在计数器峰值附近出现不均匀间隔的触发器。*




DCDR = 0（触发器在计数器复位或翻转事件上生成），DCDS = 0（触发器在比较事件2上生成）




DCDR = 0（触发器在输出1设置事件上生成），DCDS = 0（触发器在输出1复位事件上生成）

图 21.75 DAC 斜率补偿触发

下图提供了所有可用 DAC 触发器的概述。

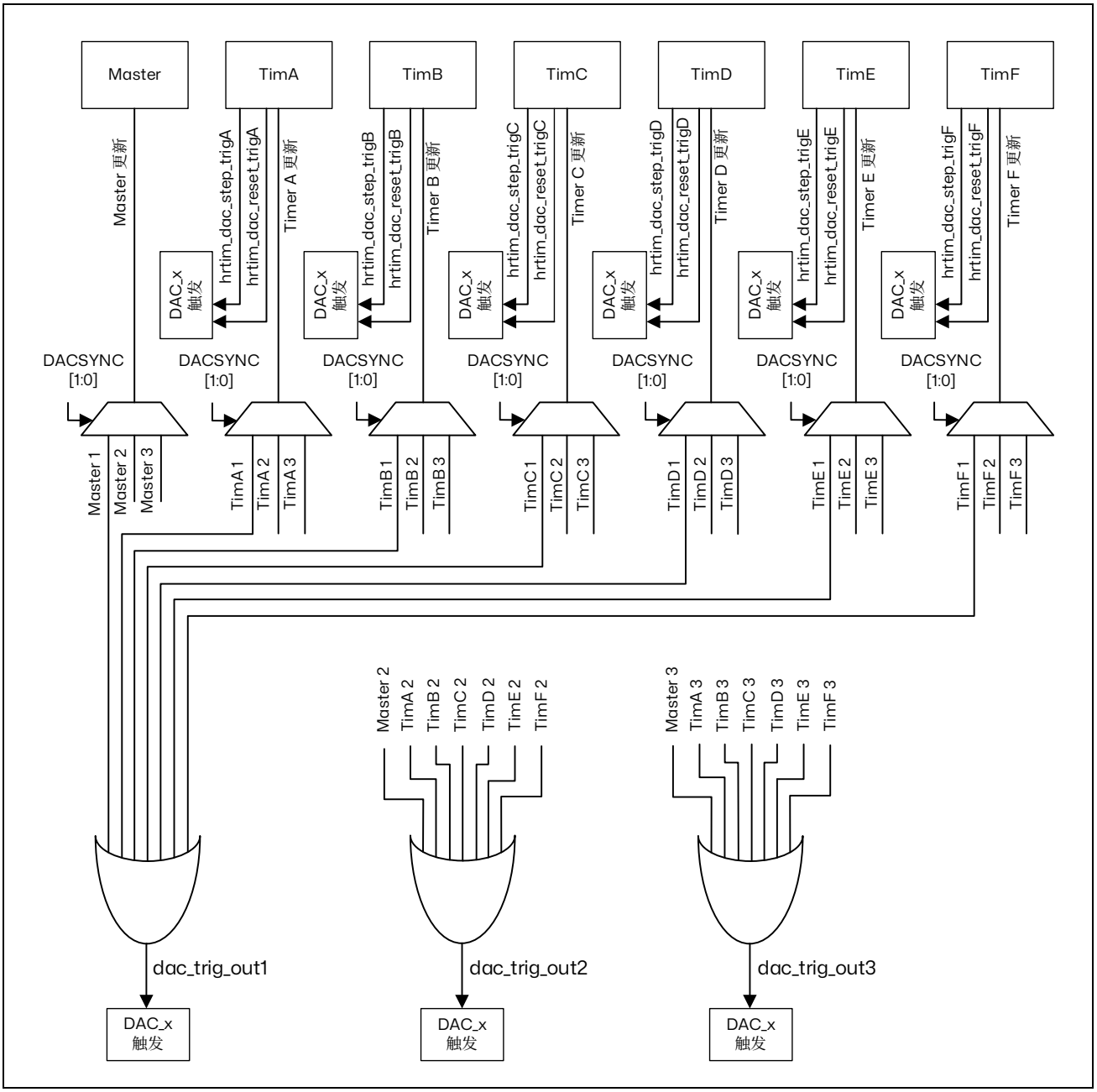



图 21.76 DAC 触发器概述

### 21.3.22 中断

主定时器可以生成 7 个中断:

- 主定时器寄存器更新
- 接收到同步事件
- 主定时器重复事件
- 主定时器比较 1 到 4 事件

每个定时单元可以生成 14 个中断：

- 延迟保护触发
- 计数器复位或翻转事件
- 输出 1 和输出 2 复位（从有效电平跳变为无效电平）
- 输出 1 和输出 2 置位（从无效电平跳变为有效电平）
- 捕获 1 和 2 事件
- 定时单元寄存器更新
- 重复事件
- 比较 1 到 4 事件

会为整个 HRTIM 生成 8 个全局中断：

- 系统故障和故障 1 到 6（不考虑定时单元的因素）
- DLL 校准完成
- 突发模式周期结束

中断请求会分到 8 个向量组中，具体如下：

- hrtim_it1：主定时器中断（主定时器更新、同步输入、重复、MCMP1..4）和全局中断（故障除外，如突发模式周期和 DLL 就绪中断）
- hrtim_it2：TIMA 中断
- hrtim_it3：TIMB 中断
- hrtim_it4：TIMC 中断
- hrtim_it5：TIMD 中断
- hrtim_it6：TIME 中断
- hrtim_it7：TIMF 中断
- hrtim_it8：专用的故障中断向量，用于允许高优先级中断处理

下表总结了中断请求及其映射以及关联的控制和状态位。

表 21.34 HRTIM 中断汇总

| 中断向量 | 中断事件 | 事件标志位 | 使能控制位 | 标志清零位 |
|---|---|---|---|---|
| hrtim_it1 | 突发模式周期结束 | BMPER | BMPERIE | BMPERC |
| | DLL 校准完成 | DLLRDY | DLLRDYIE | DLLRDYC |
| | 主定时器寄存器更新 | MUPD | MUPDIE | MUPDC |
| | 接收到同步事件 | SYNC | SYNCIE | SYNCC |
| | 主定时器重复事件 | MREP | MREPIE | MREPC |
| | 主定时器比较 1 到 4 事件 | MCMP1 | MCMP1IE | MCP1C |
| | | MCMP2 | MCMP2IE | MCP2C |
| | | MCMP3 | MCMP3IE | MCP3C |
| | | MCMP4 | MCMP4IE | MCP4C |
| hrtim_it2 | 延迟保护已触发 | DLYPRT | DLYPRTIE | DLYPRTC |
| hrtim_it3 | 计数器复位或翻转事件 | RST | RSTIE | RSTC |

| 中断向量 | 中断事件 | 事件标志位 | 使能控制位 | 标志清零位 |
|---|---|---|---|---|
| hrtim_it4<br>hrtim_it5<br>hrtim_it6<br>hrtim_it7 | 输出 1 和输出 2 复位（从有效电平跳变为无效电平） | RSTx1 | RSTx1IE | RSTx1C |
| | | RSTx2 | RSTx2IE | RSTx2C |
| | 输出 1 和输出 2 置位（从无效电平跳变为有效电平） | SETx1 | SETx1IE | SETx1C |
| | | SETx2 | SETx2IE | SETx2C |
| | 捕获 1 和 2 事件 | CPT1 | CPT1IE | CPT1C |
| | | CPT2 | CPT2IE | CPT2C |
| | 定时单元寄存器更新 | UPD | UPDIE | UPDC |
| | 重复事件 | REP | REPIE | REPC |
| | 比较 1 到 4 事件 | CMP1 | CMP1IE | CMP1C |
| | | CMP2 | CMP2IE | CMP2C |
| | | CMP3 | CMP3IE | CMP3C |
| | | CMP4 | CMP4IE | CMP4C |
| hrtim_it8 | 系统故障 | SYSFLT | SYSFLTIE | SYSFLTC |
| | 故障 1 到 6 | FLT1 | FLT1IE | FLT1C |
| | | FLT2 | FLT2IE | FLT2C |
| | | FLT3 | FLT3IE | FLT3C |
| | | FLT4 | FLT4IE | FLT4C |
| | | FLT5 | FLT5IE | FLT5C |
| | | FLT6 | FLT6IE | FLT6C |

### 21.3.23 DMA

大部分能够生成中断的事件也可以生成 DMA 请求，甚至可以同时生成中断和 DMA 请求。每个定时器（主定时器、TIMA...F）都有自己的 DMA 使能寄存器。

各 DMA 请求会在进行或运算后发送到 7 条通道中，具体如下：

- 主定时器 1 条通道
- 每个定时单元（TIMA...F）各 1 条通道

*注：禁止* DMA *通道之前（*TIMxDIER *中的* DMA *使能位复位），需要先禁止* DMA *控制器。*

下表总结了事件及其关联的 DMA 使能位。

表 21.35 HRTIM DMA 请求总结

| DMA 通道 | 事件 | DMA 功能 | DMA 使能位 |
|---|---|---|---|
| hrtim_dma1 (主定时器) | 突发模式周期结束 | 无 | N/A |
| | DLL 校准完成 | 无 | N/A |
| | 主定时器寄存器更新 | 有 | MUPDDE |
| | 接收到同步事件 | 有 | SYNCDE |
| | 主定时器重复事件 | 有 | MREPDE |
| | 主定时器比较 1 到 4 事件 | 有 | MCMP1DE |
| | | 有 | MCMP2DE |
| | | 有 | MCMP3DE |
| | | 有 | MCMP4DE |
| hrtim_dma2 (定时器 A) hrtim_dma3 (定时器 B) hrtim_dma4 (定时器 C) hrtim_dma5 (定时器 D) hrtim_dma6 (定时器 E) hrtim_dma7 (定时器 F) | 延迟保护已触发 | 有 | DLYPRTDE |
| | 计数器复位或翻转事件 | 有 | RSTDE |
| | 输出 1 和输出 2 复位（从有效电平跳变为无效电平） | 有 | RSTx1DE |
| | | | RSTx2DE |
| | 输出 1 和输出 2 置位（从无效电平跳变为有效电平） | 有 | SETx1DE |
| | | | SETx2DE |
| | 捕获 1 和 2 事件 | 有 | CPT1DE |
| | | | CPT2DE |
| | 定时单元寄存器更新 | 有 | UPDDE |
| | 重复事件 | 有 | REPDE |
| | 比较器 1 到 4 事件 | 有 | CMP1DE |
| | | 有 | CMP2DE |
| | | 有 | CMP3DE |
| | | 有 | CMP4DE |
| N/A | 系统故障 | 无 | N/A |
| | 故障 1 至 6 | 无 | N/A |
| | 突发模式周期结束 | 无 | N/A |
| | DLL 校准完成 | 无 | N/A |

### 突发 DMA 传输

除了标准 DMA 请求之外，HRTIM 还配有 DMA 突发控制器，可通过单个 DMA 请求更新多 个寄存器。具体包括：

- 仅通过一条 DMA 通道更新多个数据寄存器，
- 如果转换器使用多路定时器输出，可动态地对一个或多个定时单元重新进行编程。

突发 DMA 功能仅可用于一条 DMA 通道，但可选择 6 条通道中的任意一条进行突发 DMA 传输。

通过 DMA 写入，编程那些寄存器是核心内容。主定时器和 TIMA..E 包含突发 DMA 更新寄存器，其中的大部分控制和数据寄存器都关联到选择位：HRTIM_BDMUPR、HRTIM_BDTAUPR 到 HRTIM_BDTEUPR（这一点仅适用于寄存器的写访问）。重定向机制允许自动将 DMA 写访问转发到 HRTIM 寄存器，如下图所示。

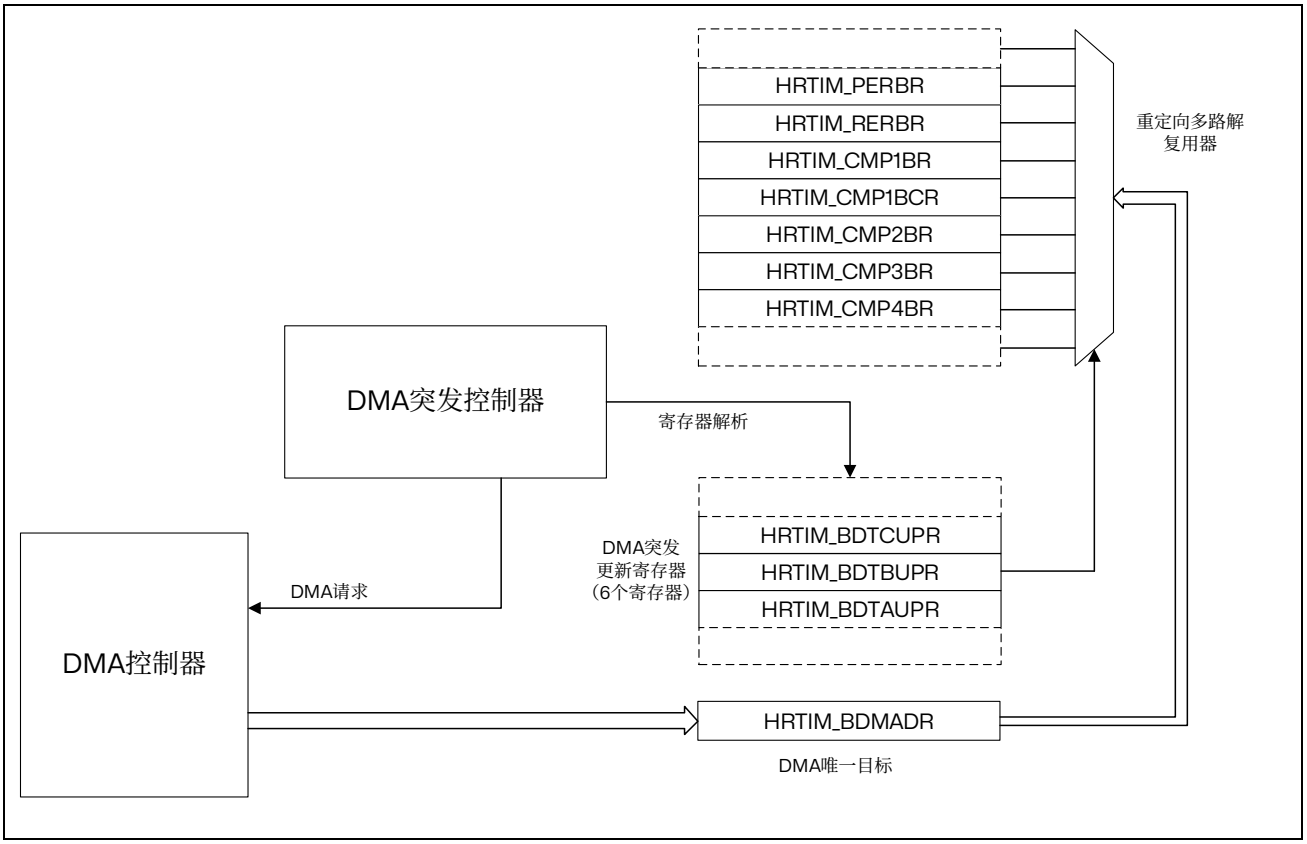



图 21.77 DMA 突发概览

发生 DMA 触发时，HRTIM 会生成多个 32 位 DMA 请求，并会解析更新寄存器。如果控制位置 1，写访问会重定向到关联的寄存器。如果此位复位，则会跳过寄存器更新并恢复寄存器解析，直至检测到此位再次置 1 触发新请求。对 6 个更新寄存器（HRTIM_BDMUPR、5x HRTIM_BDTxUPR）进行解析后，突发结束，系统准备好处理另一 DMA 触发（请参见下图中的流程图）。

*注：任何在突发正在进行时发生的触发都会被丢弃，但最后一次数据传输期间发生的触发除外。*

突发 DMA 模式永久使能（没有使能位）。突发 DMA 工作是通过对 HRTIM_BDMADR 寄存器进行的第一次写访问启动的。

仅需要使 DMA 控制器指向作为目标的 HRTIM_BDMADR 寄存器，在存储器中指向禁止了外设递增模式的外设配置（HRTIM 会在内部处理到最终目标寄存器的数据重发）。

如果突发 DMA 模式在事务进行过程中中断，要重新初始化突发 DMA 模式，至少需要写入 6 个更新寄存器中的其中一个。

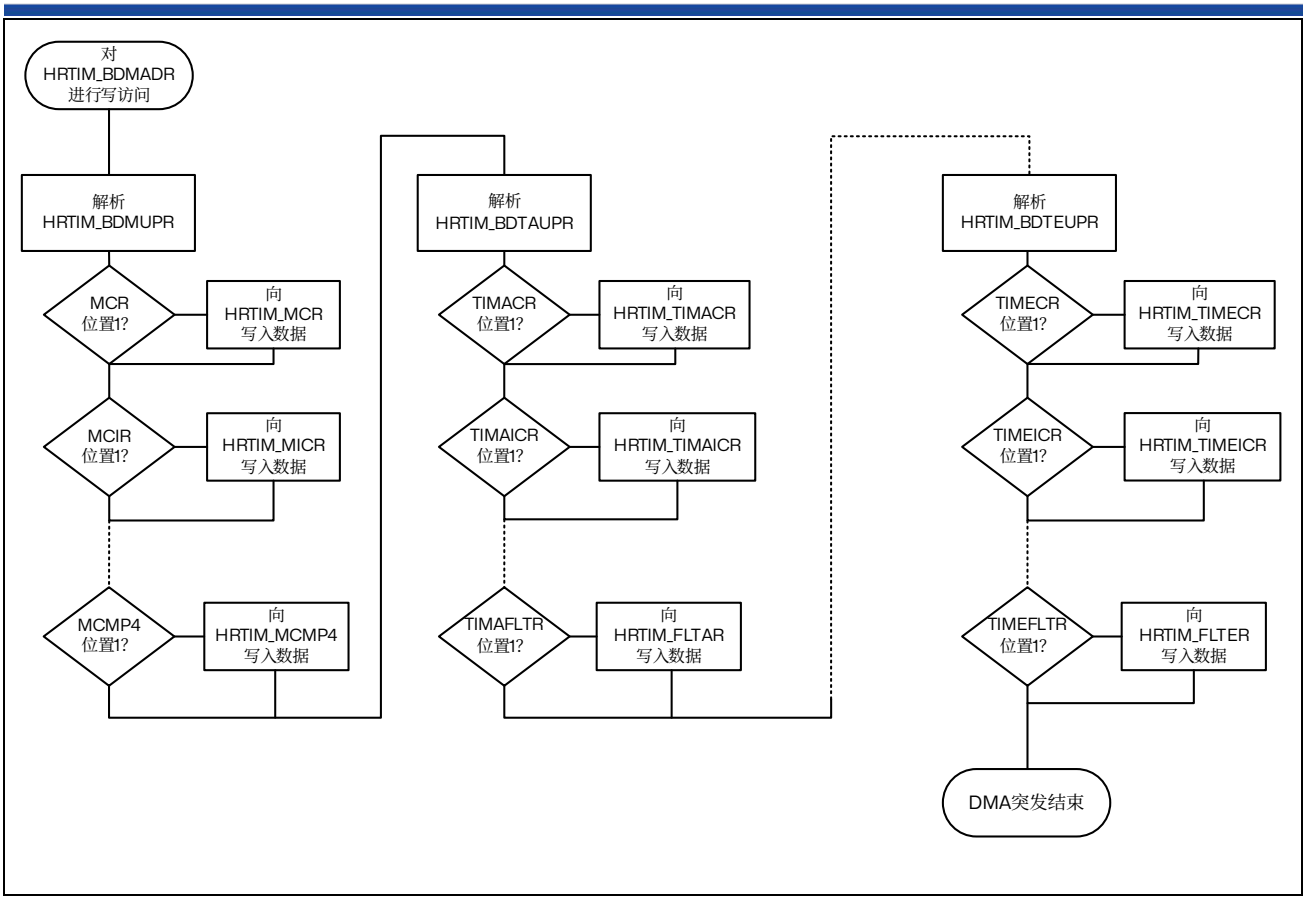



图 21.78 突发 DMA 工作流程图

DMA 突发结束后，有多种选项可供使用，具体视寄存器更新策略而定。

如果 PREEN 位复位（禁止预装载），通过 DMA 写入的值会立即传输到活动寄存器中，并会按照 DMA 事务的速度连续更新寄存器。

当预装载使能时（PREEN 位置 1），有 3 个用例：

1. 更新独立于 DMA 突发传输进行（HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0000，HRTIM_MCR 中的 BRSTDMA[1:0] = 00）。在这种情况下，如果需要同时考虑所有传输的数据，用户必须检查 DMA 突发是否在发生更新事件之前结束。相反，如果更新事件在 DMA 传输进行时发生，则仅会装载部分寄存器，完整的寄存器更新将需要 2 个连续的更新事件。
2. 更新在 DMA 突发传输完成后进行（HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0000，HRTIM_MCR 中的 BRSTDMA[1:0] = 01）。此模式可确保同时传输所有新寄存器值。此操作独立于计数器值进行，如有必要，可与常规更新事件相结合（例如，TxRSTU 置 1 的情况下，在发生计数器复位事件时进行更新）。
3. 在更新在 DMA 突发传输完成后发生更新事件时进行（HRTIM_TIMxCR 中的 UPDGAT[3:0] = 0010，HRTIM_MCR 中的 BRSTDMA[1:0] = 10）。此模式可确保对所有已传输数据、以及与常规更新事件的同步、与定时器计数器的同步进行一致的更新。在这种情况下，如果正在进行传输时发生常规更新请求，该请求会被丢弃，并会在下一次发出更新请求时进行有效更新。

下图上的时序图显示了 3 种情况下的活动寄存器内容：PREEN=0、UPDGAT[3:0] = 0001 以及 UPDGAT[3:0] = 0001（PREEN = 1 时）。

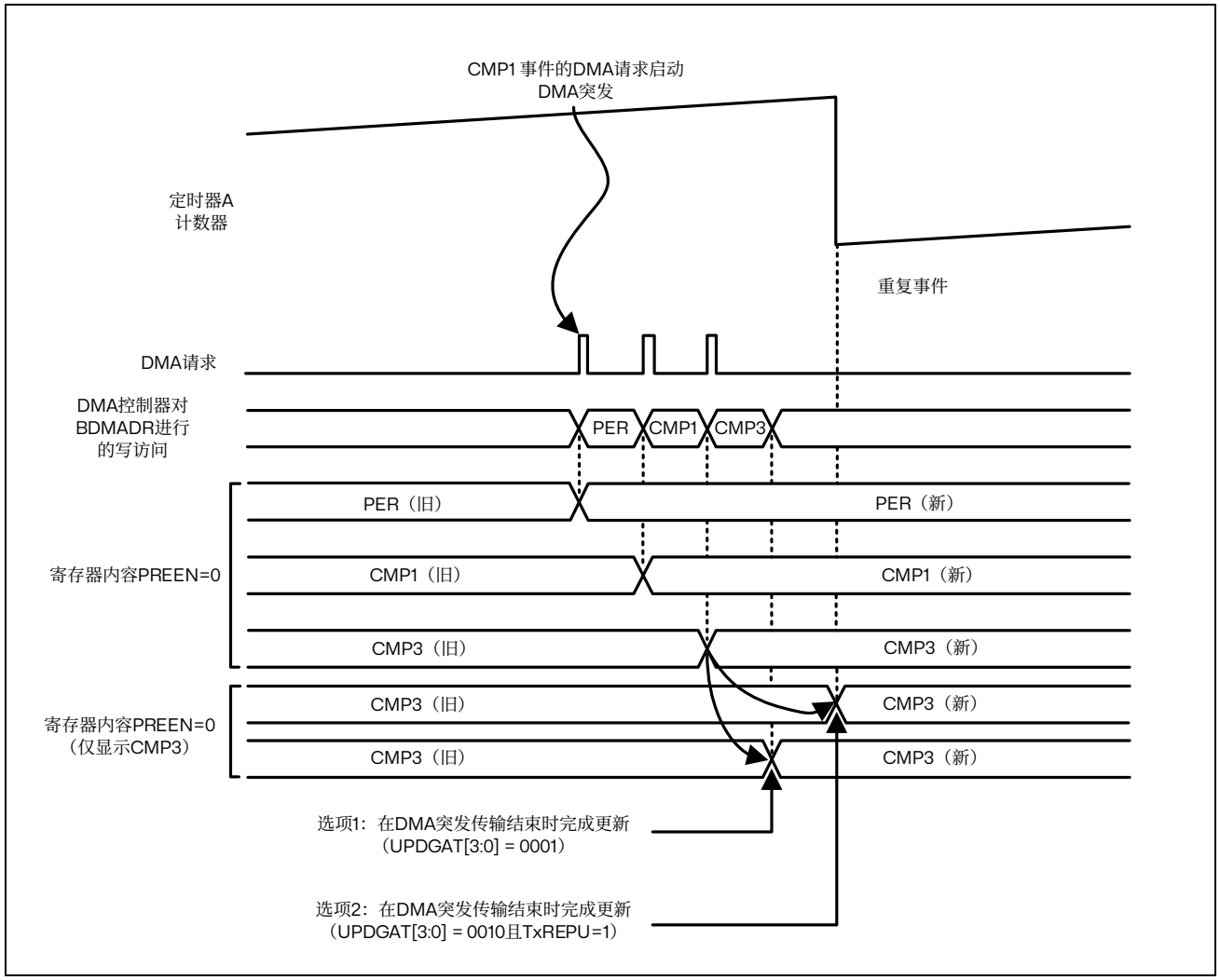



图 21.79 DMA 突发传输后进行寄存器更新

### 21.3.24 HRTIM 初始化

本节介绍了建议的 HRTIM 初始化程序，包括其他相关 MCU 外设。

必须在复位和时钟控制单元（RCC）中使能 HRTIM 时钟源，同时遵守 DLL 锁定所需的 $f_{HRTIM}$ 范围。
必须通过设置 HRTIM_DLLCR 寄存器中的 CAL 位来启动 DLL 校准。
在不同 $f_{HRTIM}$ 范围时 HRTIM 时钟预分频器 CKPSC 初始化配置如下：

- 当 $f_{HRTIM}$ 小于 100MHz 时，CKPSC 只能配置为 101b。
- 当 $f_{HRTIM}$ 为 100~192MHz 时，按照以下步骤配置 CKPSC：
  1. 禁用所有 HRTIM 计数器使能位（MCEN，TxCEN）和输出使能位（TxyOEN）；
  2. 配置 CKPSC 为 101b（HRTIM_MCR 寄存器和 HRTIM_TIMxCR 寄存器的 CKPSC[2:0]）；
  3. 使能 DLL 旁路，配置 HRTIM_OPT 寄存器的 DLLBYPASS 位为 1；
  4. 根据 $f_{HRTIM}$ 配置 HRTIM_DLLCR 寄存器的 CKS 位，如果 $f_{HRTIM}$ 为 80~140MHz，则将 CLK 置 0，如果 $f_{HRTIM}$ 为 140~192MHz，则将 CKS 置 1；
  5. 使能 DLL 校准功能，配置 HRTIM_DLLCR 寄存器的 CALEN 为 1；

6. 启动 DLL 校准，将 HRTIM_DLLCR 寄存器的 CAL 置 1；
7. 等待 HRTIM_ISR 寄存器的 DLLRDY 置起，后可清零；
8. 配置 CKPSC 为自定义倍频；
9. 禁止 DLL 旁路，配置 HRTIM_OPT 寄存器的 DLLBYPASS 位为 0；

只有高分辨率单元准备就绪后，才能启动 HRTIM 主单元和定时单元。这由 DLLRDY 标志位指示。可以在恢复初始化之前轮询 DLLRDY 标志位，或者在初始化 HRTIM 或其他 MCU 外设的其他寄存器时，在后台运行校准。在这种情况下，必须在启动计数器之前检查 DLLRDY 标志位（如果需要，可以通过在 HRTIM_IER 中使能 DLLRDYIE 标志来发出校准结束中断）。一旦 DLL 校准完成，必须设置 CALEN 位以使其定期执行，并补偿潜在的电压和温度漂移。校准周期性通过 HRTIM_DLLCR 寄存器中的 CALRTE[1:0]位域定义。

HRTIM 控制寄存器可按照电源转换器拓扑和定时单元用例进行初始化。必须配置所有输入（源、极性、边沿有效性）。

最后必须设置 HRTIM 输出，设置顺序如下：

- 必须使用 HRTIM_OUTxR 中的 POLx 位定义极性
- 必须使用 HRTIM_OUTxR 中的 FAULTx[1:0]和 IDLESx 位配置 FAULT 和 IDLE 状态

HRTIM 输出已准备好连接到 MCU 的 I/O。在 GPIO 控制器中，必须按照产品数据手册中的复用功能映射表对所选 HRTIM I/O 进行配置。

从此刻起，HRTIM 会控制处于 IDLE 状态的输出。

要将输出配置为 RUN 模式，应将 HRTIM_OENR 寄存器中的 TxyOEN 位置 1。RUN 模式下发生第一个有效置位/复位事件之前，2 路输出都处于无效状态。只要 TxCEN 位复位，任何输出置位/复位事件（使用 SST、SRT 的软件请求除外）都会被忽略，突发模式请求也是如此（会忽略 IDLEM 位值）。

同样，任何来自突发模式控制器的计数器复位请求也会被忽略 （如果 TxBM 位置 1）。

*注：如果使能了死区插入（DTEN 位置 1），则需要通过软件强制设置输出状态（使用 SST 和 RS 位），使输出在进入 RUN 模式后立即处于互补状态。*

最终可通过将 HRTIM_MCR 中的 TxCEN 或 MCEN 位置 1 的方式启动 HRTIM 工作。

如果 HRTIM 外设通过复位和时钟控制器复位，HRTIM 输出会进入 IDLE 模式且为低电平。建议先断开 HRTIMER 与输出的连接（使用 GPIO 控制器），然后再执行外设复位。

### 21.3.25 Debug 调试

当微控制器进入调试模式（内核停止）时，TIMx 计数器会根据 DBG 模块中的 DBG_HRTIM_STOP 配置位选择继续正常工作或者停止工作。

- DBG_HRTIM_STOP = 0：无行为变化，HRTIM 继续工作。
- DBG_HRTIM_STOP = 1：所有 HRTIM 定时器（包括主定时器）均停止工作。
  如果 FAULTx[1:0] = 01、10、11，RUN 模式下的输出会进入 FAULT 状态，如果 FAULTx[1:0] = 00，输出会保持当前状态。处于空闲状态的输出会保持此状态。即使 MCU 退出停止模式，也会永久保持此状态。这样可在执行步进期间保持安全状态。可通过将 TxyOEN 位置 1 再次使能输出（需要使用调试器）。

#### MCU 停止工作期间的定时器行为（DBG_HRTIM_STOP = 1 时）

置位/复位纵横开关、死区和推挽单元、空闲/均衡故障检测以及在 RUN 模式下驱动正常输出的所有逻辑均不受调试的影响。输出将在内部维持栓所状态，以便在 TxyOEN 再次置 1 时（MCU 停止工作期间或停

止工作之后）重新获取输出的常规信号。如果输出已禁止，关联的触发信号和滤波器也会跟随内部波形。

MCU 停止工作期间，FAULT 输入和事件（任何源）会使能。

如果此时发生故障，则在 MCU 停止工作期间，故障状态位会置 1，TxyOEN 位会复位（TxyOEN 和 TxyODS 不受 DBG_HRTIM_STOP 位状态的影响）。

在调试模式下，会丢弃同步、计数器复位、启动、复位-启动事件以及捕获事件。这样做是为了确保所有相关寄存器在 MCU 停止期间的稳定性。

计数器到达断点时会停止计数。但计数器使能信号不会置位；因此退出调试模式时不会发出启动事件。所有计数器复位和捕获触发都会被禁用，外部事件也会被禁用（只要 MCU 停止工作，就会忽略这类事件）。输出 SET 和 RST 标志会冻结，但强制软件置位/复位的情况除外。调试过程中会屏蔽电平有效事件，但退出调试模式时，电平有效事件会立即再次激活。对于边沿有效事件，如果 MCU 停止工作期间信号保持有效，退出调试模式时不会生成新边沿。

更新事件会被丢弃。这样可避免触发对 hrtim_upd_en[3:1]输入的更新。DMA 触发会被禁止。突发模式电路冻结：触发会被忽略，突发模式计数器停止工作。

虽然 MCU 处于停止状态，但 DLL 校准不会被阻塞（DLLRDY 标志可以被设置）。

## 21.4 应用案例

### 21.4.1 降压转换器

降压转换器通常用于降压。HRTIM 最多能够以 7 个独立的开关频率控制 12 个降压转换器。
转换器通常以固定频率工作，Vin/Vout 比取决于应用到电源开关的占空比 D：

$$V_{out} = D \times V_{in}$$

下图给出了降压转换器的拓扑结构，以及连接到 ADC 进行电压读取的接口。

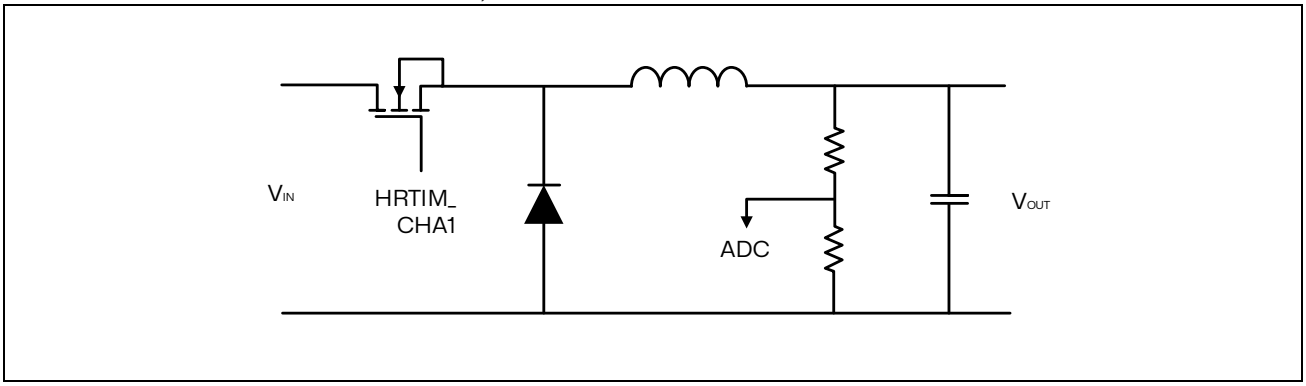



图 21.80 降压转换器拓扑结构

下图显示了如何利用相同频率的 PWM 信号来管理两个转换器。输出的定义如下：

- HRTIM_CHA1 在周期开始时置位，在发生 CMP2 时复位
- HRTIM_CHA2 在发生 CMP3 时置位，在发生 PER 时复位

ADC 每个周期会触发两次，分别通过 CMP1 和 CMP4 事件在 ON 时间的中点精确触发。

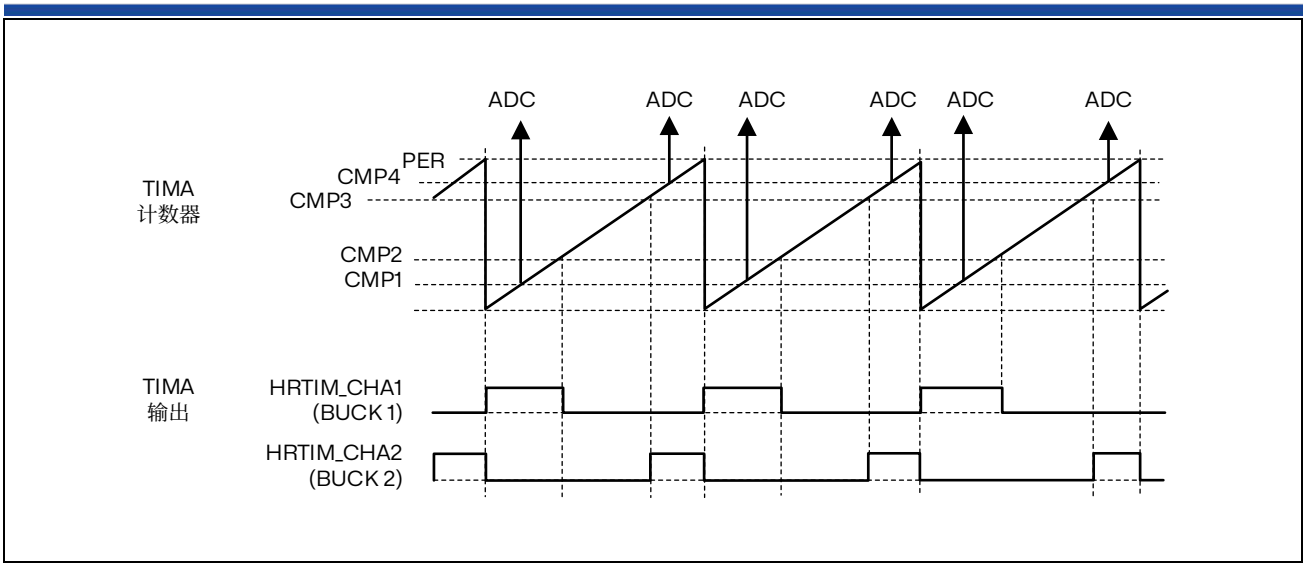



图 21.81 双降压转换器管理

定时器 A..E 提供 12 个成对耦合的降压转换器（每对转换器的开关频率均相同）或 7 个完全独立的转换器（每个转换器都有不同的开关频率），第二种情况使用主定时器作为第 7 个时基。

### 21.4.2 具有同步整流功能的降压转换器

同步整流使用 FET 代替续流二极管，从而最大限度地降低降压转换器中的损耗。同步整流可实时开启或关闭，具体视输出电流水平而定（如下图所示）。

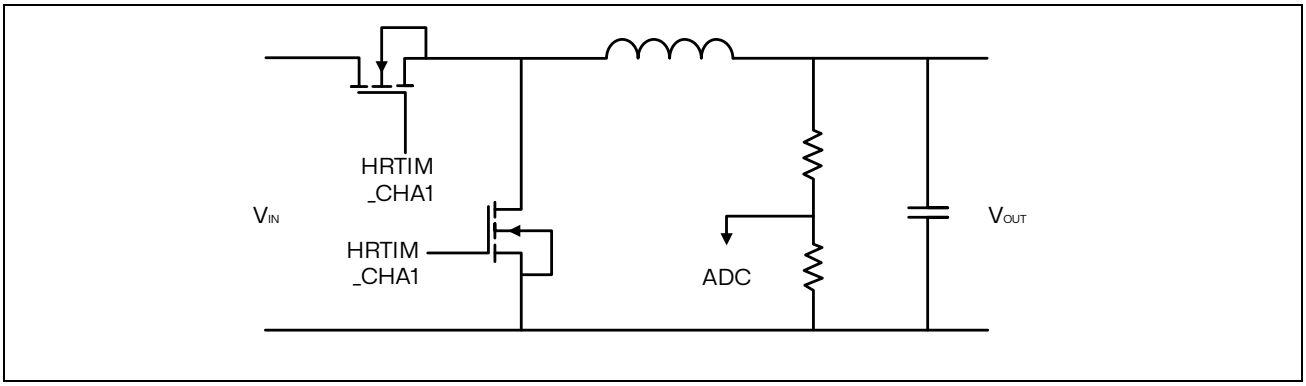



图 21.82 同步整流取决于输出电流

与单开关降压转换器的主要区别是增加了死区，从而可根据 HRTIM_CHA1 上的参考波形在 HRTIM_CHA2 上生成几乎互补的波形（如下图所示）。

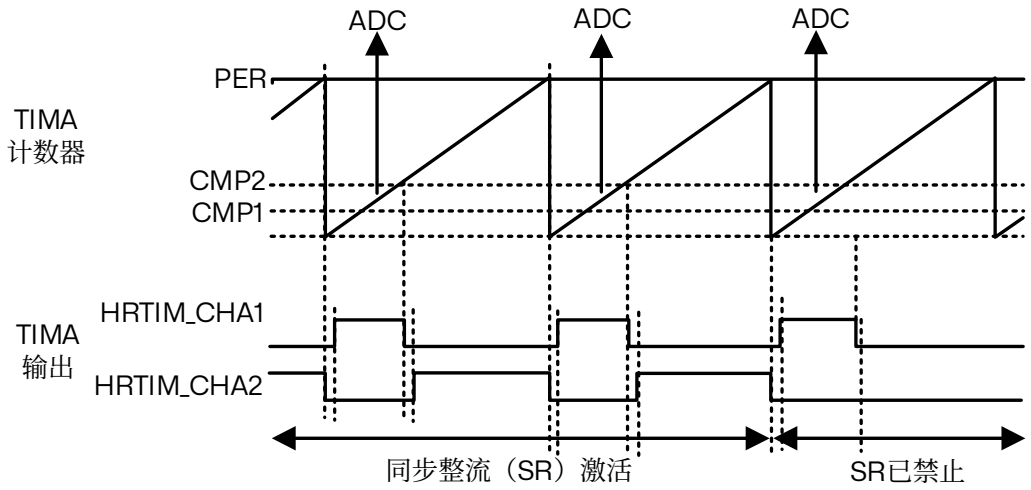



图 21.83 采用同步整流功能实现降压

### 21.4.3 多相转换器

多相技术可应用于多种电源转换拓扑（降压、反激）。此项技术的主要优势为：

- 减少输入和输出电容的电流波纹
- 减少电磁干扰（EMI）
- 通过动态更改相位数（相位分离）提高轻载条件下的效率

HRTIM 能够管理多个转换器。可控制的转换器数量取决于所使用的拓扑和资源（包括 ADC 触发信号）：

- 5 个具有同步整流（SR）功能的降压转换器，使用主定时器和 5 个定时器
- 4 个降压转换器（无 SR 功能），使用主定时器和 2 个定时器

下图显示了三相交错降压转换器的拓扑。

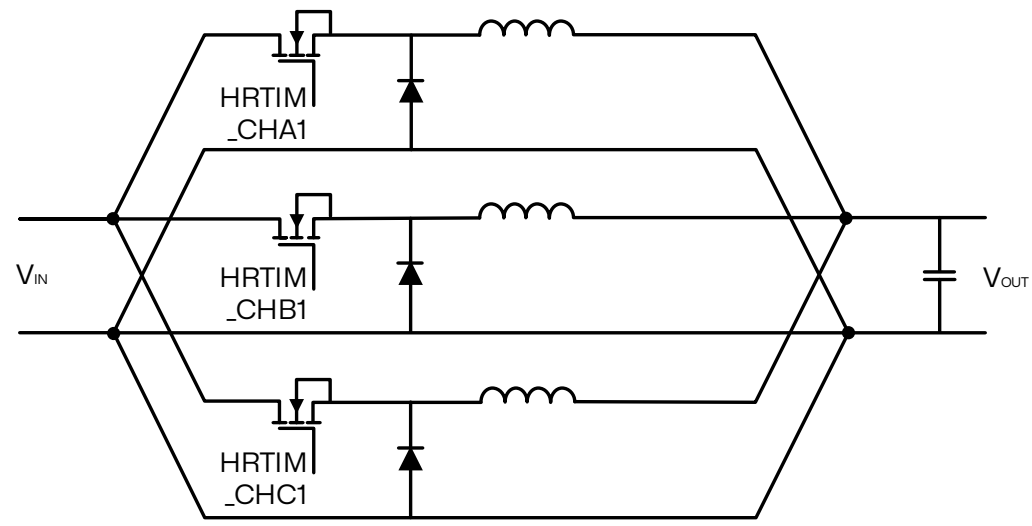



图 21.84 三相交错降压转换器

主定时器负责进行相位管理：它通过定期复位定时器来定义转换器之间的相位关系。相移为 360°除以相位数，所给示例中的相移为 120°。

占空比随后会编程到每个定时器中。输出的定义如下：

- HRTIM_CHA1 在主定时器周期开始时置位，在发生 TACMP1 时复位
- HRTIM_CHB1 在发生主定时器 MCMP1 时置位，在发生 TBCMP1 时复位
- HRTIM_CHC1 在发生主定时器 MCMP2 时置位，在发生 TCCMP1 时复位

发生 TxCMP2 比较事件时会生成 ADC 触发信号。由于所有 ADC 触发源的相位均由于转换器拓扑而发生了偏移，因此可以将所有触发源合并为单个 ADC 触发源，从而节省 ADC 资源（例如将 1 个 ADC 常规通道用于整个多相转换器）

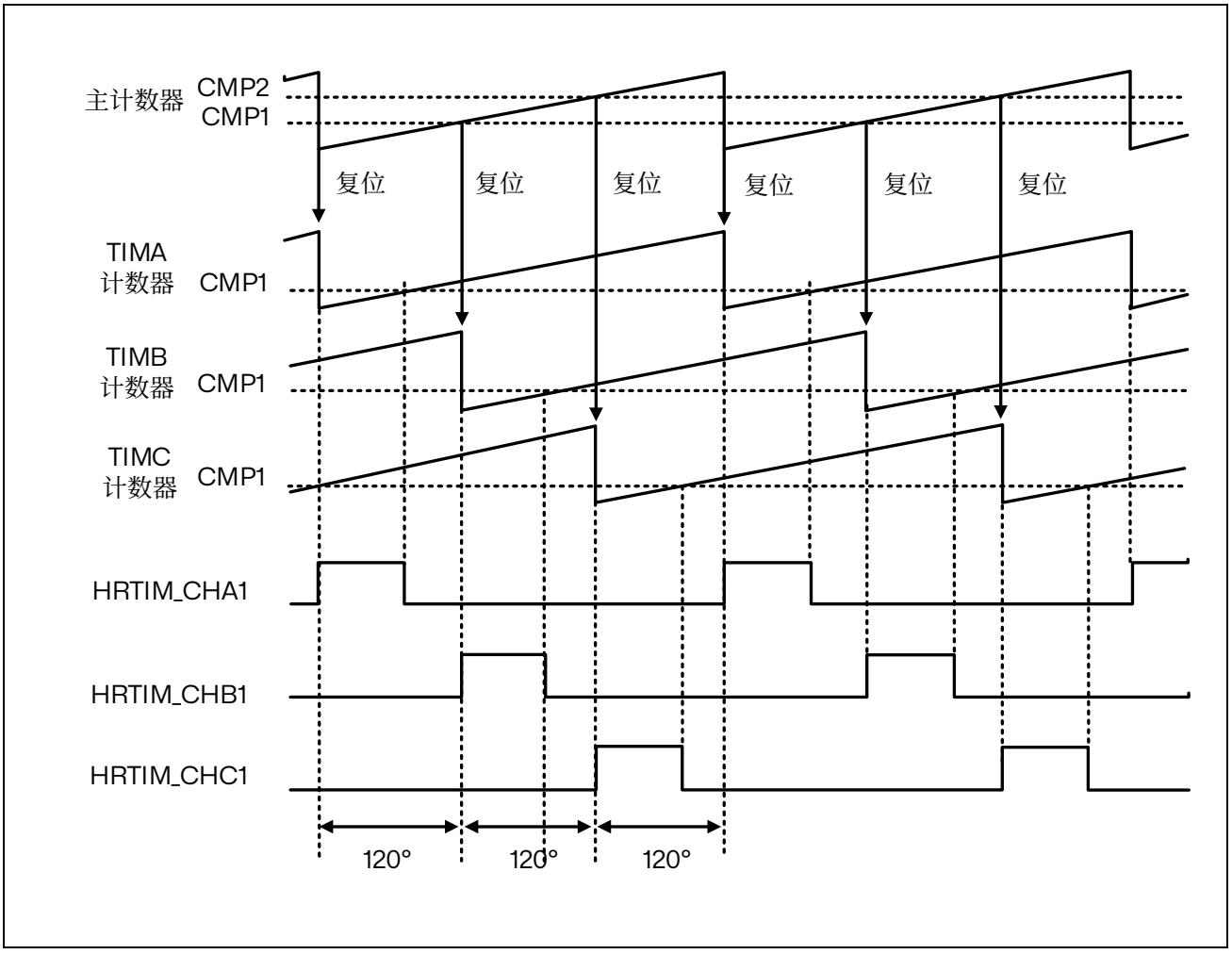



图 21.85 三相交错降压转换器控制

### 21.4.4 过渡模式功率因数校正

基本工作原理是，在固定的 Ton 时间内建立进入电感的电流。该电流将随后在 Toff 时间内消退，当电流为零时重新开始此循环周期。可通过过零检查电路 (ZCD) 检测此现象，如下图所示。由于有恒定 Ton 时间，电感中的电流峰值与整流的交流输入电压成比例，这提供了功率因数校正。

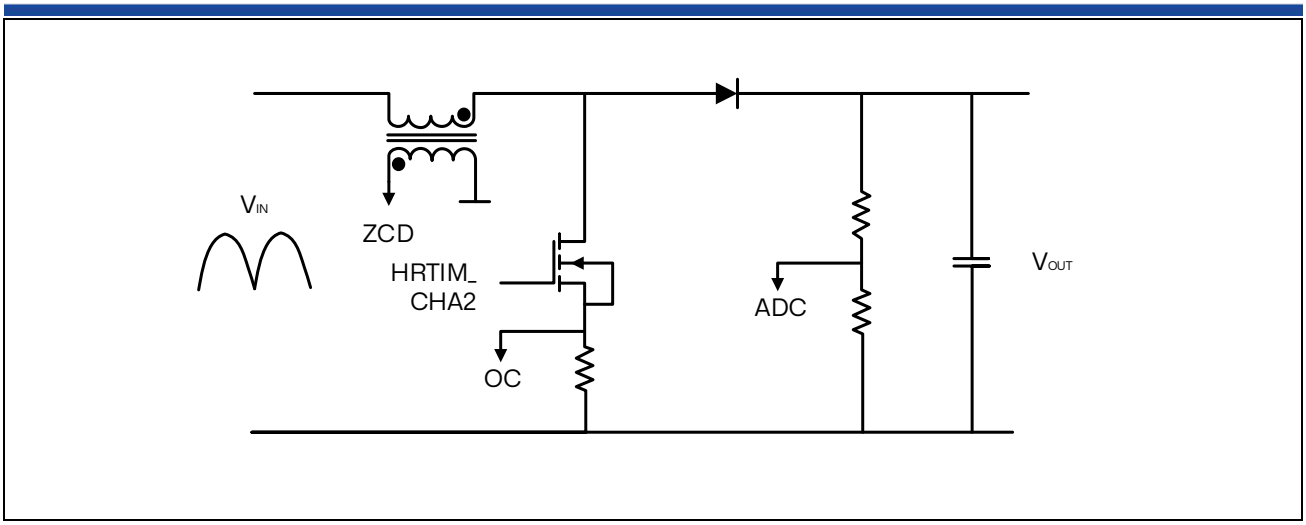



图 21.86 过渡模式功率因数校正（PFC）

此转换器具有恒定的 Ton 时间，由于 Toff 时间变化其频率也是变化的（取决于输入电压）。它还必须包括一些功能，以便在未检测到过零电压时工作，或在过流 (OC) 时限制 Ton 时间。OC 反馈通常由内置比较器调节，发送到外部事件输入。

下图显示了各工作模式中的波形，其参数定义如下：

- Ton Min：屏蔽伪过流（续流二极管恢复电流），用 OC 消隐表示。
- Ton Max：实际上是转换器置位点。它由 CMP1 定义。
- Toff Min：在电流限值接近零时（退磁非常快）限制频率。它由 CMP2 定义。
- Toff Max：避免在无 ZCD 发生时系统卡死。自动延迟模式下它由 CMP4 定义。

由于值必须与输出下降沿相关，因此两个 Toff 值都会自动延迟。

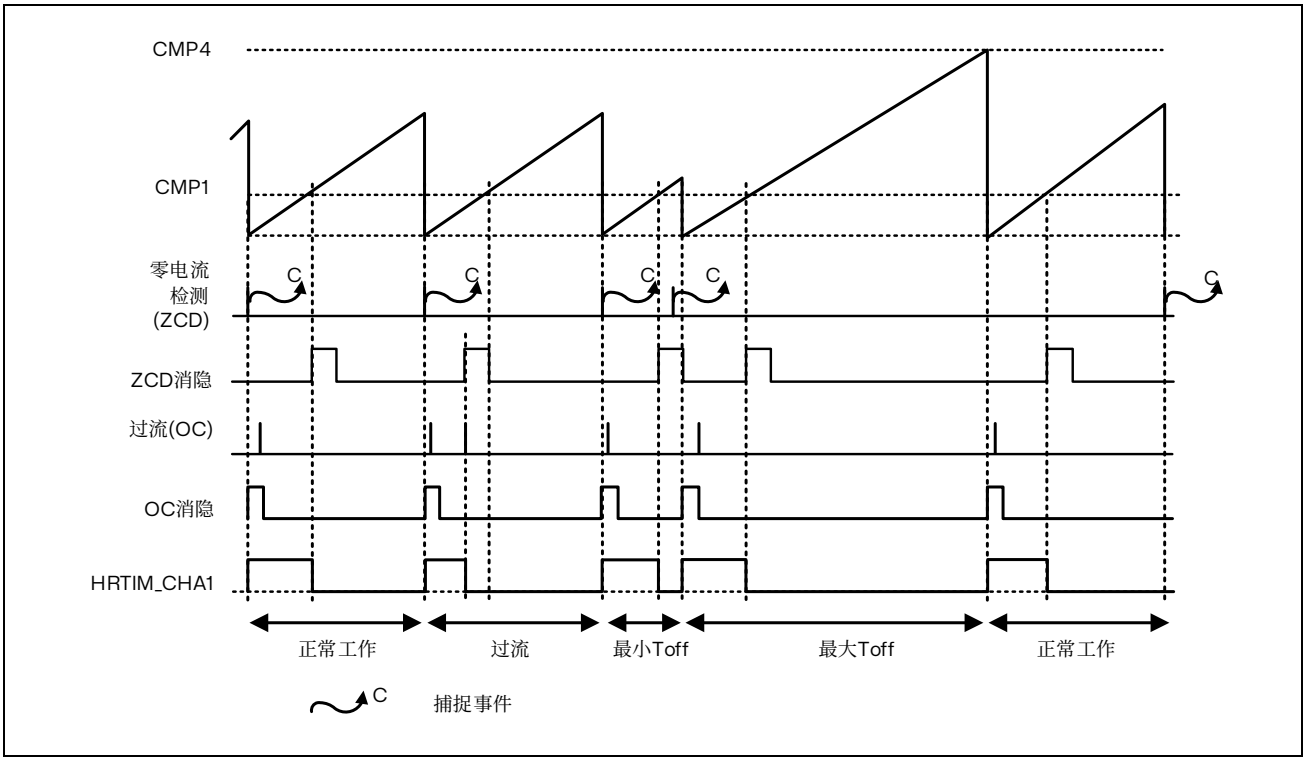



图 21.87 过渡模式功率因数校正（PFC）波形

## 21.5 HRTIM 寄存器

### 21.5.1 HRTIM 主定时器控制寄存器（HRTIM_MCR）

偏移地址：0x000

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRSTDMA[1:0] | | MREPU | Res | PREEN | DACSYNC[1:0] | | Reserved | | TFCEN | TECEN | TDCEN | TCCEN | TBCEN | TACEN | MCEN |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNCSRC[1:0] | | SYNCOUT[1:0] | | SYNCSTRTM | SYNCRSTM | SYNCIN[1:0] | | INTLVD[1:0] | | HALF | RETRIG | CONT | CKPSC[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:30 **BRSTDMA[1:0]**：突发 DMA 更新（Burst DMA update）

这些位定义更新与突发 DMA 事务的相对关系。

00：独立于 DMA 突发传输进行更新。

01：DMA 突发传输结束时进行更新。

10：在 DMA 突发传输结束后的主定时器翻转时进行更新。此模式仅适用于连续模式。

11：保留

Bit 29 **MREPU**：主定时器重复更新（Master timer repetition update）

此位定义主定时器重复周期结束后是否进行更新（因翻转或复位事件而更新）。

仅当 BRSTDMA[1:0] = 00 或 01 时，才可将 MREPU 置 1。

0：禁止与重复有关的更新

1：使能与重复有关的更新

Bit 28 保留，必须保持为复位值。

Bit 27 **PREEN**：预加载使能（Preload enable）

此位可使能寄存器预装载机制，并定义对存储器映射寄存器的写访问是在 HRTIM 的活动寄存器中进行还是在预装载寄存器中进行。

0：禁止预装载：直接对活动寄存器进行写访问

1：使能预装载：对预装载寄存器进行写访问

Bit 26:25 **DACSYNC[1:0]**：DAC 同步（DAC synchronization）

发生主定时器更新事件时，可使能并生成 DAC 同步事件。这些位会定义在哪路输出上发送 DAC 同步事件（详情请参考章节：DAC 触发连接）。

00：未生成 DAC 触发事件

01：在 hrtim_dac_trg1 上生成触发事件

10：在 hrtim_dac_trg2 上生成触发事件

11：在 hrtim_dac_trg3 上生成触发事件

Bit 24:23 保留，必须保持为复位值。

Bit 22 **TFCEN**：定时器 F 计数器使能（Timer F counter enable）

此位用于启动定时器 F 计数器。

0：禁用定时器 F 计数器

1：使能定时器 F 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 21 **TECEN**：定时器 E 计数器使能（Timer E counter enable）

此位用于启动定时器 E 计数器。

0：禁用定时器 E 计数器

1：使能定时器 E 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 20 **TDCEN**：定时器 D 计数器使能（Timer D counter enable）

此位用于启动定时器 D 计数器。

0：禁用定时器 D 计数器

1：使能定时器 D 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 19 **TCCEN**：定时器 C 计数器使能（Timer C counter enable）

此位用于启动定时器 C 计数器。

0：禁用定时器 C 计数器

1：使能定时器 C 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 18 **TBCEN**：定时器 B 计数器使能（Timer B counter enable）

此位用于启动定时器 B 计数器。

0：禁用定时器 B 计数器

1：使能定时器 B 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 17 **TACEN**：定时器 A 计数器使能（Timer A counter enable）

此位用于启动定时器 A 计数器。

0：禁用定时器 A 计数器

1：使能定时器 A 计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 16 **MCEN**：主定时器计数器使能（Master timer counter enable）

此位用于启动主定时器的计数器。

0：禁止主定时器计数器

1：使能主定时器计数器

*注：此位至少在 8 个 $f_{HRTIM}$ 时钟周期内不得更改。*

Bit 15:14 **SYNCSRC[1:0]**：同步源（Synchronization source）

这些位用于定义将在同步输出 SYNCOUT[2:1]上发送的源和事件。

00：主定时器启动

01：主定时器比较 1 事件

10：定时器 A 启动/复位

11：定时器 A 比较 1 事件

Bit 13:12 **SYNCOUT[1:0]**：同步输出（Synchronization output）

这些位用于定义同步输出事件的路由和调节。

00：禁用

01：保留。

10：在 HRTIM_SCOUT 输出上的正脉冲（16 个 $f_{HRTIM}$ 时钟周期）

11：在 HRTIM_SCOUT 输出上的负脉冲（16 个 $f_{HRTIM}$ 时钟周期）

*注：一旦计数器被使能（TxCEN 位置1），就不应修改此位域。*

Bit 11 **SYNCSTRTM**：主定时器同步启动（Synchronization starts master）

该位能够使主定时器在接收到同步输入事件时启动：

0：对主定时器没有影响

1：同步输入事件会启动主定时器

Bit 10 **SYNCRSTM**：主定时器同步复位（Synchronization resets master）

该位能够使主定时器在接收到同步输入事件时复位：

0：对主定时器没有影响

1：同步输入事件会复位主定时器

Bit 9:8 **SYNCIN[1:0]**：同步输入（Synchronization input）

这些位定义了同步输入源：

00：禁用。HRTIM 不进行同步，以独立模式运行。

01：保留。

10：内部事件通过 hrtim_in_sync[2]：HRTIM 与片上定时器同步（见同步输入部分）。

11：外部事件通过 hrtim_in_sync[3]：HRTIM_SCIN 输入上的正脉冲触发 HRTIM。

*注：一旦受影响的定时器被使能，就不能更改这个参数。*

Bit 7:6 **INTLVD[1:0]**：交错模式（Interleaved mode）

这个位域仅在 HALF 位被清零时有效。它使能了交错模式：

00：交错模式禁用

01：三重交错模式：当写入 HRTIM_MPER 寄存器时，HRTIM_MCMP1R 有效寄存器会自动更新为 HRTIM_MPER/3 的值，HRTIM_MCMP2R 有效寄存器会自动更新为 2x(HRTIM_MPER/3)的值。

10：四重交错模式：当写入 HRTIM_MPER 寄存器时，HRTIM_MCMP1R 有效寄存器会自动更新为 HRTIM_MPER/4 的值，HRTIM_MCMP2R 有效寄存器会自动更新为 HRTIM_MPER/2 的值，HRTIM_MCMP3R 有效寄存器会自动更新为 3x(HRTIM_MPER/4)的值。

11：交错模式禁用

Bit 5 **HALF**：半占空比模式（Half mode）

此位用于使能半占空比模式：HRTIM_MPER 寄存器写入内容后，HRTIM_MCMP1R 活动寄存器会自动更新为 HRTIM_MPER/2 值。

0：禁止半占空比模式

1：使能半占空比模式

Bit 4 **RETRIG**：可再触发模式（Re-triggerable mode）

这个位定义了主定时器计数器在单次触发模式下的行为：

0：定时器不可再触发：仅当计数器停止（周期结束）时才能进行计数器复位。

1：定时器可再触发：无论计数器的状态（运行中或已停止），都可以进行计数器复位。

Bit 3 **CONT**：连续模式（Continuous mode）

这个位定义了主定时器计数器在单次触发模式下的行为：

0：定时器以单次触发模式工作，并在达到 MPER 值时停止。

1：定时器以连续（自由运行）模式工作，并在达到 MPER 值时翻转到零。

Bit 2:0 **CKPSC [2:0]**：时钟预分频器（Clock prescaler）

这些位定义了主定时器高分辨率时钟的预分频器比率。
计数器时钟等效频率（$f_{COUNTER}$）等于 $f_{HRCK}$ / $2^{CKPSC[2:0]}$。
一旦定时器被使能，预分频比率就不能再修改。

### 21.5.2 HRTIM 主定时器中断状态寄存器（HRTIM_MISR）

偏移地址：0x004
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Resverd | | | | | | | | | MUPD | SYNC | MREP | MCMP4 | MCMP3 | MCMP2 | MCMP1 |
| | | | | | | | | | r | r | r | r | r | r | r |

Bit 31:7 保留，必须保持复位值。

Bit 6 **MUPD**：主定时器更新中断标志（Master update interrupt flag）
当主定时器寄存器被更新时，该位由硬件置 1。
0：未发生主定时器更新中断
1：已发生主定时器更新中断

Bit 5 **SYNC**：同步输入中断标志（Sync input interrupt flag）
当接收到同步输入事件时，该位由硬件置 1。
0：未发生同步输入中断
1：已发生同步输入中断

Bit 4 **MREP**：主定时器重复中断标志（Master repetition interrupt flag）
当主定时器重复周期结束时，该位由硬件置 1。
0：未发生主定时器重复中断
1：已发生主定时器重复中断

Bit 3 **MCMP4**：主定时器比较 4 中断标志（Master compare 4 interrupt flag）
参考 MCMP1 的描述

Bit 2 **MCMP3**：主定时器比较 3 中断标志（Master compare 3 interrupt flag）
参考 MCMP1 的描述

Bit 1 **MCMP2**：主定时器比较 2 中断标志（Master compare 2 interrupt flag）
参考 MCMP1 的描述

Bit 0 **MCMP1**：主定时器比较 1 中断标志（Master compare 1 interrupt flag）
当主定时器计数器与主定时器比较 1 寄存器中编程的值匹配时，该位由硬件置 1。
0：未发生主定时器比较 1 中断
1：已发生主比定时器较 1 中断

### 21.5.3 HRTIM 主定时器中断清除寄存器（HRTIM_MICR）

偏移地址：0x008

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Resverd | | | | | | | | | MUPDC | SYNCC | MREPC | MCMP4C | MCMP3C | MCMP2C | MCMP1C |
| | | | | | | | | | w | w | w | w | w | w | w |

Bit 31:7 保留，必须保持复位值。

Bit 6 **MUPDC**：主更新中断标志清除（Master update interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MUPD 标志。

Bit 5 **SYNCC**：同步输入中断标志清除（Sync input interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 SYNC 标志。

Bit 4 **MREPC**：重复中断标志清除（Repetition interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MREP 标志。

Bit 3 **MCMP4C**：主比较 4 中断标志清除（Master compare 4 interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MCMP4 标志。

Bit 2 **MCMP3C**：主比较 3 中断标志清除（Master compare 3 interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MCMP3 标志。

Bit 1 **MCMP2C**：主比较 2 中断标志清除（Master compare 2 interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MCMP2 标志。

Bit 0 **MCMP1C**：主比较 1 中断标志清除（Master compare 1 interrupt flag clear）
向此位写入 1 将清除 HRTIM_MISR 寄存器中的 MCMP1 标志。

### 21.5.4 HRTIM 主定时器 DMA/中断使能寄存器（HRTIM_MDIER）

偏移地址：0x00C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | MUPDDE | SYNCDE | MREPDE | MCMP4DE | MCMP3DE | MCMP2DE | MCMP1DE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | MUPDIE | SYNCIE | MREPIE | MCMP4IE | MCMP3IE | MCMP2IE | MCMP1IE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

Bit 31:23 保留，必须保持复位值。

Bit 22 **MUPDDE**：主定时器更新 DMA 请求使能（Master update DMA request enable）

此位由软件设置和清除，以使能/禁用主定时器更新 DMA 请求。

0：主定时器更新 DMA 请求禁用

1：主定时器更新 DMA 请求使能

Bit 21 **SYNCDE**：同步输入 DMA 请求使能（Sync input DMA request enable）

此位由软件设置和清除，以使能/禁用同步输入 DMA 请求。

0：同步输入 DMA 请求禁用

1：同步输入 DMA 请求使能

Bit 20 **MREPDE**：主定时器重复 DMA 请求使能（Master repetition DMA request enable）

此位由软件设置和清除，以使能/禁用主定时器重复 DMA 请求。

0：重复 DMA 请求禁用

1：重复 DMA 请求使能

Bit 19 **MCMP4DE**：主定时器比较 4 DMA 请求使能（Master compare 4 DMA request enable）

参考 MCMP1DE 的描述

Bit 18 **MCMP3DE**：主定时器比较 3 DMA 请求使能（Master compare 3 DMA request enable）

参考 MCMP1DE 的描述

Bit 17 **MCMP2DE**：主定时器比较 2 DMA 请求使能（Master compare 2 DMA request enable）

参考 MCMP1DE 的描述

Bit 16 **MCMP1DE**：主定时器比较 1 DMA 请求使能（Master compare 1 DMA request enable）

此位由软件设置和清除，以使能/禁用主定时器比较 1 的 DMA 请求。

0：比较 1 DMA 请求禁用

1：比较 1 DMA 请求使能

Bit 15:7 保留，必须保持复位值。

Bit 6 **MUPDIE**：主定时器更新中断使能（Master update interrupt enable）

此位由软件设置和清除，以使能/禁用主定时器寄存器更新的中断。

0：主定时器更新中断禁用

1：主定时器更新中断使能

Bit 5 **SYNCIE**：同步输入中断使能（Sync input interrupt enable）

此位由软件设置和清除，以使能/禁用同步输入中断。

0：同步输入中断禁用

1：同步输入中断使能

Bit 4 **MREPIE**：主定时器重复中断使能（Master repetition interrupt enable）

此位由软件设置和清除，以使能/禁用主定时器重复中断。

0：主定时器重复中断禁用

1：主定时器重复中断使能

Bit 3 **MCMP4IE**：主定时器比较 4 中断使能（Master compare 4 interrupt enable）

参考 MCMP1IE 的描述

Bit 2 **MCMP3IE**：主定时器比较 3 中断使能（Master compare 3 interrupt enable）

参考 MCMP1IE 的描述

Bit 1 **MCMP2IE**：主定时器比较 2 中断使能（Master compare 2 interrupt enable）

参考 MCMP1IE 的描述

Bit 0 **MCMP1IE**：主定时器比较 1 中断使能（Master compare 1 interrupt enable）

此位由软件设置和清除，以使能/禁用主定时器比较 1 中断。

0：比较 1 中断禁用

1：比较 1 中断使能

### 21.5.5 HRTIM 主定时器计数器寄存器（HRTIM_MCNTR）

偏移地址：0x010

复位值：0x0000 0000

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MCNT[15:0]：**计数器值（Counter value）

该位域存储主定时器的当前计数器值。这个寄存器仅当主定时器停止时（即 HRTIM_MCR 寄存器中的 MCEN 位为 0）才允许写入。

*注：当 HR 时钟的预分频比率低于 32（即 CKPSC[2:0] 小于 5）时，计数器的最低有效位没有意义。这些位无法写入，并且在读取时返回 0。*

*注：如果设置的计数器值大于 HRTIM_MPER 寄存器的值，则定时器的行为无法得到保证。*

### 21.5.6 HRTIM 主定时器周期寄存器（HRTIM_MPER）

偏移地址：0x014

复位值：0x0000 FFDF

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MPER[15:0]：**主定时器周期值（Master timer period value）

此寄存器定义了计数器溢出值。

周期值必须大于或等于 $f_{HRTIM}$ 时钟的 3 个周期，即如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18，以此类推。

最大值为 0x0000FFDF。

### 21.5.7 HRTIM 主定时器重复寄存器（HRTIM_MREP）

偏移地址：0x018

复位值：0x0000 0000

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

Bit 31:8 保留，必须保持复位值。

Bit 7:0 **MREP[7:0]：**主定时器重复周期值（Master timer repetition period value）

此寄存器保存主定时器计数器的重复周期值。该寄存器为预装载寄存器，禁止预装载时则为活动寄存器。

### 21.5.8 HRTIM 主定时器比较 1 寄存器（HRTIM_MCMP1R）

偏移地址：0x01C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MCMP1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MCMP1[15:0]：**主定时器比较 1 值（Master timer compare 1 value）

该寄存器保存主定时器比较 1 值。该寄存器为预装载寄存器，禁止预装载时则为活动寄存器。

比较值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18，以此类推。

### 21.5.9 HRTIM 主定时器比较 2 寄存器（HRTIM_MCMP2R）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MCMP2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MCMP2[15:0]：**主定时器比较 2 值（Master timer compare 2 value）

该寄存器保存主定时器比较 2 值。该寄存器为预装载寄存器，禁止预装载时则为活动寄存器。

比较值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18，以此类推。

### 21.5.10 HRTIM 主定时器比较 3 寄存器（HRTIM_MCMP3R）

偏移地址：0x028

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MCMP3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MCMP3[15:0]：**主定时器比较 3 值（Master timer compare 3 value）

该寄存器保存主定时器比较 3 值。该寄存器为预装载寄存器，禁止预装载时则为活动寄存器。

比较值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18，以此类推。

### 21.5.11 HRTIM 主定时器比较 4 寄存器（HRTIM_MCMP4R）

偏移地址：0x02C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MCMP4[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **MCMP4[15:0]：**主定时器比较 4 值（Master timer compare 4 value）

该寄存器保存主定时器比较 4 值。该寄存器为预装载寄存器，禁止预装载时则为活动寄存器。

比较值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0] = 0，则为 0x60；如果 CKPSC[2:0] = 1，则为 0x30；如果 CKPSC[2:0] = 2，则为 0x18，以此类推。

### 21.5.12 HRTIM 定时器 x 控制寄存器 (HRTIM_TIMxCR) (x = A~F)

偏移地址：块 A：0x080
偏移地址：块 B：0x100
偏移地址：块 C：0x180
偏移地址：块 D：0x200
偏移地址：块 E：0x280
偏移地址：块 F：0x300
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UPDGAT[3:0] | | | | PREEN | DACSYNC[1:0] | | MSTU | TEU | TDU | TCU | TBU | TAU | TxRSTU | TxREPU | TFU |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DELCMP4[1:0] | | DELCMP2[1:0] | | SYNCSTRTx | SYNCRSTx | RSYNCU | INTLVD[1:0] | | PSHPLL | HALF | RETRIG | CONT | CKPSCx[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:28 **UPDGAT[3:0]**：更新门控（Update gating）

这些位定义了更新操作相对于突发 DMA 事务和外部更新请求如何发生，这些外部更新请求通过更新使能输入 hrtim_upd_en[3:1]进行。

更新事件可以包括：MSTU、TFU、TEU、TDU、TCU、TBU、TAU、TxRSTU、TxREPU。

0000：更新独立于 DMA 突发传输发生

0001：当 DMA 突发传输完成时发生更新

0010：在 DMA 突发传输完成后的更新事件发生时进行更新

0011：在 hrtim_upd_en1 的上升沿时发生更新

0100：在 hrtim_upd_en2 的上升沿时发生更新

0101：在 hrtim_upd_en3 的上升沿时发生更新

0110：在 hrtim_upd_en1 上升沿后的更新事件发生时进行更新

0111：在 hrtim_upd_en2 上升沿后的更新事件发生时进行更新

1000：在 hrtim_upd_en3 上升沿后的更新事件发生时进行更新

其他：保留

*注：在编程新值之前，必须重置此位域。*

*对于*UPDGAT[3:0]*的值等于*0001、0011、0100、0101 *时，可能有多个并发的更新源（例如*RSTU *和*DMA *突发）。*

Bit 27 **PREEN**：预加载使能（Preload enable）

此位使能寄存器的预加载机制，并定义写入预加载寄存器时是写入有效寄存器还是预加载寄存器。

0：预加载禁用：直接对活动寄存器进行写访问

1：预加载使能：对预装载寄存器进行写访问

Bit 26:25 **DACSYNC[1:0]**：DAC 同步（DAC synchronization）

当定时器更新发生时，会生成一个 DAC 同步事件。这些位定义了 DAC 同步是在哪个输出上发送的（详情参考章节：DAC 触发连接）。

00：不生成 DAC 触发事件

01：在 hrtim_dac_trg1 上生成触发事件

10：在 hrtim_dac_trg2 上生成触发事件

11：在 hrtim_dac_trg3 上生成触发事件

Bit 24 **MSTU**：主定时器更新（Master timer update）

当主定时器更新时，会触发寄存器更新。

0：由主定时器触发的更新禁用

1：由主定时器触发的更新使能

Bit 23 **TEU**：定时器 E 更新（Timer E update）

当定时器 E 更新时，会触发寄存器更新。

0：由定时器 E 触发的更新禁用

1：由定时器 E 触发的更新使能

*注：这个位是为*HRTIM_TIMECR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMFCR *可用。*

Bit 22 **TDU**：定时器 D 更新（Timer D update）

当定时器 D 更新时，会触发寄存器更新。

0：由定时器 D 触发的更新禁用
1：由定时器 D 触发的更新使能
*注：这个位是为*HRTIM_TIMECR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMFCR *可用。*

Bit 21 **TCU**：定时器 C 更新（Timer C update）
当定时器 C 更新时，会触发寄存器更新。
0：由定时器 C 触发的更新禁用
1：由定时器 C 触发的更新使能
*注：这个位是为*HRTIM_TIMECR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMFCR *可用。*

Bit 20 **TBU**：定时器 B 更新（Timer B update）
当定时器 B 更新时，会触发寄存器更新。
0：由定时器 B 触发的更新禁用
1：由定时器 B 触发的更新使能
*注：这个位是为*HRTIM_TIMECR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMFCR *可用。*

Bit 19 **TAU**：定时器 A 更新（Timer A update）
当定时器 A 更新时，会触发寄存器更新。
0：由定时器 A 触发的更新禁用
1：由定时器 A 触发的更新使能
*注：这个位是为*HRTIM_TIMECR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMFCR *可用。*

Bit 18 **TxRSTU**：定时器 x 复位更新（Timer x reset update）
当定时器 x 的计数器复位或在连续模式下达到周期值后翻转到 0 时会触发寄存器更新。
0：由定时器 x 复位或翻转触发的更新禁用
1：由定时器 x 复位或翻转触发的更新使能

Bit 17 **TxREPU**：定时器 x 重复更新（Timer x repetition update）
当计数器翻转且 HRTIM_REPx = 0 时，会触发寄存器更新。
0：重复时更新禁用
1：重复时更新使能

Bit 16 **TFU**：定时器 F 更新（Timer F update）
当定时器 F 更新时，会触发寄存器更新。
0：由定时器 F 触发的更新禁用
1：由定时器 F 触发的更新使能
*注：这个位是为*HRTIM_TIMFCR *保留的。它仅对*HRTIM_TIMACR、HRTIM_TIMBCR、HRTIM_TIMCCR、HRTIM_TIMDCR、HRTIM_TIMECR *可用。*

Bit 15:14 **DELCMP4[1:0]**：CMP4 自动延迟模式（CMP4 auto-delayed mode）
这个位域定义了比较寄存器是工作在标准模式（当计数器值等于比较值时立即发出比较匹配），还是工作在自动延迟模式（参见“自动延迟模式”章节）。

00：CMP4 寄存器始终有效（标准比较模式）
01：CMP4 值在捕获 2 事件发生后重新计算并生效
10：CMP4 值在捕获 2 事件发生后重新计算并生效，或者在比较 1 匹配后重新计算并生效（如果缺少捕获 2 事件，则作为超时功能）
11：CMP4 值在捕获 2 事件发生后重新计算并生效，或者在比较 3 匹配后重新计算并生效（如果缺少捕获事件，则作为超时功能）

*注：一旦计数器使能（TxCEN 位设置），就不应修改这个位域。*

Bit 13:12 **DELCMP2 [1:0]：**CMP2 自动延迟模式（CMP2 auto-delayed mode）

这个位域定义了比较寄存器是工作在标准模式（当计数器值等于比较值时立即发出比较匹配），还是工作在自动延迟模式（参见“自动延迟模式”章节）。

00：CMP2 寄存器始终活跃（标准比较模式）
01：CMP2 值在捕获 1 事件发生后重新计算并生效
10：CMP2 值在捕获 1 事件发生后重新计算并生效，或者在比较 1 匹配后重新计算并生效（如果缺少捕获事件，则作为超时功能）
11：CMP2 值在捕获 1 事件发生后重新计算并生效，或者在比较 3 匹配后重新计算并生效如果缺少捕获事件，则作为超时功能）

*注：一旦计数器使能（TxCEN 位设置），就不应修改这个位域。*

Bit 11 **SYNCSTRTx**：同步启动定时器 x（Synchronization starts timer x）

这个位定义了定时器 x 在同步事件发生后的行为：
0：对定时器 x 无影响
1：同步输入事件启动定时器 x

Bit 10 **SYNCRSTx**：同步复位定时器 x（Synchronization resets timer x）

这个位定义了定时器 x 在同步事件发生后的行为：
0：对定时器 x 无影响
1：同步输入事件复位定时器 x

Bit 9 **RSYNCU**：重新同步更新（Re-synchronized update）

这个位指定了来自定时单元外部的更新源是否需要进行同步：
0：来自相邻定时器（当 MSTU、TAU、TBU、TCU、TDU、TEU、TFU 位设置时）或来自软件更新（TxSWU 位）的更新会立即被考虑
1：来自相邻定时器（当 MSTU、TAU、TBU、TCU、TDU、TEU、TFU 位设置时）或来自软件更新（TxSWU 位）的更新会在下一个重置/翻转事件时被考虑

*注：此位仅在 UPDGAT[3:0] = 0000 时有效，否则会被忽略。*

Bit 8:7 **INTLVD[1:0]：**交错模式（Interleaved mode）

此位域在 HALF 位被重置时有效，用于使能交错模式。
00：交错模式禁用
01：三重交错模式：当写入 HRTIM_PERxR 寄存器时，HRTIM_CMP1xR 有效寄存器会自动更新为 HRTIM_PERxR/3 的值，HRTIM_CMP2xR 有效寄存器会自动更新为 2x(HRTIM_PERxR/3)的值。
10：四重交错模式：当写入 HRTIM_PERxR 寄存器时，HRTIM_CMP1xR 有效寄存器会自动更新为 HRTIM_PERxR/4 的值，HRTIM_CMP2xR 有效寄存器会自动更新为 HRTIM_PERxR/2 的值，HRTIM_CMP3xR 有效寄存器会自动更新为 3x(HRTIM_PERxR/4)的值。

11：交错模式禁用

Bit 6 **PSHPLL**：推挽模式使能（Push-pull mode enable）

此位用于使能推挽模式。

0：推挽模式禁用

1：推挽模式使能

*注：一旦计数器使能（*TxCEN *位设置），就不应修改这个位域。*

Bit 5 **HALF**：半占空比模式使能（Half mode enable）

此位用于使能半占空比模式：当写入 HRTIM_PERxR 寄存器时，HRTIM_CMP1xR 有效寄存器会自动更新为 HRTIM_PERxR/2 的值。

0：半占空比模式禁用

1：半占空比模式使能

Bit 4 **RETRIG**：可再触发模式（Re-triggerable mode）

此位定义了计数器在单次触发模式下的行为。

0：定时器不可再触发：如果计数器停止（在单次触发模式下周期已过或在连续模式下计数器停止），则执行计数器复位。

1：定时器可再触发：无论计数器的状态如何，都会执行计数器复位。

Bit 3 **CONT**：连续模式（Continuous mode）

此位定义了定时器的工作模式。

0：定时器以单次触发模式运行，并在达到 TIMxPER 值时停止。

1：定时器以连续模式运行，并在达到 TIMxPER 值时翻转到零。

Bit 2:0 **CKPSCx[2:0]**：HRTIM 定时器 x 时钟预分频器（HRTIM timer x clock prescaler）

这些位定义了主定时器高分辨率时钟的预分频比例。

计数器的等效频率（$f_{COUNTER}$）等于 $f_{HRCK} / 2^{CKPSC[2:0]}$。

一旦定时器使能，就不能修改预分频比例。

### 21.5.13 HRTIM 定时器 x 中断状态寄存器 (HRTIM_TIMxISR) (x = A~F)

偏移地址：块 A：0x084
偏移地址：块 B：0x104
偏移地址：块 C：0x184
偏移地址：块 D：0x204
偏移地址：块 E：0x284
偏移地址：块 F：0x304
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | O2CPY | O1CPY | O2 STAT | O1 STAT | IPP STAT | CPP STAT |
| | | | | | | | | | | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | DLYPRT | RST | RSTx2 | SETx2 | RSTx1 | SETx1 | CPT2 | CPT1 | UPD | Res | REP | CMP4 | CMP3 | CMP2 | CMP1 |
| | r | r | r | r | r | r | r | r | r | | r | r | r | r | r |

Bit 31:22 保留，必须保持复位值。

Bit 21 **O2CPY**：输出 2 副本（Output 2 copy）

这此状态位是输出级之前输出 2 状态的原始副本（斩波、极性），允许在延迟保护后重新使能输出之 前检查当前输出状态。

0：输出 2 无效

1：输出 2 有效

Bit 20 **O1CPY**：输出 1 副本（Output 1 copy）

这此状态位是输出级之前输出 1 状态的原始副本（斩波、极性），允许在延迟保护后重新使能输出之 前检查当前输出状态。

0：输出 1 无效

1：输出 1 有效

Bit 19 **O2STAT**：输出 2 状态（Output 2 status）

此状态位指示延迟空闲保护触发时的输出 2 状态。进入任何新的延迟保护时，会更新此位。均衡空闲模式下，不会更新此位。

0：输出 2 无效

1：输出 2 有效

Bit 18 **O1STAT**：输出 1 状态（Output 1 status）

此状态位指示延迟空闲保护触发时的输出 1 状态。进入任何新的延迟保护时，会更新此位。均衡空闲模式下，不会更新此位。

0：输出 1 无效

1：输出 1 有效

Bit 17 **IPPSTAT**：空闲推挽状态（Idle push-pull Status）

此状态位指示触发保护时，在推挽模式、均衡故障模式或延迟空闲模式下将信号应用到哪路输出（不考虑输出状态有效还是无效）。

0：输出 1 有效时进行保护，输出 2 强制无效

1：输出 2 有效时进行保护，输出 1 强制无效

Bit 16 **CPPSTAT**：当前推挽状态（Current push-pull status）

此状态位指示推挽模式下当前将信号应用到哪路输出上。此位仅在该配置中有意义。

0：信号应用到输出 1 上，输出 2 强制无效

1：信号应用到输出 2 上，输出 1 强制无效

Bit 15 保留，必须保持复位值。

Bit 14 **DLYPRT**：延迟保护标志（Delayed protection flag）

此位指示延迟空闲或均衡空闲模式进入。

0：没有发生延迟保护中断

1：发生了延迟空闲或均衡空闲模式进入

Bit 13 **RST**：复位和/或翻转中断标志（Reset and/or roll-over interrupt flag）

定时器 x 在连续模式下复位或翻转时，此位由硬件置 1。

0：没有发生 TIMx 计数器复位/翻转中断

1：发生了 TIMx 计数器复位/翻转中断

Bit 12 **RSTx2**：输出 2 复位中断标志（Output 2 reset interrupt flag）

请参考 RSTx1 的描述

Bit 11 **SETx2**：输出 2 置位中断标志（Output 2 set interrupt flag）

请参考 SETx1 的描述

Bit 10 **RSTx1**：输出 1 复位中断标志（Output 1 reset interrupt flag）

当 Tx1 输出复位时（从有效模式变为无效模式），此位由硬件置 1。

0：没有发生 Tx1 输出复位中断

1：发生了 Tx1 输出复位中断

Bit 9 **SETx1**：输出 1 置位中断标志（Output 1 set interrupt flag）

当 Tx1 输出复位时（从有效模式变为无效模式），此位由硬件置 1。

0：没有发生 Tx1 输出置位中断

1：发生了 Tx1 输出置位中断

Bit 8 **CPT2**：捕获 2 中断标志（Capture 2 interrupt flag）

请参考 CPT1 的描述。

Bit 7 **CPT1**：捕获 1 中断标志（Capture 1 interrupt flag）

发生定时器 x 捕获 1 事件时，该位由硬件置 1。

0：没有发生定时器 x 捕获 1 中断

1：发生了定时器 x 捕获 1 中断

Bit 6 **UPD**：更新中断标志（Update interrupt flag）

发生定时器 x 捕获 1 事件时，该位由硬件置 1。

0：没有发生定时器 x 更新中断

1：发生了定时器 x 更新中断

Bit 5 保留，必须保持复位值。

Bit 4 **REP**：重复中断标志（Repetition interrupt flag）

定时器 x 重复周期已过时，该位由硬件置 1。

0：没有发生定时器 x 重复中断

1：发生定时器 x 重复中断

Bit 3 **CMP4**：比较 4 中断标志（Compare 4 interrupt flag）

请参考 CMP1 的描述。

Bit 2 **CMP3**：比较 3 中断标志（Compare 3 interrupt flag）

请参考 CMP1 的描述。

Bit 1 **CMP2**：比较 2 中断标志（Compare 2 interrupt flag）

请参考 CMP1 的描述。

Bit 0 **CMP1**：比较 1 中断标志（Compare 1 interrupt flag）

当定时器 x 计数器值与在比较 1 寄存器中编程的值相匹配时，此位由硬件置 1。

0：没有发生比较 1 中断

1：发生了比较 1 中断

### 21.5.14 HRTIM 定时器 x 中断清除寄存器（HRTIM_TIMxICR）(x = A~F)

偏移地址：块 A：0x088

偏移地址：块 B：0x108

偏移地址：块 C：0x188

偏移地址：块 D：0x208

偏移地址：块 E：0x288

偏移地址：块 F：0x308

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res | DLYPRTC | RSTC | RSTx2C | SETx2C | RSTx1C | SETx1C | CPT2C | CPT1C | UPDC | Res | REPC | CMP4C | CMP3C | CMP2C | CMP1C |
| | w | w | w | w | w | w | w | w | w | | w | w | w | w | w |

Bit 31:15 保留，必须保持复位值。

Bit 14 **DLYPRTC**：延迟保护标志清除（Delayed protection flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 DLYPRT 标志。

Bit 13 **RSTC**：复位中断标志清除（Reset interrupt flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 RST 标志。

Bit 12 **RSTx2C**：输出 2 复位标志清除（Output 2 reset flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 RSTx2 标志。

Bit 11 **SETx2C**：输出 2 置位标志清除（Output 2 set flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 SETx2 标志。

Bit 10 **RSTx1C**：输出 1 复位标志清除（Output 1 reset flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 RSTx1 标志。

Bit 9 **SETx1C**：输出 1 置位标志清除（Output 1 set flag clear）

向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 SETx1 标志。

Bit 8 **CPT2C**：捕获 2 中断标志清除（Capture 2 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CPT2 标志。

Bit 7 **CPT1C**：捕获 1 中断标志清除（Capture 1 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CPT1 标志。

Bit 6 **UPDC**：更新中断标志清除（Update interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 UPD 标志。

Bit 5 保留，必须保持复位值。

Bit 4 **REPC**：重复中断标志清除（Repetition interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 REP 标志。

Bit 3 **CMP4C**：比较 4 中断标志清除（Compare 4 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CMP4 标志。

Bit 2 **CMP3C**：比较 3 中断标志清除（Compare 3 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CMP3 标志。

Bit 1 **CMP2C**：比较 2 中断标志清除（Compare 2 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CMP2 标志。

Bit 0 **CMP1C**：比较 1 中断标志清除（: Compare 1 interrupt flag clear）
向此位写入 1 可清除 HRTIM_TIMxISR 寄存器中的 CMP1 标志。

### 21.5.15 HRTIM 定时器 x DMA 中断使能寄存器（HRTIM_TIMxDIER) (x = A~F)

偏移地址：块 A：0x08C
偏移地址：块 B：0x10C
偏移地址：块 C：0x18C
偏移地址：块 D：0x20C
偏移地址：块 E：0x28C
偏移地址：块 F：0x30C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | DLYPRTDE | RSTDE | RSTx2DE | SETx2DE | RSTx1DE | SETx1DE | CPT2DE | CPT1DE | UPDDE | Res | REPDE | CMP4DE | CMP3DE | CMP2DE | CMP1DE |
| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | DLYPRTIE | RSTIE | RSTx2IE | SETx2IE | RSTx1IE | SETx1IE | CPT2IE | CPT1IE | UPDIE | Res | REPIE | CMP4IE | CMP3IE | CMP2IE | CMP1IE |
| | r | r | r | r | r | r | r | r | r | | r | r | r | r | r |

Bit 31 保留，必须保持复位值。

Bit 30 **DLYPRTDE**：延迟保护 DMA 请求使能（Delayed protection DMA request enable）

此位由软件设置和清除，以使能/禁用延迟保护的 DMA 请求。

0：延迟保护 DMA 请求禁用

1：延迟保护 DMA 请求使能

Bit 29 **RSTDE**：复位/翻转 DMA 请求使能（Reset/roll-over DMA request enable）

此位由软件设置和清除以在定时器 x 计数器复位或连续模式下翻转时使能/禁用 DMA 请求。

0：定时器 x 计数器复位/翻转 DMA 请求禁用

1：定时器 x 计数器复位/翻转 DMA 请求使能

Bit 28 **RSTx2DE**：输出 2 复位 DMA 请求使能（Output 2 reset DMA request enable）

参考 RSTx1DE 的描述。

Bit 27 **SETx2DE**：输出 2 置位 DMA 请求使能（Output 2 set DMA request enable）

参考 SETx1DE 的描述。

Bit 26 **RSTx1DE**：输出 1 复位 DMA 请求使能（Output 1 reset DMA request enable）

此位由软件设置和清除，以使能/禁用 Tx1 输出复位的 DMA 请求。

0：Tx1 输出复位 DMA 请求禁用

1：Tx1 输出复位 DMA 请求使能

Bit 25 **SETx1DE**：输出 1 设置 DMA 请求使能（Output 1 set DMA request enable）

此位由软件设置和清除，以使能/禁用 Tx1 输出设置的 DMA 请求。

0：Tx1 输出置位 DMA 请求禁用

1：Tx1 输出置位 DMA 请求使能

Bit 24 **CPT2DE**：捕获 2 DMA 请求使能 Capture 2 DMA request enable）

参考 CPT1DE 的描述。

Bit 23 **CPT1DE**：捕获 1 DMA 请求使能（Capture 1 DMA request enable）

此位由软件设置和清除，以使能/禁用捕获 1 的 DMA 请求。

0：捕获 1 DMA 请求禁用

1：捕获 1 DMA 请求使能

Bit 22 **UPDDE**：更新 DMA 请求使能（Update DMA request enable）

此位由软件设置和清除，以在更新事件时使能/禁用 DMA 请求。

0：更新 DMA 请求禁用

1：更新 DMA 请求使能

Bit 21 保留，必须保持复位值。

Bit 20 **REPDE**：重复 DMA 请求使能（Repetition DMA request enable）

此位由软件设置和清除，以在重复事件时使能/禁用 DMA 请求。

0：重复 DMA 请求禁用

1：重复 DMA 请求使能

Bit 19 **CMP4DE**：比较 4 DMA 请求使能（Compare 4 DMA request enable）

参考 CMP1DE 的描述。

Bit 18 **CMP3DE**：比较 3 DMA 请求使能（Compare 3 DMA request enable）

参考 CMP1DE 的描述。

Bit 17 **CMP2DE**：比较 2 DMA 请求使能（Compare 2 DMA request enable）

参考 CMP1DE 的描述。

Bit 16 **CMP1DE**：比较 1 DMA 请求使能（Compare 1 DMA request enable）

此位由软件设置和清除，以使能/禁用比较 1 的 DMA 请求。

0：比较 1 DMA 请求禁用

1：比较 1 DMA 请求使能

Bit 15 保留，必须保持复位值。

Bit 14 **DLYPRTIE**：延迟保护中断使能（Delayed protection interrupt enable）

此位由软件设置和清除，以使能/禁用延迟保护的中断。

0：延迟保护中断禁用

1：延迟保护中断使能

Bit 13 **RSTIE**：复位/翻转中断使能（Reset/roll-over interrupt enable）

此位由软件设置和清除，以在定时器 x 计数器复位或在连续模式下翻转时使能/禁用中断。

0：定时器 x 计数器复位/翻转中断禁用

1：定时器 x 计数器复位/翻转中断使能

Bit 12 **RSTx2IE**：输出 2 复位中断使能（Output 2 reset interrupt enable）

请参考 RSTx1IE 的描述。

Bit 11 **SETx2IE**：输出 2 设置中断使能（Output 2 set interrupt enable）

请参考 SETx1IE 的描述。

Bit 10 **RSTx1IE**：输出 1 复位中断使能（Output 1 reset interrupt enable）

此位由软件设置和清除，以使能/禁用 Tx1 输出复位的中断。

0：Tx1 输出复位中断禁用

1：Tx1 输出复位中断使能

Bit 9 **SETx1IE**：输出 1 设置中断使能（Output 1 set interrupt enable）

此位由软件设置和清除，以使能/禁用 Tx1 输出设置的中断。

0：Tx1 输出设置中断禁用

1：Tx1 输出设置中断使能

Bit 8 **CPT2IE**：捕获 2 中断使能（Capture 2 interrupt enable）

请参考 CPT1IE 的描述。

Bit 7 **CPT1IE**：捕获 1 中断使能（Capture 1 interrupt enable）

此位由软件设置和清除，以使能/禁用捕获 1 的中断。

0：捕获 1 中断禁用

1：捕获 1 中断使能

Bit 6 **UPDIE**：更新中断使能（Update interrupt enable）

此位由软件设置和清除，以使能/禁用更新事件的中断。

0：更新中断禁用

1：更新中断使能

Bit 5 保留，必须保持复位值。

Bit 4 **REPIE**：重复中断使能（Repetition interrupt enable）

此位由软件设置和清除，以使能/禁用重复事件的中断。

0：重复中断禁用

1：重复中断使能

Bit 3 **CMP4IE**：比较 4 中断使能（Compare 4 interrupt enable）

请参考 CMP1IE 的描述。

Bit 2 **CMP3IE**：比较 3 中断使能（Compare 3 interrupt enable）

请参考 CMP1IE 的描述。

Bit 1 **CMP2IE**：比较 2 中断使能（Compare 2 interrupt enable）

请参考 CMP1IE 的描述。

Bit 0 **CMP1IE**：比较 1 中断使能（Compare 1 interrupt enable）

此位由软件设置和清除，以使能/禁用比较 1 的中断。

0： 比较 1 中断禁用

1： 比较 1 中断使能

### 21.5.16 HRTIM 定时器 x 计数器寄存器（HRTIM_CNTxR）(x = A~F)

偏移地址：块 A：0x090
偏移地址：块 B：0x110
偏移地址：块 C：0x190
偏移地址：块 D：0x210
偏移地址：块 E：0x290
偏移地址：块 F：0x310
复位值：0x0000 0000

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **CNTx[15:0]：**定时器 x 计数器值（Timer x counter value）

此寄存器保存了定时器 x 的计数值。该值仅当定时器停止（HRTIM_TIMxCR 寄存器中的 TxCEN=0）时才能写入。

*注：当 HR 时钟预分频比低于 32（CKPSC[2:0]小于 5）时，计数器的最低有效位并不重要。它们无法被写入，并且在读取时返回 0。*

*注：如果计数值大于 HRTIM_PERxR 寄存器的值，则无法保证定时器的行为。*

### 21.5.17 HRTIM 定时器 x 周期寄存器（HRTIM_PERxR）(x = A~F)

偏移地址：块 A：0x094
偏移地址：块 B：0x114
偏移地址：块 C：0x194
偏移地址：块 D：0x214
偏移地址：块 E：0x294
偏移地址：块 F：0x314
复位值：0x0000 FFDF

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **PERx[15:0]：**定时器 x 的周期值（Timer x period value）

此寄存器保存定时器 x 的周期值。

如果禁止预装载，该寄存器保存预装载寄存器的内容和活动寄存器的内容。

周期值必须大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0]=0，则为 0x60；如果 CKPSC[2:0]=1，则为 0x30；如果 CKPSC[2:0]=2，则为 0x18；以此类推。

最大值为 0x0000 FFDF。

### 21.5.18 HRTIM 定时器 x 重复寄存器 (HRTIM_REPxR) (x = A~F)

偏移地址：块 A：0x098
偏移地址：块 B：0x118
偏移地址：块 C：0x198
偏移地址：块 D：0x218
偏移地址：块 E：0x298
偏移地址：块 F：0x318
复位值：0x0000 0000

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **REPx[7:0]：**定时器 x 的重复周期值（Timer x repetition period value）

此寄存器保存重复周期值。

该寄存器保存预装载寄存器的内容，如果禁止预装载，则会保存活动寄存器的内容。

### 21.5.19 HRTIM 定时器 x 比较 1 寄存器 (HRTIM_CMP1xR) (x = A~F)

偏移地址：块 A：0x09C
偏移地址：块 B：0x11C
偏移地址：块 C：0x19C
偏移地址：块 D：0x21C
偏移地址：块 E：0x29C
偏移地址：块 F：0x31C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMP1x[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **CMP1x[15:0]：**定时器 x 的比较 1 值（Timer x compare 1 value）

此寄存器保存比较 1 值。

如果预装载功能被禁用，此寄存器将保存预装载寄存器的内容或有效寄存器的内容。

比较值必须为 0 或大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0]=0，则为 0x60；如果 CKPSC[2:0]=1，则为 0x30；如果 CKPSC[2:0]=2，则为 0x18 以此类推。

0 值的编程遵循“空占空比异常情况”一节中描述的用例。

### 21.5.20 HRTIM 定时器 x 比较 1 复合寄存器（HRTIM_CMP1CxR) (x = A~F)

偏移地址：块 A：0x0A0
偏移地址：块 B：0x120
偏移地址：块 C：0x1A0
偏移地址：块 D：0x220
偏移地址：块 E：0x2A0
偏移地址：块 F：0x320
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | REPx[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP1x[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 23:16 **REPx[7:0]：**定时器 x 的重复值（HRTIM_REPx 寄存器中的别名）
此位域是 HRTIMx_REPxR 寄存器中 REPx[7:0]位域的别名。

Bit 15:0 **CMP1x[15:0]：**定时器 x 的比较值 1（Timer x compare 1 value）
此位域是 HRTIMx_CMP1xR 寄存器中 CMP1x[15:0]位域的别名。

### 21.5.21 HRTIM 定时器 x 比较 2 寄存器（HRTIM_CMP2xR) (x = A~F)

偏移地址：块 A：0x0A4
偏移地址：块 B：0x124
偏移地址：块 C：0x1A4
偏移地址：块 D：0x224
偏移地址：块 E：0x2A4
偏移地址：块 F：0x324
复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP2x[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **CMP2x[15:0]：**定时器 x 的比较 2 值（Timer x compare 2 value）
此寄存器保存比较 2 值。
该寄存器保存预装载寄存器的内容，如果禁止预装载，则会保存活动寄存器的内容。
比较值必须为 0 或大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0]=0，则为 0x60；如果 CKPSC[2:0]=1，则为 0x30；如果 CKPSC[2:0]=2，则为 0x18 以此类推。
如果通过 HRTIM_TIMxCR 中的 DELCMP2[1:0]位使能，此寄存器将作为自动延迟比较寄存器工作。

### 21.5.22 HRTIM 定时器 x 比较 3 寄存器（HRTIM_CMP3xR）(x = A~F)

偏移地址：块 A：0x0A8
偏移地址：块 B：0x128
偏移地址：块 C：0x1A8
偏移地址：块 D：0x228
偏移地址：块 E：0x2A8
偏移地址：块 F：0x328
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMP3x[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **CMP3x[15:0]：**定时器 x 的比较 3 值（Timer x compare 3 value）

此寄存器保存比较 3 值。

该寄存器保存预装载寄存器的内容，如果禁止预装载，则会保存活动寄存器的内容。

比较值必须为 0 或大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0]=0，则为 0x60；如果 CKPSC[2:0]=1，则为 0x30；如果 CKPSC[2:0]=2，则为 0x18 以此类推。

0 值的编程遵循“空占空比异常情况”一节中描述的用例。

### 21.5.23 HRTIM 定时器 x 比较 4 寄存器（HRTIM_CMP4xR）(x = A~F)

偏移地址：块 A：0x0AC
偏移地址：块 B：0x12C
偏移地址：块 C：0x1AC
偏移地址：块 D：0x22C
偏移地址：块 E：0x2AC
偏移地址：块 F：0x32C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resverd | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMP4x[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **CMP4x[15:0]：**定时器 x 的比较 4 值（Timer x compare 4 value）

此寄存器保存比较 4 值。

该寄存器保存预装载寄存器的内容，如果禁止预装载，则会保存活动寄存器的内容。

比较值必须为 0 或大于或等于 3 个 $f_{HRTIM}$ 时钟周期，即如果 CKPSC[2:0]=0，则为 0x60；如果 CKPSC[2:0]=1，则为 0x30；如果 CKPSC[2:0]=2，则为 0x18 以此类推。

如果通过 HRTIM_TIMxCR 中的 DELCMP4[1:0]位使能，此寄存器将作为自动延迟比较寄存器工作。

### 21.5.24 HRTIM 定时器 x 捕获 1 寄存器（HRTIM_CPT1xR）（x = A~F）

偏移地址：块 A：0x0B0
偏移地址：块 B：0x130
偏移地址：块 C：0x1B0
偏移地址：块 D：0x230
偏移地址：块 E：0x2B0
偏移地址：块 F：0x330
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | DIR |
| | | | | | | | | | | | | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPT1x[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:17 保留，必须保持复位值。

Bit 16 **DIR：**定时器 x 捕获 1 方向状态（Timer x capture 1 direction status）

此寄存器保存捕获 1 事件发生时的计数方向值：
0：定时器正在向上计数
1：定时器正在向下计数
在向上计数模式（UDM 位复位）下，DIR 位始终读取为 0。

Bit 15:0 **CPT1x[15:0]：**定时器 x 捕获 1 值（Timer x capture 1 value）

此寄存器保存捕获 1 事件发生时的计数器值。

*注意：* *在上下计数模式（UDM 位设置为1）下，捕获值是指：*
- *向上计数时的计数复位*
- *向下计数时的*PER *事件*

DIR *位允许在读取捕获值时区分上下计数阶段。*

*注意：* *这是一个常规分辨率寄存器：对于*HR *时钟预分频比低于*32 *（*CKPSC[2:0] < 5*）的情况，计数器的最低有效位不重要。它们无法写入，并且在读取时返回*0。

### 21.5.25 HRTIM 定时器 x 捕获 2 寄存器（HRTIM_CPT2xR）（x = A~F）

偏移地址：块 A：0x0B4
偏移地址：块 B：0x134
偏移地址：块 C：0x1B4
偏移地址：块 D：0x234
偏移地址：块 E：0x2B4
偏移地址：块 F：0x334
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | DIR |
| | | | | | | | | | | | | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPT2x[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:17 保留，必须保持复位值。

Bit 16 **DIR：**定时器 x 捕获 1 方向状态（Timer x capture 1 direction status）

此寄存器保存捕获 1 事件发生时的计数方向值：
0：定时器正在向上计数
1：定时器正在向下计数
在向上计数模式（UDM 位复位）下，DIR 位始终读取为 0。

Bit 15:0 **CPT2x[15:0]：**定时器 x 捕获 2 值（Timer x capture 2 value）

此寄存器保存捕获 2 事件发生时的计数器值。

*注意：* *在上下计数模式（UDM 位设置为1）下，捕获值是指：*
- *向上计数时的计数复位*
- *向下计数时的PER 事件*

DIR *位允许在读取捕获值时区分上下计数阶段。*

*注意：* *这是一个常规分辨率寄存器：对于HR 时钟预分频比低于32（CKPSC[2:0] < 5）的情况，计数器的最低有效位不重要。它们无法写入，并且在读取时返回0。*

### 21.5.26 HRTIM 定时器 x 死区时间寄存器（HRTIM_DTxR）（x = A~F）

偏移地址：块 A：0x0B8
偏移地址：块 B：0x138
偏移地址：块 C：0x1B8
偏移地址：块 D：0x238
偏移地址：块 E：0x2B8
偏移地址：块 F：0x338
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DTFLKx | DTFSLKx | Reserved | | | | SDTFx | DTFx[8:0] | | | | | | | | |
| rwo | rwo | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DTRLKx | DTRSLKx | Res | DTPRSC[1:0] | | | SDTRx | DTRx[8:0] | | | | | | | | |
| rwo | rwo | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **DTFLKx：**死区时间下降沿锁定（Deadtime falling lock）

若使能这一仅可写入一次的位，则会禁止修改死区（符号和值）。

0：可写入死区下降沿值和符号

1：只能读取死区下降沿值和符号

*注：此位不能预加载。*

Bit 30 **DTFSLKx：**死区时间下降沿符号锁定（Deadtime falling sign lock）

若使能这一仅可写入一次的位，则会禁止修改死区下降沿的符号。

0：可写入死区下降沿符号

1：只能读取死区下降沿符号

*注：此位不能预加载。*

Bit 29:26 保留，必须保持复位值。

Bit 25 **SDTFx：**死区时间下降沿符号值（Sign deadtime falling value）

该寄存器决定了死区为正值（信号不重叠）还是负值（信号重叠）。

0：死区在下降沿为正值

1：死区在下降沿为负值

Bit 24:16 **DTFx[8:0]：**死区时间下降沿值（Deadtime falling value）

该寄存器保存参考 PWM 信号下降沿之后的死区值。

$t_{DTF}$ = DTFx[8:0] x $t_{DTG}$

Bit 15 **DTRLKx：**死区时间上升沿锁定（Deadtime rising lock）

若使能这一仅可写入一次的位，则会禁止修改死区（符号和值）。

0：可写入死区上升沿值和符号

1：只能读取死区上升沿值和符号

*注：此位不能预加载。*

Bit 14 **DTRSLKx：**死区时间上升沿符号锁定（Deadtime rising sign lock）

若使能这一仅可写入一次的位，则会禁止修改死区上升沿的符号。

0：可写入死区上升沿符号

1：只能读取死区上升沿符号

*注：此位不能预加载。*

Bit 13 保留，必须保持复位值。

Bit 12:10 **DTPRSC[2:0]：**死区时间预分频器（Deadtime prescaler）

该寄存器保存死区时钟预分频器的值。

$t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$

（例如：000: 651 ps, 001: 1.302 ns,...）

如果使能了任何锁定位（DTFLKs，DTFSLKx，DTRLKx，DTRSLKx），此位域将变为只读。

Bit 9 **SDTRx：**死区时间上升沿符号值（Sign deadtime rising value）

该寄存器决定了死区为正值还是负值（信号重叠）。

0：死区在上升沿为正值

1：死区在上升沿为负值

Bit 8:0 **DTRx[2:0]：**死区时间上升沿值（Deadtime rising value）

该寄存器保存参考 PWM 信号上升沿之后的死区值。

$t_{DTR} = DTRx[8:0] \times t_{DTG}$

### 21.5.27 HRTIM 定时器 x 输出 1 置位寄存器（HRTIM_SETx1R）（x = A~F）

偏移地址：块 A：0x0BC
偏移地址：块 B：0x13C
偏移地址：块 C：0x1BC
偏移地址：块 D：0x23C
偏移地址：块 E：0x2BC
偏移地址：块 F：0x33C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UPDATE | EXTEVNT10 | EXTEVNT9 | EXTEVNT8 | EXTEVNT7 | EXTEVNT6 | EXTEVNT5 | EXTEVNT4 | EXTEVNT3 | EXTEVNT2 | EXTEVNT1 | TIMEVNT9 | TIMEVNT8 | TIMEVNT7 | TIMEVNT6 | TIMEVNT5 |
| 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 |
| TIMEVNT4 | TIMEVNT3 | TIMEVNT2 | TIMEVNT1 | MSTCMP4 | MSTCMP3 | MSTCMP2 | MSTCMP1 | MSTPER | CMP4 | CMP3 | CMP2 | CMP1 | PER | RESYNC | SST |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **UPDATE**：寄存器更新（Registers update）（从预装载寄存器传输到活动寄存器）
寄存器更新事件会强制将输出设为其有效状态。

Bit 30 **EXTEVNT10**：外部事件 10（External event 10）
参考 EXTEVNT1 的描述。

Bit 29 **EXTEVNT9**：外部事件 9（External event 9）
参考 EXTEVNT1 的描述。

Bit 28 **EXTEVNT8**：外部事件 8（External event 8）
参考 EXTEVNT1 的描述。

Bit 27 **EXTEVNT7**：外部事件 7（External event 7）
参考 EXTEVNT1 的描述。

Bit 26 **EXTEVNT6**：外部事件 6（External event 6）
参考 EXTEVNT1 的描述。

Bit 25 **EXTEVNT5**：外部事件 5（External event 5）
参考 EXTEVNT1 的描述。

Bit 24 **EXTEVNT4**：外部事件 4（External event 4）
参考 EXTEVNT1 的描述。

Bit 23 **EXTEVNT3**：外部事件 3（External event 3）
参考 EXTEVNT1 的描述。

Bit 22 **EXTEVNT2**：外部事件 2（External event 2）
参考 EXTEVNT1 的描述。

Bit 21 **EXTEVNT1**：外部事件 1（External event 1）
外部事件 1 会强制将输出设为其有效状态。

Bit 20 **TIMEVNT9**：定时器事件 9（Timer event 9）
参考 TIMEVNT1 的描述。

Bit 19 **TIMEVNT8**：定时器事件 8（Timer event 8）
参考 TIMEVNT1 的描述。

Bit 18 **TIMEVNT7**：定时器事件 7（Timer event 7）
参考 TIMEVNT1 的描述。

Bit 17 **TIMEVNT6**：定时器事件 6（Timer event 6）
参考 TIMEVNT1 的描述。

Bit 16 **TIMEVNT5**：定时器事件 5（Timer event 5）
参考 TIMEVNT1 的描述。

Bit 15 **TIMEVNT4**：定时器事件 4（Timer event 4）
参考 TIMEVNT1 的描述。

Bit 14 **TIMEVNT3**：定时器事件 3（Timer event 3）
参考 TIMEVNT1 的描述。

Bit 13 **TIMEVNT2**：定时器事件 2（Timer event 2）
参考 TIMEVNT1 的描述。

Bit 12 **TIMEVNT1**：定时器事件 1（Timer event 1）
定时器事件 1 会强制将输出设为其有效状态

Bit 11 **MSTCMP4**：主比较 4（Master compare 4）
主定时器比较 4 事件会强制将输出设为其有效状态。

Bit 10 **MSTCMP3**：主比较 3（Master compare 3）
主定时器比较 3 事件会强制将输出设为其有效状态。

Bit 9 **MSTCMP2**：主比较 2（Master compare 2）
主定时器比较 2 事件会强制将输出设为其有效状态。

Bit 8 **MSTCMP1**：主比较 1（Master compare 1）
主定时器比较 1 事件会强制将输出设为其有效状态。

Bit 7 **MSTPER**：主周期（Master period）
主定时器计数器在连续模式下翻转或主定时器在单发模式下复位都会强制将输出设为其有效状态。

Bit 6 **CMP4**：定时器 x 比较 4（Timer x compare 4）
定时器 A 比较 4 事件会强制将输出设为其有效状态。

Bit 5 **CMP3**：定时器 x 比较 3（Timer x compare 3）
定时器 A 比较 3 事件会强制将输出设为其有效状态。

Bit 4 **CMP2**：定时器 x 比较 2（Timer x compare 2）
定时器 A 比较 2 事件会强制将输出设为其有效状态。

Bit 3 **CMP1**：定时器 x 比较 1（Timer x compare 1）
定时器 A 比较 1 事件会强制将输出设为其有效状态。

Bit 2 **PER**：定时器 x 周期（Timer x period）

定时器 A 周期事件会强制将输出设为其有效状态。

*注：在上下模式（*UDM *位设置为*1*）中，计数器周期事件根据*OUTROM[1:0]*位的设置来定义。*

Bit 1 **RESYNC**：定时器 A 重新同步（Timer A resynchronization）

只有来自软件或 SYNC 输入的定时器 A 复位事件才会强制将输出设为其有效状态。

*注：当*RESYNC=1 *时，其他定时器复位不会影响输出。*

Bit 0 **SST**：软件设置触发器（Software set trigger）

该位会强制将输出设为其有效状态。该位仅可通过软件置 1，并由硬件复位。

*注：此位不是预加载的。*

### 21.5.28 HRTIM 定时器 x 输出 1 复位寄存器（HRTIM_RSTx1R）（x = A~F）

偏移地址：块 A：0x0C0
偏移地址：块 B：0x140
偏移地址：块 C：0x1C0
偏移地址：块 D：0x240
偏移地址：块 E：0x2C0
偏移地址：块 F：0x340
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UP DATE | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 | EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | TIM EVNT9 | TIM EVNT8 | TIM EVNT7 | TIM EVNT6 | TIM EVNT5 |
| 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 |
| TIM EVNT4 | TIM EVNT3 | TIM EVNT2 | TIM EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP3 | CMP2 | CMP1 | PER | RESYN C | SRT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 请参考 HRTIM_SETx1R 位的描述。
这些位定义了可强制将 Tx1 输出设为其无效状态的源。

### 21.5.29 HRTIM 定时器 x 输出 2 置位寄存器（HRTIM_SETx2R）（x = A~F）

偏移地址：块 A：0x0C4
偏移地址：块 B：0x144
偏移地址：块 C：0x1C4
偏移地址：块 D：0x244
偏移地址：块 E：0x2C4
偏移地址：块 F：0x344
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UP DATE | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 | EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | TIM EVNT9 | TIM EVNT8 | TIM EVNT7 | TIM EVNT6 | TIM EVNT5 |
| 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 |
| TIM EVNT4 | TIM EVNT3 | TIM EVNT2 | TIM EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP3 | CMP2 | CMP1 | PER | RESYN C | SST |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 请参考 HRTIM_SETx1R 位的描述。
这些位定义了可强制将 Tx2 输出设为其无效状态的源。

### 21.5.30 HRTIM 定时器 x 输出 2 复位寄存器（HRTIM_RSTx2R）（x = A~F）

偏移地址：块 A：0x0C8
偏移地址：块 B：0x148
偏移地址：块 C：0x1C8
偏移地址：块 D：0x248
偏移地址：块 E：0x2C8
偏移地址：块 F：0x348
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UP DATE | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 | EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | TIM EVNT9 | TIM EVNT8 | TIM EVNT7 | TIM EVNT6 | TIM EVNT5 |
| 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 |
| TIM EVNT4 | TIM EVNT3 | TIM EVNT2 | TIM EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP3 | CMP2 | CMP1 | PER | RESYN C | SRT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 请参考 HRTIM_SETx1R 位的描述。
这些位定义了可强制将 Tx2 输出设为其无效状态的源。

### 21.5.31 HRTIM 定时器 x 外部事件过滤寄存器 1 (HRTIM_EEFxR1) (x = A~F)

偏移地址：块 A：0x0CC
偏移地址：块 B：0x14C
偏移地址：块 C：0x1CC
偏移地址：块 D：0x24C
偏移地址：块 E：0x2CC
偏移地址：块 F：0x34C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | EE5FLTR[3:0] | | | | EE5LTCH | Res | EE4FLTR[3:0] | | | | EE4LTCH | Res | EE3FLTR[3] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EE3FLTR[2:0] | | | EE3LTCH | Res | EE2FLTR[3:0] | | | | EE2LTCH | Res | EE1FLTR[3:0] | | | | EE1LTCH |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持复位值。

Bit 28:25 **EE5FLTR[3:0]**：外部事件 5 过滤器（External event 5 filter）
参考 EE1FLTR[3:0]的描述。

Bit 24 **EE5LTCH**：外部事件 5 锁存（External event 5 latch）
参考 EE1LTCH 的描述

Bit 23 保留，必须保持复位值。

Bit 22:19 **EE4FLTR[3:0]**：外部事件 4 过滤器（External event 4 filter）
参考 EE1FLTR[3:0]的描述。

Bit 18 **EE4LTCH**：外部事件 4 锁存（: External event 4 latch）
参考 EE1LTCH 的描述

Bit 17 保留，必须保持复位值。

Bit 16:13 **EE3FLTR[3:0]**：外部事件 3 过滤器（External event 3 filter）
参考 EE1FLTR[3:0]的描述。

Bit 12 **EE3LTCH**：外部事件 3 锁存（External event 3 latch）
参考 EE1LTCH 的描述

Bit 11 保留，必须保持复位值。

Bit 10:7 **EE2FLTR[3:0]**：外部事件 2 过滤器（External event 2 filter）
参考 EE1FLTR[3:0]的描述。

Bit 6 **EE2LTCH**：外部事件 2 锁存（External event 2 latch）
参考 EE1LTCH 的描述。

Bit 5 保留，必须保持复位值。

Bit 4:1 **EE1FLTR[3:0]**：外部事件 1 过滤器（External event 1 filter）

0000：无滤波

0001：消隐持续时间从计数器复位/翻转到比较 1

0010：向上计数模式（UDM 位复位），消隐持续时间从计数器复位/翻转到比较 2
上下计数模式（UDM 位置位）：仅在向上计数阶段从比较 1 到比较 2 的消隐。

0011：消隐持续时间从计数器复位/翻转到比较 3

0100：向上计数模式（UDM 位复位），消隐持续时间从计数器复位/翻转到比较 4
上下计数模式（UDM 位置位）：仅在向上计数阶段从比较 3 到比较 4 的消隐。

0101：消隐与另一定时单元相关：TIMFLTR1 源

0110：消隐与另一定时单元相关：TIMFLTR2 源

0111：消隐与另一定时单元相关：TIMFLTR3 源

1000：消隐与另一定时单元相关：TIMFLTR4 源

1001：消隐与另一定时单元相关：TIMFLTR5 源

1010：消隐与另一定时单元相关：TIMFLTR6 源

1011：消隐与另一定时单元相关：TIMFLTR7 源

1100：消隐与另一定时单元相关：TIMFLTR8 源

1101：向上计数模式（UDM 位复位），窗口持续时间从计数器复位/翻转到比较 2
上下计数模式（UDM 位置位）：窗口持续时间仅在向上计数阶段从比较 2 到比较 3

1110：向上计数模式（UDM 位复位），窗口持续时间从计数器复位/翻转到比较 3
上下计数模式（UDM 位置位）：窗口持续时间仅在向上计数阶段从比较 2 到比较 3

1111：向上计数模式（UDM 位复位），窗口与另一定时单元相关：TIMWIN 源
上下计数模式（UDM 位置位）：窗口持续时间从向上计数阶段的比较 2 到向下计数阶段的比较 3。

*注：每当使用比较寄存器进行滤波时，其值必须大于 0。*
*一旦计数器被使能（TxCEN 位设置），则不应修改此位域。*

Bit 0 **EE1LTCH**：外部事件 1 锁存（External event 1 latch）

0：如果事件 1 在消隐期间发生，会被忽略，如果在窗口持续时间内发生，则会通过。

1：事件 1 会锁存并延迟，直至消隐或窗口周期结束。

*注：如果 EE1LTCH = 0，则会在窗口模式下生成超时事件（EE1FLTR[3:0]=1101、1110、1111），但外部事件在快速模式下编程的情况除外 (EExFAST = 1)。*
*一旦计数器被使能（TxCEN 位置1），则不应修改此位域。*

### 21.5.32 HRTIM 定时器 x 外部事件过滤寄存器 2 (HRTIM_EEFxR2) (x = A~F)

偏移地址：块 A：0x0D0
偏移地址：块 B：0x150
偏移地址：块 C：0x1D0
偏移地址：块 D：0x250
偏移地址：块 E：0x2D0
偏移地址：块 F：0x350
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | EE10FLTR[3:0] | | | | EE10LTCH | Res | EE9FLTR[3:0] | | | | EE9LTCH | Res | EE8FLTR[3] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EE8FLTR[2:0] | | | EE8LTCH | Res | EE7FLTR[3:0] | | | | EE7LTCH | Res | EE6FLTR[3:0] | | | | EE6LTCH |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持复位值。

Bit 28:25 **EE10FLTR[3:0]**：外部事件 10 过滤器（External event 10 filter）
参考 EE1FLTR[3:0]的描述。

Bit 24 **EE10LTCH**：外部事件 10 锁存（External event 10 latch）
参考 EE1LTCH 的描述

Bit 23 保留，必须保持复位值。

Bit 22:19 **EE9FLTR[3:0]**：外部事件 9 过滤器（External event 9 filter）
参考 EE1FLTR[3:0]的描述。

Bit 18 **EE9LTCH**：外部事件 9 锁存（: External event 9 latch）
参考 EE1LTCH 的描述

Bit 17 保留，必须保持复位值。

Bit 16:13 **EE8FLTR[3:0]**：外部事件 8 过滤器（External event 8 filter）
参考 EE1FLTR[3:0]的描述。

Bit 12 **EE8LTCH**：外部事件 8 锁存（External event 8 latch）
参考 EE1LTCH 的描述

Bit 11 保留，必须保持复位值。

Bit 10:7 **EE7FLTR[3:0]**：外部事件 7 过滤器（External event 7 filter）
参考 EE1FLTR[3:0]的描述。

Bit 6 **EE7LTCH**：外部事件 7 锁存（External event 7 latch）
参考 EE1LTCH 的描述。

Bit 5 保留，必须保持复位值。

Bit 4:1 **EE6FLTR[3:0]**：外部事件 6 过滤器（External event 6 filter）
参考 EE1FLTR[3:0]的描述。

Bit 0 **EE6LTCH**：外部事件 6 锁存（External event 6 latch）
参考 EE1LTCH 的描述。

### 21.5.33 HRTIM 定时器 A 复位寄存器（HRTIM_RSTAR）

偏移地址：0x0D4
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMF CMP2 | TIME CMP4 | TIME CMP2 | TIME CMP1 | TIMD CMP4 | TIMD CMP2 | TIMD CMP1 | TIMC CMP4 | TIMC CMP2 | TIMC CMP1 | TIMB CMP4 | TIMB CMP2 | TIMB CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIMF CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMFCPM2**：定时器 F 比较 2（Timer F compare 2）
当定时器 F 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 30 **TIMECPM4**：定时器 E 比较 4（Timer E compare 4）
当定时器 E 的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 29 **TIMECMP2**：定时器 E 比较 2（Timer E compare 2）
当定时器 E 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 28 **TIMECMP1**：定时器 E 比较 1（Timer E compare 1）
当定时器 E 的比较 1 事件发生时，定时器 A 的计数器将被复位。

Bit 27 **TIMDCMP4**：定时器 D 比较 4（Timer D compare 4）
当定时器 D 的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 26 **TIMDCMP2**：定时器 D 比较 2（Timer D compare 2）
当定时器 D 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 25 **TIMDCMP1**：定时器 D 比较 1（Timer D compare 1）
当定时器 D 的比较 1 事件发生时，定时器 A 的计数器将被复位。

Bit 24 **TIMCCMP4**：定时器 C 比较 4（Timer C compare 4）
当定时器 C 的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 23 **TIMCCMP2**：定时器 C 比较 2（Timer C compare 2）
当定时器 C 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 22 **TIMCCMP1**：定时器 C 比较 1（Timer C compare 1）
当定时器 C 的比较 1 事件发生时，定时器 A 的计数器将被复位。

Bit 21 **TIMBCMP4**：定时器 B 比较 4（Timer B compare 4）
当定时器 B 的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 20 **TIMBCMP2**：定时器 B 比较 2（Timer B compare 2）
当定时器 B 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 19 **TIMBCMP1**：定时器 B 比较 1（Timer B compare 1）
当定时器 B 的比较 1 事件发生时，定时器 A 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event）
当外部事件 10 发生时，定时器 A 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）
当外部事件 9 发生时，定时器 A 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 A 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 A 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 A 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 A 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 A 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 A 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 A 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 A 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 A 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 A 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 A 的计数器将被复位。

Bit 3 **CMP4**：定时器 A 比较 4 复位（Timer A compare 4 reset）
当定时器 A 的比较 4 事件发生时，定时器 A 的计数器将被复位。

Bit 2 **CMP2**：定时器 A 比较 2 复位（Timer A compare 2 reset）
当定时器 A 的比较 2 事件发生时，定时器 A 的计数器将被复位。

Bit 1 **UPDT**：定时器 A 更新复位（Timer A update reset）
当更新事件发生时，定时器 A 的计数器将被复位。

Bit 0 **TIMFCMP1**：定时器 F 比较 1（Timer F compare 1）
当定时器 F 的比较 1 事件发生时，定时器 A 的计数器将被复位。

### 21.5.34 HRTIM 定时器 B 复位寄存器（HRTIM_RSTBR）

偏移地址：0x154

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMF CMP2 | TIME CMP4 | TIME CMP2 | TIME CMP1 | TIMD CMP4 | TIMD CMP2 | TIMD CMP1 | TIMC CMP4 | TIMC CMP2 | TIMC CMP1 | TIMA CMP4 | TIMA CMP2 | TIMA CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIMF CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMFCPM2**：定时器 F 比较 2（Timer F compare 2）

当定时器 F 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 30 **TIMECPM4**：定时器 E 比较 4（Timer E compare 4）

当定时器 E 的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 29 **TIMECMP2**：定时器 E 比较 2（Timer E compare 2）

当定时器 E 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 28 **TIMECMP1**：定时器 E 比较 1（Timer E compare 1）

当定时器 E 的比较 1 事件发生时，定时器 B 的计数器将被复位。

Bit 27 **TIMDCMP4**：定时器 D 比较 4（Timer D compare 4）

当定时器 D 的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 26 **TIMDCMP2**：定时器 D 比较 2（Timer D compare 2）

当定时器 D 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 25 **TIMDCMP1**：定时器 D 比较 1（Timer D compare 1）

当定时器 D 的比较 1 事件发生时，定时器 B 的计数器将被复位。

Bit 24 **TIMCCMP4**：定时器 C 比较 4（Timer C compare 4）

当定时器 C 的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 23 **TIMCCMP2**：定时器 C 比较 2（Timer C compare 2）

当定时器 C 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 22 **TIMCCMP1**：定时器 C 比较 1（Timer C compare 1）

当定时器 C 的比较 1 事件发生时，定时器 B 的计数器将被复位。

Bit 21 **TIMACMP4**：定时器 A 比较 4（Timer A compare 4）

当定时器 A 的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 20 **TIMACMP2**：定时器 A 比较 2（Timer A compare 2）

当定时器 A 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 19 **TIMACMP1**：定时器 A 比较 1（Timer A compare 1）

当定时器 A 的比较 1 事件发生时，定时器 B 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event 10）

当外部事件 10 发生时，定时器 B 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）

当外部事件 9 发生时，定时器 B 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 B 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 B 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 B 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 B 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 B 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 B 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 B 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 B 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 B 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 B 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 B 的计数器将被复位。

Bit 3 **CMP4**：定时器 B 比较 4 复位（Timer B compare 4 reset）
当定时器 B 的比较 4 事件发生时，定时器 B 的计数器将被复位。

Bit 2 **CMP2**：定时器 B 比较 2 复位（Timer B compare 2 reset）
当定时器 B 的比较 2 事件发生时，定时器 B 的计数器将被复位。

Bit 1 **UPDT**：定时器 B 更新复位（Timer B update reset）
当更新事件发生时，定时器 B 的计数器将被复位。

Bit 0 **TIMFCMP1**：定时器 F 比较 1（Timer F compare 1）
当定时器 F 的比较 1 事件发生时，定时器 B 的计数器将被复位。

### 21.5.35 HRTIM 定时器 C 复位寄存器（HRTIM_RSTCR）

偏移地址：0x2D4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMF CMP2 | TIME CMP4 | TIME CMP2 | TIME CMP1 | TIMD CMP4 | TIMD CMP2 | TIMD CMP1 | TIMB CMP4 | TIMB CMP2 | TIMB CMP1 | TIMA CMP4 | TIMA CMP2 | TIMA CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIMF CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMFCPM2**：定时器 F 比较 2（Timer F compare 2）

当定时器 F 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 30 **TIMECPM4**：定时器 E 比较 4（Timer E compare 4）

当定时器 E 的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 29 **TIMECMP2**：定时器 E 比较 2（Timer E compare 2）

当定时器 E 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 28 **TIMECMP1**：定时器 E 比较 1（Timer E compare 1）

当定时器 E 的比较 1 事件发生时，定时器 C 的计数器将被复位。

Bit 27 **TIMDCMP4**：定时器 D 比较 4（Timer D compare 4）

当定时器 D 的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 26 **TIMDCMP2**：定时器 D 比较 2（Timer D compare 2）

当定时器 D 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 25 **TIMDCMP1**：定时器 D 比较 1（Timer D compare 1）

当定时器 D 的比较 1 事件发生时，定时器 C 的计数器将被复位。

Bit 24 **TIMBCMP4**：定时器 B 比较 4（Timer B compare 4）

当定时器 B 的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 23 **TIMBCMP2**：定时器 B 比较 2（Timer B compare 2）

当定时器 B 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 22 **TIMBCMP1**：定时器 B 比较 1（Timer B compare 1）

当定时器 B 的比较 1 事件发生时，定时器 C 的计数器将被复位。

Bit 21 **TIMACMP4**：定时器 A 比较 4（Timer A compare 4）

当定时器 A 的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 20 **TIMACMP2**：定时器 A 比较 2（Timer A compare 2）

当定时器 A 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 19 **TIMACMP1**：定时器 A 比较 1（Timer A compare 1）

当定时器 A 的比较 1 事件发生时，定时器 C 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event 10）

当外部事件 10 发生时，定时器 C 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）

当外部事件 9 发生时，定时器 C 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 C 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 C 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 C 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 C 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 C 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 C 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 C 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 C 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 C 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 C 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 C 的计数器将被复位。

Bit 3 **CMP4**：定时器 C 比较 4 复位（Timer C compare 4 reset）
当定时器 C 的比较 4 事件发生时，定时器 C 的计数器将被复位。

Bit 2 **CMP2**：定时器 C 比较 2 复位（Timer C compare 2 reset）
当定时器 C 的比较 2 事件发生时，定时器 C 的计数器将被复位。

Bit 1 **UPDT**：定时器 C 更新复位（Timer C update reset）
当更新事件发生时，定时器 C 的计数器将被复位。

Bit 0 **TIMFCMP1**：定时器 F 比较 1（Timer F compare 1）
当定时器 F 的比较 1 事件发生时，定时器 C 的计数器将被复位。

### 21.5.36 HRTIM 定时器 D 复位寄存器（HRTIM_RSTDR）

偏移地址：0x254

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMF CMP2 | TIME CMP4 | TIME CMP2 | TIME CMP1 | TIMC CMP4 | TIMC CMP2 | TIMC CMP1 | TIMB CMP4 | TIMB CMP2 | TIMB CMP1 | TIMA CMP4 | TIMA CMP2 | TIMA CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIMF CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMFCPM2**：定时器 F 比较 2（Timer F compare 2）

当定时器 F 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 30 **TIMECPM4**：定时器 E 比较 4（Timer E compare 4）

当定时器 E 的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 29 **TIMECMP2**：定时器 E 比较 2（Timer E compare 2）

当定时器 E 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 28 **TIMECMP1**：定时器 E 比较 1（Timer E compare 1）

当定时器 E 的比较 1 事件发生时，定时器 D 的计数器将被复位。

Bit 27 **TIMCCMP4**：定时器 C 比较 4（Timer C compare 4）

当定时器 C 的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 26 **TIMCCMP2**：定时器 C 比较 2（Timer C compare 2）

当定时器 C 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 25 **TIMCCMP1**：定时器 C 比较 1（Timer C compare 1）

当定时器 C 的比较 1 事件发生时，定时器 D 的计数器将被复位。

Bit 24 **TIMBCMP4**：定时器 B 比较 4（Timer B compare 4）

当定时器 B 的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 23 **TIMBCMP2**：定时器 B 比较 2（Timer B compare 2）

当定时器 B 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 22 **TIMBCMP1**：定时器 B 比较 1（Timer B compare 1）

当定时器 B 的比较 1 事件发生时，定时器 D 的计数器将被复位。

Bit 21 **TIMACMP4**：定时器 A 比较 4（Timer A compare 4）

当定时器 A 的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 20 **TIMACMP2**：定时器 A 比较 2（Timer A compare 2）

当定时器 A 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 19 **TIMACMP1**：定时器 A 比较 1（Timer A compare 1）

当定时器 A 的比较 1 事件发生时，定时器 D 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event 10）

当外部事件 10 发生时，定时器 D 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）

当外部事件 9 发生时，定时器 D 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 D 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 D 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 D 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 D 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 D 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 D 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 D 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 D 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 D 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 D 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 D 的计数器将被复位。

Bit 3 **CMP4**：定时器 D 比较 4 复位（Timer D compare 4 reset）
当定时器 D 的比较 4 事件发生时，定时器 D 的计数器将被复位。

Bit 2 **CMP2**：定时器 D 比较 2 复位（Timer D compare 2 reset）
当定时器 D 的比较 2 事件发生时，定时器 D 的计数器将被复位。

Bit 1 **UPDT**：定时器 D 更新复位（Timer D update reset）
当更新事件发生时，定时器 D 的计数器将被复位。

Bit 0 **TIMFCMP1**：定时器 F 比较 1（Timer F compare 1）
当定时器 F 的比较 1 事件发生时，定时器 D 的计数器将被复位。

### 21.5.37 HRTIM 定时器 E 复位寄存器（HRTIM_RSTER）

偏移地址：0x2D4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMF CMP2 | TIMD CMP4 | TIMD CMP2 | TIMD CMP1 | TIMC CMP4 | TIMC CMP2 | TIMC CMP1 | TIMB CMP4 | TIMB CMP2 | TIMB CMP1 | TIMA CMP4 | TIMA CMP2 | TIMA CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIMF CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMFCPM2**：定时器 F 比较 2（Timer F compare 2）

当定时器 F 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 30 **TIMDCPM4**：定时器 D 比较 4（Timer D compare 4）

当定时器 D 的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 29 **TIMDCMP2**：定时器 D 比较 2（Timer D compare 2）

当定时器 D 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 28 **TIMDCMP1**：定时器 D 比较 1（Timer D compare 1）

当定时器 D 的比较 1 事件发生时，定时器 E 的计数器将被复位。

Bit 27 **TIMCCMP4**：定时器 C 比较 4（Timer C compare 4）

当定时器 C 的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 26 **TIMCCMP2**：定时器 C 比较 2（Timer C compare 2）

当定时器 C 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 25 **TIMCCMP1**：定时器 C 比较 1（Timer C compare 1）

当定时器 C 的比较 1 事件发生时，定时器 E 的计数器将被复位。

Bit 24 **TIMBCMP4**：定时器 B 比较 4（Timer B compare 4）

当定时器 B 的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 23 **TIMBCMP2**：定时器 B 比较 2（Timer B compare 2）

当定时器 B 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 22 **TIMBCMP1**：定时器 B 比较 1（Timer B compare 1）

当定时器 B 的比较 1 事件发生时，定时器 E 的计数器将被复位。

Bit 21 **TIMACMP4**：定时器 A 比较 4（Timer A compare 4）

当定时器 A 的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 20 **TIMACMP2**：定时器 A 比较 2（Timer A compare 2）

当定时器 A 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 19 **TIMACMP1**：定时器 A 比较 1（Timer A compare 1）

当定时器 A 的比较 1 事件发生时，定时器 E 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event 10）

当外部事件 10 发生时，定时器 E 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）

当外部事件 9 发生时，定时器 E 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 E 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 E 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 E 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 E 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 E 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 E 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 E 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 E 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 E 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 E 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 E 的计数器将被复位。

Bit 3 **CMP4**：定时器 E 比较 4 复位（Timer E compare 4 reset）
当定时器 E 的比较 4 事件发生时，定时器 E 的计数器将被复位。

Bit 2 **CMP2**：定时器 E 比较 2 复位（Timer E compare 2 reset）
当定时器 E 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 1 **UPDT**：定时器 E 更新复位（Timer E update reset）
当更新事件发生时，定时器 E 的计数器将被复位。

Bit 0 **TIMFCMP1**：定时器 F 比较 1（Timer F compare 1）
当定时器 F 的比较 1 事件发生时，定时器 E 的计数器将被复位。

### 21.5.38 HRTIM 定时器 F 复位寄存器（HRTIM_RSTFR）

偏移地址：0x354

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME CMP2 | TIMD CMP4 | TIMD CMP2 | TIMD CMP1 | TIMC CMP4 | TIMC CMP2 | TIMC CMP1 | TIMB CMP4 | TIMB CMP2 | TIMB CMP1 | TIMA CMP4 | TIMA CMP2 | TIMA CMP1 | EXT EVNT10 | EXT EVNT9 | EXT EVNT8 |
| 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 |
| EXT EVNT7 | EXT EVNT6 | EXT EVNT5 | EXT EVNT4 | EXT EVNT3 | EXT EVNT2 | EXT EVNT1 | MST CMP4 | MST CMP3 | MST CMP2 | MST CMP1 | MST PER | CMP4 | CMP2 | UPDT | TIME CMP1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **TIMECPM2**：定时器 E 比较 2（Timer E compare 2）

当定时器 F 的比较 2 事件发生时，定时器 E 的计数器将被复位。

Bit 30 **TIMDCPM4**：定时器 D 比较 4（Timer D compare 4）

当定时器 D 的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 29 **TIMDCMP2**：定时器 D 比较 2（Timer D compare 2）

当定时器 D 的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 28 **TIMDCMP1**：定时器 D 比较 1（Timer D compare 1）

当定时器 D 的比较 1 事件发生时，定时器 F 的计数器将被复位。

Bit 27 **TIMCCMP4**：定时器 C 比较 4（Timer C compare 4）

当定时器 C 的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 26 **TIMCCMP2**：定时器 C 比较 2（Timer C compare 2）

当定时器 C 的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 25 **TIMCCMP1**：定时器 C 比较 1（Timer C compare 1）

当定时器 C 的比较 1 事件发生时，定时器 F 的计数器将被复位。

Bit 24 **TIMBCMP4**：定时器 B 比较 4（Timer B compare 4）

当定时器 B 的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 23 **TIMBCMP2**：定时器 B 比较 2（Timer B compare 2）

当定时器 B 的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 22 **TIMBCMP1**：定时器 B 比较 1（Timer B compare 1）

当定时器 B 的比较 1 事件发生时，定时器 F 的计数器将被复位。

Bit 21 **TIMACMP4**：定时器 A 比较 4（Timer A compare 4）

当定时器 A 的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 20 **TIMACMP2**：定时器 A 比较 2（Timer A compare 2）

当定时器 A 的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 19 **TIMACMP1**：定时器 A 比较 1（Timer A compare 1）

当定时器 A 的比较 1 事件发生时，定时器 F 的计数器将被复位。

Bit 18 **EXTEVNT10**：外部事件 10（External event 10）

当外部事件 10 发生时，定时器 F 的计数器将被复位。

Bit 17 **EXTEVNT9**：外部事件 9（External event 9）

当外部事件 9 发生时，定时器 F 的计数器将被复位。

Bit 16 **EXTEVNT8**：外部事件 8（External event 8）
当外部事件 8 发生时，定时器 F 的计数器将被复位。

Bit 15 **EXTEVNT7**：外部事件 7（External event 7）
当外部事件 7 发生时，定时器 F 的计数器将被复位。

Bit 14 **EXTEVNT6**：外部事件 6（External event 6）
当外部事件 6 发生时，定时器 F 的计数器将被复位。

Bit 13 **EXTEVNT5**：外部事件 5（External event 5）
当外部事件 5 发生时，定时器 F 的计数器将被复位。

Bit 12 **EXTEVNT4**：外部事件 4（External event 4）
当外部事件 4 发生时，定时器 F 的计数器将被复位。

Bit 11 **EXTEVNT3**：外部事件 3（External event 3）
当外部事件 3 发生时，定时器 F 的计数器将被复位。

Bit 10 **EXTEVNT2**：外部事件 2（External event 2）
当外部事件 2 发生时，定时器 F 的计数器将被复位。

Bit 9 **EXTEVNT1**：外部事件 1（External event 1）
当外部事件 1 发生时，定时器 F 的计数器将被复位。

Bit 8 **MSTCMP4**：主定时器比较 4（Master compare 4）
当主定时器的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 7 **MSTCMP3**：主定时器比较 3（Master compare 3）
当主定时器的比较 3 事件发生时，定时器 F 的计数器将被复位。

Bit 6 **MSTCMP2**：主定时器比较 2（Master compare 2）
当主定时器的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 5 **MSTCMP1**：主定时器比较 1（Master compare 1）
当主定时器的比较 1 事件发生时，定时器 F 的计数器将被复位。

Bit 4 **MSTPER**：主定时器周期（Master timer period）
当主定时器的周期事件发生时，定时器 F 的计数器将被复位。

Bit 3 **CMP4**：定时器 F 比较 4 复位（Timer F compare 4 reset）
当定时器 F 的比较 4 事件发生时，定时器 F 的计数器将被复位。

Bit 2 **CMP2**：定时器 F 比较 2 复位（Timer F compare 2 reset）
当定时器 F 的比较 2 事件发生时，定时器 F 的计数器将被复位。

Bit 1 **UPDT**：定时器 F 更新复位（Timer F update reset）
当更新事件发生时，定时器 F 的计数器将被复位。

Bit 0 **TIMECMP1**：定时器 E 比较 1（Timer E compare 1）
当定时器 E 的比较 1 事件发生时，定时器 F 的计数器将被复位。

### 21.5.39 HRTIM 定时器 x 斩波寄存器（HRTIM_CHPxR）（x = A~F）

偏移地址：块 A：0x0D8
偏移地址：块 B：0x158
偏移地址：块 C：0x1D8
偏移地址：块 D：0x258
偏移地址：块 E：0x2D8
偏移地址：块 F：0x358
复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | STRTPW[3:0] | | | | CARDTY[2:0] | | | CARFRQ[3:0] | | | |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:11 保留，必须保持复位值。

Bit 10:7 **STRPW[3:0]**：定时器 x 启动脉冲宽度（Timer x start pulsewidth）
该寄存器定义输出信号上升沿后的初始脉宽。
如果其中一个 CHPx 位置 1，此位域不能修改。
$t_{1STPW}$ = (STRPW[3:0]+1) x 16 x $t_{HRTIM}$。

0000：83.3 ns（1/12 MHz）
...
1111：1.33 µs（16/12 MHz）

Bit 6:4 **CARDTY[2:0]**：定时器 x 斩波占空比值（Timer x chopper duty cycle value）
该寄存器定义载波信号的占空比。如果其中一个 CHPx 位置 1，此位域不能修改。
000：0/8（即，只会出现第一个脉冲）
...
111：7/8

Bit 3:0 **CARFRQ[3:0]**：定时器 x 载波频率值（Timer x carrier frequency value）
此寄存器定义了载波频率 $F_{CHPFRQ}$ = $f_{HRTIM}$ / (16 x (CARFRQ[3:0]+1))。
如果其中一个 CHPx 位置 1，此位域不能修改。
0000：12 MHz（$f_{HRTIM}$ / 16）
...
1111：750 KHz（$f_{HRTIM}$ / 256）

### 21.5.40 HRTIM 定时器 A 捕获 1 控制寄存器（HRTIM_CPT1ACR）

偏移地址：0x0DC

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TE CMP2 | TE CMP1 | TE1 RST | TE1 SET | TD CMP2 | TD CMP1 | TD1 RST | TD1 SET | TC CMP2 | TC CMP1 | TC1 RST | TC1 SET | TB CMP2 | TB CMP1 | TB1 RST | TB1 SET |
| 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 |
| TF CMP2 | TF CMP1 | TF1 RST | TF1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.41 HRTIM 定时器 B 捕获 1 控制寄存器（HRTIM_CPT1BCR）

偏移地址：0x15C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TE CMP2 | TE CMP1 | TE1 RST | TE1 SET | TD CMP2 | TD CMP1 | TD1 RST | TD1 SET | TC CMP2 | TC CMP1 | TC1 RST | TC1 SET | TF CMP2 | TF CMP1 | TF1 RST | TF1 SET |
| 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 |
| TA CMP2 | TA CMP1 | TA1 RST | TA1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.42 HRTIM 定时器 C 捕获 1 控制寄存器（HRTIM_CPT1CCR）

偏移地址：0x1DC

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TE CMP2 | TE CMP1 | TE1 RST | TE1 SET | TD CMP2 | TD CMP1 | TD1 RST | TD1 SET | TF CMP2 | TF CMP1 | TF1 RST | TF1 SET | TB CMP2 | TB CMP1 | TB1 RST | TB1 SET |
| 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 |
| TA CMP2 | TA CMP1 | TA1 RST | TA1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.43 HRTIM 定时器 D 捕获 1 控制寄存器（HRTIM_CPT1DCR）

偏移地址：0x25C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TE CMP2 | TE CMP1 | TE1 RST | TE1 SET | TF CMP2 | TF CMP1 | TF1 RST | TF1 SET | TC CMP2 | TC CMP1 | TC1 RST | TC1 SET | TB CMP2 | TB CMP1 | TB1 RST | TB1 SET |
| 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 |
| TA CMP2 | TA CMP1 | TA1 RST | TA1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.44 HRTIM 定时器 E 捕获 1 控制寄存器（HRTIM_CPT1ECR）

偏移地址：0x2DC

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TF CMP2 | TF CMP1 | TF1 RST | TF1 SET | TD CMP2 | TD CMP1 | TD1 RST | TD1 SET | TC CMP2 | TC CMP1 | TC1 RST | TC1 SET | TB CMP2 | TB CMP1 | TB1 RST | TB1 SET |
| 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 |
| TA CMP2 | TA CMP1 | TA1 RST | TA1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.45 HRTIM 定时器 F 捕获 1 控制寄存器（HRTIM_CPT1FCR）

偏移地址：0x35C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TE CMP2 | TE CMP1 | TE1 RST | TE1 SET | TD CMP2 | TD CMP1 | TD1 RST | TD1 SET | TC CMP2 | TC CMP1 | TC1 RST | TC1 SET | TB CMP2 | TB CMP1 | TB1 RST | TB1 SET |
| 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 |
| TA CMP2 | TA CMP1 | TA1 RST | TA1 SET | EXEV1 0CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参见 HRTIM_CPT2xCR 位描述

### 21.5.46 HRTIM 定时器 x 捕获 2 控制寄存器（HRTIM_CPT2xCR）（x = A~F）

偏移地址：块 A：0x0E0
偏移地址：块 B：0x160
偏移地址：块 C：0x1E0
偏移地址：块 D：0x260
偏移地址：块 E：0x2E0
偏移地址：块 F：0x360
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TF CMP2 / TE CMP2 | TF CMP1 / TE CMP1 | TF1 RST / TE1 RST | TF1 SET / TE1 SET | TF CMP2 / TD CMP2 | TF CMP1 / TD CMP1 | TF1 RST / TD1 RST | TF1 SET / TD1 SET | TF CMP2 / TC CMP2 | TF CMP1 / TC CMP1 | TF1 RST / TC1 RST | TF1 SET / TC1 SET | TF CMP2 / TB CMP2 | TF CMP1 / TB CMP1 | TF1 RST / TB1 RST | TF1 SET / TB1 SET |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TF CMP2 / TA CMP2 | TF CMP1 / TA CMP1 | TF1 RST / TA1 RST | TF1 SET / TA1 SET | EXEV10CPT | EXEV9 CPT | EXEV8 CPT | EXEV7 CPT | EXEV6 CPT | EXEV5 CPT | EXEV4 CPT | EXEV3 CPT | EXEV2 CPT | EXEV1 CPT | UPD CPT | SW CPT |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2FCR 中：

**TECMP2**：定时器 E 比较 2（Timer E compare 2）

参考 TACMP2 描述。

在 HRTIM_CPT2ECR 中：

**TFCMP2**：定时器 F 比较 2（Timer F compare 2）

参考 TACMP2 描述。

Bit 30：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2FCR 中：

**TECMP1**：定时器 E 比较 1（Timer E compare 1）

参考 TACMP1 描述。

在 HRTIM_CPT2ECR 中：

**TFCMP1**：定时器 F 比较 1（Timer F compare 1）

参考 TACMP1 描述。

Bit 29：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2FCR 中：

**TE1RST**：定时器 E 输出 1 复位（Timer E output 1 reset）

参考 TA1RST 描述。

在 HRTIM_CPT2ECR 中：

**TF1RST**：定时器 F 输出 1 复位（Timer F output 1 reset）

参考 TA1RST 描述。

Bit 28：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2FCR 中：

**TE1SET**：定时器 E 输出 1 置位（Timer E output 1 set）

参考 TA1SET 描述。

在 HRTIM_CPT2ECR 中：

**TF1SET**：定时器 F 输出 1 置位（Timer F output 1 set）

参考 TA1SET 描述。

Bit 27：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TDCMP2**：定时器 D 比较 2（Timer D compare 2）

参考 TACMP2 描述。

在 HRTIM_CPT2DCR 中：

**TFCMP2**：定时器 F 比较 2（Timer F compare 2）

参考 TACMP2 描述。

Bit 26：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TDCMP1**：定时器 D 比较 1（Timer D compare 1）

参考 TACMP1 描述。

在 HRTIM_CPT2DCR 中：

**TFCMP1**：定时器 F 比较 1（Timer F compare 1）

参考 TACMP1 描述。

Bit 25：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TD1RST**：定时器 D 输出 1 复位（Timer D output 1 reset）

参考 TA1RST 描述。

在 HRTIM_CPT2DCR 中：

**TF1RST**：定时器 F 输出 1 复位（Timer F output 1 reset）

参考 TA1RST 描述。

Bit 24：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TD1SET**：定时器 D 输出 1 置位（Timer D output 1 set）

参考 TA1SET 描述。

在 HRTIM_CPT2DCR 中：

**TF1SET**：定时器 F 输出 1 置位（Timer F output 1 set）

参考 TA1SET 描述。

Bit 23：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TCCMP2**：定时器 C 比较 2（Timer C compare 2）

参考 TACMP2 描述。

在 HRTIM_CPT2CCR 中：

**TFCMP2**：定时器 F 比较 2（Timer F compare 2）

参考 TACMP2 描述。

Bit 22：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TCCMP1**：定时器 C 比较 1（Timer C compare 1）

参考 TACMP1 描述。

在 HRTIM_CPT2CCR 中：

**TFCMP1**：定时器 F 比较 1（Timer F compare 1）

参考 TACMP1 描述。

Bit 21：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TC1RST**：定时器 C 输出 1 复位（Timer C output 1 reset）

参考 TA1RST 描述。

在 HRTIM_CPT2CCR 中：

**TF1RST**：定时器 F 输出 1 复位（Timer F output 1 reset）

参考 TA1RST 描述。

Bit 20：在 HRTIM_CPT2ACR、HRTIM_CPT2BCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TC1SET**：定时器 C 输出 1 置位（Timer C output 1 set）

参考 TA1SET 描述。

在 HRTIM_CPT2CCR 中：

**TF1SET**：定时器 F 输出 1 置位（Timer F output 1 set）

参考 TA1SET 描述。

Bit 19：在 HRTIM_CPT2ACR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TBCMP2**：定时器 B 比较 2（Timer B compare 2）

参考 TACMP2 描述。

在 HRTIM_CPT2BCR 中：

**TFCMP2**：定时器 F 比较 2（Timer F compare 2）

参考 TACMP2 描述。

Bit 18：在 HRTIM_CPT2ACR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TBCMP1**：定时器 B 比较 1（Timer B compare 1）

参考 TACMP1 描述。

在 HRTIM_CPT2BCR 中：

**TFCMP1**：定时器 F 比较 1（Timer F compare 1）

参考 TACMP1 描述。

Bit 17：在 HRTIM_CPT2ACR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TB1RST**：定时器 B 输出 1 复位（Timer B output 1 reset）

参考 TA1RST 描述。

在 HRTIM_CPT2BCR 中：

**TF1RST**：定时器 F 输出 1 复位（Timer F output 1 reset）

参考 TA1RST 描述。

Bit 16：在 HRTIM_CPT2ACR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TB1SET**：定时器 B 输出 1 置位（Timer B output 1 set）

参考 TA1SET 描述。

在 HRTIM_CPT2BCR 中：

**TF1SET**：定时器 F 输出 1 置位（Timer F output 1 set）

参考 TA1SET 描述。

Bit 15：在 HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TACMP2**：定时器 A 比较 2（Timer A compare 2）

0：无操作

1：定时器 A 比较 2 触发捕获 2

在 HRTIM_CPT2ACR 中：

**TFCMP2**：定时器 F 比较 2（Timer F compare 2）

参考 TACMP2 描述。

Bit 14：在 HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TACMP1**：定时器 A 比较 1（Timer A compare 1）

0：无操作

1：定时器 A 比较 1 触发捕获 2

在 HRTIM_CPT2ACR 中：

**TFCMP1**：定时器 F 比较 1（Timer F compare 1）

参考 TACMP1 描述。

Bit 13：在 HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TA1RST**：定时器 A 输出 1 复位（Timer A output 1 reset）

0：无操作

1：TA1 输出从有效电平跳变为无效电平会触发捕获 2

在 HRTIM_CPT2ACR 中：

**TF1RST**：定时器 F 输出 1 复位（Timer F output 1 reset）

参考 TA1RST 描述。

Bit 12：在 HRTIM_CPT2BCR、HRTIM_CPT2CCR、HRTIM_CPT2DCR、HRTIM_CPT2ECR、HRTIM_CPT2FCR 中：

**TA1SET**：定时器 A 输出 1 置位（Timer A output 1 set）

0：无操作

1：TA1 输出从有效电平跳变为无效电平会触发捕获 2

在 HRTIM_CPT2ACR 中：

**TF1SET**：定时器 F 输出 1 置位（Timer F output 1 set）

参考 TA1SET 描述。

Bit 11 **EXEV10CPT**：外部事件 10 捕获（External event 10 capture）

参考 EXEV1CPT 描述。

Bit 10 **EXEV9CPT**：外部事件 9 捕获（External event 9 capture）

参考 EXEV1CPT 描述。

Bit 9 **EXEV8CPT**：外部事件 8 捕获（External event 8 capture）

参考 EXEV1CPT 描述。

Bit 8 **EXEV7CPT**：外部事件 7 捕获（External event 7 capture）

参考 EXEV1CPT 描述。

Bit 7 **EXEV6CPT**：外部事件 6 捕获（External event 6 capture）

参考 EXEV1CPT 描述。

Bit 6 **EXEV5CPT**：外部事件 5 捕获（External event 5 capture）

参考 EXEV1CPT 描述。

Bit 5 **EXEV4CPT**：外部事件 4 捕获（External event 4 capture）

参考 EXEV1CPT 描述。

Bit 4 **EXEV3CPT**：外部事件 3 捕获（External event 3 capture）

参考 EXEV1CPT 描述。

Bit 3 **EXEV2CPT**：外部事件 2 捕获（External event 2 capture）

参考 EXEV1CPT 描述。

Bit 2 **EXEV1CPT**：外部事件 1 捕获（External event 1 capture）

0：无操作

1：外部事件 1 触发捕获 2

Bit 1 **UPDCPT**：更新捕获（Update capture）

0：无操作

1：更新事件触发捕获 2

Bit 0 **SWCPT**：软件捕获（Software capture）

0：无操作

1：此位通过软件强制触发捕获 2。此位仅可置位，由硬件复位。

### 21.5.47 HRTIM 定时器 x 输出寄存器 (HRTIM_OUTxR) (x = A~F)

偏移地址：块 A：0x0E4
偏移地址：块 B：0x164
偏移地址：块 C：0x1E4
偏移地址：块 D：0x264
偏移地址：块 E：0x2E4
偏移地址：块 F：0x364
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | DIDL2 | CHP2 | FAULT2[1:0] | | IDLES2 | IDLEM2 | POL2 | Res |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | BIAR | Res | DLYPRT[2:0] | | | DLYPRTEN | DTEN | DIDL1 | CHP1 | FAULT1[1:0] | | IDLES1 | IDLEM1 | POL1 | Res |
| | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | |

Bit 31:24 保留，必须保持复位值。

Bit 23 **DIDL2**：进入突发模式空闲状态的输出 2 死区

此位会在输出切换为其空闲状态之前强制插入死区，从而延迟进入空闲模式。此设置仅适用于在突发模式工作期间进入空闲状态的情况。

0：编程的空闲状态立即应用到输出 2

1：死区（无效电平）会在进入空闲模式之前插入到输出 2 上。死区值由 DTFx[8:0]设置

*注：一旦定时器 x 被使能，这个参数就不能更改。*

仅当其中一路输出在突发模式期间有效（IDLES=1）、且死区为正值（SDTR/SDTF 设为 0）时，才可设置 DIDL=1。

Bit 22 **CHP2**：输出 2 斩波使能（Output 2 chopper enable）

该位在输出 2 上使能斩波。

0：输出信号不变

1：输出信号被载波信号斩波

*注：一旦定时器 x 被使能，这个参数就不能更改。*

Bit 21:20 **FAULT2[1:0]**：输出 2 故障状态（Output 2 fault state）

这些位选择故障事件后的输出 2 状态。

00：无动作：输出不受故障输入影响，保持在运行状态。

01：有效

10：无效

11：高阻态

*注：*如果 FLTENx 位已置 1，或者输出处于 FAULT 状态，则定时器 x 使能后（TxCEN 位置 1），不能更改此参数。

Bit 19 **IDLES2**：输出 2 空闲状态（Output 2 idle state）

此位选择输出 2 的空闲状态。

0：无效

1：有效

*注：在 HRTIM 控制输出之前，必须设置此参数。*

Bit 18 **IDLEM2**：输出 2 空闲模式（Output 2 idle mode）

此位选择输出 2 的空闲模式。
0：无操作：输出不受突发模式工作的影响。
1：当突发模式控制器请求时，输出处于空闲状态。
*注：此位进行了预装载，可在运行期间更改，但不能在突发模式激活时更改。*

Bit 17 **POL2**：输出 2 极性（Output 2 polarity）
此位选择输出 2 的极性。
0：正极性（输出高电平有效）
1：负极性（输出低电平有效）
*注：一旦定时器 x 被使能，这个参数就不能更改。*

Bit 16:15 保留，必须保持复位值。

Bit 14 **BIAR**：均衡空闲自动恢复（Balanced Idle Automatic Resume）
此位选择是否在均衡空闲事件后自动重新使能输出。
此位仅在 DLYPRT[2:0] = 011 或 111 时有意义，否则会被忽略。
0：禁用
1：使能
*注：一旦定时器 x 被使能，这个参数就不能更改。*

Bit 13 保留，必须保持复位值。

Bit 12:10 **DLYPRT[2:0]**：延迟保护（Delayed protection）
这些位定义应用延迟保护机制的源和输出。
**在 HRTIM_OUTAR、HRTIM_OUTBR、HRTIM_OUTCR 中：**
000：发生外部事件 6 时输出 1 进入延迟空闲状态
001：发生外部事件 6 时输出 2 进入延迟空闲状态
010：发生外部事件 6 时输出 1 和输出 2 均进入延迟空闲状态
011：发生外部事件 6 时进入均衡空闲状态
100：发生外部事件 7 时输出 1 进入延迟空闲状态
101：发生外部事件 7 时输出 2 进入延迟空闲状态
110：发生外部事件 7 时输出 1 和输出 2 均进入延迟空闲状态
111：发生外部事件 7 时进入均衡空闲状态
**在 HRTIM_OUTDR、HRTIM_OUTER、HRTIM_OUTFR 中：**
000：发生外部事件 8 时输出 1 进入延迟空闲状态
001：发生外部事件 8 时输出 2 进入延迟空闲状态
010：发生外部事件 8 时输出 1 和输出 2 均进入延迟空闲状态
011：发生外部事件 8 时进入均衡空闲状态
100：发生外部事件 9 时输出 1 进入延迟空闲状态
101：发生外部事件 9 时输出 2 进入延迟空闲状态
110：发生外部事件 9 时输出 1 和输出 2 均进入延迟空闲状态
111：发生外部事件 9 时进入均衡空闲状态
*注：一旦使能延迟保护（DLYPRTEN 位设置），则不得修改此位域。*

Bit 9 **DLYPRTEN**：延迟保护使能（Delayed protection enable）
此位使能延迟保护方案。
0：无动作
1：根据 DLYPRT[2:0]位使能延迟保护

*注：一旦定时器x 被使能（TxEN 位设置），则不能更改此参数。*

Bit 8 **DTEN**：死区时间使能（Deadtime enable）

此位使能输出 1 和输出 2 上的死区时间插入。

0：输出 1 和输出 2 信号是独立的。

1：在输出 1 和输出 2 之间插入死区时间（参考信号是输出 1 信号发生器）

*注：一旦定时器开始运行（TxEN 位设置）或其输出被另一个定时器使能和置位/ 重置，则不能更改此参数。*

Bit 7 **DIDL1**：进入突发模式空闲状态的输出 1 死区时间

此位会在输出切换为其空闲状态之前强制插入死区，从而延迟进入空闲模式。此设置仅适用于在突发模式工作期间进入空闲状态的情况。

0：编程的空闲状态立即应用于输出 1

1：死区（无效电平）会在进入空闲模式之前插入到输出 1 上。死区值由 DTRx[8:0]置位

*注：一旦定时器x 被使能，则不能更改此参数。*

仅当其中一路输出在突发模式期间有效（IDLES=1）、且死区为正值（SDTR/SDTF 设为 0）时， 才可设置 DIDL=1。

Bit 6 **CHP1**：输出 1 斩波使能（Output 1 chopper enable）

此位使能输出 1 上的斩波。

0：输出信号不变

1：输出信号被载波信号斩波

*注：一旦定时器x 被使能，则不能更改此参数。*

Bit 5:4 **FAULT1[1:0]**：输出 1 故障状态（Output 1 fault state）

这些位选择故障事件后输出 1 的状态

00：无操作：输出不受故障输入影响，保持在运行模式。

01：有效状态

10：无效状态

11：高阻态

*注：如果FLTENx 位已置 1，或者输出处于FAULT 状态，则定时器x 使能后（TxCEN 位置1），不能更改此参数*

Bit 3 **IDLES1**：输出 1 空闲状态（Output 1 Idle State）

此位选择输出 1 的空闲状态。

0：无效状态

1：有效状态

*注：在HRTIM 控制输出之前，必须设置此参数。*

Bit 2 **IDLEM1**：输出 1 空闲模式（Output 1 Idle mode）

此位选择输出 1 的空闲模式。

0：无操作：输出不受突发模式操作影响。

1：当突发模式控制器请求时，输出处于空闲状态。

*注：此位进行了预装载，可在运行期间更改，但不能在突发模式激活时更改。*

Bit 1 **POL1**：输出 1 极性（Output 1 polarity）

此位选择输出 1 的极性。

0：正极性（输出有效高电平）

1：负极性（输出有效低电平）

*注：一旦定时器 x 被使能，则无法更改此参数。*

Bit 0 保留，必须保持复位值。

### 21.5.48 HRTIM 定时器 x 故障寄存器（HRTIM_FLTxR）（x = A~F）

偏移地址：块 A：0x0E8
偏移地址：块 B：0x168
偏移地址：块 C：0x1E8
偏移地址：块 D：0x268
偏移地址：块 E：0x2E8
偏移地址：块 F：0x368
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FTLCK | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | FLT6EN | FLT5EN | FLT4EN | FLT3EN | FLT2EN | FLT1EN |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

Bit 31 **FLTLCK**：故障源锁定（Fault sources lock）

0：FLT1EN..FLT6EN 位可读/写

1：FLT1EN..FLT6EN 位只读

FLTLCK 位为一次性写入。一旦设置，直到下一次系统复位前无法修改。

Bit 30:6 保留，必须保持复位值。

Bit 5 **FLT6EN**：故障 6 使能（Fault 6 enable）

0：忽略故障 6 输入

1：故障 6 输入有效并禁用 HRTIM 输出

Bit 4 **FLT5EN**：故障 5 使能（Fault 5 enable）

0：忽略故障 5 输入

1：故障 5 输入有效并禁用 HRTIM 输出

Bit 3 **FLT4EN**：故障 4 使能（Fault 4 enable）

0：忽略故障 4 输入

1：故障 4 输入有效并禁用 HRTIM 输出

Bit 2 **FLT3EN**：故障 3 使能（Fault 3 enable）

0：忽略故障 3 输入

1：故障 3 输入有效并禁用 HRTIM 输出

Bit 1 **FLT2EN**：故障 2 使能（Fault 2 enable）

0：忽略故障 2 输入

1：故障 2 输入有效并禁用 HRTIM 输出

Bit 0 **FLT1EN**：故障 1 使能（Fault 1 enable）

0：忽略故障 1 输入

1：故障 1 输入有效并禁用 HRTIM 输出

### 21.5.49 HRTIM 定时器 x 控制寄存器 2（HRTIM_TIMxCR2）（x = A~F）

偏移地址：块 A：0x0EC
偏移地址：块 B：0x16C
偏移地址：块 C：0x1EC
偏移地址：块 D：0x26C
偏移地址：块 E：0x2EC
偏移地址：块 F：0x36C
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TRGHLF | Reserved | | GTCMP3 | GTCMP1 |
| | | | | | | | | | | | rw | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FEROM[1:0] | | BMROM[1:0] | | ADROM[1:0] | | OUTROM[1:0] | | ROM[1:0] | | Res | UDM | Res | DCDR | DCDS | DCDE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | rw |

Bit 30:18 保留，必须保持复位值。

Bit 20 **TRGHLF**：半触发模式（Triggered-half mode）

此位域定义比较 2 寄存器是在标准模式下工作（计数器等于比较值时立即发出比较匹配），还是在半触发模式下工作（请参阅章节 22.3.4 中的半触发模式）。

0：CMP2 寄存器仅由用户写入（标准比较模式）

1：当捕获 1 事件发生时，CMP2 的值由硬件设置。它被加载为（捕获 1 值除以 2）的结果。初始值可以由用户写入（只要 TRGHLF 被复位），但在触发第一个捕获后被忽略（当设置 TRGHLF 位时，CMP2 的预加载机制被禁用）。

*注：计数器使能后（TxCEN 位设置），不得修改此位域。*

Bit 19:18 保留，必须保持复位值。

Bit 17 **GTCMP3**：大于比较 3 的 PWM 模式（Greater than compare 3 PWM mode）

此位定义比较 3 的操作模式。

0：当计数器等于比较值时生成比较 3 事件（比较匹配模式）

1：当计数器大于比较值时生成比较 3 事件。如果比较值在运行时更改，则将新的比较值与当前计数器值进行比较，并可以生成输出置位或复位。

Bit 16 **GTCMP1**：大于比较 1 的 PWM 模式（Greater than compare 1 PWM mode）

此位定义比较 1 的操作模式。

0：当计数器等于比较值时生成比较 1 事件（比较匹配模式）

1：当计数器大于比较值时生成比较 1 事件。如果比较值在运行时更改，则将新的比较值与当前计数器值进行比较，并可以生成输出置位或复位。

Bit 15:14 **FEROM[1:0]**：故障和事件翻转模式（Fault and event roll-over mode）

此位域定义在上下计数模式下何时生成溢出。它仅涉及故障和事件计数器所使用的翻转事件。

00：当计数器等于 0 或等于 HRTIM_PERxR 值时生成事件

01：当计数器等于 0 时生成事件

10：当计数器等于 HRTIM_PERxR 时生成事件

11：保留

*注：此设置仅当* UDM *位设置时适用。否则它不重要。*

*注：一旦定时器开始运行（*TxEN *位设置），此位域无法更改。*

Bit 13:12 **BMROM[1:0]**：突发模式翻转模式（Burst mode roll-over mode）

此位域定义在上下计数模式下何时生成翻转。它仅涉及在突发模式控制器中使用的翻转事件，作为时钟以及突发模式触发。

00：当计数器等于 0 或等于 HRTIM_PERxR 值时生成事件

01：当计数器等于 0 时生成事件

10：当计数器等于 HRTIM_PERxR 时生成事件

11：保留

*注：此设置仅当* UDM *位设置时适用。否则它不重要。*

*注：一旦定时器开始运行（*TxEN *位设置），此参数无法更改。*

Bit 11:10 **ADROM[1:0]**：ADC 翻转模式（ADC roll-over mode）

此位域定义在上下计数模式下何时生成翻转。它仅涉及触发 ADC 的翻转事件。

00：当计数器等于 0 或等于 HRTIM_PERxR 值时生成事件

01：当计数器等于 0 时生成事件

10：当计数器等于 HRTIM_PERxR 时生成事件

11：保留

*注：此设置仅当* UDM *位设置时适用。否则它不重要。*

*注：一旦定时器开始运行（*TxEN *位设置），此位域无法更改。*

Bit 9:8 **OUTROM[1:0]**：输出翻转模式（Output roll-over mode）

此位域定义在上下计数模式下何时生成翻转。它仅涉及根据 HRTIM_SETxyR 和 HRTIM_RSTxyR 设置来设置和/或重置输出的翻转事件。

00：当计数器等于 0 或等于 HRTIM_PERxR 值时生成事件

01：当计数器等于 0 时生成事件

10：当计数器等于 HRTIM_PERxR 时生成事件

11：保留

注：此设置仅当 UDM 位设置时适用。否则它不重要。

注：一旦定时器开始运行（TxEN 位设置），此位域无法更改。

Bit 7:6 **ROM[1:0]**：翻转模式（Roll-over mode）

此位域定义在上下计数模式下何时生成翻转。它仅涉及具有以下目标的翻转事件：更新触发（将内容从预加载寄存器传输到有效寄存器）、IRQ 和 DMA 请求、重复计数器递减和外部事件过滤。

00：当计数器等于 0 或等于 HRTIM_PERxR 值时生成事件

01：当计数器等于 0 时生成事件

10：当计数器等于 HRTIM_PERxR 时生成事件
11：保留
*注：此设置仅当 UDM 位设置时适用。否则它不重要。*
*注：一旦定时器开始运行（TxEN 位设置），此位域无法更改。*

Bit 5 保留，必须保持复位值。

Bit 4 **UDM**：上下计数模式（Up-Down mode）
此位定义计数器是否以向上或上下计数模式运行。
0：计数器以向上计数模式运行
1：计数器以上下计数模式运行
*注：一旦定时器开始运行（TxEN 位设置），此位无法更改。*

Bit 3 保留，必须保持复位值。

Bit 2 **DCDR**：双通道 DAC 复位触发（Dual channel DAC reset trigger）
此位定义何时生成 hrtim_dac_reset_trgx 触发。
0：在计数器复位或溢出事件时生成触发
1：在输出 1 设置事件时生成触发
*注：当 DCDE 位复位（双通道 DAC 触发禁用）时，DCDR 位无效。*

Bit 1 **DCDS**：双通道 DAC 步进触发（Dual channel DAC Step trigger）
此位定义何时生成 hrtim_dac_step_trgx 触发。
0：在比较 2 事件时生成触发
1：在输出 1 复位事件时生成触发
*注：当 DCDE 位复位（双通道 DAC 触发禁用）时，DCDS 位无效。*

位 0 **DCDE**：双通道 DAC 触发使能（Dual channel DAC trigger enable）
此位使能双通道 DAC 触发机制。
0：双通道 DAC 触发禁用
1：双通道 DAC 触发使能
*注：一旦定时器开始运行（TxEN 位设置），此位无法更改。*

### 21.5.50 HRTIM 定时器 x 外部事件滤波寄存器 3 (HRTIM_TIMxEEFR3)（x = A~F）

偏移地址：块 A：0x0F0
偏移地址：块 B：0x170
偏移地址：块 C：0x1F0
偏移地址：块 D：0x270
偏移地址：块 E：0x2F0
偏移地址：块 F：0x370
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | EEVACNT[5:0] | | | | | | EEVASEL[3:0] | | | | Res | EEVARSTM | EEVACRES | EEVACE |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw |

Bit 30:14 保留，必须保持复位值。

Bit 13:8 **EEVACNT[5:0]**：外部事件 A 计数器（External event A counter）

此位域选择外部事件 A 计数器的阈值。当事件数量等于（EEVACNT[5:0]+1）值时，该事件被视为有效。

Bit 7:4 **EEVASEL[3:0]**：外部事件 A 选择（External event A Selection）

此位选择外部事件 A 的源。

0：外部事件 1 用作外部事件 A 源

1：外部事件 2 用作外部事件 A 源

...

9：外部事件 10 用作外部事件 A 源

其他：保留

Bit 3 保留，必须保持复位值。

Bit 2 **EEVARSTM**：外部事件 A 复位模式（External event A reset mode）

此位选择外部事件 x 计数器的复位模式。

0：外部事件计数器 A 在每个复位/翻转事件时复位

1：外部事件计数器 A 仅在最后一个计数周期内没有事件发生时，在每个复位/翻转事件时复位

Bit 1 **EEVACRES**：外部事件 A 计数器复位（External event A counter reset）

此位复位外部事件 A 计数器。它由软件设置并由硬件复位。

0：无操作

1：外部事件计数器 A 复位

Bit 0 **EEVACE**：外部事件 A 计数器使能（External event A counter enable）

此位使能外部事件 x 计数器。

0：外部事件 A 计数器禁用

1：外部事件 A 计数器使能

### 21.5.51 HRTIM 控制寄存器 1 (HRTIM_CR1)

偏移地址：0x380

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | AD4USRC[2:0] | | | AD3USRC[2:0] | | | AD2USRC[2:0] | | | AD1USRC[2:0] | | |
| | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | TFUDIS | TEUDIS | TDUDIS | TCUDIS | TBUDIS | TAUDIS | MUDIS |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

Bit 30:28 保留，必须保持复位值。

Bit 27:25 **AD4USRC[2:0]**：ADC 触发 4 更新源（ADC trigger 4 update source）

参考 AD1USRC[2:0]的描述。

Bit 24:22 **AD3USRC[2:0]**：ADC 触发 3 更新源（ADC trigger 3 update source）

参考 AD1USRC[2:0]的描述。

Bit 21:19 **AD2USRC[2:0]**：ADC 触发 2 更新源（ADC trigger 2 update source）

参考 AD1USRC[2:0]的描述。

Bit 18:16 **AD1USRC[2:0]**：ADC 触发 1 更新源（ADC trigger 1 update source）

这些位定义将触发 HRTIM_ADC1R 寄存器更新的源（从预装载寄存器传输到活动寄存器）。它只定义源定时器。准确条件在定时器自身的 HRTIM_MCR 或 HRTIM_TIMxCR 中定义。

000：主定时器

001：定时器 A

010：定时器 B

011：定时器 C

100：定时器 D

101：定时器 E

110：定时器 F

111：保留

Bit 15:7 保留，必须保持复位值。

Bit 6 **TFUDIS**：定时器 F 更新禁止（Timer F update disable）

参考 TAUDIS 的描述。

Bit 5 **TEUDIS**：定时器 E 更新禁止（Timer E update disable）

参考 TAUDIS 的描述。

Bit 4 **TDUDIS**：定时器 D 更新禁止（Timer D update disable）

参考 TAUDIS 的描述。

Bit 3 **TCUDIS**：定时器 C 更新禁止（Timer C update disable）

参考 TAUDIS 的描述。

Bit 2 **TBUDIS**：定时器 B 更新禁止（Timer B update disable）

参考 TAUDIS 的描述。

Bit 1 **TAUDIS**：定时器 A 更新禁止（Timer A update disable）

此位由软件置 1 和清零，用于暂时使能/禁止定时器 A 上的更新事件生成。

0：使能更新。生成所选源后立即进行更新。

1：禁止更新。暂时禁止更新，使软件能够写入多个需要同时考虑的寄存器。

Bit 0 **MUDIS**：主定时器更新使能（Master update disable）

此位由软件置 1 和清零，用于暂时使能/禁止更新事件生成。

0：使能更新。

1：禁止更新。暂时禁止更新，使软件能够写入多个需要同时考虑的寄存器。

### 21.5.52 HRTIM 控制寄存器 2 (HRTIM_CR2)

偏移地址：0x384

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | SWPF | SWPE | SWPD | SWPC | SWPB | SWPA |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | TFRST | TERST | TDRST | TCRST | TBRST | TARST | MRST | Res | TF SWU | TE SWU | TD SWU | TC SWU | TB SWU | TA SWU | MSWU |
| | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

Bit 31:22 保留，必须保持复位值。

Bit 21 **SWPF**：交换定时器 F 输出（Swap timer F outputs）

参考 SWPA 的描述。

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 20 **SWPE**：交换定时器 E 输出（Swap timer E outputs）

参考 SWPA 的描述。

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 19 **SWPD**：交换定时器 D 输出（Swap timer D outputs）

参考 SWPA 的描述。

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 18 **SWPC**：交换定时器 C 输出（Swap timer C outputs）

参考 SWPA 的描述。

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 17 **SWPB**：交换定时器 B 输出（Swap timer B outputs）

参考 SWPA 的描述。

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 16 **SWPA**：交换定时器 A 输出（Swap timer A outputs）

此位允许交换定时器 A 的输出。

0：HRTIM_SETA1R 和 HRTIM_RSTA1R 编码为输出 A1，HRTIM_SETA2R 和 HRTIM_RSTA2R 编码为输出 A2

1：HRTIM_SETA1R 和 HRTIM_RSTA1R 编码为输出 A2，HRTIM_SETA2R 和 HRTIM_RSTA2R 编码为输出 A1

*注：当使能推挽模式（PSHPLL = 1）时，此位无效。*

Bit 15 保留，必须保持复位值。

Bit 14 **TFRST**：定时器 F 计数器软件复位（Timer F counter software reset）

参考 TARST 的描述。

Bit 13 **TERST**：定时器 E 计数器软件复位（Timer E counter software reset）

参考 TARST 的描述。

Bit 12 **TDRST**：定时器 D 计数器软件复位（Timer D counter software reset）

参考 TARST 的描述。

Bit 11 **TCRST**：定时器 C 计数器软件复位（Timer C counter software reset）
参考 TARST 的描述。

Bit 10 **TBRST**：定时器 B 计数器软件复位（Timer B counter software reset）
参考 TARST 的描述。

Bit 9 **TARST**：定时器 A 计数器软件复位（Timer A counter software reset）
将此位置 1 会复位定时器 A 计数器。
此位自动通过硬件复位。

Bit 8 **MRST**：主定时器计数器软件复位（Master counter software reset）
将此位置 1 会复位主定时器计数器。
此位自动通过硬件复位。

Bit 7 保留，必须保持复位值。

Bit 6 **TFSWU**：定时器 F 软件更新（Timer F software update）
参考 TASWU 的描述。

Bit 5 **TESWU**：定时器 E 软件更新（Timer E software update）
参考 TASWU 的描述。

Bit 4 **TDSWU**：定时器 D 软件更新（Timer D software update）
参考 TASWU 的描述。

Bit 3 **TCSWU**：定时器 C 软件更新（Timer C software update）
参考 TASWU 的描述。

Bit 2 **TBSWU**：定时器 B 软件更新（Timer B software update）
参考 TASWU 的描述。

Bit 1 **TASWU**：定时器 A 软件更新（Timer A software update）
此位由软件置 1，并由硬件自动复位。它会强制立即执行从预装载寄存器到活动寄存器的传输操作，并会取消任何挂起的更新请求。

Bit 0 **MSWU**：主定时器软件更新（Master timer software update）
此位由软件置 1，并由硬件自动复位。它会强制立即执行从主定时器中的预装载寄存器到活动寄存器的传输操作，并会取消任何挂起的更新请求。

### 21.5.53 HRTIM 中断状态寄存器 (HRTIM_ISR)

偏移地址：0x388

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | BMPER | DLLRDY |
| | | | | | | | | | | | | | | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FLT6 | SYSFLY | FLT5 | FLT4 | FLT3 | FLT2 | FLT1 |
| | | | | | | | | | r | | r | r | r | r | r |

Bit 31:18 保留，必须保持复位值。

Bit 17 **BMPER**：突发模式周期中断标志（Burst mode period interrupt flag）

单发突发模式工作结束、或连续模式下突发模式周期结束时，此位会由硬件置 1。通过软件向该位写入 1 可将其清零。

0：未发生突发模式周期中断

1：发生突发模式周期中断

Bit 16 **DLLRDY**：DLL 就绪中断标志（DLL Ready Interrupt Flag）

当 DLL 校准完成时，硬件会设置此位。通过软件将其写为 1 来清除。

0：未发生 DLL 校准就绪中断

1：DLL 校准就绪中断发生

Bit 15:7 保留，必须保持复位值。

Bit 6 **FLT6**：故障 6 中断标志（Fault 6 interrupt flag）

参考 FLT1 的描述。

Bit 5 **SYSFLT**：系统故障中断标志（System fault interrupt flag）

参考 FLT1 的描述。

Bit 4 **FLT5**：故障 5 中断标志（Fault 5 interrupt flag）

参考 FLT1 的描述。

Bit 3 **FLT4**：故障 4 中断标志（Fault 4 interrupt flag）

参考 FLT1 的描述。

Bit 2 **FLT3**：故障 3 中断标志（Fault 3 interrupt flag）

参考 FLT1 的描述。

Bit 1 **FLT2**：故障 2 中断标志（Fault 2 interrupt flag）

参考 FLT1 的描述。

Bit 0 **FLT1**：故障 1 中断标志（Fault 1 interrupt flag）

当发生故障 1 事件时，硬件会设置此位。通过软件将其写为 1 来清除。

0：未发生故障 1 中断

1：发生故障 1 中断

### 21.5.54 HRTIM 中断清除寄存器（HRTIM_ICR）

偏移地址：0x38C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | BMPERC | DLLRDYC |
| | | | | | | | | | | | | | | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FLT6C | SYSFLYC | FLT5C | FLT4C | FLT3C | FLT2C | FLT1C |
| | | | | | | | | | w | w | w | w | w | w | w |

Bit 31:18 保留，必须保持复位值。

Bit 17 **BMPERC**：突发模式周期标志清除（Burst mode period flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 BMPER 标志。

Bit 16 **DLLRDYC**：DLL 就绪中断标志清除（DLL Ready Interrupt flag Clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 DLLRDY 标志。

Bit 15:7 保留，必须保持复位值。

Bit 6 **FLT6C**：故障 6 中断标志清除（Fault 6 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT6 标志。

Bit 5 **SYSFLTC**：系统故障中断标志清除（System fault interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 SYSFLT 标志。

Bit 4 **FLT5C**：故障 5 中断标志清除（Fault 5 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT5 标志。

Bit 3 **FLT4C**：故障 4 中断标志清除（Fault 4 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT4 标志。

Bit 2 **FLT3C**：故障 3 中断标志清除（Fault 3 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT3 标志。

Bit 1 **FLT2C**：故障 2 中断标志清除（Fault 2 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT2 标志。

Bit 0 **FLT1C**：故障 1 中断标志清除（Fault 1 interrupt flag clear）
向此位写入 1 将清除 HRTIM_ISR 寄存器中的 FLT1 标志。

### 21.5.55 HRTIM 中断使能寄存器（HRTIM_IER）

偏移地址：0x390

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | BMPERIE | DLLRDYIE |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FLT6IE | SYSFLYIE | FLT5IE | FLT4IE | FLT3IE | FLT2IE | FLT1IE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

Bit 31:18 保留，必须保持复位值。

Bit 17 **BMPERIE**：突发模式周期中断使能（Burst mode period interrupt enable）

此位由软件设置和清除，以使能/禁用突发模式周期中断。

0：突发模式周期中断禁用

1：突发模式周期中断使能

Bit 16 **DLLRDYIE**：DLL 就绪中断使能（DLL Ready Interrupt Enable）

此位由软件设置和清除，以使能/禁用 DLL 就绪中断。

0：DLL 就绪中断禁用

1：DLL 就绪中断使能

Bit 15:7 保留，必须保持复位值。

Bit 6 **FLT6IE**：故障 6 中断使能（Fault 6 interrupt enable）

参考 FLT1IE 的描述。

Bit 5 **SYSFLT IE**：系统故障中断使能（System fault interrupt enable）

参考 FLT1IE 的描述。

Bit 4 **FLT5 IE**：故障 5 中断使能（Fault 5 interrupt enable）

参考 FLT1IE 的描述。

Bit 3 **FLT4 IE**：故障 4 中断使能（Fault 4 interrupt enable）

参考 FLT1IE 的描述。

Bit 2 **FLT3 IE**：故障 3 中断使能（Fault 3 interrupt enable）

参考 FLT1IE 的描述。

Bit 1 **FLT2 IE**：故障 2 中断使能（Fault 2 interrupt enable）

参考 FLT1IE 的描述。

Bit 0 **FLT1 IE**：故障 1 中断使能（Fault 1 interrupt enable）

此位由软件设置和清除，以使能/禁用故障 1 中断。

0：故障 1 中断禁用

1：故障 1 中断使能

### 21.5.56 HRTIM 输出使能寄存器（HRTIM_OENR）

偏移地址：0x394

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | TF2OEN | TF1OEN | TE2OEN | TE1OEN | TD2OEN | TD1OEN | TC2OEN | TC1OEN | TB2OEN | TB1OEN | TA2OEN | TA1OEN |
| | | | | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs | rs |

Bit 31:12 保留，必须保持复位值。

Bit 11 **TF2OEN**：定时器 F 输出 2 使能（Timer F output 2 enable）
参考 TA1OEN 的描述。

Bit 10 **TF1OEN**：定时器 F 输出 1 使能（Timer F output 1 enable）
参考 TA1OEN 的描述。

Bit 9 **TE2OEN**：定时器 E 输出 2 使能（Timer E output 2 enable）
参考 TA1OEN 的描述。

Bit 8 **TE1OEN**：定时器 E 输出 1 使能（Timer E output 1 enable）
参考 TA1OEN 的描述。

Bit 7 **TD2OEN**：定时器 D 输出 2 使能（Timer D output 2 enable）
参考 TA1OEN 的描述。

Bit 6 **TD1OEN**：定时器 D 输出 1 使能（Timer D output 1 enable）
参考 TA1OEN 的描述。

Bit 5 **TC2OEN**：定时器 C 输出 2 使能（Timer C output 2 enable）
参考 TA1OEN 的描述。

Bit 4 **TC1OEN**：定时器 C 输出 1 使能（Timer C output 1 enable）
参考 TA1OEN 的描述。

Bit 3 **TB2OEN**：定时器 B 输出 2 使能（Timer B output 2 enable）
参考 TA1OEN 的描述。

Bit 2 **TB1OEN**：定时器 B 输出 1 使能（Timer B output 1 enable）
参考 TA1OEN 的描述。

Bit 1 **TA2OEN**：定时器 A 输出 2 使能（Timer A output 2 enable）
参考 TA1OEN 的描述。

Bit 0 **TA1OEN**：定时器 A 输出 1 使能（Timer A output 1 enable）
设置此位将使能定时器 A 的输出 1。写入“0”无影响。
读取此位将返回输出使能/禁用状态。
当与定时器相关的故障输入处于有效状态时，此位会由硬件异步清除。
0：输出 A1 禁用。输出处于故障或空闲状态。
1：输出 A1 使能
*注：禁止状态对应空闲和故障两种状态。输出禁止状态由* HRTIM_ODSR *寄存器中的* TA1ODS *位提供。*

### 21.5.57 HRTIM 输出禁用寄存器（HRTIM_ODISR）

偏移地址：0x398

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | TF2 ODIS | TF1 ODIS | TE2 ODIS | TE1 ODIS | TD2 ODIS | TD1 ODIS | TC2 ODIS | TC1 ODIS | TB2 ODIS | TB1 ODIS | TA2 ODIS | TA1 ODIS |
| | | | | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31:12 保留，必须保持复位值。

Bit 11 **TF2ODIS**：定时器 F 输出 2 禁用（Timer F output 2 disable）
参考 TA1ODIS 的描述。

Bit 10 **TF1ODIS**：定时器 F 输出 1 禁用（Timer F output 1 disable）
参考 TA1ODIS 的描述。

Bit 9 **TE2ODIS**：定时器 E 输出 2 禁用（Timer E output 2 disable）
参考 TA1ODIS 的描述。

Bit 8 **TE1ODIS**：定时器 E 输出 1 禁用（Timer E output 1 disable）
参考 TA1ODIS 的描述。

Bit 7 **TD2ODIS**：定时器 D 输出 2 禁用（Timer D output 2 disable）
参考 TA1ODIS 的描述。

Bit 6 **TD1ODIS**：定时器 D 输出 1 禁用（Timer D output 1 disable）
参考 TA1ODIS 的描述。

Bit 5 **TC2ODIS**：定时器 C 输出 2 禁用（Timer C output 2 disable）
参考 TA1ODIS 的描述。

Bit 4 **TC1ODIS**：定时器 C 输出 1 禁用（Timer C output 1 disable）
参考 TA1ODIS 的描述。

Bit 3 **TB2ODIS**：定时器 B 输出 2 禁用（Timer B output 2 disable）
参考 TA1ODIS 的描述。

Bit 2 **TB1ODIS**：定时器 B 输出 1 禁用（Timer B output 1 disable）
参考 TA1ODIS 的描述。

Bit 1 **TA2ODIS**：定时器 A 输出 2 禁用（Timer A output 2 disable）
参考 TA1ODIS 的描述。

Bit 0 **TA1ODIS**：定时器 A 输出 1 禁用（Timer A output 1 enable）
将此位置 1 会禁止定时器 A 输出 1。输出从运行状态或故障状态进入空闲状态。
写入“0”无影响。

### 21.5.58 HRTIM 输出禁用状态寄存器（HRTIM_ODSR）

偏移地址：0x39C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | TF2ODS | TF1ODS | TE2ODS | TE1ODS | TD2ODS | TD1ODS | TC2ODS | TC1ODS | TB2ODS | TB1ODS | TA2ODS | TA1ODS |
| | | | | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:12 保留，必须保持复位值。

Bit 11 **TF2ODS**：定时器 F 输出 2 禁用状态（Timer F output 2 disable status）
参考 TA1ODS 的描述。

Bit 10 **TF1ODS**：定时器 F 输出 1 禁用状态（Timer F output 1 disable status）
参考 TA1ODS 的描述。

Bit 9 **TE2ODS**：定时器 E 输出 2 禁用状态（Timer E output 2 disable status）
参考 TA1ODS 的描述。

Bit 8 **TE1ODS**：定时器 E 输出 1 禁用状态（Timer E output 1 disable status）
参考 TA1ODS 的描述。

Bit 7 **TD2ODS**：定时器 D 输出 2 禁用状态（Timer D output 2 disable status）
参考 TA1ODS 的描述。

Bit 6 **TD1ODS**：定时器 D 输出 1 禁用状态（Timer D output 1 disable status）
参考 TA1ODS 的描述。

Bit 5 **TC2ODS**：定时器 C 输出 2 禁用状态（Timer C output 2 disable status）
参考 TA1ODS 的描述。

Bit 4 **TC1ODS**：定时器 C 输出 1 禁用状态（Timer C output 1 disable status）
参考 TA1ODS 的描述。

Bit 3 **TB2ODS**：定时器 B 输出 2 禁用状态（Timer B output 2 disable status）
参考 TA1ODS 的描述。

Bit 2 **TB1ODS**：定时器 B 输出 1 禁用状态（Timer B output 1 disable status）
参考 TA1ODS 的描述。

Bit 1 **TA2ODS**：定时器 A 输出 2 禁用状态（Timer A output 2 disable status）
参考 TA1ODS 的描述。

Bit 0 **TA1ODS**：定时器 A 输出 1 禁用状态（Timer A output 1 enable status）
读取此位将返回输出禁用状态。当输出处于有效状态时（Tx1OEN 或 Tx2OEN = 1），此位没有意义。
0：输出 A1 禁用，处于空闲状态。
1：输出 A1 禁用，处于故障状态。

### 21.5.59 HRTIM 突发模式控制寄存器（HRTIM_BMCR）

偏移地址：0x3A0

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BMSTAT | Reserved | | | | | | | | TFBM | TEBM | TDBM | TCBM | TBBM | TABM | MTBM |
| rc_w0 | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BMPREN | BMPRSC[3:0] | | | | BMCLK[3:0] | | | | BMOM | BME |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **BMSTAT**：突发模式状态（Burst mode status）

此位提供当前工作状态。

0：正常工作

1：正在进行突发操作。将此位写入 0 会使突发模式提前终止。

Bit 30:23 保留，必须保持复位值。

Bit 22 **TFBM**：定时器 F 突发模式（Timer F burst mode）

参考 TABM 描述。

Bit 21 **TEBM**：定时器 E 突发模式（Timer E burst mode）

参考 TABM 描述。

Bit 20 **TDBM**：定时器 D 突发模式（Timer D burst mode）

参考 TABM 描述。

Bit 19 **TCBM**：定时器 C 突发模式（Timer C burst mode）

参考 TABM 描述。

Bit 18 **TBBM**：定时器 B 突发模式（Timer B burst mode）

参考 TABM 描述。

Bit 17 **TABM**：定时器 A 突发模式（Timer A burst mode）

此位定义定时器在突发模式工作期间的行为。突发模式使能后，不能更改此位域。

0：定时器 A 计数器时钟保持，定时器正常工作

1：定时器 A 计数器时钟停止，计数器复位

*注：当均衡空闲模式激活时（*DLYPRT[2:0] = 0x11*），不得将此位置*1

Bit 16 **MTBM**：主定时器突发模式（Master timer burst mode）

此位定义定时器在突发模式工作期间的行为。突发模式使能后，不能更改此位域。

0：主定时器计数器时钟保持，定时器正常工作

1：主定时器计数器时钟停止，计数器复位

Bit 15:11 保留，必须保持复位值。

Bit 10 **BMPREN**：突发模式预装载使能（Burst mode preload enable）

此位可使能寄存器预装载机制，并定义对可预装载寄存器（HRTIM_BMCMPR、HRTIM_BMPER）的写访问是在活动寄存器中进行还是在预装载寄存器中进行。

0：禁止预装载：直接对活动寄存器进行写访问

1：使能预装载：对预装载寄存器进行写访问

Bit 9:6 **BMPRSC[3:0]**：突发模式预分频器（Burst mode prescaler）

定义突发模式控制器的 $f_{HRTIM}$ 时钟预分频比。突发模式使能后，不能更改此位域。

0000：时钟不分频

0001：2 分频

0010：4 分频

0011：8 分频

0100：16 分频

0101：32 分频

0110：64 分频

0111：128 分频

1000：256 分频

1001：512 分频

1010：1024 分频

1011：2048 分频

1100：4096 分频

1101：8192 分频

1110：16384 分频

1111：32768 分频

Bit 5:2 **BMCLK[3:0]**：突发模式时钟源（Burst mode clock source）

此位域定义突发模式计数器的时钟源。当突发模式使能时，它无法更改。

0000：主定时器计数器复位/翻转

0001：定时器 A 计数器复位/翻转

0010：定时器 B 计数器复位/翻转

0011：定时器 C 计数器复位/翻转

0100：定时器 D 计数器复位/翻转

0101：定时器 E 计数器复位/翻转

0110：片上事件 1（hrtim_bm_ck1）作为突发模式计数器时钟

0111：片上事件 2（hrtim_bm_ck2）作为突发模式计数器时钟

1000：片上事件 3（hrtim_bm_ck3）作为突发模式计数器时钟

1001：片上事件 4（hrtim_bm_ck4）作为突发模式计数器时钟

1010：预分频 $f_{HRTIM}$ 时钟（根据 BMPRSC[3:0]设置）

1011：定时器 F 计数器复位/翻转

其他：保留

Bit 1 **BMOM**：突发模式工作模式（Burst mode operating mode）

此位定义进入一次突发模式还是连续工作。

0：单发模式

1：连续工作

Bit 0 **BME**：突发模式使能（Burst mode enable）

此位用于启动已准备好接收启动触发信号的突发模式控制器。

向此位写入 0 会使突发模式提前终止。

0：突发模式禁用

1：突发模式使能

### 21.5.60 HRTIM 突发模式触发寄存器 (HRTIM_BMTRGR)

偏移地址：0x3A4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OCHPEV | EEV8 | EEV7 | TDEEV8 | TAEEV7 | TECMP2 | TECMP1 | TEREP | TFCMP1 | TDCMP2 | TFREP | TDREP | TDRST | TFRST | TCCMP1 | TCREP |
| 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 |
| TCRST | TBCMP2 | TBCMP1 | TBREP | TBRST | TACMP2 | TACMP1 | TAREP | TARST | MSTCMP4 | MSTCMP3 | MSTCMP2 | MSTCMP1 | MSTREP | MSTRST | SW |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **OCHPEV**：片上事件（On-chip event）

当 hrtim_bm_trg 输入（连接到通用定时器 TRGO 输出）上发生上升沿时，将触发突发模式进入。

Bit 30 **EEV8**：外部事件 8 (External Event 8)（应用 TIMD 过滤器）

通过 TIMD 过滤器调节的外部事件 8 会启动突发模式工作。

Bit 29 **EEV7**：外部事件 7 (External Event 7)（应用 TIMA 过滤器）

通过 TIMA 过滤器调节的外部事件 7 会启动突发模式工作。

Bit 28 **TDEEV8**：外部事件 8 后的定时器 D 周期 (Timer D period following External Event 8)

外部事件 8（通过 TIMD 过滤器调节）后的定时器 D 周期会启动突发模式工作。

Bit 27 **TAEEV7**：外部事件 7 后的定时器 A 周期 (Timer A period following External Event 7)

外部事件 7（通过 TIMA 过滤器调节）后的定时器 A 周期会启动突发模式工作。

Bit 26 **TECMP2**：定时器 E 比较 2 事件（Timer E compare 2 event）

参考 TACMP1 的描述。

Bit 25 **TECMP1**：定时器 E 比较 1 事件（Timer E compare 1 event）

参考 TACMP1 的描述。

Bit 24 **TEREP**：定时器 E 重复（Timer E repetition）

参考 TAREP 的描述。

Bit 23 **TFCMP1**：定时器 F 比较 1 事件（Timer F compare 1 event）

参考 TACMP1 的描述。

Bit 22 **TDCMP2**：定时器 D 比较 2 事件（Timer D compare 2 event）

参考 TACMP1 的描述。

Bit 21 **TFREP**：定时器 F 重复（Timer F repetition）

参考 TAREP 的描述。

Bit 20 **TDREP**：定时器 D 重复（Timer D repetition）

参考 TAREP 的描述。

Bit 19 **TDRST**：定时器 D 复位或翻转（Timer D reset or roll-over）

参考 TARST 的描述。

Bit 18 **TFRST**：定时器 F 复位（Timer F reset）

参考 TARST 的描述。

Bit 17 **TCCMP1**：定时器 C 比较 1 事件（Timer C compare 1 event）
参考 TACMP1 的描述。

Bit 16 **TCREP**：定时器 C 重复（Timer C repetition）
参考 TAREP 的描述。

Bit 15 **TCRST**：定时器 C 复位或翻转（Timer C reset or roll-over）
参考 TARST 的描述。

Bit 14 **TBCMP2**：定时器 B 比较 2 事件（Timer B compare 2 event）
参考 TACMP1 的描述。

Bit 13 **TBCMP1**：定时器 B 比较 1 事件（Timer B compare 1 event）
参考 TACMP1 的描述。

Bit 12 **TBREP**：定时器 B 重复（Timer B repetition）
参考 TAREP 的描述。

Bit 11 **TBRST**：定时器 B 复位或翻转（Timer B reset or roll-over）
参考 TARST 的描述。

Bit 10 **TACMP2**：定时器 A 比较 2 事件（Timer A compare 2 event）
参考 TACMP1 的描述。

Bit 9 **TACMP1**：定时器 A 比较 1 事件（Timer A compare 1 event）
定时器 A 的比较 1 事件将启动突发模式操作。

Bit 8 **TAREP**：定时器 A 重复（Timer A repetition）
定时器 A 的重复事件将启动突发模式操作。

Bit 7 **TARST**：定时器 A 复位或翻转（Timer A reset or roll-over）
定时器 A 的复位或翻转事件将启动突发模式操作。

Bit 6 **MSTCMP4**：主定时器比较 4（Master compare 4）
参考 MSTCMP1 的描述。

Bit 5 **MSTCMP3**：主定时器比较 3（Master compare 3）
参考 MSTCMP1 的描述。

Bit 4 **MSTCMP2**：主定时器比较 2（Master compare 2）
参考 MSTCMP1 的描述。

Bit 3 **MSTCMP1**：主定时器比较 1（Master compare 1）
主定时器的比较 1 事件将启动突发模式工作。

Bit 2 **MSTREP**：主定时器重复（Master repetition）
主定时器重复事件将启动突发模式工作。

Bit 1 **MSTRST**：主定时器复位或翻转（Master reset or roll-over）
主定时器复位和翻转事件将启动突发模式工作。

Bit 0 **SW**：软件启动（Software start）
此位由软件置 1，并由硬件自动复位。
如果此位置 1，则会立即启动突发模式工作。
如果突发模式未使能（BME 位复位），此位无效。

### 21.5.61 HRTIM 突发模式比较寄存器（HRTIM_BMCMPR）

偏移地址：0x3A8

复位值：0x0000 0000

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **BMCMP[15:0]**：突发模式比较值（Burst mode compare value）

这个位域定义了所选定时器处于空闲状态的周期数。

该寄存器保留预装载寄存器的内容，如果禁止预装载，则会保留活动寄存器的内容。

*注：如果使用未进行预分频的* $f_{HRTIM}$ *时钟作为突发模式时钟源（*BMCLK[3:0] = 1010 *且* BMPRESC[3:0] = 0000*）时，*BMCMP[15:0]*不能设置为*0x0000。

### 21.5.62 HRTIM 突发模式周期寄存器（HRTIM_BMPER）

偏移地址：0x3AC

复位值：0x0000 0000

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

Bit 31:16 保留，必须保持复位值。

Bit 15:0 **BMPER[15:0]**：突发模式周期（Burst mode period）

定义突发模式的重复周期。

如果禁止预装载，则该寄存器保存预装载寄存器的内容或活动寄存器的内容。

*注：当突发模式被使能时，*BMPER[15:0]*的值不能为零。*

### 21.5.63 HRTIM 定时器外部事件控制寄存器 1 (HRTIM_EECR1)

偏移地址：0x3B0

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | EE5FAST | EE5SNS[1:0] | | EE5POL | EE5SRC[1:0] | | EE4FAST | EE4SNS[1:0] | | EE4POL | EE4SRC[1:0] | | EE3FAST | EE3SNS[1] |
| | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EE3SNS[0] | EE3POL | EE3SRC[1:0] | | EE2FAST | EE2SNS[1:0] | | EE2POL | EE2SRC[1:0] | | EE1FAST | EE1SNS[1:0] | | EE1POL | EE1SRC[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:30 保留，必须保持复位值。

Bit 29 **EE5FAST**：外部事件 5 快速模式（External event 5 fast mode）

参考 EE1FAST 的描述。

Bit 28:27 **EE5SNS[1:0]**：外部事件 5 有效性（External event 5 sensitivity）

参考 EE1SNS[1:0]的描述。

Bit 26 **EE5POL**：外部事件 5 极性（External event 5 polarity）

参考 EE1POL 的描述。

Bit 25:24 **EE5SRC[1:0]**：外部事件 5 源（External event 5 source）

参考 EE1SRC[1:0]的描述。

Bit 23 **EE4FAST**：外部事件 4 快速模式（External event 4 fast mode）

参考 EE1FAST 的描述。

Bit 22:21 **EE4SNS[1:0]**：外部事件 4 有效性（External event 4 sensitivity）

参考 EE1SNS[1:0]的描述。

Bit 20 **EE4POL**：外部事件 4 极性（External event 4 polarity）

参考 EE1POL 的描述。

Bit 19:18 **EE4SRC[1:0]**：外部事件 4 源（External event 4 source）

参考 EE1SRC[1:0]的描述。

Bit 17 **EE3FAST**：外部事件 3 快速模式（External event 3 fast mode）

参考 EE1FAST 的描述。

Bit 16:15 **EE3SNS[1:0]**：外部事件 3 有效性（External event 3 sensitivity）

参考 EE1SNS[1:0]的描述。

Bit 14 **EE3POL**：外部事件 3 极性（External event 3 polarity）

参考 EE1POL 的描述。

Bit 13:12 **EE3SRC[1:0]**：外部事件 3 源（External event 3 source）

参考 EE1SRC[1:0]的描述。

Bit 11 **EE2FAST**：外部事件 2 快速模式（External event 2 fast mode）

参考 EE1FAST 的描述。

Bit 10:9 **EE2SNS[1:0]**：外部事件 2 有效性（External event 2 sensitivity）

参考 EE1SNS[1:0]的描述。

Bit 8 **EE2POL**：外部事件 2 极性（External event 2 polarity）

参考 EE1POL 的描述。

Bit 7:6 **EE2SRC[1:0]**：外部事件 2 源（External event 2 source）

参考 EE1SRC[1:0]的描述。

Bit 5 **EE1FAST**：外部事件 1 快速模式（External event 1 fast mode）

0：外部事件 1 会在作用于输出之前通过 HRTIM 逻辑进行重新同步，以增加 $f_{HRTIM}$ 时钟相关延迟

1：外部事件异步作用于输出上（低延迟模式）

*注：使用了事件的计数器使能后（*TxCEN *位置*1*），不得修改此位。*

Bit 4:3 **EE1SNS[1:0]**：外部事件 1 有效性（External event 1 sensitivity）

00：由 EE1POL 位定义有效电平

01：上升沿，与 EE1POL 位值无关

10：下降沿，与 EE1POL 位值无关

11：上升沿和下降沿，与 EE1POL 位值无关

Bit 2 **EE1POL**：外部事件 1 极性（External event 1 polarity）

仅当 EE1SNS[1:0] = 00 时，此位才有效。

0：外部事件在高电平时有效

1：外部事件在低电平时有效

*注：一旦定时器*x *被使能，此参数就不能更改。它必须在设置*EE1FAST *位之前配置。*

Bit 1:0 **EE1SRC[1:0]**：外部事件 1 源（External event 1 source）

此位域用于选择外部事件 1 的源。。

00：hrtim_eev1_1

01：hrtim_eev1_2

10：hrtim_eev1_3

11：hrtim_eev1_4

*注：一旦定时器*x *被使能，此参数就不能更改。它必须在设置*EE1FAST *位之前配置。*

### 21.5.64 HRTIM 定时器外部事件控制寄存器 2 (HRTIM_EECR2)

偏移地址：0x3B4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | EE10SNS[1:0] | | EE10POL | EE10SRC[1:0] | | Res | EE9SNS[1:0] | | EE9POL | EE9SRC[1:0] | | Res | EE8SNS[1] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EE8SNS[0] | EE8POL | EE8SRC[1:0] | | Res | EE7SNS[1:0] | | EE7POL | EE7SRC[1:0] | | Res | EE6SNS[1:0] | | EE6POL | EE6SRC[1:0] | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持复位值。

Bit 28:27 **EE10SNS[1:0]**：外部事件 10 有效性（External event 10 sensitivity）
参考 EE1SNS[1:0]的描述。

Bit 26 **EE10POL**：外部事件 10 极性（External event 10 polarity）
参考 EE1POL 的描述。

Bit 25:24 **EE10SRC[1:0]**：外部事件 10 源（External event 10 source）
参考 EE1SRC[1:0]的描述。

Bit 23 保留，必须保持复位值。

Bit 22:21 **EE9SNS[1:0]**：外部事件 9 有效性（External event 9 sensitivity）
参考 EE1SNS[1:0]的描述。

Bit 20 **EE9POL**：外部事件 9 极性（External event 9 polarity）
参考 EE1POL 的描述。

Bit 19:18 **EE9SRC[1:0]**：外部事件 9 源（External event 9 source）
参考 EE1SRC[1:0]的描述。

Bit 17 保留，必须保持复位值。

Bit 16:15 **EE8SNS[1:0]**：外部事件 8 有效性（External event 8 sensitivity）
参考 EE1SNS[1:0]的描述。

Bit 14 **EE8POL**：外部事件 8 极性（External event 8 polarity）
参考 EE1POL 的描述。

Bit 13:12 **EE8SRC[1:0]**：外部事件 8 源（External event 8 source）
参考 EE1SRC[1:0]的描述。

Bit 11 保留，必须保持复位值。

Bit 10:9 **EE7SNS[1:0]**：外部事件 7 有效性（External event 7 sensitivity）
参考 EE1SNS[1:0]的描述。

Bit 8 **EE7POL**：外部事件 7 极性（External event 7 polarity）
参考 EE1POL 的描述。

Bit 7:6 **EE7SRC[1:0]**：外部事件 7 源（External event 7 source）
参考 EE1SRC[1:0]的描述。

Bit 5 保留，必须保持复位值。

Bit 4:3 **EE6SNS[1:0]**：外部事件 6 有效性（External event 6 sensitivity）
请参考 EE1SNS[1:0]的描述。

Bit 2 **EE6POL**：外部事件 6 极性（External event 6 polarity）
请参考 EE1POL 的描述。

Bit 1:0 **EE6SRC[1:0]**：外部事件 6 源（External event 6 source）
请参考 EE1SRC[1:0]的描述。

### 21.5.65 HRTIM 定时器外部事件控制寄存器 3 (HRTIM_EECR3)

偏移地址：0x3B8
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EEVSD[1:0] | | Reserved | | EE10F[3:0] | | | | Reserved | | EE9F[3:0] | | | | Reserved | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EE8F[3:0] | | | | Reserved | | EE7F[3:0] | | | | Reserved | | EE6F[3:0] | | | |
| rw | rw | rw | rw | | | rw | rw | rw | rw | | | rw | rw | rw | rw |

Bit 31:30 **EEVSD[1:0]**：外部事件采样时钟分频比（External event sampling clock division）
此位域指示定时器时钟频率（$f_{HRTIM}$）与数字滤波器使用的外部事件信号采样时钟（$f_{EEVS}$）之间的分频比。
00：$f_{EEVS} = f_{HRTIM}$
01：$f_{EEVS} = f_{HRTIM} / 2$
10：$f_{EEVS} = f_{HRTIM} / 4$
11：$f_{EEVS} = f_{HRTIM} / 8$

Bit 29:28 保留，必须保持复位值。

Bit 27:24 **EE10F[3:0]**：外部事件 10 过滤器（External event 10 filter）
请参考 EE6F[3:0] 的描述。

Bit 23:22 保留，必须保持复位值。

Bit 21:18 **EE9F[3:0]**：外部事件 9 过滤器（External event 9 filter）
请参考 EE6F[3:0] 的描述。

Bit 17:16 保留，必须保持复位值。

Bit 15:12 **EE8F[3:0]**：外部事件 8 过滤器（External event 8 filter）
请参考 EE6F[3:0] 的描述。

Bit 11:10 保留，必须保持复位值。

Bit 9:6 **EE7F[3:0]**：外部事件 7 过滤器（External event 7 filter）

参考 EE6F[3:0]的描述。

Bit 4:5 保留，必须保持复位值。

Bit 3:0 **EE6F[3:0]**：外部事件 6 过滤器（External event 6 filter）

这个位域定义了用于采样外部事件 6 输入的频率以及应用于 EEV6 的数字滤波器的长度。数字滤波器由计数器组成，需要使用 N 个有效样本来验证输出跳变。

0000：滤波器禁用

0001：$f_{SAMPLING} = f_{HRTIM}$，N = 2

0010：$f_{SAMPLING} = f_{HRTIM}$，N = 4

0011：$f_{SAMPLING} = f_{HRTIM}$，N = 8

0100：$f_{SAMPLING} = f_{EEVS}$ /2，N = 6

0101：$f_{SAMPLING} = f_{EEVS}$ /2，N = 8

0110：$f_{SAMPLING} = f_{EEVS}$ /4，N = 6

0111：$f_{SAMPLING} = f_{EEVS}$ /4，N = 8

1000：$f_{SAMPLING} = f_{EEVS}$ /8，N = 6

1001：$f_{SAMPLING} = f_{EEVS}$ /8，N = 8

1010：$f_{SAMPLING} = f_{EEVS}$ /16，N = 5

1011：$f_{SAMPLING} = f_{EEVS}$ /16，N = 6

1100：$f_{SAMPLING} = f_{EEVS}$ /16，N = 8

1101：$f_{SAMPLING} = f_{EEVS}$ /32，N = 5

1110：$f_{SAMPLING} = f_{EEVS}$ /32，N = 6

1111：$f_{SAMPLING} = f_{EEVS}$ /32，N = 8

### 21.5.66 HRTIM ADC 触发 1 寄存器（HRTIM_ADC1R）

偏移地址：0x3BC

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC1 TEPER | ADC1 TEC4 | ADC1 TEC3 | ADC1 TFRST | ADC1 TDPER | ADC1 TDC4 | ADC1 TDC3 | ADC1 TFPER | ADC1 TCPER | ADC1 TCC4 | ADC1 TCC3 | ADC1 TFC4 | ADC1 TBRST | ADC1 TBPER | ADC1 TBC4 | ADC1 TBC3 |
| 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 |
| ADC1 TFC3 | ADC1 TARST | ADC1 TAPER | ADC1 TAC4 | ADC1 TAC3 | ADC1 TFC2 | ADC1 EEV5 | ADC1 EEV4 | ADC1 EEV3 | ADC1 EEV2 | ADC1 EEV1 | ADC1 MPER | ADC1 MC4 | ADC1 MC3 | ADC1 MC2 | ADC1 MC1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参考 HRTIM_ADC3R 位的描述。

### 21.5.67 HRTIM ADC 触发 2 寄存器（HRTIM_ADC2R）

偏移地址：0x3C0

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC2 TERST | ADC2 TEC4 | ADC2 TEC3 | ADC2 TEC2 | ADC2 TDRST | ADC2 TDPER | ADC2 TDC4 | ADC2 TFPER | ADC2 TDC2 | ADC2 TCPER | ADC2 TCPER | ADC2 TCC4 | ADC2 TFC4 | ADC2 TCC2 | ADC2 TBPER | ADC2 TBC4 |
| 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 |
| ADC2 TFC3 | ADC2 TBC2 | ADC2 TAPER | ADC2 TAC4 | ADC2 TFC2 | ADC2 TAC2 | ADC2 EEV10 | ADC2 EEV9 | ADC2 EEV8 | ADC2 EEV7 | ADC2 EEV6 | ADC2 MPER | ADC2 MC4 | ADC2 MC3 | ADC2 MC2 | ADC2 MC1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 参考 HRTIM_ADC4R 位的描述。

### 21.5.68 HRTIM ADC 触发 3 寄存器（HRTIM_ADC3R）

偏移地址：0x3C4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC3 TEPER | ADC3 TEC4 | ADC3 TEC3 | ADC3 TFRST | ADC3 TDPER | ADC3 TDC4 | ADC3 TDC3 | ADC3 TFPER | ADC3 TCPER | ADC3 TCC4 | ADC3 TCC3 | ADC3 TFC4 | ADC3 TBRST | ADC3 TBPER | ADC3 TBC4 | ADC3 TBC3 |
| 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 |
| ADC3 TFC3 | ADC3 TARST | ADC3 TAPER | ADC3 TAC4 | ADC3 TAC3 | ADC3 TFC2 | ADC3 EEV5 | ADC3 EEV4 | ADC3 EEV3 | ADC3 EEV2 | ADC3 EEV1 | ADC3 MPER | ADC3 MC4 | ADC3 MC3 | ADC3 MC2 | ADC3 MC1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **ADC3TEPER**：定时器 E 周期时的 ADC 触发 3
参考 ADC3TAPER 的描述。

Bit 30 **ADC3TEC4**：定时器 E 比较 4 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 29 **ADC3TEC3**：定时器 E 比较 3 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 28 **ADC3TFRST**：定时器 F 重置和计数器翻转时的 ADC 触发 3 (1)
参考 ADC3TARST 的描述。

Bit 27 **ADC3TDPER**：定时器 D 周期时的 ADC 触发 3
参考 ADC3TAPER 的描述。

Bit 26 **ADC3TDC4**：定时器 D 比较 4 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 25 **ADC3TDC3**：定时器 D 比较 3 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 24 **ADC3TFPER**：定时器 F 周期时的 ADC 触发 3
参考 ADC3TAPER 的描述。

Bit 23 **ADC3TCPER**：定时器 C 周期时的 ADC 触发 3
参考 ADC3TAPER 的描述。

Bit 22 **ADC3TCC4**：定时器 C 比较 4 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 21 **ADC3TCC3**：定时器 C 比较 3 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 20 **ADC3TFC4**：定时器 F 比较 4 时的 ADC 触发 3
参考 ADC3TAC2 的描述。

Bit 19 **ADC3TBRST**：定时器 B 复位和计数器翻转时的 ADC 触发 3 (1)
此位可在发生定时器 B 复位和翻转事件时在 ADC 触发 1 输出上生成 ADC 触发信号。

Bit 18 **ADC3TBPER**：定时器 B 周期时的 ADC 触发 3
参考 ADC3TAPER 的描述。

Bit 17 **ADC3TBC4**：定时器 B 比较 4 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 16 **ADC3TBC3**：定时器 B 比较 3 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 15 **ADC3TFC3**：定时器 B 比较 2 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 14 **ADC3TARST**：定时器 A 复位和计数器翻转时的 ADC 触发 3
此位可在发生定时器 A 复位和翻转事件时在 ADC 触发 1 输出上生成 ADC 触发信号。

Bit 13 **ADC3TAPER**：定时器 A 周期时的 ADC 触发 3
此位可在发生定时器 A 周期事件时在 ADC 触发 3 输出上生成 ADC 触发信号。

Bit 12 **ADC3TAC4**：定时器 A 比较 4 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 11 **ADC3TAC3**：定时器 A 比较 3 时的 ADC 触发 3
参考 ADC3TFC2 的描述。

Bit 10 **ADC3TFC2**：定时器 F 比较 2 时的 ADC 触发 3
此位可在发生定时器 F 比较 2 事件时在 ADC 触发 3 输出上生成 ADC 触发信号。

Bit 9 **ADC3EEV5**：外部事件 5 时的 ADC 触发 3 (1)
参考 ADC3EEV1 的描述。

Bit 8 **ADC3EEV4**：外部事件 4 时的 ADC 触发 3 (1)
参考 ADC3EEV1 的描述。

Bit 7 **ADC3EEV3**：外部事件 3 时的 ADC 触发 3 (1)
参考 ADC3EEV1 的描述。

Bit 6 **ADC3EEV2**：外部事件 2 时的 ADC 触发 3 (1)
参考 ADC3EEV1 的描述。

Bit 5 **ADC3EEV1**：外部事件 1 时的 ADC 触发 3 (1)
此位可在发生外部事件 1 时在 ADC 触发 3 输出上生成 ADC 触发信号。

Bit 4 **ADC3MPER**：主定时器周期时的 ADC 触发 3
此位可在发生主定时器周期事件时在 ADC 触发 3 输出上生成 ADC 触发信号。

Bit 3 **ADC3MC4**：主定时器比较 4 时的 ADC 触发 3
参考 ADC3MC1 的描述。

Bit 2 **ADC3MC3**：主定时器比较 3 时的 ADC 触发 3
参考 ADC3MC1 的描述。

Bit 1 **ADC3MC2**：主定时器比较 2 时的 ADC 触发 3
参考 ADC3MC1 的描述。

Bit 0 **ADC3MC1**：主定时器比较 1 时的 ADC 触发 3
此位可在发生主定时器比较 1 事件时在 ADC 触发 3 输出上生成 ADC 触发信号。

1. 这些触发与 HRTIM_ADC2R/HRTIM_ADC4R 和 HRTIM_ADC1R/HRTIM_ADC3R 是不同的。

### 21.5.69 HRTIM ADC 触发 4 寄存器（HRTIM_ADC4R）

偏移地址：0x3C8

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC4 TERST | ADC4 TEC4 | ADC4 TEC3 | ADC4 TEC2 | ADC4 TDRST | ADC4 TDPER | ADC4 TDC4 | ADC4 TFPER | ADC4 TDC2 | ADC4 TCPER | ADC4 TCPER | ADC4 TCC4 | ADC4 TFC4 | ADC4 TCC2 | ADC4 TBPER | ADC4 TBC4 |
| 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 |
| ADC4 TFC3 | ADC4 TBC2 | ADC4 TAPER | ADC4 TAC4 | ADC4 TFC2 | ADC4 TAC2 | ADC4 EEV10 | ADC4 EEV9 | ADC4 EEV8 | ADC4 EEV7 | ADC4 EEV6 | ADC4 MPER | ADC4 MC4 | ADC4 MC3 | ADC4 MC2 | ADC4 MC1 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **ADC4TERST**：定时器 E 复位和计数器翻转时的 ADC 触发 4 (1)

参考 ADC4TCRST 的描述。

Bit 30 **ADC4TEC4**：定时器 E 比较 4 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 29 **ADC4TEC3**：定时器 E 比较 3 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 28 **ADC4TEC2**：定时器 E 比较 2 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 27 **ADC4TDRST**：定时器 D 复位和计数器翻转时的 ADC 触发 4 (1)

参考 ADC4TCRST 的描述。

Bit 26 **ADC4TDPER**：定时器 D 周期时的 ADC 触发 4

参考 ADC4TAPER 的描述。

Bit 25 **ADC4TDC4**：定时器 D 比较 4 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 24 **ADC4TFPER**：定时器 F 周期时的 ADC 触发 4

参考 ADC4TAPER 的描述。

Bit 23 **ADC4TDC2**：定时器 D 比较 2 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 22 **ADC4TCRST**：定时器 C 复位和计数器翻转时的 ADC 触发 4 (1)

此位可在发生定时器 C 复位和翻转事件时在 ADC 触发 4 输出上生成 ADC 触发信号。

Bit 21 **ADC4TCPER**：定时器 C 周期时的 ADC 触发 4

请参见 ADC4TAPER 说明

Bit 20 **ADC4TCC4**：定时器 C 比较 4 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 19 **ADC4TFC4**：定时器 F 比较 3 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 18 **ADC4TCC2**：定时器 C 比较 2 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 17 **ADC4TBPER**：定时器 B 周期时的 ADC 触发 4

参考 ADC4TAPER 的描述。

Bit 16 **ADC4TBC4**：定时器 B 比较 4 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 15 **ADC4TFC3**：定时器 F 比较 3 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 14 **ADC4TBC2**：定时器 B 比较 2 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 13 **ADC4TAPER**：定时器 A 周期时的 ADC 触发 4

此位可在发生定时器 A 事件时在 ADC 触发 4 输出上生成 ADC 触发信号。

Bit 12 **ADC4TAC4**：定时器 A 比较 4 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 11 **ADC4TFC2**：定时器 F 比较 2 时的 ADC 触发 4

参考 ADC4TAC2 的描述。

Bit 10 **ADC4TAC2**：定时器 A 比较 2 时的 ADC 触发 4

此位可在发生定时器 A 比较 2 时在 ADC 触发 4 输出上生成 ADC 触发信号。

Bit 9 **ADC4EEV10**：外部事件 10 时的 ADC 触发 4 (1)

参考 ADC4EEV6 的描述。

Bit 8 **ADC4EEV9**：外部事件 9 时的 ADC 触发 4 (1)

参考 ADC4EEV6 的描述。

Bit 7 **ADC4EEV8**：外部事件 8 时的 ADC 触发 4 (1)

参考 ADC4EEV6 的描述。

Bit 6 **ADC4EEV7**：外部事件 7 时的 ADC 触发 4 (1)

参考 ADC4EEV6 的描述。

Bit 5 **ADC4EEV6**：外部事件 6 时的 ADC 触发 4 (1)

此位可在发生外部事件 6 时在 ADC 触发 4 输出上生成 ADC 触发信号。

Bit 4 **ADC4MPER**：主定时器周期时的 ADC 触发 4

此位可在发生主定时器周期事件时在 ADC 触发 4 输出生成 ADC 触发信号。

Bit 3 **ADC4MC4**：主定时器比较 4 时的 ADC 触发 4

参考 ADC4MC1 的描述。

Bit 2 **ADC4MC3**：主定时器比较 3 时的 ADC 触发 4

参考 ADC4MC1 的描述。

Bit 1 **ADC4MC2**：主定时器比较 2 时的 ADC 触发 4

参考 ADC4MC1 的描述。

Bit 0 **ADC4MC1**：主比定时器较 1 时的 ADC 触发 4

此位可在发生主定时器比较 1 事件时在 ADC 触发 4 输出上生成 ADC 触发信号。

1. 这些触发与 HRTIM_ADC1R/HRTIM_ADC3R 和 HRTIM_ADC2R/HRTIM_ADC4R 不同。

### 21.5.70 HRTIM DLL 控制寄存器（HRTIM_DLLCR）

偏移地址：0x3CC

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CKS | Reserved | | | | CALRTE[1:0] | | CALEN | CAL |
| | | | | | | | rw | | | | | rw | rw | rw | rw |

Bit 31:9 保留，必须保持复位值。

Bit 8 **CKS**：HRTIM 时钟状态（HRTIM clock status）

此位定义了 DLL 当前所处 HRTIM 时钟的状态。

0：$f_{HRTIM}$ = 80MHz ~ 140MHz

1：$f_{HRTIM}$ = 140MHz ~ 192MHz

Bit 7:4 保留，必须保持复位值。

Bit 3:2 **CALRTE[1:0]**：DLL 校准速率（DLL Calibration rate）

这定义了 DLL 校准的周期性。

00：1048576 * $t_{HRTIM}$（对于 $f_{HRTIM}$ = 192MHz，为 5.461ms）

01：131072 * $t_{HRTIM}$（对于 $f_{HRTIM}$ = 192MHz，为 683µs）

10：16384 * $t_{HRTIM}$（对于 $f_{HRTIM}$ = 192MHz，为 85µs）

11：2048 * $t_{HRTIM}$（对于 $f_{HRTIM}$ = 192MHz，为 11µs）

Bit 1 **CALEN**：DLL 校准使能（DLL Calibration Enable）

此位根据 CALRTE[1:0]位设置使能周期性 DLL 校准。当 CALEN 位复位时，可以使用 CAL 位以单次模式启动校准。

0：禁用周期性校准

1：根据 CALRTE[1:0] 设置进行周期性校准

*注：CALEN 和 CAL 位不能同时设置*

Bit 0 **CAL**：DLL 校准开始（DLL Calibration Start）

此位启动 DLL 校准过程。它只写不读。

0：无校准请求

1：开始校准

*注：CAL 和 CALEN 位不能同时设置*

### 21.5.71 HRTIM 故障输入寄存器 1 (HRTIM_FLTINR1)

偏移地址：0x3D0

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT4LCK | FLT4F[3:0] | | | | FLT4SRC[0] | FLT4P | FLT4E | FLT3LCK | FLT3F[3:0] | | | | FLT3SRC[0] | FLT3P | FLT3E |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT2LCK | FLT2F[3:0] | | | | FLT2SRC[0] | FLT2P | FLT2E | FLT1LCK | FLT1F[3:0] | | | | FLT1SRC[0] | FLT1P | FLT1E |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **FLT4LCK**：故障 4 锁定（Fault 4 lock）

参考 HRTIM_FLTINR2 寄存器中的 FLT5LCK 描述。

Bit 30:27 **FLT4F[3:0]**：故障 4 过滤器（Fault 4 filter）

参考 HRTIM_FLTINR2 寄存器中的 FLT5F[3:0]描述。

Bit 26 **FLT4SRC**：故障 4 源位 0（Fault 4 source bit 0）

参考 HRTIM_FLTINR2 寄存器中的 FLT5SRC[0]描述。

Bit 25 **FLT4P**：故障 4 极性（Fault 4 polarity）

参考 HRTIM_FLTINR2 寄存器中的 FLT5P 描述。

Bit 24 **FLT4E**：故障 4 使能（Fault 4 enable）

参考 HRTIM_FLTINR2 寄存器中的 FLT5E 描述。

Bit 23 **FLT3LCK**：故障 3 锁定（Fault 3 lock）

参考 HRTIM_FLTINR2 寄存器中的 FLT5LCK 描述。

Bit 22:19 **FLT3F[3:0]**：故障 3 过滤器（Fault 3 filter）

参考 HRTIM_FLTINR2 寄存器中的 FLT5F[3:0]描述。

Bit 18 **FLT3SRC[0]**：故障 3 源位 0（Fault 3 source bit 0）

参考 HRTIM_FLTINR2 寄存器中的 FLT5SRC[0]描述。

Bit 17 **FLT3P**：故障 3 极性（Fault 3 polarity）

参考 HRTIM_FLTINR2 寄存器中的 FLT5P 描述。

Bit 16 **FLT3E**：故障 3 使能（Fault 3 enable）

参考 HRTIM_FLTINR2 寄存器中的 FLT5E 描述。

Bit 15 **FLT2LCK**：故障 2 锁定（Fault 2 lock）

参考 HRTIM_FLTINR2 寄存器中的 FLT5LCK 描述。

Bit 14:11 **FLT2F[3:0]**：故障 2 过滤器（Fault 2 filter）

参考 HRTIM_FLTINR2 寄存器中的 FLT5F[3:0]描述。

Bit 10 **FLT2SRC[0]**：故障 2 源位 0（Fault 2 source bit 0）

参考 HRTIM_FLTINR2 寄存器中的 FLT5SRC[0]描述。

Bit 9 **FLT2P**：故障 2 极性（Fault 2 polarity）

参考 HRTIM_FLTINR2 寄存器中的 FLT2P 描述。

Bit 8 **FLT2E**：故障 2 使能（Fault 2 enable）

参考 HRTIM_FLTINR2 寄存器中的 FLT5E 描述。

Bit 7 **FLT1LCK**：故障 1 锁定（Fault 1 lock）

参考 HRTIM_FLTINR2 寄存器中的 FLT5LCK 描述。

Bit 6:3 **FLT1F[3:0]**：故障 1 过滤器（Fault 1 filter）

参考 HRTIM_FLTINR2 寄存器中的 FLT5F[3:0]描述。

Bit 2 **FLT1SRC[0]**：故障 1 源位 0（Fault 1 source bit 0）

参考 HRTIM_FLTINR2 寄存器中的 FLT5SRC[0]描述。

Bit 1 **FLT1P**：故障 1 极性（Fault 1 polarity）

参考 HRTIM_FLTINR2 寄存器中的 FLT5P 描述。

Bit 0 **FLT1E**：故障 1 使能（Fault 1 enable）

参考 HRTIM_FLTINR2 寄存器中的 FLT5E 描述。

### 21.5.72 HRTIM 故障输入寄存器 2 (HRTIM_FLTINR2)

偏移地址：0x3D4

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | FLTSD[1:0] | | Reserved | | FLT6 SRC[1] | FLT5 SRC[1] | FLT4 SRC[1] | FLT3 SRC[1] | FLT2 SRC[1] | FLT1 SRC[1] |
| | | | | | | rw | rw | | | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT6 LCK | FLT6F[3:0] | | | | FLT6 SRC[0] | FLT6P | FLT6E | FLT5 LCK | FLT5F[3:0] | | | | FLT5 SRC[0] | FLT5P | FLT5E |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:26 保留，必须保持为复位值。

Bit 25:24 **FLTSD[1:0]**：故障采样时钟分频（Fault sampling clock division）

此位域指示定时器时钟频率（$f_{HRTIM}$）与数字滤波器使用的故障信号采样时钟（$f_{FLTS}$）之间的分频比。

00：$f_{FLTS} = f_{HRTIM}$

01：$f_{FLTS} = f_{HRTIM} / 2$

10：$f_{FLTS} = f_{HRTIM} / 4$

11：$f_{FLTS} = f_{HRTIM} / 8$

*注：此位域必须先于任何* FLTxE *使能位写入。*

Bit 23:22 保留，必须保持为复位值。

Bit 21 **FLT6SRC[1]**：故障 6 源位 1（Fault 6 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 20 **FLT5SRC[1]**：故障 5 源位 1（Fault 5 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 19 **FLT4SRC[1]**：故障 4 源位 1（Fault 4 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 18 **FLT3SRC[1]**：故障 3 源位 1（Fault 3 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 17 **FLT2SRC[1]**：故障 2 源位 1（Fault 2 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 16 **FLT1SRC[1]**：故障 1 源位 1（Fault 1 source bit 1）

参考 FLT5SRC[0]的描述。

Bit 15 **FLT6LCK**：故障 6 锁定（Fault 6 lock）

参考 FLT5LCK 的描述。

Bit 14:11 **FLT6F[3:0]**：故障 6 过滤器（Fault 6 filter）

参考 FLT5F[3:0]的描述。

Bit 10 **FLT6SRC[0]**：故障 6 源位 0（Fault 6 source bit 0）

参考 FLT5SRC[0]的描述。

Bit 9 **FLT6P**：故障 6 极性（Fault 6 polarity）

参考 FLT5P 的描述。

Bit 8 **FLT6E**：故障 6 使能（Fault 6 enable）

参考 FLT5E 的描述。

Bit 7 **FLT5LCK**：故障 5 锁定（Fault 5 lock）

FLT5LCK 位会修改故障编程位的写属性，以防止误写访问修改写属性。

该位仅可写入一次。该位置 1 后，在下一次系统复位之前不能进行修改。

0：FLT5E、FLT5P、FLT5SRC[1:0]、FLT5F[3:0]和 FLT5BLK 位是可读写的。

1：FLT5E、FLT5P、FLT5SRC[1:0]、FLT5F[3:0]和 FLT5BLK 位不能被写入（只读模式）。

Bit 6:3 **FLT5F[3:0]**：故障 5 滤波器（Fault 5 filter）

此位域可定义 FLT5 输入的采样频率和应用于 FLT5 的数字滤波器长度。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，FLT5 异步工作

0001：$f_{SAMPLING} = f_{HRTIM}$，N = 2

0010：$f_{SAMPLING} = f_{HRTIM}$，N = 4

0011：$f_{SAMPLING} = f_{HRTIM}$，N = 8

0100：$f_{SAMPLING} = f_{FLTS}/2$，N = 6

0101：$f_{SAMPLING} = f_{FLTS}/2$，N = 8

0110：$f_{SAMPLING} = f_{FLTS}/4$，N = 6

0111：$f_{SAMPLING} = f_{FLTS}/4$，N = 8

1000：$f_{SAMPLING} = f_{FLTS}/8$，N = 6

1001：$f_{SAMPLING} = f_{FLTS}/8$，N = 8

1010：$f_{SAMPLING} = f_{FLTS}/16$，N = 5

1011：$f_{SAMPLING} = f_{FLTS}/16$，N = 6

1100：$f_{SAMPLING} = f_{FLTS}/16$，N = 8

1101：$f_{SAMPLING} = f_{FLTS}/32$，N = 5

1110：$f_{SAMPLING} = f_{FLTS}/32$，N = 6

1111：$f_{SAMPLING} = f_{FLTS}/32$，N = 8

*注：此位域仅在* FLT5E *使能位复位时写入。当* FLT5LOCK *被编程时，此位域将被修改。*

Bit 2 **FLT5SRC[0]**：故障 5 源位 0（Fault 5 source bit 0）

FLT5SRC[1:0]位域用于选择 FAULT5 输入源

00：故障 5 输入是 HRTIM_FLT5 输入引脚

01：故障 5 输入连接到 COMPx 输出

10：故障 5 输入是 EEV5_muxout 输入引脚

11：保留

*注：此位域仅在* FLT5E *使能位复位时写入。*

Bit 1 **FLT5P**：故障 5 极性（Fault 5 polarity）

此位选择 FAULT5 输入的极性。

0：故障 5 输入为低电平有效

1：故障 5 输入为高电平有效

*注：此位域仅在* FLT5E *使能位复位时写入。*

Bit 0 **FLT5E**：故障 5 使能（Fault 5 enable）

此位使能全局 FAULT5 输入电路。

0：故障 5 输入禁用

1：故障 5 输入使能

### 21.5.73 HRTIM 突发 DMA 主定时器更新寄存器（HRTIM_BDMUPR）

偏移地址：0x3D8

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | MCMP4 | MCMP3 | MCMP2 | MCMP1 | MREP | MPER | MCNT | MDIER | MICR | MCR |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:10 保留，必须保持为复位值。

Bit 9 **MCMP4**：MCMP4R 寄存器更新使能（MCMP4R register update enable）

请参考 MCR 描述。

Bit 8 **MCMP3**：MCMP3R 寄存器更新使能（MCMP3R register update enable）

请参考 MCR 描述。

Bit 7 **MCMP2**：MCMP2R 寄存器更新使能（MCMP2R register update enable）

请参考 MCR 描述。

Bit 6 **MCMP1**：MCMP1R 寄存器更新使能（MCMP1R register update enable）

请参考 MCR 描述。

Bit 5 **MREP**：MREP 寄存器更新使能（MREP register update enable）

请参考 MCR 描述。

Bit 4 **MPER**：MPER 寄存器更新使能（MPER register update enable）

请参考 MCR 描述。

Bit 3 **MCNT**：MCNTR 寄存器更新使能（MCNTR register update enable）

请参考 MCR 描述。

Bit 2 **MDIER**：MDIER 寄存器更新使能（MDIER register update enable）

请参考 MCR 描述。

Bit 1 **MICR**：MICR 寄存器更新使能（MICR register update enable）

请参考 MCR 描述。

Bit 0 **MCR**：MCR 寄存器更新使能（MCR register update enable）

此位用于定义主定时器 MCR 寄存器是否包含在要通过突发 DMA 更新的寄存器列表中。

0：MCR 寄存器不会通过突发 DMA 访问更新

1：MCR 寄存器会通过突发 DMA 访问更新

### 21.5.74 HRTIM 突发 DMA 定时器 x 更新寄存器（HRTIM_BDTxUPR）（x = A~F）

偏移地址：块 A：0x3DC
偏移地址：块 B：0x3E0
偏移地址：块 C：0x3E4
偏移地址：块 D：0x3E8
偏移地址：块 E：0x3EC
偏移地址：块 F：0x3F4
复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | TIMx EEFR3 | TIMx CR2 | TIMx FLTR | TIMx OUTR | TIMx CHPR | TIMx RSTR | TIMx EEFR2 |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMx EEFR1 | TIMx RST2R | TIMx SET2R | TIMx RST1R | TIMx SET1R | TIMx DTxR | TIMx CMP4 | TIMx CMP3 | TIMx CMP2 | TIMx CMP1 | TIMx REP | TIMx PER | TIMx CNT | TIMx DIER | TIMx ICR | TIMx CR |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 30:14 保留，必须保持复位值。

Bit 22 **TIMxEEFR3**：HRTIM_EEFxR3 寄存器更新使能
请参考 TIMxCR 描述。

Bit 21 **TIMxCR2**：HRTIM_TIMxCR2 寄存器更新使能
请参考 TIMxCR 描述。

Bit 20 **TIMxFLTR**：HRTIM_FLTxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 19 **TIMxOUTR**：HRTIM_OUTxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 18 **TIMxCHPR**：HRTIM_CHPxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 17 **TIMxRSTR**：HRTIM_RSTxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 16 **TIMxEEFR2**：HRTIM_EEFxR2 寄存器更新使能
请参考 TIMxCR 描述。

Bit 15 **TIMxEEFR1**：HRTIM_EEFxR1 寄存器更新使能
请参考 TIMxCR 描述。

Bit 14 **TIMxRST2R**：HRTIM_RST2xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 13 **TIMxSET2R**：HRTIM_SET2xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 12 **TIMxRST1R**：HRTIM_RST1xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 11 **TIMxSET1R**：HRTIM_SET1xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 10 **TIMxDTR**：HRTIM_DTxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 9 **TIMxCMP4**：HRTIM_CMP4xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 8 **TIMxCMP3**：HRTIM_CMP3xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 7 **TIMxCMP2**：HRTIM_CMP2xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 6 **TIMxCMP1**：HRTIM_CMP1xR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 5 **TIMxREP**：HRTIM_REPxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 4 **TIMxPER**：HRTIM_PERxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 3 **TIMxCNT**：HRTIM_CNTxR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 2 **TIMxDIER**：HRTIM_TIMxDIER 寄存器更新使能
请参考 TIMxCR 描述。

Bit 1 **TIMxICR**：HRTIM_TIMxICR 寄存器更新使能
请参考 TIMxCR 描述。

Bit 0 **TIMxCR**：HRTIM_TIMxCR 寄存器更新使能
此位用于定义主定时器 MCR 寄存器是否包含在要通过突发 DMA 更新的寄存器列表中。
0：HRTIM_TIMxCR 寄存器不会通过突发 DMA 访问更新
1：HRTIM_TIMxCR 寄存器会通过突发 DMA 访问更新

### 21.5.75 HRTIM 突发 DMA 数据寄存器（HRTIM_BDMADR）

偏移地址：0x3F0
复位值：0x0000 0000

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

Bit 31:0 **BDMADR[31:0]**：突发 DMA 数据寄存器（Burst DMA data register）
对该寄存器的写访问会触发以下操作：
- 将数据值复制到在 BDTxUPR 和 BDMUPR 寄存器位中使能的寄存器中
- 将寄存器指针递增到下一个要填充的位置

### 21.5.76 HRTIM ADC 扩展触发寄存器（HRTIM_ADCER）

偏移地址：0x3F8

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | ADC10TRG[4:0] | | | | | ADC9TRG[4:0] | | | | | ADC8TRG[4:0] | | | | |
| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | ADC7TRG[4:0] | | | | | ADC6TRG[4:0] | | | | | ADC5TRG[4:0] | | | | |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 保留，必须保持为复位值。

Bit 30:26 **ADC10TRG[4:0]**：ADC 触发 10 选择（ADC trigger 10 selection）

此位组选择 ADC 触发 10 的来源。

请参考 ADC6TRG[4:0]的描述。

Bit 25:21 **ADC9TRG[4:0]**：ADC 触发 9 选择（ADC trigger 9 selection）

此位组选择 ADC 触发 9 的来源。

请参考 ADC5TRG[4:0]的描述。

Bit 20:16 **ADC8TRG[4:0]**：ADC 触发 8 选择（ADC trigger 8 selection）

此位组选择 ADC 触发 8 的来源。

请参考 ADC6TRG[4:0]的描述。

Bit 15 保留，必须保持为复位值。

Bit 14:10 **ADC7TRG[4:0]**：ADC 触发 7 选择（ADC trigger 7 selection）

这些位用于选择 ADC 触发 7 的来源。

请参考 ADC5TRG[4:0]的描述。

Bit 9:5 **ADC6TRG[4:0]**：ADC 触发 6 选择（ADC trigger 6 selection）

这些位用于选择 ADC 触发 6 的来源。具体的选项如下：

0：触发于主定时器比较 1
1：触发于主定时器比较 2
2：触发于主定时器比较 3
3：触发于主定时器比较 4
4：触发于主定时器周期
5：触发于外部事件 6
6：触发于外部事件 7
7：触发于外部事件 8
8：触发于外部事件 9
9：触发于外部事件 10
10：触发于定时器 A 比较 2
11：触发于定时器 A 比较 4
12：触发于定时器 A 周期
13：触发于定时器 B 比较 2
14：触发于定时器 B 比较 4
15：触发于定时器 B 周期
16：触发于定时器 C 比较 2
17：触发于定时器 C 比较 4
18：触发于定时器 C 周期
19：触发于定时器 C 重置和计数器溢出
20：触发于定时器 D 比较 2
21：触发于定时器 D 比较 4
22：触发于定时器 D 周期
23：触发于定时器 D 重置和计数器溢出
24：触发于定时器 E 比较 2
25：触发于定时器 E 比较 3
26：触发于定时器 E 比较 4
27：触发于定时器 E 重置和计数器溢出
28：触发于定时器 F 比较 2
29：触发于定时器 F 比较 3
30：触发于定时器 F 比较 4
31：触发于定时器 F 周期

Bit 4:0 **ADC5TRG[4:0]**：ADC 触发 5 选择（ADC trigger 5 selection）

这些位用于选择 ADC 触发 5 的来源。具体的选项如下：

0：触发于主定时器比较 1
1：触发于主定时器比较 2
2：触发于主定时器比较 3
3：触发于主定时器比较 4
4：触发于主定时器周期
5：触发于外部事件 1
6：触发于外部事件 2
7：触发于外部事件 3
8：触发于外部事件 4
9：触发于外部事件 5
10：触发于定时器 A 比较 3
11：触发于定时器 A 比较 4
12：触发于定时器 A 周期
13：触发于定时器 A 重置和计数器溢出
14：触发于定时器 B 比较 3
15：触发于定时器 B 比较 4
16：触发于定时器 B 周期
17：触发于定时器 B 重置和计数器溢出
18：触发于定时器 C 比较 3
19：触发于定时器 C 比较 4
20：触发于定时器 C 周期
21：触发于定时器 D 比较 3
22：触发于定时器 D 比较 4
23：触发于定时器 D 周期
24：触发于定时器 E 比较 3
25：触发于定时器 E 比较 4
26：触发于定时器 E 周期
27：触发于定时器 F 比较 2
28：触发于定时器 F 比较 3
29：触发于定时器 F 比较 4
30：触发于定时器 F 周期
31：触发于定时器 F 重置和计数器溢出

### 21.5.77 HRTIM ADC 触发更新寄存器（HRTIM_ADCUR）

偏移地址：0x3FC

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | AD10USRC[2:0] | | | Res | AD9USRC[2:0] | | |
| | | | | | | | | | rw | rw | rw | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | AD8USRC[2:0] | | | Res | AD7USRC[2:0] | | | Res | AD6USRC[2:0] | | | Res | AD5USRC[2:0] | | |
| | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw |

Bit 31:23 保留，必须保持为复位值。

Bit 22:20 **AD10USRC[2:0]**：ADC 触发 10 更新源（ADC trigger 10 update source）

参考 AD5USRC[2:0]的描述。

Bit 19 保留，必须保持复位值。

Bit 18:16 **AD9USRC[2:0]**：ADC 触发 9 更新源（ADC trigger 9 update source）

参考 AD5USRC[2:0]的描述。

Bit 15 保留，必须保持复位值。

Bit 14:12 **AD8USRC[2:0]**：ADC 触发 8 更新源（ADC trigger 8 update source）

参考 AD5USRC[2:0]的描述。

Bit 11 保留，必须保持复位值。

Bit 10:8 **AD7USRC[2:0]**：ADC 触发 7 更新源（ADC trigger 7 update source）

参考 AD5USRC[2:0]的描述。

Bit 7 保留，必须保持复位值。

Bit 6:4 **AD6USRC[2:0]**：ADC 触发 6 更新源（ADC trigger 6 update source）

参考 AD5USRC[2:0]的描述。

Bit 3 保留，必须保持复位值。

Bit 2:0 **AD5USRC[2:0]**：ADC 触发 5 更新源（ADC trigger 5 update source）

这些位定义了触发 HRTIM_ADCER 寄存器中 ADC5TRG[4:0]位域更新的源（从预加载寄存器传输到有效寄存器）。它仅定义了源定时器。

确切的条件在定时器本身中定义，通过 HRTIM_MCR 寄存器中的 BRSTDMA[1:0]位域或 HRTIM_TIMxCR 寄存器中的 UPDGAT[3:0]位域。

000：主定时器

001：定时器 A

010：定时器 B

011：定时器 C

100：定时器 D

101：定时器 E

110：定时器 F

111：保留

### 21.5.78 HRTIM ADC 后缩放器寄存器 1 (HRTIM_ADCPS1)

偏移地址：0x400

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | ADC5PSC[4:0] | | | | | Res | ADC4PSC[4:0] | | | | | Res | ADC3 PSC[4] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC3PSC[3:0] | | | | Res | ADC2PSC[4:0] | | | | | Res | ADC1PSC[4:0] | | | | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持为复位值。

Bit 28:24 **ADC5PSC[4:0]**：ADC 5 后缩放器（ADC 5 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 23 保留，必须保持复位值。

Bit 22:18 **ADC4PSC[4:0]**：ADC 4 后缩放器（ADC 4 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 17 保留，必须保持复位值。

Bit 16:12 **ADC3PSC[4:0]**：ADC 3 后缩放器（ADC 3 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 11 保留，必须保持复位值。

Bit 10:6 **ADC2PSC[4:0]**：ADC 2 后缩放器（ADC 2 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 5 保留，必须保持复位值。

Bit 4:0 **ADC1PSC[4:0]**：ADC 1 后缩放器（ADC 1 post scaler）

这些位用于选择 ADC 1 的后缩放器比率。

### 21.5.79 HRTIM ADC 后缩放器寄存器 2 (HRTIM_ADCPS2)

偏移地址：0x404

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | ADC10PSC[4:0] | | | | | Res | ADC9PSC[4:0] | | | | | Res | ADC8 PSC[4] |
| | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC8PSC[3:0] | | | | Res | ADC7PSC[4:0] | | | | | Res | ADC6PSC[4:0] | | | | |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

Bit 31:29 保留，必须保持为复位值。

Bit 28:24 **ADC10PSC[4:0]**：ADC 10 后缩放器（ADC 10 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 23 保留，必须保持复位值。

Bit 22:18 **ADC9PSC[4:0]**：ADC 9 后缩放器（ADC 9 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 17 保留，必须保持复位值。

Bit 16:12 **ADC8PSC[4:0]**：ADC 8 后缩放器（ADC 8 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 11 保留，必须保持复位值。

Bit 10:6 **ADC7PSC[4:0]**：ADC 7 后缩放器（ADC 7 post scaler）

参考 ADC1PSC[4:0] 的描述。

Bit 5 保留，必须保持复位值。

Bit 4:0 **ADC6PSC[4:0]**：ADC 6 后缩放器（ADC 6 post scaler）

参考 ADC1PSC[4:0] 的描述。

### 21.5.80 HRTIM 故障输入寄存器 3 (HRTIM_FLTINR3)

偏移地址：0x408

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT4 RSTM | FLT4 CRES | FLT4CNT[3:0] | | | | FLT4 BLKS | FLT4 BLKE | FLT3 RSTM | FLT3 CRES | FLT3CNT[3:0] | | | | FLT3 BLKS | FLT3 BLKE |
| 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 |
| FLT2 RSTM | FLT2 CRES | FLT2CNT[3:0] | | | | FLT2 BLKS | FLT2 BLKE | FLT1 RSTM | FLT1 CRES | FLT1CNT[3:0] | | | | FLT1 BLKS | FLT1 BLKE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **FLT4RSTM**：故障 4 复位模式（Fault 4 reset mode）

参考 FLT5RSTM 的描述。

Bit 30 **FLT4CRES**：故障 4 计数器复位（Fault 4 counter reset）

参考 FLT5CRES 的描述。

Bit 29:26 **FLT4CNT[3:0]**：故障 4 计数器（Fault 4 counter）

参考 FLT5CNT 的描述。

Bit 25 **FLT4BLKS**：故障 4 消隐源（Fault 4 blanking source）

参考 FLT5BLKS 的描述。

Bit 24 **FLT4BLKE**：故障 4 消隐使能（Fault 4 blanking enable）

参考 FLT5BLKE 的描述。

Bit 23 **FLT3RSTM**：故障 3 复位模式（Fault 3 reset mode）

参考 FLT5RSTM 的描述。

Bit 22 **FLT3CRES**：故障 3 计数器复位（Fault 3 counter reset）

参考 FLT5CRES 的描述。

Bit 21:18 **FLT3CNT[3:0]**：故障 3 计数器（Fault 3 counter）

参考 FLT5CNT 的描述。

Bit 17 **FLT3BLKS**：故障 3 消隐源（Fault 3 blanking source）

参考 FLT5BLKS 的描述。

Bit 16 **FLT3BLKE**：故障 3 消隐使能（Fault 3 blanking enable）

参考 FLT5BLKE 的描述。

Bit 15 **FLT2RSTM**：故障 2 复位模式（Fault 2 reset mode）

参考 FLT5RSTM 的描述。

Bit 14 **FLT2CRES**：故障 2 计数器复位（Fault 2 counter reset）

参考 FLT5CRES 的描述。

Bit 13:10 **FLT2CNT[3:0]**：故障 2 计数器（Fault 2 counter）

参考 FLT5CNT 的描述。

Bit 9 **FLT2BLKS**：故障 2 消隐源（Fault 2 blanking source）

参考 FLT5BLKS 的描述。

Bit 8 **FLT2BLKE**：故障 2 消隐使能（Fault 2 blanking enable）

参考 FLT5BLKE 的描述。

Bit 7 **FLT1RSTM**：故障 1 复位模式（Fault 1 reset mode）

参考 FLT5RSTM 的描述。

Bit 6 **FLT1CRES**：故障 1 计数器复位（Fault 1 counter reset）

参考 FLT5CRES 的描述。

Bit 5:2 **FLT1CNT[3:0]**：故障 1 计数器（Fault 1 counter）

参考 FLT5CNT 的描述。

Bit 1 **FLT1BLKS**：故障 1 消隐源（Fault 1 blanking source）

参考 FLT5BLKS 的描述。

Bit 0 **FLT1BLKE**：故障 1 消隐使能（Fault 1 blanking enable）

参考 FLT5BLKE 的描述。

### 21.5.81 HRTIM 故障输入寄存器 4（HRTIM_FLTINR4）

偏移地址：0x40C

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLT6 RSTM | FLT6 CRES | FLT6CNT[3:0] | | | | FLT6 BLKS | FLT6 BLKE | FLT5 RSTM | FLT5 CRES | FLT5CNT[3:0] | | | | FLT5 BLKS | FLT5 BLKE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持为复位值。

Bit 15 **FLT6RSTM**：故障 6 复位模式（Fault 6 reset mode）

参考 FLT5RSTM 的描述。

Bit 14 **FLT6CRES**：故障 6 计数器复位（Fault 6 counter reset）

参考 FLT5CRES 的描述。

Bit 13:10 **FLT6CNT[3:0]**：故障 6 计数器（Fault 6 counter）

参考 FLT5CNT 的描述。

Bit 9 **FLT6BLKS**：故障 6 消隐源（Fault 6 blanking source）

参考 FLT5BLKS 的描述。

Bit 8 **FLT6BLKE**：故障 6 消隐使能（Fault 6 blanking enable）

参考 FLT5BLKE 的描述。

Bit 7 **FLT5RSTM**：故障 5 复位模式（Fault 5 reset mode）

此位选择 FAULT5 计数器的复位模式

0：FAULT5 计数器在每个复位/溢出事件时复位

1：如果上一个计数周期内没有发生故障，则 FAULT5 计数器在每个复位/溢出事件时复位。

*注：此位域仅在* FLT5E *使能位复位时写入。*

Bit 6 **FLT5CRES**：故障 5 计数器复位（Fault 5 counter reset）

此位复位 FAULT5 计数器。它由软件设置并由硬件复位。

0：无操作

1：FAULT5 计数器复位

Bit 5:2 **FLT5CNT[3:0]**：故障 5 计数器（Fault 5 counter）

此位域选择 FAULT5 计数器的阈值。当事件数量等于（FLT5CNT[3:0]+1）的值时， 故障被视为有效。

Bit 1 **FLT5BLKS**：故障 5 消隐源（Fault 5 blanking source）

FTL5BLKS 位选择 FAULT5 的消隐源。

0：故障 5 复位对齐的消隐窗口

1：故障 5 移动消隐窗口

*注：此位域仅在* FLT5E *使能位复位时写入。*

Bit 0 **FLT5BLKE**：故障 5 消隐使能（Fault 5 blanking enable）

FTL5BLKE 位选择 FAULT5 的消隐模式。消隐源由 FLT5BLKS 位定义。

0：故障 5 无消隐

1：故障 5 消隐模式

*注：此位域仅在* FLT5E *使能位复位时写入。*

### 21.5.82 HRTIM 选项寄存器（HRTIM_OPT）

偏移地址：0x420

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | DLL BYPASS | Reserved | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:3 保留，必须保持为复位值。

Bit 2 **DLLBYPASS**：DLL 旁路（DLL BYPASS）

此位使能 DLL 旁路。

0：禁止 DLL 旁路

1：使能 DLL 旁路

Bit 1:0 保留，必须保持为复位值。

# 22 高级控制定时器（TIM1/TIM8）

## 22.1 TIM1/TIM8 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，它由一个可编程的预分频器驱动。

它适合多种用途，包括包括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/TIM8）和通用定时器（TIMy）是完全独立的，它们不共享任何资源。它们可以同步操作。

## 22.2 TIM1/TIM8 主要特征

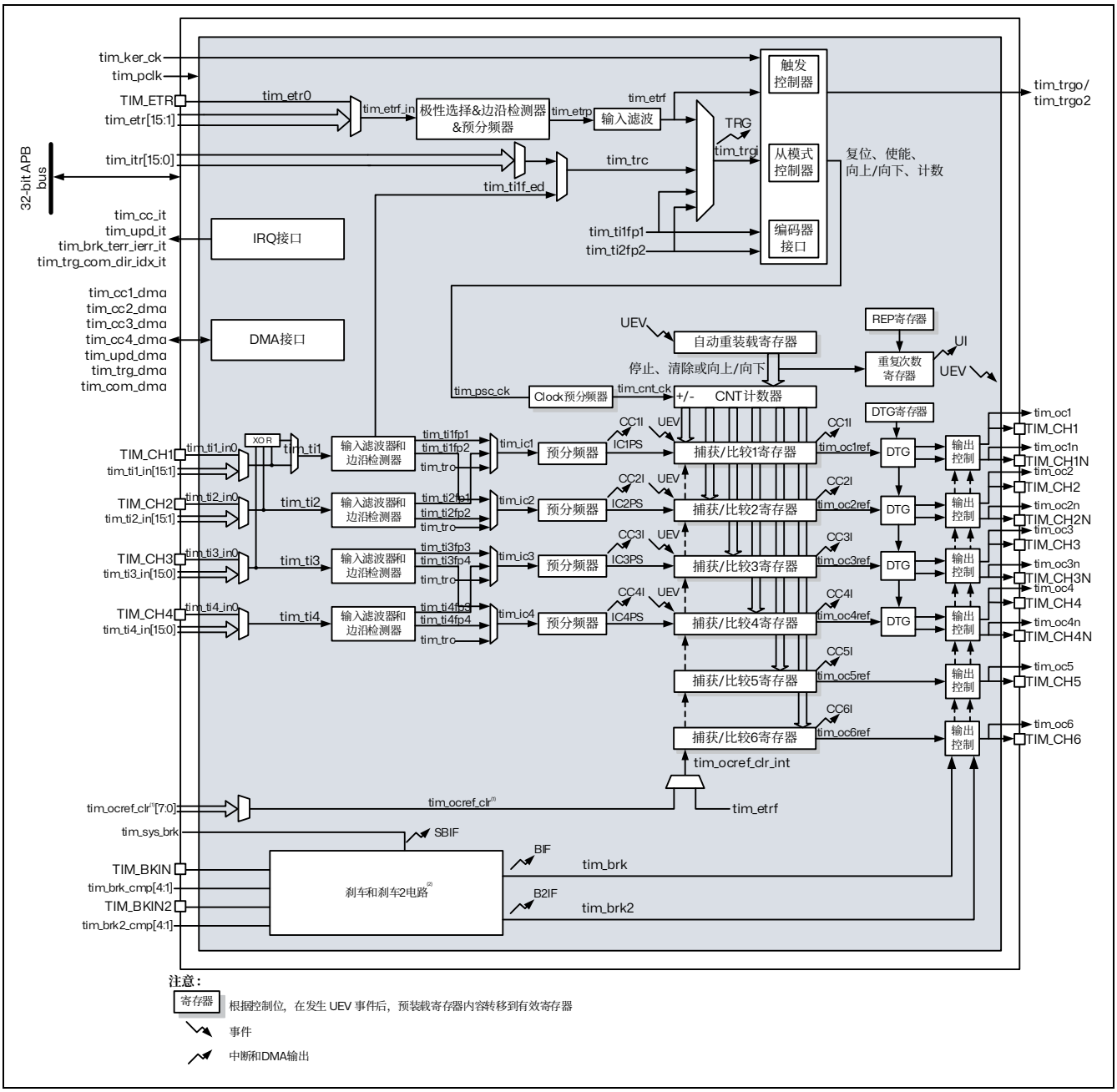
TIM1/TIM8 定时器的功能包括：

- 16 位向上、向下、向上/下自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 多达 6 个独立通道：
  - 输入捕获（除了通道 5 和 6）
  - 输出比较
  - PWM 生成（边缘或中间对齐模式）
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 用于将定时器输出信号置于安全的用户选择配置的两路刹车输入信号
- 如下事件发生时产生中断/DMA 请求：
  - 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
- 非对称模式

## 22.3 TIM1/TIM8 功能描述

### 22.3.1 框图

图 22.1 高级控制定时器框图




1. 这一特性详见下一节：TIM1/TIM8 引脚和内部信号
2. 参见刹车和刹车 2 电路以获得更多细节

### 22.3.2 TIM1/TIM8 引脚和内部信号

本章节中的表格总结了 TIM 的输入和输出。

表 22.1 TIM 输入/输出引脚

| 引脚名称 | 信号类型 | 描述 |
|---|---|---|
| TIM_CH1<br>TIM_CH2<br>TIM_CH3<br>TIM_CH4 | 输入/输出 | 定时器多用途通道。<br>每个通道都可以用于捕获、比较或 PWM（脉冲宽度调制）。<br>TIM_CH1 和 TIM_CH2 还可以作为外部时钟（时钟频率低于 tim_ker_ck 的 1/4）、外部触发器和四象限编码器输入使用。<br>TIM_CH1、TIM_CH2 和 TIM_CH3 可以用于与数字霍尔效应传感器接口。 |
| TIM_CH1N<br>TIM_CH2N<br>TIM_CH3N<br>TIM_CH4N | 输出 | 由 TIM_CHx 输出派生的定时器的互补输出，<br>具有插入死区时间的能力。 |
| TIM_ETR | 输入 | 外部触发器输入。这个输入可被用作外部触发器或外部时钟源。如果使用 tim_etr_in 预分频器，则该输入可以接收频率高于 tim_ker_ck 的时钟。 |
| TIM_BKIN<br>TIM_BKIN2 | 输入/输出 | 刹车和刹车 2 输入。这些输入也可以配置为双向模式。 |

表 22.2 TIM 内部输入/输出信号

| 内部信号名称 | 信号类型 | 描述 |
|---|---|---|
| tim_ti1_in[15:0]<br>tim_ti2_in[15:0]<br>tim_ti3_in[15:0]<br>tim_ti4_in[15:0] | 输入 | 内部定时器输入总线。<br>tim_ti1_in[15:0]和 tim_ti2_in[15:0]输入可用于捕获或作为外部时钟（时钟频率低于 tim_ker_ck 的 1/4），以及用于四象限编码器信号。 |
| tim_etr[15:0] | 输入 | 外部触发器内部输入总线。这些输入可被用作触发器、外部时钟或用于硬件的逐周期脉冲宽度控制。如果使用 tim_etr_in 预分频器，这些输入可以接收频率高于 tim_ker_ck 的时钟。 |
| tim_itr[15:0] | 输入 | 内部触发输入总线。这些输入可用于从模式控制器或作为输入时钟(tim_ker_ck 时钟的 1/4 以下)。 |
| tim_trgo/tim_trgo2 | 输出 | 内部触发器输出。这些触发器被其他定时器或其他外设使用。 |
| tim_ocref_clr[7:0] | 输入 | 定时器 tim_ocref_clr 输入总线。这些输入可以用来清除 tim_ocxref 信号，通常用于硬件的逐周期脉冲宽度控制。 |
| tim_brk_cmp[4:1] | 输入 | 内部信号的刹车输入。 |
| tim_brk2_cmp[4:1] | 输入 | 内部信号的刹车 2 输入。 |
| tim_sys_brk[n:0] | 输入 | 系统刹车输入。该输入收集 MCU 的系统级错误。 |
| tim_pclk | 输入 | 定时器 APB 时钟。 |
| tim_ker_ck | 输入 | 定时器内核时钟。 |

| 内部信号名称 | 信号类型 | 描述 |
|---|---|---|
| tim_cc_it | 输出 | 定时器捕获/比较中断。 |
| tim_upd_it | 输出 | 定时器更新事件中断。 |
| tim_brk_terr_ierr_it | 输出 | 定时器刹车、刹车 2、转换错误和索引错误中断。 |
| tim_trg_com_dir_idx_it | 输出 | 定时器触发、换向、方向和索引中断。 |
| tim_cc1_dma<br>tim_cc2_dma<br>tim_cc3_dma<br>tim_cc4_dma | 输出 | 定时器捕获/比较 1 至 4 dma 请求。 |
| tim_upd_dma | 输出 | 定时器更新 dma 请求。 |
| tim_trg_dma | 输出 | 定时器触发 dma 请求。 |
| tim_com_dma | 输出 | 定时器换向 dma 请求。 |

下表列出连接到 tim_ti[4:1]输入多路复用器的源。

表 22.3 连接到 tim_ti1 的输入多路复用器

| tim_ti1 输入 | 源 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_ti1_in0 | TIM1_CH1 | TIM8_CH1 |
| tim_ti1_in1 | comp1_out | comp1_out |
| tim_ti1_in2 | comp2_out | comp2_out |
| tim_ti1_in3 | comp3_out | comp3_out |
| tim_ti1_in4 | comp4_out | comp4_out |
| tim_ti1_in[15:5] | 保留 | |

表 22.4 连接到 tim_ti2 的输入多路复用器

| tim_ti2 输入 | 源 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_ti2_in0 | TIM1_CH2 | TIM8_CH2 |
| tim_ti2_in[15:1] | 保留 | |

表 22.5 连接到 tim_ti3 的输入多路复用器

| tim_ti3 输入 | 源 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_ti3_in0 | TIM1_CH3 | TIM8_CH3 |
| tim_ti3_in[15:1] | 保留 | |

表 22.6 连接到 tim_ti4 的输入多路复用器

| tim_ti4 输入 | 源 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_ti4_in0 | TIM1_CH4 | TIM8_CH4 |
| tim_ti4_in[15:1] | 保留 | |

下表列出连接到 tim_itr 输入多路复用器的内部源。

表 22.7 TIMx 内部触发连接

| tim_itrx 输入 | TIM1 | TIM8 |
|---|---|---|
| tim_itr0 | 保留 | tim1_trgo |
| tim_itr1 | tim2_trgo | tim2_trgo |
| tim_itr2 | tim3_trgo | tim3_trgo |
| tim_itr3 | tim4_trgo | tim4_trgo |
| tim_itr4 | tim5_trgo | tim5_trgo |
| tim_itr5 | tim8_trgo | 保留 |
| tim_itr6 | tim15_trgo | tim15_trgo |
| tim_itr7 | tim16_oc1 | tim16_oc1 |
| tim_itr8 | tim17_oc1 | tim17_oc1 |
| tim_itr9 | 保留 | |
| tim_itr10 | hrtim_out_sync2 | hrtim_out_sync2 |
| tim_itr[15:11] | 保留 | |

下表列出连接到 tim_etr 输入多路复用器的内部源。

表 22.8 连接到 tim_etr 的输入多路复用器

| 定时器外部触发输入信号 | 定时器外部触发信号分配 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_etr0 | TIM1_ETR | TIM8_ETR |
| tim_etr1 | comp1_out | comp1_out |
| tim_etr2 | comp2_out | comp2_out |
| tim_etr3 | comp3_out | comp3_out |
| tim_etr4 | comp4_out | comp4_out |
| tim_etr5 | 保留 | |
| tim_etr6 | | |
| tim_etr7 | | |
| tim_etr8 | adc1_awd | adc2_awd |
| tim_etr9 | 保留 | 保留 |
| tim_etr10 | | |
| tim_etr11 | | adc3_awd |
| tim_etr12 | | 保留 |
| tim_etr13 | | |
| tim_etr[15:14] | 保留 | |

下三张表列出了连接到 tim_brk 和 tim_brk2 输入的源。

表 22.9 定时器刹车互连

| tim_brk 输入 | TIM1 | TIM8 |
|---|---|---|
| TIM_BKIN | TIM1_BKIN pin | TIM8_BKIN pin |
| tim_brk_cmp1 | comp1_out | comp1_out |
| tim_brk_cmp2 | comp2_out | comp2_out |
| tim_brk_cmp3 | comp3_out | comp3_out |
| tim_brk_cmp4 | comp4_out | comp4_out |

表 22.10 定时器刹车 2 互连

| tim_brk2 输入 | TIM1 | TIM8 |
|---|---|---|
| TIM_BKIN2 | TIM1_BKIN2 pin | TIM8_BKIN2 pin |
| tim_brk2_cmp1 | comp1_out | comp1_out |
| tim_brk2_cmp2 | comp2_out | comp2_out |
| tim_brk2_cmp3 | comp3_out | comp3_out |
| tim_brk2_cmp4 | comp4_out | comp4_out |

表 22.11 系统中断互连

| tim_sys_brk 输入 | TIM1 / TIM8 | SYSCFG_CFGR2 寄存器的使能位 |
|---|---|---|
| tim_sys_brk0 | 内核锁住 | CLL |
| tim_sys_brk1 | 可编程电压检测器（PVD） | PVDL |
| tim_sys_brk2 | SRAM 奇偶校验错误 | SPL |
| tim_sys_brk3 | Flash ECC 错误 | ECCL |
| tim_sys_brk4 | 时钟安全系统（CSS） | 无（始终使能） |

下表列出连接到 tim_ocref_clr 输入多路复用器的内部源。

表 22.12 连接到 ocref_clr 的输入多路复用器

| 定时器 OCREF 清除信号 | 定时器 tim_ocref_clr 信号分配 | |
|---|---|---|
| | TIM1 | TIM8 |
| tim_ocref_clr0 | comp1_out | comp1_out |
| tim_ocref_clr1 | comp2_out | comp2_out |
| tim_ocref_clr2 | comp3_out | comp3_out |
| tim_ocref_clr3 | comp4_out | comp4_out |
| tim_ocref_clr4 | 保留 | |
| tim_ocref_clr5 | | |
| tim_ocref_clr6 | | |
| tim_ocref_clr7 | | |

### 22.3.3 时基单元

可编程高级控制定时器的主要模块是一个 16 位计数器及其相关的自动重载寄存器。计数器可递增计数、递减计数或同时进行递增和递减计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

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）

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件（UEV）时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位（ARPE）。当计数器达到上溢值（或者在递减计数时达到下溢值）并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生 进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位（CEN）置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

*注：计数器将在* TIMx_CR1 *寄存器的* CEN *位置* 1 *时刻的一个时钟周期后开始计数。*

#### 预分频器描述

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此可对预分频器进行实时更改。而新的预分频比将在下一更新事件发生时被采用。

下两张图以一些示例说明在预分频比实时变化时计数器的行为：

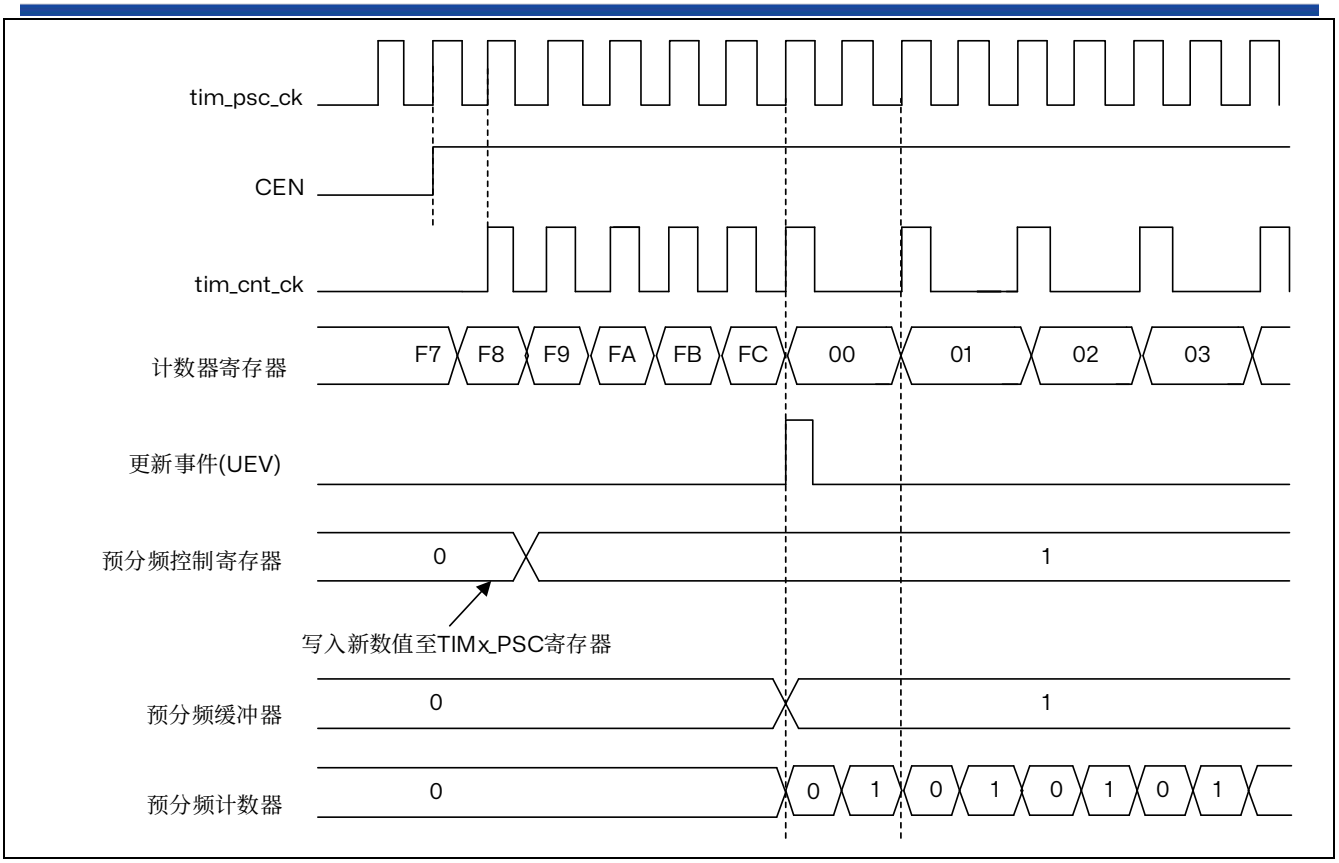



图 22.2 当预分频器的参数从 1 变成 2 时，计数器的时序图

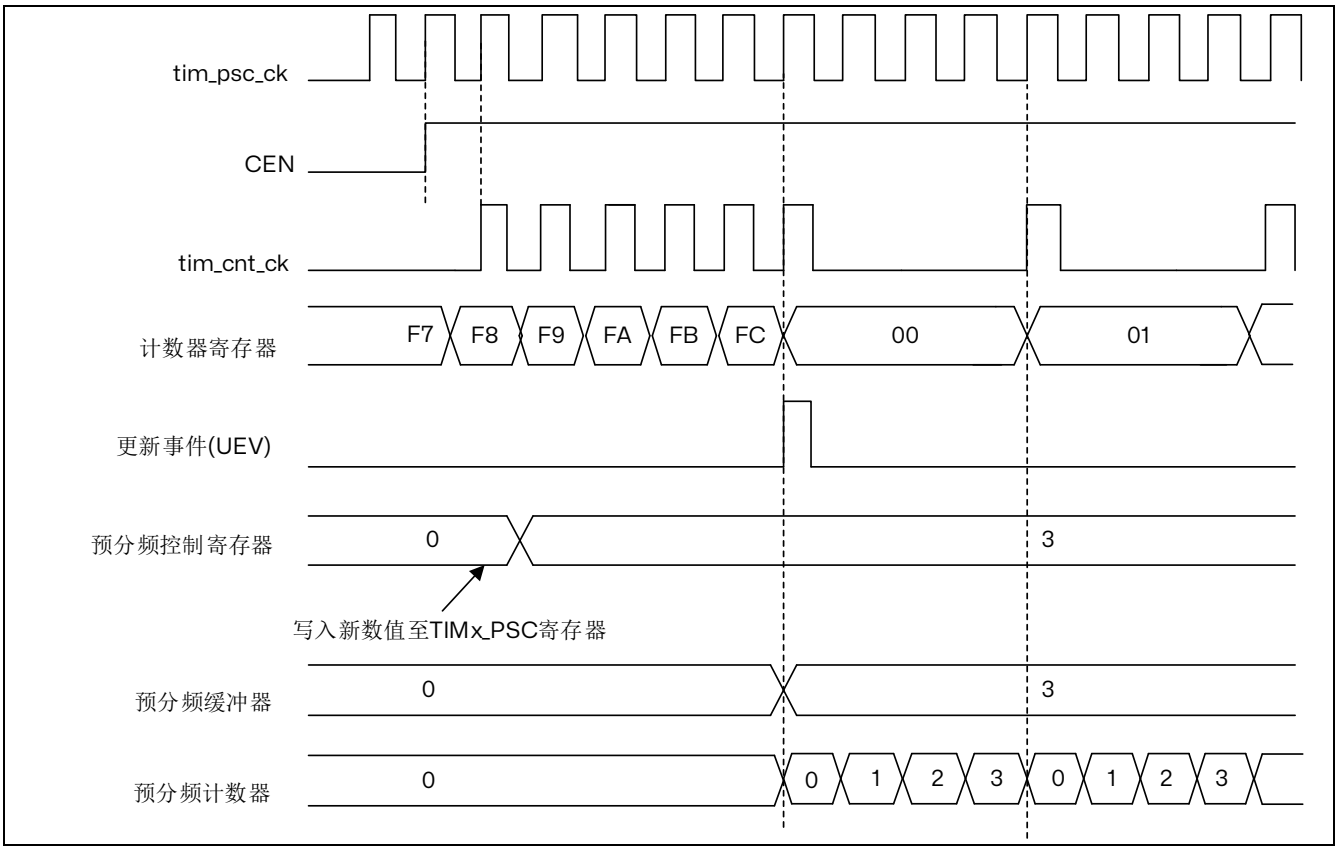



图 22.3 当预分频器的参数从 1 变成 4 时，计数器的时序图

### 22.3.4 计数模式

**递增计数模式**

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

如果使用重复计数器，则当递增计数的重复次数达到重复计数器寄存器中编程的次数加一次（TIMx_RCR+1）后，将生成更新事件(UEV)。否则，将在每次计数器上溢时产生更新事件。

将 TIMx_EGR 寄存器的 UG 位置 1（通过软件或使用从模式控制器）时，也将产生更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 自动重载影子寄存器将以预装载值(TIMx_ARR)进行更新
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

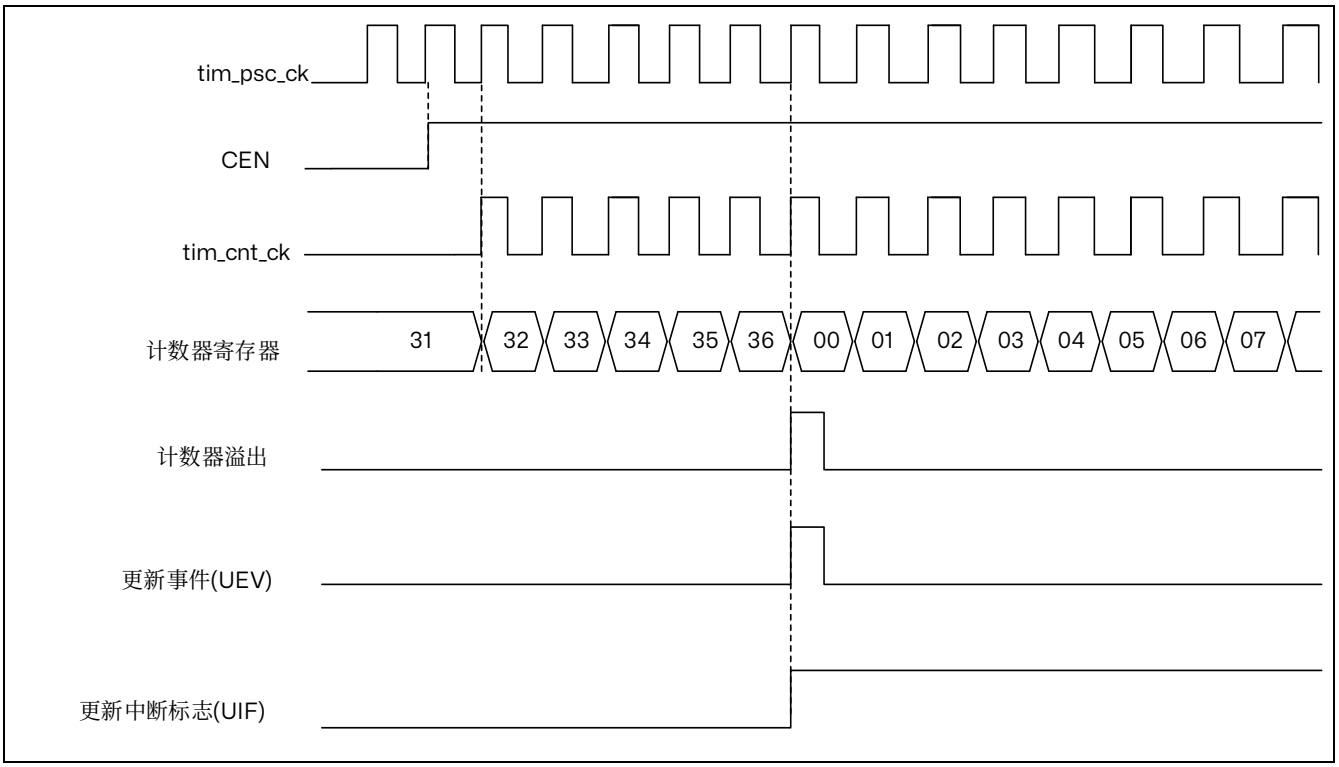



图 22.4 计数器时序图，内部时钟分频系数为 1

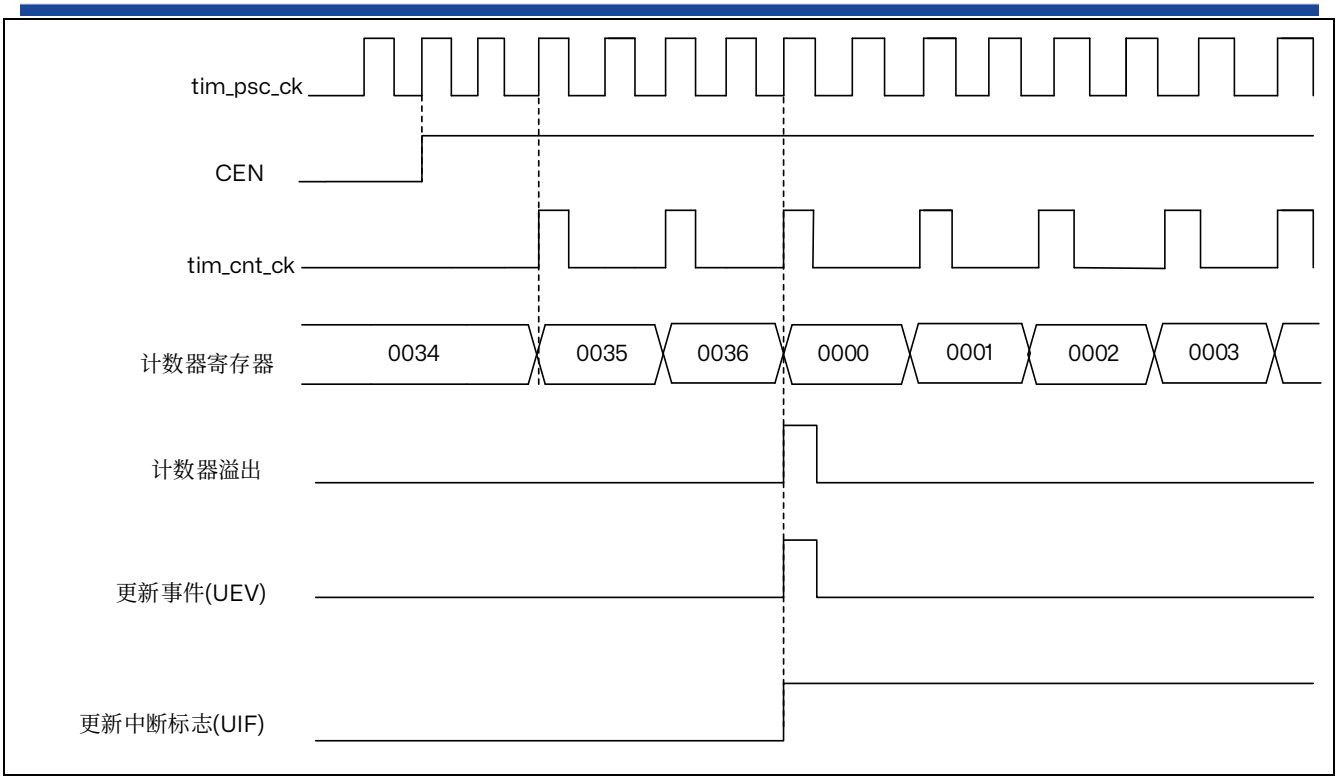



图 22.5 计数器时序图，内部时钟分频系数为 2

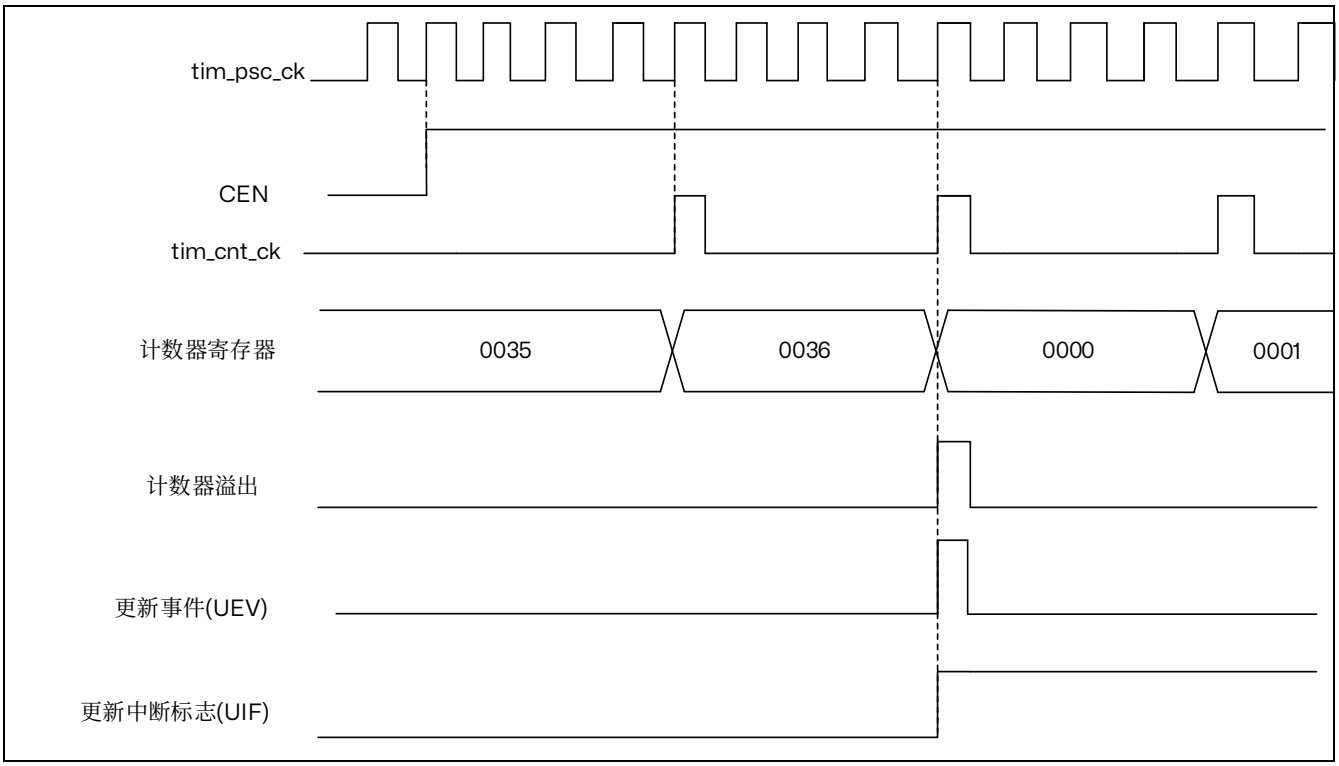



图 22.6 计数器时序图，内部时钟分频系数为 4

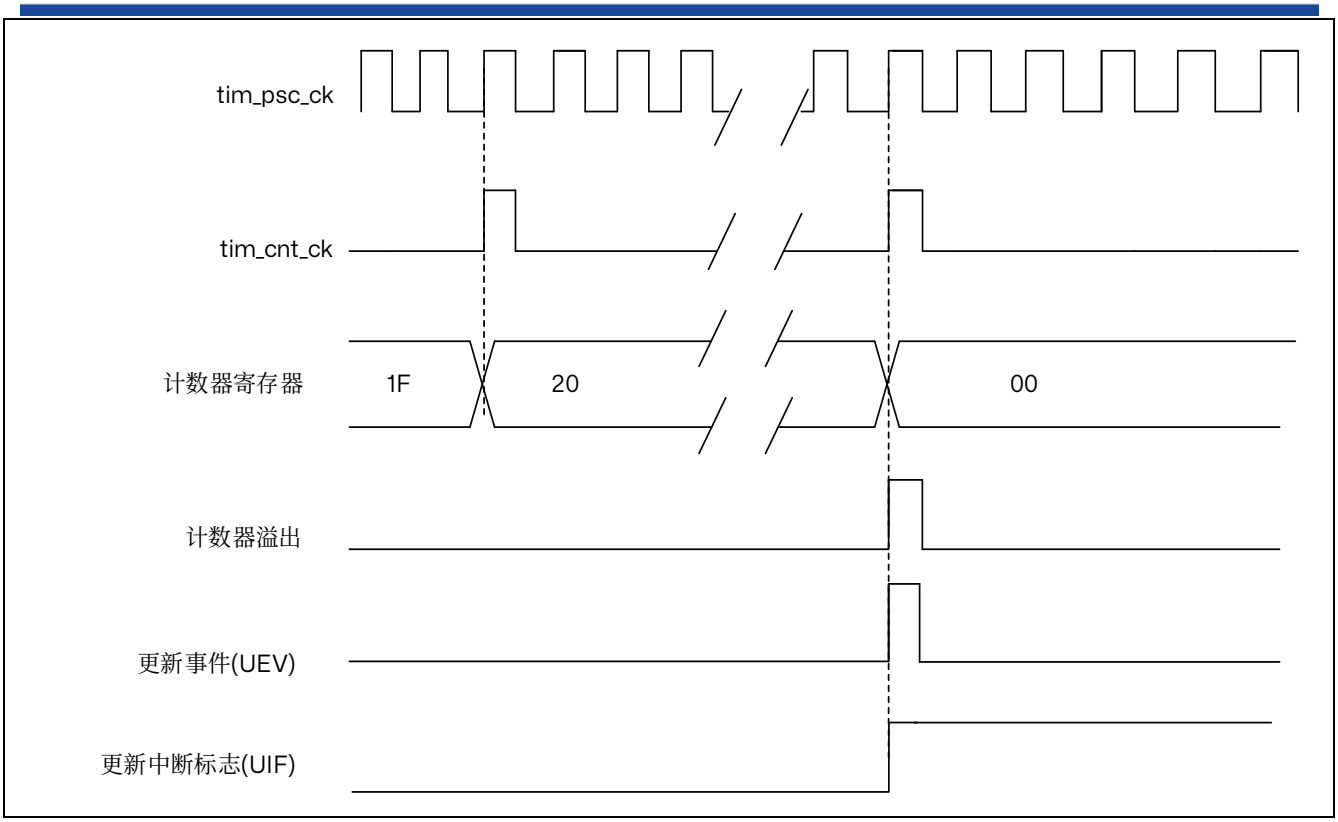



图 22.7 计数器时序图，内部时钟分频系数为 N

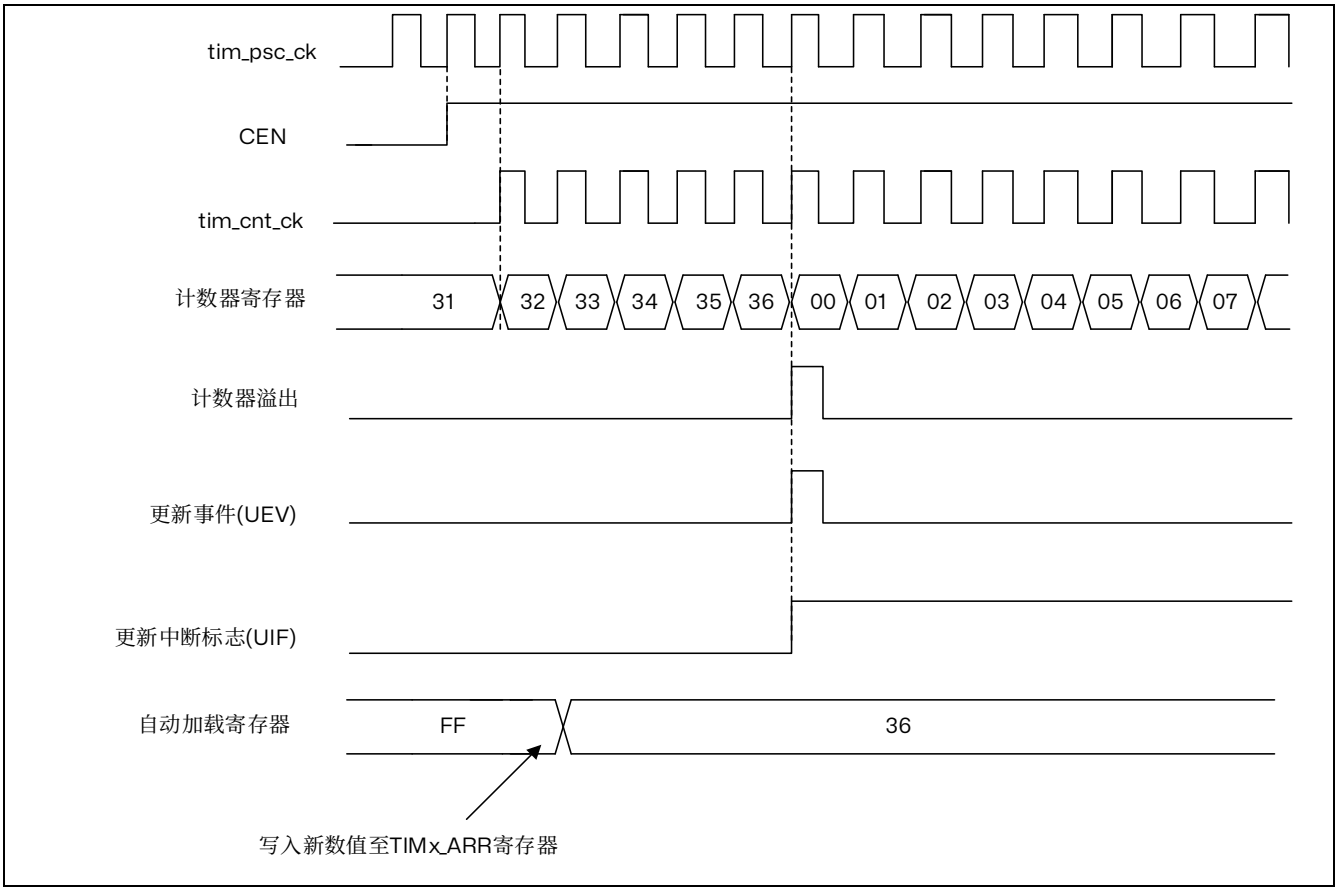



图 22.8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

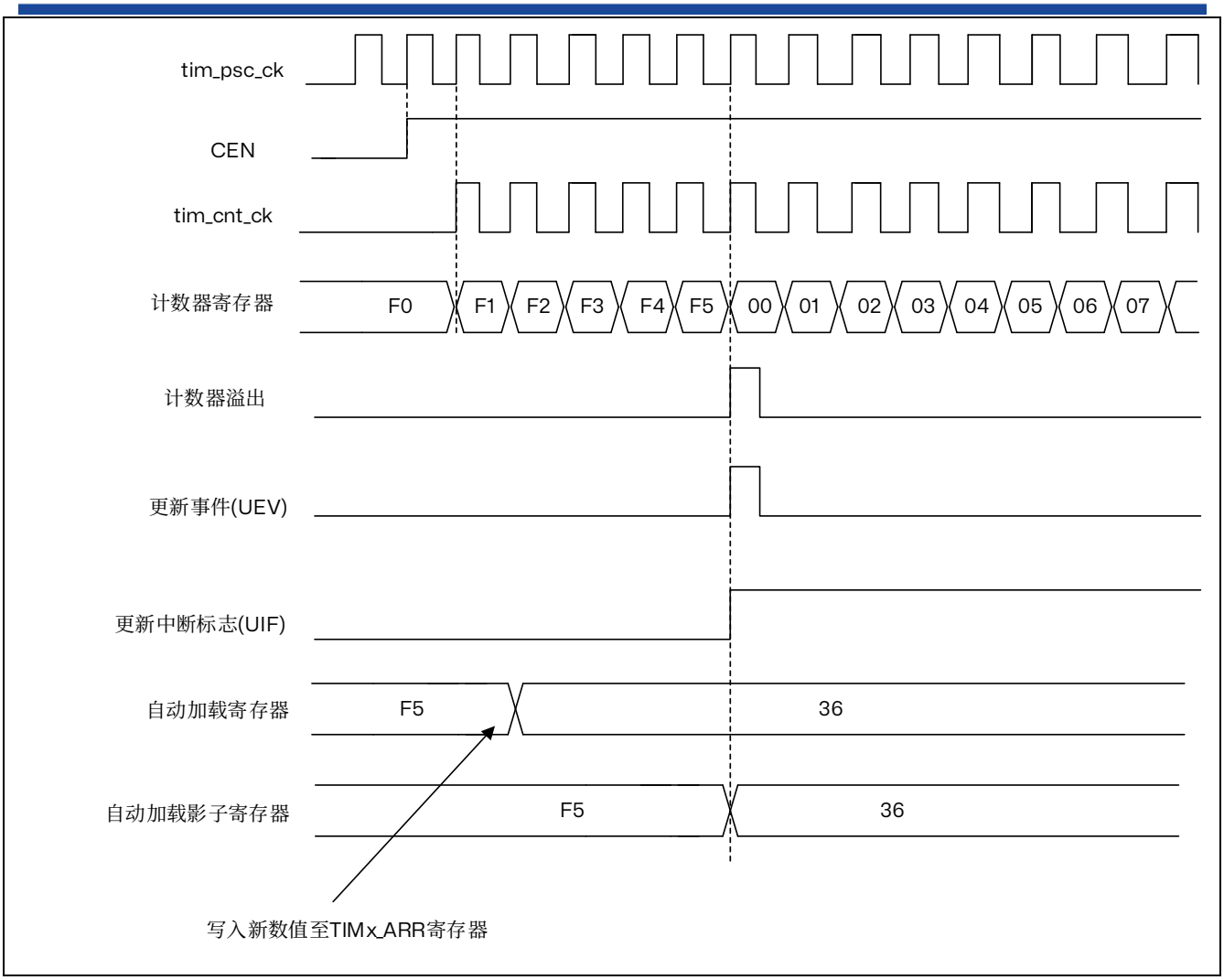



图 22.9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

**递减计数模式**

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

如果使用重复计数器，则当递减计数的重复次数达到重复计数器寄存器中编程的次数加一次（TIMx_RCR+1）后，将生成更新事件（UEV）。否则，将在每次计数器下溢时产生更新事件。

将 TIMx_EGR 寄存器的 UG 位置 1（通过软件或使用从模式控制器）时，也将产生更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断请求或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，自动重载寄存器会在计数器重载之前得到更新，因此，下一个计数周期就是我们所希望的新的周期长度

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

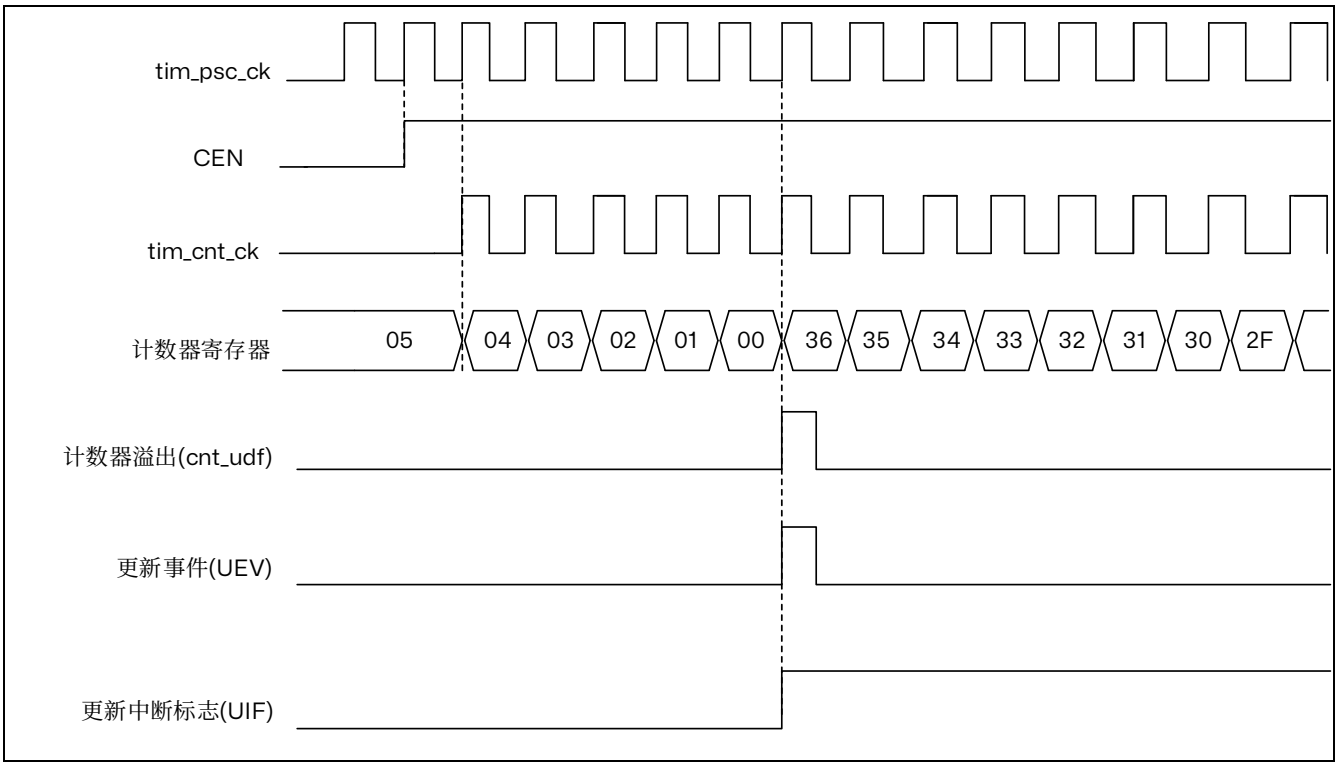



图 22.10 计数器时序图，内部时钟分频系数为 1

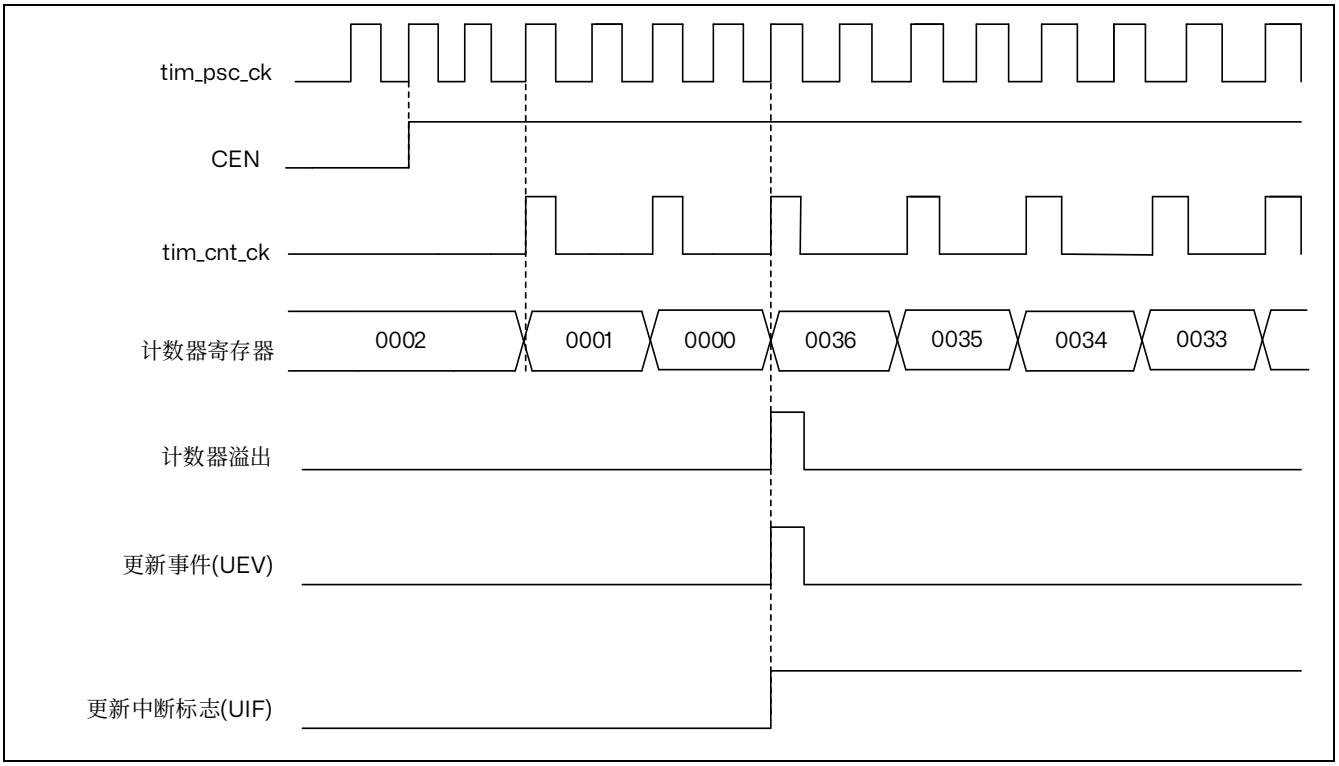


图 22.11 计数器时序图，内部时钟分频系数为 2

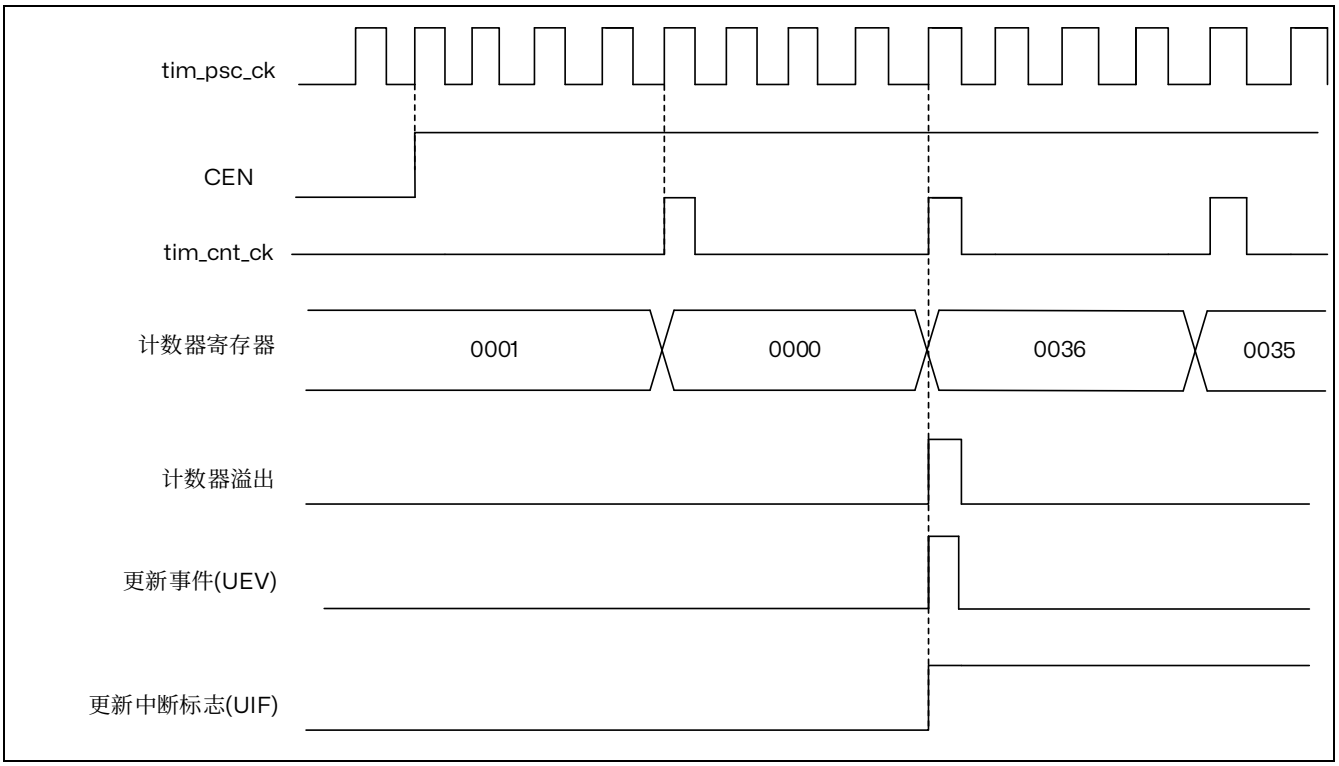



图 22.12 计数器时序图，内部时钟分频系数为 4

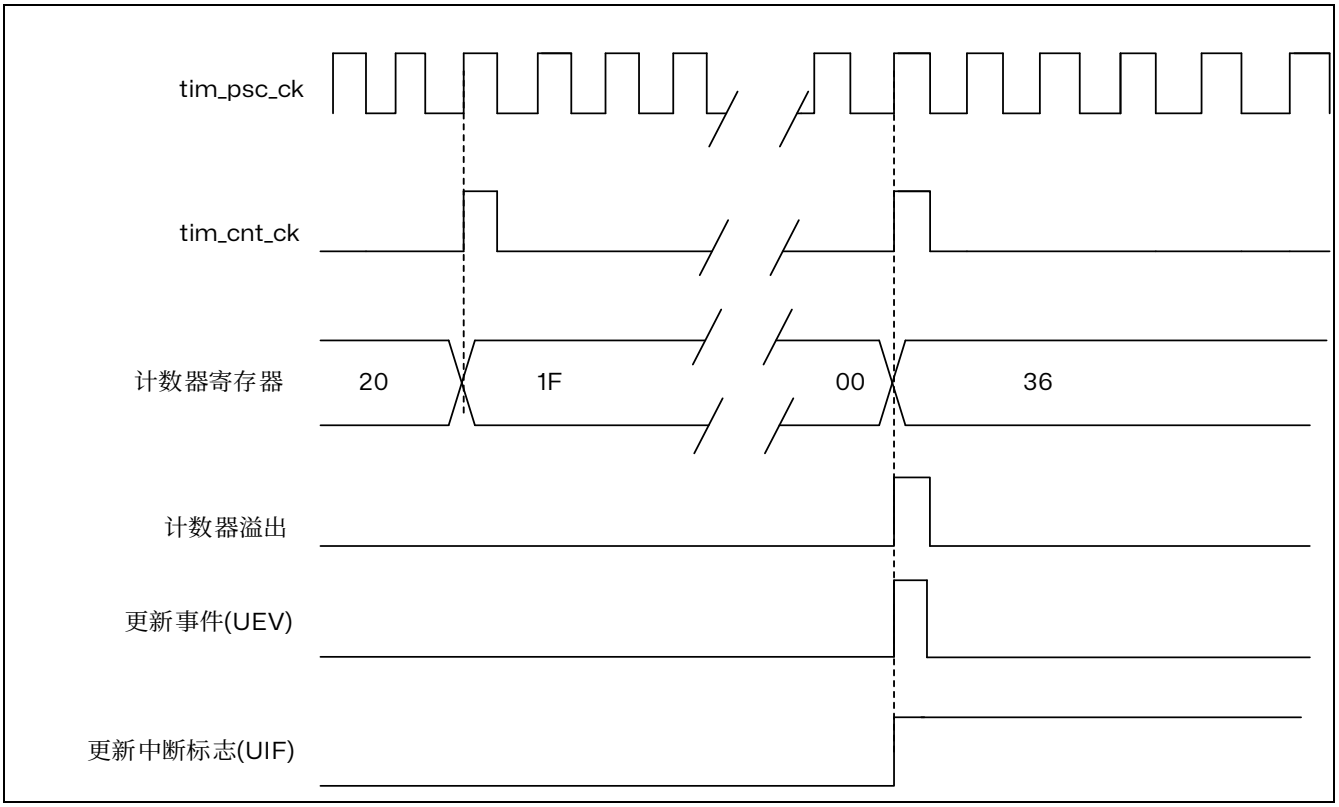



图 22.13 计数器时序图，内部时钟分频系数为 N

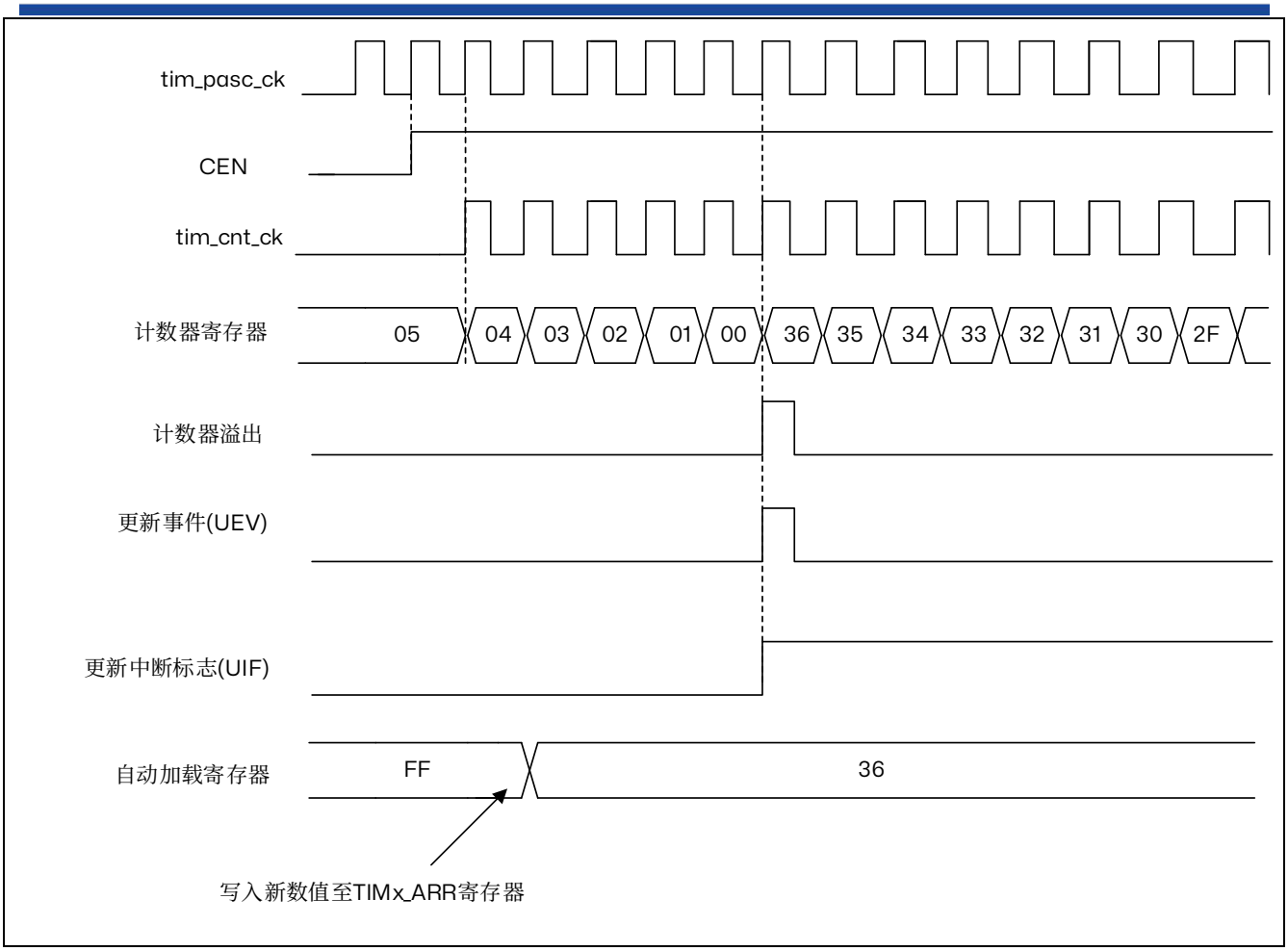



图 22.14 计数器时序图，当没有使用重复计数器时的更新事件

**中心对齐模式（递增/递减计数）**

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS = “01”）、计数器递增计数（中心对齐模式 2，CMS = “10”）以及计数器递增/递减计数（中心对齐模式 3，CMS = “11”）。

在此模式下，TIMx_CR1 寄存器的 DIR 方向位不可写入值，而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成 UEV 更新事件，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则自动重载寄存器在重载计数器之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。

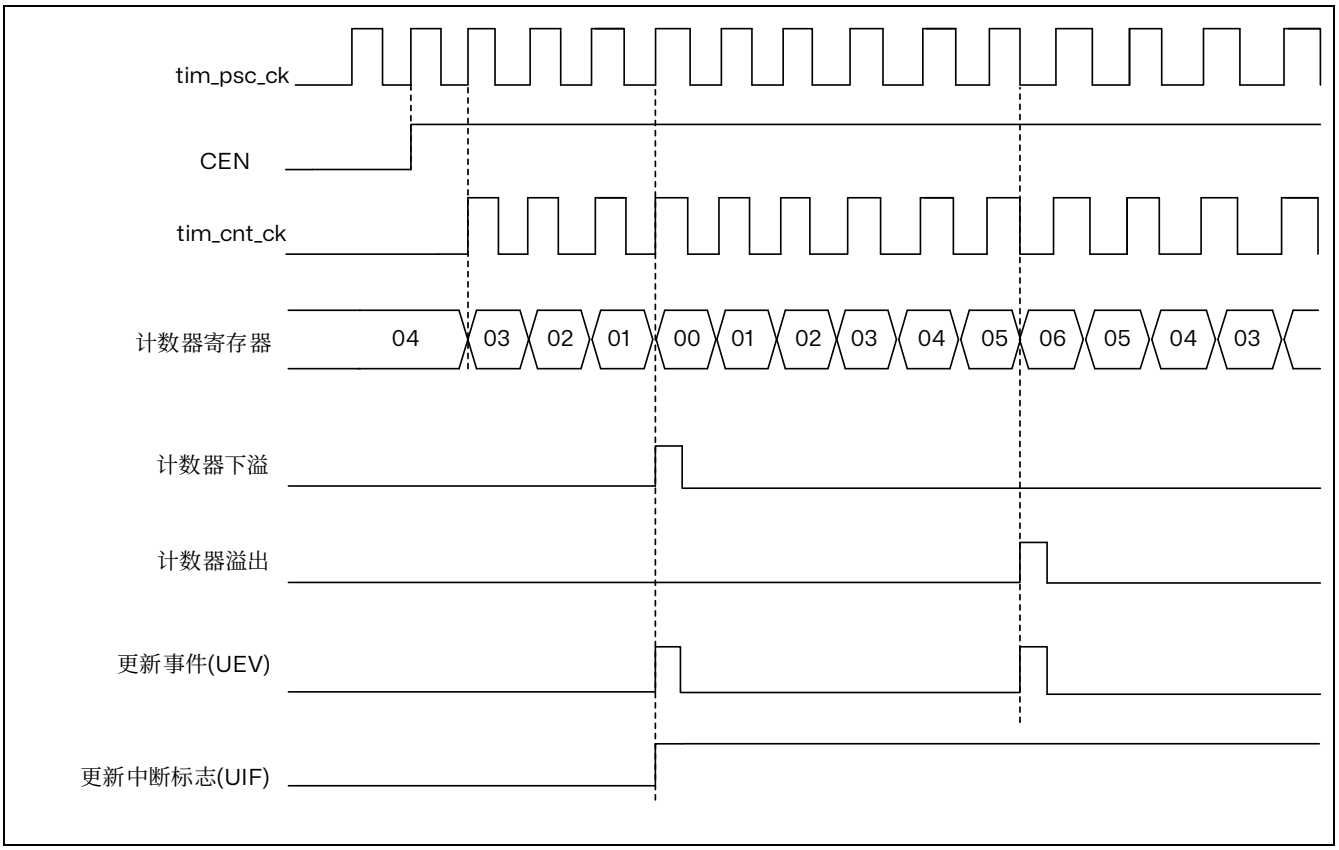



图 22.15 计数器时序图，内部时钟分频系数为 1，TIMx_ARR=0x6

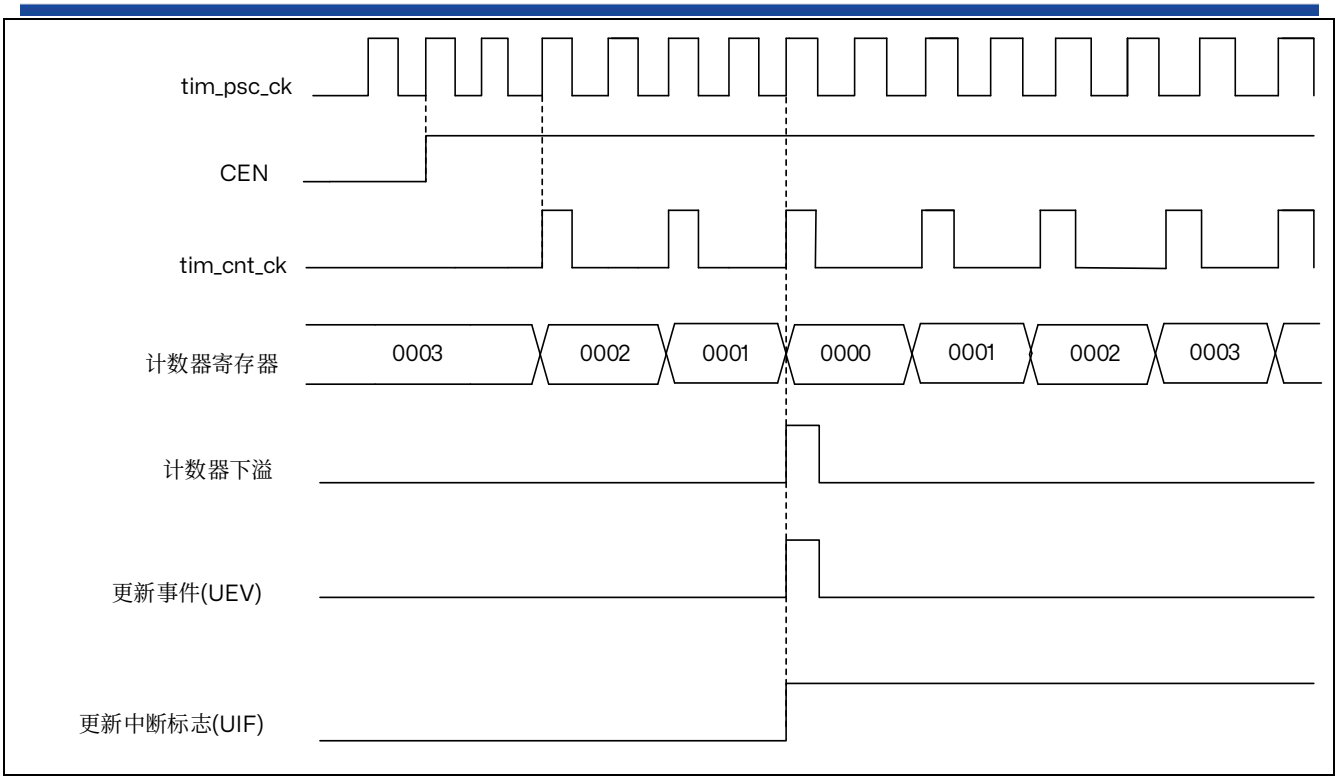



图 22.16 计数器时序图，内部时钟分频系数为 2

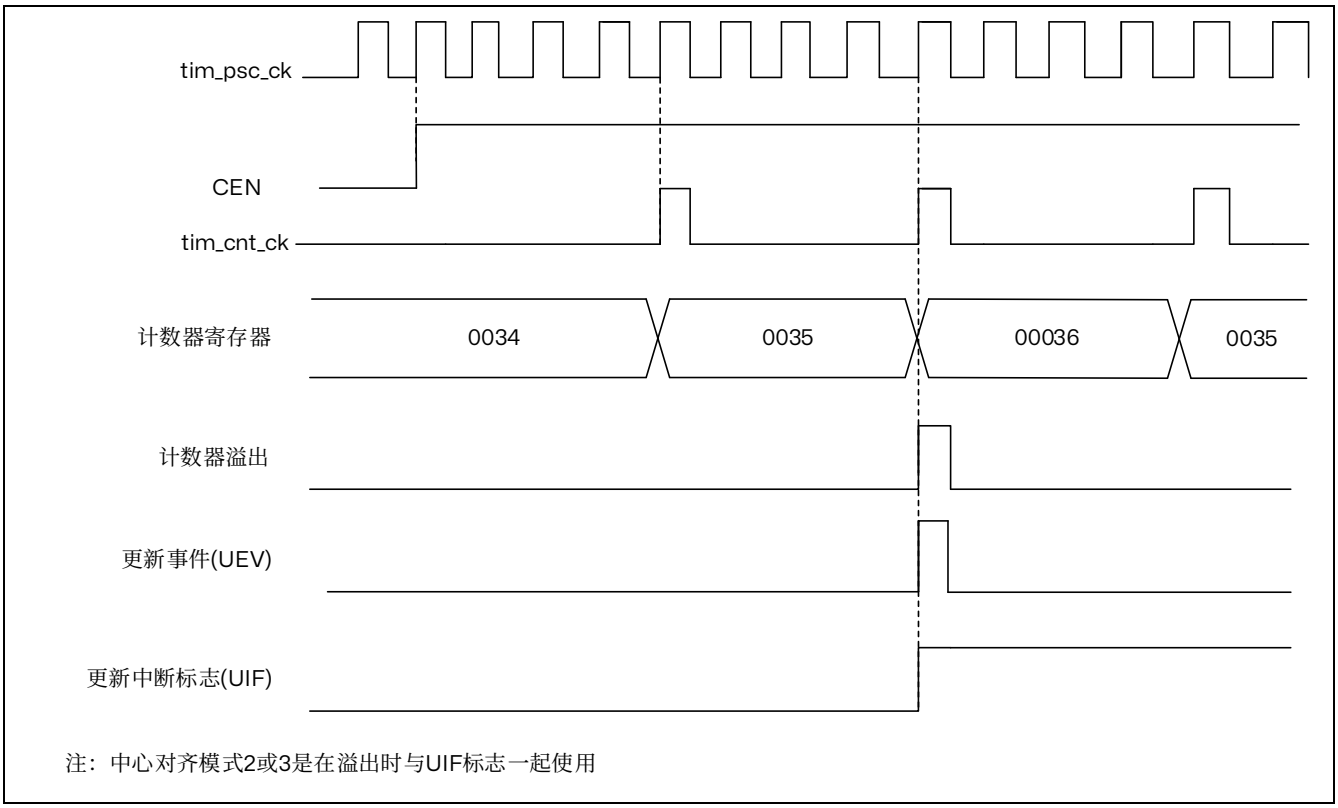



图 22.17 计数器时序图，内部时钟分频系数为 4，TIMx_ARR=0x36

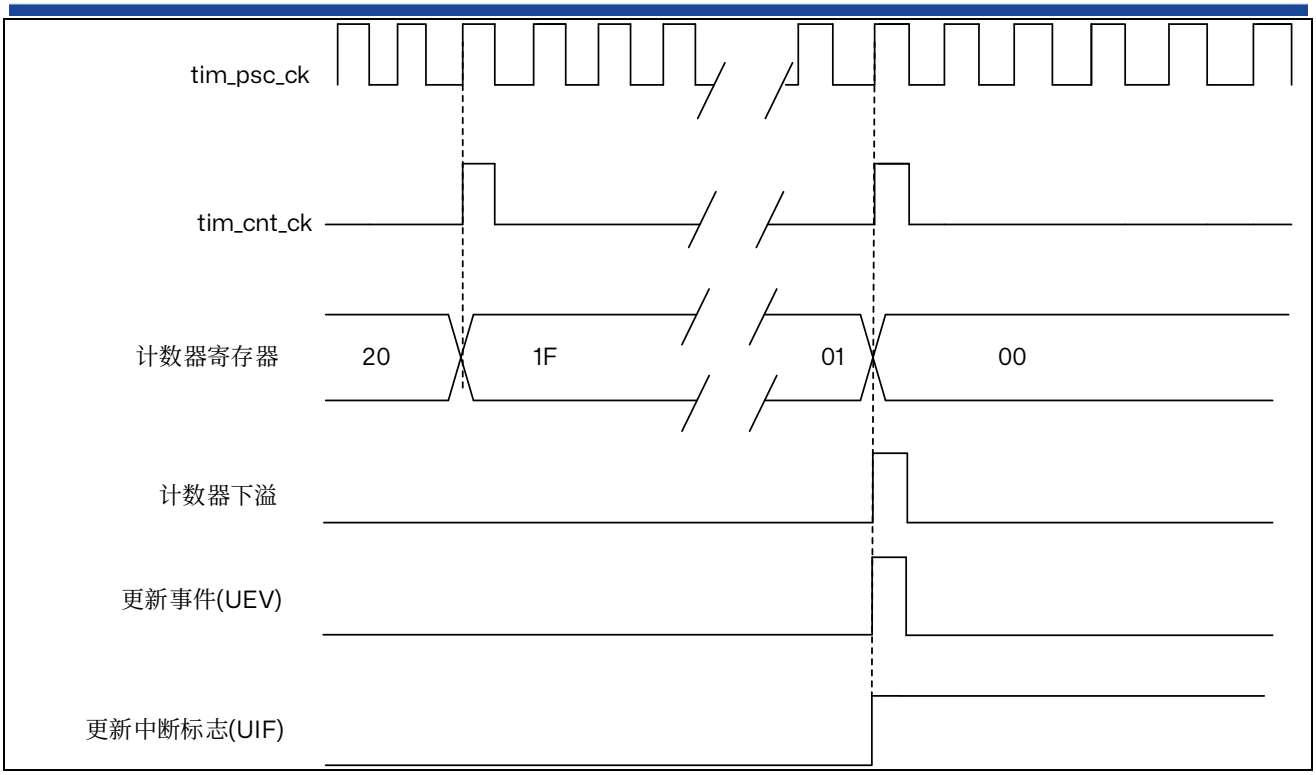



图 22.18 计数器时序图，内部时钟分频系数为 N

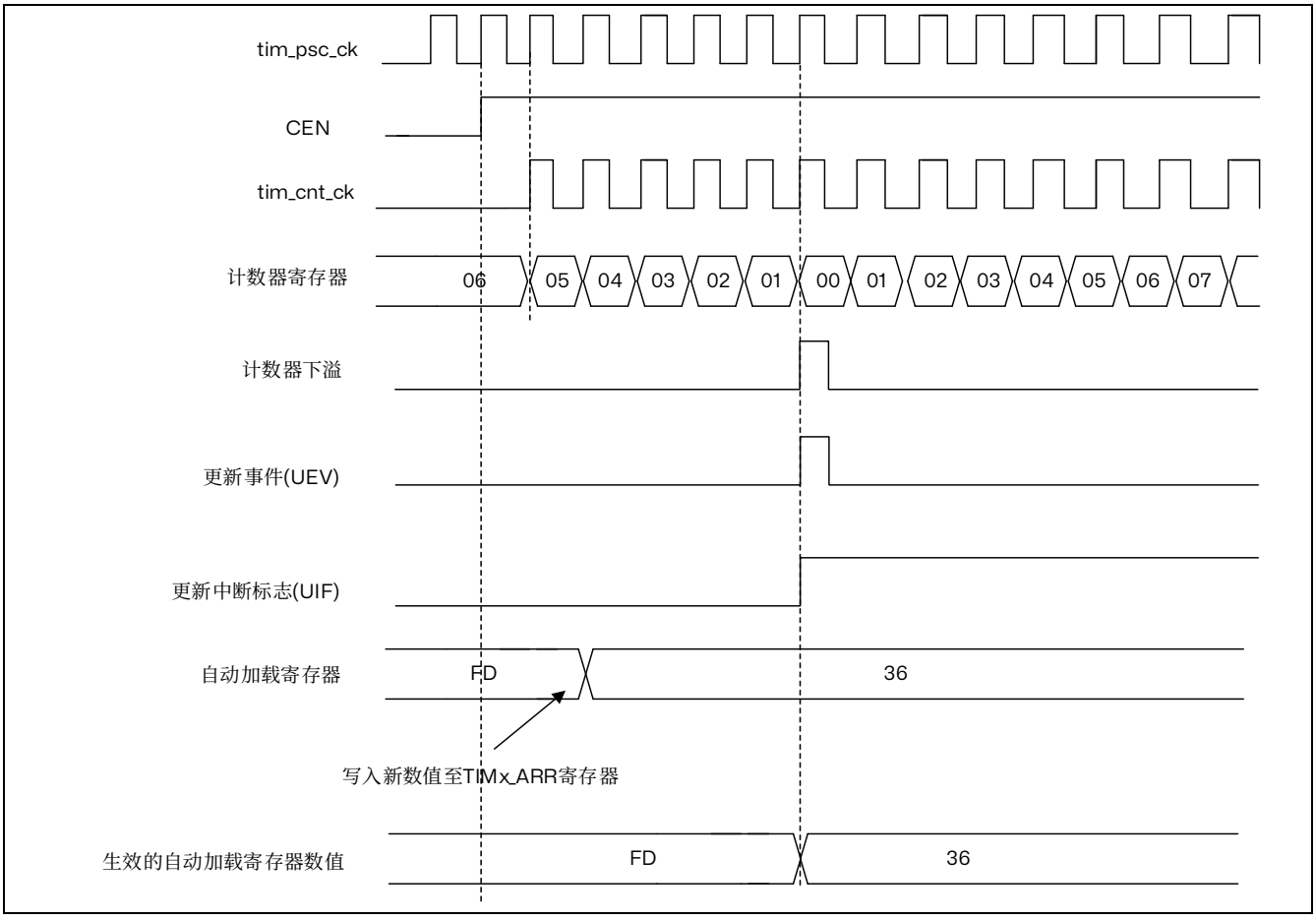



图 22.19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

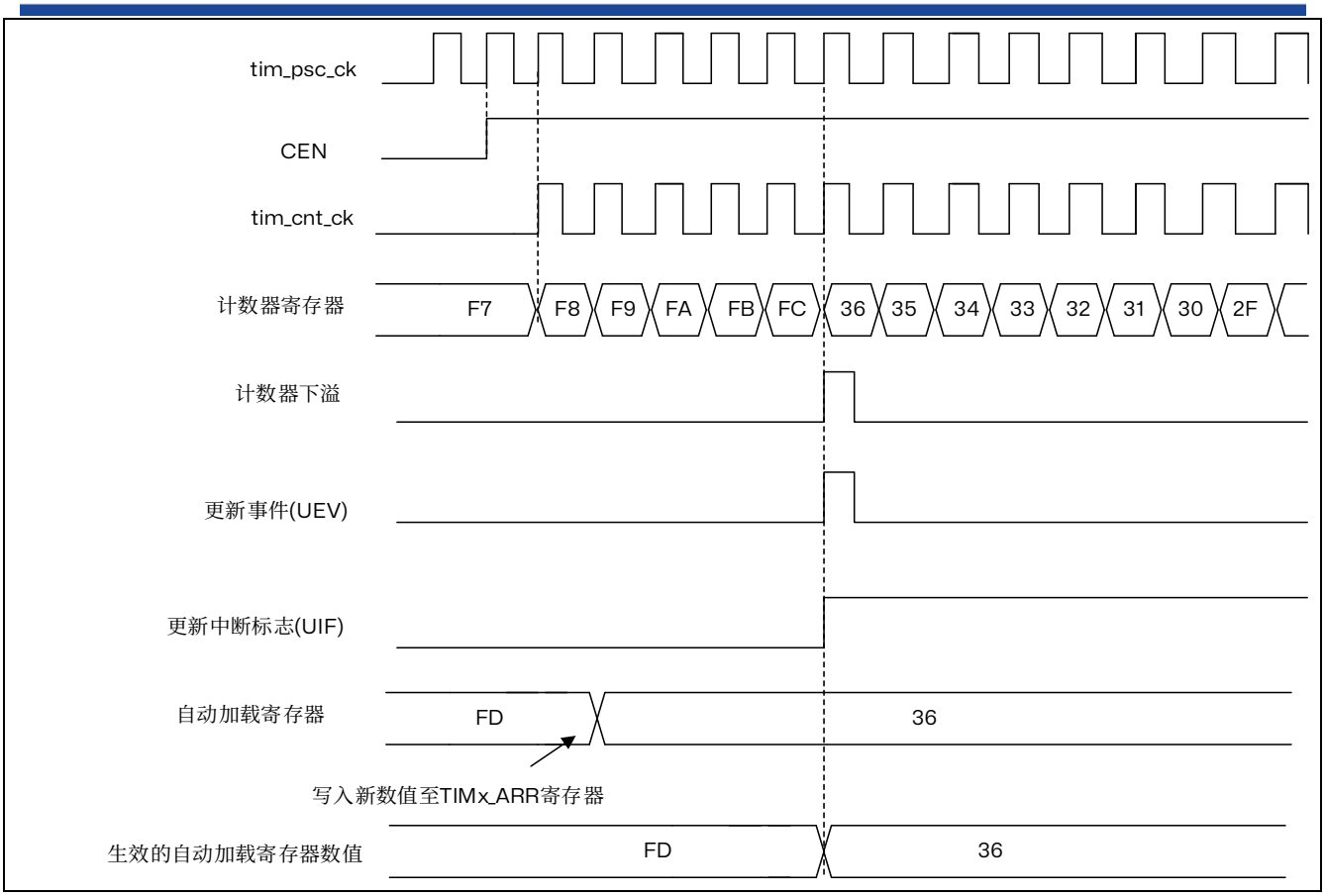



图 22.20 计数器时序图，ARPE=1 时的更新事件（计数器上溢）

### 22.3.5 重复计数器

时基单元介绍如何因计数器上溢/下溢而生成更新事件（UEV）。实际上，只有当重复计数器达到零时，才会生成更新事件。这在生成 PWM 信号时很有用。

这意味着，每当发生 N+1 个计数器上溢或下溢（其中，N 是 TIMx_RCR 重复计数器寄存器中的值），数据就将从预装载寄存器转移到影子寄存器（TIMx_ARR 自动重载寄存器、TIMx_PSC 预分频器寄存器以及比较模式下的 TIMx_CCRx 捕获/比较寄存器）。

重复计数器在下列情况下递减：

- 递增计数模式下的每个计数器上溢。
- 递减计数模式下的每个计数器下溢。
- 中心对齐模式下每个计数器上溢和计数器下溢。

  尽管这限制了最大重复次数为 32768 个 PWM 周期，但它使得每个 PWM 周期可以更新两次占空比。在中心对齐模式下，每个 PWM 周期仅刷新一次比较寄存器时，由于模式的对称性，最大分辨率为 $2xT_{ck}$。

重复计数器是自动重载类型；其重复率为 TIMx_RCR 寄存器所定义的值。当更新事件由软件（通过将 TIMx_EGR 寄存器的 UG 位置 1）或硬件（通过从模式控制器）生成时，无论重复计数器的值为多少，更新事件都将立即发生，并且在重复计数器中重新装载 TIMx_RCR 寄存器的内容。

在中心对齐模式下，如果 RCR 值为奇数，更新事件将在上溢或下溢时发生，这取决于何时写入 RCR 寄存器以及何时启动计数器。如果在启动计数器前写入 RCR，则 UEV 在上溢时发生。如果在启动计数器后写入 RCR，则 UEV 在下溢时发生。

例如，如果 RCR = 3，UEV 将在每个周期的第四个上溢或下溢事件时生成（取决于何时写入 RCR）。

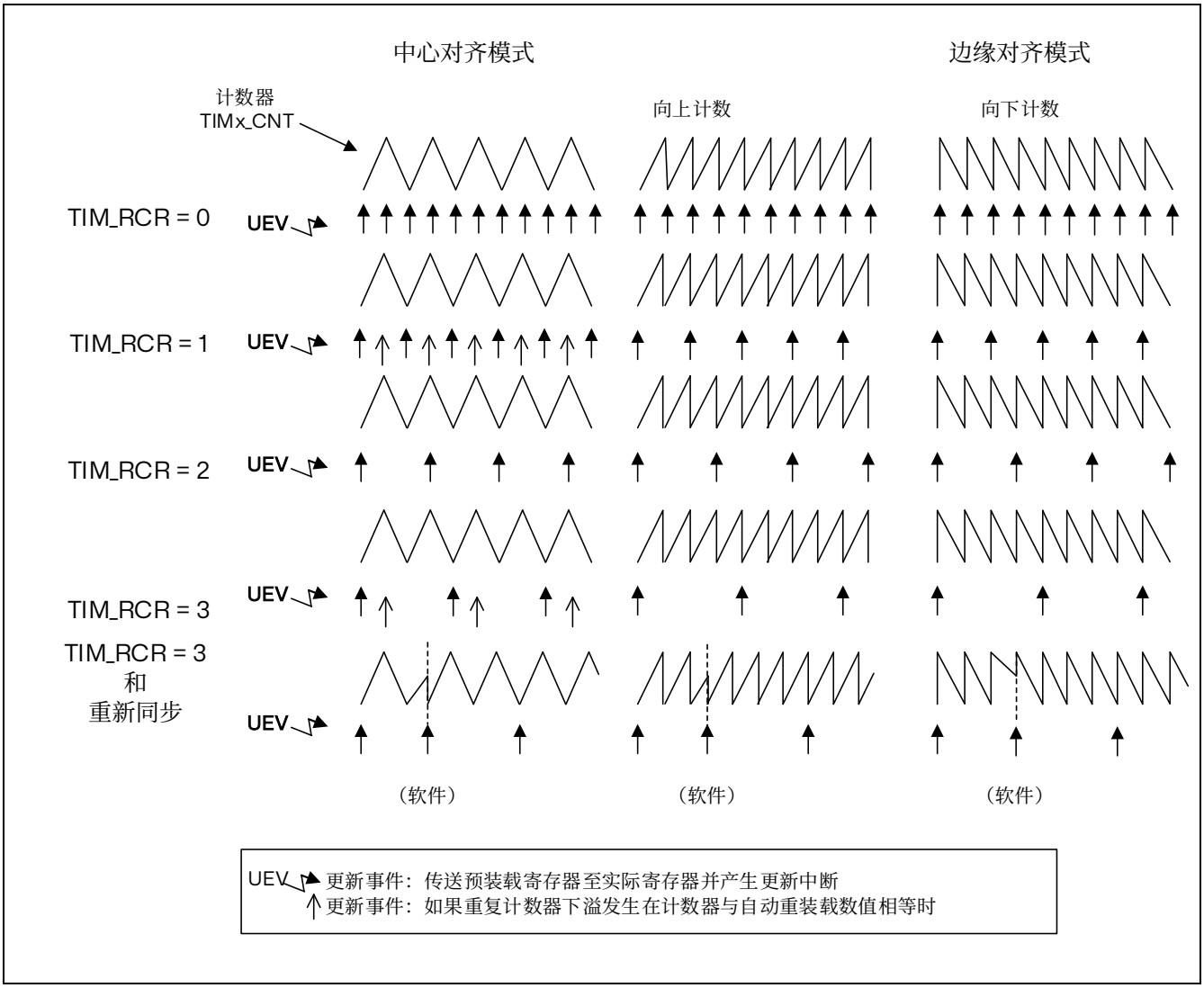



图 22.21 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

### 22.3.6 外部触发输入

定时器具有一个外部触发输入 tim_etr_in。它可以用于：

- 外部时钟（外部时钟模式 2）
- 从模式触发
- PWM 重置输入，用于逐周期电流调节

下图描述了 tim_etr_in 输入调节。输入极性由 TIMxSMCR 寄存器的 ETP 位定义。触发器可以使用 ETPS[1:0]位域编程的分频器进行预分频，并使用 ETF[3:0]位域进行数字滤波。所得信号（tim_etrf）可用于三种目的：作为外部时钟、用于输出调节（通常用于重置 PWM 输出以进行电流限制）以及作为从模式控制器的触发器。

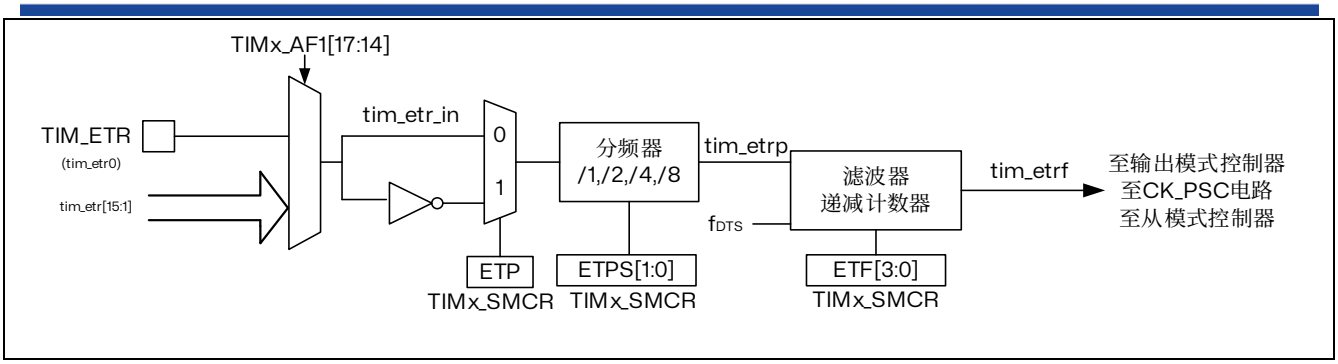



图 22.22 外部触发输入框图

tim_etr_in 输入来自多个来源：输入引脚（默认配置）或内部源。选择是通过 TIMx_AF1 寄存器中的 ETRSEL[3:0]位域完成的。

有关连接到 etr_in 输入的源列表，请参考之前的章节：TIM1/TIM8 引脚和内部信号。

### 22.3.7 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（tim_ker_ck）
- 外部时钟模式 1：外部输入引脚（tim_ti1 或 tim_ti2）
- 外部时钟模式 2：外部触发输入（tim_etr_in）
- 外编码器模式

#### 内部时钟源（tim_ker_ck）

如果禁止从模式控制器（SMS=000），则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍保持自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 tim_ker_ck 提供。

下图显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

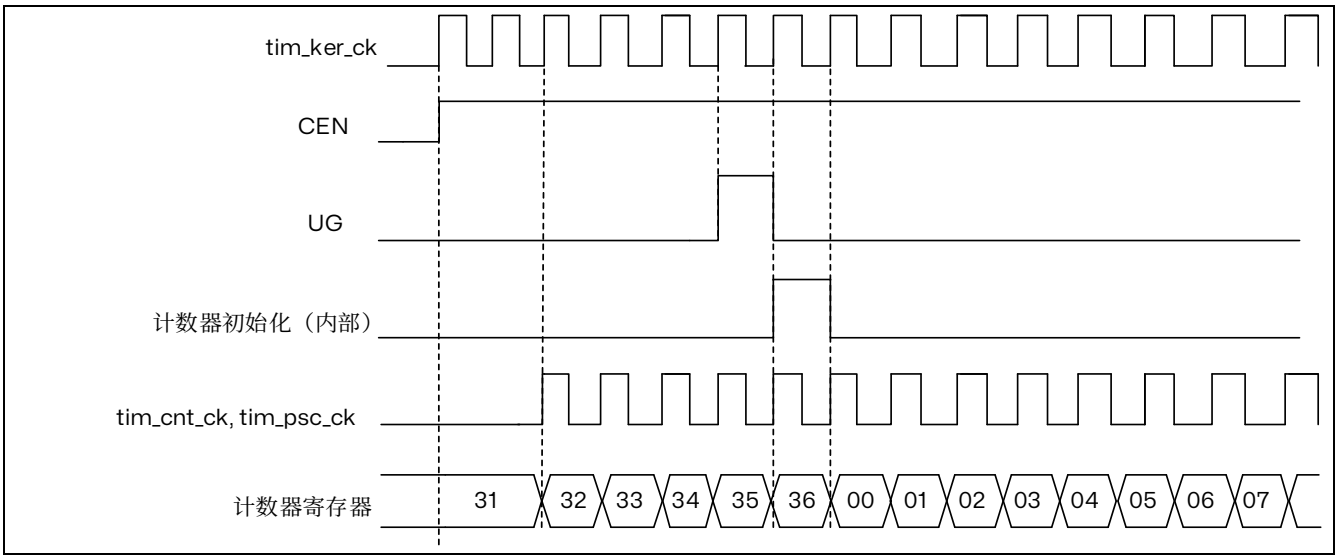



图 22.23 一般模式下的控制电路，内部时钟分频系数为 1

#### 外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

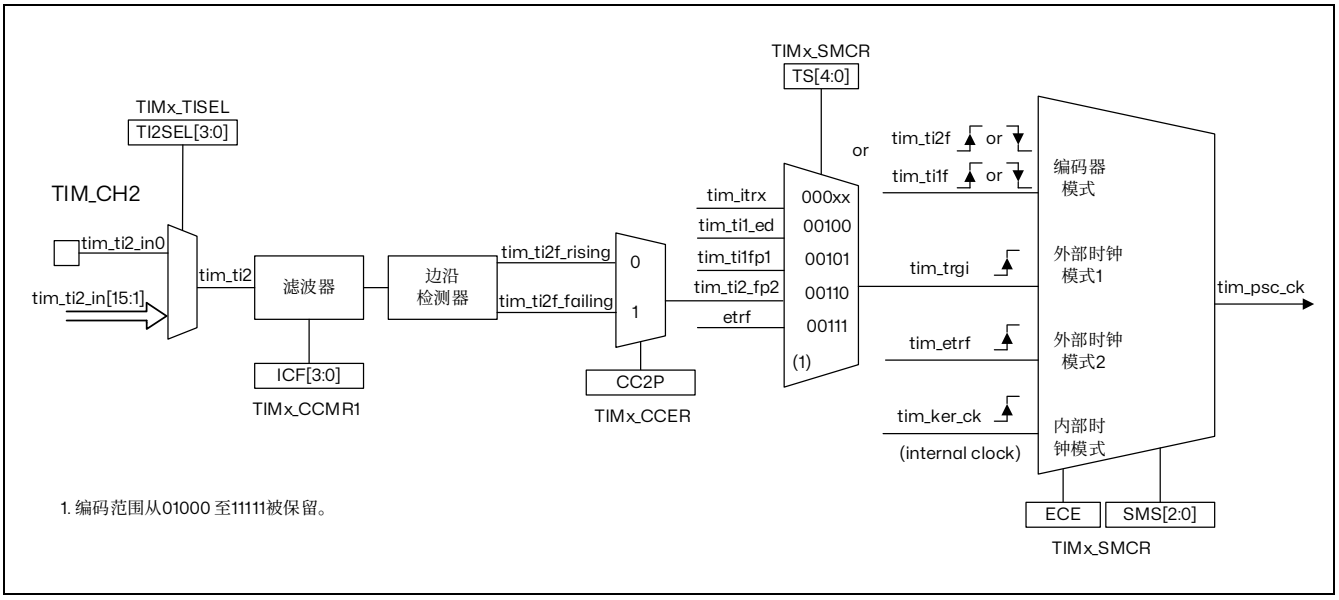



图 22.24 外部触发输入框图

例如，要使递增计数器在 tim_ti2 输入出现上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S = “01”来配置通道 2，使其能够检测 tim_ti2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0]位来配置输入滤波时间（如果不需要任何滤波，请保持 IC2F=0000）。
3. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS=00110 来选择 tim_ti2 作为触发输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

*注：由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

*当* tim_ti2 *出现上升沿时，计数器便会计数一次并且* TIF *标志置*1。

tim_ti2 *的上升沿与实际计数器时钟之间的延迟是由于* tim_ti2 *输入的重新同步电路引起的。*

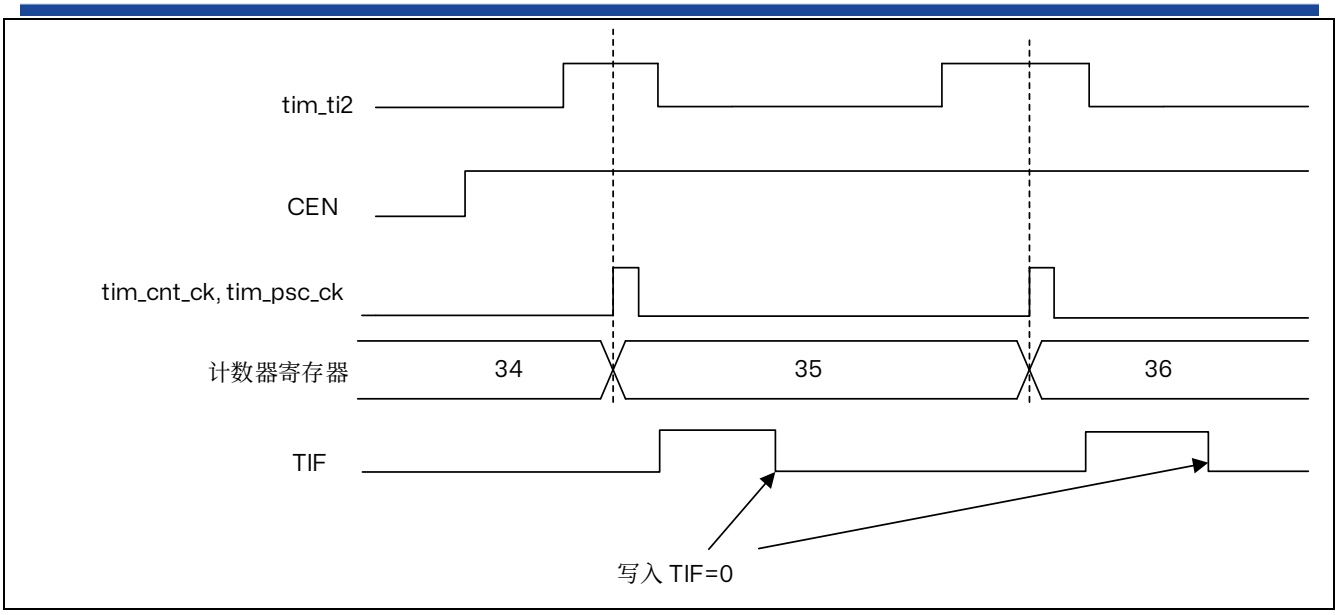



图 22.25 外部时钟模式 1 下的控制电路

**外部时钟模式 2**

通过在 TIMx_SMCR 寄存器中写入 ECE=1 可选择此模式。
计数器可在外部触发输入 tim_etr_in 出现上升沿或下降沿时计数。
下图介绍了外部触发输入模块。

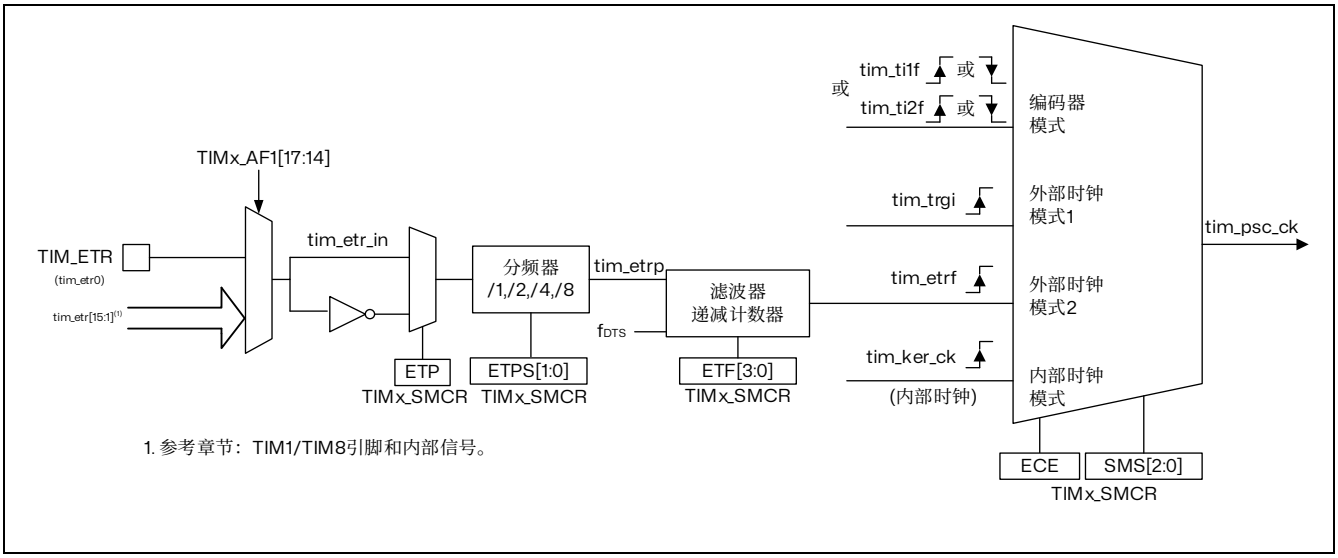



图 22.26 外部触发输入框图

例如，要使递增计数器在 tim_etr_in 每出现 2 个上升沿时计数，请执行以下步骤：

1. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
2. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器。
3. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测。
4. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
5. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

tim_etr_in 每出现 2 个上升沿，计数器计数一次。

在 tim_etr_in 的上升沿和计数器实际时钟之间的延时取决于在 tim_etrp 信号端的重新同步电路。因此，计数器可以正确捕获的最大频率最高为 tim_ker_ck 频率的¼。当 ETRP 信号更快时，用户应通过适当的 ETPS 预分频器设置对外部信号进行分频。

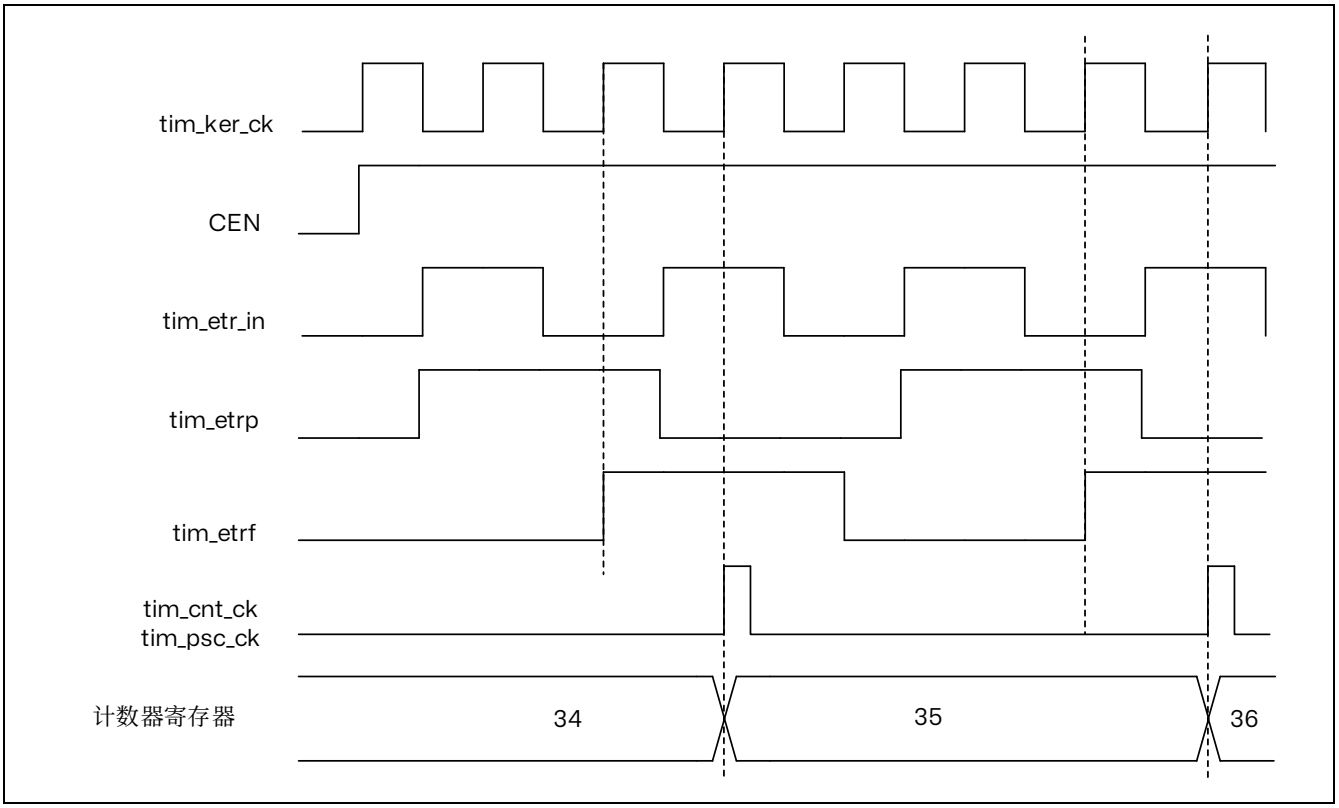


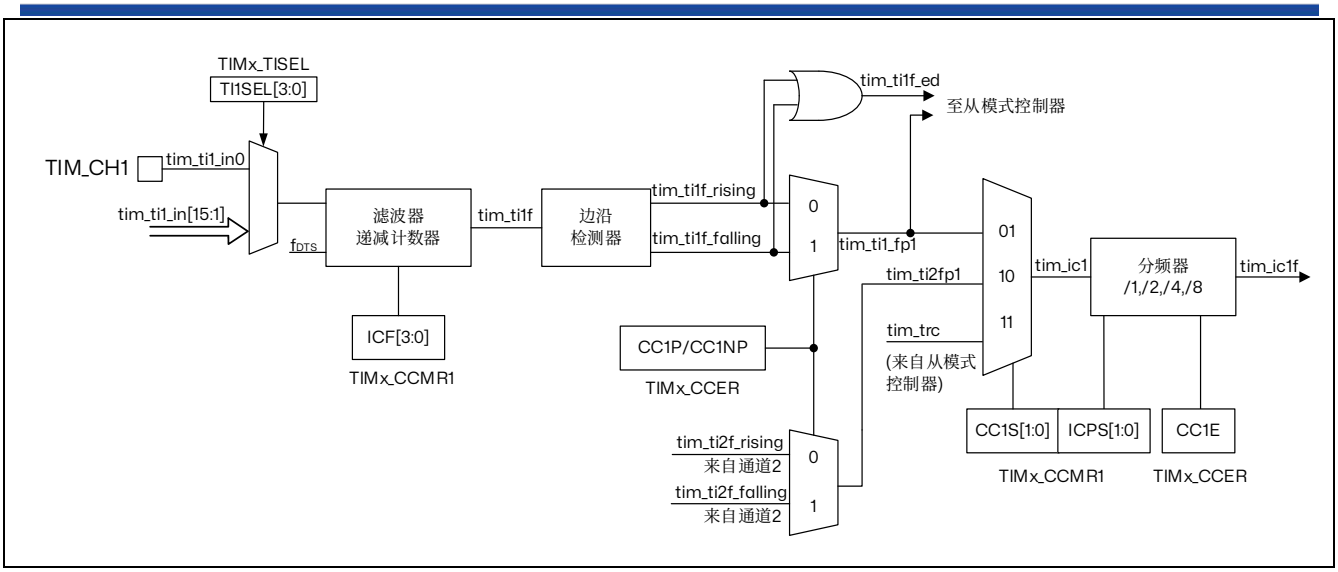

图 22.27 外部时钟模式 2 下的控制电路

### 22.3.8 捕获比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（包括数字滤波、多路复用和预分频器，但通道 5 和通道 6 除外）和一个输出阶段（比较器和输出控制）构建而成。

下列图提供了一个捕获/比较通道的概述。

输入阶段对相应的 tim_tix 输入进行采样，生成一个滤波后的信号 tim_tixf。然后，带有极性选择功能的边沿检测器生成一个信号（tim_tixfpy），该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频（ICxPS），而后再进入捕获寄存器。




输出阶段生成一个中间波形作为基准：tim_ocxref（高电平有效）。链的末端决定最终输出信号的极性。

图 22.28 捕获/比较通道（如：通道 1 输入部分）

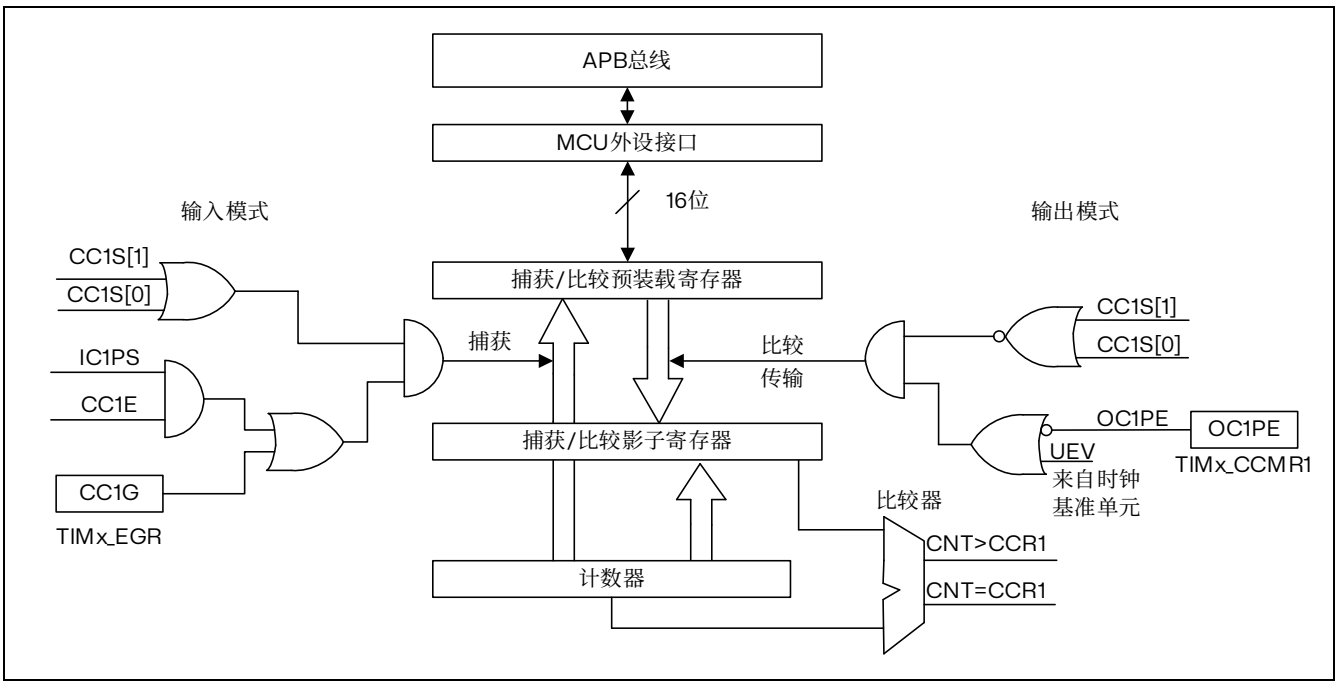



图 22.29 捕获/比较通道 1 的主电路

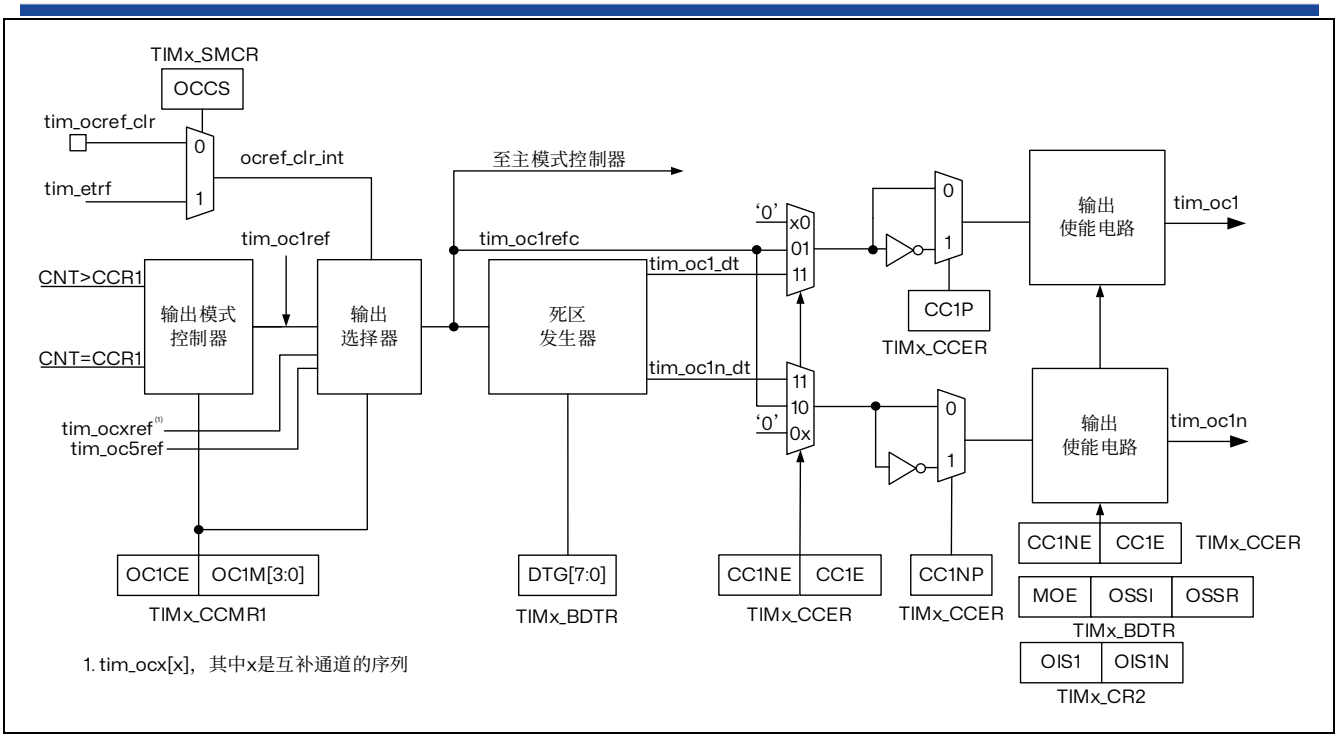



图 22.30 捕获/比较通道的输出阶段（通道 1，通道 2、3 和 4 同上）

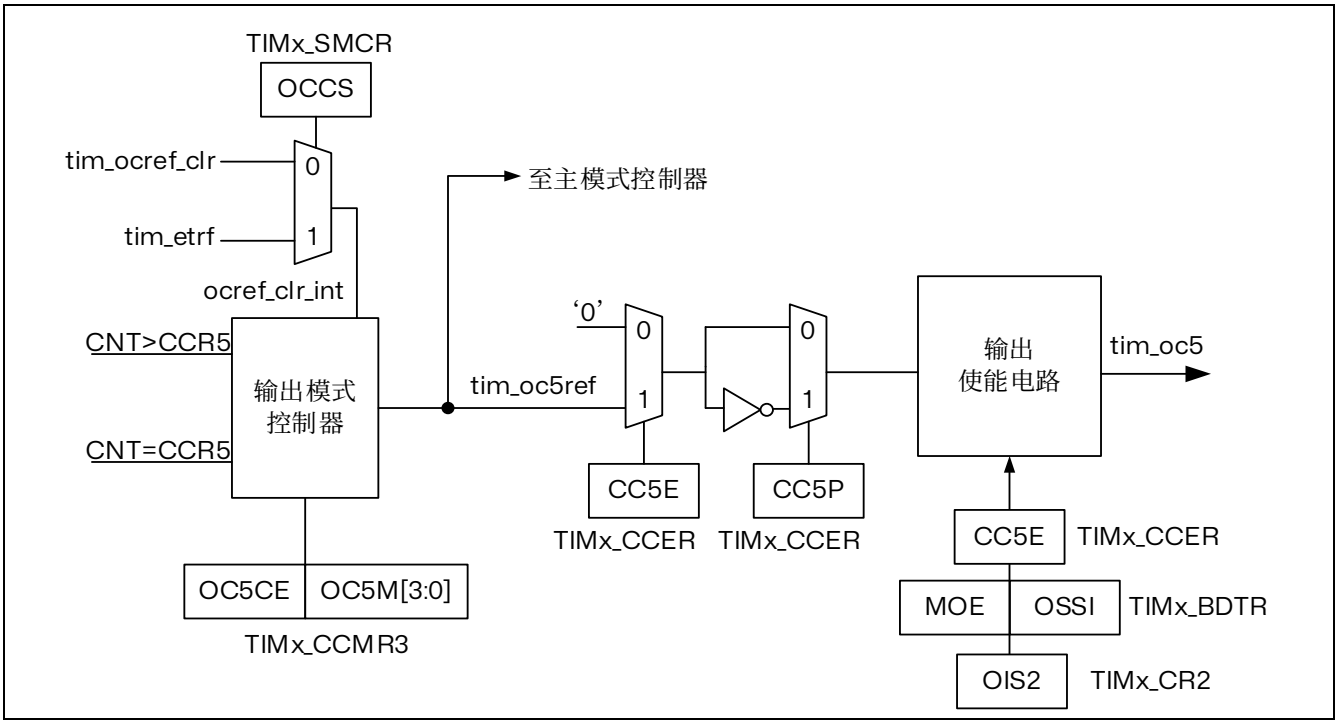



图 22.31 捕获/比较通道的输出阶段（通道 5，通道 6 同上）

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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将被复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

### 22.3.9 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器(TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入 0 后会将其清零。

以下示例说明了如何在 tim_ti1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 选择有效输入：TIMx_CCR1 必须连接到 tim_ti1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
2. 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 tim_tix 输入之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 $f_{DTS}$ 频率采样）后，可以确认 tim_ti1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。
3. 通过向 TIMx_CCER 寄存器中的 CC1P 位和 CC1NP 位写入 0，选择 tim_ti1 通道的有效转换边沿（本例中为上升沿）。
4. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PSC 位写入 00）。
5. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
6. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求，和/或通过将 TIMx_DIER 寄存器中的 CC1DE 位置 1 来使能相关 DMA 请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成一个中断。
- 根据 CC1DE 位生成一个 DMA 请求

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

*注：通过软件将* TIMx_EGR *寄存器中的相应* CCxG *位置*1 *可生成* IC *中断和/或* DMA *请求。*

### 22.3.10 PWM 输入模式

此模式允许测量连接到单个 tim_tix 输入的 PWM 信号的周期和占空比：

- TIMx_CCR1 寄存器保持周期值（两个连续上升沿之间的间隔）。
- TIMx_CCR2 寄存器保持脉冲宽度（两个连续上升沿和下降沿之间的间隔）。

此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 ICx 信号被映射至同一个 tim_tixfp1 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 tim_tixfp 信号之一作为触发输入，并将从模式控制器配置为复位模式。

可通过以下步骤测量施加在 tim_ti1 上的 PWM 信号的周期和脉冲宽度：

1. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 tim_ti1）。
2. 选择 tim_ti1fp1 的有效极性（用于 TIMx_CCR1 中的捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
3. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 tim_ti1）。
4. 选择 tim_ti1fp2 的有效极性（用于 TIMx_CCR2 中的捕获）：向 CC2P 位写入 1 和 CC2NP 位写入“10”（下降沿有效）。
5. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 00101（选择 tim_ti1fp1）。
6. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 0100。
7. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

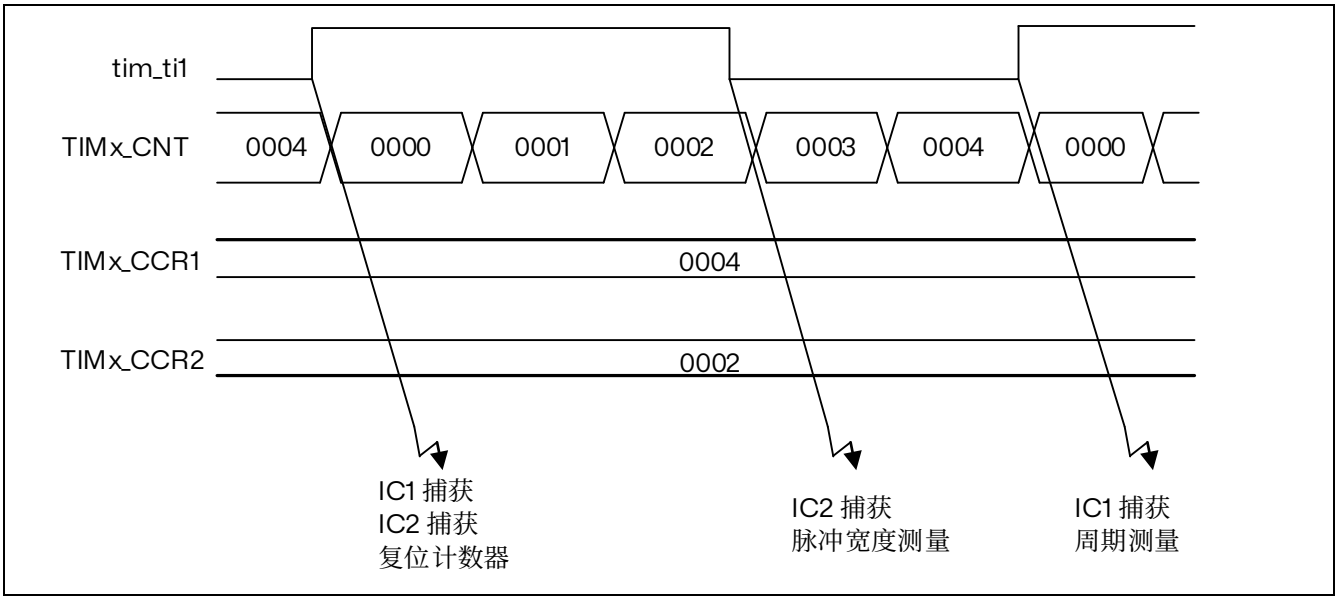



图 22.32 PWM 输入模式时序

### 22.3.11 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位= 00）下，可直接由软件将每个输出比较信号（tim_ocxref 和 tim_ocx/tim_ocxn）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号（tim_ocxref/tim_ocx）强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 0101。tim_ocxref 进而强制设置为高电平（tim_ocxref 始终为高电平有效），同时 tim_ocx 获取 CCxP 极性位的相反值。

例如：CCxP=0（tim_ocx 高电平有效）=> tim_ocx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 tim_ocxref 信号强制设置为低电平。无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断和 DMA 请求。下面的输出比较模式一节对此进行了介绍。

### 22.3.12 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。通道 1 至 4 可以输出，而通道 5 和 6 仅可在微控制器内部使用（例如，用于复合波形生成或 ADC 触发）。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平（OCXM=0000），也可设置为有效电平（OCXM=0001）、无效电平（OCXM=0010）或进行翻转（OCxM=0011）。

- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。
- 如果相应 DMA 使能位（TIMx_DIER 寄存器的 CCxDE 位，TIMx_CR2 寄存器的 CCDS 位，用来选择 DMA 请求）置 1，将发送 DMA 请求。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

在输出比较模式下，更新事件 UEV 对 tim_ocxref 和 tim_ocx 出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，则需将 CCxIE 位置 1。
4. 选择输出模式。例如：
   — 当 CNT 与 CCRx 匹配时，写入 OCxM =0011 以翻转 tim_ocx 输出引脚
   — 写入 OCxPE = 0 以禁止预装载寄存器
   — 写入 CCxP = 0 以选择高电平有效极性
   — 写入 CCxE = 1 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可通过软件随时更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预加载寄存器（OCxPE= “0”，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。下图给出了一个示例。

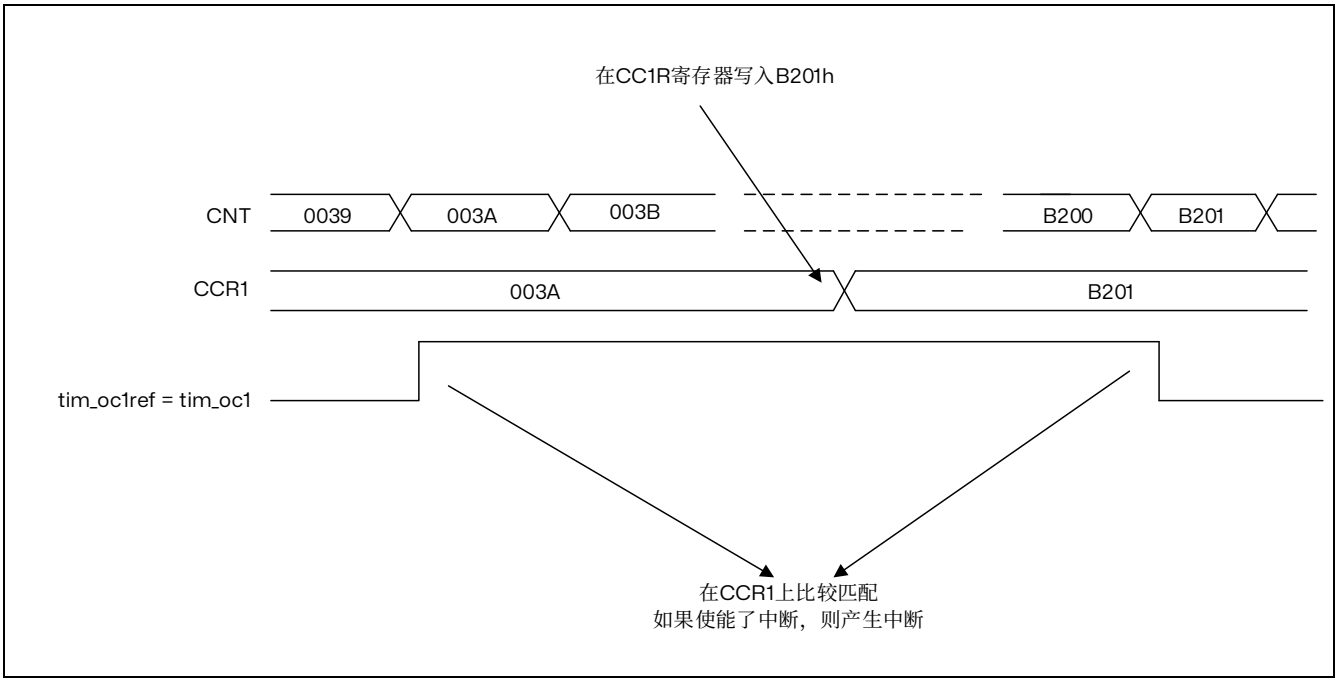



图 22.33 输出比较模式，翻转 OC1

### 22.3.13 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入“0110”（PWM 模式 1）或“0111”（PWM 模式 2），可以独立选择各通道（每个 OCx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器（在递增计数或中心对齐模式下）。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

tim_ocx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。通过 CCxE、CCxNE、MOE、OSSI 和 OSSR 位（TIMx_CCER 和 TIMx_BDTR 寄存器）的组合使能 OCx 输出。有关详细信息，请参见 TIMx_CCER 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 总是与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx ≤ TIMx_CNT 还是 TIMx_CNT ≤ TIMx_CCRx（取决于计数器计数方向）。

根据 TIMx_CR1 寄存器中的 CMS 位状态，定时器能够产生边沿对齐模式或中心对齐模式的 PWM 信号。

**PWM 边沿对齐模式**

- 递增计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。请参见递增计数模式一节。

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 tim_ocxref 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 tim_ocxref 保持为“1”。如果比较值为 0，则 tim_ocxref 保持为“0”。下图举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

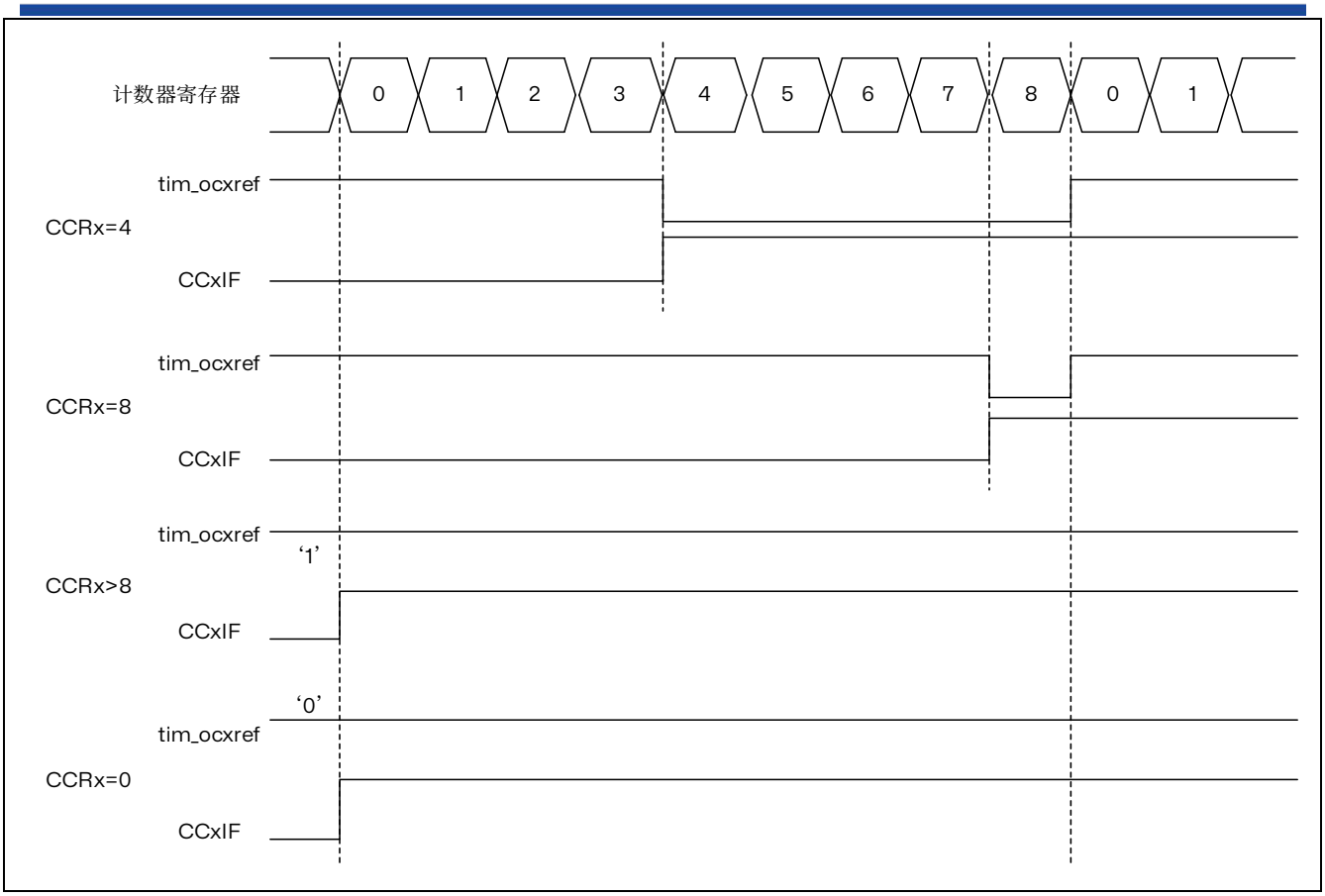



图 22.34 边沿对齐的 PWM 波形（ARR=8）

- 递减计数配置

当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。请参见递减计数模式一节。

在 PWM 模式 1 下，只要 TIMx_CNT > TIMx_CCRx，参考信号 tim_ocxref 即为低电平，否则其为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，tim_ocxref 保持为“1”。此模式下不可能产生 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”（其余所有配置对 tim_ocxref/tim_ocx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位（DIR）由硬件更新，不得通过软件更改。请参见中心对齐模式（递增/递减计数）。

下图显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8
- PWM 模式为 PWM 模式 1,
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

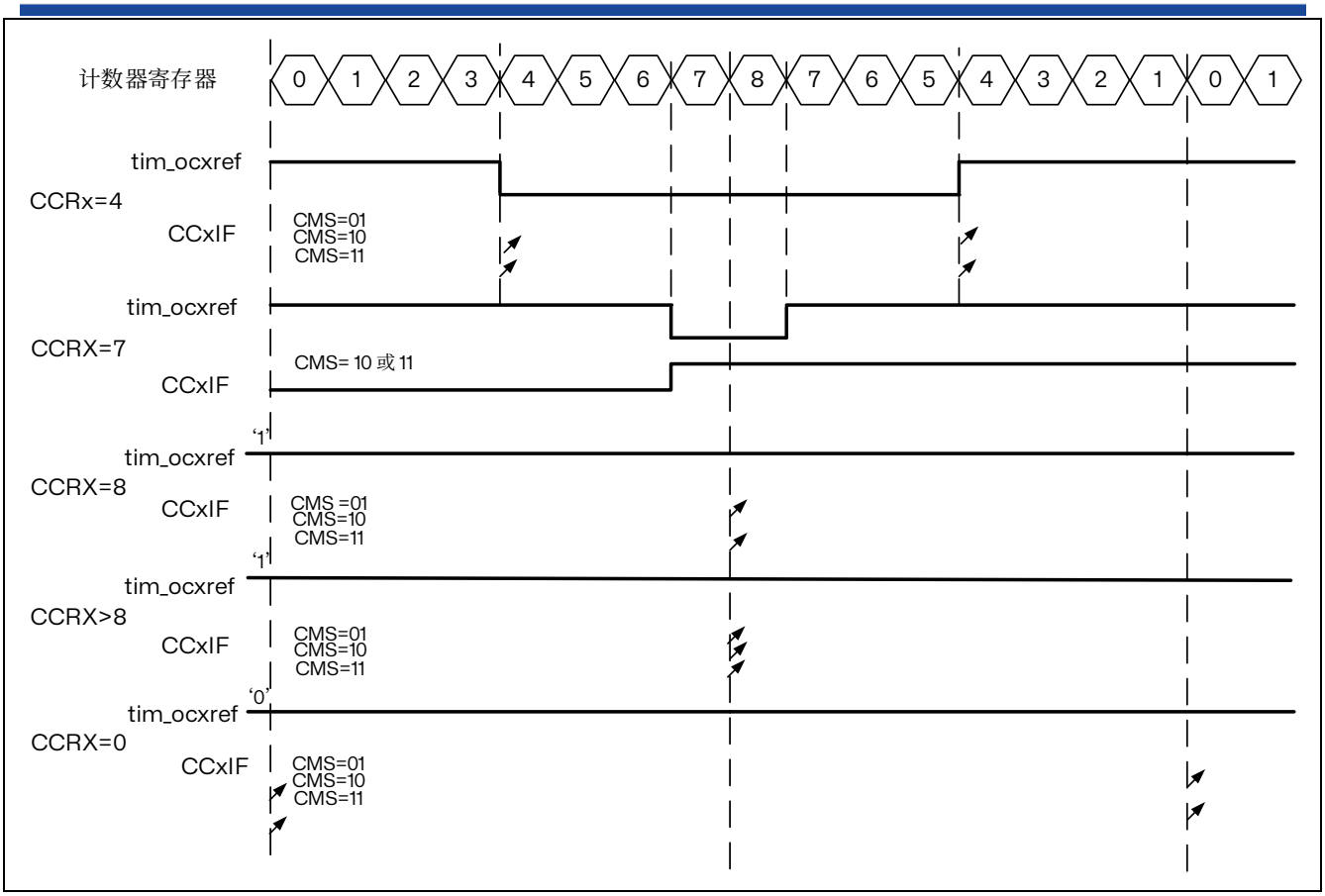



图 22.35 中央对齐的 PWM 波形（ARR=8）

关于使用中央对齐模式的提示：

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。
- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：
  — 如果写入计数器中的值大于自动重载值（TIMx_CNT>TIMx_ARR），计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
  — 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。
- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

**抖动模式**

通过使能抖动模式，使用 TIMx_CR1 寄存器中的 DITHEN 位，可以提高 PWM 模式的有效分辨率。这既适用于 CCR（用于增加占空比分辨率），也适用于 ARR（用于增加 PWM 频率分辨率）。

工作原理是在 16 个连续的 PWM 周期中，以预定义的模式使实际的 CCR（或 ARR）值略有变化（增加或减少一个定时器时钟周期）。这使得占空比或 PWM 期间的平均值提高了 16 倍分辨率。下图展示了应用于 4 个连续 PWM 周期的抖动原理。

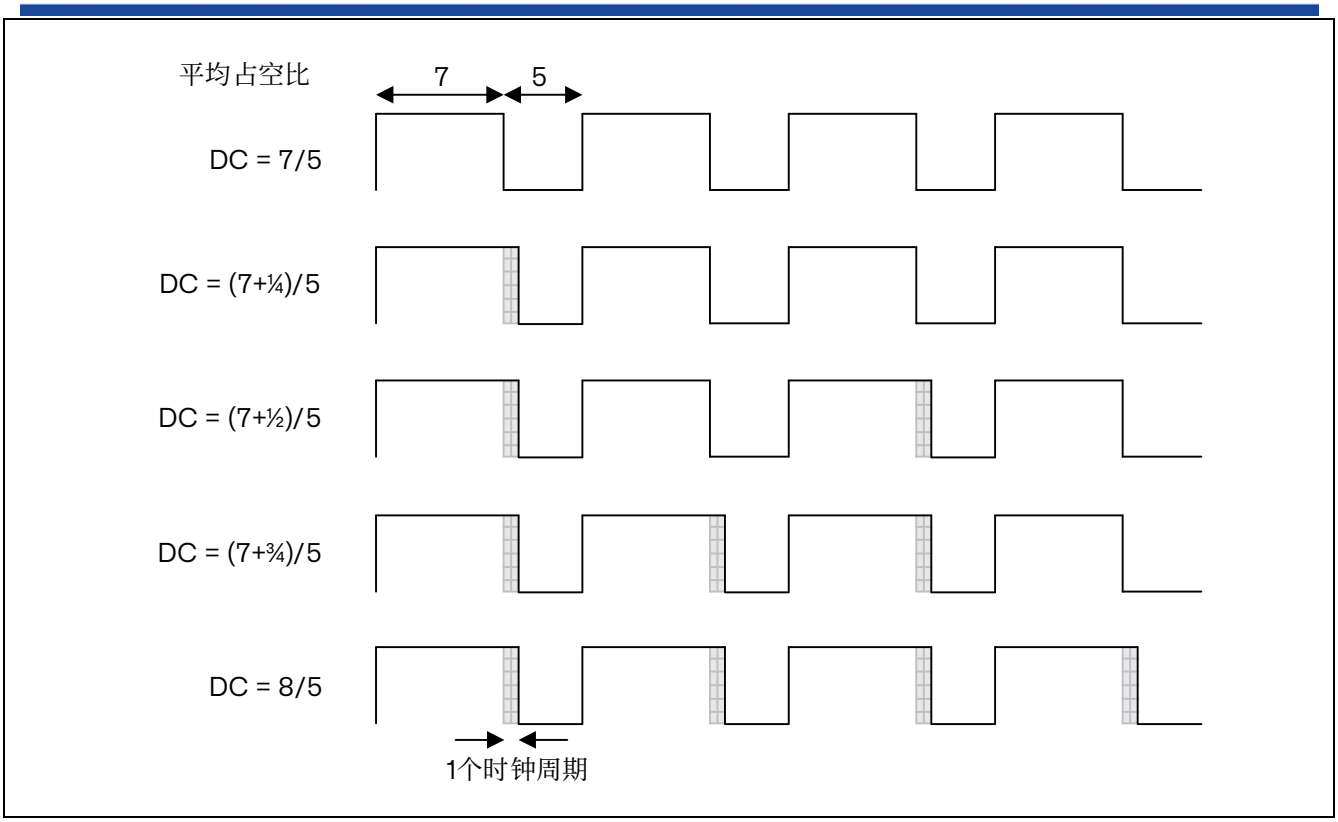



图 22.36 抖动原理

当抖动模式被使能时，寄存器编码将如下所示进行变化（参见下图示例）

- 4 个最低有效位（LSB）用于编码增强的分辨率部分（小数部分）
- 最高有效位（MSB）左移至 19:4 位，用于编码基本值

*注：如果设置清除*DITHEN *位，*ARR *和*CCR *值将自动更新。（例如，如果*ARR=0x05 *且*DITHEN=0，*它将更新为*ARR=0x50 *且*DITHEN=1*）。*

1. *必须重置*CEN *和*ARPE *位*
2. ARR[3:0]*位必须重置*
3. DITHEN *位必须重置*
4. CCIF *标志必须清除*
5. *可以设置*CEN *位（最终设置*ARPE=1*）。*

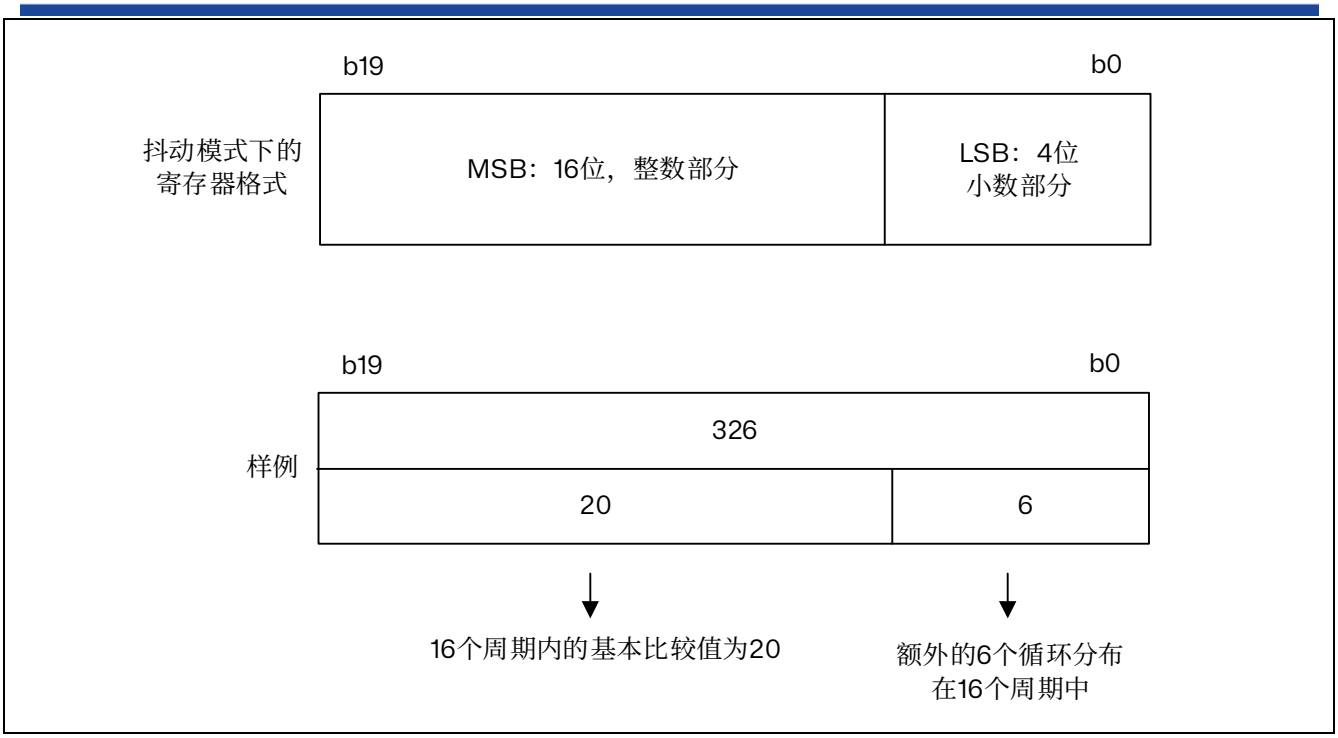



图 22.37 抖动模式下的数据格式和寄存器编码

最小频率由以下公式给出：

$$\text{分辨率} = \frac{\text{FTim}}{\text{Fpwm}} \Rightarrow \text{FpwmMin} = \frac{\text{FTim}}{\text{最大分辨率}}$$

$$\text{禁用抖动模式：FpwmMin} = \frac{\text{FTim}}{65536}$$

$$\text{使能抖动模式：FpwmMin} = \frac{\text{FTim}}{65535 + \frac{15}{16}}$$

*注意：在抖动模式下，*TIMx_ARR *和*TIMx_CCRy *的最大值限制为*0xFFFEF *（对应于整数部分的*65534 *和抖动部分的*15*）。*

如下图所示，无论 PWM 频率如何，抖动模式都可以增加 PWM 分辨率。

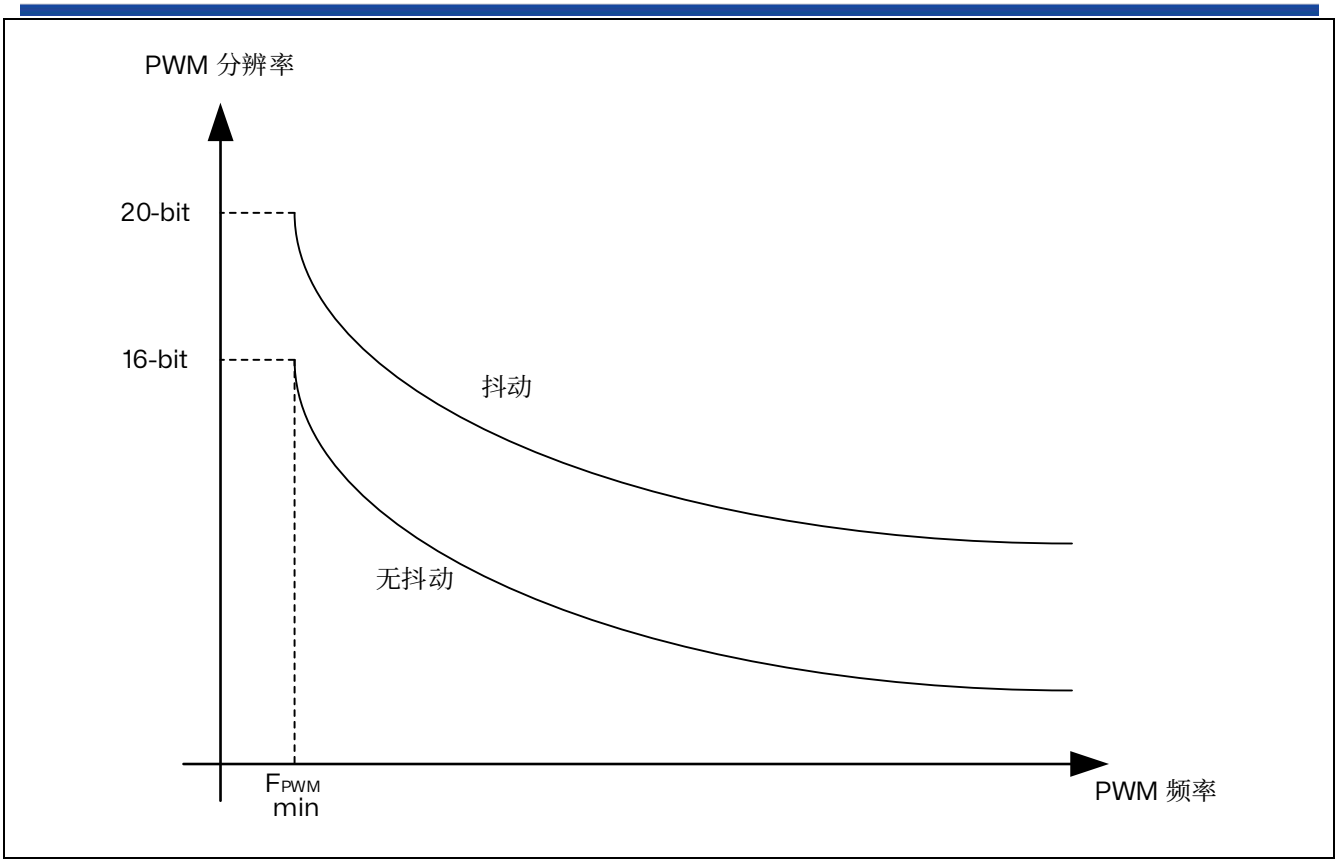



图 22.38 PWM 分辨率 vs 频率

如下图所示，占空比和/或周期的变化分散在 16 个连续周期。

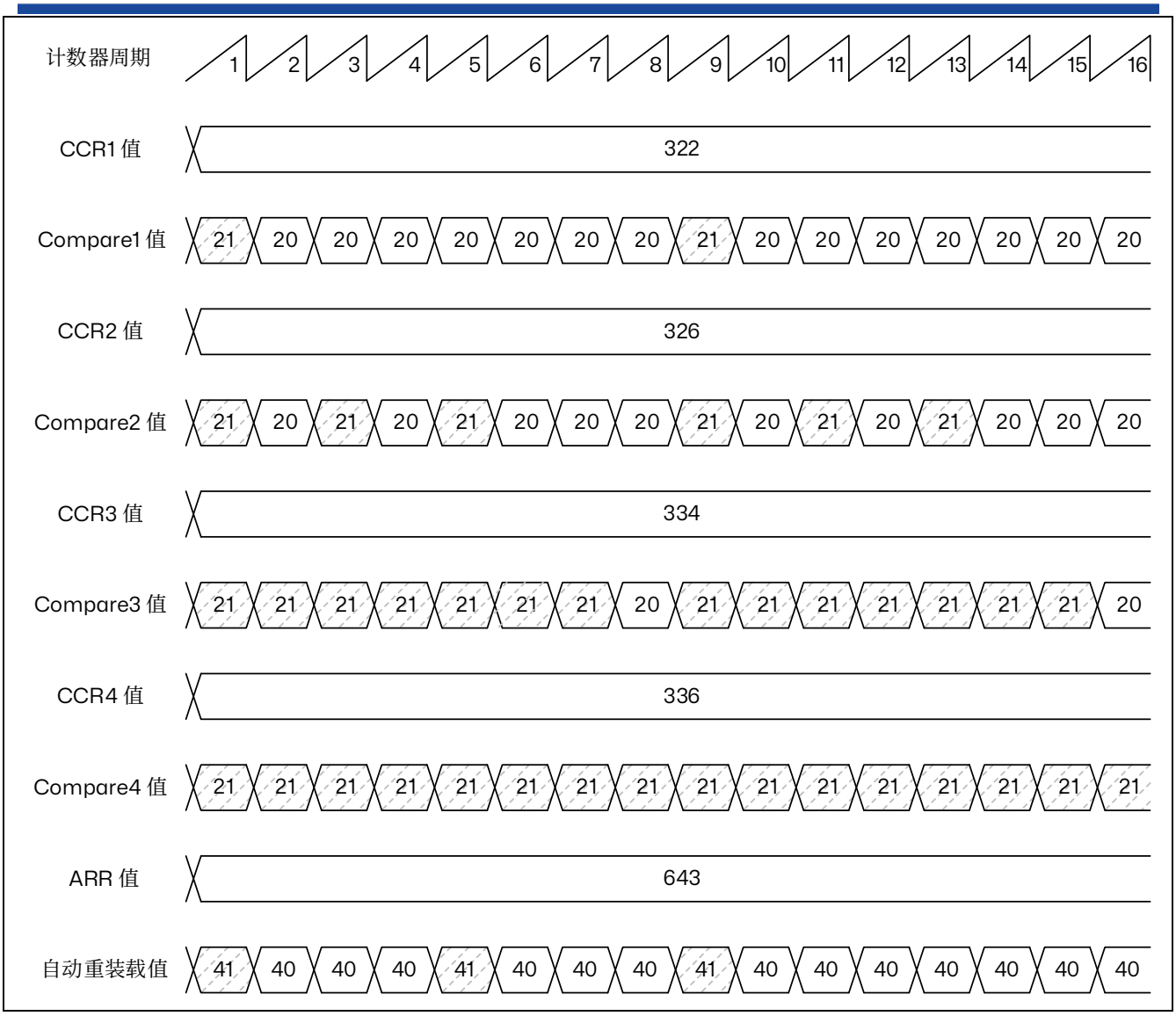



图 22.39 PWM 抖动模式

自动重新加载和比较值的增量按照下表中描述的特定模式分布。抖动序列是为了使增量分布尽可能均匀并最大限度地减少整体波动。

表 22.13 CCR 和 ARR 寄存器的变化抖动模式

| LSB 值 | 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 | - |

在中心对齐 PWM 模式下（TIMx_CR1 寄存器中的 CMS 位不为“00”），也提供了抖动模式。在这种模式下，抖动模式考虑到递增递减计数阶段，应用于 8 个连续的 PWM 周期，如下图所示。

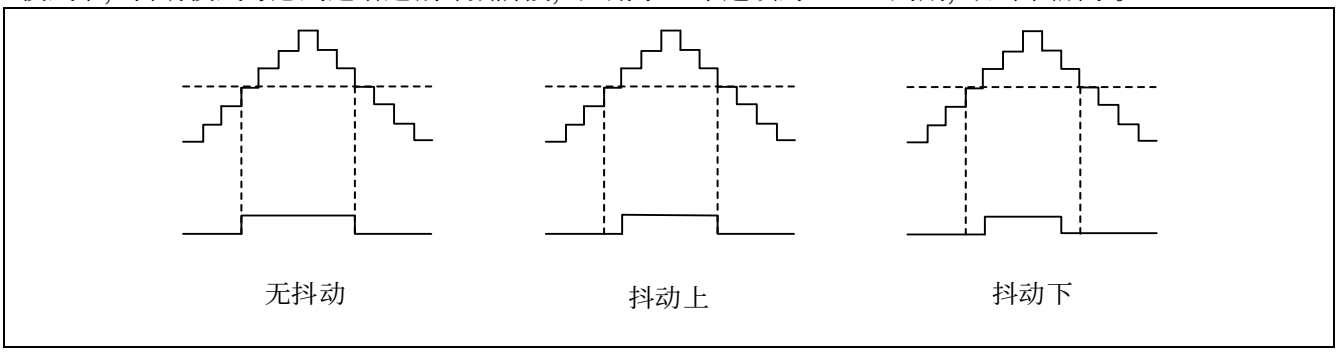



图 22.40 中心对齐 PWM 模式下抖动对占空比的影响

下表显示了如何在中心对齐 PWM 模式下添加抖动模式。

表 22.14 CCR 寄存器在中心对齐 PWM 模式下的变化抖动模式

| LSB 值 | PWM 周期 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | |
| | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 |
| 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 | - |

### 22.3.14 非对称 PWM 模式

非对称模式允许生成两个中心对齐的 PWM 信号，并具有可编程的相位偏移。虽然频率由 TIMx_ARR 寄存器的值决定，但占空比和相位偏移由一对 TIMx_CCRx 寄存器确定。一个寄存器控制 PWM 在计数上升期间，第二个寄存器在计数下降期间，这样 PWM 每半个 PWM 周期调整一次：

— tim_oc1refc（或 tim_oc2refc）由 TIMx_CCR1 和 TIMx_CCR2 控制
— tim_oc3refc（或 tim_oc4refc）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过将“1110”（非对称 PWM 模式 1）或“1111”（非对称 PWM 模式 2）写入 TIMx_CCMRx 寄存器的 OCxM 位，可以在两个通道上独立选择非对称 PWM 模式（每个 CCR 寄存器对一个 tim_ocx 输出）。

*注意：为了兼容性原因，*OCxM[3:0]*位域被分为两个部分，最高有效位与最低*3 *位不连续。*

当某个通道用作非对称 PWM 通道时，其互补通道也可以使用。例如，如果 tim_oc1refc 信号在通道 1 上生产（非对称 PWM 模式 1），可以在通道 2 上输出 tim_oc1refc 信号，或者产生非对称 PWM 模式 1 的结果 tim_oc2refc 信号。

下图标识使用非对称 PWM 模式生成信号的示例（通道 1 到 4 配置为非对称 PWM 模式 2）。与死区时间生成器一起使用，可以控制全桥移相 DC 到 DC 转换器。

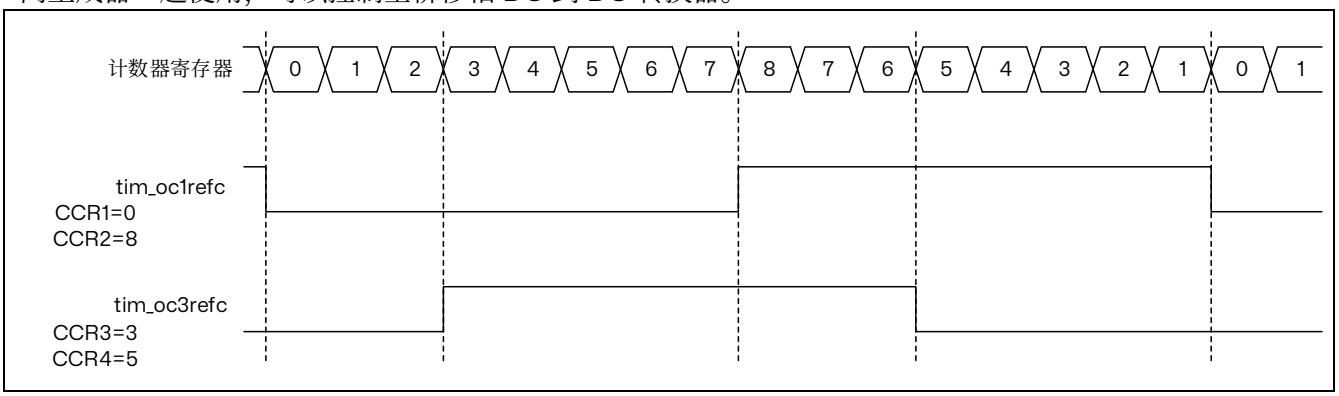



图 22.41 生成两个具有 50%占空比的相位偏移 PWM 信号

### 22.3.15 组合 PWM 模式

组合 PWM 模式允许生成两个边缘对齐或中心对齐的 PWM 信号，各自脉冲之间的可编程延迟和相位偏移。虽然频率由 TIMx_ARR 寄存器的值决定，但占空比和延迟由两个 TIMx_CCRx 寄存器确定。结果信号 tim_ocxrefc 由两个参考 PWM 的 OR 或 AND 逻辑组合而成：

— tim_oc1refc（或 tim_oc2refc）由 TIMx_CCR1 和 TIMx_CCR2 控制
— tim_oc3refc（或 tim_oc4refc）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过在 TIMx_CCMRx 寄存器的 OCxM 位中写入“1100”（组合 PWM 模式 1）或“1101”（组合 PWM 模式 2），可以在两个通道上独立选组合 PWM 模式（每个 CCR 寄存器对一个 tim_ocx 输出）。

当某个通道用作组合 PWM 通道时，其互补通道必须配置为相反的 PWM 模式（例如，一个通道为组合 PWM 模式 1，另一个通道为组合 PWM 模式 2）。

*注意：为了兼容性原因，*OCxM[3:0]*位域被分为两个部分，最高有效位与最低*3 *位不连续。*

下图表示使用组合 PWM 模式生成信号的示例，该示例的配置如下：

— 通道 1 配置为组合 PWM 模式 2。
— 通道 2 配置为 PWM 模式 1。
— 通道 3 配置为组合 PWM 模式 2。

— 通道 4 配置为 PWM 模式 1。

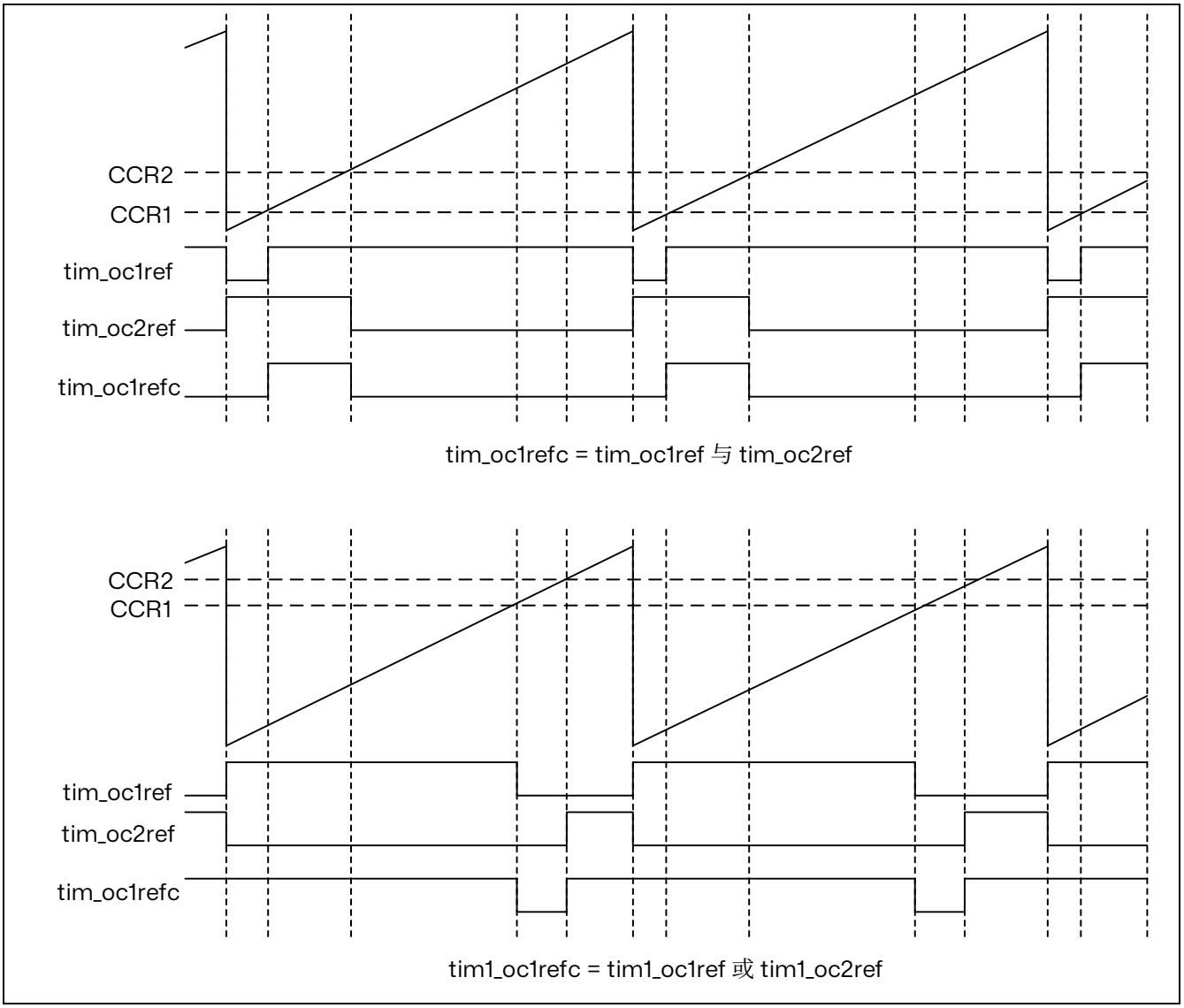



图 22.42 通道 1 和通道 3 的组合 PWM 模式

### 22.3.16 三相组合 PWM 模式

三相组合 PWM 模式允许生成一个到三个以脉冲中间的单个可编程信号 AND 为中心对齐的 PWM 信号。使用 tim_oc5ref 信号定义结果组合信号。TIMx_CCR5 中的 3 位 GC5C[3:1]允许选择 tim_oc5ref 与哪个参考信号组合。结果信号 tim_ocxrefc 由两个参考 PWM 的 AND 逻辑组合而成。

— 如果设置了 GC5C1，则 tim_oc1refc 由 TIMx_CCR1 和 TIMx_CCR5 控制
— 如果设置了 GC5C2，则 tim_oc2refc 由 TIMx_CCR2 和 TIMx_CCR5 控制
— 如果设置了 GC5C3，则 tim_oc3refc 由 TIMx_CCR3 和 TIMx_CCR5 控制

通过设置 3 位 GC5C[3:1]中的至少一位，可以在通道 1 到 3 上独立选择三相组合 PWM 模式。

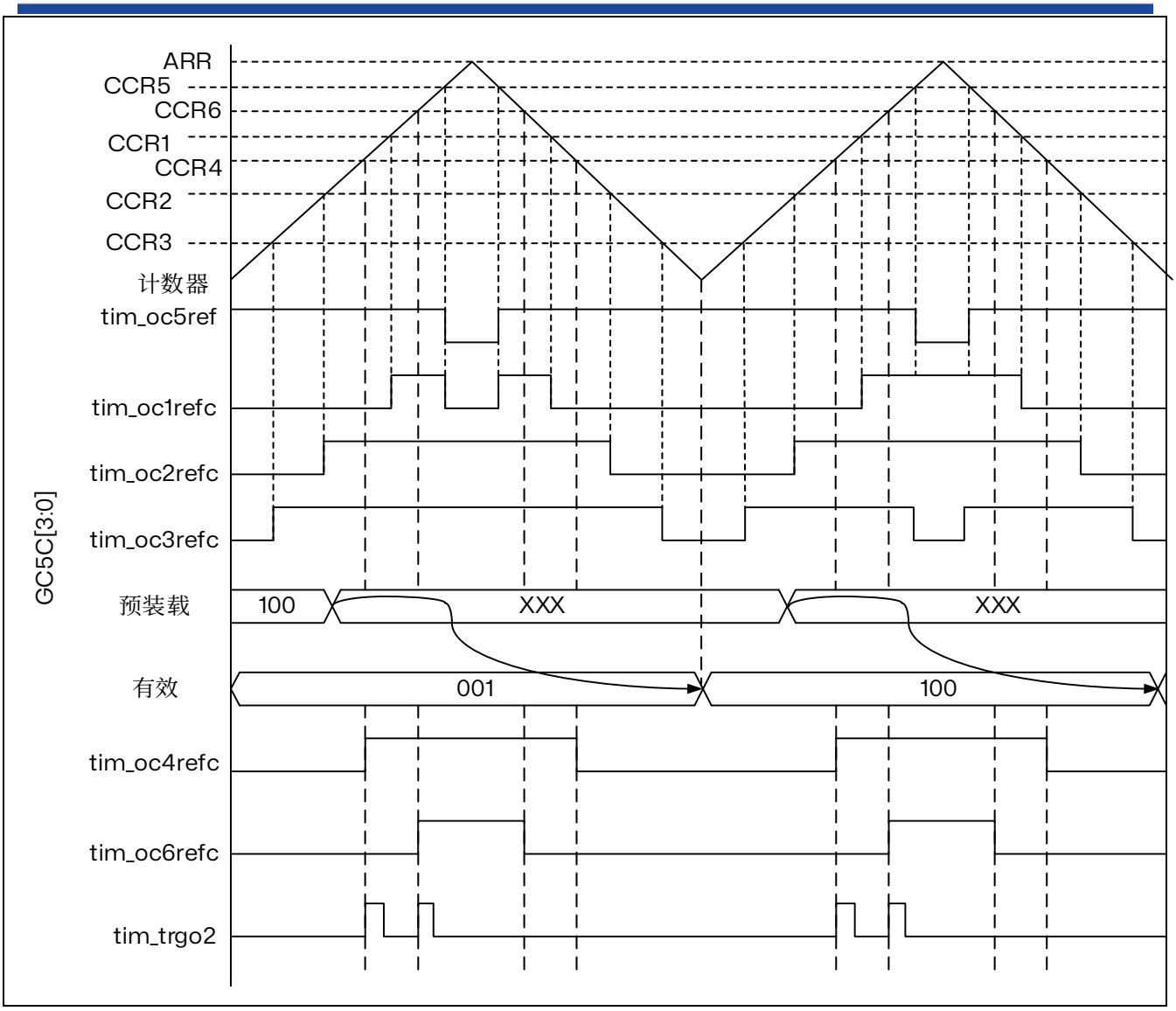



图 22.43 每周期具有多个触发脉冲的三相组合 PWM 信号

tim_trgo2 波形显示了 ADC 如何对给定的三相 PWM 信号进行同步。更多详细信息请参考本章后续小节：ADC 触发器。

### 22.3.17 互补输出和死区插入

高级控制定时器（TIM1/TIM8）可以输出两路互补信号，并管理输出的关断与接通瞬间。

这段时间通常称为死区，用户必须根据与输出相连接的器件及其特性（电平转换器的固有延迟、开关器件产生的延迟...）来调整死区时间。

每路输出可以独立选择输出极性（主输出 tim_ocx 或互补输出 tim_ocxn）。可通过对 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位执行写操作来完成极性选择。

互补信号 tim_ocx 和 tim_ocxn 通过以下多个控制位的组合进行激活：TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位以及 TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位。更多详细信息，请参考表格：带刹车功能的互补输出通道 tim_ocx 和 tim_ocxn 的控制位。应当注意，切换至 idle（MOE 下降到 0）的时刻，死区仍然有效。

死区时间插入功能是同时设置 CCxE 和 CCxNE 位以及（如果存在刹车电路）MOE 位来使能的。每

个通道都有一个 10 位的死区时间生成器。将基于参考波形 tim_ocxref 生成 2 个输出 tim_ocx 和 tim_ocxn。如果 tim_ocx 和 tim_ocxn 为高电平有效：

- 输出信号 tim_ocx 与参考信号相同，只是其上升沿相对参考上升沿存在延迟。
- 输出信号 tim_ocxn 与参考信号相反，并且其上升沿相对参考下降沿存在延迟。

如果延迟时间大于有效输出（tim_ocx 或 tim_ocxn）的宽度，则不会产生相应的脉冲。

下图所示为死区发生器的输出信号与参考信号 tim_ocxref 之间的关系。（在这些示例中，假定 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

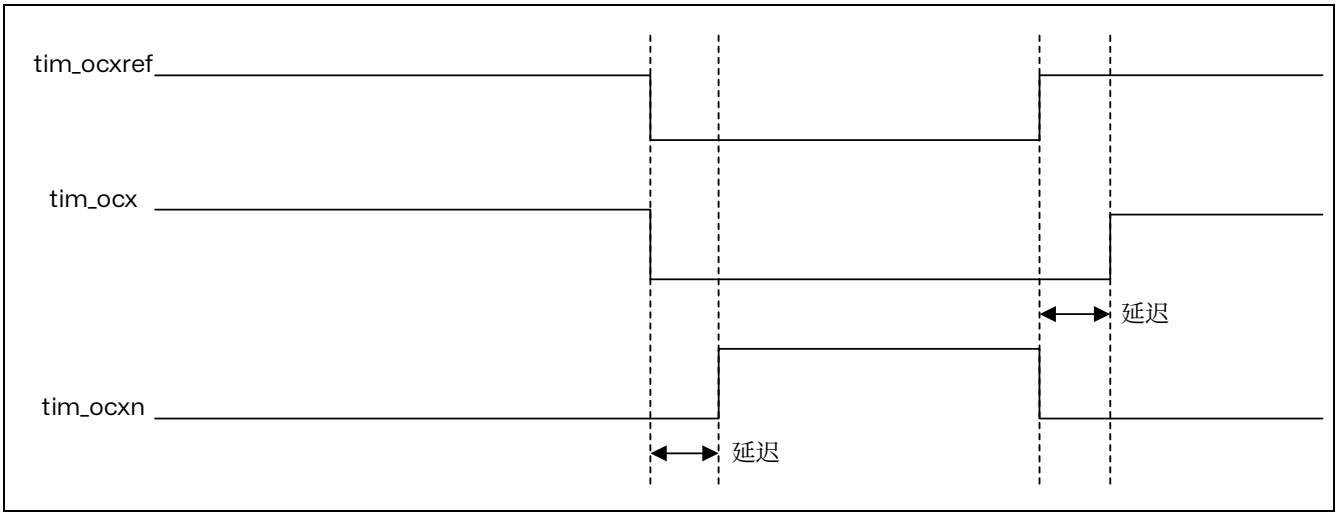



图 22.44 带死区插入的互补输出

TIMx_DTR2 中的 DTAE 位允许对参考信号上升沿和下降沿的死区值进行区分，如下图所示。

在非对称模式（DTAE=1）下，上升沿参考死区由 TIMx_BDTR 寄存器中的 DTG[7:0]位域定义，而下降沿参考死区由 TIMx_DTR2 寄存器中的 DTGF[7:0]位域定义。在使能计数器之前，必须写入 DTAE 位，并且在 CEN=1 时不得修改。

使用预加载机制可以在 PWM 操作期间动态更新死区值。当 TIMx_DTR2 寄存器中的 DTPE 位被设置时，死区位域 DTG[7:0]和 DTGF[7:0]将被预加载。预加载值将在下一个更新事件上加载到活动寄存器中。

*注意：如果* DTPE *位在计数器使能时被使能。自上次更新以来写入的新值将被丢弃，并使用先前的值。*

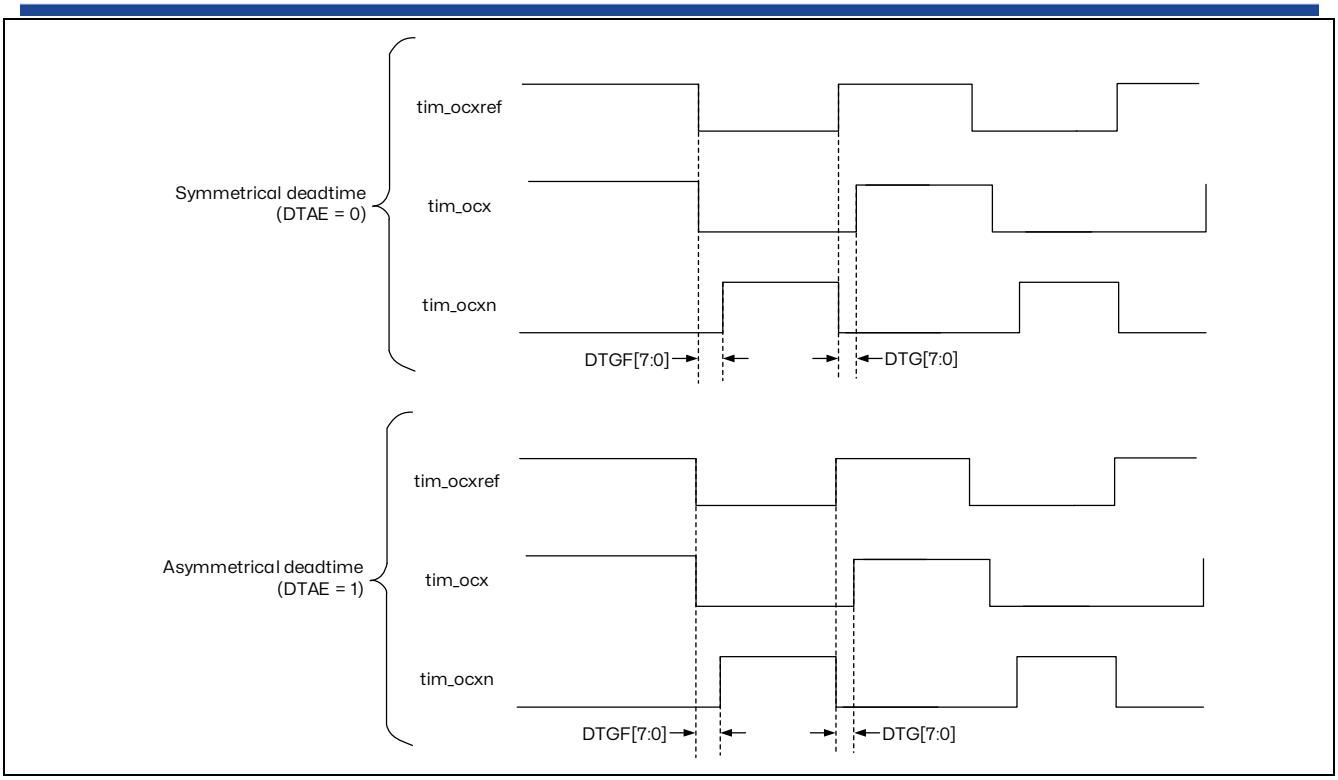



图 22.45 非对称死区时间

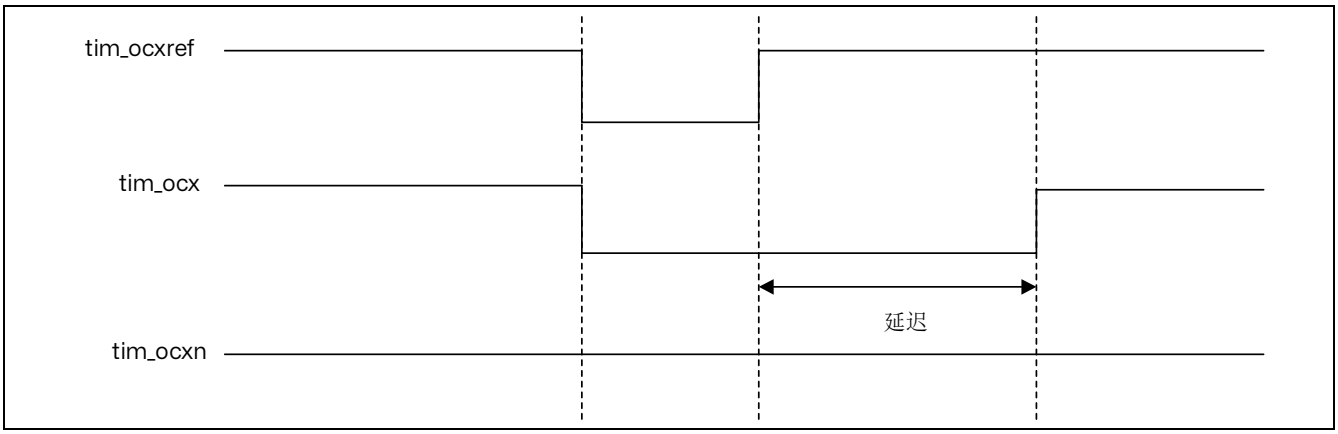



图 22.46 死区波形延迟大于负脉冲

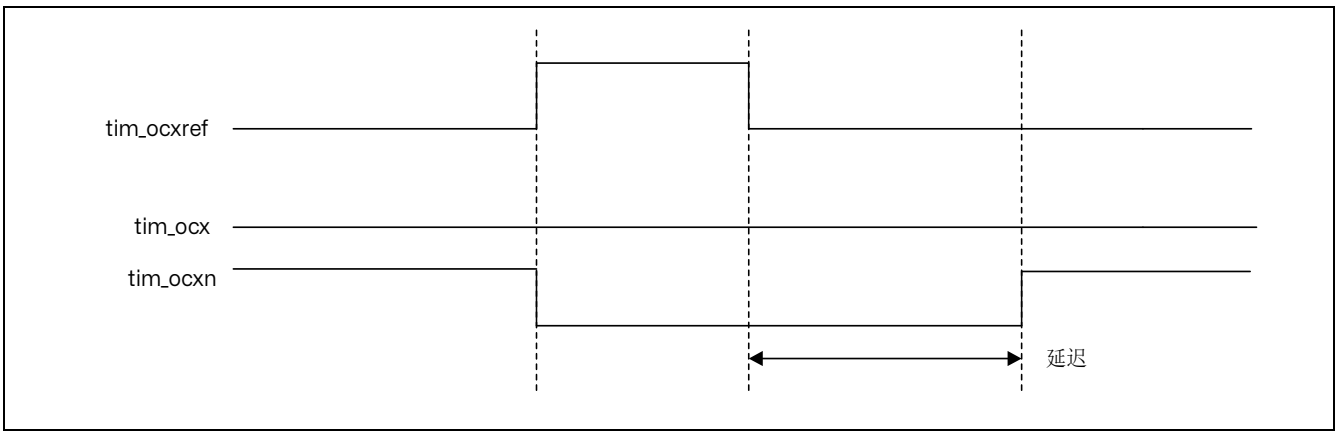



图 22.47 死区波形延迟大于正脉冲

死区延迟对于所有通道均相同，可通过 TIMx_BDTR 寄存器中的 DTG 位进行编程。有关延迟时间计算的信息，请参见后续小节：TIMx 刹车和死区寄存器（TIMx_BDTR）（x=1，8）。

**将 OCxREF 重定向到 OCx 或 OCxN**

在输出模式（强制输出模式、输出比较模式或 PWM 模式）下，通过配置 TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位，可将 tim_ocxref 重定向到 tim_ocx 输出或 tim_ocxn 输出。

通过此功能，可以在一个输出上发送特定波形（如 PWM 或静态有效电平），而同时使互补输出保持其无效电平。或者，使两个输出同时保持无效电平，或者两个输出同时处于有效电平，两者互补并且带死区。

*注：如果仅使能* tim_ocxn (CCxE=0, CCxNE=1)，*两者不互补，一旦* tim_ocxref *为高电平，* tim_ocxn *即变为有效。例如，如果* CCxNP=0，*则* tim_ocxn= tim_ocxref。*另一方面，如果同时使能* tim_ocxn *和* tim_ocxref (CCxE=CCxNE=1)，tim_ocx *在* tim_ocxref *为高电平时变为有效，而* tim_ocxn *则与之互补，在* tim_ocxref *为低电平时变为有效。*

### 22.3.18 使用刹车功能

刹车功能的目的是保护由定时器产生的 PWM 信号驱动的功率开关。两个刹车输入通常连接到功率级和三相逆变器的故障输出。当激活时，刹车电路会关闭 PWM 输出并强制它们进入预定义的安全状态。还可以选择许多内部 MCU 事件来触发输出关闭。

刹车具有两个通道。一个刹车通道可以收集系统级故障（时钟故障、ECC/奇偶校验错误，.....）和应用故障（来自输入引脚和内置比较器），并可以在死区时间后将输出强制到一个预定义的水平（无论是有效还是无效）。一个刹车 2 通道只包括应用故障，并能够将输出强制到无效状态。

刹车期间输出使能信号和输出电平取决于几个控制位：

- TIMx_BDTR 寄存器中的 MOE 位允许通过软件使能/禁用输出，并在发生刹车或刹车 2 事件时重置。
- TIMx_BDTR 寄存器中的 OSSI 位定义了定时器是否在无效状态下控制输出或将其控制权释放给 GPIO 控制器（通常是为了使其处于高阻模式）。
- TIMx_CR2 寄存器中的 OISx 和 OISxN 位设置输出关闭电平，可以是有效电平或无效电平。无论 OISx 和 OISxN 的值如何，tim_ocx 和 tim_ocxn 输出不能同时在给定时间内设置为有效电平。更多详细信息请参考后续表格：具有刹车功能的互补 tim_ocx 和 tim_ocxn 通道的输出控制位。

当退出重置状态时，刹车电路被禁用，并且 MOE 位为低电平。通过设置 TIMx_BDTR 寄存器中的 BKE 和 BK2E 位，可以使能刹车功能。通过配置相同的寄存器中的 BKP 和 BK2P 位，可以选择刹车输入极性。BKEx 和 BKPx 可以同时修改。在写入 BKEx 和 BKPx 位后，需要等待 1 个 APB 时钟周期才能使写入生效。因此，在写入操作后，需要等待 1 个 APB 时钟周期才能正确地读取该位。

由于 MOE 下降沿可能是异步的，因此在实际信号（作用于输出）和同步控制位（在 TIMx_BDTR 寄存器中访问）之间插入了一个重新同步电路。这导致异步信号和同步信号之间存在一些延迟。特别是当 MOE 被设置为 1 而之前它是低电平时，在正确读取之前必须插入一个延迟（伪指令）。这是因为写入作用于异步信号，而读取反映的是同步信号。

刹车（tim_brk）通道的源为：

- 连接到 TIMx_BKIN 引脚的外部源（根据 GPIO 备用功能选择寄存器中的选择而定），具有极性选择和可选的数字滤波
- 内部源：

— 来自 tim_brk_cmpx 输入（请参考章节：TIM1/TIM8 引脚和内部信号）
— 来自 tim_sys_brk 输入的系统中断请求（请参考章节：TIM1/TIM8 引脚和内部信号）

刹车 2（tim_brk2）的源为：

- 连接到 TIMx_BKIN2 引脚的外部源（根据 GPIO 备用功能选择寄存器中的选择而定），具有极性选择和可选的数字滤波。
- 来自 tim_brk2_cmpx 输入的内部源（请参考章节：TIM1/TIM8 引脚和内部信号）

也可以通过软件使用 TIMx_EGR 寄存器中的 BG 和 B2G 位生成刹车事件。

所有源在进入定时器的 tim_brk 或 tim_brk2 输入之前进行 OR 操作，如下图所示。

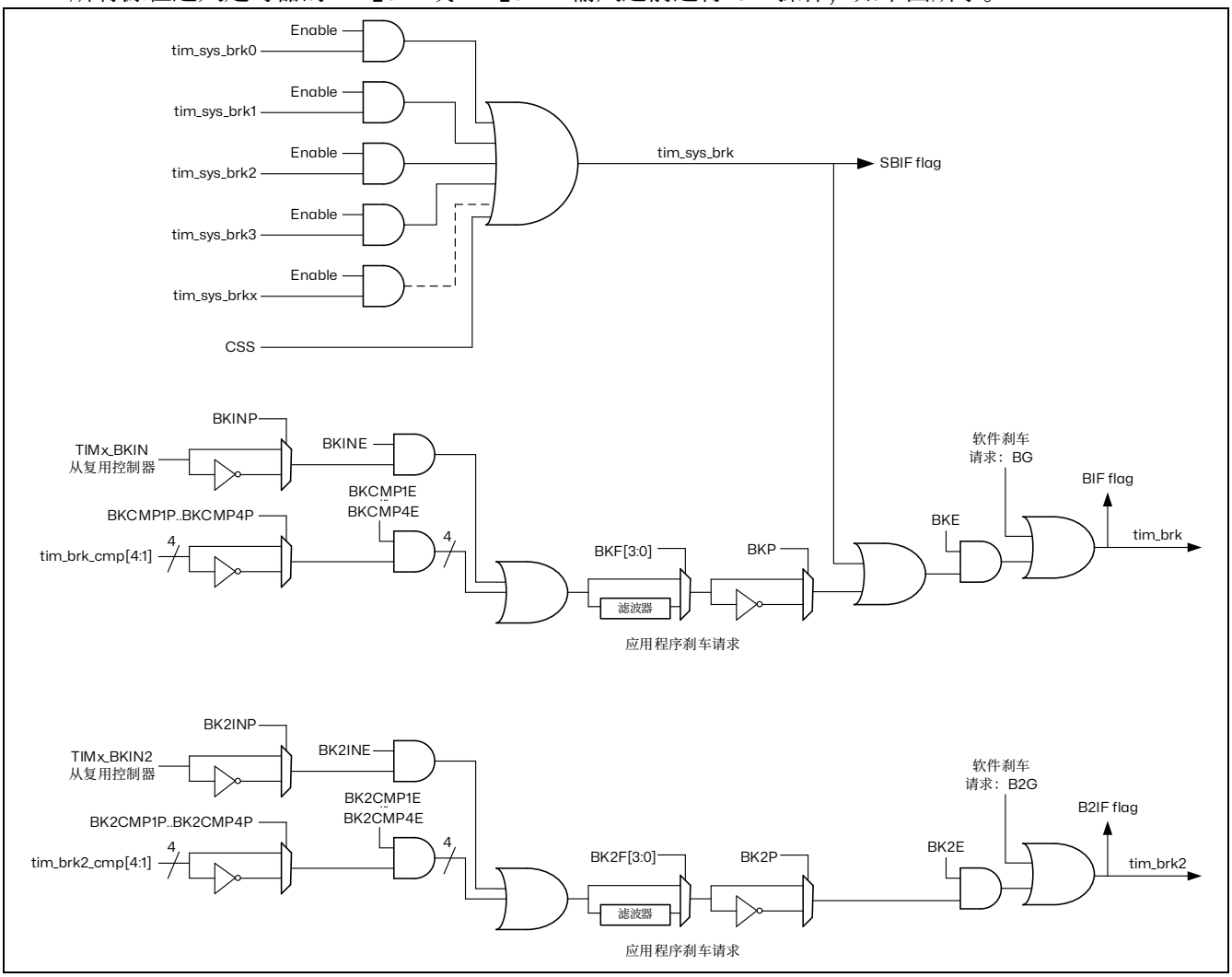



图 22.48 刹车和刹车 2 电路概述

*注：只有当可编程滤波器禁用时，才保证异步（无时钟）操作。如果使能，必须使用故障安全时钟模式（例如使用内部PLL 和/或CSS）来保证刹车事件得到处理。*

当其中一个刹车发生时（在所选的刹车输入线上检测到选定电平）：

- MOE 位会被异步清除，使输出处于无效状态、空闲状态或甚至将控制权释放给 GPIO 控制器（由 OSSI 位选择）。即使 MCU 振荡器关闭，此功能也会使能。
- 一旦 MOE=0，每个输出通道将驱动 TIMx_CR2 寄存器中 OISx 位编程的电平。如果 OSSI=0，定时器将释放输出控制（由 GPIO 控制器接管），否则使能输出保持高电平。
- 当使用互补输出时：

— 输出首先处于无效状态（取决于极性）。这是异步完成的，因此即使没有向定时器提供时钟也可以正常工作。

— 如果定时器的时钟仍然存在，则重新激活死区生成器，以便在死区后驱动 OISx 和 OISxN 位中编程的电平的输出。即使在这种情况下，tim_ocx 和 tim_ocxn 也不能同时驱动到其活动电平。请注意，由于在 MOE 上的重新同步，死区持续时间比平常稍长（大约 2 个 tim_ker_ck 时钟周期）。

— 如果 OSSI=0，定时器将释放输出控制（由 GPIO 控制器接管，该控制器强制高阻态），否则使能输出保持或变为高电平，一旦 CCxE 或 CCxNE 位为高电平。

- 设置刹车标志位（TIMx_SR 寄存器中的 SBIF、BIF 和 B2IF 位）。如果 TIMx_DIER 寄存器中的 BIE 位已设置，则生成中断。如果 TIMx_DIER 寄存器中的 BDE 位已设置，则生成 DMA 请求。
- 如果 TIMx_BDTR 寄存器中的 AOE 位已设置，则在下一个更新事件（UEV）时自动重新设置 MOE 位。例如，这可以用于执行调节。否则，MOE 保持低电平，直到应用程序再次将其设置为“1”。在这种情况下，它可以用于安全，并将中断输入连接到来自功率驱动器、温度传感器或任何安全组件的警报。

*注：如果 MOE 被 CPU 复位，而 AOE 位已设置，则输出处于空闲状态，并强制处于无效电平或高阻态，具体取决于 OSSI 值。如果 MOE 和 AOE 位都被 CPU 复位，则输出处于禁用状态，并驱动在 TIMx_CR2 寄存器中 OISx 位编程的电平。*

*注：刹车输入在电平上有效。因此，当刹车输入处于有效状态时（无论是自动还是通过软件设置），MOE 都无法设置。同时，状态标志 BIF 和 B2IF 不能清除。*

除刹车输入和输出管理外，刹车电路内部还实施了写保护，用以保护应用的安全。通过该功能，用户可冻结多个参数配置（死区持续时间、tim_ocx/tim_ocxn 极性和禁止时的状态、OCxM 配置、刹车使能和极性）。可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从 3 种保护级别中进行选择。请参考章节：TIMx 刹车和死区寄存器（TIMx_BDTR）(x=1，8)。MCU 复位后只能对 LOCK 位执行一次写操作。

下图所示为输出对刹车响应行为的示例。

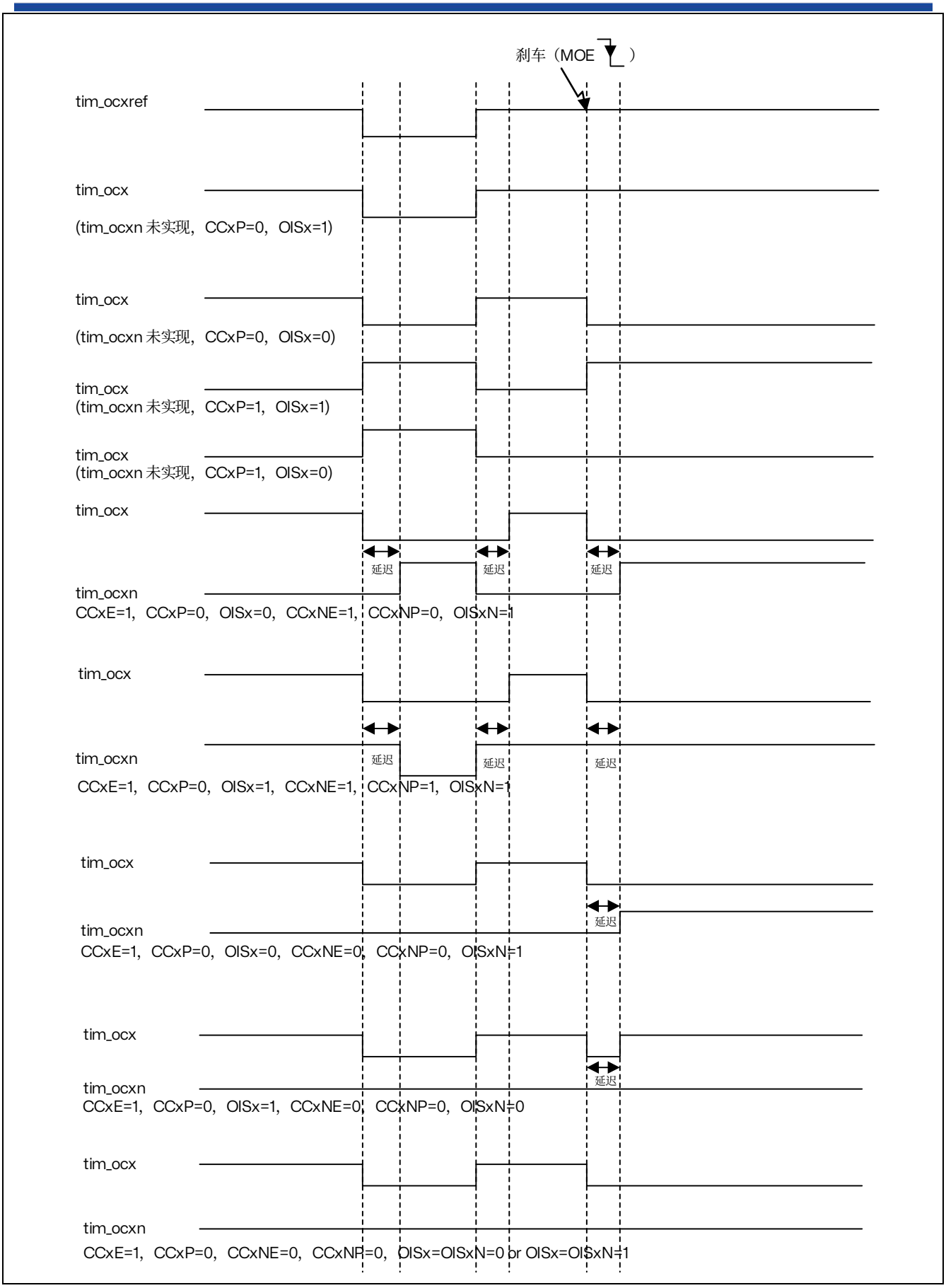



图 22.49 tim_brk 在发生刹车事件时的各种输出行为（OSSI=1）

两个中断输入在计时器输出上具有相同的行为：

— tim_brk/ tim_brk2 输入可禁止（无效状态）PWM 输出，也可将 PWM 输出强制为预定义的安全状态。

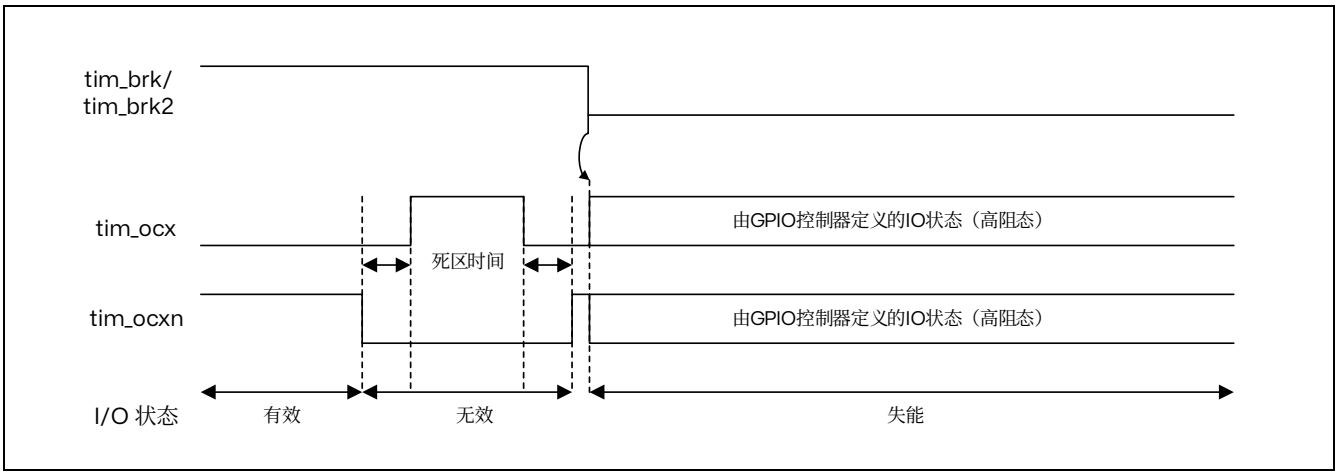



图 22.50 当 tim_brk/tim_brk2 有效（OSSI=0）时 PWM 输出状态

### 22.3.19 双向刹车输入

TIM1/TIM8 具有双向刹车 I/O，如下图所示。

这为以下方面提供支持：

- 板级全局刹车信号，用于向外部 MCU 或门驱动器发送故障信号，使用唯一的引脚作为输入和输出状态引脚
- 内部刹车源和多个外部开漏源 OR 连接在一起，以触发唯一的刹车事件，当多个内部和外部刹车源必须合并时

使用 TIMxBDTR 寄存器中的 BKBID 和 BK2BID 位配置 tim_brk 和 tim_brk2 输入为双向模式。BKBID 编程位可以使用 TIMxBDTR 寄存器中的 LOCK 位锁定为只读模式（在 LOCK 级别 1 或更高）。

tim_brk 和 tim_brk2 输入都支持双向模式，并且要求 I/O 配置为开漏模式，使用低电平有效极性（使用 BKINP、BKP、BK2INP 和 BK2P 位）。来自系统（例如 CSS）、片上外设或刹车输入的任何刹车请求都会强制将刹车输入置于低电平，以指示故障事件。出于安全原因，如果极性位没有正确设置（高电平有效），则双向模式将被抑制。

刹车软件事件（BG 和 B2G）也会强制将刹车 I/O 置为“0”，以指示外部组件定时器已进入刹车状态。然而，只有使能了刹车（BKE 或 B2KE = 1）时，这才有效。当使用 BKE 或 B2KE = 0 生成软件刹车事件时，输出将处于安全状态，并且会设置刹车标志，但 TIMx_BKIN 和 TIMx_BKIN2 I/O 没有影响。

安全解发机制可防止系统被彻底锁定（刹车输入低电平触发刹车，强制该输入为低电平）。

当 BKDSRM（BK2DSRM）位设置为 1 时，这将释放刹车输出以清除故障信号并允许重新给系统装载。

在任何时候都不能禁用刹车保护电路：

- 刹车输入路径始终处于有效状态：即使设置了 BKDSRM（BK2DSRM）位并释放了开漏控制，刹车事件也是有效的。这可以防止在存在刹车条件时重新启动 PWM 输出。
- 只要输出使能（MOE 位设置），BKDSRM（BK2DSRM）位就不能解除刹车保护（参见下表）。

表 22.15 刹车保护解除条件

| MOE | BKBID (BK2BID) | BKDSRM (BK2DSRM) | 刹车保护状态 |
|---|---|---|---|
| 0 | 0 | X | 装载 |
| 0 | 1 | 0 | 装载 |
| 0 | 1 | 1 | 解除 |
| 1 | X | X | 装载 |

**装载和重装载刹车电路**

默认情况下（外设复位配置），刹车电路（输入或双向模式）处于装载状态。
在发生刹车（刹车 2）事件后，必须按照以下步骤重新装载保护：

- BKDSRM（BK2DSRM）位必须设置为释放输出控制
- 软件必须等待系统刹车条件消失（如果有的话），并清除 SBIF 状态标志（或在重新装载之前系统地清除它）
- 软件必须轮询 BKDSRM（BK2DSRM）位，直到硬件清除该位（当应用程序刹车条件消失时）

从这一点开始，刹车电路处于装载和有效状态，并且可以设置 MOE 位以重新使能 PWM 输出。

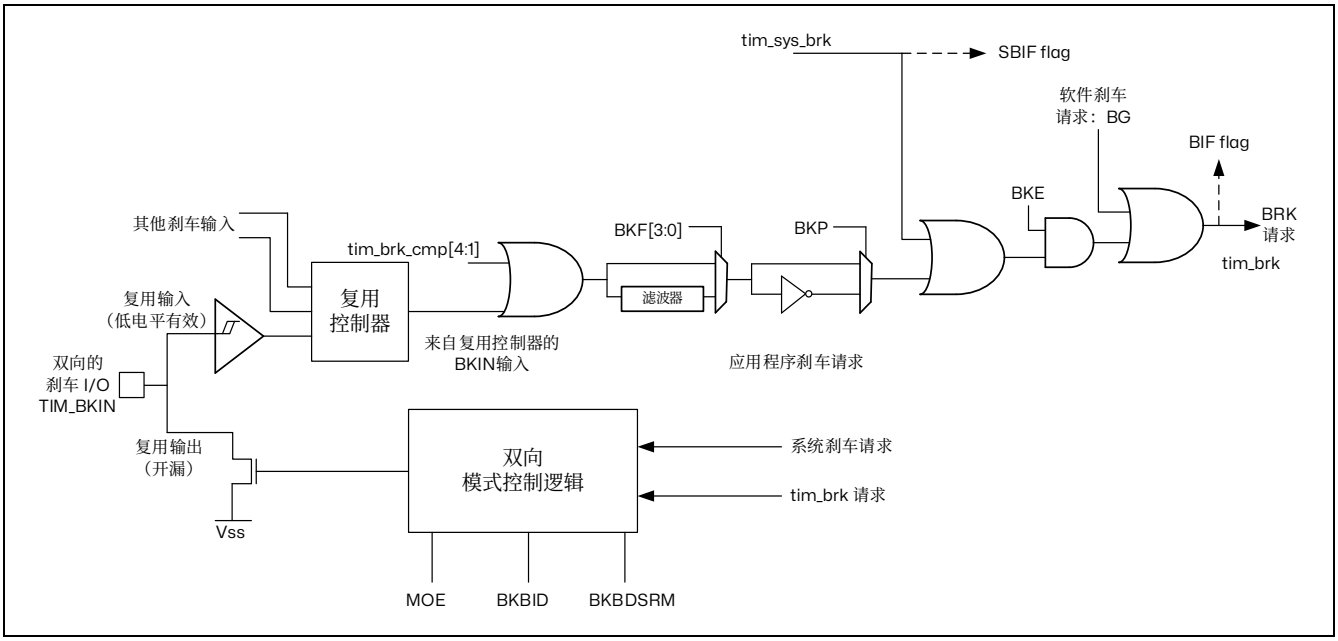



图 22.51 输出重定向（未表示 tim_brk2 请求）

### 22.3.20 在外部事件时清除 tim_ocxref 信号

对于给定通道，在 ETRF 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 OCxREF 信号变为低电平。OCxREF 信号将保持低电平，直到发生下一更新事件（UEV）（特别注意，当 TIMx_RCR>0 时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。

此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。可以通过配置 TIMx_SMCR 寄存器中的 OCCS 位，将 tim_ocref_clr_int 输入选择为 tim_ocref_clr 输入和 tim_etrf（滤波后的 tim_etr_in）之间。

tim_ocref_clr 输入可以在多个输入中进行选择，使用 TIMx_AF2 寄存器中的 OCRSEL[2:0]位域，如下图所示。请参考章节：TIM1/TIM8 引脚和内部信号，以获取产品中可用源的列表。

当配置组合 PWM 模式和三相组合 PWM 模式，对于给定通道，在 tim_ocref_clr_int 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使组合结果 tim_ocxrefc 信号变为低电平。tim_ocxrefc 信号将保持低电平，直到发生下一更新事件（UEV）（特别注意，当 TIMx_RCR>0 时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。

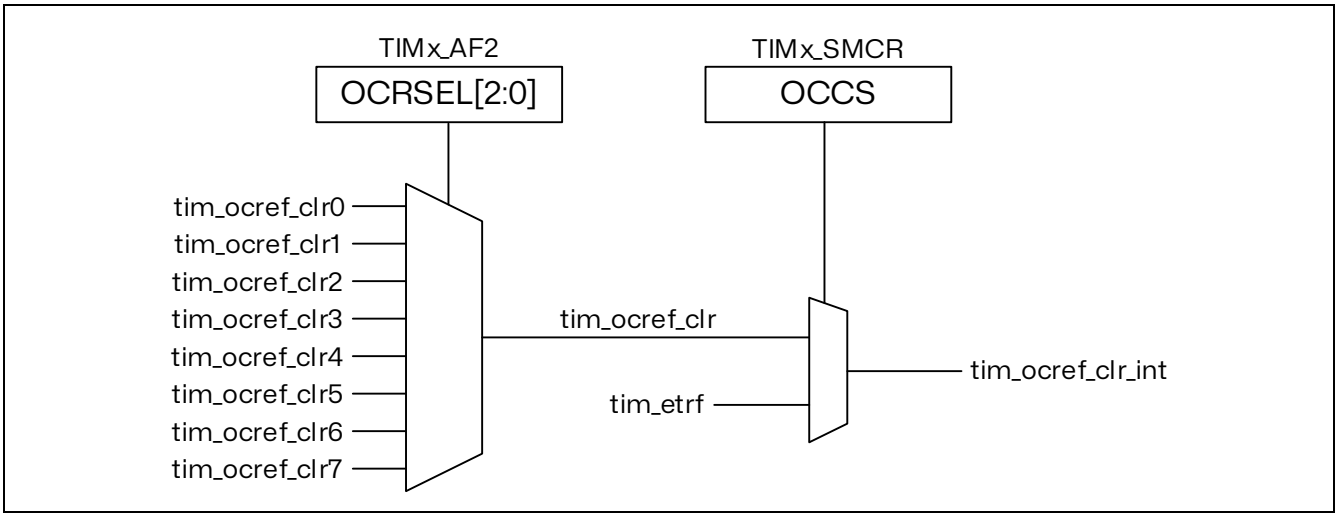



图 22.52 tim_ocref_clr 输入选择复用器

当 tim_etrf 被选择。tim_etr_in 必须如下配置：

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0]位清除为 00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位清除为 0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可根据应用需要进行配置。（根据连接到触发的源的极性以及使用滤波器去除噪声的最终需求）。

下图对比了使能位 OCxCE 在不同值下的情况，显示了当 tim_etrf 输入变为高电平时 tim_etrf 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

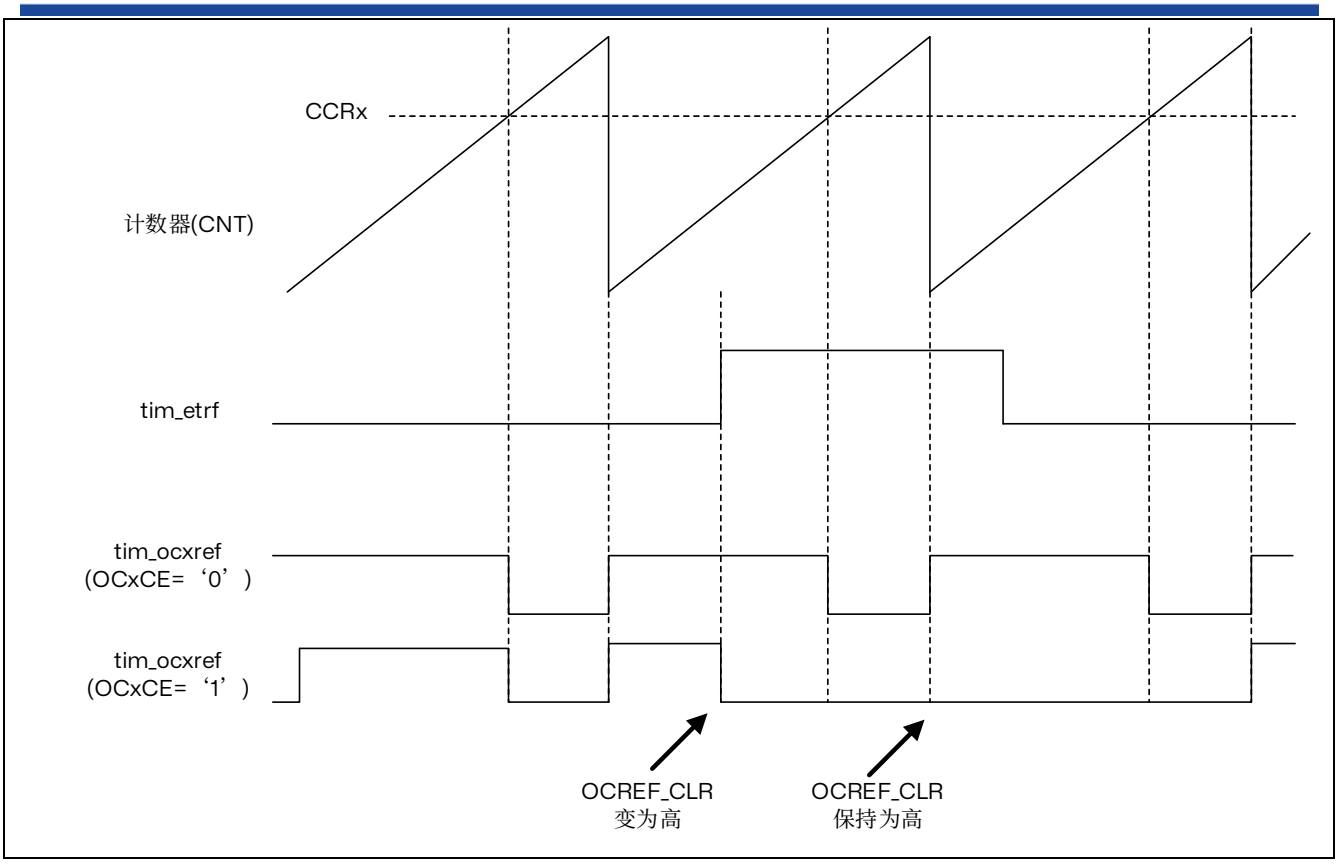



图 22.53 清除 TIMx 的 tim_ocxref

注意：如果 PWM 的占空比为 100% (CCRx>ARR)，则下次计数器溢出时会再次使能 tim_ocxref。

### 22.3.21 生成 6 步 PWM

当通道使用互补输出时，OCxM、CCxE 和 CCxNE 位上提供预装载位。发生 COM 换向事件时，这些预装载位将传输到影子位。因此，用户可以预先编程下一步骤的配置，并同时更改所有通道的配置。COM 可由软件通过将 TIMx_EGR 寄存器中的 COM 位置 1 而生成，也可以由硬件在 tim_trgi 上升沿生成。

发生 COM 事件时，某个标志位（TIMx_SR 寄存器中的 COMIF 位）将会置 1。这时将产生中断（如果 TIMx_DIER 寄存器中的 COMIE 位置 1）或 DMA 请求（如果 TIMx_DIER 寄存器中的 COMDE 位置 1）。

下图以 3 种不同的编程配置为例，显示了发生 COM 事件时 tim_ocx 和 tim_ocxn 输出的行为。

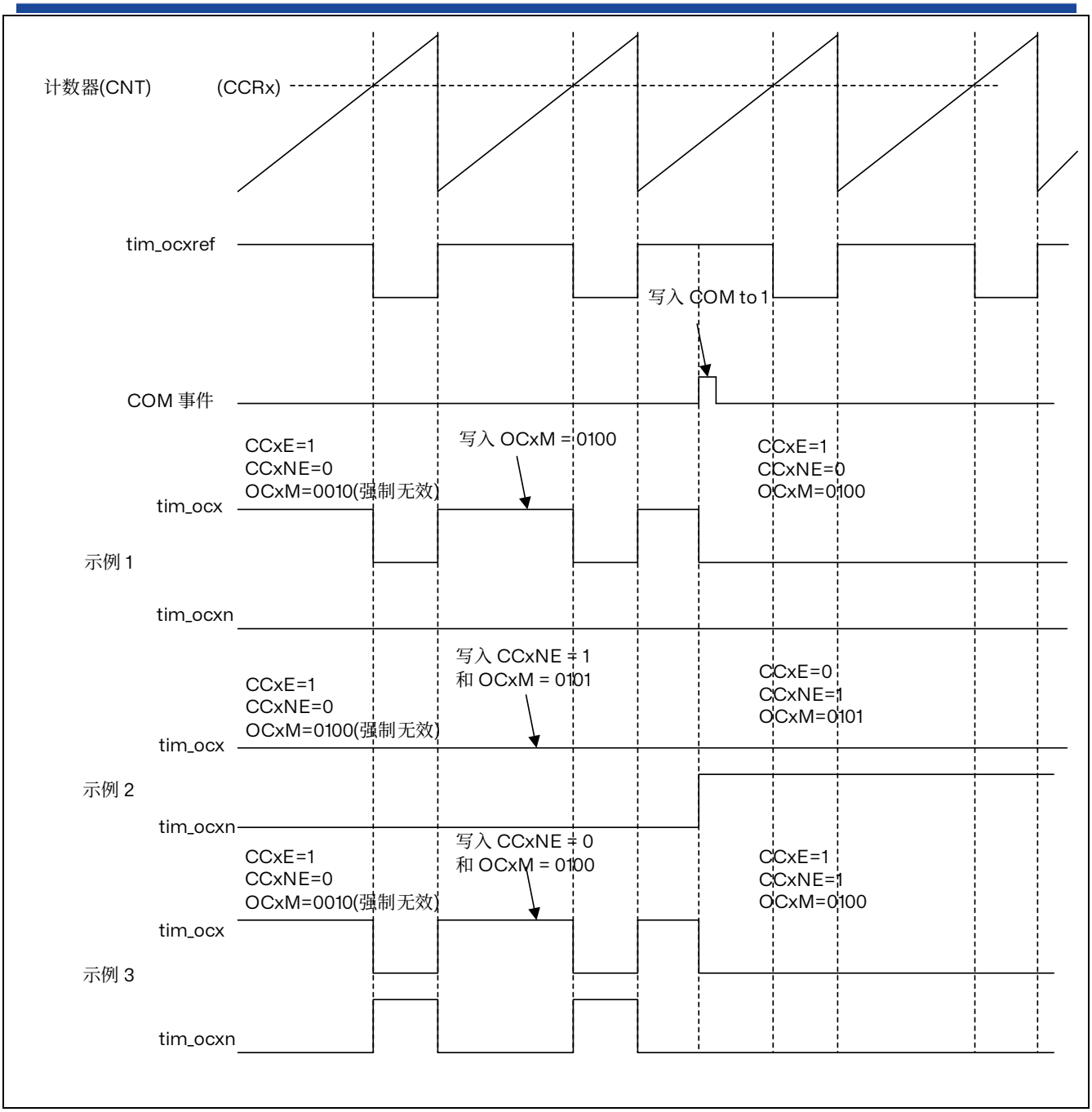



图 22.54 COM 事件生成 6 步 PWM 的示例（OSSR=1）

### 22.3.22 单脉冲模式

单脉冲模式（OPM）是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- 递增计数模式下：CNT < CCRx ≤ ARR（特别注意，0<CCRx）
- 递减计数模式下：CNT > CCRx

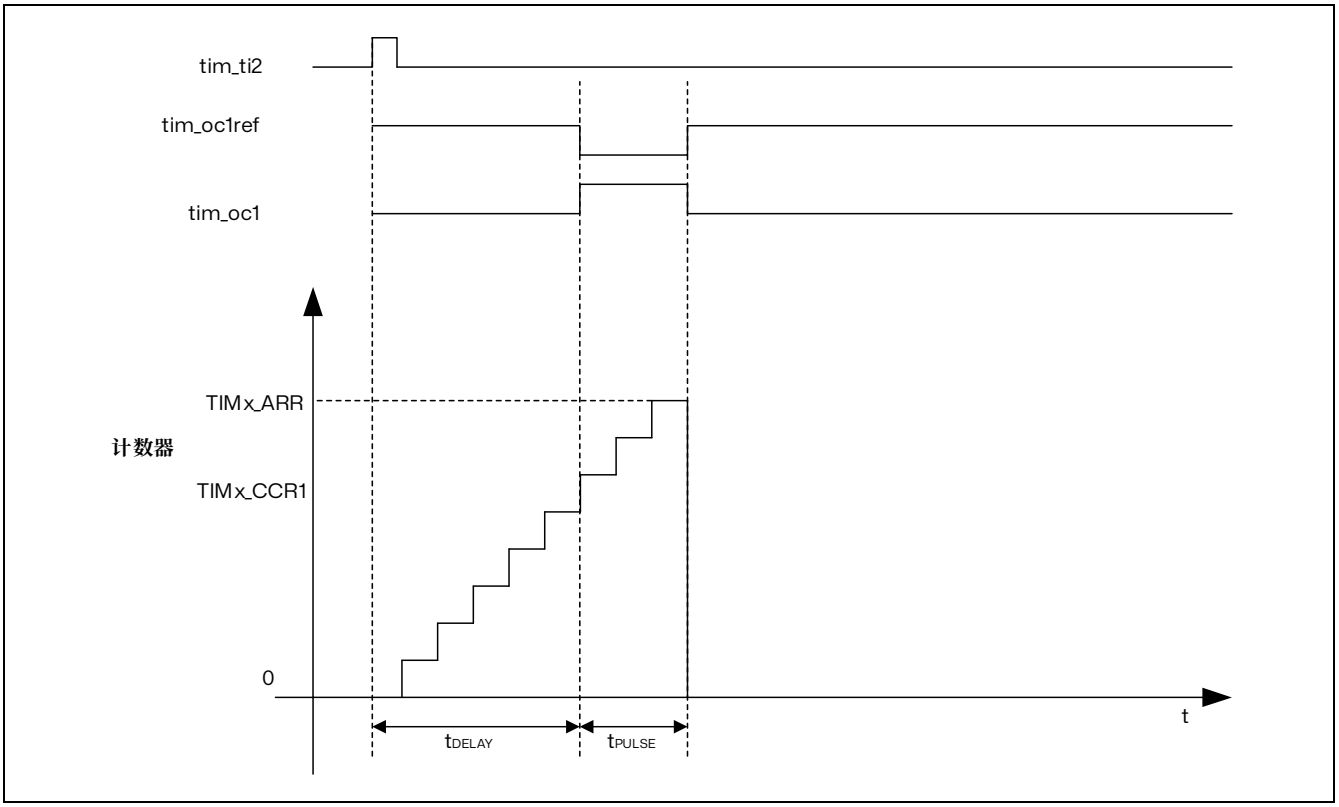



图 22.55 单脉冲模式的例子

例如，用户希望达到这样的效果：在 tim_ti2 输入引脚检测到正沿时，经过 $t_{DELAY}$ 的延迟，在 tim_oc1 上产生一个长度为 $t_{PULSE}$ 的正脉冲。

使用 tim_ti2fp2 作为触发 1:

1. 在 TIMx_CCMR1 寄存器中写入 CC2S=“01”，将 tim_ti2fp2 映射到 tim_ti2。
2. 在 TIMx_CCER 寄存器中写入 CC2P=“0”和 CC2NP=“0”，使 tim_ti2fp2 能够检测上升沿。
3. 在 TIMx_SMCR 寄存器中写入 TS=00110，将 tim_ti2fp2 配置为从模式控制器的触发 (tim_trgi)。
4. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），使用 tim_ti2fp2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- $t_{DELAY}$ 由写入 TIMx_CCR1 寄存器的值定义。
- $t_{PULSE}$ 由自动重载值与比较值（TIMx_ARR - TIMx_CCR1）之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值

时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 tim_ti2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位清“0”时，即选择重复模式。

**特例：tim_ocx 快速使能：**

在单脉冲模式下，tim_tix 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（$t_{DELAY}$ 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 tim_ocxref（和 tim_ocx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用

### 22.3.23 可重触发单脉冲模式

此模式允许计数器在接收到刺激后开始计数，并生成一个具有可编程长度的脉冲，但与上一节中描述的非重触发单脉冲模式存在以下差异：

— 只要触发器发生，脉冲就会立即开始（没有可编程的延迟）

— 如果在先前触发的脉冲完成之前发生新的触发，则脉冲会被延长

定时器必须处于从模式，TIMx_SMCR 寄存器的 SMS[3:0]位设置为“1000”（组合重置+触发模式），OCxM[3:0]位设置为“1000”或“1001”，用于可重触发 OPM 模式 1 或 2。

如果定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）。如果定时器配置为递减计数模式，CCRx 必须大于或等于 ARR。

*注：OCxM[[3:0]和SMS[3:0]位域由于兼容性的原因被分成两个部分，最高有效位与最低3 位不是连续的。OCxM 这种模式不能与中央对齐PWM 模式一起使用。TIMx_CR1 寄存器中的CMS[1:0]必须设置为00。*

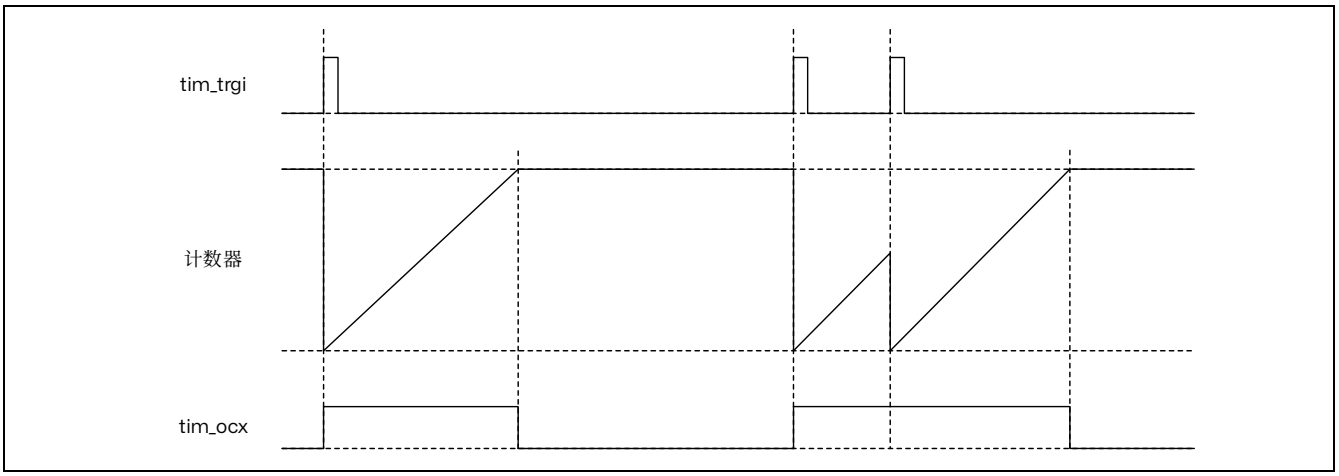



图 22.56 可重触发单脉冲模式

### 22.3.24 在比较模式下的脉冲

在比较匹配事件发生时，可以生成一个脉冲。当计数器的值等于给定的比较值时，可以生成一个具有可编程脉冲宽度的信号，用于调试或同步。

此模式适用于任何从机模式选择，包括边沿对齐和中心对齐计数模式。它仅适用于通道 3 和通道 4。脉冲发生器是唯一的，由两个通道共享，如下图所示。

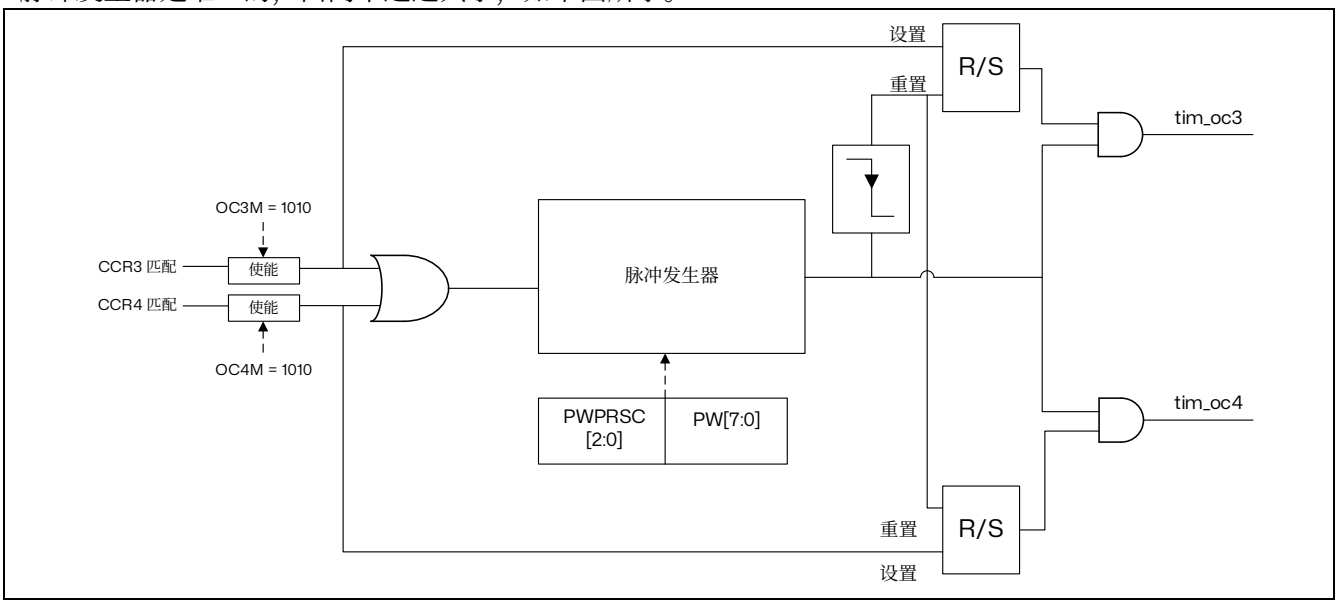



图 22.57 脉冲发生器电路

下图展示了如何在边沿对齐模式下生成脉冲。

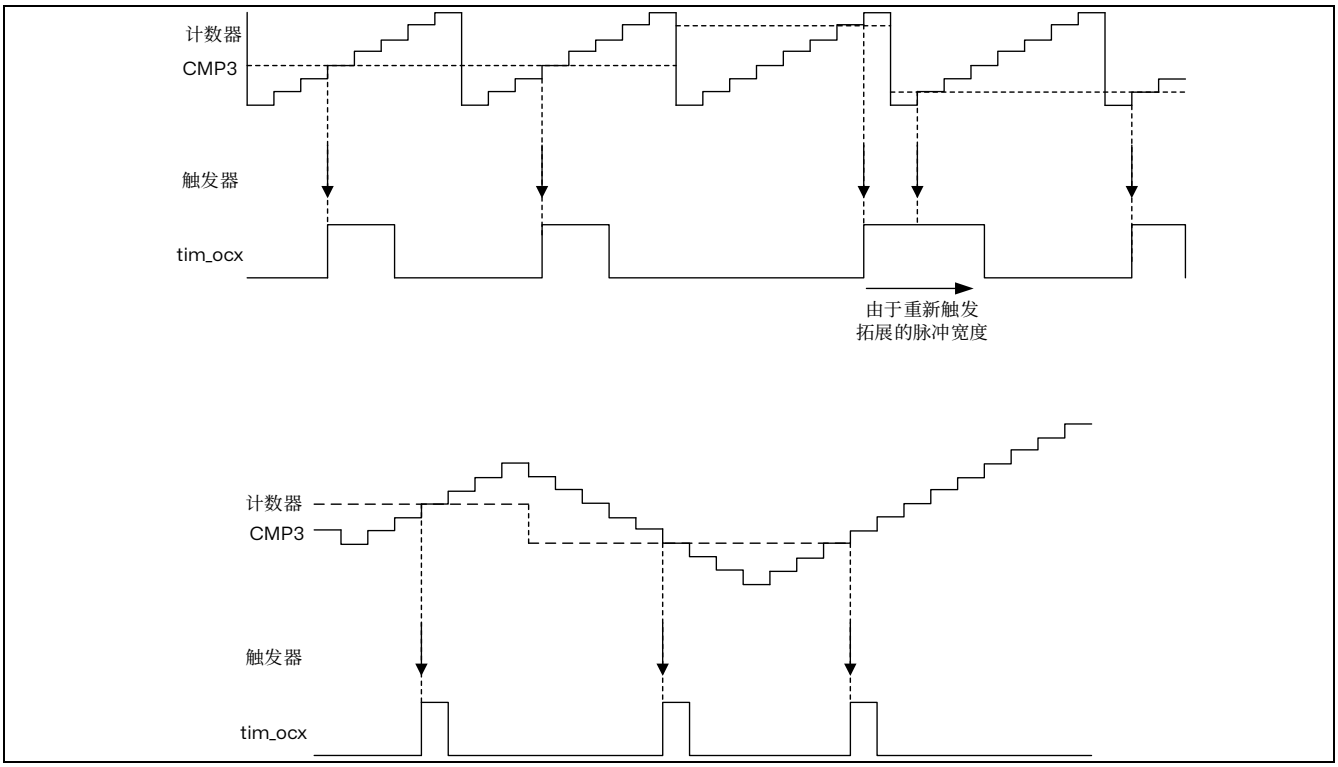



图 22.58 在比较事件上的脉冲生成，适用于边沿对齐模式

此输出比较模式是通过 TIMx_CCMR2 寄存器中的 OC3M[3:0]和 OC4M[3:0]位域进行选择的。

脉冲宽度通过寄存器中的 PW[7:0]位域进行编程，使用特定的时钟，并根据 PWPRSC[2:0]位的预设进行预分频，具体如下：

$t_{PW} = PW[7:0] \times t_{PWG}$

其中 $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

给出了分辨率和最大值，具体取决于预分频器的值。

脉冲可以重新触发：在脉冲正在进行时触发新的脉冲，会导致脉冲被延长。

*注：如果同时使能两个通道，只要一个通道上的触发器不与并发输出上生成的脉冲重叠，脉冲就会独立发出。相反，如果两个触发器重叠，与第一个到达的触发器相关的脉冲宽度会被延长（由于重新触发），而最后一个到达的触发器的脉冲宽度是正确的（如下图所示）。*

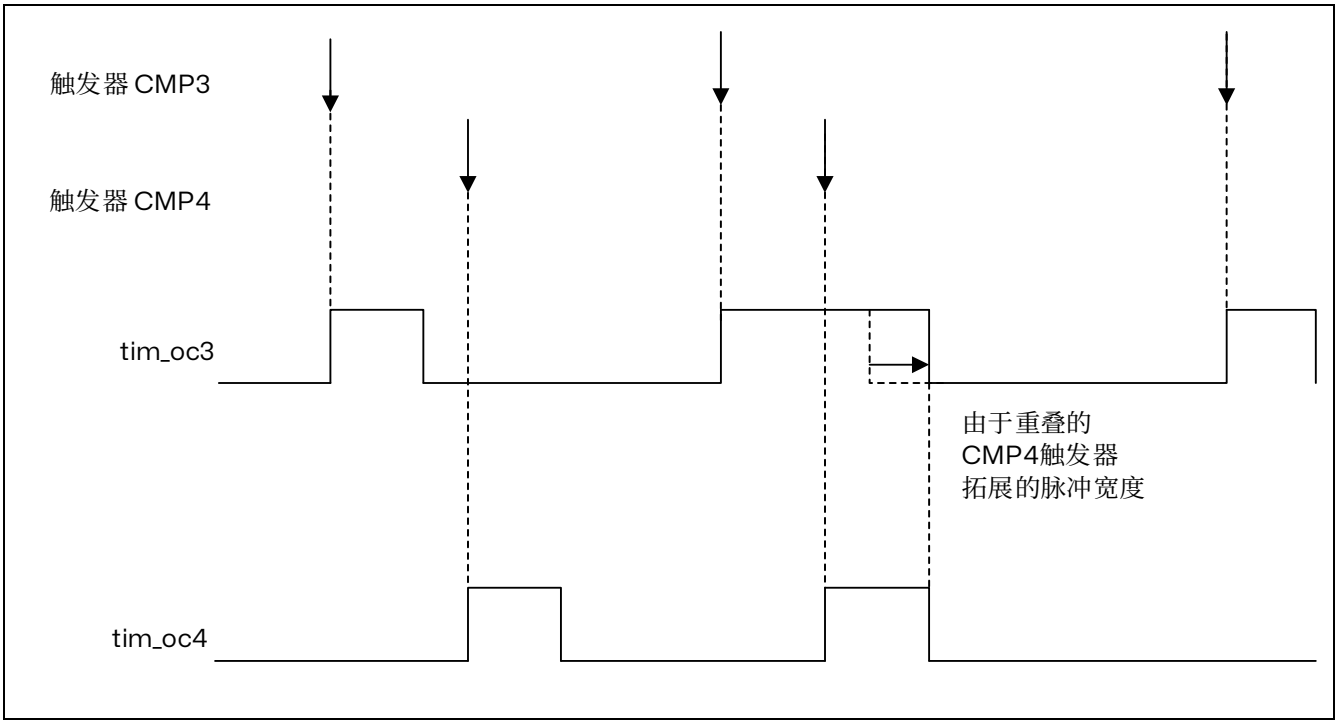



图 22.59 在同时触发的情况下延长脉冲宽度

### 22.3.25 编码器接口模式

#### 正交编码器

选择编码器接口模式时，如果计数器仅在 tim_ti2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=0001；如果计数器仅在 tim_ti1 边沿处计数，写入 SMS=0010；如果计数器在 tim_ti1 和 tim_ti2 边沿处均计数，则写入 SMS=0011。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 tim_ti1 和 tim_ti2 极性。如果需要，还可对输入滤波器进行编程。CC1NP 和 CC2NP 必须保持为低电平。

tim_ti1 和 tim_ti2 两个输入用于连接增量编码器。请参见下表。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 tim_ti1fp1 或 tim_ti2fp2 上的每次有效信号转换提供。tim_ti1fp1 和 tim_ti2fp2 是进行输入滤波器和极性选择后 tim_ti1 和 tim_ti2 的信号，如果不进行滤波和反

相，则 tim_ti1fp1=tim_ti1，tim_ti2fp2=tim_ti2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（tim_ti1 或 tim_ti2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 tim_ti1 或 tim_ti2 边沿处计数，还是同时在 tim_ti1 和 tim_ti2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、触发输出功能继续正常工作。编码器模式和外部时钟模式 2 不兼容，因此 不能同时选择。

在此模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 tim_ti1 和 tim_ti2 不同时切换）。

表 22.16 计数方向与编码器信号的关系（CC1P = CC2P = 0）

| 有效边沿 | SMS[3:0] | 相对信号的电平（tim_ti1fp1 对应 tim_ti2, tim_ti2fp2 对应 tim_ti1） | tim_ti1fp1 信号 | | tim_ti2fp2 信号 | |
|---|---|---|---|---|---|---|
| | | | 上升 | 下降 | 上升 | 下降 |
| 仅 x1 模式在 tim_ti1 计数 | 1110 | 高 | 递减 | 递增 | 不计数 | 不计数 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| 仅 x1 模式在 tim_ti2 计数 | 1111 | 高 | 不计数 | 不计数 | 递增 | 递减 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| 仅 x2 模式在 tim_ti1 计数 | 0001 | 高 | 递减 | 递增 | 不计数 | 不计数 |
| | | 低 | 递增 | 递减 | 不计数 | 不计数 |
| 仅 x2 模式在 tim_ti2 计数 | 0010 | 高 | 不计数 | 不计数 | 递增 | 递减 |
| | | 低 | 不计数 | 不计数 | 递减 | 递增 |
| x4 模式在 tim_ti1 和 tim_ti2 计数 | 0011 | 高 | 递减 | 递增 | 递增 | 递减 |
| | | 低 | 递增 | 递减 | 递减 | 递增 |

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

下图以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S=“01”（TIMx_CCMR1 寄存器，tim_ti1fp1 映射到 tim_ti1 上）。
- CC2S=“01”（TIMx_CCMR2 寄存器，tim_ti2fp2 映射到 tim_ti2 上）。
- CC1P=“0”，CC1NP=“0”（TIMx_CCER 寄存器，tim_ti1fp1 未反相，tim_ti1fp1= tim_ti1）。
- CC2P=“0”，CC2NP=“0”（TIMx_CCER 寄存器，tim_ti2fp2 未反相，tim_ti2fp2= tim_ti2）。
- SMS= “011”（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿都处于有效状态）。
- CEN=“1”（TIMx_CR1 寄存器，使能计数器）。

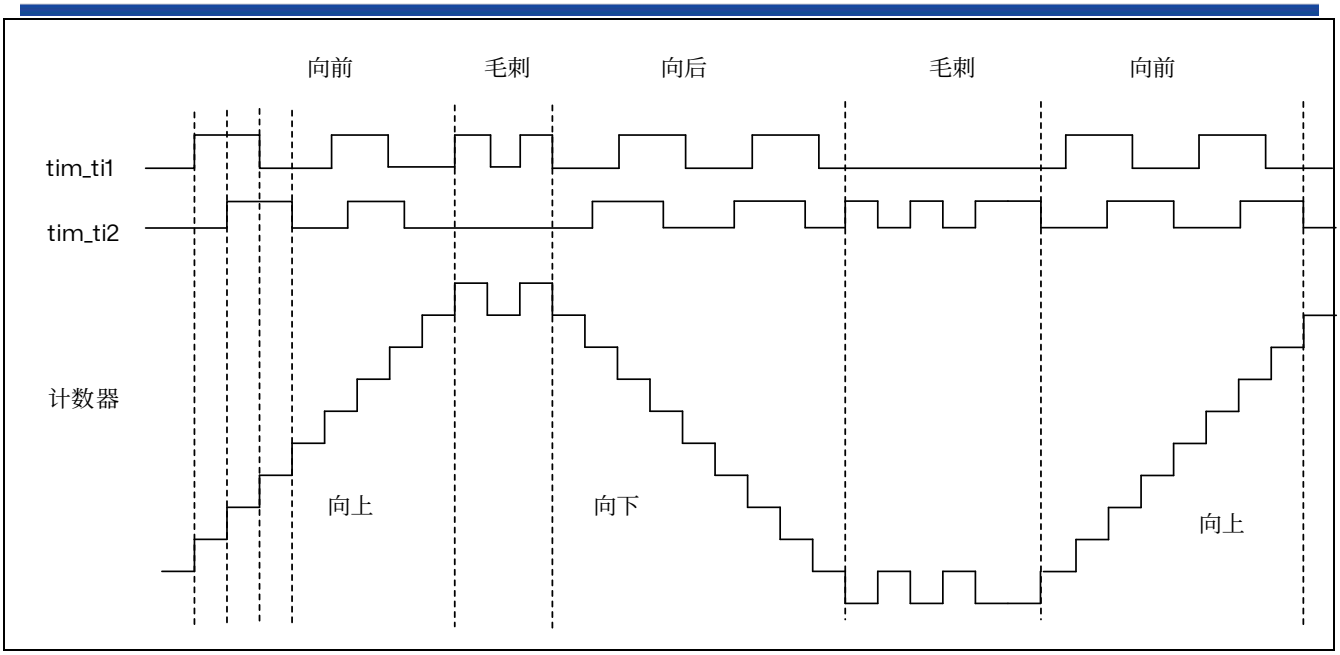



图 22.60 编码器接口模式下的计数器工作示例

下图举例说明 tim_ti1fp1 极性反相时计数器的行为（除 CC1P=“1”外，其它配置与上例相同）。

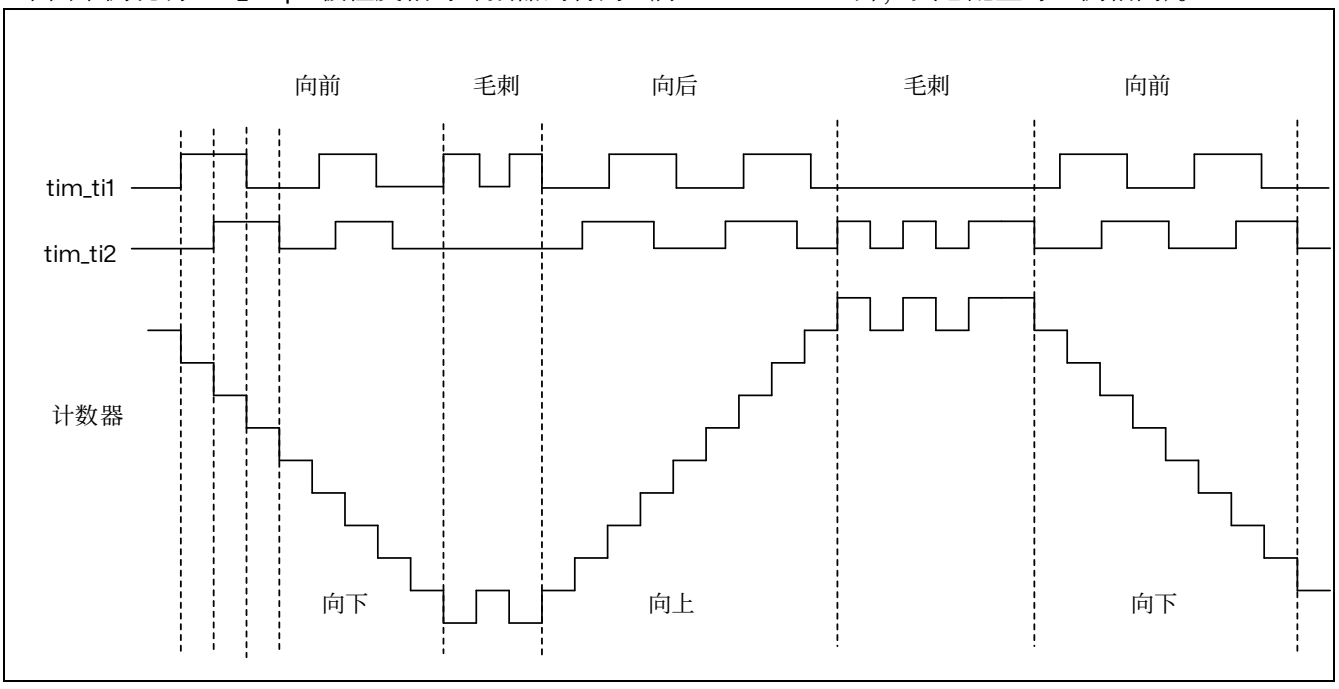



图 22.61 tim_ti1fp1 极性反相时的编码器接口模式示例

下图显示了各种计数模式下速度反转期间计时器计数器的值。

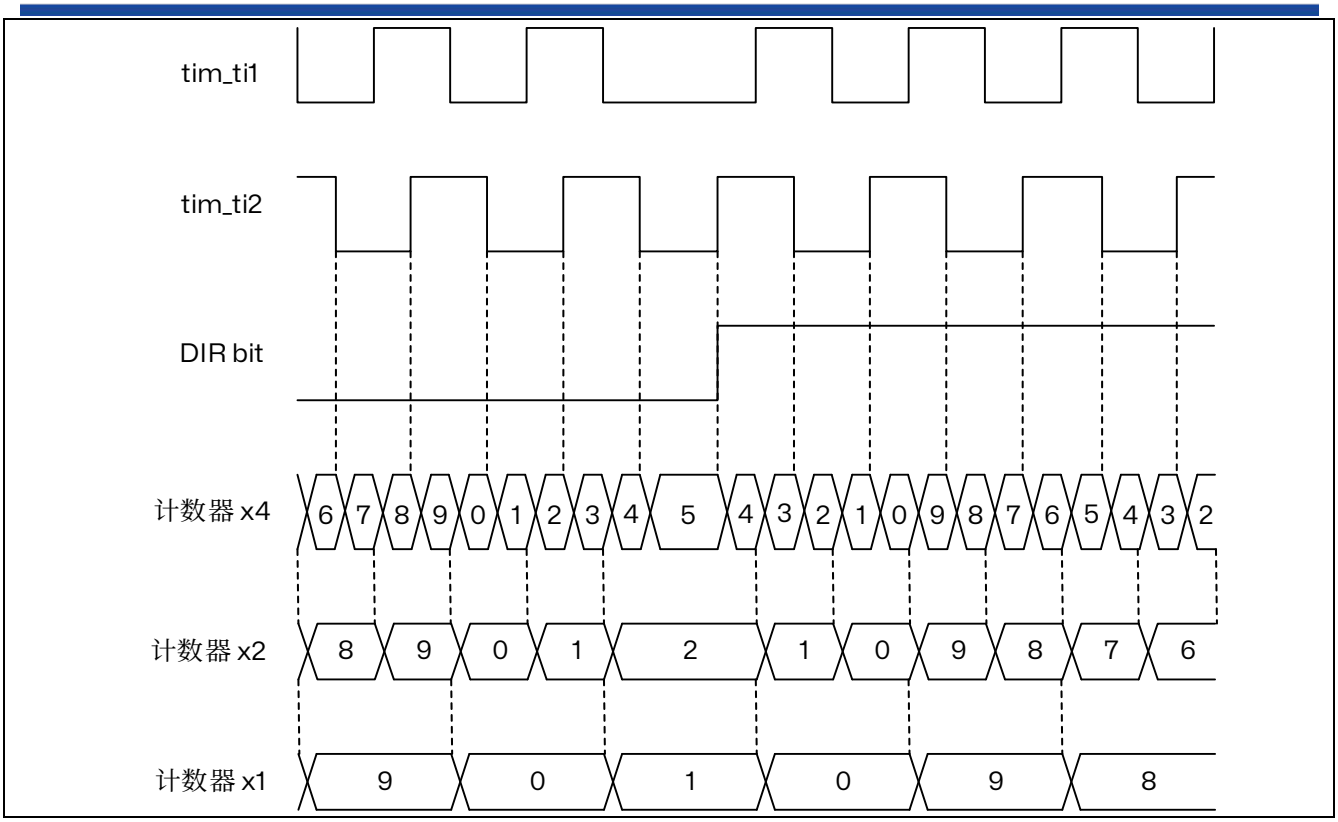



图 22.62 正交编码器的计数模式

当定时器配置为编码器接口模式时，可以提供传感器当前位置的信息。通过测量两个编码器事件之间的周期，使用配置为捕获模式的第二个定时器，可以获得动态信息（速度、加速度、减速度）。指示机械零的编码器输出可用于此目的。根据两个事件之间的时间，还可以定期读取计数器的值。如果可用，可以通过将计数器的值锁存到第三个输入捕获寄存器来实现（此时捕获信号必须是周期性的，并且可以由另一个定时器生成）。当可用时，也可以通过 DMA 请求读取其值。

TIMx_CR1 寄存器中的 IUFREMAP 位将更新中断标志（UIF）的连续副本强制复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许原子地读取计数器的值和由 UIFCPY 标志触发的潜在的翻转条件。这可以通过避免在后台任务（计数器读取）和中断（更新中断）之间共享处理来简化角速度的计算。

UIF 和 UIFCPY 标志之间没有延迟。

在 32 位定时器实现中，当 IUFREMAP 位被设置时，在读取访问时，计数器的第 31 位将被 UIFCPY 标志覆盖（计数器最高有效位仅可在写入模式下访问）。

### 时钟加方向编码器模式

除了正交编码器模式外，定时器还支持其他类型的编码器。

在下图中显示的“时钟加方向”模式下，时钟在 tim_ti2 上提供单线，而方向使用 tim_ti1 输入强制。

此模式通过 TIMx_SMCR 寄存器中的 SMS[3:0]位域使能，如下所示：

- 1010：x2 模式，计数器在时钟的上升沿和下降沿都更新
- 1011：x1 模式，计数器在单个时钟边沿更新，根据 CC2P 位的值而定：CC2P=0 对应上升沿敏感，CC2P=1 对应下降沿敏感。

tim_ti1 上的方向信号极性通过 CC1P 位设置：0 对应正极性（当 tim_ti1 为高电平时向上计数，当 tim_ti1 为低电平时向下计数），CC1P=1 对应负极性（当 tim_ti1 为低电平时向上计数）。

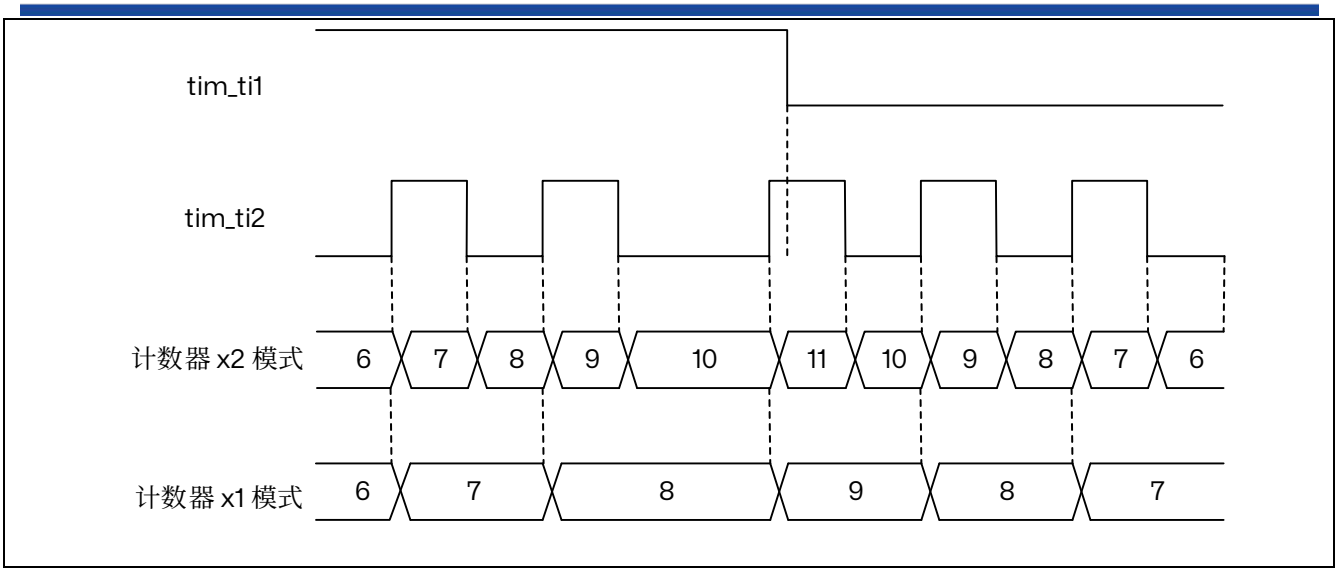



图 22.63 方向加时钟编码器模式

### 方向时钟编码器模式

在下图中的“方向时钟”模式下，时钟在两条线上提供，根据方向一次使用一条，以便有一条向上计数时钟线和一条向下计数时钟线。

此模式通过 TIMx_SMCR 寄存器中的 SMS[3:0]位域使能，如下所示：

- 1100：x2 模式，计数器在两条时钟线的上升沿和下降沿都更新。CC1P 和 CC2P 位编码时钟空闲状态。CCxP=0 对应高电平空闲状态（参考下图），CCxP=1 对应低电平空闲状态（参考下图）。
- 1101：x1 模式，计数器在单个时钟边沿更新，根据 CC1P 和 CC2P 位的值而定。CCxP=0 对应下降沿敏感性和高电平空闲状态（参考下图），CCxP=1 对应上升沿敏感性和低电平空闲状态（参考下图）。

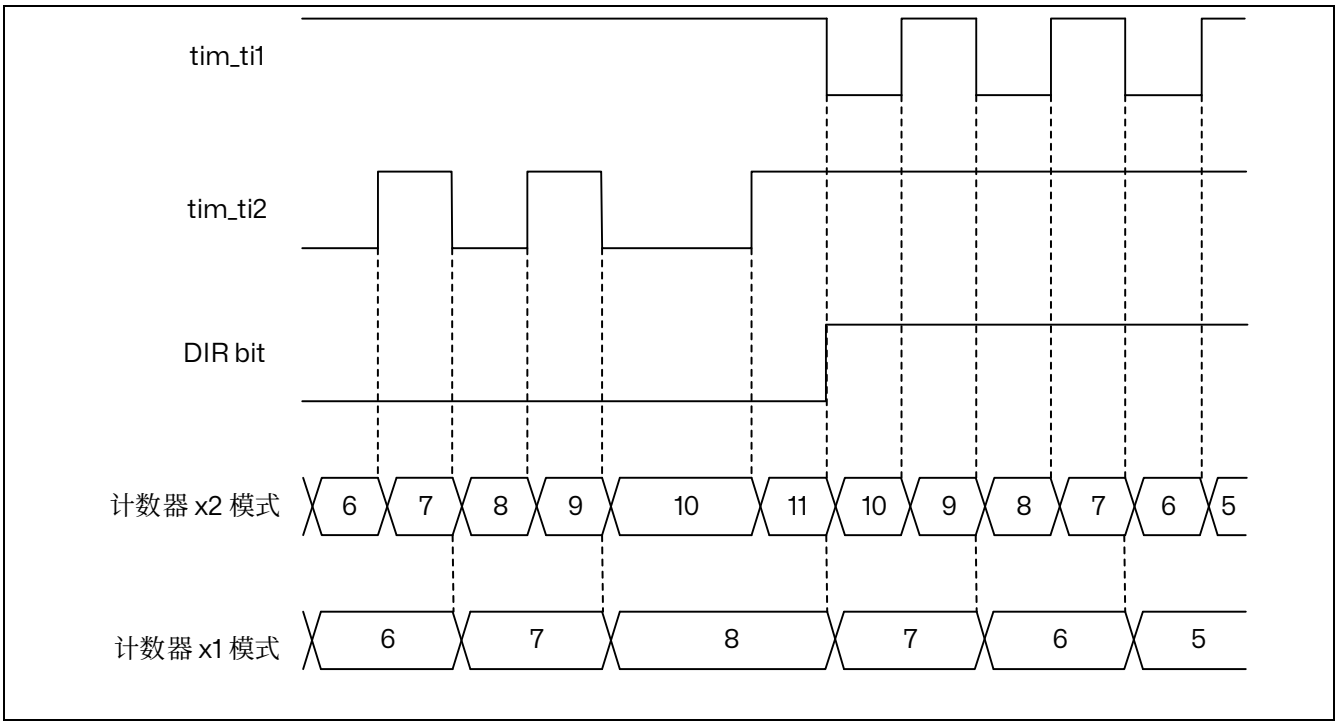



图 22.64 方向时钟编码器模式（CC1P=CC2P=0）

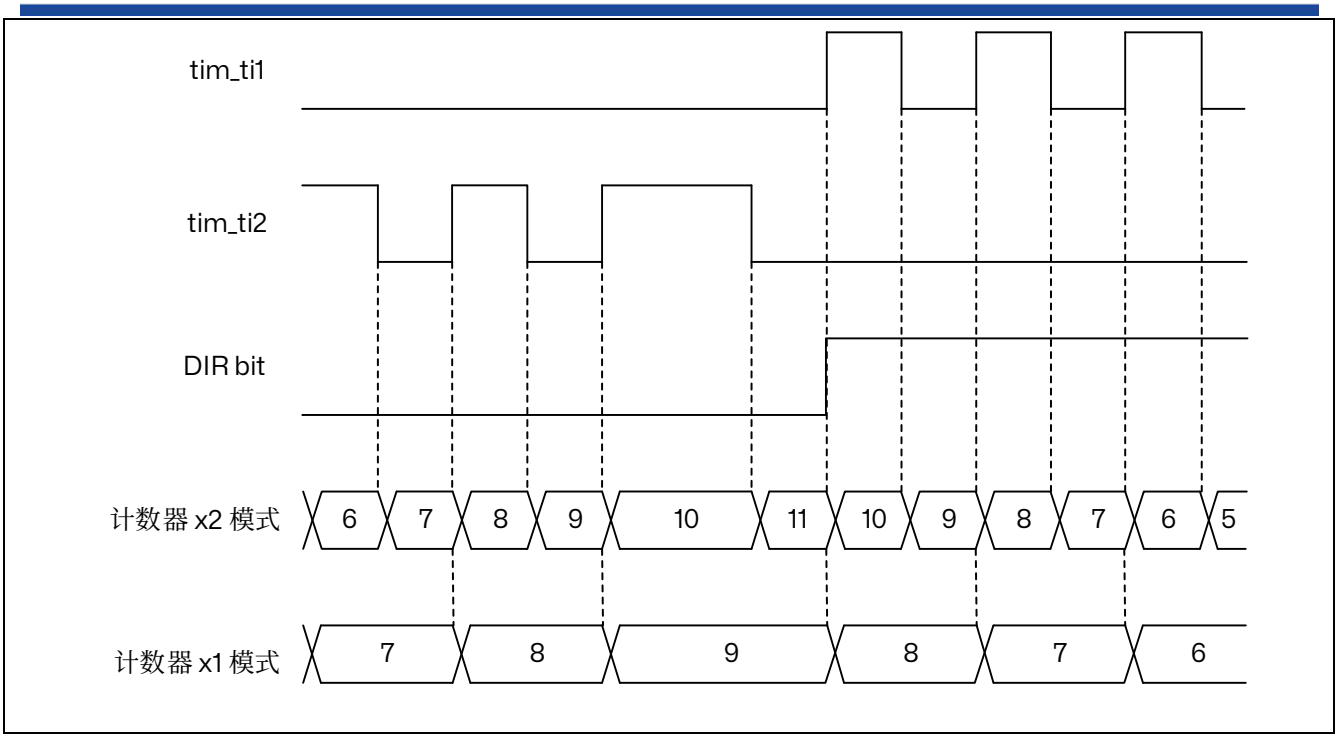



图 22.65 方向时钟编码器模式（CC1P=CC2P=1）

下表详细说明了方向时钟模式对任何输入转变的操作方式。

表 22.17 计数方向与编码器信号及极性设置的关系

| 方向时钟模式 | SMS[3:0] | 相对信号的电平（tim_ti1fp1 对应 tim_ti2, tim_ti2fp2 对应 tim_ti1） | tim_ti1fp1 信号 | | tim_ti2fp2 信号 | |
|---|---|---|---|---|---|---|
| | | | 上升 | 下降 | 上升 | 下降 |
| x2 模式 CCxP=0 | 1100 | 高 | 递减 | 递减 | 递增 | 递增 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| x2 模式 CCxP=1 | 1100 | 高 | 不计数 | 不计数 | 不计数 | 不计数 |
| | | 低 | 递减 | 递减 | 递增 | 递增 |
| x1 模式 CCxP=0 | 1101 | 高 | 不计数 | 递减 | 不计数 | 递增 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| x1 模式 CCxP=1 | 1101 | 高 | 不计数 | 不计数 | 不计数 | 不计数 |
| | | 低 | 递减 | 不计数 | 递增 | 不计数 |

### 索引输入

计数器可以通过来自编码器的索引信号进行复位，以指示绝对参考位置。索引信号必须连接到 tim_etr_in 输入。它可以使用数字输入滤波器进行滤波。

通过设置 TIMX_ECR 寄存器中的 IE 位，可以使能索引功能。只有在编码器模式下，当 SMS[3:0]位域具有以下值时，才可以设置 IE 位：0001、0010、0011、1010、1011、1100、1101、1110、1111。

根据下图所示，有几种可设置的编码器选项，用于索引脉冲的调节：

- 与 A 和 B 门控：脉冲宽度为 1/4 的通道周期，与 A 和 B 边缘对齐
- 与 A（或与 B）门控：脉冲宽度为 1/2 的通道周期，与通道 A（或通道 B）的两个边缘对齐
- 非门控：脉冲宽度可达一个通道周期，不与任何边缘对齐

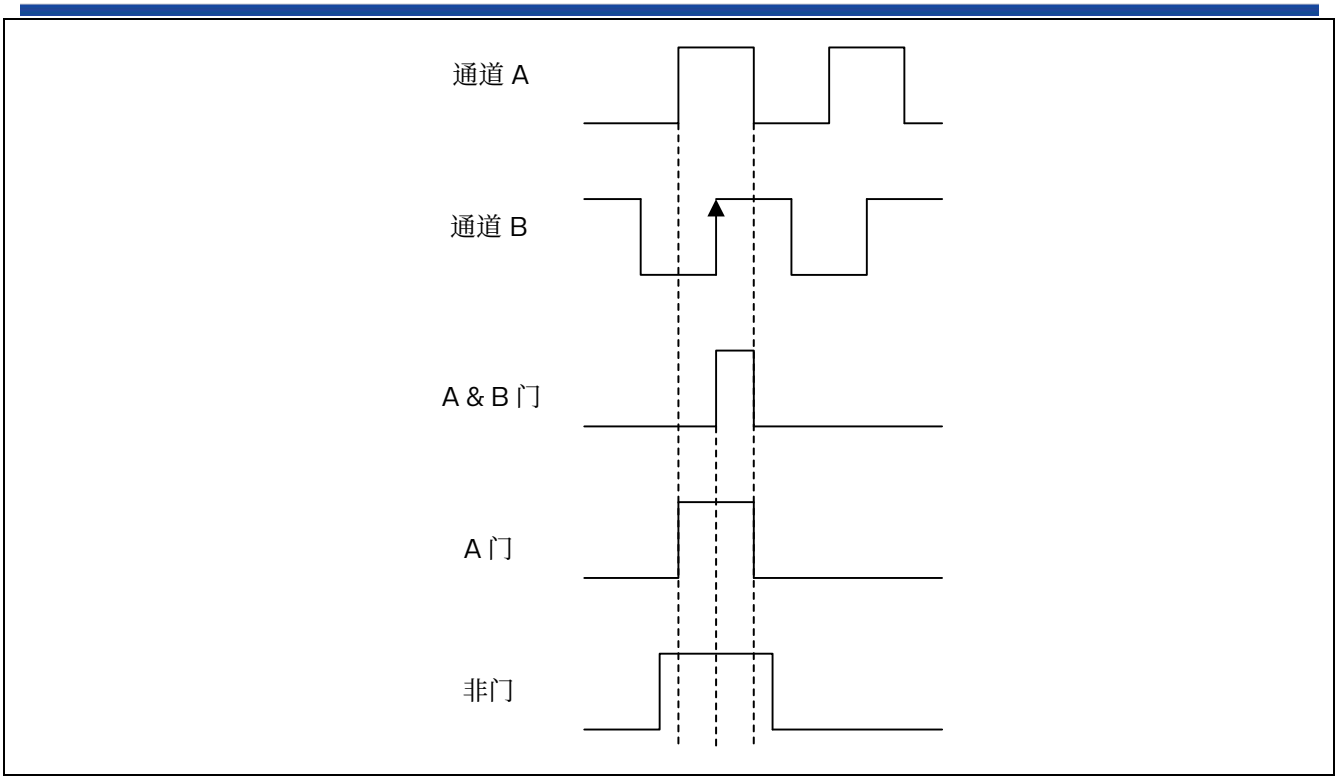



图 22.66 索引门控选项

电路可以容忍索引信号的抖动，无论门控模式如何，如下图所示。

在非门控模式下，信号必须严格低于 2 个编码器周期。如果脉冲宽度大于或等于 2 个编码器周期，则计数器会多次复位。

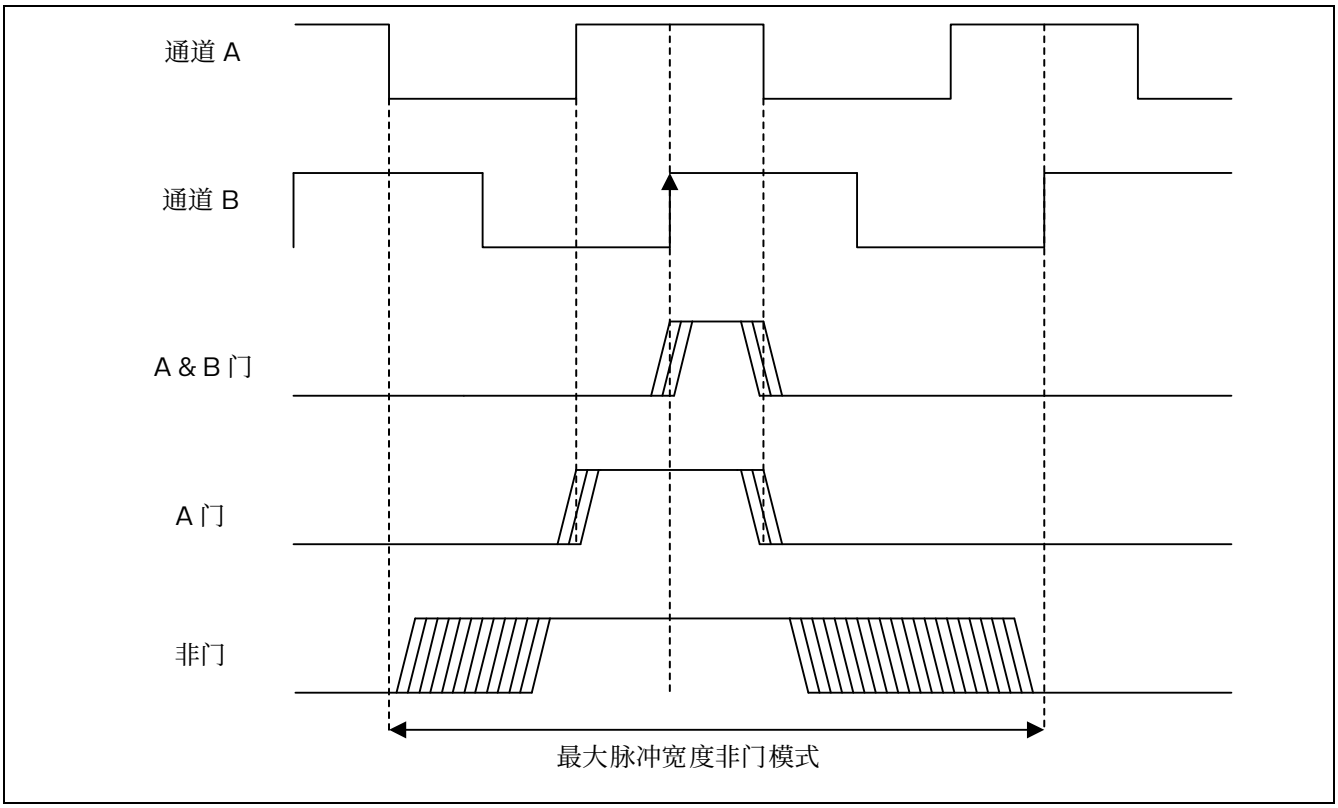



图 22.67 抖动的索引信号

定时器支持 3 种门控选项，不需要任何特定的编程。只需要定义在哪个编码器状态（即通道 A 和通道 B 状态组合）上，索引必须同步，使用 TIMx_ECR 寄存器中的 IPOS[1:0]位域。

索引检测事件根据计数方向的不同而有所不同，以确保速度反转时的对称操作：

- 递增计数时（DIR 位=0），计数器被复位。
- 递减计数时，计数器被设置为 TIMx_ARR。

这使得索引可以在相同的机械角度位置上生成，无论计数方向如何。下图显示了在一个简单的例子中（一个编码器每次机械旋转提供 4 个边缘），索引在哪个位置生成。

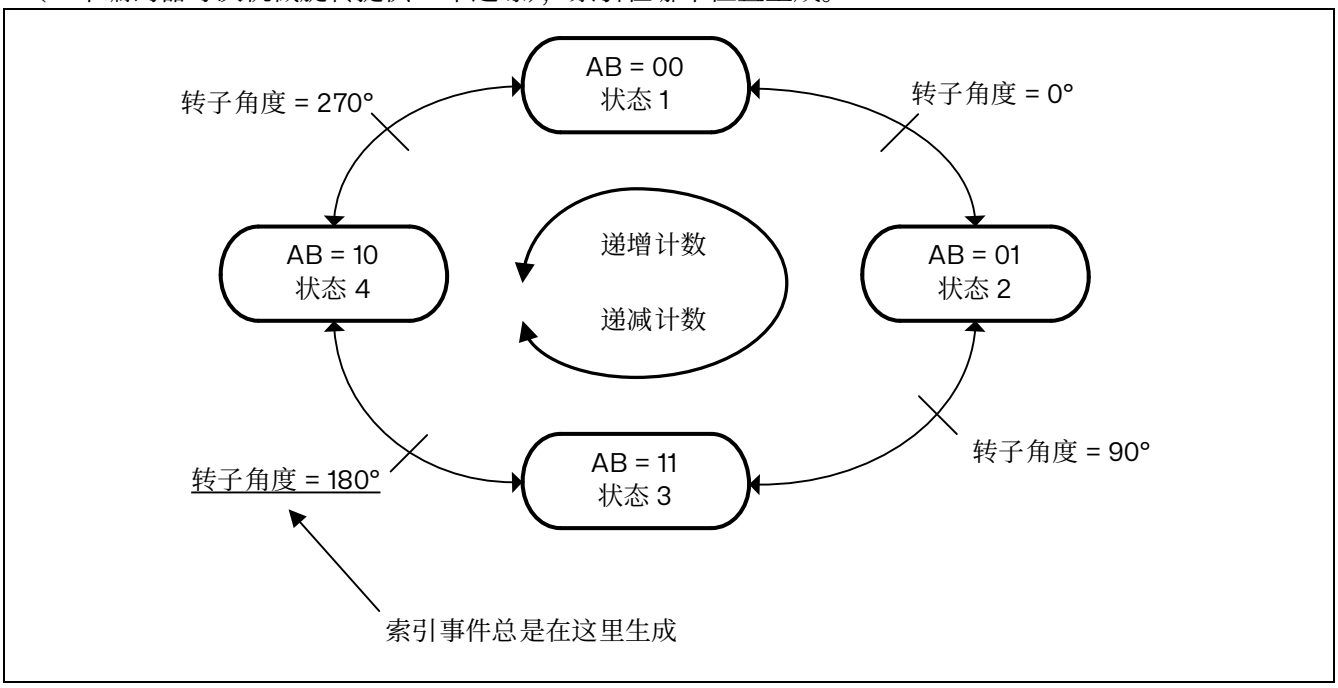



图 22.68 IPOS[1:0] = 11 时的索引生成

下图展示了 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会自动根据计数方向进行调整：

- 递增计数时（DIR 位=0），当编码器状态为“11”（ChA=1，ChB=1）时，计数器设为 0。
- 递减计数时（DIR 位=1），当退出“11”状态时，计数器设为 TIMx_ARR。

可以在索引检测事件发生时发出中断。

箭头指示在哪个转换上生成索引事件中断。

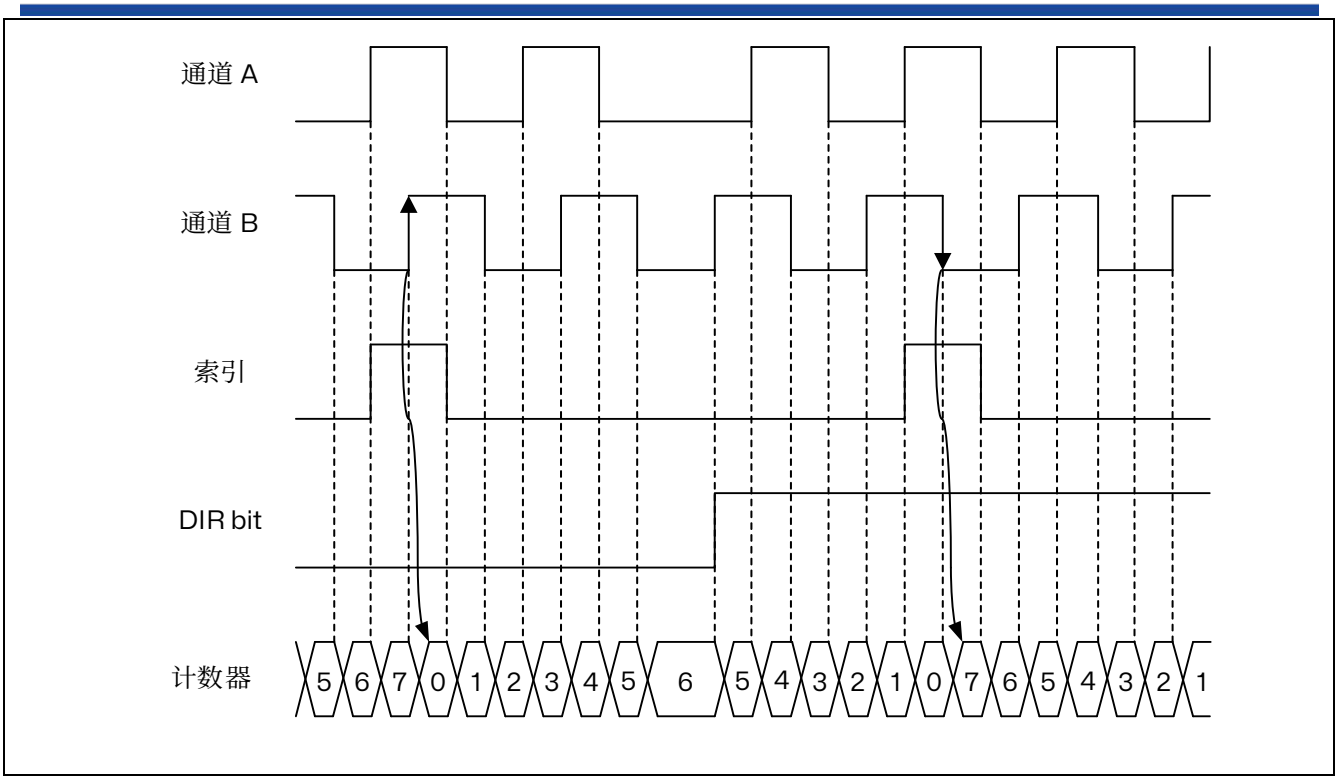



图 22.69 通道 A 上带索引门控的计数器读取（IPOS[1:0] = 11）

下图展示了非门控模式的波形和相应值。箭头指示在哪个转换上生成索引事件。

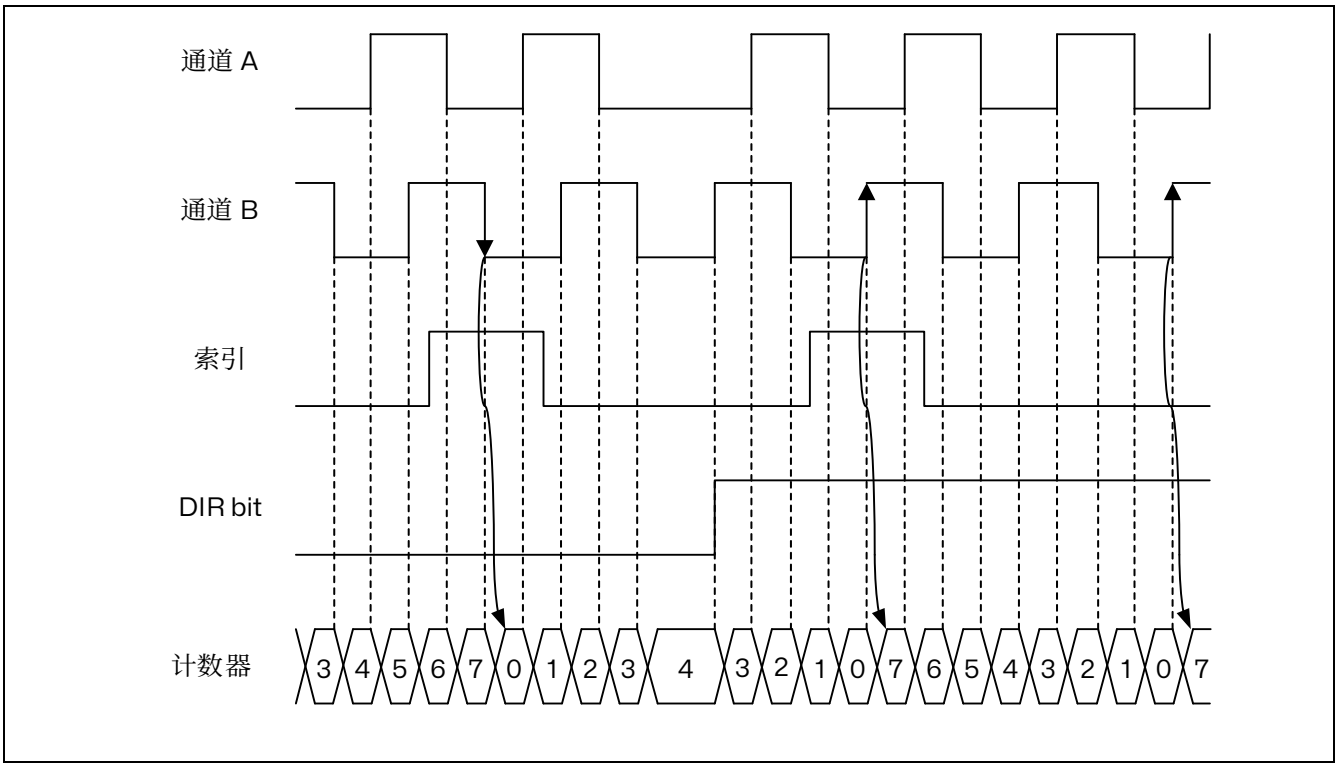



图 22.70 非门控模式下索引的计数器读取（IPOS[1:0] = 00）

下图展示了针对各种脉冲对齐情况下的“在 A 和 B 门控”模式处理方式。箭头指示在哪个转换上生成索引事件。

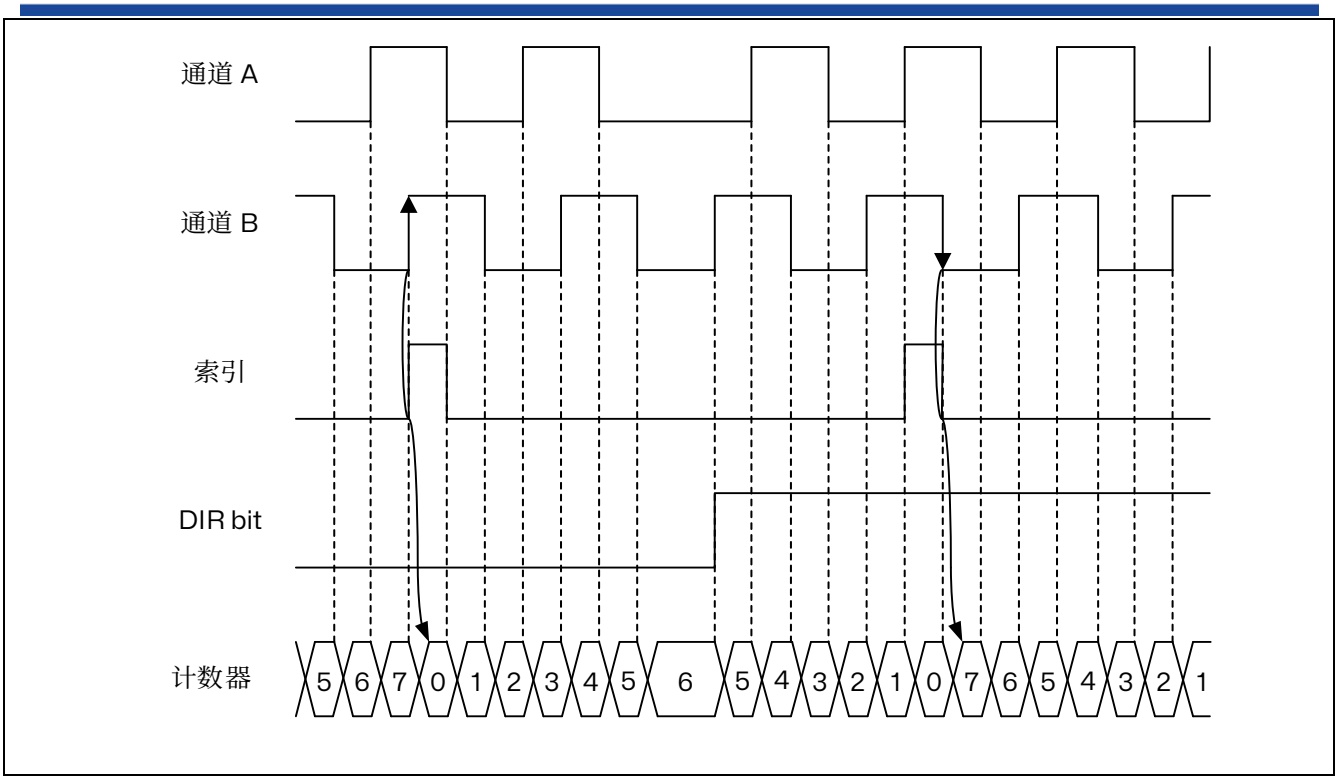
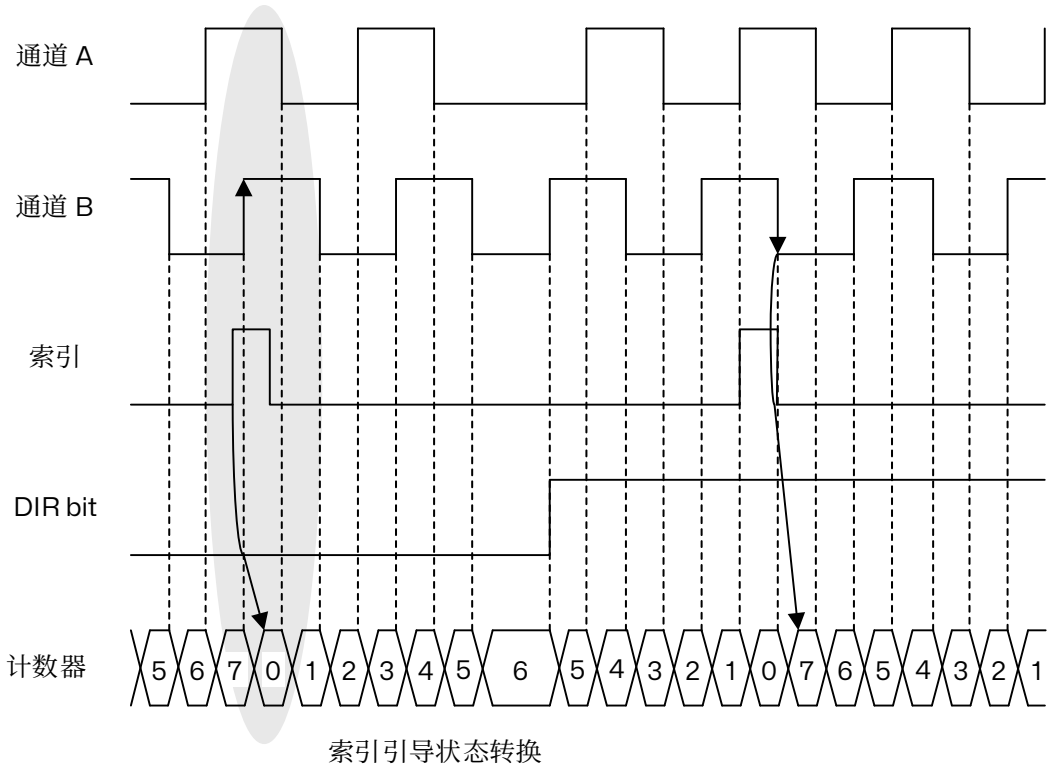
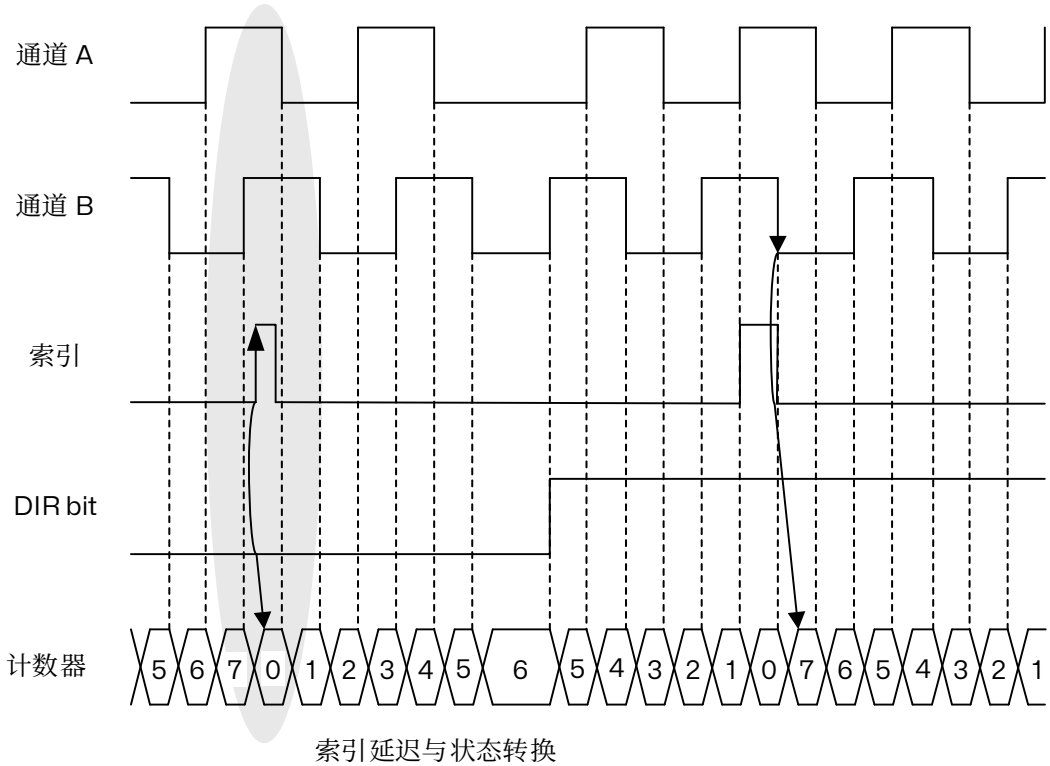



图 22.71 通道 A 和 B 上带索引门控的计数器读取

下两张图详细说明了后续索引脉冲可能比编码器时钟周期的四分之一窄的情况。




索引引导状态转换




索引延迟与状态转换

图 22.72 窄索引脉冲情况下的编码器模式行为（IPOS[1:0] = 11）

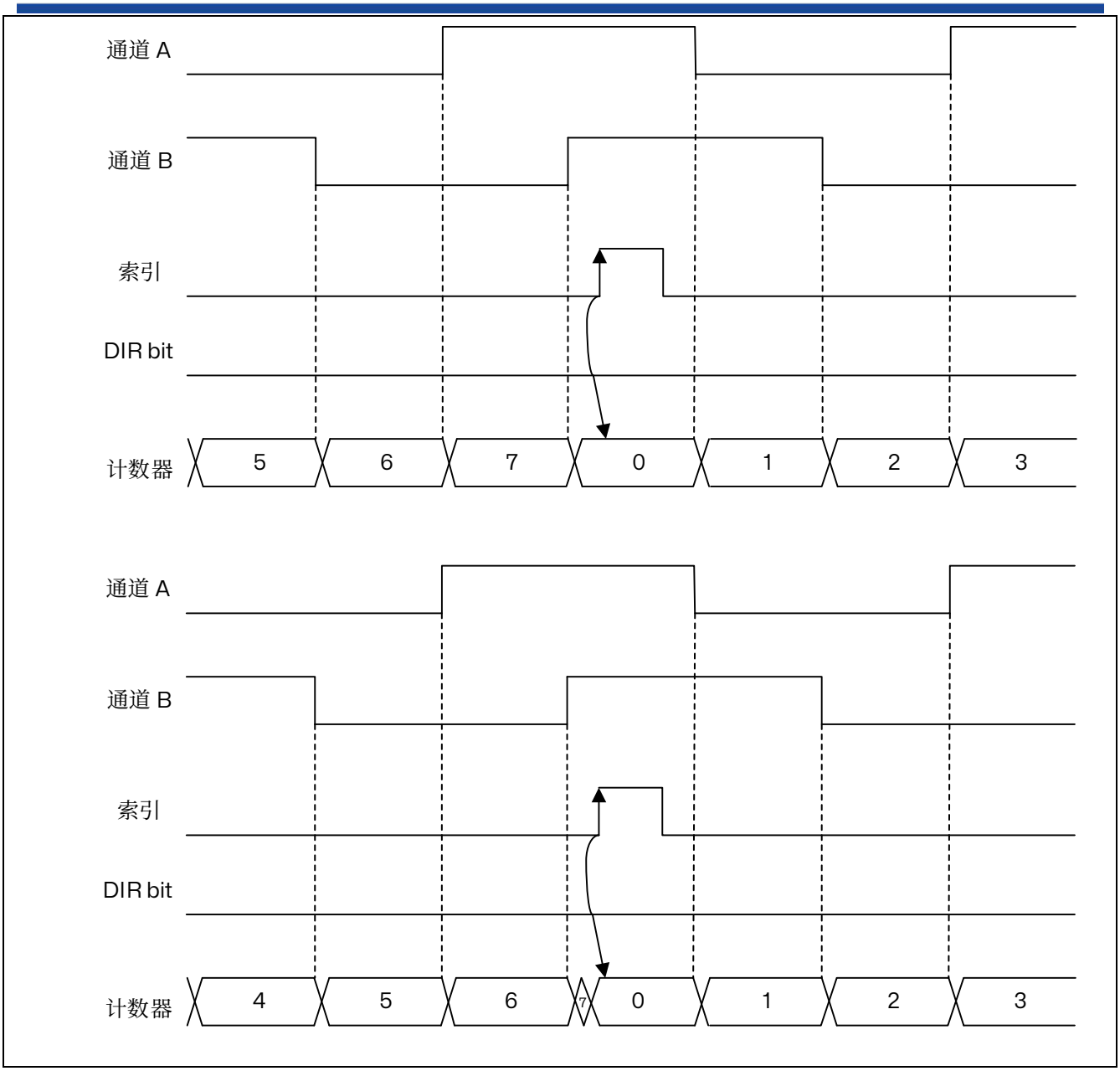



图 22.73 计数器重置窄索引脉冲（更近距离，ARR = 0x07）

下图说明了 x1 和 x2 模式下的索引管理方式。

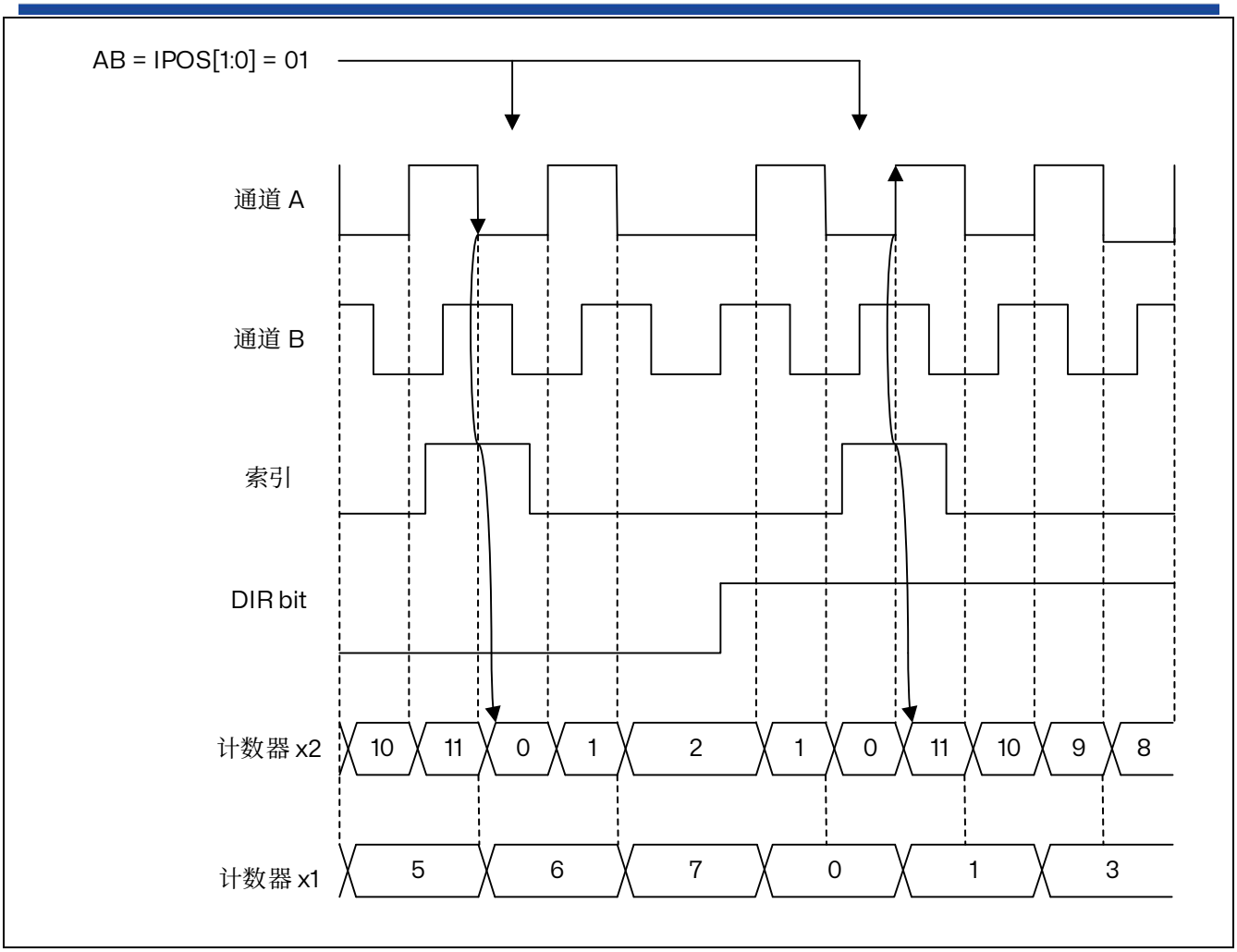



图 22.74 x1 和 x2 模式下的索引行为（IPOS[1:0] = 01）

方向索引灵敏度

TIMx_ECR 寄存器中的 IDIR[1:0]位域允许索引仅在选定计数方向上有效。

下图显示了根据 IDIR[1:0]值索引和计数器重置事件之间的关系。

*注：当* IE *位被复位时（禁用索引模式），必须写入* IDR[1:0] *位域。*

*注：在时钟+方向模式下不支持定向索引灵敏度。当* SMS[3:0] = 1010 *或* 1011 *时，* IDIR[1:0] *必须设置为* 00。

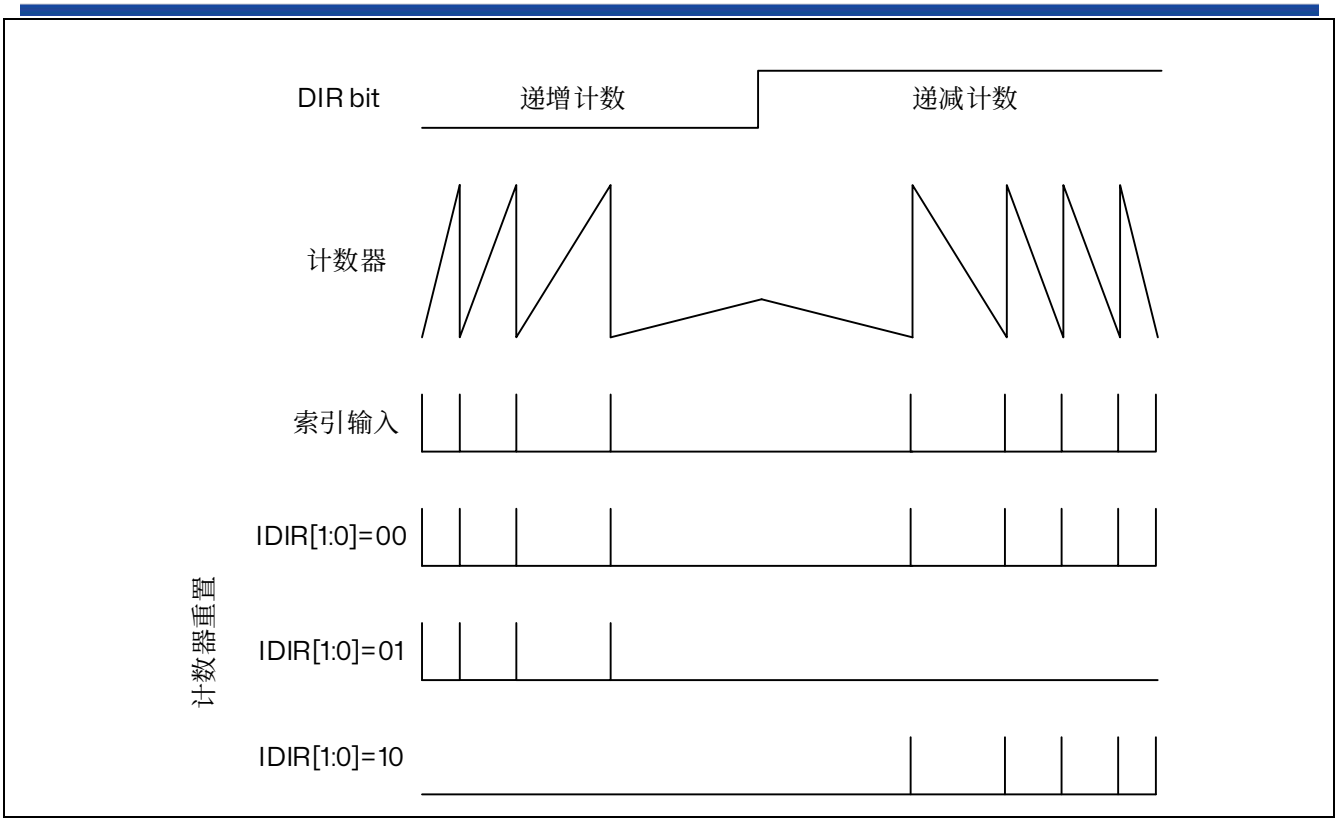



图 22.75 定向索引灵敏度

### 特殊的第一索引事件管理

TIMx_ECR 寄存器中的 FIDX 位允许仅捕获一次索引，如下图所示。一旦第一个索引到达，任何后续索引都会被忽略。如果需要，可以通过将 FIDX 位写入 0 并再次将其设置为 1 来重新启动电路。

*注：当 FIDX = 1 时，如果方向在位置 0 处改变（索引激活），索引可以发布两次（IDXF 标志设置）。*

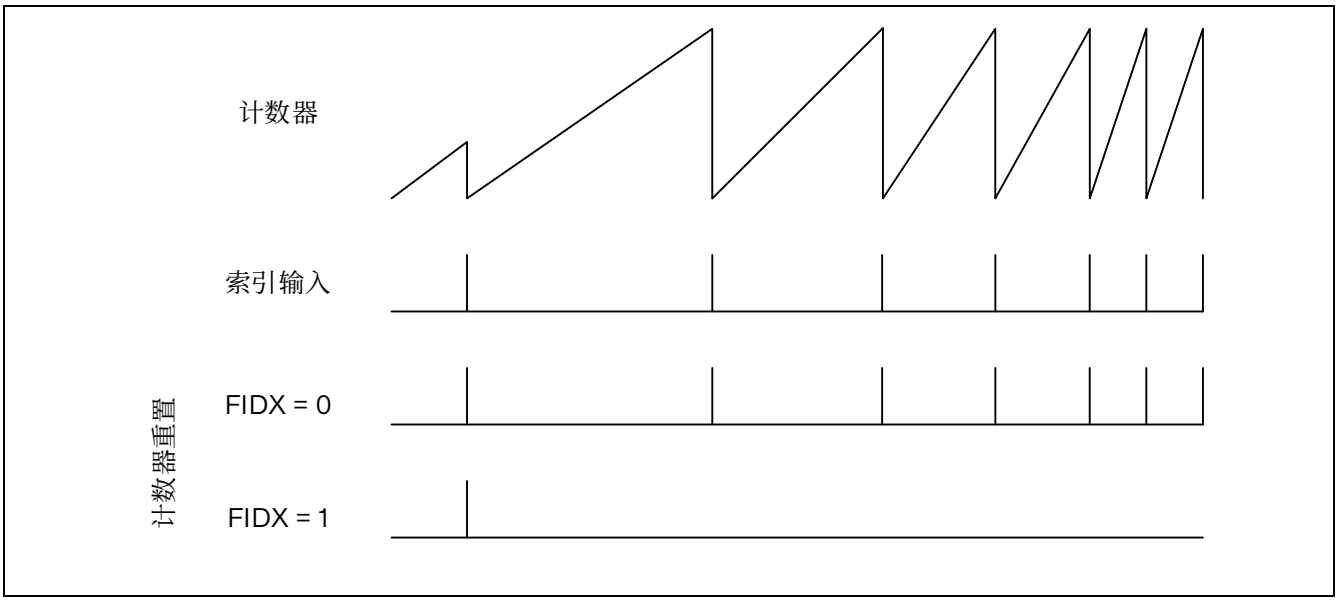



图 22.76 作为 FIDX 位设置的功能的计数器重置

### 非正交模式下的索引管理

下面两张图详细说明了当 SMS[3:0]位域等于 1010、1011、1100、1101 时，如何在方向时钟模式和时钟加方向模式下管理索引。

对于这两种模式，索引灵敏度通过 IPOS[0]位进行设置，具体如下：

- IPOS[0] = 0：在时钟低电平上检测索引
- IPOS[0] = 1：在时钟高电平上检测索引

IPOS[1]位不重要。

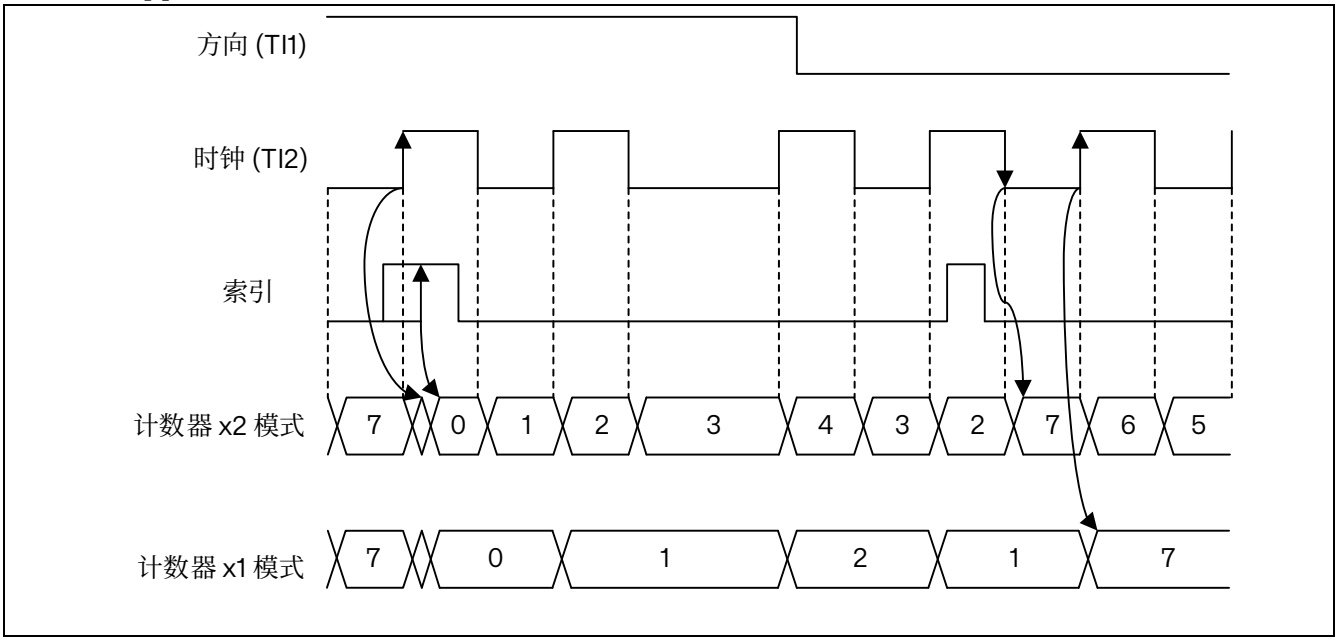



图 22.77 时钟+方向模式下的索引行为，IPOS[0] = 1

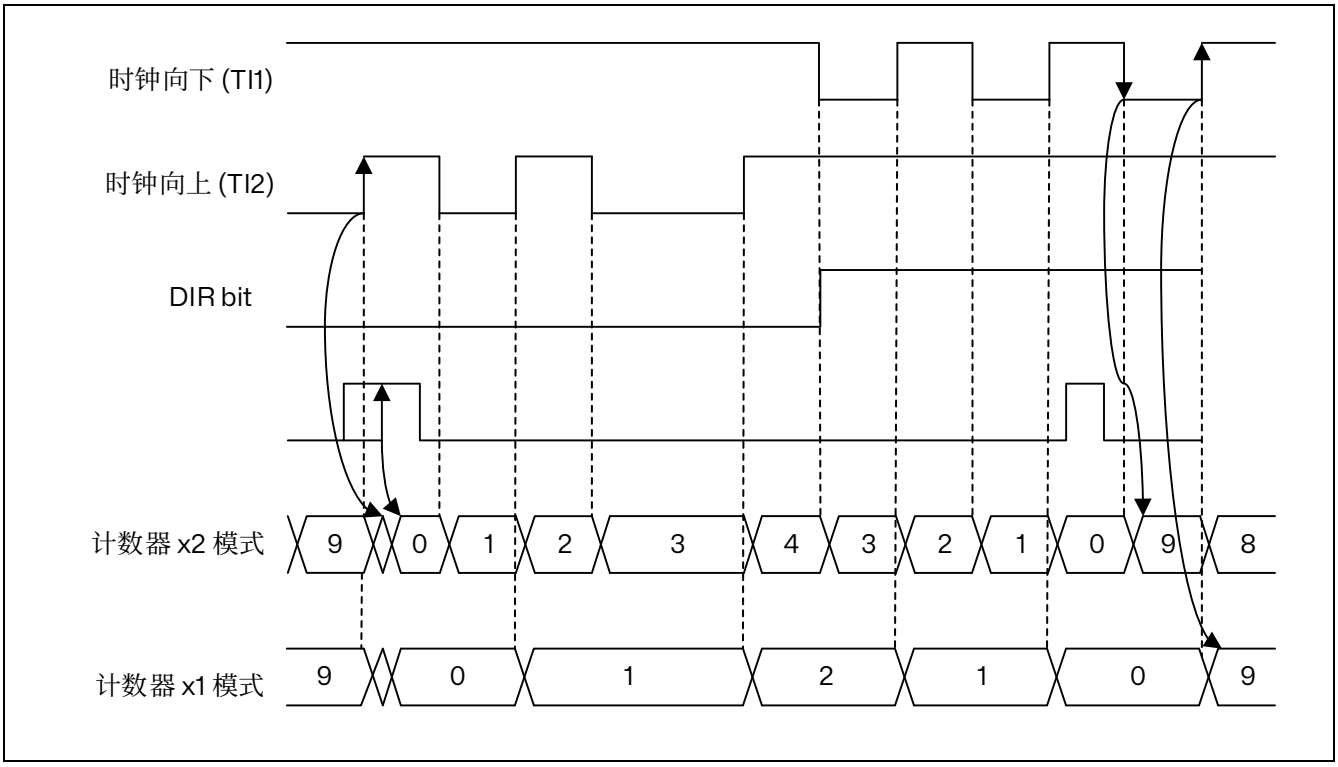



图 22.78 方向时钟模式下的索引行为，IPOS[0] = 1

### 编码器错误管理

对于具有 2 个正交信号的编码器配置，可以检测到转换错误。2 个输入的读数对应于一个 2 位灰度码，该灰度码可以表示为状态图，如以下图所示。期望单个比特立即更改。错误的转换会在 TIMx_SR 状态寄存器中设置 TERRF 中断标志。如果 TIMx_DIER 寄存器中的 TERRIE 位被设置，将产生转换错误中断。

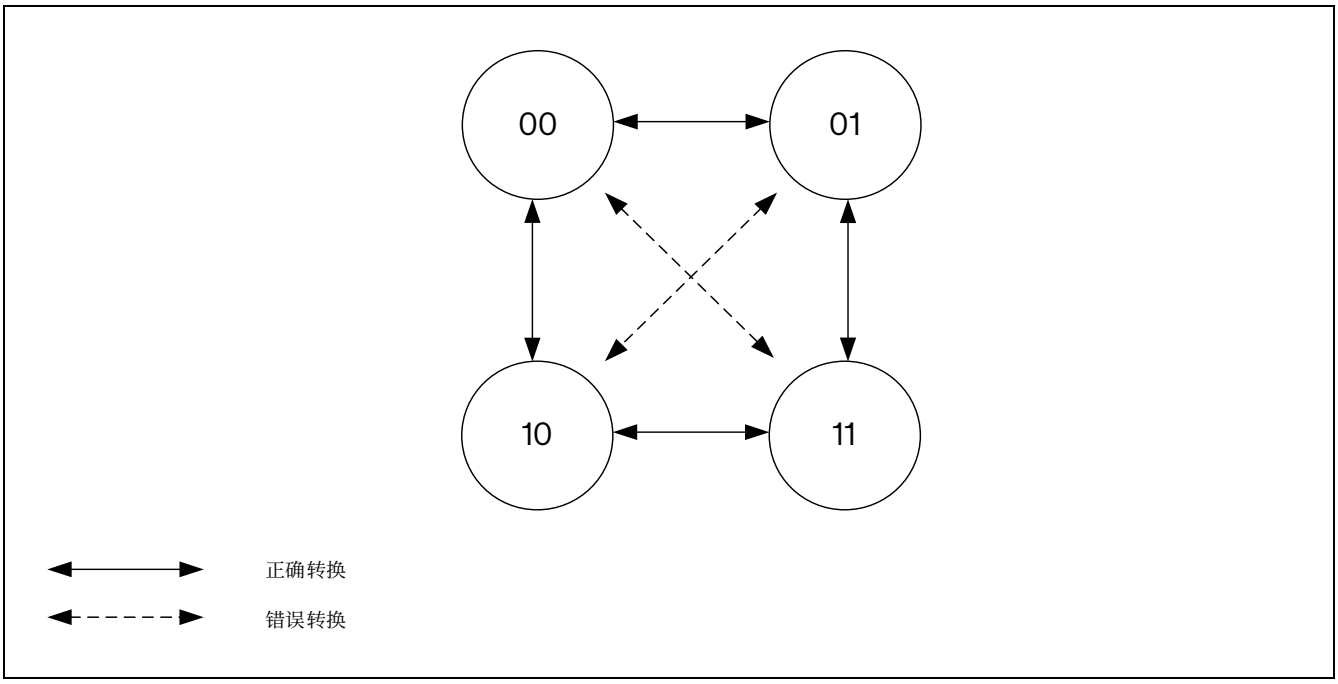



图 22.79 正交编码信号的状态图

对于具有索引信号的编码器，可以检测到导致每转脉冲过多的异常操作。一个编码器每转提供 N 个脉冲，则每转有 4xN 个计数。索引信号每 4xN 个时钟周期重置一次计数器。

如果计数器的值从 TIMx_ARR 增加到 0 或从 0 减少到 TIMxARR 的值而没有索引事件，则报告为索引位置错误。

可使用 TIMx_ARR 寄存器来编程溢出阈值。一个 1000 线的编码器导致计数器的值在 0 和 3999 之间（在 4 倍读取模式下）。必须通过设置 TIMx_ARR = 3999 + 1 = 4000 来编程溢出检测阈值。

错误断言被延迟到在递增计数时从 0 到 1 的转换。这是为了处理门控 A 和 B 模式中的窄索引脉冲，如下图所示。

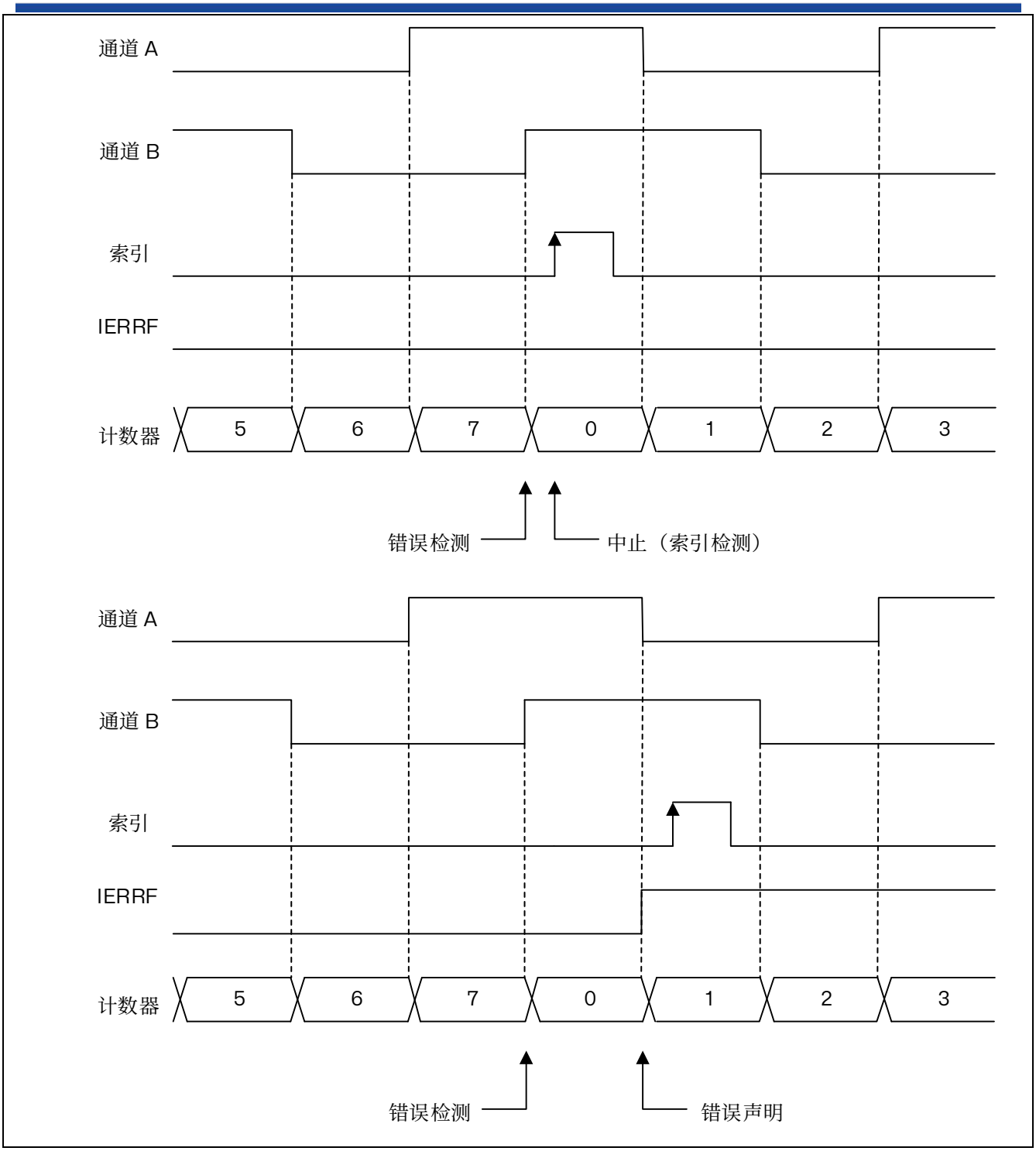



图 22.80 递增计数编码器错误检测

在递减计数模式下，检测是由从 1 到 0 的初步转换引起的。这是为了处理门控 A 和 B 模式中的窄索引脉冲，如下图所示，以避免编码器在索引检测后立即在 TIMx_ARR 和 0 之间抖动时出现任何错误检测。

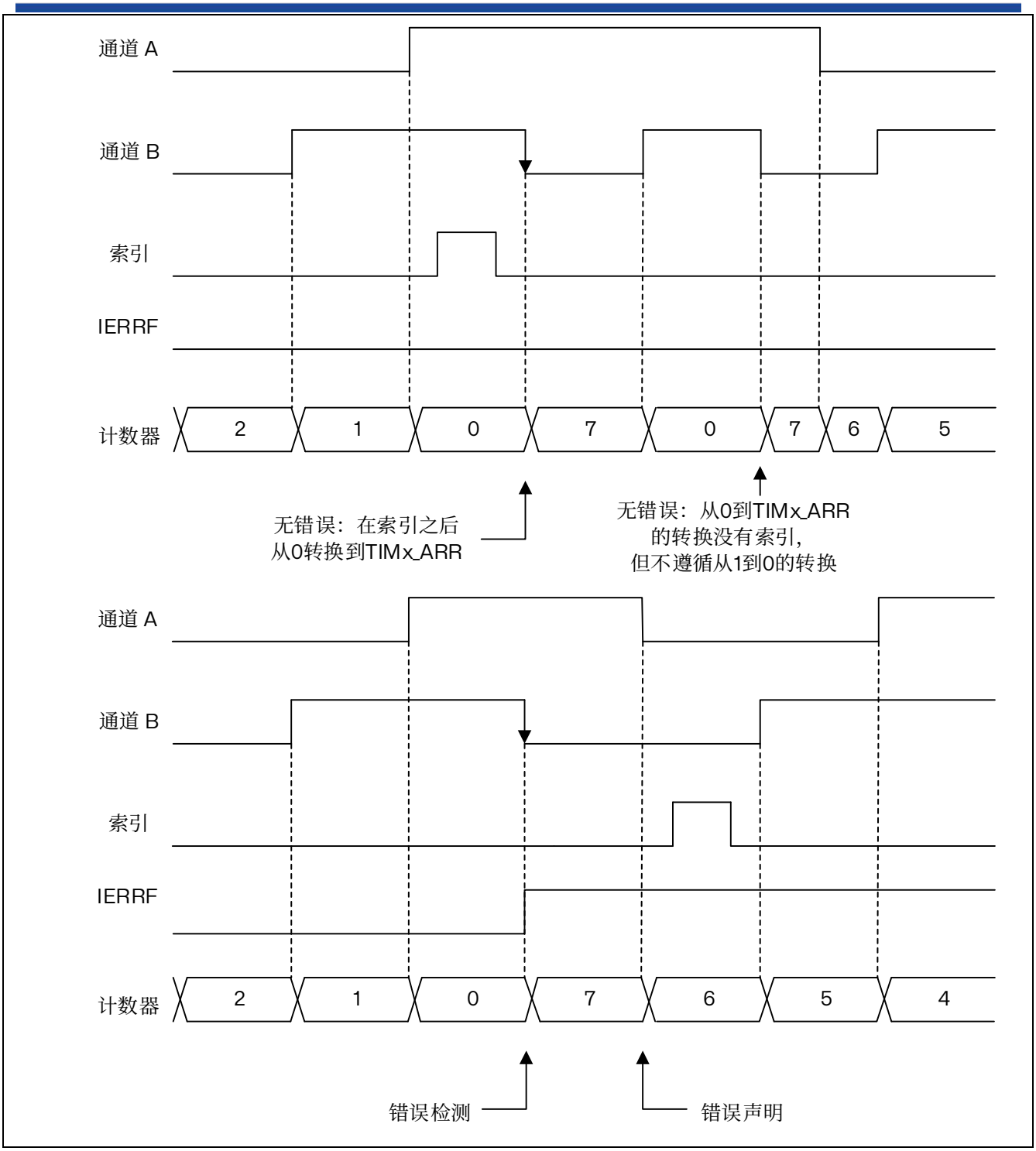



图 22.81 递减计数编码器错误检测

索引错误会在 TIMx_SR 状态寄存器中设置 IERRF 中断标志。如果 TIMx_DIER 寄存器中的 IERRIE 位被设置，将产生索引错误中断。

### 功能编码器中断

编码器模式下也可使用以下中断：

- 方向改变：在编码器模式下，任何计数方向的改变都会导致 TIMx_CR1 寄存器中的 DIR 位发生翻转。方向改变会设置 TIMx_SR 状态寄存器中的 DIRF 中断标志。如果 DIRIE 位在 TIMx_DIER 寄

存器中被设置，则产生方向改变中断。

- 索引事件：索引事件会设置 TIMx_SR 状态寄存器中的 IDXF 中断标志。如果 IDXIE 位在 TIMx_DIER 寄存器中被设置，则产生索引中断。

### 运行时更新编码器模式时从机模式选择预加载

在运行过程中，可能需要从一个编码器模式切换到另一个编码器模式。这通常在高速度下完成，以降低更新中断频率，从 x4 模式切换到 x2 模式再到 x1 模式，如下图所示。

为此，可以预加载 SMS[3:0]位。通过设置 TIMx_SMCR 寄存器中的 SMSPE 使能位可以使能此功能。从 SMS[3:0]预加载值转换到有效值可以由 TIMx_SMCR 寄存器中的 SMSPS 位选择触发。

- SMSPS = 0：当计数器递增时溢出并在计数器递减时下溢时，由更新事件(UEV)触发的转换。只有在索引被禁用时（位 IE = 0）才可以使用此模式。
- SMSPS = 1：由索引事件触发的转换。

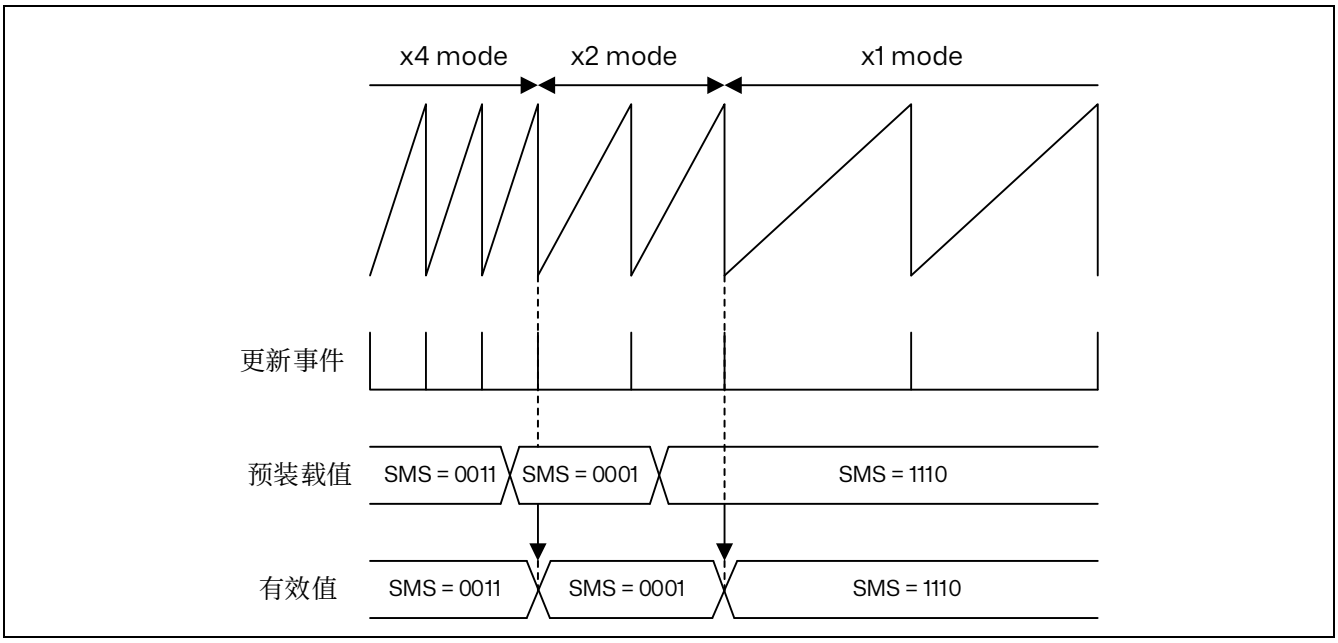



图 22.82 更新时带有预加载转移的编码器模式更改（SMSPS = 0）

### 编码器时钟输出

编码器模式的操作原理并不完全适合于高分辨率速度测量，在低速时，因为它需要相对较长的积分时间才能获得足够数量的时钟边缘和精确的测量。

在低速时，更好的解决方案是进行边到边的时钟周期测量。这可以通过使用从属定时器来实现。定时器可以在 tim_trgo 输出上输出编码器时钟信息。然后，从属定时器可以进行周期测量并为每个编码器时钟边缘提供速度信息。

通过将 TIMx_CR2 寄存器中的 MMS[3:0]位域设置为 1000 可以使能此模式。对于以下 SMS[3:0]值有效：0001、0010、0011、1010、1011、1100、1101、1110、1111。不允许任何其他 SMS[3:0]代码，并可能导致意外行为。

## 22.3.26 方向位输出

可以将方向信号从定时器输出到 tim_oc3n 和 tim_oc4 输出信号（TIMx_CR1 寄存器中的 DIR 位的副本）。这是通过将 TIMx_CCMR2 寄存器中的 OC3M[3:0]或 OC4M[3:0]位域设置为 1011 来实现的。

此功能可用于在编码器模式下监视计数方向（或旋转方向），或在中心对齐的 PWM 模式下具有指示

上/下相位的信号。

### 22.3.27 UIF 位重映射

TIMx_CR1 寄存器中的 UIFREMAP 位强制将更新中断标志 UIF 连续复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许以原子方式读取计数器的值和一个潜在的由 UIFCPY 标志指示的翻转条件。在特定情况下，它可以避免由背景任务（读取计数器）和中断（更新中断）之间的处理引起的竞争条件，从而简化计算。

UIF 和 UIFCPY 标志之间没有延迟。

### 22.3.28 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，可将通道 1 的输入滤波器连接到异或门的输出，从而将 tim_ti1，tim_ti2，tim_ti3 这三个输入引脚组合在一起。

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。这对于测量两个输入信号之间的边缘间隔非常方便，如下图所示。

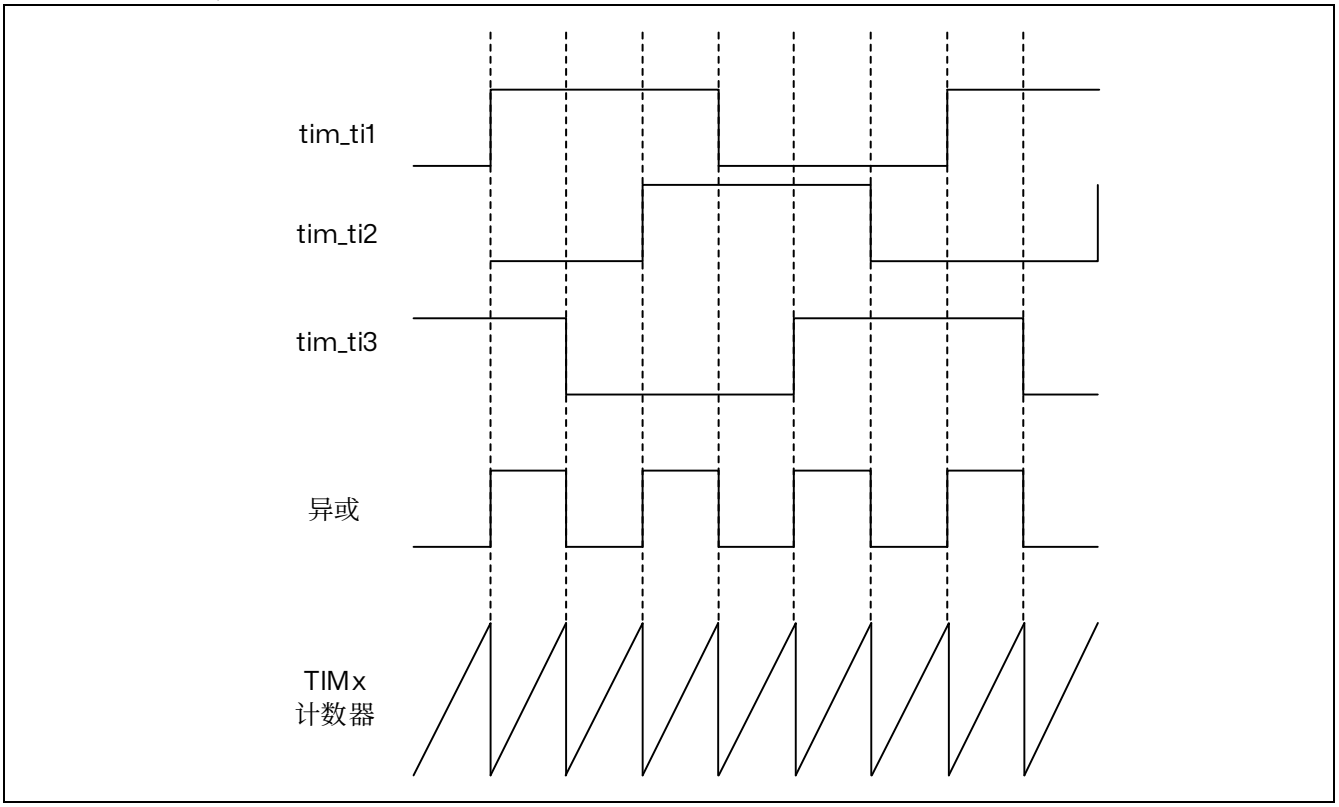



图 22.83 测量 3 个信号边缘之间的时间间隔

### 22.3.29 与霍尔传感器接口

这是通过使用高级控制定时器生成 PWM 信号来驱动电机，并使用另一个被称为“接口定时器”的 TIMx 定时器（如下图）来实现的。“接口定时器”捕获通过 XOR 连接到 tim_ti1 输入通道（通过设置 TIMx_CR2 寄存器中的 TI1S 位来选择）的 3 个定时器输入引脚（tim_ti1、tim_ti2 和 tim_ti3）。

从属模式控制器配置为复位模式；从属输入是 tim_ti1f_ed。因此，每当 3 个输入中的任何一个切换时，计数器都会从 0 开始重新计数。这创建了一个由霍尔输入上的任何变化触发的时间基准。

在“接口定时器”上，捕获/比较通道 1 配置为捕获模式，捕获信号是 tim_trc。捕获的值对应于输入

上两次变化之间的时间间隔，提供了有关电机速度的信息。

“接口定时器”可以用于输出模式，以生成一个脉冲，该脉冲会改变高级控制定时器的通道配置（通过触发 COM 事件）。高级控制定时器用于生成驱动电机的 PWM 信号。为此，必须编程接口定时器的通道，以便在编程延迟后生成正脉冲（在输出比较或 PWM 模式下）。该脉冲通过 tim_trgo 输出发送到高级控制定时器。

例如：每当连接到其中一个 TIMx 定时器的霍尔输入发生变化时，都希望在编程延迟后更改高级控制定时器的 PWM 配置。

- 通过将 TIMx_CR2 寄存器中的 TI1S 位设置为“1”，使 3 个定时器输入经过异或运算后进入 tim_ti1 输入通道，
- 时基编程：将 TIMx_ARR 写入最大值（计数器必须通过 tim_ti1 的变化清零。设置预分频器，以得到最大计数器周期，该周期长于传感器上两次变化的间隔时间
- 将通道 1 配置为捕获模式（选择 tim_trc）：将 TIMx_CCMR1 寄存器中的 CC1S 位写入“01”。如果需要，还可以编程数字滤波器，
- 设置通道 2 为 PWM2 模式，并具有所需延时：将 TIMx_CCMR1 寄存器中的 OC2M 位写入“111”，将 CC2S 位写入“00”，
- 选择 tim_oc2ref 作为 tim_trgo 上的触发输出：将 TIMx_CR2 寄存器中的 MMS 位写入“101”，

在高级控制定时器中，必须选择正确的 tim_itrx 输入作为触发输入，定时器被编程为生成 PWM 信号，捕获/比较控制信号被预加载（在 TIMx_CR2 寄存器中设置 CCPC=1），并且 COM 事件由触发输入控制（在 TIMx_CR2 寄存器中设置 CCUS=1）。PWM 控制位（CCxE、OCxM）在下一步的 COM 事件后写入（这可以在由 tim_oc2ref 的上升边沿生成的中断子程序中完成）。

下图描述了这个例子。

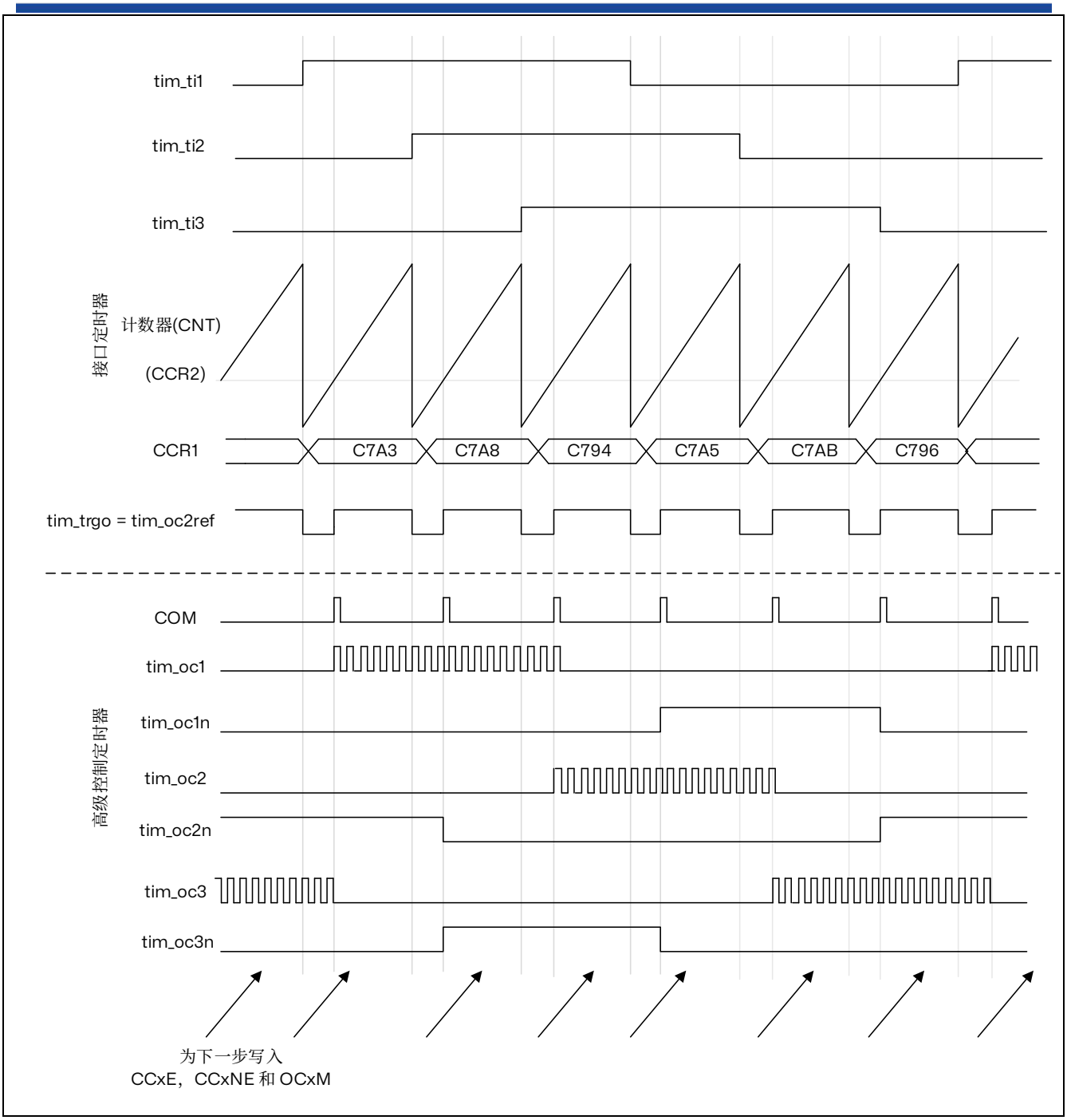



图 22.84 霍尔传感器接口的实例

### 22.3.30 定时器同步

TIMx 定时器在内部连接在一起，用于定时器同步或链接。它们可以在几种模式下同步：复位模式，门控模式，触发模式，复位+触发和门控+复位模式。

**从模式：复位模式**

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 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，tim_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 tim_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 和 CC1NP=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 tim_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 tim_ti1 出现一个上升沿；当 tim_ti1 上升时，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，中断请求或 DMA 请求可以被发送（取决于 TIMx_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。

在 tim_ti1 上升沿和计数器的实际复位之间的延时取决于 tim_ti1 输入端的重同步电路。

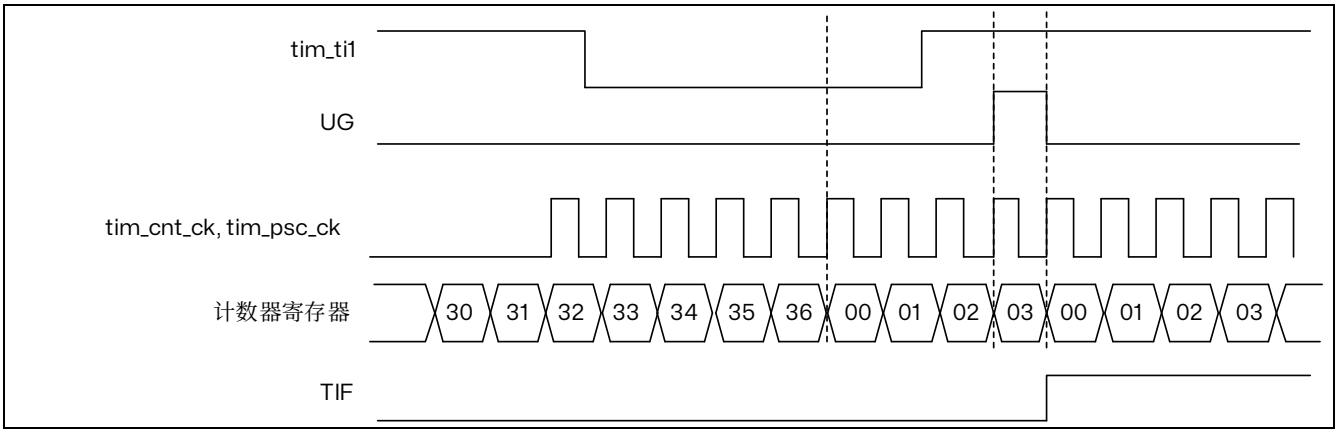



图 22.85 复位模式下的控制电路

**从模式：门控模式**

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 tim_ti1 为低时向上计数：

- 配置通道 1 以检测 tim_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 和 CC1NP=0 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 tim_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 tim_ti1 为低，计数器开始依据内部时钟计数，一旦 tim_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。

tim_ti1 上升沿和计数器实际停止之间的延时取决于 tim_ti1 输入端的重同步电路。

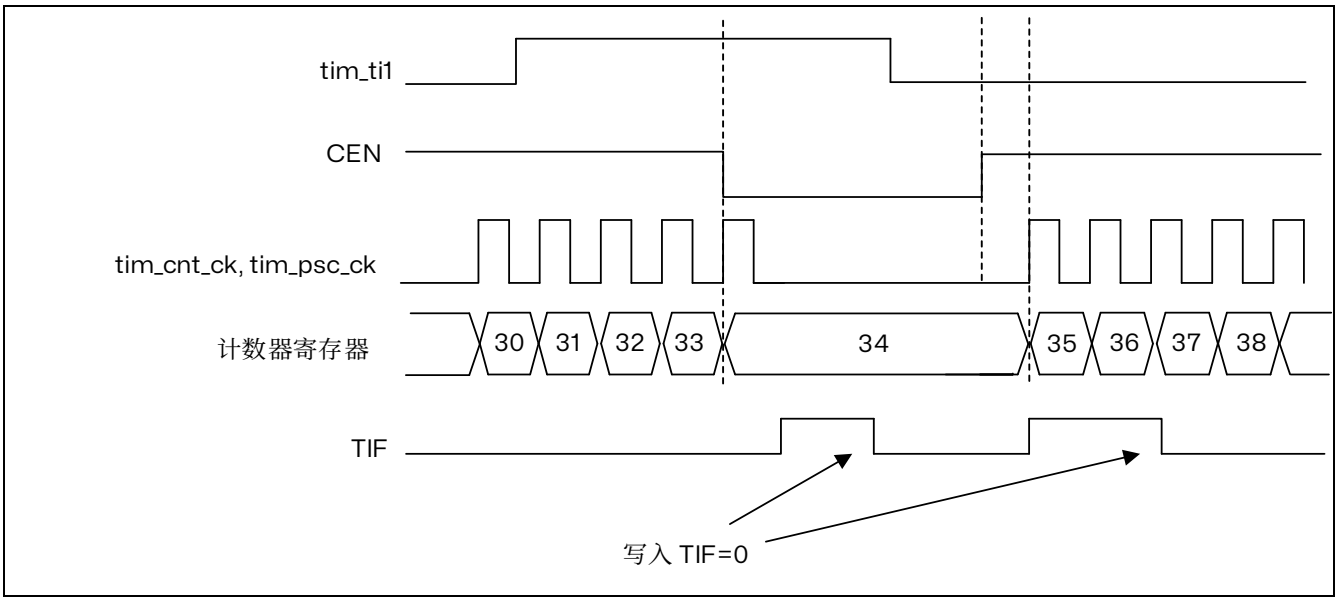



图 22.86 门控模式下的控制电路

**从模式：触发模式**

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 tim_ti2 输入的上升沿开始向上计数：

- 配置通道 2 检测 tim_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 和 CC2NP=0 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 tim_ti2 作为输入源。

当 tim_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。

tim_ti2 上升沿和计数器启动计数之间的延时，取决于 tim_ti2 输入端的重同步电路。

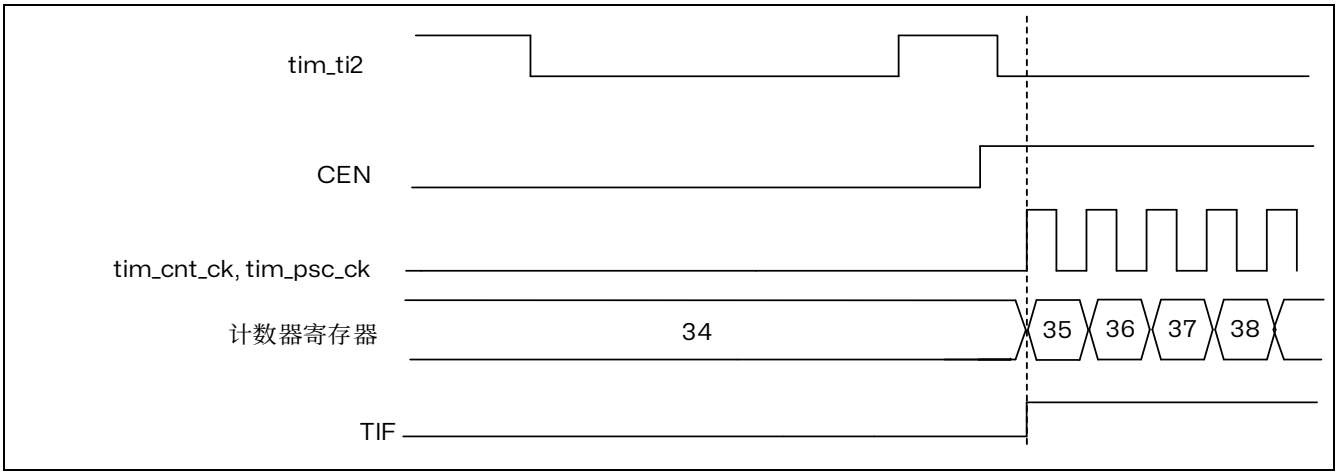



图 22.87 触发器模式下的控制电路

**从模式：组合复位+触发模式**

在这种情况下，选择的触发输入（tim_trgi）的上升沿重新初始化计数器，生成寄存器的更新，并启动计数器。

该模式用于单脉冲模式。

**从模式：组合门控+复位模式**

触发输入（tim_trgi）高时使能计数器。一旦触发变低，计数器停止并复位。计数器的开始和停止都是受控的。

该模式允许检测超出范围的 PWM 信号（占空比超过最大期望值）。

**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，tim_etr_in 信号被用作外部时钟的输入，可以选择另一个输入作为触发输入（复位模式、门控模式或触发模式）。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 tim_etr_in 作为 tim_trgi。

在下面的例子中，一旦在 tim_ti1 上出现一个上升沿，计数器即在 tim_etr_in 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 tim_etr_in 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 TI 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 和 CC1NP = 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=00101，选择 tim_ti1 作为输入源。

当 tim_ti1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 tim_etr_in 的上升沿计数。

tim_etr_in 信号的上升沿和计数器实际复位间的延时，取决于 tim_etrp 输入端的重同步电路。

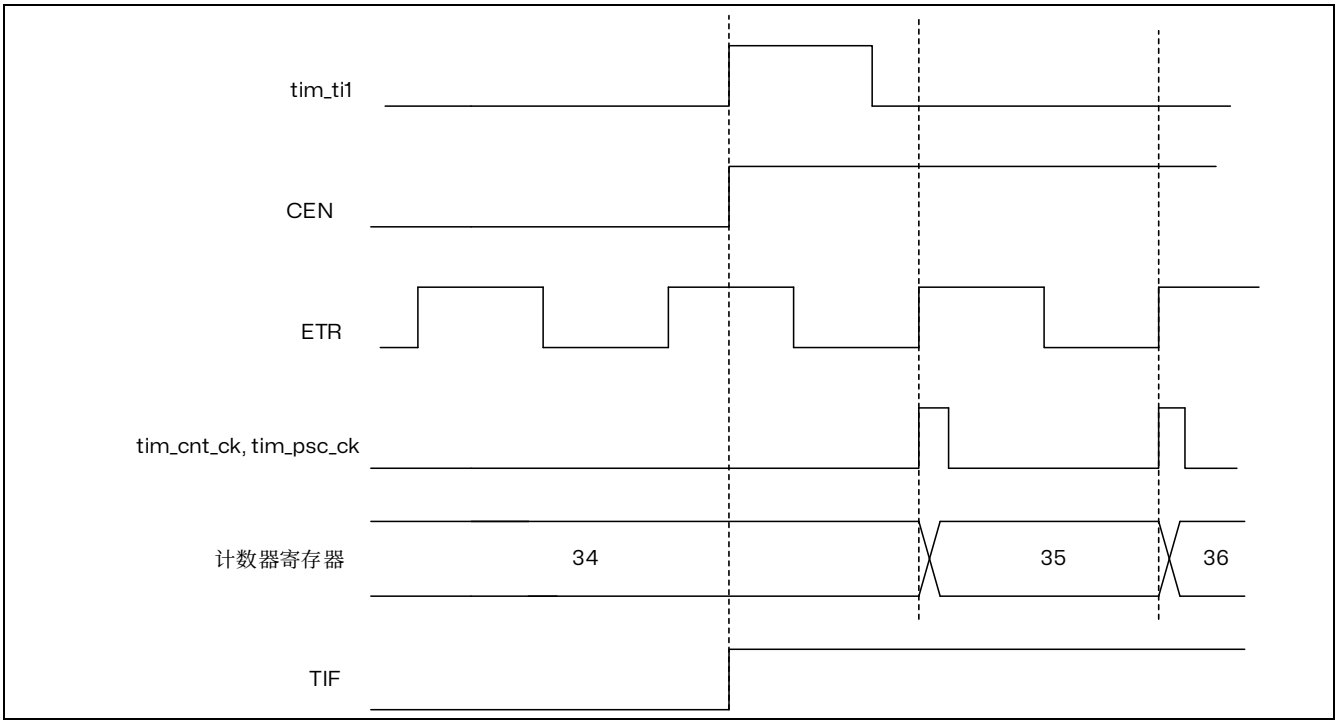



图 22.88 外部时钟模式 2+触发模式下的控制电路

*注：接收 tim_trgo 或 tim_trgo2 信号的从外设（定时器，ADC，...）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能实时更改。*

### 22.3.31 ADC 触发器

定时器可以用各种内部信号（例如重置、使能或比较事件）生成 ADC 触发事件。也可以生成由内部边缘检测器发出的脉冲。例如：

- OC4ref 的上升沿和下降沿
- OC5ref 的上升沿或 OC6ref 的下降沿

触发器在 tim_trgo2 内部线路发出，该线路被重定向到 ADC。共有 16 个可能的事件，可以使用 TIMx_CR2 寄存器中的 MMS2[3:0]位选择。前章节：三相组合 PWM 模式中给出了一个 3 相马达驱动的应用示例。

*注：接收 tim_trgo 或 tim_trgo2 信号的从属外设（定时器、ADC 等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能实时更改。*

*注：ADC 的时钟必须在从主定时器接收事件之前使能，并且在接收到定时器的触发器时不能更改*

### 22.3.32 DMA 突发模式

TIMx 定时器具有在单个事件上生成多个 DMA 请求的能力。其主要目的是能够在不增加软件开销的情况下多次重新编程定时器的一部分，但它也可以用于在固定间隔内连续读取多个寄存器。

DMA 控制器的目标地址是唯一的，必须指向虚拟寄存器 TIMx_DMAR。在给定的定时器事件上，定时器启动一系列 DMA 请求（突发）。对 TIMx_DMAR 寄存器的每次写入实际上都被重定向到定时器寄存器之一。

TIMx_DCR 寄存器中的 DBL[4:0]位设置了 DMA 突发长度。当对 TIMx_DMAR 地址进行读取或写入访问时，定时器会识别出突发传输，即传输的数量（以半字或字节为单位）。

TIMx_DCR 寄存器中的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址（当通过 TIMx_DMAR 地址进行读取/写入访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始定义的偏移量：

示例：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发模式用于在更新事件发生时更新 CCRx 寄存器（x = 2，3，4）的内容，其中 DMA 将半字传输到 CCRx 寄存器中。

这是通过以下步骤完成的：

1. 配置相应的 DMA 通道如下：
   - 无滤波器 DMA 通道外设地址是 DMAR 寄存器的地址。
   - DMA 通道内存地址是 RAM 中缓冲区的地址，其中包含要通过 DMA 传输到 CCRx 寄存器的数据。
   - 要传输的数据数量 = 3（见下文注释）。
   - 循环模式禁用。
2. 通过配置 DBA 和 DBL 位域来配置 DCR 寄存器，如下所示：
   DBL = 3 次传输，DBA = 0xE。
3. 使能 TIMx 更新 DMA 请求（在 DIER 寄存器中设置 UDE 位）。
4. 使能 TIMx
5. 使能 DMA 通道

此示例适用于每个 CCRx 寄存器都需要更新一次的情况。例如，如果要更新每个 CCRx 寄存器两

次，则要传输的数据数量应为 6。让我们以一个在 RAM 中包含 data1，data2，data3，data4，data5 和 data6 的缓冲区为例。数据按照以下方式传输到 CCRx 寄存器：在第一个更新 DMA 请求时，data1 传输到 CCR2，data2 传输到 CCR3，data3 传输到 CCR4；在第二个更新 DMA 请求时，data4 传输到 CCR2，data5 传输到 CCR3，data6 传输到 CCR4。

*注：可以将空值写入保留的寄存器。*

### 22.3.33 TIM1/TIM8 DMA 请求

TIM1/TIM8 以生成 DMA 请求，如下表所示。

表 22.18 DMA 请求

| DMA 请求信号 | DMA 缩写 | DMA 请求 | 使能控制位 |
|---|---|---|---|
| tim_upd_dma | TIM_UP | 更新 | UDE |
| tim_cc1_dma | TIM_CH1 | 捕获/比较 1 | CC1DE |
| tim_cc2_dma | TIM_CH2 | 捕获/比较 2 | CC2DE |
| tim_cc3_dma | TIM_CH3 | 捕获/比较 3 | CC3DE |
| tim_cc4_dma | TIM_CH4 | 捕获/比较 4 | CC4DE |
| tim_com_dma | TIM_COM | 换向（COM） | COMDE |
| tim_trg_dma | TIM_TRIG | 触发 | TDE |

### 22.3.34 调试模式

当微控制器进入调试模式（内核停止）时，TIMx 计数器的行为取决于 DBG 模块中的 DBG_TIMx_STOP 配置位，它可以继续正常工作或停止。

在调试模式下，每个定时器的行为可以通过 Debug 支持（DBG）模块的专用配置位进行编程。

出于安全考虑，当计数器停止时，输出被禁用（就像将 MOE 位复位一样）。输出可以强制处于无效状态（OSSI 位=1），或者由 GPIO 控制器接管控制（OSSI 位=0）以强制其处于高阻态。

有关更多详细信息，请参阅调试部分。

## 22.4 TIM1/TIM8 低功耗模式

表 22.19 低功耗模式对 TIM1/TIM8 的影响

| 模式 | 描述 |
|---|---|
| 睡眠 | 无影响，外设是运行的。中断会导致设备退出睡眠模式。 |
| 停机 | 定时器操作被停止并保留其寄存器内容。无法生成中断。 |
| 待机 | 定时器被断电并且退出待机模式后必须重新初始化。 |

## 22.5 TIM1/TIM8 中断

TIM1/TIM8 可以生成多个中断，如下表所示。

表 22.20 中断请求

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机和待机退出 |
|---|---|---|---|---|---|---|
| TIM_UP | 更新 | UIF | UIE | UIF 写 0 | 是 | 否 |
| TIM_CC | 捕获/比较 1 | CC1IF | CC1IE | CC1IF 写 0 | 是 | 否 |

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机和待机退出 |
|---|---|---|---|---|---|---|
| | 捕获/比较 2 | CC2IF | CC2IE | CC2IF 写 0 | 是 | 否 |
| | 捕获/比较 3 | CC3IF | CC3IE | CC3IF 写 0 | 是 | 否 |
| | 捕获/比较 4 | CC4IF | CC4IE | CC4IF 写 0 | 是 | 否 |
| TIM_TRG_COM | 换向 (COM) | COMIF | COMIE | COMIF 写 0 | 是 | 否 |
| | 触发 | TIF | TIE | TIF 写 0 | 是 | 否 |
| TIM_DIR_IDX | 索引 | IDXF | IDXIE | IDXF 写 0 | 是 | 否 |
| | 方向 | DIRF | DIRIE | DIRF 写 0 | 是 | 否 |
| TIM_BRK | 刹车 | BIF | BIE | BIF 写 0 | 是 | 否 |
| | 刹车 2 | B2IF | | B2IF 写 0 | 是 | 否 |
| | 系统中断 | SBIF | | SBIF 写 0 | 是 | 否 |
| TIM_IERR | 索引错误 | IERRF | IERRIE | IERRF 写 0 | 是 | 否 |
| TIM_TER | 转换错误 | TERRF | TERRIE | TERRF 写 0 | 是 | 否 |

## 22.6 TIM1/TIM8 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。

### 22.6.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 1，8）

偏移地址：0x000

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | 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 15:13 保留，必须保持复位值。

Bit 12 **DITHEN**：使能抖动（Dithering enable）

0：抖动禁止

1：抖动使能

*注：*DITHEN *位只能在*CEN *位复位时修改*

Bit 11 **UIFREMAP**：UIF 状态位重映射（UIF status bit remapping）

0：无重映射。UIF 状态位未复制到 TIMx_CNT 寄存器位 31

1：重映射使能。UIF 状态位复制到 TIMx_CNT 寄存器位 31

Bit 10 保留，必须保持复位值。

Bit 9:8 **CKD[1:0]**：时钟分频（Clock division）

此位域指示定时器时钟（tim_ker_ck）频率与死区发生器以及数字滤波器（tim_etr_in, tim_tix）所使用的死区及采样时钟（$t_{DTS}$）之间的分频比

00：$t_{DTS} = t_{tim_ker_ck}$

01：$t_{DTS} = 2 \times t_{tim_ker_ck}$

10：$t_{DTS} = 4 \times t_{tim_ker_ck}$

11：保留，不要设置成此值

Bit 7 **ARPE**：自动重载预装载使能（Auto-reload preload enable）

0：TIMx_ARR 寄存器不进行缓冲；

1：TIMx_ARR 寄存器进行缓冲。

Bit 6:5 **CMS[1:0]**：中心对齐模式选择（Center-aligned mode selection）

00：边沿对齐模式。计数器根据方向位（DIR）递增计数或递减计数。

01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

*注：只要计数器处于使能状态* (CEN=1)，*就不得从边沿对齐模式切换为中心对齐模式。*

Bit 4 **DIR**：方向（Direction）

0：计数器递增计数；

1：计数器递减计数。

*注：当定时器配置为中心对齐模式或编码器模式时，该位为只读状态。*

Bit 3 **OPM**：单脉冲模式（One pulse mode）

0：计数器在发生更新事件时不会停止计数；

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

Bit 2 **URS**：更新请求源（Update request source）

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

Bit 1 **UDIS**：更新禁止（Update disable）

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新（UEV）事件可通过以下事件之一生成：

- 计数器上出/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

Bit 0 **CEN**：计数器使能（Counter enable）

0：禁止计数器；

1：使能计数器。

*注：只有事先通过软件将*CEN *位置*1*，才可以使用外部时钟，门控模式和编码器模式。而触发模式可通过硬件自动将*CEN *位置*1*。*

### 22.6.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 1，8）

偏移地址：0x004

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | 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 31:26 保留，必须保持复位值。

Bit 24 保留，必须保持复位值。

Bit 23:20 **MMS2[3:0]**：主模式选择 2（Master mode selection 2）

这些位允许将信息发送到 ADC 进行同步（tim_trgo2）以进行选择。组合如下：

0000：**重置** - 来自 TIMx_EGR 寄存器的 UG 位用作触发器输出（tim_trgo2）。如果重置是由触发器输入（配置为重置模式的从机模式控制器）生成的，则 tim_trgo2 上的信号与实际重置相比会有所延迟。

0001：**使能** - 计数器使能信号 CNT_EN 用作触发器输出（tim_trgo2）。它对于同时启动多个定时器或在一段时间内控制从机定时器的使能窗口是有用的。当在门控模式下配置时，计数器使能信号是由 CEN 控制位和触发器输入之间的逻辑 AND 生成的。当计数器使能信号由触发器输入控制时，除选择了主/从模式的情况外（请参见 TIMx_SMCR 寄存器中的 MSM 位描述），tim_trgo2 上会有延迟。

0010：**更新** - 选择更新事件作为触发器输出（tim_trgo2）。例如，主定时器可以用作从机定时器的预分频器。

0011：**比较脉冲** - 当 CC1IF 标志被设置时（即使它已经是高电平），当发生捕获或比较匹配时，触发输出会发送一个正脉冲（tim_trgo2）。

0100：**比较** - tim_oc1refc 信号用作触发器输出（tim_trgo2）

0101：**比较** - tim_oc2refc 信号用作触发器输出（tim_trgo2）

0110：**比较** - tim_oc3refc 信号用作触发器输出（tim_trgo2）

0111：**比较** - tim_oc4refc 信号用作触发器输出（tim_trgo2）

1000：**比较** - tim_oc5refc 信号用作触发器输出（tim_trgo2）

1001：**比较** - tim_oc6refc 信号用作触发器输出（tim_trgo2）

1010：**比较脉冲** - tim_oc4refc 上升或下降沿在 tim_trgo2 上生成脉冲

1011：**比较脉冲** - tim_oc6refc 上升或下降沿在 tim_trgo2 上生成脉冲

1100：**比较脉冲** - tim_oc4refc 或 tim_oc6refc 上升沿在 tim_trgo2 上生成脉冲

1101：**比较脉冲** - tim_oc4refc 上升或 tim_oc6refc 下降沿在 tim_trgo2 上生成脉冲

1110：**比较脉冲** - tim_oc5refc 或 tim_oc6refc 上升沿在 tim_trgo2 上生成脉冲

1111：**比较脉冲** - tim_oc5refc 上升或 tim_oc6refc 下降沿在 tim_trgo2 上生成脉冲

*注：在从主定时器接收事件之前，必须使能从机定时器或* ADC *的时钟，并且在从主定时器接收触发器时不能随意更改。*

*注：若选* tim_ocxrefc *为* tim_trgo2 *输出，当* tim_ocref_clr_int *清零信号生效时，* tim_ocxrefc *信号将保持低电平，* tim_trgo2 *也不会有触发输出，直到发生下一更新事件（*UEV*）（特别注意，当* TIMx_RCR>0 *时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。*

Bit 19 保留，必须保持复位值。

Bit 18 **OIS6**：输出空闲状态 6（OC6 输出）
参见 OIS1 位

Bit 17 保留，必须保持复位值。

Bit 16 **OIS5**：输出空闲状态 5（OC5 输出）
参见 OIS1 位

Bit 15 **OIS4N**：输出空闲状态 4（OC4N 输出）
参见 OIS1 位

Bit 14 **OIS4**：输出空闲状态 4（OC4 输出）
参见 OIS1 位

Bit 13 **OIS3N**：输出空闲状态 3（OC3N 输出）
参见 OIS1 位

Bit 12 **OIS3**：输出空闲状态 3（OC3 输出）
参见 OIS1 位

Bit 11 **OIS2N**：输出空闲状态 2（OC2N 输出）
参见 OIS1 位

Bit 10 **OIS2**：输出空闲状态 2（OC2 输出）
参见 OIS1 位

Bit 9 **OIS1N**：输出空闲状态 1（OC1N 输出）
0：当 MOE=0 时，经过死区时间后 tim_oc1n=0
1：当 MOE=0 时，经过死区时间后 tim_oc1n=1

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别* 1、2 *或*3，*此位即无法修改。*

Bit 8 **OIS1**：输出空闲状态 1（OC1 输出）
0：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=0
1：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=1

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别* 1、2 *或*3，*此位即无法修改。*

Bit 7 **TI1S**：tim_ti1 选择
0：tim_ti1_in[15:0]多路复用器输出被连接至 tim_ti1 输入
1：tim_ti1_in[15:0]，tim_ti2_in[15:0]和 tim_ti3_in[15:0]多路复用器输出被异或并连接至 tim_ti1 输入

Bit 25, 6:4 **MMS[3:0]**：主模式选择
这些位可选择主模式下将要发送到从定时器以实现同步的信息（tim_trgo）。这些位的组合如下：
0000：**复位** – TIMx_EGR 寄存器中的 UG 位用作触发输出（tim_trgo）。如果复位由触发输入生成（从模式控制器配置为复位模式），则 tim_trgo 上的信号相比实际复位会有延迟。

0001：**使能** – 计数器使能信号 CNT_EN 用作触发输出（tim_trgo）。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由 CEN 控制位产生。当配置为门控模式时，也可由触发输入产生。当计数器使能信号由触发输入控制时，tim_trgo 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明）。

0010：**更新** – 选择更新事件作为触发输出（tim_trgo）。例如，主定时器可用作从定时器的预分频器。

0011：**比较脉冲** – 一旦发生比较匹配事件，当 CC1IF 被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(tim_trgo)。

0100：**比较** – tim_oc1refc 信号用作触发输出（tim_trgo）

0101：**比较** – tim_oc2refc 信号用作触发输出（tim_trgo）

0110：**比较** – tim_oc3refc 信号用作触发输出（tim_trgo）

0111：**比较** – tim_oc4refc 信号用作触发输出（tim_trgo）

1000：**编码器时钟输出** – 编码器时钟信号被用于触发器输出（tim_trgo）。此编码对以下 SMS[3:0]值是有效的：0001，0010，0011，1010，1011，1100，1101，1110，1111。任何其他的 SMS[3:0]编码是不被允许的，这可能导致不可预知的行为。

其他编码保留

*注：在从主定时器接收事件之前，必须使能从机定时器或 ADC 的时钟，并且在从主定时器接收触发器时不能随意更改。*

*注：若选* tim_ocxrefc *为* tim_trgo *输出，当* tim_ocref_clr_int *清零信号生效时，* tim_ocxrefc *信号将保持低电平，* tim_trgo *也不会有触发输出，直到发生下一更新事件（UEV）（特别注意，当* TIMx_RCR>0 *时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。*

Bit 3 **CCDS**：捕获/比较 DMA 选择（Capture/compare DMA selection）

0：发生 CCx 事件时发送 CCx DMA 请求

1：发生更新事件时发送 CCx DMA 请求

Bit 2 **CCUS**：捕获/比较控制更新选择

0：如果捕获/比较控制位（CCPC=1）进行预装载，仅通过将 COMG 位置 1 来对这些位进行更新；

1：如果捕获/比较控制位（CCPC=1）进行预装载，可通过将 COMG 位置 1 或 tim_trgi 的上升沿对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。。*

Bit 1 保留，必须保持复位值。

Bit 0 **CCPC**：捕获/比较预装载控制

0：CCxE，CCxNE 和 OCxM 位未进行预装载；

1：CCxE，CCxNE 和 OCxM 位进行了预装载，写入这些位后，仅当发生换向事件（COM）（COMG 位置 1 或在 tim_trgi 上检测到上升沿，取决于 CCUS 位）时才会对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。*

### 22.6.3 TIMx 从模式控制寄存器（TIMx_SMCR）（x = 1，8）

偏移地址：0x008

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | SMSPS | SMSPE | Reserved | | TS[4:3] | | Reserved | | | 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 31:26 保留，必须保持复位值。

Bit 25 **SMSPS**：SMS 预加载源（SMS preload source）

此位选择触发 SMS[3:0]位域从预加载到活动的事件来源：

0：由定时器的更新事件触发

1：由索引事件触发

Bit 24 **SMSPE**：SMS 预加载使能（SMS preload enable）

此位选择是否使能 SMS[3:0]位域的预加载：

0：SMS[3:0]位域不进行预加载

1：使能 SMS[3:0]预加载

Bit 23:22 保留，必须保持复位值。

Bit 21:20 **TS[4:3]**：触发器选项- 位 4:3（Trigger selection - bit 4:3）

参考 TS[2:0]的描述 - 位 6:4

Bit 19:17 保留，必须保持复位值。

Bit 16 **SMS[3]**：参考 SMS[2:0]的描述

Bit 15 **ETP**：外部触发极性（External trigger polarity）

此位可选择将 tim_etr_in 还是 tim_etr_in 的反相用于触发操作

0：tim_etr_in 未反相，高电平或上升沿有效。

1：tim_etr_in 反相，低电平或下降沿有效。

Bit 14 **ECE**：外部时钟使能位（External clock enable）

此位可使能外部时钟模式 2。

0：禁止外部时钟模式 2

1：使能外部时钟模式 2。计数器时钟由 tim_etrf 信号的任意有效边沿提供。

*注：将* ECE *位置*1 *与选择外部时钟模式*1 *并将* tim_trgi *连接到* tim_etrf *（*SMS=111 *且* TS=00111*）具有相同效果。*

*外部时钟模式*2 *可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下* tim_trgi *不得连接* tim_etrf *（*TS *位不得为*00111*）。*

*如果同时使能外部时钟模式*1 *和外部时钟模式*2*，则外部时钟输入为* tim_etrf*。*

Bit 13:12 **ETPS[1:0]**：外部触发预分频器（External trigger prescaler）

外部触发信号 tim_etrp 频率不得超过 TIMxCLK 频率的 1/4。可通过使能预分频器来降低 tim_etrp 频率。这种方法在 tim_etr_in 输入快速外部时钟时非常有用。

00：预分频器关闭

01：2 分频 tim_etr_in 频率

10：4 分频 tim_etr_in 频率

11：8 分频 tim_etr_in 频率

Bit 11:8 **ETF[3:0]**：外部触发滤波器（External trigger filter）

此位域可定义 tim_etrp 信号的采样频率和适用于 tim_etrp 的数字滤波时间。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{CK_INT}$，N=2。

0010：$f_{SAMPLING}=f_{CK_INT}$，N=4。

0011：$f_{SAMPLING}=f_{CK_INT}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 7 **MSM：**主/从模式（Master/slave mode）

0：不执行任何操作；

1：当前定时器的触发输入事件（tim_trgi）的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 tim_trgo）。此设置适用于由单个外部事件对多个定时器进行同步的情况。

Bit 6:4 **TS[2:0]：**触发选择（Trigger selection）

此位域和位 TS[4:3]组合

此位域可选择将要用于同步计数器的触发输入。

00000：内部触发 0（tim_itr0）

00001：内部触发 1（tim_itr1）

00010：内部触发 2（tim_itr2）

00011：内部触发 3（tim_itr3）

00100：tim_ti1 边沿检测器（tim_ti1f_ed）

00101：滤波后的定时器输入 1（tim_ti1fp1）

00110：滤波后的定时器输入 2（tim_ti2fp2）

00111：外部触发输入（tim_etrf）

01000：内部触发器 4 (tim_itr4)

01001：内部触发器 5 (tim_itr5)

01010：内部触发器 6 (tim_itr6)

01011：内部触发器 7 (tim_itr7)

01100：内部触发器 8 (tim_itr8)

01101：内部触发器 9 (tim_itr9)

01110：内部触发器 10 (tim_itr10)

01111：内部触发器 11 (tim_itr11)

10000：内部触发器 12 (tim_itr12)

10001：内部触发器 13 (tim_itr13)

10010：内部触发器 14 (tim_itr14)

10011：内部触发器 15 (tim_itr15)

其他：保留

有关定时器 ITRx 含义的详细信息，请参见章节 TIMx 引脚和内部信号中的表格：TIMx 内部触发连接。

*注：这些位只能在未使用的情况下（例如，*SMS=000 *时）进行更改，以避免转换时出现错误的边沿检测。*

Bit 3 **OCCS：**OCREF 清除源选择（OCREF clear selection）

此位用于选择 OCREF 清除源。

0：tim_ocref_clr_int 信号连接到 tim_ocref_clr 输入

1：tim_ocref_clr_int 信号连接到 tim_etrf

Bit 16, 2:0 **SMS[2:0]**：从模式选择（Slave mode selection）

当选择了外部信号，触发信号（tim_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：编码器模式 1，X2 模式 – 计数器根据 tim_ti1fp1 电平在 tim_ti2fp2 边沿递增/递减计数。

0010：编码器模式 2，X2 模式– 计数器根据 tim_ti2fp2 电平在 tim_ti1fp1 边沿递增/递减计数。

0011：编码器模式 3，X4 模式 – 计数器在 tim_ti1fp1 和 tim_ti2fp2 的边沿计数，计数的方向取决于另外一个信号的电平。

0100：复位模式 – 选中的触发输入（tim_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（tim_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 tim_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（tim_trgi）的上升沿驱动计数器。

1000：组合复位+触发模式 - 选定触发器输入（tim_trgi）的上升沿会重新初始化计数器，生成寄存器的更新并启动计数器。

1001：组合门控+复位模式 - 当触发器输入（tim_trgi）为高电平时，计数器时钟使能。一旦触发器变为低电平，计数器将停止并重置。同时控制计数器的启动和停止。

1010：编码器模式：时钟加方向，x2 模式。

1011：编码器模式：时钟加方向，x1 模式，CC2P 设置 tim_ti2fp2 边沿敏感度。

1100：编码器模式：方向时钟，x2 模式。

1101：编码器模式：方向时钟，x1 模式，CC1P 和 CC2P 设置 tim_ti1fp1 和 tim_ti2fp2 边沿敏感度。

1110：四象限编码器模式：x1 模式，仅计算 tim_ti1fp1 边沿，CC1P 设置边沿敏感度。

1111：四象限编码器模式：x1 模式，仅计算 tim_ti2fp2 边沿，CC2P 设置边沿敏感度。

*注：如果选择* tim_ti1f_ed *作为触发输入（*TS=00100*），则不得使用门控模式。实际上* tim_ti1f_ed *为* TI1F *上的每个过渡输出* 1 *个脉冲，而门控模式检查触发信号的电平。*

*注：接收* tim_trgo *或* tim_trgo2 *信号的从机外设（定时器、*ADC *等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能随意更改。*

### 22.6.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 1，8）

偏移地址：0x00C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TERRIE | IERRIE | DIRIE | IDXIE | Reserved | | | |
| | | | | | | | | 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 31:24 保留，必须保持复位值。

Bit 23 **TERRIE**：转换错误中断使能（Transition error interrupt enable）

0：禁止转换错误中断；

1：使能转换错误中断。

Bit 22 **IERRIE**：索引错误中断使能（Index error interrupt enable）

0：禁止索引错误中断；

1：使能索引错误中断。

Bit 21 **DIRIE**：方向改变中断使能（Direction change interrupt enable）

0：禁止方向改变中断；

1：使能方向改变中断。

Bit 20 **IDXIE**：索引中断使能（Index interrupt enable）

0：禁止索引中断；

1：使能索引中断。

Bit 19:15 保留，必须保持复位值。

Bit 14 **TDE**：触发 DMA 请求使能（Trigger DMA request enable）

0：禁止触发 DMA 请求；

1：使能触发 DMA 请求。

Bit 13 **COMDE**：换向 DMA 请求使能（COM DMA request enable）

0：禁止换向 DMA 请求；

1：使能换向 DMA 请求。

Bit 12 **CC4DE**：捕获/比较 4 DMA 请求使能（Capture/Compare 4 DMA request enable）

0：禁止 CC4 DMA 请求；

1：使能 CC4 DMA 请求。

Bit 11 **CC3DE**：捕获/比较 3 DMA 请求使能（Capture/Compare 3 DMA request enable）

0：禁止 CC3 DMA 请求；

1：使能 CC3 DMA 请求。

Bit 10 **CC2DE**：捕获/比较 2 DMA 请求使能（Capture/Compare 2 DMA request enable）

0：禁止 CC2 DMA 请求；

1：使能 CC2 DMA 请求。

Bit 9 **CC1DE**：捕获/比较 1 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止 CC1 DMA 请求；

1：使能 CC1 DMA 请求。

Bit 8 **UDE**：更新 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止更新 DMA 请求；

1：使能更新 DMA 请求。

Bit 7 **BIE**：刹车中断使能（Break interrupt enable）

0：禁止刹车中断；

1：使能刹车中断。

Bit 6 **TIE**：触发信号（TGRI）中断使能（Trigger interrupt enable）

0：禁止触发信号（TGRI）中断；

1：使能触发信号（TGRI）中断。

Bit 5 **COMIE**：COM 中断使能（COM interrupt enable）

0：禁止 COM 中断；

1：使能 COM 中断。

Bit 4 **CC4IE**：捕获/比较 4 中断使能（Capture/Compare 4 interrupt enable）

0：禁止 CC4 中断；

1：使能 CC4 中断。

Bit 3 **CC3IE**：捕获/比较 3 中断使能（Capture/Compare 3 interrupt enable）

0：禁止 CC3 中断；

1：使能 CC3 中断。

Bit 2 **CC2IE**：捕获/比较 2 中断使能（Capture/Compare 2 interrupt enable）

0：禁止 CC2 中断；

1：使能 CC2 中断。

Bit 1 **CC1IE**：捕获/比较 1 中断使能（Capture/Compare 1 interrupt enable）

0：禁止 CC1 中断；

1：使能 CC1 中断。

Bit 0 **UIE**：更新中断使能（Update interrupt enable）

0：禁止更新中断；

1：使能更新中断。

### 22.6.5 TIMx 状态寄存器（TIMx_SR）（x = 1，8）

偏移地址：0x010

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TERRF | IERRF | DIRF | IDXF | Reserved | | CC6IF | CC5IF |
| | | | | | | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rc_w0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SBIF | CC4OF | CC3OF | CC2OF | CC1OF | B2IF | BIF | TIF | COMIF | 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 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

Bit 31:24 保留，必须保持复位值。

Bit 23 **TERRF**：转换错误中断标志（Transition error interrupt flag）

当在编码器模式下检测到转换错误时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未检测到编码器转换错误

1：已检测到编码器转换错误

Bit 22 **IERRF**：索引错误中断标志（Index error interrupt flag）

当检测到索引错误时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未检测到索引错误

1：已检测到索引错误

Bit 21 **DIRF**：方向改变中断标志（Direction change interrupt flag）

当编码器模式中的方向发生变化时（TIMx_CR 中的 DIR 位值正在改变），硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：方向未改变

1：方向已改变

Bit 20 **IDXF**：索引中断标志（Index interrupt flag）

当检测到索引事件时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未发生索引事件

1：已发生索引事件

Bit 19:18 保留，必须保持复位值。

Bit 17 **CC6IF**：比较 6 中断标志

参考 CC1IF 描述

*注：通道 6 只能被配置为输出。*

Bit 16 **CC5IF**：比较 5 中断标志

参考 CC1IF 描述

*注：通道 5 只能被配置为输出。*

Bit 15:14 保留，必须保持复位值。

Bit 13 **SBIF**：系统刹车中断标志（System break interrupt flag）

该标志由硬件在系统刹车输入激活时立即设置。如果系统刹车输入不处于有效状态，则可以通过软件清除该标志。

要重新启动 PWM 操作，必须将此标志重置为 0。

0：未发生刹车事件。

1：在系统刹车输入上检测到有效电平。如果 TIMx_DIER 寄存器中的 BIE=1，则生成中断。

Bit 12 **CC4OF**：捕获/比较 4 重复捕获标志

参考 CC1OF 的描述

Bit 11 **CC3OF**：捕获/比较 3 重复捕获标志

参考 CC1OF 的描述

Bit 10 **CC2OF**：捕获/比较 2 重复捕获标志

参考 CC1OF 的描述

Bit 9 **CC1OF**：捕获/比较 1 重复捕获标志

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获

1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

Bit 8 **BI2F**：刹车 2 中断标志（Break 2 interrupt flag）

只要刹车 2 输入变为有效状态，此标志便由硬件置 1。刹车 2 输入无效后可通过软件对其清零。

0：无刹车事件产生；

1：刹车 2 输入上检测到有效电平。如果 TIMx_DIER 寄存器上 BIE=1 则产生一个中断。

Bit 7 **BIF**：刹车中断标志（Break interrupt flag）

只要刹车输入变为有效状态，此标志便由硬件置 1。刹车输入无效后可通过软件对其清零。

0：无刹车事件产生；

1：刹车输入上检测到有效电平。如果 TIMx_DIER 寄存器上 BIE=1 则产生一个中断。

Bit 6 **TIF**：触发中断标志（Trigger interrupt flag）

在除门控模式以外的所有模式下，当使能从模式控制器后在 tim_trgi 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。

0：未发生触发事件。

1：触发中断挂起。

Bit 5 **COMIF**：COM 中断标志（COM interrupt flag）

此标志在发生 COM 事件时（捕获/比较控制位 CCxE、CCxNE 和 OCxM 已更新时）由硬件置 1。但需要通过软件清零。

0：未发生 COM 事件。

1：COM 中断挂起。

Bit 4 **CC4IF**：捕获/比较 4 中断标志（Capture/Compare 4 interrupt flag）

参考 CC1IF 描述。

Bit 3 **CC3IF**：捕获/比较 3 中断标志（Capture/Compare 3 interrupt flag）

参考 CC1IF 描述。

Bit 2 **CC2IF**：捕获/比较 2 中断标志（Capture/Compare 2 interrupt flag）

参考 CC1IF 描述。

Bit 1 **CC1IF**：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag）

此标志由硬件设置，它可以通过软件（输入捕获或输出比较模式）或读取 TIMx_CCR1 寄存器（仅输入捕获模式）来清除。

0：无比较匹配/无输入捕获发生

1：发生比较匹配或输入捕获

**如果通道 CC1 配置为输出：**当计数器 TIMx_CNT 的内容与 TIMx_CCR1 寄存器内容匹配时，会设置此标志。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。在中心对齐模式下有 3 种可能的标志设置选项，请参考 TIMx_CR1 寄存器中的 CMS 位以获取完整描述。

**如果通道 CC1 配置为输入：**当在 TIMx_CCR1 寄存器中捕获到计数器值时（在 TIMx_CCER 中定义的 CC1P 和 CC1NP 位设置所确定的 IC1 上检测到边沿时），会设置此位。

Bit 0 **UIF**：更新中断标志（Update interrupt flag）

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– TIMx_CR1 寄存器中的 UDIS=0，并且重复计数器值上溢或下溢时（重复计数器= 0 时更新）。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（请参见章节：TIMx 从模式控制寄存器（TIMx_SMCR）（x=1，8））。

### 22.6.6 TIMx 事件产生寄存器（TIMx_EGR）（x = 1，8）

偏移地址：0x014

复位：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | CC6G | CC5G | B2G | BG | TG | COMG | CC4G | CC3G | CC2G | CC1G | UG |
| | | | | | w | w | w | w | w | w | w | w | w | w | w |

Bit 15:11 保留，必须保持复位值。

Bit 10 **CC6G**：捕获/比较 6 事件生成（Capture/Compare 6 generation）

参考 CC1G 描述。

Bit 9 **CC5G**：捕获/比较 5 事件生成（Capture/Compare 5 generation）

参考 CC1G 描述。

Bit 8 **B2G**：刹车 2 生成（Break generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：生成刹车 2 事件。MOE 位清零且 BIF 标志置 1。使能后可发生相关中断事件或 DMA 传输事件。

Bit 7 **BG**：刹车生成（Break generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：生成刹车事件。MOE 位清零且 BIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

Bit 6 **TG**：生成触发信号（Trigger generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

Bit 5 **COMG**：捕获/比较控制更新生成（Capture/Compare control update generation）

该位可通过软件置 1，并由硬件自动清零

0：不执行任何操作；

1：CCPC 位置 1 时，可更新 CCxE、CCxNE 和 OCxM 位。

*注：此位仅对具有互补输出的通道有效。*

Bit 4 **CC4G**：捕获/比较 4 生成（Capture/Compare 4 generation）

参考 CC1G 描述。

Bit 3 **CC3G**：捕获/比较 3 生成（Capture/Compare 3 generation）

参考 CC1G 描述。

Bit 2 **CC2G**：捕获/比较 2 生成（Capture/Compare 2 generation）

参考 CC1G 描述。

Bit 1 **CC1G**：捕获/比较 1 生成（Capture/Compare 1 generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：通道 1 上生成捕获/比较事件：

**若通道 CC1 配置为输出：**

使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。

**如果通道 CC1 配置为输入：**

TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

Bit 0 **UG**：更新生成（Update generation）

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作；

1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。如果选择中心对齐模式或 DIR=0（递增计数），计数器将清零；如果 DIR=1（递减计数），计数器将使用自动重载值（TIMx_ARR）。

### 22.6.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 1，8）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IC2F[3:0] | | | | IC2PSC[1:0] | | CC2S[1:0] | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:16 保留，始终读为 0。

Bit 15:12 **IC2F[3:0]：**输入捕获 2 滤波器（Input capture 2 filter）

Bit 11:10 **IC2PSC[1:0]：**输入/捕获 2 预分频器（input capture 2 prescaler）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择 （Capture/compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道被配置为输出；

01：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti2 上；

10：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti1 上；

11：CC2 通道被配置为输入，tim_ic2 映射在 tim_trc 上。此模此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIMx_CCER *中的* CC2E = 0*），才可向* CC2S *位写入数据。*

Bit 7:4 **IC1F[3:0]：**输入捕获 1 滤波器（Input capture 1 filter）

此位域可定义 tim_ti1 输入的采样频率和适用于 tim_ti1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 3:2 **IC1PSC[1:0]：**输入/捕获 1 预分频器（Input capture 1 prescaler）

此位域定义 CC1 输入 (tim_ic1) 的预分频比。

只要 CC1E=0（TIMx_CCER 寄存器），预分频器便立即复位。

00：无预分频器，捕获输入上每检测到一个边沿便执行捕获

01：每发生 2 个事件便执行一次捕获；

10：每发生 4 个事件便执行一次捕获；

11：每发生 8 个事件便执行一次捕获。

Bit 1:0 **CC1S[1:0]：**捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道被配置为输出；

01：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti1 上；

10：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti2 上；

11：CC1 通道被配置为输入，tim_ic1 映射在 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。*

### 22.6.8 TIMx 捕获/比较模式寄存器 1[复用]（TIMx_CCMR1）（x = 1，8）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC2M[3] | Reserved | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式**

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC2CE：**输出比较 2 清零使能（Output Compare 2 clear enable）

Bit 24, 14:12 **OC2M[3:0]：**输出比较 2 模式（Output Compare 2 mode）

Bit 11 **OC2PE：**输出比较 2 预装载使能（Output Compare 2 preload enable）

Bit 10 **OC2FE：**输出比较 2 快速使能（Output Compare 2 fast enable）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择（Capture/Compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，tim_ic2 映射到 tim_ti2 上

10：CC2 通道配置为输入，tim_ic2 映射到 tim_ti1 上

11：CC2 通道配置为输入，tim_ic2 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据*

Bit 7 **OC1CE：**输出比较 1 清零使能（Output Compare 1 clear enable）

0：tim_oc1ref 不受 tim_ocref_clr_int 信号影响；

1：tim_oc1ref 在 tim_ocref_clr_int 信号中被检测到高电平，tim_oc1ref 立即清零。（tim_ocref_clr 输入或 tim_etrf 输入）

Bit 16, 6:4 **OC1M[3:0]：**输出比较 1 模式（Output Compare 1 mode）

这些位定义提供 tim_oc1 和 tim_oc1n 的输出参考信号 tim_oc1ref 的行为。tim_oc1ref 为高电平有效，而 tim_oc1 和 tim_oc1n 的有效电平则取决于 CC1P 位和 CC1NP 位。

0000：**冻结** - 输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

0001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为高电平。

0010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为低电平。

0011：**翻转** - TIMx_CNT=TIMx_CCR1 时，tim_oc1ref 发生翻转。

0100：**强制变为无效电平** - tim_oc1ref 强制变为低电平。

0101：**强制变为有效电平** - tim_oc1ref 强制变为高电平。

0110：**PWM 模式 1**-在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态（tim_oc1ref=0），否则为有效状态（tim_oc1ref=1）。

0111：**PWM 模式 2** - 在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

1000：**可重触发 OPM 模式 1**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于无效状态。

1001：**可重触发 OPM 模式 2**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 2 那样执行比较，并在下次更新时使通道再次处于无效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。

1010：保留。

1011：保留。

1100：**组合 PWM 模式 1**- tim_oc1ref 的行为与 PWM 模式 1 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑 OR。

1101：**组合 PWM 模式 2**- tim_oc1ref 的行为与 PWM 模式 2 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑 AND。

1110：**非对称 PWM 模式 1**- tim_oc1ref 的行为与 PWM 模式 1 相同。当计数器向上计数时，tim_oc1refc 输出 tim_oc1ref；当计数器向下计数时，tim_oc1refc 输出 tim_oc2ref。

1111：**非对称 PWM 模式 2**- tim_oc1ref 的行为与 PWM 模式 2 相同。当计数器向上计数时，tim_oc1refc 输出 tim_oc1ref；当计数器向下计数时，tim_oc1refc 输出 tim_oc2ref。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*3 *且*CC1S= *“*00*”（通道配置为输出），这些位即无法修改。*

*注：在* PWM *模式*1 *或*PWM *模式*2 *下，仅当比较结果发生改变或输出比较模式由“冻*

*结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。*

*注：此位域将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的 CCPC 位置 1，则仅当生成 COM 事件时，OC1M 有效位才会从预装载位获取新值。*

Bit 3 **OC1PE**：输出比较 1 预装载使能（Output Compare 1 preload enable）

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

*注：只要编程了 LOCK（TIMx_BDTR 寄存器中的 LOCK 位）级别 3 且 CC1S=“00”（通道配置为输出），这些位即无法修改。*

Bit 2 **OC1FE**：输出比较 1 快速使能（Output Compare 1 fast enable）

此位减少了触发事件和定时器输出上的转换之间的延迟。它必须在单脉冲模式（在 TIMx_CR1 寄存器中设置 OPM 位）中使用，以便在启动触发后尽快开始输出脉冲。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

Bit 1:0 **CC1S[1:0]**：捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，tim_ic1 映射到 tim_ti1 上

10：CC1 通道配置为输入，tim_ic1 映射到 tim_ti2 上

11：CC1 通道配置为输入，tim_ic1 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据*

### 22.6.9 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 1，8）

偏移地址：0x01C

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC4F[3:0] | | | | IC4PSC[1:0] | | CC4S[1:0] | | IC3F[3:0] | | | | IC3PSC[1:0] | | CC3S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:16 保留，始终读为 0。

Bit 15:12 **IC4F[3:0]：**输入捕获 4 滤波器（Input capture 4 filter）

Bit 11:10 **IC4PSC[1:0]：**输入/捕获 4 预分频器（input capture 4 prescaler）

Bit 9:8 **CC4S[1:0]：**捕获/比较 4 选择 （Capture/compare 4 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC4 通道被配置为输出；

01：CC4 通道被配置为输入，tim_ic4 映射在 tim_ti4 上；

10：CC4 通道被配置为输入，tim_ic4 映射在 tim_ti3 上；

11：CC4 通道被配置为输入，tim_ic4 映射在 tim_trc 上。此模此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据。*

Bit 7:4 **IC3F[3:0]：**输入捕获 3 滤波器（Input capture 3 filter）

Bit 3:2 **IC3PSC[1:0]：**输入/捕获 1 预分频器（Input capture 3 prescaler）

Bit 1:0 **CC3S[1:0]：**捕获/比较 3 选择（Capture/Compare 3 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC3 通道被配置为输出；

01：CC3 通道被配置为输入，tim_ic3 映射在 tim_ti3 上；

10：CC3 通道被配置为输入，tim_ic3 映射在 tim_ti4 上；

11：CC3 通道被配置为输入，tim_ic3 映射在 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据。*

### 22.6.10 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 1，8）

偏移地址：0x01C

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC4M[3] | Reserved | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式：**

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC4CE：**输出比较 4 清零使能（Output Compare 4 clear enable）

Bit 24, 14:12 **OC4M[2:0]：**输出比较 4 模式（Output Compare 4 mode）

参考 OC3M[3:0]位的描述

Bit 11 **OC4PE：**输出比较 4 预装载使能（Output Compare 4 preload enable）

Bit 10 **OC4FE：**输出比较 4 快速使能（Output Compare 4 fast enable）

Bit 9:8 **CC4S[1:0]：**捕获/比较 4 选择（Capture/Compare 4 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC4 通道配置为输出

01：CC4 通道配置为输入，tim_ic4 映射到 tim_ti4 上

10：CC4 通道配置为输入，tim_ic4 映射到 tim_ti3 上

11：CC4 通道配置为输入，tim_ic4 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIMx_CCER *中的*CC4E = 0*），才可向*CC4S *位写入数据*

Bit 7 **OC3CE：**输出比较 3 清零使能（Output Compare 3 clear enable）

Bit 16, 6:4 **OC3M[3:0]：**输出比较 3 模式（Output Compare 3 mode）

这些位定义提供 tim_oc3 和 tim_oc3n 的输出参考信号 tim_oc3ref 的行为。tim_oc3ref 为高电平有效，而 tim_oc3 和 tim_oc3n 的有效电平则取决于 CC3P 位和 CC3NP 位。

0000：**冻结** - 输出比较寄存器 TIMx_CCR3 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

0001：将通道 3 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 3（TIMx_CCR3）匹配时，tim_oc3ref 信号强制变为高电平。

0010：将通道 3 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 3（TIMx_CCR3）匹配时，tim_oc3ref 信号强制变为低电平。

0011：**翻转** - TIMx_CNT=TIMx_CCR3 时，tim_oc3ref 发生翻转。

0100：**强制变为无效电平** - tim_oc3ref 强制变为低电平。

0101：**强制变为有效电平** - tim_oc3ref 强制变为高电平。

0110：**PWM 模式 1**-在递增计数模式下，只要 TIMx_CNT<TIMx_CCR3，通道 3 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR3，通道 3 便为无效状态（tim_oc3ref=0），否则为有效状态（tim_oc3ref=1）。

0111：**PWM 模式 2** - 在递增计数模式下，只要 TIMx_CNT<TIMx_CCR3，通道 3 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR3，通道 3 便为有效状态，否则为无效状态。

1000：**可重触发 OPM 模式 1**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于无效状态。

1001：**可重触发 OPM 模式 2**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 2 那样执行比较，并在下次更新时使通道再次处于无效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。

1010：**比较脉冲输出**：当 CCR3 匹配事件发生时，根据 TIMxECR 中的 PWPRSC[2:0]和 PW[7:0]位域编程，产生 tim_oc3ref 脉冲。

1011：**方向输出**。tim_oc3ref 信号被 DIR 位的副本覆盖。

1100：**组合 PWM 模式 1** - tim_oc3ref 的行为与 PWM 模式 1 相同。tim_oc3refc 是 tim_oc3ref 和 tim_oc4ref 之间的逻辑 OR。

1101：**组合 PWM 模式 2** - tim_oc3ref 的行为与 PWM 模式 2 相同。tim_oc3refc 是 tim_oc3ref 和 tim_oc4ref 之间的逻辑 AND。

1110：**非对称 PWM 模式 1** - tim_oc3ref 的行为与 PWM 模式 1 相同。当计数器向上计数时，tim_oc3refc 输出 tim_oc3ref；当计数器向下计数时，tim_oc3refc 输出 tim_oc4ref。

1111：**非对称 PWM 模式 2** - tim_oc3ref 的行为与 PWM 模式 2 相同。当计数器向上计数时，tim_oc3refc 输出 tim_oc3ref；当计数器向下计数时，tim_oc3refc 输出 tim_oc4ref。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*3 *且*CC1S= *“*00*”*

*（通道配置为输出），这些位即无法修改。*

*注：在 PWM 模式1 或PWM 模式2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。*

*注：此位域将在具有互补输出的通道上进行预装载。如果TIMx_CR2 寄存器中的CCPC 位置1，则仅当生成COM 事件时，OC3M 有效位才会从预装载位获取新值。*

Bit 3 **OC3PE：**输出比较 3 预装载使能（Output Compare 3 preload enable）

Bit 2 **OC3FE：**输出比较 3 快速使能（Output Compare 3 fast enable）

Bit 1:0 **CC3S[1:0]：**捕获/比较 3 选择（Capture/Compare 3 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC3 通道配置为输出

01：CC3 通道配置为输入，tim_ic3 映射到 tim_ti3 上

10：CC3 通道配置为输入，tim_ic3 映射到 tim_ti4 上

11：CC3 通道配置为输入，tim_ic3 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的CC3E = 0），才可向CC3S 位写入数据*

### 22.6.11 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 1，8）

偏移地址：0x020

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | CC6P | CC6E | Reserved | | 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 31:22 保留，必须保持复位值。

Bit 21 **CC6P：**输入/捕获 6 输出极性

参考 CC1P 的描述。

Bit 20 **CC6E：**输入/捕获 6 输出使能

参考 CC1E 的描述。

Bit 19:18 保留，必须保持复位值。

Bit 17 **CC5P**：输入/捕获 5 输出极性

参考 CC1P 的描述。

Bit 16 **CC5E：**输入/捕获 5 输出使能

参考 CC1E 的描述。

Bit 15 **CC4NP**：输入/捕获 4 互补输出极性

参考 CC1NP 的描述。

Bit 14 **CC4NE**：输入/捕获 4 互补输出使能

参考 CC1NE 的描述。

Bit 13 **CC4P**：输入/捕获 4 输出极性

参考 CC1P 的描述。

Bit 12 **CC4E**：输入/捕获 4 输出使能

参考 CC1E 的描述。

Bit 11 **CC3NP**：输入/捕获 3 互补输出极性
参考 CC1NP 的描述。
Bit 10 **CC3NE**：输入/捕获 3 互补输出使能
参考 CC1NE 的描述。
Bit 9 **CC3P**：输入/捕获 3 输出极性
参考 CC1P 的描述。
Bit 8 **CC3E**：输入/捕获 3 输出使能
参考 CC1E 的描述。
Bit 7 **CC2NP**：输入/捕获 2 互补输出极性
参考 CC1NP 的描述。
Bit 6 **CC2NE**：输入/捕获 2 互补输出使能
参考 CC1NE 的描述。
Bit 5 **CC2P**：输入/捕获 2 输出极性
参考 CC1P 的描述。
Bit 4 **CC2E**：输入/捕获 2 输出使能
参考 CC1E 的描述。
Bit 3 **CC1NP**：输入/捕获 1 互补输出极性
**CC1 通道配置为输出：**
0：tim_oc1n 高电平有效；
1：tim_oc1n 低电平有效。
**CC1 通道配置为输入：**
此位与 CC1P 配合使用，用以定义 tim_ti1fp1 和 tim_ti2fp1 的极性。请参见 CC1P 说明。
*注：此位将在具有互补输出的通道上进行预装载。如果 TIMx_CR2 寄存器中的CCPC 位置 1，则仅当生成换向事件时，CC1NP 有效位才会从预装载位获取新值。*
*注：只要编程了LOCK（TIMx_BDTR 寄存器中的LOCK 位）级别 2 或 3 且 CC1S=00（通道配置为输出），此位立即变为不可写状态。*
Bit 2 **CC1NE**：输入/捕获 1 互补输出使能
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。
*注：此位将在具有互补输出的通道上进行预装载。如果TIMx_CR2 寄存器中的CCPC 位置 1，则仅当生成换向事件时，CC1NE 有效位才会从预装载位获取新值。*

Bit 1 **CC1P**：输入/捕获 1 输出极性

0：OC1 高电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

1：OC1 低电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

**CC1 通道配置为输入，**CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

CC1NP=0，CC1P=0：非反相/上升沿触发。电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=0，CC1P=1：反相/下降沿触发。电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=1，CC1P=1：非反相/上升沿和下降沿均触发。电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

CC1NP=1，CC1P=0：此配置是保留的，不应该使用此配置。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *或*3*，此位立即变为不可写状态。*

*注：此位将在具有互补输出的通道上进行预装载。如果*TIMx_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1P *有效位才会从预装载位获取新值。*

Bit 0 **CC1E**：输入/捕获 1 输出使能

0：禁止捕获模式/OC1 未激活（参见以下）

1：使能捕获模式/ OC1 信号输出到对应的输出引脚

**当 CC1 通道配置为输出，**C1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值，不论 CC1E 位状态。请参考下表获得更多细节

*注：此位将在具有互补输出的通道上进行预装载。如果*TIMx_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1E *有效位才会从预装载位获取新值。*

表 22.21 带刹车功能的互补通道 tim_ocx 和 tim_ocxn 的输出控制位

| 控制位 | | | | | 输出状态 (1) | |
|---|---|---|---|---|---|---|
| MOE 位 | OSSI 位 | OSSR 位 | CCxE 位 | CCxNE 位 | tim_ocx 输出状态 | tim_ocxn 输出状态 |
| 1 | X | X | 0 | 0 | 禁止输出（不由定时器驱动：高阻态）<br>tim_ocx = 0，tim_ocxn = 0 | |
| | | 0 | 0 | 1 | 禁止输出（不由定时器驱动：高阻态）tim_ocx = 0 | tim_ocxref + 极性，tim_ocxn = tim_ocxref 异或 CCxNP |
| | | 0 | 1 | 0 | tim_ocxref + 极性，tim_ocx = tim_ocxref 异或 CCxP | 禁止输出（不由定时器驱动：高阻态） tim_ocxn = 0 |
| | | X | 1 | 1 | tim_ocxref + 极性 + 死区 | tim_ocxref 互补项（对 tim_ocxref 进行“非”运算）+ 极性 + 死区 |
| | | 1 | 0 | 1 | 关闭状态（输出使能为无效状态）<br>tim_ocx = CCxP | tim_ocxref + 极性，tim_ocxn = tim_ocxref 异或 CCxNP |
| | | 1 | 1 | 0 | tim_ocxref + 极性，tim_ocx = tim_ocxref 异或 CCxP | 关闭状态（输出使能为无效状态）<br>tim_ocxn = CCxNP |
| 0 | 0 | X | X | X | 禁止输出（不再由定时器驱动）。 | |
| | 1 | | 0 | 0 | 输出状态由 GPIO 控制器定义，可以是高电平、低电平或高阻态。 | |
| | | | 0 | 1 | 关闭状态（输出使能为无效状态） | |
| | | | 1 | 0 | 异步：tim_ocx = CCxP、tim_ocxn = CCxNP<br>（如果触发 tim_brk 或 tim_brk2）。<br>随后，如果存在时钟：在死区后 tim_ocx = OISx 且 tim_ocxn = OISxN，<br>假定 OISx 和 OISxN 并没有都设置成 tim_ocx 及 tim_ocxn 的有效电平<br>（否则在半桥配置下驱动开关时可能导致短路）。 | |
| | | | 1 | 1 | | |

1. 如果一个通道的两个输出均未使用（由 GPIO 接管控制），则 OISx、OISxN、CCxP 和 CCxNP 位必须保持清零状态。

*注：　与互补通道* tim_ocx *和* tim_ocxn *相连的外部* I/O *引脚的状态取决于通道* tim_ocx *和* tim_ocxn *的状态以及* GPIO *寄存器。*

### 22.6.12 TIMx 计数器（TIMx_CNT）（x = 1，8）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 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 31 **UIFCPY**：UIF 副本（UIF copy）

该位是 TIMx_SR 寄存器中 UIF 位的一个只读副本。如果 TIMx_CR1 上的 UIFREMAP 位被清零，则位 31 被保留并读为 0。

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CNT[15:0]**：计数器的值（Counter value）

在非抖动模式下（DITHEN = 0）

该寄存器（CNT[15:0]）保存的是计数器的值。

在抖动模式下（DITHEN = 1）

该寄存器（CNT[15:0]）只保存计数器的不抖动部分。抖动部分不可用。

### 22.6.13 TIMx 预分频器（TIMx_PSC）（x = 1，8）

偏移地址：0x028

复位值：0x0000

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

Bit 15:0 **PSC[15:0]**：预分频器的值（Prescaler value）

计数器时钟频率（$f_{tim_cnt_ck}$）等于 $f_{tim_psc_ck}$ /（PSC[15:0] + 1）。

PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

### 22.6.14 TIMx 自动重装载寄存器（TIMx_ARR）（x = 1，8）

偏移地址：0x02C

复位：0x0000 FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | 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 31:20 保留，必须保持复位值。

Bit 19:0 **ARR[19:0]：**自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的详细信息，请参考章节 TIMx 功能描述：时基单元。

当自动重载值为空时，计数器不工作。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是自动重新加载的值

在抖动模式下（DITHEN = 1）

该寄存器存储的是 ARR[19:4]中的整数部分，而 ARR[3:0]位域存储的是抖动的部分

### 22.6.15 TIMx 重复计数寄存器（TIMx_RCR）（x = 1，8）

偏移地址：0x030

复位值：0x0000

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

Bit 15:0 **REP[15:0]**：重复计数器的值（Repetition counter value）

使能预装载寄存器时，用户可通过这些位设置比较寄存器的更新频率（即从预装载寄存器向活动寄存器周期性传输数据）；使能更新中断时，也可设置更新中断的生成速率。

与 REP_CNT 相关的减计数器每次计数到 0 时，都将生成一个更新事件并且计数器从 REP 值重新开始计数。由于只有生成重复更新事件 U_RC 时，REP_CNT 才会重载 REP 值，因此在生成下一重复更新事件之前，无论向 TIMx_RCR 寄存器写入何值都无影响。

这意味着 PWM 模式下（REP+1）相当于：

– 边沿对齐模式下的 PWM 周期数；
– 中心对称模式下的 PWM 半周期数；

### 22.6.16 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 1，8）

偏移地址：0x034

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR1 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR1[19:0]：**捕获/比较 1 值（Capture/Compare 1 value）

**若 CC1 通道配置为输出：**

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 1 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR1[15:0]中的比较值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR1[19:4]中的整数部分。CCR1[3:0]位域存储的是抖动的部分。

**若 CC1 通道配置为输入：**

CCR1 为上一个输入捕获 1 事件（tim_ic1）发生时的计数器值。TIMx_CCR1 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR1[15:0]位存储了捕获值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR1[19:4]中。CCR1[3:0]位会被重置。

### 22.6.17 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 1，8）

偏移地址：0x038

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR2 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR2[19:0]：**捕获/比较 2 值（Capture/Compare 2 value）

**若 CC2 通道配置为输出：**

CCR2 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc2 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR2[15:0]中的比较值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR2[19:4]中的整数部分。CCR2[3:0]位域存储的是抖动的部分。

**若 CC2 通道配置为输入：**

CCR2 为上一个输入捕获 2 事件（tim_ic2）发生时的计数器值。TIMx_CCR2 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR2[15:0]位存储了捕获值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR2[19:4]中。CCR2[3:0]位会被重置。

### 22.6.18 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 1，8）

偏移地址：0x03C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR3 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCR3 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR3[19:0]：**捕获/比较 3 值（Capture/Compare 3 value）

**若 CC3 通道配置为输出：**

CCR3 为要装载到实际捕获/比较 3 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC3PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 3 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc3 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR3[15:0]中的比较值。CCR3[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR3[19:4]中的整数部分。CCR3[3:0]位域存储的是抖动的部分。

**若 CC3 通道配置为输入：**

CCR3 为上一个输入捕获 3 事件（tim_ic3）发生时的计数器值。TIMx_CCR3 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR3[15:0]位存储了捕获值。CCR3[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR3[19:4]中。CCR3[3:0]位会被重置。

### 22.6.19 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 1，8）

偏移地址：0x040

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR4 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR4 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR4[19:0]：**捕获/比较 4 值（Capture/Compare 4 value）

**若 CC4 通道配置为输出：**

CCR4 为要装载到实际捕获/比较 4 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC4PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 4 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc4 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR4[15:0]中的比较值。CCR4[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR4[19:4]中的整数部分。CCR4[3:0]位域存储的是抖动的部分。

**若 CC4 通道配置为输入：**

CCR4 为上一个输入捕获 4 事件（tim_ic4）发生时的计数器值。TIMx_CCR4 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR4[15:0]位存储了捕获值。CCR4[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR4[19:4]中。CCR4[3:0]位会被重置。

### 22.6.20 TIMx 刹车和死区寄存器（TIMx_BDTR）（x = 1，8）

偏移地址：0x044

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BK2BID | BKBID | BK2 DSRM | BK DSRM | 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：由于可以根据*LOCK *配置锁定位*BKBID/BK2BID/BK2P、BK2E、BK2F[3:0]、BKF[3:0]、AOE、BKP、BKE、OSSI、OSSR *和*DTG[7:0]*的写操作，因此必须在第一次对* TIMx_BDTR *寄存器执行写访问时对这些位进行配置。*

Bit 31:30 保留，必须保持复位值。

Bit 29 **BK2BID：**刹车 2 双向（Break2 bidirectional）

请参阅 BKBID 说明

Bit 28 **BKBID：**刹车双向（Break bidirectional）

0：刹车输入 tim_brk 处于输入模式

1：刹车输入 tim_brk 处于双向模式

在双向模式（BKBID 置 1）下，刹车输入同时配置为输入模式和开漏输出模式。任何有效的 break 事件会在刹车输入上输出低电平，以向外部设备指示内部 break 事件。

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 27 **BK2DSRM：**刹车 2 解除（Break2 disarm）

请参阅 BKDSRM 说明

Bit 26 **BKDSRM：**刹车解除保护（Break disarm）

0：刹车输入 tim_brk 处于保护状态

1：刹车输入 tim_brk 解除保护

当没有刹车源处于有效状态时，该位由硬件清零。

软件需置位 BKDSRM 以释放双向输出控制（开漏输出处于高阻态），并轮询直至硬件将其清零，表明故障状态已消失。

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 25 **BK2P：**刹车 2 极性（Break 2 polarity）

0：刹车输入 tim_brk2 为低电平有效

1：刹车输入 tim_brk2 为高电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 24 **BK2E：**刹车 2 使能（Break 2 enable）

该位可使能完整的刹车 2 保护（参考图片：刹车和刹车 2 电路概述）。

0：禁用 Break2 功能

1：使能 Break2 功能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 23:20 **BK2F[3:0]：**刹车 2 滤波器（Break 2 filter）

此位域可定义 tim_brk2 输入的采样频率和适用于 tim_brk2 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，tim_brk2 异步工作

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

*注：只要编程了*LOCK*（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 19:16 **BKF[3:0]：**刹车滤波器（Break filter）

此位域可定义 tim_brk 输入的采样频率和适用于 tim_brk 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，tim_brk 异步工作

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

*注：只要编程了*LOCK*（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 15 **MOE**：主输出使能（Main output enable）

只要刹车输入（tim_brk 或 tim_brk2）为有效状态，此位便由硬件异步清零。此位由软件置 1，也可根据 AOE 位状态自动置 1。此位仅对配置为输出的通道有效。

0：响应刹车事件或向 MOE 写入 0 时：tim_ocx 和 tim_ocxn 输出被禁止或被强制为空闲状态，具体取决于 OSSI 位。

1：如果 tim_ocx 和 tim_ocxn 输出的相应使能位（TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位）均置 1，则使能 tim_ocx 和 tim_ocxn 输出。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIMx 捕获/比较使能寄存器（TIMx_CCER）（x=1，8））。

Bit 14 **AOE**：自动输出使能（Automatic output enable）

0：MOE 只能由软件置 1

1：MOE 可由软件置 1，也可在发生下一更新事件时自动置 1（如果刹车输入 tim_brk 和 tim_brk2 均无效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 13 **BKP**：刹车极性（Break polarity）

0：刹车输入 tim_brk 为低电平有效

1：刹车输入 tim_brk 为高电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 12 **BKE**：刹车使能（Break enable）

该位可使能完整的刹车保护（包括所有连接到 tim_sys_brk 和 BKIN 的源，参考图片：刹车和刹车 2 电路概述）。

0：禁用 Break 功能

1：使能 Break 功能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 11 **OSSR**：运行模式下的关闭状态选择（Off-state selection for Run mode）

此位在 MOE=1 时作用于配置为输出模式且具有互补输出的通道。如果定时器中没有互补输出，则不存在 OSSR。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIMx 捕获/比较使能寄存器（TIMx_CCER）（x=1，8））。

0：处于无效状态时，禁止 tim_ocx/tim_ocxn 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，一旦 CCxE=1 或 CCxNE=1，便使能 tim_ocx/tim_ocxn 输出并将其设为无效电平（输出仍由定时器控制）。

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*2，*此位即无法修改。*

Bit 10 **OSSI**：空闲模式下的关闭状态选择（Off-state selection for Idle mode）

当由于刹车事件或软件写操作而使 MOE=0 时，此位作用于配置为输出的通道。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIMx 捕获/比较使能寄存器（TIMx_CCER）（x=1，8））。

0：处于无效状态时，禁止 tim_ocx/tim_ocxn 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，首先将 tim_ocx/tim_ocxn 输出强制为其无效电平，然后在死区后将其强制为空闲电平。定时器始终控制输出。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*2*，此位即无法修改。*

Bit 9:8 **LOCK[1:0]**：锁定设置（Lock configuration）

这些位用于针对软件错误提供写保护。

00：关闭锁定 – 不对任何位提供写保护。

01：锁定级别 1，此时无法对 TIMx_BDTR 寄存器中的 DTG 位、TIMx_CR2 寄存器中的 OISx 和 OISxN 位以及 TIMx_BDTR 寄存器中的 BKBID/BK2BID/BKE/BKP/AOE 位执行写操作。

10：锁定级别 2，此时无法对锁定级别 1 中适用的各位、CC 极性位（TIMx_CCER 寄存器中的 CCxP/CCxNP 位，只要通过 CCxS 位将相关通道配置为输出）以及 OSSR 和 OSSI 位执行写操作。

11：锁定级别 3，此时无法对锁定级别 2 中适用的各位、CC 控制位（TIMx_CCMRx 寄存器中的 OCxM 和 OCxPE 位，只要通过 CCxS 位将相关通道配置为输出）执行写操作。

*注：复位后只能对*LOCK *位执行一次写操作。对*TIMx_BDTR *寄存器执行写操作后其中的内容将冻结，直到下一次复位。*

Bit 7:0 **DTG[7:0]**：配置死区发生器（Dead-time generator setup）

此位域定义插入到互补输出之间的死区持续时间。DT 与该持续时间相对应。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3*，此位域即无法修改。*

### 22.6.21 TIMx 捕获/比较寄存器 5（TIMx_CCR5）（x = 1，8）

偏移地址：0x048

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GC5C3 | GC5C2 | GC5C1 | Reserved | | | | | | | | | CCR5 [19:16] | | | |
| rw | rw | rw | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR5 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **GC5C3：**通道 5 和通道 3 组（Group channel 5 and channel 3）

通道 3 输出失真：

0：tim_oc5ref 对 tim_oc3refc 无影响

1：tim_oc3refc 是 tim_oc3ref 和 tim_oc5ref 的逻辑 AND 操作

此位要么立即生效，要么被预加载并在更新事件后被考虑（如果 TIMx_CCMR2 中选择了预加载功能）。

*注：也可以在组合* PWM *信号上应用这种失真。*

Bit 30 **GC5C2：**通道 5 和通道 2 组（Group channel 5 and channel 2）

通道 2 输出失真：

0：tim_oc5ref 对 tim_oc2refc 无影响

1：tim_oc2refc 是 tim_oc2ref 和 tim_oc5ref 的逻辑 AND 操作

此位要么立即生效，要么被预加载并在更新事件后被考虑（如果 TIMx_CCMR1 中选择了预加载功能）。

*注：也可以在组合* PWM *信号上应用这种失真。*

Bit 29 **GC5C1：**通道 5 和通道 1 组（Group channel 5 and channel 1）

通道 1 输出失真：

0：tim_oc5ref 对 tim_oc1refc 无影响

1：tim_oc1refc 是 tim_oc1ref 和 tim_oc5ref 的逻辑 AND 操作

此位要么立即生效，要么被预加载并在更新事件后被考虑（如果 TIMx_CCMR1 中选择了预加载功能）。

*注：也可以在组合* PWM *信号上应用这种失真。*

Bit 28:20 保留，必须保持复位值。

Bit 19:0 **CCR5[15:0]：**捕获/比较 5 值（Capture/Compare 5 value）

CCR5 为要装载到实际捕获/比较 5 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR3 寄存器中的 OC5PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 5 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc5 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR5[15:0]中的比较值。CCR5[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR5[19:4]中的整数部分。CCR5[3:0]位域存储的是抖动的部分。

### 22.6.22 TIMx 捕获/比较寄存器 6（TIMx_CCR6）（x = 1，8）

偏移地址：0x04C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR6 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR6 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR6[15:0]：**捕获/比较 6 值（Capture/Compare 6 value）

CCR6 为要装载到实际捕获/比较 6 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR3 寄存器中的 OC6PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 6 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc6 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR6[15:0]中的比较值。CCR6[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR6[19:4]中的整数部分。CCR6[3:0]位域存储的是抖动的部分。

### 22.6.23 TIMx 捕获/比较模式寄存器 3（TIMx_CCMR3）（x = 1，8）

偏移地址：0x050

复位值：0x0000 0000

参看以上 CCMR1 寄存器的描述，通道 5 和 6 只能配置为输出。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC6M[3] | Reserved | | | | | | | 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 | Reserved | | OC5CE | OC5M[2:0] | | | OC5PE | OC5FE | Reserved | |
| rw | rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | | |

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC6CE：**输出比较 6 清零使能（Output Compare 6 clear enable）

Bit 24, 14:12 **OC6M[2:0]：**输出比较 6 模式（Output Compare 6 mode）

Bit 11 **OC6PE：**输出比较 6 预装载使能（Output Compare 6 preload enable）

Bit 10 **OC6FE：**输出比较 6 快速使能（Output Compare 6 fast enable）

Bit 9:8 保留，必须保持复位值。

Bit 7 **OC5CE：**输出比较 5 清零使能（Output Compare 5 clear enable）

Bit 16, 6:4 **OC5M[2:0]：**输出比较 5 模式（Output Compare5 mode）

Bit 3 **OC5PE：**输出比较 5 预装载使能（Output Compare 5 preload enable）

Bit 2 **OC5FE：**输出比较 5 快速使能（Output Compare 5 fast enable）

Bit 1:0 保留，必须保持复位值。

### 22.6.24 TIMx 定时器死区寄存器 2（TIMx_DTR2）（x = 1，8）

偏移地址：0x054

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DTPE | DTAE |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 31:18 保留，必须保持复位值。

Bit 17 **DTPE：**死区时间预加载使能（Deadtime preload enable）

0：死区时间值未预加载

1：使能死区时间值预加载

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 16 **DTAE：**死区时间非对称使能（Deadtime asymmetric enable）

0：上升沿和下降沿的死区时间是相同的，并且使用 DTG[7:0]寄存器定义

1：上升沿的死区时间使用 DTG[7:0]寄存器定义，下降沿的死区时间使用 DTGF[7:0]位定义。

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 15:8 保留，必须保持复位值。

Bit 7:0 **DTGF[7:0]：**死区下降沿发生器（Dead-time falling edge generator setup）

这个位域定义了在下降沿时，互补输出之间插入死区时间的长度。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

### 22.6.25 TIMx 定时器编码器控制寄存器（TIMx_ECR）（x = 1，8）

偏移地址：0x058

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | IPOS[1:0] | | FIDX | Reserved | | IDIR[1:0] | | IE |
| | | | | | | | | rw | rw | rw | | | rw | rw | rw |

Bit 31:27 保留，必须保持复位值。

Bit 26:24 **PWPRSC[2:0]：**脉冲宽度预分频器（Pulse width prescaler）

这个位域为脉冲发生器设置时钟预分频器，具体如下：

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bit 23:16 **PW[7:0]：**脉冲宽度（Pulse width）

这个位域定义脉冲的持续时间，具体如下：

$t_{PW} = PW[7:0] \times t_{PWG}$

Bit 15:8 保留，必须保持复位值。

Bit 7:6 **IPOS[1:0]：索引位置**（Index positioning）

在正交编码器模式（SMS[3:0] = 0001，0010，0011，1110，1111）下，该位指示了索引事件在哪个 AB 输入配置下会重置计数器。

00：当 AB=00 时，索引重置计数器

01：当 AB=01 时，索引重置计数器

10：当 AB=10 时，索引重置计数器

11：当 AB=11 时，索引重置计数器

在方向时钟模式或时钟加方向模式（SMS[3:0] = 1010，1011，1100，1101）下，此位指示了索引事件在哪个时钟水平上重置计数器。在双向时钟模式下，这适用于两个时钟输入。具体为：

x0：当时钟=0 时，索引重置计数器

x1：当时钟=1 时，索引重置计数器

*注：*IPOS[1]*位不重要*

Bit 5 **FIDX[1:0]：首位索引**（First index）

该位指示是否仅考虑首位索引

0：索引始终处于活跃状态

1：仅首个索引会重置计数器

Bit 4:3 保留，必须保持复位值。

Bit 2:1 **IDIR[1:0]：索引方向**（Index direction）

该位指示索引事件在哪个方向上重置计数器。

00：无论方向如何，索引都会重置计数器

01：仅在递增计数时，索引会重置计数器

10：仅在递减计数时，索引会重置计数器

11：保留

注：当 IE 位被复位（禁止索引）时，必须写入 IDR[1:0]位域

Bit 0 **IE：索引使能**（Index enable）

该位指示索引事件是否能重置计数器

0：禁止索引

1：使能索引

### 22.6.26 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x =1，8）

偏移地址：0x05C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TI4SEL[3:0] | | | | Reserved | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TI2SEL[3:0] | | | | Reserved | | | | TI1SEL[3:0] | | | |
| | | | | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bit 31:28 保留，必须保持复位值。

Bit 27:24 **TI4SEL[3:0]：**选择 tim_ti4[15:0]输入（Selects tim_ti4[15:0] input）

0000：tim_ti4_in0: TIMx_CH4

0001：tim_ti4_in1

...

1111：tim_ti4_in15

参考章节：TIM1/TIM8 引脚和内部信号获得关联列表。

Bit 23:20 保留，必须保持复位值。

Bit 19:16 **TI3SEL[3:0]：**选择 tim_ti3[15:0]输入（Selects tim_ti3[15:0] input）

0000：tim_ti3_in0: TIMx_CH3

0001：tim_ti3_in1

...

1111：tim_ti3_in15

参考章节：TIM1/TIM8 引脚和内部信号获得关联列表。

Bit 15:12 保留，必须保持复位值。

Bit 11:8 **TI2SEL[3:0]：**选择 tim_ti2[15:0]输入（Selects tim_ti2[15:0] input）

0000：tim_ti2_in0: TIMx_CH2

0001：tim_ti2_in1

...

1111：tim_ti2_in15

参考章节：TIM1/TIM8 引脚和内部信号获得关联列表。

Bit 7:4 保留，必须保持复位值。

Bit 3:0 **TI1SEL[3:0]：**选择 tim_ti1[15:0]输入（Selects tim_ti1[15:0] input）

0000：tim_ti1_in0: TIMx_CH1

0001：tim_ti1_in1

...

1111：tim_ti1_in15

参考章节：TIM1/TIM8 引脚和内部信号获得关联列表。

### 22.6.27 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 1，8）

偏移地址：0x060

复位值：0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | 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 | Reserved | | | | BK CMP4E | BK CMP3E | BK CMP2E | BK CMP1E | BKINE |
| rw | rw | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw |

Bit 31:18 保留，必须保持复位值。

Bit 17:14 **ETRSEL[3:0]：**etr_in 源选择（etr_in source selection）

这些位选择 etr_in 输入源。

0000：tim_etr0: TIMx_ETR 输入

0001：tim_etr1

...

1111：tim_etr15

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 13 **BKCMP4P：**tim_brk_cmp4 输入极性（tim_brk_cmp4 input polarity）

此位选择 tim_brk_cmp4 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp4 输入极性未反转（如果 BKP=0，则低电平有效，如果 BKP=1，则高电平有效）

1：tim_brk_cmp4 输入极性反转（如果 BKP=0，则高电平有效，如果 BKP=1，则低电平有效）

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 12 **BKCMP3P：**tim_brk_cmp3 输入极性（tim_brk_cmp3 input polarity）

此位选择 tim_brk_cmp3 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp3 输入极性未反转（如果 BKP=0，则低电平有效，如果 BKP=1，则高电平有效）

1：tim_brk_cmp3 输入极性反转（如果 BKP=0，则高电平有效，如果 BKP=1，则低电平有效）

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 11 **BKCMP2P：**tim_brk_cmp2 输入极性（tim_brk_cmp2 input polarity）

此位选择 tim_brk_cmp2 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp2 输入极性未反转（如果 BKP=0，则低电平有效，如果 BKP=1，则高电平有效）

1：tim_brk_cmp2 输入极性反转（如果 BKP=0，则高电平有效，如果 BKP=1，则低电平有效）

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 10 **BKCMP1P：**tim_brk_cmp1 输入极性（tim_brk_cmp1 input polarity）

此位选择 tim_brk_cmp1 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp1 输入极性未反转（如果 BKP=0，则低电平有效，如果 BKP=1，则高电平有效）

1：tim_brk_cmp1 输入极性反转（如果 BKP=0，则高电平有效，如果 BKP=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 9 **BKINP**：TIMx_BKIN 输入极性（TIMx_BKIN input polarity）

该位选择 TIMx_BKIN 备用功能输入敏感度。必须对其和 BKP 极性位进行编程。

0：TIMx_BKIN 输入极性未反转（如果 BKP=0，则低电平有效，如果 BKP=1，则高电平有效）

1：TIMx_BKIN 输入极性反转（如果 BKP=0，则高电平有效，如果 BKP=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 8:5 保留，必须保持复位值。

Bit 4 **BKCMP4E**：tim_brk_cmp4 使能（tim_brk_cmp4 enable）

该位用于使能 tim_brk 输入的 tim_brk_cmp4。tim_brk_cmp4 输出与其他 tim_brk 源进行“或”运算。

0：tim_brk_cmp4 输入禁用

1：tim_brk_cmp4 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 3 **BKCMP3E**：tim_brk_cmp3 使能（tim_brk_cmp3 enable）

该位用于使能 tim_brk 输入的 tim_brk_cmp3。tim_brk_cmp3 输出与其他 tim_brk 源进行“或”运算。

0：tim_brk_cmp3 输入禁用

1：tim_brk_cmp3 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 2 **BKCMP2E**：tim_brk_cmp2 使能（tim_brk_cmp2 enable）

该位用于使能 tim_brk 输入的 tim_brk_cmp2。tim_brk_cmp2 输出与其他 tim_brk 源进行“或”运算。

0：tim_brk_cmp2 输入禁用

1：tim_brk_cmp2 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 1 **BKCMP1E**：tim_brk_cmp1 使能（tim_brk_cmp1 enable）

该位用于使能 tim_brk 输入的 tim_brk_cmp1。tim_brk_cmp1 输出与其他 tim_brk 源进行“或”运算。

0：tim_brk_cmp1 输入禁用

1：tim_brk_cmp1 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 0 **BKINE**：TIMx_BKIN 输入使能（TIMx_BKIN input enable）

该位用于使能 TIMx_BKIN 复用功能输入作为定时器的 tim_brk 输入。TIMx_BKIN 输入与其他 tim_brk 源进行“或”运算。

0：TIMx_BKIN 输入禁用

1：TIMx_BKIN 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注意：有关产品特定实现的信息，请参考章节：*TIM1/TIM8 *引脚和内部信号。*

### 22.6.28 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 1，8）

偏移地址：0x064

复位值：0x0000 0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | OCRSEL[2:0] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BK2 CMP4P | BK2 CMP3P | BK2 CMP2P | BK2 CMP1P | BK2INP | Reserved | | | | BK2 CMP4E | BK2 CMP3E | BK2 CMP2E | BK2 CMP1E | BK2INE |
| | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw |

Bit 31:19 保留，必须保持复位值。

Bit 18:16 **OCRSEL[2:0]：**ocref_clr 源选择（ocref_clr source selection）

这些位选择 ocref_clr 输入源。

000：tim_ocref_clr0

001：tim_ocref_clr1

...

111：tim_ocref_clr7

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 15:14 保留，必须保持复位值。

Bit 13 **BK2CMP4P：**tim_brk2_cmp4 输入极性（tim_brk2_cmp4 input polarity）

此位选择 tim_brk2_cmp4 输入敏感度。它必须与 BK2P 极性位一起编程。

0：tim_brk2_cmp4 输入极性未反转（如果 BK2P=0，则低电平有效，如果 BK2P=1，则高电平有效）

1：tim_brk2_cmp4 输入极性反转（如果 BK2P=0，则高电平有效，如果 BK2P=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 12 **BK2CMP3P：**tim_brk2_cmp3 输入极性（tim_brk2_cmp3 input polarity）

此位选择 tim_brk2_cmp3 输入敏感度。它必须与 BK2P 极性位一起编程。

0：tim_brk2_cmp3 输入极性未反转（如果 BK2P=0，则低电平有效，如果 BK2P=1，则高电平有效）

1：tim_brk2_cmp3 输入极性反转（如果 BK2P=0，则高电平有效，如果 BK2P=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 11 **BK2CMP2P：**tim_brk2_cmp2 输入极性（tim_brk2_cmp2 input polarity）

此位选择 tim_brk2_cmp2 输入敏感度。它必须与 BK2P 极性位一起编程。

0：tim_brk2_cmp2 输入极性未反转（如果 BK2P=0，则低电平有效，如果 BK2P=1，则高电平有效）

1：tim_brk2_cmp2 输入极性反转（如果 BK2P=0，则高电平有效，如果 BK2P=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 10 **BK2CMP1P：**tim_brk2_cmp1 输入极性（tim_brk2_cmp1 input polarity）

此位选择 tim_brk2_cmp1 输入敏感度。它必须与 BK2P 极性位一起编程。

0：tim_brk2_cmp1 输入极性未反转（如果 BK2P=0，则低电平有效，如果 BK2P=1，

则高电平有效）

1：tim_brk2_cmp1 输入极性反转（如果 BK2P=0，则高电平有效，如果 BK2P=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 9 **BK2INP**：TIMx_BKIN2 输入极性（TIMx_BKIN2 input polarity）

该位选择 TIMx_BKIN2 复用功能输入敏感度。它必须与 BK2P 极性位一起编程。

0：TIMx_BKIN2 输入极性未反转（如果 BK2P=0，则低电平有效，如果 BK2P=1，则高电平有效）

1：TIMx_BKIN2 输入极性反转（如果 BK2P=0，则高电平有效，如果 BK2P=1，则低电平有效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 8:5 保留，必须保持复位值。

Bit 4 **BK2CMP4E**：tim_brk2_cmp4 使能（tim_brk2_cmp4 enable）

该位用于使能 tim_brk2 输入的 tim_brk_cmp4。tim_brk2_cmp4 输出与其他 tim_brk2 源进行“或”运算。

0：tim_brk2_cmp4 输入禁用

1：tim_brk2_cmp4 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 3 **BK2CMP3E**：tim_brk2_cmp3 使能（tim_brk2_cmp3 enable）

该位用于使能 tim_brk2 输入的 tim_brk_cmp3。tim_brk2_cmp3 输出与其他 tim_brk2 源进行“或”运算。

0：tim_brk2_cmp3 输入禁用

1：tim_brk2_cmp3 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 2 **BK2CMP2E**：tim_brk2_cmp2 使能（tim_brk2_cmp2 enable）

该位用于使能 tim_brk2 输入的 tim_brk_cmp2。tim_brk2_cmp2 输出与其他 tim_brk2 源进行“或”运算。

0：tim_brk2_cmp2 输入禁用

1：tim_brk2_cmp2 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 1 **BK2CMP1E**：tim_brk2_cmp1 使能（tim_brk2_cmp1 enable）

该位用于使能 tim_brk2 输入的 tim_brk_cmp1。tim_brk2_cmp1 输出与其他 tim_brk2 源进行“或”运算。

0：tim_brk2_cmp1 输入禁用

1：tim_brk2_cmp1 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 0 **BK2INE**：TIMx_BKIN2 输入使能（TIMx_BKIN2 input enable）

该位用于使能 TIMx_BKIN2 复用功能输入作为定时器的 tim_brk2 输入。TIMx_BKIN2 输入与其他 tim_brk2 源进行“或”运算。

0：TIMx_BKIN2 输入禁用

1：TIMx_BKIN2 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注意：有关产品特定实现的信息，请参考章节：*TIM1/TIM8 *引脚和内部信号。*

### 22.6.29 TIMx 捕获/比较寄存器 1N（TIMx_CCR1N）（x = 1，8）

偏移地址：0x068

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EN | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1N[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **EN**：若 CC1 通道配置为输出：

CCR1N_EN=1：CNT 向下数比较的寄存器为 CCR1N

CCR1N_EN=0：CNT 向下数比较的寄存器为 CCR1

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CCR1N[15:0]**：捕获/比较通道 1N 的值（Capture/Compare 1N value）

若 CC1 通道配置为输出：

CCR1N 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。CCR1N_EN 必须为 1，寄存器才有效果。

### 22.6.30 TIMx 捕获/比较寄存器 2N（TIMx_CCR2N）（x = 1，8）

偏移地址：0x06C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EN | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2N[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **EN**：若 CC2 通道配置为输出：

CCR2N_EN=1：CNT 向下数比较的寄存器为 CCR2N

CCR2N_EN=0：CNT 向下数比较的寄存器为 CCR2

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CCR2N[15:0]**：捕获/比较通道 2N 的值（Capture/Compare 2N value）

若 CC2 通道配置为输出：

CCR2N 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。CCR2N_EN 必须为 1，寄存器才有效果。

### 22.6.31 TIMx 捕获/比较寄存器 3N（TIMx_CCR3N）（x = 1，8）

偏移地址：0x070

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EN | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR3N[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **EN**：若 CC3 通道配置为输出：

CCR3N_EN=1：CNT 向下数比较的寄存器为 CCR3N

CCR3N_EN=0：CNT 向下数比较的寄存器为 CCR3

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CCR3N[15:0]**：捕获/比较通道 3N 的值（Capture/Compare 3N value）

若 CC3 通道配置为输出：

CCR3N 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 1 寄存器中。CCR3N_EN 必须为 1，寄存器才有效果。

### 22.6.32 TIMx 捕获/比较寄存器 4N（TIMx_CCR4N）（x = 1，8）

偏移地址：0x074

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EN | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR4N[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **EN**：若 CC4 通道配置为输出：

CCR4N_EN=1：CNT 向下数比较的寄存器为 CCR4N

CCR4N_EN=0：CNT 向下数比较的寄存器为 CCR4

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CCR4N[15:0]**：捕获/比较通道 4N 的值（Capture/Compare 4N value）

若 CC4 通道配置为输出：

CCR4N 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 1 寄存器中。CCR4N_EN 必须为 1，寄存器才有效果。

### 22.6.33 TIMx 刹车输入控制选择（TIMx_BICS）（x = 1，8）

偏移地址：0x078

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | BICE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

Bit 15:7 保留，必须保持复位值。

Bit 6 **CH6**：刹车输入控制选择 CH6（Break Iutput Control Selection CH6）

当 BICE=1 时：

0：CH6 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH6 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 5 **CH5**：刹车输入控制选择 CH5（Break Iutput Control Selection CH5）

当 BICE=1 时：

0：CH5 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH5 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 4 **CH4**：刹车输入控制选择 CH4（Break Iutput Control Selection CH4）

当 BICE=1 时：

0：CH4 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH4 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 3 **CH3**：刹车输入控制选择 CH3（Break Iutput Control Selection CH3）

当 BICE=1 时：

0：CH3 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH3 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 2 **CH2**：刹车输入控制选择 CH2（Break Iutput Control Selection CH2）

当 BICE=1 时：

0：CH2 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH2 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 1 **CH1**：刹车输入控制选择 CH1（Break Iutput Control Selection CH1）

当 BICE=1 时：

0：CH1 刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

1：CH1 刹车无效，设置 MOE=0 无效，通道维持原有输出

Bit 0 **BICE**：刹车输入控制使能（Break Iutput Control Enable）

当刹车输入或设置 MOE=0 时，此位控制刹车或设置 MOE=0 是否对 CHx/CHyN(x=1,2,3,4,5,6)(y=1,2,3,4)有效：

BICE=0：刹车有效或设置 MOE=0 有效，通道根据极性配置输出对应电平

BICE=1：根据 BICS_CHx 位可选某些通道忽略刹车输入或忽略设置 MOE=0 操作维持原有输出

### 22.6.34 TIMx DMA 控制寄存器（TIMx_DCR）（x = 1，8）

偏移地址：0x3DC

复位值：0x0000 0000

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

Bit 31:13 保留，必须保持复位值。

Bit 12:8 **DBL[4:0]：**DMA 突发长度（DMA burst length）

这个 5 位向量定义了 DMA 传输的长度（当对 TIMx_DMAR 地址进行读或写访问时，定时器识别一次突发传输），即传输次数。传输可以是半字或字节（见下面的示例）。

00000：1 次传输

00001：2 次传输

00010：3 次传输

…

11010：26 次传输

**示例：**让我们考虑以下传输：DBL = 7 字节 & DBA = TIM2_CR1。如果 DBL = 7 字节，并且 DBA = TIM2_CR1 表示要传输的字节的地址，那么传输的地址应由以下公式给出：(TIMx_CR1 地址) + DBA + (DMA 索引)，其中 DMA 索引 = DBL

在这个例子中，将 7 字节添加到(TIMx_CR1 地址) + DBA，这给出了数据从/复制到的地址。在这种情况下，传输是从以下地址开始的 7 个寄存器：(TIMx_CR1 地址) + DBA

根据 DMA 数据大小的配置，可能会发生几种情况：

– 如果 DMA 数据大小配置为半字，则将 16 位数据传输到 7 个寄存器中的每一个。

– 如果 DMA 数据大小配置为字节，则数据也传输到 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，依此类推。

因此，在使用传输定时器时，还必须指定 DMA 传输的数据大小。

Bit 7:5 保留，必须保持复位值。

Bit 4:0 **DBA[4:0]：**DMA 基地址（DMA base address）

这个 5 位向量定义了 DMA 传输的基地址（当通过 TIMx_DMAR 地址进行读/写访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始的偏移量。

00000：TIMx_CR1,

00001：TIMx_CR2,

00010：TIMx_SMCR,

…

### 22.6.35 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 1，8）

偏移地址：0x3E0

复位值：0x0000 0000

| 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 31:0 **DMAB[31:0]：**用于突发访问的 DMA 寄存器（DMA register for burst accesses）

对 DMAR 寄存器进行的读或写操作将访问位于以下地址的寄存器：

(TIMx_CR1 地址) + (DBA + DMA 索引) × 4

其中，TIMx_CR1 地址是控制寄存器 1 的地址，DBA 是在 TIMx_DCR 寄存器中配置的 DMA 基地址，DMA 索引由 DMA 传输自动控制，其范围从 0 到 DBL（在 TIMx_DCR 中配置的 DBL）。

# 23 通用定时器（TIM2/TIM3/TIM4/TIM5）

## 23.1 TIM2/TIM3/TIM4/TIM5 简介

通用定时器包含一个 16 位或 28 位自动重载计数器，该计数器由可编程预分频器驱动。

它们可用于多种用途，包括测量输入信号的脉冲宽度（输入捕获）或生成输出波形（输出比较和 PWM）。使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。这些定时器彼此完全独立，不共享任何资源，它们可以同步操作。

## 23.2 TIM2/TIM3/TIM4/TIM5 主要功能

通用 TIMx 定时器功能包括：

- 16 位（TIM3 和 TIM4）或 28 位（TIM2 和 TIM5）向上、向下、向上/向下自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿或中间对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA 请求：
  - 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

## 23.3 TIM2/TIM3/TIM4/TIM5 实现

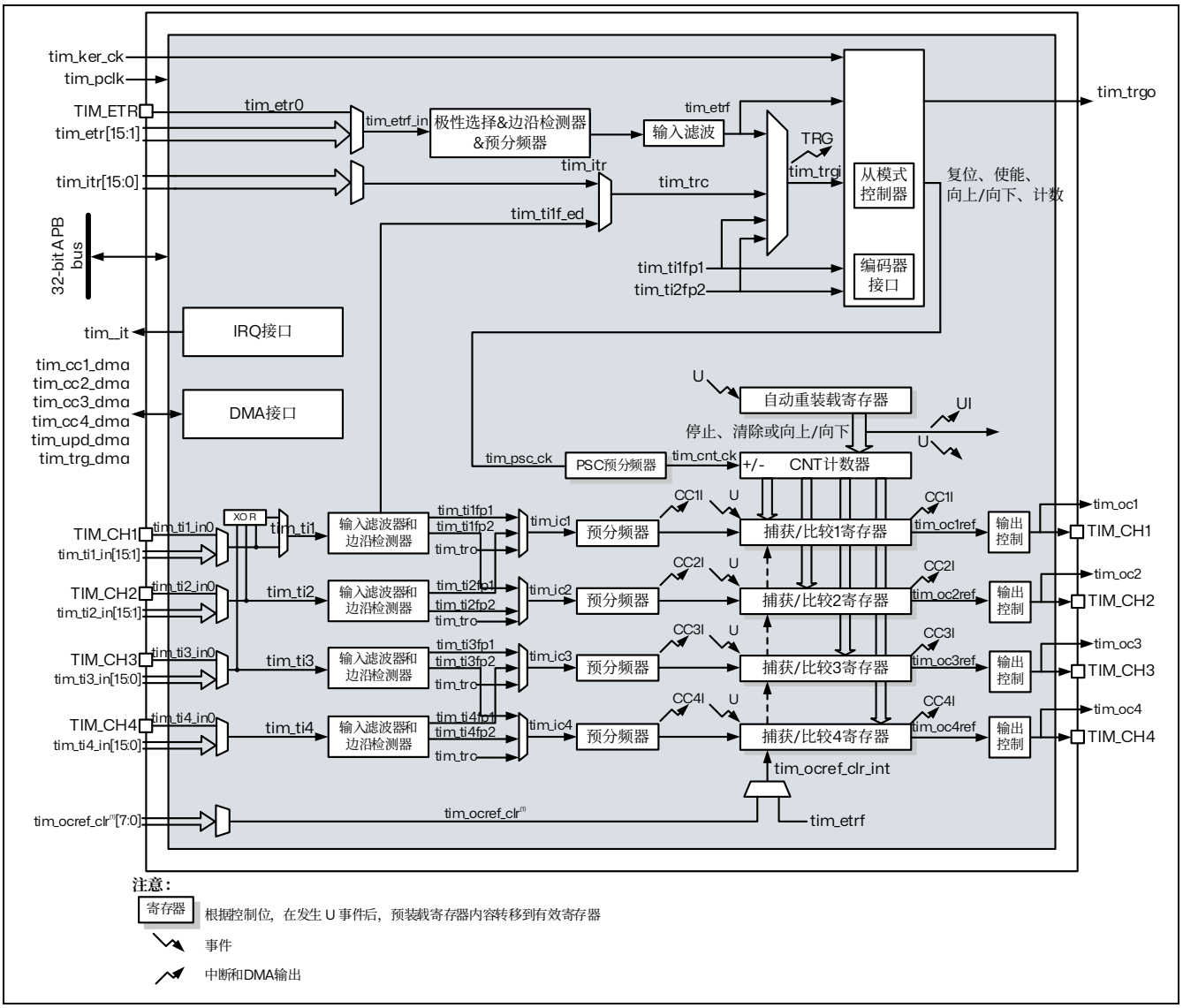
表 23.1 RX32G410 系列通用定时器

| 定时器实例 | TIM2 | TIM3 | TIM4 | TIM5 |
|---|---|---|---|---|
| 分辨率 | 28 位 | 16 位 | 16 位 | 28 位 |
| OCREF 清除选择源 | 是<br>tim_etrf<br>tim_ocref_clr[7:0] | 是<br>tim_etrf<br>tim_ocref_clr[7:0] | 否<br>tim_etrf<br>- | 否<br>tim_etrf<br>- |

## 23.4 TIM2/TIM3/TIM4/TIM5 功能描述

### 23.4.1 框图

图 23.1 通用定时器框图




注 1. 此功能具体请参考章节：TIM2/TIM3/TIM4/TIM5 实现

### 23.4.2 TIM2/TIM3/TIM4/TIM5 引脚和内部信号

本节中的下两张表格总结了 TIM 的输入和输出。

表 23.2 TIM 输入/输出引脚

| 引脚名称 | 信号类型 | 描述 |
|---|---|---|
| TIM_CH1<br>TIM_CH2<br>TIM_CH3<br>TIM_CH4 | 输入/输出 | 定时器多功能通道。<br>每个通道用于捕获、比较或 PWM。<br>TIM_CH1 和 TIM_CH2 也可用作外部时钟（tim_ker_ck 时钟的 1/4 以下），外部触发器和正交编码器输入。<br>TIM_CH1，TIM_CH2 和 TIM_CH3 可被用作数字霍尔效应传感器接口。 |
| TIM_ETR | 输入 | 外部触发器输入。这个输入可被用作外部触发器或外部时钟源。<br>如果使用 tim_etr_in 预分频器，则该输入可以接收频率高于 tim_ker_ck 的时钟。 |

表 23.3 TIM 内部输入/输出信号

| 内部信号名称 | 信号类型 | 描述 |
|---|---|---|
| tim_ti1_in[15:0]<br>tim_ti2_in[15:0]<br>tim_ti3_in[15:0]<br>tim_ti4_in[15:0] | 输入 | 内部定时器输入总线。<br>tim_ti1_in[15:0]和 tim_ti2_in[15:0]输入可被用作捕获或作为外部时钟（tim_ker_ck 时钟的 1/4 以下）和正交编码器信号。 |
| tim_etr[15:0] | 输入 | 外部触发器内部输入总线。这些输入可被用作触发器、外部时钟或用于硬件的逐周期脉冲宽度控制。如果使用 tim_etr_in 预分频器，这些输入可以接收频率高于 tim_ker_ck 的时钟。 |
| tim_itr[15:0] | 输入 | 内部触发输入总线。这些输入可用于从模式控制器或作为输入时钟（tim_ker_ck 时钟的 1/4 以下）。 |
| tim_trgo | 输出 | 内部触发器输出。此触发器可以触发其他片上外设。 |
| tim_ocref_clr[7:0] | 输入 | 定时器 tim_ocref_clr 输入总线。这些输入可以用来清除 tim_ocxref 信号，通常用于硬件的逐周期脉冲宽度控制。 |
| tim_pclk | 输入 | 定时器 APB 时钟。 |
| tim_ker_ck | 输入 | 定时器内核时钟。 |
| tim_it | 输出 | 全局定时器中断，收集捕获/比较、更新和中断触发器请求。 |
| tim_cc1_dma<br>tim_cc2_dma<br>tim_cc3_dma<br>tim_cc4_dma | 输出 | 定时器捕获/比较 1 至 4 dma 请求。 |
| tim_upd_dma | 输出 | 定时器更新 dma 请求。 |
| tim_trg_dma | 输出 | 定时器触发 dma 请求。 |

如下四张表列出了连接到 tim_ti[4:1]输入多路复用器的源。

表 23.4 连接到 tim_ti1 的输入多路复用器

| tim_ti1 输入 | 源 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_ti1_in0 | TIM2_CH1 | TIM3_CH1 | TIM4_CH1 | TIM5_CH1 |
| tim_ti1_in1 | comp1_out | comp1_out | comp1_out | LSI |
| tim_ti1_in2 | comp2_out | comp2_out | comp2_out | 保留 |
| tim_ti1_in3 | comp3_out | comp3_out | comp3_out | RTC 唤醒 |
| tim_ti1_in4 | comp4_out | comp4_out | comp4_out | comp1_out |
| tim_ti1_in5 | 保留 | | | comp2_out |
| tim_ti1_in6 | | | | comp3_out |
| tim_ti1_in7 | | | | comp4_out |

表 23.5 连接到 tim_ti2 的输入多路复用器

| tim_ti2 输入 | 源 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_ti2_in0 | TIM2_CH2 | TIM3_CH2 | TIM4_CH2 | TIM5_CH2 |
| tim_ti2_in1 | comp1_out | comp1_out | comp1_out | comp1_out |
| tim_ti2_in2 | comp2_out | comp2_out | comp2_out | comp2_out |
| tim_ti2_in3 | comp3_out | comp3_out | comp3_out | comp3_out |
| tim_ti2_in4 | comp4_out | comp4_out | comp4_out | comp4_out |

表 23.6 连接到 tim_ti3 的输入多路复用器

| tim_ti3 输入 | 源 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_ti3_in0 | TIM2_CH3 | TIM3_CH3 | TIM4_CH3 | TIM5_CH3 |
| tim_ti3_in1 | comp4_out | comp3_out | 保留 | 保留 |
| tim_ti2_in[2..15] | 保留 | | | |

表 23.7 连接到 tim_ti4 的输入多路复用器

| tim_ti4 输入 | 源 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_ti4_in0 | TIM2_CH4 | TIM3_CH4 | TIM4_CH4 | TIM5_CH4 |
| tim_ti4_in1 | comp1_out | 保留 | | |
| tim_ti4_in2 | comp2_out | | | |
| tim_ti4_in[3..15] | 保留 | | | |

下表列出连接到 tim_itr 输入多路复用器的内部源

表 23.8 TIMx 内部触发连接

| tim_itrx 输入 | TIM2 | TIM3 | TIM4 | TIM5 |
|---|---|---|---|---|
| tim_itr0 | tim1_trgo | tim1_trgo | tim1_trgo | tim1_trgo |
| tim_itr1 | 保留 | tim2_trgo | tim2_trgo | tim2_trgo |
| tim_itr2 | tim3_trgo | 保留 | tim3_trgo | tim3_trgo |
| tim_itr3 | tim4_trgo | tim4_trgo | 保留 | tim4_trgo |
| tim_itr4 | tim5_trgo | tim5_trgo | tim5_trgo | 保留 |
| tim_itr5 | tim8_trgo | tim8_trgo | tim8_trgo | tim8_trgo |
| tim_itr6 | tim15_trgo | tim15_trgo | tim15_trgo | tim15_trgo |
| tim_itr7 | tim16_oc1 | tim16_oc1 | tim16_oc1 | tim16_oc1 |
| tim_itr8 | tim17_oc1 | tim17_oc1 | tim17_oc1 | tim17_oc1 |
| tim_itr9 | 保留 | | | |
| tim_itr10 | hrtim_out_sync2 | hrtim_out_sync2 | hrtim_out_sync2 | hrtim_out_sync2 |
| tim_itr11 | USB SOF SYNC | 保留 | | |
| tim_itr[15:12] | 保留 | | | |

下表列出连接到 tim_etr 输入多路复用器的内部源

表 23.9 连接到 tim_etr 的输入多路复用器

| 定时器外部触发输入信号 | 定时器外部触发信号分配 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_etr0 | TIM2_ETR | TIM3_ETR | TIM4_ETR | TIM5_ETR |
| tim_etr1 | comp1_out | comp1_out | comp1_out | comp1_out |
| tim_etr2 | comp2_out | comp2_out | comp2_out | comp2_out |
| tim_etr3 | comp3_out | comp3_out | comp3_out | comp3_out |
| tim_etr4 | comp4_out | comp4_out | comp4_out | comp4_out |
| tim_etr5 | 保留 | | | |
| tim_etr6 | | | | |
| tim_etr7 | | | | |
| tim_etr8 | TIM3_ETR | TIM2_ETR | TIM3_ETR | TIM2_ETR |
| tim_etr9 | TIM4_ETR | TIM4_ETR | TIM5_ETR | TIM3_ETR |
| tim_etr10 | TIM5_ETR | 保留 | 保留 | |
| tim_etr11 | 保留 | adc2_awd | | |
| tim_etr12 | | 保留 | | |
| tim_etr13 | | | | |
| tim_etr[15:14] | 保留 | | | |

下表列出连接到 tim_ocref_clr 输入多路复用器的内部源

表 23.10 连接到 ocref_clr 的输入多路复用器

| 定时器信号 tim_ocref_clr | 定时器 tim_ocref_clr 信号分配 | | | |
|---|---|---|---|---|
| | TIM2 | TIM3 | TIM4 | TIM5 |
| tim_ocref_clr0 | comp1_out | comp1_out | 保留 | |
| tim_ocref_clr1 | comp2_out | comp2_out | | |
| tim_ocref_clr2 | comp3_out | comp3_out | | |
| tim_ocref_clr3 | comp4_out | comp4_out | | |

### 23.4.3 时基单元

可编程定时器的主要模块由一个 16 位/28 位计数器及其相关的自动重装寄存器组成。计数器可递增计数、递减计数或同时进行递增和递减计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

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）

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件（UEV）时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位（ARPE）。当计数器达到上溢值（或者在递减计数时达到下溢值）并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生 进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位（CEN）置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

请注意，真正的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

**预分频器描述**

预分频器可对计数器时钟频率进行分频，分频系数介于 1 到 65536 之间。该预分频器基于 16 位/28 位寄存器（TIMx_PSC 寄存器）所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用.

下两张图给出了在预分频器运行时，更改计数器参数的例子。

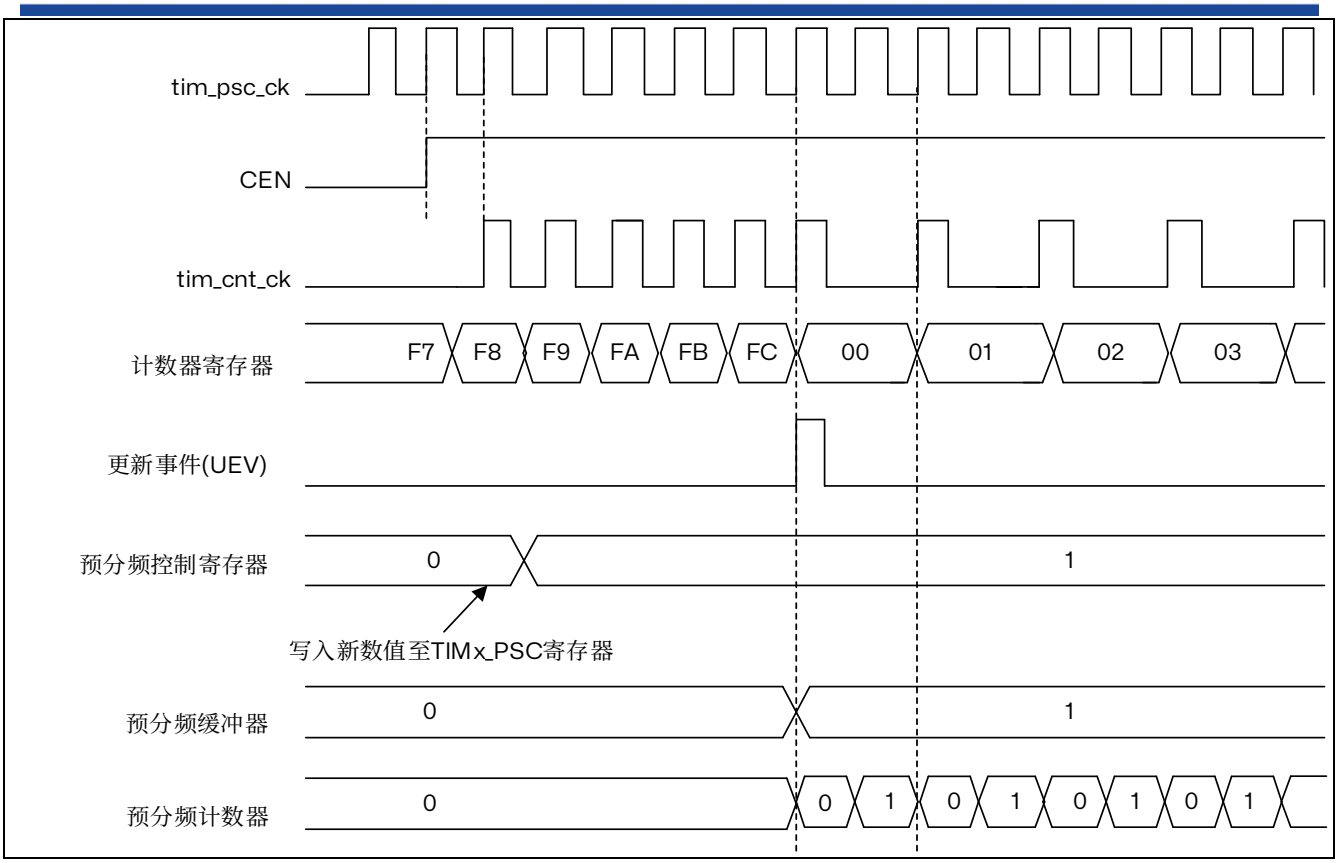



图 23.2 当预分频器的参数从 1 变到 2 时，计数器的时序图

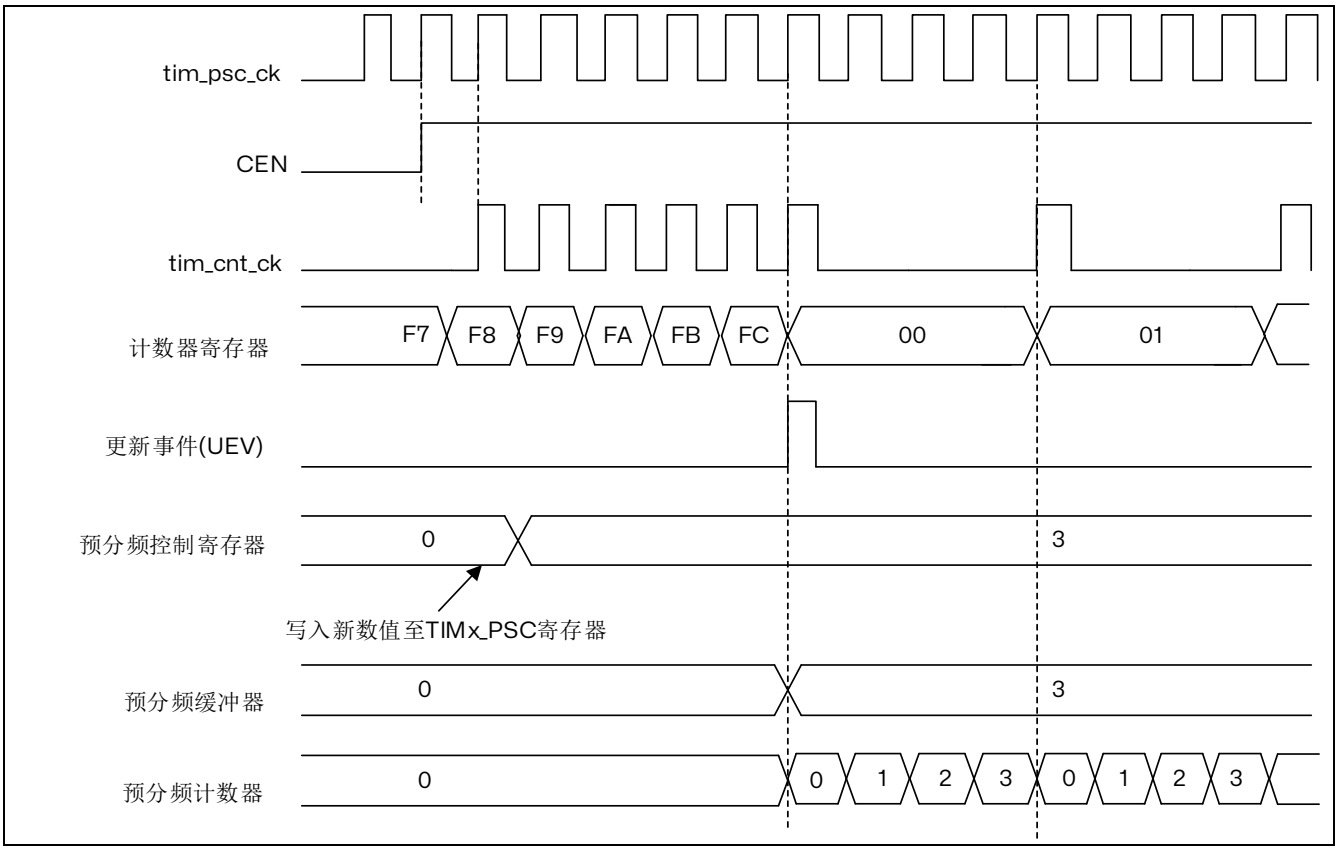



图 23.3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 23.4.4 计数器模式

#### 递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 自动重载影子寄存器将以预装载值进行更新

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

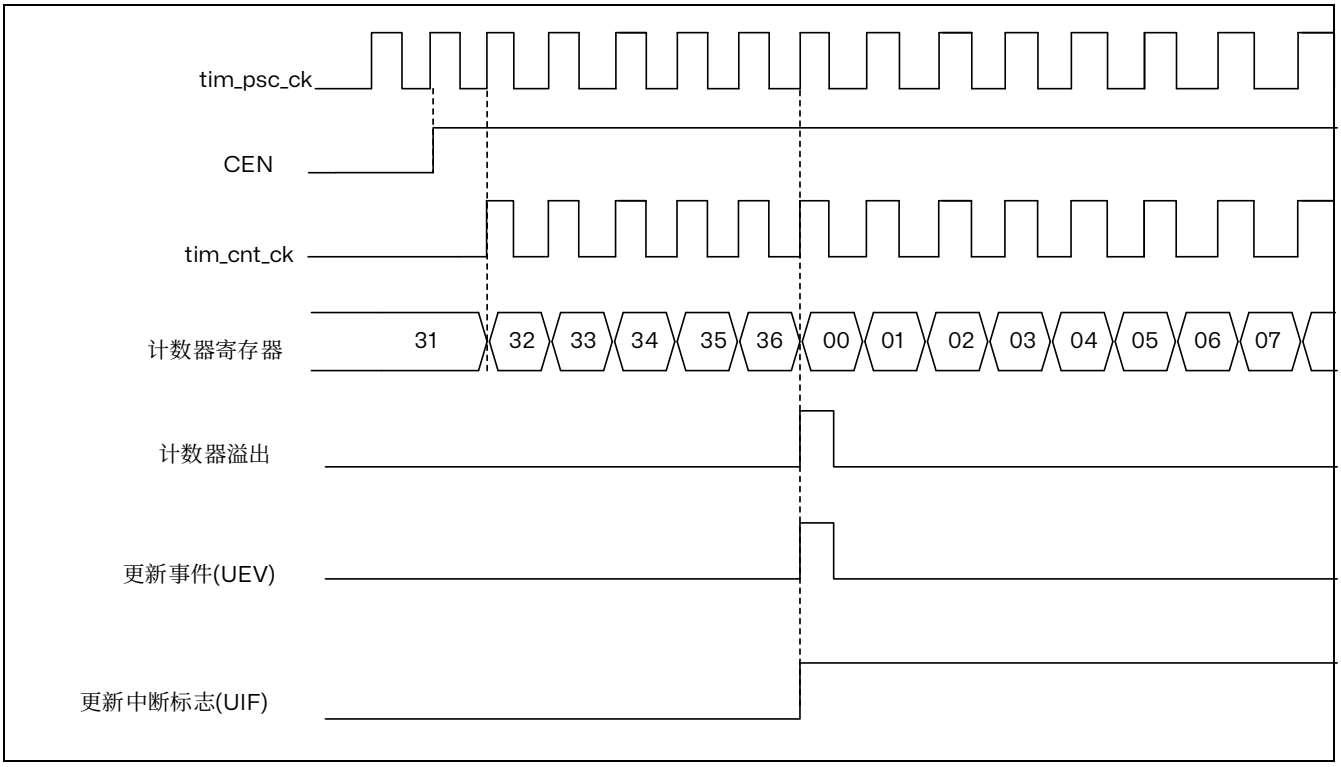



图 23.4 计数器时序图，内部时钟分频系数为 1

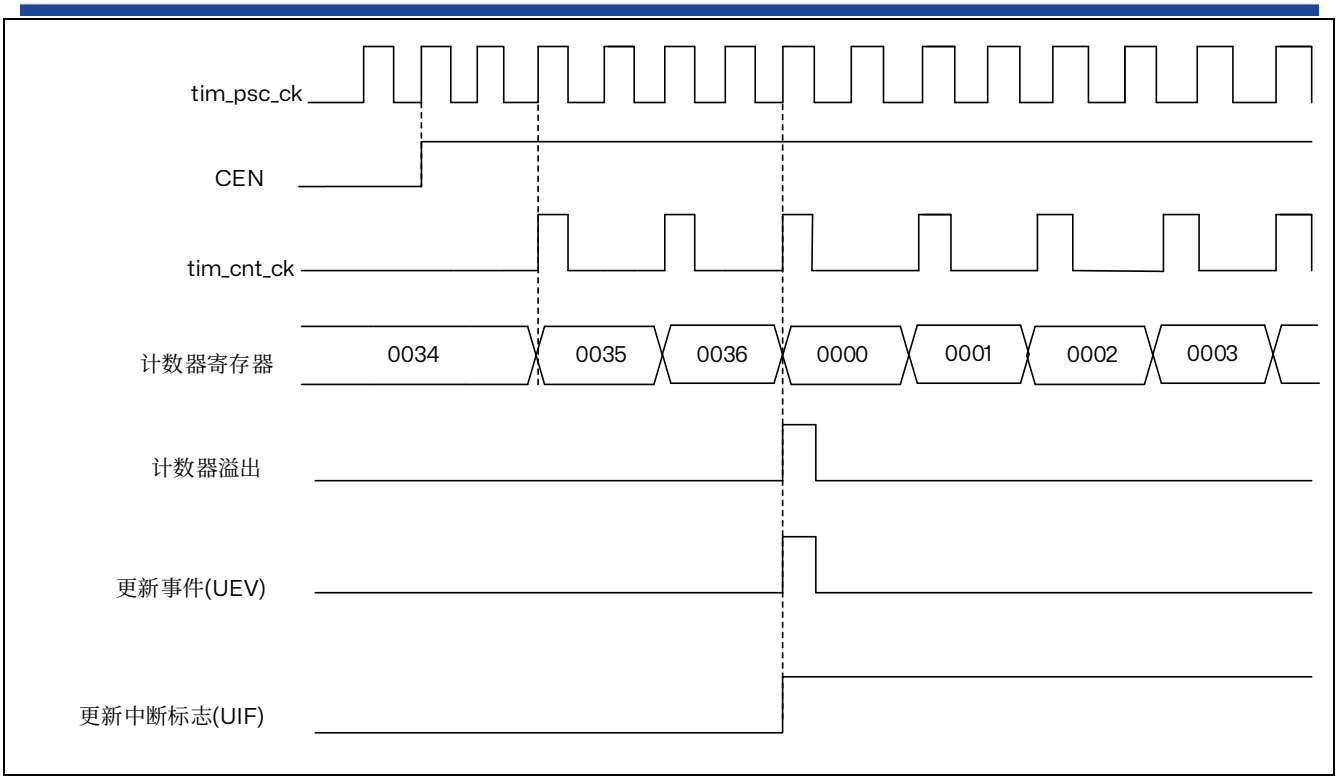



图 23.5 计数器时序图，内部时钟分频系数为 2

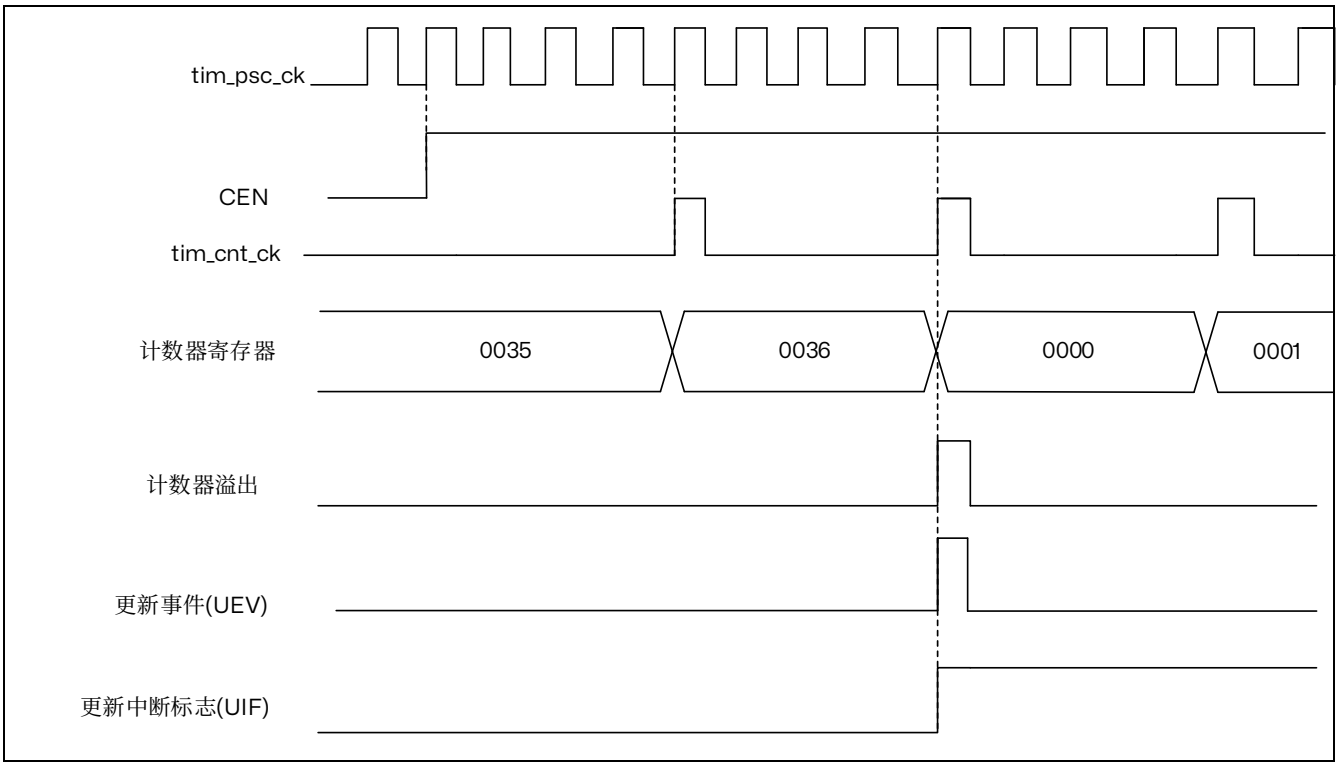



图 23.6 计数器时序图，内部时钟分频系数为 4

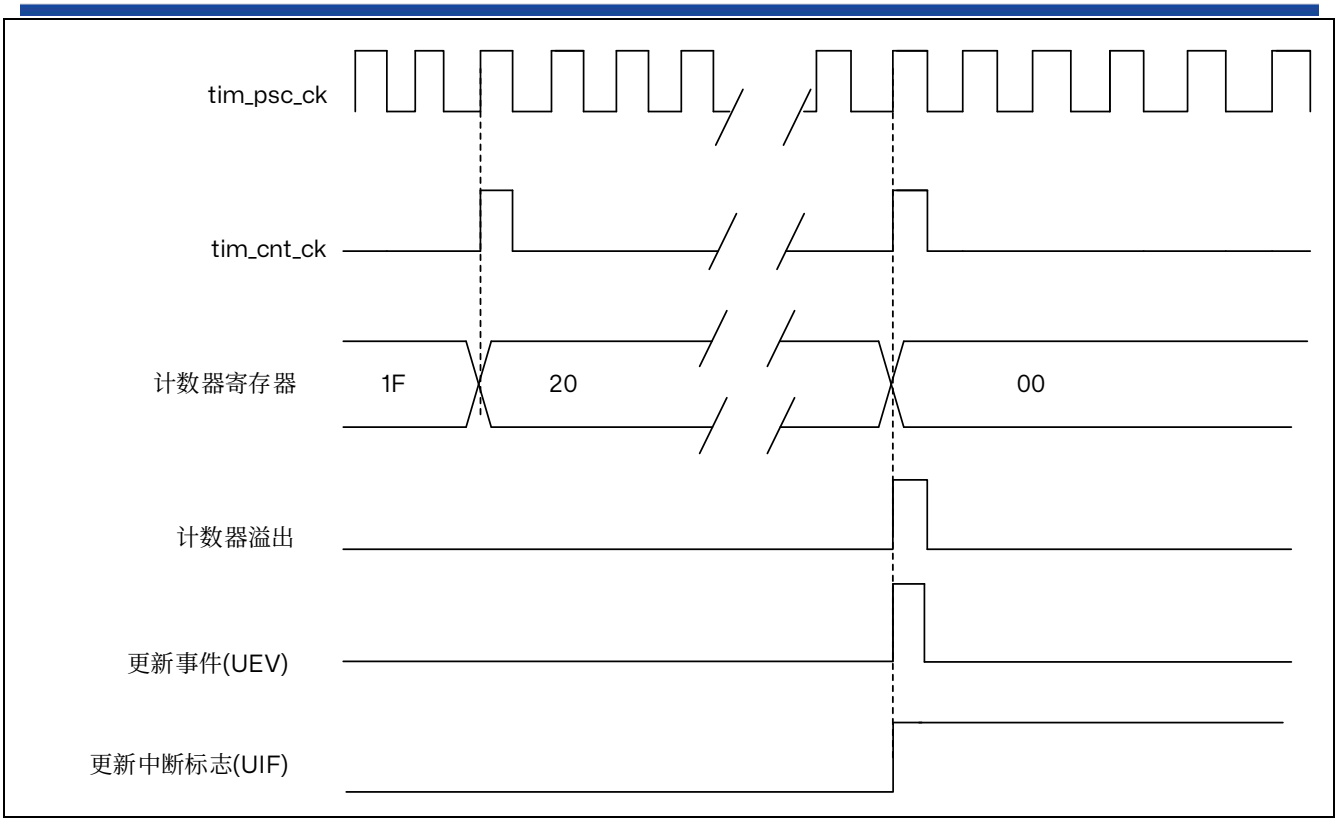



图 23.7 计数器时序图，内部时钟分频系数为 N

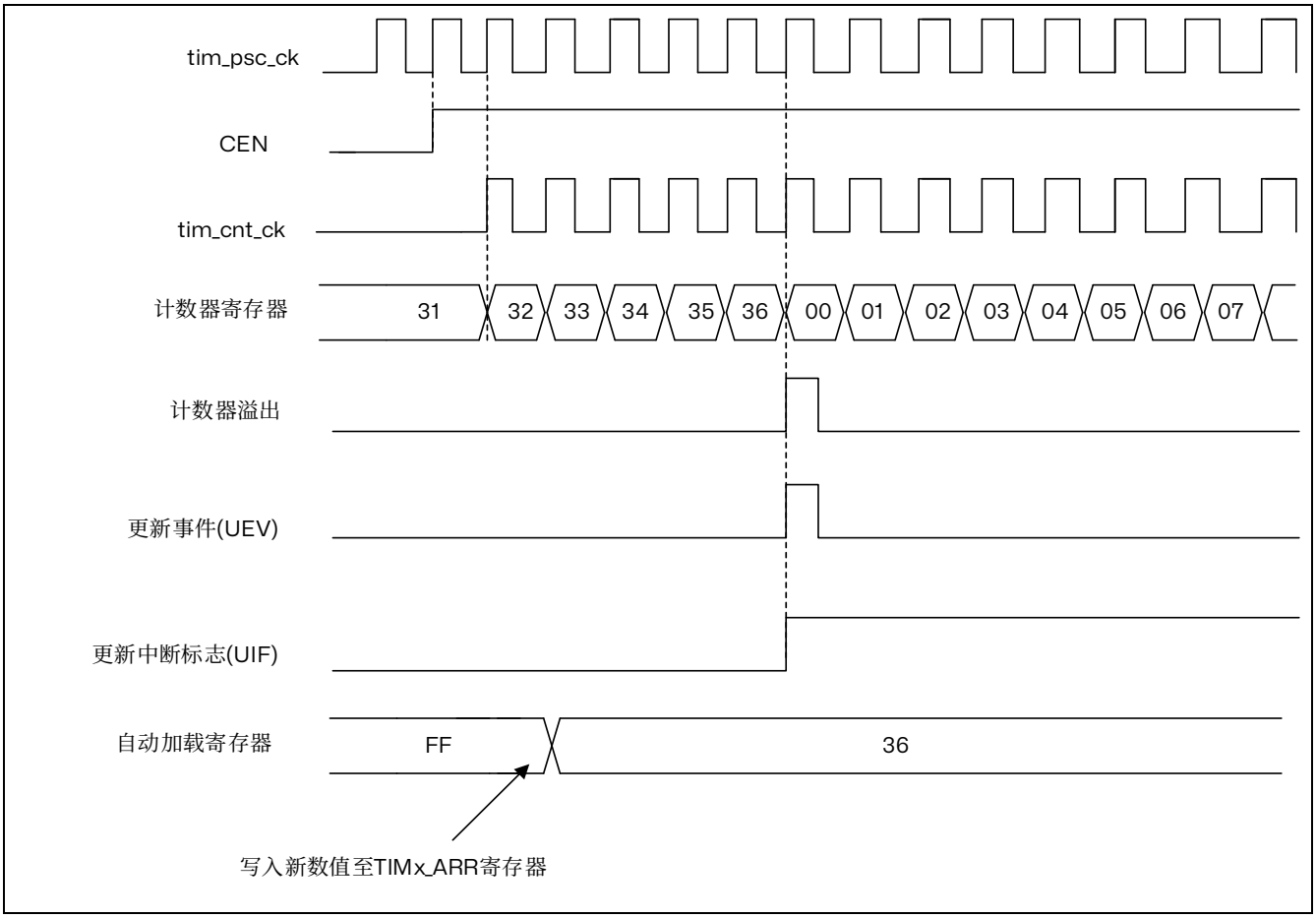



图 23.8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

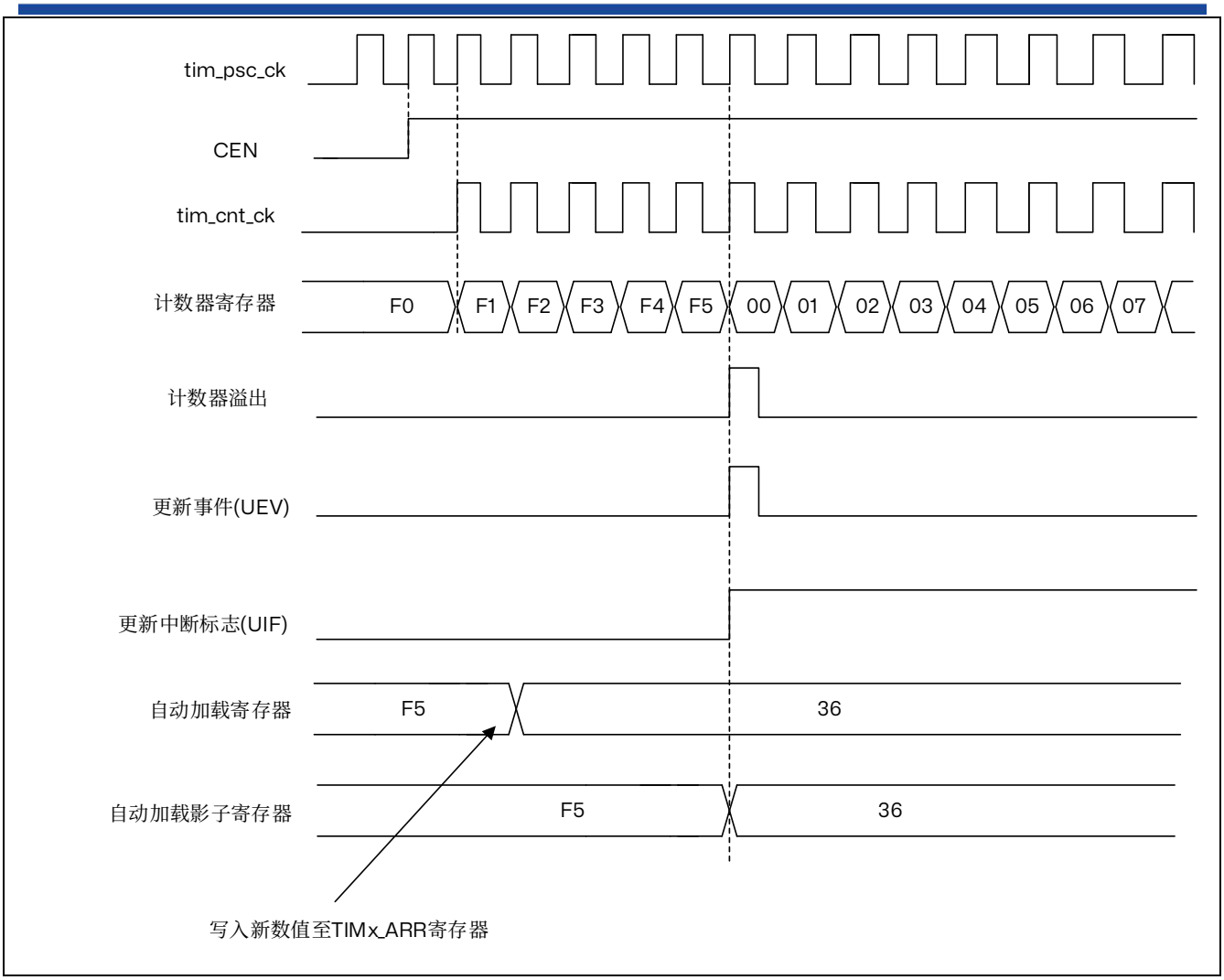



图 23.9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

**递减计数模式**

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

每次发生计数器下溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，自动重载寄存

器会在计数器重载之前得到更新，因此，下一个计数周期就是我们所希望的新的周期长度。

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

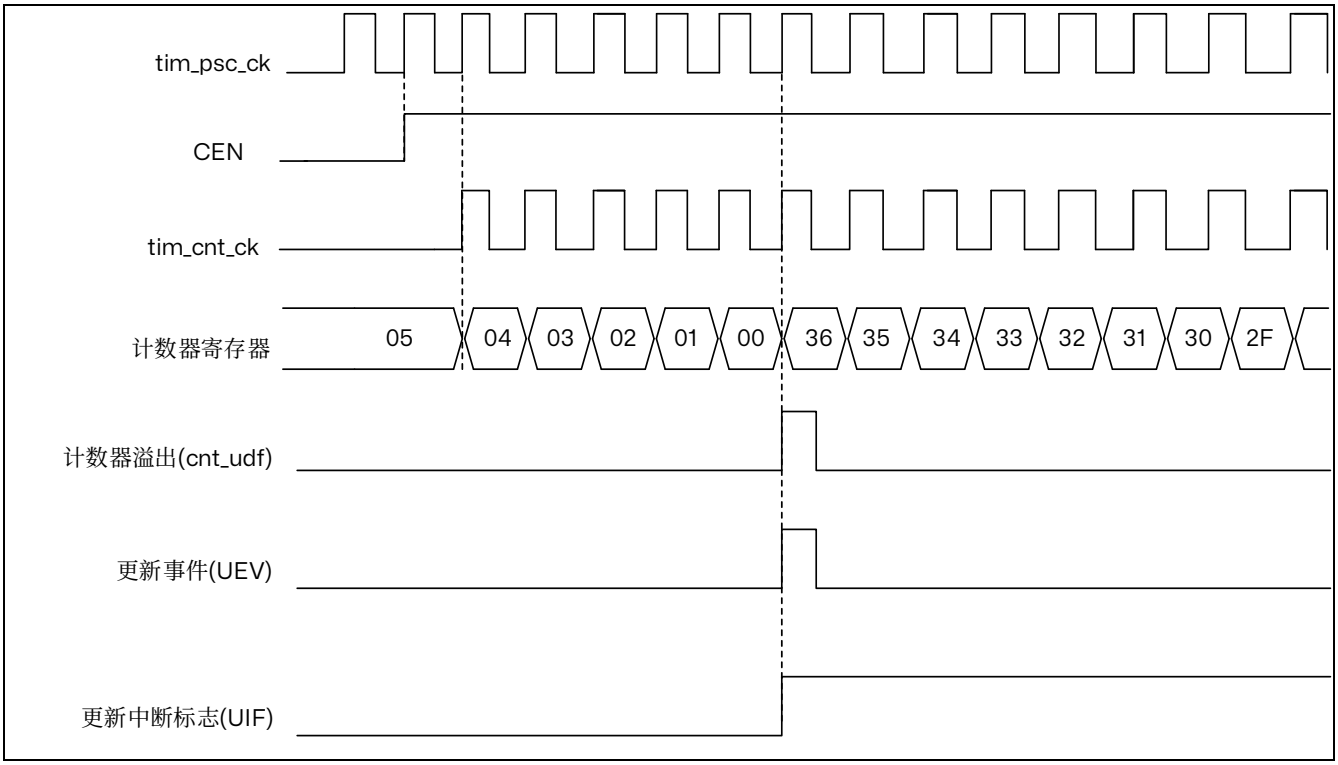



图 23.10 计数器时序图，内部时钟分频系数为 1

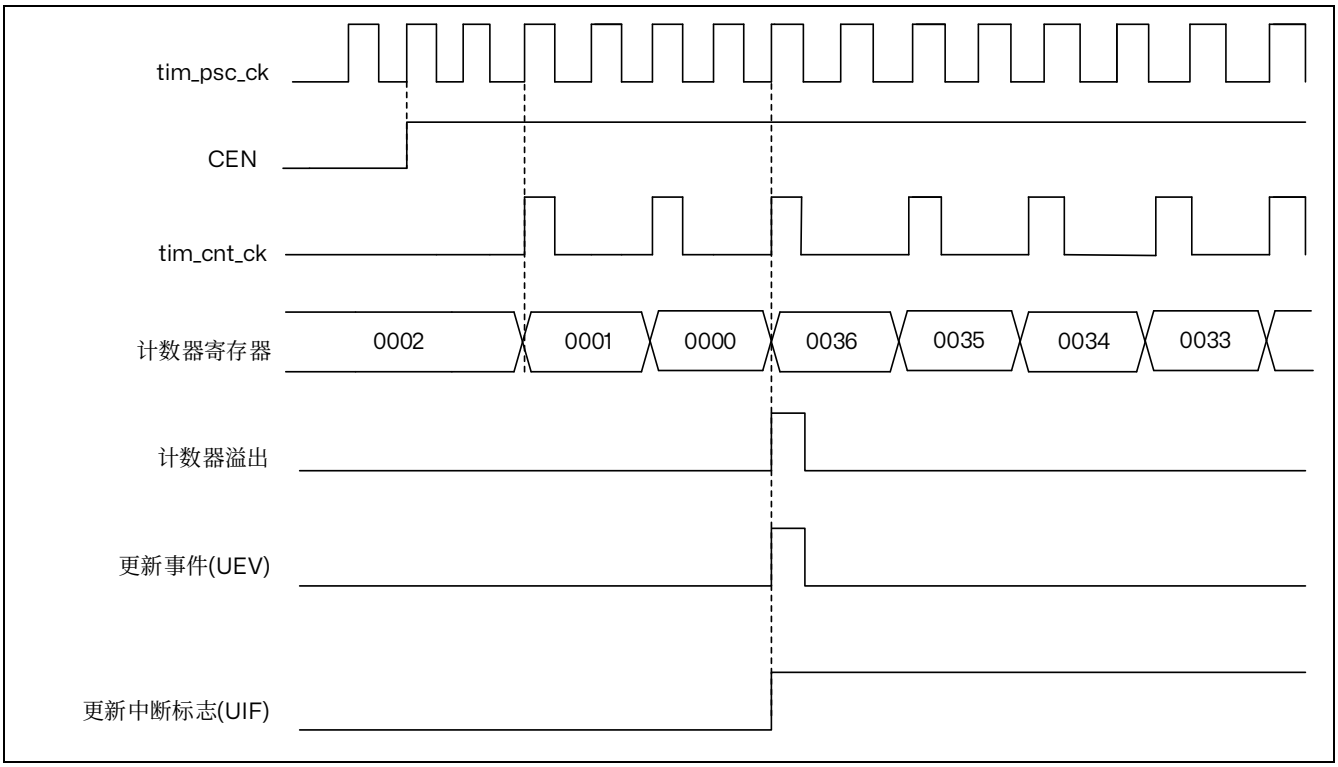



图 23.11 计数器时序图，内部时钟分频系数为 2

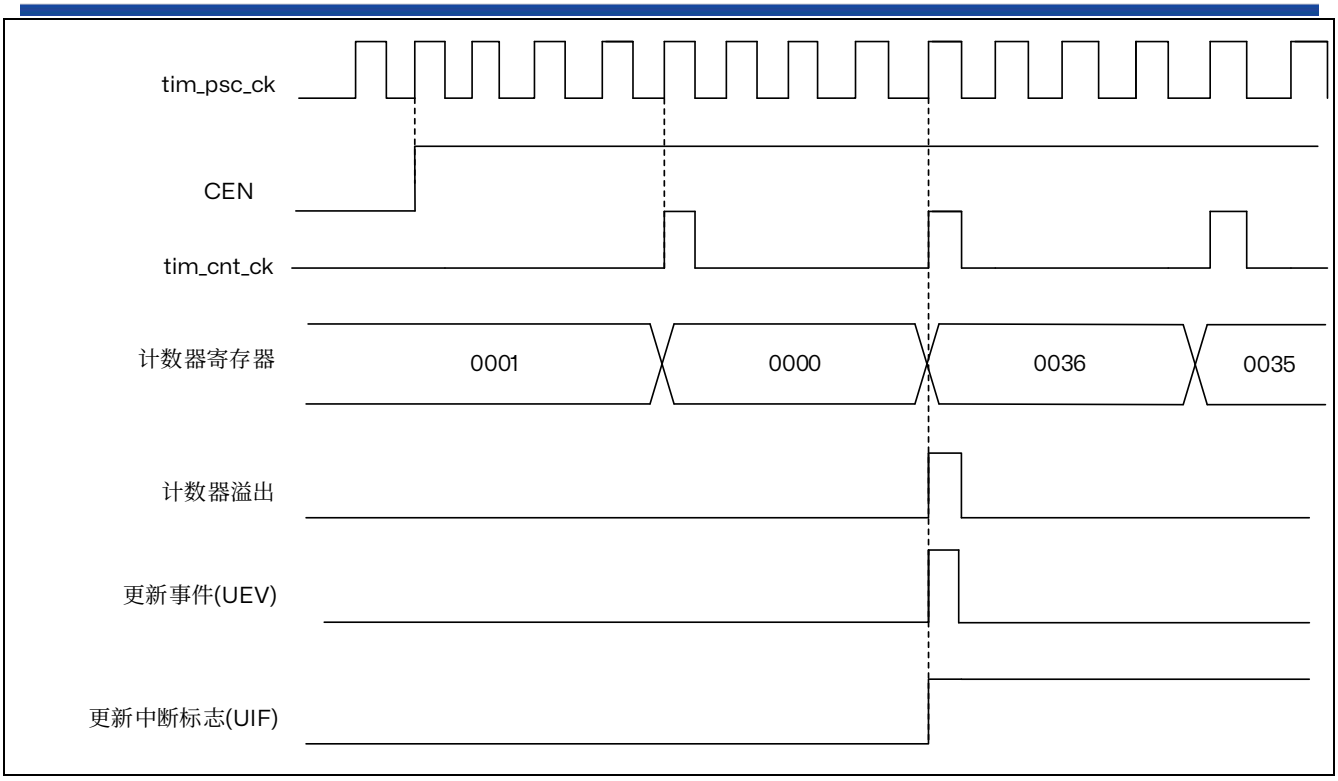



图 23.12 计数器时序图，内部时钟分频系数为 4

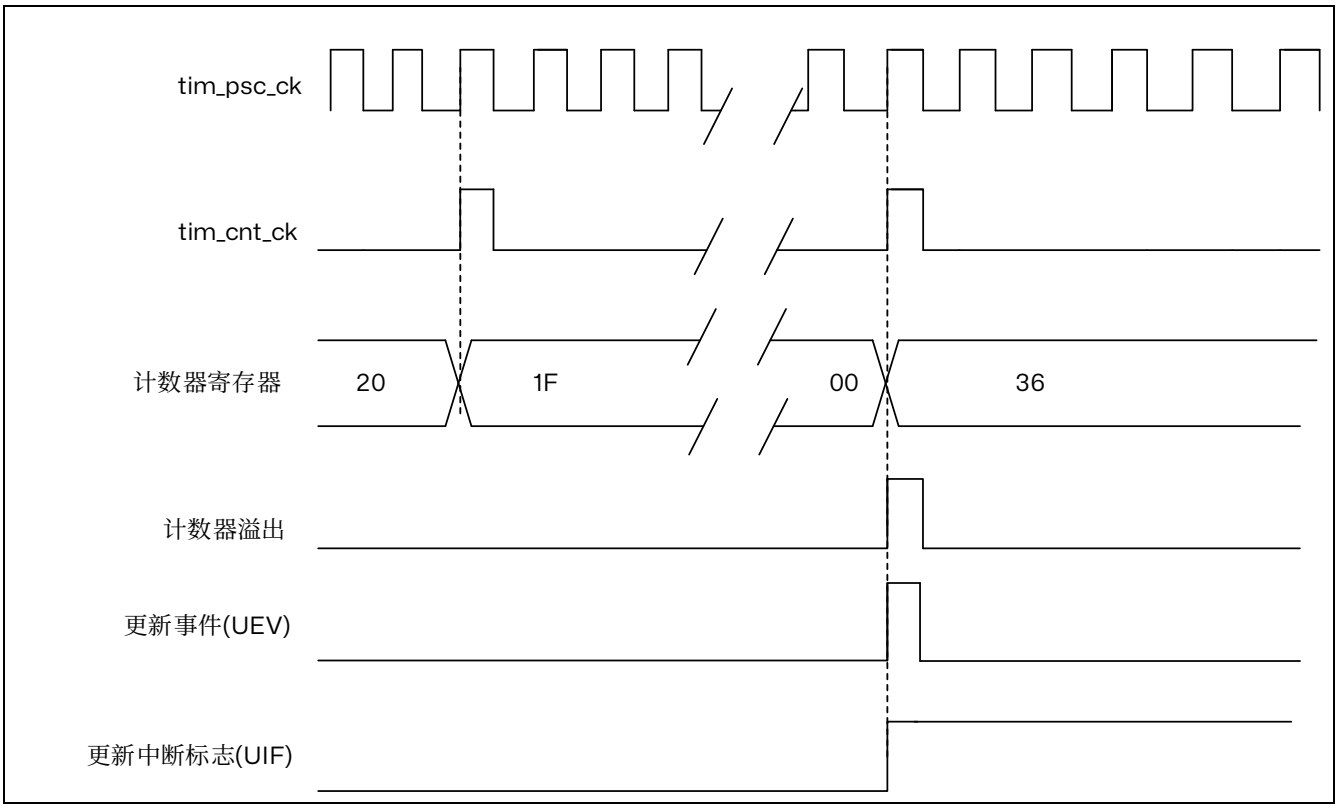



图 23.13 计数器时序图，内部时钟分频系数为 N

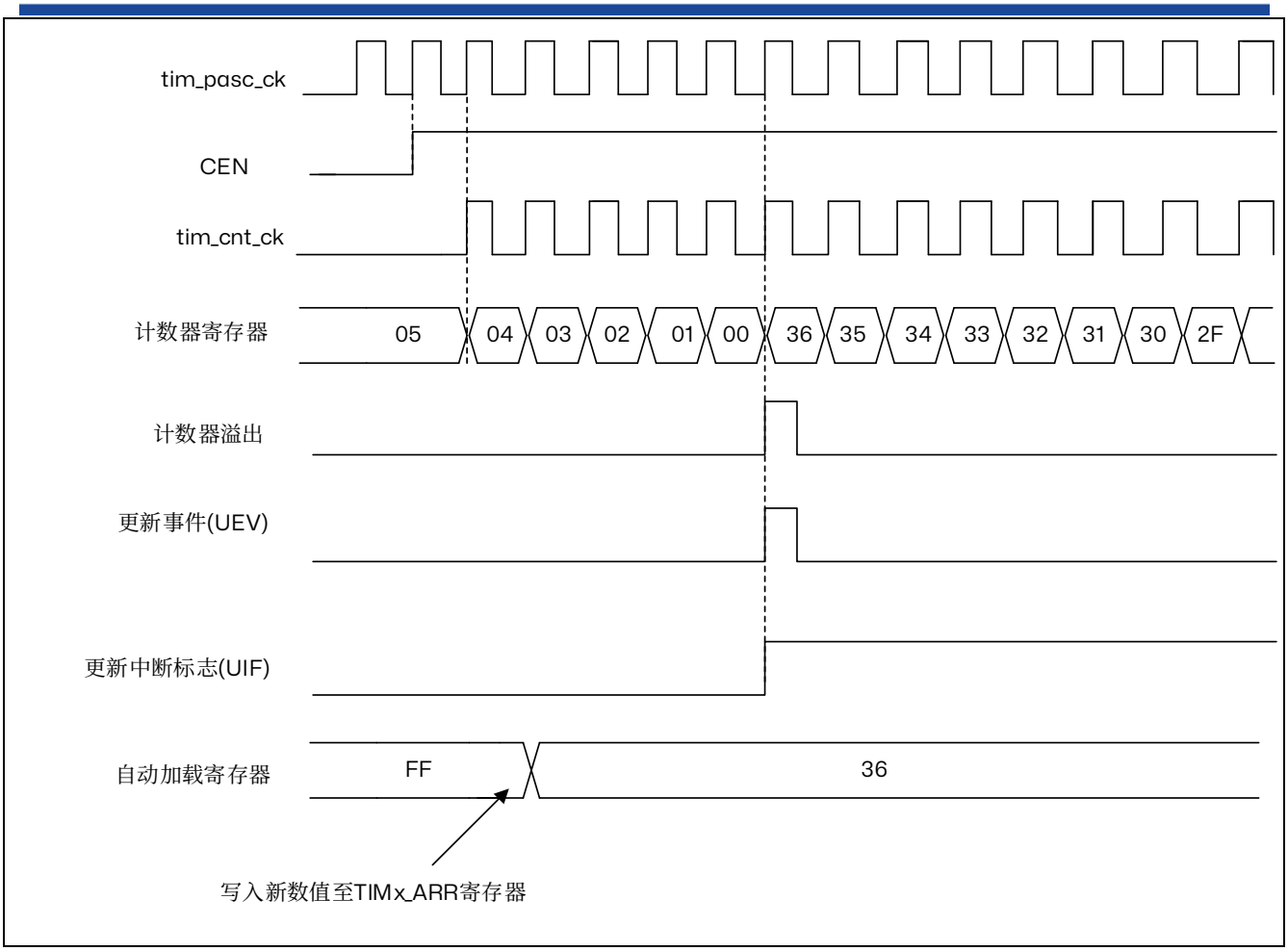



图 23.14 计数器时序图，更新事件

**中央对齐模式（递增/递减计数）**

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS = “01”）、计数器递增计数（中心对齐模式 2，CMS = “10”）以及计数器递增/递减计数（中心对齐模式 3，CMS = “11”）。

此模式下无法写入方向位（TIMx_CR1 寄存器中的 DIR 位）。而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

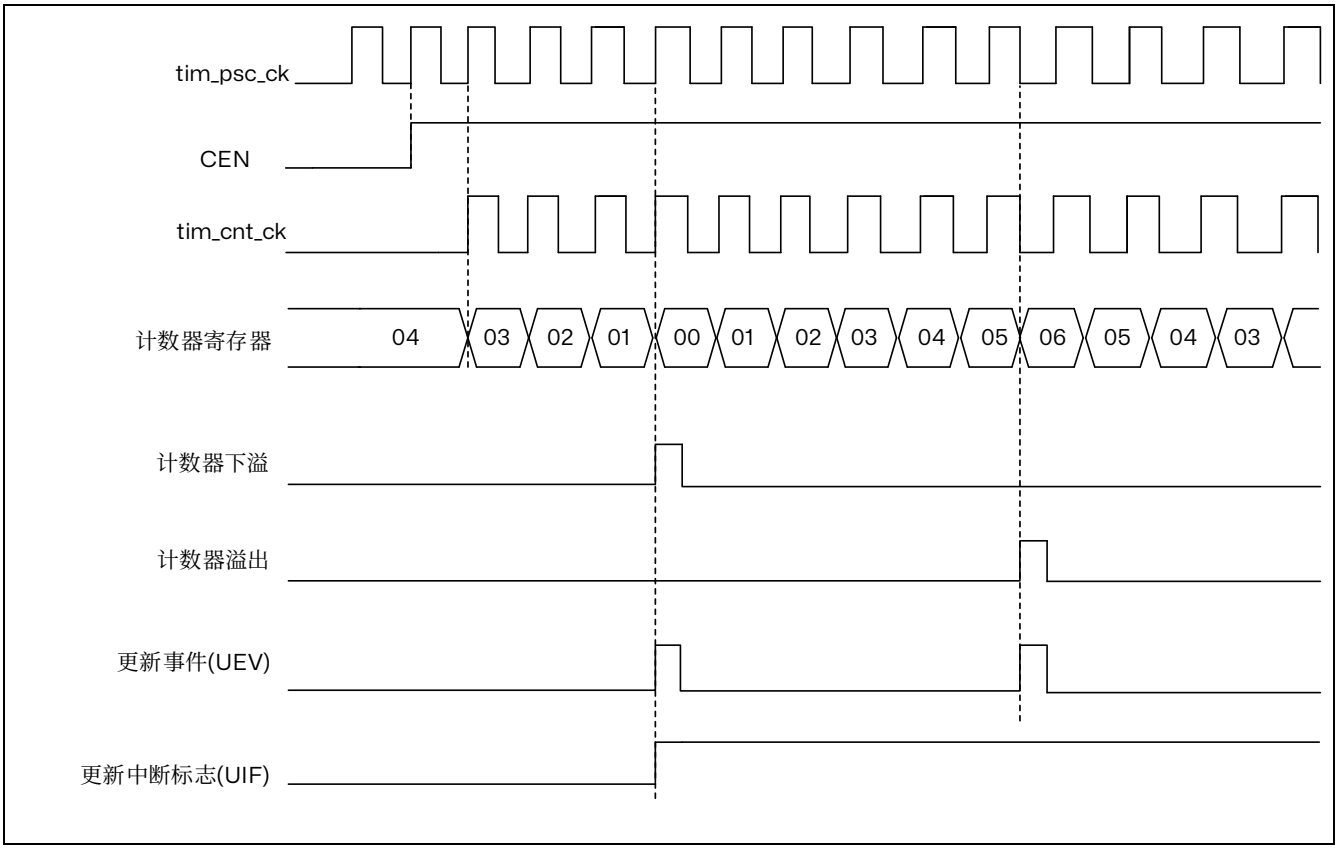
此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断请求或 DMA 请求）。这样一来，如果在发生

捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 当自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则自动重载寄存器在重载计数器之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。




1. 这里使用了中心对齐模式 1

图 23.15 计数器时序图，内部时钟分频系数为 1，TIMx_ARR=0x6

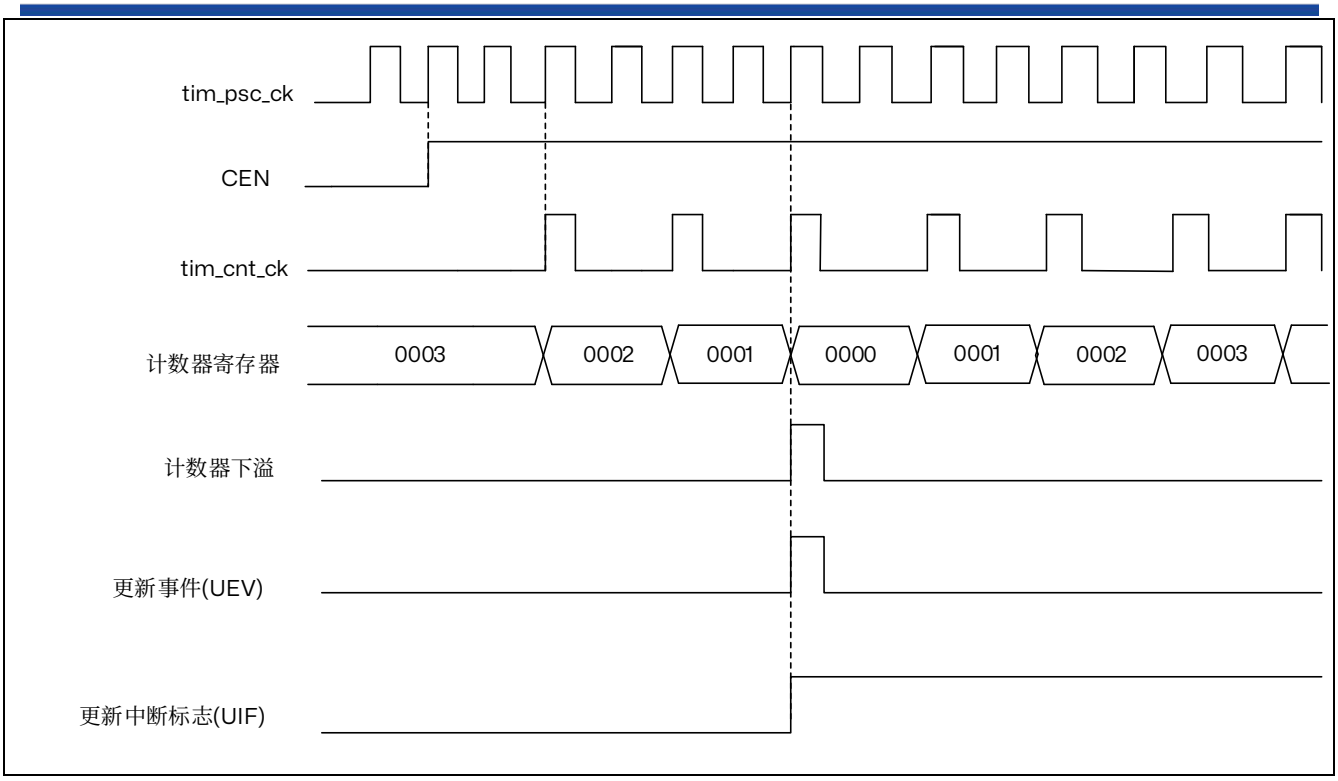



图 23.16 计数器时序图，内部时钟分频系数为 2

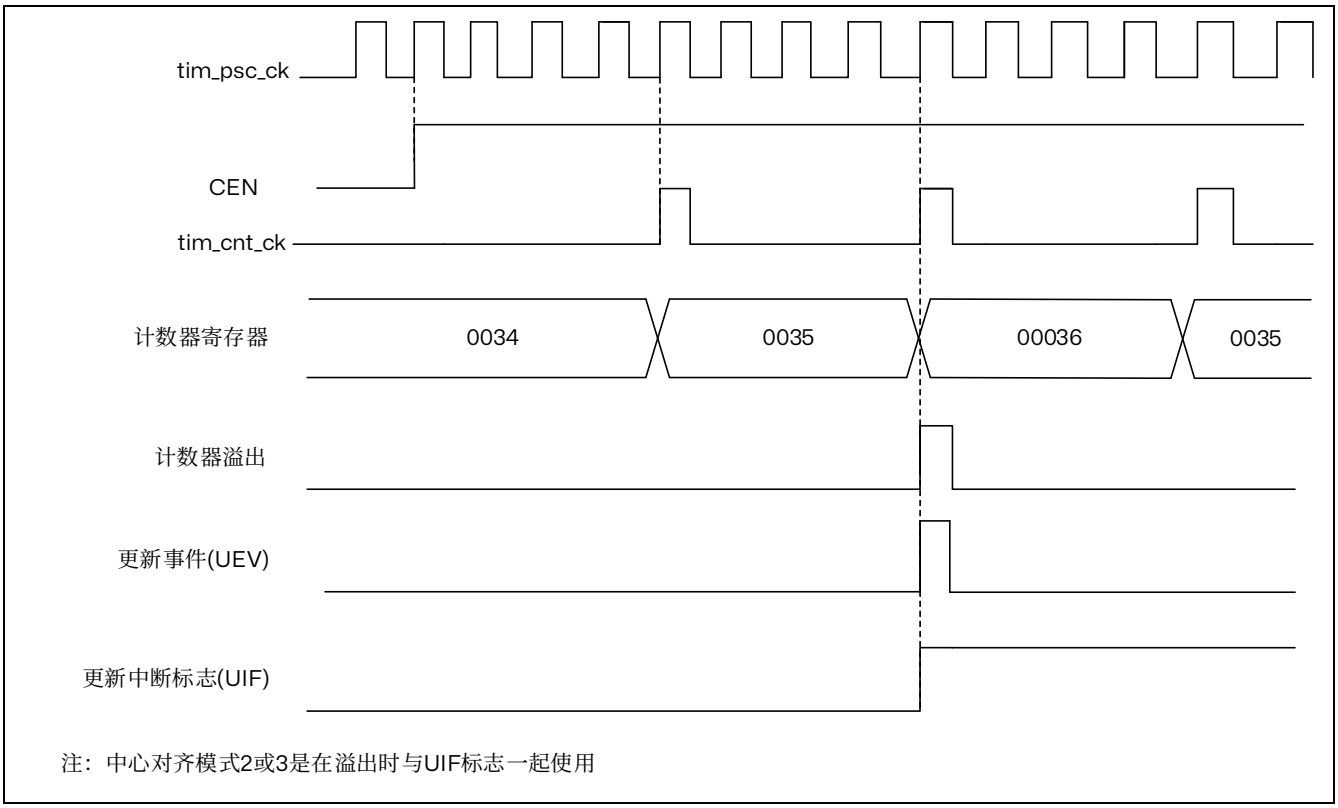



图 23.17 计数器时序图，内部时钟分频系数为 4，TIMx_ARR=0x36

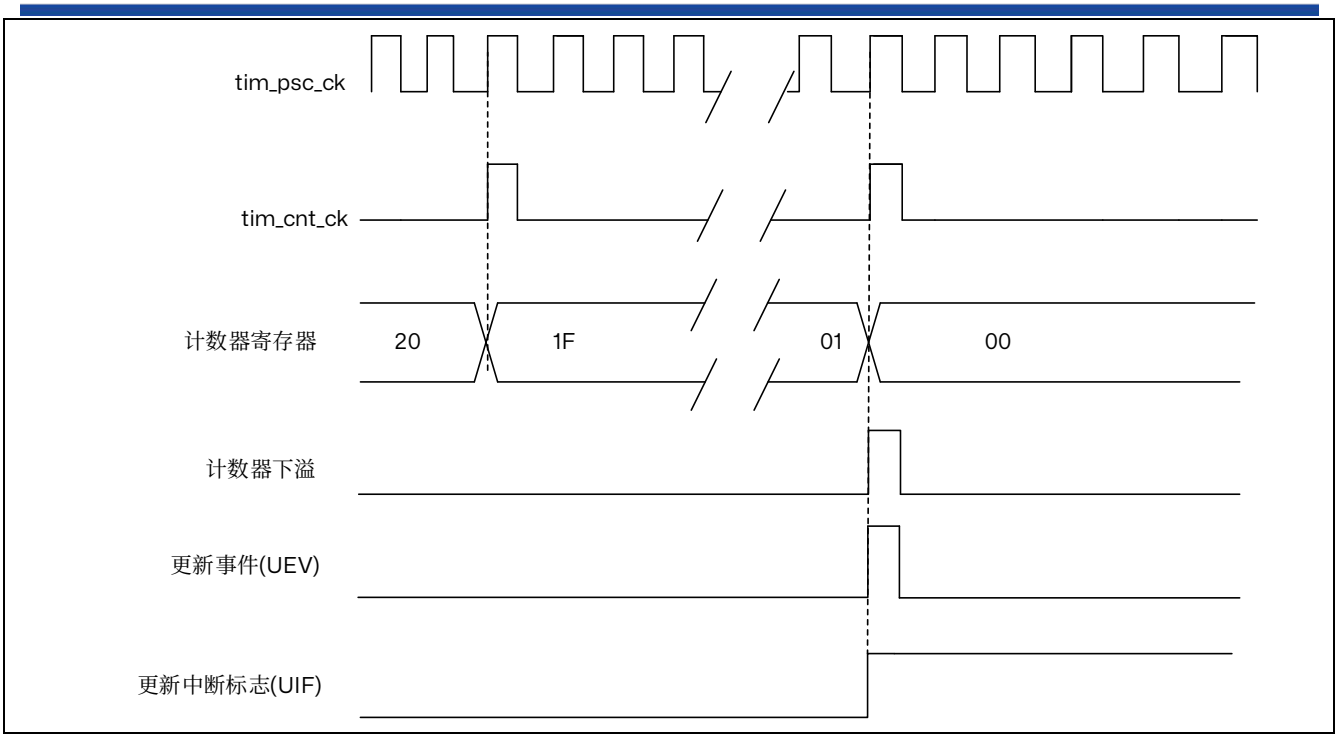



图 23.18 计数器时序图，内部时钟分频系数为 N

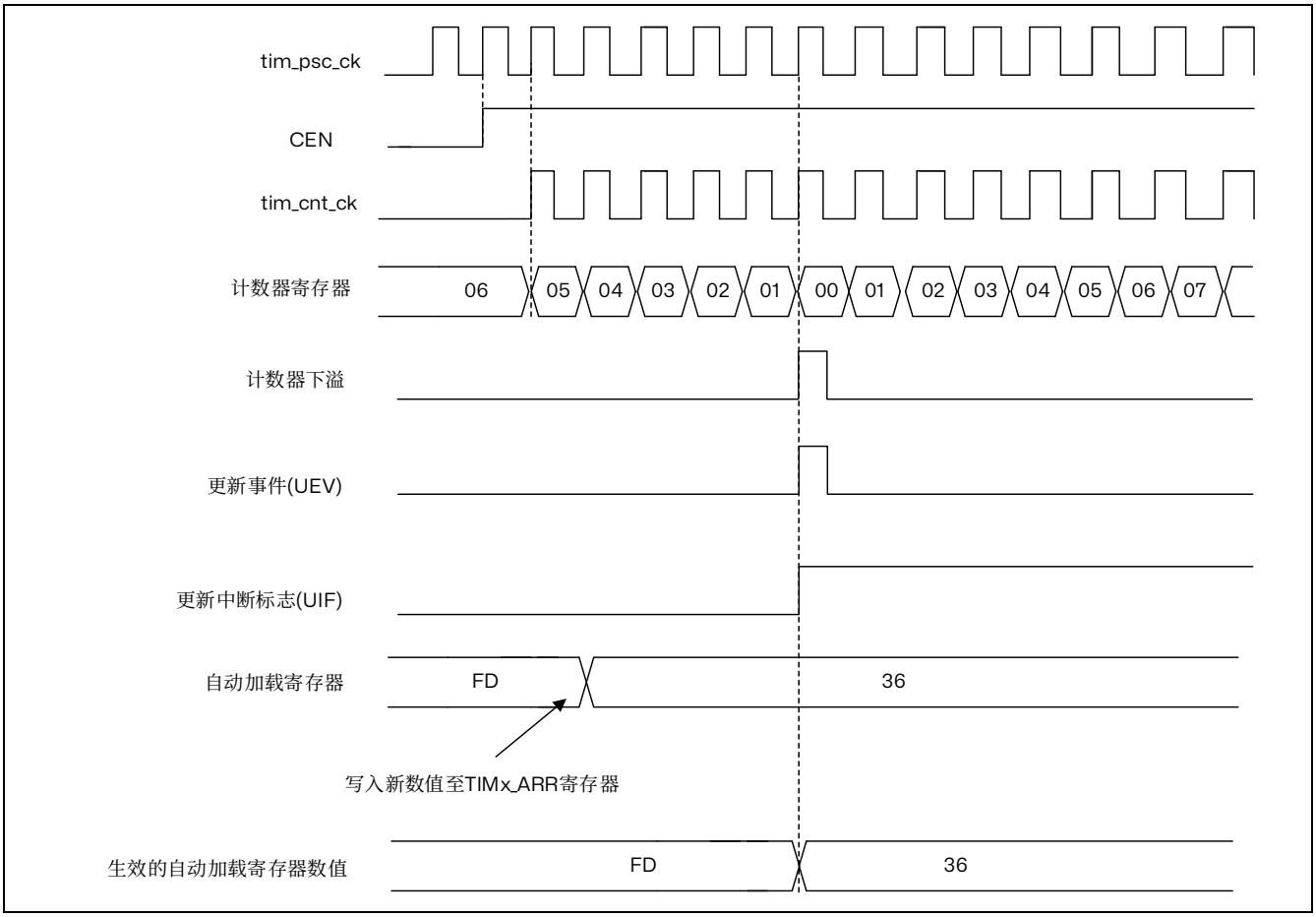



图 23.19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

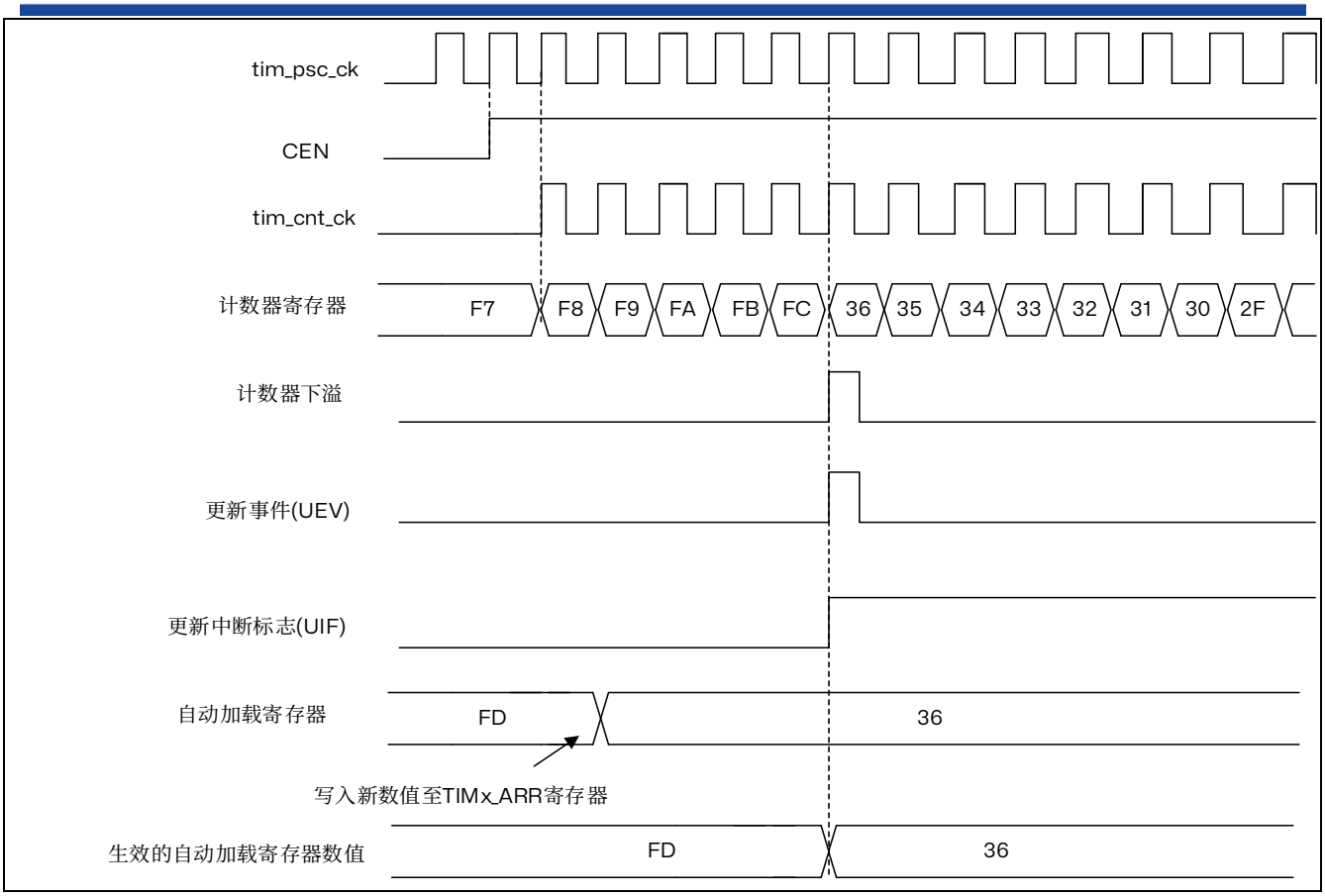



图 23.20 计数器时序图，ARPE=1 时的更新事件（计数器溢出）

### 23.4.5 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（tim_ker_ck）
- 外部时钟模式 1：外部输入引脚（tim_ti1 或 tim_ti2）
- 外部时钟模式 2：外部触发输入（tim_etr_in）
- 内部触发输入（tim_itr）：使用一个定时器作为另一个定时器的预分频器，例如可以将定时器 8 配置为定时器 2 的预分频器。

**内部时钟源（tim_ker_ck）**

如果禁止从模式控制器（TIMx_SMCR 寄存器中 SMS=000），则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 tim_ker_ck 提供。

下图显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

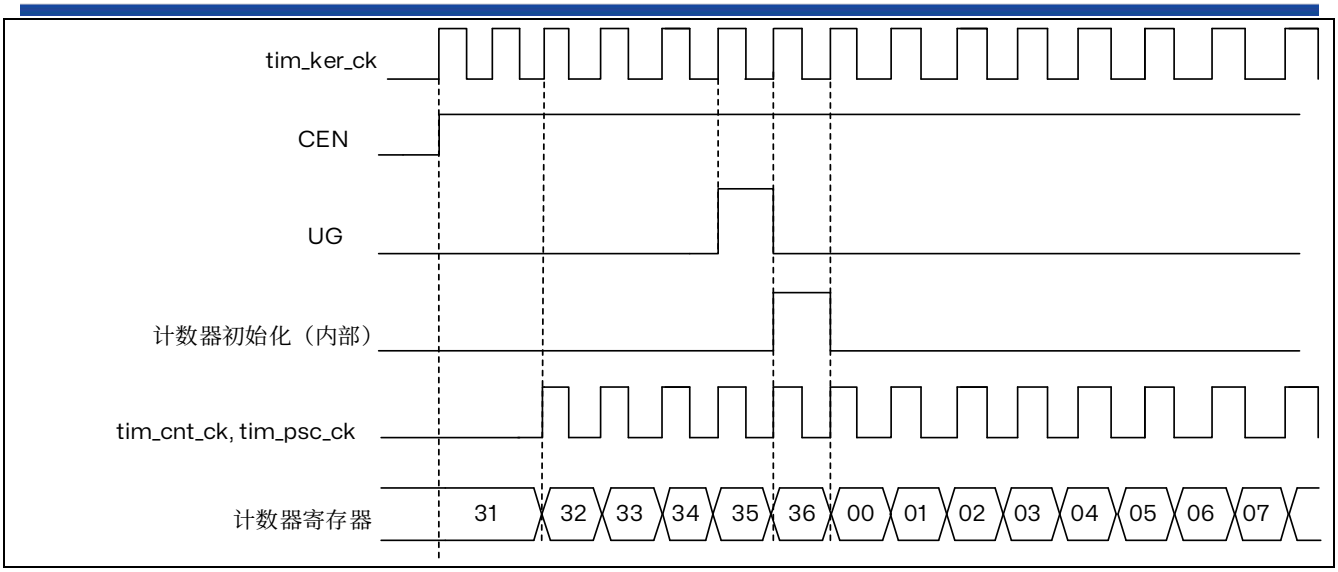



图 23.21 一般模式下的控制电路，内部时钟分频系数为 1

**外部时钟源模式 1**

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

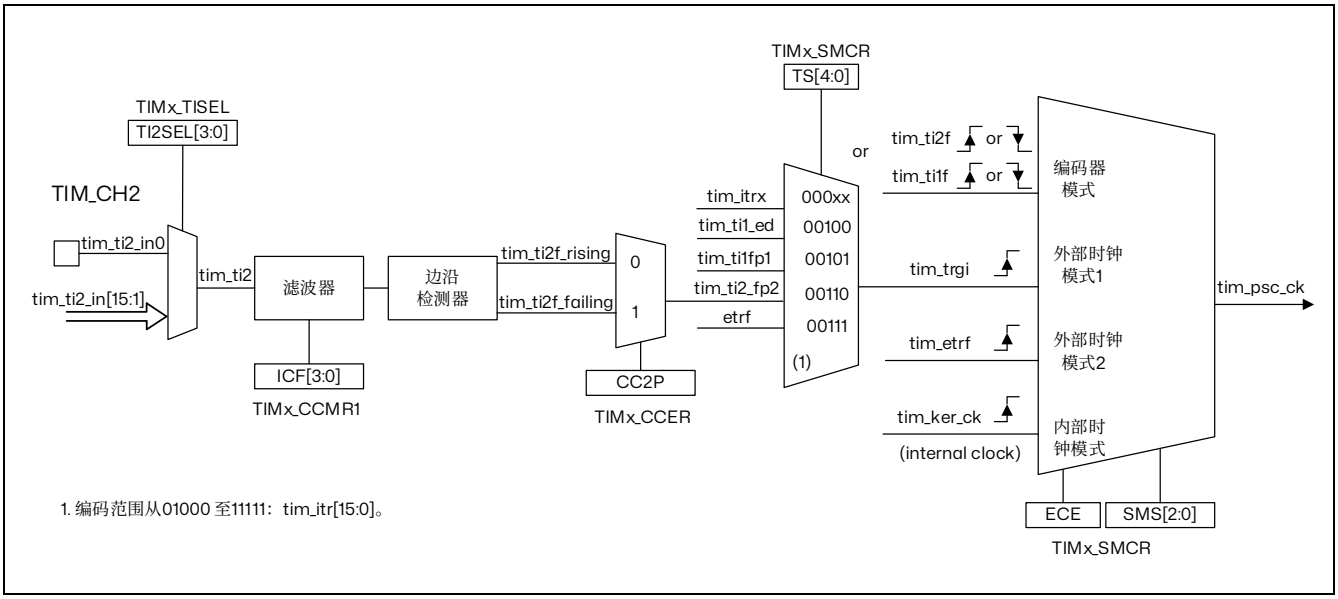



图 23.22 tim_ti2 外部时钟连接例子

例如，要使递增计数器在 tim_ti2 输入出现上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_TISEL 寄存器中使用 TI2SEL[3:0]位选择合适的 tim_ti2_in[15:0]源（内部或外部）。
2. 通过在 TIMx_CCMR1 寄存器中写入 CC2S=“01”来配置通道 2，使其能够检测 tim_ti2 输入的上升沿。
3. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0]位来配置输入滤波时间（如果不需要任何滤波，请保持 IC2F=0000）。

*注：由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

4. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。

5. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
6. 通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 tim_ti2 作为输入源。
7. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

当 tim_ti2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。
tim_ti2 的上升沿与实际计数器时钟之间的延迟是由于 tim_ti2 输入的重新同步电路引起的。

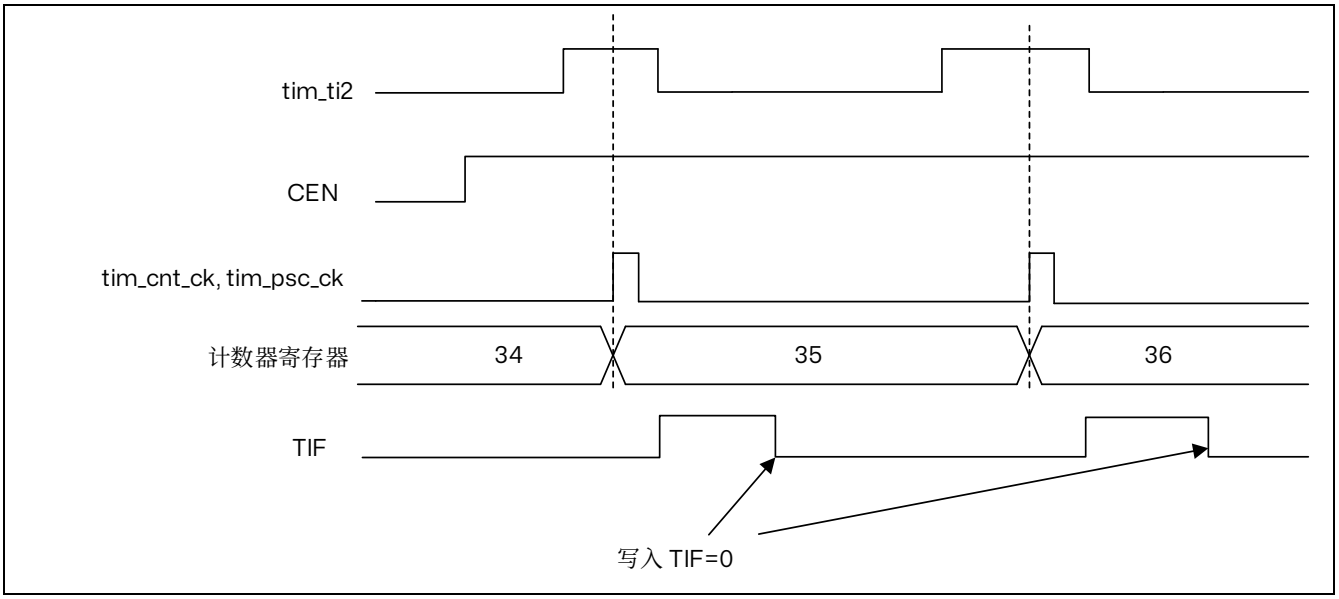



图 23.23 外部时钟模式 1 下的控制电路

**外部时钟源模式 2**

通过在 TIMx_SMCR 寄存器中写入 ECE=1 选择此模式。
计数器可在外部触发输入 ETR 出现上升沿或下降沿时计数。
下图是外部触发输入的框图

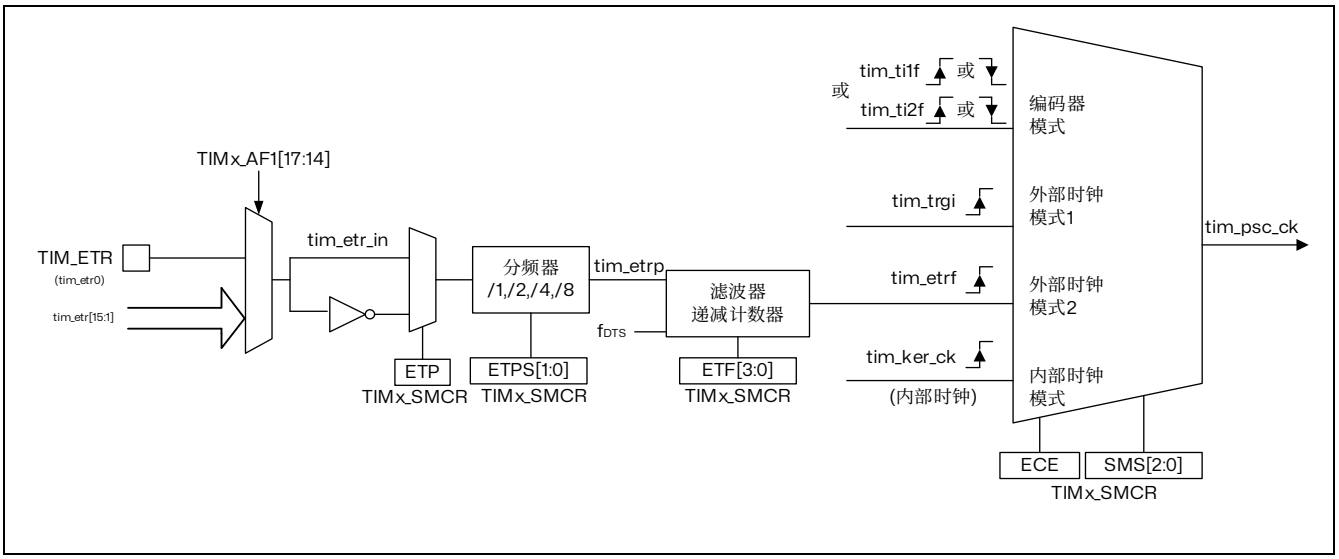



图 23.24 外部触发输入框图

例如，要使递增计数器在 ETR 每出现 2 个上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_AF1 寄存器中使用 ETRSEL[3:0]位选择合适的 tim_etr_in 源（内部或外部）。

2. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
3. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器。
4. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 tim_etr_in 引脚的上升沿检测。
5. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

tim_etr_in 每出现 2 个上升沿，计数器计数一次。

在 tim_etr_in 的上升沿和计数器实际时钟之间的延时取决于在 tim_etrp 信号端的重新同步电路。因此，计数器可以正确捕获的最大频率最高为 TIMxCLK 频率的¼。当 ETRP 信号更快时，用户应通过适当的 ETPS 预分频器设置对外部信号进行分频。

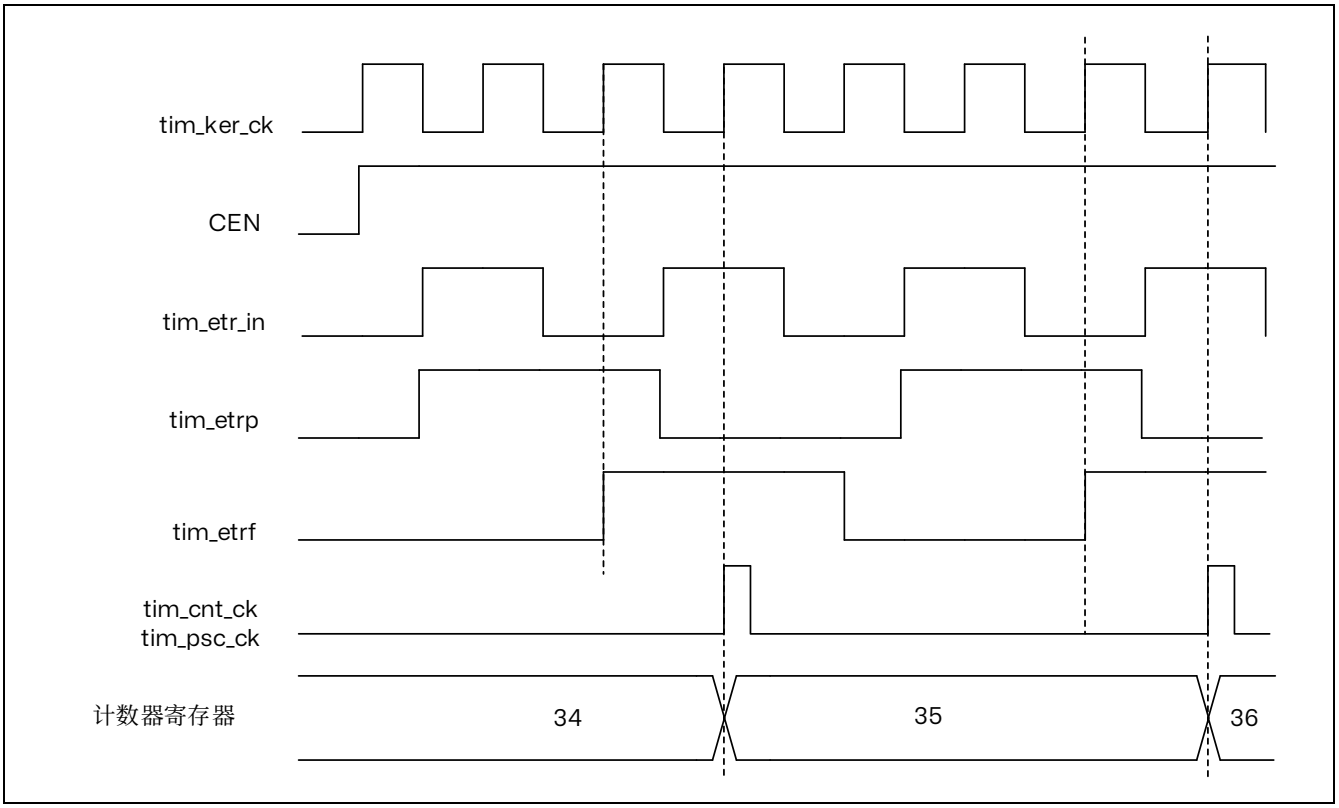


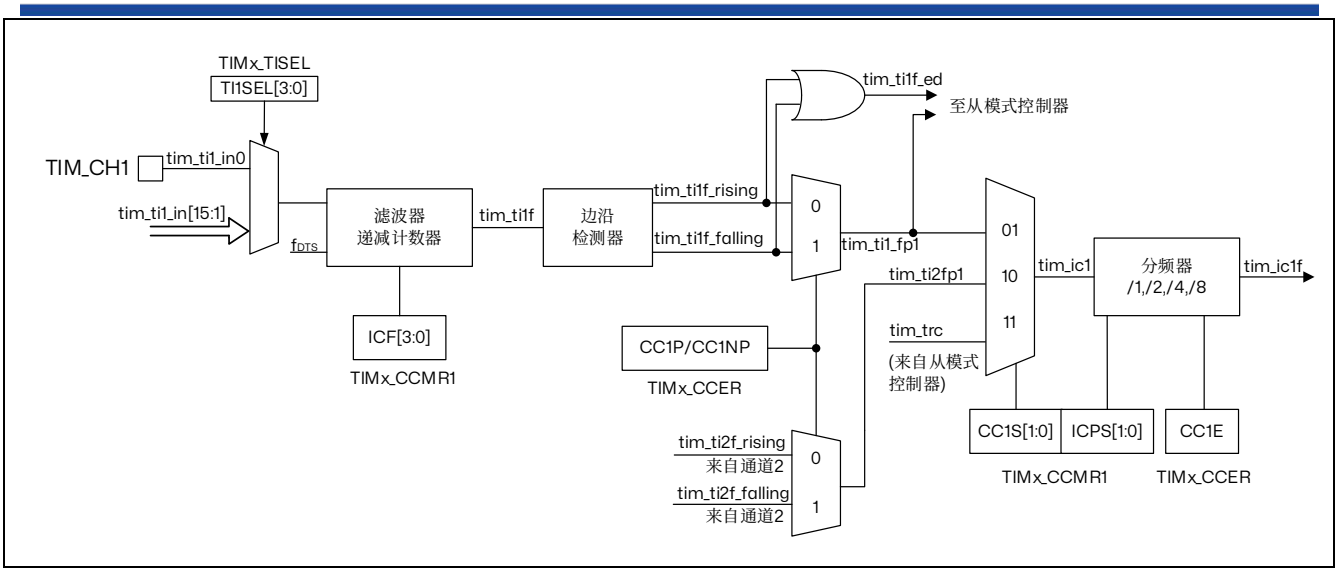

图 23.25 外部时钟模式 2 下的控制电路

### 23.4.6 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

下图概括介绍了一个捕获/比较通道。

输入阶段对相应的 tim_tix 输入进行采样，生成一个滤波后的信号 tim_tixf。然后，带有极性选择功能的边沿检测器生成一个信号（tim_tixfpy），该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频（ICxPS），而后再进入捕获寄存器。




输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 23.26 捕获/比较通道（如：通道 1 输入部分）

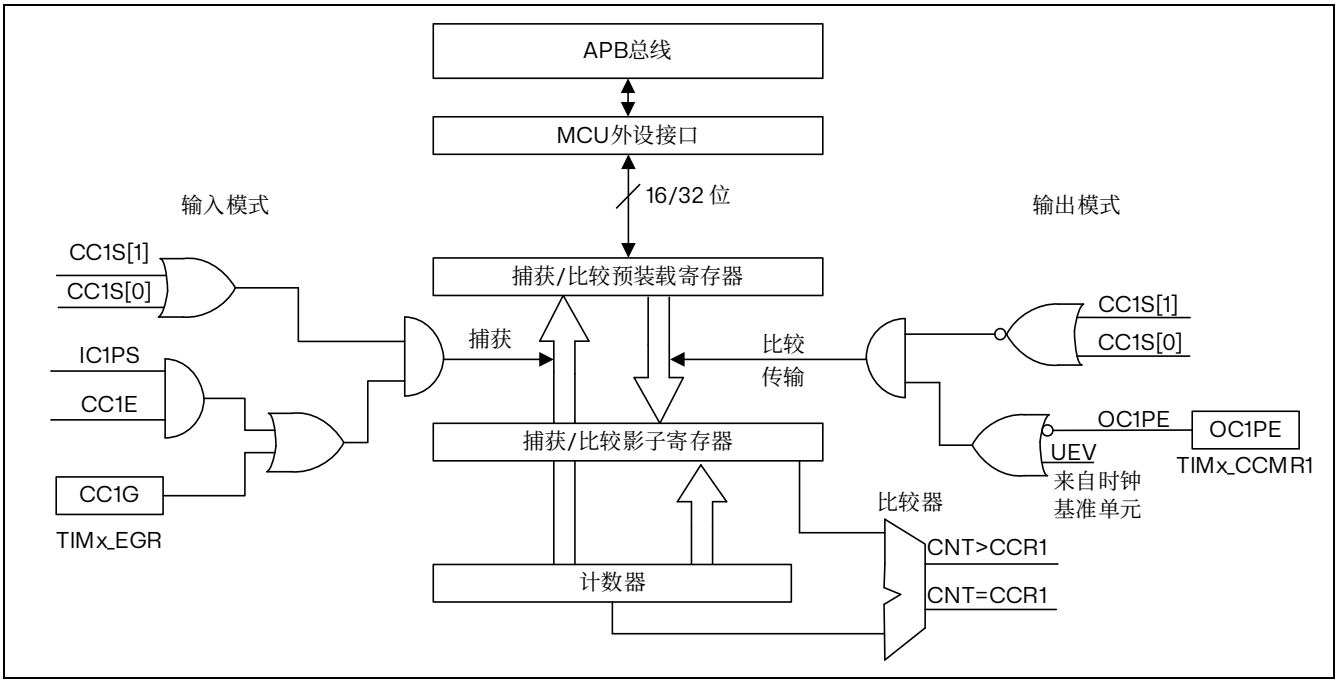



图 23.27 捕获/比较通道 1 的主电路

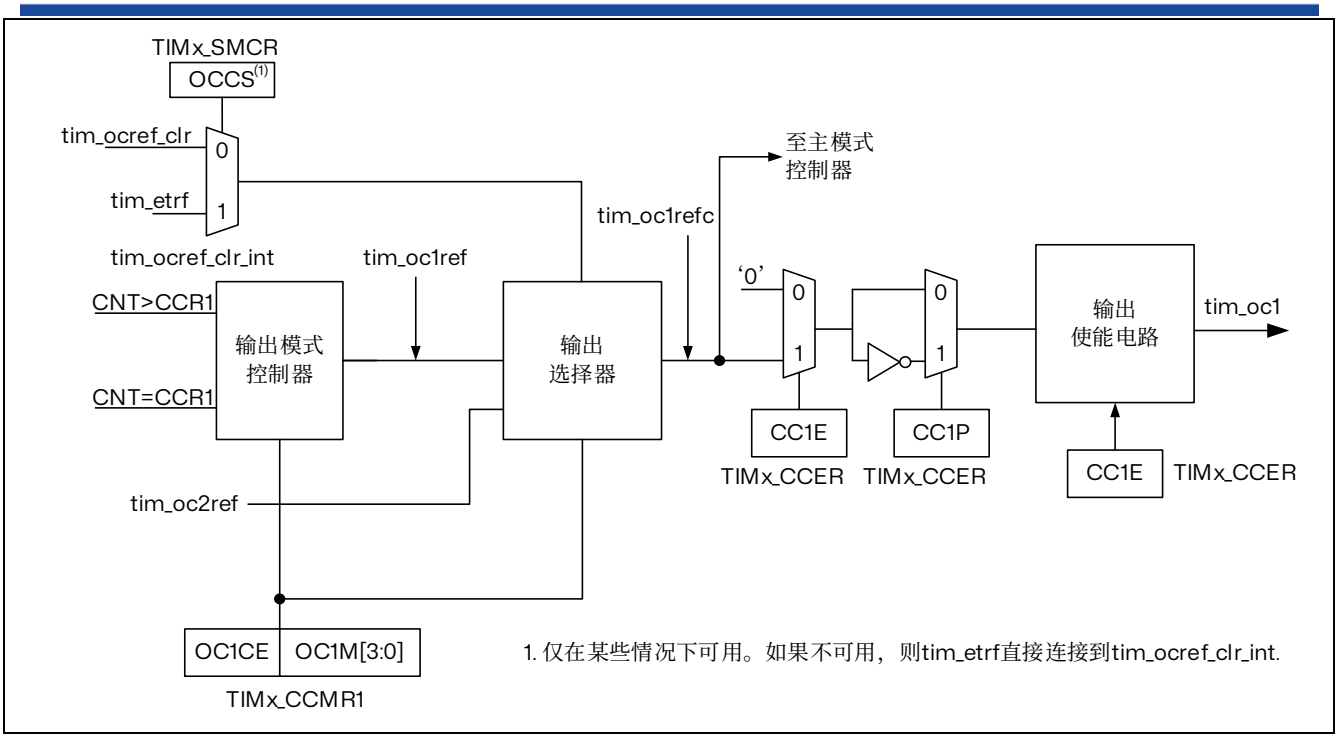



图 23.28 捕获/比较通道的输出阶段（通道 1，通道 2、3 和 4 同上）

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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

### 23.4.7 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器(TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入 0 后会将其清零。

以下示例说明了如何在 tim_ti1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 通过在 TIMx_TISEL 寄存器中使用 TI1SEL[3:0]位选择合适的 tim_tix_in[15:0]源（内部或外部）。
2. 选择有效输入：TIMx_CCR1 必须连接到 tim_ti1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
3. 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 tim_tix 输入之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 $f_{DTS}$ 频率采样）后，可以确认 tim_ti1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。
4. 通过向 TIMx_CCER 寄存器中的 CC1P 位和 CC1NP 位写入 0，选择 tim_ti1 通道的有效转换边沿

（本例中为上升沿）。

5. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PSC 位写入 00）。
6. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
7. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成一个中断。
- 根据 CC1DE 位生成一个 DMA 请求。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

*注：通过软件将*TIMx_EGR *寄存器中的相应*CCxG *位置*1 *可生成*IC *中断请求或*DMA *请求。*

### 23.4.8 PWM 输入模式

此模式允许测量连接到单个 tim_tix 输入的 PWM 信号的周期和占空比：

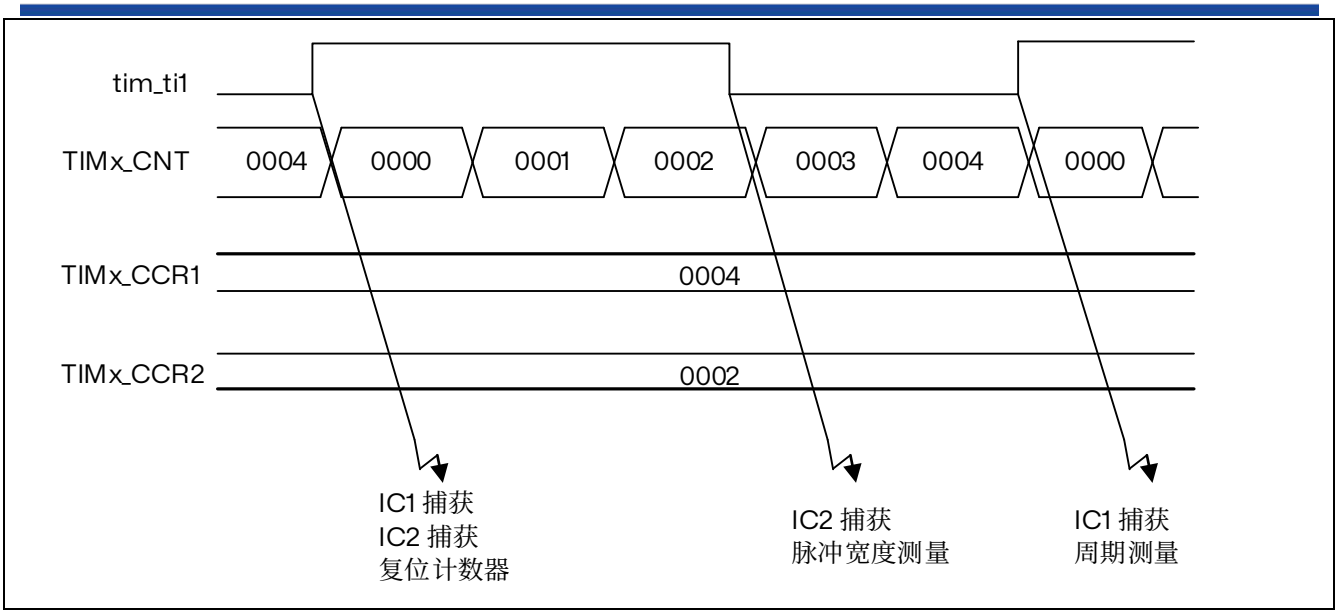
- TIMx_CCR1 寄存器保持周期值（两个连续上升沿之间的间隔）。
- TIMx_CCR2 寄存器保持脉冲宽度（两个连续上升沿和下降沿之间的间隔）。

此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 ICx 信号被映射至同一个 tim_tix 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

可通过以下步骤测量施加在 tim_ti1 上的 PWM 信号的周期和脉冲宽度：

1. 通过在 TIMx_TISEL 寄存器中使用 TI1SEL[3:0]位选择合适的 tim_tix_in[15:0]源（内部或外部）。
2. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 tim_ti1）。
3. 选择 tim_ti1fp1 的有效极性（用于 TIMx_CCR1 中的捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
4. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 tim_ti1）。
5. 选择 tim_ti1fp2 的有效极性（用于 TIMx_CCR2 中的捕获）：向 CC2P 位写入“1”和 CC2NP 位写入“0”（下降沿有效）。
6. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 101（选择 tim_ti1fp1）。
7. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 100。
8. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。




PWM 输入模式只能与 TIMx_CH1/TIMx_CH2 信号一起使用，因为只有 tim_ti1fp1 和 tim_ti2fp2 连接到从模式控制器。

图 23.29 PWM 输入模式时序

### 23.4.9 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位= 00）下，可直接由软件将每个输出比较信号（tim_ocxref 和 tim_ocx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号（tim_ocxref/tim_ocx）强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 101。tim_ocxref 进而强制设置为高电平（tim_ocxref 始终为高电平有效），同时 tim_ocx 获取 CCxP 极性位的相反值。

例如：CCxP=0（tim_ocx 高电平有效）=> tim_ocx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 tim_ocxref 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断请求和 DMA 请求。输出比较模式一节对此进行了介绍。

### 23.4.10 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平（OCXM=000），也可设置为有效电平（OCXM=001）、无效电平（OCXM=010）或进行翻转（OCxM=011）。
- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。
- 如果相应 DMA 使能位（TIMx_DIER 寄存器的 CCxDE 位，TIMx_CR2 寄存器的 CCDS 位，用来选择 DMA 请求）置 1，将发送 DMA 请求。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存

器。

在输出比较模式下，更新事件 UEV 对 tim_ocxref 和 tim_ocx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

**步骤：**

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，将 CCxIE 位和/或 CCxDE 位置 1。
4. 选择输出模式。例如，
   a) 写入 OCxM=0011 以在 CNT 匹配 CCRx 时翻转 tim_ocx 输出引脚
   b) 写入 OCxPE=0 以禁用预装载寄存器
   c) 写入 CCxP=0 以选择高电平有效
   d) 写入 CCxE=1 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可随时通过软件更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预装载寄存器（OCxPE=0，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。

下图列出了相关示例。

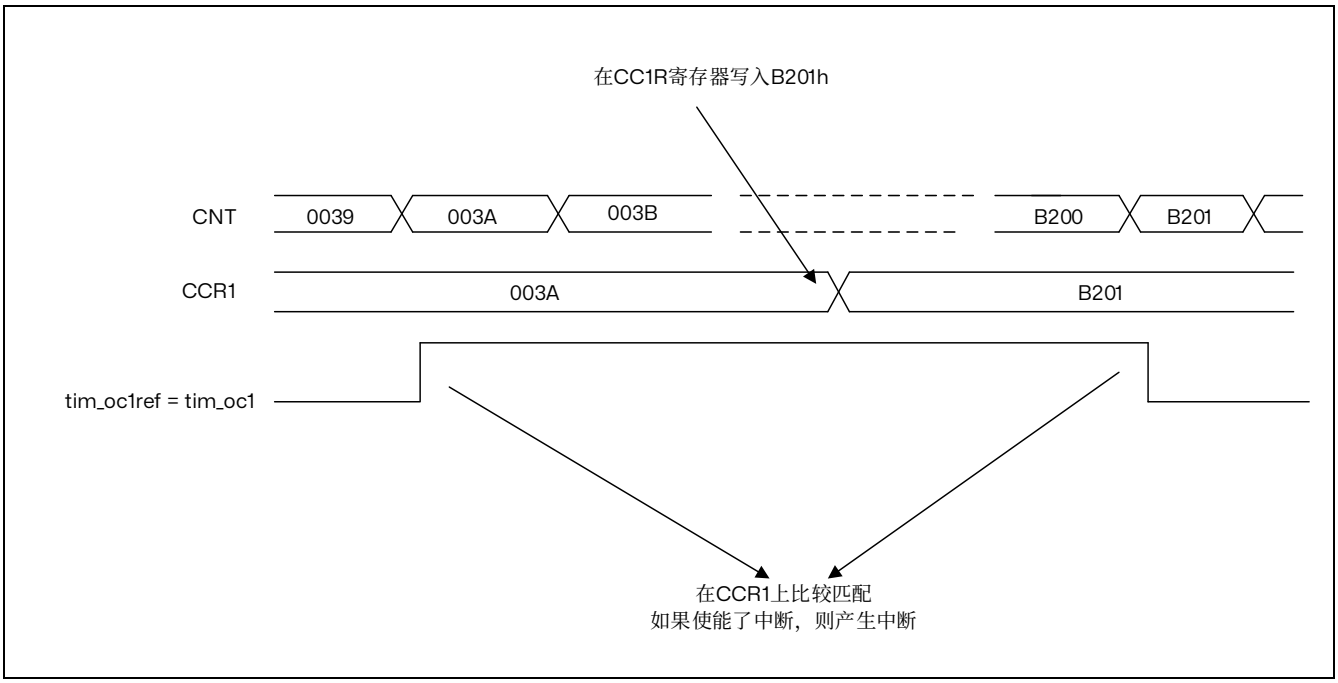



图 23.30 输出比较模式，翻转 tim_oc1

### 23.4.11 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110（PWM 模式 1）或 111（PWM 模式 2），可以独立选择各通道（每个 tim_ocx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

tim_ocx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。tim_ocx 输出通过将 TIMx_CCER 寄存器中的 CCxE 位置 1 来使能。有关详细信息，请参见 TIMx_CCERx 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 始终与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx ≤ TIMx_CNT 还是 TIMx_CNT ≤ TIMx_CCRx（取决于计数器计数方向）。外部事件可以通过 tim_etr_in 或 tim_oceref_clr 信号。在本例中仅声明了 tim_ocref_clr 信号

- 比较匹配事件之后
- 输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）从“冻结”配置（不进行比较，OCxM=“000”）切换为任一 PWM 模式（OCxM=“110”或“111”）。定时器运行期间，可以通过软件强制 PWM 输出。

根据 TIMx_CR1 寄存器 CMS 位状态，定时器能够产生边沿对齐模式或中心对齐模式的 PWM 信号。

**PWM 边沿对齐模式**

递增计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 tim_ocxref 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 tim_ocxref 保持为“1”。如果比较值为 0，则 tim_ocxref 保持为“0”。下图举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

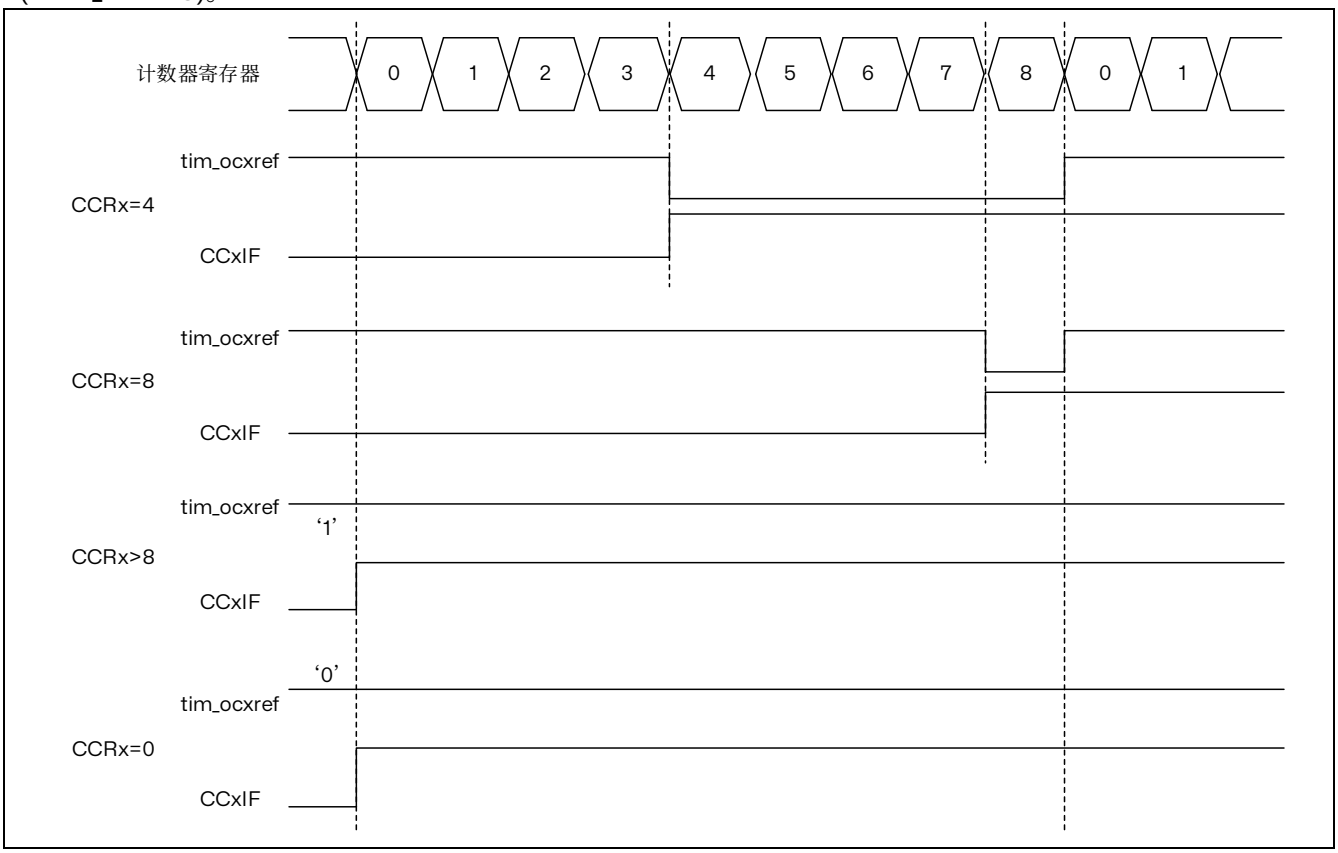



图 23.31 边沿对齐的 PWM 波形（ARR=8）

**递减计数配置**

当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。

在 PWM 模式 1 下，只要 TIMx_CNT>TIMx_CCRx，参考信号 tim_ocxref 便为低电平，否则为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，则 tim_ocxref 保持为“1”。此模式下不可能产生 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（其余所有配置对 tim_ocxref/tim_ocx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位（DIR）由硬件更新，不得通过软件更改。

下图显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8
- PWM 模式为 PWM 模式 1
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

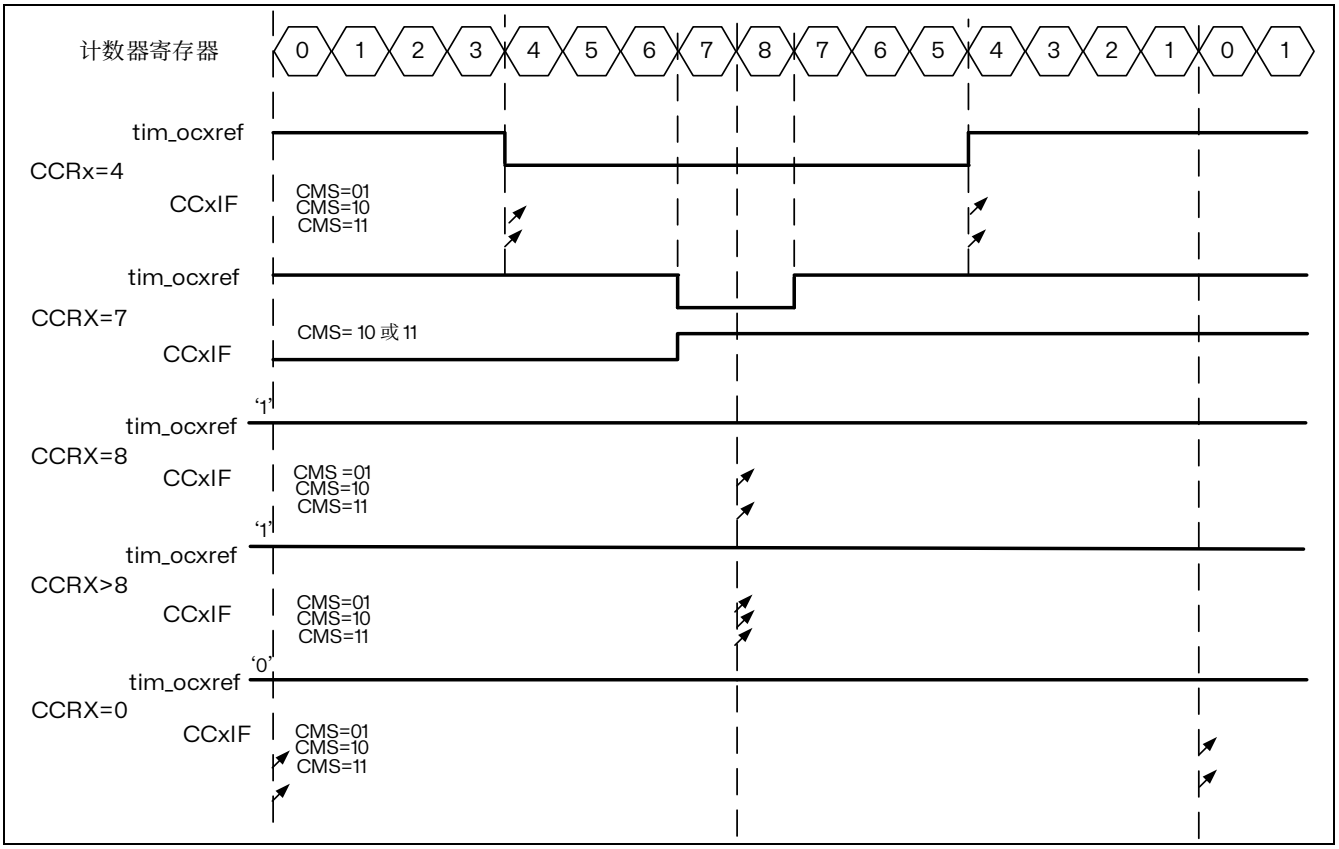



图 23.32 中央对齐的 PWM 波形（APR=8）

**中心对齐模式使用建议：**

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。
- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：

— 如果写入计数器中的值大于自动重载值（TIMx_CNT>TIMx_ARR），计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
— 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。

- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

**抖动模式**

通过使能抖动模式，使用 TIMx_CR1 寄存器中的 DITHEN 位，可以提高 PWM 模式的有效分辨率。这既适用于 CCR（用于增加占空比分辨率），也适用于 ARR（用于增加 PWM 频率分辨率）。

工作原理是在 16 个连续的 PWM 周期中，以预定义的模式使实际的 CCR（或 ARR）值略有变化（增加或减少一个定时器时钟周期）。这使得占空比或 PWM 期间的平均值提高了 16 倍分辨率。下图展示了应用于 4 个连续 PWM 周期的抖动原理。

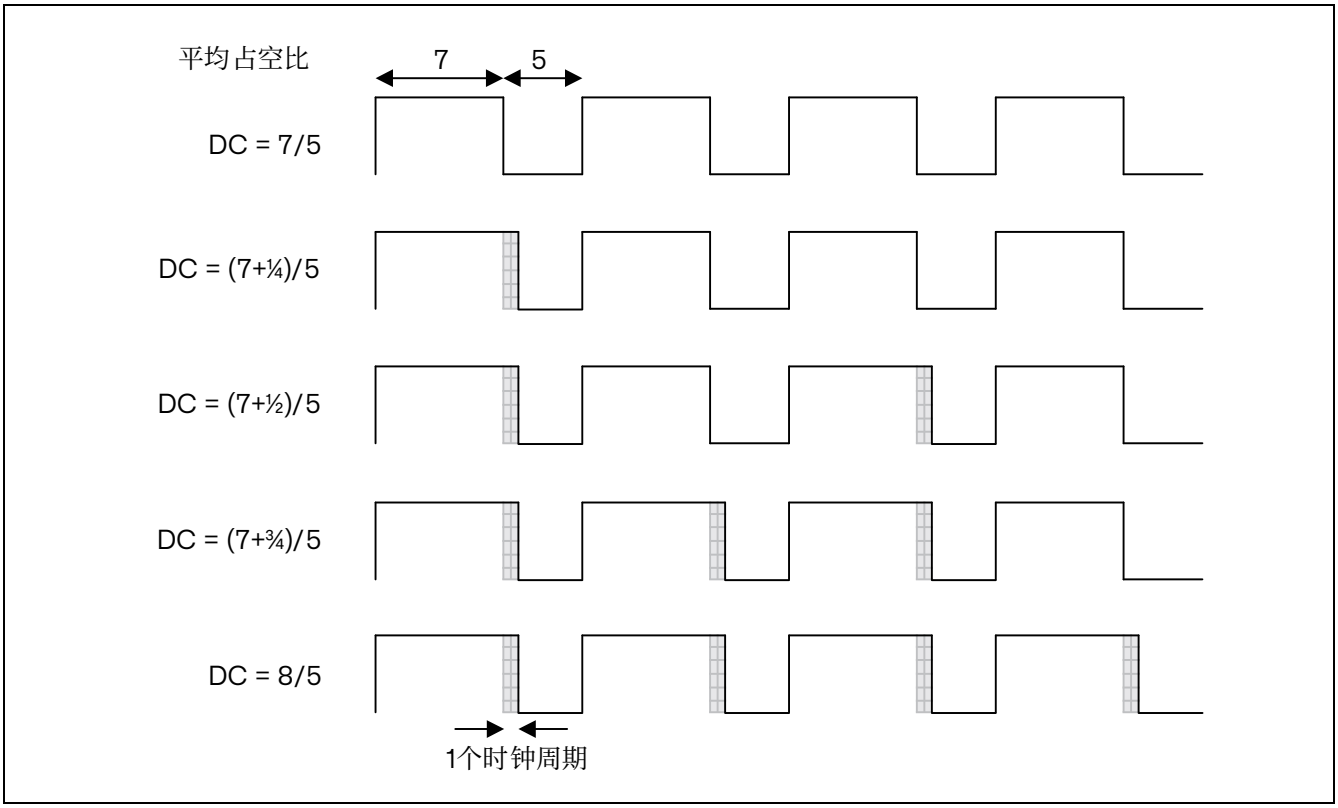



图 23.33 抖动原理

当抖动模式被使能时，寄存器编码将如下所示进行变化（参见下图示例）

- 4 个最低有效位（LSB）用于编码增强的分辨率部分（小数部分）
- 最高有效位（MSB）左移至 19:4 位，用于编码基本值

*注：如果设置清除* DITHEN *位，* ARR *和* CCR *值将自动更新。（例如，如果* ARR=0x05 *且* DITHEN=0，*它将更新为* ARR=0x50 *且* DITHEN=1*）。*

1. *必须重置* CEN *和* ARPE *位*
2. ARR[3:0]*位必须重置*
3. DITHEN *位必须重置*
4. CCIF *标志必须清除*
5. *可以设置* CEN *位（最终设置* ARPE=1*）。*

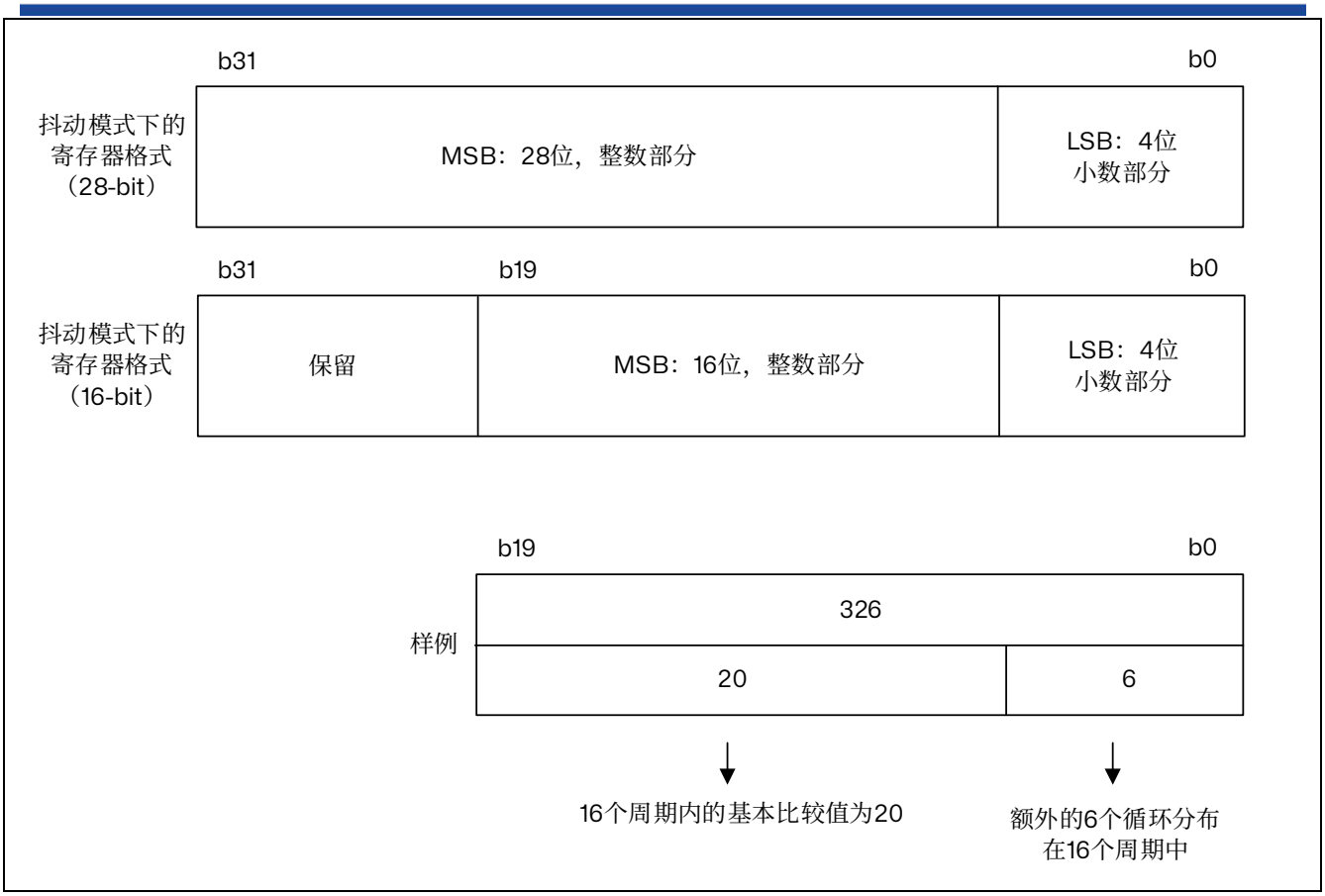



图 23.34 抖动模式下的数据格式和寄存器编码

最小频率由以下公式给出：

$$\text{分辨率} = \frac{\text{FTim}}{\text{Fpwm}} \Rightarrow \text{FpwmMin} = \frac{\text{FTim}}{\text{最大分辨率}}$$

$$\text{禁用抖动模式：FpwmMin} = \frac{\text{FTim}}{65536}$$

$$\text{使能抖动模式（16 位定时器）：FpwmMin} = \frac{\text{FTim}}{65535 + \frac{15}{16}}$$

$$\text{使能抖动模式（28 位定时器）：FpwmMin} = \frac{\text{FTim}}{268435454 + \frac{15}{16}}$$

*注意：对于16 位定时器，在抖动模式下，*TIMx_ARR *和*TIMx_CCRy *的最大值限制为*0xFFFEF *（对应于整数部分的*65534 *和抖动部分的*15*）。*

*对于28 位定时器，在抖动模式下，*TIMx_ARR *和*TIMx_CCRy *的最大值限制为*0xFFFFFFEF *（对应于整数部分的*264435454 *和抖动部分的*15*）。*

如下两张图所示，抖动模式可以增加 PWM 分辨率。

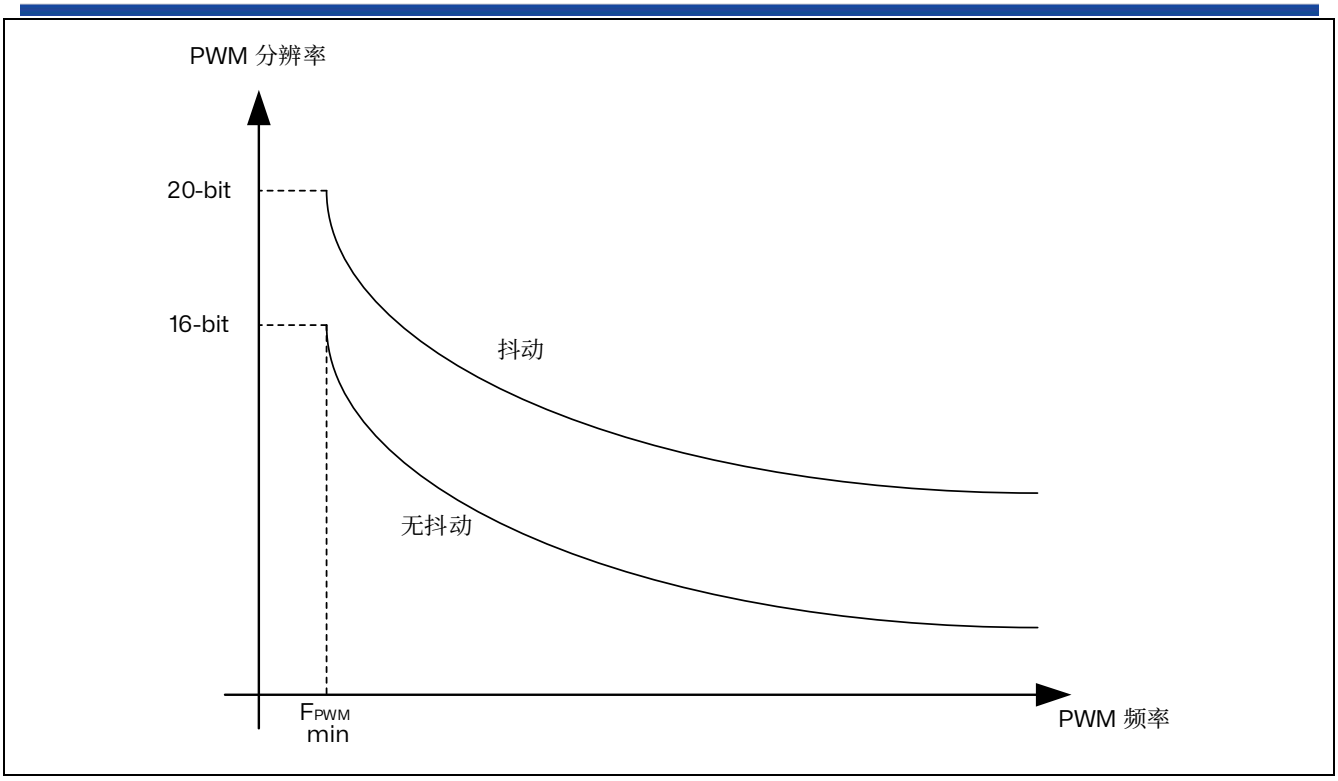



图 23.35 PWM 分辨率 vs 频率（16 位模式）

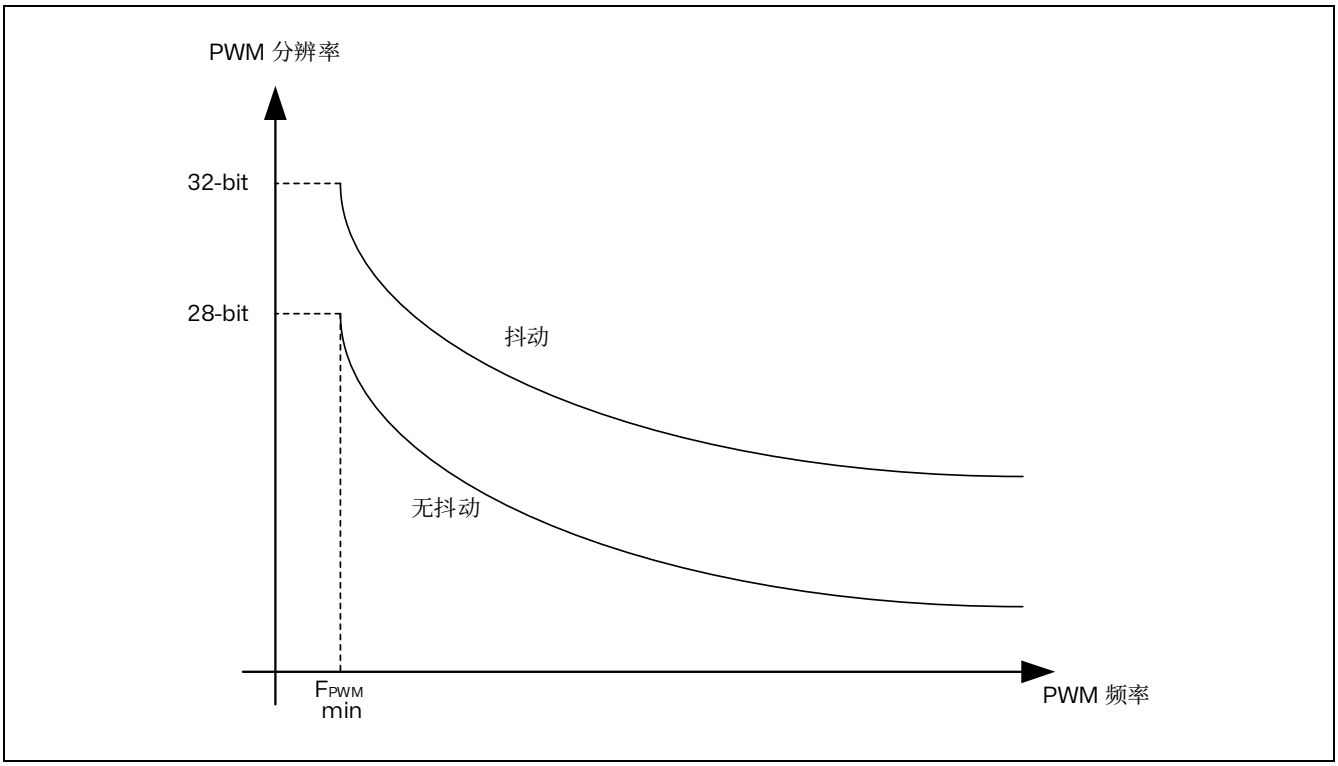



图 23.36 PWM 分辨率 vs 频率（28 位模式）

如下图所示，占空比和/或周期的变化分散在 16 个连续周期。

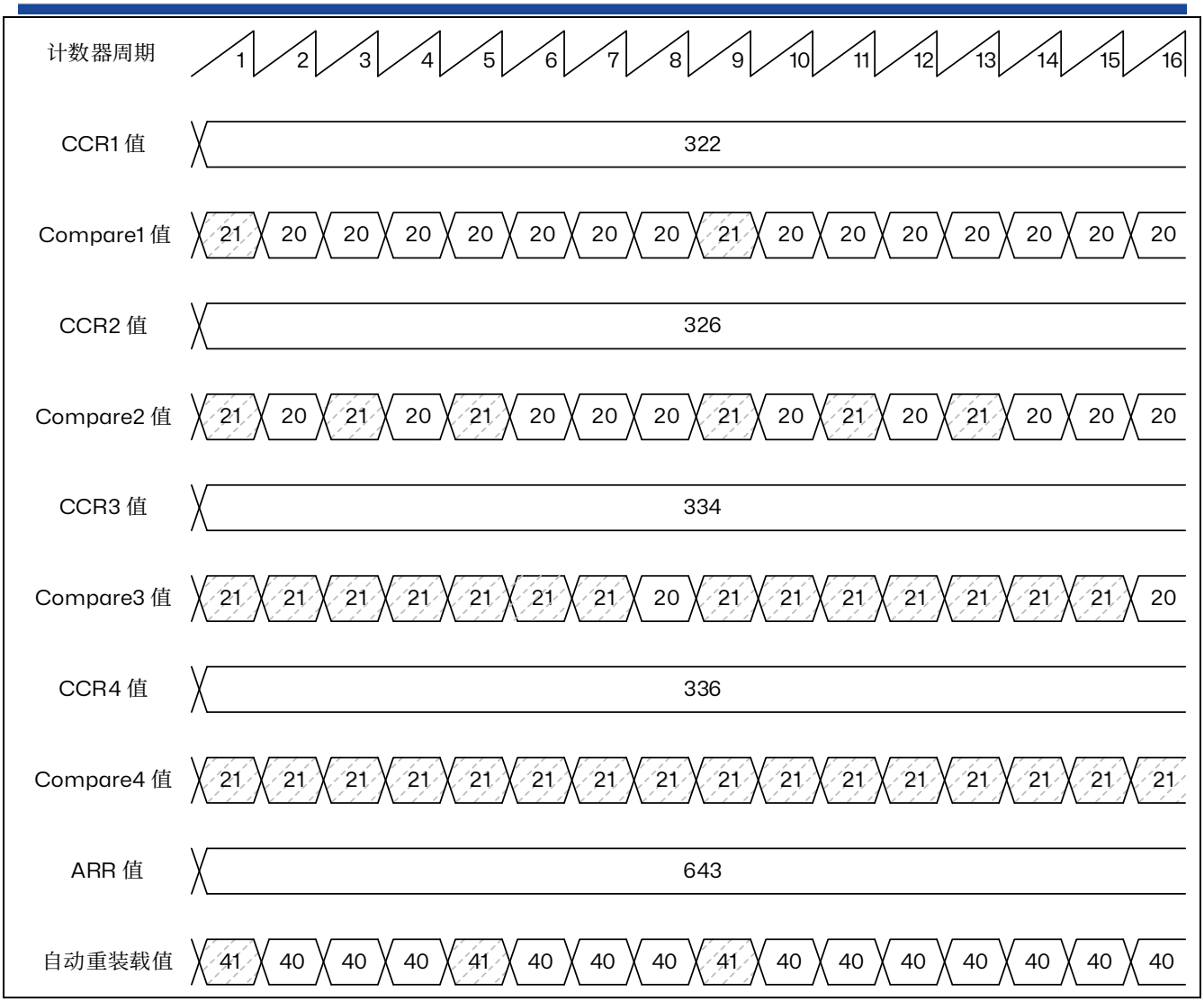



图 23.37 PWM 抖动模式

自动重新加载和比较值的增量按照下表中描述的特定模式分布。抖动序列是为了使增量分布尽可能均匀并最大限度地减少整体波动。

表 23.11 CCR 和 ARR 寄存器的变化抖动模式

| LSB 值 | 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 | - |

在中心对齐 PWM 模式下（TIMx_CR1 寄存器中的 CMS 位不为“00”），也提供了抖动模式。在这种模式下，抖动模式考虑到递增递减计数阶段，应用于 8 个连续的 PWM 周期，如下图所示。

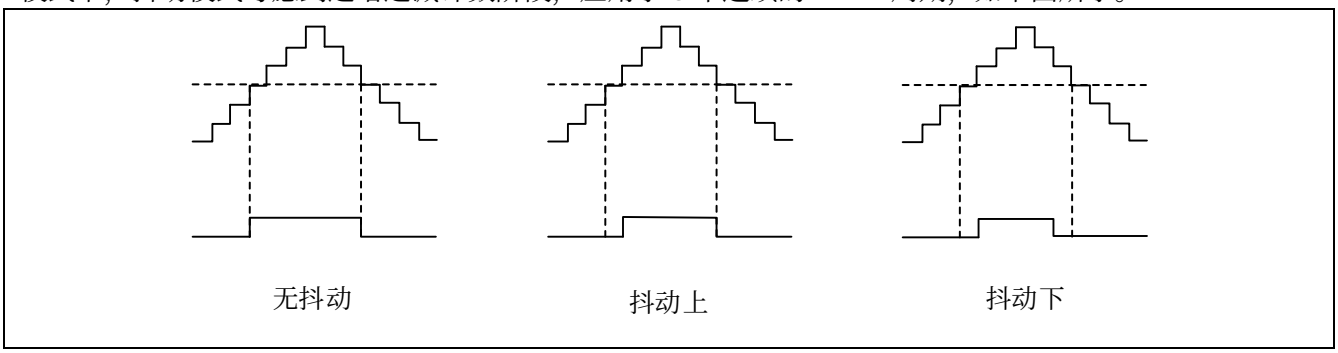



图 23.38 中心对齐 PWM 模式下抖动对占空比的影响

下表显示了如何在中心对齐 PWM 模式下添加抖动模式。

表 23.12 CCR 寄存器在中心对齐 PWM 模式下的变化抖动模式

| LSB 值 | PWM 周期 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | |
| | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 | 上 | 下 |
| 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 | - |

### 23.4.12 非对称 PWM 模式

非对称模式允许生成两个中心对齐的 PWM 信号，并具有可编程的相位偏移。虽然频率由 TIMx_ARR 寄存器的值决定，但占空比和相位偏移由一对 TIMx_CCRx 寄存器确定。一个寄存器控制 PWM 在计数上升期间，第二个寄存器在计数下降期间，这样 PWM 每半个 PWM 周期调整一次：

— tim_oc1refc（或 tim_oc2refc）由 TIMx_CCR1 和 TIMx_CCR2 控制
— tim_oc3refc（或 tim_oc4refc）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过将“1110”（非对称 PWM 模式 1）或“1111”（非对称 PWM 模式 2）写入 TIMx_CCMRx 寄存器的 OCxM 位，可以在两个通道上独立选择非对称 PWM 模式（每个 CCR 寄存器对一个 tim_ocx 输出）。

*注意：为了兼容性原因，*OCxM[3:0]*位域被分为两个部分，最高有效位与最低*3 *位不连续。*

当某个通道用作非对称 PWM 通道时，其互补通道也可以使用。例如，如果 tim_oc1refc 信号在通道 1 上生产（非对称 PWM 模式 1），可以在通道 2 上输出 tim_oc1refc 信号，或者产生非对称 PWM 模式 1 的结果 tim_oc2refc 信号。

下图标识使用非对称 PWM 模式生成信号的示例（通道 1 到 4 配置为非对称 PWM 模式 2）。

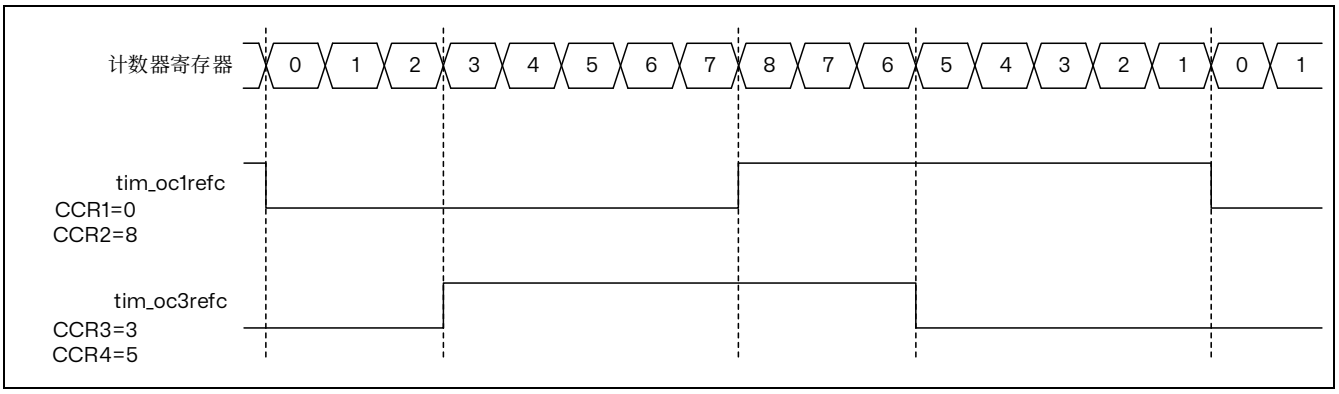



图 23.39 生成两个具有 50%占空比的相位偏移 PWM 信号

### 23.4.13 组合 PWM 模式

组合 PWM 模式允许生成两个边缘对齐或中心对齐的 PWM 信号，各自脉冲之间的可编程延迟和相位偏移。虽然频率由 TIMx_ARR 寄存器的值决定，但占空比和延迟由两个 TIMx_CCRx 寄存器确定。结果信号 tim_ocxrefc 由两个参考 PWM 的 OR 或 AND 逻辑组合而成：

— tim_oc1refc（或 tim_oc2refc）由 TIMx_CCR1 和 TIMx_CCR2 控制
— tim_oc3refc（或 tim_oc4refc）由 TIMx_CCR3 和 TIMx_CCR4 控制

通过在 TIMx_CCMRx 寄存器的 OCxM 位中写入“1100”（组合 PWM 模式 1）或“1101”（组合 PWM 模式 2），可以在两个通道上独立选组合 PWM 模式（每个 CCR 寄存器对一个 tim_ocx 输出）。

当某个通道用作组合 PWM 通道时，其互补通道必须配置为相反的 PWM 模式（例如，一个通道为组合 PWM 模式 1，另一个通道为组合 PWM 模式 2）。

*注意：为了兼容性原因，*OCxM[3:0]*位域被分为两个部分，最高有效位与最低*3 *位不连续。*

下图表示使用组合 PWM 模式生成信号的示例，该示例的配置如下：

— 通道 1 配置为组合 PWM 模式 2。
— 通道 2 配置为 PWM 模式 1。
— 通道 3 配置为组合 PWM 模式 2。

— 通道 4 配置为 PWM 模式 1。

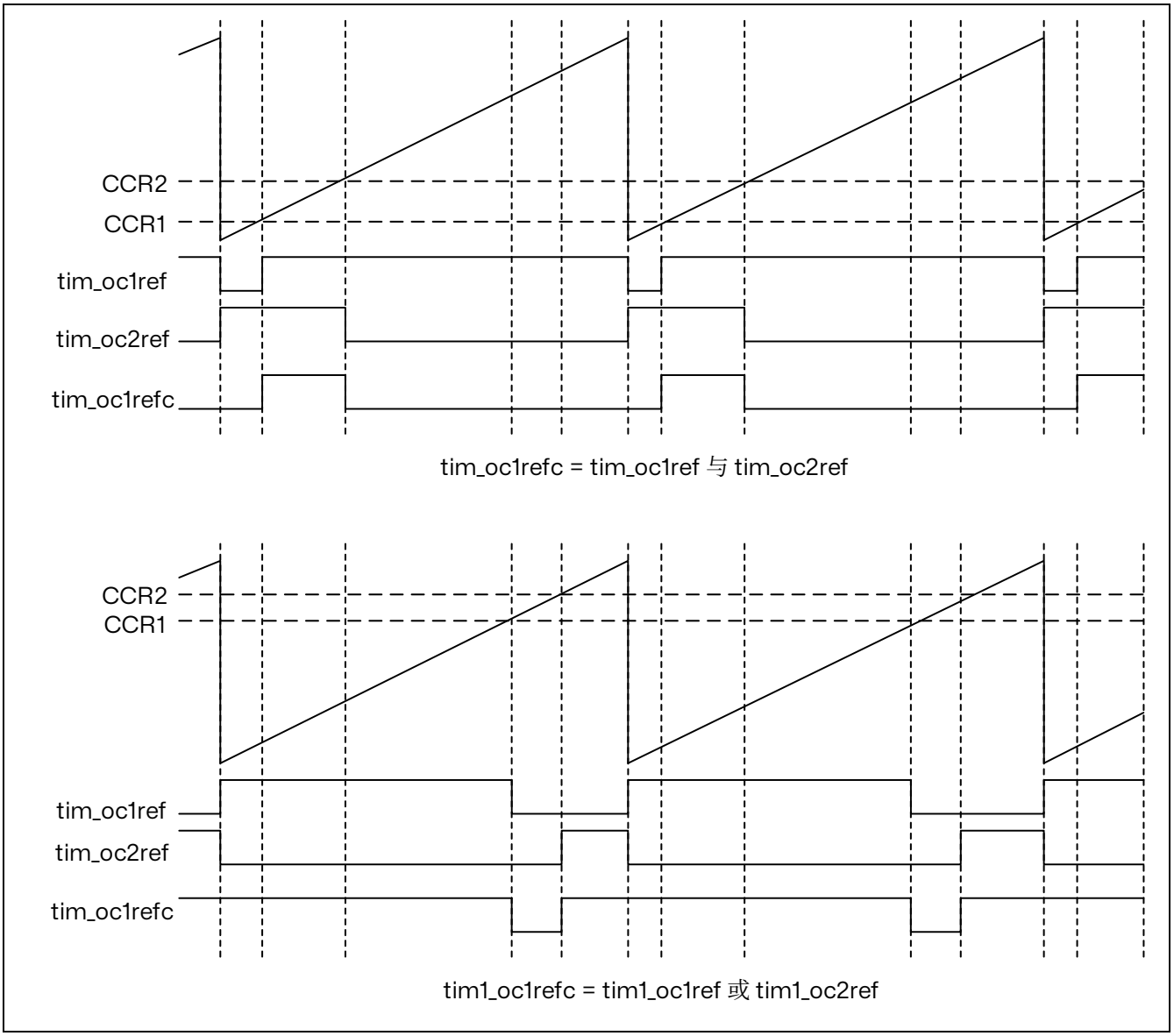



图 23.40 通道 1 和通道 3 的组合 PWM 模式

### 23.4.14 在外部事件时清除 tim_ocxref 信号

对于给定通道，在 tim_ocref_clr_int 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 tim_ocxref 信号变为低电平。tim_ocxref 信号将保持低电平，直到发生下一更新事件(UEV)。此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。

tim_ocref_clr_int 源取决于 OCREF 清除选择功能实施，请参考前序章节：TIM2/TIM3/TIM4/TIM5 实现。

如果实施了 OCREF 清除选择功能，则可以选择 tim_ocref_clr_int 在 tim_ocref_clr 输入和 tim_etrf 输入（滤波器后的 tim_etr_in）之间，通过以下方式进行在 TIMx_SMCR 寄存器中配置 OCCS 位。tim_ocref_clr 输入可以是使用中 OCRSEL[2:0]位域，从多个 tim_ocref_clr[7:0]输入中选出 TIMx_AF2 寄存器，如下图所示。

当配置组合 PWM 模式，对于给定通道，在 tim_ocref_clr_int 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使组合结果 tim_ocxrefc 信号变为低电平。tim_ocxrefc 信号将保持低电平，直到发生下一更新事件（UEV）。

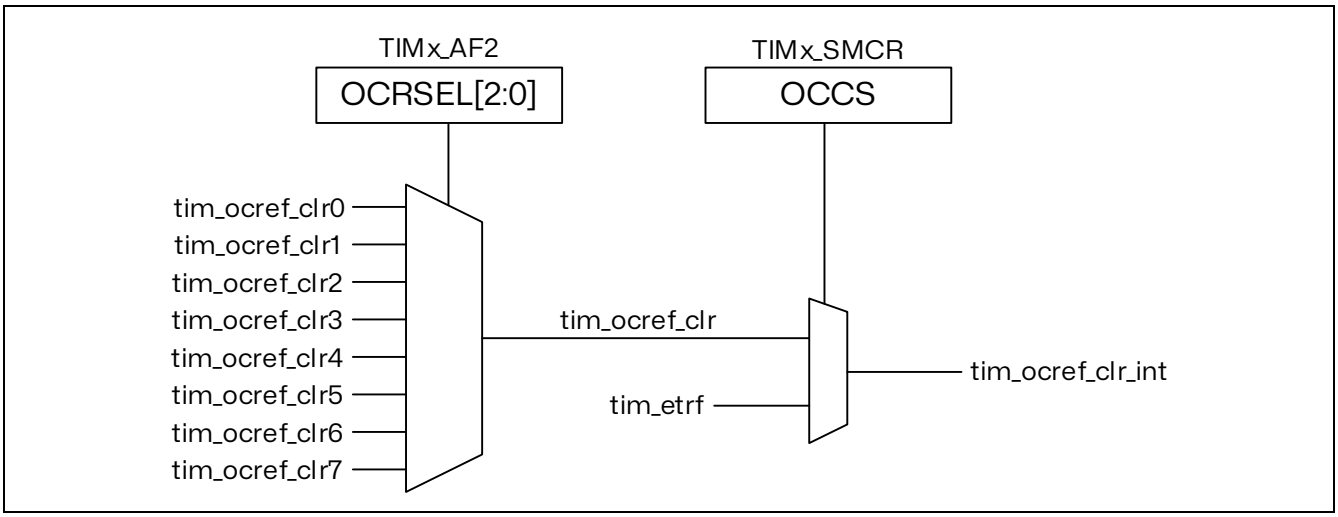



图 23.41 tim_ocref_clr 输入选择复用器

如果未实现 OCREF 清除选择功能，则 tim_ocref_clr_int 输入为直接连接到 tim_etrf 输入。

例如，tim_ocref_clr_int 信号可以连接到比较器的输出，用于控制电流。此时，tim_etr_in 必须如下配置：

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0]位清除为 00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位清除为 0。
3. 外部触发极性（ETP）和外部触发滤波器（ETF）可根据应用需要进行配置。

下图对比了使能位 OCxCE 在不同值下的情况，显示了当 tim_etrf 输入变为高电平时 tim_ocxref 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

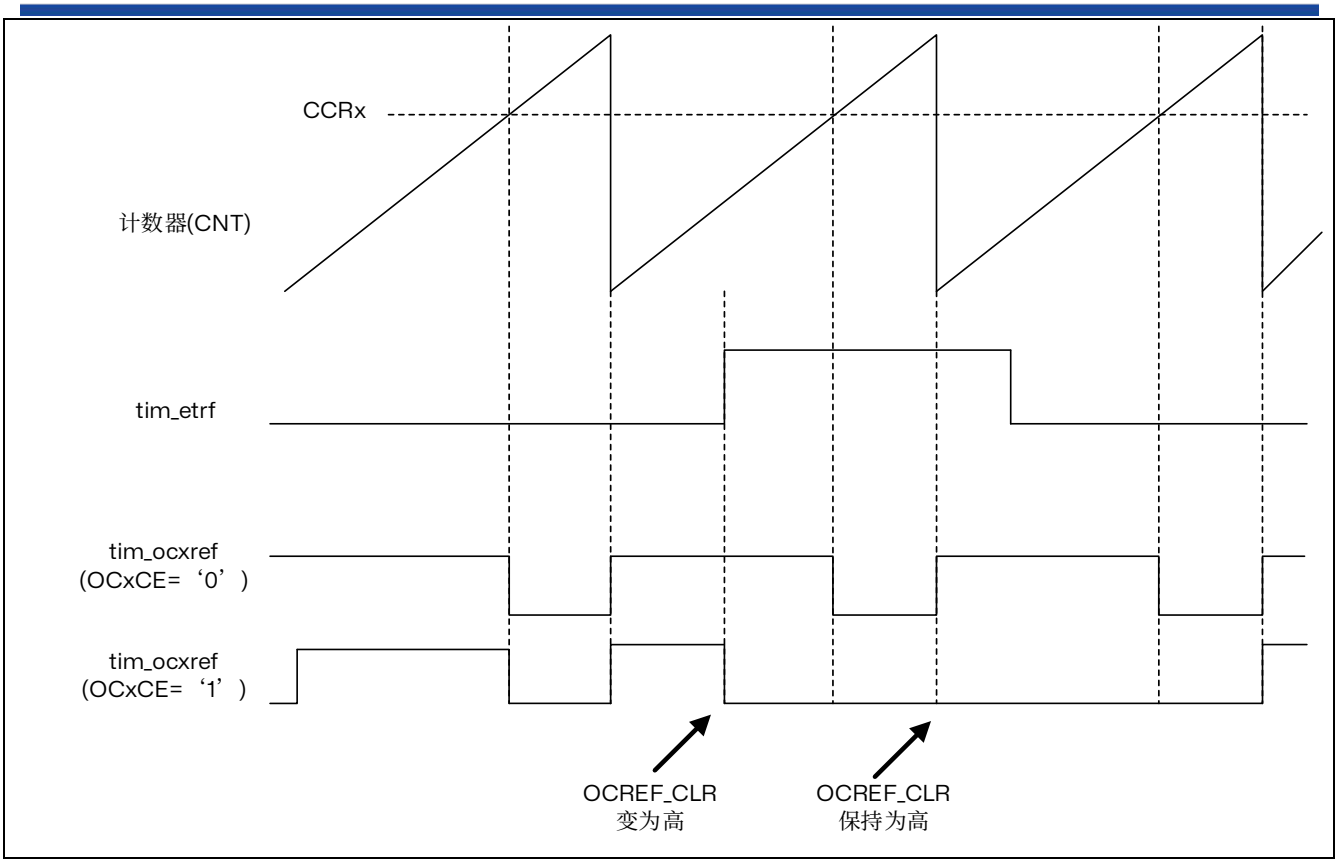



图 23.42 清除 TIMx 的 tim_ocxref

*注：如果* PWM *的占空比为* 100% (CCRx>ARR)，*则下次计数器上溢时会再次使能* tim_ocxref。

### 23.4.15 单脉冲模式

单脉冲模式（OPM）是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- CNT < CCRx ≤ ARR（特别注意，0<CCRx）

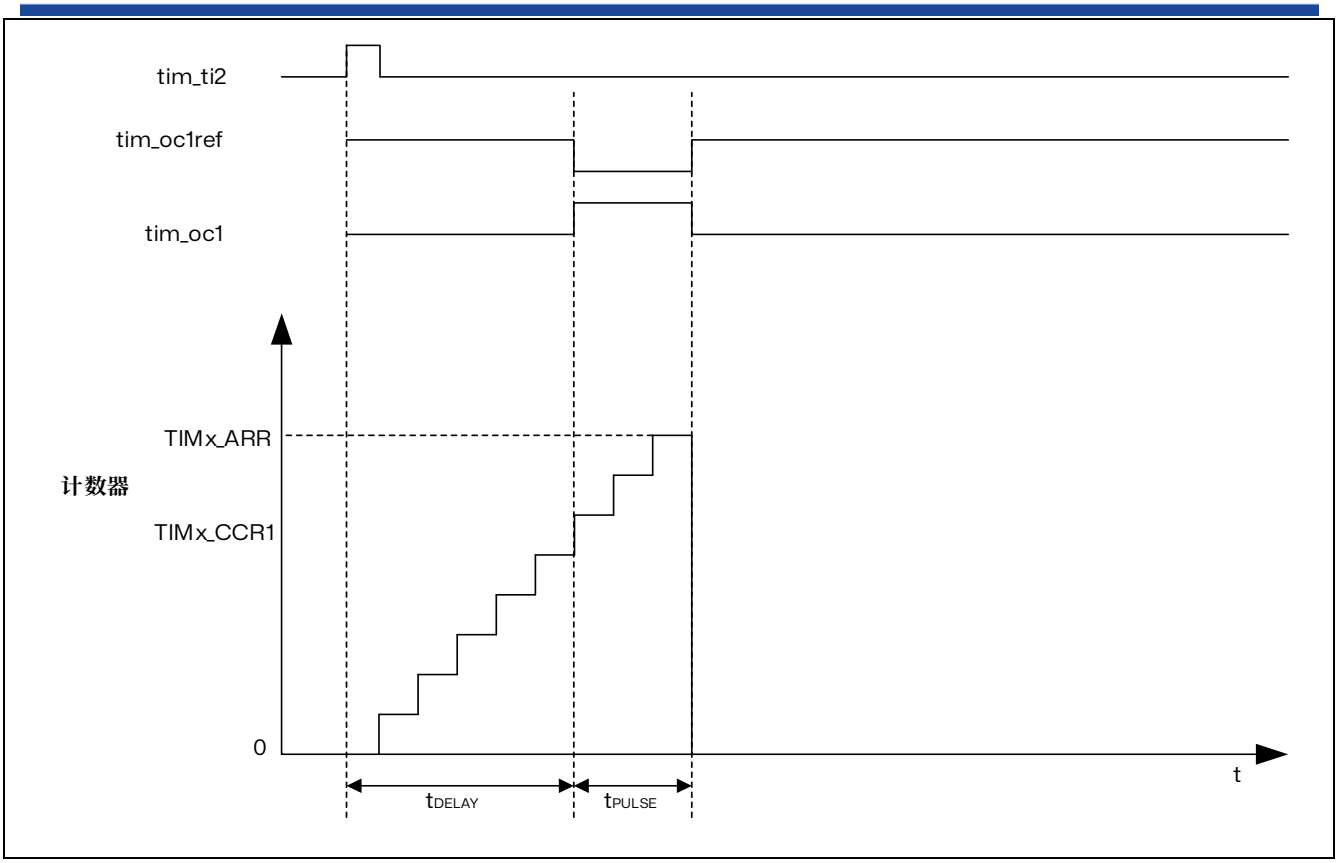



图 23.43 单脉冲模式的例子

例如，用户希望达到这样的效果：在 tim_ti2 输入引脚检测到正沿时，经过 $t_{DELAY}$ 的延迟，在 tim_oc1 上产生一个长度为 $t_{PULSE}$ 的正脉冲。

使用 tim_ti2fp2 作为触发 1:

1. 通过在 TIMx_TISEL 寄存器中使用 TI2SEL[3:0]位选择合适的 tim_ti2_in[15:0]源（内部或外部）。
2. 在 TIMx_CCMR1 寄存器中写入 CC2S=01，将 tim_ti2fp2 映射到 tim_ti2。
3. 在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=“0”，使 tim_ti2fp2 能够检测上升沿。
4. 在 TIMx_SMCR 寄存器中写入 TS=00110，将 tim_ti2fp2 配置为从模式控制器的触发 (tim_trgi)。
5. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），使用 tim_ti2fp2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- $t_{DELAY}$ 由写入 TIMx_CCR1 寄存器的值定义。
- $t_{PULSE}$ 由自动重载值与比较值（TIMx_ARR - TIMx_CCR1）之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 tim_ti2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位清

“0”时，即选择重复模式。

**特例：tim_ocx 快速使能：**

在单脉冲模式下，tim_tix 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（$t_{DELAY}$ 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 tim_ocxref（和 tim_ocx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用

### 23.4.16 可重触发单脉冲模式

此模式允许计数器在接收到刺激后开始计数，并生成一个具有可编程长度的脉冲，但与上一节中描述的非重触发单脉冲模式存在以下差异：

- 只要触发器发生，脉冲就会立即开始（没有可编程的延迟）
- 如果在先前触发的脉冲完成之前发生新的触发，则脉冲会被延长

定时器必须处于从模式，TIMx_SMCR 寄存器的 SMS[3:0]位设置为“1000”（组合重置+触发模式），OCxM[3:0]位设置为“1000”或“1001”，用于可重触发 OPM 模式 1 或 2。

如果定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）。如果定时器配置为递减计数模式，CCRx 必须大于或等于 ARR。

*注：在可重复触发单脉冲模式中，*CCxIF *标志不重要。*

OCxM[ [3:0]*和*SMS[3:0]*位域由于兼容性的原因被分成两个部分，最高有效位与最低*3 *位不是连续的。*

OCxM *这种模式不能与中央对齐*PWM *模式一起使用。*TIMx_CR1 *寄存器中的*CMS[1:0]*必须设置为*00。

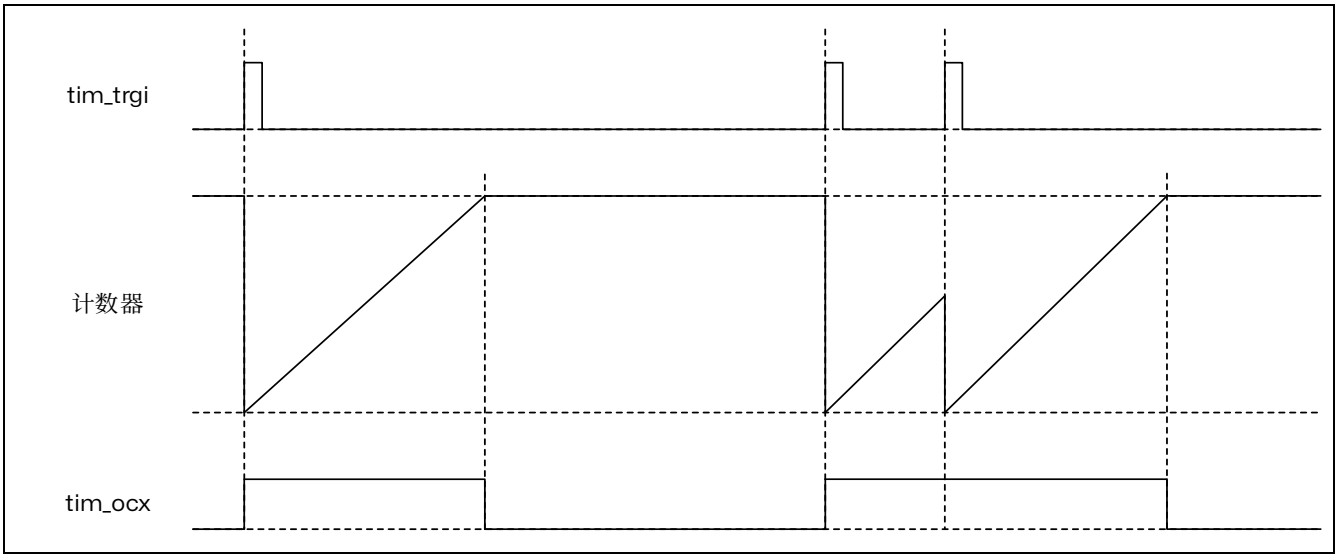



图 23.44 可重触发单脉冲模式

### 23.4.17 在比较模式下的脉冲

在比较匹配事件发生时，可以生成一个脉冲。当计数器的值等于给定的比较值时，可以生成一个具有

可编程脉冲宽度的信号，用于调试或同步。

此模式适用于任何从机模式选择，包括编码器模式、边沿对齐和中心对齐计数模式。它仅适用于通道 3 和通道 4。脉冲发生器是唯一的，由两个通道共享，如下图所示。

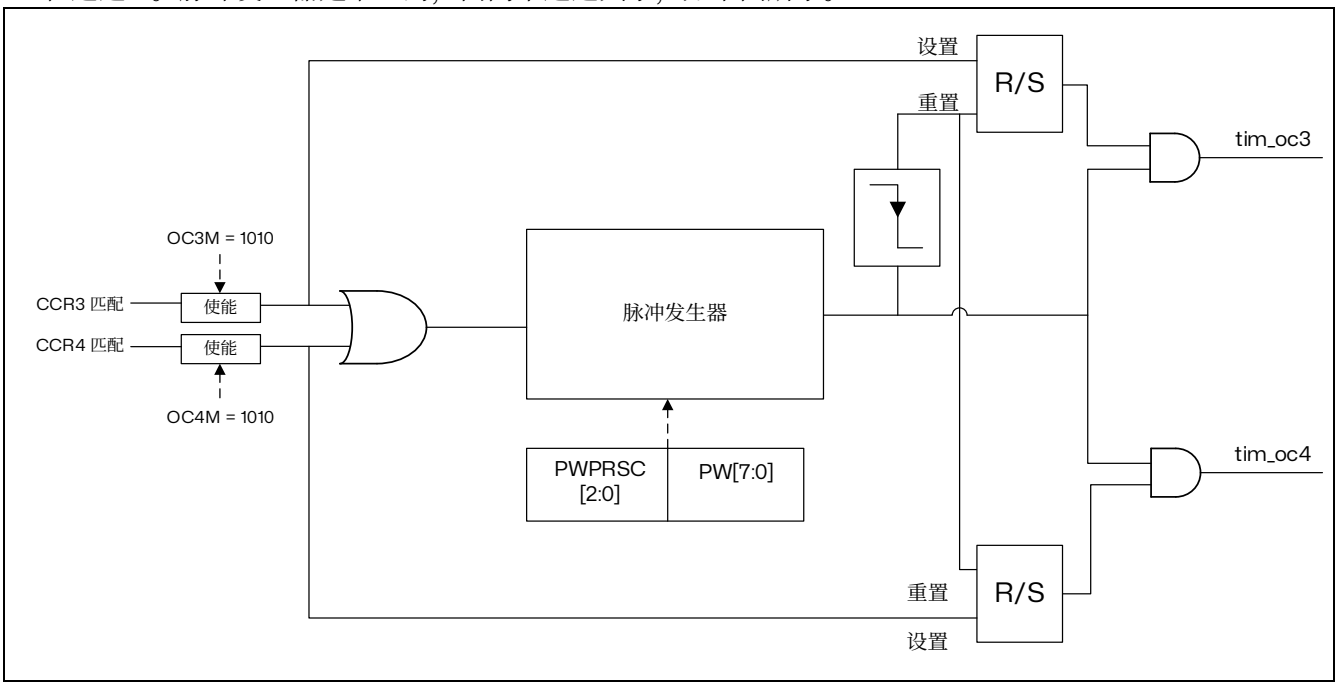



图 23.45 脉冲发生器电路

下图展示了如何在边沿对齐模式和编码器工作模式下生成脉冲。

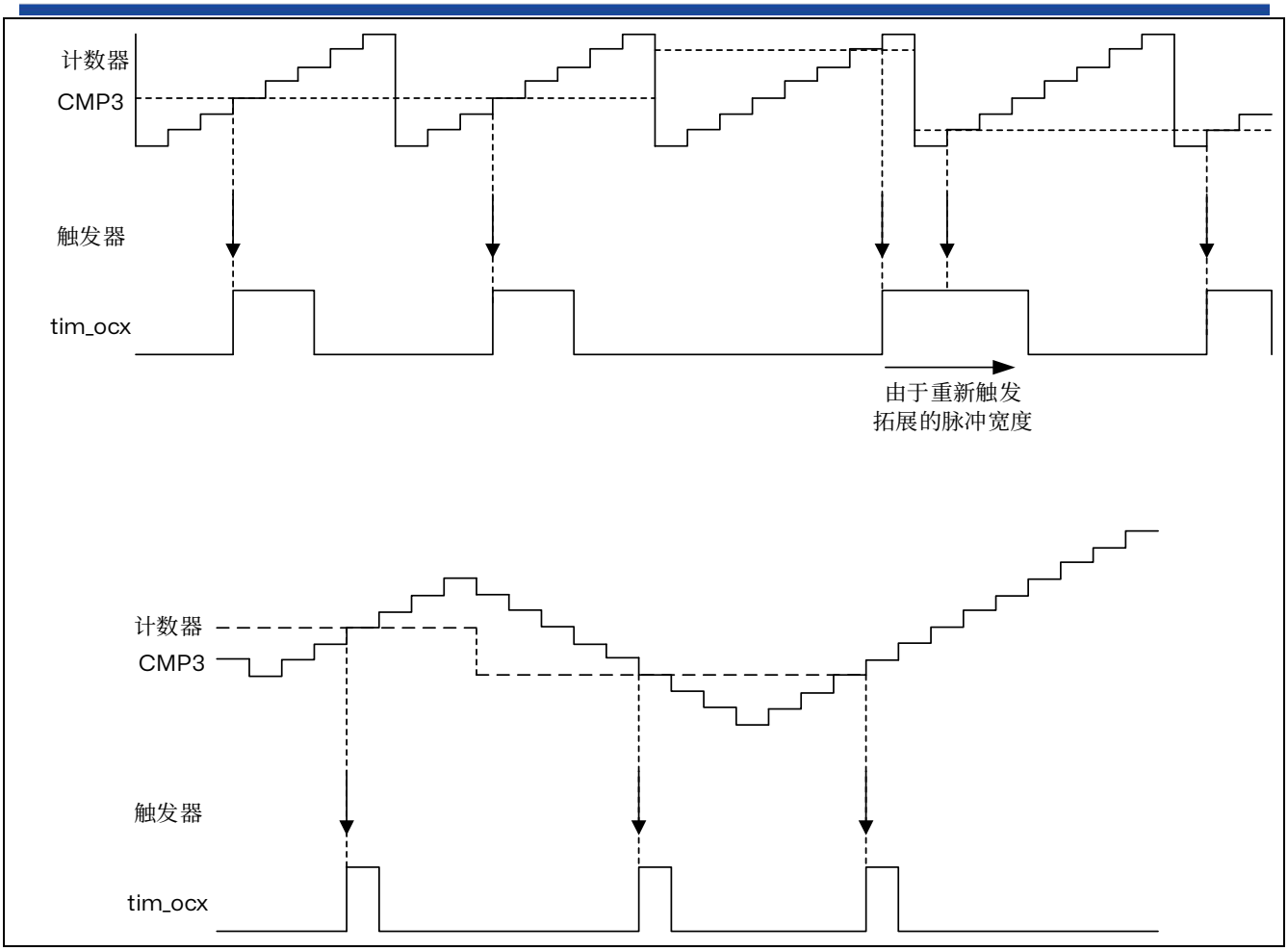



图 23.46 在比较事件上的脉冲生成，适用于边沿对齐模式和编码器模式

此输出比较模式是通过 TIMx_CCMR2 寄存器中的 OC3M[3:0]和 OC4M[3:0]位域进行选择的。

脉冲宽度通过寄存器中的 PW[7:0]位域进行编程，使用特定的时钟，并根据 PWPRSC[2:0]位的预设进行预分频，具体如下：

$t_{PW} = PW[7:0] \times t_{PWG}$

其中 $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$。

给出了分辨率和最大值，具体取决于预分频器的值。

脉冲可以重新触发：在脉冲正在进行时触发新的脉冲，会导致脉冲被延长。

*注意：如果同时使能两个通道，只要一个通道上的触发器不与并发输出上生成的脉冲重叠，脉冲就会独立发出。相反，如果两个触发器重叠，与第一个到达的触发器相关的脉冲宽度会被延长（由于重新触发），而最后一个到达的触发器的脉冲宽度是正确的（如下图所示）。*

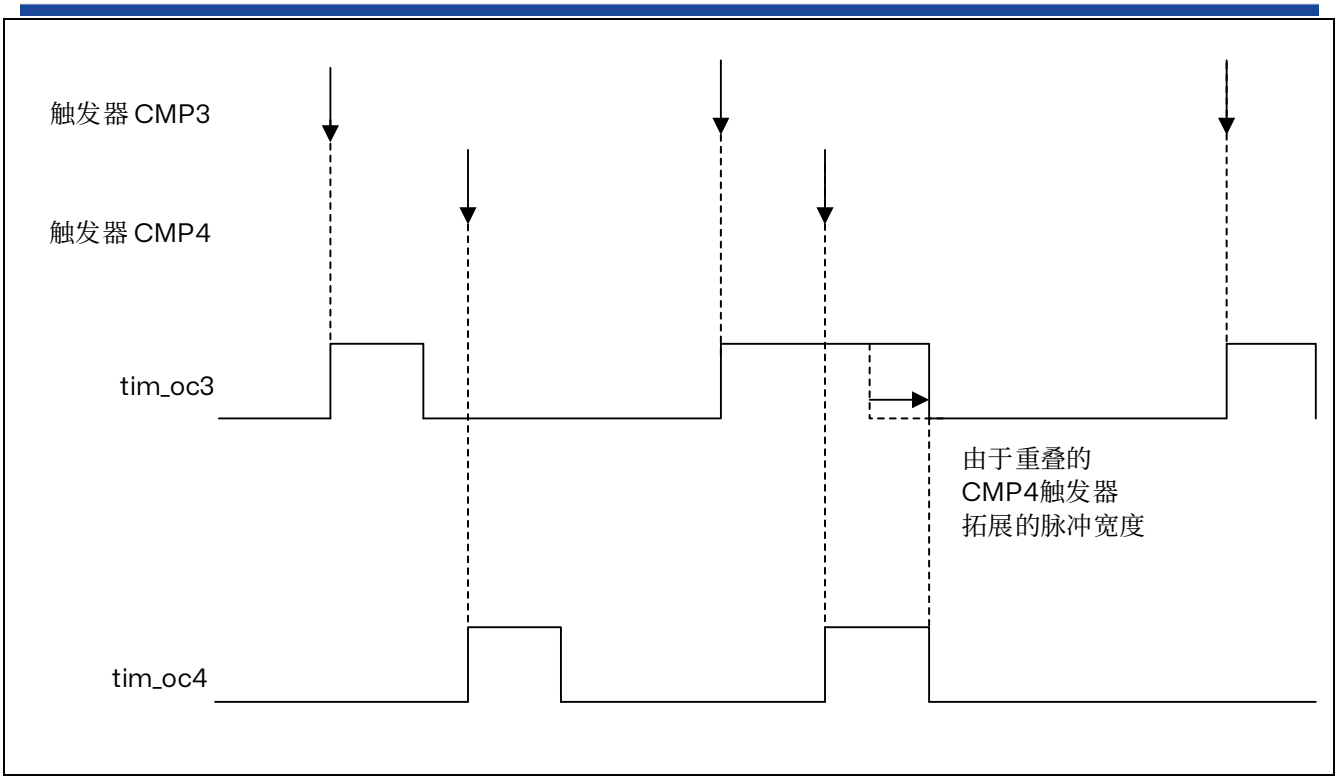



图 23.47 在同时触发的情况下延长脉冲宽度

### 23.4.18 编码器接口模式

#### 正交编码器

选择编码器接口模式时，如果计数器仅在 tim_ti2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=0001；如果计数器仅在 tim_ti1 边沿处计数，写入 SMS=0010；如果计数器在 tim_ti1 和 tim_ti2 边沿处均计数，则写入 SMS=0011。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 tim_ti1 和 tim_ti2 极性。如果需要，还可对输入滤波器进行编程。

tim_ti1 和 tim_ti2 两个输入用于连接增量编码器。请参见下表。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 tim_ti1fp1 或 tim_ti2fp2 上的每次有效信号转换提供。tim_ti1fp1 和 tim_ti2fp2 是进行输入滤波器和极性选择后 tim_ti1 和 tim_ti2 的信号，如果不进行滤波和反相，则 tim_ti1fp1=tim_ti1，tim_ti2fp2=tim_ti2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（tim_ti1 或 tim_ti2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 tim_ti1 或 tim_ti2 边沿处计数，还是同时在 tim_ti1 和 tim_ti2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、触发输出功能继续正常工作。

在此模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 tim_ti1 和 tim_ti2 不同时切换）。

表 23.13 计数方向与编码器信号的关系（CC1P = CC2P = 0）

| 有效边沿 | SMS[3:0] | 相对信号的电平（tim_ti1fp1 对应 tim_ti2, tim_ti2fp2 对应 tim_ti1） | tim_ti1fp1 信号 | | tim_ti2fp2 信号 | |
|---|---|---|---|---|---|---|
| | | | 上升 | 下降 | 上升 | 下降 |
| 仅 x1 模式在 tim_ti1 计数 | 1110 | 高 | 递减 | 递增 | 不计数 | 不计数 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| 仅 x1 模式在 tim_ti2 计数 | 1111 | 高 | 不计数 | 不计数 | 递增 | 递减 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| 仅 x2 模式在 tim_ti1 计数 | 0001 | 高 | 递减 | 递增 | 不计数 | 不计数 |
| | | 低 | 递增 | 递减 | 不计数 | 不计数 |
| 仅 x2 模式在 tim_ti2 计数 | 0010 | 高 | 不计数 | 不计数 | 递增 | 递减 |
| | | 低 | 不计数 | 不计数 | 递减 | 递增 |
| x4 模式在 tim_ti1 和 tim_ti2 计数 | 0011 | 高 | 递减 | 递增 | 递增 | 递减 |
| | | 低 | 递增 | 递减 | 递减 | 递增 |

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

下图以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S=“01”（TIMx_CCMR1 寄存器，tim_ti1fp1 映射到 tim_ti1 上）。
- CC2S=“01”（TIMx_CCMR2 寄存器，tim_ti2fp2 映射到 tim_ti2 上）。
- CC1P=“0”，CC1NP=“0”（TIMx_CCER 寄存器，tim_ti1fp1 未反相，tim_ti1fp1= tim_ti1）。
- CC2P=“0”，CC2NP=“0” TIMx_CCER 寄存器，tim_ti2fp2 未反相，tim_ti2fp2= tim_ti2）。
- SMS=“011”（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿都处于有效状态）。
- CEN=“1”（TIMx_CR1 寄存器，使能计数器）。

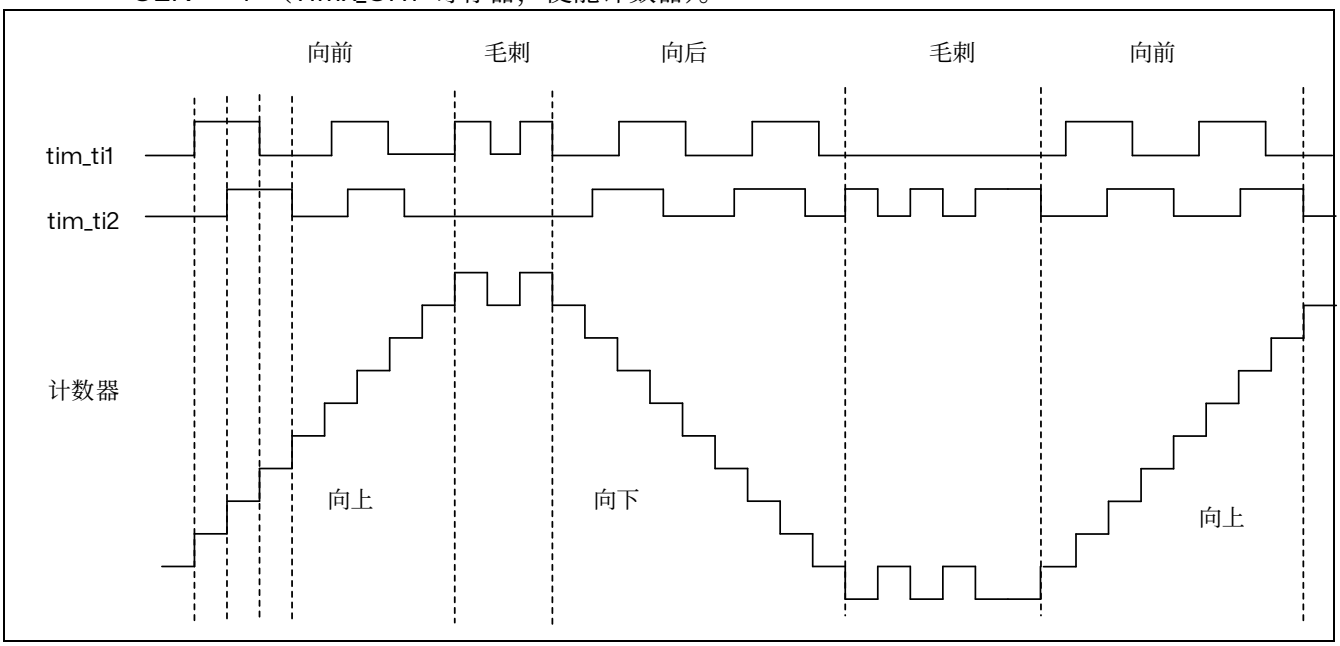



图 23.48 编码器接口模式下的计数器工作示例

下图举例说明 tim_ti1fp1 极性反相时计数器的行为（除 CC1P=“1”外，其它配置与上例相同）。

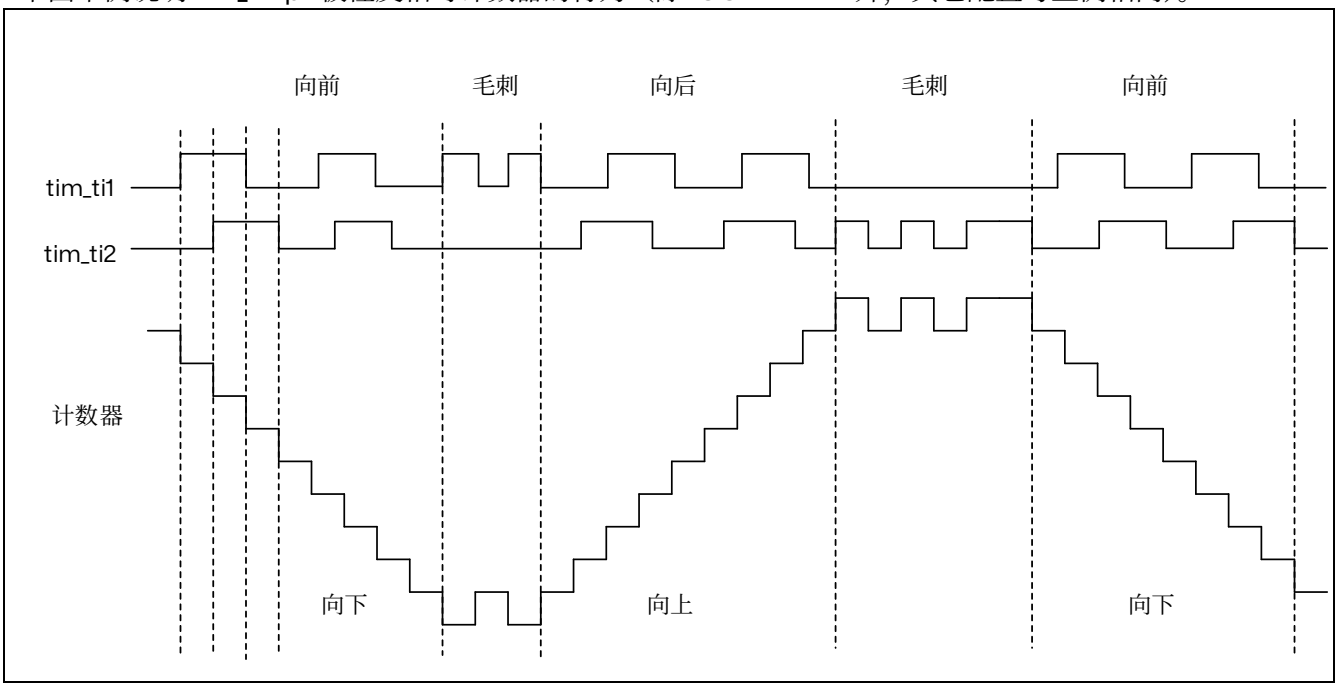



图 23.49 tim_ti1fp1 极性反相时的编码器接口模式示例

下图显示了各种计数模式下速度反转期间计时器计数器的值。

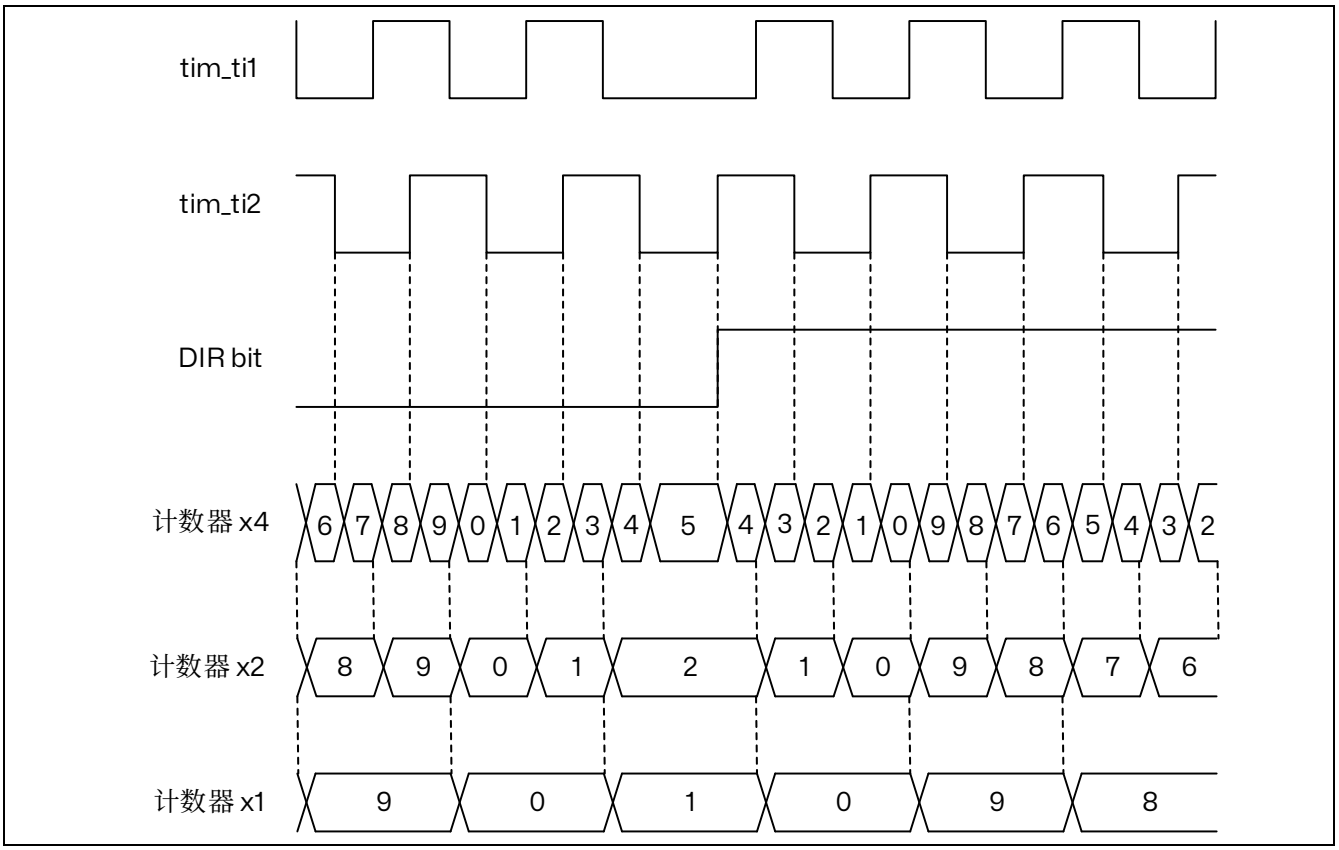



图 23.50 正交编码器的计数模式

当定时器配置为编码器接口模式时，可以提供传感器当前位置的信息。通过测量两个编码器事件之间

的周期，使用配置为捕获模式的第二个定时器，可以获得动态信息（速度、加速度、减速度）。指示机械零的编码器输出可用于此目的。根据两个事件之间的时间，还可以定期读取计数器的值。如果可用，可以通过将计数器的值锁存到第三个输入捕获寄存器来实现（此时捕获信号必须是周期性的，并且可以由另一个定时器生成）。

TIMx_CR1 寄存器中的 IUFREMAP 位将更新中断标志（UIF）的连续副本强制复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许原子地读取计数器的值和由 UIFCPY 标志触发的潜在的翻转条件。这可以通过避免在后台任务（计数器读取）和中断（更新中断）之间共享处理来简化角速度的计算。

UIF 和 UIFCPY 标志之间没有延迟。

### 时钟加方向编码器模式

除了正交编码器模式外，定时器还支持其他类型的编码器。

在下图中显示的“时钟加方向”模式下，时钟在 tim_ti2 上提供单线，而方向使用 tim_ti1 输入强制。

此模式通过 TIMx_SMCR 寄存器中的 SMS[3:0]位域使能，如下所示：

- 1010：x2 模式，计数器在时钟的上升沿和下降沿都更新
- 1011：x1 模式，计数器在单个时钟边沿更新，根据 CC2P 位的值而定：CC2P=0 对应上升沿敏感，CC2P=1 对应下降沿敏感。

tim_ti1 上的方向信号极性通过 CC1P 位设置：0 对应正极性（当 tim_ti1 为高电平时向上计数，当 tim_ti1 为低电平时向下计数），CC1P=1 对应负极性（当 tim_ti1 为低电平时向上计数）。

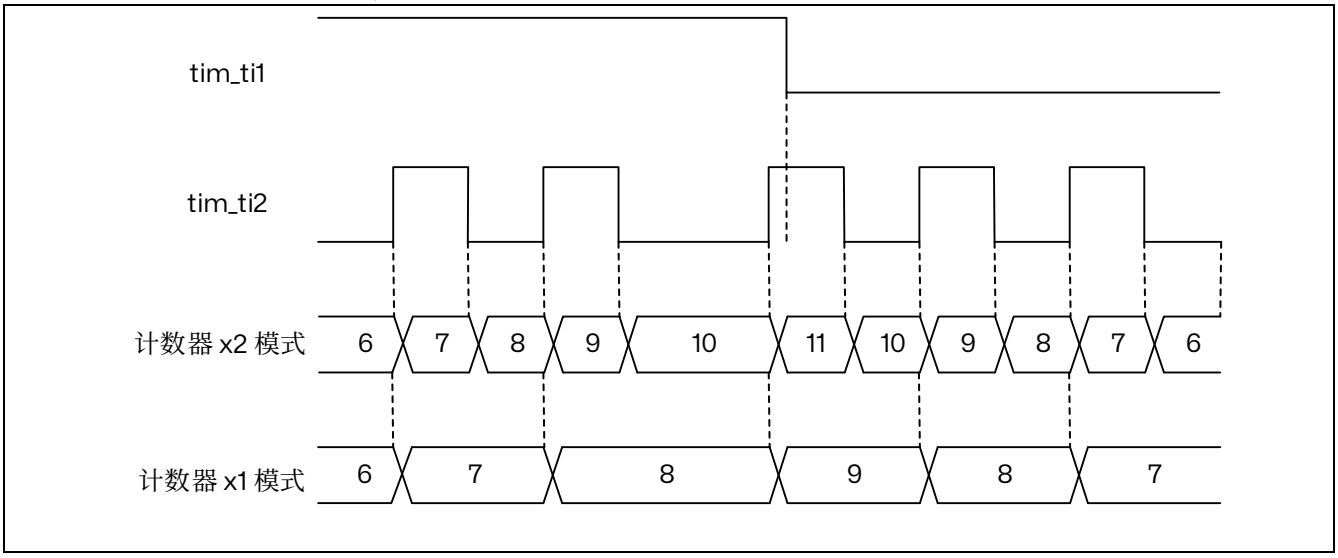



图 23.51 方向加时钟编码器模式

### 方向时钟编码器模式

在下图中的“方向时钟”模式下，时钟在两条线上提供，根据方向一次使用一条，以便有一条向上计数时钟线和一条向下计数时钟线。

此模式通过 TIMx_SMCR 寄存器中的 SMS[3:0]位域使能，如下所示：

- 1100：x2 模式，计数器在两条时钟线的上升沿和下降沿都更新。CC1P 和 CC2P 位编码时钟空闲状态。CCxP=0 对应高电平空闲状态（参考下图），CCxP=1 对应低电平空闲状态（参考下图）。
- 1101：x1 模式，计数器在单个时钟边沿更新，根据 CC1P 和 CC2P 位的值而定。CCxP=0 对应下降沿敏感性和高电平空闲状态（参考下图），CCxP=1 对应上升沿敏感性和低电平空闲状态（参考下图）。

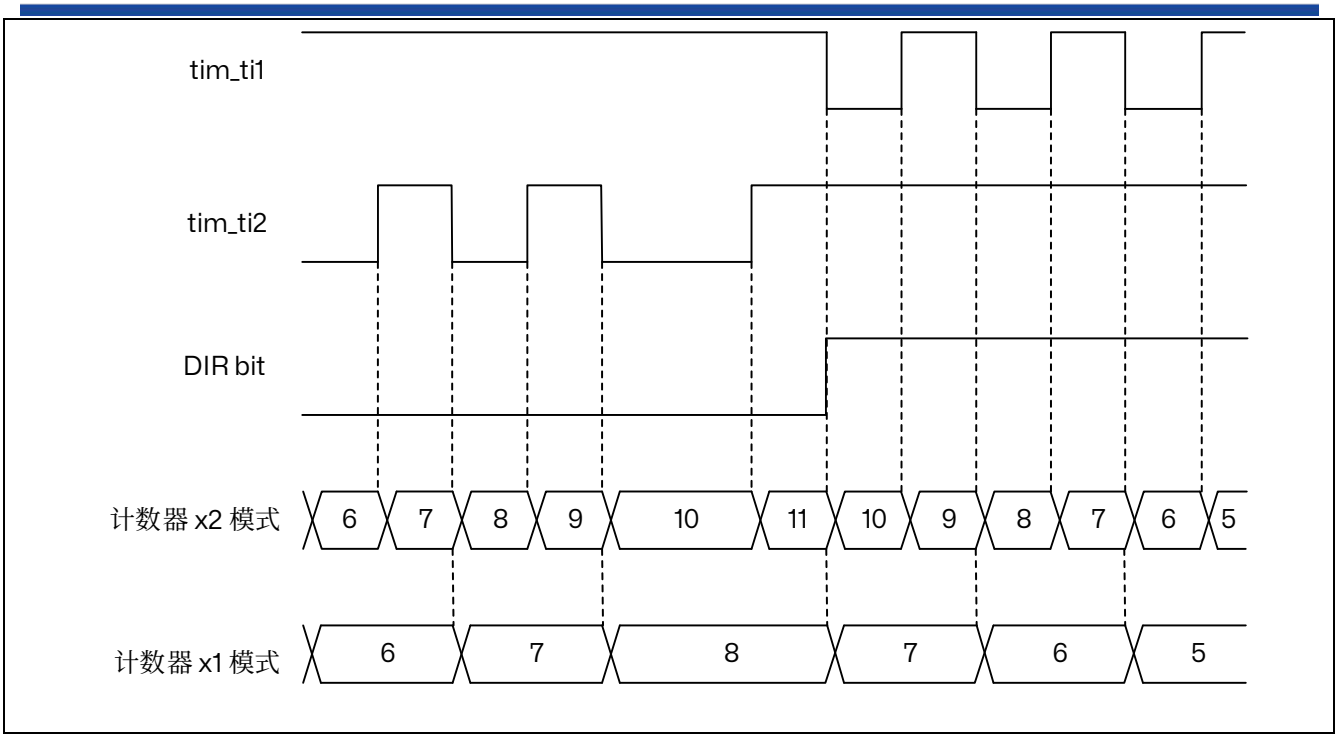



图 23.52 方向时钟编码器模式（CC1P=CC2P=0）

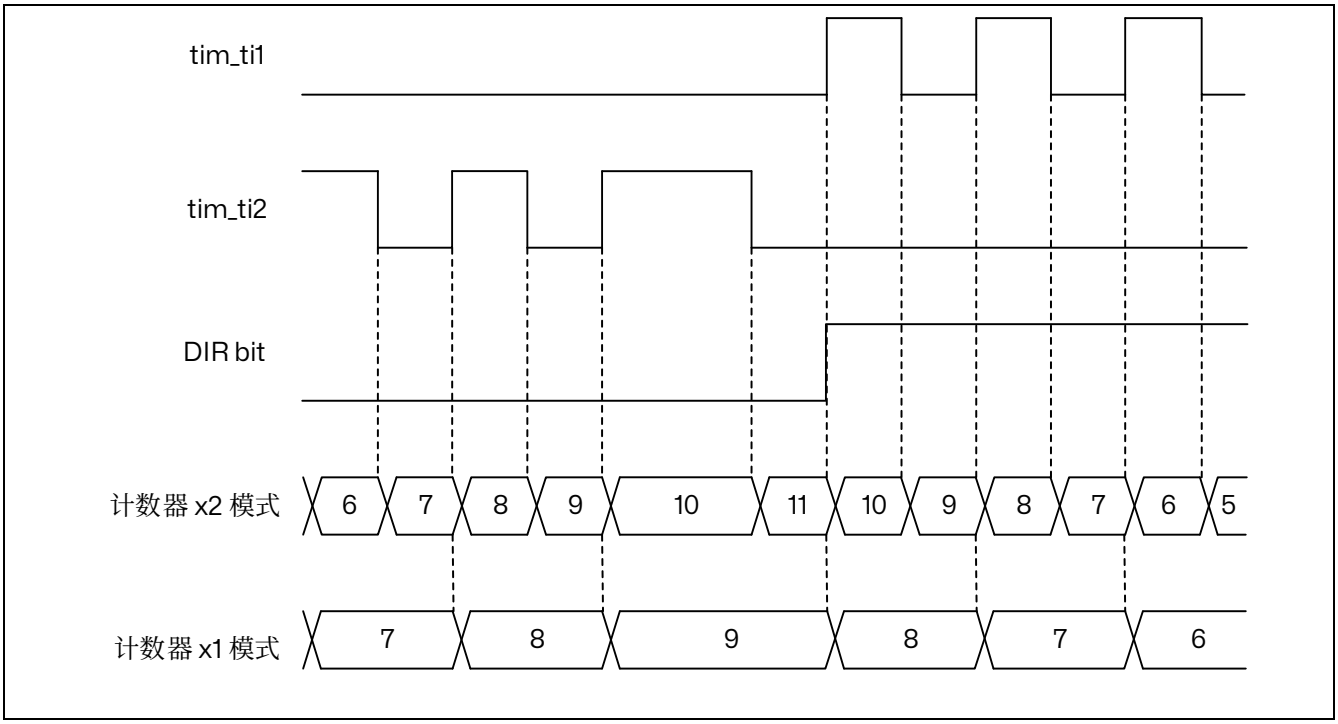



图 23.53 方向时钟编码器模式（CC1P=CC2P=1）

下表详细说明了方向时钟模式对任何输入转变的操作方式。

表 23.14 计数方向与编码器信号及极性设置的关系

| 方向时钟模式 | SMS[3:0] | 相对信号的电平（tim_ti1fp1 对应 tim_ti2, tim_ti2fp2 对应 tim_ti1） | tim_ti1fp1 信号 | | tim_ti2fp2 信号 | |
|---|---|---|---|---|---|---|
| | | | 上升 | 下降 | 上升 | 下降 |
| x2 模式 CCxP=0 | 1100 | 高 | 递减 | 递减 | 递增 | 递增 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| x2 模式 CCxP=1 | 1100 | 高 | 不计数 | 不计数 | 不计数 | 不计数 |
| | | 低 | 递减 | 递减 | 递增 | 递增 |
| x1 模式 CCxP=0 | 1101 | 高 | 不计数 | 递减 | 不计数 | 递增 |
| | | 低 | 不计数 | 不计数 | 不计数 | 不计数 |
| x1 模式 CCxP=1 | 1101 | 高 | 不计数 | 不计数 | 不计数 | 不计数 |
| | | 低 | 递减 | 不计数 | 递增 | 不计数 |

### 索引输入

计数器可以通过来自编码器的索引信号进行复位，以指示绝对参考位置。索引信号必须连接到 tim_etr_in 输入。它可以使用数字输入滤波器进行滤波。

通过设置 TIMX_ECR 寄存器中的 IE 位，可以使能索引功能。只有在编码器模式下，当 SMS[3:0]位域具有以下值时，才可以设置 IE 位：0001、0010、0011、1010、1011、1100、1101、1110、1111。

根据下图所示，有几种可设置的编码器选项，用于索引脉冲的调节：

- 与 A 和 B 门控：脉冲宽度为 1/4 的通道周期，与 A 和 B 边缘对齐
- 与 A（或与 B）门控：脉冲宽度为 1/2 的通道周期，与通道 A（或通道 B）的两个边缘对齐
- 非门控：脉冲宽度可达一个通道周期，不与任何边缘对齐

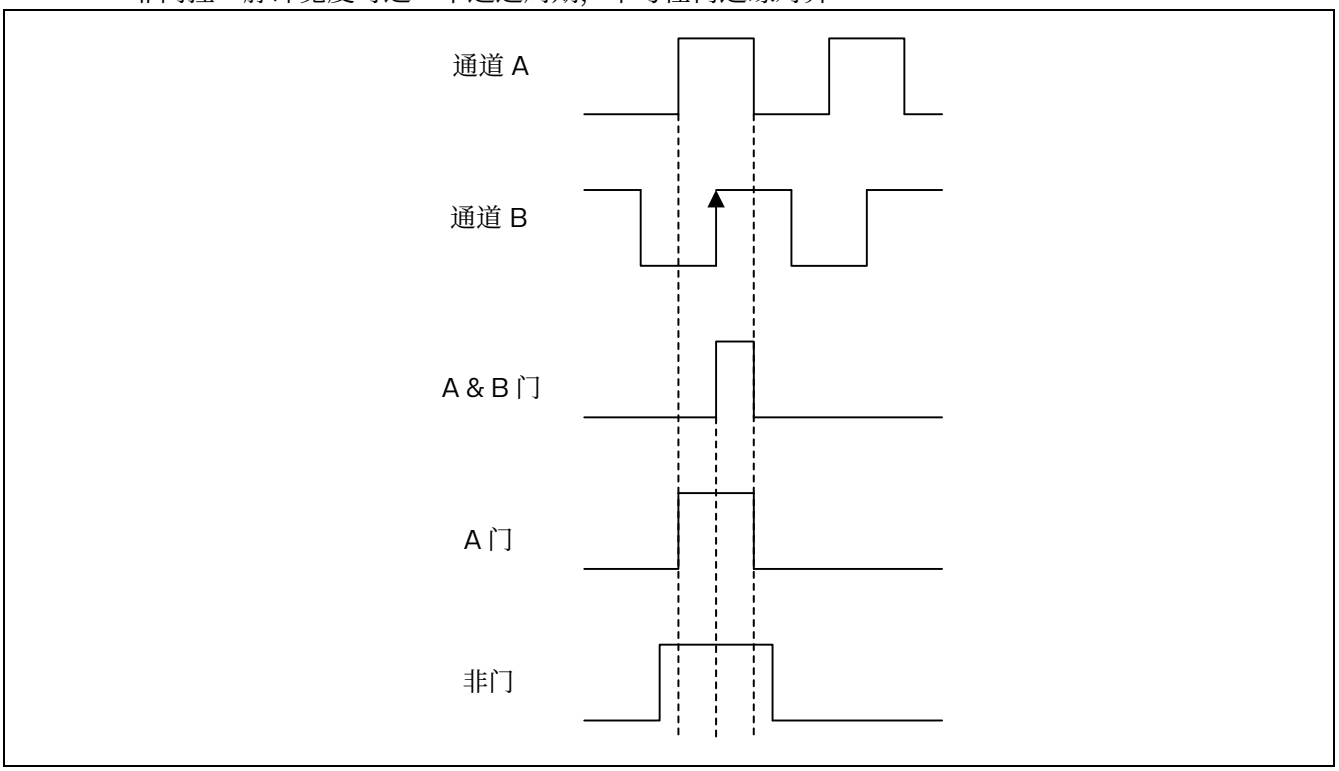



图 23.54 索引门控选项

电路可以容忍索引信号的抖动，无论门控模式如何，如下图所示。

在非门控模式下，信号必须严格低于 2 个编码器周期。如果脉冲宽度大于或等于 2 个编码器周期，则

计数器会多次复位。

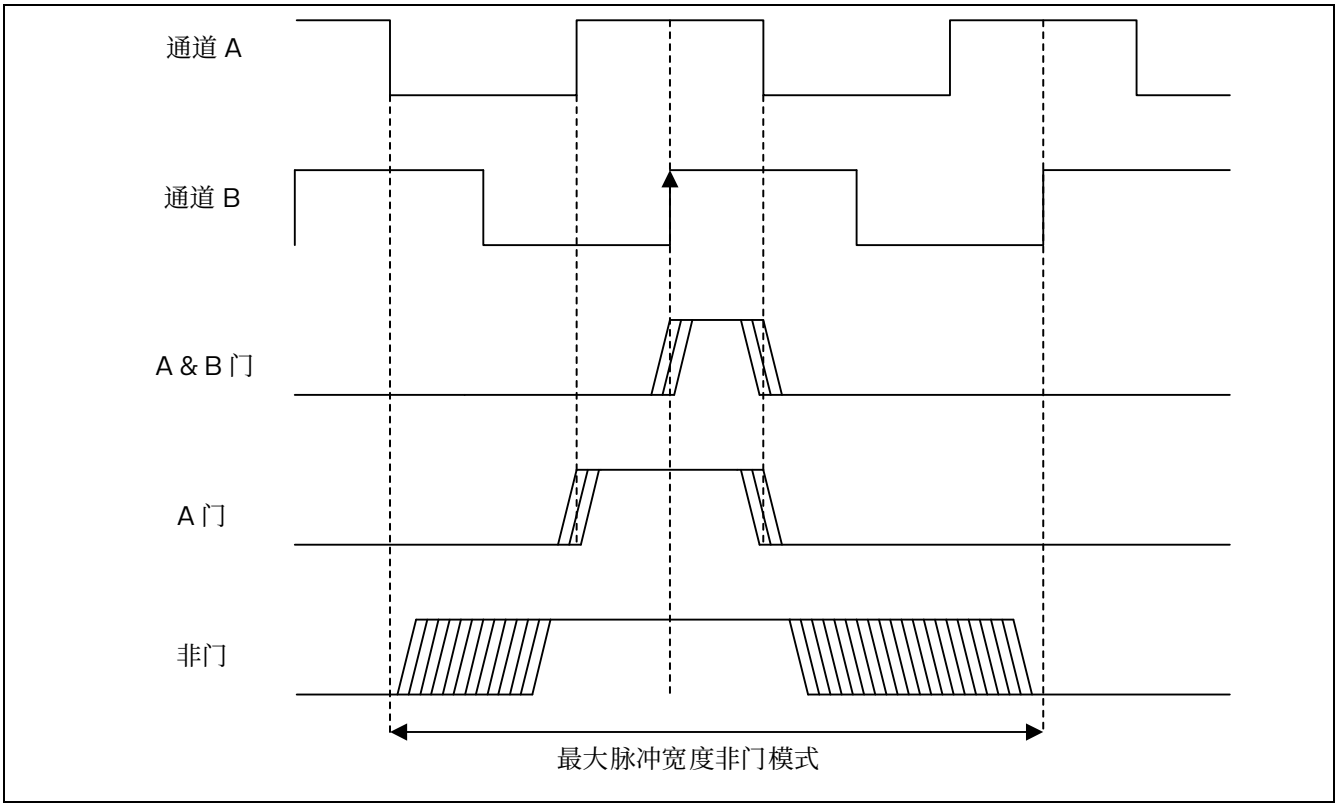



图 23.55 抖动的索引信号

定时器支持 3 种门控选项，不需要任何特定的编程。只需要定义在哪个编码器状态（即通道 A 和通道 B 状态组合）上，索引必须同步，使用 TIMx_ECR 寄存器中的 IPOS[1:0]位域。

索引检测事件根据计数方向的不同而有所不同，以确保速度反转时的对称操作：

- 递增计数时（DIR 位=0），计数器被复位。
- 递减计数时，计数器被设置为 TIMx_ARR。

这使得索引可以在相同的机械角度位置上生成，无论计数方向如何。下图显示了在一个简单的例子中（一个编码器每次机械旋转提供 4 个边缘），索引在哪个位置生成。

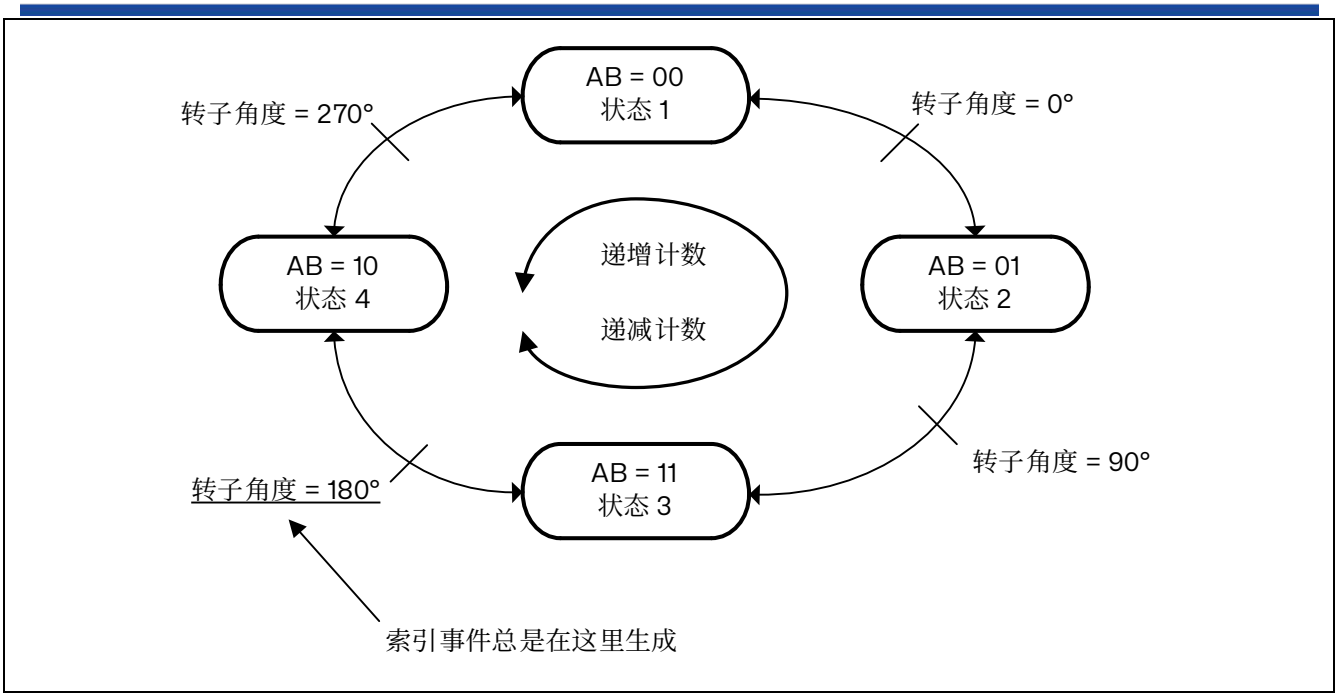



图 23.56 IPOS[1:0] = 11 时的索引生成

下图展示了 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会自动根据计数方向进行调整：

- 递增计数时（DIR 位=0），当编码器状态为“11”（ChA=1，ChB=1）时，计数器设为 0。
- 递减计数时（DIR 位=1），当退出“11”状态时，计数器设为 TIMx_ARR。

可以在索引检测事件发生时发出中断。

箭头指示在哪个转换上生成索引事件中断。

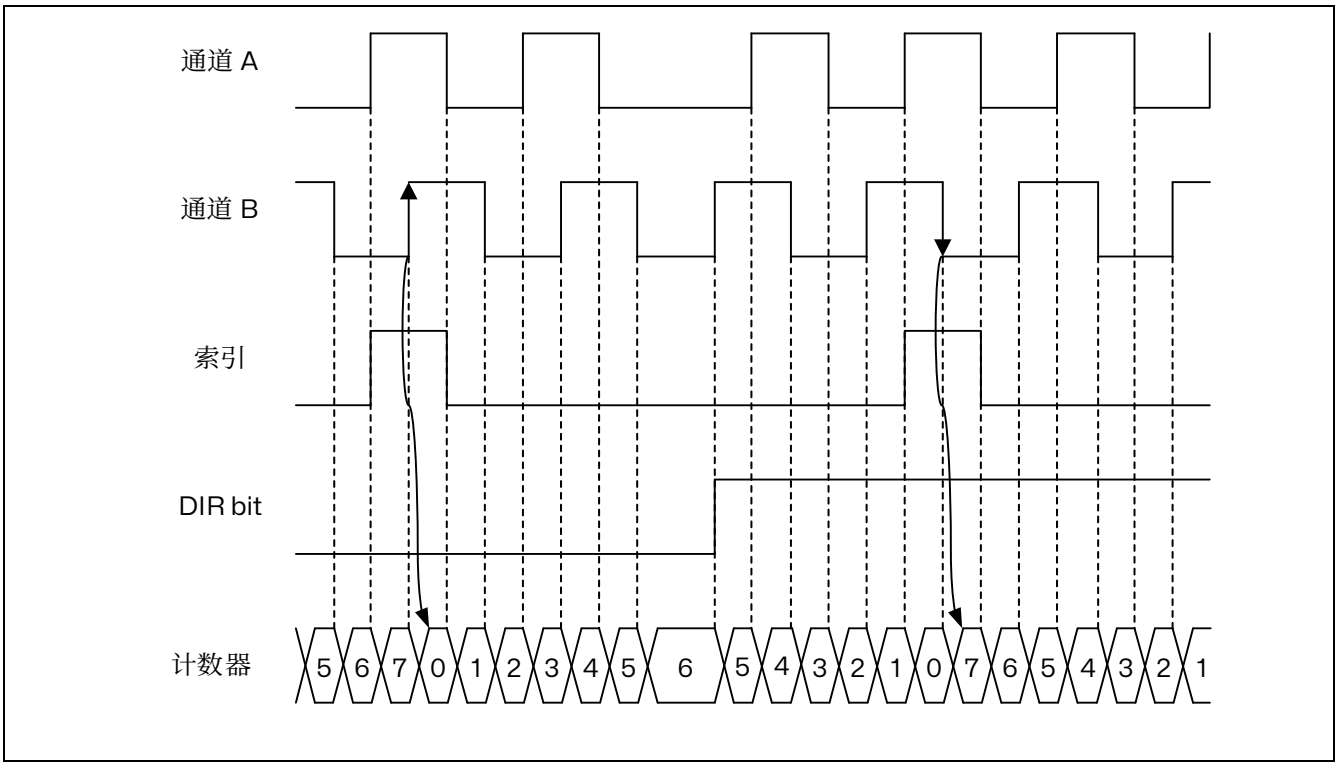



图 23.57 通道 A 上带索引门控的计数器读取（IPOS[1:0] = 11）

下图展示了非门控模式的波形和相应值。箭头指示在哪个转换上生成索引事件。

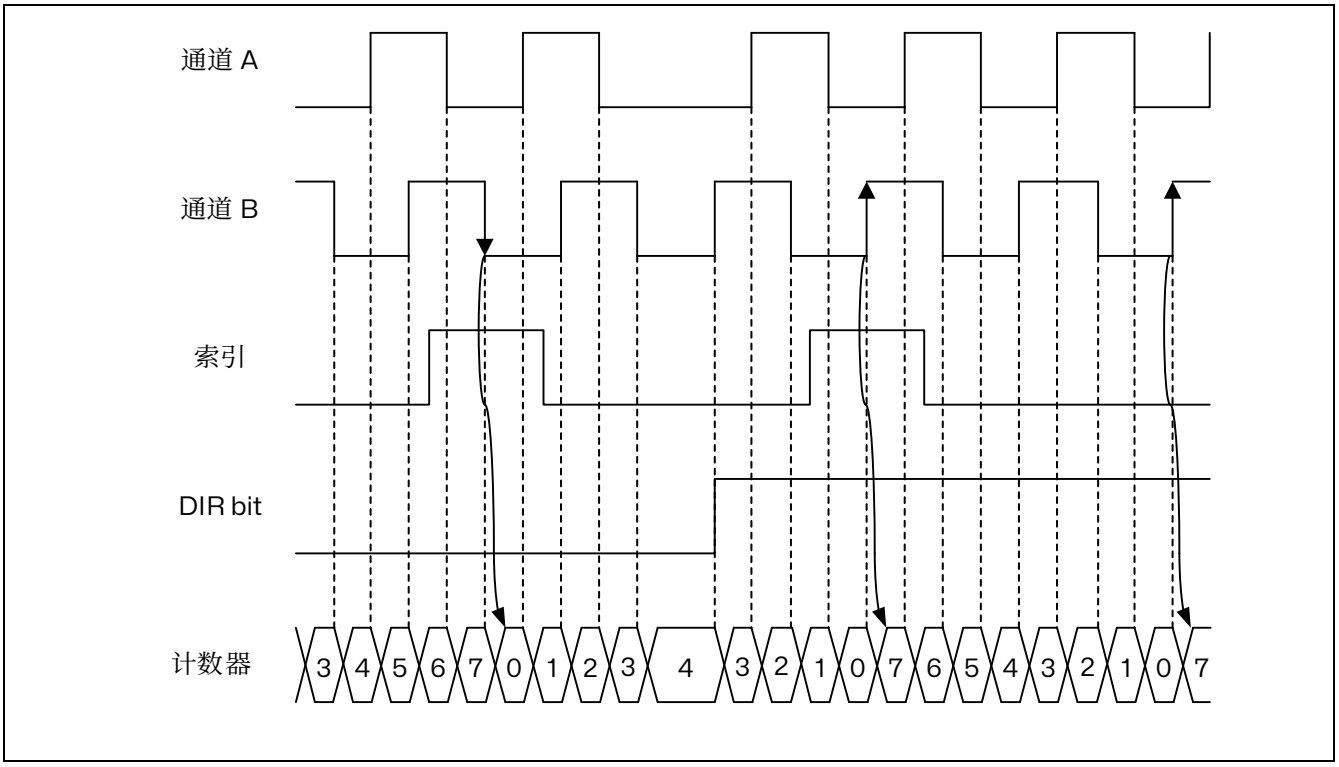



图 23.58 非门控模式下索引的计数器读取（IPOS[1:0] = 00）

下图展示了针对各种脉冲对齐情况下的“在 A 和 B 门控”模式处理方式。箭头指示在哪个转换上生成索引事件。

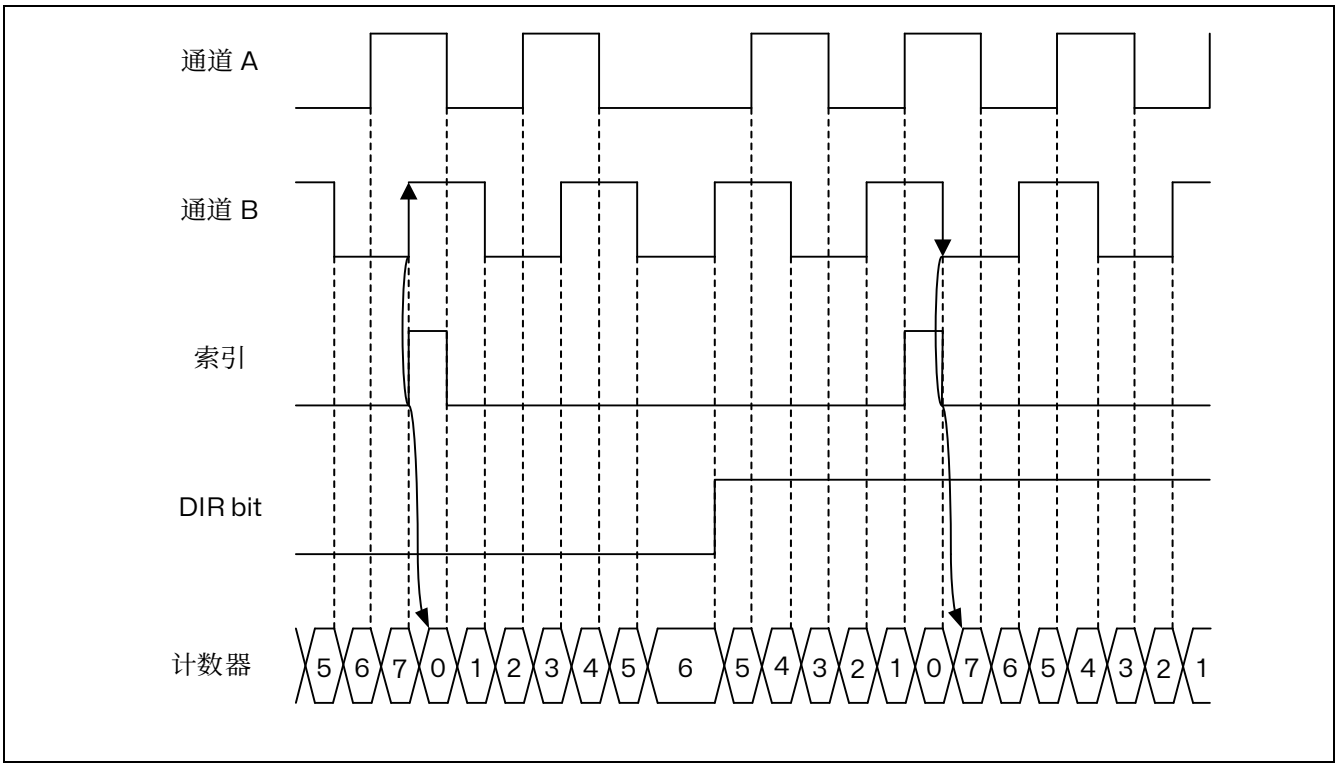
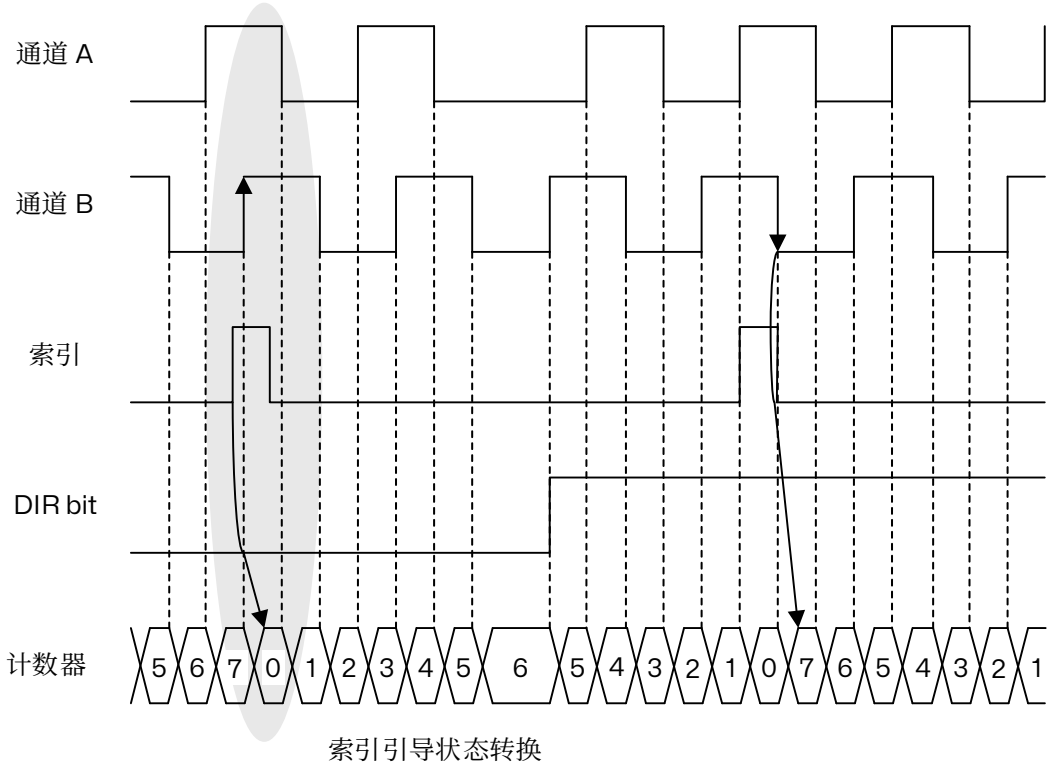
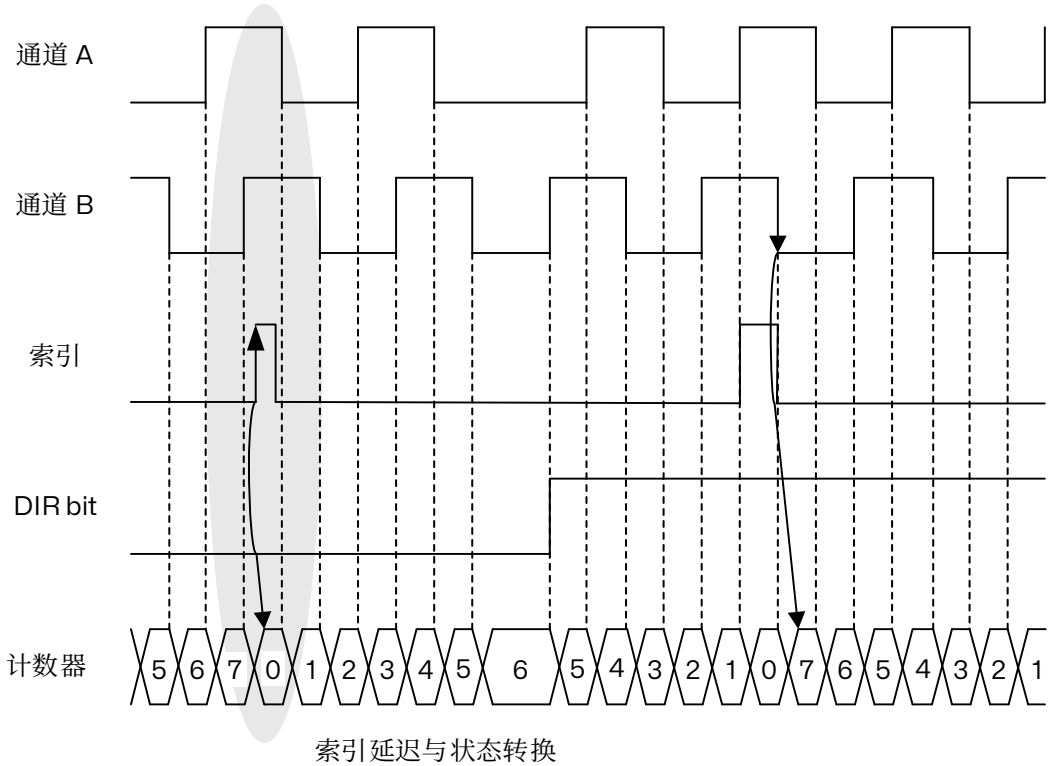



图 23.59 通道 A 和 B 上带索引门控的计数器读取

下两张图详细说明了后续索引脉冲可能比编码器时钟周期的四分之一窄的情况。




索引引导状态转换




索引延迟与状态转换

图 23.60 窄索引脉冲情况下的编码器模式行为（IPOS[1:0] = 11）

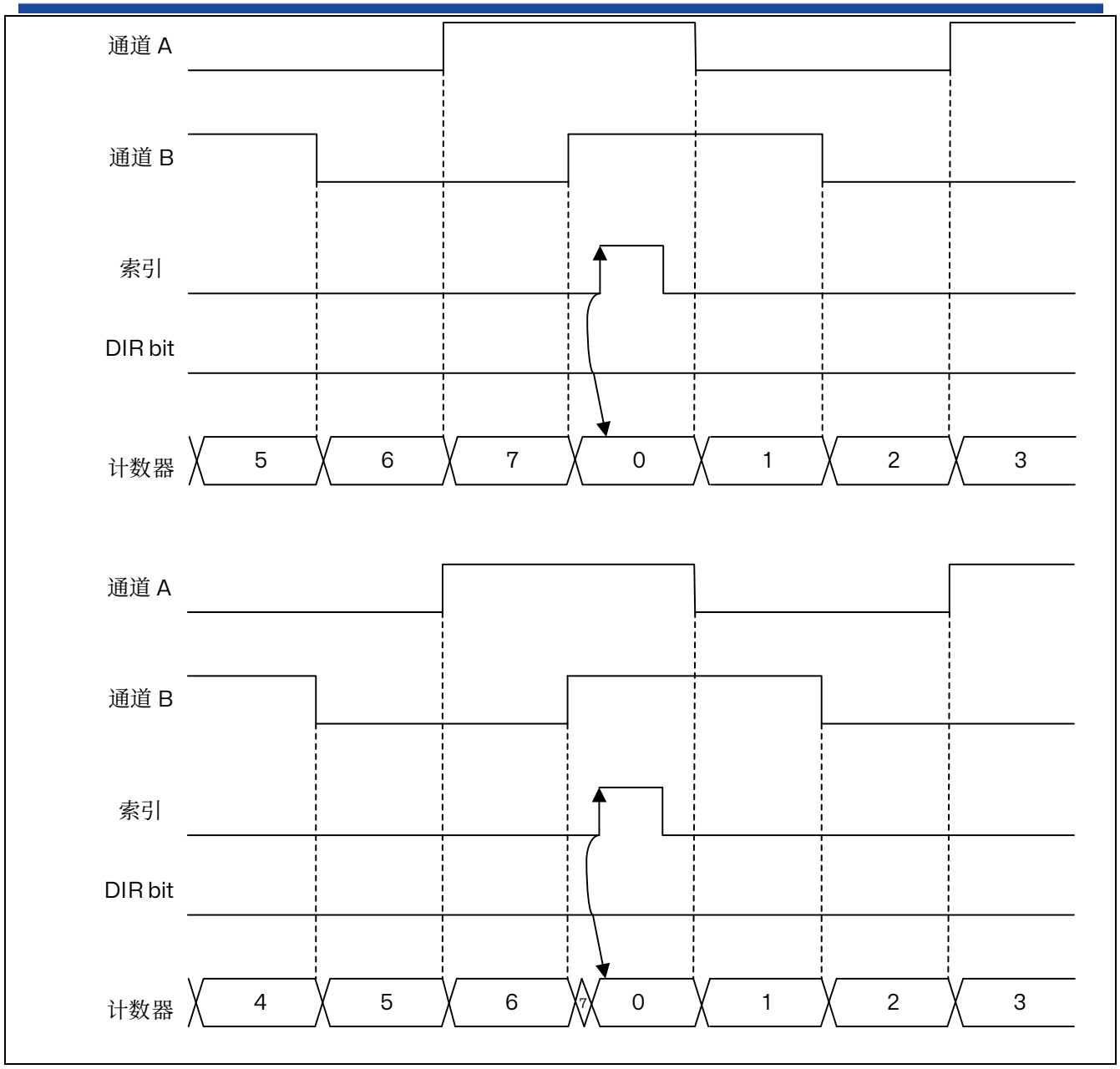



图 23.61 计数器重置窄索引脉冲（更近距离，ARR = 0x07）

下图说明了 x1 和 x2 模式下的索引管理方式。

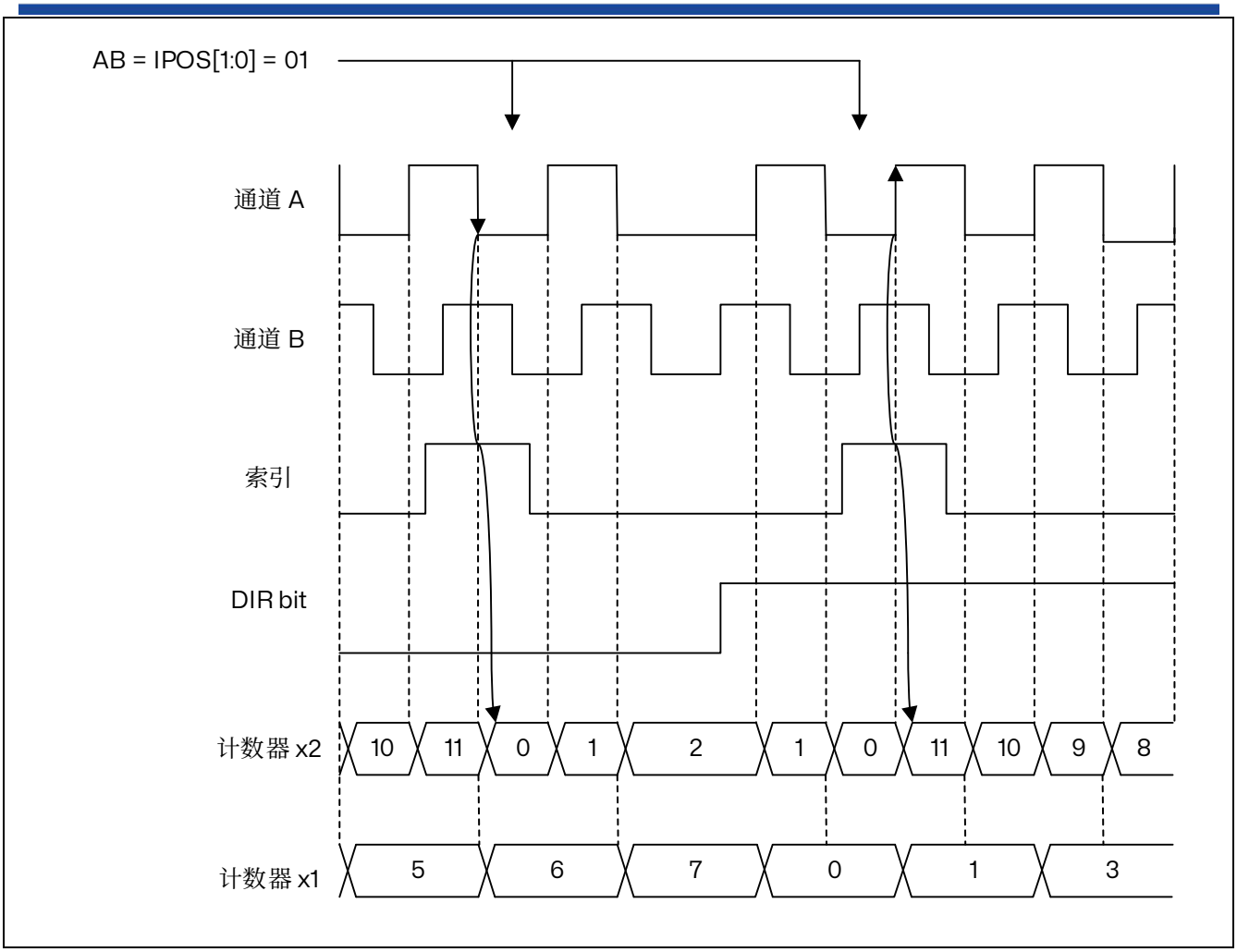



图 23.62 x1 和 x2 模式下的索引行为（IPOS[1:0] = 01）

方向索引灵敏度

TIMx_ECR 寄存器中的 IDIR[1:0]位域允许索引仅在选定计数方向上有效。

下图显示了根据 IDIR[1:0]值索引和计数器重置事件之间的关系。

*注：当* IE *位被复位时（禁用索引模式），必须写入* IDR[1:0] *位域。*

*注：在时钟+方向模式下不支持定向索引灵敏度。当* SMS[3:0] = 1010 *或* 1011 *时，* IDIR[1:0] *必须设置为* 00。

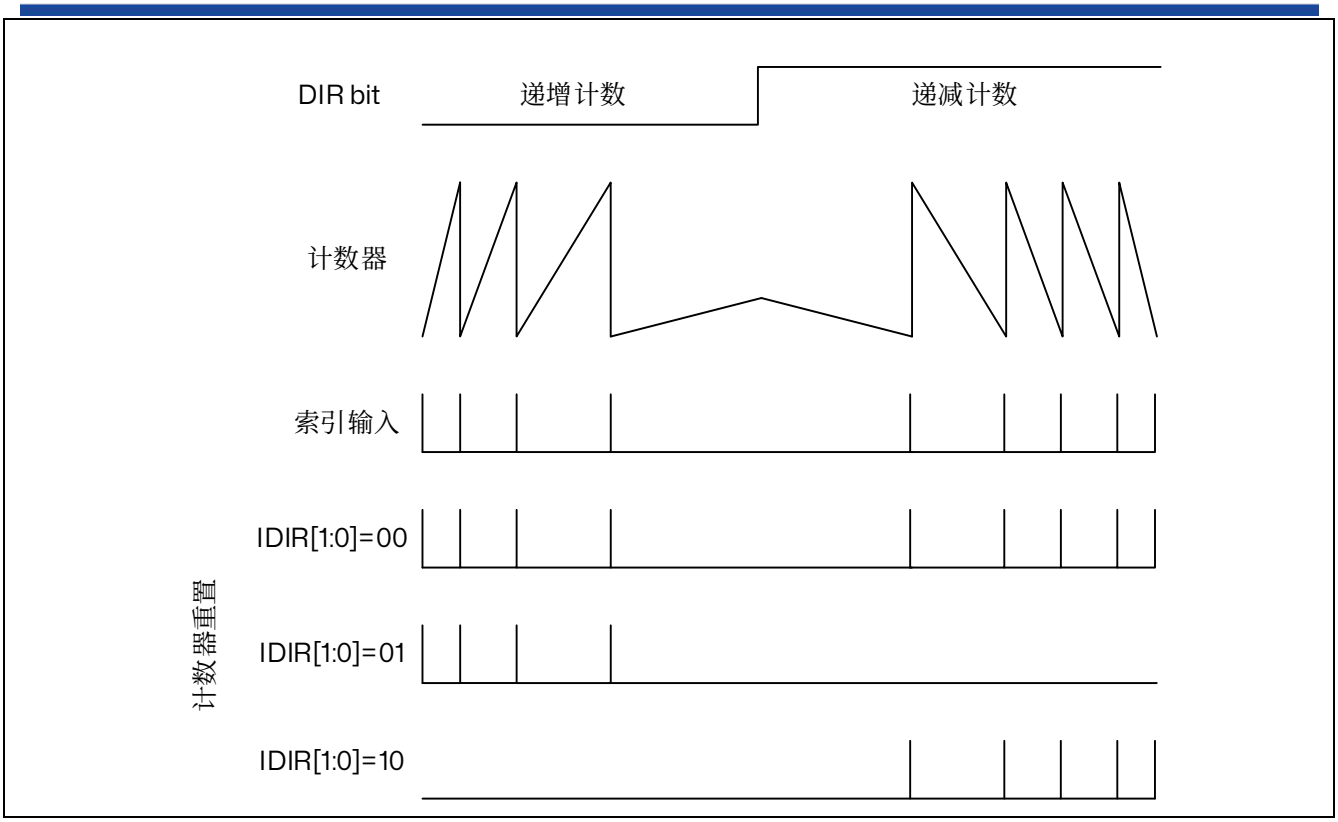



图 23.63 定向索引灵敏度

### 特殊的第一索引事件管理

TIMx_ECR 寄存器中的 FIDX 位允许仅捕获一次索引，如下图所示。一旦第一个索引到达，任何后续索引都会被忽略。如果需要，可以通过将 FIDX 位写入 0 并再次将其设置为 1 来重新启动电路。

*注：当 FIDX = 1 时，如果方向在位置 0 处改变（索引激活），索引可以发布两次（IDXF 标志设置）。*

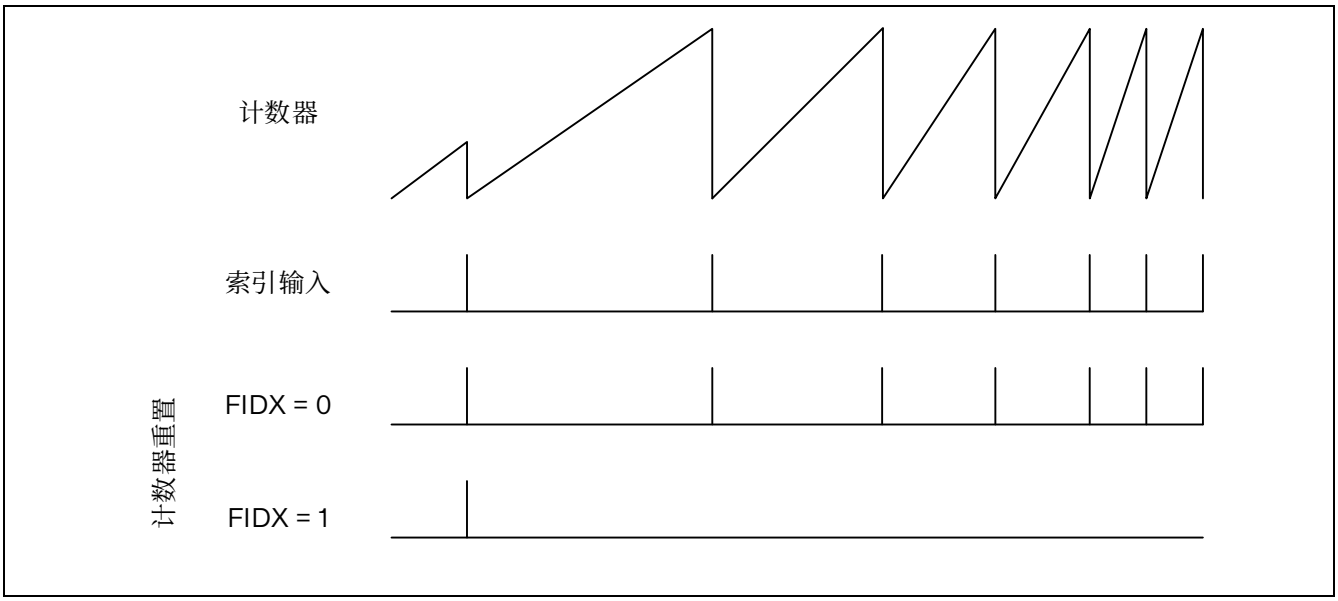



图 23.64 作为 FIDX 位设置的功能的计数器重置

### 非正交模式下的索引管理

下面两张图详细说明了当 SMS[3:0]位域等于 1010、1011、1100、1101 时，如何在方向时钟模式和时钟加方向模式下管理索引。

对于这两种模式，索引灵敏度通过 IPOS[0]位进行设置，具体如下：

- IPOS[0] = 0：在时钟低电平上检测索引
- IPOS[0] = 1：在时钟高电平上检测索引

IPOS[1]位不重要。

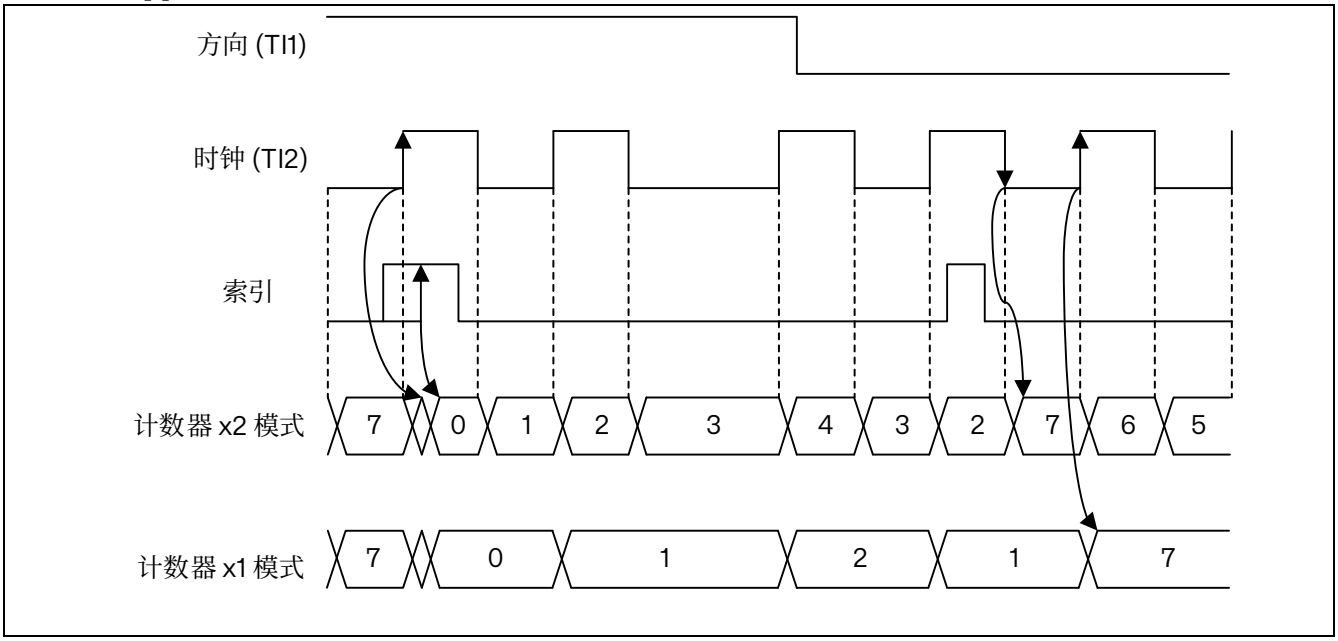



图 23.65 时钟+方向模式下的索引行为，IPOS[0] = 1

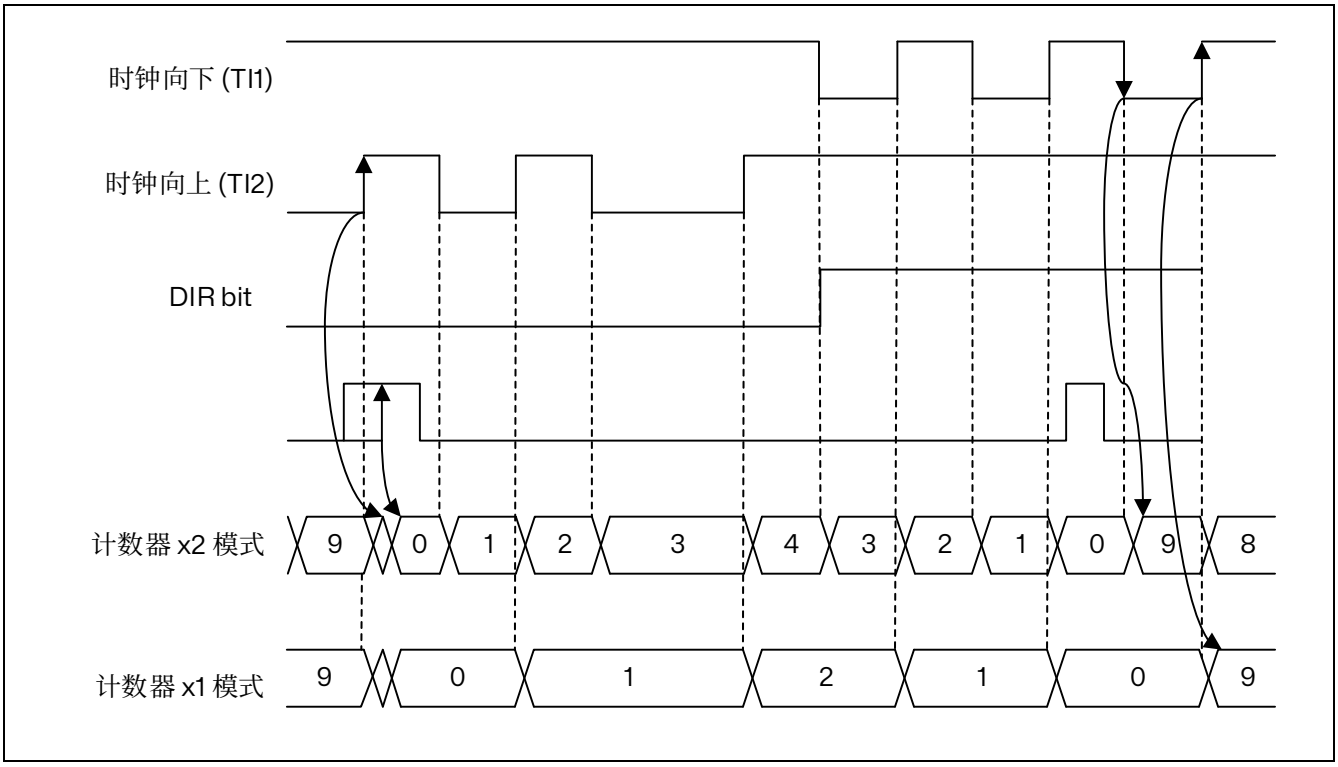



图 23.66 方向时钟模式下的索引行为，IPOS[0] = 1

### 编码器错误管理

对于具有 2 个正交信号的编码器配置，可以检测到转换错误。2 个输入的读数对应于一个 2 位灰度码，该灰度码可以表示为状态图，如以下图所示。期望单个比特立即更改。错误的转换会在 TIMx_SR 状态寄存器中设置 TERRF 中断标志。如果 TIMx_DIER 寄存器中的 TERRIE 位被设置，将产生转换错误中断。

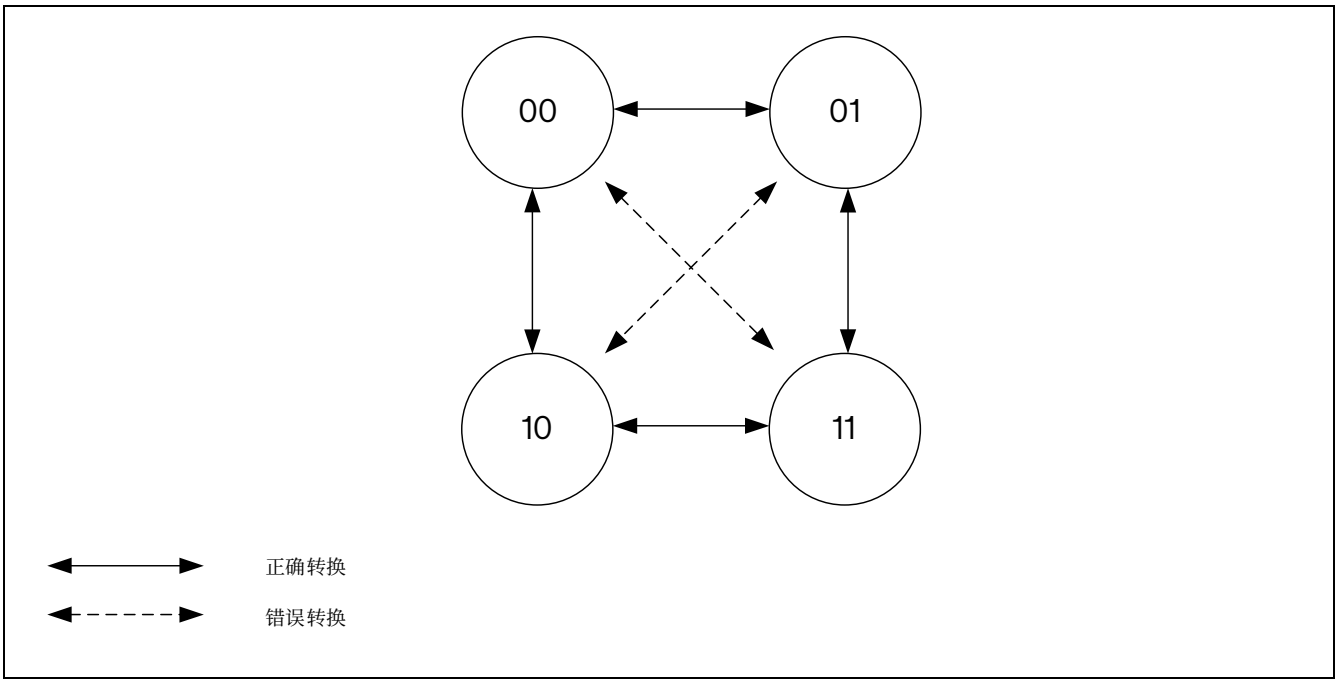



图 23.67 正交编码信号的状态图

对于具有索引信号的编码器，可以检测到导致每转脉冲过多的异常操作。一个编码器每转提供 N 个脉冲，则每转有 4xN 个计数。索引信号每 4xN 个时钟周期重置一次计数器。

如果计数器的值从 TIMx_ARR 增加到 0 或从 0 减少到 TIMxARR 的值而没有索引事件，则报告为索引位置错误。

可使用 TIMx_ARR 寄存器来编程溢出阈值。一个 1000 线的编码器导致计数器的值在 0 和 3999 之间（在 4 倍读取模式下）。必须通过设置 TIMx_ARR = 3999 + 1 = 4000 来编程溢出检测阈值。

错误断言被延迟到在递增计数时从 0 到 1 的转换。这是为了处理门控 A 和 B 模式中的窄索引脉冲，如下图所示。

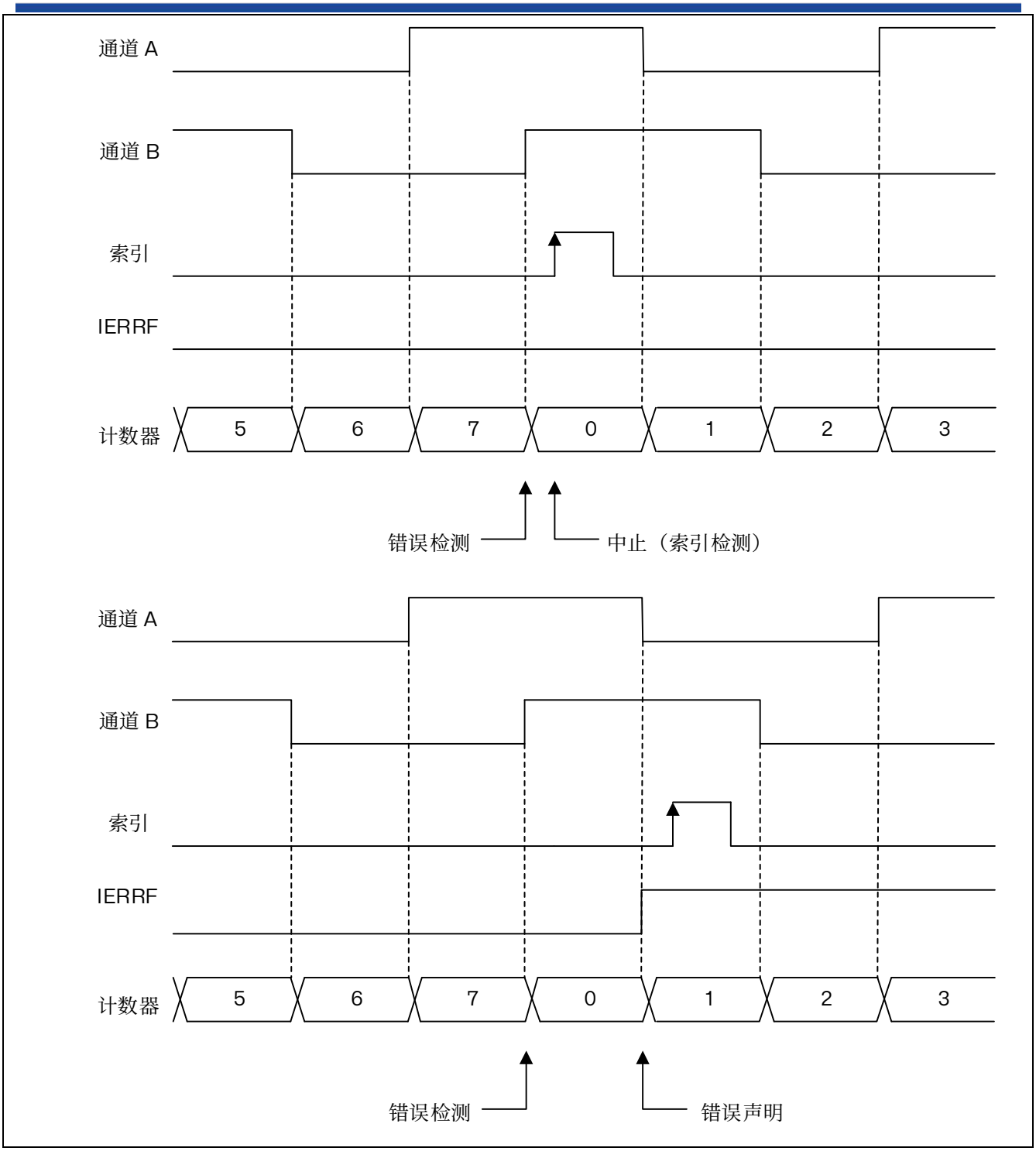



图 23.68 递增计数编码器错误检测

在递减计数模式下，检测是由从 1 到 0 的初步转换引起的。这是为了处理门控 A 和 B 模式中的窄索引脉冲，如下图所示，以避免编码器在索引检测后立即在 TIMx_ARR 和 0 之间抖动时出现任何错误检测。

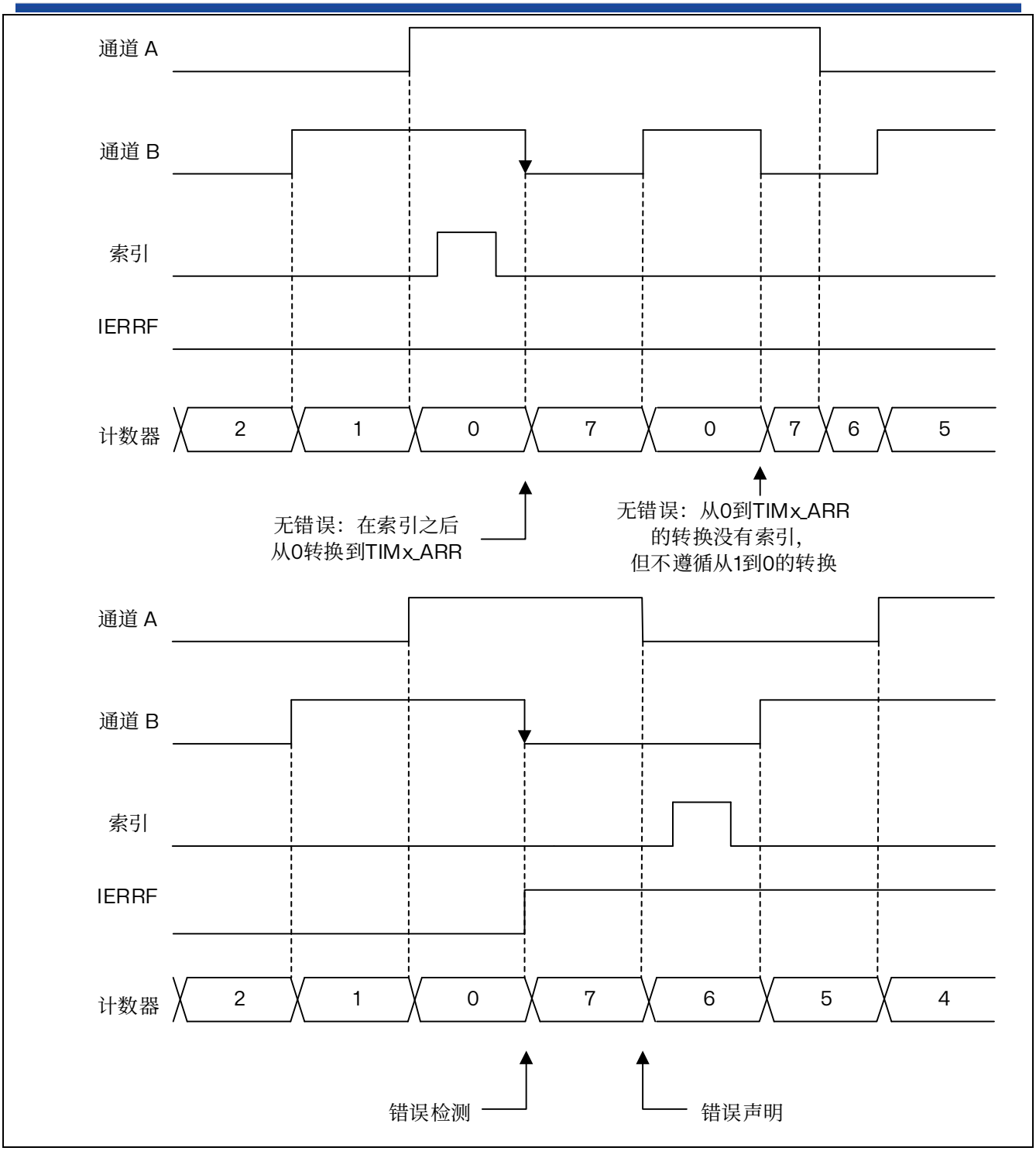



图 23.69 递减计数编码器错误检测

索引错误会在 TIMx_SR 状态寄存器中设置 IERRF 中断标志。如果 TIMx_DIER 寄存器中的 IERRIE 位被设置，将产生索引错误中断。

### 功能编码器中断

编码器模式下也可使用以下中断：

- 方向改变：在编码器模式下，任何计数方向的改变都会导致 TIMx_CR1 寄存器中的 DIR 位发生翻转。方向改变会设置 TIMx_SR 状态寄存器中的 DIRF 中断标志。如果 DIRIE 位在 TIMx_DIER 寄

存器中被设置，则产生方向改变中断。

- 索引事件：索引事件会设置 TIMx_SR 状态寄存器中的 IDXF 中断标志。如果 IDXIE 位在 TIMx_DIER 寄存器中被设置，则产生索引中断。

### 运行时更新编码器模式时从机模式选择预加载

在运行过程中，可能需要从一个编码器模式切换到另一个编码器模式。这通常在高速度下完成，以降低更新中断频率，从 x4 模式切换到 x2 模式再到 x1 模式，如下图所示。

为此，可以预加载 SMS[3:0]位。通过设置 TIMx_SMCR 寄存器中的 SMSPE 使能位可以使能此功能。从 SMS[3:0]预加载值转换到有效值可以由 TIMx_SMCR 寄存器中的 SMSPS 位选择触发。

- SMSPS = 0：当计数器递增时溢出并在计数器递减时下溢时，由更新事件(UEV)触发的转换。只有在索引被禁用时（位 IE = 0）才可以使用此模式。
- SMSPS = 1：由索引事件触发的转换。

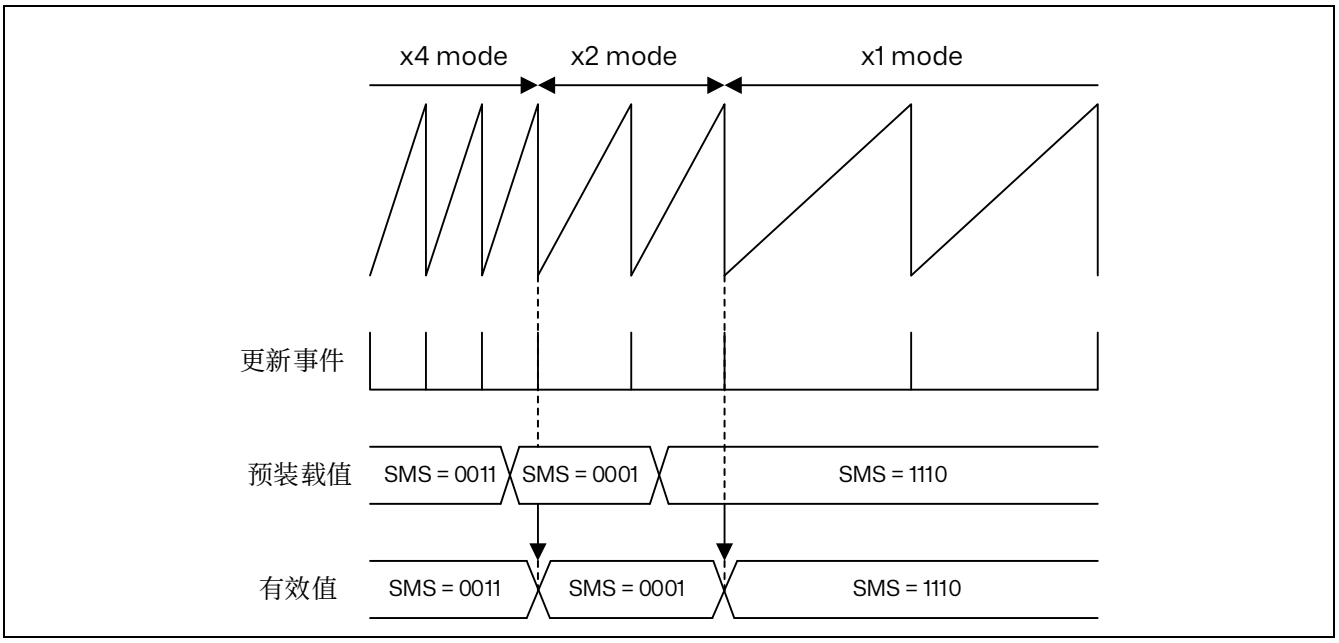



图 23.70 更新时带有预加载转移的编码器模式更改（SMSPS = 0）

### 编码器时钟输出

编码器模式的操作原理并不完全适合于高分辨率速度测量，在低速时，因为它需要相对较长的积分时间才能获得足够数量的时钟边缘和精确的测量。

在低速时，更好的解决方案是进行边到边的时钟周期测量。这可以通过使用从属定时器来实现。定时器可以在 tim_trgo 输出上输出编码器时钟信息。然后，从属定时器可以进行周期测量并为每个编码器时钟边缘提供速度信息。

通过将 TIMx_CR2 寄存器中的 MMS[3:0]位域设置为 1000 可以使能此模式。对于以下 SMS[3:0]值有效：0001、0010、0011、1010、1011、1100、1101、1110、1111。任何可能导致意外行为的其他 SMS[3:0]代码是不被允许的。

### 23.4.19 方向位输出

可以将方向信号从定时器输出到 tim_oc3n 和 tim_oc4 输出信号（TIMx_CR1 寄存器中的 DIR 位的副本）。这是通过将 TIMx_CCMR2 寄存器中的 OC3M[3:0]或 OC4M[3:0]位域设置为 1011 来实现的。

此功能可用于监测编码器模式下的计数方向（或旋转方向），或在中心对齐的 PWM 模式下有一个信号指示上升/下降相位。

### 23.4.20 UIF 位重映射

TIMx_CR1 寄存器中的 UIFREMAP 位强制将更新中断标志 UIF 连续复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许以原子方式读取计数器的值和一个潜在的由 UIFCPY 标志指示的翻转条件。在特定情况下，它可以避免由背景任务（读取计数器）和中断（更新中断）之间的处理引起的竞争条件，从而简化计算。

UIF 和 UIFCPY 标志之间没有延迟。

### 23.4.21 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，可将通道 1 的输入滤波器连接到异或门的输出，从而将 tim_ti1，tim_ti2，tim_ti3 这三个输入引脚组合在一起。

异或输出可与触发或输入捕获等所有定时器输入功能配合使用。

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。

### 23.4.22 定时器和外部触发的同步

TIMx 定时器以下列模式与外部触发实现同步：复位模式、门控模式、触发模式、复位+触发和门控+复位模式。

**从模式：复位模式**

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 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，tim_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 tim_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 和 CC1NP=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 tim_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 tim_ti1 出现一个上升沿；当 tim_ti1 上升时，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，中断请求或 DMA 请求可以被发送（取决于 TIMx_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。

在 tim_ti1 上升沿和计数器的实际复位之间的延时取决于 tim_ti1 输入端的重同步电路。

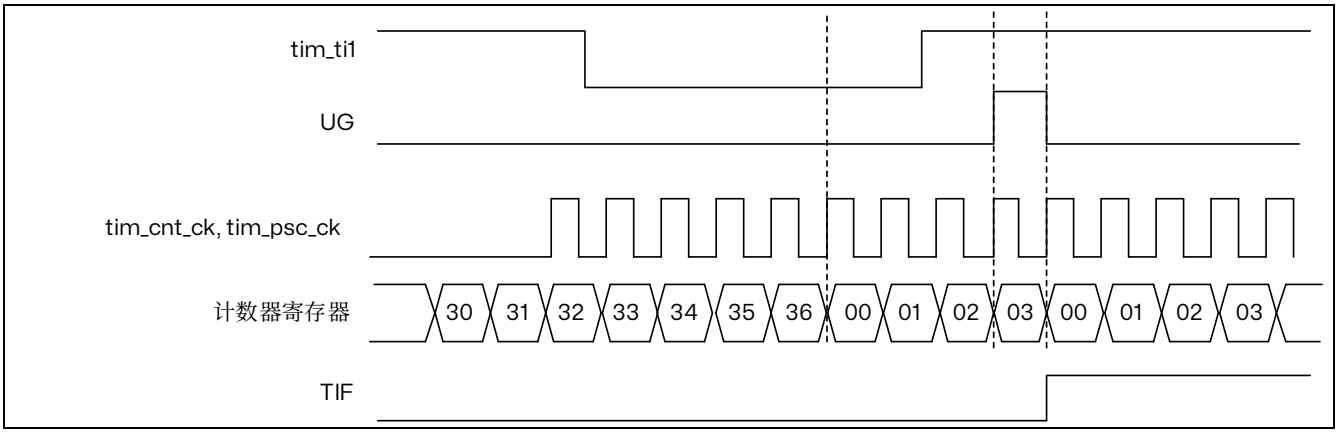



图 23.71 复位模式下的控制电路

**从模式：门控模式**

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 tim_ti1 为低时向上计数：

- 配置通道 1 以检测 tim_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 和 CC1NP=0 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 tim_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 tim_ti1 为低，计数器开始依据内部时钟计数，一旦 tim_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。

tim_ti1 上升沿和计数器实际停止之间的延时取决于 tim_ti1 输入端的重同步电路。

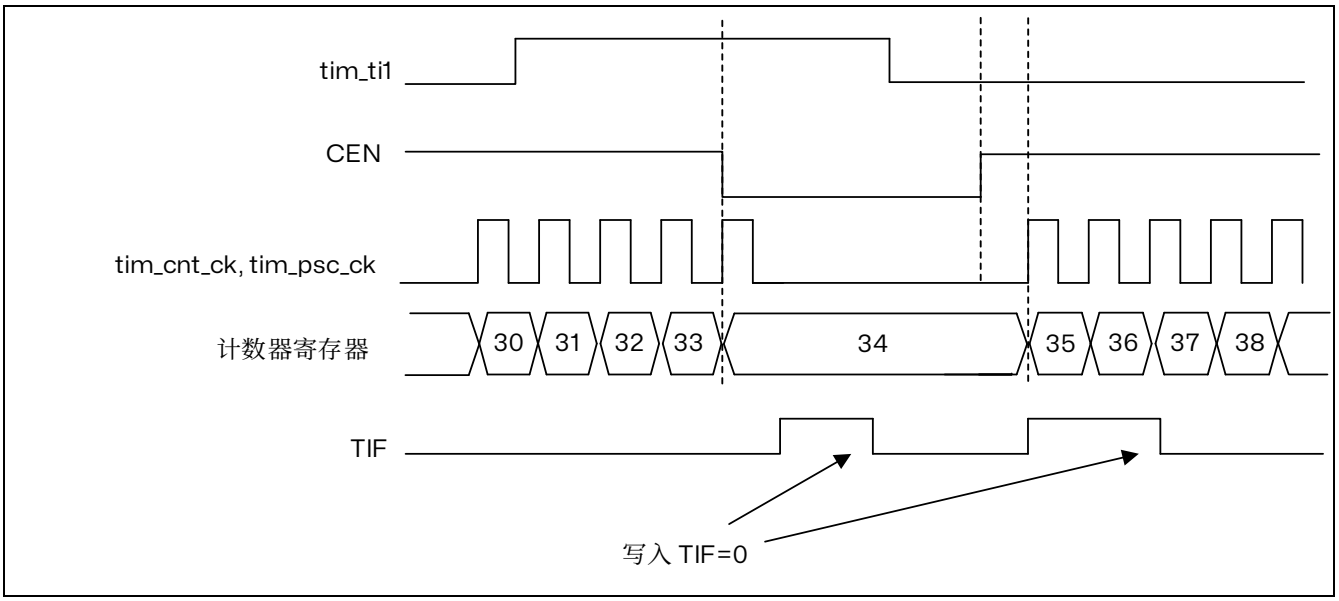



图 23.72 门控模式下的控制电路

*注：由于门控模式作用于电平而非边沿，因此在门控模式下，“CCxP=CCxNP=1”（同时检测上升沿和下降沿）不发挥任何作用。*

**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 tim_ti2 输入的上升沿开始向上计数：

- 配置通道 2 检测 tim_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 和 CC2NP=0 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 tim_ti2 作为输入源。

当 tim_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。

tim_ti2 上升沿和计数器启动计数之间的延时，取决于 tim_ti2 输入端的重同步电路。

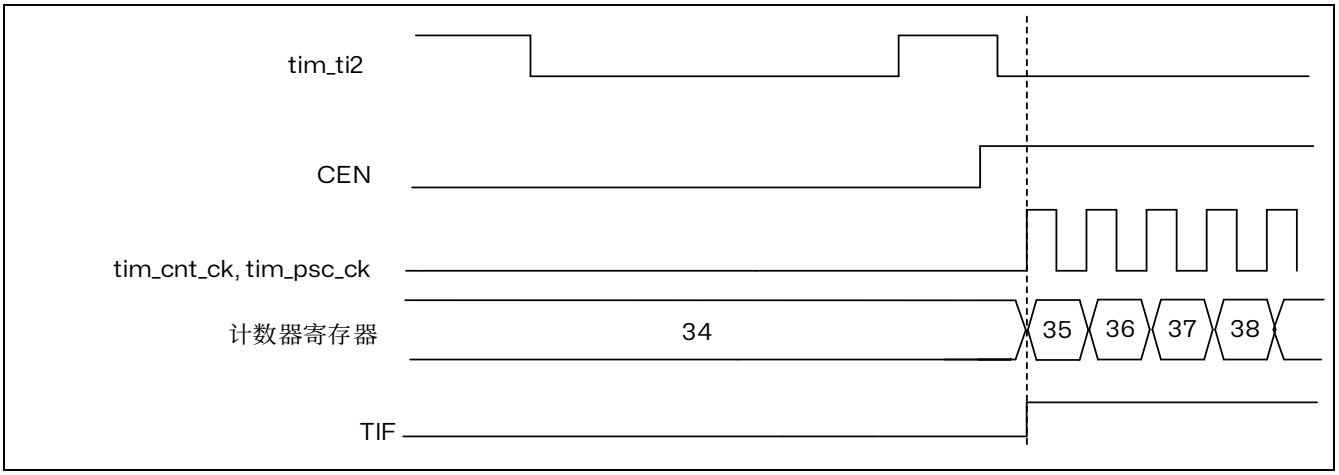



图 23.73 触发器模式下的控制电路

**对实时更新编码器模式的从模式选择预装载**

SMS[3:0]位可以预加载。这是通过在 TIMx_SMCR 寄存器中设置 SMSPE 使能位来实现的。从 SMS[3:0]预加载到有效值的触发器是计数器溢出时发生的更新事件（UEV）。

**从模式 - 组合重置 + 触发模式**

在这种情况下，选择的触发输入（tim_trgi）的上升沿重新初始化计数器，生成寄存器的更新，并启动计数器。

该模式用于单脉冲模式。

**从模式 - 组合重置 + 门控模式**

触发输入（tim_trgi）高时使能计数器。一旦触发变低，计数器停止并复位。计数器的开始和停止都是受控的。

该模式允许检测超出范围的 PWM 信号（占空比超过最大期望值）。

**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，tim_etr_in 信号被用作外部时钟的输入，可以选择另一个输入作为触发输入（复位模式、门控模式或触发模式）。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 tim_etr_in 作为 tim_trgi。

在下面的例子中，一旦在 tim_ti1 上出现一个上升沿，计数器即在 tim_etr_in 的每一个上升沿向上计数一次：

4. 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 tim_etr_in 的上升沿，置 ECE=1 使能外部时钟模式 2。
5. 按如下配置通道 1，检测 TI 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 和 CC1NP = 0 以确定极性（只检测上升沿）
6. 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=00101，选择 tim_ti1 作为输入源。

当 tim_ti1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 tim_etr_in 的上升沿计数。

tim_etr_in 信号的上升沿和计数器实际复位间的延时，取决于 tim_etrp 输入端的重同步电路。

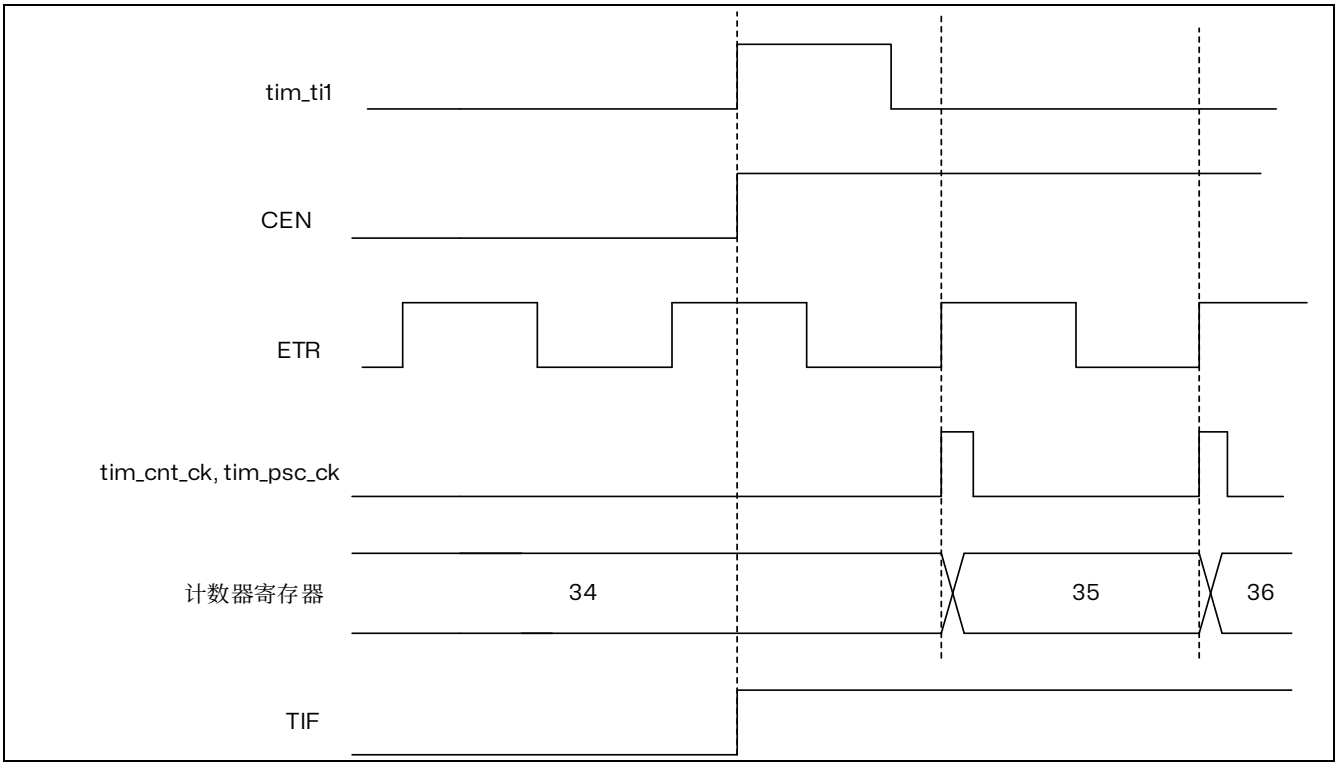



图 23.74 外部时钟模式 2+触发模式下的控制电路

### 23.4.23 定时器同步

TIMx 定时器从内部连接在一起，以实现定时器同步或级联。当某个定时器配置为主模式时，可对另一个配置为从模式的定时器的计数器执行复位、启动、停止操作或为其提供时钟。下图显示了触发选择和主模式选择模块的概况。

下图简要介绍了触发选择和主模式选择框图。

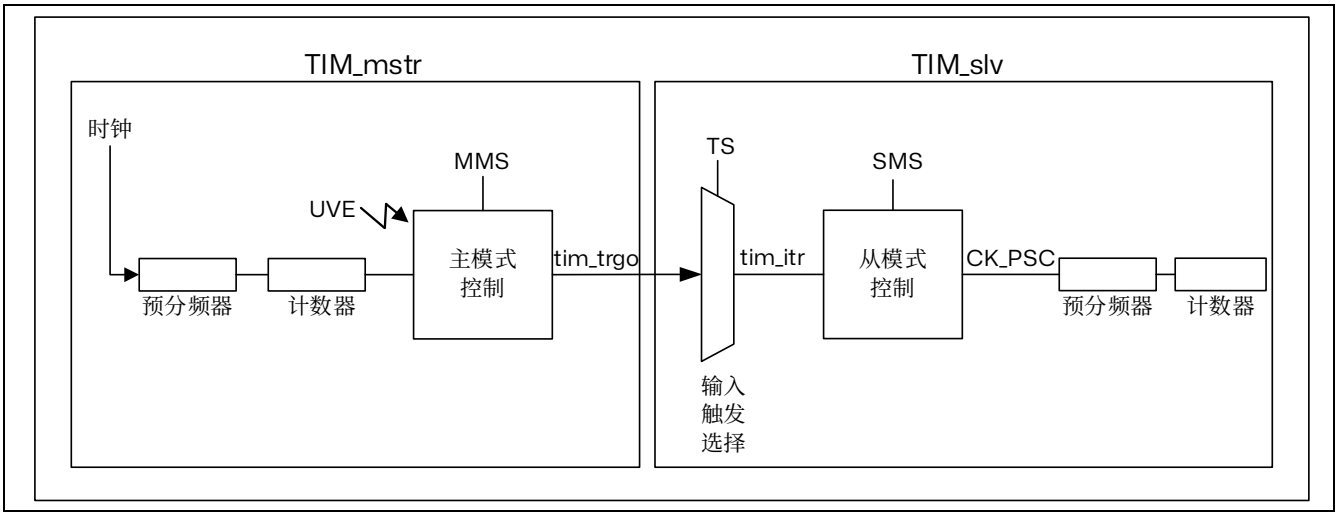



图 23.75 主/从定时器的例子

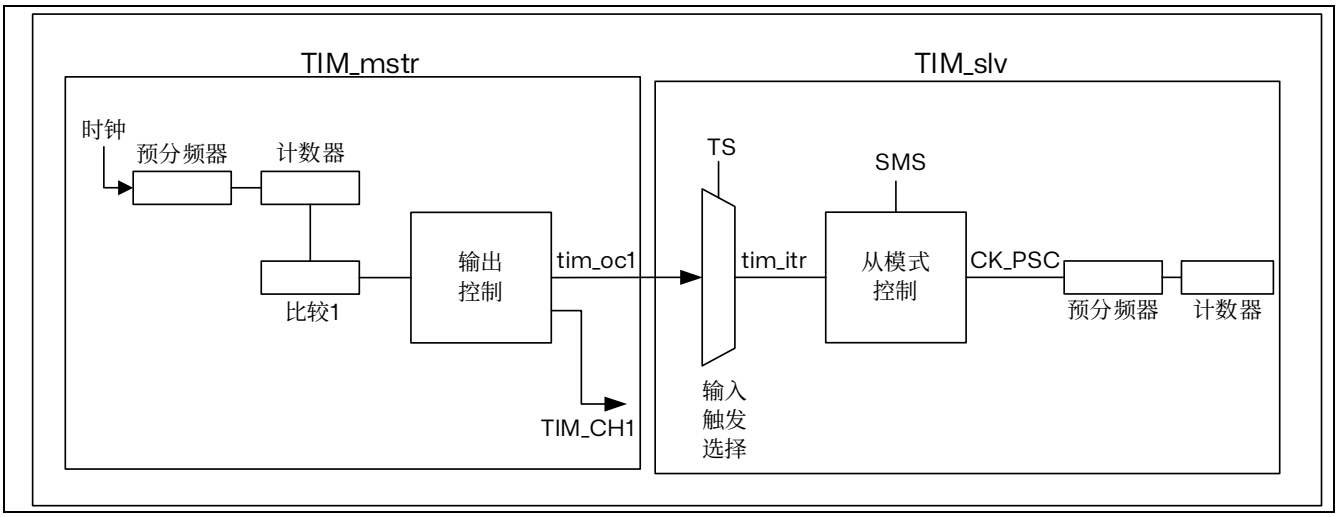



图 23.76 仅定时器通道 1 主/从连接示例

*注：只有一个通道的计时器（参见上图）不具有主模式。然而，*tim_oc1 *输出信号可以作为从属定时器的触发信号（参见*TIMx *内部触发连接表：*TIM2/TIM3/TIM4/TIM5 *引脚和内部信号）。*
*必须将*tim_oc1 *信号脉冲宽度编程为至少为目的地定时器的*2 *个时钟周期，以确保从属定时器检测到触发信号。例如，如果目的地定时器*tim_ker_ck *时钟比源定时器慢*4 *倍，则*OC1 *脉冲宽度必须是*8 *个时钟周期。*

**使用一个定时器作为另一个定时器的预分频器**

例如，可以将 TIM_mstr 配置为 TIM_slv 的预分频器。请参见上图。为此：

1. 将 TIM_mstr 配置为主模式，以便每次发生更新事件 UEV 时都输出一个周期性触发信号。如果

在 TIM_mstr_CR2 寄存器中写入 MMS=010，则每次生成更新事件时，tim_trgo 都会输出一个上升沿。

2. 要将 TIM_mstr 的 tim_trgo 输出连接到 TIM_slv，必须将 TIM_slv 配置为从模式，使用 ITR0 作为内部触发。通过 TIM_slv_SMCR 寄存器中的 TS 位（写入 TS=000）可对此进行选择。
3. 然后将从模式控制器设为外部时钟模式 1（在 TIM_slv_SMCR 寄存器中写入 SMS=111）。这样一来，TIM_slv 的时钟将由 TIM_mstr 周期性触发信号的上升沿（与 TIM_mstr 的计数器上溢对应）提供。
4. 最后必须通过将这两个定时器的相应 CEN 位（TIMx_CR1 寄存器）置 1 同时使能二者。

*注：如果选择* TIM_mstr *的* tim_ocx *信号作为触发输出（*MMS=1xx*），该信号的上升沿将用于驱动* TIM_slv *的计数器。*

**使用一个定时器使能另一个定时器**

本例中通过 TIM_mstr 的输出比较 1 来使能 TIM_slv。相关连接图，参考下图。仅当 TIM_mstr 的 tim_oc1ref 为高电平时，TIM_slv 才根据分频后的内部时钟进行计数。两个计数器的时钟频率都基于 tim_ker_ck 通过预分频器执行 3 分频（$f_{tim_cnt_ck} = f_{tim_ker_ck}/3$）

1. 将 TIM_mstr 配置为主模式，发送其输出比较 1 参考信号（tim_oc1ref）作为触发输出（TIM_mstr_CR2 寄存器中的 MMS=100）。
2. 配置 TIM_mstr 的 tim_oc1ref 波形（TIM_mstr_CCMR1 寄存器）。
3. 配置 TIM_slv 以接收来自 TIM_mstr 的输入触发（TIM_slv_SMCR 寄存器中的 TS=00010）。
4. 将 TIM_slv 配置为门控模式（TIM_slv_SMCR 寄存器中的 SMS=101）。
5. 通过向 CEN 位（TIM_slv_CR1 寄存器）写入“1”使能 TIM_slv。
6. 通过向 CEN 位（TIM_mstr_CR1 寄存器）写入“1”启动 TIM_mstr。

*注：计数器* 2 *的时钟与计数器* 1 *不同步，此模式仅影响* TIM_slv *的计数器使能信号。*

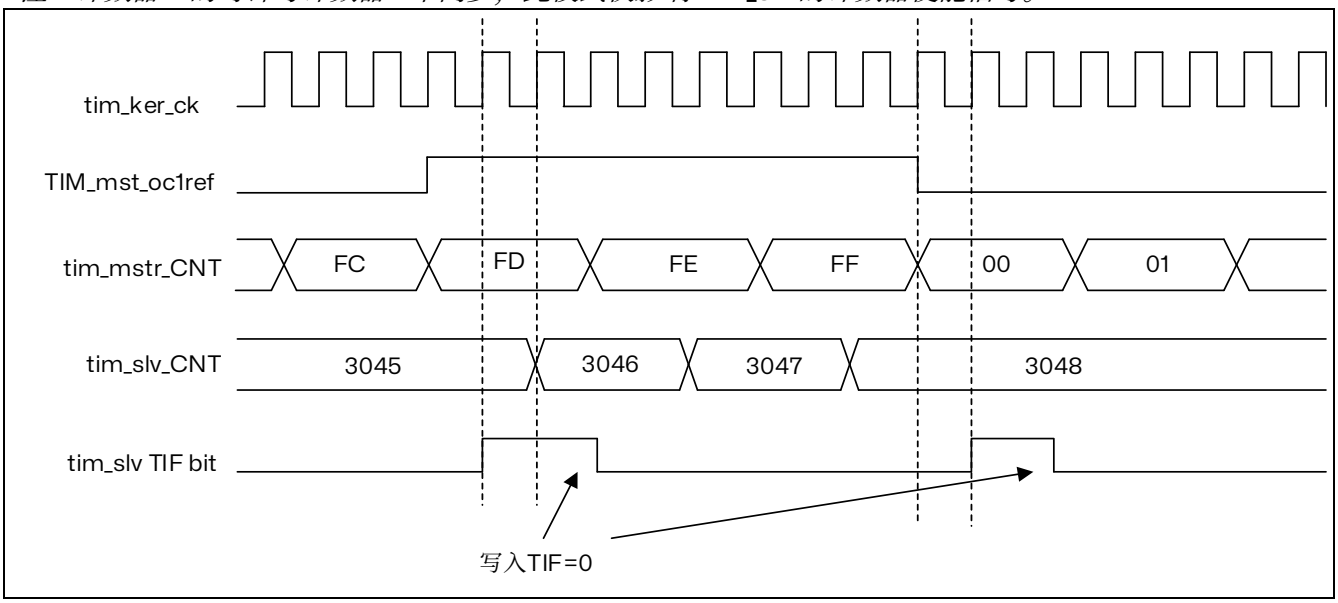



图 23.77 TIM_mstr 的 tim_oc1ref 控制 TIM_slv

在上图示例中，TIM_slv 的计数器和预分频器在启动前未进行初始化。因此从各自的当前值开始计数。启动 TIM_mstr 之前，通过复位这两个定时器可以从指定值开始计数。这样便可以在定时器计数器中写入所需的任意值。两个定时器都可通过软件使用 TIMx_EGR 寄存器中的 UG 位轻松复位。

在下一示例中，TIM_mstr 与 TIM_slv 同步。TIM_mstr 为主模式，从 0 开始计数。TIM_slv 为从模

式，从 0xE7 开始计数。两个定时器的预分频比相同。在 TIM_mstr_CR1 寄存器中通过向 CEN 位写入“0”来禁止 TIM_mstr 时，TIM_slv 将停止：

1. 将 TIM_mstr 配置为主模式，发送其输出比较 1 参考信号（tim_oc1ref）作为触发输出（TIM_mstr_CR2 寄存器中的 MMS=100）。
2. 配置 TIM_mstr 的 tim_oc1ref（TIM_mstr_CCMR1 寄存器）。
3. 配置 TIM_slv 以接收来自 TIM_mstr 的输入触发（TIM_slv_SMCR 寄存器中的 TS=00010）。
4. 将 TIM_slv 配置为门控模式（TIM_slv_SMCR 寄存器中的 SMS=101）。
5. 通过向 UG 位（TIM_mstr_EGR 寄存器）写入“1”复位 TIM_mstr。
6. 通过向 UG 位（TIM_slv_EGR 寄存器）写入“1”复位 TIM_slv。
7. 通过在 TIM_slv 的计数器（TIM_slv_CNT）中写入“0xE7”使 TIM_slv 初始化为 0xE7。
8. 通过向 CEN 位（TIM_slv_CR1 寄存器）写入“1”使能 TIM_slv。
9. 通过向 CEN 位（TIM_mstr_CR1 寄存器）写入“1”启动 TIM_mstr。
10. 通过向 CEN 位（TIM_mstr_CR1 寄存器）写入“0”停止 TIM_mstr。

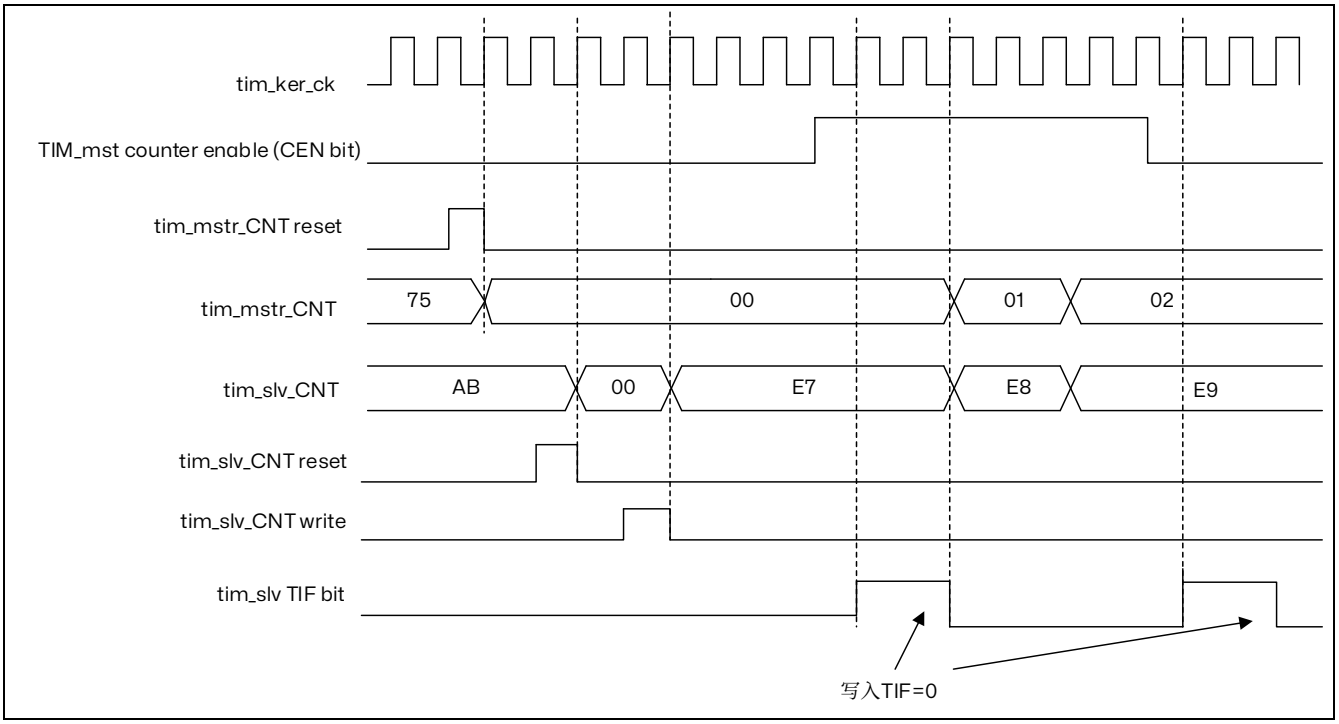



图 23.78 通过使能 TIM_mstr 可以控制 TIM_slv

**使用一个定时器去启动另一个定时器**

本例中使用 TIM_mstr 的更新事件使能 TIM_slv。相关连接图，请参见下图。只要 TIM_mstr 生成更新事件，TIM_slv 便根据分频后的内部时钟从当前值（可以不为 0）开始计数。TIM_slv 收到触发信号时，其 CEN 位自动置 1，并且计数器开始计数，直到向 TIM_slv_CR1 寄存器的 CEN 位写入“0”后停止计数。两个计数器的时钟频率都基于 CK_INT 通过预分频器执行 3 分频（$f_{tim_cnt_ck} = f_{tim_ker_ck}/3$）。

1. 将 TIM_mstr 配置为主模式，发送其更新事件（UEV）作为触发输出（TIM_mstr_CR2 寄存器中的 MMS=010）。
2. 配置 TIM_mstr 的周期（TIM_mstr_ARR 寄存器）。
3. 配置 TIM_slv 以接收来自 TIM_mstr 的输入触发（TIM_slv_SMCR 寄存器中的 TS=00010）。
4. 将 TIM_slv 配置为触发模式（TIM_slv_SMCR 寄存器中的 SMS=110）。

5. 通过向 CEN 位（TIM_mstr_CR1 寄存器）写入“1”启动 TIM_mstr。

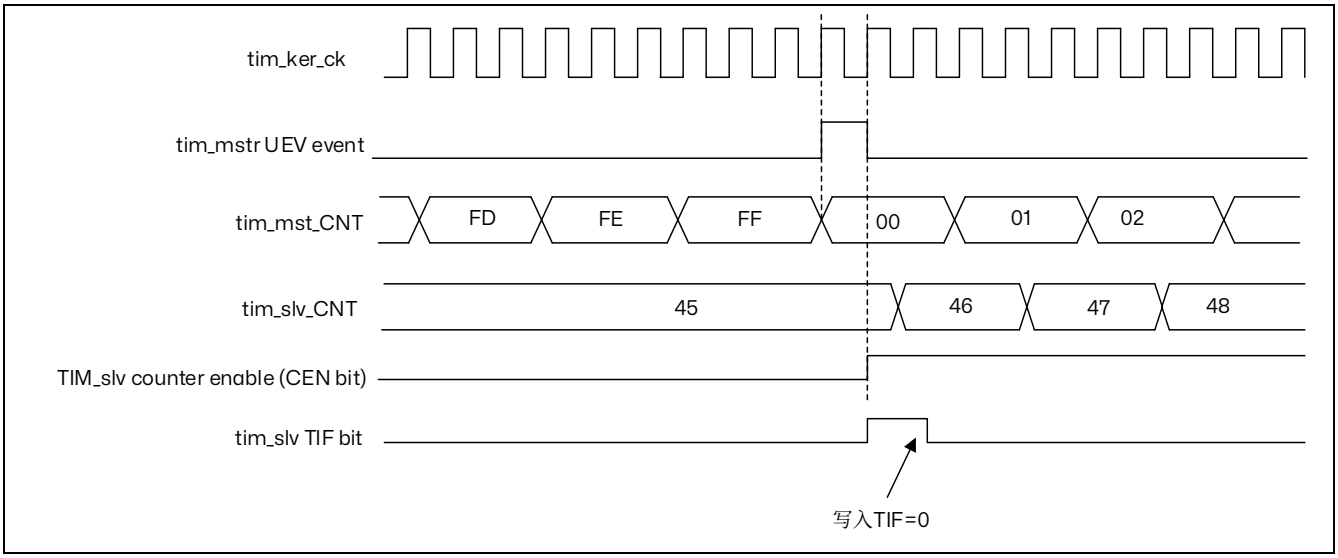



图 23.79 使用 TIM_mstr 的更新触发 TIM_slv

如上述示例所示，用户可以在开始计数之前初始化两个计数器。下图显示了与上图具有相同配置，只不过处于触发模式（TIM_slv_SMCR 寄存器中的 SMS=110）而非门控模式的计数行为。

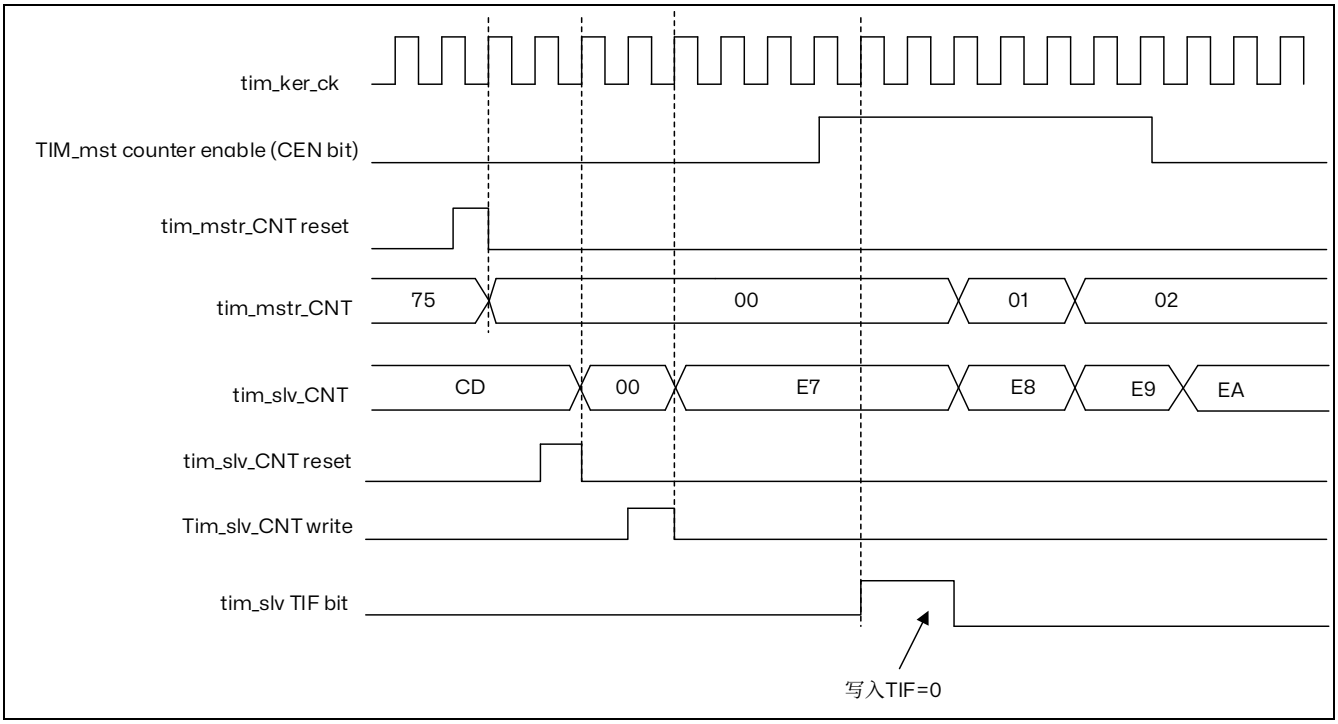



图 23.80 利用 TIM_mstr 的使能触发 TIM_slv

**使用一个外部触发同步地启动 2 个定时器**

本例中，TIM_mstr 的 tim_ti1 输入出现上升沿时使能 TIM_mstr，使能 TIM_mstr 的同时使能 TIM_slv。要确保两个计数器对齐，TIM_mstr 必须配置为主/从模式（对应的 tim_ti1 为从，对应 TIM_slv 为主）：

1. 将 TIM_mstr 配置为主模式，发送其使能信号作为触发输出（TIM_mstr_CR2 寄存器中

MMS=001）。

2. 将 TIM_mstr 配置为从模式以接收来自 tim_ti1 的输入触发（TIM_mstr_SMCR 寄存器中的 TS=00100）。
3. 将 TIM_mstr 配置为触发模式（TIM_mstrSMCR 寄存器中的 SMS=110）。
4. 通过写入 MSM=1（TIM_mstr_SMCR 寄存器）将 TIM_mstr 配置为主/从模式。
5. 配置 TIM_slv 以接收来自 TIM_mstr 的输入触发（TIM_slv_SMCR 寄存器中的 TS=00000）。
6. 将 TIM_slv 配置为触发模式（TIM_slv_SMCR 寄存器中的 SMS=110）。

当 tim_ti1（TIM_mstr）出现上升沿时，两个计数器开始根据内部时钟同步计数，并且两个 TIF 标志都置 1。

*注：本例中，两个定时器都在启动之前进行了初始化（通过将各自的UG 位置1）。两个计数器都从 0 开始计数，但可以通过对任意一个计数器寄存器（TIMx_CNT）进行写操作，在二者之间轻松插入一个偏移量。可注意到主/从模式在TIM_mstr 的CNT_EN 与CK_PSC 之间产生了延迟。*

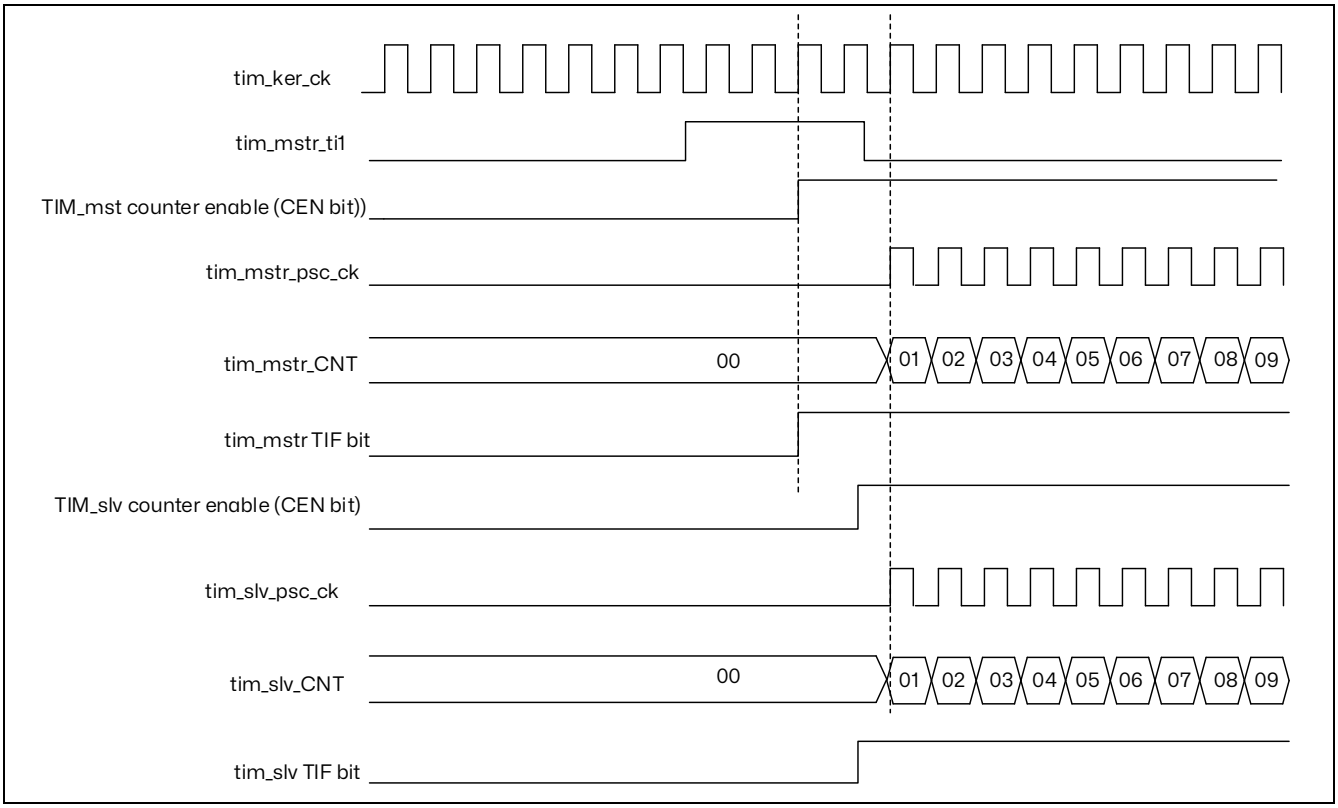



图 23.81 使用 TIM_mstr 的 tim_ti1 输入触发 TIM_mstr 和 TIM_slv

*注：接收tim_trgo 信号的从外设（定时器，ADC 等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能实时更改。*

### 23.4.24 ADC 触发

定时器可以用各种内部信号（例如重置、使能或比较事件）生成 ADC 触发事件。

*注意：接收tim_trgo 信号的从外设（例如定时器，ADC）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能实时更改。*

### 23.4.25 DMA 突发模式

TIMx 定时器具有在单个事件上生成多个 DMA 请求的能力。其主要目的是能够在不增加软件开销的情

况下多次重新编程定时器的一部分，但它也可以用于在固定间隔内连续读取多个寄存器。

DMA 控制器的目标地址是唯一的，必须指向虚拟寄存器 TIMx_DMAR。在给定的定时器事件上，定时器启动一系列 DMA 请求（突发）。对 TIMx_DMAR 寄存器的每次写入实际上都被重定向到定时器寄存器之一。

TIMx_DCR 寄存器中的 DBL[4:0]位设置了 DMA 突发长度。当对 TIMx_DMAR 地址进行读取或写入访问时，定时器会识别出突发传输，即传输的数量（以半字或字节为单位）。

TIMx_DCR 寄存器中的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址（当通过 TIMx_DMAR 地址进行读取/写入访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始定义的偏移量：

示例：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发模式用于在更新事件发生时更新 CCRx 寄存器（x = 2，3，4）的内容，其中 DMA 将半字传输到 CCRx 寄存器中。

这是通过以下步骤完成的：

6. 配置相应的 DMA 通道如下：
   - 无滤波器 DMA 通道外设地址是 DMAR 寄存器的地址。
   - DMA 通道内存地址是 RAM 中缓冲区的地址，其中包含要通过 DMA 传输到 CCRx 寄存器的数据。
   - 要传输的数据数量 = 3（见下文注释）。
   - 循环模式禁用。
7. 通过配置 DBA 和 DBL 位域来配置 DCR 寄存器，如下所示：
   DBL = 3 次传输，DBA = 0xE。
8. 使能 TIMx 更新 DMA 请求（在 DIER 寄存器中设置 UDE 位）。
9. 使能 TIMx
10. 使能 DMA 通道

此示例适用于每个 CCRx 寄存器都需要更新一次的情况。例如，如果要更新每个 CCRx 寄存器两次，则要传输的数据数量应为 6。让我们以一个在 RAM 中包含 data1，data2，data3，data4，data5 和 data6 的缓冲区为例。数据按照以下方式传输到 CCRx 寄存器：在第一个更新 DMA 请求时，data1 传输到 CCR2，data2 传输到 CCR3，data3 传输到 CCR4；在第二个更新 DMA 请求时，data4 传输到 CCR2，data5 传输到 CCR3，data6 传输到 CCR4。

*注：可以将空值写入保留的寄存器。*

### 23.4.26 TIM2/TIM3/TIM4/TIM5 DMA 请求

TIM2/TIM3/TIM4/TIM5 可以生成 DMA 请求，如下表所示。

表 23.15 DMA 请求

| DMA 请求信号 | DMA 缩写 | DMA 请求 | 使能控制位 |
|---|---|---|---|
| tim_upd_dma | TIM_UP | 更新 | UDE |
| tim_cc1_dma | TIM_CH1 | 捕获/比较 1 | CC1DE |
| tim_cc2_dma | TIM_CH2 | 捕获/比较 2 | CC2DE |
| tim_cc3_dma | TIM_CH3 | 捕获/比较 3 | CC3DE |

| DMA 请求信号 | DMA 缩写 | DMA 请求 | 使能控制位 |
|---|---|---|---|
| tim_cc4_dma | TIM_CH4 | 捕获/比较 4 | CC4DE |
| tim_trg_dma | TIM_TRIG | 触发 | TDE |

*注：某些定时器的DMA 请求可能未连接到DMA 控制器。请参考DMA 部分以获取更多详细信息。*

### 23.4.27 调试模式

当微控制器进入调试模式（内核停止）时，TIMx 计数器可以继续正常工作或停止。
在调试模式下，每个定时器的行为可以通过 Debug 支持（DBG）模块的专用配置位进行编程。
有关更多详细信息，请参阅调试部分。

## 23.5 TIM2/TIM3/TIM4/TIM5 低功耗模式

表 23.16 低功耗模式对 TIM2/TIM3/TIM4/TIM5 的影响

| 模式 | 描述 |
|---|---|
| 睡眠 | 无影响，外设是运行的。中断会导致设备退出睡眠模式。 |
| 停机 | 定时器操作被停止并保留其寄存器内容。无法生成中断。 |
| 待机 | 定时器被断电并且退出待机模式后必须重新初始化。 |

## 23.6 TIM2/TIM3/TIM4/TIM5 中断

TIM2/TIM3/TIM4/TIM5 可以生成多个中断，如下表所示。

表 23.17 中断请求

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机和待机退出 |
|---|---|---|---|---|---|---|
| TIM_UP | 更新 | UIF | UIE | UIF 写 0 | 是 | 否 |
| TIM_CC | 捕获/比较 1 | CC1IF | CC1IE | CC1IF 写 0 | 是 | 否 |
| | 捕获/比较 2 | CC2IF | CC2IE | CC2IF 写 0 | 是 | 否 |
| | 捕获/比较 3 | CC3IF | CC3IE | CC3IF 写 0 | 是 | 否 |
| | 捕获/比较 4 | CC4IF | CC4IE | CC4IF 写 0 | 是 | 否 |
| TIM_TRG | 触发 | TIF | TIE | TIF 写 0 | 是 | 否 |
| TIM_DIR_IDX | 索引 | IDXF | IDXIE | IDXF 写 0 | 是 | 否 |
| | 方向 | DIRF | DIRIE | DIRF 写 0 | 是 | 否 |
| TIM_IERR | 索引错误 | IERRF | IERRIE | IERRF 写 0 | 是 | 否 |
| TIM_TER | 转换错误 | TERRF | TERRIE | TERRF 写 0 | 是 | 否 |

## 23.7 TIM2/TIM3/TIM4/TIM5 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。
外设寄存器可以通过半字（16 位）或字（32 位）进行访问。

### 23.7.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 2~5）

偏移地址：0x000
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | 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 15:13 保留，必须保持复位值。

Bit 12 **DITHEN**：使能抖动（Dithering enable）

0：抖动禁止

1：抖动使能

*注：*DITHEN *位只能在*CEN *位复位时修改*

Bit 11 **UIFREMAP**：UIF 状态位重映射（UIF status bit remapping）

0：无重映射。UIF 状态位未复制到 TIMx_CNT 寄存器位 31

1：重映射使能。UIF 状态位复制到 TIMx_CNT 寄存器位 31

Bit 10 保留，必须保持复位值。

Bit 9:8 **CKD[1:0]**：时钟分频（Clock division）

此位域指示定时器时钟（tim_ker_ck）频率以及数字滤波器（tim_etr_in，tim_tix）所使用的采样时钟（$t_{DTS}$）之间的分频比

00：$t_{DTS} = t_{tim_ker_ck}$

01：$t_{DTS} = 2 \times t_{tim_ker_ck}$

10：$t_{DTS} = 4 \times t_{tim_ker_ck}$

11：保留，不要设置成此值

Bit 7 **ARPE**：自动重载预装载使能（Auto-reload preload enable）

0：TIMx_ARR 寄存器不进行缓冲；

1：TIMx_ARR 寄存器进行缓冲。

Bit 6:5 **CMS[1:0]**：中心对齐模式选择（Center-aligned mode selection）

00：边沿对齐模式。计数器根据方向位（DIR）递增计数或递减计数。

01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

*注：只要计数器处于使能状态* (CEN=1)，*就不得从边沿对齐模式切换为中心对齐模式。*

Bit 4 **DIR**：方向（Direction）

0：计数器递增计数；

1：计数器递减计数。

*注：当定时器配置为中心对齐模式或编码器模式时，该位为只读状态。*

Bit 3 **OPM**：单脉冲模式（One pulse mode）

0：计数器在发生更新事件时不会停止计数；

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

Bit 2 **URS**：更新请求源（Update request source）

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

Bit 1 **UDIS**：更新禁止（Update disable）

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新（UEV）事件可通过以下事件之一生成：

- 计数器上出/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

Bit 0 **CEN**：计数器使能（Counter enable）

0：禁止计数器；

1：使能计数器。

*注：只有事先通过软件将*CEN *位置*1*，才可以使用外部时钟、门控模式和编码器模式。而触发模式可通过硬件自动将*CEN *位置*1*。*

### 23.7.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 2~5）

偏移地址：0x004

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | MMS[3] | Reserved | | | | | | | | |
| | | | | | | rw | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TI1S | MMS[2:0] | | | CCDS | Reserved | | |
| | | | | | | | | rw | rw | rw | rw | rw | | | |

Bit 31:26 保留，必须保持复位值。

Bit 24:8 保留，必须保持复位值。

Bit 7 **TI1S**：tim_ti1 选择（tim_ti1 selection）

0：tim_ti1_in[15:0]引脚连接到 tim_ti1 输入

1：tim_ti1_in[15:0]，tim_ti2_in[15:0]和 tim_ti3_in[15:0]多路复用器输出被异或并连接至 tim_ti1 输入

Bit 25, 6:4 **MMS[3:0]：**主模式选择（Master mode selection）

这些位可选择主模式下将要发送到从定时器以实现同步的信息(tim_trgo)。这些位的组合如下：

0000：**复位** – TIMx_EGR 寄存器中的 UG 位用作触发输出（tim_trgo)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。

0001：**使能** – 计数器使能信号 CNT_EN 用作触发输出（tim_trgo)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由 CEN 控制位产生。当配置为门控模式时，也可由触发输入产生。当计数器使能信号由触发输入控制时，tim_trgo 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明)。

0010：**更新** – 选择更新事件作为触发输出（tim_trgo)。例如，主定时器可用作从定时器的预分频器。

0011：**比较脉冲** – 一旦发生输入捕获或比较匹配事件，当 CC1IF 被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(tim_trgo)。

0100：**比较** – tim_oc1refc 信号用作触发输出（tim_trgo）

0101：**比较** – tim_oc2refc 信号用作触发输出（tim_trgo）

0110：**比较** – tim_oc3refc 信号用作触发输出（tim_trgo）

0111：**比较** – tim_oc4refc 信号用作触发输出（tim_trgo）

1000：**编码器时钟输出** – 编码器时钟信号用作出发输出（tim_trgo)。此编码对以下 SMS[3:0]值有效：0001，0010，0011，1010，1011，1100，1101，1110，1111。任何其他 SMS[3:0]编码都是不允许的，可能会导致意外行为。

其他编码保留

*注：在从主定时器接收事件之前，必须使能从机定时器或* ADC *的时钟，并且在从主定时器接收触发器时不能随意更改。*

*注：若选* tim_ocxrefc *为* tim_trgo *输出，当* tim_ocref_clr_int *清零信号生效时，* tim_ocxrefc *信号将保持低电平，* tim_trgo *也不会有触发输出，直到发生下一更新事件（*UEV*）（特别注意，当* TIMx_RCR>0 *时，只有重复计数器达到零才会生成更新事件，波形才会恢复)。*

Bit 3 **CCDS**：捕获/比较 DMA 选择（Capture/compare DMA selection）

0：发生 CCx 事件时发送 CCx DMA 请求

1：发生更新事件时发送 CCx DMA 请求

Bit 2:0 保留，必须保持复位值。

### 23.7.3 TIMx 从模式控制寄存器（TIMx_SMCR）（x = 2~5）

偏移地址：0x008

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | SMSPS | SMSPE | Reserved | | TS[4:3] | | Reserved | | | 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 31:26 保留，必须保持复位值。

Bit 25 **SMSPS**：SMS 预加载源（SMS preload source）

此位选择触发 SMS[3:0]位域从预加载到活动的事件来源：

0：由定时器的更新事件触发

1：由索引事件触发

Bit 24 **SMSPE**：SMS 预加载使能（SMS preload enable）

此位选择是否使能 SMS[3:0]位域的预加载：

0：SMS[3:0]位域不进行预加载

1：使能 SMS[3:0]预加载

Bit 23:22 保留，必须保持复位值。

Bit 19:17 保留，必须保持复位值。

Bit 15 **ETP**：外部触发极性（External trigger polarity）

此位可选择将 tim_etr_in 还是 tim_etr_in 的反相用于触发操作

0：tim_etr_in 未反相，高电平或上升沿有效。

1：tim_etr_in 反相，低电平或下降沿有效。

Bit 14 **ECE**：外部时钟使能位（External clock enable）

此位可使能外部时钟模式 2。

0：禁止外部时钟模式 2

1：使能外部时钟模式 2。计数器时钟由 tim_etrf 信号的任意有效边沿提供。

*注：将* ECE *位置*1 *与选择外部时钟模式*1 *并将* tim_trgi *连接到* tim_etrf *（*SMS=111 *且* TS=00111*）具有相同效果。*

*外部时钟模式*2 *可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下* tim_trgi *不得连接* tim_etrf *（*TS *位不得为*00111*）。*

*如果同时使能外部时钟模式*1 *和外部时钟模式*2*，则外部时钟输入为* tim_etrf*。*

Bit 13:12 **ETPS[1:0]**：外部触发预分频器（External trigger prescaler）

外部触发信号 tim_etrp 频率不得超过 TIMxCLK 频率的 1/4。可通过使能预分频器来降低 tim_etrp 频率。这种方法在 tim_etr_in 输入快速外部时钟时非常有用。

00：预分频器关闭

01：2 分频 tim_etrp 频率

10：4 分频 tim_etrp 频率

11：8 分频 tim_etrp 频率

Bit 11:8 **ETF[3:0]**：外部触发滤波器（External trigger filter）

此位域可定义 tim_etrp 信号的采样频率和适用于 tim_etrp 的数字滤波时间。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{CK_INT}$，N=2。

0010：$f_{SAMPLING}=f_{CK_INT}$，N=4。

0011：$f_{SAMPLING}=f_{CK_INT}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 7 **MSM：**主/从模式（Master/slave mode）

0：不执行任何操作；

1：当前定时器的触发输入事件(tim_trgi)的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 tim_trgo）。此设置适用于由单个外部事件对多个定时器进行同步的情况。

Bit 21,20,6,5,4 **TS[4:0]：**触发选择（Trigger selection）

此位域可选择将要用于同步计数器的触发输入。

00000：内部触发 0（tim_itr0）

00001：内部触发 1（tim_itr1）

00010：内部触发 2（tim_itr2）

00011：内部触发 3（tim_itr3）

00100：tim_ti1 边沿检测器（tim_ti1f_ed）

00101：滤波后的定时器输入 1（tim_ti1fp1）

00110：滤波后的定时器输入 2（tim_ti2fp2）

00111：外部触发输入（tim_etrf）

01000：内部触发器 4 (tim_itr4)

01001：内部触发器 5 (tim_itr5)

01010：内部触发器 6 (tim_itr6)

01011：内部触发器 7 (tim_itr7)

01100：内部触发器 8 (tim_itr8)

01101：内部触发器 9 (tim_itr9)

01110：内部触发器 10 (tim_itr10)

01111：内部触发器 11 (tim_itr11)

10000：内部触发器 12 (tim_itr12)

10001：内部触发器 13 (tim_itr13)

10010：内部触发器 14 (tim_itr14)

10011：内部触发器 15 (tim_itr15)

其他：保留

有关定时器 ITRx 含义的详细信息，请参见章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号

*注：这些位只能在未使用的情况下（例如，*SMS=000 *时）进行更改，以避免转换时出现错误的边沿检测。*

Bit 3 **OCCS：**OCREF 清除源选择（OCREF clear selection）

此位用于选择 OCREF 清除源。

0：tim_ocref_clr_int 信号连接到 tim_ocref_clr 输入

1：tim_ocref_clr_int 信号连接到 tim_etrf

*注：如果不支持*OCREF *清除选择功能，则此位被保留并由硬件强制设置为“*0*”。参考章节：*TIM2/TIM3/TIM4/TIM5 *实现。*

Bit 16, 2,1,0 **SMS[3:0]**：从模式选择（Slave mode selection）

当选择了外部信号，触发信号（tim_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：**编码器模式 1** – 计数器根据 tim_ti1fp1 电平在 tim_ti2fp2 边沿递增/递减计数。

0010：**编码器模式 2** – 计数器根据 tim_ti2fp2 电平在 tim_ti1fp1 边沿递增/递减计数。

0011：**编码器模式 3** – 计数器在 tim_ti1fp1 和 tim_ti2fp2 的边沿计数，计数的方向取决于另外一个信号的电平。

0100：**复位模式** – 选中的触发输入（tim_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（tim_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 tim_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（tim_trgi）的上升沿驱动计数器。

1000：**组合复位+触发模式** - 选定触发器输入（tim_trgi）的上升沿会重新初始化计数器，生成寄存器的更新并启动计数器。

1001：**组合门控+复位模式** - 当触发器输入（tim_trgi）为高电平时计数器时钟使能。一旦触发器变为低电平计数器将停止并重置。同时控制计数器的启动和停止。

1010：**编码器模式**：时钟加方向，x2 模式。

1011：**编码器模式**：时钟加方向，x1 模式，CC2P 设置 tim_ti2fp2 边沿敏感度。

1100：**编码器模式**：方向时钟，x2 模式。

1101：**编码器模式**：方向时钟，x1 模式，CC1P 和 CC2P 设置 tim_ti1fp1 和 tim_ti2fp2 边沿敏感度.

1110：**四象限编码器模式**：x1 模式，仅计算 tim_ti1fp1 边沿，CC1P 设置边沿敏感度。

1111：**四象限编码器模式**：x1 模式，仅计算 tim_ti2fp2 边沿，CC2P 设置边沿敏感度。

*注：如果选择* tim_ti1f_ed *作为触发输入（*TS=00100*），则不得使用门控模式。实际上，*tim_ti1f_ed *为* TI1F *上的每个过渡输出* 1 *个脉冲，而门控模式检查触发信号的电平。*

*注：接收* tim_trgo *或* tim_trgo2 *信号的从机外设（定时器、*ADC *等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能随意更改。*

### 23.7.4 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 2~5）

偏移地址：0x00C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TERRIE | IERRIE | DIRIE | IDXIE | Reserved | | | |
| | | | | | | | | 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 | Res | TIE | Res | CC4IE | CC3IE | CC2IE | CC1IE | UIE |
| | rw | | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw |

Bit 31:24 保留，必须保持复位值。

Bit 23 **TERRIE**：转换错误中断使能（Transition error interrupt enable）

0：禁止转换错误中断；

1：使能转换错误中断。

Bit 22 **IERRIE**：索引错误中断使能（Index error interrupt enable）

0：禁止索引错误中断；

1：使能索引错误中断。

Bit 21 **DIRIE**：方向改变中断使能（Direction change interrupt enable）

0：禁止方向改变中断；

1：使能方向改变中断。

Bit 20 **IDXIE**：索引中断使能（Index interrupt enable）

0：禁止索引中断；

1：使能索引中断。

Bit 19:15 保留，必须保持复位值。

Bit 14 **TDE**：触发 DMA 请求使能（Trigger DMA request enable）

0：禁止触发 DMA 请求；

1：使能触发 DMA 请求。

Bit 13 保留，必须保持复位值。

Bit 12 **CC4DE**：捕获/比较 4 DMA 请求使能（Capture/Compare 4 DMA request enable）

0：禁止 CC4 DMA 请求；

1：使能 CC4 DMA 请求。

Bit 11 **CC3DE**：捕获/比较 3 DMA 请求使能（Capture/Compare 3 DMA request enable）

0：禁止 CC3 DMA 请求；

1：使能 CC3 DMA 请求。

Bit 10 **CC2DE**：捕获/比较 2 DMA 请求使能（Capture/Compare 2 DMA request enable）

0：禁止 CC2 DMA 请求；

1：使能 CC2 DMA 请求。

Bit 9 **CC1DE**：捕获/比较 1 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止 CC1 DMA 请求；

1：使能 CC1 DMA 请求。

Bit 8 **UDE**：更新 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止更新 DMA 请求；

1：使能更新 DMA 请求。

Bit 7 保留，必须保持复位值。

Bit 6 **TIE**：触发信号（TGRI）中断使能（Trigger interrupt enable）

0：禁止触发信号（TGRI）中断；

1：使能触发信号（TGRI）中断。

Bit 5 保留，必须保持复位值。

Bit 4 **CC4IE**：捕获/比较 4 中断使能（Capture/Compare 4 interrupt enable）

0：禁止 CC4 中断；

1：使能 CC4 中断。

Bit 3 **CC3IE**：捕获/比较 3 中断使能（Capture/Compare 3 interrupt enable）

0：禁止 CC3 中断；

1：使能 CC3 中断。

Bit 2 **CC2IE**：捕获/比较 2 中断使能（Capture/Compare 2 interrupt enable）

0：禁止 CC2 中断；

1：使能 CC2 中断。

Bit 1 **CC1IE**：捕获/比较 1 中断使能（Capture/Compare 1 interrupt enable）

0：禁止 CC1 中断；

1：使能 CC1 中断。

Bit 0 **UIE**：更新中断使能（Update interrupt enable）

0：禁止更新中断；

1：使能更新中断。

### 23.7.5 TIMx 状态寄存器（TIMx_SR）（x = 2~5）

偏移地址：0x010

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TERRF | IERRF | DIRF | IDXF | Reserved | | | |
| | | | | | | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CC4OF | CC3OF | CC2OF | CC1OF | Reserved | | 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 31:24 保留，必须保持复位值。

Bit 23 **TERRF**：转换错误中断标志（Transition error interrupt flag）

当在编码器模式下检测到转换错误时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未检测到编码器转换错误

1：已检测到编码器转换错误

Bit 22 **IERRF**：索引错误中断标志（Index error interrupt flag）

当检测到索引错误时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未检测到索引错误

1：已检测到索引错误

Bit 21 **DIRF**：方向改变中断标志（Direction change interrupt flag）

当编码器模式中的方向发生变化时（TIMx_CR 中的 DIR 位值正在改变），硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：方向未改变

1：方向已改变

Bit 20 **IDXF**：索引中断标志（Index interrupt flag）

当检测到索引事件时，硬件会设置此标志当软件将其写入“0”时，该标志将被清除

0：未发生索引事件

1：已发生索引事件

Bit 19:13 保留，必须保持复位值。

Bit 12 **CC4OF**：捕获/比较 4 重复捕获标志

参考 CC1OF 的描述

Bit 11 **CC3OF**：捕获/比较 3 重复捕获标志

参考 CC1OF 的描述

Bit 10 **CC2OF**：捕获/比较 2 重复捕获标志

参考 CC1OF 的描述

Bit 9 **CC1OF**：捕获/比较 1 重复捕获标志

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获

1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

Bit 8:7 保留，必须保持复位值。

Bit 6 **TIF**：触发中断标志（Trigger interrupt flag）

在除门控模式以外的所有模式下，当使能从模式控制器后在 tim_trgi 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。

0：未发生触发事件。

1：触发中断挂起。

Bit 5 保留，必须保持复位值。

Bit 4 **CC4IF**：捕获/比较 4 中断标志（Capture/Compare 4 interrupt flag）

参考 CC1IF 描述。

Bit 3 **CC3IF**：捕获/比较 3 中断标志（Capture/Compare 3 interrupt flag）

参考 CC1IF 描述。

Bit 2 **CC2IF**：捕获/比较 2 中断标志（Capture/Compare 2 interrupt flag）

参考 CC1IF 描述。

Bit 1 **CC1IF**：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag）

此标志由硬件设置，它可以通过软件（输入捕获或输出比较模式）或读取 TIMx_CCR1 寄存器（仅输入捕获模式）来清除。

0：无比较匹配/无输入捕获发生

1：发生比较匹配或输入捕获

**如果通道 CC1 配置为输出：**当计数器 TIMx_CNT 的内容与 TIMx_CCR1 寄存器内容匹配时，会设置此标志。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。在中心对齐模式下有 3 种可能的标志设置选项，请参考 TIMx_CR1 寄存器中的 CMS 位以获取完整描述。

**如果通道 CC1 配置为输入：**当在 TIMx_CCR1 寄存器中捕获到计数器值时（在 TIMx_CCER 中定义的 CC1P 和 CC1NP 位设置所确定的 IC1 上检测到边沿时），会设置此位。

Bit 0 **UIF**：更新中断标志（Update interrupt flag）

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– TIMx_CR1 寄存器中的 UDIS=0，并且在上溢或下溢时。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（参考同步控制寄存器描述）。

### 23.7.6 TIMx 事件产生寄存器（TIMx_EGR）（x = 2~5）

偏移地址：0x14

复位：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | TG | Res | CC4G | CC3G | CC2G | CC1G | UG |
| | | | | | | | | | w | | w | w | w | w | w |

Bit 15:7 保留，必须保持复位值。

Bit 6 **TG**：生成触发信号（Trigger generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

Bit 5 保留，必须保持复位值。

Bit 4 **CC4G**：捕获/比较 4 生成（Capture/Compare 4 generation）

参考 CC1G 描述。

Bit 3 **CC3G**：捕获/比较 3 生成（Capture/Compare 3 generation）

参考 CC1G 描述。

Bit 2 **CC2G**：捕获/比较 2 生成（Capture/Compare 2 generation）

参考 CC1G 描述。

Bit 1 **CC1G**：捕获/比较 1 生成（Capture/Compare 1 generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：通道 1 上生成捕获/比较事件：

**若通道 CC1 配置为输出：**

使能时，CC1IF 标志置 1 并发送相应的中断。

**如果通道 CC1 配置为输入：**

TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

Bit 0 **UG**：更新生成（Update generation）

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作；

1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。如果选择中心对齐模式或 DIR=0（递增计数），计数器将清零；如果 DIR=1（递减计数），计数器将使用自动重载值（TIMx_ARR）。

### 23.7.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 2~5）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IC2F[3:0] | | | | IC2PSC[1:0] | | CC2S[1:0] | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:16 保留，必须保持复位值。

Bit 15:12 **IC2F[3:0]**：输入捕获 2 滤波器（Input capture 2 filter）

Bit 11:10 **IC2PSC[1:0]**：输入/捕获 2 预分频器（input capture 2 prescaler）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择 （Capture/compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道被配置为输出；

01：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti2 上；

10：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti1 上；

11：CC2 通道被配置为输入，tim_ic2 映射在 tim_trc 上。此模此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIMx_CCER *中的*CC2E = 0*），才可向*CC2S *位写入数据。*

Bit 7:4 **IC1F[3:0]：**输入捕获 1 滤波器（Input capture 1 filter）

此位域可定义 tim_ti1 输入的采样频率和适用于 tim_ti1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 3:2 **IC1PSC[1:0]：**输入/捕获 1 预分频器（Input capture 1 prescaler）

此位域定义 CC1 输入（tim_ic1）的预分频比。

只要 CC1E=0（TIMx_CCER 寄存器），预分频器便立即复位。

00：无预分频器，捕获输入上每检测到一个边沿便执行捕获

01：每发生 2 个事件便执行一次捕获；

10：每发生 4 个事件便执行一次捕获；

11：每发生 8 个事件便执行一次捕获。

Bit 1:0 **CC1S[1:0]：**捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道被配置为输出；

01：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti1 上；

10：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti2 上；

11：CC1 通道被配置为输入，tim_ic1 映射在 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIMx_CCER *中的* CC1E = 0*），才可向* CC1S *位写入数据。*

### 23.7.8 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 2~5）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC2M[3] | Reserved | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式**

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC2CE：**输出比较 2 清零使能（Output Compare 2 clear enable）

Bit 24, 14:12 **OC2M[2:0]：**输出比较 2 模式（Output Compare 2 mode）

参考 OC1M 描述

Bit 11 **OC2PE：**输出比较 2 预装载使能（Output Compare 2 preload enable）

Bit 10 **OC2FE：**输出比较 2 快速使能（Output Compare 2 fast enable）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择（Capture/Compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，tim_ic2 映射到 tim_ti2 上

10：CC2 通道配置为输入，tim_ic2 映射到 tim_ti1 上

11：CC2 通道配置为输入，tim_ic2 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIMx_CCER *中的*CC2E = 0*），才可向*CC2S *位写入数据*

Bit 7 **OC1CE：**输出比较 1 清零使能（Output Compare 1 clear enable）

0：tim_oc1ref 不受 tim_ocref_clr_int 输入的影响；

1：一旦在 tim_ocref_clr_int 输入上检测到高电平，tim_oc1ref 立即清零。

Bit 16, 6:4 **OC1M[3:0]：**输出比较 1 模式（Output Compare 1 mode）

这些位定义提供 tim_oc1 和 tim_oc1n 的输出参考信号 tim_oc1ref 的行为。tim_oc1ref 为高电平有效，而 tim_oc1 和 tim_oc1n 的有效电平则取决于 CC1P 位和 CC1NP 位。

0000：**冻结** - 输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

0001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为高电平。

0010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为低电平。

0011：**翻转** - TIMx_CNT=TIMx_CCR1 时，tim_oc1ref 发生翻转。

0100：**强制变为无效电平** - tim_oc1ref 强制变为低电平。

0101：**强制变为有效电平** - tim_oc1ref 强制变为高电平。

0110：**PWM 模式 1**-在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态（tim_oc1ref=0），否则为有效状态（tim_oc1ref=1）。

0111：**PWM 模式 2** - 在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

1000：**可重触发 OPM 模式 1**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于无效状态。

1001：**可重触发 OPM 模式 2**- 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 2 那样执行比较，并在下次更新时使通道再次处于无效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。

1010：保留。

1011：保留。

1100：**组合 PWM 模式 1**- tim_oc1ref 的行为与 PWM 模式 1 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑 OR。

1101：**组合 PWM 模式 2**- tim_oc1ref 的行为与 PWM 模式 2 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑 AND。

1110：**非对称 PWM 模式 1**- tim_oc1ref 的行为与 PWM 模式 1 相同。当计数器向上计数时，tim_oc1refc 输出 tim_oc1ref；当计数器向下计数时，tim_oc1refc 输出 tim_oc2ref。

1111：**非对称 PWM 模式 2**- tim_oc1ref 的行为与 PWM 模式 2 相同。当计数器向上计数时，tim_oc1refc 输出 tim_oc1ref；当计数器向下计数时，tim_oc1refc 输出 tim_oc2ref。

*注：在 PWM 模式下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，tim_ocref_clr 电平才会发生更改。*

Bit 3 **OC1PE：**输出比较 1 预装载使能（Output Compare 1 preload enable）

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

Bit 2 **OC1FE：**输出比较 1 快速使能（Output Compare 1 fast enable）

此位减少了触发事件和定时器输出上的转换之间的延迟。它必须在单脉冲模式（在 TIMx_CR1 寄存器中设置 OPM 位）中使用，以便在启动触发后尽快开始输出脉冲。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

Bit 1:0 **CC1S[1:0]：**捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，tim_ic1 映射到 tim_ti1 上

10：CC1 通道配置为输入，tim_ic1 映射到 tim_ti2 上

11：CC1 通道配置为输入，tim_ic1 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据*

### 23.7.9 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 2~5）

偏移地址：0x01C

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IC4F[3:0] | | | | IC4PSC[1:0] | | CC4S[1:0] | | IC3F[3:0] | | | | IC3PSC[1:0] | | CC3S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:16 保留，必须保持复位值。

Bit 15:12 **IC4F[3:0]：**输入捕获 4 滤波器（Input capture 4 filter）

Bit 11:10 **IC4PSC[1:0]：**输入/捕获 4 预分频器（input capture 4 prescaler）

Bit 9:8 **CC4S[1:0]：**捕获/比较 4 选择 （Capture/compare 4 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC4 通道被配置为输出；

01：CC4 通道被配置为输入，tim_ic4 映射在 tim_ti4 上；

10：CC4 通道被配置为输入，tim_ic4 映射在 tim_ti3 上；

11：CC4 通道被配置为输入，tim_ic4 映射在 tim_trc 上。此模此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向CC4S 位写入数据。*

Bit 7:4 **IC3F[3:0]：**输入捕获 3 滤波器（Input capture 3 filter）

Bit 3:2 **IC3PSC[1:0]：**输入/捕获 1 预分频器（Input capture 3 prescaler）

Bit 1:0 **CC3S[1:0]：**捕获/比较 3 选择（Capture/Compare 3 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC3 通道被配置为输出；

01：CC3 通道被配置为输入，tim_ic3 映射在 tim_ti3 上；

10：CC3 通道被配置为输入，tim_ic3 映射在 tim_ti4 上；

11：CC3 通道被配置为输入，tim_ic3 映射在 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向CC3S 位写入数据。*

### 23.7.10 TIMx 捕获/比较模式寄存器 2 [复用]（TIMx_CCMR2）（x = 2~5）

偏移地址：0x01C

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC4M[3] | Reserved | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式**

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC4CE：**输出比较 4 清零使能（Output Compare 4 clear enable）

Bit 24, 14:12 **OC4M[2:0]：**输出比较 4 模式（Output Compare 4 mode）

参考 OC1M 描述（TIMx_CCMR1 寄存器）

Bit 11 **OC4PE：**输出比较 4 预装载使能（Output Compare 4 preload enable）

Bit 10 **OC4FE：**输出比较 4 快速使能（Output Compare 4 fast enable）

Bit 9:8 **CC4S[1:0]：**捕获/比较 4 选择（Capture/Compare 4 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC4 通道配置为输出

01：CC4 通道配置为输入，tim_ic4 映射到 tim_ti4 上

10：CC4 通道配置为输入，tim_ic4 映射到 tim_ti3 上

11：CC4 通道配置为输入，tim_ic4 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据*

Bit 7 **OC3CE：**输出比较 3 清零使能（Output Compare 3 clear enable）

Bit 16, 6:4 **OC3M[3:0]：**输出比较 3 模式（Output Compare 1 mode）

参考 OC1M 描述（TIMx_CCMR1 寄存器）

Bit 3 **OC3PE：**输出比较 3 预装载使能（Output Compare 3 preload enable）

Bit 2 **OC3FE：**输出比较 3 快速使能（Output Compare 3 fast enable）

Bit 1:0 **CC3S[1:0]：**捕获/比较 3 选择（Capture/Compare 3 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC3 通道配置为输出

01：CC3 通道配置为输入，tim_ic3 映射到 tim_ti3 上

10：CC3 通道配置为输入，tim_ic3 映射到 tim_ti4 上

11：CC3 通道配置为输入，tim_ic3 映射到 tim_trc 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据*

### 23.7.11 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 2~5）

偏移地址：0x020

复位值：0x0000

| 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 15 **CC4NP**：输入/捕获 4 输出极性（Capture/Compare 4 output polarity）

参考 CC1NP 的描述。

Bit 14 保留，必须保持复位值。

Bit 13 **CC4P**：输入/捕获 4 输出极性（Capture/Compare 4 output polarity）

参考 CC1P 的描述。

Bit 12 **CC4E**：输入/捕获 4 输出使能（Capture/Compare 4 output enable）

参考 CC1E 的描述。

Bit 11 **CC3NP**：输入/捕获 3 输出极性（Capture/Compare 3 output polarity）

参考 CC1NP 的描述。

Bit 10 保留，必须保持复位值。

Bit 9 **CC3P**：输入/捕获 3 输出极性（Capture/Compare 3 output polarity）

参考 CC1P 的描述。

Bit 8 **CC3E**：输入/捕获 3 输出使能（Capture/Compare 3 output enable）

参考 CC1E 的描述。

Bit 7 **CC2NP**：输入/捕获 2 输出极性（Capture/Compare 2 output polarity）

参考 CC1NP 的描述。

Bit 6 保留，必须保持复位值。

Bit 5 **CC2P**：输入/捕获 2 输出极性（Capture/Compare 2 output polarity）

参考 CC1P 的描述。

Bit 4 **CC2E**：输入/捕获 2 输出使能（Capture/Compare 2 output enable）

参考 CC1E 的描述。

Bit 3 **CC1NP**：输入/捕获 1 输出极性（Capture/Compare 1 output polarity）

**CC1 通道配置为输出：**

在这种情况下， CC1NP 必须保持清零。

**CC1 通道配置为输入：**

此位与 CC1P 配合使用，用以定义 tim_ti1fp1/ tim_ti2fp1 的极性。请参见 CC1P 说明。

Bit 2 保留，必须保持复位值。

Bit 1 **CC1P**：输入/捕获 1 输出极性（Capture/Compare 1 output polarity）

0：OC1 高电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

1：OC1 低电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

**CC1 通道配置为输入，**CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

CC1NP=0，CC1P=0：非反相/上升沿触发。电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=0，CC1P=1：反相/下降沿触发。电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=1，CC1P=1：非反相/上升沿和下降沿均触发。电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

CC1NP=1，CC1P=0：此配置是保留的，不应该使用此配置。

Bit 0 **CC1E**：输入/捕获 1 输出使能（Capture/Compare 1 output enable）

0：禁止捕获模式/OC1 未激活

1：使能捕获模式/ OC1 信号输出到对应的输出引脚

表 23.18 标准 tim_ocx 通道的输出控制位

| CCxE 位 | tim_ocx 输出状态 |
|---|---|
| 0 | 禁止输出（不由定时器驱动：高阻态） |
| 1 | 输出使能（tim_ocx = tim_ocxref + 极性） |

*注：当*CCxE=0 *时，连接到*tim_ocx *通道的外部*IO *引脚状态仅取决于*GPIO *寄存器*

### 23.7.12 TIMx 计数器（TIMx_CNT）（x = 3，4）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 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 31 **UIFCPY：** 值取决于 TIMx_CR1 的 UIFREMAP

如果 UIFREMAP = 0，保留

如果 UIFREMAP = 1

**UIFCPY：**复制 UIF

该位是 TIMx_SR 寄存器中 UIF 位的一个只读副本。

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CNT[15:0]：**计数器的值（Counter value）

在非抖动模式下（DITHEN = 0）

该寄存器（CNT[15:0]）保存的是计数器的值。

在抖动模式下（DITHEN = 1）

该寄存器（CNT[15:0]）只保存计数器的不抖动部分。抖动部分不可用。

### 23.7.13 TIMx 计数器（TIMx_CNT）（x = 2，5）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | Reserved | | | CNT[27:16] | | | | | | | | | | | |
| r | | | | 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 31 **UIFCPY：**值取决于 TIMx_CR1 的 UIFREMAP

如果 UIFREMAP = 0

保留

如果 UIFREMAP = 1

**UIFCPY：**复制 UIF

该位是 TIMx_SR 寄存器中 UIF 位的一个只读副本。

Bit 30:28 保留，必须保持复位值。

Bit 27:0 **CNT[27:0]：**计数器值的最小有效部分（Least significant part of counter value）

在非抖动模式下（DITHEN = 0）

该寄存器（CNT[27:0]）保存的是计数器的值。

在抖动模式下（DITHEN = 1）

该寄存器（CNT[27:0]）只保存计数器的不抖动部分。抖动部分不可用。

### 23.7.14 TIMx 预分频器（TIMx_PSC）（x = 2~5）

偏移地址：0x028

复位值：0x0000

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

Bit 15:0 **PSC[15:0]**：预分频器的值（Prescaler value）

计数器时钟频率（$f_{tim_cnt_ck}$）等于 $f_{tim_psc_ck}$ / (PSC[15:0] + 1)。

PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

### 23.7.15 TIMx 自动重装载寄存器（TIMx_ARR）（x = 3，4）

偏移地址：0x02C

复位：0x0000 FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | 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 31:20 保留，必须保持复位值。

Bit 19:0 **ARR[19:0]**：自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

当自动重载值为空时，计数器不工作。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 ARR[15:0]中的自动重新加载的值

在抖动模式下（DITHEN = 1）

该寄存器存储的是 ARR[19:4]中的整数部分，而 ARR[3:0]位域存储的是抖动的部分

### 23.7.16 TIMx 自动重装载寄存器（TIMx_ARR）（x = 2，5）

偏移地址：0x02C

复位：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 31:0 **ARR[31:0]**：自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

当自动重载值为空时，计数器不工作。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 ARR[27:0]中的自动重新加载的值

在抖动模式下（DITHEN = 1）

该寄存器存储的是 ARR[31:4]中的整数部分，而 ARR[3:0]位域存储的是抖动的部分

### 23.7.17 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 3，4）

偏移地址：0x034

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR1 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR1[19:0]：**捕获/比较 1 值（Capture/Compare 1 value）

**若 CC1 通道配置为输出：**

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 1 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR1[15:0]中的比较值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR1[19:4]中的整数部分。CCR1[3:0]位域存储的是抖动的部分。

**若 CC1 通道配置为输入：**

CCR1 为上一个输入捕获 1 事件（tim_ic1）发生时的计数器值。TIMx_CCR1 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR1[15:0]位存储了捕获值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR1[19:4]中。CCR1[3:0]位会被重置。

### 23.7.18 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 2，5）

偏移地址：0x034

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **CCR1[31:0]：**捕获/比较 1 值（Capture/Compare 1 value）

**若 CC1 通道配置为输出：**

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 1 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR1[27:0]中的比较值。CCR1[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR1[31:4]中的整数部分。CCR1[3:0]位域存储的是抖动的部分。

**若 CC1 通道配置为输入：**

CCR1 为上一个输入捕获 1 事件（tim_ic1）发生时的计数器值。TIMx_CCR1 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR1[27:0]位存储了捕获值。CCR1[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR1[31:4]中。CCR1[3:0]位会被重置。

### 23.7.19 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 3，4）

偏移地址：0x038

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR2 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR2[19:0]：**捕获/比较 2 值（Capture/Compare 2 value）

**若 CC2 通道配置为输出：**

CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc2 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR2[15:0]中的比较值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR2[19:4]中的整数部分。CCR2[3:0]位域存储的是抖动的部分。

**若 CC2 通道配置为输入：**

CCR2 为上一个输入捕获 2 事件（tim_ic2）发生时的计数器值。TIMx_CCR2 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR2[15:0]位存储了捕获值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR2[19:4]中。CCR2[3:0]位会被重置。

### 23.7.20 TIMx 捕获/比较寄存器 2（TIMx_CCR2）（x = 2，5）

偏移地址：0x038

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2[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 |
| CCR2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **CCR2[31:0]：**捕获/比较 2 值（Capture/Compare 2 value）

**若 CC2 通道配置为输出：**

CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR2[27:0]中的比较值。CCR2[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR2[31:4]中的整数部分。CCR2[3:0]位域存储的是抖动的部分。

**若 CC2 通道配置为输入：**

CCR2 为上一个输入捕获 2 事件（tim_ic2）发生时的计数器值。TIMx_CCR2 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR2[27:0]位存储了捕获值。CCR2[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR2[31:4]中。CCR2[3:0]位会被重置。

### 23.7.21 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 3，4）

偏移地址：0x03C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR3 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR3[19:0]：**捕获/比较 3 值（Capture/Compare 3 value）

**若 CC3 通道配置为输出：**

CCR3 为要装载到实际捕获/比较 3 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC3PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 3 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc3 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR3[15:0]中的比较值。CCR3[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR3[19:4]中的整数部分。CCR3[3:0]位域存储的是抖动的部分。

**若 CC3 通道配置为输入：**

CCR3 为上一个输入捕获 3 事件（tim_ic3）发生时的计数器值。TIMx_CCR3 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR3[15:0]位存储了捕获值。CCR3[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR3[19:4]中。CCR3[3:0]位会被重置。

### 23.7.22 TIMx 捕获/比较寄存器 3（TIMx_CCR3）（x = 2，5）

偏移地址：0x03C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR3[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 |
| CCR3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **CCR3[31:0]：**捕获/比较 3 值（Capture/Compare 3 value）

**若 CC3 通道配置为输出：**

CCR3 为要装载到实际捕获/比较 3 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC3PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 3 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc3 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR3[27:0]中的比较值。CCR3[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR3[31:4]中的整数部分。CCR3[3:0]位域存储的是抖动的部分。

**若 CC3 通道配置为输入：**

CCR3 为上一个输入捕获 3 事件（tim_ic3）发生时的计数器值。TIMx_CCR3 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR3[27:0]位存储了捕获值。CCR3[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR3[31:4]中。CCR3[3:0]位会被重置。

### 23.7.23 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 3，4）

偏移地址：0x040

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR4[19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR4[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR4[19:0]：**捕获/比较 4 值（Capture/Compare 4 value）

**若 CC4 通道配置为输出：**

CCR4 为要装载到实际捕获/比较 4 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC4PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 4 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc4 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR4[15:0]中的比较值。CCR4[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR4[19:4]中的整数部分。CCR4[3:0]位域存储的是抖动的部分。

**若 CC4 通道配置为输入：**

CCR4 为上一个输入捕获 4 事件（tim_ic4）发生时的计数器值。TIMx_CCR4 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR4[15:0]位存储了捕获值。CCR4[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR4[19:4]中。CCR4[3:0]位会被重置。

### 23.7.24 TIMx 捕获/比较寄存器 4（TIMx_CCR4）（x = 2，5）

偏移地址：0x040

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR4[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 |
| CCR4[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:0 **CCR4[31:0]：**捕获/比较 4 值（Capture/Compare 4 value）

**若 CC4 通道配置为输出：**

CCR4 为要装载到实际捕获/比较 4 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR2 寄存器中的 OC4PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 4 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc4 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR4[27:0]中的比较值。CCR4[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR4[31:4]中的整数部分。CCR4[3:0]位域存储的是抖动的部分。

**若 CC4 通道配置为输入：**

CCR4 为上一个输入捕获 4 事件（tim_ic4）发生时的计数器值。TIMx_CCR4 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR4[27:0]位存储了捕获值。CCR4[31:28]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR4[31:4]中。CCR4[3:0]位会被重置。

### 23.7.25 TIMx 编码器控制寄存器（TIMx_ECR）（x = 2~5）

偏移地址：0x058

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | IPOS[1:0] | | FIDX | Reserved | | IDIR[1:0] | | IE |
| | | | | | | | | rw | rw | rw | | | rw | rw | rw |

Bit 31:27 保留，必须保持复位值。

Bit 26:24 **PWPRSC[2:0]：**脉冲宽度预分频器（Pulse width prescaler）

该位域设置脉冲发生器的时钟预分频，如下所示：

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bit 23:16 **PW[7:0]：**脉冲宽度（Pulse width）

该位域定义脉冲持续时间，如下所示：

$t_{PW} = PW[7:0] \times t_{PWG}$

Bit 15:8 保留，必须保持复位值。

Bit 7:6 **IPOS[1:0]：索引位置**（Index positioning）

在正交编码器模式（SMS[3:0] = 0001，0010，0011，1110，1111）下，该位指示了索引事件在哪个 AB 输入配置下会重置计数器。

00：当 AB=00 时，索引重置计数器

01：当 AB=01 时，索引重置计数器

10：当 AB=10 时，索引重置计数器

11：当 AB=11 时，索引重置计数器

在方向时钟模式或时钟加方向模式（SMS[3:0] = 1010，1011，1100，1101）下，此位指示了索引事件在哪个时钟水平上重置计数器。在双向时钟模式下，这适用于两个时钟输入。具体为：

x0：当时钟=0 时，索引重置计数器

x1：当时钟=1 时，索引重置计数器

*注：IPOS[1]位不重要*

Bit 5 **FIDX[1:0]：首位索引**（First index）

该位指示是否仅考虑首位索引

0：索引始终处于活跃状态

1：仅首个索引会重置计数器

Bit 4:3 保留，必须保持复位值。

Bit 2:1 **IDIR[1:0]：索引方向**（Index direction）

该位指示索引事件在哪个方向上重置计数器。

00：无论方向如何，索引都会重置计数器

01：仅在递增计数时，索引会重置计数器

10：仅在递减计数时，索引会重置计数器

11：保留

*注：当 IE 位被复位（禁止索引）时，必须写入 IDR[1:0]位域*

Bit 0 **IE：索引使能**（Index enable）

该位指示索引事件是否能重置计数器

0：禁止索引

1：使能索引

### 23.7.26 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x = 2~5）

偏移地址：0x05C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TI4SEL[3:0] | | | | Reserved | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TI2SEL[3:0] | | | | Reserved | | | | TI1SEL[3:0] | | | |
| | | | | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bit 31:28 保留，必须保持复位值。

Bit 27:24 **TI4SEL[3:0]：**选择 tim_ti4[15:0]输入（Selects tim_ti4[15:0] input）

0000：tim_ti4_in0: TIMx_CH4

0001：tim_ti4_in1

...

1111：tim_ti4_in15

参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号获得关联列表。

Bit 23:20 保留，必须保持复位值。

Bit 19:16 **TI3SEL[3:0]：**选择 tim_ti3[15:0]输入（Selects tim_ti3[15:0] input）

0000：tim_ti3_in0: TIMx_CH3

0001：tim_ti3_in1

...

1111：tim_ti3_in15

参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号获得关联列表。

Bit 15:12 保留，必须保持复位值。

Bit 11:8 **TI2SEL[3:0]：**选择 tim_ti2[15:0]输入（Selects tim_ti2[15:0] input）

0000：tim_ti2_in0: TIMx_CH2

0001：tim_ti2_in1

...

1111：tim_ti2_in15

参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号获得关联列表。

Bit 7:4 保留，必须保持复位值。

Bit 3:0 **TI1SEL[3:0]：**选择 tim_ti1[15:0]输入（Selects tim_ti1[15:0] input）

0000：tim_ti1_in0: TIMx_CH1

0001：tim_ti1_in1

...

1111：tim_ti1_in15

参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号获得关联列表。

### 23.7.27 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 2~5）

偏移地址：0x060

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | ETRSEL[3:2] | |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 31:18 保留，必须保持复位值。

Bit 17:14 **ETRSEL[3:0]：**etr_in 的源选择（etr_in source selection）

这些位选择 etr_in 的输入源

0000：tim_etr0: TIMx_ETR 输入

0001：tim_etr1

...

1111：tim_etr15

请参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号，了解特定于产品的实现。

Bit 13:0 保留，必须保持复位值。

### 23.7.28 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 2~5）

偏移地址：0x064

复位值：0x0000 0000

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

Bit 31:19 保留，必须保持复位值。

Bit 18:16 **OCRSEL[2:0]：**ocref_clr 的源选择（ocref_clr source selection）

这些位选择 ocref_clr 的输入源

000：tim_ocref_clr0

001：tim_ocref_clr1

...

111：tim_ocref_clr7

请参考章节：TIM2/TIM3/TIM4/TIM5 引脚和内部信号，了解特定于产品的实现。

Bit 15:0 保留，必须保持复位值。

### 23.7.29 TIMx DMA 控制寄存器（TIMx_DCR）（x = 2~5）

偏移地址：0x3DC

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DBL[4:0] | | | | | Reserved | | | DBA[4:0] | | | | |
| | | | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持复位值。

Bit 12:8 **DBL[4:0]：**DMA 突发长度（DMA burst length）

这个 5 位向量定义了 DMA 传输的长度（当对 TIMx_DMAR 地址进行读或写访问时，定时器识别一次突发传输），即传输次数。传输可以是半字或字节（见下面的示例）。

00000：1 次传输

00001：2 次传输

00010：3 次传输

…

11010：26 次传输

**示例：**让我们考虑以下传输：DBL = 7 字节 & DBA = TIM2_CR1。如果 DBL = 7 字节，并且 DBA = TIM2_CR1 表示要传输的字节的地址，那么传输的地址应由以下公式给出：(TIMx_CR1 地址) + DBA + (DMA 索引)，其中 DMA 索引 = DBL

在这个例子中，将 7 字节添加到(TIMx_CR1 地址) + DBA，这给出了数据从/复制到的地址。在这种情况下，传输是从以下地址开始的 7 个寄存器：(TIMx_CR1 地址) + DBA

根据 DMA 数据大小的配置，可能会发生几种情况：

– 如果 DMA 数据大小配置为半字，则将 16 位数据传输到 7 个寄存器中的每一个。

– 如果 DMA 数据大小配置为字节，则数据也传输到 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，依此类推。

因此，在使用传输定时器时，还必须指定 DMA 传输的数据大小。

Bit 7:5 保留，必须保持复位值。

Bit 4:0 **DBA[4:0]：**DMA 基地址（DMA base address）

这个 5 位向量定义了 DMA 传输的基地址（当通过 TIMx_DMAR 地址进行读/写访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始的偏移量。

00000：TIMx_CR1,

00001：TIMx_CR2,

00010：TIMx_SMCR,

…

### 23.7.30 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 2~5）

偏移地址：0x3E0

复位值：0x0000 0000

| 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 31:0 **DMAB[31:0]：**用于突发访问的 DMA 寄存器（DMA register for burst accesses）

对 DMAR 寄存器进行的读或写操作将访问位于以下地址的寄存器：

(TIMx_CR1 地址) + (DBA + DMA 索引) × 4

其中，TIMx_CR1 地址是控制寄存器 1 的地址，DBA 是在 TIMx_DCR 寄存器中配置的 DMA 基地址，DMA 索引由 DMA 传输自动控制，其范围从 0 到 DBL（在 TIMx_DCR 中配置的 DBL）。

# 24 通用定时器（TIM15/TIM16/TIM17）

## 24.1 TIM15/TIM16/TIM17 简介

TIM15/TIM16/TIM17 定时器包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

它可用于多种用途，包括测量输入信号的脉冲宽度（输入捕获）或生成输出波形（输出比较和 PWM 和带死区插入的互补 PWM）。 使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。TIM15/TIM16/TIM17 定时器是完全独立的，不与其他定时器共享任何资源。TIM15 可以按照章节：定时器同步（仅 TIM15）的描述进行同步。

## 24.2 TIM15 主要功能

TIM15 定时器功能包括：

- 16 位自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 2 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成（边缘模式）
  - 单脉冲模式输出
- 带可编程死区的互补输出（仅限通道 1）
- 使用外部信号控制定时器和定时器互连的同步电路
- 重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器。
- 用于将定时器的输出信号置于复位状态或已知状态的刹车输入。
- 如下事件发生时产生中断/DMA：
  - 更新：计数器溢出，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
  - 刹车输入（中断请求）

## 24.3 TIM16/TIM17 主要功能

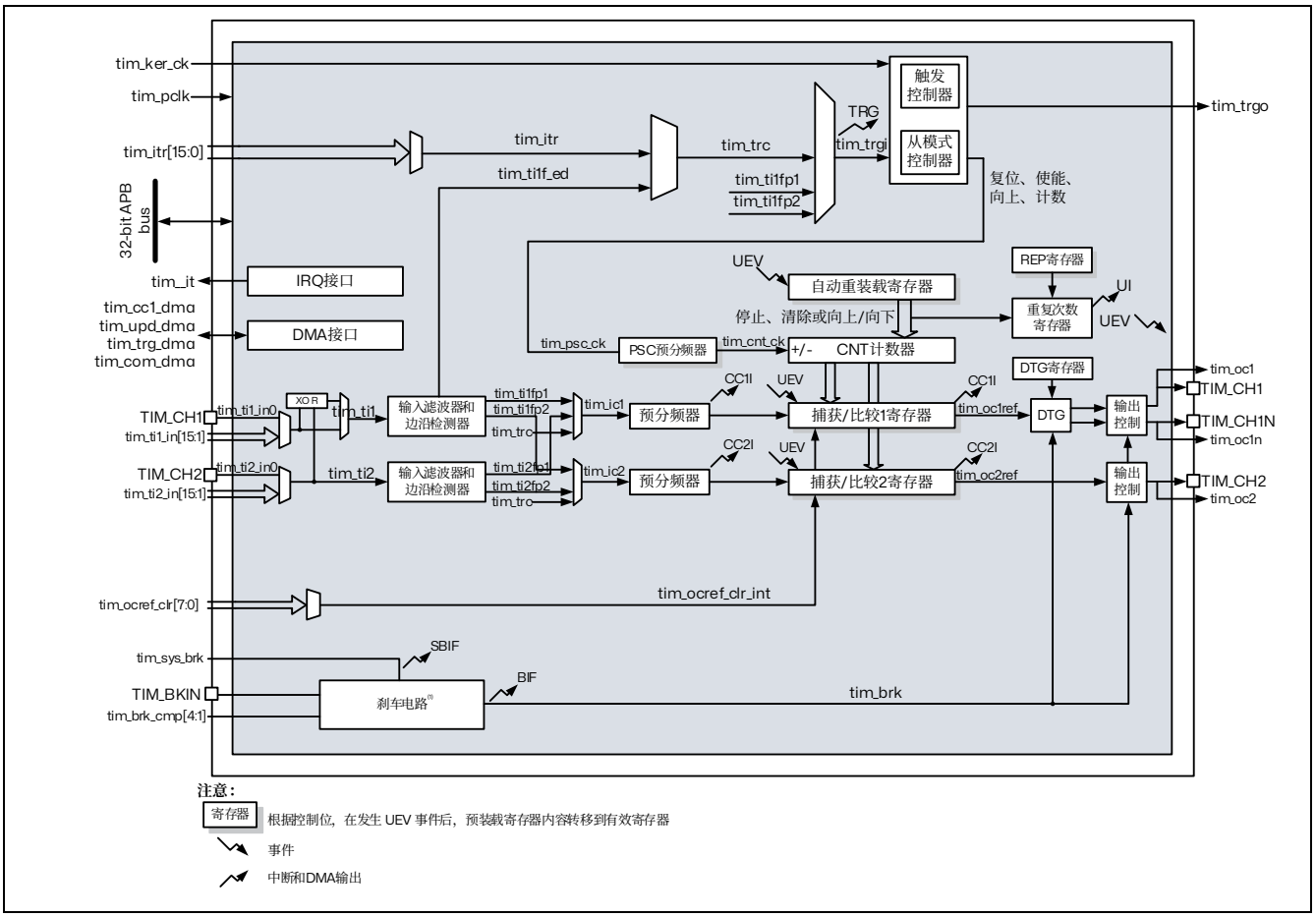
TIM16/TIM17 定时器功能包括：

- 16 位自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 1 个通道：
  - 输入捕获
  - 输出比较
  - PWM 生成（边缘模式）
  - 单脉冲模式输出
- 带可编程死区的互补输出
- 重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器。
- 用于将定时器的输出信号置于复位状态或已知状态的刹车输入。
- 如下事件发生时产生中断/DMA：
  - 更新：计数器溢出
  - 输入捕获
  - 输出比较
  - 刹车输入

## 24.4 TIM15/TIM16/TIM17 功能描述

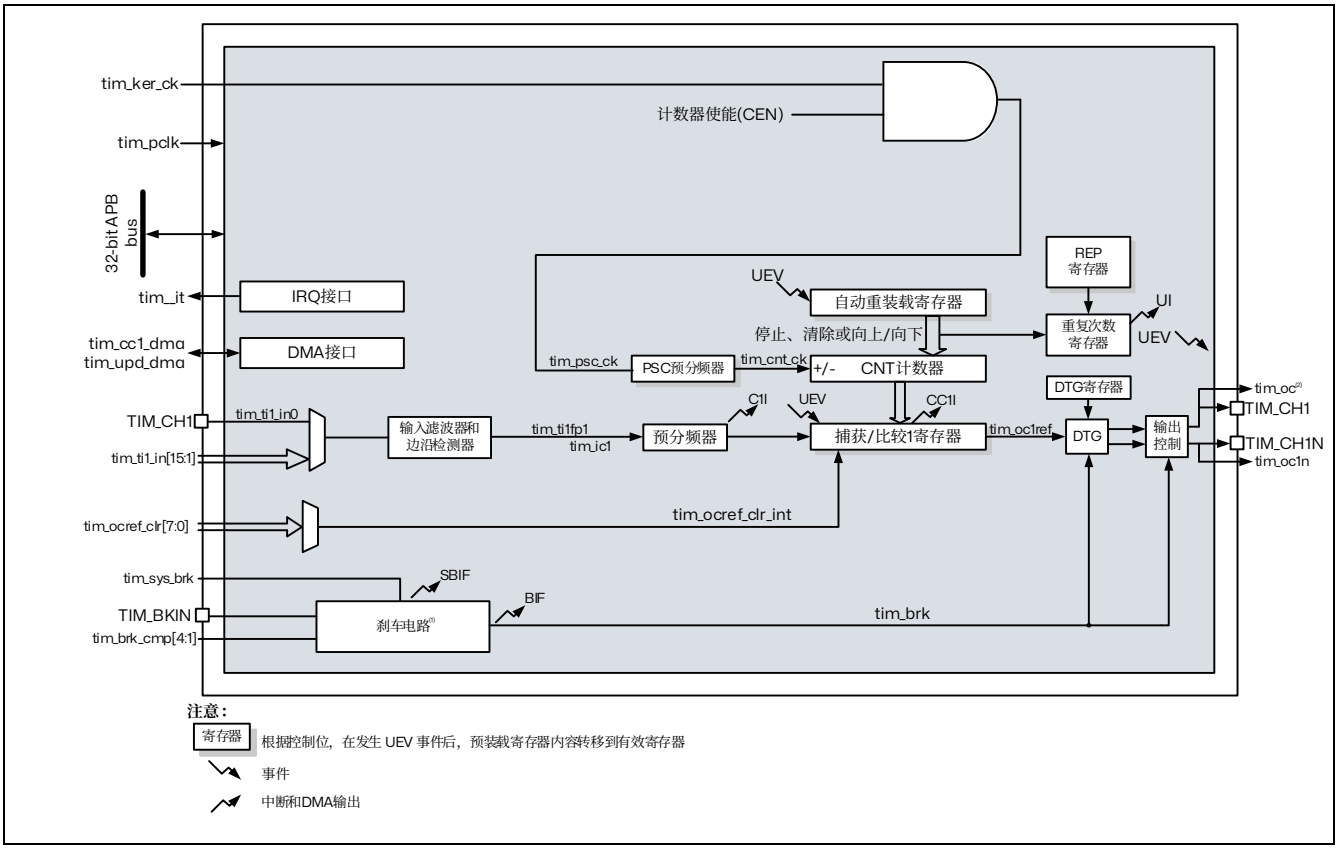
### 24.4.1 框图

图 24.1 TIM15 框图




注 1. 参考章节：使用刹车功能以获得细节

图 24.2 TIM16/TIM17 框图




注 1. 参考章节：使用刹车功能以获得细节

注 2. 此信号可作为某些从属定时器的触发信号（请参阅下一节中的内部触发连接表）。有关详细信息，请参考章节：使用定时器输出作为其他定时器的触发信号（仅 TIM16/TIM17）.

### 24.4.2 TIM15/TIM16/TIM17 引脚和内部信号

本节中的下两张表格总结了 TIM 的输入和输出。

表 24.1 TIM 输入/输出引脚

| 引脚名称 | 信号类型 | 描述 |
|---|---|---|
| TIM_CH1<br>TIM_CH2(1) | 输入/输出 | 定时器多功能通道。<br>每个通道用于捕获、比较或 PWM。<br>TIM_CH1 和 TIM_CH2 也可用作外部时钟（tim_ker_ck 时钟的 1/4 以下）和外部触发器输入。 |
| TIM_CH1N | 输出 | 由 TIM_CH1 输出派生的定时器的互补输出，具有插入死区时间的能力。 |
| TIM_BKIN | 输入/输出 | 刹车输入。此输入也可以配置为双向模式。 |

1. 仅供 TIM15 使用。

表 24.2 TIM 内部输入/输出信号

| 内部信号名称 | 信号类型 | 描述 |
|---|---|---|
| tim_ti1_in[15:0]<br>tim_ti2_in[15:0](1) | 输入 | 内部定时器输入总线。<br>这些输入可被用作捕获或作为外部时钟<br>(tim_ker_ck 时钟的 1/4 以下)。 |
| tim_itr[15:0] (1) | 输入 | 内部触发输入总线。这些输入可用于从模式控制器或作为输入时钟(tim_ker_ck 时钟的 1/4 以下)。 |
| tim_trgo(1) | 输出 | 内部触发器输出。此触发器可以触发其他片上外设。 |
| tim_ocref_clr[7:0] | 输入 | 定时器 tim_ocref_clr 输入总线。这些输入可以用来清除 tim_ocxref 信号，通常用于硬件的逐周期脉冲宽度控制。 |
| tim_brk_cmp[4:1] | 输入 | 内部信号的刹车输入。 |
| tim_sys_brk[n:0] | 输入 | 系统刹车输入。该输入收集 MCU 的系统级错误。 |
| tim_pclk | 输入 | 定时器 APB 时钟。 |
| tim_ker_ck | 输入 | 定时器内核时钟。此时钟必须与 tim_pclk 同步（来自同一源）。<br>时钟比率 tim_ker_ck/ tim_pclk 必须为整数：<br>1，2，3，...，16（最大值）。 |
| tim_it | 输出 | 全局定时器中断，收集捕获/比较、更新、中断触发和换向请求。 |
| tim_cc1_dma | 输出 | 定时器捕获/比较 1 dma 请求。 |
| tim_upd_dma | 输出 | 定时器更新 dma 请求。 |
| tim_trg_dma | 输出 | 定时器触发 dma 请求。 |
| tim_com_dma | 输出 | 定时器换向 dma 请求。 |

1. 仅供 TIM15 使用。

如下两张表列出了连接到 tim_ti[2:1]输入多路复用器的源。

表 24.3 连接到 tim_ti1 的输入多路复用器

| tim_ti1 输入 | 源 | | |
|---|---|---|---|
| | TIM15 | TIM16 | TIM17 |
| tim_ti1_in0 | TIM15_CH1 | TIM16_CH1 | TIM17_CH1 |
| tim_ti1_in1 | 保留 | | |
| tim_ti1_in2 | comp1_out | MCO | MCO |
| tim_ti1_in3 | comp2_out | HSE / 32(1) | HSE / 32(1) |
| tim_ti1_in4 | 保留 | RTC Clock | RTC Clock |
| tim_ti1_in5 | 保留 | | |
| tim_ti1_in6 | 保留 | LSI | LSI |
| tim_ti1_in[15:7] | 保留 | | |

1. 仅供 TIM15 使用。此信号仅在 TIMx_OR1 寄存器中设置了 HSE32EN 位时可用，有关详细信息，请参考章节：TIMx 选项寄存器 1（TIMx_OR1）（x = 16~17）。

表 24.4 连接到 tim_ti2 的输入多路复用器

| tim_ti2 输入 | 源 |
|---|---|
| | TIM15 |
| tim_ti2_in0 | TIM15_CH2 |
| tim_ti2_in1 | comp2_out |
| tim_ti2_in2 | comp3_out |
| tim_ti2_in3 | 保留 |
| tim_ti2_in4 | |
| tim_ti2_in[15:5] | |

下表列出连接到 tim_itr 输入多路复用器的内部源

表 24.5 TIMx 内部触发连接

| tim_itrx 输入 | TIM15 |
|---|---|
| tim_itr0 | tim1_trgo |
| tim_itr1 | tim2_trgo |
| tim_itr2 | tim3_trgo |
| tim_itr3 | tim4_trgo |
| tim_itr4 | tim5_trgo |
| tim_itr5 | tim8_trgo |
| tim_itr6 | 保留 |
| tim_itr7 | tim16_oc1 |
| tim_itr8 | tim17_oc1 |
| tim_itr9 | 保留 |
| tim_itr10 | hrtim_out_sync2 |
| tim_itr[15:11] | 保留 |

下两张表列出了连接到 tim_brk 输入的源

表 24.6 定时器刹车互连

| tim_brk 输入 | TIM15 | TIM16 | TIM17 |
|---|---|---|---|
| TIM_BKIN | TIM15_BKIN pin | TIM16_BKIN pin | TIM17_BKIN pin |
| tim_brk_cmp1 | comp1_out | comp1_out | comp1_out |
| tim_brk_cmp2 | comp2_out | comp2_out | comp2_out |
| tim_brk_cmp3 | comp3_out | comp3_out | comp3_out |
| tim_brk_cmp4 | comp4_out | comp4_out | comp4_out |

表 24.7 系统中断互连

| tim_sys_brk 输入 | TIM15/TIM16/TIM17 | SYSCFG_CFGR2 寄存器的使能位 |
|---|---|---|
| tim_sys_brk0 | 内核锁住 | CLL |
| tim_sys_brk1 | 可编程电压检测器（PVD） | PVDL |
| tim_sys_brk2 | SRAM 奇偶校验错误 | SPL |
| tim_sys_brk3 | Flash ECC 错误 | ECCL |
| tim_sys_brk4 | 时钟安全系统(CSS) | 无（始终使能） |

下表列出连接到 tim_ocref_clr 输入多路复用器的内部源

表 24.8 连接到 ocref_clr 的输入多路复用器

| 定时器 OCREF 清除信号 | 定时器 tim_ocref_clr 信号分配 | | |
|---|---|---|---|
| | TIM15 | TIM16 | TIM17 |
| tim_ocref_clr0 | comp1_out | comp1_out | comp1_out |
| tim_ocref_clr1 | comp2_out | comp2_out | comp2_out |
| tim_ocref_clr2 | comp3_out | comp3_out | comp3_out |
| tim_ocref_clr3 | comp4_out | comp4_out | comp4_out |

### 24.4.3 时基单元

可编程定时器的主要模块由一个 16 位递增计数器及其相关的自动重装寄存器组成。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

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）
- 重复计数器寄存器 (TIMx_RCR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件(UEV)时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到溢出并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位（CEN）置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

注意，计数器将在 TIMx_CR1 寄存器的 CEN 位置 1 时刻的一个时钟周期后开始计数。

#### 预分频器描述

预分频器可对计数器时钟频率进行分频，分频系数介于 1 到 65536 之间。该预分频器基于 16 位寄存器（TIMx_PSC 寄存器）所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用.

下图给出了在预分频器运行时，更改计数器参数的例子。

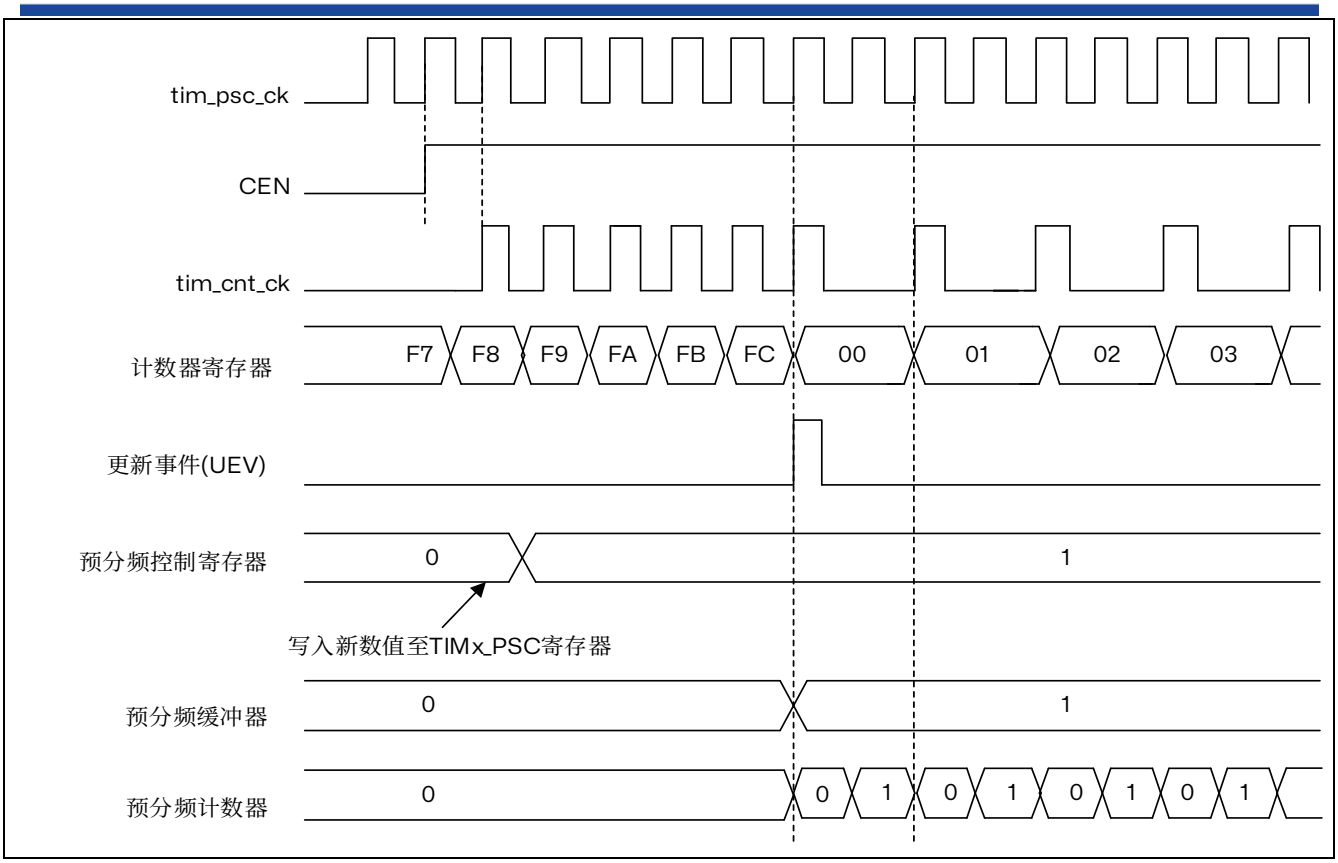



图 24.3 当预分频器的参数从 1 变到 2 时，计数器的时序图

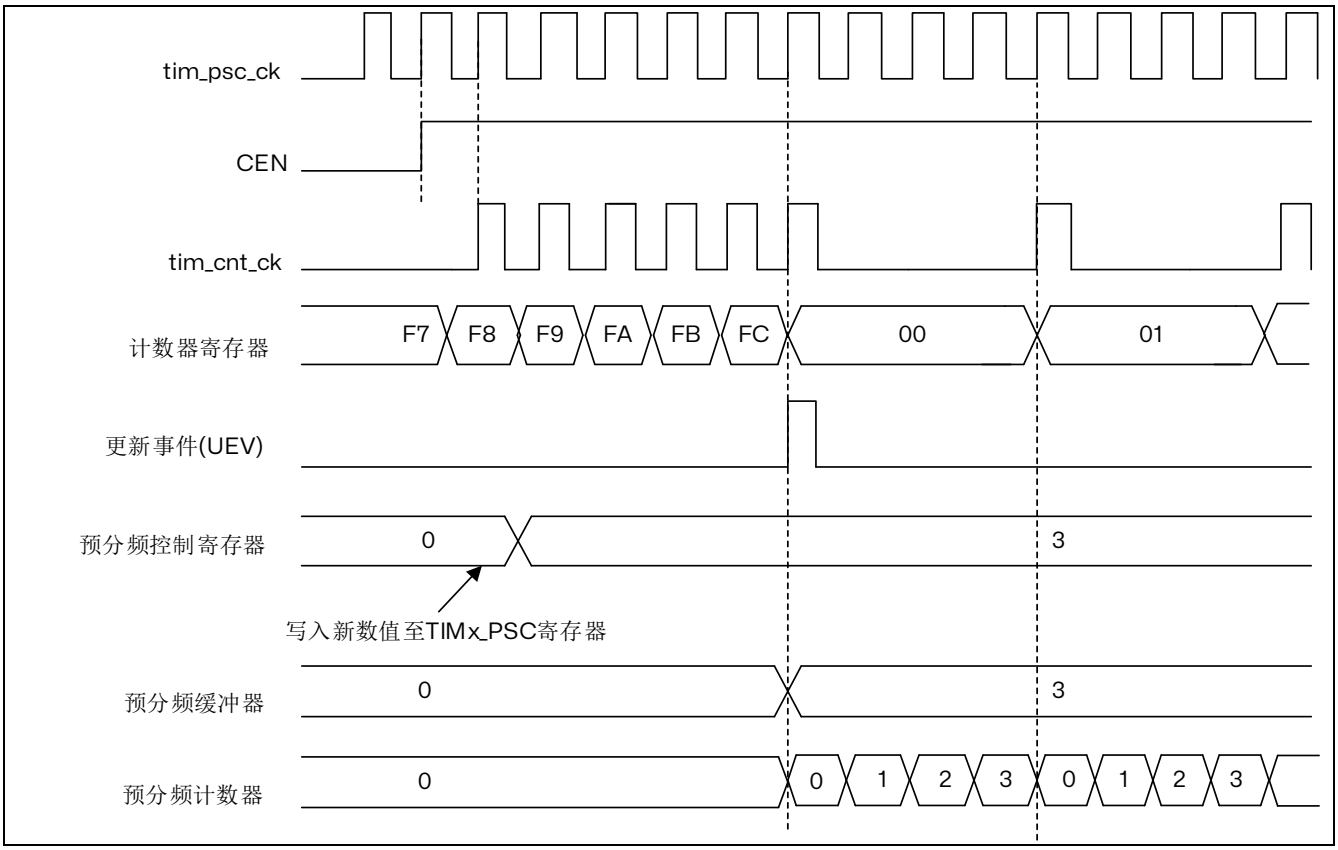



图 24.4 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 24.4.4 计数器模式

#### 递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

如果使用重复计数器，则当递增计数的重复次数达到重复计数器寄存器中编程的次数加一次(TIMx_RCR+1)后，将生成更新事件（UEV)。否则，将在每次计数器上溢时产生更新事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变)。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断)。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位)：

- 重复计数器中将重新装载 TIMx_RCR 寄存器的内容
- 自动重载影子寄存器将以预装载值进行更新（TIMx_ARR）
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

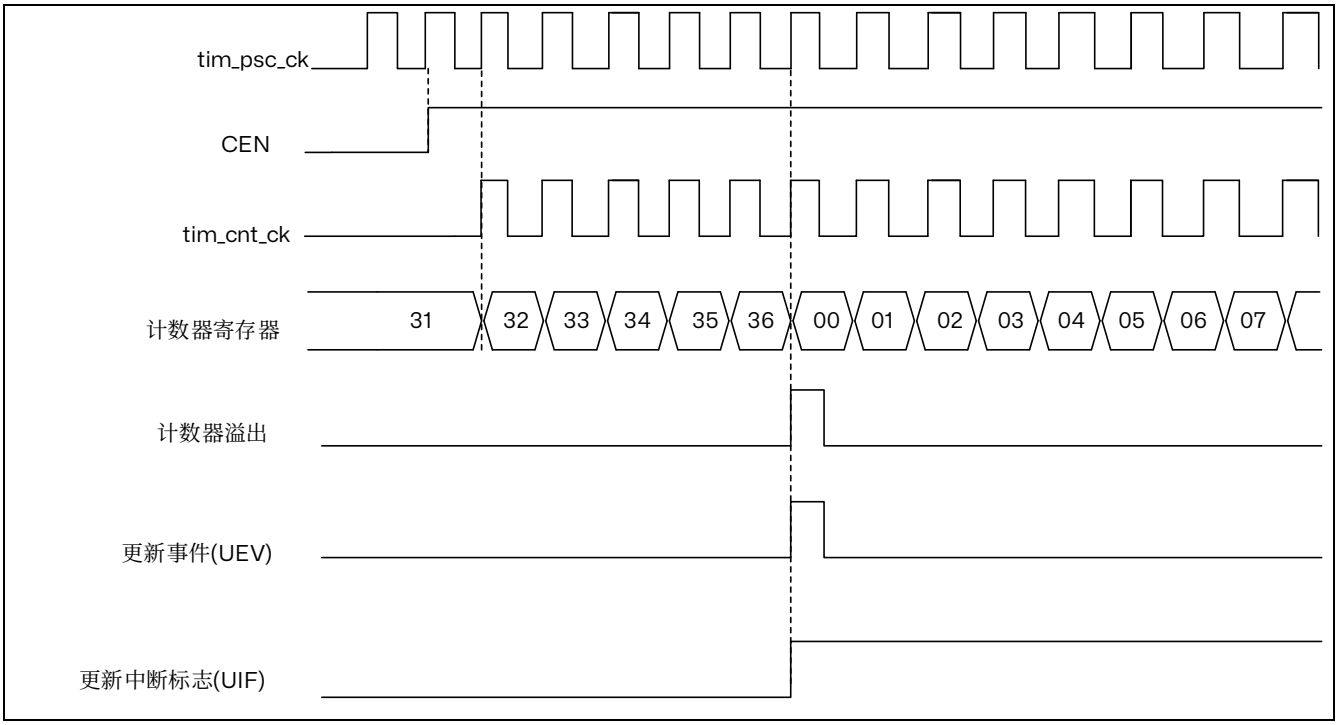



图 24.5 计数器时序图，内部时钟分频系数为 1

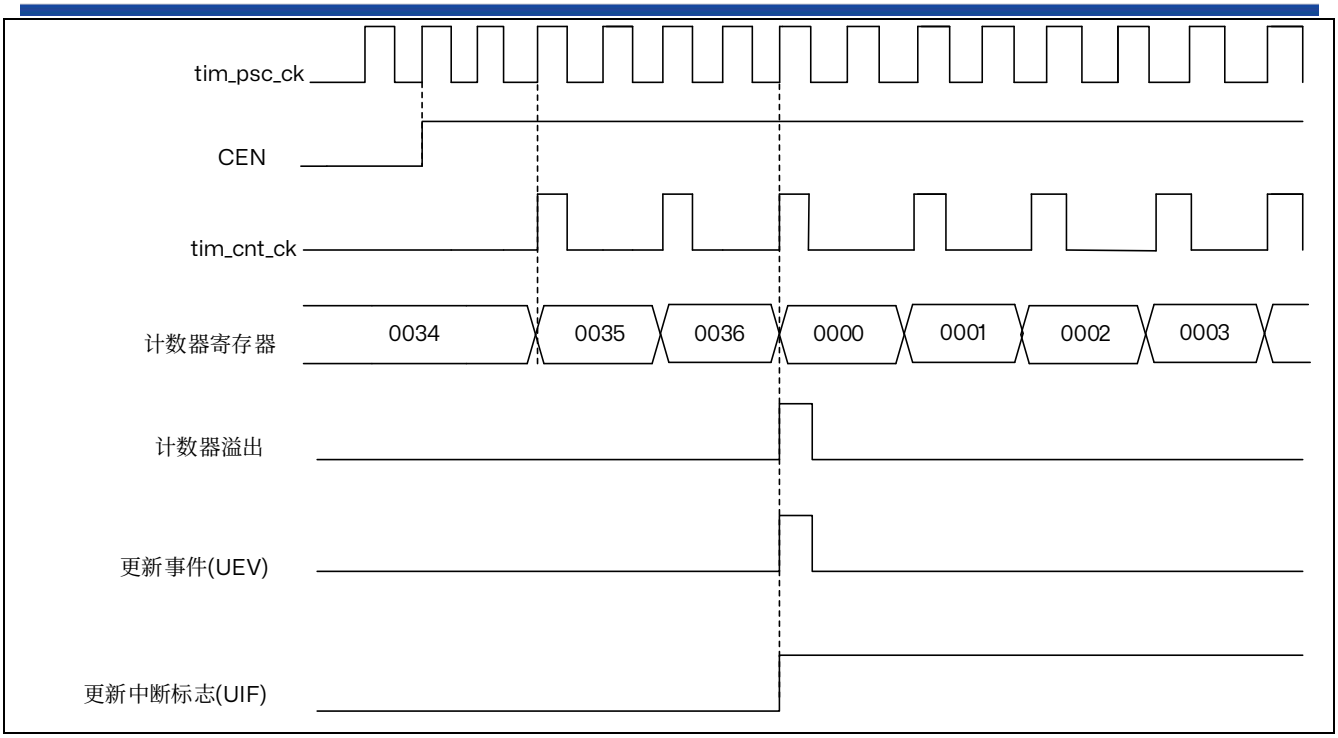



图 24.6 计数器时序图，内部时钟分频系数为 2

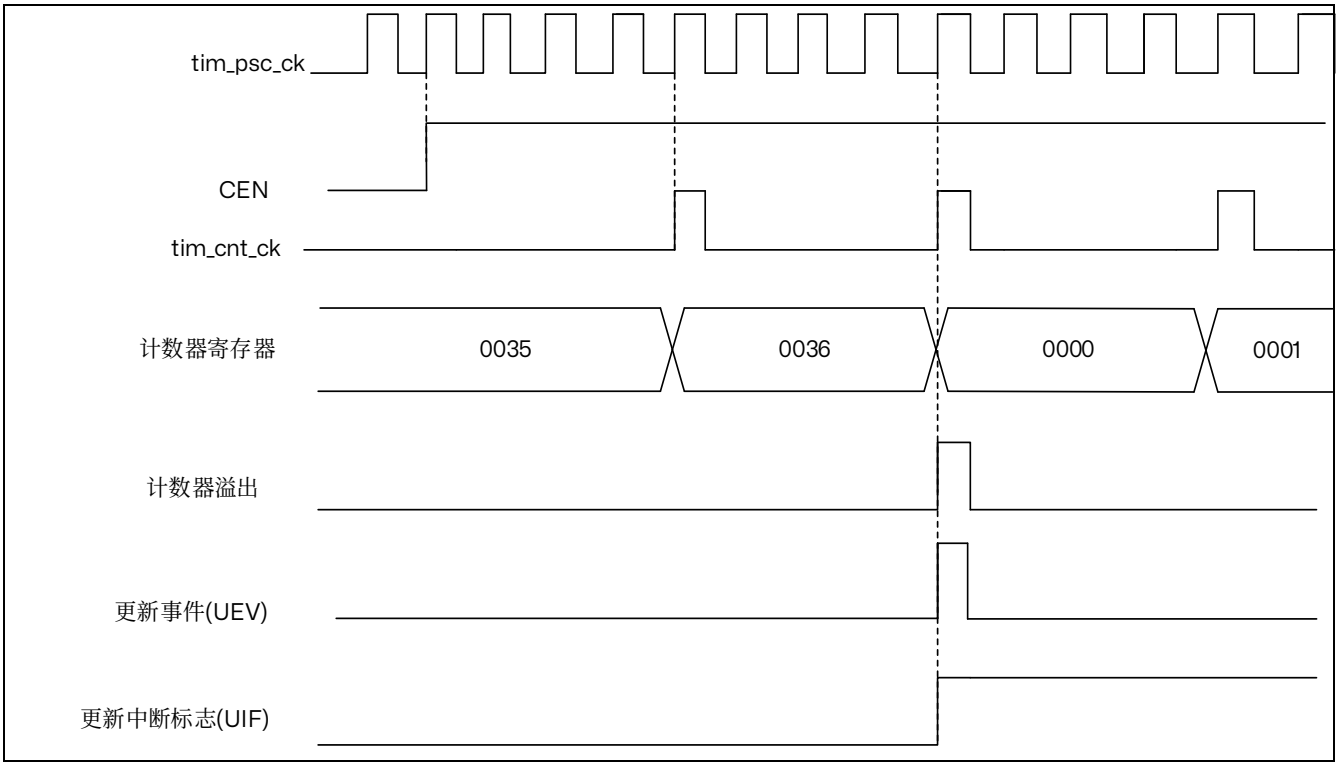



图 24.7 计数器时序图，内部时钟分频系数为 4

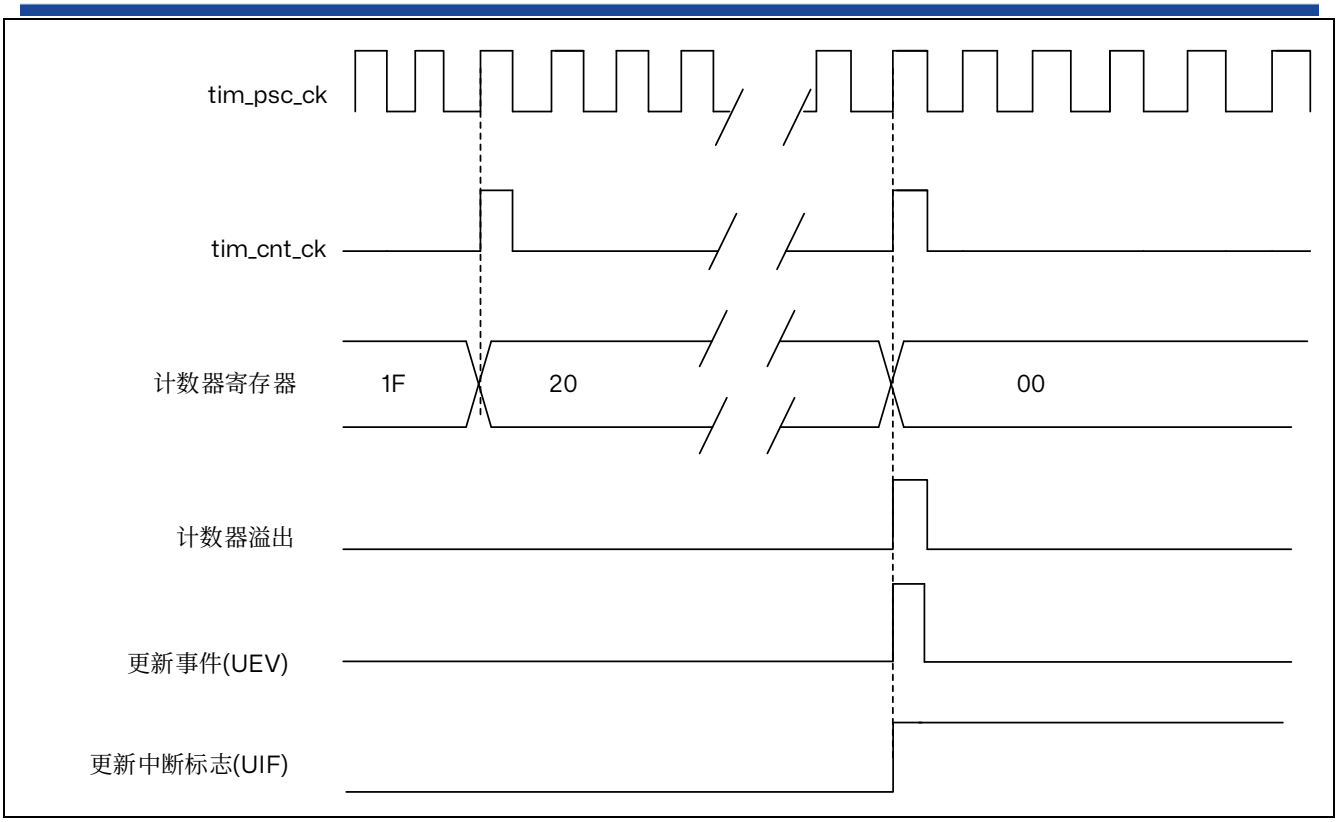



图 24.8 计数器时序图，内部时钟分频系数为 N

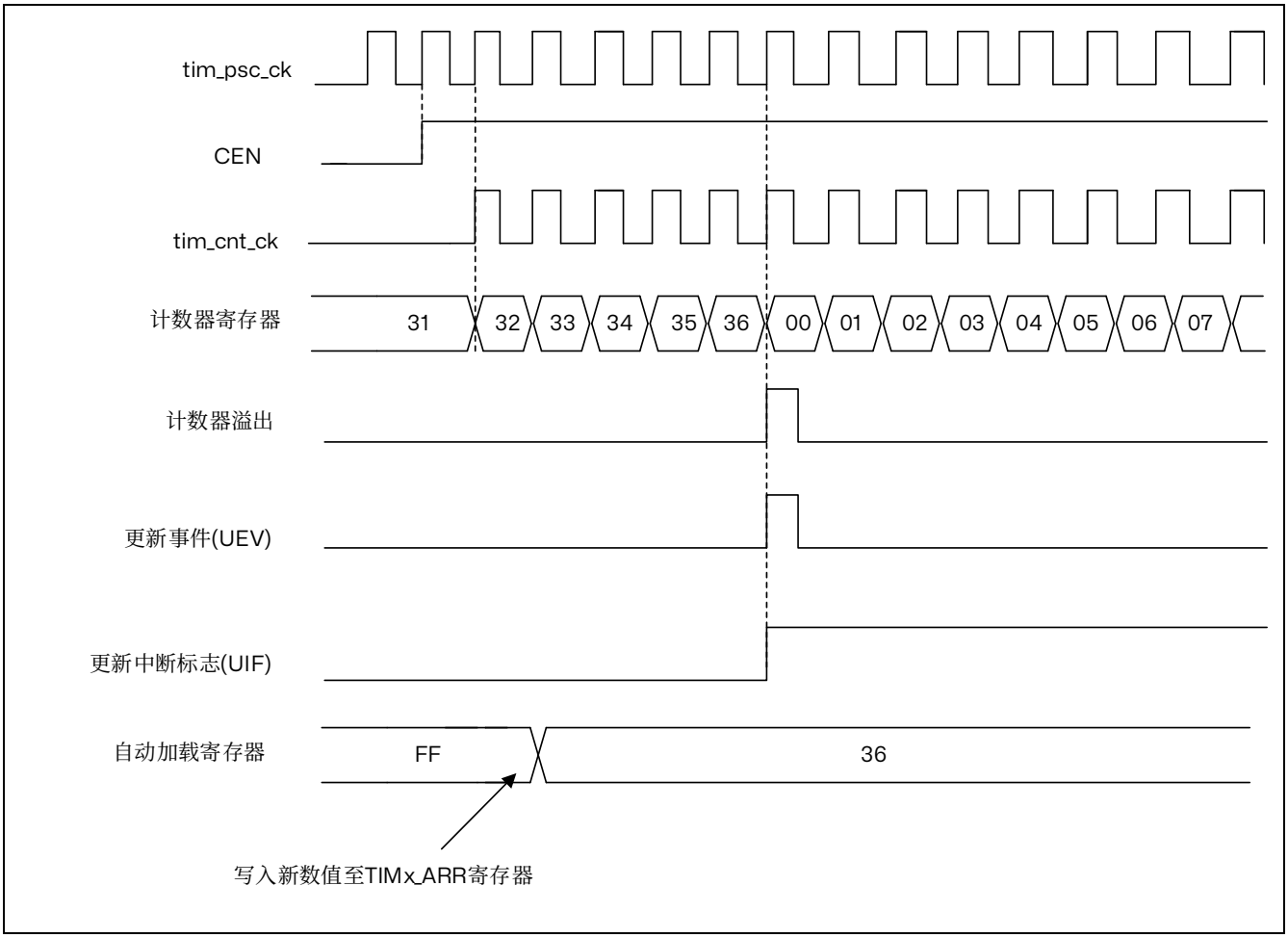


图 24.9 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

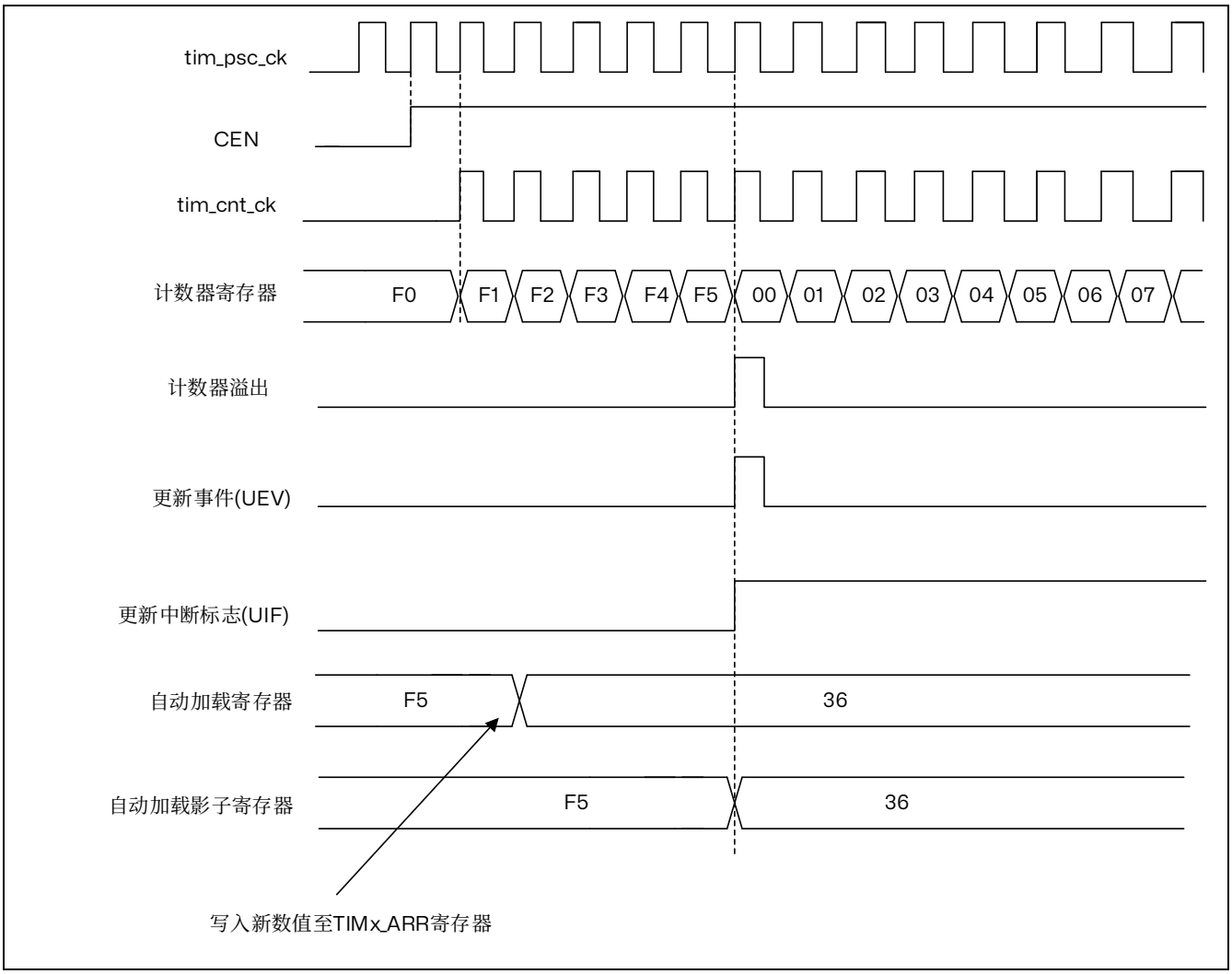



图 24.10 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

### 24.4.5 重复计数器

时基单元介绍如何因计数器溢出而生成更新事件(UEV)。实际上，只有当重复计数器达到零时，才会生成更新事件。这在生成 PWM 信号时很有用。

这意味着，每当发生 N 个计数器溢出（其中，N 是 TIMx_RCR 重复计数器寄存器中的值），数据就将从预装载寄存器转移到影子寄存器（TIMx_ARR 自动重载寄存器、TIMx_PSC 预分频器寄存器以及比较模式下的 TIMx_CCRx 捕获/比较寄存器）。

重复计数器在每次计数器溢出时递减：

重复计数器是自动重载类型；其重复率为 TIMx_RCR 寄存器所定义的值。当更新事件由软件（通过将 TIMx_EGR 寄存器的 UG 位置 1）或硬件（通过从模式控制器）生成时，无论重复计数器的值为多少，更新事件都将立即发生，并且在重复计数器中重新装载 TIMx_RCR 寄存器的内容。

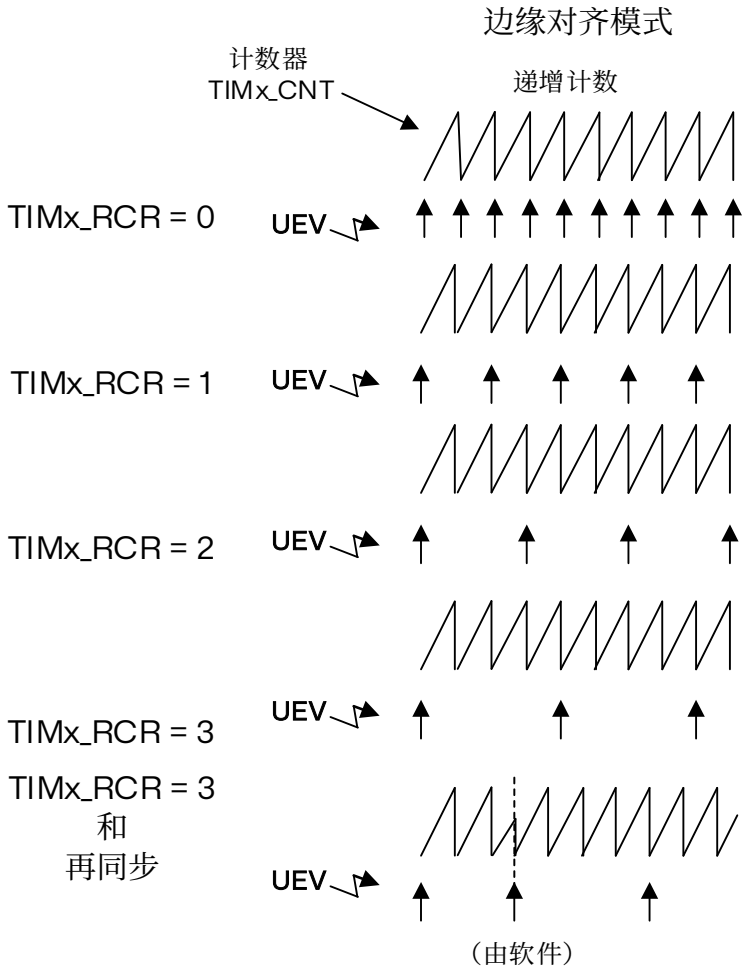



图 24.11 取决于模式和 TIMx_RCR 寄存器设置的更新频率示例

### 24.4.6 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（tim_ker_ck）
- 外部时钟模式 1：外部输入引脚（tim_ti1 或 tim_ti2，如果可用）
- 内部触发输入(tim_itrx)：使用一个定时器作为另一个定时器的预分频器，例如可以将 TIM1 配置为 TIM15 的预分频器。

**内部时钟源（tim_ker_ck）**

如果禁止从模式控制器（TIMx_SMCR 寄存器中 SMS=000），则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 tim_ker_ck 提供。

下图显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

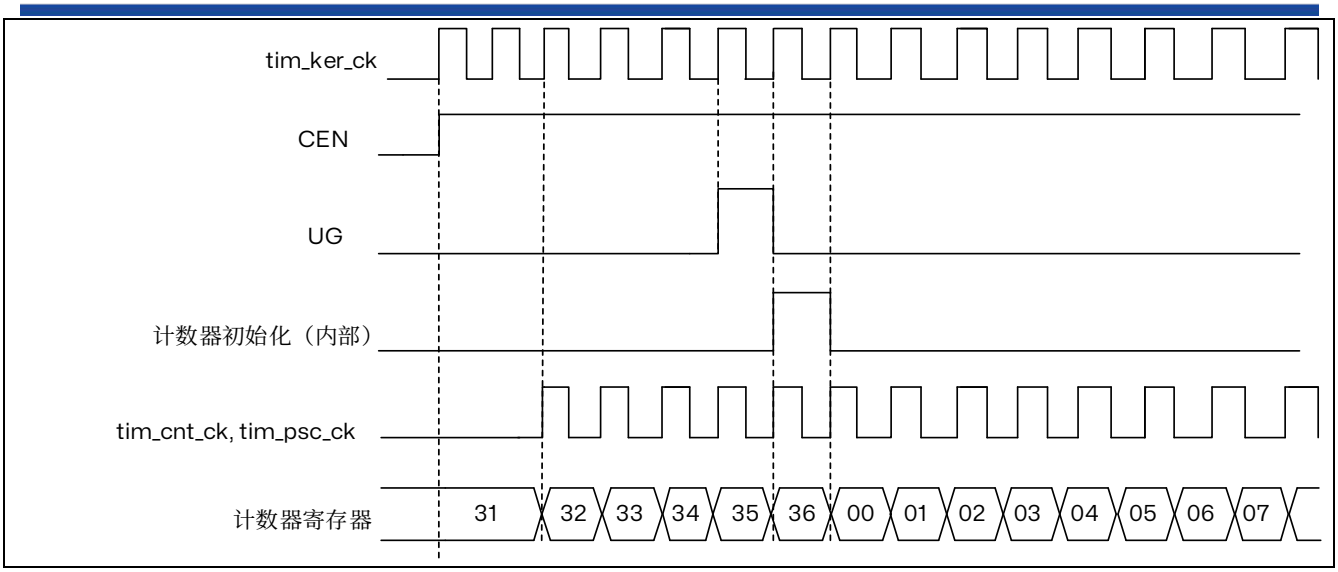



图 24.12 一般模式下的控制电路，内部时钟分频系数为 1

**外部时钟源模式 1**

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

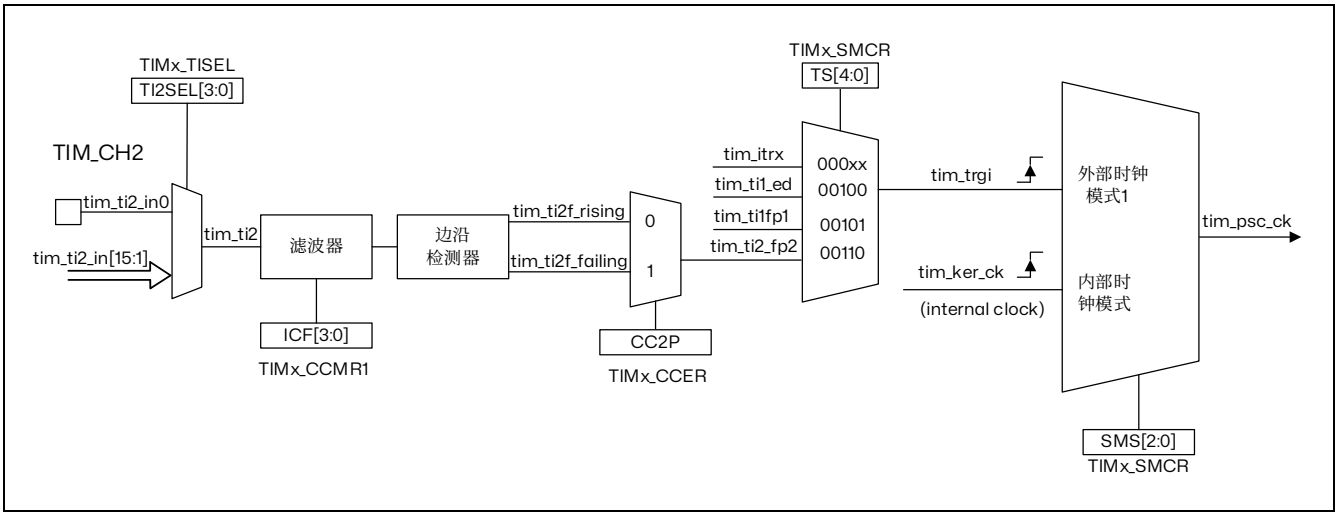



图 24.13 tim_ti2 外部时钟连接例子

例如，要使递增计数器在 tim_ti2 输入出现上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_TISEL 寄存器中使用 TI2SEL[3:0]位选择合适的 tim_ti2_in[15:0]源（内部或外部）。
2. 通过在 TIMx_CCMR1 寄存器中写入 CC2S=“01”来配置通道 2，使其能够检测 tim_ti2 输入的上升沿。
3. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0]位来配置输入滤波时间（如果不需要任何滤波，请保持 IC2F=0000）。
4. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。
5. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
6. 通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 tim_ti2 作为输入源。
7. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

*注：由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

当 tim_ti2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。

tim_ti2 的上升沿与实际计数器时钟之间的延迟是由于 tim_ti2 输入的重新同步电路引起的。

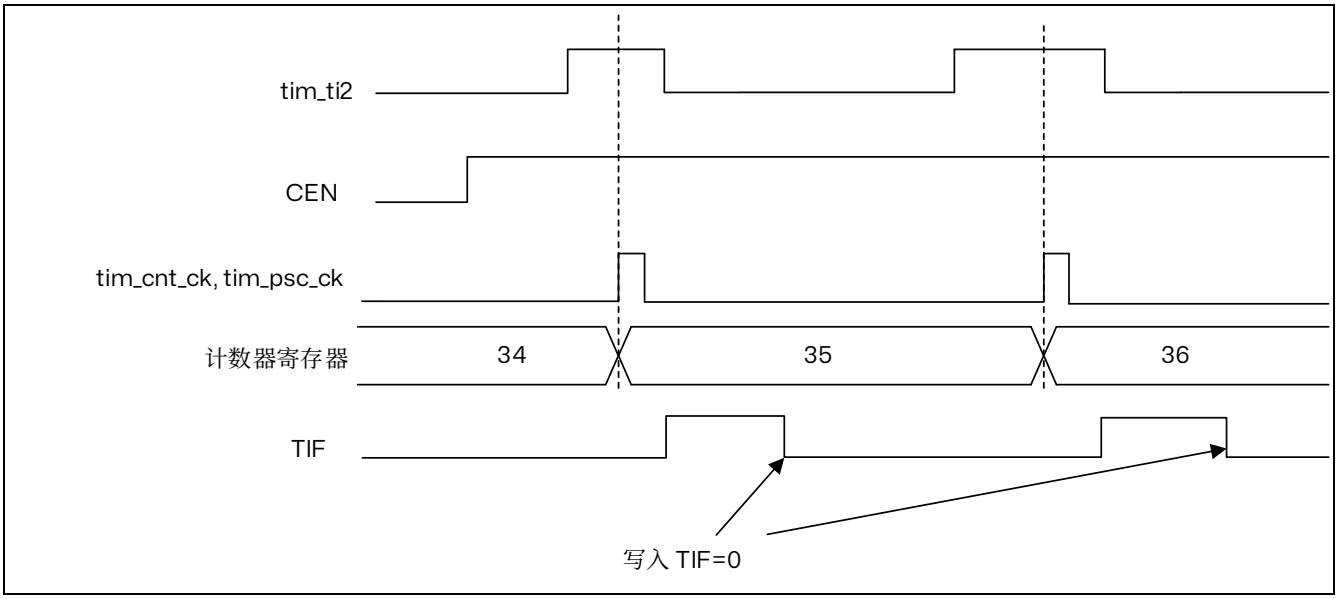


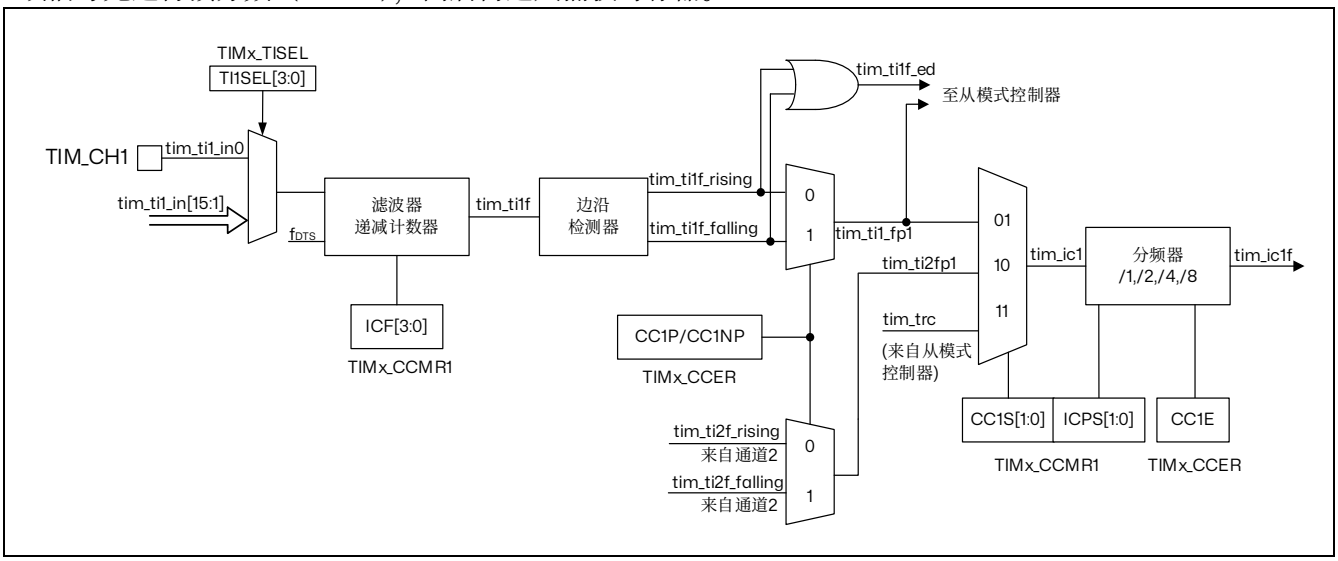

图 24.14 外部时钟模式 1 下的控制电路

### 24.4.7 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

下图概括介绍了一个捕获/比较通道。

输入阶段对相应的 tim_tix 输入进行采样，生成一个滤波后的信号 tim_tixf。然后，带有极性选择功能的边沿检测器生成一个信号（tim_tixfpy），该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频（ICxPS），而后再进入捕获寄存器。




输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 24.15 捕获/比较通道（如：通道 1 输入部分）

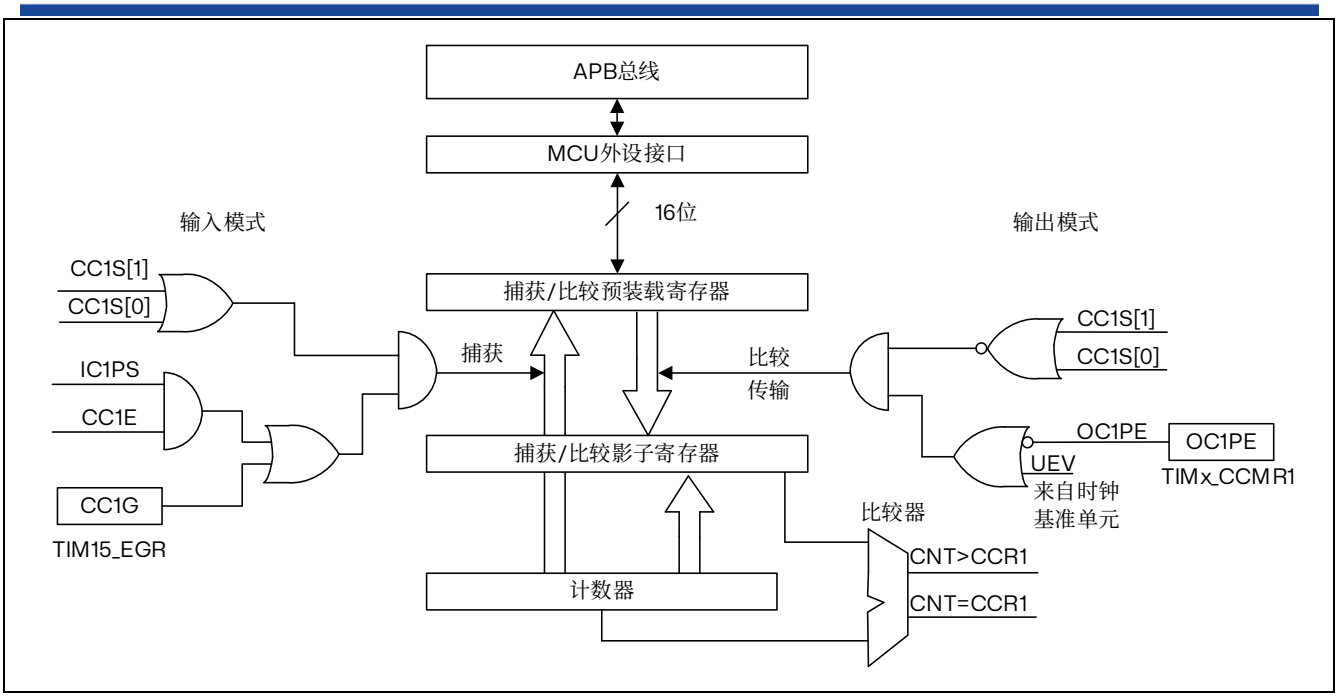



图 24.16 捕获/比较通道 1 的主电路

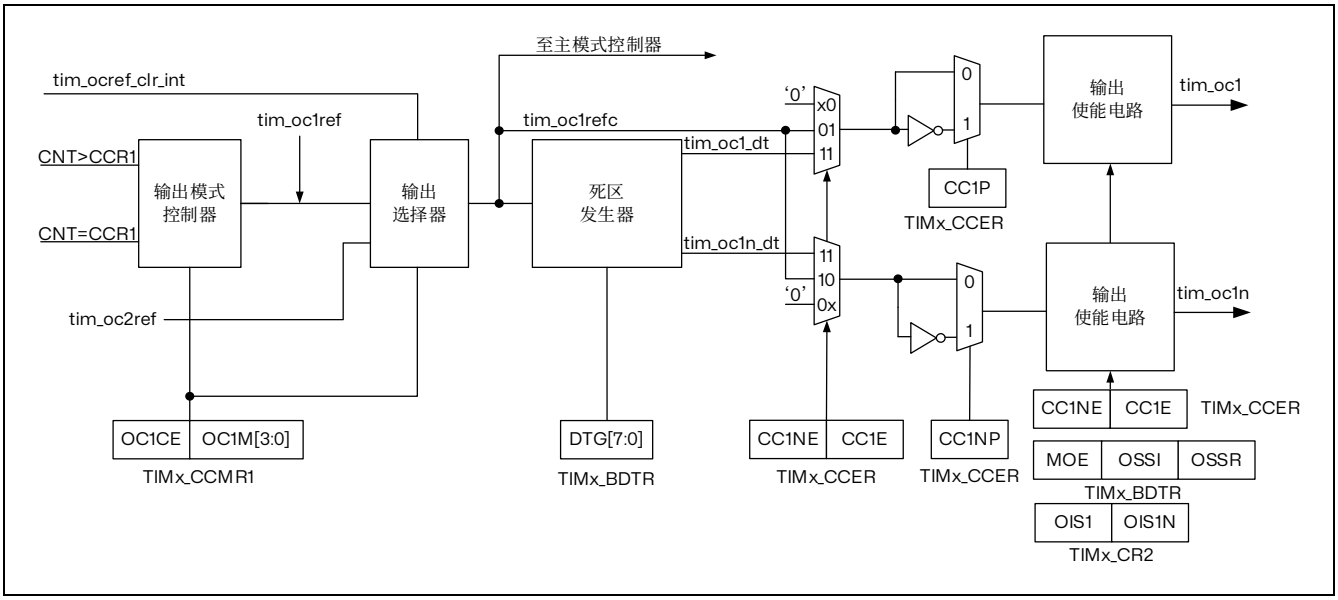



图 24.17 捕获/比较通道的输出阶段（通道 1）

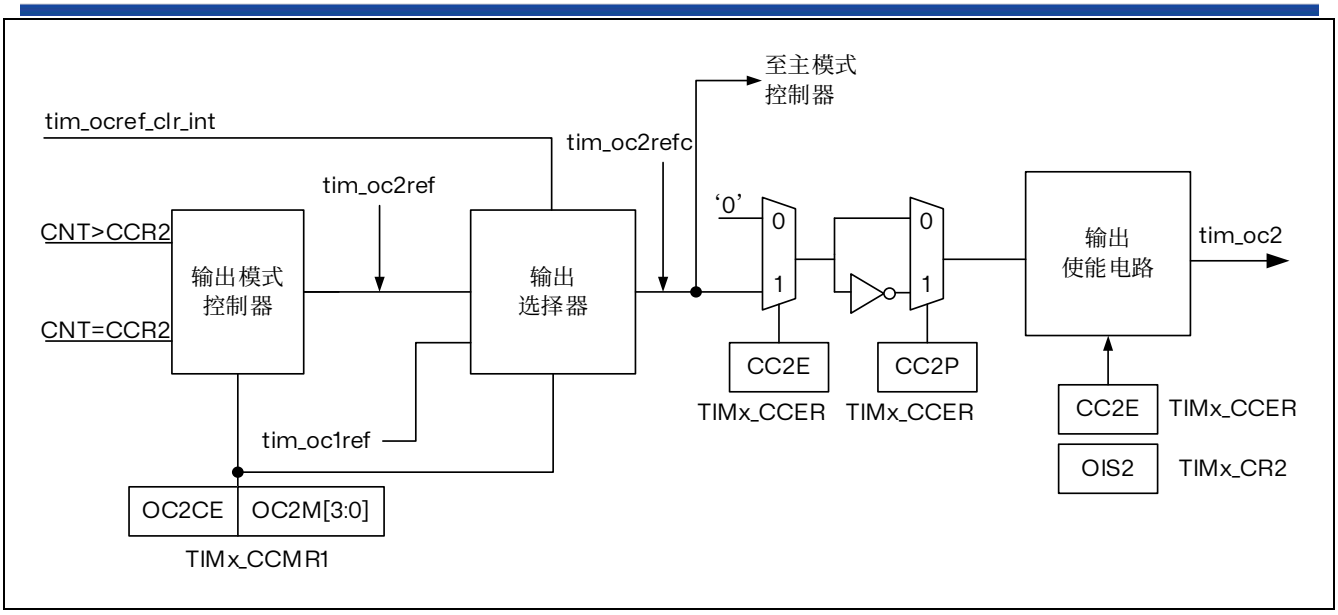



图 24.18 捕获比较通道的输出阶段（TIM15 的通道 2）

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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容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将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

### 24.4.8 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器(TIMx_CCRx)来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入 0 后会将其清零。

以下示例说明了如何在 tim_ti1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 通过在 TIMx_TISEL 寄存器中使用 TI1SEL[3:0]位选择合适的 tim_tix_in[15:1]源（内部或外部）。
2. 选择有效输入：TIMx_CCR1 必须连接到 tim_ti1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
3. 根据连接到定时器的信号，对所需的输入滤波时间进行编程（如果输入为 tim_tix 输入之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波时间设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 $f_{DTS}$ 频率采样）后，可以确认 tim_ti1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。
4. 通过向 TIMx_CCER 寄存器中的 CC1P 位和 CC1NP 位写入 0，选择 tim_ti1 通道的有效转换边沿（本例中为上升沿）。
5. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止

预分频器（向 TIMx_CCMR1 寄存器中的 IC1PSC 位写入 00）。

6. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
7. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。
- 是否生成 DMA 请求取决于 CC1DE 位。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

*注：通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断请求。*

### 24.4.9 PWM 输入模式（仅 TIM15）

此模式允许测量连接到单个 tim_tix 输入的 PWM 信号的周期和占空比：

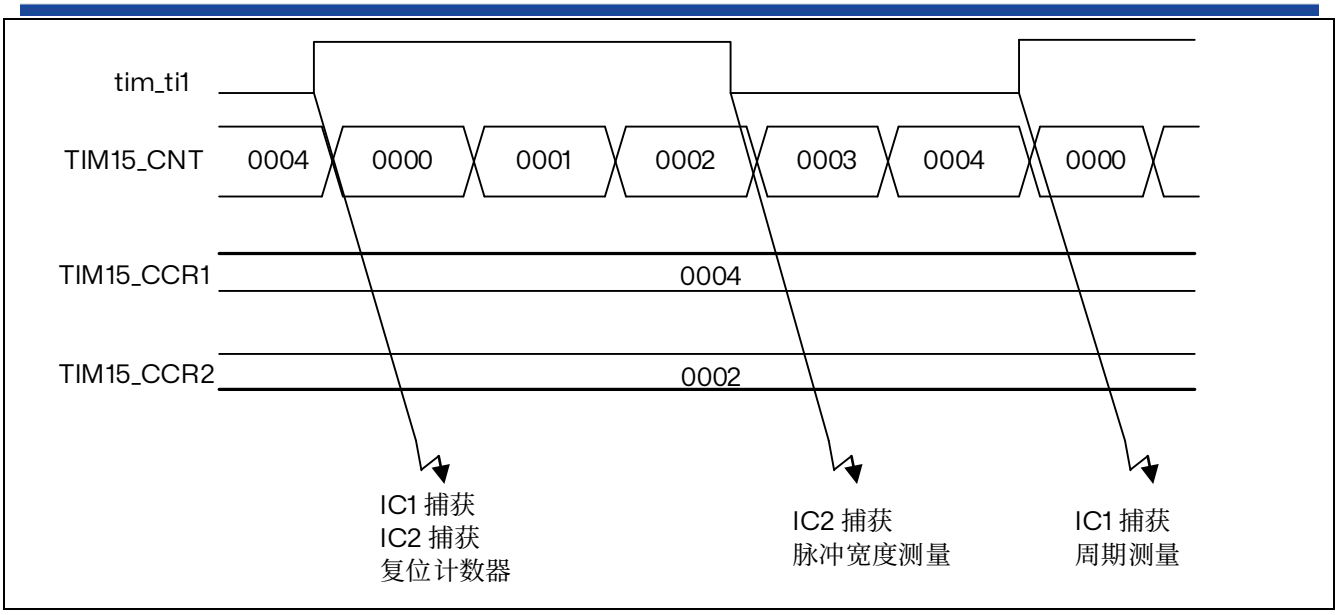
- TIMx_CCR1 寄存器保持周期值（两个连续上升沿之间的间隔）。
- TIMx_CCR2 寄存器保持脉冲宽度（两个连续上升沿和下降沿之间的间隔）。

此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

- 两个 tim_icx 信号被映射至同一个 tim_tix 输入。
- 这两个 tim_icx 信号在边沿处有效，但极性相反。
- 选择两个 tim_tixfpy 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤测量施加在 tim_ti1 上的 PWM 的周期（在 TIMx_CCR1 寄存器中）和占空比（在 TIMx_CCR2 寄存器中）（取决于 tim_ker_ck 频率和预分频器）：

1. 通过在 TIMx_TISEL 寄存器中使用 TI1SEL[3:0]位选择合适的 tim_tix_in[15:0]源（内部或外部）。
2. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 tim_ti1）。
3. 选择 tim_ti1fp1 的有效极性（用于 TIMx_CCR1 中的捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
4. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 tim_ti1）。
5. 选择 tim_ti1fp2 的有效极性（用于 TIMx_CCR2 中的捕获）：向 CC2P 位写入 1 和 CC2NP 位写入“0”（下降沿有效）。
6. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 101（选择 tim_ti1fp1）。
7. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 100。
8. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。




PWM 输入模式只能与 TIM15_CH1/TIM15_CH2 信号一起使用，因为只有 tim_ti1fp1 和 tim_ti2fp2 连接到从模式控制器。

图 24.19 PWM 输入模式时序

### 24.4.10 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位= 00）下，可直接由软件将每个输出比较信号（tim_ocxref 和 tim_ocx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号（tim_ocxref/tim_ocx）强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 101。tim_ocxref 进而强制设置为高电平（tim_ocxref 始终为高电平有效），同时 tim_ocx 获取 CCxP 极性位的相反值。

例如：CCxP=0（tim_ocx 高电平有效）=> tim_ocx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 tim_ocxref 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断请求。输出比较模式一节对此进行了介绍。

### 24.4.11 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（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。匹配时，输出引脚既可保持其电平（OCXM=000），也可设置为有效电平（OCXM=001）、无效电平（OCXM=010）或进行翻转（OCxM=011）。
- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

在输出比较模式下，更新事件 UEV 对 tim_ocxref 和 tim_ocx 输出毫无影响。同步的精度可以达到计

数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

**步骤：**

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，将 CCxIE 位和/或 CCxDE 位置 1。
4. 选择输出模式。例如，
   a) 写入 OCxM=0011 以在 CNT 匹配 CCRx 时翻转 tim_ocx 输出引脚
   b) 写入 OCxPE=0 以禁用预装载寄存器
   c) 写入 CCxP=0 以选择高电平有效
   d) 写入 CCxE=1 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

可随时通过软件更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预装载寄存器（OCxPE=0，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。

下图列出了相关示例。

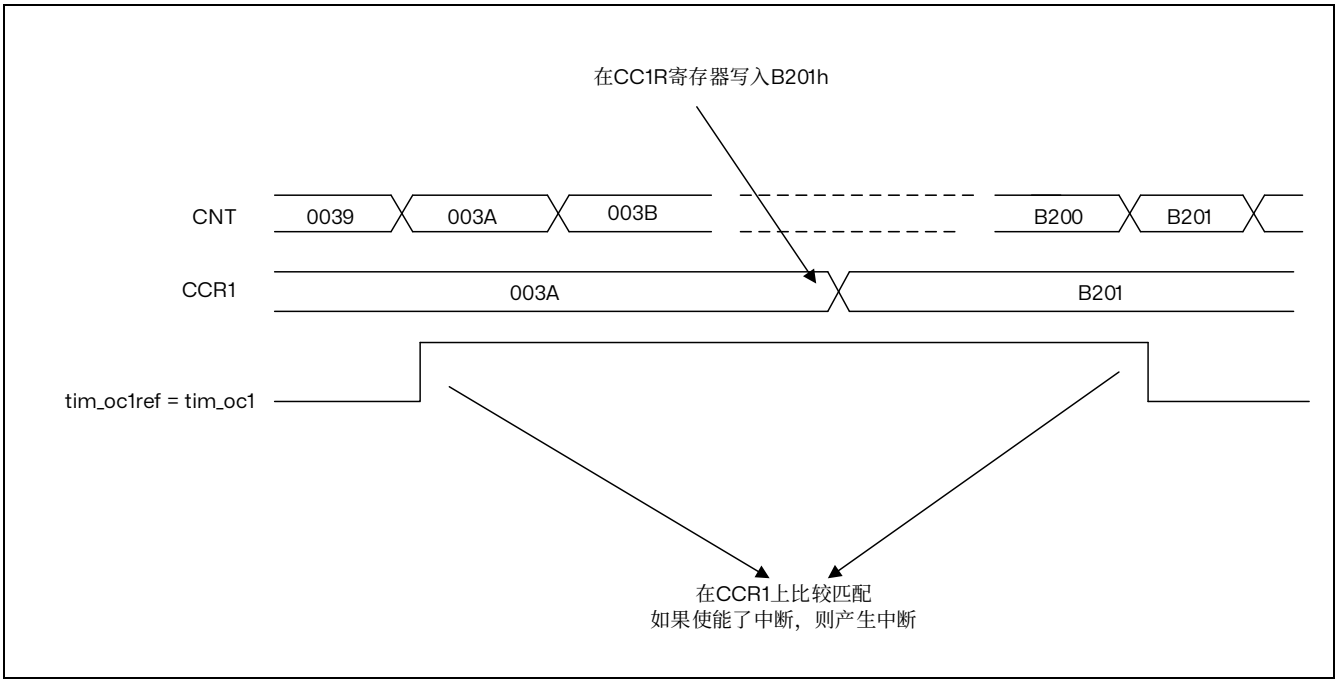



图 24.20 输出比较模式，翻转 tim_oc1

### 24.4.12 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110（PWM 模式 1）或 111（PWM 模式 2），可以独立选择各通道（每个 tim_ocx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

tim_ocx 极性可使用 TIMx_CCER 寄存器的 CCxP 位来编程。既可以设为高电平有效，也可以设为低电平有效。通过 CCxE、CCxNE、MOE、OSSI 和 OSSR 位（TIMx_CCER 和 TIMx_BDTR 寄存器）的组合使能 tim_ocx 输出。有关详细信息，请参见 TIMx_CCER 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 始终与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx ≤ TIMx_CNT 还是 TIMx_CNT ≤ TIMx_CCRx（取决于计数器计数方向）。

TIM15/TIM16/TIM17 只能递增计数，具体请参考递增计数模式。

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 tim_ocxref 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 tim_ocxref 保持为“1”。如果比较值为 0，则 tim_ocxref 保持为“0”。下图举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

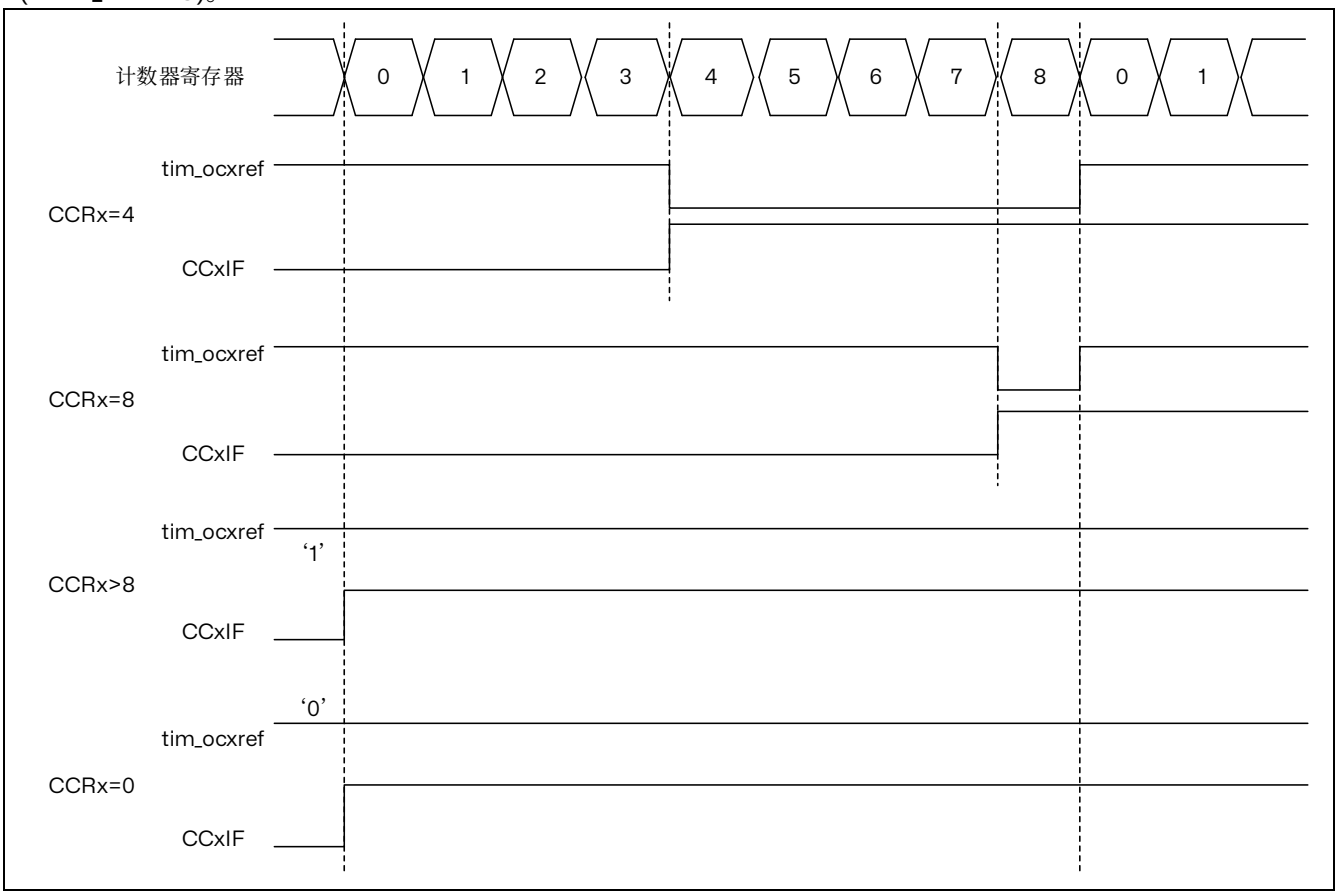



图 24.21 边沿对齐的 PWM 波形（ARR=8）

**抖动模式**

通过使能抖动模式，使用 TIMx_CR1 寄存器中的 DITHEN 位，可以提高 PWM 模式的有效分辨率。这既适用于 CCR（用于增加占空比分辨率），也适用于 ARR（用于增加 PWM 频率分辨率）。

工作原理是在 16 个连续的 PWM 周期中，以预定义的模式使实际的 CCR（或 ARR）值略有变化（增加或减少一个定时器时钟周期）。这使得占空比或 PWM 期间的平均值提高了 16 倍分辨率。下图展示了应用于 4 个连续 PWM 周期的抖动原理。

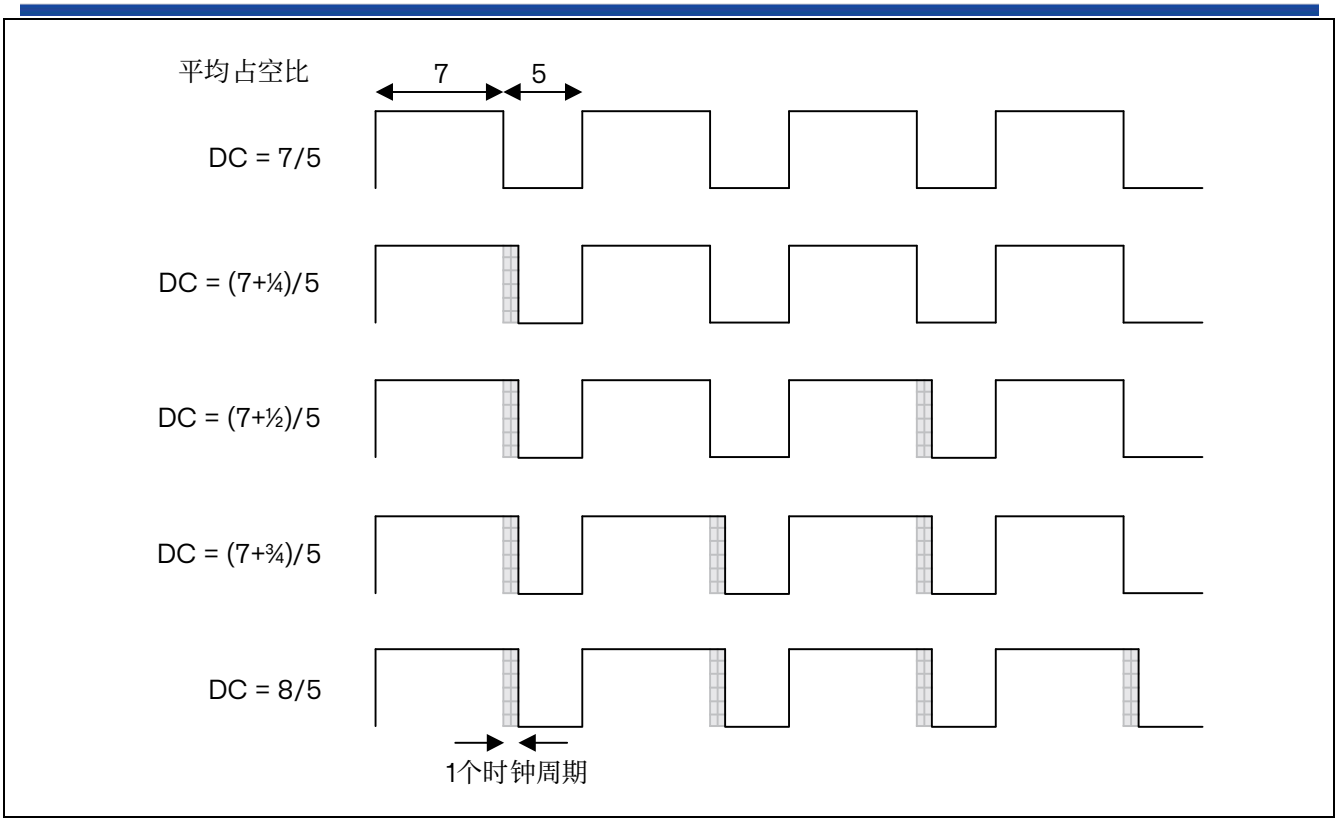



图 24.22 抖动原理

当抖动模式被使能时，寄存器编码将如下所示进行变化（参见下图示例）

- 4 个最低有效位（LSB）用于编码增强的分辨率部分（小数部分）
- 最高有效位（MSB）左移至 19:4 位，用于编码基本值

*注意：如果设置/清除*DITHEN *位，*ARR *和*CCR *值将自动更新。（例如，如果*ARR=0x05 *且*DITHEN=0，*它将更新为*ARR=0x50 *且*DITHEN=1*）。*

1. 必须重置 CEN 和 ARPE 位
2. ARR[3:0]位必须重置
3. DITHEN 位必须重置
4. CCIF 标志必须清除
5. 可以设置 CEN 位（最终设置 ARPE=1）。

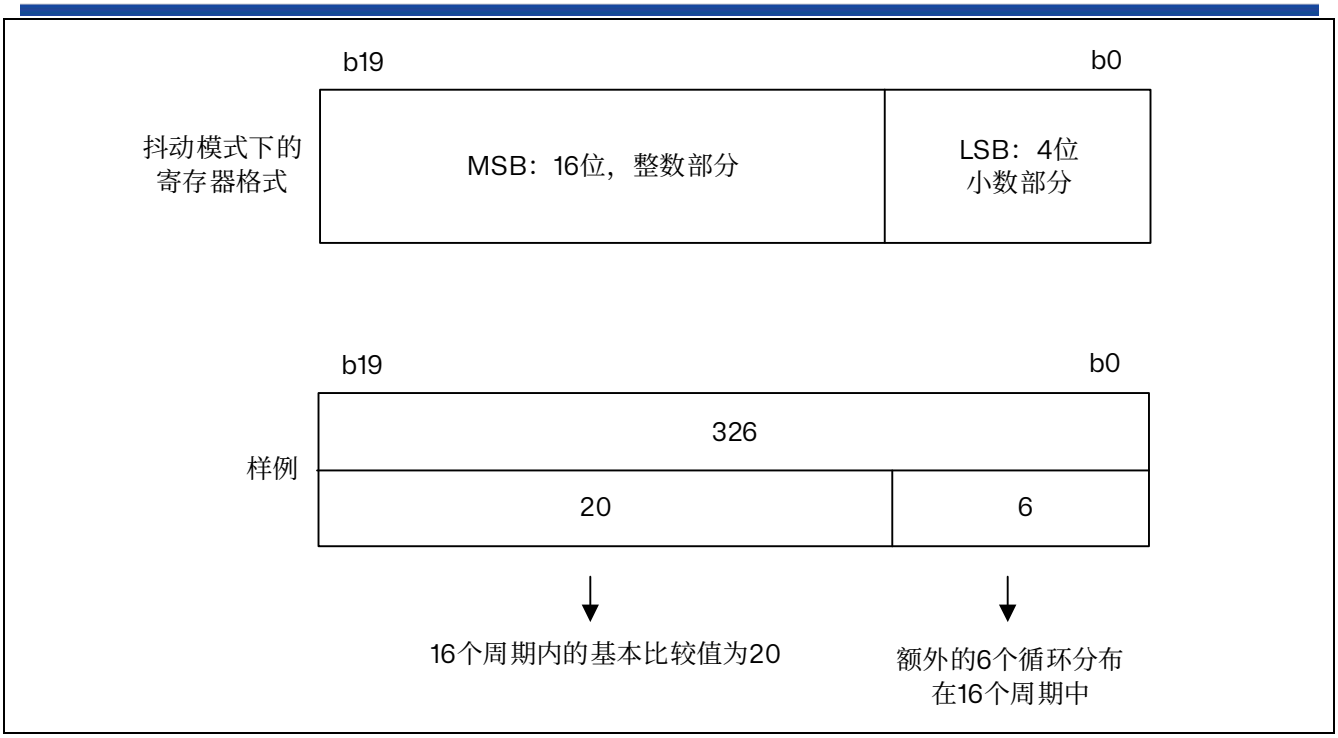



图 24.23 抖动模式下的数据格式和寄存器编码

最小频率由以下公式给出：

$$\text{分辨率} = \frac{\text{FTim}}{\text{Fpwm}} \Rightarrow \text{FpwmMin} = \frac{\text{FTim}}{\text{最大分辨率}}$$

$$\text{禁用抖动模式：FpwmMin} = \frac{\text{FTim}}{65536}$$

$$\text{使能抖动模式：FpwmMin} = \frac{\text{FTim}}{65535 + \frac{15}{16}}$$

*注意：在抖动模式下，*TIMx_ARR *和*TIMx_CCRy *的最大值限制为*0xFFFEF *（对应于整数部分的*65534 *和抖动部分的*15*）。*

如下图所示，无论 PWM 频率如何，抖动模式都可以增加 PWM 分辨率。

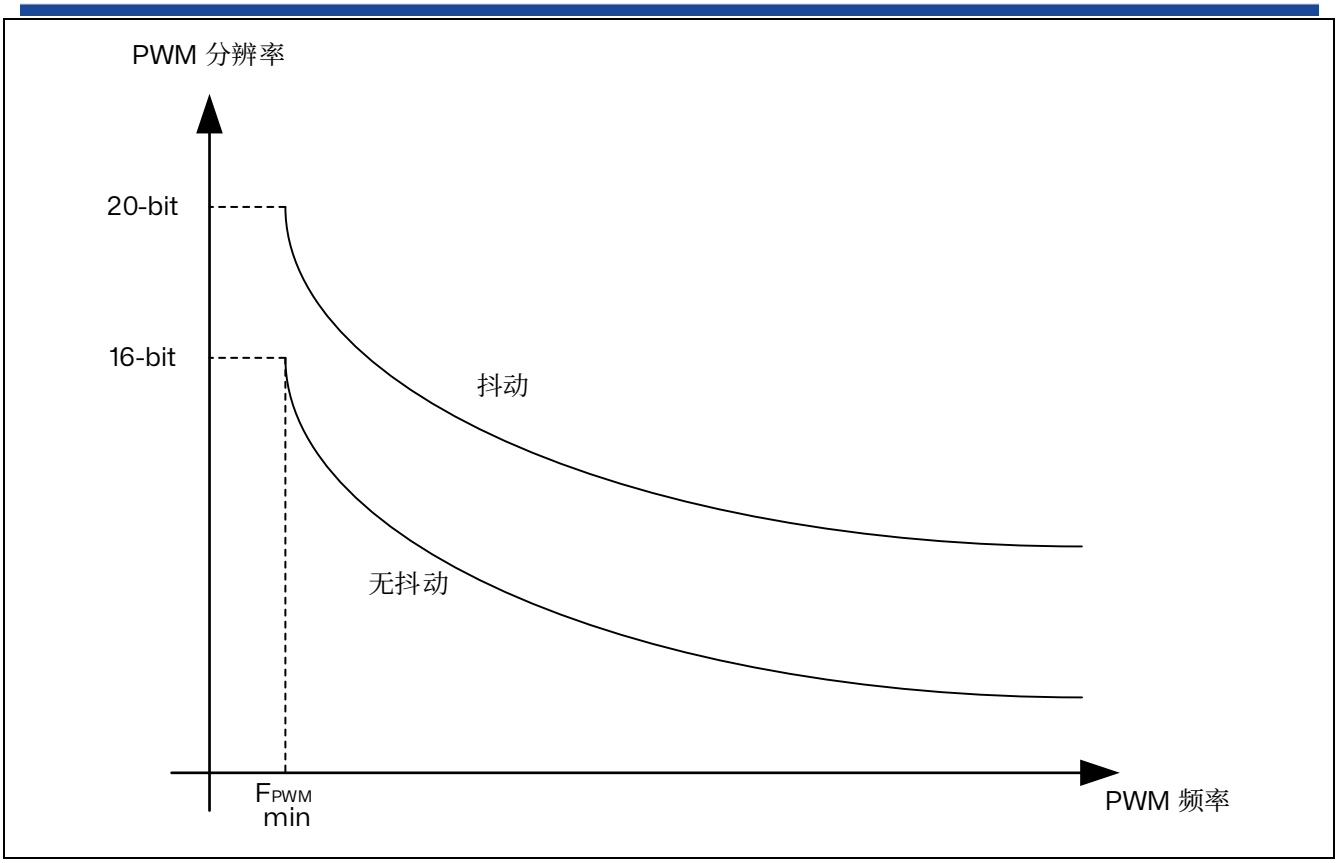



图 24.24 PWM 分辨率 vs 频率

如下图所示，占空比和/或周期的变化分散在 16 个连续周期。

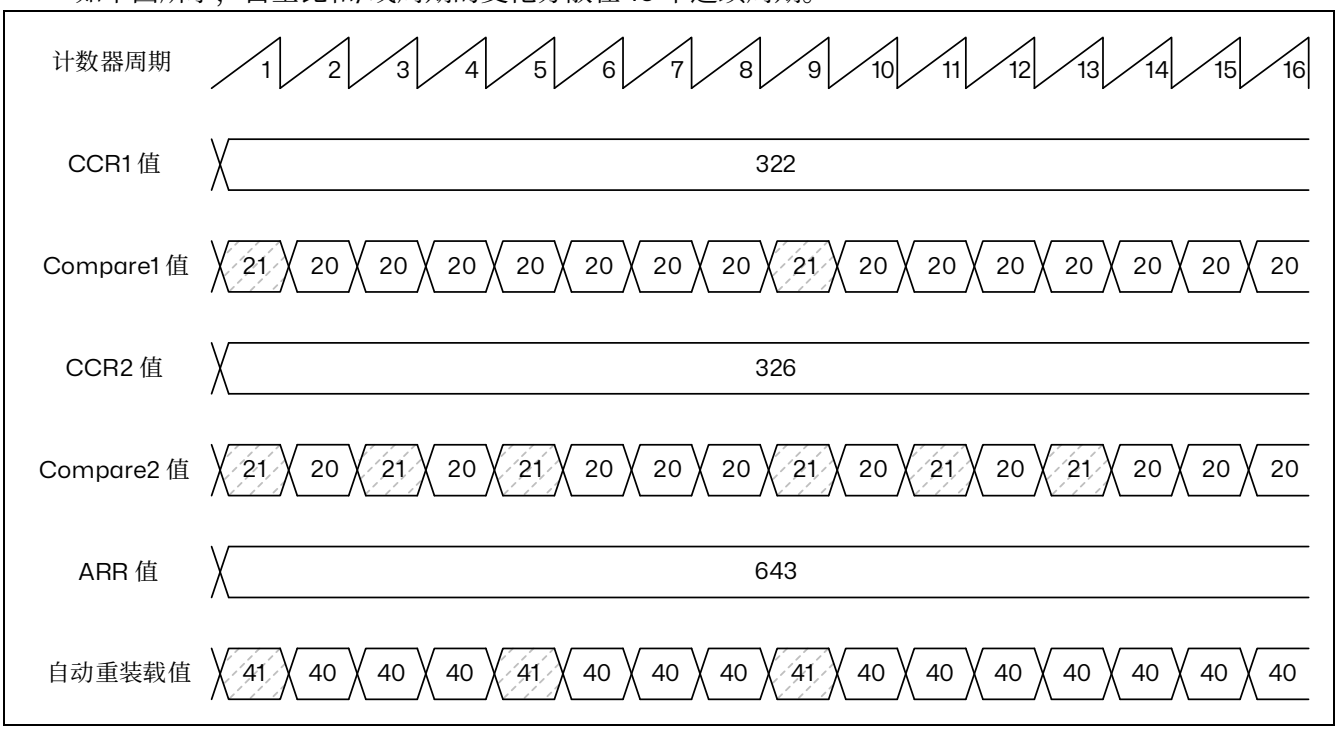



图 24.25 PWM 抖动模式

自动重新加载和比较值的增量按照下表中描述的特定模式分布。抖动序列是为了使增量分布尽可能均

匀并最大限度地减少整体波动。

表 24.9 CCR 和 ARR 寄存器的变化抖动模式

| LSB 值 | 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 | - |

### 24.4.13 组合 PWM 模式（仅 TIM15）

组合 PWM 模式允许生成两个边缘对齐或中心对齐的 PWM 信号，各自脉冲之间的可编程延迟和相位偏移。虽然频率由 TIMx_ARR 寄存器的值决定，但占空比和延迟由两个 TIMx_CCRx 寄存器确定。结果信号 tim_ocxrefc 由两个参考 PWM 的 OR 或 AND 逻辑组合而成：

— tim_oc1refc（或 tim_oc2refc）由 TIMx_CCR1 和 TIMx_CCR2 控制

通过在 TIMx_CCMRx 寄存器的 OCxM 位中写入“1100”（组合 PWM 模式 1）或“1101”（组合 PWM 模式 2），可以在两个通道上独立选组合 PWM 模式（每个 CCR 寄存器对一个 tim_ocx 输出）。

当某个通道用作组合 PWM 通道时，其互补通道必须配置为相反的 PWM 模式（例如，一个通道为组合 PWM 模式 1，另一个通道为组合 PWM 模式 2）。

*注意：为了兼容性原因，*OCxM[3:0]*位域被分为两个部分，最高有效位与最低*3 *位不连续。*

下图表示使用组合 PWM 模式生成信号的示例，该示例的配置如下：

— 通道 1 配置为组合 PWM 模式 2。

— 通道 2 配置为 PWM 模式 1。

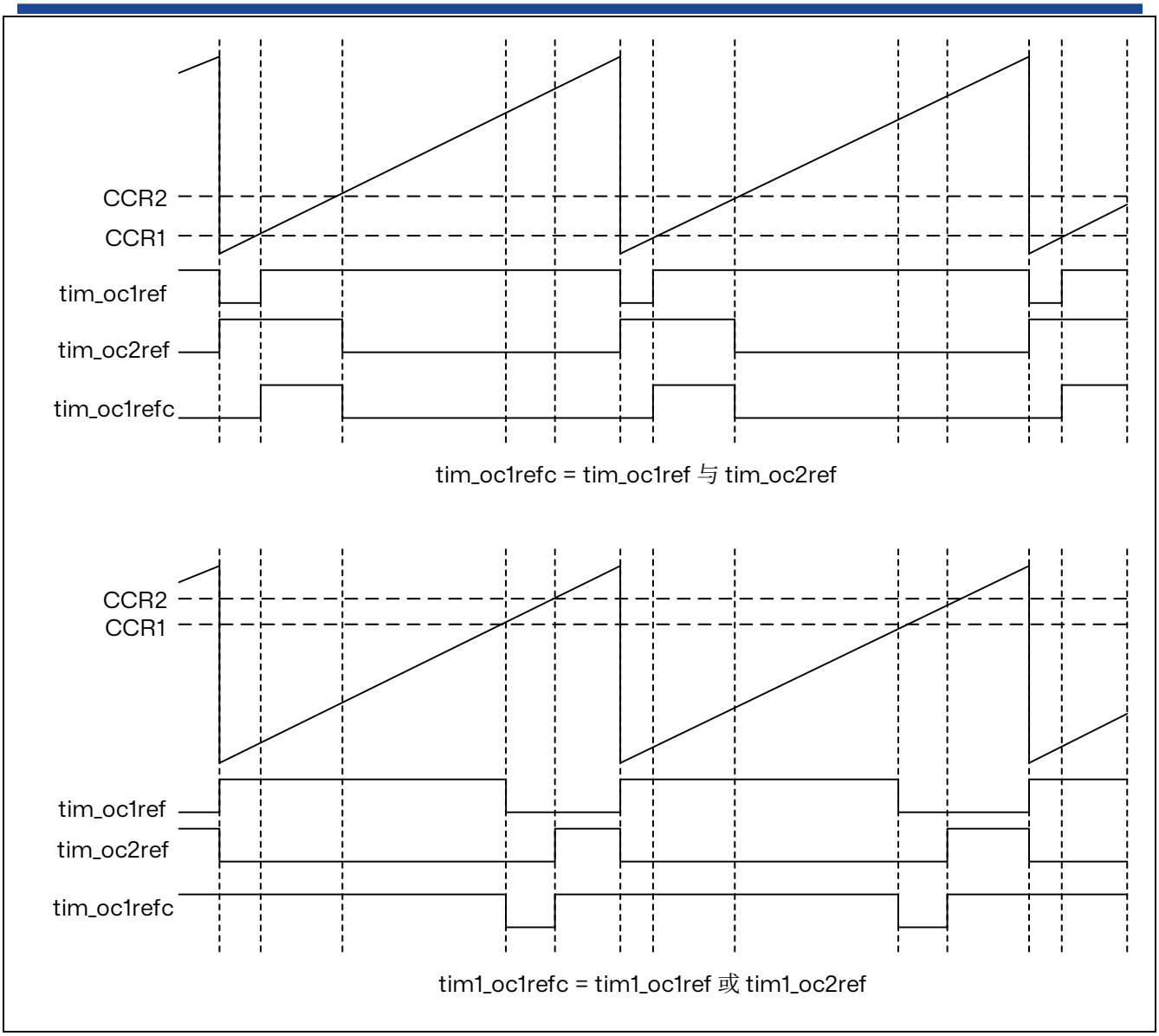



图 24.26 通道 1 和通道 2 的组合 PWM 模式

### 24.4.14 互补输出和死区插入

通用定时器（TIM15/TIM16/TIM17）可以输出一路互补信号，并管理输出的关断与接通瞬间。

这段时间通常称为死区，用户必须根据与输出相连接的器件及其特性（电平转换器的固有延迟、开关器件产生的延迟...）来调整死区时间

每路输出可以独立选择输出极性（主输出 tim_ocx 或互补输出 tim_ocxn）。可通过对 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位执行写操作来完成极性选择。

互补信号 tim_ocx 和 tim_ocxn 通过以下多个控制位的组合进行激活：TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位以及 TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位。更多详细信息，请参考表格：带刹车功能的互补输出通道 tim_oc1 和 tim_oc1n 的控制位（TIM16/TIM17）。应当注意，切换至 idle（MOE 下降到 0）的时刻，死区仍然有效。

CCxE 和 CCxNE 位同时置 1 并且 MOE 位置 1（如果存在刹车）时，将使能死区插入。每个通道都有一个 10 位的死区时间生成器，将基于参考波形 tim_ocxref 生成 2 个输出 tim_ocx 和 tim_ocxn。如果

tim_ocx 和 tim_ocxn 为高电平有效：

- 输出信号 OCx 与参考信号相同，只是其上升沿相对参考上升沿存在延迟。
- 输出信号 OCxN 与参考信号相反，并且其上升沿相对参考下降沿存在延迟。

如果延迟时间大于有效输出（OCx 或 OCxN）的宽度，则不会产生相应的脉冲。

下图所示为死区发生器的输出信号与参考信号 OCxREF 之间的关系。（在这些示例中，假定 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

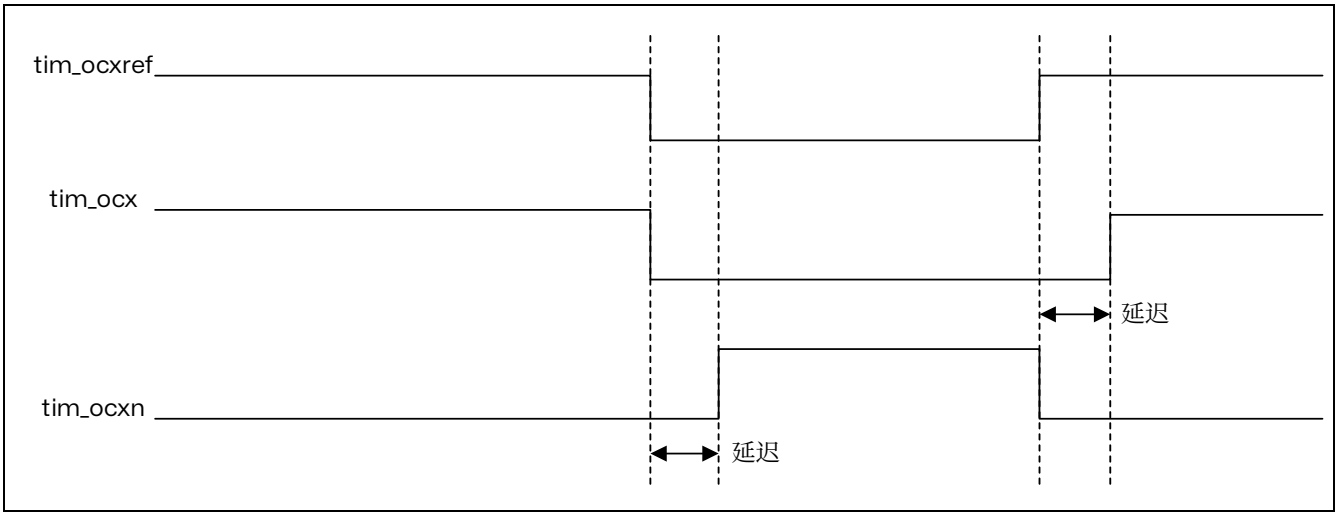



图 24.27 带死区插入的互补输出

TIMx_DTR2 中的 DTAE 位允许对参考信号上升沿和下降沿的死区值进行区分，如下图所示。

在非对称模式（DTAE=1）下，上升沿参考死区由 TIMx_BDTR 寄存器中的 DTG[7:0]位域定义，而下降沿参考死区由 TIMx_DTR2 寄存器中的 DTGF[7:0]位域定义。在使能计数器之前，必须写入 DTAE 位，并且在 CEN=1 时不得修改。

使用预加载机制可以在 PWM 操作期间动态更新死区值。当 TIMx_DTR2 寄存器中的 DTPE 位被设置时，死区位域 DTG[7:0]和 DTGF[7:0]将被预加载。预加载值将在下一个更新事件上加载到活动寄存器中。

*注：如果 DTPE 位在计数器使能时被使能。自上次更新以来写入的新值将被丢弃，并使用先前的值。*

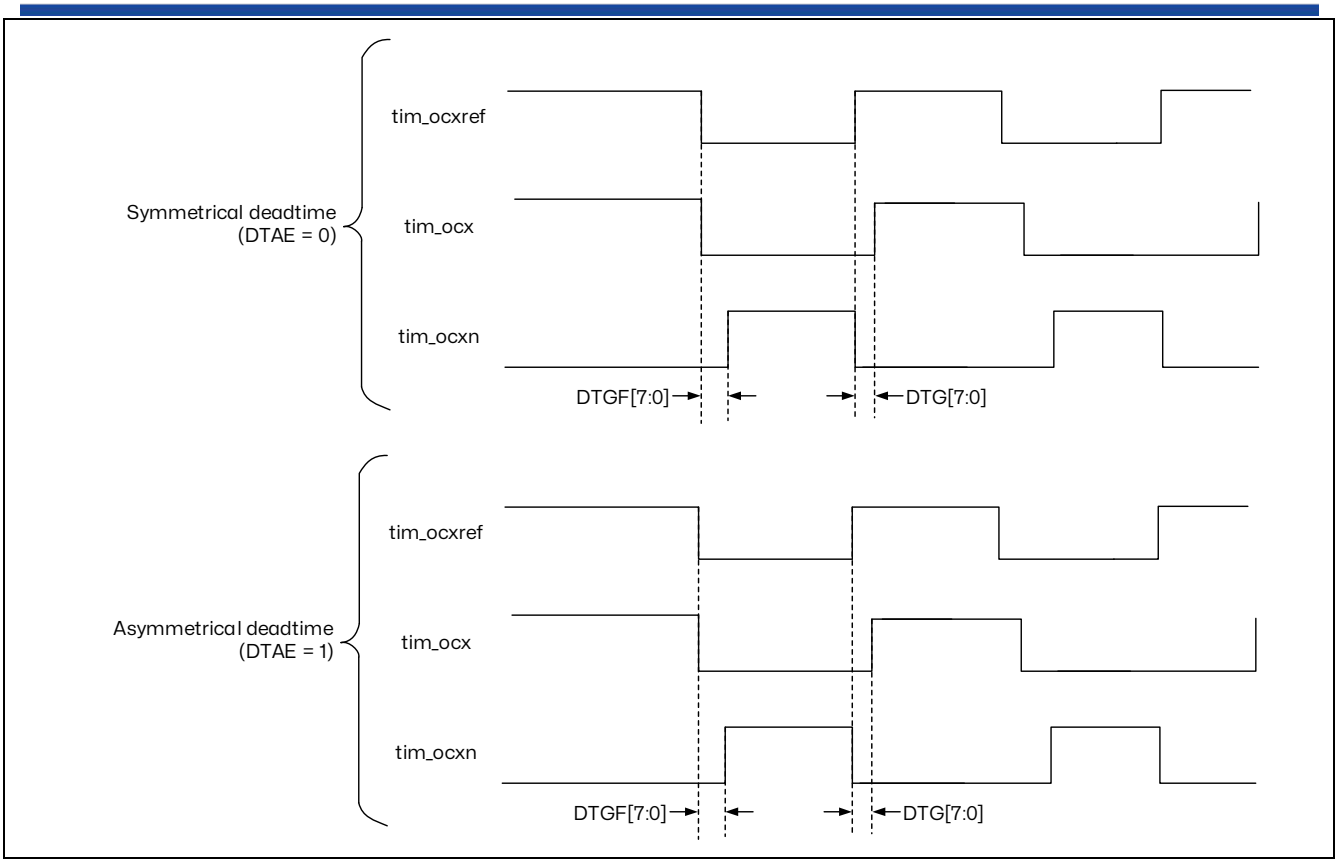



图 24.28 非对称死区时间

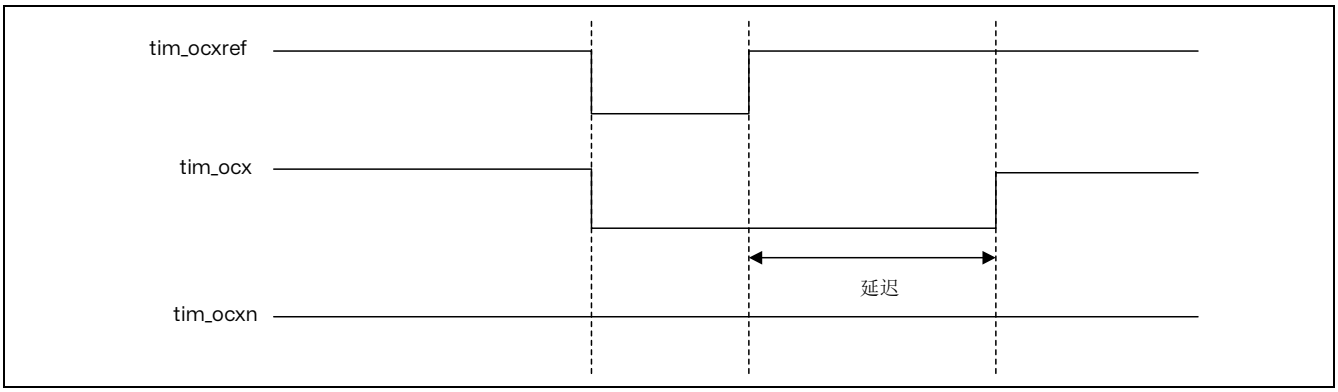



图 24.29 死区波形延迟大于负脉冲

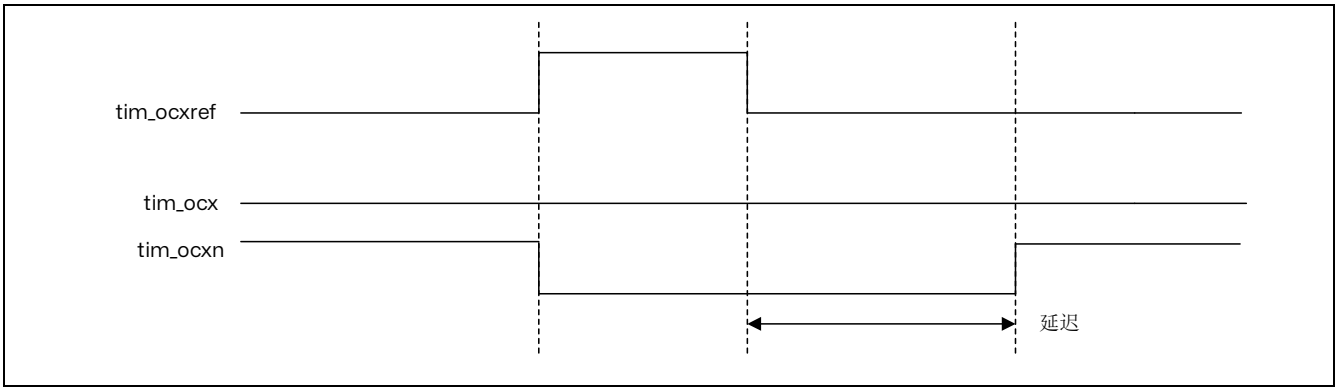



图 24.30 死区波形延迟大于正脉冲

死区延迟对于所有通道均相同，可通过 TIMx_BDTR 寄存器中的 DTG 位进行编程。有关延迟时间计算的信息，请参见后续章节：TIMx 刹车和死区寄存器(TIMx_BDTR)（x = 16~17）。

**将 OCxREF 重定向到 OCx 或 OCxN**

在输出模式（强制输出模式、输出比较模式或 PWM 模式）下，通过配置 TIMx_CCER 寄存器中的 CCxE 和 CCxNE 位，可将 tim_ocxref 重定向到 tim_ocx 输出或 tim_ocxn 输出。通过此功能，可以在一个输出上发送特定波形（如 PWM 或静态有效电平），而同时使互补输出保持其无效电平。或者，使两个输出同时保持无效电平，或者两个输出同时处于有效电平，两者互补并且带死区。

*注：如果仅使能* tim_ocxn *（*CCxE=0, CCxNE=1*），两者不互补，一旦* tim_ocxref *为高电平，*tim_ocxn *即变为有效。例如，如果* CCxNP=0，*则*tim_ocxn= tim_ocxref*。另一方面，如果同时使能* tim_ocxn *和*tim_ocxref *（*CCxE=CCxNE=1*），*tim_ocx *在*tim_ocxref *为高电平时变为有效，而*tim_ocxn *则与之互补，在*tim_ocxref *为低电平时变为有效。*

### 24.4.15 使用刹车功能

刹车功能的目的是保护由定时器产生的 PWM 信号驱动的功率开关。刹车输入通常连接到功率级和三相逆变器的故障输出。当激活时，刹车电路会关闭 PWM 输出并强制它们进入预定义的安全状态。

刹车通道可以收集系统级故障（时钟故障、ECC/奇偶校验错误，.....）和应用故障（来自输入引脚和内置比较器），并可以在死区时间后将输出强制到一个预定义的水平（无论是有效还是无效）。

刹车期间输出使能信号和输出电平取决于几个控制位：

- TIMx_BDTR 寄存器中的 MOE 位允许通过软件使能/禁用输出，并在发生刹车事件时重置。
- TIMx_BDTR 寄存器中的 OSSI 位定义了定时器是否在无效状态下控制输出或将其控制权释放给 GPIO 控制器（通常使其处于高阻模式）。
- TIMx_CR2 寄存器中的 OISx 和 OISxN 位用于设置输出关闭电平，可以是有效电平或无效电平。tim_oc1 和 tim_oc1n 输出不能同时设为有效电平。更多详细信息请参考后续表格：具有刹车功能的互补 tim_oc1 和 tim_oc1n 通道的输出控制位。

退出复位时，刹车电路被禁用，MOE 位为低电平。通过设置 TIMx_BDTR 寄存器中的 BKE 位，可以使能刹车功能。通过配置同一寄存器中的 BKP 位，可以选择刹车输入极性。BKE 和 BKP 可以同时修改。在写入 BKE 和 BKP 位后，需要等待 1 个 APB 时钟周期才能使写入生效。因此，在写入操作后，需要等待 1 个 APB 时钟周期才能正确地读取该位。

因为 MOE 下降沿可以是异步的，所以在实际的信号（作用于输出）和同步控制位（在 TIMx_BDTR 寄存器中访问）之间插入了一个重新同步电路。这导致异步信号和同步信号之间存在一些延迟。特别是当 MOE 被设置为 1 而之前它是低电平时，在正确读取之前必须插入一个延迟（伪指令）。这是因为写入作用于异步信号，而读取反映的是同步信号。

刹车是由 tim_brk 输入产生的，它具有：

- 可编程极性（TIMx_BDTR 寄存器中的 BKP 位）
- 可编程使能位（TIMx_BDTR 寄存器中的 BKE 位）
- 可编程滤波器（TIMx_BDTR 寄存器中的 BKF[3:0]位），以避免产生虚假事件。

可以使用 TIMx_AF1 寄存器，从多个源生成刹车，这些源可以单独使能，并具有可编程的边沿敏感度。

刹车（tim_brk）通道的源为：

- 连接到 TIMx_BKIN 引脚的外部源（根据 GPIO 备用功能选择寄存器中的选择而定），具有极性选择和可选的数字滤波

- 内部源：
  - 来自 tim_brk_cmpx 输入（请参考章节：TIM15/TIM16/TIM17 引脚和内部信号）
  - 来自 tim_sys_brk 输入的系统中断请求（请参考章节：TIM15/TIM16/TIM17 引脚和内部信号）

也可以通过软件使用 TIMx_EGR 寄存器中的 BG 位生成刹车事件。

所有源在进入定时器的 tim_brk 输入之前进行 OR 操作，如下图所示。

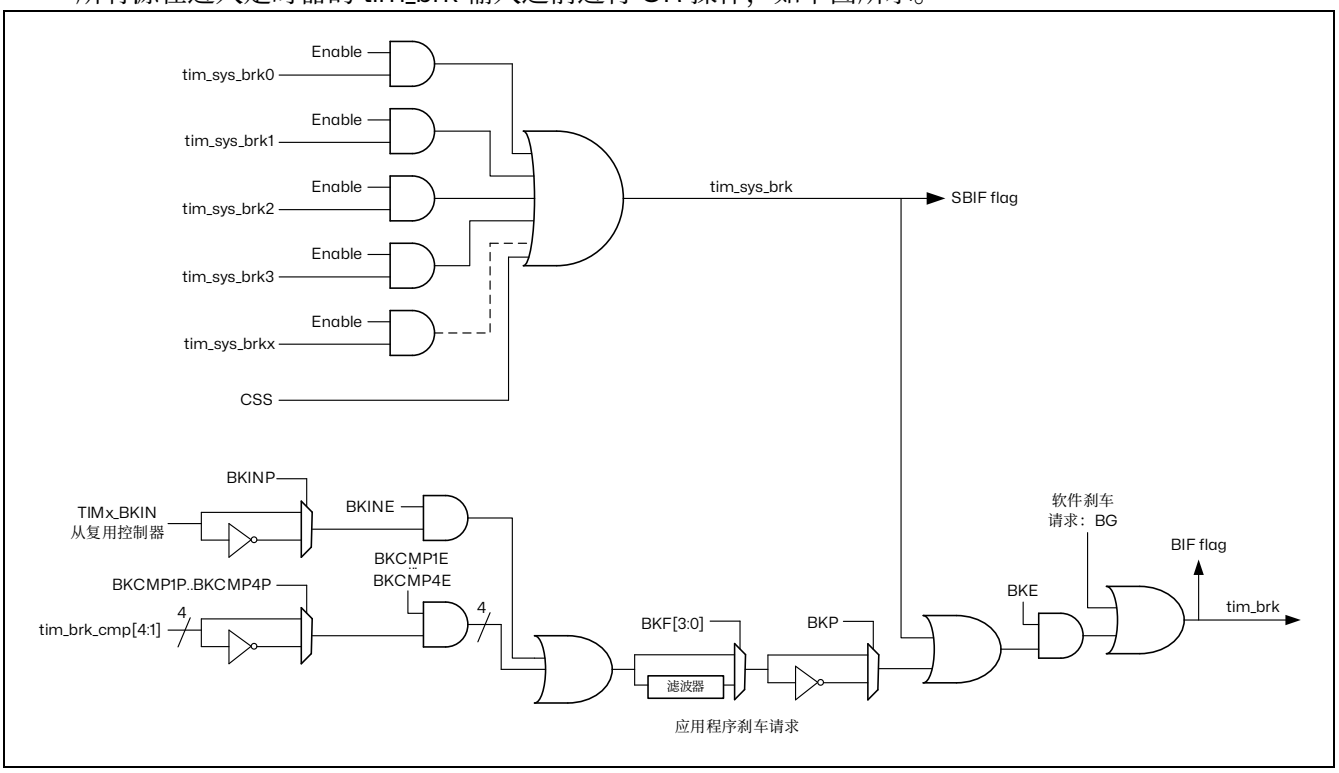



图 24.31 刹车电路概述

*注：只有当可编程滤波器禁用时，才保证异步（无时钟）操作。如果使能，必须使用故障安全时钟模式（例如使用内部*PLL *和/或*CSS*）来保证刹车事件得到处理。*

当一个刹车发生时（在刹车输入上选择电平）：

- MOE 位被异步清除，使输出处于无效状态、空闲状态，甚至释放控制权给 GPIO（由 OSSI 位选择）。即使 MCU 振荡器关闭，此功能也可工作。
- 每当 MOE=0 时，每个输出通道都以 TIMx_CR2 寄存器中的 OISx 位中编程的电平驱动。如果 OSSI=0，定时器将释放输出控制（由 GPIO 接管），否则使能输出保持高电平。
- 当使用互补输出时：
  - 输出首先被置于复位状态（取决于极性），这是异步完成的，因此即使没有向定时器提供时钟也可以工作。
  - 如果定时器的时钟仍然存在，则重新激活死区时间发生器，以便在死区时间后驱动输出到编程在 OISx 和 OISxN 位中的电平。即使在这种情况下，tim_ocx 和 tim_ocxn 也不能同时驱动到有效电平。请注意，由于在 MOE 上进行了重新同步，死区时间持续时间比通常情况稍长（大约 2 个 tim_ker_ck 时钟周期）。
  - 如果 OSSI=0，则定时器释放使能输出（由 GPIO 接管，强制进入高阻态），否则使能输出在 CCxE 或 CCxNE 位中的任意一位变为高电平时立即变为高电平。
- 刹车状态标志（TIMx_SR 寄存器中的 BIF 位）被设置。如果 TIMx_DIER 寄存器中的 BIE 位被设

置，则可以生成刹车。

- 如果 TIMx_BDTR 寄存器中的 AOE 位被设置，则 MOE 位将在下一个更新事件 UEV 处自动重新设置。这可以用于执行调节等操作。否则，MOE 保持低电平，直到再次写入 1。在这种情况下，它可以用于安全，刹车输入可以连接到来自功率驱动器、温度传感器或任何安全组件的警报。

*注：如果 MOE 被 CPU 复位，而 AOE 位已设置，则输出处于空闲状态，并强制处于非活动电平或高阻态，具体取决于 OSSI 值。如果 MOE 和 AOE 位都被 CPU 复位，则输出处于禁用状态，并驱动在 TIMx_CR2 寄存器中 OISx 位编程的电平。*

*注：刹车输入在电平上有效。因此，当刹车输入处于有效状态时（无论是自动还是通过软件设置），MOE 都无法设置。同时，状态标志 BIF 不能被清除。*

除刹车输入和输出管理外，刹车电路内部还实施了写保护，用以保护应用的安全。通过该功能，用户可冻结多个参数配置（死区持续时间、tim_ocx/tim_ocxn 极性和禁止时的状态、OCxM 配置、刹车使能和极性）。可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从 3 种保护级别中进行选择。请参考章节：TIMx 刹车和死区寄存器（TIMx_BDTR）。MCU 复位后只能对 LOCK 位执行一次写操作。

下图所示为输出对刹车响应行为的示例。

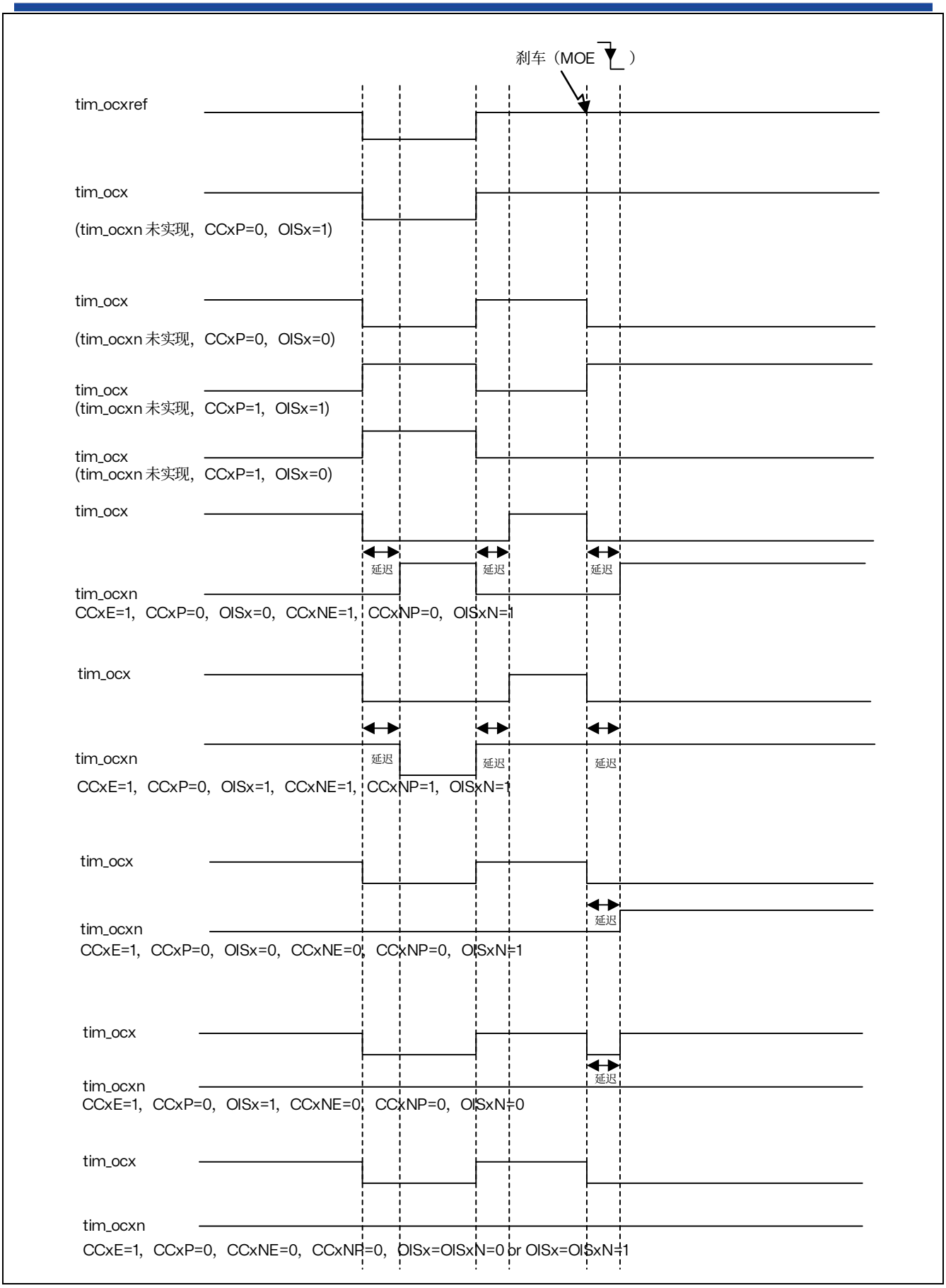



图 24.32 tim_brk 在发生刹车事件时的各种输出行为（OSSI=1）

### 24.4.16 双向刹车输入

TIM15/TIM16/TIM17 具有双向刹车 I/O，如下图所示。

这为以下方面提供支持：

- 板级全局刹车信号，用于向外部 MCU 或门驱动器发送故障信号，使用唯一的引脚作为输入和输出状态引脚
- 内部刹车源和多个外部开漏源 OR 连接在一起，以触发唯一的刹车事件，当多个内部和外部刹车源必须合并时

使用 TIMxBDTR 寄存器中的 BKBID 位配置 tim_brk 输入为双向模式。BKBID 编程位可以使用 TIMxBDTR 寄存器中的 LOCK 位锁定为只读模式（在 LOCK 级别 1 或更高）。

双向模式要求 I/O 配置为开漏模式，使用低电平有效极性（使用 BKINP、BKP 位）。来自系统（例如 CSS）、片上外设或刹车输入的任何刹车请求都会强制将刹车输入置于低电平，以指示故障事件。出于安全原因，如果极性位没有正确设置（高电平有效），则双向模式将被抑制。

刹车软件事件（通过设置 BG 位触发）也会强制将刹车 I/O 置为“0”，以指示外部组件定时器已进入刹车状态。然而，只有使能了刹车（BKE = 1）时，这才有效。当使用 BKE = 0 生成软件刹车事件时，输出将处于安全状态，并且会设置刹车标志，但 TIMx_BKIN I/O 没有影响。

安全解发机制可防止系统被彻底锁定（刹车输入低电平触发刹车，强制该输入为低电平）。

当 BKDSRM 位设置为 1 时，这将释放刹车输出以清除故障信号并允许重新给系统装载。

在任何时候都不能禁用刹车保护电路：

- 刹车输入路径始终处于有效状态：即使设置了 BKDSRM 位并且开放漏极控制被释放，刹车事件仍然处于有效状态。这可以防止 PWM 输出在刹车条件存在时重新启动。
- 只要输出使能（MOE 位被设置），BKDSRM 位就不能解除刹车保护（参见下表）

表 24.10 刹车保护解除条件

| MOE | BKBID | BKDSRM | 刹车保护状态 |
|---|---|---|---|
| 0 | 0 | X | 装载 |
| 0 | 1 | 0 | 装载 |
| 0 | 1 | 1 | 解除 |
| 1 | X | X | 装载 |

**装载和重装载刹车电路**

默认情况下（外设复位配置），刹车电路（输入或双向模式）处于装载状态。

在发生刹车事件后，必须按照以下步骤重新装载保护：

- BKDSRM 位必须设置为释放输出控制
- 软件必须等待系统刹车条件消失，并清除 SBIF 状态标志（或在重新装载之前系统地清除它）
- 软件必须轮询 BKDSRM 位，直到硬件清除该位（当应用程序刹车条件消失时）

从这一点开始，刹车电路处于装载和有效状态，并且可以设置 MOE 位以重新使能 PWM 输出。

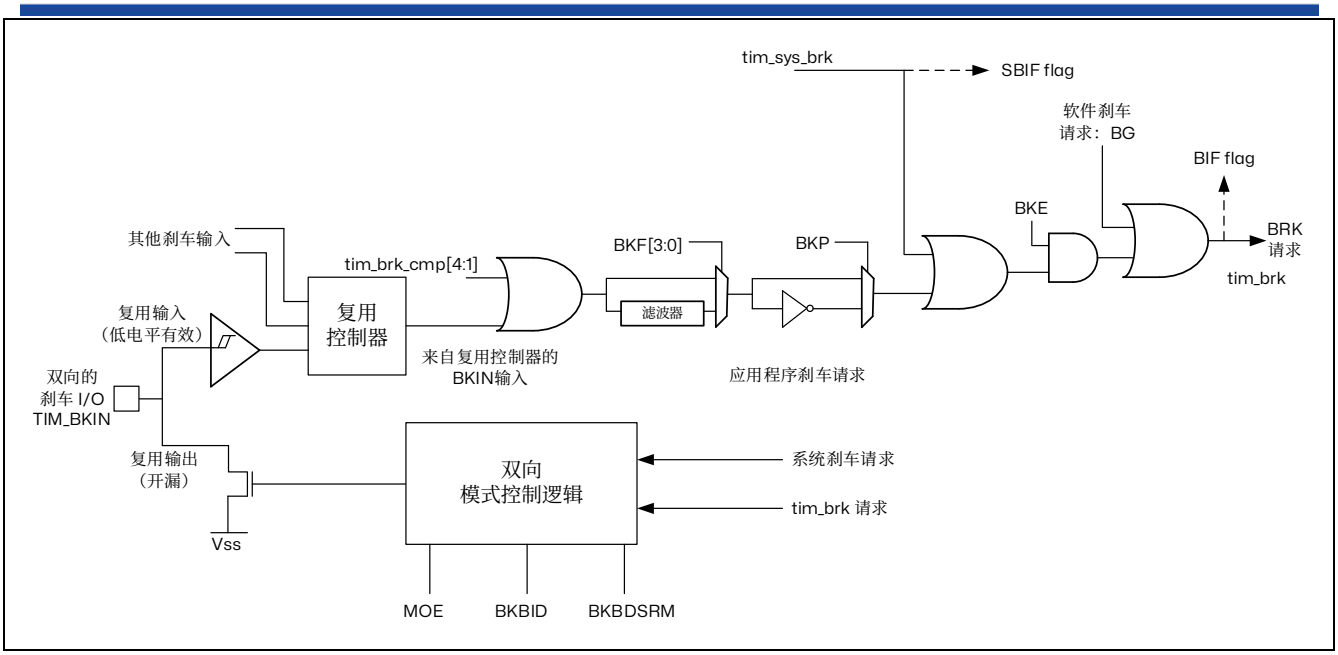



图 24.33 输出重定向

### 24.4.17 在外部事件时清除 tim_ocxref 信号

对于给定通道，在 tim_ocref_clr_int 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 tim_ocxref 信号变为低电平。tim_ocxref 信号将保持低电平，直到发生下一更新事件(UEV)（特别注意，当 TIMx_RCR>0 时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。

当配置组合 PWM 模式（仅 TIM15），对于 TIM15_CH1，在 tim_ocref_clr_int 输入施加高电平（相应 TIM15_CCMR1 寄存器中的 OC1CE 使能位置“1”），可使组合结果 tim_oc1refc 信号变为低电平。tim_oc1refc 信号将保持低电平，直到发生下一更新事件（UEV）（特别注意，当 TIM15_RCR>0 时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。

tim_ocref_clr 输入可以在多个输入中进行选择，如下图所示。

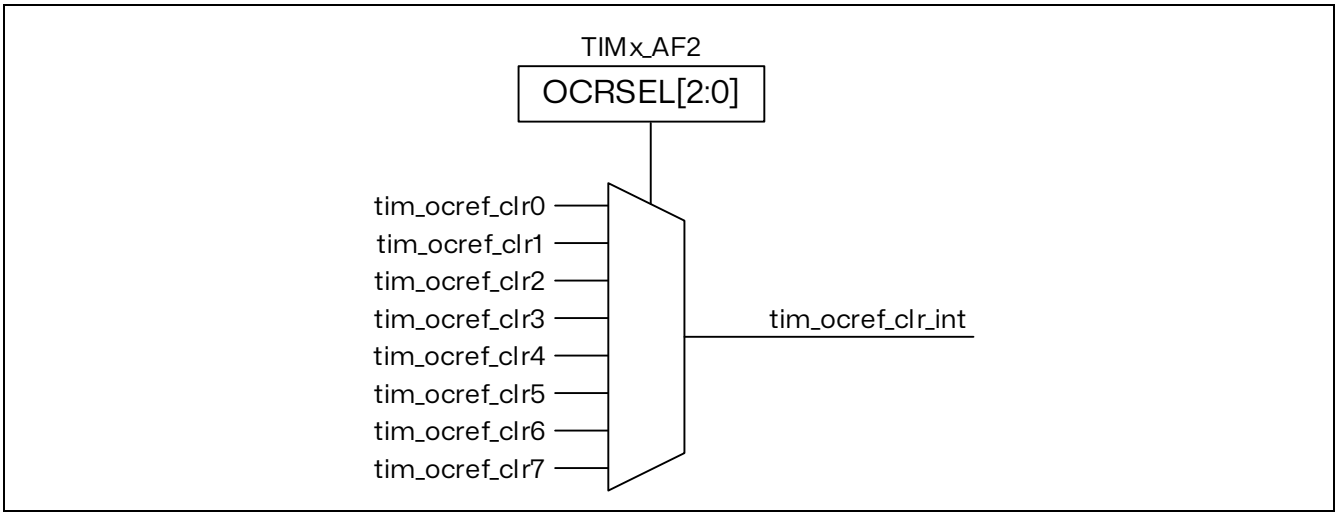



图 24.34 tim_ocref_clr 输入选择复用器

### 24.4.18 单脉冲模式

单脉冲模式（OPM）是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- CNT < CCRx ≤ ARR（特别注意，0<CCRx）

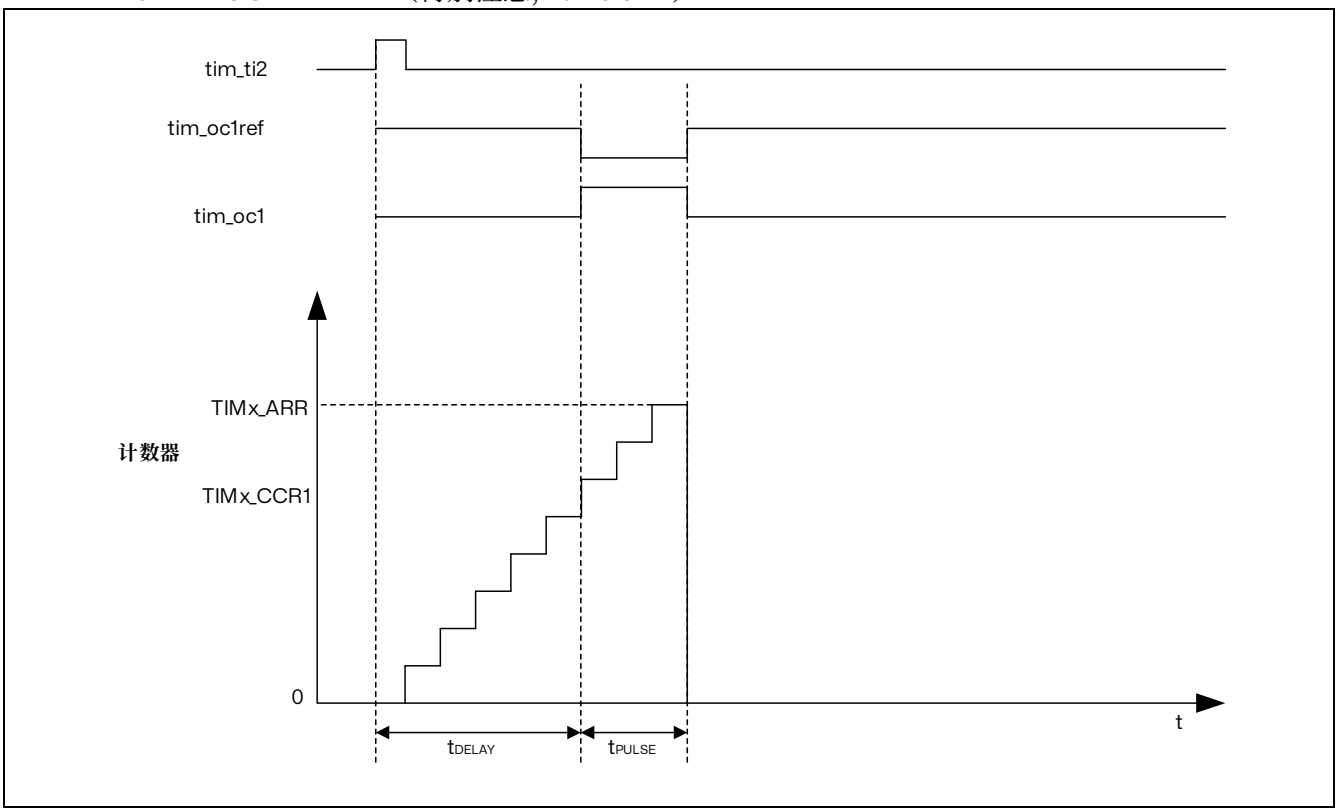



图 24.35 单脉冲模式的例子

例如，用户希望达到这样的效果：在 tim_ti2 输入引脚检测到正沿时，经过 $t_{DELAY}$ 的延迟，在 tim_oc1 上产生一个长度为 $t_{PULSE}$ 的正脉冲。

使用 tim_ti2fp2 作为触发 1:

1. 通过在 TIMx_TISEL 寄存器中使用 TI2SEL[3:0]位选择合适的 tim_ti2_in[15:0]源（内部或外部）。
2. 在 TIMx_CCMR1 寄存器中写入 CC2S=01，将 tim_ti2fp2 映射到 tim_ti2。
3. 在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=“0”，使 tim_ti2fp2 能够检测上升沿。
4. 在 TIMx_SMCR 寄存器中写入 TS=00110，将 tim_ti2fp2 配置为从模式控制器的触发(tim_trgi)。
5. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），使用 tim_ti2fp2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- $t_{DELAY}$ 由写入 TIMx_CCR1 寄存器的值定义。
- $t_{PULSE}$ 由自动重载值与比较值（TIMx_ARR - TIMx_CCR1）之差来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式

2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 tim_ti2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位清“0”时，即选择重复模式。

**特例：OCx 快速使能：**

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（$t_{DELAY}$ 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 tim_ocxref（和 tim_ocx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用

### 24.4.19 可重触发单脉冲模式（仅 TIM15）

此模式允许计数器在接收到刺激后开始计数，并生成一个具有可编程长度的脉冲，但与上一节中描述的非重触发单脉冲模式存在以下差异：

— 只要触发器发生，脉冲就会立即开始（没有可编程的延迟）

— 如果在先前触发的脉冲完成之前发生新的触发，则脉冲会被延长

定时器必须处于从模式，TIMx_SMCR 寄存器的 SMS[3:0]位设置为“1000”（组合重置+触发模式），OCxM[3:0]位设置为“1000”或“1001”，用于可重触发 OPM 模式 1 或 2。

如果定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）。如果定时器配置为递减计数模式，CCRx 必须大于或等于 ARR。

*注：OCxM[[3:0]和 SMS[3:0]位域由于兼容性的原因被分成两个部分，最高有效位与最低 3 位不是连续的。OCxM 这种模式不能与中央对齐 PWM 模式一起使用。TIMx_CR1 寄存器中的 CMS[1:0]必须设置为 00。*

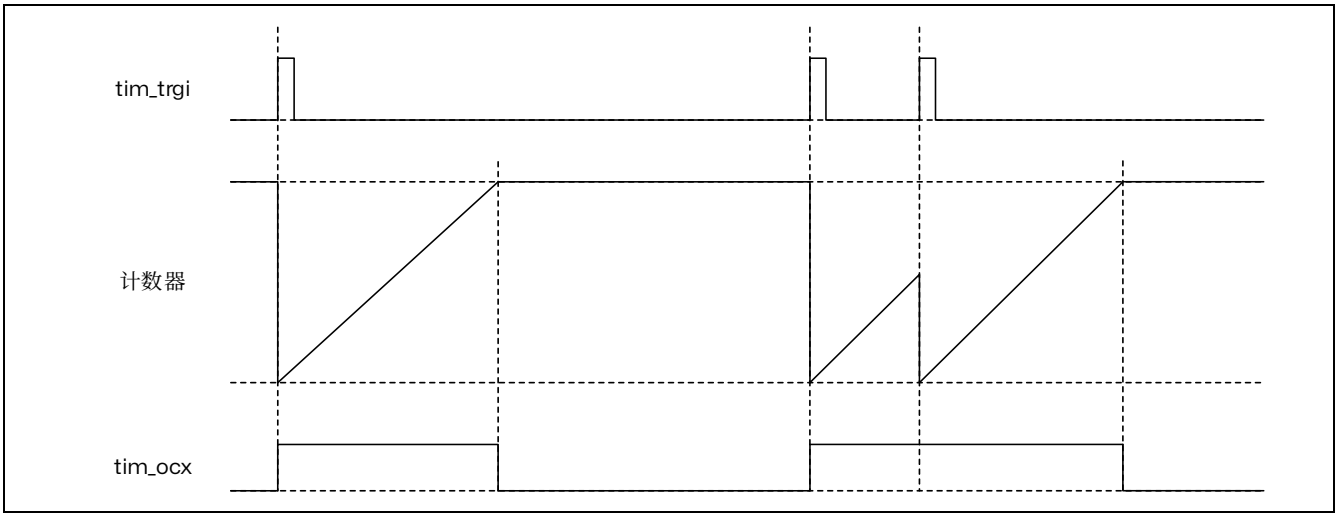



图 24.36 可重触发单脉冲模式

### 24.4.20 UIF 位重映射

TIMx_CR1 寄存器中的 UIFREMAP 位强制将更新中断标志 UIF 连续复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许以原子方式读取计数器的值和一个潜在的由 UIFCPY 标志指示的翻转条件。在特定情况下，它可以避免由背景任务（读取计数器）和中断（更新中断）之间的处理引起的竞争条件，从而简化计算。

UIF 和 UIFCPY 标志之间没有延迟。

### 24.4.21 定时器输入异或功能（仅 TIM15）

TIM15_CR2 寄存器中的 TI1S 位允许将通道 1 的输入滤波器连接到异或门的输出，将两个输入引脚 tim_ti1 和 tim_ti2 组合起来。

异或输出可以与所有定时器输入功能一起使用，例如触发器或输入捕获。它对于测量两个输入信号之间的边沿间隔非常有用，如下图所示。

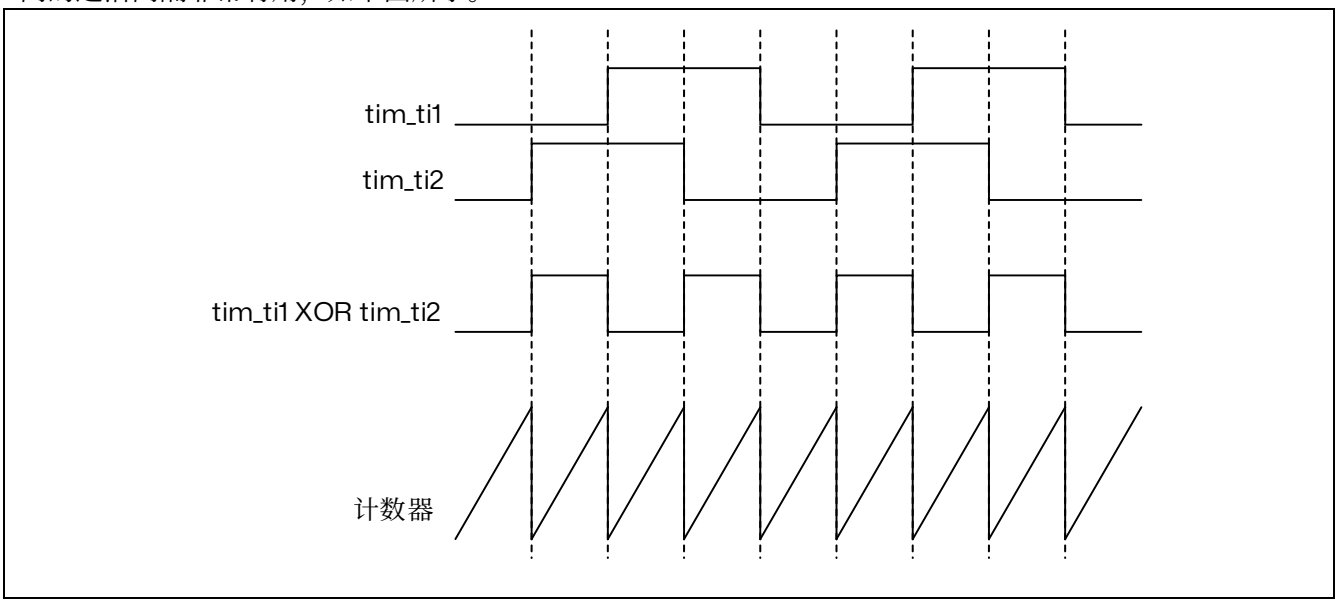



图 24.37 测量两个信号边沿之间的间隔时间

### 24.4.22 定时器和外部触发的同步（仅 TIM15）

TIM 定时器在互相连接用于定时器的同步或链接。

TIM15 定时器以下列模式与外部触发实现同步：复位模式、门控模式、触发模式、复位+触发和门控+复位模式。

**从模式：复位模式**

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIM15_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIM15_ARR，TIM15_CCRx）都被更新了。

在以下的例子中，tim_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 tim_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 TIM15_CCMR1 寄存器中 CC1S=01。置 TIM15_CCER 寄存器中 CC1P=0 和 CC1NP=0 以确定极性（只检测上升沿）。
- 置 TIM15_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIM15_SMCR 寄存器中 TS=00101，选择 tim_ti1 作为输入源。
- 置 TIM15_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 tim_ti1 出现一个上升沿；当 tim_ti1 上升时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TIM15_SR 寄存器中的 TIF 位）被设置，中断请求或 DMA 请求可以被发送（取决于 TIM15_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示当自动重装载寄存器 TIM15_ARR=0x36 时的动作。

在 tim_ti1 上升沿和计数器的实际复位之间的延时取决于 tim_ti1 输入端的重同步电路。

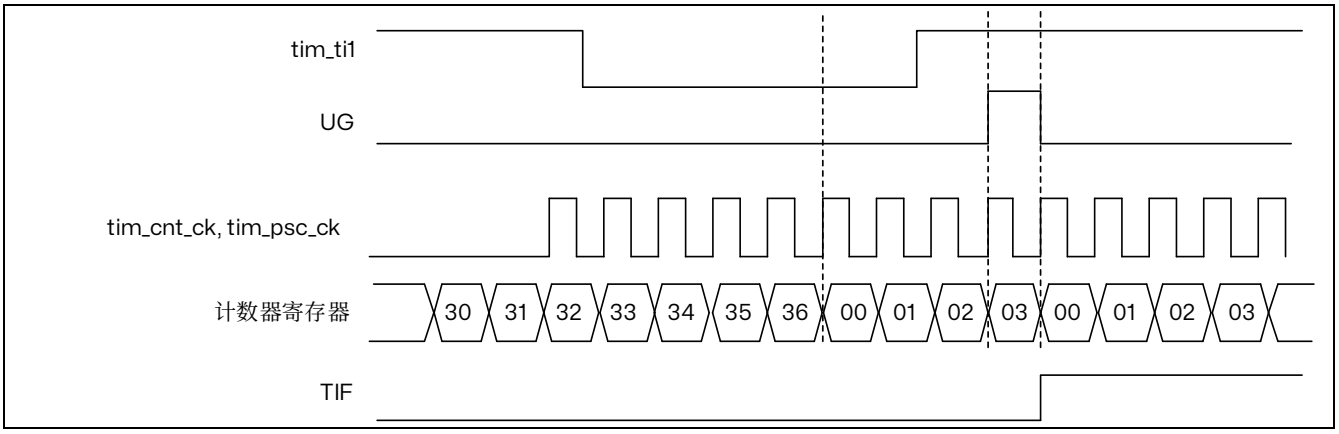



图 24.38 复位模式下的控制电路

**从模式：门控模式**

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 tim_ti1 为低时向上计数：

- 配置通道 1 以检测 tim_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 TIM15_CCMR1 寄存器中 CC1S=01。置 TIM15_CCER 寄存器中 CC1P=1 和 CC1NP=0 以确定极性（只检测低电平）。
- 置 TIM15_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIM15_SMCR 寄存器中 TS=00101，选择 tim_ti1 作为输入源。
- 置 TIM15_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 tim_ti1 为低，计数器开始依据内部时钟计数，一旦 tim_ti1 变高则停止计数。当计数器开始或停止时都设置 TIM15_SR 中的 TIF 标置。

tim_ti1 上升沿和计数器实际停止之间的延时取决于 tim_ti1 输入端的重同步电路。

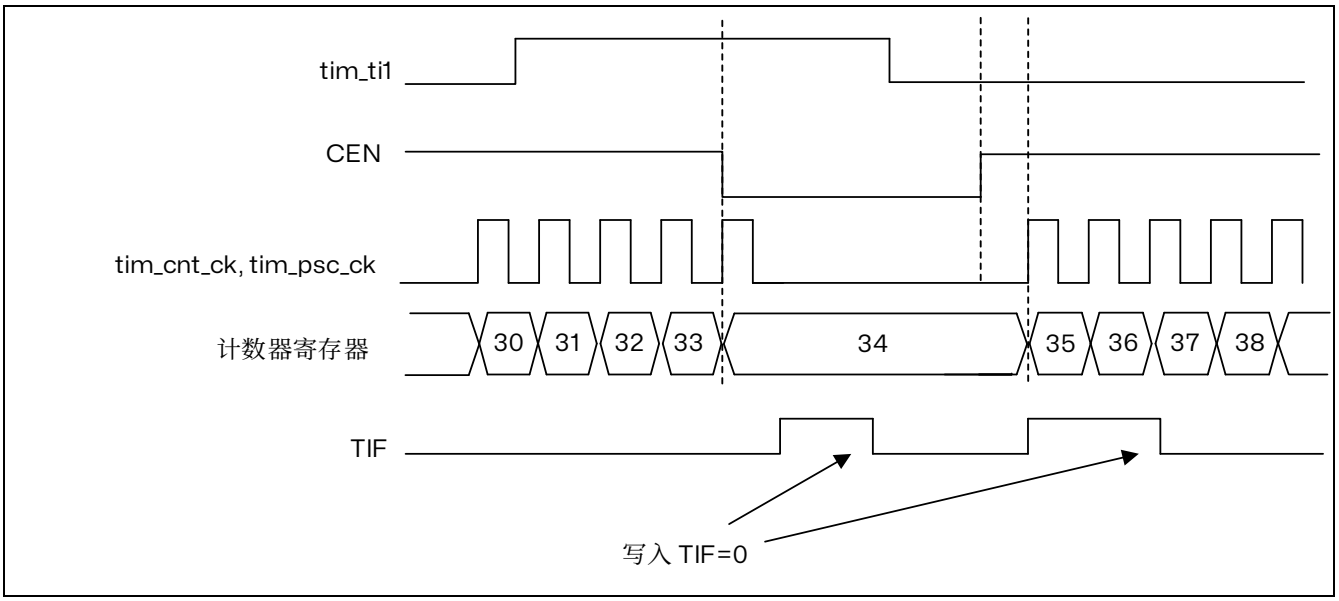



图 24.39 门控模式下的控制电路

**从模式：触发模式**

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 tim_ti2 输入的上升沿开始向上计数：

- 配置通道 2 检测 tim_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 TIM15_CCMR1 寄存器中 CC2S=01。置 TIM15_CCER 寄存器中 CC2P=1 和 CC2NP=0 以确定极性（只检测低电平）。
- 置 TIM15_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIM15_SMCR 寄存器中 TS=00110，选择 tim_ti2 作为输入源。

当 tim_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。

tim_ti2 上升沿和计数器启动计数之间的延时，取决于 tim_ti2 输入端的重同步电路。

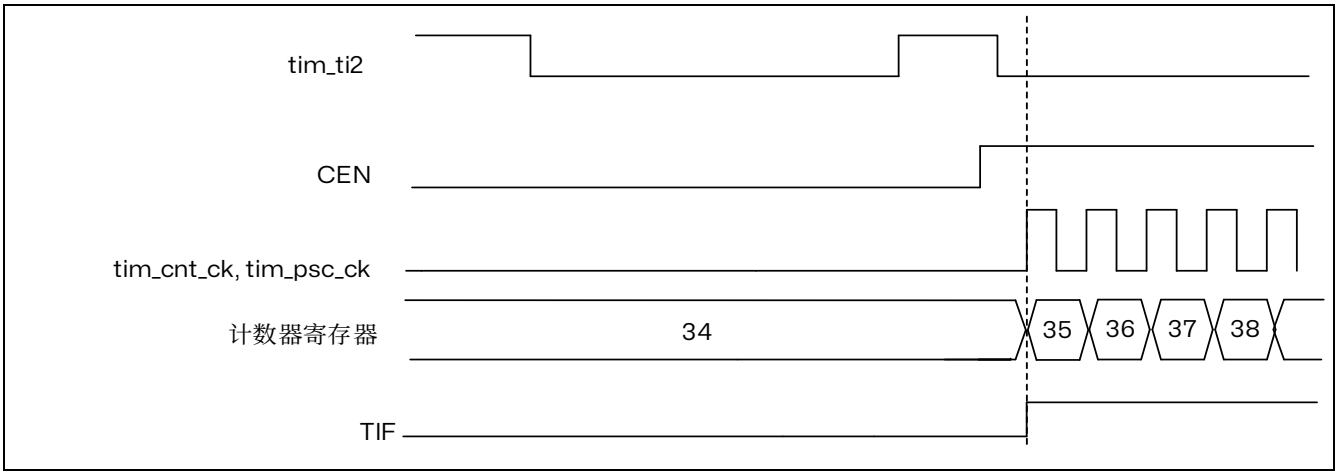



图 24.40 触发器模式下的控制电路

**对实时更新编码器模式的从模式选择预装载**

SMS[3:0]位可以预加载。这是通过在 TIM15_SMCR 寄存器中设置 SMSPE 使能位来实现的。从 SMS[3:0]预加载到有效值的触发器是计数器溢出时发生的更新事件（UEV）。

**从模式 - 组合复位 + 触发模式**

在这种情况下，选择的触发输入（tim_trgi）的上升沿重新初始化计数器，生成寄存器的更新，并启动计数器。

该模式用于单脉冲模式。

**从模式 - 组合复位 + 门控模式**

触发输入（tim_trgi）高时使能计数器。一旦触发变低，计数器停止并复位。计数器的开始和停止都是受控的。

该模式允许检测超出范围的 PWM 信号（占空比超过最大期望值）。

### 24.4.23 定时器同步（仅 TIM15）

TIM15 定时器从内部连接在一起，以实现定时器同步或级联。

注：接收 tim_trgo 信号的从外设（定时器、ADC...）的时钟必须在主定时器接收事件前使能，并且在主定时器接收触发时，时钟频率（预分频器）不得动态更改。

### 24.4.24 使用定时器输出作为其他定时器的触发器（仅 TIM16/TIM17）

只有一个通道的定时器不具备主模式。但是，OC1 输出信号可用于触发其他定时器（包括本文档中其他部分描述的定时器）。请检查设备上任何定时器的“TIMx 内部触发器连接”表，以确定哪些定时器可以作为从定时器。

从定时器检测到触发器的 OC1 信号脉冲宽度必须至少为目标定时器的 2 个时钟周期。

例如，如果目标定时器的 CK_INT 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。

### 24.4.25 ADC 触发（仅 TIM15）

定时器可以使用各种内部信号（例如重置、使能或比较事件）生成 ADC 触发事件。

*注意：接收*tim_trgo *信号的从属外设（定时器、*ADC *等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能更改。*

### 24.4.26 DMA 突发模式

TIMx 定时器具有在单个事件上生成多个 DMA 请求的能力。其主要目的是能够在不增加软件开销的情况下多次重新编程定时器的一部分，但它也可以用于在固定间隔内连续读取多个寄存器。

DMA 控制器的目标地址是唯一的，必须指向虚拟寄存器 TIMx_DMAR。在给定的定时器事件上，定时器启动一系列 DMA 请求（突发）。对 TIMx_DMAR 寄存器的每次写入实际上都被重定向到定时器寄存器之一。

TIMx_DCR 寄存器中的 DBL[4:0]位设置了 DMA 突发长度。当对 TIMx_DMAR 地址进行读取或写入访问时，定时器会识别出突发传输，即传输的数量（以半字或字节为单位）。

TIMx_DCR 寄存器中的 DBA[4:0]位定义了 DMA 传输的 DMA 基地址（当通过 TIMx_DMAR 地址进行读取/写入访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始定义的偏移量：

示例：

00000：TIMx_CR1

00001：TIMx_CR2

00010：TIMx_SMCR

作为一个例子，定时器 DMA 突发模式用于在更新事件发生时更新 CCRx 寄存器（x = 2，3，4）的内容，其中 DMA 将半字传输到 CCRx 寄存器中。

这是通过以下步骤完成的：

1. 配置相应的 DMA 通道如下：
   - DMA 通道外设地址是 DMAR 寄存器的地址。
   - DMA 通道内存地址是 RAM 中缓冲区的地址，其中包含要通过 DMA 传输到 CCRx 寄存器的数据。
   - 要传输的数据数量 = 3（见下文注释）。
   - 循环模式禁用。
2. 通过配置 DBA 和 DBL 位域来配置 DCR 寄存器，如下所示：

DBL = 3 次传输，DBA = 0xE。

3. 使能 TIMx 更新 DMA 请求（在 DIER 寄存器中设置 UDE 位）。
4. 使能 TIMx
5. 使能 DMA 通道

此示例适用于每个 CCRx 寄存器都需要更新一次的情况。例如，如果要更新每个 CCRx 寄存器两次，则要传输的数据数量应为 6。让我们以一个在 RAM 中包含 data1，data2，data3，data4，data5 和 data6 的缓冲区为例。数据按照以下方式传输到 CCRx 寄存器：在第一个更新 DMA 请求时，data1 传输到 CCR2，data2 传输到 CCR3，data3 传输到 CCR4；在第二个更新 DMA 请求时，data4 传输到 CCR2，data5 传输到 CCR3，data6 传输到 CCR4。

*注：可以将空值写入保留的寄存器。*

### 24.4.27 TIM15/TIM16/TIM17 DMA 请求

TIM15/TIM16/TIM17 可以生成 DMA 请求，如下表所示。

表 24.11 DMA 请求

| DMA 请求信号 | DMA 缩写 | DMA 请求 | 使能控制位 |
|---|---|---|---|
| tim_upd_dma | TIM_UP | 更新 | UDE |
| tim_cc1_dma | TIM_CH1 | 捕获/比较 1 | CC1DE |
| tim_com_dma(1) | TIM_COM | 换向（COM） | COMDE |
| tim_trg_dma(1) | TIM_TRIG | 触发 | TDE |

1. 仅供 TIM15 使用。

### 24.4.28 调试模式

当微控制器进入调试模式（内核停止）时，TIMx 计数器可以继续正常工作或停止。

在调试模式下，每个定时器的行为可以通过 Debug 支持（DBG）模块的专用配置位进行编程。

出于安全考虑，当计数器停止时，输出被禁用（就像将 MOE 位复位一样）。输出可以强制处于无效状态（OSSI 位=1），或者由 GPIO 控制器接管控制（OSSI 位=0）以强制其处于高阻态。

有关更多详细信息，请参阅调试部分。

## 24.5 TIM15/TIM16/TIM17 低功耗模式

表 24.12 低功耗模式对 TIM15/TIM16/TIM17 的影响

| 模式 | 描述 |
|---|---|
| 睡眠 | 无影响，外设是运行的。中断会导致设备退出睡眠模式。 |
| 停机 | 定时器操作被停止并保留其寄存器内容。无法生成中断。 |
| 待机 | 定时器被断电并且退出待机模式后必须重新初始化。 |

## 24.6 TIM15/TIM16/TIM17 中断

TIM15 可以生成多个中断，如下表所示。

表 24.13 中断请求

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机和待机退出 |
|---|---|---|---|---|---|---|
| TIM | 更新 | UIF | UIE | UIF 写 0 | 是 | 否 |
| | 捕获/比较 1 | CC1IF | CC1IE | CC1IF 写 0 | 是 | 否 |
| | 捕获/比较 2(1) | CC2IF | CC2IE | CC2IF 写 0 | 是 | 否 |
| | 换向（COM） | COMIF | COMIE | COMIF 写 0 | 是 | 否 |
| | 触发(1) | TIF | TIE | TIF 写 0 | 是 | 否 |
| | 刹车 | BIF | BIE | BIF 写 0 | 是 | 否 |

1. 仅供 TIM15 使用。

## 24.7 TIM15 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。

### 24.7.1 TIM15 控制寄存器 1（TIM15_CR1）

偏移地址：0x000

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DITHEN | UIFREMAP | Res | CKD[1:0] | | ARPE | Reserved | | | OPM | URS | UDIS | CEN |
| | | | rw | rw | | rw | rw | rw | | | | rw | rw | rw | rw |

Bit 15:13 保留，必须保持复位值。

Bit 12 **DITHEN**：使能抖动（Dithering enable）

0：抖动禁止

1：抖动使能

*注：*DITHEN *位只能在*CEN *位复位时修改*

Bit 11 **UIFREMAP**：UIF 状态位重映射（UIF status bit remapping）

0：无重映射。UIF 状态位未复制到 TIM15_CNT 寄存器位 31

1：重映射使能。UIF 状态位复制到 TIM15_CNT 寄存器位 31

Bit 10 保留，必须保持复位值。

Bit 9:8 **CKD[1:0]**：时钟分频（Clock division）

此位域指示定时器时钟（tim_ker_ck）频率以及数字滤波器（tim_tix）所使用的采样时钟（$t_{DTS}$）之间的分频比

00：$t_{DTS} = t_{tim_ker_ck}$

01：$t_{DTS} = 2 \times t_{tim_ker_ck}$

10：$t_{DTS} = 4 \times t_{tim_ker_ck}$

11：保留，不要设置成此值

Bit 7 **ARPE**：自动重载预装载使能（Auto-reload preload enable）

0：TIM15_ARR 寄存器不进行缓冲；

1：TIM15_ARR 寄存器进行缓冲。

Bit 6:4 保留，必须保持复位值。

Bit 3 **OPM**：单脉冲模式（One pulse mode）

0：计数器在发生更新事件时不会停止计数；

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

Bit 2 **URS**：更新请求源（Update request source）

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断。此类事件包括：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断。

Bit 1 **UDIS**：更新禁止（Update disable）

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新（UEV）事件可通过以下事件之一生成：

- 计数器上出/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

Bit 0 **CEN**：计数器使能（Counter enable）

0：禁止计数器；

1：使能计数器。

*注：只有事先通过软件将*CEN *位置*1，*才可以使用外部时钟、门控模式。而触发模式可通过硬件自动将*CEN *位置*1。

### 24.7.2 TIM15 控制寄存器 2（TIM15_CR2）

偏移地址：0x004

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | OIS2 | OIS1N | OIS1 | TI1S | MMS[2:0] | | | CCDS | CCUS | Res | CCPC |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw |

Bit 15:11 保留，必须保持复位值。

Bit 10 **OIS2**：输出空闲状态 2（OC2 输出）(Output idle state 2 (tim_oc2 output))

0：当 MOE=0 时，tim_oc2=0

1：当 MOE=0 时，tim_oc2=1

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 9 **OIS1N**：输出空闲状态 1（OC1N 输出）(Output idle state 1 (tim_oc1n output))

0：当 MOE=0 时，经过死区时间后 tim_oc1n=0

1：当 MOE=0 时，经过死区时间后 tim_oc1n=1

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 8 **OIS1**：输出空闲状态 1（OC1 输出）(Output idle state 1 (tim_oc1 output))

0：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=0

1：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=1

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别* 1、2 *或*3，*此位即无法修改。*

Bit 7 **TI1S**：tim_ti1 选择（tim_ti1 selection）

0：tim_ti1_in[15:0]多路复用器输出连接到 tim_ti1_in 输入

1：tim_ti1_in[15:0]和 tim_ti2_in[15:0]多路复用器输出连接到 tim_ti1_in 输入（异或组合）

Bit 6:4 **MMS[2:0]：**主模式选择（Master mode selection）

这些位可选择主模式下将要发送到从定时器以实现同步的信息（tim_trgo）。这些位的组合如下：

000：**复位** – TIM15_EGR 寄存器中的 UG 位用作触发输出（tim_trgo）。如果复位由触发输入生成（从模式控制器配置为复位模式），则 tim_trgo 上的信号相比实际复位会有延迟。

001：**使能** – 计数器使能信号 CNT_EN 用作触发输出（tim_trgo）。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由 CEN 控制位产生。当配置为门控模式时，也可由触发输入产生。当计数器使能信号由触发输入控制时，tim_trgo 上会存在延迟，选择主/从模式时除外（请参见 TIM15_SMCR 寄存器中 MSM 位的说明）。

010：**更新** – 选择更新事件作为触发输出（tim_trgo）。例如，主定时器可用作从定时器的预分频器。

011：**比较脉冲** – 一旦发生输入捕获或比较匹配事件，当 CC1IF 被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(tim_trgo)。

100：**比较** – tim_oc1refc 信号用作触发输出（tim_trgo）

101：**比较** – tim_oc2refc 信号用作触发输出（tim_trgo）

*注：若选* tim_ocxrefc *为* tim_trgo *输出，当* tim_ocref_clr_int *清零信号生效时，* tim_ocxrefc *信号将保持低电平，* tim_trgo *也不会有触发输出，直到发生下一更新事件（*UEV*）（特别注意，当* TIMx_RCR>0 *时，只有重复计数器达到零才会生成更新事件，波形才会恢复）。*

Bit 3 **CCDS**：捕获/比较 DMA 选择（Capture/compare DMA selection）

0：发生 CCx 事件时发送 CCx DMA 请求

1：发生更新事件时发送 CCx DMA 请求

Bit 2 **CCUS**：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(CCPC=1)进行预装载，仅通过将 COMG 位置 1 来对这些位进行更新；

1：如果捕获/比较控制位(CCPC=1)进行预装载，可通过将 COMG 位置 1 或 TRGI 的上升沿对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。*

Bit 1 保留，必须保持复位值。

Bit 0 **CCPC**：捕获/比较预装载控制（Capture/compare preloaded control）

0：CCxE，CCxNE 和 OCxM 位未进行预装载；

1：CCxE，CCxNE 和 OCxM 位进行了预装载，写入这些位后，仅当发生换向事件(COM)（COMG 位置 1 或在 TRGI 上检测到上升沿，取决于 CCUS 位）时才会对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。*

### 24.7.3 TIM15 从模式控制寄存器（TIM15_SMCR）

偏移地址：0x008

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | TS[4:3] | | Reserved | | | SMS[3] |
| | | | | | | | | | | rw | rw | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | MSM | TS[2:0] | | | Res | SMS[2:0] | | |
| | | | | | | | | rw | rw | rw | rw | | rw | rw | rw |

Bit 31:22 保留，必须保持复位值。

Bit 19:17 保留，必须保持复位值。

Bit 15:8 保留，必须保持复位值。

Bit 7 **MSM：**主/从模式（Master/slave mode）

0：不执行任何操作；

1：当前定时器的触发输入事件（tim_trgi）的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 tim_trgo）。此设置适用于由单个外部事件对多个定时器进行同步的情况。

Bit 21,20,6,5,4 **TS[4:0]：**触发选择（Trigger selection）

此位域可选择将要用于同步计数器的触发输入。

00000：内部触发 0（tim_itr0）

00001：内部触发 1（tim_itr1）

00010：内部触发 2（tim_itr2）

00011：内部触发 3（tim_itr3）

00100：tim_ti1 边沿检测器（tim_ti1f_ed）

00101：滤波后的定时器输入 1（tim_ti1fp1）

00110：滤波后的定时器输入 2（tim_ti2fp2）

00111：保留

01000：内部触发器 4 (tim_itr4)

01001：内部触发器 5 (tim_itr5)

01010：内部触发器 6 (tim_itr6)

01011：内部触发器 7 (tim_itr7)

01100：内部触发器 8 (tim_itr8)

01101：内部触发器 9 (tim_itr9)

01110：内部触发器 10 (tim_itr10)

01111：内部触发器 11 (tim_itr11)

10000：内部触发器 12 (tim_itr12)

10001：内部触发器 13 (tim_itr13)

10010：内部触发器 14 (tim_itr14)

10011：内部触发器 15 (tim_itr15)

其他：保留

有关定时器 ITRx 含义的详细信息，请参见章节：TIM15/TIM16/TIM17 引脚和内部信号。

*注：这些位只能在未使用的情况下（例如，SMS=000 时）进行更改，以避免转换时出现错误的边沿检测。*

Bit 3 保留，必须保持复位值。

Bit 16, 2,1,0 **SMS[3:0]**：从模式选择（Slave mode selection）

当选择了外部信号，触发信号（tim_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（tim_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（tim_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 tim_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（tim_trgi）的上升沿驱动计数器。

1000：**组合复位+触发模式** - 当选择的触发输入（tim_trgi）上升沿出现时，会重新初始化计数器，生成寄存器的更新并启动计数器。

1001：**组合门控+复位模式** - 当触发输入（tim_trgi）为高电平时，计数器时钟被使能。一旦触发变为低电平，计数器就会停止并重置。这样，计数器的启动和停止都得到了控制。

其他：保留

*注：如果选择了*tim_ti1f_ed *作为触发输入（*TS=00100*），则不能使用门控模式。事实上，*tim_ti1f_ed *为每个*tim_ti1f *上的转换输出一个脉冲，而门控模式检查触发信号的电平。*

*注：接收*tim_trgo *信号的从设备（定时器、*ADC *等）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发时不能改变。*

### 24.7.4 TIM15 DMA/中断使能寄存器（TIM15_DIER）

偏移地址：0x00C

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | TDE | COMDE | Reserved | | | CC1DE | UDE | BIE | TIE | COMIE | Reserved | | CC2IE | CC1IE | UIE |
| | rw | rw | | | | rw | rw | rw | rw | rw | | | rw | rw | rw |

Bit 15 保留，必须保持复位值。

Bit 14 **TDE**：触发 DMA 请求使能（Trigger DMA request enable）

0：禁止触发 DMA 请求；

1：使能触发 DMA 请求。

Bit 13 **COMDE**：COM DMA 请求使能（COM DMA request enable）

0：禁止换向 DMA 请求；

1：使能换向 DMA 请求。

Bit 12:10 保留，必须保持复位值。

Bit 9 **CC1DE**：捕获/比较 1 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止 CC1 DMA 请求；

1：使能 CC1 DMA 请求。

Bit 8 **UDE**：更新 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止更新 DMA 请求；

1：使能更新 DMA 请求。

Bit 7 **BIE**：刹车中断使能（Break interrupt enable）

0：禁止刹车中断；

1：使能刹车中断。

Bit 6 **TIE**：触发信号（TGRI）中断使能（Trigger interrupt enable）

0：禁止触发信号（TGRI）中断；

1：使能触发信号（TGRI）中断。

Bit 5 **COMIE**：COM 中断使能（COM interrupt enable）

0：禁止 COM 中断；

1：使能 COM 中断。

Bit 4:3 保留，必须保持复位值。

Bit 2 **CC2IE**：捕获/比较 2 中断使能（Capture/Compare 2 interrupt enable）

0：禁止 CC2 中断；

1：使能 CC2 中断。

Bit 1 **CC1IE**：捕获/比较 1 中断使能（Capture/Compare 1 interrupt enable）

0：禁止 CC1 中断；

1：使能 CC1 中断。

Bit 0 **UIE**：更新中断使能（Update interrupt enable）

0：禁止更新中断；

1：使能更新中断。

### 24.7.5 TIM15 状态寄存器（TIM15_SR）

偏移地址：0x010

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | CC2OF | CC1OF | Res | BIF | TIF | COMIF | Reserved | | CC2IF | CC1IF | UIF |
| | | | | | rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rc_w0 | rc_w0 |

Bit 15:11 保留，必须保持复位值。

Bit 10 **CC2OF**：捕获/比较 2 重复捕获标志

参考 CC1OF 的描述

Bit 9 **CC1OF**：捕获/比较 1 重复捕获标志

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获

1：TIM15_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

Bit 8 保留，必须保持复位值。

Bit 7 **BIF**：刹车中断标志（Break interrupt flag）

只要 tim_brk 输入变为有效状态，此标志便由硬件置 1。刹车输入无效后可通过软件对其清零。

0：未发生刹车事件。

1：在刹车输入上检测到有效电平。

Bit 6 **TIF**：触发中断标志（Trigger interrupt flag）

在除门控模式以外的所有模式下，当使能从模式控制器后在 tim_trgi 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。

0：未发生触发事件。

1：触发中断挂起。

Bit 5 **COMIF**：COM 中断标志（COM interrupt flag）

此标志在发生 COM 事件时（捕获/比较控制位 CCxE、CCxNE 和 OCxM 已更新时）由硬件置 1。但需要通过软件清零。

0：未发生 COM 事件。

1：COM 中断挂起。

Bit 4:3 保留，必须保持复位值。

Bit 2 **CC2IF**：捕获/比较 2 中断标志（Capture/Compare 2 interrupt flag）

参考 CC1IF 描述。

Bit 1 **CC1IF**：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag）

此标志由硬件设置，它可以通过软件（输入捕获或输出比较模式）或读取 TIM15_CCR1 寄存器（仅输入捕获模式）来清除。

0：未比较匹配/无输入捕获发生

1：发生比较匹配或输入捕获

**如果通道 CC1 配置为输出：**当计数器与比较值匹配时，此标志由硬件置 1，当 TIM15_CCR1 的值大于 TIM15_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。在中心对齐模式下有 3 种可能的标志设置选项，请参考 TIM15_CR1 寄存器中的 CMS 位以获取完整描述。

**如果通道 CC1 配置为输入：**当在 TIM15_CCR1 寄存器中捕获到计数器值时（在 IC1 上检测到边沿，具体取决于在 TIM15_CCER 中通过 CC1P 和 CC1NP 位设置定义的边沿灵敏度），会设置此位。

Bit 0 **UIF**：更新中断标志（Update interrupt flag）

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

- TIM15_CR1 寄存器中的 UDIS=0，并且重复计数器值上溢或下溢时（重复计数器=0 时更新）。
- TIM15_CR1 寄存器中的 URS=0 且 UDIS=0，并且由软件使用 TIM15_EGR 寄存器中的 UG 位重新初始化 CNT 时。
- TIM15_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（参考 TIM15 从模式控制寄存器（TIM15_SMCR））。

### 24.7.6 TIM15 事件产生寄存器（TIM15_EGR）

偏移地址：0x014

复位：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BG | TG | COMG | Reserved | | CC2G | CC1G | UG |
| | | | | | | | | w | w | rw | | | w | w | w |

Bit 15:8 保留，必须保持复位值。

Bit 7 **BG**：刹车生成（Break generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：生成刹车事件。MOE 位清零且 BIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

Bit 6 **TG**：生成触发信号（Trigger generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断事件或 DMA 传输。

Bit 5 **COMG**：捕获/比较控制更新生成（Capture/Compare control update generation）

该位可通过软件置 1，并由硬件自动清零

0：不执行任何操作；

1：CCPC 位置 1 时，可更新 CCxE、CCxNE 和 OCxM 位。

*注：此位仅对具有互补输出的通道有效。*

Bit 4:3 保留，必须保持复位值。

Bit 2 **CC2G**：捕获/比较 2 生成（Capture/Compare 2 generation）

参考 CC1G 描述。

Bit 1 **CC1G**：捕获/比较 1 生成（Capture/Compare 1 generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：通道 1 上生成捕获/比较事件：

**若通道 CC1 配置为输出：**

使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。

**如果通道 CC1 配置为输入：**

TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

Bit 0 **UG**：更新生成（Update generation）

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作；

1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。

### 24.7.7 TIM15 捕获/比较模式寄存器 1 [复用]（TIM15_CCMR1）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IC2F[3:0] | | | | IC2PSC[1:0] | | CC2S[1:0] | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:16 保留，始终读为 0。

Bit 15:12 **IC2F[3:0]：**输入捕获 2 滤波器（Input capture 2 filter）

Bit 11:10 **IC2PSC[1:0]：**输入/捕获 2 预分频器（input capture 2 prescaler）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择（Capture/compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道被配置为输出；

01：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti2 上；

10：CC2 通道被配置为输入，tim_ic2 映射在 tim_ti1 上；

11：CC2 通道被配置为输入，tim_ic2 映射在 tim_trc 上。此模此模式仅在通过 TS 位（TIM15_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIM15_CCER *中的*CC2E = 0*），才可向*CC2S *位写入数据。*

Bit 7:4 **IC1F[3:0]：**输入捕获 1 滤波器（Input capture 1 filter）

此位域可定义 tim_ti1 输入的采样频率和适用于 tim_ti1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 3:2 **IC1PSC[1:0]：**输入/捕获 1 预分频器（Input capture 1 prescaler）

此位域定义 CC1 输入（tim_ic1）的预分频比。

只要 CC1E=0（TIM15_CCER 寄存器），预分频器便立即复位。

00：无预分频器，捕获输入上每检测到一个边沿便执行捕获

01：每发生 2 个事件便执行一次捕获；

10：每发生 4 个事件便执行一次捕获；

11：每发生 8 个事件便执行一次捕获。

Bit 1:0 **CC1S[1:0]：**捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道被配置为输出；

01：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti1 上；

10：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti2 上；

11：CC1 通道被配置为输入，tim_ic1 映射在 tim_trc 上。此模式仅在通过 TS 位（TIM15_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（TIM15_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。*

### 24.7.8 TIM15 捕获/比较模式寄存器 1 [复用]（TIM15_CCMR1）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | OC2M[3] | Reserved | | | | | | | 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] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式**

Bit 31:25 保留，必须保持复位值。

Bit 23:17 保留，必须保持复位值。

Bit 15 **OC2CE：**输出比较 2 清零使能（Output Compare 2 clear enable）

Bit 24, 14:12 **OC2M[2:0]：**输出比较 2 模式（Output Compare 2 mode）

Bit 11 **OC2PE：**输出比较 2 预装载使能（Output Compare 2 preload enable）

Bit 10 **OC2FE：**输出比较 2 快速使能（Output Compare 2 fast enable）

Bit 9:8 **CC2S[1:0]：**捕获/比较 2 选择（Capture/Compare 2 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，tim_ic2 映射到 tim_ti2 上

10：CC2 通道配置为输入，tim_ic2 映射到 tim_ti1 上

11：CC2 通道配置为输入，tim_ic2 映射到 tim_trc 上。此模式仅在通过 TS 位（TIM15_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIM15_CCER *中的* CC2E = 0*），才可向* CC2S *位写入数据*

Bit 7 **OC1CE：**输出比较 1 清零使能（Output Compare 1 clear enable）

0：tim_oc1ref 不受 tim_ocref_clr_int 输入影响；

1：tim_oc1ref 在 tim_ocref_clr_int 输入中被检测到高电平，tim_oc1ref 立即清零。

Bit 16, 6:4 **OC1M[3:0]：**输出比较 1 模式（Output Compare 1 mode）

这些位定义提供 tim_oc1 和 tim_oc1n 的输出参考信号 tim_oc1ref 的行为。tim_oc1ref 为高电平有效，而 tim_oc1 和 tim_oc1n 的有效电平则取决于 CC1P 位和 CC1NP 位。

0000：**冻结** — 输出比较寄存器 TIM15_CCR1 与计数器 TIM15_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

0001：将通道 1 设置为匹配时输出有效电平。当计数器 TIM15_CNT 与捕获/比较寄存器 1（TIM15_CCR1）匹配时，tim_oc1ref 信号强制变为高电平。

0010：将通道 1 设置为匹配时输出无效电平。当计数器 TIM15_CNT 与捕获/比较寄存器 1（TIM15_CCR1）匹配时，tim_oc1ref 信号强制变为低电平。

0011：**翻转** – TIM15_CNT=TIM15_CCR1 时，tim_oc1ref 发生翻转。

0100：**强制变为无效电平** – tim_oc1ref 强制变为低电平。

0101：**强制变为有效电平** – tim_oc1ref 强制变为高电平。

0110：**PWM 模式 1** – 在递增计数模式下，只要 TIM15_CNT<TIM15_CCR1，通道 1 便为有效状态，否则为无效状态。

0111：**PWM 模式 2** – 在递增计数模式下，只要 TIM15_CNT<TIM15_CCR1，通道 1 便为无效状态，否则为有效状态。

1000：**可重触发 OPM 模式 1** - 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于无效状态。

1001：**可重触发 OPM 模式 2** - 在向上计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于无效状态。然后，会像 PWM 模式 2 那样执行比较，并在下次更新时使通道再次处于无效状态。在向下计数模式下，通道在检测到触发事件（tim_trgi 信号）之前一直处于有效状态。然后，会像 PWM 模式 1 那样执行比较，并在下次更新时使通道再次处于有效状态。

1010：保留

1011：保留

1100：**组合 PWM 模式 1** - tim_oc1ref 的行为与 PWM 模式 1 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑或。

1101：**组合 PWM 模式 2** - tim_oc1ref 的行为与 PWM 模式 2 相同。tim_oc1refc 是 tim_oc1ref 和 tim_oc2ref 之间的逻辑与。

1110：保留

1111：保留

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*3 *且*CC1S= *“*00*”（通道配置为输出），这些位即无法修改。*

*在* PWM *模式下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“*PWM*”模式时，*tim_ocref_clr *电平才会发生更改。*

*对于具有互补输出的通道，此位域是预加载的。如果*TIM15_CR2 *寄存器中*CCPC *位被设置，则*OC1M *有效位仅在生成*COM *事件时从预加载位获取新值。*

Bit 3 **OC1PE**：输出比较 1 预装载使能（Output Compare 1 preload enable）

0：禁止与 TIM15_CCR1 相关的预装载寄存器。可随时向 TIM15_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIM15_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIM15_CCR1 预装载值在每次生成更新事件时都会装载到有效寄存器中。

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*3 *且* CC1S=00 *（通道配置为输出），这些位即无法修改。*

Bit 2 **OC1FE**：输出比较 1 快速使能（Output Compare 1 fast enable）

此位减少了触发事件和定时器输出上的转换之间的延迟。它必须在单脉冲模式（在 TIM15_CR1 寄存器中设置 OPM 位）中使用，以使输出脉冲在启动触发后尽快开始。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

Bit 1:0 **CC1S[1:0]**：捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，tim_ic1 映射到 tim_ti1 上

10：CC1 通道配置为输入，tim_ic1 映射到 tim_ti2 上

11：CC1 通道配置为输入，tim_ic1 映射到 tim_trc 上。此模式仅在通过 TS 位（TIM15_SMCR 寄存器）选择内部触发输入时有效

*注：仅当通道关闭时（*TIM15_CCER *中的* CC1E = 0*），才可向* CC1S *位写入数据*

### 24.7.9 TIM15 捕获/比较使能寄存器（TIM15_CCER）

偏移地址：0x020

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CC2NP | Res | CC2P | CC2E | CC1NP | CC1NE | CC1P | CC1E |
| | | | | | | | | rw | | rw | rw | rw | rw | rw | rw |

Bit 15:8 保留，必须保持复位值。

Bit 7 **CC2NP**：输入/捕获 2 输出极性（Capture/Compare 2 output polarity）

参考 CC1NP 的描述。

Bit 6 保留，必须保持复位值。

Bit 5 **CC2P**：输入/捕获 2 输出极性（Capture/Compare 2 output polarity）

参考 CC1P 的描述。

Bit 4 **CC2E**：输入/捕获 2 输出使能（Capture/Compare 2 output enable）

参考 CC1E 的描述。

Bit 3 **CC1NP**：输入/捕获 1 输出极性（Capture/Compare 1 output polarity）

**CC1 通道配置为输出：**

0：tim_oc1n 高电平有效；

1：tim_oc1n 低电平有效。

**CC1 通道配置为输入：**

此位与 CC1P 配合使用，用以定义 tim_ti1fp1/ tim_ti2fp1 的极性。请参见 CC1P 说明。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *或*3 *且*CC1S=00 *（通道配置为输出），此位立即变为不可写状态*

*注：此位将在具有互补输出的通道上进行预装载。如果* TIM15_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1NP *有效位才会从预装载位获取新值。*

Bit 2 **CC1NE**：输入/捕获 1 互补输出使能（Capture/Compare 1 complementary output enable）

0：关闭 –-OC1N 未激活。OC1N 电平是 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的函数。

1：开启 -- 在相应输出引脚上输出 OC1N 信号，具体取决于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位。

Bit 1 **CC1P**：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（输出模式）/边沿灵敏度选择（输入模式，参见以下）

1：OC1 低电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

**CC1 通道配置为输入，**CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

CC1NP=0，CC1P=0：非反相/上升沿触发。电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=0，CC1P=1：反相/下降沿触发。电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=1，CC1P=1：非反相/上升沿和下降沿均触发。电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

CC1NP=1，CC1P=0：此配置是保留的，不应该使用此配置。

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的* LOCK *位）级别*2 *或*3*，此位立即变为不可写状态。*

*注：此位将在具有互补输出的通道上进行预装载。如果* TIM15_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1P *有效位才会从预装载位获取新值。*

Bit 0 **CC1E**：输入/捕获 1 输出使能（Capture/Compare 1 output enable）

0：禁止捕获模式/OC1 未激活（参见以下）

1：使能捕获模式/ OC1 信号输出到对应的输出引脚

**当 CC1 通道配置为输出，**C1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值，不论 CC1E 位状态。请参考下表获得更多细节

表 24.14 带刹车功能的互补输出通道 tim_oc1 和 tim_oc1n 的控制位（TIM15）

| 控制位 | | | | | 输出状态 (1) | |
|---|---|---|---|---|---|---|
| MOE 位 | OSSI 位 | OSSR 位 | CCxE 位 | CCxNE 位 | tim_ocx 输出状态 | tim_ocxn 输出状态 |
| 1 | X | X | 0 | 0 | 禁止输出（不由定时器驱动：高阻态）<br>tim_ocx = 0，tim_ocxn = 0 | |
| | | 0 | 0 | 1 | 禁止输出（不由定时器驱动：高阻态）tim_ocx = 0 | tim_ocxref + 极性，tim_ocxn = tim_ocxref 异或 CCxNP |
| | | 0 | 1 | 0 | tim_ocxref + 极性，tim_ocx = tim_ocxref 异或 CCxP | 禁止输出（不由定时器驱动：高阻态）tim_ocxn = 0 |
| | | X | 1 | 1 | tim_ocxref + 极性 + 死区 | tim_ocxref 互补项（对 tim_ocxref 进行“非”运算）+ 极性 + 死区 |
| | | 1 | 0 | 1 | 关闭状态（输出使能为无效状态）<br>tim_ocx = CCxP | tim_ocxref + 极性，tim_ocxn = tim_ocxref 异或 CCxNP |
| | | 1 | 1 | 0 | tim_ocxref + 极性，tim_ocx = tim_ocxref 异或 CCxP | 关闭状态（输出使能为无效状态）<br>tim_ocxn = CCxNP |
| 0 | 0 | X | X | X | 禁止输出（不再由定时器驱动）。<br>输出状态由 GPIO 控制器定义，可以是高电平、低电平或高阻态。 | |
| | 1 | | 0 | 0 | | |
| | | | 0 | 1 | 关闭状态（输出使能为无效状态）<br>异步：tim_ocx = CCxP、tim_ocxn = CCxNP<br>如果存在时钟：在死区后 tim_ocx = OISx 且 tim_ocxn = OISxN，假定 OISx 和 OISxN 并没有都设置成 tim_ocx 及 tim_ocxn 的有效电平。 | |
| | | | 1 | 0 | | |
| | | | 1 | 1 | | |

1. 如果一个通道的两个输出均未使用（由 GPIO 接管控制控制），则 OISx、OISxN、CCxP 和 CCxNP 位必须保持清零状态。

*注：　与互补通道* tim_ocx *和* tim_ocxn *相连的外部* I/O *引脚的状态取决于通道* tim_ocx *和* tim_ocxn *的状态以及* GPIO *寄存器。*

### 24.7.10 TIM15 计数器（TIM15_CNT）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 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 31 **UIFCPY：**复制 UIF

该位是 TIM15_SR 寄存器中 UIF 位的一个只读副本。

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CNT[15:0]：**计数器的值（Counter value）

在非抖动模式下（DITHEN = 0）

该寄存器（CNT[15:0]）保存的是计数器的值。

在抖动模式下（DITHEN = 1）

该寄存器（CNT[15:0]）只保存计数器的不抖动部分。抖动部分不可用。

### 24.7.11 TIM15 预分频器（TIM15_PSC）

偏移地址：0x028

复位值：0x0000

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

Bit 15:0 **PSC[15:0]：**预分频器的值（Prescaler value）

计数器时钟频率（$f_{tim_cnt_ck}$）等于 $f_{tim_psc_ck}$ / (PSC[15:0] + 1)。

PSC 包含每次发生更新事件（包括计数器通过 TIM15_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

### 24.7.12 TIM15 自动重装载寄存器（TIM15_ARR）

偏移地址：0x02C

复位：0x0000 FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | 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 31:20 保留，必须保持复位值。

Bit 19:0 **ARR[19:0]：**自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

当自动重载值为空时，计数器不工作。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是自动重新加载的值

在抖动模式下（DITHEN = 1）

该寄存器存储的是 ARR[19:4]中的整数部分，而 ARR[3:0]位域存储的是抖动的部分

### 24.7.13 TIM15 重复计数寄存器（TIM15_RCR）

偏移地址：0x030

复位值：0x0000

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

Bit 15:8 保留，必须保持复位值。

Bit 7:0 **REP[7:0]**：重复计数器的值（Repetition counter value）

使能预装载寄存器时，用户可通过这些位设置比较寄存器的更新频率（即从预装载寄存器向活动寄存器周期性传输数据）；使能更新中断时，也可设置更新中断的生成速率。

与 REP_CNT 相关的减计数器每次计数到 0 时，都将生成一个更新事件并且计数器从 REP 值重新开始计数。由于只有生成重复更新事件 U_RC 时，REP_CNT 才会重载 REP 值，因此在生成下一重复更新事件之前，无论向 TIMx_RCR 寄存器写入何值都无影响。

这意味着 PWM 模式下（REP+1）相当于：

– 边沿对齐模式下的 PWM 周期数；

– 中心对称模式下的 PWM 半周期数；

### 24.7.14 TIM15 捕获/比较寄存器 1（TIM15_CCR1）

偏移地址：0x034

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR1 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR1[19:0]：**捕获/比较 1 值（Capture/Compare 1 value）

**若 CC1 通道配置为输出：**

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIM15_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 1 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIM15_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR1[15:0]中的比较值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR1[19:4]中的整数部分。CCR1[3:0]位域存储的是抖动的部分。

**若 CC1 通道配置为输入：**

CCR1 为上一个输入捕获 1 事件（tim_ic1）发生时的计数器值。TIM15_CCR1 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR1[15:0]位存储了捕获值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR1[19:4]中。CCR1[3:0]位会被重置。

### 24.7.15 TIM15 捕获/比较寄存器 2（TIM15_CCR2）

偏移地址：0x038

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR2 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR2 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR2[19:0]：**捕获/比较 2 值（Capture/Compare 2 value）

**若 CC2 通道配置为输出：**

CCR2 为要装载到实际捕获/比较 2 寄存器的值（预装载值）。

如果没有通过 TIM15_CCMR1 寄存器中的 OC2PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 2 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIM15_CNT 进行比较并在 tim_oc2 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR2[15:0]中的比较值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR2[19:4]中的整数部分。CCR2[3:0]位域存储的是抖动的部分。

**若 CC2 通道配置为输入：**

CCR2 为上一个输入捕获 2 事件（tim_ic2）发生时的计数器值。TIM15_CCR2 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR2[15:0]位存储了捕获值。CCR2[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR2[19:4]中。CCR2[3:0]位会被重置。

### 24.7.16 TIM15 刹车和死区寄存器（TIM15_BDTR）

偏移地址：0x044

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | BKBID | Res | BK DSRM | Reserved | | | | | | 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 |

*注：由于可以根据*LOCK *配置锁定位*BKBID、BKF[3:0]、AOE、BKP、BKE、OSSI、OSSR *和*DTG[7:0]*的写操作，因此必须在第一次对*TIM15_BDTR *寄存器执行写访问时对这些位进行配置。*

Bit 31:29 保留，必须保持复位值。

Bit 28 **BKBID：**刹车双向（Break bidirectional）

0：刹车输入 tim_brk 处于输入模式

1：刹车输入 tim_brk 处于双向模式

在双向模式（BKBID 置 1）下，刹车输入同时配置为输入模式和开漏输出模式。任何有效的 break 事件会在刹车输入上输出低电平，以向外部设备指示内部 break 事件。

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 27 保留，必须保持为复位值。

Bit 26 **BKDSRM：**刹车解除保护（Break disarm）

0：刹车输入 tim_brk 处于保护状态

1：刹车输入 tim_brk 解除保护

当没有刹车源处于有效状态时，该位由硬件清零。

软件需置位 BKDSRM 以释放双向输出控制（开漏输出处于高阻态），并轮询直至硬件将其清零，表明故障状态已消失。

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 25:20 保留，必须保持为复位值。

Bit 19:16 **BKF[3:0]：**刹车滤波器（Break filter）

此位域可定义 tim_brk 输入的采样频率和适用于 tim_brk 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，tim_brk 异步工作

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 15 **MOE**：主输出使能（Main output enable）

只要刹车输入（tim_brk）为有效状态，此位便由硬件异步清零。此位由软件置 1，也可根据 AOE 位状态自动置 1。此位仅对配置为输出的通道有效。

0：响应刹车事件或向 MOE 写入 0 时：tim_ocx 和 tim_ocxn 输出被禁止或被强制为空闲状态，具体取决于 OSSI 位。

1：如果 tim_ocx 和 tim_ocxn 输出的相应使能位（TIM15_CCER 寄存器中的 CCxE 和 CCxNE 位）均置 1，则使能 tim_ocx 和 tim_ocxn 输出。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIM15 捕获/比较使能寄存器（TIM15_CCER））。

Bit 14 **AOE**：自动输出使能（Automatic output enable）

0：MOE 只能由软件置 1

1：MOE 可由软件置 1，也可在发生下一更新事件时自动置 1（如果刹车输入 tim_brk 无效）

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 13 **BKP**：刹车极性（Break polarity）

0：刹车输入 tim_brk 为低电平有效

1：刹车输入 tim_brk 为高电平有效

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 12 **BKE**：刹车使能（Break enable）

0：禁止刹车输入（tim_brk 及 tim_sys_brk 事件）；

1：使能刹车输入（tim_brk 及 tim_sys_brk 事件）。

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 11 **OSSR**：运行模式下的关闭状态选择（Off-state selection for Run mode）

此位在 MOE=1 时作用于配置为输出模式且具有互补输出的通道。如果定时器中没有互补输出，则不存在 OSSR。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIM15 捕获/比较使能寄存器（TIM15_CCER））。

0：处于无效状态时，禁止 tim_ocx/tim_ocxn 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，一旦 CCxE=1 或 CCxNE=1，便使能 tim_ocx/tim_ocxn 输出并将其设为无效电平（输出仍由定时器控制）。

*注：只要编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*2，*此位即无法修改。*

Bit 10 **OSSI**：空闲模式下的关闭状态选择（Off-state selection for Idle mode）

当由于刹车事件或软件写操作而使 MOE=0 时，此位作用于配置为输出的通道。

有关详细信息，请参见 tim_ocx/tim_ocxn 使能说明（TIM15 捕获/比较使能寄存器（TIM15_CCER））。

0：处于无效状态时，禁止 tim_ocx/tim_ocxn 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，首先将 tim_ocx/tim_ocxn 输出强制为其无效电平，然后在死区后将其强制为空闲电平。定时器始终控制输出。

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*2*，此位即无法修改。*

Bit 9:8 **LOCK[1:0]**：锁定设置（Lock configuration）

这些位用于针对软件错误提供写保护。

00：关闭锁定 – 不对任何位提供写保护。

01：锁定级别 1，此时无法对 TIM15_BDTR 寄存器中的 DTG 位、TIM15_CR2 寄存器中的 OISx 和 OISxN 位以及 TIM15_BDTR 寄存器中的 BKBID/BKE/BKP/AOE 位执行写操作。

10：锁定级别 2，此时无法对锁定级别 1 中适用的各位、CC 极性位（TIM15_CCER 寄存器中的 CCxP/CCxNP 位，只要通过 CCxS 位将相关通道配置为输出）以及 OSSR 和 OSSI 位执行写操作。

11：锁定级别 3，此时无法对锁定级别 2 中适用的各位、CC 控制位（TIM15_CCMRx 寄存器中的 OCxM 和 OCxPE 位，只要通过 CCxS 位将相关通道配置为输出）执行写操作。

*注：复位后只能对*LOCK *位执行一次写操作。对*TIM15_BDTR *寄存器执行写操作后其中的内容将冻结，直到下一次复位。*

Bit 7:0 **DTG[7:0]**：配置死区发生器（Dead-time generator setup）

此位域定义插入到互补输出之间的死区持续时间。DT 与该持续时间相对应。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3*，此位域即无法修改。*

### 24.7.17 TIM15 定时器死区寄存器 2（TIM15_DTR2）

偏移地址：0x054

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DTPE | DTAE |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 31:18 保留，必须保持复位值。

Bit 17 **DTPE：**死区时间预加载使能（Deadtime preload enable）

0：死区时间值未预加载

1：使能死区时间值预加载

*注：编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 16 **DTAE：**死区时间非对称使能（Deadtime asymmetric enable）

0：上升沿和下降沿的死区时间是相同的，并且使用 DTG[7:0]寄存器定义

1：上升沿的死区时间使用 DTG[7:0]寄存器定义，下降沿的死区时间使用 DTGF[7:0]位定义。

*注：编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 15:8 保留，必须保持复位值。

Bit 7:0 **DTGF[7:0]：**死区下降沿发生器（Dead-time falling edge generator setup）

这个位域定义了在下降沿时，互补输出之间插入死区时间的长度。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：编程了*LOCK（TIM15_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

### 24.7.18 TIM15 定时器输入选择寄存器（TIM15_TISEL）

偏移地址：0x05C

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TI2SEL[3:0] | | | | Reserved | | | | TI1SEL[3:0] | | | |
| | | | | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

Bit 31:12 保留，必须保持复位值。

Bit 11:8 **TI2SEL[3:0]：**选择 tim_ti2[15:0]输入（Selects tim_ti2[15:0] input）

0000：TIM15_CH2 输入（tim_ti2_in0）

0001：tim_ti2_in1

...

1111：tim_ti2_in15

参考章节：TIM15/TIM16/TIM17 引脚和内部信号获得关联列表。

Bit 7:4 保留，必须保持复位值。

Bit 3:0 **TI1SEL[3:0]：**选择 tim_ti1[15:0]输入（Selects tim_ti1[15:0] input）

0000：TIM15_CH1 输入（tim_ti1_in0）

0001：tim_ti1_in1

...

1111：tim_ti1_in15

参考章节：TIM15/TIM16/TIM17 引脚和内部信号获得关联列表。

### 24.7.19 TIM15 复用功能选择寄存器 1（TIM15_AF1）

偏移地址：0x060

复位值：0x0000 0001

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BKCMP4P | BKCMP3P | BKCMP2P | BKCMP1P | BKINP | Reserved | | | | BKCMP4E | BKCMP3E | BKCMP2E | BKCMP1E | BKINE |
| | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw |

*注：有关产品特定实现的信息，请参考章节：*TIM15/TIM16/TIM17 *引脚和内部信号。*

Bit 31:14 保留，必须保持复位值。

Bit 13 **BKCMP4P：**tim_brk_cmp4 输入极性（tim_brk_cmp4 input polarity）

此位选择 tim_brk_cmp2 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp4 输入高电平有效

1：tim_brk_cmp4 输入低电平有效

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 12 **BKCMP3P：**tim_brk_cmp3 输入极性（tim_brk_cmp3 input polarity）

此位选择 tim_brk_cmp3 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp3 输入高电平有效

1：tim_brk_cmp3 输入低电平有效

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 11 **BKCMP2P：**tim_brk_cmp2 输入极性（tim_brk_cmp2 input polarity）

此位选择 tim_brk_cmp2 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp2 输入高电平有效

1：tim_brk_cmp2 输入低电平有效

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 10 **BKCMP1P**：tim_brk_cmp1 输入极性（tim_brk_cmp1 input polarity）

此位选择 tim_brk_cmp1 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp1 输入高电平有效

1：tim_brk_cmp1 输入低电平有效

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 9 **BKINP**：TIMx_BKIN 输入极性（TIMx_BKIN input polarity）

该位选择 TIMx_BKIN 备用功能输入敏感度。必须对其进行编程

0：TIMx_BKIN 输入电平有效

1：TIMx_BKIN 输入低电平有效

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 8:5 保留，必须保持复位值。

Bit 4 **BKCMP4E**：tim_brk_cmp4 使能（tim_brk_cmp4 enable）

该位用于使能或禁用 tim_brk_cmp4 输入。如果该位设置为 1，则 tim_brk_cmp4 输入将被使能，否则将被禁用。

0：tim_brk_cmp4 输入禁用

1：tim_brk_cmp4 输入使能

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 4 **BKCMP3E**：tim_brk_cmp3 使能（tim_brk_cmp3 enable）

该位用于使能或禁用 tim_brk_cmp3 输入。如果该位设置为 1，则 tim_brk_cmp3 输入将被使能，否则将被禁用。

0：tim_brk_cmp3 输入禁用

1：tim_brk_cmp3 输入使能

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 2 **BKCMP2E**：tim_brk_cmp2 使能（tim_brk_cmp2 enable）

该位用于使能或禁用 tim_brk_cmp2 输入。如果该位设置为 1，则 tim_brk_cmp2 输入将被使能，否则将被禁用。

0：tim_brk_cmp2 输入禁用

1：tim_brk_cmp2 输入使能

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 1 **BKCMP1E**：tim_brk_cmp1 使能（tim_brk_cmp1 enable）

该位用于使能或禁用 tim_brk_cmp1 输入。如果该位设置为 1，则 tim_brk_cmp1 输入将被使能，否则将被禁用。

0：tim_brk_cmp1 输入禁用

1：tim_brk_cmp1 输入使能

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 0 **BKINE**：TIMx_BKIN 输入使能（TIMx_BKIN input enable）

该位用于使能或禁用 TIMx_BKIN alternate function 输入作为定时器的 tim_brk 输入。TIMx_BKIN 输入与其他 tim_brk 源进行“或”操作。

0：TIMx_BKIN 输入禁用

1：TIMx_BKIN 输入使能

*注：只要编程了*LOCK *（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

### 24.7.20 TIM15 复用功能选择寄存器 2（TIM15_AF2）

偏移地址：0x064

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | OCRSEL[2:0] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

Bit 31:19 保留，必须保持复位值。

Bit 18:16 **OCRSEL[2:0]：**ocref_clr 源选择（ocref_clr source selection）

这些位选择 ocref_clr 输入源。

000：tim_ocref_clr0

001：tim_ocref_clr1

010：tim_ocref_clr2

011：tim_ocref_clr3

100：tim_ocref_clr4

101：tim_ocref_clr5

110：tim_ocref_clr6

111：tim_ocref_clr7

参考章节：TIM15/TIM16/TIM17 引脚和内部信号获取更多特定产品信息。

*注：只要编程了*LOCK*（*TIM15_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 15:0 保留，必须保持复位值。

### 24.7.21 TIM15 DMA 控制寄存器（TIM15_DCR）

偏移地址：0x3DC

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DBL[4:0] | | | | | Reserved | | | DBA[4:0] | | | | |
| | | | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持复位值。

Bit 12:8 **DBL[4:0]：**DMA 突发长度（DMA burst length）

这个 5 位向量定义了 DMA 传输的长度（当对 TIM15_DMAR 地址进行读或写访问时，定时器识别一次突发传输），即传输次数。传输可以是半字或字节（见下面的示例）。

00000：1 次传输

00001：2 次传输

00010：3 次传输

...

10001：18 次传输

Bit 7:5 保留，必须保持复位值。

Bit 4:0 **DBA[4:0]：**DMA 基地址（DMA base address）

这个 5 位向量定义了 DMA 传输的基地址（当通过 TIM15_DMAR 地址进行读/写访问时）。DBA 是从 TIM15_CR1 寄存器的地址开始的偏移量。

00000：TIM15_CR1,

00001：TIM15_CR2,

00010：TIM15_SMCR,

...

### 24.7.22 TIM15 用于完整传输的 DMA 地址（TIM15_DMAR）

偏移地址：0x3E0

复位值：0x0000 0000

| 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 31:0 **DMAB[31:0]：**用于突发访问的 DMA 寄存器（DMA register for burst accesses）

对 DMAR 寄存器进行的读或写操作将访问位于以下地址的寄存器：

(TIM15_CR1 地址) + (DBA + DMA 索引) × 4

其中，TIM15_CR1 地址是控制寄存器 1 的地址，DBA 是在 TIM15_DCR 寄存器中配置的 DMA 基地址，DMA 索引由 DMA 传输自动控制，其范围从 0 到 DBL（在 TIM15_DCR 中配置的 DBL）。

## 24.8 TIM16/TIM17 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。

### 24.8.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 16~17）

偏移地址：0x000

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DITHEN | UIFREMAP | Res | CKD[1:0] | | ARPE | Reserved | | | OPM | URS | UDIS | CEN |
| | | | rw | rw | | rw | rw | rw | | | | rw | rw | rw | rw |

Bit 15:13 保留，必须保持复位值。

Bit 12 **DITHEN**：使能抖动（Dithering enable）

0：抖动禁止

1：抖动使能

*注：*DITHEN *位只能在*CEN *位复位时修改*

Bit 11 **UIFREMAP**：UIF 状态位重映射（UIF status bit remapping）

0：无重映射。UIF 状态位未复制到 TIMx_CNT 寄存器位 31

1：重映射使能。UIF 状态位复制到 TIMx_CNT 寄存器位 31

Bit 10 保留，必须保持复位值。

Bit 9:8 **CKD[1:0]**：时钟分频（Clock division）

此位域指示定时器时钟（tim_ker_ck）频率以及数字滤波器（tim_tix）所使用的采样时钟（$t_{DTS}$）之间的分频比

00：$t_{DTS} = t_{tim_ker_ck}$

01：$t_{DTS} = 2 \times t_{tim_ker_ck}$

10：$t_{DTS} = 4 \times t_{tim_ker_ck}$

11：保留，不要设置成此值

Bit 7 **ARPE**：自动重载预装载使能（Auto-reload preload enable）

0：TIMx_ARR 寄存器不进行缓冲；

1：TIMx_ARR 寄存器进行缓冲。

Bit 6:4 保留，必须保持复位值。

Bit 3 **OPM**：单脉冲模式（One pulse mode）

0：计数器在发生更新事件时不会停止计数；

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

Bit 2 **URS**：更新请求源（Update request source）

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断或 DMA 请求。此类事件包括：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

Bit 1 **UDIS**：更新禁止（Update disable）

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新（UEV）事件可通过以下事件之一生成：

- 计数器上出/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

Bit 0 **CEN**：计数器使能（Counter enable）

0：禁止计数器；

1：使能计数器。

*注：只有事先通过软件将*CEN *位置*1*，才可以使用外部时钟、门控模式。而触发模式可通过硬件自动将*CEN *位置*1*。*

### 24.8.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 16~17）

偏移地址：0x004

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | OIS1N | OIS1 | Reserved | | | | CCDS | CCUS | Res | CCPC |
| | | | | | | rw | rw | | | | | rw | rw | | rw |

Bit 15:10 保留，必须保持复位值。

Bit 9 **OIS1N**：输出空闲状态 1（OC1N 输出）（Output idle state 1 (tim_oc1n output)）

0：当 MOE=0 时，经过死区时间后 tim_oc1n=0

1：当 MOE=0 时，经过死区时间后 tim_oc1n=1

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3*，此位即无法修改。*

Bit 8 **OIS1**：输出空闲状态 1（OC1 输出）（Output idle state 1 (tim_oc1 output)）

0：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=0

1：当 MOE=0 时，（如果 OC1N 有效，则经过死区时间之后）tim_oc1=1

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别* 1、2 *或*3*，此位即无法修改。*

Bit 7:4 保留，必须保持复位值。

Bit 3 **CCDS**：捕获/比较 DMA 选择（Capture/compare DMA selection）

0：发生 CCx 事件时发送 CCx DMA 请求

1：发生更新事件时发送 CCx DMA 请求

Bit 2 **CCUS**：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(CCPC=1)进行预装载，仅通过将 COMG 位置 1 来对这些位进行更新；

1：如果捕获/比较控制位(CCPC=1)进行预装载，可通过将 COMG 位置 1 或 TRGI 的上升沿对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。。*

Bit 1 保留，必须保持复位值。

Bit 0 **CCPC**：捕获/比较预装载控制（Capture/compare preloaded control）

0：CCxE，CCxNE 和 OCxM 位未进行预装载；

1：CCxE，CCxNE 和 OCxM 位进行了预装载，写入这些位后，仅当发生换向事件（COM）（COMG 位置 1 或在 TRGI 上检测到上升沿，取决于 CCUS 位）时才会对这些位进行更新。

*注：此位仅对具有互补输出的通道有效。*

### 24.8.3 TIMx DMA/中断使能寄存器（TIMx_DIER）（x = 16~17）

偏移地址：0x00C

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CC1DE | UDE | BIE | Res | COMIE | Reserved | | | CC1IE | UIE |
| | | | | | | rw | rw | rw | | rw | | | | rw | rw |

Bit 15:10 保留，必须保持复位值。

Bit 9 **CC1DE**：捕获/比较 1 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止 CC1 DMA 请求；

1：使能 CC1 DMA 请求。

Bit 8 **UDE**：更新 DMA 请求使能（Capture/Compare 1 DMA request enable）

0：禁止更新 DMA 请求；

1：使能更新 DMA 请求。

Bit 7 **BIE**：刹车中断使能（Break interrupt enable）

0：禁止刹车中断；

1：使能刹车中断。

Bit 6 保留，必须保持复位值。

Bit 5 **COMIE**：COM 中断使能（COM interrupt enable）

0：禁止 COM 中断；

1：使能 COM 中断。

Bit 4:2 保留，必须保持复位值。

Bit 1 **CC1IE**：捕获/比较 1 中断使能（Capture/Compare 1 interrupt enable）

0：禁止 CC1 中断；

1：使能 CC1 中断。

Bit 0 **UIE**：更新中断使能（Update interrupt enable）

0：禁止更新中断；

1：使能更新中断。

### 24.8.4 TIMx 状态寄存器（TIMx_SR）（x = 16~17）

偏移地址：0x010

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CC1OF | Res | BIF | Res | COMIF | Reserved | | | CC1IF | UIF |
| | | | | | | rc_w0 | | rc_w0 | | rc_w0 | | | | rc_w0 | rc_w0 |

Bit 15:10 保留，必须保持复位值。

Bit 9 **CC1OF**：捕获/比较 1 重复捕获标志

仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。

0：未检测到重复捕获

1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

Bit 8 保留，必须保持复位值。

Bit 7 **BIF**：刹车中断标志（Break interrupt flag）

只要 tim_brk 输入变为有效状态，此标志便由硬件置 1。刹车输入无效后可通过软件对其清零。

0：未发生刹车事件。

1：在刹车输入上检测到有效电平。

Bit 6 保留，必须保持复位值。

Bit 5 **COMIF**：COM 中断标志（COM interrupt flag）

此标志在发生 COM 事件时（捕获/比较控制位 CCxE、CCxNE 和 OCxM 已更新时）由硬件置 1。但需要通过软件清零。

0：未发生 COM 事件。

1：COM 中断挂起。

Bit 4:2 保留，必须保持复位值。

Bit 1 **CC1IF**：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag）

此标志由硬件设置，它可以通过软件（输入捕获或输出比较模式）或读取 TIMx_CCR1 寄存器（仅输入捕获模式）来清除。

0：未比较匹配/无输入捕获发生

1：发生比较匹配或输入捕获

**如果通道 CC1 配置为输出：**当计数器与比较值匹配时，此标志由硬件置 1，当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。在中心对齐模式下有 3 种可能的标志设置选项，请参考 TIMx_CR1 寄存器中的 CMS 位以获取完整描述。

**如果通道 CC1 配置为输入：**当在 TIMx_CCR1 寄存器中捕获到计数器值时（在 IC1 上检测到边沿，具体取决于在 TIMx_CCER 中通过 CC1P 和 CC1NP 位设置定义的边沿灵敏度），会设置此位。

Bit 0 **UIF**：更新中断标志（Update interrupt flag）

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– TIMx_CR1 寄存器中的 UDIS=0，并且重复计数器值上溢或下溢时（重复计数器= 0 时更新）。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

### 24.8.5 TIMx 事件产生寄存器（TIMx_EGR）（x = 16~17）

偏移地址：0x014

复位：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BG | Res | COMG | Reserved | | | CC1G | UG |
| | | | | | | | | w | | rw | | | | w | w |

Bit 15:8 保留，必须保持复位值。

Bit 7 **BG**：刹车生成（Break generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：生成刹车事件。MOE 位清零且 BIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

Bit 6 保留，必须保持复位值。

Bit 5 **COMG**：捕获/比较控制更新生成（Capture/Compare control update generation）

该位可通过软件置 1，并由硬件自动清零

0：不执行任何操作；

1：CCPC 位置 1 时，可更新 CCxE、CCxNE 和 OCxM 位。

*注：此位仅对具有互补输出的通道有效。*

Bit 4:2 保留，必须保持复位值。

Bit 1 **CC1G**：捕获/比较 1 生成（Capture/Compare 1 generation）

此位由软件置 1 以生成事件，并由硬件自动清零。

0：不执行任何操作；

1：通道 1 上生成捕获/比较事件：

**若通道 CC1 配置为输出：**

使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。

**如果通道 CC1 配置为输入：**

TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

Bit 0 **UG**：更新生成（Update generation）

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作；

1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。

### 24.8.6 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 16~17）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输入捕获模式（本节）或输出比较模式（下节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | IC1F[3:0] | | | | IC1PSC[1:0] | | CC1S[1:0] | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

**输入捕获模式**

Bit 31:8 保留，始终读为 0。

Bit 7:4 **IC1F[3:0]：**输入捕获 1 滤波器（Input capture 1 filter）

此位域可定义 tim_ti1 输入的采样频率和适用于 tim_ti1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个事件才视为一个有效边沿：

0000：无滤波器，按 $f_{DTS}$ 频率进行采样

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

Bit 3:2 **IC1PSC[1:0]：**输入/捕获 1 预分频器（Input capture 1 prescaler）

此位域定义 CC1 输入（tim_ic1）的预分频比。

只要 CC1E=0（TIMx_CCER 寄存器），预分频器便立即复位。

00：无预分频器，捕获输入上每检测到一个边沿便执行捕获

01：每发生 2 个事件便执行一次捕获；

10：每发生 4 个事件便执行一次捕获；

11：每发生 8 个事件便执行一次捕获。

Bit 1:0 **CC1S[1:0]：**捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道被配置为输出；

01：CC1 通道被配置为输入，tim_ic1 映射在 tim_ti1 上；

其他：保留

*注：仅当通道关闭时（*TIMx_CCER *中的*CC1E = 0*），才可向*CC1S *位写入数据。*

### 24.8.7 TIMx 捕获/比较模式寄存器 1 [复用]（TIMx_CCMR1）（x = 16~17）

偏移地址：0x018

复位值：0x0000 0000

相同的寄存器可以用于输出比较模式（本节）或输入捕获模式（上节）。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。可以独立组合这两种模式（例如，通道 1 用作输入捕获模式而通道 2 用作输出比较模式）

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | OC1M[3] |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | 0C1CE | 0C1M[2:0] | | | 0C1PE | 0C1FE | CC1S[1:0] | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

**输出比较模式**

Bit 31:17 保留，必须保持复位值。

Bit 15:8 保留，必须保持复位值。

Bit 7 **OC1CE：**输出比较 1 清零使能（Output Compare 1 clear enable）

0：tim_oc1ref 不受 tim_ocref_clr 输入信号影响；

1：tim_oc1ref 在 tim_ocref_clr 输入信号中被检测到高电平，tim_oc1ref 立即清零。

Bit 16, 6:4 **OC1M[3:0]：**输出比较 1 模式（Output Compare 1 mode）

这些位定义提供 tim_oc1 和 tim_oc1n 的输出参考信号 tim_oc1ref 的行为。tim_oc1ref 为高电平有效，而 tim_oc1 和 tim_oc1n 的有效电平则取决于 CC1P 位和 CC1NP 位。

0000：**冻结** — 输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

0001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为高电平。

0010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1（TIMx_CCR1）匹配时，tim_oc1ref 信号强制变为低电平。

0011：**翻转** – TIMx_CNT=TIMx_CCR1 时，tim_oc1ref 发生翻转。

0100：**强制变为无效电平** – tim_oc1ref 强制变为低电平。

0101：**强制变为有效电平** – tim_oc1ref 强制变为高电平。

0110：**PWM 模式 1** – 在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

0111：**PWM 模式 2** – 在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。

其他：保留

*注：只要编程了*LOCK*（*TIMx_BDTR *寄存器中的*LOCK *位）级别*3 *且*CC1S=00*（通道配置为输出），这些位即无法修改。*

*在* PWM *模式*1 *或*2 *中，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“*PWM*”模式时，*tim_ocref_clr *电平才会发生更改。*

*对于具有互补输出的通道，此位域是预加载的。如果*TIMx_CR2 *寄存器中*CCPC *位被设置，则*OC1M *有效位仅在生成*COM *事件时从预加载位获取新值。*

Bit 3 **OC1PE**：输出比较 1 预装载使能（Output Compare 1 preload enable）

0：禁止与 TIMxCCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到有效寄存器中。

*注：只要编程了*LOCK*（*TIMx_BDTR *寄存器中的*LOCK *位）级别*3 *且*CC1S=00*（通道配置为输出），这些位即无法修改。*

Bit 2 **OC1FE**：输出比较 1 快速使能（Output Compare 1 fast enable）

此位减少了触发事件和定时器输出上的转换之间的延迟。它必须在单脉冲模式（在 TIMx_CR1 寄存器中设置 OPM 位）中使用，以使输出脉冲在启动触发后尽快开始。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

Bit 1:0 **CC1S[1:0]**：捕获/比较 1 选择（Capture/Compare 1 selection）

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，tim_ic1 映射到 tim_ti1 上

其他：保留

*注：仅当通道关闭时（*TIMx_CCER *中的* CC1E = 0*），才可向* CC1S *位写入数据*

### 24.8.8 TIMx 捕获/比较使能寄存器（TIMx_CCER）（x = 16~17）

偏移地址：0x020

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CC1NP | CC1NE | CC1P | CC1E |
| | | | | | | | | | | | | rw | rw | rw | rw |

Bit 15:4 保留，必须保持复位值。

Bit 3 **CC1NP**：输入/捕获 1 输出极性（Capture/Compare 1 output polarity）

**CC1 通道配置为输出：**

0：tim_oc1n 高电平有效。

1：tim_oc1n 低电平有效。

**CC1 通道配置为输入：**

此位与 CC1P 配合使用，用以定义 tim_ti1fp1 的极性。请参见 CC1P 说明。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *或*3 *且*CC1S=00 *（通道配置为输出），此位立即变为不可写状态*

*此位将在具有互补输出的通道上进行预装载。如果* TIMx_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1NP *有效位才会从预装载位获取新值。*

Bit 2 **CC1NE**：输入/捕获 1 互补输出使能（Capture/Compare 1 complementary output enable）

0：关闭 – OC1N 未激活。OC1N 电平是 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的函数。

1：开启 – 在相应输出引脚上输出 OC1N 信号，具体取决于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位。

Bit 1 **CC1P**：输入/捕获 1 输出极性（Capture/Compare 1 output polarity）

0：OC1 高电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

1：OC1 低电平有效（输出模式）/边沿灵敏度选择（输入模式，参见以下）

**CC1 通道配置为输入，**CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。

CC1NP=0，CC1P=0：非反相/上升沿触发。电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=0，CC1P=1：反相/下降沿触发。电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。

CC1NP=1，CC1P=1：非反相/上升沿和下降沿均触发。电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

CC1NP=1，CC1P=0：此配置是保留的，不应该使用此配置。

*注：只要编程了*LOCK *（*TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *或*3 *且*CC1S=00 *（通道配置为输出），此位立即变为不可写状态*

*此位将在具有互补输出的通道上进行预装载。如果* TIMx_CR2 *寄存器中的*CCPC *位置*1*，则仅当生成换向事件时，*CC1P *有效位才会从预装载位获取新值。*

Bit 0 **CC1E**：输入/捕获 1 输出使能（Capture/Compare 1 output enable）

0：禁止捕获模式/OC1 未激活（参见以下）

1：使能捕获模式/ OC1 信号输出到对应的输出引脚

**当 CC1 通道配置为输出，**C1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值，不论 CC1E 位状态。请参考下表获得更多细节

表 24.15 带刹车功能的互补输出通道 tim_oc1 和 tim_oc1n 的控制位（TIM16/TIM17）

| 控制位 | | | | | 输出状态 (1) | |
|---|---|---|---|---|---|---|
| MOE 位 | OSSI 位 | OSSR 位 | CC1E 位 | CC1NE 位 | tim_oc1 输出状态 | tim_oc1n 输出状态 |
| 1 | X | X | 0 | 0 | 禁止输出（不由定时器驱动：高阻态）<br>tim_oc1 = 0，tim_oc1n = 0 | |
| | | 0 | 0 | 1 | 禁止输出（不由定时器驱动：高阻态）tim_oc1 = 0 | tim_oc1ref + 极性，tim_oc1n = tim_oc1ref 异或 CC1NP |
| | | 0 | 1 | 0 | tim_oc1ref + 极性，tim_oc1 = tim_oc1ref 异或 CC1P | 禁止输出（不由定时器驱动：高阻态） tim_oc1n=0 |
| | | X | 1 | 1 | tim_oc1ref + 极性 + 死区 | tim_oc1ref 互补项（对 tim_oc1ref 进行“非”运算）+ 极性 + 死区 |
| | | 1 | 0 | 1 | 关闭状态（输出使能为无效状态）<br>tim_oc1 = CC1P | tim_oc1ref + 极性，tim_oc1n = tim_oc1ref 异或 CC1NP |
| | | 1 | 1 | 0 | tim_oc1ref + 极性，tim_oc1 = tim_oc1ref 异或 CC1P | 关闭状态（输出使能为无效状态）<br>tim_oc1n = CC1NP |
| 0 | 0 | X | X | X | 禁止输出（不再由定时器驱动）。<br>输出状态由 GPIO 控制器定义，可以是高电平、低电平或高阻态。 | |
| | 1 | | 0 | 0 | | |
| | | | 0 | 1 | 关闭状态（输出使能为无效状态）<br>异步：tim_oc1 = CC1P、tim_oc1n = CC1NP<br>如果存在时钟：在死区后 tim_oc1 = OIS1 且 tim_oc1n = OIS1N，假定 OIS1 和 OIS1N 并没有都设置成 tim_oc1 及 tim_oc1n 的有效电平。 | |
| | | | 1 | 0 | | |
| | | | 1 | 1 | | |

1. 如果一个通道的两个输出均未使用（由 GPIO 接管控制控制），则 OIS1、OIS1N、CC1P 和 CC1NP 位必须保持清零状态。

*注：　与互补通道* tim_oc1 *和* tim_oc1n *相连的外部* I/O *引脚的状态取决于通道* tim_oc1 *和* tim_oc1n *的状态以及* GPIO *寄存器。*

### 24.8.9 TIMx 计数器（TIMx_CNT）（x = 16~17）

偏移地址：0x024

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIFCPY | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 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 31 **UIFCPY：**复制 UIF

该位是 TIMx_SR 寄存器中 UIF 位的一个只读副本。如果 TIMx_CR1 上的 UIFREMAP 位被清零，则位 31 被保留并读为 0。

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CNT[15:0]：**计数器的值（Counter value）

在非抖动模式下（DITHEN = 0）

该寄存器（CNT[15:0]）保存的是计数器的值。

在抖动模式下（DITHEN = 1）

该寄存器（CNT[15:0]）只保存计数器的不抖动部分。抖动部分不可用。

### 24.8.10 TIMx 预分频器（TIMx_PSC）（x = 16~17）

偏移地址：0x028

复位值：0x0000

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

Bit 15:0 **PSC[15:0]：**预分频器的值（Prescaler value）

计数器时钟频率（$f_{tim_cnt_ck}$）等于 $f_{tim_psc_ck}$ / (PSC[15:0] + 1)。

PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

### 24.8.11 TIMx 自动重装载寄存器（TIMx_ARR）（x = 16~17）

偏移地址：0x02C

复位：0x0000 FFFF

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | 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 31:20 保留，必须保持复位值。

Bit 19:0 **ARR[19:0]：**自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

当自动重载值为空时，计数器不工作。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是自动重新加载的值

在抖动模式下（DITHEN = 1）

该寄存器存储的是 ARR[19:4]中的整数部分，而 ARR[3:0]位域存储的是抖动的部分

### 24.8.12 TIMx 重复计数寄存器（TIMx_RCR）（x = 16~17）

偏移地址：0x030

复位值：0x0000

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

Bit 15:8 保留，必须保持复位值。

Bit 7:0 **REP[7:0]**：重复计数器的值（Repetition counter value）

使能预装载寄存器时，用户可通过这些位设置比较寄存器的更新频率（即从预装载寄存器向活动寄存器周期性传输数据）；使能更新中断时，也可设置更新中断的生成速率。

与 REP_CNT 相关的减计数器每次计数到 0 时，都将生成一个更新事件并且计数器从 REP 值重新开始计数。由于只有生成重复更新事件 U_RC 时，REP_CNT 才会重载 REP 值，因此在生成下一重复更新事件之前，无论向 TIMx_RCR 寄存器写入何值都无影响。

这意味着 PWM 模式下（REP+1）相当于：

– 边沿对齐模式下的 PWM 周期数；

– 中心对称模式下的 PWM 半周期数；

### 24.8.13 TIMx 捕获/比较寄存器 1（TIMx_CCR1）（x = 16~17）

偏移地址：0x034

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCR1 [19:16] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCR1 [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:20 保留，必须保持复位值。

Bit 19:0 **CCR1[19:0]：**捕获/比较 1 值（Capture/Compare 1 value）

**若 CC1 通道配置为输出：**

CCR1 为要装载到实际捕获/比较 1 寄存器的值（预装载值）。

如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，写入的数值会被直接传输至当前寄存器中。否则只有发生更新事件时，预装载值才会复制到活动捕获/比较 1 寄存器中。实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 tim_oc1 输出上发出信号的值。

在非抖动模式下（DITHEN = 0）

该寄存器存储的是 CCR1[15:0]中的比较值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储的是 CCR1[19:4]中的整数部分。CCR1[3:0]位域存储的是抖动的部分。

**若 CC1 通道配置为输入：**

CCR1 为上一个输入捕获 1 事件（tim_ic1）发生时的计数器值。TIMx_CCR1 是只读的并且不可编程。

在非抖动模式下（DITHEN = 0）

CCR1[15:0]位存储了捕获值。CCR1[19:16]位会被重置。

在抖动模式下（DITHEN = 1）

该寄存器存储捕获在 CCR1[19:4]中。CCR1[3:0]位会被重置。

### 24.8.14 TIMx 刹车和死区寄存器（TIMx_BDTR）（x = 16~17）

偏移地址：0x044

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | BKBID | Res | BK DSRM | Reserved | | | | | | 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 |

*注：由于可以根据*LOCK *配置锁定位*BKBID、BKF[3:0]、AOE、BKP、BKE、OSSI、OSSR *和*DTG[7:0]*的写操作，因此必须在第一次对* TIMx_BDTR *寄存器执行写访问时对这些位进行配置。*

Bit 31:29 保留，必须保持复位值。

Bit 28 **BKBID：**刹车双向（Break bidirectional）

0：刹车输入 tim_brk 处于输入模式

1：刹车输入 tim_brk 处于双向模式

在双向模式（BKBID 置 1）下，刹车输入同时配置为输入模式和开漏输出模式。任何有效的 break 事件会在刹车输入上输出低电平，以向外部设备指示内部 break 事件。

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 27 保留，必须保持复位值。

Bit 26 **BKDSRM：**刹车解除保护（Break disarm）

0：刹车输入 tim_brk 处于保护状态

1：刹车输入 tim_brk 解除保护

当没有刹车源处于有效状态时，该位由硬件清零。

软件需置位 BKDSRM 以释放双向输出控制（开漏输出处于高阻态），并轮询直至硬件将其清零，表明故障状态已消失。

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 25:20 保留，必须保持复位值。

Bit 19:16 **BKF[3:0]：**刹车滤波器（Break filter）

此位域可定义 tim_brk 输入的采样频率和适用于 tim_brk 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，tim_brk 异步工作

0001：$f_{SAMPLING}=f_{tim_ker_ck}$，N=2。

0010：$f_{SAMPLING}=f_{tim_ker_ck}$，N=4。

0011：$f_{SAMPLING}=f_{tim_ker_ck}$，N=8。

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6。

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8。

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6。

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8。

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6。

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8。

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5。

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6。

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8。

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5。

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6。

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8。

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 15 **MOE**：主输出使能（Main output enable）

只要刹车输入（tim_brk）为有效状态，此位便由硬件异步清零。此位由软件置 1，也可根据 AOE 位状态自动置 1。此位仅对配置为输出的通道有效。

0：响应刹车事件或向 MOE 写入 0 时：tim_oc1 和 tim_oc1n 输出被禁止或被强制为空闲状态，具体取决于 OSSI 位。

1：如果 tim_oc1 和 tim_oc1n 输出的相应使能位（TIMx_CCER 寄存器中的 CC1E 和 CC1NE 位）均置 1，则使能 tim_oc1 和 tim_oc1n 输出。

有关详细信息，请参见 tim_oc1/tim_oc1n 使能说明（TIMx 捕获/比较使能寄存器(TIMx_CCER)(x=16~17)）。

Bit 14 **AOE**：自动输出使能（Automatic output enable）

0：MOE 只能由软件置 1

1：MOE 可由软件置 1，也可在发生下一更新事件时自动置 1（如果刹车输入 tim_brk 无效）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 13 **BKP**：刹车极性（Break polarity）

0：刹车输入 tim_brk 为低电平有效

1：刹车输入 tim_brk 为高电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 12 **BKE**：刹车使能（Break enable）

0：禁止刹车输入（tim_brk 及 tim_sys_brk 事件）；

1：使能刹车输入（tim_brk 及 tim_sys_brk 事件）。

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

*注：对该位执行任何写操作后，都需要经过*1 *个*APB *时钟周期的延迟才生效。*

Bit 11 **OSSR**：运行模式下关闭状态选择（Off-state selection for Run mode）

此位在 MOE=1 时作用于配置为输出模式且具有互补输出的通道。如果定时器中没有互补输出，则不存在 OSSR。

有关详细信息，请参见 tim_oc1/tim_oc1n 使能说明（TIMx 捕获/比较使能寄存器(TIMx_CCER)(x=16~17)）。

0：处于无效状态时，禁止 tim_oc1/tim_oc1n 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，一旦 CCxE=1 或 CCxNE=1，便使能 tim_oc1/tim_oc1n 输出并将其设为无效电平（输出仍由定时器控制）。

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *后，此位即无法修改。*

Bit 10 **OSSI**：空闲模式下的关闭状态选择（Off-state selection for Idle mode）

当由于刹车事件或软件写操作而使 MOE=0 时，此位作用于配置为输出的通道。

有关详细信息，请参见 tim_oc1/tim_oc1n 使能说明（TIMx 捕获/比较使能寄存器(TIMx_CCER)(x=16~17)）。

0：处于无效状态时，禁止 tim_oc1/tim_oc1n 输出（定时器释放输出控制，由强制高阻态的 GPIO 逻辑接管）。

1：处于无效状态时，首先将 tim_oc1/tim_oc1n 输出强制为其无效电平，然后在死区后将其强制为空闲电平。定时器始终控制输出。

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*2 *后，此位即无法修改。*

Bit 9:8 **LOCK[1:0]**：锁定设置（Lock configuration）

这些位用于针对软件错误提供写保护。

00：关闭锁定 – 不对任何位提供写保护。

01：锁定级别 1，此时无法对 TIMx_BDTR 寄存器中的 DTG 位、TIMx_CR2 寄存器中的 OISx 和 OISxN 位以及 TIMx_BDTR 寄存器中的 BKBID/BKE/BKP/AOE 位执行写操作。

10：锁定级别 2，此时无法对锁定级别 1 中适用的各位、CC 极性位（TIMx_CCER 寄存器中的 CCxP/CCxNP 位，只要通过 CCxS 位将相关通道配置为输出）以及 OSSR 和 OSSI 位执行写操作。

11：锁定级别 3，此时无法对锁定级别 2 中适用的各位、CC 控制位（TIMx_CCMRx 寄存器中的 OCxM 和 OCxPE 位，只要通过 CCxS 位将相关通道配置为输出）执行写操作。

*注：复位后只能对*LOCK *位执行一次写操作。对*TIMx_BDTR *寄存器执行写操作后其中的内容将冻结，直到下一次复位。*

Bit 7:0 **DTG[7:0]**：配置死区发生器（Dead-time generator setup）

此位域定义插入到互补输出之间的死区持续时间。DT 与该持续时间相对应。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位域即无法修改。*

### 24.8.15 TIMx 定时器死区寄存器 2（TIMx_DTR2）（x = 16~17）

偏移地址：0x054

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DTPE | DTAE |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 31:18 保留，必须保持复位值。

Bit 17 **DTPE：**死区时间预加载使能（Deadtime preload enable）

0：死区时间值未预加载

1：使能死区时间值预加载

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 16 **DTAE：**死区时间非对称使能（Deadtime asymmetric enable）

0：上升沿和下降沿的死区时间是相同的，并且使用 DTG[7:0]寄存器定义

1：上升沿的死区时间使用 DTG[7:0]寄存器定义，下降沿的死区时间使用 DTGF[7:0]位定义。

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

Bit 15:8 保留，必须保持复位值。

Bit 7:0 **DTGF[7:0]：**死区下降沿发生器（Dead-time falling edge generator setup）

这个位域定义了在下降沿时，互补输出之间插入死区时间的长度。

DTG[7:5]=0xx => DT= DTG[7:0] × $T_{dtg}$，其中 $T_{dtg}$ = $T_{DTS}$；

DTG[7:5]=10x => DT=（64+DTG[5:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 2 × $T_{DTS}$；

DTG[7:5]=110 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 8 × $T_{DTS}$；

DTG[7:5]=111 => DT=（32+DTG[4:0]）× $T_{dtg}$，其中 $T_{dtg}$ = 16 × $T_{DTS}$；

例：若 $T_{DTS}$ = 125ns（8MHZ），可能的死区时间为：

0 到 15875 ns（步长为 125 ns），

16 us 到 31750 ns（步长为 250 ns），

32 us 到 63us（步长为 1 us），

64 us 到 126 us（步长为 2 us）

*注：编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1、2 *或*3，*此位即无法修改。*

### 24.8.16 TIMx 定时器输入选择寄存器（TIMx_TISEL）（x = 16~17）

偏移地址：0x05C

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | TI1SEL[3:0] | | | |
| | | | | | | | | | | | | rw | rw | rw | rw |

Bit 31:4 保留，必须保持复位值。

Bit 3:0 **TI1SEL[3:0]**：选择 tim_ti1[15:0]输入（Selects tim_ti1[15:0] input）

0000：TIMx_CH1 输入（tim_ti1_in0）

0001：tim_ti1_in1

...

1111：tim_ti1_in15

参考章节：TIM15/TIM16/TIM17 引脚和内部信号获得关联列表。

### 24.8.17 TIMx 复用功能选择寄存器 1（TIMx_AF1）（x = 16~17）

偏移地址：0x060

复位值：0x0000 0001

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BKCMP4P | BKCMP3P | BKCMP2P | BKCMP1P | BKINP | Reserved | | | | BKCMP4E | BKCMP3E | BKCMP2E | BKCMP1E | BKINE |
| | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw |

*注：有关产品特定实现的信息，请参考章节：*TIM15/TIM16/TIM17 *引脚和内部信号。*

Bit 31:14 保留，必须保持复位值。

Bit 13 **BKCMP4P**：tim_brk_cmp4 输入极性（tim_brk_cmp4 input polarity）

此位选择 tim_brk_cmp2 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp4 输入高电平有效

1：tim_brk_cmp4 输入低电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 12 **BKCMP3P**：tim_brk_cmp3 输入极性（tim_brk_cmp3 input polarity）

此位选择 tim_brk_cmp3 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp3 输入高电平有效

1：tim_brk_cmp3 输入低电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 11 **BKCMP2P**：tim_brk_cmp2 输入极性（tim_brk_cmp2 input polarity）

此位选择 tim_brk_cmp2 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp2 输入高电平有效

1：tim_brk_cmp2 输入低电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 10 **BKCMP1P**：tim_brk_cmp1 输入极性（tim_brk_cmp1 input polarity）

此位选择 tim_brk_cmp1 输入敏感度。它必须与 BKP 极性位一起编程。

0：tim_brk_cmp1 输入高电平有效

1：tim_brk_cmp1 输入低电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 9 **BKINP**：TIMx_BKIN 输入极性（TIMx_BKIN input polarity）

该位选择 TIMx_BKIN 备用功能输入敏感度。必须对其进行编程

0：TIMx_BKIN 输入电平有效

1：TIMx_BKIN 输入低电平有效

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 8:5 保留，必须保持复位值。

Bit 4 **BKCMP4E**：tim_brk_cmp4 使能（tim_brk_cmp4 enable）

该位用于使能或禁用 tim_brk_cmp4 输入。如果该位设置为 1，则 tim_brk_cmp4 输入将被使能，否则将被禁用。

0：tim_brk_cmp4 输入禁用

1：tim_brk_cmp4 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 4 **BKCMP3E**：tim_brk_cmp3 使能（tim_brk_cmp3 enable）

该位用于使能或禁用 tim_brk_cmp3 输入。如果该位设置为 1，则 tim_brk_cmp3 输入将被使能，否则将被禁用。

0：tim_brk_cmp3 输入禁用

1：tim_brk_cmp3 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 2 **BKCMP2E**：tim_brk_cmp2 使能（tim_brk_cmp2 enable）

该位用于使能或禁用 tim_brk_cmp2 输入。如果该位设置为 1，则 tim_brk_cmp2 输入将被使能，否则将被禁用。

0：tim_brk_cmp2 输入禁用

1：tim_brk_cmp2 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 1 **BKCMP1E**：tim_brk_cmp1 使能（tim_brk_cmp1 enable）

该位用于使能或禁用 tim_brk_cmp1 输入。如果该位设置为 1，则 tim_brk_cmp1 输入将被使能，否则将被禁用。

0：tim_brk_cmp1 输入禁用

1：tim_brk_cmp1 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

Bit 0 **BKINE**：TIMx_BKIN 输入使能（TIMx_BKIN input enable）

该位用于使能或禁用 TIMx_BKIN alternate function 输入作为定时器的 tim_brk 输入。TIMx_BKIN 输入与其他 tim_brk 源进行“或”操作。

0：TIMx_BKIN 输入禁用

1：TIMx_BKIN 输入使能

*注：只要编程了*LOCK（TIMx_BDTR *寄存器中的*LOCK *位）级别*1，*此位即无法修改。*

### 24.8.18 TIMx 复用功能选择寄存器 2（TIMx_AF2）（x = 16~17）

偏移地址：0x064

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | OCRSEL[2:0] | | |
| | | | | | | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

Bit 31:19 保留，必须保持复位值。

Bit 18:16 **OCRSEL[2:0]：**tim_ocref_clr 源选择（tim_ocref_clr source selection）

这些位选择 tim_ocref_clr 输入源。

000：tim_ocref_clr0

001：tim_ocref_clr1

010：tim_ocref_clr2

011：tim_ocref_clr3

100：tim_ocref_clr4

101：tim_ocref_clr5

110：tim_ocref_clr6

111：tim_ocref_clr7

参考章节：TIM15/TIM16/TIM17 引脚和内部信号获取更多特定产品信息。

*注：只要编程了*LOCK*（*TIMx_BDTR *寄存器中的*LOCK *位）级别*1*，此位即无法修改。*

Bit 15:0 保留，必须保持复位值。

### 24.8.19 TIMx 选项寄存器 1（TIMx_OR1）（x = 16~17）

偏移地址：0x068

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | HSE32 EN |
| | | | | | | | | | | | | | | | rw |

Bit 31:1 保留，必须保持复位值。

Bit 0 **HSE32EN：**HSE 除 32 使能（HSE Divided by 32 enable）

该位用于使能 tim_ti_in3 的 HSE 除 32 的分频器。有关详细信息，请参阅表：与 tim_ti1 输入多路复用器的互联。

0：禁用 HSE 除 32

1：使能 HSE 除 32

### 24.8.20 TIMx DMA 控制寄存器（TIMx_DCR）（x = 16~17）

偏移地址：0x3DC

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DBL[4:0] | | | | | Reserved | | | DBA[4:0] | | | | |
| | | | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

Bit 31:13 保留，必须保持复位值。

Bit 12:8 **DBL[4:0]：** DMA 突发长度（DMA burst length）

这个 5 位向量定义了 DMA 传输的长度（当对 TIMx_DMAR 地址进行读或写访问时，定时器识别一次突发传输），即传输次数。传输可以是半字或字节（见下面的示例）。

00000：1 次传输

00001：2 次传输

00010：3 次传输

...

10001：18 次传输

Bit 7:5 保留，必须保持复位值。

Bit 4:0 **DBA[4:0]：** DMA 基地址（DMA base address）

这个 5 位向量定义了 DMA 传输的基地址（当通过 TIMx_DMAR 地址进行读/写访问时）。DBA 是从 TIMx_CR1 寄存器的地址开始的偏移量。

00000：TIMx_CR1,

00001：TIMx_CR2,

00010：TIMx_SMCR,

...

**样例**：让我们考虑以下传输：DBL=7 传输和 DBA=TIMx_CR1。在这种情况下，从 TIMx_CR1 地址开始，传输是到/从 7 个寄存器完成的

### 24.8.21 TIMx 用于完整传输的 DMA 地址（TIMx_DMAR）（x = 16~17）

偏移地址：0x3E0

复位值：0x0000 0000

| 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 31:0 **DMAB[31:0]：** 用于突发访问的 DMA 寄存器（DMA register for burst accesses）

对 DMAR 寄存器进行的读或写操作将访问位于以下地址的寄存器：

(TIMx_CR1 地址) + (DBA + DMA 索引) × 4

其中，TIMx_CR1 地址是控制寄存器 1 的地址，DBA 是在 TIMx_DCR 寄存器中配置的 DMA 基地址，DMA 索引由 DMA 传输自动控制，其范围从 0 到 DBL（在 TIMx_DCR 中配置的 DBL）。

# 25 基本定时器（TIM6/TIM7）

## 25.1 TIM6/TIM7 简介

基本定时器 TIM6/TIM7 包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。
他们可以用作通用定时器以生成时基。
基本定时器还可以用于触发数模转换器（DAC）。这是通过定时器的触发输出完成的。
这些定时器彼此完全独立，不共享任何资源。

## 25.2 TIM6/TIM7 主要功能

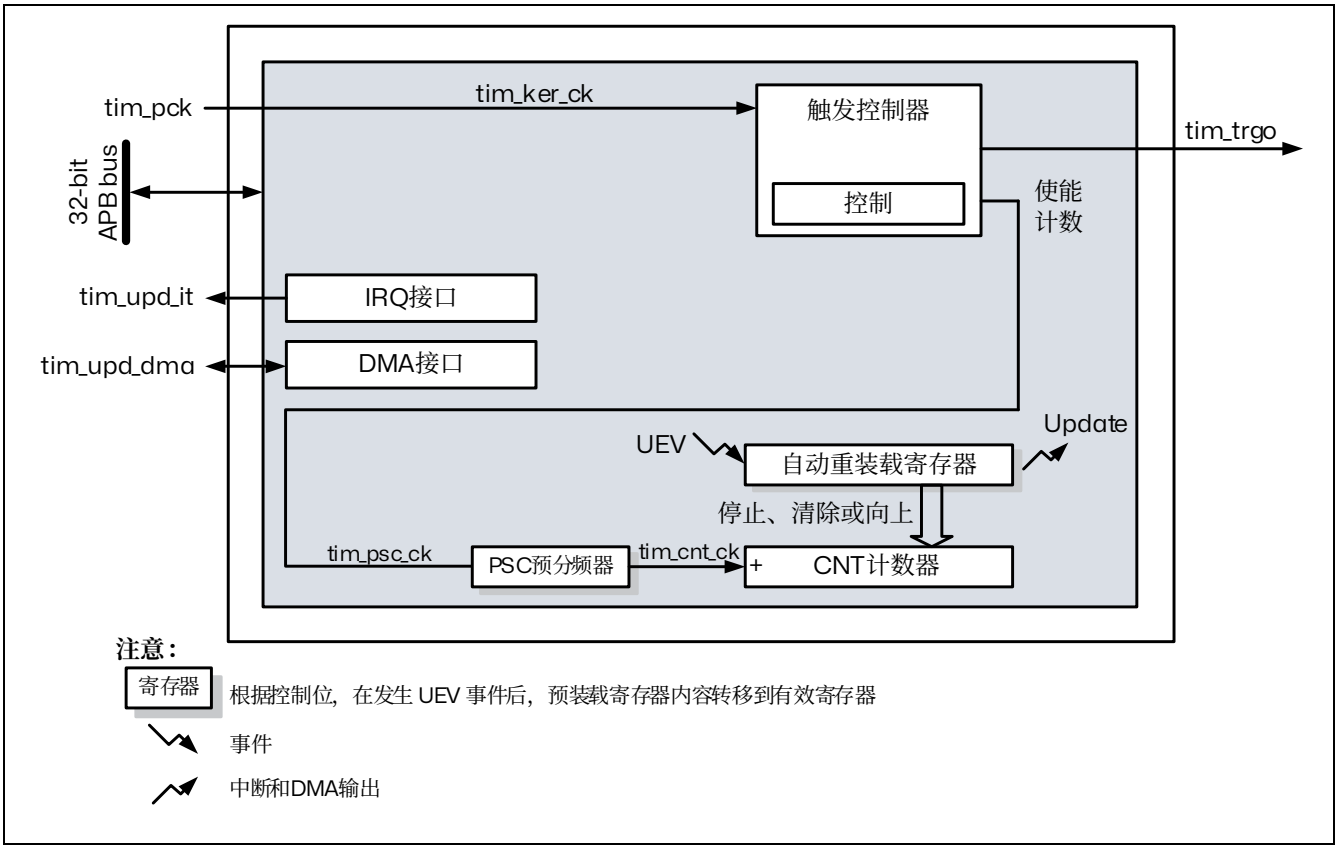
基本定时器（TIM6/TIM7）功能包括：

- 16 位自动重载递增计数器
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65536 之间
- 用于触发 DAC 的同步电路
- 发生如下更新事件时会生成中断/DMA 请求：计数器上溢

## 25.3 TIM6/TIM7 功能描述

### 25.3.1 框图

图 25.1 基本定时器框图



### 25.3.2 TIM6/TIM7 内部信号

本节中的下表格总结了 TIM 的输入和输出。

表 25.1 TIM 内部输入/输出信号

| 内部信号名称 | 信号类型 | 描述 |
|---|---|---|
| tim_pclk | 输入 | 定时器 APB 时钟。 |
| tim_ker_ck | 输入 | 定时器内核时钟。此时钟必须与 tim_plck 同步（来自同一源）。<br>时钟比率 tim_ker_ck/ tim_pclk 必须为整数：<br>1，2，3，...，16（最大值）。 |
| tim_trgo | 输出 | 内部触发器输出。此触发器可以触发其他片上外设（DAC）。 |
| tim_upd_it | 输出 | 定时器更新事件中断。 |
| tim_upd_dma | 输出 | 定时器更新 DMA 请求。 |

### 25.3.3 TIM6/TIM7 时钟

定时器总线接口由 tim_pclk APB 时钟驱动。

计数器时钟 tim_ker_ck 连接到 tim_pclk 输入。

CEN 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 tim_ker_ck 提供。

下图显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

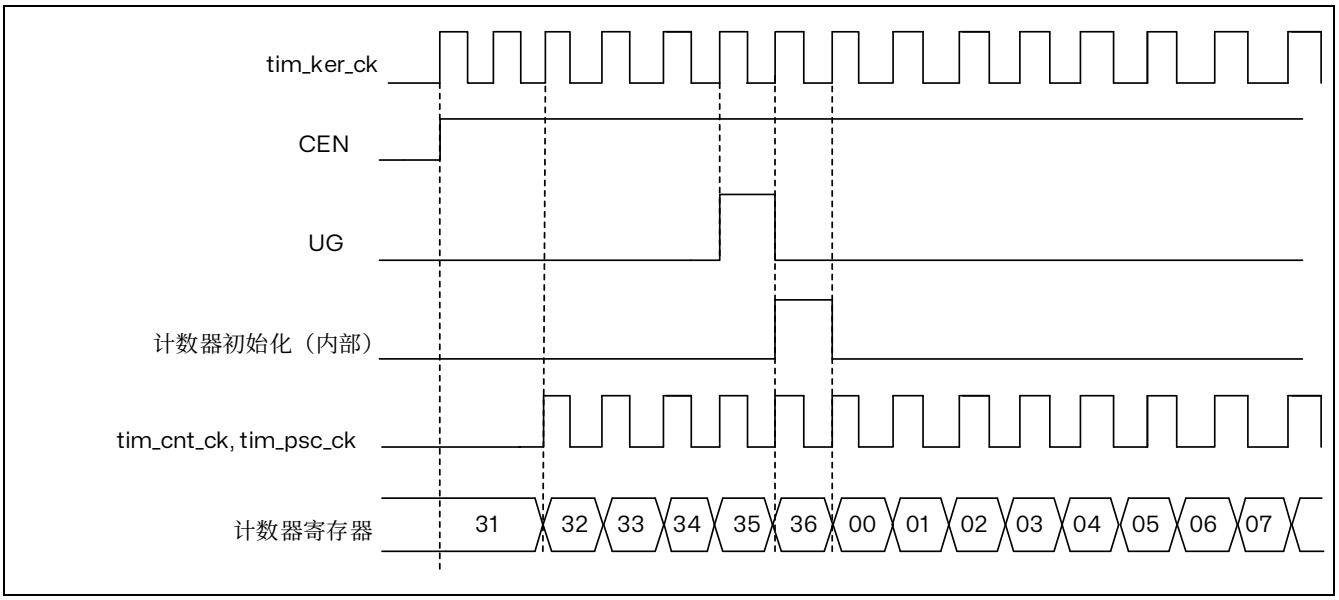



图 25.2 一般模式下的控制电路，内部时钟分频系数为 1

### 25.3.4 时基单元

可编程定时器的主要模块由一个 16 位递增计数器及其相关的自动重载寄存器组成。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

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）

自动重载寄存器是预装载的。每次尝试对自动重载寄存器执行读写操作时，都会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 UEV 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位(ARPE)。当计数器达到上溢值并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位（CEN）置 1 时，才会启动计数器。

请注意，实际的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

**预分频器描述**

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于 TIMx_PSC 控制寄存器有缓冲，因此可对预分频器进行实时更改。而新的预分频比将在下一更新事件发生时被采用。

下图给出了在预分频比发生实时变化时一些计数器行为的示例。

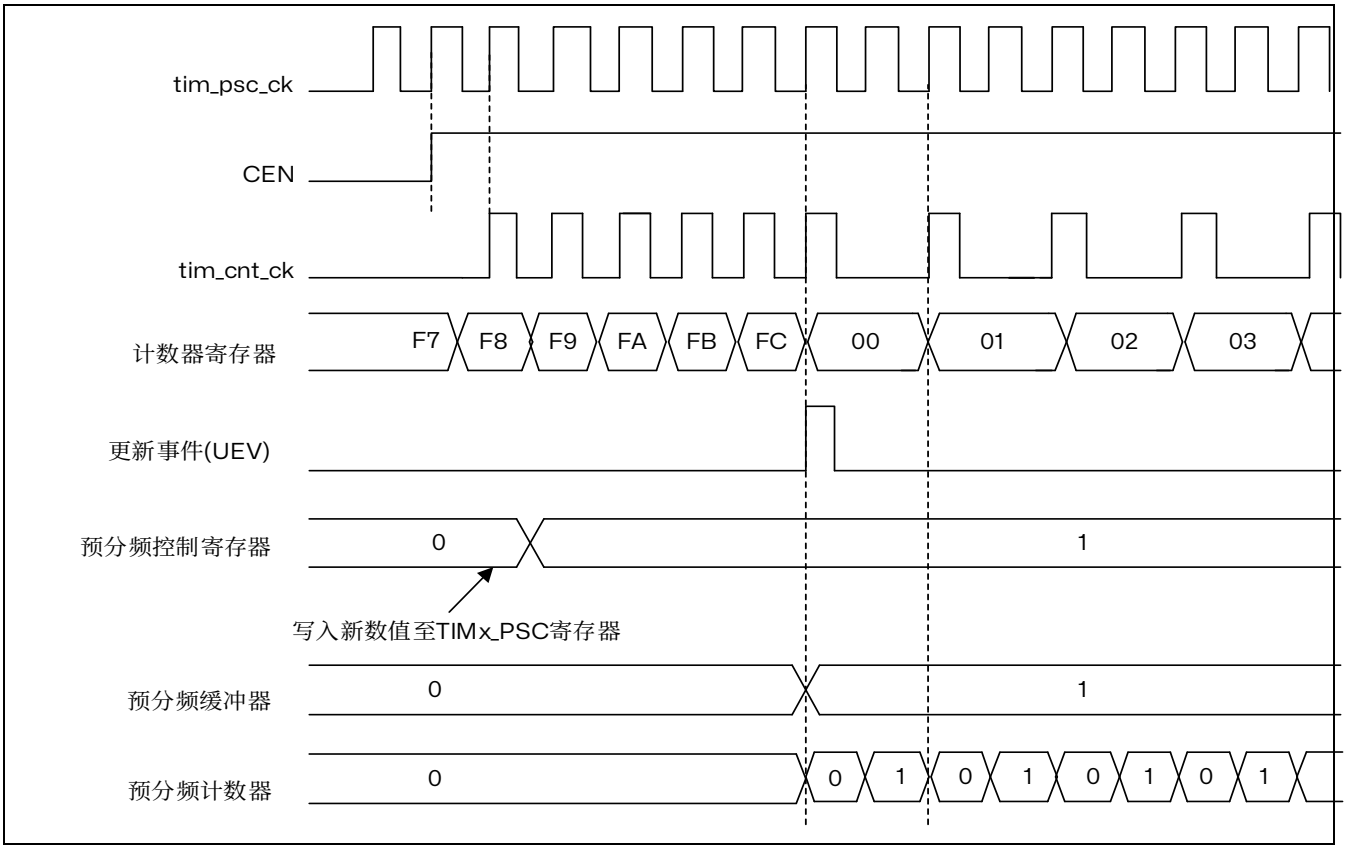



图 25.3 当预分频器的参数从 1 变到 2 时，计数器的时序图

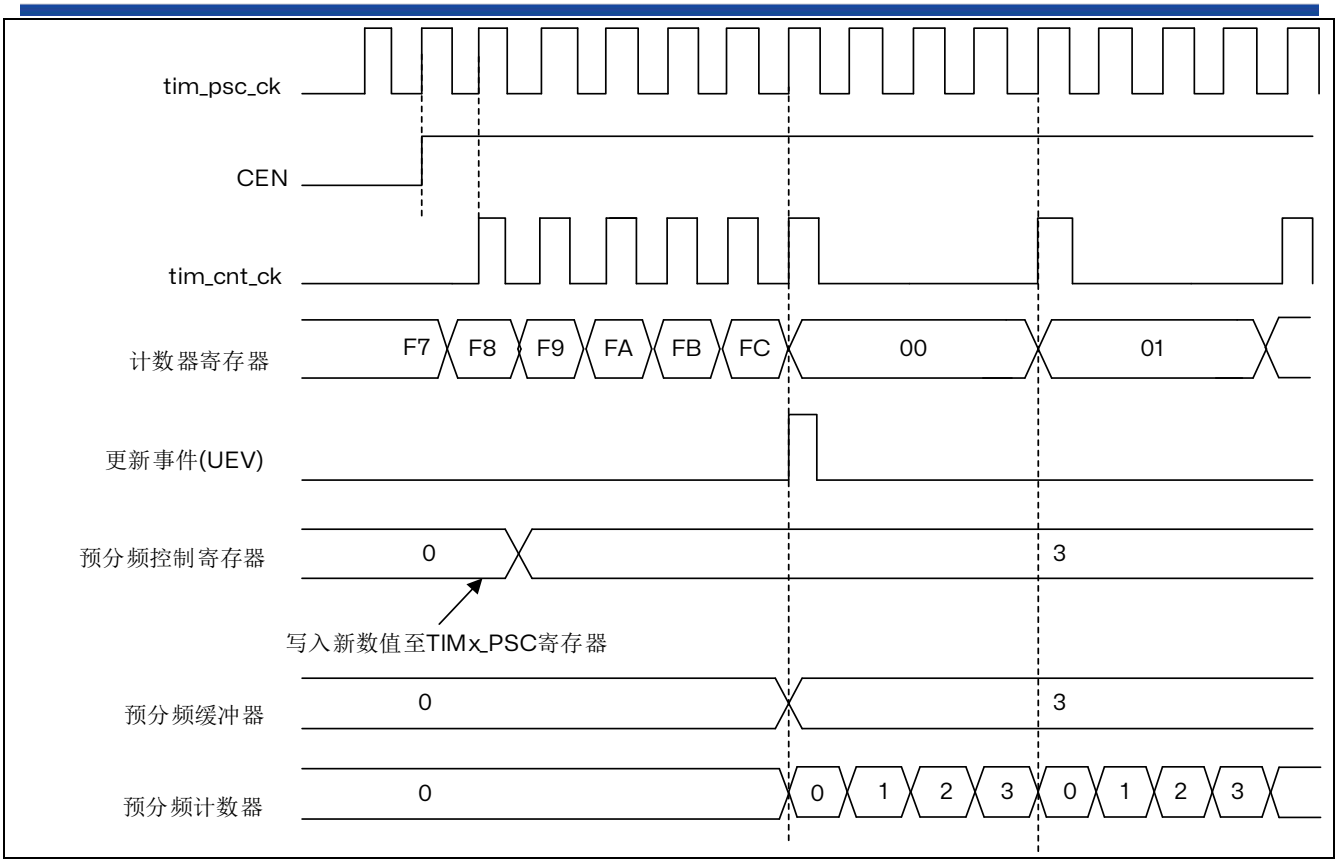



图 25.4 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 25.3.5 计数器模式

计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。这样，直到 UDIS 位中写入 0 前便不会生成任何更新事件，但计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断请求）。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 使用预装载值（TIMx_PSC 寄存器的内容）重新装载预分频器的缓冲区
- 使用预装载值（TIMx_ARR）更新自动重载影子寄存器

以下各图显示了当 TIMx_ARR=0x36 时不同时钟频率下计数器行为的示例。

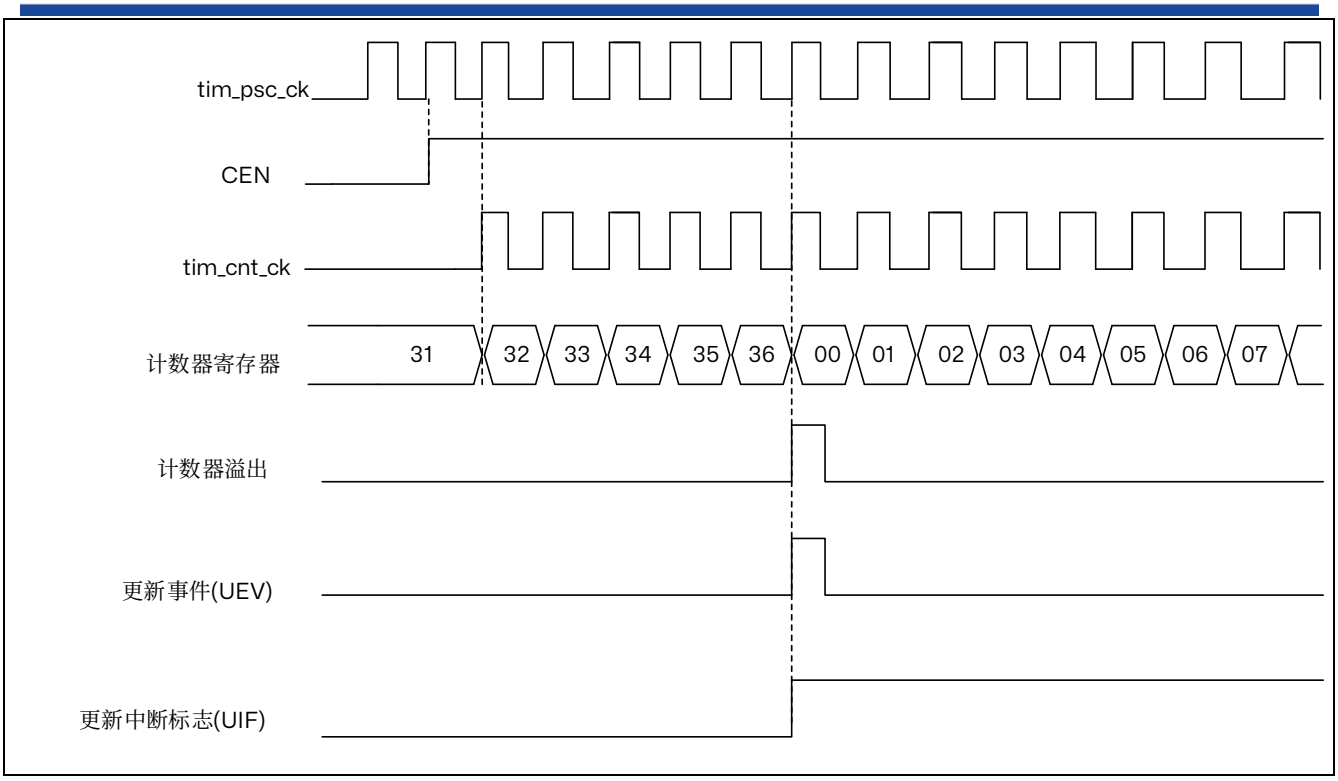



图 25.5 计数器时序图，内部时钟分频系数为 1

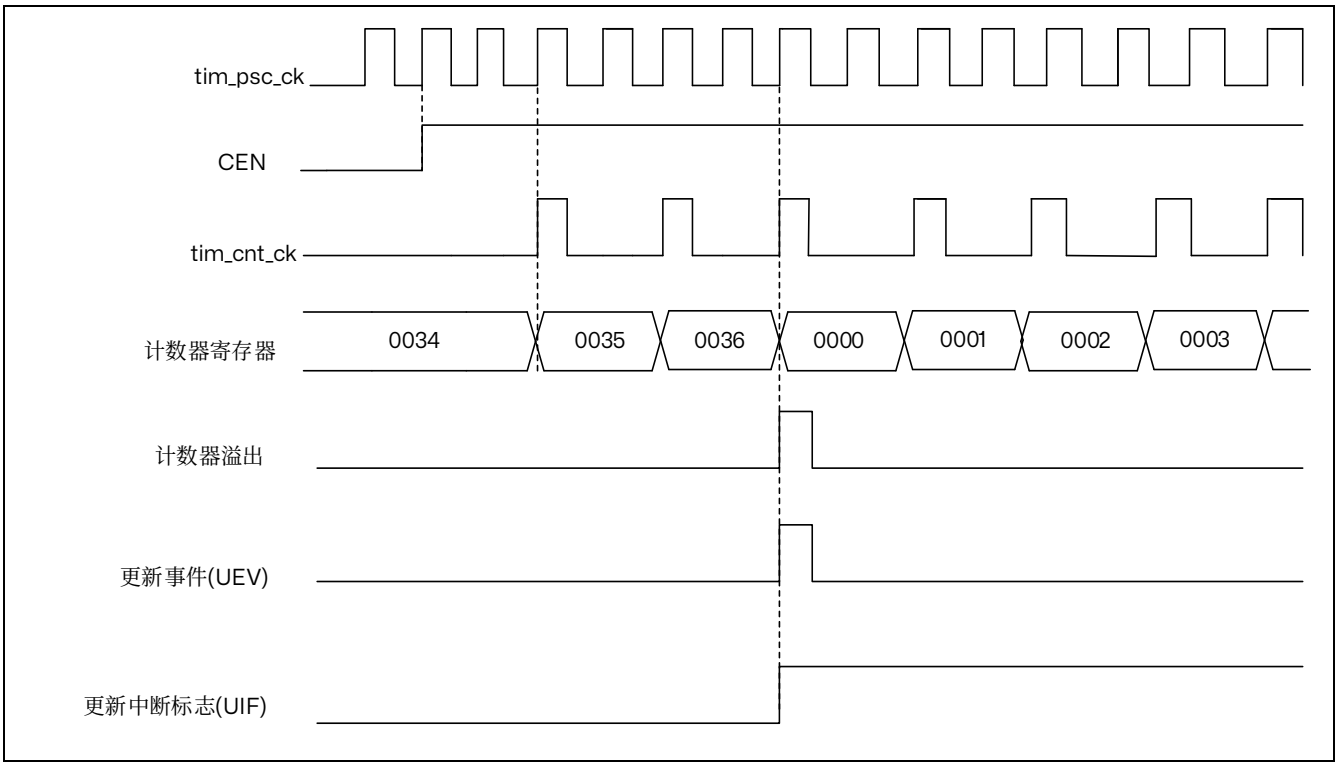



图 25.6 计数器时序图，内部时钟分频系数为 2

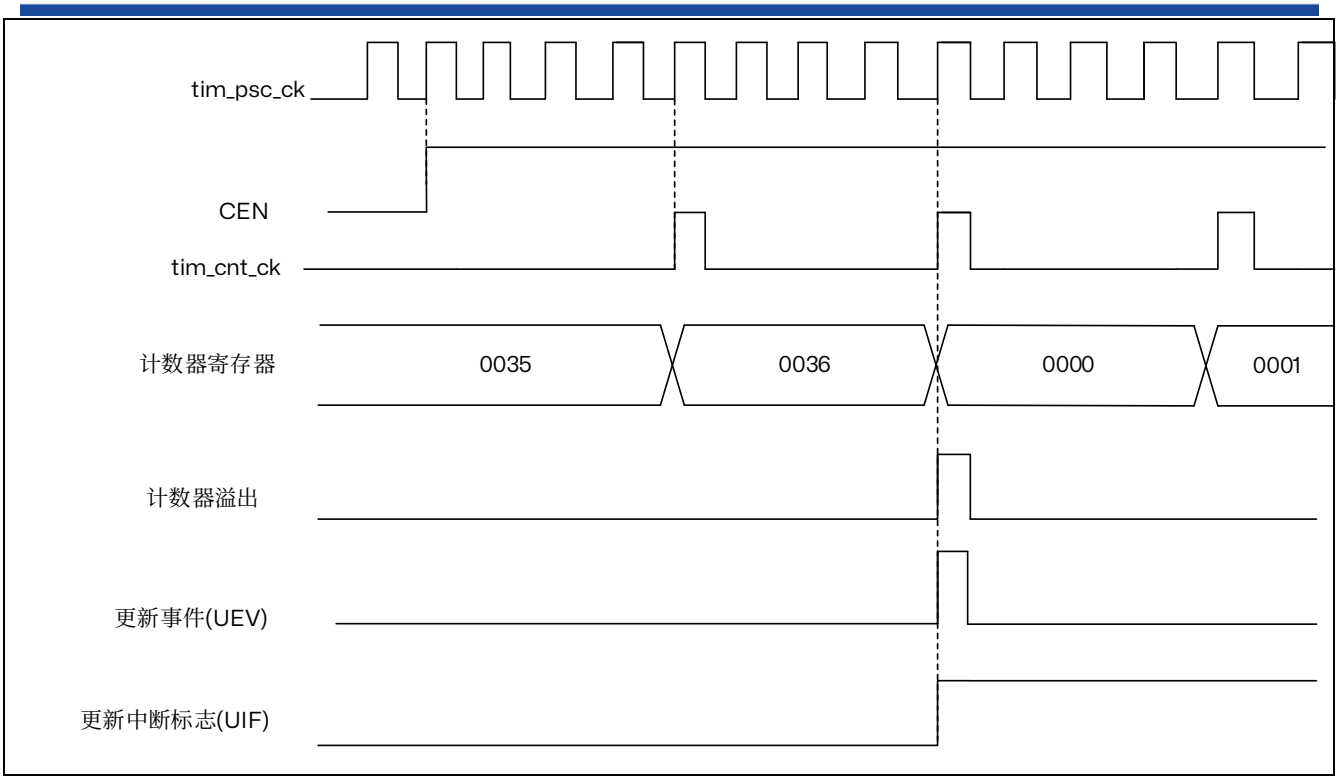



图 25.7 计数器时序图，内部时钟分频系数为 4

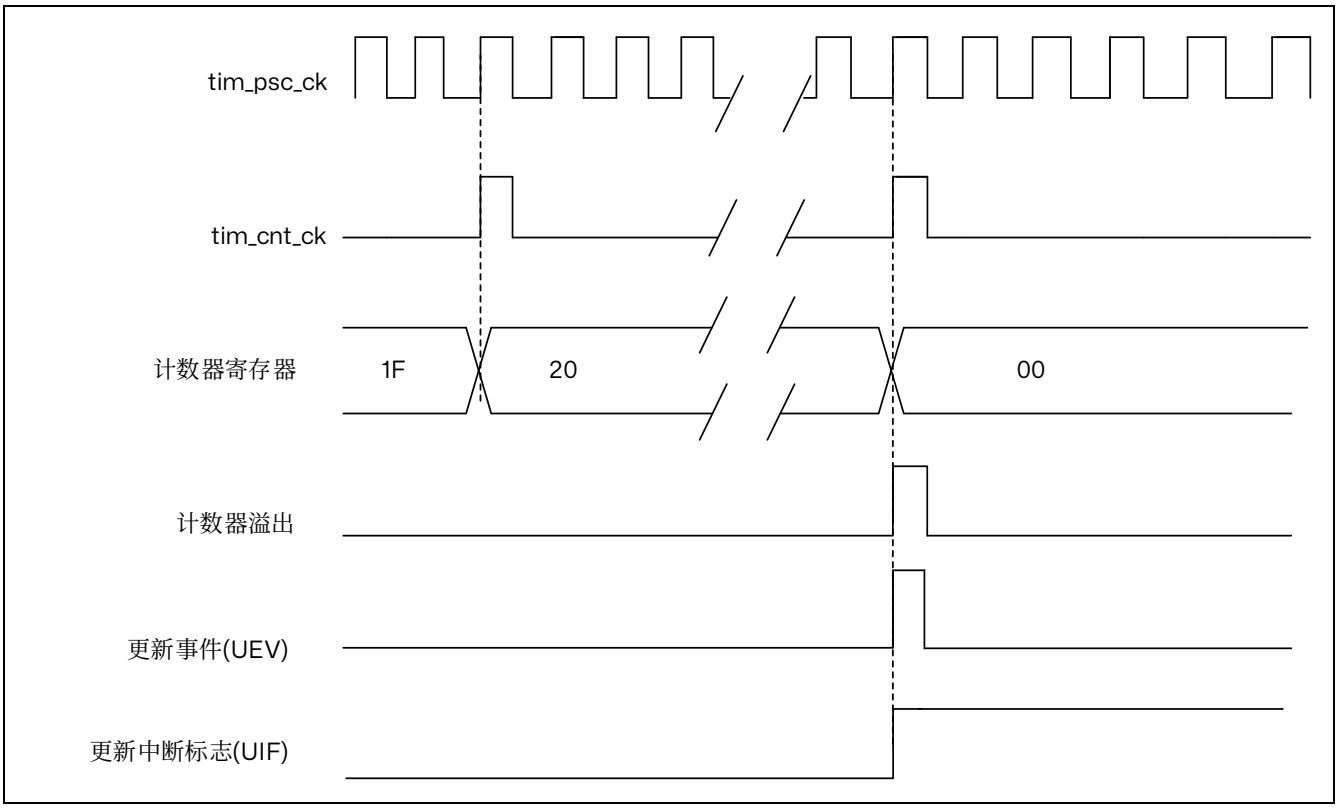



图 25.8 计数器时序图，内部时钟分频系数为 N

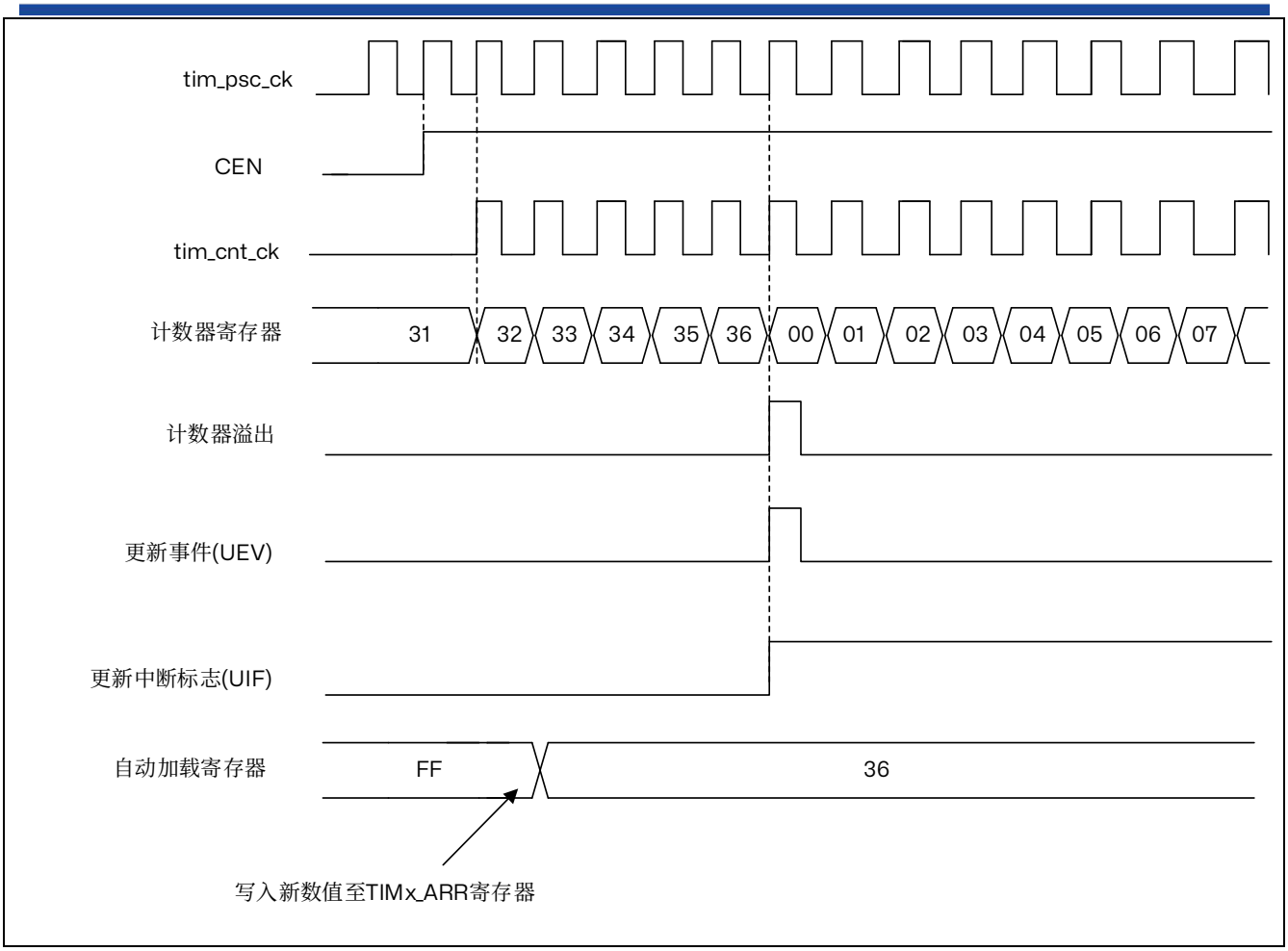



图 25.9 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

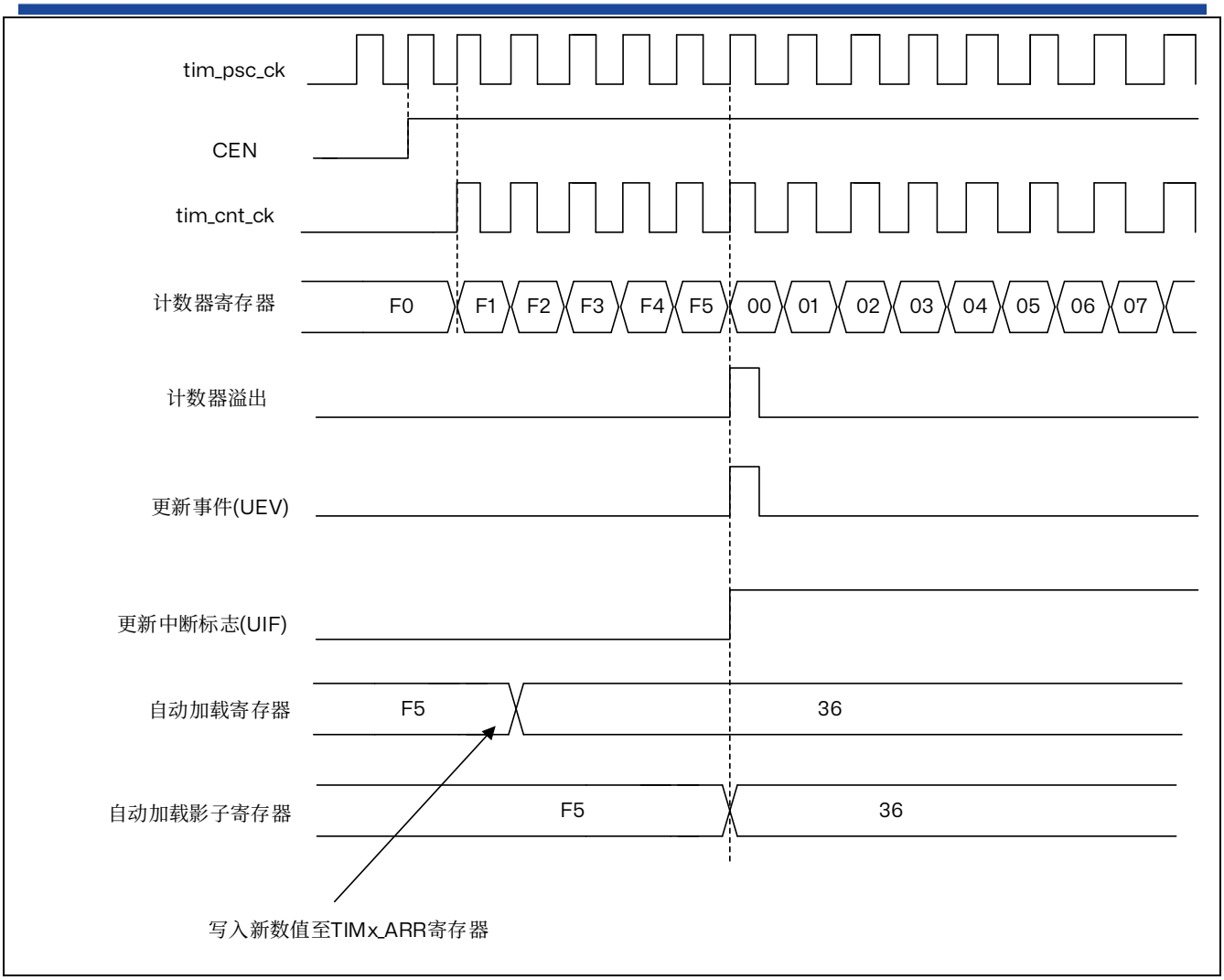



图 25.10 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

### 25.3.6 UIF 位重映射

TIMx_CR1 寄存器中的 IUFREMAP 位强制将更新中断标志 UIF 连续复制到定时器计数器寄存器的第 31 位（TIMxCNT[31]）。这允许以原子方式读取计数器的值和一个潜在的由 UIFCPY 标志指示的翻转条件。在特定情况下，它可以避免由背景任务（读取计数器）和中断（更新中断）之间的处理引起的竞争条件，从而简化计算。

UIF 和 UIFCPY 标志之间没有延迟。

### 25.3.7 ADC 触发

定时器可以使用各种内部信号（例如重置、使能或比较事件）生成 ADC 触发事件。

*注意：接收 tim_trgo 信号的从属外设（例如定时器、ADC）的时钟必须在从主定时器接收事件之前使能，并且时钟频率（预分频器）在从主定时器接收触发器时不能更改。*

### 25.3.8 TIM6/TIM7 DMA 请求

TIM6/TIM7 可以生成 DMA 请求，如下表所示。

表 25.2 DMA 请求

| DMA 请求信号 | DMA 缩写 | DMA 请求 | 使能控制位 |
|---|---|---|---|
| tim_upd_dma | TIM_UP | 更新 | UDE |

### 25.3.9 调试模式

当微控制器进入调试模式（内核停止）时，TIMx 计数器可以继续正常工作或停止。
在调试模式下，每个定时器的行为可以通过 Debug 支持（DBG）模块的专用配置位进行编程。
有关更多详细信息，请参阅调试部分。

## 25.4 TIM6/TIM7 低功耗模式

表 25.3 低功耗模式对 TIM6/TIM7 的影响

| 模式 | 描述 |
|---|---|
| 睡眠 | 无影响，外设是运行的。中断会导致设备退出睡眠模式。 |
| 停机 | 定时器操作被停止并保留其寄存器内容。无法生成中断。 |
| 待机 | 定时器被断电并且退出待机模式后必须重新初始化。 |

## 25.5 TIM6/TIM7 中断

TIM6/TIM7 可以生成一个单一中断，如下表所示。

表 25.4 中断请求

| 中断缩写 | 中断事件 | 事件标志 | 使能控制位 | 中断清除方法 | 从睡眠模式退出 | 从停机和待机退出 |
|---|---|---|---|---|---|---|
| TIM6<br>TIM7 | 更新 | UIF | UIE | UIF 写 0 | 是 | 否 |

## 25.6 TIMx 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。
外设寄存器可以通过半字（16 位）或字（32 位）进行访问。

### 25.6.1 TIMx 控制寄存器 1（TIMx_CR1）（x = 6/7）

偏移地址：0x00
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | UIFREMAP | Reserved | | | ARPE | Reserved | | | OPM | URS | UDIS | CEN |
| | | | | rw | | | | rw | | | | rw | rw | rw | rw |

Bit 15:12 保留，必须保持复位值。

Bit 11 **UIFREMAP**：UIF 状态位重映射（UIF status bit remapping）

0：无重映射。UIF 状态位未复制到 TIMx_CNT 寄存器位 31

1：重映射使能。UIF 状态位复制到 TIMx_CNT 寄存器位 31

Bit 10:8 保留，必须保持复位值。

Bit 7 **ARPE**：自动重载预装载使能（Auto-reload preload enable）

0：TIMx_ARR 寄存器不进行缓冲；

1：TIMx_ARR 寄存器进行缓冲。

Bit 6:4 保留，必须保持复位值。

Bit 3 **OPM**：单脉冲模式（One pulse mode）

0：计数器在发生更新事件时不会停止计数；

1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

Bit 2 **URS**：更新请求源（Update request source）

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会生成更新中断。此类事件包括：

- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断。

Bit 1 **UDIS**：更新禁止（Update disable）

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新（UEV）事件可通过以下事件之一生成：

- 计数器上出/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

Bit 0 **CEN**：计数器使能（Counter enable）

0：禁止计数器；

1：使能计数器。

单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

### 25.6.2 TIMx 控制寄存器 2（TIMx_CR2）（x = 6/7）

偏移地址：0x04

复位值：0x0000

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

Bit 15:7 保留，必须保持复位值。

Bit 6:4 **MMS[2:0]：**主模式选择（Master mode selection）

这些位可选择主模式下将要发送到从定时器以实现同步的信息(tim_trgo)。这些位的组合如下：

000：**复位** – TIMx_EGR 寄存器中的 UG 位用作触发输出（tim_trgo）。

001：**使能** – 计数器使能信号 CNT_EN 用作触发输出（tim_trgo）。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号可由 CEN 控制位产生。

010：**更新** – 选择更新事件作为触发输出（tim_trgo）。例如，主定时器可用作从定时器的预分频器。

*注：从定时器或接收* tim_trgo *的外设时钟必须在从主定时器接收事件之前使能，在主定时器接收触发期间不能实时更改。*

Bit 3:0 保留，必须保持复位值。

### 25.6.3 TIMx/DMA/中断使能寄存器（TIMx_DIER）（x = 6/7）

偏移地址：0x0C

复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | UDE | Reserved | | | | | | | UIE |
| | | | | | | | rw | | | | | | | | rw |

Bit 15:9 保留，必须保持复位值。

Bit 8 **UDE**：更新 DMA 请求使能（Update DMA request enable）

0：禁止更新 DMA 请求；

1：使能更新 DMA 请求。

Bit 7:1 保留，必须保持复位值。

Bit 0 **UIE**：更新中断使能（Update interrupt enable）

0：禁止更新中断；

1：使能更新中断。

### 25.6.4 TIMx 状态寄存器（TIMx_SR）（x = 6/7）

偏移地址：0x10

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | UIF |
| | | | | | | | | | | | | | | | rc_w0 |

Bit 15:1 保留，必须保持复位值。

Bit 0 **UIF**：更新中断标志（Update interrupt flag）

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– TIMx_CR1 寄存器中的 UDIS=0，并且重复计数器值上溢或下溢时（重复计数器= 0 时更新）。

– TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

### 25.6.5 TIMx 事件产生寄存器（TIMx_EGR）（x = 6/7）

偏移地址：0x14

复位：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | UG |
| | | | | | | | | | | | | | | | w |

Bit 15:1 保留，必须保持复位值。

Bit 0 **UG**：更新生成（Update generation）

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作；

1：重新初始化计数器并生成一个寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。

### 25.6.6 TIMx 计数器（TIMx_CNT）（x = 6/7）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UIF CPY | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 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 31 **UIFCPY**：复制 UIF

该位是 TIMx_SR 寄存器中 UIF 位的一个只读副本。如果 TIMx_CR1 中的 UIFREMAP 位被复位，那位 31 被保留且读为 0

Bit 30:16 保留，必须保持复位值。

Bit 15:0 **CNT[15:0]**：计数器的值（Counter value）

该寄存器（CNT[15:0]）保存的是计数器的值。

### 25.6.7 TIMx 预分频器（TIMx_PSC）（x = 6/7）

偏移地址：0x28

复位值：0x0000

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

Bit 15:0 **PSC[15:0]：**预分频器的值（Prescaler value）

计数器时钟频率（$f_{tim_cnt_ck}$）等于 $f_{tim_psc_ck}$ / (PSC[15:0] + 1)。

PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零）时要装载到活动预分频器寄存器的值。

### 25.6.8 TIMx 自动重装载寄存器（TIMx_ARR）（x = 6/7）

偏移地址：0x2C

复位：0x0000 FFFF

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

| 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 31:16 保留，必须保持复位值。

Bit 15:0 **ARR[15:0]：**自动重装载的值（Auto-reload value）

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的更多详细信息，请参阅前序章节：时基单元。

当自动重载值为空时，计数器不工作。

该寄存器存储的是 ARR[15:0]中自动重装载值。

# 26 参考电压缓冲器（VREFBUF）

## 26.1 VREFBUF 简介

RX32G410 器件内置有参考电压缓冲器，既可用作 ADC 和 DAC 的参考电压，也可通过 VREF+引脚用作外部元件的参考电压。当 VREF+引脚与 VDDA 引脚短接时，参考电压缓冲器将无法使用，并且必须保持禁用状态。

## 26.2 VREFBUF 功能描述

内部参考电压缓冲器支持三种电压[a]，这些电压通过 VREFBUF_CSR 寄存器中的 VRS 位进行配置：

- VRS = 00：大约为 2.0 V。
- VRS = 01：大约为 2.5 V。
- VRS = 10：大约为 2.8 V。

内部参考电压配置为四种不同模式，具体取决于 ENVR 和 HIZ 位的配置。下表列出了这些模式：

表 26.1 VREF 缓冲器模式

| ENVR | HIZ | VREF 缓冲器配置 |
|---|---|---|
| 0 | 0 | VREFBUF 缓冲器关闭<br>– $V_{REF+}$引脚下拉到 $V_{SSA}$ |
| 0 | 1 | 外部参考电压模式（默认值）：<br>– VREFBUF 缓冲器关闭<br>– $V_{REF+}$引脚输入模式 |
| 1 | 0 | 内部参考电压模式：<br>– VREFBUF 缓冲器开启<br>– $V_{REF+}$引脚连接到 VREFBUF 缓冲器输出 |
| 1 | 1 | 保持模式：<br>– VREF+引脚悬空。借助外部电容来保持电压<br>– 禁止 VRR 检测，VRR 位保持最后一个状态 |

通过将 VREFBUF_CSR 寄存器中的 ENVR 位置 1 并将 HIZ 位清零使能 VREFBUF 后，用户必须等待至 VRR 位置 1，指示参考电压输出已达到预期值。

a. 最小 $V_{DDA}$ 电压取决于 VRS 设置，请参阅产品数据手册。

## 26.3 VREFBUF 校准

VREFBUF 的输出电压由睿兴在出厂时进行校准。在复位时，以及每次更改 VRS 设置时，校准数据会自动加载到 TRIM 寄存器中。

用户还可以选择通过直接更改 TRIM 寄存器的位来微调输出电压。在这种情况下，直到设备复位，VRS 设置将不再对 TRIM 寄存器产生影响。

## 26.4 VREFBUF 内部参考电压档位切换步骤

内部参考电压缓冲器支持三种电压，这些电压档位在相互切换时需要遵循如下步骤：

1. VREFBUF 缓冲器关闭：ENVR=0，HIZ=0
2. 内部参考电压选择：VRS[1:0]切换至 0d00 / 0d01 / 0d11
3. VREFBUF 缓冲器打开：ENVR=1，HIZ=0

## 26.5 VREFBUF 切换外部参考电压步骤

内部参考电压缓冲器可关闭，此时可通过模拟开关将参考电压连接到 $V_{REF+}$引脚，步骤如下：

1. VREFBUF 缓冲器关闭：ENVR=0，HIZ=0
2. 模拟开关连接到 $V_{REF+}$引脚：ENVR=0，HIZ=1

## 26.6 VREFBUF 寄存器描述

### 26.6.1 VREFBUF 控制和状态寄存器（VREFBUF_CSR）

地址偏移：0x00

复位值：0x0000 0012

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | VRS[1:0] | | VRR | Res | HIZ | ENVR |
| | | | | | | | | | | rw | rw | r | | rw | rw |

Bit 31:6 保留，必须保持为复位值。

Bit 5:4 **VRS[1:0]：**参考电压调节

这些位选择由电压参考缓冲器生成的值。

00：参考电压设为 2.0 V

01：参考电压设为 2.5 V

10：参考电压设为 2.8 V

11：保留

*注：仅当*VREFBUF *被禁用*（ENVR = 0）*时，软件才能编程该位域。*

Bit 3 **VRR：**参考电压缓冲器就绪

0：参考电压缓冲器输出未就绪。

1：参考电压缓冲器输出已达到请求值。

Bit 2 保留，必须保持为复位值。

Bit 1 **HIZ：**高阻抗模式

此位控制模拟开关是否连接到 $V_{REF+}$引脚。

0：此位控制模拟开关是否连接到 $V_{REF+}$引脚。

1：$V_{REF+}$引脚为高阻态。

关于 ENVR 位配置及模式说明，请参见上述表格：VREF 缓冲器模式。

Bit 0 **ENVR：**参考电压缓冲器模式使能

此位用于使能参考电压缓冲器模式。

0：禁止内部参考电压模式（外部参考电压模式）。

1：使能内部参考电压模式（参考缓冲器使能或保持模式）。

### 26.6.2 VREFBUF 校准控制寄存器（VREFBUF_CCR）

地址偏移：0x04

复位值：0x0000 000X

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

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

Bit 31:4 保留，必须保持为复位值。

Bit 3:0 **TRIM[3:0]：**微调代码

TRIM 代码是一个 4 位无符号数据（最小值 0000，最大值 1111），它根据以下描述的机制进行设置和更新。

重置：

TRIM[3:0]在生产测试期间会自动初始化为存储在闪存中的 VRS=0 微调值。

VRS 更改：

TRIM[3:0]在生产测试期间会自动初始化为存储在闪存中的微调值（对应于 VRS 设置）。

TRIM[3:0]写入：

用户可以将 TRIM[3:0]修改为任意值。这将永久禁用 VRS 对微调值的控制（直到设备重置为止）。

*注：如果用户应用程序执行微调，微调代码必须从* 000 *开始，按升序到* 111。

# 27 实时时钟（RTC）

## 27.1 RTC 简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统（RCC_BDCR 寄存器）处于备份区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对备份寄存器和 RTC 的访问被禁止，这是为了防止对备份区域（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_CR1 的 DBP 位，使能对备份寄存器和 RTC 的访问。

## 27.2 主要特性

- 可编程的预分频系数：分频系数最高为 $2^{20}$。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟（RTC 时钟的频率必须小于 PCLK1 时钟频率的四分之一以上）。
- 可以选择以下两种 RTC 的时钟源：
  - HSE 时钟除以 128；
  - 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
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号（最长可达 1 秒）。
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态。

## 27.3 功能描述

### 27.3.1 概述

RTC 由两个主要部分组成（参见下图）。第一部分（APB1 接口）用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作。APB1 接口由 APB1 总线时钟驱动，用来与 APB1 总线接口。

另一部分（RTC 核心）由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 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。

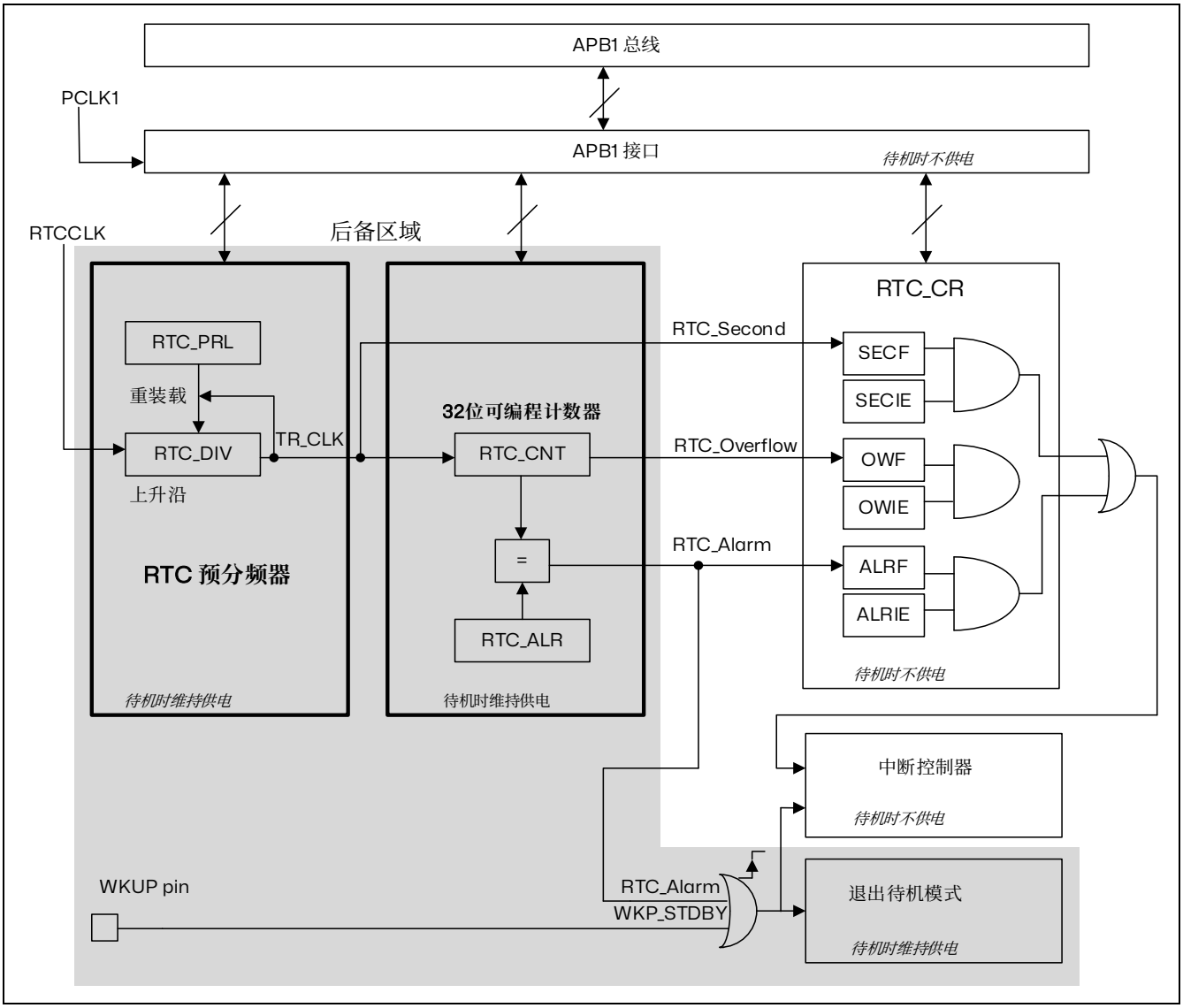



图 27.1 简化的 RTC 框图

### 27.3.2 复位过程

除了 RTC_PRL、RTC_ALR、RTC_CNT 和 RTC_DIV 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

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。

### 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 APB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次的内部寄存器更新之前，从 APB1 上读出的 RTC 寄存器数值可能被破坏了（通常读到 0）。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒。
- 系统刚从停机模式唤醒。

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_CRL 寄存器中的 RSF 位（寄存器同步标志）被硬件置“1”。

*注：RTC 的 APB1 接口不受低功耗模式的影响。*

### 27.3.4 配置 RTC 寄存器

必须设置 RTC_CRL 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：**

1. 查询 RTOFF 位，直到 RTOFF 的值变为“1”
2. 置 CNF 值为 1，进入配置模式
3. 对一个或多个 RTC 寄存器进行写操作
4. 清除 CNF 标志位，退出配置模式
5. 查询 RTOFF，直至 RTOFF 位变为“1”以确认写操作已经完成。

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 RTCCLK 周期。

### 27.3.5 RTC 标志的设置

在每一个 RTC 核心的时钟周期中，更改 RTC 计数器之前设置 RTC 秒标志（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）。

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）。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志同步：

- 使用 RTC 闹钟中断，并在中断处理程序中修改 RTC 闹钟和/或 RTC 计数器。
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟和/或 RTC 计数器。

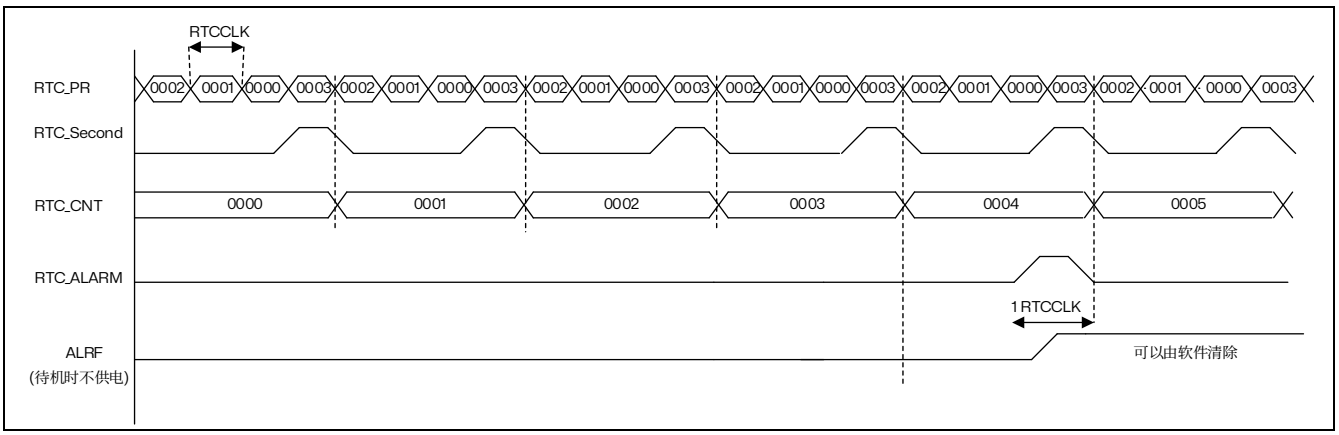



图 27.2 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004

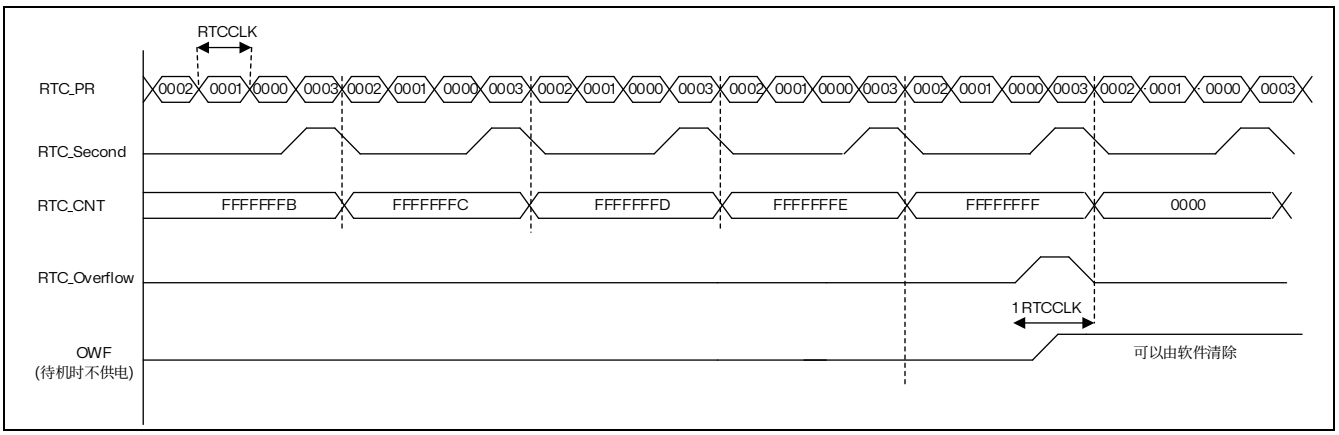



图 27.3 RTC 溢出波形图示例，PR=0003

## 27.4 RTC 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器必须按半字（16 位）或字（32 位）进行访问。

### 27.4.1 RTC 控制寄存器高位（RTC_CRH）

偏移地址：0x00
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | OWIE | ALRIE | SECIE |
| | | | | | | | | | | | | | rw | rw | rw |

Bit 15:3 保留，必须保持为复位值。

Bit 2 **OWIE**：允许溢出中断位（Overflow interrupt enable）
0：屏蔽（不允许）溢出中断
1：允许溢出中断

Bit 1 **ALRIE**：允许闹钟中断（Alarm interrupt enable）
0：屏蔽（不允许）闹钟中断
1：允许闹钟中断

Bit 0 **SECIE**：允许秒中断（Second interrupt enable）
0：屏蔽（不允许）秒中断
1：允许秒中断

这些位用来屏蔽中断请求。

*注：系统复位后所有的中断被屏蔽，因此可通过写* RTC *寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时（标志位* RTOFF=0*），不能对* RTC_CRH *寄存器进行写操作。*

RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成。

### 27.4.2 RTC 控制寄存器低位（RTC_CRL）

偏移地址：0x04
复位值：0x0020

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | RTOFF | CNF | RSF | OWF | ALRF | SECF |
| | | | | | | | | | | r | rw | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

Bit 15:6 保留，必须保持为复位值。

Bit 5 **RTOFF**：RTC 操作关闭（RTC operation OFF）
RTC 模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否完成。若此位为“0”，则表示无法对任何的 RTC 寄存器进行写操作。此位为只读位。
0：上一次对 RTC 寄存器的写操作仍在进行；
1：上一次对 RTC 寄存器的写操作已经完成。

Bit 4 **CNF**：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

寄存器写入数据。只有当此位在被置“1”并重新由软件清“0”后，才会执行写操作。
0：退出配置模式（开始更新 RTC 寄存器）；
1：进入配置模式。

Bit 3 **RSF**：寄存器同步标志（Registers synchronized flag）

每当 RTC_CNT 寄存器和 RTC_DIV 寄存器由软件更新或清“0”时，此位由硬件置“1”。在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清“0”。要进行任何的读操作之前，用户程序必须等待这位被硬件置“1”，以确保 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。

Bit 2 **OWF**：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”。对此位写“1”是无效的。
0：无溢出；
1：32 位可编程计数器溢出。

Bit 1 **ALRF**：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”。对此位写“1”是无效的。
0：无闹钟；
1：有闹钟。

Bit 0 **SECF**：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。对此位写“1”是无效的。
0：秒标志条件不成立；
1：秒标志条件成立。

RTC 的功能由这个控制寄存器控制。当前一个写操作还未完成时，不能写 RTC_CR 寄存器。

*注：1. 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受。*

*2. 在复位时禁止所有中断，无挂起的中断请求，可以对*RTC *寄存器进行写操作。*

*3. 当*APB1 *时钟不运行时，*OWF、ALRF、SECF *和*RSF *位不被更新。*

*4.* OWF、ALRF、SECF *和*RSF *位只能由硬件置位，由软件来清零。*

*5 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。*

*6 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（RTC_PRLH/RTC_PRLL）

预分频装载寄存器用来保存 RTC 预分频器的周期计数值。它们受 RTC_CR 寄存器的 RTOFF 位保护，仅当 RTOFF 值为“1”时允许进行写操作。

**RTC 预分频装载寄存器高位（RTC_PRLH）**

偏移地址：0x08
只写
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PRL[19:16] | | | |
| | | | | | | | | | | | | w | w | w | w |

Bit 15:4 保留，必须保持为复位值。

Bit 3:0 **PRL[19:16]：** RTC 预分频装载值高位（RTC prescaler reload value high）
根据以下公式，这些位用来定义计数器的时钟频率：
$f_{TR_CLK} = f_{RTCCLK}/（PRL[19:0]+1）$

*注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。*

**RTC 预分频装载寄存器低位（RTC_PRLL）**

偏移地址：0x0C
只写
复位值：0x8000

| 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 15:0 **PRL[15:0]：** RTC 预分频装载值低位
根据以下公式，这些位用来定义计数器的时钟频率：
$f_{TR_CLK} = f_{RTCCLK}/（PRL[19:0]+1）$

*注：如果输入时钟频率是 32.768kHz（$f_{RTCCLK}$），这个寄存器中写入 7FFFh 可获得周期为 1 秒钟的信号。*

### 27.4.4 RTC 预分频器余数寄存器（RTC_DIVH / RTC_DIVL）

在 TR_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 RTC_PRL 寄存器的值。用户可通过读取 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。此寄存器是只读寄存器，其值在 RTC_PRL 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

#### RTC 预分频器余数寄存器高位（RTC_DIVH）

偏移地址：0x10
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | RTC_DIV[19:16] | | | |
| | | | | | | | | | | | | r | r | r | r |

Bit 15:4 保留，必须保持为复位值。

Bit 3:0 **RTC_DIV[19:16]：**RTC 时钟分频器余数高位（RTC clock divider high）

#### RTC 预分频器余数寄存器低位（RTC_DIVL）

偏移地址：0x14
复位值：0x8000

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

Bit 15:0 **RTC_DIV[15:0]：**RTC 时钟器余数低位（RTC clock divider low）

### 27.4.5 RTC 计数器寄存器（RTC_CNTH/RTC_CNTL）

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 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。他们受 RTC_CR 的位 RTOFF 写保护，仅当 RTOFF 值为“1”时，允许写操作。在高或低寄存器（RTC_CNTH 或 RTC_CNTL）上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。当进行读操作时，直接返回计数器内的计数值（系统时间）。

#### RTC 计数器寄存器高位（RTC_CNTH）

偏移地址：0x18
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_CNT[31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15:0 **RTC_CNT[31:16]：**RTC 计数器高位（RTC counter high）

可通过读 RTC_CNTH 寄存器来获得 RTC 计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式。

**RTC 计数器寄存器低位（RTC_CNTL）**

偏移地址：0x1C
复位值：0x0000

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

Bit 15:0 **RTC_CNT[15:0]**：RTC 计数器低位。

可通过读 RTC_CNTL 寄存器来获得 RTC 计数器当前值的低位部分。要对此寄存器进行写操作，必须先进入配置模式。

## 27.4.6 RTC 闹钟寄存器（RTC_ALRH/RTC_ALRL）

当可编程计数器的值与 RTC_ALR 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器受 RTC_CR 寄存器里的 RTOFF 位写保护，仅当 RTOFF 值为“1”时，允许写操作。

**RTC 闹钟寄存器高位（RTC_ALRH）**

偏移地址：0x20
只写
复位值：0xFFFF

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_ALR[31:16] | | | | | | | | | | | | | | | |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 15:0 **RTC_ALR[31:16]**：RTC 闹钟值高位（RTC alarm high）

此寄存器用来保存由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先进入配置模式。

**RTC 闹钟寄存器低位（RTC_ALRL）**

偏移地址：0x24
只写
复位值：0xFFFF

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

Bit 15:0 **RTC_ALR[15:0]**：RTC 闹钟值低位（RTC alarm low）

此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式。

# 28 独立看门狗（IWDG）

## 28.1 简介

设备内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备（独立看门狗和窗口看门狗）可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断（仅适用于窗口型看门狗）或产生系统复位。

独立看门狗（IWDG）由专用的低速时钟（LSI）驱动，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

## 28.2 IWDG 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（可在停止和待机模式下工作）
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

## 28.3 IWDG 功能描述

下图为独立看门狗模块的功能框图。

在键寄存器（IWDG_KR）中写入 0xCCCC，开始使能独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号（IWDG_RESET）。

无论何时，只要在键寄存器 IWDG_KR 中写入 0xAAAA，IWDG_RLR 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

### 28.3.1 硬件看门狗

如果用户在选项字节中使能了“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。

### 28.3.2 寄存器访问保护

IWDG_PR 和 IWDG_RLR 寄存器具有写保护功能。要修改这两个寄存器的值，必须先向 IWDG_KR 寄存器中写入 0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器将重新被保护。重装载操作（即写入 0xAAAA）也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

### 28.3.3 调试模式

当微控制器进入调试模式时（内核停止），根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。

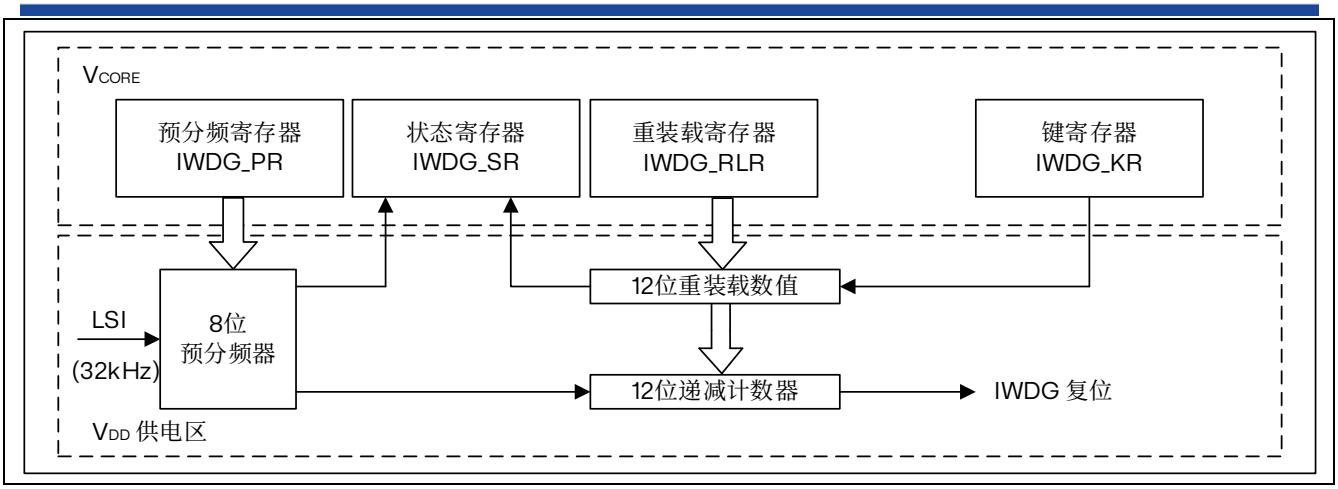



图 28.1 独立看门狗框图

*注：看门狗功能处于VDD 供电区，即在停机和待机模式时仍能正常工作。*

表 28.1 看门狗超时时间（32kHz 的输入时钟（LSI））(1)

| 预分频系数 | PR[2:0]位 | 最短时间（ms）<br>RL[11:0] = 0x000 | 最短时间（ms）<br>RL[11:0] = 0xFFF |
|---|---|---|---|
| /4 | 0 | 0.125 | 512 |
| /8 | 1 | 0.25 | 1024 |
| /16 | 2 | 0.5 | 2048 |
| /32 | 3 | 1 | 4096 |
| /64 | 4 | 2 | 8192 |
| /128 | 5 | 4 | 16384 |
| /256 | （6 或 7） | 8 | 32768 |

1. 这些时序参数是基于 32kHz 时钟给出的，但微控制器内部 RC 频率可能会有所变化。有关最大值和最小值，请参考器件数据手册中的 LSI 振荡器特性表。

## 28.4 IWDG 寄存器描述

有关寄存器说明中使用的缩写，请参见第 2 章节：文档约定中的第 1 节 。
外设寄存器必须按半字（16 位）或字（32 位）进行访问。

### 28.4.1 键寄存器（IWDG_KR）

地址偏移：0x00
复位值：0x0000 0000（在待机模式复位）

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

| 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 31:16 保留，必须保持为复位值。

Bit 15:0 **KEY[15:0]：**键值（只写寄存器，读出值为 0x0000）（Key value）
软件必须以一定的间隔写入 0xAAAA，否则，当计数器为 0 时，看门狗会产生复位。
写入 0x5555 表示允许访问 IWDG_PR 和 IWDG_RLR 寄存器。
写入 0xCCCC，启动看门狗工作（若选择了硬件看门狗则不受此命令字限制）。

### 28.4.2 预分频寄存器（IWDG_PR）

地址偏移：0x04
复位值：0x0000 0000

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

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

Bit 31:3 保留，必须保持为复位值。

Bit 2:0 **PR[2:0]：**预分频系数（Prescaler divider）
这些位具有写保护设置。通过设置这些位来选择计数器时钟的预分频系数。要改变预分频系数，IWDG_SR 寄存器的 PVU 位必须为 0。

000：预分频系数=4　　100：预分频系数=64
001：预分频系数=8　　101：预分频系数=128
010：预分频系数=16　　110：预分频系数=256
011：预分频系数=32　　111：预分频系数=256

*注：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 PVU 位为 0 时，读出的值才有效。*

### 28.4.3 重装载寄存器（IWDG_RLR）

地址偏移：0x08

复位值：0x0000 0FFF（待机模式时复位）

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

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

Bit 31:12 保留，必须保持为复位值。

Bit 11:0 **RL[11:0]：**看门狗计数器重装载值（Watchdog counter reload value）

这些位具有写保护功能。用于定义看门狗计数器的重装载值，每当向 IWDG_KR 寄存器写入 0xAAAA 时，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算。

只有当 IWDG_SR 寄存器中的 RVU 位为 0 时，才能对此寄存器进行修改。

*注：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 RVU 位为 0 时，读出的值才有效。*

### 28.4.4 状态寄存器（IWDG_SR）

地址偏移：0x0C

复位值：0x0000 0000（待机模式时不复位）

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | RVU | PVU |
| | | | | | | | | | | | | | | r | r |

Bit 31:2 保留，必须保持为复位值。

Bit 1 **RVU：**看门狗计数器重装载值更新（Watchdog counter reload value update）

此位由硬件置“1”用来指示重装载值的更新正在进行中。当在 $V_{DD}$ 域中的重装载更新结束后，此位由硬件清“0”（最多需 5 个 32kHz 的 RC 周期）。重装载值只有在 RVU 位被清“0”后才可更新。

Bit 0 **PVU：**看门狗预分频值更新（Watchdog prescaler value update）

此位由硬件置“1”用来指示预分频值的更新正在进行中。当在 $V_{DD}$ 域中的预分频值更新结束后，此位由硬件清“0”（最多需 5 个 32kHz 的 RC 周期）。预分频值只有在 PVU 位被清“0”后才可更新。

*注：如果在应用程序中使用了多个重装载值或预分频值，则必须在 RVU 位被清除后才能重新改变预装载值，在 PVU 位被清除后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不必等待 RVU 或 PVU 复位，可继续执行下面的代码。（即是在低功耗模式下，此写操作仍会被继续执行完成。）*

# 29 窗口看门狗（WWDG）

## 29.1 WWDG 简介

窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 T6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减计数器数值（在控制寄存器中）被刷新，那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

## 29.2 WWDG 主要特性

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于 0x40，（若看门狗被启动）则产生复位。
  - 当递减计数器在窗口外被重新装载，（若看门狗被启动）则产生复位。
- 如果启动了看门狗并且允许中断，当递减计数器等于 0x40 时产生早期唤醒中断（EWI），它可以被用于重装载计数器以避免 WWDG 复位。

## 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（T6 位清零）时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

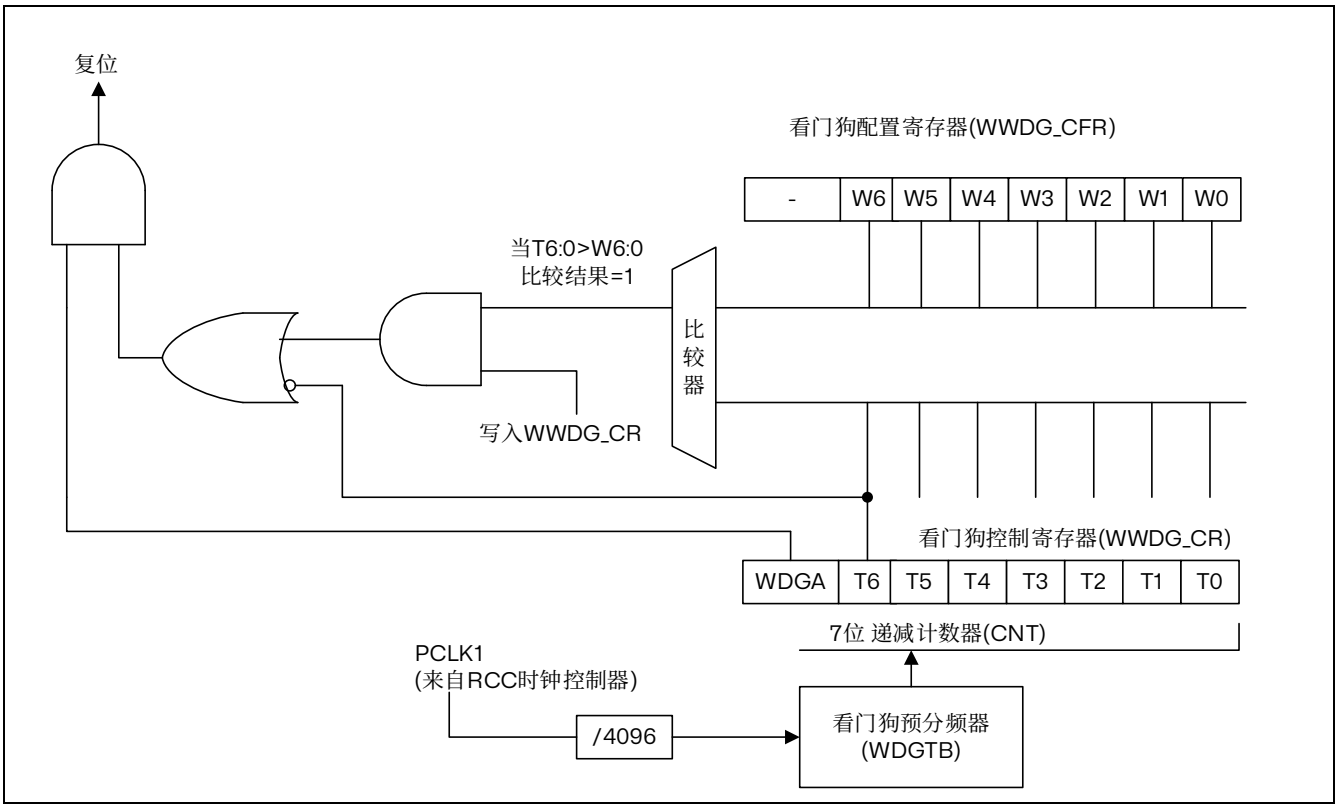



图 29.1 窗口看门狗框图

应用程序在正常运行过程中必须定期地写入 WWDG_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 WWDG_CR 寄存器中的数值必须在 0xFF 和 0xC0 之间：

**启动看门狗**

在系统复位后，看门狗总是处于关闭状态，设置 WWDG_CR 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

**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。当看门狗被使能时，T6 位必须被设置，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（见下图）。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，下图描述了窗口寄存器的工作过程。

*注：可以用*T6 *位产生一个软件复位（设置*WDGA *位为“1”，*T6 *位为“0”)。*

**高级看门狗中断功能**

如果在实际复位产生之前必须执行特定的安全操作或数据记录，则可使用提前唤醒中断（EWI）。通过设置 WWDG_CFR 寄存器中的 EWI 位来使能 EWI 中断。当递减计数器达到 0x40 时，会产生 EWI 中断，且可使用相应的中断服务程序（ISR）在复位设备之前触发特定操作（如通信或数据记录）。

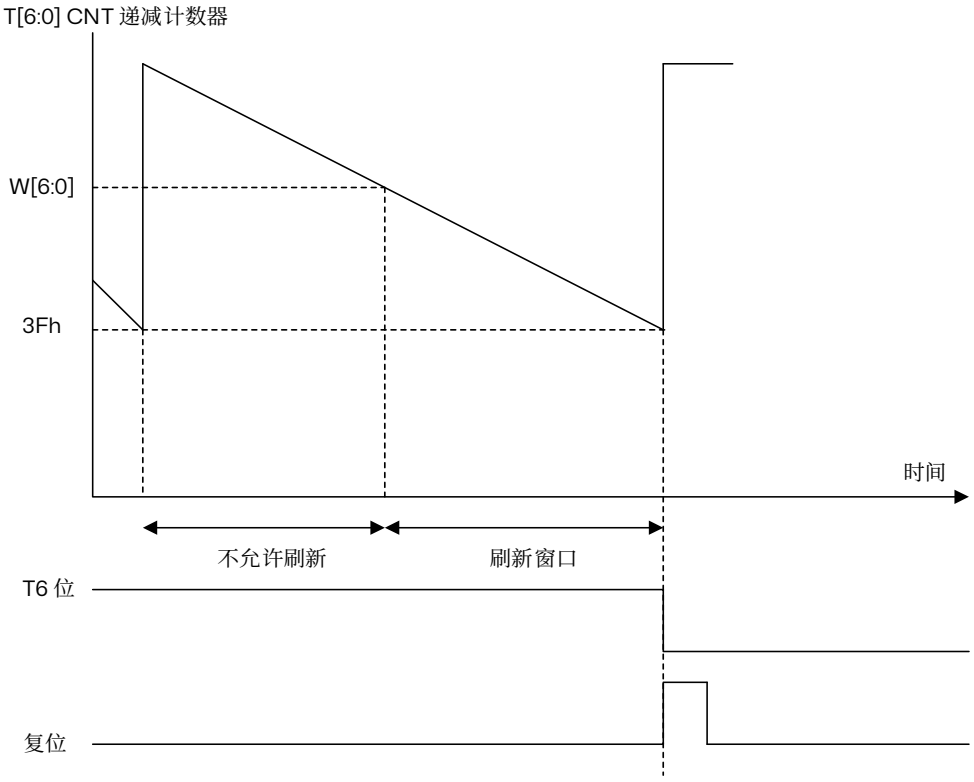
在某些应用中，EWI 中断可用于管理软件系统检查和/或系统恢复/优雅降级，而无需产生 WWDG 复位。在这种情况下，相应的中断服务程序（ISR）应重新加载 WWDG 计数器，以避免 WWDG 复位，然后触发所需操作。通过向 WWDG_SR 寄存器中的 EWIF 位写入“0”来清除 EWI 中断。

*注：如果无法处理*EWI *中断（例如，由于系统锁定在更高优先级任务中)，最终将产生*WWDG *复位。*

## 29.4 如何编写看门狗超时程序

可以使用下图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 T6 位为‘1’以避免立即产生一个复位。**




计算超时的公式如下：

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{WDGTB} \times (T[5:0] + 1); \quad (ms)$$

其中：

$T_{WWDG}$：WWDG超时时间

$T_{PCLK1}$：APB1以ms为单位的时钟间隔

在PCLK1 = 96MHz时的最小-最大超时值

| WDGTB | 最小超时值 | 最大超时值 |
|---|---|---|
| 0 | 21µs | 2.73ms |
| 1 | 43µs | 5.46ms |
| 2 | 85µs | 10.92ms |
| 3 | 171µs | 21.85ms |

图 29.2 窗口看门狗时序图

## 29.5 调试模式

当微控制器进入调试模式时（内核停止），根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。

## 29.6 WWDG 寄存器描述

### 29.6.1 控制寄存器（WWDG_CR）

偏移地址：0x00

复位值：0x0000 007F

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDGA | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | | | | | | | rs | rw | rw | rw | rw | rw | rw | rw |

Bit 31:8 保留，必须保持为复位值。

Bit 7 WDGA：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：使能看门狗

Bit 6:0 **T[6:0]：** 7 位计数器（MSB 至 LSB）（7-bit counter）

这些位用来存储看门狗的计数器值。每（$4096x2^{WDGTB}$）个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时（T6 变成 0），产生看门狗复位。

### 29.6.2 配置寄存器（WWDG_CFR）

偏移地址：0x04

复位值：0x0000 007F

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | EWI | WDGTB[1:0] | | W6 | W5 | W4 | W3 | W2 | W1 | W0 |
| | | | | | | rs | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:10 保留，必须保持为复位值。

Bit 9 **EWI：** 提前唤醒中断（Early wakeup interrupt）

此位若置‘1’，则当计数器值达到 40h，即产生中断。

此中断只能由硬件在复位后清除。

Bit 8:7 **WDGTB[1:0]：** 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

Bit 6:0 **W[6:0]：** 7 位窗口值（7-bit window value）

这些位包含了用来与递减计数器进行比较用的窗口值。

### 29.6.3 状态寄存器（WWDG_SR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | EWIF |
| | | | | | | | | | | | | | | | rc_w0 |

Bit 31:1 保留，必须保持为复位值。

Bit 0 **EWIF：**提前唤醒中断标志（Early wakeup interrupt flag）

当计数器值达到 40h 时，此位由硬件置‘1’。它必须通过软件写‘0’来清除。对此位写‘1’无效。若中断未被使能，此位也会被置‘1’。

# 30 USB 全速设备接口（USB）

## 30.1 USB 简介

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。

USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

## 30.2 USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点
- CRC(循环冗余校验)生成/校验，反向不归零（NRZI）编码/解码和位填充
- 支持同步传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/恢复操作
- 帧锁定时钟脉冲生成

*注：* USB *和* CAN *共用一个专用的的* SRAM *存储器用于数据的发送和接收，因此不能同时使用* USB *和* CAN *（共享的* SRAM *被* USB *和* CAN *模块互斥地访问）。*USB *和* CAN *可以同时用于一个应用中但不能在同一个时间使用。*

## 30.3 USB 框图

下图是 USB 外设的框图。

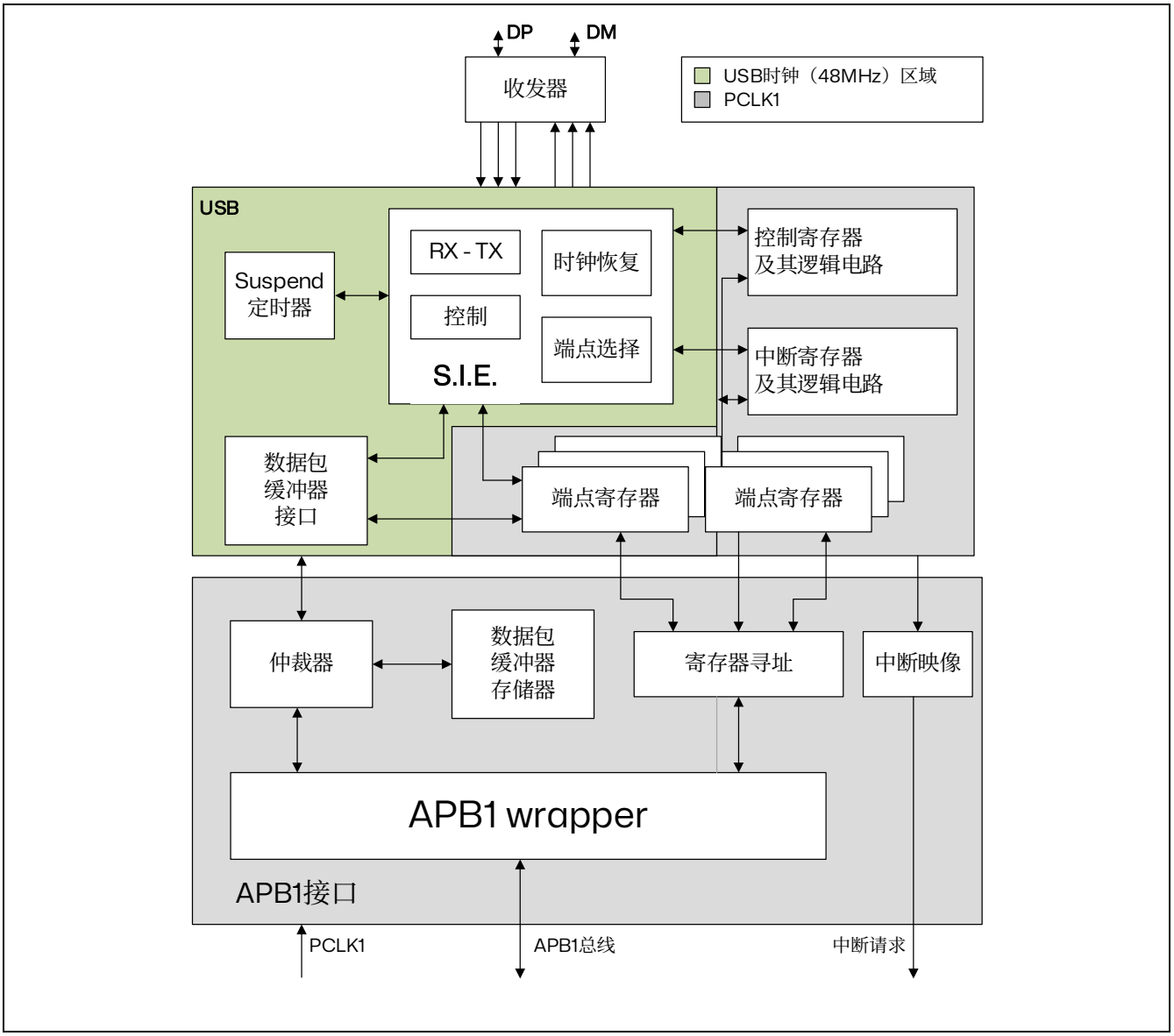



图 30.1 USB 框图

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512 字节缓冲区，最多可用于 16 个单向或 8 个双向端点。USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输的格式由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个缓冲区描述块，描述该端点使用的缓冲区地址、大小和需要传输的字节数。当 USB 模块识别出一个有效的功能/端点的令牌分组时，（如果需要传输数据并且端点已配置）随之发生相关的数据传输。USB 模块通过一个内部的 16 位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务
- 产生如位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类型的传输。

USB 模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了 USB 外设总是可以使用另一个缓冲区。

在任何不需要使用 USB 模块的时候，通过写控制寄存器总可以使 USB 模块置于低功耗模式（SUSPEND 模式）。在这种模式下，不产生任何静态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。也可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常的时钟系统，并支持直接启动或停止时钟系统。

### 30.3.1 USB 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口控制器（SIE）：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首（Start of Frame），USB 复位，数据错误等等，这些信号用来产生中断。
- 定时器：本模块的功能是产生一个与帧开始报文同步的时钟脉冲，并在 3ms 内没有数据传输的状态，检测出（主机的）全局挂起条件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。
- 端点相关寄存器：每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点[a]或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。
- 控制寄存器：这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复，低功耗等 USB 事件。
- 中断寄存器：这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

a. 端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512 字节，由 256 个 16 位的字构成。
- 仲裁器：该部件负责处理来自 APB1 总线和 USB 接口的存储器请求。它通过向 APB1 提供较高的访问优先权来解决总线的冲突，并且总是保留一半的存储器带宽供 USB 完成传输。它采用时分复用的策略实现了虚拟的双端口 SRAM，即在 USB 传输的同时，允许应用程序访问存储器。此策略也允许任意长度的多字节 APB1 传输。
- 寄存器映射单元：此部件将 USB 模块的各种字节宽度和位宽度的寄存器映射成能被 APB1 寻址的 16 位宽度的内存集合。
- APB1 封装：此部件为缓冲区和寄存器提供了到 APB1 的接口，并将整个 USB 模块映射到 APB1 地址空间。
- 中断映射单元：将可能产生中断的 USB 事件映射到三个不同的中断请求线上：
  - USB 低优先级中断（通道 20）：可由所有 USB 事件触发（正确传输，USB 复位等）。固件在处理中断前应当首先确定中断源。
  - USB 高优先级中断（通道 19）：仅能由同步和双缓冲批量传输的正确传输事件触发，目的是保证最大的传输速率。
  - USB 唤醒中断（通道 42）：由 USB 挂起模式的唤醒事件触发。

## 30.4 编程中需要考虑的问题

在下面的章节中，将介绍 USB 模块和应用程序之间的交互过程，有利于简化应用程序的开发。

### 30.4.1 通用 USB 设备编程

这一部分描述了实现 USB 设备功能的应用程序需要完成的任务。除了介绍一般的 USB 事件中应该采取的操作外，还着重介绍了双缓冲端点和同步传输的操作。这些相关的操作都是由 USB 模块初始化，并由以下几节所描述的 USB 事件所驱动。

### 30.4.2 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。

其次，必须配置 CNTR 寄存器的 PDWN 位用以开启 USB 收发器相关的模拟部分，这点需要特别的处理。此位能打开为端点收发器供电的内部参照电压。由于打开内部电压需要一段启动时间（数据手册中的 $t_{STARTUP}$），在此期间内 USB 收发器处于不确定状态，所以在设置 CNTR 寄存器的 PDWN 后必需等待一段时间之后，才能清除 USB 模块的复位信号（清除 CNTR 寄存器上的 FRES 位），和 ISTR 寄存器的内容，以便在使能其他任何单元的操作之前清除未处理的假中断标志。

当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化（比如中断使能的选择，分组缓冲区地址的选择等）。接下来按照 USB 复位处理（参见下段）。

#### USB 复位（RESET 中断）

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态：所有端点的通信都被禁（USB 模块不会响应任何分组）。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点（端点 0）也需要被使能。这可以通过配置 USB_DADDR 寄存器的 EF 位，EP0R 寄存器和相关的分组缓冲区来实现。在 USB 设备的枚举阶段，主机将分配给设备一个唯一的地址，这个地址必须写 USB_DADDR 寄存器的 ADD[6:0]位中，同时配置其他所需的端点。

当复位中断产生时，应用程序必需在中断产生后的 10ms 之内使能端点 0 的传输。

#### 分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并等待确认信息后返回。为防止产生微控制器与 USB 模块对缓冲区的访问冲突，缓冲区接口模块使用仲裁机制，使 APB1 总线的一半周期用于微控制器的访问，另一半保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。USB 模块使用固定的时钟，按照 USB 标准，此时钟频率被固定为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

*注：为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 8MHz，以避免数据缓冲区溢出或不满*

每个端点对应于两个分组缓冲区（一般一个用于发送，另一个用于接收）。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由 USB_BTABLE 寄存器确定。缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的字组成，因此缓冲区描述表的起始地址按 8 字节对齐（寄存器的最低 3 位总是‘000’）。请参考章节：缓冲区描述表获得详细介绍。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区（即发送方向上的分组缓冲区）。其他未用到的端点或某个未使用的方向上的缓冲区描述表项可以用于其他用途。同步和双缓冲批量端点有特殊的分组缓冲区处理方法（请分别参考章节：同步传输和章节：双缓冲端点）。下图描述了缓冲区描述表项和分组缓冲区区域的关系。

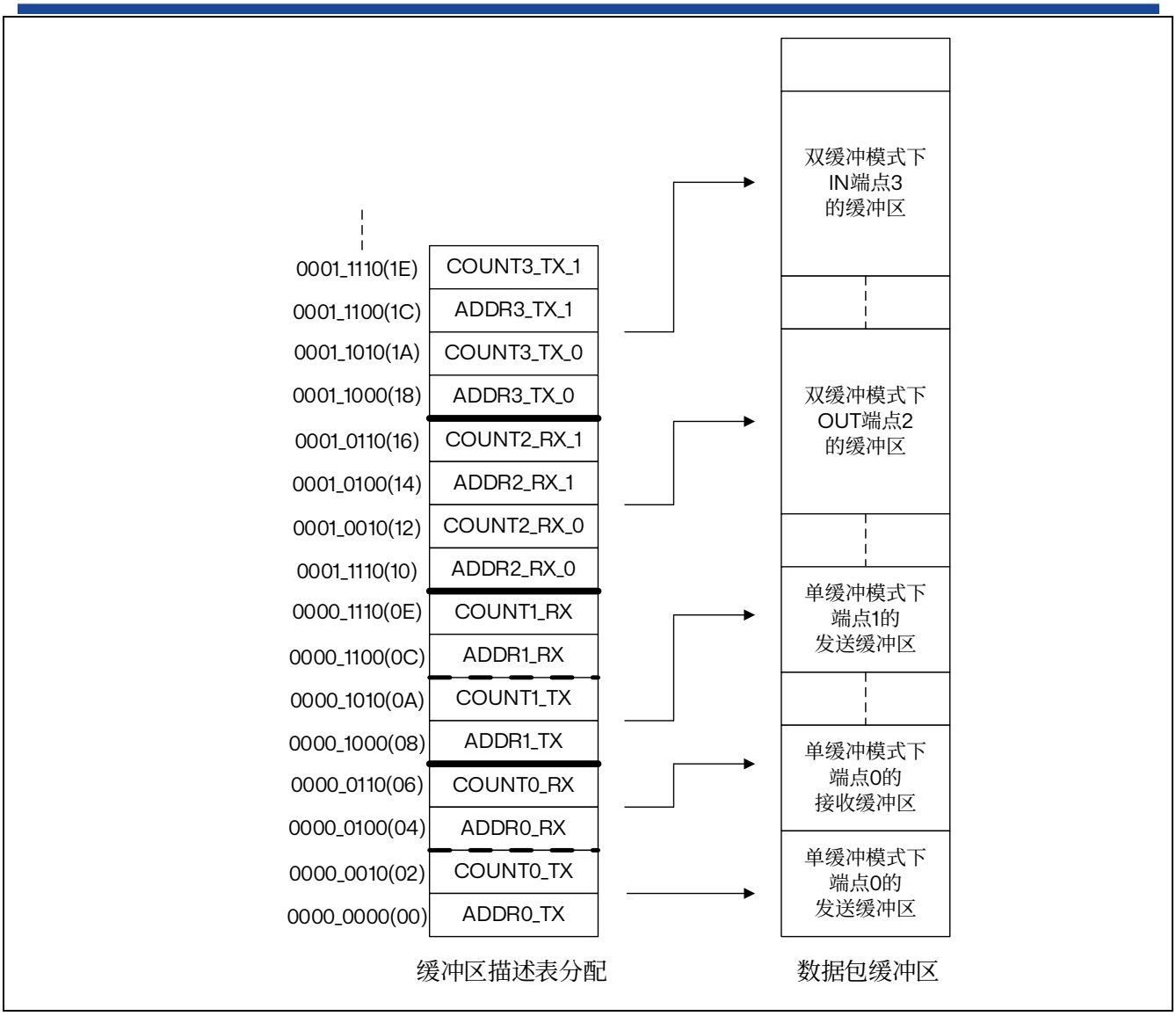



图 30.2 分组缓冲区对应的缓冲区描述表项定位

不管是接收还是发送，分组缓冲区都是从底部开始使用的。USB 模块不会改变超出当前分配到的缓冲区区域以外的其他缓冲区的内容。如果缓冲区收到一个比自己大的数据分组，它只会接收最大为自身大小的数据，其他的丢掉，即发生了所谓的缓冲区溢出异常。

### 端点初始化

初始化端点的第一步是把适当的值写到 ADDRn_TX 或 ADDRn_RX 寄存器中，以便 USB 模块能找到要传输的数据或准备好接收数据的缓冲区。USB_EPnR 寄存器的 EP_TYPE 位确定端点的基本类型，EP_KIND 位确定端点的特殊特性。作为发送方，需要设置 USB_EPnR 寄存器的 STAT_TX 位来使能端点，并配置 COUNTn_TX 位决定发送长度。作为接收方，需要设置 STAT_RX 位来使能端点，并且设置 BL_SIZE 和 NUM_BLOCK 位，确定接收缓冲区的大小，以检测缓冲区溢出的异常。对于非同步非双缓冲批量传输的单向端点，只需要设置一个传输方向上的寄存器。一旦端点被使能，应用程序就不能再修改 USB_EPnR 寄存器的值和 ADDRn_TX / ADDRn_RX，COUNTn_TX / COUNTn_RX 所在的位置，因为这些值会被硬件实时修改。当数据传输完成时，CTR 中断会产生，此时上述寄存器可以被访问，并重新使能新的传输。

### IN 分组（用于数据发送）

当接收到一 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项，访问相应的 ADDRn_TX 和 COUNTn_TX 寄存器，并将这些寄存器中的数值存储到内部的 16 位寄存器 ADDR 和 COUNT(应用程序无法访问)中。此时，USB 模块开始根据 DTOG_TX 位发送 DATA0 或 DATA1 分组，并访问缓冲区（请参考‘分组缓冲区的结构和用途’段落）。在 IN 分组传输完毕之后，从缓冲区读到的第一个字节将被装载到输出移位寄存器中，并开始发送。最后一个数据字节发送完成之后，计算好的 CRC 将被发送。如果收到的分组所对应的端点是无效的，将根据 USB_EPnR 寄存器上的 STAT_TX 位发送 NAK 或 STALL 握手分组而不发送数据。

ADDR 内部寄存器被用作当前缓冲区的指针，COUNT 寄存器用于记录剩下未传输的字节数。USB 总线使用低字节在先的方式传输从缓冲区中读出的数据。数据从 ADDRn_TX 指向的数据分组缓冲区开始读取，长度为 COUNTn_TX/2 个字。如果发送的数据分组为奇数个字节，则只使用最后一个字的低 8 位。

在接收到主机响应的 ACK 后，USB_EPnR 寄存器的值有以下更新：DTOG_TX 位被翻转，STAT_TX 位为‘10’，使端点无效，CTR_TX 位被置位。应用程序需要通过 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别产生中断的 USB 端点。CTR_TX 事件的中断服务程序需要首先清除中断标志位，然后准备好需要发送的数据缓冲区，更新 COUNTn_TX 为下次需要传输的字节数，最后再设置 STAT_TX 位为‘11’（端点有效），再次使能数据传输。当 STAT_TX 位为‘10’时（端点为 NAK 状态），任何发送到该端点的 IN 请求都会被 NAK，USB 主机会重发 IN 请求直到该端点确认请求有效。上述操作过程是必需遵守的，以避免丢失紧随上一次 CTR 中断请求的下一个 IN 传输请求。

### OUT 分组和 SETUP 分组（用于数据接收）

USB 模块对这两种分组的处理方式基本相同；对 SETUP 分组的特殊处理将在下面关于控制传输部分详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，USB 模块将访问缓冲区描述表，找到与该端点相关的 ADDRn_RX 和 COUNTn_RX 寄存器，并将 ADDRn_RX 寄存器的值保存在内部 ADDR 寄存器中。同时，COUNT 会被被复位，从 COUNTn_RX 中读出的 BL_SIZE 和 NUM_BLOCK 的值用于初始化内部 16 位寄存器 BUF_COUNT，该寄存器用于检测缓冲区溢出（所有的内部寄存器都不能被应用程序访问）。USB 模块将随后收到的数据按字方式组织（先收到的为低字节），并存储到 ADDR 指向的分组缓冲区中。同时，BUF_COUNT 值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时，USB 模块校验收到 CRC 的正确性。如果传输中没有任何错误发生，则发送 ACK 握手分组到主机。即使发生 CRC 错误或者其他类型的错误（位填充，帧错误等），数据还是会被保存到分组缓冲区中，至少会保存到发生错误的数据点，只是不会发送 ACK 分组，并且 USB_ISTR 寄存器的 ERR 位将会置位。在这种情况下，应用程序通常不需要干涉处理，USB 模块将从传输错误中自动恢复，并为下一次传输做好准备。如果收到的分组所对应的端点没有准备好，USB 模块将根据 USB_EPnR 寄存器的 STAT_RX 位发送 NAK 或 STALL 分组，数据将不会被写入接收缓冲区。

ADDRn_RX 的值决定接收缓冲区的起始地址，长度由包含 CRC 的数据分组的长度（即有效数据长度+2）决定，但不能超过 BL_SIZE 和 NUM_BLOCK 所定义的缓冲区的长度。如果接收到的数据分组的长度超出了缓冲区的范围，超过范围的数据不会被写入缓冲区，USB 模块将报告缓冲区发生溢出，并向主机发送 STALL 握手分组，通知此次传输失败，也不产生中断。

如果传输正确完成，USB 模块将发送 ACK 握手分组，内部的 COUNT 寄存器的值会被复制到相应的 COUNTn_RX 寄存器中，BL_SIZE 和 NUM_BLOCK 的值保持不变，也不需要重写。USB_EPnR 寄存器按下列方式更新：DTOG_RX 位翻转，STAT_RX=10(NAK)使端点无效，CTR_RX 位置位（如果 CTR 中断

已使能，将触发中断）。如果传输过程中发生了错误或者缓冲区溢出，前面所列出的动作都不会发生。CRT 中断发生时，应用程序需要首先根据 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别是哪个端点的中断请求。在处理 CTR_RX 中断事件时，应用程序首先要确定传输的类型（根据 USB_EPnR 寄存器的 SETUP 位），同时清除中断标志位，然后读相关的缓冲区描述表表项指向的 COUNTn_RX 寄存器，获得此次传输的总字节数。处理完接收到的数据后，应用程序需要将 USB_EPnR 中的 STAT_RX 位置成‘11’，使能下一次的的传输。当 STAT_RX 位为‘10’时（NAK），任何一个发送到端点上的 OUT 请求都会被 NAK，PC 主机将不断重发被 NAK 的分组，直到收到端点的 ACK 握手分组。以上描述的操作次序是必需遵守的，以避免丢失紧随上一个 CTR 中断的另一个 OUT 分组请求。

### 控制传输

控制传输由 3 个阶段组成，首先是主机发送 SETUP 分组的 SETUP 阶段，然后是主机发送零个或多个数据的数据阶段，最后是状态阶段，由与数据阶段方向相反的数据分组构成。SETUP 传输只发生在控制端点，它非常类似于 OUT 分组的传输过程。使能 SETUP 传输除了需要分别初始化 DTOG_TX 位为‘1’，DTOG_RX 位为‘0’外，还需要设置 STAT_TX 位和 STAT_RX 位为 10(NAK)，由应用程序根据 SETUP 分组的相应位域决定后面的传输是 IN 还是 OUT。控制端点在每次发生 CTR_RX 中断时，都必须检查 USB_EPnR 寄存器的 SETUP 位，以识别是普通的 OUT 分组还是 SETUP 分组。USB 设备应该能够通过 SETUP 分组中的相应数据决定数据阶段传输的字节数和方向，并且能在发生错误的情况下发送 STALL 分组，拒绝数据的传输。

因此在数据阶段，未被使用到的方向都应该被设置成 STALL，并且在开始传输数据阶段的最后一个数据分组时，其反方向的传输仍设成 NAK 状态，这样，即使主机立刻改变了传输方向（进入状态阶段），仍然可以保持为等待控制传输结束的状态。在控制传输成功结束后，应用程序可以把 NAK 变为 VALD，如果控制传输出错，就改为 STALL。此时，如果状态分组是由主机发送给设备的，那么 STATUS_OUT 位（USB_EPnR 寄存器中的 EP_KIND）应该被置位，只有这样，在状态传输过程中收到了非零长度的数据分组，才会产生传输错误。在完成状态传输阶段后，应用程序应该清 STATUS_OUT 位，并且将 STAT_RX 设为 VALID 表示已准备好接收一个新的命令请求，STAT_TX 则设为 NAK，表示在下一个 SETUP 分组传输完成前，不接受数据传输的请求。

USB 规范定义 SETUP 分组不能以非 ACK 握手分组来响应，如果 SETUP 分组传输失败，则会引发下一个 SETUP 分组。因此，以 NAK 或 STALL 分组响应主机的 SETUP 分组是被禁止的。

当 STAT_RX 位被设置为‘01’（STALL）或‘10’（NAK）时，如果收到 SETUP 分组，USB 模块会接收分组，开始分组所要求的数据传输，并回送 ACK 握手分组。如果应用程序在处理前一个 CTR_RX 事件时 USB 模块又收到了 SETUP 分组（即 CTR_RX 仍然保持置位），USB 模块会丢掉收到的 SETUP 分组，并且不回答任何握手分组，以此来模拟一个接收错误，迫使主机再次发送 SETUP 分组。这样做是为了避免丢失紧随一次 CTR_RX 中断之后的又一个 SETUP 分组传输。

### 30.4.3 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理（对于 IN 分组来说，情况是一样的）。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_EPnR 寄存器必需配置为单向。所以只需要设定 STAT_RX 位（作为双缓冲批量接收端点）或者 STAT_TX 位（作为双缓冲批量发送端点）。如果需要一个双向的双缓冲批量端点，则须使用两个 USB_EPnR 寄存器。

为尽可能利用双缓冲的优势，达到较高的传输速率，双缓冲批量端点的流量控制流程与其他端点的稍有不同。它只在缓冲区发生访问冲突时才会设置端点为 NAK 状态，而不是在每次传输成功后都将端点设为 NAK 状态。DTOG 位用来标识 USB 模块当前所使用的储存缓冲区。双缓冲批量端点接收方向的缓冲区由 DTOG_RX（USB_EPnR 寄存器的第 14 位）标识，而双缓冲批量端点发送方向的缓冲区由 DTOG_TX（USB_EPnR 寄存器的第 6 位）标识。同时，USB 模块也需要知道当前哪个缓冲区正在被应用程序使用，以避免发生冲突。由于 USB_EPnR 寄存器中有 2 个 DTOG 位，而 USB 模块只使用其中的一位来标识硬件所使用的缓冲区，因此，应用程序可使用另一位来标识当前正在使用哪个缓冲区，这个新的标识被称为 SW_BUF 位。下表列出了双缓冲批量端点在实现发送和接收操作时，USB_EPNR 寄存器的 DTOG 位和 SW_BUF 位之间的关系。

表 30.1 双缓冲批量端点缓冲区标识定义

| 缓冲区标识位 | 作为发送端点 | 作为接收端点 |
|---|---|---|
| DTOG | DTOG_TX（USB_EPnR 寄存器的第 6 位） | DTOG_RX（USB_EPnR 寄存器的第 14 位） |
| SW_BUF | USB_EPnR 寄存器的第 14 位 | USB_EPnR 寄存器的第 6 位 |

USB 模块当前使用的缓冲区由 DTOG 位标识，而应用程序所使用的缓冲区由 SW_BUF 位标识，这两个位的标识方式相同，下表描述了这种标识方式。

表 30.2 双缓冲批量端点的缓冲区使用标识

| 端点类型 | DTOG 位 | SW_BUF 位 | USB 模块使用的缓冲区 | 应用程序使用的缓冲区 |
|---|---|---|---|---|
| IN 端点 | 0 | 1 | ADDRn_TX_0 / COUNTn_TX_0 | ADDRn_TX_1 / COUNTn_TX_1 |
| | 1 | 0 | ADDRn_TX_1 / COUNTn_TX_1 | ADDRn_TX_0 / COUNTn_TX_0 |
| | 0 | 0 | 无(1) | ADDRn_TX_0 / COUNTn_TX_0 |
| | 1 | 1 | 无(1) | ADDRn_TX_0 / COUNTn_TX_0 |
| OUT 端点 | 0 | 1 | ADDRn_RX_0 / COUNTn_RX_0 | ADDRn_RX_1 / COUNTn_RX_1 |
| | 1 | 0 | ADDRn_RX_1 / COUNTn_RX_1 | ADDRn_RX_0 / COUNTn_RX_0 |
| | 0 | 0 | 无(1) | ADDRn_RX_0 / COUNTn_RX_0 |
| | 1 | 1 | 无(1) | ADDRn_RX_0 / COUNTn_RX_0 |

1. 端点处于 NAK 状态

可以通过以下方式设置一个双缓冲批量端点：

- 将 USB_EPnR 寄存器的 EP_TYPE 位设为‘00’，定义端点为批量端点
- 将 USB_EPnR 寄存器的 EP_KIND 位设为‘1’，定义端点为双缓冲端点

应用程序根据传输开始时用到的缓冲区来初始化 DTOG 和 SW_BUF 位；这需要考虑到这两位的数据翻转特性。设置好 DBL_BUF 位之后，每完成一次传输后，USB 模块将根据双缓冲批量端点的流量控制操作，并且持续到 DBL_BUF 变为无效为止。每次传输结束，根据端点的传输方向，CTR_RX 位或 CTR_TX 位将会置为‘1’。与此同时，硬件将设置相应的 DTOG 位，完全独立于软件来实现缓冲区交换机制。DBL_BUF 位设置后，每次传输结束时，双缓冲批量端点的 STAT 位的取值不会像其他类型端点一样受到传输过程的影响，而是一直保持为‘11’（有效）。但是，如果在收到新的数据分组的传输请求时，USB 模块和应用程序发生了缓冲区访问冲突（即 DTOG 和 SW_BUF 为相同的值，见上表），状态位将会被置为‘10’（NAK）。应用程序响应 CTR 中断时，首先要清除中断标志，然后再处理传输完成的数据。应用程序访问缓冲区之后，需要翻转 SW_BUF 位，以通知 USB 模块该块缓冲区已变为可用状态。由此，双缓冲批量传输的 NAK 分组的数目只由应用程序处理一次数据传输的快慢所决定：如果数据处理的时间小于 USB 总线上完成一次数据传输的时间，则不会发生重传，此时，数据的传输率仅受限于 USB 主机。

应用程序也可以不考虑双缓冲批量端点的特殊控制流程，直接在相应 USB_EPnR 寄存器的 STAT 位写入非‘11’的任何状态，在这种情况下，USB 模块将按照写入的状态执行流程而忽略缓冲器实际的使用情况。

### 30.4.4 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组（由端点传输方向确定分组类型）。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送 ACK 分组。同样，同步传输只传送 PID（分组 ID）为 DATA0 的数据包，而不会用到数据翻转机制。

通过设置 USB_EPnR 寄存器 EP_TYPE 为‘10’，可以使其成为同步端点。同步端点没有握手机制，根据 USB 标准中的说明，USB_EPnR 寄存器的 STAT_RX 位和 STAT_TX 位分别只能设成‘00’（禁止）和‘11’（有效）。同步传输通过实现双缓冲机制来简化软件应用程序开发，它同样使用两个缓冲区，以确保在 USB 模块使用其中一块缓冲区时，应用程序可以访问另外一块缓冲区。

USB 模块使用的缓冲区根据不同的传输方向，由不同的 DTOG 位来标识。（同一寄存器中的 DTOG_RX 位用来标识接收同步端点，DTOG_TX 位用来标识发送同步端点），见下表。

表 30.3 同步端点的缓冲区使用标识

| 端点类型 | DTOG 位值 | USB 模块使用的缓冲区 | 应用程序使用的缓冲区 |
|---|---|---|---|
| IN 端点 | 0 | ADDRn_TX_0 / COUNTn_TX_0 | ADDRn_TX_1 / COUNTn_TX_1 |
| | 1 | ADDRn_TX_1 / COUNTn_TX_1 | ADDRn_TX_0 / COUNTn_TX_0 |
| OUT 端点 | 0 | ADDRn_RX_0 / COUNTn_RX_0 | ADDRn_RX_1 / COUNTn_RX_1 |
| | 1 | ADDRn_RX_1 / COUNTn_RX_1 | ADDRn_RX_0 / COUNTn_RX_0 |

与双缓冲批量端点一样，一个 USB_EPnR 寄存器只能处理同步端点单方向的数据传输，如果要求同步端点在两个传输方向上都有效，则需要使用两个 USB_EPnR 寄存器。

应用程序需要根据首次传输的数据分组来初始化 DTOG 位；它的取值还需要考虑到 DTOG_RX 或 DTOG_TX 两位的数据翻转特性。每次传输完成时，USB_EPnR 寄存器的 CTR_RX 位或 CTR_TX 位置位。与此同时，相关的 DTOG 位由硬件翻转，从而使得交换缓冲区的操作完全独立于应用程序。传输结束时，STAT_RX 或 STAT_TX 位不会发生变化，因为同步传输没有握手机制，所以不需要任何流量控制，而一直设为‘11’（有效）。同步传输中，即使 OUT 分组发生 CRC 错误或者缓冲区溢出，本次传输仍被看作是正确的，并且可以触发 CTR_RX 中断事件；但是，发生 CRC 错误时硬件会设置 USB_ISTR 寄存器的 ERR 位，提醒应用程序数据可能损坏。

### 30.4.5 挂起/恢复事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 2.5mA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 SB_ISTR 寄存器的 SUSP 位，以触发挂起中断。

USB 设备进入挂起状态之后，将由“唤醒”序列唤醒。所谓的“唤醒”序列，可以由 USB 主机发起，也可以由 USB 设备本身触发；但是，只有 USB 主机可以结束“唤醒”序列。被挂起的 USB 模块必须至少还具备检测 RESET 信号的功能，它会将其当作一次正常的复位操作来执行。

实际的挂起操作过程对于不同的 USB 设备来说是不同的，因为需要不同的操作来降低电源消耗。

下面描述了一起典型的挂起操作，重点介绍应用程序如何响应 USB 模块的 SUSP 信号。

1. 将 USB_CNTR 寄存器的 FSUSP 置为‘1’，这将使 USB 模块进入挂起状态。USB 模块一旦进入挂起状态，对 SOF 的检测立刻停止，以避免在 USB 挂起时又发生新的 SUSP 事件。
2. 消除或减少 USB 模块以外的其他模块的静态电流消耗。
3. 将 USB_CNTR 寄存器的 LP_MODE 位置为‘1’，这将消除模拟 USB 收发器的静态电流消耗，但仍能检测到唤醒信号。
4. 可以选择关闭外部振荡器和设备的 PLL，以停止设备内部的任何活动。

当设备处于挂起状态时发生 USB 事件，该设备会被唤醒，并需要调用“唤醒”例程来恢复系统时钟，和 USB 数据传输。如果唤醒设备的是 USB 复位操作，则应该保证唤醒的过程不要超过 10 毫秒（参见“USB 协议规范”）。USB 模块处于挂起状态时，唤醒或复位事件需要清除 USB_CNTR 寄存器的 LP_MODE 位。即使唤醒事件可以立刻触发一个 WKUP 中断事件，但由于恢复系统时钟需要比较长的延迟时间，处理 WKUP 中断的中断服务程序必须非常小心；为了减短系统唤醒的时间，建议将唤醒代码直接写在挂起代码后面，这样就可以在系统时钟重启后迅速进入唤醒代码中执行。为防止或减少 ESD 等干扰意外地唤醒系统（从挂起模式退出是一个异步事件），在挂起过程中数据线被过滤，滤波宽度大约为 70nS。

下面是唤醒操作的过程：

1. 启动外部振荡器和设备的 PLL（此项可选）。
2. 清零 USB_CNTR 寄存器的 FSUSP 位。
3. USB_FNR 寄存器的 RXDP 和 RXDM 位可以用来判断是什么触发了唤醒事件，如下表所示，它还同时列出了各种情况软件应该采取的操作。如果需要的话，可以通过检测这两位变成‘10’（代表空闲总线状态）的时间来知道唤醒或复位事件的结束。此外，在复位事件结束时 USB_ISTR 寄存器的 RESET 位被置为‘1’，如果 RESET 中断被使能，就会产生中断。此中断应该按正常的复位操作处理。

表 30.4 唤醒事件检测

| [RXDP，RXDM]的状态 | 唤醒事件 | 应用程序应执行的操作 |
|---|---|---|
| 00 | 复位 | 无 |
| 10 | 无（总线干扰） | 恢复到挂起状态 |
| 01 | 恢复挂起 | 无 |
| 11 | 未定义的值（总线干扰） | 恢复到挂起状态 |

设备可能不是被与 USB 模块相关的事件唤醒的（例如一个鼠标的移动可唤醒整个系统）。在这种情况下，先将 USB_CNTR 寄存器的 RESUME 位置为‘1’，然后在 1ms－15ms 之间再把它清为 0 可以启动唤醒序列(这个间隔可以用 ESOF 中断来实现，该中断在内核正常运行时每 1ms 发生一次)。RESUME 位被清零后，唤醒过程将由主机 PC 完成，可以利用 USB_FNR 寄存器的 RXDP 和 RXDM 位来判断唤醒是否完成。

*注：　只有在*USB *模块被设置为挂起状态时（设置*USB_CNTR *寄存器的*FSUSP *位为*‘1’*），才可以设置* RESUME *位。*

## 30.5 USB 寄存器描述

USB 模块的寄存器有以下三类::

- 通用类寄存器：中断寄存器和控制寄存器
- 端点类寄存器：端点配置寄存器和状态寄存器
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器

缓冲区描述表类寄存器的基地址由 USB_BTABLE 寄存器指定，所有其他寄存器的基地址则为 USB 模块的基地址 0x4000 5C00。由于 APB1 总线按 32 位寻址，因此所有的 16 位寄存器的地址都是按 32 位字对齐的。同样的地址对齐方式也用于从 0x4000 6000 开始的分组缓冲存储区。

在寄存器描述中使用的缩写列表请参考章节 2.1。

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

### 30.5.1 通用寄存器

#### USB 控制寄存器(USB_CNTR)

偏移地址：0x40
复位值：0x0003

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTRM | PMA OVRM | ERRM | WKUPM | SUSPM | RE SETM | SOFM | ESOFM | Reserved | | | RE SUME | FSUSP | LP_ MODE | PDMN | FRES |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

Bit 31:16 保留，必须保持为复位值。

Bit 15 **CTRM**：正确传输（CTR）中断屏蔽位（Correct transfer interrupt mask）

0：正确传输(CTR)中断禁止

1：正确传输(CTR)中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 14 **PMAOVRM**：分组缓冲区溢出中断屏蔽位（Packet memory area over / underrun interrupt mask）

0：PMAOVR 中断禁止

1：PMAOVR 中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 13 **ERRM**：出错中断屏蔽位（Error interrupt mask）

0：出错中断禁止

1：出错中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 12 **WKUPM**：唤醒中断屏蔽位（Wakeup interrupt mask）

0：唤醒中断禁止

1：唤醒中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 11 **SUSPM**：挂起中断屏蔽位（Suspend mode interrupt mask）

0：挂起（SUSP）中断禁止

1：挂起（SUSP）中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 10 **RESETM**：USB 复位中断屏蔽位（USB reset interrupt mask）

0：USB RESET 中断禁止

1：USB RESET 中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 9 **SOFM**：帧首中断屏蔽位（Start of frame interrupt mask）

0：SOF 中断禁止

1：SOF 中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 8 **ESOFM**：期望帧首中断屏蔽位（Expected start of frame interrupt mask）

0：ESOF 中断禁止

1：ESOF 中断使能，在中断寄存器的相应位被置 1 时产生中断。

Bit 7:5 保留，必须保持为复位值。

Bit 4 **RESUME**：唤醒请求（Resume request）

设置此位将向 PC 主机发送唤醒请求。根据 USB 协议，如果此位在 1ms 到 15ms 内保持有效，主机将对 USB 模块实行唤醒操作。

Bit 3 **FSUSP**：强制挂起（Force suspend）

当 USB 总线上保持 3ms 没有数据通信时，SUSP 中断会被触发，此时软件必需设置此位。

0：无效

1：进入挂起模式，USB 模拟收发器的时钟和静态功耗仍然保持。如果需要进入低功耗状态(总线供电类的设备)，应用程序需要先置位 FSUSP 再置位 LP_MODE。

Bit 2 **LP_MODE**：低功耗模式（Low-power mode）

此模式用于在 USB 挂起状态下降低功耗。在此模式下，除了外接上拉电阻的供电，其他的静态功耗都被关闭，系统时钟将会停止或者降低到一定的频率来减少耗电。USB 总线上的活动（唤醒事件）将会复位此位（软件也可以复位此位）。

0：非低功耗模式

1：低功耗模式

Bit 1 **PDWN**：断电模式（Power down）

此模式用于彻底关闭 USB 模块。当此位被置位时，不能使用 USB 模块。

0：退出断电模式

1：进入断电模式

Bit 0 **FRES**：强制 USB 复位（Force USB Reset）

0：清除 USB 复位信号

1：对 USB 模块强制复位，类似于 USB 总线上的复位信号。USB 模块将一直保持在复位状态下直到软件清除此位。如果 USB 复位中断被使能，将产生一个复位中断。

### USB 中断状态寄存器(USB_ISTR)

偏移地址：0x44
复位值：0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTR | PMA OVR | ERR | WKUP | SUSP | RESET | SOF | ESOF | Reserved | | | DIR | EP_ID[3:0] | | | |
| r | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | | r | r | r | r | r |

此寄存器包含所有中断源的状态信息，以供应用程序确认产生中断请求的事件。

寄存器的高 8 位各表示一个中断源。当相关事件发生时，这些位被硬件置位，如果 USB_CNTR 寄存器上的相应位也被置位，则会产生相应的中断。中断服务程序需要检查每个位，在执行必要的操作后必需清除相应的状态位，不然中断信号线一直保持为高，同样的中断会再次被触发。如果同时多个中断标志被设置，也只会产生一个中断。

应用程序可以使用不同的方式处理传输完成中断，以减少中断响应的延迟时间。端点在成功完成一次传输后，CTR 位会被硬件置起，如果 USB_CNTR 上的相应位也被设置的话，就会产生中断。与端点相关的中断标志和 USB_CNTR 寄存器的 CTRM 位无关。这两个中断标志位将一直保持有效，直到应用程序清除了 USB_EPnR 寄存器中的相关中断挂起位（CTR 位是个只读位）。USB 模块有两路中断请求源：

高优先级的 USB IRQ：用于高优先级的端点(同步和双缓冲批量端点)的中断请求，并且该中断不能被屏蔽。

低优先级 USB IRQ：用于其他中断事件，可以是低优先级的不可屏蔽中断，也可以是由 USB_ISTR 寄存器的高 8 位标识的可屏蔽中断。

对于端点产生的中断，应用程序可以通过 DIR 寄存器和 EP_ID 只读位来识别中断请求由哪个端点产生，并调用相应的中断服务程序。

用户在处理同时发生的多个中断事件时，可以在中断服务程序里检查 USB_ISTR 寄存器各个位的顺序来确定这些事件的优先级。在处理完相应位的中断后需要清零该中断标志。完成一次中断服务后，另一中断请求将会产生，用以请求处理剩下的中断事件。

为了避免意外清零某些位，建议使用加载指令，对所有不需改变的位写“1”，对需要清除的位写“0”。对于该寄存器，不建议使用读出－修改－写入的流程，因为在读写操作之间，硬件可能需要设置某些位，而这些位会在写入时被清零。

Bit 31:16 保留，必须保持为复位值。

Bit 15 **CTR**：正确的传输（Correct transfer）

此位在端点正确完成一次数据传输后由硬件置位。应用程序可以通过 DIR 和 EP_ID 位来识别是哪个端点完成了正确的数据传输。

此位应用程序只读

Bit 14 **PMAOVR**：分组缓冲区溢出（Packet memory area over / underrun）

此位在微控制器长时间没有响应一个访问 USB 分组缓冲区请求时由硬件置位。USB 模块通常在以下情况时置位该位：在接收过程中一个 ACK 握手分组没有被发送，或者在发送过程中发生了比特填充错误，在以上两种情况下主机都会要求数据重传。在正常的数据传输中不会产生 PMAOVR 中断。由于失败的传输都将由主机发起重传，应用程序就可以在这个中断的服务程序中加速设备的其他操作，并准备重传。但这个中断不会在同步传输中产生（同步传输不支持重传）因此数据可能会丢失。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 13 **ERR**：出错（Error）

在下列错误发生时硬件会置位此位。

NANS：无应答。主机的应答超时。

CRC：循环冗余校验码错误。数据或令牌分组中的 CRC 校验出错。

BST：位填充错误。PID，数据或 CRC 中检测出位填充错误。

FVIO：帧格式错误。收到非标准帧（如 EOP 出现在错误的时刻，错误的令牌等）。

USB 应用程序通常可以忽略这些错误，因为 USB 模块和主机在发生错误时都会启动重传机制。此位产生的中断可以用于应用程序的开发阶段，可以用来监测 USB 总线的传输质量，标识用户可能发生的错误（连接线松，环境干扰严重，USB 线损坏等）。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 12 **WKUP**：唤醒请求（Wakeup）

当 USB 模块处于挂起状态时，如果检测到唤醒信号，此位将由硬件置位。此时 CTLR 寄存器的 LP_MODE 位将被清零，同时 USB_WAKEUP 被激活，通知设备的其他部分（如唤醒单元）将开始唤醒过程。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 11 **SUSP**：挂起模块请求（Suspend mode request）

此位在 USB 线上超过 3ms 没有信号传输时由硬件置位，用以指示一个来自 USB 总线的挂起请求。USB 复位后硬件立即使能对挂起信号的检测，但在挂起模式下（FSUSP＝1）硬件不会再检测挂起信号直到唤醒过程结束。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 10 **RESET**：USB 复位请求（USB reset request）

此位在 USB 模块检测到 USB 复位信号输入时由硬件置位。此时 USB 模块将复位内部协议状态机，并在中断使能的情况下触发复位中断来响应复位信号。USB 模块的发送和接收部分将被禁止，直到此位被清除。所有的配置寄存器不会被复位，除非应用程序对他们清零。这用来保证在复位后 USB 传输还可以立即正确执行。但设备的地址和端点寄存器会被 USB 复位所复位。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 9 **SOF：**帧首标志（Start of frame）

此位在 USB 模块检测到总线上的 SOF 分组时由硬件置位，标志一个新的 USB 帧的开始。中断服务程序可以通过检测 SOF 事件来完成与主机的 1ms 同步，并正确读出寄存器在收到 SOF 分组时的更新内容（此功能在同步传输时非常有意义）。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 8 **ESOF：**期望帧首标识位（Expected start of frame）

此位在 USB 模块未收到期望的 SOF 分组时由硬件置位。主机应该每毫秒都发送 SOF 分组，但如果 USB 模块没有收到，挂起定时器将触发此中断。如果连续发生 3 次 ESOF 中断，也就是连续 3 次未收到 SOF 分组，将产生 SUSP 中断。即使在挂起定时器未被锁定时发生 SOF 分组丢失，此位也会被置位。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 7:5 保留，必须保持为复位值。

Bit 4 **DIR：**传输方向（Direction of transaction）

此位在完成数据传输产生中断后由硬件根据传输方向写入。

如果 DIR＝0，相应端点的 CTR_TX 位被置位，标志一个 IN 分组（数据从 USB 模块传输到 PC 主机）的传输完成。

如果 DIR＝1，相应端点的 CTR_RX 位被置位，标志一个 OUT 分组（数据从 PC 主机传输到 USB 模块）的传输完成。如果 CTR_TX 位同时也被置位，就标志同时存在挂起的 OUT 分组和 IN 分组。应用程序可以利用该信息访问 USB_EPnR 位对应的操作，它表示挂起中断传输方向的信息。该位为只读。

Bit 3:0 **EP_ID[3:0]：**端点 ID（Endpoint Identifier）

此位在 USB 模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点 0 具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。该位为只读。

### USB 帧编号寄存器（USB_FNR）

偏移地址：0x48
复位值：0x0XXX，X 代表未定义数值

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXDP | RXDM | LCK | LSOF[1:0] | | FN[10:0] | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:16 保留，必须保持为复位值。

Bit 15 **RXDP**：D+状态位（Receive data + line status）
此位用于观察 USB D+数据线的状态，可在挂起状态下检测唤醒条件的出现。

Bit 14 **RXDM**：D-状态位（Receive data - line status）
此位用于观察 USB D-数据线的状态，可在挂起状态下检测唤醒条件的出现。

Bit 13 **LCK**：锁定位（Locked）
USB 模块在复位或唤醒序列结束后会检测 SOF 分组，如果连续检测到至少 2 个 SOF 分组，则硬件会置位此位。此位一旦锁定，帧计数器将停止计数，一直等到 USB 模块复位或总线挂起时再恢复计数。

Bit 12:11 **LSOF[1:0]**：帧首丢失标志位（Lost SOF）
当 ESOF 事件发生时，硬件会将丢失的 SOF 分组的数目写入此位。如果再次收到 SOF 分组，引脚会清除此位。

Bit 10:0 **FN[10:0]**：帧编号（Frame number）
此部分记录了最新收到的 SOF 分组中的 11 位帧编号。主机每发送一个帧，帧编号都会自加，这对于同步传输非常有意义。此部分发生 SOF 中断时更新。

### USB 设备地址寄存器（USB_DADDR）

偏移地址：0x4C
复位值：0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | EF | ADD[6:0] | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:8 保留，必须保持为复位值。

Bit 7 **EF**：USB 模块使能位（Enable function）

此位在需要使能 USB 模块时由应用程序置位。如果此位为 0，USB 模块将停止工作，忽略所有寄存器的设置，不响应任何 USB 通信。

Bit 6:0 **ADD[6:0]**：设备地址（evice address）

此位记录了 USB 主机在枚举过程中为 USB 设备分配的地址值。该地址值和端点地址（EA）必需和 USB 令牌分组中的地址信息匹配，才能在指定的端点进行正确的 USB 传输。

### USB 分组缓冲区描述表地址寄存器（USB_BTABLE）

偏移地址：0x50
复位值：0x0000

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

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:3 **BTABLE[15:3]**：缓冲表（Buffer table）

此位记录分组缓冲区描述表的起始地址。分组缓冲区描述表用来指示每个端点的分组缓冲区地址和大小，按 8 字节对齐（即最低 3 位为 000）。每次传输开始时，USB 模块读取相应端点所对应的分组缓冲区描述表获得缓冲区地址和大小信息。

Bit 2:0 保留，必须保持为复位值。

### 30.5.2 端点寄存器

端点寄存器的数量由 USB 模块所支持的端点数目决定。USB 模块最多支持 8 个双向端点。每个 USB 设备必须支持一个控制端点，控制端点的地址(EA 位)必需为 0。不同的端点需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_EpnR 寄存器，用于存储该端点的各种状态信息。

**USB 端点 n 寄存器（USB_EPnR）（n=[0..7]）**

偏移地址：0x00 至 0x1C
复位值：0x0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTR _RX | DTOG _RX | STAT_RX[1:0] | | SETUP | EP TYPE[1:0] | | EP_ KIND | CTR _TX | DTOG _TX | STAT_TX[1:0] | | EA[3:0] | | | |
| rc_w0 | t | t | t | r | rw | rw | rw | rc_w0 | t | t | t | rw | rw | rw | rw |

当 USB 模块收到 USB 总线复位信号，或 CTLR 寄存器的 FRES 位置位时，USB 模块将会复位。该寄存器除了 CTR_RX 和 CTR_TX 位保持不变以处理紧随的 USB 传输外，其他位都被复位。每个端点对应一个 USB_EPnR 寄存器，其中 n 为端点地址，即端点 ID 号

对于此类寄存器应避免执行读出－修改－写入操作，因为在读和写操作之间，硬件可能会设置某些位，而这些位又会在写入时被修改，导致应用程序错过相应的操作。因此，这些位都有一个写入无效的值，建议用 Load 指令修改这些寄存器，以免应用程序修改了不需要修改的位。

Bit 31:16 保留，必须保持为复位值。

Bit 15 **CTR_RX：**正确接收标志位（Correct Transfer for reception）

此位在正确接收到 OUT 或 SETUP 分组时由硬件置位，应用程序只能对此位清零。如果 CTRM 位已置位，相应的中断会产生。收到的是 OUT 分组还是 SETUP 分组可以通过下面描述的 SETUP 位确定。以 NAK 或 STALL 结束的分组和出错的传输不会导致此位置位，因为没有真正传输数据。

此位应用程序可读可写，但只有写 0 有效，写 1 无效。

Bit 14 **DTOG_RX：**用于数据接收的数据翻转位（Data Toggle，for reception transfers）

对于非同步端点，此位由硬件设置，用于标记希望接收的下一个数据分组的 Toggle 位（0=DATA0，1=DATA1）。在接收到 PID（分组 ID）正确的数据分组之后，USB 模块发送 ACK 握手分组，并翻转此位。对于控制端点，硬件在收到 SETUP 分组后清除此位。

对于双缓冲端点，此位还用于支持双缓冲区的交换（请参考章节：双缓冲端点）。

对于同步端点，由于仅发送 DATA0，因此此位仅用于支持双缓冲区的交换（请参考章节：同步传输）而不需进行翻转。同步传输不需要握手分组，因此硬件在收到数据分组后立即设置此位。

应用程序可以对此位进行初始化(对于非控制端点，初始化是必需的)，或者翻转此位用于特殊用途。

此位应用程序可读可写，但写 0 无效，写 1 可以翻转此位。

Bit 13:12 **STAT_RX[1:0]：**用于数据接收的状态位（Status Bit，for reception transfers）

此位用于指示端点当前的状态，表：接收状态编码列出了端点的所有状态。当一次正确的 OUT 或 SETUP 数据传输完成后(CTR_RX=1)，硬件会自动设置此位为 NAK 状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。

对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态（请参考章节：双缓冲端点）。

对于同步端点，由于端点状态只能是有效或禁用，因此硬件不会在正确的传输之后设置此位。

如果应用程序将此位设为 STALL 或者 NAK，USB 模块响应的操作是未定义的。

此位应用程序可读可写，但写 0 无效，写 1 翻转此位。

Bit 11 **SETUP：**SETUP 分组传输完成标志位（Setup transaction completed）

此位在 USB 模块收到一个正确的 SETUP 分组后由硬件置位，只有控制端点才使用此位。在接收完成后(CTR_RX=1)，应用程序需要检测此位以判断完成的传输是否是 SETUP 分组。为了防止中断服务程序在处理 SETUP 分组时下一个令牌分组修改了此位，只有 CTR_RX 为 0 时，此位才可以被修改，CTR_RX 为 1 时不能修改。

此位应用程序只读。

Bit 10:9 **EP_TPYE[1:0]：**端点类型位（Endpoint type）

此位用于指示端点当前的类型，所有的端点类型都在表：端点类型编码中列出。所有的 USB 设备都必需包含一个地址为 0 的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有 SETUP 传输，其他类型的端点无视此类传输。SETUP 传输不能以 NAK 或 STALL 分组响应，如果控制端点在收到 SETUP 分组时处于 NAK 状态，USB 模块将不响应分组，就会出现接收错误。如果控制端点处于 STALL 状态，SETUP 分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的 OUT 分组安装普通端点的方式处理。

批量端点和中断端点的处理方式非常类似，仅在对 EP_KIND 位的处理上有差别。

同步端点的用法请参考章节：同步传输。

Bit 8 **EP_KIND：**端点特殊类型位（Endpoint kind）

此位的需要和 EP_TYPE 位配合使用，具体的定义请参考表：端点特殊类型定义。

DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。详见章节：双缓冲端点。

STATUS_OUT：应用程序设置此位表示 USB 设备期望主机发送一个状态数据分组，此时，设备对于任何长度不为 0 的数据分组都响应 STALL 分组。此功能仅用于控制端点，有利于提供应用程序对于协议层错误的检测。如果 STATUS_OUT 位被清除，OUT 分组可以包含任意长度的数据。

Bit 7 **CTR_TX：**正确发送标志位（Correct transfer for transmission）

此位由硬件在一个正确的 IN 分组传输完成后置位。如果 CTRM 位已被置位，会产生相应的中断。应用程序需要在处理完该事件后清除此位。在 IN 分组结束时，如果主机响应 NAK 或 STALL 则此位不会被置位，因为数据传输没有成功。

此位应用程序可读可写，但写 0 有效，写 1 无效。

Bit 6 **DTOG_RX：**发送数据翻转位（Data Toggle，for transmission transfers）

对于非同步端点，此位用于指示下一个要传输的数据分组的 Toggle 位(0＝DATA0, 1=DATA1)。在一个成功传输的数据分组后，如果 USB 模块接收到主机发送的 ACK 分组，就会翻转此位。对于控制端点，USB 模块会在收到正确的 SETUP PID 后置位此位。

对于双缓冲端点，此位还可用于支持分组缓冲区交换（请参考章节：双缓冲端点）。

对于同步端点，由于只传送 DATA0，因此该位只用于支持分组缓冲区交换（请参考章节：同步传输）。由于同步传输不需要握手分组，因此硬件在接收到数据分组后即设置该位。

应用程序可以初始化该位(对于非控制端点，初始化此位时必需的)，也可以设置该位用于特殊用途。

此位应用程序可读可写，但写 0 无效，写 1 翻转此位。

Bit 5:4 **STAT_TX[1:0]：**用于发送数据的状态位（Status Bit，for transmission transfers）

此位用于标识端点的当前状态，表：发送状态编码列出了所有的状态。应用程序可以翻转这些位来初始化状态信息。在正确完成一次 IN 分组的传输后(CTR_TX=1)，硬件会自动设置此位为 NAK 状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。

对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的（请参考章节：双缓冲端点）。对于同步端点，由于端点的状态只能是有效或禁用，因此硬件不会在数据传输结束时改变端点的状态。如果应用程序将此位设为 STALL 或者 NAK，则 USB 模块后续的操作是未定义的。

此位应用程序可读可写，但写 0 无效，写 1 翻转此位。

Bit 3:0 **EA[3:0]：**端点地址（Endpoint address）

应用程序必需设置此 4 位，在使能一个端点前为它定义一个地址。

表 30.5 接收状态编码

| STAT_RX[1:0] | 描述 |
|---|---|
| 00 | **DISABLED**：端点忽略所有的接收请求 |
| 01 | **STALL**：端点以 STALL 分组响应所有的接收请求 |
| 10 | **NAK**：端点以 NAK 分组响应所有的接收请求 |
| 11 | **VALID**：端点可用于接收 |

表 30.6 端点类型编码

| EP_TYPE[1:0] | 描述 |
|---|---|
| 00 | **BULK**：批量端点 |
| 01 | **CONTROL**：控制端点 |
| 10 | **ISO**：同步端点 |
| 11 | **INTERRUPT**：中断端点 |

表 30.7 端点特殊类型定义

| EP_TYPE[1:0] | | EP_KIND 意义 |
|---|---|---|
| 00 | BULK | **DBL_BUF**：双缓冲端点 |
| 01 | CONTROL | **STATUS_OUT** |
| 10 | ISO | 未使用 |
| 11 | INTERRUPT | 未使用 |

表 30.8 发送状态编码

| STAT_RX[1:0] | 描述 |
|---|---|
| 00 | **DISABLED**：端点忽略所有的发送请求 |
| 01 | **STALL**：端点以 STALL 分组响应所有的发送请求 |
| 10 | **NAK**：端点以 NAK 分组响应所有的发送请求 |
| 11 | **VALID**：端点可用于发送 |

### 30.5.3 缓冲区描述表

虽然缓冲区描述表位于分组缓冲区内，但仍可将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓冲区地址都使用 32 位对齐的地址，而不是 USB_BTABLE 寄存器和缓冲区描述表所使用的地址。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为 0x4000 6000。下面将描述与 USB_EPnR 寄存器相关的缓冲区描述表。

完整的分组缓冲区的说明和缓冲区描述表的用法请参考：分组缓冲区的结构和用途。

**发送缓冲区地址寄存器 n（USB_ADDRn_TX）**

偏移地址：[USB_BTABLE] + n×16
USB 本地地址：[USB_BTABLE] + n×8

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRn_TX[14:0] | | | | | | | | | | | | | | | Res |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | |

Bit 31:16 保留，必须保持为复位值。

Bit 15:1 **ADDRn_TX[15:1]：**发送缓冲区地址（Transmission buffer address）
此位记录了收到下一个 IN 分组时，需要发送的数据所在的缓冲区起始地址。

Bit 0 因为分组缓冲区的地址必须按字对齐，所以此位必须为‘0’。

**发送数据字节数寄存器 n（USB_COUNTn_TX）**

偏移地址：[USB_BTABLE] + n×16 + 4
USB 本地地址：[USB_BTABLE] + n×8 + 2

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

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

Bit 31:10 由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略这些位。

Bit 9:0 **COUNTn_TX[9:0]：**发送数据字节数（Transmission byte count）
此位记录了收到下一个 IN 分组时要传输的数据字节数。

*注：双缓冲区和同步* IN *端点有两个* USB_COUNTn_TX *寄存器：分别为* USB_COUNTn_TX_1 *和* USB_COUNTn_TX_0，*内容如下：*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | COUNTn_TX_1[9:0] | | | | | | | | | |
| | | | | | | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | COUNTn_TX_0[9:0] | | | | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**接收缓冲区地址寄存器 n（USB_ADDRn_RX）**

偏移地址：[USB_BTABLE] + n×16 + 8
USB 本地地址：[USB_BTABLE] + n×8 + 4

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

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:1 **ADDRn_RX[15:1]：**接收缓冲区地址（Reception buffer address）
此位记录了收到下一个 OUT 或者 SETUP 分组时，用于保存数据的缓冲区起始地址

Bit 0 因为分组缓冲区的地址按字对齐，所以此位必需为‘0’。

**接收数据字节数寄存器 n（USB_COUNTn_RX）**

偏移地址：[USB_BTABLE] + n×16 + 12
USB 本地地址：[USB_BTABLE] + n×8 + 6

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

该寄存器用于存放接收分组时需要使用到的两个参数。高 6 位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低 10 位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。(具体信息请参考”USB 2.0 协议规范”)

Bit 31:16 保留，必须保持为复位值。

Bit 15 **BL_SIZE：**存储区块的大小（Block size）

此位用于定义决定缓冲区大小的存储区块的大小。

- 如果 BL_SIZE=0，存储区块的大小为 2 字节，因此能分配的分组缓冲区的大小范围为 2—62 个字节。
- 如果 BL_SIZE=1，存储区块的大小为 32 字节，因此能分配的分组缓冲区的大小范围为 32—512 字节，符合 USB 协议定义的最大分组长度限制。

Bit 14:10 **NUM_BLOCK[4:0]：**存储区块的数目（Number of blocks）

此位用以记录分配的存储区块的数目，从而决定最终使用的分组缓冲区的大小。具体请参考下表。

Bit 9:0 **COUNTn_RX[9:0]：**接收到的字节数（Reception byte count）

此位由 USB 模块写入，用以记录端点收到的最新的 OUT 或 SETUP 分组的实际字节数。

*注：　双缓冲区和同步 IN 端点有两个 USB_COUNTn_RX 寄存器：分别为 USB_COUNTn_RX_1 和 USB_COUNTn_TX_0，内容如下：*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BLSIZE_1 | NUM_BLOCK_1[4:0] | | | | | COUNTn_RX_1[9:0] | | | | | | | | | |
| 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 |
| BLSIZE_0 | NUM_BLOCK_0[4:0] | | | | | COUNTn_RX_0[9:0] | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

表 30.9 分组缓冲区大小的定义

| NUM_BLOCK[4:0]的值 | BL_SIZE=0 时的分组缓冲区大小 | 当 BL_SIZE=1 时的分组缓冲区大小 |
|---|---|---|
| 0 (‘00000) | 不允许使用 | 32 字节 |
| 1 (‘00001) | 2 字节 | 64 字节 |
| 2 (‘00010) | 4 字节 | 96 字节 |
| 3 (‘00011) | 6 字节 | 128 字节 |
| … | … | … |
| 15 (‘01111) | 30 字节 | 512 字节 |
| 16 (‘10000) | 32 字节 | 保留 |
| 17 (‘10001) | 34 字节 | 保留 |
| 18 (‘10010) | 36 字节 | 保留 |
| … | … | … |
| 30 (‘11110) | 60 字节 | 保留 |
| 31 (‘11111) | 62 字节 | 保留 |

# 31 控制器局域网（bxCAN）

## 31.1 bxCAN 简介

bxCAN 是基本扩展 CAN（Basic Extended CAN）的缩写，它支持 CAN 协议 2.0A 和 2.0B。它的设计目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求（优先级特性可软件配置）。

对于安全紧要的应用，bxCAN 提供所有支持时间触发通信模式所需的硬件功能。

## 31.2 bxCAN 主要特点

- 支持 CAN 协议 2.0A 和 2.0B 主动模式
- 波特率最高可达 1 兆位/秒
- 支持时间触发通信功能

**发送**

- 3 个发送邮箱
- 发送报文的优先级特性可软件配置
- 记录发送 SOF 时刻的时间戳

**接收**

- 3 级深度的 2 个接收 FIFO
- 可变的过滤器组
- 标识符列表
- FIFO 溢出处理方式可配置
- 记录接收 SOF 时刻的时间戳

**时间触发通信模式**

- 禁止自动重传模式
- 16 位自由运行定时器
- 可在最后 2 个数据字节发送时间戳

**管理**

- 中断可屏蔽
- 邮箱占用单独 1 块地址空间，便于提高软件效率

*注：* USB *和* CAN *共用一个专用的的* SRAM *存储器用于数据的发送和接收，因此不能同时使用* USB *和* CAN *（共享的* SRAM *被* USB *和* CAN *模块互斥地访问）。*USB *和* CAN *可以同时用于一个应用中但不能在同一个时间使用。*

## 31.3 bxCAN 总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量（每个节点都需要处理）急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

- 需要一个增强的过滤机制来处理各种类型的报文

此外，应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。

- 接收 FIFO 的方案允许，CPU 花很长时间处理应用层任务而不会丢失报文。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间有高效的接口。

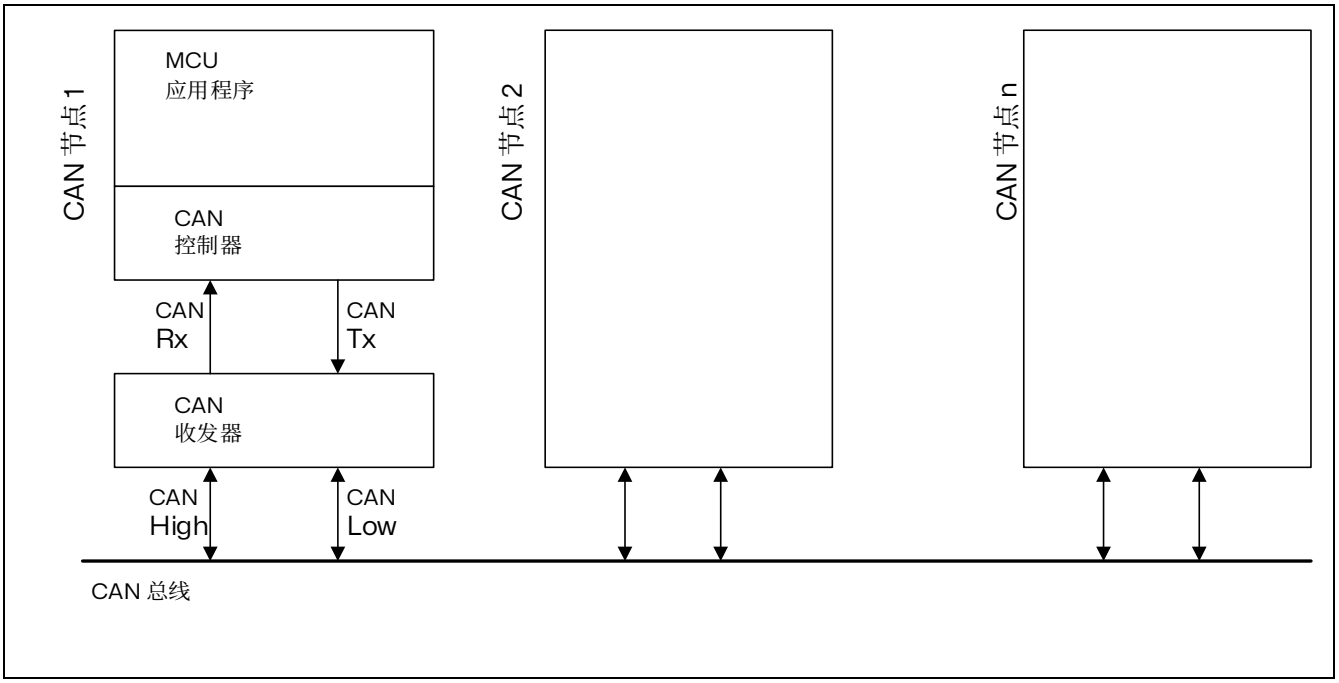



图 31.1 CAN 网拓扑结构

### 31.3.1 CAN 2.0B 主动内核

bxCAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符（11 位）和扩展标识符（29 位）。

### 31.3.2 控制、状态和配置寄存器

应用程序通过这些寄存器，可以：

- 配置 CAN 参数，如波特率
- 请求发送报文
- 处理报文接收
- 管理中断
- 获取诊断信息

### 31.3.3 发送邮箱

共有 3 个发送邮箱供软件来发送报文。发送调度器根据优先级决定哪个邮箱的报文先被发送。

### 31.3.4 接收过滤器

**接收 FIFO**

共有 2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文。它们完全由硬件来管理

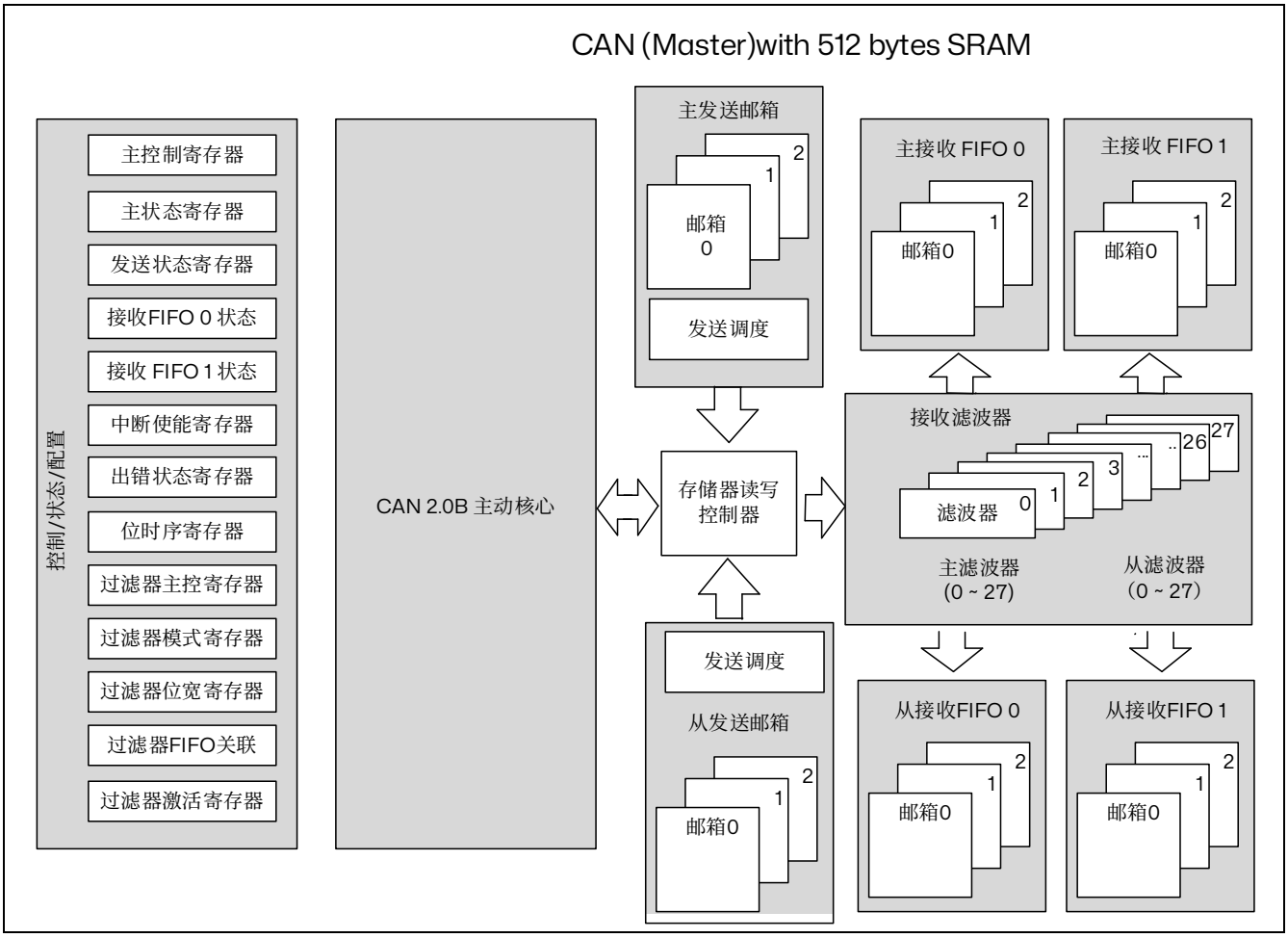



图 31.2 CAN 框图

## 31.4 bxCAN 工作模式

bxCAN 有 3 个主要的工作模式：**初始化**、**正常**和**睡眠**模式。在硬件复位后，bxCAN 工作在睡眠模式以节省电能，同时 CANTX 引脚的内部上拉电阻被激活。软件通过对 CAN_MCR 寄存器的 INRQ 或 SLEEP 位置‘1’，可以请求 bxCAN 进入初始化或睡眠模式。一旦进入了**初始化**或**睡眠**模式，bxCAN 就对 CAN_MSR 寄存器的 INAK 或 SLAK 位置‘1’来进行确认，同时内部上拉电阻被禁用。当 INAK 和 SLAK 位都为‘0’时，bxCAN 就处于**正常**模式。在进入**正常**模式前，bxCAN 必须跟 CAN 总线取得**同步**；为取得同步，bxCAN 要等待 CAN 总线达到空闲状态，即在 CANRX 引脚上监测到 11 个连续的隐性位。

### 31.4.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置 CAN_MCR 寄存器的 INRQ 位为‘1’，请求 bxCAN 进入初始化模式，然后等待硬件对 CAN_MSR 寄存器的 INAK 位置‘1’来进行确认。

清除 CAN_MCR 寄存器的 INRQ 位为‘0’，请求 bxCAN 退出初始化模式，当硬件对 CAN_MSR 寄存器的 INAK 位清‘0’就确认了初始化模式的退出。

当 bxCAN 处于初始化模式时，禁止报文的接收和发送，并且 CANTX 引脚输出隐性位（高电平）。

初始化模式的进入，不会改变配置寄存器。

软件对 bxCAN 的初始化，至少包括位时间特性（CAN_BTR）和控制（CAN_MCR）这 2 个寄存器。

在对 bxCAN 的过滤器组（模式、位宽、FIFO 关联、激活和过滤器值）进行初始化前，软件要对 CAN_FMR 寄存器的 FINIT 位设置‘1’。对过滤器的初始化可以在非初始化模式下进行。

*注：当 FINIT=1 时，报文的接收被禁止。*

*可以先对过滤器激活位清‘0’（在 CAN_FA1R 中），然后修改相应过滤器的值。*

*如果过滤器组没有使用，那么就应该让它处于非激活状态（保持其 FACT 位为清‘0’状态）。*

### 31.4.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。

软件可以通过对 CAN_MCR 寄存器的 INRQ 位清‘0’，来请求从初始化模式进入正常模式，然后要等待硬件对 CAN_MSR 寄存器的 INAK 位置‘0’的确认。在跟 CAN 总线取得同步，即在 CANRX 引脚上监测到 11 个连续的隐性位（等效于总线空闲）后，bxCAN 才能正常接收和发送报文。

不需要在初始化模式下进行过滤器初值的设置，但必须在它处在非激活状态下完成（相应的 FACT 位为 0）。而过滤器的位宽和模式的设置，则必须在初始化模式中进入正常模式前完成。

### 31.4.3 睡眠模式（低功耗）

bxCAN 可工作在低功耗的睡眠模式。软件通过对 CAN_MCR 寄存器的 SLEEP 位置‘1’，来请求进入这一模式。在该模式下，bxCAN 的时钟停止了，但软件仍然可以访问邮箱寄存器。

当 bxCAN 处于**睡眠**模式，软件必须对 CAN_MCR 寄存器的 INRQ 位置‘1’并且同时对 SLEEP 位清‘0’，才能进入**初始化**模式。

有 2 种方式可以唤醒（退出睡眠模式）bxCAN：通过软件对 SLEEP 位清‘0’，或硬件检测到 CAN 总线的活动。

如果 CAN_MCR 寄存器的 AWUM 位为‘1’，一旦检测到 CAN 总线的活动，硬件就自动对 SLEEP 位清‘0’来唤醒 bxCAN。如果 CAN_MCR 寄存器的 AWUM 位为‘0’，软件必须在唤醒中断里对 SLEEP 位清‘0’才能退出睡眠状态。

*注：如果唤醒中断被允许（CAN_IER 寄存器的 WKUIE 位为‘1’），那么一旦检测到 CAN 总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒 bxCAN。*

*在对 SLEEP 位清‘0’后，睡眠模式的退出必须与 CAN 总线同步，下图：bxCAN 工作模式。当硬件对 SLAK 位清‘0’时，就确认了睡眠模式的退出。*

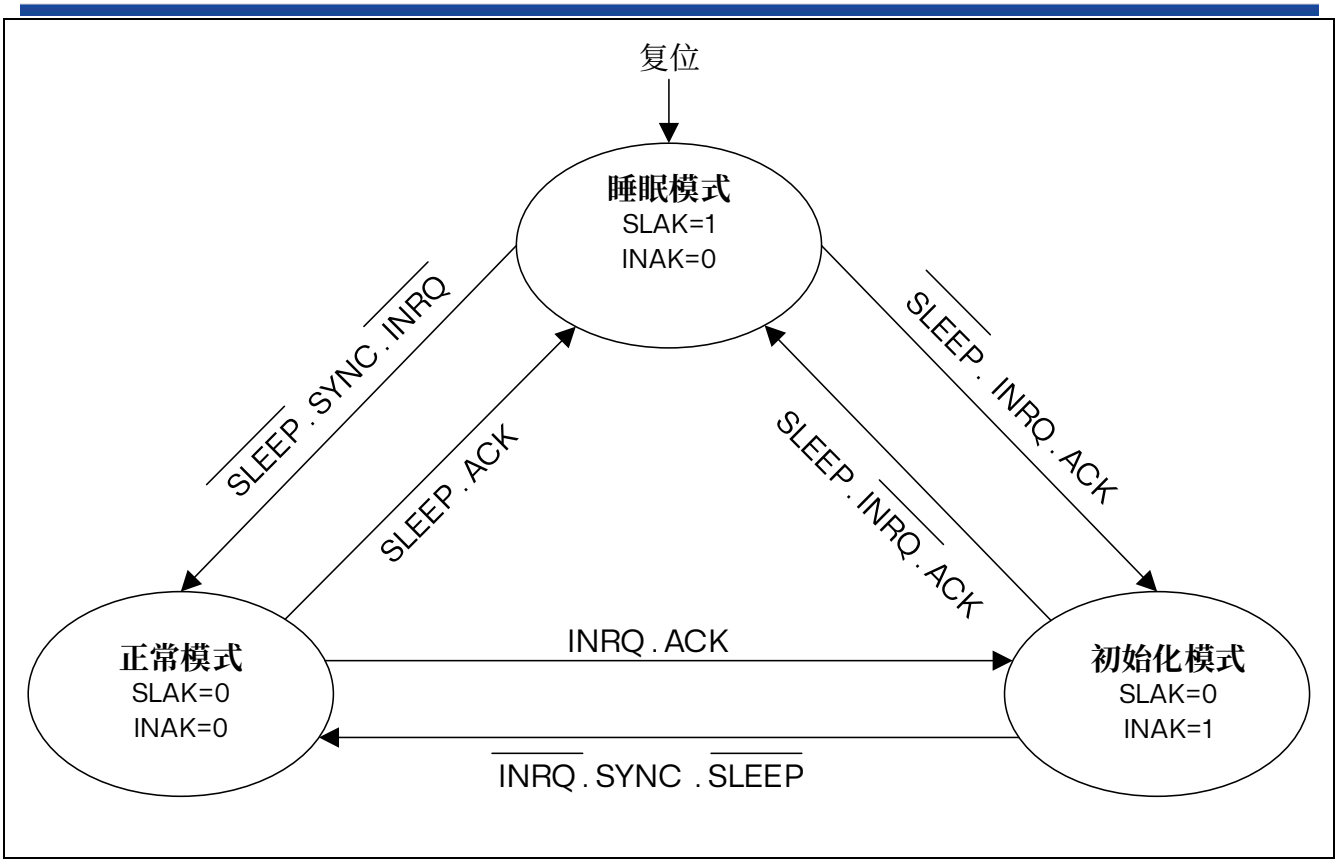



图 31.3 bxCAN 工作模式

1. ACK = *硬件响应睡眠或初始化请求,而对* CAN_MSR *寄存器的* INAK *或* SLAK *位置*1 *的状态*
2. SYNC = bxCAN *等待* CAN *总线变为空闲的状态，即在* CANRX *引脚上检测到连续的*11 *个隐性位*

### 31.5 测试模式

通过对 CAN_BTR 寄存器的 SILM/或 LBKM 位置‘1’，来选择一种测试模式。只能在初始化模式下，修改这 2 位。在选择了一种测试模式后，软件需要对 CAN_MCR 寄存器的 INRQ 位清‘0’，来真正进入测试模式。

#### 31.5.1 静默模式

通过对 CAN_BTR 寄存器的 SILM 位置‘1’，来选择静默模式。

在静默模式下，bxCAN 可以正常地接收数据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果 bxCAN 需要发出显性位（确认位、过载标志、主动错误标志），那么这样的显性位在内部被接回来从而可以被 CAN 内核检测到，同时 CAN 总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析 CAN 总线的活动，而不会对总线造成影响－显性位（确认位、错误帧）不会真正发送到总线上。

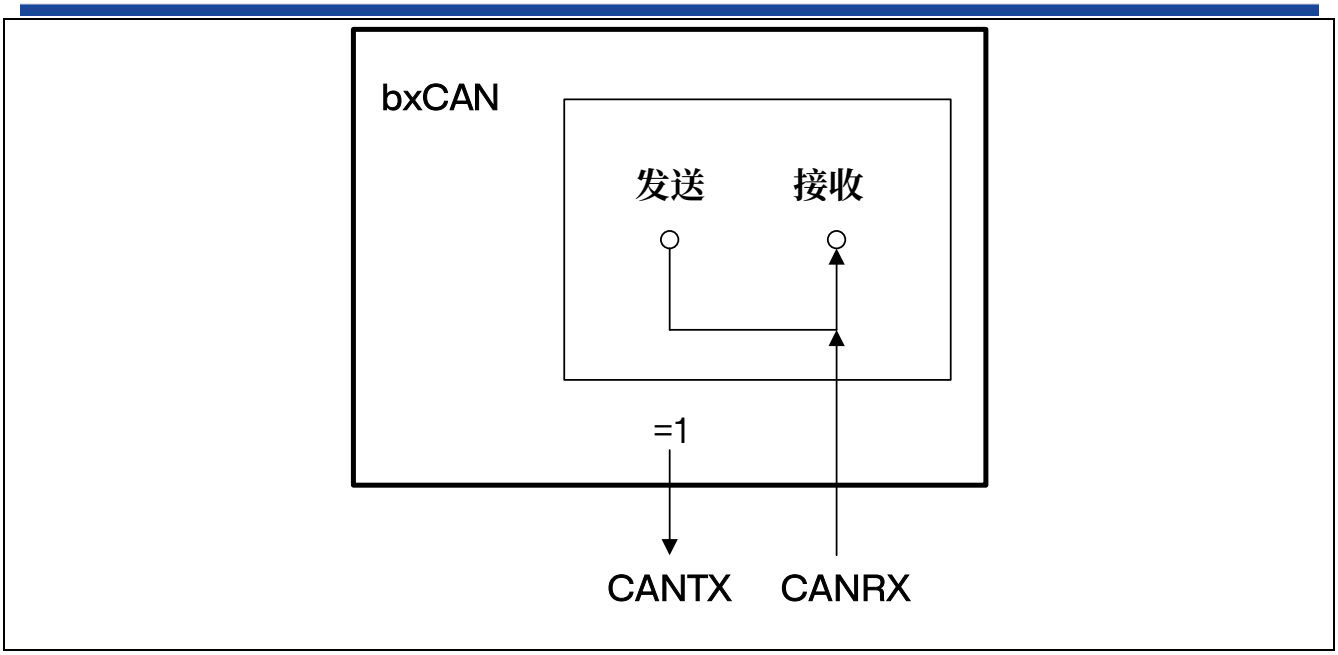



图 31.4 bxCAN 工作在静默模式

### 31.5.2 环回模式

通过对 CAN_BTR 寄存器的 LBKM 位置‘1’，来选择环回模式。在环回模式下，bxCAN 把发送的报文当作接收的报文并保存（如果可以通过接收过滤）在接收邮箱里。

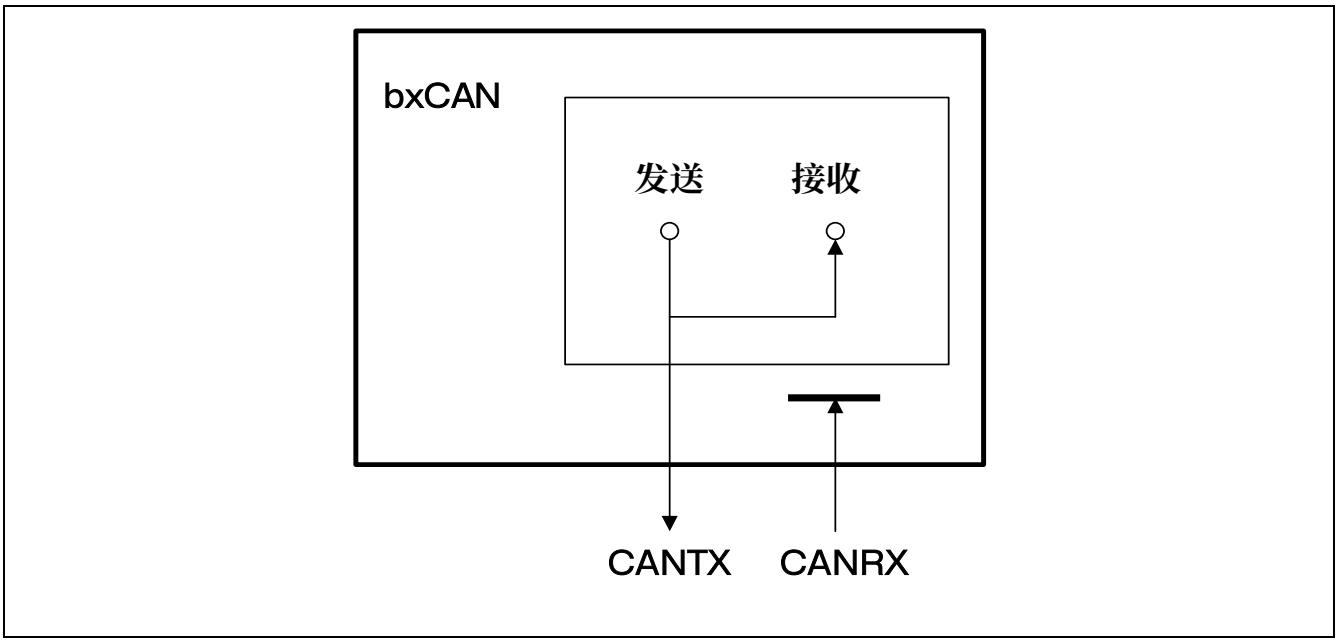



图 31.5 bxCAN 工作在环回模式

环回模式可用于自测试。为了避免外部的影响，在环回模式下 CAN 内核忽略确认错误（在数据/远程帧的确认位时刻，不检测是否有显性位）。在环回模式下，bxCAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态。发送的报文可以在 CANTX 引脚上检测到。

### 31.5.3 环回静默模式

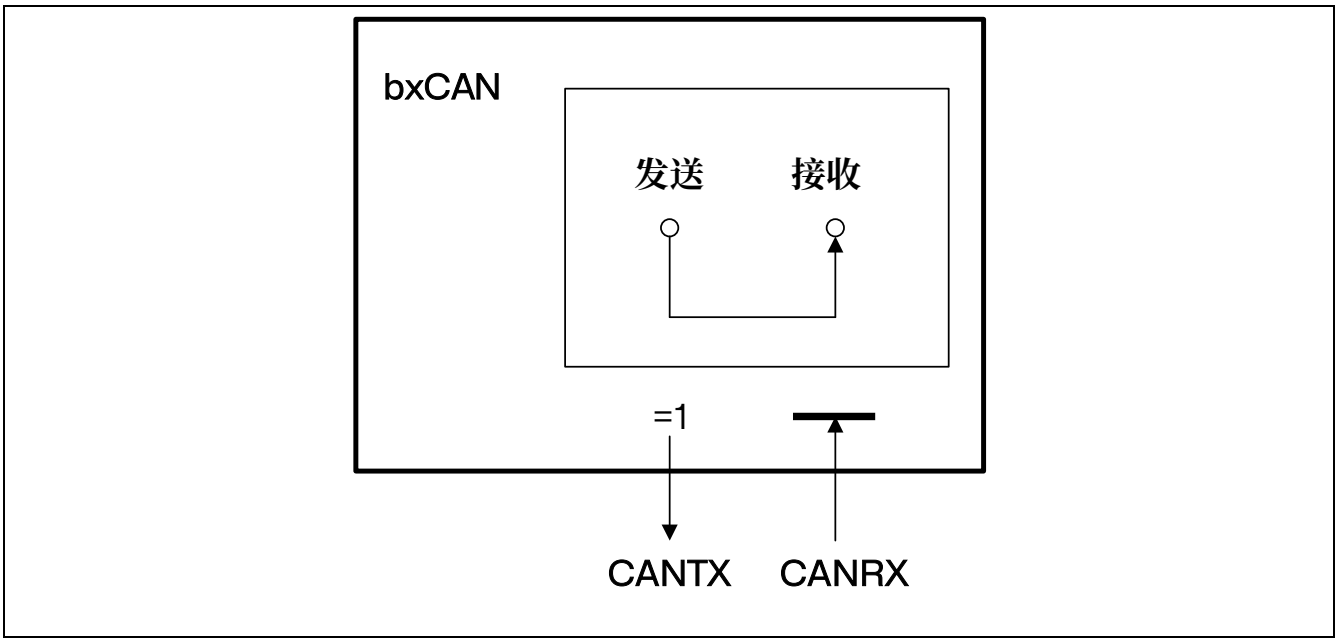



图 31.6 bxCAN 工作在环回静默模式

通过对 CAN_BTR 寄存器的 LBKM 和 SILM 位同时置‘1’，可以选择环回静默模式。该模式可用于“热自测试”，即可以像环回模式那样测试 bxCAN，但却不会影响 CANTX 和 CANRX 所连接的整个 CAN 系统。在环回静默模式下，CANRX 引脚与 CAN 总线断开，同时 CANTX 引脚被驱动到隐性位状态。

## 31.6 处于调试模式时

当微控制器处于调试模式时，内核处于暂停状态，依据下述配置位的状态，bxCAN 可以继续正常工作或停止工作：

- 调试（DBG）模块中 CAN1 的 DBG_CAN1_STOP 位或 CAN2 的 DBG_CAN2_STOP 位。
- CAN_MCR 中的 DBF 位。

## 31.7 bxCAN 功能描述

### 31.7.1 发送处理

发送报文的流程为：应用程序选择 1 个**空置**的发送邮箱；设置标识符，数据长度和待发送数据；然后对 CAN_TIxR 寄存器的 TXRQ 位置‘1’，来请求发送。TXRQ 位置‘1’后，邮箱就不再是空邮箱；而一旦邮箱不再为**空置**，软件对邮箱寄存器就不再有写的权限。TXRQ 位置 1 后，邮箱马上进入**挂号**状态，并等待成为最高优先级的邮箱，参见发送优先级。一旦邮箱成为最高优先级的邮箱，其状态就变为**预定**发送状态。一旦 CAN 总线进入空闲状态，预定发送邮箱中的报文就马上被发送(进入**发送**状态)。一旦邮箱中的报文被成功发送后，它马上变为**空置**邮箱；硬件相应地对 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表明一次成功发送。

如果发送失败，由于仲裁引起的就对 CAN_TSR 寄存器的 ALST 位置‘1’，由于发送错误引起的就对 TERR 位置‘1’。

**发送优先级**

**由标识符决定**

当有超过 1 个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据 CAN 协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

**由发送请求次序决定**

通过对 CAN_MCR 寄存器的 TXFP 位置‘1’，可以把发送邮箱配置为发送 FIFO。在该模式下，发送的优先级由发送请求次序决定。

该模式对分段发送很有用。

**中止**

通过对 CAN_TSR 寄存器的 ABRQ 位置‘1’，可以中止发送请求。邮箱如果处于挂号或预定状态，发送请求马上就被中止了。如果邮箱处于发送状态，那么中止请求可能导致 2 种结果。如果邮箱中的报文被成功发送，那么邮箱变为空置邮箱，并且 CAN_TSR 寄存器的 TXOK 位被硬件置‘1’。如果邮箱中的报文发送失败了，那么邮箱变为预定状态，然后发送请求被中止，邮箱变为空置邮箱且 TXOK 位被硬件清‘0’。因此如果邮箱处于发送状态，那么在发送操作结束后，邮箱都会变为空置邮箱。

**禁止自动重传模式**

该模式主要用于满足 CAN 标准中，时间触发通信选项的需求。通过对 CAN_MCR 寄存器的 NART 位置‘1’，来让硬件工作在该模式。

在该模式下，发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动发送该报文。

在一次发送操作结束后，硬件认为发送请求已经完成，从而对 CAN_TSR 寄存器的 RQCP 位置‘1’，同时发送的结果反映在 TXOK、ALST 和 TERR 位上。

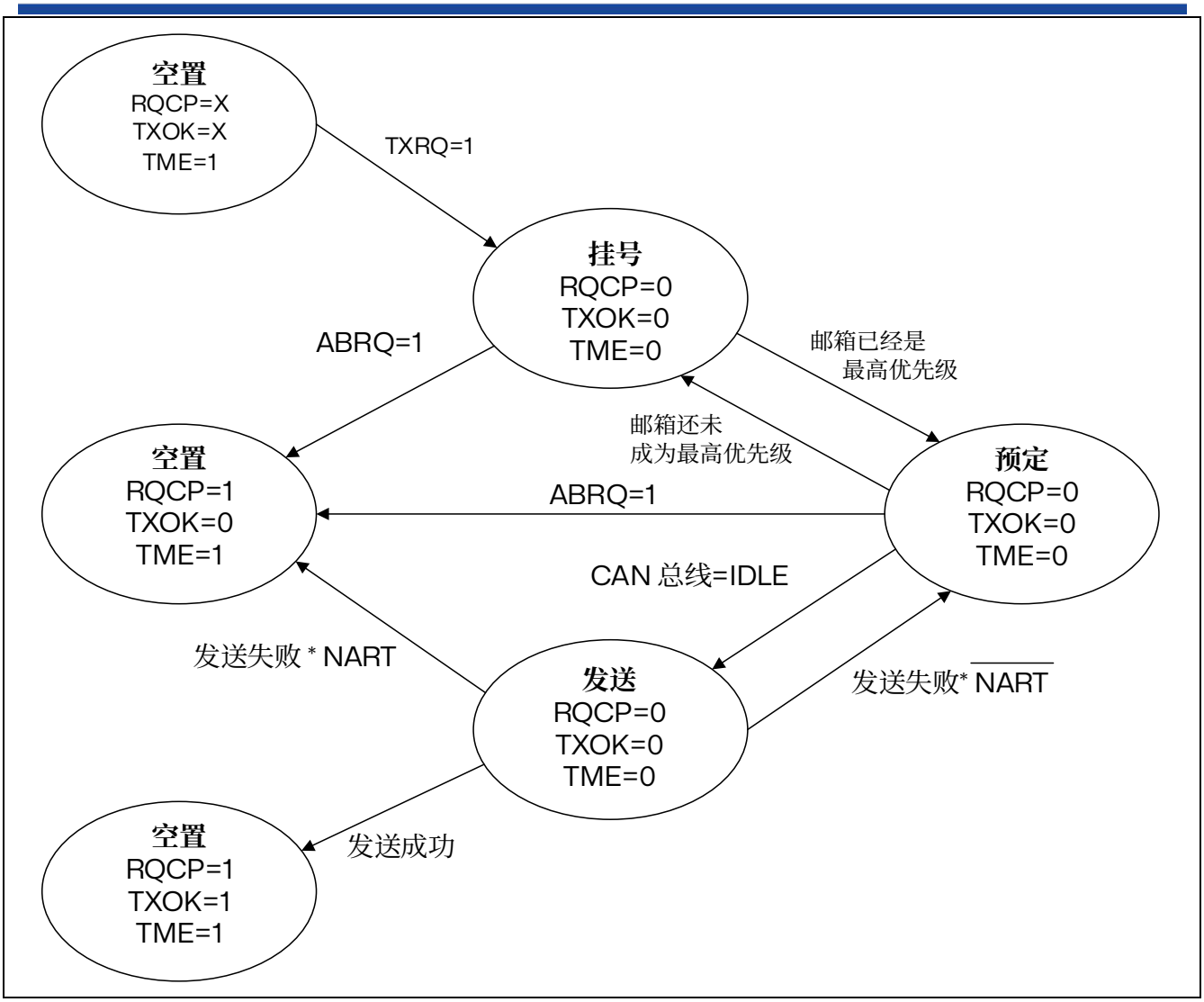



图 31.7 发送邮箱状态

### 31.7.2 时间触发通信模式

在该模式下，CAN 硬件的内部定时器被激活，并且被用于产生（发送与接收邮箱的）时间戳，分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部定时器在每个 CAN 位时间累加（参见章节：位时间特性）。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

### 31.7.3 接收管理

接收到的报文，被存储在 3 级邮箱深度的 FIFO 中。FIFO 完全由硬件来管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序只能通过读取 FIFO 输出邮箱，来读取 FIFO 中最先收到的报文。

**有效报文**

根据 CAN 协议，当报文被正确接收(直到 EOF 域的最后一位都没有错误)，且通过了标识符过滤，那么该报文被认为是有效报文。

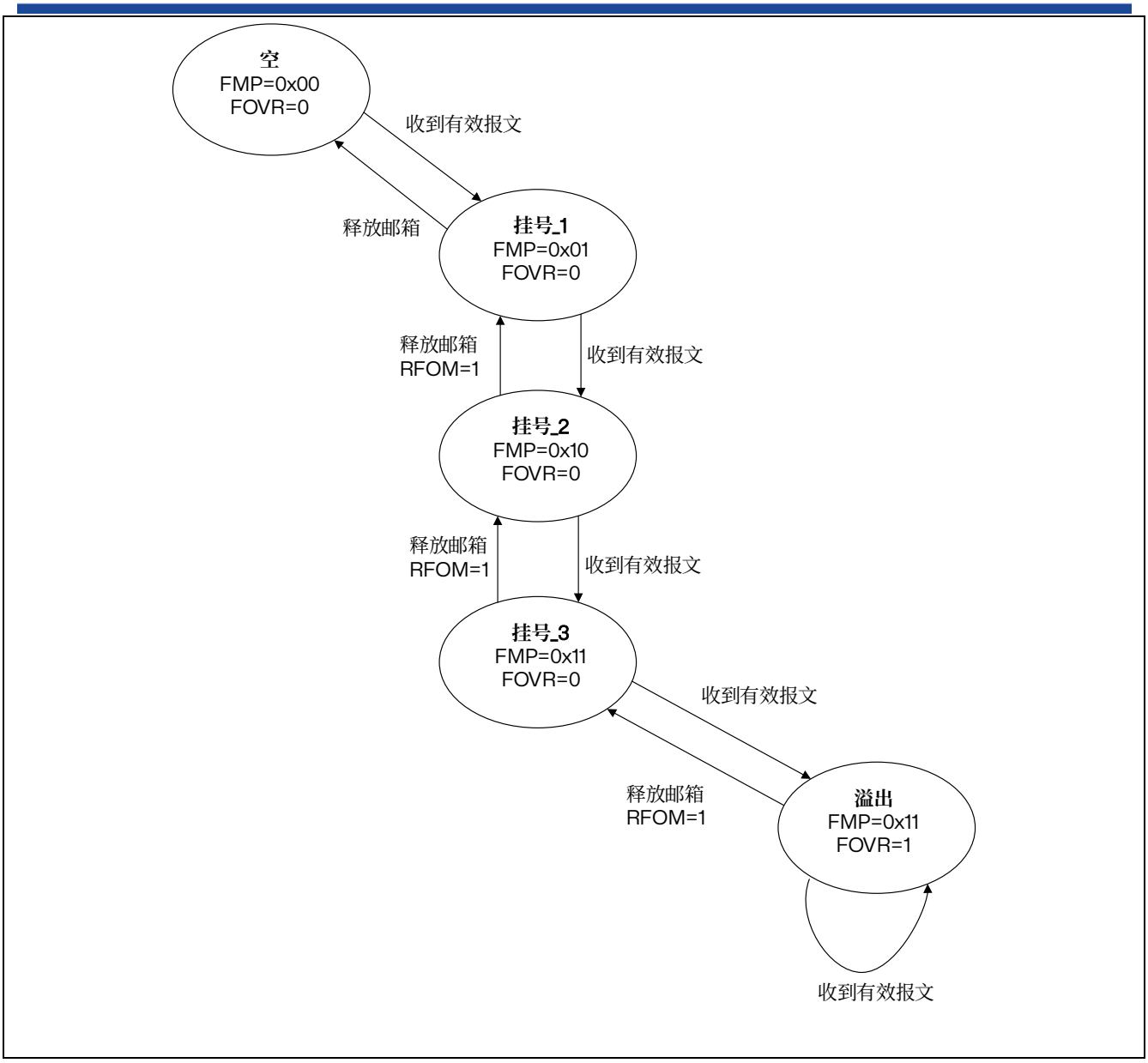



图 31.8 接收 FIFO 状态

**FIFO 管理**

FIFO 从空状态开始，在接收到第一个有效的报文后，FIFO 状态变为挂号_1（pending_1），硬件相应地把 CAN_RFR 寄存器的 FMP[1:0]设置为‘01’（二进制 01b）。软件可以读取 FIFO 输出邮箱来读出邮箱中的报文，然后通过对 CAN_RFR 寄存器的 RFOM 位设置‘1’来释放邮箱，这样 FIFO 又变为空状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么 FIFO 仍然保留在挂号_1 状态，软件可以读取 FIFO 输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO 状态变为挂号_2（pending_2），硬件相应地把 FMP[1:0]设置为‘10’（二进制 10b）。重复上面的过程，第三个有效的报文把 FIFO 变为挂号_3 状态（FMP[1:0]＝11b）。此时，软件必须对 RFOM 位设置 1 来释放邮箱，以便 FIFO 可以有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失

参见章节：报文存储。

**溢出**

当 FIFO 处于挂号_3 状态（即 FIFO 的 3 个邮箱都是满的），下一个有效的报文就会导致溢出，并且一个报文会丢失。此时，硬件对 CAN_RFR 寄存器的 FOVR 位进行置'1'来表明溢出情况。至于哪个报文会被丢弃，取决于对 FIFO 的设置：

- 如果禁用了 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位被清‘0’），那么 FIFO 中最后收到的报文就被新报文所覆盖。这样，最新收到的报文不会被丢弃掉。
- 如果使能了 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位被置‘1’），那么新收到的报文就被丢弃，软件可以读到 FIFO 中最早收到的 3 个报文。

**接收相关的中断**

一旦往 FIFO 存入一个报文，硬件就会更新 FMP[1:0]位，并且如果 CAN_IER 寄存器的 FMPIE 位为‘1’，那么就会产生一个中断请求。

当 FIFO 变满时(即第 3 个报文被存入)，CAN_RFR 寄存器的 FULL 位就被置‘1’，并且如果 CAN_IER 寄存器的 FFIE 位为‘1’，那么就会产生一个满中断请求。

在溢出的情况下，FOVR 位被置‘1’，并且如果 CAN_IER 寄存器的 FOVIE 位为‘1’，那么就会产生一个溢出中断请求。

### 31.7.4 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者乙广播的形式把报文发送给所有的接收者。节点在接收报文时－根据标识符的值－决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，bxCAN 控制器为应用程序提供了 14 个位宽可变的、可配置的过滤器组（13~0），以便只接收那些软件需要的报文。硬件过滤的做法节省了 CPU 开销，否则 就必须由软件过滤从而占用一定的 CPU 开销。每个过滤器组 x 由 2 个 32 位寄存器，CAN_FxR1 和 CAN_FxR2 组成。

**可变的位宽**

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括：STID[10:0]、EXID[17:0]、IDE 和 RTR 位
- 2 个 16 位过滤器，包括：STID[10:0]、IDE、RTR 和 EXID[17:15]位

此外过滤器可配置为，屏蔽位模式和标识符列表模式。

**屏蔽位模式**

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理。

**标识符列表模式**

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用。因此，不是采用一个标识符加一个屏蔽位的方式，而是使用 2 个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

**过滤器组位宽和模式的设置**

过滤器组可以通过相应的 CAN_FMR 寄存器配置。在配置一个过滤器组前，必须通过清除 CAN_FAR 寄存器的 FACT 位，把它设置为禁用状态。通过设置 CAN_FS1R 的相应 FSCx 位，可以配置一个过滤器组的位宽。通过 CAN_FMR 的 FBMx 位，可以配置对应的屏蔽/标识符寄存器的标识符列表模式或屏蔽位模式。

为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号为（叫做过滤器号）从 0 开始，到某个最大数值－取决于过滤器组的模式和位宽的设置。

关于过滤器配置，参见下图。

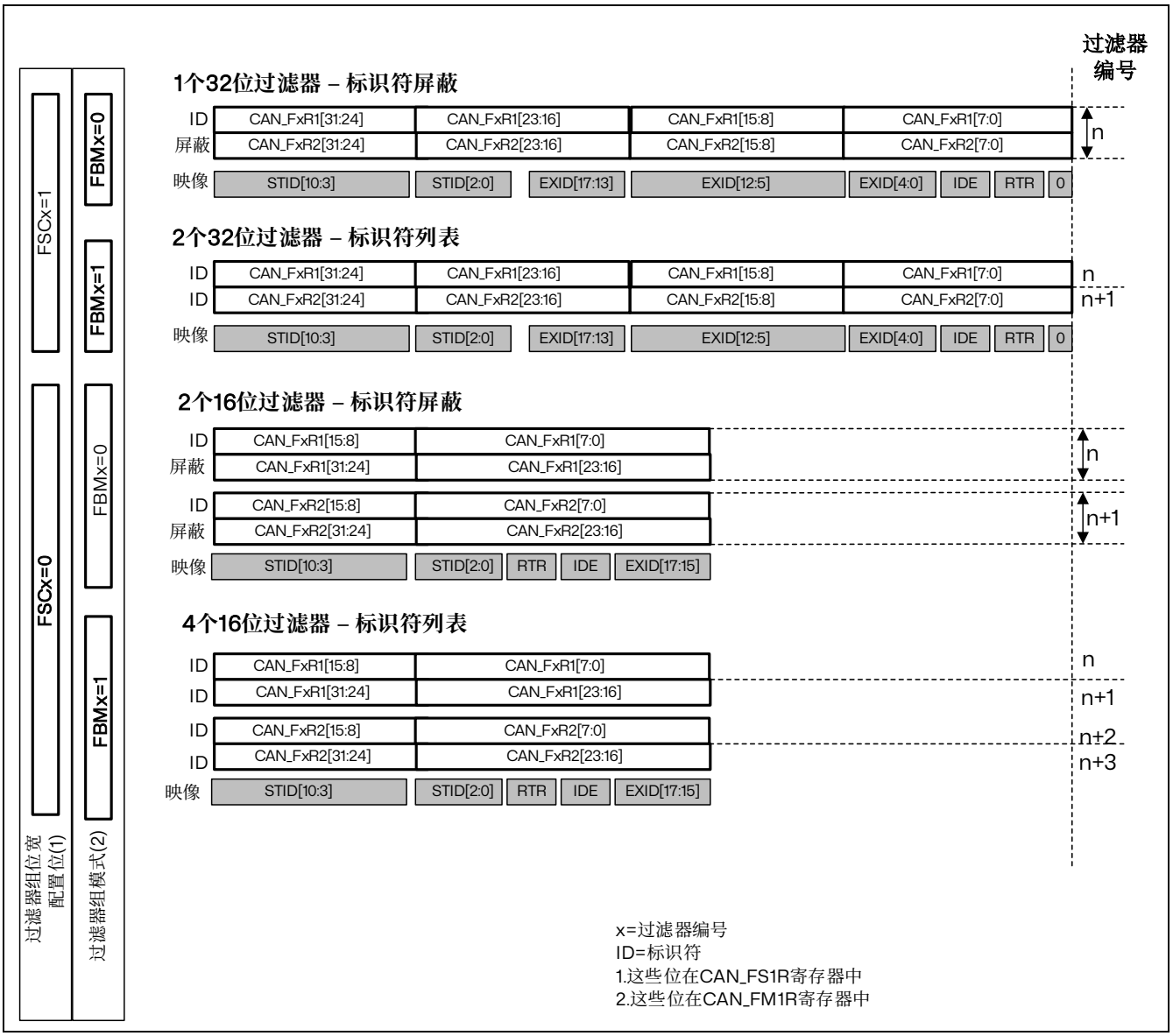



图 31.9 过滤器组位宽设置－寄存器组织

**过滤器匹配序号**

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到 SRAM 中。bxCAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

对于标识符列表模式下的过滤器（非屏蔽方式的过滤器），软件不需要直接跟标识符进行比较。

对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位（必须匹配的位）进行比较即可。在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个 FIFO 各自对其关联的过滤器进行编号。请参考下图的例子。

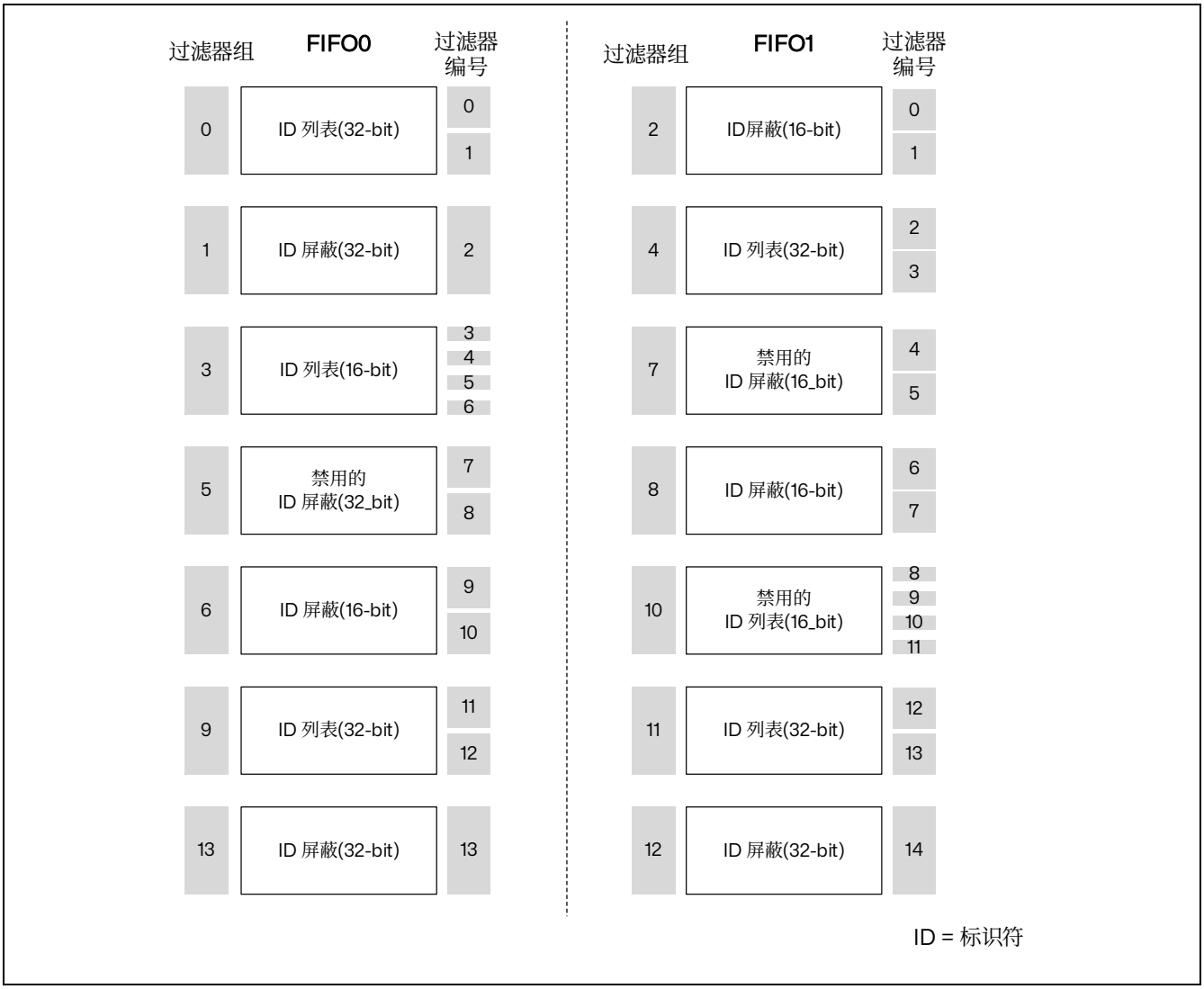



图 31.10 过滤器编号的例子

**过滤器优先级规则**

根据过滤器的不同配置，有可能一个报文标识符能通过多个过滤器的过滤；在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据下列优先级规则来确定：

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高

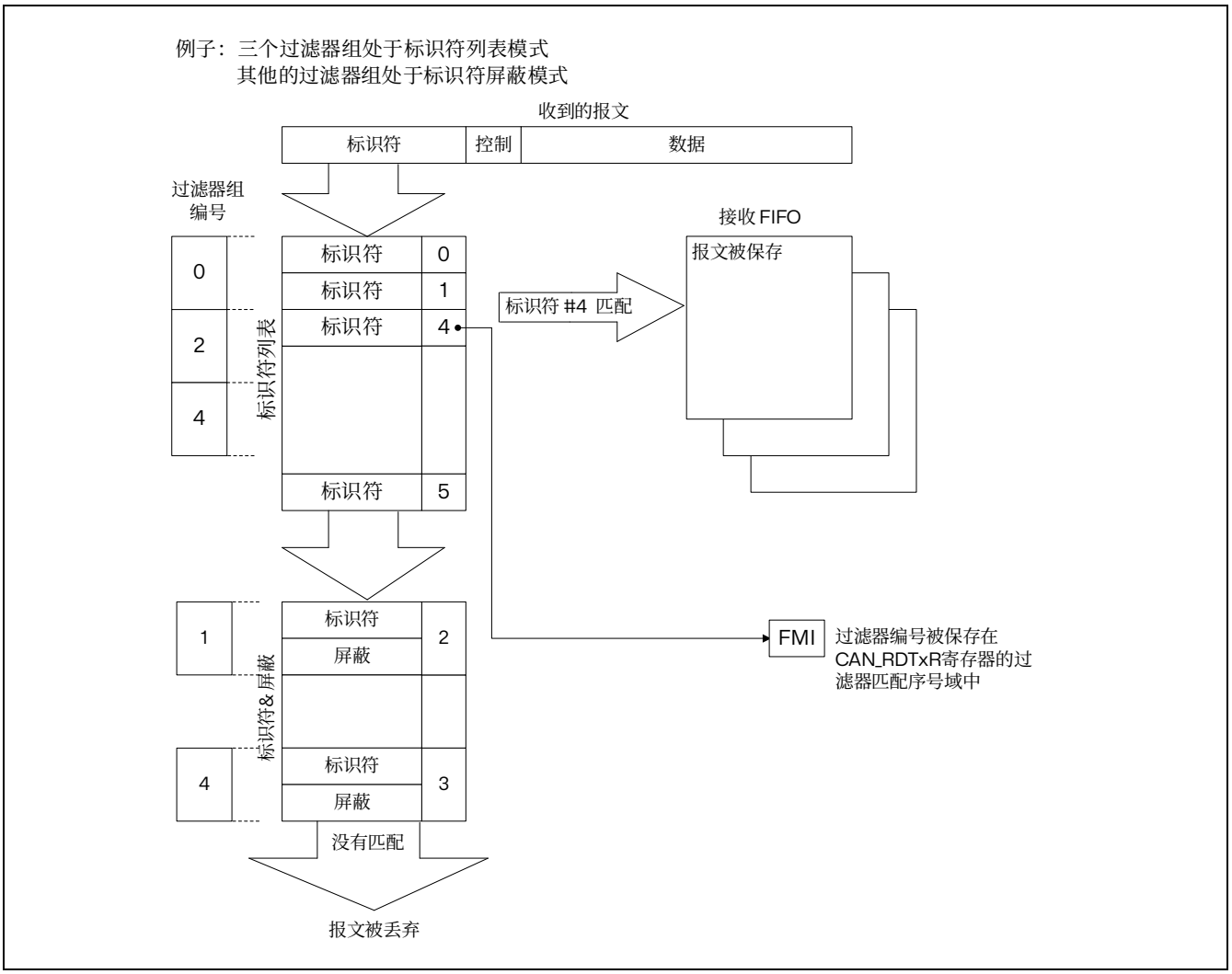



图 31.11 过滤器机制的例子

上面的例子说明了 bxCAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报文文内容和 FMI4 被存入 FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。

如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

### 31.7.5 报文存储

邮箱是软件和硬件之间传递报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

**发送邮箱**

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好（然后再发出发送的请求）。发送的状态可通过查询 CAN_TSR 寄存器获知。

表 31.1 发送邮箱寄存器列表

| 相对发送邮箱基地址的偏移量 | 寄存器名 |
|---|---|
| 0 | CAN_TIxR |
| 4 | CAN_TDTxR |
| 8 | CAN_TDLxR |
| 12 | CAN_TDHxR |

**接收邮箱(FIFO)**

在接收到一个报文后，软件就可以访问接收 FIFO 的输出邮箱来读取它。一旦软件处理了报文（如把它读出来），软件就应该对 CAN_RFxR 寄存器的 RFOM 位进行置‘1’，来释放该报文，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在 CAN_RDTxR 寄存器的 FMI 域中。16 位的时间戳存放在 CAN_RDTxR 寄存器的 TIME[15:0]域中。

表 31.2 接收邮箱寄存器列表

| 相对接收邮箱基地址的偏移量 | 寄存器名 |
|---|---|
| 0 | CAN_RIxR |
| 4 | CAN_RDTxR |
| 8 | CAN_RDLxR |
| 12 | CAN_RDHxR |

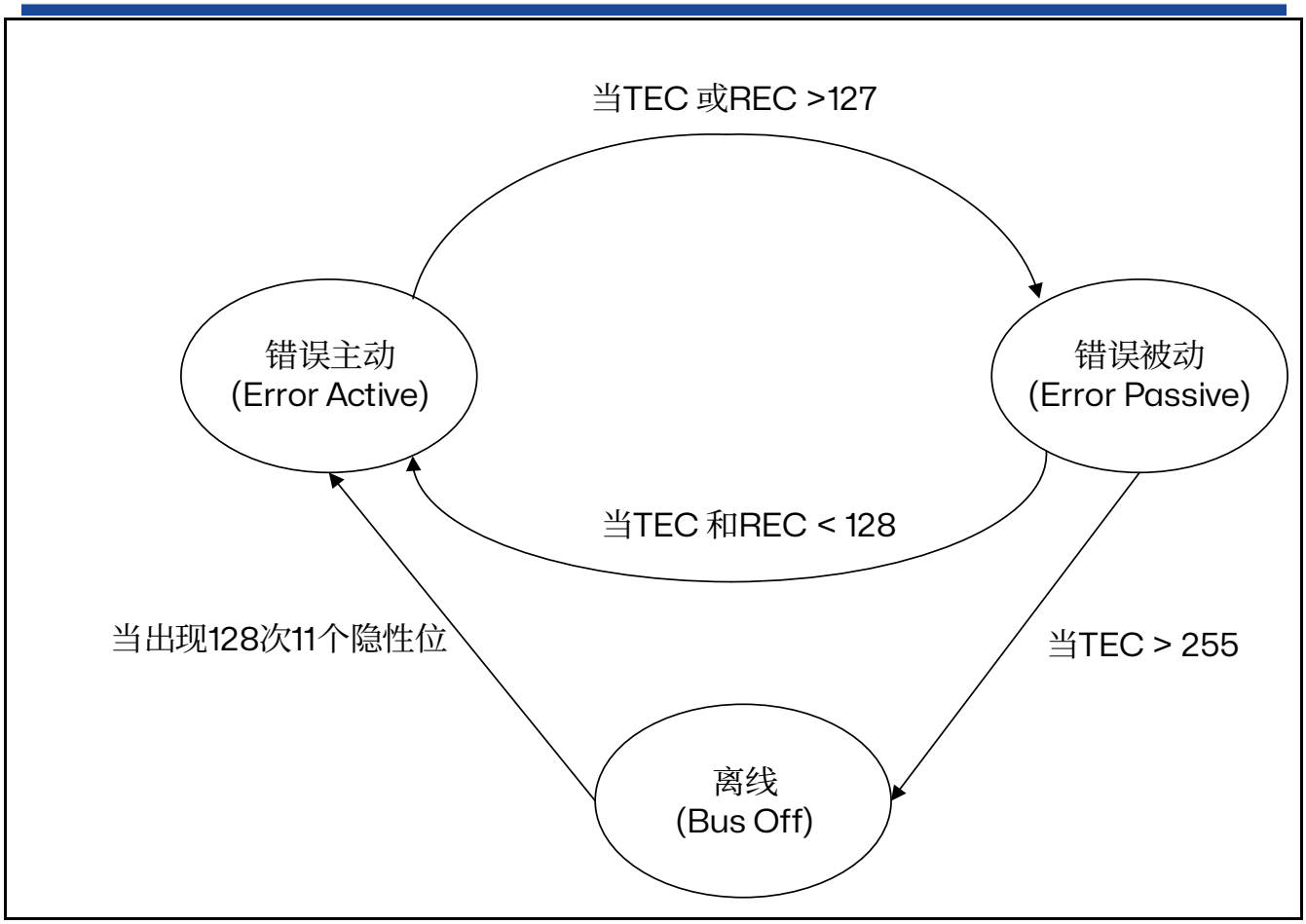



图 31.12 CAN 错误状态图

### 31.7.6 出错管理

CAN 协议描述的出错管理，完全由硬件通过发送错误计数器（CAN_ESR 寄存器里的 TEC 域），和接收错误计数器（CAN_ESR 寄存器里的 REC 域）来实现，其值根据错误的情况而增加或减少。关于 TEC 和 REC 管理的详细信息，请参考 CAN 标准。

软件可以读出它们的值来判断 CAN 网络的稳定性。

此外，CAN_ESR 寄存器提供了当前错误状态的详细信息。通过设置 CAN_IER 寄存器（比如 ERRIE 位），当检测到出错时软件可以灵活地控制中断的产生。

**离线恢复**

当 TEC 大于 255 时，bxCAN 就进入离线状态，同时 CAN_ESR 寄存器的 BOFF 位被置‘1’。在离线状态下，bxCAN 无法接收和发送报文。

根据 CAN_MCR 寄存器中 ABOM 位的设置，bxCAN 可以自动或在软件的请求下，从离线状态恢复（变为错误主动状态）。在这两种情况下，bxCAN 都必须等待一个 CAN 标准所描述的恢复过程（CAN RX 引脚上检测到 128 次 11 个连续的隐性位）。

如果 ABOM 位为 1，bxCAN 进入离线状态后，就自动开启恢复过程。

如果 ABOM 位为 0，软件必须先请求 bxCAN 进入然后再退出初始化模式，随后恢复过程才被开启。

*注：　在初始化模式下，*bxCAN *不会监视* CAN RX *引脚的状态，这样就不能完成恢复过程。为了完成恢复过程，*bxCAN *必须工作在正常模式。*

### 31.7.7 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过与帧起始位的边沿进行同步，即通过与后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位时间分为 3 段：

- **同步段（SYNC_SEG）：**通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元（1 x $t_q$）。
- **时间段（BS1）：**定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- **时间段 2（BS2）：**定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度（SJW）定义了，在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。

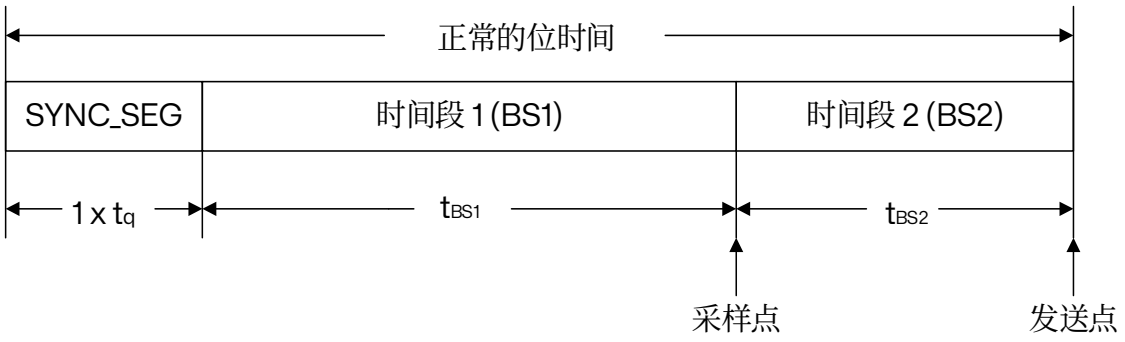
有效跳变被定义为，当 bxCAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。

如果在时间段 1（BS1）而不是在同步段（SYNC_SEG）检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2（BS2）而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器（CAN_BTR）的设置，只能在 bxCAN 处于初始化状态下进行。

*注：　关于CAN 位时间特性和重同步机制的详细信息，请参考*ISO11898 *标准。*




$$波特率 = \frac{1}{正常的位时间}$$

正常的位时间=1 x tq + $t_{BS1}$ + $t_{BS2}$

其中:

$t_{BS1} = t_q$ x (TS1[3:0]+1)

$t_{BS2} = t_q$ x (TS2[2:0]+1)

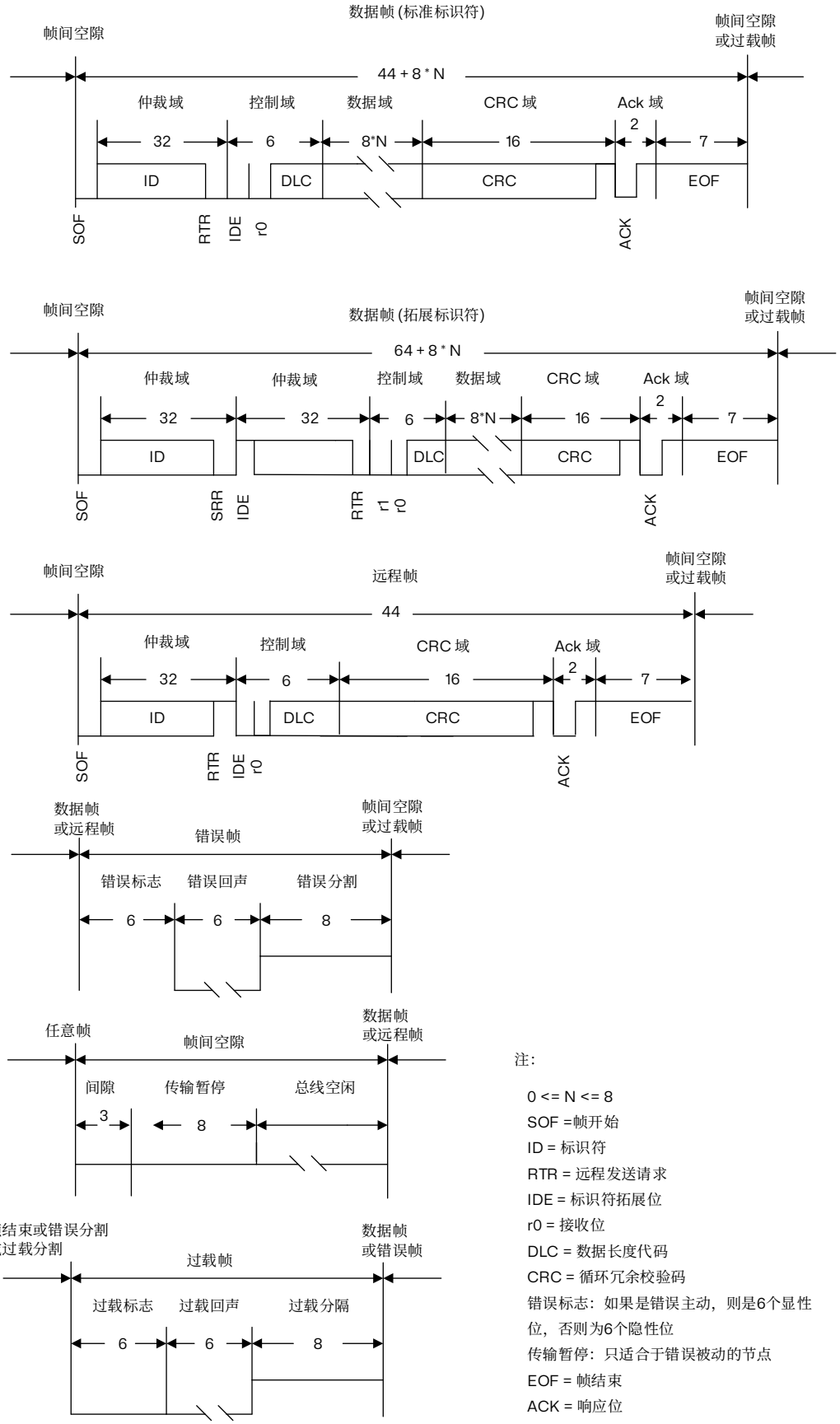
$t_q$ = (BRP[9:0]+1) x $t_{CANCLK}$

这里 $t_q$ 表示1个时间单元

$t_{CANCLK}$ = CAN时钟的时间周期

BRP[9:0]，TS1[3:0] 和TS2[2:0] 在CAN_BTR寄存器中定义

图 31.13 位时序




注：

0 <= N <= 8
SOF =帧开始
ID = 标识符
RTR = 远程发送请求
IDE = 标识符拓展位
r0 = 接收位
DLC = 数据长度代码
CRC = 循环冗余校验码
错误标志：如果是错误主动，则是6个显性位，否则为6个隐性位
传输暂停：只适合于错误被动的节点
EOF = 帧结束
ACK = 响应位

图 31.14 各种 CAN 帧

## 31.8 bxCAN 中断

bxCAN 占用 4 个专用的中断向量。通过设置 CAN 中断允许寄存器（CAN_IER），每个中断源都可以单独允许和禁用。

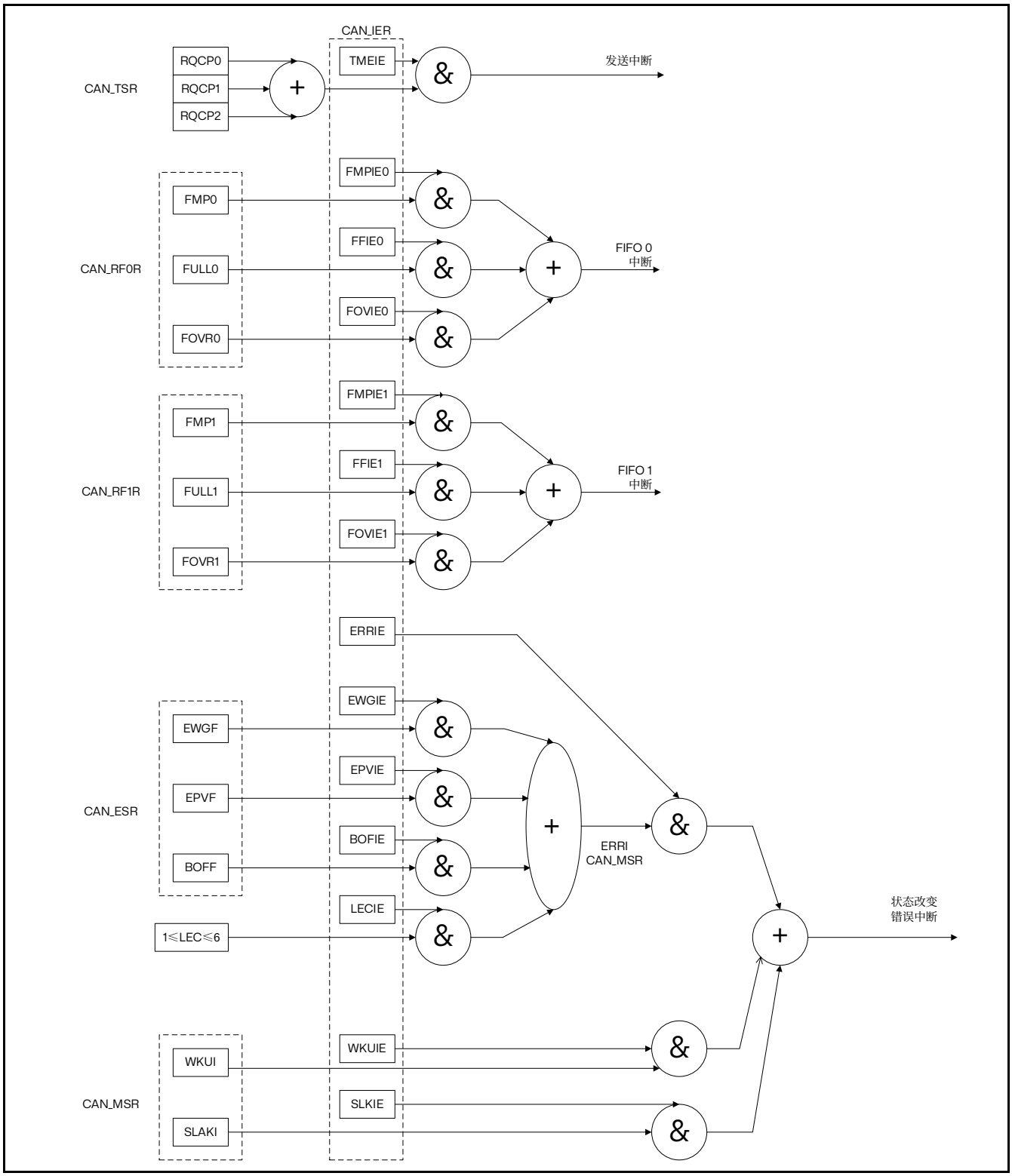



图 31.15 事件标志和中断产生

- **发送中断**可由下列事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位被置‘1’。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位被置‘1’。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位被置‘1’。
- **FIFO0 中断**可由下列事件产生：
  - FIFO0 接收到一个新报文，CAN_RF0R 寄存器的 FMP0 位不再是‘00’。
  - FIFO0 变为满的情况，CAN_RF0R 寄存器的 FULL0 位被置‘1’。
  - FIFO0 发生溢出的情况，CAN_RF0R 寄存器的 FOVR0 位被置‘1’。
- **FIFO1 中断**可由下列事件产生：
  - FIFO1 接收到一个新报文，CAN_RF1R 寄存器的 FMP1 位不再是‘00’。
  - FIFO1 变为满的情况，CAN_RF1R 寄存器的 FULL1 位被置‘1’。
  - FIFO1 发生溢出的情况，CAN_RF1R 寄存器的 FOVR1 位被置‘1’。
- **错误和状态变化中断**可由下列事件产生：
  - 出错情况，关于出错情况的详细信息请参考 CAN 错误状态寄存器（CAN_ESR）。
  - 唤醒情况，在 CAN 接收引脚上监视到帧起始位（SOF）。
  - CAN 进入睡眠模式。

## 31.9 寄存器访问保护

对某些寄存器的错误访问会导致一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，软件只能在 CAN 处于初始化模式时修改 CAN_BTR 寄存器。

虽然错误数据的发送对 CAN 网的网络层不会带来问题，但却会对应用程序造成严重影响。因此，软件只能在发送邮箱为空的状态改变它，请参考图 32.7

过滤器的数值只能在关闭对应过滤器组的状态下，或设置 FINIT 位为‘1’后才能修改。此外，只有在设置整个过滤器为初始化模式下（即 FINIT=1），才能修改过滤器的设置，即修改 CAN_FMxR，CAN_FSxR 和 CAN_FFAR 寄存器。

## 31.10 CAN 控制和状态寄存器描述

### 31.10.1 CAN 主控制寄存器（CAN_MCR）

偏移地址：0x00
复位值：0x0001 0002

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | DBF |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESET | Reserved | | | | | | | TTCM | ABOM | AWUM | NART | RFLM | TXFP | SLEEP | INRQ |
| rs | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:17 保留，必须保持为复位值。

Bit 16 **DBF**：调试冻结（Debug freeze）

0：在调试时，CAN 照常工作

1：在调试时，冻结 CAN 的接收/发送。仍然可以正常地读写和控制接收 FIFO。

Bit 15 **RESET**：bxCAN 软件复位（bxCAN software master reset）

0：本外设正常工作；

1：对 bxCAN 进行强行复位，复位后 bxCAN 进入睡眠模式（FMP 位和 CAN_MCR 寄存器被初始化为其复位值）。此后硬件自动对该位清‘0’。

Bit 14:8 保留，必须保持为复位值。

Bit 7 **TTCM**：时间触发通信模式（Time triggered communication mode）

0：禁止时间触发通信模式；

1：允许时间触发通信模式。

*注：要想了解关于时间触发通信模式的更多信息，请参考章节：时间触发通信模式。*

Bit 6 **ABOM**：自动离线（Bus-Off）管理（Automatic bus-off management）

该位决定 CAN 硬件在什么条件下可以退出离线状态。

0：离线状态的退出过程是，软件对 CAN_MCR 寄存器的 INRQ 位进行置‘1’随后清‘0’后，一旦硬件检测到 128 次 11 位连续的隐性位，则退出离线状态；

1：一旦硬件检测到 128 次 11 位连续的隐性位，则自动退出离线状态。

*注：关于离线状态的更多信息，请参考章节：出错管理。*

Bit 5 **AWUM**：自动唤醒模式（Automatic wakeup mode）

该位决定 CAN 处在睡眠模式时由硬件还是软件唤醒

0：睡眠模式通过清除 CAN_MCR 寄存器的 SLEEP 位，由软件唤醒；

1：睡眠模式通过检测 CAN 报文，由硬件自动唤醒。

唤醒的同时，硬件自动对 CAN_MSR 寄存器的 SLEEP 和 SLAK 位清‘0’。

Bit 4 **NART**：禁止报文自动重传（No automatic retransmission）

0：按照 CAN 标准，CAN 硬件在发送报文失败时会一直自动重传直到发送成功；

1：CAN 报文只被发送 1 次，不管发送的结果如何（成功、出错或仲裁丢失）。

Bit 3 **RFLM**：接收 FIFO 锁定模式（Receive FIFO locked mode）

0：在接收溢出时 FIFO 未被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会覆盖原有的报文；

1：在接收溢出时 FIFO 被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会被丢弃。

Bit 2 **TXFP**：发送 FIFO 优先级（Transmit FIFO priority）

当有多个报文同时在等待发送时，该位决定这些报文的发送顺序

0：优先级由报文的标识符来决定；

1：优先级由发送请求的顺序来决定。

Bit 1 **SLEEP**：睡眠模式请求（Sleep mode request）

软件对该位置‘1’可以请求 CAN 进入睡眠模式，一旦当前的 CAN 活动（发送或接收报文）结束，CAN 就进入睡眠。

软件对该位清‘0’使 CAN 退出睡眠模式。

当设置了 AWUM 位且在 CAN Rx 信号中检测出 SOF 位时，硬件对该位清‘0’。

在复位后该位被置‘1’，即 CAN 在复位后处于睡眠模式。

Bit 0 **INRQ**：初始化请求（Initialization request）

软件对该位清‘0’可使 CAN 从初始化模式进入正常工作模式：当 CAN 在接收引脚检测到连续的 11 个隐性位后，CAN 就达到同步，并为接收和发送数据作好准备了。为此，硬件相应地对 CAN_MSR 寄存器的 INAK 位清‘0’。

软件对该位置 1 可使 CAN 从正常工作模式进入初始化模式：一旦当前的 CAN 活动（发送或接收）结束，CAN 就进入初始化模式。相应地，硬件对 CAN_MSR 寄存器的 INAK 位置‘1’。

### 31.10.2 CAN 主状态寄存器（CAN_MSR）

偏移地址：0x04

复位值：0x0000 0C02

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | RX | SAMP | RXM | TXM | Reserved | | | SLAKI | WKUI | ERRI | SLAK | INAK |
| | | | | r | r | r | r | | | | rc_w1 | rc_w1 | rc_w1 | r | r |

Bit 31:12 保留，必须保持为复位值。

Bit 11 **RX**：CAN 接收电平（CAN Rx signal）

该位反映 CAN 接收引脚（CAN_RX）的实际电平。

Bit 10 **SAMP**：上次采样值（Last sample point）

CAN 接收引脚的上次采样值（对应于当前接收位的值）。

Bit 9 **RXM**：接收模式（Receive mode）

该位为‘1’表示 CAN 当前为接收器。

Bit 8 **TXM**：发送模式（Transmit mode）

该位为‘1’表示 CAN 当前为发送器。

Bit 7:5 保留，必须保持为复位值。

Bit 4 **SLAKI**：睡眠确认中断（Sleep acknowledge interrupt）

当 SLKIE=1，一旦 CAN 进入睡眠模式硬件就对该位置‘1’，紧接着相应的中断被触发。当设置该位为‘1’时，如果设置了 CAN_IER 寄存器中的 SLKIE 位，将产生一个状态改变中断。

软件可对该位清‘0’，当 SLAK 位被清'0'时硬件也对该位清‘0’。

*注：当*SLKIE=0，*不应该查询该位，而应该查询*SLAK *位来获知睡眠状态。*

Bit 3 **WKUI**：唤醒中断挂号（Wakeup interrupt）

当 CAN 处于睡眠状态，一旦检测到帧起始位（SOF），硬件就置该位为‘1’；并且如果 CAN_IER 寄存器的 WKUIE 位为‘1’，则产生一个状态改变中断。

该位由软件清‘0’。

Bit 2 **ERRI**：出错中断挂号（Error interrupt）

当检测到错误时，CAN_ESR 寄存器的某位被置‘1’，如果 CAN_IER 寄存器的相应中断使能位也被置‘1’时，则硬件对该位置‘1’；如果 CAN_IER 寄存器的 ERRIE 位为‘1’，则产生状态改变中断。

该位由软件清‘0’。

Bit 1 **SLAK**：睡眠模式确认

该位由硬件置‘1’，指示软件 CAN 模块正处于睡眠模式。该位是对软件请求进入睡眠模式的确认（对 CAN_MCR 寄存器的 SLEEP 位置‘1’）。

当 CAN 退出睡眠模式时硬件对该位清‘0’（需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。

*注：通过软件或硬件对*CAN_MCR *的*SLEEP *位清‘0’，将启动退出睡眠模式的过程。有关清除*SLEEP *位的详细信息，参见*CAN_MCR *寄存器的*AWUM *位的描述。*

Bit 0 **INAK**：初始化确认

该位由硬件置‘1’，指示软件 CAN 模块正处于初始化模式。该位是对软件请求进入初始化模式的确认（对 CAN_MCR 寄存器的 INRQ 位置‘1’）。

当 CAN 退出初始化模式时硬件对该位清‘0’（需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。

### 31.10.3 CAN 发送状态寄存器（CAN_TSR）

偏移地址：0x08

复位值：0x1C00 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOW2 | LOW1 | LOW0 | TME2 | TME1 | TME0 | CODE[1:0] | | ABRQ2 | Reserved | | | TERR2 | ALST2 | TXOK2 | RQCP2 |
| r | r | r | r | r | r | r | r | rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABRQ1 | Reserved | | | TERR1 | ALST1 | TXOK1 | RQCP1 | ABRQ0 | Reserved | | | TERR0 | ALST0 | TXOK0 | RQCP0 |
| rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

Bit 31 **LOW2**：邮箱 2 最低优先级标志（Lowest priority flag for mailbox 2）
当多个邮箱在等待发送报文，且邮箱 2 的优先级最低时，硬件对该位置‘1’。

Bit 30 **LOW1**：邮箱 1 最低优先级标志（Lowest priority flag for mailbox 1）
当多个邮箱在等待发送报文，且邮箱 1 的优先级最低时，硬件对该位置‘1’。

Bit 29 **LOW0**：邮箱 0 最低优先级标志（Lowest priority flag for mailbox 0）
当多个邮箱在等待发送报文，且邮箱 0 的优先级最低时，硬件对该位置‘1’。

*注：如果只有1 个邮箱在等待，则*LOW[2:0]*被清‘0’。*

Bit 28 **TME2**：发送邮箱 2 空（Transmit mailbox 2 empty）
当邮箱 2 中没有等待发送的报文时，硬件对该位置‘1’。

Bit 27 **TME1**：发送邮箱 1 空（Transmit mailbox 1 empty）
当邮箱 1 中没有等待发送的报文时，硬件对该位置‘1’。

Bit 26 **TME0**：发送邮箱 0 空（Transmit mailbox 0 empty）
当邮箱 0 中没有等待发送的报文时，硬件对该位置‘1’。

Bit 25:24 **CODE[1:0]**：邮箱号（Mailbox code）
当有至少 1 个发送邮箱为空时，这 2 位表示下一个空的发送邮箱号。
当所有的发送邮箱都为空时，这 2 位表示优先级最低的那个发送邮箱号。

Bit 23 **ABRQ2**：邮箱 2 中止发送（Abort request for mailbox 2）
软件对该位置‘1’，可以中止邮箱 2 的发送请求，当邮箱 2 的发送报文被清除时硬件对该位清‘0’。
如果邮箱 2 中没有等待发送的报文，则对该位置‘1’没有任何效果。

Bit 22:20 保留，必须保持为复位值。

Bit 19 **TERR2**：邮箱 2 发送失败（Transmission error of mailbox 2）
当邮箱 2 因为出错而导致发送失败时，对该位置‘1’。

Bit 18 **ALST2**：邮箱 2 仲裁丢失（Arbitration lost for mailbox 2）
当邮箱 2 因为仲裁丢失而导致发送失败时，对该位置‘1’。

Bit 17 **TXOK2**：邮箱 2 发送成功（Transmission OK of mailbox 2）
每次在邮箱 2 进行发送尝试后，硬件对该位进行更新：
0：上次发送尝试失败；
1：上次发送尝试成功。
当邮箱 2 的发送请求被成功完成后，硬件对该位置‘1’。参考图 32.7 发送邮箱状态。

Bit 16 **RQCP2**：邮箱 2 请求完成（Request completed mailbox 2）
当上次对邮箱 2 的请求（发送或中止）完成后，硬件对该位置‘1’。
软件对该位写‘1’可以对其清‘0’；当硬件接收到发送请求时也对该位清‘0’（CAN_TI2R 寄存器的 TXRQ 位被置‘1’）。
该位被清‘0’时，邮箱 2 的其它发送状态位（TXOK2，ALST2 和 TERR2）也被清‘0’。

Bit 15 **ABRQ1**：邮箱 1 中止发送（Abort request for mailbox 1）
软件对该位置‘1’，可以中止邮箱 1 的发送请求，当邮箱 1 的发送报文被清除时硬件对该位清‘0’。

如果邮箱 1 中没有等待发送的报文，则对该位置‘1’没有任何效果。

Bit 14:12 保留，必须保持为复位值。

Bit 11 **TERR1：**邮箱 1 发送失败（Transmission error of mailbox 1）

当邮箱 1 因为出错而导致发送失败时，对该位置‘1’。

Bit 10 **ALST1：**邮箱 1 仲裁丢失（Arbitration lost for mailbox 1）

当邮箱 1 因为仲裁丢失而导致发送失败时，对该位置‘1’。

Bit 9 **TXOK1：**邮箱 1 发送成功（Transmission OK of mailbox 1）

每次在邮箱 1 进行发送尝试后，硬件对该位进行更新：

0：上次发送尝试失败；

1：上次发送尝试成功。

当邮箱 1 的发送请求被成功完成后，硬件对该位置‘1’。参考图 32.7 发送邮箱状态。

Bit 8 **RQCP1：**邮箱 1 请求完成（Request completed mailbox 1）

当上次对邮箱 1 的请求（发送或中止）完成后，硬件对该位置‘1’。

软件对该位写‘1’可以对其清‘0’；当硬件接收到发送请求时也对该位清‘0’（CAN_TI1R 寄存器的 TXRQ 位被置‘1’）。

该位被清‘0’时，邮箱 1 的其它发送状态位（TXOK1，ALST1 和 TERR1）也被清‘0’。

Bit 7 **ABRQ0：**邮箱 0 中止发送（Abort request for mailbox 0）

软件对该位置‘1’可以中止邮箱 0 的发送请求，当邮箱 0 的发送报文被清除时硬件对该位清‘0’。

如果邮箱 0 中没有等待发送的报文，则对该位置 1 没有任何效果。

Bit 6:4 保留，必须保持为复位值。

Bit 3 **TERR0：**邮箱 0 发送失败（Transmission error of mailbox 0）

当邮箱 0 因为出错而导致发送失败时，对该位置‘1’。

Bit 2 **ALST0：**邮箱 0 仲裁丢失（Arbitration lost for mailbox 0）

当邮箱 0 因为仲裁丢失而导致发送失败时，对该位置‘1’。

Bit 1 **TXOK0：**邮箱 0 发送成功（Transmission OK of mailbox 0）

每次在邮箱 0 进行发送尝试后，硬件对该位进行更新：

0：上次发送尝试失败；

1：上次发送尝试成功。

当邮箱 0 的发送请求被成功完成后，硬件对该位置‘1’。参考图 32.7 发送邮箱状态。

Bit 0 **RQCP0：**邮箱 0 请求完成（Request completed mailbox 0）

当上次对邮箱 0 的请求（发送或中止）完成后，硬件对该位置‘1’。

软件对该位写‘1’可以对其清‘0’；当硬件接收到发送请求时也对该位清‘0’（CAN_TI0R 寄存器的 TXRQ 位被置‘1’）。

该位被清‘0’时，邮箱 0 的其它发送状态位（TXOK0，ALST0 和 TERR0）也被清‘0’。

### 31.10.4 CAN 接收 FIFO 0 寄存器（CAN_RF0R）

偏移地址：0x0C

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | RFOM0 | FOVR0 | FULL0 | Res | FMP0[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | r |

Bit 31:6 保留，必须保持为复位值。

Bit 5 **RFOM0：**释放接收 FIFO 0 输出邮箱（Release FIFO 0 output mailbox）

软件通过对该位置‘1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置‘1’没有任何效果，即只有当 FIFO 中有报文时对该位置‘1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。

当输出邮箱被释放时，硬件对该位清‘0’。

Bit 4 **FOVR0：**FIFO 0 溢出（FIFO 0 overrun）

当 FIFO 0 已满，又收到新的报文且报文符合过滤条件，硬件对该位置‘1’。

该位由软件清‘0’。

Bit 3 **FULL0：**FIFO 0 满（FIFO 0 full）

当 FIFO 0 中有 3 个报文时，硬件对该位置‘1’。

该位由软件清‘0’。

Bit 2 保留，必须保持为复位值。

Bit 1:0 **FMP0[1:0]：**FIFO 0 报文数目（FIFO 0 message pending）

FIFO 0 报文数目这 2 位反映了当前接收 FIFO 0 中存放的报文数目。

每当 1 个新的报文被存入接收 FIFO 0，硬件就对 FMP0 加 1。

每当软件对 RFOM0 位写‘1’来释放输出邮箱，FMP0 就被减 1，直到其为 0。

### 31.10.5 CAN 接收 FIFO 1 寄存器（CAN_RF1R）

偏移地址：0x10

复位值：0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | RFOM1 | FOVR1 | FULL1 | Res | FMP1[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | r |

Bit 31:6 保留，必须保持为复位值。

Bit 5 **RFOM1：**释放接收 FIFO 1 输出邮箱（Release FIFO 1 output mailbox）

软件通过对该位置‘1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置‘1’没有任何效果，即只有当 FIFO 中有报文时对该位置‘1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。

当输出邮箱被释放时，硬件对该位清‘0’。

Bit 4 **FOVR1：**FIFO 1 溢出（FIFO 1 overrun）

当 FIFO 1 已满，又收到新的报文且报文符合过滤条件，硬件对该位置‘1’。

该位由软件清‘0’。

Bit 3 **FULL1：**FIFO 1 满（FIFO 1 full）

当 FIFO 1 中有 3 个报文时，硬件对该位置‘1’。

该位由软件清‘0’。

Bit 2 保留，必须保持为复位值。

Bit 1:0 **FMP1[1:0]：**FIFO 1 报文数目（FIFO 1 message pending）

FIFO 1 报文数目这 2 位反映了当前接收 FIFO 1 中存放的报文数目。

每当 1 个新的报文被存入接收 FIFO 1，硬件就对 FMP0 加 1。

每当软件对 RFOM1 位写‘1’来释放输出邮箱，FMP1 就被减 1，直到其为 0。

### 31.10.6 CAN 中断使能寄存器（CAN_IER）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SLKIE | WKUIE |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRIE | Reserved | | | LECIE | BOFIE | EPVIE | EWGIE | Res | FOVIE1 | FFIE1 | FMPIE1 | FOVIE0 | FFIE0 | FMPIE0 | TMEIE |
| rw | | | | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

Bit 31:18 保留，必须保持为复位值。

Bit 17 **SLKIE**：睡眠中断使能（Sleep interrupt enable）

0：当 SLAKI 位被置‘1’时，不产生中断；

1：当 SLAKI 位被置‘1’时，产生中断。

Bit 16 **WKUIE**：唤醒中断使能（Wakeup interrupt enable）

0：当 WKUI 位被置‘1’时，不产生中断；

1：当 WKUI 位被置‘1’时，产生中断。

Bit 15 **ERRIE**：错误中断使能（Error interrupt enable）

0：当 CAN_ESR 寄存器有错误挂号时，不产生中断；

1：当 CAN_ESR 寄存器有错误挂号时，产生中断。

Bit 14:12 保留，必须保持为复位值。

Bit 11 **LECIE**：上次错误号中断使能（Last error code interrupt enable）

0：当检测到错误，硬件设置 LEC[2:0]时，不设置 ERRI 位；

1：当检测到错误，硬件设置 LEC[2:0]时，设置 ERRI 位为‘1’。

Bit 10 **BOFIE**：离线中断使能（Bus-off interrupt enable）

0：当 BOFF 位被置‘1’时，不设置 ERRI 位；

1：当 BOFF 位被置‘1’时，设置 ERRI 位为‘1’。

Bit 9 **EPVIE**：错误被动中断使能（Error Passive Interrupt Enable）

0：当 EPVF 位被置‘1’时，不设置 ERRI 位；

1：当 EPVF 位被置‘1’时，设置 ERRI 位为‘1’。

Bit 8 **EWGIE**：错误警告中断使能（Error warning interrupt enable)

0：当 EWGF 位被置‘1’时，不设置 ERRI 位；

1：当 EWGF 位被置‘1’时，设置 ERRI 位为‘1’。

Bit 7 保留，必须保持为复位值。

Bit 6 **FOVIE1：**FIFO 1 溢出中断使能（FIFO overrun interrupt enable）

0：当 FIFO 1 的 FOVR 位被置‘1’时，不产生中断；

1：当 FIFO 1 的 FOVR 位被置‘1’时，产生中断。

Bit 5 **FFIE1：**FIFO 1 满中断使能（FIFO full interrupt enable）

0：当 FIFO 1 的 FULL 位被置‘1’时，不产生中断；

1：当 FIFO 1 的 FULL 位被置‘1’时，产生中断。

Bit 4 **FMPIE1**：FIFO 1 消息挂号中断使能（FIFO message pending interrupt enable）

0：当 FIFO 1 的 FMP[1:0]位为非 0 时，不产生中断；

1：当 FIFO 1 的 FMP[1:0]位为非 0 时，产生中断。

Bit 3 **FOVIE0：**FIFO 0 溢出中断使能（FIFO overrun interrupt enable）

0：当 FIFO 0 的 FOVR 位被置‘1’时，不产生中断；

1：当 FIFO 0 的 FOVR 位被置‘1’时，产生中断。

Bit 2 **FFIE0：**FIFO 0 满中断使能（FIFO full interrupt enable）

0：当 FIFO 0 的 FULL 位被置‘1’时，不产生中断；

1：当 FIFO 0 的 FULL 位被置‘1’时，产生中断。

Bit 1 **FMPIE0：**FIFO 0 消息挂号中断使能（FIFO message pending interrupt enable）

0：当 FIFO 0 的 FMP[1:0]位为非 0 时，不产生中断；

1：当 FIFO 0 的 FMP[1:0]位为非 0 时，产生中断。

Bit 0 **TMEIE：**发送邮箱空中断使能（Transmit mailbox empty interrupt enable）

0：当 RQCPx 位被置‘1’时，不产生中断；

1：当 RQCPx 位被置‘1’时，产生中断。

*注：请参考章节：*bxCAN *中断。*

### 31.10.7 CAN 错误状态寄存器（CAN_ESR）

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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REC[7:0] | | | | | | | | TEC[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | LEC[2:1] | | | | Res | BOFF | EPVF | WEGF |
| | | | | | | | | rw | rw | rw | | | r | r | r |

Bit 31:24 **REC[7:0]：**接收错误计数器（Receive error counter）

这个计数器按照 CAN 协议的故障界定机制的接收部分实现。按照 CAN 的标准，当接收出错时，根据出错的条件，该计数器加 1 或加 8；而在每次接收成功后，该计数器减 1，或当该计数器的值大于 127 时，设置它的值为 120。当该计数器的值超过 127 时，CAN 进入错误被动状态。

Bit 23:16 **TEC[7:0]：**9 位发送错误计数器的低 8 位（Least significant byte of the 9-bit transmit error counter）

与上面相似，这个计数器按照 CAN 协议的故障界定机制的发送部分实现。

Bit 15:7 保留，必须保持为复位值。

Bit 6:4 **LEC[2:0]：**上次错误代码（Last error code）

在检测到 CAN 总线上发生错误时，硬件根据出错情况设置。当报文被正确发送或接收后，硬件清除其值为‘0’。

硬件没有使用错误代码 7，软件可以设置该值，从而可以检测代码的更新。

000：没有错误；

001：位填充错；

010：格式（Form）错；

011：确认（ACK）错；

100：隐性位错；

101：显性位错；

110：CRC 错；

111：由软件设置。

Bit 3 保留，必须保持为复位值。

Bit 2 **BOFF：**离线标志（Bus-off flag）

当进入离线状态时，硬件对该位置‘1’。当发送错误计数器 TEC 溢出，即大于 255 时，CAN 进入离线状态。请参考章节：出错管理。

Bit 1 **EPVF：**错误被动标志（Error passive flag）

当出错次数达到错误被动的阈值时，硬件对该位置’1’。

（接收错误计数器或发送错误计数器的值>127）。

Bit 0 **EWGF：**错误警告标志（Error warning flag）

当出错次数达到警告的阈值时，硬件对该位置‘1’。

（接收错误计数器或发送错误计数器的值≥96）。

### 31.10.8 CAN 位时序寄存器（CAN_BTR）

偏移地址：0x1C

复位值：0x0123 0000

*当 CAN 处于初始化模式时，该寄存器只能由软件访问。*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SILM | LBKM | Reserved | | | | SJW[1:0] | | Res | TS2[2:0] | | | TS1[3:0] | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | BRP[9:0] | | | | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **SILM**：静默模式（用于调试）（Silent mode（debug））

0：正常状态；

1：静默模式。

Bit 30 **LBKM**：环回模式（用于调试）（Loop back mode（debug））

0：禁止环回模式；

1：允许环回模式。

Bit 19:26 保留，必须保持为复位值。

Bit 25:24 **SJW[1:0]**：重新同步跳跃宽度（Resynchronization jump width）

为了重新同步，该位域定义了 CAN 硬件在每位中可以延长或缩短多少个时间单元的上限。

$t_{RJW} = t_q \times (SJW[1:0] + 1)$。

Bit 23 保留，必须保持为复位值。

Bit 22:20 **TS2[2:0]**：时间段 2（Time segment 2）

该位域定义了时间段 2 占用了多少个时间单元

$t_{BS2} = t_q \times (TS2[2:0] + 1)$。

Bit 19:16 **TS1[3:0]**：时间段 1（Time segment 1）

该位域定义了时间段 1 占用了多少个时间单元

$t_{BS1} = t_q \times (TS1[3:0] + 1)$

关于位时间特性信息，请参考章节：位时间特性。

Bit 15:10 保留，必须保持为复位值。

Bit 9:0 **BRP[9:0]**：波特率分频器（Baud rate prescaler）

该位域定义了时间单元(tq)的时间长度

$t_q = (BRP[9:0]+1) \times t_{PCLK}$

### 31.11 CAN 邮箱寄存器描述

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考章节：报文存储。

除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的 FMI 域；
- 接收邮箱是只读的；
- 发送邮箱只有在它为空时才是可写的，CAN_TSR 寄存器的相应 TME 位为‘1’，表示发送邮箱为空。

共有 3 个发送邮箱和 2 个接收邮箱。每个接收邮箱为 3 级深度的 FIFO，并且只能访问 FIFO 中最先收到的报文。每个邮箱包含 4 个寄存器。

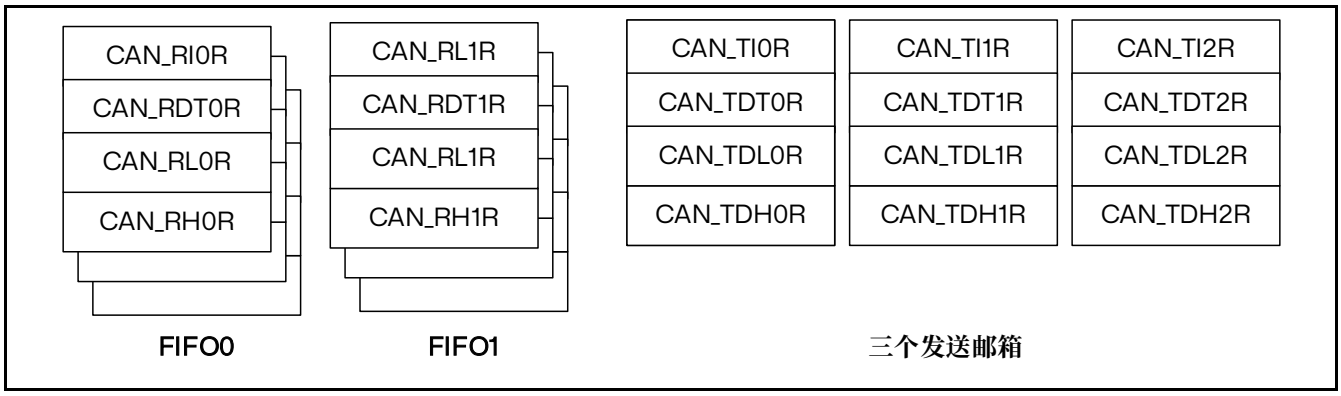



图 31.16 RX 和 TX 邮箱

### 31.11.1 发送邮箱标识符寄存器（CAN_TIxR）（x=0..2）

偏移地址：0x180，0x190，0x1A0
复位值：0xXXXX XXXX，X=未定义位（除了第 0 位，复位时 TXRQ=0）

*当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的*

*该寄存器实现了发送请求控制功能(第 0 位)—复位值为 0*

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

Bit 31:21 **STID[10:0]/EXID[28:18]：**标准标识符或扩展标识符（Standard identifier or extended identifier）

依据 IDE 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。

Bit 20:3 **EXID[17:0]：**扩展标识符（Extended identifier）

扩展身份标识的低字节。

Bit 2 **IDE：**标识符选择（Identifier extension）

该位决定发送邮箱中报文使用的标识符类型

0：使用标准标识符；

1：使用扩展标识符。

Bit 1 **RTR：**远程发送请求（Remote transmission request）

0：数据帧；

1：远程帧。

Bit 0 **TXRQ：**发送数据请求（Transmit mailbox request）

由软件对其置‘1’，来请求发送邮箱的数据。当数据发送完成，邮箱为空时，硬件对其清‘0’。

### 31.11.2 发送邮箱数据长度和时间戳寄存器（CAN_TDTxR）（x=0..2）

当邮箱不在空置状态时，该寄存器的所有位为写保护。

偏移地址：0x184，0x194，0x1A4

复位值：0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | TGT | Reserved | | | | DLC[3:0] | | | |
| | | | | | | | rw | | | | | rw | rw | rw | rw |

Bit 31:16 **TIME[15:0]：**报文时间戳（Message time stamp）

该域包含了，在发送该报文 SOF 的时刻，16 位定时器的值。

Bit 15:9 保留，必须保持为复位值。

Bit 8 **TGT：**发送时间戳（Transmit global time）

只有在 CAN 处于时间触发通信模式，即 CAN_MCR 寄存器的 TTCM 位为‘1’时，该位才有效。

0：不发送时间戳 TIME[15:0]；

1：发送时间戳 TIME[15:0]。在长度为 8 的报文中，时间戳 TIME[15:0]是最后 2 个发送的字节：TIME[7:0]作为第 7 个字节，TIME[15:8]为第 8 个字节，它们替换了写入 CAN_TDHxR[31:16]的数据（DATA6[7:0]和 DATA7[7:0]）。为了把时间戳的 2 个字节发送出去，DLC 必须编程为 8。

Bit 7:4 保留，必须保持为复位值。

Bit 3:0 **DLC[3:0]：**发送数据长度（Data length code）

该域指定了数据报文的数据长度或者远程帧请求的数据长度。1 个报文包含 0 到 8 个字节数据，而这由 DLC 决定。

### 31.11.3 发送邮箱低字节数据寄存器（CAN_TDLxR）（x=0..2）

当邮箱不在空置状态时，该寄存器的所有位为写保护。
偏移地址：0x188，0x198，0x1A8
复位值：0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | |
| 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 |
| DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:24 **DATA3[7:0]：**数据字节 3（Data byte 3）
报文的数据字节 3。

Bit 23:16 **DATA2[7:0]：**数据字节 2（Data byte 2）
报文的数据字节 2。

Bit 15:8 **DATA1[7:0]：**数据字节 1（Data byte 1）
报文的数据字节 1。

Bit 7:0 **DATA0[7:0]：**数据字节 0（Data byte 0）
报文的数据字节 0。
报文包含 0 到 8 个字节数据，且从字节 0 开始。

### 31.11.4 发送邮箱高字节数据寄存器（CAN_TDHxR）（x=0..2）

当邮箱不在空置状态时，该寄存器的所有位为写保护。
偏移地址：0x18C，0x19C，0x1AC
复位值：0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | |
| 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 |
| DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:24 **DATA7[7:0]：**数据字节 7（Data byte 7）
报文的数据字节 7。

*注：如果*CAN_MCR *寄存器的*TTCM *位为“1”，且该邮箱的*TGT *位也为“1”，那么*DATA7 *和*DATA6 *将被*TIME *时间戳代替。*

Bit 23:16 **DATA6[7:0]：**数据字节 6（Data byte 6）
报文的数据字节 6。

Bit 15:8 **DATA5[7:0]：**数据字节 5（Data byte 5）
报文的数据字节 5。

Bit 7:0 **DATA4[7:0]：**数据字节 4（Data byte 4）
报文的数据字节 4。

### 31.11.5 接收 FIFO 邮箱标识符寄存器（CAN_RIxR）（x=0..1）

偏移地址：0x1B0，0x1C0
复位值：0xXXXX XXXX

所有接收邮箱寄存器都是只读的。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:13] | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXID[12:0] | | | | | | | | | | | | | IDE | RTR | Res |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | |

Bit 31:21 **STID[10:0]/EXID[28:18]：**标准标识符或扩展标识符（Standard identifier or extended identifier）

依据 IDE 位的内容，这些位或是标准标识符，或是扩展身份标识的高字节。

Bit 20:3 **EXID[17:0]：**扩展标识符（Extended identifier）

扩展标识符的低字节。

Bit 2 **IDE：**标识符选择（Identifier extension）

该位决定接收邮箱中报文使用的标识符类型

0：使用标准标识符；

1：使用扩展标识符。

Bit 1 **RTR：**远程发送请求（Remote transmission request）

0：数据帧；

1：远程帧。

Bit 0 保留，必须保持为复位值。

### 31.11.6 接收 FIFO 邮箱数据长度和时间戳寄存器（CAN_RDTxR）（x=0..1）

偏移地址：0x1B4，0x1C4
复位值：0xXXXX XXXX

所有接收邮箱寄存器都是只读的。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME[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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMI[7:0] | | | | | | | | Reserved | | | | DLC[3:0] | | | |
| r | r | r | r | r | r | r | r | | | | | r | r | r | r |

Bit 31:16 **TIME[15:0]：**报文时间戳（Message time stamp）
该域包含了，在接收该报文 SOF 的时刻，16 位定时器的值。

Bit 15:8 **FMI[15:0]：**过滤器匹配序号（Filter match index）
这里是存在邮箱中的信息传送的过滤器序号。请参考章节：标识符过滤。

Bit 7:4 保留，必须保持为复位值。

Bit 3:0 **DLC[15:0]：**接收数据长度（Data length code）
该域表明接收数据帧的数据长度（0～8）。对于远程帧请求，数据长度 DLC 恒为 0。

### 31.11.7 接收 FIFO 邮箱低字节数据寄存器（CAN_RDLxR）（x=0..1）

当邮箱非空状态时，该寄存器的所有位均受写保护。

偏移地址：0x1B8，0x1C8
复位值：0xXXXX XXXX

所有接收邮箱寄存器都是只读的。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:24 **DATA3[7:0]：**数据字节 3（Data byte 3）
报文的数据字节 3。

Bit 23:16 **DATA2[7:0]：**数据字节 2（Data byte 2）
报文的数据字节 2。

Bit 15:8 **DATA1[7:0]：**数据字节 1（Data byte 1）
报文的数据字节 1。

Bit 7:0 **DATA0[7:0]：**数据字节 0（Data byte 0）
报文的数据字节 0。
报文包含 0 到 8 个字节数据，且从字节 0 开始。

### 31.11.8 接收 FIFO 邮箱高字节数据寄存器（CAN_RDHxR）（x=0..1）

偏移地址：0x1BC，0x1CC
复位值：0xXXXX XXXX

所有接收邮箱寄存器都是只读的。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:24 **DATA7[7:0]：**数据字节 7（Data byte 7）
报文的数据字节 7。

Bit 23:16 **DATA6[7:0]**：数据字节 6（Data byte 6）
报文的数据字节 6。

Bit 15:8 **DATA5[7:0]：**数据字节 5（Data byte 5）
报文的数据字节 5。

Bit 7:0 **DATA4[7:0]：**数据字节 4（Data byte 4）
报文的数据字节 4。

## 31.12 CAN 过滤器寄存器描述

### 31.12.1 CAN 过滤器主控寄存器（CAN_FMR）

偏移地址：0x200
复位值：0x2A1C 0E01

该寄存器的非保留位完全由软件控制。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | FINIT |
| | | | | | | | | | | | | | | | rw |

Bit 31:1 保留，必须保持为复位值。

Bit 0 **FINIT：**过滤器初始化模式（Filter init mode）

针对所有过滤器组的初始化模式设置。

0：过滤器组工作在正常模式；

1：过滤器组工作在初始化模式。

### 31.12.2 CAN 过滤器模式寄存器（CAN_FM1R）

偏移地址：0x204
复位值：0x0000 0000

只有在设置 CAN_FMR（ FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FBM13 | FBM12 | FBM11 | FBM10 | FBM9 | FBM8 | FBM7 | FBM6 | FBM5 | FBM4 | FBM3 | FBM2 | FBM1 | FBM0 |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

*注：参考图32.9 过滤器组位宽设置—寄存器组织*

Bit 31:14 保留，必须保持为复位值。

Bit 13:0 **FBMx：**过滤器模式（Filter mode）

过滤器组 x 的工作模式。

0：过滤器组 x 的 2 个 32 位寄存器工作在标识符屏蔽位模式；

1：过滤器组 x 的 2 个 32 位寄存器工作在标识符列表模式。

### 31.12.3 CAN 过滤器位宽寄存器（CAN_FS1R）

偏移地址：0x20C
复位值：0x0000 0000

只有在设置 CAN_FMR（FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FSC13 | FSC12 | FSC11 | FSC10 | FSC9 | FSC8 | FSC7 | FSC6 | FSC5 | FSC4 | FSC3 | FSC2 | FSC1 | FSC0 |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

*注：参考图32.9 过滤器组位宽设置—寄存器组织*

Bit 31:14 保留，必须保持为复位值。

Bit 13:0 **FSCx：**过滤器位宽设置（Filter scale configuration）
过滤器组 x（13～0）的位宽。
0：过滤器位宽为 2 个 16 位；
1：过滤器位宽为单个 32 位。

### 31.12.4 CAN 过滤器 FIFO 关联寄存器（CAN_FFA1R）

偏移地址：0x214
复位值：0x0000 0000

只有在设置 CAN_FMR（FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FFA13 | FFA12 | FFA11 | FFA10 | FFA9 | FFA8 | FFA7 | FFA6 | FFA5 | FFA4 | FFA3 | FFA2 | FFA1 | FFA0 |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:14 保留，必须保持为复位值。

Bit 13:0 **FFAx：**过滤器 FIFO 关联设置（Filter FIFO assignment for filter x）
报文在通过了某过滤器的过滤后，将被存放到其关联的 FIFO 中。
0：过滤器被关联到 FIFO0；
1：过滤器被关联到 FIFO1。

### 31.12.5 CAN 过滤器激活寄存器（CAN_FA1R）

偏移地址：0x21C

复位值：0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FACT13 | FACT12 | FACT11 | FACT10 | FACT9 | FACT8 | FACT7 | FACT6 | FACT5 | FACT4 | FACT3 | FACT2 | FACT1 | FACT0 |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31:14 保留，必须保持为复位值。

Bit 13:0 **FACTx：**过滤器激活（Filter active）

软件对某位设置‘1’来激活相应的过滤器。只有对 FACTx 位清‘0’，或对 CAN_FMR 寄存器的 FINIT 位设置‘1’后，才能修改相应的过滤器寄存器 x（CAN_FxR[0:1]）。

0：过滤器被禁用；

1：过滤器被激活。

### 31.12.6 CAN 过滤器组 i 的寄存器 x（CAN_FiRx）（i＝0..13；x=1..2）

偏移地址：0x240h..0x31C

复位值：0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FB31 | FB30 | FB29 | FB28 | FB27 | FB26 | FB25 | FB24 | FB23 | FB22 | FB21 | FB20 | FB19 | FB18 | FB17 | FB16 |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FB15 | FB14 | FB13 | FB12 | FB11 | FB10 | FB9 | FB8 | FB7 | FB6 | FB35 | FB4 | FB3 | FB2 | FB1 | FB0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

在所有的配置情况下：

Bit 31:0 **FB[31:0]：**过滤器位（Filter Bit）

**标识符模式**

寄存器的每位对应于所期望的标识符的相应位的电平。

0：期望相应位为显性位；

1：期望相应位为隐性位。

**屏蔽位模式**

寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。

0：不关心，该位不用于比较；

1：必须匹配，到来的标识符位必须与滤波器对应的标识符寄存器位相一致。

*注：根据过滤器位宽和模式的不同设置，过滤器组中的两个寄存器的功能也不尽相同。关于过滤器的映射，功能描述和屏蔽寄存器的关联，请参见章节：标识符过滤。*

*屏蔽位模式下的屏蔽/ 标识符寄存器，跟标识符列表模式下的寄存器位定义相同。*

# 32 串行外设接口（SPI）

## 32.1 SPI 简介

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。

---

**警告：由于 SPI1 的部分引脚与 JTAG 引脚共享（SPI1_NSS 与 JTDI ，SPI1_SCK 与 JTDO，SPI1_MISO 与 JTRST），因此这些引脚不受 IO 控制器控制，他们（在每次复位后）被默认保留为 JTAG 用途。如果用户想把引脚配置给 SPI1，必须（在调试时）关闭 JTAG 并切换至串行调试接口，或者（在标准应用时）同时关闭 JTAG 和串行调试接口。详见章节：I/O 引脚复用功能复用器和映射。**

---

## 32.2 SPI 主要特性

- 3 线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8 或 16 位传输帧格式选择
- 主或从操作
- 支持多主模式
- 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 $f_{PCLK}/2$）
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 可触发中断的主模式故障、过载
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接受请求

## 32.3 SPI 功能描述

### 32.3.1 概述

SPI 的框图参见下图。

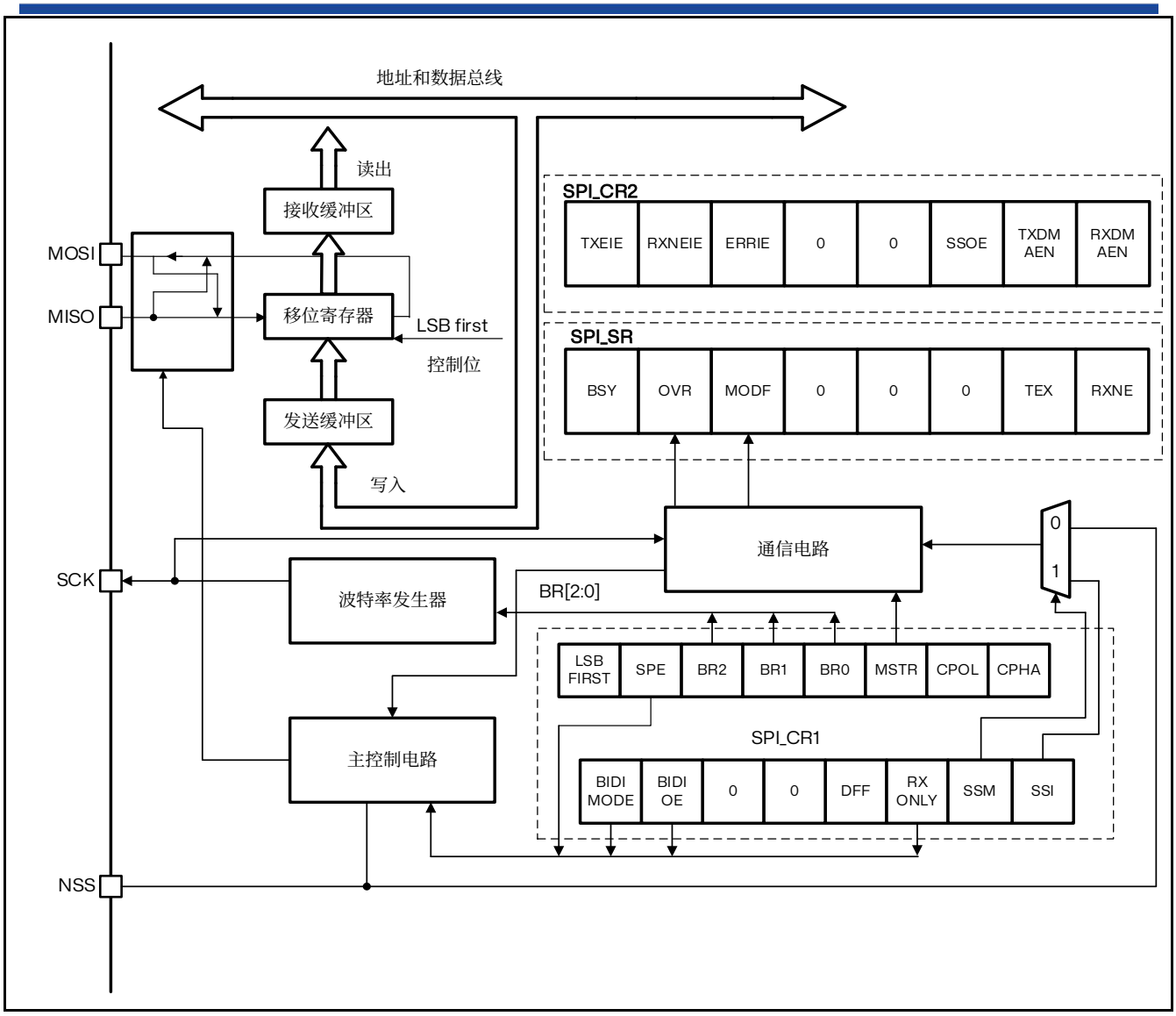
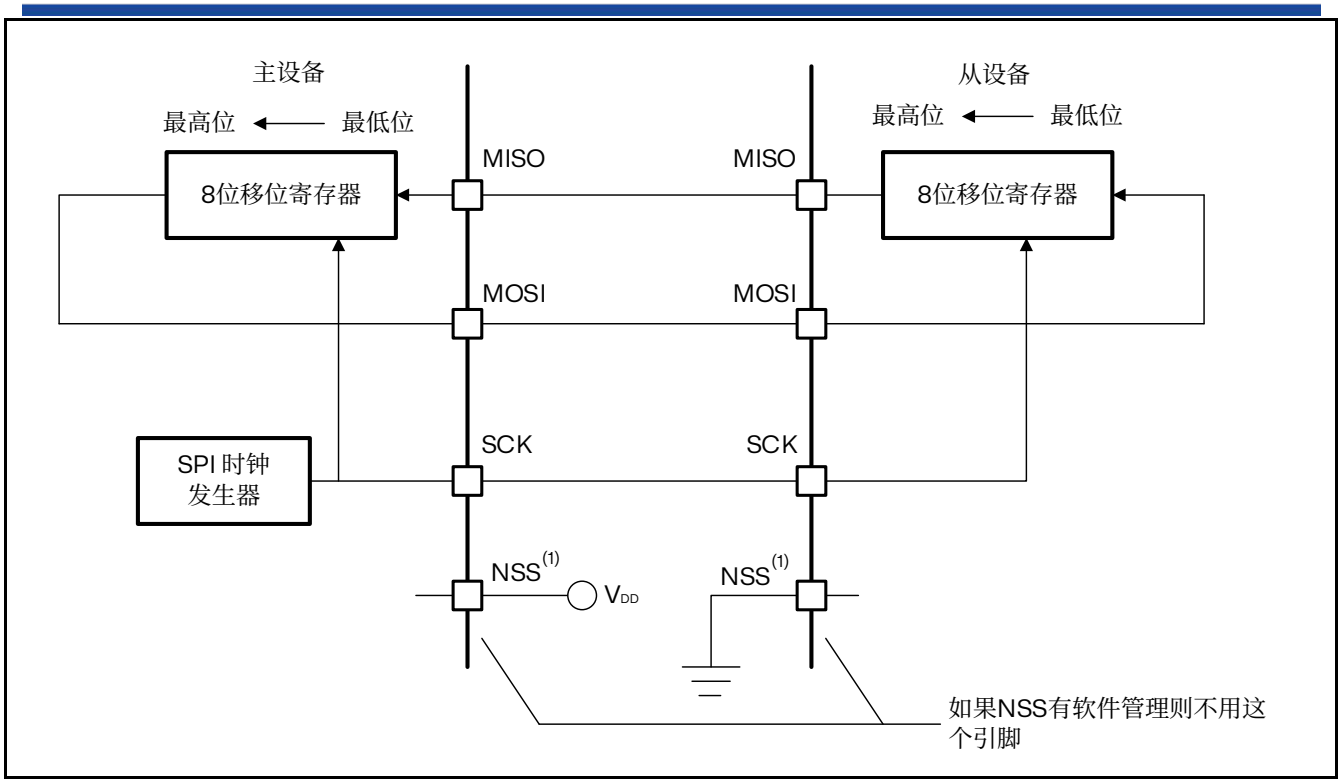



图 32.1 SPI 框图

通常 SPI 通过 4 个引脚与外部器件相连：

- MISO：主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入
- NSS：从设备选择。这是一个可选的引脚，用来选择主/从设备。它的功能是用来作为“片选引脚”，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 引脚可以由主设备的一个标准 I/O 引脚来驱动。一旦被使能（SSOE 位），NSS 引脚也可以作为输出引脚，并在 SPI 处于主模式时拉低；此时，所有的 SPI 设备，如果它们的 NSS 引脚连接到主设备的 NSS 引脚，则会检测到低电平，如果它们被设置为 NSS 硬件模式，就会自动进入从设备状态。当配置为主设备、NSS 配置为输入引脚（MSTR=1，SSOE=0）时，如果 NSS 被拉低，则这个 SPI 设备进入主模式失败状态：即 MSTR 位被自动清除，此设备进入从模式。

下图是一个单主和单从设备互连的例子。




1. 这里 NSS 引脚设置为输入

图 32.2 单主和单从应用

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输（MSB 位在前）。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

**从选择（NSS）脚管理**

可以使用 SPI_CR1 寄存器中的 SSM 位设置硬件或软件管理从器件选择。

- 软件管理 NSS（SSM = 1）
  从器件选择信息在内部由 SPI_CR1 寄存器中的 SSI 位的值驱动。外部 NSS 引脚空闲，可供其它应用使用。
- 硬件 NSS 模式，分两种情况：硬件管理 NSS（SSM = 0）
  根据 NSS 输出配置（SPI_CR1 寄存器中的 SSOE 位），硬件管理 NSS 有两种模式。
  - NSS 输出使能（SSM = 0，SSOE = 1）
    仅当器件在主模式下工作时才使用此配置。当主器件开始通信时，NSS 信号驱动为低电平，并保持到 SPI 被关闭为止。
  - NSS 输出禁止（SSM = 0，SSOE = 0）
    对于在主模式下工作的器件，此配置允许多主模式功能。对于设置为从模式的器件，NSS 引脚用作传统 NSS 输入：在 NSS 为低电平时片选该从器件，在 NSS 为高电平时取消对它的片选。

**时钟相位和时钟极性**

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（时钟极性）和 CPHA（时钟相位）位的组合用于选择数据捕获时钟边沿。

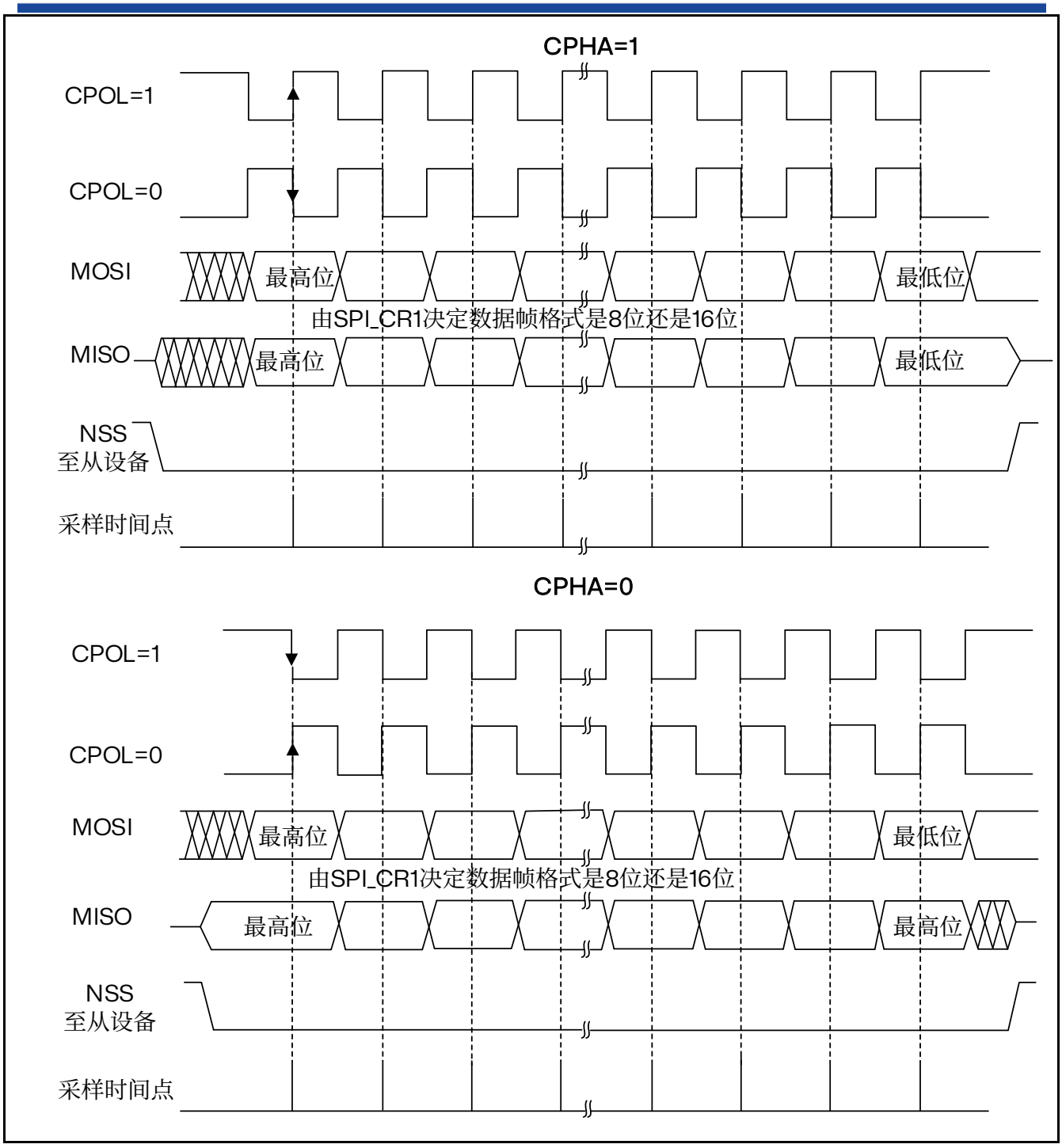
下图显示了在 CPHA 和 CPOL 位的四种组合下的 SPI 传输。可以将该图解释为主器件或从器件时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主器件和从器件之间。

*注：在切换* CPOL/CPHA *位之前，必须通过复位* SPE *位来关闭* SPI*。*

*必须以同一时序模式对主器件和从器件进行编程。*

SCK *的空闲状态必须与* SPI_CR1 *寄存器中选择的极性相对应（如果* CPOL=1*，则上拉* SCK*；如果* CPOL=0*，则下拉* SCK*）。*

*通过* SPI_CR1 *寄存器中的* DFF *位选择数据帧格式（*8 *或* 16 *位），该格式决定了发送/接收过程中的数据长度。*




1. 所示的时序为 SPI_CR1 寄存器中的 LSBFIRST 位复位时的时序。

图 32.3 数据时钟时序图

**数据帧格式**

根据 SPI_CR1 寄存器中的 LSBFIRST 位，输出数据位时可以 MSB 在先也可以 LSB 在先。

根据 SPI_CR1 寄存器的 DFF 位，每个数据帧可以是 8 位或是 16 位。所选择的数据帧格式对发送和/或接收都有效。

### 32.3.2 配置 SPI 为从模式

在从模式下，SCK 引脚用于接收从主设备来的串行时钟。SPI_CR1 寄存器中 BR[2:0]的设置不影响数据传输速率。

*注：建议在主设备发送时钟之前使能 SPI 从设备，否则可能会发生意外的数据传输。在通信时钟的第一个边沿到来之前或正在进行的通信结束之前，从设备的数据寄存器必须就绪。在使能从设备和主设备之前，通信时钟的极性必须处于稳定的数值。*

请按照以下步骤配置 SPI 为从模式：

**配置步骤**

1. 设置 DFF 位以定义数据帧格式为 8 位或 16 位。
2. 选择 CPOL 和 CPHA 位来定义数据传输和串行时钟之间的相位关系。为保证正确的数据传输，从设备和主设备的 CPOL 和 CPHA 位必须配置成相同的方式。
3. 帧格式（SPI_CR1 寄存器中的 LSBFIRST 位定义的“MSB 在前”还是“LSB 在前”）必须与主设备相同。
4. 硬件模式下（参考从选择（NSS）脚管理部分），在完整的数据帧（8 位或 16 位）传输过程中，NSS 引脚必须为低电平。在 NSS 软件模式下，设置 SPI_CR1 寄存器中的 SSM 位并清除 SSI 位。
5. 清除 MSTR 位、设置 SPE 位（SPI_CR1 寄存器），使相应引脚工作于 SPI 模式下。

在这个配置中，MOSI 引脚是数据输入，MISO 引脚是数据输出。

**数据发送过程**

在写操作中，数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始（此时第一个位被发送出去）。余下的位（对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有 15 位）被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_SP 寄存器的 TXE 标志被设置，如果设置了 SPI_CR2 寄存器的 TXEIE 位，将会产生中断。

**数据接收过程**

对于接收器，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI_SR 寄存器中的 RXNE 标志被设置。
- 如果设置了 SPI_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置“1”，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI_DR 寄存器时，SPI 设备返回这个接收缓冲器的数值。

读 SPI_DR 寄存器时，RXNE 位被清除。

### 32.3.3 配置 SPI 为主模式

在主配置时，在 SCK 脚产生串行时钟。

**配置步骤**

1. 通过 SPI_CR1 寄存器的 BR[2:0]位定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位，定义数据传输和串行时钟间的相位关系。
3. 设置 DFF 位来定义 8 位或 16 位数据帧格式。
4. 配置 SPI_CR1 寄存器的 LSBFIRST 位定义帧格式。
5. 如果需要 NSS 引脚工作在输入模式，硬件模式下，在整个数据帧传输期间应把 NSS 脚连接到高电平；在软件模式下，需设置 SPI_CR1 寄存器的 SSM 位和 SSI 位。如果 NSS 引脚工作在输出模式，则只需设置 SSOE 位。
6. 必须设置 MSTR 位和 SPE 位（只当 NSS 脚被连到高电平，这些位才能保持置位）。

在这个配置中，MOSI 引脚是数据输出，而 MISO 引脚是数据输入。

**数据发送过程**

当写入数据至发送缓冲器时，发送过程开始。

在发送第一个数据位时，数据字被并行地（通过内部总线）传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI_CR1 寄存器中的 LSBFIRST 位的设置。数据从发送缓冲器传输到移位寄存器时 TXE 标志将被置位，如果设置了 SPI_CR1 寄存器中的 TXEIE 位，将产生中断。

**数据接收过程**

对于接收器来说，当数据传输完成时：

- 传送移位寄存器里的数据到接收缓冲器，并且 RXNE 标志被置位。
- 如果设置了 SPI_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后采样时钟沿，RXNE 位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读 SPI_DR 寄存器时，SPI 设备返回接收缓冲器中的数据。

读 SPI_DR 寄存器将清除 RXNE 位。

一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。在试图写发送缓冲器之前，需确认 TXE 标志应该为“1”。

*注：在 NSS 硬件模式下，从设备的 NSS 输入由 NSS 引脚控制或另一个由软件驱动的 GPIO 引脚控制。*

### 32.3.4 配置 SPI 为单工通信

SPI 模块能够以两种配置工作于单工方式：

- 1 条时钟线和 1 条双向数据线；
- 1 条时钟线和 1 条数据线（只接收或只发送）；

**1 条时钟线和 1 条双向数据线（BIDIMODE=1）**

设置 SPI_CR1 寄存器中的 BIDIMODE 位而使能此模式。在这个模式下，SCK 引脚作为时钟，主设备使用 MOSI 引脚而从设备使用 MISO 引脚作为数据通信。传输的方向由 SPI_CR1 寄存器里的 BIDIOE 控制，当这个位是“1”的时候，数据线是输出，否则是输入。

**1 条时钟和 1 条单向数据线（BIDIMODE=0）**

在这个模式下，SPI 模块可以或者作为只发送，或者作为只接收。

- 只发送模式类似于全双工模式（BIDIMODE=0，RXONLY=0）：数据在发送引脚（主模式时是 MOSI、从模式时是 MISO）上传输，而接收引脚（主模式时是 MISO、从模式时是 MOSI）可以作为通用的 I/O 使用。此时，软件不必理会接收缓冲器中的数据（如果读出数据寄存器，它不包含任何接收数据）。
- 在只接收模式，可以通过设置 SPI_CR2 寄存器的 RXONLY 位而关闭 SPI 的输出功能；此时，发送引脚（主模式时是 MOSI、从模式时是 MISO）被释放，可以作为其它功能使用。

配置并使能 SPI 模块为只接收模式的方式是：

- 在主模式时，一旦使能 SPI，通信立即启动，当清除 SPE 位时立即停止当前的接收。在此模式下，不必读取 BSY 标志，在 SPI 通信期间这个标志始终为“1”。
- 在从模式时，只要 NSS 被拉低（或在 NSS 软件模式时，SSI 位为“0”）同时 SCK 有时钟脉冲，SPI 就一直在接收。

### 32.3.5 数据发送与接收过程

**接收与发送缓冲器**

在接收时，接收到的数据被存放在一个内部的接收缓冲器中；在发送时，在被发送之前，数据将首先被存放在一个内部的发送缓冲器中。

对 SPI_DR 寄存器的读操作，将返回接收缓冲器的内容；写入 SPI_DR 寄存器的数据将被写入发送缓冲器中。

**主模式下开始传输**

- 全双工模式（BIDIMODE=0 并且 RXONLY=0）
  - 当写入数据到 SPI_DR 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 与此同时，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器（接收缓冲器）中。
- 单向的只接收模式（BIDIMODE=0 并且 RXONLY=1）
  - SPE=1 时，传输开始；
  - 只有接收器被激活，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器（接收缓冲器）中。
- 双向模式，发送时（BIDIMODE=1 并且 BIDIOE=1）
  - 当写入数据到 SPI_DR 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 不接收数据。
- 双向模式，接收时（BIDIMODE=1 并且 BIDIOE=0）
  - SPE=1 并且 BIDIOE=0 时，传输开始；
  - 在 MOSI 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPI_DR 寄存器（接收缓冲器）中。

– 不激活发送器，没有数据被串行地送到 MOSI 引脚上。

**从模式下开始传输**

- 全双工模式（BIDIMODE=0 并且 RXONLY=0）
  – 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后的数据位依次移动进入移位寄存器；
  – 与此同时，在传输第一个数据位时，发送缓冲器中的数据被并行地传送到 8 位的移位寄存器，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据。
- 单向的只接收模式（BIDIMODE=0 并且 RXONLY=1）
  – 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后数据位依次移动进入移位寄存器；
  – 不启动发送器，没有数据被串行地传送到 MISO 引脚上。
- 双向模式，发送时（BIDIMODE=1 并且 BIDIOE=1）
  – 当从设备接收到时钟信号并且发送缓冲器中的第一个数据位被传送到 MISO 引脚上的时候，数据传输开始；
  – 在第一个数据位被传送到 MISO 引脚上的同时，发送缓冲器中要发送的数据被平行地 传送到 8 位的移位寄存器中，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据；
  – 不接收数据。
- 双向模式，接收时（BIDIMODE=1 并且 BIDIOE=0）
  – 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始；
  – 从 MISO 引脚上接收到的数据被串行地传送到 8 位的移位寄存器中，然后被平行地传送到 SPI_DR 寄存器（接收缓冲器）；
  – 不启动发送器，没有数据被串行地传送到 MISO 引脚上。

**处理数据的发送与接收**

当数据从发送缓冲器传送到移位寄存器时，设置 TXE 标志（发送缓冲器空），它表示内部的发送缓冲器可以接收下一个数据；如果在 SPI_CR2 寄存器中设置了 TXEIE 位，则此时会产生一个中断；写入 SPI_DR 寄存器即可清除 TXE 位。

*注：在写入发送缓冲器之前，软件必须确认* TXE *标志为“1”，否则新的数据会覆盖已经在发送缓冲器中的数据。*

在采样时钟的最后一个边沿，当数据被从移位寄存器传送到接收缓冲器时，设置 RXNE 标志（接收缓冲器非空）；它表示数据已经就绪，可以从 SPI_DR 寄存器读出；如果在 SPI_CR2 寄存器中设置了 RXNEIE 位，则此时会产生一个中断；读出 SPI_DR 寄存器即可清除 RXNIE 标志位。在一些配置中，传输最后一个数据时，可以使用 BSY 标志等待数据传输的结束。

**主或从模式下（BIDIMODE=0 并且 RXONLY=0）全双工发送和接收过程模式**

软件必须遵循下述过程，发送和接收数据：

1. 设置 SPE 位为“1”，使能 SPI 模块；
2. 在 SPI_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。等待 RXNE=1，然后读出 SPI_DR 寄存器并获得第

一个接收到的数据，读 SPI_DR 的同时清除了 RXNE 位。重复这些操作，发送后续的数据同时接收 n-1 个数据；

4. 等待 RXNE=1，然后接收最后一个数据；
5. 等待 TXE=1，在 BSY=0 之后关闭 SPI 模块。

也可以在响应 RXNE 或 TXE 标志的上升沿产生的中断的处理程序中实现这个过程。

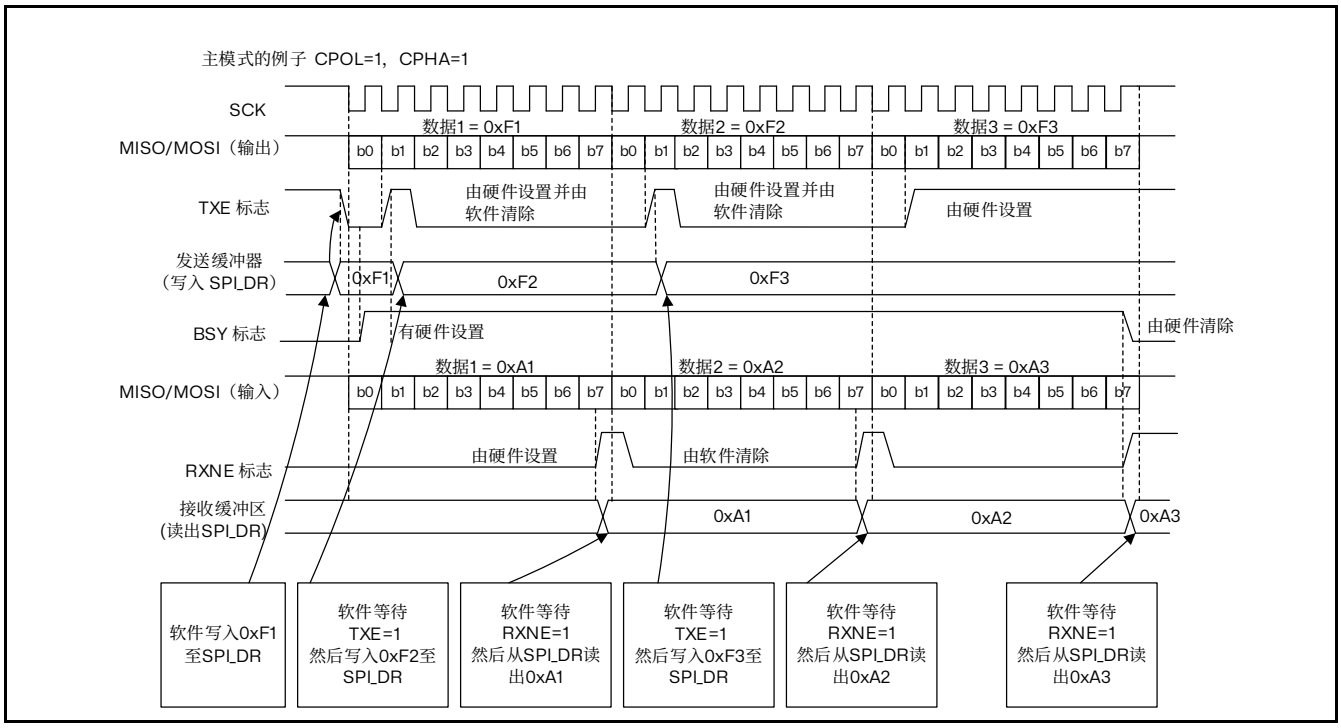



图 32.4 主/全双工模式下（BIDIMODE=0 且 RXONLY=0）连续传输时，TXE/RXNE/BSY 行为

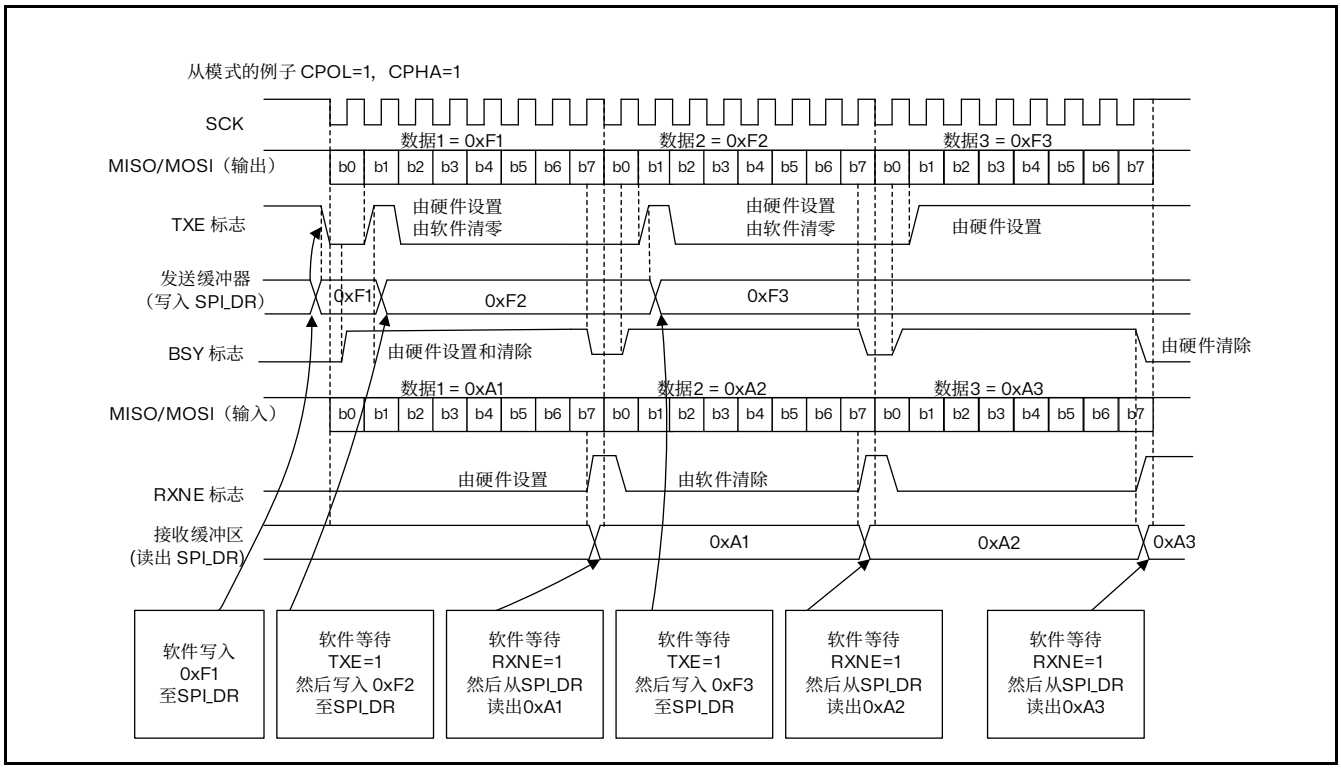



图 32.5 从/全双工模式下（BIDIMODE=0 且 RXONLY=0）连续传输时，TXE/RXNE/BSY 行为

**只发送过程（BIDIMODE=0 并且 RXONLY=0）**

在此模式下，传输过程可以简要说明如下，使用 BSY 位等待传输的结束：

1. 设置 SPE 位为“1”，使能 SPI 模块；
2. 在 SPI_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。重复这个操作，发送后续的数据；
4. 写入最后一个数据到 SPI_DR 寄存器之后，等待 TXE=1；然后等待 BSY=0，这表示最后一个数据的传输已经完成。

也可以在响应 TXE 标志的上升沿产生的中断的处理程序中实现这个过程。

*注：对于不连续的传输，在写入 SPI_DR 寄存器的操作与设置 BSY 位之间有 2 个 APB 时钟周期的延迟，因此在只发送模式下，写入最后一个数据后，最好先等待 TXE=1，然后再等待 BSY=0。*

*只发送模式下，在传输 2 个数据之后，由于不会读出接收到的数据，SPI_SR 寄存器中的 OVR 位会变为“1”。（注：软件不必理会这个 OVR 标志位）*

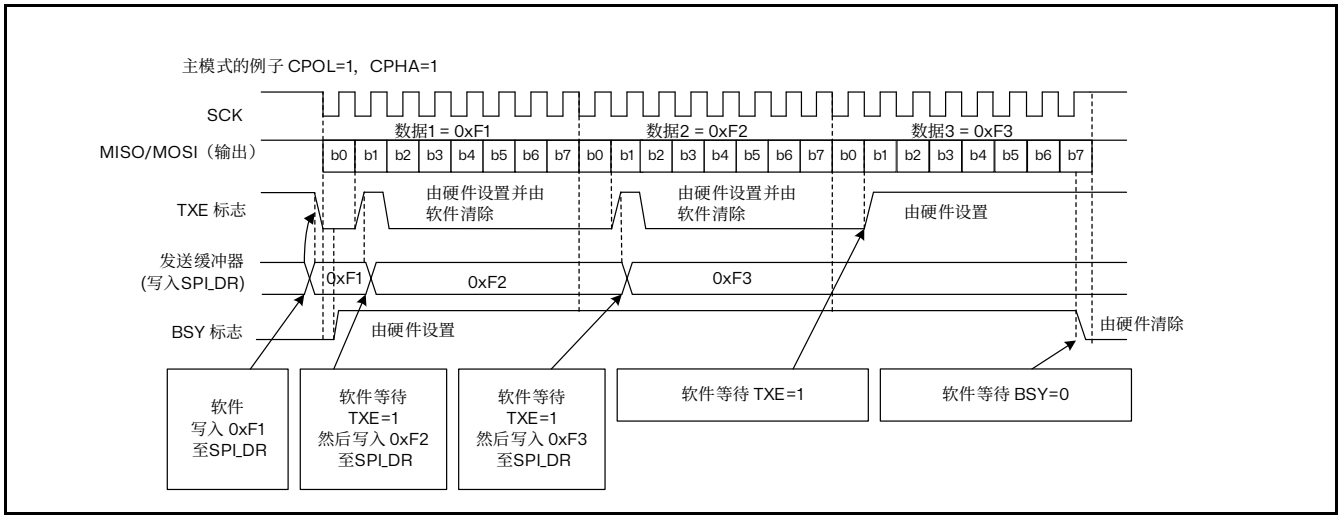



图 32.6 主设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 行为

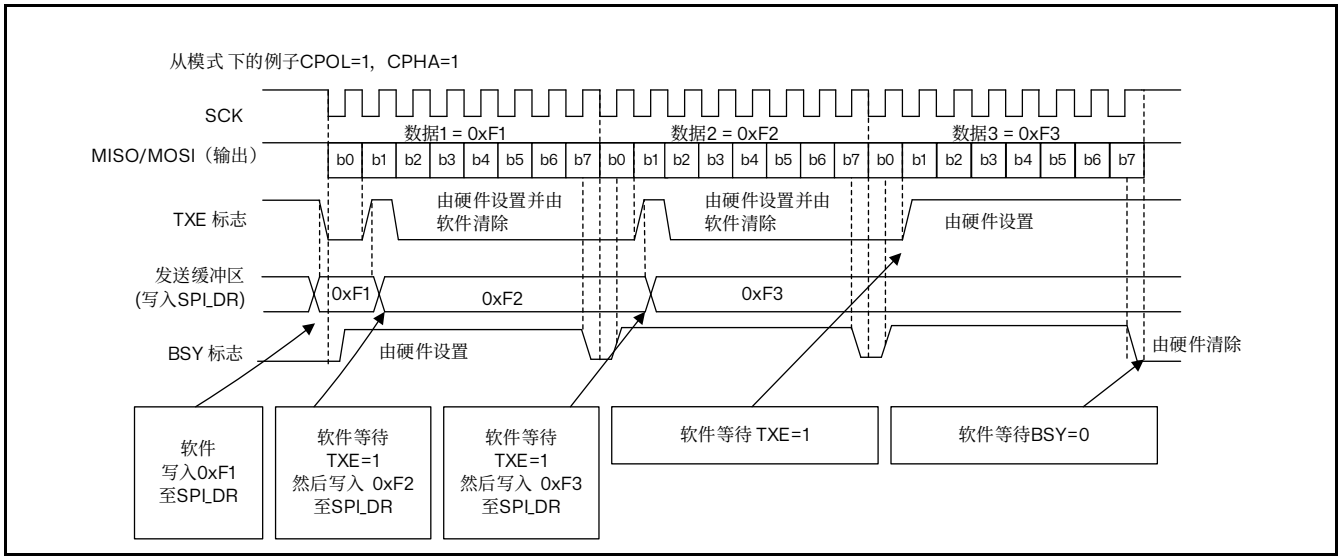



图 32.7 从设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 行为

**双向发送过程（BIDIMODE=1 并且 BIDIOE=1）**

在此模式下，操作过程类似于只发送模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CR1 寄存器中同时设置 BIDIMODE 和 BIDIOE 位为“1”。

**单向只接收模式（BIDIMODE=0 并且 RXONLY=1）**

在此模式下，传输过程可以简要说明如下：

1. 在 SPI_CR1 寄存器中，设置 RXONLY=1；
2. 设置 SPE=1，使能 SPI 模块：
   a）主模式下，立刻产生 SCK 时钟信号，在关闭 SPI（SPE=0）之前，不断地接收串行数据；
   b）从模式下，当 SPI 主设备拉低 NSS 信号并产生 SCK 时钟时，接收串行数据。
3. 等待 RXNE=1，然后读出 SPI_DR 寄存器以获得收到的数据（同时会清除 RXNE 位）。重复这个操作接收所有数据。

也可以在响应 RXNE 标志的上升沿产生的中断的处理程序中实现这个过程。

注：如果在最后一个数据传输结束后关闭 SPI 模块，请按照章节：关闭 SPI 中的建议操作。

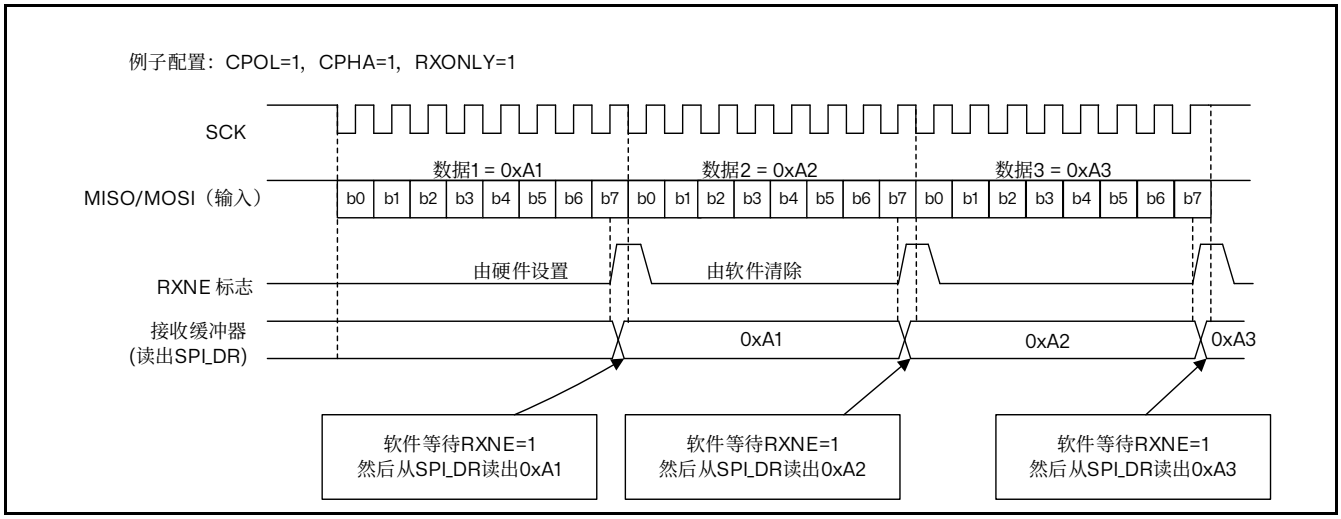



图 32.8 只接收模式（BIDIMODE=0 并且 RXONLY=1）下连续传输时，RXNE 行为

**单向接收过程（BIDIMODE=1 并且 BIDIOE=0）**

在此模式下，操作过程类似于只接收模式，不同的是：在使能 SPI 模块之前，需要在 SPI_CR1 寄存器中设置 BIDIMODE 为“1”并清除 BIDIOE 位为“0”。

**连续和非连续传输**

当在主模式下发送数据时，如果软件足够快，能够在检测到每次 TXE 的上升沿（或 TXE 中断），并立即在正在进行的传输结束之前写入 SPI_DR 寄存器，则能够实现连续的通信；此时，在每个数据项的传输之间的 SPI 时钟保持连续，同时 BSY 位不会被清除。

如果软件不够快，则会导致不连续的通信；这时，在每个数据传输之间会被清除（见下图）。

在主模式的只接收模式下（RXONLY=1），通信总是连续的，而且 BSY 标志始终为“1”。

在从模式下，通信的连续性由 SPI 主设备决定。不管怎样，即使通信是连续的，BSY 标志会在每个数据项之间至少有一个 SPI 时钟周期为低。

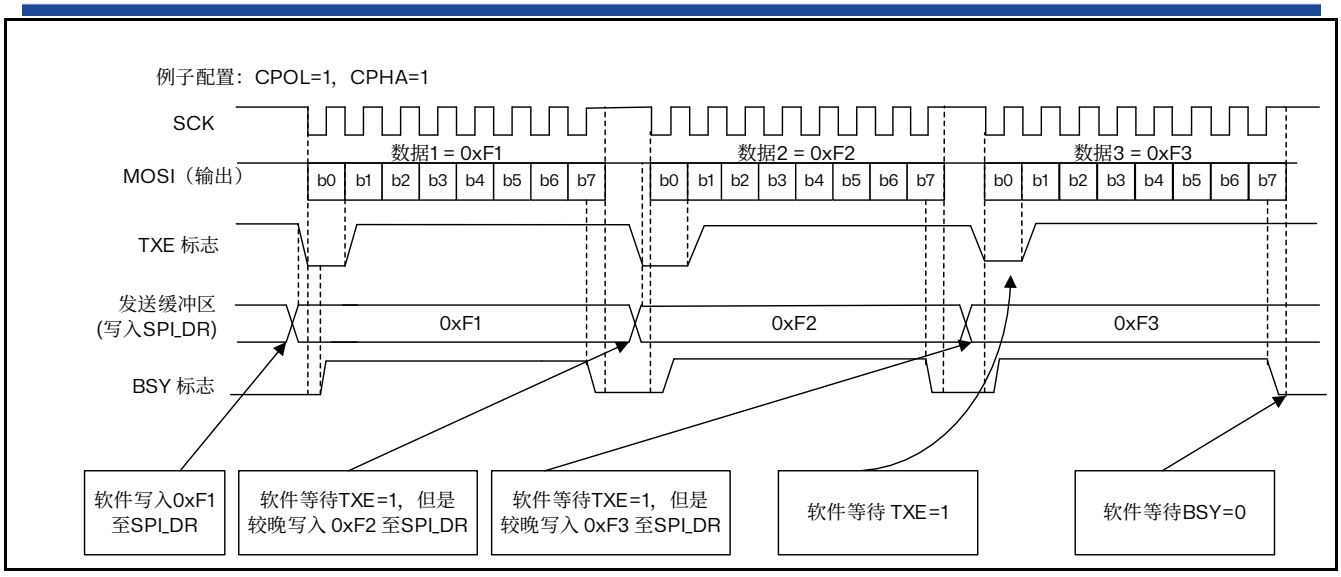



图 32.9 非连续传输发送（BIDIMODE=0 并且 RXONLY=0）时，TXE/BSY 行为

### 32.3.6 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

**发送缓冲器空闲标志（TXE）**

此标志为“1”时表明发送缓冲器为空，可以写下一个待发送的数据进入缓冲器中。当写入 SPI_DR 时，TXE 标志被清除。

**接收缓冲器非空（RXNE）**

此标志为“1”时表明在接收缓冲器中包含有效的接收数据。读 SPI 数据寄存器可以清除此标志。

**忙（Busy）标志**

BSY 标志由硬件设置与清除（写入此位无效果），此标志表明 SPI 通信层的状态。

当它被设置为“1”时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下（MSTR=1、BDM=1 并且 BDOE=0），在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式（或关闭设备时钟）之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主系统中避免写冲突。

除了主模式的双向接收模式（MSTR=1、BDM=1 并且 BDOE=0），当传输开始时，BSY 标志被置“1”。

以下情况时此标志将被清除为“0”：

- 当传输结束（主模式下，如果是连续通信的情况例外）；
- 当关闭 SPI 模块；
- 当产生主模式失效（MODF=1）。

如果通信不是连续的，则在每个数据项的传输之间，BSY 标志为低。

当通信是连续时：

- 主模式下：在整个传输过程中，BSY 标志保持为高；
- 从模式下：在每个数据项的传输之间，BSY 标志在一个 SPI 时钟周期中为低。

*注：不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TXE 和 RXNE 标志。*

### 32.3.7 关闭 SPI

当通讯结束，可以通过关闭 SPI 模块来终止通讯。清除 SPE 位即可关闭 SPI。

在某些配置下，如果再传输还未完成时，就关闭 SPI 模块并进入停机模式，则可能导致当前的传输被破坏，而且 BSY 标志也变得不可信。

为了避免发生这种情况，关闭 SPI 模块时，建议按照下述步骤操作：

**在主或从模式下的全双工模式（BIDIMODE=0，RXONLY=0）**

1. 等待 RXNE=1 并接收最后一个数据；
2. 等待 TXE=1；
3. 等待 BSY=0；
4. 关闭 SPI（SPE=0），最后进入停机模式（或关闭该模块的时钟）。

**在主或从模式下的单向只发送模式（BIDIMODE=0，RXONLY=0）或双向的发送模式（BIDIMODE=1，BIDIOE=1）**

在 SPI_DR 寄存器中写入最后一个数据后：

1. 等待 TXE=1；
2. 等待 BSY=0；
3. 关闭 SPI（SPE=0），最后进入停机模式（或关闭该模块的时钟）。

**在主或从模式下的单向只接收模式（MSTR=1，BIDIMODE=0，RXONLY=1）或双向的接收模式（MSTR=1，BIDIMODE=1，BIDIOE=0）**

这种情况需要特别地处理，以保证 SPI 不会开始一次新的传输：

1. 等待倒数第二个（第 n-1 个）RXNE=1；
2. 在关闭 SPI（SPE=0）之前等待一个 SPI 时钟周期（使用软件延迟）；
3. 在进入停机模式（或关闭该模块的时钟）之前等待最后一个 RXNE=1。

*注：在主模式下的单向只发送模式（*MSTR=1，BDM=1，BDOE=0*）时，传输过程中* BSY *标志始终为低。*

**在从模式下的只接收模式（MSTR=0，BIDIMODE=0，RXONLY=1）或双向的接收模式（MSTR=0，BIDIMODE=1，BIDIOE=0）**

1. 可以在任何时候关闭 SPI（SPE=0），SPI 会在当前的传输结束后被关闭；
2. 如果希望进入停机模式，在进入停机模式（或关闭该模块的时钟）之前必须首先等待 BSY=0。

### 32.3.8 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI_CR2 寄存器上的对应使能位被设置时，SPI 模块可以发出 DMA 传输请求。发送缓冲器和接收缓冲器亦有各自的 DMA 请求。

- 发送时，在每次 TXE 被设置为“1”时发出 DMA 请求，DMA 控制器则写数据至 SPI_DR 寄存器，TXE 标志因此而被清除。
- 接收时，在每次 RXNE 被设置为“1”时发出 DMA 请求，DMA 控制器则从 SPI_DR 寄存器读出数据，RXNE 标志因此而被清除。

当只使用 SPI 发送数据时，只需使能 SPI 的发送 DMA 通道。此时，因为没有读取收到的数据，OVR

被置为“1”（注：软件不必理会这个标志）。

当只使用 SPI 接收数据时，只需使能 SPI 的接收 DMA 通道。

在发送模式下，当 DMA 已经传输了所有要发送的数据（DMA_ISR 寄存器的 TCIF 标志变为“1”）后，可以通过监视 BSY 标志以确认 SPI 通信结束，这样可以避免在关闭 SPI 或进入停止模式时，破坏最后一个数据的传输。因此软件需要先等待 TXE=1，然后等待 BSY=0。

*注：在不连续的通信中，在写数据到SPI_DR 的操作与BSY 位被置为“1”之间，有2 个APB 时钟周期的延迟，因此，在写完最后一个数据后需要先等待TXE=1 再等待BSY=0。*

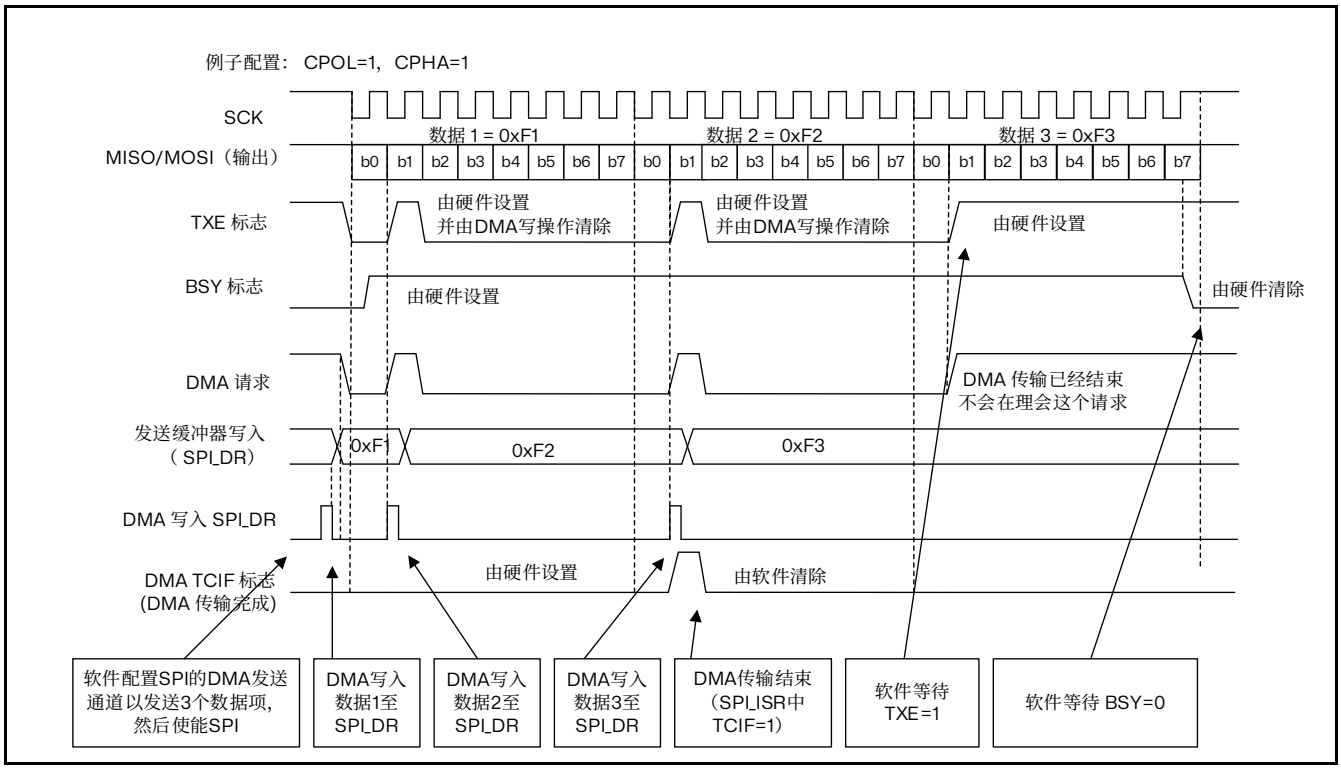



图 32.10 使用 DMA 发送

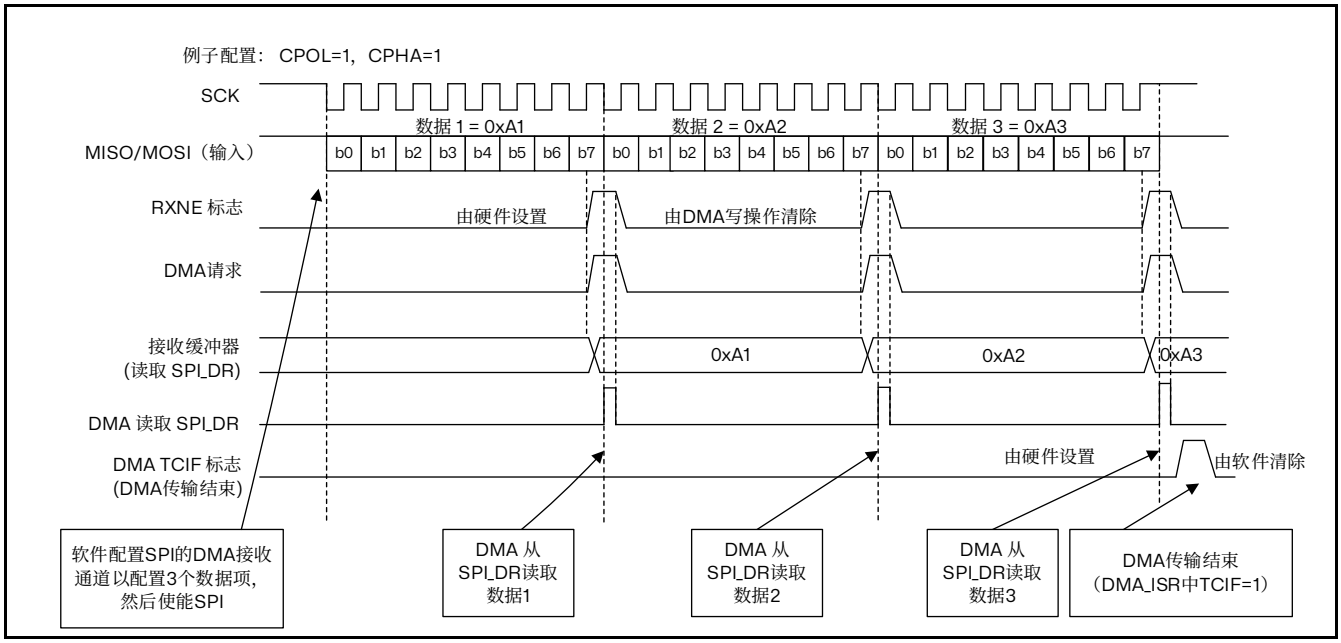



图 32.11 使用 DMA 接收

### 32.3.9 错误标志

**主模式失效错误（MODF）**

主模式失效仅发生在：NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，SSI 位被置为“0”时；MODF 位被自动置位。主模式失效对 SPI 设备有以下影响：

- MODF 位被置为“1”，如果设置了 ERRIE 位，则产生 SPI 中断；
- 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。

通常配置下，从设备的 MODF 位不能被置为“1”。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从设备模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

**溢出错误**

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的 RXNE 时，即为溢出错误。当产生溢出错误时：

- OVR 位被置为“1”；当设置了 ERRIE 位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读 SPI_DR 寄存器返回的是之前未读的数据，所有随后传送的数据都被丢弃。

依次读出 SPI_DR 寄存器和 SPI_SR 寄存器可将 OVR 清除。

### 32.3.10 SPI 中断

表 32.1 SPI 中断请求

| 中断事件 | 事件标志 | 使能控制位 |
|---|---|---|
| 发送缓冲器空标志 | TXE | TXEIE |
| 接收缓冲器非空标志 | RXNE | RXNEIE |
| 主模式失效事件 | MODF | ERRIE |
| 溢出错误 | OVR | |

## 32.4 SPI 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。
可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

### 32.4.1 SPI 控制寄存器 1（SPI_CR1）

地址偏移：0x00
复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BIDI MODE | BIDIOE | Reserved | | DFF | RX ONLY | SSM | SSI | LSB FIRST | SPE | BR[2:0] | | | MSTR | CPOL | CPHA |
| rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15 **BIDIMODE**：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。

Bit 14 **BIDIOE**：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）。

*注：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。*

Bit 13:12 保留，必须保持为复位值。

Bit 11 **DFF**：数据帧格式（Data frame format）

0：使用 8 位数据帧格式进行发送/接收；
1：使用 16 位数据帧格式进行发送/接收。

*注：只有当*SPI *禁止（*SPE=0*）时，才能写该位，否则出错。*

Bit 10 **RXONLY**：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）。

Bit 9 **SSM**：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：使能软件从设备管理。

Bit 8 **SSI**：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。

Bit 7 **LSBFIRST**：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。*

Bit 6 **SPE**：SPI 使能（SPI enable）

0：禁止 SPI 设备；

1：开启 SPI 设备。

*注：当关闭SPI 设备时，请按照章节：关闭SPI 中描述的过程操作。*

Bit 5:3 **BR[2:0]**：波特率控制（Baud rate control）

000：$f_{PCLK}/2$

001：$f_{PCLK}/4$

010：$f_{PCLK}/8$

011：$f_{PCLK}/16$

100：$f_{PCLK}/32$

101：$f_{PCLK}/64$

110：$f_{PCLK}/128$

111：$f_{PCLK}/256$

*注：当通信正在进行的时候，不能修改这些位。*

Bit 2 **MSTR**：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。*

Bit 1 **CPOL**：时钟极性（Clock polarity）

0：空闲状态时，SCK 保持低电平；

1：空闲状态时，SCK 保持高电平。

*注：当通信正在进行的时候，不能修改该位。*

Bit 0 **CPHA**：时钟相位（Clock phase）

0：数据采样从第一个时钟边沿开始；

1：数据采样从第二个时钟边沿开始。

*注：当通信正在进行的时候，不能修改该位。*

### 32.4.2 SPI 控制寄存器 2（SPI_CR2）

地址偏移：0x04

复位值：0x0000 0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TXEIE | RXNEIE | ERRIE | Reserved | | SSOE | TXDMAEN | RXDMAEN |
| | | | | | | | | rw | rw | rw | | | rw | rw | rw |

Bit 15:8 保留，必须保持为复位值。

Bit 7 **TXEIE**：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。

Bit 6 **RXNEIE**：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。

Bit 5 **ERRIE**：错误中断使能（Error interrupt enable）

当错误（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。

Bit 4:3 保留，必须保持为复位值。

Bit 2 **SSOE**：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。

Bit 1 **TXDMAEN**：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。

Bit 0 **RXDMAEN**：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。

### 32.4.3 SPI 状态寄存器（SPI_SR）

地址偏移：0x08

复位值：0x0000 0002

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BSY | OVR | MODF | Reserved | | | TXE | RXNE |
| | | | | | | | | r | r | r | | | | r | r |

Bit 15:8 保留，必须保持为复位值。

Bit 7 **BSY**：忙标志（Busy flag）

0：SPI 不忙；

1：SPI 正忙于通信，或者发送缓冲非空。

该位由硬件置位或者复位。

*注：使用这个标志时需要特别注意，详见章节：状态标志和关闭* SPI。

Bit 6 **OVR**：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。

Bit 5 **MODF**：模式错误（Mode fault）

0：没有出现模式错误；

1：出现模式错误。

该位由硬件置位，由软件序列复位。

Bit 4:2 保留，必须保持为复位值。

Bit 1 **TXE**：发送缓冲为空（Transmit buffer empty）

0：发送缓冲非空；

1：发送缓冲为空。

Bit 0 **RXNE**：接收缓冲非空（Receive buffer not empty）

0：接收缓冲为空；

1：接收缓冲非空。

### 32.4.4 SPI 数据寄存器（SPI_DR）

地址偏移：0x0C

复位值：0x0000

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

Bit 15:0 **DR[15:0]：**数据寄存器（Data register）

待发送或者已经收到的数据

数据寄存器对应两个缓冲区：一个用于写（发送缓冲）；另外一个用于读（接收缓冲）。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。

*注：对SPI 模式的注释：*

*根据SPI_CR1 的DFF 位对数据帧格式的选择，数据的发送和接收可以是8 位或者16 位的。为保证正确的操作，需要在使能SPI 之前就确定好数据帧格式。*

*对于8 位的数据，缓冲器是8 位的，发送和接收时只会用到SPI_DR[7:0]。在接收时，SPI_DR[15:8]被强制为0。*

*对于16 位的数据，缓冲器是16 位的，发送和接收时会用到整个数据寄存器，即SPI_DR[15:0]。*

# 33 I²C 接口（I2C）

## 33.1 I²C 简介

I²C（芯片间）总线接口连接微控制器和串行 I²C 总线。它提供多主机功能，控制所有 I²C 总线特定的时序、协议、仲裁和定时。它支持标准模式（Sm，最高 100kHz）和快速模式（Fm，最高 400kHz）。

I²C 模块有多种用途，包括 CRC 码的生成和校验、SMBus（系统管理总线–System Management Bus）和 PMBus（电源管理总线–Power Management Bus）。

根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

## 33.2 I²C 主要特点

- 并行总线/I²C 总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I²C 主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I²C 从设备功能
  - 可编程的 I²C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测 7 位/10 位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度（高达 100 kHz）
  - 快速（高达 400 kHz）
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I²C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答（ACK）错误
  - 检测到错位的起始或停止条件
  - 禁止拉长时钟功能时的上溢或下溢
- 2 个中断向量
  - 1 个中断用于地址/数据通讯成功
  - 1 个中断用于错误
- 可选的拉长时钟功能
- 具单字节缓冲器的 DMA
- 可配置的 PEC（信息包错误检测）的产生或校验：
  - 发送模式中 PEC 值可以作为最后一个字节传输
  - 用于最后一个接收字节的 PEC 错误校验
- 兼容 SMBus 2.0
  - 25 ms 时钟低超时延时

- 10 ms 主设备累积时钟低扩展时间
- 25 ms 从设备累积时钟低扩展时间
- 带 ACK 控制的硬件 PEC 产生/校验
- 支持地址分辨协议（ARP）

- 兼容 PMBus

*注：不是所有产品中都包含上述所有特性。请参考相关的数据手册，确认该产品支持的 I²C 功能。*

## 33.3 I²C 功能描述

I²C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚（SDA）和时钟引脚（SCL）连接到 I²C 总线。允许连接到标准（高达 100kHz）或快速（高达 400kHz）的 I²C 总线。

### 33.3.1 模式选择

接口可以下述 4 种模式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

该模块默认地工作于从模式。接口在生成起始条件后自动地从从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

**通信流**

主模式时，I²C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I²C 接口能识别它自己的地址（7 位或 10 位）和广播呼叫地址。软件能够控制开启或禁止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）。地址只在主模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位（ACK）给发送器。参考下图。

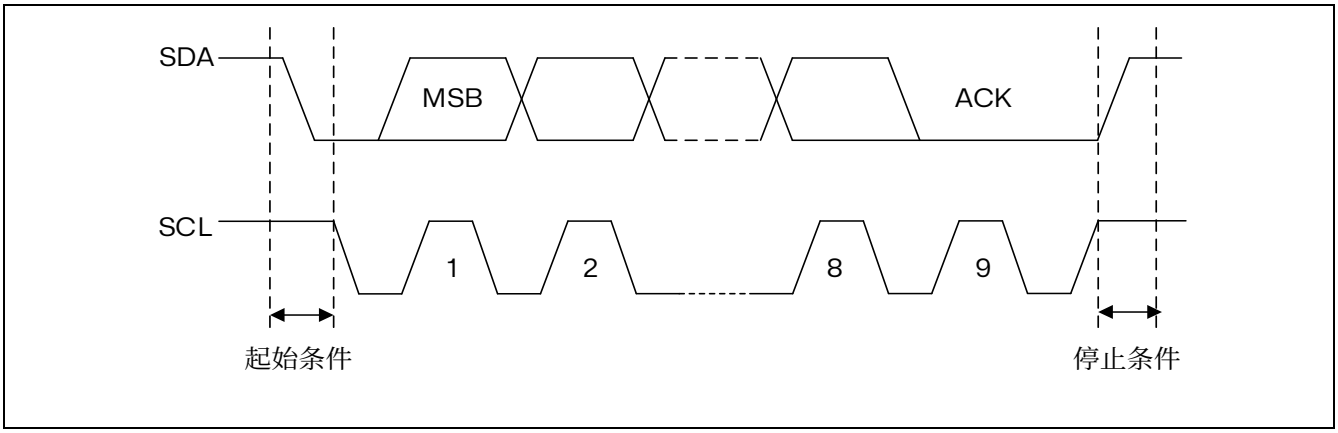
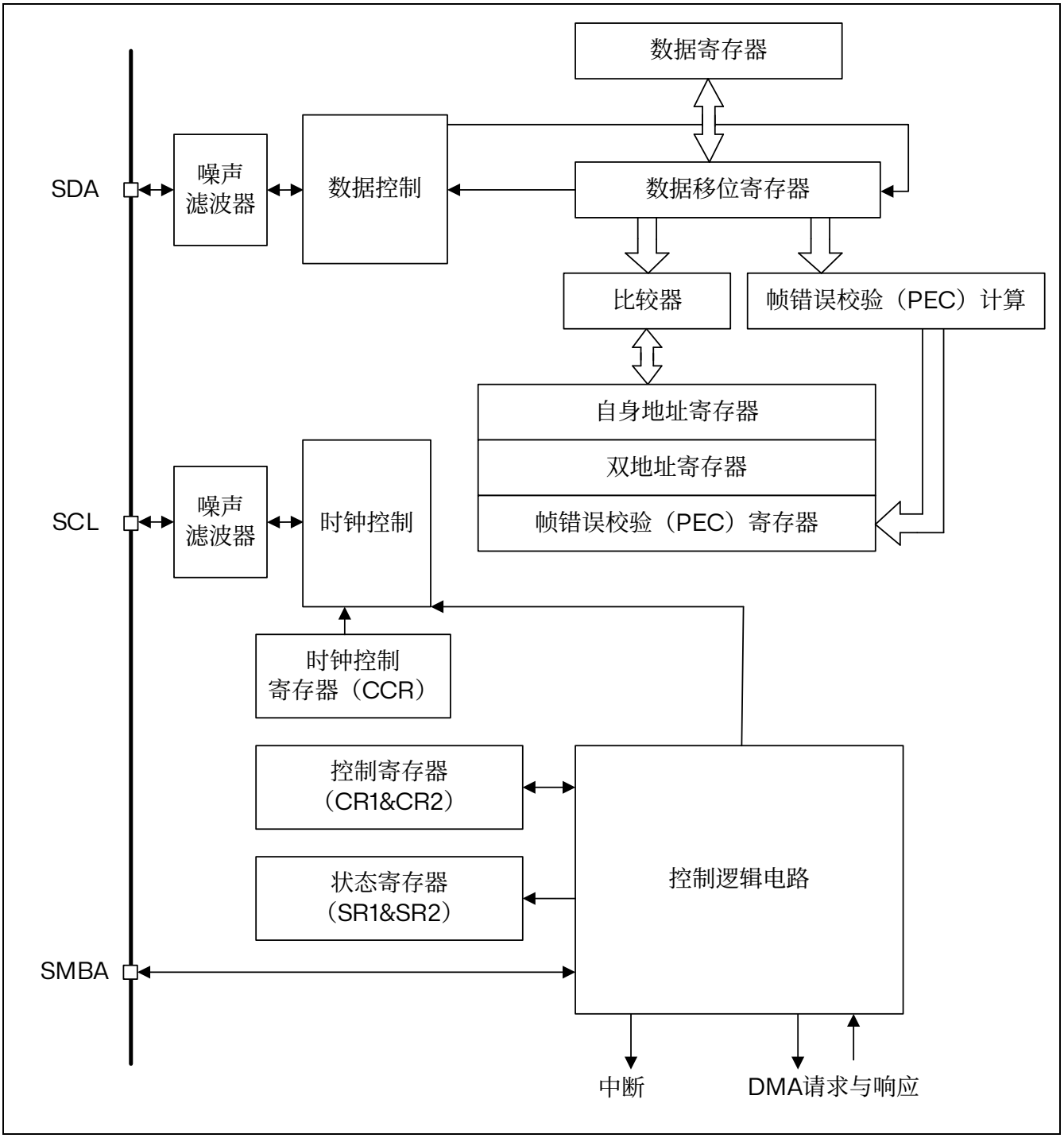



图 33.1 I²C 总线协议

软件可以开启或禁止应答（ACK），并可以设置 I²C 接口的地址（7 位、10 位地址或广播呼叫地址）。I²C 接口的功能框图示于下图。




1. 在 SMBus 模式下，SMBA 是可选信号。如果禁止了 SMBus，则不能使用该信号。

图 33.2 I²C 的功能框图

### 33.3.2 I²C 从模式

默认情况下，I²C 接口总是工作在从模式。从从模式切换到主模式，需要产生一个起始条件。

为产生正确的时序，必须在 I2C_CR2 寄存器设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址被送到移位寄存器。然后与芯片自己的地址 OAR1 和 OAR2（当 ENDUAL=1）或者广播呼叫地址（如果 ENGC=1）相比较。

*注：在 10 位地址模式时，比较包括头段序列（11110xx0），其中的 xx 是地址的两个最高有效位。*

**头段或地址不匹配：**I²C 接口将其忽略并等待另一个起始条件。

**头段匹配（仅 10 位模式）：**如果 ACK 位被置“1”，I²C 接口产生一个应答脉冲并等待 8 位从地址。

**地址匹配：**I²C 接口产生以下时序：

- 如果 ACK 被置‘1’，则产生一个应答脉冲
- 硬件设置 ADDR 位；如果设置了 ITEVFEN 位，则产生一个中断
- 如果 ENDUAL=1，软件必须读 DUALF 位，以确认响应了哪个从地址。

在 10 位模式，接收到地址序列后，从设备总是处于接收器模式。在收到与地址匹配的头序列并且最低位为“1”（即 11110xx1）后，当接收到重复的起始条件时，将进入发送器模式。

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。

#### 从发送器

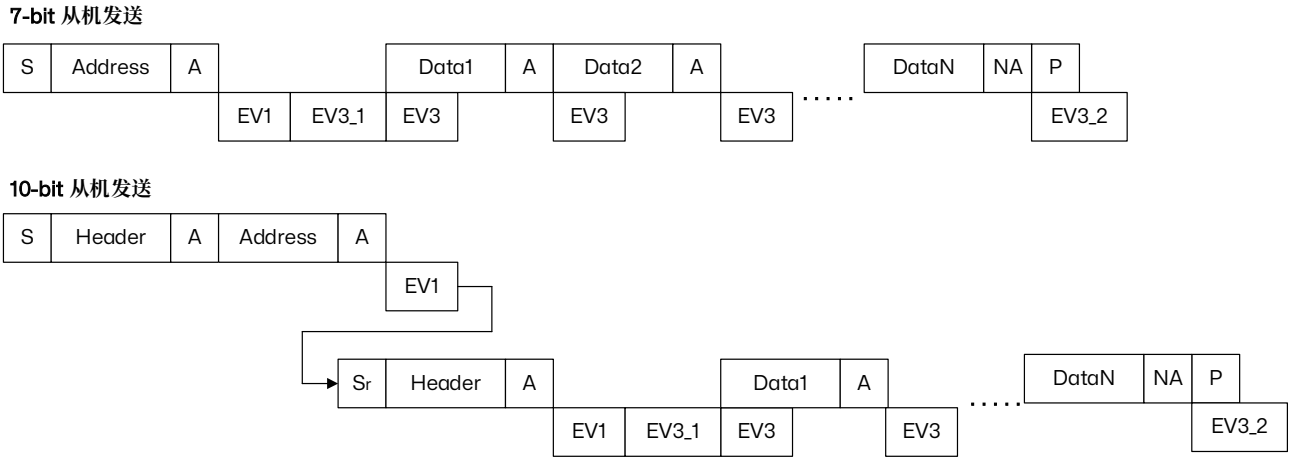
在接收到地址和清除 ADDR 位后，从发送器将字节从 DR 寄存器经由内部移位寄存器发送到 SDA 线上。

从设备保持 SCL 为低电平，直到 ADDR 位被清除并且待发送数据已写入 DR 寄存器。（见下图中的 EV1 和 EV3）。

当收到应答脉冲时：

- TxE 位被硬件置位，如果设置了 ITEVFEN 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，在清除 BTF 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器将清除 BTF 位。




**图例：** S = 开始，Sr = 重复开始，P = 停止，A = 应答，NA = 非应答，
EVx = 事件（若ITEVFEN=1，则伴随中断）

**EV1：** ADDR=1，读SR1然后读SR2将清除该事件。
**EV3_1：** TxE=1，移位寄存器空，数据寄存器空，写 DR。
**EV3：** TxE=1，移位寄存器非空，数据寄存器空，写 DR将清除该事件。
**EV3_2：** AF=1；在SR1寄存器的AF位写‘0’可清除AF位。

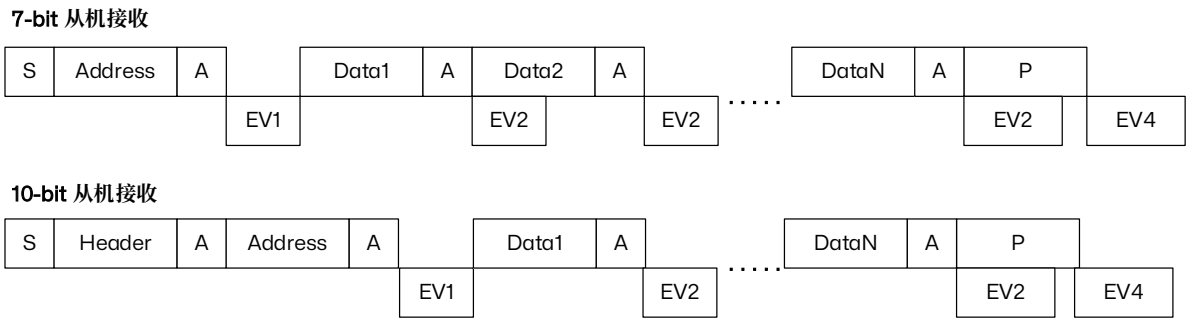
图 33.3 从发送器的传送序列图

**从接收器**

在接收到地址并清除 ADDR 后，从接收器将通过内部移位寄存器从 SDA 线接收到的字节存进 DR 寄存器。I²C 接口在接收到每个字节后都执行下列操作：

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVFEN 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，BTF 位被置位，在清除 BTF 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器将清除 BTF 位。（见下图）。




**图例：** S = 开始，Sr = 重复开始，P = 停止，A = 应答，
EVx = 事件（若ITEVFEN=1，则伴随中断）

**EV1：** ADDR=1，读SR1然后读SR2将清除该事件。
**EV2：** RxNE=1，读DR将清除该事件。
**EV4：** STOPF=1，读SR1然后写CR1寄存器将清除该事件。

图 33.4 从接收器的传送序列图

1. EV1 事件将 SCL 拉低，直到相应的软件序列结束。
2. EV2 软件序列必须在当前字节传输结束之前完成。
3. 在检查 SR1 寄存器内容后，用户应对每个已设置的标志执行完整的清除序列。
   因此，对于 ADDR 和 STOPF 标志，I2C 中断例程中需要以下序列：
   READ SR1
   if (ADDR == 1) {READ SR1; READ SR2}
   f (STOPF == 1) {READ SR1; WRITE CR1}
   目的是确保在发现 ADDR 和 STOPF 标志均被设置时，能够清除这两个标志。

**关闭从通信**

在传输完最后一个数据字节后，主设备产生一个停止条件，I²C 接口检测到这一条件时：

- 设置 STOPF=1，如果设置了 ITEVFEN 位，则产生一个中断。

然后 I²C 接口等待读 SR1 寄存器，再写 CR1 寄存器。（见上图的 EV4）。

### 33.3.3 I²C 主模式

在主模式时，I²C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。

当通过 START 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

- 在 I2C_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程 I2C_CR1 寄存器启动外设
- 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：2MHz
- 快速模式下为：4MHz

**SCL 主机时钟生成**

CCR 位用于生成 SCL 时钟的高电平和低电平，分别从上升沿和下降沿（分别）的生成开始。由于从设备可能会拉长 SCL 线，外设在上升沿生成后，会在 TRISE 位编程的时间结束时检查来自总线的 SCL 输入。

- 如果 SCL 线为低电平，表示有从设备正在拉长总线，高电平计数器停止计数，直到检测到 SCL 线为高电平。这可以确保 SCL 时钟参数的最小高电平周期。
- 如果 SCL 线为高电平，高电平计数器继续计数。

实际上，即使没有从设备拉长时钟，外设生成 SCL 上升沿到外设检测 SCL 上升沿的反馈循环也需要时间。这个反馈循环持续时间与 SCL 上升时间（影响 SCL VIH 输入检测）有关，加上 SCL 输入路径上噪声滤波器引起的延迟，以及 SCL 输入与 APB 时钟内部同步引起的延迟。反馈循环使用的最大时间在 TRISE 位中编程，以便无论 SCL 上升时间如何，SCL 频率都保持稳定。

**起始条件**

当 BUSY=0 时，设置 START=1，I²C 接口将产生一个开始条件并切换至主模式（MSL 位置位）。

*注：在主模式下，设置*START *位将在当前字节传输完后由硬件产生一个重开始条件。*

一旦发出开始条件：

- SB 位被硬件置位，如果设置了 ITEVFEN 位，则会产生一个中断。

然后主设备等待读 SR1 寄存器，紧跟着将从地址写入 DR 寄存器。（见下两张图，传输序列 EV5）。

**从地址的发送**

从地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  – ADD10 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  然后主设备等待读 SR1 寄存器，再将第二个地址字节写入 DR 寄存器（见下两张图，传输序列）。
  – 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（见下两张图，传输序列）。
- 在 7 位地址模式时，只需送出一个地址字节。
  一旦该地址字节被送出，
  – 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（见下两张图，传输序列）。

根据送出从地址的最低位（LSB），主设备决定进入发送器模式还是进入接收器模式。

- 在 7 位地址模式时，
  – 要进入发送器模式，主设备发送从地址时置最低位（LSB）为‘0’。
  – 要进入接收器模式，主设备发送从地址时置最低位（LSB）为‘1’。
- 在 10 位地址模式时
  – 要进入发送器模式，主设备先送头字节（11110xx0），然后送最低位为‘0’的从地址。（这里

xx 代表 10 位地址中的最高 2 位。）

– 要进入接收器模式，主设备先送头字节（11110xx0），然后送最低位为‘1’的从地址。然后再重新发送一个开始条件，后面跟着头字节（11110xx1）（这里 xx 代表 10 位地址中的最高 2 位。）

TRA 位指示主设备是在接收器模式还是发送器模式。

**主发送器**

在发送地址和清除 ADDR 位后，主设备通过内部移位寄存器将字节从 DR 寄存器发送到 SDA 线上。

主设备等待直到第一个数据字节被写入 I2C_DR（见下图，传输序列 EV8_1）。

当收到应答脉冲时，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 被置位并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 之前 I²C 接口将保持 SCL 为低电平；读出 I2C_SR1 之后再写入 I2C_DR 寄存器将清除 BTF 位。

**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（见下图，传输序列 EV8_2）。接口自动返回到从设备模式（MSL 位被清除）。

*注：当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8_2 事件时。*

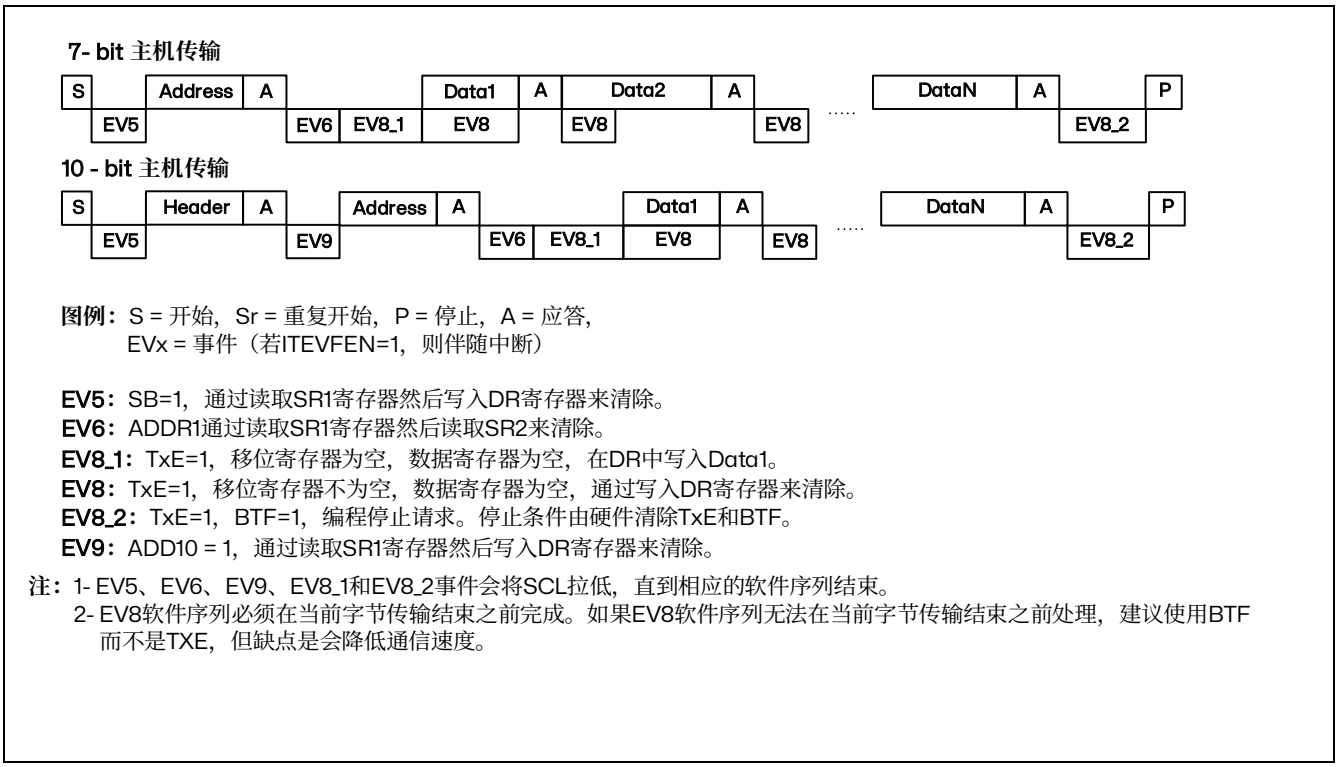



图 33.5 主发送器传送序列图

**主接收器**

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。在每个字节后，I²C 接口依次执行以下操作：

1. 如果 ACK 位被置位，发出一个应答脉冲。
2. 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断。（见下图，传输序列 EV7）。

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。

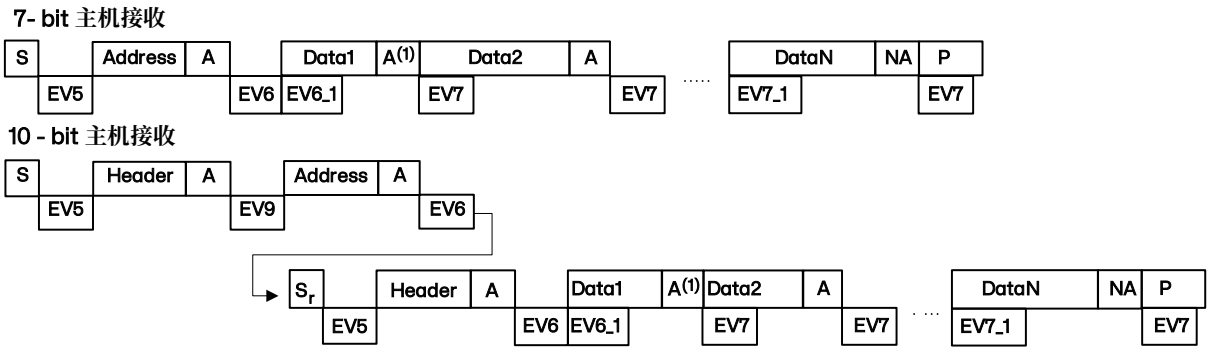
**关闭通信**

**方法 1：**此方法适用于 I2C 在应用中使用最高优先级中断的情况。

主设备在从从设备接收到最后一个字节后发送一个 NACK。接收到 NACK 后，从设备释放对 SCL 和 SDA 线的控制；主设备就可以发送一个停止/重起始条件。

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/重起始条件，软件必须在读倒数第二个数据字节之后（在倒数第二个 RxNE 事件之后）设置 STOP/START 位。
3. 只接收一个字节时，刚好在 EV6 之后（EV6_1 时，清除 ADDR 之后）要关闭应答和停止条件的产生位。

在产生了停止条件后，I²C 接口自动回到从模式（MSL 位被清除）。



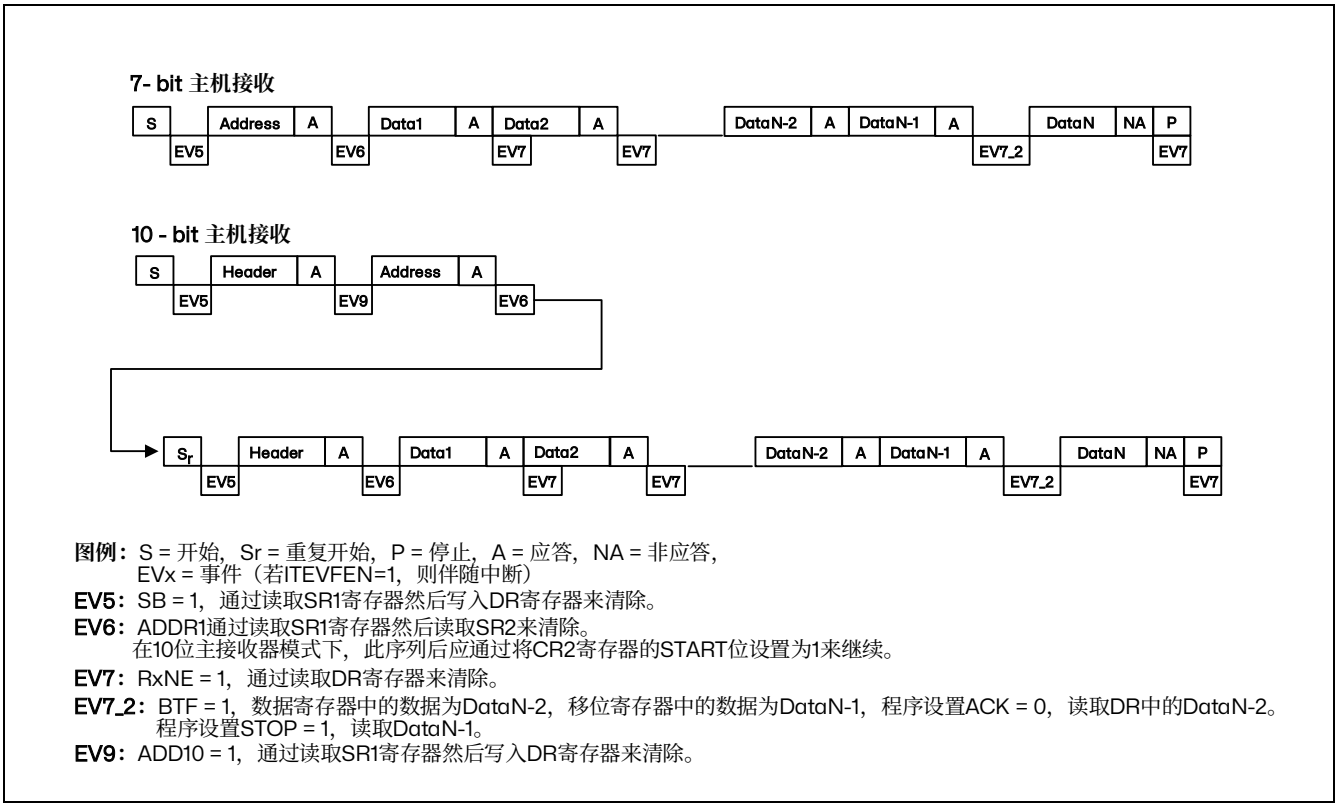

**图例：**S = 开始，Sr = 重复开始，P = 停止，A = 应答，NA = 非应答，
EVx = 事件（若ITEVFEN=1，则伴随中断）
**EV5：**SB=1，通过读取SR1寄存器然后写入DR寄存器来清除。
**EV6：**ADDR1通过读取SR1寄存器然后读取SR2来清除。
在10位主接收器模式下，此序列后应通过将CR2寄存器的START位设置为1来继续。
**EV6_1：**无相关标志事件，仅用于接收1个字节。
在EV6之后，即ADDR被清除后，立即禁用应答并生成停止条件。
**EV7：**RxNE=1，通过读取DR寄存器来清除。
**EV7_1：**RxNE=1，通过读取DR寄存器来清除，同时编程ACK=0并请求STOP。
**EV9：**ADD10 = 1，通过读取SR1寄存器然后写入DR寄存器来清除。

图 33.6 方法 1：主接收器传送序列图

1. 如果收到一个单独的字节，则为 NA 非应答。
2. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。
3. EV7 的软件序列必须在当前字节传输结束前完成。如果 EV7 软件序列无法在当前字节传输结束之前处理，建议使用 BTF 而不是 RXNE，但缺点是会降低通信速度。
4. EV6_1 或 EV7_1 的软件序列 必须在当前传输字节的 ACK 应答脉冲之前完成。

**方法 2：**此方法适用于 I2C 在应用中未使用最高优先级中断的情况或 I2C 采用轮询方式使用的情况。

在此方法中，不读取 DataN-2，因此在 DataN-1 之后，通信会被拉长（即 RxNE 和 BTF 均被置位）。然后，在读取数据寄存器（DR）中的 DataN-2 之前，清除应答（ACK）位，以确保在 DataN 应答脉冲之前清除该位。之后，紧接着读取 DataN-2 之后，设置停止（STOP）/开始（START）位，并读取 DataN-1。当 RxNE 被置位后，读取 DataN。以下是对此过程的图示说明：




**图例：**S = 开始，Sr = 重复开始，P = 停止，A = 应答，NA = 非应答，
EVx = 事件（若ITEVFEN=1，则伴随中断）
**EV5：**SB = 1，通过读取SR1寄存器然后写入DR寄存器来清除。
**EV6：**ADDR1通过读取SR1寄存器然后读取SR2来清除。
在10位主接收器模式下，此序列后应通过将CR2寄存器的START位设置为1来继续。
**EV7：**RxNE = 1，通过读取DR寄存器来清除。
**EV7_2：**BTF = 1，数据寄存器中的数据为DataN-2，移位寄存器中的数据为DataN-1，程序设置ACK = 0，读取DR中的DataN-2。
程序设置STOP = 1，读取DataN-1。
**EV9：**ADD10 = 1，通过读取SR1寄存器然后写入DR寄存器来清除。

图 33.7 方法 2：N>2 时主接收器传送序列图

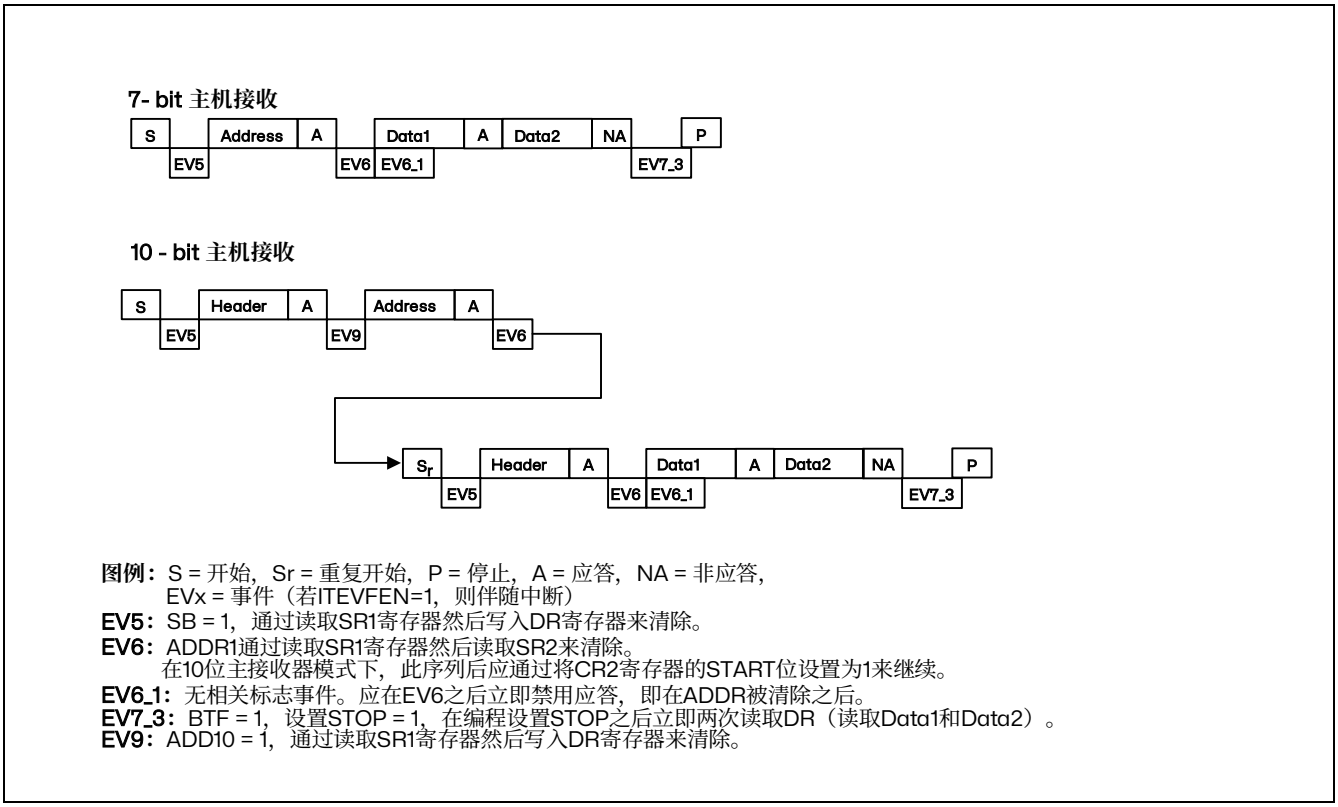
1. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。
2. EV7 的软件序列必须在当前字节传输结束前完成。如果 EV7 软件序列无法在当前字节传输结束之前处理，建议使用 BTF 而不是 RXNE，但缺点是会降低通信速度。

当剩余 3 个字节待读取时：

- RxNE = 1 => 无操作（DataN-2 尚未读取）。
- 已接收 DataN-1。
- BTF = 1，因为移位寄存器和数据寄存器均已满：DR（数据寄存器）中有 DataN-2，移位寄存器中有 DataN-1 => SCL 被拉低：总线上不会接收其他数据。
- 清除 ACK 位。
- 从 DR 中读取 DataN-2 => 这将启动移位寄存器中的 DataN 接收。
- 已接收 DataN（伴随 NACK）。
- 编程 START/STOP。
- 读取 DataN-1。
- RxNE = 1。
- 读取 DataN。

上述过程适用于 N>2 的情况。当需要接收单个字节或两个字节时，应按下述方式另行处理：

- 接收单个字节的情况：
  - 在 ADDR 事件中，清除 ACK 位
  - 清除 ADDR 标志
  - 编程设置 STOP/START 位
  - 在 RxNE 标志置位后读取数据
- 接收两个字节的情况：
  - 设置 POS 和 ACK 位
  - 等待 ADDR 标志置位
  - 清除 ADDR 标志
  - 清除 ACK 位
  - 等待 BTF 标志置位
  - 编程设置 STOP
  - 两次读取 DR




**图例：** S = 开始，Sr = 重复开始，P = 停止，A = 应答，NA = 非应答，
EVx = 事件（若ITEVFEN=1，则伴随中断）
**EV5：** SB = 1，通过读取SR1寄存器然后写入DR寄存器来清除。
**EV6：** ADDR1通过读取SR1寄存器然后读取SR2来清除。
在10位主接收器模式下，此序列后应通过将CR2寄存器的START位设置为1来继续。
**EV6_1：** 无相关标志事件。应在EV6之后立即禁用应答，即在ADDR被清除之后。
**EV7_3：** BTF = 1，设置STOP = 1，在编程设置STOP之后立即两次读取DR（读取Data1和Data2）。
**EV9：** ADD10 = 1，通过读取SR1寄存器然后写入DR寄存器来清除。

图 33.8 方法 2：N=2 时主接收器传送序列图

1. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。
2. EV6_1 软件序列必须在当前字节传输的 ACK 脉冲之前完成。

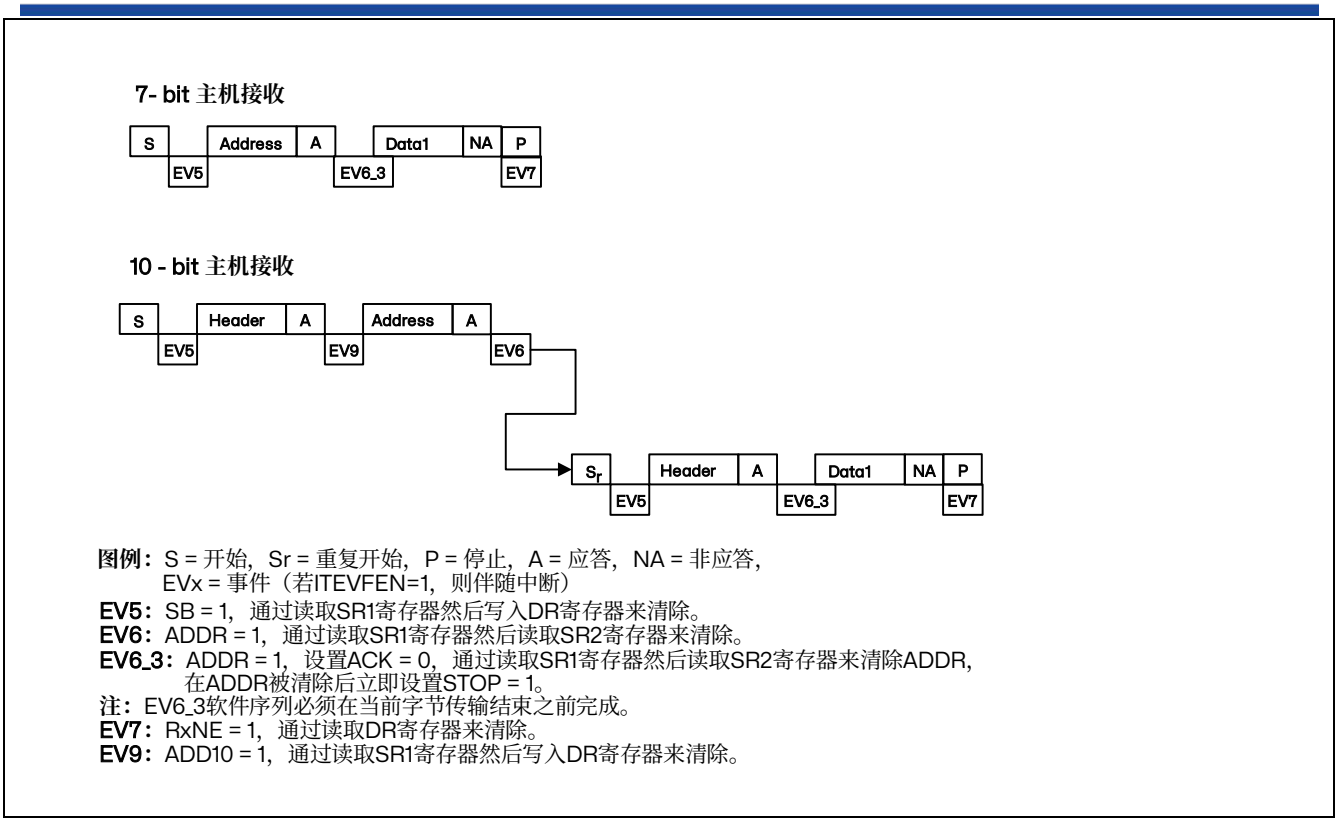



图 33.9 方法 2：N=1 时主接收器传送序列图

1. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。

### 33.3.4 错误条件

以下条件可能造成通讯失败。

**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；
- 在从模式情况下，数据被丢弃，硬件释放总线：
  - 如果是错误的开始条件，从设备认为是一个重启动，并等待地址或停止条件。
  - 如果是错误的停止条件，从设备按正常的停止条件操作，同时硬件释放总线。
- 在主模式情况下，硬件不释放总线，同时不影响当前的传输状态。此时由软件决定是否要中止当前的传输。

**应答错误（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：
  - 如果是处于从模式，硬件释放总线。
  - 如果是处于主模式，软件必须生成一个停止条件或重复起始条件。

**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设备发送重起始条件之后响应；
- 硬件释放总线。

**过载/欠载错误（OVR）**

在从模式下，如果禁止时钟延长，I²C 接口正在接收数据时，当它已经接收到一个字节（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错时，接收端应丢弃重复接收到的数据。发送端应按 I²C 总线标准在规定的时间更新 DR 寄存器。

在发送第一个字节时，必须在清除 ADDR 之后并且第一个 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。

### 33.3.5 SDA/SCL 线控制

- 如果允许时钟延长：
  - 发送器模式：如果 TxE=1 且 BTF=1：I²C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器（缓冲器和移位寄存器都是空的）。
  - 接收器模式：如果 RxNE=1 且 BTF=1：I²C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR（缓冲器和移位寄存器都是满的）。
- 如果在从模式中禁止时钟延长：
  - 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载错。接收到的最后一个字节丢失。
  - 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载错。相同的字节将被重复发出。
  - 不控制重复写冲突。

### 33.3.6 SMBus

#### 介绍

系统管理总线（SMBus）是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I²C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线（SMBus）标准涉及三类设备。从设备：接收或响应命令的设备。主设备：用来发送命令、产生时钟和终止发送的设备。主机：一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能并且必须支持 SMBus 提醒协议。一个系统里只允许有一个主机。

#### SMBus 和 I²C 之间的相似点

- 2 条线的总线协议（1 个时钟，1 个数据） + 可选的 SMBus 提醒线；
- 主-从通信，主设备提供时钟；
- 多主机功能
- SMBus 数据格式类似于 I²C 的 7 位地址格式；

#### SMBus 和 I2C 之间的不同点

下表列出了 SMBus 和 I²C 的不同点。

表 33.1 SMBus 与 I²C 的比较

| SMBus | I²C |
|---|---|
| 最大传输速度 100kHz | 最大传输速度 400kHz |
| 最小传输速度 10kHz | 无最小传输速度 |
| 35ms 时钟低超时 | 无时钟超时 |
| 固定的逻辑电平 | 逻辑电平由 $V_{DD}$ 决定 |
| 不同的地址类型（保留的、动态的等） | 7 位、10 位和广播呼叫从地址类型 |
| 不同的总线协议（快速命令、处理呼叫等） | 无总线协议 |

#### SMBus 应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

#### 设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的 SMBus 规范（http://smbus.org/）。

#### 总线协议

SMBus 技术规范支持 9 个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考 2.0 版的 SMBus 规范。这些协议由用户的软件来执行。

**地址解析协议（ARP）**

通过给每个从设备动态地分配一个新的唯一地址，可以解决 SMBus 的从地址冲突。地址解析协议（ARP）具有以下的特性：

- 使用标准 SMBus 物理层仲裁机制分配地址；
- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销（也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的）；
- 任何一个 SMBus 主设备可以遍历总线。

**唯一的设备标识符（UDID）**

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

有关 128 位 UDID 和 ARP 的更多信息，请参阅 SMBus 规范版本 2.0。

**SMBus 提醒模式**

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBA 和 SCL、SDA 信号一样，是一种线与信号。SMBA 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。

一个只具有从功能的设备，可以通过设置 I2C_CR1 寄存器上的 ALERT 位，使用 SMBA 给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址 ARA（Alert Response Address，地址值为 0001100x）访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备 能应答 ARA。此状态是由 I2C_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 SMBA 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 SMBA，如果当信息传输完成后，主机仍看到 SMBA 低，就知道需要再次读 ARA。

没有实现 SMBA 信号的主机可以定期访问 ARA。

有关 SMBus 提醒模式的更多详细信息，请参阅 SMBus 规范版本 2.0。

**超时错误**

在定时规范上 I²C 和 SMBus 之间有很多差别。

SMBus 定义了一个时钟低超时，35ms 的超时。SMBus 规定 TLOW:SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW:MEXT 为主设备的累积时钟低扩展时间。有关这些超时的更多详细信息，请参阅 SMBus 规范版本 2.0。

I2C_SR1 中的状态标志 Timeout 或 Tlow 错误表明了这个特性的状态。

**如何使用 SMBus 模式的接口**

为了从 I²C 模式切换到 SMBus 模式，应该执行下列步骤：

- 设置 I2C_CR1 寄存器中的 SMBus 位；
- 按应用要求配置 I2C_CR1 寄存器中的 SMBTYPE 和 ENARP 位。

如果您想将设备配置为主设备，请遵循章节：I2C 主模式中的起始条件生成程序。否则，请遵循章

节：I2C 从模式中的序列。

软件程序必须处理多种 SMBus 协议。

- 如果 ENARP=1 且 SMBTYPE=0，使用 SMB 设备默认地址。
- 如果 ENARP=1 且 SMBTYPE=1，使用 SMB 主设备头位域。
- 如果 SMBA=1，使用 SMB 提醒响应地址。

### 33.3.7 DMA 请求

DMA 请求（当被使能时）仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。必须在 ADDR 事件之前设置 I2C_CR2 寄存器中的 DMAEN 位。在主模式或从模式（且时钟延长启用时）下，也可以在 ADDR 事件期间、清除 ADDR 标志之前设置 DMAEN 位。必须在当前字节传输结束之前处理 DMA 请求。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 EOT 到 I2C 接口，并且在中断允许时产生一个传输完成中断：

- 主发送器：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后设置停止条件。
- 主接收器：当要接收的数据数目大于或等于 2 时，DMA 控制器发送一个硬件信号 EOT_1，它对应 DMA 传输（字节数-1）。如果在 I2C_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。

#### 利用 DMA 发送

通过设置 I2C_CR2 寄存器中的 DMAEN 位可以激活 DMA 模式。只要 TxE 位被置位，数据将 DMA 从预置的存储区装载进 I2C_DR 寄存器。为 I²C 分配一个 DMA 通道，须执行以下步骤(x 是通道号)：

1. 在 DMA_CPARx 寄存器中设置 I2C_DR 寄存器地址。数据将在每个 TxE 事件后从存储器传送至这个地址。
2. 在 DMA_CMARx 寄存器中设置存储器地址。数据在每个 TxE 事件后从这个存储区传送至 I2C_DR。
3. 在 DMA_CNDTRx 寄存器中设置所需的传输字节数。在每个 TxE 事件后，此值将被递减。
4. 利用 DMA_CCRx 寄存器中的 PL[0:1]位配置通道优先级。
5. 设置 DMA_CCRx 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置 DMA_CCRx 寄存器上的 EN 位激活通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

*注：如果使用* DMA *进行发送时，不要设置* I2C_CR2 *寄存器的* ITBUFEN *位。*

#### 利用 DMA 接收

通过设置 I2C_CR2 寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2C_DR 寄存器的数据传送到设置的存储区（参考 DMA 说明）。设置 DMA 通道进行 I²C 接收，须执行以下步骤（x 是通道号）：

1. 在 DMA_CPARx 寄存器中设置 I2C_DR 寄存器的地址。数据将在每次 RxNE 事件后从此地址传送到存储区。

2. 在 DMA_CMARx 寄存器中设置存储区地址。数据将在每次 RxNE 事件后从 I2C_DR 寄存器传送到此存储区。
3. 在 DMA_CNDTRx 寄存器中设置所需的传输字节数。在每个 RxNE 事件后，此值将被递减。
4. 用 DMA_CCRx 寄存器中的 PL[0:1]配置通道优先级。
5. 清除 DMA_CCRx 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置 DMA_CCRx 寄存器中的 EN 位激活该通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I²C 接口发送一个传输结束的 EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

*注：如果使用* DMA *进行接收时，不要设置* I2C_CR2 *寄存器的* ITBUFEN *位。*

### 33.3.8 包错误校验（PEC）

包错误校验（PEC）计算器是用于提高通信的可靠性，这个计算器使用下述 CRC-8 多项式对每一位串行数据进行计算：$C(x) = x^8 + x^2 + x + 1$。

- PEC 计算由 I2C_CR1 寄存器的 ENPEC 位激活。PEC 使用 CRC-8 算法对所有信息字节进行计算，包括地址和读/写位在内。
  - 在发送时：在最后一个 TxE 事件时设置 I2C_CR1 寄存器的 PEC 传输位，PEC 将在最后一个字节后被发送。
  - 在接收时：在最后一个 RxNE 事件之后设置 I2C_CR1 寄存器的 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 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 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。

## 33.4 I²C 中断请求

下表列出了所有的 I²C 中断请求

表 33.2 I²C 中断请求

| 中断事件 | 事件标志 | 开启控制位 |
|---|---|---|
| 起始位已发送（主） | SB | ITEVFEN |
| 地址已发送（主）或地址匹配（从） | ADDR | |
| 10 位头段已发送（主） | ADD10 | |

| 中断事件 | 事件标志 | 开启控制位 |
|---|---|---|
| 已收到停止（从） | STOPF | |
| 数据字节传输完成 | BTF | |
| 接收缓冲区非空 | RxNE | ITEVFEN 和 ITBUFEN |
| 发送缓冲区空 | TxE | |
| 总线错误 | BERR | |
| 仲裁丢失（主） | ARLO | |
| 响应失败 | AF | |
| 过载/欠载 | OVR | ITERREN |
| PEC 错误 | PECERR | |
| 超时/Tlow 错误 | TIMEOUT | |
| SMBus 提醒 | SMBALERT | |

*注：* SB、ADDR、ADD10、STOPF、BTF、RxNE *和* TxE *通过逻辑或汇到同一个中断通道中。*

BERR、ARLO、AF、OVR、PECERR、TIMEOUT *和* SMBALERT *通过逻辑或汇到同一个中断通道中。*

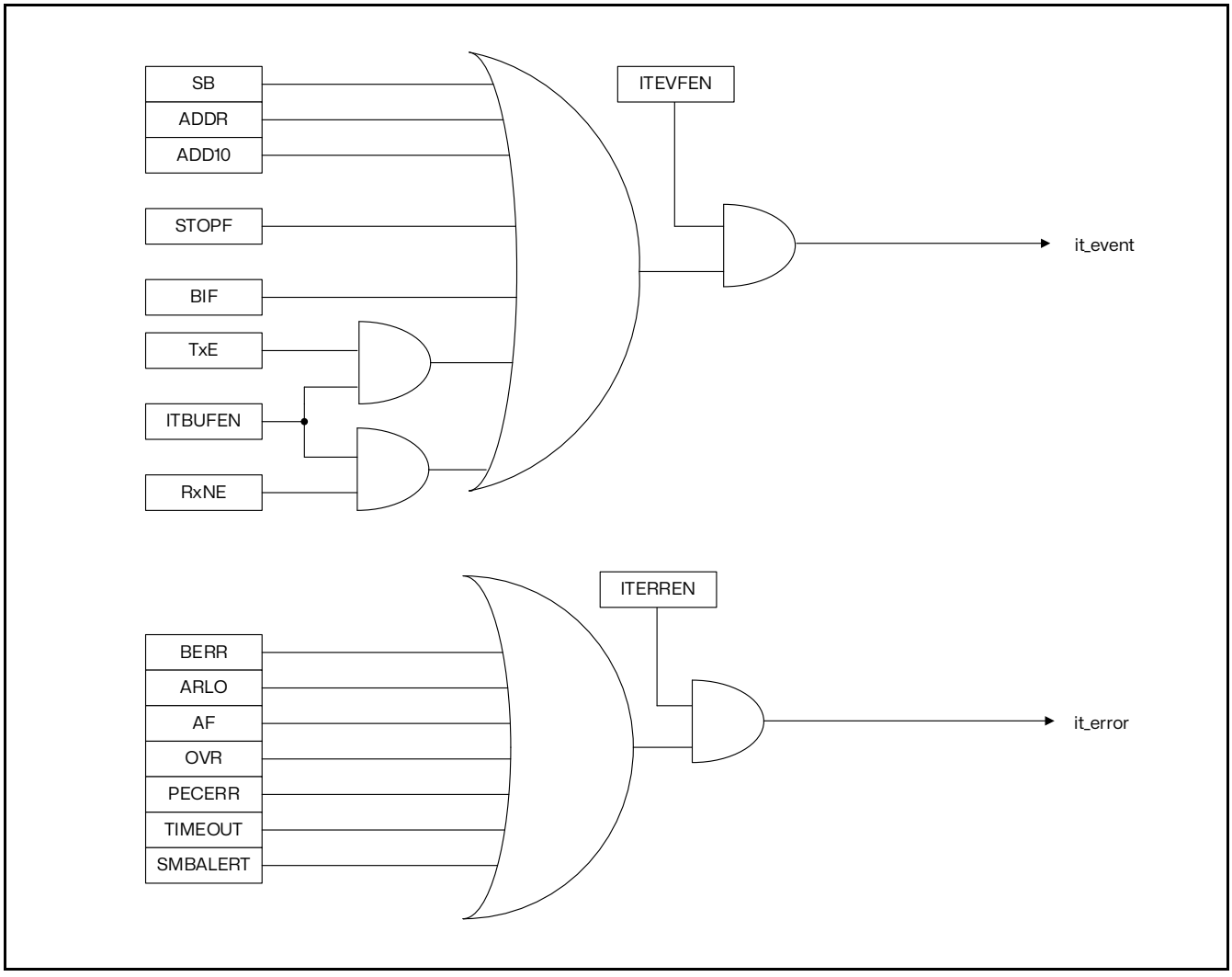



图 33.10 I²C 中断映射图

## 33.5 I²C 调试模式

当微控制器进入调试模式（内核处于停止状态）时，根据 DBG 模块中的 DBG_I2Cx_STOP 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。有关详细信息，请参见章节 36.7.2：对定时器、RTC、看门狗和 I2C 的调试支持。

## 33.6 I²C 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。
外设寄存器可以通过半字（16 位）或字（32 位）进行访问。

### 33.6.1 控制寄存器 1（I2C_CR1）

地址偏移：0x00
复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWRST | Res | ALERT | PEC | POS | ACK | STOP | START | NOSTRETCH | ENGC | ENPEC | ENARP | SMBTYPE | Res | SMBUS | PE |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw |

Bit 15 **SWRST**：软件复位（Software reset）

当被置位时，I²C 处于复位状态。在复位该位前确信 I²C 的引脚被释放，总线是空的。

0：I²C 模块不处于复位状态；

1：I²C 模块处于复位状态。

*注：当出现错误或锁定状态后，可使用此位重新初始化外设。例如，如果* BUSY *位已置*1 *但因母线干扰而不能复位，则可使用* SWRST *位退出此状态。*

Bit 14 保留，必须保持为复位值。

Bit 13 **ALERT**：SMBus 提醒（SMBus alert）

软件可以设置或清除该位；当 PE=0 时，由硬件清除。

0：释放 SMBAlert 引脚使其变高。提醒响应地址头紧跟在 NACK 信号后面；

1：驱动 SMBAlert 引脚使其变低。提醒响应地址头紧跟在 ACK 信号后面。

Bit 12 **PEC**：数据包出错检测（Packet error checking）

软件可以设置或清除该位；当传送 PEC 后，或起始或停止条件时，或当 PE=0 时硬件将其清除。

0：无 PEC 传输；

1：PEC 传输（在发送或接收模式）。

*注：仲裁丢失时，*PEC *的计算失效。*

Bit 11 **POS**：应答/PEC 位置（用于数据接收）（Acknowledge/PEC Position）

软件可以设置或清除该位，或当 PE=0 时，由硬件清除。

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：当遵循接收 2 个字节的程序时（见方法 2：N=2 时主接收器传输时序图），会使用到 POS 位。POS 位必须在数据接收开始之前配置好。在此情况下，若要对第 2 个字节发送非应答（NACK），必须在 ADDR（地址）位被清除后立即清除 ACK（应答）位。若要将第 2 个字节作为数据包错误检测码（PEC）进行检查，则必须在配置好 POS 位后，在 ADDR 拉伸事件期间设置 PEC 位。*

Bit 10 **ACK**：应答使能（Acknowledge enable）

软件可以设置或清除该位，或当 PE=0 时，由硬件清除。

0：无应答返回；

1：在接收到一个字节后返回一个应答（匹配的地址或数据）。

Bit 9 **STOP**：停止条件产生（Stop generation）

软件可以设置或清除该位；或当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件将其置位。

在主模式下：

0：无停止条件产生；

1：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：在当前字节传输或释放 SCL 和 SDA 线。

Bit 8 **START**：起始条件产生（Start generation）

软件可以设置或清除该位，或当起始条件发出后或 PE=0 时，由硬件清除。

在主模式下：

0：无起始条件产生；

1：重复产生起始条件。

在从模式下：

0：无起始条件产生；

1：当总线空闲时，产生起始条件。

Bit 7 **NOSTRETCH**：禁止时钟延长（从模式）（Clock stretching disable (Slave mode)）

该位用于当 ADDR 或 BTF 标志被置位，在从模式下禁止时钟延长，直到它被软件复位。

0：允许时钟延长；

1：禁止时钟延长。

Bit 6 **ENGC**：广播呼叫使能（General call enable）

0：禁止广播呼叫。以非应答响应地址 00h；

1：允许广播呼叫. 以应答响应地址 00h。

Bit 5 **ENPEC**：PEC 使能（PEC enable）

0：禁止 PEC 计算；

1：开启 PEC 计算。

Bit 4 **ENARP**：ARP 使能（ARP enable）

0：禁止 ARP；

1：使能 ARP。

如果 SMBTYPE=0，使用 SMBus 设备的默认地址。

如果 SMBTYPE=1，使用 SMBus 的主地址。

Bit 3 **SMBTYPE**：SMBus 类型（SMBus type）

0：SMBus 设备；

1：SMBus 主机。

Bit 2 保留，必须保持为复位值。

Bit 1 **SMBUS**：SMBus 模式（SMBus mode）

0：I²C 模式；

1：SMBus 模式。

Bit 0 **PE**：I²C 模块使能 （Peripheral enable）

0：禁用 I²C 模块；

1：使能 I²C 模块。

*注：如果清除该位时通讯正在进行，在当前通讯结束后，I²C 模块被禁用并返回空闲状态。*
*由于在通讯结束后发生 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。*

*注：* STOP、START *或* PEC *位置*1 *之后，在硬件将该位清零之前，软件不能对* I2C_CR1 *执行任何写操作。否则，可能会再次发出* STOP、START *或* PEC *置位请求。*

### 33.6.2 控制寄存器 2（I2C_CR2）

地址偏移：0x04

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | LAST | DMAEN | ITBUFEN | ITEVTEN | ITERREN | Reserved | | FREQ[5:0] | | | | | |
| | | | rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw |

Bit 15:13 保留，必须保持为复位值。

Bit 12 **LAST**：DMA 最后一次传输（DMA last transfer）

0：下一次 DMA 的 EOT 不是最后的传输；

1：下一次 DMA 的 EOT 是最后的传输。

*注：该位在主接收模式使用，使得在最后一次接收数据时可以产生一个* NACK。

Bit 11 **DMAEN**：DMA 请求使能（DMA requests enable）

0：禁止 DMA 请求；

1：当 TxE=1 或 RxNE =1 时，允许 DMA 请求。

Bit 10 **ITBUFEN**：缓冲器中断使能（Buffer interrupt enable）

0：当 TxE=1 或 RxNE=1 时，不产生任何中断；

1：当 TxE=1 或 RxNE=1 时，产生事件中断（不管 DMAEN 是何种状态）。

Bit 9 **ITEVTEN**：事件中断使能（Event interrupt enable）

0：禁止事件中断；

1：允许事件中断。

在下列条件下，将产生该中断：

- SB = 1（主模式）
- ADDR = 1（主/从模式）
- ADD10 = 1（主模式）
- STOPF = 1（从模式）
- BTF = 1，但是没有 TxE 或 RxNE 事件
- 如果 ITBUFEN = 1，TxE 事件为 1
- 如果 ITBUFEN = 1，RxNE 事件为 1

Bit 8 **ITERREN**：出错中断使能（Error interrupt enable）

0：禁止出错中断；

1：允许出错中断。

在下列条件下，将产生该中断：

- BERR = 1
- ARLO = 1
- AF = 1
- OVR = 1
- PECERR = 1
- TIMEOUT = 1
- SMBAlert = 1

Bit 7:6 保留，必须保持为复位值。

Bit 5:0 **FREQ[5:0]**：外设时钟频率（Peripheral clock frequency）

外设利用 FREQ 位域生成符合 I2C 规范的数据建立时间和保持时间。最小运行频率为 2MHz，最大频率受 APB 频率限制，且不得超过 36MHz（外设固有最大限制）。

0b000000：禁用

0b000001：禁用

0b000010：2MHz

...

0b100100：36MHz

大于 100100：禁用。

### 33.6.3 自身地址寄存器 1（I2C_OAR1）

地址偏移：0x08

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD MODE | Reserved | | | | | ADD[9:8] | | ADD[7:1] | | | | | | | ADD0 |
| rw | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15 **ADDMODE：**寻址模式（从模式）（Addressing mode (slave mode)）

0：7 位从地址（不响应 10 位地址）；

1：10 位从地址（不响应 7 位地址）。

Bit 14 必须始终由软件保持为“1”。

Bit 13:10 保留，必须保持为复位值 0。

Bit 9:8 **ADD[9:8]：**接口地址（Interface address）

7 位地址模式时不用关心。

10 位地址模式时为地址的 9~8 位。

Bit 7:1 **ADD[7:1]：**接口地址（Interface address）

地址的 7~1 位。

Bit 0 **ADD0：**接口地址（Interface address）

7 位地址模式时不用关心。

10 位地址模式时为地址第 0 位。

### 33.6.4 自身地址寄存器 2（I2C_OAR2）

地址偏移：0x0C

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | ADD2[7:1] | | | | | | | ENDUAL |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15:8 保留，必须保持为复位值。

Bit 7:1 **ADD2[7:1]：**接口地址（Interface address）

在双地址模式下地址的 7~1 位。

Bit 0 **ENDUAL：**双地址模式使能位（Dual addressing mode enable）

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。

### 33.6.5 数据寄存器（I2C_DR）

地址偏移：0x10

复位值：0x0000

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

Bit 15:8 保留，必须保持为复位值 0。

Bit 7:0 **DR[7:0]：** 8 位数据寄存器（8-bit data register）

用于存放接收到的数据或放置用于发送到总线的数据

发送器模式：当写一个字节至 DR 寄存器时，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（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传送。

*注：在从模式下，地址不会被拷贝进数据寄存器 DR；*

*注：硬件不管理写冲突（如果 TxE=0，仍能写入数据寄存器）；*

*注：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，因此不能读到它。*

### 33.6.6 状态寄存器 1（I2C_SR1）

地址偏移：0x14

复位值：0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMB ALERT | TIME OUT | Res | PEC ERR | OVR | AF | ARLO | BERR | TxE | RxNE | Res | STOPF | ADD10 | BTF | ADDR | SB |
| rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | r | r | | r | r | r | r | r |

Bit 15 **SMBALERT：** SMBus 提醒（SMBus alert）

在 SMBus 主机模式下：

0：无 SMBus 提醒；

1：在引脚上产生 SMBAlert 提醒事件。

在 SMBus 从机模式下：

0：没有 SMBAlert 响应地址头序列；

1：收到 SMBAlert 响应地址头序列至 SMBAlert 变低。

– 该位由软件写“0”清除，或在 PE=0 时由硬件清除。

Bit 14 **TIMEOUT：** 超时或 Tlow 错误（Timeout or Tlow error）

0：无超时错误；

1：SCL 处于低已达到 25ms（超时）；或者主机低电平累积时钟扩展时间超过 10ms（Tlow:mext）；或从设备低电平累积时钟扩展时间超过 25ms（Tlow:sext）。

– 当在从模式下设置该位：从设备复位通讯，硬件释放总线。

– 当在主模式下设置该位：硬件发出停止条件。

– 该位由软件写“0”清除，或在 PE=0 时由硬件清除。

Bit 13 保留，必须保持为复位值。

Bit 12 **PECERR**：在接收时发生 PEC 错误（PEC Error in reception）

0：无 PEC 错误：接收到 PEC 后接收器返回 ACK（如果 ACK=1）；

1：有 PEC 错误：接收到 PEC 后接收器返回 NACK（不管 ACK 是什么值）。

– 该位由软件写“0”清除，或在 PE=0 时由硬件清除。

Bit 11 **OVR**：过载/欠载（Overrun/Underrun）

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，同时：

– 在接收模式中当收到一个新的字节时（包括 ACK 应答脉冲），数据寄存器里的内容还未被读出，则新接收的字节将丢失。

– 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。

– 该位由软件写“0”清除，或在 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。*

Bit 10 **AF**：应答失败（Acknowledge failure）

0：没有应答失败；

1：应答失败。

– 当没有返回应答时，硬件将置该位为“1”。

– 该位由软件写“0”清除，或在 PE=0 时由硬件清除。

Bit 9 **ARLO**：仲裁丢失（主模式）（Arbitration lost（master mode））

0：没有检测到仲裁丢失；

1：检测到仲裁丢失。

当接口失去对总线的控制给另一个主机时，硬件将将置该位为“1”。

– 该位由软件写“0”清除，或在 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，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间（不包括地址的应答）。*

Bit 8 **BERR**：总线出错（Bus error）

0：无起始或停止条件出错；

1：起始或停止条件出错。

– 当接口检测到错误的起始或停止条件，硬件将该位置“1”。

– 该位由软件写“0”清除，或在 PE=0 时由硬件清除。

Bit 7 **TxE**：数据寄存器为空（发送时）（Data register empty (transmitters)）

0：数据寄存器非空；

1：数据寄存器空。

– 在发送数据时，数据寄存器为空时该位被置“1”，在发送地址阶段不设置该位。

– 软件写数据到 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，这是因为数据寄存器仍然为空。*

Bit 6 **RxNE**：数据寄存器非空（接收时）（Data register not empty (receivers)）

0：数据寄存器为空；

1：数据寄存器非空。

– 在接收时，当数据寄存器不为空，该位被置“1”。在接收地址阶段，该位不被置位。

– 软件对数据寄存器的读写操作清除该位，或当 PE=0 时由硬件清除。

在发生 ARLO 事件时，RxNE 不被置位。

*注：当设置了*BTF *时，读取数据不能清除*RxNE *位，因为数据寄存器仍然为满。*

Bit 5 保留，必须保持为复位值。

Bit 4 **STOPF**：停止条件检测位（从模式）（Stop detection (slave mode)）

0：没有检测到停止条件；

1：检测到停止条件。

– 在一个应答之后（如果 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”。

– 软件读取 SR1 寄存器后，对 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 *位不被置位。*

*建议在*STOPF *置*1 *后执行完整的清零序列（首先读取*SR1，*然后写入*CR1*）。请参见图：从接收器的传输序列图。*

Bit 3 **ADD10**：10 位头序列已发送（主模式）（10-bit header sent (Master mode)）

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，当主设备已经将第一个字节发送出去时，硬件将该位置“1”。

– 软件读取 SR1 寄存器后，对 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：收到一个*NACK *后，*ADD10 *位不被置位。*

Bit 2 **BTF**：字节发送结束（Byte transfer finished）

0：字节发送未完成；

1：字节发送结束。

当 NOSTRETCH=0 时，在下列情况下硬件将该位置“1”：

– 在接收时，当收到一个新字节（包括 ACK 脉冲）且数据寄存器还未被读取（RxNE=1）。

– 在发送时，当一个新数据将被发送且数据寄存器还未被写入新的数据（TxE=1）。

– 在软件读取 SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除该位。

*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_SR2 *寄存器中*TRA *为“*1*”，同时*I2C_CR1 *寄存器中*PEC *为“*1*”），*BTF *位不会被置位。*

Bit 1 **ADDR：**地址已被发送（主模式）/地址匹配（从模式）（Address sent (master mode) /matched (slave mode)）

在软件读取 SR1 寄存器后，对 SR2 寄存器的读操作将清除该位，或当 PE=0 时，由硬件清除该位。

**地址匹配（从模式）**

0：地址不匹配或没有收到地址；

1：收到的地址匹配。

– 当收到的从地址与 OAR 寄存器中的内容相匹配、或发生广播呼叫、或 SMBus 设备默认地址或 SMBus 主机识别出 SMBus 提醒时，硬件就将该位置“1”（当对应的设置被使能时）。

*注：在从模式下，建议在*ADDR *置*1 *后执行完整的清零序列（首先读取*SR1，*然后写入*SR2*）。请参见图：从接收器的传输序列图。*

**地址已被发送（主模式）**

0：地址发送没有结束；

1：地址发送结束。

– 10 位地址模式时，当收到地址的第二个字节的 ACK 后该位被置“1”。

– 7 位地址模式时，当收到地址的 ACK 后该位被置“1”。

*注：在收到*NACK *后，*ADDR *位不会被置位。*

Bit 0 **SB：**起始位（主模式）（Start bit (Master mode)）

0：未发送起始条件；

1：起始条件已发送。

– 当发送出起始条件时该位被置“1”。

– 软件读取 SR1 寄存器后，写数据寄存器的操作将清除该位，或当 PE=0 时，硬件清除该位。

### 33.6.7 状态寄存器 2（I2C_SR2）

地址偏移：0x18

复位值：0x0000

*注：读取*I2C_SR1 *后再读取*I2C_SR2 *可将*ADDR *标志清零，即使*ADDR *标志在读取*I2C_SR1 *之后置*1 *也如此。因此，必须仅在*I2C_SR1 *中的*ADDR *位已置*1 *或者*STOPF *位已清零后读取*I2C_SR2*。*

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PEC[7:0] | | | | | | | | DUALF | SMB HOST | SMBDE FAULT | GEN CALL | Res | TRA | BUSY | MSL |
| r | r | r | r | r | r | r | r | r | r | r | r | | r | r | r |

Bit 15:8 **PEC[7:0]：**数据包出错检测（Packet error checking register）

当 ENPEC=1 时，PEC[7:0]存放内部的 PEC 的值。

Bit 7 **DUALF：**双标志（从模式）（Dual flag）

0：接收到的地址与 OAR1 内的内容相匹配；

1：接收到的地址与 OAR2 内的内容相匹配。

– 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。

Bit 6 **SMBHOST：**SMBus 主机头系列（从模式）（SMBus host header）

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。

– 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。

Bit 5 **SMBDEFAULT：**SMBus 设备默认地址（从模式）（SMBus device default address）

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。

– 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。

Bit 4 **GENCALL：**广播呼叫地址（从模式）（General call address）

0：未收到广播呼叫地址；

1：当 ENGC=1 时，收到广播呼叫的地址

– 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。

Bit 3 保留，必须保持为复位值。

Bit 2 **TRA：**发送/接收（Transmitter/receiver）

0：接收到数据；

1：数据已发送；

在整个地址传输阶段的结尾，该位根据地址字节的 R/W 位来设定。

在检测到停止条件（STOPF=1）、重复的起始条件或总线仲裁丢失（ARLO=1）后，或当 PE=0 时，硬件将其清除。

Bit 1 **BUSY：**总线忙（Bus busy）

0：总线上无数据通讯；

1：总线上正在进行数据通讯。

– 在检测到 SDA 或 SCI 为低电平时，硬件将该位置“1”；

– 当检测到一个停止条件时，硬件将该位清除。

该位指示当前正在进行的总线通讯，当接口被禁用（PE=0）时该信息仍然被更新。

Bit 0 **MSL：**主从模式（Master/slave）

0：从模式；

1：主模式。

– 当接口处于主模式（SB=1）时，硬件将该位置位；

– 当总线上检测到一个停止条件、仲裁丢失（ARLO=1 时）、或当 PE=0 时，硬件清除该位。

*注：读取* I2C_SR1 *后再读取* I2C_SR2 *可将* ADDR *标志清零，即使* ADDR *标志在读取* I2C_SR1 *之后置* 1 *也如此。因此，必须仅在* I2C_SR1 *中的* ADDR *位已置* 1 *或者* STOPF *位已清零后读取* I2C_SR2*。*

### 33.6.8 时钟控制寄存器（I2C_CCR）

地址偏移：0x1C

复位值：0x0000

*注：$f_{PCLK1}$ 必须至少为2MHz，才能达到标准模式 I²C 频率。必须至少为 4 MHz 才能达到快速模式 I²C 频率。必须为10MHz 的倍数才能实现最大 400 kHz 的 I²C 快速模式时钟。*

*CCR 寄存器只有在关闭 I²C 时（PE=0）才能设置。*

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F/S | DUTY | Reserved | | CCR[11:0] | | | | | | | | | | | |
| rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 15 **F/S**：I²C 主模式选项（I2C master mode selection）

0：标准模式的 I²C；

1：快速模式的 I²C。

Bit 14 **DUTY**：快速模式时的占空比（Fast mode duty cycle）

0：快速模式下：$t_{low}/t_{high}$ = 2；

1：快速模式下：$t_{low}/t_{high}$ = 16/9（参见 CCR）。

Bit 13:12 保留，必须保持为复位值。

Bit 11:0 **CCR[11:0]**：快速/标准模式下的时钟控制分频系数（主模式）（Clock control register in Fast/Standard mode (Master mode)）

该分频系数用于设置主模式下的 SCL 时钟。

在 I²C 标准模式或 SMBus 模式下：

$T_{high} = CCR \times T_{PCLK1}$

$T_{low} = CCR \times T_{PCLK1}$

在 I²C 快速模式下：

如果 DUTY = 0：

$T_{high} = CCR \times T_{PCLK1}$

$T_{low} = 2 \times CCR \times T_{PCLK1}$

如果 DUTY = 1：

$T_{high} = 9 \times CCR \times T_{PCLK1}$

$T_{low} = 16 \times CCR \times T_{PCLK1}$

例如：在标准模式下，产生 100kHz 的 SCL 的频率：

如果 FREQ=08，$T_{PCLK1}$ = 125ns，则 CCR 必须写入 0x28

（0x28 <=> 40d × 125ns = 5000 ns）。

***注：****允许设定的最小值为0x04，在快速 DUTY 模式下允许的最小值为0x01；*

$t_{high} = t_{r(SCL)}+t_{w(SCLH)}$，*详见数据手册中对这些参数的定义；*

$t_{low} = t_{f(SCL)}+t_{w(SCLL)}$，*详见数据手册中对这些参数的定义；*

I2C *通信速率* $f_{SCL} \approx 1/(t_{high} + t_{low})$。*实际频率可能因模拟噪声滤波器输入延迟存在差异；*

*只有在关闭 I²C 时（PE = 0）才能设置 CCR 寄存器；*

### 33.6.9 TRISE 寄存器（I2C_TRISE）

地址偏移：0x20

复位值：0x0002

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

Bit 15:6 保留，必须保持为复位值。

Bit 5:0 **TRISE[5:0]：**在快速/标准模式下的最大上升时间（主模式）（Maximum rise time in Fast/Standard mode (Master mode)）

这些位用于设置主模式中 SCL 反馈环路的最大持续时间，其目的是无论 SCL 上升沿持续时间如何变化，均能维持 SCL 频率的稳定性。

编程时需将 I2C 总线规范中规定的最大 SCL 上升时间值加 1 后写入该位域。

例如：标准模式中最大允许 SCL 上升时间为 1000ns。若 $T_{PCLK1}$=125ns，故 TRISE[5:0]中必须写入 09h（1000ns/125 ns = 8+1）。

滤波器的值也可以加到 TRISE[5:0]内。

如果结果不是一个整数，则将整数部分写入 TRISE[5:0]以确保 $t_{HIGH}$ 参数。

*注：只有当* I²C *被禁用（*PE=0*）时，才能设置* TRISE[5:0]。

# 34 通用异步收发器（UART）

## 34.1 UART 介绍

通用异步收发器（UART）提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用分数波特率发生器提供宽范围的波特率选择。

它支持单向通信和半双工单线通信，也支持 LIN（局部互连网），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。

## 34.2 UART 主要特性

- 全双工的，异步通信
- NRZ 标准格式
- 分数波特率发生器系统
- 可编程数据字长度（8 位或 9 位）
- 可配置的停止位-支持 1 或 2 个停止位
- LIN 主发送同步断开符的能力以及 LIN 从检测断开符的能力
  - 当 UART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 单线半双工通信
- 可配置的使用 DMA 的多缓冲器通信
  - 在 SRAM 里利用集中式 DMA 缓冲接收/发送字节
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误
  - 噪音错误
  - 帧错误
  - 校验错误
- 9 个带标志的中断源
  - CTS 改变
  - LIN 断开符检测
  - 发送数据寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 溢出错误
  - 帧错误

  - 噪音错误
  - 校验错误
- 多处理器通信
  - 如果地址不匹配，则进入静默模式
- 从静默模式中唤醒（通过地址标志检测）
- 唤醒接收器的方式：地址位（MSB，第 9 位）

## 34.3 UART 功能概述

接口通过三个引脚与其他设备连接在一起。任何 UART 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音，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。当发送器被激活，并且未发送数据时，TX 引脚处于高电平；在发送完数据后，TX 引脚处于高阻态，需要配置内部上拉来保持 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。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字（8 或 9 位），最低有效位在前
- 1 或 2 个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器 –– 12 位整数和 4 位小数的表示方法。
- 一个状态寄存器（UART_SR）
- 数据寄存器（UART_DR）
- 一个波特率寄存器（UART_BRR），12 位的整数和 4 位小数

关于以上寄存器中每个位的具体定义，请参考章节：UART 寄存器描述。

下列引脚在硬件流控模式中需要：

- nCTS：清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- nRTS：发送请求，若是低电平，表明 UART 准备好接收数据。

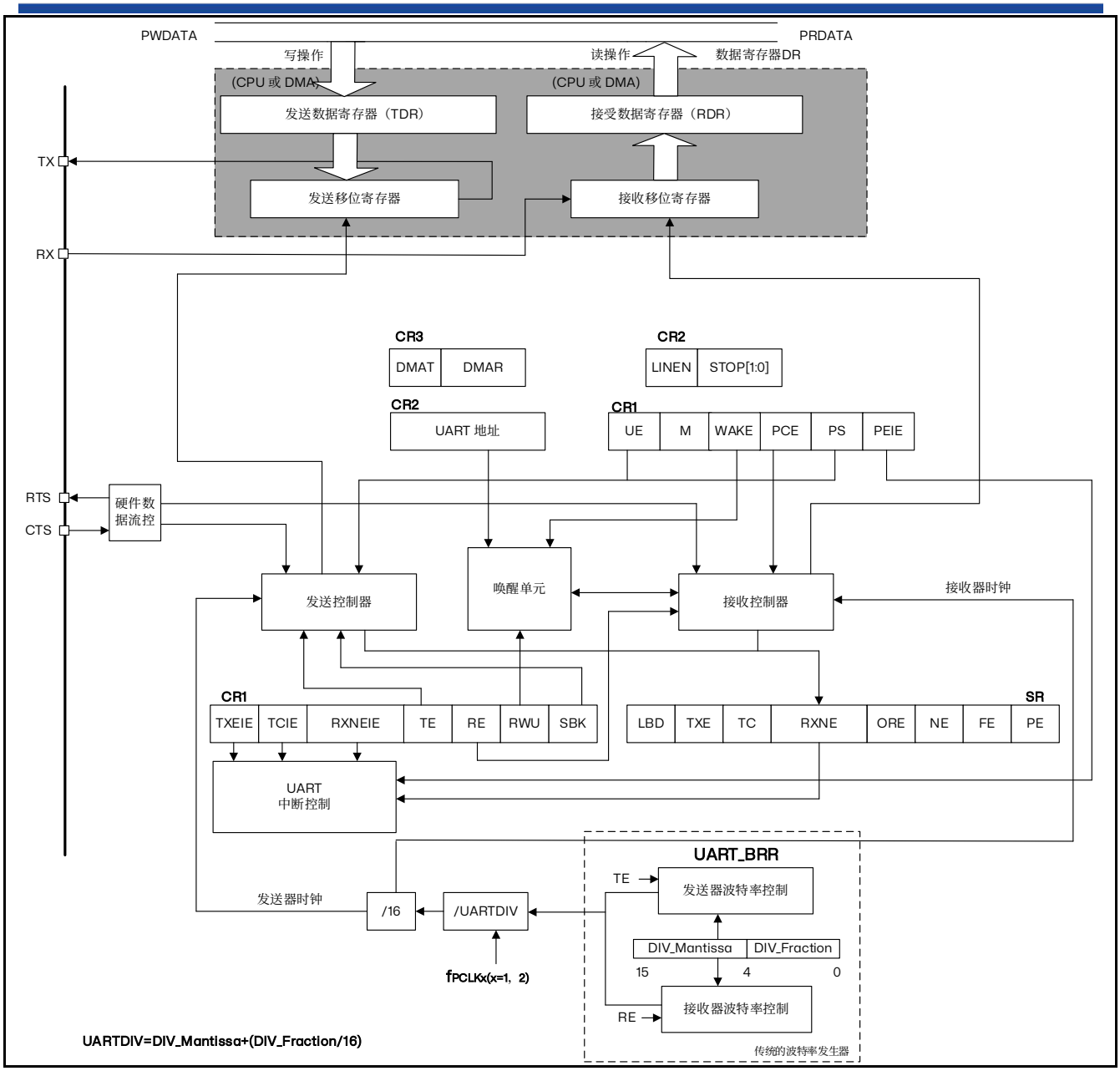



图 34.1 UART 框图

### 34.3.1 UART 特性描述

字长可以通过编程 UART_CR1 寄存器中的 M 位，选择成 8 或 9 位。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

断开符号被视为在一个帧周期内全部收到“0”（包括停止位期间，也是“0”）。在断开帧结束时，发送器再插入 1 或 2 个停止位（“1”）来应答起始位。

发送和接收由一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。

随后将有每个功能块的详细说明。

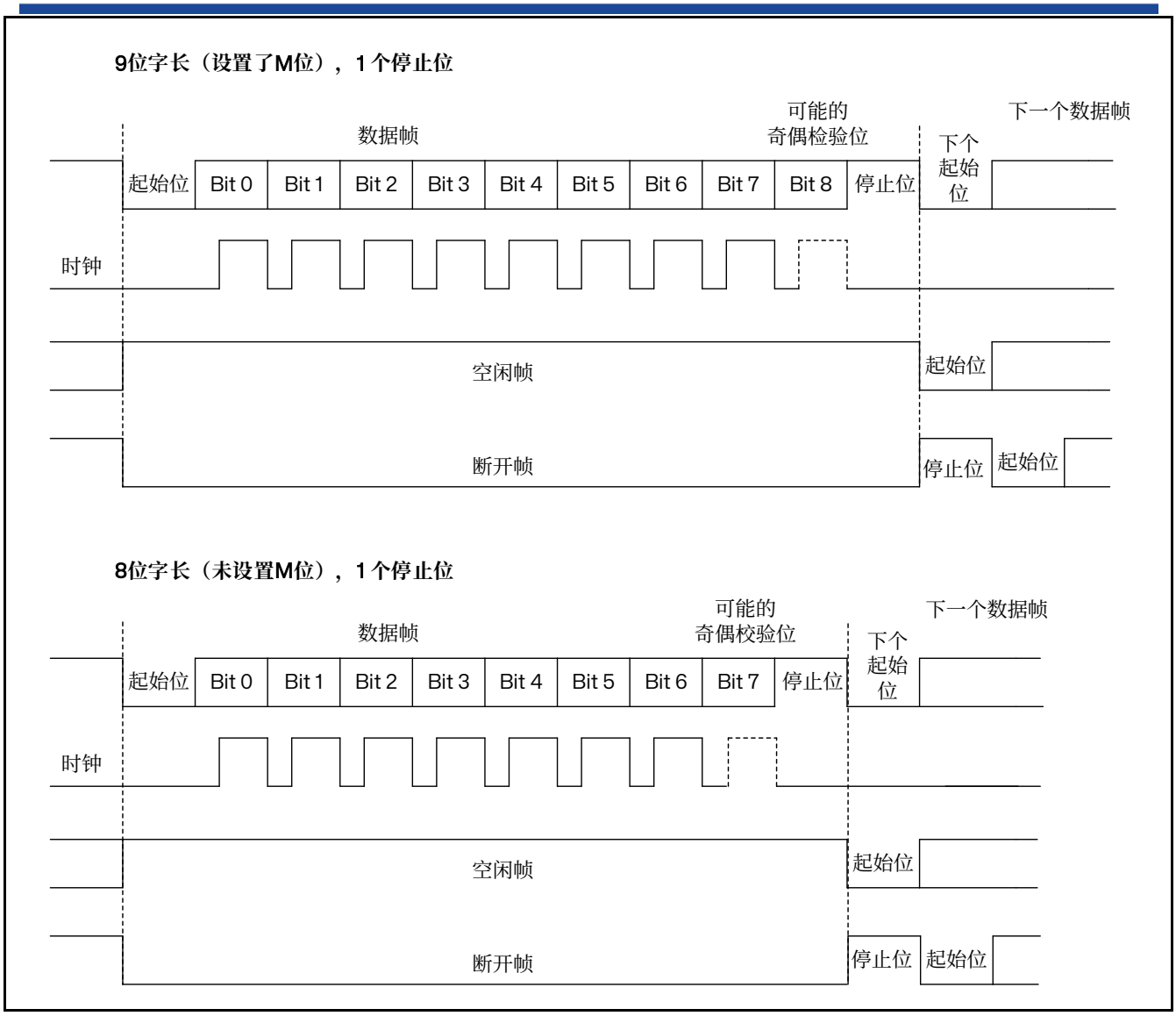



图 34.2 字长设置

### 34.3.2 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。

**字符发送**

在 UART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，UART_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。

UART 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。*

*注：TE 位被激活后将发送一个空闲帧。*

**可配置的停止位**

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 UART 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）。

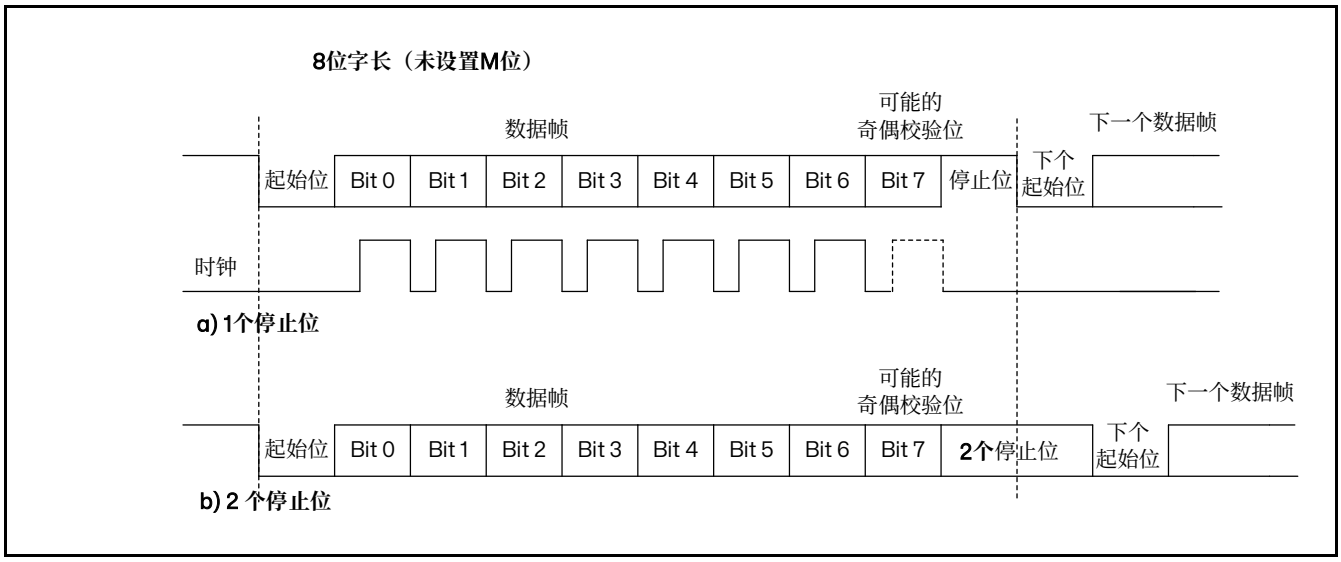



图 34.3 配置停止位

配置步骤：

1. 通过在 UART_CR1 寄存器上置位 UE 位来激活 UART
2. 编程 UART_CR1 的 M 位来定义字长。
3. 在 UART_CR2 中编程停止位的位数。
4. 如果采用多缓冲器通信，配置 UART_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 UART_BRR 寄存器选择要求的波特率。
6. 设置 UART_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 UART_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 UART_DR 寄存器中写入最后一个数据字后，要待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 UART 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。

**单字节通信**

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 UART_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 UART 正在发送数据，对 UART_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 UART 没有在发送数据，处于空闲状态，对 UART_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 UART_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

在 UART_DR 寄存器中写入了最后一个数据字后，在关闭 UART 模块之前或设置微控制器进入低功耗模式之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1. 读一次 UART_SR 寄存器；
2. 写一次 UART_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。*

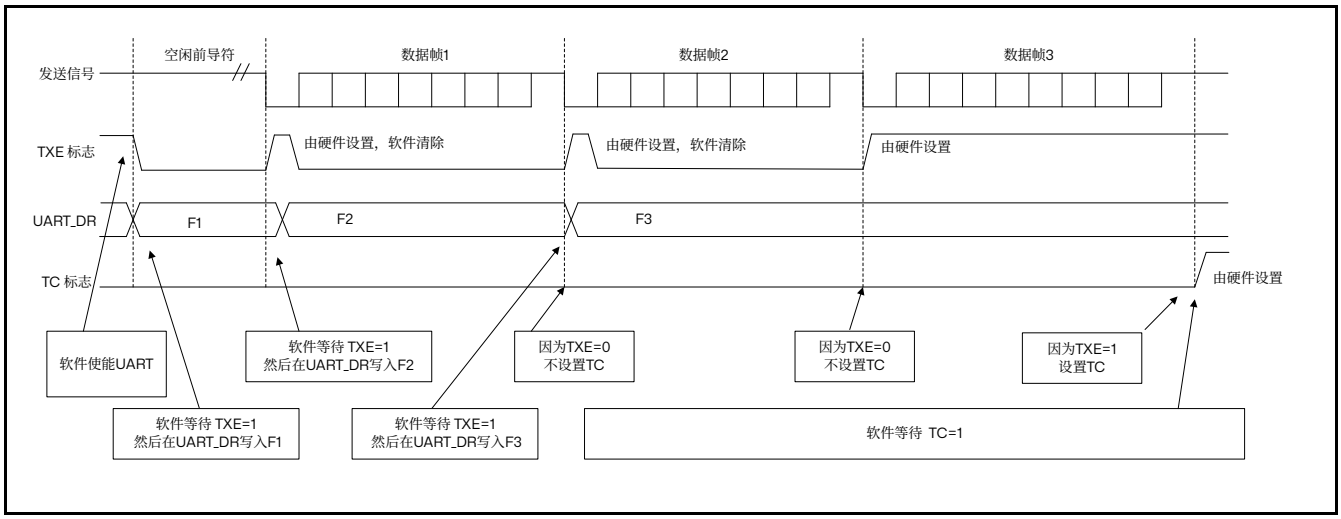



图 34.4 发送时 TC/TXE 的变化情况

**断开符号**

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。UART 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：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。*

**空闲符号**

置位 TE 将使得 UART 在第一个数据帧前发送一空闲帧。

### 34.3.3 接收器

UART 可以根据 UART_CR1 的 M 位接收 8 位或 9 位的数据字。

**起始位侦测**

在 UART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。
该序列为：1110X0X0X0000

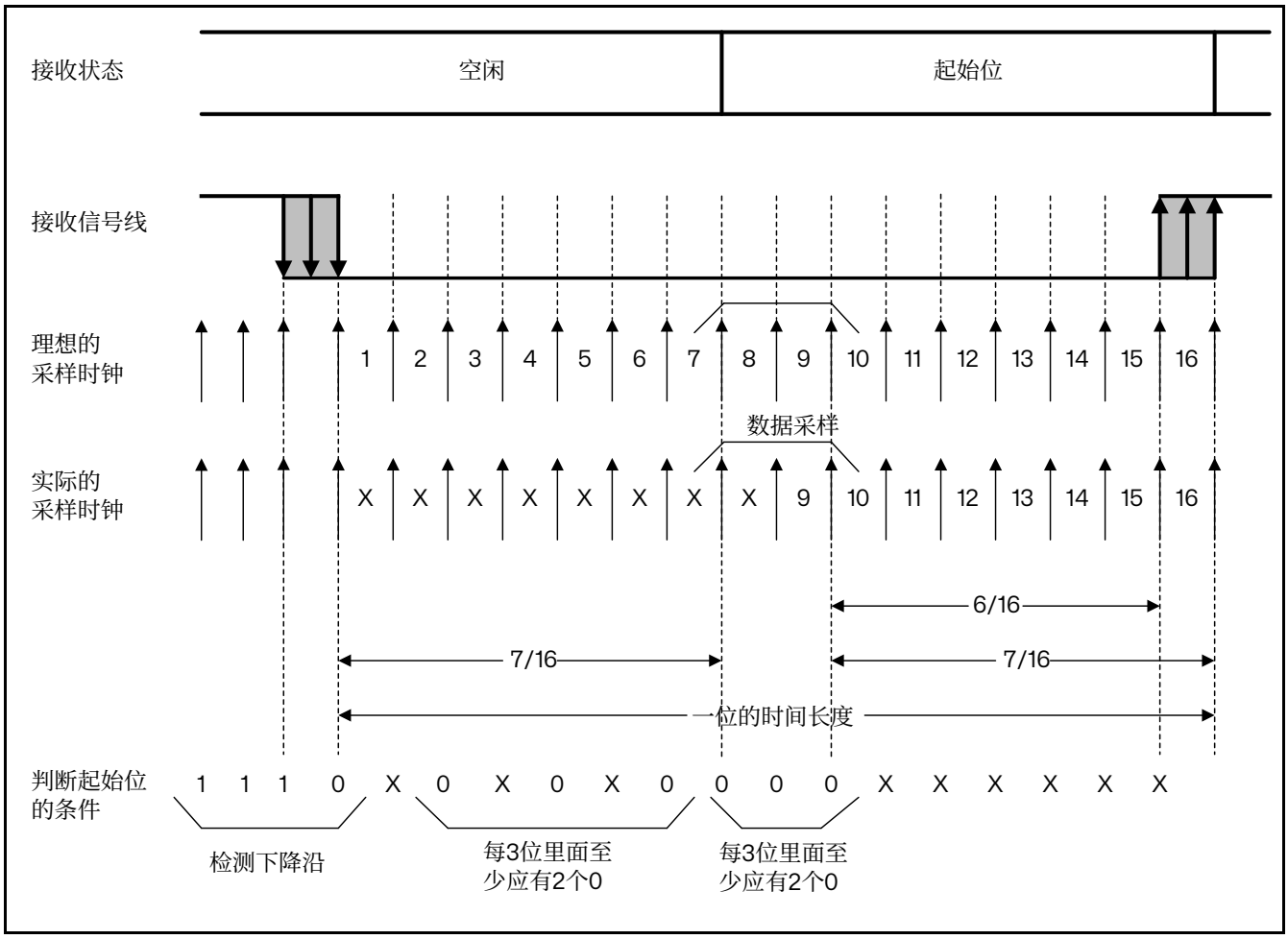



图 34.5 起始位侦测

*注：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。*

**字符接收**

在 UART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，UART_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 UART_CR1 寄存器的 UE 置 1 来激活 UART。
2. 编程 UART_CR1 的 M 位定义字长
3. 在 UART_CR2 中编写停止位的个数
4. 如果需多缓冲器通信，选择 UART_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 UART_BRR 选择希望的波特率。
6. 设置 UART_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 UART_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：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*

**断开符号**

当接收到一个断开帧时，UART 像处理帧错误一样处理它。

**溢出错误**

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 UART_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 UART_SR 和 UART_DR 寄存器的读操作，可复位 ORE 位

*注：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 UART_SR 寄存器读访问和 UART_DR 读访问之间）接收到新的数据，此种情况也可能发生。

**噪音错误**

使用过采样技术（同步模式除外），通过区别有效输入数据和噪音来进行数据恢复。

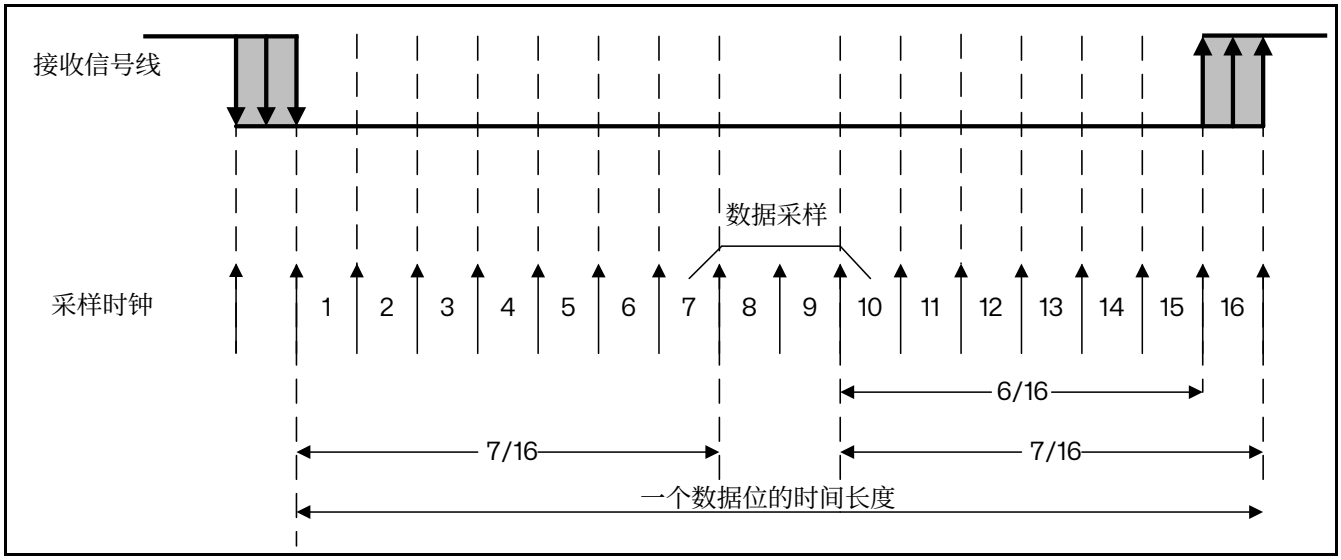



图 34.6 检测噪声的数据采样

表 34.1 检测噪声的数据采样

| 采样值 | NE 状态 | 接收的位 | 数据有效性 |
|---|---|---|---|
| 000 | 0 | 0 | 有效 |
| 001 | 1 | 0 | 无效 |
| 010 | 1 | 0 | 无效 |
| 011 | 1 | 1 | 无效 |
| 100 | 1 | 0 | 无效 |
| 101 | 1 | 1 | 无效 |
| 110 | 1 | 1 | 无效 |
| 111 | 0 | 1 | 有效 |

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志。
- 无效数据从移位寄存器传送到 UART_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 UART_CR3 寄存器中 EIE 位，将产生一个中断。

先读出 UART_SR，再读出 UART_DR 寄存器，将清除 NE 标志位

**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 UART_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 UART_CR3 寄存器中 EIE 位被置位的话，将产生中断。顺序执行对 UART_SR 和 UART_DR 寄存器的读操作，可复位 FE 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. **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. **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。

### 34.3.4 分数波特率的产生

接收器和发送器的波特率在 UARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx / Rx 波特率} = \frac{f_{ck}}{(16*UARTDIV)}$$

*这里的*$f_{ck}$ *是给外设的时钟（*PCLK1 *用于*UART2、3、4、5，PCLK2 *用于*UART1*）*

UARTDIV 是一个无符号的定点数。这 12 位的值设置在 UART_BRR 寄存器。

*注：在写入* UART_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。*

**如何从 UART_BRR 寄存器值得到 UARTDIV**

***示例1：***

如果 DIV_Mantissa = 0d27，DIV_Fraction = 0d12（UART_BRR=0x1BC），于是
尾数（UARTDIV）= 0d27
小数（UARTDIV）= 12/16 = 0d0.75
所以 UARTDIV = 0d27.75

***示例2：***

要求 UARTDIV = 0d25.62,
就有：
DIV_Fraction = 16*0d0.62 = 0d9.92
最接近的整数是：0d10 = 0x0A
DIV_Mantissa = 尾数（0d25.620）= 0d25 = 0x19
于是，UART_BRR = 0x19A，因此 UARTDIV = 0d25.625

***示例3：***

要求 UARTDIV = 0d50.99
就有：
DIV_Fraction = 16*0d0.99 = 0d15.84
最接近的整数是：0d16 = 0x10 => DIV_frac[3:0]溢出 => 尾数必须添加进位
DIV_Mantissa = 尾数（0d50.990 + 进位）= 0d51 = 0x33
于是：UART_BRR = 0x330，UARTDIV = 0d51.000

表 34.2 设置波特率时的误差计算

| 波特率 | | $f_{PCLK}$ = 96MHz | | | $f_{PCLK}$ = 192MHz | | |
|---|---|---|---|---|---|---|---|
| 序号 | Kbps | 实际 | 置于波特率寄存器中的值 | 误差% | 实际 | 置于波特率寄存器中的值 | 误差% |
| 1 | 2.4 | 2.4 | 2500 | 0% | 不可能 | 不可能 | 不可能 |
| 2 | 9.6 | 9.6 | 625 | 0% | 9.6 | 1250 | 0% |
| 3 | 19.2 | 19.2 | 312.5 | 0% | 19.2 | 625 | 0% |
| 4 | 38.4 | 38.4 | 156.25 | 0% | 38.4 | 312.5 | 0% |
| 5 | 115.2 | 115.2460 | 52.0625 | 0.04% | 115.1769 | 104.1875 | 0.02% |
| 6 | 230.4 | 230.2158 | 26.0625 | 0.08% | 230.4921 | 52.0625 | 0.04% |
| 7 | 460.8 | 461.5384 | 13 | 0.16% | 460.4316 | 26.0625 | 0.08% |
| 8 | 921.6 | 923.0769 | 6.5 | 0.16% | 923.0769 | 13 | 0.16% |
| 9 | 6000 | 6000 | 1 | 0% | 6000 | 2 | 0% |
| 10 | 12000 | 不可能 | 不可能 | 不可能 | 12000 | 1 | 0% |

*注：CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。*

### 34.3.5 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，它依赖于下述选择：

- 由 UART_CR1 寄存器的 M 位定义的 10 或 11 位字符长度
- 是否使用分数波特率产生

表 34.3 当 DIV_Fraction=0 时，UART 接收器的容忍度

| M 位 | 认为 NF 是错误 | 不认为 NF 是错误 |
|---|---|---|
| 0 | 3.75% | 4.375% |
| 1 | 3.41% | 3.97% |

表 34.4 当 DIV_Fraction!=0 时，UART 接收器的容忍度

| M 位 | 认为 NF 是错误 | 不认为 NF 是错误 |
|---|---|---|
| 0 | 3.33% | 3.88% |
| 1 | 3.03% | 3.53% |

*注：在特殊的情况下，即当收到的帧包含一些在 M=0 时，正好是 10 位（M=1 时是 11 位）的空闲帧，上面 2 个表格中的数据可能会有少许出入。*

### 34.3.6 多处理器通信

通过 UART 可以实现多处理器通信（将几个 UART 连在一个网络里）。例如某个 UART 设备可以是主，它的 TX 输出和其他 UART 从设备的 RX 输入相连接；UART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，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 UART 服务开销。

未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- UART_CR1 寄存器中 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。

设置 UART_CR1 寄存器中的 WAKE 位=1，UART 可根据地址标记检测来进入或退出静默模式。

- 如果 WAKE 位被设置：进行地址标记检测。

**地址标记（address mark）检测**

设置 WAKE=1 进入地址标记检测模式，在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 UART_CR2 寄存器的 ADD。

如果接收到的字节与它的编程地址不匹配时，UART 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 UART 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，UART 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(UART_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。

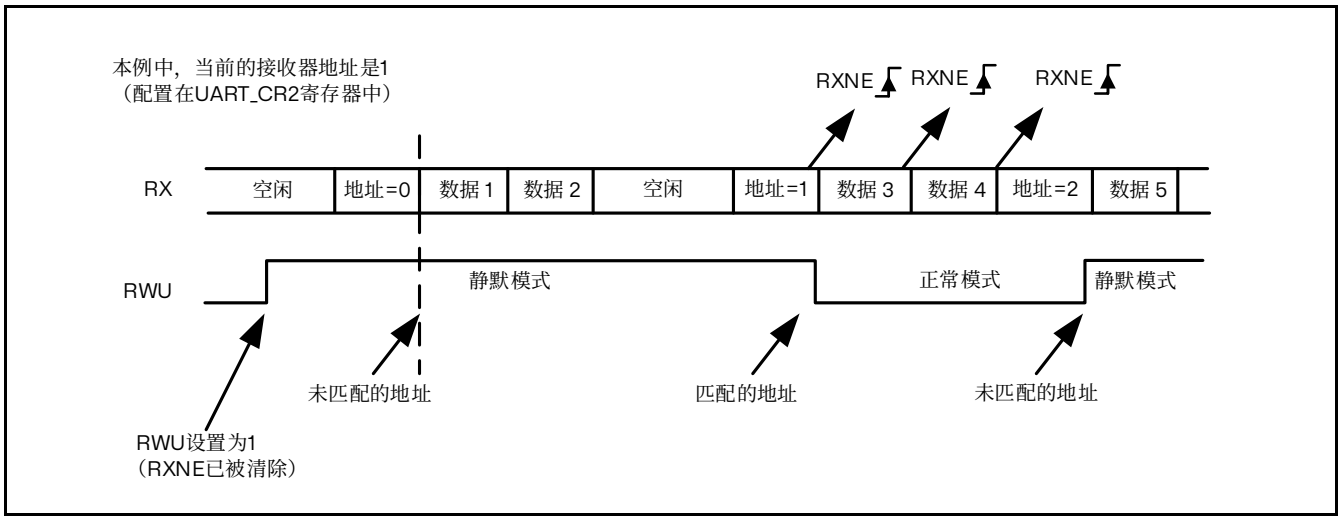



图 34.7 利用地址标记检测的静默模式

### 34.3.7 校验控制

设置 UART_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）。根据 M 位定义的帧长度，可能的 UART 帧格式列在下表中。

表 34.5 帧格式

| M 位 | PCE 位 | UART 帧 |
|---|---|---|
| 0 | 0 | 起始位 \| 8 位数据 \| 停止位 |
| 0 | 1 | 起始位 \| 7 位数据 \| 奇偶检验位 \| 停止位 |
| 1 | 0 | 起始位 \| 9 位数据 \| 停止位 |
| 1 | 1 | 起始位 \| 8 位数据 \| 奇偶检验位 \| 停止位 |

*注：在用地址标记唤醒设备时，地址的匹配只考虑到数据的*MSB *位，而不用关心校验位。(*MSB *是数据位中最后发出的，后面紧跟校验位或者停止位)*

**偶校验：**校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中“1”的个数为偶数。例如：数据=00110101，有 4 个“1”，如果选择偶校验（在 UART_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 UART_CR1 中的 PS＝1），校验位将是“1”。

**传输模式：**如果 UART_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，UART_SR 寄存器中的 PE 标志被置“1”，并且如果 UART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

### 34.3.8 LIN（局域互联网）模式

LIN 模式是通过设置 UART_CR2 寄存器的 LINEN 位选择。在 LIN 模式下，下列位必须保持为 0：

- UART_CR2 寄存器的 STOP[1:0]
- UART_CR3 寄存器的 HDSEL。

**LIN 发送**

UART 章节：发送器其中所描述的同样步骤适用于 LIN 主发送，但和正常 UART 发送有以下区别：

- 清零 M 位以配置 8 位字长
- 置位 LINEN 位以进入 LIN 模式。这时，置位 SBK 将发送 13 位“0”作为断开符号。然后发一位“1”，以允许对下一个开始位的检测。

**LIN 接收**

当 LIN 模式被使能时，断开符号检测电路被激活。该检测完全独立于 UART 接收器。断开符号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其间，数据帧还未完成，又插入了断开符号的发送。

当接收器被激活时（UART_CR1 的 RE=1），电路监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。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 UART_CR2 的 LBDL = 0）或 11 个（当 UART_CR2 的 LBDL = 1）连续位都是“0”，并且又跟着一个定界符，UART_SR 的 LBD 标志被设置。如果 LBDIE 位=1，中断产生。在确认断开符号前，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 UART 那样工作，不需要考虑检测断开符号。

如果 LIN 模式没有被激活（LINEN=0），接收器仍然正常工作于 UART 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，直到断开符号检测电路接收到一个“1”（这种情况发生于断开符号没有完整的发出来），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符号）。

下面两张图分别说明了断开符号检测器状态机的行为和断开符号标志的关系以及给出了一个断开帧的例子。

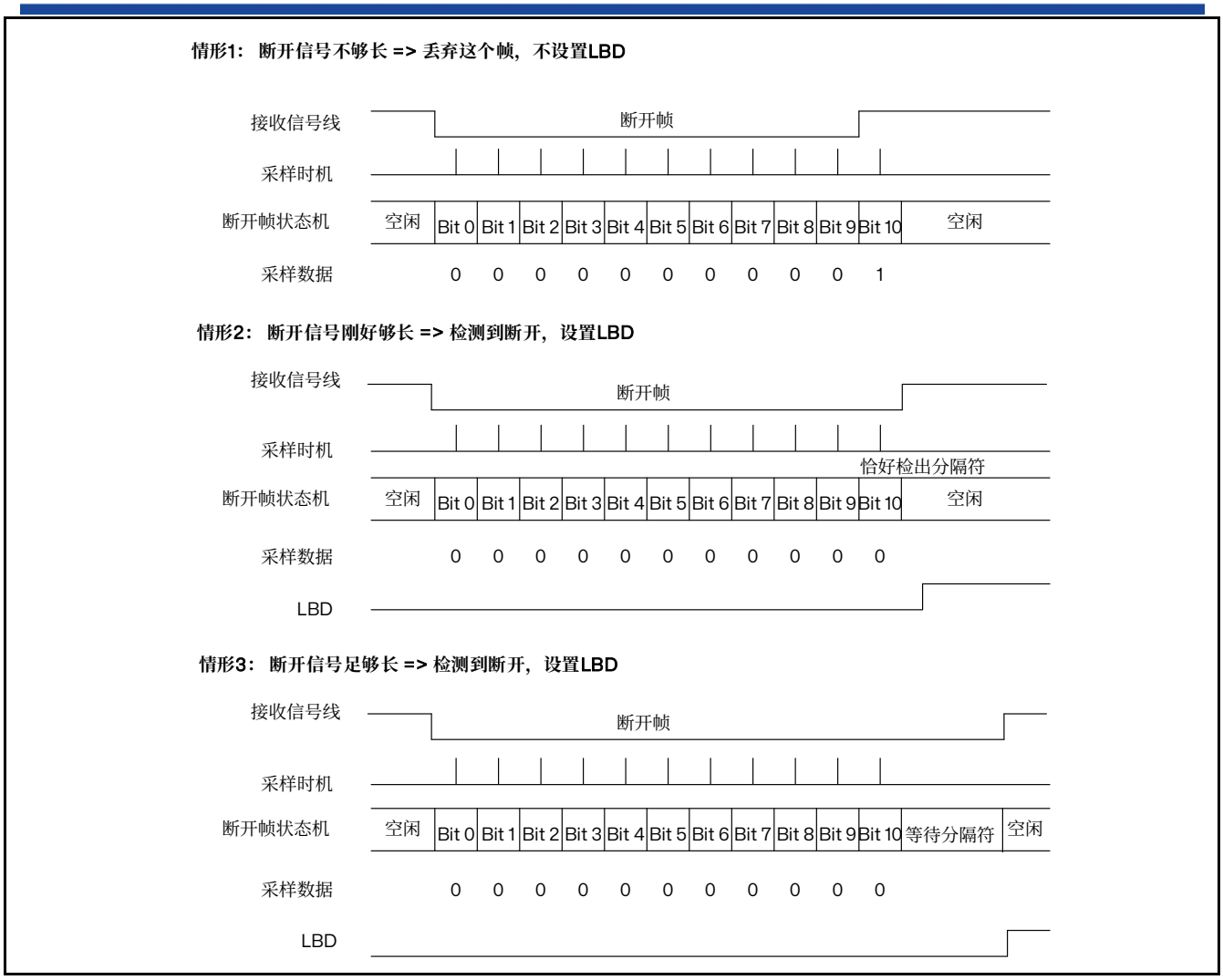



图 34.8 LIN 模式下的断开检测（11 位断开长度 – 设置了 LBDL 位）

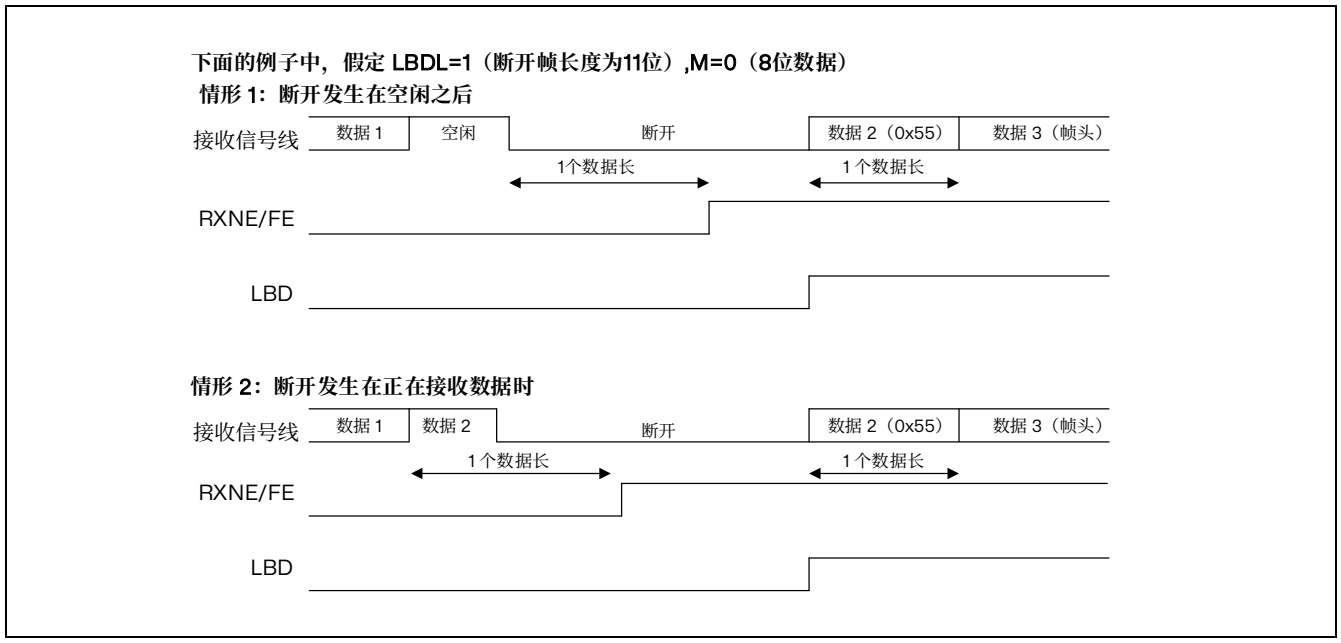



图 34.9 LIN 模式下的断开检测与帧错误的检测

### 34.3.9 单线半双工通信

单线半双工模式通过设置 UART_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：

- UART_CR2 寄存器的 LINEN 位

UART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位“HALF DUPLEX SEL”（UART_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，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 UART 驱动时，必须配置成悬空输入（或开漏的输出高）。

除此以外，通信与正常 UART 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。

### 34.3.10 利用 DMA 连续通信

UART 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：参考产品技术说明以确定是否可用* DMA *控制器。如果所用产品无* DMA *功能，应按照章节：发送器或者章节：接收器里所指定的方法是用* UART，*在* UART_SR *寄存器里，可以清零* TXE 或 RXNE 标志来实现连续通信。

**利用 DMA 发送**

用 DMA 进行发送，可以通过设置 UART_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 UART_DR 寄存器。为 UART 的发送分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 在 DMA 控制寄存器上将 UART_DR 寄存器地址配置成 DMA 传输的目的地址。在每个 TXE 事件后，数据将被传送到这个地址。
2. 在 DMA 控制寄存器上将存储器地址配置成 DMA 传输的源地址。在每个 TXE 事件后，将从此存储器区读出数据并传送到 UART_DR 寄存器。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。
5. 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 通过向 SR 寄存器中的 TC 位写入 0 来清除该位。
7. 在 DMA 寄存器上激活该通道。

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。在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 DMA_ISR 寄存器的 TCIF 标志；监视 UART_SR 寄存器的 TC 标志可以确认 UART 通信是否结束，这样可以在关闭 UART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等 TC=1。

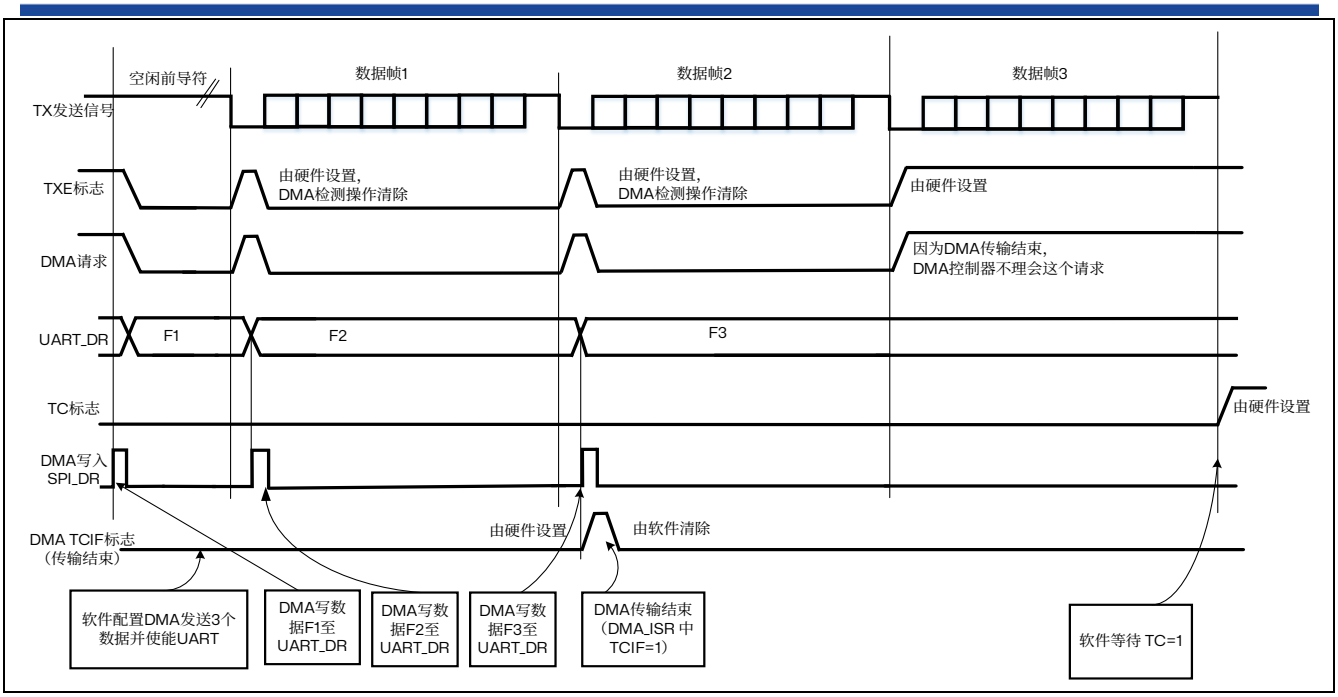



图 34.10 利用 DMA 发送

**利用 DMA 接收**

以通过设置 UART_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 控制器就就把数据从 UART_DR 寄存器传送到指定的 SRAM 区（参考 DMA 相关说明）。为 UART 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 UART_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 UART_DR 传输到此存储器区。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 控制寄存器上激活该通道。

当接收完成 DMA 控制器指定的传输量时，DMA 控制器在该 DMA 通道的中断矢量上产生一中断。

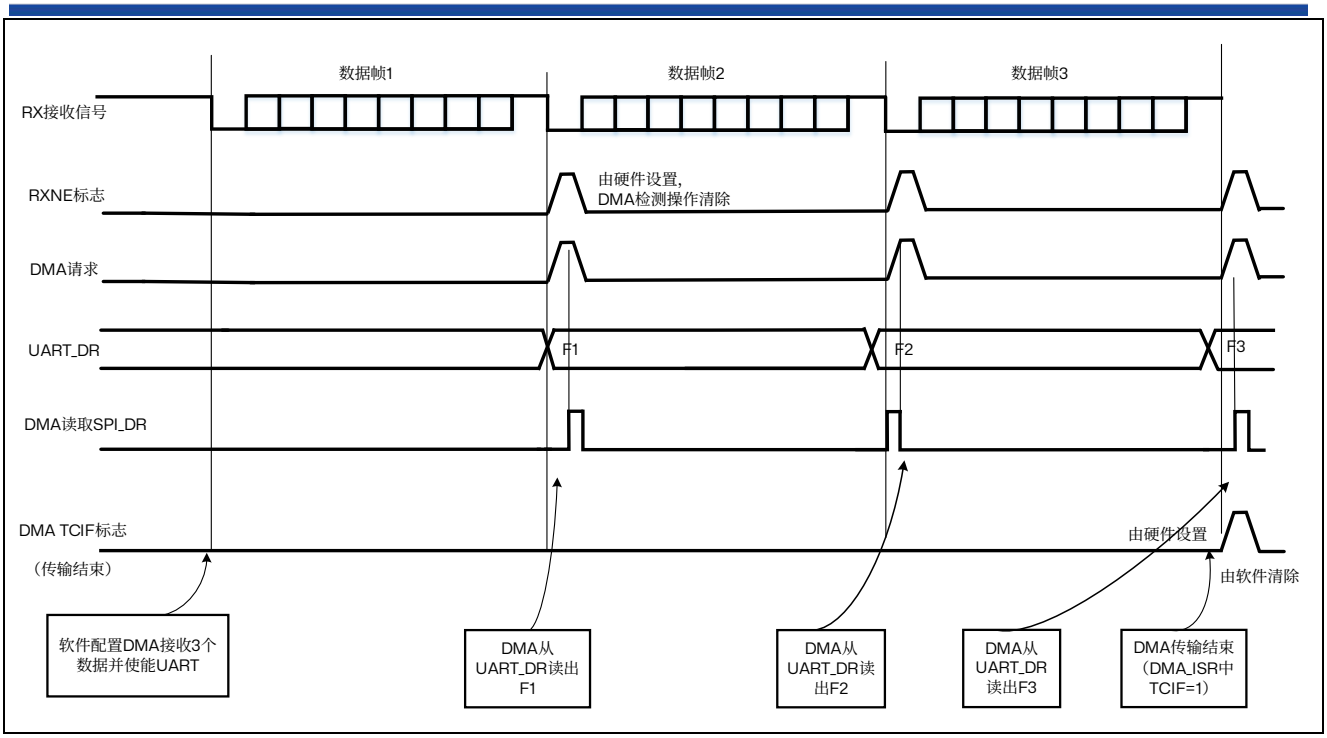



图 34.11 利用 DMA 接收

**多缓冲器通信中的错误标志和中断产生**

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和 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。

### 34.3.11 硬件流控制

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。

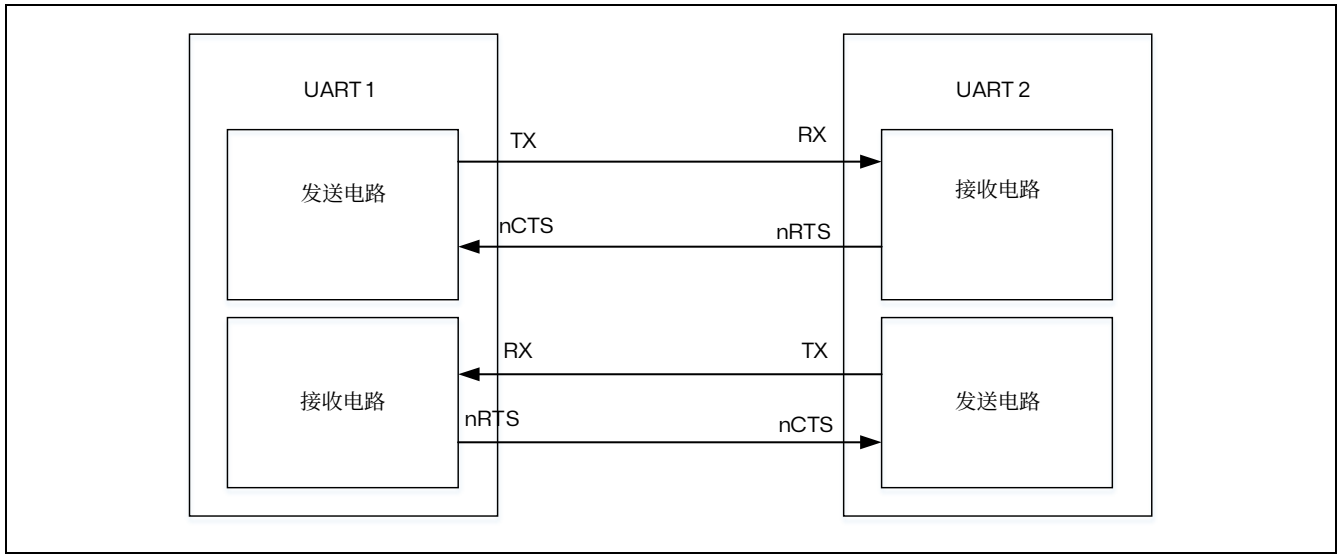



图 34.12 两个 UART 间的硬件流控制

通过将 UART_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 UART 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。

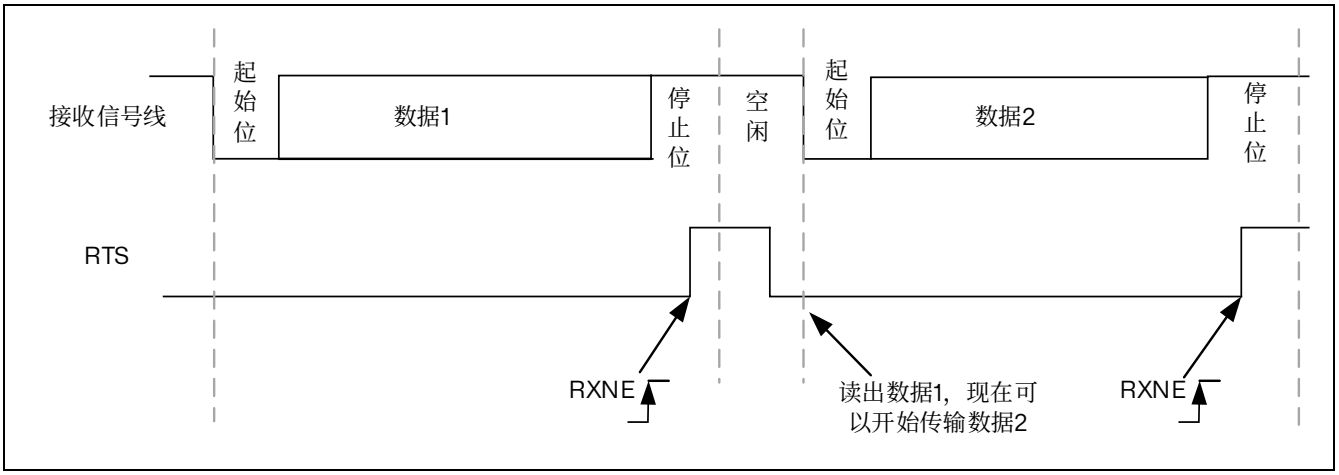



图 34.13 RTS 流控制

### 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 UART_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。

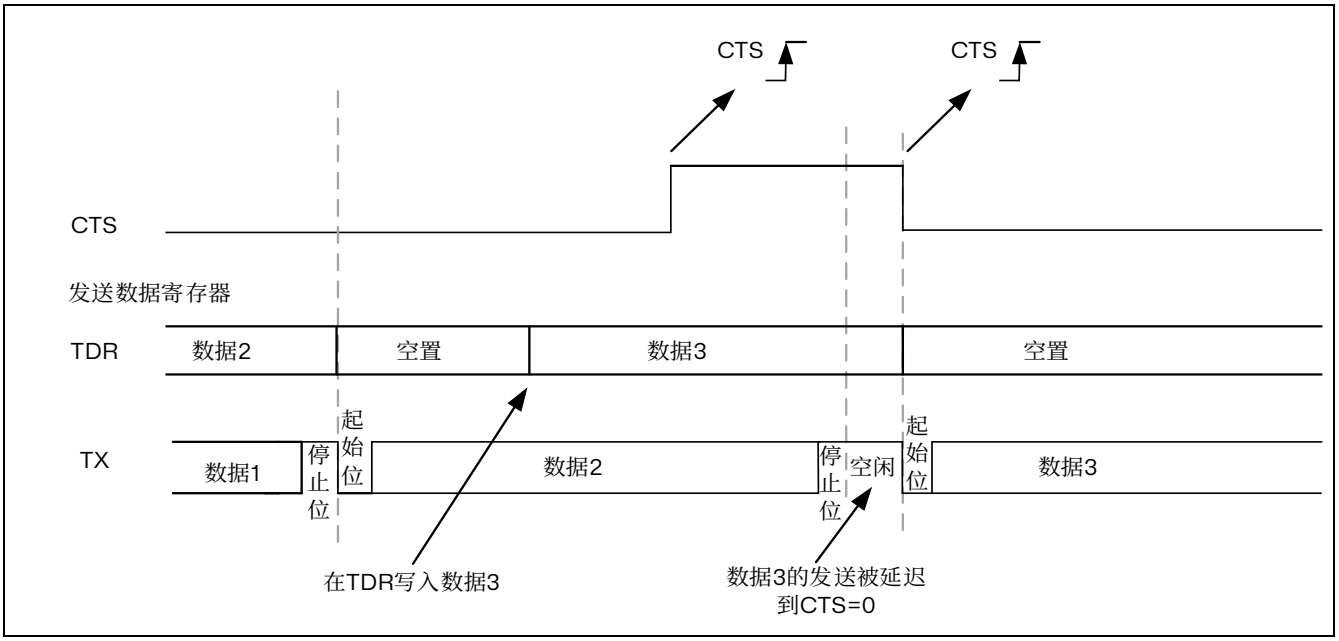



图 34.14 CTS 流控制

## 34.4 UART 中断请求

表 34.6 UART 中断请求

| 中断事件 | 事件标志 | 使能位 |
|---|---|---|
| 发送数据寄存器空 | TXE | TXEIE |
| CTS 标志 | CTS | CTSIE |
| 发送完成 | TC | TCIE |
| 接收数据就绪可读 | RXNE | RXNEIE |
| 检测到数据溢出 | ORE | |
| 奇偶检验错 | PE | PEIE |
| 断开标志 | LBD | LBDIE |
| 噪声标志，多缓冲通信中的溢出错误和帧错误 | NE 或 ORT 或 FE | EIE(1) |

1. 仅当使用 DMA 接收数据时，才使用这个标志位。

UART 的各种中断事件被连接到同一个中断向量（见下图），有以下各种中断事件：

- 发送期间：发送完成、清除发送、发送数据寄存器非空。
- 接收期间：溢出错误、接收数据寄存器非空、校验错误、LIN 断开符号检测、噪音标志（仅在多缓冲器通信）和帧错误（仅在多缓冲器通信）。

如果设置了对应的使能控制位，这些事件就可以产生各自的中断。

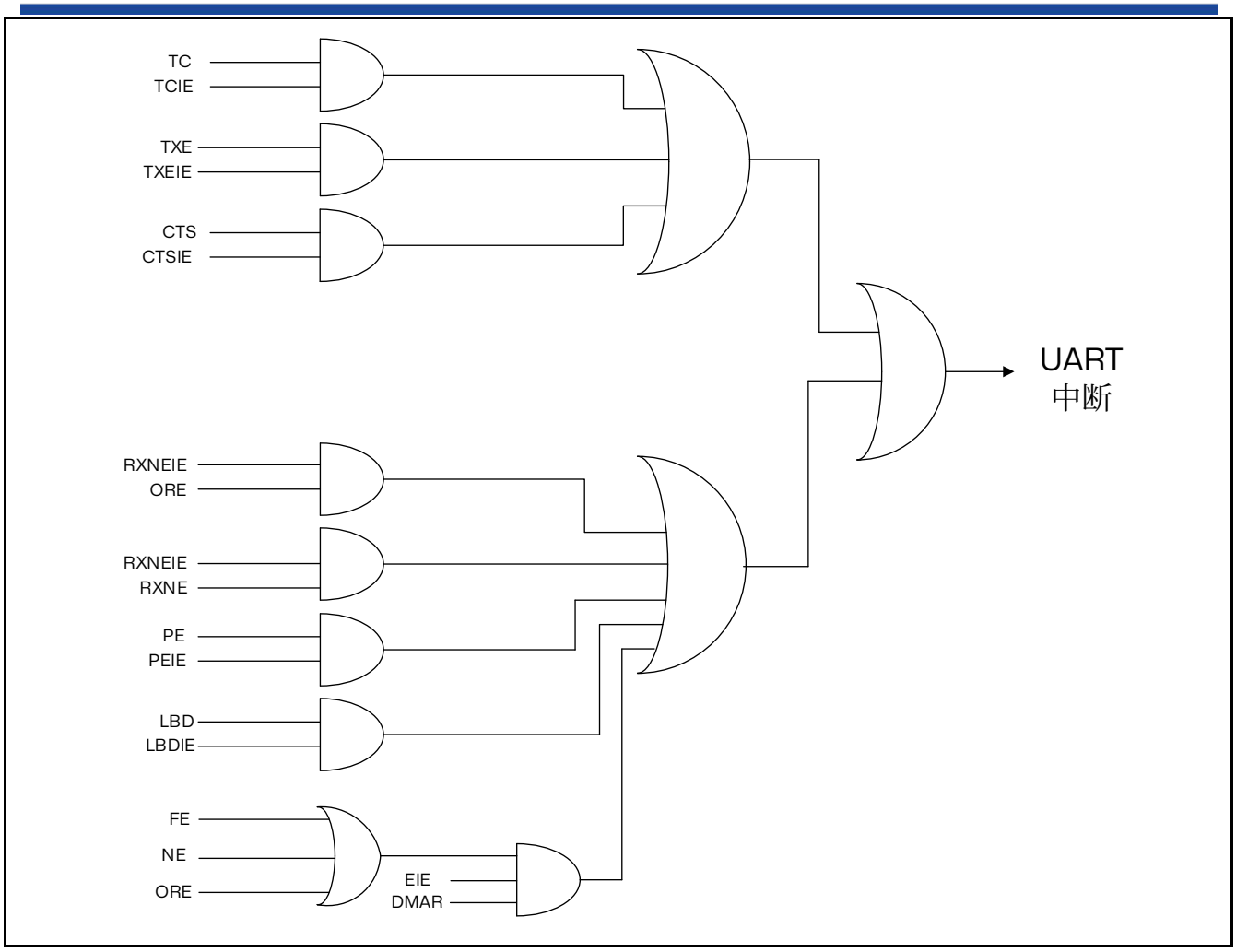



图 34.15 UART 中断映像图

## 34.5 UART 模式配置

表 34.7 UART 模式设置 (1)

| UART 模式 | UART1 | UART2 | UART3 | UART4 | UART5 |
|---|---|---|---|---|---|
| 异步模式 | × | × | × | × | × |
| 硬件流控制 | × | × | × | × | × |
| 多缓存通讯（DMA） | × | × | × | × | × |
| 多处理器通讯 | × | × | × | × | × |
| 半双工（单线模式） | × | × | × | × | × |
| LIN | × | × | × | × | × |

1. ×=支持；NA = 不支持该应用

## 34.6 UART 寄存器描述

在寄存器描述中使用的缩写列表请参考章节 2.1。
可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

### 34.6.1 状态寄存器（UART_SR）

地址偏移：0x00
复位值：0x0000 00C0

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CTS | LBD | TXE | TC | RXNE | Res | ORE | NE | FE | PE |
| | | | | | | rc_w0 | rc_w0 | r | rc_w0 | rc_w0 | | r | r | r | r |

Bit 31:10 保留，必须保持为复位值。

Bit 9 **CTS**：CTS 标志（CTS flag）
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 UART_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。

Bit 8 **LBD**：LIN 断开检测标志（LIN break detection flag）
当探测到 LIN 断开时，该位由硬件置“1”，由软件清“0”（向该位写 0）。如果 UART_CR2 中的 LBDIE = 1，则产生中断。
0：没有检测到 LIN 断开；
1：检测到 LIN 断开。

*注：若 LBDIE=1，当 LBD 为“1”时要产生中断。*

Bit 7 **TXE**：发送数据寄存器空（Transmit data register empty）
当 TDR 寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。如果 UART_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 UART_DR 的写操作，将该位清零。
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。

*注：单缓冲器传输中使用该位。*

Bit 6 **TC**：发送完成（Transmission complete）
当包含有数据的一帧发送完成后，并且 TXE=1 时，由硬件将该位置“1”。如果 UART_CR1 中的 TCIE 为“1”，则产生中断。由软件序列清除该位（先读 UART_SR，然后写入 UART_DR）。TC 位也可以通过写入“0”来清除，只有在多缓存通讯中才推荐这种清除程序。
0：发送还未完成；
1：发送完成。

Bit 5 **RXNE**：读数据寄存器非空（Read data register not empty）
当 RDR 移位寄存器中的数据被转移到 UART_DR 寄存器中，该位被硬件置位。如果 UART_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 UART_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程序。
0：数据没有收到；
1：收到数据，可以读出。

Bit 4 保留，必须保持为复位值。

Bit 3 **ORE**：过载错误（Overrun error）

当 RXNE 仍然是“1”的时候，当前被接收在移位寄存器中的数据，需要传送至 RDR 寄存器时，硬件将该位置位。如果 UART_CR1 中的 RXNEIE 为“1”的话，则产生中断。由软件序列将其清零（先读 UART_SR，然后读 UART_CR）。
0：没有过载错误；
1：检测到过载错误。

*注：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 EIE 位，在多缓冲器通信模式下，ORE 标志置位会产生中断的。*

Bit 2 **NE**：噪声错误标志（Noise error flag）

在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清零（先读 UART_SR，再读 UART_DR）。
0：没有检测到噪声；
1：检测到噪声。

*注：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。*

Bit 1 **FE**：帧错误（Framing error）

当检测到同步错位，过多的噪声或者检测到断开符，该位被硬件置位。由软件序列将其清零（先读 UART_SR，再读 UART_DR）。
0：没有检测到帧错误；
1：检测到帧错误或者 break 符。

*注：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。*

*注：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。*

Bit 0 **PE**：校验错误（Parity error）

在接收模式下，如果出现奇偶校验错误，硬件对该位置位。
由软件序列对其清零（依次读 UART_SR 和 UART_DR）。在清除 PE 位前，软件必须等待 RXNE 标志位被置“1”。如果 UART_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。

### 34.6.2 数据寄存器（UART_DR）

地址偏移：0x04

复位值：不确定

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

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

Bit 31:9 保留，必须保持为复位值。

Bit 8:0 **DR[8:0]：**数据值（Data value）

包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用（TDR），一个给接收用（RDR），该寄存器兼具读和写的功能。TDR 寄存器提供了内部总线和输出移位寄存器之间的并行接口。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。

当使能校验位（UART_CR1 中 PCE 位被置位）进行发送时，写到 MSB 的值（根据数据的长度不同，MSB 是第 7 位或者第 8 位）会被后来的校验位该取代。

当使能校验位进行接收时，读到的 MSB 位是接收到的校验位。

### 34.6.3 波特比率寄存器（UART_BRR）

*注：如果*TE *或*RE *被分别禁止，波特计数器停止计数*

地址偏移：0x08

复位值：0x0000 0000

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

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

Bit 31:16 保留，必须保持为复位值。

Bit 15:4 **DIV_Mantissa[11:0]：**UARTDIV 的整数部分

这 12 位定义了 UART 分频器除法系数（UARTDIV）的整数部分。

Bit 3:0 **DIV_Fraction[3:0]：**UARTDIV 的小数部分

这 4 位定义了 UART 分频器除法系数（UARTDIV）的小数部分。

### 34.6.4 控制寄存器 1（UART_CR1）

地址偏移：0x0C

复位值：0x0000 0000

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | UE | M | WAKE | PCE | PS | PEIE | TXEIE | TCIE | RXNEIE | Res | TE | RE | RWU | SBK |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw |

Bit 31:14 保留，必须保持为复位值。

Bit 13 **UE**：UART 使能（UART enable）

当该位被清零，在当前字节传输完成后 UART 的分频器和输出停止工作，以减少功耗。该位由软件设置和清零。

0：UART 分频器和输出被禁止；

1：UART 模块使能。

Bit 12 **M**：字长（Word length）

该位定义了数据字的长度，由软件对其设置和清零

0：一个起始位，8 个数据位，n 个停止位；

1：一个起始位，9 个数据位，n 个停止位。

*注：在数据传输过程中（发送或者接收时），不能修改这个位。*

*注：当使能校验控制*PCE *时，校验位的插入会导致数据帧长度改变。数据帧长度根据*M *位定义，可能的*UART *帧格式请参考章节*35.3.7 *校验控制中的帧格式表格。*

Bit 11 **WAKE**：唤醒的方法（Wakeup method）

这位决定了把 UART 唤醒的方法，由软件对该位设置和清零。

0：保留；

1：被地址标记唤醒。

Bit 10 **PCE**：校验控制使能（Parity control enable）

用该位选择是否进行硬件校验控制（对于发送来说就是校验位的产生；对于接收来说就是校验位的检测）。当使能了该位，在发送数据的最高位（如果 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。

Bit 9 **PS**：校验选择（Parity selection）

当校验控制使能后，该位用来选择是采用偶校验还是奇校验。软件对它置“1”或清“0”。当前字节传输完成后，该选择生效。

0：偶校验；

1：奇校验。

Bit 8 **PEIE**：PE 中断使能（PE interrupt enable）

该位由软件设置或清除。

0：禁止产生中断；

1：当 UART_SR 中的 PE 为“1”时，产生 UART 中断。

Bit 7 **TXEIE**：发送缓冲区空中断使能（TXE interrupt enable）

该位由软件设置或清除。

0：禁止产生中断；

1：当 UART_SR 中的 TXE 为“1”时，产生 UART 中断。

Bit 6 **TCIE**：发送完成中断使能（Transmission complete interrupt enable）

该位由软件设置或清除。

0：禁止产生中断；

1：当 UART_SR 中的 TC 为“1”时，产生 UART 中断。

Bit 5 **RXNEIE**：接收缓冲区非空中断使能（RXNE interrupt enable）

该位由软件设置或清除。

0：禁止产生中断；

1：当 UART_SR 中的 ORE 或者 RXNE 为“1”时，产生 UART 中断。

Bit 4 保留，必须保持为复位值。

Bit 3 **TE**：发送使能（Transmitter enable）

该位使能发送器。该位由软件设置或清除。

0：禁止发送；

1：使能发送。

*注：在数据传输过程中，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”（空闲总线）。*

*注：当 TE 被设置后，在真正发送开始之前，有一个比特时间的延迟。*

Bit 2 **RE**：接收使能（Receiver enable）

该位由软件设置或清除。

0：禁止接收；

1：使能接收，并开始搜寻 RX 引脚上的起始位。

Bit 1 **RWU**：接收唤醒（Receiver wakeup）

该位用来决定是否把 UART 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。

*注：在把 UART 置于静默模式（设置 RWU 位）之前，UART 要已经先接收了一个数据字节。*

*注：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。*

Bit 0 **SBK**：发送断开帧（Send break）

使用该位来发送断开字符。该位可以由软件设置或清除。操作过程应该是软件设置位它，然后在断开帧的停止位时，由硬件将该位复位。

0：没有发送断开字符；

1：将要发送断开字符。

### 34.6.5 控制寄存器 2（UART_CR2）

地址偏移：0x10

复位值：0x0000 0000

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res | LINEN | STOP[1:0] | | Reserved | | | | | LBDIE | LBDL | Res | ADD[3:0] | | | |
| | rw | rw | rw | | | | | | rw | rw | | rw | rw | rw | rw |

Bit 31:15 保留，必须保持为复位值。

Bit 14 **LINEN**：LIN 模式使能（LIN mode enable）

该位由软件设置或清除。

0：禁止 LIN 模式；

1：使能 LIN 模式。

Bit 13:12 **STOP**：停止位（STOP Bit）

这 2 位用来设置停止位的位数

00：1 个停止位；

10：2 个停止位；

其他：保留

Bit 11:7 保留，必须保持为复位值。

Bit 6 **LBDIE**：LIN 断开符检测中断使能（LIN break detection interrupt enable）

断开符中断屏蔽（使用断开分隔符来检测断开符）

0：禁止中断；

1：只要 UART_SR 寄存器中的 LBD 为“1”就产生中断。

Bit 5 **LBDL**：LIN 断开符检测长度（LIN break detection length）

该位用来选择是 11 位还是 10 位的断开符检测

0：10 位的断开符检测；

1：11 位的断开符检测。

Bit 4 保留，必须保持为复位值。

Bit 3:0 **ADD[3:0]**：本设备的 UART 节点地址

该位域给出本设备 UART 节点的地址。

这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个 UART 设备。

### 34.6.6 控制寄存器 3（UART_CR3）

地址偏移：0x14

复位值：0x0000 0000

| 31 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | CTSIE | CTSE | RTSE | DMAT | DMAR | Reserved | | HDSEL | Reserved | | EIE |
| | | | | | rw | rw | rw | rw | rw | | | rw | | | rw |

Bit 31:11 保留，必须保持为复位值。

Bit 10 **CTSIE**：CTS 中断使能（CTS interrupt enable）

0：禁止中断；

1：UART_SR 寄存器中的 CTS 为“1”时产生中断。

Bit 9 **CTSE**：CTS 使能（CTS enable）

0：禁止 CTS 硬件流控制；

1：CTS 模式使能，只有 nCTS 输入信号有效（拉成低电平）时才能发送数据。如果在数据传输的过程中，nCTS 信号变成无效，那么发完这个数据后，传输就停止下来。如果当 nCTS 为无效时，往数据寄存器里写数据，则要等到 nCTS 有效时才会发送这个数据。

Bit 8 **RTSE**：RTS 使能（RTS enable）

0：禁止 RTS 硬件流控制；

1：RTS 中断使能，只有接收缓冲区内有空余的空间时才请求下一个数据。当前数据发送完成后，发送操作就需要暂停下来。如果可以接收数据了，将 nRTS 输出置为有效（拉至低电平）。

Bit 7 **DMAT**：DMA 使能发送（DMA enable transmitter）

该位由软件设置或清除。

0：禁止发送时的 DMA 模式。

1：使能发送时的 DMA 模式；

Bit 6 **DMAR**：DMA 使能接收（DMA enable receiver）

该位由软件设置或清除。

0：禁止接收时的 DMA 模式。

1：使能接收时的 DMA 模式；

Bit 5:4 保留，必须保持为复位值。

Bit 3 **HDSEL**：半双工选择（Half-duplex selection）

选择单线半双工模式

0：不选择半双工模式；

1：选择半双工模式。

Bit 2:1 保留，必须保持为复位值。

Bit 0 **EIE**：错误中断使能（Error interrupt enable）

0：禁止中断；

1：只要 UART_CR3 中的 DMAR=1，并且 UART_SR 中的 FE=1，或者 ORE=1，或者 NE=1，则产生中断。

# 35 调试支持（DBG）

### 35.1.1 对定时器、RTC、看门狗和 I2C 的调试支持

断点期间，必须选择定时器、RTC 和看门狗的计数器的行为方式：

- 在产生断点时，计数器继续计数。例如，当 PWM 控制电机时，通常需要这种方式。
- 在产生断点时，计数器停止计数。用于看门狗时需要这种方式。

对于 I2C，用户可以选择在断点期间阻止 SMBUS 超时。

### 35.1.2 MCU 调试配置写/器件 ID 寄存器（IDCODE_CRW）

地址：0xE004 2000

访问：仅支持 32 位访问。

相同的寄存器可以用于器件 ID 读取（本节）或 MCU 调试配置写入（下节）。此寄存器在读取和写入下的功能均不同。可以独立操作读取和写入（例如，读取该寄存器获得 IDCODE，写入该寄存器以配置 MCU 在不同低功耗模式下的功能选项）。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDCODE[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 |
| IDCODE[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

**执行读取操作**

Bit 31:0 **IDCODE[31:0]**：器件 ID 代码

默认器件 ID 为：0x20036524。

### 35.1.3 MCU 调试配置写/器件 ID 寄存器（IDCODE_CRW）

地址：0xE004 2000
上电复位：0x0000 0000
系统复位：无影响
访问：仅支持 32 位访问。

相同的寄存器可以用于 MCU 调试配置写入（本节）或器件 ID 读取（上节）。此寄存器在读取和写入下的功能均不同。可以独立操作读取和写入（例如，读取该寄存器获得 IDCODE，写入该寄存器以配置 MCU 在不同低功耗模式下的功能选项）。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | DBG_STANDBY | DBG_STOP | DBG_SLEEP |
| | | | | | | | | | | | | | w | w | w |

**执行写入操作**

Bit 31:3 保留，必须保持复位值。

Bit 2 **DBG_STANDBY：**调试待机模式

0：（FCLK=关闭，HCLK=关闭）将整个数字部分断电。从软件角度来看，退出待机模式相当于复位（指示 MCU 从待机状态恢复的几个状态位除外）

1：（CLK=开启，HCLK=开启）在这种情况下，数字部分不断电，FCLK 和 HCLK 由保持激活状态的内部 RC 振荡器提供。此外，MCU 会在待机模式期间产生系统复位，这样退出待机模式相当于复位

Bit 1 **DBG_STOP：**调试停止模式

0：（FCLK=关闭，HCLK=关闭）在停机模式下，时钟控制器禁止所有时钟（包括 HCLK 和 FCLK）。退出停机模式时，时钟配置与复位后的时钟配置相同（CPU 时钟由 16MHz 内部 RC 振荡器（HSI）提供）。因此，软件必须重新编程时钟控制器以使能 PLL 和晶振等。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入停机模式时，FCLK 和 HCLK 由在停机模式下仍保持激活状态的内部 RC 振荡器提供。退出停机模式时，软件必须重新编程时钟控制器以使能 PLL 和晶振等（操作步骤与在 DBG_STOP=0 时相同）

Bit 0 **DBG_SLEEP：**调试睡眠模式

0：（FCLK=开启，HCLK=关闭）在睡眠模式下，FCLK 由原先配置好的系统时钟驱动。在睡眠模式下，时钟控制器配置不复位，保持先前设置的状态。因此，退出睡眠模式时，软件不需要重新配置时钟控制器。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入睡眠模式时，将为 HCLK 和 FCLK 馈送相同的时钟（软件先前配置的系统时钟）。

### 35.1.4 MCU 调试配置读寄存器（DBGMCU_CRR）

地址：0xE004 2004
上电复位：0x0000 0000
系统复位：无影响
访问：仅支持 32 位访问，只读。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | DBG_STANDBY | DBG_STOP | DBG_SLEEP |
| | | | | | | | | | | | | | r | r | r |

Bit 31:3 保留，必须保持复位值。

Bit 2 **DBG_STANDBY**：调试待机模式

0：（FCLK=关闭，HCLK=关闭）将整个数字部分断电。从软件角度来看，退出待机模式相当于复位（指示 MCU 从待机状态恢复的几个状态位除外）

1：（FCLK=开启，HCLK=开启）在这种情况下，数字部分不断电，FCLK 和 HCLK 由保持激活状态的内部 RC 振荡器提供。此外，MCU 会在待机模式期间产生系统复位，这样退出待机模式相当于复位

Bit 1 **DBG_STOP**：调试停止模式

0：（FCLK=关闭，HCLK=关闭）在停机模式下，时钟控制器禁止所有时钟（包括 HCLK 和 FCLK）。退出停机模式时，时钟配置与复位后的时钟配置相同（CPU 时钟由 16MHz 内部 RC 振荡器（HSI）提供）。因此，软件必须重新编程时钟控制器以使能 PLL 和晶振等。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入停机模式时，FCLK 和 HCLK 由在停机模式下仍保持激活状态的内部 RC 振荡器提供。退出停机模式时，软件必须重新编程时钟控制器以使能 PLL 和晶振等（操作步骤与在 DBG_STOP=0 时相同）

Bit 0 **DBG_SLEEP**：调试睡眠模式

0：（FCLK=开启，HCLK=关闭）在睡眠模式下，FCLK 由原先配置好的系统时钟驱动。在睡眠模式下，时钟控制器配置不复位，保持先前设置的状态。因此，退出睡眠模式时，软件不需要重新配置时钟控制器。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入睡眠模式时，将为 HCLK 和 FCLK 馈送相同的时钟（软件先前配置的系统时钟）。

### 35.1.5 MCU APB1 调试冻结寄存器（DBGMCU_APB1_FZR1）

地址：0xE004 2008
上电复位（POR）：0x0000 0000
系统复位：无影响
访问：仅支持 32 位访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | DBG_I2C2_STOP | DBG_I2C1_STOP | Reserved | | | | |
| | | | | | | | | | rw | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | DBG_IWDG_STOP | DBG_WWDG_STOP | DBG_RTC_STOP | Reserved | | | | DBG_TIM7_STOP | DBG_TIM6_STOP | DBG_TIM5_STOP | DBG_TIM4_STOP | DBG_TIM3_STOP | DBG_TIM2_STOP |
| | | | rw | rw | rw | | | | | rw | rw | rw | rw | rw | rw |

Bit 31:23 保留，必须保持复位值。

Bit 22 **DBG_I2C2_STOP**：内核停止时停止 I2C2 SMBUS 超时模式

0：行为方式与正常模式下相同

1：冻结 I2C2 SMBUS 超时

Bit 22 **DBG_I2C1_STOP**：内核停止时停止 I2C1 SMBUS 超时模式

0：行为方式与正常模式下相同

1：冻结 I2C1 SMBUS 超时

Bit 20:13 保留，必须保持复位值。

Bit 12 **DBG_IWDG_STOP**：内核停止时停止独立看门狗

0：即使内核停止，独立看门狗计数器时钟仍继续工作

1：内核停止时，独立看门狗计数器时钟停止

Bit 11 **DBG_WWDG_STOP**：内核停止时停止窗口看门狗

0：即使内核停止，窗口看门狗计数器时钟仍继续工作

1：内核停止时，窗口看门狗计数器时钟停止

Bit 10 **DBG_RTC_STOP**：内核停止时停止 RTC

0：即使内核停止，RTC 计数器时钟仍继续工作

1：内核停止时停止相关定时器计数器的时钟

Bit 9:6 保留，必须保持复位值。

Bit 5 **DBG_TIM7_STOP**：内核停止时 TIM7 计数器停止

0：即使内核停止，仍然馈送 TIM7 计数器的时钟

1：内核停止时停止 TIM7 计数器的时钟

Bit 4 **DBG_TIM6_STOP**：内核停止时 TIM6 计数器停止

0：即使内核停止，仍然馈送 TIM6 计数器的时钟

1：内核停止时停止 TIM6 计数器的时钟

Bit 3 **DBG_TIM5_STOP**：内核停止时 TIM5 计数器停止

0：即使内核停止，仍然馈送 TIM5 计数器的时钟

1：内核停止时停止 TIM5 计数器的时钟

Bit 2 **DBG_TIM4_STOP**：内核停止时 TIM4 计数器停止

0：即使内核停止，仍然馈送 TIM4 计数器的时钟

1：内核停止时停止 TIM4 计数器的时钟

Bit 1 **DBG_TIM3_STOP**：内核停止时 TIM3 计数器停止

0：即使内核停止，仍然馈送 TIM3 计数器的时钟

1：内核停止时停止 TIM3 计数器的时钟

Bit 0 **DBG_TIM2_STOP**：内核停止时 TIM2 计数器停止

0：即使内核停止，仍然馈送 TIM2 计数器的时钟
1：内核停止时停止 TIM2 计数器的时钟

### 35.1.6 MCU APB2 调试冻结寄存器（DBGMCU_APB2_FZR）

地址：0xE004 2010
上电复位（POR）：0x0000 0000
系统复位：无影响
访问：仅支持 32 位访问

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | DBG_HRTIM_STOP | Reserved | | | | | | | DBG_TIM17_STOP | DBG_TIM16_STOP | DBG_TIM15_STOP |
| | | | | | rw | | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DBG_TIM8_STOP | Res | DBG_TIM1_STOP | Reserved | | | | | | | | | | |
| | | rw | | rw | | | | | | | | | | | |

Bit 31:27 保留，必须保持复位值。

Bit 26 **DBG_HRTIM_STOP**：内核停止时 HRTIM 计数器停止
0：即使内核停止，仍然馈送 HRTIM 计数器的时钟
1：内核停止时停止 HRTIM 计数器的时钟

Bit 25:19 保留，必须保持复位值。

Bit 18 **DBG_TIM17_STOP**：内核停止时 TIM17 计数器停止
0：即使内核停止，仍然馈送 TIM17 计数器的时钟
1：内核停止时停止 TIM17 计数器的时钟

Bit 17 **DBG_TIM16_STOP**：内核停止时 TIM16 计数器停止
0：即使内核停止，仍然馈送 TIM16 计数器的时钟
1：内核停止时停止 TIM16 计数器的时钟

Bit 16 **DBG_TIM15_STOP**：内核停止时 TIM15 计数器停止
0：即使内核停止，仍然馈送 TIM15 计数器的时钟
1：内核停止时停止 TIM15 计数器的时钟

Bit 15:14 保留，必须保持复位值。

Bit 13 **DBG_TIM8_STOP**：内核停止时 TIM8 计数器停止
0：即使内核停止，仍然馈送 TIM8 计数器的时钟
1：内核停止时停止 TIM8 计数器的时钟

Bit 12 保留，必须保持复位值。

Bit 11 **DBG_TIM1_STOP**：内核停止时 TIM1 计数器停止
0：即使内核停止，仍然馈送 TIM1 计数器的时钟
1：内核停止时停止 TIM1 计数器的时钟

Bit 10:0 保留，必须保持复位值。

# 36 器件电子签名

电子签名存放在闪存存储器模块的系统存储区域。它所包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的 RX32G410xx 微控制器。

## 36.1 闪存容量寄存器

基地址：0x1FFF 07E0

只读，它的内容在出厂时编写

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

Bit 15:0 **FLASH_SIZE[15:0]：**闪存存储器容量

以 K 字节为单位指示产品中闪存存储器容量。

例：0x0200 = 512K 字节

## 36.2 产品唯一身份标识寄存器（96 位）

产品唯一的身份标识非常适合：

- 用来作为序列号（例如 USB 字符序列号或者其他的终端应用）
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器内的安全性。
- 用来激活带安全机制的自举过程

96 位的产品唯一身份标识所提供的参考号码对任意一个 RX32 微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

这个 96 位的产品唯一身份标识，按照用户不同的用法，可以以字节（8 位）为单位读取，也可以以半字（16 位）或者全字（32 位）读取。

**基地址：0x1FFF 07E4**

地址偏移：0x00

只读，其值在出厂时编写

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UID[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 |
| UID[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:0 **UID[31:0]：**唯一身份标志 31:0 位

地址偏移：0x04

只读，其值在出厂时编写

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UID[63:48] | | | | | | | | | | | | | | | |
| 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 |
| UID[47:32] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:0 **UID[63:32]**：唯一身份标志 63:32 位

地址偏移：0x08

只读，其值在出厂时编写

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UID[95:80] | | | | | | | | | | | | | | | |
| 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 |
| UID[79:64] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31:0 **UID[95:64]**：唯一身份标志 95:64 位

# 37 版本历史

表 37.1 版本历史

| 日期 | 版本 | 更改内容 |
| --- | --- | --- |
| 2025/06/10 | V2.0 | 新版 |
| 2025/07/24 | V2.1 | 1. 修正 DMAMUX 章节 18.6.1 寄存器 DMAMUX_CxCR 参考表格序号<br>2. 更新 TIM2 和 TIM5 定时器为 28 位定时器<br>3. 删除 UART 章节 IDLE 和 IDLE 中断相关描述<br>4. 新增 Flash 章节配置 RDP[7:0]=0xCC 来设置读保护等级为 Level2 |