W55MH32 参考手册

Version 1.0.1

http://www.wiznet.io/

目 录

1 文中的缩写 32

2 存储器和总线架构 33

2.1 系统框架 33

2.2 存储器组织 34

2.3 存储器映像 34

2.3.1 嵌入式 SRAM 36

2.3.2 位段 36

2.3.3 嵌入式闪存 36

2.4 启动配置 38

3 CRC 计算单元 40

3.1 CRC 简介 40

3.2 CRC 主要特性 40

3.3 CRC 功能描述 40

3.4 CRC 寄存器 41

3.4.1 数据寄存器(CRC_DR) 41

3.4.2 独立数据寄存器(CRC_IDR) 41

3.4.3 控制寄存器(CRC_CR) 41

3.4.4 CRC 寄存器映像 42

4 电源控制(PWR) 43

4.1 电源 43

4.1.1 独立的 A/D 转换器供电和参考电压 43

4.1.2 电池备份区域 44

4.1.3 电压调节器 44

4.2 电源管理器 44

4.2.1 上电复位(POR)和掉电复位(PDR) 44

4.2.2 可编程电压监测器(PVD) 45

4.3 低功耗模式 46

4.3.1 降低系统时钟 46

4.3.2 外部时钟的控制 46

4.3.3 睡眠模式 47

4.3.4 停止模式 48

4.3.5 待机模式 49

4.3.6 低功耗模式下的自动唤醒(AWU) 50

4.4 电源控制寄存器 50

4.4.1 电源控制寄存器(PWR_CR) 50

4.4.2 电源控制/状态寄存器(PWR_CSR) 51

4.4.3 PWR 寄存器地址映像 52

5 备份寄存器(BKP) 53

5.1 BKP 简介 53

5.2 BKP 特性 53

5.3 BKP 功能描述 53

5.3.1 侵入检测 53

5.3.2 RTC 校准 54
5.4 BKP 寄存器描述 54
5.4.1 备份数据寄存器 x(BKP_DRx)(x=1…10) 54
5.4.2 RTC 时钟校准寄存器(BKP_RTCCR) 54
5.4.3 备份控制寄存器(BKP_CR) 55
5.4.4 备份控制/状态寄存器(BKP_CSR) 55
5.4.5 BKP 寄存器映像 56
6 复位和时钟控制(RCC) 59
6.1 复位 59
6.1.1 系统复位 59
6.1.2 电源复位 59
6.1.3 备份域复位 60
6.2 时钟 60
6.2.1 HSE 时钟 61
6.2.2 HSI 时钟 62
6.2.3 PLL 63
6.2.4 LSE 时钟 63
6.2.5 LSI 时钟 63
6.2.6 系统时钟(SYSCLK)选择 64
6.2.7 时钟安全系统(CSS) 64
6.2.8 RTC 时钟 64
6.2.9 看门狗时钟 65
6.2.10 时钟输出 65
6.3 RCC 寄存器描述 65
6.3.1 时钟控制寄存器(RCC_CR) 65
6.3.2 时钟配置寄存器(RCC_CFGR) 66
6.3.3 时钟中断寄存器(RCC_CIR) 68
6.3.4 APB2 外设复位寄存器(RCC_APB2RSTR) 70
6.3.5 APB1 外设复位寄存器(RCC_APB1RSTR) 72
6.3.6 AHB 外设时钟使能寄存器(RCC_AHBENR) 74
6.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR) 75
6.3.8 APB1 外设时钟使能寄存器(RCC_APB1ENR) 76
6.3.9 备份域控制寄存器(RCC_BDCR) 78
6.3.10 控制/状态寄存器(RCC_CSR) 79
6.3.11 RCC 寄存器地址映像 80
7 通用和复用功能 I/O(GPIO 和 AFIO) 82
7.1 GPIO 功能描述 82
7.1.1 通用 I/O(GPIO) 83
7.1.2 单独的位设置或位清除 84
7.1.3 外部中断/唤醒线 84
7.1.4 复用功能(AF) 84
7.1.5 软件重新映射 I/O 复用功能 84
7.1.6 GPIO 锁定机制 84
7.1.7 输入配置 84
7.1.8 输出配置 85

7.1.9 复用功能配置 86
7.1.10 模拟输入配置 87
7.1.11 外设的 GPIO 配置 87
7.2 GPIO 寄存器描述 89
7.2.1 端口配置低寄存器(GPIOx_CRL)(x=A..G) 89
7.2.2 端口配置高寄存器(GPIOx_CRH)(x=A..G) 90
7.2.3 端口输入数据寄存器(GPIOx_IDR)(x=A..G) 90
7.2.4 端口输出数据寄存器(GPIOx_ODR)(x=A..G) 90
7.2.5 端口位设置/清除寄存器(GPIOx_BSRR)(x=A..G) 91
7.2.6 端口位清除寄存器(GPIOx_BRR)(x=A..G) 91
7.2.7 端口配置锁定寄存器(GPIOx_LCKR)(x=A..G) 91
7.3 复用功能 I/O 和调试配置(AFIO) 92
7.3.1 把 OSC32_IN/OSC32_OUT 作为 GPIO 端口 PC14/PC15 92
7.3.2 把 OSC_IN/OSC_OUT 引脚作为 GPIO 端口 PD0/PD1 92
7.3.3 CAN1 复用功能重映射 92
7.3.4 JTAG/SWD 复用功能重映射 93
7.3.5 ADC 复用功能重映射 93
7.3.6 定时器复用功能重映射 94
7.3.7 USART 复用功能重映射 95
7.3.8 I2C1 复用功能重映射 96
7.3.9 SPI1 复用功能重映射 96
7.3.10 SPI3 复用功能重映射 96
7.4 AFIO 寄存器描述 96
7.4.1 事件控制寄存器(AFIO_EVCR) 96
7.4.2 复用重映射和调试 I/O 配置寄存器(AFIO_MAPR) 97
7.4.3 外部中断配置寄存器 1(AFIO_EXTICR1) 99
7.4.4 外部中断配置寄存器 2(AFIO_EXTICR2) 100
7.4.5 外部中断配置寄存器 3(AFIO_EXTICR3) 100
7.4.6 外部中断配置寄存器 4(AFIO_EXTICR4) 100
7.4.7 AF 重新映射并调试 I/O 配置寄存器 2(AFIO_MAPR2) 101
7.5 GPIO 和 AFIO 寄存器地址映象 101
8 中断和事件 103
8.1 嵌套向量中断控制器 103
8.1.1 系统嘀嗒(SysTick)校准值寄存器 103
8.1.2 中断和异常向量 103
8.2 外部中断/事件控制器(EXTI) 105
8.2.1 主要特性 105
8.2.2 框图 106
8.2.3 唤醒事件管理 106
8.2.4 功能说明 106
8.2.5 外部中断/事件线路映像 108
8.3 EXTI 寄存器描述 108
8.3.1 中断屏蔽寄存器(EXTI_IMR) 109
8.3.2 事件屏蔽寄存器(EXTI_EMR) 109
8.3.3 上升沿触发选择寄存器(EXTI_RTSR) 109

8.3.4 下降沿触发选择寄存器(EXTI_FTSR) 110
8.3.5 软件中断事件寄存器(EXTI_SWIER) 110
8.3.6 挂起寄存器(EXTI_PR) 111
8.3.7 外部中断/事件寄存器映像 111
9 TCP/IP 卸载引擎(TOE) 112
9.1 TOE 简介 112
9.2 TOE 主要特性 112
9.3 寄存器和内存构成 113
9.3.1 通用寄存器区 114
9.3.2 Socket 寄存器区 115
9.4 内存 Memory 115
9.5 寄存器描述 116
9.5.1 通用寄存器 116
9.5.2 Socket 寄存器 122
10 DMA 控制器(DMA) 132
10.1 DMA 简介 132
10.2 DMA 主要特性 132
10.3 功能描述 133
10.3.1 DMA 处理 133
10.3.2 仲裁器 134
10.3.3 DMA 通道 134
10.3.4 可编程的数据传输宽度、对齐方式和数据大小端 135
10.3.5 错误管理 136
10.3.6 中断 136
10.3.7 DMA 请求映像 137
10.4 DMA 寄存器 139
10.4.1 DMA 中断状态寄存器(DMA_ISR) 139
10.4.2 DMA 中断标志清除寄存器(DMA_IFCR) 140
10.4.3 DMA 通道 x 配置寄存器(DMA_CCRx)(x=1...7) 140
10.4.4 DMA 通道 x 传输数量寄存器(DMA_CNDTRx)(x=1...7) 141
10.4.5 DMA 通道 x 外设地址寄存器(DMA_CPARx)(x=1...7) 142
10.4.6 DMA 通道 x 存储器地址寄存器(DMA_CMARx)(x=1...7) 142
10.4.7 DMA 寄存器映像 143
11 模拟/数字转换(ADC) 145
11.1 ADC 介绍 145
11.2 ADC 主要特征 145
11.3 ADC 功能描述 146
11.3.1 ADC 开关控制 147
11.3.2 ADC 时钟 147
11.3.3 通道选择 147
11.3.4 单次转换模式 147
11.3.5 连续转换模式 148
11.3.6 时序图 148
11.3.7 模拟看门狗 148

11.3.8 扫描模式 149
11.3.9 注入通道管理 149
11.3.10 间断模式 150
11.4 校准 151
11.5 数据对齐 151
11.6 可编程的通道采样时间 152
11.7 外部触发转换 152
11.8 DMA 请求 153
11.9 双 ADC 模式 154
11.9.1 同步注入模式 155
11.9.2 同步规则模式 156
11.9.3 快速交叉模式 156
11.9.4 慢速交叉模式 157
11.9.5 交替触发模式 157
11.9.6 独立模式 158
11.9.7 混合的规则/注入同步模式 158
11.9.8 混合的同步规则+交替触发模式 158
11.9.9 混合同步注入+交叉模式 159
11.10 温度传感器 159
11.11 ADC 中断 160
11.12 ADC 寄存器 161
11.12.1 ADC 状态寄存器(ADC_SR) 161
11.12.2 ADC 控制寄存器 1(ADC_CR1) 162
11.12.3 ADC 控制寄存器 2(ADC_CR2) 164
11.12.4 ADC 采样时间寄存器 1(ADC_SMPR1) 166
11.12.5 ADC 采样时间寄存器 2(ADC_SMPR2) 166
11.12.6 ADC 注入通道数据偏移寄存器 x(ADC_JOFRx)(x=1..4) 167
11.12.7 ADC 看门狗高阀值寄存器(ADC_HTR) 167
11.12.8 ADC 看门狗低阀值寄存器(ADC_LRT) 167
11.12.9 ADC 规则序列寄存器 1(ADC_SQR1) 168
11.12.10 ADC 规则序列寄存器 2(ADC_SQR2) 168
11.12.11 ADC 规则序列寄存器 3(ADC_SQR3) 169
11.12.12 ADC 注入序列寄存器(ADC_JSQR) 169
11.12.13 ADC 注入数据寄存器 x(ADC_JDRx)(x=1..4) 170
11.12.14 ADC 规则数据寄存器(ADC_DR) 170
11.12.15 ADC 寄存器地址映像 170
12 数字/模拟转换(DAC) 172
12.1 DAC 简介 172
12.2 DAC 主要特征 172
12.3 DAC 功能描述 173
12.3.1 使能 DAC 通道 173
12.3.2 使能 DAC 输出缓存 174
12.3.3 DAC 数据格式 174
12.3.4 DAC 转换 175

12.3.5 DAC 输出电压 175
12.3.6 选择 DAC 触发 175
12.3.7 DMA 请求 176
12.3.8 噪声生成 176
12.3.9 三角波生成 177
12.4 双 DAC 通道转换 178
12.4.1 不使用波形发生器的独立触发 178
12.4.2 使用相同 LFSR 的独立触发 178
12.4.3 使用不同 LFSR 的独立触发 178
12.4.4 产生相同三角波的独立触发 179
12.4.5 产生不同三角波的独立触发 179
12.4.6 同时软件启动 179
12.4.7 不使用波形发生器的同时触发 179
12.4.8 使用相同 LFSR 的同时触发 179
12.4.9 使用不同 LFSR 的同时触发 180
12.4.10 使用相同三角波发生器的同时触发 180
12.4.11 使用不同三角波发生器的同时触发 180
12.5 DAC 寄存器 181
12.5.1 DAC 控制寄存器(DAC_CR) 181
12.5.2 DAC 软件触发寄存器(DAC_SWTRIGR) 183
12.5.3 DAC 通道 1 的 12 位右对齐数据保持寄存器(DAC_DHR12R1) 184
12.5.4 DAC 通道 1 的 12 位左对齐数据保持寄存器(DAC_DHR12L1) 184
12.5.5 DAC 通道 1 的 8 位右对齐数据保持寄存器(DAC_DHR8R1) 184
12.5.6 DAC 通道 2 的 12 位右对齐数据保持寄存器(DAC_DHR12R2) 185
12.5.7 DAC 通道 2 的 12 位左对齐数据保持寄存器(DAC_DHR12L2) 185
12.5.8 DAC 通道 2 的 8 位右对齐数据保持寄存器(DAC_DHR8R2) 185
12.5.9 双 DAC 的 12 位右对齐数据保持寄存器(DAC_DHR12RD) 186
12.5.10 双 DAC 的 12 位左对齐数据保持寄存器(DAC_DHR12LD) 186
12.5.11 双 DAC 的 8 位右对齐数据保持寄存器(DAC_DHR8RD) 186
12.5.12 DAC 通道 1 数据输出寄存器(DAC_DOR1) 187
12.5.13 DAC 通道 2 数据输出寄存器(DAC_DOR2) 187
12.5.14 DAC 寄存器映像 188
13 高级控制定时器(TIM1 和 TIM8) 189
13.1 TIM1 和 TIM8 简介 189
13.2 TIM1 和 TIM8 主要特性 189
13.3 TIM1 和 TIM8 功能描述 190
13.3.1 时基单元 190
13.3.2 计数器模式 192
13.3.3 重复计数器 199
13.3.4 时钟选择 200
13.3.5 捕获/比较通道 202
13.3.6 输入捕获模式 203
13.3.7 PWM 输入模式 204
13.3.8 强置输出模式 205
13.3.9 输出比较模式 205

13.3.10 PWM 模式 206
13.3.11 互补输出和死区插入 208
13.3.12 使用刹车功能 210
13.3.13 在外部事件时清除 OCxREF 信号 211
13.3.14 产生六步 PWM 输出 212
13.3.15 单脉冲模式 213
13.3.16 编码器接口模式 215
13.3.17 定时器输入异或功能 216
13.3.18 与霍尔传感器的接口 216
13.3.19 TIMx 定时器和外部触发的同步 218
13.3.20 定时器同步 221
13.3.21 调试模式 221
13.4 TIM1 和 TIM8 寄存器描述 221
13.4.1 TIM1 和 TIM8 控制寄存器 1(TIMx_CR1) 221
13.4.2 TIM1 和 TIM8 控制寄存器 2(TIMx_CR2) 222
13.4.3 TIM1 和 TIM8 从模式控制寄存器(TIMx_SMCR) 223
13.4.4 TIM1 和 TIM8DMA/中断使能寄存器(TIMx_DIER) 225
13.4.5 TIM1 和 TIM8 状态寄存器(TIMx_SR) 226
13.4.6 TIM1 和 TIM8 事件产生寄存器(TIMx_EGR) 227
13.4.7 TIM1 和 TIM8 捕获/比较模式寄存器 1(TIMx_CCMR1) 228
13.4.8 TIM1 和 TIM8 捕获/比较模式寄存器 2(TIMx_CCMR2) 231
13.4.9 TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER) 232
13.4.10 TIM1 和 TIM8 计数器(TIMx_CNT) 234
13.4.11 TIM1 和 TIM8 预分频器(TIMx_PSC) 234
13.4.12 TIM1 和 TIM8 自动重装载寄存器(TIMx_ARR) 234
13.4.13 TIM1 和 TIM8 重复计数寄存器(TIMx_RCR) 235
13.4.14 TIM1 和 TIM8 捕获/比较寄存器 1(TIMx_CCR1) 235
13.4.15 TIM1 和 TIM8 捕获/比较寄存器 2(TIMx_CCR2) 235
13.4.16 TIM1 和 TIM8 捕获/比较寄存器 3(TIMx_CCR3) 236
13.4.17 TIM1 和 TIM8 捕获/比较寄存器(TIMx_CCR4) 236
13.4.18 TIM1 和 TIM8 刹车和死区寄存器(TIMx_BDTR) 236
13.4.19 TIM1 和 TIM8 DMA 控制寄存器(TIMx_DCR) 238
13.4.20 TIM1 和 TIM8 连续模式的 DMA 地址(TIMx_DMAR) 239
13.4.21 TIM1 和 TIM8 寄存器 240
14 通用定时器(TIM2 到 TIM5) 242
14.1 TIM2 到 TIM5 简介 242
14.2 TIM2 到 TIM5 主要功能 242
14.3 TIM2 到 TIM5 功能描述 243
14.3.1 时基单元 243
14.3.2 计数器模式 245
14.3.3 时钟选择 252
14.3.4 捕获/比较通道 254
14.3.5 输入捕获模式 255
14.3.6 PWM 输入模式 256
14.3.7 强置输出模式 257

14.3.8 输出比较模式 257
14.3.9 PWM 模式 258
14.3.10 单脉冲模式 260
14.3.11 在外部事件时清除 OCxREF 信号 262
14.3.12 编码器接口模式 262
14.3.13 定时器输入异或功能 264
14.3.14 定时器和外部触发的同步 265
14.3.15 定时器同步 267
14.3.16 调试模式 271
14.4 TIM2 到 TIM5 寄存器描述 271
14.4.1 TIM2 到 TIM5 控制寄存器 1(TIMx_CR1) 271
14.4.2 TIM2 到 TIM5 控制寄存器 2(TIMx_CR2) 272
14.4.3 TIM2 到 TIM5 从模式控制寄存器(TIMx_SMCR) 273
14.4.4 TIM2 到 TIM5DMA/中断使能寄存器(TIMx_DIER) 275
14.4.5 TIM2 到 TIM5 状态寄存器(TIMx_SR) 276
14.4.6 TIM2 到 TIM5 事件产生寄存器(TIMx_EGR) 277
14.4.7 TIM2 到 TIM5 捕获/比较模式寄存器 1(TIMx_CCMR1) 277
14.4.8 TIM2 到 TIM5 捕获/比较模式寄存器 2(TIMx_CCMR2) 280
14.4.9 TIM2 到 TIM5 捕获/比较使能寄存器(TIMx_CCER) 281
14.4.10 TIM2 到 TIM5 计数器(TIMx_CNT) 282
14.4.11 TIM2 到 TIM5 预分频器(TIMx_PSC) 282
14.4.12 TIM2 到 TIM5 自动重装载寄存器(TIMx_ARR) 282
14.4.13 TIM2 到 TIM5 捕获/比较寄存器 1(TIMx_CCR1) 282
14.4.14 TIM2 到 TIM5 捕获/比较寄存器 2(TIMx_CCR2) 283
14.4.15 TIM2 到 TIM5 捕获/比较寄存器 3(TIMx_CCR3) 283
14.4.16 TIM2 到 TIM5 捕获/比较寄存器 4(TIMx_CCR4) 284
14.4.17 TIM2 到 TIM5DMA 控制寄存器(TIMx_DCR) 284
14.4.18 TIM2 到 TIM5 连续模式的 DMA 地址(TIMx_DMAR) 284
14.4.19 TIM2 到 TIM5 寄存器 285
15 通用定时器(TIM9 到 TIM14) 287
15.1 TIM9 到 TIM14 简介 287
15.2 TIM9 到 TIM14 的主要功能 287
15.2.1 TIM9/TIM12 的主要功能 287
15.2.2 TIM10/TIM11 和 TIM13/TIM14 的主要功能 288
15.3 TIM9 到 TIM14 功能描述 289
15.3.1 时基单元 289
15.3.2 计数器模式 290
15.3.3 时钟选择 293
15.3.4 捕获/比较通道 294
15.3.5 输入捕获模式 295
15.3.6 PWM 输入模式(仅适用于 TIM9/12) 296
15.3.7 强置输出模式 297
15.3.8 输出比较模式 297
15.3.9 PWM 模式 298
15.3.10 单脉冲模式 299

15.3.11 TIM9/12 定时器和外部触发的同步 300
15.3.12 定时器同步（TIM9/12） 302
15.3.13 调试模式 302
15.4 TIM9/TIM12 寄存器 302
15.4.1 TIM9/12 控制寄存器 1(TIMx_CR1) 302
15.4.2 TIM9/12 从模式控制寄存器(TIMx_SMCR) 303
15.4.3 TIM9/12 中断使能寄存器(TIMx_DIER) 304
15.4.4 TIM9/12 状态寄存器(TIMx_SR) 304
15.4.5 TIM9/12 事件产生寄存器(TIMx_EGR) 305
15.4.6 TIM9/12 捕获/比较模式寄存器 1(TIMx_CCMR1) 306
15.4.7 TIM9/12 捕获/比较使能寄存器(TIMx_CCER) 308
15.4.8 TIM9/12 计数器(TIMx_CNT) 309
15.4.9 TIM9/12 预分频器(TIMx_PSC) 309
15.4.10 TIM9/12 自动重装载寄存器(TIMx_ARR) 309
15.4.11 TIM9/12 捕获/比较寄存器 1(TIMx_CCR1) 309
15.4.12 TIM9/12 捕获/比较寄存器 2(TIMx_CCR2) 310
15.4.13 TIM9/12 寄存器 310
15.5 TIM10/11/13/14 寄存器 311
15.5.1 TIM10/11/13/14 控制寄存器 1(TIMx_CR1) 312
15.5.2 TIM10/11/13/14 中断使能寄存器(TIMx_DIER) 312
15.5.3 TIM10/11/13/14 状态寄存器(TIMx_SR) 313
15.5.4 TIM10/11/13/14 事件产生寄存器(TIMx_EGR) 313
15.5.5 TIM10/11/13/14 捕获/比较模式寄存器 1(TIMx_CCMR1) 314
15.5.6 TIM10/11/13/14 捕获/比较使能寄存器(TIMx_CCER) 316
15.5.7 TIM10/11/13/14 计数器(TIMx_CNT) 317
15.5.8 TIM10/11/13/14 预分频器(TIMx_PSC) 317
15.5.9 TIM10/11/13/14 自动重装载寄存器(TIMx_ARR) 317
15.5.10 TIM10/11/13/14 捕获/比较寄存器 1(TIMx_CCR1) 317
15.5.11 TIM10/11/13/14 寄存器 318
16 基本定时器(TIM6 和 TIM7) 320
16.1 TIM6 和 TIM7 简介 320
16.2 TIM6 和 TIM7 的主要特性 320
16.3 TIM6 和 TIM7 的功能 320
16.3.1 时基单元 320
16.3.2 计数模式 322
16.3.3 时钟源 324
16.3.4 调试模式 325
16.4 TIM6 和 TIM7 寄存器 325
16.4.1 TIM6 和 TIM7 控制寄存器 1(TIMx_CR1) 325
16.4.2 TIM6 和 TIM7 控制寄存器 2(TIMx_CR2) 326
16.4.3 TIM6 和 TIM7DMA/中断使能寄存器(TIMx_DIER) 326
16.4.4 TIM6 和 TIM7 状态寄存器(TIMx_SR) 327
16.4.5 TIM6 和 TIM7 事件产生寄存器(TIMx_EGR) 327
16.4.6 TIM6 和 TIM7 计数器(TIMx_CNT) 327
16.4.7 TIM6 和 TIM7 预分频器(TIMx_PSC) 328

16.4.8 TIM6 和 TIM7 自动重装载寄存器(TIMx_ARR) 328
16.4.9 TIM6 和 TIM7 寄存器 329
17 实时时钟(RTC) 330
17.1 RTC 简介 330
17.2 主要特性 330
17.3 功能描述 330
17.3.1 概述 330
17.3.2 复位过程 331
17.3.3 读 RTC 寄存器 331
17.3.4 配置 RTC 寄存器 332
17.3.5 RTC 标志的设置 332
17.4 RTC 寄存器描述 333
17.4.1 RTC 控制寄存器高位(RTC_CRH) 333
17.4.2 RTC 控制寄存器低位(RTC_CRL) 333
17.4.3 RTC 预分频装载寄存器(RTC_PRLH/RTC_PRLL) 334
17.4.4 RTC 预分频器余数寄存器(RTC_DIVH/RTC_DIVL) 335
17.4.5 RTC 计数器寄存器(RTC_CNTH/RTC_CNTL) 335
17.4.6 RTC 闹钟寄存器(RTC_ALRH/RTC_ALRL) 336
17.4.7 RTC 寄存器映像 337
18 独立看门狗(IWDG) 338
18.1 简介 338
18.2 IWDG 主要性能 338
18.3 IWDG 功能描述 338
18.3.1 硬件看门狗 338
18.3.2 寄存器访问保护 338
18.3.3 调试模式 339
18.4 IWDG 寄存器描述 339
18.4.1 键寄存器(IWDG_KR) 339
18.4.2 预分频寄存器(IWDG_PR) 340
18.4.3 重装载寄存器(IWDG_RLR) 340
18.4.4 状态寄存器(IWDG_SR) 341
18.4.5 IWDG 寄存器映像 341
19 窗口看门狗(WWDG) 342
19.1 WWDG 简介 342
19.2 WWDG 主要特性 342
19.3 WWDG 功能描述 342
19.4 如何编写看门狗超时程序 343
19.5 调试模式 344
19.6 寄存器描述 344
19.6.1 控制寄存器(WWDG_CR) 344
19.6.2 配置寄存器(WWDG_CFR) 344
19.6.3 状态寄存器(WWDG_SR) 345
19.6.4 WWDG 寄存器映像 345
20 SDIO 接口(SDIO) 346

20.1 SDIO 主要功能 346
20.2 SDIO 总线拓扑 346
20.3 SDIO 功能描述 348
20.3.1 SDIO 适配器 349
20.3.2 SDIOAHB 接口 357
20.4 卡功能描述 358
20.4.1 卡识别模式 358
20.4.2 卡复位 358
20.4.3 操作电压范围确认 358
20.4.4 卡识别过程 358
20.4.5 写数据块 359
20.4.6 读数据块 360
20.4.7 数据流操作，数据流写入和数据流读出(只适用于多媒体卡) 360
20.4.8 擦除：成组擦除和扇区擦除 361
20.4.9 宽总线选择和解除选择 361
20.4.10 保护管理 362
20.4.11 卡状态寄存器 364
20.4.12 SD 状态寄存器 366
20.4.13 SD 的 I/O 模式 369
20.4.14 命令与响应 370
20.5 响应格式 373
20.5.1 R1(普通响应命令) 373
20.5.2 R1b 373
20.5.3 R2(CID、CSD 寄存器) 373
20.5.4 R3(OCR 寄存器) 374
20.5.5 R4(快速 I/O) 374
20.5.6 R4b 374
20.5.7 R5(中断请求) 375
20.5.8 R6(中断请求) 375
20.6 SDIOI/O 卡特定的操作 375
20.6.1 使用 SDIO_D2 信号线的 SDIOI/O 读等待操作 376
20.6.2 使用停止 SDIO_CK 的 SDIO 读等待操作 376
20.6.3 SDIO 暂停/恢复操作 376
20.6.4 SDIO 中断 376
20.7 CE-ATA 特定操作 376
20.7.1 命令完成指示关闭 377
20.7.2 命令完成指示使能 377
20.7.3 CE-ATA 中断 377
20.7.4 中止 CMD61 377
20.8 硬件流控制 377
20.9 SDIO 寄存器 377
20.9.1 SDIO 电源控制寄存器(SDIO_POWER) 377
20.9.2 SDIO 时钟控制寄存器(SDIO_CLKCR) 378
20.9.3 SDIO 参数寄存器(SDIO_ARG) 379
20.9.4 SDIO 命令寄存器(SDIO_CMD) 379

20.9.5 SDIO 命令响应寄存器(SDIO_RESPCMD)..... 380
20.9.6 SDIO 响应 1..4 寄存器(SDIO_RESPx)..... 380
20.9.7 SDIO 数据定时器寄存器(SDIO_DTIMER)..... 381
20.9.8 SDIO 数据长度寄存器(SDIO_DLEN)..... 381
20.9.9 SDIO 数据控制寄存器(SDIO_DCTRL)..... 381
20.9.10 SDIO 数据计数器寄存器(SDIO_DCOUNT)..... 382
20.9.11 SDIO 状态寄存器(SDIO_STA)..... 383
20.9.12 SDIO 清除中断寄存器(SDIO_ICR)..... 384
20.9.13 SDIO 中断屏蔽寄存器(SDIO_MASK)..... 385
20.9.14 SDIOFIFO 计数器寄存器(SDIO_FIFOCNT)..... 387
20.9.15 SDIO 数据 FIFO 寄存器(SDIO_FIFO)..... 387
20.9.16 SDIO 寄存器映像..... 388
21 USB 全速设备接口(USB)..... 390
21.1 USB 简介..... 390
21.2 USB 主要特征..... 390
21.3 USB 功能描述..... 391
21.3.1 USB 功能模块描述..... 392
21.4 编程中需要考虑的问题..... 393
21.4.1 系统复位和上电复位..... 393
21.4.2 双缓冲端点..... 397
21.4.3 同步传输..... 398
21.4.4 挂起/恢复事件..... 399
21.5 USB 寄存器描述..... 400
21.5.1 通用寄存器..... 400
21.5.2 端点寄存器..... 405
21.5.3 缓冲区描述表..... 407
21.5.4 USB 寄存器映像..... 409
22 控制器局域网(bxCAN)..... 412
22.1 bxCAN 简介..... 412
22.2 bxCAN 主要特点..... 412
22.3 bxCAN 总体描述..... 412
22.3.1 CAN2.0B 主动内核..... 413
22.3.2 控制、状态和配置寄存器..... 413
22.3.3 发送邮箱..... 413
22.3.4 接收过滤器..... 413
22.4 bxCAN 工作模式..... 413
22.4.1 初始化模式..... 414
22.4.2 正常模式..... 414
22.4.3 睡眠模式(低功耗)..... 414
22.5 测试模式..... 415
22.5.1 静默模式..... 415
22.5.2 环回模式..... 416
22.5.3 环回静默模式..... 416
22.6 W55MH32 处于调试模式时..... 416

22.7 bxCAN 功能描述 416
22.7.1 发送处理 417
22.7.2 时间触发通信模式 418
22.7.3 接收管理 418
22.7.4 标识符过滤 420
22.7.5 报文存储 423
22.7.6 出错管理 424
22.7.7 位时间特性 425
22.8 bxCAN 中断 427
22.9 CAN 寄存器描述 428
22.9.1 寄存器访问保护 428
22.9.2 CAN 控制和状态寄存器 428
22.9.3 CAN 邮箱寄存器 435
22.9.4 CAN 过滤器寄存器 439
22.9.5 bxCAN 寄存器列表 442
23 串行外设接口(SPI) 445
23.1 SPI 简介 445
23.2 SPI 和 I²S 主要特征 445
23.2.1 SPI 特征 445
23.2.2 I²S 功能 446
23.3 SPI 功能描述 446
23.3.1 概述 446
23.3.2 配置 SPI 为从模式 449
23.3.3 配置 SPI 为主模式 450
23.3.4 配置 SPI 为单工通信 451
23.3.5 数据发送与接收过程 452
23.3.6 CRC 计算 456
23.3.7 状态标志 458
23.3.8 关闭 SPI 458
23.3.9 利用 DMA 的 SPI 通信 459
23.3.10 错误标志 461
23.3.11 SPI 中断 461
23.4 I2S 功能描述 462
23.4.1 I2S 功能描述 462
23.4.2 支持的音频协议 463
23.4.3 时钟发生器 469
23.4.4 I²S 主模式 470
23.4.5 I²S 从模式 472
23.4.6 状态标志位 473
23.4.7 错误标志位 474
23.4.8 I²S 中断 474
23.4.9 DMA 功能 474
23.5 SPI 和 I²S 寄存器描述 474
23.5.1 SPI 控制寄存器 1(SPI_CR1)(I²S 模式下不使用) 474
23.5.2 SPI 控制寄存器 2(SPI_CR2) 476

23.5.3 SPI 状态寄存器(SPI_SR) 477
23.5.4 SPI 数据寄存器(SPI_DR) 478
23.5.5 SPICRC 多项式寄存器(SPI_CRCPR)(I²S 模式下不使用) 478
23.5.6 SPIRxCRC 寄存器(SPI_RXCRCR)(I²S 模式下不使用) 478
23.5.7 SPITxCRC 寄存器(SPI_TXCRCR)(I²S 模式下不使用) 478
23.5.8 SPI_I²S 配置寄存器(SPI_I2SCFGR) 479
23.5.9 SPI_I²S 预分频寄存器(SPI_I2SPR) 480
23.5.10 寄存器地址映象 480
24 I2C 接口 482
24.1 I²C 简介 482
24.2 I²C 主要特点 482
24.3 I²C 功能描述 483
24.3.1 模式选择 483
24.3.2 I²C 从模式 484
24.3.3 I²C 主模式 486
24.3.4 错误条件 489
24.3.5 SDA/SCL 线控制 490
24.3.6 SMBus 490
24.3.7 DMA 请求 492
24.3.8 包错误校验(PEC) 493
24.4 I²C 中断请求 493
24.5 I²C 调试模式 494
24.6 I²C 寄存器描述 495
24.6.1 控制寄存器 1(I2C_CR1) 495
24.6.2 控制寄存器 2(I2C_CR2) 496
24.6.3 自身地址寄存器 1(I2C_OAR1) 497
24.6.4 自身地址寄存器 2(I2C_OAR2) 498
24.6.5 数据寄存器(I2C_DR) 498
24.6.6 状态寄存器 1(I2C_SR1) 499
24.6.7 状态寄存器 2(I2C_SR2) 501
24.6.8 时钟控制寄存器(I2C_CCR) 502
24.6.9 TRISE 寄存器(I2C_TRISE) 503
24.6.10 I2C 寄存器地址映象 503
25 通用同步异步收发器(USART) 505
25.1 USART 介绍 505
25.2 USART 主要特性 505
25.3 USART 功能概述 506
25.3.1 USART 特性描述 507
25.3.2 发送器 508
25.3.3 接收器 510
25.3.4 分数波特率的产生 514
25.3.5 USART 接收器容忍时钟的变化 515
25.3.6 多处理器通信 515
25.3.7 校验控制 517
25.3.8 LIN(局域互联网)模式 517

25.3.9 USART 同步模式 519
25.3.10 单线半双工通信 521
25.3.11 智能卡 521
25.3.12 IrDA SIR ENDEC 功能模块 523
25.3.13 利用 DMA 连续通信 525
25.3.14 硬件流控制 526
25.4 USART 中断请求 527
25.5 USART 模式配置 528
25.6 USART 寄存器描述 528
25.6.1 状态寄存器(USART_SR) 529
25.6.2 数据寄存器(USART_DR) 530
25.6.3 波特比率寄存器(USART_BRR) 531
25.6.4 控制寄存器 1(USART_CR1) 531
25.6.5 控制寄存器 2(USART_CR2) 533
25.6.6 控制寄存器 3(USART_CR3) 534
25.6.7 保护时间和预分频寄存器(USART_GTPR) 535
25.6.8 USART 寄存器地址映象 536
26 USBOTG 全速(OTG_FS) 537
26.1 OTG 模块介绍 537
26.2 OTG_FS 主要功能 537
26.2.1 通用功能 537
26.2.2 主机模式功能 538
26.3 OTG_FS 功能描述 539
26.3.1 OTG 引脚 539
26.3.2 OTG 全速控制器 539
26.3.3 全速 OTGPHY(物理接口) 540
26.4 OTG 双角色设备(DRD) 540
26.4.1 ID 信号检测 540
26.4.2 HNP 双角色设备 541
26.4.3 SRP 双角色设备 541
26.5 USB 设备模式 541
26.5.1 具备 SRP 功能的设备 542
26.5.2 设备状态 542
26.5.3 设备端点 543
26.6 USB 主机 544
26.6.1 具备 SRP 功能的主机 545
26.6.2 USB 主机状态 546
26.6.3 主机通道 547
26.6.4 主机调度器 548
26.7 SOF 触发 549
26.7.1 主机 SOFs 549
26.7.2 设备 SOFs 550
26.8 供电选项 550
26.9 OTG FS HFIR 寄存器的动态更新 551

26.10 USB 数据 FIFO 551
26.11 设备模式下的 FIFO 结构 552
26.11.1 设备模式下的接收 FIFO 553
26.11.2 设备模式下的发送 FIFO 553
26.12 主机模式下的 FIFO 结构 554
26.12.1 主机模式下的接收 FIFO 554
26.12.2 主机模式下的发送 FIFO 554
26.13 FIFO RAM 分配 555
26.13.1 设备模式 555
26.14 USB 系统性能 555
26.15 OTG_FS 中断 556
26.16 OTG_FS 控制和状态寄存器 557
26.16.1 CSR 存储器映像 558
26.16.2 OTG_FS 全局寄存器 561
26.16.3 主机模式下的寄存器 577
26.16.4 设备模式下的寄存器 584
26.16.5 OTG_FS 电源和时钟门控寄存器(OTG_FS_PCGCCTL) 600
26.16.6 OTG_FS 寄存器映像 601
26.17 OTG_FS 编程规则 609
26.17.1 控制器初始化 609
26.17.2 主机模式下的初始化 609
26.17.3 设备模式下的初始化 610
26.17.4 主机模式下的编程规则 610
26.17.5 设备模式下的编程规则 625
26.17.6 操作流程 627
26.17.7 最差情况下的响应时间 641
26.17.8 OTG 编程规则 642
27 器件电子签名 648
27.1 存储器容量寄存器 648
27.1.1 闪存容量寄存器 648
27.2 产品唯一身份标识寄存器(96 位) 648
28 调试支持(DBG) 650
28.1 概况 650
28.2 ARM 参考文献 651
28.3 SWJ 调试端口(serial wire and JTAG) 651
28.3.1 JTAG-DP 和 SW-DP 切换的机制 651
28.4 引脚分布和调试端口脚 652
28.4.1 SWJ 调试端口脚 652
28.4.2 灵活的 SWJ-DP 脚分配 652
28.4.3 JTAG 脚上的内部上拉和下拉 653
28.4.4 利用串行接口并释放不用的调试脚作为普通 I/O 口 654
28.5 W55MH32JTAG TAP 连接 654
28.6 ID 代码和锁定机制 655
28.6.1 微控制器设备 ID 编码 655

28.6.2 边界扫描 TAP 655
28.6.3 Cortex-M3 TAP 656
28.6.4 Cortex-M3 JEDEC-106 ID 代码 656
28.7 JTAG 调试端口 656
28.8 SW 调试端口 657
28.8.1 SW 协议介绍 657
28.8.2 SW 协议序列 657
28.8.3 SW-DP 状态机(Reset,idle states,IDcode) 658
28.8.4 DP 和 AP 读/写访问 658
28.8.5 SW-DP 寄存器 659
28.8.6 SW-AP 寄存器 659
28.9 对于 JTAG-DP 或 SWDP 都有效的 AHB-AP(AHB 访问端口) 660
28.10 内核调试 660
28.11 调试器主机在系统复位下的连接能力 661
28.12 FPB(Flash patch breakpoint) 661
28.13 DWT(数据观察点触发 data watch point trigger) 661
28.14 ITM(指令跟踪微单元 instrumentation trace macrocell) 662
28.14.1 概述 662
28.14.2 时间戳包，同步和溢出包 662
28.15 ETM 模块(嵌入式跟踪微单元 Embedded Trace Macrocell) 663
28.15.1 概述 663
28.15.2 信号协议和包类型 663
28.15.3 主要的 ETM 寄存器 663
28.15.4 配置实例 664
28.16 MCU 调试模块(MCUDBG) 664
28.16.1 低功耗模式的调试支持 664
28.16.2 支持定时器、看门狗、bxCAN 和 I2C 的调试 664
28.16.3 调试 MCU 配置寄存器 665
28.17 TPIU(跟踪端口接口单元 Trace Port Interface Unit) 667
28.17.1 导言 667
28.17.2 跟踪引脚分配 667
28.17.3 TPUI 格式器 669
28.17.4 TPUI 帧异步包 669
28.17.5 同步帧包的发送 669
28.17.6 同步模式 670
28.17.7 异步模式 670
28.17.8 TRACECLKIN 在 W55MH32 内部的连接 670
28.17.9 TPIU 寄存器 671
28.17.10 配置的例子 671
28.18 DBG 寄存器地址映象 671
29 文档历史信息 673

插图清单

图 1 系统结构 33
图 2 CRC 计算单元框图 40
图 3 电源框图 43
图 4 上电复位和掉电复位的波形图 45
图 5 PVD 的门限 45
图 6 复位电路 60
图 7 时钟树 61
图 8 HSE/LSE 时钟源 62
图 9 I/O 端口位的基本结构 82
图 10 5 伏兼容 I/O 端口位的基本结构 83
图 11 输入浮空/上拉/下拉配置 85
图 12 输出配置 86
图 13 复用功能配置 86
图 14 高阻抗的模拟输入配置 87
图 15 外部中断/事件控制器框图 106
图 16 外部中断通用 I/O 映像 108
图 17 寄存器及内存构成 113
图 18 INTLEVEL 时序 118
图 19 DMA 框图 133
图 20 DMA1 请求映像 137
图 21 DMA2 请求映像 138
图 22 单个 ADC 框图 146
图 23 时序图 148
图 24 模拟看门狗警戒区 149
图 25 注入转换延时 150
图 26 校准时序图 151
图 27 数据右对齐 152
图 28 数据左对齐 152
图 29 双 ADC 框图(1) 155
图 30 在 4 个通道上的同步注入模式 156
图 31 在 16 个通道上的同步规则模式 156
图 32 在 1 个通道上连续转换模式下的快速交叉模式 157
图 33 在 1 个通道上的慢速交叉模式 157
图 34 交替触发：每个 ADC1 的注入通道组 158
图 35 交替触发：在间断模式下每个 ADC 上的 4 个注入通道 158
图 36 交替+规则同步 159
图 37 触发事件发生在注入转换期间 159
图 38 交叉的单通道转换被注入序列 CH11 和 CH12 中断 159
图 39 温度传感器和 VREFINT 通道框图 160

图 40 DAC 通道模块框图 173
图 41 单 DAC 通道模式的数据寄存器 174
图 42 双 DAC 通道模式的数据寄存器 175
图 43 TEN=0 触发失能时转换的时间框图 175
图 44 DACLFSR 寄存器算法 176
图 45 带 LFSR 波形生成的 DAC 转换(使能软件触发) 177
图 46 DAC 三角波生成 177
图 47 带三角生成的 DAC 转换(使能软件触发) 177
图 48 级控制定时器框图 190
图 49 当预分频器的参数从 1 变到 2 时，计数器的时序图 191
图 50 当预分频器的参数从 1 变到 4 时，计数器的时序图 191
图 51 计数器时序图，内部时钟分频因子为 1 192
图 52 计数器时序图，内部时钟分频因子为 2 193
图 53 计数器时序图，内部时钟分频因子为 4 193
图 54 计数器时序图，内部时钟分频因子为 N 193
图 55 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入) 194
图 56 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR) 194
图 57 计数器时序图，内部时钟分频因子为 1 195
图 58 计数器时序图，内部时钟分频因子为 2 195
图 59 计数器时序图，内部时钟分频因子为 4 195
图 60 计数器时序图，内部时钟分频因子为 N 196
图 61 计数器时序图，当没有使用重复计数器时的更新事件 196
图 62 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6 197
图 63 计数器时序图，内部时钟分频因子为 2 197
图 64 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36 197
图 65 计数器时序图，内部时钟分频因子为 N 198
图 66 计数器时序图，ARPE=1 时的更新事件(计数器下溢) 198
图 67 计数器时序图，ARPE=1 时的更新事件(计数器溢出) 198
图 68 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置 199
图 69 一般模式下的控制电路，内部时钟分频因子为 1 200
图 70 TI2 外部时钟连接例子 200
图 71 外部时钟模式 1 下的控制电路 201
图 72 外部触发输入框图 201
图 73 外部时钟模式 2 下的控制电路 202
图 74 捕获/比较通道(如：通道 1 输入部分) 202
图 75 捕获/比较通道 1 的主电路 202
图 76 捕获/比较通道的输出部分(通道 1 至 3) 203
图 77 捕获/比较通道的输出部分(通道 4) 203
图 78 PWM 输入模式时序 205
图 79 输出比较模式，翻转 OC1 206
图 80 边沿对齐的 PWM 波形(ARR=8) 207

图 81 中央对齐的 PWM 波形(APR=8)....208
图 82 带死区插入的互补输出....209
图 83 死区波形延迟大于负脉冲....209
图 84 死区波形延迟大于正脉冲....209
图 85 响应刹车的输出....211
图 86 清除 TIMx 的 OCxREF....212
图 87 产生六步 PWM，使用 COM 的例子(OSSR=1)....213
图 88 单脉冲模式的例子....214
图 89 编码器模式下的计数器操作实例....216
图 90 IC1FP1 反相的编码器接口模式实例....216
图 91 霍尔传感器接口的实例....218
图 92 复位模式下的控制电路....219
图 93 门控模式下的控制电路....219
图 94 触发器模式下的控制电路....220
图 95 外部时钟模式 2 + 触发模式下的控制电路....221
图 96 通用定时器框图....243
图 97 当预分频器的参数从 1 变到 2 时，计数器的时序图....244
图 98 当预分频器的参数从 1 变到 4 时，计数器的时序图....244
图 99 计数器时序图，内部时钟分频因子为 1....245
图 100 计数器时序图，内部时钟分频因子为 2....245
图 101 计数器时序图，内部时钟分频因子为 4....246
图 102 计数器时序图，内部时钟分频因子为 N....246
图 103 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)....246
图 104 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)....247
图 105 计数器时序图，内部时钟分频因子为 1....248
图 106 计数器时序图，内部时钟分频因子为 2....248
图 107 计数器时序图，内部时钟分频因子为 4....248
图 108 计数器时序图，内部时钟分频因子为 N....248
图 109 计数器时序图，当没有使用重复计数器时的更新事件....249
图 110 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6....250
图 111 计数器时序图，内部时钟分频因子为 2....250
图 112 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36....250
图 113 计数器时序图，内部时钟分频因子为 N....251
图 114 计数器时序图，ARPE=1 时的更新事件(计数器下溢)....251
图 115 计数器时序图，ARPE=1 时的更新事件(计数器溢出)....251
图 116 一般模式下的控制电路，内部时钟分频因子为 1....252
图 117 TI2 外部时钟连接例子....252
图 118 外部时钟模式 1 下的控制电路....253
图 119 外部触发输入框图....253
图 120 外部时钟模式 2 下的控制电路....254
图 121 捕获/比较通道(如：通道 1 输入部分)....254

图 122 捕获/比较通道 1 的主电路 255
图 123 捕获/比较通道的输出部分(通道 1) 255
图 124 PWM 输入模式时序 257
图 125 输出比较模式，翻转 OC1 258
图 126 边沿对齐的 PWM 波形(ARR=8) 259
图 127 中央对齐的 PWM 波形(APR=8) 260
图 128 单脉冲模式的例子 261
图 129 清除 TIMx 的 OCxREF 262
图 130 编码器模式下的计数器操作实例 264
图 131 IC1FP1 反相的编码器接口模式实例 264
图 132 复位模式下的控制电路 265
图 133 门控模式下的控制电路 266
图 134 触发器模式下的控制电路 266
图 135 外部时钟模式 2 + 触发模式下的控制电路 267
图 136 主/从定时器的例子 267
图 137 定时器 1 的 OC1REF 控制定时器 2 268
图 138 通过使能定时器 1 可以控制定时器 2 269
图 139 使用定时器 1 的更新触发定时器 2 269
图 140 利用定时器 1 的使能触发定时器 2 270
图 141 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2 271
图 142 通用定时器框图（TIM9/TIM12） 288
图 143 通用定时器框图（TIM10/11/12/13/14） 289
图 144 当预分频器的参数从 1 变到 2 时，计数器的时序图 290
图 145 当预分频器的参数从 1 变到 4 时，计数器的时序图 290
图 146 计数器时序图，内部时钟分频因子为 1 291
图 147 计数器时序图，内部时钟分频因子为 2 291
图 148 计数器时序图，内部时钟分频因子为 4 291
图 149 计数器时序图，内部时钟分频因子为 N 292
图 150 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入) 292
图 151 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR) 292
图 152 一般模式下的控制电路，内部时钟分频因子为 1 293
图 153 TI2 外部时钟连接例子 293
图 154 外部时钟模式 1 下的控制电路 294
图 155 捕获/比较通道(如：通道 1 输入部分) 294
图 156 捕获/比较通道 1 的主电路 295
图 157 捕获/比较通道的输出部分(通道 1) 295
图 158 PWM 输入模式时序 297
图 159 输出比较模式，翻转 OC1 298
图 160 边沿对齐的 PWM 波形(ARR=8) 299
图 161 单脉冲模式的例子 299
图 162 复位模式下的控制电路 301

图 163 门控模式下的控制电路.....301
图 164 触发器模式下的控制电路.....302
图 165 基本定时器框图.....320
图 166 预分频系数从 1 变到 2 的计数器时序图.....321
图 167 预分频系数从 1 变到 4 的计数器时序图.....322
图 168 计数器时序图，内部时钟分频系数为 1.....322
图 169 计数器时序图，内部时钟分频系数为 2.....323
图 170 计数器时序图，内部时钟分频系数为 4.....323
图 171 计数器时序图，内部时钟分频系数为 N.....323
图 172 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装载).....324
图 173 计数器时序图，当 ARPE=1 时的更新事件(预装载 TIMx_ARR).....324
图 174 普通模式时序图，内部时钟分频系数为 1.....325
图 175 简化的 RTC 框图.....331
图 176 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004.....332
图 177 RTC 溢出波形图示例，PR=0003.....332
图 178 独立看门狗框图.....339
图 179 看门狗框图.....342
图 180 窗口看门狗时序图.....343
图 181 SDIO“无响应”和“无数据”操作.....347
图 182 SDIO(多)数据块读操作.....347
图 183 SDIO(多)数据块写操作.....347
图 184 SDIO 连续读操作.....348
图 185 SDIO 连续写操作.....348
图 186 SDIO 框图.....348
图 187 SDIO 适配器.....349
图 188 控制单元.....350
图 189 SDIO 适配器命令通道.....351
图 190 命令通道状态机(CPSM).....351
图 191 SDIO 命令传输.....352
图 192 数据通道.....354
图 193 数据通道状态机(DPSM).....355
图 194 USB 设备框图.....391
图 195 分组缓冲区对应的缓冲区描述表项定位.....394
图 196 CAN 网拓扑结构.....413
图 197 bxCAN 工作模式.....415
图 198 bxCAN 工作在静默模式.....415
图 199 bxCAN 工作在环回模式.....416
图 200 bxCAN 工作在环回静默模式.....416
图 201 发送邮箱状态.....418
图 202 接收 FIFO 状态.....419
图 203 过滤器组位宽设置 - 寄存器组织.....421

图 204 过滤器编号的例子 422
图 205 过滤器机制的例子 423
图 206 CAN 错误状态图 424
图 207 位时序 425
图 208 各种 CAN 帧 426
图 209 事件标志和中断产生 427
图 210 SPI 框图 446
图 211 单主和单从应用 447
图 212 硬件/软件的从选择管理 448
图 213 数据时钟时序图 449
图 214 主模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图 454
图 215 从模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图 454
图 216 主设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，，TXE/BSY 变化示意图 455
图 217 从设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，TXE/BSY 变化示意图 455
图 218 只接收模式(BIDIMODE=0 并且 RXONLY=1)下连续传输时，RXNE 变化示意图 456
图 219 非连续传输发送(BIDIMODE=0 并且 RXONLY=0)时，TXE/BSY 变化示意图 456
图 220 使用 DMA 发送 460
图 221 使用 DMA 接收 460
图 222 I2S 框图 462
图 223 I2S 飞利浦协议波形(16/32 位全精度，CPOL=0) 464
图 224 I2S 飞利浦协议标准波形(24 位帧，CPOL=0) 464
图 225 发送 0x8EAA33 464
图 226 接收 0x8EAA33 465
图 227 I2S 飞利浦协议标准波形(16 位扩展至 32 位包帧，CPOL=0) 465
图 228 示例 465
图 229 MSB 对齐 16 位或 32 位全精度，CPOL=0 466
图 230 MSB 对齐 24 位数据，CPOL=0 466
图 231 MSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0 466
图 232 LSB 对齐 16 位或 32 位全精度，CPOL=0 467
图 233 LSB 对齐 24 位数据，CPOL=0 467
图 234 要求发送 0x3478AE 的操作 467
图 235 要求接收 0x3478AE 的操作 467
图 236 LSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0 468
图 237 示例 468
图 238 PCM 标准波形(16 位) 468
图 239 PCM 标准波形(16 位扩展到 32 位包帧) 469
图 240 音频采样频率定义 469

图 241 I2S 时钟发生器结构.....469
图 242 I²C 总线协议.....483
图 243 I²C 的功能框图.....484
图 244 从发送器的传送序列图.....485
图 245 从接收器的传送序列图.....486
图 246 主发送器传送序列图.....487
图 247 主接收器传送序列图.....488
图 248 I²C 中断映射图.....494
图 249 USART 框图.....507
图 250 字长设置.....508
图 251 配置停止位.....509
图 252 发送时 TC/TXE 的变化情况.....510
图 253 起始位侦测.....511
图 254 检测噪声的数据采样.....512
图 255 利用空闲总线检测的静默模式.....516
图 256 利用地址标记检测的静默模式.....516
图 257 LIN 模式下的断开检测(11 位断开长度-设置了 LBDL 位).....518
图 258 LIN 模式下的断开检测与帧错误的检测.....519
图 259 USART 同步传输的例子.....520
图 260 USART 数据时钟时序示例(M=0).....520
图 261 USART 数据时钟时序示例(M=1).....520
图 262 RX 数据采样/保持时间.....521
图 263 ISO7816-3 异步协议.....522
图 264 使用 1.5 停止位检测奇偶检验错.....523
图 265 IrDA SIR ENDEC 框图.....524
图 266 IrDA 数据调制(3/16)普通模式.....524
图 267 利用 DMA 发送.....525
图 268 利用 DMA 接收.....526
图 269 两个 USART 间的硬件流控制.....526
图 270 RTS 流控制.....527
图 271 CTS 流控制.....527
图 272 USART 中断映像图.....528
图 273 框图.....539
图 274 OTG 的 A-B 设备连接.....540
图 275 单纯的 USB 外设连接.....542
图 276 单纯的 USB 主机连接.....545
图 277 SOF 连接.....549
图 278 动态更新 OTG FS HFIR.....551
图 279 OTG_FS 控制器框图.....552
图 280 设备模式下的 FIFO 地址映像和 AHB 的 FIFO 操作映像.....553
图 281 主机模式 FIFO 地址映像和 AHB 的 FIFO 操作映像.....554

图 282 中断架构 557
图 283 CSR 存储器映像 559
图 284 发送 FIFO 写任务 612
图 285 接收 FIFO 读出任务 612
图 286 普通的块(Bulk)/控制(Control)的 OUT/SETUP 和块/控制的 IN 传输过程 614
图 287 块(Bulk)/控制(Control)的 IN 传输过程 617
图 288 普通的中断(Interrupt)OUT/IN 传输过程 619
图 289 普通的同步(Isochronous)OUT/IN 传输过程 623
图 290 在从模式下读出接收 FIFO 的数据报文 628
图 291 处理一个 SETUP 数据报文 630
图 292 从模式下块(Bulk)OUT 传输过程 635
图 293 TRDT 最大时序的情况 642
图 294 A 设备的 SRP 643
图 295 B 类设备 SRP 644
图 296 A 类设备 HNP 645
图 297 B 类设备 HNP 646
图 298 W55MH32 级别和 Cortex™-M3 级别的调试框图 650
图 299 SWJ 调试端口 651
图 300 JTAG TAP 连接 655
图 301 TPIU 框图 667

列表清单

表 1 寄存器组起始地址 34
表 2 闪存模块组织 37
表 3 启动模式 38
表 4 CRC 计算单元寄存器映像和复位值 42
表 5 低功耗模式一览 46
表 6 SLEEP-NOW 模式 47
表 7 SLEEP-ON-EXIT 模式 47
表 8 停止模式 48
表 9 待机模式 49
表 10 PWR 寄存器地址映像和复位值 52
表 11 BKP 寄存器映像和复位值 56
表 12 RCC 寄存器地址映像和复位值 80
表 13 端口位配置表 83
表 14 输出模式位 83
表 15 高级定时器 TIM1/TIM8 87
表 16 通用定时器 TIM2/3/4/5 87
表 17 USARTs 87
表 18 SPI 88
表 19 I2S 88
表 20 I2C 接口 88
表 21 BxCAN 88
表 22 USB 88
表 23 SDIO 88
表 24 ADC/DAC 89
表 25 其它 I/O 功能 89
表 26 CAN1 复用功能重映射 93
表 27 调试接口信号 93
表 28 调试端口映像 93
表 29 ADC1 外部触发注入转换复用功能重映射 93
表 30 ADC1 外部触发规则转换复用功能重映射 93
表 31 ADC2 外部触发注入转换复用功能重映射 94
表 32 ADC2 外部触发规则转换复用功能重映射 94
表 33 TIM5 复用功能重映像 94
表 34 TIM4 复用功能重映像 94
表 35 TIM3 复用功能重映像 94
表 36 TIM2 复用功能重映像 94
表 37 TIM1 复用功能重映像 94
表 38 TIM9 重映射 (1) 95
表 39 TIM10 重映射 (1) 95

表 40 TIM11 重映射 (1) 95
表 41 TIM13 重映射 (1) 95
表 42 TIM14 重映射 (1) 95
表 43 USART3 重映像 95
表 44 USART2 重映像 95
表 45 USART1 重映像 96
表 46 I2C1 重映像 96
表 47 SPI1 重映像 96
表 48 SPI3 重映射 96
表 49 GPIO 寄存器地址映像和复位值 101
表 50 AFIO 寄存器地址映像和复位值 102
表 51 W55MH32 产品向量表 103
表 52 外部中断/事件控制器寄存器映像和复位值 111
表 53 通用寄存器的偏移地址 114
表 54 Socket n 寄存器中的偏移地址(0≤n≤7) 115
表 55 可编程的数据传输宽度和大小端操作(当 PINC=MINC=1) 135
表 56 DMA 中断请求 136
表 57 各个通道的 DMA1 请求一览 138
表 58 各个通道的 DMA2 请求一览 139
表 59 DMA 寄存器映像和复位 143
表 60 ADC 引脚 147
表 61 模拟看门狗通道选择 149
表 62 ADC1 和 ADC2 用于规则通道的外部触发 152
表 63 ADC1 和 ADC2 用于注入通道的外部触发 152
表 64 ADC3 用于规则通道的外部触发 153
表 65 ADC3 用于注入通道的外部触发 153
表 66 ADC 中断 161
表 67 ADC 寄存器映像和复位值 170
表 68 DAC 引脚 173
表 69 外部触发 175
表 70 DAC 寄存器映像 188
表 71 计数方向与编码器信号的关系 215
表 72 TIMx 内部触发连接 225
表 73 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位 233
表 74 TIM1 和 TIM8 寄存器和复位值 240
表 75 计数方向与编码器信号的关系 263
表 76 TIMx 内部触发连接 274
表 77 标准 OCx 通道的输出控制位 282
表 78 TIM2 到 TIM5 寄存器和复位值 285
表 79 TIMx 内部触发连接 304
表 80 标准 OCx 通道的输出控制位 309

表 81 TIM9/12 寄存器和复位值.....311
表 82 标准 OCx 通道的输出控制位.....316
表 83 TIM10/11/13/14 寄存器和复位值.....319
表 84 TIM6 和 TIM7 寄存器和复位值.....329
表 85 RTC-寄存器映像和复位值.....337
表 86 看门狗超时时间(40kHz 的输入时钟(LSI).....339
表 87 IWDG 寄存器映像和复位值.....341
表 88 WWDG 寄存器映像和复位值.....345
表 89 SDIO 引脚定义.....349
表 90 命令格式.....352
表 91 短响应格式.....353
表 92 长响应格式.....353
表 93 命令通道状态标志.....353
表 94 数据令牌格式.....356
表 95 发送 FIFO 状态标志.....356
表 96 接收 FIFO 状态标志.....357
表 97 卡状态.....364
表 98 SD 状态.....366
表 99 速度类型代码.....367
表 100 移动性能代码.....367
表 101 AU_SIZE 代码.....368
表 102 最大的 AU 长度.....368
表 103 ERASE_SIZE 代码.....368
表 104 擦除超时代码.....368
表 105 擦除偏移代码.....369
表 106 基于块传输的写命令.....371
表 107 基于块传输的写保护命令.....371
表 108 擦除命令.....372
表 109 I/O 模式命令.....372
表 110 上锁命令.....372
表 111 应用相关命令.....372
表 112 R1 响应.....373
表 113 R2 响应.....373
表 114 R3 响应.....374
表 115 R4 响应.....374
表 116 R4b 响应.....374
表 117 R5 响应.....375
表 118 R6 响应.....375
表 119 响应类型和 SDIO_RESPx 寄存器.....380
表 120 SDIO 寄存器映像.....388
表 121 双缓冲批量端点缓冲区标识定义.....397

表 122 双缓冲批量端点的缓冲区使用标识 397
表 123 同步端点的缓冲区使用标识 398
表 124 唤醒事件检测 400
表 125 接收状态编码 407
表 126 端点类型编码 407
表 127 端点特殊类型定义 407
表 128 发送状态编码 407
表 129 分组缓冲区大小的定义 409
表 130 USB 寄存器映像和复位值 409
表 131 发送邮箱寄存器列表 423
表 132 接收邮箱寄存器列表 424
表 133 bxCAN - 寄存器列表及其复位值 442
表 134 SPI 中断请求 461
表 135 使用标准的 8MHzHSE 时钟得到精确的音频频率 470
表 136 I^2S 中断请求 474
表 137 SPI 寄存器列表及其复位值 481
表 138 SMBus 与 I^2C 的比较 490
表 139 I^2C 中断请求表: 494
表 140 I2C 寄存器地址映象和复位值 503
表 141 检测噪声的数据采样 513
表 142 设置波特率时的误差计算 514
表 143 当 DIV_Fraction=0 时，USART 接收器的容忍度 515
表 144 当 DIV_Fraction!=0 时，USART 接收器的容忍度 515
表 145 帧格式 517
表 146 USART 模式设置(1) 528
表 147 USART 寄存器列表及其复位值 536
表 148 OTG 引脚 539
表 149 W55MH32 低功耗和 OTG 的兼容性 550
表 150 控制器全局控制和状态寄存器(CSRs) 559
表 151 主机模式下的控制和状态寄存器(CSRs) 560
表 152 设备模式控制和状态寄存器 560
表 153 数据 FIFO(DFIFO)访问寄存器 561
表 154 供电和门控控制和状态寄存器 561
表 155 TRDT 值 566
表 156 OTG_FS 模块的寄存器及其复位值 601
表 157 SWJ 调试端口引脚 652
表 158 灵活的 SWJ_DP 引脚分配 653
表 159 JTAG 调试端口数据寄存器 656
表 160 由 A[3:2]定义的 32 位调试接口寄存器地址 657
表 161 请求包(8 比特位) 658
表 162 ACK 定义(3 比特位) 658

表 163 传输数据(33 比特位) 658
表 164 SW-DP 寄存器 659
表 165 Cortex-M3AHB-AP 寄存器..... 660
表 166 内核调试寄存器..... 660
表 167 主要的 ITM 寄存器..... 662
表 168 主要的 ETM 寄存器 663
表 169 异步跟踪引脚分配 668
表 170 同步跟踪引脚分配 668
表 171 灵活的跟踪引脚分配 668
表 172 重要的 TPIU 寄存器..... 671
表 173 DBG-寄存器和复位值..... 671

1 文中的缩写

在对寄存器的描述中使用了下列缩写：

术语	描述
read/write(rw)	软件能读写此位。
read-only(r)	软件只能读此位。
write-only(w)	软件只能写此位，读此位将返回复位值。
read/clear(rc_w1)	软件可以读此位，也可以通过写'1'清除此位，写'0'对此位无影响。
read/clear(rc_w0)	软件可以读此位，也可以通过写'0'清除此位，写'1'对此位无影响。
read/clear by read(rc_r)	软件可以读此位，读此位将自动地清除它为'0'，写'0'对此位无影响。
read/set(rs)	软件可以读也可以设置此位，写'0'对此位无影响。
read-only write trigger(rt_w)	软件可以读也可以设置此位：写'0'或'1'触发一个事件但对此位数值没有影响。
toggle(t)	软件只能通过写'1'来翻转此位，写'0'对此位无影响。
reserved(Res.)	保留位，必须保持默认值不变。

2 存储器和总线架构

2.1 系统框架

主系统由以下部分构成：

- 四个驱动单元：
  - Cortex™-M3 内核 DCode 总线(D-bus)，和系统总线(S-bus)
  - 通用 DMA1 和通用 DMA2
- 三个被动单元
  - 内部 SRAM
  - 内部闪存存储器
  - AHB 到 APB 的桥(AHB2APBx)，它连接所有的 APB 设备

这些都是通过一个多级的 AHB 总线构架相互连接的，如下图图 1 所示：

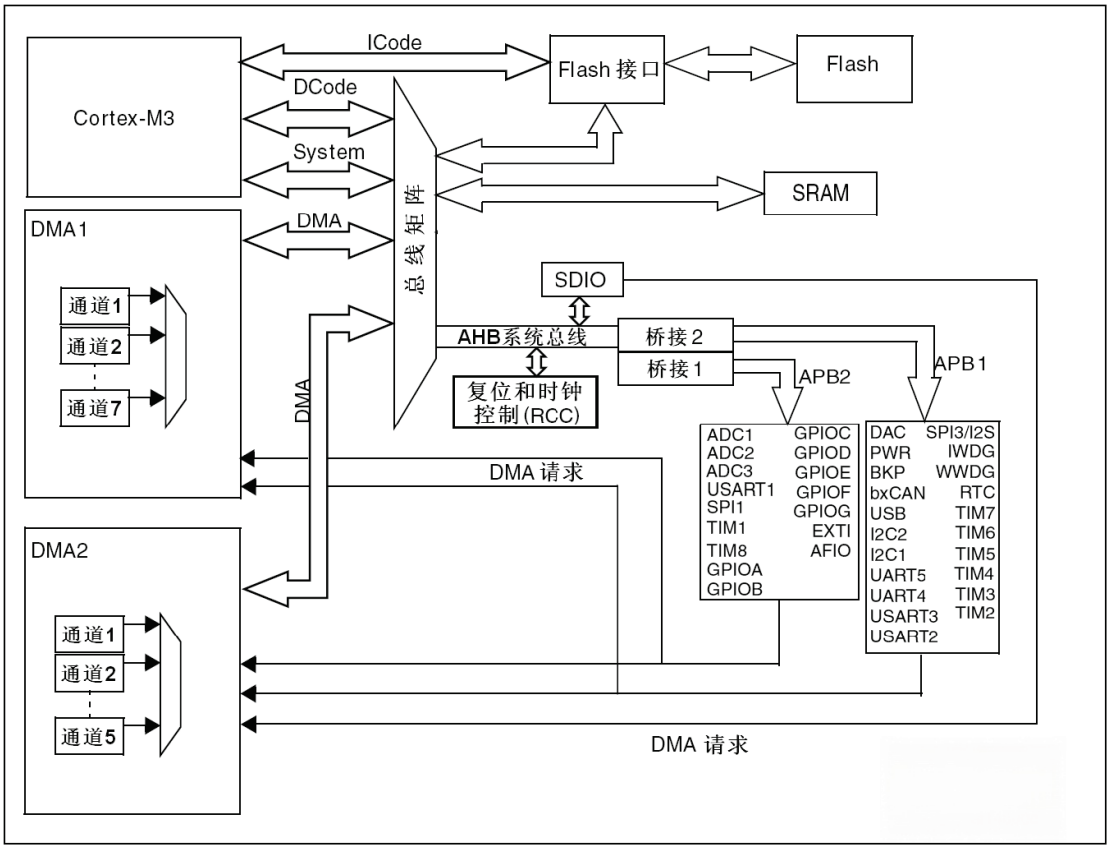


图 1 系统结构

ICode 总线

该总线将 Cortex™-M3 内核的指令总线与闪存指令接口相连接。指令预取在此总线上完成。

DCode 总线

该总线将 Cortex™-M3 内核的 DCode 总线与闪存存储器的数据接口相连接(常量加载和调试访问)。

系统总线

此总线连接 Cortex™-M3 内核的系统总线(外设总线)到总线矩阵，总线矩阵协调着内核和 DMA 间的访问。

DMA 总线

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

总线矩阵

总线矩阵协调内核系统总线和 DMA 主控总线之间的访问仲裁，仲裁利用轮换算法。总线矩阵包含 4 个驱动部件(CPU 的 DCode、系统总线、DMA1 总线和 DMA2 总线)和 3 个被动部件(闪存存储器接口、SRAM 和 AHB2APB 桥)。

AHB 外设通过总线矩阵与系统总线相连，允许 DMA 访问。

AHB/APB 桥(APB)

两个 AHB/APB 桥在 AHB 和 2 个 APB 总线间提供同步连接。APB1 操作速度限于 108MHz，APB2 操作于全速(最高 216MHz)。

有关连接到每个桥的不同外设的地址映射请参考表 1。在每一次复位以后，所有除 SRAM 和 FLITF 以外的外设都被关闭，在使用一个外设之前，必须设置寄存器 RCC_AHBENR 来打开该外设的时钟。

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

2.2 存储器组织

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4GB 的线性地址空间内。

数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

外设寄存器的映像请参考相关章节。

可访问的存储器空间被分成 8 个主要块，每个块为 512MB。

其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间。

2.3 存储器映像

表 1 列出了所用 W55MH32 中内置外设的起始地址。

表 1 寄存器组起始地址

起始地址	外设	总线	寄存器映像
0x5000 0000–0x5003 FFFF	USB OTG全速	AHB	参见26.16.6节
0x4003 0000–0x4FFF FFFF	保留		
0x4002 8000–0x4002 9FFF	保留		
0x4002 3400-0x4002 7FFF	保留	AHB	
0x4002 3000-0x4002 33FF	CRC		参见3.4.4节
0x4002 2400-0x4002 2FFF	保留		
0x4002 2000-0x4002 23FF	闪存存储器接口		
0x4002 1400-0x4002 1FFF	保留		
0x4002 1000-0x4002 13FF	复位和时钟控制(RCC)		参见6.3.11节
0x4002 0800-0x4002 0FFF	保留		
0x4002 0400-0x4002 07FF	DMA2		参见10.4.7节
0x4002 0000-0x4002 03FF	DMA1		参见10.4.7节
0x4001 8400-0x4001 FFFF	保留		
0x4001 8000-0x4001 83FF	SDIO		参见20.9.16节

起始地址	外设	总线	寄存器映像
0x4001 5800-0x4001 7FFF	保留		
0x4001 5400-0x4001 57FF	TIM11定时器		参见15.5.11节
0x4001 5000-0x4001 53FF	TIM10定时器		参见15.5.11节
0x4001 4C00-0x4001 4FFF	TIM9定时器		参见15.4.13节
0x4001 4000-0x4001 4BFF	保留	APB2	
0x4001 3C00-0x4001 3FFF	ADC3		参见11.12.15节
0x4001 3800-0x4001 3BFF	USART1		参见25.6.8节
0x4001 3400-0x4001 37FF	TIM8定时器		参见13.4.21节
0x4001 3000-0x4001 33FF	SPI1		参见23.5.10节
0x4001 2C00-0x4001 2FFF	TIM1定时器		参见13.4.21节
0x4001 2800-0x4001 2BFF	ADC2		参见11.12.15节
0x4001 2400-0x4001 27FF	ADC1		参见11.12.15节
0x4001 2000-0x4001 23FF	GPIO端口G		参见7.5节
0x4001 1C00-0x4001 1FFF	GPIO端口F		参见7.5节
0x4001 1800-0x4001 1BFF	GPIO端口E		参见7.5节
0x4001 1400-0x4001 17FF	GPIO端口D		参见7.5节
0x4001 1000-0x4001 13FF	GPIO端口C		参见7.5节
0X4001 0C00-0x4001 0FFF	GPIO端口B		参见7.5节
0x4001 0800-0x4001 0BFF	GPIO端口A		参见7.5节
0x4001 0400-0x4001 07FF	EXTI		参见8.3.7节
0x4001 0000-0x4001 03FF	AFIO		参见7.5节
0x4000 7800-0x4000 FFFF	保留	APB1	
0x4000 7400-0x4000 77FF	DAC		参见12.5.14节
0x4000 7000-0x4000 73FF	电源控制(PWR)		参见4.4.3节
0x4000 6C00-0x4000 6FFF	后备寄存器(BKP)		参见5.4.5节
0x4000 6800-0x4000 6BFF	保留		
0x4000 6400-0x4000 67FF	bxCAN		参见22.9.5节
0x4000 6000-0x4000 63FF	USB/CAN共享的512字节SRAM		
0x4000 5C00-0x4000 5FFF	USB全速设备寄存器		参见21.5.4节
0x4000 5800-0x4000 5BFF	I2C2		参见24.6.10节
0x4000 5400-0x4000 57FF	I2C1		参见24.6.10节
0x4000 5000-0x4000 53FF	UART5		参见25.6.8节
0x4000 4C00-0x4000 4FFF	UART4		参见25.6.8节
0x4000 4800-0x4000 4BFF	USART3		参见25.6.8节
0x4000 4400-0x4000 47FF	USART2		参见25.6.8节
0x4000 4000-0x4000 43FF	保留		
0x4000 3C00-0x4000 3FFF	SPI3/I2S3		参见23.5.10节
0x4000 3800-0x4000 3BFF	保留		
0x4000 3400-0x4000 37FF	保留		
0x4000 3000-0x4000 33FF	独立看门狗(IWDG)		参见18.4.5节
0x4000 2C00-0x4000 2FFF	窗口看门狗(WWDG)		参见19.6.4节
0x4000 2800-0x4000 2BFF	RTC		参见17.4.7节
0x4000 2400-0x4000 27FF	保留		
0x4000 2000-0x4000 23FF	TIM14定时器		参见15.5.11节
0x4000 1C00-0x4000 1FFF	TIM13定时器		参见15.5.11节
0x4000 1800-0x4000 1BFF	TIM12定时器		参见15.4.13节
0x4000 1400-0x4000 17FF	TIM7定时器		参见16.4.9节
0x4000 1000-0x4000 13FF	TIM6定时器		参见16.4.9节

起始地址	外设	总线	寄存器映像
0x4000 0C00-0x4000 0FFF	TIM5定时器		参见14.4.19节
0x4000 0800-0x4000 0BFF	TIM4定时器		参见14.4.19节
0x4000 0400-0x4000 07FF	TIM3定时器		参见14.4.19节
0x4000 0000-0x4000 03FF	TIM2定时器		参见14.4.19节

2.3.1 嵌入式 SRAM

W55MH32 内置 96K 字节的静态 SRAM。它可以以字节、半字(16 位)或全字(32 位)访问。SRAM 的起始地址是 0x2000 0000。

2.3.2 位段

Cortex™-M3 存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

在 W55MH32 里，外设寄存器和 SRAM 都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：

bit_word_addr=bit_band_base+(byte_offset×32)+(bit_number×4)

其中：

bit_word_addr 是别名存储器区中字的地址，它映射到某个目标位。

bit_band_base 是别名区的起始地址。

byte_offset 是包含目标位的字节在位段里的序号

bit_number 是目标位所在位置(0-31)

例子：

下面的例子说明如何映射别名区中 SRAM 地址为 0x2000 0300 的字节中的位 2：

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

对 0x2200 6008 地址的写操作与对 SRAM 中地址 0x2000 0300 字节的位 2 执行读-改-写操作有着相同的效果。

读 0x2200 6008 地址返回 SRAM 中地址 0x2000 0300 字节的位 2 的值(0x01 或 0x00)。

请参考《Cortex™-M3 技术参考手册》以了解更多有关位段的信息。

2.3.3 嵌入式闪存

高性能的闪存模块有以下的主要特性：

- 高达 1M 字节闪存存储器结构：闪存存储器有主存储块和信息块组成：
  - 主存储块容量：
    主存储块最大为 128K×64 位，每个存储块划分为 256 个 4K 字节的页(见表 2)
  - 信息块容量：
    258×64 位(见表 2)

闪存存储器接口的特性为：

- 带预取缓冲器的读接口(每字为 2×64 位)
- 选择字节加载器
- 闪存编程/擦除操作

- 访问/写保护

表 2 闪存模块组织

模块	名称	地址	大小
主存储块	页0	0x0800 0000 - 0x0800 0FFF	4K
	页1	0x0800 1000 - 0x0800 1FFF	4K
	页2	0x0800 2000 - 0x08002FFF	4K
	页3	0x0800 3000 - 0x08003FFF	4K
	… …	… …	
	页255	0x080F F000 - 0x080F FFFF	4K
信息块	系统存储器	0x1FFF F000 - 0x1FFF F7FF	2K
	选择字节	0x1FFF F000 - 0x1FFF F7FF	16
闪存存储器接口寄存器	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FALSH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	保留	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

闪存读取

闪存的指令和数据访问是通过 AHB 总线完成的。预取模块是用于通过 ICode 总线读取指令的。仲裁是作用在闪存接口，并且 DCode 总线上的数据访问优先。

读访问可以有以下配置选项：

- 等待时间：可以随时更改的用于读取操作的等待状态的数量。
- 预取缓冲区(2 个 64 位)：在每一次复位以后被自动打开，由于每个缓冲区的大小(64 位)与闪存的带宽相同，因此只通过需一次读闪存的操作即可更新整个缓冲区的内容。由于预取缓冲区的存在，CPU 可以工作在更高的主频。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。
- 半周期：用于功耗优化。

注: 1.这些选项应与闪存存储器的访问时间一起使用。等待周期体现了系统时钟(SYSCLK)频率与闪存访问时间的关系：

0 等待周期，当 0<SYSCLK<24MHz

1 等待周期，当 24MHz<SYSCLK≤48MHz

2 等待周期，当 48MHz<SYSCLK≤72MHz

2.半周期配置不能与使用了预分频器的 AHB 一起使用，时钟系统应该等于 HCLK 时钟。该特性只能用在时钟频率为 8MHz 或低于 8MHz 时，可以直接使用的内部 RC 振荡器(HSI)，或者是主振荡器(HSE)，但不能用 PLL。

3.当 AHB 预分频系数不为 1 时，必须置预取缓冲区处于开启状态。

4.只有在系统时钟(SYSCLK)小于 24MHz 并且没有打开 AHB 的预分频器(即 HCLK 必须等于 SYSHCLK)时，才能执行预取缓冲器的打开和关闭操作。一般而言，在初始化过程中执行预取缓冲器的打开和关闭操作，这时微控制器的时钟由 8MHz 的内部 RC 振荡器(HSI)提供。

5.使用 DMA：DMA 在 DCode 总线上访问闪存存储器，它的优先级比 ICode 上的取指高。DMA 在每次传送完成后具有一个空余的周期。有些指令可以和 DMA 传输一起执行。

编程和擦除闪存

闪存编程一次可以写入 16 位(半字)。

闪存擦除操作可以按页面擦除或完全擦除(全擦除)。全擦除不影响信息块。

为了确保不发生过度编程，闪存编程和擦除控制器块是由一个固定的时钟控制的。

写操作(编程或擦除)结束时可以触发中断。仅当闪存控制器接口时钟开启时，此中断可以用来从 WFI 模式退出。

源控制/状态寄存器(PWR_CSR)

地址偏移：0x00

复位值：0x0000 0030

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

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

位	符号	说明
31:6	Reserved	保留
5	PRFTBS	**PRFTBS**：预取缓冲区状态 0： 预取缓冲区禁用 1：预取缓冲区启用
4	PRFTBE	**PRFTBE**：启用预取缓冲区 0：预取缓冲区禁用 1：预取缓冲区启用
3	HLFCYA	**HLFCYA:**启用闪存半周期访问 0：半周期被禁止 1：半周期被启用
2:0	LATENCY	**LATENCY:**延迟 这些位表示系统时钟周期与 Flash 访问时间比率 000 零等待状态 0 < SYSCLK<=24 MHz 001 一个等待状态 24 MHz < SYSCLK <=48 MHz 010 两种等待状态 48 MHz < SYSCLK <=72 MHz

2.4 启动配置

在 W55MH32 里，可以通过 BOOT[1:0]引脚选择三种不同启动模式。

表 3 启动模式

启动模式选择引脚		启动模式	说明
BOOT1	BOOT0		
X	0	主闪存存储器	主闪存存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域
1	1	内置 SRAM	内置 SRAM 被选为启动区域

在系统复位后，SYSCLK 的第 4 个上升沿，BOOT 引脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，来选择在复位后的启动模式。

在从待机模式退出时，BOOT 引脚的值将被被重新锁存；因此，在待机模式下 BOOT 引脚应保持为需要的启动配置。在启动延迟之后，CPU 从地址 0x0000 0000 获取堆栈顶的地址，并从启动存储器的 0x0000 0004 指示的地址开始执行代码。

因为固定的存储器映像，代码区始终从地址 0x0000 0000 开始(通过 ICode 和 DCode 总线访问)，而数据区(SRAM)始终从地址 0x2000 0000 开始(通过系统总线访问)。Cortex-M3 的 CPU 始终从

ICode 总线获取复位向量，即启动仅适合于从代码区开始(典型地从 Flash 启动)。W55MH32 微控制器实现了一个特殊的机制，系统可以不仅仅从 Flash 存储器或系统存储器启动，还可以从内置 SRAM 启动。

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

- 从主闪存存储器启动：主闪存存储器被映射到启动空间(0x0000 0000)，但仍然能够在它原有的地址(0x0800 0000)访问它，即闪存存储器的内容可以在两个地址区域访问，0x0000 0000 或 0x0800 0000。
- 从系统存储器启动：系统存储器被映射到启动空间(0x0000 0000)，但仍然能够在它原有的地址(互联型产品原有地址为 0x1FFF B000，其它产品原有地址为 0x1FFF F000)访问它。
- 从内置 SRAM 启动：只能在 0x2000 0000 开始的地址区访问 SRAM。

注意： *当从内置 SRAM 启动，在应用程序的初始化代码中，必须使用 NVIC 的异常表和偏移寄存器，重新映射向量表至 SRAM 中。*

内嵌的自举程序

内嵌的自举程序存放在系统存储区，在生产线上写入，用于通过可用的串行接口对闪存存储器进行重新编程。

3 CRC 计算单元

3.1 CRC 简介

循环冗余校验(CRC)计算单元是根据固定的生成多项式得到任一 32 位全字的 CRC 计算结果。

在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准 EN/IEC60335-1 即提供了一种核实闪存存储器完整性的方法。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

3.2 CRC 主要特性

- 使用 CRC-32(以太网)多项式：0x4C11DB7
  - X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X4+X2+X+1
- 一个 32 位数据寄存器用于输入/输出
- CRC 计算时间：4 个 AHB 时钟周期(HCLK)
- 通用 8 位寄存器(可用于存放临时数据)

下图为 CRC 计算单元框图

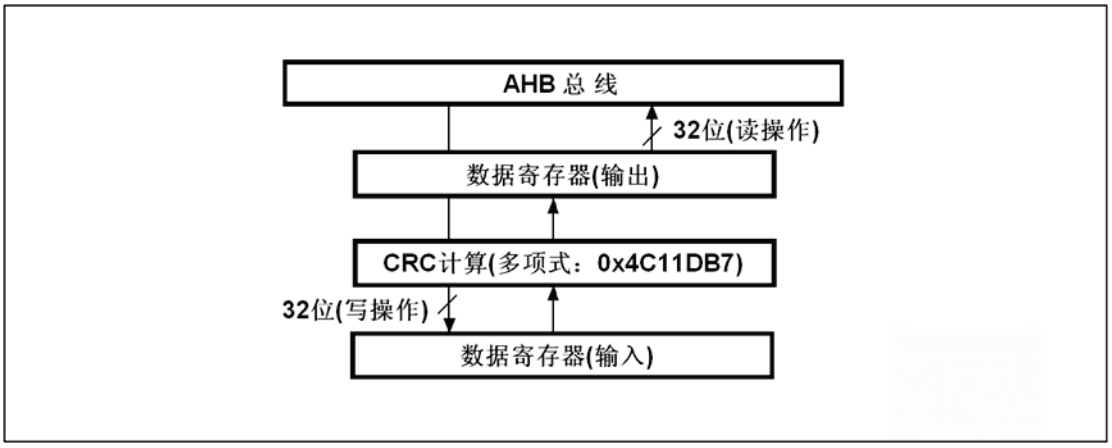


图 2 CRC 计算单元框图

3.3 CRC 功能描述

CRC 计算单元含有 1 个 32 位数据寄存器：

- 对该寄存器进行写操作时，作为输入寄存器，可以输入要进行 CRC 计算的新数据。
- 对该寄存器进行读操作时，返回上一次 CRC 计算的结果。

每一次写入数据寄存器，其计算结果是前一次 CRC 计算结果和新计算结果的组合(对整个 32 位字进行 CRC 计算，而不是逐字节地计算)。

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

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

3.4 CRC 寄存器

CRC 计算单元包括 2 个数据寄存器和 1 个控制寄存器

3.4.1 数据寄存器(CRC_DR)

地址偏移：0x00

复位值：0xFFFF FFFF

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

位	符号	说明
31:0	CRC_DR	数据寄存器位 写入 CRC 计算器的新数据时，作为输入寄存器读取时返回 CRC 计算的结果

3.4.2 独立数据寄存器(CRC_IDR)

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
保留															

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

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

3.4.3 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	28	17	16
保留															

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

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

3.4.4 CRC 寄存器映像

下表列出了 CRC 的寄存器映像和复位值

表 4 CRC 计算单元寄存器映像和复位值

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

关于寄存器的起始地址，参见表 1

4 电源控制(PWR)

4.1 电源

W55MH32 的工作电压(VDD)为 2.0～3.6V。通过内置的电压调节器提供所需的 1.8V 电源。当主电源 VDD 掉电后，通过 VBAT 脚为实时时钟(RTC)和备份寄存器提供电源。

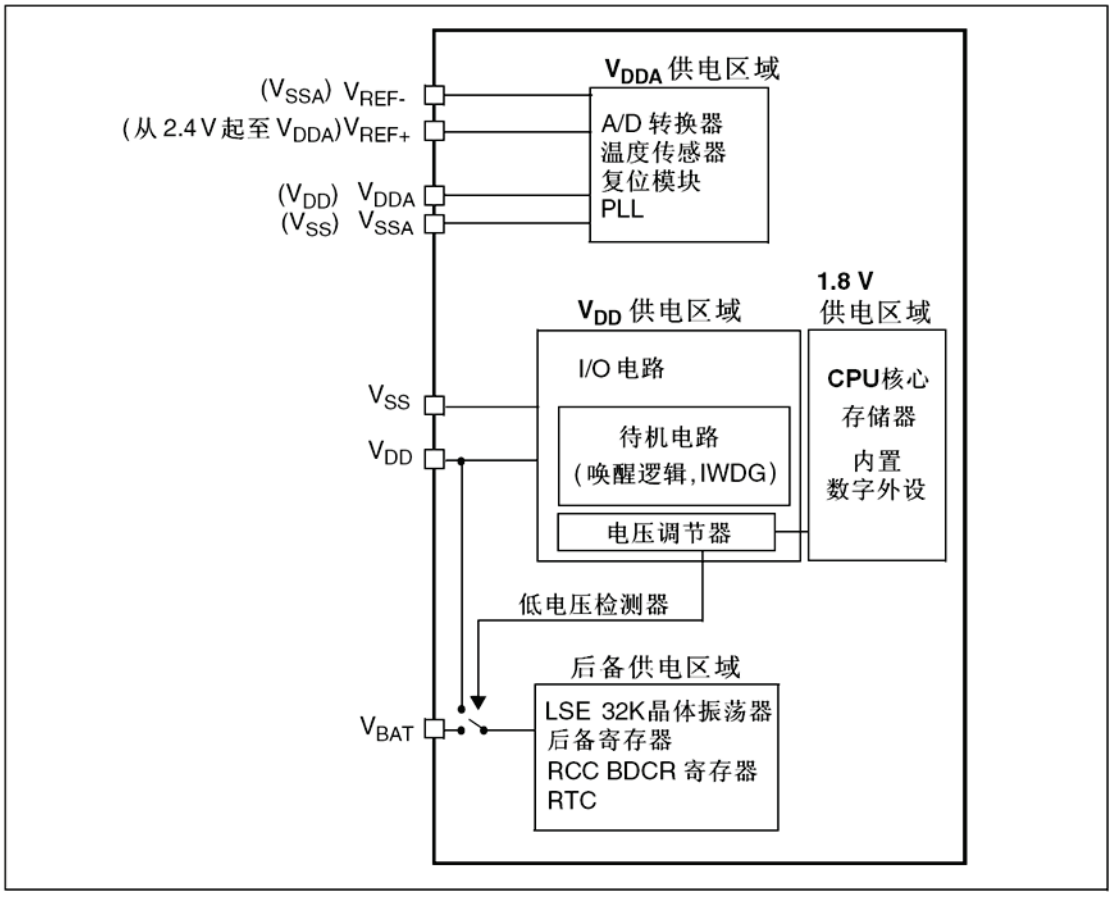


图 3 电源框图

注：VDDA 和 VSSA 必须分别联到 VDD 和 VSS。

4.1.1 独立的 A/D 转换器供电和参考电压

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

- ADC 的电源引脚为 VDDA
- 独立的电源地 VSSA

如果有 VREF-引脚(根据封装而定)，它必须连接到 VSSA。

为了确保输入为低压时获得更好精度，用户可以连接一个独立的外部参考电压 ADC 到 VREF+和 VREF-脚上。在 VREF+的电压范围为 2.4V～VDDA。

4.1.2 电池备份区域

使用电池或其他电源连接到 VBAT 脚上，当 VDD 断电时，可以保存备份寄存器的内容和维持 RTC 的功能。

VBAT 脚为 RTC、LSE 振荡器和 PC13 至 PC15 端口供电，可以保证当主电源被切断时 RTC 能继续工作。切换到 VBAT 供电的开关，由复位模块中的掉电复位功能控制。

警告：

在 VDD 上升阶段(tRSTTEMPO)或者探测到 PDR(掉电复位)之后，VBAT 和 VDD 之间的电源开关仍会保持连接在 VBAT。

在 VDD 上升阶段，如果 VDD 在小于 tRSTTEMPO 的时间内达到稳定状态(关于 tRSTTEMPO 数值可参考数据手册中的相关部分)，且 VDD>VBAT+0.6V 时，电流可能通过 VDD 和 VBAT 之间的内部二极管注入到 VBAT。

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

如果在应用中没有外部电池，建议 VBAT 在外部连接到 VDD 并连接一个 100nF 的陶瓷滤波电容。

当备份区域由 VDD(内部模拟开关连到 VDD)供电时，下述功能可用：

- PC14 和 PC15 可以用于 GPIO 或 LSE 引脚
- PC13 可以作为通用 I/O 口、TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出(参见第 5 章：备份寄存器(BKP))

注：因为模拟开关只能通过少量的电流(3mA)，在输出模式下使用 PC13 至 PC15 的 I/O 口功能是有限制的：速度必须限制在 2MHz 以下，最大负载为 30pF，而且这些 I/O 口绝对不能当作电流源(如驱动 LED)。

当后备区域由 VBAT 供电时(VDD 消失后模拟开关连到 VBAT)，可以使用下述功能：

- PC14 和 PC15 只能用于 LSE 引脚
- PC13 可以作为 TAMPER 引脚、RTC 闹钟或秒输出(参见第 5.4.2 节：RTC 时钟校准寄存器(BKP_RTCCR))

4.1.3 电压调节器

复位后调节器总是使能的。根据应用方式它以 3 种不同的模式工作。

- 运转模式：调节器以正常功耗模式提供 1.8V 电源(内核，内存和外设)。
- 停止模式：调节器以低功耗模式提供 1.8V 电源，以保存寄存器和 SRAM 的内容。
- 待机模式：调节器停止供电。除了备用电路和备份域外，寄存器和 SRAM 的内容全部丢失。

4.2 电源管理器

4.2.1 上电复位(POR)和掉电复位(PDR)

W55MH32 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 2V 时系统既能正常工作。

当 VDD/VDDA 低于指定的限位电压 VPOR/VPDR 时，系统保持为复位状态，而无需外部复位电路。

关于上电复位和掉电复位的细节请参考数据手册的电气特性部分。

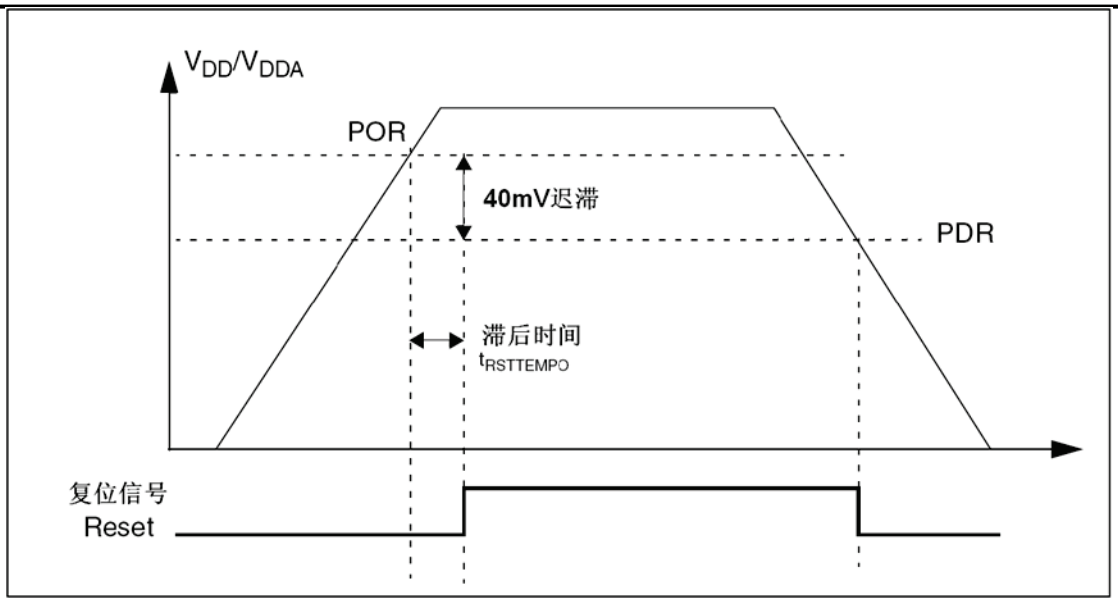


图 4 上电复位和掉电复位的波形图

4.2.2 可编程电压监测器(PVD)

用户可以利用 PVD 对 VDD 电压与电源控制寄存器(PWR_CR)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阀值。

通过设置 PVDE 位来使能 PVD。

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

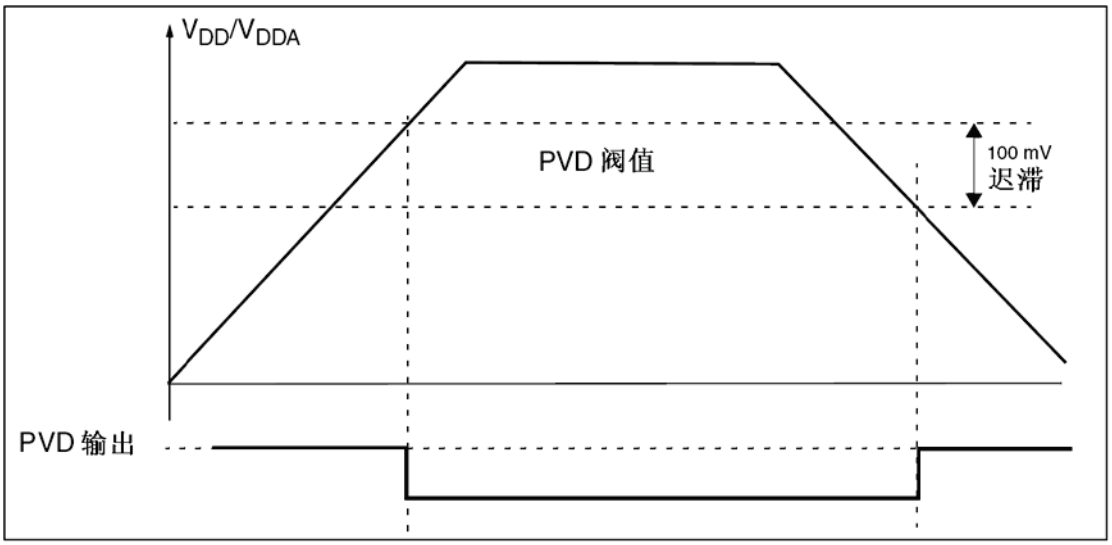


图 5 PVD 的门限

4.3 低功耗模式

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

三种低功耗模式：

- 睡眠模式(Cortex™-M3 内核停止，所有外设包括 Cortex-M3 核心的外设，如 NVIC、系统时钟(SysTick)等仍在运行)
- 停止模式(所有的时钟都已停止)
- 待机模式(1.8V 电源关闭)

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

- 降低系统时钟
- 关闭 APB 和 AHB 总线上未被使用的外设时钟。

表 5 低功耗模式一览

模式	进入	唤醒	对 1.8V 区域时钟的影响	对 V_{DD} 区域时钟的影响	电压调节器
睡眠 (SLEEP-NOW 或 SLEEP-ON-EXIT)	WFI	任一中断	CPU 时钟关，对其他时钟和 ADC 时钟无影响	无	开
	WFE	唤醒事件			
停机	PDDS 和 LPDS 位+SLEEPDEEP 位+WFI 或 WFE	任一外部中断(在外部中断寄存器中设置)	关闭所有 1.8V 区域的时钟	HSI 和 HSE 的振荡器关闭	开启或处于低功耗模式(依据电源控制寄存器(PWR_CR)的设定)
待机	PDDS 位+SLEEPDEEP 位+WFI 或 WFE	WKUP 引脚的上升沿、RTC 闹钟事件、NRST 引脚上的外部复位、IWDG 复位			关

4.3.1 降低系统时钟

在运行模式下，通过对预分频寄存器进行编程，可以降低任意一个系统时钟(SYSCLK、HCLK、PCLK1、PCLK2)的速度。进入睡眠模式前，也可以利用预分频器来降低外设的时钟。详见第 6.3.2 节：时钟配置寄存器(RCC_CFGR)。

4.3.2 外部时钟的控制

在运行模式下，任何时候都可以通过停止为外设和内存提供时钟(HCLK 和 PCLKx)来减少功耗。为了在睡眠模式下更多地减少功耗，可在执行 WFI 或 WFE 指令前关闭所有外设的时钟。

通过设置 AHB 外设时钟使能寄存器(RCC_AHBENR)、APB2 外设时钟使能寄存器(RCC_APB2ENR)和 APB1 外设时钟使能寄存器(RCC_APB1ENR)来开关各个外设模块的时钟。

4.3.3 睡眠模式

进入睡眠模式

通过执行 WFI 或 WFE 指令进入睡眠状态。根据 Cortex™-M3 系统控制寄存器中的 SLEEPONEXIT 位的值，有两种选项可用于选择睡眠模式进入机制：

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

在睡眠模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

关于如何进入睡眠模式，更多的细节参考表 6 和表 7。

退出睡眠模式

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

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

- 在外设控制寄存器中使能一个中断，而不是在 NVIC(嵌套向量中断控制器)中使能，并且在 Cortex-M3 系统控制寄存器中使能 SEVONPEND 位。当 MCU 从 WFE 中唤醒后，外设的中断挂起位和外设的 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)必须被清除。
- 配置一个外部或内部的 EXIT 线为事件模式。当 MCU 从 WFE 中唤醒后，因为与事件线对应的挂起位未被设置，不必清除外设的中断挂起位或外设的 NVIC 中断通道挂起位。

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。关于如何退出睡眠模式，更多的细节参考表 6 和表 7。

表 6 SLEEP-NOW 模式

SLEEP-NOW 模式	说明
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： – SLEEPDEEP=0 和 – SLEEPONEXIT=0 参考 Cortex-M3 系统控制寄存器。
退出	如果执行 WFI 进入睡眠模式： 中断：参考中断向量表(表 51) 如果执行 WFE 进入睡眠模式： 唤醒事件：参考唤醒事件管理(第 8.2.3 节)

表 7 SLEEP-ON-EXIT 模式

SLEEP-ON_EXIT 模式	说明
进入	在以下条件下执行 WFI 指令： – SLEEPDEEP=0 和 – SLEEPONEXIT=1 参考 Cortex™-M3 系统控制寄存器
退出	中断：参考中断向量表(表 51)
唤醒延时	无

4.3.4 停止模式

停止模式是在 Cortex™-M3 的深睡眠模式基础上结合了外设的时钟控制机制，在停止模式下电压调节器可运行在正常或低功耗模式。此时在 1.8V 供电区域的的所有时钟都被停止，PLL、HSI 和 HSERC 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

在停止模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

进入停止模式

关于如何进入停止模式，详见表 8。

在停止模式下，通过设置电源控制寄存器(PWR_CR)的 LPDS 位使内部调节器进入低功耗模式，能够降低更多的功耗。

如果正在进行闪存编程，直到对内存访问完成，系统才进入停止模式。

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

- 独立看门狗(IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见 17.3 节。
- 实时时钟(RTC)：通过备份域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSIRC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE)：通过备份域控制寄存器(RCC_BDCR)的 LSEON 位设置。

在停止模式下，如果在进入该模式前 ADC 和 DAC 没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器 ADC_CR2 的 ADON 位和寄存器 DAC_CR 的 ENx 位为 0 可关闭这 2 个外设。

退出停止模式

关于如何退出停止模式，详见下表。

当一个中断或唤醒事件导致退出停止模式时，HSIRC 振荡器被选为系统时钟。

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

表 8 停止模式

停止模式	说明
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： –设置 Cortex-M3 系统控制寄存器中的 SLEEPDEEP 位 –清除电源控制寄存器(PWR_CR)中的 PDDS 位 –通过设置 PWR_CR 中 LPDS 位选择电压调节器的模式 注：为了进入停止模式，所有的外部中断的请求位(挂起寄存器(EXTI_PR))和 RTC 的闹钟标志都必须被清除，否则停止模式的进入流程将会被跳过，程序继续运行。
退出	如果执行 WFI 进入停止模式： 设置任一外部中断线为中断模式(在 NVIC 中必须使能相应的外部中断向量)。参见中断向量表(表 51)。 如果执行 WFE 进入停止模式： 设置任一外部中断线为事件模式。参见唤醒事件管理(第 8.2.3 节)。
唤醒延时	HSIRC 唤醒时间+电压调节器从低功耗唤醒的时间。

4.3.5 待机模式

待机模式可实现系统的最低功耗。该模式是在 Cortex-M3 深睡眠模式时关闭电压调节器。整个 1.8V 供电区域被断电。PLL、HSI 和 HSE 振荡器也被断电。SRAM 和寄存器内容丢失。只有备份的寄存器和待机电路维持供电(见图 3)。

进入待机模式

关于如何进入待机模式，详见表 9。

可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗(IWDG)：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了独立看门狗，除了系统复位，它不能再被停止。详见 18.3 节。
- 实时时钟(RTC)：通过备用区域控制寄存器(RCC_BDCR)的 RTCEN 位来设置。
- 内部 RC 振荡器(LSIRC)：通过控制/状态寄存器(RCC_CSR)的 LSION 位来设置。
- 外部 32.768kHz 振荡器(LSE)：通过备用区域控制寄存器(RCC_BDCR)的 LSEON 位设置。

退出待机模式

当一个外部复位(NRST 引脚)、IWDG 复位、WKUP 引脚上的上升沿或 RTC 闹钟事件的上升沿发生时(见图 175：简化的 RTC 框图)，微控制器从待机模式退出。从待机唤醒后，除了电源控制/状态寄存器(PWR_CSR)(见第 4.4.2 节)，所有寄存器被复位。

从待机模式唤醒后的代码执行等同于复位后的执行(采样启动模式引脚、读取复位向量等)。电源控制/状态寄存器(PWR_CSR)(见第 4.4.2 节)将会指示内核由待机状态退出。

关于如何退出待机模式，详见下表。

表 9 待机模式

待机模式	说明
进入	在以下条件下执行 WFI(等待中断)或 WFE(等待事件)指令： – 设置 Cortex™-M3 系统控制寄存器中的 SLEEPDEEP 位 – 设置电源控制寄存器(PWR_CR)中的 PDDS 位 – 清除电源控制/状态寄存器(PWR_CSR)中的 WUF 位
退出	WKUP 引脚的上升沿、RTC 闹钟事件的上升沿、NRST 引脚上外部复位、IWDG 复位。
唤醒延时	复位阶段时电压调节器的启动。

待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

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

调试模式

默认情况下，如果在进行调试微处理器时，使微处理器进入停止或待机模式，将失去调试连接。这是因为 Cortex™-M3 的内核失去了时钟。

然而，通过设置 DBGMCU_CR 寄存器中的某些配置位，可以在使用低功耗模式下调试软件。更多的细节请参考第 28.16.1 节：低功耗模式的调试支持。

4.3.6 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，用于周期性从停止或待机模式下唤醒。通过对备份区域控制寄存器(RCC_BDCR)的 RTCSEL[1:0]位的编程，三个 RTC 时钟源中的二个时钟源可以选作实现此功能。

- 低功耗 32.768kHz 外部晶振(LSE)

该时钟源提供了一个低功耗且精确的时间基准。(在典型情形下消耗小于 1μA)

- 低功耗内部 RC 振荡器(LSIRC)

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。

4.4 电源控制寄存器

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

4.4.1 电源控制寄存器(PWR_CR)

地址偏移：0x00

复位值：0x0000 0000(从待机模式唤醒时清除)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位	符号	说明	
31:9	Reserved	保留	
8	DBP	**DBP**：取消后备区域的写保护 在复位后，RTC 和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 0：禁止写入 RTC 和后备寄存器 1：允许写入 RTC 和后备寄存器 注：如果 RTC 的时钟是 HSE/128，该位必须保持为'1'。	
7:5	PLS	**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 001：2.3V 010：2.4V 011：2.5V	 100：2.6V 101：2.7V 110：2.8V 111：2.9V
		注：详细说明参见数据手册中的电气特性部分。	
4	PVDE	**PVDE**：电源电压监测器(PVD)使能 0：禁止 PVD 1：开启 PVD	
3	CSBF	**CSBF**：清除待机位始终读出为 0 0：无功效	

		1：清除 SBF 待机位(写)
2	CWUF	**CWUF**：清除唤醒位始终读出为 0 0：无功效 1：2 个系统时钟周期后清除 WUF 唤醒位(写)
1	PDDS	**PDDS**：掉电深睡眠与 LPDS 位协同操作 0：当 CPU 进入深睡眠时进入停机模式，调压器的状态由 LPDS 位控制。 1：CPU 进入深睡眠时进入待机模式。
0	LPDS	**LPDS**：深睡眠下的低功耗 PDDS=0 时，与 PDDS 位协同操作 0：在停机模式下电压调压器开启 1：在停机模式下电压调压器处于低功耗模式

4.4.2 电源控制/状态寄存器(PWR_CSR)

地址偏移：0x04

复位值：0x0000 0000(从待机模式唤醒时不被清除)

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期

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

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

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

4.4.3 PWR 寄存器地址映像

以下表格列出所有 PWR 寄存器。

表 10 PWR 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	PWR_CR	保留																							DBP	PLS [2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	复位值																								0	0	0	0	0	0	0	0	0
004h	PWR_CSR	保留																							EWUP	保留					PVDD	SBF	WUF
	复位值																								0						0	0	0

关于寄存器的起始地址，参见表 1。

5 备份寄存器(BKP)

5.1 BKP 简介

备份寄存器是 42 个 16 位的寄存器，可用来存储 84 个字节的用户应用程序数据。他们处在备份域里，当 VDD 电源被切断，他们仍然由 VBAT 维持供电。当系统在待机模式下被唤醒，或系统复位或电源复位时，他们也不会被复位。

此外，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_APB1ENR 的 PWREN 和 BKPEN 位来打开电源和后备接口的时钟
- 电源控制寄存器(PWR_CR)的 DBP 位来使能对后备寄存器和 RTC 的访问。

5.2 BKP 特性

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

5.3 BKP 功能描述

5.3.1 侵入检测

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

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

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

设置 BKP_CSR 寄存器的 TPIE 位为'1'，当检测到侵入事件时就会产生一个中断。

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

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

5.3.2 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。

5.4 BKP 寄存器描述

关于在寄存器描述里面所用到的缩写，可参考第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

5.4.1 备份数据寄存器 x(BKP_DRx)(x=1...10)

地址偏移：0x04 到 0x28，0x40 到 0xBC

复位值：0x0000 0000

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

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

5.4.2 RTC 时钟校准寄存器(BKP_RTCCR)

地址偏移：0x2C

复位值：0x00000000

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

位	符号	说明
15:8	Reserved	保留，始终读为 0。启时)由侵入引脚事件复位。
9	ASOS	**ASOS**：闹钟或秒输出选择(Alarm or second output selection) 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：该位只能被后备区的复位所清除
8	ASOE	**ASOE**：允许输出闹钟或秒脉冲(Alarm or second output enable) 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：该位只能被后备区的复位所清除
7	CCO	**CCO**：校准时钟输出(Calibration clock output) 0：无影响 1：此位置 1 可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 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。

6:0	CAL	**CAL[6:0]**：校准值(Calibration value) 校准值表示在每 220 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 1000000/220ppm 的比例减慢时钟。 RTC 时钟可以被减慢 0~121ppm。

5.4.3 备份控制寄存器(BKP_CR)

偏移地址：0x30

复位值：0x0000 0000

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

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

注：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。

5.4.4 备份控制/状态寄存器(BKP_CSR)

偏移地址：0x34

复位值：0x00000000

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

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9	TIF	**TIF**：侵入中断标志(Tamper interrupt flag) 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：产生侵入中断 注意：仅当系统复位或由待机模式唤醒后才复位该位
8	TEF	**TEF**：侵入事件标志(Tamper event flag) 当检测到侵入事件时此位由硬件置 1。通过向 CTE 位写 1 可清除此标志位 0：无侵入事件 1：检测到侵入事件 注：侵入事件会复位所有的 BKP_DRx 寄存器。只要 TEF 为 1，所有的 BKP_DRx 寄存器就一直保持复位状态。当此位被置 1 时，若对 BKP_DRx 进行写操作，写入的值不会被保存。
7:3	Reserved	保留，始终读为 0。
2	TPIE	**TPIE**：允许侵入 TAMPER 引脚中断(TAMPER pin interrupt enable) 0：禁止侵入检测中断 1：允许侵入检测中断(BKP_CR 寄存器的 TPE 位也必须被置 1) 注 1：侵入中断无法将系统内核从低功耗模式唤醒。 注 2：仅当系统复位或由待机模式唤醒后才复位该位。

1	CTI	**CTI**：清除侵入检测中断(Clear tamper interrupt) 此位只能写入，读出值为 0。 0：无效 1：清除侵入检测中断和 TIF 侵入检测中断标志
0	CTE	**CTE**：清除侵入检测事件(Clear tamper event) 此位只能写入，读出值为 0。 0：无效 1：清除 TEF 侵入检测事件标志(并复位侵入检测器)。

5.4.5 BKP 寄存器映像

BKP 寄存器是 16 位的可寻址寄存器。

表 11 BKP 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	保留																																
004h	BKP_DR1	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	BKP_DR2	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	BKP_DR3	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	BKP_DR4	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	BKP_DR5	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	BKP_DR6	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	BKP_DR7	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	BKP_DR8	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	BKP_DR9	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	BKP_DR10	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	BKP_RTCCR	保留																						ASOS	ASOE	OCO	CAL[6:0]						
	复位值																							0	0	0	0	0	0	0	0	0	0
030h	RTC_CR	保留																														TPAL	TPE
	复位值																															0	0
034h	RTC_CSR	保留																												TIF	TFE	TPIE	CTI	CTE
	复位值																													0	0	0	0	0
038h	保留																																
03Ch	保留																																

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
040h	BKP_DR11	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	BKP_DR12	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	BKP_DR13	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	BKP_DR14	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	BKP_DR15	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	BKP_DR16	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
058h	BKP_DR17	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	BKP_DR18	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	BKP_DR19	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	BKP_DR20	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	BKP_DR21	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06Ch	BKP_DR22	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	BKP_DR23	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	BKP_DR24	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	BKP_DR25	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	BKP_DR26	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080h	BKP_DR27	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	BKP_DR28	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	BKP_DR29	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	BKP_DR30	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	BKP_DR31	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
094h	BKP_DR32	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
098h	BKP_DR33	保留																D[15:0]															

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09Ch	BKP_DR34	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A0h	BKP_DR35	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A4h	BKP_DR36	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A8h	BKP_DR37	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0ACh	BKP_DR38	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B0h	BKP_DR39	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B4h	BKP_DR40	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B8h	BKP_DR41	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0BCh	BKP_DR42	保留																D[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，参见表 1。

6 复位和时钟控制(RCC)

6.1 复位

W55MH32 支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

6.1.1 系统复位

除了时钟控制器的 RCC_CSR 寄存器中的复位标志位和备份区域中的寄存器(见图 3)以外，系统复位将复位所有寄存器至它们的复位状态。

当发生以下任一事件时，产生一个系统复位：

1. NRST 引脚上的低电平(外部复位)
2. 窗口看门狗计数终止(WWDG 复位)
3. 独立看门狗计数终止(IWDG 复位)
4. 软件复位(SW 复位)
5. 低功耗管理复位

可通过查看 RCC_CSR 控制状态寄存器中的复位状态标志位识别复位事件来源。

软件复位

通过将 Cortex™-M3 中断应用和复位控制寄存器中的 SYSRESETREQ 位置'1'，可实现软件复位。请参考 Cortex™-M3 技术参考手册获得进一步信息。

低功耗管理复位

在以下两种情况下可产生低功耗管理复位：

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

6.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

1. 上电/掉电复位(POR/PDR 复位)
2. 从待机模式中返回

电源复位将复位除了备份区域外的所有寄存器。(见图 3)

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

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

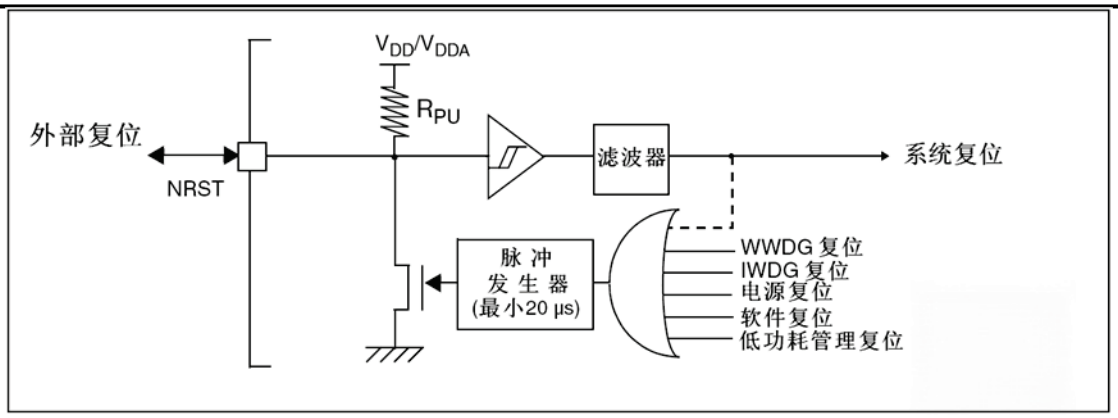


图 6 复位电路

6.1.3 备份域复位

备份区域拥有两个专门的复位，它们只影响备份区域(见图 3)。当以下事件中之一发生时，产生备份区域复位。

1. 软件复位，备份区域复位可由设置备份域控制寄存器(RCC_BDCR)(见 6.3.9 节)中的 BDRST 位产生。
2. 在 VDD 和 VBAT 两者掉电的前提下，VDD 或 VBAT 上电将引发备份区域复位。

6.2 时钟

三种不同的时钟源可被用来驱动系统时钟(SYSCLK)：

- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

这些设备有以下 2 种二级时钟源：

- 40kHz 低速内部 RC，可以用于驱动独立看门狗和通过程序选择驱动 RTC。RTC 用于从停机/待机模式下自动唤醒系统。
- 32.768kHz 低速外部晶体也可用来通过程序选择驱动 RTC(RTCCLK)。

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

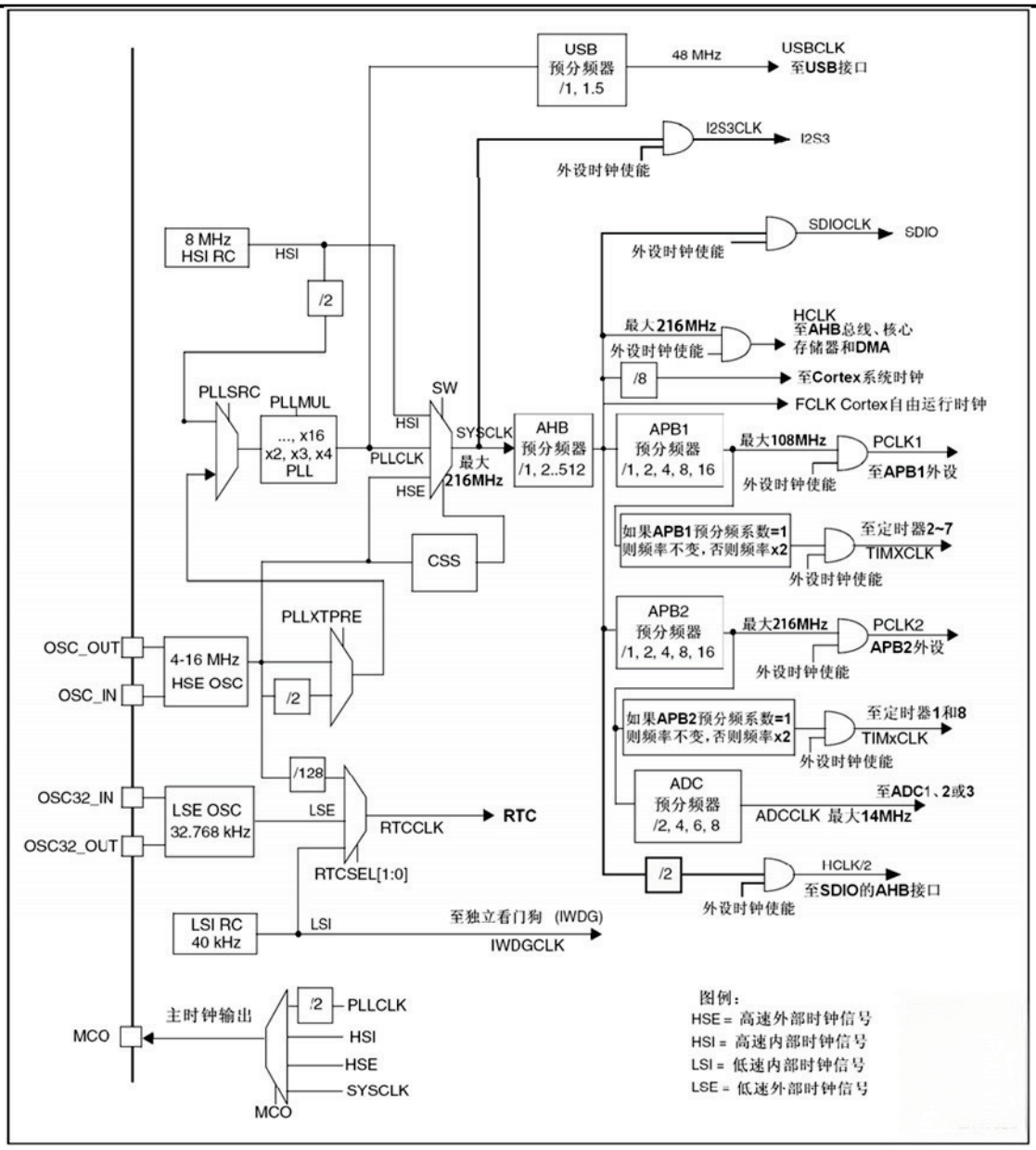


图 7 时钟树

1. 当 HSI 被用于作为 PLL 时钟的输入时，系统时钟能得到的最大频率是 108MHz。
2. 对于内部和外部时钟源的特性，请参考相应产品数据手册中“电气特性”章节。

用户可通过多个预分频器配置 AHB、高速 APB(APB2)和低速 APB(APB1)域的频率。AHB 和 APB2 域的最大频率是 216MHz。APB1 域的最大允许频率是 108MHz。SDIO 接口的时钟频率固定为 HCLK/2。RCC 通过 AHB 时钟(HCLK)8 分频后作为 Cortex 系统定时器(SysTick)的外部时钟。通过对 SysTick 控制与状态寄存器的设置，可选择上述时钟或 Cortex(HCLK)时钟作为 SysTick 时钟。ADC 时钟由高速 APB2 时钟经 2、4、6 或 8 分频后获得。

定时器时钟频率分配由硬件按以下 2 种情况自动设置：

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

FCLK 是 Cortex™-M3 的自由运行时钟。详情见 ARM 的 Cortex™-M3 技术参考手册。

6.2.1 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。

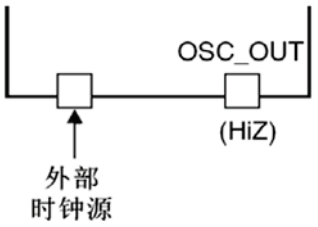
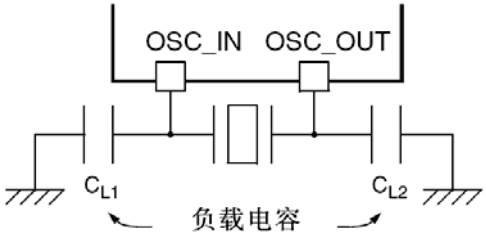
	硬件配置
外部时钟	
晶体/陶瓷谐振器	

图 8 HSE/LSE 时钟源

外部时钟源(HSE 旁路)

在这个模式里，必须提供外部时钟。它的频率最高可达 25MHz。用户可通过设置在时钟控制寄存器中的 HSEBYP 和 HSEON 位来选择这一模式。外部时钟信号(50%占空比的方波、正弦波或三角波)必须连到 SOC_IN 引脚，同时保证 OSC_OUT 引脚悬空。见图 8。

外部晶体/陶瓷谐振器(HSE 晶体)

4~16Mz 外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考图 8，进一步信息可参考数据手册的电气特性部分。

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

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

6.2.2 HSI 时钟

HSI 时钟信号由内部 8MHz 的 RC 振荡器产生，可直接作为系统时钟或在 2 分频后作为 PLL 输入。HSIRC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡器短。然而，即使在校准之后它的时钟频率精度仍较差。

校准

制造工艺决定了不同芯片的 RC 振荡器频率会不同，这就是为什么每个芯片的 HSI 时钟频率在出厂前已经被 ST 校准到 1%(25°C)的原因。系统复位时，工厂校准值被装载到时钟控制寄存器的 HSICAL[7:0]位。

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。可以通过时钟控制寄存器里的 HSITRIM[4:0]位来调整 HSI 频率。

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

如果 HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源。参考 6.2.7 节时钟安全系统。

6.2.3 PLL

内部 PLL 可以用来倍频 HSIRC 的输出时钟或 HSE 晶体输出时钟。参考图 7 和时钟控制寄存器。

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

如果 PLL 中断在时钟中断寄存器里被允许，当 PLL 准备就绪时，可产生中断申请。

如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48 或 72MHZ 时钟，用于提供 48MHz 的 USBCLK 时钟。

6.2.4 LSE 时钟

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

LSE 晶体通过在备份域控制寄存器(RCC_BDCR)里的 LSEON 位启动和关闭。

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

外部时钟源(LSE 旁路)

在这个模式里必须提供一个 32.768kHz 频率的外部时钟源。你可以通过设置在备份域控制寄存器(RCC_BDCR)里的 LSEBYP 和 LSEON 位来选择这个模式。具有 50%占空比的外部时钟信号(方波、正弦波或三角波)必须连到 OSC32_IN 引脚，同时保证 OSC32_OUT 引脚悬空，见图 8。

6.2.5 LSI 时钟

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

LSIRC 可以通过控制/状态寄存器(RCC_CSR)里的 LSION 位来启动或关闭。

在控制/状态寄存器(RCC_CSR)里的 LSIRDY 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为'1'后，此时钟才被释放。如果在时钟中断寄存器(RCC_CIR)里被允许，将产生 LSI 中断申请。

LSI 校准

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

校准可以通过使用 TIM5 的输入时钟(TIM5_CLK)测量 LSI 时钟频率实现。测量以 HSE 的精度为保证，软件可以通过调整 RTC 的 20 位预分频器来获得精确的 RTC 时钟基数，以及通过计算得到精确的独立看门狗(IWDG)的超时时间。

LSI 校准步骤如下：

1.打开 TIM5，设置通道 4 为输入捕获模式；
2.设置 AFIO_MAPR 的 TIM5_CH4_IREMAP 位为'1'，在内部把 LSI 连接到 TIM5 的通道 4；
3.通过 TIM5 的捕获/比较 4 事件或者中断来测量 LSI 时钟频率；
4.根据测量结果和期望的 RTC 时间基数和独立看门狗的超时时间，设置 20 位预分频器。

6.2.6 系统时钟(SYSCLK)选择

系统复位后，HSI 振荡器被选为系统时钟。当时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。
只有当目标时钟源准备就绪了(经过启动稳定阶段的延迟或 PLL 稳定)，从一个时钟源到另一个时钟源的切换才会发生。在被选择时钟源没有就绪时，系统时钟的切换不会发生。直至目标时钟源就绪，才发生切换。
在时钟控制寄存器(RCC_CR)里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

6.2.7 时钟安全系统(CSS)

时钟安全系统可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。
如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟失效事件将被送到高级定时器(TIM1 和 TIM8)的刹车输入端，并产生时钟安全中断 CSSI，允许软件完成营救操作。此 CSSI 中断连接到 Cortex™ -M3 的 NMI 中断(不可屏蔽中断)。

注：一旦 CSS 被激活，并且 HSE 时钟出现故障，CSS 中断就产生，并且 NMI 也自动产生。NMI 将被不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器(RCC_CIR)里的 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 HSI 振荡器，同时外部 HSE 振荡器被关闭。在时钟失效时，如果 HSE 振荡器时钟(被分频或未被分频)是用作系统时钟的 PLL 的输入时钟，PLL 也将被关闭。

6.2.8 RTC 时钟

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

- 如果 LSE 被选为 RTC 时钟：
  - 只要 VBAT 维持供电，尽管 VDD 供电被切断，RTC 仍继续工作。
- 如果 LSI 被选为自动唤醒单元(AWU)时钟：
  - 如果 VDD 供电被切断，AWU 状态不能被保证。有关 LSI 校准，详见 6.2.5 节 LSI 时钟。
- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 VDD 供电被切断或内部电压调压器被关闭(1.8V 域的供电被切断)，则 RTC 状态不确定。
  - 必须设置电源控制寄存器(见 4.4.1 节)的 DPB 位(取消后备区域的写保护)为'1'。

6.2.9 看门狗时钟

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

6.2.10 时钟输出

微控制器允许输出时钟信号到外部 MCO 引脚。

相应的 GPIO 端口寄存器必须被配置为相应功能。以下四个时钟信号可被选作 MCO 时钟：

- SYSCLK
- HSI
- HSE
- 除 2 的 PLL 时钟

时钟的选择由时钟配置寄存器(RCC_CFGR)中的 MCO[2:0]位控制。

6.3 RCC 寄存器描述

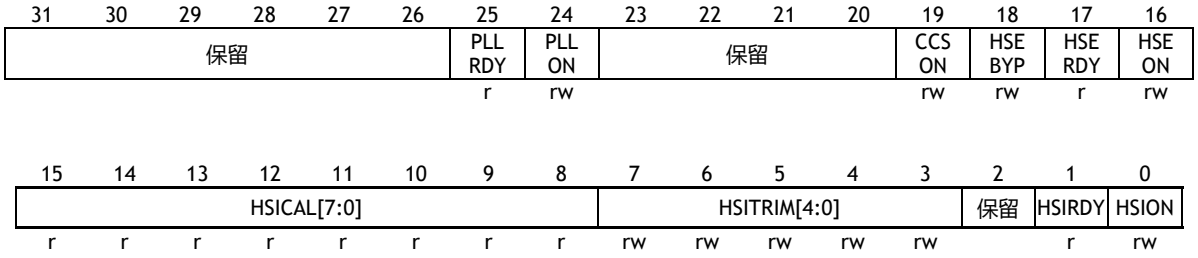
请参考第 1 节中有关寄存器描述中用到的缩写。

6.3.1 时钟控制寄存器(RCC_CR)

偏移地址:0x00

复位值:0x0000 XX83，X 代表未定义

访问:无等待状态,字,半字和字节访问



位	符号	说明
31:26	Reserved	保留，始终读为 0。
25	PLLRDY	**PLLRDY**：PLL 时钟就绪标志(PLL clock ready flag) PLL 锁定后由硬件置'1'。 0：PLL 未锁定; 1：PLL 锁定。
24	PLLON	**PLLON**：PLL 使能(PLL enable) 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零。当 PLL 时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0：PLL 关闭; 1：PLL 使能。
23:20	Reserved	保留，始终读为 0。
19	CSSON	**CSSON**：时钟安全系统使能(Clock security system enable) 由软件置'1'或清零以使能时钟监测器。 0：时钟监测器关闭; 1：如果外部 4-16MHz 振荡器就绪，时钟监测器开启。
18	HSEBYP	**HSEBYP**：外部高速时钟旁路(External high-speed clock bypass)

		在调试模式下由软件置'1'或清零来旁路外部晶体振荡器。只有在外部 4-16MHz 振荡器关闭的情况下，才能写入该位。 0：外部 4-16MHz 振荡器没有旁路； 1：外部 4-16MHz 外部晶体振荡器被旁路。
17	**HSERDY**	**HSERDY**：外部高速时钟就绪标志(External high-speed clock ready flag) 由硬件置'1'来指示外部 4-16MHz 振荡器已经稳定。在 HSEON 位清零后，该位需要 6 个外部 4-25MHz 振荡器周期清零。 0：外部 4-16MHz 振荡器没有就绪； 1：外部 4-16MHz 振荡器就绪。
16	**HSEON**	**HSEON**：外部高速时钟使能(External high-speed clock enable) 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零，关闭 4-16MHz 外部振荡器。当外部 4-16MHz 振荡器被用作或被选择将要作为系统时钟时，该位不能被清零。 0：HSE 振荡器关闭； 1：HSE 振荡器开启。
15:8	**HSICAL[7:0]**	**HSICAL[7:0]**：内部高速时钟校准(Internal high-speed clock calibration) 在系统启动时，这些位被自动初始化
7:3	**HSITRIM [4:0]**	**HSITRIM[4:0]**：内部高速时钟调整(Internal high-speed clock trimming) 由软件写入来调整内部高速时钟，它们被叠加在 HSICAL[5:0]数值上。 这些位在 HSICAL[7:0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSIRC 振荡器的频率。 默认数值为 16，可以把 HSI 调整到 8MHz±1%；每步 HSICAL 的变化调整约 40kHz。
2	**Reserved**	保留，始终读为 0。
1	**HSIRDY**	**HSIRDY**：内部高速时钟就绪标志(Internal high-speed clock ready flag) 由硬件置'1'来指示内部 8MHz 振荡器已经稳定。在 HSION 位清零后，该位需要 6 个内部 8MHz 振荡器周期清零。 0：内部 8MHz 振荡器没有就绪； 1：内部 8MHz 振荡器就绪。
0	**HSION**	**HSION**：内部高速时钟使能(Internal high-speed clock enable) 由软件置'1'或清零。 当从待机和停止模式返回或用作系统时钟的外部 4-16MHz 振荡器发生故障时，该位由硬件置'1'来启动内部 8MHz 的 RC 振荡器。当内部 8MHz 振荡器被直接或间接地用作或被选择将要作为系统时钟时，该位不能被清零。 0：内部 8MHz 振荡器关闭； 1：内部 8MHz 振荡器开启。

6.3.2 时钟配置寄存器(RCC_CFGR)

偏移地址:0x04

复位值:0x0000 0000

访问:0 到 2 个等待周期，字，半字和字节访问

只有当访问发生在时钟切换时，才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					MCO[2:0]			保留	USBPRE	PLLMUL[3:0]				PLLXTPRE	PLLSRC
					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
ADCPRE[1:0]		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	rw	rw	r	r	rw	rw

位	符号	说明
31:27	**Reserved**	保留，始终读为 0。

26:24	MCO	**MCO**：微控制器时钟输出(Microcontroller clock output) 由软件置'1'或清零。 0xx：没有时钟输出； 100：系统时钟(SYSCLK)输出； 101：内部 RC 振荡器时钟(HSI)输出； 110：外部振荡器时钟(HSE)输出； 111：PLL 时钟 2 分频后输出。 注意：-该时钟输出在启动和切换 MCO 时钟源时可能会被截断。 -在系统时钟作为输出至 MCO 引脚时，请保证输出时钟频率不超过 50MHz(I/O 口最高频率)。
23	Reserved	保留，始终读为 0。
22	USBPRE	**USBPRE**：USB 预分频(USB prescaler) 由软件置'1'或清'0'来产生 48MHz 的 USB 时钟。在 RCC_APB1ENR 寄存器中使能 USB 时钟之前，必须保证该位已经有效。如果 USB 时钟被使能，该位不能被清零。 0：PLL 时钟 1.5 倍分频作为 USB 时钟 1：PLL 时钟直接作为 USB 时钟
21:18	PLLMUL	**PLLMUL**：PLL 倍频系数(PLL multiplication factor) 由软件设置来确定 PLL 倍频系数。只有在 PLL 关闭的情况下才可被写入。 注意：PLL 的输出频率不能超过 72MHz 0000：PLL2 倍频输出　　1000：PLL10 倍频输出 0001：PLL3 倍频输出　　1001：PLL11 倍频输出 0010：PLL4 倍频输出　　1010：PLL12 倍频输出 0011：PLL5 倍频输出　　1011：PLL13 倍频输出 0100：PLL6 倍频输出　　1100：PLL14 倍频输出 0101：PLL7 倍频输出　　1101：PLL15 倍频输出 0110：PLL8 倍频输出　　1110：PLL16 倍频输出 0111：PLL9 倍频输出　　1111：PLL16 倍频输出
17	PLLXTPRE	**PLLXTPRE**：HSE 分频器作为 PLL 输入(HSE divider for PLL entry) 由软件置'1'或清'0'来分频 HSE 后作为 PLL 输入时钟。只能在关闭 PLL 时才能写入此位。 0：HSE 不分频 1：HSE2 分频
16	PLLSRC	**PLLSRC**：PLL 输入时钟源(PLL entry clock source) 由软件置'1'或清'0'来选择 PLL 输入时钟源。只能在关闭 PLL 时才能写入此位。 0：HSI 振荡器时钟经 2 分频后作为 PLL 输入时钟 1：HSE 时钟作为 PLL 输入时钟。
15:14	ADCPRE [1:0]	**ADCPRE[1:0]**：ADC 预分频(ADC prescaler) 由软件置'1'或清'0'来确定 ADC 时钟频率 00：PCLK22 分频后作为 ADC 时钟 01：PCLK24 分频后作为 ADC 时钟 10：PCLK26 分频后作为 ADC 时钟 11：PCLK28 分频后作为 ADC 时钟
13:11	PPRE2 [2:0]	**PPRE2[2:0]**：高速 APB 预分频(APB2)(APB high-speed prescaler(APB2)) 由软件置'1'或清'0'来控制高速 APB2 时钟(PCLK2)的预分频系数。 0xx：HCLK 不分频 100：HCLK2 分频 101：HCLK4 分频 110：HCLK8 分频 111：HCLK16 分频
10:8	PPRE1 [2:0]	**PPRE1[2:0]**：低速 APB 预分频(APB1)(APB low-speed prescaler(APB1)) 由软件置'1'或清'0'来控制低速 APB1 时钟(PCLK1)的预分频系数。 警告：软件必须保证 APB1 时钟频率不超过 36MHz。 0xx：HCLK 不分频

		100：HCLK2 分频 101：HCLK4 分频 110：HCLK8 分频 111：HCLK16 分频
7:4	**HPRE[3:0]**	**HPRE[3:0]**：AHB 预分频(AHB Prescaler) 由软件置'1'或清'0'来控制 AHB 时钟的预分频系数。 0xxx：SYSCLK 不分频 1000：SYSCLK2 分频　　1100：SYSCLK64 分频 1001：SYSCLK4 分频　　1101：SYSCLK128 分频 1010：SYSCLK8 分频　　1110：SYSCLK256 分频 1011：SYSCLK16 分频　　1111：SYSCLK512 分频
		注意：当 AHB 时钟的预分频系数大于 1 时，必须开启预取缓冲器。详见闪存读取(第 2.3.3 节)。
3:2	**SWS[1:0]**	**SWS[1:0]**：系统时钟切换状态(System clock switch status) 由硬件置'1'或清'0'来指示哪一个时钟源被作为系统时钟。 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。
1:0	**SW[1:0]**	**SW[1:0]**：系统时钟切换(System clock switch) 由软件置'1'或清'0'来选择系统时钟源。 在从停止或待机模式中返回时或直接或间接作为系统时钟的 HSE 出现故障时，由硬件强制选择 HSI 作为系统时钟(如果时钟安全系统已经启动) 00：HSI 作为系统时钟； 01：HSE 作为系统时钟； 10：PLL 输出作为系统时钟； 11：不可用。

6.3.3 时钟中断寄存器(RCC_CIR)

偏移地址:0x08

复位值:0x0000 0000

访问:无等待周期,字,半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留								CSSC	保留		PLL RDYC	HSE RDYC	HIS RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	保留		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

位	符号	说明
31:24	**Reserved**	保留，始终读为 0。
23	**MCO**	**CSSC**：清除时钟安全系统中断(Clock security system interrupt clear) 由软件置'1'来清除 CSSF 安全系统中断标志位 CSSF。 0：无作用； 1：清除 CSSF 安全系统中断标志位。
22:21	**Reserved**	保留，始终读为 0。
20	**PLLRDYC**	**PLLRDYC**：清除 PLL 就绪中断(PLL ready interrupt clear) 由软件置'1'来清除 PLL 就绪中断标志位 PLLRDYF。 0：无作用； 1：清除 PLL 就绪中断标志位 PLLRDYF。

19	HSERDYC	**HSERDYC**：清除 HSE 就绪中断(HSE ready interrupt clear) 由软件置'1'来清除 HSE 就绪中断标志位 HSERDYF。 0：无作用； 1：清除 HSE 就绪中断标志位 HSERDYF。
18	HSIRDYC	**HSIRDYC**：清除 HSI 就绪中断(HSI ready interrupt clear) 由软件置'1'来清除 HSI 就绪中断标志位 HSIRDYF。 0：无作用； 1：清除 HSI 就绪中断标志位 HSIRDYF。
17	LSERDYC	**LSERDYC**：清除 LSE 就绪中断(LSEreadyinterruptclear) 由软件置'1'来清除 LSE 就绪中断标志位 LSERDYF。 0：无作用； 1：清除 LSE 就绪中断标志位 LSERDYF。
16	LSIRDYC	**LSIRDYC**：清除 LSI 就绪中断(LSI ready interrupt clear) 由软件置'1'来清除 LSI 就绪中断标志位 LSIRDYF。 0：无作用； 1：清除 LSI 就绪中断标志位 LSIRDYF。
15:13	Reserved	保留，始终读为 0。
12	PLLRDYIE	**PLLRDYIE**：PLL 就绪中断使能(PLL ready interrupt enable) 由软件置'1'或清'0'来使能或关闭 PLL 就绪中断。 0：PLL 就绪中断关闭； 1：PLL 就绪中断使能。
11	HSERDYIE	**HSERDYIE**：HSE 就绪中断使能(HSE ready interrupt enable) 由软件置'1'或清'0'来使能或关闭外部 4-16MHz 振荡器就绪中断。 0：HSE 就绪中断关闭； 1：HSE 就绪中断使能。
10	HSIRDYIE	**HSIRDYIE**：HSI 就绪中断使能(HSI ready interrupt enable) 由软件置'1'或清'0'来使能或关闭内部 8MHzRC 振荡器就绪中断。 0：HSI 就绪中断关闭； 1：HSI 就绪中断使能。
9	LSERDYIE	**LSERDYIE**：LSE 就绪中断使能(LSE ready interrupt enable) 由软件置'1'或清'0'来使能或关闭外部 32kHzRC 振荡器就绪中断。 0：LSE 就绪中断关闭； 1：LSE 就绪中断使能。
8	LSIRDYIE	**LSIRDYIE**：LSI 就绪中断使能(LSI ready interrupt enable) 由软件置'1'或清'0'来使能或关闭内部 40kHzRC 振荡器就绪中断。 0：LSI 就绪中断关闭； 1：LSI 就绪中断使能。
7	CSSF	**CSSF**：时钟安全系统中断标志(Clock security system interrupt flag) 在外部 4-16MHz 振荡器时钟出现故障时，由硬件置'1'。由软件通过置'1'CSSC 位来清除。 0：无 HSE 时钟失效产生的安全系统中断； 1：HSE 时钟失效导致了时钟安全系统中断。
6:5	Reserved	保留，始终读为 0。
4	PLLRDYF	**PLLRDYF**：PLL 就绪中断标志(PLL ready interrupt flag) 在 PLL 就绪且 PLLRDYIE 位被置'1'时，由硬件置'1'。由软件通过置'1'PLLRDYC 位来清除。 0：无 PLL 上锁产生的时钟就绪中断； 1：PLL 上锁导致时钟就绪中断。
3	HSERDYF	**HSERDYF**：HSE 就绪中断标志(HSE ready interrupt flag) 在外部低速时钟就绪且 HSERDYIE 位被置'1'时，由硬件置'1'。由软件通过置'1'HSERDYC 位来清除。 0：无外部 4-16MHz 振荡器产生的时钟就绪中断； 1：外部 4-16MHz 振荡器导致时钟就绪中断。

2	HSIRDYF	**HSIRDYF**：HSI 就绪中断标志(HSI ready interrupt flag) 在内部高速时钟就绪且 HSIRDYIE 位被置'1'时，由硬件置'1'。由软件通过置'1'HSIRDYC 位来清除。 0：无内部 8MHzRC 振荡器产生的时钟就绪中断； 1：内部 8MHzRC 振荡器导致时钟就绪中断。
1	LSERDYF	**LSERDYF**：LSE 就绪中断标志(LSE ready interrupt flag) 在外部低速时钟就绪且 LSERDYIE 位被置'1'时，由硬件置'1'。由软件通过置'1'LSERDYC 位来清除。 0：无外部 32kHz 振荡器产生的时钟就绪中断； 1：外部 32kHz 振荡器导致时钟就绪中断。
0	LSIRDYF	**LSIRDYF**：LSI 就绪中断标志(LSI ready interrupt flag) 在内部低速时钟就绪且 LSIRDYIE 位被置'1'时，由硬件置'1'。由软件通过置'1'LSIRDYC 位来清除。 0：无内部 40kHzRC 振荡器产生的时钟就绪中断； 1：内部 40kHzRC 振荡器导致时钟就绪中断。

6.3.4 APB2 外设复位寄存器(RCC_APB2RSTR)

偏移地址:0x0C

复位值:0x0000 0000

访问：无等待周期,字,半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										TIM11 RST	TIM10 RST	TIM9 RST	保留		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	IOPG RST	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	保留	AFIO RST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw

位	符号	说明
31:22	Reserved	保留，始终读为 0。
21	TIM11RST	**TIM8RST**：TIM11 定时器复位(TIM11 timer reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM11 定时器。
20	TIM10RST	**TIM10RST**：TIM10 定时器复位(TIM10 timer reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM10 定时器。
19	TIM9RST	**TIM9RST**：TIM9 定时器复位(TIM8 timer reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM9 定时器。
18:16	Reserved	保留，始终读为 0。
15	ADC3RST	**ADC3RST**：ADC3 接口复位(ADC3 interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 ADC3 接口。
14	USART1RST	**USART1RST**：USART1 复位(USART1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USART1。

13	TIM8RST	**TIM8RST**：TIM8 定时器复位(TIM8 timer reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM8 定时器。
12	SPI1RST	**SPI1RST**：SPI1 复位(SPI1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 SPI1。
11	TIM1RST	**TIM1RST**：TIM1 定时器复位(TIM1 timer reset)由软件置'1'或清'0' 0：无作用； 1：复位 TIM1 定时器。
10	ADC2RST	**ADC2RST**：ADC2 接口复位(ADC2 interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 ADC2 接口。
9	ADC1RST	**ADC1RST**：ADC1 接口复位(ADC1 interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 ADC1 接口。
8	IOPGRST	**IOPGRST**：IO 端口 G 复位(IO portG reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 G。
7	IOPFRST	**IOPFRST**：IO 端口 F 复位(IO portF reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 F。
6	IOPERST	**IOPERST**：IO 端口 E 复位(IO portE reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 E。
5	IOPDRST	**IOPDRST**：IO 端口 D 复位(IO portD reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 D。
4	IOPCRST	**IOPCRST**：IO 端口 C 复位(IO portC reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 C。
3	IOPBRST	**IOPBRST**：IO 端口 B 复位(IO portB reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 B。
2	IOPARST	**IOPARST**：IO 端口 A 复位(IO portA reset) 由软件置'1'或清'0' 0：无作用； 1：复位 IO 端口 A。
1	Reserved	保留，始终读为 0。
0	AFIORST	**AFIORST**：辅助功能 IO 复位(Alternate function I/O reset) 由软件置'1'或清'0' 0：无作用；

		1：复位辅助功能。

6.3.5 APB1 外设复位寄存器(RCC_APB1RSTR)

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期，字，半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	保留			WWDG RST	保留		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw				rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:30	**Reserved**	保留，始终读为 0。
29	**DACRST**	**DACRST**：DAC 接口复位(DAC interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位 DAC 接口。
28	**PWRRST**	**PWRRST**：电源接口复位(Power interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位电源接口。
27	**BKPRST**	**BKPRST**：备份接口复位(Backup interface reset) 由软件置'1'或清'0' 0：无作用； 1：复位备份接口。
26	**Reserved**	保留，始终读为 0。
25	**CANRST**	**CANRST**：CAN 复位(CAN reset) 由软件置'1'或清'0' 0：无作用； 1：复位 CAN。
24	**Reserved**	保留，始终读为 0。
23	**USBRST**	**USBRST**：USB 复位(USB reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USB。
22	**I2C2RST**	**I2C2RST**：I2C2 复位(I2C2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 I2C2。
21	**I2C1RST**	**I2C1RST**：I2C1 复位(I2C1 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 I2C1。
20	**UART5RST**	**UART5RST**：UART5 复位(UART5 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 UART5。

19	UART4RST	**UART4RST**：UART4 复位(UART4 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 UART4。
18	USART3RST	**USART3RST**：USART3 复位(USART3 reset)由软件置'1'或清'0' 0：无作用； 1：复位 USART3。
17	USART2RST	**USART2RST**：USART2 复位(USART2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 USART2。
16	Reserved	保留，始终读为 0。
15	SPI3RST	**SPI3RSTSPI3** 复位(SPI3 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 SPI3。
14:12	Reserved	保留，始终读为 0。
11	WWDGRST	**WWDGRST**：窗口看门狗复位(Window watchdog reset) 由软件置'1'或清'0' 0：无作用； 1：复位窗口看门狗。
10:9	Reserved	保留，始终读为 0。
8	TIM14RST	**TIM14RST**：定时器 14 复位(Timer14 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM14 定时器。
7	TIM13RST	**TIM13RST**：定时器 13 复位(Timer13 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM13 定时器。
6	TIM12RST	**TIM12RST**：定时器 12 复位(Timer12 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM12 定时器。
5	TIM7RST	**TIM7RST**：定时器 7 复位(Timer7 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM7 定时器。
4	TIM6RST	**TIM6RST**：定时器 6 复位(Timer6 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM6 定时器。
3	TIM5RST	**TIM5RST**：定时器 5 复位(Timer5 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM5 定时器。
2	TIM4RST	**TIM4RST**：定时器 4 复位(Timer4 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM4 定时器。
1	TIM3RST	**TIM3RST**：定时器 3 复位(Timer3 reset)

		由软件置'1'或清'0' 0：无作用； 1：复位 TIM3 定时器。
0	TIM2RST	**TIM2RST**：定时器 2 复位(Timer2 reset) 由软件置'1'或清'0' 0：无作用； 1：复位 TIM2 定时器。

6.3.6 AHB 外设时钟使能寄存器(RCC_AHBENR)

偏移地址：0x14

复位值：0x0000 0014

访问：无等待周期,字,半字和字节访问

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					SDIO EN	保留			CRC EN	保留	FLITF EN	保留	SRAM EN	DMA2 EN	DMA1 EN
					rw				rw		rw		rw	rw	rw

位	符号	说明
31:11	Reserved	保留，始终读为 0。
10	SDIOEN	**SDIOEN**：SDIO 时钟使能(SDIO clock enable) 由软件置'1'或清'0'。 0：SDIO 时钟关闭； 1：SDIO 时钟开启。
9:7	Reserved	保留，始终读为 0。
6	CRCEN	**CRCEN**：CRC 时钟使能(CRC clock enable) 由软件置'1'或清'0'。 0：CRC 时钟关闭； 1：CRC 时钟开启。
5	Reserved	保留，始终读为 0。
4	FLITFEN	**FLITFEN**：闪存接口电路时钟使能(FLITF clock enable) 由软件置'1'或清'0'来开启或关闭睡眠模式时闪存接口电路时钟。 0：睡眠模式时闪存接口电路时钟关闭； 1：睡眠模式时闪存接口电路时钟开启。
3	Reserved	保留，始终读为 0。
2	SRAMEN	**SRAMEN**：SRAM 时钟使能(SRAM interface clock enable) 由软件置'1'或清'0'来开启或关闭睡眠模式时 SRAM 时钟。 0：睡眠模式时 SRAM 时钟关闭； 1：睡眠模式时 SRAM 时钟开启。
1	DMA2EN	**DMA2EN**：DMA2 时钟使能(DMA2 clock enable) 由软件置'1'或清'0'。 0：DMA2 时钟关闭； 1：DMA2 时钟开启。
0	DMA1EN	**DMA1EN**：DMA1 时钟使能(DMA1 clock enable) 由软件置'1'或清'0'。 0：DMA1 时钟关闭； 1：DMA1 时钟开启。

6.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR)

偏移地址：0x18

复位值：0x0000 0000

访问：字，半字和字节访问

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										TIM11EN	TIM10EN	TIM9EN	保留		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	IOPGEN	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	保留	AFIOEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位	符号	说明
31:22	Reserved	保留，始终读为 0。
21	TIM11EN	**TIM11EN**：TIM11 定时器时钟使能(TIM11 Timer clock enable) 由软件置'1'或清'0' 0：TIM11 定时器时钟关闭； 1：TIM11 定时器时钟开启。
20	TIM10EN	**TIM10EN**：TIM10 定时器时钟使能(TIM10 Timer clock enable) 由软件置'1'或清'0' 0：TIM10 定时器时钟关闭； 1：TIM10 定时器时钟开启。
19	TIM9EN	**TIM9EN**：TIM9 定时器时钟使能(TIM9 Timer clock enable) 由软件置'1'或清'0' 0：TIM9 定时器时钟关闭； 1：TIM9 定时器时钟开启。
18:16	Reserved	保留，始终读为 0。
15	ADC3EN	**ADC3EN**：ADC3 接口时钟使能(ADC3 interface clock enable) 由软件置'1'或清'0' 0：ADC3 接口时钟关闭； 1：ADC3 接口时钟开启。
14	USART1EN	**USART1EN**：USART1 时钟使能(USART1 clock enable) 由软件置'1'或清'0' 0：USART1 时钟关闭； 1：USART1 时钟开启。
13	TIM8EN	**TIM8EN**：TIM8 定时器时钟使能(TIM8 Timer clock enable) 由软件置'1'或清'0' 0：TIM8 定时器时钟关闭； 1：TIM8 定时器时钟开启。
12	SPI1EN	**SPI1EN**：SPI1 时钟使能(SPI1 clock enable) 由软件置'1'或清'0' 0：SPI1 时钟关闭； 1：SPI1 时钟开启。
11	TIM1EN	**TIM1EN**：TIM1 定时器时钟使能(TIM1 Timer clock enable) 由软件置'1'或清'0' 0：TIM1 定时器时钟关闭；

		1：TIM1 定时器时钟开启。
10	ADC2EN	**ADC2EN**：ADC2 接口时钟使能(ADC2 interface clock enable) 由软件置'1'或清'0' 0：ADC2 接口时钟关闭； 1：ADC2 接口时钟开启。
9	ADC1EN	**ADC1EN**：ADC1 接口时钟使能(ADC1 interface clock enable) 由软件置'1'或清'0' 0：ADC1 接口时钟关闭； 1：ADC1 接口时钟开启。
8	IOPGEN	**IOPGEN**：IO 端口 G 时钟使能(I/O portG clock enable) 由软件置'1'或清'0' 0：IO 端口 G 时钟关闭； 1：IO 端口 G 时钟开启。
7	IOPFEN	**IOPFEN**：IO 端口 F 时钟使能(I/O portF clock enable) 由软件置'1'或清'0' 0：IO 端口 F 时钟关闭； 1：IO 端口 F 时钟开启。
6	IOPEEN	**IOPEEN**：IO 端口 E 时钟使能(I/O portE clock enable) 由软件置'1'或清'0' 0：IO 端口 E 时钟关闭； 1：IO 端口 E 时钟开启。
5	IOPDEN	**IOPDEN**：IO 端口 D 时钟使能(I/O portD clock enable) 由软件置'1'或清'0' 0：IO 端口 D 时钟关闭； 1：IO 端口 D 时钟开启。
4	IOPCEN	**IOPCEN**：IO 端口 C 时钟使能(I/O portC clock enable) 由软件置'1'或清'0' 0：IO 端口 C 时钟关闭； 1：IO 端口 C 时钟开启。
3	IOPBEN	**IOPBEN**：IO 端口 B 时钟使能(I/O portB clock enable) 由软件置'1'或清'0' 0：IO 端口 B 时钟关闭； 1：IO 端口 B 时钟开启。
2	IOPAEN	**IOPAEN**：IO 端口 A 时钟使能(I/O portA clock enable) 由软件置'1'或清'0' 0：IO 端口 A 时钟关闭； 1：IO 端口 A 时钟开启。
1	Reserved	保留，始终读为 0。
0	AFIOEN	**AFIOEN**：辅助功能 IO 时钟使能(Alternate function I/O clock enable) 由软件置'1'或清'0' 0：辅助功能 IO 时钟关闭； 1：辅助功能 IO 时钟开启。

6.3.8 APB1 外设时钟使能寄存器(RCC_APB1ENR)

偏移地址：0x1C

复位值：0x0000 0000

访问：字、半字和字节访问

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留		DACEN	PWREN	BKPEN	保留	CANEN	保留	USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	保留
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	保留			WWDGEN	保留		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw				rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:30	Reserved	保留，始终读为 0。
29	DACEN	**DACEN:**DAC 接口时钟使能(DAC interface clock enable)由软件置'1'或清'0' 0：DAC 接口时钟关闭; 1：DAC 接口时钟开启。
28	PWREN	**PWREN**：电源接口时钟使能(Power interface clock enable)由软件置'1'或清'0' 0：电源接口时钟关闭; 1：电源接口时钟开启。
27	BKPEN	**BKPEN**：备份接口时钟使能(Backup interface clock enable)由软件置'1'或清'0' 0：备份接口时钟关闭; 1：备份接口时钟开启。
26	Reserved	保留，始终读为 0。
25	CANEN	**CANEN**：CAN 时钟使能(CAN clock enable)由软件置'1'或清'0' 0：CAN 时钟关闭; 1：CAN 时钟开启。
24	Reserved	保留，始终读为 0。
23	USBEN	**USBEN**：USB 时钟使能(USB clock enable)由软件置'1'或清'0' 0：USB 时钟关闭; 1：USB 时钟开启。
22	I2C2EN	**I2C2EN**：I2C2 时钟使能(I2C2 clock enable)由软件置'1'或清'0' 0：I2C2 时钟关闭; 1：I2C2 时钟开启。
21	I2C1EN	**I2C1EN**：I2C1 时钟使能(I2C1 clock enable)由软件置'1'或清'0' 0：I2C1 时钟关闭; 1：I2C1 时钟开启。
20	UART5EN	**UART5EN**：UART5 时钟使能(UART5 clock enable)由软件置'1'或清'0' 0：UART5 时钟关闭; 1：UART5 时钟开启。
19	UART4EN	**UART4EN**：UART4 时钟使能(UART4 clock enable)由软件置'1'或清'0' 0：UART4 时钟关闭; 1：UART4 时钟开启。
18	USART3EN	**USART3EN**：USART3 时钟使能(USART3 clock enable)由软件置'1'或清'0' 0：USART3 时钟关闭; 1：USART3 时钟开启。
17	USART2EN	**USART2EN**：USART2 时钟使能(USART2 clock enable)由软件置'1'或清'0' 0：USART2 时钟关闭; 1：USART2 时钟开启。
16	Reserved	保留，始终读为 0。
15	SPI3EN	**SPI3EN**：SPI3 时钟使能(SPI3 clock enable)由软件置'1'或清'0' 0：SPI3 时钟关闭; 1：SPI3 时钟开启。
14	Reversed	保留，始终读为 0。

13:12	Reserved	保留，始终读为 0。
11	WWDGEN	**WWDGEN**：窗口看门狗时钟使能(Window watchdog clock enable)由软件置'1'或清'0' 0：窗口看门狗时钟关闭； 1：窗口看门狗时钟开启。
10:6	Reserved	保留，始终读为 0。
5	TIM7EN	**TIM7EN**：定时器 7 时钟使能(Timer7 clock enable)由软件置'1'或清'0' 0：定时器 7 时钟关闭； 1：定时器 7 时钟开启。
4	TIM6EN	**TIM6EN**：定时器 6 时钟使能(Timer6 clock enable)由软件置'1'或清'0' 0：定时器 6 时钟关闭； 1：定时器 6 时钟开启。
3	TIM5EN	**TIM5EN**：定时器 5 时钟使能(Timer5 clock enable)由软件置'1'或清'0' 0：定时器 5 时钟关闭； 1：定时器 5 时钟开启。
2	TIM4EN	**TIM4EN**：定时器 4 时钟使能(Timer4 clock enable)由软件置'1'或清'0' 0：定时器 4 时钟关闭； 1：定时器 4 时钟开启。
1	TIM3EN	**TIM3EN**：定时器 3 时钟使能(Timer3 clock enable)由软件置'1'或清'0' 0：定时器 3 时钟关闭； 1：定时器 3 时钟开启。
0	TIM2EN	**TIM2EN**：定时器 2 时钟使能(Timer2 clock enable)由软件置'1'或清'0' 0：定时器 2 时钟关闭； 1：定时器 2 时钟开启。

6.3.9 备份域控制寄存器(RCC_BDCR)

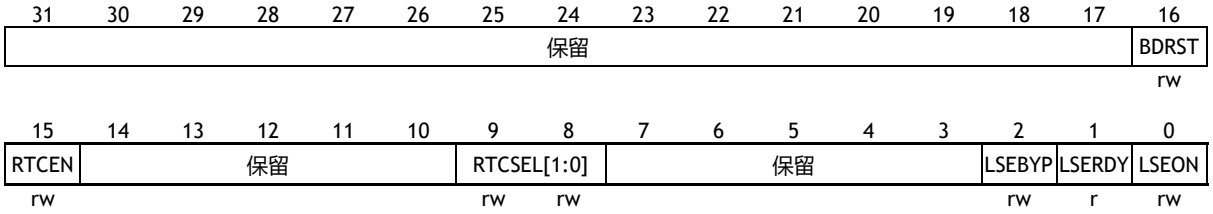
偏移地址：0x20

复位值：0x0000 0000，只能由备份域复位有效复位

访问：0 到 3 等待周期，字、半字和字节访问

当连续对该寄存器进行访问时，将插入等待状态。

注意：备份域控制寄存器中(RCC_BDCR)的 LSEON、LSEBYP、RTCSEL 和 RTCEN 位处于备份域。因此，这些位在复位后处于写保护状态，只有在电源控制寄存器(PWR_CR)中的 DBP 位置'1'后才能对这些位进行改动。进一步信息请参考 5.1 节。这些位只能由备份域复位清除(见 6.1.3 节)。任何内部或外部复位都不会影响这些位。



位	符号	说明
31:17	Reserved	保留，始终读为 0。
16	BDRST	**BDRST**：备份域软件复位(Backup domain software reset) 由软件置'1'或清'0' 0：复位未激活； 1：复位整个备份域。
15	RTCEN	**RTCEN**：RTC 时钟使能(RTC clock enable) 由软件置'1'或清'0' 0：RTC 时钟关闭；

		1：RTC 时钟开启。
14:10	Reserved	保留，始终读为 0。
9:8	RTCSEL [1:0]	**RTCSEL[1:0]**：RTC 时钟源选择(RTC clock source selection) 由软件设置来选择 RTC 时钟源。一旦 RTC 时钟源被选定，直到下次后备域被复位，它不能在被改变。可通过设置 BDRST 位来清除。 00：无时钟； 01：LSE 振荡器作为 RTC 时钟；10：LSI 振荡器作为 RTC 时钟； 11：HSE 振荡器在 128 分频后作为 RTC 时钟。
7:3	Reserved	保留，始终读为 0。
2	LSEBYP	**LSEBYP**：外部低速时钟振荡器旁路(External low-speed oscillator bypass) 在调试模式下由软件置'1'或清'0'来旁路 LSE。只有在外部 32kHz 振荡器关闭时，才能写入该位 0：LSE 时钟未被旁路； 1：LSE 时钟被旁路。
1	LSERDY	**LSERDY**：外部低速 LSE 就绪(External low-speed oscillator ready) 由硬件置'1'或清'0'来指示是否外部 32kHz 振荡器就绪。在 LSEON 被清零后，该位需要 6 个外部低速振荡器的周期才被清零。 0：外部 32kHz 振荡器未就绪； 1：外部 32kHz 振荡器就绪。
0	LSEON	**LSEON**：外部低速振荡器使能(External low-speed oscillator enable) 由软件置'1'或清'0' 0：外部 32kHz 振荡器关闭； 1：外部 32kHz 振荡器开启。

6.3.10 控制/状态寄存器(RCC_CSR)

偏移地址：0x24

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

访问：0 到 3 等待周期，字、半字和字节访问

当连续对该寄存器进行访问时，将插入等待状态。

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

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

位	符号	说明
31	LPWRRSTF	**LPWRRSTF**：低功耗复位标志(Low-power reset flag) 在低功耗管理复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无低功耗管理复位发生； 1：发生低功耗管理复位. 关于低功耗管理复位的详细信息，请参考 6.1.1 节的”低功耗管理复位”。
30	WWDGRSTF	**WWDGRSTF**：窗口看门狗复位标志(Window watchdog reset flag)在窗口看门狗复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无窗口看门狗复位发生； 1：发生窗口看门狗复位。
29	IWDGRSTF	**IWDGRSTF**：独立看门狗复位标志(Independent watchdog reset flag) 在独立看门狗复位发生在 VDD 区域时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无独立看门狗复位发生； 1：发生独立看门狗复位。
28	SFTRSTF	**SFTRSTF**：软件复位标志(Software reset flag)

		在软件复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无软件复位发生； 1：发生软件复位。
27	**PORRSTF**	**PORRSTF**：上电/掉电复位标志(POR/PDR reset flag) 在上电/掉电复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无上电/掉电复位发生； 1：发生上电/掉电复位。
26	**PINRSTF**	**PINRSTF**：NRST 引脚复位标志(PIN reset flag) 在 NRST 引脚复位发生时由硬件置'1'；由软件通过写 RMVF 位清除。 0：无 NRST 引脚复位发生； 1：发生 NRST 引脚复位。
25	**Reserved**	保留，读操作返回 0
24	**RMVF**	**RMVF**：清除复位标志(Remove reset flag)由软件置'1'来清除复位标志。 0：无作用； 1：清除复位标志。
23:2	**Reserved**	保留，读操作返回 0
1	**LSIRDY**	**LSIRDY**：内部低速振荡器就绪(Internal low-speed oscillator ready) 由硬件置'1 或清'0'来指示内部 40kHzRC 振荡器是否就绪。在 LSION 清零后，3 个内部 40kHzRC 振荡器的周期后 LSIRDY 被清零。 0：内部 40kHzRC 振荡器时钟未就绪； 1：内部 40kHzRC 振荡器时钟就绪。
0	**LSION**	**LSION**：内部低速振荡器使能(Internal low-speed oscillator enable)由软件置'1 或清'0'。 0：内部 40kHzRC 振荡器关闭； 1：内部 40kHzRC 振荡器开启。

6.3.11 RCC 寄存器地址映像

下表列出了 RCC 寄存器的映像和复位值。

表 12 RCC 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	RCC_CR	保留						PLLRDY	PLLON	保留				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					保留	HSIRDY	HSION
	复位值							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	1
004h	RCC_CFGR	保留						MCO[2:0]						保留	USBPRE	PLLMUL[3:0]																	
	复位值							0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	RCC_CIR	保留								CSSC	保留			PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	保留														
	复位值									0				0	0	0	0	0															

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00Ch	RCC_APB2RSTR	保留										TIM11RST	TIM10RST	TIM9RST	保留			ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	IOPGRST	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	保留	AFIORST
	复位值											0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
010h	RCC_APB1RSTR	保留		DACRST	PWRRST	BKPRST	保留	CANRST	保留	USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	保留	SPI3RST	保留			WWDGRST	保留		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST	
	复位值			0	0	0		0		0	0	0	0	0	0	0		0				0			0	0	0	0	0	0	0	0	0
014h	RCC_AHBENR	保留																					SDIOEN	保留	FSMCEN	保留	CRCEN	保留	FLITFEN	保留	SRAMEN	DMA2EN	DMA1EN
	复位值																						0		0		0		1		1	0	0
018h	RCC_APB2ENR											TIM11EN	TIM10EN	TIM9EN				ADC3EN	USART1EN	TIM8RST	SPI1EN	TIM1EN	ADC2EN	ADC1EN	IOPGEN	IOPFEN	IOPEEN	IPODEN	IOPDEN	IOPCEN	IOPBEN	保留	IOPAEN
	复位值											0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
01Ch	RCC_APB1ENR	保留		DACRST	PWREN	BKPEN	保留	CANEN	保留	USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	保留	SPI3EN	保留			WWDGEN	保留		TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN				
	复位值			0	0	0		0		0	0	0	0	0	0	0		0				0			0	0	0	0	0	0			
020h	RCC_BDCR	保留															BDRST	RTCEN	保留				RTC SEL [1:0]		保留						LSEBYP	LSERDYF	LSEON
	复位值																0	0					0	0							0	0	0
024h	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	保留	RMVF	保留																						LSIRDY	LSION
	复位值	0	0	0	0	1	1		0																							0	0

有关寄存器的起始地址，请参考表 1。

7 通用和复用功能 I/O(GPIO 和 AFIO)

7.1 GPIO 功能描述

每个 GPI/O 端口有两个 32 位配置寄存器(GPIOx_CRL，GPIOx_CRH)，两个 32 位数据寄存器(GPIOx_IDR 和 GPIOx_ODR)，一个 32 位置位/复位寄存器(GPIOx_BSRR)，一个 16 位复位寄存器(GPIOx_BRR)和一个 32 位锁定寄存器(GPIOx_LCKR)。

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

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

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

下图给出了一个 I/O 端口位的基本结构。

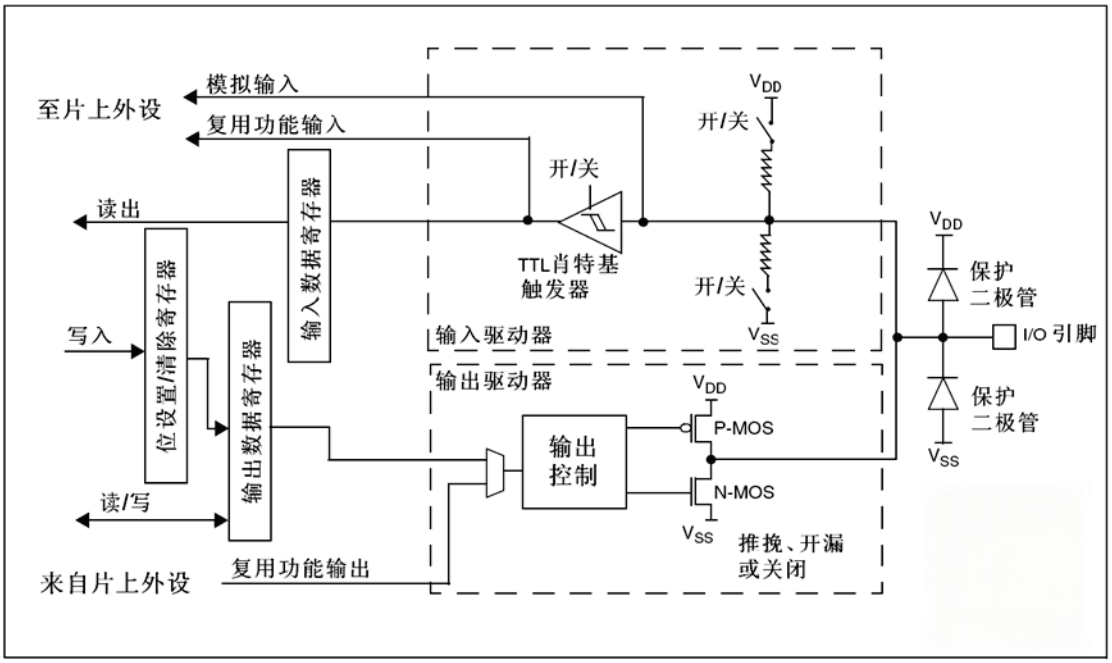


图 9 I/O 端口位的基本结构

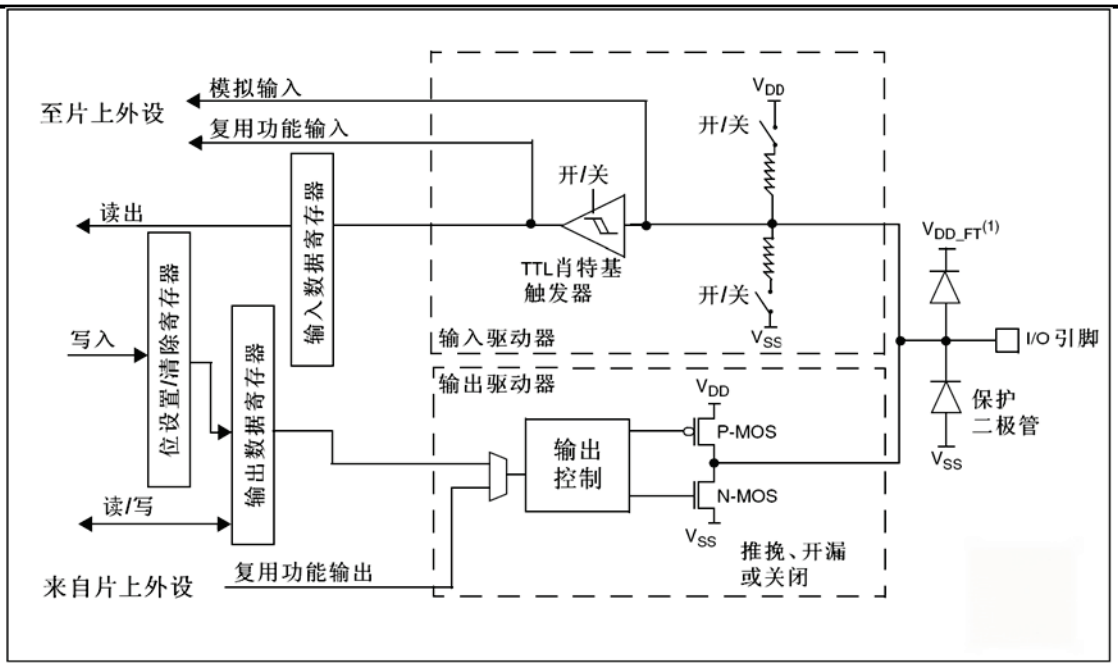


图 10 5 伏兼容 I/O 端口位的基本结构

(1)V_{DD_FT}对 5 伏容忍 I/O 脚是特殊的，它与 VDD 不同

表 13 端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR 寄存器
通用输出	推挽(Push-Pull)	0	0	01 10 11 见表 14		0 或 1
	开漏(Open-Drain)		1			0 或 1
复用功能输出	推挽(Push-Pull)	1	0			不使用
	开漏(Open-Drain)		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表 14 输出模式位

MODE[1:0]	意义
00	保留
01	最大输出速度为 10MHz
10	最大输出速度为 2MHz
11	最大输出速度为 50MHz

7.1.1 通用 I/O(GPIO)

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

复位后，JTAG 引脚被置于输入上拉或下拉模式：

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

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

输入数据寄存器(GPIOx_IDR)在每个 APB2 时钟周期捕捉 I/O 引脚上的数据。
所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以被断开。

7.1.2 单独的位设置或位清除

当对 GPIOx_ODR 的个别位编程时，软件不需要禁止中断：在单次 APB2 写操作里，可以只更改一个或多个位。
这是通过对“置位/复位寄存器”(GPIOx_BSRR，复位是 GPIOx_BRR)中想要更改的位写'1'来实现的。没被选择的位将不被更改。

7.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，参考：

- 第 8.2 节：外部中断/事件控制器(EXTI)；
- 第 8.2.3 节：唤醒事件管理。

7.1.4 复用功能(AF)

使用默认复用功能前必须对端口位配置寄存器编程。

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

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

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

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

7.1.5 软件重新映射 I/O 复用功能

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

7.1.6 GPIO 锁定机制

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

7.1.7 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入被激活

- 根据输入配置(上拉，下拉或浮动)的不同，弱上拉和下拉电阻被连接
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

下图给出了 I/O 端口位的输入配置

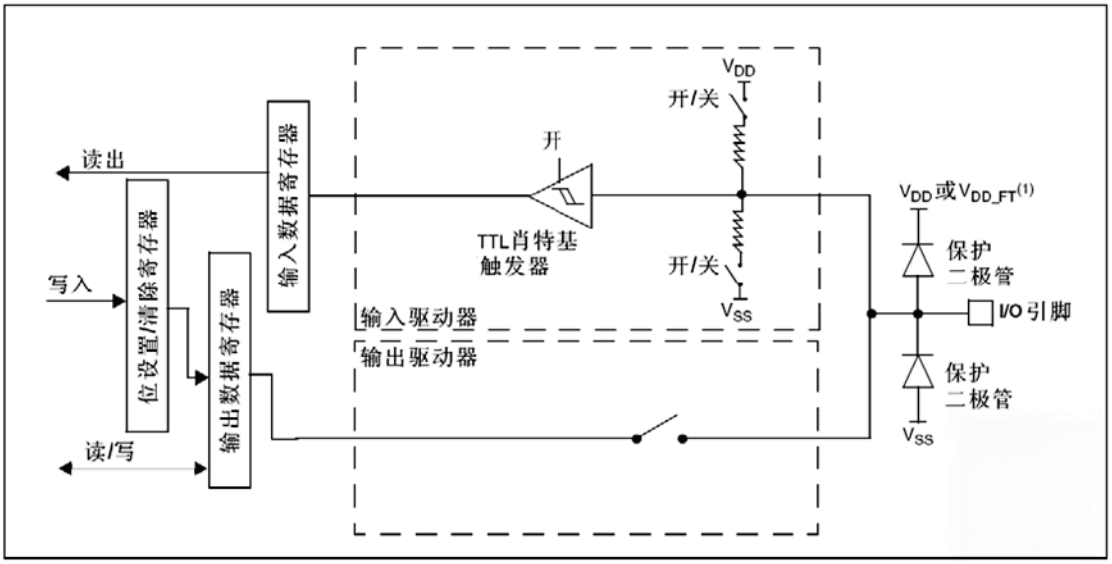


图 11 输入浮空/上拉/下拉配置

(1)V_{DD_FT}对 5 伏容忍 I/O 脚是特殊的，它与 V_{DD}不同

7.1.8 输出配置

当 I/O 端口被配置为输出时:

- 输出缓冲器被激活
  - 开漏模式：输出寄存器上的'0'激活 N-MOS，而输出寄存器上的'1'将端口置于高阻状态(P-MOS 从不被激活)。
  - 推挽模式：输出寄存器上的'0'激活 N-MOS，而输出寄存器上的'1'将激活 P-MOS。
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 在开漏模式时，对输入数据寄存器的读访问可得到 I/O 状态
- 在推挽式模式时，对输出数据寄存器的读访问得到最后一次写的值。

下图给出了 I/O 端口位的输出配置。

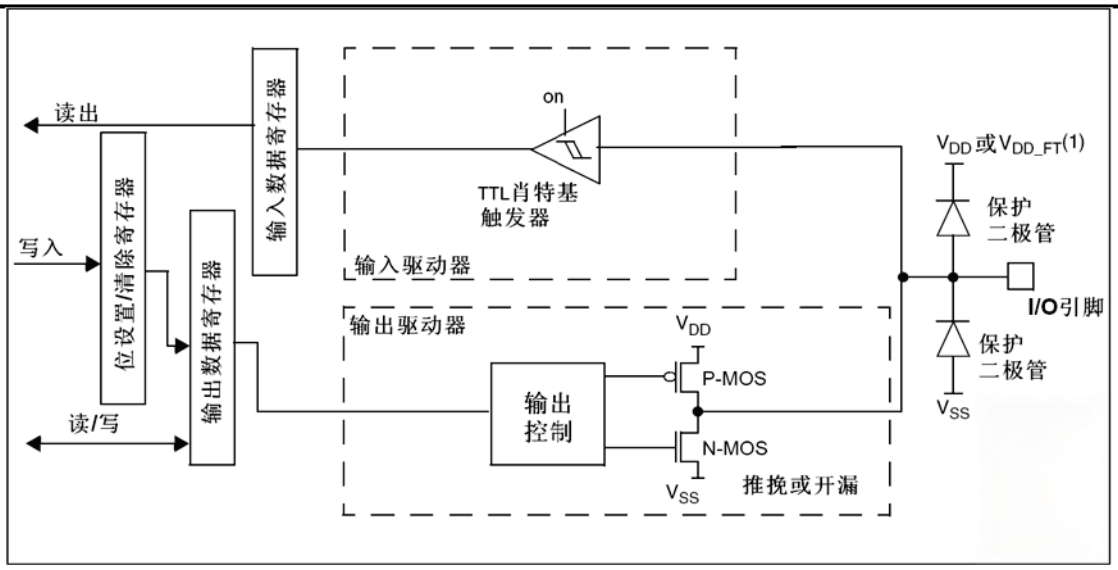


图 12 输出配置

(1)V_{DD_FT}对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD}不同

7.1.9 复用功能配置

当 I/O 端口被配置为复用功能时：

- 在开漏或推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器(复用功能输出)
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 在每个 APB2 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器
- 开漏模式时，读输入数据寄存器时可得到 I/O 口状态
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值

下图示出了 I/O 端口位的复用功能配置。详见 7.3.10 节-AFIO 寄存器描述。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映象到不同的引脚。

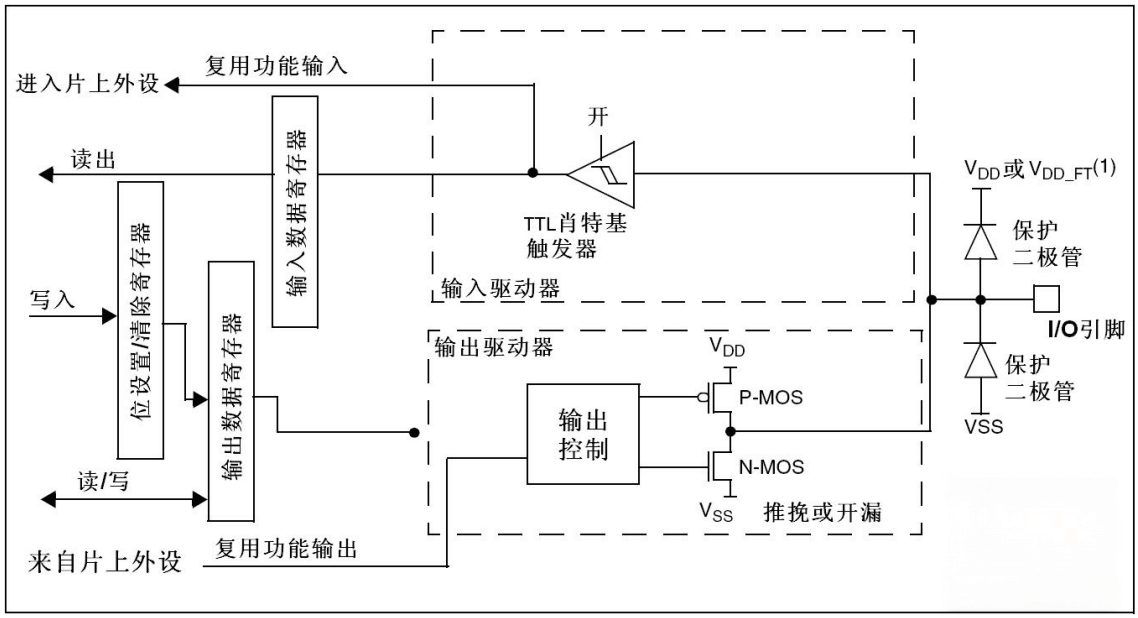


图 13 复用功能配置

(1)V_{DD_FT}对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD}不同

7.1.10 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

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

下图示出了 I/O 端口位的高阻抗模拟输入配置：

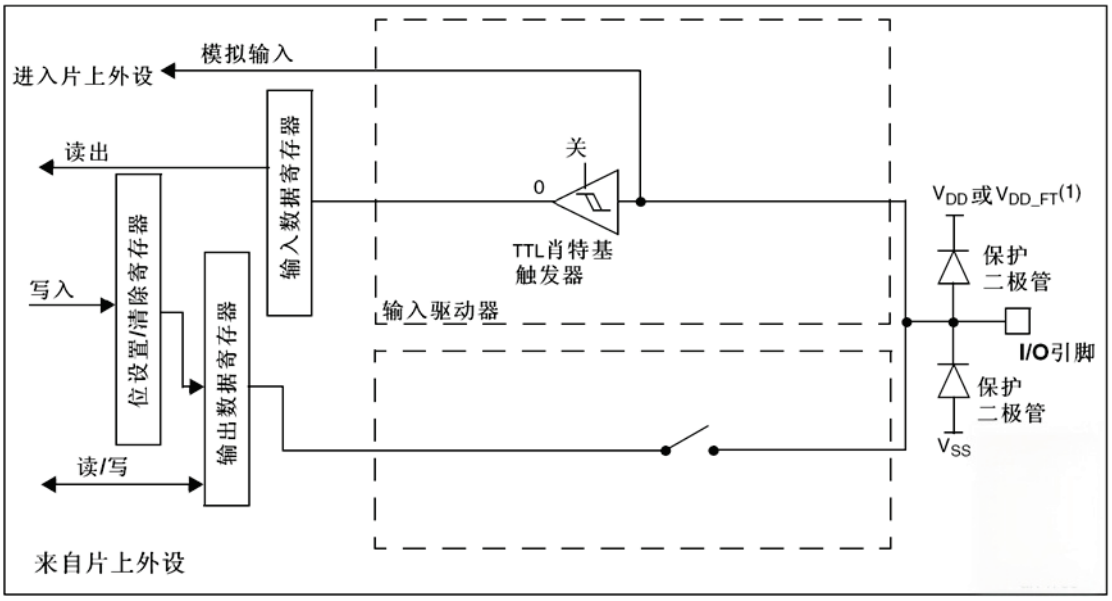


图 14 高阻抗的模拟输入配置

(1)V_{DD_FT}对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD}不同

7.1.11 外设的 GPIO 配置

下列表格列出了各个外设的引脚配置。

表 15 高级定时器 TIM1/TIM8

TIM1/TIM8 引脚	配置	GPIO 配置
TIM1/8_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM1/8_CHxN	互补输出通道 x	推挽复用输出
TIM1/8_BKIN	刹车输入	浮空输入
TIM1/8_ETR	外部触发时钟输入	浮空输入

表 16 通用定时器 TIM2/3/4/5

TIM2/3/4/5 引脚	配置	GPIO 配置
TIM2/3/4/5_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM2/3/4/5_ETR	外部触发时钟输入	浮空输入

表 17 USARTs

USART 引脚	配置	GPIO 配置
USARTx_TX	全双工模式	推挽复用输出
	半双工同步模式	推挽复用输出

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

表 18 SPI

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

表 19 I2S

I2S 引脚	配置	GPIO 配置
I2Sx_WS	主模式	推挽复用输出
	从模式	浮空输入
I2Sx_CK	主模式	推挽复用输出
	从模式	浮空输入
I2Sx_SD	发送器	推挽复用输出
	接收器	浮空输入或带上拉输入或带下拉输入
I2Sx_MCK	主模式	推挽复用输出
	从模式	未用，可作为通用 I/O

表 20 I2C 接口

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

表 21 BxCAN

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

表 22 USB

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

表 23 SDIO

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

ADC 输入引脚必须配置为模拟入

表 24 ADC/DAC

ADC/DAC 引脚	GPIO 配置
ADC/DAC	模拟输入

表 25 其它 I/O 功能

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

7.2 GPIO 寄存器描述

请参考第 1 章中有关寄存器描述中用到的缩写。

必须以字(32 位)的方式操作这些外设寄存器。

7.2.1 端口配置低寄存器(GPIOx_CRL)(x=A..G)

偏移地址：0x00

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[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
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]	**CNFy[1:0]**：端口 x 配置位(y=0…7)(Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态)10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)：00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
29:28 25:24 21:20 17:16 13:12 9:8 5:4 1:0	MODEy[1:0]	**MODEy[1:0]**：端口 x 的模式位(y=0…7)(Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度 10MHz 10：输出模式，最大速度 2MHz 11：输出模式，最大速度 50MHz

7.2.2 端口配置高寄存器(GPIOx_CRH)(x=A..G)

偏移地址：0x04

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[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
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]	**CNFy[1:0]**：端口 x 配置位(y=8...15)(Port x configuration bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态)10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)：00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
29:28 25:24 21:20 17:16 13:12 9:8 5:4 1:0	MODEy[1:0]	**MODEy[1:0]**：端口 x 的模式位(y=8...15)(Port x mode bits) 软件通过这些位配置相应的 I/O 端口，请参考表 13 端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度 10MHz 10：输出模式，最大速度 2MHz 11：输出模式，最大速度 50MHz

7.2.3 端口输入数据寄存器(GPIOx_IDR)(x=A..G)

地址偏移：0x08

复位值：0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

7.2.4 端口输出数据寄存器(GPIOx_ODR)(x=A..G)

地址偏移：0x0Ch

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0

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

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

7.2.5 端口位设置/清除寄存器(GPIOx_BSRR)(x=A..G)

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
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

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

7.2.6 端口位清除寄存器(GPIOx_BRR)(x=A..G)

地址偏移：0x14

复位值：0x00000000

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

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

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:0	BRy	**BRy:**清除端口 x 的位 y(y=0...15)(Port x Reset bit y) 这些位只能写入并只能以字(16 位)的形式操作。 0：对对应的 ODRy 位不产生影响 1：清除对应的 ODRy 位为 0

7.2.7 端口配置锁定寄存器(GPIOx_LCKR)(x=A..G)

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

每个锁定位锁定控制寄存器(CRL,CRH)中相应的 4 个位。

地址偏移：0x18

复位值：0x0000 0000

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

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

7.3 复用功能 I/O 和调试配置(AFIO)

设置复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)实现引脚的重新映射。这时，复用功能不再映射到它们的原始分配上。

7.3.1 把 OSC32_IN/OSC32_OUT 作为 GPIO 端口 PC14/PC15

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

注：1.当关闭 1.8V 电压区(进入待机模式)或后备区域使用 VBAT 供电(不再有 VDD 供电)时，不能使用 PC14/PC15 的 GPIO 口功能；

2．参见第 4.1.2 节有关 I/O 口使用的限制

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

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

这个重映射只适用于 36、48 和 64 脚的封装(100 脚和 144 脚的封装上有单独的 PD0 和 PD1 的引脚，不必重映射)

注：外部中断事件功能没有被重映射。在 36、48 和 64 脚的封装上，PD0 和 PD1 不能用来产生外部中断事件。

7.3.3 CAN1 复用功能重映射

CAN 信号可以被映射到端口 A、端口 B 或端口 D 上，如下表所示。

表 26 CAN1 复用功能重映射

复用功能	CAN_REMAP[1:0]=" 00"	CAN_REMAP[1:0]=" 10"	CAN_REMAP[1:0]=" 11"
CAN1_RX 或 AN_RX	PA11	PB8	PD0
CAN1_TX 或 AN_TX	PA12	PB9	PD1

7.3.4 JTAG/SWD 复用功能重映射

调试接口信号被映射到 GPIO 端口上，如下表所示。

表 27 调试接口信号

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

为了在调试期间可以使用更多 GPIOs，通过设置复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)的 SWJ_CFG[2:0]位，可以改变上述重映像配置。参见下表。

表 28 调试端口映像

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

1.I/O 口只可在不使用异步跟踪时使用。

7.3.5 ADC 复用功能重映射

参阅复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 29 ADC1 外部触发注入转换复用功能重映射

复用功能	ADC1_ETRGINJ_REMAP=0	ADC1_ETRGINJ_REMAP=1
ADC1 外部触发注入转换	ADC1 外部触发注入转换与 EXTI15 相连	ADC1 外部触发注入转换与 TIM8_CH4 相连

表 30 ADC1 外部触发规则转换复用功能重映射

复用功能	ADC1_ETRGREG_REMAP=0	ADC1_ETRGREG_REMAP=1

ADC1 外部触发注入转换	ADC1 外部触发注入转换与 EXTI15 相连	ADC1 外部触发注入转换与 TIM8_CH4 相连

表 31 ADC2 外部触发注入转换复用功能重映射

复用功能	ADC2_ETRGINJ_REMAP=0	ADC2_ETRGINJ_REMAP=1
ADC2 外部触发注入转换	ADC2 外部触发注入转换与 EXTI15 相连	ADC2 外部触发注入转换与 TIM8_CH4 相连

表 32 ADC2 外部触发规则转换复用功能重映射

复用功能	ADC2_ETRGREG_REMAP=0	ADC2_ETRGREG_REMAP=1
ADC2 外部触发规则转换	ADC2 外部触发规则转换与 EXTI11 相连	ADC2 外部触发规则转换与 TIM8_TRGO 相连

7.3.6 定时器复用功能重映射

定时器 4 的通道 1 到通道 4 可以从端口 B 重映射到端口 D。其他定时器的重映射列在表 33~表 37。参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 33 TIM5 复用功能重映像

复用功能	TIM5CH4_IREMAP=0	TIM5CH4_IREMAP=1
TIM5_CH4	TIM5 的通道 4 连至 PA3	LSI 内部时钟连至 TIM5_CH4 的输入作为校准使用

表 34 TIM4 复用功能重映像

复用功能	TIM4_REMAP=0	TIM4_REMAP=1
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

表 35 TIM3 复用功能重映像

复用功能	TIM3_REMAP[1:0]=00 (没有重映像)	TIM3_REMAP[1:0]=10 (部分重映像)	TIM3_REMAP[1:0]=11 (完全重映像)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

表 36 TIM2 复用功能重映像

复用功能	TIM2_REMAP[1:0] =00 (没有重映像)	TIM2_REMAP[1:0] =01 (部分重映像)	TIM2_REMAP[1:0] =10 (部分重映像)(1)	TIM2_REMAP[1:0] =11 (完全重映像)
TIM2_CH1_ETR (1)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1.TIM2_CH1 和 TIM2_ETR 共用一个引脚，但不能同时使用(因此在此使用这样的标记：TIM2_CH1_ETR)

表 37 TIM1 复用功能重映像

复用功能映像	TIM1_REMAP[1:0]=00 (没有重映像)	TIM1_REMAP[1:0]=01 (部分重映像)	TIM1_REMAP[1:0]=11 (完全重映像)
TIM1ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13

TIM1_CH4	PA11		PE14
TIM1BKIN	PB12	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14	PB0	PE10
TIM1_CH3N	PB15	PB1	PE12

表 38 TIM9 重映射 (1)

复用功能映像	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM9_CH1	PA2	PE5
TIM9_CH2	PA3	PE6

1.请参阅 AF 重映射和调试 I/O 配置寄存器，第 xx 节 AF 重映射和调试 I/O 配置寄存器 2 (AFIO_MAPR2)

表 39 TIM10 重映射 (1)

复用功能映像	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM10_CH1	PB8	PF6

1.请参阅 AF 重映射和调试 I/O 配置寄存器，第 xx 节 AF 重映射和调试 I/O 配置寄存器 2 (AFIO_MAPR2)

表 40 TIM11 重映射 (1)

复用功能映像	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM11_CH1	PB9	PF7

1.请参阅 AF 重映射和调试 I/O 配置寄存器，第 xx 节 AF 重映射和调试 I/O 配置寄存器 2 (AFIO_MAPR2)

表 41 TIM13 重映射 (1)

复用功能映像	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM13_CH1	PA6	PF8

1.请参阅 AF 重映射和调试 I/O 配置寄存器，第 xx 节 AF 重映射和调试 I/O 配置寄存器 2 (AFIO_MAPR2)

表 42 TIM14 重映射 (1)

复用功能映像	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM14_CH1	PA7	PF9

1.请参阅 AF 重映射和调试 I/O 配置寄存器，第 xx 节 AF 重映射和调试 I/O 配置寄存器 2 (AFIO_MAPR2)

7.3.7 USART 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)

表 43 USART3 重映像

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

表 44 USART2 重映像

复用功能	USART2_REMAP=0	USART2_REMAP=1
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4

USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

表 45 USART1 重映像

复用功能	USART1_REMAP=0	USART1_REMAP=1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

7.3.8 I2C1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 46 I2C1 重映像

复用功能	I2C1_REMAP=0	I2C1_REMAP=1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

7.3.9 SPI1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)。

表 47 SPI1 重映像

复用功能	SPI1_REMAP=0	SPI1_REMAP=1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

7.3.10 SPI3 复用功能重映射

表 48 SPI3 重映射

复用功能	SPI3_REMAP=0	SPI3_REMAP=1
SPI1_NSS	PA15	PA4
SPI1_SCK	PB3	PC10
SPI1_MISO	PB4	PC11
SPI1_MOSI	PB5	PC12

7.4 AFIO 寄存器描述

请参考第 1 章中有关寄存器描述中用到的缩写。

注意： *对寄存器 AFIO_EVCR，AFIO_MAPR 和 AFIO_EXTICRX 进行读写操作前，应当首先打开 AFIO 的时钟。参考第 6.3.7 节 APB2 外设时钟使能寄存器(RCC_APB2ENR)。*
必须以字(32 位)的方式操作这些外设寄存器。

7.4.1 事件控制寄存器(AFIO_EVCR)

地址偏移：0x00

复位值：0x00000000

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

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

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

7.4.2 复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)

地址偏移：0x04

复位值：0x0000 0000

寄存器映像和位定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留					SWJ_CFG[2:0]			保留			ADC2_ETRGREG_REMAP	ADC2_ETRGINJREMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJREMAP	TIM5CH4_IREMAP
					w	w	w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:27	Reserved	保留，始终读为 0。
26:24	SWJ_CFG[2:0]	**SWJ_CFG[2:0]**：串行线 JTAG 配置(Serial wire JTAG configuration) 这些位只可由软件写(读这些位，将返回未定义的数值)，用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ(串行线 JTAG)支持 JTAG 或 SWD 访问 Cortex 的调试端口。系统复位后的默认状态是启用 SWJ 但没有跟踪功能，这种状态下可以通过 JTMS/JTCK 脚上的特定信号选择 JTAG 或 SW(串行线)模式。 000：完全 SWJ(JTAG-DP+SW-DP)：复位状态； 001：完全 SWJ(JTAG-DP+SW-DP)但没有 NJTRST； 010：关闭 JTAG-DP，启用 SW-DP； 100：关闭 JTAG-DP，关闭 SW-DP；其它组合：无作用。
23:21	Reserved	保留，始终读为 0。
20	ADC2_ETRGREG_REMAP	**ADC2_ETRGREG_REMAP**：ADC2 规则转换外部触发重映射(ADC2 external trigger regular conversionre mapping)

		该位可由软件置'1'或置'0'。它控制与 ADC2 规则转换外部触发相连的触发输入。当该位置'0'时，ADC2 规则转换外部触发与 EXTI11 相连；当该位置'1'时，ADC2 规则转换外部触发与 TIM8_TRGO 相连。
19	ADC2_ETRGINJ_REMAP	**ADC2_ETRGINJ_REMAP**：ADC2 注入转换外部触发重映射(ADC2 external trigger injected conversion remapping) 该位可由软件置'1'或置'0'。它控制与 ADC2 注入转换外部触发相连的触发输入。当该位置'0'时，ADC2 注入转换外部触发与 EXTI15 相连；当该位置'1'时，ADC2 注入转换外部触发与 TIM8 通道 4 相连。
18	ADC1_ETRGREG_REMAP	**ADC1_ETRGREG_REMAP**：ADC1 规则转换外部触发重映射(ADC1 external triggerr egular conversion remapping) 该位可由软件置'1'或置'0'。它控制与 ADC2 规则转换外部触发相连的触发输入。当该位置'0'时，ADC1 规则转换外部触发与 EXTI11 相连；当该位置'1'时，ADC1 规则转换外部触发与 TIM8_TRGO 相连。
17	ADC1_ETRGINJ_REMAP	**ADC1_ETRGINJ_REMAP**：ADC1 注入转换外部触发重映射(ADC1 External trigger injected conversion remapping) 该位可由软件置'1'或置'0'。它控制与 ADC2 注入转换外部触发相连的触发输入。当该位置'0'时，ADC2 注入转换外部触发与 EXTI15 相连；当该位置'1'时，ADC1 注入转换外部触发与 TIM8 通道 4 相连。
16	TIM5CH4_IREMAP	**TIM5CH4_IREMAP**：TIM5 通道 4 内部重映射(TIM5 channel4 internal remap) 该位可由软件置'1'或置'0'。它控制 TIM5 通道 4 内部映像。当该位置'0'时，TIM5_CH4 与 PA3 相连；当该位置'1'时，LSI 内部振荡器与 TIM5_CH4 相连，目的是对 LSI 进行校准。
15	PD01_REMAP	**PD01_REMAP**：端口 D0/端口 D1 映像到 OSC_IN/OSC_OUT(PortD0/PortD1 mappingon OSC_IN/OSC_OUT) 该位可由软件置'1'或置'0'。它控制 PD0 和 PD1 的 GPIO 功能映像。当不使用主振荡器 HSE 时(系统运行于内部的 8MHz 阻容振荡器)，PD0 和 PD1 可以映像到 OSC_IN 和 OSC_OUT 引脚。 0：不进行 PD0 和 PD1 的重映像； 1：PD0 映像到 OSC_IN，PD1 映像到 OSC_OUT。
14:13	CAN_REMAP[1:0]	**CAN_REMAP[1:0]**：CAN 复用功能重映像(CAN alternate function remapping) 这些位可由软件置'1'或置'0'，控制复用功能 CAN_RX 和 CAN_TX 的重映像。 00：CAN_RX 映像到 PA11，CAN_TX 映像到 PA12； 01：未用组合； 10：CAN_RX 映像到 PB8，CAN_TX 映像到 PB9； 11：CAN_RX 映像到 PD0，CAN_TX 映像到 PD1。
12	TIM4_REMAP	**TIM4_REMAP**：定时器 4 的重映像(TIM4 remapping) 该位可由软件置'1'或置'0'，控制将 TIM4 的通道 1-4 映射到 GPIO 端口上。 0：没有重映像(TIM4_CH1/PB6，TIM4_CH2/PB7，TIM4_CH3/PB8，TIM4_CH4/PB9)； 1：完全映像(TIM4_CH1/PD12，TIM4_CH2/PD13，TIM4_CH3/PD14，TIM4_CH4/PD15)。 注：重映像不影响在 PE0 上的 TIM4_ETR。
11:10	TIM3_REMAP[1:0]	**TIM3_REMAP[1:0]**：定时器 3 的重映像(TIM3 remapping) 这些位可由软件置'1'或置'0'，控制定时器 3 的通道 1 至 4 在 GPIO 端口的映像。 00：没有重映像(CH1/PA6，CH2/PA7，CH3/PB0，CH4/PB1)；01：未用组合； 10：部分映像(CH1/PB4，CH2/PB5，CH3/PB0，CH4/PB1)；11：完全映像(CH1/PC6，CH2/PC7，CH3/PC8，CH4/PC9)。注：重映像不影响在 PD2 上的 TIM3_ETR。
9:8	TIM2_REMAP[1:0]	**TIM2_REMAP[1:0]**：定时器 2 的重映像(TIM2 remapping) 这些位可由软件置'1'或置'0'，控制定时器 2 的通道 1 至 4 和外部触发(ETR)在 GPIO 端口的映像。00：没有重映像(CH1/ETR/PA0，CH2/PA1，CH3/PA2，CH4/PA3)； 01：部分映像(CH1/ETR/PA15，CH2/PB3，CH3/PA2，CH4/PA3)； 10：部分映像(CH1/ETR/PA0，CH2/PA1，CH3/PB10，CH4/PB11)； 11：完全映像(CH1/ETR/PA15，CH2/PB3，CH3/PB10，CH4/PB11)。
7:6	TIM1_REMAP[1:0]	**TIM1_REMAP[1:0]**：定时器 1 的重映像(TIM1 remapping)

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

7.4.3 外部中断配置寄存器 1(AFIO_EXTICR1)

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

位	符号	说明
31:16	**Reserved**	保留，始终读为 0。
15:0	**EXTI**	**EXTIx[3:0]**：EXTIx 配置(x=0...3)(EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参看 7.2.5 节。 0000：PA[x]引脚　　0100：PE[x]引脚 0001：PB[x]引脚　　0101：PF[x]引脚 0010：PC[x]引脚　　0110：PG[x]引脚

		0011：PD[x]引脚

7.4.4 外部中断配置寄存器 2(AFIO_EXTICR2)

地址偏移：0x0C

复位值：0x0000

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

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:0	EXTI	**EXTIx[3:0]**：EXTIx 配置(x=4...7)(EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 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 0110：PG[x]引脚

7.4.5 外部中断配置寄存器 3(AFIO_EXTICR3)

地址偏移：0x10

复位值：0x0000

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

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:0	EXTI	**EXTIx[3:0]**：EXTIx 配置(x=8...11)(EXTI x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 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 0110：PG[x]引脚

7.4.6 外部中断配置寄存器 4(AFIO_EXTICR4)

地址偏移：0x14

复位值：0x0000

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

位	符号	说明
31:16	Reserved	保留，始终读为 0。

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

7.4.7 AF 重新映射并调试 I/O 配置寄存器 2(AFIO_MAPR2)

地址偏移：0x1C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						TIM14_REMAP	TIM13_REMAP	TIM11_REMAP	TIM10_REMAP	TIM9_REMAP	保留				
						rw	rw	rw	rw	rw					

位	符号	说明
31:10	Reserved	保留，始终读为 0。
9	TIM14_ REMAP	**TIM14_ REMAP:**TIM14 重映射 这个位由软件设置和清除。它控制 TIM14CH1 替代函数映射到 GPIO 端口。 0:无重新映射(PA7) 1:重新映射(PF9)
8	TIM13_REMAP	**TIM13_REMAP:**TIM13 重映射 这个位由软件设置和清除。它控制 TIM13_CH1 替代函数映射到 GPIO 端口。 0:无重新映射(PA6) 1:重新映射(PF8)
7	TIM11_REMAP	**TIM11_REMAP:**TIM11 重映射 这个位由软件设置和清除。它控制 TIM11_CH1 替代函数映射到 GPIO 端口。 0:无重新映射(PB9) 1:重新映射(PF7)
6	TIM10_REMAP	**TIM10_REMAP:**TIM10 重映射 这个位由软件设置和清除。它控制 TIM10_CH1 替代函数映射到 GPIO 端口。 0:无重新映射(PB8) 1:重新映射(PF6)
5	TIM9_REMAP	**TIM9_REMAP:**TIM9 重映射 这个位由软件设置和清除。它控制 TIM9_CH1 替代函数映射到 GPIO 端口。 0:无重新映射(在 PA2 上 TIM9_CH1,在 PA3 上 TIM9_CH2) 1:重新映射(在 PE5 上 TIM9_CH1,在 PE6 上 TIM9_CH2)
4:0	Reserved	保留，始终读为 0。

7.5 GPIO 和 AFIO 寄存器地址映象

关于寄存器起始地址，请参考表 1。下面列出了 GPIO 和 AFIO 寄存器映象和复位数值。

表 49 GPIO 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	GPIOx_CRL	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]		CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
	复位值	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

004h	GPIOx_CRH	CNF15 [1:0]		MODE15 [1:0]		CON14 [1:0]		MODE14 [1:0]		CON13 [1:0]		MODE13 [1:0]		CON12 [1:0]		MODE12 [1:0]		CON11 [1:0]		MODE11 [1:0]		CON10 [1:0]		MODE10 [1:0]		CON9 [1:0]		MODE9 [1:0]		CON8 [1:0]		MODE8 [1:0]	
	复位值	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
008h	GPIOx_IDR	保留																IDR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	GPIOx_ODR	保留																ODR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	GPIOx_BSRR	BR[15:0]																BSR[15:0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	GPIOx_BRR	保留																BR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	GPIOx_LCKR	保留															LCK	LCK[15:0]															
	复位值																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 50 AFIO 寄存器地址映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	AFIO_EVCR	保留																								EVOE	PORT [2:0]			PIN[3:0]			
	复位值																									0	0	0	0	0	0	0	0
004h	AFIO_MAPR	保留					SWJ_CFG [2:0]			保留			ADC2_ETRGREG_REMA	ADC2_ETRGINJ_REMA	ADC1_ETRGREG_REMA	ADC1_ETRGINJ_REMA	TIM5CH4_IREMAP	PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	复位值						0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	AFIO_EXTICR1	保留																	EXTI3 [3:0]				EXTI2 [3:0]			EXTI1 [3:0]				EXTI0 [3:0]			
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	AFIO_EXTICR2	保留																	EXTI7[3:0]				EXTI6[3:0]			EXTI5[3:0]				EXTI4[3:0]			
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	AFIO_EXTICR3	保留																	EXTI11[3:0]				EXTI10[3:0]			EXTI9[3:0]				EXTI8[3:0]			
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	AFIO_EXTICR4	保留																	EXTI15[3:0]				EXTI14[3:0]			EXTI13[3:0]				EXTI12[3:0]			
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	AFIO_EXTICR2	保留																								FSMC_NADV	TIM14_REMAP	TIM13_REMAP	TIM11_REMAP	TIM10_REMAP	TIM9_REMAP	保留		
	复位值																									0	0	0	0	0	0		

8 中断和事件

8.1 嵌套向量中断控制器

特性

- 68 个可屏蔽中断通道(不包含 16 个 Cortex™-M3 的中断线);
- 16 个可编程的优先等级(使用了 4 位中断优先级);
- 低延迟的异常和中断处理;
- 电源管理控制;
- 系统控制寄存器的实现;

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

嵌套向量中断控制器管理着包括内核异常等中断。

8.1.1 系统嘀嗒(SysTick)校准值寄存器

系统嘀嗒校准值固定为 9000，当系统嘀嗒时钟设定为 9MHz(HCLK/8 的最大值)，产生 1ms 时间基准。

8.1.2 中断和异常向量

表 51 W55MH32 产品向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000 0000
	-3	固定	Reset	复位	0x0000 0004
	-2	固定	NMI	不可屏蔽中断 RCC 时钟安全系统(CSS)联接到 NMI 向量	0x0000 0008
	-1	固定	硬件失效(Hard Fault)	所有类型的失效	0x0000 000C
	0	可设置	存储管理(Mem Manage)	存储器管理	0x0000 0010
	1	可设置	总线错误(Bus Fault)	预取指失败，存储器访问失败	0x0000 0014
	2	可设置	错误应用(Usage Fault)	未定义的指令或非法状态	0x0000 0018
	-	-	-	保留	0x0000 001C-0x0000 002B
	3	可设置	SVCall	通过 SWI 指令的系统服务调用	0x0000 002C
	4	可设置	调试监控(Debug Monitor)	调试监控器	0x0000 0030
	-	-	-	保留	0x0000 0034
	5	可设置	PendSV	可挂起的系统服务	0x0000 0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000 003C
0	7	可设置	WWDG	窗口定时器中断	0x0000 0040
1	8	可设置	PVD	连到 EXTI 的电源电压检测(PVD)中断	0x0000 0044
2	9	可设置	TAMPER	侵入检测中断	0x0000 0048
3	10	可设置	RTC	实时时钟(RTC)全局中断	0x0000 004C

位置	优先级	优先级类型	名称	说明	地址
4	11	可设置	FLASH	闪存全局中断	0x0000 0050
5	12	可设置	RCC	复位和时钟控制(RCC)中断	0x0000 0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000 0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000 005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000 0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000 0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000 0068
11	18	可设置	DMA1 通道 1	DMA1 通道 1 全局中断	0x0000 006C
12	19	可设置	DMA1 通道 2	DMA1 通道 2 全局中断	0x0000 0070
13	20	可设置	DMA1 通道 3	DMA1 通道 3 全局中断	0x0000 0074
14	21	可设置	DMA1 通道 4	DMA1 通道 4 全局中断	0x0000 0078
15	22	可设置	DMA1 通道 5	DMA1 通道 5 全局中断	0x0000 007C
16	23	可设置	DMA1 通道 6	DMA1 通道 6 全局中断	0x0000 0080
17	24	可设置	DMA1 通道 7	DMA1 通道 7 全局中断	0x0000 0084
18	25	可设置	ADC1_2	ADC1 和 ADC2 的全局中断	0x0000 0088
19	26	可设置	USB_HP_CAN_TX	USB 高优先级或 CAN 发送中断	0x0000 008C
20	27	可设置	USB_LP_CAN_RX0	USB 低优先级或 CAN 接收 0 中断	0x0000 0090
21	28	可设置	CAN_RX1	CAN 接收 1 中断	0x0000 0094
22	29	可设置	CAN_SCE	CANSCE 中断	0x0000 0098
23	30	可设置	EXTI9_5	EXTI 线[9:5]中断	0x0000 009C
24	31	可设置	TIM1BRK	TIM1 刹车中断	0x0000 00A0
25	32	可设置	TIM1_UP	TIM1 更新中断	0x0000 00A4
26	33	可设置	TIM1_TRG_COM	TIM1 触发和通信中断	0x0000 00A8
27	34	可设置	TIM1_CC	TIM1 捕获比较中断	0x0000 00AC
28	35	可设置	TIM2	TIM2 全局中断	0x0000 00B0
29	36	可设置	TIM3	TIM3 全局中断	0x0000 00B4
30	37	可设置	TIM4	TIM4 全局中断	0x0000 00B8
31	38	可设置	I2C1_EV	I2C1 事件中断	0x0000 00BC
32	39	可设置	I2C1_ER	I2C1 错误中断	0x0000 00C0
33	40	可设置	I2C2_EV	I2C2 事件中断	0x0000 00C4
34	41	可设置	I2C2_ER	I2C2 错误中断	0x0000 00C8
35	42	可设置	SPI1	SPI1 全局中断	0x0000 00CC
36	43	-	保留	保留	0x0000 00D0
37	44	可设置	USART1	USART1 全局中断	0x0000 00D4
38	45	可设置	USART2	USART2 全局中断	0x0000 00D8
39	46	可设置	USART3	USART3 全局中断	0x0000 00DC
40	47	可设置	EXTI15_10	EXTI 线[15:10]中断	0x0000 00E0
41	48	可设置	RTCAlarm	连到 EXTI 的 RTC 闹钟中断	0x0000 00E4
42	49	可设置	USB 唤醒	连到 EXTI 的从 USB 待机唤醒中断	0x0000 00E8
43	50	可设置	TIM8_BRK	TIM8 刹车中断	0x0000 00EC
44	51	可设置	TIM8_UP	TIM8 更新中断	0x0000 00F0
45	52	可设置	TIM8_TRG_COM	TIM8 触发和通信中断	0x0000 00F4
46	53	可设置	TIM8_CC	TIM8 捕获比较中断	0x0000 00F8
47	54	可设置	ADC3	ADC3 全局中断	0x0000 00FC

位置	优先级	优先级类型	名称	说明	地址
48	55	-	保留	保留	0x0000 0100
49	56	可设置	SDIO	SDIO 全局中断	0x0000 0104
50	57	可设置	TIM5	TIM5 全局中断	0x0000 0108
51	58	可设置	SPI3	SPI3 全局中断	0x0000 010C
52	59	可设置	UART4	UART4 全局中断	0x0000 0110
53	60	可设置	UART5	UART5 全局中断	0x0000 0114
54	61	可设置	TIM6	TIM6 全局中断	0x0000 0118
55	62	可设置	TIM7	TIM7 全局中断	0x0000 011C
56	63	可设置	DMA2 通道 1	DMA2 通道 1 全局中断	0x0000 0120
57	64	可设置	DMA2 通道 2	DMA2 通道 2 全局中断	0x0000 0124
58	65	可设置	DMA2 通道 3	DMA2 通道 3 全局中断	0x0000 0128
59	66	可设置	DMA2 通道 4_5	DMA2 通道 4 和 DMA2 通道 5 全局中断	0x0000 012C

8.2 外部中断/事件控制器(EXTI)

W55MH32 有 19 个能产生事件/中断请求的边沿检测器。每个输入线可以独立地配置输入类型(脉冲或挂起)和对应的触发事件(上升沿或下降沿或者双边沿都触发)。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

8.2.1 主要特性

EXTI 控制器的主要特性如下:

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

8.2.2 框图

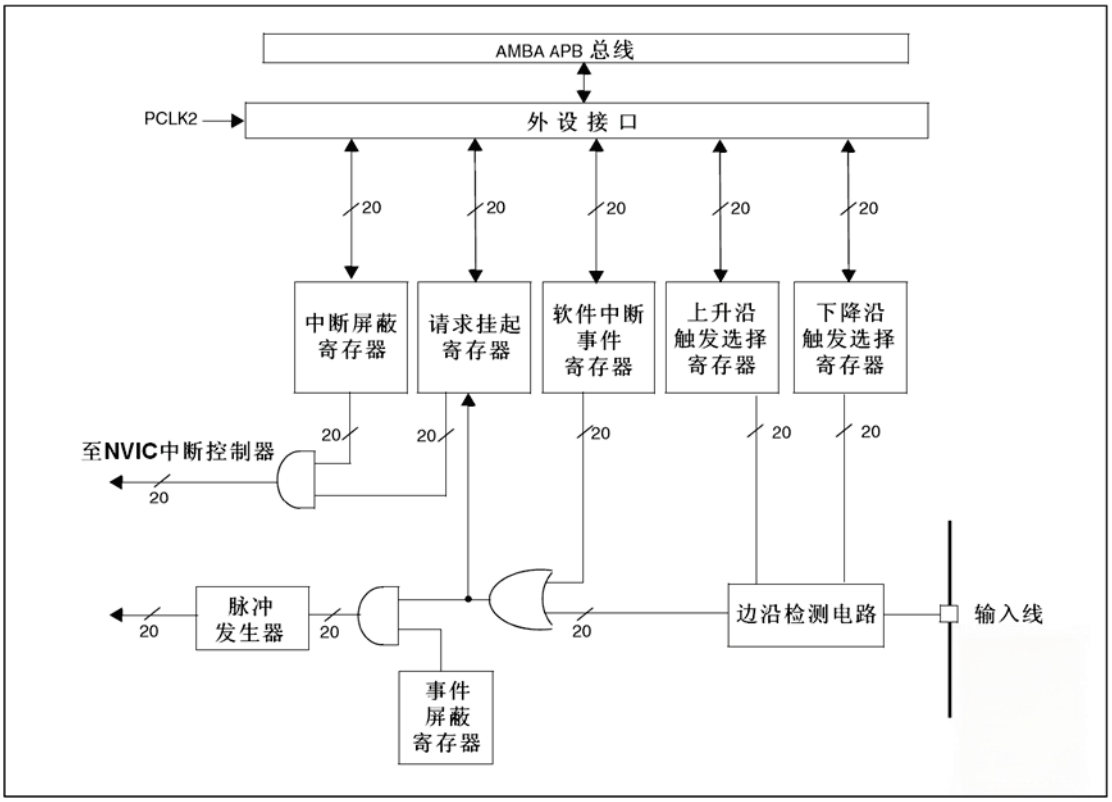


图 15 外部中断/事件控制器框图

8.2.3 唤醒事件管理

W55MH32 可以处理外部或内部事件来唤醒内核(WFE)。唤醒事件可以通过下述配置产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 Cortex-M3 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)。
- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

使用外部 I/O 端口作为唤醒事件，请参见 7.2.4 节的功能说明

8.2.4 功能说明

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

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

通过在软件中断/事件寄存器写'1'，也可以通过软件产生中断/事件请求。

硬件中断选择

通过下面的过程来配置 20 个线路做为中断源：

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

硬件事件选择

通过下面的过程，可以配置 20 个线路为事件源

- 配置 20 个事件线的屏蔽位(EXTI_EMR)
- 配置事件线的触发选择位(EXTI_RTSR 和 EXTI_FTSR)

软件中断/事件的选择

20 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

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

8.2.5 外部中断/事件线路映像

61 个通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

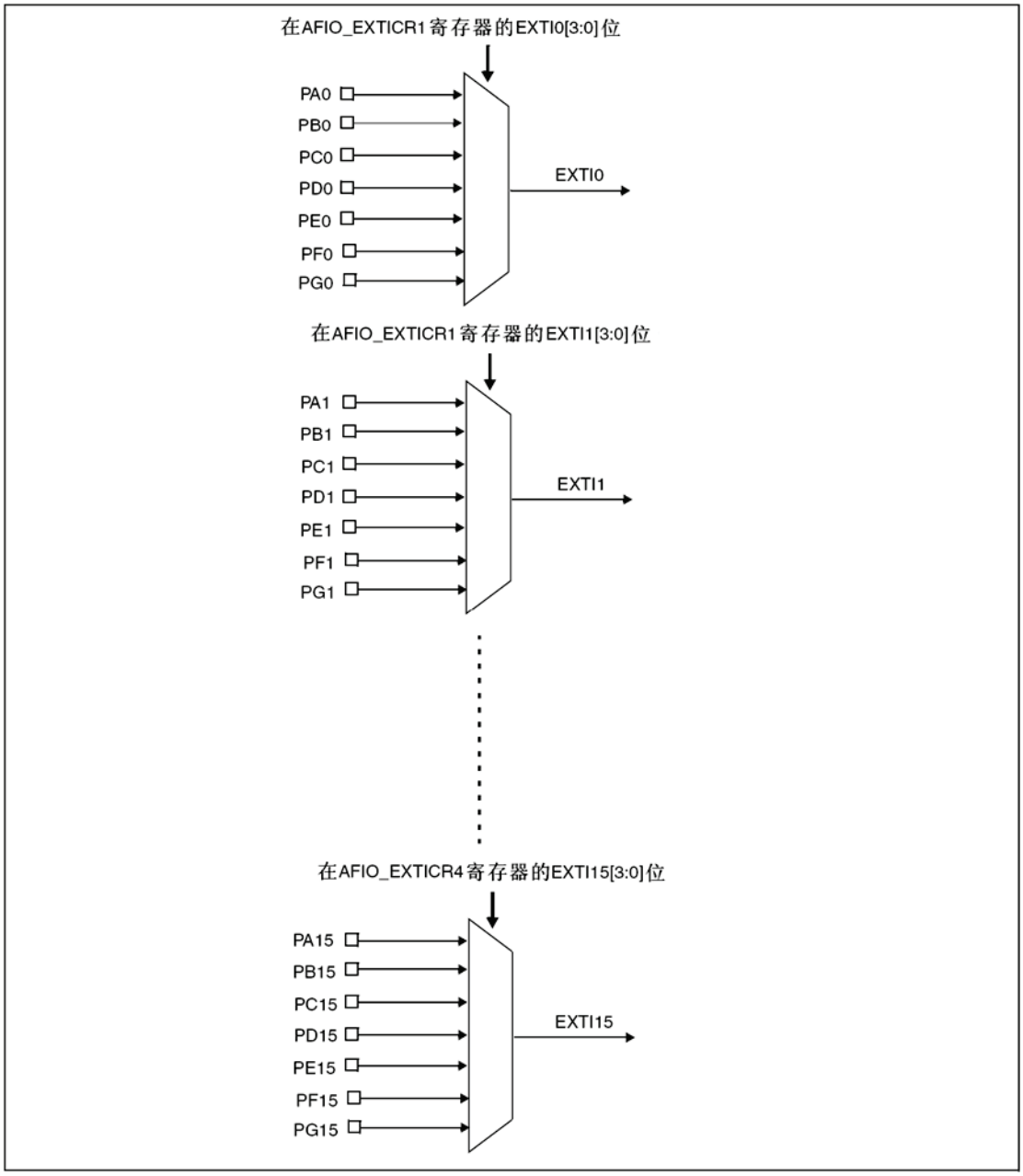


图 16 外部中断通用 I/O 映像

1．通过 AFIO_EXTICRx 配置 GPIO 线上的外部中断/事件，必须先使能 AFIO 时钟。

另外四个 EXTI 线的连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB 唤醒事件

8.3 EXTI 寄存器描述

关于寄存器描述中的缩略词，请参考第 1 节。

必须以字(32 位)的方式操作这些外设寄存器。

8.3.1 中断屏蔽寄存器(EXTI_IMR)

偏移地址：0x00

复位值：0x0000 0000

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

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

位	符号	说明
31:19	Reserved	保留，始终读为 0。
18:0	MRx	**MRx**:线 x 上的中断屏蔽(Interrupt Mask on line x) 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。

8.3.2 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
保留													MR18	MR17	MR16
													rw	rw	rw

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

位	符号	说明
31:19	Reserved	保留，始终读为 0。
18:0	MRx	**MRx**:线 x 上的事件屏蔽(Event Mask on line x) 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。

8.3.3 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
保留													TR18	TR17	TR16
													rw	rw	rw

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

位	符号	说明
31:19	Reserved	保留，始终读为 0。
18:0	TRx	**TRx**:线 x 上的上升沿触发事件配置位(Rising trigger event configuration bit of line x) 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。

在写 EXTI_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。

8.3.4 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
保留													TR18	TR17	TR16
													rw	rw	rw

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

位	符号	说明
31:19	**Reserved**	保留，始终读为 0。
18:0	**TRx**	**TRx:**线 x 上的下降沿触发事件配置位(Falling trigger event configuration bit of line x) 0：屏蔽来自线 x 上的中断请求； 1：开放来自线 x 上的中断请求。

注意: 外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。

在写 EXTI_FTSR 寄存器时，在外部中断线上的下降沿信号不能被识别，挂起位不会被置位。在同一中断线上，可以同时设置上升沿和下降沿触发。即任一边沿都可触发中断。

8.3.5 软件中断事件寄存器(EXTI_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
保留													WIER18	WIER17	WIER16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WIER15	WIER14	WIER13	WIER12	WIER11	WIER10	WIER9	WIER8	WIER7	WIER6	WIER5	WIER4	WIER3	WIER2	WIER1	WIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:19	**Reserved**	保留，始终读为 0。
18:0	**SWIERx**	**SWIERx:**线 x 上的软件中断(Software interrupt on line x) 当该位为'0'时，写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断，则此时将产生一个中断。 注：通过清除 EXTI_PR 的对应位(写入'1')，可以清除该位为'0'。

8.3.6 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
保留													WIER18	WIER17	WIER16
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WIER15	WIER14	WIER13	WIER12	WIER11	WIER10	WIER9	WIER8	WIER7	WIER6	WIER5	WIER4	WIER3	WIER2	WIER1	WIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:19	Reserved	保留，始终读为 0。
18:0	PRx	**PRx**:挂起位(Pending bit) 0：没有发生触发请求 1：发生了选择的触发请求 当在外部中断线上发生了选择的边沿事件，该位被置'1'。在该位中写入'1'可以清除它，也可以通过改变边沿检测的极性清除。

8.3.7 外部中断/事件寄存器映像

下表列出了 EXTI 寄存器的映像和复位值。

表 52 外部中断/事件控制器寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	EXTI_IMR	保留													MR[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	EXTI_EMR	保留													MR[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	EXTI_RTSR	保留													TR[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	EXTI_FTSR	保留													TR[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	EXTI_SWIER	保留													SWIER[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	EXTI_PR	保留													PR[18:0]																		
	复位值														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，参见表 1。

9 TCP/IP 卸载引擎(TOE)

9.1 TOE 简介

TCP/IP 卸载引擎（TOE）是一个嵌入式全硬件 TCP/IP 以太网控制器，它可以提供更简洁的嵌入式网络接入方案。10/100M 以太网数据链路层（MAC）及物理层（PHY），使得用户使用单芯片就能够在他们的应用中拓展网络连接。

久经市场考验的 WIZnet 全硬件 TCP/IP 协议栈支持 TCP， UDP， IPv4， ICMP，ARP， IGMP 以及 PPPoE 协议。 内嵌 32K 字节片上缓存以供以太网包处理。如果你使用 TCP/IP 卸载引擎（TOE），你只需要一些简单的 Socket 编程就能实现以太网应用。这将会比其他嵌入式以太网方案更加快捷、简便。用户可以同时使用 8 个硬件 Socket 独立通讯。为了减少系统能耗， 提供了网络唤醒模式（WOL）及掉电模式供客户选择使用。

9.2 TOE 主要特性

- 支持全硬件 TCP/IP 协议：TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE
- 支持 8 路独立 Sockets
- 支持休眠模式
- 支持 UDP 网络唤醒
- 独立 32KB TX/RX 缓存
- 支持掉电模式
- 10BaseT/100BaseTX 以太网物理层（PHY）
- 支持自动协商（10/100-Based 全双工/半双工）
- LED 状态显示（全双工/半双工，网络连接，网络速度，活动状态）

9.3 寄存器和内存构成

TCP/IP 卸载引擎（TOE） 有 1 个通用寄存器，8 个 Socket 寄存器区，以及对应每个 Socket 的收发缓存区。每一个 Socket 的发送缓存区都在一个 16KB 的物理发送内存中，初始化分配为 2KB。每一个 Socket 的接收缓存区都在一个 16KB 的物理接收内存中，初始化分配为 2KB。

无论给每个 Socket 分配多大的收/发缓存，都必须在 16 位的偏移地址范围内（从 0x0000 到 0xFFFF）。关于 16KB 收/发内存的构成及访问方式的更多信息，请参考 9.4 节。

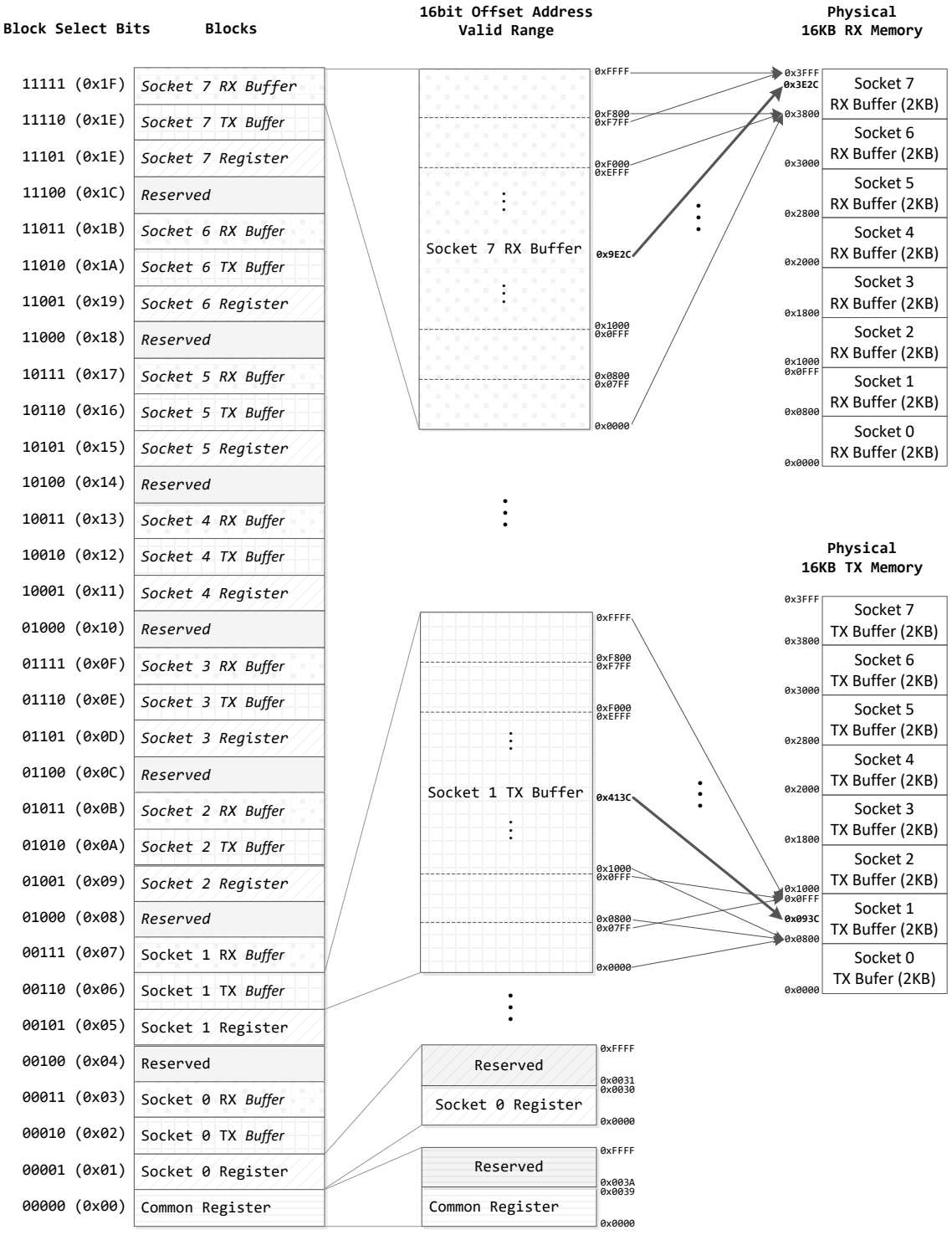


图 17 寄存器及内存构成

9.3.1 通用寄存器区

通用寄存器区配置了 TOE 的基本信息，例如：IP 及 MAC 地址。表 53 描述了通用寄存器的偏移地址。对于每个寄存器的详细信息，请参考 9.5.1 节。

表 53 通用寄存器的偏移地址

Offset	Register	Offset	Register	Offset	Register
0x0000	模式 (MR)	0x0013 0x0014	Interrupt Low Level Timer (INTLEVEL0) (INTLEVEL1)	0x0021 0x0022 0x0023	(PHAR3) (PHAR4) (PHAR5)
0x0001 0x0002 0x0003 0x0004	Gateway Address (GAR0) (GAR1) (GAR2) (GAR3)	0x0015	Interrupt (IR)	0x0024 0x0025	PPP Session Identification (PSID0) (PSID1)
		0x0016	Interrupt Mask (IMR)	0x0026 0x0027	PPP Maximum Segment Size (PMRU0) (PMRU1)
0x0005 0x0006 0x0007 0x0008	Subnet Mask Address (SUBR0) (SUBR1) (SUBR2) (SUBR3)	0x0017	Socket Interrupt (SIR)		
		0x0018	Socket Interrupt Mask (SIMR)	0x0028 0x0029 0x002A 0x002B	Unreachable IP address (UIPR0) (UIPR1) (UIPR2) (UIPR3)
0x0009 0x000A 0x000B 0x000C 0x000D 0x000E	Source Hardware Address (SHAR0) (SHAR1) (SHAR2) (SHAR3) (SHAR4) (SHAR5)	0x0019 0x001A	Retry Time (RTR0) (RTR1)		
		0x001B	Retry Count (RCR)	0x002C 0x002D	Unreachable Port (UPORTR0) (UPORTR1)
		0x001C	PPP LCP Request Timer (PTIMER)		
		0x001D	PPP LCP Magic number (PMAGIC)	0x002E	PHY Configuration (PHYCFGR)
0x000F 0x0010 0x0011 0x0012	Source IP Address (SIPR0) (SIPR1) (SIPR2) (SIPR3)	0x001E 0x001F 0x0020	PPP Destination MAC Address (PHAR0) (PHAR1) (PHAR2)	0x002F ~ 0x0038	Reserved
				0x0039	Chip version (VERSIONR)
0x003A ~ 0xFFFF		Reserved			

9.3.2 Socket 寄存器区

TOE 支持 8 个 Socket 作为通讯信道。每一个 Socket 通过 Socket n 寄存器区控制（0≤n≤7）。
表 54 定义了 Socket n 寄存器区对应的 16 位偏移地址。
关于每个寄存器，详情参考 9.5.2 节

表 54 Socket n 寄存器中的偏移地址(0≤n≤7)

Offset	Register	Offset	Register	Offset	Register
0x0000	Socket n 模式 (Sn_MR)	0x0010 0x0011	Socket n Destination Port (Sn_DPORT0) (Sn_DPORT1)	0x0024 0x0025	Socket n TX Write Pointer (Sn_TX_WR0) (Sn_TX_WR1)
0x0001	Socket n Command (Sn_CR)	0x0012 0x0013	Socket n Maximum Segment Size (Sn_MSSR0) (Sn_MSSR1)	0x0026 0x0027	Socket n RX Received Size (Sn_RX_RSR0) (Sn_RX_RSR1)
0x0002	Socket n Interrupt (Sn_IR)	0x0014	Reserved	0x0028 0x0029	Socket n RX Read Pointer (Sn_RX_RD0) (Sn_RX_RD1)
0x0003	Socket n Status (Sn_SR)	0x0015	Socket n IP TOS (Sn_TOS)	0x002A 0x002B	Socket n RX Write Pointer (Sn_RX_WR0) (Sn_RX_WR1)
0x0004 0x0005	Socket n Source Port (Sn_PORT0) (Sn_PORT1)	0x0016	Socket n IP TTL (Sn_TTL)	0x002C	Socket n Interrupt Mask (Sn_IMR)
0x0006 0x0007 0x0008 0x0009 0x000A 0x000B	Socket n Destination Hardware Address (Sn_DHAR0) (Sn_DHAR1) (Sn_DHAR2) (Sn_DHAR3) (Sn_DHAR4) (Sn_DHAR5)	0x0017 ~ 0x001D	Reserved	0x002D 0x002E	Socket n Fragment Offset in IP header (Sn_FRAG0) (Sn_FRAG1)
		0x001E	Socket n Receive Buffer Size (Sn_RXBUF_SIZE)	0x002F	Keep alive timer (Sn_KPALVTR)
0x000C 0x000D 0x000E 0x000F	Socket n Destination IP Address (Sn_DIPR0) (Sn_DIPR1) (Sn_DIPR2) (Sn_DIPR3)	0x001F	Socket n Transmit Buffer Size (Sn_TXBUF_SIZE)	0x0030 ~ 0xFFFF	Reserved
		0x0020 0x0021	Socket n TX Free Size (Sn_TX_FSR0) (Sn_TX_FSR1)		
		0x0022 0x0023	Socket n TX Read Pointer (Sn_TX_RD0) (Sn_TX_RD1)		

9.4 内存 Memory

TOE 有一个 16KB 的发送内存用于 Socket n 的发送缓存区，以及一个 16KB 的接收内存用于 Socket n 的接收缓存区。
16KB 的发送内存初始化被分配为每个 Socket 2KB 发送缓存区（2KB X 8 = 16KB）。初始化分配的 2KB Socket 发送缓存，可以通过使用 Socket 发送缓存大小寄存器（Sn_TXBUF_SIZE）重新分配。一旦所有的 Socket 发送缓存大小寄存器（Sn_TXBUF_SIZE）配置完成，16KB 的发送内存就会按照配置分配给每个 Socket 的发送缓存，并按照从 Socket 0 到 7 顺序分配。16KB 物理内存的地址是可以自增的。但是，为了避免数据传输错误，需要避免发送缓存大小寄存器（Sn_TXBUF_SIZE）的和超过 16。
16KB 的读取内存的分派方式与 16KB 的发送内存一样。16KB 的接收内存初始化被分配为每个 Socket 2KB 接收缓存区（2KB*8=16KB）。初始化分配的 2KB Socket 接收缓存，可以通过使用 Socket 接收缓存大小寄存器（Sn_XBUF_SIZE）重新分配。

一旦所有的 Socket 发缓存大小寄存器（Sn_TXBUF_SIZE）配置完成，16KB 的发送内存就会按照配置分配给每个 Socket 的发送缓存，并按照从 Socket 0 到 7 顺序分配。16KB 物理内存的地址是可以自增的。但是，为了避免数据传输错误，需要避免发送缓存大小寄存器（Sn_TXBUF_SIZE）的和超过 16。

对于 16 字节收/发内存的分配,请参考第 9.5.2 节 Sn_TXBUF_SIZE 和 Sn_RXBUF_SIZE 的相关描述。

16KB 的发送内存中分配了对应 Socket n 的发送缓存区，用于为来自主机传输的数据做缓存。Socket n 的发送缓存区。Socket n 发送缓存区的 16 位偏移地址支持 64KB 的寻址范围（从 0x0000 到 0xFFFF），关于他的配置请参考 'Socket n 发送写指针寄存器(Sn_TX_WR)'以及 Socket n 发送读指针寄存器(Sn_RX_WR)。然而，这 16 位偏移地址会自动转化为指定的 16KB 发送内存的物理地址，如图 17 所示。请参考 9.5.2 节中，关于 Sn_TX_WR & Sn_TX_RD 的介绍。

16KB 的接收内存中分配了对应 Socket n 的接收缓存区，用于为来自网络传输的数据做缓存。Socket n 的接收缓存区。Socket n 接收缓存区的 16 位偏移地址支持 64KB 的寻址范围（从 0x0000 到 0xFFFF），关于他的配置请参考 'Socket n 接受读指针寄存器(Sn_RX_RD)'以及 Socket n 接受写指针寄存器(Sn_RX_WR)。然而，这 16 位偏移地址会自动转化为指定的 16KB 接收内存的物理地址，如图 17 所示。请参考 9.5.2 节中，关于 Sn_RX_RD & Sn_RX_WR 的介绍。

9.5 寄存器描述

9.5.1 通用寄存器

MR (模式寄存 - 模式 Register) [R/W] [0x0000] [0x00][1]

该寄存器用于 S/W 复位，ping block 模式和 PPPoE 模式。

7	6	5	4	3	2	1	0
RST	保留	WOL	PB	PPPoE	保留	FARP	保留

位	符号	说明
7	RST	如果该位(bit)为'1'，内部寄存器将被初始化，它会在复位后自动清零。
6	Reserved	保留位
5	WOL	**网络唤醒 Wake on LAN** 0：关闭网络唤醒； 1：开启网络唤醒； 若启用网络唤醒，正常接收到 UDP 发来的 Magic Packet，会使中断(INTn)引脚拉低。当使用网络唤醒，无论任何源端口号都需要对 UDP Socket 端口开启。（具体请参考 Socket n 模式寄存器（Sn_MR）开启 Socket 部分）； 注意：TOE 支持的 Magic Packet 通过 UDP 传输，UDP 负载包括 6 字节的同步流（ '0xFFFF FFFF FFFF'）和 16 个目标 MAC 地址流。关于密码的设置将被忽略。你可以使用任何 UDP 源端口作为网络唤醒使用。
4	PB	**Ping Block 模式** 0: 关闭 Ping block; 1: 启用 Ping block; 如果该位(bit)设置为'1'，ping 请求时就没有响应。

[1] 注脚符号：【可读/写】【内存地址】【默认值】

3	PPPoE	**PPPoE 模式** 0：关闭 PPPoE 模式 1：启用 PPPoE 模式 如果你想使用 ADSL，改为需要设为 ‘1'。
2	Reserved	保留
1	FARP	**强迫 ARP 模式** 0：关闭强迫 ARP 模式； 1：启用强迫 ARP 模式； 在强迫 ARP 模式下，无论是否发送数据都会强迫发送 ARP 请求。
0	Reserved	保留

GAR (网关 IP 地址寄存器) [R/W] [0x0001 - 0x0004] [0x00]

该寄存器用来设置默认网关地址。

例如： “192.168.0.1”

0x0001	0x0002	0x0003	0x0004
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

SUBR (子网掩码寄存器) [R/W] [0x0005 - 0x0008] [0x00]

该寄存器用来设置子网掩码地址.

例如： “255.255.255.0”

0x0005	0x0006	0x0007	0x0008
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

SHAR (源 MAC 地址寄存器) [R/W] [0x0009 - 0x000E] [0x00]

该寄存器用来设置源 MAC 地址。

例如： “00.08.DC.01.02.03”

0x0009	0x000A	0x000B	0x000C	0x000D	0x000E
0x00	0x08	0xDC	0x01	0x02	0x03

SIPR (源 IP 地址寄存器) [R/W] [0x000F - 0x0012] [0x00]

该寄存器用来设置源 IP 地址。

例如： “192.168.0.2”

0x000F	0x0010	0x0011	0x0012
192 (0xC0)	168 (0xA8)	0 (0x00)	2 (0x02)

INTLEVEL (低电平中断定时器寄存器) [R/W] [0x0013 - 0x0014] [0x0000]

该寄存器用于设置中断生效等待的时间(IAWT)。当下一个中断触发，中断引脚将会在 INTLEVEL 时间后，拉低中断引脚（INTn）。

$$I_{AWT} = (INTLEVEL + 1) \times PLL_{CLK} \times 4 \text{ (when INTLEVEL } > 0)$$

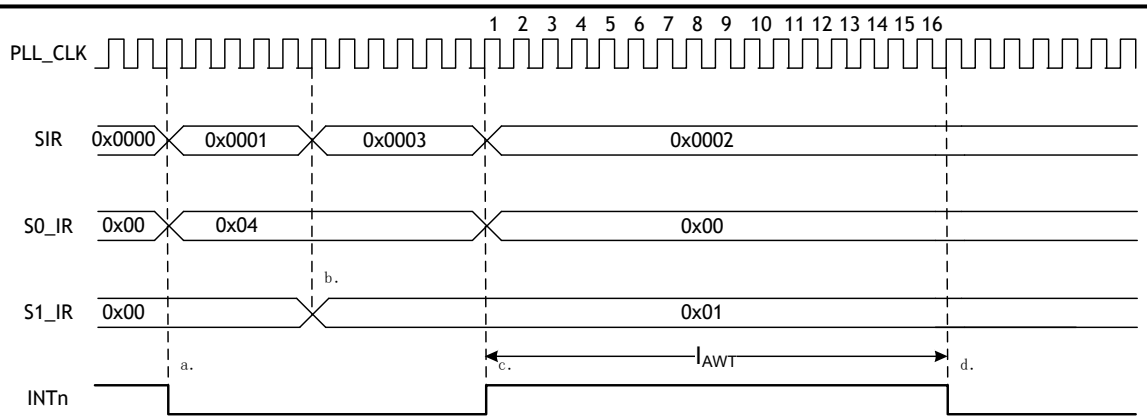


图 18 INTLEVEL 时序

1. 当 Socket 0 的超时中断被触发，S1_IR[0] & SIR[1]位被置 ‘1’，此时 INTn 引脚被拉低。
2. 当 Socket 1 的连接中断在前一个中断未处理完成之前被触发，S1_IR[0]& SIR[1]位被置 ‘1’，此时 INTn 引脚仍然为低。
3. 如果主机完全是通过清理 S0_IR[3]位来完成之前的中断，则 INTn 引脚拉高，但是 S1_IR[0] & SIR[1]仍然为 ‘1’。
4. 即 使 S1_IR[0] & SIR[1]位被设置为 ‘1’，但是在 INTLEVEL 期间，INTn 不能被拉低。只有过了 INTLEVEL 时间，INTn 才能被拉低。

IR (中断寄存器) [R/W] [0x0015] [0x00]

中断寄存器（IR）指明了中断的状态。某一位为 ‘1’时表示发生中断，主机对该位写入 ‘1’时，IR 的该位被清 ‘0’。如果 IR 不等于 ‘0x00’，INTn 引脚将会被拉低。直到其变为 ‘0x00’时，INTn 才会被拉高。

7	6	5	4	3	2	1	0
CONFLICT	UNREACH	PPPoE	MP	保留	保留	保留	保留

位	符号	说明
7	CONFLICT	**IP 冲突** 在收到 APR 请求时，发现发送方 IP 与本地 IP 重复，该位将被置 ‘1’
6	UNREACH	**目标不可抵达** 当接收到 ICMP（目的端口不可达）包后，该位置 ‘1’ 当该位为 ‘1’时，通过相应的 UIPR & UPORTR 可能查询到目标信息。 如：IP 地址和端口号。
5	PPPoE	**PPPoE 连接关闭** 当 PPPoE 模式下，PPPoE 连接断开时，该位生效
4	MP	**Magic Packet** 当网络唤醒模式启用并通过 UDP 接收到 Magic Packet 网络唤醒包时 该位生效
3~0	Reserved	保留位

IMR (中断屏蔽寄存器) [R/W][0x0016][0x00]

中断屏蔽寄存器（IMR）用来屏蔽中断源。每个中断屏蔽位对应中断寄存器（IR）中的一个位。当 IMR 的某一位为 ‘1’且 IR 对应的该位也为 ‘1’时中断发生。换而言之，当 IMR 中屏蔽位被清 ‘0’，即使对应的 IR 中断位置 ‘1’也不会产生中断。

7	6	5	4	3	2	1	0
IM_IR7	IM_IR6	IM_IR5	IM_IR4	保留	保留	保留	保留

位	符号	说明
7	IM_IR7	IP 冲突中断屏蔽 0：关闭 IP 冲突中断 1：启用 IP 冲突中断
6	IM_IR6	目的地址不能抵达中断屏蔽 0：关闭目的地址不能抵达 1：开启目的地址不能抵达
5	IM_IR5	PPPoE 关闭中断屏蔽 0：关闭 PPPoE 关闭中断 1：开启 PPPoE 关闭中断
4	IM_IR4	Magic Packet 中断屏蔽 0：关闭 Magic Packet 中断 1：开启 Magic Packet 中断
3:0	Reserved	保留位

SIR (Socket 中断寄存器) [R/W] [0x0017] [0x00]

SIR 指明了 Socket 的中断状态。SIR 被置 1 后将一直保持为 ‘1’，直到 Sn_IR 被主机写 ‘1’清零。如果 Sn_IR 不等于 ‘0x00’, 那么意味着 SIR 对应的第 n 位为 ‘1’。INTn 只有在 SIR 为 ‘0x00’时才能被拉低。

7	6	5	4	3	2	1	0
S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT

位	符号	说明
7:0	Sn_INT	当 Socket n 的中断触发，则 SIR 寄存器对应位变为 ‘1’

SIMR (Socket 中断屏蔽寄存器) [R/W] [0x0018] [0x00]

SIMR 寄存器中的每一位都对应 SIR 的相应位。当 SIMR 的一位为 ‘1’，而 SIR 的对应位为 ‘1’时，中断将被触发。换而言之，如果 SIMR 的一位为 ‘0’，那么即使 SIR 对应位为'1',中断将不会被触发。

7	6	5	4	3	2	1	0
S7_IMR	S6_IMR	S5_IMR	S4_IMR	S3_IMR	S2_IMR	S1_IMR	S0_IMR

位	符号	说明
7:0	Sn_IMR	**Socket n 中断屏蔽** 0：关闭 Socket n 中断 1：启用 Socket n 中断

RTR (重试时间值寄存器) [R/W] [0x0019 – 0x001A] [0x07D0]

RTR 配置了重传超时的时间值。每一单位数值为 100 微秒。初始化时值设为 2000（0x07D0），即相当于 200 毫秒（100us X 2000）。

在 RTR 配置的时间内，TOE 等待 Sn-CR(CONNECT，DISCON，CLOSE，SEND，SEND_MAC，SEND_KEEP command)传输后，来自对方的回应。如果在 RTR 时间段内没有回应，TOE 进行包重传或触发超时中断。

例如：当超时周期别设置为 400ms 时，RTR=(400ms/1ms)X10=4000(0x0FA0)

0x0019	0x001A
0x0F	0xA0

RCR (重试计数寄存器) [R/W] [0x001B] [0x08]

该寄存器是设置重新传送的次数。当第 ‘RCR+1'次重传时，超时中断就会置 ‘1'。（中断寄存器（Sn_IR）的 ‘中断'位(‘TIMEOUT'bit)设置为'1'）。

例如：RCR = 0x0007

0x001B
0x07

TOE 的超时可以用 RTR 和 RCR 来配置。TOE 的超时包括地址解析协议(ARP)和 TCP 重新传送超时。在 ARP 的重新传送超时（请参阅 RFC 826 http://www.ietf.org/rfc.html），TOE 会自动发 送 ARP 请求去对方(peer)的 IP 地址，从而获取 MAC 地址信息（IP、UDP 或 TCP 用于通信）。至 于等待对方(peer)的 ARP 响应方面，如在 RTR 中设置了重新传送时间时，对方(peer)的 ARP 没 有响应，超时发生和 ARP 将会请求重新传送。一直重复此步骤达'RCR+1'次。 如果在 ARP 重复请求重新传送次数达到'RCR+1'次时，仍然没有得到 ARP 响应，就会触发最终超时中断，Sn_IR(TIMEOUT)会变为'1'。

ARP 请求的最终超时值(ARPTO)如下:

$$ARP_{TO} = (RTR \times 0.1ms) \times (RCR + 1)$$

在配置了 RTR 和 RCR 期间，发生 TCP 数据包重传超时，TOE 就会发送 TCP 数据包(SYN、FIN、RST、数据包）和等待确认（ACK）。如果没有对方(peer)的 ACK 响应，就会触发一个临时超时中断且 TCP 数据包（较早前传送的）会重新传送。直到重新传送'RCR+1'次。即使 TCP 数据包重新传送'RCR+1'次，如果对方(peer) 仍 然 没 有 的 ACK 回 应 ， 就 会 触 发 最 终 超 时 中 断 且 同一 时 间 Sn_IR(TIMEOUT)='1'。

$$TCP_{TO} = \left(\sum_{N=0}^{M}(RTR \times 2^N) \quad + \quad ((RCR - M) \times RTR_{MAX})\right) \times 0.1ms$$

N : Retransmission count, 0≤N ≤ M

M : Minimum value when RTR x $2^{(M+1)}$ > 65535 and 0 ≤ M ≤ RCR

RTRMAX : RTR x 2^M

例如:

当 RTR = 2000(0x07D0),RCR = 8(0x0008),

ARPT0 = 2000 X 0.1ms X 9 = 1800ms = 1.8s

TCPT0 = (0x07D0+0x0FA0+0x1F40+0x3E80+0x7D00+0xFA00+0xFA00+0xFA00+0xFA00)X0.1ms

= (2000 + 4000 + 8000 + 16000 + 32000 + ((8 - 4) X 64000)) X 0.1ms

= 318000 X 0.1ms = 31.8s

PTIMER (PPP 连接控制协议请求定时寄存器) [R/W] [0x001C] [0x0028]

PTIMER 设置发送 LCP Echo 请求的时间，单位时间是 25 毫秒(ms)。

例如：如果 PTIMER 是 200，200 * 25(ms) = 5000(ms) = 5 seconds

PMAGIC (PPP 连接控制协议幻数寄存器) [R/W] [0x001D] [0x00]

该寄存器配置了用于 LCP 回应请求的 4 字节幻数(Magic number)。

例如：PMAGIC = 0x01，LCP Magic number = 0x01010101

0x001D

0x01

PHAR (PPPoE 模式下目标 MAC 寄存器)[R/W] [0x001E-0x0023] [0x0000]

PHAR 需要在 PPPoE 连接过程中写入 PPPoE 服务器所需的 MAC 地址。

例如：目标 MAC 地址为 00:08:DC:12:34:56

0x001E	0x001F	0x0020	0x0021	0x0022	0x0023
0x00	0x08	0xDC	0x12	0x34	0x56

PSID (PPPoE 模式下会话 ID 寄存器) [R/W] [0x0024-0x0025] [0x0000]

PSID 需要填入 PPPoE 连接过程中需要的 PPPoE 服务器会话 ID。

例如：会话 ID 为 0x1234

0x0024	0025
18 (0x12)	52(0x34)

PMRU (PPPoE 模式下最大接收单元) [R/W] [0x0026-0x0027] [0xFFFF]

PMRU 规定了 PPPoE 模式下的最大接收单元。

例如：PPPoE 模式下最大接收单元为 0x1234

0x0026	0027
18 (0x12)	52 (0x34)

UIPR (无法抵达 IP 地址寄存器) [R] [0x0028-0x002B] [0x00000000]

UPORTR (无法抵达端口寄存器) [R] [0x002C-0x002D] [0x0000]

当 TOE 发送数据给一个未开启或不可抵达的端口号时，接收到一个 ICMP 包（目的地址无法抵达），IR 变为 '1'，UIPR & UPORTR 会分别记录下目的 IP 地址和端口号。

例如： "192.168.0.11"

0x0028	0x0029	0x002A	0x002B
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0E)

例如： "0x1234"

0x002C	002D
18 (0x12)	52(0x34)

PHYCFGR (TOE PHY Configuration Register) [R/W] [0x002E] [0b10111XXX]

PHYCFGR 配置 PHY 的工作模式及 PHY 重启。此外，PHYCFGR 指明了 PHY 的状态，比如：全/办双工，速度，链接状态。

位	符号	说明
7	RST	**重启 Reset [R/W]** 当该位为 '0'时，内部 PHY 重启。 在 PHY 重启后，该位须设置为 '1'。
6	OPMD	**配置 PHY 工作模式** 1：通过 PHYCFGR 的 OPMDC[2:0]位配置； 0：通过硬件引脚配置(PMODE[2:0])；

<table>
<tr><td></td><td></td><td>该位通过 OPMDC[2:0]位或者 PMODE[2:0]引脚配置 PHY 的工作模式。
当 TOE 通过 POR 或 RSTn 引脚重启，则 PHY 的工作模式默认为通过
PMODE[2:0]引脚配置。在 POR 或 RSTn 重启后，用户仍然可以重新设置
使用 OPMDC[2:0]位来配置 PHY 工作模式。如果用户想重新设置通过
OPMDC[2:0]位来配置 PHY 工作模式。就需要用户在 PMODE[2:0]
引脚配置模式下将该位设置为 ‘1’之后，并将 RST 位置 ‘0 ‘重启 PHY。</td></tr>
<tr><td>5:3</td><td>OPMDC</td><td>工作模式配置位[R/W]
这些位选定 PHY 的工作模式如下表所示:
<table>
<tr><th>5</th><th>4</th><th>3</th><th>说明</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>10BT 半双工，关闭自动协商</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>10BT 全双工，关闭自动协商</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>100BT 半双工，关闭自动协商</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>100BT 全双工，关闭自动协商</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>100BT 半双工，启用自动协商</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>未启用</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>掉电模式</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>全功能，启用自动协商</td></tr>
</table></td></tr>
<tr><td>2</td><td>DPX</td><td>双工工作状态【只读】
1：全双工
0：半双工</td></tr>
<tr><td>1</td><td>SPD</td><td>速度状态【只读】
1: 100Mpbs based
0: 10Mpbs based</td></tr>
<tr><td>0</td><td>LNK</td><td>连接状态 [只读]
1: 已连接
0: 连接断开</td></tr>
</table>

VERSIONR (TOE 芯片版本寄存器) [R] [0x0039] [0x04]

TOE 的版本寄存器默认为 0x04。

9.5.2 Socket 寄存器

S*n*[2]_MR (Socket n 模式寄存器) [R/W] [0x0000] [0x00]

该寄存器用于配置所有 SOCKET 的选项或协议类型。

7	6	5	4	3	2	1	0
MULTI/ MFEN	BCASTB	ND / MC /MMB	UCASTB MIP6B	P3	P2	P1	P0

位	符号	说明
7	**MULTI/ MFEN**	**UDP 多播模式** 0：关闭多播 1：开启多播 该位只有当 UDP 模式(P[3:0] = ‘0010')，才能生效。 使用多播模式，需要 Sn_DIPR & Sn_DPORT 在 Socket n 通过 Sn_CR 的打开 配置命令打开之前，分开配置组播 IP 地址及端口号。 在 MACRAW 模式下开启 MAC 地址过滤 0 : 关闭 MAC Filtering

[2] *是 Socket 编号（0，1，2，3，4，5，6，7）。n 设置了 SNUM[2:0]控制位集。*

<table>
<tr><td></td><td></td><td>1：启用 MAC Filtering
0：关闭 MAC 地址过滤
1：启用 MAC 地址过滤
该位只有在 MACRAW(P[3:0] = ‘0100')模式期间，才能生效。如果设置为
‘1',TOE 只接受广播包以及发送给他本身的包。当该位设置为 ‘0',
TOE 可以接收到来自网络的所有包。如果用户希望实现混合 TCP、IP 协议
栈，建议设置该位为 ‘1'，以降低主机处理所有接收包的负载。</td></tr>
<tr><td>6</td><td>BCASTB</td><td>MACRAW 和 UDP 模式下的网络阻塞
0：关闭广播阻塞
1：开启广播阻塞
该位在 UDP 模式(P[3:0] = ‘0010')可以屏蔽接收广播包。
此外，该位在 MACRAW 模式下(P[3:0] = ‘0100')同样生效。</td></tr>
<tr><td>5</td><td>ND/MC/MMB</td><td>使用无延时 ACK
0：关闭无延时 ACK 选项
1：开启无延时 ACK 选项
该位只有在 TCP 模式下(P[3:0] = ‘0001')才能生效。
当该位设置为 ‘1'时，TOE 会在从对端接收到数据包后没有任何延时尽快地回复 ACK 包。当该位为
‘0'，TOE 发送 ACK 包需要 RTR 设定的超时时间做延时。

多播
0：使用 IGMP 版本 2
1：使用 IGMP 版本 1
该位只在 UDP 模式(P[3:0] = ‘0010')，且 MULTI= ‘1'时生效。
他配置了 IGMP 消息的版本（加入/离开/报告）。

MACRAW 模式下的多播阻塞
0：关闭多播阻塞
1：开启多播阻塞
该位只有在 MACRAW(P[3:0] = ‘0100')模式下才能生效。他可以屏蔽多播
MAC 地址的包传输。</td></tr>
<tr><td>4</td><td>UCASTB
MIP6B</td><td>UDP 模式下的单波阻塞
0： 关闭单播阻塞
1： 开启单播阻塞
在 UDP 模式(P[3:0] = ‘0010')和 MULTI= ‘1'的情况下，该位屏蔽接收单播数据包;

MACRAW 模式下，IPv6 包阻塞
0：关闭 IPv6 包阻塞
1：开启 IPv6 包阻塞
该位只有在 MACRAW 模式下才能生效(P[3:0]= ‘0100')。它将屏蔽接收 IPv6
包。</td></tr>
<tr><td>3</td><td>P3</td><td rowspan="4">协议 Protocol
该位配置 Socket n 使用的协议模式
<table><tr><th>P3</th><th>P2</th><th>P1</th><th>P0</th><th>含义</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Closed</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>TCP</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>UDP</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>MACRAW</td></tr></table>* MACRAW 只有在 Socket 0 下才可用</td></tr>
<tr><td>2</td><td>P2</td></tr>
<tr><td>1</td><td>P1</td></tr>
<tr><td>0</td><td>P0</td></tr>
</table>

Sn_CR (Socket n 配置寄存器) [R/W] [0x0001] [0x00]

值	符号	说明
0x01	OPEN	按照 Sn_MR(P3:P0)的协议选择来初始化和打开(open) Socket n。

<table>
<tr><td></td><td></td><td>下表显示了 Sn_SR 和 Sn_MR 的对应值。<table><tr><th>Sn_MR (P[3:0])</th><th>Sn_SR</th></tr><tr><td>Sn_MR_CLOSE (‘0000')</td><td>-</td></tr><tr><td>Sn_MR_TCP (‘0001')</td><td>SOCK_INIT (0x13)</td></tr><tr><td>Sn_MR_UDP (‘0010')</td><td>SOCK_UDP (0x22)</td></tr><tr><td>S0_MR_MACRAW (‘0100')</td><td>SOCK_MACRAW (0x02)</td></tr></table></td></tr>
<tr><td>0x02</td><td>LISTEN</td><td>该位只在 TCP 模式(Sn_MR(P3:P0) = Sn_MR_TCP)下生效。
在这种模式下，Socket n 被配置为一个 TCP 服务器,它是等待 “TCP 客户端” 的连接请求（SYN 数据包）。该 Sn_SR 寄存器由 SOCK_INIT 改变为SOCK_LISTEN。
当一个 TCP 客户端的连接请求成功后，该 Sn_SR 寄存器由 SOCK_LISTEN改变为 SOCK_ESTABLIESHE，与此同时 Sn_IR(0) 会变为'1'。另一方面，当连接失败时，Sn_IR(3)被设置为'1'，Sn_SR 改变为 SOCK_CLOSED。</td></tr>
<tr><td>0x04</td><td>CONNECT</td><td>此模式只适用于 TCP 模式且运行 Socket n 作为 TCP 客户端。通过与存储在目的地址寄存器（Sn_DIPR）和端口号寄存器（Sn_DPORT）中的 IP地址和端口号进行连接，一个连接请求被发送到 TCP 服务器。
当一个客户端的连接请求成功后，Sn_SR寄存器改为 SOCK_ESTABLIESHED，Sn_IR(0)会变为'1'。
以下三种情况意味着连接请求失败:
ARPTO发生超时(Sn_IR(s)= ‘1')。因为目的地的 MAC 地址不能通过ARP 过程中获取。
当没有收到 SYN/ACK 数据包，而引起 TCPT0(Sn_IR(3))被设置为'1 时。
当 RST 数据包而不是 SYN/ACK 数据包被接收时。
以上三种情况下，Sn_SR 会改为 SOCK_CLOSED。</td></tr>
<tr><td>0x08</td><td>DISCON</td><td>只在 TCP 模式下有效;
不论 “TCP 服务器” 或 “TCP 客户端” ，都使用 DISCON 断开。
主动关闭: 它传输断开请求（FIN 数据包）到所连接的对方(peer) 。
被动关闭: 当从对方(peer)收到 FIN 数据包时，回复一个 FIN 数据包到对方(peer) 。
当成功的执行断开连接操作（FIN/ACK 数据包被接收）时，Sn_SR 改为 SOCK_CLOSED。
当断开请求没有收到 ACK 时，TCPT0 超时中断就会发生(Sn_IR(3)= ‘1')，Sn_SR 改为 SOCK_CLOSED。
比照>如果用 CLOSE 命令来代替 disconnect 命令的话，意味着直接将 Sn_SR 变为 SOCK_CLOSED，将不再执行 FIN/ACK 这种断开机制，从而没有断开连接过程。如果当通讯期间从一个对方(peer)接收到一个 RST数据包，Sn_SR 将无条件地更改为 SOCK_CLOSED。</td></tr>
<tr><td>0x10</td><td>CLOSE</td><td>关闭 Socket n。
Sn_SR 改为 SOCK_CLOSED。</td></tr>
<tr><td>0x20</td><td>SEND</td><td>发送(SEND)Socket n 发送(TX)内存中的所有缓冲数据。
欲想了解更多详情，请参阅 Socket n 发送(TX)自由尺寸寄存器 (Sn_TX_FSR)，Socket n 发送(TX)写指针寄存器(Sn_TX_WR)和 Socket n 发送(TX)读指针寄存器 (Sn_TX_RD)。</td></tr>
<tr><td>0x21</td><td>SEND_MAC</td><td>只在 UDP 模式下有效;
基本操作是与发送(SEND)相同的。一般来说，发送(SEND)数据通常需要先通过自动 ARP（地址解析协议）请求获得目的地 MAC 地址才能进行传输。而 SEND_MAC 却不需要使用自动 ARP 请求。此时，目的 MAC地址使用的是主机 Sn_DHAR 的设置，而不是通过 ARP 过程获取的。</td></tr>
</table>

0x22	SEND_KEEP	只在 TCP 模式下有效; 通过发送 1 字节在线心跳包来检查连接状态。 如果对方不能在超时计数期内反馈在线心跳包，这个连接将会被关闭并触发超时中断。
0x40	RECV	RECV 命令通过使用接收读指针寄存器(Sn_RX_RD)来完成在 Socket n 接收缓存中接收数据的过程。 详情参考，Socket n 接收读大小寄存器(Sn_RX_RSR)，Socket n 接收写指针寄存器 (Sn_RX_WR) 以及 Socket n 接收读指针寄存器 (Sn_RX_RD)。

该寄存器用于设置 Socket n 的配置命令如 OPEN 、CLOSE、CONNECT、LISTEN、END 和 RECEIVE。经 TOE 识别这一命令后，Sn_CR 寄存器会自动清零为 0x00。尽管 Sn_CR 被清零为 0x00，但命令仍在处理中。为了验证该命令是否完成，请检查 Sn_IR 或 Sn_SR 寄存器。

S*n*_IR (Socket n 中断寄存器) [RCW1] [0x0002] [0x00]

Sn_IR 寄存器用于提供给 Socket n 中断类型信息，如建立(Establishment)、终止(Termination)、接收数据(Receiving data)和超时(Timeout)。当中断触发并且 Sn_IMR 的对应位是'1'的时候，Sn_IR 的对应位也将会被置 ‘1'。

如果想把 Sn_IR 位清零的话，主机应该将该位置 ‘1'。

7	6	5	4	3	2	1	0
保留	保留	保留	SEND_OK	TIMEOUT	RECV	DISCON	CON

位	符号	说明
7~5	Reserved	保留位
4	SEND_OK	**Sn_IR(SENDOK) 中断** 当 SEND 命令完成时，此位生效
3	TIMEOUT	**Sn_IR(TIMEOUT) 中断** 当 ARPTO 或者 TCPT0 超时被触发时，此位生效。
2	RECV	**Sn_IR(RECV) 中断** 无论何时，只要接收到了对方数据，此位生效。
1	DISCON	**Sn_IR(DISCON) 中断** 当接收到对方的 FIN or FIN/ACK 包时，此位生效。
0	CON	**Sn_IR(CON) 中断** 当成功与对方建立连接，且 Sn_SR 变为 SOCK_ESTABLISHED 状态时，此位生效。

S*n*_SR (Socket n 状态寄存器) [R] [0x0003] [0x00]

Sn_SR 指示了 Socket n 的状态，并根据 Sn_CR 或者一些 TCP 模式下的特殊控制包，如 SYN，FIN 包而改变。

值	符号	说明
0x00	SOCK_CLOSED	该位指示了 Socket n 处于关闭状态，资源被释放。 当 DICON, CLOSE 命令生效或当触发超时中断时，TOE 对应的 Socket n 会无视之前的状态，变为 SOCK_CLOSED。
0x13	SOCK_INIT	该位指示了 Socket n 端口打开并处于 TCP 工作模式。 当 Sn_MR (P[3:0]) = ‘0001'且 OPEN 命令生效时，Sn_SR 变为 SOCK_INIT。之后，用户才可以使用 LISTEN 或 CONNECT 命令。

0x14	**SOCK_LISTEN**	此位指示着 Socket n 工作在 TCP 服务器模式下， 且等待对方（TCP 客户端）的连接请求（SYN Packet）。 当连接请求被成功接收以后，Socket_SR 会变为 SOCK_ESTALBLISHED 状态。否则将会在触发 TCPT0 超时中断之后，变为 SOCK_CLOSED 状态。
0x17	**SOCK_ESTABLISHED**	指示了 Socket n 的连接状态。 SOCK_LISTEN 状态下，当 TCP 服务器处理 TCP 客户端的 SYN 请求包或当 CONNECT 命令配置成功时，变为 SOCK_ESTABLISHED。 在此状态下，可以使用 SEND 或者 RECV 命令进行数据包 传输。
0x1C	**SOCK_CLOSE_WAIT**	指示了 Socket n 接收到了来自连接对方发来的断开连接请求 (FIN packet)。这是一个半关闭状态，可以进行数据传输。 若要全部关闭，需要使用 DISCON 命令。而如果是要关闭 Socket，需要使用 CLOSE 命令。
0x22	**SOCK_UDP**	指示了 Socket n 处于 UDP 模式下 (Sn_MR(P[3:0]) = ‘0010')。 当 Sn_MR(P[3:0]) = ‘0010'且 OPEN 命令生效时，Sn_SR 改变为 SOCK_UDP。 不同于 TCP 模式，在这个模式下，数据包可以在无连接过程 的情况下传输。
0x42	**SOCK_MACRAW**	指示了 Socket 0 工作在 MACRAW 模式下(S0_MR(P[3:0]) = ‘0100')。MACRAW 模式仅在 Socket 0 下生效。 当 S0_MR(P[3:0] = ‘0100'且 OPEN 命令生效时，Sn_SR 改变为 SOCK_MACRAW。 如 UDP 模式一样，Socket 0 工作在 MACRAW 模式下时，也 能在无连接过程的情况下，实现 MAC 数据包(以太网帧) 传输。

下表显示了 Socket n 状态改变时的临时状态。

值	符号	说明
0x15	SOCK_SYNSENT	指示了 Socket n 已经发送连接请求（SYN Packet）到对方。 他显示了发送 CONNECT 命令后，Sn_SR 从 SOCK_INIT 到 SOCK_ESTABLISHED 的临时状态。 如果此时，收到了来自对方的接受连接请求（SYN/ACKpac ket）则，变为 SOCK_ESTABLISHED。 否则，在 TCPT0 超时(Sn_IR[TIMEOUT] = ‘1')中断之后， 转变为 SOCK_CLOSED。
0x16	SOCK_SYNRECV	指示 Socket n 成功的从对方收到了连接请求包 (SYN packet)。 如果 Socket n 成功的给对方发送了连接应答 (SYN /ACK packet)，将转变为 SOCK_ESTABLISHED 状态。 否则，在触发超时中断(Sn_IR[TIMEOUT] = ‘1')后， 变为 SOCK_CLOSED。
0x18	SOCK_FIN_WAIT	这些状况表示 SOCKET n 正在关闭。 这显示的是断开连接（主动关闭或被动关闭）的过程。 当断线程序成功完成或 TCPT0(Sn_IR(超时)='1')发生时， 它便会更改为 SOCK_CLOSED。
0x1A	SOCK_CLOSING	
0X1B	SOCK_TIME_WAIT	

0X1D	SOCK_LAST_ACK	指示了 Socket n 在被动关闭状态下， 正在等待对断开连接请求(FIN packet)做出回应(FIN/ACK packet)。 当 Socket n 成功接收到了断开连接请求的回应或出发超时中断，则变为 SOCK_CLOSED 状态。

S*n*_PORT (Socket n 源端口寄存器) [R/W] [0x0004-0x0005] [0x0000]

该寄存器配置了 Socket n 的源端口号。当 Socket n 工作在 TCP 或 UDP 模式下，该寄存器生效。

注意：必须在 OPEN 命令生效前，完成对该寄存器的设置。

例如：SOCKET 0 的端口=5000(0x1388)，配置应如下：

0x0004	0x0005
0x13	0x88

S*n*_DHAR (Socket n 目的 MAC 地址寄存器) [R/W] [0x0006-0x000B] [0xFFFFFFFFFFFF]

Sn_DHAR 寄存器指示的为：UDP 模式下，使用 Send_MAC 配置命令，配置 Socket n 的目标主机 MAC 地址；或者 CONNECT/SEND 配置命令，ARP 过程获取到的 MAC 地址。

例如：Socket 0 的目标 MAC 地址 = 08.DC.00.01.02.10，配置应如下：

0x0006	0x0007	0x0008	0x0009	0x000A	0x000B
0x08	0xDC	0x00	0x01	0x02	0x0A

S*n*_DIPR (Socket n 目标 IP 地址寄存器) [R/W] [0x000C-0x000F] [0x00000000]

Sn_DIPR 配置或指示的为 Socket n 的目标主机 IP 地址，在 TCP/UDP 模式下生效。

在 TCP 客户端模式下，在 CONNECT 配置命令前，该寄存器设置了 TCP 服务器的 IP 地址。

在 TCP 服务器模式下，他显示了在成功建立连接后，TCP 客户端的 IP 地址；在 UDP 模式下，他配置了对方主机的 IP 地址以供 SEND 或 SEND_MAC 配置命令后接收 UDP 包。

例如：Socket 0 的目标 IP 地址= 192.168.0.11， 配置应如下：

0x000C	0x000D	0x000E	0x000F
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

S*n*_DPORT (Socket n 目标端口寄存器) [R/W] [0x0010-0x0011] [0x00]

Sn_DPORT 配置或指示了 Socket n 的目标主机端口号，在 TCP/UDP 模式下生效。

在 TCP 客户端模式下，在 CONNET 配置命令前，该寄存器配置了 TCP Server 监听的端口号。

在 TCP 服务器模式下，他显示了在成功建立连接后，TCP 客户端的端口号；

在 UDP 模式下，他配置了对方主机的端口号以供 SEND 或 SEND_MAC 配置命令后接收 UDP 包。

例如：Socket 0 的目标端口号 = 5000(0x1388) ，配置应如下：

0x0010	0x0011
0x13	0x88

S*n*_MSSR (Socket n 最大分段寄存器) [R/W] [0x0012-0x0013] [0x0000]

该寄存器配置或显示了 Socket n 的最大传输单元 MTU(Maximum Transfer Unit)。

在 TCP/UDP 模式下，默认该寄存器设定的最大传输单元生效。

然而，在 PPPoE 模式下(MR[PPPoE] = ‘1’)，该寄存器将取决于 PPPoE 的最大传输单元。

Mode	Normal (MR(PPPoE)= ‘0 ‘)		PPPoE (MR(PPPoE)= ‘1 ‘)	
	Default MTU	Range	Default MTU	Range
TCP	1460	1 ~ 1460	1452	1 ~ 1452
UDP	1472	1 ~ 1472	1464	1 ~ 1464
MACRAW	1514			

当 Socket n 处于 MACRAW 模式时，由于 MTU 不在内部处理，默认的 MTU 将会生效，因此，当传输的数据比默认的 MTU 大时，主机需要手动的将数据划分成默认 MTU 大小单元进行传输。

当 Socket n 处于 TCP/UDP 模式，而传输的数据比 MTU 大时，数据将会被自动的划分成默认 MTU 单元大小传输。

在 UDP 模式下，由于不像 TCP 模式那样涉及到一些连接过程，所以使用了 MTU 配置。当不同大小的 MTU 数据传输给对方是时，可能会收到 ICMP 包（MTU 分片）。这样的话 IR(FMTU)置 ‘1', 对方的信息如 MTU 大小以及 IP 地址将分别由 FMTUR 和 UIPR 指定。如果 IR[MTU] = ‘1', 用户不能发送数据到对方。如果要重新恢复与对方的通讯，可以按照以下操作:

1、通过 CLOSED 配置命令关闭 Socket。

2、设置 Sn_MSS 指定 FMTUR 中的 MTU。

3、通过 OPEN 配置命令打开 Socket n。

4、重新与对方通信。

例如：Socket 0 的 MSS = 1460(0x05B4)， 配置应如下:

0x0012	0x0013
0x05	0xB4

S*n*_TOS (Socket n IP 服务类型寄存器) [R/W] [0x0015] [0x00]

该寄存器设置在 IP 层里 IP header 的 TOS(Type of Service - 服务类型)字段。它应在执行 OPEN 命令之前设置。

请参考 http://www.iana.org/assignments/ip-parameters.

S*n*_TTL (Socket n 生存时间寄存器) [R/W] [0x0016] [0x80]

该寄存器设置在 IP 层里 IP 头的 TTL(Time-To-Live - 生存时间) 字段。它应在执行 OPEN 命令之前设置。

请参考 http://www.iana.org/assignments/ip-parameters.

S*n*_RXBUF_SIZE (Socket n 接收缓存大小寄存器) [R/W] [0x001E] [0x02]

Sn_RXBUF_SIZE 配置了 Socket n 的接收缓存大小。Socket n 接收缓存区大小可以配置为 1，2，4，8 和 16Kbytes。如果配置为其他大小，则 TOE 不能正常的从对方主机接收数据。

即使 Socket n 的接收缓存大小初始默认为 2Kbytes。用户仍然可以使用 Sn_RXBUF_SIZE 重新定义。但是所有 Socket 接收缓存（Sn_RXBUF_SIZE）的总大小不能超过 16Kbytes。否则，将会使得接收异常。

当所有的 Sn_RXBUF_SIZE 配置完成后，就会按照 Socket 0 到 7 的顺序依次将 16Kbytes 的接收内存分配给各个 Socket 作为接收缓存使用。

不论 Socket n 的接收缓存配置的大小如何，都可以被 16 位的偏移地址寻址找到。（寻址范围: 0x0000 到 0xFFFF）

Value (dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

例如：Socket 0 RX Buffer Size = 8KB

0x001E
0x08

Sn_TXBUF_SIZE (Socket n 发送缓存大小寄存器) [R/W] [0x001F] [0x02]

Sn_TXBUF_SIZE 配置了 Socket n 的发送缓存大小。Socket n 发送缓存区大小可以配置为 1，2，4，8 和 16Kbytes。如果配置为其他大小，则 TOE 不能正常给对方主机发送数据。

即使 Socket n 的发送缓存大小初始默认为 2Kbytes。用户仍然可以使用 Sn_RXBUF_SIZE 重新定义。但是所有 Socket 发送缓存的总大小不能超过 16Kbytes。否则，将会使得发送异常。

当所有的 Sn_TXBUF_SIZE 配置完成后，就会按照 Socket 0 到 7 的顺序依次将 16Kbytes 的发送内存分配给各个 Socket 作为发送缓存使用。

不论 Socket n 的接发送存配置的大小如何，都可以被 16 位的偏移地址寻址找到。（寻址范围：0x0000 到 0xFFFF）

Value (dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

例如：Socket 0 TX Buffer Size = 4KB

0x001F
0x04

Sn_TX_FSR (Socket n 空闲发送缓存寄存器) [R] [0x0020-0x0021] [0x0800]

Sn_TX_FSR 显示了 Socket n 发送缓存的空闲空间大小。该寄存器初始化配置为 Sn_TXBUF_SIZE 大小。如果要发送的数据长度大于 Sn_Tx_FSR 的值，在发送缓存中还没被发送的数据将会被覆盖，因此数据长度大于 Sn_Tx_FSR 的值时，TOE 将不允许写入发送缓存。因此在向 Socket n 发送缓保存数据之前，需要先检查一下数据大小是否等于或小于其剩余空间，然后再保存数据到发送缓存并通过 SEND/SEND_MAC 配置命令发送。如果数据比检查到的剩余空间大，需要将数据划分成小于或等于剩余空间的大小之后，再保存数据到 Socket n 发送缓存。如果 Sn_MR(P[3:0])不是 TCP 模式(‘0001')，TOE 将计算发送写指针(Sn_TX_WR)和 Socket n 发送读指针之间的空间，并自动将数据划分成相应大小。

如果 Sn_MR(P[3:0])是 TCP 模式(‘0001')，TOE 将计算发送写指针(Sn_TX_WR)与内部 ACK 指针(指示已经从连接对方接收数据的节点位置)之间的空间。

例如：2048(0x0800) 在 S0_TX_FSR 时

0x0020	0x0021
0x08	0x00

Sn_TX_RD (Socket n 发送读指针寄存器) [R] [0x0022-0x0023] [0x0000]

Sn_TX_RD 寄存器可以通过 OPEN 配置命令进行初始化。然而，如果 Sn_MR(P[3:0])是 TCP 模式(‘0001')，该寄存器将会在 TCP 连接期间，重新进行初始化。

该寄存器初始化之后，会根据 SEND 配置命令自增。SEND 配置命令传输的是 Socket n 发送缓存中，当前 Sn_TX_RD 到 Sn_TX_WR 之间保存的数据。在传输完保存的数据之后，SEND 配置命令会使得 Sn_TX_RD 等于 Sn_TX_WR。当 Sn_TX_RD 增加的值超出最大值 0xFFFF（大于 0x10000 并产生进位），Sn_TX_RD 会忽略进位，并且自动保存为低 16 位的值。

Sn_TX_WR (Socket n 发送写指针寄存器) [R/W] [0x0024-0x0025] [0x0000]

Sn_TX_WD 寄存器可以通过 OPEN 配置命令进行初始化。然而，如果 Sn_MR(P[3:0])是 TCP 模式(‘0001')，该寄存器将会在 TCP 连接期间，重新进行初始化。

该寄存器需要读取或更新如下：

1、读取发送缓存中将要保存传输数据的首地址。

2、从 Socket n 的发送缓存对应的首地址开始，保存需要传输的数据。

3、在保存完传输数据之后，将 Sn_TX_WR 的值增加到传输数据大小。如果增加后，超过最大值 0xFFFF（比 0x10000 大且产生进位），那么将自动忽略进位，并自动更新为低 16 位的值。

4、通过使用 SEND 命令发送保存在 Socket n 发送缓存中的数据。

Sn_RX_RSR (Socket n 空闲接收缓存寄存器) [R] [0x0026-0x0027] [0x0000]

Sn_RX_RSR 显示了 Socket n 接收缓存中已接收和保存的数据大小。

Sn_RX_RSR 不会超过 n_RXBUF_SIZE 大小，且计算的为 Socket n 接收写指针(Sn_RX_WR)和 Socket n 接收读指针之间的空间大小。

例如：2048(0x0800) 在 S0_RX_RSR 时

0x0026	0x0027
0x08	0x00

Sn_RX_RD (Socket n 接收读指针寄存器) [R/W] [0x0028-0x0029] [0x0000]

Sn_RX_RD 寄存器可以通过 OPEN 配置命令进行初始化。请确保该寄存器按照以下步骤读取并更新：

1、读取保存在接收缓存中数据的首地址。

2、从保存在 Socket n 接收缓存中数据的首地址开始读取数据。

3、在读取完毕接收数据，将 Sn_RX_RD 的值更新为所读数据大小。如果增加后的值超过最大值 0xFFFF，即超过 0x10000 并产生进位，将会忽略进位，只取低 16 位值。

4、在接收到 RECV 命令后，将更新后的 Sn_RX_RD 值告知 TOE。

例如：2048(0x0800) 在 S0_RX_RD 时

0x0028	0x0029
0x08	0x00

Sn_RX_WR (Socket n 接收写指针寄存器) [R] [0x002A-0x002B] [0x0000]

Sn_RX_WR 寄存器可以通过 OPEN 配置命令进行初始化。并且随着数据接收自动增加。

如果 Sn_RX_WR 的值增长到超过最大值 0xFFFF（即超过 0x10000 并产生进位），那么将自动忽略进位，并自动更新为低 16 位的值。

例如：2048(0x0800)在 S0_RX_WR 时

0x002A	0x002B
0x08	0x00

Sn_IMR (Socket n 中断屏蔽寄存器) [R/W] [0x002C] [0xFF]

Sn_IMR 负责屏蔽 Socket n 的中断。每一位都对应了 Sn_IR 寄存器的相应位。Socket n 的中断触发并且 Sn_IMR 的对应位为 ‘1'时，Sn_IR 的对应位变为 ‘1'。如果 Sn_IMR 和 Sn_IR 的对应位均为 ‘1'且 IR 寄存器的相应为 ‘1'，INTn 引脚便会拉低并使主机产生中断。

7	6	5	4	3	2	1	0
保留	保留	保留	SEND_OK	TIMEOUT	RECV	DISCON	CON

位	符号	说明
7:5	Reserved	保留位
4	SENDOK	Sn_IR(SENDOK) 中断屏蔽
3	TIMEOUT	Sn_IR(TIMEOUT) 中断屏蔽
2	RECV	Sn_IR(RECV) 中断屏蔽
1	DISCON	Sn_IR(DISCON) 中断屏蔽
0	CON	Sn_IR(CON) 中断屏蔽

Sn_FRAG (Socket n 分段寄存器) [R/W] [0x002D-0x002E] [0x4000]

它设置了 IP 层中 IP 报头的分段字段。

例如：Sn_FRAG0 = 0x4000 (不要分段)

0x002D	0x002E
0x00	0x00

Sn_KPALVTR (Socket n 在线时间寄存器) [R/W] [0x002F] [0x00]

Sn_KPALVTR 配置了 SOCKET n 的 ‘KEEP ALIVE(KA)'在线验证心跳包传输时间。他只在 TCP 模式下生效，在其他模式下将会被忽略。单位时间为 5 秒。

KA 包会在 Sn_SR 变为 SOCK_ESTABLISHED 且与对方至少进行过一次收或发的通讯后进行传输。

如果 ‘Sn_KPALVTR > 0'，TOE 在一定时间周期会自动传输 KA 包以检查 TCP 的连接状态（自动在线验证）。如果 ‘Sn_KPALVTR = 0'，将不会启动自动在线验证，主机可以通过 SEND_KEEP 配置命令发送 KA 包（手动在线验证）。在 ‘Sn_KPALVTR > 0'时，将会无视手动在线验证。

例如：Sn_KPALVTR = 10（会每 50 秒自动发送一次在线验证包）

0x002F
0x0A

10 DMA 控制器(DMA)

10.1 DMA 简介

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须 CPU 干预，数据可以通过 DMA 快速地移动，这就节省了 CPU 的资源来做其他操作。
两个 DMA 控制器有 12 个通道(DMA1 有 7 个通道，DMA2 有 5 个通道)，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

10.2 DMA 主要特性

- 12 个独立的可配置的通道(请求)：DMA1 有 7 个通道，DMA2 有 5 个通道
- 每个通道都直接连接专用的硬件 DMA 请求，每个通道都同样支持软件触发。这些功能通过软件来配置。
- 在同一个 DMA 模块上，多个请求间的优先权可以通过软件编程设置(共有四级：很高、高、中等和低)，优先权设置相等时由硬件决定(请求 0 优先于请求 1，依此类推)。
- 独立数据源和目标数据区的传输宽度(字节、半字、全字)，模拟打包和拆包的过程。源和目标地址必须按数据传输宽度对齐。
- 支持循环的缓冲器管理
- 每个通道都有 3 个事件标志(DMA 半传输、DMA 传输完成和 DMA 传输出错)，这 3 个事件标志逻辑或成为一个单独的中断请求。
- 存储器和存储器间的传输
- 外设和存储器、存储器和外设之间的传输
- 闪存、SRAM、外设的 SRAM、APB1、APB2 和 AHB 外设均可作为访问的源和目标。
- 可编程的数据传输数目：最大为 65535

下面为功能框图：

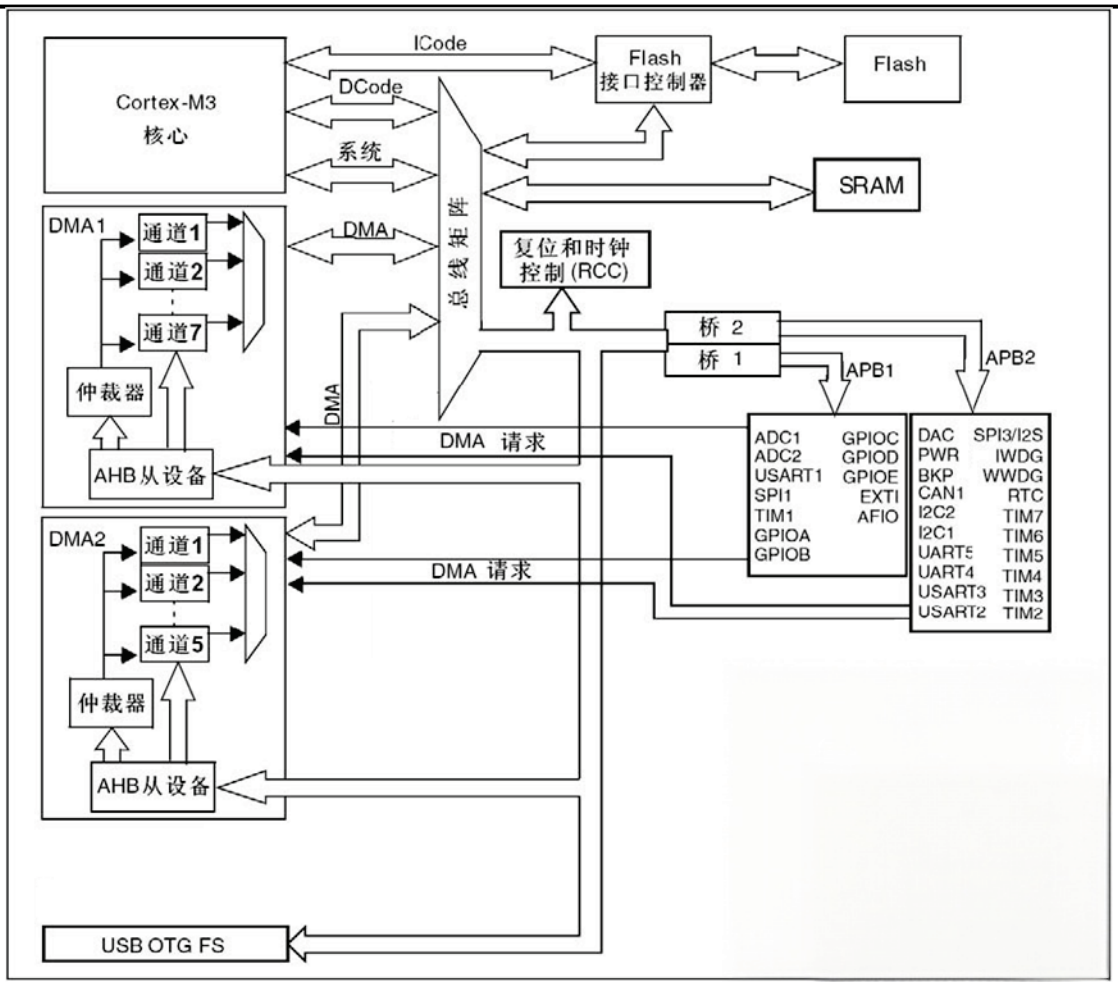


图 19 DMA 框图

10.3 功能描述

DMA 控制器和 Cortex™-M3 核心共享系统数据总线，执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标(RAM 或外设)时，DMA 请求会暂停 CPU 访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证 CPU 至少可以得到一半的系统总线(存储器或外设)带宽。

10.3.1 DMA 处理

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

总之，每次 DMA 传送由 3 个操作组成:

- 从外设数据寄存器或者从当前外设/存储器地址寄存器指示的存储器地址取数据，第一次传输时的开始地址是 DMA_CPARx 或 DMA_CMARx 寄存器指定的外设基地址或存储器单元。
- 存数据到外设数据寄存器或者当前外设/存储器地址寄存器指示的存储器地址，第一次传输时的开始地址是 DMA_CPARx 或 DMA_CMARx 寄存器指定的外设基地址或存储器单元。
- 执行一次 DMA_CNDTRx 寄存器的递减操作，该寄存器包含未完成的操作数目。

10.3.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。优先权管理分 2 个阶段：

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

10.3.3 DMA 通道

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

可编程的数据量

外设和存储器的传输数据量可以通过 DMA_CCRx 寄存器中的 PSIZE 和 MSIZE 位编程。

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。

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/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(x 代表通道号)：

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 寄存器的 PL[1:0]位中设置通道的优先级。
5. 在 DMA_CCRx 寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置 DMA_CCRx 寄存器的 ENABLE 位，启动该通道。

一旦启动了 DMA 通道，它既可响应连到该通道上的外设的 DMA 请求。

当传输一半的数据后，半传输标志(HTIF)被置 1，当设置了允许半传输中断位(HTIE)时，将产生一个中断请求。在数据传输结束后，传输完成标志(TCIF)被置 1，当设置了允许传输完成中断位(TCIE)时，将产生一个中断请求。

循环模式

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

存储器到存储器模式

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

当设置了 DMA_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。存储器到存储器模式不能与循环模式同时使用。

10.3.4 可编程的数据传输宽度、对齐方式和数据大小端

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

表 55 可编程的数据传输宽度和大小端操作(当 PINC=MINC=1)

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

			0x8/BBBAB9B8 0xC/BFBEBDBC	2:在0x4读B7B6B5B4[31:0]，在0x2写B5B4[15:0] 3:在0x8读BBBAB9B8[31:0]，在0x4写B9B8[15:0] 4:在0xC读BFBEBDBC[31:0]，在0x6写BDBC[15:0]	0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:在0x0读B3B2B1B0[31:0]，在0x0写B3B2B1B0[31:0] 2:在0x4读B7B6B5B4[31:0]，在0x4写B7B6B5B4[31:0] 3:在0x8读BBBAB9B8[31:0]，在0x8写BBBAB9B8[31:0] 4:在0xC读BFBEBDBC[31:0]，在0xC写BFBEBDBC[31:0]	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

操作一个不支持字节或半字写的 AHB 设备

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

- 当 HSIZE=半字时，写入半字'0xABCD'，DMA 将设置 HWDATA 总线为'0xABCD ABCD'。
- 当 HSIZE=字节时，写入字节'0xAB'，DMA 将设置 HWDATA 总线为'0xABAB ABAB'。
- 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'0xB0B0 B0B0'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'0xB1B0 B1B0'操作。

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

10.3.5 错误管理

读写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时，硬件会自动地清除发生错误的通道所对应的通道配置寄存器(DMA_CCRx)的 EN 位，该通道操作被停止。此时，在 DMA_IFR 寄存器中对应该通道的传输错误中断标志位(TEIF)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。

10.3.6 中断

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

表 56 DMA 中断请求

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

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

10.3.7 DMA 请求映像

DMA1 控制器

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

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

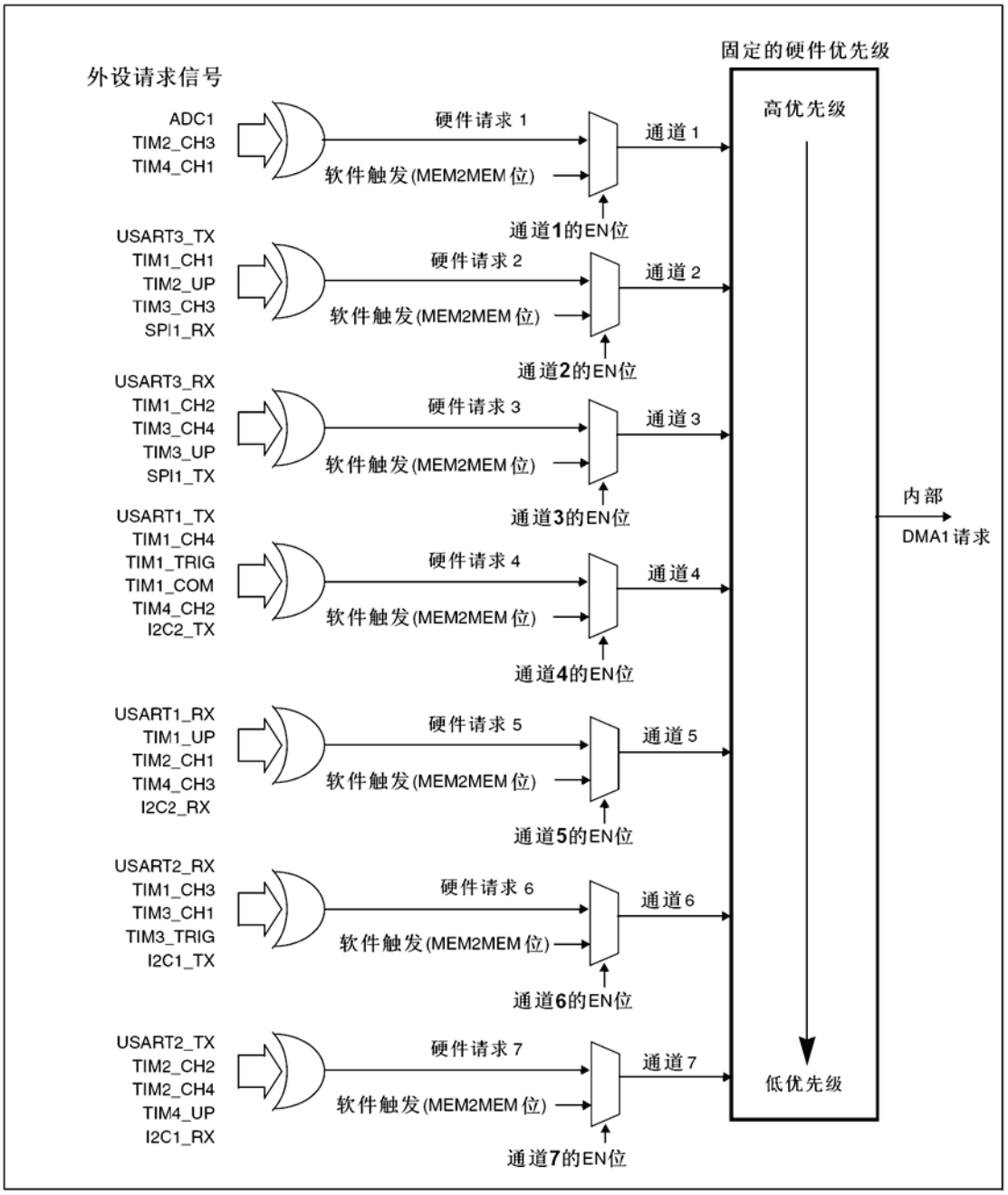


图 20 DMA1 请求映像

表 57 各个通道的 DMA1 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
ADC1	ADC1						
SPI/I2S		SPI1_RX	SPI1_TX				
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_TX4TIM1_TRIGTIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

DMA2 控制器

从外设(TIMx[5、6、7、8]、ADC3、SPI/I2S3、UART4、DAC 通道 1、2 和 SDIO)产生的 5 个请求，经逻辑或输入到 DMA2 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA2 请求映像。外设的 DMA 请求，可以通过设置相应外设寄存器中的 DMA 控制位，被独立地开启或关闭。

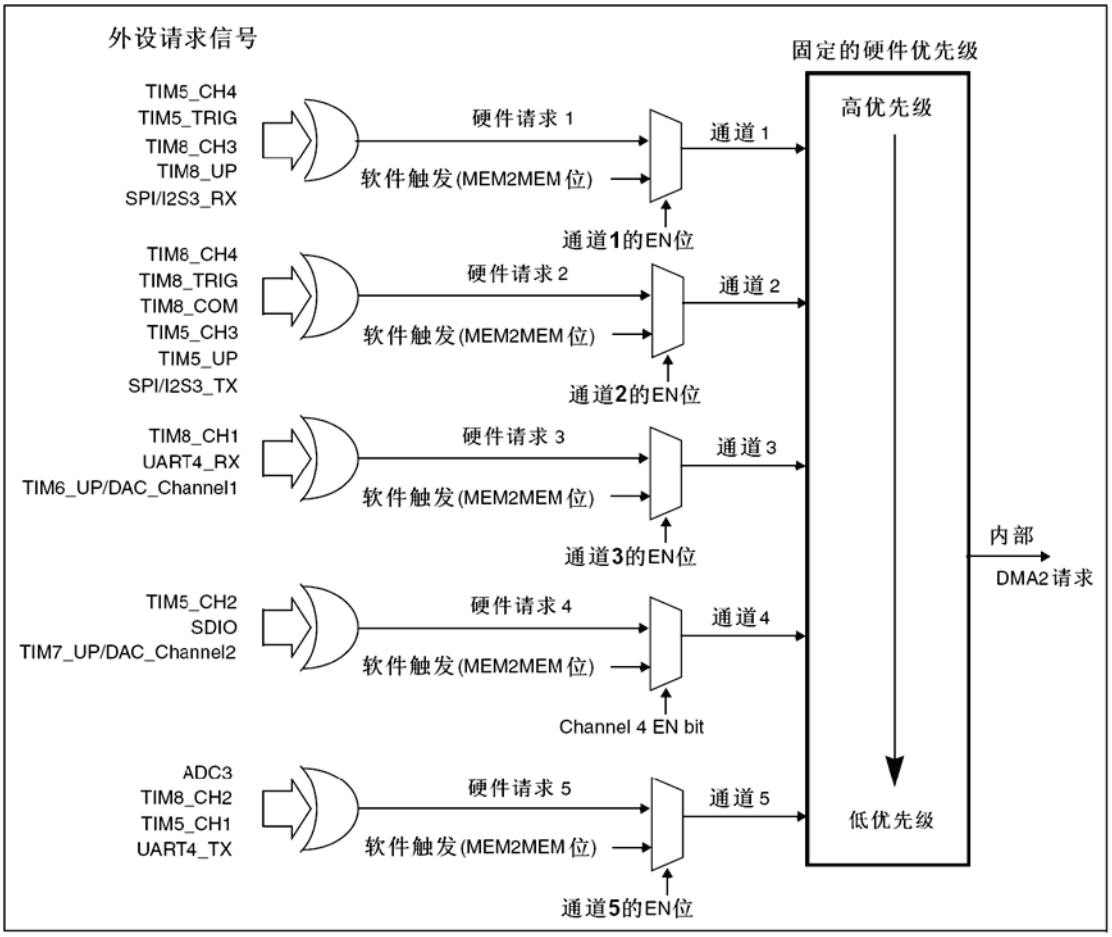


图 21 DMA2 请求映像

表 58 各个通道的 DMA2 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5
ADC3					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
TIM5	TIM5_CH4TIM5_TRIG	TIM5_CH3TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/DAC 通道 1			TIM6_UP/DAC 通道 1		
TIM7/DAC 通道 2				TIM7_UP/DAC 通道 2	
TIM8	TIM8_CH3TIM8_UP	TIM8_CH4TIM8_TRIGTIM8_COM	TIM8_CH1		TIM8_CH2

10.4 DMA 寄存器

关于寄存器描述中用到的缩写，请参见第 1 章。

注意：在以下列举的所有寄存器中，所有与通道 6 和通道 7 相关的位，对 DMA2 都不适用，因为 DMA2 只有 5 个通道。

10.4.1 DMA 中断状态寄存器(DMA_ISR)

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r

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

位	符号	说明
31:28	Reserved	保留，始终读为 0。
27,23,19,15,11,7,3	TEIFx	**TEIFx**：通道 x 的传输错误标志(x=1...7)(Channel x transfer error flag) 硬件设置这些位。在 DMA_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)。
26,22,18,14,10,6,2	HTIFx	**HTIFx**：通道 x 的半传输标志(x=1...7)(Channel x half transfer flag) 硬件设置这些位。在 DMA_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)。
25,21,17,13,9,5,1	TCIFx	**TCIFx**：通道 x 的传输完成标志(x=1...7)(Channel x transfer complete flag) 硬件设置这些位。在 DMA_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)。
24,20,16,12,8,4,0	GIFx	**GIFx**：通道 x 的全局中断标志(x=1...7)(Channel x global interrupt flag) 硬件设置这些位。在 DMA_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。

10.4.2 DMA 中断标志清除寄存器(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
保留				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:28	Reserved	保留，始终读为 0。
27,23,19,15,11,7,3	CTEIFx	**CTEIFx**：清除通道 x 的传输错误标志(x=1...7)(Channel x transfer error clear)这些位由软件设置和清除。 0：不起作用 1：清除 DMA_ISR 寄存器中的对应 TEIF 标志。
26,22,18,14,10,6,2	CHTIFx	**CHTIFx**：清除通道 x 的半传输标志(x=1...7)(Channel x half transfer clear)这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应 HTIF 标志。
25,21,17,13,9,5,1	CTCIF	**CTCIFx**：清除通道 x 的传输完成标志(x=1...7)(Channel x transfer complete clear)这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应 TCIF 标志。
24,20,16,12,8,4,0	CGIFx	**CGIFx**：清除通道 x 的全局中断标志(x=1...7)(Channel x global interrupt clear)这些位由软件设置和清除。 0：不起作用 0：清除 DMA_ISR 寄存器中的对应的 GIF、TEIF、HTIF 和 TCIF 标志。

10.4.3 DMA 通道 x 配置寄存器(DMA_CCRx)(x=1...7)

偏移地址：0x08+0d20x(通道编号−1)

复位值：0x0000 0000

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

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

位	符号	说明
31:15	Reserved	保留，始终读为 0。
14	MEM2MEM	**MEM2MEM**：存储器到存储器模式(Memory to memory mode) 该位由软件设置和清除。 0：非存储器到存储器模式； 1：启动存储器到存储器模式。
13:12	PL[1:0]	**PL[1:0]**：通道优先级(Channel priority level) 这些位由软件设置和清除。 00：低 01：中 10：高

		11：最高
11:10	MSIZE[1:0]	**MSIZE[1:0]**：存储器数据宽度(Memory size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
9:8	PSIZE[1:0]	**PSIZE[1:0]**：外设数据宽度(Peripheral size) 这些位由软件设置和清除。 00：8 位 01：16 位 10：32 位 11：保留
7	MINC	**MINC**：存储器地址增量模式(Memory increment mode) 该位由软件设置和清除。 0：不执行存储器地址增量操作 1：执行存储器地址增量操作
6	PINC	**PINC**：外设地址增量模式(Peripheral increment mode) 该位由软件设置和清除。 0：不执行外设地址增量操作 1：执行外设地址增量操作
5	CIRC	**CIRC**：循环模式(Circular mode) 该位由软件设置和清除。 0：不执行循环操作 1：执行循环操作
4	DIR	**DIR**：数据传输方向(Data transfer direction) 该位由软件设置和清除。 0：从外设读 1：从存储器读
3	TEIE	**TEIE**：允许传输错误中断(Transfer error interrupt enable) 该位由软件设置和清除。 0：禁止 TE 中断 0：允许 TE 中断
2	HTIE	**HTIE**：允许半传输中断(Half transfer interrupt enable) 该位由软件设置和清除。 0：禁止 HT 中断 0：允许 HT 中断
1	TCIE	**TCIE**：允许传输完成中断(Transfer complete interrupt enable) 该位由软件设置和清除。 0：禁止 TC 中断 0：允许 TC 中断
0	EN	**EN**：通道开启(Channel enable) 该位由软件设置和清除。 0：通道不工作 1：通道开启

10.4.4 DMA 通道 x 传输数量寄存器(DMA_CNDTRx)(x=1...7)

偏移地址：0x0C+0d20x(通道编号–1)

复位值：0x0000 0000

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

位	符号	说明

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

10.4.5 DMA 通道 x 外设地址寄存器(DMA_CPARx)(x=1...7)

偏移地址：0x10+0d20x(通道编号–1)

复位值：0x0000 0000

当开启通道(DMA_CCRx 的 EN=1)时不能写该寄存器。

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

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

10.4.6 DMA 通道 x 存储器地址寄存器(DMA_CMARx)(x=1...7)

偏移地址：0x14+20x(通道编号–1)

复位值：0x0000 0000

当开启通道(DMA_CCRx 的 EN=1)时不能写该寄存器。

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

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

10.4.7 DMA 寄存器映像

表 59 DMA 寄存器映像和复位

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	DMA_ISR	保留				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	DMA_IFCR	保留				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	复位值					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	DMA_CCR1	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	DMA_CNDTR1	保留																NDT[15:0]																
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	DMA_CPAR1	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	DMA_CMAR1	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	保留																																
01Ch	DMA_CCR2	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	DMA_CNDTR2	保留																NDT[15:0]																
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DMA_CPAR2	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	DMA_CMAR2	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	保留																																
030h	DMA_CCR3	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034h	DMA_CNDTR3	保留																NDT[15:0]																
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	DMA_CPAR3	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	DMA_CMAR3	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	保留																																

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
044h	DMA_CCR4	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	DMA_CNDTR4	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	DMA_CPAR4	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	DMA_CMAR4	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	保留																																
058h	DMA_CCR5	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	DMA_CNDTR5	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	DMA_CPAR5	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
064h	DMA_CMAR5	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
068h	保留																																
06Ch	DMA_CCR6	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	DMA_CNDTR6	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	DMA_CPAR6	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	DMA_CMAR6	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	保留																																
080h	DMA_CCR7	保留																	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	DMA_CNDTR7	保留																	NDT[15:0]														
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	DMA_CPAR7	PA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	DMA_CMAR7	MA[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	保留																																

关于寄存器的起始地址，参见表 1。

11 模拟/数字转换(ADC)

11.1 ADC 介绍

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 18 个通道，可测量 16 个外部和 2 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阈值。

ADC 的输入时钟不得超过 14MHz，它是由 PCLK2 经分频产生。

11.2 ADC 主要特征

- 12 位分辨率
- 转换结束、注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道 0 到通道 n 的自动扫描模式
- 自校准
- 带内嵌数据一致性的数据对齐
- 采样间隔可以按通道分别编程
- 规则转换和注入转换均有外部触发选项
- 间断模式
- 双重模式(带 2 个或以上 ADC 的器件)
- ADC 转换时间：

时钟为 56MHz 时为 1μs(时钟为 72MHz 为 1.17μs)

- ADC 供电要求：2.4V 到 3.6V
- ADC 输入范围：VREF-≤VIN≤VREF+
- 规则通道转换期间有 DMA 请求产生。下图是 ADC 模块的方框图。

注意：　如果有 VREF-引脚，必须和 VSSA 相连接

11.3 ADC 功能描述

下图为一个 ADC 模块的框图，表 60 为 ADC 引脚的说明。

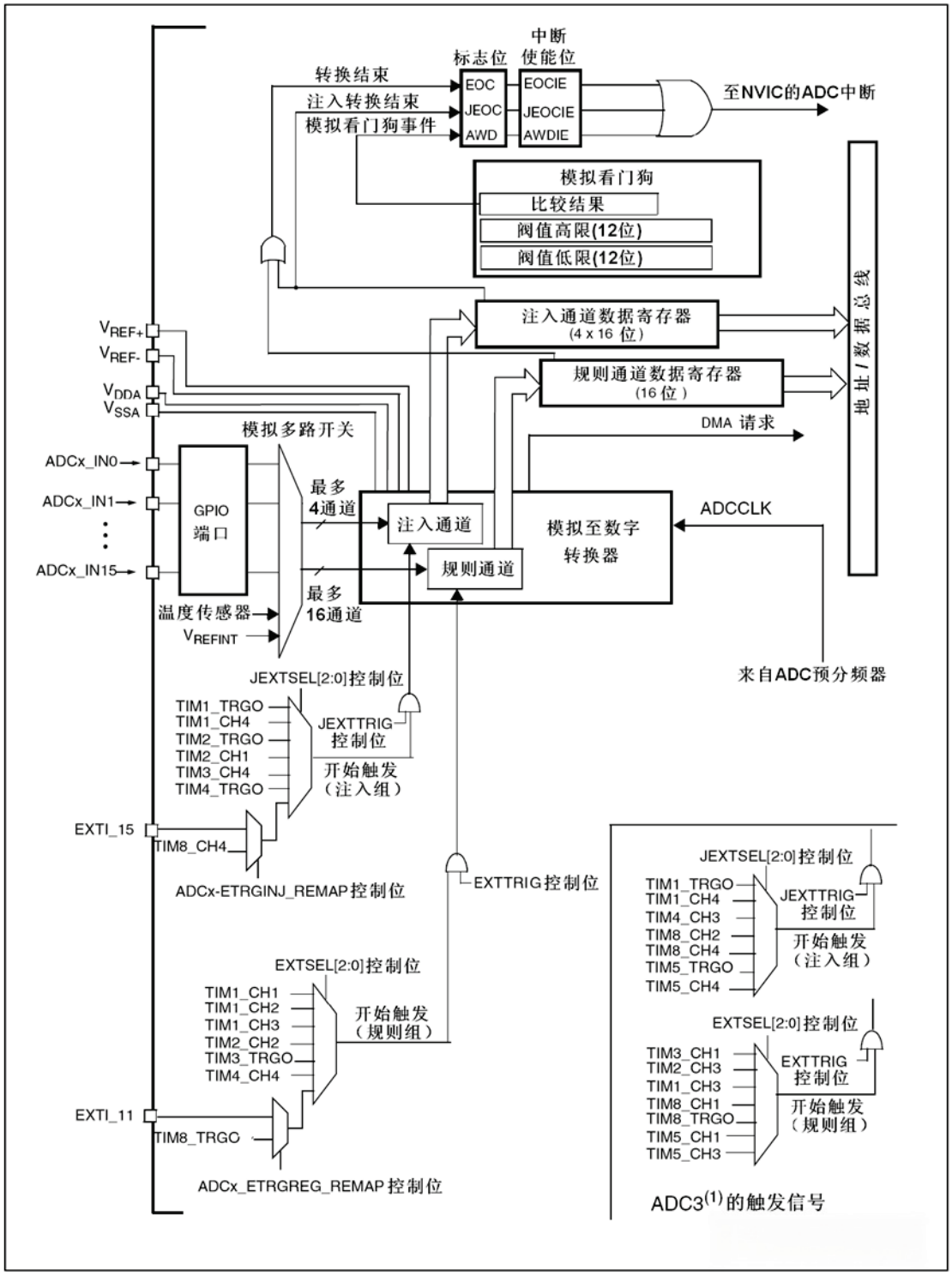


图 22 单个 ADC 框图

1．ADC3 的规则转换和注入转换触发与 ADC1 和 ADC2 的不同。

表 60 ADC 引脚

名称	信号类型	注解
VREF+	输入，模拟参考正极	ADC 使用的高端/正极参考电压，2.4V≤VREF+≤VDDA
VDDA(1)	输入，模拟电源	等效于 VDD 的模拟电源且：2.4V≤VDDA≤VDD(3.6V)
VREF-	输入，模拟参考负极	ADC 使用的低端/负极参考电压，VREF-=VSSA
VSSA(1)	输入，模拟电源地	等效于 VSS 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。

11.3.1 ADC 开关控制

通过设置 ADC_CR2 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后(tSTAB)，再次设置 ADON 位时开始进行转换。

通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。在这个模式中，ADC 几乎不耗电(仅几个μA)。

11.3.2 ADC 时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2(APB2 时钟)同步。RCC 控制器为 ADC 时钟提供一个专用的可编程预分频器，详见第 6 节-复位和时钟控制(RCC)。

11.3.3 通道选择

有 16 个多路通道。可以把转换组织成两组：规则组和注入组。在任意多个通道上以任意顺序进行的一系列转换构成成组转换。例如，可以如下顺序完成转换：通道 3、通道 8、通道 2、通道 2、通道 0、通道 2、通道 2、通道 15。

- 规则组由多达 16 个转换组成。规则通道和它们的转换顺序在 ADC_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[3:0]位中。
- 注入组由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_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 L[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。

温度传感器/VREFINT 内部通道

温度传感器和通道 ADC1_IN16 相连接，内部参照电压 VREFINT 和 ADC1_IN17 相连接。可以按注入或规则通道对这两个内部通道进行转换。

注意：　温度传感器和 VREFINT 只能出现在主 ADC1 中。

11.3.4 单次转换模式

单次转换模式下，ADC 只执行一次转换。该模式既可通过设置 ADC_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：

- 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)标志被设置
- 如果设置了 EOCIE，则产生中断。

- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

然后 ADC 停止。

11.3.5 连续转换模式

在连续转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。此模式可通过外部触发启动或通过设置 ADC_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 位的 ADC_DR 寄存器中
  - EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC_DRJ1 寄存器中
  - JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

11.3.6 时序图

如下图所示，ADC 在开始精确转换前需要一个稳定时间 tSTAB。在开始 ADC 转换和 14 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。

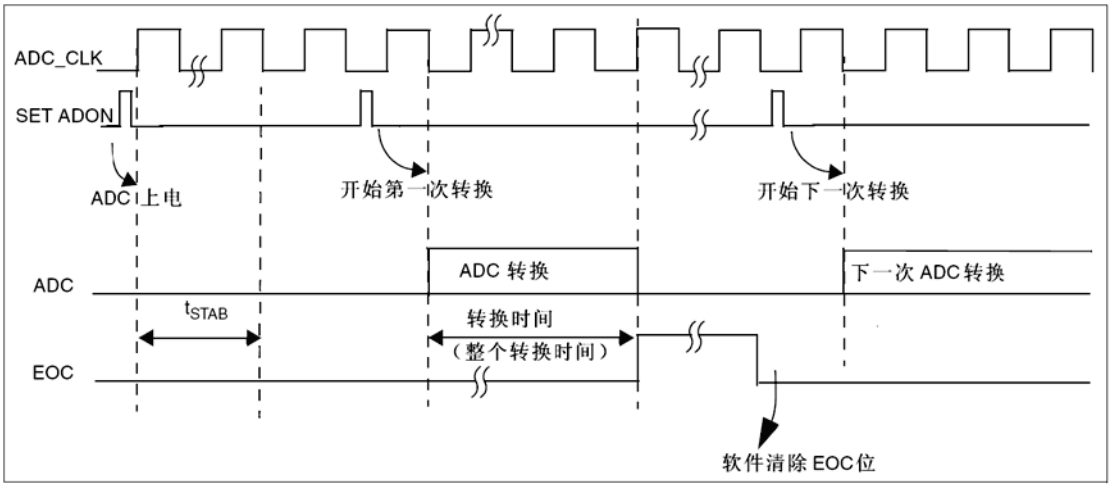


图 23 时序图

11.3.7 模拟看门狗

如果被 ADC 转换的模拟电压低于低阈值或高于高阈值，AWD 模拟看门狗状态位被设置。阈值位于 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见 11.5 节)。

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，如表 61 所示。

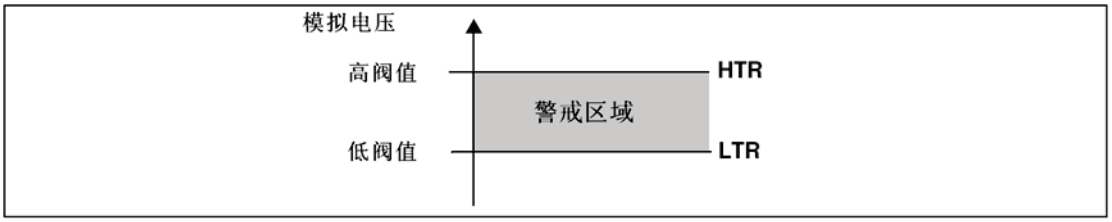


图 24 模拟看门狗警戒区

表 61 模拟看门狗通道选择

模拟看门狗警戒的通道	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

11.3.8 扫描模式

此模式用来扫描一组模拟通道。

扫描模式可通过设置 ADC_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。

如果设置了 DMA 位，在每次 EOC 后，DMA 控制器把规则组通道的转换数据传输到 SRAM 中。而注入通道转换的数据总是存储在 ADC_JDRx 寄存器中。

11.3.9 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，注入转换不会被中断，但是规则序列将在注入序列结束后被执行。图 25 是其定时图。

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 28 个 ADC 时钟周期(即 2 个具有 1.5 个时钟间隔采样时间的转换)，触发之间最小的间隔必须是 29 个 ADC 时钟周期。

自动注入

如果设置了 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 20 个转换序列。

在此模式里，必须禁止注入通道的外部触发。

如果除 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。

对于 ADC 时钟预分频系数为 4 至 8 时，当从规则转换切换到注入序列或从注入转换切换到规则序列时，会自动插入 1 个 ADC 时钟间隔；当 ADC 时钟预分频系数为 2 时，则有 2 个 ADC 时钟间隔的延迟。

注意：　不可能同时使用自动注入和间断模式。

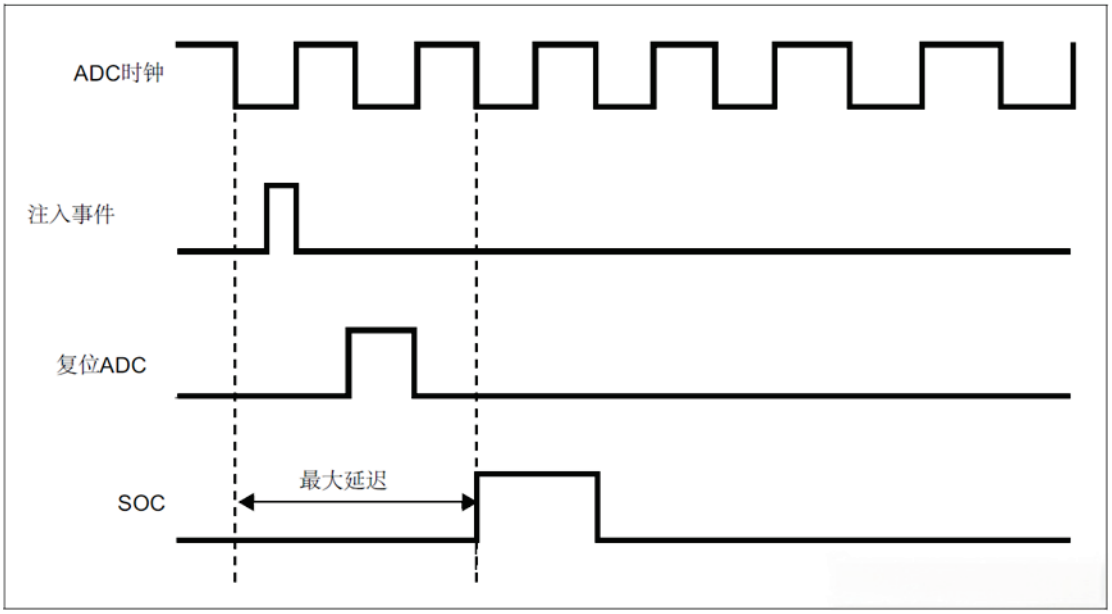


图 25 注入转换延时

11.3.10 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[3:0]定义。

举例：

n=3，被转换的通道=0、1、2、3、6、7、9、10

第一次触发：转换的序列为 0、1、2

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 0、1、2

注意：　当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。

当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0、1 和 2。

注入组

此模式通过设置 ADC_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。

举例：

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

注意：1 当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。

2 不能同时使用自动注入和间断模式。

3 必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

11.4 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_DR 中。

注意：1 建议在每次上电后执行一次校准。

2 启动校准前，ADC 必须处于关电状态(ADON='0')超过至少两个 ADC 时钟周期。

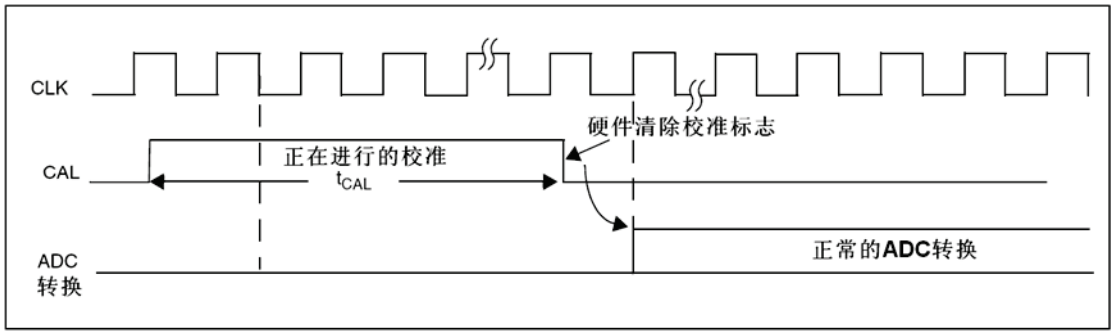


图 26 校准时序图

11.5 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，如图 27 和图 28 所示。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。

图 27 数据右对齐

图 28 数据左对齐

11.6 可编程的通道采样时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，采样周期数目可以通过 ADC_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：

TCONV=采样时间+12.5 个周期

例如：

当 ADCCLK=14MHz，采样时间为 1.5 周期

TCONV=1.5+12.5=14 周期=1μs

11.7 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[2:0]和 JEXTSEL2:0]控制位允许应用程序选择 8 个可能的事件中的某一个，可以触发规则和注入组的采样。

注意：当外部触发信号被选为 ADC 规则或注入转换时，只有它的上升沿可以启动转换。

表 62 ADC1 和 ADC2 用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
TIM1_CC1 事件	来自片上定时器的内部信号	000
TIM1_CC2 事件		001
TIM1_CC3 事件		010
TIM2_CC2 事件		011
TIM3_TRGO 事件		100
TIM4_CC4 事件		101
EXTI 线 11/TIM8_TRGO 事件(1)	外部引脚/来自片上定时器的内部信号	110
SWSTART	软件控制位	111

1．对于规则通道，选中 EXTI 线路 11 或 TIM8_TRGO 作为外部触发事件，可以分别通过设置 ADC1 和 ADC2 的 ADC1_ETRGREG_REMAP 位和 ADC2_ETRGREG_REMAP 位实现。

表 63 ADC1 和 ADC2 用于注入通道的外部触发

触发源	连接类型	JEXTSE[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号	000
TIM1_CC4 事件		001
TIM2_TRGO 事件		010
TIM2_CC1 事件		011
TIM3_CC4 事件		100
TIM4_TRGO 事件		101
EXTI 线 15/TIM8_CC4 事件(1)	外部引脚/来自片上定时器的内部信号	110
JSWSTART	软件控制位	111

1．为注入信道选择外部触发器 EXTIline15 或 TIM8-CC4 事件是通过分别为 ADC1 和 ADC2 配置位 ADC1ETRGINJREMAP 和 ADC2 ETRGINJ·REMAP 来完成的。

表 64 ADC3 用于规则通道的外部触发

触发源	连接类型	EXTSEL[2:0]
TIM3_CC1 事件	来自片上定时器的内部信号	000
TIM2_CC3 事件		001
TIM1_CC3 事件		010
TIM8_CC1 事件		011
TIM8_TRGO 事件		100
TIM5_CC1 事件		101
TIM5_CC3 事件		110
SWSTART	软件控制位	111

表 65 ADC3 用于注入通道的外部触发

触发源	连接类型	JEXTSE[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号 a	000
TIM1_CC4 事件		001
TIM4_CC3 事件		010
TIM8_CC2 事件		011
TIM8_CC4 事件		100
TIM5_TRGO 事件		101
TIM5_CC4 事件		110
JSWSTART	软件控制位	111

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。

11.8 DMA 请求

因为规则通道转换的值储存在一个仅有的数据寄存器中，所以当转换多个规则通道时需要使用 DMA，这可以避免丢失已经存储在 ADC_DR 寄存器中的数据。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC_DR 寄存器传输到用户指定的目的地址。

注： *只有 ADC1 和 ADC3 拥有 DMA 功能。由 ADC2 转化的数据可以通过双 ADC 模式，利用 ADC1 的 DMA 功能传输。*

11.9 双 ADC 模式

在有 2 个或以上 ADC 模块的产品中，可以使用双 ADC 模式(见图 29 双 ADC 框图)。

在双 ADC 模式里，根据 ADC1_CR1 寄存器中 DUALMOD[2:0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同步触发。

注意：在双 ADC 模式里，当转换配置成由外部事件触发时，用户必须将其设置成仅触发主 ADC，从 ADC 设置成软件触发，这样可以防止意外的触发从转换。但是，主和从 ADC 的外部触发必须同时被激活。

共有 6 种可能的模式：

- 同步注入模式
- 同步规则模式
- 快速交叉模式
- 慢速交叉模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同步注入模式+同步规则模式
- 同步规则模式+交替触发模式
- 同步注入模式+交叉模式

注意：在双 ADC 模式里，为了在主数据寄存器上读取从转换数据，必须使能 DMA 位，即使不使用 DMA 传输规则通道数据。

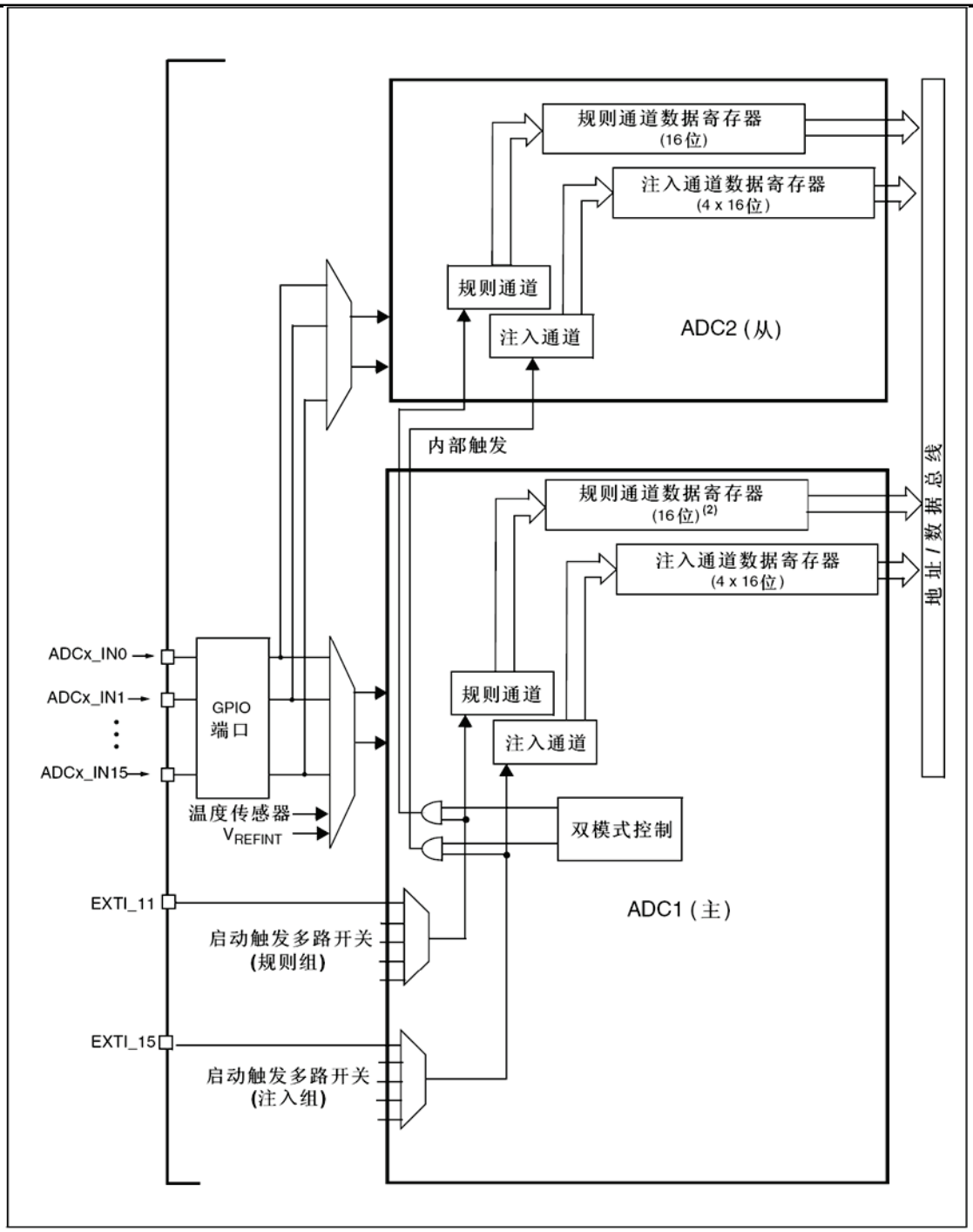


图 29 双 ADC 框图(1)

1．外部触发信号作用于 ADC2，但在本图中没有显示。

2．在某些双 ADC 模式中，在完整的 ADC1 数据寄存器(ADC1_DR)中包含了 ADC1 和 ADC2 的规则转换数据。

11.9.1 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[2:0]选择)，它同时给 ADC2 提供同步触发。

注意： *不要在 2 个 ADC 上转换相同的通道(两个 ADC 在同一个通道上的采样时间不能重叠)。*

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个 ADC 接口的 ADC_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。*

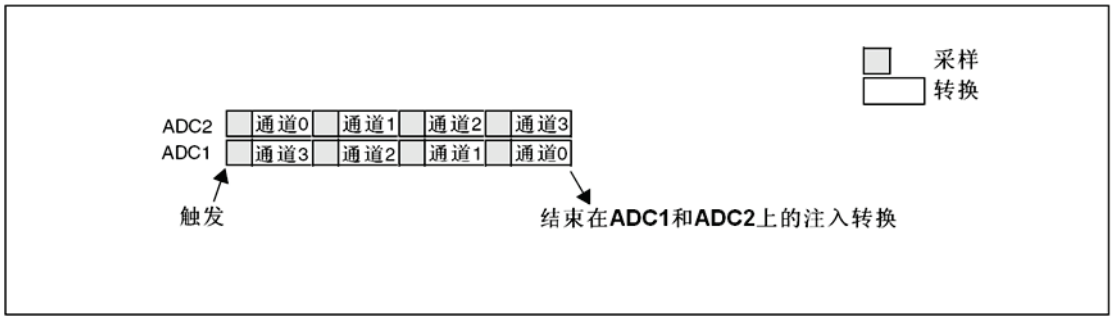


图 30 在 4 个通道上的同步注入模式

11.9.2 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[2:0]选择)，它同时给 ADC2 提供同步触发。

注意： *不要在 2 个 ADC 上转换相同的通道((两个 ADC 在同一个通道上的采样时间不能重叠)。*

在 ADC1 或 ADC2 的转换结束时：

- 产生一个 32 位 DMA 传输请求(如果设置了 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。*

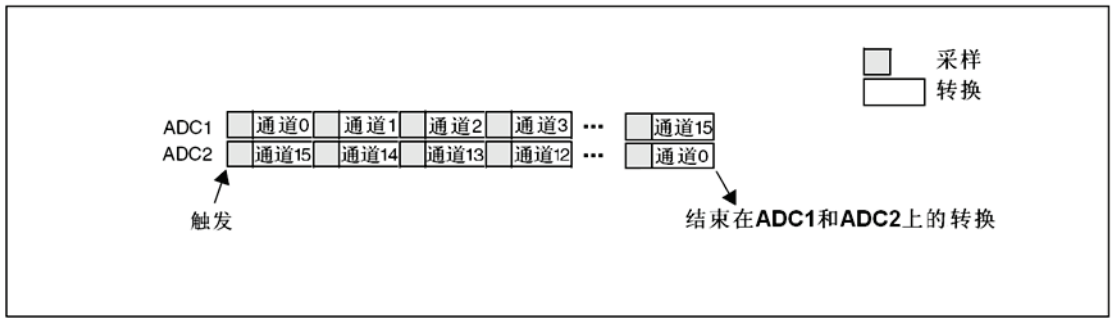


图 31 在 16 个通道上的同步规则模式

11.9.3 快速交叉模式

此模式只适用于规则通道组(通常为一个通道)。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后:

- ADC2 立即启动并且
- ADC1 在延迟 7 个 ADC 时钟周期后启动

如果同时设置了 ADC1 和 ADC2 的 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。

注意：　最大允许采样时间<7 个 ADCCLK 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。

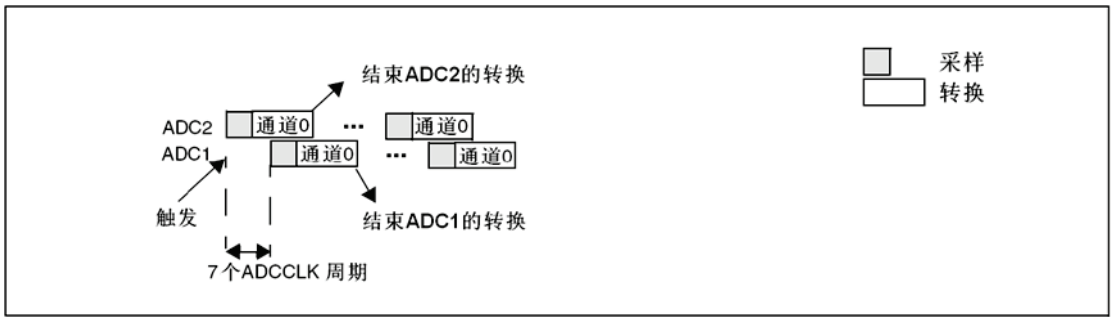


图 32 在 1 个通道上连续转换模式下的快速交叉模式

11.9.4　慢速交叉模式

此模式只适用于规则通道组(只能为一个通道)。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后:

- ADC2 立即启动并且
- ADC1 在延迟 14 个 ADC 时钟周期后启动
- 在延迟第二次 14 个 ADC 周期后 ADC2 再次启动，如此循环。

注意：　最大允许采样时间<14 个 ADCCLK 周期，以避免和下个转换重叠。

ADC1 产生一个 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 28 个 ADC 时钟周期后自动启动新的 ADC2 转换。

在这个模式下不能设置 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。

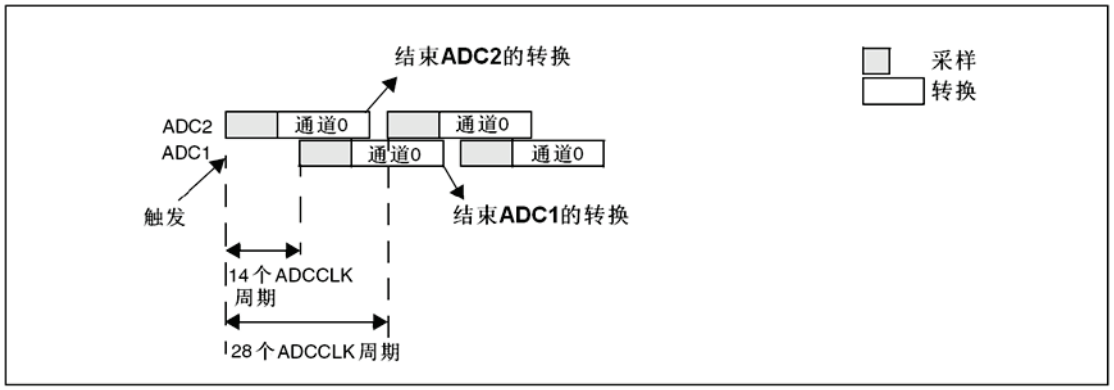


图 33 在 1 个通道上的慢速交叉模式

11.9.5　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。

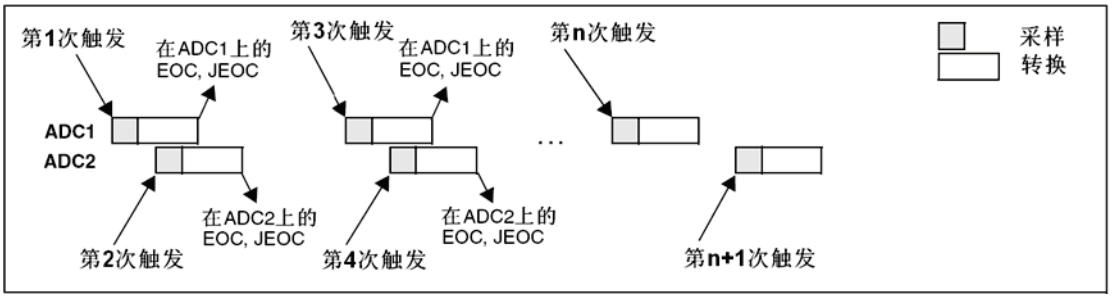


图 34 交替触发：每个 ADC1 的注入通道组

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。

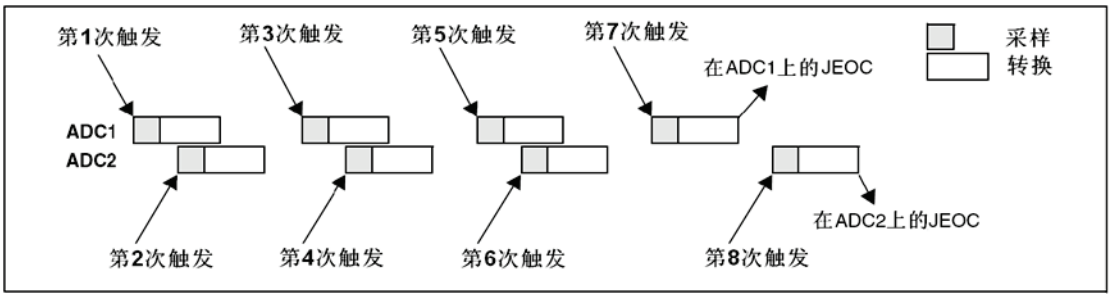


图 35 交替触发：在间断模式下每个 ADC 上的 4 个注入通道

11.9.6 独立模式

此模式里，双 ADC 同步不工作，每个 ADC 接口独立工作。

11.9.7 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。

11.9.8 混合的同步规则+交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。图 36 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。

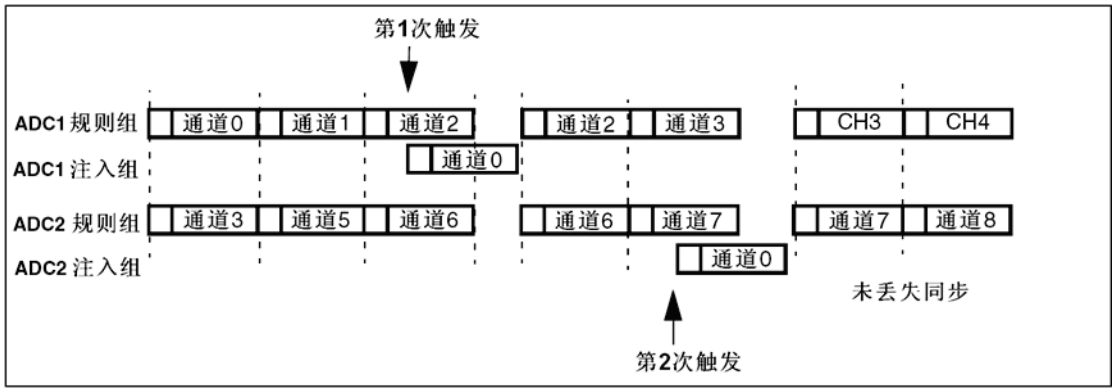


图 36 交替+规则同步

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)。

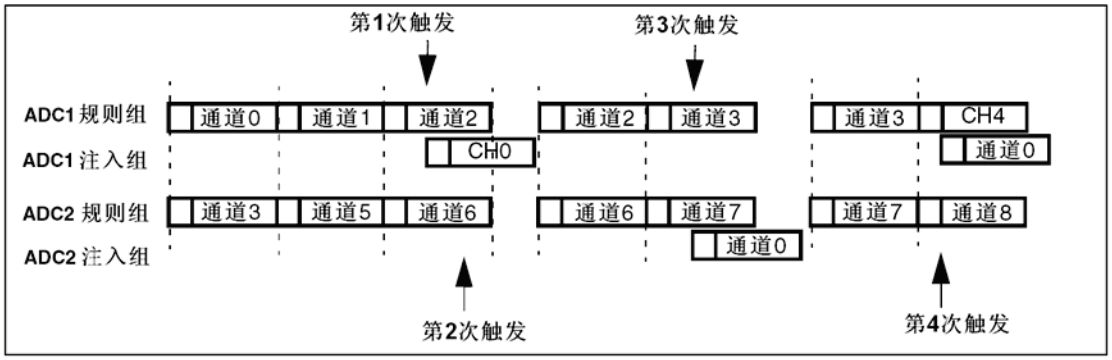


图 37 触发事件发生在注入转换期间

11.9.9 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。

注：当 ADC 时钟预分频系数设置为 4 时，交叉模式恢复后不会均匀地分配采样时间，采样间隔是 8 个 ADC 时钟周期与 6 个 ADC 时钟周期轮替，而不是均匀的 7 个 ADC 时钟周期。

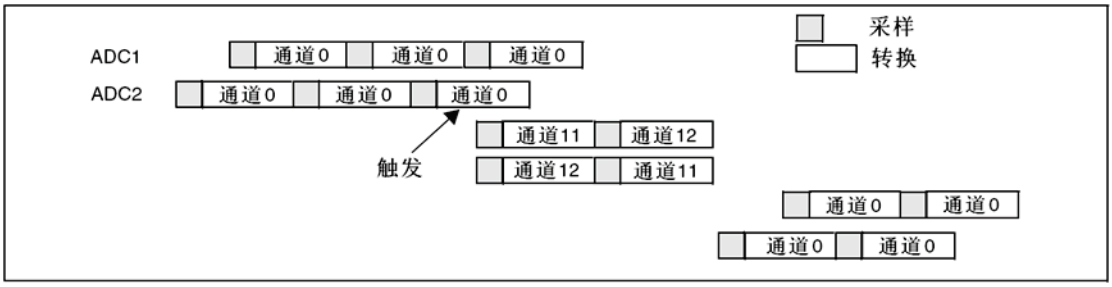


图 38 交叉的单通道转换被注入序列 CH11 和 CH12 中断

11.10 温度传感器

温度传感器可以用来测量器件周围的温度(TA)。

温度传感器在内部和 ADC1_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入推荐采样时间是 17.1μs。

图 39 是温度传感器的方框图。

当没有被使用时，传感器可以置于关电模式。

注意：　必须设置 TSVREFE 位激活内部通道：ADC1_IN16(温度传感器)和 ADC1_IN17(VREFINT)的转换。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同(最多相差 45°C)。

内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

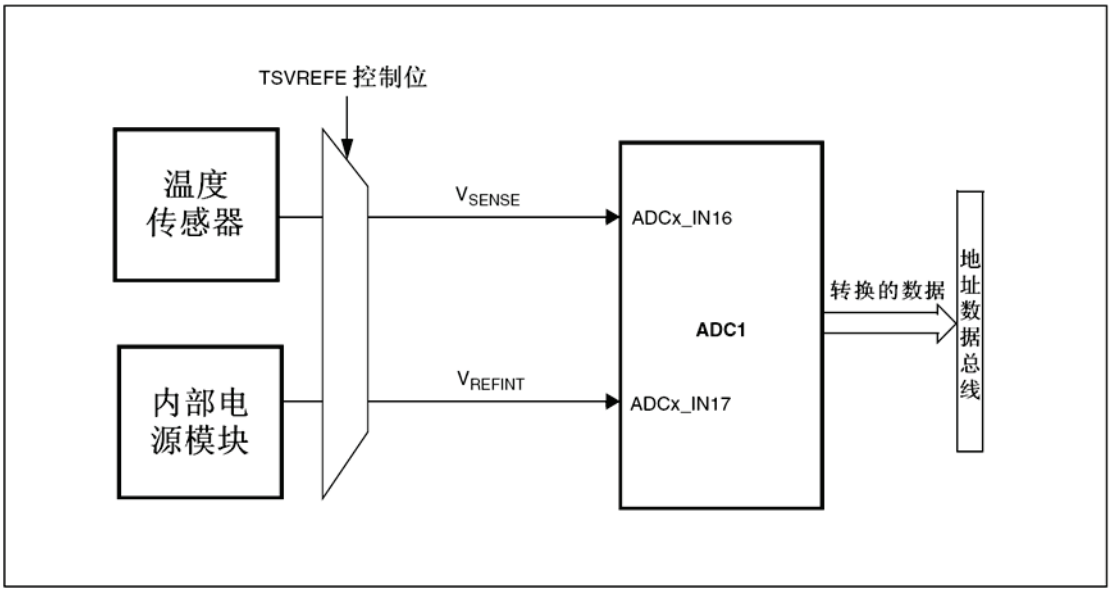


图 39 温度传感器和 VREFINT 通道框图

读温度

为使用传感器：

1. 选择 ADC1_IN16 输入通道
2. 选择采样时间为 17.1μs
3. 设置 ADC 控制寄存器 2(ADC_CR2)的 TSVREFE 位，以唤醒关电模式下的温度传感器
4. 通过设置 ADON 位启动 ADC 转换(或用外部触发)
5. 读 ADC 数据寄存器上的 VSENSE 数据结果
6. 利用下列公式得出温度

   温度(°C)={(V25-VSENSE)/Avg_Slope}+25

   这里：

   V25=VSENSE 在 25°C 时的数值

   Avg_Slope = 温度与 VSENSE 曲线的平均斜率(单位为 mV/°C 或μV/°C)参考数据手册的电气特性章节中 V25 和 Avg_Slope 的实际值。

注意：　传感器从关电模式唤醒后到可以输出正确水平的 VSENSE 前，有一个建立时间。ADC 在上电后也有一个建立时间，因此为了缩短延时，应该同时设置 ADON 和 TSVREFE 位。

11.11 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 66 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

11.12 ADC 寄存器

寄存器描述中使用的一些缩略语请参考第 1 节。

必须以字(32 位)的方式操作这些外设寄存器。

11.12.1 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
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位	符号	说明
31:15	Reserved	保留，始终读为 0。
4	STRT	**STRT**：规则通道开始位(Regular channel Start flag) 该位由硬件在规则通道转换开始时设置，由软件清除。 0：规则通道转换未开始； 1：规则通道转换已开始。
3	JSTRT	**JSTRT**：注入通道开始位(Injected channel Start flag) 该位由硬件在注入通道组转换开始时设置，由软件清除。 0：注入通道组转换未开始； 1：注入通道组转换已开始。
2	JEOC	**JEOC**：注入通道转换结束位(Injected channel end of conversion) 该位由硬件在所有注入通道组转换结束时设置，由软件清除 0：转换未完成； 1：转换完成。
1	EOC	**EOC**：转换结束位(End of conversion) 该位由硬件在(规则或注入)通道组转换结束时设置，由软件清除或由读取 ADC_DR 时清除 0：转换未完成； 1：转换完成。
0	AWD	**AWD**：模拟看门狗标志位(Analog watchdog flag) 该位由硬件在转换的电压值超出了 ADC_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。

11.12.2 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
保留								AWDEN	JAWDEN	保留		DUALMOD[3:0]			
								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

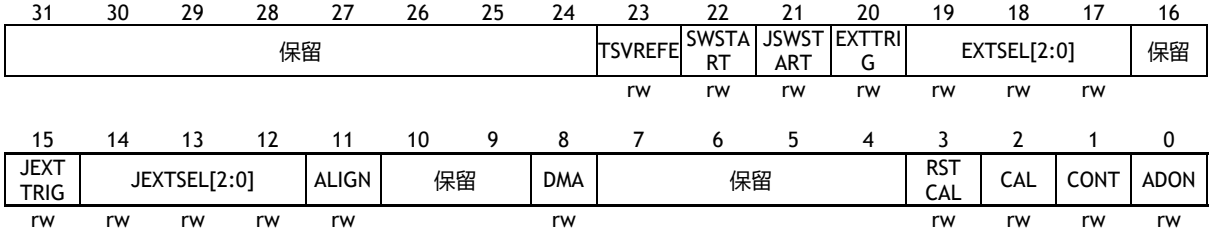
位	符号	说明
31:24	Reserved	保留，始终读为 0。
23	AWDEN	**AWDEN**：在规则通道上开启模拟看门狗(Analog watchdog enable on regular channels) 该位由软件设置和清除。 0：在规则通道上禁用模拟看门狗； 1：在规则通道上使用模拟看门狗。
22	JAWDEN	**JAWDEN**：在注入通道上开启模拟看门狗(Analog watchdog enable on injected channels) 该位由软件设置和清除。 0：在注入通道上禁用模拟看门狗； 1：在注入通道上使用模拟看门狗。
21:20	Reserved	保留。必须保持为 0。
19:16	DUALMOD [3:0]	**DUALMOD[3:0]**：双模式选择(Dual mode selection) 软件使用这些位选择操作模式。 0000：独立模式 0001：混合的同步规则+注入同步模式 0010：混合的同步规则+交替触发模式 0011：混合同步注入+快速交叉模式 0100：混合同步注入+慢速交叉模式 0101：注入同步模式 0110：规则同步模式 0111：快速交叉模式 1000：慢速交叉模式 1001：交替触发模式 注：在 ADC2 和 ADC3 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。
15:13	DISCNUM[2:0]	**DISCNUM[2:0]**：间断模式通道计数(Discontinuousmodechannelcount)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
12	JDISCEN	**JDISCEN**：在注入通道上的间断模式(Discontinuous mode on injected channels) 该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式 0：注入通道组上禁用间断模式； 1：注入通道组上使用间断模式。
11	DISCEN	**DISCEN**：在规则通道上的间断模式(Discontinuous mode on regular channels) 该位由软件设置和清除，用于开启或关闭规则通道组上的间断模式 0：规则通道组上禁用间断模式； 1：规则通道组上使用间断模式。

10	JAUTO	**JAUTO**：自动的注入通道组转换(Automatic Injected Group conversion) 该位由软件设置和清除，用于开启或关闭规则通道组转换结束后自动的注入通道组转换 0：关闭自动的注入通道组转换； 1：开启自动的注入通道组转换。
9	AWDSGL	**AWDSGL**：扫描模式中在一个单一的通道上使用看门狗(Enable the watchdog on a single channel in scan mode) 该位由软件设置和清除，用于开启或关闭由 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。
8	SCAN	**SCAN**：扫描模式(Scan mode) 该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，转换由 ADC_SQRx 或 ADC_JSQRx 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。
7	JEOCIE	**JEOCIE**：允许产生注入通道转换结束中断(Interrupt enable for injected channels)该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。 0：禁止 JEOC 中断； 1：允许 JEOC 中断。当硬件设置 JEOC 位时产生中断。
6	AWDIE	**AWDIE**：允许产生模拟看门狗中断(Analog watchdog interrupt enable) 该位由软件设置和清除，用于禁止或允许模拟看门狗产生中断。在扫描模式下，如果看门狗检测到超范围的数值时，只有在设置了该位时扫描才会中止。 0：禁止模拟看门狗中断； 1：允许模拟看门狗中断。
5	EOCIE	**EOCIE**：允许产生 EOC 中断(Interrupt enable for EOC) 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。
4:0	AWDCH[4:0]	**AWDCH[4:0]**：模拟看门狗通道选择位(Analog watchdog channel select bits)这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。 00000：ADC 模拟输入通道 000001：ADC 模拟输入通道 1 …… 01111：ADC 模拟输入通道 15 10000：ADC 模拟输入通道 16 10001：ADC 模拟输入通道 17 保留所有其他数值。 注：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 VREFINT。ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 VSS。ADC3 模拟输入通道 9、14、15、16、17 与 Vss 相连。

11.12.3 ADC 控制寄存器 2(ADC_CR2)

地址偏移：0x08

复位值：0x0000 0000



位	符号	说明
31:24	Reserved	保留，始终读为 0。
23	TSVREFE	**TSVREFE**：温度传感器和 VREFINT 使能(Temperature sensor and VREFINT enable) 该位由软件设置和清除，用于开启或禁止温度传感器和 VREFINT 通道。在多于 1 个 ADC 的器件中，该位仅出现在 ADC1 中。 0：禁止温度传感器和 VREFINT； 1：启用温度传感器和 VREFINT。
22	SWSTART	**SWSTART**：开始转换规则通道(Start conversion of regular channels) 由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 EXTSEL[2:0]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。
21	JSWSTART	**JSWSTART**：开始转换注入通道(Start conversion of injected channels) 由软件设置该位以启动转换，软件可清除此位或在转换开始后硬件马上清除此位。如果在 JEXTSEL[2:0]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。
20	EXTTRIG	**EXTTRIG**：规则通道的外部触发转换模式(External trigger conversion mode for regular channels) 该位由软件设置和清除，用于开启或禁止可以启动规则通道组转换的外部触发事件。 0：不用外部事件启动转换； 1：使用外部事件启动转换。
19:17	EXTSEL[2:0]	**EXTSEL[2:0]**：选择启动规则通道组转换的外部事件(Externaleventselectforregulargroup) 这些位选择用于启动规则通道组转换的外部事件 ADC1 和 ADC2 的触发配置如下 000：定时器 1 的 CC1 事件 001：定时器 1 的 CC2 事件　100：定时器 3 的 TRGO 事件 101：定时器 4 的 CC4 事件 010：定时器 1 的 CC3 事件 011：定时器 2 的 CC2 事件　110：EXTI 线 11/TIM8_TRGO 事件 ADC3 的触发配置如下　111：SWSTART 000：定时器 3 的 CC1 事件　100：定时器 8 的 TRGO 事件 101：定时器 5 的 CC1 事件 001：定时器 2 的 CC3 事件 010：定时器 1 的 CC3 事件　110：定时器 5 的 CC3 事件 011：定时器 8 的 CC1 事件　111：SWSTART
16	Reserved	保留，始终读为 0。
15	JEXTTRIG	**JEXTTRIG**：注入通道的外部触发转换模式(External trigger conversion mode for injected channels) 该位由软件设置和清除，用于开启或禁止可以启动注入通道组转换的外部触发事件。 0：不用外部事件启动转换；

		1：使用外部事件启动转换。
14:12	JEXTSEL[2:0]	**JEXTSEL[2:0]**：选择启动注入通道组转换的外部事件(External event select for injected group) 这些位选择用于启动注入通道组转换的外部事件。 ADC1 和 ADC2 的触发配置如下 000：定时器 1 的 TRGO 事件　100：定时器 3 的 CC4 事件 001：定时器 1 的 CC4 事件　101：定时器 4 的 TRGO 事件 010：定时器 2 的 TRGO 事件　110：EXTI 线 15/TIM8_CC4 事件 011：定时器 2 的 CC1 事件　111：JSWSTART ADC3 的触发配置如下 000：定时器 1 的 TRGO 事件　100：定时器 8 的 CC4 事件 001：定时器 1 的 CC4 事件　101：定时器 5 的 TRGO 事件 010：定时器 4 的 CC3 事件　110：定时器 5 的 CC4 事件 011：定时器 8 的 CC2 事件　111：JSWSTART
11	ALIGN	**ALIGN**：数据对齐(Data alignment) 该位由软件设置和清除，参考图 27 和图 28。 0：右对齐； 1：左对齐。
10:9	Reserved	保留。必须保持为 0。
8	DMA	**DMA**：直接存储器访问模式(Direct memory access mode) 该位由软件设置和清除。详见 DMA 控制器章节。 0：不使用 DMA 模式； 1：使用 DMA 模式。 注：只有 ADC1 和 ADC3 能产生 DMA 请求。
7:4	Reserved	保留。必须保持为 0。
3	RSTCAL	**RSTCAL**：复位校准(Reset calibration) 该位由软件设置并由硬件清除。在校准寄存器被初始化后该位将被清除。 0：校准寄存器已初始化； 1：初始化校准寄存器。 注：如果正在进行转换时设置 RSTCAL，清除校准寄存器需要额外的周期。
2	CAL	**CAL**：A/D 校准(A/D Calibration) 该位由软件设置以开始校准，并在校准结束时由硬件清除。 0：校准完成； 1：开始校准。
1	CONT	**CONT**：连续转换(Continuous conversion) 该位由软件设置和清除。如果设置了此位，则转换将连续进行直到该位被清除。 0：单次转换模式； 1：连续转换模式。
0	ADON	**ADON**：开/关 A/D 转换器(A/D converter ON/OFF) 该位由软件设置和清除。当该位为'0'时，写入'1'将把 ADC 从断电模式下唤醒。 当该位为'1'时，写入'1'将启动转换。应用程序需注意，在转换器上电至转换开始有一个延迟 tSTAB，参见图 23。 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。

11.12.4 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
保留								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:24	**Reserved**	保留，始终读为 0。
23:0	**SMPx[2:0]**	**SMPx[2:0]**：选择通道 x 的采样时间(Channel x Sample time selection) 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期 注：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 VREFINT。 ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 Vss。 ADC3 模拟输入通道 14、15、16、17 与 Vss 相连

11.12.5 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
保留		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
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:30	**Reserved**	保留，始终读为 0。
29:0	**SMPx[2:0]**	**SMPx[2:0]**：选择通道 x 的采样时间(Channel x Sample time selection) 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000：1.5 周期　　100：41.5 周期 001：7.5 周期　　101：55.5 周期 010：13.5 周期　　110：71.5 周期 011：28.5 周期　　111：239.5 周期 注：ADC3 模拟输入通道 9 与 Vss 相连

11.12.6 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
保留															

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

位	符号	说明
31:12	Reserved	保留，始终读为 0。
11:0	JOFFSETx [11:0]	**JOFFSETx[11:0]**：注入通道 x 的数据偏移(Data offset for injected channel x) 当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。转换的结果可以在 ADC_JDRx 寄存器中读出。

11.12.7 ADC 看门狗高阀值寄存器(ADC_HTR)

地址偏移：0x24

复位值：0x0000 0000

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

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

位	符号	说明
31:12	Reserved	保留，始终读为 0。
11:0	HT[11:0]	**HT[11:0]**：模拟看门狗高阀值(Analog watchdog high threshold) 这些位定义了模拟看门狗的阀值高限。

注意：当 ADC 转换正在进行时，该软件可以写入这些寄存器。在下一个转换完成时，编程值将生效。写入此寄存器时具有写延迟，这可能会对新值编程的有效时间产生不确定性。

11.12.8 ADC 看门狗低阀值寄存器(ADC_LRT)

地址偏移：0x28

复位值：0x0000 0000

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

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

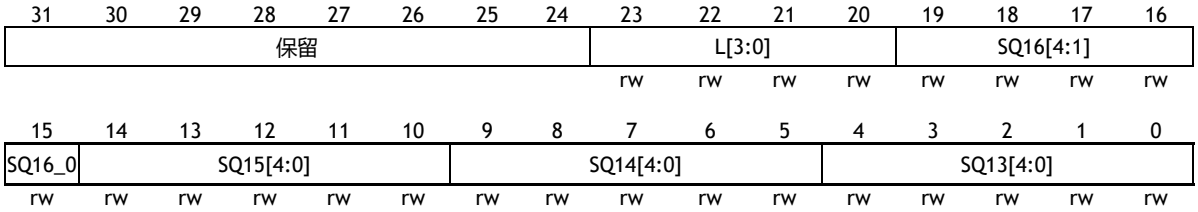
位	符号	说明
31:12	Reserved	保留，始终读为 0。
11:0	LT[11:0]	**LT[11:0]**：模拟看门狗低阀值(Analog watchdog low threshold)这些位定义了模拟看门狗的阀值低限。

注意：当 ADC 转换正在进行时，该软件可以写入这些寄存器。在下一个转换完成时，编程值将生效。写入此寄存器时具有写延迟，这可能会对新值编程的有效时间产生不确定性。

11.12.9 ADC 规则序列寄存器 1(ADC_SQR1)

地址偏移：0x2C

复位值：0x0000 0000

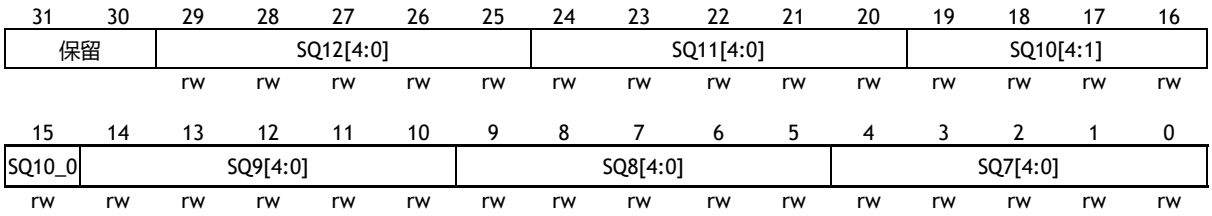


位	符号	说明
31:24	Reserved	保留，始终读为 0。
23:20	L[3:0]	**L[3:0]**：规则通道序列长度(Regular channel sequence length) 这些位由软件定义在规则通道转换序列中的通道数目。 0000：1 个转换 0001：2 个转换 ..... 1111：16 个转换
19:15	SQ16[4:0]	**SQ16[4:0]**：规则序列中的第 16 个转换(16th conversion in regular sequence)这些位由软件定义转换序列中的第 16 个转换通道的编号(0~17)。
14:10	SQ15[4:0]	**SQ15[4:0]**：规则序列中的第 15 个转换(15th conversion in regular sequence)
9:5	SQ14[4:0]	**SQ14[4:0]**：规则序列中的第 14 个转换(14th conversion in regular sequence)
4:0	SQ13[4:0]	**SQ13[4:0]**：规则序列中的第 13 个转换(13th conversion in regular sequence)

11.12.10ADC 规则序列寄存器 2(ADC_SQR2)

地址偏移：0x30

复位值：0x0000 0000



位	符号	说明
31:24	Reserved	保留，始终读为 0。
29:25	SQ12[4:0]	**SQ12[4:0]**：规则序列中的第 12 个转换(12th conversion in regular sequence)这些位由软件定义转换序列中的第 12 个转换通道的编号(0~17)。
24:20	SQ11[4:0]	**SQ11[4:0]**：规则序列中的第 11 个转换(11th conversion in regular sequence)
19:15	SQ10[4:0]	**SQ10[4:0]**：规则序列中的第 10 个转换(10th conversion in regular sequence)
14:10	SQ9[4:0]	**SQ9[4:0]**：规则序列中的第 9 个转换(9th conversion in regular sequence)
9:5	SQ8[4:0]	**SQ8[4:0]**：规则序列中的第 8 个转换(82th conversion in regular sequence)
4:0	SQ7[4:0]	**SQ7[4:0]**：规则序列中的第 7 个转换(7th conversion in regular sequence)

11.12.11ADC 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
保留		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

位	符号	说明
31:30	Reserved	保留，始终读为 0。
29:25	SQ6[4:0]	**SQ6[4:0]**：规则序列中的第 6 个转换(6th conversion in regular sequence)这些位由软件定义转换序列中的第 6 个转换通道的编号(0~17)。
24:20	SQ5[4:0]	**SQ5[4:0]**：规则序列中的第 5 个转换(5th conversion in regular sequence)
19:15	SQ4[4:0]	**SQ4[4:0]**：规则序列中的第 4 个转换(4th conversion in regular sequence)
14:10	SQ3[4:0]	**SQ3[4:0]**：规则序列中的第 3 个转换(3rd conversion in regular sequence)
9:5	SQ2[4:0]	**SQ2[4:0]**：规则序列中的第 2 个转换(2nd conversion in regular sequence)
4:0	SQ1[4:0]	**SQ1[4:0]**：规则序列中的第 1 个转换(1st conversion in regular sequence)

11.12.12ADC 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
保留										JL[3:0]		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

位	符号	说明
31:22	Reserved	保留，始终读为 0。
21:20	JL[1:0]	**JL[1:0]**：注入通道序列长度(Injected sequence length) 这些位由软件定义在规则通道转换序列中的通道数目。 00：1 个转换 01：2 个转换 10：3 个转换 11：4 个转换
19:15	JSQ4[4:0]	**JSQ4[4:0]**：注入序列中的第 4 个转换(4th conversion in injected sequence)这些位由软件定义转换序列中的第 4 个转换通道的编号(0~17)。 注：不同于规则转换序列，如果 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]=10000110001100111000010，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。
14:10	JSQ3[4:0]	**JSQ3[4:0]**：注入序列中的第 3 个转换(3rd conversion in injected sequence)
9:5	JSQ2[4:0]	**JSQ2[4:0]**：注入序列中的第 2 个转换(2nd conversion in injected sequence)
4:0	JSQ1[4:0]	**JSQ1[4:0]**：注入序列中的第 1 个转换(1st conversion in injected sequence)

11.12.13ADC 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
保留															
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

位	符号	说明
31:16	Reserved	保留，始终读为 0。
21:20	JDATA[15:0]	**JDATA[15:0]**：注入转换的数据(Injected data) 这些位为只读，包含了注入通道的转换结果。数据是左对齐或右对齐，如图 27 和图 28 所示

11.12.14ADC 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

位	符号	说明
31:16	ADC2 DATA[15:0]	**ADC2DATA[15:0]**：ADC2 转换的数据(ADC2 data) -在 ADC1 中：双模式下，这些位包含了 ADC2 转换的规则通道数据。见 11.9：双 ADC 模式 -在 ADC2 和 ADC3 中：不使用这些位。
21:20	DATA[15:0]	**DATA[15:0]**：规则转换的数据(Regular data) 这些位为只读，包含了规则通道的转换结果。数据是左对齐或右对齐，如图 27 和图 28 所示。

11.12.15ADC 寄存器地址映像

下表列出了所有的 ADC 寄存器。

表 67 ADC 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	ADC_SR	保留																											STRT	JSTRT	JEOC	EOC	AWD
	复位值																												0	0	0	0	0
04h	ADC_CR1	保留								AWDEN	JAWDEN	保留		DUALMOD [3:0]				DISCNUM [2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
	复位值									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	ADC_CR2	保留								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]			保留	JEXTTRIG	JEXTSEL [2:0]			ALIGN	保留		DMA	保留			RSTCAL	CAL	CONT	ADON
	复位值									0	0	0	0	0	0	0		0	0	0	0	0			0				0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0Ch	ADC_SMPR1	采样时间位SMPx_x																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	ADC_SMPR2	采样时间位SMPx_x																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	ADC_JOFR1	保留																						JOFFSET1[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
18h	ADC_JOFR2	保留																						JOFFSET2[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
1Ch	ADC_JOFR3	保留																						JOFFSET3[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
20h	ADC_JOFR4	保留																						JOFFSET4[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
1Ch	ADC_HTR	保留																						HT[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
20h	ADC_LTR	保留																						LT[11:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
2Ch	ADC_SQR1	保留								L[3:0]				规则通道序列SQx_x位																			
	复位值									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30h	ADC_SQR2	保留	规则通道序列SQx_x位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34h	ADC_SQR3	保留	规则通道序列SQx_x位																														
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38h	ADC_JSQR	保留											JL[1:0]	注入通道序列JSQx_x位																			
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3Ch	ADC_JDR1	保留																						JDATA[15:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
40h	ADC_JDR2	保留																						JDATA[15:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
44h	ADC_JDR3	保留																						JDATA[15:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
48h	ADC_JDR4	保留																						JDATA[15:0]									
	复位值																							0	0	0	0	0	0	0	0	0	0
4Ch	ADC_DR	ADC2DATA[15:0]																规则DATA[15:0]															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

12 数字/模拟转换(DAC)

12.1 DAC 简介

数字/模拟转换模块(DAC)是 12 位数字输入，电压输出的数字/模拟转换器。DAC 可以配置为 8 位或 12 位模式，也可以与 DMA 控制器配合使用。DAC 工作在 12 位模式时，数据可以设置成左对齐或右对齐。DAC 模块有 2 个输出通道，每个通道都有单独的转换器。在双 DAC 模式下，2 个通道可以独立地进行转换，也可以同时进行转换并同步地更新 2 个通道的输出。DAC 可以通过引脚输入参考电压 VREF+以获得更精确的转换结果。

12.2 DAC 主要特征

- 2 个 DAC 转换器：每个转换器对应 1 个输出通道
- 8 位或者 12 位单调输出
- 12 位模式下数据左对齐或者右对齐
- 同步更新功能
- 噪声波形生成
- 三角波形生成
- 双 DAC 通道同时或者分别转换
- 每个通道都有 DMA 功能
- 外部触发转换
- 输入参考电压 VREF+
- 单个 DAC 通道的框图如下图，表 68 给出了引脚的说明。

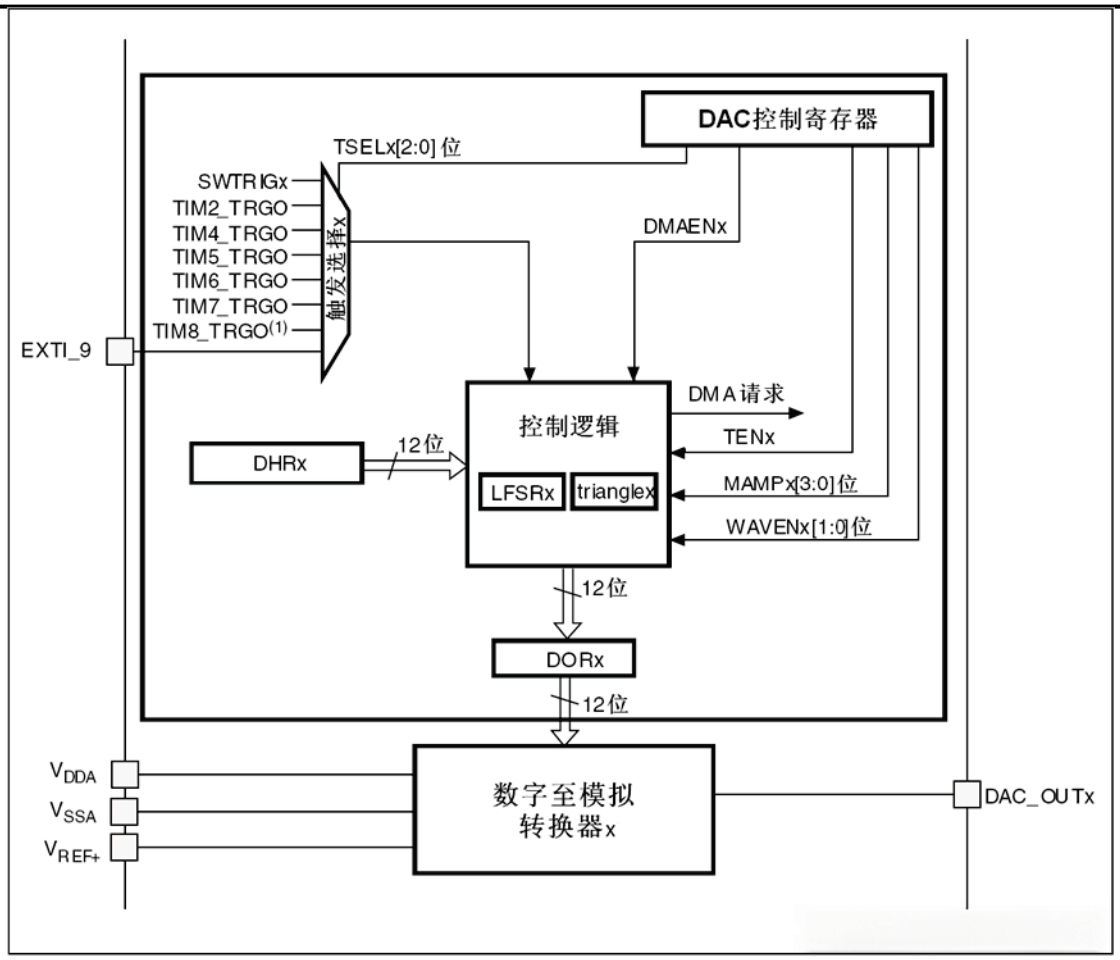


图 40 DAC 通道模块框图

表 68 DAC 引脚

名称	型号类型	注释
VREF+	输入，正模拟参考电压	DAC 使用的高端/正极参考电压，2.4V≤VREF+≤VDDA(3.3V)
VDDA	输入，模拟电源	模拟电源
VSSA	输入，模拟电源地	模拟电源的地线
DAC_OUTx	模拟输出信号	DAC 通道 x 的模拟输出

注意：一旦使能 DACx 通道，相应的 GPIO 引脚(PA4 或者 PA5)就会自动与 DAC 的模拟输出相连(DAC_OUTx)。为了避免寄生的干扰和额外的功耗，引脚 PA4 或者 PA5 在之前应当设置成模拟输入(AIN)。

12.3 DAC 功能描述

12.3.1 使能 DAC 通道

将 DAC_CR 寄存器的 ENx 位置'1'即可打开对 DAC 通道 x 的供电。经过一段启动时间 tWAKEUP，DAC 通道 x 即被使能。

注意：ENx 位只会使能 DAC 通道 x 的模拟部分，即便该位被置'0'，DAC 通道 x 的数字部分仍然工作。

12.3.2 使能 DAC 输出缓存

DAC 集成了 2 个输出缓存，可以用来减少输出阻抗，无需外部运放即可直接驱动外部负载。每个 DAC 通道输出缓存可以通过设置 DAC_CR 寄存器的 BOFFx 位来使能或者关闭。

12.3.3 DAC 数据格式

根据选择的配置模式，数据按照下文所述写入指定的寄存器：

- 单 DAC 通道 x，有 3 种情况：
  - 8 位数据右对齐：用户须将数据写入寄存器 DAC_DHR8Rx[7:0]位(实际是存入寄存器 DHRx[11:4]位)
  - 12 位数据左对齐：用户须将数据写入寄存器 DAC_DHR12Lx[15:4]位(实际是存入寄存器 DHRx[11:0]位)
  - 12 位数据右对齐：用户须将数据写入寄存器 DAC_DHR12Rx[11:0]位(实际是存入寄存器 DHRx[11:0]位)

根据对 DAC_DHRyyyx 寄存器的操作，经过相应的移位后，写入的数据被转存到 DHRx 寄存器中(DHRx 是内部的数据保存寄存器 x)。随后，DHRx 寄存器的内容或被自动地传送到 DORx 寄存器，或通过软件触发或外部事件触发被传送到 DORx 寄存器。

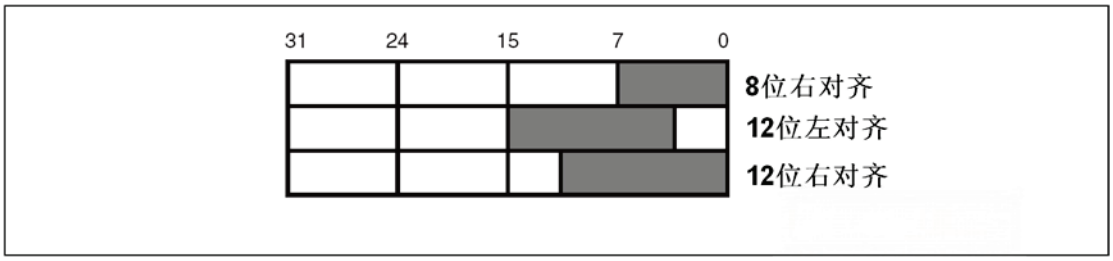


图 41 单 DAC 通道模式的数据寄存器

- 双 DAC 通道，有 3 种情况：
  - 8 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR8RD[7:0]位(实际是存入寄存器 DHR1[11:4]位)，将 DAC 通道 2 数据写入寄存器 DAC_DHR8RD[15:8]位(实际是存入寄存器 DHR2[11:4]位)
  - 12 位数据左对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12LD[15:4]位(实际是存入寄存器 DHR1[11:0]位)，将 DAC 通道 2 数据写入寄存器 DAC_DHR12LD[31:20]位(实际是存入寄存器 DHR2[11:0]位)
  - 12 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12RD[11:0]位(实际是存入寄存器 DHR1[11:0]位)，将 DAC 通道 2 数据写入寄存器 DAC_DHR12RD[27:16]位(实际是存入寄存器 DHR2[11:0]位)

根据对 DAC_DHRyyyD 寄存器的操作，经过相应的移位后，写入的数据被转存到 DHR1 和 DHR2 寄存器中(DHR1 和 DHR2 是内部的数据保存寄存器 x)。随后，DHR1 和 DHR2 的内容或被自动地传送到 DORx 寄存器，或通过软件触发或外部事件触发被传送到 DORx 寄存器。

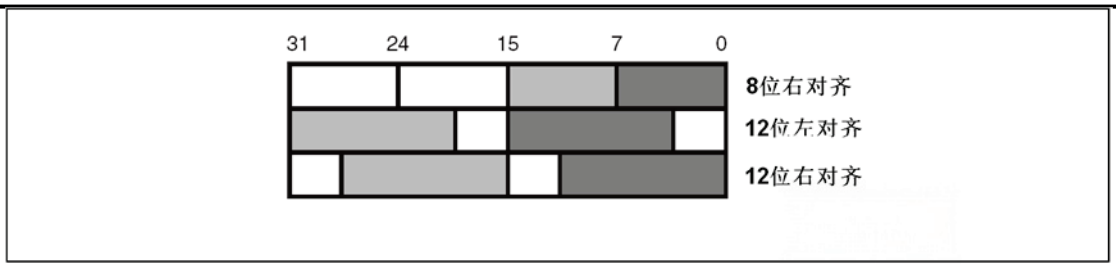


图 42 双 DAC 通道模式的数据寄存器

12.3.4 DAC 转换

不能直接对寄存器 DAC_DORx 写入数据，任何输出到 DAC 通道 x 的数据都必须写入 DAC_DHRx 寄存器(数据实际写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12LD、或者 DAC_DHR12RD 寄存器)。

如果没有选中硬件触发(寄存器 DAC_CR1 的 TENx 位置'0')，存入寄存器 DAC_DHRx 的数据会在一个 APB1 时钟周期后自动传至寄存器 DAC_DORx。如果选中硬件触发(寄存器 DAC_CR1 的 TENx 位置'1')，数据传输在触发发生以后 3 个 APB1 时钟周期后完成。

一旦数据从 DAC_DHRx 寄存器装入 DAC_DORx 寄存器，在经过时间 tSETTLING 之后，输出即有效，这段时间的长短依电源电压和模拟输出负载的不同会有所变化。

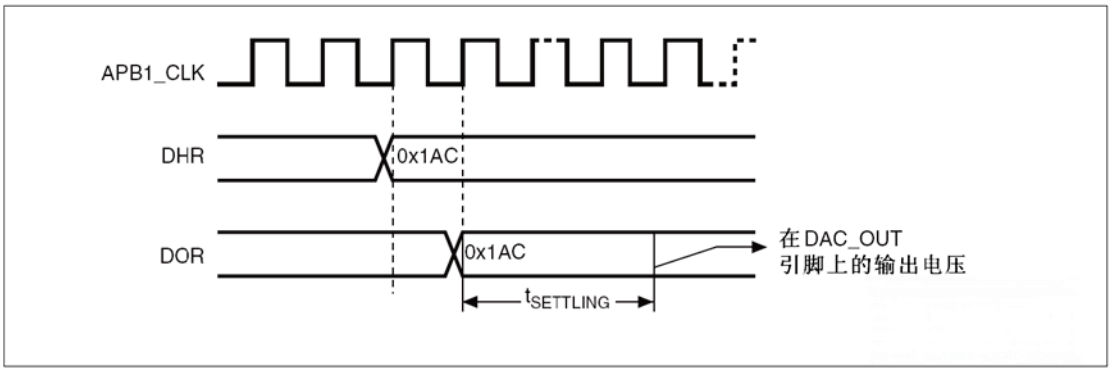


图 43 TEN=0 触发失能时转换的时间框图

12.3.5 DAC 输出电压

数字输入经过 DAC 被线性地转换为模拟电压输出，其范围为 0 到 VREF+。

任一 DAC 通道引脚上的输出电压满足下面的关系：

DAC 输出=VREFx(DOR/4095)。

12.3.6 选择 DAC 触发

如果 TENx 位被置 1，DAC 转换可以由某外部事件触发(定时器计数器、外部中断线)。配置控制位 TSELx[2:0]可以选择 8 个触发事件之一触发 DAC 转换。

表 69 外部触发

触发源	类型	TSELx[2:0]
定时器 6TRGO 事件	来自片上定时器的内部信号	000
定时器 8TRGO 事件		001
定时器 7TRGO 事件		010
定时器 5TRGO 事件		011
定时器 2TRGO 事件		100

定时器 4TRGO 事件		101
EXTI 线路 9	外部引脚	110
SWTRIG(软件触发)	软件控制位	111

每次 DAC 接口侦测到来自选中的定时器 TRGO 输出，或者外部中断线 9 的上升沿，最近存放在寄存器 DAC_DHRx 中的数据会被传送到寄存器 DAC_DORx 中。在 3 个 APB1 时钟周期后，寄存器 DAC_DORx 更新为新值。

如果选择软件触发，一旦 SWTRIG 位置'1'，转换即开始。在数据从 DAC_DHRx 寄存器传送到 DAC_DORx 寄存器后，SWTRIG 位由硬件自动清'0'。

注意： *1.不能在 ENx 为1'时改变 TSELx[2:0]位。*

2.如果选择软件触发，数据从寄存器 DAC_DHRx 传送到寄存器 DAC_DORx 只需要一个 APB1 时钟周期。

12.3.7 DMA 请求

任一 DAC 通道都具有 DMA 功能。2 个 DMA 通道可分别用于 2 个 DAC 通道的 DMA 请求。

如果 DMAENx 位置'1'，一旦有外部触发(而不是软件触发)发生，则产生一个 DMA 请求，然后 DAC_DHRx 寄存器的数据被传送到 DAC_DORx 寄存器。

在双 DAC 模式下，如果 2 个通道的 DMAENx 位都为'1'，则会产生 2 个 DMA 请求。如果实际只需要一个 DMA 传输，则应只选择其中一个 DMAENx 位置'1'。这样，程序可以在只使用一个 DMA 请求，一个 DMA 通道的情况下，处理工作在双 DAC 模式的 2 个 DAC 通道。

DAC 的 DMA 请求不会累计，因此如果第 2 个外部触发发生在响应第 1 个外部触发之前，则不能处理第 2 个 DMA 请求，也不会报告错误。

12.3.8 噪声生成

可以利用线性反馈移位寄存器(Linear Feedback Shift Register LFSR)产生幅度变化的伪噪声。设置 WAVE[1:0]位为'01'选择 DAC 噪声生成功能。寄存器 LFSR 的预装入值为 0xAAA。按照特定算法，在每次触发事件后 3 个 APB1 时钟周期之后更新该寄存器的值。

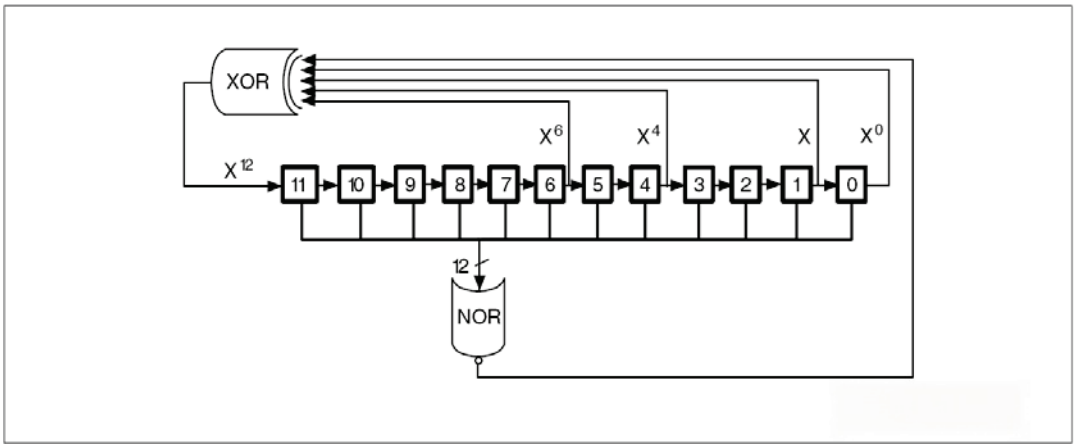


图 44 DACLFSR 寄存器算法

设置 DAC_CR 寄存器的 MAMPx[3:0]位可以屏蔽部分或者全部 LFSR 的数据，这样的得到的 LSFR 值与 DAC_DHRx 的数值相加，去掉溢出位之后即被写入 DAC_DORx 寄存器。

如果寄存器 LFSR 值为 0x000，则会注入'1'(防锁定机制)。将 WAVEx[1:0]位置'0'可以复位 LFSR 波形的生成算法。

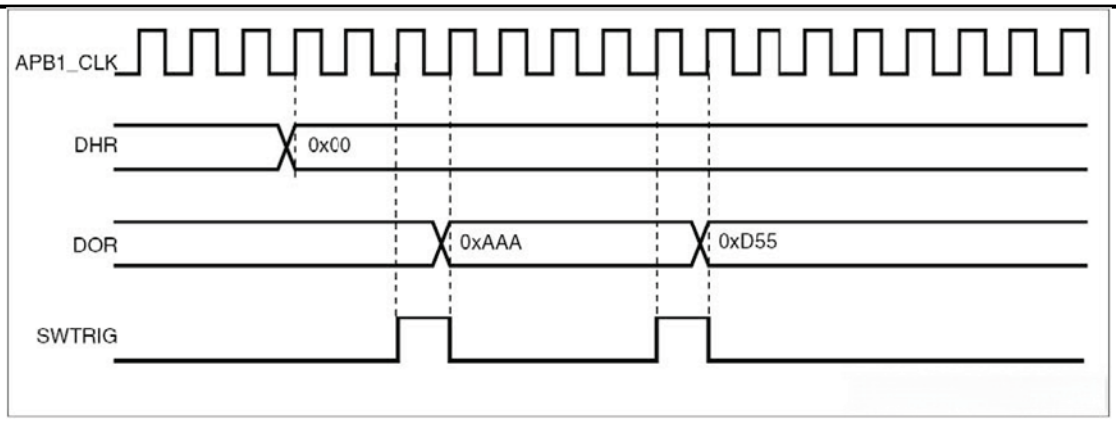


图 45 带 LFSR 波形生成的 DAC 转换(使能软件触发)

注意： 为了产生噪声，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为1'。

12.3.9 三角波生成

可以在 DC 或者缓慢变化的信号上加上一个小幅度的三角波。设置 WAVEx[1:0]位为'10'选择 DAC 的三角波生成功能。设置 DAC_CR 寄存器的 MAMPx[3:0]位来选择三角波的幅度。内部的三角波计数器每次触发事件之后 3 个 APB1 时钟周期后累加 1。计数器的值与 DAC_DHRx 寄存器的数值相加并丢弃溢出位后写入 DAC_DORx 寄存器。在传入 DAC_DORx 寄存器的数值小于 MAMP[3:0]位定义的最大幅度时，三角波计数器逐步累加。一旦达到设置的最大幅度，则计数器开始递减，达到 0 后再开始累加，周而复始。

将 WAVEx[1:0]位置'0'可以复位三角波的生成。

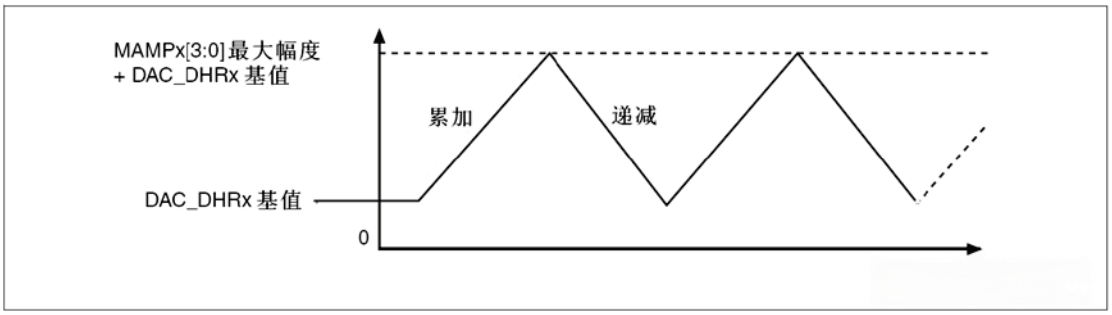


图 46 DAC 三角波生成

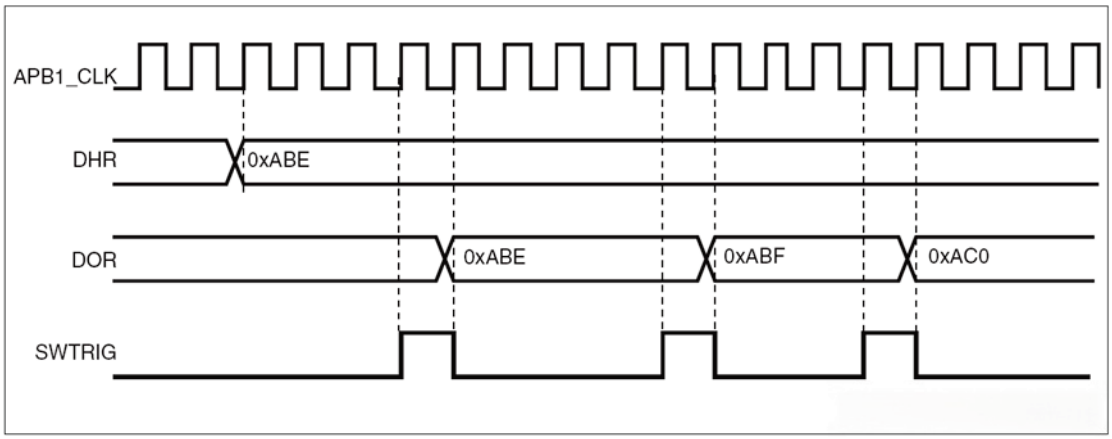


图 47 带三角生成的 DAC 转换(使能软件触发)

注意： 1.为了产生三角波，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为1'。

2.MAMP[3:0]位必须在使能 DAC 之前设置，否则其值不能修改。

12.4 双 DAC 通道转换

在需要 2 个 DAC 同时工作的情况下，为了更有效地利用总线带宽，DAC 集成了 3 个供双 DAC 模式使用的寄存器：DHR8RD、DHR12RD 和 DHR12LD，只需要访问一个寄存器即可完成同时驱动 2 个 DAC 通道的操作。

对于双 DAC 通道转换和这些专用寄存器，共有 11 种转换模式可用。这些转换模式在只使用一个 DAC 通道的情况下，仍然可通过独立的 DHRx 寄存器操作。

所有模式详述于以下章节。

12.4.1 不使用波形发生器的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，(延迟 3 个 APB1 时钟周期后)寄存器 DHR1 的值传入寄存器 DAC_DOR1。

当发生 DAC 通道 2 触发事件时，(延迟 3 个 APB1 时钟周期后)寄存器 DHR2 的值传入寄存器 DAC_DOR2。

12.4.2 使用相同 LFSR 的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设置 MAMPx[3:0]为相同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。

当发生 DAC 通道 2 触发事件时，具有相同屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.4.3 使用不同 LFSR 的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设 MAMPx[3:0]为不同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或者 DHR8RD)。

当发生 DAC 通道 1 触发事件时，按照 MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。

当发生 DAC 通道 2 触发事件时，按照 MAMP2[3:0]所设屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.4.4 产生相同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为”1x”，并设 MAMPx[3:0]为相同的三角波幅值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，相同的三角波幅值加上 DHR1 寄存器的值，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 DAC 通道 1 三角波计数器。

当发生 DAC 通道 2 触发事件时，相同的三角波幅值加上 DHR2 寄存器的值，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 DAC 通道 2 三角波计数器。

12.4.5 产生不同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为'1x'，并设 MAMPx[3:0]为不同的三角波幅值。
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生 DAC 通道 1 触发事件时，MAMP1[3:0]所设的三角波幅值加上 DHR1 寄存器数值，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 DAC 通道 1 三角波计数器。

当发生 DAC 通道 2 触发事件时，MAMP2[3:0]所设的三角波幅值加上 DHR2 寄存器数值，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 DAC 通道 2 三角波计数器。

12.4.6 同时软件启动

按照下列过程设置 DAC 工作在此转换模式：

- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

在此配置下，一个 APB1 时钟周期后，DHR1 和 DHR2 寄存器的数值即被分别传入 DAC_DOR1 和 DAC_DOR2 寄存器。

12.4.7 不使用波形发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，(延迟 3 个 APB1 时钟周期后)DHR1 和 DHR2 寄存器的数值分别传入 DAC_DOR1 和 DAC_DOR2 寄存器。

12.4.8 使用相同 LFSR 的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；

- 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设 MAMPx[3:0]为相同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)；

当发生触发事件时，MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DHR1 寄存器的数值相加，(延迟 3 个 APB1 时钟周期后)结果传入 DAC_DOR1 寄存器，然后更新 LFSR1 计数器。

同样，MAMP1[3:0]所设屏蔽的 LFSR2 计数器值与 DHR2 寄存器的数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.4.9 使用不同 LFSR 的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'；
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源；
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为'01'，并设 MAMPx[3:0]为不同的 LFSR 屏蔽值；
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。

同时，具有相同屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.4.10 使用相同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'：
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为'1x'，并设 MAMPx[3:0]为相同的三角波幅值。
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，相同的三角波幅值与 DHR1 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。

同时，相同的三角波幅值与 DHR2 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.4.11 使用不同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'：
- 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
- 设置 2 个 DAC 通道的 WAVEx[1:0]位为'1x'，并设 MAMPx[3:0]为不同的三角波幅值。
- 将双 DAC 通道转换数据装入所需的 DHR 寄存器(DHR12RD、DHR12LD 或 DHR8RD)。

当发生触发事件时，MAMP1[3:0]所设的三角波幅值与 DHR1 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。

同时，MAMP2[3:0]所设的三角波幅值与 DHR2 寄存器数值相加，(延迟 3 个 APB1 时钟周期后)结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

12.5 DAC 寄存器

必须以字(32 位)的方式操作这些外设寄存器。

12.5.1 DAC 控制寄存器(DAC_CR)

地址偏移：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留			DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			DMAEN1	MAMP1[3:0]				WAVE1[2:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:29	Reserved	保留，始终读为 0。
28	DMAEN2	**DMAEN2**：DAC 通道 2DMA 使能(DAC channel2 DMA enable) 该位由软件设置和清除。 0：关闭 DAC 通道 2DMA 模式； 1：使能 DAC 通道 2DMA 模式。
27:24	MAMP2[3:0]	**MAMP2[3:0]**：DAC 通道 2 屏蔽/幅值选择器(DAC channel2 mask/amplitude selector) 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值。 0000：不屏蔽 LSFR 位 0/三角波幅值等于 1； 0001：不屏蔽 LSFR 位[1:0]/三角波幅值等于 3； 0010：不屏蔽 LSFR 位[2:0]/三角波幅值等于 7； 0011：不屏蔽 LSFR 位[3:0]/三角波幅值等于 15； 0100：不屏蔽 LSFR 位[4:0]/三角波幅值等于 31； 0101：不屏蔽 LSFR 位[5:0]/三角波幅值等于 63； 0110：不屏蔽 LSFR 位[6:0]/三角波幅值等于 127； 0111：不屏蔽 LSFR 位[7:0]/三角波幅值等于 255； 1000：不屏蔽 LSFR 位[8:0]/三角波幅值等于 511； 1001：不屏蔽 LSFR 位[9:0]/三角波幅值等于 1023； 1010：不屏蔽 LSFR 位[10:0]/三角波幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11:0]/三角波幅值等于 4095。
23:22	WAVE2[1:0]	**WAVE2[1:0]**：DAC 通道 2 噪声/三角波生成使能(DAC channel2 noise/triangle wave generation enable) 该 2 位由软件设置和清除。 00：关闭波形发生器； 10：使能噪声波形发生器； 1x：使能三角波发生器。
21:19	TSEL2[2:0]	**TSEL2[2:0]**：DAC 通道 2 触发选择(DAC channel2 trigger selection) 该 3 位用于选择 DAC 通道 2 的外部触发事件。 000：TIM6TRGO 事件； 001：TIM8TRGO 事件； 010：TIM7TRGO 事件； 011：TIM5TRGO 事件； 100：TIM2TRGO 事件； 101：TIM4TRGO 事件； 110：外部中断线 9；

		111：软件触发。 注意：该 3 位只能在 TEN2=1(DAC 通道 2 触发使能)时设置。
18	**TEN2**	**TEN2**：DAC 通道 2 触发使能(DAC channel2 trigger enable) 该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的触发。 0：关闭 DAC 通道 2 触发，写入 DAC_DHRx 寄存器的数据在 1 个 APB1 时钟周期后传入 DAC_DOR2 寄存器； 1：使能 DAC 通道 2 触发，写入 DAC_DHRx 寄存器的数据在 3 个 APB1 时钟周期后传入 DAC_DOR2 寄存器。 注意：如果选择软件触发，写入寄存器 DAC_DHRx 的数据只需要 1 个 APB1 时钟周期就可以传入寄存器 DAC_DOR2。
17	**BOFF2**	**BOFF2**：关闭 DAC 通道 2 输出缓存(DAC channel2 output buffer disable)该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的输出缓存。 0：使能 DAC 通道 2 输出缓存； 1：关闭 DAC 通道 2 输出缓存。
16	**EN2**	**EN2**：DAC 通道 2 使能(DAC channel2 enable) 该位由软件设置和清除，用来使能/关闭 DAC 通道 2。 0：关闭 DAC 通道 2； 1：使能 DAC 通道 2。
15:13	**Reserved**	保留。
12	**DMAEN1**	**DMAEN1**：DAC 通道 1DMA 使能(DAC channel1 DMA enable) 该位由软件设置和清除。 0：关闭 DAC 通道 1DMA 模式； 1：使能 DAC 通道 1DMA 模式。
11:8	**MAMP1[3:0]**	**MAMP1[3:0]**：DAC 通道 1 屏蔽/幅值选择器(DAC channel1 mask/amplitude selector) 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值。 0000：不屏蔽 LSFR 位 0/三角波幅值等于 1； 0001：不屏蔽 LSFR 位[1:0]/三角波幅值等于 3； 0010：不屏蔽 LSFR 位[2:0]/三角波幅值等于 7； 0011：不屏蔽 LSFR 位[3:0]/三角波幅值等于 15； 0100：不屏蔽 LSFR 位[4:0]/三角波幅值等于 31； 0101：不屏蔽 LSFR 位[5:0]/三角波幅值等于 63； 0110：不屏蔽 LSFR 位[6:0]/三角波幅值等于 127； 0111：不屏蔽 LSFR 位[7:0]/三角波幅值等于 255； 1000：不屏蔽 LSFR 位[8:0]/三角波幅值等于 511； 1001：不屏蔽 LSFR 位[9:0]/三角波幅值等于 1023； 1010：不屏蔽 LSFR 位[10:0]/三角波幅值等于 2047； ≥1011：不屏蔽 LSFR 位[11:0]/三角波幅值等于 4095。
7:6	**WAVE1[1:0]**	**WAVE1[1:0]**：DAC 通道 1 噪声/三角波生成使能(DAC channel1 noise/triangle wave generation enable) 该 2 位由软件设置和清除。 00：关闭波形生成； 10：使能噪声波形发生器； 1x：使能三角波发生器。
5:3	**TSEL1[2:0]**	**TSEL1[2:0]**：DAC 通道 1 触发选择(DAC channel1 trigger selection) 该位用于选择 DAC 通道 1 的外部触发事件。 000：TIM6TRGO 事件； 001：TIM8TRGO 事件； 010：TIM7TRGO 事件； 011：TIM5TRGO 事件； 100：TIM2TRGO 事件；

		101：TIM4TRGO 事件； 110：外部中断线 9； 111：软件触发。 注意：该位只能在 TEN1=1(DAC 通道 1 触发使能)时设置。
2	TEN1	**TEN1**：DAC 通道 1 触发使能(DAC channel1 trigger enable) 该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的触发。 0：关闭 DAC 通道 1 触发，写入寄存器 DAC_DHRx 的数据在 1 个 APB1 时钟周期后传入寄存器 DAC_DOR1； 1：使能 DAC 通道 1 触发，写入寄存器 DAC_DHRx 的数据在 3 个 APB1 时钟周期后传入寄存器 DAC_DOR1。 注意：如果选择软件触发，写入寄存器 DAC_DHRx 的数据只需要 1 个 APB1 时钟周期就可以传入寄存器 DAC_DOR1。
1	BOFF1	**BOFF1**：关闭 DAC 通道 1 输出缓存(DAC channel1 output buffer disable)该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的输出缓存。 0：使能 DAC 通道 1 输出缓存； 1：关闭 DAC 通道 1 输出缓存。
0	EN1	**EN1**：DAC 通道 1 使能(DAC channel1 enable) 该位由软件设置和清除，用来使能/失能 DAC 通道 1。 0：关闭 DAC 通道 1； 1：使能 DAC 通道 1。

12.5.2 DAC 软件触发寄存器(DAC_SWTRIGR)

地址偏移：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														SWTRIG2	SWTRIG1
														w	w

位	符号	说明
31:29	Reserved	保留，始终读为 0。
1	SWTRIG2	**SWTRIG2**：DAC 通道 2 软件触发(DAC channel2 software trigger)该位由软件设置和清除，用来使能/关闭软件触发。 0：关闭 DAC 通道 2 软件触发； 1：使能 DAC 通道 2 软件触发。 注意：一旦寄存器 DAC_DHR2 的数据传入寄存器 DAC_DOR2，(1 个 APB1 时钟周期后)该位由硬件置'0'。
0	SWTRIG1	**SWTRIG1**：DAC 通道 1 软件触发(DAC channel1 software trigger)该位由软件设置和清除，用来使能/关闭软件触发。 0：关闭 DAC 通道 1 软件触发； 1：使能 DAC 通道 1 软件触发。 注意：一旦寄存器 DAC_DHR1 的数据传入寄存器 DAC_DOR1，(1 个 APB1 时钟周期后)该位由硬件置'0'。

12.5.3 DAC 通道 1 的 12 位右对齐数据保持寄存器(DAC_DHR12R1)

地址偏移：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:16	Reserved	保留，始终读为 0。
11:0	DACC1DHR[11:0]	**DACC1DHR[11:0]**：DAC 通道 1 的 12 位右对齐数据(DAC channel1 12-bit right-aligned data) 该位由软件写入，表示 DAC 通道 1 的 12 位数据。

12.5.4 DAC 通道 1 的 12 位左对齐数据保持寄存器(DAC_DHR12L1)

地址偏移：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:4	DACC1DHR [11:0]	**DACC1DHR[11:0]**：DAC 通道 1 的 12 位右对齐数据(DAC channel1 12-bit right-aligned data) 该位由软件写入，表示 DAC 通道 1 的 12 位数据。
3:0	Reserved	保留，始终读为 0。

12.5.5 DAC 通道 1 的 8 位右对齐数据保持寄存器(DAC_DHR8R1)

地址偏移：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:18	Reserved	保留，始终读为 0。
7:0	DACC1DHR[7:0]	**DACC1DHR[7:0]**：DAC 通道 1 的 8 位右对齐数据(DAC channel1 8-bit right-aligned data) 该位由软件写入，表示 DAC 通道 1 的的 8 位数据。

12.5.6 DAC 通道 2 的 12 位右对齐数据保持寄存器(DAC_DHR12R2)

地址偏移：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:12	Reserved	保留，始终读为 0。
11:0	DACC2DHR [11:0]	**DACC2DHR[11:0]**：DAC 通道 2 的 12 位右对齐数据(DAC channel2 12-bit right-aligned data) 该位由软件写入，表示 DAC 通道 2 的 12 位数据。

12.5.7 DAC 通道 2 的 12 位左对齐数据保持寄存器(DAC_DHR12L2)

地址偏移：0x18

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												保留			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:4	DAC2DHR [11:0]	**DAC2DHR[11:0]**：DAC 通道 2 的 12 位左对齐数据(DAC channel2 12-bit left-aligned data) 该位由软件写入，表示 DAC 通道 2 的的 12 位数据。
3:0	Reserved	保留，始终读为 0。

12.5.8 DAC 通道 2 的 8 位右对齐数据保持寄存器(DAC_DHR8R2)

地址偏移：0x1C

复位值：0x0000 0000

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

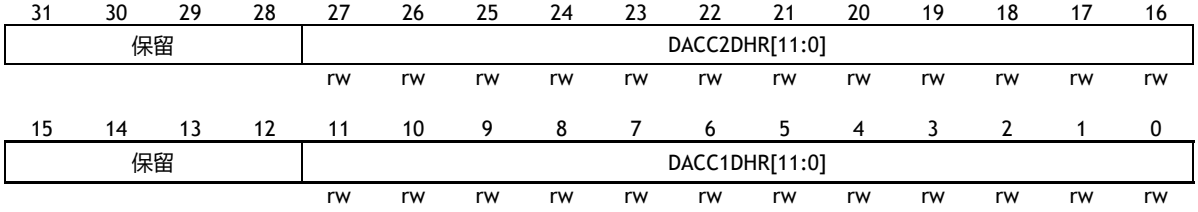
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:18	Reserved	保留，始终读为 0。
7:0	DACC2DHR [7:0]	**DACC2DHR[7:0]**：DAC 通道 2 的 8 位右对齐数据(DAC channel2 8-bit right-aligned data) 该位由软件写入，表示 DAC 通道 2 的的 8 位数据。

12.5.9 双 DAC 的 12 位右对齐数据保持寄存器(DAC_DHR12RD)

地址偏移：0x20

复位值：0x0000 0000

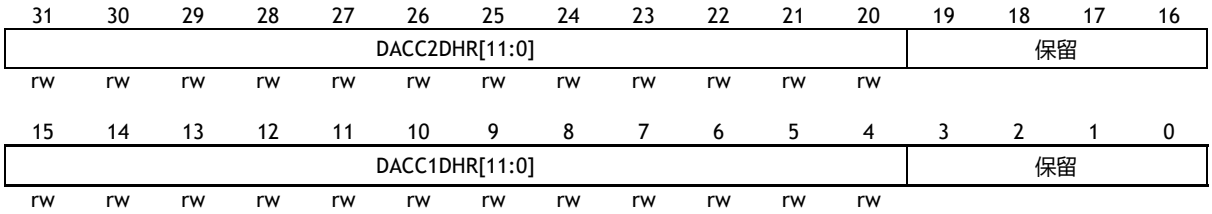


位	符号	说明
31:28	Reserved	保留，始终读为 0。
27:16	DACC2DHR [11:0]	**DACC2DHR [11:0]:**DAC 通道 2 的 12 位右对齐数据 该位由软件写入，表示 DAC 通道 2 的的 12 位数据。
7:0	DACC2DHR [11:0]	**DACC2DHR[11:0]**：DAC 通道 1 的 12 位右对齐数据 该位由软件写入，表示 DAC 通道 1 的的 12 位数据。

12.5.10 双 DAC 的 12 位左对齐数据保持寄存器(DAC_DHR12LD)

地址偏移：0x24

复位值：0x0000 0000



位	符号	说明
31:20	DACC2DHR[11:0]	**DACC2DHR[11:0]**：DAC 通道 2 的 12 位左对齐数据(DAC channel2 12-bit left-aligned data) 该位由软件写入，表示 DAC 通道 2 的 12 位数据。
19:16	Reserved	保留，始终读为 0。
15:4	DACC1DHR[11:0]	**DACC1DHR[11:0]**：DAC 通道 1 的 12 位左对齐数据(DAC channel1 12-bit left-aligned data) 该位由软件写入，表示 DAC 通道 1 的 12 位数据。
3:0	Reserved	保留，始终读为 0。

12.5.11 双 DAC 的 8 位右对齐数据保持寄存器(DAC_DHR8RD)

地址偏移：0x28

复位值：0x0000 0000

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:8	DACC2DHR [7:0]	**DACC2DHR[7:0]**：DAC 通道 2 的 8 位右对齐数据(DAC channel2 8-bit right-aligned data)

		该位由软件写入，表示 DAC 通道 2 的的 8 位数据。
7:0	**DACC1DHR [7:0]**	**DACC1DHR[7:0]**：DAC 通道 1 的 8 位右对齐数据(DAC channel1 8-bit right-aligned data) 该位由软件写入，表示 DAC 通道 1 的的 8 位数据。

12.5.12 DAC 通道 1 数据输出寄存器(DAC_DOR1)

地址偏移：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC1DOR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:12	**Reserved**	保留，始终读为 0。
11:0	**DACC1DOR [11:0]**	**DACC1DOR[11:0]**：DAC 通道 1 输出数据(DAC channel1 data output)该位由软件写入，表示 DAC 通道 1 的输出数据。

12.5.13 DAC 通道 2 数据输出寄存器(DAC_DOR2)

地址偏移：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				DACC2DOR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:12	**Reserved**	保留，始终读为 0。
11:0	**DACC2DOR [11:0]**	**DACC2DOR[11:0]**：DAC 通道 2 输出数据(DAC channel2 data output)该位由软件写入，表示 DAC 通道 2 的输出数据。

12.5.14 DAC 寄存器映像

下表列出了所有 DAC 寄存器。

表 70 DAC 寄存器映像

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	保留			DMAEN2	MAMP2[3:0]				WAVE2 [2:0]		TSEL2 [2:0]			TEN2	BOFF2	EN2	保留			DMAEN1	MAMP13:0]				WAVE1 [2:0]		TSEL1 [2:0]			TEN1	BOFF1	EN1
	复位值				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_SWTRIGR	保留																														SWTRIG2	SWTRIG1
	复位值																															0	0
0x08	DAC_DHR12R1	保留																				DACC1DHR[11:0]											
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	保留																DACC1DHR[11:0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	保留																								DACC1DHR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	保留																				DACC2DHR[11:0]											
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	保留																DACC2DHR[11:0]														保留	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_DHR8R2	保留																								DACC2DHR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x20	DAC_DHR12RD	保留				DACC2DHR[11:0]												保留				DACC1DHR[11:0]											
	复位值					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DACC2DHR[11:0]												保留				DACC1DHR[11:0]														保留	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	DAC_DHR8RD	保留																DACC2DHR[7:0]								DACC1DHR[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	保留																				DACC1DOR[11:0]											
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	保留																				DACC2DOR[11:0]											
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

13 高级控制定时器(TIM1 和 TIM8)

13.1 TIM1 和 TIM8 简介

高级控制定时器(TIM1 和 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，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM 等)。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

高级控制定时器(TIM1 和 TIM8)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看 13.3.20 节。

13.2 TIM1 和 TIM8 主要特性

TIM1 和 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 1～65535 之间的任意数值
- 多达 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

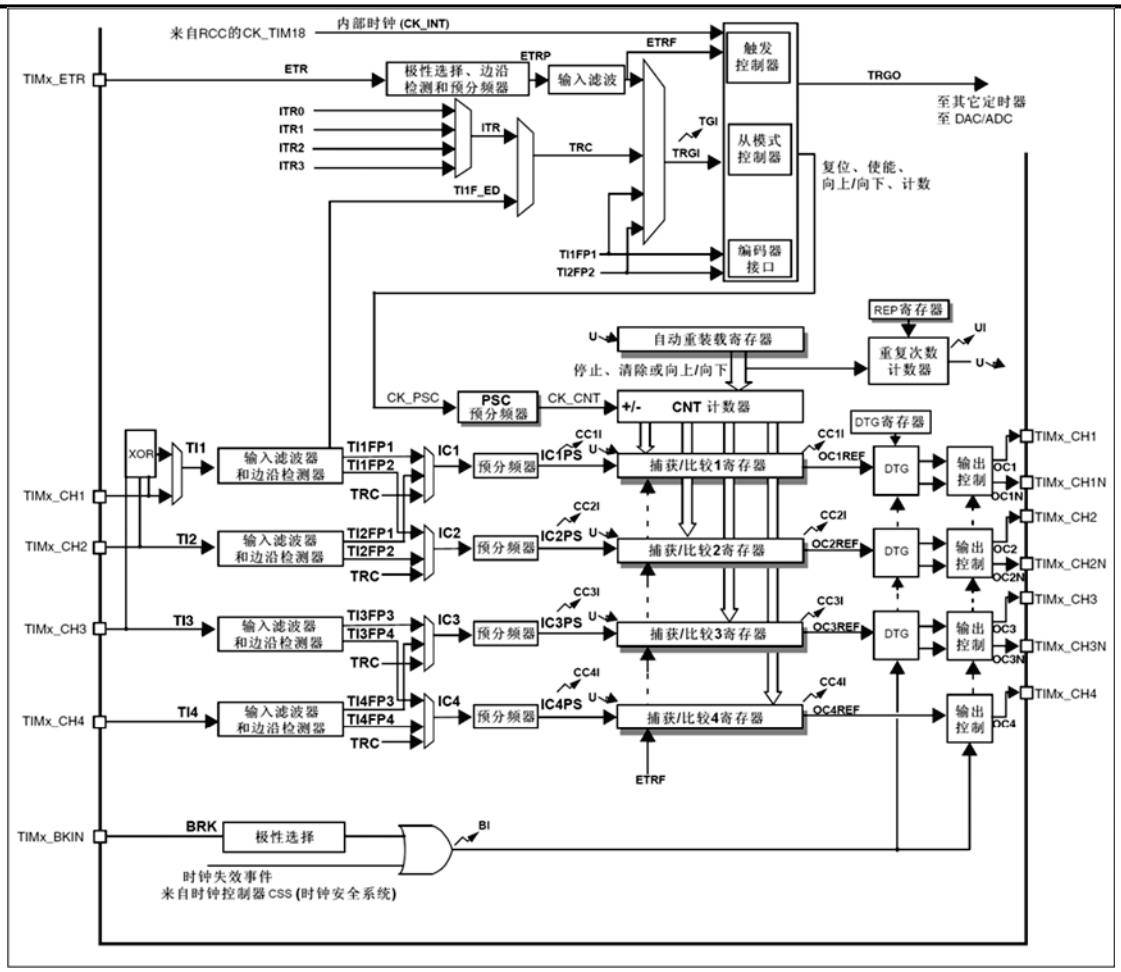


图 48 级控制定时器框图

注： Reg *根据控制位的设定，在 U(更新)事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

13.3 TIM1 和 TIM8 功能描述

13.3.1 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器(TIMx_PSC)
- 自动装载寄存器(TIMx_ARR)
- 重复次数寄存器(TIMx_RCR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIMx_CR1 寄存

器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(更多有关使能计数器的细节，请参见控制器的从模式描述)。

注意：　在设置了 TIMx_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

图 49 和图 50 给出了在预分频器运行时，更改计数器参数的例子。

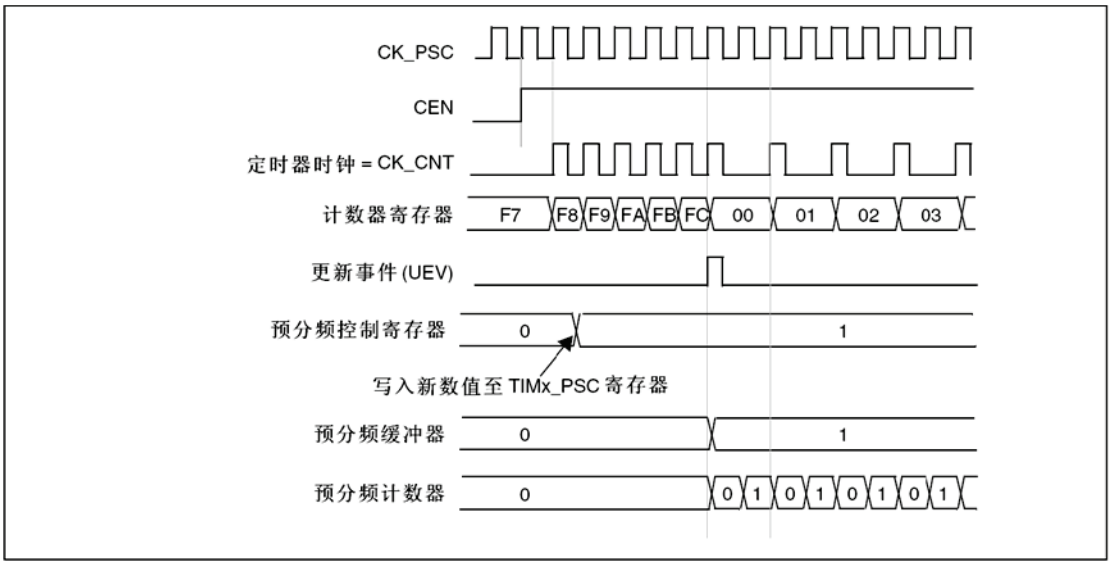


图 49 当预分频器的参数从 1 变到 2 时，计数器的时序图

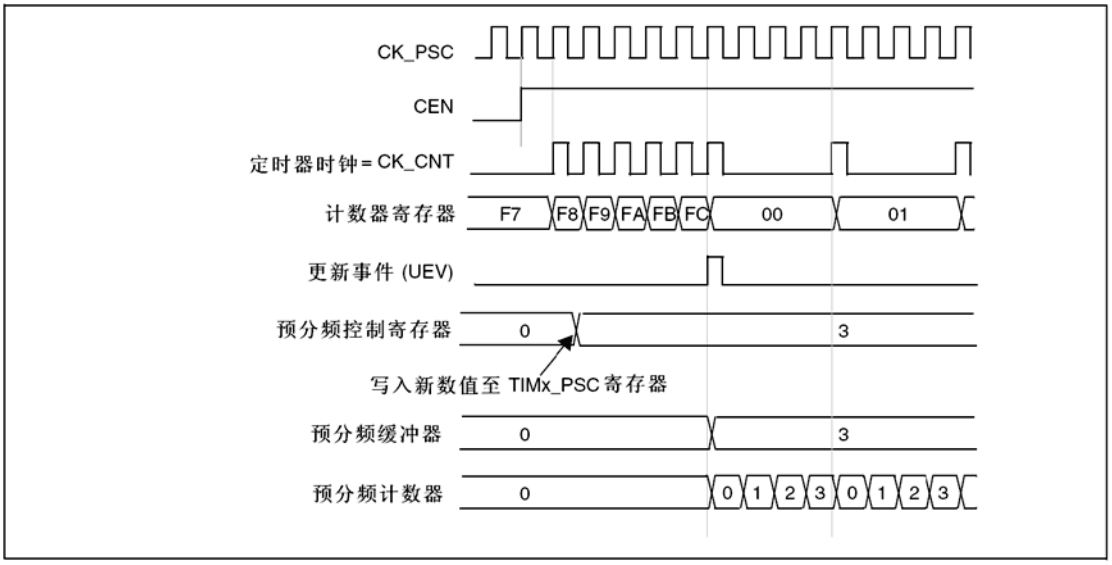


图 50 当预分频器的参数从 1 变到 4 时，计数器的时序图

13.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数(TIMx_RCR)时，产生更新事件(UEV)；否则每次计数器溢出时才产生更新事件。

在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清'0'，同时预分频器的计数也被请 0(但预分频器的数值不变)。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志(即不产生中断或 DMA 请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 重复计数器被重新加载为 TIMx_RCR 寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

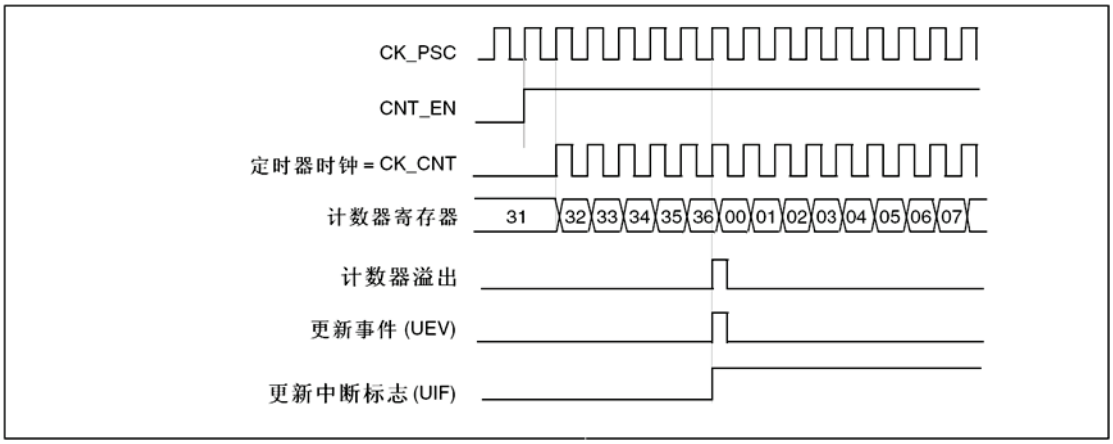


图 51 计数器时序图，内部时钟分频因子为 1

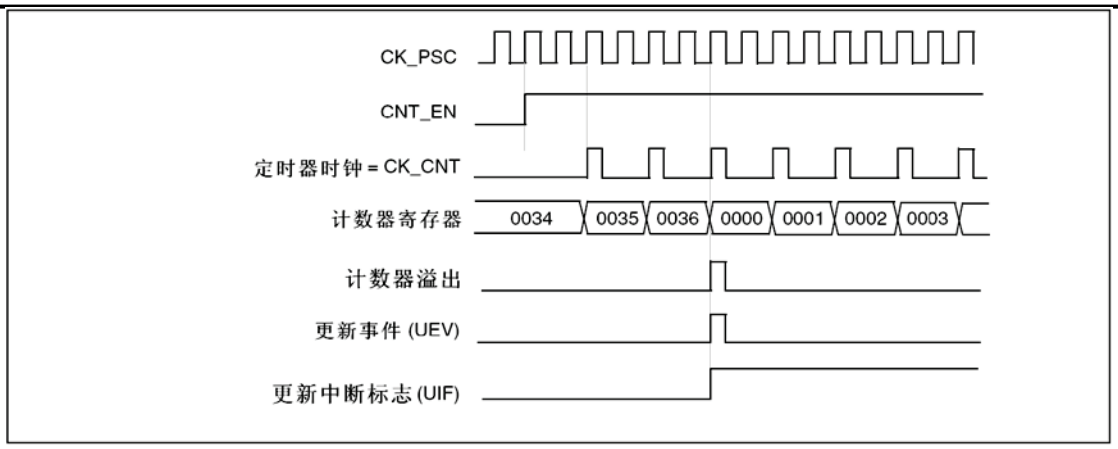


图 52 计数器时序图，内部时钟分频因子为 2

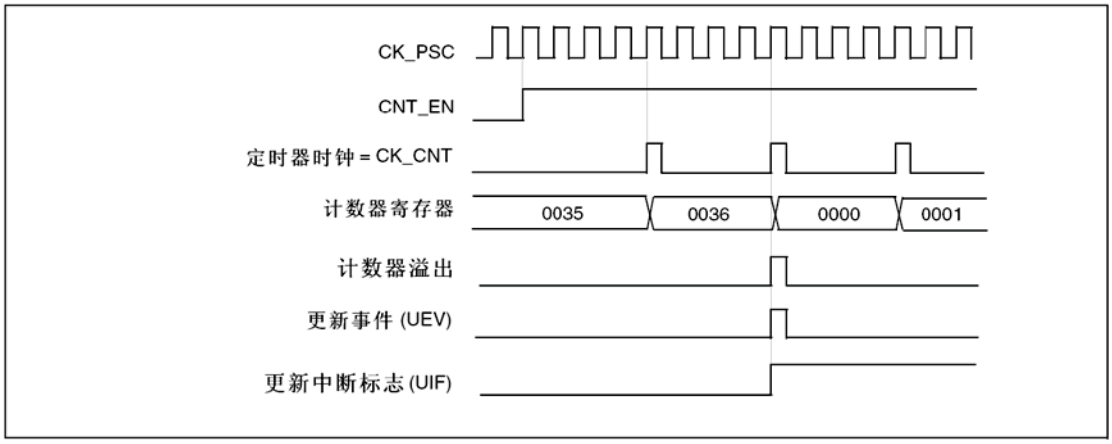


图 53 计数器时序图，内部时钟分频因子为 4

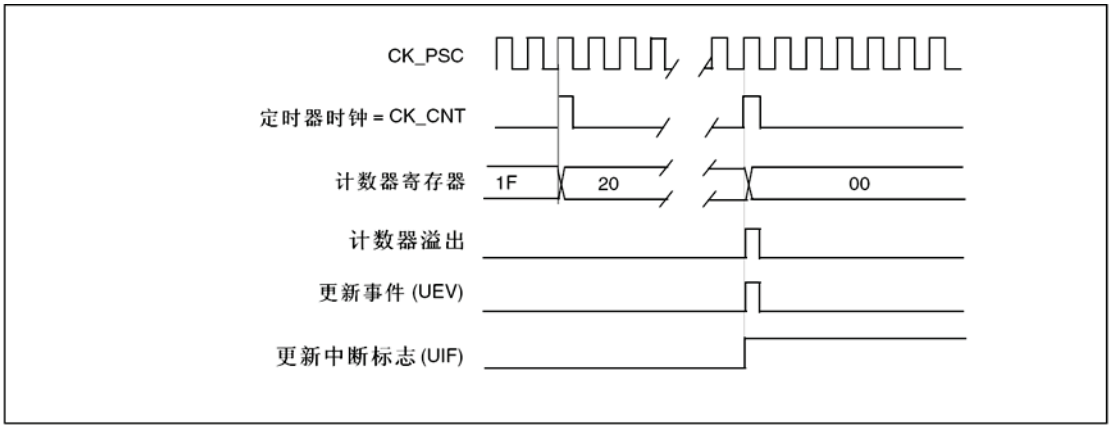


图 54 计数器时序图，内部时钟分频因子为 N

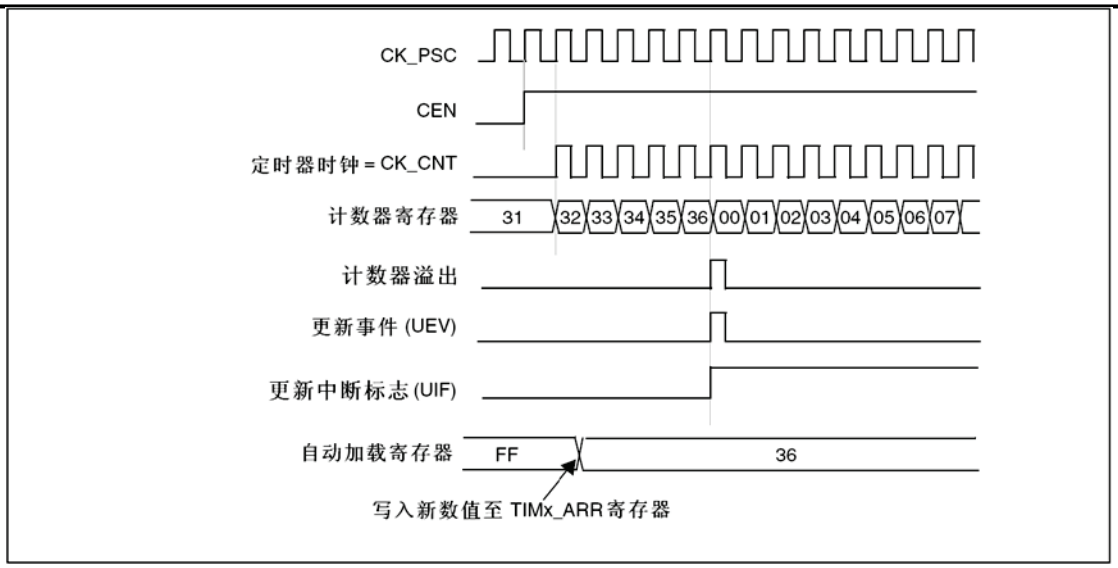


图 55 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

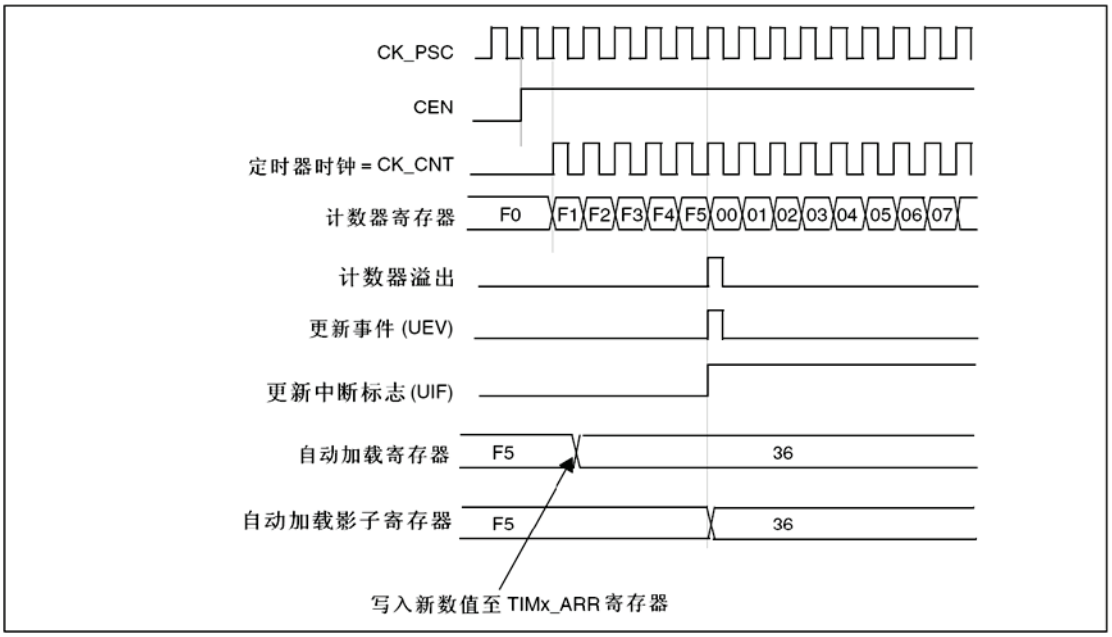


图 56 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)

向下计数模式

在向下模式中，计数器从自动装入的值(TIMx_ARR 计数器的值)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

如果使用了重复计数器，当向下计数重复了重复计数寄存器(TIMx_RCR)中设定的次数后，将产生更新事件(UEV)，否则每次计数器下溢时才产生更新事件。

在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始(但预分频系数不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载的值(TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：　自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

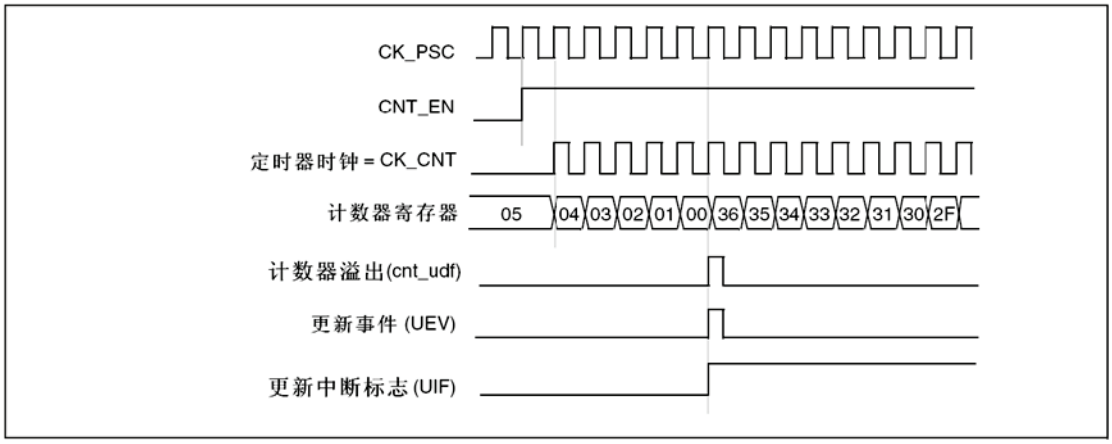


图 57 计数器时序图，内部时钟分频因子为 1

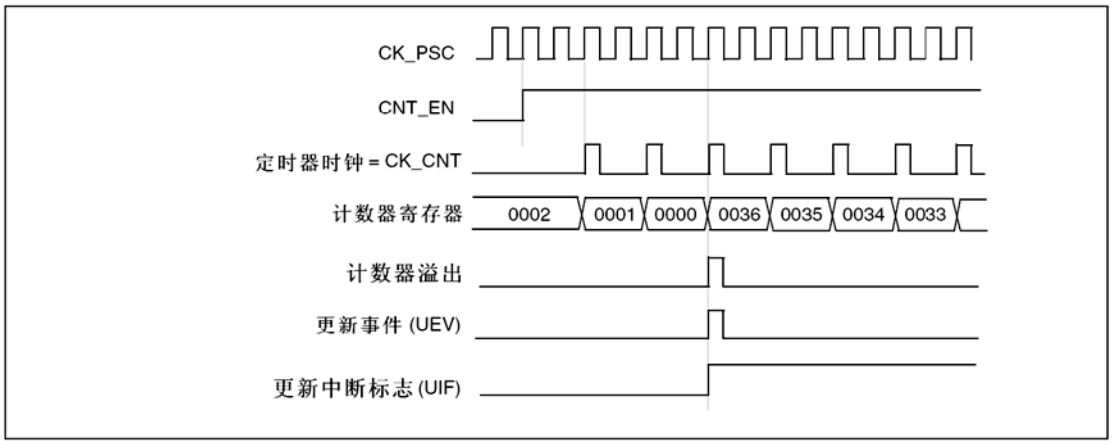


图 58 计数器时序图，内部时钟分频因子为 2

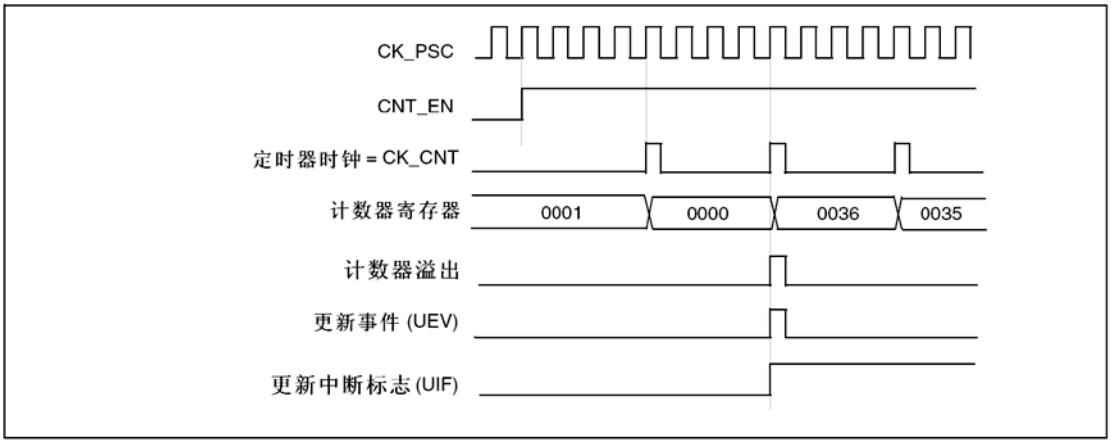


图 59 计数器时序图，内部时钟分频因子为 4

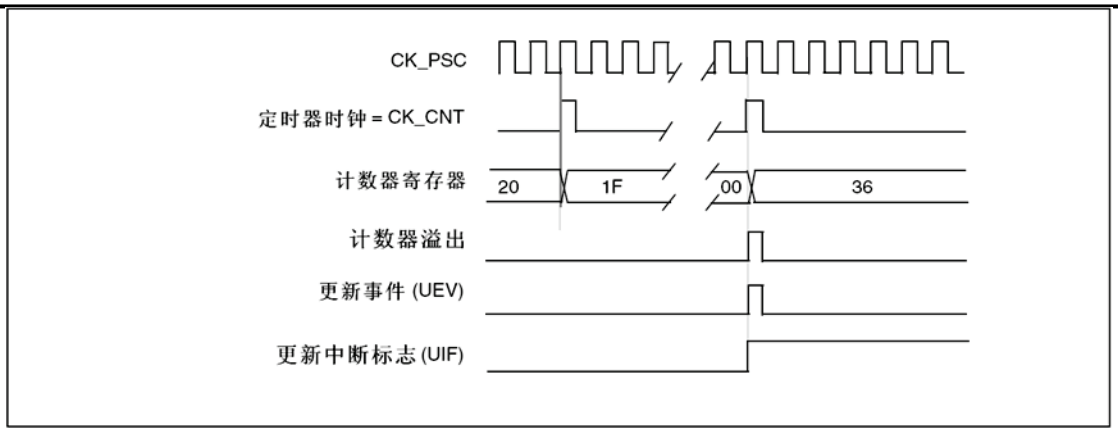


图 60 计数器时序图，内部时钟分频因子为 N

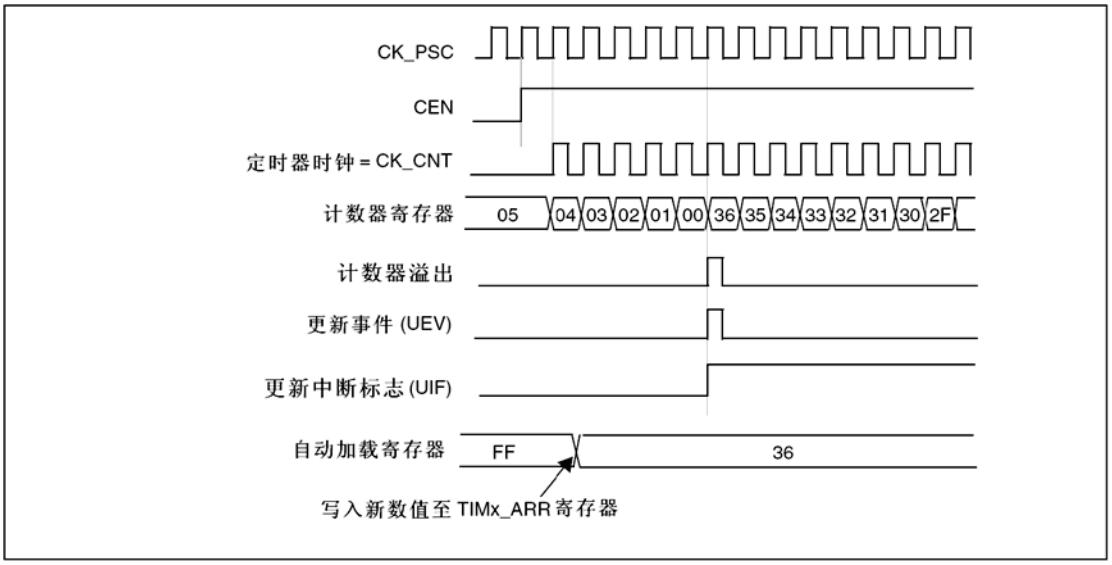


图 61 计数器时序图，当没有使用重复计数器时的更新事件

中央对齐模式(向上/向下计数)

在中央对齐模式，计数器从 0 开始计数到自动加载的值(TIMx_ARR 寄存器)-1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在此模式下，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄存器中的 UIF 位)也被设置。

- 重复计数器被重置为 TIMx_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：　如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

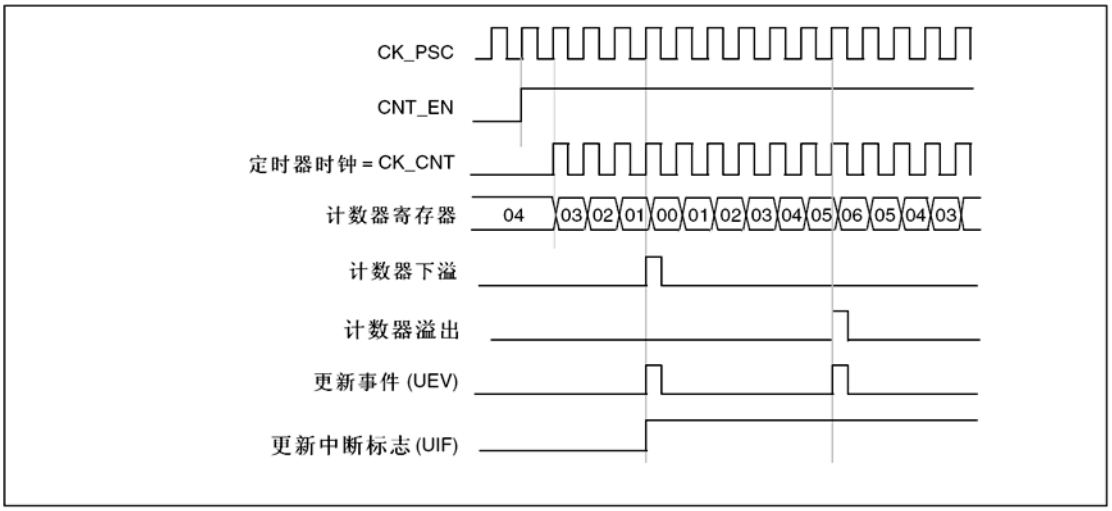


图 62 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

1．这里使用了中心对齐模式 1(详见 13.4.1 节)。

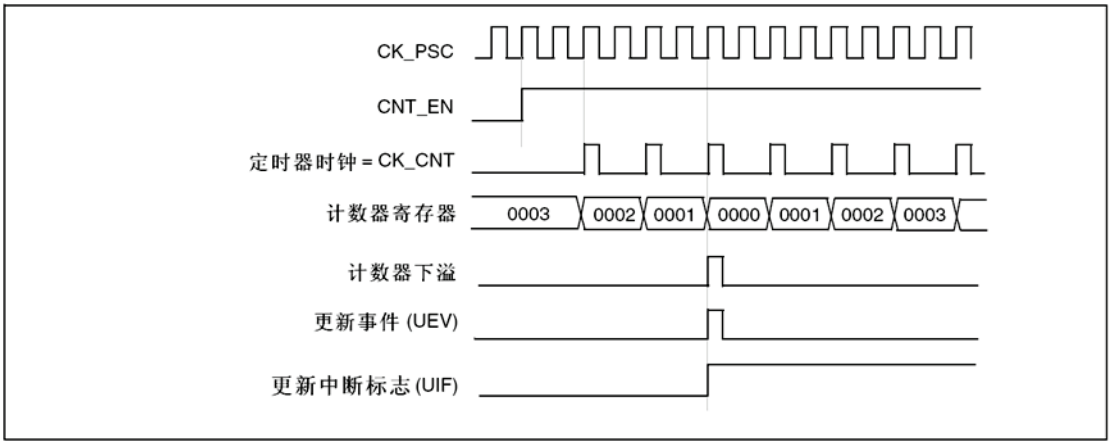


图 63 计数器时序图，内部时钟分频因子为 2

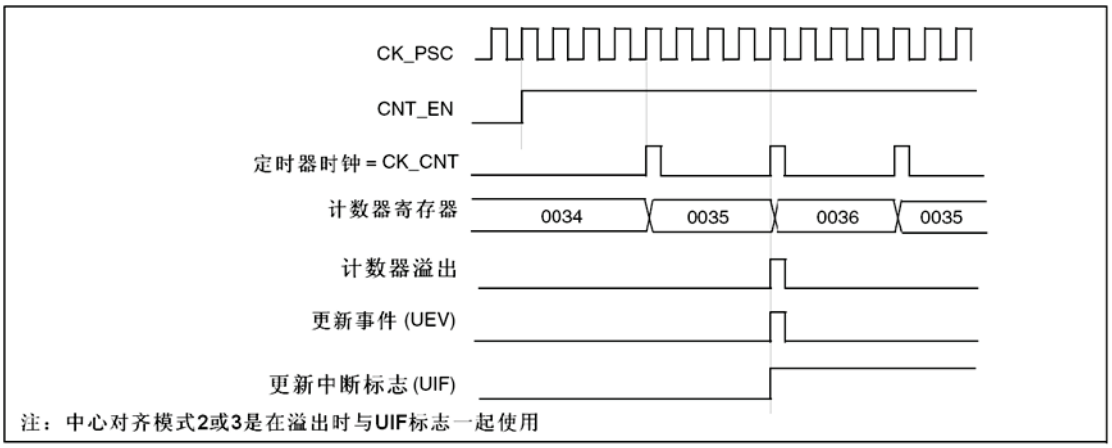


图 64 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

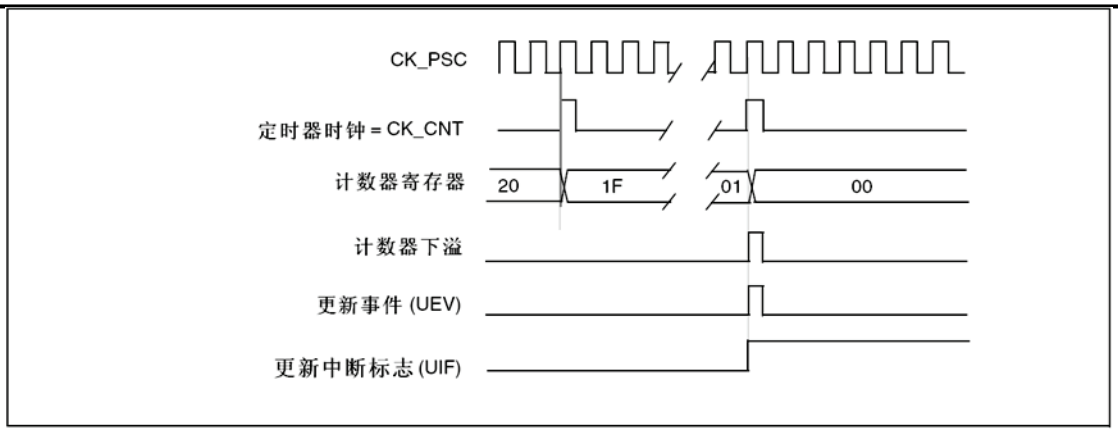


图 65 计数器时序图，内部时钟分频因子为 N

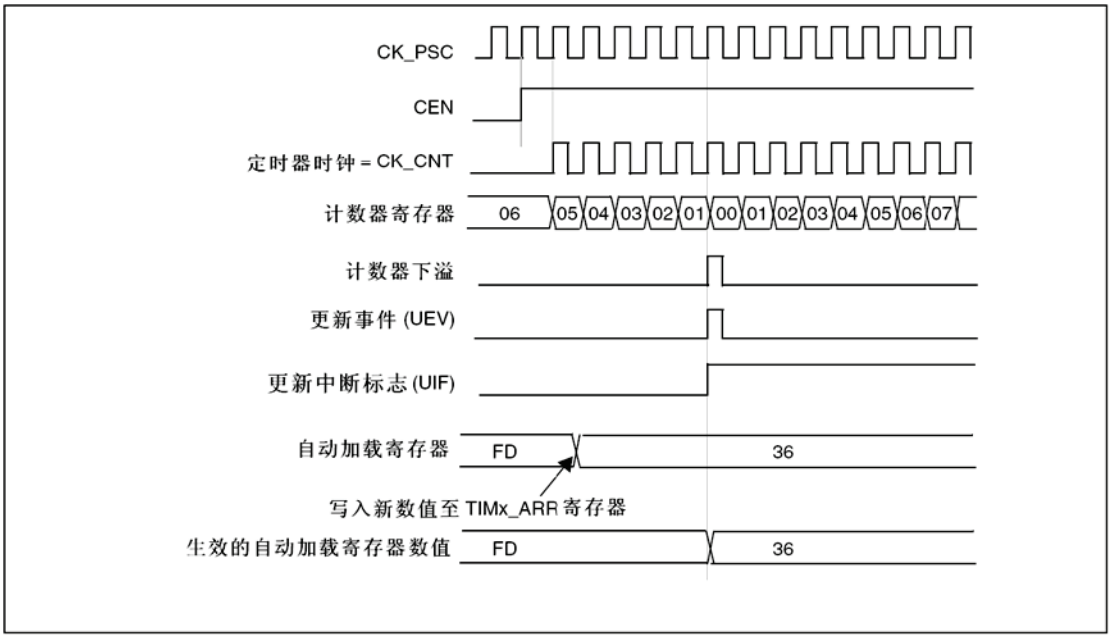


图 66 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

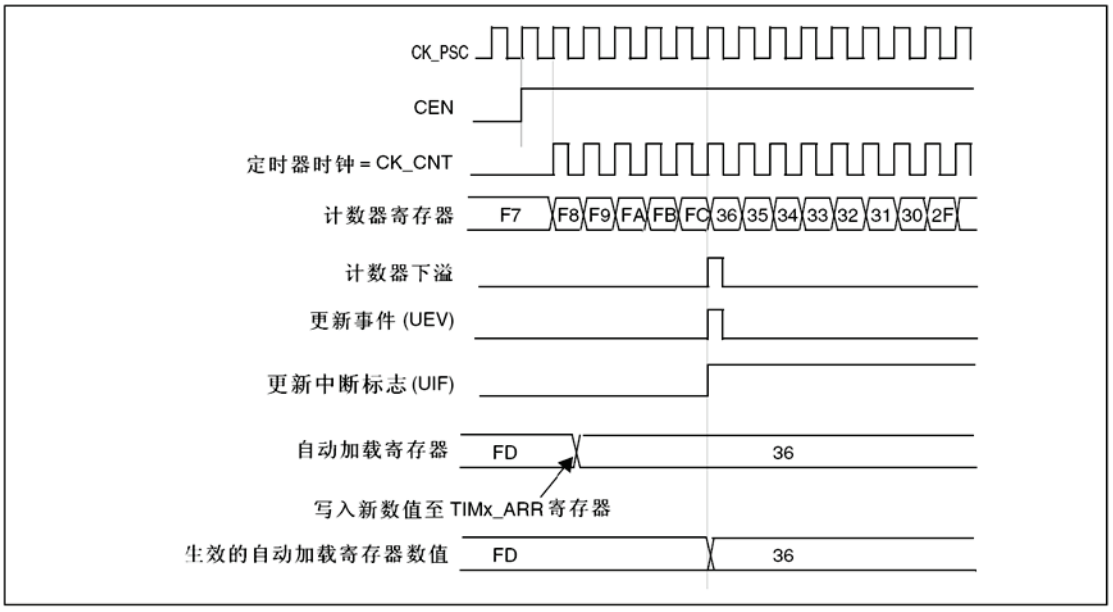


图 67 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

13.3.3 重复计数器

13.3.1 节“时基单元”解释了计数器上溢/下溢时更新事件(UEV)是如何产生的，然而事实上它只能在重复计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器(TIMx_ARR 自动重载入寄存器，TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx)，N 是 TIMx_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器溢出时，
- 向下计数模式下每次计数器下溢时，
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为 2xTck。

重复计数器是自动加载的，重复速率是由 TIMx_RCR 寄存器的值定义(参看图 68)。当更新事件由软件产生(通过设置 TIMx_EGR 中的 UG 位)或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 TIMx_RCR 寄存器中的内容被重载入到重复计数器。

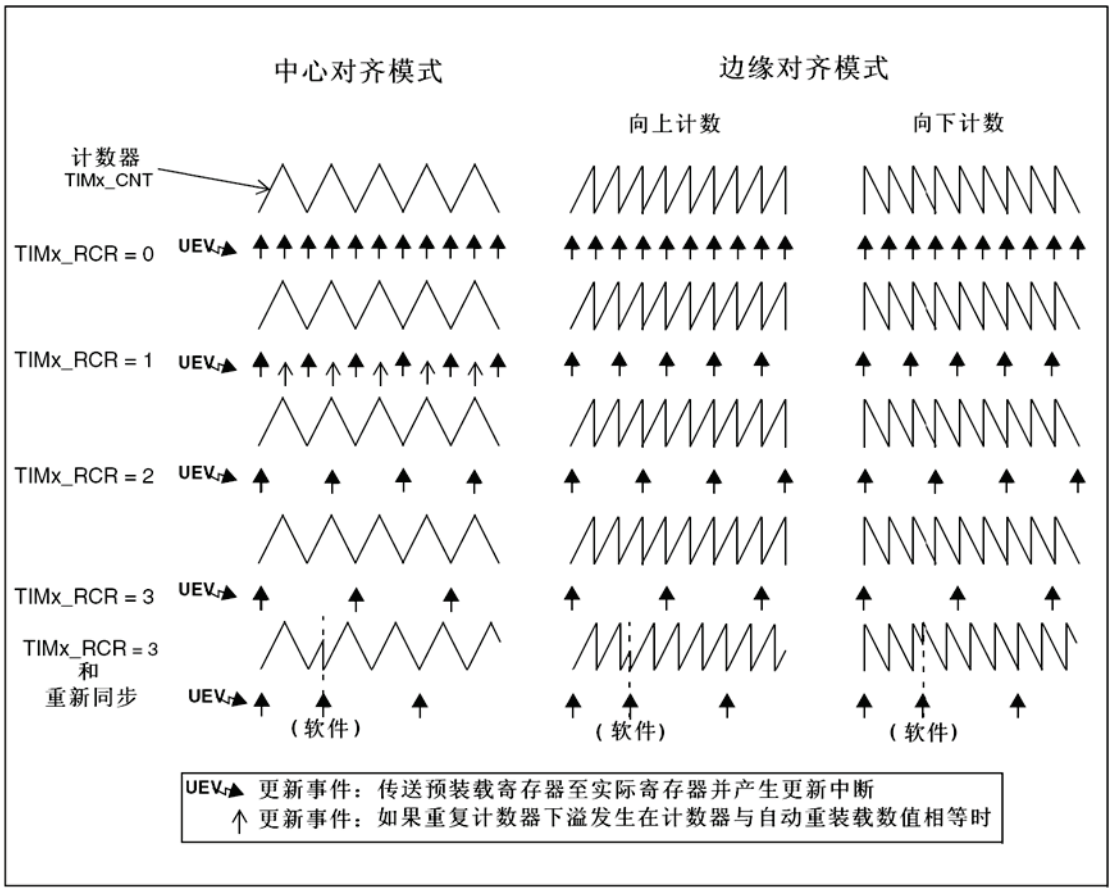


图 68 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

13.3.4 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。详见下一章。

内部时钟源(CK_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN、DIR(TIMx_CR1 寄存器)和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

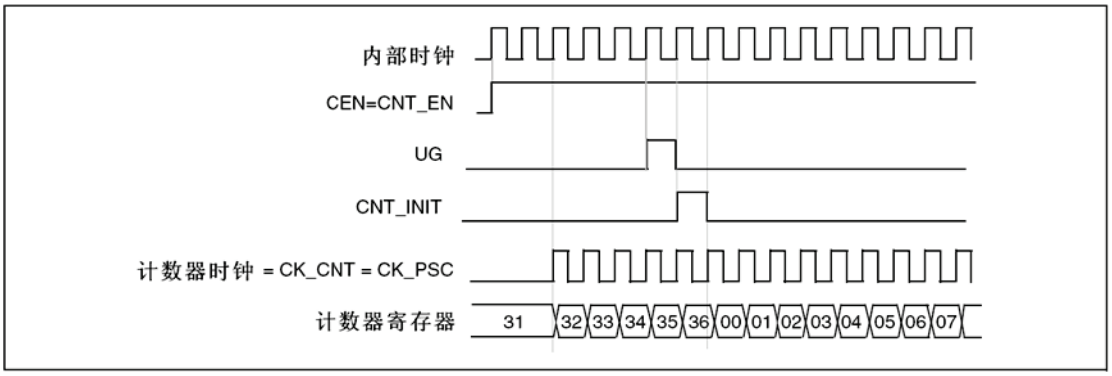


图 69 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

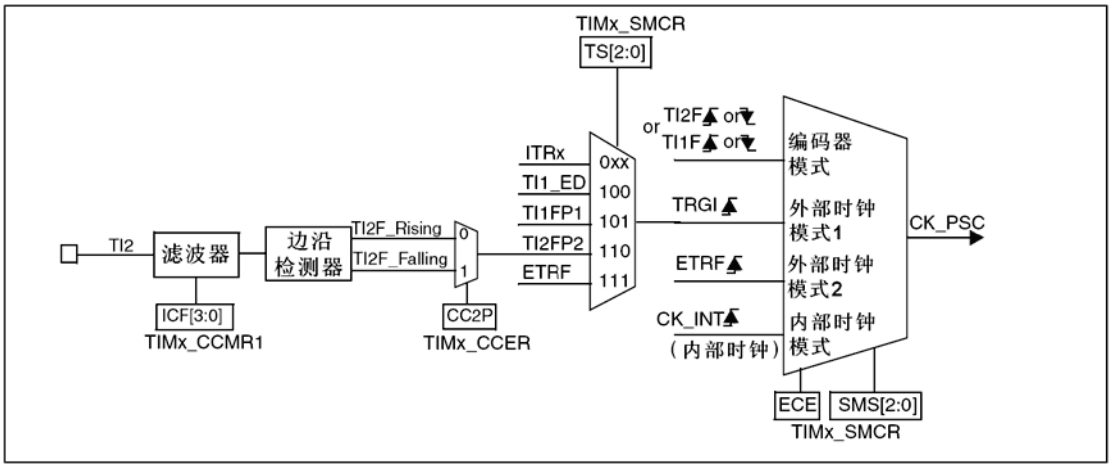


图 70 TI2 外部时钟连接例子

例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：配置 TIMx_CCMR1 寄存器 CC2S=01，配置通道 2 检测 TI2 输入的上升沿

配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)

配置 TIMx_CCER 寄存器的 CC2P=0，选定上升沿极性

配置 TIMx_SMCR 寄存器的 SMS=111，选择定时器外部时钟模式 1 配置 TIMx_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源

设置 TIMx_CR1 寄存器的 CEN=1，启动计数器

注：　捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

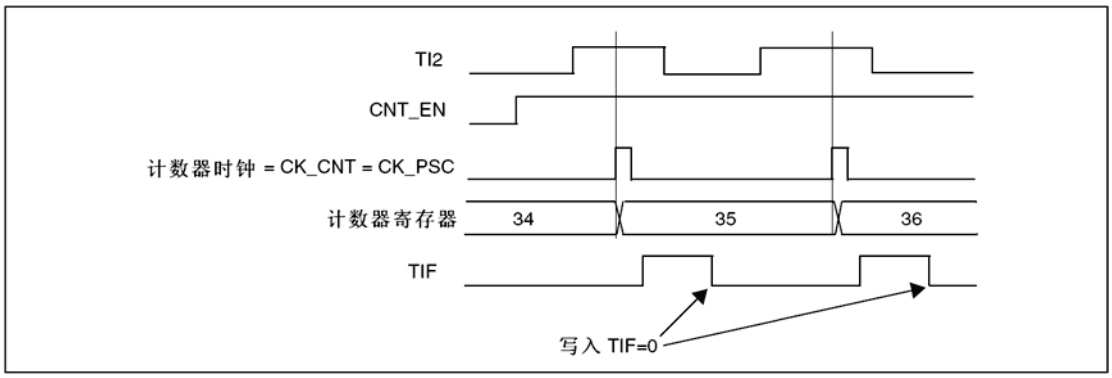


图 71 外部时钟模式 1 下的控制电路

外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=1 计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图

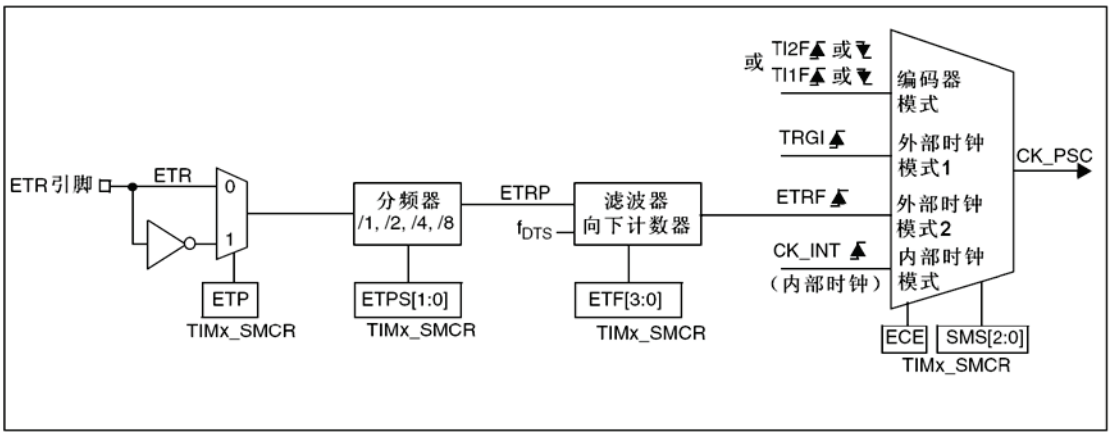


图 72 外部触发输入框图

例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

4.本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000

设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01

选择 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0 开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1

启动计数器，写 TIMx_CR1 寄存器中的 CEN=1 计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

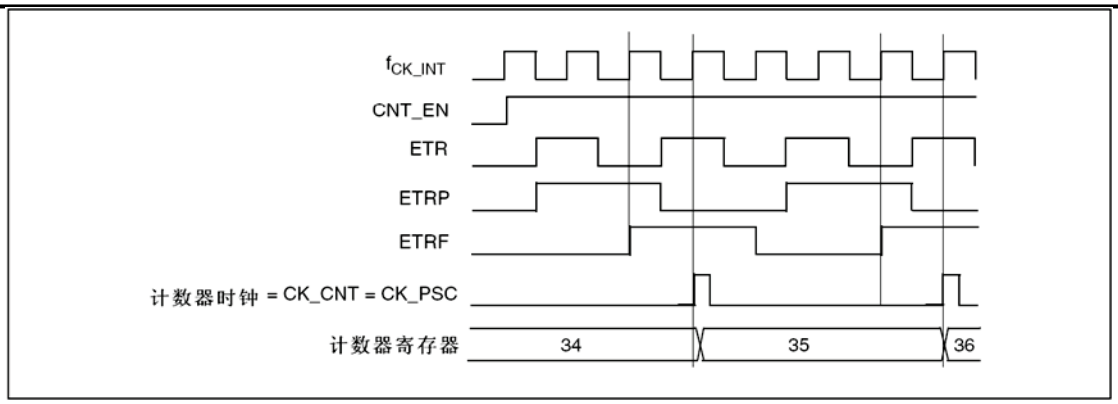


图 73 外部时钟模式 2 下的控制电路

13.3.5 捕获/比较通道

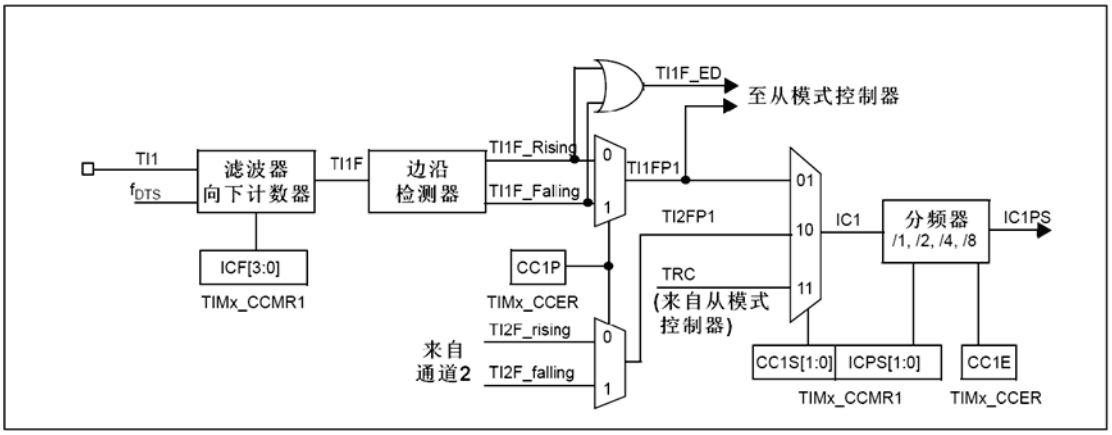


图 74 捕获/比较通道(如：通道 1 输入部分)

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

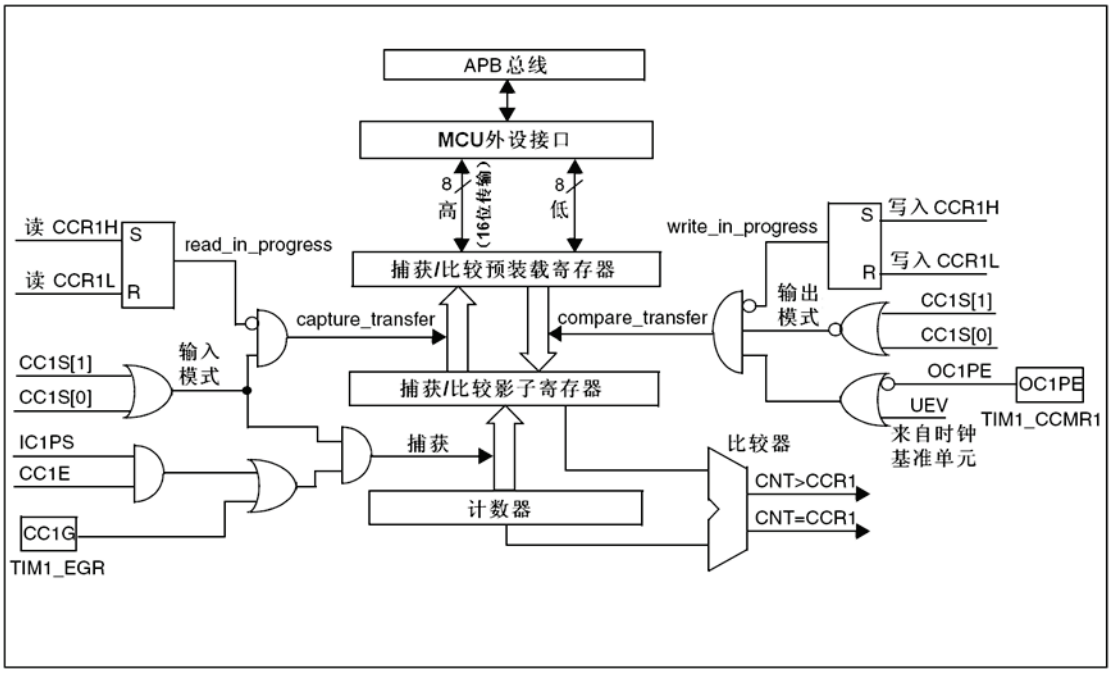


图 75 捕获/比较通道 1 的主电路

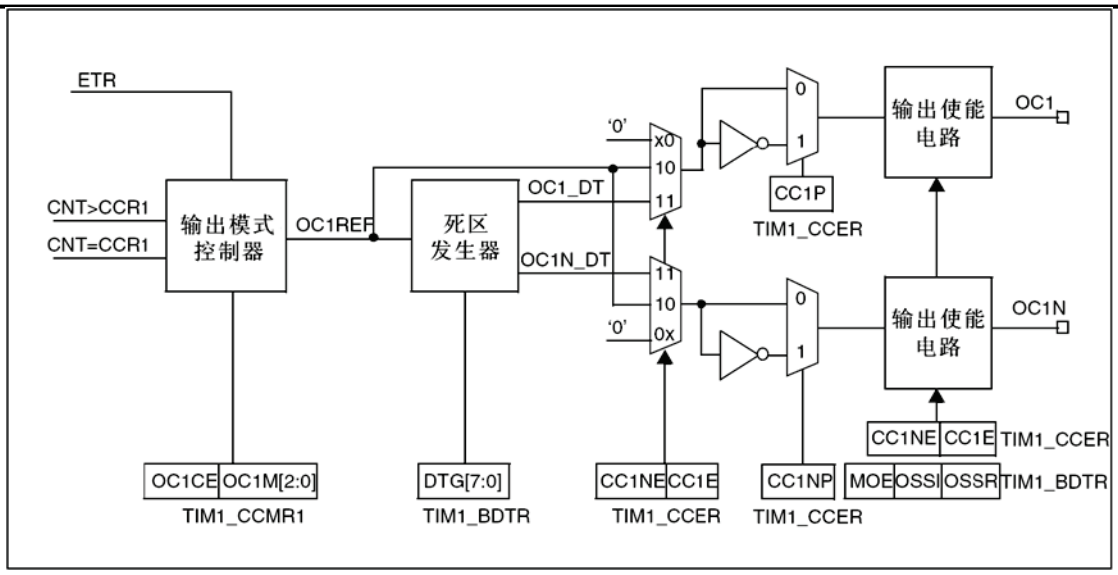


图 76 捕获/比较通道的输出部分(通道 1 至 3)

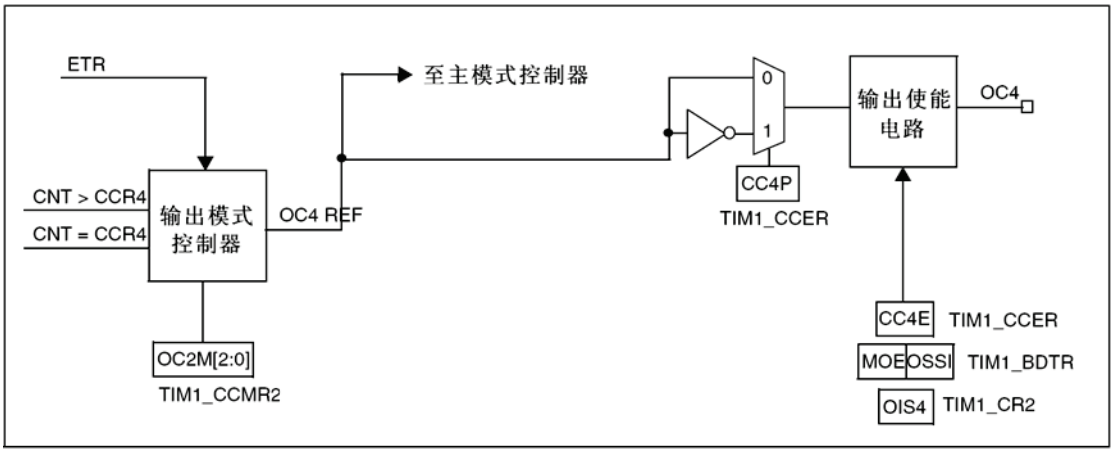


图 77 捕获/比较通道的输出部分(通道 4)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

13.3.6 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的
- CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我

们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。

- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：　设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

13.3.7 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选中 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

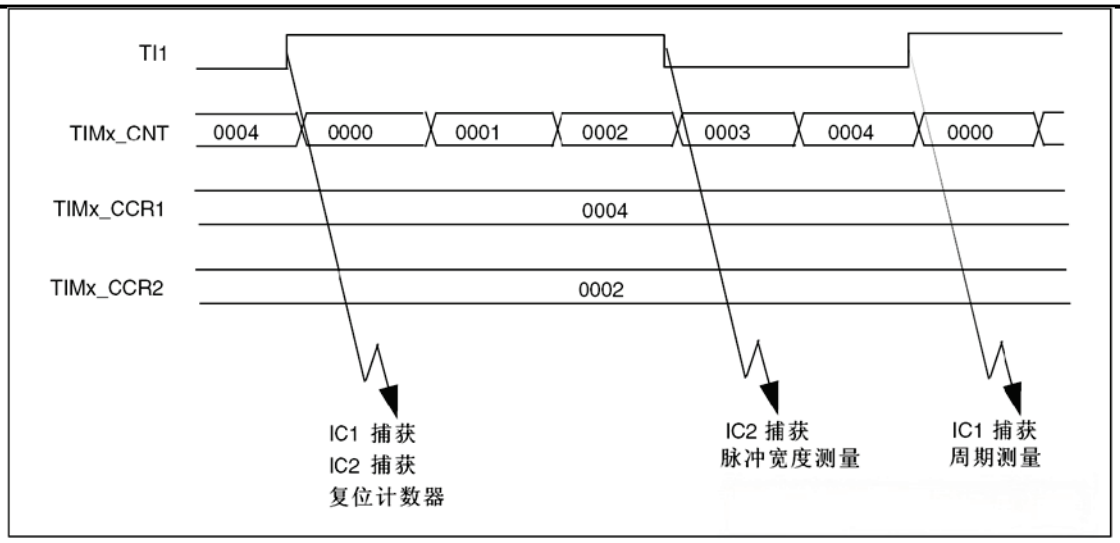


图 78 PWM 输入模式时序

因为只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx_CH1/TIMx_CH2 信号。

13.3.8 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

13.3.9 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)。
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
   - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
   - 置 OCxPE=0 禁用预装载寄存器
   - 置 CCxP=0 选择极性为高电平有效
   - 置 CCxE=1 使能输出
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器(OCxPE='0'，否则 TIMx_CCRx 的影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

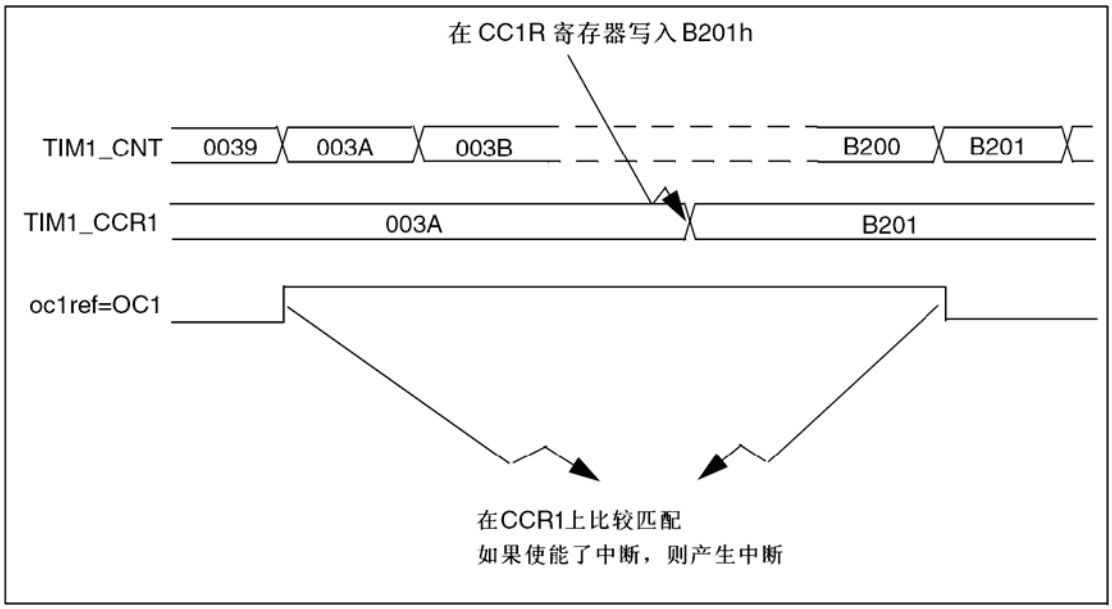


图 79 输出比较模式，翻转 OC1

13.3.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'110'(PWM 模式 1)或'111'(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过(TIMx_CCER 和 TIMx_BDTR 寄存器中)CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

PWM 边沿对齐模式

- 向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看 13.3.2 节。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为'1'。如果比较值为 0，则 OCxREF 保持为'0'。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

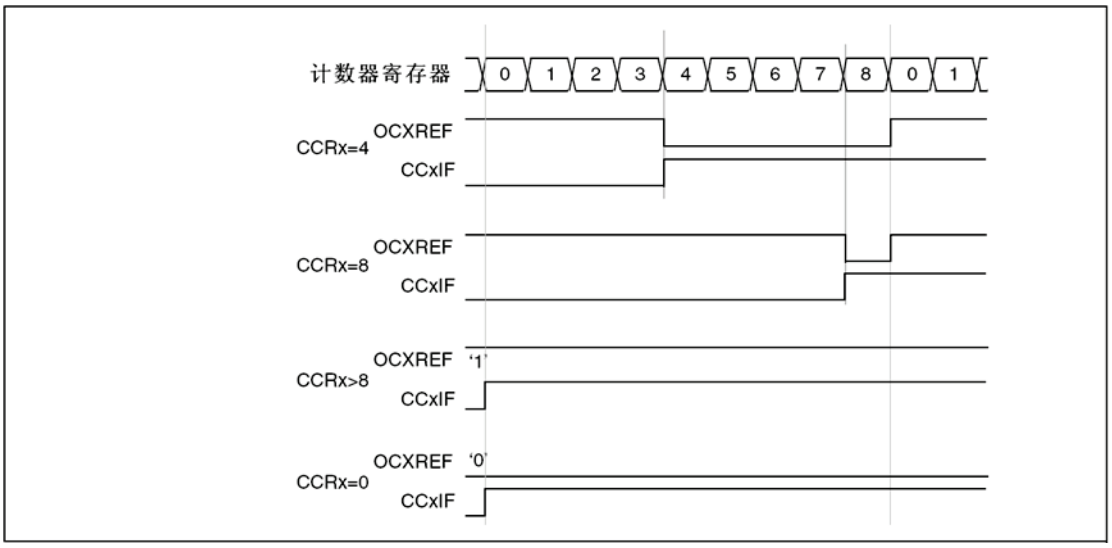


图 80 边沿对齐的 PWM 波形(ARR=8)

- 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。参看 13.3.2 节。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生 0%的 PWM 波形。

PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式(所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。参看 13.3.2 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器的 CMS=01，在中央对齐模式 1 下，当计数器向下计数时设置比较标志。

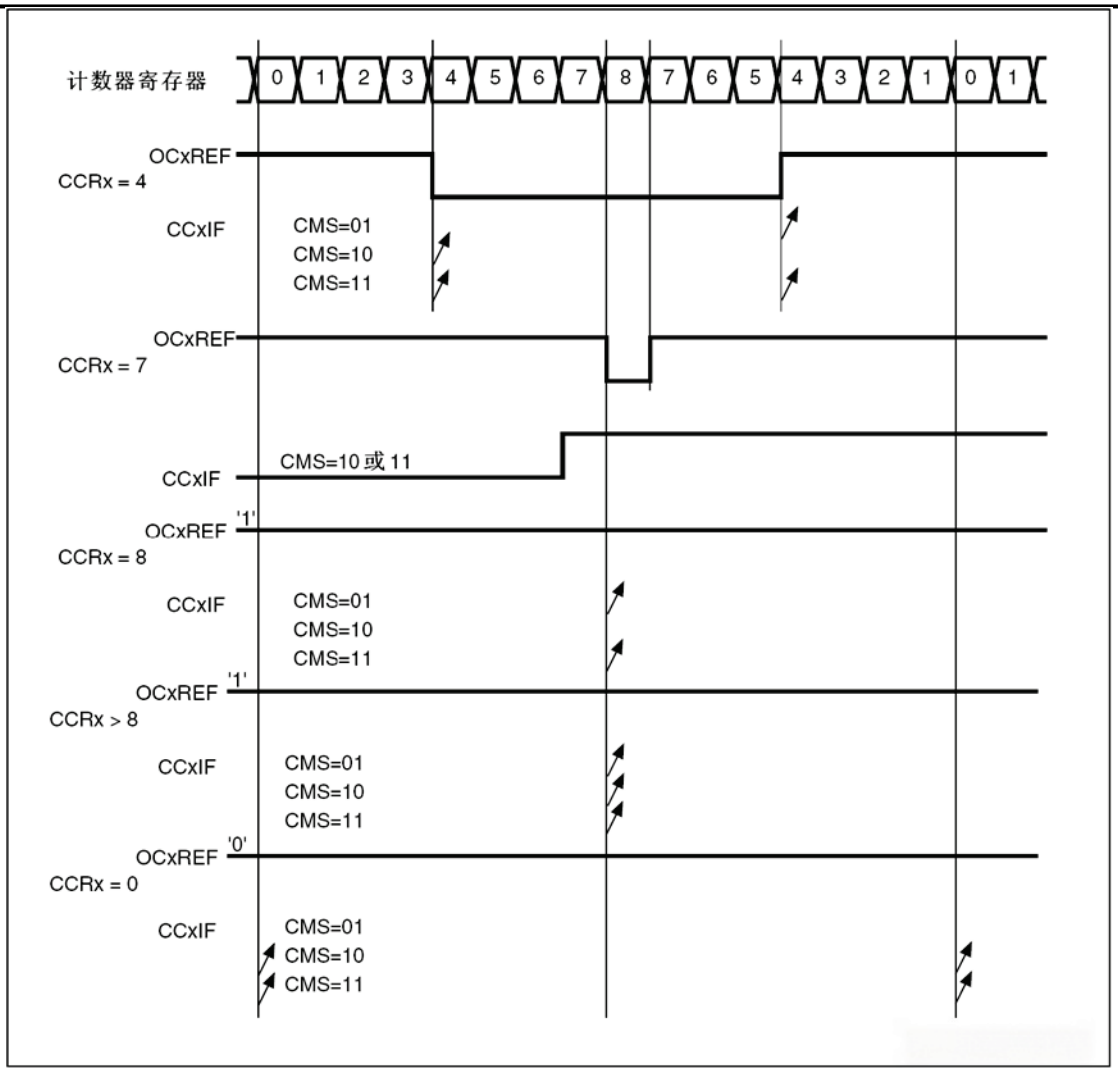


图 81 中央对齐的 PWM 波形(APR=8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx_CNT>TIMx_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

13.3.11 互补输出和死区插入

高级控制定时器(TIM1 和 TIM8)能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性(电平转换的延时、电源开关的延时等)来调整死区时间。

配置 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性(主输出 OCx 或互补输出 OCxN)。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见表 73 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLE 状态时(MOE 下降到 0)死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 10 位的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。如果延迟大于当前有效的输出宽度(OCx 或者 OCxN)，则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

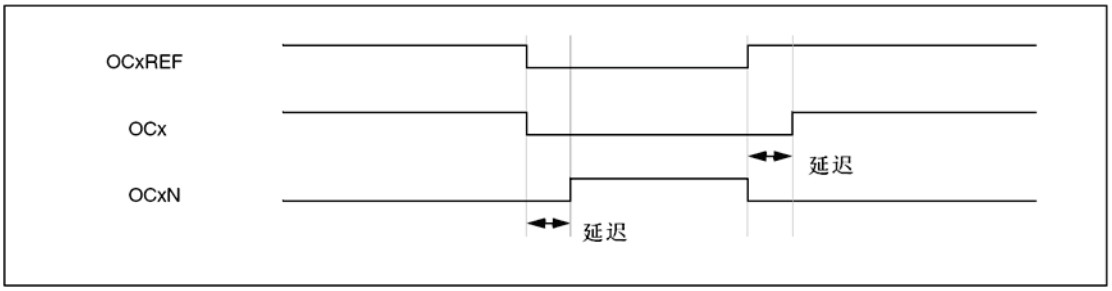


图 82 带死区插入的互补输出

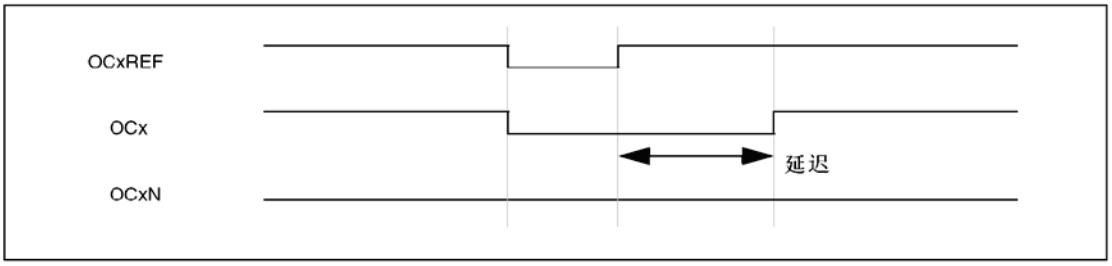


图 83 死区波形延迟大于负脉冲

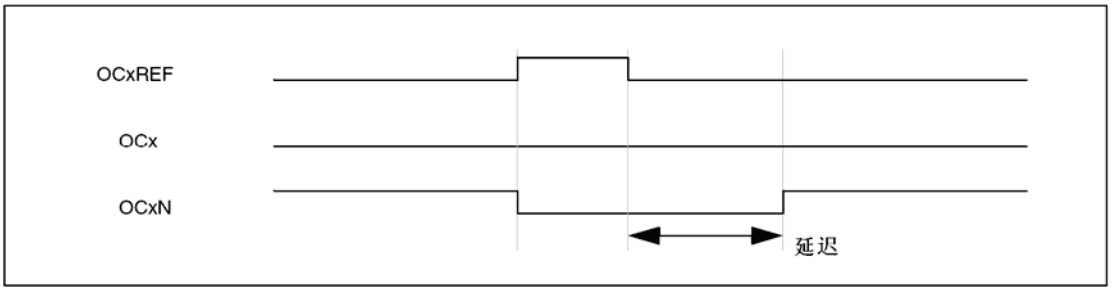


图 84 死区波形延迟大于正脉冲

每一个通道的死区延时都是相同的，是由 TIMx_BDTR 寄存器中的 DTG 位编程配置。详见 13.4.18 节 TIM1 和 TIM8 刹车和死区寄存器(TIMx_BDTR)中的延时计算。

重定向 OCxREF 到 OCx 或 OCxN

在输出模式下(强置、输出比较或 PWM)，通过配置 TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形(例如 PWM 或者静态有效电平)。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注：　当只使能 OCxN(CCxE=0,CCxNE=1)时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1)，当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

13.3.12 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIMx_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx_CR2 寄存器中的 OISx 和 OISxN 位)，输出使能信号和无效电平都会被修改。但无论何时，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见表 73 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。

刹车源既可以是刹车输入引脚又可以是一个时钟失败事件。时钟失败事件由复位时钟控制器中的时钟安全系统产生，详见 6.2.7 节时钟安全系统(CSS)。

系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号(作用在输出端)和同步控制位(在 TIMx_BDTR 寄存器中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由 OSSI 位选择)。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态(取决于极性)。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。

注：　因为重新同步 MOE，死区时间比通常情况下长一些(大约 2 个 ck_tim 的时钟周期)。

  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 如果设置了 TIMx_DIER 寄存器中的 BIE 位，当刹车状态标志(TIMx_SR 寄存器中的 BIF 位)为'1'时，则产生一个中断。如果设置了 TIMx_DIER 寄存器中的 BDE 位，则产生一个 DMA 请求。
- 如果设置了 TIMx_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注：　刹车输入为电平有效。所以，当刹车输入有效时，不能同时(自动地或者通过软件)设置 MOE。同时，状态标志 BIF 不能被清除。

刹车由 BRK 输入产生，它的有效极性是可编程的，且由 TIMx_BDTR 寄存器中的 BKE 位开启。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数(死区长度，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性)。用户可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，参看 13.4.18 节 TIM1 和 TIM8 刹车和死区寄存器(TIMx_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

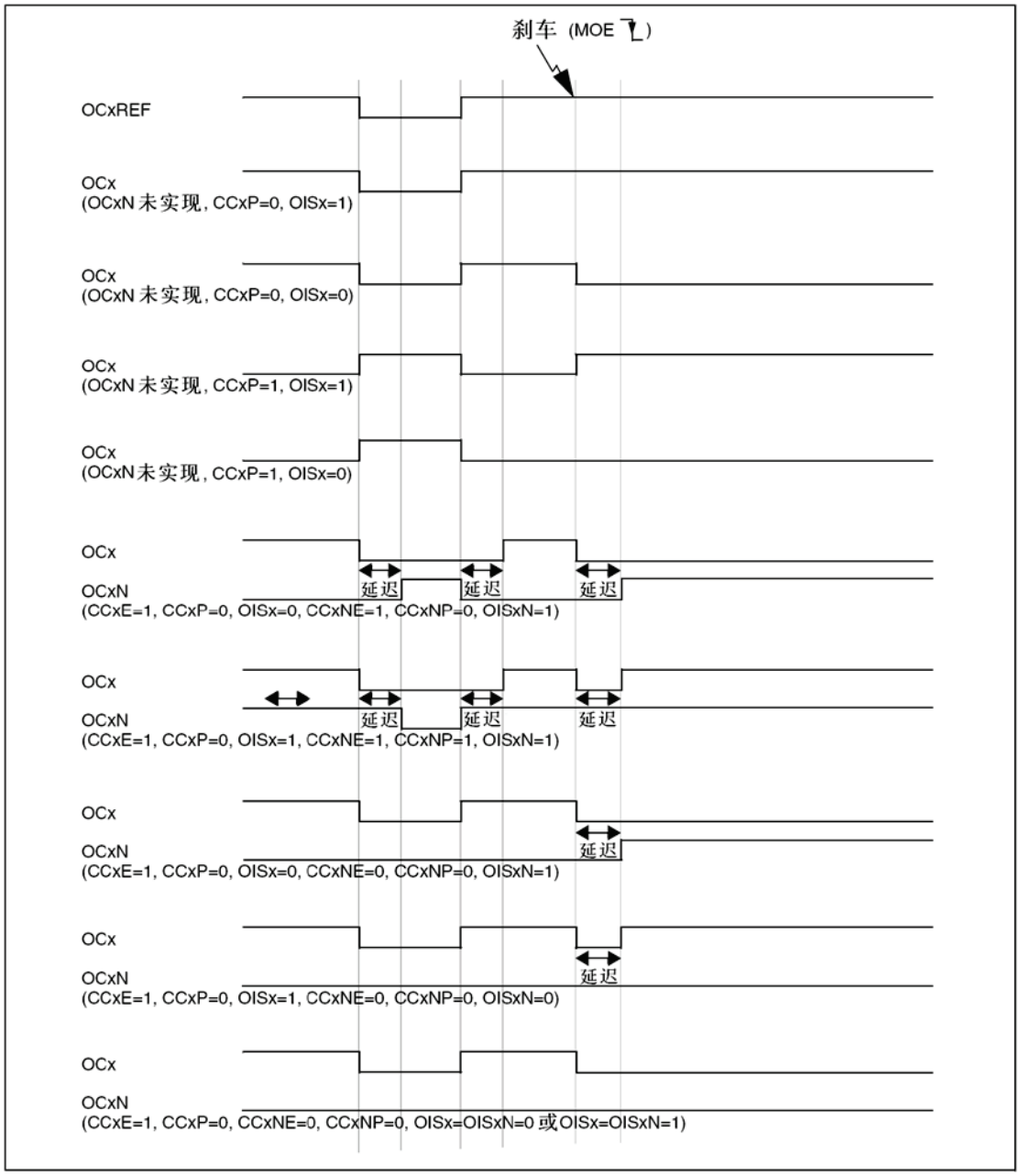


图 85 响应刹车的输出

13.3.13 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM 模式。

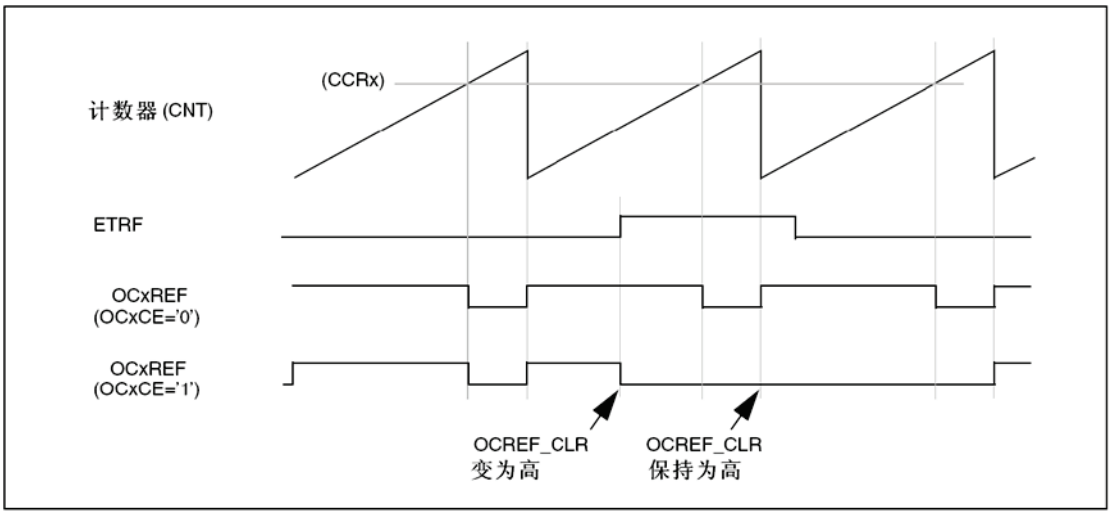


图 86 清除 TIMx 的 OCxREF

13.3.14 产生六步 PWM 输出

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修更改所有通道的配置。COM 可以通过设置 TIMx_EGR 寄存器的 COM 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位(TIMx_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIMx_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

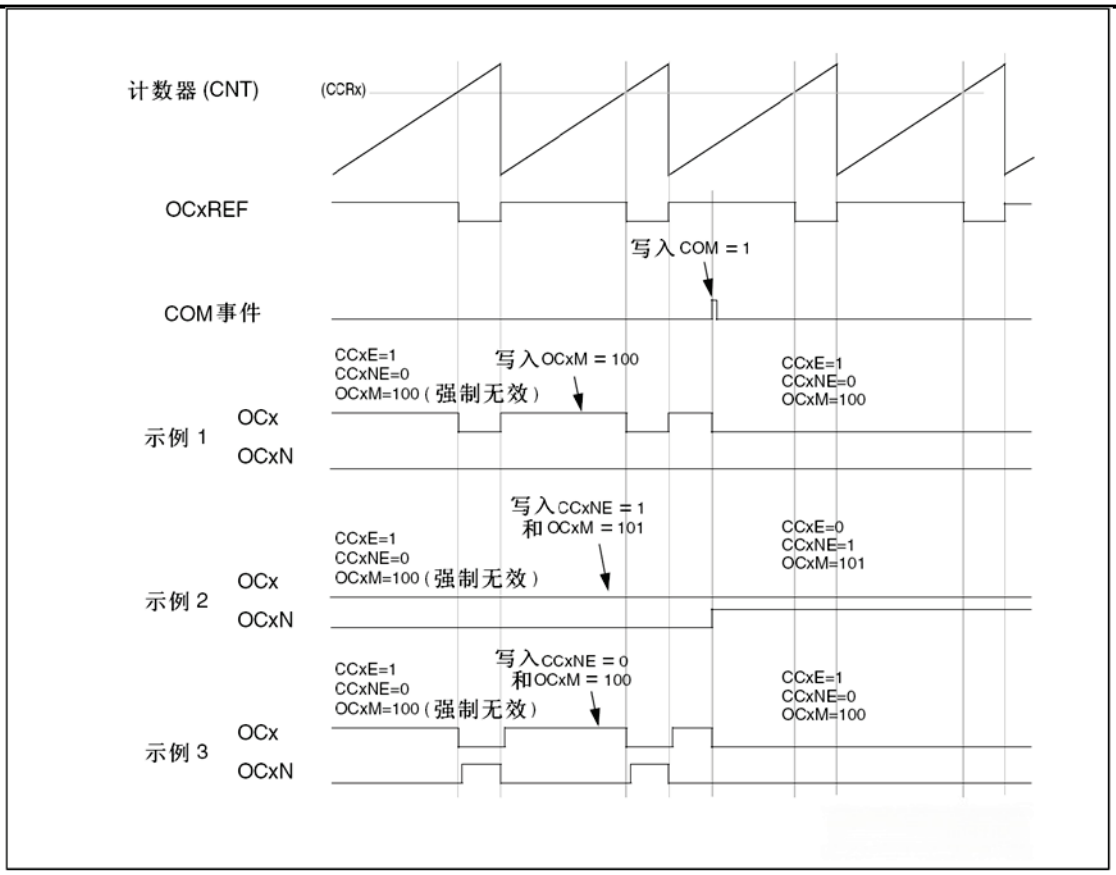


图 87 产生六步 PWM，使用 COM 的例子(OSSR=1)

13.3.15 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器 CNT<CCRx≤ARR(特别地,0<CCRx)，
- 向下计数方式：计数器 CNT>CCRx。

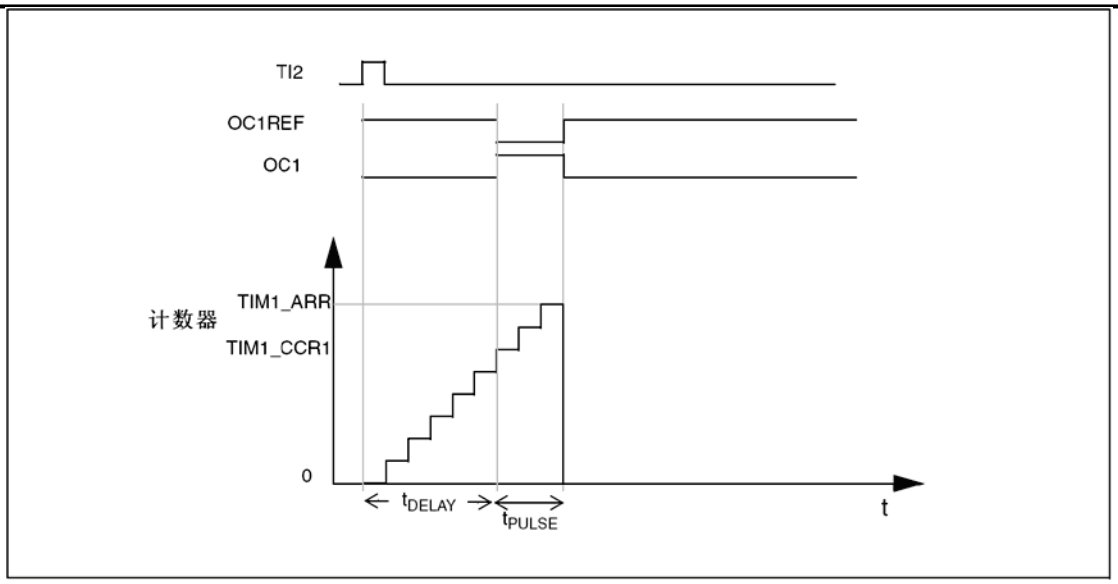


图 88 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长

度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。OPM 的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)
- tDELAY 由 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR-TIMx_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=1，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

13.3.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 71，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 71 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1 对应 TI2,TI2FP2 对应 TI1)	TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S='01'(TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S='01'(TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P='0'(TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P='0'(TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS='011'(TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN='1'(TIMx_CR1 寄存器，计数器使能)

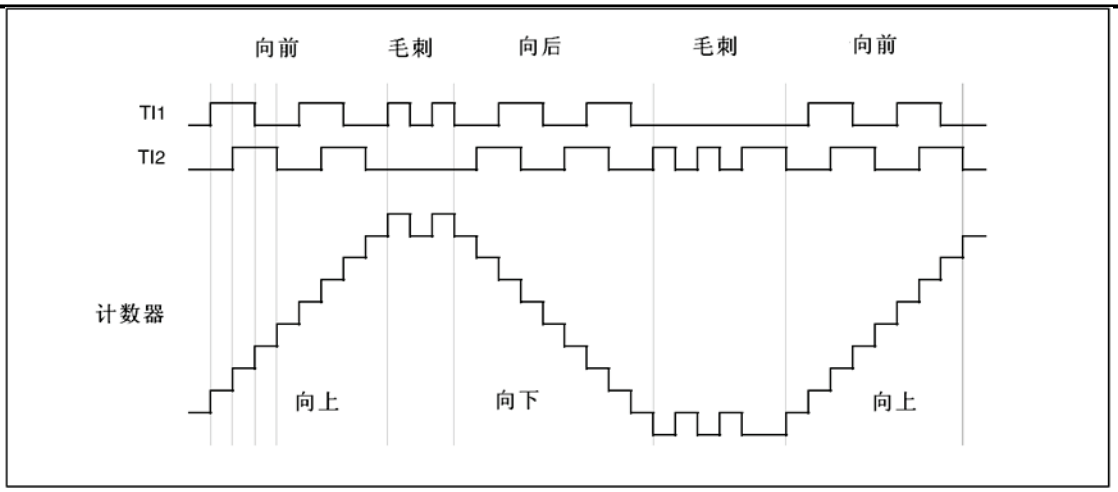


图 89 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1'，其他配置与上例相同)

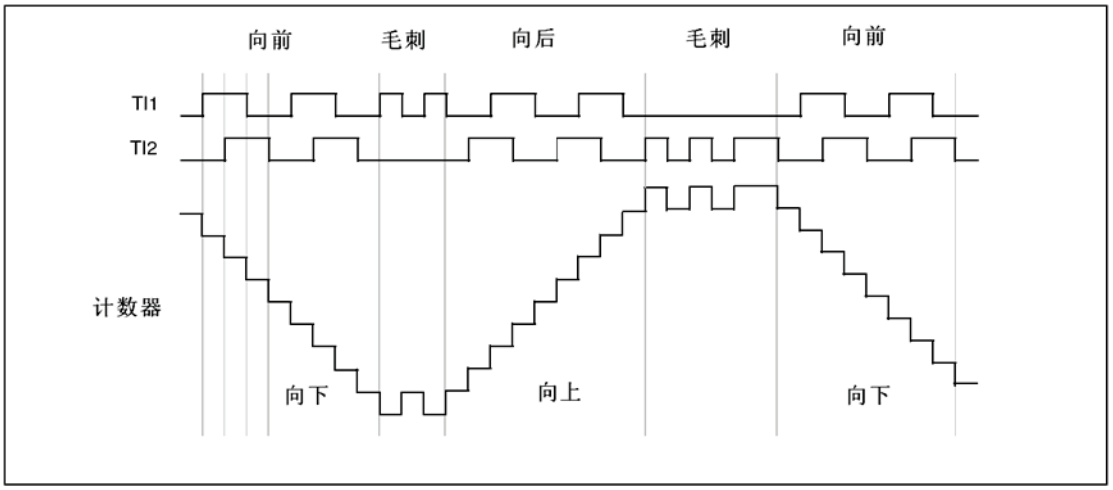


图 90 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

13.3.17 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下节 13.3.18 给出了此特性用于连接霍尔传感器的例子。

13.3.18 与霍尔传感器的接口

使用高级控制定时器(TIM1 或 TIM8)产生 PWM 信号驱动马达时，可以用另一个通用 TIMx(TIM2、TIM3、TIM4 或 TIM5)定时器作为“接口定时器”来连接霍尔传感器，见图 91，3 个定时器输入脚(CC1、CC2、CC3)通过一个异或门连接到 TI1 输入通道(通过设置 TIMx_CR2 寄存器中的 TI1S 位来选择)，“接口定时器”捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F_ED。每当 3 个输入之一变化时，计数器从新从 0 开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC(见图 74)。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以用来在输出模式产生一个脉冲，这个脉冲可以(通过触发一个 COM 事件)用于改变高级定时器 TIM1 或 TIM8 各个通道的属性，而高级控制定时器产生 PWM 信号驱动马达。因此“接口定时器”通道必须编程为在一个指定的延时(输出比较或 PWM 模式)之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TIM1 或 TIM8。

举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIMx 的 PWM 配置。

- 置 TIMx_CR2 寄存器的 TI1S 位为'1'，配置三个定时器输入逻辑或到 TI1 输入，
- 时基编程：置 TIMx_ARR 为其最大值(计数器必须通过 TI1 的变化清零)。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。
- 设置通道 1 为捕获模式(选中 TRC)：置 TIMx_CCMR1 寄存器中 CC1S=01，如果需要，还可以设置数字滤波器。
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIMx_CCMR1 寄存器中的 OC2M=111 和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的(TIMx_CR2 寄存器中 CCPC=1)，同时触发输入控制 COM 事件(TIMx_CR2 寄存器中 CCUS=1)。在一次 COM 事件后，写入下一步的 PWM 控制位(CCxE、OCxM)，这可以在处理 OC2REF 上升沿的中断子程序里实现。

下图显示了这个实例

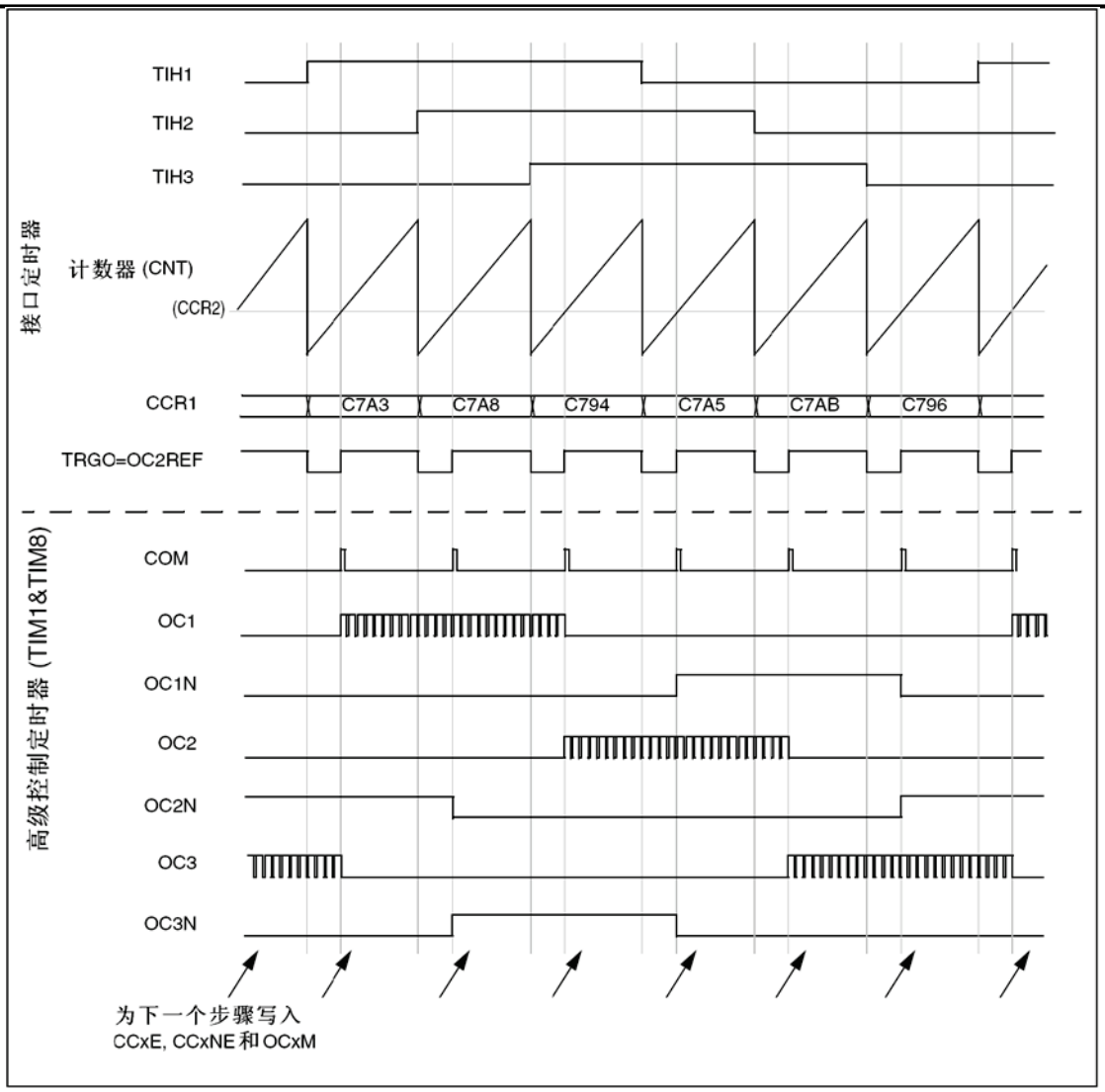


图 91 霍尔传感器接口的实例

13.3.19 TIMx 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 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，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

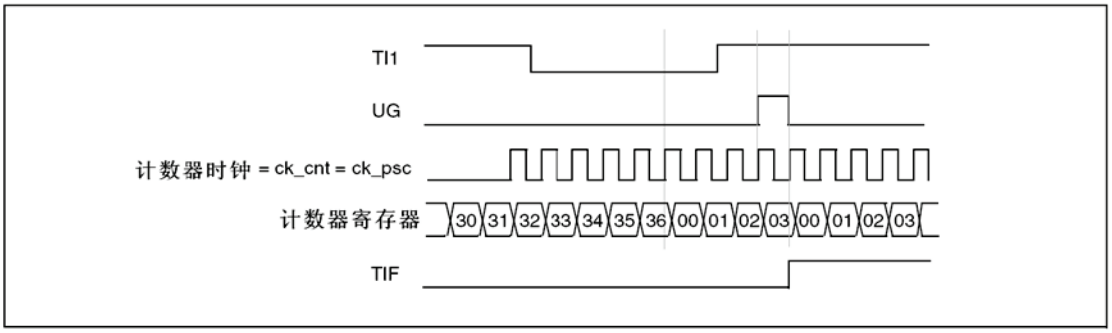


图 92 复位模式下的控制电路

从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

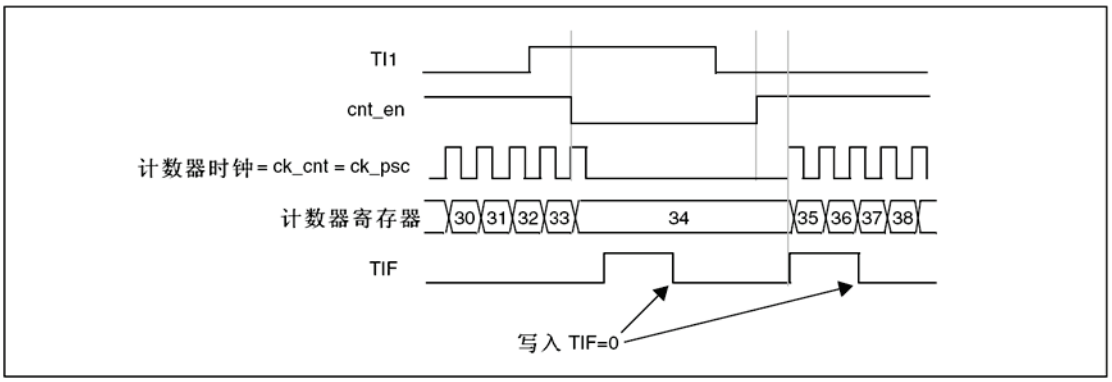


图 93 门控模式下的控制电路

从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

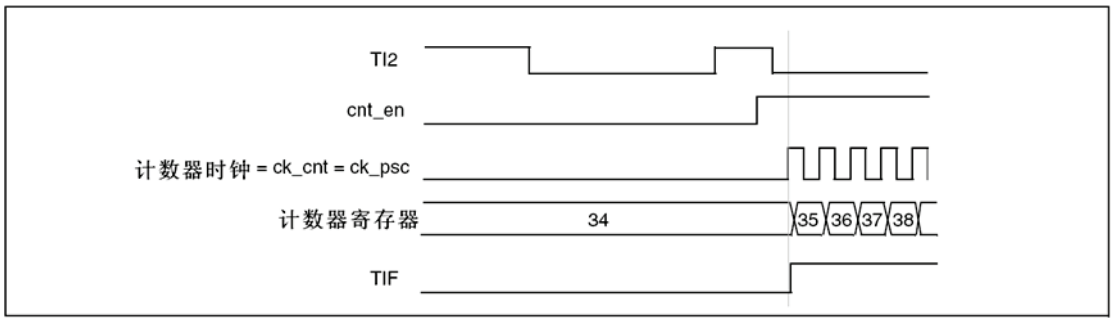


图 94 触发器模式下的控制电路

从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。
2. 按如下配置通道 1，检测 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 以确定极性(只检测上升沿)
3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

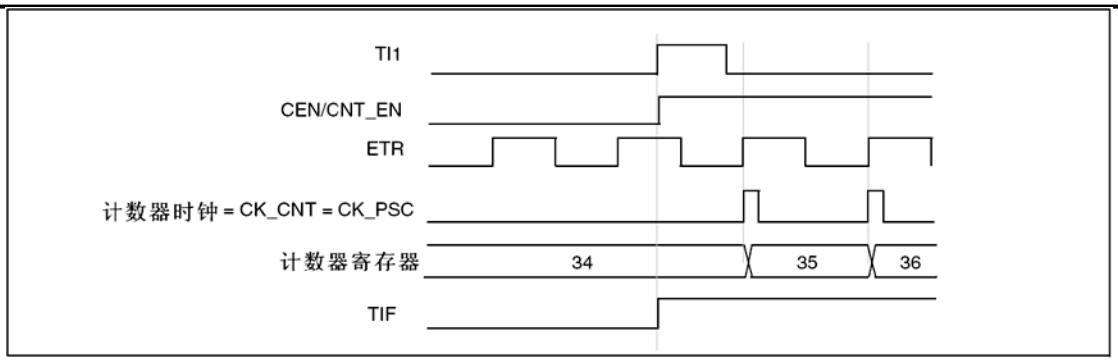


图 95 外部时钟模式 2 + 触发模式下的控制电路

13.3.20 定时器同步

所有 TIM 定时器在内部相连，用于定时器同步或链接。详见下一章 14.3.15 节。

13.3.21 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详见随后的 28.16.2 节。

13.4 TIM1 和 TIM8 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 章。
可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

13.4.1 TIM1 和 TIM8 控制寄存器 1(TIMx_CR1)

偏移地址：0x00
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9:8	CKD[1:0]	**CKD[1:0]**：时钟分频因子(Clock division) 这 2 位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟之间的分频比例。 00：tDTS=tCK_INT 01：tDTS=2xtCK_INT 10：tDTS=4xtCK_INT 11：保留，不要使用这个配置
7	ARPE	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:5	CMS[1:0]	**CMS[1:0]**：选择中央对齐模式(Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。

		11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
4	**DIR**	**DIR**：方向(Direction) 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
3	**OPM**	**OPM**：单脉冲模式(One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	**URS**	**URS**：更新请求源(Update request source) 软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
1	**UDIS**	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	**CEN**	**CEN**：使能计数器(Counter enable) 0：禁止计数器； 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

13.4.2 TIM1 和 TIM8 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	保留	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位	符号	说明
15	**Reserved**	保留，始终读为 0。
14	**OIS4**	**OIS4**：输出空闲状态 4(OC4 输出)。参见 OIS1 位。
13	**OIS3N**	**OIS3N**：输出空闲状态 3(OC3N 输出)。参见 OIS1N 位。
12	**OIS3**	**OIS3**：输出空闲状态 3(OC3 输出)。参见 OIS1 位。
11	**OIS2N**	**OIS2N**：输出空闲状态 2(OC2N 输出)。参见 OIS1N 位。
10	**OIS2**	**OIS2**：输出空闲状态 2(OC2 输出)。参见 OIS1 位。
9	**OIS1N**	**OIS1N**：输出空闲状态 1(OC1N 输出)(Output Idle state1) 0：当 MOE=0 时，死区后 OC1N=0； 1：当 MOE=0 时，死区后 OC1N=1。

		注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
8	OIS1	**OIS1**：输出空闲状态 1(OC1 输出)(Output Idle state1) 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0； 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 注：已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后，该位不能被修改。
7	TI1S	**TI1S**：TI1 选择(TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
6:4	MMS[2:0]	**MMS[2:0]**：主模式选择(Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：复位–TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的复位(从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能–计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：更新–更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲–在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：比较–OC1REF 信号被用于作为触发输出(TRGO)。 101：比较–OC2REF 信号被用于作为触发输出(TRGO)。 110：比较–OC3REF 信号被用于作为触发输出(TRGO)。 111：比较–OC4REF 信号被用于作为触发输出(TRGO)。
3	CCDS	**CCDS**：捕获/比较的 DMA 选择(Capture/compare DMA selection) 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
2	CCUS	**CCUS**：捕获/比较控制更新选择(Capture/compare control update selection) 0：如果捕获/比较控制位是预装载的(CCPC=1)，只能通过设置 COM 位更新它们； 1：如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注：该位只对具有互补输出的通道起作用。
1	Reserved	保留，始终读为 0。
0	CCPC	**CCPC**：捕获/比较预装载控制位(Capture/comparep reloaded control) 0：CCxE，CCxNE 和 OCxM 位不是预装载的； 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新。 注：该位只对具有互补输出的通道起作用。

13.4.3 TIM1 和 TIM8 从模式控制寄存器(TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位	符号	说明
15	ETP	**ETP**：外部触发极性(External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效；

<table>
<tr><td></td><td></td><td>1：ETR 被反相，低电平或下降沿有效。</td></tr>
<tr><td>14</td><td>ECE</td><td>ECE：外部时钟使能位(External clock enable)
该位启用外部时钟模式 2
0：禁止外部时钟模式 2；
1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。
注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=111)具有相同功效。
注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是'111')。
注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。</td></tr>
<tr><td>13:12</td><td>ETPS[1:0]</td><td>ETPS[1:0]：外部触发预分频(External trigger prescaler)
外部触发信号 ETRP 的频率必须最多是 TIMxCLK 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。
00：关闭预分频；
01：ETRP 频率除以 2；10：ETRP 频率除以 4；
11：ETRP 频率除以 8。</td></tr>
<tr><td>11:8</td><td>ETF[3:0]</td><td>ETF[3:0]：外部触发滤波(External trigger filter)
这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。
0000：无滤波器，以 fDTS 采样
0001:采样频率 fSAMPLING=fCK_INT，N=2
0010：采样频率 fSAMPLING=fCK_INT，N=4
0011：采样频率 fSAMPLING=fCK_INT，N=8
0100：采样频率 fSAMPLING=fDTS/2，N=6
0101：采样频率 fSAMPLING=fDTS/2，N=8
0110：采样频率 fSAMPLING=fDTS/4，N=6
0111：采样频率 fSAMPLING=fDTS/4，N=8
1000：采样频率 fSAMPLING=fDTS/8，N=6
1001：采样频率 fSAMPLING=fDTS/8，N=8
1010：采样频率 fSAMPLING=fDTS/16，N=5
1011：采样频率 fSAMPLING=fDTS/16，N=6
1100：采样频率 fSAMPLING=fDTS/16，N=8
1101：采样频率 fSAMPLING=fDTS/32，N=5
1110：采样频率 fSAMPLING=fDTS/32，N=6
1111：采样频率 fSAMPLING=fDTS/32，N=8</td></tr>
<tr><td>7</td><td>MSM</td><td>MSM：主/从模式(Master/slave mode)
0：无作用；
1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过 TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。</td></tr>
<tr><td rowspan="2">6:4</td><td rowspan="2">TS[2:0]</td><td>TS[2:0]：触发选择(Trigger selection)
这 3 位选择用于同步计数器的触发输入。
000：内部触发 0(ITR0)
001：内部触发 1(ITR1)
010：内部触发 2(ITR2)
011：内部触发 3(ITR3)
100：TI1 的边沿检测器(TI1F_ED)
101：滤波后的定时器输入 1(TI1FP1)
110：滤波后的定时器输入 2(TI2FP2)
111：外部触发输入(ETRF)</td></tr>
<tr><td>更多有关 ITRx 的细节，参见表 72。
注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。</td></tr>
<tr><td>3</td><td>Reserved</td><td>保留，始终读为 0。</td></tr>
<tr><td>2:0</td><td>SMS[2:0]</td><td>SMS[2:0]：从模式选择(Slave mode selection)
当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明)
000：关闭从模式–如果 CEN=1，则预分频器直接由内部时钟驱动。
001：编码器模式 1–根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。
010：编码器模式 2–根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。</td></tr>
</table>

		011：编码器模式 3–根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100：复位模式–选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式–当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式–计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1–选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。

表 72 TIMx 内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM1	TIM5_TROG	TIM2_TROG	TIM3_TROG	TIM4_TROG
TIM8	TIM1_TROG	TIM2_TROG	TIM4_TROG	TIM5_TROG

13.4.4 TIM1 和 TIM8DMA/中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

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

位	符号	说明
15	Reserved	保留，始终读为 0。
14	TDE	**TDE**：允许触发 DMA 请求 (Trigger DMA request enable) 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。
13	COMDE	**COMDE**：允许 COM 的 DMA 请求 (COM DMA request enable) 0：禁止 COM 的 DMA 请求； 1：允许 COM 的 DMA 请求。
12	CC4DE	**CC4DE**：允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。
11	CC3DE	**CC3DE**：允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
10	CC2DE	**CC2DE**：允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。
9	CC1DE	**CC1DE**：允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
8	UDE	**UDE**：允许更新的 DMA 请求 (Update DMA request enable) 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
7	BIE	**BIE**：允许刹车中断 (Break interrupt enable) 0：禁止刹车中断； 1：允许刹车中断。
6	TIE	**TIE**：触发中断使能 (Trigger interrupt enable)

		0：禁止触发中断； 1：使能触发中断。
5	COMIE	**COMIE**：允许 COM 中断 (COM interrupt enable) 0：禁止 COM 中断； 1：允许 COM 中断。
4	CC4IE	**CC4IE**：允许捕获/比较 4 中断 (Capture/Compare 4 interrupt enable) 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。
3	CC3IE	**CC3IE**：允许捕获/比较 3 中断 (Capture/Compare 3 interrupt enable) 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
2	CC2IE	**CC2IE**：允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
1	CC1IE	**CC1IE**：允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
0	UIE	**UIE**：允许更新中断 (Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

13.4.5 TIM1 和 TIM8 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留	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

位	符号	说明
15:13	Reserved	保留，始终读为 0。
12	CC4OF	**CC4OF**： 捕获/比较 4 重复捕获标记 (Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	**CC3OF**： 捕获/比较 3 重复捕获标记 (Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	**CC2OF**： 捕获/比较 2 重复捕获标记 (Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	**CC1OF**： 捕获/比较 1 重复捕获标记 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置 1。写 0 可清除该位。 0：无重复捕获产生； 1：计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。
8	Reserved	保留，始终读为 0。
7	BIF	**BIF**： 刹车中断标记 (Break interrupt flag) 一旦刹车输入有效，由硬件对该位置'1'。如果刹车输入无效，则该位可由软件清'0'。 0：无刹车事件产生； 1：刹车输入上检测到有效电平。
6	TIF	**TIF**： 触发器中断标记 (Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效 边沿，或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发中断等待响应。

5	COMIF	**COMIF**： COM 中断标记 (COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位：CCxE 、CCxNE 、OCxM 已被更新)该位由硬件置'1'。 它由软件清'0'。 0：无 COM 事件产生； 1： COM 中断等待响应。
4	CC4IF	**CC4IF**： 捕获/比较 4 中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	**CC3IF**： 捕获/比较 3 中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	**CC2IF**： 捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	**CC1IF**： 捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外(参考 TIMx_CR1 寄存器 的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1： TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 的内容时，在向上或向上/下计数模式时计数器溢出，或 向下计数模式时的计数器下溢条件下，CC1IF 位变高 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。 0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	**UIF**： 更新中断标记 (Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： - 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时(重复计数器=0 时产生更新事件)。 - 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 - 若 TIMx_CR1 寄存器的 URS=0 、UDIS=0，当计数器 CNT 被触发事件重新初始化时。(参考 12.4.3: TIM1 和 TIM8 从模式控制寄存器(TIMx_SMCR))。

13.4.6 TIM1 和 TIM8 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

位	符号	说明
15:8	Reserved	保留，始终读为 0。
7	BG	**BG**： 产生刹车事件 (Break generation) 该位由软件置'1'，用于产生一个刹车事件，由硬件自动清'0'。 0：无动作； 1：产生一个刹车事件。此时 MOE=0 、BIF=1，若开启对应的中断和 DMA，则产生相应的中断 和 DMA。
6	TG	**TG**： 产生触发事件 (Trigger generation)

		该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1 ： TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5	**COMG**	**COMG** ： 捕获/比较事件，产生控制更新 (Capture/Compare control update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：当 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 注：该位只对拥有互补输出的通道有效。
4	**CC4G**	**CC4G** ： 产生捕获/比较 4 事件 (Capture/Compare 4 generation) 参考 CC1G 描述。
3	**CC3G**	**CC3G** ： 产生捕获/比较 3 事件 (Capture/Compare 3 generation) 参考 CC1G 描述。
2	**CC2G**	**CC2G** ： 产生捕获/比较 2 事件 (Capture/Compare 2 generation) 参考 CC1G 描述。
1	**CC1G**	**CC1G** ： 产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 CC1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则 产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	**UG**	**UG** ： 产生更新事件 (Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系 数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'；若 DIR=1(向下计数)则计数 器取 TIMx_ARR 的值。

13.4.7 TIM1 和 TIM8 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位	符号	说明
15	**OC2CE**	**OC2CE**：输出比较 2 清 0 使能 (Output Compare 2 clear enable)
14:12	**OC2M[2:0]**	**OC2M[2:0]**：输出比较 2 模式 (Output Compare 2 mode)
11	**OC2PE**	**OC2PE**：输出比较 2 预装载使能 (Output Compare 2 preload enable)
10	**OC2FE**	**OC2FE**：输出比较 2 快速使能 (Output Compare 2 fast enable)
9:8	**CC2S[1:0]**	**CC2S[1:0]**：捕获/比较 2 选择。(Capture/Compare 2 selection) 该位定义通道的方向(输入/输出)，及输入脚的选择：

		00 ： CC2 通道被配置为输出； 01 ： CC2 通道被配置为输入，IC2 映射在 TI2 上； 10 ： CC2 通道被配置为输入，IC2 映射在 TI1 上； 11 ： CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	**OC1CE**	**OC1CE**：输出比较 1 清'0'使能 (Output Compare 1 clear enable) 0 ： OC1REF 不受 ETRF 输入的影响； 1 ： 一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
6:4	**OC1M[2:0]**	**OC1M[2:0]**：输出比较 1 模式 (Output Compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 、OC1N 的值。 OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001 ： 匹配 时 设置通道 1 为 有效电平。 当计数器 TIMx_CNT 的值与 捕获/ 比较寄 存器 1 (TIMx_CCR1)相同时，强制 OC1REF 为高。 010 ： 匹配 时 设置通道 1 为 无效电平。 当计数器 TIMx_CNT 的值与 捕获/ 比较寄 存器 1 (TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110 ： PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为 无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平 (OC1REF=0)，否 则为有效电平(OC1REF=1)。 111 ： PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为 有效电平；在向下计数时， 一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电 平。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输 出)则该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式 切换到 PWM 模式时，OC1REF 电平才改变。
3	**OC1PE**	**OC1PE** ： 输出比较 1 预装载使能 (Output Compare 1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器， 并且新写入的数 值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的 预装载值在更新事件到来时被加载至当前寄存器中。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S=00(该通道配置成输 出)则该位不能被修改。 注 2：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM=1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	**OC1FE**	**OC1FE** ： 输出比较 1 快速使能 (Output Compare 1 fast enable) 该位用于加快 CC 输出对触发输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有 效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与 比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 OCFE 只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1:0	**CC1S[1:0]**	**CC1S[1:0]**：捕获/比较 1 选择。(Capture/Compare 1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC1 通道被配置为输出；

		01 ： CC1 通道被配置为输入，IC1 映射在 TI1 上； 10 ： CC1 通道被配置为输入，IC1 映射在 TI2 上； 11 ： CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

输入捕获模式：

位	符号	说明
15:12	IC2F[3:0]	**IC2F[3:0]**：输入捕获 2 滤波器 (Input capture 2 filter)
11:10	IC2PSC [1:0]	**IC2PSC[1:0]**：输入/捕获 2 预分频器 (Input capture 2 prescaler)
9:8	CC2S[1:0]	**CC2S[1:0]**：捕获/比较 2 选择 (Capture/Compare 2 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC2 通道被配置为输出； 01 ： CC2 通道被配置为输入，IC2 映射在 TI2 上； 10 ： CC2 通道被配置为输入，IC2 映射在 TI1 上； 11 ： CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7:4	IC1F[3:0]	**IC1F[3:0]**：输入捕获 1 滤波器 (Input capture 1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记 录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1000：采样频率 fSAMPLING=fDTS/8，N=6 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6 1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC [1:0]	**IC1PSC[1:0]**：输入/捕获 1 预分频器 (Input capture 1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。一旦 CC1E=0(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
1:0	CC1S[1:0]	**CC1S[1:0]** ： 捕获/比较 1 选择 (Capture/Compare 1 Selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC1 通道被配置为输出； 01 ： CC1 通道被配置为输入，IC1 映射在 TI1 上； 10 ： CC1 通道被配置为输入，IC1 映射在 TI2 上； 11 ： CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。

13.4.8 TIM1 和 TIM8 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位	符号	说明
15	OC4CE	**OC4CE**：输出比较 4 清 0 使能 (Output compare 4 clear enable)
14:12	OC4M[2:0]	**OC4M[2:0]**：输出比较 4 模式 (Output compare 4 mode)
11	OC4PE	**OC4PE**：输出比较 4 预装载使能 (Output compare 4 preload enable)
10	OC4FE	**OC4FE**：输出比较 4 快速使能 (Output compare 4 fast enable)
9:8	CC4S[1:0]	**CC4S[1:0]**：捕获/比较 4 选择 (Capture/Compare 4 selection) 该 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC4 通道被配置为输出； 01 ： CC4 通道被配置为输入，IC4 映射在 TI4 上； 10 ： CC4 通道被配置为输入，IC4 映射在 TI3 上； 11 ： CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7	OC3CE	**OC3CE**：输出比较 3 清 0 使能 (Output compare 3 clear enable)
6:4	OC3M[2:0]	**OC3M[2:0]**：输出比较 3 模式 (Output compare 3 mode)
3	OC3PE	**OC3PE**：输出比较 3 预装载使能 (Output compare 3 preload enable)
2	OC3FE	**OC3FE**：输出比较 3 快速使能 (Output compare 3 fast enable)
1:0	CC3S[1:0]	**CC3S[1:0]**：捕获/比较 3 选择 (Capture/Compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC3 通道被配置为输出； 01 ： CC3 通道被配置为输入，IC3 映射在 TI3 上； 10 ： CC3 通道被配置为输入，IC3 映射在 TI4 上； 11 ： CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

输入捕获模式：

位	符号	说明
15:12	IC4F[3:0]	**IC4F[3:0]**：输入捕获 4 滤波器 (Input capture 4 filter)
11:10	IC4PSC [1:0]	**IC4PSC[1:0]**：输入/捕获 4 预分频器 (Input capture 4 prescaler)
9:8	CC4S[1:0]	**CC4S[1:0]**：捕获/比较 4 选择 (Capture/Compare 4 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC4 通道被配置为输出； 01 ： CC4 通道被配置为输入，IC4 映射在 TI4 上； 10 ： CC4 通道被配置为输入，IC4 映射在 TI3 上；

		11 ： CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7:4	**IC3F[3:0]**	**IC3F[3:0]**：输入捕获 3 滤波器 (Input capture 3 filter)
3:2	**IC3PSC [1:0]**	**IC3PSC[1:0]**：输入/捕获 3 预分频器 (Input capture 3 prescaler)
1:0	**CC3S[1:0]**	**CC3S[1:0]**：捕获/比较 3 选择 (Capture/compare 3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00 ： CC3 通道被配置为输出； 01 ： CC3 通道被配置为输入，IC3 映射在 TI3 上； 10 ： CC3 通道被配置为输入，IC3 映射在 TI4 上； 11 ： CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

13.4.9 TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	保留	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

位	符号	说明
15	**CC4NP**	**CC4NP:**输入/捕获 4 输出极性,参考 CC1NP 的描述。
14	**Reserved**	保留，始终读为 0。
13	**CC4P**	**CC4P**：输入/捕获 4 输出极性 (Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	**CC4E**	**CC4E**：输入/捕获 4 输出使能 (Capture/Compare 4 output enable) 参考 CC1E 的描述。
11	**CC3NP**	**CC3NP**：输入/捕获 3 互补输出极性 (Capture/Compare 3 complementary output polarity) 参考 CC1NP 的描述。
10	**CC3NE**	**CC3NE**：输入/捕获 3 互补输出使能 (Capture/Compare 3 complementary output enable) 参考 CC1NE 的描述。
9	**CC3P**	**CC3P**：输入/捕获 3 输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	**CC3E**	**CC3E**：输入/捕获 3 输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
7	**CC2NP**	**CC2NP**：输入/捕获 2 互补输出极性 (Capture/Compare 2 complementary output polarity) 参考 CC1NP 的描述。
6	**CC2NE**	**CC2NE**：输入/捕获 2 互补输出使能 (Capture/Compare 2 complementary output enable) 参考 CC1NE 的描述。
5	**CC2P**	**CC2P**：输入/捕获 2 输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	**CC2E**	**CC2E**：输入/捕获 2 输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	**CC1NP**	**CC1NP**：输入/捕获 1 互补输出极性 (Capture/Compare 1 complementary output polarity) 0 ： OC1N 高电平有效； 1 ： OC1N 低电平有效。

		注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出) 则该位不能被修改。
2	**CC1NE**	**CC1NE**：输入/捕获 1 互补输出使能 (Capture/Compare 1 complementary output enable) 0：关闭 - OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI 、OSSR 、OIS1 、OIS1N 和 CC1E 位的值。 1 ： 开启 - OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE 、OSSI 、OSSR、OIS1、OIS1N 和 CC1E 位的值。
1	**CC1P**	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； 1 ： OC1 低电平有效。 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2，则该位不能被修改。
0	**CC1E**	**CC1E** ： 输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出： 0 ： 关闭 - OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE 、OSSI 、OSSR 、OIS1、OIS1N 和 CC1NE 位的值。 1 ： 开启 - OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE 、OSSI 、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 0：捕获使能。

表 73 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

控制位					**输出状态**(1)	
MOE 位	OSSI 位	OSSR 位	CCxE 位	CCxNE 位	OCx 输出状态	OCxN 输出状态
1	X	0	0	0	输出禁止(与定时器断开) OCx=0，OCx_EN=0	输出禁止(与定时器断开) OCxN=0，OCxN_EN=0
		0	0	1	输出禁止(与定时器断开) OCx=0，OCx_EN=0	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		0	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	输出禁止(与定时器断开) OCxN=0，OCxN_EN=0
		0	1	1	OCxREF + 极性 + 死区， OCx_EN=1	OCxREF 反相 + 极性 + 死区， OCxN_EN=1
		1	0	0	输出禁止(与定时器断开) OCx=CCxP，OCx_EN=0	输出禁止(与定时器断开) OCxN=CCxNP ，OCxN_EN=0
		1	0	1	关闭状态(输出使能且为无效 电平) OCx=CCxP,OCx_EN=1	OCxREF + 极性， OCxN= OCxREF xor CCxNP， OCxN_EN=1
		1	1	0	OCxREF + 极性， OCx= OCxREF xor CCxP， OCx_EN=1	关闭状态(输出使能且为无效电 平) OCxN=CCxNP ，OCxN_EN=1
		1	1	1	OCxREF + 极性 + 死区， OCx_EN=1	OCxREF 反相 + 极性 + 死区， OCxN_EN=1
0	0	X	0	0	输出禁止(与定时器断开) 异 步 地 ： OCx=CCxP ， OCx_EN=0 ，OCxN=CCxNP ， OCxN_EN=0；	
	0		0	1		
	0		1	0		
	0		1	1		

					若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。
	1		0	0	关闭状态(输出使能且为无效电平)
	1		0	1	异步地：OCx=CCxP，OCx_EN=1，OCxN=CCxNP，OCxN_EN=1；
	1		1	0	
	1		1	1	若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。

1. 如果一个通道的 2 个输出都没有使用(CCxE = CCxNE = 0)，那么 OISx，OISxN，CCxP 和 CCxNP 都必须清零。

注：引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

13.4.10 TIM1 和 TIM8 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

位	符号	说明
15:0	CNT[15:0]	**CNT[15:0]**：计数器的值 (Counter value)

13.4.11 TIM1 和 TIM8 预分频器(TIMx_PSC)

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

位	符号	说明
15:0	PSC[15:0]	**PSC[15:0]**：预分频器的值 (Prescaler value) 计数器的时钟频率(CK_CNT)等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

13.4.12 TIM1 和 TIM8 自动重装载寄存器(TIMx_ARR)

偏移地址:0x2C

复位值:0xFFFF

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

位	符号	说明
15:0	ARR[15:0]	**ARR[15:0]:** 自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。详细参考 13.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

13.4.13 TIM1 和 TIM8 重复计数寄存器(TIMx_RCR)

偏移地址：0x30

复位值：0x0000

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

位	符号	说明
15:8	Reserved	保留，始终读为 0。
7:0	REP[7:0]	**REP[7:0]:** 重复计数器的值 (Repetition counter value) 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率(即周期性地从预装载寄存器传输到当前寄存器)；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1)对应着： - 在边沿对齐模式下，PWM 周期的数目； - 在中心对称模式下，PWM 半周期的数目；

13.4.14 TIM1 和 TIM8 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:0	CCR1[15:0]	**CCR1[15:0]:** 捕获/比较通道 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

13.4.15 TIM1 和 TIM8 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:0	CCR2[15:0]	**CCR2[15:0]:** 捕获/比较通道 2 的值 (Capture/Compare 2 value)

		若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。

13.4.16 TIM1 和 TIM8 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位	符号	说明
15:0	CCR3[15:0]	**CCR3[15:0]**: 捕获/比较通道 3 的值 (Capture/Compare 3 value) 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。

13.4.17 TIM1 和 TIM8 捕获/比较寄存器(TIMx_CCR4)

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:0	CCR4[15:0]	**CCR4[15:0]**: 捕获/比较通道 4 的值 (Capture/Compare 4 value) 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。

13.4.18 TIM1 和 TIM8 刹车和死区寄存器(TIMx_BDTR)

偏移地址：0x44

复位值：0x0000

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

根据锁定设置，AOE、BKP、BKE、OSSI、OSSR 和 DTG[7:0]位均可被写保护，有必要在第一次写入 TIMx_BDTR 寄存器时对它们进行配置。

位	符号	说明
15	MOE	**MOE:** 主输出使能 (Main output enable) 一旦刹车输入有效，该位被硬件异步清'0'。根据 AOE 位的设置值，该位可以由软件清'0'或被自 动置 1。它仅对配置为输出的通道有效。 0：禁止 OC 和 OCN 输出或强制为空闲状态； 1：如果设置了相应的使能位(TIMx_CCER 寄存器的 CCxE 、CCxNE 位)，则开启 OC 和 OCN 输 出。 有关 OC/OCN 使能的细节，参见 13.4.9 节，TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER)。
14	AOE	**AOE:** 自动输出使能 (Automatic output enable) 0 ： MOE 只能被软件置'1'； 1 ： MOE 能被软件置'1'或在下一个更新事件被自动置'1'(如果刹车输入无效)。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。
13	BKP	**BKP:** 刹车输入极性 (Break polarity) 0：刹车输入低电平有效； 1：刹车输入高电平有效。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
12	BKE	**BKE:** 刹车功能使能 (Break enable) 0：禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1：开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
11	OSSR	**OSSR:** 运行模式下 “关闭状态” 选择 (Off-stateselection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(13.4.9 节，TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，首先开启 OC/OCN 并输出无效电平，然后 置 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	**OSSI:** 空闲模式下 “关闭状态” 选择 (Off-stateselection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(13.4.9 节，TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER))。 0：当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1 ： 当定时器不工作时， 一旦 CCxE=1 或 CCxNE=1 ， OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOOK[1:0]	**LOOK[1:0]:** 锁定设置 (Lock configuration) 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别 1，不能写入 TIMx_BDTR 寄存器的 DTG 、BKE 、BKP 、AOE 位和 TIMx_CR2 寄存 器的 OISx/OISxN 位； 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位(一旦相关通道通过

		CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 位； 11：锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位(一旦相关通道通过 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复 位。
7:0	**UTG[7:0]**	**UTG[7:0]**: 死区发生器设置 (Dead-time generator setup) 这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg ， Tdtg = TDTS； DTG[7:5]=10x => DT=(64+DTG[5:0]) × Tdtg ， Tdtg = 2 × TDTS ； DTG[7:5]=110 => DT=(32+DTG[4:0]) × Tdtg ， Tdtg = 8 × TDTS ； DTG[7:5]=111 => DT=(32+DTG[4:0])× Tdtg ， Tdtg = 16 × TDTS； 例：若 TDTS = 125ns(8MHZ)，可能的死区时间为： 0 到 15875ns，若步长时间为 125ns； 16us 到 31750ns，若步长时间为 250ns； 32us 到 63us，若步长时间为 1us； 64us 到 126us，若步长时间为 2us； 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 1 、2 或 3，则不能修改这些位。

13.4.19 TIM1 和 TIM8 DMA 控制寄存器(TIMx_DCR)

偏移地址：0x48

复位值：0x0000

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

位	符号	说明
15:13	**Reserved**	保留，始终读为 0。
12:8	**DBL[4:0]**	**DBL[4:0]**: DMA 连续传送长度 (DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则 进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节： 00000 ： 1 次传输　　00001 ： 2 次传输 00010 ： 3 次传输　　…… ……　　10001 ： 18 次传输 例：我们考虑这样的传输：DBL=7，DBA=TIM2_CR1 - 如果 DBL=7，DBA=TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出： (TIMx_CR1 的地址) + DBA + (DMA 索引), 其中 DMA 索引 = DBL 其中(TIMx_CR1 的地址) + DBA 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传 输将发生在从地址(TIMx_CR1 的地址) + DBA 开始的 7 个寄存器。 根据 DMA 数据长度的设置，可能发生以下情况： - 如果设置数据为半字(16 位)，那么数据就会传输给全部 7 个寄存器。 - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字 节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传 输的数据宽度。
7:5	**Reserved**	保留，始终读为 0。
4:0	**DBA[4:0]**	**DBA[4:0]**: DMA 基地址 (DMA base address) 这些位定义了 DMA 在连续模式下的基地址(当对 TIMx_DMAR 寄存器进行读或写时)， DBA 定义 为从 TIMx_CR1 寄存器所在地址开始的偏移量：

		00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， ……

13.4.20 TIM1 和 TIM8 连续模式的 DMA 地址(TIMx_DMAR)

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
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

位	符号	说明
15:0	DMAB [31:0]	**DMAB[31:0]**: DMA 连续传送寄存器 (DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： TIMx_CR1 地址 + DBA + DMA 索引，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

如何使用 DMA 突发特性的示例

在此示例中，定时器 DMA 突发特性用于更新 CCRx 寄存器的内容(x=2，3，4)，DMA 将半个字转移到 CCRx 寄存在中。

这是通过以下步骤完成的:

1.配置相应的 DMA 通道如下:

- DMA 通道外围地址是 DMAR 寄存器地址
- DMA 通道内存地址是 RAM 中的缓冲区地址，该缓冲区包含要由 DMA 传输到 CCRx 寄存器的数据。
- 要传输的数据数量=3(见下注)。
- 关闭圆形模式。

2. 通过配置 DBA 和 DBL 位字段来配置 DCR 寄存器如下:DBL=3 次传输，DBA=0xE
3. 3.使能 TIMx update DMA 请求(在 DIER 寄存器中设置 UDE 位)。
4. 4.启用 TIMx
5. 启用 DMA 通道

警告： 这个例子适用于每个 CCRx 寄存器需要更新一次的情况。如果每个 CCRx 寄存器需要更新两次，那么需要传输的数据数量应该是 6。以 RAM 中包含数据 1、数据 2、数据 3、数据 4、数据 5 和数据 6 的缓冲区为例。将数据传输到 CCRx 寄存器如下:在第一次更新 DMA 请求时，数据 1 被传输到

CCR2，数据 2 被传输到 CCR3，数据 3 被传输到 CCR4;在第二次更新 DMA 请求时，数据 4 被传输到 CCR2，数据 5 被传输到 CCR3，数据 6 被传输到 CCR4。

13.4.21 TIM1 和 TIM8 寄存器

下表中将 TIM1 和 TIM8 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 74 TIM1 和 TIM8 寄存器和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD [1:0]		APRE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	保留																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	保留	CCPC
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
008h	TIMx_SMCR	保留																ETP	ECE	ETPS [1:0]		EFT [3:0]				MSM	TS [2:0]			保留	SMS [2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
00Ch	TIMx_DIER	保留																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	TIMx_SR	保留																			CC4OF	CC3OF	CC2OF	CC1OF	保留	BIF	TIF	COM1F	CC41F	CC3IF	CC2IF	CC1IF	UTF
	复位值																				0	0	0	0		0	0	0	0	0	0	0	0
014h	TIMx_EGR	保留																								BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	复位值																									0	0	0	0	0	0	0	0
018h	TIMx_CCMR1 输出比较模式	保留																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 输入捕获模式	保留																	IC2F [3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F [3:0]			IC1 PSC [1:0]		CC1S [1:0]	
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIMx_CCMR2 输出比较模式	保留																OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值	保留																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 输入捕获模式	保留																IC4F [3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F [3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIMx_CCER	保留																CC4NP	保留	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	复位值																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	TIMx_RCR	保留																								REP[7:0]							
	复位值																									0	0	0	0	0	0	0	0
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	TIMx_BDTR	保留																MOE	AOE	BKE	BKP	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	TIMx_DCR	保留																				DBL[4:0]				保留				DBA[4:0]			
	复位值																					0	0	0	0					0	0	0	0
04Ch	TIMx_DMAR	DMAB[31:0]																															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，请参见表 1。

14 通用定时器(TIM2 到 TIM5)

14.1 TIM2 到 TIM5 简介

通用定时器是一个通过可编程预分频器驱动的 16 位自动装载计数器构成。

它适用于多种场合，包括测量输入信号的脉冲长度(输入捕获)或者产生输出波形(输出比较和 PWM)。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个 毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见 14.3.15 节。

14.2 TIM2 到 TIM5 主要功能

通用 TIMx (TIM2、TIM3、TIM4 和 TIM5)定时器功能包括：

- 16 位向上、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(边缘或中间对齐模式)
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

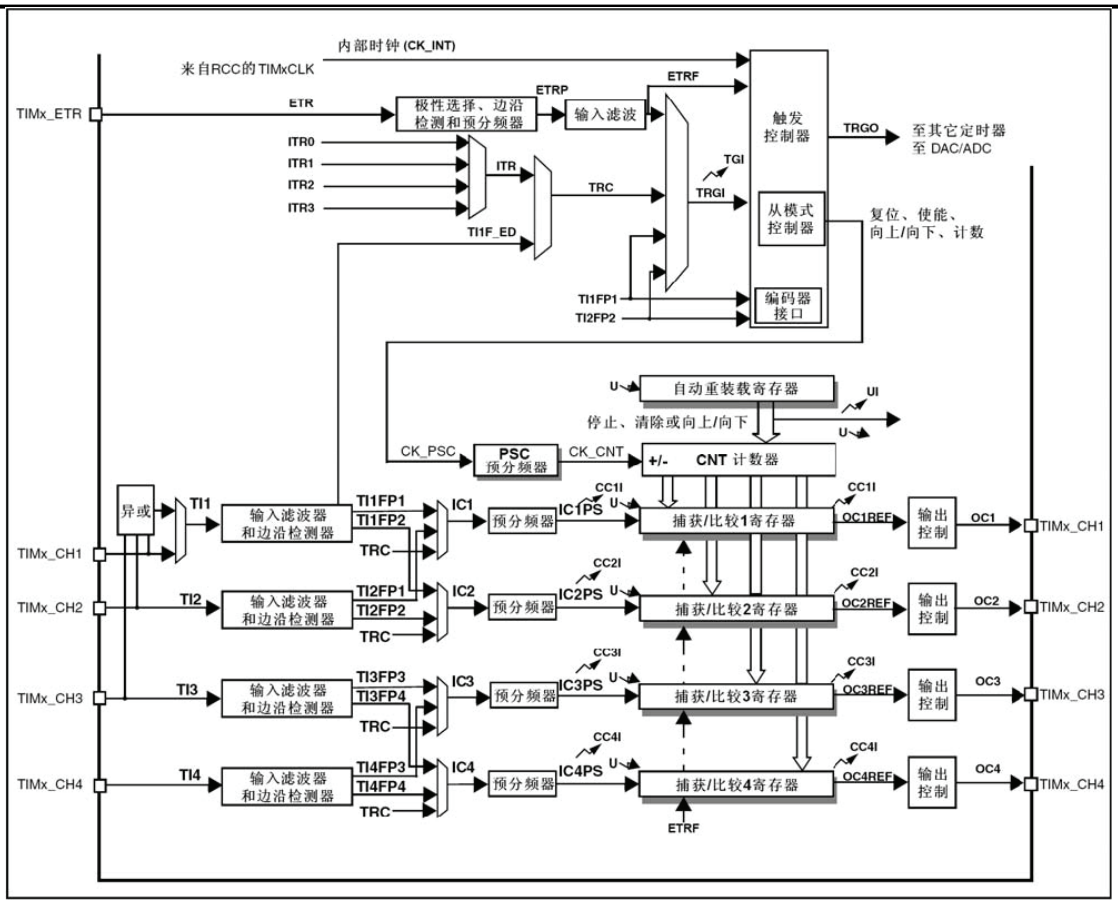


图 96 通用定时器框图

注： Reg *根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器*

事件

中断和 DMA 输出

14.3 TIM2 到 TIM5 功能描述

14.3.1 时基单元

可编程通用定时器的主要部分是一个16位计数器和与其相关的自动装载寄存器。这个计数器可 以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。 时基单元包含:

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动装载寄存器 (TIMx_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。 根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在 每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIMx_CR1 寄存器中的 UDIS 位等于'0'时，产生更新事件。更新事件也可以由软件产生。随后会 详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(有关计数器使能的细节，请参见控制器的从模式描述)。

注：　真正的计数器使能信号 CNT_EN 是在 CEN 的一个时钟周期后被设置。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

图 97 和图 98 给出了在预分频器运行时，更改计数器参数的例子。

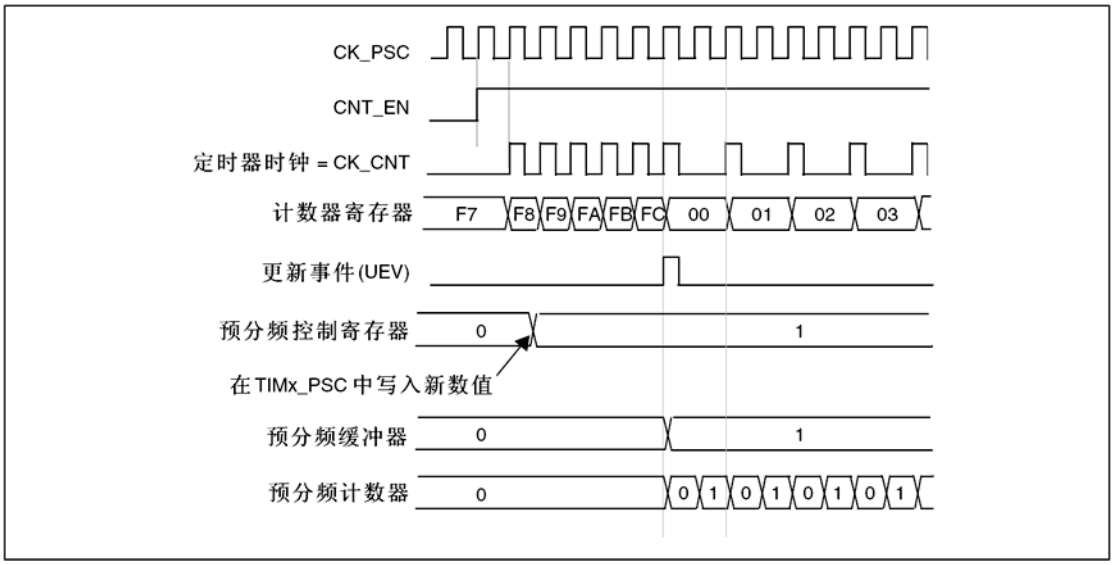


图 97 当预分频器的参数从 1 变到 2 时，计数器的时序图

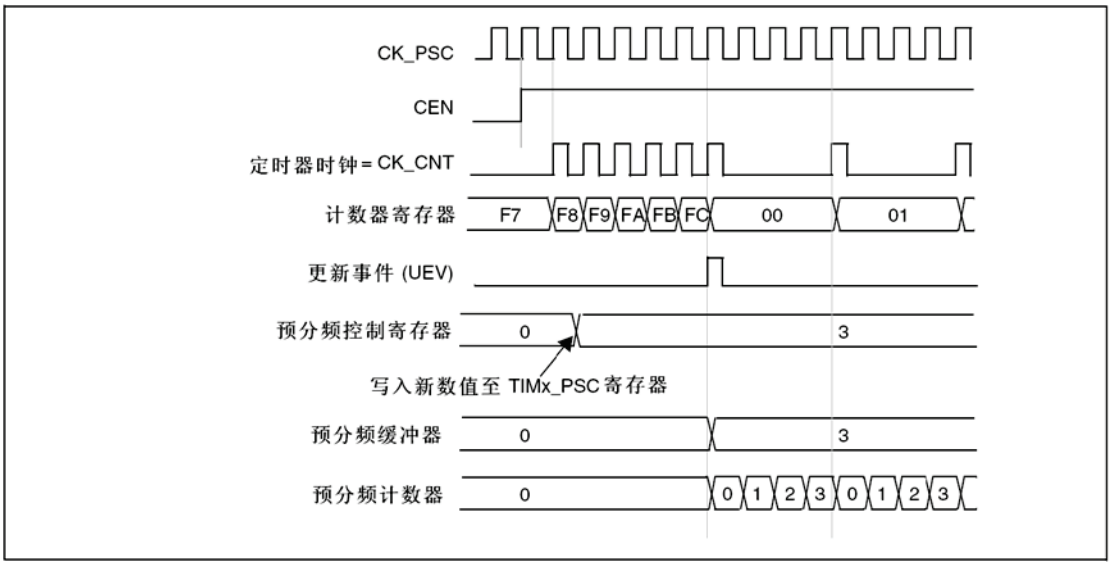


图 98 当预分频器的参数从 1 变到 4 时，计数器的时序图

14.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控 制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写 入新值时更新影子寄存器。在 UDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事 件时，计数器仍会被清'0'，同时预分频器的计数也被请 0(但预分频系数不变)。此外，如果设置 了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件 不设置 UIF 标志(即不产生中断或 DMA 请求)；这是为了避免在捕获模式下清除计数器时，同时产 生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

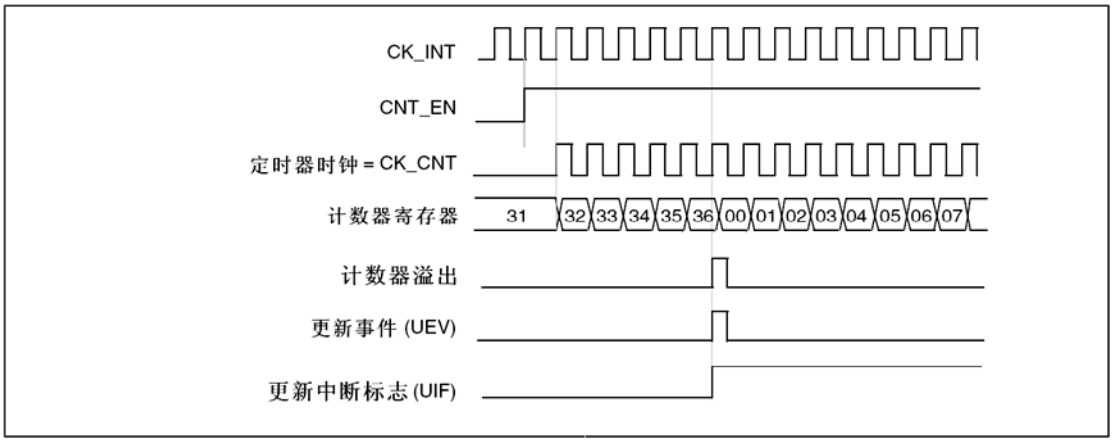


图 99 计数器时序图，内部时钟分频因子为 1

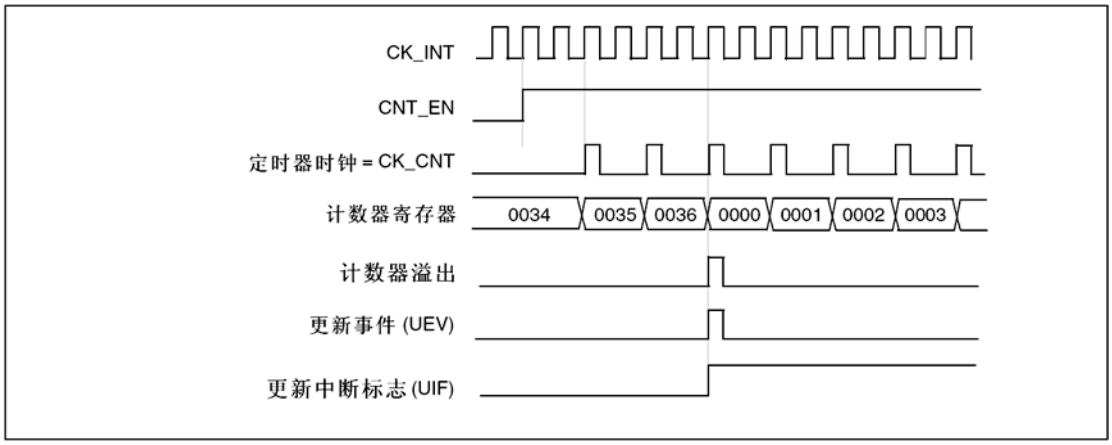


图 100 计数器时序图，内部时钟分频因子为 2

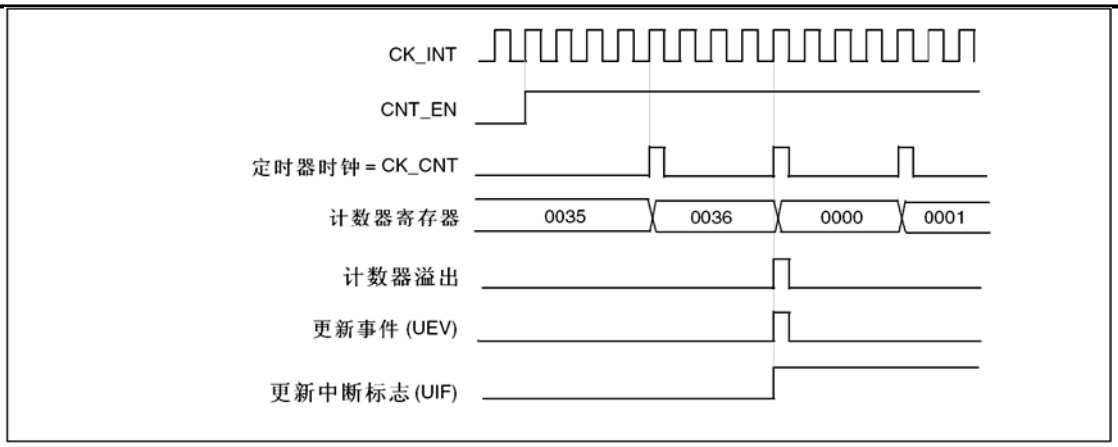


图 101 计数器时序图，内部时钟分频因子为 4

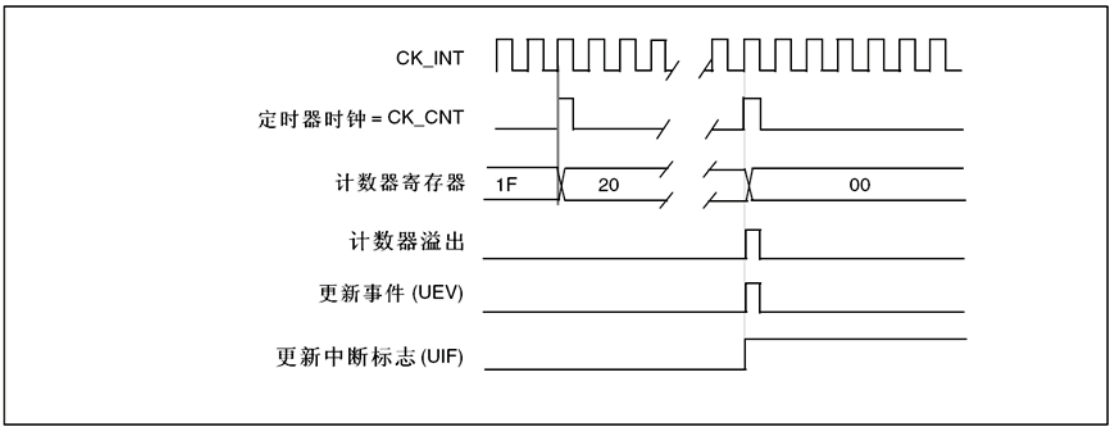


图 102 计数器时序图，内部时钟分频因子为 N

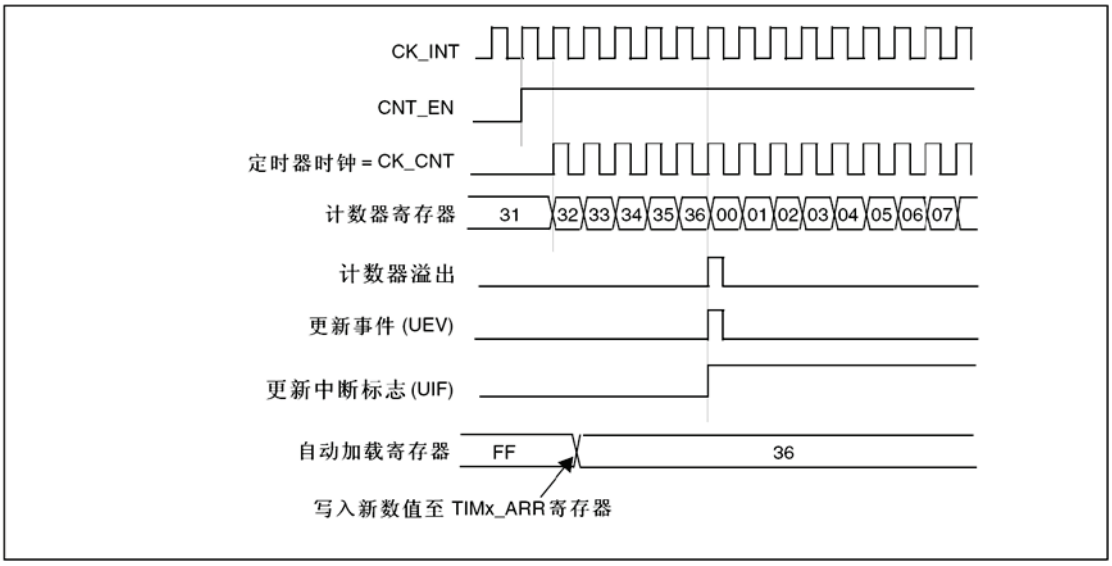


图 103 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

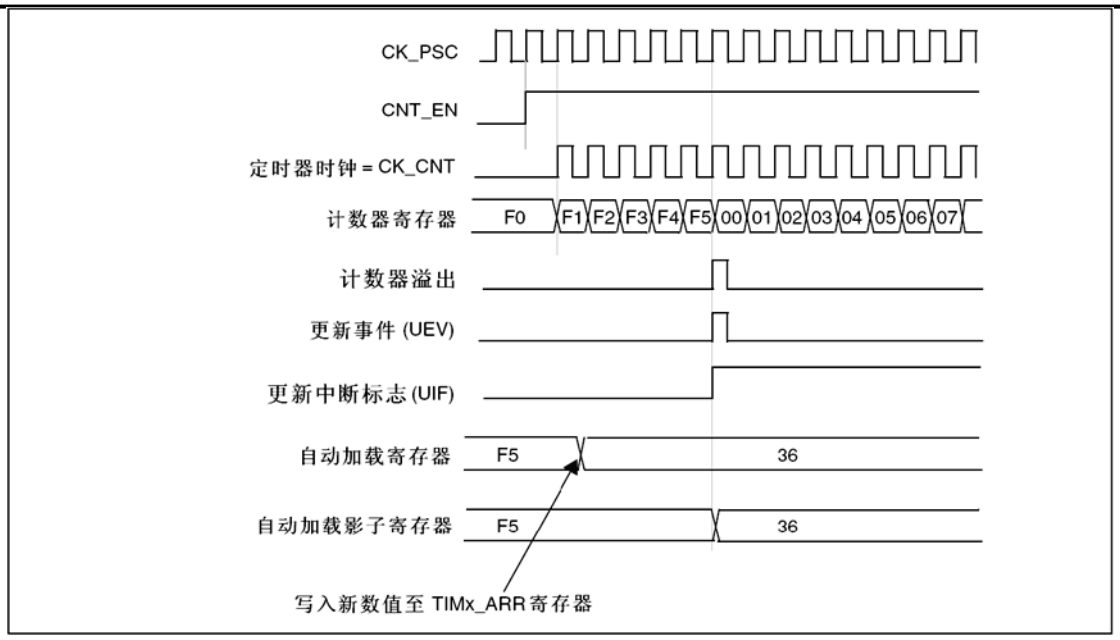


图 104 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)

向下计数模式

在向下模式中，计数器从自动装入的值(TIMx_ARR 计数器的值)开始向下计数到 0，然后从自动 装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控 制器)设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值 时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会从当前自 动加载值重新开始计数，同时预分频器的计数器重新从 0 开始(但预分频系数不变)。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄 存器中的 UIF 位)也被设置。

- 预分频器的缓存器被置入预装载寄存器的值(TIMx_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容) 。注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

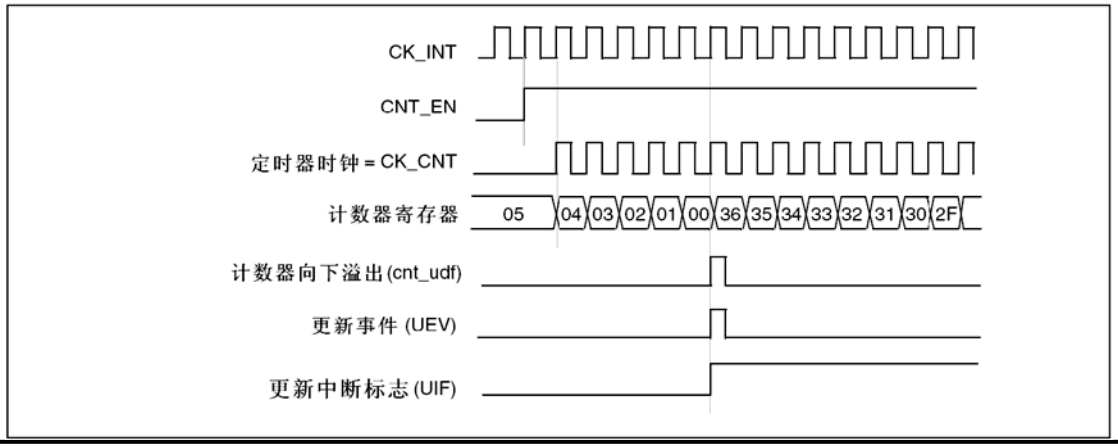


图 105 计数器时序图，内部时钟分频因子为 1

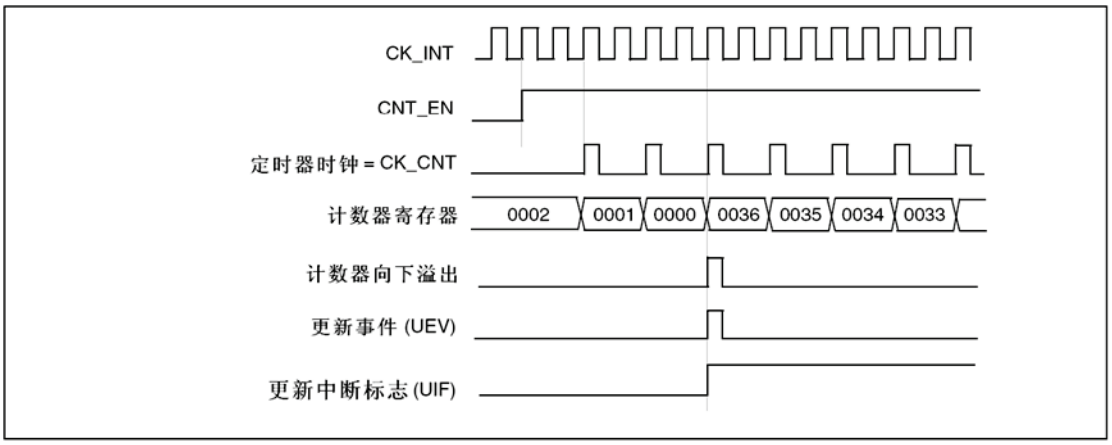


图 106 计数器时序图，内部时钟分频因子为 2

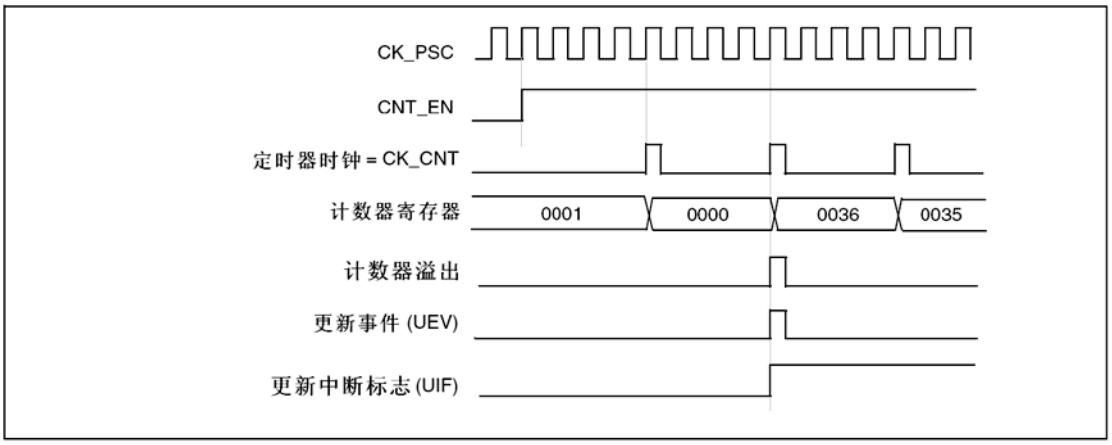


图 107 计数器时序图，内部时钟分频因子为 4

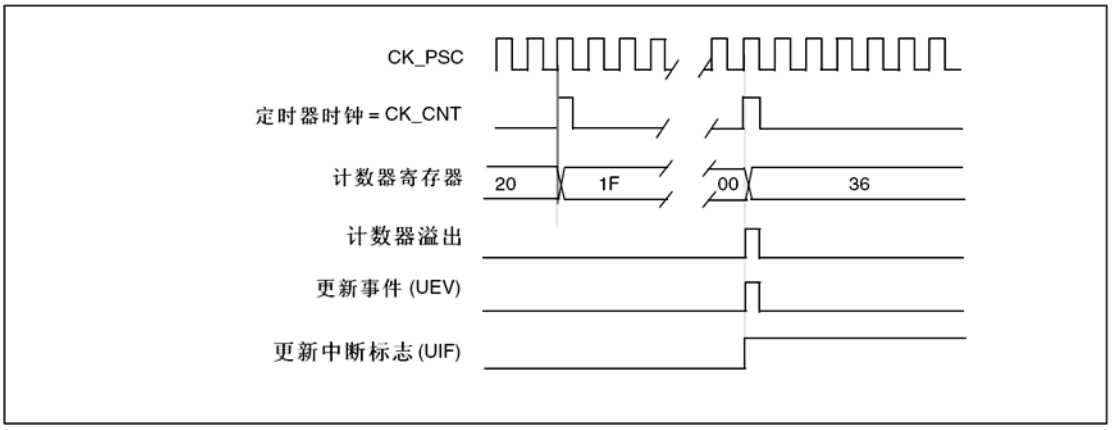


图 108 计数器时序图，内部时钟分频因子为 N

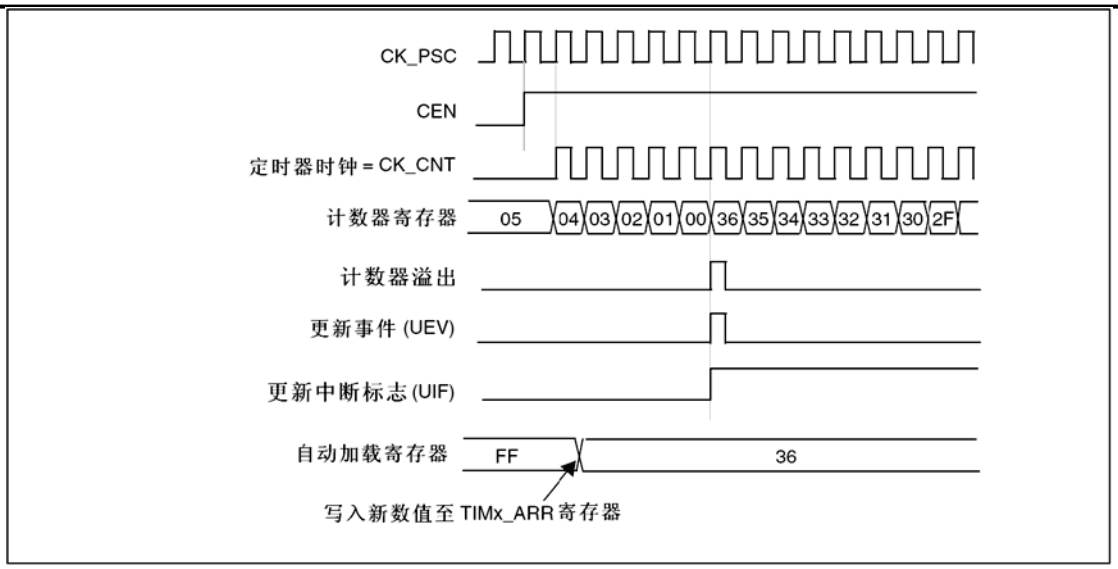


图 109 计数器时序图，当没有使用重复计数器时的更新事件

中央对齐模式(向上/向下计数)

在中央对齐模式，计数器从 0 开始计数到自动加载的值(TIMx_ARR 寄存器)-1，产生一个计数器 溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入 新值时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会根据 当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位(选择更新请求) ，设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx_SR 寄 存器中的 UIF 位)也被设置。

- 预分频器的缓存器被加载为预装载(TIMx_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx_ARR 寄存器中的内容)。

注：　如果因为计数 器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期 的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

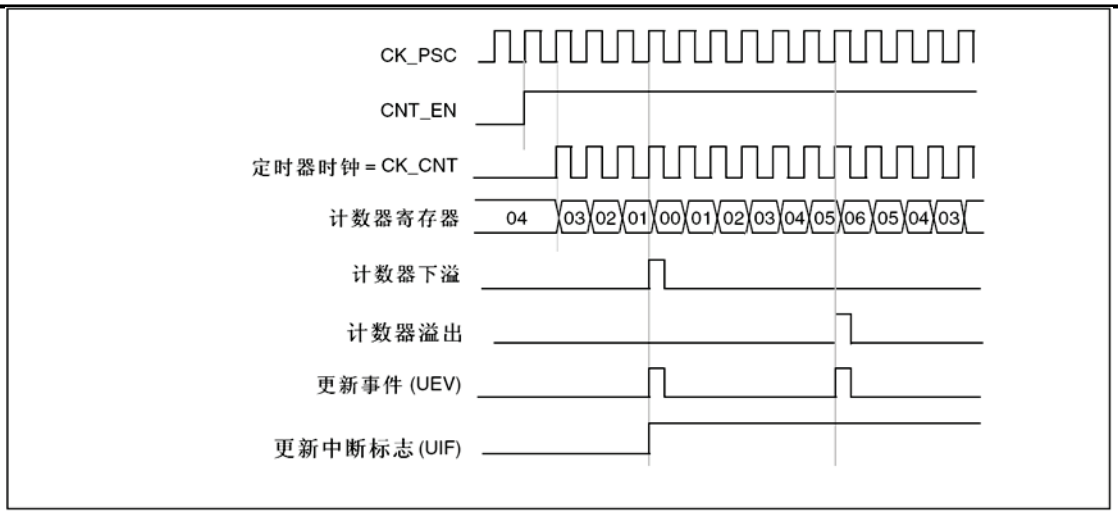


图 110 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

1. 这里使用了中心对齐模式 1(详见 14.4.1 节)。

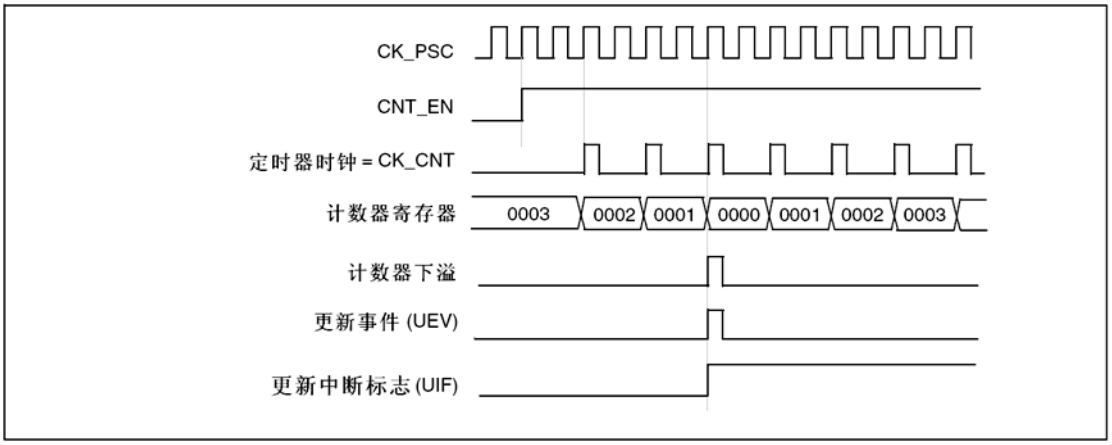


图 111 计数器时序图，内部时钟分频因子为 2

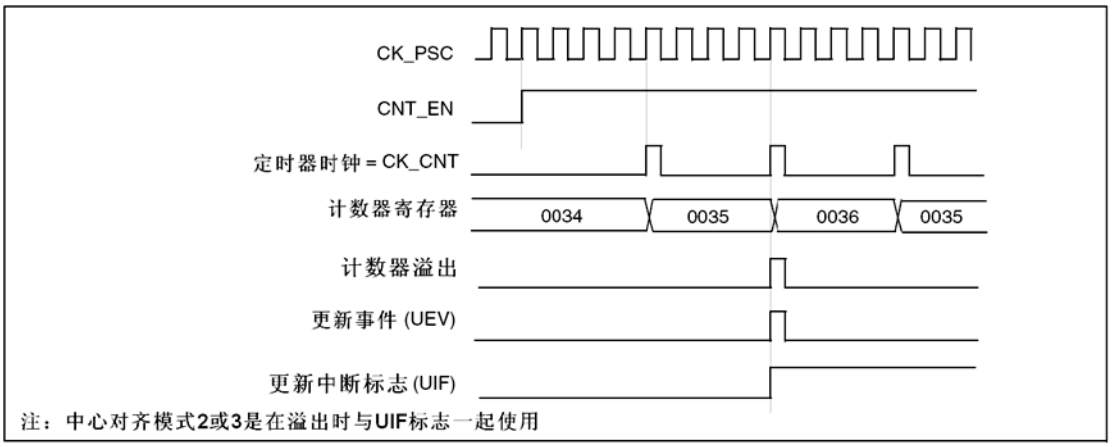


图 112 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

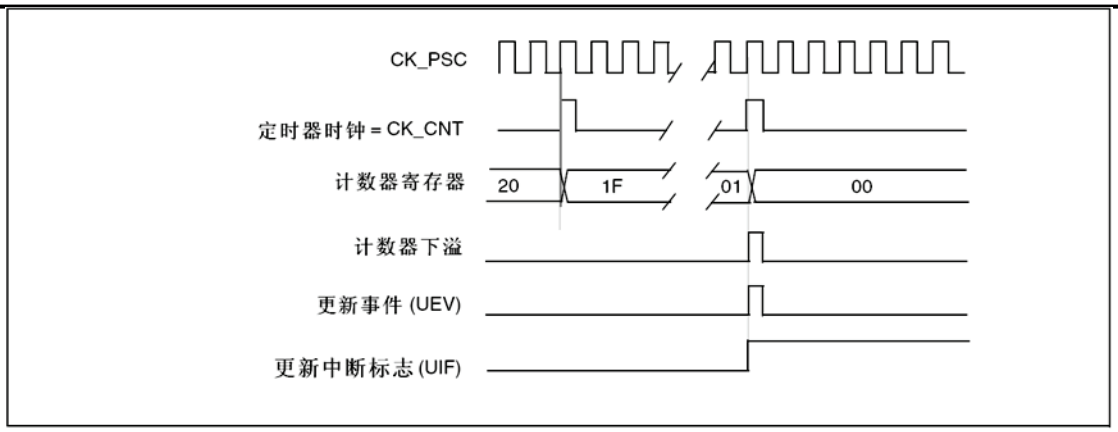


图 113 计数器时序图，内部时钟分频因子为 N

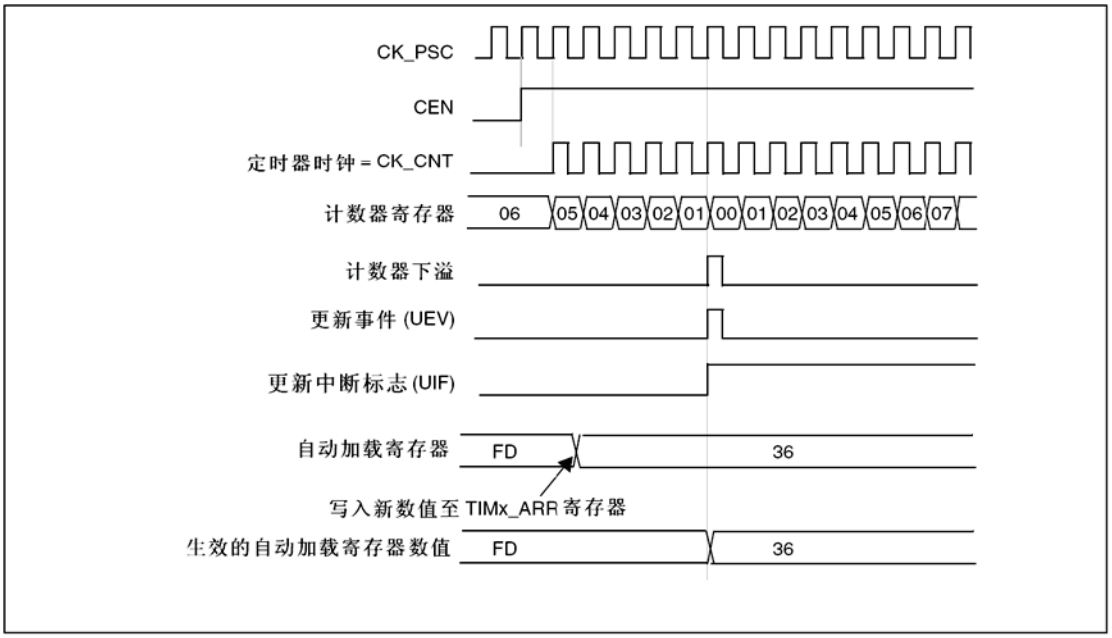


图 114 计数器时序图，ARPE=1 时的更新事件(计数器下溢)

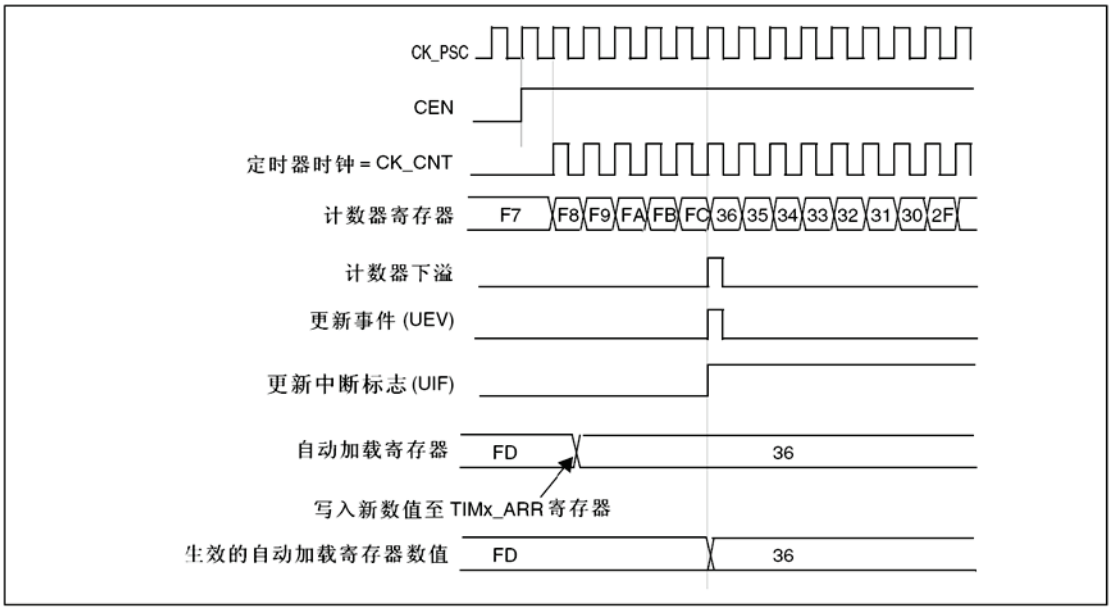


图 115 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

14.3.3 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK_INT)
- 外部时钟模式 1：外部输入脚(TIx)
- 外部时钟模式 2：外部触发输入(ETR)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。参见 13.3.15。

内部时钟源(CK_INT)

如果禁止了从模式控制器(TIMx_SMCR 寄存器的 SMS=000)，则 CEN 、DIR(TIMx_CR1 寄存器) 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只 要 CEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

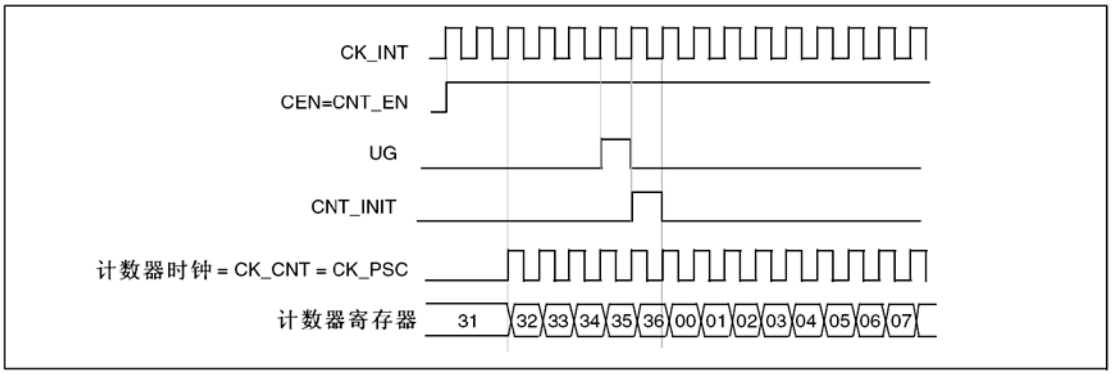


图 116 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿 或下降沿计数。

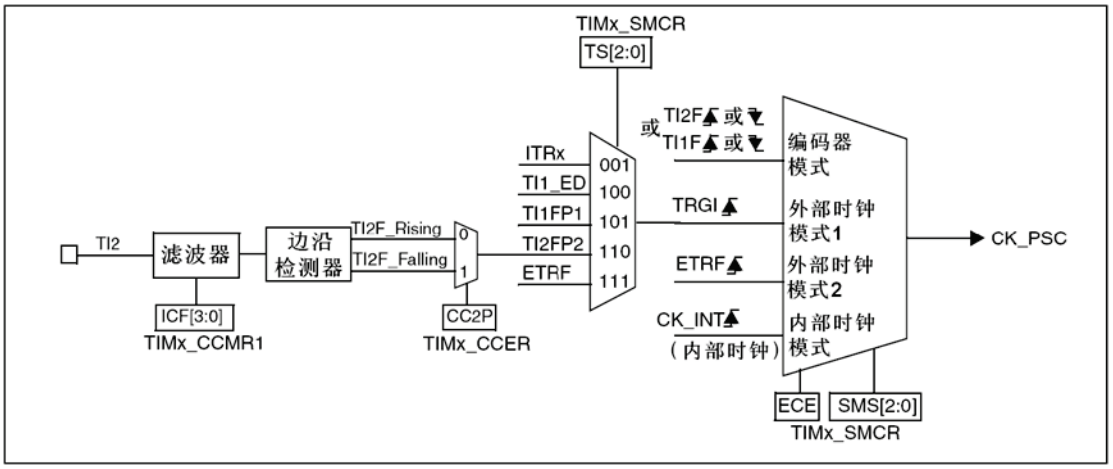


图 117 TI2 外部时钟连接例子

例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置 TIMx_CCMR1 寄存器 CC2S='01'，配置通道 2 检测 TI2 输入的上升沿
2. 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)

注：　捕获预分频器不用作触发，所以不需要对它进行配置

3. 配置 TIMx_CCER 寄存器的 CC2P='0'，选定上升沿极性
4. 配置 TIMx_SMCR 寄存器的 SMS='111'，选择定时器外部时钟模式 1
5. 配置 TIMx_SMCR 寄存器中的 TS='110'，选定 TI2 作为触发输入源
6. 设置 TIMx_CR1 寄存器的 CEN='1'，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

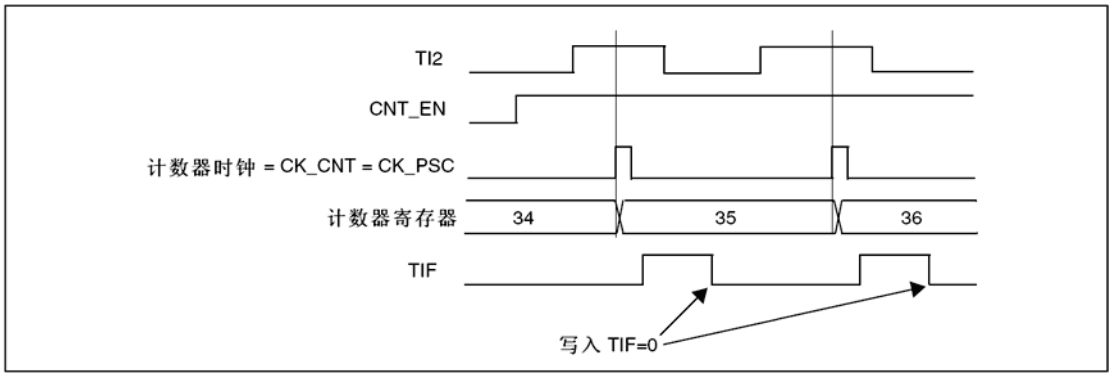


图 118 外部时钟模式 1 下的控制电路

外部时钟源模式 2

选定此模式的方法为：令 TIMx_SMCR 寄存器中的 ECE=1 计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。 下图是外部触发输入的框图

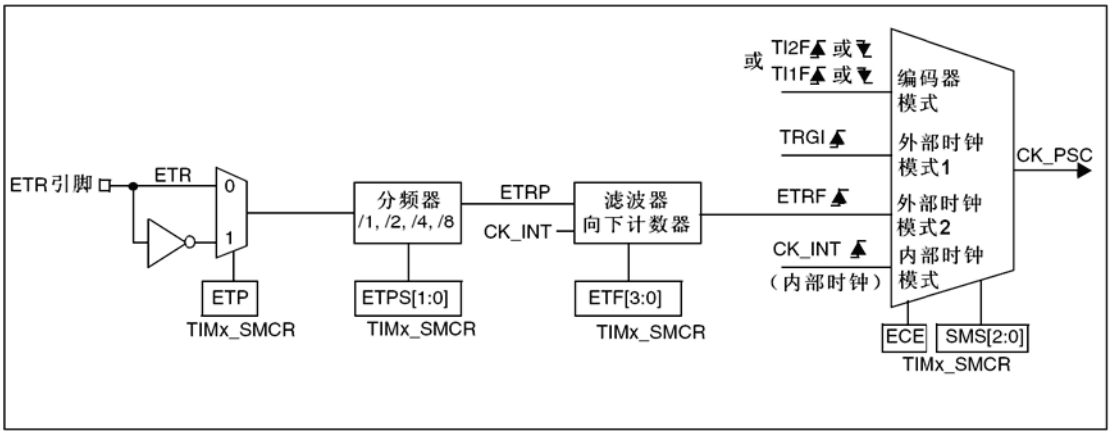


图 119 外部触发输入框图

例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000
2. 设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01
3. 设置在 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0
4. 开启外部时钟模式 2，置 TIMx_SMCR 寄存器中的 ECE=1
5. 启动计数器，置 TIMx_CR1 寄存器中的 CEN=1 计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

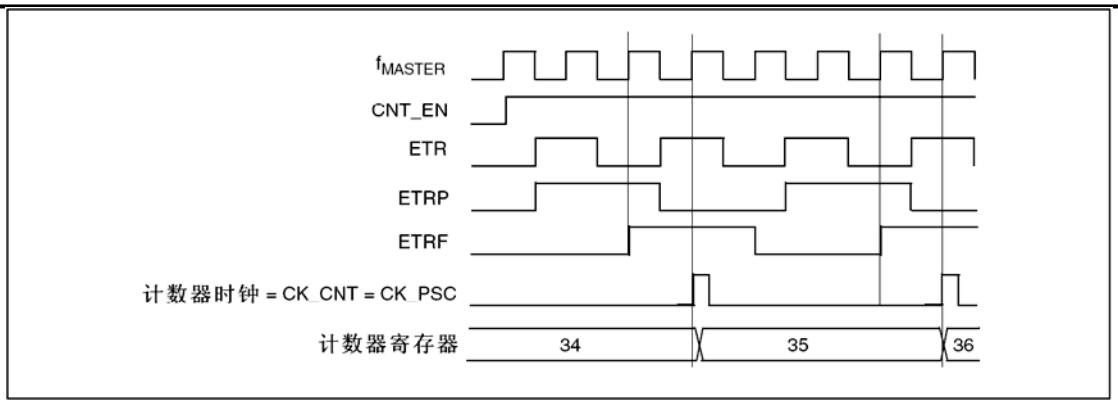


图 120 外部时钟模式 2 下的控制电路

14.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。 下面几张图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘检测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器(ICxPS)。

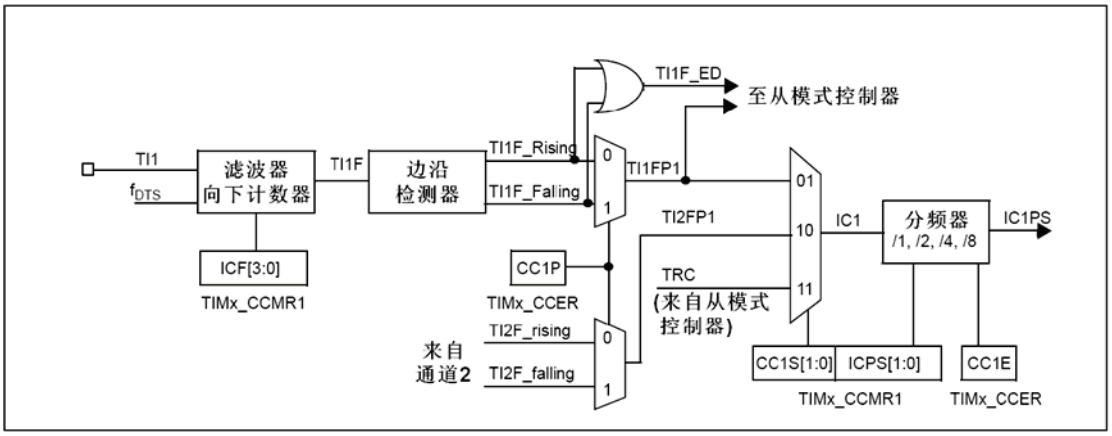


图 121 捕获/比较通道(如：通道 1 输入部分)

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

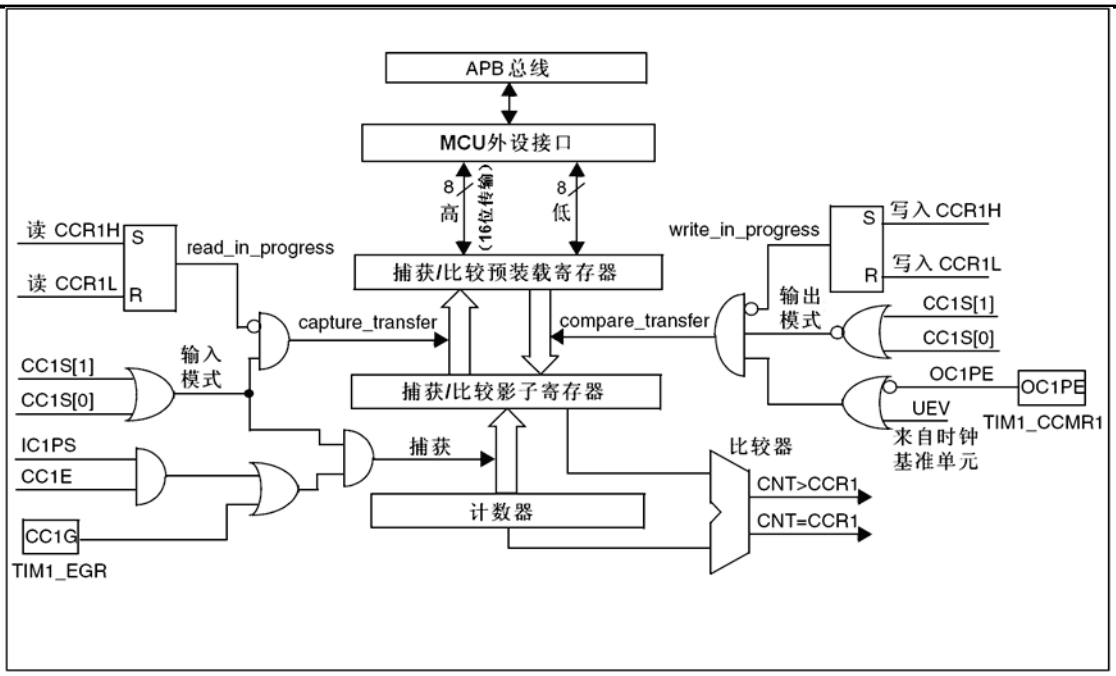


图 122 捕获/比较通道 1 的主电路

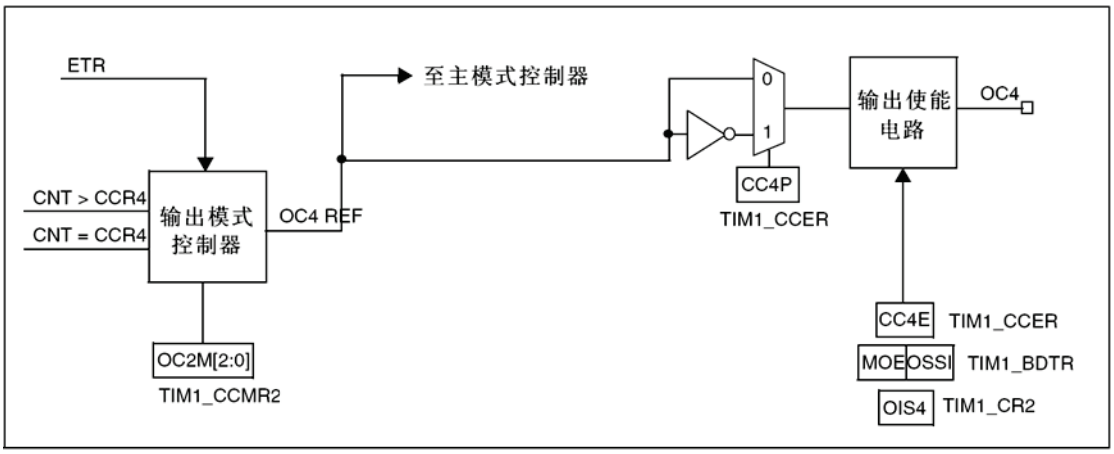


图 123 捕获/比较通道的输出部分(通道 1)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器 进行比较。

14.3.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当捕获事件发生时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置'1'，如果使能 了中断或者 DMA 操作， 则将产生中断或者 DMA 操作。如果捕获事件发生时 CCxIF 标志已经为 高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存 储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。

- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 fDTS 频率)连续采样 8 次，以确 认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预 分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除， CC1OF 也被置'1'。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢 出标志之后和读取数据之前可能产生的捕获溢出信息。

注： 设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

14.3.6 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

- 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选择 TI1)。
- 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿 有效)。
- 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选择 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
- 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

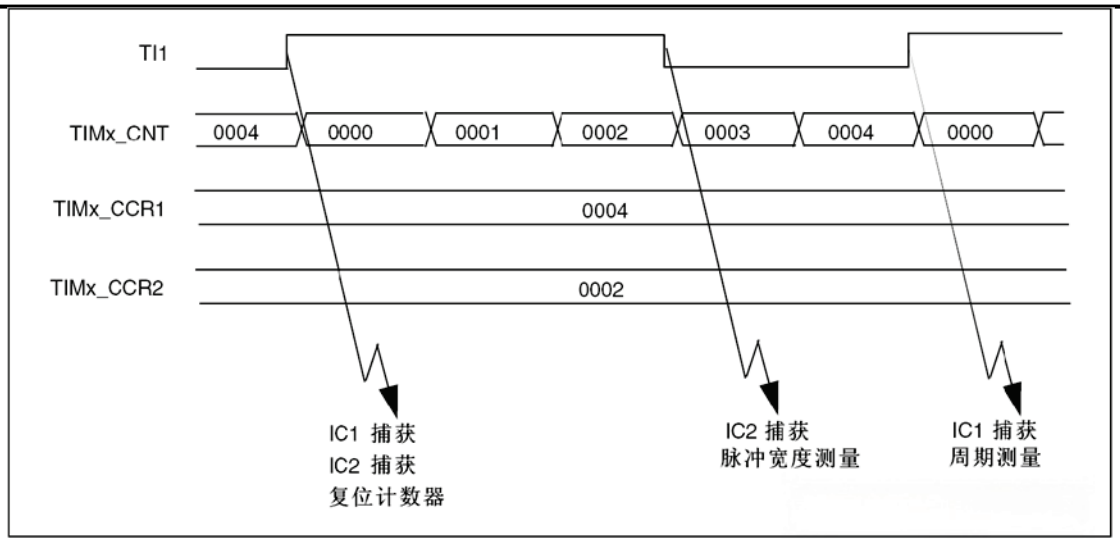


图 124 PWM 输入模式时序

由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器 ，所以 PWM 输入模式只能使用 TIMx_CH1 /TIMx_CH2 信号。

14.3.7 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx)能够 直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状 态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性位相 反的值。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修 改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

14.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。 当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作:

- 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻 转(OCxM=011)。
- 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位(TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位 选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。 在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中
3. 如果要产生一个中断请求和/或一个 DMA 请求，设置 CCxIE 位和/或 CCxDE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出引脚，CCRx 预装载未 用，开启 OCx 输出且高电平有效，则必须设置 OCxM='011' 、OCxPE='0' 、CCxP='0'和 CCxE='1'。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄 存器(OCxPE='0'，否则 TIMx_CCRx 影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

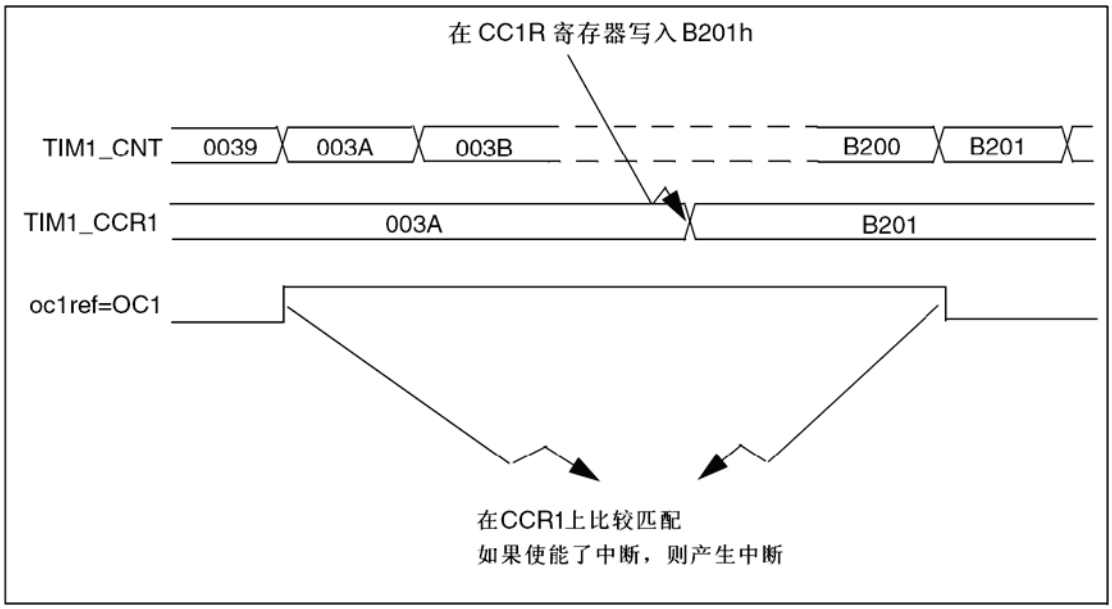


图 125 输出比较模式，翻转 OC1

14.3.9 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空 比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'110'(PWM 模式 1)或'111'(PWM 模式 2)，能够独立地设 置每个 OCx 输出通道产生一路 PWM。必须设置 TIMx_CCMRx 寄存器 OCxPE 位以使能相应的预 装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能 自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计 数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或 低电平有效。TIMx_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx_CCERx 寄存器的 描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向) 以确定是否符合 TIMx_CCRx ≤ TIMx_CNT 或者 TIMx_CNT ≤ TIMx_CCRx 。然而 为了与 OCREF_CLR 的功能(在下一个 PWM 周期之前， ETR 信号上的一个外部事件能够清除 OCxREF) 一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变

- 当输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)从 “冻结” (无比较，OCxM='000')切换到某个 PWM 模式(OCxM='110'或'111')。

这样在运行中可以通过软件强置 PWM 输出。

根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

PWM 边沿对齐模式

向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看 0 节。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时 PWM 信号参考 OCxREF 为高，否 则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为'1'。

如果比较值为 0，则 OCxREF 保持为'0' 。 下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

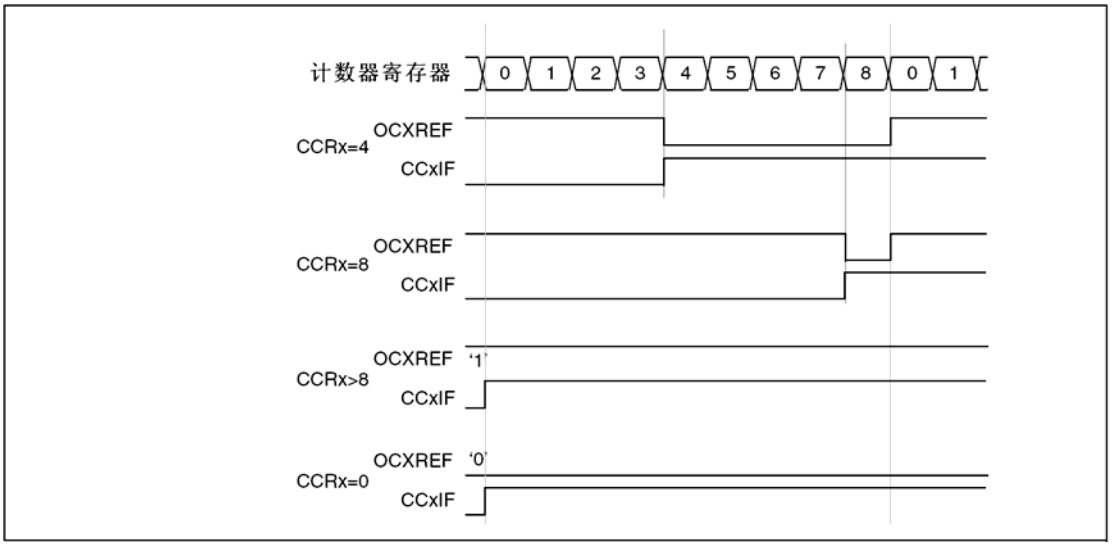


图 126 边沿对齐的 PWM 波形(ARR=8)

向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。参看 0 节。

在 PWM 模式 1 ，当 TIMx_CNT>TIMx_CCRx 时参考信 号 OCxREF 为低，否 则为高。 如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不 能产生 0%的 PWM 波形。

PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为'00'时，为中央对齐模式(所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置'1'、在 计数器向下计数时被置'1'、或在计数器向上和向下计数时被置'1' 。TIMx_CR1 寄存器中的计数方 向位(DIR)由硬件更新，不要用软件修改它。参看 0 节的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器中的 CMS=01，在中央对齐模式 1 时，当计数器向下计数时设置比较标 志。

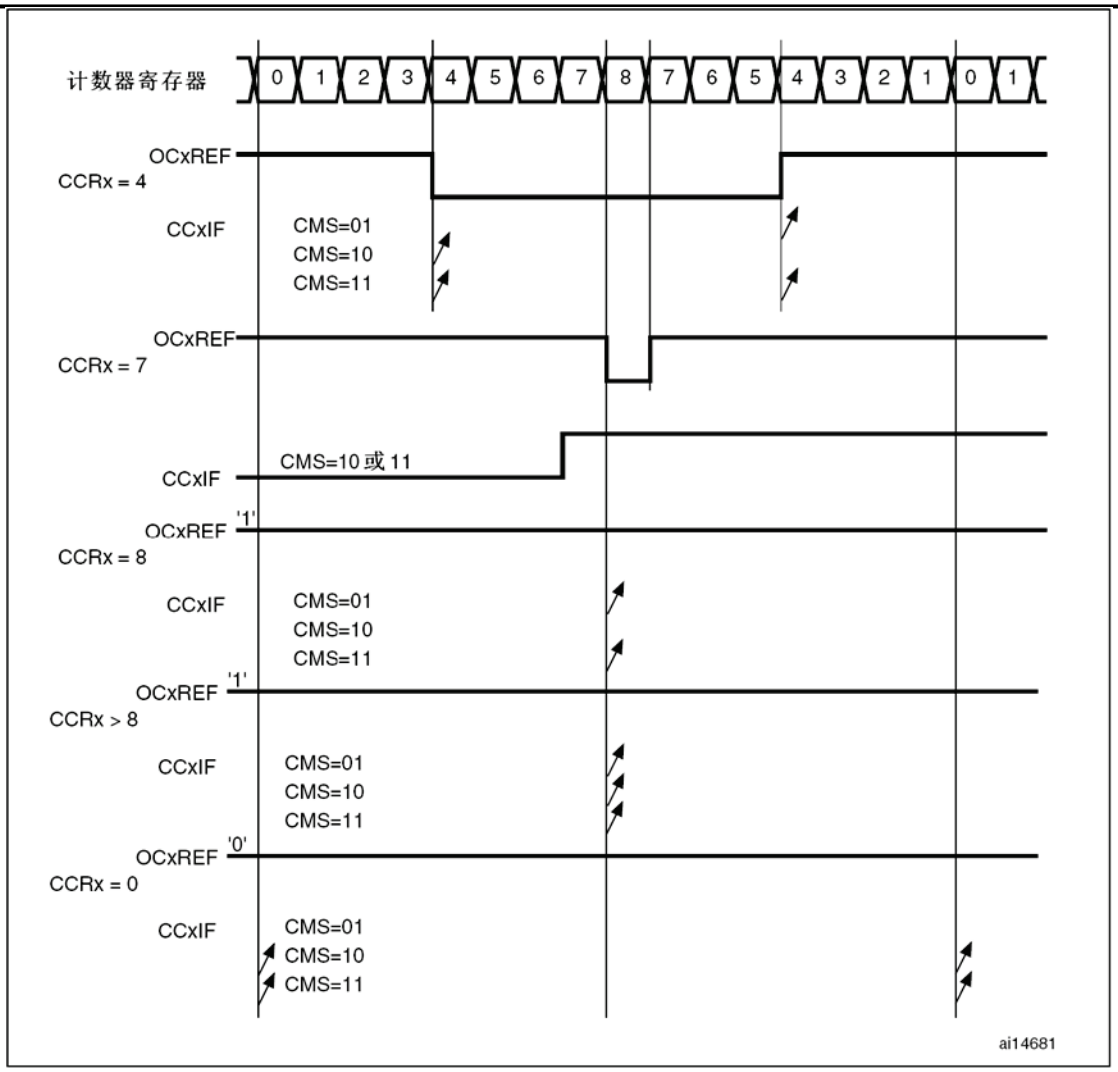


图 127 中央对齐的 PWM 波形(APR=8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx_CNT>TIMx_ARR)，则方向不会被更新。 例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx_EGR 位中的 UG 位)，不要在计数进行过程中修改计数器的值。

14.3.10 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个 程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新 事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

向上计数方式：CNT < CCRx ≤ ARR (特别地，0 < CCRx)，

向下计数方式：CNT > CCRx。

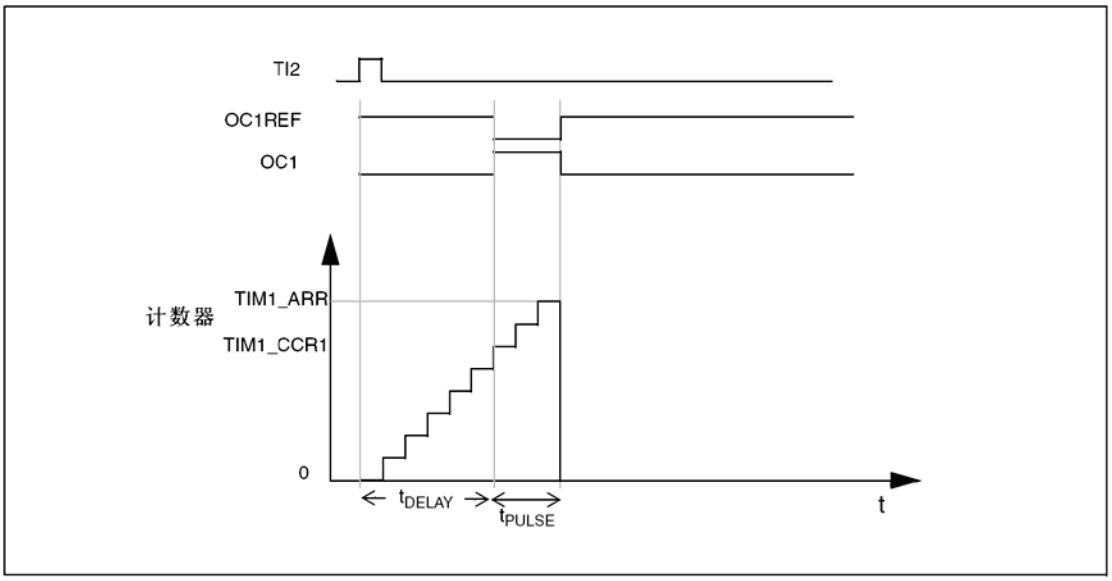


图 128 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长

度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发 1：

- 置 TIMx_CCMR1 寄存器中的 CC2S='01'，把 TI2FP2 映像到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P='0'，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS='110' ，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx_SMCR 寄存器中的 SMS='110'(触发模式)，TI2FP2 被用来启动计数器。 OPM 波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)
- tDELAY 由写入 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1' 到'0'的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M='111'，进入 PWM 模式 2；根据需要有 选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE='1'和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，修改 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中， CC1P='0'。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM='1'，在下一个更新事件(当计数 器从自动装载值翻转到 0)时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的 最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只 在通道配置为 PWM1 和 PWM2 模式时起作用。

14.3.11 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端 的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]='00'。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE='0'。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子 中，定时器 TIMx 被置于 PWM 模式。

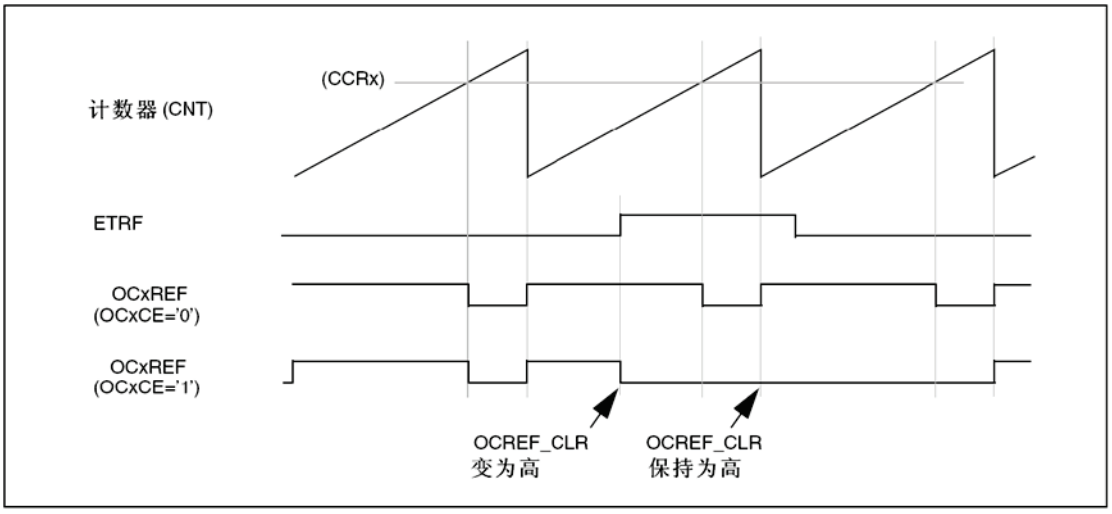


图 129 清除 TIMx 的 OCxREF

14.3.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则 置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 75，假定计数器已经启动(TIMx_CR1 寄存器中的 CEN='1')，计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号； 如果没有滤波和变相， 则 TI1FP1=TI1 ，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号 的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。 不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数。在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、触发输出 特性等仍工作如常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指 示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合， 假设 TI1 和 TI2 不同时变换。

表 75 计数方向与编码器信号的关系

有效边沿	**相对信号的电平 (TI1FP1 对应 TI2, TI2FP2 对应 TI1)**	**TI1FP1 信号**		**TI2FP2 信号**	
		上升	下降	上升	下降
仅在 TI1 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1 和 TI2 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是， 一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信 号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边 沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个 例子中，我们假定配置如下：

- CC1S='01' (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S='01' (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P='0' (TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P='0' (TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS='011' (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN='1' (TIMx_CR1 寄存器，计数器使能)

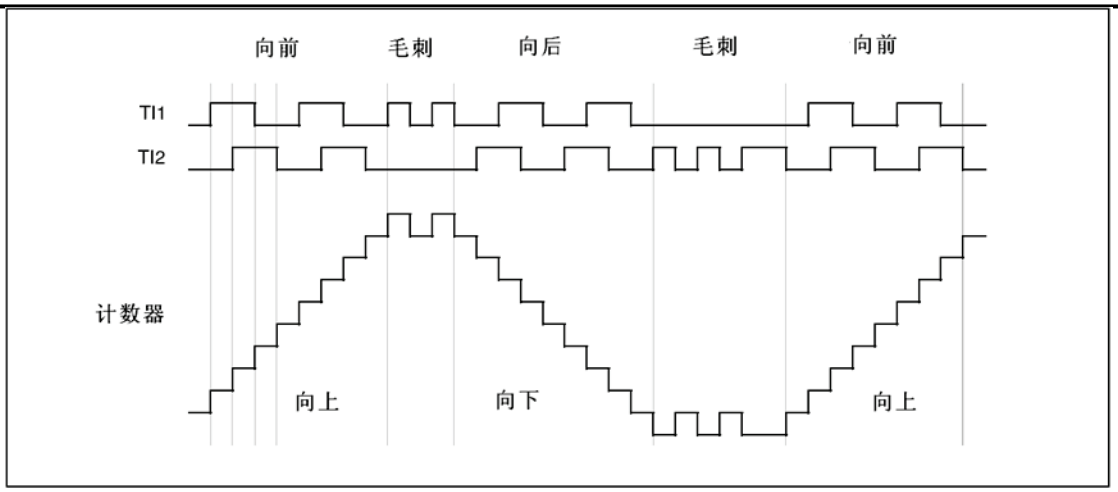


图 130 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1'，其他配置与上例相同)

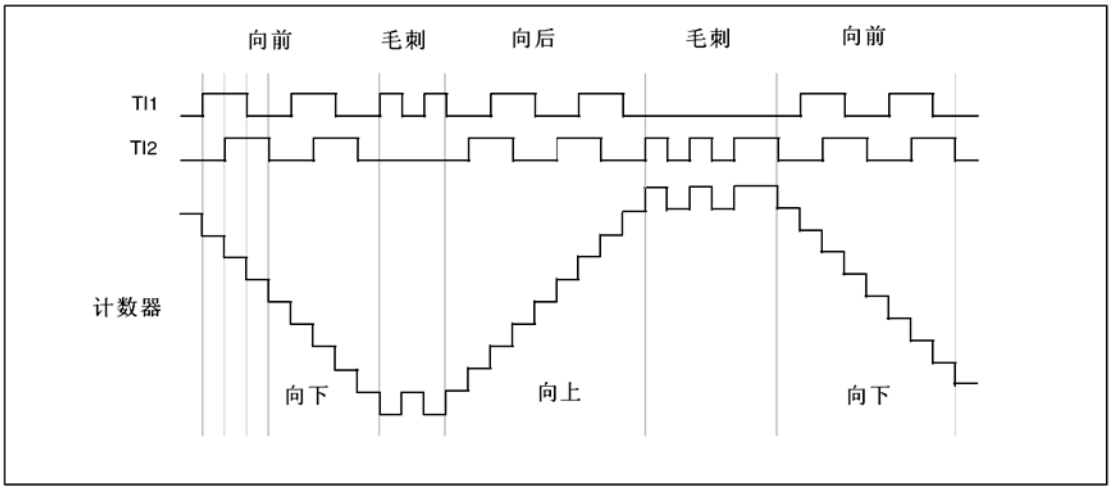


图 131 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式 的定时器，可以测量两个编码器事件的间隔，获得动态的信息(速度，加速度，减速度)。指示机 械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数 器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的 并且可以由另一个定时器产生)；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

14.3.13 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。上一章 13.3.18 节给出了此特性用于连接霍尔传感器的例子。

14.3.14 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR ，TIMx_CCRx)都会被更新。

在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置它。CC1S 位只选 择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以 确定极性(只检测上升沿)。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄 存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之 间的延时，取决于 TI1 输入端的重同步电路。

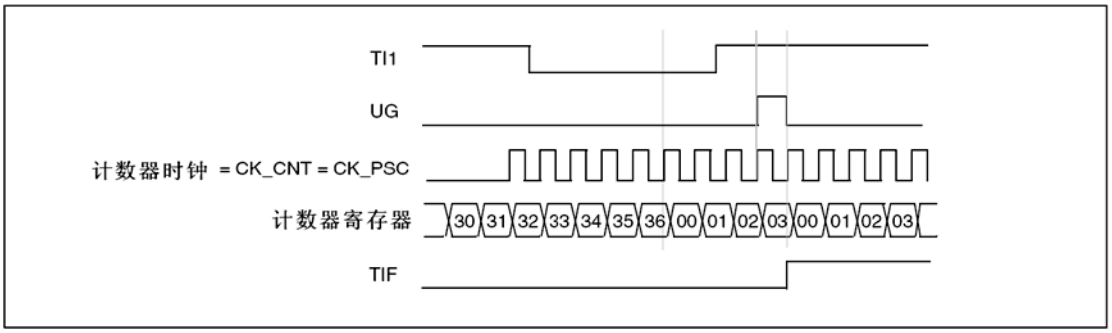


图 132 复位模式下的控制电路

从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入 捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极 性(只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能 启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

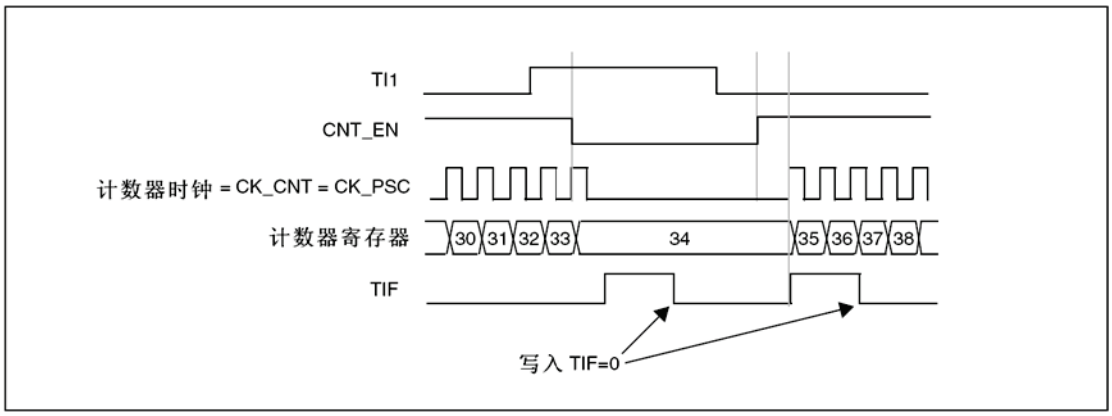


图 133 门控模式下的控制电路

从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕 获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性 (只检测低电平)。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

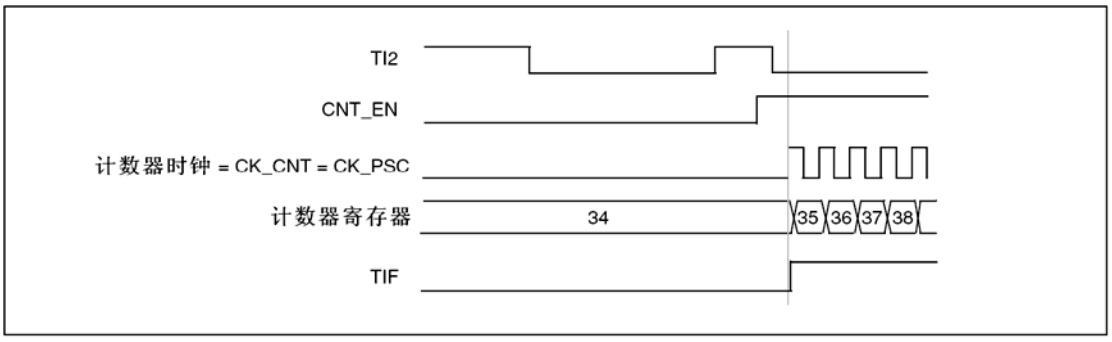


图 134 触发器模式下的控制电路

从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式(外部时钟模式 1 和编码器模式除外)一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发 输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

下面的例子中，TI1 上出现一个上升沿之后，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2
2. 按如下配置通道 1，检测 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 以确定极性(只检测上升沿)

3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

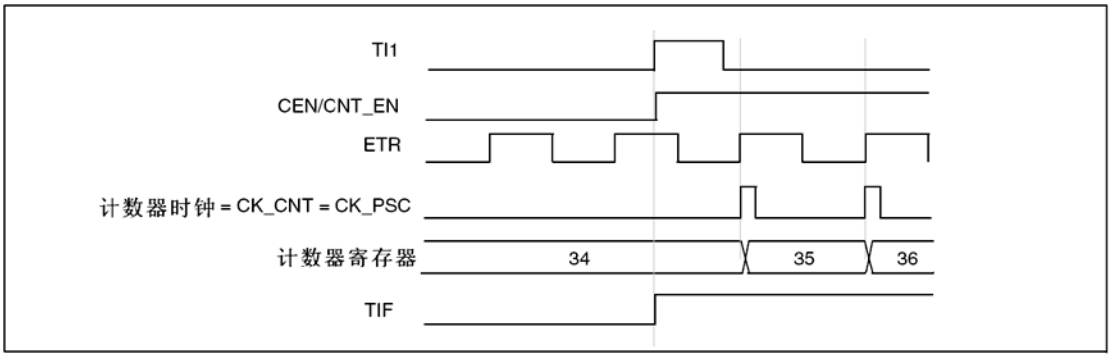


图 135 外部时钟模式 2 + 触发模式下的控制电路

14.3.15 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

使用一个定时器作为另一个定时器的预分频器

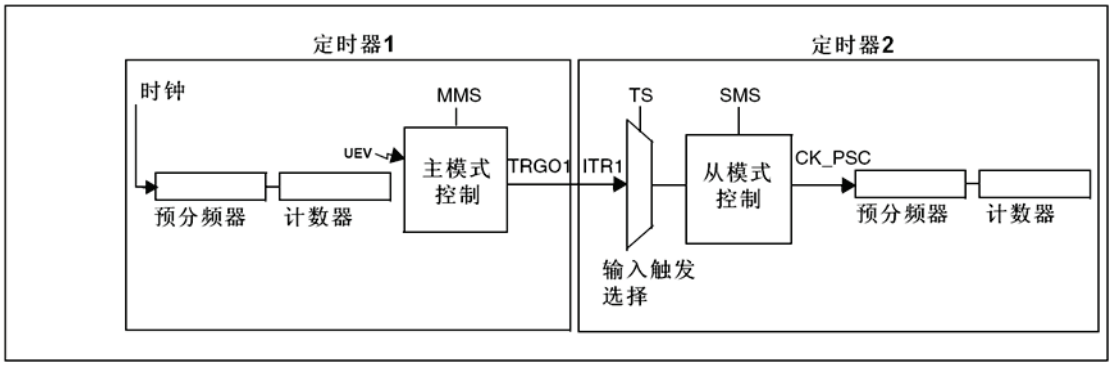


图 136 主/从定时器的例子

如：可以配置定时器 1 作为定时器 2 的预分频器。参考图 136，进行下述操作：

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1_CR2 寄存器的 MMS='010'时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。
- 连接定时器 1 的 TRGO1 输出至定时器 2，设置 TIM2_SMCR 寄存器的 TS='000'，配置定时器 2 为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式 1(TIM2_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期性的上升沿(即定时器 1 的计数器溢出)信号驱动。
- 最后，必须设置相应(TIMx_CR1 寄存器)的 CEN 位分别启动两个定时器。

注：　如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

使用一个定时器使能另一个定时器

在这个例子中，定时器 2 的使能由定时器 1 的输出比较控制。参考图 136 的连接。只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3(fCK_CNT=fCK_INT/3)得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1_CR2 寄存器的 MMS=100)
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM2_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1

注：　定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

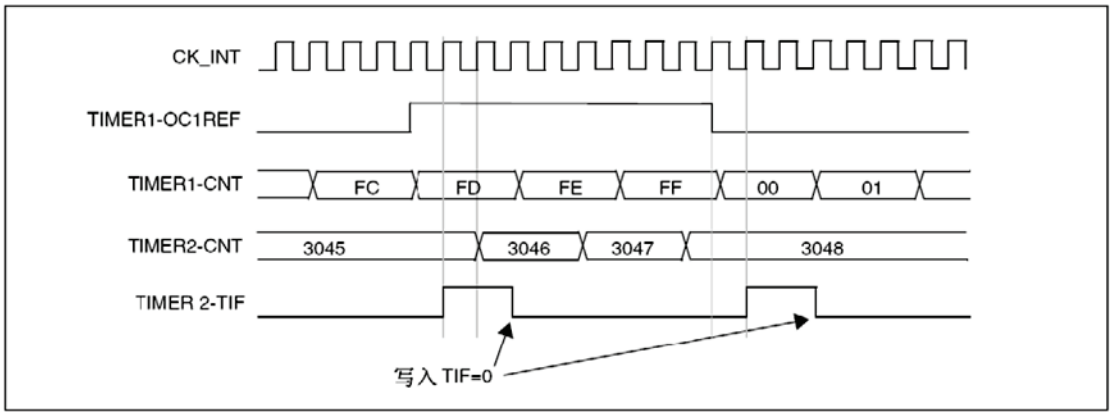


图 137 定时器 1 的 OC1REF 控制定时器 2

在图 137 的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写'0'到 TIM1_CR1 的 CEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1_CR2 寄存器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2_SMCR 寄存器的 SMS=101)
- 置 TIM1_EGR 寄存器的 UG='1'，复位定时器 1。
- 置 TIM2_EGR 寄存器的 UG='1'，复位定时器 2。
- 写'0xE7'至定时器 2 的计数器(TIM2_CNTL)，初始化它为 0xE7。
- 置 TIM2_CR1 寄存器的 CEN='1'以使能定时器 2。
- 置 TIM1_CR1 寄存器的 CEN='1'以启动定时器 1。
- 置 TIM1_CR1 寄存器的 CEN='0'以停止定时器 1。

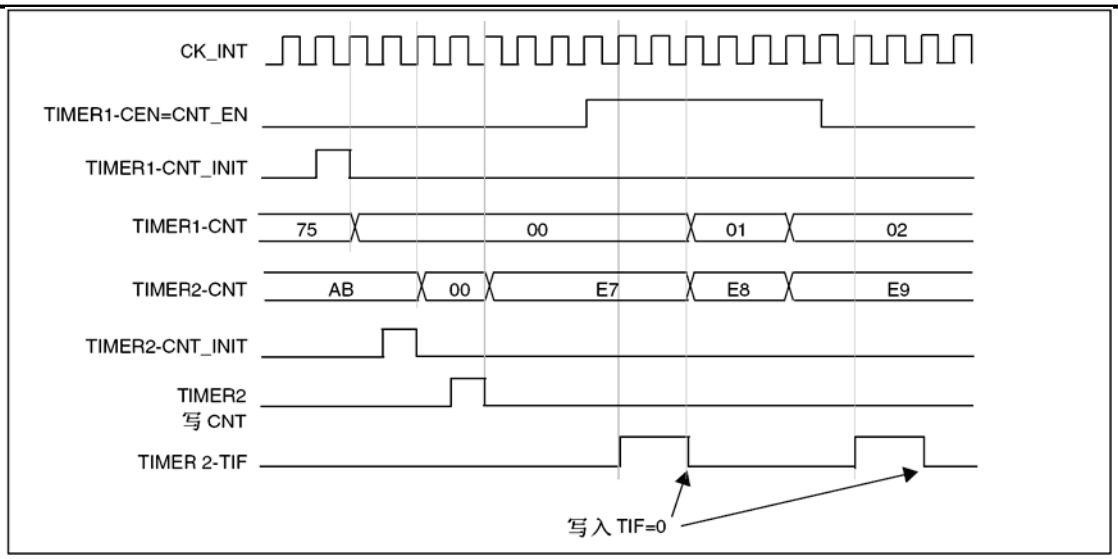


图 138 通过使能定时器 1 可以控制定时器 2

使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 2。参考图 136 的连接。一旦定时器 1 产生更新事件，定时器 2 即从它当前的数值(可以是非 0)按照分频的内部时钟开始计数。在收到触发信号时，定时器 2 的 CEN 位被自动地置'1'，同时计数器开始计数直到写'0'到 TIM2_CR1 寄存器的 CEN 位。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3(fCK_CNT=fCK_INT/3)。

- 配置定时器 1 为主模式，送出它的更新事件(UEV)做为触发输出(TIM1_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS=110)
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

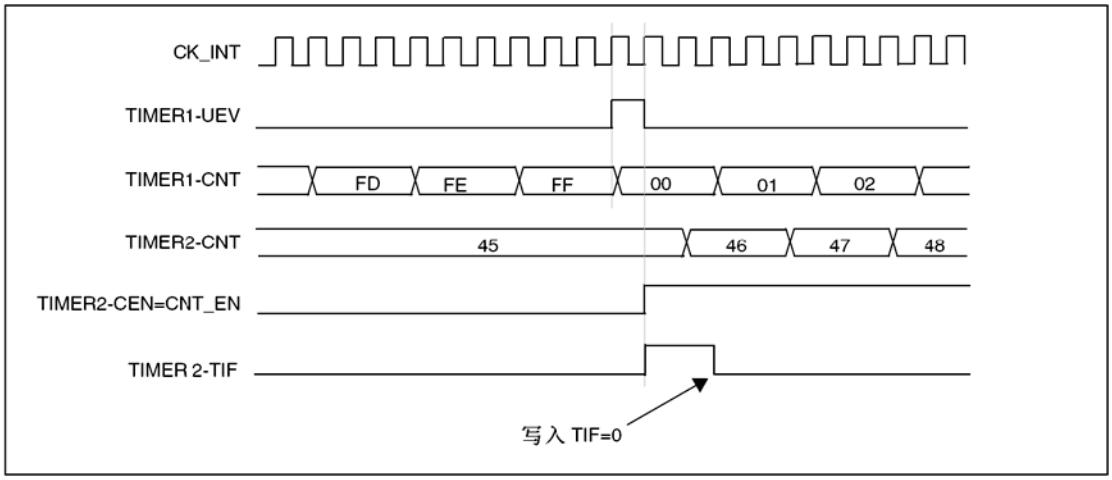


图 139 使用定时器 1 的更新触发定时器 2

在上一个例子中，可以在启动计数之前初始化两个计数器。图 140 显示在与 0 相同配置情况下，使用触发模式而不是门控模式(TIM2_SMCR 寄存器的 SMS=110)的动作。

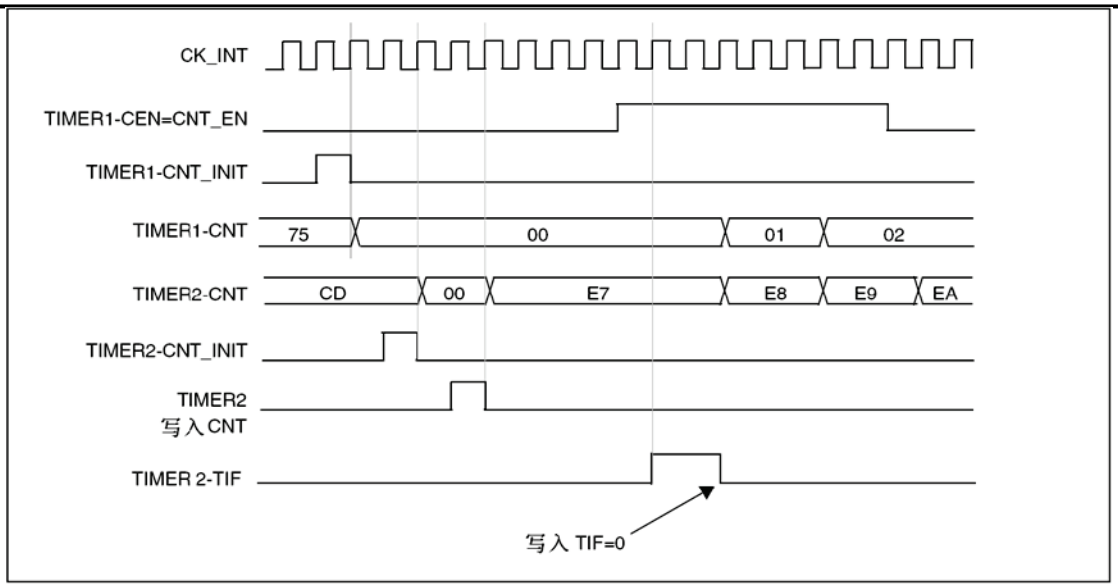


图 140 利用定时器 1 的使能触发定时器 2

使用一个定时器作为另一个的预分频器

这个例子使用定时器 1 作为定时器 2 的预分频器。参考图 136 的连接，配置如下：

- 配置定时器 1 为主模式，送出它的更新事件 UEV 做为触发输出(TIM1_CR2 寄存器的 MMS='010')。然后每次计数器溢出时输出一个周期信号。
- 配置定时器 1 的周期(TIM1_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 使用外部时钟模式(TIM2_SMCR 寄存器的 SMS=111)
- 置 TIM1_CR2 寄存器的 CEN=1 以启动定时器 2。
- 置 TIM1_CR1 寄存器的 CEN=1 以启动定时器 1。

使用一个外部触发同步地启动 2 个定时器

这个例子中当定时器 1 的 TI1 输入上升时使能定时器 1，使能定时器 1 的同时使能定时器 2，参见图 136。为保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1_CR2 寄存器的 MMS='001')。
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1_SMCR 寄存器的 TS='100')。
- 配置定时器 1 为触发模式(TIM1_SMCR 寄存器的 SMS='110')。
- 配置定时器 1 为主/从模式，TIM1_SMCR 寄存器的 MSM='1'。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2_SMCR 寄存器的 SMS='110')。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT_EN 和 CK_PSC 之间有个延迟。

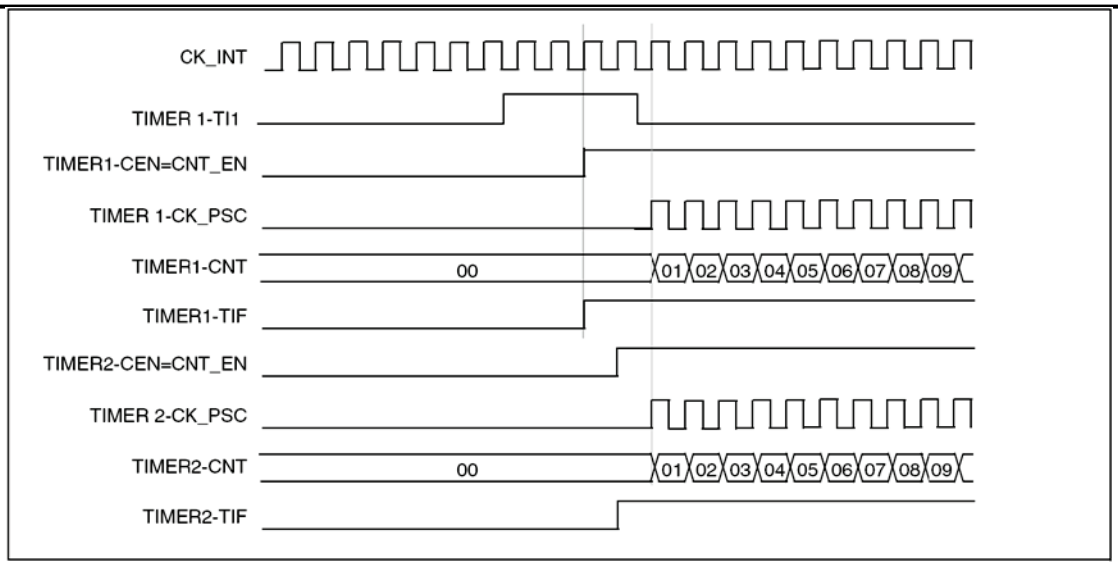


图 141 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

14.3.16 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见随后第 28.16.2 节：支持定时器、看门狗、bxCAN 和 I2C 的调试。

14.4 TIM2 到 TIM5 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 节。
可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

14.4.1 TIM2 到 TIM5 控制寄存器 1(TIMx_CR1)

偏移地址：0x00
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9:8	CKD[1:0]	**CKD[1:0]**:时钟分频因子(Clock division) 定义在定时器时钟(CK_INT)频率与数字滤波器(ETR，TIx)使用的采样频率之间的分频比例。 00：tDTS=tCK_INT 01：tDTS=2xtCK_INT 10：tDTS=4xtCK_INT 11：保留
7	ARPE	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:5	CMS[1:0]	**CMS[1:0]**：选择中央对齐模式(Center-aligned mode selection) 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。

		01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向下计数时被设置。 10：中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，只在计数器向上计数时被设置。 11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位，在计数器向上和向下计数时均被设置。 注：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	**DIR**：方向(Direction) 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
3	OPM	**OPM**：单脉冲模式(Onepulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	URS	**URS**：更新请求源(Update request source)软件通过该位选择 UEV 事件的源 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
1	UDIS	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	**CEN**：使能计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。在单脉冲模式下，当发生更新事件时，CEN 被自动清除。

14.4.2 TIM2 到 TIM5 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TI1S	MMS[2:0]			CCDS	保留		
								rw	rw	rw	rw	rw			

位	符号	说明
15:8	Reserved	保留，始终读为 0。
7	TI1S	**TI1S**：TI1 选择(TI1 selection) 0：TIMx_CH1 引脚连到 TI1 输入； 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。见上一章 13.3.18 的与霍尔传感器的接口一节。
6:4	MMS[2:0]	**MMS[2:0]**：主模式选择(Master mode selection)

		这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下： 000：复位–TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的复位(从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001：使能–计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。 当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010：更新–更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 011：比较脉冲–在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 100：比较–OC1REF 信号被用于作为触发输出(TRGO)。101：比较–OC2REF 信号被用于作为触发输出(TRGO)。110：比较–OC3REF 信号被用于作为触发输出(TRGO)。 111：比较–OC4REF 信号被用于作为触发输出(TRGO)。
3	**CCDS**	**CCDS**：捕获/比较的 DMA 选择 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1：当发生更新事件时，送出 CCx 的 DMA 请求。
2:0	**Reserved**	保留，始终读为 0。

14.4.3 TIM2 到 TIM5 从模式控制寄存器(TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位	符号	说明
15	**ETP**	**ETP**：外部触发极性(External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0：ETR 不反相，高电平或上升沿有效； 1：ETR 被反相，低电平或下降沿有效。
14	**ECE**	**ECE**：外部时钟使能位(External clock enable) 该位启用外部时钟模式 2 0：禁止外部时钟模式 2； 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 注 1：设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF(SMS=111 和 TS=111)具有相同功效。 注 2：下述从模式可以与外部时钟模式 2 同时使用：复位模式、门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF(TS 位不能是'111')。 注 3：外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。
13:12	**ETPS[1:0]**	**ETPS[1:0]**：外部触发预分频(External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 CK_INT 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00：关闭预分频； 01：ETRP 频率除以 2； 10：ETRP 频率除以 4； 11：ETRP 频率除以 8。
11:8	**ETF[3:0]**	**ETF[3:0]**：外部触发滤波(External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。

<table>
<tr><td></td><td></td><td>0000：无滤波器，以 fDTS 采样
0001：采样频率 fSAMPLING=fCK_INT，N=2
0010：采样频率 fSAMPLING=fCK_INT，N=4
0011：采样频率 fSAMPLING=fCK_INT，N=8
0100：采样频率 fSAMPLING=fDTS/2，N=6
0101：采样频率 fSAMPLING=fDTS/2，N=8
0110：采样频率 fSAMPLING=fDTS/4，N=6
0111：采样频率 fSAMPLING=fDTS/4，N=8</td><td>1000：采样频率 fSAMPLING=fDTS/8，N=6
1001：采样频率 fSAMPLING=fDTS/8，N=8
1010：采样频率 fSAMPLING=fDTS/16，N=5
1011：采样频率 fSAMPLING=fDTS/16，N=6
1100：采样频率 fSAMPLING=fDTS/16，N=8
1101：采样频率 fSAMPLING=fDTS/32，N=5
1110：采样频率 fSAMPLING=fDTS/32，N=6
1111：采样频率 fSAMPLING=fDTS/32，N=8</td></tr>
<tr><td>7</td><td>MSM</td><td colspan="2">MSM：主/从模式(Master/slave mode)
0：无作用；
1：触发输入(TRGI)上的事件被延迟了，以允许在当前定时器(通过 TRGO)与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。</td></tr>
<tr><td rowspan="2">6:4</td><td rowspan="2">TS[2:0]</td><td>TS[2:0]：触发选择(Trigger selection)
这 3 位选择用于同步计数器的触发输入。
000：内部触发 0(ITR0)，TIM1
001：内部触发 1(ITR1)，TIM2
010：内部触发 2(ITR2)，TIM3
011：内部触发 3(ITR3)，TIM4</td><td>

100：TI1 的边沿检测器(TI1F_ED)
101：滤波后的定时器输入 1(TI1FP1)
110：滤波后的定时器输入 2(TI2FP2)
111：外部触发输入(ETRF)</td></tr>
<tr><td colspan="2">关于每个定时器中 ITRx 的细节，参见表 76。
注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。</td></tr>
<tr><td>3</td><td>Reserved</td><td colspan="2">保留，始终读为 0。</td></tr>
<tr><td>2:0</td><td>SMS[2:0]</td><td colspan="2">SMS[2:0]：从模式选择(Slave mode selection)
当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明)
000：关闭从模式–如果 CEN=1，则预分频器直接由内部时钟驱动。
001：编码器模式 1–根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。
010：编码器模式 2–根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。
011：编码器模式 3–根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。
100：复位模式–选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。
101：门控模式–当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。
110：触发模式–计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。
111：外部时钟模式 1–选中的触发输入(TRGI)的上升沿驱动计数器。
注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。</td></tr>
</table>

表 76 TIMx 内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

14.4.4 TIM2 到 TIM5DMA/中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位	符号	说明
15	Reserved	保留，始终读为 0。
14	TDE	**TDE**：允许触发 DMA 请求(Trigger DMA request enable) 0：禁止触发 DMA 请求； 1：允许触发 DMA 请求。
13	Reserved	保留，始终读为 0。
12	CC4DE	**CC4DE**：允许捕获/比较 4 的 DMA 请求(Capture/Compare4 DMA request enable) 0：禁止捕获/比较 4 的 DMA 请求； 1：允许捕获/比较 4 的 DMA 请求。
11	CC3DE	**CC3DE**：允许捕获/比较 3 的 DMA 请求(Capture/Compare3 DMA request enable) 0：禁止捕获/比较 3 的 DMA 请求； 1：允许捕获/比较 3 的 DMA 请求。
10	CC2DE	**CC2DE**：允许捕获/比较 2 的 DMA 请求(Capture/Compare2 DMA request enable) 0：禁止捕获/比较 2 的 DMA 请求； 1：允许捕获/比较 2 的 DMA 请求。
9	CC1DE	**CC1DE**：允许捕获/比较 1 的 DMA 请求(Capture/Compare1 DMA request enable) 0：禁止捕获/比较 1 的 DMA 请求； 1：允许捕获/比较 1 的 DMA 请求。
8	UDE	**UDE**：允许更新的 DMA 请求(Update DMA request enable) 0：禁止更新的 DMA 请求； 1：允许更新的 DMA 请求。
7	Reserved	保留，始终读为 0。
6	TIE	**TIE**：触发中断使能(Trigger interrupt enable) 0：禁止触发中断； 1：使能触发中断。
5	Reserved	保留，始终读为 0。
4	CC4IE	**CC4IE**：允许捕获/比较 4 中断(Capture/Compare4 interrupt enable) 0：禁止捕获/比较 4 中断； 1：允许捕获/比较 4 中断。
3	CC3IE	**CC3IE**：允许捕获/比较 3 中断(Capture/Compare3 interrupt enable) 0：禁止捕获/比较 3 中断； 1：允许捕获/比较 3 中断。
2	CC2IE	**CC2IE**：允许捕获/比较 2 中断(Capture/Compare2 interrupt enable) 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
1	CC1IE	**CC1IE**：允许捕获/比较 1 中断(Capture/Compare1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
0	UIE	**UIE**：允许更新中断(Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

14.4.5 TIM2 到 TIM5 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留			CC4OF	CC3OF	CC2OF	CC1OF	保留		TIF	保留	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

位	符号	说明
15:13	Reserved	保留，始终读为 0。
12	CC4OF	**CC4OF**：捕获/比较 4 重复捕获标记(Capture/Compare4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	**CC3OF**：捕获/比较 3 重复捕获标记(Capture/Compare3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	**CC2OF**：捕获/比较 2 重复捕获标记(Capture/Compare2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	**CC1OF**：捕获/比较 1 重复捕获标记(Capture/Compare1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置'1'。写'0'可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。
8:7	Reserved	保留，始终读为 0。
6	TIF	**TIF**：触发器中断标记(Triggerinterruptflag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发器中断等待响应。
5	Reserved	保留，始终读为 0。
4	CC4IF	**CC4IF**：捕获/比较 4 中断标记(Capture/Compare4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	**CC3IF**：捕获/比较 3 中断标记(Capture/Compare3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	**CC2IF**：捕获/比较 2 中断标记(Capture/Compare2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	**CC1IF**：捕获/比较 1 中断标记(Capture/Compare1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置'1'，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	**UIF**：更新中断标记(Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0：无更新事件产生； 1：更新中断等待响应。 当寄存器被更新时该位由硬件置'1'： -若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新事件(软件对计数器 CNT 重新初始化)；

		-若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重初始化时产生更新事件。(参考同步控制寄存器的说明)

14.4.6 TIM2 到 TIM5 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TG	保留	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位	符号	说明
15:7	Reserved	保留，始终读为 0。
6	TG	**TG**：产生触发事件(Trigger generation) 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
5	Reserved	保留，始终读为 0。
4	CC4G	**CC4G**：产生捕获/比较 4 事件(Capture/compare4 generation) 参考 CC1G 描述。
3	CC3G	**CC3G**：产生捕获/比较 3 事件(Capture/compare3 generation) 参考 CC1G 描述。
2	CC2G	**CC2G**：产生捕获/比较 2 事件(Capture/compare2 generation) 参考 CC1G 描述。
1	CC1G	**CC1G**：产生捕获/比较 1 事件(Capture/compare1generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 若通道 CC1 配置为输入： 当前的计数器值捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	**UG**：产生更新事件(Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'，若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

14.4.7 TIM2 到 TIM5 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			

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

输出比较模式：

位	符号	说明
15	OC2CE	**OC2CE**：输出比较 2 清 0 使能(Output compare2 cleare nable)
14:12	OC2M[2:0]	**OC2M[2:0]**：输出比较 2 模式(Output compare2 mode)
11	OC2PE	**OC2PE**：输出比较 2 预装载使能(Output compare2 preload enable)
10	OC2FE	**OC2FE**：输出比较 2 快速使能(Output compare2 fast enable)
9:8	CC2S[1:0]	**CC2S[1:0]**：捕获/比较 2 选择(Capture/Compare2selection) 该位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
7	OC1CE	**OC1CE**：输出比较 1 清 0 使能(Output compare1 clear enable) 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
6:4	OC1M[2:0]	**OC1M[2:0]**：输出比较 1 模式(Output compare1 enable) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S='00'(该通道配置成输出)则该位不能被修改。 注 2：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
3	OC1PE	**OC1PE**：输出比较 1 预装载使能(Output compare1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被传送至当前寄存器中。 注 1：一旦 LOCK 级别设为 3(TIMx_BDTR 寄存器中的 LOCK 位)并且 CC1S='00'(该通道配置成输出)则该位不能被修改。 注 2：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM='1')，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	OC1FE	**OC1FE**：输出比较 1 快速使能(Output compare1 fast enable)

		该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1:0	CC1S[1:0]	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

输入捕获模式：

位	符号	说明
15:12	IC2F[3:0]	**IC2F[3:0]**：输入捕获 2 滤波器(Input capture2 filter)
11:10	IC2PSC[1:0]	**IC2PSC[1:0]**：输入/捕获 2 预分频器(input capture2 prescaler)
9:8	CC2S[1:0]	**CC2S[1:0]**：捕获/比较 2 选择(Capture/compare2 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
7:4	IC1F[3:0]	**IC1F[3:0]**：输入捕获 1 滤波器(Input capture1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=21001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=41010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=81011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=61100：采样频率 fSAMPLING=fDTS/16，N=8 0101：采样频率 fSAMPLING=fDTS/2，N=81101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=61110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=81111：采样频率 fSAMPLING=fDTS/32，N=8 注：当 ICxF[3:0]=1、2 或 3 时，公式中的 fDTS 由 CK_INT 替代。
3:2	IC1PSC[1:0]	**IC1PSC[1:0]**：输入/捕获 1 预分频器(Input capture1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。 一旦 CC1E='0'(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
1:0	CC1S[1:0]	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上；

		11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

14.4.8 TIM2 到 TIM5 捕获/比较模式寄存器 2(TIMx_CCMR2)

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15	OC4CE	**OC4CE**：输出比较 4 清 0 使能(Output compare4 clear enable)
14:12	OC4M[2:0]	**OC4M[2:0]**：输出比较 4 模式(Output compare4 mode)
11	OC4PE	**OC4PE**：输出比较 4 预装载使能(Output compare4 preload enable)
10	OC4FE	**OC4FE**：输出比较 4 快速使能(Output compare4 fast enable)
9:8	CC4S[1:0]	**CC4S[1:0]**：捕获/比较 4 选择(Capture/Compare4 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上；10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E='0')才是可写的。
7	OC3CE	**OC3CE**：输出比较 3 清 0 使能(Output compare3 clear enable)
6:4	OC3M[2:0]	**OC3M[2:0]**：输出比较 3 模式(Output compare3 mode)
3	OC3PE	**OC3PE**：输出比较 3 预装载使能(Output compare3 preload enable)
2	OC3FE	**OC3FE**：输出比较 3 快速使能(Output compare3 fast enable)
1:0	CC3S[1:0]	**CC3S[1:0]**：捕获/比较 3 选择(Capture/Compare3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRGI 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E='0')才是可写的。

输入捕获模式：

位	符号	说明
15:12	IC4F[3:0]	**IC4F[3:0]**：输入捕获 4 滤波器(Input capture4 filter)
11:10	IC4PSC[1:0]	**IC4PSC[1:0]**：输入/捕获 4 预分频器(input capture4 prescaler)
9:8	CC4S[1:0]	**CC4S[1:0]**：捕获/比较 4 选择(Capture/compare4selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC4 通道被配置为输出； 01：CC4 通道被配置为输入，IC4 映射在 TI4 上； 10：CC4 通道被配置为输入，IC4 映射在 TI3 上； 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。

		注：CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E='0')才是可写的。
7:4	IC3F[3:0]	**IC3F[3:0]**：输入捕获 3 滤波器(Input capture3 filter)
3:2	IC3PSC[1:0]	**IC3PSC[1:0]**：输入/捕获 3 预分频器(Input capture3 prescaler)
1:0	CC3S[1:0]	**CC3S[1:0]**：捕获/比较 3 选择(Capture/Compare3 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC3 通道被配置为输出； 01：CC3 通道被配置为输入，IC3 映射在 TI3 上； 10：CC3 通道被配置为输入，IC3 映射在 TI4 上； 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E='0')才是可写的。

14.4.9 TIM2 到 TIM5 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		CC4P	CC4E	保留		CC3P	CC3E	保留		CC2P	CC2E	保留		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

位	符号	说明
15:14	Reserved	保留，始终读为 0。
13	CC4P	**CC4P**：输入/捕获 4 输出极性(Capture/Compare4 output polarity) 参考 CC1P 的描述。
12	CC4E	**CC4E**：输入/捕获 4 输出使能(Capture/Compare4 output enable) 参考 CC1E 的描述。
11:10	Reserved	保留，始终读为 0。
9	CC3P	**CC3P**：输入/捕获 3 输出极性(Capture/Compare3 output polarity) 参考 CC1P 的描述。
8	CC3E	**CC3E**：输入/捕获 3 输出使能(Capture/Compare3 output enable) 参考 CC1E 的描述。
7:6	Reserved	保留，始终读为 0。
5	CC2P	**CC2P**：输入/捕获 2 输出极性(Capture/Compare2 output polarity) 参考 CC1P 的描述。
4	CC2E	**CC2E**：输入/捕获 2 输出使能(Capture/Compare2 output enable) 参考 CC1E 的描述。
3:2	Reserved	保留，始终读为 0。
1	CC1P	**CC1P**：输入/捕获 1 输出极性(Capture/Compare1 output polarity) CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
0	CC1E	**CC1E**：输入/捕获 1 输出使能(Capture/Compare1 output enable) CC1 通道配置为输出： 0：关闭 - OC1 禁止输出。 1：开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。

		0：捕获禁止； 0：捕获使能。

表 77 标准 OCx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出(OCx=0，OCx_EN=0)
1	OCx=OCxREF+极性，OCx_EN=1

注：　连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

14.4.10 TIM2 到 TIM5 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

位	符号	说明
15:0	CNT[15:0]	**CNT[15:0]**：计数器的值(Counter value)

14.4.11 TIM2 到 TIM5 预分频器(TIMx_PSC)

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

位	符号	说明
15:0	PSC[15:0]	**PSC[15:0]**：预分频器的值(Prescaler value) 计数器的时钟频率 CK_CNT 等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值。

14.4.12 TIM2 到 TIM5 自动重装载寄存器(TIMx_ARR)

偏移地址：0x2C

复位值：0x0000

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

位	符号	说明
15:0	ARR[15:0]	**ARR[15:0]:**自动重装载的值(Auto reload value) ARR 包含了将要传送至实际的自动重装载寄存器的数值。详细参考 14.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

14.4.13 TIM2 到 TIM5 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															

rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0

位	符号	说明
15:0	CCR1[15:0]	**CCR1[15:0]**:捕获/比较 1 的值(Capture/Compare1 value) 若 CC1 通道配置为输出: CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入: CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

14.4.14 TIM2 到 TIM5 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

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

CCR2[15:0]

rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0

位	符号	说明
15:0	CCR2[15:0]	**CCR2[15:0]**:捕获/比较 2 的值(Capture/Compare2 value) 若 CC2 通道配置为输出: CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 若 CC2 通道配置为输入: CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。

14.4.15 TIM2 到 TIM5 捕获/比较寄存器 3(TIMx_CCR3)

偏移地址：0x3C

复位值：0x0000

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

CCR3[15:0]

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

位	符号	说明
15:0	CCR3[15:0]	**CCR3[15:0]**:捕获/比较 3 的值(Capture/Compare3 value) 若 CC3 通道配置为输出: CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。 如果在 TIMx_CCMR3 寄存器(OC3PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 若 CC3 通道配置为输入: CCR3 包含了由上一次输入捕获 3 事件(IC3)传输的计数器值。

14.4.16 TIM2 到 TIM5 捕获/比较寄存器 4(TIMx_CCR4)

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0

位	符号	说明
15:0	CCR4[15:0]	**CCR4[15:0]**:捕获/比较 4 的值(Capture/Compare4 value) 若 CC4 通道配置为输出: CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。 如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 若 CC4 通道配置为输入: CCR4 包含了由上一次输入捕获 4 事件(IC4)传输的计数器值。

14.4.17 TIM2 到 TIM5DMA 控制寄存器(TIMx_DCR)

偏移地址：0x48

复位值：0x0000

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

位	符号	说明
15:13	Reserved	保留，始终读为 0。
12:8	DBL[4:0]	**DBL[4:0]**:DMA 连续传送长度(DMA burst length) 这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的字节数目: 00000：1 个字节　　00001：2 个字节 00010：3 个字节　　…… ……　　10001：18 个字节
7:5	Reserved	保留，始终读为 0。
4:0	DBA[4:0]	**DBA[4:0]**:DMA 基地址(DMA base address) 这些位定义了 DMA 在连续模式下的基地址(当对 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量: 00000：TIMx_CR1， 00001：TIMx_CR2， 00010：TIMx_SMCR， ……

14.4.18 TIM2 到 TIM5 连续模式的 DMA 地址(TIMx_DMAR)

偏移地址：0x4C

复位值：0x0000

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

位	符号	说明

15:0	DMAB[15:0]	**DMAB[15:0]**:DMA 连续传送寄存器(DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作：TIMx_CR1 地址+DBA+DMA 索引，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

如何使用 DMA 突发特性的示例

在此示例中，定时器 DMA 突发特性用于更新 CCRx 寄存器的内容(x=2，3，4)，DMA 将半个字转移到 CCRx 寄存在中。

这是通过以下步骤完成的:

1.配置相应的 DMA 通道如下:

- DMA 通道外围地址是 DMAR 寄存器地址
- DMA 通道内存地址是 RAM 中的缓冲区地址，该缓冲区包含要由 DMA 传输到 CCRx 寄存器的数据。
- 要传输的数据数量=3(见下注)
- -关闭圆形模式。

2.通过配置 DBA 和 DBL 位字段来配置 DCR 寄存器如下:DBL=3 次传输，DBA=0xE。

3.使能 TIMx update DMA 请求(在 DIER 寄存器中设置 UDE 位)。

4.启用 TIMX

5.启用 DMA 通道

注意：这个例子适用于每个 CCRx 寄存器需要更新一次的情况。如果每个 CCRx 寄存器需要更新两次，那么需要传输的数据数量应该是 6。以 RAM 中包含数据 1、数据 2、数据 3、数据 4、数据 5 和数据 6 的缓冲区为例。将数据传输到 CCRx 寄存器如下:在第一次更新 DMA 请求时，数据 1 被传输到 CCR2，数据 2 被传输到 CCR3，数据 3 被传输到 CCR4;在第二次更新 DMA 请求时，数据 4 被传输到 CCR2，数据 5 被传输到 CCR3，数据 6 被传输到 CCR4。

14.4.19 TIM2 到 TIM5 寄存器

下表中将 TIMx 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 78 TIM2 到 TIM5 寄存器和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	复位值																							0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	保留																								TI1S	MMS[2:0]			CCDS	保留		
	复位值																									0	0	0	0	0			
008h	TIMx_SMCR	保留																ETP	ECE	ETPS[1:0]		EFT[3:0]				MSM	TS[2:0]			保留	SMS[2:0]		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
00Ch	TIMx_DIER	保留																	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	T1E	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	复位值																		0		0	0	0	0	0		0		0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010h	TIMx_SR	保留																			CC4OF	CC3OF	CC2OF	CC1OF	保留		TI1	保留	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	复位值																				0	0	0	0			0		0	0	0	0	0
014h	TIMx_EGR	保留																								保留	TG	保留	CC4G	CC3G	CC2G	CC1G	UG
	复位值																										0		0	0	0	0	0
018h	TIMx_CCMR1输出比较模式	保留																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1输入捕获模式	保留																IC2F [3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F [3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	TIMx_CCMR2输出比较模式	保留																OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2输入捕获模式	保留																IC4F [3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F [3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	TIMx_CCER	保留																						CC4P	CC4E	保留	CC3P	CC3E	保留	CC2P	CC2E	保留	CC1P	CC1E
	复位值																							0	0		0	0		0	0		0	0
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	保留																																
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	保留																CCR3[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	保留																CCR4[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	保留																																
048h	TIMx_DCR	保留																			DBL[4:0]					保留	DBA[4:0]					
	复位值																				0	0	0	0	0		0	0	0	0	0	
04Ch	TIMx_DMAR	保留																DMAB[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器的起始地址，参见表 1。

15 通用定时器(TIM9 到 TIM14)

15.1 TIM9 到 TIM14 简介

通用定时器是一个通过可编程预分频器驱动的 16 位自动装载计数器构成。

它适用于多种场合，包括测量输入信号的脉冲长度(输入捕获)或者产生输出波形(输出比较和 PWM)。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个 毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见 15.3.12 节。

15.2 TIM9 到 TIM14 的主要功能

15.2.1 TIM9/TIM12 的主要功能

通用 TIM9 至 TIM14 定时器功能包括:

- 16 位向上、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(边缘或中间对齐模式)
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断:
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

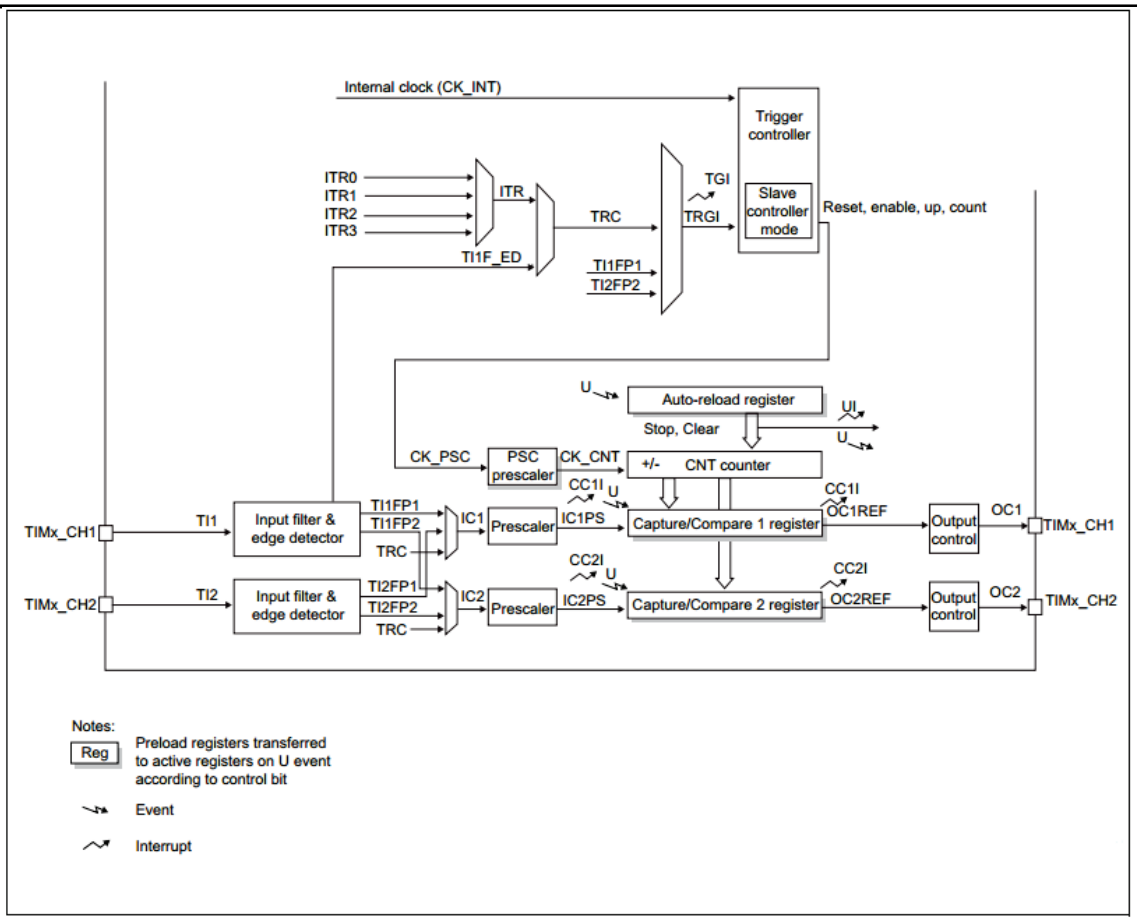


图 142 通用定时器框图（TIM9/TIM12）

15.2.2 TIM10/TIM11 和 TIM13/TIM14 的主要功能

通用 TIM10/TM11 和 TIM13/TIM4 定时器功能包括：

- 16 位向上、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(边缘或中间对齐模式)
  - 单脉冲模式输出
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 输入捕获
  - 输出比较

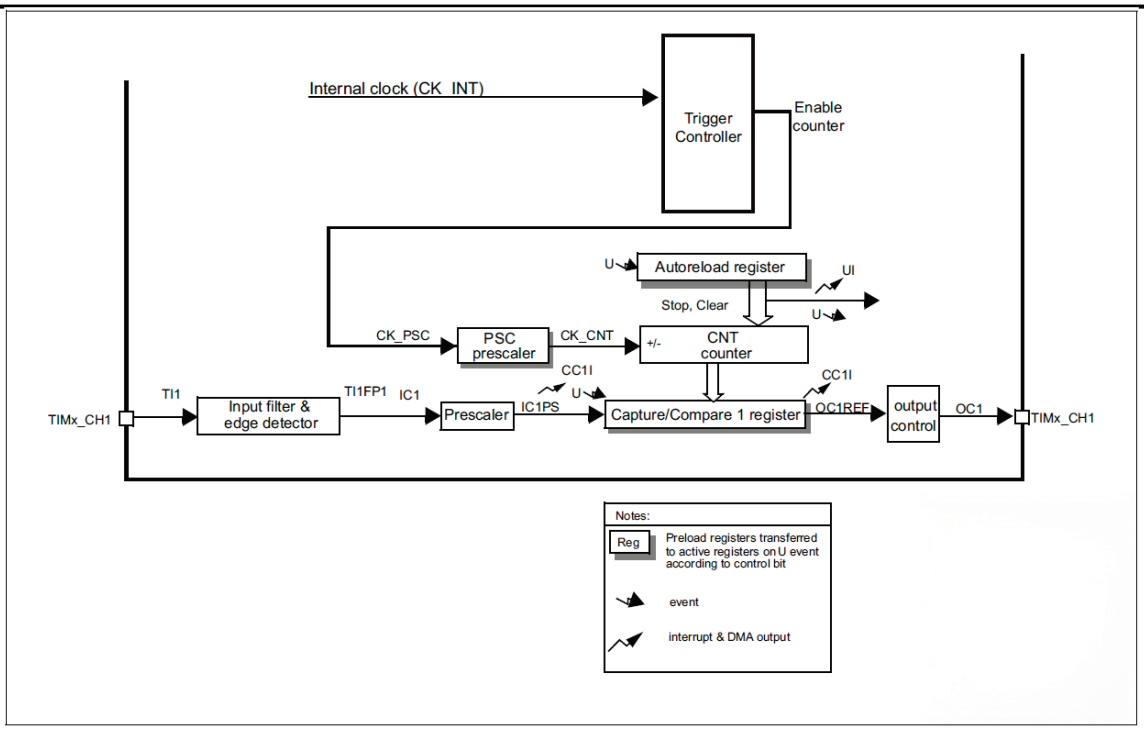


图 143 通用定时器框图（TIM10/11/12/13/14）

15.3 TIM9 到 TIM14 功能描述

15.3.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。

此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。

时基单元包含：

- 计数器寄存器(TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动装载寄存器 (TIMx_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。 根据在 TIMx_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在 每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIMx_CR1 寄存器中的 UDIS 位等于'0'时，产生更新事件。更新事件也可以由软件产生。随后会 详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位(CEN)时，CK_CNT 才有效。(有关计数器使能的细节，请参见控制器的从模式描述)。

注： 真正的计数器使能信号 CNT_EN 是在 CEN 的一个时钟周期后被设置。

预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

图 144 和图 145 给出了在预分频器运行时，更改计数器参数的例子。

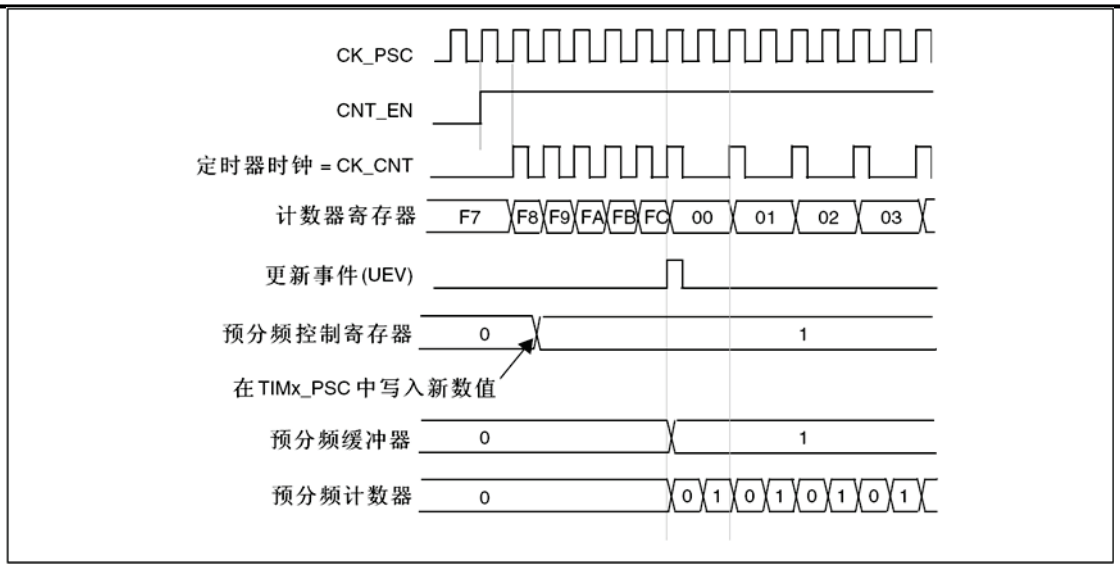


图 144 当预分频器的参数从 1 变到 2 时，计数器的时序图

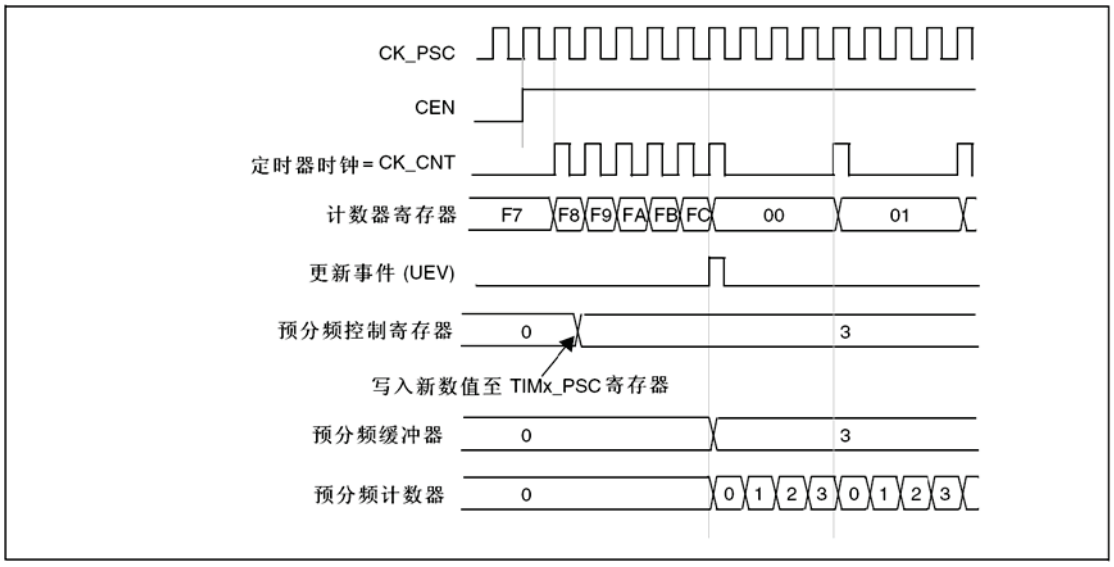


图 145 当预分频器的参数从 1 变到 4 时，计数器的时序图

15.3.2 计数器模式

向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx_ARR 计数器的内容)，然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中(通过软件方式或者使用从模式控 制器)设置 UG 位也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写 入新值时更新影子寄存器。在 UDIS 位被清'0'之前，将不产生更新事件。但是在应该产生更新事 件时，计数器仍会被清'0'，同时预分频器的计数也被请 0(但预分频系数不变)。此外，如果设置 了 TIMx_CR1 寄存器中的 URS 位(选择更新请求)，设置 UG 位将产生一个更新事件 UEV，但硬件 不设置 UIF 标志(即不产生中断请求)；这是为了避免在捕获模式下清除计数器时，同时产 生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx_SR 寄存器中的 UIF 位)。

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx_PSC 寄存器的内容)。
- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

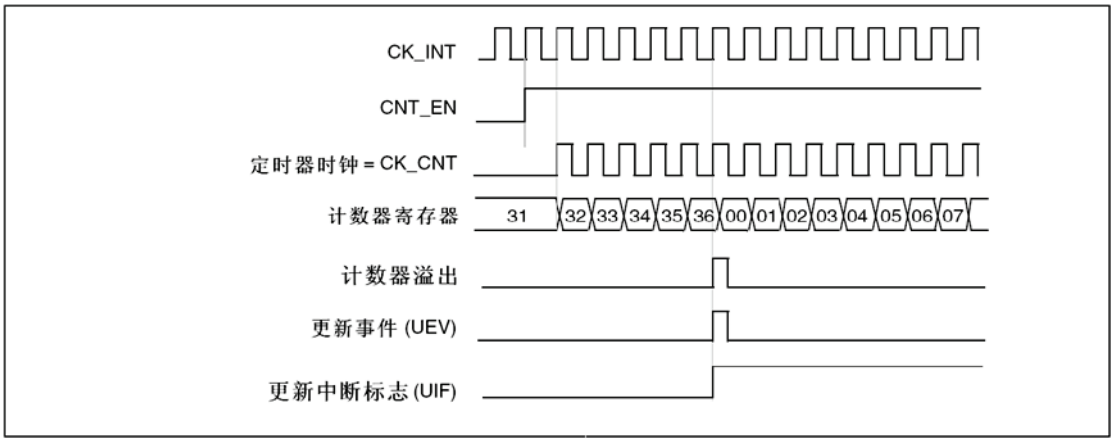


图 146 计数器时序图，内部时钟分频因子为 1

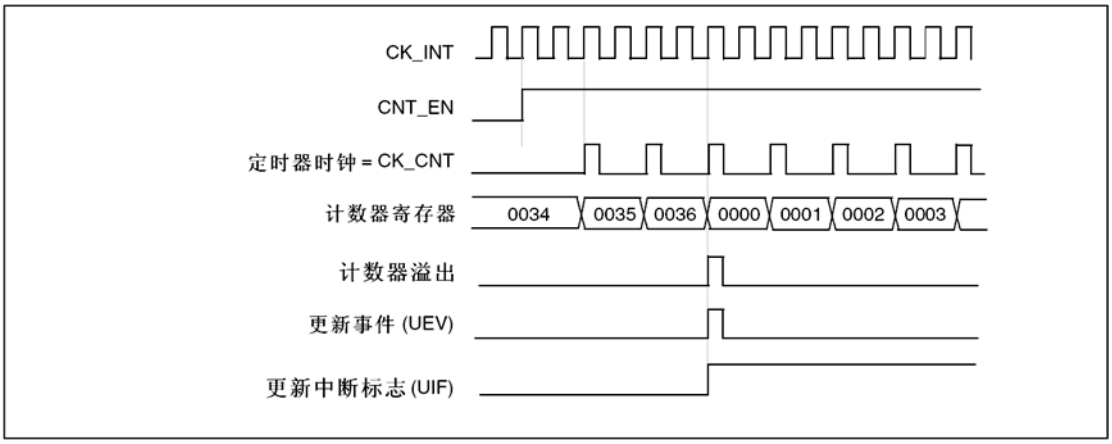


图 147 计数器时序图，内部时钟分频因子为 2

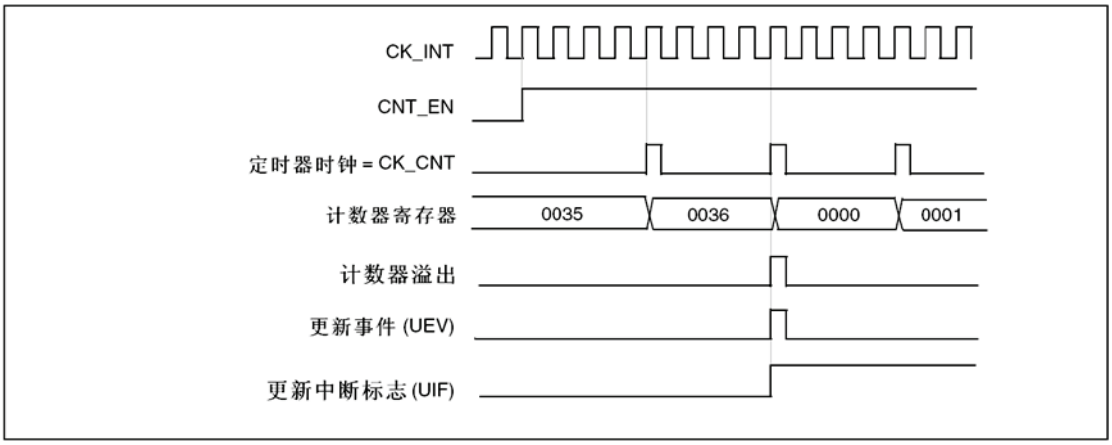


图 148 计数器时序图，内部时钟分频因子为 4

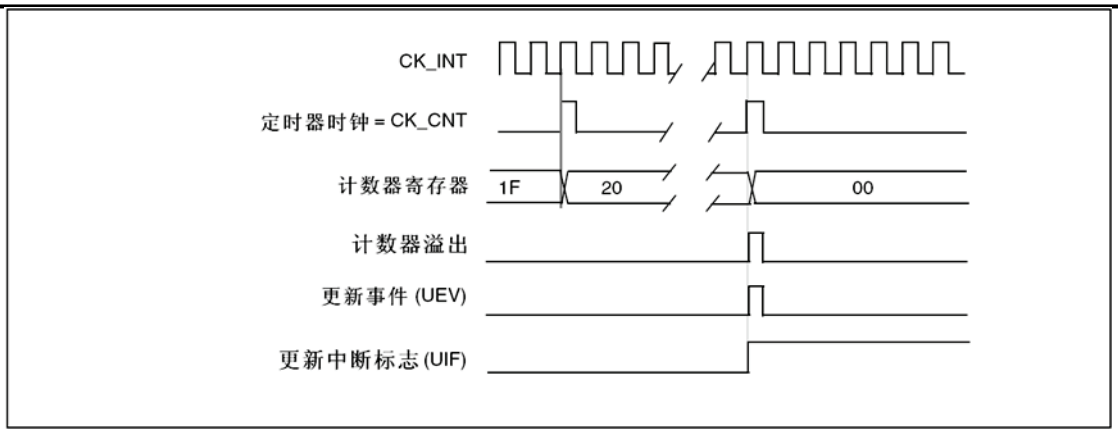


图 149 计数器时序图，内部时钟分频因子为 N

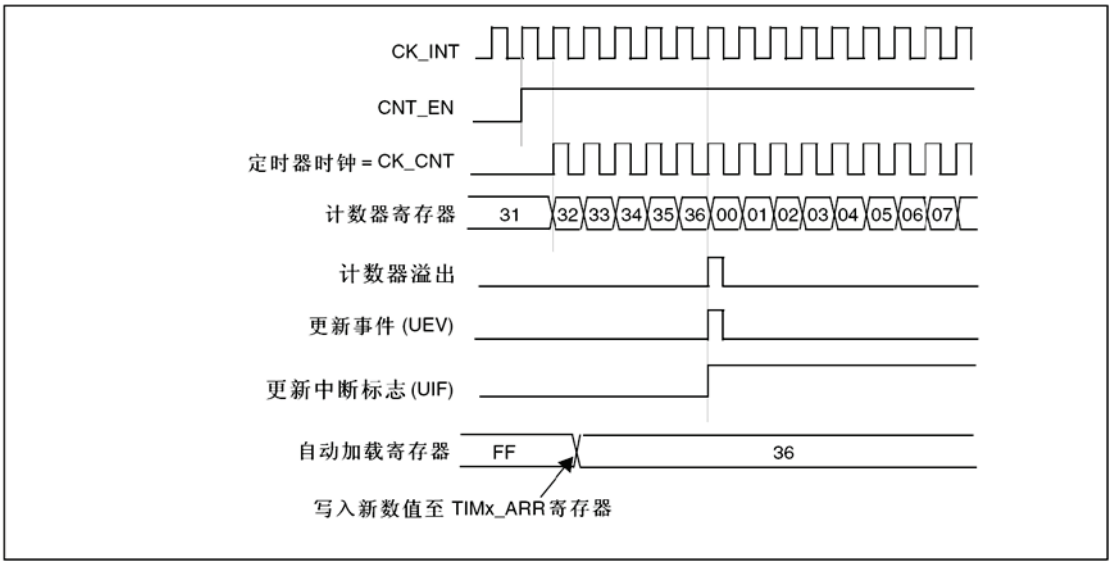


图 150 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装入)

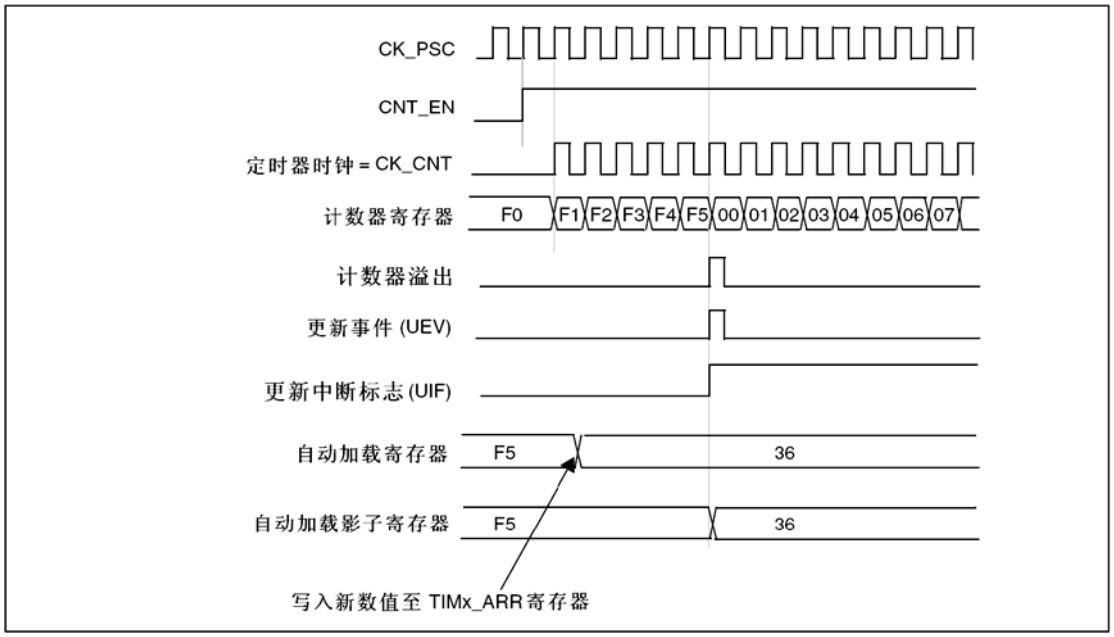


图 151 计数器时序图，当 ARPE=1 时的更新事件(预装入了 TIMx_ARR)

15.3.3 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK_INT)
- 外部时钟模式 1(适用于 TIM9 和 TIM12)：外部输入脚(TIx)
- 内部触发输入(ITRx)(适用于 TIM9 和 TIM12)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时 器 Timer1 而作为另一个定时器 Timer2 的预分频器。参见 13.3.15。

内部时钟源(CK_INT)

如果禁止了从模式控制器(TIMx_SMCR 寄存器的 SMS=000)，则 CEN 、DIR(TIMx_CR1 寄存器) 和 UG 位(TIMx_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只 要 CEN 位被写成'1'，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

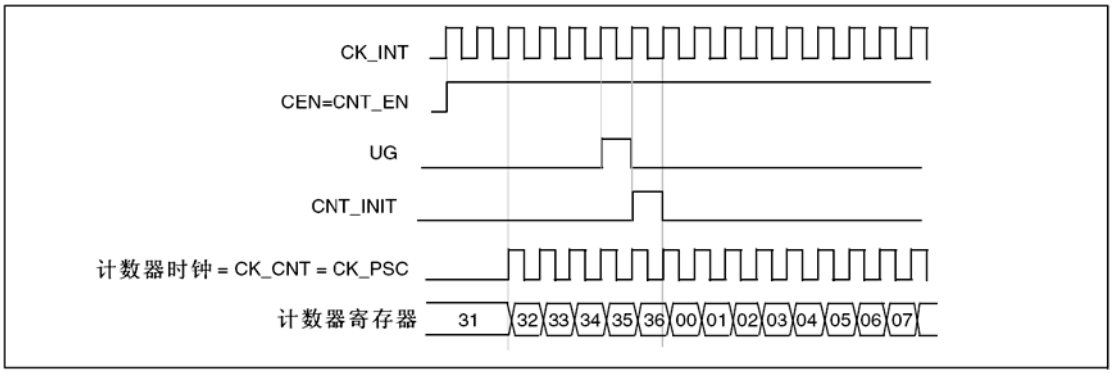


图 152 一般模式下的控制电路，内部时钟分频因子为 1

外部时钟源模式 1 (TIM9 和 TIM12)

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿 或下降沿计数。

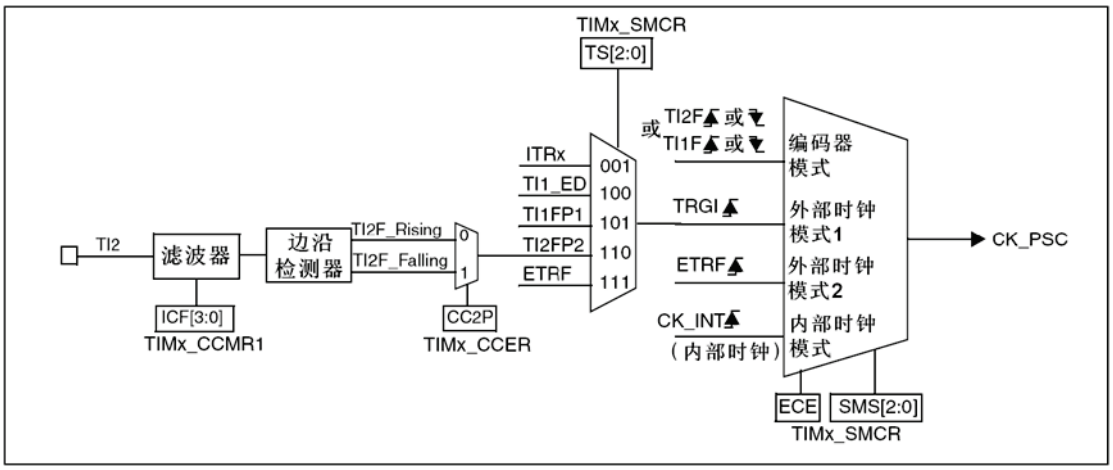


图 153 TI2 外部时钟连接例子

例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置 TIMx_CCMR1 寄存器 CC2S='01'，配置通道 2 检测 TI2 输入的上升沿
2. 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽(如果不需要滤波器，保持 IC2F=0000)

注：捕获预分频器不用作触发，所以不需要对它进行配置

3. 配置 TIMx_CCER 寄存器的 CC2P='0'，选定上升沿极性

4. 配置 TIMx_SMCR 寄存器的 SMS='111'，选择定时器外部时钟模式 1
5. 配置 TIMx_SMCR 寄存器中的 TS='110'，选定 TI2 作为触发输入源
6. 设置 TIMx_CR1 寄存器的 CEN='1'，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

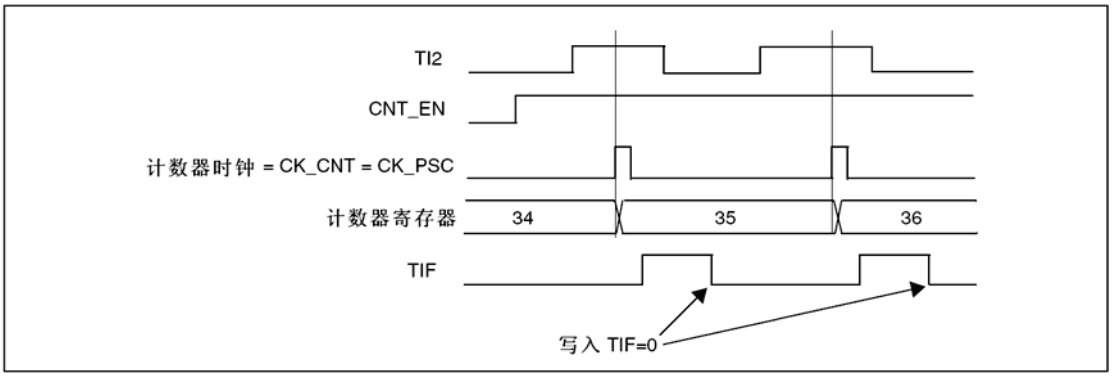


图 154 外部时钟模式 1 下的控制电路

15.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。 下面几张图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘检测器产生一个信号(TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器(ICxPS)。

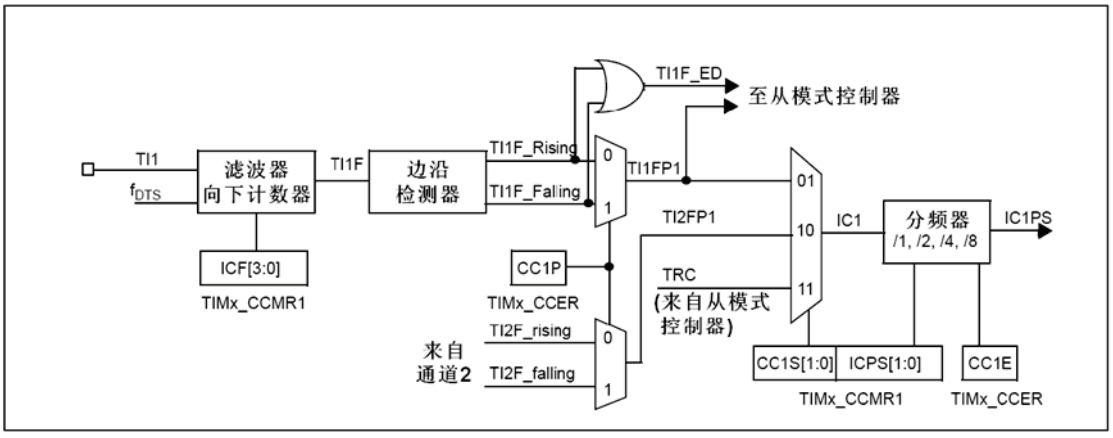


图 155 捕获/比较通道(如：通道 1 输入部分)

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

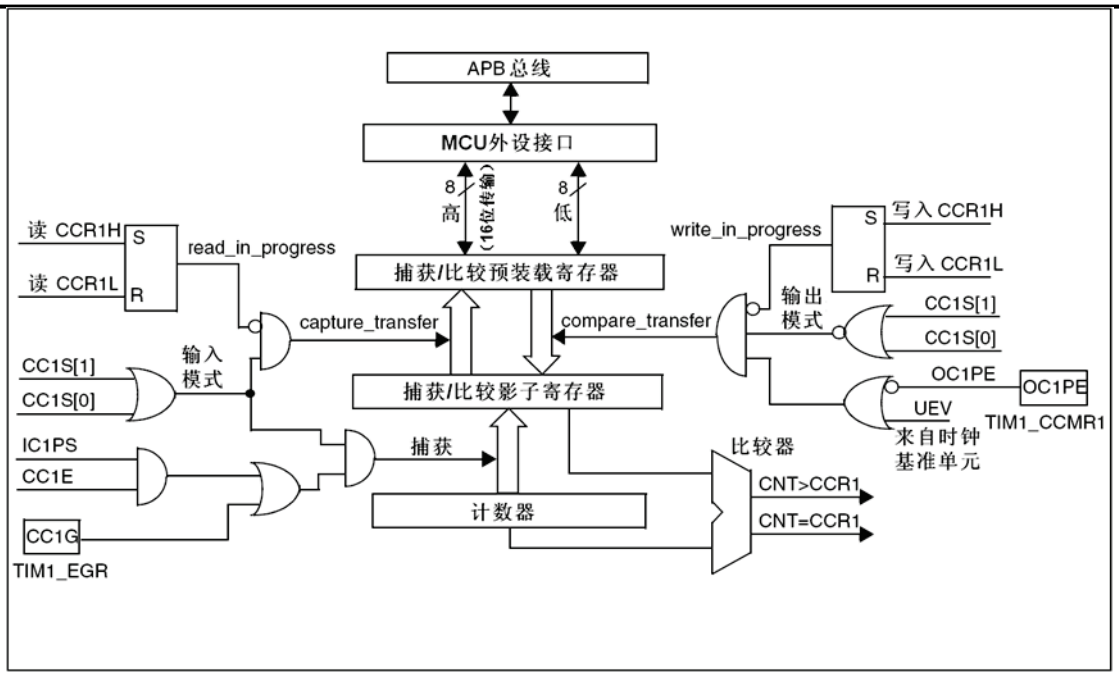


图 156 捕获/比较通道 1 的主电路

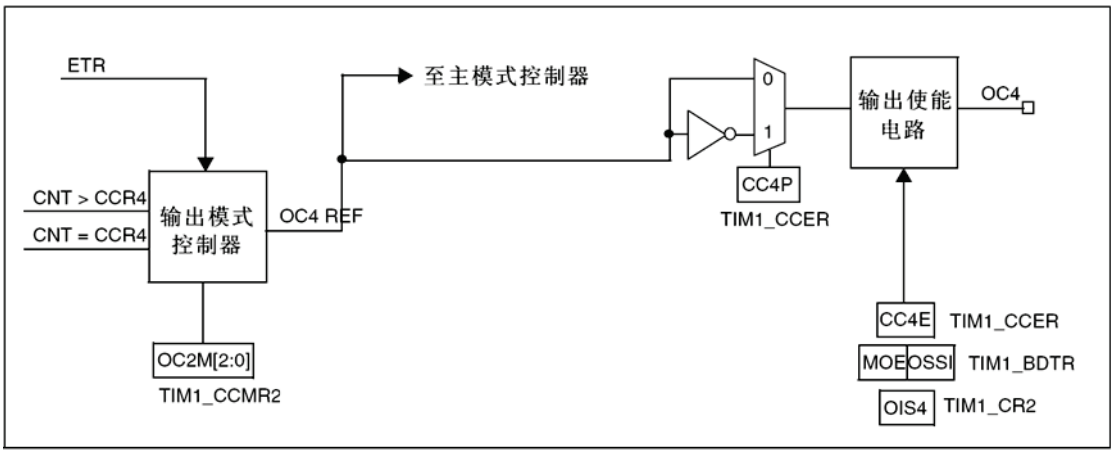


图 157 捕获/比较通道的输出部分(通道 1)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器 进行比较。

15.3.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx_CCRx)中。当捕获事件发生时，相应的 CCxIF 标志(TIMx_SR 寄存器)被置'1'，如果使能了中断操作，则将产生中断操作。如果捕获事件发生时 CCxIF 标志已经为 高，那么重复捕获标志 CCxOF(TIMx_SR 寄存器)被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存 储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

1. 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。

2. 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以(以 fDTS 频率)连续采样 8 次，以确 认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
3. 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0(上升沿)。
4. 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预 分频器被禁止(写 TIMx_CCMR1 寄存器的 IC1PS=00)。
5. 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求。

当发生一个输入捕获时：

1. 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
2. CC1IF 标志被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除， CC1OF 也被置'1'。
3. 中断的产生取决于 CC1IE 位。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢 出标志之后和读取数据之前可能产生的捕获溢出信息。

注： 设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/ 。

15.3.6 PWM 输入模式(仅适用于 TIM9/12)

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的周期(TIMx_CCR1 寄存器)和占空比(TIMx_CCR2 寄存器)，具体步骤如下(取决于 CK_INT 的频率和预分频器的值)

1. 1.选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01(选择 TI1)。
2. 选择 TI1FP1 的有效极性(用来捕获数据到 TIMx_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
3. 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10(选择 TI1)。
4. 选择 TI1FP2 的有效极性(捕获数据到 TIMx_CCR2)：置 CC2P 和 CC2NP 设置为 11(下降沿有效)。
5. 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101(选择 TI1FP1)。
6. 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
7. 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

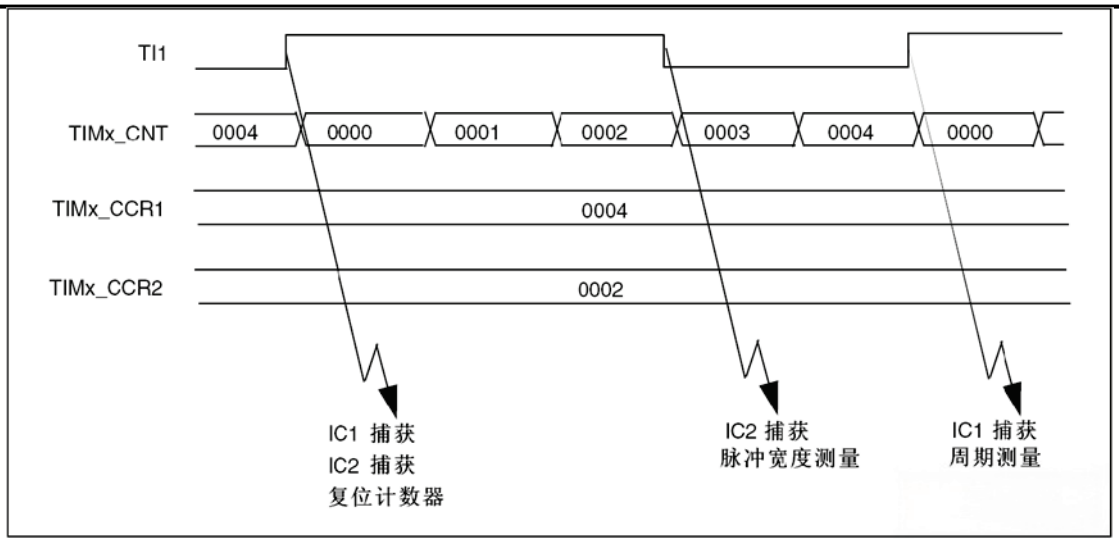


图 158 PWM 输入模式时序

由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器 ，所以 PWM 输入模式只能使用 TIMx_CH1 /TIMx_CH2 信号。

15.3.7 强置输出模式

在输出模式(TIMx_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx)能够 直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状 态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性位相 反的值。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修 改。因此仍然会产生相应的中断请求。这将会在下面的输出比较模式一节中介绍。

15.3.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。 当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作:

1. 将输出比较模式(TIMx_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻 转(OCxM=011)。
2. 设置中断状态寄存器中的标志位(TIMx_SR 寄存器中的 CCxIF 位)。
3. 若设置了相应的中断屏蔽(TIMx_DIER 寄存器中的 CCxIE 位)，则产生一个中断。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。 在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤:

1. 选择计数器时钟(内部，外部，预分频器)

2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出引脚，CCRx 预装载未 用，开启 OCx 输出且高电平有效，则必须设置 OCxM='011' 、OCxPE='0' 、CCxP='0'和 CCxE='1'。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄 存器(OCxPE='0'，否则 TIMx_CCRx 影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

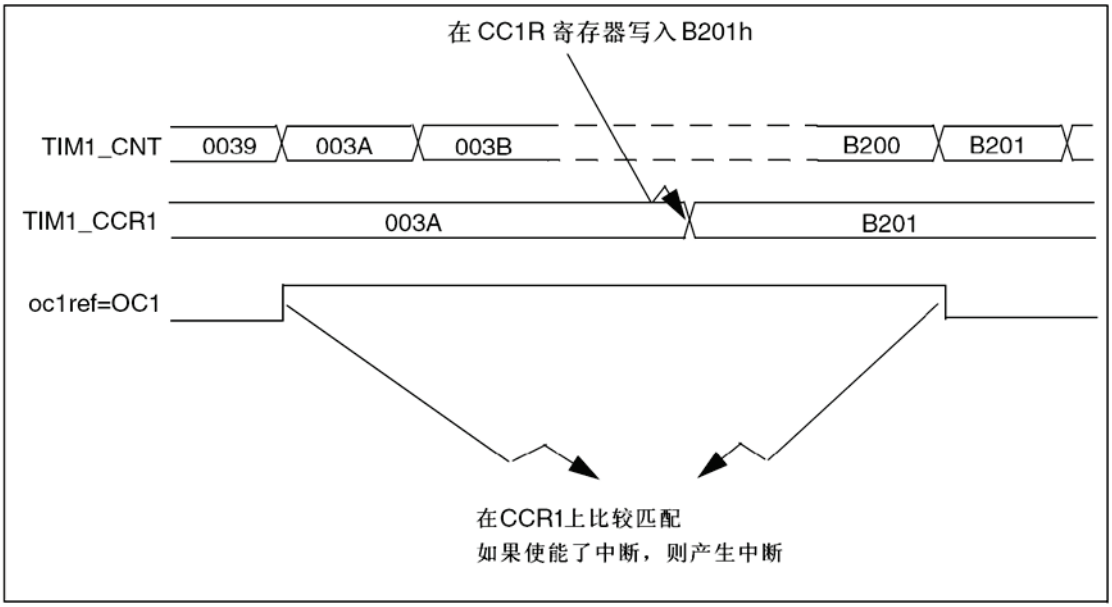


图 159 输出比较模式，翻转 OC1

15.3.9 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空 比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'110'(PWM 模式 1)或'111'(PWM 模式 2)，能够独立地设 置每个 OCx 输出通道产生一路 PWM。必须设置 TIMx_CCMRx 寄存器 OCxPE 位以使能相应的预 装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，使能 自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计 数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或 低电平有效。TIMx_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx_CCERx 寄存器的 描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，(依据计数器的计数方向) 以确定是否符合 TIMx_CCRx ≤ TIMx_CNT 或者 TIMx_CNT ≤ TIMx_CCRx 。

计时器只有在计数器上计时时才能在边缘对齐模式下产生 PWM。

PWM 边沿对齐模式

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时 PWM 信号参考 OCxREF 为高，否 则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值(TIMx_ARR)，则 OCxREF 保持为'1'。

如果比较值为 0，则 OCxREF 保持为'0' 。 下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

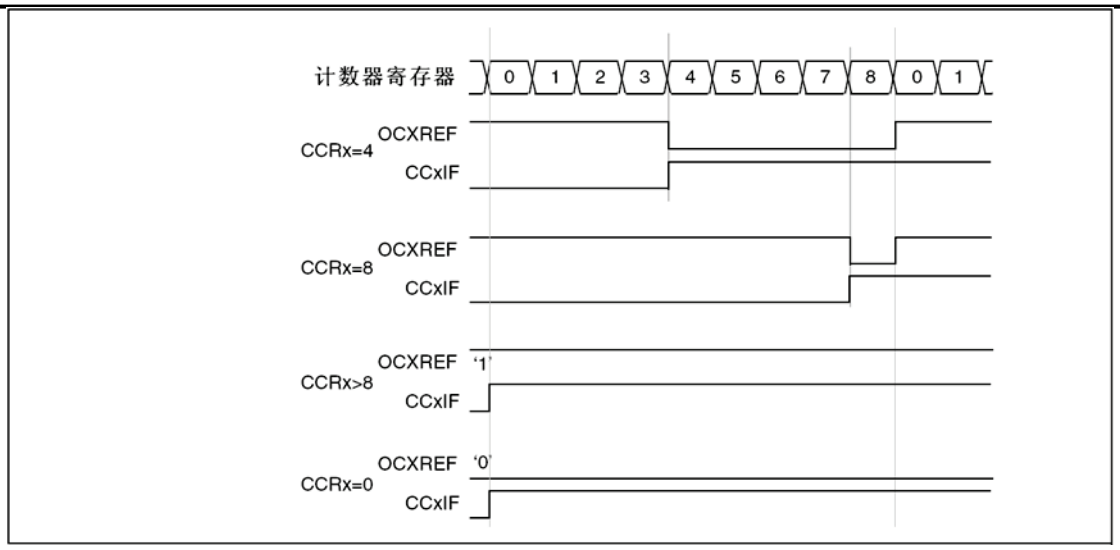


图 160 边沿对齐的 PWM 波形(ARR=8)

15.3.10 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个 程序可控的延时之后，产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新 事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

向上计数方式：CNT < CCRx ≤ ARR (特别地，0 < CCRx)

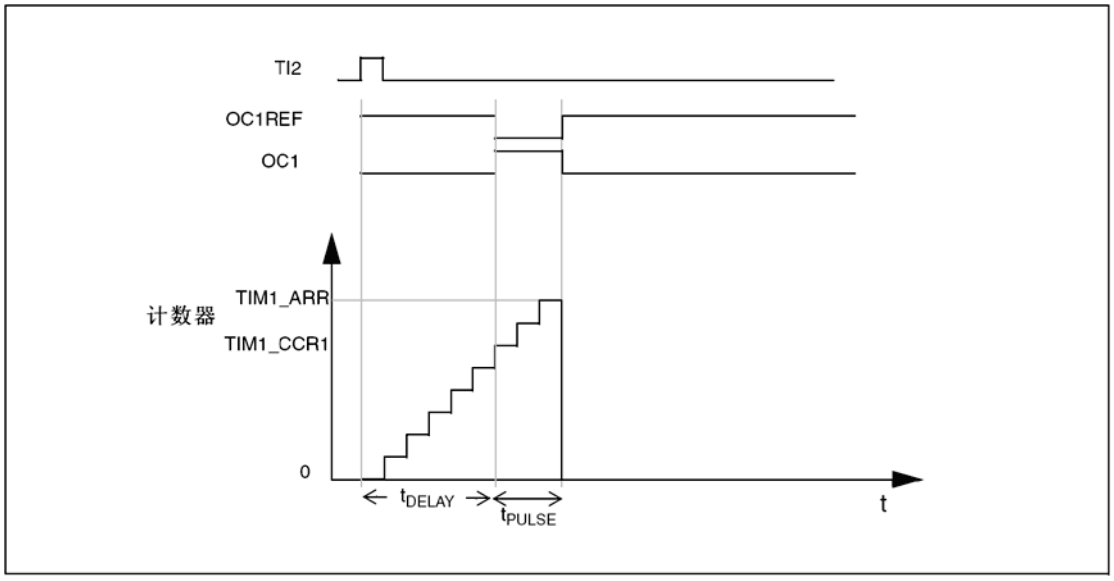


图 161 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发 1：

1. 置 TIMx_CCMR1 寄存器中的 CC2S='01'，把 TI2FP2 映像到 TI2。
2. 置 TIMx_CCER 寄存器中的 CC2P='0'和 CC2NP='0'，使 TI2FP2 能够检测上升沿。

3. 置 TIMx_SMCR 寄存器中的 TS='110' ，TI2FP2 作为从模式控制器的触发(TRGI)。
4. 置 TIMx_SMCR 寄存器中的 SMS='110'(触发模式)，TI2FP2 被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- tDELAY 由写入 TIMx_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx_ARR - TIMx_CCR1)。
- 假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1' 到'0'的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M='111'，进入 PWM 模式 2；根据需要有 选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE='1'和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，修改 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中， CC1P='0'。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。因为只需一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM='1'，在下一个更新事件(当计数 器从自动装载值翻转到 0)时停止计数。

特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的 最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只 在通道配置为 PWM1 和 PWM2 模式时起作用。

15.3.11 TIM9/12 定时器和外部触发的同步

TIM9/12 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx_ARR ，TIMx_CCRx)都会被更新。

在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

1. 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置它。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 和 CC1NP 为 0 以 确定极性(只检测上升沿)。
2. 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=101，选择 TI1 作为输入源。
3. 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx_SR 寄存器中的 TIF 位)被设置，根据 TIMx_DIER 寄 存器中 TIE(中断使能)位的设置，产生一个中断请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之 间的延时，取决于 TI1 输入端的重同步电路。

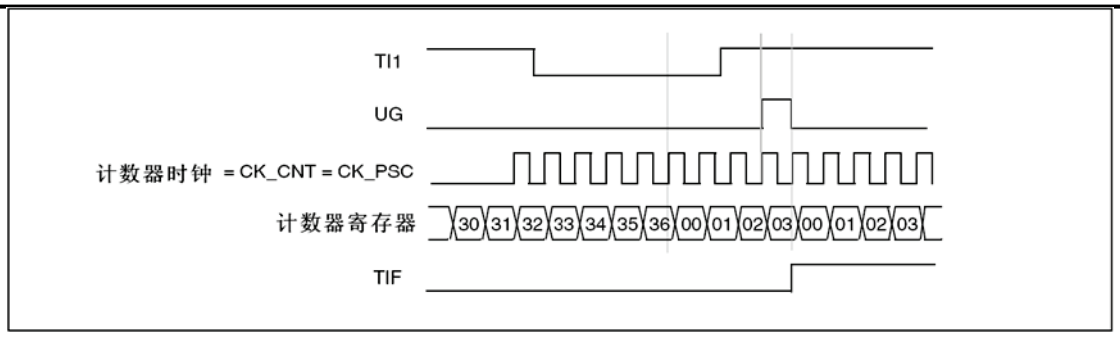


图 162 复位模式下的控制电路

从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

1. 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入 捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 和 CC1NP=0 以确定极 性(只检测低电平)。
2. 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
3. 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能 启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

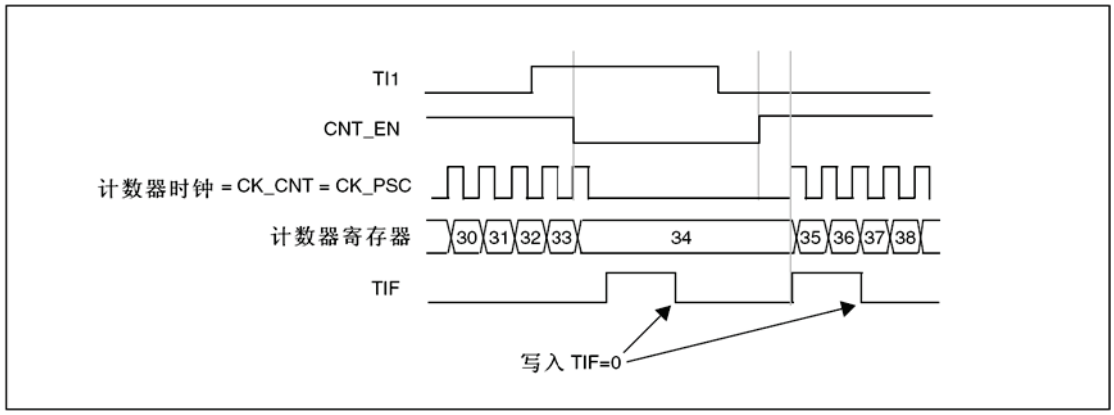


图 163 门控模式下的控制电路

从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

1. 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕 获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 和 CC2NP=0 以确定极性 (只检测低电平)。
2. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。
TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

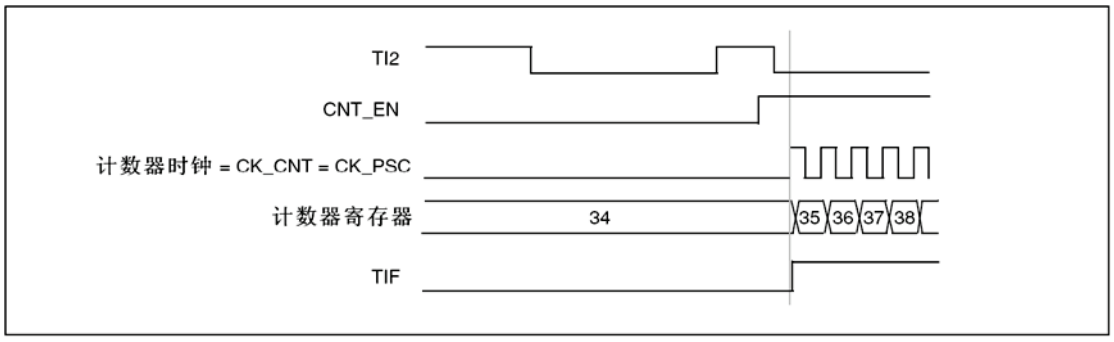


图 164 触发器模式下的控制电路

15.3.12 定时器同步（TIM9/12）

TIM 定时器在内部连接在一起，以实现定时器同步或链结。有关详细信息，请参第 14.3.15 节:定时器一致性。

注意: 从属计时器的时钟必须在从主计时器接收事件之前启用，并且在从主定时器接收触发器时，不得在运行时更改。

15.3.13 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)，根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见随后第 28.16.2 节：支持定时器、看门狗、bxCAN 和 I2C 的调试。

15.4 TIM9/TIM12 寄存器

关于在寄存器描述里面所用到的缩写，详见第 1 节。
可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

15.4.1 TIM9/12 控制寄存器 1(TIMx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	保留			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9:8	CKD[1:0]	**CKD[1:0]**:时钟分频因子(Clock division) 定义在定时器时钟(CK_INT)频率与数字滤波器(ETR，TIx)使用的采样频率之间的分频比例。 00：tDTS=tCK_INT 01：tDTS=2xtCK_INT 10：tDTS=4xtCK_INT 11：保留
7	ARPE	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	Reserved	保留，始终读为 0。

3	OPM	**OPM**：单脉冲模式(One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	URS	**URS**：更新请求源(Update request source)软件通过该位选择 UEV 事件的源 0：如果使能了更新中断请求，则下述任一事件产生更新中断请求： -计数器溢出/下溢 -设置 UG 位 1：如果使能了更新中断请求，则只有计数器溢出/下溢才产生更新中断请求。
1	UDIS	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	**CEN**：使能计数器 0：禁止计数器； 1：使能计数器。 注：在单脉冲模式下，当发生更新事件时，CEN 被自动清除。

15.4.2 TIM9/12 从模式控制寄存器(TIMx_SMCR)

偏移地址：0x08

复位值：0x0000

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

位	符号	说明	
15:7	Reserved	保留	
6:4	TS[2:0]	**TS[2:0]**：触发选择(Trigger selection) 这 3 位选择用于同步计数器的触发输入。 000：内部触发 0(ITR0)，TIM1 001：内部触发 1(ITR1)，TIM2 010：内部触发 2(ITR2)，TIM3 011：内部触发 3(ITR3)，TIM4	100：TI1 的边沿检测器(TI1F_ED) 101：滤波后的定时器输入 1(TI1FP1) 110：滤波后的定时器输入 2(TI2FP2) 111：保留
		关于每个定时器中 ITRx 的细节，参见表 79。 注：这些位只能在未用到(如 SMS=000)时被改变，以避免在改变时产生错误的边沿检测。	
3	Reserved	保留，始终读为 0。	
2:0	SMS[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(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：关闭从模式–如果 CEN=1，则预分频器直接由内部时钟驱动。 001：保留 010：保留 011：保留 100：复位模式–选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式–当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。	

		110：触发模式–计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式 1–选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果 TI1F_EN 被选为触发输入(TS=100)时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。 注:从属计时器的时钟必须在从主计时器接收事件之前启用，并且在从主定时器接收触发器时，不得在运行时更改。

表 79 TIMx 内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM9	TIM2_TRG0	TIM3_TRG0	TIM10_OC	TIM11_OC
TIM12	TIM4_TRG0	TIM5_TRG0	TIM13_OC	TIM14_OC

15.4.3 TIM9/12 中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TIE	保留			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

位	符号	说明
15:7	Reserved	保留，始终读为 0。
6	TIE	**TIE**：触发中断使能(Trigger interrup tenable) 0：禁止触发中断； 1：使能触发中断。
5:3	Reserved	保留，始终读为 0。
2	CC2IE	**CC2IE**：允许捕获/比较 2 中断(Capture/Compare2 interrupt enable) 0：禁止捕获/比较 2 中断； 1：允许捕获/比较 2 中断。
1	CC1IE	**CC1IE**：允许捕获/比较 1 中断(Capture/Compare1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
0	UIE	**UIE**：允许更新中断(Updateinterruptenable) 0：禁止更新中断； 1：允许更新中断。

15.4.4 TIM9/12 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CC2OF	CC1OF	保留		TIF	保留			CC2IF	CC1IF	UIF
					rcw0	rcw0			rcw0				rcw0	rcw0	rcw0

位	符号	说明
15:11	Reserved	保留，始终读为 0。
10	CC2OF	**CC2OF**：捕获/比较 2 重复捕获标记(Capture/Compare2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	**CC1OF**：捕获/比较 1 重复捕获标记(Capture/Compare1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置'1'。写'0'可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。

8:7	Reserved	保留，始终读为 0。
6	TIF	**TIF**：触发器中断标记(Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0：无触发器事件产生； 1：触发器中断等待响应。
5:3	Reserved	保留，始终读为 0。
2	CC2IF	**CC2IF**：捕获/比较 2 中断标记(Capture/Compare2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	**CC1IF**：捕获/比较 1 中断标记(Capture/Compare1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置'1'，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	**UIF**：更新中断标记(Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： -若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新事件(软件对计数器 CNT 重新初始化)； -若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重初始化时产生更新事件。(参考同步控制寄存器的说明)

15.4.5 TIM9/12 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									TG	保留			CC2G	CC1G	UG
									w				w	w	w

位	符号	说明
15:7	Reserved	保留，始终读为 0。
6	TG	**TG**：产生触发事件(Trigger generation) 该位由软件置'1'，用于产生一个触发事件，由硬件自动清'0'。 0：无动作； 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断，则产生相应的中断。
5:3	Reserved	保留，始终读为 0。
2	CC2G	**CC2G**：产生捕获/比较 2 事件(Capture/compare2 generation) 参考 CC1G 描述。
1	CC1G	**CC1G**：产生捕获/比较 1 事件(Capture/compare1 generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断，则产生相应的中断。 若通道 CC1 配置为输入：

		当前的计数器值捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断，则产生相应的中断。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	**UG**：产生更新事件(Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'，若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

15.4.6 TIM9/12 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位	符号	说明
15	OC2CE	**OC2CE**：输出比较 2 清 0 使能(Output compare2 clear enable)
14:12	OC2M[2:0]	**OC2M[2:0]**：输出比较 2 模式(Output compare2 mode)
11	OC2PE	**OC2PE**：输出比较 2 预装载使能(Output compare2 preload enable)
10	OC2FE	**OC2FE**：输出比较 2 快速使能(Output compare2 fast enable)
9:8	CC2S[1:0]	**CC2S[1:0]**：捕获/比较 2 选择(Capture/Compare2 selection) 该位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
7	OC1CE	**OC1CE**：输出比较 1 清 0 使能(Output compare1 clear enable) 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
6:4	OC1M[2:0]	**OC1M[2:0]**：输出比较 1 模式(Output compare1 enable) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。

		110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
3	**OC1PE**	**OC1PE**：输出比较 1 预装载使能(Output compare1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被传送至当前寄存器中。 注：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM='1')，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	**OC1FE**	**OC1FE**：输出比较 1 快速使能(Output compare1 fast enable) 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1:0	**CC1S[1:0]**	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

输入捕获模式：

位	符号	说明
15:12	**IC2F[3:0]**	**IC2F[3:0]**：输入捕获 2 滤波器(Input capture2 filter)
11:10	**IC2PSC[1:0]**	**IC2PSC[1:0]**：输入/捕获 2 预分频器(input capture2 prescaler)
9:8	**CC2S[1:0]**	**CC2S[1:0]**：捕获/比较 2 选择(Capture/compare2selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入，IC2 映射在 TI2 上； 10：CC2 通道被配置为输入，IC2 映射在 TI1 上； 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E='0')才是可写的。
7:4	**IC1F[3:0]**	**IC1F[3:0]**：输入捕获 1 滤波器(Input capture1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 1001：采样频率 fSAMPLING=fDTS/8，N=8 0010：采样频率 fSAMPLING=fCK_INT，N=4 1010：采样频率 fSAMPLING=fDTS/16，N=5 0011：采样频率 fSAMPLING=fCK_INT，N=8 1011：采样频率 fSAMPLING=fDTS/16，N=6 0100：采样频率 fSAMPLING=fDTS/2，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8

		0101：采样频率 fSAMPLING=fDTS/2，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 0110：采样频率 fSAMPLING=fDTS/4，N=6 1110：采样频率 fSAMPLING=fDTS/32，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1111：采样频率 fSAMPLING=fDTS/32，N=8 注：当 ICxF[3:0]=1、2 或 3 时，公式中的 fDTS 由 CK_INT 替代。
3:2	**IC1PSC[1:0]**	**IC1PSC[1:0]**：输入/捕获 1 预分频器(Input capture1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。 一旦 CC1E='0'(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
1:0	**CC1S[1:0]**	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

15.4.7 TIM9/12 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CC2NP	保留	CC2P	CC2E	CC1NP	保留	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

位	符号	说明
15:8	**Reserved**	保留，始终读为 0。
7	**CC2NP**	CC2NP:捕获/比较 2 输出极性参见 CC1NP 描述
6	**Reserved**	保留，始终读为 0。
5	**CC2P**	**CC2P**：输入/捕获 2 输出极性(Capture/Compare2 output polarity) 参考 CC1P 的描述。
4	**CC2E**	**CC2E**：输入/捕获 2 输出使能(Capture/Compare2 output enable) 参考 CC1E 的描述。
3	**CC1NP**	CC1NP:捕获/比较 1 互补输出极性 CC1 通道配置为输出:CC1NP 必须保持清空 CC1 通道配置为输入:CC1NP 与 CC1P 一起使用以定义 TI1FP1/TI2FP1 极性(参见 CC1P 描述)。
2	**Reserved**	保留，始终读为 0。
1	**CC1P**	**CC1P**：输入/捕获 1 输出极性(Capture/Compare1 output polarity) **CC1 通道配置为输出**: 0：OC1 高电平有效 1：OC1 低电平有效 **CC1 通道配置为输入**: 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
0	**CC1E**	**CC1E**：输入/捕获 1 输出使能(Capture/Compare1 output enable) CC1 通道配置为输出：

		0：关闭 - OC1 禁止输出。 1：开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 0：捕获使能。

表 80 标准 OCx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出(OCx=0，OCx_EN=0)
1	OCx=OCxREF+极性，OCx_EN=1

注：　连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

15.4.8 TIM9/12 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

位	符号	说明
15:0	CNT[15:0]	**CNT[15:0]**：计数器的值(Counter value)

15.4.9 TIM9/12 预分频器(TIMx_PSC)

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

位	符号	说明
15:0	PSC[15:0]	**PSC[15:0]**：预分频器的值(Prescaler value) 计数器的时钟频率 CK_CNT 等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值。

15.4.10 TIM9/12 自动重装载寄存器(TIMx_ARR)

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

位	符号	说明
15:0	ARR[15:0]	**ARR[15:0]**:自动重装载的值(Auto reload value) ARR 包含了将要传送至实际的自动重装载寄存器的数值。详细参考 15.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

15.4.11 TIM9/12 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位	符号	说明
15:0	CCR1[15:0]	**CCR1[15:0]**:捕获/比较 1 的值(Capture/Compare1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

15.4.12 TIM9/12 捕获/比较寄存器 2(TIMx_CCR2)

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位	符号	说明
15:0	CCR2[15:0]	**CCR2[15:0]**:捕获/比较 2 的值(Capture/Compare2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。

15.4.13 TIM9/12 寄存器

下表中将 TIM9/12 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 81 TIM9/12 寄存器和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	保留			OPM	URS	UDIS	CEN
	复位值																							0	0	0				0	0	0	0
008h	TIMx_SMCR	保留																									TS[2:0]			保留	SMS[2:0]		
	复位值																										0	0	0		0	0	0
00Ch	TIMx_DIER	保留																									T1E	保留			CC2IE	CC1IE	UIE
	复位值																										0				0	0	0
010h	TIMx_SR	保留																						CC2OF	CC1OF	保留	TI1	保留			CC2IF	CC1IF	UIF
	复位值																							0	0		0				0	0	0
014h	TIMx_EGR	保留																									TG	保留			CC2G	CC1G	UG
	复位值																										0				0	0	0
018h	TIMx_CCMR1输出比较模式	保留																		OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		保留	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	复位值																			0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1输入捕获模式	保留																		IC2F[3:0]					IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]			IC1PSC[1:0]		CC1S[1:0]	
	复位值																			0	0	0	0		0	0	0	0	0	0	0	0	0	0
01Ch	保留																																
020h	TIMx_CCER	保留																								CC2NP	保留	CC2P	CC2E	CC1NP	保留	CC1P	CC1E
	复位值																									0		0	0			0	0
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	保留																																
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	保留																CCR2[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch到04Ch	保留																																

有关寄存器的起始地址，参见表 1。

15.5 TIM10/11/13/14 寄存器

关于在寄存器描述里面所用到的缩写，详见第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

15.5.1 TIM10/11/13/14 控制寄存器 1(TIMx_CR1)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	保留			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9:8	CKD[1:0]	**CKD[1:0]**:时钟分频因子(Clock division) 定义在定时器时钟(CK_INT)频率与数字滤波器(ETR，TIx)使用的采样频率之间的分频比例。 00：tDTS=tCK_INT 01：tDTS=2xtCK_INT 10：tDTS=4xtCK_INT 11：保留
7	ARPE	**ARPE**：自动重装载预装载允许位(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲； 1：TIMx_ARR 寄存器被装入缓冲器。
6:4	Reserved	保留，始终读为 0。
3	OPM	**OPM**：单脉冲模式(One pulse mode) 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
2	URS	**URS**：更新请求源(Update request source)软件通过该位选择 UEV 事件的源 0：如果使能了更新中断请求，则下述任一事件产生更新中断请求： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 1：如果使能了更新中断请求，则只有计数器溢出/下溢才产生更新中断请求。
1	UDIS	**UDIS**：禁止更新(Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0：允许 UEV。更新(UEV)事件由下述任一事件产生： -计数器溢出/下溢 -设置 UG 位 -从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注：更新影子寄存器) 1：禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	**CEN**：使能计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。在单脉冲模式下，当发生更新事件时，CEN 被自动清除。

15.5.2 TIM10/11/13/14 中断使能寄存器(TIMx_DIER)

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

保留	CC1IE	UIE
	rw	rw

位	符号	说明
15:2	Reserved	保留，始终读为 0。
1	CC1IE	**CC1IE**：允许捕获/比较 1 中断(Capture/Compare1 interrupt enable) 0：禁止捕获/比较 1 中断； 1：允许捕获/比较 1 中断。
0	UIE	**UIE**：允许更新中断(Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

15.5.3 TIM10/11/13/14 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CC1OF	保留							CC1IF	UIF
						rcw0								rcw0	rcw0

位	符号	说明
15:10	Reserved	保留，始终读为 0。
9	CC1OF	**CC1OF**：捕获/比较 1 重复捕获标记(Capture/Compare1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置'1'。写'0'可清除该位。 0：无重复捕获产生； 1：当计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为'1'。
8:2	Reserved	保留，始终读为 0。
1	CC1IF	**CC1IF**：捕获/比较 1 中断标记(Capture/Compare1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置'1'，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清'0'。 0：无匹配发生； 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置'1'，它由软件清'0'或通过读 TIMx_CCR1 清'0'。0：无输入捕获产生； 1：计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	**UIF**：更新中断标记(Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。0：无更新事件产生； 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： -若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新事件(软件对计数器 CNT 重新初始化)； -若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重初始化时产生更新事件。(参考同步控制寄存器的说明)

15.5.4 TIM10/11/13/14 事件产生寄存器(TIMx_EGR)

偏移地址:0x14

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														CC1G	UG
														w	w

位	符号	说明
15:2	Reserved	保留，始终读为 0。
1	CC1G	**CC1G**：产生捕获/比较 1 事件(Capture/compare1 generation) 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清'0'。 0：无动作； 1：在通道 CC1 上产生一个捕获/比较事件： 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断，则产生相应的中断。 若通道 CC1 配置为输入： 当前的计数器值捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断，则产生相应的中断。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	**UG**：产生更新事件(Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'，若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

15.5.5 TIM10/11/13/14 捕获/比较模式寄存器 1(TIMx_CCMR1)

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
保留								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位	符号	说明
15:7	Reserved	保留，始终读为 0。
6:4	OC1M[2:0]	**OC1M[2:0]**：输出比较 1 模式(Output compare1 enable) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。

		110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。 111：PWM 模式 2 - 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
3	**OC1PE**	**OC1PE**：输出比较 1 预装载使能(Output compare1 preload enable) 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被传送至当前寄存器中。 注：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM='1')，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
2	**OC1FE**	**OC1FE**：输出比较 1 快速使能(Output compare1 fast enable) 该位用于加快 CC 输出对触发器输入事件的响应。 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1:0	**CC1S[1:0]**	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

输入捕获模式：

位	符号	说明
15:8	**Reserved**	保留，始终读为 0。
7:4	**IC1F[3:0]**	**IC1F[3:0]**：输入捕获 1 滤波器(Input capture1 filter) 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000：无滤波器，以 fDTS 采样 1000：采样频率 fSAMPLING=fDTS/8，N=6 0001：采样频率 fSAMPLING=fCK_INT，N=2 0010：采样频率 fSAMPLING=fCK_INT，N=4 0011：采样频率 fSAMPLING=fCK_INT，N=8 0100：采样频率 fSAMPLING=fDTS/2，N=6 0101：采样频率 fSAMPLING=fDTS/2，N=8 0110：采样频率 fSAMPLING=fDTS/4，N=6 0111：采样频率 fSAMPLING=fDTS/4，N=8 1001：采样频率 fSAMPLING=fDTS/8，N=8 1010：采样频率 fSAMPLING=fDTS/16，N=5 1011：采样频率 fSAMPLING=fDTS/16，N=6 1100：采样频率 fSAMPLING=fDTS/16，N=8 1101：采样频率 fSAMPLING=fDTS/32，N=5 1110：采样频率 fSAMPLING=fDTS/32，N=6

		1111：采样频率 fSAMPLING=fDTS/32，N=8
3:2	IC1PSC[1:0]	**IC1PSC[1:0]**：输入/捕获 1 预分频器(Input capture1 prescaler) 这 2 位定义了 CC1 输入(IC1)的预分频系数。 一旦 CC1E='0'(TIMx_CCER 寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每 2 个事件触发一次捕获； 10：每 4 个事件触发一次捕获； 11：每 8 个事件触发一次捕获。
1:0	CC1S[1:0]	**CC1S[1:0]**：捕获/比较 1 选择(Capture/Compare1 selection) 这 2 位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1 通道被配置为输出； 01：CC1 通道被配置为输入，IC1 映射在 TI1 上； 10：CC1 通道被配置为输入，IC1 映射在 TI2 上； 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E='0')才是可写的。

15.5.6 TIM10/11/13/14 捕获/比较使能寄存器(TIMx_CCER)

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												CC1NP	保留	CC1P	CC1E
												rw		rw	rw

位	符号	说明
15:4	Reserved	保留，始终读为 0。
3	CC1NP	CC1NP:捕获/比较 1 个互补输出极性。 CC1 通道配置为输出:CC1NP 必须保持清空。 CC1 信道配置为输入:CC1NP 位与 CC1P 一起使用以定义 TI1FP1 极性(参见 CC1P 描述)。
2	Reserved	保留，始终读为 0。
1	CC1P	**CC1P**：输入/捕获 1 输出极性(Capture/Compare1 output polarity) CC1 通道配置为输出： 0：OC1 高电平有效 1：OC1 低电平有效 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0：不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1：反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
0	CC1E	**CC1E**：输入/捕获 1 输出使能(Capture/Compare1 output enable) CC1 通道配置为输出： 0：关闭 - OC1 禁止输出。 1：开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 0：捕获使能。

表 82 标准 OCx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出(OCx=0，OCx_EN=0)
1	OCx=OCxREF+极性，OCx_EN=1

注：　连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

15.5.7 TIM10/11/13/14 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

位	符号	说明
15:0	CNT[15:0]	**CNT[15:0]**：计数器的值(Counter value)

15.5.8 TIM10/11/13/14 预分频器(TIMx_PSC)

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

位	符号	说明
15:0	PSC[15:0]	**PSC[15:0]**：预分频器的值(Prescaler value) 计数器的时钟频率 CK_CNT 等于 fCK_PSC/(PSC[15:0]+1)。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值。

15.5.9 TIM10/11/13/14 自动重装载寄存器(TIMx_ARR)

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

位	符号	说明
15:0	ARR[15:0]	**ARR[15:0]**:自动重装载的值(Auto reload value) ARR 包含了将要传送至实际的自动重装载寄存器的数值。详细参考 15.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

15.5.10 TIM10/11/13/14 捕获/比较寄存器 1(TIMx_CCR1)

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

位	符号	说明
15:0	CCR1[15:0]	**CCR1[15:0]**:捕获/比较 1 的值(Capture/Compare1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

15.5.11 TIM10/11/13/14 寄存器

下表中将 TIM10/11/13/14 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 83 TIM10/11/13/14 寄存器和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																						CKD[1:0]		ARPE	保留			OPM	URS	UDIS	CEN
	复位值																							0	0	0				0	0	0	0
008h	保留																																
00Ch	TIMx_DIER	保留																														CC1IE	UIE
	复位值																															0	0
010h	TIMx_SR	保留																						CC1OF	保留							CC1IF	UIF
	复位值																							0								0	0
014h	TIMx_EGR	保留																														CC1G	UG
	复位值																															0	0
018h	TIMx_CCMR1输出比较模式	保留																									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	复位值																										0	0	0	0	0	0	0
	TIMx_CCMR1输入捕获模式	保留																								IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
	复位值																									0	0	0	0	0	0	0	0
01Ch	保留																																
020h	TIMx_CCER	保留																												CC1NP	保留	CC1P	CC1E
	复位值																													0		0	0
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	保留																																
034h	TIMx_CCR1	保留																CCR1[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h~04Ch	保留																																

有关寄存器的起始地址，参见表 1。

16 基本定时器(TIM6 和 TIM7)

16.1 TIM6 和 TIM7 简介

基本定时器 TIM6 和 TIM7 各包含一个 16 位自动装载计数器，由各自的可编程预分频器驱动。它们可以作为通用定时器提供时间基准，特别地可以为数模转换器(DAC)提供时钟。实际上，它们在芯片内部直接连接到 DAC 并通过触发输出直接驱动 DAC。这 2 个定时器是互相独立的，不共享任何资源。

16.2 TIM6 和 TIM7 的主要特性

TIM6 和 TIM7 定时器的主要功能包括：

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

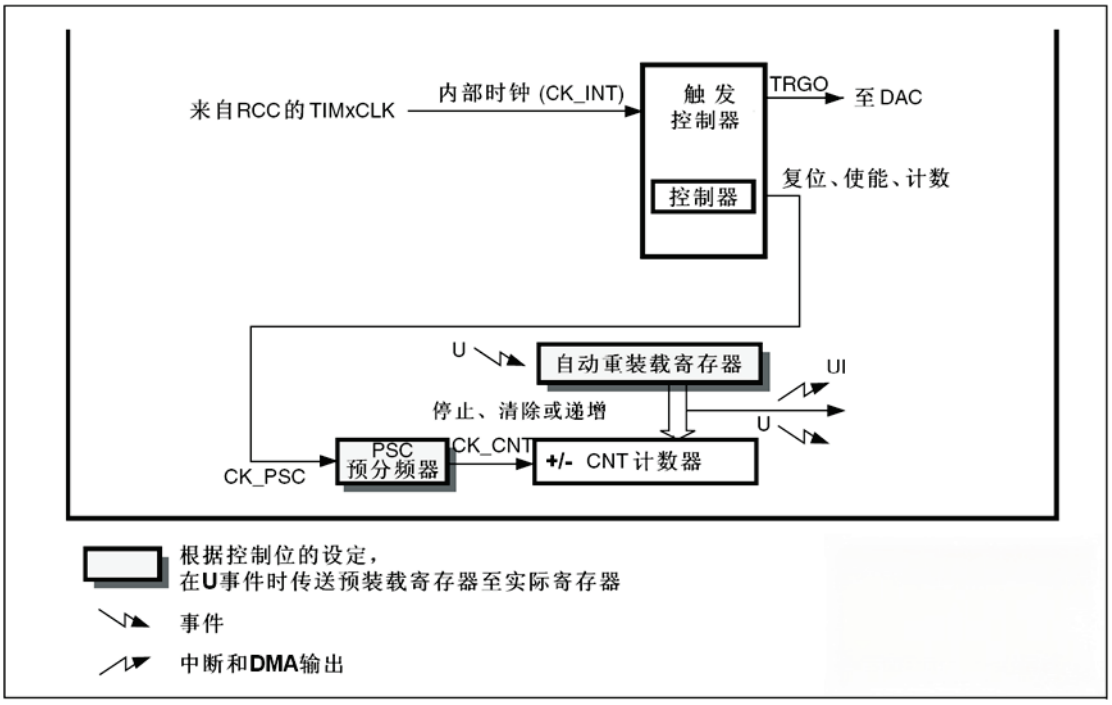


图 165 基本定时器框图

16.3 TIM6 和 TIM7 的功能

16.3.1 时基单元

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

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

- 计数器寄存器(TIMx_CNT)
- 预分频寄存器(TIMx_PSC)
- 自动重装载寄存器(TIMx_ARR)

自动重装载寄存器是预加载的，每次读写自动重装载寄存器时，实际上是通过读写预加载寄存器实现。根据 TIMx_CR1 寄存器中的自动重装载预加载使能位(ARPE)，写入预加载寄存器的内容能够立即或在每次更新事件时，传送到它的影子寄存器。当 TIMx_CR1 寄存器的 UDIS 位为'0'，则每当计数器达到溢出值时，硬件发出更新事件；软件也可以产生更新事件；关于更新事件的产生，随后会有详细的介绍。

计数器由预分频输出 CK_CNT 驱动，设置 TIMx_CR1 寄存器中的计数器使能位(CEN)使能计数器计数。

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

预分频器

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

以下两图是在运行过程中改变预分频系数的例子。

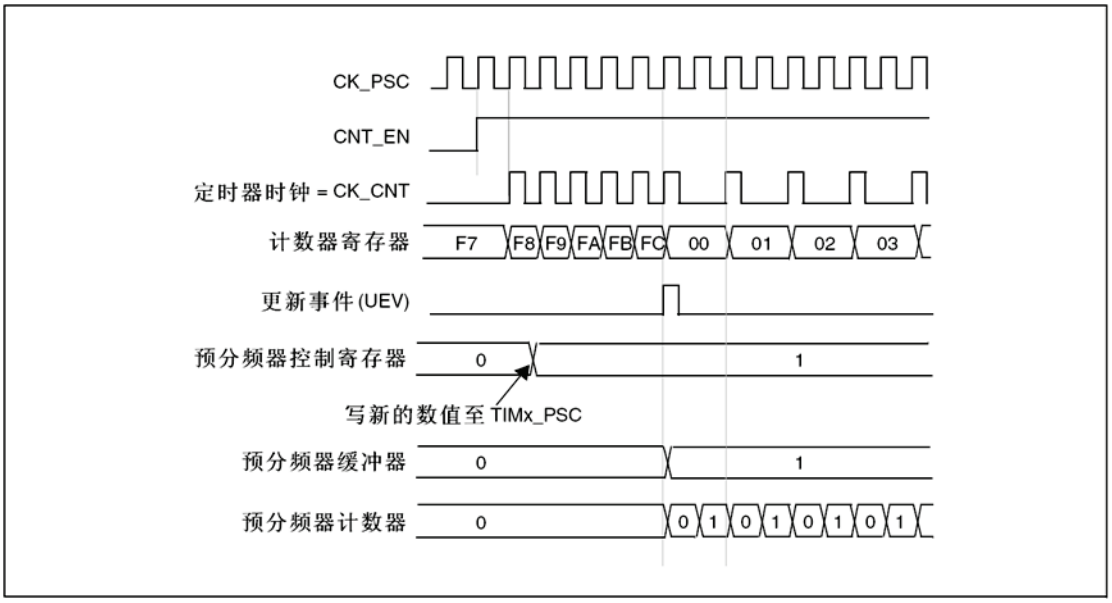


图 166 预分频系数从 1 变到 2 的计数器时序图

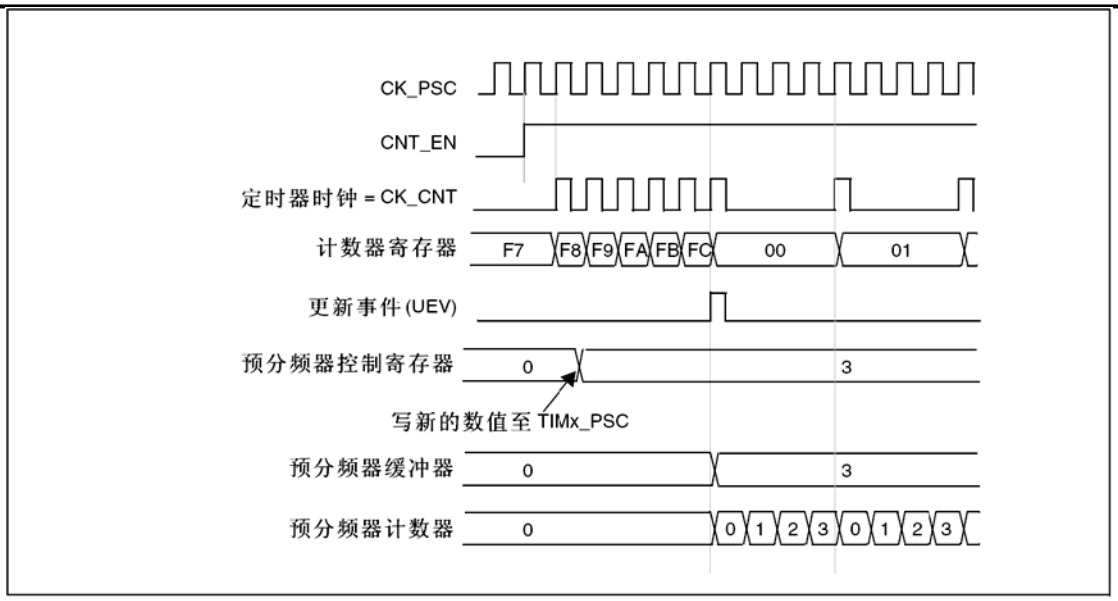


图 167 预分频系数从 1 变到 4 的计数器时序图

16.3.2 计数模式

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

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

设置 TIMx_CR1 中的 UDIS 位可以禁止产生 UEV 事件，这可以避免在写入预加载寄存器时更改影子寄存器。在清除 UDIS 位为'0'之前，将不再产生更新事件，但计数器和预分频器依然会在应产生更新事件时重新从 0 开始计数(但预分频系数不变)。另外，如果设置了 TIMx_CR1 寄存器中的 URS(选择更新请求)，设置 UG 位可以产生一次更新事件 UEV，但不设置 UIF 标志(即没有中断)。

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

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

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

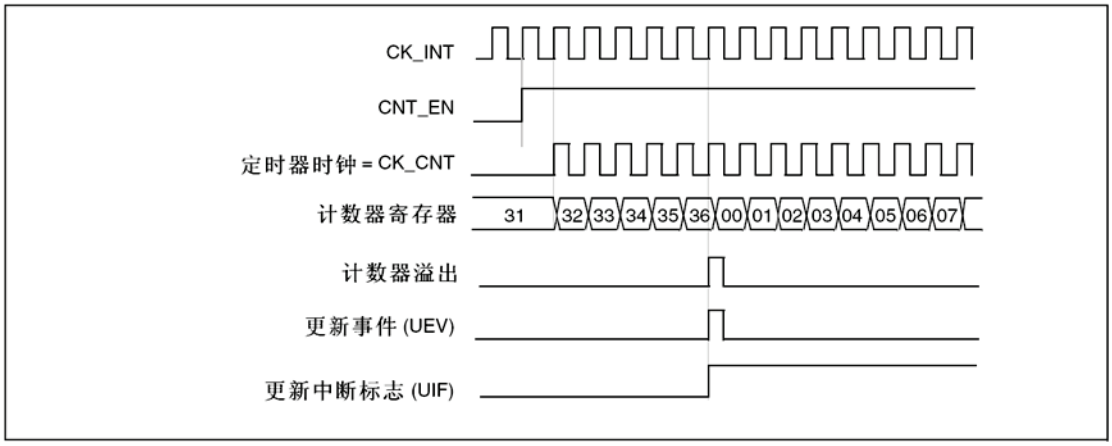


图 168 计数器时序图，内部时钟分频系数为 1

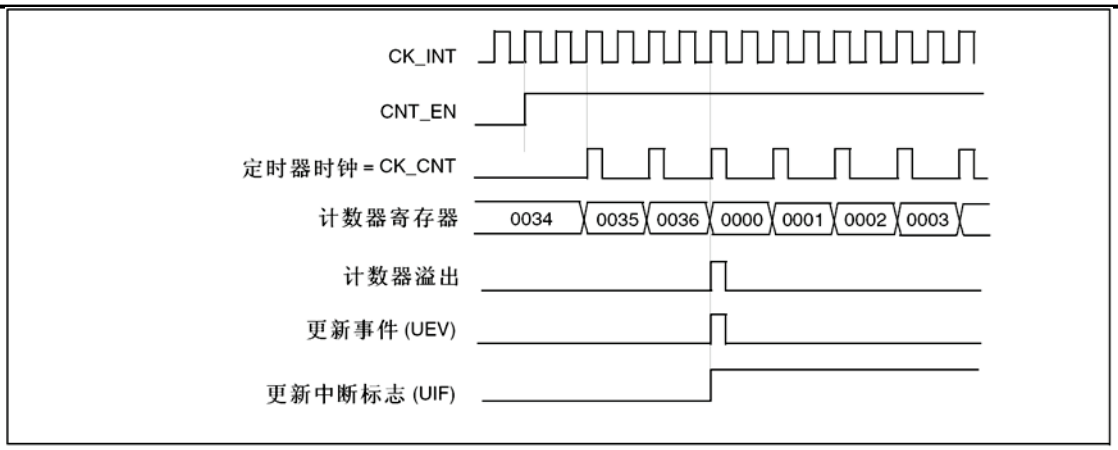


图 169 计数器时序图，内部时钟分频系数为 2

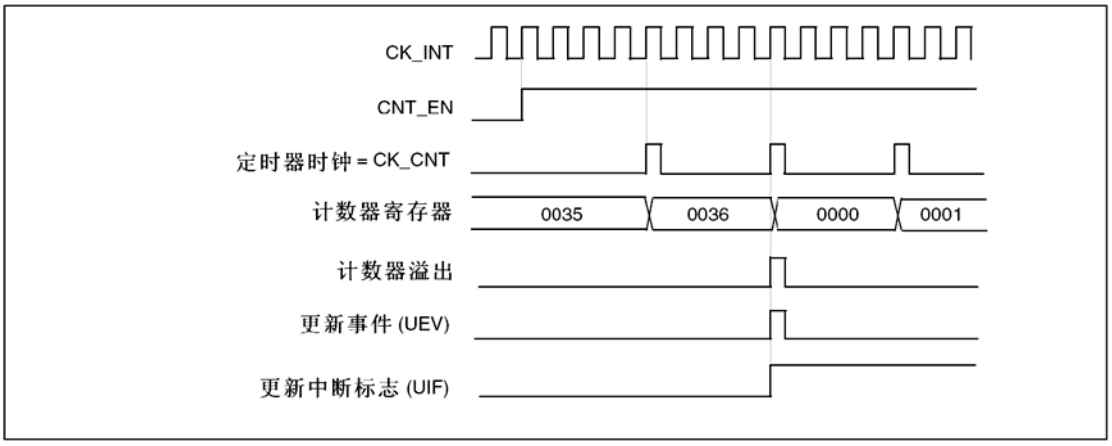


图 170 计数器时序图，内部时钟分频系数为 4

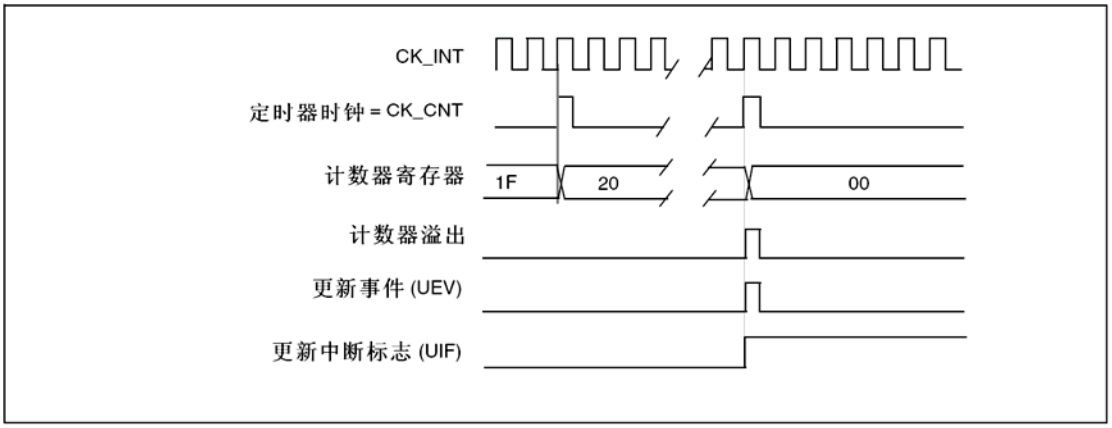


图 171 计数器时序图，内部时钟分频系数为 N

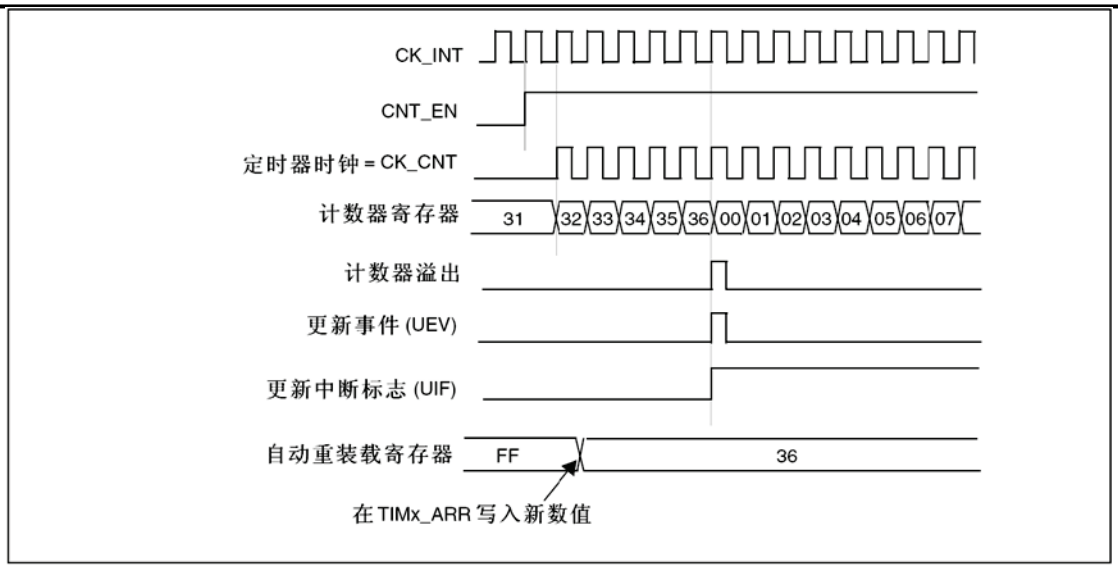


图 172 计数器时序图，当 ARPE=0 时的更新事件(TIMx_ARR 没有预装载)

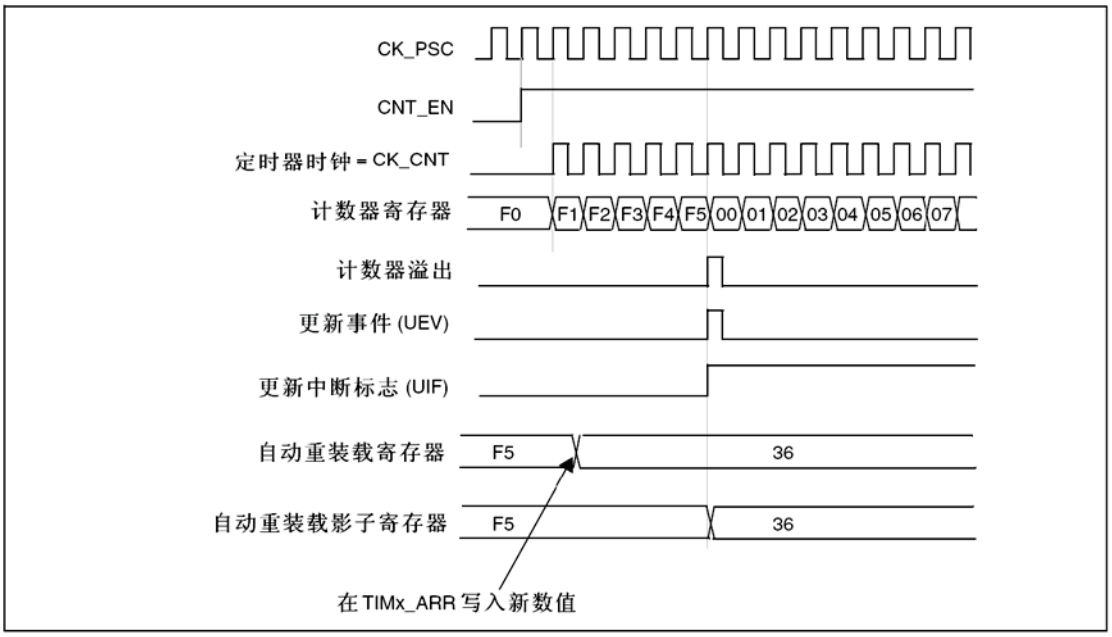


图 173 计数器时序图，当 ARPE=1 时的更新事件(预装载 TIMx_ARR)

16.3.3 时钟源

计数器的时钟由内部时钟(CK_INT)提供。

TIMx_CR1 寄存器的 CEN 位和 TIMx_EGR 寄存器的 UG 位是实际的控制位，(除了 UG 位被自动清除外)只能通过软件改变它们。一旦置 CEN 位为'1'，内部时钟即向预分频器提供时钟。

下图示出控制电路和向上计数器在普通模式下，没有预分频器时的操作。

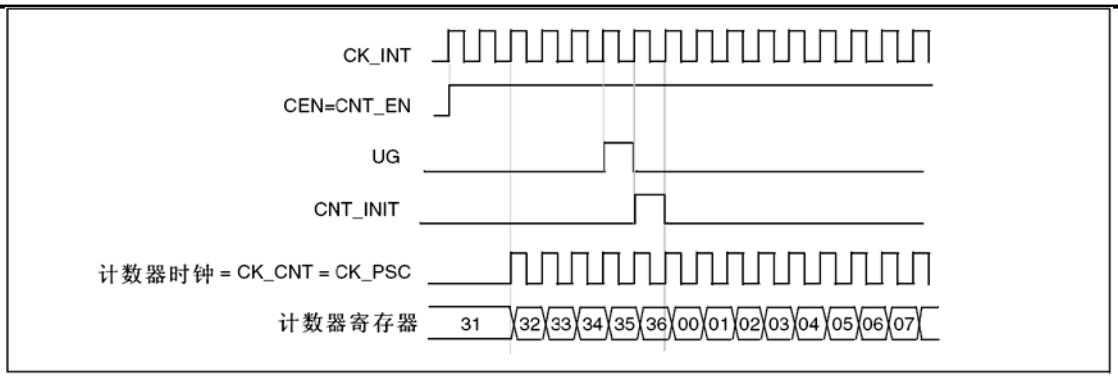


图 174 普通模式时序图，内部时钟分频系数为 1

16.3.4 调试模式

当微控制器进入调试模式(Cortex-M3 核心停止)时，根据 DBG 模块中的配置位 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续计数或者停止工作。详见第 28.16.2 节。

16.4 TIM6 和 TIM7 寄存器

有关寄存器描述中用到的缩写，请参考第 1 节。
可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

16.4.1 TIM6 和 TIM7 控制寄存器 1(TIMx_CR1)

偏移地址：0x00
复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ARPE	保留			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

位	符号	说明
15:8	Reserved	保留，始终读为 0。
7	ARPE	**ARPE**：自动重装载预装载使能(Auto-reload preload enable) 0：TIMx_ARR 寄存器没有缓冲 1：TIMx_ARR 寄存器具有缓冲
6:4	Reserved	保留，始终读为 0。
3	OPM	**OPM**：单脉冲模式(One-pulse mode) 0：在发生更新事件时，计数器不停止 1：在发生下次更新事件时，计数器停止计数(清除 CEN 位)。
2	URS	URS：更新请求源(Update request source) 该位由软件设置和清除，以选择 UEV 事件的请求源。 0：如果使能了中断，以下任一事件可以产生一个更新中断请求： -计数器上溢或下溢 -设置 UG 位 -通过从模式控制器产生的更新 1：如果使能了中断，只有计数器上溢或下溢可以产生更新中断请求。
1	UDIS	**UDIS**：禁止更新(Update disable) 该位由软件设置和清除，以使能或禁止 UEV 事件的产生。0：UEV 使能。更新事件(UEV)可以由下列事件产生： -计数器上溢或下溢 -设置 UG 位

		-通过从模式控制器产生的更新 产生更新事件后，带缓冲的寄存器被加载为预加载数值。 1：禁止 UEV。不产生更新事件(UEV)，影子寄存器保持它的内容(ARR、PSC)。但是如果设置了 UG 位或从模式控制器产生了一个硬件复位，则计数器和预分频器将被重新初始化。
0	**CEN**	**CEN**：计数器使能(Counter enable) 0：关闭计数器 1：使能计数器 注：门控模式只能在软件已经设置了 CEN 位时有效，而触发模式可以自动地由硬件设置 CEN 位。 在单脉冲模式下，当产生更新事件时 CEN 被自动清除。

16.4.2 TIM6 和 TIM7 控制寄存器 2(TIMx_CR2)

偏移地址：0x04

复位值：0x0000

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

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

16.4.3 TIM6 和 TIM7DMA/中断使能寄存器(TIMx_DIER)

偏移地址：0x0C

复位值：0x0000

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

位	符号	说明
15:9	**Reserved**	保留，始终读为 0。
8	**UDE**	**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
7:1	**Reserved**	保留，始终读为 0。
0	**UIE**	**UIE**：更新中断使能(Update interrupt enable)

		0：禁止更新中断 1：使能更新中断

16.4.4 TIM6 和 TIM7 状态寄存器(TIMx_SR)

偏移地址：0x10

复位值：0x0000

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

位	符号	说明
15:1	Reserved	保留，始终读为 0。
0	UIF	**UIF**：更新中断标志(Update interrupt flag) 硬件在更新中断时设置该位，它由软件清除。 0：没有产生更新。 1：产生了更新中断。下述情况下由硬件设置该位： –计数器产生上溢或下溢并且 TIMx_CR1 中的 UDIS=0； –如果 TIMx_CR1 中的 URS=0 并且 UDIS=0，当使用 TIMx_EGR 寄存器的 UG 位重新初始化计数器 CNT 时。

16.4.5 TIM6 和 TIM7 事件产生寄存器(TIMx_EGR)

偏移地址：0x14

复位值：0x0000

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

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

16.4.6 TIM6 和 TIM7 计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

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

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

16.4.7 TIM6 和 TIM7 预分频器(TIMx_PSC)

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

位	符号	说明
15:0	PSC[15:0]	**PSC[15:0]**：预分频器数值(Prescaler value) 计数器的时钟频率 CK_CNT 等于 fCK_PSC/(PSC[15:0]+1)。 在每一次更新事件时，PSC 的数值被传送到实际的预分频寄存器中。

16.4.8 TIM6 和 TIM7 自动重装载寄存器(TIMx_ARR)

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

位	符号	说明
15:1	ARR[15:0]	**ARR[15:0]**：自动重装载数值(Prescaler value) ARR 的数值将传送到实际的自动重装载寄存器中。 关于 ARR 的更新和作用，详见 14.3.1。 如果自动重装载数值为 0，则计数器停止。

16.4.9 TIM6 和 TIM7 寄存器

下表中将 TIMx 的所有寄存器映射到一个 16 位可寻址(编址)空间。

表 84 TIM6 和 TIM7 寄存器和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	保留																								ARPE	保留			OPM	URS	UDIS	CEN
	复位值																									0				0	0	0	0
004h	TIMx_CR2	保留																										MMS [2:0]			保留				
	复位值																											0	0	0				
008h	保留																																
00Ch	TIMx_DIER	保留																							UDE	保留							UIE
	复位值																								0								0
010h	TIMx_SR	保留																															UIF
	复位值																																0
014h	TIMx_EGR	保留																															UG
	复位值																																0
018h	保留																																
01Ch	保留																																
020h	保留																																
024h	TIMx_CNT	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	TIMx_PSC	保留																PSC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	保留																ARR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器的起始地址，参见表 1。

17 实时时钟(RTC)

17.1 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)处于后备区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对后备寄存器和 RTC 的访问被禁止，这是为了防止对后备区域(BKP)的意外写操作。

执行以下操作将使能对后备寄存器和 RTC 的访问：

- 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位，使能电源和后备接口时钟
- 设置寄存器 PWR_CR 的 DBP 位，使能对后备寄存器和 RTC 的访问。

17.2 主要特性

- 可编程的预分频系数：分频系数最高为 220。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟(RTC 时钟的频率必须小于 PCLK1 时钟频率的四分之一以上)。
- 可以选择以下三种 RTC 的时钟源：
  - HSE 时钟除以 128；
  - LSE 振荡器时钟；
  - LSI 振荡器时钟(详见 6.2.8 节 RTC 时钟)。
- 2 个独立的复位类型：
  - APB1 接口由系统复位；
  - RTC 核心(预分频器、闹钟、计数器和分频器)只能由后备域复位(详见 6.1.3 节)。
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号(最长可达 1 秒)。
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态。

17.3 功能描述

17.3.1 概述

RTC 由两个主要部分组成(参见下图)。第一部分(APB1 接口)用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作(参见 17.4 节)。APB1 接口由 APB1 总线时钟驱动，用来与 APB1 总线接口。

另一部分(RTC 核心)由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 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。

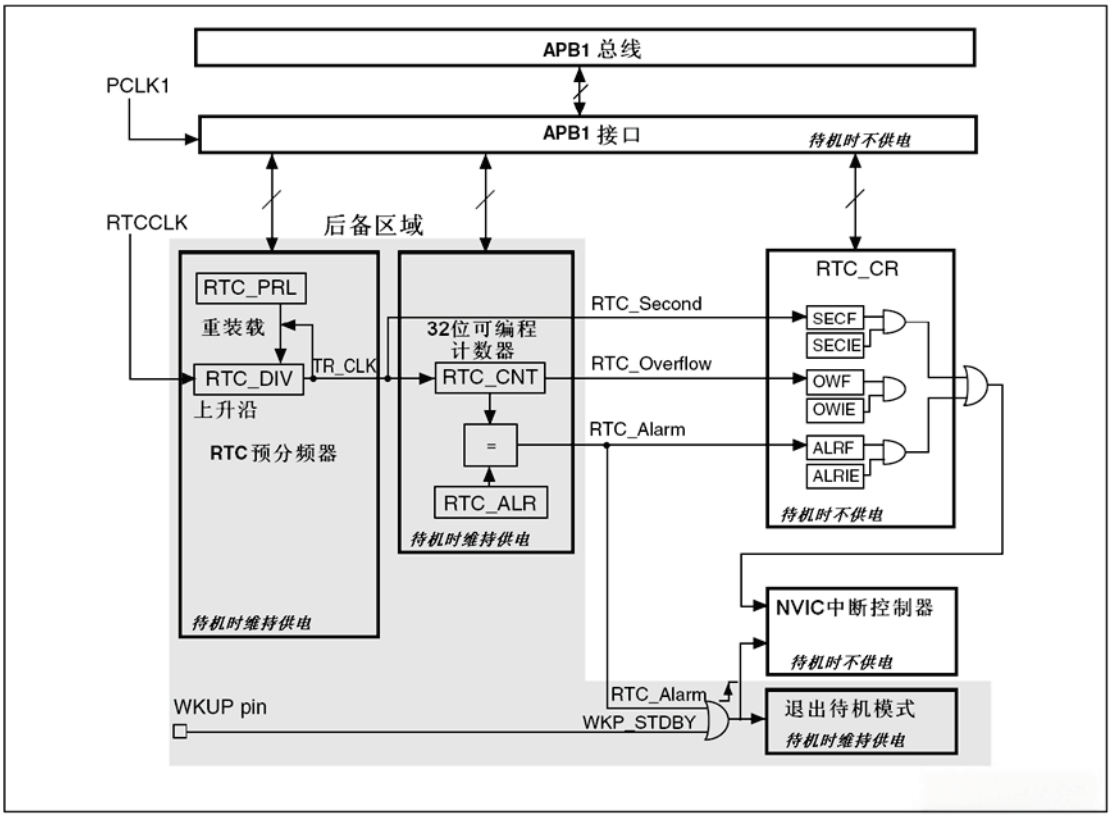


图 175 简化的 RTC 框图

17.3.2 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，详见第 6.1.3 节。

17.3.3 读 RTC 寄存器

RTC 核完全独立于 RTCAPB1 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 RTCAPB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次的内部寄存器更新之前，从 APB1 上读出的 RTC 寄存器数值可能被破坏了(通常读到 0)。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒(参见第 4.3 节：低功耗模式)。
- 系统刚从停机模式唤醒(参见第 4.3 节：低功耗模式)。

所有以上情况中，APB1 接口被禁止时(复位、无时钟或断电)RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC_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 WFI 和 WFE 等低功耗模式的影响。

17.3.4 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。

17.3.5 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。

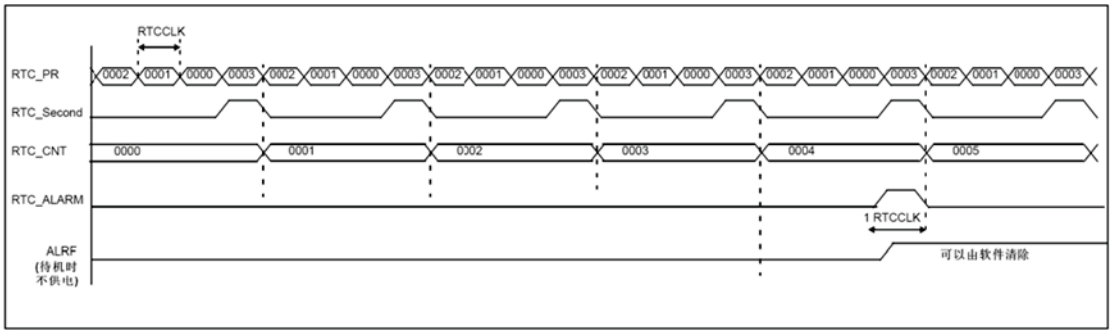


图 176 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004

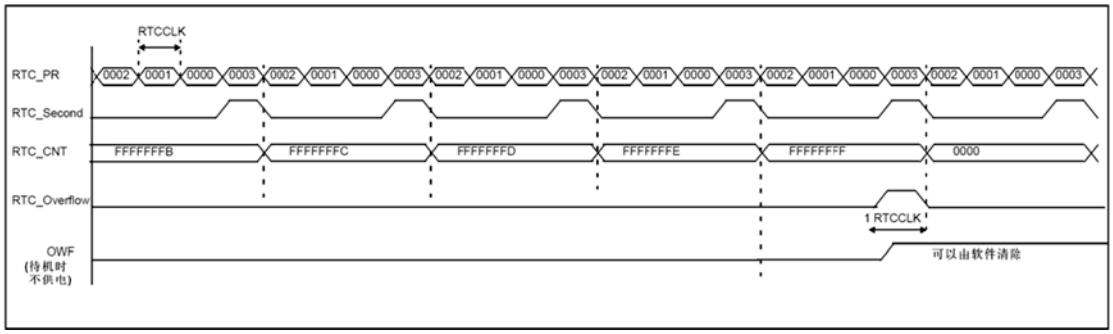


图 177 RTC 溢出波形图示例，PR=0003

17.4 RTC 寄存器描述

关于寄存器描述中的缩略词，请参考第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

17.4.1 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
保留													OWIE	ALRIE	SECIE
													rw	rw	rw

位	符号	说明
15:3	Reserved	保留，被硬件强制为 0。
2	OWIE	**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
1	ALRIE	**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
0	SECIE	**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。

注意：系统复位后所有的中断被屏蔽，因此可通过写 RTC 寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时(标志位 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(见 17.3.4 节)。

17.4.2 RTC 控制寄存器低位(RTC_CRL)

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

位	符号	说明
15:6	Reserved	保留，被硬件强制为 0。
5	RTOFF	**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。
4	CNF	**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。
3	RSF	**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。
2	OWF	**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。
1	ALRF	**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。
0	SECF	**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(RTOFF=0 时，详见 17.3.4 节)，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)。

17.4.3 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(参见 17.3.4 节)

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												PRL[19:16]			
												w	w	w	w

位	符号	说明
15:4	Reserved	保留，被硬件强制为 0。
3:0	PRL[19:16]	**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： fTR_CLK=fRTCCLK/(PRL[19:0]+1) 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。

RTC 预分频装载寄存器低位（RTC_PRLL)

偏移地址：0x0C

只写(参见 17.3.4 节)

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

位	符号	说明
15:0	PRL[15:0]	**PRL[15:0]**：RTC 预分频装载值高位(RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： fTR_CLK=fRTCCLK/(PRL[19:0]+1) 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。

注: 如果输入时钟频率(fRTCCLK)为 32.768 千赫，则在此寄存器中写入 7FFFh 以获得 1 秒的信号周期

17.4.4 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
保留												RTC_DIV[19:16]			
												r	r	r	r

位	符号	说明
15:6	Reserved	保留，被硬件强制为 0。
3:0	RTC_DIV [19:16]	**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

位	符号	说明
15:6	RTC_DIV [15:0]	**RTC_DIV[15:0]**：RTC 时钟器余数低位(RTC clock divider low)

17.4.5 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

位	符号	说明
15:6	RTC_CNT [31:16]	**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(参见 17.3.4 节)。

RTC 计数器寄存器低位(RTC_CNTL)

偏移地址：0x1C

复位值：0x0000

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

位	符号	说明
15:6	RTC_CNT [15:0]	**RTC_CNT[15:0]**：RTC 计数器低位(RTC counter high) 可通过读 RTC_CNTH 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前，必须先进入配置模式(参见 17.3.4 节)。

17.4.6 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(参见 17.3.4 节)

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

位	符号	说明
15:6	RTC_ALR [31:16]	**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(参见 17.3.4 节)。

RTC 闹钟寄存器低位(RTC_ALRL)

偏移地址：0x24

只写(参见 17.3.4 节)

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

位	符号	说明
15:0	RTC_ALR [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(参见 17.3.4 节)。

17.4.7 RTC 寄存器映像

RTC 寄存器是 16 位可寻址寄存器，具体描述如下：

表 85 RTC-寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	RTC_CRH	保留																													OWIE	ALRIE	SECIE
	复位值																														0	0	0
004h	RTC_CRL	保留																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	复位值																											0	0	0	0	0	0
008h	RTC_PRLH	保留																												PRL[19:16]			
	复位值																													0	0	0	0
00Ch	RTC_PRLL	保留																PRL[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	RTC_DIVH	保留																DIV[31:16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	RTC_DIVL	保留																DIV[15:0]															
	复位值																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	RTC_CNTH	保留																CNT[31:16]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	RTC_CNTL	保留																CNT[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	RTC_ALRH	保留																ALR[31:16]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
024h	RTC_ALRL	保留																ALR[15:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

有关寄存器的起始地址，参见表 1。

18 独立看门狗(IWDG)

18.1 简介

W55MH32 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。

关于窗口看门狗的详情，请参看第 17 章。

18.2 IWDG 主要性能

- 自由运行的递减计数器
- 时钟由独立的 RC 振荡器提供(可在停止和待机模式下工作)
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

18.3 IWDG 功能描述

图 178 为独立看门狗模块的功能框图。

在键寄存器(IWDG_KR)中写入 0xCCCC ，开始启用独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号(IWDG_RESET)。

无论何时，只要在键寄存器 IWDG_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。

18.3.1 硬件看门狗

如果用户在选择字节中启用了“硬件看门狗”功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

18.3.2 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。

18.3.3 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。详见有关调试模块的章节。

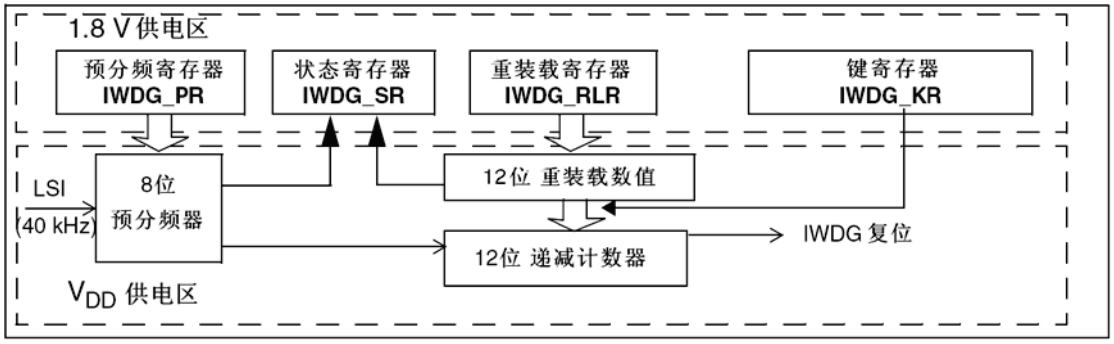


图 178 独立看门狗框图

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 86 看门狗超时时间(40kHz 的输入时钟(LSI)

预分频系数	PR[2:0]位	最短时间 (ms)RL[11:0]=0x000	最长时间 (ms)RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 或 7)	6.4	26214.4

注：　这些时间是按照 40kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 之间变化。

此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。

通过对 LSI 进行校准可获得相对精确的看门狗超时时间。有关 LSI 校准的问题，详见 6.2.5 节。

18.4 IWDG 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

18.4.1 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
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:0	KEY[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。(见 18.3.2 节) 写入 0xCCCC，启动看门狗工作(若选择了硬件看门狗则不受此命令字限制)。

18.4.2 预分频寄存器(IWDG_PR)

地址偏移：0x04

复位值：0x0000 0000

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

位	符号	说明
31:3	Reserved	保留，始终读为 0。
2:0	PR[2:0]	**PR[2:0]**:预分频因子(Prescaler divider) 这些位具有写保护设置，参见 18.3.2 节。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR 寄存器的 PVU 位必须为 0。 000:预分频因子=4　　100:预分频因子=64 001:预分频因子=8　　101:预分频因子=128 010:预分频因子=16　　110:预分频因子=256 011:预分频因子=32　　111:预分频因子=256 注意：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当 IWDG_SR 寄存器的 PVU 位为 0 时，读出的值才有效。

18.4.3 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
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:12	Reserved	保留，始终读为 0。
11:0	RL[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，参看 18.3.2 节。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，参照表 86。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。

18.4.4 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
保留															

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

位	符号	说明
31:2	Reserved	保留，始终读为 0。
1	RVU	**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 VDD 域中的重装载更新结束后，此位由硬件清'0'(最多需 5 个 40kHz 的 RC 周期)。重装载值只有在 RVU 位被清'0'后才可更新。
0	PVU	**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 VDD 域中的预分频值更新结束后，此位由硬件清'0'(最多需 5 个 40kHz 的 RC 周期)。预分频值只有在 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。)

18.4.5 IWDG 寄存器映像

表 87 IWDG 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	IWDG_KR	保留																KEY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	IWDG_PR	保留																														PR[2:0]	
	复位值																															0	0	0
008h	IWDG_RLR	保留																RL[11:0]															
	复位值																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00Ch	IWDG_SR	保留																														RVU	PVU
	复位值																															0	0

有关寄存器的起始地址，参见表 1。

19 窗口看门狗(WWDG)

19.1 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。

19.2 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。见 0。
- 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。

19.3 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。

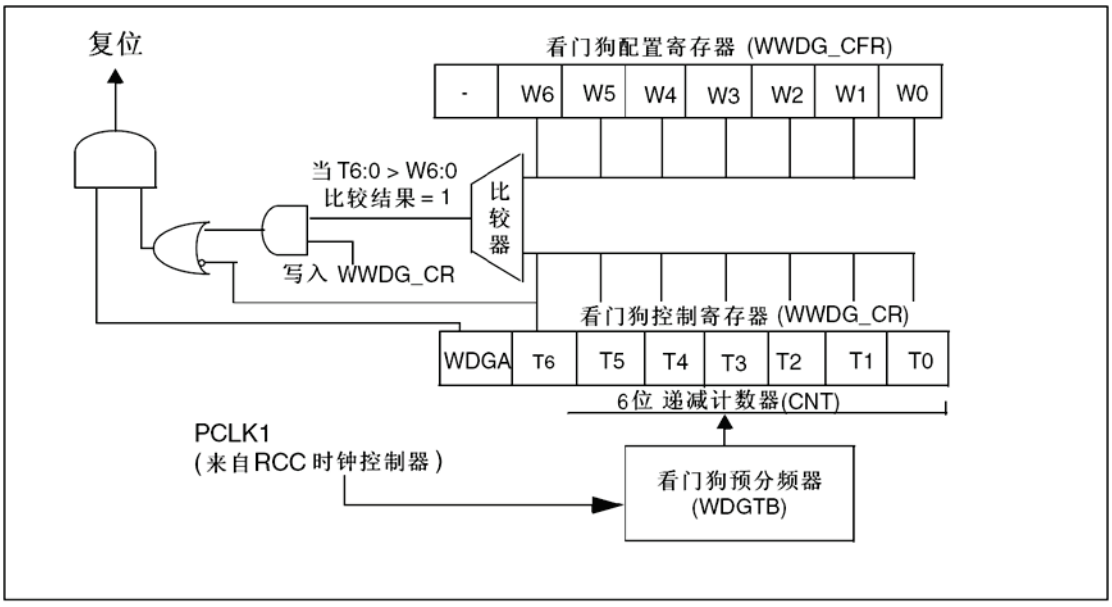


图 179 看门狗框图

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。当看门狗被启用时，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。

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，0 描述了窗口寄存器的工作过程。

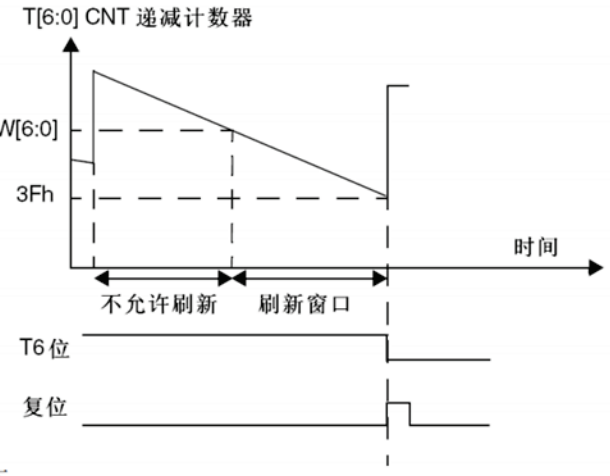
另一个重装载计数器的方法是利用早期唤醒中断(EWI)。设置 WWDG_CFR 寄存器中的 WEI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序(ISR)可以用 来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写'0'可以清除该中断。

注：可以用 T6 位产生一个软件复位(设置 WDGA 位为1'，T6 位为0')。

19.4 如何编写看门狗超时程序

可以使用 0 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 = 36MHz时的最小-最大超时值

WDGTB	最小超时值	最大超时值
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

图 180 窗口看门狗时序图

19.5 调试模式

当微控制器进入调试模式时(Cortex-M3 核心停止)，根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见第 28.16.2 节。

19.6 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 节。
可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

19.6.1 控制寄存器(WWDG_CR)

地址偏移量：0x00
复位值：0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								WDGA	T6	T5	T4	T3	T2	T1	T0
								rs	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:8	Reserved	保留，始终读为 0。
7	WDGA	**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：启用看门狗
6:0	T[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(4096x2WDGTB)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。

19.6.2 配置寄存器(WWDG_CFR)

地址偏移量：0x04
复位值：0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						EWI	WDGTB1	WDGTB0	W6	W5	W4	W3	W2	W1	W0
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:8	Reserved	保留，始终读为 0。
9	EWI	**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。
8:7	WDGTB[1:0]	**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

6:0	W[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。

19.6.3 状态寄存器(WWDG_SR)

地址偏移量：0x08

复位值：0x0000 0000

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

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

位	符号	说明
31:1	Reserved	保留，始终读为 0。
0	EWIF	**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'。

19.6.4 WWDG 寄存器映像

表 88 WWDG 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	WWDG_CR	保留																								WDGA	T[6:0]						
	复位值																									0	1	1	1	1	1	1	1
004h	WWDG_CFR	保留																						EWI	WDGTB1	WDGTB0	W[6:0]						
	复位值																							0	0	0	1	1	1	1	1	1	1
008h	WWDG_SR	保留																															EWIF
	复位值																																0

有关寄存器的起始地址，参见表 1。

20 SDIO 接口(SDIO)

20.1 SDIO 主要功能

SD/SDIOMMC 卡主机模块(SDIO)在 AHB 外设总线和多媒体卡(MMC)、SD 存储卡、SDIO 卡和 CE-ATA 设备间提供了操作接口。

多媒体卡系统规格书由 MMCA 技术委员会发布，可以在多媒体卡协会的网站上(www.mmca.org)获得。

CE-ATA 系统规格书可以在 CE-ATA 工作组的网站上(www.ce-ata.org)获得。SDIO 的主要功能如下：

- 与多媒体卡系统规格书版本 4.2 全兼容。支持三种不同的数据总线模式：1 位(默认)、4 位和 8 位。
- 与较早的多媒体卡系统规格版本全兼容(向前兼容)。
- 与 SD 存储卡规格版本 2.0 全兼容。
- 与 SDI/O 卡规格版本 2.0 全兼容：支持良种不同的数据总线模式：1 位(默认)和 4 位。
- 完全支持 CE-ATA 功能(与 CE-ATA 数字协议版本 1.1 全兼容)。
- 8 位总线模式下数据传输速率可达 48MHz。
- 数据和命令输出使能信号，用于控制外部双向驱动器。

注： *1．SDIO 没有 SPI 兼容的通信模式*

2．在多媒体卡系统规格书版本 2.11 中，定义 SD 存储卡协议是多媒体卡协议的超集。只支持 I/O 模式的 SD 卡或复合卡中的 I/O 部分不能支持 SD 存储设备中很多需要的命令，这里有些命令在 SDI/O 设备中不起作用，如擦除命令，因此 SDIO 不支持这些命令。另外，SD 存储卡和 SDI/O 卡中有些命令是不同的，SDIO 也不支持这些命令。细节可以参考 SDI/O 卡规格书版本 1.0。使用现有的 MMC 命令机制，在 MMC 接口上可以实现 CE-ATA 的支持。SDIO 接口的电气和信号定义详见 MMC 参考资料。

多媒体卡/ SD 总线将所有卡与控制器相连。

当前版本的 SDIO 在同一时间里只能支持一个 SD/SDIO/MMC4.2 卡，但可以支持多个 MMC 版本 4.1 或以前版本的卡。

20.2 SDIO 总线拓扑

总线上的通信是通过传送命令和数据实现。

在多媒体卡/SD/SDI/O 总线上的基本操作是命令/响应结构，这样的总线操作在命令或总线机制下实现信息交换；另外，某些操作还具有数据令牌。

在 SD/SDIO 存储器卡上传送的数据是以数据块的形式传输；在 MMC 上传送的数据是以数据块或数据流的形式传输；在 CE-ATA 设备上传送的数据也是以数据块的形式传输。

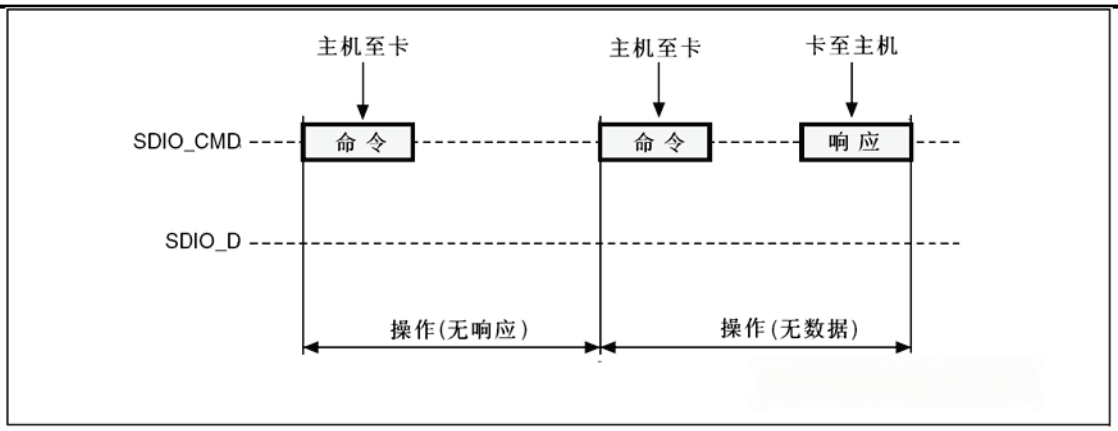


图 181 SDIO“无响应”和“无数据”操作

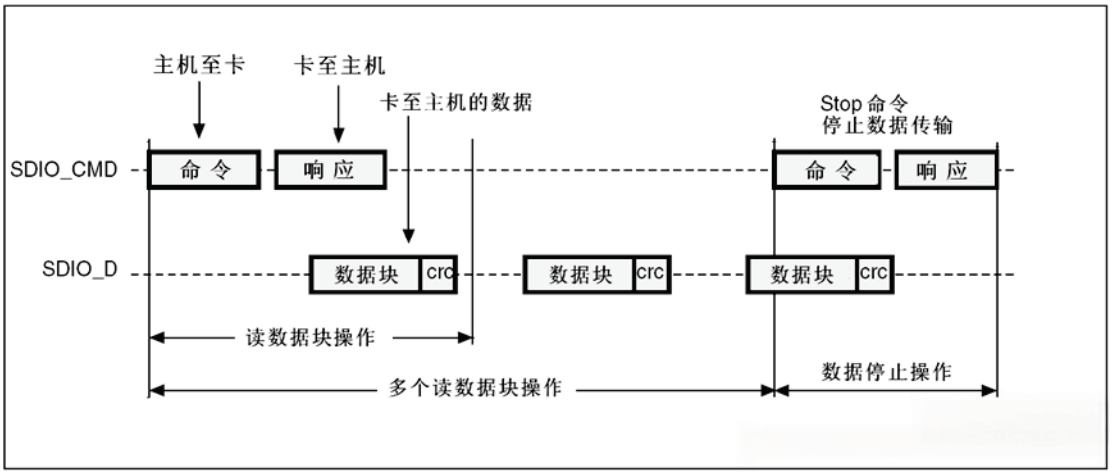


图 182 SDIO(多)数据块读操作

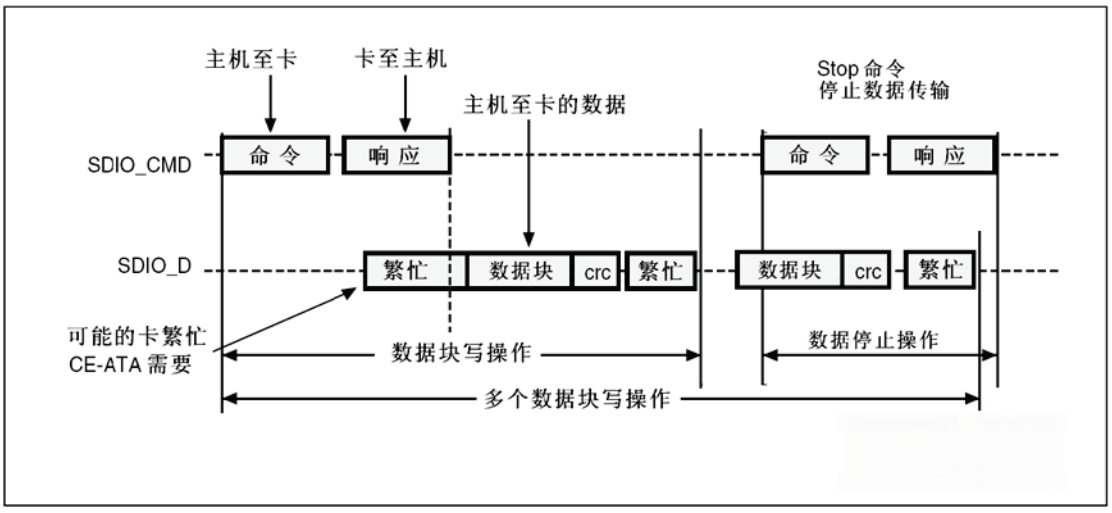


图 183 SDIO(多)数据块写操作

注：　当有 Busy(繁忙)信号时，SDIO(SDIO_D0 被拉低)将不会发送任何数据。

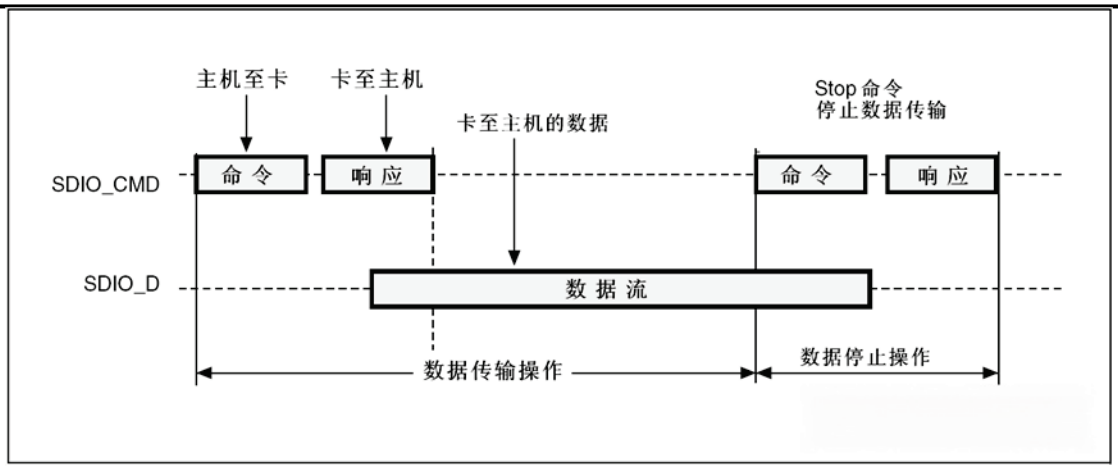


图 184 SDIO 连续读操作

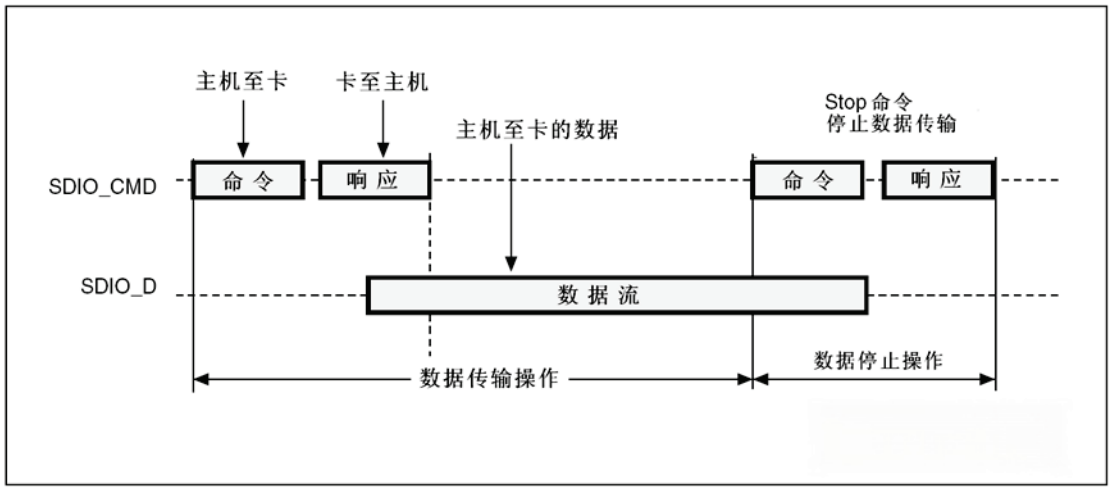


图 185 SDIO 连续写操作

20.3 SDIO 功能描述

SDIO 包含 2 个部分：

- SDIO 适配器模块：实现所有 MMC/SD/SDI/O 卡的相关功能，如时钟的产生、命令和数据的传送。
- AHB 总线接口：操作 SDIO 适配器模块中的寄存器，并产生中断和 DMA 请求信号。

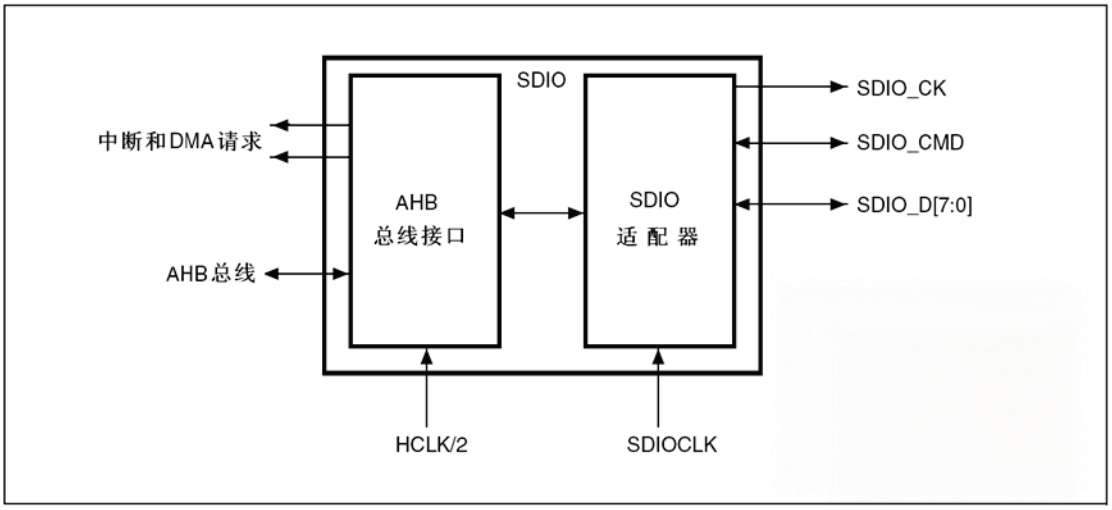


图 186 SDIO 框图

复位后默认情况下 SDIO_D0 用于数据传输。初始化后主机可以改变数据总线的宽度。

如果一个多媒体卡接到了总线上，则 SDIO_D0、SDIO_D[3:0]或 SDIO_D[7:0]可以用于数据传输。

MMC 版本 V3.31 和之前版本的协议只支持 1 位数据线，所以只能用 SDIO_D0。

如果一个 SD 或 SDI/O 卡接到了总线上，可以通过主机配置数据传输使用 SDIO_D0 或 SDIO_D[3:0]。

所有的数据线都工作在推挽模式。

SDIO_CMD 有两种操作模式：

- 用于初始化时的开路模式(仅用于 MMC 版本 V3.31 或之前版本)
- 用于命令传输的推挽模式(SD/SDI/O 卡和 MMCV4.2 在初始化时也使用推挽驱动)

SDIO_CK 是卡的时钟：每个时钟周期在命令和数据线上传输 1 位命令或数据。对于多媒体卡 V3.31 协议，时钟频率可以在 0MHz 至 20MHz 间变化；对于多媒体卡 V4.0/4.2 协议，时钟频率可以在 0MHz 至 48MHz 间变化；对于 SD 或 SDI/O 卡，时钟频率可以在 0MHz 至 25MHz 间变化。

SDIO 使用两个时钟信号：

- SDIO 适配器时钟(SDIOCLK=HCLK)
- AHB 总线时钟(HCLK/2)

下表适用于多媒体卡/SD/SDI/O 卡总线：

表 89 SDIO 引脚定义

引脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO 卡时钟。这是从主机至卡的时钟线。
SDIO_CMD	双向	多媒体卡/SD/SDIO 卡命令。这是双向的命令/响应信号线。
SDIO_D[7:0]	双向	多媒体卡/SD/SDIO 卡数据。这些是双向的数据总线。

20.3.1 SDIO 适配器

下图是简化的 SDIO 适配器框图：

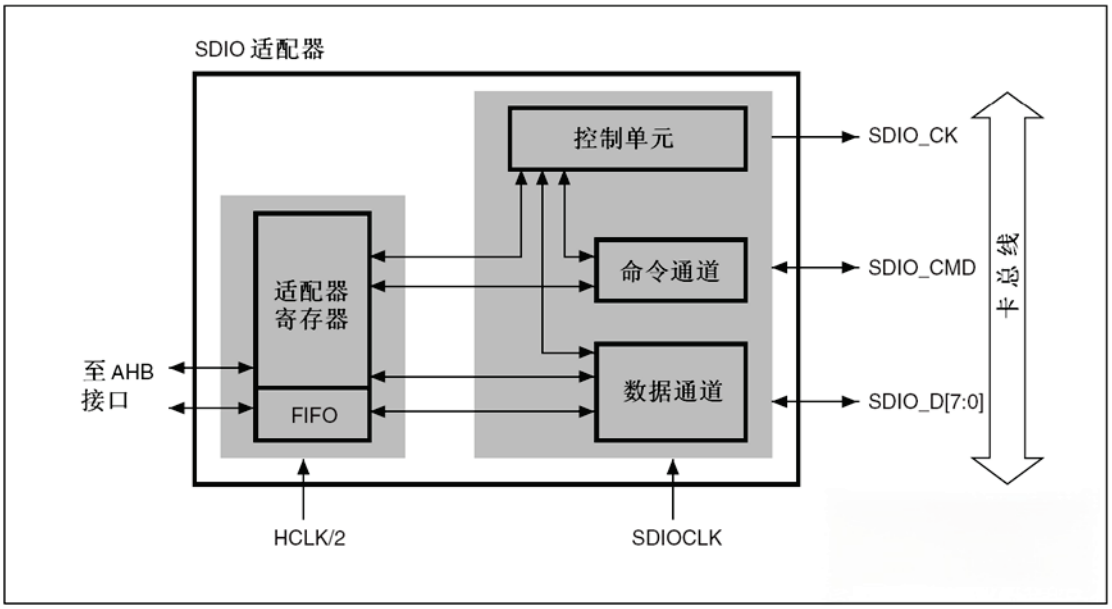


图 187 SDIO 适配器

SDIO 适配器是多媒体/加密数字存储卡总线的主设备(主机)，用于连接一组多媒体卡或加密数字存储卡，它包含以下 5 个部分：

- 适配器寄存器模块
- 控制单元
- 命令通道

- 数据通道
- 数据 FIFO

注：　适配器寄存器和 FIFO 使用 AHB 总线一侧的时钟(HCLK/2)，控制单元、命令通道和数据通道使用 SDIO 适配器一侧的时钟(SDIOCLK)。

适配器寄存器模块

适配器寄存器模块包含所有系统寄存器。该模块还产生清除多媒体卡中静态标记的信号，当在 SDIO 清除寄存器中的相应位写'1'时会产生清除信号。

控制单元

控制单元包含电源管理功能和为存储器卡提供的时钟分频。共有三种电源阶段：

- 电源关闭
- 电源启动
- 电源开

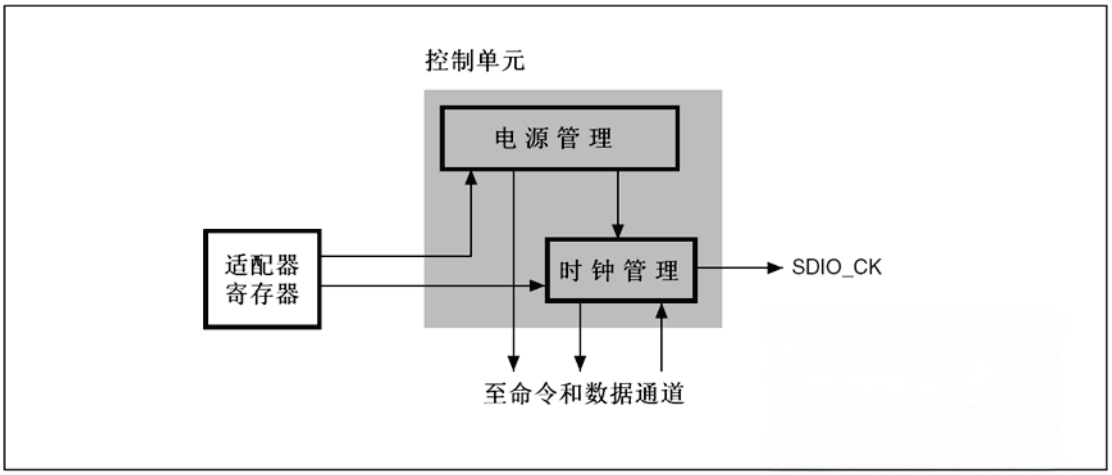


图 188 控制单元

上图为控制单元的框图，有电源管理和时钟管理子单元。

在电源关闭和电源启动阶段，电源管理子单元会关闭卡总线上的输出信号。

时钟管理子单元产生和控制 SDIO_CK 信号。SDIO_CK 输出可以使用时钟分频或时钟旁路模式。下述情况下没有时钟输出：

- 复位后
- 在电源关闭和电源启动阶段
- 当启动了省电模式并且卡总线处于空闲状态(命令通道和数据通道子单元进入空闲阶段后的 8 个时钟周期)

命令通道

命令通道单元向卡发送命令并从卡接收响应。

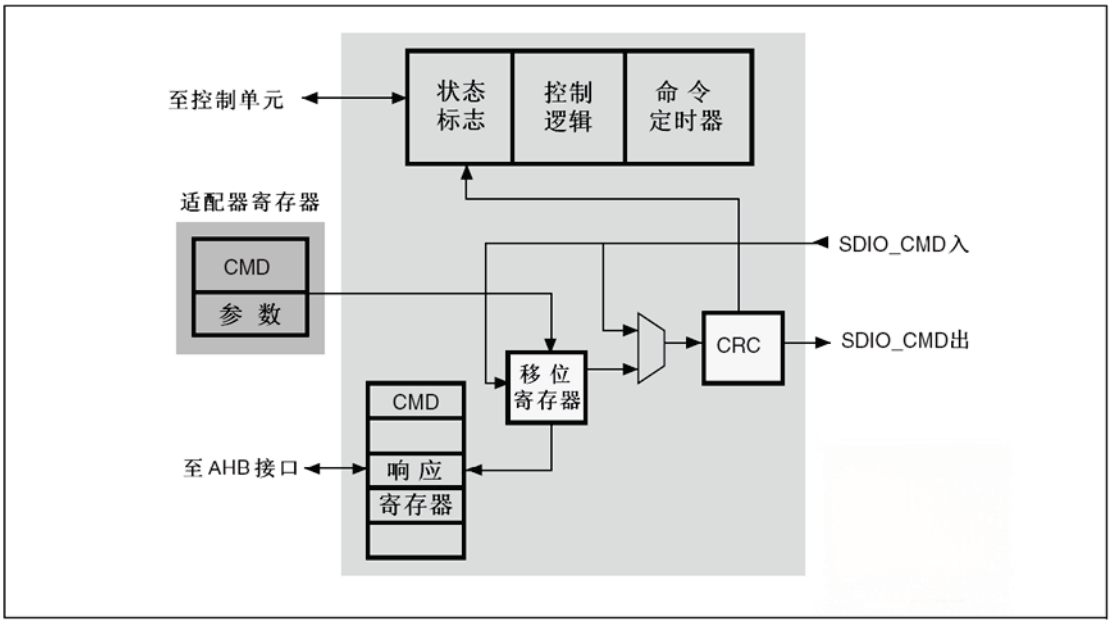


图 189 SDIO 适配器命令通道

- 命令通道状态机(CPSM)
  - 当写入命令寄存器并设置了使能位，开始发送命令。命令发送完成时，命令通道状态机(CPSM)设置状态标志并在不需要响应时进入空闲状态(见下图)。当收到响应后，接收到的 CRC 码将会与内部产生的 CRC 码比较，然后设置相应的状态标志。

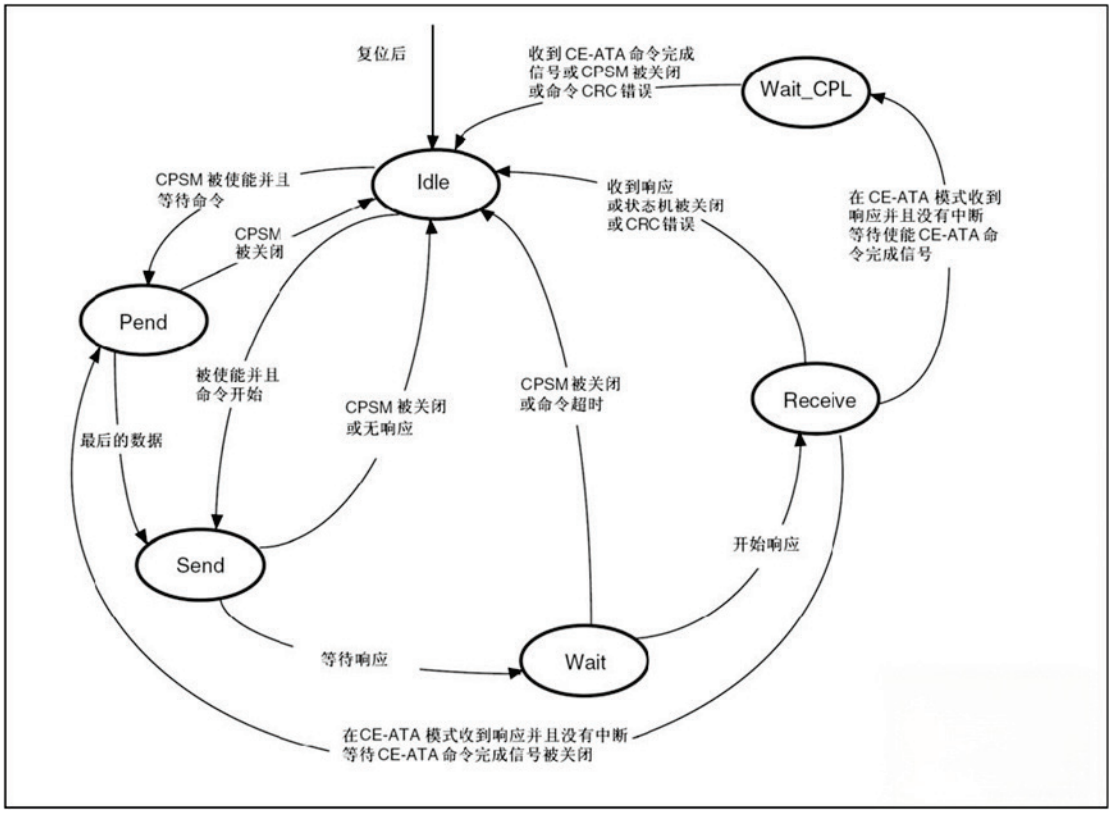


图 190 命令通道状态机(CPSM)

当进入等待(Wait)状态时，命令定时器开始运行；当 CPSM 进入接收(Receive)状态之前，产生了超时，则设置超时标志并进入空闲(Idle)状态。

注：　命令超时固定为 64 个 SDIO_CK 时钟周期。

如果在命令寄存器设置了中断位，则关闭定时器，CPSM 等待某一个卡发出的中断请求。如果命令寄存器中设置挂起位，CPSM 进入挂起(Pend)状态并等待数据通道子单元发出的 CmdPend 信号，在检测到 CmdPend 信号时，CPSM 进入发送(Send)状态，这将触发数据计数器发送停止命令的功能。

注：　CPSM 保持在空闲状态至少 8 个 SDIO_CK 周期，以满足 NCC 和 NRC 时序限制。NCC 是两个主机命令间的最小间隔；NRC 是主机命令与卡响应之间的最小间隔。

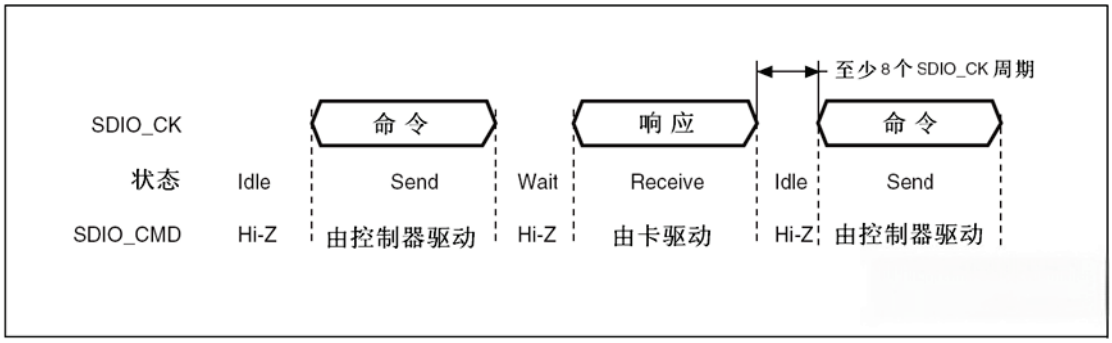


图 191 SDIO 命令传输

- 命令格式
  - 命令：命令是用于开始一项操作。主机向一个指定的卡或所有的卡发出带地址的命令或广播命令(广播命令只适合于 MMCV3.31 或之前的版本)。命令在 CMD 线上串行传送。所有命令的长度固定为 48 位，下表给出了多媒体卡、SD 存储卡和 SDIO 卡上一般的命令格式。CE-ATA 命令是 MMCV4.2 命令的扩充，所以具有相同的格式。

命令通道操作于半双工模式，这样命令和响应可以分别发送和接收。如果 CPSM 不处在发送状态，SDIO_CMD 输出处于高阻状态，如图 191 所示。SDIO_CMD 上的数据与 SDIO_CK 的上升沿同步。

表 90 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

  - 响应：响应是由一个被指定地址的卡发送到主机，对于 MMCV3.31 或以前版本所有的卡同时发送响应；响应是对先前接收到命令的一个应答。响应在 CMD 线上串行传送。

SDIO 支持 2 种响应类型，2 种类型都有 CRC 错误检测：

- 48 位短响应
- 136 位长响应

注：　如果响应不包含 CRC(如 CMD1 的响应)，设备驱动应该忽略 CRC 失败状态。

表 91 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7(或 1111111)
0	1	1	结束位

表 92 长响应格式

位	宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	111111	保留
[127:1]	127	-	CID 或 CSD(包含内部 CRC7)
0	1	1	结束位

命令寄存器包含命令索引(发至卡的 6 位)和命令类型；命令本身决定了是否需要响应和响应的类型、48 位还是 136 位(见 20.9.4 节)。命令通道中的状态标志示于下表：

表 93 命令通道状态标志

标志	说明
CMDREND	响应的 CRC 正确
CCRCFAIL	响应的 CRC 错误
CMDSENT	命令(不需要响应的命令)已经送出
CTIMEOUT	响应超时
CMDACT	正在发送命令

CRC 发生器计算 CRC 码之前所有位的 CRC 校验和，包括开始位、发送位、命令索引和命令参数(或卡状态)。对于长响应格式，CRC 校验和计算的是 CID 或 CSD 的前 120 位；

注意： *长响应格式中的开始位、传输位和 6 个保留位不参与 CRC 计算。*

CRC 校验和是一个 7 位的数值：

CRC[6:0]=余数[(M(x)*x7)/G(x)]

G(x)=x7+x3+1

M(x)=(开始位)*x39+...+(CRC 前的最后一位)*x0,或

M(x)=(开始位)*x119+...+(CRC 前的最后一位)*x0,或

数据通道

数据通道子单元在主机与卡之间传输数据。下图是数据通道的框图。

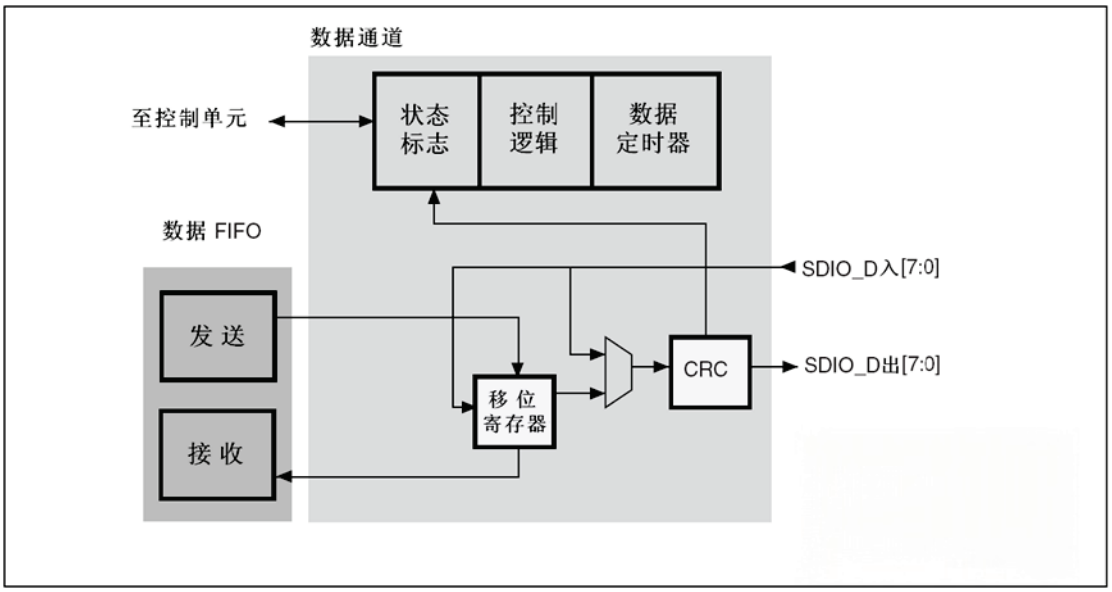


图 192 数据通道

在时钟控制寄存器中可以配置卡的数据总线宽度。如果选择了 4 位总线模式，则每个时钟周期四条数据信号线(SDIO_D[3:0])上将传输 4 位数据；如果选择了 8 位总线模式，则每个时钟周期八条数据信号线(SDIO_D[7:0])上将传输 8 位数据；如果没有选择宽总线模式，则每个时钟周期只在 SDIO_D0 上传输 1 位数据。

根据传输的方向(发送或接收)，使能时数据通道状态机(DPSM)将进入 Wait_S 或 Wait_R 状态：

- 发送：DPSM 进入 Wait_S 状态。如果发送 FIFO 中有数据，则 DPSM 进入发送状态，同时数据通道子单元开始向卡发送数据。
- 接收：DPSM 进入 Wait_R 状态并等待开始位；当收到开始位时，DPSM 进入接收状态，同时数据通道子单元开始从卡接收数据。

数据通道状态机(DPSM)

DPSM 工作在 SDIO_CK 频率，卡总线信号与 SDIO_CK 的上升沿同步。DPSM 有 6 个状态，如下图所示：

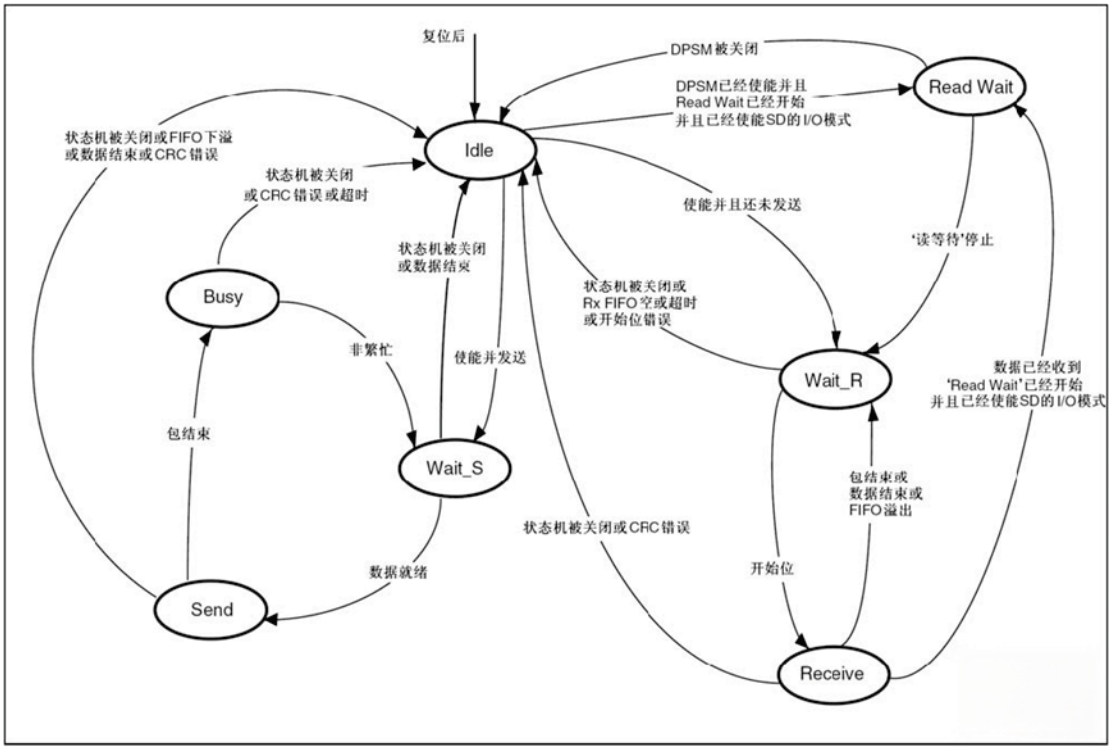


图 193 数据通道状态机(DPSM)

- 空闲(Idle)：数据通道不工作，SDIO_D[7:0]输出处于高阻状态。当写入数据控制寄存器并设置使能位时，DPSM 为数据计数器加载新的数值，并依据数据方向位进入 Wait_S 或 Wait_R 状态。
- Wait_R：如果数据计数器等于 0，当接收 FIFO 为空时 DPSM 进入到空闲(Idle)状态。如果数据计数器不等于 0，DPSM 等待 SDIO_D 上的开始位。如果 DPSM 在超时之前接收到一个开始位，它会进入接收(Receive)状态并加载数据块计数器。如果 DPSM 在检测到一个开始位前出现超时，或发生开始位错误，DPSM 将进入空闲状态并设置超时状态标志。
- 接收(Receive)：接收到的串行数据被组合为字节并写入数据 FIFO。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 等待接收 CRC 码，如果接收到的代码与内部产生的 CRC 码匹配，则 DPSM 进入 Wait_R 状态，否则设置 CRC 失败状态标志同时 DPSM 进入到空闲状态。
  - 在流模式下，当数据计数器不为 0 时，DPSM 接收数据；当计数器为 0 时，将移位寄存器中的剩余数据写入数据 FIFO，同时 DPSM 进入 Wait_R 状态。如果产生了 FIFO 上溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。
- Wait_S：如果数据计数器为 0，DPSM 进入空闲状态；否则 DPSM 等待数据 FIFO 空标志消失后，进入发送状态。

注： *DPSM 会在 Wait_S 状态保持至少 2 个时钟周期，以满足 NWR 的时序要求，NWR 是接收到卡的响应至主机开始数据传输的间隔。*

- 发送(Send)：DPSM 开始发送数据到卡设备。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：

- 在块模式下，当数据块计数器达到 0 时，DPSM 发送内部产生的 CRC 码，然后是结束位，并进入繁忙状态。
- 在流模式下，当使能位为高同时数据计数器不为 0 时，DPSM 向卡设备发送数据，然后进入空闲状态。

如果产生了 FIFO 下溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。

- 繁忙(Busy)：DPSM 等待 CRC 状态标志：
  - 如果没有接收到正确的 CRC 状态，则 DPSM 进入空闲状态并设置 CRC 失败状态标志。
  - 如果接收到正确的 CRC 状态，则当 SDIO_D0 不为低时(卡不繁忙)DPSM 进入 Wait_S 状态。

当 DPSM 处于繁忙状态时发生了超时，DPSM 则设置数据超时标志并进入空闲状态。

当 DPSM 处于 Wait_R 或繁忙状态时，数据定时器被使能，并能够产生数据超时错误：

- 发送数据时，如果 DPSM 处于繁忙状态超过程序设置的超时间隔，则产生超时。
- 接收数据时，如果未收完所有数据，并且 DPSM 处于 Wait_R 状态超过程序设置的超时间隔，则产生超时。

- 数据：数据可以从主机传送到卡，也可以反向传输。数据在数据线上传输。数据存储在一个 32 字的 FIFO 中，每个字为 32 位宽。

表 94 数据令牌格式

说明	开始位	数据	CRC16	结束位
块数据	0	-	有	1
流数据	0	-	无	1

数据 FIFO

数据 FIFO(先进先出)子单元是一个具有发送和接收单元的数据缓冲区。

FIFO 包含一个每字 32 位宽、共 32 个字的数据缓冲区，和发送与接收电路。因为数据 FIFO 工作在 AHB 时钟区域(HCLK/2)，所有与 SDIO 时钟区域(SDIOCLK)连接的信号都进行了重新同步。

依据 TXACT 和 RXACT 标志，可以关闭 FIFO、使能发送或使能接收。TXACT 和 RXACT 由数据通道子单元设置而且是互斥的：

- 当 TXACT 有效时，发送 FIFO 代表发送电路和数据缓冲区
- 当 RXACT 有效时，接收 FIFO 代表接收电路和数据缓冲区

- 发送 FIFO：当使能了 SDIO 的发送功能，数据可以通过 AHB 接口写入发送 FIFO。

发送 FIFO 有 32 个连续的地址。发送 FIFO 中有一个数据输出寄存器，包含读指针指向的数据字。当数据通道子单元装填了移位寄存器后，它移动读指针至下个数据并传输出数据。

如果未使能发送 FIFO，所有的状态标志均处于无效状态。当发送数据时，数据通道子单元设置 TXACT 为有效。

表 95 发送 FIFO 状态标志

标志	说明
TXFIFOF	当所有 32 个发送 FIFO 字都有有效的数据时，该标志为高。
TXFIFOE	当所有 32 个发送 FIFO 字都没有有效的数据时，该标志为高。
TXFIFOHE	当 8 个或更多发送 FIFO 字为空时，该标志为高。该标志可以作为 DMA 请求。
TXDAVL	当发送 FIFO 包含有效数据时，该标志为高。该标志的意思刚好与 TXFIFOE 相反。
TXUNDERR	当发生下溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

- 接收 FIFO：当数据通道子单元接收到一个数据字，它会把数据写入 FIFO，写操作结束后，写指针自动加一；在另一端，有一个读指针始终指向 FIFO 中的当前数据。如果关闭了接收 FIFO，

所有的状态标志会被清除，读写指针也被复位。在接收到数据时数据通道子单元设置 RXACT。下表列出了接收 FIFO 的状态标志。通过 32 个连续的地址可以访问接收 FIFO。

表 96 接收 FIFO 状态标志

标志	说明
RXFIFOF	当所有 32 个接收 FIFO 字都有有效的数据时，该标志为高。
RXFIFOE	当所有 32 个接收 FIFO 字都没有有效的数据时，该标志为高。
RXFIFOHF	当 8 个或更多接收 FIFO 字有有效的数据时，该标志为高。该标志可以作为 DMA 请求。
RXDAVL	当接收 FIFO 包含有效数据时，该标志为高。该标志的意思刚好与 RTXFIFOE 相反。
RXOVERR	当发生上溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

20.3.2 SDIOAHB 接口

AHB 接口产生中断和 DMA 请求，并访问 SDIO 接口寄存器和数据 FIFO。它包含一个数据通道、寄存器译码器和中断/DMA 控制逻辑。

SDIO 中断

当至少有一个选中的状态标志为高时，中断控制逻辑产生中断请求。有一个屏蔽寄存器用于选择可以产生中断的条件，如果设置了相应的屏蔽标志，则对应的状态标志可以产生中断。

SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程

在下面的例子中，主机控制器使用 CMD24(WRITE_BLOCK)从主机传送 512 字节到 MMC 卡，DMA 控制器用于从存储器向 SDIO 的 FIFO 填充数据。

1. 执行卡识别过程
2. 提高 SDIO_CK 频率
3. 发送 CMD7 命令选择卡
4. 按下述步骤配置 DMA2:
   a) 使能 DMA2 控制器并清除所有的中断标志位
   b) 设置 DMA2 通道 4 的源地址寄存器为存储器缓冲区的基地址，DMA2 通道 4 的目标地址寄存器为 SDIO_FIFO 寄存器的地址
   c) 设置 DMA2 通道 4 控制寄存器(存储器递增，非外设递增，外设和源的数据宽度为字宽度)
   d) 使能 DMA2 通道 4
5. 发送 CMD24(WRITE_BLOCK)，操作如下:
   a) 设置 SDIO 数据长度寄存器(SDIO 数据时钟寄存器应该在执行卡识别过程之前设置好)
   b) 设置 SDIO 参数寄存器为卡中需要传送数据的地址
   c) 设置 SDIO 命令寄存器：CmdIndex 置为 24(WRITE_BLOCK)；WaitRest 置为 1(SDIO 卡主机等待响应)；CPSMEN 置为 1(使能 SDIO 卡主机发送命令)，保持其它域为他们的复位值。
   d) 等待 SDIO_STA[6]=CMDREND 中断，然后设置 SDIO 数据寄存器：DTEN 置为 1(使能 SDIO 卡主机发送数据)；DTDIR 置为 0(控制器至卡方向)；DTMODE 置为 0(块数据传送)；DMAEN 置为 1(使能 DMA)；DBLOCKSIZE 置为 9(512 字节)；其它域不用设置。
   e) 等待 SDIO_STA[10]=DBCKEND
6. 查询 DMA 通道的使能状态寄存器，确认没有通道仍处于使能状态。

20.4 卡功能描述

20.4.1 卡识别模式

在卡识别模式，主机复位所有的卡、检测操作电压范围、识别卡并为总线上每个卡设置相对地址(RCA)。在卡识别模式下，所有数据通信只使用命令信号线(CMD)。

20.4.2 卡复位

GO_IDLE_STATE 命令(CMD0)是一个软件复位命令，它把多媒体卡和 SD 存储器置于空闲状态。IO_RW_DIRECT 命令(CMD52)复位 SDI/O 卡。上电后或执行 CMD0 后，所有卡的输出端都处于高阻状态，同时所有卡都被初始化至一个默认的相对卡地址(RCA=0x0001)和默认的驱动器寄存器设置(最低的速度，最大的电流驱动能力)。

20.4.3 操作电压范围确认

所有的卡都可以使用任何规定范围内的电压与 SDIO 卡主机通信，可支持的最小和最大电压 VDD 数值由卡上的操作条件寄存器(OCR)定义。

内部存储器存储了卡识别号(CID)和卡特定数据(CSD)的卡，仅能在数据传输 VDD 条件下传送这些信息。当 SDIO 卡主机模块与卡的 VDD 范围不一致时，卡将不能完成识别周期，也不能发送 CSD 数据；因此，在 VDD 范围不匹配时，SDIO 卡主机可以用下面几个特殊命令去识别和拒绝卡：SEND_OP_COND(CMD1)、SD_APP_OP_COND(SD 存储卡的 ACMD41)和 IO_SEND_OP_COND(SDI/O 卡的 CMD5)。SDIO 卡主机在执行这几个命令时会产生需要的 VDD 电压。不能在指定的电压范围进行数据传输的卡，将从总线断开并进入非激活状态。

使用这些不包含电压范围作为操作数的命令，SDIO 卡主机能够查询每个卡并在确定公共的电压范围前，把不在此范围内的卡置于非激活状态。当 SDIO 卡主机能够选择公共的电压范围或用户需要知道卡是否能用时，SDIO 卡主机可以进行这样的查询。

20.4.4 卡识别过程

多媒体卡和 SD 卡的卡识别过程是有区别的；对于多媒体卡，卡识别过程以时钟频率 Fod 开始，所有 SDIO_CMD 输出为开路驱动，允许在这个过程中的卡的并行连接，识别过程如下：

1. 总线被激活
2. SDIO 卡主机广播发送 SEND_OP_COND(CMD1)命令，并接收操作条件
3. 得到的响应是所有卡的操作条件寄存器内容的“线与”
4. 不兼容的卡会被置于非激活状态
5. SDIO 卡主机广播发送 ALL_SEND_CID(CMD2)至所有激活的卡
6. 所有激活的卡同时串行地发送他们的 CID 号，那些检测到输出的 CID 位与命令线上的数据不相符的卡必须停止发送，并等待下一个识别周期。最终只有一个卡能够成功地传送完整的 CID 至 SDIO 卡主机并进入识别状态。
7. SDIO 卡主机发送 SET_RELATIVE_ADDR(CMD3)命令至这个卡，这个新的地址被称为相对卡地址(RCA)，它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态，并不再响应新的识别过程，同时它的输出驱动从开路转变为推挽模式。
8. SDIO 卡主机重复上述步骤 5 至 7，直到收到超时条件。

对于 SD 卡而言，卡识别过程以时钟频率 Fod 开始，所有 SDIO_CMD 输出为推挽驱动而不是开路驱动，识别过程如下：

1. 总线被激活
2. SDIO 卡主机广播发送 SEND_APP_OP_COND(ACMD41)命令
3. 得到的响应是所有卡的操作条件寄存器的内容
4. 不兼容的卡会被置于非激活状态
5. SDIO 卡主机广播发送 ALL_SEND_CID(CMD2)至所有激活的卡
6. 所有激活的卡发送回他们唯一卡识别号(CID)并进入识别状态。
7. SDIO 卡主机发送 SET_RELATIVE_ADDR(CMD3)命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址(RCA)，它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态。SDIO 卡主机可以再次发送该命令更改 RCA，卡的 RCA 将是最后一次的赋值。
8. SDIO 卡主机对所有激活的卡重复上述步骤 5 至 7。

对于 SDI/O 卡而言，卡识别过程如下：

1. 总线被激活
2. SDIO 卡主机发送 IO_SEND_OP_COND(CMD5)命令
3. 得到的响应是卡的操作条件寄存器的内容
4. 不兼容的卡会被置于非激活状态
5. SDIO 卡主机发送 SET_RELATIVE_ADDR(CMD3)命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址(RCA)，它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态。SDIO 卡主机可以再次发送该命令更改 RCA，卡的 RCA 将是最后一次的赋值。

20.4.5 写数据块

执行写数据块命令(CMD24-27)时，主机把一个或多个数据块从主机传送到卡中，同时在每个数据块的末尾传送一个 CRC 码。一个支持写数据块命令的卡应该始终能够接收由 WRITE_BL_LEN 定义的数据块。如果 CRC 校验错误，卡通过 SDIO_D 信号线指示错误，传送的数据被丢弃而不被写入，所有后续(在多块写模式下)传送的数据块将被忽略。

如果主机传送部分数据，而累计的数据长度未与数据块对齐，当不允许块错位(未设置 CSD 的参数 WRITE_BLK_MISALIGN)，卡将在第一个错位的块之前检测到块错位错误(设置状态寄存器中的 ADDRESS_ERROR 错误位)。当主机试图写一个写保护区域时，写操作也会被中止，此时卡会设置 WP_VIOLATION 位。

设置 CID 和 CSD 寄存器不需要事先设置块长度，传送的数据也是通过 CRC 保护的。如果 CSD 或 CID 寄存器的部分是存储在 ROM 中，则这个不能更改的部分必须与接收缓冲区的对应部分相一致，如果有不一致之处，卡将报告一个错误同时不修改任何寄存器的内容。有些卡需要长的甚至不可预计的时间完成写一个数据块，在接收一个数据块并完成 CRC 检验后，卡开始写操作，如果它的写缓冲区已经满并且不能再从新的 WRITE_BLOCK 命令接受新的数据时，它会把 SDIO_D 信号线拉低。主机可以在任何时候使用 SEND_STATUS(CMD13)查询卡的状态，卡将返回当前状态。READY_FOR_DATA 状态位指示卡是否可以接受新的数据或写操作是否还在进行。主机可以使用 CMD7(选择另一个卡)不选中某个卡，而把这个卡置于断开状态，这样可以释放 SDIO_D 信号线而不中断未完成的写操作；当重新选择了一个卡，如果写操作仍然在进行并且写缓冲区仍不能使用，它会重新通过拉低 SDIO_D 信号线指示忙的状态。

20.4.6 读数据块

在读数据块模式下，数据传输的基本单元是数据块，它的大小在 CSD 中(READ_BL_LEN)定义。如果设置了 READ_BL_PARTIAL，同样可以传送较小的数据块，较小数据块是指开始和结束地址完全包含在一个物理块中，READ_BL_LEN 定义了物理块的大小。为保证数据传输的正确，每个数据块后都有一个 CRC 校验码。CMD17(READ_SINGLE_BLOCK)启动一次读数据块操作，在传输结束后卡返回到发送状态。

CMD18(READ_MULTIPLE_BLOCK)启动一次连续多个数据块的读操作。

主机可以在多数据块读操作的任何时候中止操作，而不管操作的类型。发送停止传输命令即可中止操作。

如果在多数据块读操作中(任一种类型)卡检测到错误(例如：越界、地址错位或内部错误)，它将停止数据传输并仍处于数据状态；此时主机必须发送停止传输命令中止操作。在停止传输命令的响应中报告读错误。

如果主机发送停止传输命令时，卡已经传输完一个确定数目的多个数据块操作中的最后一个数据块，因为此时卡已经不在数据状态，主机会得到一个非法命令的响应。如果主机传输部分数据块，而累计的数据长度不能与物理块对齐同时不允许块错位，卡会在出现第一个未对齐的块时检测出一个块对齐错误，并在状态寄存器中设置 ADDRESS_ERROR 错误标志。

20.4.7 数据流操作，数据流写入和数据流读出(只适用于多媒体卡)

在数据流模式，数据按字节传输，同时每个数据块后没有 CRC。

数据流写(只适用于多媒体卡)

WRITE_DAT_UNTIL_STOP(CMD20)开始从 SDIO 卡主机至卡的数据传输，从指定的地址开始连续传输直到 SDIO 卡主机发出一个停止命令。如果允许部分数据块传输(设置了 CSD 参数 WRITE_BL_PARTIAL)，则数据流可以在卡的地址空间中的任意地址开始和停止，否则数据流只能在数据块的边界开始和停止。因为传输的数据数目没有事先设定，不能使用 CRC 校验。如果发送数据时达到了存储器的最大地址，即使 SDIO 卡主机没有发送停止命令，随后传输的数据也会被丢弃。

数据流写操作的最大时钟频率可以通过下式计算

$$Maximumspeed = Min(TRANSPEED,\ \frac{(8 * 2^{writebllen})(-NSAC)}{TAAC * R2WFACTOR})$$

- Maximumspeed=最大写频率
- TRANSPEED=最大数据传输率
- writebllen=最大写数据块长度
- NSAC=以 CLK 周期计算的数据读操作时间 2
- TAAC=数据读操作时间 1
- R2WFACTOR=写速度因子

如果主机试图使用更高的频率，卡可能不能处理数据并停止编程，同时在状态寄存器中设置 OVERRUN 错误位，丢弃所有随后传输的数据并(在接收数据状态)等待停止命令。如果主机试图写入一个写保护区域，写操作将被中止，同时卡将设置 WP_VIOLATION 位。

数据流读(只适用于多媒体卡)

READ_DAT_UNTIL_STOP(CMD11)控制数据流数据传输。

这个命令要求卡从指定的地址读出数据，直到 SDIO 卡主机发送 STOP_TRANSMISSION(CMD12)。因为串行命令传输的延迟，停止命令的执行会有延迟，数据传送会在停止命令的结束位后停止。如果发送数据时达到了存储器的最大地址，SDIO 卡主机没有发送停止命令，随后传输的数据将是无效数据。

数据流读操作的最大时钟频率可以通过下式计算

$$\text{Maximumspeed} = \text{Min}(\text{TRANSPEED}, \frac{\text{n}(8 * 2^{\text{writebllen}})(-\text{NSAC})}{\text{TAAC} * \text{R2WFACTOR}})$$

- Maximumspeed=最大写频率
- TRANSPEED=最大数据传输率
- readbllen=最大读数据块长度
- NSAC=以 CLK 周期计算的数据读操作时间 2
- TAAC=数据读操作时间 1
- R2WFACTOR=写速度因子

如果主机试图使用更高的频率，卡将不能处理数据传输，此时卡在状态寄存器中设置 UNDERRUN 错误位，中止数据传输并在数据状态等待停止命令。

20.4.8 擦除：成组擦除和扇区擦除

多媒体卡的擦除单位是擦除组，擦除组是以写数据块计算，写数据块是卡的基本写入单位。擦除组的大小是卡的特定参数，在 CSD 中定义。

主机可以擦除一个连续范围的擦除组，开始擦除操作有三个步骤。

首先，主机使用 ERASE_GROUP_START(CMD35)命令定义连续范围的开始地址，然后使用 ERASE_GROUP_END(CMD36)命令定义连续范围的结束地址，最后发送擦除命令 ERASE(CMD38)开始擦除操作。擦除命令的地址域是以字节为单位的擦除组地址。卡会舍弃未与擦除组大小对齐的部分，把地址边界对齐到擦除组的边界。

如果未按照上述步骤收到了擦除命令，卡在状态寄存器中设置 ERASE_SEQ_ERROR 位，并重新等待第一个步骤。

如果收到了除 SEND_STATUS 和擦除命令之外的其它命令，卡在状态寄存器中设置 ERASE_RESET 位，解除擦除序列并执行新的命令。

如果擦除范围包含了写保护数据块，这些块不被擦除，只有未保护的块被擦除，同时卡在状态寄存器中设置 WP_ERASE_SKIP 状态位。

在擦除过程中，卡拉低 SDIO_D 信号。实际的擦除时间可能很长，主机可以使用 CMD7 解除卡的选择。

20.4.9 宽总线选择和解除选择

可以通过 SET_BUS_WIDTH(ACMD6)命令选择或不选择宽总线(4 位总线宽度)操作模式，上电后或 GO_IDLE_STATE(CMD0)命令后默认的总线宽度为 1 位。SET_BUS_WIDTH(ACMD6)命令仅在传输状态时有效，即只有在使用 SELECT/DESELECT_CARD(CMD7)命令选择了卡后才能改变总线宽度。

20.4.10 保护管理

SDIO 卡主机模块支持三种保护方式：

1. 内部卡保护(卡内管理)
2. 机械写保护开关(仅由 SDIO 卡主机模块管理)
3. 密码管理的卡锁操作

内部卡的写保护

卡的数据可以被保护不被覆盖或擦除。在 CSD 中永久地或临时地设置写保护位，生产厂商或内容提供商可以永久地对整个卡施行写保护。对于支持在 CSD 中设置 WP_GRP_ENABLE 位从而提供一组扇区写保护的卡，部分数据可以被保护，写保护可以通过程序改变。写保护的基本单位是 CSD 参数 WP_GRP_SIZE 个扇区。SET_WRITE_PROT 和 CLR_WRITE_PROT 命令控制指定组的保护，SEND_WRITE_PROT 命令与单数据块读命令类似，卡送出一个包含 32 个写保护位(代表从指定地址开始的 32 个写保护组)的数据块，跟着一个 16 位的 CRC 码。写保护命令的地址域是一个以字节为单位的组地址。

卡将截断所有组大小以下的地址。

机械写保护开关

在卡的侧面有一个机械的滑动开关，允许用户设置或清除卡的写保护。当滑动开关置于小窗口打开的位置时，卡处于写保护状态，当滑动开关置于小窗口关闭的位置时，可以更改卡中内容。在卡的插槽上的对应部位也有一个开关指示 SDIO 卡主机模块，卡是否处于写保护状态。卡的内部电路不知道写保护开关的位置。

密码保护

密码保护功能允许 SDIO 卡主机模块使用密码对卡实行上锁或解锁。密码存储在 128 位的 PWD 寄存器中，它的长度设置在 8 位的 PWD_LEN 寄存器中。这些寄存器是不可挥发的，即掉电后它们的内容不丢失。已上锁的卡能够响应和执行相应的命令，即允许 SDIO 卡主机模块执行复位、初始化和查询状态等操作，但不允许操作卡中的数据。当设置了密码后(即 PWD_LEN 的数值不为 0)，上电后卡自动处于上锁状态。正如 CSD 和 CID 寄存器写命令，上锁/解锁命令仅在传输状态下有效，在这个状态下，命令中没有地址参数，但卡已经被选中。卡的上锁/解锁命令具有单数据块写命令的结构和总线操作类型，传输的数据块包含所有命令所需要的信息(密码设置模式、PWD 内容和上锁/解锁指示)。在发送卡的上锁/解锁命令之前，命令数据块的长度由 SDIO 卡主机模块定义，，命令结构示于表 110。

位的设置如下：

- ERASE：设置该位将执行强制擦除，所有其它位必须为 0，只发送命令字节。
- LOCK_UNLOCK：设置该位锁住卡，LOCK_UNLOCK 与 SET_PWD 可以同时设置，但不能与 CLR_PWD 同时设置。
- CLR_PWD：设置该位清除密码数据。
- SET_PWD：设置该位将密码数据保存至存储器。
- PWD_LEN：以字节为单位定义密码的长度。
- PWD：密码(依不同的命令，新的密码或正在使用的密码)

以下几节列出了设置/清除密码、上锁/解锁和强制擦除的命令序列。

设置密码

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)。
2. 定义要在 8 位的卡上锁/解锁模式下发送的数据块长度(SET_BLOCKLEN，CMD16)，8 位的 PWD_LEN，新密码的字节数目。当更换了密码后，发送命令的数据块长度必须同时考虑新旧密码的长度。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK(CMD42)命令，并包含 16 位的 CRC 码。数据块包含了操作模式(SET_PWD=1)、长度(PWD_LEN)和密码(PWD)。当更换了密码后，长度数值(PWD_LEN)包含了新旧两个密码的长度，PWD 域包含了旧的密码(正在使用的)和新的密码。
4. 当旧的密码匹配后，新的密码和它的长度被分别存储在 PWD 和 PWD_LEN 域。如果送出的旧密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时密码不变。

密码长度域(PWD_LEN)指示当前是否设置了密码，如果该域为非零，则表示使用了密码，卡在上电时自动上锁。在不断电的情况下，如果设置了密码，可以通过设置 LOCK_UNLOCK 位或发送一个额外的上锁命令，立即锁住卡。

清除密码

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)。
2. 定义要在 8 位的卡上锁/解锁模式下发送的数据块长度(SET_BLOCKLEN，CMD16)，8 位的 PWD_LEN，当前使用密码的字节数目。
3. 当密码匹配后，PWD 域被清除同时 PWD_LEN 被设为 0。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时密码不变。

卡上锁

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 定义要在 8 位的卡上锁/解锁模式(见表 110 的字节 0)下发送的数据块长度(SET_BLOCKLEN，CMD16)，8 位的 PWD_LEN，和当前密码的字节数目。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK(CMD42)命令，并包含 16 位的 CRC 码。数据块包含了操作模式(LOCK_UNLOCK=1)、长度(PWD_LEN)和密码(PWD)。
4. 当密码匹配后，卡被上锁并则设置状态寄存器中的 CARD_IS_LOCKED 状态位。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时上锁操作失败。

设置密码和为卡上锁可以在同一个操作序列中进行，此时 SDIO 卡主机模块按照前述的步骤设置密码，但在发送新密码命令的第 3 步需要设置 LOCK_UNLOCK 位。

如果曾经设置过密码(PWD_LEN 不为 0)，卡会在上电复位时自动地上锁。对已经上锁的卡执行上锁操作或对没有密码的卡执行上锁操作会导致失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

卡解锁

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 定义要在 8 位的卡上锁/解锁模式(见表 110 的字节 0)下发送的数据块长度(SET_BLOCKLEN，CMD16)，8 位的 PWD_LEN，和当前密码的字节数目。

3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK(CMD42)命令，并包含 16 位的 CRC 码。数据块包含了操作模式(LOCK_UNLOCK=0)、长度(PWD_LEN)和密码(PWD)。
4. 当密码匹配后，卡锁被解除，同时状态寄存器中的 CARD_IS_LOCKED 位被清除。如果送出的密码与期望的密码(长度或内容)不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时卡仍保持上锁状态。

解锁状态只在当前的供电过程中有效，只要不清除 PWD 域，下次上电后卡会被自动上锁。试图对已经解了锁的卡执行解锁操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

强制擦除

如果用户忘记了密码(PWD 的内容)，可以在清除卡中的所有内容后使用卡。强制擦除操作擦除所有卡中的数据和密码。

1. 选择一个卡(SELECT/DESELECT_CARD，CMD7)
2. 设置发送的数据块长度(SET_BLOCKLEN，CMD16)为 1，仅发送 8 位的卡上锁/解锁字节(见表 110 的字节 0)。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK(CMD42)命令，并包含 16 位的 CRC 码。数据块包含了操作模式(ERASE=1)所有其它位为 0。
4. 当 ERASE 位是数据域中仅有的位时，卡中的所有内容将被擦除，包括 PWD 和 PWD_LEN 域，同时卡不再被上锁。如果有任何其它位不为 0，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，卡中的数据保持不变，同时卡仍保持上锁状态。

试图对已经解了锁的卡执行擦除操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

20.4.11 卡状态寄存器

响应格式 R1 包含了一个 32 位的卡状态域，这个域是用于向卡主机发送卡的状态信息(这些信息有可能存在本地的状态寄存器中)。除非特别说明，卡返回的状态始终是与之前的命令相关的。

表 97 定义了不同的状态信息。表中有关类型和清除条件域的缩写定义如下：类型：

- E:错误位
- S:状态位
- R:检测位，并依据实际的命令响应而设置
- X:检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A:依据卡的当前状态
- B:始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C:读即可清除

表 97 卡状态

位	名称	类型	数值	说明	清除条件
31	ADDRESS_OUT_OF_RANGE	ERX	'0'=无错误 '1'=错误	命令中的地址参数超出了卡的允许范围。一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写超出卡的容量的部分。	C

30	ADDRESS_MISALIGN		‘0’=无错误 ‘1’=错误	命令中的地址参数(与当前的数据块长度对照)定义的第一个数据块未与卡的物理块对齐。 一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写未与物理块对齐的数据块。	C
29	BLOCK_LEN_ERROR		‘0’=无错误 ‘1’=错误	SET_BLOCKLEN 命令的参数超出了卡的最大允许范围，或先前定义的数据块长度对于当前命令来说是非法的(例如：主机发出一个写命令，当前的块长度小于卡所允许的最小长度，同时又不允许写入部分数据块)。	C
28	ERASE_SEQ_ERROR		‘0’=无错误 ‘1’=错误	发送擦除命令的顺序错误。	C
27	ERASE_PARAM	EX	‘0’=无错误 ‘1’=错误	擦除时选择了非法的擦除组。	C
26	WP_VIOLATION	EX	‘0’=无错误 ‘1’=错误	试图对一个写保护的数据块编程。	C
25	CARD_IS_LOCKED	SR	‘0’=卡未锁 ‘1’=卡已锁	当设置了该位，表示卡已经被锁住。	A
24	LOCK_UNLOCK_FAILED	EX	‘0’=无错误 ‘1’=错误	在上锁/解锁中有命令的顺序错误或检测到密码错误。	C
23	COM_CRC_ERROR	ER	‘0’=无错误 ‘1’=错误	之前的命令中 CRC 校验错误。	B
22	ILLEGAL_COMMAND	ER	‘0’=无错误 ‘1’=错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	EX	‘0’=成功 ‘1’=失败	卡的内部实施了 ECC 校验，但在更正数据时失败。	C
20	CC_ERROR	ER	‘0’=无错误 ‘1’=错误	(标准中未定义)卡内部发生错误，与主机的命令无关。	C
19	ERROR	EX	‘0’=无错误 ‘1’=错误	产生了与执行上一个主机命令相关的(标准中未定义)卡内部的错误(例如：读或写错误)。	C
18	保留				
17	保留				
16	CID/CSD_OVERWRITE	EX	‘0’=无错误 ‘1’=错误	可以是任何一个下述的错误： n 已经写入了 CID 寄存器，不能覆盖 nCSD 的只读部分与卡的内容不匹配 n 试图进行拷贝或永久写保护的反向操作，即恢复原状或解除写保护。	C
15	WP_ERASE_SKIP	EX	‘0’=未保护 ‘1’=已保护	遇到已经存在的写保护数据块，仅有部分地址空间被擦除。	C
14	CARD_ECC_DISABLED	SX	‘0’=允许 ‘1’=不允许	执行命令时没有使用内部的 ECC。	A
13	ERASE_RESET		‘0’=清除 ‘1’=设置	因为收到一个擦除顺序之外的命令(非 CMD35、CMD36、CMD38 或 CMD13 命令)，进入擦除过程的序列被中止。	C
12:9	CURRENT_STATE	SR	0=空闲 1=就绪 2=识别 3=待机 4=发送	当收到命令时卡内状态机的状态。如果命令的执行导致状态的变化，这个变化将会在下个命令的响应中反映出来。这四个位按十进制数 0 至 15 解释。	B

			5=数据 6=接收 7=编程 8=断开 9=忙测试 10~15=保留		
8	READY_FOR_DATA	SR	'0'=未就绪 '1'=就绪	与总线上的缓冲器空的信号相对应。	
7	SWITCH_ERROR	EX	'0'=无错误 '1'=转换错	卡没有按照 SWITCH 命令的要求转换到希望的模式。	B
6	保留				
5	APP_CMD	SR	'0'=不允许 '1'=允许	卡期望 ACMD，或指示命令已经被解释为 ACMD 命令。	C
4	保留给 SDI/O 卡				
3	AKE_SEQ_ERROR	ER	'0'=无错误 '1'=错误	验证的顺序有错误。	C
2	保留给与应用相关的命令。				
1,0	保留给生产厂家的测试模式。				

20.4.12 SD 状态寄存器

SD 状态包含与 SD 存储器卡特定功能相关的状态位和一些与未来应用相关的状态位，SD 状态的长度是一个 512 位的数据块。收到 ACMD13 命令(CMD55，然后是 CMD13)后，这个寄存器的内容被传送到 SDIO 卡主机。只有卡处于传输状态时(卡已被选择)才能发送 ACMD13 命令。

表 98 定义了不同的 SD 状态寄存器信息。表中有关类型和清除条件域的缩写定义如下：类型：

- E:错误位
- S:状态位
- R:检测位，并依据实际的命令响应而设置
- X:检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A:依据卡的当前状态
- B:始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C:读即可清除

表 98 SD 状态

位	名称	类型	数值	说明	清除条件
511:510	DAT_BUS_WIDTH	SR	'00'=1(默认) '01'=保留 '10'=4 位宽 '11'=保留	由 SET_BUS_WIDTH 命令定义的当前数据总线宽度。	A
509	SECURED_MODE	SR	'0'=未处于保密模式 '1'=处于保密模式	卡处于保密操作模式(详见 "SD 保密规范")。	A
508:496	保留				
495:480	SD_CARD_TYPE	SR	'00xxh'=在物理规范版本 1.01~2.00 的 SD 存储器卡('x'	这个域的低 8 位可以在未来定义 SD 存储卡的不同变种(每个位可以用于定义不同的 SD 类型)。高 8 位可以用于定义那些不遵守当前的 SD 物理层规范的 SD 卡。	A

			表示任意值)。已定义的卡有: '0000'=通用 SD 读写卡 '0001'=SDROM 卡		
479:448	SIZE_OF_PROTECTED_AREA	SR	受保护的区域大小(见以下说明)	(见以下说明)	A
447:440	SPEED_CLASS	SR	卡的速度类型(见以下说明)	(见以下说明)	A
439:432	PERFORMANCE_MOVE	SR	以 1MB/秒为单位的传输性能(见以下说明)	(见以下说明)	A
431:428	AU_SIZE	SR	AU 的大小(见以下说明)	(见以下说明)	A
427:424	保留				
423:408	ERASE_SIZE	SR	一次可以擦除的 AU 数目	(见以下说明)	A
407:402	ERASE_TIMEOUT	SR	擦除 UNIT_OF_ERASE_AU 指定的范围的超时数值	(见以下说明)	A
401:400	ERASE_OFFSET	SR	在擦除时增加的固定偏移数值	(见以下说明)	A
399:312	保留				
311:0	保留给生产厂商				

SIZE_OF_PROTECTED_AREA

标准容量卡和高容量卡设置该位的方式不同。对于标准容量卡，受保护区域的容量由下式计算:
受保护区域=SIZE_OF_PROTECTED_AREA*MULT*BLOCK_LENSIZE_OF_PROTECTED_AREA 的单位是 MULT*BLOCK_LEN。
对于高容量卡，受保护区域的容量由下式计算:
受保护区域=SIZE_OF_PROTECTED_AREASIZE_OF_PROTECTED_AREA 的单位是字节。

SPEED_CLASS

这 8 位指示速度的类型和可以通过计算 PW/2 的数值(PW 是写的性能)。

表 99 速度类型代码

SPEED_CLASS	数值定义
00h	类型 0
01h	类型 2
02h	类型 4
03h	类型 6
04h~FFh	保留

PERFORMANCE_MOVE

这 8 位以 1MB/秒为单位指示移动性能(Pm)。如果卡不用 RU(纪录单位)移动数据，应该认为 Pm 是无穷大。设置这个域为 FFh 表示无穷大。

表 100 移动性能代码

PERFORMANCE_MOVE	数值定义
00h	未定义

01h	1MB/秒
02h	2MB/秒
.....	.....
FEh	254MB/秒
FFh	无穷大

AU_SIZE

这 4 位指示 AU 的长度，数值是 16K 字节为单位 2 的幂次的倍数。

表 101 AU_SIZE 代码

AU_SIZE	数值定义
00h	未定义
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB
Ah~Fh	保留

依据卡的容量，最大的 AU 长度由下表定义。卡可以在 RU 长度和最大的 AU 长度之间设置任意的 AU 长度。

表 102 最大的 AU 长度

容量	16MB~64MB	128MB~256MB	512MB	1GB~32GB
最大的 AU 长度	512KB	1MB	2MB	4MB

ERASE_SIZE

这个 16 位域给出了 NERASE，当 NERASE 个 AU 被擦除时，ERASE_TIMEOUT 定义了超时时间。主机应该确定适当的一次操作中擦除的 AU 数目，这样主机可以显示擦除操作的进度。如果该域为 0，则不支持擦除的超时计算。

表 103 ERASE_SIZE 代码

ERASE_SIZE	数值定义
0000h	不支持擦除的超时计算
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
.....	.....
FFFFh	65535 个 AU

ERASE_TIMEOUT

这 6 位给出了 TERASE，当 ERASE_SIZE 指示的多个 AU 被擦除时，这个数值给出了从偏移量算起的擦除超时。ERASE_TIMEOUT 的范围可以定义到最多 63 秒，卡的生产商可以根据具体实现选择合适的 ERASE_SIZE 与 ERASE_TIMEOUT 的组合，先确定 ERASE_TIMEOUT 再确定 ERASE_SIZE。

表 104 擦除超时代码

ERASE_TIMEOUT	数值定义
00	不支持擦除的超时计算

01	1 秒
02	2 秒
03	3 秒
.....	.....
63	63 秒

ERASE_OFFSET

这 2 位给出了 TOFFSET，当 ERASE_SIZE 和 ERASE_TIMEOUT 同为 0 时这个数值没有意义。

表 105 擦除偏移代码

ERASE_OFFSET	数值定义
0h	0 秒
1h	1 秒
2h	2 秒
3h	3 秒

20.4.13 SD 的 I/O 模式

SD 的 I/O 中断

为了让 SDI/O 卡能够中断多媒体卡/SD 模块，在 SD 接口上有一个具有中断功能的引脚——第 8 脚，在 4 位 SD 模式下这个脚是 SDIO_D1，卡用它向多媒体卡/SD 模块提出中断申请。对于每一个卡或卡内的功能，中断功能是可选的。SDI/O 的中断是电平有效，即在被识别并得到多媒体卡/SD 模块的响应之前，中断信号线必须保持有效电平(低)，在中断过程结束后保持无效电平(高)。在多媒体卡/SD 模块服务了中断请求后，通过一个 I/O 写操作，写入适当的位到 SDI/O 卡的内部寄存器，即可清除中断状态位。所有 SDI/O 卡的中断输出是低电平有效，多媒体卡/SD 模块在所有数据线(SDIO/D[3:0])上提供上拉电阻。多媒体卡/SD 模块在中断阶段对第 8 脚(SDIO_D/IRQ)采样并进行中断检测，其它时间该信号线上的数值将被忽略。

存储器操作和 I/O 操作都具有中断阶段，单个数据块操作的中断阶段定义与多个数据块传输操作的中断阶段定义不同。

SD 的 I/O 暂停和恢复

在一个多功能的 SDI/O 卡或同时具有 I/O 和存储器功能的卡中，多个设备(I/O 和存储器)共用 MMC/SD 总线。为了使 MMC/SD 模块中的多个设备能够共用总线，SDI/O 卡和复合卡可以有选择地实现暂停/恢复的概念；如果一个卡支持暂停/恢复，MMC/SD 模块能够暂时地停止一个功能或存储器的数据传输操作(暂停)，借此让出总线给具有更高优先级的其它功能或存储器，在这个具有更高优先级的传输完成后，再恢复原先暂停的传输。支持暂停/恢复的操作是可选的。在 MMC/SD 总线上执行暂停/恢复操作有下述步骤：

1. 确定 SDIO_D[3:0]信号线的当前功能
2. 请求低优先级或慢的操作暂停
3. 等待暂停操作完成，确认设备已暂停
4. 开始高优先级的传输
5. 等待高优先级的传输结束
6. 恢复暂停的操作

SDI/O 读等待(Read Wait)

可选的读等待(RW)操作只适用于 SD 卡的 1 位或 4 位模式。读等待操作允许 MMC/SD 模块在一个卡正在读多个寄存器(IO_RW_EXTENDED,CMD53)时，要求它暂时停止数据传输，同时允许 MMC/SD 模块发送命令到 SDI/O 设备中的其他功能。判断一个卡是否支持读等待协议，MMC/SD 模块应该检测卡的内部寄存器。读等待的时间与中断阶段相关。

20.4.14 命令与响应

应用相关命令和通用命令

SD 卡主机模块系统是用于提供一个适用于多种应用类型的标准接口，但同时又要兼顾特定用户和应用的功能，因此标准中定义了两类通用命令：应用相关命令(ACMD)和通用命令(GEN_CMD)。

当卡收到 APP_CMD(CMD55)命令时，卡期待下一个命令是应用相关命令。应用相关命令(ACMD)具有普通多媒体卡相同的格式结构，并可以使用相同的 CMD 号码，因为它是出现在 APP_CMD(CMD55)后面，所以卡把它识别为 ACMD 命令。如果跟随 APP_CMD(CMD55)之后不是一个已经定义的应用相关命令，则认为它是一个标准命令；例如：有一个 SD_STATUS(ACMD13)应用相关命令，如果在紧随 APP_CMD(CMD55)之后收到 CMD13，它将被解释为 SD_STATUS(ACMD13)；但是如果卡在紧随 APP_CMD(CMD55)之后收到 CMD7，而这个卡没有定义 ACMD7，则它将被解释为一个标准的 CMD7(SELECT/DESELECT_CARD)命令。

如果要使用生产厂商自定义的 ACMD，SD 卡主机需要做以下操作：

1. 发送 APP_CMD(CMD55)命令
   卡送回响应给多媒体/SD 卡模块，指示设置了 APP_CMD 位并等待 ACMD 命令。
2. 发送指定的 ACMD
   卡送回响应给多媒体/SD 卡模块，指示设置了 APP_CMD 位，收到的命令已经正确地按照 ACMD 命令解析；如果发送了一个非 ACMD 命令，卡将按照普通的多媒体卡命令处理同时清除卡中状态寄存器的 APP_CMD 位。

如果发送了一个非法的命令(不管是 ACMD 还是 CMD)，将被按照标准的非法多媒体卡命令进行错误处理。

GEN_CMD 命令的总线操作过程，与单数据块读写命令(WRITE_BLOCK，CMD24 或 READ_SINGLE_BLOCK，CMD17)相同；这时命令的参数表示数据传输的方向而不是地址，数据块具有用户自定义的格式和意义。

发送 GEN_CMD(CMD56)命令之前，卡必须被选中(状态机处于传输状态)，数据块的长度由 SET_BLOCKLEN(CMD16)定义。GEN_CMD(CMD56)命令的响应是 R1b 格式。

命令类型

应用相关命令和通用命令有四种不同的类型：

1. 广播命令(BC)：发送到所有卡，没有响应返回。
2. 带响应的广播命令(BCR)：发送到所有卡，同时收到从所有卡返回的响应。
3. 带寻址(点对点)的命令(AC)：发送到选中的卡，在 SDIO_D 信号线上不包括数据传输。
4. 带寻址(点对点)的数据传输命令(AC)：发送到选中的卡，在 SDIO_D 信号线上包含数据传输。

命令格式

命令格式参见表 90。

多媒体卡/SD 卡模块的命令

表 106 基于块传输的写命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD23	ac	[31:16]=0 [15:0]=数据块数目	R1	SET_BLOCK_COUNT	定义在随后的多块读或写命令中需要传输块的数目。
CMD24	adtc	[31:0]=数据地址	R1	WRITE_BLOCK	按照 SET_BLOCKLEN 命令选择的长度写一个块。
CMD25	adtc	[31:0]=数据地址	R1	WRITE_MULTIPLE_BLOCK	收到一个 STOP_TRANSMISSION 命令或达到了指定的块数目之前，连续地写数据块。
CMD26	adtc	[31:0]=填充位	R1	PROGRAM_CID	对卡的识别寄存器编程。对于每个卡只能发送一次这个命令。卡中有硬件机制防止多次的编程操作。通常该命令保留给生产厂商。
CMD27	adtc	[31:0]=填充位	R1	PROGRAM_CSD	对卡的 CSD 中可编程的位编程。
CMD28	ac	[31:0]=数据地址	R1b	SET_WRITE_PROT	如果卡有写保护功能，该命令设置指定组的写保护位。写保护特性设置在卡的特殊数据区(WP_GRP_SIZE)。
CMD29	ac	[31:0]=数据地址	R1b	CLR_WRITE_PROT	如果卡有写保护功能，该命令清除指定组的写保护位。
CMD30	adtc	[31:0]=写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表 107 基于块传输的写保护命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD28	c	[31:0]=数据地址	R1b	SET_WRITE_PROT	如果卡具有写保护的功能，该命令设置指定组的写保护位。写保护的属性设置在卡的特定数据域(WP_GRP_SIZE)。

CMD29	c	[31:0]=数据地址	R1b	CLR_WRITE_PROT	如果卡具有写保护的功能，该命令清除指定组的写保护位。
CMD30	dtc	[31:0]=写保护数据地址	R1	SEND_WRITE_PROT	如果卡具有写保护的功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表 108 擦除命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD32 ... CMD34	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这些命令代码。				
CMD35	ac	[31:0]=数据地址	R1	ERASE_GROUP_START	在选择的擦除范围内，设置第一个擦除组的地址。
CMD36	ac	[31:0]=数据地址	R1	ERASE_GROUP_END	在选择的连续擦除范围内，设置最后一个擦除组的地址。
CMD37	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这个命令代码。				
CMD38	ac	[31:0]=填充位	R1	ERASE	擦除之前选择的数据块。

表 109 I/O 模式命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD39	ac	[31:16]=RCA [15]=寄存器写标志 [14:8]=寄存器地址[7:0]=寄存器数据	R4	FAST_IO	用于写和读 8 位(寄存器)数据域。该命令指定一个卡和寄存器，如果设置了写标志还提供写入的数据。R4 响应包含从指定寄存器读出的数据。该命令访问未在多媒体卡标准中定义的与应用相关的寄存器。
CMD40	bcr	[31:0]=填充位	R5	GO_IRQ_STATE	置系统于中断模式。
CMD41	保留				

表 110 上锁命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD42	adtc	[31:0]=填充位	R1b	LOCK_UNLOCK	设置/清除密码或对卡上锁/解锁。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD43 ... CMD54	保留。				

表 111 应用相关命令

CMD 索引	类型	参数	响应格式	缩写	说明

CMD55	ac	[31:16]=RCA[15:0]=填充位	R1	APP_CMD	指示卡下一个命令是应用相关命令而不是一个标准命令。
CMD56	adtc	[31:1]=填充位 [0]=RD/WR	-	-	在通用或应用相关命令中，或者用于向卡中传输一个数据块，或者用于从卡中读取一个数据块。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD57 ... CMD59	保留。				
CMD60 ... CMD63	保留给生产厂商。				

20.5 响应格式

所有的响应是通过 MCCMD 命令在 SDIO_CMD 信号线上传输。响应的传输总是从对应响应字的位串的最左面开始，响应字的长度与响应的类型相关。

一个响应总是有一个起始位(始终为 0)，跟随着传输的方向位(卡=0)。下表中标示为 x 的数值表示一个可变的部分。除了 R3 响应类型，所有的响应都有 CRC 保护。每一个命令码字都有一个结束位(始终为 1)。

共有 5 种响应类型，它们的格式定义如下：

20.5.1 R1(普通响应命令)

代码长度=48 位。位 45:40 指示要响应的命令索引，它的数值介于 0 至 63 之间。卡的状态由 32 位进行编码。

表 112 R1 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	X	命令索引
[39:8]	32	X	卡状态
[7:1]	7	X	CRC7
0	1	1	结束位

20.5.2 R1b

与 R1 格式相同，但可以选择在数据线上发送一个繁忙信号。收到这些命令后，依据收到命令之前的状态，卡可能变为繁忙。

20.5.3 R2(CID、CSD 寄存器)

代码长度=136 位。CID 寄存器的内容将作为 CMD2 和 CMD10 的响应发出。CSD 寄存器的内容将作为 CMD9 的响应发出。卡只送出 CID 和 CSD 的位[127...1]，在接收端这些寄存器的位 0 被响应的结束位所取代。卡通过拉低 MCDAT 指示它正在进行擦除操作；实际擦除操作的时间可能非常长，主机可以发送 CMD7 命令不选中这个卡。

表 113 R2 响应

位	域宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	'111111'	命令索引
[127:1]	127	X	卡状态
0	1	1	结束位

20.5.4 R3(OCR 寄存器)

代码长度=48 位。OCR 寄存器的内容将作为 CMD1 的响应发出。电平代码的定义是：限制的电压窗口=低，卡繁忙=低。

表 114 R3 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'111111'	保留
[39:8]	32	X	OCR 寄存器
[7:1]	7	'1111111'	保留
0	1	1	结束位

20.5.5 R4(快速 I/O)

代码长度=48 位。参数域包含指定卡的 RCA、需要读出或写入寄存器的地址、和它的内容。

表 115 R4 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	'100111'	CMD39
[39:8]参数域	[31:16]	16	X	RCA
	[15:8]	8	X	寄存器地址
	[7:0]	8	X	读寄存器的内容
[7:1]		7	X	CRC7
0		1	1	结束位

20.5.6 R4b

仅适合 SDI/O 卡：一个 SDIO 卡收到 CMD5 后将返回一个唯一的 SDIO 响应 R4。

表 116 R4b 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	X	保留
[39:8]参数域	39	16	X	卡已就绪
	[38:36]	3	X	I/O 功能数目
	35	1	X	当前存储器
	[34:32]	3	X	填充位

	[31:8]	24	X	I/OORC
[7:1]		7	X	保留
0		1	1	结束位

当一个 SDI/O 卡收到命令 CMD5，卡的 I/O 部分被使能并能够正常地响应所有后续的命令。I/O 卡的使能状态将保持到下一次复位、断电或收到 I/O 复位的 CMD52 命令。

注意：一个只包含存储器功能的 SD 卡可以响应 CMD5 命令，它的正确响应可以是：当前存储器=1，I/O 功能数目=0。按照 SD 存储器卡规范版本 1.0 设计的只包含存储器功能的 SD 卡，可以检测到 CMD5 命令为一个非法命令并不响应它。可以处理 I/O 卡的主机将发送 CMD5 命令，如果卡返回响应 R4，则主机会依据 R4 响应中的数据确定卡的配置。

20.5.7 R5(中断请求)

仅适用于多媒体卡。代码长度=48 位。如果这个响应由主机产生，则参数中的 RCA 域为 0x0。

表 117 R5 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	'101000'	CMD40
[39:8]参数域	[31:16]	16	X	成功的卡或主机的 RCA[31:16]
	[15:0]	16	X	未定义。可以作为中断数据。
[7:1]		7	X	CRC7
0		1	1	结束位

20.5.8 R6(中断请求)

仅适用于 SDI/O 卡。这是一个存储器设备对 CMD3 命令的正常响应。

表 118 R6 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	'101000'	CMD40
[39:8]参数域	[31:16]	16	X	成功的卡或主机的 RCA[31:16]
	[15:0]	16	X	未定义。可以作为中断数据。
[7:1]		7	X	CRC7
0		1	1	结束位

当发送 CMD3 命令到只有 I/O 功能的卡时，卡的状态位[23:8]会改变；此时，响应中的 16 位将是只有 I/O 功能的 SD 卡中的数值：

- 位 15=COM_CRC_ERROR
- 位 14=ILLEGAL_COMMAND
- 位 13=ERROR
- 位[12:0]=保留

20.6 SDIOI/O 卡特定的操作

下述功能是 SDI/O 卡特定的操作:

- 由 SDIO_D2 信号线实现的 SDIO 读等待操作。
- 通过停止时钟实现的 SDIO 读等待操作。

- SDIO 暂停/恢复操作(写和读暂停)
- SDIO 中断

只有设置了 SDIO_DCTRL[11]位时，SDIO 才支持这些操作；但读暂停除外，因为它不需要特殊的硬件操作。

20.6.1 使用 SDIO_D2 信号线的 SDIOI/O 读等待操作

在收到第一个数据块之前即可以开始读等待过程，使能数据通道(设置 SDIO_DCTRL[0]位)、使能 SDIO 特定操作(设置 SDIO_DCTRL[11]位)、开始读等待(SDIO_DCTRL[10]=0 并且 SDIO_DCTRL[8]=1)，同时数据传输方向是从卡至 SDIO 主机(SDIO_DCTRL[1]=1)，DPSM 将直接从空闲进入读等待状态。在读等待状态时，2 个 SDIO_CK 时钟周期后，DPSM 驱动 SDIO_D2 为'0'，在此状态，如果设置 RWSTOP 位(SDIO_DCTRL[9])，则 DPSM 会在等待状态多停留 2 个 SDIO_CK 时钟周期，(根据 SDIO 规范)并在一个时钟周期中驱动 SDIO_D2 为'1'。然后 DPSM 开始等待从卡里接收数据。在接收数据块时，即使设置了开始读等待，DPSM 也不会进入读等待，读等待过程将在收到 CRC 后开始。必须清除 RWSTOP 才能开始新的读等待操作。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

20.6.2 使用停止 SDIO_CK 的 SDIO 读等待操作

如果 SDIO 卡不能支持前述的读等待操作，SDIO 可以停止 SDIO_CK 进入读等待(按照 20.6.1 节介绍的方式设置 SDIO_DCTRL，但置 SDIO_DCTRL[10]=1)，在接收当前数据块结束位之后的 2 个 SDIO_CK 周期后，DSPM 停止时钟，在设置了读等待开始位后恢复时钟。

因为 SDIO_CK 停止了，可以向卡发送任何命令。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

20.6.3 SDIO 暂停/恢复操作

在向卡发送数据时，SDIO 可以暂停写操作。设置 SDIO_CMD[11]位，这指示 CPSM 当前的命令是一个暂停命令。CPSM 分析响应，在从卡收到 ACK 时(暂停被接受)，它确认在收到当前数据块的 CRC 后进入空闲状态。

硬件不会保存结束暂停操作之后，剩余的发送数据块数目。

可以通过软件暂停写操作：在收到卡对暂停命令的 ACK 时，停止 DPSM(SDIO_DCTRL[0]=0)，DPSM 即可进入空闲状态。

暂停读操作：DPSM 在 Wait_r 状态等待，在停止数据传输进入暂停之前，已经发送完成完整的数据包。随后应用程序继续读出 RxFIFO 直到 FIFO 变空，最后 DPSM 自动地进入空闲状态。

20.6.4 SDIO 中断

当设置了 SDIO_DCTRL[11]位，SDIO 主机在 SDIO_D1 信号线上监测 SDIO 中断。

20.7 CE-ATA 特定操作

下面是 CE-ATA 的特定操作:

- 送出命令完成信号能够关闭 CE-ATA 设备
- 从 CE-ATA 设备接收命令完成信号
- 使用状态位和/或中断，向 CPU 发送 CE-ATA 命令完成信号

仅当设置了 SDIO_CMD[14]位时，即 SDIO 主机只对 CE-ATA 的 CMD61 命令支持这些操作。

20.7.1 命令完成指示关闭

如果未设置 SDIO_CMD[12]中的“允许 CMD 结束位”并且设置了 SDIO_CMD[13]中的“非中断使能位”，则在收到一个短响应后的 8 个位周期之后，发出命令完成关闭信号。

在命令移位寄存器中写入关闭序列“00001”并且在命令计数器中写入 43，则 CPSM 进入暂停状态。8 个周期后，一个触发将 CPSM 移至发送状态。当命令计数器达到 48 时，因为没有要等待的响应，CPSM 变为空闲状态。

20.7.2 命令完成指示使能

如果设置 SDIO_CMD[12]中的“允许 CMD 结束位”并且设置了 SDIO_CMD[13]中的“非中断使能位”，CPSM 在 Waitcpl 状态下等待命令完成信号。

当在 CMD 信号上收到'0'，CPSM 进入空闲状态。在个 7 位周期之内不能发送新命令。然后，在最后 5 个周期(上述 7 个周期之外)，在推挽模式下 CMD 信号变为'1'。

20.7.3 CE-ATA 中断

命令完成是由状态位 SDIO_STA[23]通知 CPU，使用清除位 SDIO_ICR[23]可以清除该位。

根据屏蔽位 SDIO_MASKx[23]的设置，SDIO_STA[23]状态位可以在每一个中断线上产生中断。

20.7.4 中止 CMD61

如果还未发送”命令完成指示关闭”信号，但需要中止 CMD61 命令，命令状态机必须被关闭。然后它变成空闲，并且可以发送 CMD12 命令。在此操作期间，不传送”命令完成指示关闭”信号。

20.8 硬件流控制

使用硬件流控制功能可以避免 FIFO 下溢(发送模式)和上溢(接收模式)错误。

操作过程是停止 SDIO_CK 并冻结 SDIO 状态机，在 FIFO 不能进行发送和接收数据时，数据传输暂停。只有由 SDIOCLK 驱动的状态机被冻结，AHB 接口还在工作。即使在流控制起作用时，仍然可以读出或写入 FIFO。

必须设置 SDIO_CLKCR[14]位为'1'，才能使能硬件流控制。复位后，硬件流控制功能关闭。

20.9 SDIO 寄存器

设备通过可以在 AHB 上操作的 32 位控制寄存器与系统通信。

必须以字(32 位)的方式操作这些外设寄存器。

20.9.1 SDIO 电源控制寄存器(SDIO_POWER)

地址偏移：0x00

复位值：0x0000 0000

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

位	符号	说明
31:2	Reserved	保留，始终读为 0。
1:0	PWRCTRL	**PWRCTRL**：电源控制位(Power supply control bits) 这些位用于定义卡时钟的当前功能状态： 00：电源关闭，卡的时钟停止。 01：保留。 10：保留的上电状态。 11：上电状态，卡的时钟开启。

注意：　写数据后的7 个 HCLK 时钟周期内，不能写入这个寄存器。

20.9.2 SDIO 时钟控制寄存器(SDIO_CLKCR)

地址偏移：0x04

复位值：0x0000 0000

SDIO_CLKCR 寄存器控制 SDIO_CK 输出时钟。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
																	rw	rw	rw		rw	rw	rw	r/wr/wr/wr/wr/wr/wr/wr/w							

位	符号	说明
31:15	Reserved	保留，始终读为 0。
14	HWFC_EN	**HWFC_EN**：硬件流控制使能(HW Flow Control enable) 0：关闭硬件流控制 1：使能硬件流控制 当使能硬件流控制后，关于 TXFIFOE 和 RXFIFOF 中断信号的意义请参考 0 节的 SDIO 状态寄存器的定义。
13	NEGEDGE	**NEGEDGE**：SDIO_CK 相位选择位(SDIO_CK dephasing selection bit) 0：在主时钟 SDIOCLK 的上升沿产生 SDIO_CK。 1：在主时钟 SDIOCLK 的下降沿产生 SDIO_CK。
12:11	WIDBUS	**WIDBUS**：宽总线模式使能位(Wide bus mode enable bit) 00：默认总线模式，使用 SDIO_D0。 01：4 位总线模式，使用 SDIO_D[3:0]。 10：8 位总线模式，使用 SDIO_D[7:0]。
10	BYPASS	**BYPASS**：旁路时钟分频器(Clock divider bypass enable bit) 0：关闭旁路：驱动 SDIO_CK 输出信号之前，依据 CLKDIV 数值对 SDIOCLK 分频。 1：使能旁路：SDIOCLK 直接驱动 SDIO_CK 输出信号。
9	PWRSAV	**PWRSAV**：省电配置位(Power saving configuration bit) 为了省电，当总线为空闲时，设置 PWRSAV 位可以关闭 SDIO_CK 时钟输出。 0：始终输出 SDIO_CK。 1：仅在有总线活动时才输出 SDIO_CK。
8	CLKEN	**CLKEN**：时钟使能位(Clock enable bit) 0：SDIO_CK 关闭。

		1：SDIO_CK 使能。
7:0	**CLKDIV**	**CLKDIV**：时钟分频系数(Clock divide factor) 这个域定义了输入时钟(SDIOCLK)与输出时钟(SDIO_CK)间的分频系数： SDIO_CK 频率=SDIOCLK/[CLKDIV+2]。

注意：1 当 SD/SDIO 卡或多媒体卡在识别模式，SDIO_CK 的频率必须低于 400kHz。

2 当所有卡都被赋予了相应的地址后，时钟频率可以改变到卡总线允许的最大频率。

3 写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。对于 SDIO 卡，在读等待期间可以停止 SDIO_CK，此时 SDIO_CLKCR 寄存器不控制 SDIO_CK。

20.9.3 SDIO 参数寄存器(SDIO_ARG)

地址偏移：0x08

复位值：0x0000 0000

SDIO_ARG 寄存器包含 32 位命令参数，它将作为命令的一部分发送到卡中。

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

位	符号	说明
31:0	**CMDARG**	**CMDARG**：命令参数(Command argument) 命令参数是发送到卡中命令的一部分，如果一个命令包含一个参数，必须在写命令到命令寄存器之前加载这个寄存器。

20.9.4 SDIO 命令寄存器(SDIO_CMD)

地址偏移：0x0C

复位值：0x0000 0000

SDIO_CMD 寄存器包含命令索引和命令类型位。命令索引是作为命令的一部分发送到卡中。命令类型位控制命令通道状态机(CPSM)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																	CE_ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX						
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:15	**Reserved**	保留，始终读为 0。
14	**ATACMD**	**ATACMD**：CE-ATA 命令(CE-ATA command) 如果设置该位，CPSM 转至 CMD61。
13	**nIEN**	**nIEN**：不使能中断(not interrupt enable) 如果未设置该位，则使能 CE-ATA 设备的中断。
12	**ENCMDcompl**	**ENCMDcompl**：使能 CMD 完成(Enable CMDcompletion) 如果设置该位，则使能命令完成信号。
11	**SDIOSuspend**	**SDIOSuspend**：SDI/O 暂停命令(SDI/O suspend command) 如果设置该位，则将要发送的命令是一个暂停命令(只能用于 SDIO 卡)。
10	**CPSMEN**	**CPSMEN**：命令通道状态机(CPSM)使能位(Command path state machine (CPSM) Enable bit) 如果设置该位，则使能 CPSM。

9	WAITPEND	**WAITPEND**：CPSM 等待数据传输结束(CmdPend 内部信号)(CPSM Waits for ends of data transfer(CmdPend internal signal)) 如果设置该位，则 CPSM 在开始发送一个命令之前等待数据传输结束。
8	WAITINT	**WAITINT**：CPSM 等待中断请求(CPSM waits for interrupt request) 如果设置该位，则 CPSM 关闭命令超时控制并等待中断请求。
7:6	WAITRESP	**WAITRESP**：等待响应位(Wait for response bits) 这 2 位指示 CPSM 是否需要等待响应，如果需要等待响应，则指示响应类型。 00：无响应，期待 CMDSENT 标志 01：短响应，期待 CMDREND 或 CCRCFAIL 标志 10：无响应，期待 CMDSENT 标志 11：长响应，期待 CMDREND 或 CCRCFAIL 标志
5:0	CMDINDEX	**CMDINDEX**：命令索引(Command index) 命令索引是作为命令的一部分发送到卡中。

注意：1 写数据后的7 个 HCLK 时钟周期内不能写入这个寄存器。

2 多媒体卡可以发送 2 种响应：48 位长的短响应，或 136 位长的长响应。SD 卡和 SDIO 卡只能发送短响应，参数可以根据响应的类型而变化，软件将根据发送的命令区分响应的类型。CE-ATA 设备只发送短响应。

20.9.5 SDIO 命令响应寄存器(SDIO_RESPCMD)

地址偏移：0x10

复位值：0x0000 0000

SDIO_RESPCMD 寄存器包含最后收到的命令响应中的命令索引。如果传输的命令响应不包含命令索引(长响应或 OCR 响应)，尽管它应该包含 111111b(响应中的保留域值)，但 RESPCMD 域的内容未知。

31:6	5:0
保留	RESPCMD
r	r r r r r r

位	符号	说明
31:6	Reserved	保留，始终读为 0。
5:0	RESPCMD	**RESPCMD**：响应的命令索引(Response command index) 只读位，包含最后收到的命令响应中的命令索引。

20.9.6 SDIO 响应 1..4 寄存器(SDIO_RESPx)

地址偏移：0x14+4*(x-1)，其中 x=1..4

复位值：0x0000 0000

SDIO_RESP1/2/3/4 寄存器包含卡的状态，即收到响应的部分信息。

31:0
CARDSTATUSx
r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r

位	符号	说明
31:0	CARDSTATUSx	**CARDSTATUSx**：见下表。

根据响应状态，卡的状态长度是 32 位或 127 位。

表 119 响应类型和 SDIO_RESPx 寄存器

寄存器	短响应	长响应
SDIO_RESP1	卡状态[31:0]	卡状态[127:96]
SDIO_RESP2	不用	卡状态[95:64]

SDIO_RESP3	不用	卡状态[63:32]
SDIO_RESP4	不用	卡状态[31:1]

20.9.7 SDIO 数据定时器寄存器(SDIO_DTIMER)

地址偏移：0x24

复位值：0x0000 0000

SDIO_DTIMER 寄存器包含以卡总线时钟周期为单位的数据超时时间。

一个计数器从 SDIO_DTIMER 寄存器加载数值，并在数据通道状态机(DPSM)进入 Wait_R 或繁忙状态时进行递减计数，当 DPSM 处在这些状态时，如果计数器减为 0，则设置超时标志。

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

位	符号	说明
31:0	DATATIME	**DATATIME**：数据超时时间(Data timeout period) 以卡总线时钟周期为单位的数据超时时间。

注意在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

20.9.8 SDIO 数据长度寄存器(SDIO_DLEN)

地址偏移：0x28

复位值：0x0000 0000

SDIO_DLEN 寄存器包含需要传输的数据字节长度。当数据传输开始时，这个数值被加载到数据计数器中。

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

位	符号	说明
31:25	Reserved	保留，始终读为 0。
25:0	DATALENGTH	**DATALENGTH**：数据长度(Data length value) 要传输的数据字节数目。

注意: 对于块数据传输，数据长度寄存器中的数值必须是数据块长度(见 SDIO_DCTRL)的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

20.9.9 SDIO 数据控制寄存器(SDIO_DCTRL)

地址偏移：0x2C

复位值：0x0000 0000

SDIO_DCTRL 寄存器控制数据通道状态机(DPSM)。

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

保留	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE	DMAEN	DTMODE	DTDIR	DTEN
	rw	rw	rw	rw	rw rw rw rw	rw	rw	rw	rw

位	符号	说明
31:12	Reserved	保留，始终读为 0。
11	SDIOEN	**SDIOEN**：SDI/O 使能功能(SD I/O enable functions) 如果设置了该位，则 DPSM 执行 SDI/O 卡特定的操作。
10	RWMOD	**RWMOD**：读等待模式(Read wait mode) 0：停止 SDIO_CK 控制读等待； 1：使用 SDIO_D2 控制读等待。
9	RWSTOP	**RWSTOP**：读等待停止(Read wait stop) 0：如果设置了 RWSTART，执行读等待； 1：如果设置了 RWSTART，停止读等待。
8	RWSTART	**RWSTART**：读等待开始(Read wait start) 设置该位开始读等待操作。
7:4	DBLOCKSIZE	**DBLOCKSIZE**：数据块长度(Data block size) 当选择了块数据传输模式，该域定义数据块长度： 0000：块长度=20=1 字节；　1000：块长度=28=256 字节； 0001：块长度=21=2 字节；　1001：块长度=29=512 字节； 0010：块长度=22=4 字节；　1010：块长度=210=1024 字节； 0011：块长度=23=8 字节；　1011：块长度=211=2048 字节； 0100：(十进制 4)块长度=24=16 字节；　1100：块长度=212=4096 字节； 0101：(十进制 5)块长度=25=32 字节；　1101：块长度=213=8192 字节； 0110：(十进制 6)块长度=26=64 字节；　1110：块长度=214=16384 字节； 0111：块长度=27=128 字节；　1111：保留。
3	DMAEN	DMAEN：DMA 使能位(DMA enable bit) 0：关闭 DMA； 1：使能 DMA。
2	DTMODE	DTMODE：数据传输模式(Data transfer mode selection) 0：块数据传输； 1：流数据传输。
1	DTDIR	DTDIR：数据传输方向(Data transfer direction selection) 0：控制器至卡； 1：卡至控制器。
0	DTEN	DTEN：数据传输使能位(Data transfer enabled bit) 如果设置该位为 1，则开始数据传输。根据 DTSIR 方向位，DPSM 进入 Wait_S 或 Wait_R 状态，如果在传输的一开始就设置了 RWSTART 位，则 DPSM 进入读等待状态。不需要在数据传输结束后清除使能位，但必须更改 SDIO_DCTRL 以允许新的数据传输。

20.9.10 SDIO 数据计数器寄存器(SDIO_DCOUNT)

地址偏移：0x30

复位值：0x0000 0000

当 DPSM 从空闲状态进入 Wait_R 或 Wait_S 状态时，SDIO_DCOUNT 寄存器从数据长度寄存器加载数值(见 SDIO_DLEN)，在数据传输过程中，该计数器的数值递减直到减为 0，然后 DPSM 进入空闲状态并设置数据状态结束标志 DATAEND。

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

位	符号	说明
31:25	Reserved	保留，始终读为 0。
25:0	DATACOUNT	**DATACOUNT**：数据计数数值(Data count value) 读这个寄存器时返回待传输的数据字节数，写这个寄存器无作用。

20.9.11 SDIO 状态寄存器(SDIO_STA)

地址偏移：0x34

复位值：0x0000 0000

SDIO_STA 是一个只读寄存器，它包含两类标志：

- 静态标志(位[23:22、10:0])：写入 SDIO 中断清除寄存器(见 SDIO_ICR)，可以清除这些位。
- 动态标志(位[21:11])：这些位的状态变化根据它们对应的那部分逻辑而变化(例如：FIFO 满和空标志变高或变低随 FIFO 的数据写入变化)。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位	符号	说明
31:24	Reserved	保留，始终读为 0。
23	CEATAEND	**CEATAEND**：在 CMD61 接收到 CE-ATA 命令完成信号(CE-ATA command completion signal received for CMD61)
22	SDIOIT	**SDIOIT**：收到 SDIO 中断(SDIO interrupt received)
21	RXDVAL	**RXDVAL**：在接收 FIFO 中的数据可用(Data available in receive FIFO)
20	TXDVAL	**TXDVAL**：在发送 FIFO 中的数据可用(Data available in transmit FIFO)
19	RXFIFOE	**RXFIFOE**：接收 FIFO 空(Receive FIFO empty)
18	TXFIFOE	**TXFIFOE**：发送 FIFO 空(Transmit FIFO empty) 若使用了硬件流控制，当 FIFO 包含 2 个字时，TXFIFOE 信号变为有效。
17	RXFIFOF	**RXFIFOF**：接收 FIFO 满(Receive FIFO full) 若使用了硬件流控制，当 FIFO 还差 2 个字满时，RXFIFOF 信号变为有效。
16	TXFIFOF	**TXFIFOF**：发送 FIFO 满(Transmit FIFO full)
15	RXFIFOHF	**RXFIFOHF**：接收 FIFO 半满(Receive FIFO half full)：FIFO 中至少还有 8 个字。
14	TXFIFOHE	**TXFIFOHE**：发送 FIFO 半空(Transmit FIFO half empty)：FIFO 中至少还可以写入 8 个字。
13	RXACT	**RXACT**：正在接收数据(Data receive inprogress)
12	TXACT	**TXACT**：正在发送数据(Data transmit inprogress)
11	CMDACT	**CMDACT**：正在传输命令(Command transfer inprogress)
10	DBCKEND	**DBCKEND**：已发送/接收数据块(CRC 检测成功)(Data block sent/received(CRC check passed))
9	STBITERR	**STBITERR**：在宽总线模式，没有在所有数据信号上检测到起始位(Start bit not detected on all data signals in wide bus mode)

8	DATAEND	**DATAEND**：数据结束(数据计数器，SDIO_DCOUNT=0)(Data end (data counter,SDID COUNT,is zero))
7	CMDSENT	**CMDSENT**：命令已发送(不需要响应)(Command sent(no response required))
6	CMDREND	**CMDREND**：已接收到响应(CRC 检测成功)(Command response)
5	RXOVERR	**RXOVERR**：接收 FIFO 上溢错误(Received FIFO overrun error)
4	TXUNDERR	**TXUNDERR**：发送 FIFO 下溢错误(Transmit FIFO underrun error)
3	DTIMEOUT	**DTIMEOUT**：数据超时(Data timeout)
2	CTIMEOUT	**CTIMEOUT**：命令响应超时(Command response timeout)命令超时时间是一个固定的值，为 64 个 SDIO_CK 时钟周期。
1	DCRCFAIL	**DCRCFAIL**：已发送/接收数据块(CRC 检测失败)(Data block sent/received)
0	CCRCFAIL	**CCRCFAIL**：已收到命令响应(CRC 检测失败)(Command response received)

20.9.12 SDIO 清除中断寄存器(SDIO_ICR)

地址偏移：0x38

复位值：0x0000 0000

SDIO_ICR 是一个只写寄存器，在对应寄存器位写'1'将清除 SDIO_STA 状态寄存器中的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAENDC	SDIOITC	保留											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
								rw	rw												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:24	Reserved	保留，始终读为 0。
23	CEATAENDC	**CEATAENDC**：CEATAEND 标志清除位(CEATAEND flag clear bit) 软件设置该位以清除 CEATAEND 标志。
22	SDIOITC	**SDIOITC**：SDIOIT 标志清除位(SDIOIT flag clear bit) 软件设置该位以清除 SDIOIT 标志。
21:11	Reserved	保留，始终读为 0。
10	DBCKENDC	**DBCKENDC**：DBCKEND 标志清除位(DBCKEND flagclear bit)软件设置该位以清除 DBCKEND 标志。
9	STBITERRC	**STBITERRC**：STBITERR 标志清除位(STBITERR flagclear bit)软件设置该位以清除 STBITERR 标志。
8	DATAENDC	**DATAENDC**：DATAEND 标志清除位(DATAEND flag clear bit) 软件设置该位以清除 DATAEND 标志。
7	CMDSENTC	**CMDSENTC**：CMDSENT 标志清除位(CMDSENT flag clear bit)软件设置该位以清除 CMDSENT 标志。
6	CMDRENDC	**CMDRENDC**：CMDREND 标志清除位(CMDREND flag clear bit)软件设置该位以清除 CMDREND 标志。
5	RXOVERRC	**RXOVERRC**：RXOVERR 标志清除位(RXOVERR flag clear bit)软件设置该位以清除 RXOVERR 标志。
4	TXUNDERRC	**TXUNDERRC**：TXUNDERR 标志清除位(TXUNDERR flag clear bit) 软件设置该位以清除 TXUNDERR 标志。
3	DTIMEOUTC	**DTIMEOUTC**：DTIMEOUT 标志清除位(DTIMEOUT flag clear bit) 软件设置该位以清除 DTIMEOUT 标志。
2	CTIMEOUT	**CTIMEOUT**：CTIMEOUT 标志清除位(CTIMEOUT flag clear bit)软件设置该位以清除 CTIMEOUT 标志。

1	DCRCFAILC	**DCRCFAILC**：DCRCFAIL 标志清除位(DCRCFAIL flag clear bit)软件设置该位以清除 DCRCFAIL 标志。
0	CCRCFAILC	**CCRCFAILC**：CCRCFAIL 标志清除位。(CCRCFAIL flag clear bit) 软件设置该位以清除 CCRCFAIL 标志。

20.9.13 SDIO 中断屏蔽寄存器(SDIO_MASK)

地址偏移：0x3C

复位值：0x0000 0000

在对应位置'1'，SDIO_MASK 中断屏蔽寄存器决定哪一个状态位产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENIE	CMDRENIE	RXOVERRIE	TXUNDERRIE	DTMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:24	Reserved	保留，始终读为 0。
23	CEATAENDIE	**CEATAENDIE**：允许接收到 CE-ATA 命令完成信号产生中断(CE-ATA command completion signal received interrupt enable) 由软件设置/清除该位，允许/关闭在收到 CE-ATA 命令完成信号产生中断功能。 0：收到 CE-ATA 命令完成信号时不产生中断 1：收到 CE-ATA 命令完成信号时产生中断
22	SDIOITIE	**SDIOITIE**：允许 SDIO 模式中断已接收中断(SDIO mode interrupt received interrupt enable) 由软件设置/清除该位，允许/关闭 SDIO 模式中断已接收中断功能。 1：SDIO 模式中断已接收不产生中断 0：SDIO 模式中断已接收产生中断
21	RXDVALIE	**RXDVALIE**：接收 FIFO 中的数据有效产生中断(Data available in RxFIFO interrupt enable) 由软件设置/清除该位，允许/关闭接收 FIFO 中的数据有效中断。 0：接收 FIFO 中的数据有效不产生中断 1：接收 FIFO 中的数据有效产生中断
20	TXDVALIE	**TXDVALIE**：发送 FIFO 中的数据有效产生中断(Dat aavailable in RxFIFO interrupt enable) 由软件设置/清除该位，允许/关闭发送 FIFO 中的数据有效中断。 0：发送 FIFO 中的数据有效不产生中断 1：发送 FIFO 中的数据有效产生中断
19	RXFIFOEIE	**RXFIFOEIE**：接收 FIFO 空产生中断(RxFIFO empty interrupt enable) 由软件设置/清除该位，允许/关闭接收 FIFO 空中断。 0：接收 FIFO 空不产生中断 1：接收 FIFO 空产生中断
18	TXFIFOEIE	**TXFIFOEIE**：发送 FIFO 空产生中断(TxFIFO empty interrupt enable)

		由软件设置/清除该位，允许/关闭发送 FIFO 空中断。 0：发送 FIFO 空不产生中断 1：发送 FIFO 空产生中断
17	RXFIFOFIE	**RXFIFOFIE**：接收 FIFO 满产生中断(RxFIFO full interrupt enable) 由软件设置/清除该位，允许/关闭接收 FIFO 满中断。 0：接收 FIFO 满不产生中断 1：接收 FIFO 满产生中断
16	TXFIFOFIE	**TXFIFOFIE**：发送 FIFO 满产生中断(TxFIFO full interrupt enable) 由软件设置/清除该位，允许/关闭发送 FIFO 满中断。 0：发送 FIFO 满不产生中断 1：发送 FIFO 满产生中断
15	RXFIFOHFIE	**RXFIFOHFIE**：接收 FIFO 半满产生中断(RxFIFO half full interrupt enable) 由软件设置/清除该位，允许/关闭接收 FIFO 半满中断。 0：接收 FIFO 半满不产生中断 1：接收 FIFO 半满产生中断
14	TXFIFOHE	**TXFIFOHE**：发送 FIFO 半空产生中断(TxFIFO half empty interrupt enable) 由软件设置/清除该位，允许/关闭发送 FIFO 半空中断。 0：发送 FIFO 半空不产生中断 1：发送 FIFO 半空产生中断
13	RXACTIE	**RXACTIE**：正在接收数据产生中断(Data receive acting interrupt enable) 由软件设置/清除该位，允许/关闭正在接收数据中断。 0：正在接收数据不产生中断 1：正在接收数据产生中断
12	TXACTIE	**TXACTIE**：正在发送数据产生中断(Data transmit acting interrupt enable)由软件设置/清除该位，允许/关闭正在发送数据中断。 0：正在发送数据不产生中断 1：正在发送数据产生中断
11	CMDACTIE	**CMDACTIE**：正在传输命令产生中断(Command acting interrupt enable) 由软件设置/清除该位，允许/关闭正在传输命令中断。 0：正在传输命令不产生中断 1：正在传输命令产生中断
10	DBCKENDIE	**DBCKENDIE**：数据块传输结束产生中断(Data block end interrupt enable) 由软件设置/清除该位，允许/关闭数据块传输结束中断。 0：数据块传输结束不产生中断 1：数据块传输结束产生中断
9	STBITERRIE	**STBITERRIE**：起始位错误产生中断(Start bit error interrupt enable) 由软件设置/清除该位，允许/关闭起始位错误中断。 0：起始位错误不产生中断 1：起始位错误产生中断
8	DATAENDIE	**DATAENDIE**：数据传输结束产生中断(Dataendinterruptenable) 由软件设置/清除该位，允许/关闭数据传输结束中断。 0：数据传输结束不产生中断 1：数据传输结束产生中断
7	CMDSENTIE	**CMDSENTIE**：命令已发送产生中断(Command sent interrupt enable) 由软件设置/清除该位，允许/关闭命令已发送中断。 0：命令已发送不产生中断 1：命令已发送产生中断
6	CMDRENDIE	**CMDRENDIE**：接收到响应产生中断(Command response received interrupt enable) 由软件设置/清除该位，允许/关闭接收到响应中断。 0：接收到响应不产生中断

		1：接收到响应产生中断
5	RXOVERRIE	**RXOVERRIE**：接收 FIFO 上溢错误产生中断(RxFIFO overrun error interrupt enable) 由软件设置/清除该位，允许/关闭接收 FIFO 上溢错误中断。 0：接收 FIFO 上溢错误不产生中断 1：接收 FIFO 上溢错误产生中断
4	TXUNDERRIE	**TXUNDERRIE**：发送 FIFO 下溢错误产生中断(TxFIFO underrun error interrupt enable) 由软件设置/清除该位，允许/关闭发送 FIFO 下溢错误中断。 0：发送 FIFO 下溢错误不产生中断 1：发送 FIFO 下溢错误产生中断
3	DTIMEOUTIE	**DTIMEOUTIE**：数据超时产生中断(Data timeout interrupt enable) 由软件设置/清除该位，允许/关闭数据超时中断。 0：数据超时不产生中断 1：数据超时产生中断
2	CTIMEOUTIE	**CTIMEOUTIE**：命令超时产生中断(Command timeout interrupt enable) 由软件设置/清除该位，允许/关闭命令超时中断。 0：命令超时不产生中断 1：命令超时产生中断
1	DCRCFAILIE	**DCRCFAILIE**：数据块 CRC 检测失败产生中断(Data CRC fail interrupt enable) 由软件设置/清除该位，允许/关闭数据块 CRC 检测失败中断。 0：数据块 CRC 检测失败不产生中断 1：数据块 CRC 检测失败产生中断
0	CCRCFAILIE	**CCRCFAILIE**：命令 CRC 检测失败产生中断(Command CRC fail interrupt enable) 由软件设置/清除该位，允许/关闭命令 CRC 检测失败中断。 0：命令 CRC 检测失败不产生中断 1：命令 CRC 检测失败产生中断

20.9.14 SDIOFIFO 计数器寄存器(SDIO_FIFOCNT)

地址偏移：0x48

复位值：0x0000 0000

SDIO_FIFOCNT 寄存器包含还未写入 FIFO 或还未从 FIFO 读出的数据字数目。当在数据控制寄存器(SDIO_DCTRL)中设置了数据传输使能位 DTEN，并且 DPSM 处于空闲状态时，FIFO 计数器从数据长度寄存器(见 SDIO_DLEN)加载数值。如果数据长度未与字对齐(4 的倍数)，则最后剩下的 1~3 个字节被当成一个字处理。

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

位	符号	说明
31:24	Reserved	保留，始终读为 0。
23:0	FIFOCOUNT	**FIFOCOUNT**：将要写入 FIFO 或将要从 FIFO 读出数据字的数目。

20.9.15 SDIO 数据 FIFO 寄存器(SDIO_FIFO)

地址偏移：0x80

复位值：0x0000 0000

接收和发送 FIFO 是 32 位的宽度读或写一组寄存器，它在连续的 32 个地址上包含 32 个寄存器，CPU 可以使用 FIFO 读写多个操作数。

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

FIFODATA
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

位	符号	说明
31:0	FIFODATA	**FIFODATA**：接收或发送 FIFO 数据(Receive and transmit FIFO data) FIFO 数据占据 32 个 32 位的字，地址为： (SDIO 基址+0x80)至(SDIO 基址+0xFC)

20.9.16 SDIO 寄存器映像

下表是 SDIO 寄存器的总结。

表 120 SDIO 寄存器映像

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	保留																															PWRCTRL
0x04	SDIO_CLKCR	保留																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	SDIO_ARG	CMDARG																															
0x0C	SDIO_CMD	保留																	CE-ATACMD	nIEN	ENCMDcompl	SDIOsuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX						
0x10	SDIO_RESPCMD	保留																															RESPCMD
0x14	SDIO_RESP1	CARDSTATUS1																															
0x18	SDIO_RESP2	CARDSTATUS2																															
0x1C	SDIO_RESP3	CARDSTATUS3																															
0x20	SDIO_RESP4	CARDSTATUS4																															
0x24	SDIO_DTIMER	DATATIME																															
0x28	SDIO_DLEN	保留								DATALENGTH																							
0x2C	SDIO_DCTRL	保留																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE			DMAEN	DTMODE	DTDIR	DTEN
0x30	SDIO_DCOUNT	保留								DATACOUNT																							
0x34	SDIO_STA	保留								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	SDIO_ICR	保留								CEATAENDC	SDIOITC	保留											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	SDIO_MASK	保留								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x48	SDIO_FIFOCNT	保留								FIFOCOUNT																							
0x80	SDIO_FIFO	FIFODATA																															

有关寄存器的起始地址，参见表 1。

21 USB 全速设备接口(USB)

21.1 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。

21.2 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 512 字节的 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。*

下图是 USB 外设的方框图

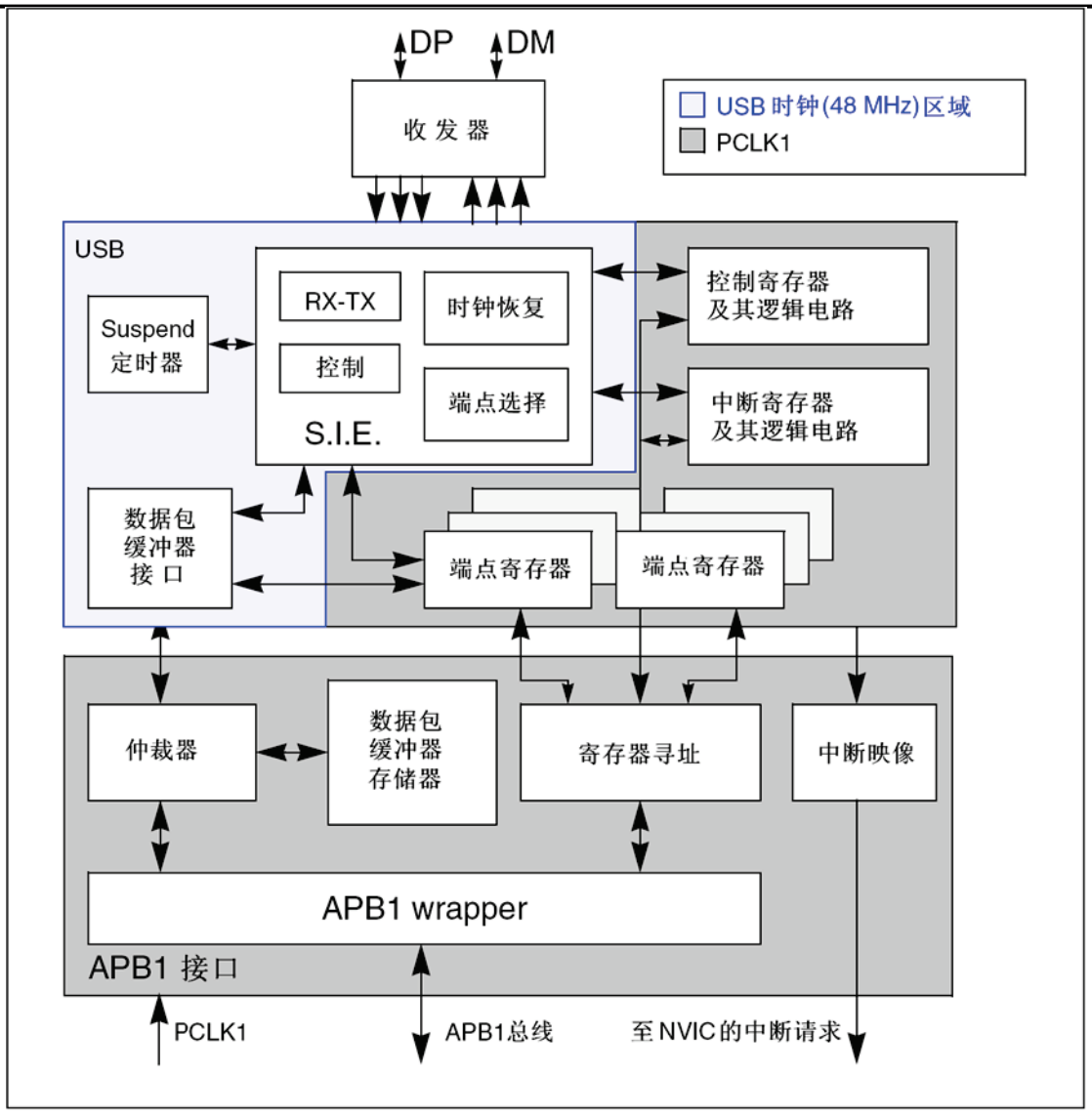


图 194 USB 设备框图

21.3 USB 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512 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。

21.3.1 USB 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口控制器(SIE)：该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧首(Start of Frame)，USB 复位，数据错误等等，这些信号用来产生中断。
- 定时器：本模块的功能是产生一个与帧开始报文同步的时钟脉冲，并在 3ms 内没有数据传输的状态，检测出(主机的)全局挂起条件。
- 分组缓冲器接口：此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。
- 端点相关寄存器：每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。
- 控制寄存器：这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复，低功耗等 USB 事件。
- 中断寄存器：这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

注意：　端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512 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 NVIC 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。

21.4 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。

21.4.1 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：
首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。
其次，必须配置 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 tSTARTUP)，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。
最后，应用程序需要通过配置设备时钟管理逻辑的相应控制位来为 USB 模块提供标准所定义的 48MHz 时钟。
当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化(比如中断使能的选择，分组缓冲区地址的选择等)。接下来按照 USB 复位处理(参见下段)。

USB 复位(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止(USB 模块不会响应任何分组)。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点(端点 0)也需要被使能。这可以通过配置 USB_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，这个地址必须写入 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。

注意： *为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 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')。第 21.5.3 节详细介绍缓冲区描述表表项。

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(请分别参考第 21.4.3 节：同步传输和第 21.4.2 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。

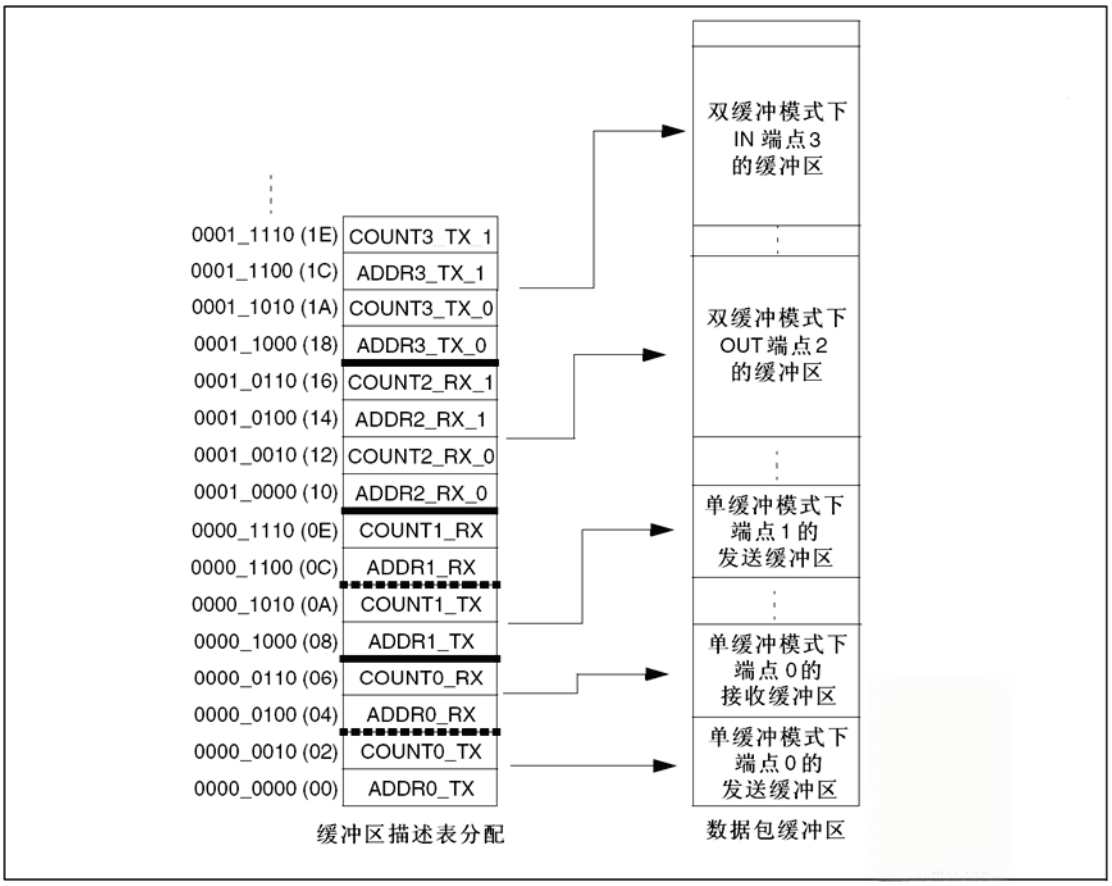


图 195 分组缓冲区对应的缓冲区描述表项定位

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 分组的相应字段决定后面的传输是 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，应用程序应该清除 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。

21.4.2 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理(对于 IN 分组来说，情况是一样的)。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_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 121 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 122 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，见表 122)，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。

21.4.3 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组(由端点传输方向确定分组类型)。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送 ACK 分组。同样，同步传输只传送 PID(分组 ID)为 DATA0 的数据包,而不会用到数据翻转机制。

通过设置 USB_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 123 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。

21.4.4 挂起/恢复事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 500uA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 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，如表 124 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 124 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。

注意：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。

21.5 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。

关于寄存器描述中使用的一些缩略语，请参考第 1 节。可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

21.5.1 通用寄存器

这组寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

USB 控制寄存器(USB_CNTR)

地址偏移：0x40

复位值：0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	保留			RESUME	FSUSP	LPMODE	PDMN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位	符号	说明
15	CTRM	**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。
14	PMAOVRM	**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。
13	ERRM	**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。
12	WKUPM	**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。
11	SUSPM	**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。
10	RESETM	**RESETM**：USB 复位中断屏蔽位(USB reset interrupt mask) 0：USBRESET 中断禁止 1：USBRESET 中断使能，在中断寄存器的相应位被置 1 时产生中断。
9	SOFM	**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。
8	ESOFM	**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。
7:5	Reserved	保留位，始终为 0
4	RESUME	**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。
3	FSUSP	**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。
2	LP_MODE	**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
1	PDWN	**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
0	FRES	**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)

地址偏移：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留			DIR	EP_ID[3:0]			
r	rcw0	rcw0	rcw0	rcw0	rcw0	rcw0	rcw0				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 USBIRQ：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 USBIRQ：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。

下面详细描述每个位：

位	符号	说明
位 15	**CTR**	**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
位 14	**PMAOVR**	**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。
位 13	**ERR**	**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。
位 12	**WKUP**	**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。
位 11	**SUSP**	**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。
位 10	**RESET**	**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。
位 9	**SOF**	**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。
位 8	**ESOF**	**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。
位 7:5	**Reserved**	保留位，始终为 0
位 4	**DIR**	**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
位 3:0	**EP_ID[3:0]**	**EP_ID[3:0]**：端点 ID(End point 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)

地址偏移：0x48

复位值：0x0XXX，X 代表未定义数值

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

位	符号	说明
15	**RXDP**	**RXDP**：D+状态位(Receive data+line status) 此位用于观察 USBD+数据线的状态，可在挂起状态下检测唤醒条件的出现。
14	**RXDM**	**RXDM**：D-状态位(Receive data-line status) 此位用于观察 USBD-数据线的状态，可在挂起状态下检测唤醒条件的出现。
13	**LCK**	**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。
12:11	**LSOF[1:0]**	**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。
10:0	**FN[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)

地址偏移：0x4C

复位值：0x0000

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

位	符号	说明
15:8	**Reserved**	保留位，始终为 0
7	**EF**	**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。
6:0	**ADD[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)

地址偏移：0x50

复位值：0x0000

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

位	符号	说明
15	BTABLE[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。
14	Reserved	保留位，由硬件置为 0

21.5.2 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。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_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]

地址偏移：0x00 至 0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX		SETUP	EPTYPE[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

位	符号	说明
15	CTR_RX	**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。
14	DTOG_RX	**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 21.4.3 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 21.4.4 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。
13:12	STAT_RX[1:0]	**STAT_RX[1:0]**：用于数据接收的状态位(Status bits,for reception transfers) 此位用于指示端点当前的状态，表 125 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 21.4.3 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。
11	SETUP	**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。
10:9	EP_TPYE[1:0]	**EP_TPYE[1:0]**：端点类型位(End point type) 此位用于指示端点当前的类型，所有的端点类型都在表 126 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 21.4.4 同步传输。
8	EP_KIND	**EP_KIND**：端点特殊类型位(End point kind) 此位的需要和 EP_TYPE 位配合使用，具体的定义请参考表 127。 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 21.4.3 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。
7	CTR_TX	**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。
6	DTOG_RX	**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 SETUPPID 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 21.4.3 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 21.4.4 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。

5:4	STAT_TX[1:0]	**STAT_TX[1:0]**：用于发送数据的状态位(Status bits,for transmission transfers) 此位用于标识端点的当前状态，表 128 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 21.4.3 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。
3:0	EA[3:0]	**EA[3:0]**：端点地址(End point 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 125 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 126 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 127 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 128 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。

21.5.3 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 21.4.1 节。

发送缓冲区地址寄存器 n(USB_ADDRn_TX)

地址偏移：[USB_BTABLE]+n×16

USB 本地地址：[USB_BTABLE]+n×8

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

位	符号	说明
15:1	ADDRn_TX [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。
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)

地址偏移：[USB_BTABLE]+n×16

USB 本地地址：[USB_BTABLE]+n×8+2

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

位	符号	说明
15:10	Reserved	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。
9:0	COUNTn_TX [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
-						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
-						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)

地址偏移：[USB_BTABLE]+n×16+8

USB 本地地址：[USB_BTABLE]+n×8+4

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

位	符号	说明
15:1	ADDRn_RX [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。
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'。

接收数据字节数寄存器 n(USB_COUNTn_RX)

地址偏移：[USB_BTABLE]+n×16+12

USB 本地地址：[USB_BTABLE]+n×8+6

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	r	r	r	r	r	r	r	r	r	r

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”USB2.0 协议规范”)

位	符号	说明
15	**BL_SIZE**	**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。
14:10	**NUM_BLOCK[4:0]**	**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。具体请参考表 129。
9:0	**COUNTn_RX[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 129 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
00000	不允许使用	32 字节
00001	2 字节	64 字节
00010	4 字节	96 字节
00011	6 字节	128 字节
…	…	…
01111	30 字节	512 字节
10000	32 字节	保留
10001	34 字节	保留
10010	36 字节	保留
…	…	…
11110	60 字节	保留
11111	62 字节	保留

21.5.4 USB 寄存器映像

表 130 USB 寄存器映像和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	USB_EP0R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04h	USB_EP1R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	USB_EP2R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0Ch	USB_EP3R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	USB_EP4R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	USB_EP5R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18h	USB_EP6R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Ch	USB_EP7R	保留																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]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20h~3Fh	保留																																
40h	USB_CNTR	保留																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	保留		RESUME	FSUSP	LPMODE	PDWN	FRES	
	复位值																	0	0	0	0	0	0	0	0			0	0	0	1	1	
44h	USB_ISTR	保留																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	保留		DIR	EP_ID[3:0]				
	复位值																	0	0	0	0	0	0	00	0			0	0	0	0	0	

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
48h	USB_FNR	保留																RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
	复位值																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
4Ch	USB_DADDR	保留																保留								EF	ADD[6:0]						
	复位值																									0	0	0	0	0	0	0	0
50h	USB_BTABLE	保留																BTABLE[15:3]															保留
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

关于寄存器的起始地址，请参见表 1。

22 控制器局域网(bxCAN)

22.1 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。

22.2 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：W55MH32 有 14 个过滤器组
- 标识符列表
- FIFO 溢出处理方式可配置
- 记录接收 SOF 时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16 位自由运行定时器
- 可在最后 2 个数据字节发送时间戳

管理

- 中断可屏蔽
- 邮箱占用单独 1 块地址空间，便于提高软件效率

注：USB 和 CAN 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不同同时使用 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。

22.3 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

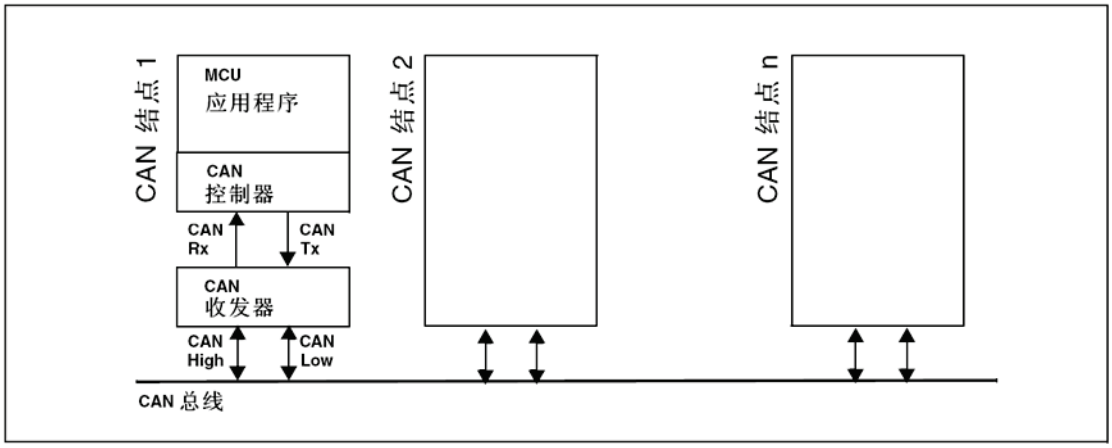


图 196 CAN 网拓扑结构

22.3.1 CAN2.0B 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)。

22.3.2 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

22.3.3 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。

22.3.4 接收过滤器

W55MH32 有 14 个位宽可变/可配置的标识符过滤器组。

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。

22.4 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。

22.4.1 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)。

22.4.2 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'1'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。

22.4.3 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'1'，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，请参考图 197：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。

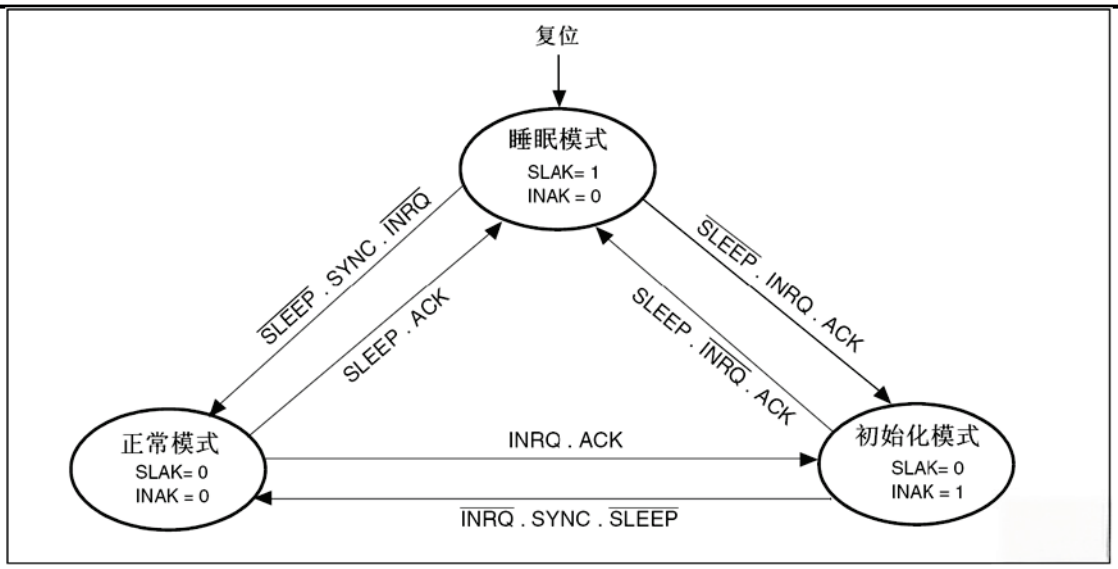


图 197 bxCAN 工作模式

注：　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

22.5 测试模式

通过对 CAN_BTR 寄存器的 SILM 和/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。

22.5.1 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。

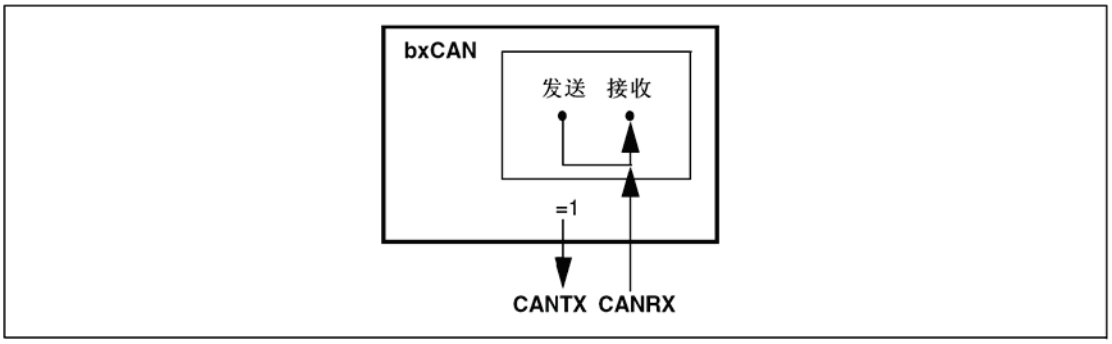


图 198 bxCAN 工作在静默模式

22.5.2 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。

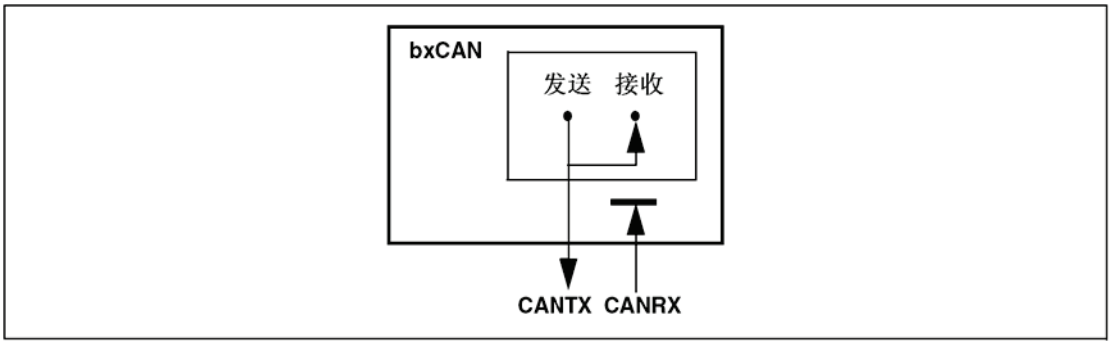


图 199 bxCAN 工作在环回模式

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。

22.5.3 环回静默模式

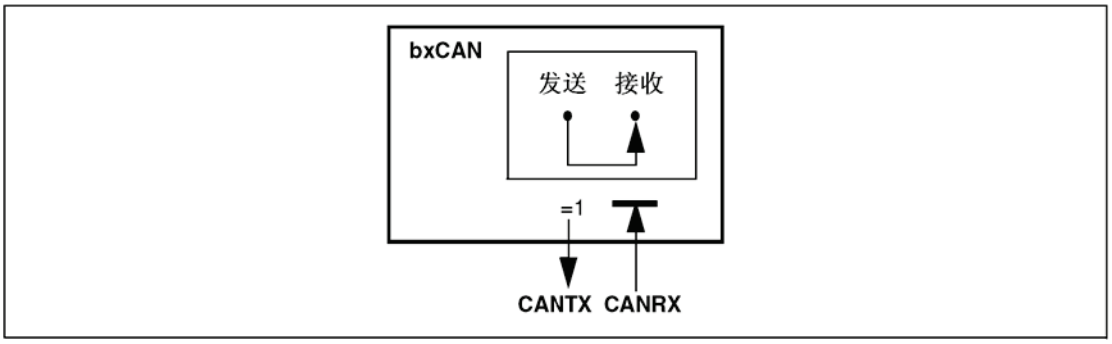


图 200 bxCAN 工作在环回静默模式

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。

22.6 W55MH32 处于调试模式时

当微控制器处于调试模式时，Cortex-M3 核心处于暂停状态，依据下述配置位的状态，bxCAN 可以继续正常工作或停止工作：

- 调试(DBG)模块中 CAN1 的 DBG_CAN1_STOP 位或 CAN2 的 DBG_CAN2_STOP 位。详见第 28.16.2 节：支持定时器、看门狗、bxCAN 和 I2C 的调试。
- CAN_MCR 中的 DBF 位。详见第 22.9.2 节：CAN 控制和状态寄存器。

22.7 bxCAN 功能描述

22.7.1 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。

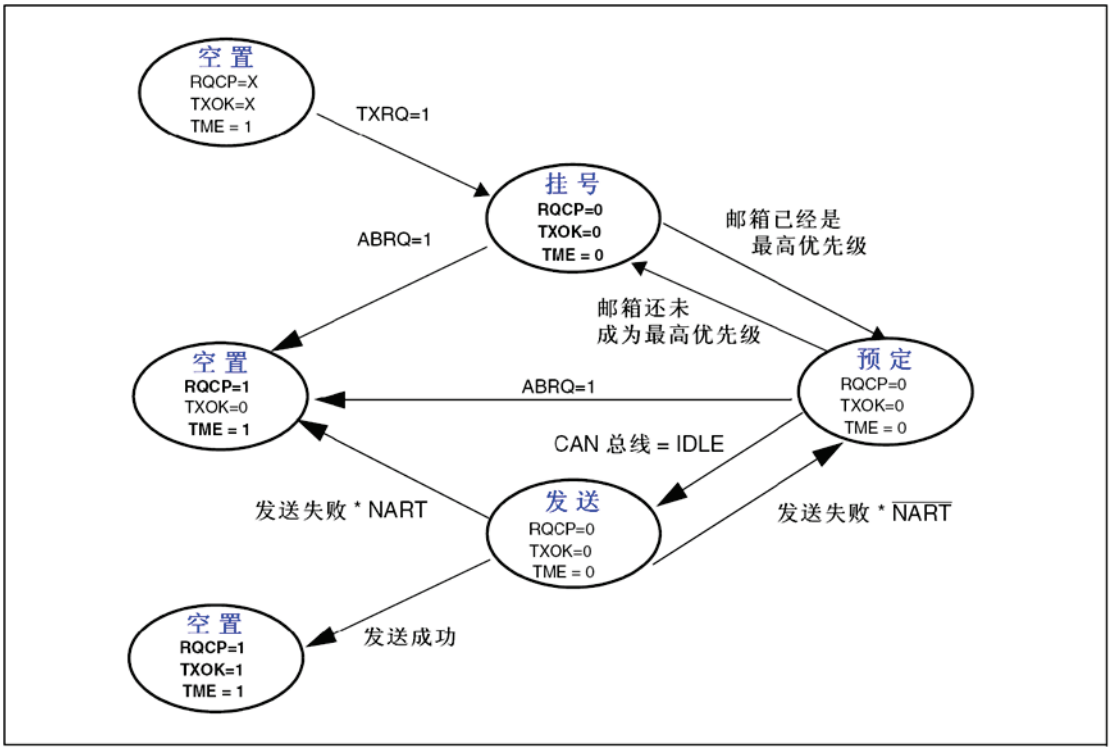


图 201 发送邮箱状态

22.7.2 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(见 22.7.7 节)累加。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

22.7.3 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。请参考 22.7.4 节：标识符过滤。

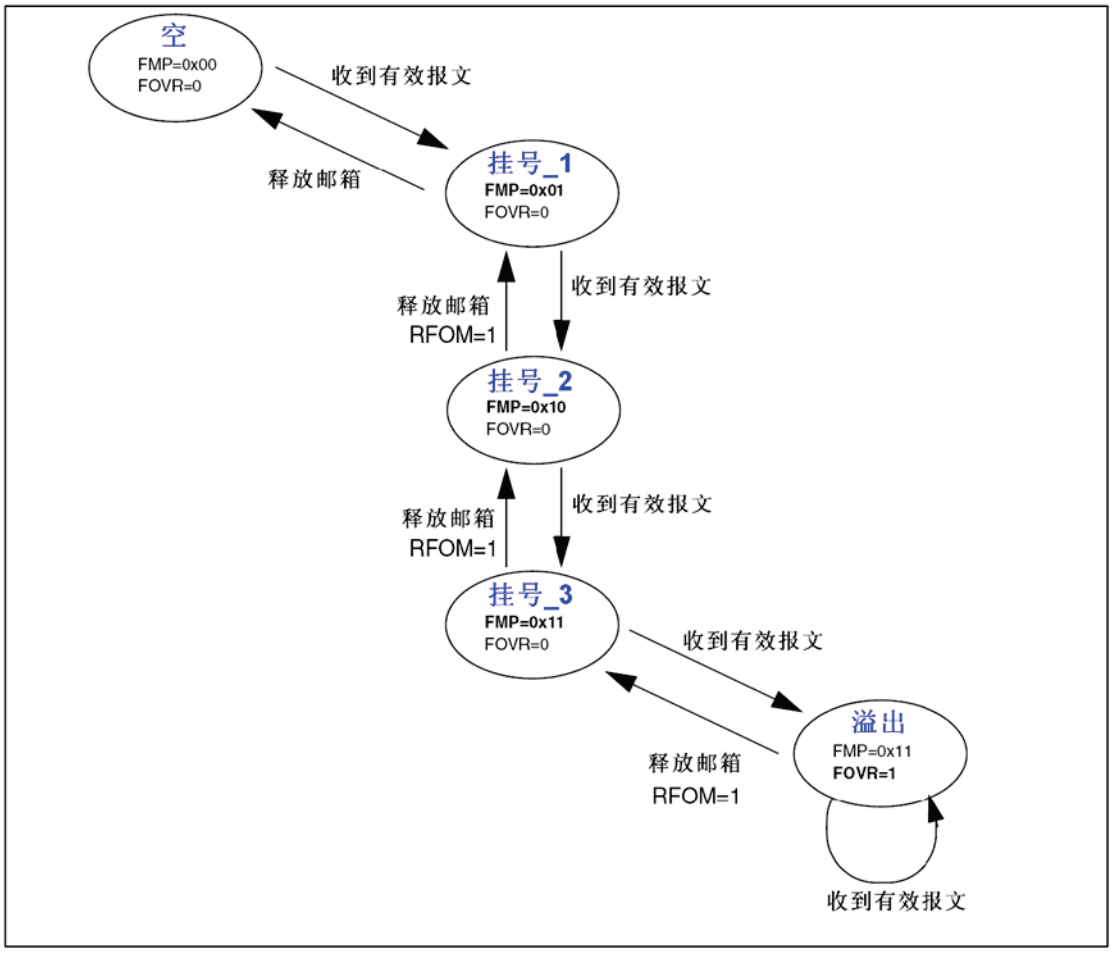


图 202 接收 FIFO 状态

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 22.7.5 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。

- 如果启用了 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。

22.7.4 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_FxR0 和 CAN_FxR1 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：STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位
- 2 个 16 位过滤器，包括：STDID[10:0]、IDE、RTR 和 EXTID[17:15]位可参见图 203。

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，请参见图 203。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。

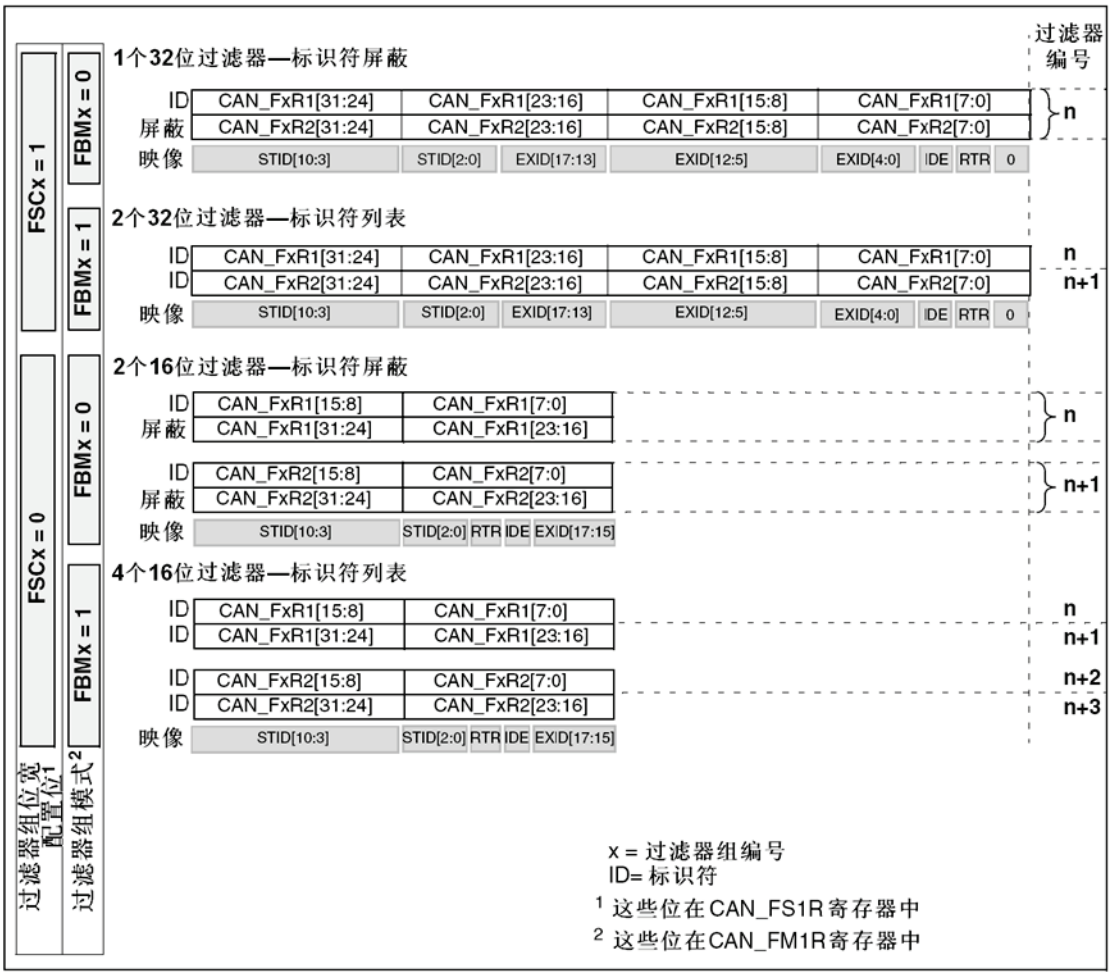


图 203 过滤器组位宽设置 - 寄存器组织

过滤器匹配序号

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到 SRAM 中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。bxCAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作一个索引来访问目标地址

对于标识符列表模式下的过滤器(非屏蔽方式的过滤器)，软件不需要直接跟标识符进行比较。对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位(必须匹配的位)进行比较即可。

在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个 FIFO 各自对其关联的过滤器进行编号。请参考下图的例子。

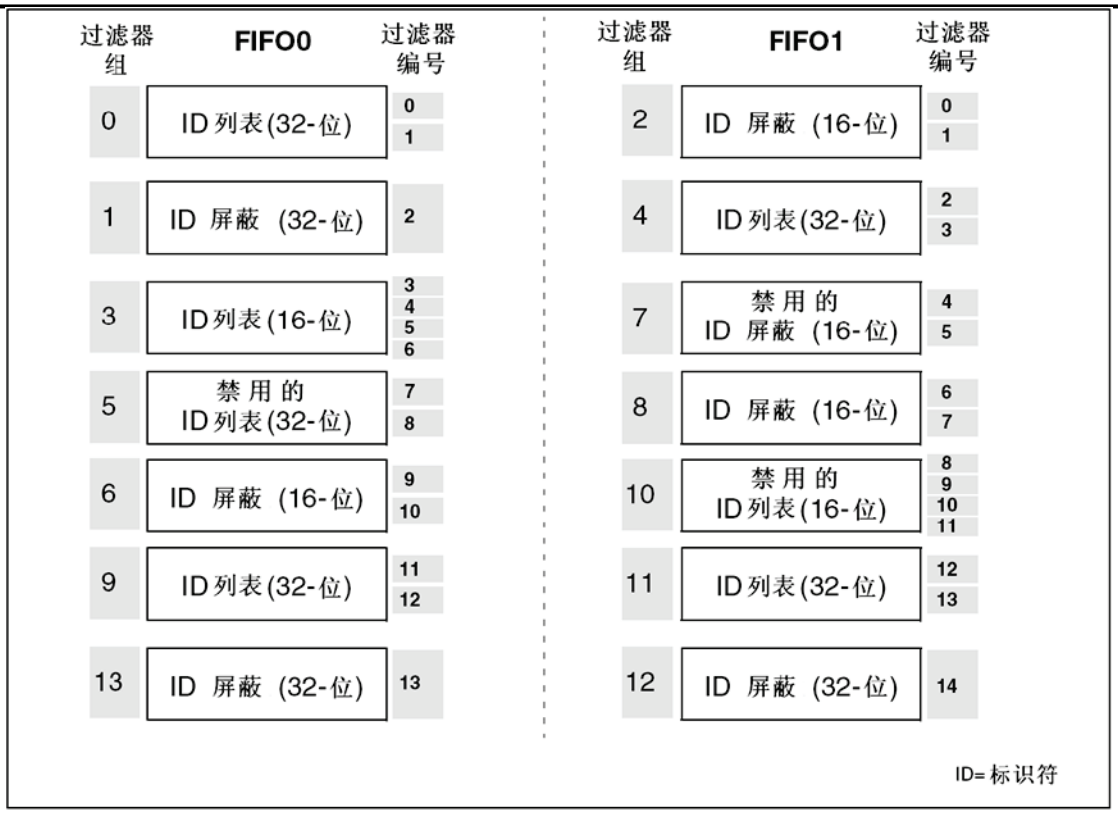


图 204 过滤器编号的例子

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

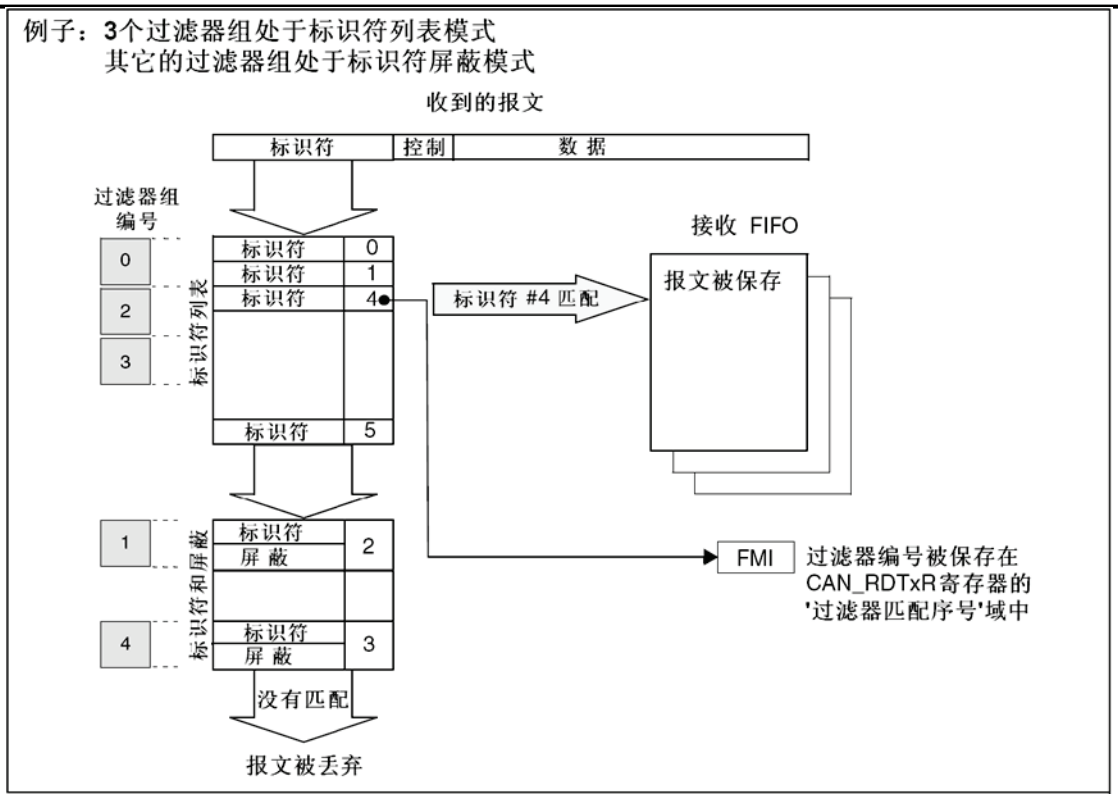


图 205 过滤器机制的例子

上面的例子说明了 bxCAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报文内容和 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。

22.7.5 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 131 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 132 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

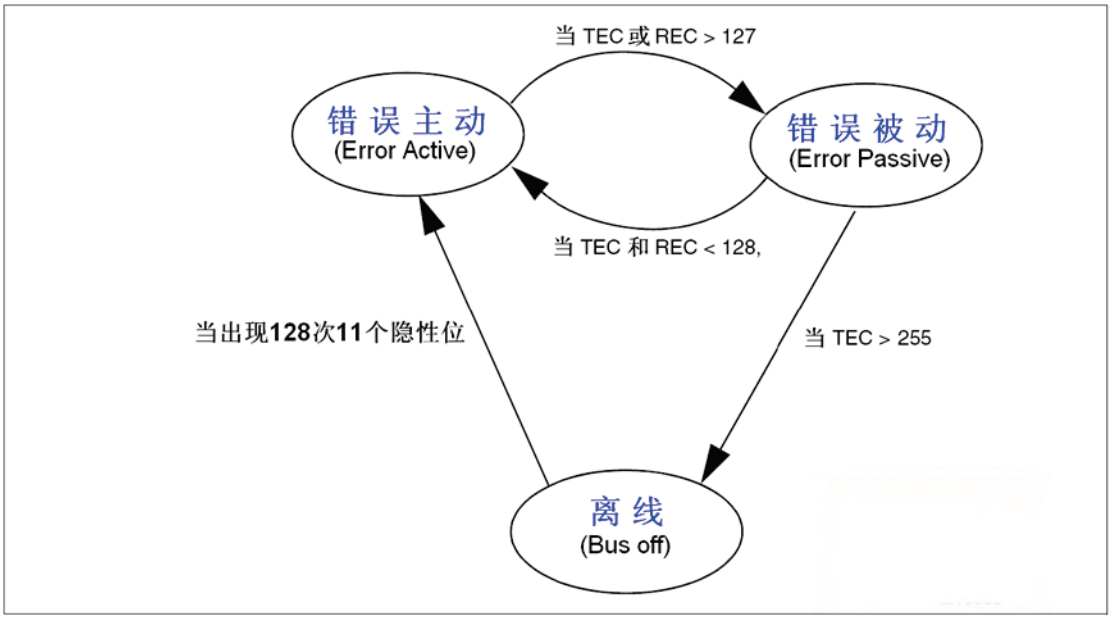


图 206 CAN 错误状态图

22.7.6 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(CANRX 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 CANRX 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。

22.7.7 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，及通过与后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位时间分为 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(1xtCAN)。
- 时间段 1(BS1)：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- 时间段 2(BS2)：定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度(SJW)定义了，在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。

有效跳变被定义为，当 bxCAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。

如果在时间段 1(BS1)而不是在同步段(SYNC_SEG)检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2(BS2)而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器(CAN_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。

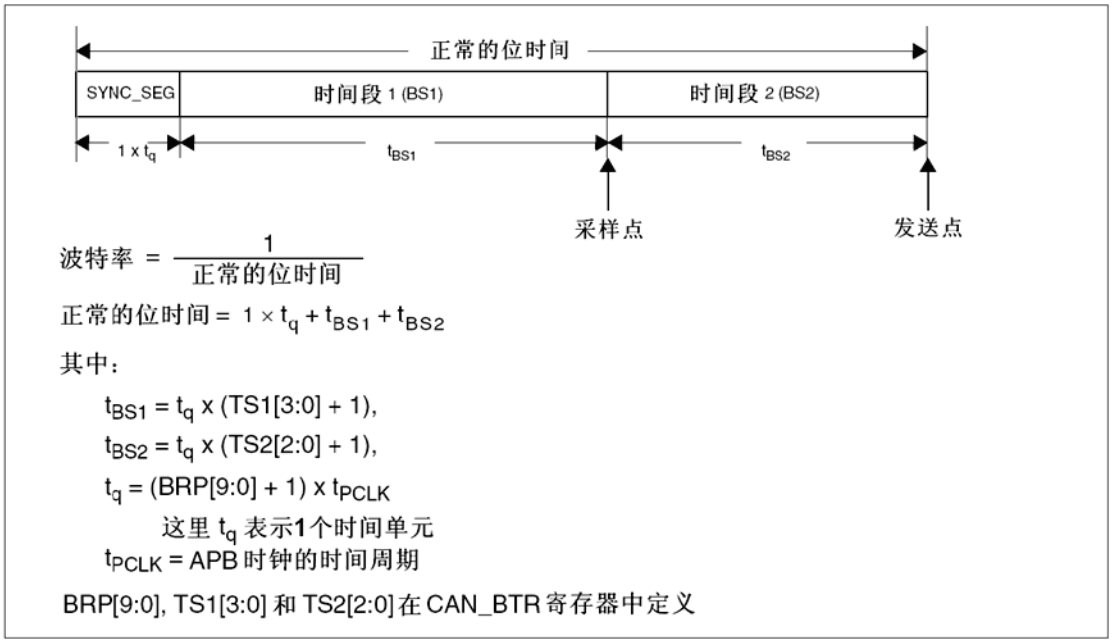


图 207 位时序

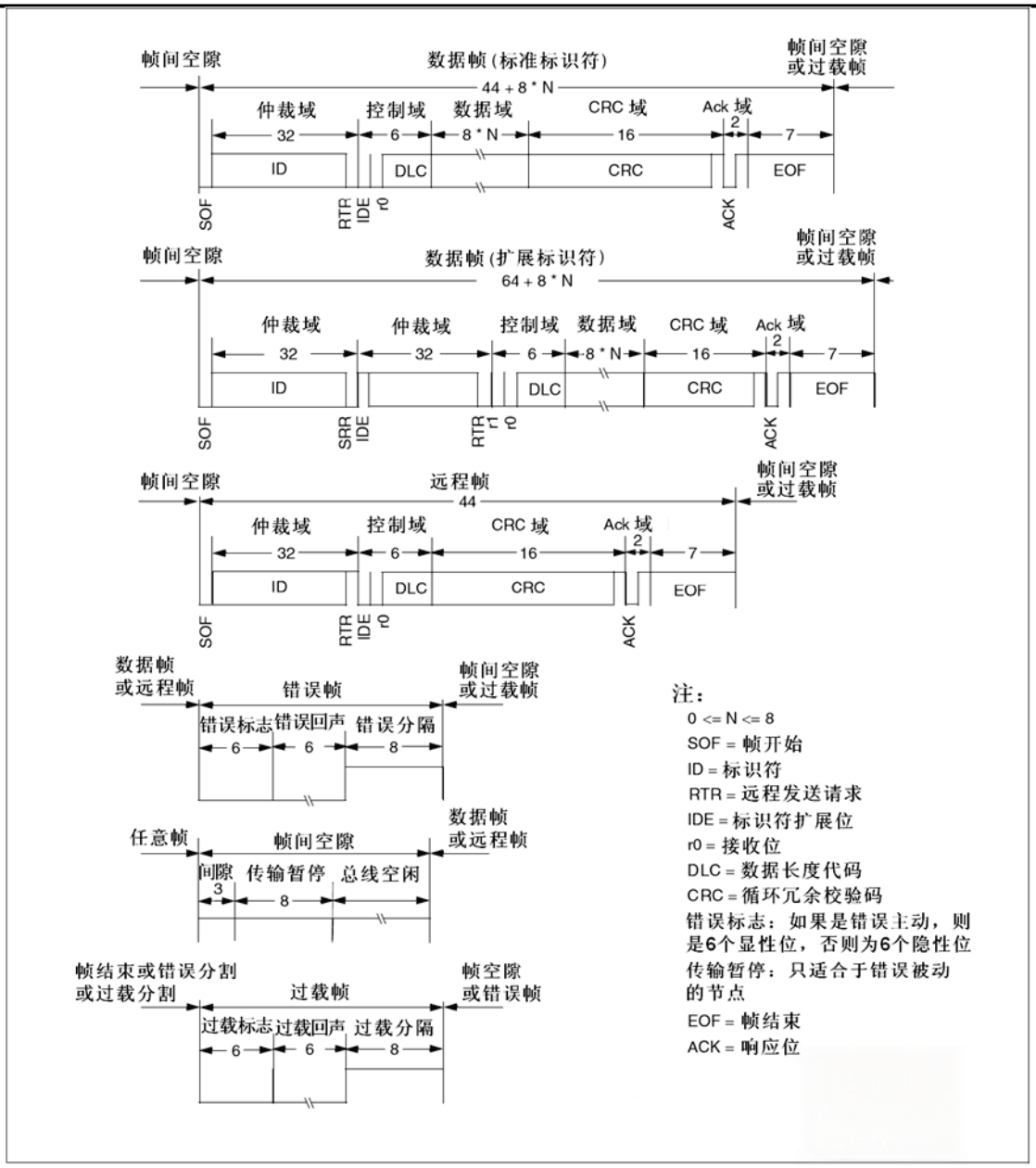

帧间空隙
数据帧(标准标识符)
帧间空隙或过载帧
44 + 8 * N
仲裁域
控制域
数据域
CRC 域
Ack 域
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
帧间空隙
数据帧(扩展标识符)
帧间空隙或过载帧
64 + 8 * N
仲裁域
仲裁域
控制域
数据域
CRC 域
Ack 域
32
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
SRR
IDE
RTR
r1
r0
ACK
帧间空隙
远程帧
帧间空隙或过载帧
44
仲裁域
控制域
CRC 域
Ack 域
32
6
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
数据帧或远程帧
错误帧
帧间空隙或过载帧
错误标志
错误回声
错误分隔
6
6
8
任意帧
帧间空隙
数据帧或远程帧
间隙
传输暂停
总线空闲
3
8
帧结束或错误分割或过载分割
过载帧
帧空隙或错误帧
过载标志
过载回声
过载分隔
6
6
8
注：
0 <= N <= 8
SOF = 帧开始
ID = 标识符
RTR = 远程发送请求
IDE = 标识符扩展位
r0 = 接收位
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 208 各种 CAN 帧

22.8 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。

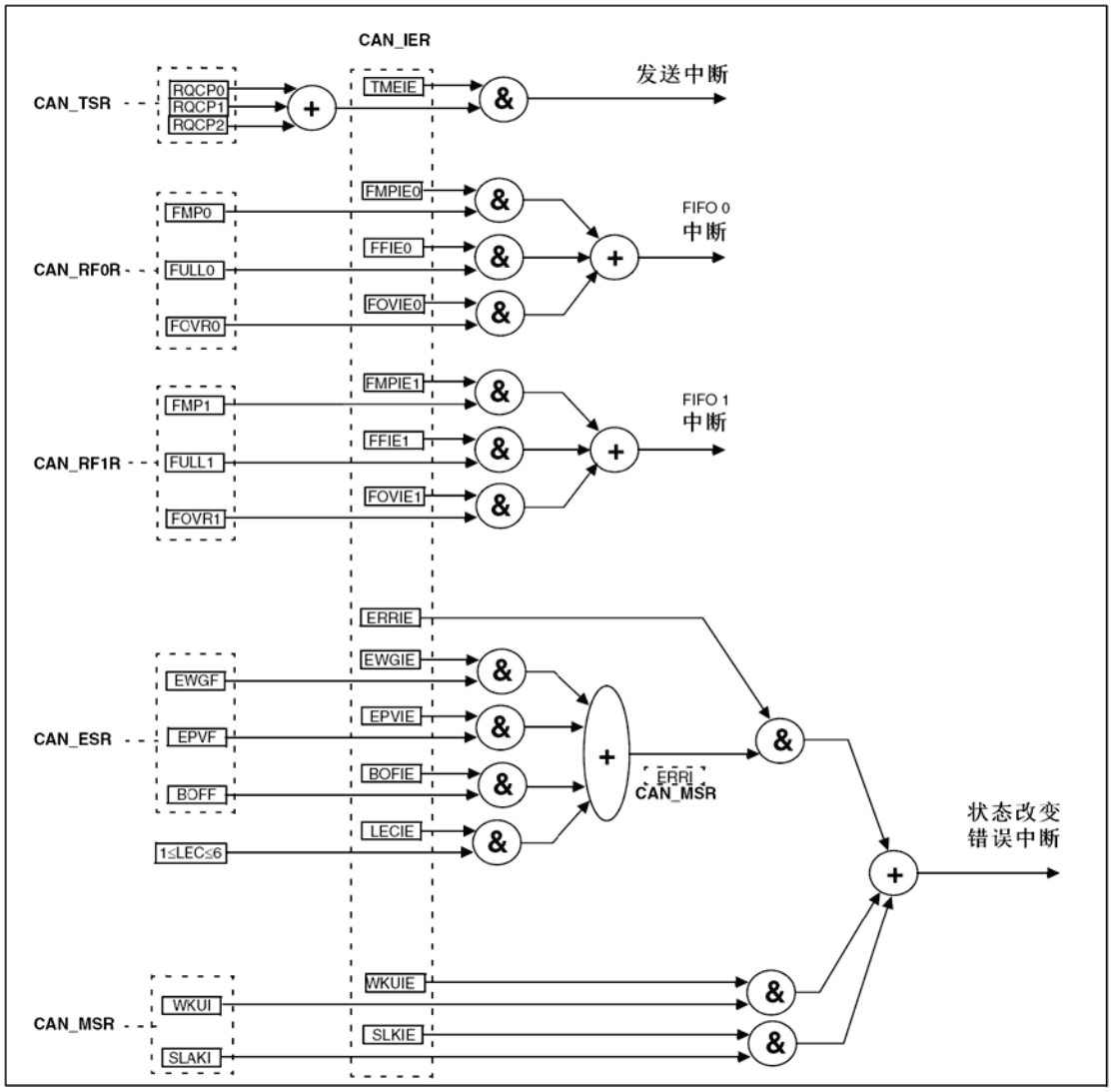


图 209 事件标志和中断产生

- 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。

22.9 CAN 寄存器描述

关于寄存器描述中所用到的缩略词可参见第 1 节。必须以字(32 位)的方式操作这些外设寄存器。

22.9.1 寄存器访问保护

对某些寄存器的错误访问会导致一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，软件只能在 CAN 处于初始化模式时修改 CAN_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，请参见图 201。

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。

22.9.2 CAN 控制和状态寄存器

有关寄存器说明中的缩写，参见 1 节。

CAN 主控制寄存器(CAN_MCR)

地址偏移量:0x00

复位值:0x0001 0002

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:17	Reserved	保留，始终读为 0。
16	DBF	**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。
15	RESET	**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'。
14:8	Reserved	保留，硬件强制为 0。
7	TTCM	**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 22.7.2：时间触发通信模式。
6	ABOM	**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 22.7.6：出错管理。
5	**AWUM**	**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'。
4	**NART**	**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)。
3	**RFLM**	**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。
2	**TXFP**	**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。
1	**SLEEP**	**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 CANRx 信号中检测出 SOF 位时，硬件对该位清'0'。在复位后该位被置'1'，即 CAN 在复位后处于睡眠模式。
0	**INRQ**	**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'。

CAN 主状态寄存器(CAN_MSR)

地址偏移量:0x04

复位值:0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位	符号	说明
31:12	**Reserved**	保留，始终读为 0。
11	**RX**	**RX**：CAN 接收电平(CANRx signal) 该位反映 CAN 接收引脚(CAN_RX)的实际电平。
10	**SAMP**	**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)。
9	**RXM**	**RXM**：接收模式(Receive mode)

		该位为'1'表示 CAN 当前为接收器。
8	**TXM**	**TXM**：发送模式(Transmit mode) 该位为'1'表示 CAN 当前为发送器。
7:5	**Reserved**	保留，始终读为 0。
4	**SLAKI**	**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。
3	**WKUI**	**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'。
2	**ERRI**	**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'。
1	**SLAK**	**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。
0	**INAK**	**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。

CAN 发送状态寄存器(CAN_TSR)

地址偏移量:0x08

复位值:0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	保留			TERR2	ALST2	TXOK2	RQCP2
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	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留			TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

位	符号	说明
31	**LOW2**	**LOW2**:邮箱 2 最低优先级标志(Lowest priority flag for mailbox2) 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'。
30	**LOW1**	**LOW1**:邮箱 1 最低优先级标志(Lowest priority flag for mailbox1) 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'。
29	**LOW0**	**LOW0**:邮箱 0 最低优先级标志(Lowest priority flagfor mailbox0) 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'。
28	**TME2**	**TME2**:发送邮箱 2 空(Transmit mailbox2 empty) 当邮箱 2 中没有等待发送的报文时，硬件对该位置'1'。

27	TME1	**TME1**:发送邮箱 1 空(Transmit mailbox1 empty) 当邮箱 1 中没有等待发送的报文时，硬件对该位置'1'。
26	TME0	**TME0**:发送邮箱 0 空(Transmit mailbox0 empty) 当邮箱 0 中没有等待发送的报文时，硬件对该位置'1'。
25:24	CODE[1:0]	**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。
23	ABRQ2	**ABRQ2**:邮箱 2 中止发送(Abort request for mailbox2) 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。
22:20	Reserved	保留位，硬件强制其值为 0
19	TERR2	**TERR2**:邮箱 2 发送失败(Transmission error of mailbox2)当邮箱 2 因为出错而导致发送失败时，对该位置'1'。
18	ALST2	**ALST2**:邮箱 2 仲裁丢失(Arbitration lost for mailbox2)当邮箱 2 因为仲裁丢失而导致发送失败时，对该位置'1'。
17	TXOK2	**TXOK2**:邮箱 2 发送成功(Transmission OK of mailbox2)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'。请参见图 201。
16	RQCP2	**RQCP2**:邮箱 2 请求完成(Request completed mailbox2) 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'。
15	ABRQ1	**ABRQ1**:邮箱 1 中止发送(Abort request for mailbox1) 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。
14:12	Reserved	保留位，硬件强制其值为 0
11	TERR1	**TERR1**:邮箱 1 发送失败(Transmission error of mailbox1)当邮箱 1 因为出错而导致发送失败时，对该位置'1'。
10	ALST1	**ALST1**:邮箱 1 仲裁丢失(Arbitration lost for mailbox1)当邮箱 1 因为仲裁丢失而导致发送失败时，对该位置'1'。
9	TXOK1	**TXOK1**:邮箱 1 发送成功(Transmission OK of mailbox1)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'。请参见图 201。
8	RQCP1	**RQCP1**:邮箱 1 请求完成(Request completed mailbox1) 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'。
7	ABRQ0	**ABRQ0**:邮箱 0 中止发送(Abort request for mailbox0) 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。
6:4	Reserved	保留位，硬件强制其值为 0
3	TERR0	**TERR0**:邮箱 0 发送失败(Transmission error of mailbox0)当邮箱 0 因为出错而导致发送失败时，对该位置'1'。
2	ALST0	**ALST0**:邮箱 0 仲裁丢失(Arbitration lost for mailbox0)当邮箱 0 因为仲裁丢失而导致发送失败时，对该位置'1'。

1	TXOK0	**TXOK0**:邮箱 0 发送成功(Transmission OK of mailbox0)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'。请参见图 201。
0	RQCP1	**RQCP1**:邮箱 0 请求完成(Request completed mailbox0) 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'。

CAN 接收 FIFO0 寄存器(CAN_RF0R)

地址偏移量:0x0C

复位值:0x00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM0	FOVR0	FULL0	保留	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

位	符号	说明
31:6	Reserved	保留位，硬件强制为 0
5	RFOM0	**RFOM0**:释放接收 FIFO0 输出邮箱(Release FIFO0 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'。
4	FOVR0	**FOVR0**:FIFO0 溢出(FIFO0 overrun) 当 FIFO0 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'。
3	FULL0	**FULL0**:FIFO0 满(FIFO0 full) 当 FIFO0 中有 3 个报文时，硬件对该位置'1'。该位由软件清'0'。
2	Reserved	保留位，硬件强制其值为 0
1:0	FMP0[1:0]	**FMP0[1:0]**:FIFO0 报文数目(FIFO0 messagepending) FIFO0 报文数目这 2 位反映了当前接收 FIFO0 中存放的报文数目。每当 1 个新的报文被存入接收 FIFO0，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。

CAN 接收 FIFO1 寄存器(CAN_RF1R)

地址偏移量:0x10

复位值:0x00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										RFOM1	FOVR1	FULL1	保留	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

位	符号	说明
31:6	Reserved	保留位，硬件强制为 0
5	RFOM1	**RFOM1**:释放接收 FIFO1 输出邮箱(Release FIFO1 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'。
4	**FOVR1**	**FOVR1**:FIFO1 溢出(FIFO1 overrun) 当 FIFO1 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'。
3	**FULL1**	**FULL1**:FIFO1 满(FIFO1 full) 当 FIFO1 中有 3 个报文时，硬件对该位置'1'。该位由软件清'0'。
2	**Reserved**	保留位，硬件强制其值为 0
1:0	**FMP1[1:0]**	**FMP1[1:0]**:FIFO1 报文数目(FIFO1 messagepending) FIFO1 报文数目这 2 位反映了当前接收 FIFO1 中存放的报文数目。每当 1 个新的报文被存入接收 FIFO1，硬件就对 FMP1 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。

CAN 中断使能寄存器(CAN_IER)

地址偏移量:0x14

复位值:0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
rw				rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:18	**Reserved**	保留位，硬件强制为 0
17	**SLKIE**	**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。
16	**WKUIE**	**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。
15	**ERRIE**	**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。
14:12	**Reserved**	保留位，硬件强制为 0。
11	**LECIE**	**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'。
10	**BOFIE**	**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'。
9	**EPVIE**	**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'。
8	**EWGIE**	**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'。
7	**Reserved**	保留位，硬件强制为 0

6	FOVIE1	**FOVIE1**:FIFO1 溢出中断使能(FIFO overrun interrupt enable) 0：当 FIFO1 的 FOVR 位被置'1'时，不产生中断； 1：当 FIFO1 的 FOVR 位被置'1'时，产生中断。
5	FFIE1	**FFIE1**:FIFO1 满中断使能(FIFO full interrupt enable) 0：当 FIFO1 的 FULL 位被置'1'时，不产生中断； 1：当 FIFO1 的 FULL 位被置'1'时，产生中断。
4	FMPIE1	**FMPIE1**:FIFO1 消息挂号中断使能(FIFO messagepending interrupt enable) 0：当 FIFO1 的 FMP[1:0]位为非 0 时，不产生中断； 1：当 FIFO1 的 FMP[1:0]位为非 0 时，产生中断。
3	FOVIE0	**FOVIE0**:FIFO0 溢出中断使能(FIFO overrun interrupt enable) 0：当 FIFO0 的 FOVR 位被置'1'时，不产生中断； 1：当 FIFO0 的 FOVR 位被置'1'时，产生中断。
2	FFIE0	**FFIE0**:FIFO0 满中断使能(FIFO full interrupt enable) 0：当 FIFO0 的 FULL 位被置'1'时，不产生中断； 1：当 FIFO0 的 FULL 位被置'1'时，产生中断。
1	FMPIE0	**FMPIE0**:FIFO0 消息挂号中断使能(FIFO messagepending interrupt enable) 0：当 FIFO0 的 FMP[1:0]位为非 0 时，不产生中断； 1：当 FIFO0 的 FMP[1:0]位为非 0 时，产生中断。
0	TMEIE	**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 22.8 节 bxCAN 中断。

CAN 错误状态寄存器(CAN_ESR)

地址偏移量:0x18

复位值:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									LEC[2:0]			保留	BOFF	EPVF	WEGF
									rw	rw	rw	res	r	r	r

位	符号	说明
31:24	REC[7:0]	**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。
23:16	TEC[7:0]	**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。
15:17	Reserved	保留位，硬件强制为 0。
6:4	LEC[2:0]	**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。

3	Reserved	保留位，硬件强制为 0。
2	BOFF	**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 22.7.6。
1	EPVF	**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)。
0	EWGF	**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)。

CAN 位时序寄存器(CAN_BTR)

地址偏移量:0x1C

复位值:0x0123 0000

注：　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	保留				SJW[1:0]		保留	TS2[2:0]			TS1[3:0]			
rw	rw					rw			rw			rw			

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

位	符号	说明
31	SILM	**SILM**:静默模式(用于调试)(Silent mode(debug)) 0:正常状态; 1:静默模式。
30	LBKM	**LBKM**:环回模式(用于调试)(Loopback mode(debug)) 0:禁止环回模式; 1:允许环回模式。
29:16	Reserved	保留位，硬件强制为 0。
25:24	SJW[1:0]	**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。tRJW=tCANx(SJW[1:0]+1)。
23	Reserved	保留位，硬件强制为 0。
22:20	TS2[2:0]	**TS2[2:0]**:时间段 2(Timesegment2) 该位域定义了时间段 2 占用了多少个时间单元 tBS2=tCANx(TS2[2:0]+1)。
19:16	TS1[3:0]	**TS1[3:0]**:时间段 1(Time segment1) 该位域定义了时间段 1 占用了多少个时间单元 tBS1=tCANx(TS1[3:0]+1) 关于位时间特性的详细信息，请参考 22.7.7 节位时间特性。
15:10	Reserved	保留位，硬件强制其值为 0。
9:0	BRP[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 tq=(BRP[9:0]+1)xtPCLK

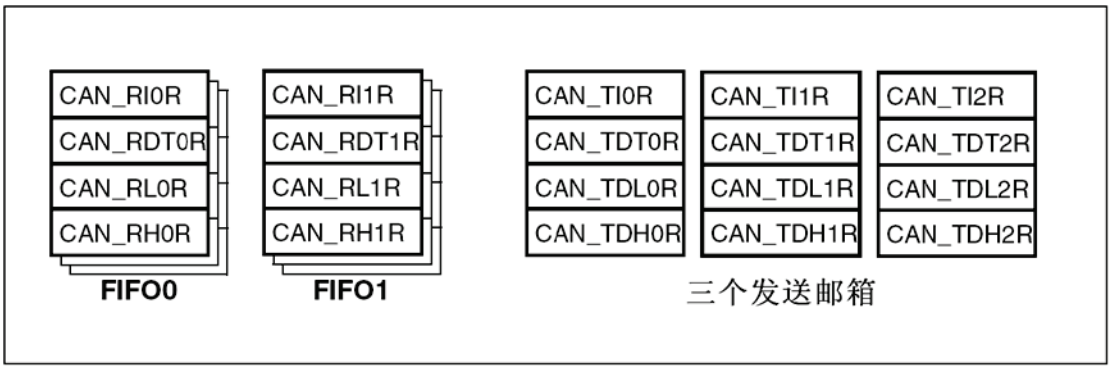
22.9.3 CAN 邮箱寄存器

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考 22.7.5 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。



RX 和 TX

发送邮箱标识符寄存器(CAN_TIxR)(x=0..2)

地址偏移量:0x180，0x190，0x1A0

复位值:0xXXXX XXXX，X=未定义位(除了第 0 位，复位时 TXRQ=0)

注：1 当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的

2 该寄存器实现了发送请求控制功能(第 0 位) - 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

位	符号	说明
31:21	STID[10:0]/ EXID[28:18]	**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。
20:3	EXID[17:0]	**EXID[17:0]**:扩展标识符(Extended identifier)扩展身份标识的低字节。
2	IDE	**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。
1	RTR	**RTR**:远程发送请求(Remote transmission request) 0：数据帧； 1：远程帧。
0	TXRQ	**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'。

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。

地址偏移量：0x184，0x194，0x1A4

复位值：未定义

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
保留							TGT	保留				DLC[3:0]			
							rw					rw	rw	rw	rw

位	符号	说明
31:16	TIME[15:0]	**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。
15:9	Reserved	保留位
8	TGT	**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。
7:4	Reserved	保留位。
3:0	DLC[15:0]	**DLC[15:0]**: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。

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。

地址偏移量：0x188，0x198，0x1A8

复位值：未定义位

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

位	符号	说明
31:24	DATA3[7:0]	**DATA3[7:0]**:数据字节 3(Data byte3)报文的数据字节 3。
23:16	DATA2[7:0]	**DATA2[7:0]**:数据字节 2(Data byte2)报文的数据字节 2。
15:8	DATA1[7:0]	**DATA1[7:0]**:数据字节 1(Data byte1)报文的数据字节 1。
7:0	DATA0[7:0]	**DATA0[7:0]**:数据字节 0(Data byte0)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。

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。

地址偏移量：0x18C，0x19C，0x1AC

复位值：未定义位

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

位	符号	说明

31:24	**DATA7[7:0]**	**DATA7[7:0]**:数据字节 7(Data byte7)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。
23:16	**DATA6[7:0]**	**DATA6[7:0]**:数据字节 6(Data byte6)报文的数据字节 6。
15:8	**DATA5[7:0]**	**DATA5[7:0]**:数据字节 5(Data byte5)报文的数据字节 5。
7:0	**DATA4[7:0]**	**DATA4[7:0]**:数据字节 4(Data byte4)报文的数据字节 4。

接收 FIFO 邮箱标识符寄存器(CAN_RIxR)(x=0..1)

地址偏移量：0x1B0，0x1C0

复位值：未定义位

注：　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	保留
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

位	符号	说明
32:21	**STID[10:0]/ EXID[28:18]**	**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。
20:3	**EXID[17:0]**	**EXID[17:0]**:扩展标识符(Extended identifier)扩展标识符的低字节。
2	**IDE**	**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。
1	**RTR**	**RTR**:远程发送请求(Remote transmission request) 0：数据帧; 1：远程帧。
0	**Reserved**	保留位。

接收 FIFO 邮箱数据长度和时间戳寄存器(CAN_RDTxR)(x=0..1)

地址偏移量：0x1B4，0x1C4

复位值：未定义

注：　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]								保留				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

位	符号	说明
31:16	**TIME[15:0]**	**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。
15:8	**FMI[15:0]**	**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。关于标识符过滤的细节，请参考 22.7.4 节中有关过滤器匹配序号。
7:4	**Reserved**	保留位，硬件强制为 0。
3:0	**DLC[15:0]**	**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。

接收 FIFO 邮箱低字节数据寄存器(CAN_RDLxR)(x=0..1)

地址偏移量：0x1B8，0x1C8

复位值：未定义位

注：　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

位	符号	说明
31:24	**DATA3[7:0]**	**DATA3[7:0]**:数据字节 3(Data byte3)报文的数据字节 3。
23:16	**DATA2[7:0]**	**DATA2[7:0]**:数据字节 2(Data byte2)报文的数据字节 2。
15:8	**DATA1[7:0]**	**DATA1[7:0]**:数据字节 1(Data byte1)报文的数据字节 1。
7:0	**DATA0[7:0]**	**DATA0[7:0]**:数据字节 0(Data byte0)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。

接收 FIFO 邮箱高字节数据寄存器(CAN_RDHxR)(x=0..1)

地址偏移量：0x1BC，0x1CC

复位值：未定义位

注：　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

位	符号	说明
31:24	**DATA7[7:0]**	**DATA7[7:0]**:数据字节 7(Data byte7)报文的数据字节 7
23:16	**DATA6[7:0]**	**DATA6[7:0]**:数据字节 6(Data byte6)报文的数据字节 6。
15:8	**DATA5[7:0]**	**DATA5[7:0]**:数据字节 5(Data byte5)报文的数据字节 5。
7:0	**DATA4[7:0]**	**DATA4[7:0]**:数据字节 4(Data byte4)报文的数据字节 4。

22.9.4 CAN 过滤器寄存器

CAN 过滤器主控寄存器(CAN_FMR)

地址偏移量:0x200

复位值:0x2A1C 0E01

注：　该寄存器的非保留位完全由软件控制。

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

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

位	符号	说明
31:1	Reserved	保留位，强制为复位值。
0	FINIT	**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。

CAN 过滤器模式寄存器(CAN_FM1R)

地址偏移量:0x204

复位值:0x0000 0000

注：　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。

注：　请参考图 203 过滤器组位宽设置 - 寄存器组织

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

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

位	符号	说明
31:14	Reserved	保留位，强制为复位值。
13:0	FBMx	**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。

CAN 过滤器位宽寄存器(CAN_FS1R)

地址偏移量:0x20C

复位值:0x0000 0000

注：　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
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		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：　请参考图 203 过滤器组位宽设置 - 寄存器组织

位	符号	说明
31:14	Reserved	保留位，强制为复位值。
13:0	FSCx	**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。

CAN 过滤器 FIFO 关联寄存器(CAN_FFA1R)

地址偏移量:0x214

复位值:0x0000 0000

注：　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
保留															

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

位	符号	说明
31:14	Reserved	保留位，强制为复位值。
13:0	FFAx	**FFAx**:过滤器位宽设置(Filter FIFO assignment for filterx) 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。

CAN 过滤器激活寄存器(CAN_FA1R)

地址偏移量:0x21C

复位值:0x0000 0000

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

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

位	符号	说明
31:14	Reserved	保留位，强制为复位值。
13:0	FFACTx	**FFACTx**: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 FINI 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。

CAN 过滤器组 i 的寄存器 x（CAN_FiRx）(i=0..27,x=1,2)

地址偏移量：0x240h..0x31Ch

复位值：未定义位

注：　W55MH32 共有 14 组过滤器：i=0..13。每组过滤器由 2 个 32 位的寄存器，CAN_FiR[2:1]组成。

只有在 CAN_FAxR 寄存器相应的 FACTx 位清'0'，或 CAN_FMR 寄存器的 FINIT 位为'1'时，才能修改相应的过滤器寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	FB5	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：

位	符号	说明
31:0	FB[31:0]	**FB[31:0]**：过滤器位(Filter bits) **标识符模式** 寄存器的每位对应于所期望的标识符的相应位的电平。 0:期望相应位为显性位；1:期望相应位为隐性位。 **屏蔽位模式**

		寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。0: 不关心，该位不用于比较; 1:必须匹配，到来的标识符位必须与滤波器对应的标识符寄存器位相一致。

注：　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 22.7.4 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。关于过滤器组寄存器的地址，请参见表 133。

22.9.5　bxCAN 寄存器列表

表 133 bxCAN - 寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	CAN_MCR	保留															DBF	RESET	保留							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	复位值																1	0								0	0	0	0	0	0	1	0
004h	CAN_MSR	保留																				RX	SAMP	RXM	TXM	保留			SLAKI	WKUI	ERRI	SLAK	INAK
	复位值																					1	1	0	0				0	0	0	1	0
008h	CAN_TSR	LOW [2:0]			TME [2:0]			CODE [1:0]		ABRQ2	保留			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	保留			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	保留		TERR0	ALST0	TXOK0	RQCP0
	复位值	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0			0	0	0	0
00Ch	CAN_RFOR	保留																										RFOM0	FOVR0	FULL0	保留	FMP0 [1:0]	
	复位值																											0	0	0		0	0
010h	CAN_RF1F	保留																										RFOM1	FOVR1	FULL1	保留	FMP1 [1:0]	
	复位值																											0	0	0		0	0
014h	CAN_IER	保留														SLKIE	WKUE	ERRIE	保留			LECIE	BOFIE	EPVIE	EWGIE	保留	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	复位值															0	0	0				0	0	0	0		0	0	0	0	0	0	0
018h	CAN_ESR	REC[7:0]								TEC[7:0]									保留							LEC[2:0]			保留	BOFF	EPVF	EWGF	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0	
01Ch	CAN_BTR	SILM	LBKM	保留						SJW [1:0]		保留	TS2[2:0]			TS1[3:0]			保留							BRP[9:0]							
	复位值	0	0							0	0		0	1	0	0	0	1								0	0	0	0	0	0	0	0	0
020h~ 17Fh	保留																																
180h	CAN_TIOR	STID[10:0]/EXID[28:18]												EXID[17:0]																	IDE	RTR	TXRQ

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
184h	CAN_TDTOR	TIME[15:0]																保留							TGT	保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
188h	CAN_TDLOR	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
18Ch	CAN_TDHOR	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
190h	CAN_TI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
194h	CAN_TDT1R	TIME[15:0]																保留							TGT	保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
198h	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
19Ch	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1A0h	CAN_TI2R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
1A4h	CAN_TDT2R	TIME[15:0]																保留							TGT	保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
1A8h	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1ACh	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B0h	CAN_RIOR	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B4h	CAN_RDTOR	TIME[15:0]																FMI[7:0]								保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
1B8h	CAN_RDLOR	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1BCh	CAN_RDHOR	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C0h	CAN_RI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	保留
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C4h	CAN_RDT1R	TIME[15:0]																FMI[7:0]								保留				DLC[3:0]			
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
1C8h	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1CCh	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1D0h~1FFh	保留																																
200h	CAN_FMR	保留																															FINIT
	复位值																																1
204h	CAN_FM1R	保留																															
	复位值	FBM[27:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
208h	保留																																
20Ch	CAN_FS1R	保留																															
	复位值	FSC[27:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
210h	保留																																
214h	CAN_FFA1R	保留																															
	复位值	FFA[27:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
218h	保留																																
21Ch	CAN_FA1R	保留																															
	复位值	FACT[27:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220h	保留																																
224h~23Fh	保留																																
240h	CAN_FOR1	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
244h	CAN_FOR2	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
248h	CAN_F1R1	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
24Ch	CAN_F1R2	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . .	. . . .	. . . .																															
318h	CAN_F27R1	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
31Ch	CAN_F27R2	FB[31:0]																															
	复位值	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

关于寄存器的起始地址，参见表 1。

23 串行外设接口(SPI)

23.1 SPI 简介

SPI 接口可以配置为支持 SPI 协议或者支持 I2S 音频协议。SPI 接口默认工作在 SPI 方式，可以通过软件把功能从 SPI 模式切换到 I2S 模式。

串行外设接口(SPI)允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

它可用于多种用途，包括使用一条双向数据线的双线单工同步传输，还可使用 CRC 校验的可靠通信。

I2S 也是一种 3 引脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦 I2S 标准，MSB 和 LSB 对齐标准，以及 PCM 标准。它在半双工通讯中，可以工作在主和从 2 种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

警告：由于 SPI3/I2S3 的部分引脚与 JTAG 引脚共享(SPI3_NSS/I2S3_WS 与 JTDI，SPI3_SCK/I2S3_CK 与 JTDO)，因此这些引脚不受 IO 控制器控制，他们(在每次复位后)被默认保留为 JTAG 用途。如果用户想把引脚配置给 SPI3/I2S3，必须(在调试时)关闭 JTAG 并切换至 SWD 接口，或者(在标准应用时)同时关闭 JTAG 和 SWD 接口。详见第 7.3.4 节：JTAG/SWD 复用功能重映射。

23.2 SPI 和 I²S 主要特征

23.2.1 SPI 特征

- 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 fPCLK/2)
- 从模式频率(最大为 fPCLK/2)
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持可靠通信的硬件 CRC
  - 在发送模式下，CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 可触发中断的主模式故障、过载以及 CRC 错误标志
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接受请求

23.2.2 I²S 功能

- 单工通信(仅发送或接收)
- 主或者从操作
- 8 位线性可编程预分频器，获得精确的音频采样频率(8KHz 到 96kHz)
- 数据格式可以是 16 位，24 位或者 32 位
- 音频信道固定数据包帧为 16 位(16 位数据帧)或 32 位(16、24 或 32 位数据帧)
- 可编程的时钟极性(稳定态)
- 从发送模式下的下溢标志位和主/从接收模式下的溢出标志位
- 16 位数据寄存器用来发送和接收，在通道两端各有一个寄存器
- 支持的 I2S 协议：
  - I2S 飞利浦标准
  - MSB 对齐标准(左对齐)
  - LSB 对齐标准(右对齐)
  - PCM 标准(16 位通道帧上带长或短帧同步或者 16 位数据帧扩展为 32 位通道帧)
- 数据方向总是 MSB 在先
- 发送和接收都具有 DMA 能力
- 主时钟可以输出到外部音频设备，比率固定为 256xFs(Fs 为音频采样频率)

23.3 SPI 功能描述

23.3.1 概述

SPI 的方框图见下图。

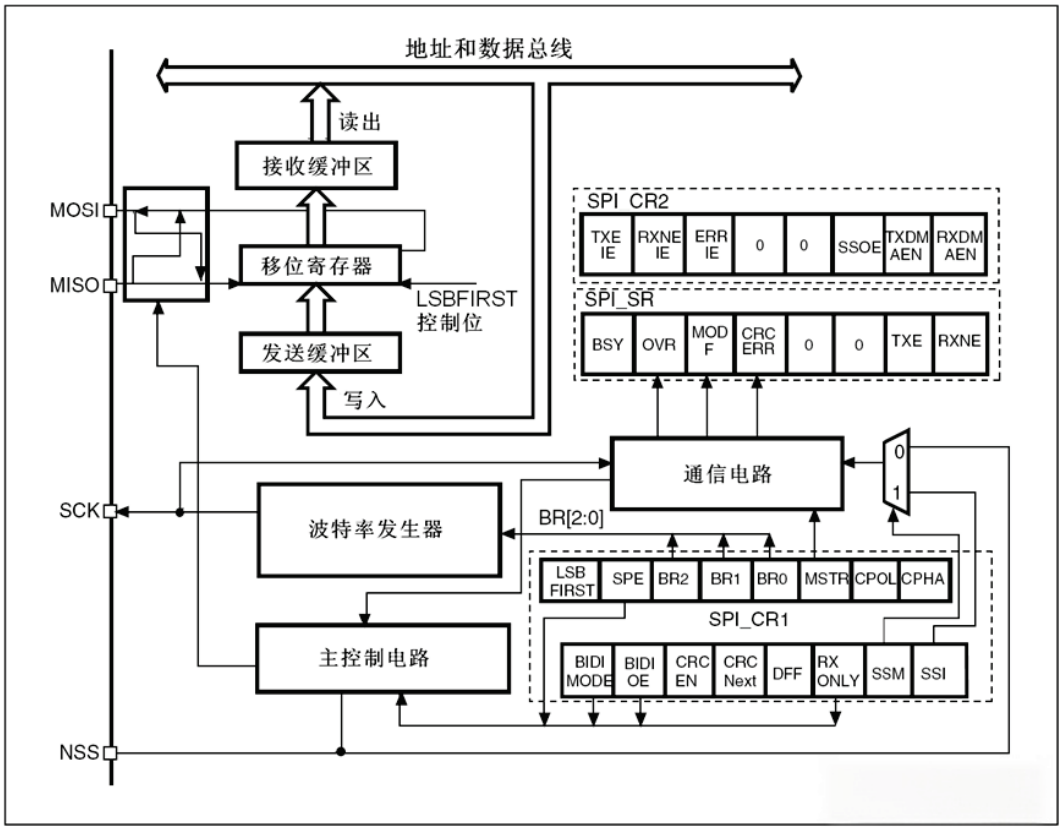


图 210 SPI 框图

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(参见)

下图是一个单主和单从设备互连的例子。

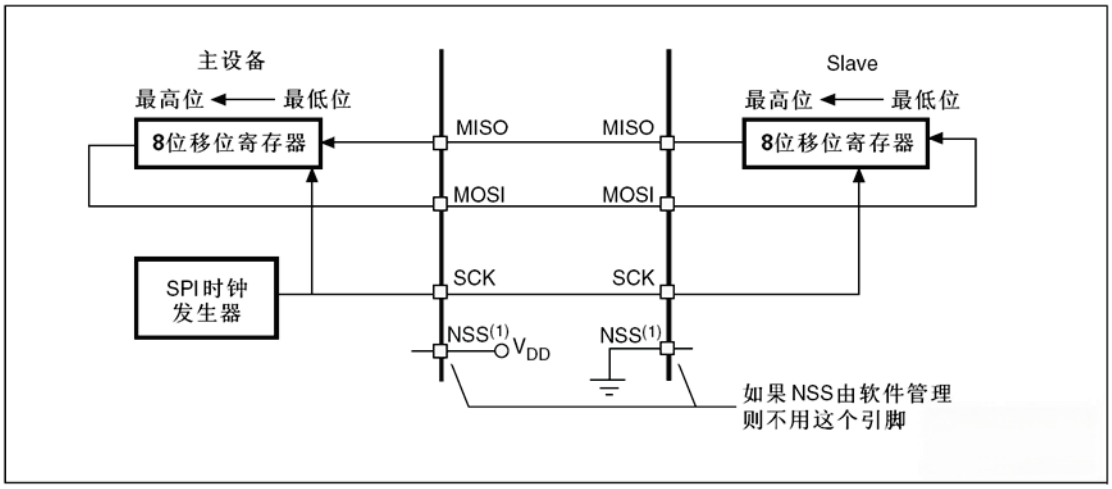


图 211 单主和单从应用

1.这里 NSS 引脚设置为输入

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输(MSB 位在前)。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

从选择(NSS)脚管理

有 2 种 NSS 模式：

- 软件 NSS 模式：可以通过设置 SPI_CR1 寄存器的 SSM 位来使能这种模式(见图 212)。在这种模式下 NSS 引脚可以用作它用，而内部 NSS 信号电平可以通过写 SPI_CR1 的 SSI 位来驱动
- 硬件 NSS 模式，分两种情况：
  - NSS 输出被使能：当 W55MH32 工作为主 SPI，并且 NSS 输出已经通过 SPI_CR2 寄存器的 SSOE 位使能，这时 NSS 引脚被拉低，所有 NSS 引脚与这个主 SPI 的 NSS 引脚相连并配置为硬件 NSS 的 SPI 设备，将自动变成从 SPI 设备。

    当一个 SPI 设备需要发送广播数据，它必须拉低 NSS 信号，以通知所有其它的设备它是主设备；如果它不能拉低 NSS，这意味着总线上有另外一个主设备在通信，这时将产生一个硬件失败错误(Hard Fault)。
  - NSS 输出被关闭：允许操作于多主环境。

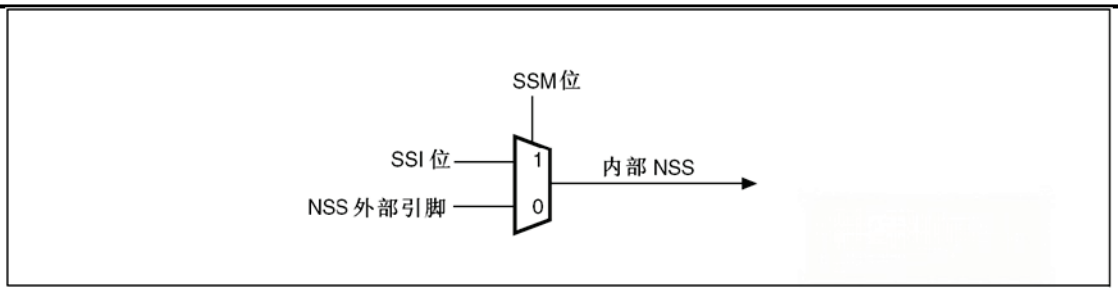


图 212 硬件/软件的从选择管理

时钟信号的相位和极性

SPI_CR 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL(时钟极性)位控制在没有数据传输时时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清'0'，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置'1'，SCK 引脚在空闲状态保持高电平。如果 CPHA(时钟相位)位被置'1'，SCK 时钟的第二个边沿(CPOL 位为 0 时就是下降沿，CPOL 位为'1'时就是上升沿)进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清'0'，SCK 时钟的第一边沿(CPOL 位为'0'时就是上升沿，CPOL 位为'1'时就是下降沿)进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。

图 213 显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

注意： *1.在改变 CPOL/CPHA 位之前，必须清除 SPE 位将 SPI 禁止。*

2.主和从必须配置成相同的时序模式。

3.SCK 的空闲状态必须和 SPI_CR1 寄存器指定的极性一致(CPOL 为'1'时，空闲时应上拉 SCK 为高电平；CPOL 为'0'时，空闲时应下拉 SCK 为低电平)。

4.数据帧格式(8 位或 16 位)由 SPI_CR1 寄存器的 DFF 位选择，并且决定发送/ 接收的数据长度。

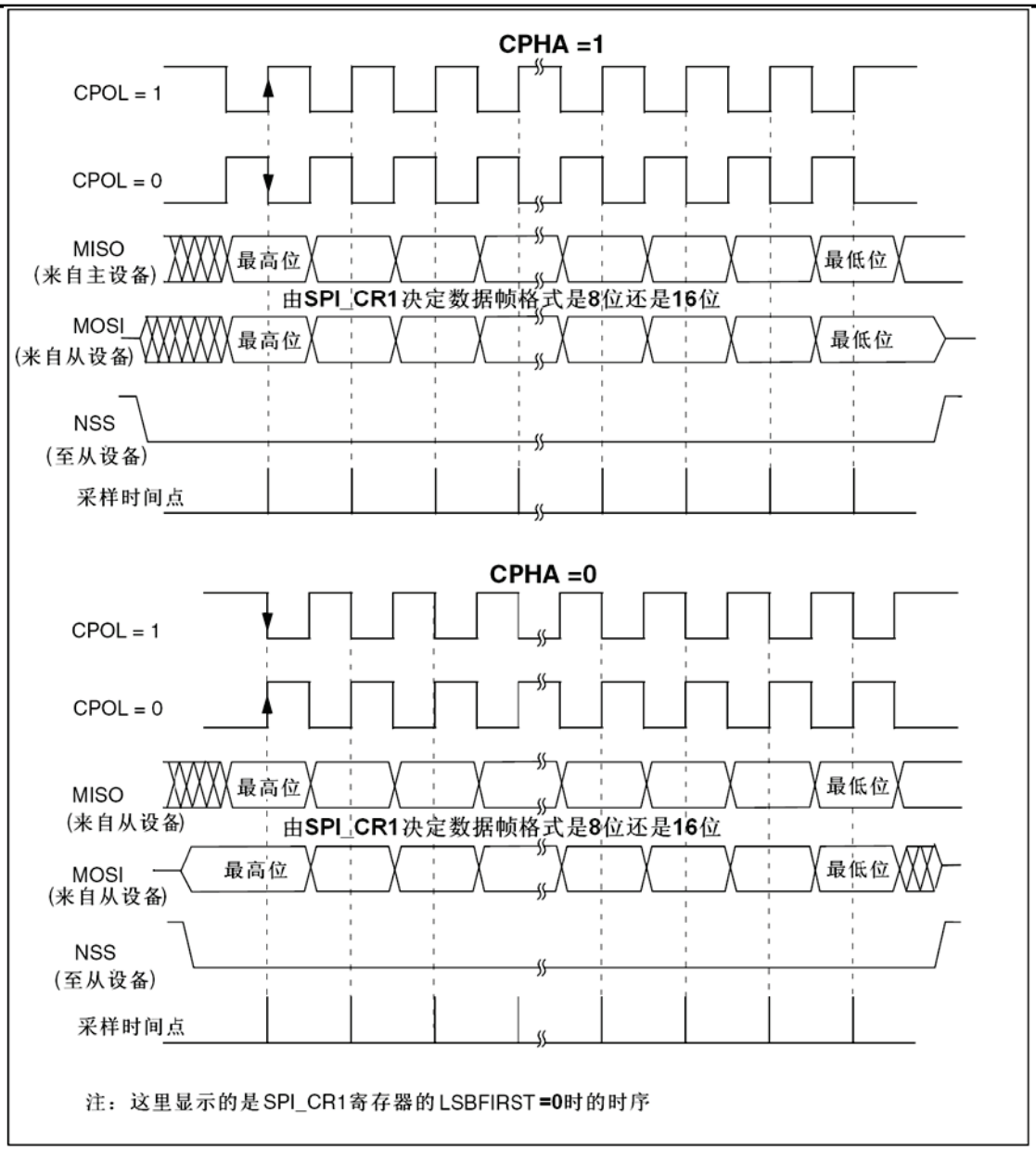


图 213 数据时钟时序图

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。

23.3.2 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(见图 213)。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(注：此时第一个位被发送出去)。余下的位(对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有 15 位)被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI_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。

23.3.3 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(见图 213)。
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。

23.3.4 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 位而启用此模式。在这个模式下，SCK 引脚作为时钟，主设备使用 MOSI 引脚而从设备使用 MISO 引脚作为数据通信。传输的方向由 SPI_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。

23.3.5 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(见图 214 和图 215)：

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。

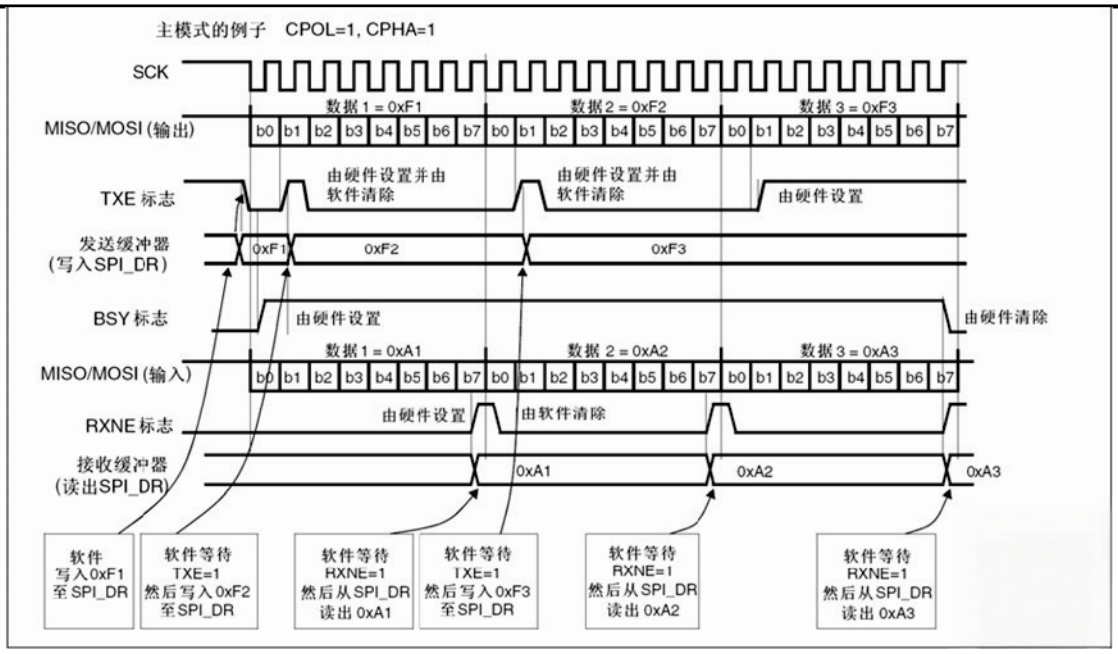


图 214 主模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图

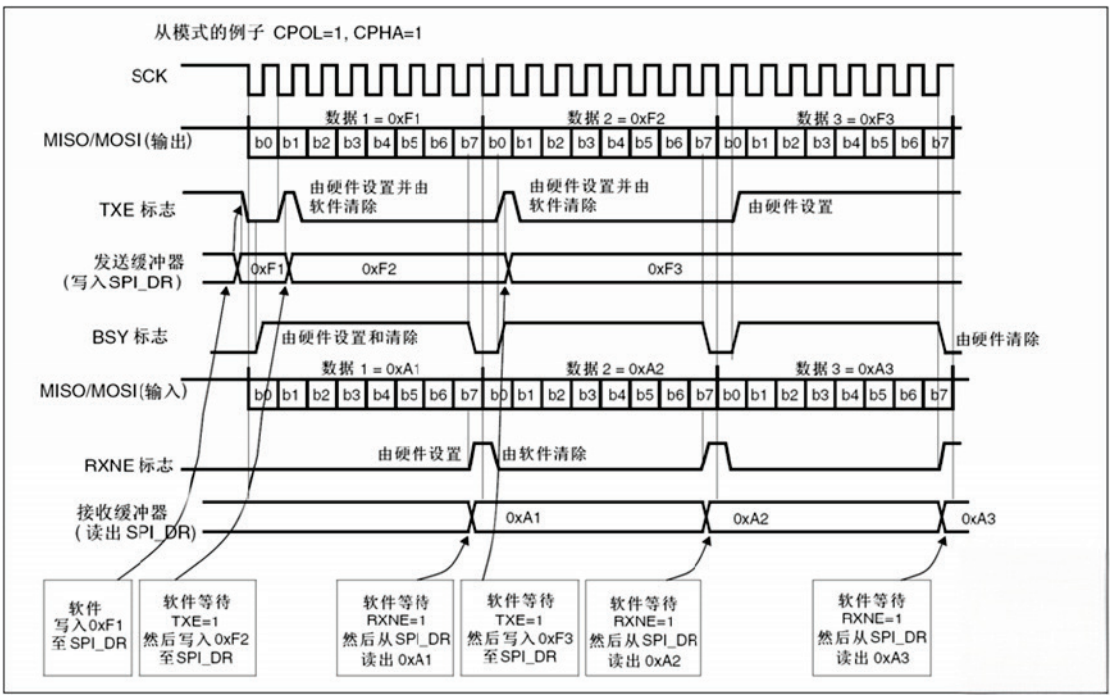


图 215 从模式、全双工模式下(BIDIMODE=0 并且 RXONLY=0)连续传输时，TXE/RXNE/BSY 的变化示意图

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(见图 216 和图 217)：

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： *1.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。*

2．只发送模式下，在传输 2 个数据之后，由于不会读出接收到的数据，SPI_SR 寄存器中的 OVR 位会变为1'。软件不必理会这个 OVR 标志位。

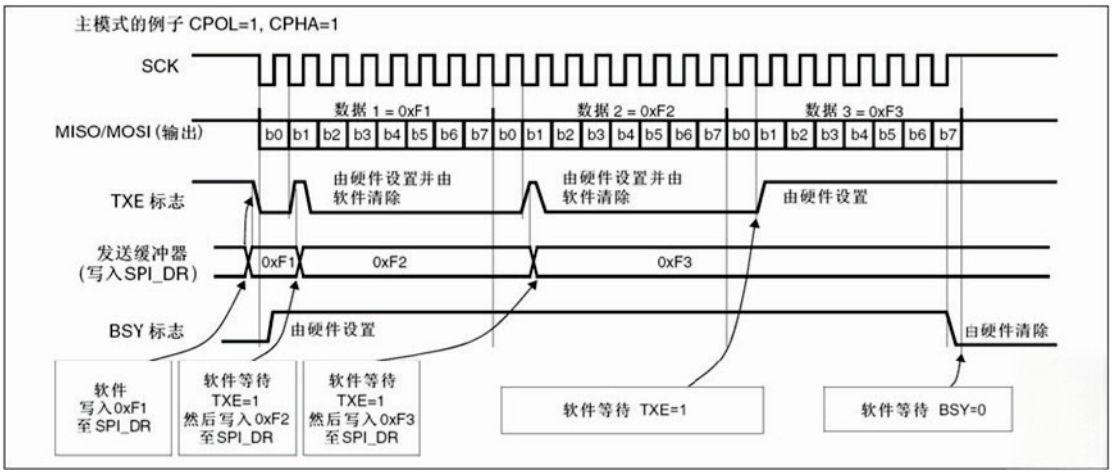


图 216 主设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，，TXE/BSY 变化示意图

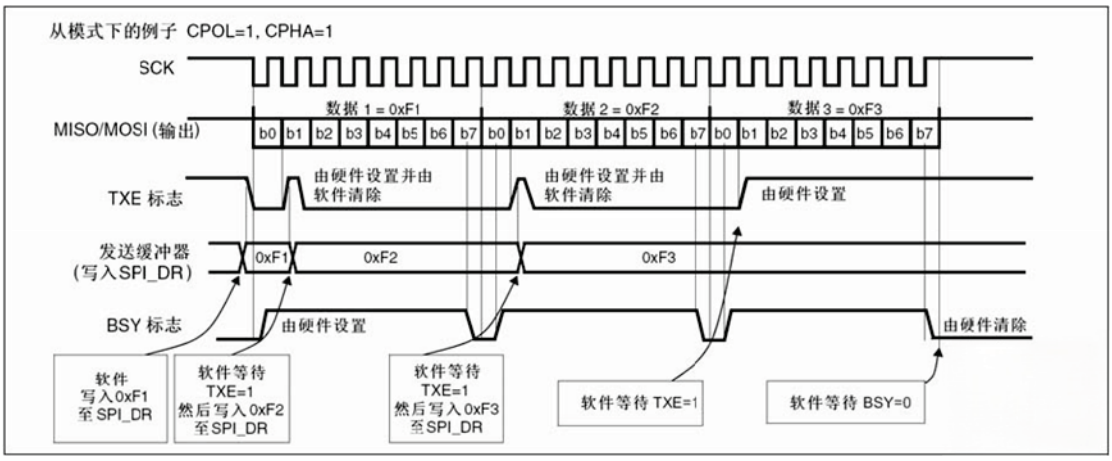


图 217 从设备只发送模式(BIDIMODE=0 并且 RXONLY=0)下连续传输时，TXE/BSY 变化示意图

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_CR2 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(见)：

1. 在 SPI_CR2 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，请按照第 23.3.8 节的建议操作。

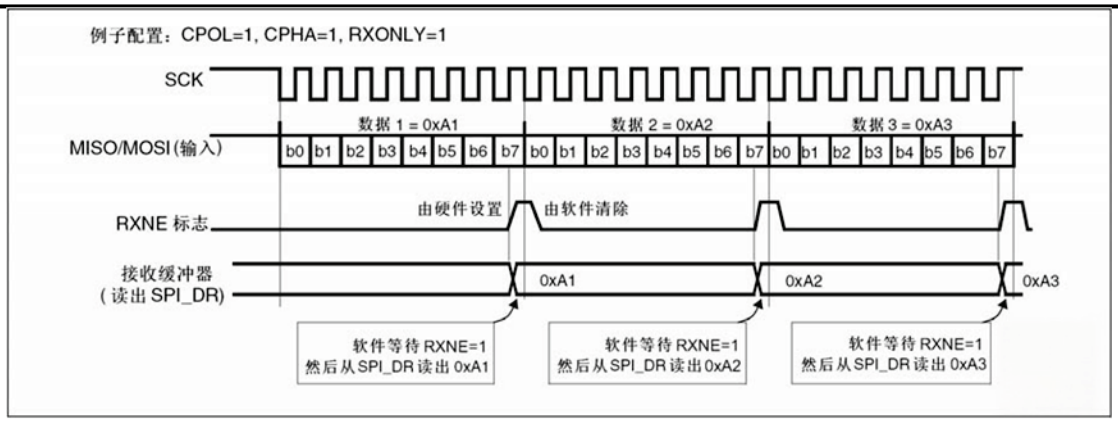


图 218 只接收模式(BIDIMODE=0 并且 RXONLY=1)下连续传输时，RXNE 变化示意图

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_CR2 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(见图 217)。

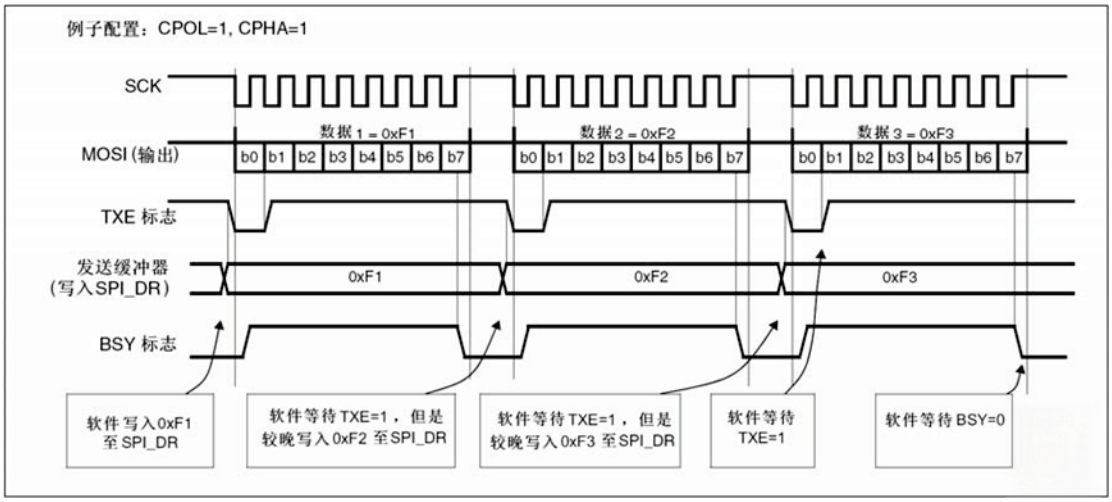


图 219 非连续传输发送(BIDIMODE=0 并且 RXONLY=0)时，TXE/BSY 变化示意图

23.3.6 CRC 计算

CRC 校验用于保证全双工通信的可靠性。数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。CRC 的计算是在由 SPI_CR1 寄存器中 CPHA 和 CPOL 位定义的采样时钟边沿进行的。

注意： *该 SPI 接口提供了两种 CRC 计算方法，取决于所选的发送和/或接收的数据帧格式：8 位数据帧采用 CR8；16 位数据帧采用 CRC16。*

CRC 计算是通过设置 SPI_CR1 寄存器中的 CRCEN 位启用的。设置 CRCEN 位时同时复位 CRC 寄存器(SPI_RXCRCR 和 SPI_TXCRCR)。当设置了 SPI_CR1 的 CRCNEXT 位，SPI_TXCRCR 的内容将在当前字节发送之后发出。

在传输 SPI_TXCRCR 的内容时，如果在移位寄存器中收到的数值与 SPI_RXCRCR 的内容不匹配，则 SPI_SR 寄存器的 CRCERR 标志位被置 1。

如果在 TX 缓冲器中还有数据，CRC 的数值仅在数据字节传输结束后传送。在传输 CRC 期间，CRC 计算器关闭，寄存器的数值保持不变。

注意：　请参考产品说明书，以确认有此功能(不是所有型号都有此功能)。

SPI 通信可以通过以下步骤使用 CRC：

- 设置 CPOL、CPHA、LSBFirst、BR、SSM、SSI 和 MSTR 的值；
- 在 SPI_CRCPR 寄存器输入多项式；
- 通过设置 SPI_CR1 寄存器 CRCEN 位使能 CRC 计算，该操作也会清除寄存器 SPI_RXCRCR 和 SPI_TXCRC；
- 设置 SPI_CR1 寄存器的 SPE 位启动 SPI 功能；
- 启动通信并且维持通信，直到只剩最后一个字节或者半字；
- 在把最后一个字节或半字写进发送缓冲器时，设置 SPI_CR1 的 CRCNext 位，指示硬件在发送完成最后一个数据之后，发送 CRC 的数值。在发送 CRC 数值期间，停止 CRC 计算；
- 当最后一个字节或半字被发送后，SPI 发送 CRC 数值，CRCNext 位被清除。同样，接收到的 CRC 与 SPI_RXCRCR 值进行比较，如果比较不相配，则设置 SPI_SR 上的 CRCERR 标志位，当设置了 SPI_CR2 寄存器的 ERRIE 时，则产生中断。

注意：　当 SPI 模块处于从设备模式时，请注意在时钟稳定之后再使能 CRC 计算，否则可能会得到错误的 CRC 计算结果。事实上，只要设置了 CRCEN 位，只要在 SCK 引脚上有输入时钟，不管 SPE 位的状态，都会进行 CRC 的计算。

当 SPI 时钟频率较高时，用户在发送 CRC 时必须小心。在 CRC 传输期间，使用 CPU 的时间应尽可能少；为了避免在接收最后的数据和 CRC 时出错，在发送 CRC 过程中应禁止函数调用。必须在发送/接收最后一个数据之前完成设置 CRCNEXT 位的操作。

当 SPI 时钟频率较高时，因为 CPU 的操作会影响 SPI 的带宽，建议采用 DMA 模式以避免 SPI 降低的速度。

当配置为从模式并且使用了 NSS 硬件模式，NSS 引脚应该在数据传输和 CRC 传输期间保持为低。

当配置 SPI 为从模式并且使用 CRC 的功能，即使 NSS 引脚为高时仍然会执行 CRC 的计算(译注：当 NSS 信号为高时，如果 SCK 引脚上有时钟脉冲，则 CRC 计算会继续执行)。例如：当主设备交替地与多个从设备进行通信时，将会出现这种情况(注：此时要想办法避免 CRC 的误操作)。

在不选中一个从设备(NSS 信号为高)转换到选中一个新的从设备(NSS 信号为低)的时候，为了保持主从设备端下次 CRC 计算结果的同步，应该清除主从两端的 CRC 数值。

按照下述步骤清除 CRC 数值：

1. 关闭 SPI 模块(SPE=0)；
2. 清除 CRCEN 位为'0'；
3. 设置 CRCEN 位为'1'；
4. 使能 SPI 模块(SPE=1)。

23.3.7 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

发送缓冲器空闲标志(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。

23.3.8 关闭 SPI

当通讯结束，可以通过关闭 SPI 模块来终止通讯。清除 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。

23.3.9 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI_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(见)。

- 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。

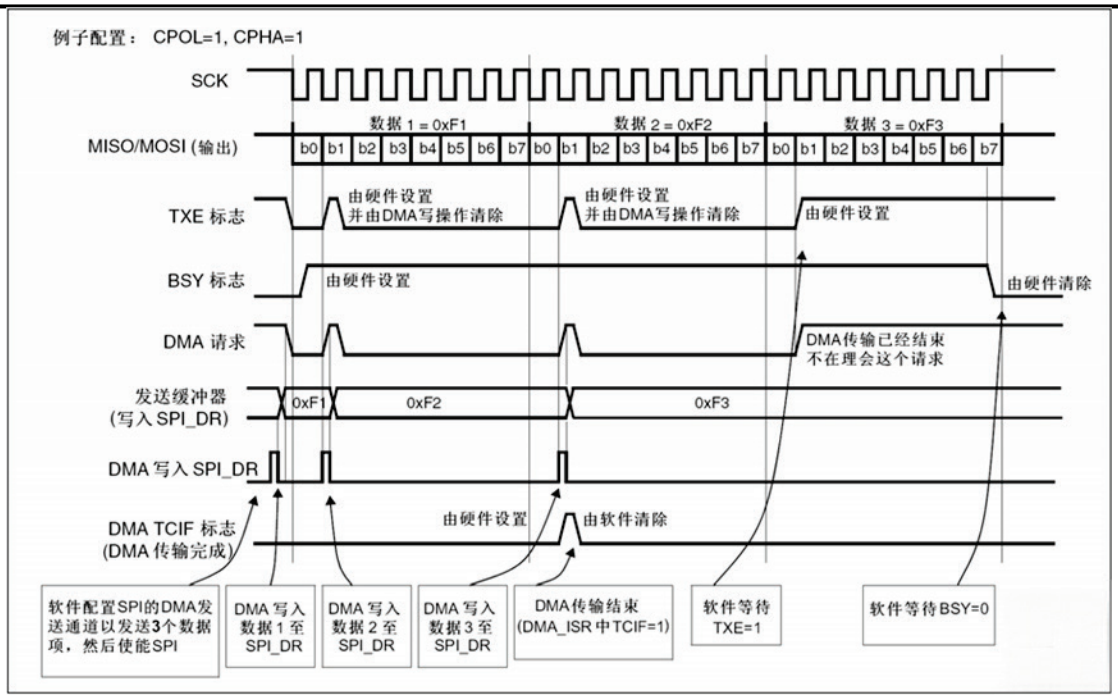


图 220 使用 DMA 发送

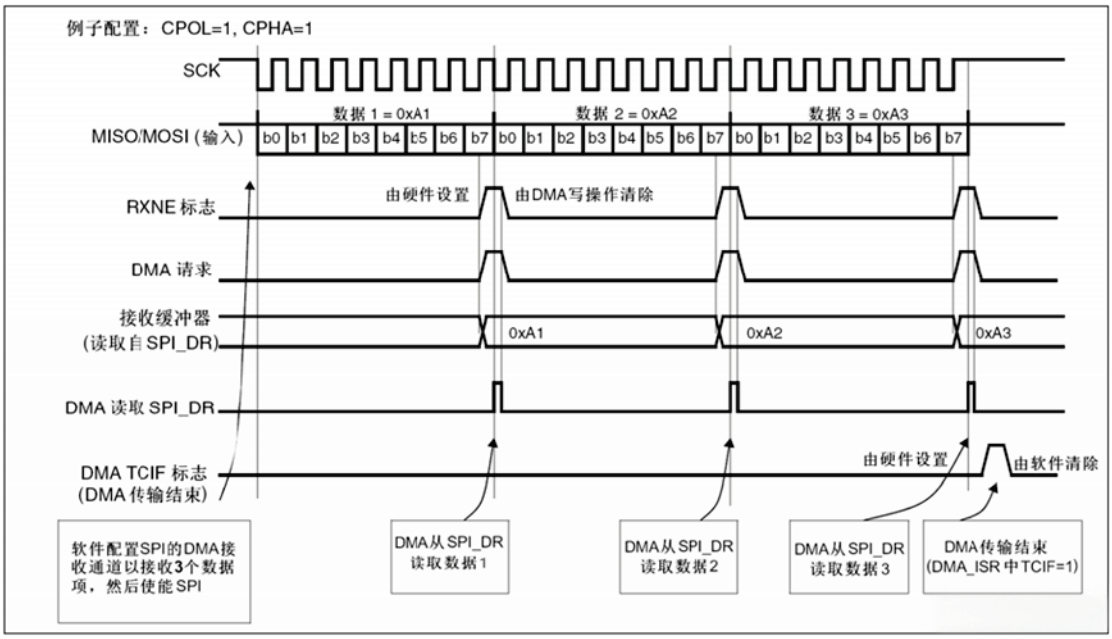


图 221 使用 DMA 接收

带 CRC 的 DMA 功能

当使能 SPI 使用 CRC 检验并且启用 DMA 模式时，在通信结束时，CRC 字节的发送和接收是自动完成的。

数据和 CRC 传输结束时，SPI_SR 寄存器的 CRCERR 标志为'1'表示在传输期间发生错误。

23.3.10 错误标志

主模式失效错误(MODF)

主模式失效仅发生在：NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，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。

CRC 错误

当设置了 SPI_CR 寄存器上的 CRCEN 位时，CRC 错误标志用来核对接收数据的有效性。如果移位寄存器中接收到的值(发送方发送的 SPI_TXCRCR 数值)与接收方 SPI_RXCRCR 寄存器中的数值不匹配，则 SPI_SR 寄存器上的 CRCERR 标志被置位为'1'。

23.3.11 SPI 中断

表 134 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	
CRC 错误标志	CRCERR	

23.4 I2S 功能描述

23.4.1 I2S 功能描述

I2S 的框图如下图所示：

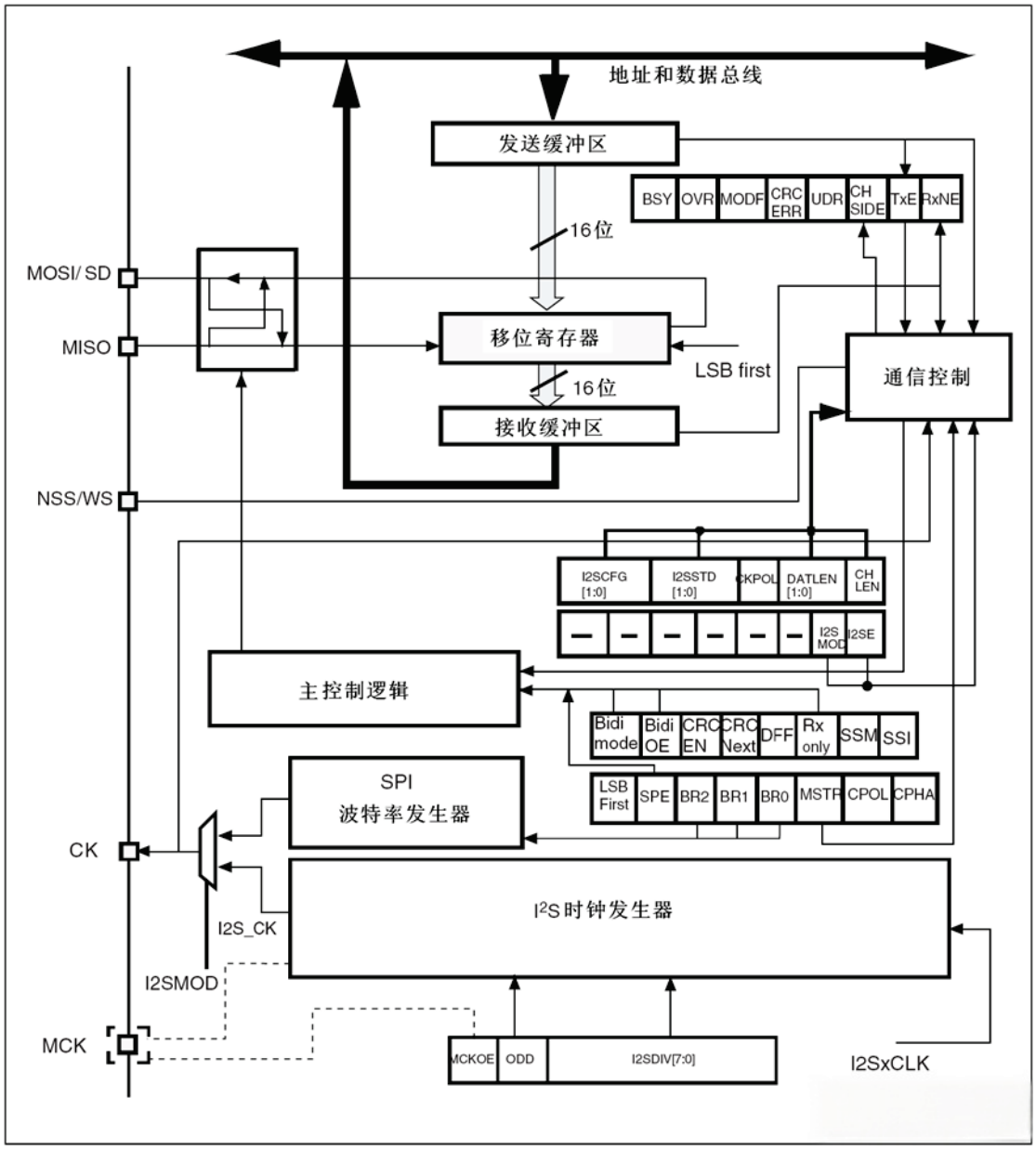


图 222 I2S 框图

通过将寄存器 SPI_I2SCFGR 的 I2SMOD 位置为'1'，即可使能 I2S 功能。此时，可以把 SPI 模块用作 I2S 音频接口。I2S 接口与 SPI 接口使用大致相同的引脚、标志和中断。

I2S 与 SPI 共用 3 个引脚：

- SD：串行数据(映射至 MOSI 引脚)，用来发送和接收 2 路时分复用通道的数据；
- WS：字选(映射至 NSS 引脚)，主模式下作为数据控制信号输出，从模式下作为输入；
- CK：串行时钟(映射至 SCK 引脚)，主模式下作为时钟信号输出，从模式下作为输入。

在某些外部音频设备需要主时钟时，可以另有一个附加引脚输出时钟：

- MCK：主时钟(独立映射)，在 I2S 配置为主模式，寄存器 SPI_I2SPR 的 MCKOE 位为'1'时，作为输出额外的时钟信号引脚使用。输出时钟信号的频率预先设置为 256×Fs，其中 Fs 是音频信号的采样频率。

设置成主模式时，I2S 使用自身的时钟发生器来产生通信用的时钟信号。这个时钟发生器也是主时钟输出的时钟源。I2S 模式下有 2 个额外的寄存器，一个是与时钟发生器配置相关的寄存器 SPI_I2SPR，另一个是 I2S 通用配置寄存器 SPI_I2SCFGR(可设置音频标准、从/主模式、数据格式、数据包帧、时钟极性等参数)。

在 I2S 模式下不使用寄存器 SPI_CR1 和所有的 CRC 寄存器。同样，I2S 模式下也不使用寄存器 SPI_CR2 的 SSOE 位，和寄存器 SPI_SR 的 MODF 位和 CRCERR 位。

I2S 使用与 SPI 相同的寄存器 SPI_DR 用作 16 位宽模式数据传输。

23.4.2 支持的音频协议

三线总线支持 2 个声道上音频数据的时分复用：左声道和右声道，但是只有一个 16 位寄存器用作发送或接收。因此，软件必须在对数据寄存器写入数据时，根据当前传输中的声道写入相应的数据；同样，在读取寄存器数据时，通过检查寄存器 SPI_SR 的 CHSIDE 位来判明接收到的数据属于哪个声道。左声道总是先于右声道发送数据(CHSIDE 位在 PCM 协议下无意义)。

有四种可用的数据和包帧组合。可以通过以下四种数据格式发送数据：

- 16 位数据打包进 16 位帧
- 16 位数据打包进 32 位帧
- 24 位数据打包进 32 位帧
- 32 位数据打包进 32 位帧

在使用 16 位数据扩展到 32 位帧时，前 16 位(MSB)是有意义的数据，后 16 位(LSB)被强制为 0，该操作不需要软件干预，也不需要有 DMA 请求(仅需要一次读/写操作)。

24 位和 32 位数据帧需要 CPU 对寄存器 SPI_DR 进行 2 次读或写操作，在使用 DMA 时，需要 2 次 DMA 传输。对于 24 位数据，扩展到 32 位后，最低 8 位由硬件置 0。对于所有的数据格式和通讯标准，总是先发送最高位(MSB)。

I2S 接口支持四种音频标准，可以通过设置寄存器 SPI_I2SCFGR 的 I2SSTD[1:0]位和 PCMSYNC 位来选择。

I2S 飞利浦标准

在此标准下，引脚 WS 用来指示正在发送的数据属于哪个声道。在发送第一位数据(MSB)前 1 个时钟周期，该引脚即为有效。

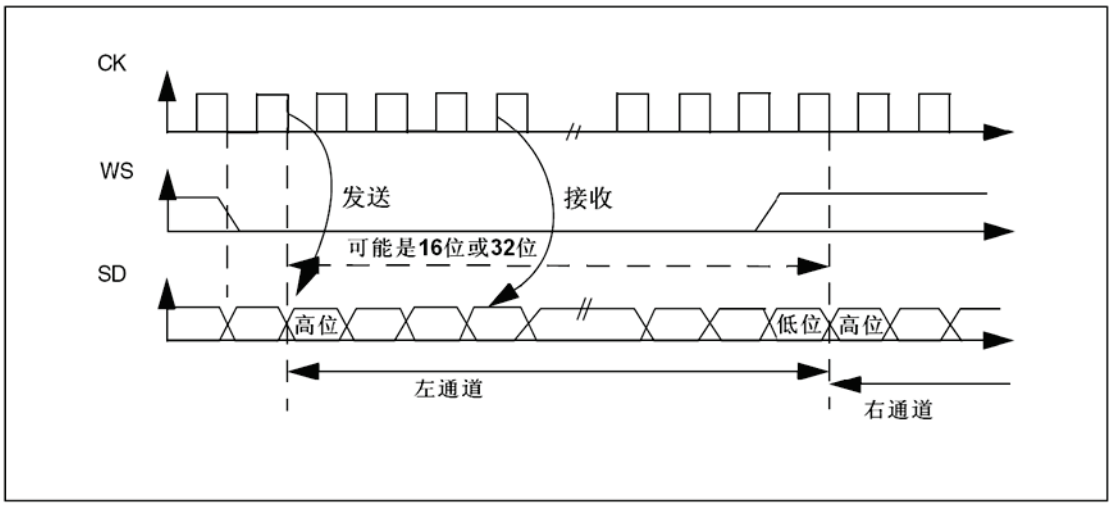


图 223 I2S 飞利浦协议波形(16/32 位全精度，CPOL=0)

发送方在时钟信号(CK)的下降沿改变数据，接收方在上升沿读取数据。WS 信号也在时钟信号的下降沿变化。

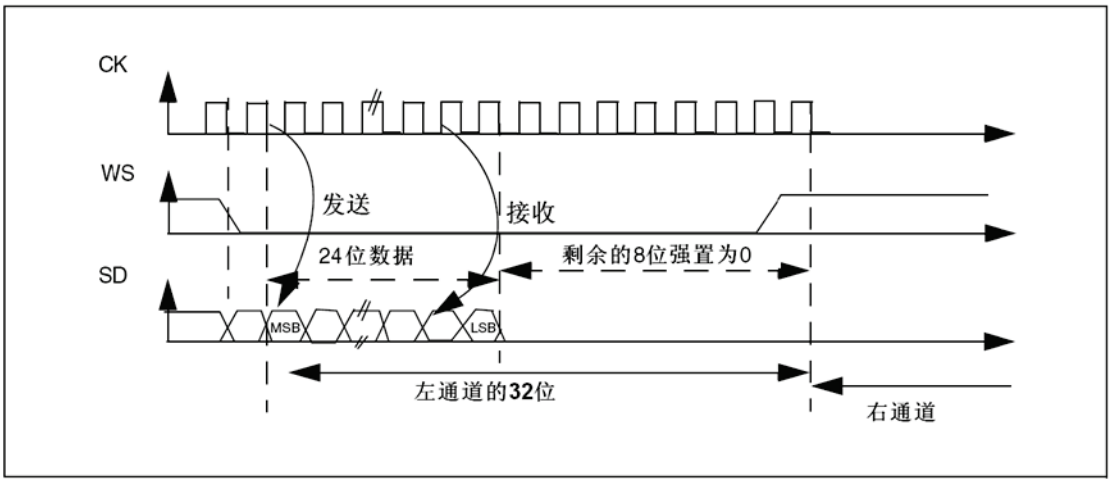


图 224 I2S 飞利浦协议标准波形(24 位帧，CPOL=0)

此模式需要对寄存器 SPI_DR 进行 2 次读或写操作。

- 在发送模式下：如果需要发送 0x8EAA33(24 位)：

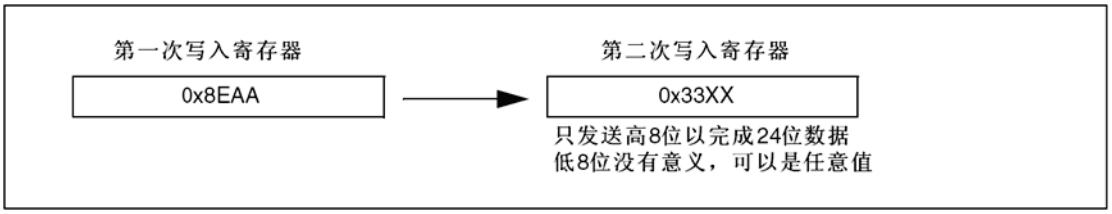


图 225 发送 0x8EAA33

- 在接收模式下：如果接收 0x8EAA33：

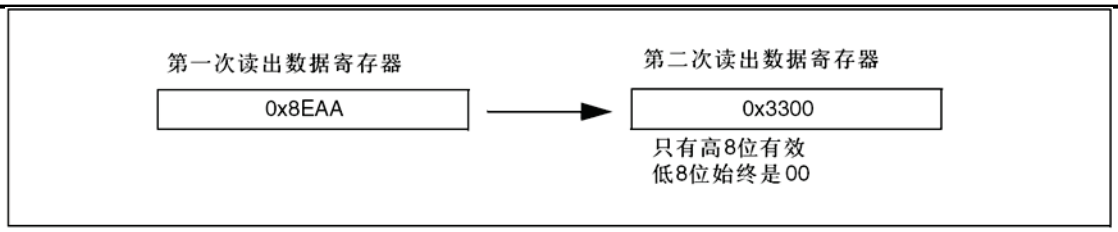


图 226 接收 0x8EAA33

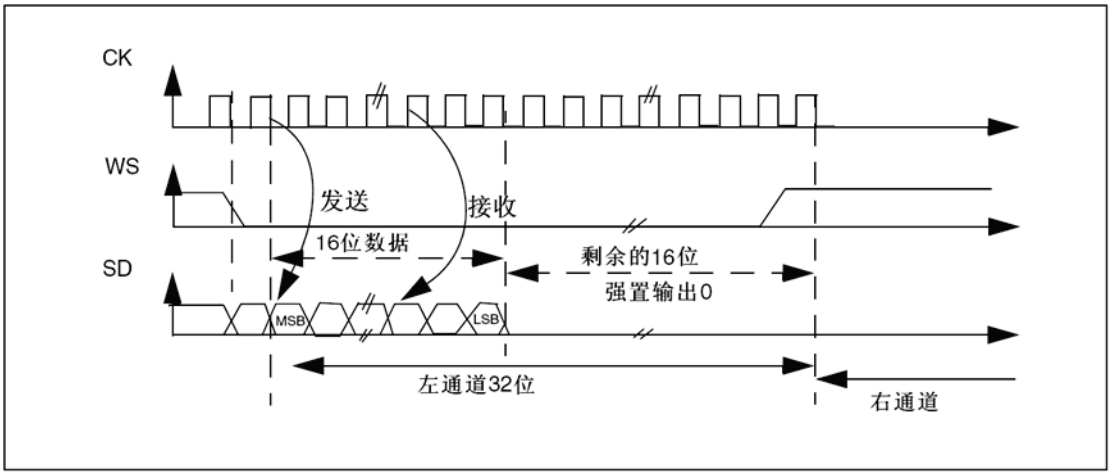


图 227 I2S 飞利浦协议标准波形(16 位扩展至 32 位包帧，CPOL=0)

在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 位声道帧，只需要访问一次寄存器 SPI_DR。用来扩展到 32 位的低 16 位被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3(扩展到 32 位是 0x76A3 0000)，需要的操作如下图所示。

图 228 示例

在发送时需要将 MSB 写入寄存器 SPI_DR；标志位 TXE 为'1'表示可以写入新的数据，如果允许了相应的中断，则可以产生中断。发送是由硬件完成的，即使还未发送出后 16 位的 0x0000，也会设置 TXE 并产生相应的中断。

接收时，每次收到高 16 位半字(MSB)后，标志位 RXNE 置'1'，如果允许了相应的中断，则可以产生中断。

这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

MSB 对齐标准

在此标准下，WS 信号和第一个数据位，即最高位(MSB)同时产生。

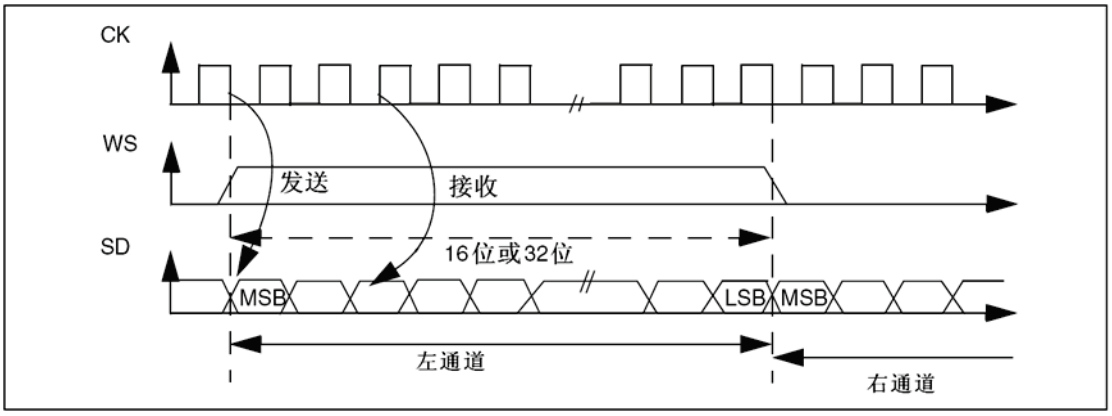


图 229 MSB 对齐 16 位或 32 位全精度，CPOL=0

发送方在时钟信号的下降沿改变数据；接收方是在上升沿读取数据。

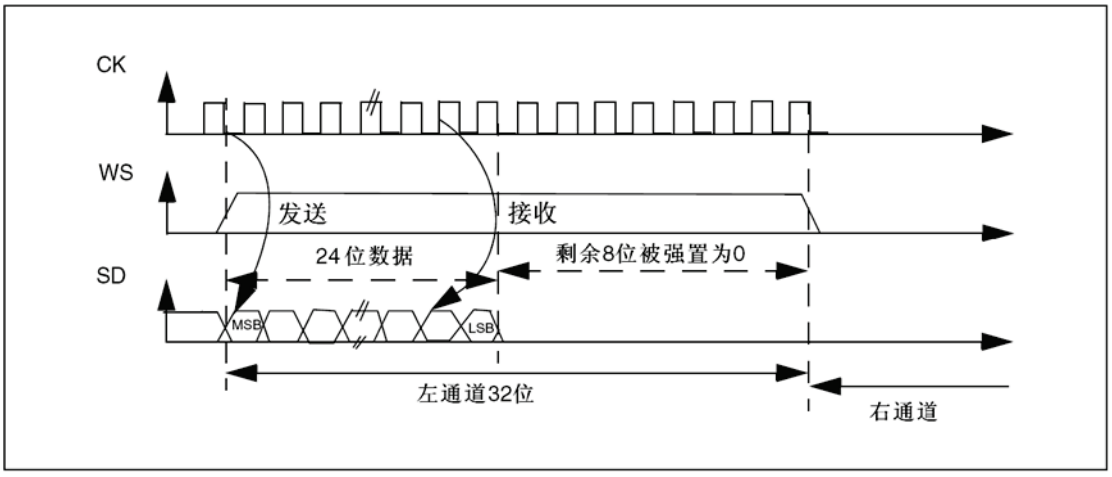


图 230 MSB 对齐 24 位数据，CPOL=0

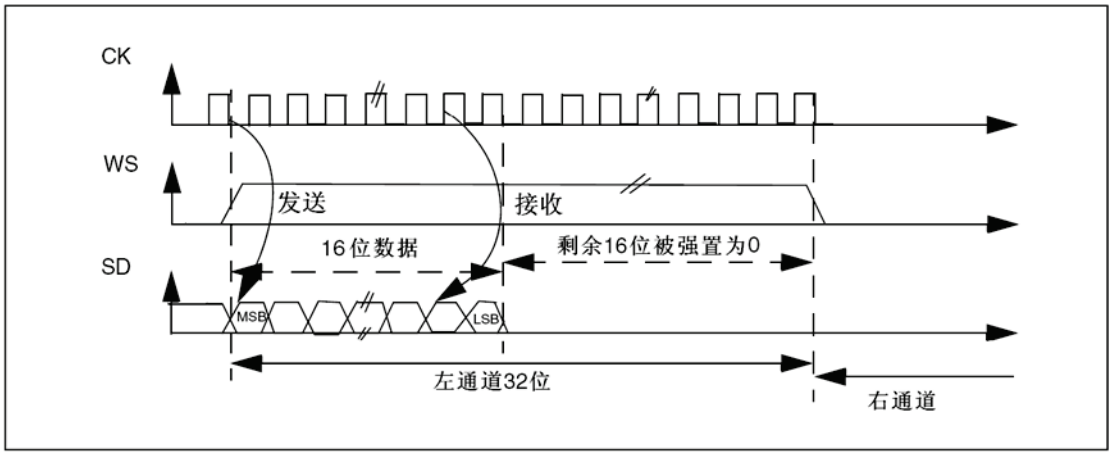


图 231 MSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0

LSB 对齐标准

此标准与 MSB 对齐标准类似(在 16 位或 32 位全精度帧格式下无区别)。

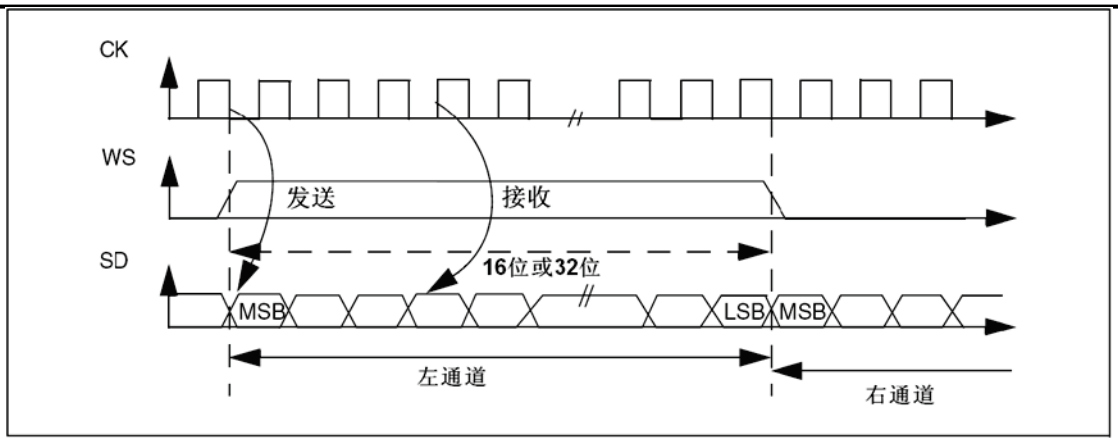


图 232 LSB 对齐 16 位或 32 位全精度，CPOL=0

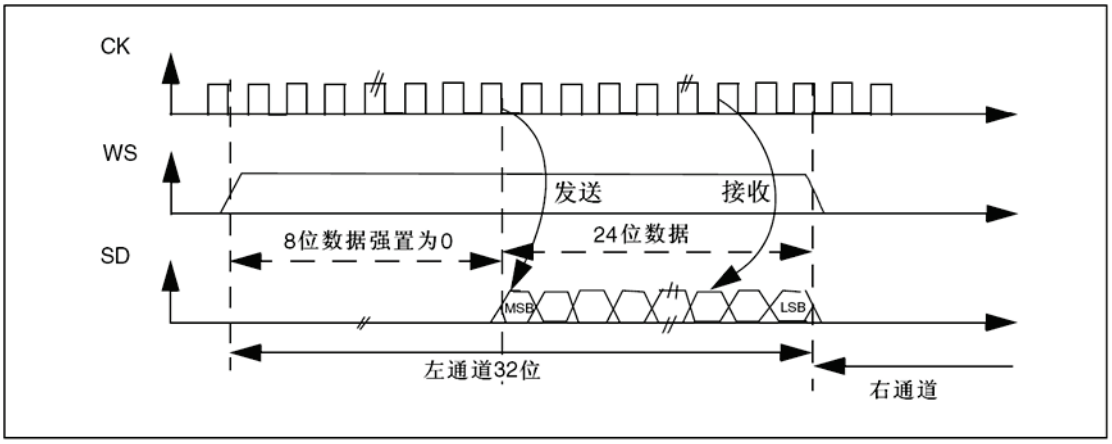


图 233 LSB 对齐 24 位数据，CPOL=0

- 在发送模式下

如果要发送数据 0x3478AE，需要通过软件或者 DMA 对寄存器 SPI_DR 进行 2 次写操作。操作流程如下图所示。

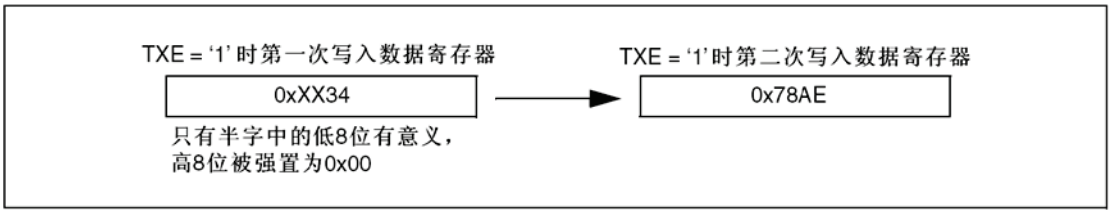


图 234 要求发送 0x3478AE 的操作

- 在接收模式下

如果要接收数据 0x3478AE，需要在 2 个连续的 RXNE 事件发生时，分别对寄存器 SPI_DR 进行 1 次读操作。

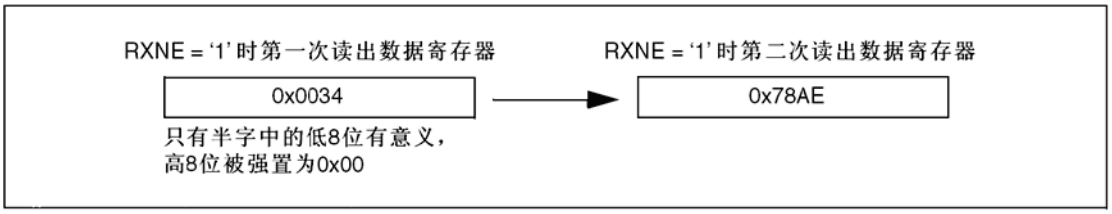


图 235 要求接收 0x3478AE 的操作

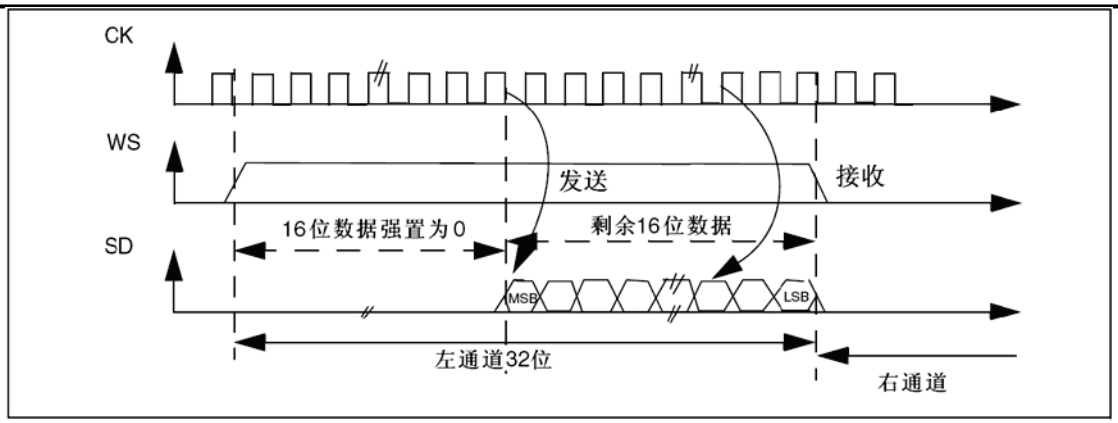


图 236 LSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0

在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 声道帧，只需要访问一次寄存器 SPI_DR。此时，扩展到 32 位后的高半字(16 位 MSB)被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3(扩展到 32 位是 0x0000 76A3)，需要的操作如下图所示。

图 237 示例

在发送时，如果 TXE 为'1'，用户需要写入待发送的数据(即 0x76A3)。用来扩展到 32 位的 0x0000 部分由硬件首先发送出去，一旦有效数据开始从 SD 引脚送出，即发生下一次 TXE 事件。

在接收时，一旦接收到有效数据(而不是 0x0000 部分)，即发生 RXNE 事件。这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

PCM 标准

在 PCM 标准下，不存在声道选择的信息。PCM 标准有 2 种可用的帧结构，短帧或者长帧，可以通过设置寄存器 SPI_I2SCFGR 的 PCMSYNC 位来选择。

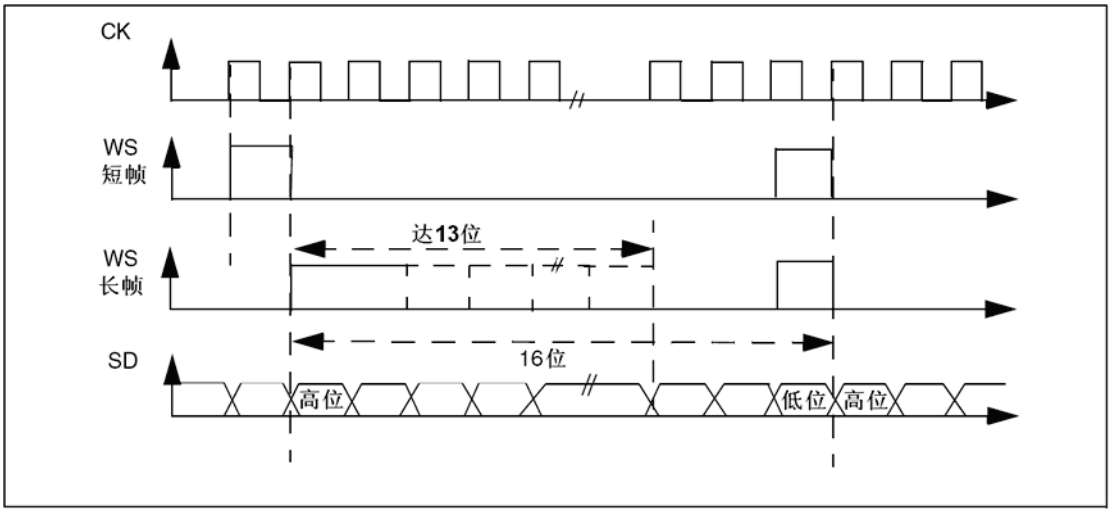


图 238 PCM 标准波形(16 位)

对于长帧，主模式下，用来同步的 WS 信号有效的时间固定为 13 位。

对于短帧，用来同步的 WS 信号长度只有 1 位。

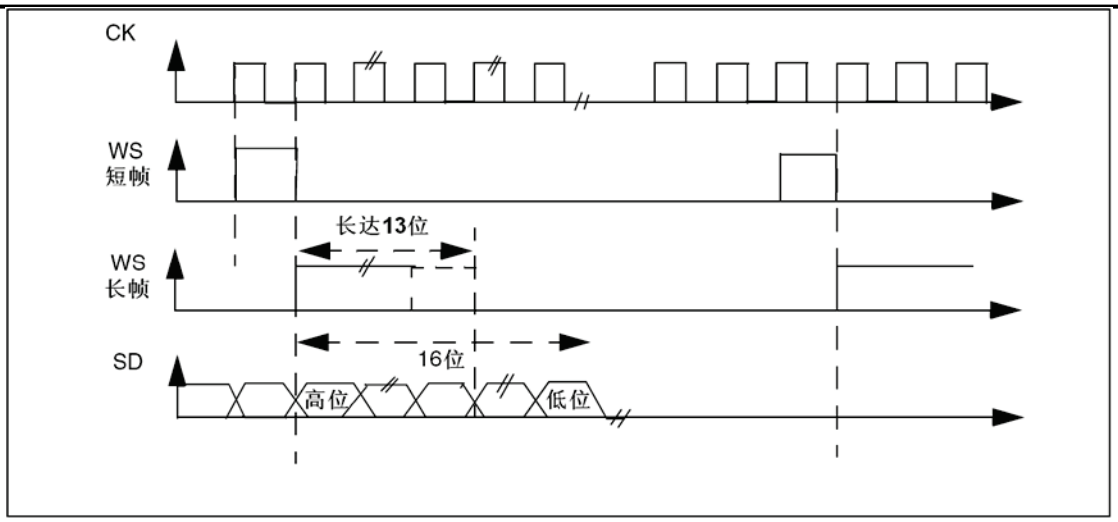


图 239 PCM 标准波形(16 位扩展到 32 位包帧)

注意：　无论哪种模式(主或从)、哪种同步方式(短帧或长帧)，连续的 2 帧数据之间和 2 个同步信号之间的时间差，(即使是从模式)需要通过设置 SPI_I2SCFGR 寄存器的 DATLEN 位和 CHLEN 位来确定。

23.4.3 时钟发生器

I2S 的比特率即确定了在 I2S 数据线上的数据流和 I2S 的时钟信号频率。I2S 比特率=每个声道的比特数×声道数目×音频采样频率

对于一个具有左右声道和 16 位音频信号，I2S 比特率计算如下 I2S 比特率=16×2×Fs

如果包长为 32 位，则有：I2S 比特率=32×2×Fs

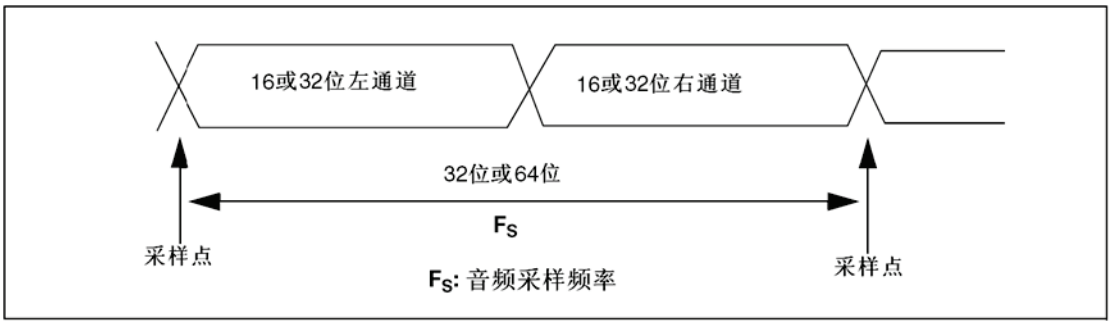


图 240 音频采样频率定义

在主模式下，为了获得需要的音频频率，需要正确地对线性分频器进行设置。

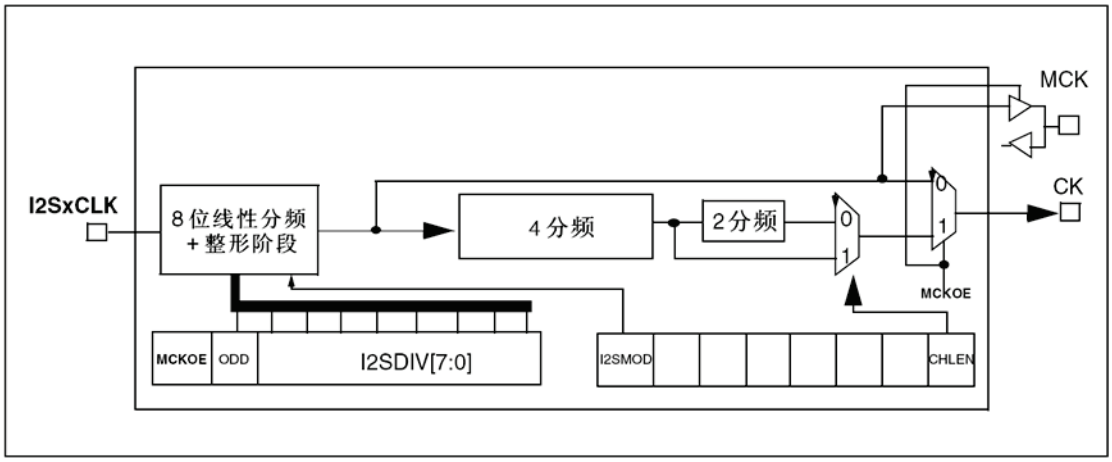


图 241 I2S 时钟发生器结构

1.图中 x 可以是 2 或者 3。

上图中 I2SxCLK 的时钟源是系统时钟(即驱动 AHB 时钟的 HSI、HSE 或 PLL)。

音频的采样频率可以是 96kHz、48kHz、44.1kHz、32kHz、22.05kHz、16kHz、11.025kHz 或者 8kHz(或任何此范围内的数值)。为了获得需要的频率，需按照以下公式设置线性分频器：

当需要生成主时钟时(寄存器 SPI_I2SPR 的 MCKOE 位为'1')：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)*8]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)*4]当关闭主时钟时(MCKOE 位为'0')：

声道的帧长为 16 位时，Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)]

声道的帧长为 32 位时，Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)]下面 2 张表给出了不同时钟配置时，精确参数的例子。

注：　可以使用其它配置以达到优化时钟精确度的目的。

表 135 使用标准的 8MHzHSE 时钟得到精确的音频频率

SYSCLK (MHz)	12S_DIV		12S_ODD		MCLK	期望值 FS(Hz)	实际的 FS(Hz)		误差	
	16 位		32 位				16 位	32 位	16 位	32 位
72	11	6	1	0	无	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	无	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	无	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	无	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	无	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	无	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	无	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	无	8000	8007.11	7978.72	0.09%	0.27%
72	2	2	0	0	有	96000	70312.15	70312.15	26.76 %	26.76%
72	3	3	0	0	有	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	有	44100	46875	46875	6.29%	6.29%
72	9	9	0	0	有	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	有	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	有	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	有	11025	10817.3	10817.3	1.88%	1.88%
72	17	17	1	1	有	8000	8035.71	8035.71	0.45%	0.45%

23.4.4 I²S 主模式

设置 I²S 工作在主模式，串行时钟由引脚 CK 输出，字选信号由引脚 WS 产生。可以通过设置寄存器 SPI_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟(MCK)。

流程

1. 设置寄存器 SPI_I2SPR 的 I2SDIV[7:0]定义与音频采样频率相符的串行时钟波特率。同时也要定义寄存器 SPI_I2SPR 的 ODD 位。
2. 设置 CKPOL 位定义通信用时钟在空闲时的电平状态。如果需要向外部的 DAC/ADC 音频器件提供主时钟 MCK，将寄存器 SPI_I2SPR 的 MCKOE 位置为'1'。(按照不同的 MCK 输出状态，计算 I2SDIV 和 ODD 的值，详见 23.4.3 节)。
3. 设置寄存器 SPI_I2SCFGR 的 I2SMOD 位为'1'激活 I2S 功能，设置 I2SSTD[1:0]和 PCMSYNC 位选择所用的 I2S 标准，设置 CHLEN 选择每个声道的数据位数。还要设置寄存器 SPI_I2SCFGR 的 I2SCFG[1:0]选择 I2S 主模式和方向(发送端还是接收端)。
4. 如果需要，可以通过设置寄存器 SPI_CR2 来打开所需的中断功能和 DMA 功能。

5. 必须将寄存器 SPI_I2SCFGR 的 I2SE 位置为'1'。
6. 引脚 WS 和 CK 需要配置为输出模式。如果寄存器 SPI_I2SPR 的 MCKOE 位为'1'，引脚 MCK 也要配置成输出模式。

发送流程

当写入 1 个半字(16 位)的数据至发送缓存，发送流程开始。

假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时，标志位 TXE 置'1'，这时，要把对应右声道的数据写入发送缓存。标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。标志位 CHSIDE 的值在 TXE 为'1'时更新，因此它在 TXE 为'1'时有意义。在先左声道后右声道的数据都传输完成后，才能被认为是一个完整的数据帧。不可以只传输部分数据帧，如仅有左声道的数据。

当发出第一位数据的同时，半字数据被并行地传送至 16 位移位寄存器，然后后面的位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓存移至移位寄存器时，标志位 TXE 置为'1'，如果寄存器 SPI_CR2 的 TXEIE 位为'1'，则产生中断。

写入数据的操作取决于所选择的 I2S 标准，详见 23.4.2 节。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI_DR 写入下一个要传输的数据。

建议在要关闭 I2S 功能时，等待标志位 TXE=1 及 BSY=0，再将 I2SE 位清'0'。

接收流程

接收流程的配置步骤除了第 3 点外，与发送流程的一致(参见前述的"发送流程")，需要通过配置 I2SCFG[1:0]来选择主接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收。即每次填满接收缓存后，标志位 RXNE 置'1'，如果寄存器 SPI_CR2 的 RXNEIE 位为'1'，则产生中断。根据配置的数据和声道长度，收到左声道或右声道的数据会需要 1 次或者 2 次把数据传送到接收缓存的过程。

对寄存器 SPI_DR 进行读操作即可清除 RXNE 标志位。

每次接收以后即更新 CHSIDE。它的值取决于 I^2S 单元产生的 WS 信号。读取数据的操作取决于所选择的 I^2S 标准，详见 23.4.2 节。

如果前一个接收到的数据还没有被读取，又接收到新数据，即发生上溢，标志位 OVR 被置为'1'，如果寄存器 SPI_CR2 的 ERRIE 位为'1'，则产生中断，表示发生了错误。

若要关闭 I^2S 功能，需要执行特别的操作，以保证 I^2S 模块可以正常地完成传输周期而不会开始新的数据传输。操作过程与数据配置和通道长度、以及音频协议的模式相关：

- 16 位数据扩展到 32 位通道长度(DATLEN=00 并且 CHLEN=1)，使用 LSB(低位)对齐模式(I2SSTD=10)
  1. 等待倒数第二个(n-1)RXNE=1；
  2. 等待 17 个 I2S 时钟周期(使用软件延迟)；
  3. 关闭 I2S(I2SE=0)。
- 16 位数据扩展到 32 位通道长度(DATLEN=00 并且 CHLEN=1)，使用 MSB(高位)对齐、I^2S 或 PCM 模式(分别为 I2SSTD=00，I2SSTD=01 或 I2SSTD=11)
  1. 等待最后一个 RXNE=1；
  2. 等待 1 个 I^2S 时钟周期(使用软件延迟)；
  3. 关闭 I^2S(I2SE=0)。
- 所有其它 DATLEN 和 CHLEN 的组合，I2SSTD 选择的任意音频模式，使用下述方式关闭 I2S：

1. 等待倒数第二个(n-1)RXNE=1；
2. 等待一个 I2S 时钟周期(使用软件延迟)；
3. 关闭 I2S(I2SE=0)。

注：　在传输期间 BSY 标志始终为低。

23.4.5 I²S 从模式

在从模式下，I²S 可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要 I²S 接口提供时钟。时钟信号和 WS 信号都由外部主 I²S 设备提供，连接到相应的引脚上。因此用户无需配置时钟。

配置步骤列举如下：

1. 设置寄存器 SPI_I2SCFGR 的 I2SMOD 位激活 I²S 功能；设置 I2SSTD[1:0]来选择所用的 I²S 标准；设置 DATLEN[1:0]选择数据的比特数；设置 CHLEN 选择每个声道的数据位数。设置寄存器 SPI_I2SCFGR 的 I2SCFG[1:0]选择 I²S 从模式的数据方向(发送端还是接收端)。
2. 根据需要，设置寄存器 SPI_CR2 打开所需的中断功能和 DMA 功能。
3. 必须设置寄存器 SPI_I2SCFGR 的 I2SE 位为'1'。

发送流程

当外部主设备发送时钟信号，并且当 NSS_WS 信号请求传输数据时，发送流程开始。必须先使能从设备，并且写入 I²S 数据寄存器之后，外部主设备才能开始通信。

对于 I2S 的 MSB 对齐和 LSB 对齐模式，第一个写入数据寄存器的数据项对应左声道的数据。当开始通信时，数据从发送缓冲器传送到移位寄存器，然后标志位 TXE 置为'1'；这时，要把对应右声道的数据项写入 I²S 数据寄存器。

标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。与主模式的发送流程相比，在从模式中，CHSIDE 取决于来自外部主 I²S 的 WS 信号。这意味着从 I²S 在接收到主端生成的时钟信号之前，就要准备好第一个要发送的数据。WS 信号为'1'表示先发送左声道。

注意：　设置 I2SE 位为'1'的时间，应当比 CK 引脚上的主 I²S 时钟信号早至少 2 个 PCLK 时钟周期。

当发出第一位数据的时候，半字数据并行地通过 I²S 内部总线传输至 16 位移位寄存器，然后其它位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓冲器传送至移位寄存器时，标志位 TXE 置'1'，如果寄存器 SPI_CR2 的 TXEIE 位为'1'，则产生中断。

注意：　在对发送缓冲器写入数据前，要确认标志位 TXE 为'1'。写入数据的操作取决于所选中的 I2S 标准，详见 23.4.2 节。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI_DR 写入下一个要传输的数据。如果在代表下一个数据传输的第一个时钟边沿到达之前，新的数据仍然没有写入寄存器 SPI_DR，下溢标志位会置'1'，并可能产生中断；它指示软件发送数据错误。如果寄存器 SPI_CR2 的 ERRIE 位为'1'，在寄存器 SPI_SR 的标志位 UDR 为高是，就会产生中断。建议在这时关闭 I²S，然后重新从左声道开始发送数据。

建议在清除 I2SE 位关闭 I²S 之前，先等待 TXE=1 并且 BSY=0。

接收流程

配置步骤除了第 1 点外，与发送流程一致。需要通过配置 I2SCFG[1:0]来选择主接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收，即每次填满接收缓存，标志位 RXNE 置'1'，如果寄存器 SPI_CR2 的 RXNEIE 位为'1'，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要 1 次或者 2 次传输数据至接收缓冲器的过程。

每次接收到数据(将要从 SPI_DR 读出)以后即更新 CHSIDE，它对应 I2S 单元产生的 WS 信号。读取 SPI_DR 寄存器，将清除 RXNE 位。

读取数据的操作取决于所选中的 I²S 标准，详见 23.4.2 节。

在还没有读出前一个接收到的数据，又接收到新数据时，即产生上溢，并设置标志位 OVR 为'1'；如果寄存器 SPI_CR2 的 ERRIE 位为'1'，则产生中断，指示发生了错误。要关闭 I²S 功能时，需要在接收到最后一次 RXNE=1 时将 I2SE 位清'0'。

注意：　外部主 I2S 器件需要有通过音频声道发送/ 接收 16 位或 32 位数据包的功能。

23.4.6 状态标志位

有 3 个状态标志位供用户监控 I2S 总线的状态。

忙标志位(BSY)

BSY 标志由硬件设置与清除(写入此位无效果)，该标志位指示 I2S 通信层的状态。

该位为'1'时表明 I²S 通讯正在进行中，但有一个例外：主接收模式(I2SCFG=11)下，在接收期间 BSY 标志始终为低。

在软件要关闭 SPI 模块之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

当传输开始时，BSY 标志被置为'1'，除非 I²S 模块处于主接收模式。下述情况时，该标志位被清除：

- 当传输结束时(除了主发送模式，这种模式下通信是连续的)；
- 当关闭 I²S 模块时。

当通信是连续的时候：

- 在主发送模式时，整个传输期间，BSY 标志始终为高；
- 在从模式时，每个数据项传输之间，BSY 标志在 1 个 I²S 时钟周期内变低。

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。

发送缓存空标志位(TXE)

该标志位为'1'表示发送缓冲器为空，可以对发送缓冲器写入新的待发送数据。在发送缓冲器中已有数据时，标志位清'0'。在 I²S 被关闭时(I2SE 位为'0')，该标志位也为'0'。

接收缓存非空标志位(RXNE)

该标志位置'1'表示在接收缓存里有接收到的有效数据。在读取寄存器 SPI_DR 时，该位清'0'。

声道标志位(CHSIDE)

在发送模式下，该标志位在 TXE 为高时刷新，指示从 SD 引脚上发送的数据所在的声道。如果在从发送模式下发生了下溢错误，该标志位的值无效，在重新开始通讯前需要把 I²S 关闭再打开。

在接收模式下，该标志位在寄存器 SPI_DR 接收到数据时刷新，指示接收到的数据所在的声道。注意，如果发生错误(如上溢 OVR)，该标志位无意义，需要将 I²S 关闭再打开(同时，如果必要修改 I2S 的配置)。

在 PCM 标准下，无论短帧格式还是长帧格式，这个标志位都没有意义。

如果寄存器 SPI_SR 的标志位 OVR 或 UDR 为'1'，且寄存器 SPI_CR2 的 ERRIE 位为'1'，则会产生中断。(中断源已经被清除后)可以通过读寄存器 SPI_SR 来清除中断标志。

23.4.7 错误标志位

I²S 单元有 2 个错误标志位。

下溢标志位(UDR)

在从发送模式下，如果数据传输的第一个时钟边沿到达时，新的数据仍然没有写入 SPI_DR 寄存器，该标志位会被置'1'。在寄存器 SPI_I2SCFGR 的 I2SMOD 位置'1'后，该标志位才有效。如果寄存器 SPI_CR2 的 ERRIE 位为'1'，就会产生中断。

通过对寄存器 SPI_SR 进行读操作来清除该标志位。

上溢标志位(OVR)

如果还没有读出前一个接收到的数据时，又接收到新的数据，即产生上溢，该标志位置'1'，如果寄存器 SPI_CR2 的 ERRIE 位为'1'，则产生中断指示发生了错误。

这时，接收缓存的内容，不会刷新为从发送设备送来的新数据。对寄存器 SPI_DR 的读操作返回最后一个正确接收到的数据。其他所有在上溢发生后由发送设备发出的 16 位数据都会丢失。

通过先读寄存器 SPI_SR 再读寄存器 SPI_DR，来清除该标志位。

23.4.8 I²S 中断

下表列举了全部 I2S 中断

表 136 I²S 中断请求

中断事件	事件标志位	使能标志位
发送缓冲器空标志位	TXE	TXEIE
接收缓冲器非空标志位	RXNE	RXNEIE
下溢标志位	OVR	ERRIE
上溢标志位	UDR	

23.4.9 DMA 功能

DMA 的工作方式在 I2S 模式除了 CRC 功能不可用以外，与在 SPI 模式完全相同。因为在 I2S 模式下没有数据传输保护系统。

23.5 SPI 和 I²S 寄存器描述

关于寄存器描述中所用到的缩略词可参见第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

23.5.1 SPI 控制寄存器 1(SPI_CR1)(I²S 模式下不使用)

地址偏移：0x00

复位值：0x0000

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

BIDI MODE	BIDIOE	CRCEN	CRCNE XT	DFF	RXO NLY	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	rw	rw

位	符号	说明
15	BIDIMODE	**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。 注：I2S 模式下不使用。
14	BIDIOE	**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)。 这个“单线”数据线在主设备端为 MOSI 引脚，在从设备端为 MISO 引脚。 注：I2S 模式下不使用。
13	CRCEN	**CRCEN**：硬件 CRC 校验使能(Hardware CRC calculation enable) 0：禁止 CRC 计算； 1：启动 CRC 计算。 注：只有在禁止 SPI 时(SPE=0)，才能写该位，否则出错。该位只能在全双工模式下使用。 注：I2S 模式下不使用。
12	CRCNEXT	**CRCNEXT**：下一个发送 CRC(Transmit CRC next) 0：下一个发送的值来自发送缓冲区。 1：下一个发送的值来自发送 CRC 寄存器。 注：在 SPI_DR 寄存器写入最后一个数据后应马上设置该位。 注：I2S 模式下不使用。
11	DFF	**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。 注：I2S 模式下不使用。
10	RXONLY	**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)。注：I2S 模式下不使用。
9	SSM	**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：启用软件从设备管理。 注：I2S 模式下不使用。
8	SSI	**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。 注：I2S 模式下不使用。
7	LSBFIRST	**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。 注：I2S 模式下不使用。
6	SPE	**SPE**：SPI 使能(SPI enable) 0：禁止 SPI 设备；

		1：开启 SPI 设备。 注：I2S 模式下不使用。 注：当关闭 SPI 设备时，请按照第 23.3.8 节的过程操作。
5:3	BR[2:0]	**BR[2:0]**：波特率控制(Baud rate control) 000：fPCLK/2　100：fPCLK/32　010：fPCLK/8　011：fPCLK/16 001：fPCLK/4　101：fPCLK/64　110：fPCLK/128　111：fPCLK/256 当通信正在进行的时候，不能修改这些位。 注意：I2S 模式下不使用。
2	MSTR	**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。 注：I2S 模式下不使用。
1	CPOL	**CPOL**：时钟极性(Clock polarity) 0：空闲状态时，SCK 保持低电平； 1：空闲状态时，SCK 保持高电平。 注：当通信正在进行的时候，不能修改该位。 注：I2S 模式下不使用。
0	CPHA	**CPHA**：时钟相位(Clock phase) 0：数据采样从第一个时钟边沿开始； 1：数据采样从第二个时钟边沿开始。 注：当通信正在进行的时候，不能修改该位。 注：I2S 模式下不使用。

23.5.2 SPI 控制寄存器 2(SPI_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
保留								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAEN	RXDMAEN
								rw	rw	rw	res	res	rw	rw	rw

位	符号	说明
15:8	Reserved	保留位，硬件强制为 0
7	TXEIE	**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。
6	RXNEIE	**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。
5	ERRIR	**ERRIR**：错误中断使能(Error interrupt enable) 当错误(CRCERR、OVR、MODF)产生时，该位控制是否产生中断 0：禁止错误中断； 1：允许错误中断。
4:3	Reserved	保留位，硬件强制为 0。
2	SSOE	**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。 注：I2S 模式下不使用。

1	TXDMAEN	**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。
0	RXDMAEN	**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。

23.5.3 SPI 状态寄存器(SPI_SR)

地址偏移：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
								r	r	r	rc_w0	r	r	r	r

位	符号	说明
15:8	Reserved	保留位，硬件强制为 0
7	BSY	**BSY**：忙标志(Busy flag) 0：SPI 不忙； 1：SPI 正忙于通信，或者发送缓冲非空。该位由硬件置位或者复位。 注：使用这个标志时需要特别注意，详见第 23.3.7 节和第 23.3.8 节。
6	OVR	**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。关于软件序列的详细信息，参考 23.4.7 节。
5	MODF	**MODF**：模式错误(Mode fault) 0：没有出现模式错误； 1：出现模式错误。 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考 23.3.10 节。 注：I2S 模式下不使用。
4	CRCERR	**CRCERR**：CRC 错误标志(CRC error flag) 0：收到的 CRC 值和 SPI_RXCRCR 寄存器中的值匹配； 1：收到的 CRC 值和 SPI_RXCRCR 寄存器中的值不匹配。该位由硬件置位，由软件写'0'而复位。 注：I2S 模式下不使用。
3	UDR	**UDR**：下溢标志位(Underrun flag) 0：未发生下溢； 1：发生下溢。 该标志位由硬件置'1'，由一个软件序列清'0'，详见 23.4.7 节。 注：在 SPI 模式下不使用。
2	CHSIDE	**CHSIDE**：声道(Channel side) 0：需要传输或者接收左声道； 1：需要传输或者接收右声道。 注：在 SPI 模式下不使用。在 PCM 模式下无意义。
1	TXE	**TXE**：发送缓冲为空(Transmit buffer empty) 0：发送缓冲非空； 1：发送缓冲为空。
0	RXNE	**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。

23.5.4 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

位	符号	说明
15:0	DR[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。 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，需要在启用 SPI 之前就确定好数据帧格式。 对于 8 位的数据，缓冲器是 8 位的，发送和接收时只会用到 SPI_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]。

23.5.5 SPICRC 多项式寄存器(SPI_CRCPR)(I²S 模式下不使用)

地址偏移：0x10 复位值：0x0007

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

位	符号	说明
15:0	CRCPOLY[15:0]	**CRCPOLY[15:0]**：CRC 多项式寄存器(CRC polynomial register)该寄存器包含了 CRC 计算时用到的多项式。 其复位值为 0x0007，根据应用可以设置其他数值。注：在 I2S 模式下不使用。

23.5.6 SPIRxCRC 寄存器(SPI_RXCRCR)(I²S 模式下不使用)

地址偏移：0x14

复位值:0x0000

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

位	符号	说明
15:0	RXCRC[15:0]	**RXCRC[15:0]**：接收 CRC 寄存器 在启用 CRC 计算时，RXCRC[15:0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CR1 的 CRCEN 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在 I2S 模式下不使用。

23.5.7 SPITxCRC 寄存器(SPI_TXCRCR)(I²S 模式下不使用)

地址偏移：0x18

复位值：0x0000

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

位	符号	说明
15:0	TxCRC[15:0]	**TxCRC[15:0]**：发送 CRC 寄存器 在启用 CRC 计算时，TXCRC[15:0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPI_CR1 中的 CRCEN 位写入'1'时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 个位都参与计算，并且按照 CRC16 的标准。 注：当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。 注：在 I2S 模式下不使用。

23.5.8 SPI_I²S 配置寄存器(SPI_I2SCFGR)

地址偏移：0x1C

复位值:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				I2SMOD	I2SE	I2SCFG		PCM SYNC	保留	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw

位	符号	说明
15:12	Reserved	保留位，硬件强制为 0
11	I2SMOD	**I2SMOD**：I2S 模式选择(I2S mode selection) 0:选择 SPI 模式； 1:选择 I2S 模式。 注：该位只有在关闭了 SPI 或者 I2S 时才能设置。
10	I2SE	**I2SE**:I2S 使能(I2S enable) 0:关闭 I2S； 1:I2S 使能。 注：在 SPI 模式下不使用。
9:8	I2SCFG	**I2SCFG**：I2S 模式设置(I2S configuration mode) 00:从设备发送； 01:从设备接收； 10:主设备发送； 11:主设备接受。 注：该位只有在关闭了 I2S 时才能设置。在 SPI 模式下不使用。
7	PCMSYNC	**PCMSYNC**：PCM 帧同步(PCM frame synchronization) 0:短帧同步； 1:长帧同步。 注:该位只在 I2SSTD=11(使用 PCM 标准)时有意义。在 SPI 模式下不使用。
6	Reserved	保留位，硬件强制为 0。
5:4	I2SSTD	**I2SSTD**：I2S 标准选择(I2S standard selection) 00:I2S 飞利浦标准； 01:高字节对齐标准(左对齐)； 10:低字节对齐标准(右对齐)； 11:PCM 标准。 关于 I2S 标准的细节，详见 23.4.2 节。

		注：为了正确操作，只有在关闭了 I2S 时才能设置该位。在 SPI 模式下不使用。
3	CKPOL	**CKPOL**：静止态时钟极性(Steady state clock polarity) 0:I2S 时钟静止态为低电平； 1:I2S 时钟静止态为高电平。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。在 SPI 模式下不使用。
2:1	DATLEN	**DATLEN**：待传输数据长度(Data length to be transferred) 00:16 位数据长度； 01:24 位数据长度； 10:32 位数据长度； 11:不允许。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。在 SPI 模式下不使用。
0	CHLEN	**CHLEN**：声道长度(每个音频声道的数据位数)(Channel length(number of bits per audio channel)) 0:16 位宽； 1:32 位宽。 只有在 DATLEN=00 时该位的写操作才有意义，否则声道长度都由硬件固定为 32 位。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。在 SPI 模式下不使用。

23.5.9 SPI_I²S 预分频寄存器(SPI_I2SPR)

地址偏移：0x20 复位值:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						MCKOE	ODD	I2SDIV							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:10	Reserved	保留位，硬件强制为 0
9	MCKOE	**MCKOE**：主设备时钟输出使能(Master clock output enable) 0:关闭主设备时钟输出； 1:主设备时钟输出使能。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。
8	ODD	**ODD**：奇系数预分频(Odd factor for the prescaler) 0:实际分频系数=I2SDIV*2； 1:实际分频系数=(I2SDIV*2)+1。参见 23.4.3 节。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。
7:0	I2SDIV	**I2SDIV**：I2S 线性预分频(I2S linear prescaler)禁止设置 I2SDIV[7:0]=0 或者 I2SDIV[7:0]=1 参见 23.4.3 节。 注：为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。在 SPI 模式下不使用。

23.5.10 寄存器地址映象

表 137 SPI 寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	SPI_CR1	保留																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXOnly	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	SPI_CR2	保留																								TXEIE	RXNEIE	ERRIE	保留		SSOE	TXDMAE	RXDMAE
	复位值																									0	0	0			0	0	0
008h	SPI_SR	保留																								BSY	OVR	MODF	CRCER	保留	TXE	RXNE	
	复位值																									0	0	0	0		1	0	
00Ch	SPI_DR	保留																DR[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	SPI_CRCPR	保留																CRCPOLY[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
014h	SPI_RXCRCR	保留																RxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	SPI_TXCRCR	保留																TxCRC[15:0]															
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	SPI_I2SCFGR	保留																				I2SMOD	I2SE	I2SCFG		PCMSYNC	保留	I2SSTD		CKPOL	DATLEN		CHLEN
	复位值																					0	0	0	0	0		0	0	0	0	0	0
020h	SPI_I2SPR	保留																						MCKOE	ODD	I2SDIV							
	复位值																							0	0	0	0	0	0	0	1	0	

关于寄存器的起始地址，参见表 1。

24 I2C 接口

24.1 I²C 简介

I²C(芯片间)总线接口连接微控制器和串行 I²C 总线。它提供多主机功能，控制所有 I²C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式，同时与 SMBus2.0 兼容。

I²C 模块有多种用途，包括 CRC 码的生成和校验、SMBus(系统管理总线—SystemManagementBus)和 PMBus(电源管理总线—PowerManagementBus)。

根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

24.2 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 100kHz)
  - 快速(高达 400kHz)
- 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 SMBus2.0
  - 25ms 时钟低超时延时
  - 10ms 主设备累积时钟低扩展时间
  - 25ms 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 SMBus

24.3 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。

24.3.1 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。

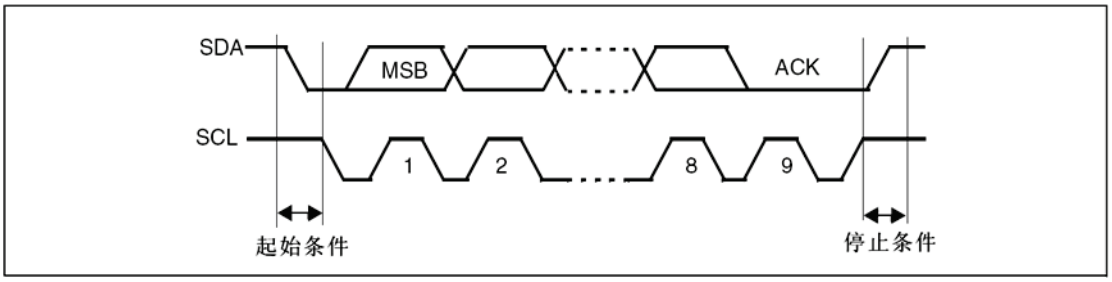


图 242 I²C 总线协议

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。

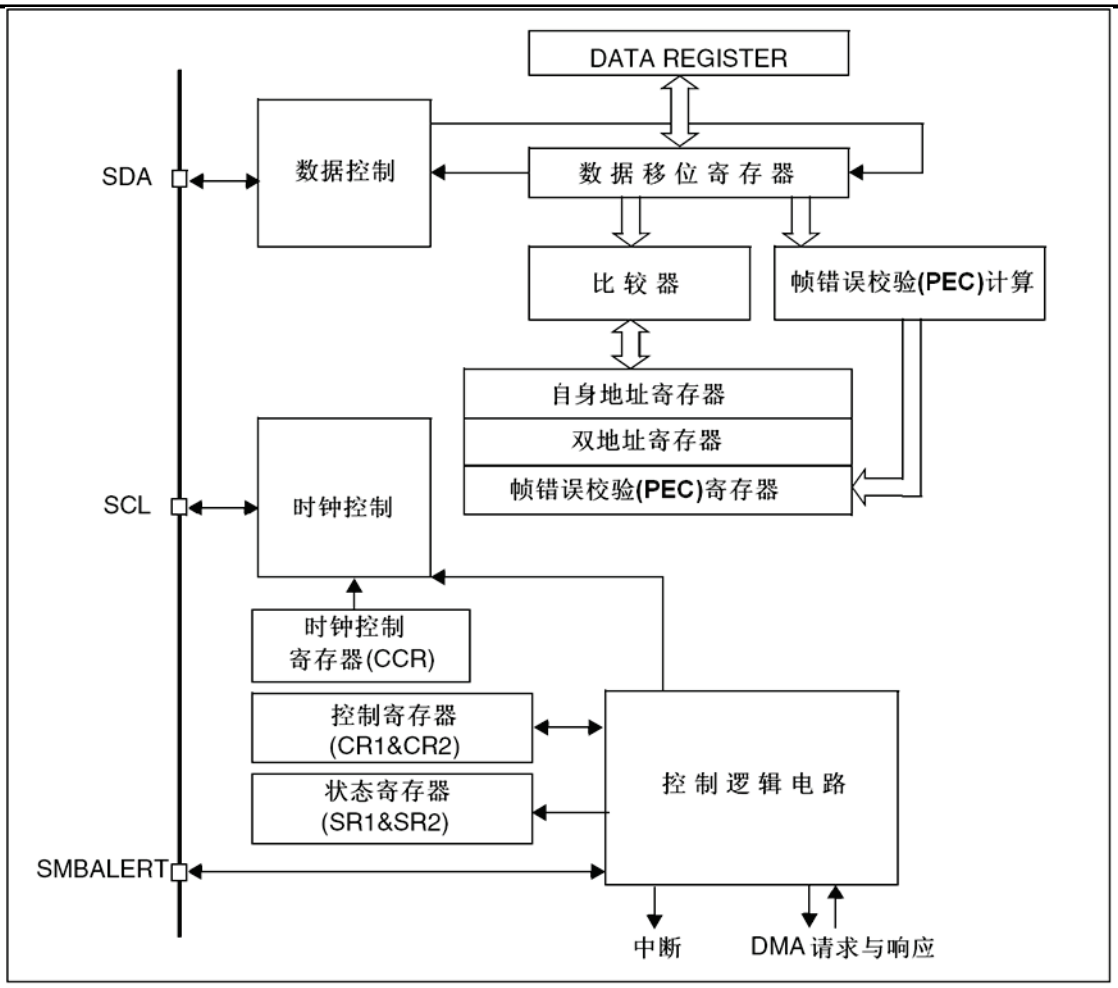


图 243 I²C 的功能框图

注：　在 SMBus 模式下，SMBALERT 是可选信号。如果禁止了 SMBus，则不能使用该信号。

24.3.2 I²C 从模式

默认情况下，I²C 接口总是工作在从模式。从从模式切换到主模式，需要产生一个起始条件。

为了产生正确的时序，必须在 I²C_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：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。

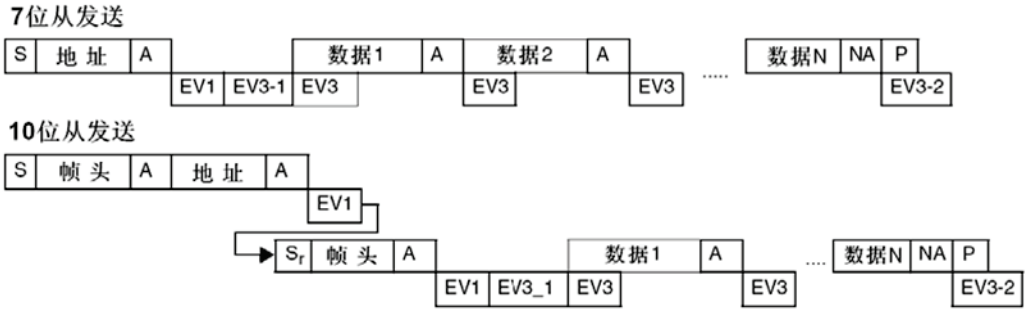


图 244 从发送器的传送序列图

说明:S=Start(起始条件)，S,=重复的起始条件，P=Stop(停止条件)，A=应，NA=非响应 EVx=事件(ITEVFEN=1 时产生中断)

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。

注: 1-EV1 和 EV3-1 事件拉长 SCL 低的时间，直到对应的软件序列结束。

2-EV3 的软件序列必须在当前字节传输结束之前完成。

从接收器

在接收到地址并清除 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)。

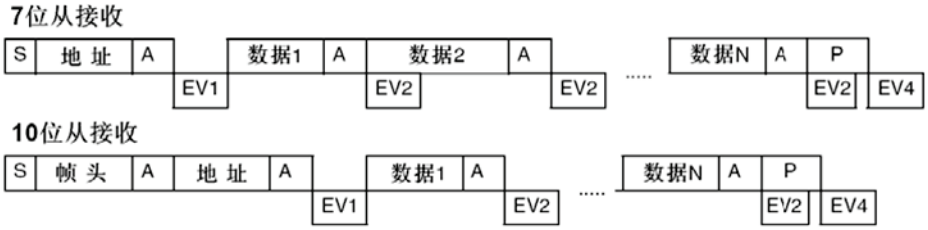


图 245 从接收器的传送序列图

说明:S=Start(起始条件)，S,=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)

EV1:ADDR=1，读 SR1 然后读 SR2 将清除该事件。

EV2:RxNE=1，读 DR 将消除该事件。

EV4:STOPF=1，读 SR1 然后写 CR1 寄存器将清除该事件。

注: 1-EV1 事件拉长 SCL 低的时间，直到对应的软件序列结束。

2-EV2 的软件序列必须在当前字节传输结束之前完成。

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)。

24.3.3 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

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(M/SL 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(见图 245 和图 246 的 EV5)。

从地址的发送

从地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  - 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(见图 246 和图 247)。

  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(见图 246 和图 247)。
- 在 7 位地址模式时，只需送出一个地址字节。
  - 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(见图 246 和图 247)。

根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在 7 位地址模式时，
  - 要进入发送器模式，主设备发送从地址时置最低位为'0'。
  - 要进入接收器模式，主设备发送从地址时置最低位为'1'。
- 在 10 位地址模式时
  - 要进入发送器模式，主设备先送头字节(11110xx0)，然后送最低位为'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

在发送了地址和清除了 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 TxE 被清除，(见图 246 的 EV8)。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(见图 246 的 EV8_2)，然后 I2C 接口将自动回到从模式(M/S 位清除)。

注：　当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8_2 事件时。

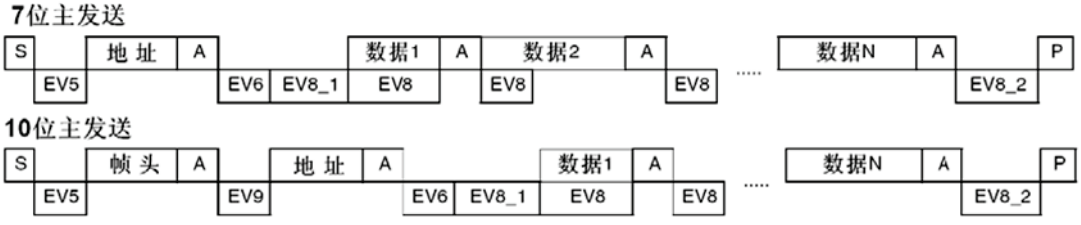


图 246 主发送器传送序列图

说明:S=Star(起始条件)，S;=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)。

EV5:SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件。

EV6:ADDR=1，读 SR1 然后读 SR2 将清除该事件。

EV8 1:TxE=1，移位寄存器空，数据寄存器空，写 DR 寄存器。

EV8:TxE=1，移位寄存器非空，数据寄存器空，写入 DR 寄存器将清除该事件。

EV8 2:TxE=1，BTF=1，请求设置停止位。TxE 和 BTF 位由硬件在产生停止条件时清除。

EV9:ADDR10=1，读 SR1 然后写入 DR 寄存将清除该事件。

注：　1-EV5、EV6、EV9、EV8 1 和 EV8 2 事件拉长 SCL 低的时间，直到对应的软件序列结束。

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：

- 如果 ACK 位被置位，发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断(见图 247bookmark2040 的 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

主设备在从从设备接收到最后一个字节后发送一个 NACK。接收到 NACK 后，从设备释放对 SCL 和 SDA 线的控制；主设备就可以发送一个停止/重起始条件。

- 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)必须清除 ACK 位。
- 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置 STOP/START 位。
- 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(M/SL 位被清除)。

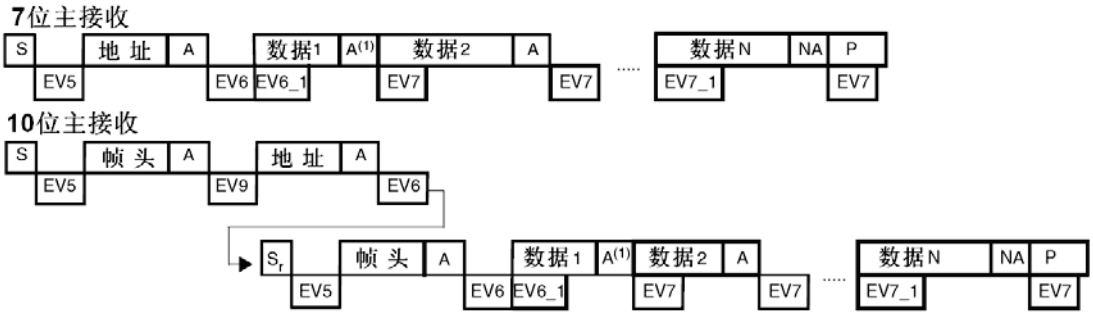


图 247 主接收器传送序列图

说明:S=Start(起始条件)，S:=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件(ITEVFEN=1 时产生中断)

EV5:SB=1，读 SR1 然后将地址写入 DR 寄存器将除该事件。EV6:ADDR=1，读 SR1 然后读 SR2 将清除该事件。在 10 位主接收模式下，该事件后应设置 CR2 的 START=1.EV6 1:没有对应的事件标志，只适于接收 1 个字节的情况。恰好在 EV6 之后(即清除了 ADDR 之后)，要洁除响应和停止条件的产生位。EV7:RxNE=1，读 DR 寄存器清除该事件。

EV7 1:RXNE=1，读 DR 寄存器清除该事件。设置 ACK=0 和 STOP 请求。

EV9:ADDR10=1，读 SR1 然后写入 DR 寄存将清除该事件。

1. 如果收到一个单独的字节，则是 NA。
2. EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。
3. EV7 的软件序列必须在当前字节传输结束前完成。
4. EV6_1 或 EV7_1 的软件序列必须在当前传输字节的 ACK 脉冲之前完成。

24.3.4 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。

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(M/SL 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。

24.3.5 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。

24.3.6 SMBus

介绍

系统管理总线(SMBus)是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I²C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线(SMBus)标准涉及三类设备。从设备：接收或响应命令的设备。主设备：用来发送命令、产生时钟和终止发送的设备。主机：一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能并且必须支持 SMBus 提醒协议。一个系统里只允许有一个主机。

SMBus 和 I²C 之间的相似点

- 2 条线的总线协议(1 个时钟，1 个数据)+可选的 SMBus 提醒线；
- 主-从通信，主设备提供时钟；
- 多主机功能
- SMBus 数据格式类似于 I²C 的 7 位地址格式(见图 242)；

SMBus 和 I2C 之间的不同点

下表列出了 SMBus 和 I²C 的不同点。

表 138 SMBus 与 I²C 的比较

SMBus	I2C
最大传输速度 100kHz	最大传输速度 400kHz
最小传输速度 10kHz	无最小传输速度
35ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 VDD 决定
不同的地址类型(保留的、动态的等)	7 位、10 位和广播呼叫从地址类型
不同的总线协议(快速命令、处理呼叫等)	无总线协议

SMBus 应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的 SMBus 规范(http://smbus.org/specs/)。

总线协议

SMBus 技术规范支持 9 个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考 2.0 版的 SMBus 规范(http://smbus.org/specs/)。这些协议由用户的软件来执行。

地址解析协议(ARP)

通过给每个从设备动态地分配一个新的唯一地址，可以解决 SMBus 的从地址冲突。地址解析协议(ARP)具有以下的特性：

- 使用标准 SMBus 物理层仲裁机制分配地址；
- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)；
- 任何一个 SMBus 主设备可以遍历总线。

唯一的设备标识符(UDID)

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。
关于在 ARP 上 128 位的 UDID 的详细信息，参考 2.0 版的 SMBus 规范(http://smbus.org/specs/)。

SMBus 提醒模式

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL、SDA 信号一样，是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。
一个只具有从功能的设备，可以通过设置 I2C_CR1 寄存器上的 ALERT 位，使用 SMBALERT 给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址 ARA(Alert Response Address，地址值为 0001 100x)访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2C_SR1 寄存器中的 SMBALERT 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上，第八个位可以是'0'或'1'。
如果多个设备把 SMBALERT 拉低，最高优先级设备(最小的地址)将在地址传输期间通过标准仲裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。
没有实现 SMBALERT 信号的主机可以定期访问 ARA。
有关 SMBus 提醒模式的更多详细资料，请参考 2.0 版的 SMBus 规范(http://smbus.org/specs/)。

超时错误

在定时规范上 I^2C 和 SMBus 之间有很多差别。

SMBus 定义了一个时钟低超时，35ms 的超时。SMBus 规定 TLOW:SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW:MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范(http://smbus.org/specs/)。

I2C_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。

如果要把设备配置成主设备，产生起始条件的步骤见 24.3.3 节 I²C 主模式。否则，参见 22.3.2 节 I²C 从模式。

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 主设备头字段。
- 如果 SMBALERT=1，使用 SMB 提醒响应地址。

24.3.7 DMA 请求

DMA 请求(当被使能时)仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 ETO 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，数据将由 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_CCTx 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。

24.3.8 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。

24.4 I²C 中断请求

下表列出了所有的 I²C 中断请求

表 139 I²C 中断请求表:

中断事件	事件标志	开启控制位
起始位已发送(主)	SB	ITEVFEN
地址已发送(主)或地址匹配(从)	ADDR	
10 位头段已发送(主)	ADD10	
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	ITERREN
仲裁丢失(主)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

注: 1.SB、ADDR、ADD10、STOPF、BTF、RxNE 和 TxE 通过逻辑或汇到同一个中断通道中。

2.BERR、ARLO、AF、OVR、PECERR、TIMEOUT 和 SMBALERT 通过逻辑或汇到同一个中断通道中。

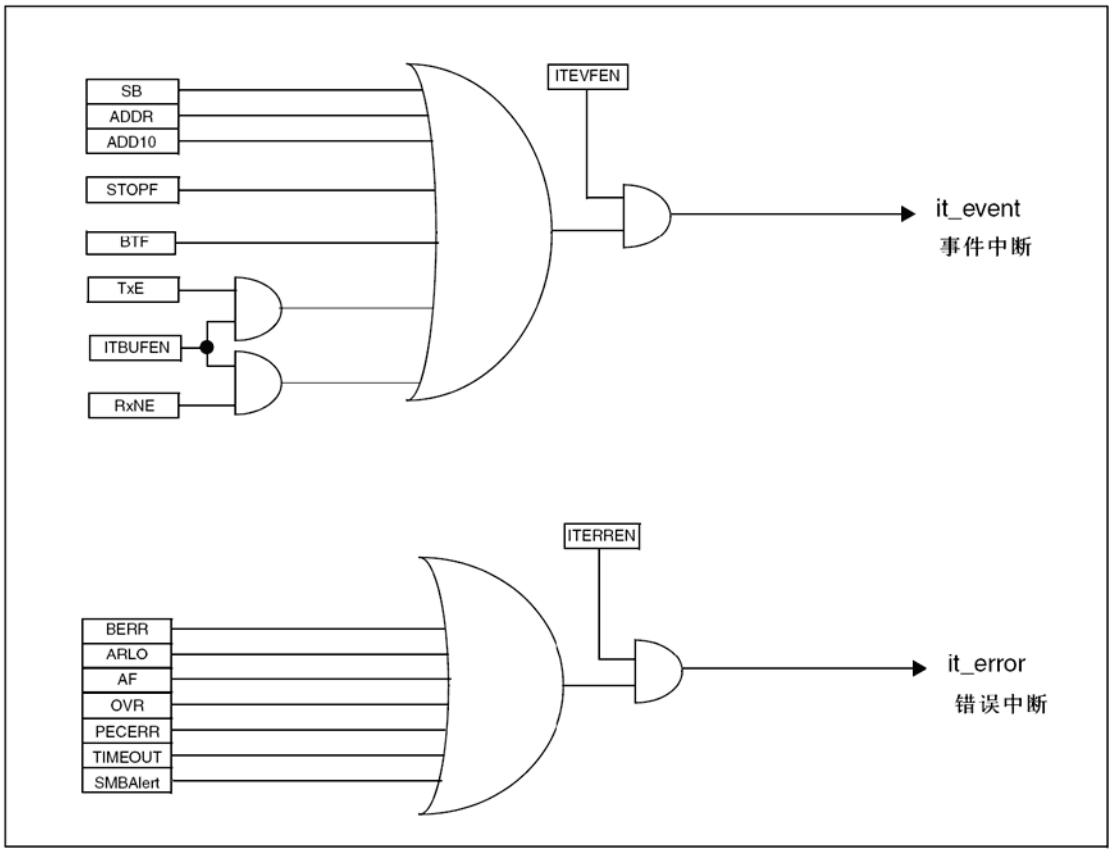


图 248 I²C 中断映射图

24.5 I²C 调试模式

当微控制器进入调试模式(Cortex-M3 核心处于停止状态)时，根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。详见第 28.16.2 节。

24.6 I²C 寄存器描述

关于在寄存器描述里面所用到的缩写，详见第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

24.6.1 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	保留	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMBTYPE	保留	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

位	符号	说明
15	SWRST	**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：该位可以用于 BUSY 位为'1'，在总线上又没有检测到停止条件时。
14	Reserved	保留，始终读为 0。
13	ALERT	**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。
12	PEC	**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。
11	POS	**POS**:应答/PEC 位置(用于数据接收)(Acknowledge/PEC Position(for data reception))软件可以设置或清除该位，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：POS 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。 为了 NACK 第 2 个字节，必须在清除 ADDR 为之后清除 ACK 位。 为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，拉伸 ADDR 事件时设置 PEC 位。
10	ACK	**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)。
9	STOP	**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。 注：当设置了 STOP、START 或 PEC 位，在硬件清除这个位之前，软件不要执行任何对 I2C_CR1 的写操作；否则有可能会第 2 次设置 STOP、START 或 PEC 位。
8	**START**	**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。
7	**NOSTRETCH**	**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。
6	**ENGC**	**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。
5	**ENPEC**	**ENPEC**：PEC 使能(PEC enable) 0：禁止 PEC 计算； 1：开启 PEC 计算。
4	**ENARP**	**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。
3	**SMBTYPE**	**SMBTYPE**：SMBus 类型(SMBus type) 0：SMBus 设备； 1：SMBus 主机。
2	**Reserved**	保留位，硬件强制为 0。
1	**SMBUS**	**SMBUS**：SMBus 模式(SMBus mode) 0：I2C 模式； 1：SMBus 模式。
0	**PE**	**PE**：I2C 模块使能(Peripheral enable) 0：禁用 I2C 模块； 1：启用 I2C 模块：根据 SMBus 位的设置，相应的 I/O 口需配置为复用功能。 注：如果清除该位时通讯正在进行，在当前通讯结束后，I2C 模块被禁用并返回空闲状态。由于在通讯结束后发生 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。

24.6.2 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

保留	LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留	FREQ[5:0]
	rw	rw	rw	rw	rw		rw rw rw rw rw rw

位	符号	说明
15:13	Reserved	保留位，硬件强制为 0
12	LAST	**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。
11	DMAEN	**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。
10	ITBUFEN	**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)。
9	ITEVTEN	**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。
8	ITERREN	**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。
7:6	Reserved	保留位，硬件强制为 0。
5:0	FREQ[5:0]	**FREQ[5:0]**：I2C 模块时钟频率(Peripheral clock frequency) 必须设置正确的输入时钟频率以产生正确的时序，允许的范围在 2 ~ 36MHz 之间： 000000：禁用 000001：禁用 000010：2MHz … 100100：36MHz 大于 100100：禁用。

24.6.3 自身地址寄存器 1(I2C_OAR1)

复位地址偏移：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	保留	保留				ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15	ADDMODE	**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)。
14	Reserved	必须始终由软件保持为'1'。
13:10	Reserved	保留位，硬件强制为 0。
9:8	ADD[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。
7:1	ADD[7:1]	**ADD[7:1]**：接口地址(Interface address)地址的 7~1 位。
0	ADD0	**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。

24.6.4 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
保留								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:8	Reserved	保留位，硬件强制为 0
7:1	ADD2[7:1]	**ADD2[7:1]**：接口地址(Interface address)在双地址模式下地址的 7~1 位。
0	ENDUAL	**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。

24.6.5 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
保留								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15:8	Reserved	保留位，硬件强制为 0
7:0	DR[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。

24.6.6 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
SMBALERT	TIMEOUT	保留	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	保留	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

位	符号	说明
15	SMBALERT	**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。
14	TIMEOUT	**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。
13	Reserved	保留位，硬件强制为 0。
12	PECERR	**PECERR**：在接收时发生 PEC 错误(PEC Error inreception) 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。
11	OVR	**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。
10	AF	**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。
9	ARLO	**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，I2C 接口自动切换回从模式(M/SL=0)。 注：在 SMBUS 模式下，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间(不包括地址的应答)。
8	BERR	**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。
7	TxE	**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。
6	RxNE	**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。
5	Reserved	保留位，硬件强制为 0
4	STOPF	**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。
3	ADD10	**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。
2	BTF	**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。
1	ADDR	**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)。 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。
0	**SB**	**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。

24.6.7 状态寄存器 2(I2C_SR2)

地址偏移：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMB DEFAULT	GENCALL	保留	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

位	符号	说明
15:8	**PEC[7:0]**	**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。
7	**DUALF**	**DUALF**：双标志(从模式)(Dualflag(Slave mode)) 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。
6	**SMBHOST**	**SMBHOST**:SMBus 主机头系列(从模式)(SMBus host header(Slave mode)) 0：未收到 SMBus 主机的地址； 1：当 SMBTYPE=1 且 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。
5	**SMBDEFAULT**	**SMBDEFAULT**：SMBus 设备默认地址(从模式)(SMBus device default address(Slave mode)) 0：未收到 SMBus 设备的默认地址； 1：当 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。
4	**GENCALL**	**GENCALL**：广播呼叫地址(从模式)(General call address(Slave mode)) 0：未收到广播呼叫地址； 1：当 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。
3	**Reserved**	保留位，硬件强制为 0
2	**TRA**	**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。
1	**BUSY**	**BUSY**：总线忙(Bus busy) 0：在总线上无数据通讯； 1：在总线上正在进行数据通讯。 –在检测到 SDA 或 SCl 为低电平时，硬件将该位置'1'； –当检测到一个停止条件时，硬件将该位清除。 该位指示当前正在进行的总线通讯，当接口被禁用(PE=0)时该信息仍然被更新。
0	**MSL**	**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。

24.6.8 时钟控制寄存器(I2C_CCR)

地址偏移：0x1C

复位值：0x0000

注： *1.要求 FPCLK1 应当是 10MHz 的整数倍，这样可以正确地产生 400KHz 的快速时钟。*

2.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	保留		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
15	**F/S**	**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。
14	**DUTY**	**DUTY**：快速模式时的占空比(Fast mode duty cycle) 0：快速模式下：Tlow/Thigh=2； 1:快速模式下：Tlow/Thigh=16/9(见 CCR)。
13:12	**Reserved**	保留位，硬件强制为 0。
11:0	**CCR[11:0]**	**CCR[11:0]**：快速/标准模式下的时钟控制分频系数(主模式)(ClockcontrolregisterinFast/Standardmode(Mastermode)) 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： Thigh=CCR×TPCLK1Tlow=CCR×TPCLK1 在 I2C 快速模式下：如果 DUTY=0： Thigh=CCR×TPCLK1 Tlow=2×CCR×TPCLK1 如果 DUTY=1：(速度达到 400kHz)Thigh=9×CCR×TPCLK1 Tlow=16×CCR×TPCLK1 例如：在标准模式下，产生 100kHz 的 SCL 的频率： 如果 FREQR=08，TPCLK1=125ns，则 CCR 必须写入 0x28(40×125ns=5000ns)。 注：1.允许设定的最小值为 0x04，在快速 DUTY 模式下允许的最小值为 0x01； 2.Thigh=tr(SCL)+tw(SCLH)，详见数据手册中对这些参数的定义； 3.Tlow=tf(SCL)+tw(SCLL)，详见数据手册中对这些参数的定义； 4.这些延时没有过滤器； 5.只有在关闭 I2C 时(PE=0)才能设置 CCR 寄存器；

		6.fCK 应当是 10MHz 的整数倍，这样可以正确产生 400kHz 的快速时钟。

24.6.9 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
保留										TRISE[5:0]					
										rw	rw	rw	rw	rw	rw

位	符号	说明
15:6	Reserved	保留位，硬件强制为 0
5:0	TRISE[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)) 这些位必须设置为 I2C 总线规范里给出的最大的 SCL 上升时间，增长步幅为 1。 例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CR2 寄存器中 FREQ[5:0]中的值等于 0x08 且 TPCLK1=125ns，故 TRISE[5:0]中必须写入 09h(1000ns/125ns=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 tHIGH 参数。 注：只有当 I2C 被禁用(PE=0)时，才能设置 TRISE[5:0]。

24.6.10 I2C 寄存器地址映象

表 140 I2C 寄存器地址映象和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	保留																SWRST	保留	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	保留	SMBUS	PE
	复位值																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	保留																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	保留	FREQ[5:0]						
	复位值																				0	0	0	0	0		0	0	0	0	0	0
0x08	I2C_OAR1	保留																ADDMODE	保留	保留				ADD [9:8]		ADD[7:1]							ADD0
	复位值																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	保留																								ADD2[7:1]							ENDUAL
	复位值																									0	0	0	0	0	0	0	0
0x10	I2C_DR	保留																								DR[7:0]							
	复位值																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	保留																SMBALERT	TIMEOUT	保留	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
	复位值																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	保留																PEC[7:0]								DUALF	SMBHOST	SMBDEFAU	GENCALL	保留	TRA	BUSY	MSL
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	保留																F/S	DUTY	保留	CCR[11:0]												
	复位值																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	保留																										TRISE[5:0]					
	复位值																											0	0	0	0	0	0

关于寄存器起始地址，参见表 1。

25 通用同步异步收发器(USART)

25.1 USART 介绍

通用同步异步收发器(USART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。
它支持同步单向通信和半双工单线通信，也支持 LIN(局部互连网)，智能卡协议和 IrDA(红外数据组织)SIRENDEC 规范，以及调制解调器(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。

25.2 USART 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
  - 发送和接收共用的可编程波特率，最高达 4.5Mbits/s
- 可编程数据字长度(8 位或 9 位)
- 可配置的停止位-支持 1 或 2 个停止位
- LIN 主发送同步断开符的能力以及 LIN 从检测断开符的能力
  - 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 发送方为同步传输提供时钟
- IRDASIR 编码器解码器
  - 在正常模式下支持 3/16 位的持续时间
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3 标准里定义的异步智能卡协议
  - 智能卡用到的 0.5 和 1.5 个停止位
- 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
- 10 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
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪音错误
  - 校验错误
- 多处理器通信--如果地址不匹配，则进入静默模式
- 从静默模式中唤醒(通过空闲总线检测或地址标志检测)
- 两种唤醒接收器的方式：地址位(MSB，第 9 位)，总线空闲

25.3 USART 功能概述

接口通过三个引脚与其他设备连接在一起(见图 249)。任何 USART 双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

RX：接收数据串行输。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，并且不发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同时用于数据的发送和接收。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(8 或 9 位)，最低有效位在前
- 0.5，1.5，2 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(USART_SR)
- 数据寄存器(USART_DR)
- 一个波特率寄存器(USART_BRR)，12 位的整数和 4 位小数
- 一个智能卡模式下的保护时间寄存器(USART_GTPR)

关于以上寄存器中每个位的具体定义，请参考寄存器描述第 25.6 节：USART 寄存器描述。在同步模式中需要下列引脚：

- CK：发送器时钟输出。此引脚输出用于同步传输的时钟，(在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲)。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备(例如 LCD 驱动器)。时钟相位和极性都是软件可编程的。在智能卡模式里，CK 可以为智能卡提供时钟。在 IrDA 模式里需要下列引脚：
- IrDA_RDI:IrDA 模式下的数据输入。
- IrDA_TDO:IrDA 模式下的数据输出。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 USART 准备好接收数据

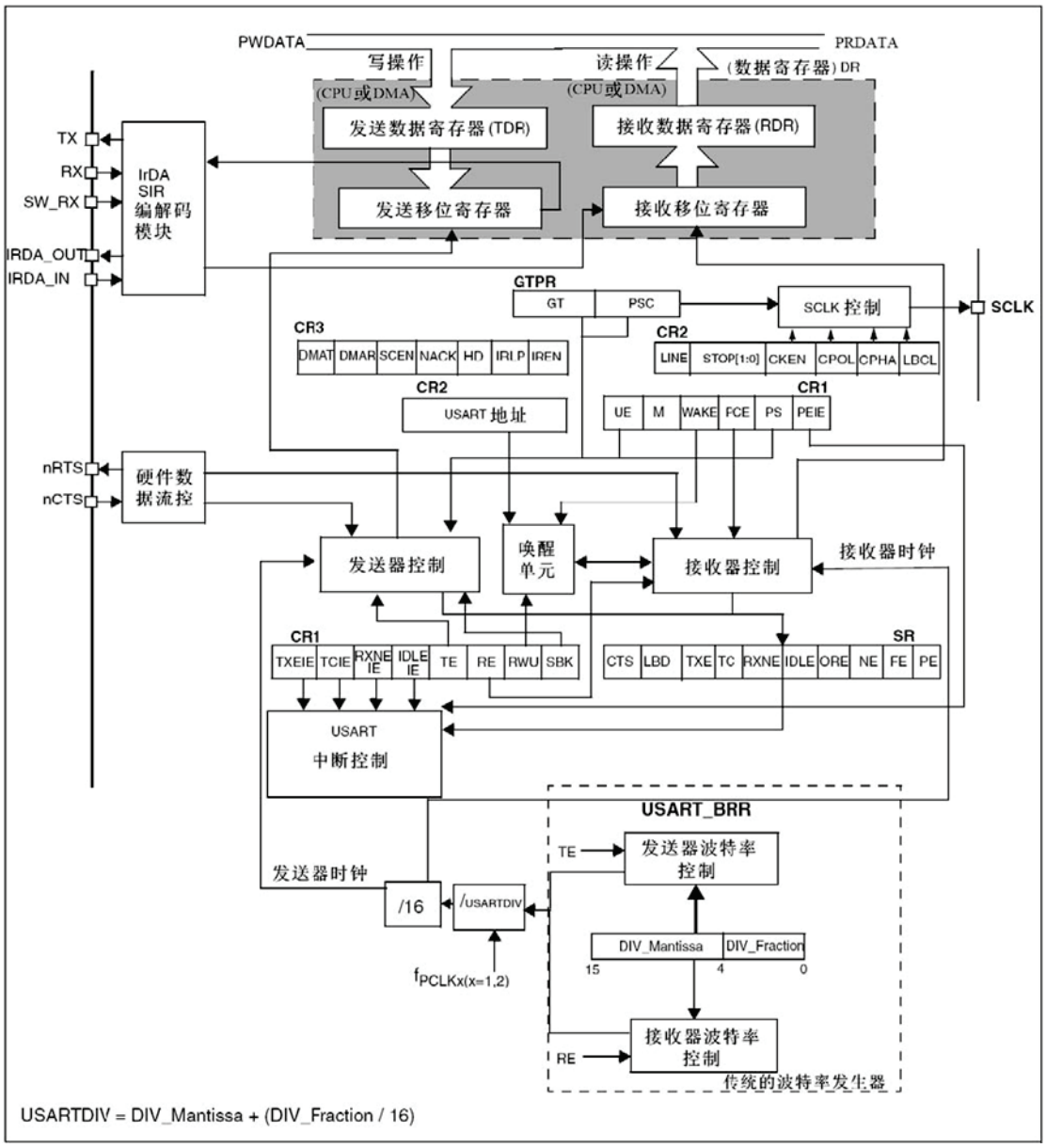


图 249 USART 框图

25.3.1 USART 特性描述

字长可以通过编程 USART_CR1 寄存器中的 M 位，选择成 8 或 9 位(见图 250)。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由'1'组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位('1'的位数也包括了停止位的位数)。

断开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。

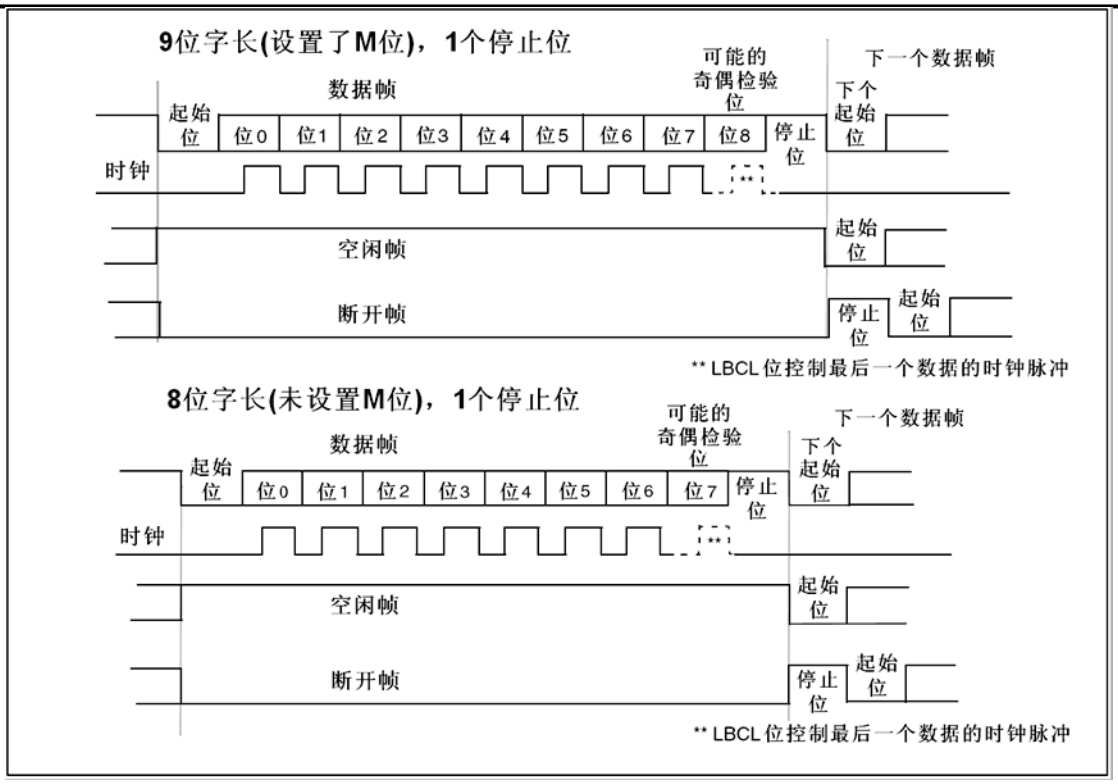


图 250 字长设置

25.3.2 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，相应的时钟脉冲在 CK 脚上输出。

字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，USART_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(见图 249)。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

注： *1．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。*

2．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 USART 模式、单线模式以及调制解调器模式。
3. 0.5 个停止位：在智能卡模式下接收数据时使用。
4. 1.5 个停止位：在智能卡模式下发送和接收数据时使用。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位(当 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)。

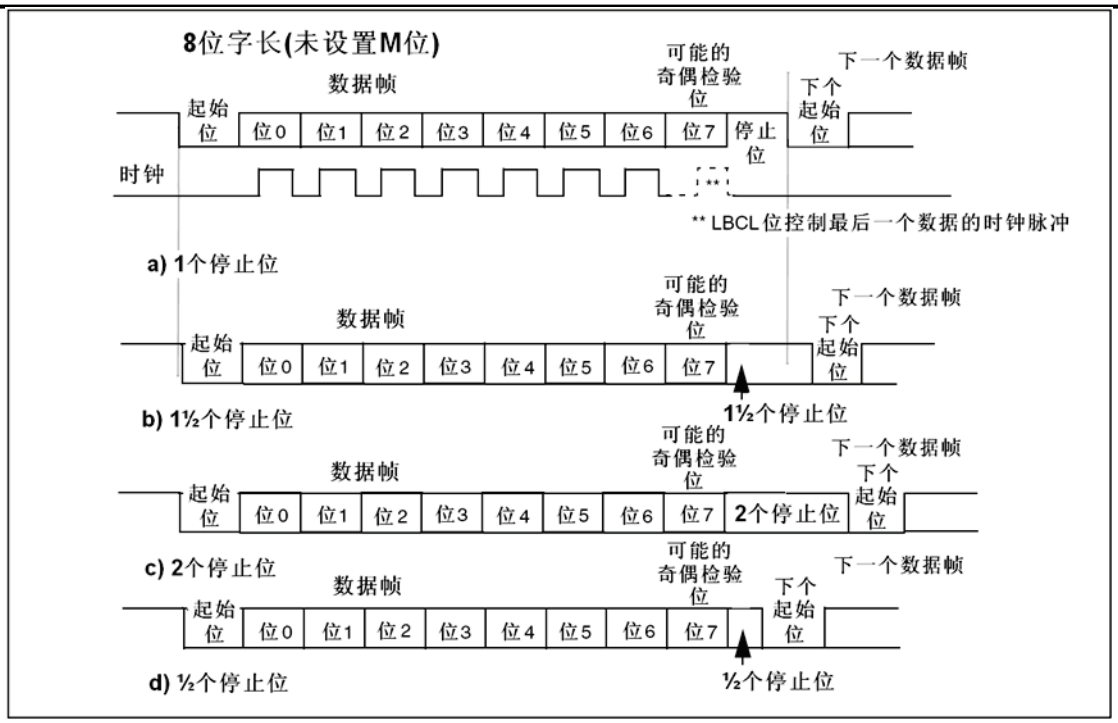


图 251 配置停止位

配置步骤:

1. 通过在 USART_CR1 寄存器上置位 UE 位来激活 USART
2. 编程 USART_CR1 的 M 位来定义字长。
3. 在 USART_CR2 中编程停止位的位数。
4. 如果采用多缓冲器通信，配置 USART_CR3 中的 DMA 使能位(DMAT)。按多缓冲器通信中的描述配置 DMA 寄存器。
5. 利用 USART_BRR 寄存器选择要求的波特率。
6. 设置 USART_CR1 中的 TE 位，发送一个空闲帧作为第一次数据发送。
7. 把要发送的数据写进 USART_DR 寄存器(此动作清除 TXE 位)。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 7。
8. 在 USART_DR 寄存器中写入最后一个数据字后，要等待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次传输。

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 USART_DR 寄存器而不会覆盖先前的数据如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USART_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART_DR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE 位立即被置起。

当一帧发送完成时(停止位发送后)并且设置了 TXE 位，TC 位被置起，如果 USART_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

在 USART_DR 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式(详见下图)之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1．读一次 USART_SR 寄存器；

2．写一次 USART_DR 寄存器。

注：　TC 位也可以通过软件对它写0'来清除。此清零方式只推荐在多缓冲器通信模式下使用。

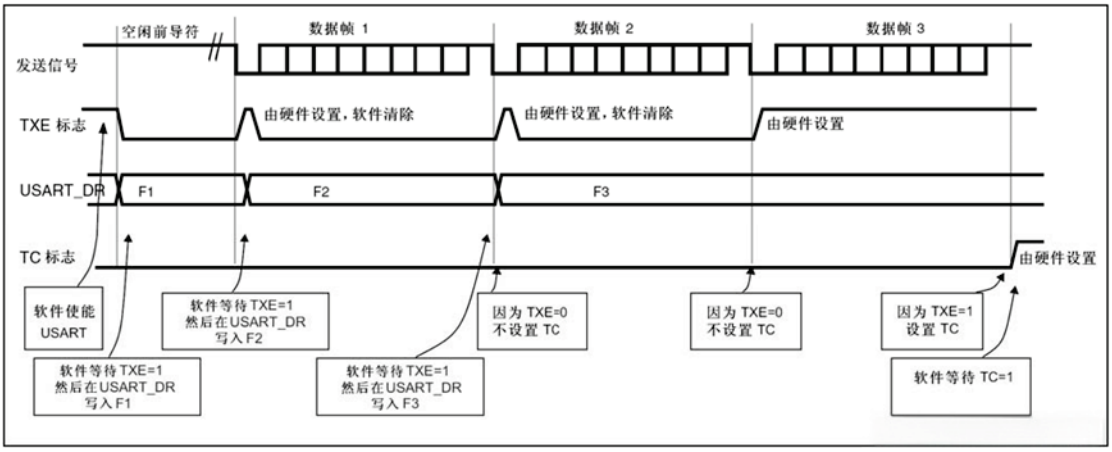


图 252 发送时 TC/TXE 的变化情况

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(见图 250)。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑'1'，以保证能识别下一帧的起始位。

注意：　如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置起。

空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

25.3.3 接收器

USART 可以根据 USART_CR1 的 M 位接收 8 位或 9 位的数据字。

起始位侦测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。

该序列为：1110X0X0X0000

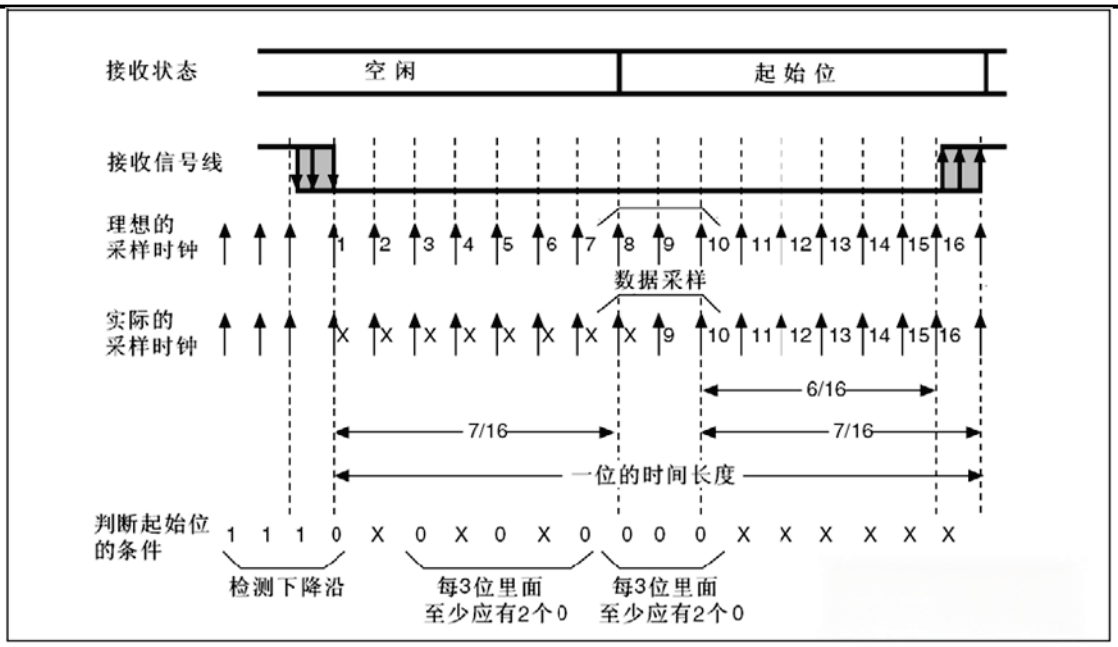


图 253 起始位侦测

注意： *如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)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 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USART_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤:

1. 将 USART_CR1 寄存器的 UE 置 1 来激活 USART。
2. 编程 USART_CR1 的 M 位定义字长
3. 在 USART_CR2 中编写停止位的个数
4. 如果需多缓冲器通信，选择 USART_CR3 中的 DMA 使能位(DMAR)。按多缓冲器通信所要求的配置 DMA 寄存器。
5. 利用波特率寄存器 USART_BRR 选择希望的波特率。
6. 设置 USART_CR1 的 RE 位。激活接收器，使它开始寻找起始位。当一字符被接收到时，

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出(包括与之有关的错误标志)。
- 如果 RXNEIE 位被设置，产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起，
- 在多缓冲器通信时，RXNE 在每个字节接收后被置起，并由 DMA 对数据寄存器的读操作而清零。

- 在单缓冲器模式里，由软件读 USART_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

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

断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

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 USART_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 ORE 位

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

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。
- 如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的(也就是丢失的)数据时，此种情况可能发生。在读序列期间(在 USART_SR 寄存器读访问和 USART_DR 读访问之间)接收到新的数据，此种情况也可能发生。

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。

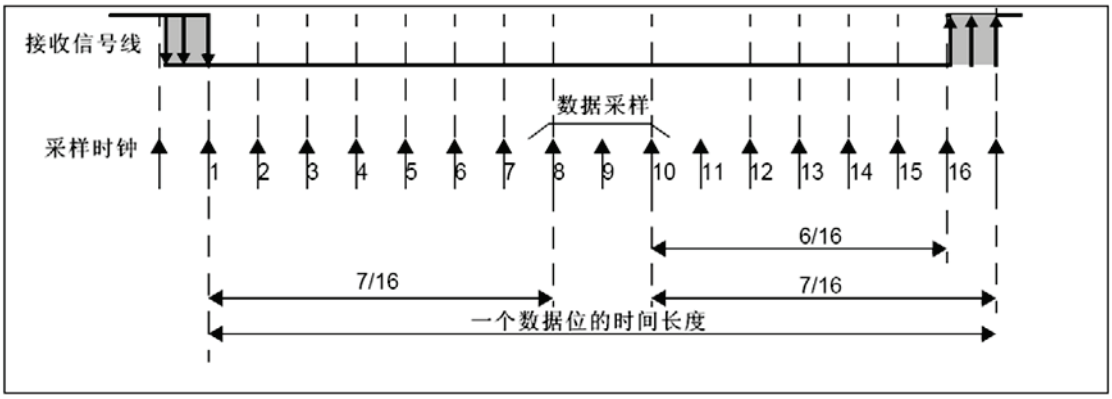


图 254 检测噪声的数据采样

表 141 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 USART_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，因为 NE 标志位和 RXNE 标志位是同时被设置，RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USART_CR3 寄存器中 EIE 位，将产生一个中断。

先读出 USART_SR，再读出 USART_DR 寄存器，将清除 NE 标志位

帧错误

当以下情况发生时检测到帧错误：

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。

当帧错误被检测到时：

- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART_DR 寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 位同时置起，后者将产生中断。在多缓冲器通信情况下，如果 USART_CR3 寄存器中 EIE 位被置位的话，将产生中断。

顺序执行对 USART_SR 和 USART_DR 寄存器的读操作，可复位 FE 位。

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，在智能卡模式里可能是 0.5 或 1.5 个。

1. 0.5 个停止位(智能卡模式中的接收)：不对 0.5 个停止位进行采样。因此，如果选择 0.5 个停止位则不能检测帧错误和断开帧。
2. 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行。
3. 1.5 个停止位(智能卡模式)：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活(USART_CR1 寄存器中的 RE=1)，并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样 NACK 信号时，即总线上停止位对应的时间内时，拉低数据线，以此表示出现了帧错误。FE 在 1.5 个停止位结束时和 RXNE 一起被置起。对 1.5 个停止位的采样是在第 16，第 17 和第 18 采样点进行的。1.5 个的停止位可以被分成 2 部分：一个是 0.5 个时钟周期，期间不做任何事情。随后是 1 个时钟周期的停止位，在这段时间的中点处采样。详见第 25.3.11 节：智能卡。
4. 2 个停止位：对 2 个停止位的采样是在第一停止位的第 8，第 9 和第 10 个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时 RXNE 标志将被设置。

25.3.4 分数波特率的产生

接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx/Rx波特率} = \frac{f_{ck}}{(16 * \text{USARTDIV})}$$

这里的 fCK 是给外设的时钟(PCLK1 用于 USART2、3、4、5，PCLK2 用于 USART1)

USARTDIV 是一个无符号的定点数。这 12 位的值设置在 USART_BRR 寄存器。

注：在写入 USART_BRR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

如何从 USART_BRR 寄存器值得到 USARTDIV

例 1:

如果 DIV_Mantissa=0d27，DIV_Fraction=0d12(USART_BRR=0x1BC)，

于是

Mantissa(USARTDIV)=0d27

Fraction(USARTDIV)=12/16=0d0.75

所以 USARTDIV=0d27.75

例 2:

要求 USARTDIV=0d25.62，

就有:

DIV_Fraction=16*0.62=0d9.92

最接近的整数是：10=0x0A

DIV_Mantissa=mantissa(0d25.620)=0d25=0x19

于是，USART_BRR=0x19A 因此 USARTDIV = 0d25.625

例 3:

要求 USARTDIV=0d50.99

就有:

DIV_Fraction=16*0d0.99=0d15.84

最接近的整数是：16=0x10=>DIV_frac[3:0]溢出=>进位必须加到小数部分

DIV_Mantissa=mantissa(0d50.990+进位)=0d51=0x33

于是：USART_BRR=0x330，USARTDIV=0d51.000

表 142 设置波特率时的误差计算

波特率		fPCLK=36MHz			fPCLK=72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%

10	4500	不可能	不可能	不可能	4500	1	0%

注：1.CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。

2.只有 USART1 使用 PCLK2(最高 72MHz)。其它 USART 使用 PCLK1(最高 36MHz)。

25.3.5 USART 接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化(包括发送器端振荡器的变化)
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化(通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成)。

需要满足：DTRA+DQUANT+DREC+DTCL<USART 接收器的容忍度

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USART_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 143 当 DIV_Fraction=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.75%	4.375%
1	3.41%	3.97%

表 144 当 DIV_Fraction!=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.33%	3.88%
1	3.03%	3.53%

注：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。

25.3.6 多处理器通信

通过 USART 可以实现多处理器通信(将几个 USART 连在一个网络里)。例如某个 USART 设备可以是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。

未被寻址的设备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。
- USART_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

根据 USART_CR1 寄存器中的 WAKE 位状态，USART 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

空闲总线检测(WAKE=0)

当 RWU 位被写 1 时，USART 进入静默模式。当检测到一空闲帧时，它被唤醒。然后 RWU 被硬件清零，但是 USART_SR 寄存器中的 IDLE 位并不置起。RWU 还可以被软件写 0。下图给出利用空闲总线检测来唤醒和进入静默模式的一个例子

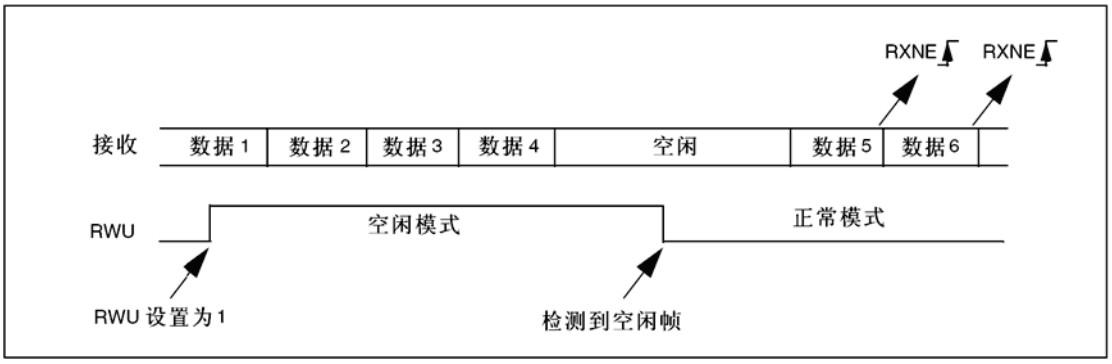


图 255 利用空闲总线检测的静默模式

地址标记(addressmark)检测(WAKE=1)

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART_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，USART 进入静默模式。此时，硬件设置 RWU 位。接收该字节既不会设置 RXNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时(USART_SR 的 RXNE=0)，RWU 位可以被写 0 或 1。否则，该次写操作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

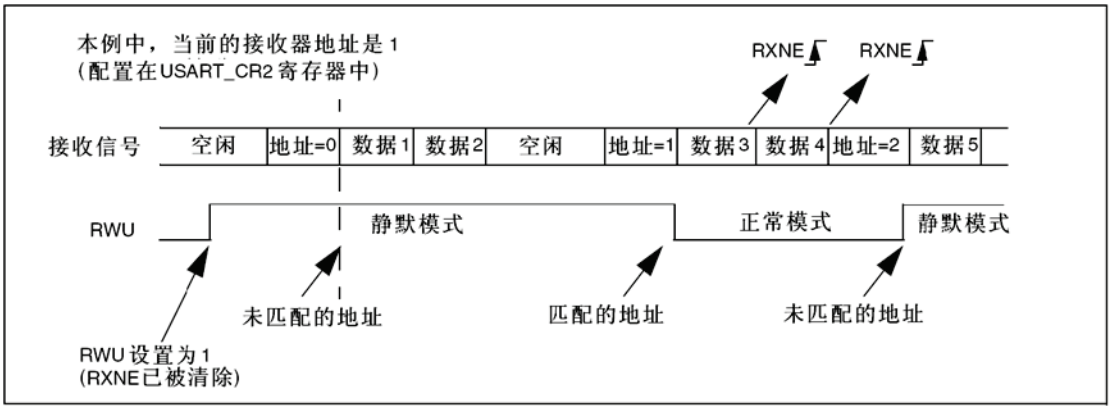


图 256 利用地址标记检测的静默模式

25.3.7 校验控制

设置 USART_CR1 寄存器上的 PCE 位，可以使能奇偶控制(发送时生成一个奇偶位，接收时进行奇偶校验)。根据 M 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 145 帧格式

M 位	PCE 位	USART 帧
0	0	|起始位|8 位数据|停止位|
0	1	|起始位|7 位数据|奇偶检验位|停止位|
1	0	|起始位|9 位数据|停止位|
1	1	|起始位|8 位数据|奇偶检验位|停止位|

注意：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 USART_CR1 中的 PS = 0)，校验位将是'0'。

奇校验：此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为奇数。

例如：数据=00110101，有 4 个'1'，如果选择奇校验(在 USART_CR1 中的 PS = 1)，校验位将是'1'。

传输模式：如果 USART_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去(如果选择偶校验偶数个'1'，如果选择奇校验奇数个'1')。如果奇偶校验失败，USART_SR 寄存器中的 PE 标志被置'1'，并且如果 USART_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

25.3.8 LIN(局域互联网)模式

LIN 模式是通过设置 USART_CR2 寄存器的 LINEN 位选择。在 LIN 模式下，下列位必须保持为 0：

- USART_CR2 寄存器的 CLKEN 位
- USART_CR3 寄存器的 STOP[1:0]，SCEN，HDSEL 和 IREN

LIN 发送

25.3.2 节里所描述的同样步骤适用于 LIN 主发送，但和正常 USART 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 USART 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(USART_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 USART_CR2 的 LBDL=0)或 11 个(当 USART_CR2 的 LBDL=1)连续位都是'0'，并且又跟着一个定界符，USART_SR 的 LBD 标志被设置。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 USART 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 USART 模式，不会进行断开检测。如果 LIN 模式被激活(LINEN=1)，只要一发生帧错误(也就是停止位检测到'0'，这种情况出现在断开帧)，接收器就停止，直到断开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)。

图 257 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。

图 258 给出了一个断开帧的例子。

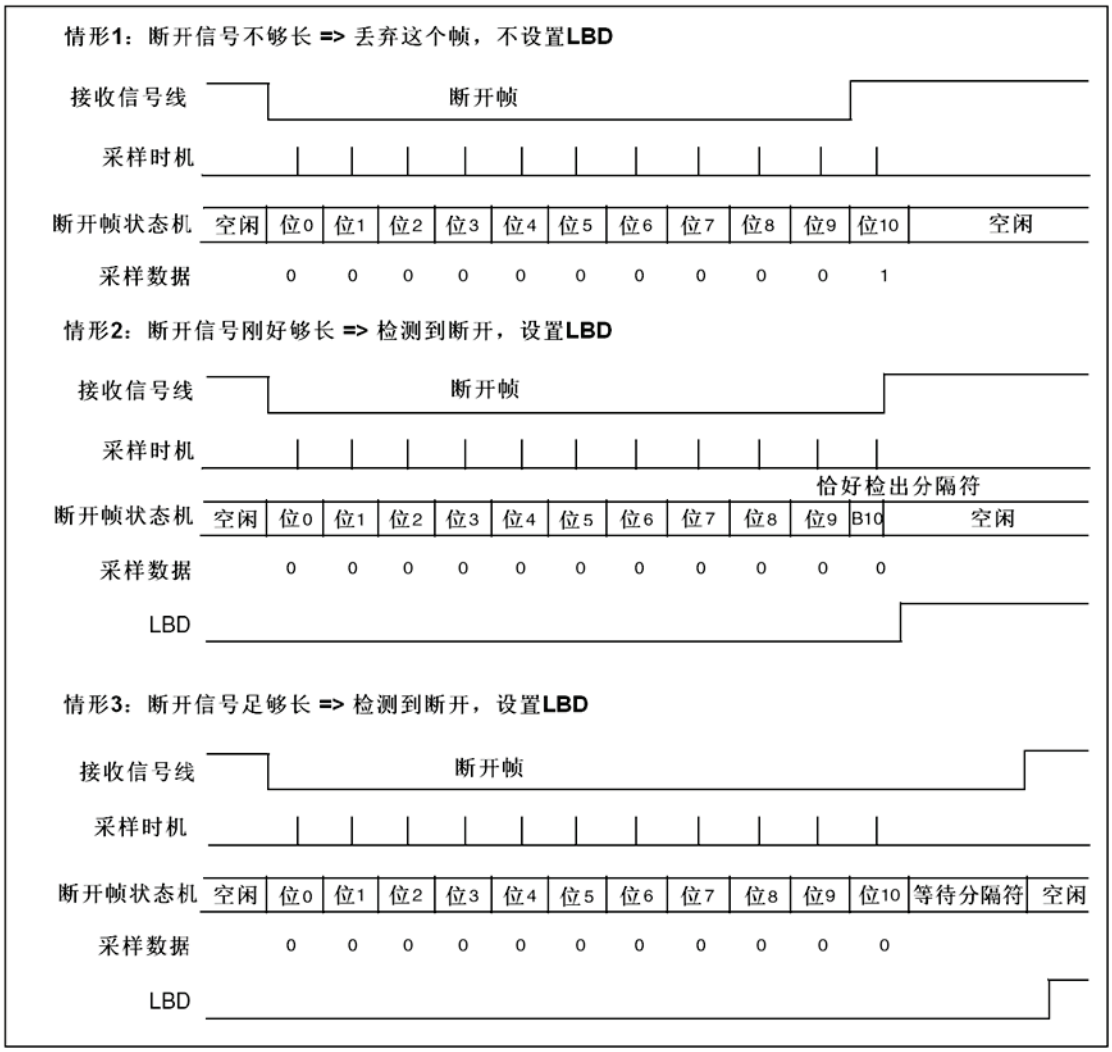


图 257 LIN 模式下的断开检测(11 位断开长度-设置了 LBDL 位)

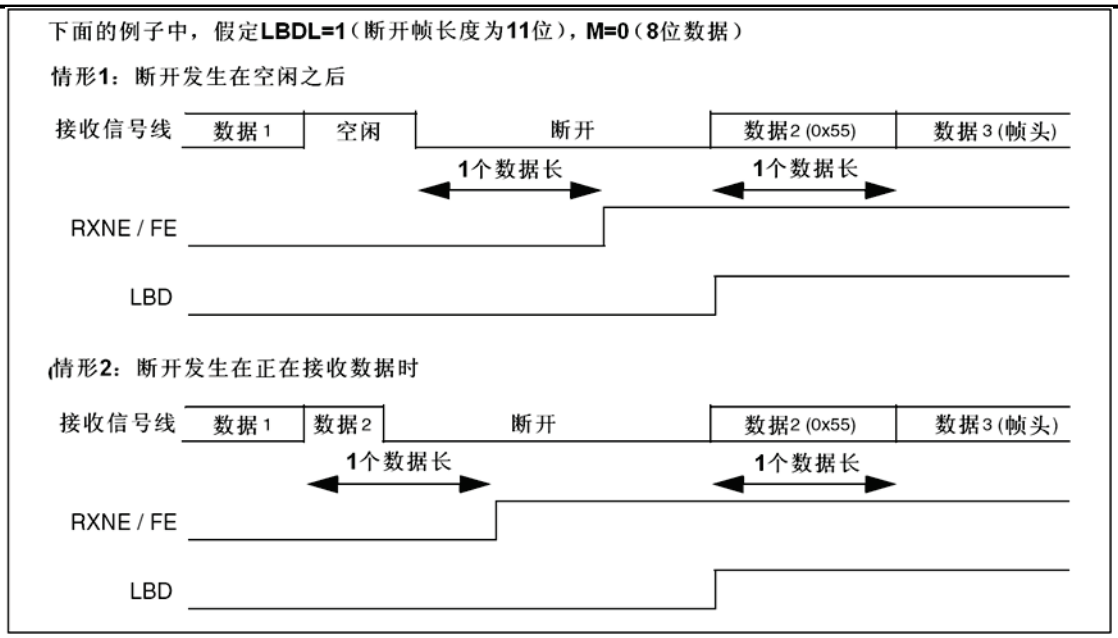


图 258 LIN 模式下的断开检测与帧错误的检测

25.3.9 USART 同步模式

通过在 USART_CR2 寄存器上写 CLKEN 位选择同步模式

在同步模式里，下列位必须保持清零状态：

- USART_CR2 寄存器中的 LINEN 位
- USART_CR3 寄存器中的 SCEN,HDSEL 和 IREN 位

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USART_CR2 寄存器中 LBCL 位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART_CR2 寄存器的 CPOL 位允许用户选择时钟极性，USART_CR2 寄存器上的 CPHA 位允许用户选择外部时钟的相位(见图 259、图 260 和图 261)。在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 CK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的(根据 CPOL 和 CPHA)，所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 CK 上采样(根据 CPOL 和 CPHA 决定在上升沿还是下降沿)，不需要任何的过采样。但必须考虑建立时间和持续时间(取决于波特率，1/16 位时间)。

注意：1．CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器(TE =1)，并且发送数据时(写入数据至 USART_DR 寄存器)才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。

2．LBCL,CPOL 和 CPHA 位的正确配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变

3．建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。

4．USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据(CK 永远是输出)。

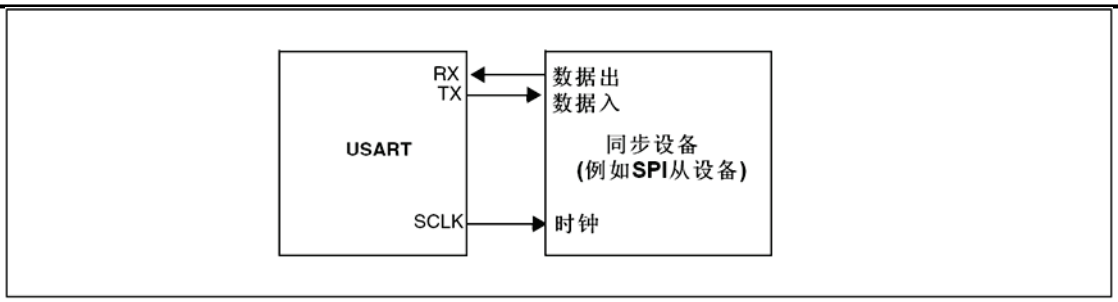


图 259 USART 同步传输的例子

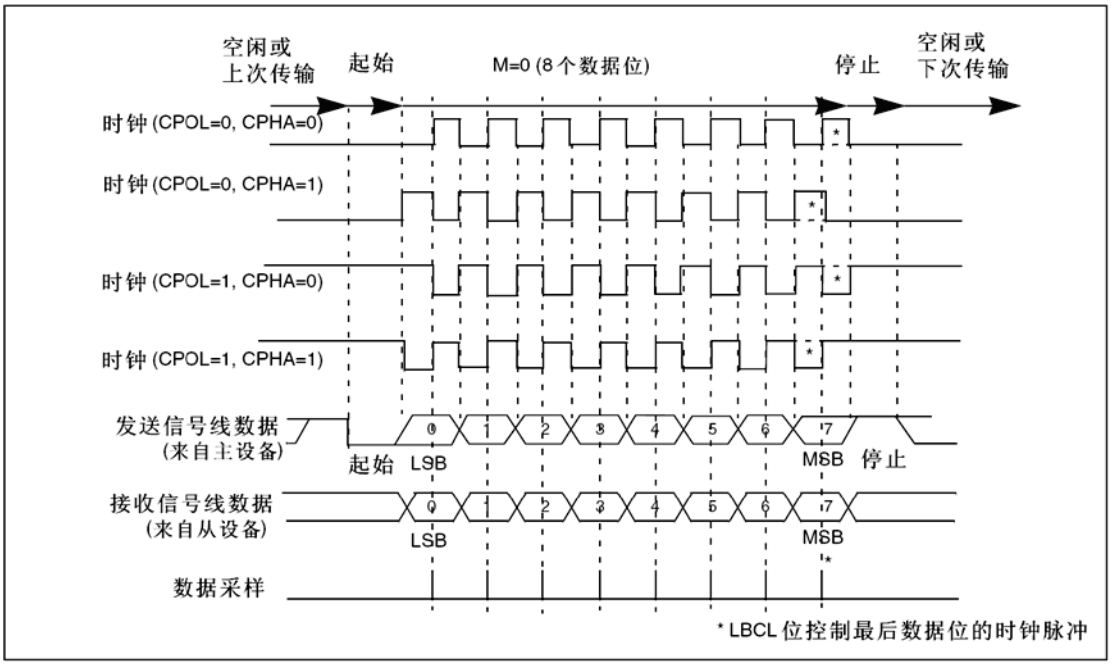


图 260 USART 数据时钟时序示例(M=0)

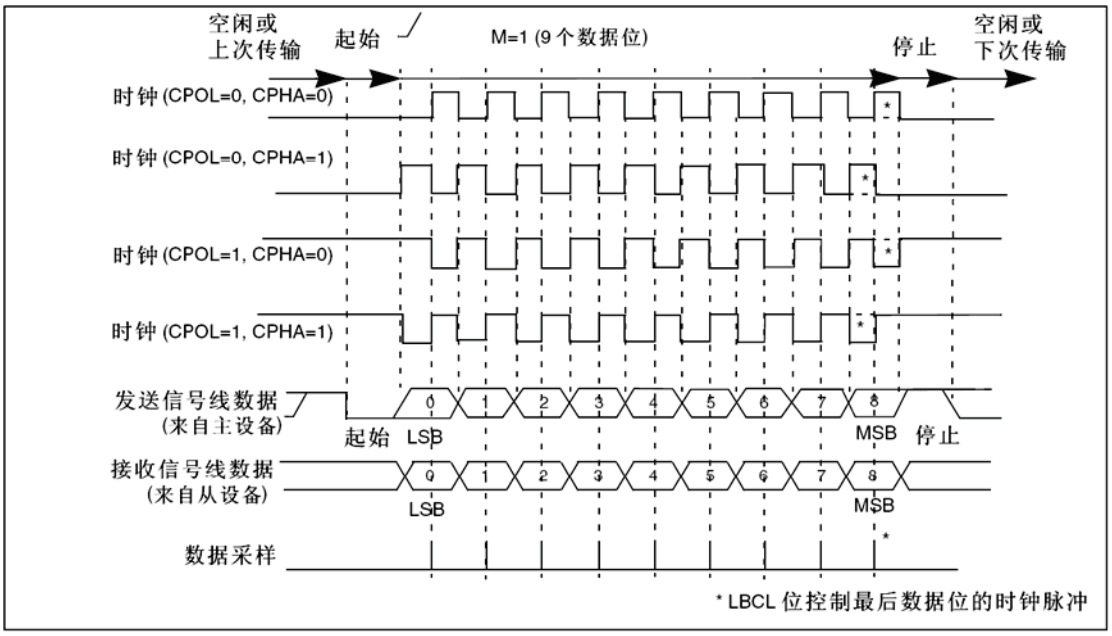


图 261 USART 数据时钟时序示例(M=1)

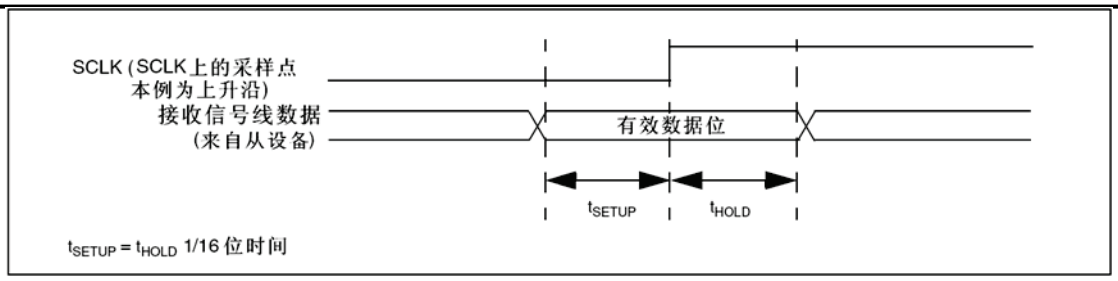


图 262 RX 数据采样/保持时间

注：　在智能卡模式下 CK 的功能不同，有关细节请参考智能卡模式部分。

25.3.10 单线半双工通信

单线半双方模式通过设置 USART_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART_CR2 寄存器的 LINEN 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位"HALF DUPLEX SEL"(USART_CR3 中的 HDSEL 位)选择半双工和全双工通信。当 HDSEL 为'1'时

- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入(或开漏的输出高)。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

25.3.11 智能卡

设置 USART_CR3 寄存器的 SCEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN 位
- USART_CR3 寄存器的 HDSEL 位和 IREN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

该接口符合 ISO7816-3 标准，支持智能卡异步协议。USART 应该被设置为：

- 8 位数据位加校验位：此时 USART_CR1 寄存器中 M=1、PCE=1
- 发送和接收时为 1.5 个停止位：即 USART_CR2 寄存器的 STOP=11

注：　也可以在接收时选择 0.5 个停止位，但为了避免在 2 种配置间转换，建议在发送和接收时使用 1.5 个停止位。

下图给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

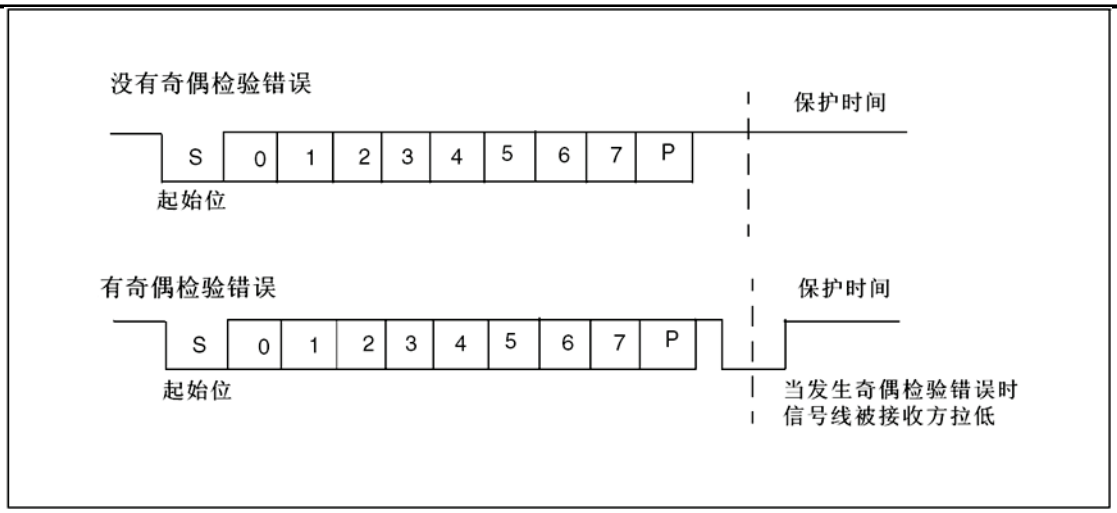


图 263 ISO7816-3 异步协议

当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线。为了做到这点，SW_RX 必须和 TX 连接到相同的 I/O 口。在发送开始位和数据字节期间，发送器的输出使能位 TX_EN 被置起，在发送停止位期间被释放(弱上拉)，因此在发现校验错误的情况下接收器可以将数据线拉低。如果 TX_EN 不被使用，在停止位期间 TX 被拉到高电平：这样的话，只要 TX 配置成开漏，接收器也可以驱动这根线。

智能卡是一个单线半双工通信协议

- 从发送移位寄存器把数据发送出去，要被延时最小 1/2 波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟 1/2 波特时钟。
- 如果在接收一个设置为 0.5 或 1.5 个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后(即停止位结束时)，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确地接收到。此 NACK 信号(拉低发送线一个波特时钟周期)在发送端将产生一个帧错误(发送端被配置成 1.5 个停止位)。应用程序可以根据协议处理重新发送数据。如果设置了 NACK 控制位，发生校验错误时接收器会给出一个 NACK 信号；否则就不会发送 NACK。
- TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC 被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC 在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。
- TC 标志的撤销不受智能卡模式的影响。
- 如果发送器检测到一个帧错误(收到接收器的 NACK 信号)，发送器的接收功能模块不会把 NACK 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是 1 或 2 波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意：　1.断开符号在智能卡模式里没有意义。一个带帧错误的 00h 数据将被当成数据而不是断开符号。

2.当来回切换 TE 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

下图详述了 USART 是如何采样 NACK 信号的。在这个例子里，USART 正在发送数据，并且被配置成 1.5 个停止位。为了检查数据的完整性和 NACK 信号，USART 的接收功能块被激活。

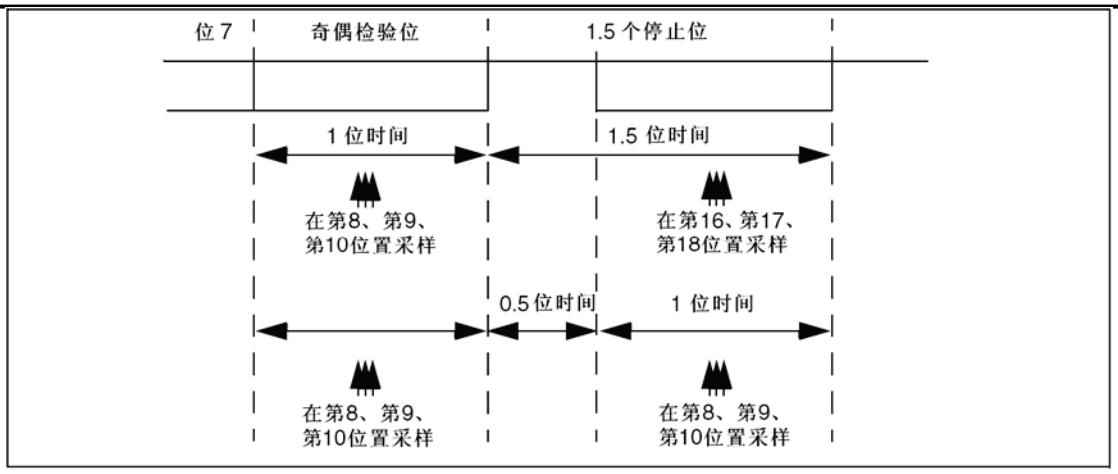


图 264 使用 1.5 停止位检测奇偶检验错

USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式里，CK 不和通信直接关联，而是先通过一个 5 位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 USART_GTPR 中配置。CK 频率可以从 fCK/2 到 fCK/62，这里的 fCK 是外设输入时钟。

25.3.12 IrDA SIR ENDEC 功能模块

通过设置 USART_CR3 寄存器的 IREN 位选择 IrDA 模式。在 IRDA 模式里，下列位必须保持清零：

- USART_CR2 寄存器的 LINEN,STOP 和 CLKEN 位
- USART_CR3 寄存器的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零调制方案(RZI)，该方案用一个红外光脉冲代表逻辑'0'(见图 265)。SIR 发送编码器对从 USART 输出的 NRZ(非归零)比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外 LED。USART 为 SIR ENDEC 最高只支持到 115.2Kbps 速率。在正常模式里，脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高(标记状态 markingstate)。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙(也就是 USART 正在送数据给 IrDA 编码器)

IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙(也就是 USART 正在接收从 IrDA 解码器来的解码数据)，从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。

- SIR 发送逻辑把'0'作为高脉冲发送，把'1'作为低电平发送。脉冲的宽度规定为正常模式时位周期的 3/16(见图 266)。
- SIR 接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于 1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于 2 个 PSC 周期的脉冲(PSC 是在 IrDA 低功耗波特率寄存器 USART_GTPR 中编程的预分频值)。宽度小于 1 个 PSC 周期的脉冲一定被滤除掉，但是那些宽度大于 1 个而小于 2 个 PSC 周期的脉冲可能被接收或滤除，那些宽度大于 2 个周期的将被视为一个有效的脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。

- 在 IrDA 模式里，USART_CR2 寄存器上的 STOP 位必须配置成 1 个停止位。

IrDA 低功耗模式

发送器

在低功耗模式，脉冲宽度不再持续 3/16 个位周期。取而代之，脉冲的宽度是低功耗波特率的 3 倍，它最小可以是 1.42MHz。通常这个值是 1.8432MHz(1.42MHz<PSC<2.12MHz)。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

接收器

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于 1 个 PSC 的脉冲。只有持续时间大于 2 个周期的 IrDA 低功耗波特率时钟(USART_GTPR 中的 PSC)的低电平信号才被接受为有效的信号。

注意： *1.宽度小于 2 个大于 1 个 PSC 周期的脉冲可能会也可能不会被滤除。*

2.接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有 10ms 的延时(IrDA 是一个半双工协议)。

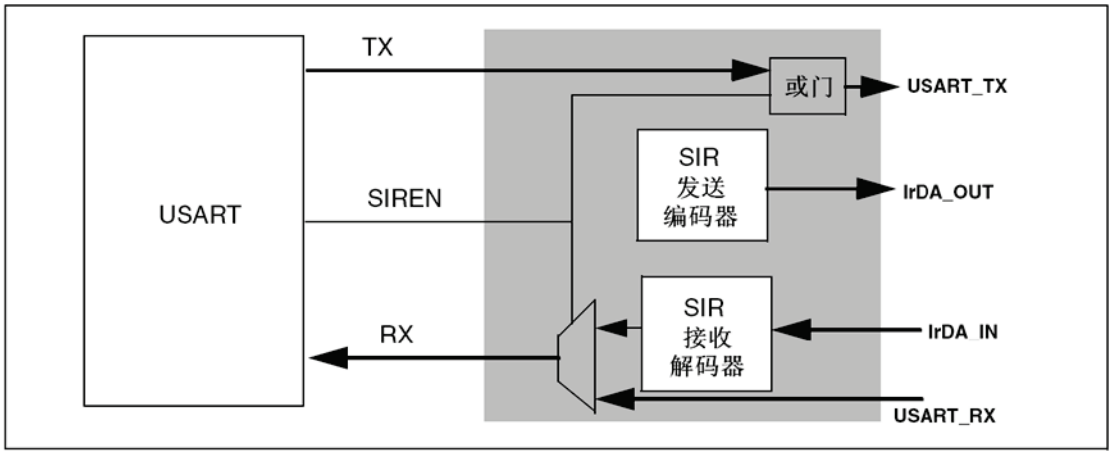


图 265 IrDA SIR ENDEC 框图

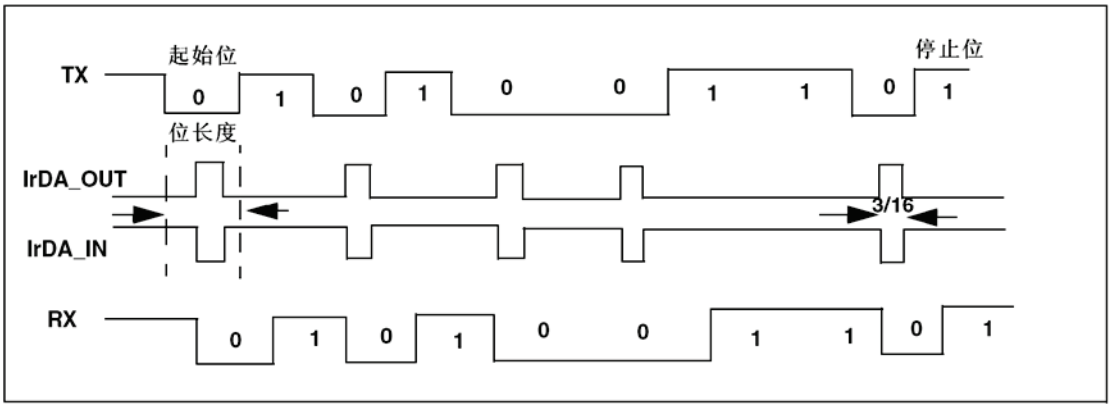


图 266 IrDA 数据调制(3/16)普通模式

25.3.13 利用 DMA 连续通信

USART 可以利用 DMA 连续通信。Rx 缓冲器和 Tx 缓冲器的 DMA 请求是分别产生的。

利用 DMA 发送

使用 DMA 进行发送，可以通过设置 USART_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为'1'时，DMA 就从指定的 SRAM 区传送数据到 USART_DR 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下(x 表示通道号)：

1. 在 DMA 控制寄存器上将 USART_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 USART_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. 在 DMA 寄存器上激活该通道。

当传输完成 DMA 控制器指定的数据量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。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 USART_SR 寄存器的 TC 标志可以确认 USART 通信是否结束，这样可以在关闭 USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等待 TC=1。

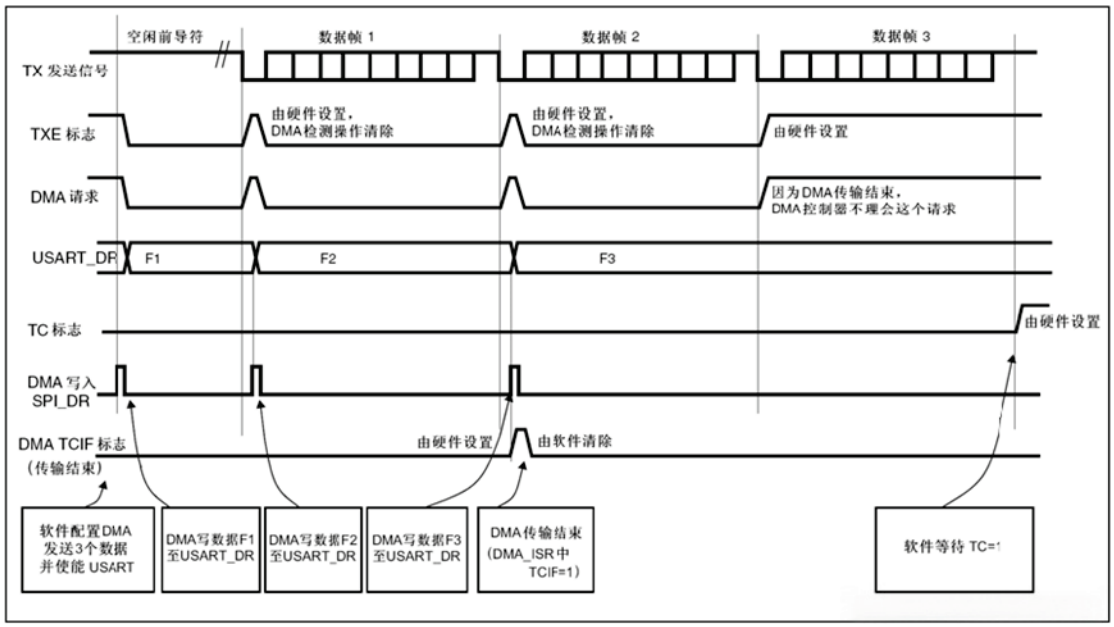


图 267 利用 DMA 发送

利用 DMA 接收

可以通过设置 USART_CR3 寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就就把数据从 USART_DR 寄存器传送到指定的 SRAM 区(参考 DMA 相关说明)。为 USART 的接收分配一个 DMA 通道的步骤如下(x 表示通道号)：

1. 通过 DMA 控制寄存器把 USART_DR 寄存器地址配置成传输的源地址。在每个 RXNE 事件后，将从此地址读出数据并传输到存储器。

2. 通过 DMA 控制寄存器把存储器地址配置成传输的目的地址。在每个 RXNE 事件后，数据将从 USART_DR 传输到此存储器区。
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。

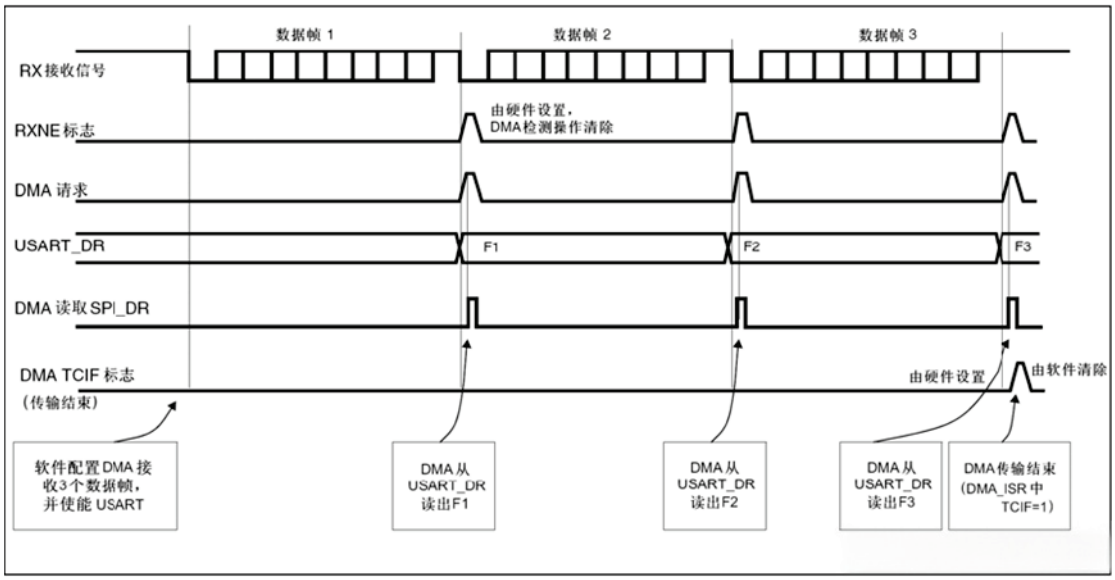


图 268 利用 DMA 接收

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。

25.3.14 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。

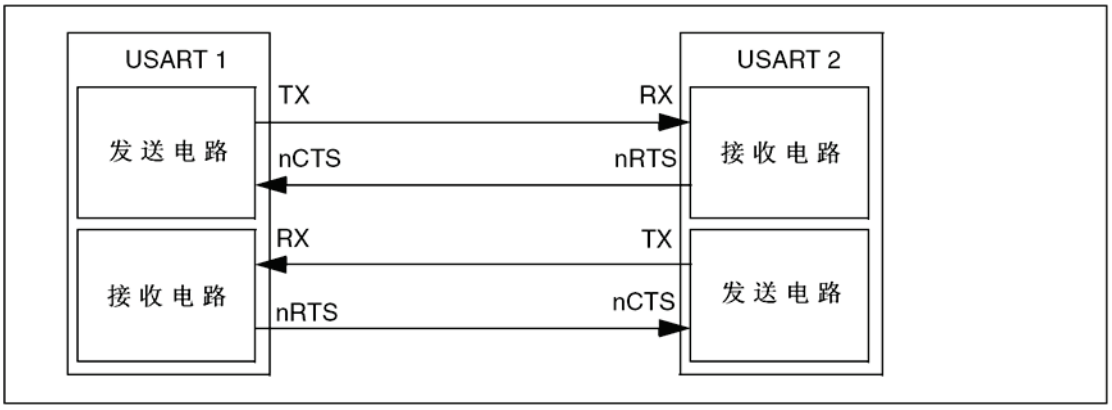


图 269 两个 USART 间的硬件流控制

通过将 UASRT_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

RTS 流控制

如果 RTS 流控制被使能(RTSE=1)，只要 USART 接收器准备好接收新的数据，nRTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

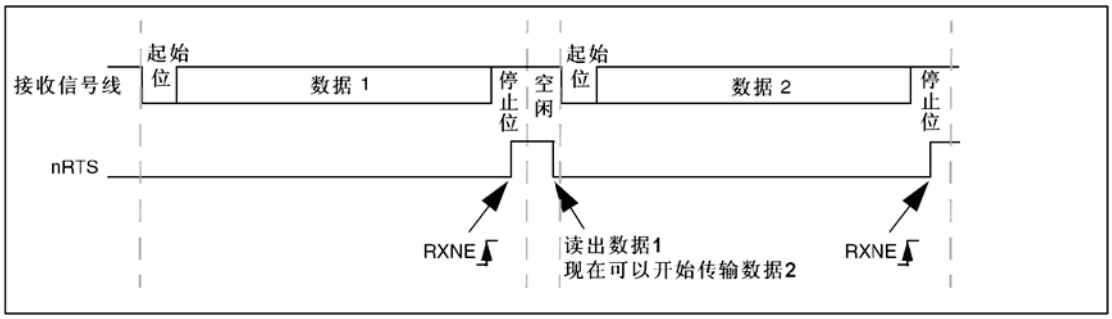


图 270 RTS 流控制

CTS 流控制

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 nCTS 输入一变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART_CT3 寄存器的 CTSIE 位，则产生中断。下图是一个启用 CTS 流控制通信的例子。

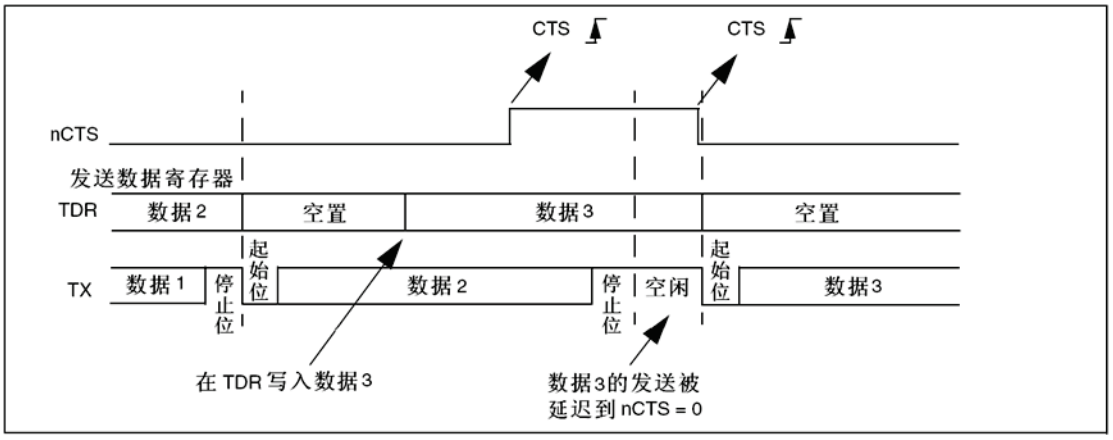


图 271 CTS 流控制

25.4 USART 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	TXNE	TXNEIE
检测到数据溢出	ORE	
检测到空闲线路	IDLE	IDLEIE
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。

USART 的各种中断事件被连接到同一个中断向量(见下图)，有以下各种中断事件：

- 发送期间：发送完成、清除发送、发送数据寄存器空。
- 接收期间：空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN 断开符号检测、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。

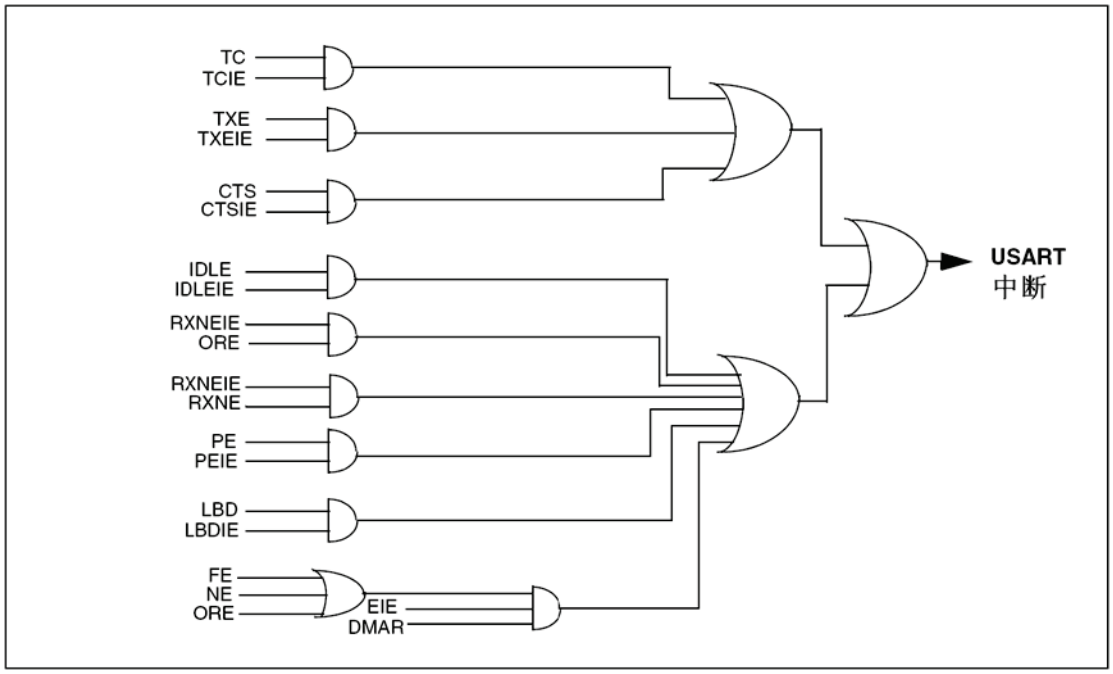


图 272 USART 中断映像图

25.5 USART 模式配置

表 146 USART 模式设置(1)

USART 模式	USART1	USART2	USART3	UART4	UART5
异步模式	◎	◎	◎	◎	◎
硬件流控制	◎	◎	◎	●	●
多缓存通讯(DMA)	◎	◎	◎	◎	●
多处理器通讯	◎	◎	◎	◎	◎
同步	◎	◎	◎	●	●
智能卡	◎	◎	◎	●	●
半双工(单线模式)	◎	◎	◎	◎	◎
IrDA	◎	◎	◎	◎	◎
LIN	◎	◎	◎	◎	◎

(1)◎=支持，●=不支持该应用

25.6 USART 寄存器描述

有关寄存器描述里所使用的缩写，请参考第 1 节。

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

25.6.1 状态寄存器(USART_SR)

地址偏移：0x00

复位值：0x00C0

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

位	符号	说明
31:10	Reserved	保留，始终读为 0。
9	CTS	**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 USART_CR3 中的 CTSIE 为'1'，则产生中断。 0：nCTS 状态线上没有变化； 1：nCTS 状态线上发生变化。 注：UART4 和 UART5 上不存在这一位。
8	LBD	**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 USART_CR3 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。 注意：若 LBDIE=1，当 LBD 为'1'时要产生中断。
7	TXE	**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 USART_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 USART_DR 的写操作，将该位清零。 0：数据还没有被转移到移位寄存器； 1：数据已经被转移到移位寄存器。 注意：单缓冲器传输中使用该位。
6	TC	**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 USART_CR1 中的 TCIE 为'1'，则产生中断。由软件序列清除该位(先读 USART_SR，然后写入 USART_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。
5	RXNE	**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 USART_DR 寄存器中，该位被硬件置位。如果 USART_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 USART_DR 的读操作可以将该位清零。RXNE 位也可以通过写入 0 来清除，只有在多缓存通讯中才推荐这种清除程序。 0：数据没有收到； 1：收到数据，可以读出。
4	IDLE	**IDLE**：监测到总线空闲(IDLE line detected) 当检测到总线空闲时，该位被硬件置位。如果 USART_CR1 中的 IDLEIE 为'1'，则产生中断。由软件序列清除该位(先读 USART_SR，然后读 USART_DR)。 0：没有检测到空闲总线； 1：检测到空闲总线。 注意：IDLE 位不会再次被置高直到 RXNE 位被置起(即又检测到一次空闲总线)
3	ORE	**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 USART_CR1 中的 RXNEIE 为'1'的话，则产生中断。由软件序列将其清零(先读 USART_SR，然后读 USART_CR)。 0：没有过载错误； 1：检测到过载错误。 注意：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 EIE 位，在多缓冲器通信模式下，ORE 标志置位会产生中断的。
2	**NE**	**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。由软件序列对其清玲(先读 USART_SR，再读 USART_DR)。 0：没有检测到噪声； 1：检测到噪声。 注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。在多缓冲区通信模式下，如果设置了 EIE 位，则设置 NE 标志时会产生中断。
1	**FE**	**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 USART_SR，再读 USART_DR)。 0：没有检测到帧错误； 1：检测到帧错误或者 break 符。 注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。 如果当前传输的数据既产生了帧错误，又产生了过载错误，硬件还是会继续该数据的传输，并且只设置 ORE 标志位。 在多缓冲区通信模式下，如果设置了 EIE 位，则设置 FE 标志时会产生中断。
0	**PE**	**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 USART_SR 和 USART_DR)。在清除 PE 位前，软件必须等待 RXNE 标志位被置'1'。如果 USART_CR1 中的 PEIE 为'1'，则产生中断。 0：没有奇偶校验错误； 1：奇偶校验错误。

25.6.2 数据寄存器(USART_DR)

地址偏移：0x04

复位值：不确定

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

位	符号	说明
31:9	**Reserved**	保留，始终读为 0。
8:0	**DR[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(参见图 249)。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位(USART_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。

25.6.3 波特比率寄存器(USART_BRR)

注意：　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

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

位	符号	说明
31:16	Reserved	保留，始终读为 0。
15:4	DIV_Mantissa [11:0]	**DIV_Mantissa[11:0]**：USARTDIV 的整数部分 这 12 位定义了 USART 分频器除法因子(USARTDIV)的整数部分。
3:0	DIV_Fraction [3:0]	**DIV_Fraction[3:0]**：USARTDIV 的小数部分 这 4 位定义了 USART 分频器除法因子(USARTDIV)的小数部分。

25.6.4 控制寄存器 1(USART_CR1)

地址偏移：0x0C

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:14	Reserved	保留，始终读为 0。
13	UE	**UE**：USART 使能(USART enable) 当该位被清零，在当前字节传输完成后 USART 的分频器和输出停止工作，以减少功耗。 该位由软件设置和清零。 0：USART 分频器和输出被禁止； 1：USART 模块使能。
12	M	**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。 注意：在数据传输过程中(发送或者接收时)，不能修改这个位。
11	WAKE	**WAKE**：唤醒的方法(Wakeup method) 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 0：被空闲总线唤醒； 1：被地址标记唤醒。
10	PCE	**PCE**：检验控制使能(Parity control enable) 用该位选择是否进行硬件校验控制(对于发送来说就是校验位的产生；对于接收来说就是校验位的检测)。当使能了该位，在发送数据的最高位(如果 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。
9	PS	**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。
8	PEIE	**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 USART_SR 中的 PE 为'1'时，产生 USART 中断。
7	TXEIE	**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 USART_SR 中的 TXE 为'1'时，产生 USART 中断。
6	TCIE	**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 USART_SR 中的 TC 为'1'时，产生 USART 中断。
5	RXNEIE	**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 USART_SR 中的 ORE 或者 RXNE 为'1'时，产生 USART 中断。
4	IDLEIE	**IDLEIE**：IDLE 中断使能(IDLE interrupt enable)该位由软件设置或清除。 0：禁止产生中断； 1：当 USART_SR 中的 IDLE 为'1'时，产生 USART 中断。
3	TE	**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。 注意：1．在数据传输过程中，除了在智能卡模式下，如果 TE 位上有个 0 脉冲(即设置为'0'之后再设置为'1')，会在当前数据字传输完成后，发送一个“前导符”(空闲总线)。 2．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。
2	RE	**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。
1	RWU	**RWU**：接收唤醒(Receiver wakeup) 该位用来决定是否把 USART 置于静默模式。该位由软件设置或清除。当唤醒序列到来时，硬件也会将其清零。 0：接收器处于正常工作模式； 1：接收器处于静默模式。 注意：1．在把 USART 置于静默模式(设置 RWU 位)之前，USART 要已经先接收了一个数据字节。否则在静默模式下，不能被空闲总线检测唤醒。 2．当配置成地址标记检测唤醒(WAKE 位=1)，在 RXNE 位被置位时，不能用软件修改 RWU 位。
0	SBK	**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。

25.6.5 控制寄存器 2(USART_CR2)

地址偏移：0x10

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

位	符号	说明
31:15	Reserved	保留，始终读为 0。
14	LINEN	**LINEN**：LIN 模式使能(LIN mode enable)该位由软件设置或清除。 0：禁止 LIN 模式； 1：使能 LIN 模式。 在 LIN 模式下，可以用 USART_CR1 寄存器中的 SBK 位发送 LIN 同步断开符(低 13 位)，以及检测 LIN 同步断开符。
13:12	STOP	**STOP**：停止位(STOP bits)这 2 位用来设置停止位的位数 00：1 个停止位； 01：0.5 个停止位； 10：2 个停止位； 11：1.5 个停止位； 注：UART4 和 UART5 不能用 0.5 停止位和 1.5 停止位。
11	CLKEN	**CLKEN**：时钟使能(Clock enable)该位用来使能 CK 引脚 0：禁止 CK 引脚； 1：使能 CK 引脚。 注：UART4 和 UART5 上不存在这一位。
10	CPOL	**CPOL**：时钟极性(Clock polarity) 在同步模式下，可以用该位选择 SLCK 引脚上时钟输出的极性。和 CPHA 位一起配合来产生需要的时钟/数据的采样关系 0：总线空闲时 CK 引脚上保持低电平； 1：总线空闲时 CK 引脚上保持高电平。 注：UART4 和 UART5 上不存在这一位。
9	CPHA	**CPHA**：时钟相位(Clock phase) 在同步模式下，可以用该位选择 SLCK 引脚上时钟输出的相位。和 CPOL 位一起配合来产生需要的时钟/数据的采样关系(参见图 260 和图 261)。 0：在时钟的第一个边沿进行数据捕获； 1：在时钟的第二个边沿进行数据捕获。 注：UART4 和 UART5 上不存在这一位。
8	LBCL	**LBCL**：最后一位时钟脉冲(Last bi tclock pulse) 在同步模式下，使用该位来控制是否在 CK 引脚上输出最后发送的那个数据字节(MSB)对应的时钟脉冲 0：最后一位数据的时钟脉冲不从 CK 输出； 1：最后一位数据的时钟脉冲会从 CK 输出。 注意：1．最后一个数据位就是第 8 或者第 9 个发送的位(根据 USART_CR1 寄存器中的 M 位所定义的 8 或者 9 位数据帧格式)。 2．UART4 和 UART5 上不存在这一位。
7	Reserved	保留位，硬件强制为 0
6	LBDIE	**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 USART_SR 寄存器中的 LBD 为'1'就产生中断。
5	**LBDL**	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。
4	**Reserved**	保留位，硬件强制为 0
3:0	**ADD[3:0]**	ADD[3:0]：本设备的 USART 节点地址该位域给出本设备 USART 节点的地址。 这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个 USART 设备。

注意：　在使能发送后不能改写这三个位(CPOL、CPHA、LBCL)。

25.6.6　控制寄存器 3(USART_CR3)

地址偏移：0x14

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31:11	**Reserved**	保留，始终读为 0。
10	**CTSIE**	**CTSIE**：CTS 中断使能(CTS interrupt enable) 0：禁止中断； 1：USART_SR 寄存器中的 CTS 为'1'时产生中断。 注：UART4 和 UART5 上不存在这一位。
9	**CTSE**	**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。 注：UART4 和 UART5 上不存在这一位。
8	**RTSE**	**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)。 注：UART4 和 UART5 上不存在这一位。
7	**DMAT**	**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； 注：UART4 和 UART5 上不存在这一位。
6	**DMAR**	**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； 注：UART4 和 UART5 上不存在这一位。
5	**SCEN**	**SCEN**:智能卡模式使能(Smartcard mode enable)该位用来使能智能卡模式 0：禁止智能卡模式； 1：使能智能卡模式。 注：UART4 和 UART5 上不存在这一位。

4	NACK	**NACK**：智能卡 NACK 使能(Smartcard NACK enable) 0：校验错误出现时，不发送 NACK； 1：校验错误出现时，发送 NACK。 注：UART4 和 UART5 上不存在这一位。
3	HDSEL	**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。
2	IRLP	**IRLP**：红外低功耗(IrDA low-power) 该位用来选择普通模式还是低功耗红外模式 0：通常模式； 1：低功耗模式。
1	IREN	**IREN**：红外模式使能(IrDA mode enable)该位由软件设置或清除。 0：不使能红外模式； 1：使能红外模式。
0	EIE	**EIE**：错误中断使能(Error interrupt enable) 在多缓冲区通信模式下，当有帧错误、过载或者噪声错误时(USART_SR 中的 FE=1，或者 ORE=1，或者 NE=1)产生中断。 0：禁止中断； 1：只要 USART_CR3 中的 DMAR=1，并且 USART_SR 中的 FE=1，或者 ORE=1，或者 NE=1，则产生中断

25.6.7 保护时间和预分频寄存器(USART_GTPR)

地址偏移：0x18

复位值：0x0000

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

位	符号	说明
31:11	Reserved	保留，始终读为 0。
15:8	GT[7:0]	GT[7:0]：保护时间值(Guard time value) 该位域规定了以波特时钟为单位的保护时间。在智能卡模式下，需要这个功能。当保护时间过去后，才会设置发送完成标志。 注：UART4 和 UART5 上不存在这一位。
7:0	PSC[7:0]	PSC[7:0]：预分频器值(Prescaler value) -在红外(IrDA)低功耗模式下：PSC[7:0]=红外低功耗波特率 对系统时钟分频以获得低功耗模式下的频率：源时钟被寄存器中的值(仅有 8 位有效)分频 00000000：保留–不要写入该值； 00000001：对源时钟 1 分频； 00000010：对源时钟 2 分频； …… -在红外(IrDA)的正常模式下：PSC 只能设置为 00000001 -在智能卡模式下： PSC[4:0]：预分频值 对系统时钟进行分频，给智能卡提供时钟。 寄存器中给出的值(低 5 位有效)乘以 2 后，作为对源时钟的分频因子 00000：保留–不要写入该值； 00001：对源时钟进行 2 分频； 00010：对源时钟进行 4 分频；

		00011：对源时钟进行 6 分频； …… 注意：1．位[7:5]在智能卡模式下没有意义。 2．UART4 和 UART5 上不存在这一位。

25.6.8 USART 寄存器地址映象

表 147 USART 寄存器列表及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	USART_SR	保留																						CTS	LBD	TXEIE	TC	RXNE	IDLE	ORE	NE	FE	PE
	复位值																							0	0	1	1	0	0	0	0	0	0
004h	USART_DR	保留																								DR[8:0]							
	复位值																									0	0	0	0	0	0	0	0
008h	USART_BRR	保留																DIV_Mantissa[15:4]												DIV_ Fraction [3:0]			
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	USART_CR1	保留																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	复位值																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	USART_CR2	保留																	LIEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]			
	复位值																		0	0	0	0	0	0	0		0	0		0	0	0	0
014h	USART_CR3	保留																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	复位值																						0	0	0	0	0	0	0	0	0	0	0
018h	USART_GTPR	保留																GT[7:0]								PSC[7:0]							
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

关于寄存器的起始地址，参见表 1。

26 USBOTG 全速(OTG_FS)

26.1 OTG 模块介绍

部分版权属于 Synopsys 公司(2004，2005)。保留所有权利。已被允许使用。本章描述了 OTG_FS 控制器的架构和如何编程使用。

本章所使用的术语：

FS： 全速

LS： 低速

USB： 通用串行总线

OTG： On-the-Go

PHY： 物理层

USB: 通用串行总线

UTMI: USB2.0 收发器宏单元接口(UTMI)

参考下列文档：

- USBOn-The-Go Supplement,Revision 1.3
- Universal Serial Bus Revision 2.0 Specification

OTG_FS 是双重角色设备(DRD)控制器，支持主机端和设备端的功能，完全遵从 On-The-Go Supplement to the USB 2.0 规范。同时，该控制器也可配置为仅支持主机端或仅支持设备端功能的控制器，遵从 USB2.0 规范。在主机模式下，OTG_FS 支持全速(FS,12Mbits/s)和低速(LS,1.5Mbits/s)通信，而在设备模式下，支持全速(FS,12Mbits/s)通信。OTG_FS 控制器支持 HNP 和 SRP 协议。外围仅在主机模式下需要配置一个针对 VBUS 的电荷泵，即可完成设计。

26.2 OTG_FS 主要功能

以下将从通用，主机模式和设备模式三方面来介绍 OTG_FS 控制器的特性。

26.2.1 通用功能

OTG_FS 控制器接口：

- 由 USB-IF 认证，符合 Universal Serial Bus Specification，Revision2.0 标准
- 完全支持在(OTG_FS 控制器的物理层(PHY))USBOn-The-Go Supplement，Revision1.3 规范中定义为可选项目 OTG 协议。
  - 对插入的 A - B 类设备的辨认(ID 线)
  - 支持主机协商协议(HNP)和会话请求协议(SRP)
  - 在 OTG 应用中，允许主机关闭 VBUS 以节省耗电
  - OTG 控制器使用内部比较器监视 VBUS 电平
  - 可以动态的切换主机/设备角色
- 可以通过软件配置，完成以下设计：
  - 支持 SRP 协议的 USB 全速设备(B 类设备)
  - 支持 SRP 协议的 USB 全速/低速主机(A 类设备)
  - USBOTG 全速双重角色设备
- 支持全速通信的 SOF 信号和低速通信的保持有效信号

- SOF 的脉冲可以输出到引脚
- SOF 在内部连接到定时器 2(TIM2)
- 可配置的帧周期
- 可配置的帧结束中断

- 提供省电功能：如在 USB 挂起时停止系统，关闭数字部分，PHY 和 DFIFO 电源管理部分的内部时钟系统。
- 提供 1.25K 字节的专用 RAM 和高级的 FIFO 管理
  - 通过软件为不同的 FIFO 配置不同的 RAM 区域，以便灵活有效的使用 RAM
  - 每个 FIFO 可以存储多个数据包
  - 允许动态的分配存储区
  - 不限定 FIFO 的长度一定是 2 的幂次，以便可以连续的使用存储区
- 不需要系统的介入就可以保证一个帧(1ms)的最大数据流量。

26.2.2 主机模式功能

OTG_FS 控制器接口：

- 需要一个外置的电荷泵为 VBUS 供电
- 支持最多 8 个主机通道，每个通道都可以动态的配置为任意一种传输类型
- 内置硬件调度控制器：
  - 在周期性硬件传输请求队列中支持多达 8 个中断和同步传输请求
  - 在非周期性硬件传输请求队列中支持多达 8 个控制和大容量传输的传输请求。
- 为有效地使用 RAM 空间，USB 的数据 RAM 区划分为一个共享的接收 FIFO、一个周期性发送 FIFO 和一个非周期性发送 FIFO。

设备模式功能

OTG_FS 控制器接口：

- 提供 1 个双向的控制端点 0
- 提供 3 个 IN 端点，支持大容量、中断或同步传输
- 提供 3 个 OUT 端点，支持大容量、中断或同步传输
- 为有效地使用 USB 的数据 RAM 区，管理一个共享的接收 FIFO，和一个发送 OUTFIFO
- 管理多达 4 个专用的发送 INFIFO(为每个 IN 端点配置一个 FIFO)，以便减少应用程序的负荷
- 支持软件的断开连接功能。

26.3 OTG_FS 功能描述

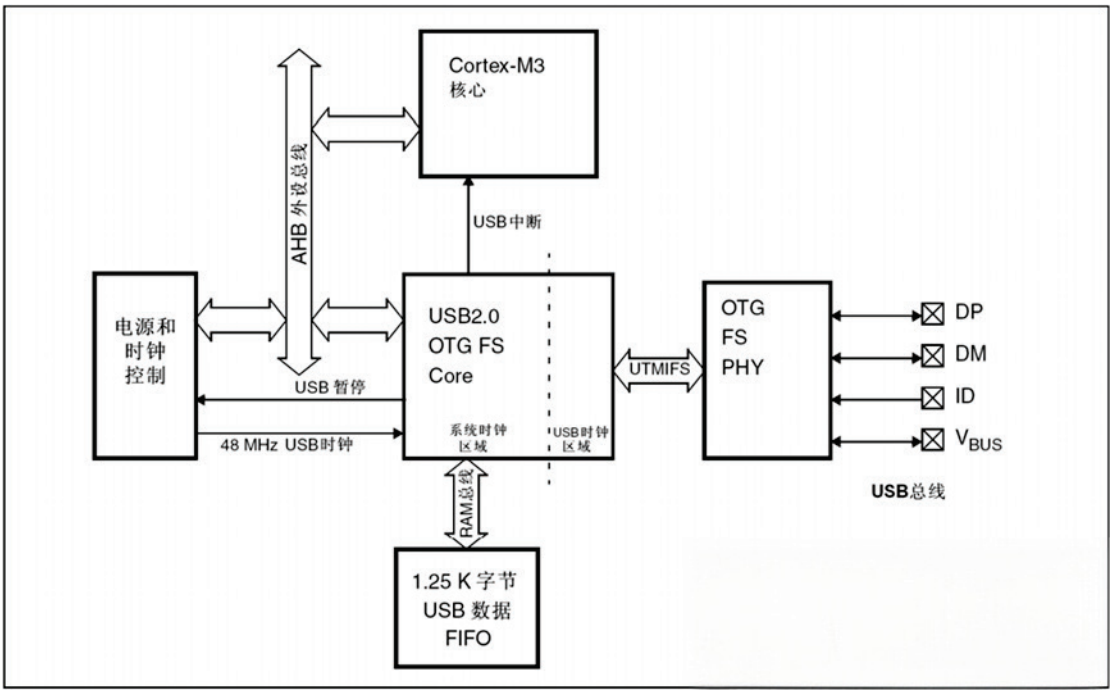


图 273 框图

26.3.1 OTG 引脚

表 148 OTG 引脚

信号名称	信号类型	描述
OTG_FS_DP	数字输入/输出	USB OTG D+ 线
OTG_FS_DM	数字输入/输出	USB OTG D-线
OTG_FS_ID	数字输入	USB OTG ID
OTG_FS_VBUS	模拟输入	USB OTG VBUS
OTG_FS_SOF	数字输出	USB OTG 帧起始 (可见性)

26.3.2 OTG 全速控制器

USBOTG 全速控制器从复位和时钟控制模块(RCC)获得由外部晶振产生的 48MHz±0.25%的时钟，这个时钟用于驱动 USB 全速模块(12Mbit/s)的 48MHz 控制时钟，必须在配置 OTG 全速控制器之前使能这个控制时钟。

微控制器内核(CPU)通过 AHB 外设总线来访问 OTG 全速控制器的寄存器，USB 事件由单独的 USBOTG 中断控制线(请参考第 26.15 节：OTG_FS 中断)管理，通知微控制器内核。

微控制器以向 OTG_FS 专用地址(PUSH 寄存器)写 32 位数据的方式向 USB 控制器传输数据，这些数据将自动存入 USB 数据 RAM 中配置好的发送 FIFO 中。每个发送 FIFO 都配置了这样一个 PUSH 寄存器，为每个 IN 端点(设备模式)或 OUT 通道(主机模式)服务。

微控制器通过读 OTG_FS 的专用地址(POP 寄存器)获得来自 USB 总线的 32 位数据，这些数据将自动从共享的接收 FIFO 中载入。接收 FIFO 位于总共 1.25K 字节的 USB 数据 RAM 区。每一个 OUT 端点或 IN 通道都有这样一个 POP 寄存器为之服务。

USB 协议层由串行接口控制器(SIE)驱动，并连接到由内置物理层(PHY)支持的 USB 全速/低速收发模块。

26.3.3 全速 OTGPHY(物理接口)

内置的全速 OTGPHY 由 OTG 全速控制器控制，并通过 UTMI+总线(UTMIFS)的全速子集来收发控制和数据信号。PHY 为 USB 通信提供了物理层的支持。全速的 OTGPHY 包括以下部分：

- 在主机和设备模式下使用的全速/低速收发模块。此模块直接在单端的 USB 线上驱动发送和接收。
- 内置了 ID 线的上拉电阻，用于区分 A/B 类设备。
- DP/DM 线内置了上拉和下拉电阻，由 OTG_FS 控制器控制以满足当前设备类型的需求。在设备模式下，当 VBUS 线上出现了有效的电平(B 类有效)，控制器使能 DP 线的上拉电阻，向主机通告接入一个 USB 全速设备。在主机模式下，控制器同时使能 DP 和 DM 线的下拉电阻。上拉和下拉电阻可以在控制器通过主机协商协议(HNP)切换角色类型时动态的切换。
- 上拉/下拉电阻的 ECN 电路。DP 线有两个由 OTG_FS 控制器分开控制的上拉电阻，符合 USB2.0 版本的 ECN 要求(Enginering Change Notice)。支持动态调整 DP 线的上拉强度，能更好的对抗噪声，并提高发送和接收信号质量。
- 内置带迟滞功能的 VBUS 探测比较器，用于检测 VBUS 的有效电平、A-B 会话有效电平和会话结束电平。这些电平用于驱动会话请求协议(SRP)、检测有效的会话开始和结束条件，并在 USB 通信中持续的检测 VBUS 线状态。
- VBUS 脉冲电路用于在 SRP 期间通过电阻对 VBUS 进行充电/放电操作(弱驱动)。

26.4 OTG 双角色设备(DRD)

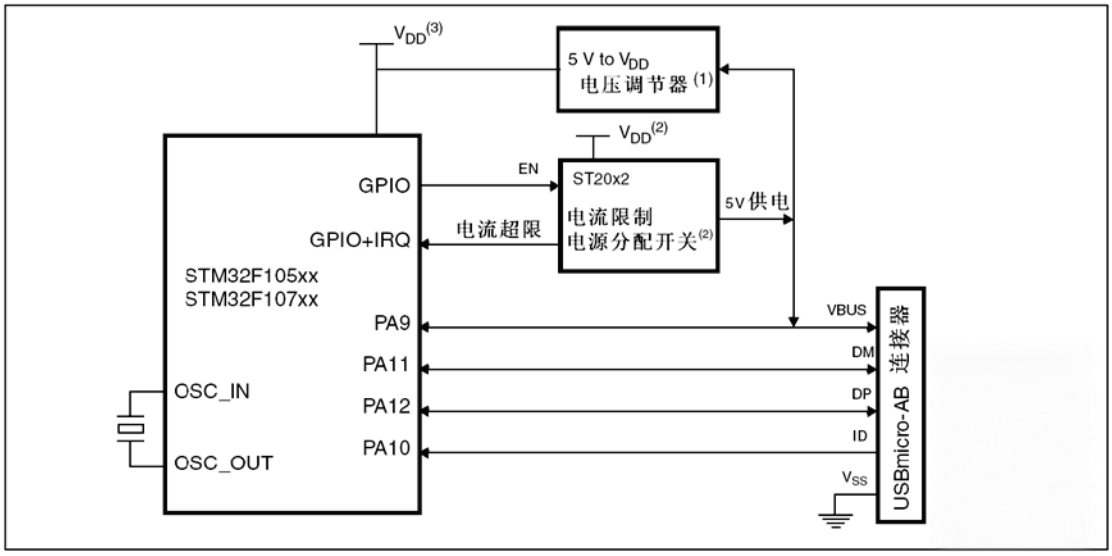


图 274 OTG 的 A-B 设备连接

1．仅在设计使用 VBUS 供电的设备时，才需要使用外部的电压调节器。

2．仅在设计使用 VBUS 供电的设备时，才需要使用 ST20x2，如果应用板上有 5V 的供电，可以使用基本的电源开关。

3．VDD 的范围从 2V 到 3.6V。

26.4.1 ID 信号检测

主机和设备(默认)的角色由 ID 线的状态定义。ID 线的状态在连入 USB 总线的一瞬间决定，并取决于插入 micro-AB 插座的 USB 电缆。

- 如果插入的是 USB 电缆的 B 端，并且 ID 线浮空，内置的上拉电阻将检测到 ID 线的高电平，此时控制器处于默认的设备模式下。在这种模式下，OTG_FS 控制器遵守 USB2.0 规范的补充 OTG1.3 规范的第 6.8.2 节：OTGB 类设备对标准 FSM 的描述。
- 如果插入的是 USB 电缆的 A 端，并且 ID 线接地。此时，OTG_FS 控制器将执行一个 ID 线状态改变中断(OTG_FS_GINTSTS 寄存器的 CIDSCHG 位)，自动切换到主机模式，并需要软件初始化主机模式。在这种模式下，OTG_FS 控制器遵守 USB2.0 规范的补充 OTG1.3 规范的的第 6.8.1 节：OTGA 类设备对标准 FSM 的描述。

26.4.2 HNP 双角色设备

USB 全局配置寄存器的 HNP 使能位(OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)允许 OTG_FS 控制器在 A 类主机和 A 类设备，以及 B 类主机和 B 类设备之间通过主机协商协议(HNP)动态地切换。当前设备的状态可以通过读 OTG 全局控制和状态寄存器的 ID 状态位(OTG_FS_GOTGCTL 寄存器的 CIDSTS 位)和全局中断和状态寄存器的当前操作模式位(OTG_FS_GINTSTS 寄存器的 CMOD 位)来综合获得。

HNP 的编程规则请参考第 26.17 节：OTG_FS 编程。

26.4.3 SRP 双角色设备

全球 USB 配置寄存器中的 SRP 能力位(OTG_FS_GUSBCFG 中的 SRPCAP 位)使 OTG_FS 核心能够关闭 A 设备的 VBus 生成，以节省电力。请注意，无论 OTGFS 的主机或外围角色如何，A 设备始终负责驱动 VBuS。

SRP A/B 设备编程模型在第 26.17 节:OTG FS 编程模型中详细描述。

26.5 USB 设备模式

本章介绍 OTG_FS 控制器在 USB 设备模式时的功能。OTG_FS 控制器在以下情况时工作在设备模式下:

- OTGB 类设备
  - 插入的是 USB 电缆的 B 端时，默认的是 OTGB 类设备模式
- OTGA 类设备
  - OTGA 类设备通过 HNP 协议切换到设备角色
- B 类设备
  - 在 ID 线存在时，插入的是 USB 电缆的 B 端，同时 USB 全局配置寄存器的 HNP 位(OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)为'0'时(请参考 OTG1.3 版的第 6.8.3 节)
- 仅作为 USB 设备(请参考下图)
  - USB 全局配置寄存器的强制设备模式位(OTG_FS_GUSBCFG 寄存器的 FDMOD 位)为'1'，将强制使 OTG_FS 控制器工作于 USB 设备模式(请参考 OTG1.3 版的第 6.8.3 节)。在这种情况下 ID 线无论是否存在，其状态都将被忽略。

注意：在设计一个总线供电的仅实现设备模式的 B 类设备时，需要有一个外部的电压变换器，从 VBUS 取电，为芯片的 VDD 供电。

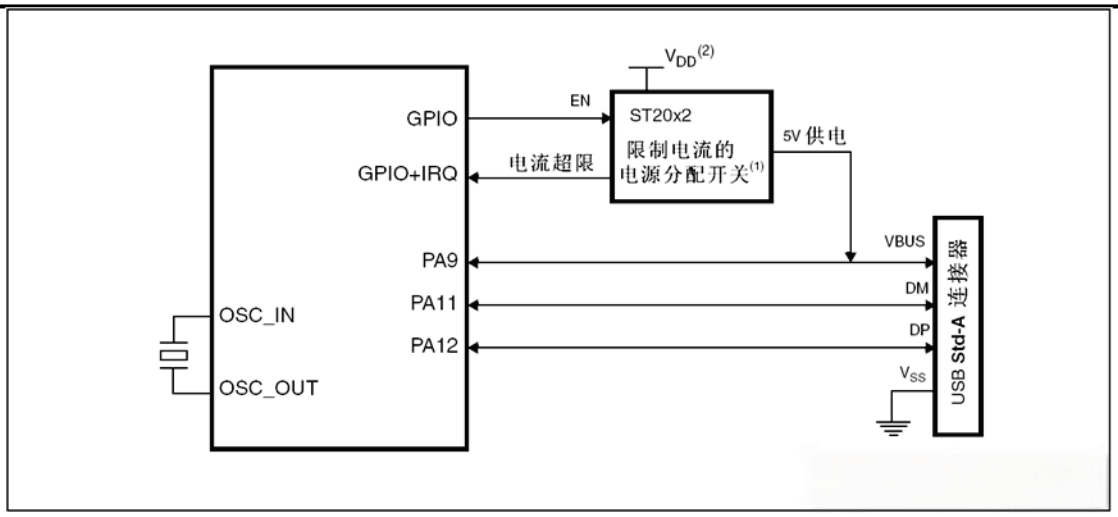


图 275 单纯的 USB 外设连接

1．使用电压变换器来设计一个总线供电的设备

2．V_{DD} 的范围是 2V 到 3.6V

26.5.1 具备 SRP 功能的设备

USB 全局配置寄存器的 SRP 使能位(OTG_FS_GUSBCFG 寄存器的 SRPCAP 位)允许 OTG_FS 控制器实现会话请求协议(SRP)。在这种情况下，A 类设备可以在 USB 会话挂 SRP 协议的编程规则请参考第 26.17.8 节的”B 类设备的会话请求协议”小节。

26.5.2 设备状态

上电状态

VBUS 线上检测到 B 类设备有效的电平后，USB 设备进入上电状态(请参考 USB2.0 的 9.1 章)。OTG_FS 控制器将自动使能 DP 线上的上拉电阻以告知主机接入了一个全速设备，并产生一个会话请求中断(OTG_FS_GINTSTS 寄存器的 SRQINT 位)，标志进入上电状态。

主机在整个 USB 通信阶段都需要提供有效的 VBUS 电平。如果检测到 VBUS 线上的电平低于 B 类有效电平(例如发生了供电扰动，或者主机方关闭了供电)，OTG_FS 控制器将自动断开 USB 连接，并产生会话结束中断(OTG_FS_GOTGINT 寄存器的 SEDET 位)，标志控制器退出上电状态。

上电状态时，OTG_FS 控制器期望从主机接收复位信号，其他 USB 操作都是无效的。当接收到复位信号后，会产生复位中断(OTG_FS_GINTSTS 寄存器的 USBRST 位)。复位完成后，会产生枚举中断(OTG_FS_GINTSTS 寄存器的 ENUMDNE 位)，标志 OTG_FS 控制器进入默认状态。

软件断开

可以使用软件来配置退出上电状态。设置设备控制寄存器的软件断开位(OTG_FS_DCTL 寄存器的 SDIS 位)可以移除 DP 线上的上拉电阻，从而使主机即使在 USB 电缆依然连接时，识别到一个设备断开事件。

默认状态

在默认状态下，OTG_FS 控制器期望接收到 SET_ADDRESS 命令，此时其他操作都是无效的。当接收到一个有效的 SET_ADDRESS 命令后，应用程序需要将相关的地址写入设备配置寄存器的设备地址位(OTG_FS_DCFG 寄存器的 DAD 位)中。OTG_FS 控制器进入地址状态，并准备好响应主机发向这个地址的传输。

挂起状态

OTG_FS 控制器持续的检测 USB 线上的活动。在检测到 3ms 的 USB 空闲状态后，将产生早期挂起中断(OTG_FS_GINTSTS 寄存器的 ESUSP 位)，在 3ms 后产生挂起中断(OTG_FS_GINTSTS 寄存器的 USBSUSP 位)确认挂起。设备状态寄存器的设备挂起位(OTG_FS_DSTS 寄存器的 SUSPSTS 位)将自动置位，OTG_FS 控制器进入挂起状态。

设备可以自发的脱离挂起状态，可以通过置位设备控制寄存器的远程唤醒信号位(OTG_FS_DCTL 寄存器的 WKUPINT 位)，并在 1 到 15ms 之间清除该位来实现。

当检测到主机发出恢复信号，会产生恢复中断(OTG_FS_GINTSTS 寄存器的 RWUSIG 位)，同时设备的挂起位被自动清除。

26.5.3 设备端点

OTG_FS 控制器支持以下 USB 端点：

- 控制端点 0
  - 双向端点，仅用于处理控制信息
  - 对于 IN 和 OUT 两个方向，各有一套独立的寄存器来控制
  - 包括控制(DIEPCTL0/DOEPCTL0)寄存器，传输配置(DIEPTSIZ0/DOEPTSIZ0)寄存器，和状态中断(DIEPINT0/DOEPINT0)寄存器。控制和传输长度寄存器中的某些位与其他端点的寄存器不同。
- 3 个 IN 端点
  - 每个端点都可以配置为同步、块传输或中断传输
  - 每个端点都有控制(DIEPCTLx)寄存器，传输配置(DIEPTSIZx)寄存器和状态中断(DIEPINTx)寄存器
  - 设备 IN 端点通用中断屏蔽寄存器(DIEPMSK)用于使能/禁止所有 IN 端点(包括端点 0)上任一类型的端点中断源
  - 支持未完成的同步 IN 传输中断(OTG_FS_GINTSTS 寄存器的 IISOIXFR 位)，在当前帧有至少一个同步 IN 传输未完成时，会产生该中断。此中断随同周期帧中断产生(OTG_FS_GINTSTS 寄存器的 EOPF)
- 3 个 OUT 端点
  - 每个端点都可以配置为同步、块传输或中断传输
  - 每个端点都有控制(DOEPCTLx)寄存器，传输配置(DOEPTSIZx)寄存器和状态中断(DOEPINTx)寄存器
  - 设备 OUT 端点通用中断屏蔽寄存器(DOEPMSK)用于使能/禁止所有 OUT 端点(包括端点 0)上任一类型的端点中断源
  - 支持未完成的同步 OUT 传输中断(OTG_FS_GINTSTS 寄存器的 INCOMPISOOUT 位)，在当前帧有至少一个同步 OUT 传输未完成时，会产生该中断。此中断随同周期帧中断产生(OTG_FS_GINTSTS 寄存器的 EOPF)

端点配置

- 以下对端点的配置，都通过设置端点 IN/OUT 控制寄存器实现(DIEPCTLx/DOEPCTLx)：
  - 端点使能/禁止
  - 在当前配置中激活端点
  - 配置 USB 传输类型(同步，块传输，中断)
  - 配置最大 USB 数据包长度
  - 配置与 IN 端点相关的发送 FIFO 编号
  - 配置期望收到的或将发送的 PID 包 DATA0/DATA1(仅对块传输和中断传输有效)
  - 配置需要发送或接收的传输帧的奇/偶帧(仅对同步传输有效)
  - 可选的配置：设置 NAK 位，无论 FIFO 的状态如何都回应主机 NAK
  - 可选的配置：设置 STALL 位，对主机发送的任何命令都回应 STALL
  - 可选的配置：设置 OUT 端点 SNOOP 模式，对于接收到的数据不检验 CRC

端点传输

设备端点传输长度寄存器(DIEPTSIZx/DOEPTSIZx)用于配置传输长度和读取传输状态。对该寄存器的配置必需在设置端口控制寄存器的使能位之前。一旦端点被使能，该寄存器便处于只读状态，只有 OTG_FS 控制器可以更新该寄存器的传输状态位。

- 需要配置的传输参数如下：
  - 以字节为单位的单次传输的长度
  - 完成整个传输的 USB 数据包的数目

端点状态/中断

设备端点中断寄存器(DIEPINTx/DOEPINTx)指示了端点与 USB 和 AHB 相关的事件状态。当设置了控制器中断寄存器的 OUT 端点中断位或者 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 位和 IEPINT 位)时，应用程序需要先读取设备所有端点中断寄存器(DAINT)，确定发生中断的端点号，然后读取设备端点中断寄存器，获得中断的详细信息。应用程序必需随即清除该寄存器的相应中断位，使之清除 DAINT 寄存器和 GINTSTS 寄存器的相应中断位。

- 设备控制器提供以下的状态位和中断事件：
  - 传输完成中断，指示了在 AHB 和 USB 方面的数据传输都完成了。
  - SETUP 阶段完成(仅对控制端点有效)
  - 相关的传输 FIFO 已经半空或者全空(IN 端点)
  - 已发送 NAK 响应至主机(仅对同步 IN 传输有效)
  - 当发送 FIFO 为空时，主机发出了 IN 请求(仅对块传输 IN/中断 IN 传输有效)
  - 在端点没有使能时，主机发出了 OUT 请求
  - 检测到串扰错误
  - 软件禁止了端点
  - 软件将端点状态设为 NAK(仅对同步 IN 传输有效)
  - 收到超过 3 个连续的 SETUP 包(仅对控制 OUT 传输有效)
  - 检测到超时错误(仅对控制 IN 传输有效)
  - 同步 OUT 包丢失，没有产生中断

26.6 USB 主机

本章介绍了 OTG_FS 控制器工作在 USB 主机模式时的功能。OTG_FS 控制器在以下条件时工作在主机模式：

- OTGA 类主机
  - 插入的是 USB 电缆的 A 端，默认的是 OTGA 类主机模式
- OTGB 类主机
  - OTGB 类设备通过 HNP 协议切换到主机角色
- A 类设备
  - 在 ID 线存在时，插入的是 USB 电缆的 A 端，同时 USB 全局配置寄存器的 HNP 位为'0'(OTG_FS_GUSBCFG 寄存器的 HNPCAP 位)。此时，内置的 DP/DM 线上的上拉电阻自动有效。
- 仅作为主机(请参考下图)
  - USB 全局配置寄存器的强制主机模式位(OTG_FS_GUSBCFG 寄存器的 FHMOD 位)能强制 OTG_FS 控制器工作的 USB 主机模式。此时即使 USB 连接器上的 ID 线存在，也依然无效。内置的 DP/DM 线上的上拉电阻自动有效。

注意：1. 控制器不能为 5VVBUS 供电，因此需要外接电荷泵，如果应用板能提供 5V 供电，可以使用基本的开关电源来驱动 5V 的 VBUS 线。外接的电荷泵可以用任一通用 I/O 口来控制。在设计 OTGA 类主机，A 类设备和 USB 主机时，都需要这样的设计。

2. 在 USB 通信期间，电荷泵需要一直保证 VBUS 线上的供电，过流输出信号应该连接到配置为中断输入的 I/O 端口上，在发生过流事件时，能产生中断，及时切断 VBUS 线的供电。

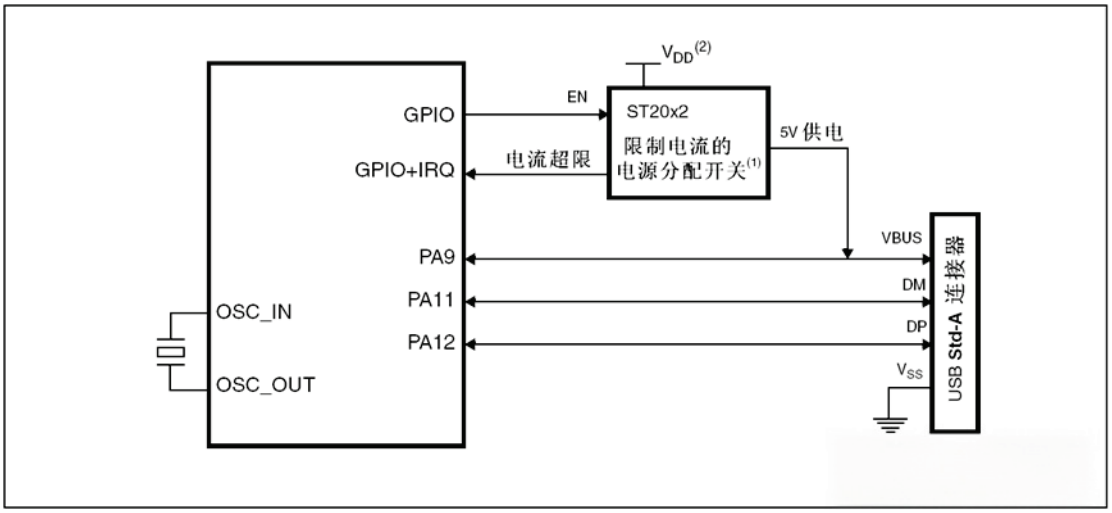


图 276 单纯的 USB 主机连接

1．如果设计的 USB 主机支持 VBUS 总线供电的设备，需要外接 ST20x2。如果应用板上能提供 5V 供电，也可以使用普通电源开关。

2．VDD 的范围是 2V 到 3.6V。

26.6.1 具备 SRP 功能的主机

通过配置 USB 全局配置寄存器的 SRP 使能位(OTG_FS_GUSBCFG 寄存器的 SRPCAP 位)，能够支持 SRP 功能。使能了 SRP 功能后，主机可以在 USB 会话挂起时，关闭 VBUS 线的供电以节省耗电。

SRP 功能的编程实现请参考第 26.17.8 节的”A 类设备的会话请求协议”小节。

26.6.2 USB 主机状态

主机端口供电

控制器内部不提供 VBUS 的 5V 供电，因此需要使用外接的电荷泵来给 VBUS 线供电，如果应用板上能提供 5V 供电，也可以使用普通的电源开关。这个外接的电荷泵可以通过普通的 I/O 口来控制。当应用程序需要设置这个普通 I/O 口来给 VBUS 线供电时，也需要设置主机端口控制和状态寄存器的供电位(OTG_FS_HPRT 寄存器的 PPWR 位)。

VBUS 线使能

在 USB 整个通信过程中，VBUS 线需要一直保证为有效的电平。任何导致 VBUS 线的电平下降到门限(4.25V)以下的事件都将导致由会话中止位触发的 OTG 中断(OTG_FS_GOTGINT 寄存器的 SEDET 位)。此时，应用程序需要关闭对 VBUS 的供电，并清除端口供电位。电荷泵的过流信号也能用来防止电气损害，将此过流信号连接到设置为中断输入的 I/O 口上，一旦产生了过流事件，应用程序需要在中断中立即关闭 VBUS 的供电，并清除端口供电位。

主机对于设备接入的监测

如果 VBUS 线始终没有有效的电平(4.75V)，无论何时插入 USB 设备或者 B 类设备，OTG_FS 控制器都不能监测到设备的插入。

当 VBUS 线处于有效电平范围内，一旦有 B 类设备插入，OTG_FS 控制器就会产生由设备插入位(OTG_FS_HPRT 寄存器的 PCDET 位)触发的主机端口中断。

主机对于设备断开的监测

断开插入的设备，将产生由设备断开事件触发的断开中断(OTG_FS_GINTSTS 寄存器的 DISCINT 位)。

主机枚举

一旦主机检测到有设备插入，需要进入枚举过程，向新插入的设备发送 USB 复位和配置命令。

由于插入的设备在 DP(全速设备)或者 DM(低速设备)线上有上拉电阻，因此会导致总线的不稳定。当总线再次回到稳定状态时，会触发反跳结束的 OTG 中断(OTG_FS_GOTGINT 寄存器的 DBCDNE 位)，此时主机才能向设备发送 USB 复位命令。

应用程序通过置位主机端口控制和状态寄存器的端口复位位(OTG_FS_HPRT 寄存器的 PRST 位)向 USB 总线发出 USB 复位信号(单端 0)。应用程序需要保证该复位信号维持在 10ms 到 20ms 之间，并及时清除端口复位位。

一旦 USB 复位序列完成，会产生由端口使能/禁止改变位触发的主机端口中断(OTG_FS_HPRT 寄存器的 PENCHNG 位)。此中断通知应用程序可以从主机端口控制和状态寄存器获得被枚举的设备的速度信息(OTG_FS_HPRT 寄存器的 PSPD 位)，随之，主机将发送 SOF(全速设备)信号或保持激活(低速状态)。此时，主机可以向设备发送配置命令，完成设备枚举。

主机挂起

应用程序可以通过置位主机控制和状态寄存器的端口挂起位(OTG_FS_HPRT 寄存器的 PSUSP 位)来挂起 USB 活动。此时，OTG_FS 控制器将停止发送 SOF 信号，并进入挂起状态。

控制器可选择支持由远程设备来唤醒主机退出挂起状态。此时，控制器在检测到远程唤醒信号后，将产生远程唤醒中断(OTG_FS_GINTSTS 寄存器的 WKUPINT 位)，随之，主机端口控制和状态寄存

器的唤醒位(OTG_FS_HPRT 寄存器的 PRES 位)将自动置位，控制器将自动向 USB 总线发送唤醒信号。应用程序需要控制唤醒信号的维持时间，并及时的清除端口唤醒位来退出挂起状态并重新开始发送 SOF 信号。

如果需要由主机自发退出挂起状态，应用程序可以设置端口唤醒位，此时唤醒信号将发送到 USB 总线，应用程序需要控制此唤醒信号维持的时间，并及时清除端口唤醒位。

26.6.3 主机通道

OTG_FS 控制器提供 8 个主机通道，每个通道对应一个 USB 主机传输(USBPIPE)。主机不支持同时发起超过 8 个的传输请求。如果应用程序未处理的传输请求超过了 8 个，主机控制器(HCD)必需在通道重新有效后，也就是收到传输完成和通道中止的中断后，重新安排通道。

每个通道都可以配置为输入/输出模式，也可以配置为任意的周期/非周期性传输类型。每个通道都有控制寄存器(HCCHARx)，传输配置寄存器(HCTSIZx)和状态/中断寄存器(HCINTx)以及相应的屏蔽寄存器(HCINTMSKx)。

主机通道控制

- 通道控制寄存器(HCCHARx)可以提供以下操作：
  - 通道使能/禁止
  - 为连接上的 USB 设备配置传输速度：全速/低速
  - 为连接上的 USB 设备配置地址
  - 为连接上的 USB 设备配置端点号
  - 配置传输方向：输入/输出
  - 配置传输类型：控制，块传输，中断，同步
  - 配置最大包长度(MPS)
  - 配置在奇数/偶数帧进行周期性传输

主机通道传输

应用程序通过主机通道传输长度寄存器(HCTSIZx)配置传输长度，并读回传输状态。必需在使能一个传输通道前，配置传输长度。一旦通道被使能，传输长度寄存器即变为只读，只有 OTG_FS 控制器可以更具当前传输状态更新该寄存器。

- 可以配置以下传输参数：
  - 以字节为单位的单个包的传输长度
  - 整个传输过程所有的包个数
  - 起始的数据 PID

主机通道状态/中断

主机通道 x 中断寄存器(HCINTx)指示了通道与 USB 和 AHB 相关的事件。当主机控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)被置位，应用程序需要先读主机所有通道中断寄存器(HCAINT)，获得产生主机通道中断的通道号，然后根据通道号读主机通道 x 中断寄存器(HCINTx)，获得中断的信息。应用程序需要清除主机通道 x 中断寄存器(HCINTx)的相应位，

以便清除 HAINT 和 GINTSTS 寄存器的相应位。可以通过 OTG_FS_HCINTMSKx 寄存器来屏蔽每个通道的中断源。

- 主机控制器提供以下的状态查询和中断：
  - 传输完成中断，指示应用程序端(AHB)和 USB 端的数据传输都已经完成。
  - 由于传输完成，USB 通信错误或者应用程序的禁止命令导致的通道中止
  - 相应的传输 FIFO 已经半空或者全空(仅对 IN 通道有效)
  - 收到 ACK 响应
  - 收到 NAK 响应
  - 收到 STALL 响应
  - 由 CRC 错误、超时、位填充错误或者错误的 EOP 导致的 USB 传输错误
  - 串扰错误
  - 帧溢出
  - 数据 PID(DATA0/DATA1)翻转错误

26.6.4 主机调度器

主机控制器内置一个硬件的调度器，用于自发的管理和驱动应用程序发起的传输请求。在每个帧开始时，主机将先处理周期性(同步和中断传输)的数据传输，再处理非周期性(控制和大容量传输)的数据传输，以便符合 USB 规范对同步和中断传输的高优先级保障。

主机通过请求队列(一个周期性队列和一个非周期性队列)来管理 USB 传输通道。每个请求队列都能保存 8 个请求，而每个请求都代表应用程序提出的一个未处理的传输。每个请求队列中的请求都带有 IN/OUT 方向，通道号以及其他的执行一个 USB 传输的信息。请求写入请求队列的顺序决定了 USB 总线上的传输序列。在每个帧首，主机将先处理周期性请求队列，再处理非周期性请求队列。如果在一个帧结束的时候，仍然有未处理的同步或者中断传输请求，将产生未完成周期性传输中断(OTG_FS_GINTSTS 寄存器的 IPXFR 位)。

周期性队列和非周期性队列的管理完全由 OTG_FS 控制器完成。应用程序可以通过只读的寄存器来获得每个请求队列的状态信息。

- 周期性传输 FIFO 和队列状态寄存器(HPTXSTS)和非周期性传输 FIFO 和队列状态寄存器(GNPTXSTS)包括：
  - 周期和非周期性队列中还可以插入的请求数(最大为 8)
  - 周期性(非周期性)发送 FIFO(对于 OUT 传输)中剩余的可用空间
  - IN/OUT 命令，主机通道数和其他状态信息

由于每个请求队列可以保存至多 8 个请求，因此，应用程序可以尽可能的利用这一特性，最多提交 8 个周期性传输请求和 8 个非周期传输请求，以提高传输效率。例如，对于块传输 OUT/IN 数据，应用程序可以配置传输高达 64(每包的最大字节数)×8(最大请求数) = 512 字节的数据，实现全速设备的最高数据传输速率。这些数据将由主机自动调度而不需要应用程序插入管理。

- 应用程序通过以下步骤向主机调度发起一个周期性(非周期性)的 OUT 传输请求：
  - 配置主机通道的传输参数
  - 使能配置的通道
  - 读 OTG_FS_GNPTXSTS 寄存器的 HPTXSTS 位，确保周期性(非周期性)队列还能容纳至少 1 个请求。

- 读 HPTXSTS(GNPTXSTS)寄存器，确保周期性(非周期性)的发送 FIFO(请参考第 26.11.2 节：主机模式下的发送 FIFO)中剩余足够的空间。如果应用程序在收到周期性(非周期性)发送 FIFO 半空或全空中断后才提交传输请求的话，此步骤可省略。
- 将数据载入相应的 FIFO 中(PUSH 寄存器)。每个通道都会配置一个 PUSH 寄存器。数据将根据 OTG_FS_HCCHARx 寄存器的 EPTYPE 位，自动载入相应的周期性或非周期性发送 FIFO 中。当最后一个 32 位的数据被写入 FIFO 中，一个请求将被插入请求队列的末端，等待调度执行。

- 应用程序通过以下步骤，向主机调度发起一个周期性(非周期性)的 IN 传输请求：
  - 配置主机通道的传输参数
  - 设置主机通道控制寄存器的通道使能位(OTG_FS_HCCHARx 寄存器的 CHENA 位)来使能配置好的通道。此操作将在周期性(非周期性)请求队列的末端插入一个传输请求，并等待调度执行。

26.7 SOF 触发

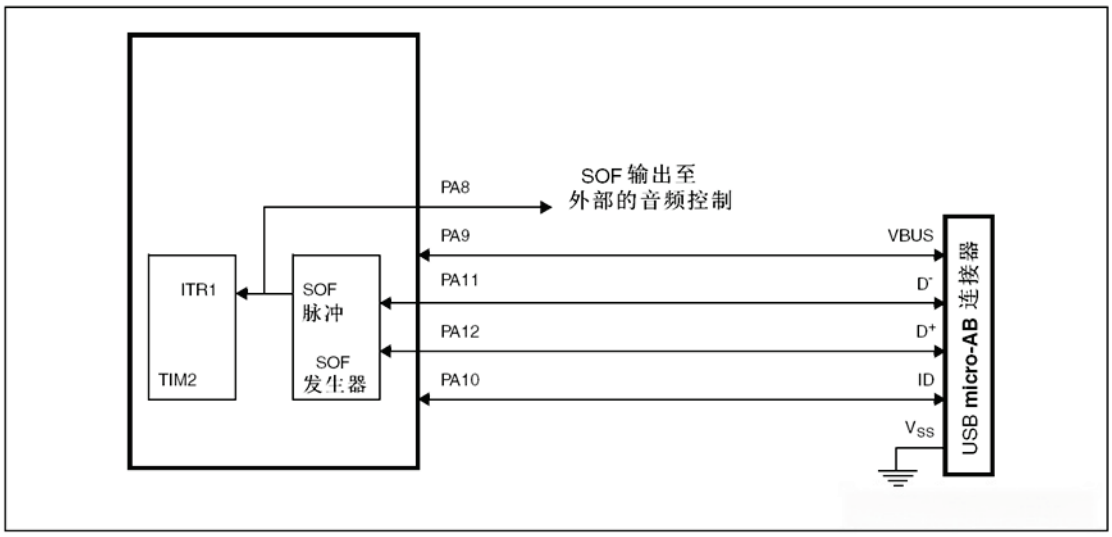


图 277 SOF 连接

OTG_FS 控制器提供：

- 对 SOF 的监控，跟踪和配置
- SOF 脉冲输出

由于音频设备需要与 PC 的数据流同步，或者主机需要根据音频设备的需求调整帧率，因此这些功能对于音频应用的时钟输出非常有用。

26.7.1 主机 SOFs

在主机模式下，可以通过配置主机帧间隔寄存器(HFIR)来设置两个连续的 SOF(全速设备)或者保持有效(低速设备)之间的 PHY 时钟数目，因此，应用程序可以用它来控制 SOF 帧周期。在每个帧首

都可以产生中断(OTG_FS_GINTSTS 寄存器的 SOF 位)。当前的帧号和时间都将保持不变直到在主机帧号寄存器(HFNUM)中出现了下一个帧号。

每个 SOF 首都将产生一个宽度为 12 个系统时钟周期的 SOF 脉冲信号，通过全局控制和配置寄存器的 SOFOUTEN 位，可以配置在 SOF 引脚上输出这个脉冲。SOF 脉冲在内部也连接到定时器 2(TIM2)的触发输入端，因此输入捕获，输出比较和定时器都可以被 SOF 脉冲触发。通过 REMAP_DBGAFR 寄存器的'位 29'可以使能 SOF 脉冲到 TIM2 的连接。

26.7.2 设备 SOFs

在设备模式下，每次在 USB 线上收到 SOF 时，都将产生帧首中断(OTG_FS_GINTSTS 寄存器的 SOF 位)。可以从设备状态寄存器中读出当前帧号(OTG_FS_DSTS 寄存器的 FNSOF 位)。此时，会产生一个宽度为 12 个系统时钟周期的 SOF 脉冲信号，通过使能全局控制和配置寄存器的 SOF 输出使能位(OTG_FS_GCCFG 寄存器的 SOFOUTEN 位)可以在 SOF 引脚上输出这个脉冲信号。SOF 脉冲信号同样在内部连接到定时器 2(TIM2)的触发输入端，此连接通过 REMAP_DBGAFR 寄存器的'位 29'使能，此时输入捕获，输出比较和定时器都可以被 SOF 脉冲触发。

周期性帧结束中断(GINTSTS 寄存器的 EOPF 位)用来告知应用程序，80%，85%，90%或 95%的帧已经完成，此间隔取决于设备配置寄存器的帧间隔位(OTG_FS_DCFG 寄存器的 PFIVL 位)。此功能用于判断一个帧内所有的同步传输是否都已经完成。

26.8 供电选项

表 149 W55MH32 低功耗和 OTG 的兼容性

模式	描述	USB 兼容性
运行	MCU 完全激活	在 USB 不在挂起状态时需要。
睡眠	USB 暂停退出会导致设备退出睡眠模式。外设寄存器内容将被保留。	在 USB 处于挂起状态时可用。
停止	USB 暂停退出导致设备退出停止模式。外设寄存器内容保留 (1)	在 USB 处于挂起状态时可用。
待机	关闭电源。退出待机模式后，必须重新初始化外围设备。	不兼容 USB 应用程序。

1.在停止模式下，有不同的可能设置。也可能存在一些限制，请参阅第 5 节:电源控制(PWR)，以了解在使用 OTG 时适用哪些(如果有)限制。

OTGPHY 的耗电由通用控制器配置寄存器的以下 3 位决定:

- PHY 供电位(GCCFG 寄存器的 PWRDWN 位)
  - 切换 PHY 全速收发器模块的开/关。必需预先设置才能允许 USB 通信。
- A 类 VBUS 监控使能位(GCCFG 寄存器的 VBUSASEN 位)
  - 切换 A 类设备 VBUS 比较器的开/关。在 A 类设备模式时(USB 主机)，在 HNP 阶段必需使能该位。
- 类 VBUS 监控使能位(GCCFG 寄存器的 VBUSASEN 位)
  - 切换 B 类设备 VBUS 比较器的开关。在 B 类设备模式时(USB 设备)，在 HNP 阶段必需使能该位。

当 USB 无通信或无 USB 设备连接时，暂停 USB，可以节省供电。

- 停止 PHY 时钟(OTG_FS_PCGCCTL 寄存器的 STPPCLK 位)

- 当设置时钟门控控制寄存器的停止 PHY 时钟位时，大部分内部连接到 OTG 全速控制器的 48MHz 时钟通过门控被关闭，此时，即使应用程序仍然使能 48MHz 的输入时钟，但由 USB 时钟造成的动态功耗会大大下降。
- 大部分的收发器会被禁止，但用于检测异步复位信号和远程唤醒事件的电路仍然处于激活状态。

- 门控 HCLK(OTG_FS_PCGCCTL 寄存器的 GATEHCLK 位)
  - 当设置时钟门控寄存器的门控 HCLK 位时，大部分内部连接到 OTG_FS 控制器的系统时钟会被门控电路关闭。只有寄存器的访问接口仍然保持有效。此时，即使应用程序仍然使能系统时钟，但由 USB 时钟造成的动态功耗会大大下降。
- USB 系统停止
  - 当 OTG_FS 控制器处于 USB 挂起状态时，应用程序需要完全关闭所有的系统时钟源来彻底降低所有功耗。先置位 PHY 时钟停止位，再设置供电控制系统模块(PWR)进入系统深度睡眠模式，将停止 USB 系统。
  - 在检测到远程唤醒(作为主机)或来自 USB 总线的唤醒信号(作为设备)时，OTG_FS 控制器将自动恢复系统活动和 USB 的时钟。

为了降低功耗，只有在 OTG_FS 控制器需要访问 FIFO 时，才向 USB 数据 FIFO 提供时钟。

26.9 OTG FS HFIR 寄存器的动态更新

USB 核心在主机模式下嵌入了 SOF 帧周期的动态修剪功能，允许将外部设备与 SOF 帧同步。当在当前 SOF 帧内更改 OTG_FS_HFIR 寄存器时，SOF 周期校正将在下一个帧中应用，如图 278 所示

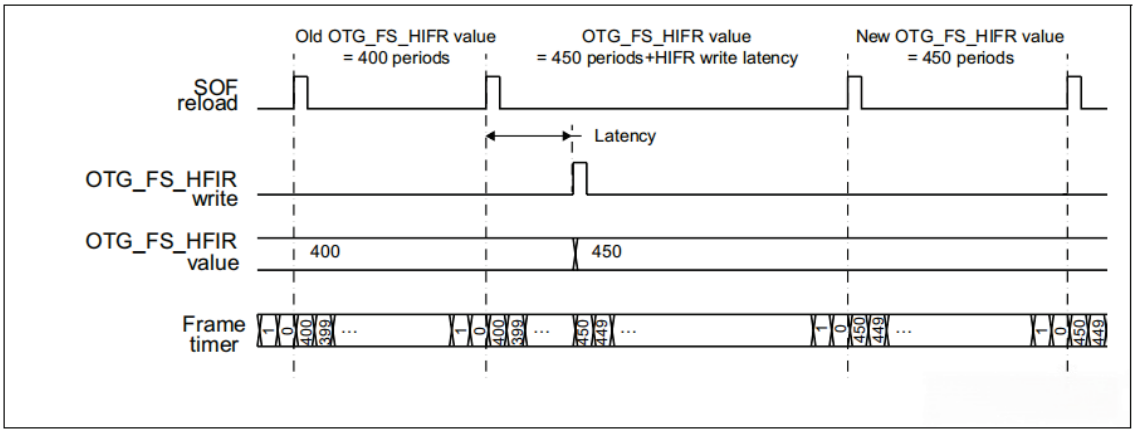


图 278 动态更新 OTG FS HFIR

26.10 USB 数据 FIFO

下图示出 OTG_FS 控制器的框图和各个部分的功能。

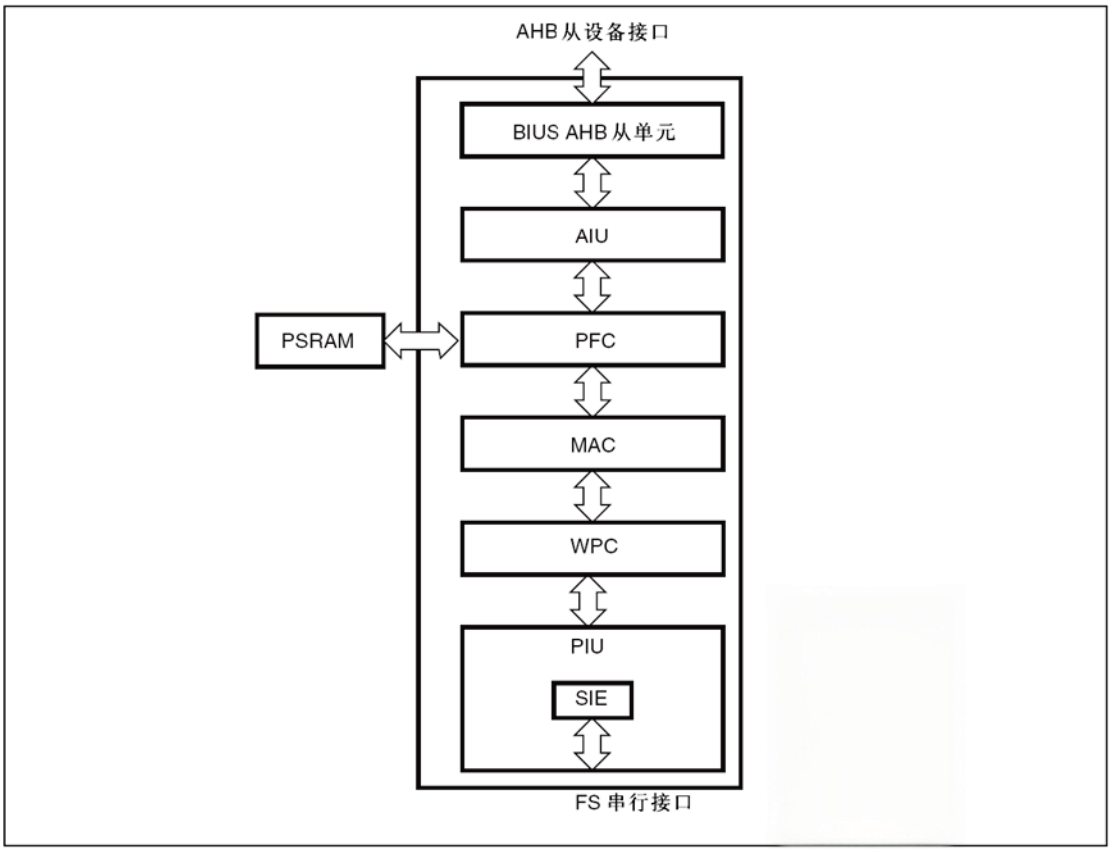


图 279 OTG_FS 控制器框图

BIUS：总线接口单元；AIU：应用程序接口单元；PFC：包 FIFO 控制器；MAC：媒体访问控制器；WPC：唤醒和供电控制器；PIU：PHY 接口单元；SIE：串行接口控制器

USB 系统规划了 1.25K 字节的专用 RAM 用于 FIFO 的管理。OTG_FS 控制器的包 FIFO 控制器(PFC)模块将此 RAM 划分为发送 FIFO 区域和接收 FIFO 区域，应用程序将数据暂时缓存到发送 FIFO 中，等待发送，而从 USB 总线上收到的数据则暂时缓存在接收 FIFO 中，等待应用程序读取。实际的 FIFO 数量及如何在专用 RAM 中规划这些 FIFO 都由实际的应用决定。比如在设备模式下，每个 IN 端点都会配置一个发送 FIFO，而这些 FIFO 的大小都可以由软件指定，这样，专用的 RAM 可以最优化的满足应用的需求。

26.11 设备模式下的 FIFO 结构

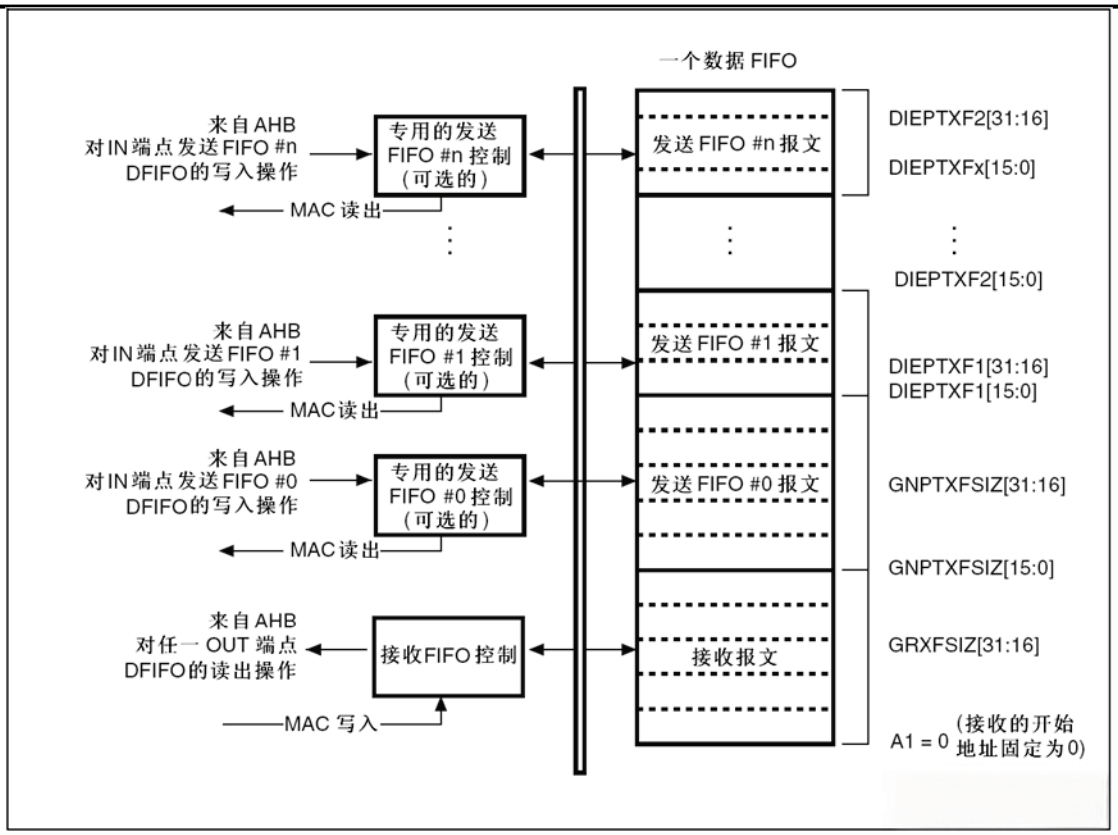


图 280 设备模式下的 FIFO 地址映像和 AHB 的 FIFO 操作映像

26.11.1 设备模式下的接收 FIFO

FS_OTG 控制器在设备模式下，使用一个单独的 FIFO 缓存所有 OUT 端点的数据。收到的数据包依次缓存在 RXFIFO 的可用空间中。收到的数据包的状态(包括 OUT 端点号，字节数，数据 PID 号和收到数据的有效性)由 PFC 模块写在收到的数据前。如果接收 FIFO 没有剩余空间，控制器将以 NACK 来回应主机，同时产生相应端点的中断。接收 FIFO 的大小可由接收 FIFO 长度寄存器(GRXFSIZ)配置。

使用单个 RXFIFO 的架构，使得 USB 设备能更有效的利用整个接收 RAM 空间。

- 所有的 OUT 端点共享整个 RAM 空间(共享的 FIFO)
- OTG_FS 控制器可以最大限度的按照主机发送 OUT 数据的次序来利用接收 FIFO

只要接收 FIFO 还有至少一个数据包等待应用程序获取，应用程序就会不断收到接收 FIFO 非空中断(OTG_FS_GINTSTS 寄存器的 RXFLVL 位)。应用程序可以通过读接收状态读和弹出寄存器(GRXSTSP)获得数据包信息，并通过读相应端点的 POP 寄存器从接收 FIFO 中获取数据包。

26.11.2 设备模式下的发送 FIFO

共享 FIFO 的模式不适用于 IN 传输，只有提前知道主机请求的次序或者能预测进程的处理过程才能将所有端点的数据包都按次序传送到同一个发送 FIFO 中。所以在设备模式下，控制器为每个 IN 端点都配置了一个专用的 FIFO。应用程序可以通过非周期性传输 FIFO 长度寄存器(GNPTXFSIZ)来配置 IN 端点 0 的 FIFO 长度，并通过设备 IN 端点传输 FIFO 寄存器(DIEPTXFx)来配置 IN 端点 x 的 FIFO 长度。

专用的 FIFO 的架构非常灵活，大大减少应用程序的负荷。这些 FIFO 没有请求序列，也没有必要在 USB 主机访问非周期性端点的时候预测访问的顺序。

据根在 AHB 配置寄存器中配置的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)的值，OTG_FS 控制器会产生发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)，提示对应 IN 端点的发送 FIFO 已经半空或者全空。应用程序先通过读设备所有端点中断寄存器(DAINT)获得需要服务的IN端点号，然后通过读设备IN端点x的发送FIFO状态寄存器(DTXFSTSx)检查 FIFO 是否有足够的剩余空间，最后通过写相应端点的 PUSH 寄存器来向发送 FIFOx 传送数据。

26.12 主机模式下的 FIFO 结构

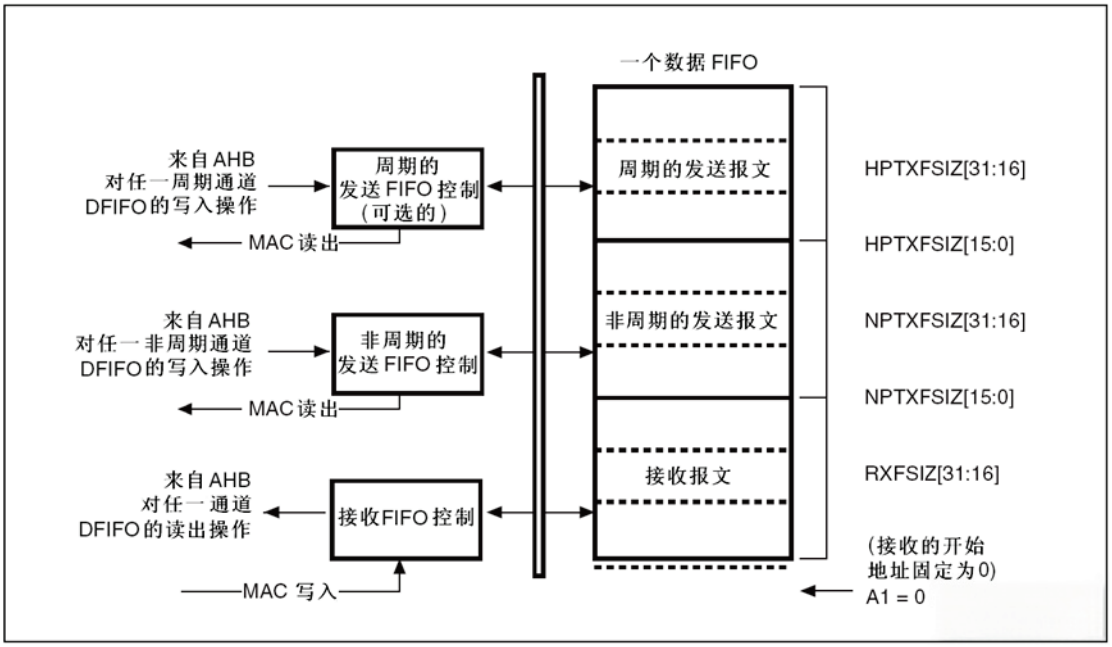


图 281 主机模式 FIFO 地址映像和 AHB 的 FIFO 操作映像

26.12.1 主机模式下的接收 FIFO

在主机模式下使用一个接收 FIFO 管理所有的周期性和非周期性的传输，此 FIFO 用于暂存从 USB 总线上收到但还未传输到系统存储区的数据(收到的数据包)。来自任一远程 IN 端点的数据包都将按次序暂存在 FIFO 中。收到数据包的状态，包括主机通道号，字节数，数据 PID 和收到数据的有效性也一同储存在 FIFO 中。应用程序可以通过接收 FIFO 长度寄存器(GRXFSIZ)来配置这个接收 FIFO 的长度。

使用单个接收 FIFO 的架构，使得 USB 主机能更有效的利用整个接收 RAM 空间。

- 所有配置好的 IN 通道共享整个 RAM 空间(共享的 FIFO)
- OTG_FS 控制器可以最大限度的按照主机发送 IN 命令的序列来利用接收 FIFO

只要接收 FIFO 还有至少一个数据包等待应用程序获取，应用程序就会收到接收 FIFO 非空中断。应用程序可以通过读接收状态读和弹出寄存器获得数据包信息，并通过读相应端点的 POP 寄存器从接收 FIFO 中获取数据包。

26.12.2 主机模式下的发送 FIFO

在主机模式下，控制器使用一个发送 FIFO 来管理所有的非周期性(控制和块传输)OUT 传输，使用另一个发送 FIFO 来管理所有的周期性(同步和中断)OUT 传输。这两个 FIFO 用于暂存需要发送到

USB 总线的数据(发送的数据包)。可以通过主机周期性(非周期性)传输 FIFO 长度寄存器(HPTXFSIZ/GNPTXFSIZ)来配置周期性(非周期性)发送 FIFO 的长度。

使用两个发送 FIFO 是因为要保证一个 USB 帧里周期性传输的高优先级。在每个帧开始的时候，内置的主机调度器先处理周期性请求队列再处理非周期性请求队列。

使用两个发送 FIFO 使 USB 主机能方便的分开优化管理周期性和非周期性的数据缓存。

- 所有用于周期性(非周期性)OUT 传输的主机通道共享同一块 RAM 缓存区(共享的 FIFO)
- OTG_FS 控制器可以根据主机软件对 OUT 命令的调度最大限度的利用周期性(非周期性)的发送 FIFO。

根据在 AHB 配置寄存器中配置的周期性发送 FIFO 空标志位(OTG_FS_GAHBCFG 寄存器的 PTXFELVL 位)的值，OTG_FS 控制器会产生周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 PTXFE 位)，提示周期性发送 FIFO 已经半空或者全空。应用程序需要先读周期性发送 FIFO 和队列状态寄存器(HPTXSTS)来获得周期性发送 FIFO 和周期性请求队列是否有剩余空间的信息，只有在发送 FIFO 和请求队列都有剩余空间的情况下，应用程序才能将数据写入 FIFO 中。

根据在 AHB 配置寄存器中配置的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)的值，OTG_FS 控制器会产生非周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)，提示非周期性发送 FIFO 已经半空或者全空。应用程序需要先读非周期性发送 FIFO 和队列状态寄存器(GNPTXSTS)来获得非周期性发送 FIFO 和非周期性请求队列是否有剩余空间的信息，只有在发送 FIFO 和请求队列都有剩余空间的情况下，应用程序才能将数据写入 FIFO 中。

26.13 FIFO RAM 分配

26.13.1 设备模式

接收 FIFO RAM 分配:应用程序应为 SETUP 数据包分配 RAM:在接收 FIFO 中必须保留 10 个位置，以在控制端点接收 SETUP 数据包。核心不会使用这些为 SETUP 数据包保留的位置来写入任何其他数据。应分配一个位置用于全局 OUTNAK。状态信息每个接收到的数据包都会被写入 FIFO。因此，必须至少分配(最大数据包大小/4)+1 的空间来接收数据包。如果启用了多个等时端点，则必须至少分配两个(最大数据包大小/4)+1 的空间来接收连续的数据包。通常，建议分配两个(最大数据包大小/4)+1 的空间，以便在将前一个数据包传输到 CPU 时，USB 可以接收后续数据包。

与每个端点的最后一个数据包一起，传输完整的状态信息也会被推送到 FIFO。通常，建议每个 OUT 端点都有一个位置。

传输 FIFO RAM 分配:每个 IN 端点所需的最小 RAM 空间传输 FIFO 是特定 IN 端点的最大数据包大小。

注意: 在传输 IN 端点 FIFO 中分配的更多空间会带来更好的 USB 性能。

26.14 USB 系统性能

通过配置大容量的 RAM 缓存区、高自由度的 FIFO 长度配置、AHBPUSH/POP 寄存器的 32 位快速访问、和高级的 FIFO 控制机制，使得 OTG_FS 控制器能最大限度的利用 RAM 空间而不必顾虑 USB 数据的次序，USB 系统达到了最优的性能。

这些优势如下：

- 应用程序的负荷减少，不用干预 USB 传输，优化了 CPU 带宽的使用率
  - 当数据通过 USB 系统有效率的传送时，应用程序可以提前准备好大量的传输数据。
  - 应用程序获得了大量的时间从一个单一的接收 FIFO 中获取数据
- USB 控制器可以控制自身的全部工作效率，也就是说，与需要应用程序干预相比，现在的架构可以提供最高的全速带宽和最大幅度的自治。
  - USB 控制器管理一个大型的数据缓存区，可以自主的管理数据在 USB 总线上的传输
  - USB 控制器管理一个大型的数据接收缓存区，可以自主的从 USB 总线上接收数据并填入缓区

由于 OTG_FS 控制器可以非常有效的使用 1.25K 字节的 RAM 缓存区，并且对于一个全速的帧来说 1.25K 字节已经足够用于管理发送/接收的数据，因此 USB 系统能在每个 USB 帧内(1ms)提供最大的全速数据传输率。

26.15 OTG_FS 中断

无论 OTG_FS 控制器工作在设备模式还是主机模式，应用程序都不能访问另一个模式下的寄存器组。如果应用程序有非法的访问，会产生模式不匹配中断并影响控制器中断寄存器的相应位(OTG_FS_GINTSTS 寄存器的 MMIS 位)。当控制器从一种模式切换到另一种模式时，新模式下的寄存器组都需要和上电复位时一样的重新初始化。

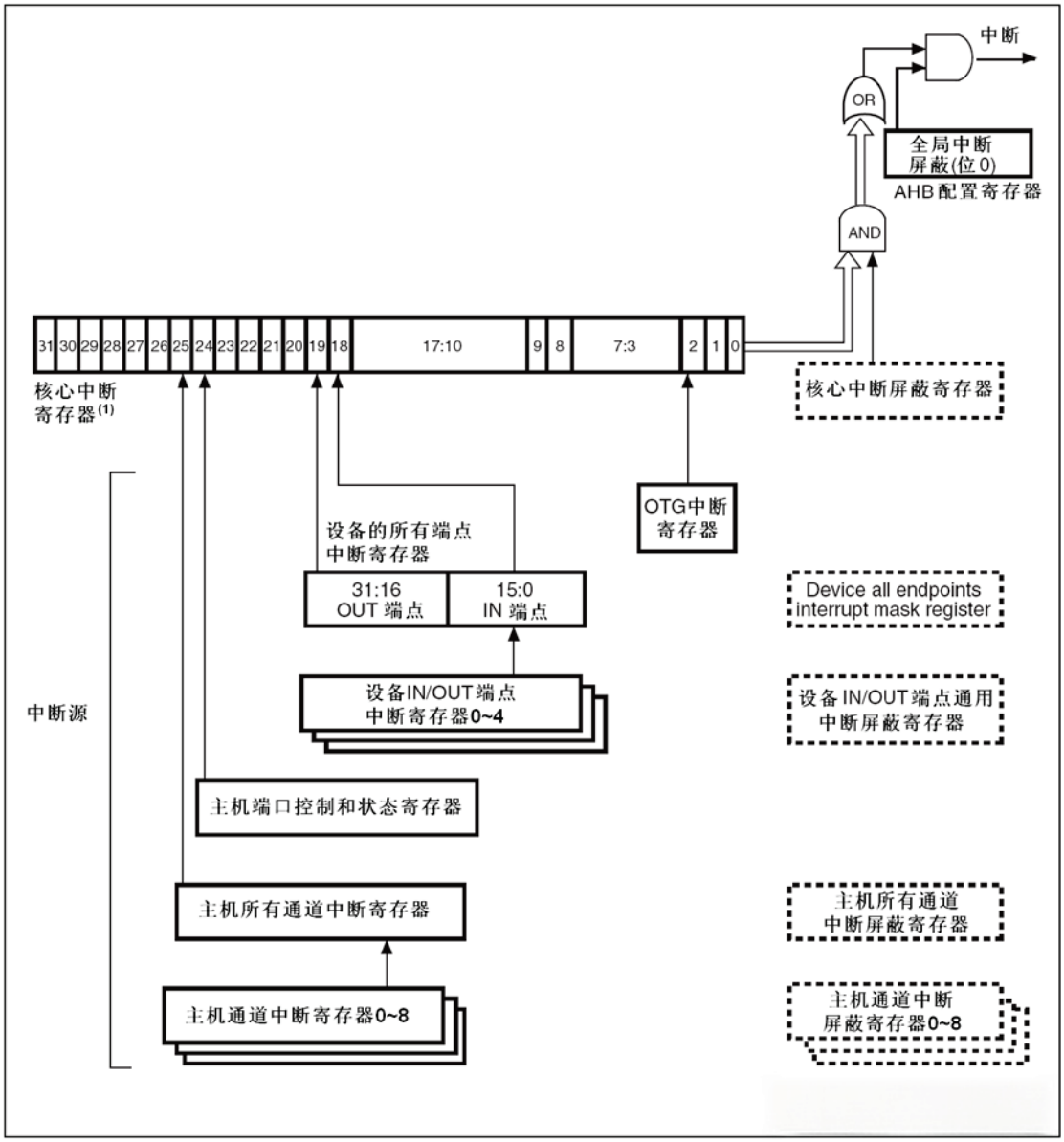


图 282 中断架构

请参考控制器中断寄存器的相关位

26.16 OTG_FS 控制和状态寄存器

应用程序通过 AHB 从接口来读写控制和状态寄存器(CSRx)，从而实现对 OTG_FS 控制器的控制。这些寄存器都是 32 位访问的，并且 32 位地址对齐。包括以下寄存器组：

- 控制器全局寄存器组
- 主机模式寄存器组
- 主机全局寄存器组
- 主机端口 CSR 寄存器组
- 主机通道相关寄存器组
- 设备模式寄存器组
- 设备全局寄存器组
- 设备端点相关寄存器组
- 供电和时钟控制寄存器组
- 数据 FIFO(DFIFO)访问寄存器组

只有控制器全局寄存器，供电和时钟控制寄存器，数据 FIFO 访问寄存器以及主机端口控制和状态寄存器在主机模式和设备模式下都有效。无论 OTG_FS 控制器工作在主机模式还是设备模式下，应用程序都不能访问另一种模式下的寄存器组。如果应用程序发生了非法访问，会产生模式不匹配中断并影响控制器中断寄存器的相应位(OTG_FS_GINTSTS 寄存器的 MMIS 位)。当控制器从一种模式切换到另一种模式时，新模式下的寄存器组都需要和上电复位时一样的重新初始化。

必须以字(32 位)的方式操作这些外设寄存器。

26.16.1 CSR 存储器映像

主机模式寄存器和设备模式寄存器占据不同的地址。所有的寄存器都由 AHB 时钟驱动。

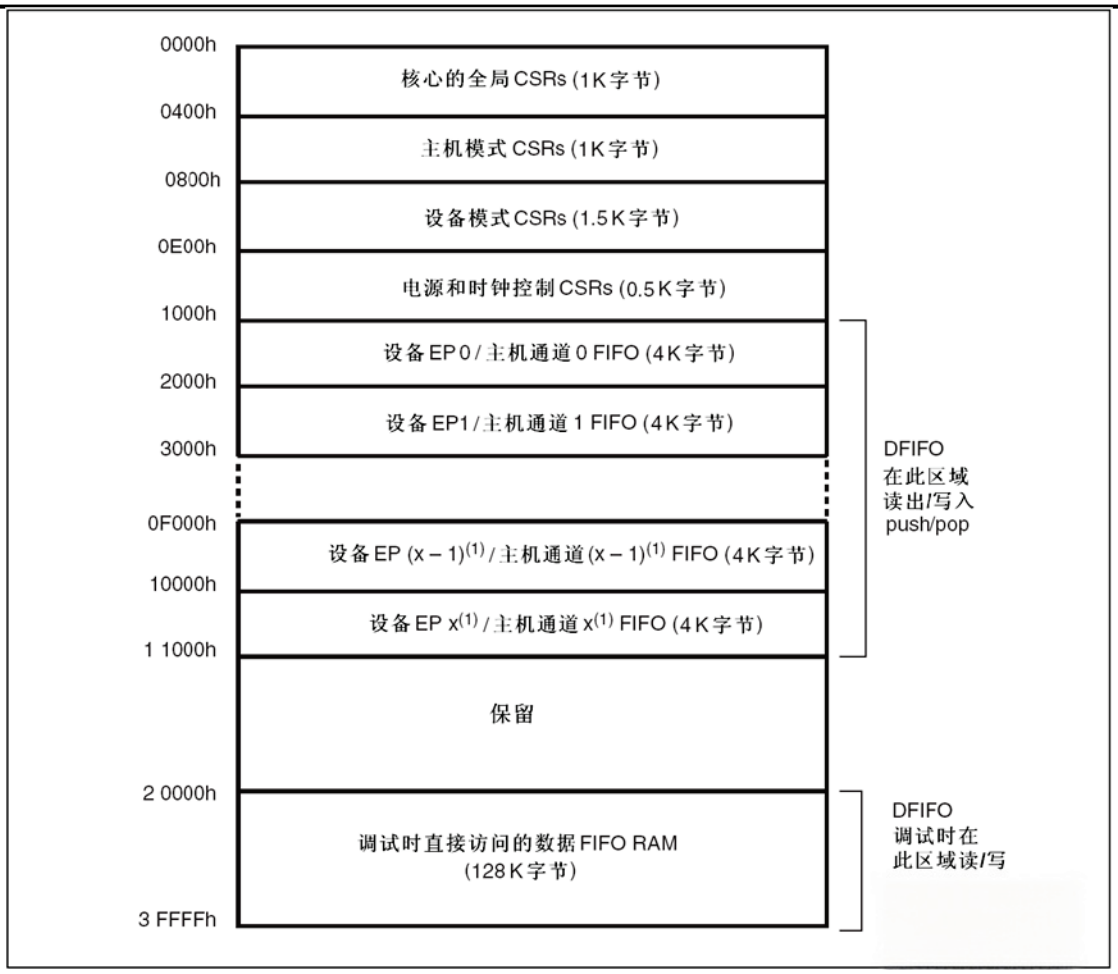


图 283 CSR 存储器映像

在设备模式下 x 为 3，在主机模式下 x 为 7

全局 CSR 地址映像

这些寄存器在主机模式和设备模式下都有效。

表 150 控制器全局控制和状态寄存器(CSRs)

寄存器代号	偏移地址	寄存器名
OTG_FS_OTGCTL	0x000	OTG_FS 控制和状态寄存器
OTG_FS_GOTGINT	0x004	OTG_FS 中断寄存器
OTG_FS_GAHBCFG	0x008	OTG_FSAHB 配置寄存器
OTG_FS_GUSBCFG	0x00C	OTG_FSUSB 配置寄存器
OTG_FS_GRSTCTL	0x010	OTG_FS 复位寄存器
OTG_FS_GINTSTS	0x014	OTG_FS 控制器中断寄存器
OTG_FS_GINTMSK	0x018	OTG_FS 中断屏蔽寄存器
OTG_FS_GRXSTSR	0x01C	OTG_FS 接收状态调试读/OTG 状态读和弹出寄存器
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	OTG_FS 接收 FIFO 长度寄存器
OTG_FS_GNPTXFSIZ	0x028	OTG_FS 非周期性发送 FIFO 长度寄存器
OTG_FS_GNPTXSTS	0x02C	OTG_FS 非周期性发送 FIFO/队列状态寄存器
OTG_FS_GCCFG	0x038	OTG_FS 通用控制配置寄存器
OTG_FS_CID	0x03C	OTG_FS 控制器 ID 寄存器
OTG_FS_HPTXFSIZ	0x100	OTG_FS 主机模式周期性发送 FIFO 长度寄存器
OTG_FS_DIEPTXFx	0x104 0x108 0x10C	OTG_FS 设备模式 IN 端点 TXFIFO 长度寄存器(x=1...4，指明 FIFO 的编号)

1.一般规则是将 OTG FS HNPTXFSIZ 用于主机模式，将 OTG·FS_DIEPTXF0 用于设备模式。

主机模式 CSR 地址映像

这些寄存器在每次切换到主机模式时都需要配置。

表 151 主机模式下的控制和状态寄存器(CSRs)

寄存器代号	偏移地址	寄存器名
OTG_FS_HCFG	0x400	OTG_FS 主机模式配置寄存器
OTG_FS_HFIR	0x404	OTG_FS 主机模式帧间隔寄存器
OTG_FS_HFNUM	0x408	OTG_FS 主机模式帧号码/剩余帧时间寄存器
OTG_FS_HPTXSTS	0x410	OTG_FS 主机模式周期性 TXFIFO/队列状态寄存器
OTG_FS_HAINT	0x414	OTG_FS 主机模式所有通道中断寄存器
OTG_FS_HAINTMSK	0x418	OTG_FS 主机模式所有通道中断屏蔽寄存器
OTG_FS_HPRT	0x440	OTG_FS 主机模式端口控制和状态寄存器
OTG_FS_HCCHARx	0x500 0x520 … 0x5E0	OTG_FS 主机模式通道 x 特性寄存器(x=0...7，x 指示通道编号)
OTG_FS_HCINTx	0x508	OTG_FS 主机模式通道 x 中断寄存器(x=0...7，x 指示通道编号)
OTG_FS_HCINTMSKx	0x50C	OTG_FS 主机模式通道 x 中断屏蔽寄存器(x=0...7，x 指示通道编号)
OTG_FS_HCTSIZx	0x510	OTG_FS 主机模式通道 x 传输长度寄存器(x=0...7，x 指示通道编号)

设备模式 CSR 地址映像

这些寄存器在每次切换到设备模式时都必须配置。

表 152 设备模式控制和状态寄存器

寄存器代号	偏移地址	寄存器名
OTG_FS_DCFG	0x800	OTG_FS 设备模式配置寄存器
OTG_FS_DCTL	0x804	OTG_FS 设备模式控制寄存器
OTG_FS_DSTS	0x808	OTG_FS 设备模式状态寄存器
OTG_FS_DIEPMSK	0x810	OTG_FS 设备模式 IN 端点共有中断屏蔽寄存器
OTG_FS_DOEPMSK	0x814	OTG_FS 设备模式 OUT 端点共有中断屏蔽寄存器
OTG_FS_DAINT	0x818	OTG_FS 设备模式所有端点中断寄存器
OTG_FS_DAINTMSK	0x81C	OTG_FS 设备模式所有端点中断屏蔽寄存器
OTG_FS_DVBUSDIS	0x828	OTG_FS 设备模式 VBUS 下电时间寄存器
OTG_FS_DVBUSPULSE	0x82C	OTG_FS 设备模式 VBUS 脉冲时间寄存器
OTG_FS_DIEPEMPMSK	0x834	OTG_FS 设备模式 IN 端点 FIFO 空中断屏蔽寄存器
OTG_FS_DIEPCTL0	0x900	OTG_FS 设备模式 IN 控制端点 0 控制寄存器
OTG_FS_DIEPCTLx	0x920 0x940 0x960	OTG_FS 设备模式 IN 端点 x 控制寄存器(x=1...3，x 指示端点编号)
OTG_FS_DIEPINTx	0x908	OTG_FS 设备模式 IN 端点 x 中断寄存器(x=1...3，x 指示端点编号)
OTG_FS_DIEPTSIZ0	0x910	OTG_FS 设备模式 IN 端点 0 传输长度寄存器
OTG_FS_DTXFSTSx	0x918	OTG_FS 设备模式 IN 端点 TXFIFO 状态寄存器(x=1...3，x 指示端点编号)
OTG_FS_DIEPTSIZx	0x930 0x950 0x970	OTG_FS 设备模式 IN 端点 x 传输长度寄存器(x=1...3，x 指示端点编号)
OTG_FS_DOEPCTL0	0xB00	OTG_FS 设备模式 OUT 控制端点 0 控制寄存器

OTG_FS_DOEPCTLx	0xB20 0xB40 0xB60	OTG_FS 设备模式 OUT 端点 x 控制寄存器(x=1...3，x 指示端点编号)
OTG_FS_DOEPINTx	0xB08	OTG_FS 设备模式 OUT 端点 x 中断寄存器(x=1...3，x 指示端点编号)
OTG_FS_DOEPTSIZx	0xB10	OTG_FS 设备模式 OUT 端点 x 传输长度寄存器(x=1...3，x 指示端点编号)
OTG_FS_DOEPTSIZx	0xB30 0xB50 0xB70	OTG_FS 设备模式 OUT 端点 x 传输大小寄存器(x=1...3，x 指示端点编号)

数据 FIFO(DFIFO)访问寄存器址映射

这组寄存器列在主机模式和设备模式下都有效，用于读写指定方向的特殊端点或通道的 FIFO。如果一个主机模式下的通道是 IN 类型的，相对应的 FIFO 只能进行读操作。同样地，如果一个主机模式下的通道是 OUT 类型的，相对应的 FIFO 只能进行写操作。

表 153 数据 FIFO(DFIFO)访问寄存器

数据 FIFO(DFIFO)访问寄存器段	地址范围	访问方式
设备模式下 IN 端点 0/主机模式下 OUT 通道 0：DFIFO 只写	0x1000~0x1FFC	写操作
设备模式下 OUT 端点 0/主机模式下 IN 通道 0：DFIFO 只读		读操作
设备模式下 IN 端点 1/主机模式下 OUT 通道 1：DFIFO 只写	0x2000~0x2FFC	写操作
设备模式下 OUT 端点 1/主机模式下 IN 通道 1：DFIFO 只读		读操作
.....	.....	.....
设备模式下 IN 端点 x/主机模式下 OUT 通道 x：DFIFO 只写	0xX000~0xXFFC	写操作
设备模式下 OUT 端点 x/主机模式下 IN 通道 x：DFIFO 只读		读操作

表中的 x 在设备模式下为 3，在主机模式下为 7

供电和时钟控制 CSR 寄存器映像

只有一个寄存器用来控制供电和时钟控制，此寄存器在设备模式下和主机模式下都有效。

表 154 供电和门控控制和状态寄存器

寄存器名	缩写	偏移地址
供电和门控时钟控制寄存器	PCGCR	0xE00~0xE04
保留		0xE05~0xFFF

26.16.2 OTG_FS 全局寄存器

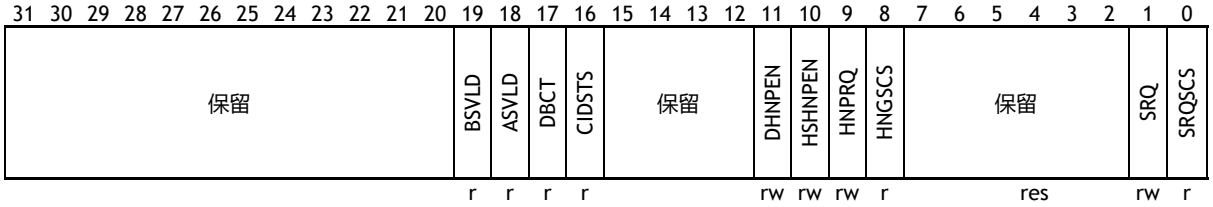
这些寄存器在设备模式和主机模式下都有效，并且在模式切换时不需要重新初始化。如果没有特别注明，寄存器中每位的值都由二进制数表示。

OTG_FS 控制和状态寄存器(OTG_FS_GOTGCTL)

偏移地址：0x000

复位值：0x0001 0000

OTG 控制和状态寄存器控制 OTG_FS 控制器的操作，并反应控制器的状态。



位	符号	说明
31:20	Reserved	保留位，硬件强制为 0
19	BSVLD	**BSVLD**：B 类会话有效(B-session valid)指示设备模式收发器的状态 0：B 类会话无效； 1：B 类会话有效。 在 OTG 模式，用户可以用此位来判断设备是否连入主机 注意：仅在设备模式下有效。
18	ASVLD	**ASVLD**：A 类会话有效(A-session valid)指示主机模式收发器的状态 0：A 类会话无效； 1：A 类会话有效。 注意：仅在主机模式下有效。
17	DBCT	**DBCT**：长/短反射时间(Long/short debounce time)指示检测到的连接的反射时间 0：长的反射时间，用于物理连接(100ms+2.5us)； 1：短的反射时间，用于软件连接(2.5us)。 注意：仅在主机模式下有效。
16	CIDSTS	**CIDSTS**：连接的 ID 线状态(Connector ID status)指示连接器上的 ID 线状态 0：OTG_FS 处于 A 类设备模式； 1：OTG_FS 处于 B 类设备模式。 注意：在设备模式和主机模式下都有效。
15:12	Reserved	保留
11	DHNPEN	**DHNPEN**：设备模式 HNP 使能(Device HNP enabled) 应用程序在成功收到一个来自 USB 主机的 SetFeature.SetHNPEnable 命令时设置此位。 0：禁止设备模式 HNP； 1：使能设备模式 HNP。 注意：仅在设备模式下有效。
10	HSHNPEN	**HSHNPEN**：主机模式 HNP 使能(Host set HNP enable) 应用程序在成功使能连接上的设备的 HNP 时(通过 SetFeature.SetHNPEnable 命令)设置此位。 0：禁止主机模式 HNP； 1：使能主机模式 HNP。 注意：仅在主机模式下有效。
9	HNPRQ	**HNPRQ**：HNP 请求(HNP request) 应用程序在需要向 USB 主机发送一个 HNP 请求时设置此位。当 OTG 中断寄存器的主机协商成功状态转换位(OTG_FS_GOTGINT 寄存器的 HNSSCHG 位)被置位时，应用程序可以通过写'0'清除此位。控制器在清除 HNSSCHG 位时清除此位。 0：无 NHP 请求；

		1：HNP 请求。 注意：仅在设备模式下有效。
8	HNGSCS	**HNGSCS**：主机协商成功(Host negotiation success) 当主机协商成功时，控制器会设置此位。当应用程序设置 HNP 请求位(HNPRQ)时，控制器会清除此位。 0：主机协商失败； 1：主机协商成功。 注意：仅在设备模式下有效。
7:2	Reserved	保留
1	SRQ	**SRQ**：会话请求(Session request) 应用程序通过设置此位在 USB 总线上发起一个会话请求。当 OTG 中断寄存器的主机协商成功状态改变位(OTG_FS_GOTGINT 寄存器的 HNSSCHG 位)被置位时，应用程序可以通过写'0'来清除此位。控制器在清除 HNSSCHG 位时清除此位。 如果用户使用 USB1.1 全速串行收发器接口来发起一个会话请求，应用程序必需在 B 类会话有效位(OTG_FS_GOTGCTL 寄存器的 BSVLD 位)被清除后，等待 VBUS 线降到 0.2V。不同的 PHY 使用不同的等待时间，由 PHY 的供应商掌控。 0：无会话请求；1：会话请求。 注意：仅在设备模式下有效。
0	SRQSCS	**SRQSCS**：会话请求成功(Session request success)控制器在会话请求成功后设置此位。 0：会话请求失败； 1：会话请求成功。 注意：仅在设备模式下有效。

OTG_FS 中断寄存器(OTG_FS_GOTGINT)

偏移地址：0x04

复位值：0x0000 0000

应用程序在发生 OTG 中断时读此寄存器，并通过清除相应位来清除 OTG 中断标志。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												DBCDNE	ADTOCHG	HNGDET	保留							HNSSCHG	SRSSCHG	保留					SEDET	保留	
												rc_w1	rc_w1	rc_w1								rc_w1	rc_w1						rc_w1		

位	符号	说明
31:20	Reserved	保留位，硬件强制为 0
19	DBCDNE	**DBCDNE**：反射完成(Debounce done) 在设备连接线路反射结束后，控制器会设置此位。应用程序可以在检测到此中断后发起 USB 复位信号。此位仅在控制器 USB 配置寄存器的 HNP 有效或者 SRP 有效位(OTG_FS_GUSBCFG 寄存器的 HNPCAP 位或 SRPCAP 位)置位时才有效。 注意：仅在主机模式下有效。
18	ADTOCHG	**ADTOCHG**：A 类设备超时(A-device timeout change)控制器在 A 类设备等待 B 类设备插入超时时设置此位。 注意：在主机模式和设备模式下都有效。
17	HNGDET	**HNGDET**：主机协商检测(Host negotiation detected)控制器在检测到 USB 总线上的主机协商请求时设置此位。 注意：在主机模式和设备模式下都有效。
16:10	Reserved	保留
9	HNSSCHG	**HNSSCHG**：主机协商成功状态改变(Host negotiation success status change)

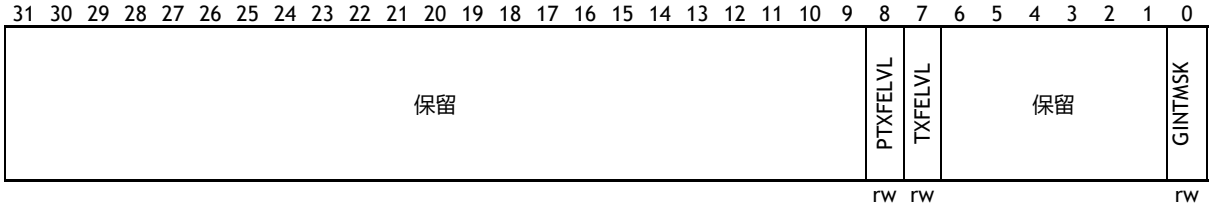
		控制器在 USB 主机协商请求成功或者失败时设置此位。应用程序需要通过读 OTG 控制和状态寄存器的主机协商成功位(OTG_FS_GOTGCTL 寄存器的 HNGSCS 位)来获得成功或者失败的信息。 注意：在主机模式和设备模式下都有效。
8	**SRSSCHG**	**SRSSCHG**：会话请求成功状态改变(Session request success status change) 控制器在会话请求成功或者失败时设置此位。应用程序需要通过读 OTG 控制和状态寄存器的会话请求成功位(OTG_FS_GOTGCTL 寄存器的 SRQSCS 位)来获得成功或者失败的信息。 注意：在主机模式和设备模式下都有效。
7:3	**Reserved**	保留
2	**SEDET**	**SEDET**：检测到会话结束(Session end detected) 控制器在检测到 VBUS<0.8V 时，设置此位，指示 B 类设备的 VBUS 线的电平无效。
1:0	**Reserved**	保留

OTG_FSAHB 配置寄存器(OTG_FS_GAHBCFG)

偏移地址：0x008

复位值：0x0000 0000

此寄存器用于在上电或改变控制器模式时配置控制器。此寄存器主要配置一些和 AHB 相关的参数。在初始化配置完成后，不要再修改此寄存器。应用程序在开始向 AHB 或 USB 传送数据前必需先配置好此寄存器。



位	符号	说明
31:9	**Reserved**	保留位，硬件强制为 0
8	**PTXFELVL**	**PTXFELVL**：周期性发送 FIFO 空级别(Periodic TxFIFO empty level) 指示在何种情况下需要触发控制器中断寄存器的周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 PTXFE 位)： 0：PTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示周期性发送 FIFO 已经半空； 1：PTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示周期性发送 FIFO 已经全空。 注意：仅在主机模式下有效。
7	**TXFELVL**	**TXFELVL**：发送 FIFO 空级别(TxFIFO empty level) 在设备模式下，此位指示在何种情况下需要触发 IN 端点发送 FIFO 空中断(OTG_FS_DIEPINTx 寄存器的 TXFE 位)： 0：TXFE(在 OTG_FS_DIEPINTx 寄存器中)中断指示 IN 端点 TXFIFO 已经半空； 1：TXFE(在 OTG_FS_DIEPINTx 寄存器中)中断指示 IN 端点 TXFIFO 已经全空。 在主机模式下，此位指示在何种情况下需要触发非周期性发送 FIFO 空中断(OTG_FS_GINTSTS 寄存器的 NPTXFE 位)： 0：NPTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示非周期性发送 FIFO 已经半空； 1：NPTXFE(在 OTG_FS_GINTSTS 寄存器中)中断表示非周期性发送 FIFO 已经全空。
6:1	**Reserved**	保留
0	**GINTMSK**	**GINTMSK**：全局中断屏蔽(Global interrupt mask) 应用程序可以通过此位选择屏蔽或者不屏蔽中断。但无论是否设置此位，控制器仍然会更新中断状态寄存器。

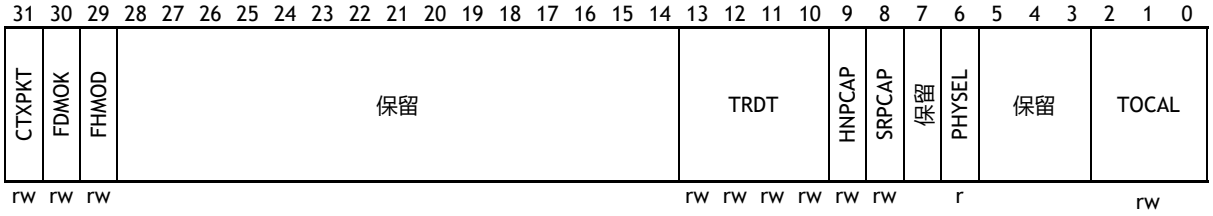
		0：屏蔽中断； 1：不屏蔽中断。 注意：在设备模式和主机模式下都有效。

OTG_FS_USB 配置寄存器(OTG_FS_GUSBCFG)

偏移地址：0x00C

复位值：0x0000 0A00

此寄存器用于在上电或模式切换时配置控制器。此寄存器多用于配置与 USB 和 USBPHY 相关的参数。应用程序在向 AHB 或 USB 传送数据之前必需先配置好此寄存器。在初始化配置完成后不要修改此寄存器。



位	符号	说明
31	CTXPKT	**CTXPKT**：错误的发送包(Corrupt Txp acket)此位仅用于调试。不能设置此位为'1'。 注意：在设备模式和主机模式下都有效。
30	FDMOD	**FDMOD**：强制设备模式(Force device mode) 不管 OTG_FS_ID 输入引脚的状态如何，设置此位为'1'将强制控制器进入设备模式。 0：普通模式； 1：强制设备模式。 设置此位后，应用程序需要等待至少 25ms，以便设置生效。 注意：在主机模式和设备模式下都有效。
29	FHMOD	**FHMOD**：强制主机模式(Forcehostmode) 不管 OTG_FS_ID 输入引脚的状态如何，设置此位为'1'将强制控制器进入主机模式。 0：普通模式 1：强制主机模式 设置此位后，应用程序需要等待至少 25ms，以便设置生效。 注意：在主机模式和设备模式下都有效。
28:14	Reserved	保留
13:10	TRDT	**TRDT**：USB 总线的周转时间(USB turnaround time) 以 PHY 层的时钟周期为单位设置周转时间。 设置一个 MAC 请求到包 FIFO 控制器(PFC)要求从 DFIFO(SPRAM)获取数据的响应时间。这些位必需被设置为： 0101：当 MAC 接口是 16 位 UTMIFS； 1001：当 MAC 接口是 8 位 UTMIFS。 注意：仅在设备模式下有效。
9	HNPCAP	**HNPCAP**：HNP 使能(HNP-capable) 应用程序通过此位来使能 OTG_FS 控制器的 HNP 功能。 0：禁止 HNP 功能； 1：使能 HNP 功能。 注意：在设备模式和主机模式下都有效。
8	SRPCAP	**SRPCAP**：SRP 使能(SRP-capable)

		应用程序通过此位来使能 OTG_FS 控制器的 SRP 功能。如果控制器工作在无 SRP 功能的 B 类设备模式下，就不能要求连接的 A 类设备(主机)使能 VBUS 线并发起一个会话。 0：禁止 SRP 功能； 1：使能 SRP 功能。 注意：在设备模式和主机模式下都有效。
7:3	**Reserved**	保留
2:0	**TOCAL**	**TOCAL**：FS 超时校准(FS timeout calibration) 考虑到由 PHY 带来的额外的时延，应用程序可以调整控制器的全速包超时判断时延，这些位以 PHY 时钟的数目给出了调整时延的长度。不同的 PHY 在控制总线状态时，带来的额外时延是不同的，通过这些位可以适应不同的总线时延。 USB 标准规定的全速包的超时判断是 16 到 18 个 BIT 时间。应用程序需要根据枚举的速度来配置此位。每个 PHY 时钟增加的 BIT 时间是 0.25 个 BIT 时间。

表 155 TRDT 值

AHB 频率范围(MHz)		**TRDT 最小值**
最小	**最大**	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6

OTG_FS 复位寄存器(OTG_FS_GRSTCTL)

偏移地址：0x10

复位值：0x2000 0000

应用程序可以通过此寄存器来复位控制器的各硬件模块。

31	30 29 28 27 26 25 24 23 22 21 20	19 18 17 16 15 14 13 12 11	10 9 8 7 6	5	4	3	2	1	0
AHBIDL	保留		TXFNUM	TXFFLSH	RXFFLSH	保留	FCRST	HSRST	CSRST
r			rw rw rw rw rw	rs	rs	res	rs	rs	rs

位	**符号**	**说明**
31	**AHBIDL**	**AHBIDL**：AHB 主控模块空闲(AHB master idle)指示 AHB 主控模块是否在空闲状态。 注意：在设备模式和主机模式下都有效。
30:11	**Reserved**	保留
10:6	**TXFNUM**	**TXFNUM**：发送 FIFO 编号(TxFIFO number) 此位指示哪个 FIFO 需要通过发送 FIFO 刷新位来刷新。只有在控制器清除了发送 FIFO 刷新位时，应用程序才能修改此位。 l00000： 主机模式下表示非周期性发送 FIFO 设备模式下表示发送 FIFO0 l00001： 主机模式下表示周期性发送 FIFO 设备模式下表示发送 FIFO1 l00010： 设备模式下表示发送 FIFO2 …… l00101：

		设备模式下表示发送 FIFO5 l10000： 在主机模式或设备模式下刷新所有发送 FIFO 注意：在主机模式和设备模式下都有效。
5	**TXFFLSH**	**TXFFLSH**：发送 FIFO 刷新(TxFIFO flush) 此位用于刷新单独的或所有的发送 FIFO，当控制器正在进行传输时，不能设置此位。 应用程序只有在检测了控制器既没有向发送 FIFO 写数据也没有从发送 FIFO 读数据时，才能设置此位。可以通过以下方式来检测： 读：NAK 有效中断，确保控制器没有从 FIFO 读数据。 写：OTG_FS_GRSTCTL 寄存器的 AHBIDL 位，确保控制器没有在写 FIFO。 注意：在设备模式和主机模式下都有效。
4	**RXFFLSH**	**RXFFLSH**：接收 FIFO 刷新(RxFIFO flush) 应用程序可以通过此位来刷新整个接收 FIFO，但必需确保控制器没有在传输数据。 应用程序只有在检测了控制器既没有写接收 FIFO 也没有读接收 FIFO 时，才能设置此位。 应用程序只有在等待此位重新被控制器清除后，才能进行其他的操作。此位需要 8 个时钟周期(以 PHY 或 AHB 时钟中较慢的那个为准)才能清除。 注意：在主机模式和设备模式下都有效。
3	**Reserved**	保留
2	**FCRST**	**FCRST**：主机帧计数器复位(Host frame counter reset) 应用程序通过写此位来复位控制器内部的帧计数器。在帧计数器复位后，控制器发送的首个 SOF 的帧号为 0。 注意：仅在主机模式下有效。
1	**HSRST**	**HSRST**：HCLK 软件复位(HCLK soft reset) 应用程序可以使用此位来刷新 AHB 时钟域的控制逻辑。仅由 AHB 时钟域驱动的模块进行复位。FIFO 不会受此位影响。 根据协议，所有的由 AHB 时钟域驱动的状态机都会在完成 AHB 当前传输后复位到空闲状态。清零 AHB 时钟域驱动的状态机中的 CSR 控制位。 为了清除此中断，用于控制中断状态及由 AHB 时钟域驱动的状态机产生的状态屏蔽位将被清除。 由于中断状态位并没有被清除，因此应用程序仍然可以获得在设置此位后产生的任何控制器事件的状态。 控制器会自动在所有必要的逻辑模块复位后清除此位。这一操作将维持若干个时钟周期，具体由控制器当前所处的状态决定。 注意：在设备模式和主机模式下都有效。
0	**CSRST**	**CSRST**：控制器软件复位(Core soft reset)复位 HCLK 和 PCLK 驱动域： 清除除了以下位之外的所有中断和 CSR 寄存器：OTG_FS_PCGCCTL 寄存器的 RSTPDMODL 位 OTG_FS_PCGCCTL 寄存器的 GAYEHCLK 位 OTG_FS_PCGCCTL 寄存器的 PWRCLMP 位 OTG_FS_PCGCCTL 寄存器的 STPPCLK 位 OTG_FS_HCFG 寄存器的 FSLSPCS 位 OTG_FS_DCFG 寄存器的 DSPD 位 所有模块的状态机(除了 AHB 从单元)都将复位到空闲状态，所有的发送 FIFO 和接收 FIFO 都被刷新。 所有在 AHB 主模块上的数据传输都在完成 AHB 传输的最后一个数据阶段后立即结束。所有在 USB 上的传输都立即结束。 应用程序可以在任何需要复位控制器的时侯设置此位。控制器将在所有必需的逻辑模块复位之后自动清除此位，这段操作将维持若干个时钟周期，具体由控制器当前所处的状态决定。一旦此位被清除，软件必需在激活 PHY 域之前等待至少 3

		个 PHY 时钟周期(同步操作的时延)。应用程序在开始其他操作之前必需确保本寄存器的位 31 为'1'(即 AHB 主控模块空闲)。 通常，只有在软件开发阶段和在上面列出的 USB 配置寄存器中动态修改 PHY 选择位时才需要进入软件复位操作。如果用户改变了 PHY 的配置，与之相应的时钟会被选中并应用到 PHY 域。一旦选中了新的时钟，PHY 域需要进行复位才能进一步的操作。 注意：在设备模式和主机模式下都有效。

OTG_FS 控制器中断寄存器(OTG_FS_GINTSTS)

偏移地址：0x014

复位值：0x0400 0020

此寄存器在发生与当前模式(设备模式或主机模式)相配的系统事件时打断应用程序。

寄存器的某些位仅在主机模式下有效，另一些位仅在设备模式下有效。寄存器也可以用来指示当前模式。应用程序需要在相应位写 1 来清除 rc_w1 类型的中断状态位。

FIFO 状态中断位是只读的，一旦应用程序在中断服务程序中对 FIFO 进行了读写操作，FIFO 的中断状态位会被自动清除。

初始化的时候，应用程序在使能某个中断位之前，需要先清除 OTG_FS_GINTSTS 寄存器的相应位，以避免由于原先的值导致不必要的中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	保留	PTXFE	HCINT	HPRTINT	保留		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	保留		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	保留		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1		r	r	r			rc_w1	rc_w1	r	r			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

位	符号	说明
31	WKUPINT	**WKUPINT**：检测到唤醒/远程唤醒中断(Resume/remote wakeup detected interrupt)在设备模式下，当检测到 USB 总线上的唤醒信号即产生此中断； 在主机模式下，在检测到 USB 总线上的一个远程唤醒信号时即产生此中断。 注意：在设备模式和主机模式下都有效。
30	SRQINT	**SRQINT**：会话请求/检测到新会话中断(Sessio nrequest/new session detected interrupt)在主机模式下，当检测到一个来自设备的会话请求时即产生此中断； 在设备模式下，当 VBUS 在 B 类设备的有效电平范围内时即产生此中断。 注意：在设备模式和主机模式下都有效。
29	DISCINT	**DISCINT**：检测到断开事件中断(Disconnect detected interrupt)当检测到设备断开时即产生此中断。 注意：仅在主机模式下有效。
28	CIDSCHG	**CIDSCHG**：连接上的 ID 线状态改变(Connector ID status change)控制器在检测到连接上的 ID 线状态改变即设置此位。 注意：在主机模式和设备模式下都有效
27	Reserved	保留
26	PTXFE	**PTXFE**：周期性发送 FIFO 空(Periodic TxFIFO empty) 当周期性发送 FIFO 半空或者全空，并且周期性请求队列里至少能写入一个请求时，产生此中断。半空还是全空的状态由控制器 AHB 配置寄存器的周期性发送 FIFO 空级别位决定(OTG_FS_GAHBCFG 寄存器的 PTXFELVL 位)。 注意：仅在主机模式下有效。
25	HCINT	**HCINT**：主机通道中断(Host channels interrupt)

		控制器设置此位，指示有一个未处理的主机通道事件(在主机模式下)。应用程序需要读主机所有通道中断寄存器(OTG_FS_HAINT 寄存器)来获得产生中断的通道号，然后读相应的主机通道 x 中断寄存器(OTG_FS_HCINTx 寄存器)获得中断的具体信息。应用程序需要通过清除 OTG_FS_HCINTx 寄存器的相应位来清除此位。 注意：仅在主机模式下有效。
24	**HPRTINT**	**HPRTINT**：主机端口中断(Host port interrupt) 控制器设置此位指示 OTG_FS 控制器的某个端口状态发生改变。应用程序需要通过读主机端口控制和状态寄存器(OTG_FS_HPRT)来获得导致此中断的具体信息。应用程序必需通过清除主机端口控制和状态寄存器的相应位来清除此位。 注意：仅在主机模式下有效。
23:22	**Reserved**	保留
21	**IPXFR**	**IPXFR**：未完成的周期性传输(Incomplete periodic transfer) 在主机模式下，控制器在帧结束时仍有属于当前帧的未完成的周期性传输时，产生此中断。 注意：仅在主机模式下有效。 INCOMPISOOUT：未完成的同步 OUT 传输(Incomplete isochronous OUT transfer) 在设备模式下，控制器在帧结束时仍有至少一个属于当前帧的未完成的同步 OUT 传输时，产生此中断。此中断随同本寄存器中的周期性帧结束中断(EOPF)产生。 注意：仅在设备模式下有效。
20	**IISOIXFR**	**IISOIXFR**：未完成的同步 IN 传输(Incomplete isochronous IN transfer) 在设备模式下，控制器在帧结束时仍有至少一个属于当前帧的未完成的同步 IN 传输时，产生此中断。此中断随同本寄存器中的周期性帧结束中断(EOPF)产生。 注意：仅在设备模式下有效。
19	**OEPINT**	**OEPINT**：OUT 端点中断(OUT end point interrupt) 在设备模式下，控制器设置此位指示有一个控制器未处理的 OUT 端点事件。应用程序需要读设备所有端点中断寄存器(OTG_FS_DAINT)来获得产生中断事件的端点号，然后读相应的设备 OUT 端点 x 中断寄存器(OTG_FS_DOEPINTx)来获得产生中断的具体信息。应用程序需要通过清除 OTG_FS_DOEPINTx 寄存器的相应位来清除此位。 注意：仅在设备模式下有效。
18	**IEPINT**	**IEPINT**：IN 端点中断(IN end point interrupt) 在设备模式下，控制器设置此位指示有一个控制器未处理的 IN 端点事件。应用程序需要读设备所有端点中断寄存器(OTG_FS_DAINT)来获得产生中断事件的端点号，然后读相应的设备 IN 端点 x 中断寄存器(OTG_FS_DIEPINTx)来获得产生中断的具体信息。应用程序需要通过清除 OTG_FS_DIEPINTx 寄存器的相应位来清除此位。 注意：仅在设备模式下有效。
17:16	**Reserved**	保留
15	**EOPF**	**EOPF**：周期性帧结束中断(End of periodic frame interrupt) 此中断指示在当前帧中，在设备配置寄存器中设置的周期性帧间隔时间(OTG_FS_DCFG 寄存器的 PFIVL 位)到达。 注意：仅在设备模式下有效。
14	**ISOODRP**	**ISOODRP**：同步 OUT 包丢失中断(Isochronous OUT packet dropped interrupt) 当接收 FIFO 没有足够空间为一个同步 OUT 端点存放一个最大长度的数据包时，将导致控制器写同步 OUT 包失败，并产生此中断。 注意：仅在设备模式下有效。
13	**ENUMDNE**	**ENUMDNE**：枚举完成(Enumeration done) 控制器设置此位指示已经获得速度枚举的信息。应用程序通过读设备状态寄存器(OTG_FS_DSTS)来获得进行枚举的速度信息。 注意：仅在设备模式下有效。
12	**USBRST**	**USBRST**：USB 复位(USB reset)

		控制器在检测到一个 USB 复位信号时设置此位。 注意：仅在设备模式下有效。
11	USBSUSP	**USBSUSP**：USB 挂起(USB suspend) 控制器在检测到 USB 线上的挂起信号时设置此位。控制器会在数据线超过 3ms 没有活动的情况下进入挂起状态。 注意：仅在设备模式下有效。
10	ESUSP	**ESUSP**：早期挂起(Early suspend) 控制器在检测到 USB 总线有 3ms 处于空闲状态时设置此位。 注意：仅在设备模式下有效。
9:8	Reserved	保留
7	GONAKEFF	**GONAKEFF**：全局 OUTNAK 状态(Global OUT NAK effective) 指示应用程序，设备控制寄存器的全局 OUTNAK 位(OTG_FS_DCTL 寄存器的 SGONAK 位)已生效。应用程序通过写设备控制寄存器的清除全局 OUTNAK 位(OTG_FS_DCTL 寄存器的 CGONAK 位)来清除此位。 注意：仅在设备模式下有效。
6	GINAKEFF	**GINAKEFF**：全局非周期性 INNAK 状态(Global IN non-periodic NAK effective) 指示应用程序，设备控制寄存器的设置全局非周期性 INNAK 位(OTG_FS_DCTL 寄存器的 SGINAK 位)已生效。也就是说，控制器已经相应了应用程序设置全局 INNAK 位的操作。应用程序通过清除设备控制寄存器的清除全局非周期性 INNAK 位(OTG_FS_DCTL 寄存器的 CGINAK 位)来清除此位。 此中断并不意味着向 USB 总线发送了 NAK 握手信号，STALL 位的优先级高于 NAK 位。 注意：仅在设备模式下有效。
5	NPTXFE	**NPTXFE**：非周期性发送 FIFO 空(Non-periodic TxFIFO empty) 当非周期性发送 FIFO 全空或者半空，并且非周期性请求队列里至少能写入一个请求时，控制器会产生此中断。FIFO 半空或全空的状态取决于控制器 AHB 配置寄存器的非周期性发送 FIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)。 注意：仅在主机模式下有效。
4	RXFLVL	**RXFLVL**：接收 FIFO 非空(RxFIFO non-empty)指示接收 FIFO 中至少有个一个数据包等待处理。 注意：在设备模式和主机模式下都有效。
3	SOF	**SOF**：帧首(Star to frame) 在主机模式下，控制器设置此位指示有一个 SOF(全速设备)或保持有效(低速设备)信号已发向 USB 总线。应用程序必需向此位写'1'来清除此中断。 在设备模式下，控制器设置此位表示在 USB 总线上检测到一个 SOF。应用程序需要读设备状态寄存器来获得帧编号。此中断仅当控制器处于全速状态时才会产生。 注意：在设备模式和主机模式下都有效。
2	OTGINT	**OTGINT**：OTG 中断(OTG interrupt) 控制器设置此位指示发生一个 OTG 协议事件。应用程序必需通过读 OTG 中断寄存器(OTG_FS_GOTGINT)来获得产生此中断的具体信息。应用程序必需通过清除 OTG_FS_GOTGINT 寄存器的相应位来清除此位。 注意：在设备模式和主机模式下都有效。
1	MMIS	**MMIS**：模式不匹配中断(Mode mismatch interrupt)控制器在应用程序试图执行以下操作时设置此位： 当控制器运行在设备模式下时，试图访问主机模式下的寄存器。当控制器运行在主机模式下时，试图访问设备模式下的寄存器。 操作寄存器时，在 AHB 端会收到正确的响应，但控制器内部会忽视此类操作，并不对控制器的操作产生任何影响。 注意：在主机模式和设备模式下都有效。
0	CMOD	**CMOD**：当前模式(Current mode of operation)指示当前模式。

		0：设备模式； 1：主机模式。 注意：在设备模式和主机模式下都有效。

OTG_FS 中断屏蔽寄存器(OTG_FS_GINTMSK)

偏移地址：0x018

复位值：0x0000 0000

此寄存器与控制器中断寄存器配合使用，产生中断。当一个中断位被屏蔽，就不会产生相应的中断，然而控制器中断寄存器(OTG_FS_GINTSTS)的相应位仍然会被置起。

31	30	29	28	27	26	25	24	23 22	21	20	19	18	17 16	15	14	13	12	11	10	9 8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	保留	PTXFEM	HCIM	PPTIM	保留	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	保留	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSM	ESUSPM	保留	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	保留
rw	rw	rw	rw		rw	rw	r		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

位	符号	说明
31	WUIM	**WUIM**：检测到唤醒/远程唤醒中断屏蔽(Resume/remote wakeup detected interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在设备模式和主机模式下都有效。
30	SRQIM	**SRQIM**：会话请求/检测到新会话中断屏蔽(Session request/new session detected interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在设备模式和主机模式下都有效。
29	DISCINT	**DISCINT**：检测到断开事件中断屏蔽(Disconnect detected interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在主机模式下有效。
28	CIDSCHGM	**CIDSCHGM**：连接上的 ID 线状态改变屏蔽(Connector ID status change mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在主机模式和设备模式下都有效
27	Reserved	保留
26	PTXFEM	**PTXFEM**：周期性发送 FIFO 空屏蔽(Periodic TxFIFO empty mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在主机模式下有效
25	HCIM	**HCIM**：主机通道中断屏蔽(Host channels interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在主机模式下有效。
24	PRTIM	**PRTIM**：主机端口中断屏蔽(Host port interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在主机模式下有效。
23:22	Reserved	保留
21	IPXFRM	**IPXFRM**：未完成的周期性传输中断屏蔽(Incomplete periodic transfer mask) 0：屏蔽该中断； 1：不屏蔽该中断。

		注意：仅在主机模式下有效。 IISOOXFRM：未完成的同步 OUT 传输中断屏蔽 (IncompleteisochronousOUTtransfermask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
20	**IISOIXFRM**	**IISOIXFRM**：未完成的同步 IN 传输中断屏蔽(Incomplete isochronous IN transfer mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
19	**OEPINT**	**OEPINT**：OUT 端点中断屏蔽(OUT end points interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
18	**IEPINT**	**IEPINT**：IN 端点中断屏蔽(IN end points interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
17:16	**Reserved**	保留
15	**EOPFM**	**EOPFM**：周期性帧结束中断屏蔽(End of periodic frame interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
14	**ISOODRPM**	**ISOODRPM**：同步 OUT 包丢失中断屏蔽(Isochronous OUT packet dropped interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
13	**ENUMDNEM**	**ENUMDNEM**：枚举完成中断屏蔽(Enumeration done mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
12	**USBRST**	**USBRST**：USB 复位中断屏蔽(USB reset mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
11	**USBSUSPM**	**USBSUSPM**：USB 挂起中断屏蔽(USB suspend mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
10	**ESUSPM**	**ESUSPM**：早期挂起中断屏蔽(Early suspend mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
9:8	**Reserved**	保留
7	**GONAKEFFM**	**GONAKEFFM**：全局 OUTNAK 状态中断屏蔽(Global OUT NAK effective mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
6	**GINAKEFFM**	**GINAKEFFM**：全局非周期性 INNAK 状态中断屏蔽(Global non-periodic IN NAK effective mask)

		0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在设备模式下有效。
5	NPTXFEM	**NPTXFEM**：非周期性发送 FIFO 空中断屏蔽(Non-periodic TxFIFO empty mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：仅在主机模式下有效。
4	RXFLVLM	**RXFLVLM**：接收 FIFO 非空中断屏蔽(Receive FIFO non-empty mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在设备模式和主机模式下都有效。
3	SOFM	**SOFM**：帧首中断屏蔽(Start of frame mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在设备模式和主机模式下都有效。
2	OTGINT	**OTGINT**：OTG 中断屏蔽(OTG interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在设备模式和主机模式下都有效。
1	MMISM	**MMISM**：模式不匹配中断屏蔽(Mode mismatch interrupt mask) 0：屏蔽该中断； 1：不屏蔽该中断。 注意：在主机模式和设备模式下都有效。
0	Reserved	保留

OTG_FS 接收状态调试读/OTG 状态读和 POP 寄存器(OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)

读操作的地址偏移：0x01C

POP 操作的地址偏移：0x020

复位值：0x0000 0000

对接收状态调试读寄存器的读操作，将返回接收 FIFO 中顶部的数据。对接收状态调试读寄存器和 POP 寄存器的读操作，将会把接收 FIFO 中顶部的数据项顶出。

在主机模式和设备模式下，对于接收状态数据的解释并不相同。如果接收 FIFO 为空，控制器将忽略对接收状态的读/POP 操作，并返回 0x0000 0000。应用程序只有在控制器中断寄存器的接收 FIFO 非空位(OTG_FS_GINTSTS 寄存器的 RXFLVL 位)为'1'时才能 POP 出接收状态 FIFO。

主机模式：

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

位	符号	说明
31:21	Reserved	保留
20:17	PKTSTS	**PKTSTS**：包状态(Packet status)指示接收到的数据包的状态 0010：接收到 IN 数据包； 0011：IN 传输完成(触发中断)； 0101：数据翻转位出错(触发中断)； 0111：通道中止(触发中断)； 其他：保留。
16:15	DPID	**DPID**：数据 PID(Data PID) 指示接收到的数据包的数据 PID

		00：DATA0； 10：DATA1； 01：DATA2； 11：MDATA。
14:4	**BCNT**	**BCNT**：字节数(Byte count)指示接收到的数据包的字节数。
3:0	**CHNUM**	**CHNUM**：通道号(Channel number)指示当前收到的数据包属于哪个通道

设备模式：

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

位	符号	说明
31:25	**Reserved**	保留
24:21	**FRMNUM**	**FRMNUM**：帧编号(Frame number) 此 4 位是从 USB 接收到的数据包所属的帧号的末 4 位，仅在同步 OUT 传输时有效。
20:17	**PKTSTS**	**PKTSTS**：包状态(Packet status)指示接收到的数据包的状态 0001：全局的 OUTNAK(触发中断)； 0010：接收到 OUT 数据包； 0011：OUT 传输完成(触发中断)； 0100：SETUP 传输完成(触发中断)； 0110：接收到 SETUP 数据包； 其他：保留。
16:15	**DPID**	**DPID**：数据 PID(Data PID)指示接收到 OUT 数据包的 PID 00：DATA0； 10：DATA1； 01：DATA2； 11：MDATA。
14:4	**BCNT**	**BCNT**：字节数(Byte count)指示接收到的数据包的字节数。
3:0	**EPNUM**	**EPNUM**：端点号 指示当前接收到的数据包所属的的端点号

OTG_FS 接收 FIFO 长度寄存器(OTG_FS_GRXFSIZ)

偏移地址：0x024

复位值：0x0000 0200

应用程序可以定义分配给接收 FIFO 的 RAM 长度。

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

位	符号	说明
31:16	**Reserved**	保留
15:0	**RXFD**	**RXFD**：接收 FIFO 的深度(RxFIFO depth)这个数值的单位是 32 位的字 最小值为 16 最大值为 256 上电复位的值是接收 FIFO 的最大深度值。

OTG_FS 非周期性 TXFIFO 大小寄存器

(OTG_FS_HNPTXFSIZ)/端点 0 传输 FIFO 大小 (OTG_FS_DIEPTXF0)

地址偏移：0x028

复位值：0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD/TX0FD																NPTXFSA/TX0FSA															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

主机模式

位	符号	说明
31:16	NPTXFD	**NPTXFD**：非周期性发送 FIFO 深度(Non-periodic TxFIFO depth)这个数值的单位是 32 位的字 最小值是 16 最大值是 256
15:0	NPTXFSA	**NPTXFSA**：非周期性接收 FIFO 在 RAM 里的起始地址(Non-periodic transmit RAM start address) 这些位的值表示非周期性接收 FIFO 在 RAM 里的起始地址

设备模式

位	符号	说明
31:16	TX0FD	TX0FD：端点 0 发送 FIFO 深度(Endpoint 0 TxFIFO depth)这个数值的单位是 32 位的字 最小值是 16 最大值是 256
15:0	TX0FSA	TX0FSA：端点 0 传输 FIFO 在 RAM 里的起始地址(Endpoint 0 TxFIFO depth) 这些位的值表示非周期性接收 FIFO 在 RAM 里的起始地址

OTG_FS 非周期性 TXFIFO/请求队列状态寄存器(OTG_FS_HNPTXSTS)

偏移地址：0x02C

复位值：0x0008 0200

注意：在设备模式下，此寄存器无效。

此寄存器为只读寄存器，储存非周期性发送 FIFO 和非周期性传输请求队列的剩余空间信息。

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

位	符号	说明
31	Reserved	保留
30:24	NPTXQTOP	NPTXQTOP：非周期性传输请求队列的顶部(Top of the non-periodic transmit request queue)正在由 MAC 模块处理的非周期性发送请求： 位[30:27]：通道/端点号；位[26:25]： 00：IN/OUT 命令； 01：0 长度的传输包(设备 IN/主机 OUT)；11：通道中止命令； 位 24：结束(选中通道/端点的最后一个请求)。
23:16	NPTQXSAV	NPTQXSAV：非周期性传输请求队列的剩余空间(Non-periodic transmit request queue space available) 指示非周期性传输请求队列的剩余空间。在主机模式下，此队列既保存 IN 的传输请求又保存 OUT 的传输请求，在设备模式下，仅保存 IN 的传输请求。 00：非周期性传输请求队列满；01：剩余 d×1 个请求空间； 02：剩余 d×2 个请求空间； B×n：剩余 d×n 个请求空间(0≤n≤d×8)；其他：保留。
15:0	NPTXFSAV	NPTXFSAV：非周期性发送 FIFO 的剩余空间(Non-periodic TxFIFO space available)指示非周期性发送 FIFO 的剩余空间，此值为 32 位。 00：非周期性发送 FIFO 满；01：剩余 d×1 个字的空间；02：剩余 d×2 个字的空间； O×n：剩余 d×n 个字的空间(0≤n≤d×256)；其他：保留。

OTG_FS 通用控制器配置寄存器(OTG_FS_GCCFG)

偏移地址：0x038

复位值：0x0000 XXXX

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

位	符号	说明
31:21	Reserved	保留
20	SOFOUTEN	**SOFOUTEN**：SOF 输出使能(SOF output enable) 0：不输出 SOF 脉冲； 1：输出 SOF 脉冲到引脚上。
19	VBUSBSEN	**VBUSBSEN**：VBUS 对于 B 类有效电平的监控使能(Enable the VBUS sensing “B” device) 0：VBUS 不监控 B 类有效电平； 1：VBUS 监控 B 类有效电平。
18	VBUSASEN	**VBUSASEN**：VBUS 对于 A 类有效电平的监控使能(Enable the VBUS sensing “A” device) 0：VBUS 不监控 A 类有效电平； 1：VBUS 监控 A 类有效电平。
17	Reserved	保留
16	PWRDWN	**PWRDWN**：掉电(Power down)用于在发送和接收时激活收发器 0：使能掉电； 1：禁止掉电(收发器激活)。
15:0	Reserved	保留

OTG_FS 控制器 ID 寄存器(OTG_FS_CID)

偏移地址：0x03C

复位值：0x0000 1200

此寄存器只读，保存产品的 ID。

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

位	符号	说明
31:0	PRODUCT_ID	PRODUCT_ID：产品 ID(Product ID field)应用程序可以编写此 ID 位。

OTG_FS 主机周期性发送 FIFO 长度寄存器(OTG_FS_HPTXFSIZ)

偏移地址：0x100

复位值：0x0200 0400

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

位	符号	说明

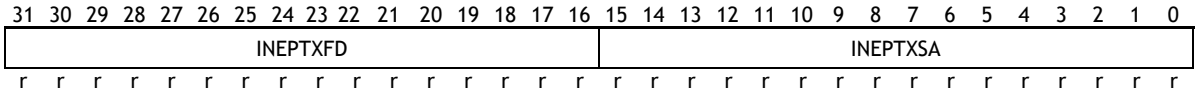
31:16	PTXFSIZ	**PTXFSIZ**：主机周期性发送 FIFO 深度(Host periodic TxFIFO depth) 此值为 32 位的字最小值为 16 最大值为 512
15:0	PTXSA	**PTXSA**：主机周期性发送 FIFO 起始地址(Host periodic TxFIFO start address)此寄存器的复位值是最大接收 FIFO 深度和最大非周期发送 FIFO 深度之和。

OTG_FS 设备 IN 端点发送 FIFO 长度寄存器(OTG_FS_DIEPTXFx)

(其中 x 是 FIFO 的编号，x=1…3)

偏移地址：0x104+(x – 1)×0x04

复位值：0x0200 0200



位	符号	说明
31:16	INEPTXFD	**INEPTXFD**：IN 端点发送 FIFO 的深度(IN endpoint TxFIFO depth) 此值为 32 位的字最小值为 16 最大值为 512 复位值是最大可能的 IN 端点发送 FIFO 的深度
15:0	INEPTXSA	**INEPTXSA**：IN 端点发送 FIFO 在 RAM 中的起始地址(IN endpoint FIFOx transmit RAM start address) 此值为 IN 端点发送 FIFO 在 RAM 中的起始地址。

26.16.3 主机模式下的寄存器

如果没有特殊说明，寄存器中的值都以二进制形式表达。

主机模式下的寄存器仅在主机模式下生效，不能在设备模式下访问，非法访问的结果是未定义的。

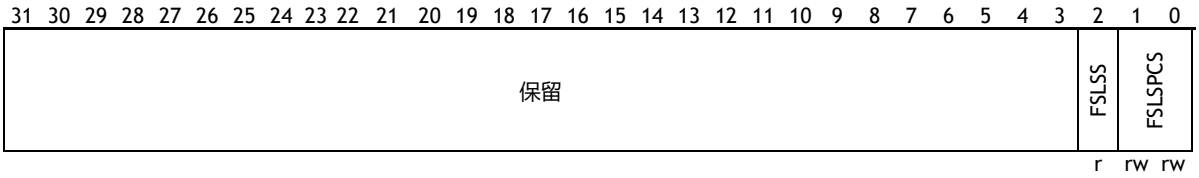
主机模式下的寄存器如下：

OTG_FS 主机模式配置寄存器(OTG_FS_HCFG)

偏移地址：0x400

复位值：0x0000 0000

此寄存器在上电后配置控制器的操作，不要在初始化后再修改此寄存器。



位	符号	说明
31:3	Reserved	保留
2	FSLSS	**FSLSS**：支持全速和低速设备(FS-and LS-only support) 应用程序通过此位来配置控制器对设备进行枚举的速度。应用程序可以通过此位使控制器用全速模式来枚举一个支持高速通信的设备。在初始化后不要修改此值。 1：仅支持全速/低速设备，即使插入的设备支持高速通信(只读)。
1:0	FSLSPCS	**FSLSPCS**：全速/低速 PHY 时钟选择(FS/LS PHY clock select)当控制器处于全速主机模式时： 01：PHY 时钟运行在 48MHz；其他值：保留。 当控制器处于低速主机模式时：00：保留； 01：PHY 时钟运行在 48MHz；

		10：PHY 时钟运行在 6MHz，根据 USB1.1 全速模式定义，当 UTMIFSPHY 低功耗模式选中时，如果 PHY 支持 6MHz 时钟，就在低速模式下使用 6MHz 时钟。用户一旦在低速模式下选中了 6MHz 时钟，需要执行一个软件复位操作； 11：保留。

OTG_FS 主机帧间隔寄存器(OTG_FS_HFIR)

偏移地址：0x404

复位值：0x0000 EA60

此寄存器为 OTG_FS 控制器在枚举时选中的速度设置帧间隔时间。

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

位	符号	说明
31:16	**Reserved**	保留
15:0	**FRIVL**	**FRIVL**：帧间隔(Frame interval) 应用程序通过此位来配置两个连续的 SOF(全速)或保持有效(低速)之间的时间间隔。此寄存器用 PHY 时钟个数来表示帧间隔。应用程序只有在设置了主机端口控制和状态寄存器的端口使能位(OTG_FS_HPRT 寄存器的 PENA 位)之后才能设置此寄存器。如果没有设置此寄存器，控制器将按照主机配置寄存器的全速/低速 PHY 时钟选择位(OTG_FS_HCFG 寄存器的 FSLSPCS 位)定义的 PHY 时钟来计算值。不要在初始化后再修改此寄存器。1ms×(全速/低速的 PHY 时钟频率)

OTG_FS 主机帧号/帧时间剩余寄存器(OTG_FS_HFNUM)

偏移地址：0x408

复位值：0x0000 3FFF

此寄存器指示了当前帧号，同样也指示了当前帧还剩余多少时间(用 PHY 时钟个数来表示)。

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

位	符号	说明
31:16	**FTREM**	**FTREM**：帧时间剩余(Frame time remaining) 指示当前帧还剩余多少时间，用 PHY 时钟数表达。每个 PHY 时钟，此值都会自减 1。当此值自减为 0，定义在帧间隔寄存器中的数值将自动载入此寄存器，并向 USB 总线发送一个新的 SOF。
15:0	**FRNUM**	**FRNUM**：帧号(Frame number) 每向 USB 总线发送一个 SOF 信号，此域自动加 1。达到 0x3FFF 时则自动归零，重新开始累加。

OTG_FS 主机周期性发送 FIFO/请求队列寄存器(OTG_FS_HPTXSTS)

偏移地址：0x410

复位值：0x0008 0100

此寄存器为只读寄存器，保存周期性发送 FIFO 和请求队列的剩余空间信息。

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

位	符号	说明
31:24	PTXQTOP	**PTXQTOP**：周期性传输请求队列顶部(Top of the periodic transmit request queue) 指示 MAC 模块正在处理的周期性发送请求项。此寄存器仅用于模块调试。 位 31：奇数/偶数帧 0：偶数帧发送； 1：奇数帧发送。 位[30:27]：通道/端点号；位[26:25]：类型 00：IN/OUT； 01：零长度数据包；11：中止通道命令。 位 24：结束(选中通道/端点的最后一个请求)。
23:16	PTXQSAV	**PTXQSAV**：周期性传输请求队列的剩余空间(Periodic transmit request queue spacea vailable)指示周期性传输请求队列的剩余空间，此请求队列包括 IN 和 OUT 的请求。 00：周期性传输请求队列满；01：剩余 d×1 个请求位置；10：剩余 d×2 个请求位置； b×n：剩余 d×n 个请求位置(0≤d×n≤8)；其他：保留。
15:0	PTXFSAVL	**PTXFSAVL**：周期性发送 FIFO 剩余空间(Periodic transmit data FIFO space available)指示周期性发送 FIFO 的剩余空间 此值为 32 位的字。 0000：周期性发送 FIFO 满； 0001：剩余 d×1 个字； 0010：剩余 d×2 个字； b×n：剩余 d×n 个字(0≤d×n≤d×512)； b×200：剩余 d×512 个字； 其他：保留。

OTG_FS 主机所有通道中断寄存器(OTG_FS_HAINT)

偏移地址：0x414

复位值：0x0000 0000

当某个通道产生了标志性的事件，主机所有通道中断会通过控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)打断应用程序。具体请参考图 282。每个通道都有相对应的通道中断位，一共有 16 个控制位。应用程序通过设置和清除相应的主机通道 x 中断寄存器的相应位来设置和清除此位。

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

位	符号	说明
31:16	Reserved	保留
15:0	HAINT	**HAINT**：通道中断(Channel interrupts) 每个位对应一个通道：位 0 对应通道 0，位 15 对应通道 15。

OTG_FS 主机所有通道中断屏蔽寄存器(OTG_FS_HAINTMSK)

偏移地址：0x418

复位值：0x0000 0000

主机所有通道中断屏蔽寄存器与主机所有通道中断寄存器配置使用，用于在产生事件时打断应用程序。每个通道都有一个相对应的中断屏蔽位，共有 16 位。

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

位	符号	说明
31:16	Reserved	保留
15:0	HAINTM	**HAINTM**：通道中断屏蔽(Channel interrupt mask) 0：屏蔽中断； 1：不屏蔽中断。 每个位对应一个通道：位 0 对应通道 0，位 15 对应通道 15。

OTG_FS 主机端口控制和状态寄存器(OTG_FS_HPRT)

偏移地址：0x440

复位值：0x0000 0000

此寄存器仅在主机模式下有效。一个 OTG 主机控制器仅支持一个端口。

此寄存器保存与主机端口相关的信息，包括每个端口的 USB 复位、使能、挂起、唤醒、连接状态和测试模式信息等。具体请参考图 282。标明 rc_w1 的位可以通过主机中断寄存器的主机端口中断位(OTG_FS_GINTSTS 寄存器的 HPRTINT 位)来触发中断，打断应用程序。在端口中断服务程序中，应用程序必需读此寄存器并清除导致中断的位。对于标明 rc_w1 的位，应用程序需要通过写'1'来清除中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留													PSPD		PTCTL				PPWR	PLSTS		保留	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
													r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

位	符号	说明
31:19	Reserved	保留
18:17	PSPD	**PSPD**：端口速度(Port speed)指示连上端口的设备速度 01：全速设备；10：低速设备；11：保留。
16:13	PTCTL	**PTCTL**：端口测试控制(Port test control) 应用程序通过写非零值到此位来进入测试模式，在端口会出现相应的信号。 0000：非测试模式；0001：Test_J 模式；0010：Test_K 模式； 0011：Test_SE0_NAK 模式；0100：Test_Packet 模式； 0101：Test_Force_Enable；其他：保留。
12	PPWR	**PPWR**：端口供电(Port power) 应用程序通过设置此位来向端口供电，控制器在发生电流溢出事件时清除此位。 0：不供电； 1：供电。
11:10	PLSTS	**PLSTS**：端口状态(Port line status)指示 USB 数据线的当前逻辑状态 位 10：OTG_FS_FS_DP 线的逻辑状态； 位 11：OTG_FS_FS_DM 线的逻辑状态。
9	Reserved	保留
8	PRST	**PRST**：端口复位(Port reset) 当应用程序设置此位时，会在端口上产生一个复位序列。应用程序需要控制复位的时间，并在复位完成后清除此位。 0：端口不复位； 1：端口复位。 应用程序需要保持此位的'1'状态至少 10ms 以便启动端口的复位序列。应用程序也可以在清除此位前增加 10ms 的'1'状态，USB 规范没有对复位信号的最长持续时间做出定义。
7	PSUSP	**PSUSP**：端口挂起(Port suspend)

		应用程序设置此位，使端口进入挂起状态。在这种状态下控制器仅停止发送 SOF 信号。应用程序需要通过设置端口时钟停止位来停止 PHY 时钟。 对此位的读操作将返回当前端口的挂起状态，当应用程序设置此寄存器的端口复位位或端口唤醒位，或控制器检测到一个远程唤醒信号，或控制器中断寄存器的唤醒/远程唤醒检测中断位或端口检测中断位(OTG_FS_GINTSTS 寄存器的 WKUINT 位或 DISCINT 位)被设置，控制器将清除此位。 0：端口不处于挂起模式；1：端口处于挂起模式。
6	**PRES**	**PRES**：端口唤醒(Port resume) 应用程序设置此位驱动端口发出唤醒信号。控制器将输出驱动唤醒信号直到应用程序清除此位。 当控制器检测到 USB 远程唤醒序列，按照控制器中断寄存器的端口唤醒/远程唤醒检测中断位的指示，控制器会自动输出唤醒序列而不需要应用程序干预。如果控制器检测到设备断开，会自动清除此位。对此位的读操作将返回控制器是否正在输出唤醒信号的信息。 0：无唤醒； 1：输出唤醒信号。
5	**POCCHNG**	**POCCHNG**：端口过流状态改变(Port overcurrent change)控制器在此寄存器的端口过流位(位 4)发生变化时设置此位。
4	**POCA**	**POCA**：端口过流位(Port overcurrent active)指示端口是否发生了过流。 0：没有过流； 1：过流。
3	**PENCHNG**	**PENCHNG**：端口使能/禁止状态改变(Port enable/disable change)控制器在本寄存器的端口使能位(位 2)发生变化时设置此位。
2	**PENA**	**PENA**：端口使能(Port enable) 端口只有在复位序列后才能被控制器使能，在发生过流、设备断开获控制器清除此位时禁止端口。应该程序不能通过寄存器写操作设置此位。此位不会触发中断。 0：端口禁止； 1：端口使能。
1	**PCDET**	**PCDET**：端口连接检测(Port connect detected) 当控制器检测到一个设备连接到端口时，会触发控制器中断寄存器的主机端口中断位(OTG_FS_GINTSTS 寄存器的 HPRTINT 位)来产生中断，并设置此位。应用程序必需通过写'1'来清除中断。
0	**PCSTS**	**PCSTS**：端口连接状态(Port connect status) 0：没有设备连接到端口； 1：有设备连接到端口。

OTG_FS 主机通道 x 特性寄存器(OTG_FS_HCCHARx)(此处 x 代码通道号，x=0...7)

偏移地址：0x500+0x20*x

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
CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位	符号	说明
31	**CHENA**	**CHENA**：通道使能(Channel enable) 此位由应用程序设置，由 FS_OTG 主机控制器清除。 0：通道禁止； 1：通道使能。

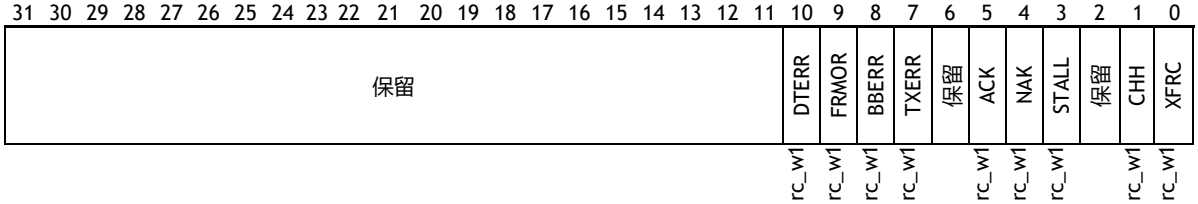
30	**CHDIS**	**CHDIS**：通道禁止(Channel disable) 应用程序通过设置此位，可以立即停止该通道上的数据收/发，即使该通道数据的传输还没有完成。应用程序只有在等到通道禁止中断产生时，才能认为该通道已被禁止。
29	**ODDFRM**	**ODDFRM**：奇数帧(Odd frame) 应用程序通过设置/清除此位来告知 OTG 主机控制器应该在奇数/偶数帧时进行传输。此位仅对周期性传输(同步或中断)有效。 0：偶数帧； 1：奇数帧。
28:22	**DAD**	**DAD**：设备地址(Device address) 应用程序通过此地址选择需要的设备。
21:20	**MCNT**	**MCNT**：设备地址(Multicount) 此域指示主机在这个周期性端点上每帧必须执行的传输数目。非周期性传输不使用这个参数。00：保留。设置这个数值将产生未定义的结果。 01：1 次传输。 10：在这个端点上，每帧需要产生 2 次传输。11：在这个端点上，每帧需要产生 3 次传输。 注：此域的数值至少应该设置为'01'。
19:18	**EPTYP**	**EPTYP**：端点类型指示选中的传输类型 00：控制传输； 01：同步传输； 10：块传输； 11：中断传输。
17	**LSDEV**	**LSDEV**：低速设备(Low speed device) 应用程序设置此位指示与之通信的设备是低速设备。
16	**Reserved**	保留
15	**EPDIR**	**EPDIR**：端点方向(Endpoint direction)指示传输是 IN 还是 OUT 方向。 0：OUT 1：IN
14:11	**EPNUM**	**EPNUM**：端点号(Endpoint number)指示与之通信的端点号。
10:0	**MPSIZ**	**MPSIZ**：最大包长度(Maximum packet size)指示选中端点的最大包长度。

OTG_FS 主机通道 x 中断寄存器(OTG_FS_HCINTx)(其中 x 代表通道号，x=0...7,)

偏移地址：0x508+(通道号×0x20)

复位值：0x0000 0000

此寄存器指示通道与 USB 和 AHB 相关时间的状态。具体请参考图 282。当控制器中断寄存器的主机通道中断位(OTG_FS_GINTSTS 寄存器的 HCINT 位)置起时，应用程序需要先读主机所有通道中断寄存器(OTG_FS_HAINT)，获得发生主机通道中断的通道号，再读取相应通道的中断寄存器，获得中断的详细信息。应用程序通过清除此寄存器的相应位，清除 OTG_FS_HAINT 寄存器和 OTG_FS_GINTSTS 寄存器的相应位。



位	符号	说明
31:11	**Reserved**	保留
10	**DTERR**	**DTERR**：数据 PID 错误(Data toggle error)

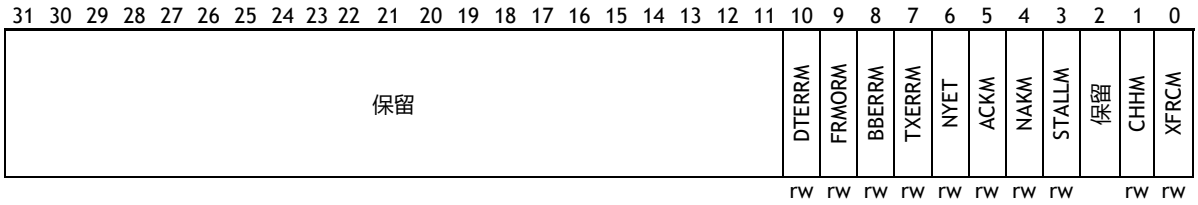
9	FRMOR	**FRMOR**：帧溢出(Frame overrun)
8	BBERR	**BBERR**：串扰错误(Babble error)
7	TXERR	**TXERR**：传输错误(Transaction error)指示发生了以下错误： CRC 校验失败超时 位填充错误 EOP 失败
6	Reserved	保留
5	ACK	**ACK**：ACK 已收到/已发送中断(ACK response received/transmitted interrupt)
4	NAK	**NAK**：NAK 已收到中断(NAK response received interrupt)
3	STALL	**STALL**：STALL 已收到中断(STALL response received interrupt)
2	Reserved	保留
1	CHH	**CHH**：通道中止(Channel halted) 指示由于 USB 传输错误，或者应用程序中止请求导致的传输异常结束。
0	XFRC	**XFRC**：传输完成(Transfer completed)传输正常完成，没有出错。

OTG_FS 主机通道 x 中断屏蔽寄存器(OTG_FS_HCINTMSKx)(其中 x 为通道号，x=0...7)

偏移地址：0x50C+(通道号×0x20)

复位值：0x0000 0000

本寄存器用于屏蔽上一节所描述的各类通道中断。



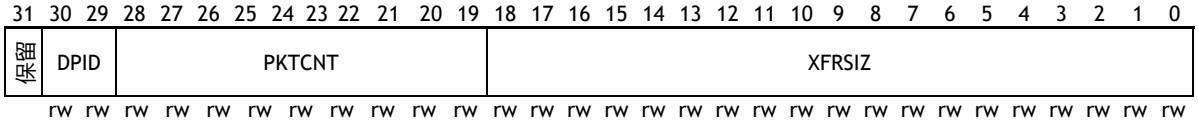
位	符号	说明
31:11	Reserved	保留
10	DTERRM	**DTERRM**：数据 PID 错误中断屏蔽(Data toggle error mask) 0：屏蔽中断； 1：不屏蔽中断。
9	FRMORM	**FRMORM**：帧溢出中断屏蔽(Frame overrun mask) 0：屏蔽中断； 1：不屏蔽中断。
8	BBERRM	**BBERRM**：串扰错误中断屏蔽(Babble error mask) 0：屏蔽中断； 1：不屏蔽中断。
7	TXERRM	**TXERRM**：传输错误中断屏蔽(Transaction error mask) 0：屏蔽中断； 1：不屏蔽中断。
6	NYET	**NYET**：收到响应中断屏蔽(response received interrupt mask) 0：屏蔽中断； 1：不屏蔽中断。
5	ACKM	**ACKM**：ACK 已收到/已发送中断屏蔽(ACK response received/transmitted interrupt mask) 0：屏蔽中断； 1：不屏蔽中断。
4	NAKM	**NAKM**：NAK 已收到中断屏蔽(NAK response received interrupt mask) 0：屏蔽中断； 1：不屏蔽中断。
3	STALML	**STALML**：STALL 已收到中断屏蔽(STALL responsereceived interrupt mask) 0：屏蔽中断；

		1：不屏蔽中断。
2		保留
1	CHHM	**CHHM**：通道中止中断屏蔽(Channel halted mask) 0：屏蔽中断； 1：不屏蔽中断。
0	XFRCM	**XFRCM**：传输完成中断屏蔽(Transfer completed mask) 0：屏蔽中断； 1：不屏蔽中断。

OTG_FS 主机通道 x 传输长度寄存器(OTG_FS_HCTSIZx)(其中 x 为通道号，x=0...7)

偏移地址：0x510+(通道号×0x20)

复位值：0x0000 0000



位	符号	说明
31	Reserved	保留
30:29	DPID	**DPID**：数据 PID(Data PID) 应用程序通过此位告知控制器首个传输所使用的 PID 类型。控制器将会自动控制后续传输的 PID 类型。 00：DATA0；01：DATA2；10：DATA1； 11：MDATA(非控制)/SETUP(控制)。
28:19	PKTCNT	**PKTCNT**：包数目(Packet count) 应用程序通过此位告知控制器期望收到(IN)的包数目或期望发送(OUT)的包数目。控制器在每次成功地发送或接收 OUT/IN 包之后，将自动地递减这个参数，一旦该域为 0，应用程序将收到中断，指示传输正常结束。
18:0	XFRSIZ	**XFRSIZ**：传输长度(Transfer size) 对于 OUT 传输，这些位指示主机在传输期间发送的数据字节数目。 对于 IN 传输，这些位指示应用程序预留的缓冲区长度。对于 IN 传输(周期性和非周期性)，应用程序需要按照最大数据包的整数倍来预定义这个参数。

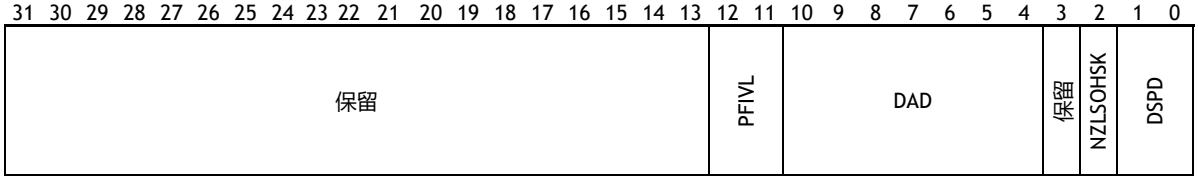
26.16.4 设备模式下的寄存器

OTG_FS 设备配置寄存器(OTG_FS_DCFG)

偏移地址：0x800

复位值：0x0220 0000

在复位、或特殊控制命令、或枚举后，此寄存器用于配置控制器在设备模式下的操作特性。在初始化后，不要再修改此寄存器。



位	符号	说明
31:13	Reserved	保留
12:11	PFIVL	**PFIVL**：周期性帧间隔(Periodic frame interval)

		指示产生周期性帧结束中断的时间占整个帧的百分比。应用程序可以通过此中断判断一个帧内的同步传输是否都已传输完成。 00：80%的帧时间； 01：85%的帧时间； 10：90%的帧时间； 11：95%的帧时间。
10:4	**DAD**	**DAD**：设备地址(Device address) 应用程序在收到 SetAddress 的控制命令后，按照参数填充此位。
3	**Reserved**	保留
2	**NZLSOHSK**	**NZLSOHSK**：非零长度的状态 OUT 握手信号(Non-zero-length status OUT handshake) 在控制传输的状态阶段，如果收到了一个非零长度的数据包，应用程序可以通过此位选择发送一个握手信号。 1：向非零长度的状态 OUT 传输发送 STALL 握手，并且不向应用程序传送收到的 OUT 包。 0：根据设备端口控制寄存器的 NAK 位和 STALL 位状态，选择发送握手信号，并将收到的 OUT 包传送给应用程序(零长度和非零长度的)。
1:0	**DSPD**	**DSPD**：设备速度(Device speed) 指示应用程序需要控制器进行枚举操作时的速度，或者是应用程序能支持的最大速度。然而，实际的总线速度只有在整个序列完成后，才能根据所连接的 USB 主机的速度决定。 00：保留； 01：保留； 10：保留； 11：全速(USB1.1 收发器，时钟为 48MHz)。

OTG_FS 设备控制寄存器(OTG_FS_DCTL)

偏移地址：0x804

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
保留																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	PWUSIG
																				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

位	符号	说明
31:12	**Reserved**	保留
11	**POPRGDNE**	**POPRGDNE**：上电配置结束(Power-on programming done) 应用程序通过此位指示从调电状态唤醒后，对寄存器的配置已经完成。
10	**CGONAK**	**CGONAK**：清除全局 OUTNAK(Clear global OUT NAK)对此位的写操作将清除全局 OUTNAK 状态。
9	**SGONAK**	**SGONAK**：设置全局 OUTNAK(Set global OUT NAK)对此位的写操作将设置全局 OUTNAK 状态。 应用程序通过此位告知控制器对所有的 OUT 端点都发送 NAK 信号。 程序在设置此位前，必需确认控制器中断寄存器的全局 OUT NAK 有效位(OTG-FS_GINTSTS 寄存器的 GONAKEFF 位)已被清除。
8	**CGINAK**	**CGINAK**：清除全局 INNAK((Clear global IN NAK))写此位将清除全局 INNAK 状态。
7	**SGINAK**	**SGINAK**：设置全局 INNAK(SetglobalINNAK) 对此位的写操作将设置全局非周期性 IN 的 NAK 状态。程序通过此位告知控制器对所有非周期性 IN 端点都发送 NAK 握手信号。

		程序在设置此位前，必需确认控制器中断寄存器的全局 IN NAK 有效位(OTG_FS_GINTSTS 寄存器的 GINAKEFF 位)已被清除。
6:4	TCTL	**TCTL**：测试控制(Test control) 000：测试模式禁止； 001：Test_J 模式； 010：Test_K 模式； 011：Test_SE0_NAK 模式； 100：Test_Packet 模式； 101：Test_Force_Enable； 其他：保留。
3	GONSTS	**GONSTS**：全局 OUTNAK 状态(Global OUT NAK status) 0：根据 FIFO 的状态和 NAK 位及 STALL 位的状态，发送握手信号。 1：不管存储区是否为空，都不写入数据到接收 FIFO。向除了 SETUP 外的所有传输都发送 NAK 握手信号，丢弃所有的同步 OUT 包。
2	GINSTS	**GINSTS**：全局 INNAK 状态(Global IN NAK status) 0：根据发送 FIFO 中数据的状态发送握手信号。 1：不管发送 FIFO 中数据的状态，对所有非周期性 IN 端点都发送 NAK 握手信号。
1	SDIS	**SDIS**：软件断开(Soft disconnect) 应用程序通过此位告知 OTG 控制器执行软件断开操作。当设置此位后，主机将看到设备已经断开，设备方将不会从 USB 总线上收到任何信号。控制器将保持在断开状态，直到程序清除此位。 0：普通操作。当此位在设备软件断开后清除，控制器将发送一个设备连接事件到主机，主机将重新执行枚举操作。 1：控制器执行设备软件断开操作。
0	RWUSIG	**RWUSIG**：远程唤醒信号(Remote wakeup signaling) 当应用程序设置此位，控制器将发送远程唤醒信号，唤醒 USB 主机。应用程序必需设置此位来使控制器退出挂起状态。根据 USB2.0 规范，程序必需在设置此位后的 1~15ms 之间再次清除它。

为了使 USB 主机能识别到设备断开的操作，软件断开(SDIS)位需要保持一段时间，下表列出了最小的持续时间(根据设备的不同状态)。为了适应时钟的抖动，建议应用程序在定义的最短时间外再额外保留一段时间的延迟。

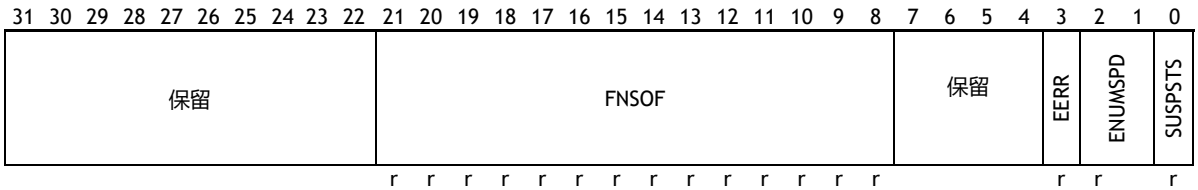
操作速度	设备状态	最短持续时间
全速	挂起	1ms+2.5us
全速	空闲	2.5us
全速	不空闲或挂起(正在执行传输操作)	2.5us

OTG_FS 设备状态寄存器(OTG_FS_DSTS)

偏移地址：0x808

复位值：0x0000 0010

此寄存器指示控制器与 USB 相关的状态。此寄存器用于在发生设备所有中断(OTG_FS_DAINT)寄存器事件时读取。



位	符号	说明
31:22	Reserved	保留

21:8	FNSOF	**FNSOF**：接收到的 SOF 的帧号(Frame number of the received SOF)
7:4	Reserved	保留
3	EERR	**EERR**：奇怪的错误(Erratic error) 控制器在发生一些奇怪的错误时设置此位。 如果发生了奇怪的错误，OTG_FS 控制器将进入挂起状态，并设置控制器中断寄存器的早期挂起位(OTG_FS_GINTSTS 寄存器的 ESUSP 位)随之产生中断。如果早期挂起中断是由于此位引起的，应用程序只能通过执行软件断开来解决。
2:1	ENUMSPD	**ENUMSPD**：枚举速度(Enumerated speed)指示 OTG_FS 控制器通过序列选定的执行速度。 01：保留；10：保留； 11：全速(PHY 时钟运行在 48MHz)；其他：保留。
0	SUSPSTS	**SUSPSTS**：挂起状态(Suspend status) 在设备模式下，如果在 USB 总线上检测到了挂起条件，将设置此位。控制器将在检测到 USB 数据线在 3ms 内没有活动的情况下进入挂起状态。控制器在以下条件时退出挂起模式： 当 USB 数据线上出现了活动 当应用程序设置设备控制寄存器的远程唤醒信号位(OTG_FS_DCTL 寄存器的 RWUSIG 位)

OTG_FS 设备 IN 端点通用中断屏蔽寄存器(OTG_FS_DIEPMSK)

偏移地址：0x810

复位值：0x0000 0000

此寄存器配合设备 IN 端点中断寄存器(OTG_FS_DIEPINTx)来产生 IN 端点中断。对应于 OTG_FS_DIEPINTx 寄存器的相应 IN 端点中断可以通过配置此寄存器来屏蔽。所有的中断在默认状态下都为屏蔽状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																		NAKM	保留						INEPNEM	INEPNMM	ITTXFEMSK	TOM	保留	EPDM	XFRCM
																		rw							rw	rw	rw	rw		rw	rw

位	符号	说明
31:14	Reserved	保留
13	NAKM	**NAKM:**NAK 中断掩码 0:屏蔽中断 1:解除掩码中断
12:7	Reserved	保留
6	INEPNEM	**INEPNEM**：IN 端点 NAK 状态有效中断屏蔽(IN endpoint NAK effective mask) 0：中断屏蔽； 1：中断不屏蔽。
5	INEPNMM	**INEPNMM**：端点收到 IN 命令不匹配中断屏蔽(IN token received with EP mismatch mask) 0：中断屏蔽； 1：中断不屏蔽。
4	ITTXFEMSK	**ITTXFEMSK**：当发送 FIFO 空时收到 IN 命令中断屏蔽(IN token received when TxFIFO empty mask) 0：中断屏蔽； 1：中断不屏蔽。
3	TOM	**TOM**：超时检测中断屏蔽(非同步端点)(Timeout condition mask(Non-isochronous endpoints))

		0：中断屏蔽； 1：中断不屏蔽。
2	**Reserved**	保留
1	**EPDM**	**EPDM**：端点被禁止中断屏蔽(Endpoint disabled interrupt mask) 0：中断屏蔽； 1：中断不屏蔽。
0	**XFRCM**	**XFRCM**：传输结束中断屏蔽(Transfer completed interrupt mask) 0：中断屏蔽； 1：中断不屏蔽。

OTG_FS 设备 OUT 端点通用中断屏蔽寄存器(OTG_FS_DOEPMSK)

偏移地址：0x814

复位值：0x0000 0000

此寄存器配合设备 OUT 端点中断寄存器(OTG_FS_DOEPINTx)使用，产生 OUT 端点中断。OTG_FS_DOEPINTx 寄存器的每一位都可以通过写此寄存器的相应位来屏蔽。在默认状态下，所有中断都屏蔽。

31:14	13	12	11:9	8	7:6	5	4	3	2	1	0
保留	NAK	BERRM	保留	OUTPKTERRM	保留	STSPHSRXM	OTEPDM	STUPM	保留	EPDM	XFRCM
	rw	rw		rw		rw	rw	rw		rw	rw

位	符号	说明
31:14	**Reserved**	保留
13	**NAKMSK**	**NAKMSK:**NAK 中断掩码 0：中断屏蔽； 1：中断不屏蔽。
12	**BERRM**	**BERRM:** 杂音错误 中断掩码 0：中断屏蔽； 1：中断不屏蔽。
11:9	**Reserved**	保留
8	**OUTPKTERRM**	**OUTPKTERRM**：输出数据包错误掩码 0：中断屏蔽； 1：中断不屏蔽。
7:6	**Reserved**	保留
5	**STSPHSRXM**	**STSPHSRXM:**控制写屏蔽器的状态相位已接收 0：中断屏蔽； 1：中断不屏蔽。
4	**OTEPDM**	**OTEPDM**：当端点被禁止时收到 OUT 命令中断屏蔽(OUT token received when endpoint disabled mask) 0：中断屏蔽； 1：中断不屏蔽。
3	**STUPM**	**STUPM**：SETUP 阶段完成中断屏蔽(SETUP phase done mask)仅对控制端点有效 0：中断屏蔽； 1：中断不屏蔽。
2	**Reserved**	保留
1	**EPDM**	**EPDM**：端点被禁止中断屏蔽(Endpoint disabled interrupt mask) 0：中断屏蔽； 1：中断不屏蔽。
0	**XFRCM**	**XFRCM**：传输结束中断屏蔽(Transfer completed interrupt mask)

		0：中断屏蔽； 1：中断不屏蔽。

OTG_FS 设备所有端点中断寄存器(OTG_FS_DAINT)

偏移地址：0x818

复位值：0x0000 0000

当每个端点发生了事件时，设备所有端点中断寄存器分别通过控制器中断寄存器的设备 OUT 端点中断位或设备 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 或 IEPINT 位)来产生中断打断应用程序。每个端点都有相对应的一位，OUT 端点有 16 位，IN 端点也有 16 位。对于双向端点，同时使用 IN 和 OUT 位。此寄存器的相应位，在应用程序设置和清除相应的设备端点 x 中断寄存器(OTG_FS_DIEPINTx 和 OTG_FS_DOEPINTx 寄存器)的相应位时自动设置和清除。

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

位	符号	说明
31:16	OEPINT	**OEPINT**：OUT 端点中断位(OUT endpoint interrupt bits)每个位对应一个 OUT 端点。 位 16 对应 OUT 端点 0，位 18 对应 OUT 端点 3。
15:0	IEPINT	**IEPINT**：IN 端点中断位(IN endpoint interrupt bits)每个位对应一个 IN 端点。 位 0 对应 IN 端点 0，位 3 对应 IN 端点 3。

OTG_FS 所有端点中断屏蔽寄存器(OTG_FS_DAINTMSK)

偏移地址：0x81C

复位值：0x0000 0000

设备所有端点中断屏蔽寄存器与设备端点中断寄存器配合使用，在发生设备端点事件时，产生中断打断应用程序。然而，设备所有端点中断寄存器(OTG_FS_DAINT)的相应位在屏蔽时仍然置位。

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

位	符号	说明
31:16	OEPM	**OEPM**：OUT 端点中断屏蔽寄存器(OUT EP interrupt mask bits)每个位对应一个 OUT 端点。 位 16 对应 OUT 端点 0，位 31 对应 OUT 端点 15。 0：屏蔽中断； 1：不屏蔽中断。
15:0	IEPM	**IEPM**：IN 端点中断屏蔽位(INEPinterruptmaskbits)每个位对应一个 IN 端点。 位 0 对应 IN 端点 0，位 15 对应 IN 端点 15。 0：屏蔽中断； 1：不屏蔽中断。

OTG_FS 设备 VBUS 放电时间寄存器(OTG_FS_DVBUSDIS)

偏移地址：0x828

复位值：0x0000 17D7

此寄存器定义 SRP 期间，在 VBUS 脉冲后的 VBUS 放电时间。

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

位	符号	说明

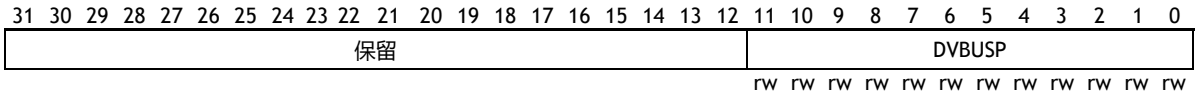
31:16	**Reserved**	保留
15:0	**VBUSDT**	**VBUSDT**：设备 VBUS 放电时间(Device VBUS discharge time) 定义在 SRP 期间，VBUS 脉冲后的 VBUS 放电时间。这个数值等于以 PHY 时钟计算的 VBUS 放电时间/1024。 根据不同的 VBUS 负载，这个数值可以适当调整。

OTG_FS 设备 VBUS 脉冲时间寄存器(OTG_FS_DVBUSPULSE)

偏移地址：0x82C

复位值：0x0000 05B8

此寄存器定义 SRP 期间 VBUS 脉冲的时间。



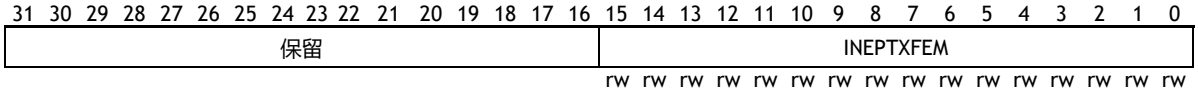
位	符号	说明
31:12	**Reserved**	保留
11:0	**DVBUSP**	**DVBUSP**：设备 VBUS 脉冲时间(Device VBUS pulsing time) 定义在 SRP 期间，VBUS 脉冲时间。这个数值等于以 PHY 时钟计算的 VBUS 脉冲时间/1024。

OTG_FS 设备 IN 端点 FIFO 空中断屏蔽寄存器(OTG_FS_DIEPEMPMSK)

偏移地址：0x834

复位值：0x0000 0000

此寄存器配合 IN 端点 FIFO 空中断寄存器(TXFE_OTG_FS_DIEPINTx)产生中断。



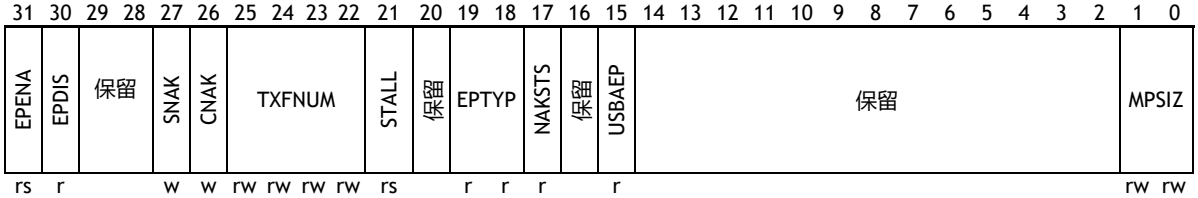
位	符号	说明
31:16	**Reserved**	保留
15:0	**INEPTXFEM**	**INEPTXFEM**：IN 端点发送 FIFO 空中断屏蔽位(IN EP TxFIFO empty interrupt mask bits)此位用于屏蔽 OTG_FS_DIEPINTx 寄存器的相应位。 TXFE 中断的一位对应一个 IN 端点：位 0 对应 IN 端点 0，位 3 对应 IN 端点 3。 0：屏蔽中断； 1：不屏蔽中断。

OTG_FS 设备控制 IN 端点 0 控制寄存器(OTG_FS_DIEPCTL0)

偏移地址：0x900

复位值：0x0000 0000

本节描述了设备控制 IN 端点 0 控制寄存器。非 0 控制端点使用对应端点 1-15 的寄存器。



位	符号	说明
31	**EPENA**	**EPENA**：端点使能(Endpoint enable) 应用程序设置此位，以启动在端点 0 上的数据传输。控制器在产生以下端点中断前会先清除此位：

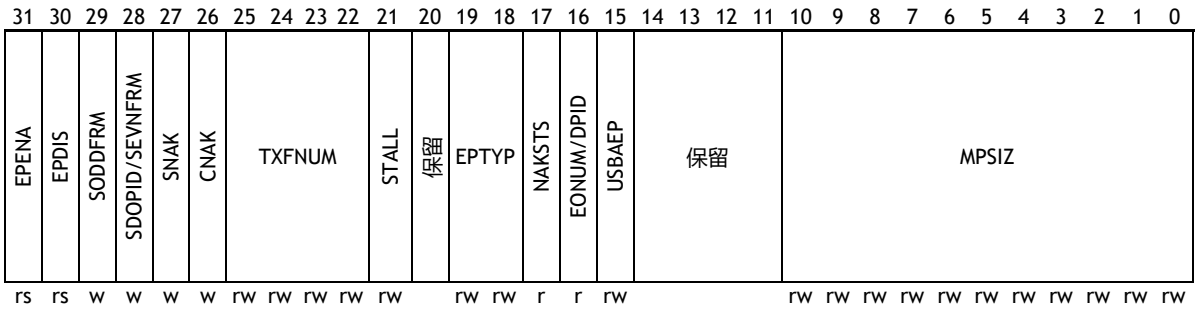
		端点禁止；传输完成。
30	**EPDIS**	**EPDIS**：端点禁止(Endpoint disable) 应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到端点禁止中断才能确为此端点已禁止。控制器在设置端点中断中断时会清除此位。应用程序只有在端点已经使能时才能设置此位。
29:28	**Reserved**	保留
27	**SNAK**	**SNAK**：设置 NAK(Set NAK)写此位会设置端点的 NAK 状态 通过设置此位，程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包时设置此位。
26	**CNAK**	**CNAK**：清除 NAK(ClearNAK)写此位会清除端点的 NAK 状态
25:22	**TXFNUM**	**TXFNUM**：发送 FIFO 的编号(TxFIFO number)此位设置分配给 IN 端点 0 的 FIFO 编号
21	**STALL**	**STALL**：STALL 握手(STALL handshake) 应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或全局 INNAK 位、或全局 OUTNAK 位，STALL 位仍有高优先级。
20	**Reserved**	保留
19:18	**EPTYP**	**EPTYP**：端点类型(Endpoint type)对于控制端点，由硬件置为'00'
17	**NAKSTS**	**NAKSTS**：NAK 状态(NAK status)指示以下： 0：根据 FIFO 状态，控制器发送非 NAK 握手信号； 1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置此位还是控制器设置此位，控制器都将停止数据传输，而不管发送 FIFO 中的数据是否有效。不管此位设置如何，控制器永远发送 ACK 握手响应 SETUP 数据包。
16	**Reserved**	保留
15	**USBAEP**	**USBAEP**：USB 活跃端点(USB active endpoint) 此位永远为'1'，指示控制端点 0 在任何配置情况下都是活跃的。
14:2	**Reserved**	保留
1:0	**MPSIZ**	**MPSIZ**：最大包长度(Maximum packet size) 应用程序通过此位配置该端点的最大数据包长度 00：64 字节； 01：32 字节； 10：16 字节； 11：8 字节。

OTG 设备端点 x 控制寄存器(OTG_FS_DIEPCTLx)(其中 x 为端点号，x=1…3)

偏移地址：0x900+(端点号×0x20)

复位值：0x0000 0000

应用程序通过这些寄存器控制非 0 端点的操作特性。



位	符号	说明
31	**EPENA**	**EPENA**：端点使能(End point enable)

		应用程序设置此位，以启动端点上的数据传输。控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成 端点禁止传输完成
30	**EPDIS**	**EPDIS**：端点禁止(End point disable) 应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到端点禁止中断才能确为此端点已禁止。控制器在设置端点中断中断时会清除此位。应用程序只有在端点已经使能时才能设置此位。
29	**SODDFRM**	**SODDFRM**：设置奇数帧(Set odd frame)仅对同步的 IN 和 OUT 端点有效。 写此位在 EONUM 中选择奇数帧。
28	**SD0PID**	**SD0PID**：设置 DATA0PID(Set DATA0 PID)仅对中断/块传输 IN 端点有效。 写此位将设置数据 PID 位为 DATA0。 SEVNFRM：设置偶数帧 仅对同步 IN 端点有效。 写此位在 EONUM 中选择偶数帧。
27	**SNAK**	**SNAK**：设置 NAK(Set NAK)写此位会设置端点的 NAK 状态 通过设置此位，程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或 OUT 端点传输结束中断时设置此位。
26	**CNAK**	**CNAK**：清除 NAK(Clear NAK)写此位会清除端点的 NAK 状态
25:22	**TXFNUM**	**TXFNUM**：发送 FIFO 的编号(TxFIFO number) 此位设置分配给该端点的 FIFO 编号，每个有效的 IN 端点都必需分配一个独立的 FIFO 编号。此位仅对 IN 端点有效。
21	**STALL**	**STALL**：STALL 握手(STALL handshake) 仅对非控制、非同步 IN 端点有效(操作类型为 rw) 应用程序设置此位，该端点会以 STALL 来响应所有的主机命令。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有最高优先级。只有应用程序能清除此位，控制器不能。 仅对控制端点有效(操作类型是 rs) 应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或全局 IN NAK 位、或全局 OUT NAK 位，STALL 位仍有高优先级。然而控制器将永远用 ACK 握手来响应 SETUP 数据包。
20	**Reserved**	保留
19:18	**EPTYP**	**EPTYP**：端点类型(Endpoint type)此位指示端点的传输类型 00：控制； 01：同步； 10：块传；11：中断；
17	**NAKSTS**	**NAKSTS**：NAK 状态(NAK status)指示以下状态： 0：根据 FIFO 状态，控制器发送非 NAK 握手；1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置还是控制器设置： 对于非同步 IN 端点，控制器都将停止数据传输，而不管发送 FIFO 中的数据是否有效。 对于同步 IN 端点，即使发送 FIFO 中的数据有效，控制器都将发送一个零长度的数据包。不管此位设置如何，控制器永远发送 ACK 握手以响应 SETUP 数据包。
16	**EONUM**	**EONUM**：偶数/奇数帧(Even/odd frame)仅对同步 IN 端点有效： 指示该端点进行同步数据收发的帧号。程序必需适用本寄存器中 SEVNFRM 和 SODDFRM 位的状态来，设置希望收发帧的偶数/奇数帧号。 0：偶数帧；1：奇数帧。 DPID：端点数据 PID(Endpoint data PID)仅对中断/块传输 IN 端点有效： 此位保存通过此端点来传输的数据包的 PID。应用程序必需在使能端点后，配置此位，选择首个发送或接收的数据包的 PID。应用程序通过 SD0PID 寄存器位来配置 DATA0 或 DATA1。

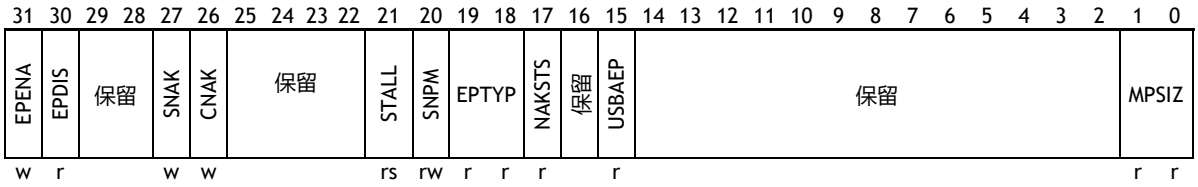
		0：DATA0；1：DATA1。
15	**USBAEP**	**USBAEP**：USB 活跃端点(USBactiveendpoint) 指示在当前配置下，此端点是否为活跃端点。控制器在检测到 USB 复位后会清除所有端点的此位(除了端点 0)，在收到 SetConfiguration 和 SetInterface 命令后，应用程序必需根据需要设置此位。
14:11	**Reserved**	保留
10:0	**MPSIZ**	**MPSIZ**：最大包长度(Maximum packet size) 应用程序通过此位配置该端点的最大数据包长度此位的值以字节为单位。

OTG_FS 设备控制 OUT 端点 0 控制寄存器(OTG_FS_DOEPCTL0)

偏移地址：0xB00

复位值：0x0000 8000

本节描述了设备控制 OUT 端点 0 的控制寄存器。非 0 控制端点使用相应的端点 1-15 控制寄存器。



位	符号	说明
31	**EPENA**	**EPENA**：端点使能(Endpoint enable) 应用程序设置此位，以启动端点 0 上的数据传输。控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成；端点禁止； 传输完成。
30	**EPDIS**	**EPDIS**：端点禁止(Endpoint disable)应用程序不可以禁止控制 OUT 端点 0。
29:28	**Reserved**	保留
27	**SNAK**	**SNAK**：设置 NAK(Set NAK)写此位会设置端点的 NAK 状态。 通过设置此位，应用程序能告知控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或传输结束中断时设置此位。
26	**CNAK**	**CNAK**：清除 NAK(Clear NAK)设置此位会清除端点的 NAK 状态。
25:22	**Reserved**	保留
21	**STALL**	**STALL**：STALL 握手(STALL handshake) 应用程序只能设置此位；当收到一个 SETUP 命令时，由控制器清除。即使同时设置了 NAK 位、或全局 OUTNAK 位，STALL 位仍有高优先级。然而无论此位如何设置，控制器总是以 ACK 握手来响应 SETUP 数据包。
20	**SNPM**	**SNPM**：监听模式(Snoop mode) 设置此位将使端点进入监听模式，在监听模式下，控制器在将数据传送入应用程序缓存区之前将不会检测 OUT 包的正确性。
19:18	**EPTYP**	**EPTYP**：端点类型(Endpoint type)由硬件设置为'00'。
17	**NAKSTS**	**NAKSTS**：NAK 状态(NAK status)指示以下状态： 0：根据 FIFO 状态，控制器发送非 NAK 握手信号。1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置还是控制器设置： 控制器都将停止数据传输，而不管接收 FIFO 中的数据是否有效。不管此位设置如何，控制器永远发送 ACK 握手信号响应 SETUP 数据包。
16	**Reserved**	保留
15	**USBAEP**	**USBAEP**：USB 活跃端点(USB active endpoint) 总是为'1'，表示控制端点 0 在任何配置下都是活跃的。
14:2	**Reserved**	保留

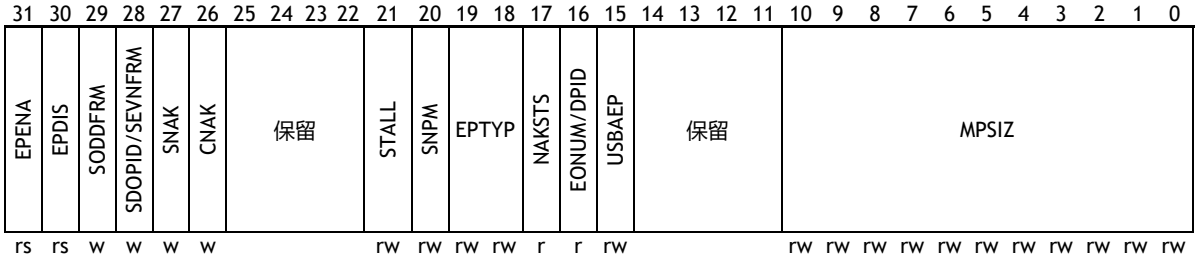
1:0	MPSIZ	**MPSIZ**：最大包长度(Maximum packet size) 控制 OUT 端点 0 的最大数据包长度必须与控制 IN 端点 0 的最大数据包长度一致。 00：64 字节； 01：32 字节； 10：16 字节； 11：8 字节。

OTG_FS 设备 OUT 端点 x 控制寄存器(OTG_FS_DOEPCTLx)(其中 x 为端点号，x=1…3)

偏移地址：0xB00+(端点号×0x20)

复位值：0x0000 0000

应用程序通过此寄存器来控制除了端点 0 的其他端点的操作特性。



位	符号	说明
31	EPENA	**EPENA**：端点使能(Endpoint enable)对于 IN 和 OUT 端点都有效。 应用程序设置此位，以启动端点上的数据传输。控制器在产生以下端点中断前会先清除此位： SETUP 阶段完成；端点禁止； 传输完成。
30	EPDIS	**EPDIS**：端点禁止(Endpoint disable) 应用程序通过设置此位可以立即停止端点上的数据传输，即使传输还未完成。应用程序需要等到端点禁止中断才能确为此端点已被禁止。控制器在设置端点中断时会清除此位。应用程序只有在端点已经使能时才能设置此位。
29	SODDFRM	**SODDFRM**：设置奇数帧(Set odd frame)仅对同步 OUT 端点有效 设置此位，则在奇偶帧域(EONUM)中选择奇数帧。
28	SD0PID	**SD0PID**：设置 DATA0PID(Set DATA0 PID)仅对中断/块传输 OUT 端点有效。 写此位将设置数据 PID 位为 DATA0。 SEVNFRM：设置偶数帧 仅对同步 OUT 端点有效。 设置此位，则在奇偶帧域(EONUM)中选择偶数帧。
27	SNAK	**SNAK**：设置 NAK(Set NAK)写此位会设置端点的 NAK 状态。 通过设置此位，应用程序能控制控制器发送 NAK 握手信号。控制器也会在收到 SETUP 包或 OUT 端点传输结束中断时设置此位。
26	CNAK	**CNAK**：清除 NAK(Clear NAK)写此位会清除端点的 NAK 状态。
25:22	Reserved	保留
21	STALL	**STALL**：STALL 握手(STALL handshake) 仅对非控制，非同步 OUT 端点有效(操作类型为 rw) 应用程序设置此位，该端点会以 STALL 来响应所有来自主机的命令。即使同时设置了 NAK 位、或全局 INNAK 位、或全局 OUTNAK 位，STALL 位仍有最高优先级。 只有应用程序能清除此位，控制器不能。 仅对控制端点有效(操作类型是 rs)

		应用程序只能设置此位，控制器会在收到 SETUP 命令时清除此位。即使同时设置了 NAK 位、或全局 INNAK 位、或全局 OUTNAK 位，STALL 位仍有高优先级。然而控制器将永远用 ACK 握手来响应 SETUP 数据包。
20	**SNPM**	**SNPM**：监听模式(Snoop mode) 设置此位将使端点进入监听模式。在监听模式下，控制器在将 OUT 数据包写入应用程序缓存区前不检查数据的正确性。
19:18	**EPTYP**	**EPTYP**：端点类型(Endpoint type)此位指示端点的传输类型： 00：控制 01：同步 10：块传输 11：中断
17	**NAKSTS**	**NAKSTS**：NAK 状态(NAK status)指示以下状态： 0：根据 FIFO 状态，控制器发送非 NAK 握手信号。1：控制器发送 NAK 握手信号。 只要设置了此位，不管是应用程序设置还是控制器设置： 控制器都将停止数据传输，而不管接收 FIFO 是否能接收数据。不管此位设置如何，控制器永远发送 ACK 握手响应 SETUP 数据包。
16	**EONUM**	**EONUM**：偶数/奇数帧(Even/odd frame)仅对同步 OUT 端点有效： 指示该端点进行同步数据收发的帧号。程序必需根据本寄存器中 SEVNFRM 和 SODDFRM 位的状态来设置偶数/奇数帧号。 0：偶数帧；1：奇数帧。 DPID：端点数据 PID(Endpoint data PID)仅对中断/块传输 OUT 端点有效： 此位保存通过此端点来传输的数据包的 PID。程序必需在使能端点后配置此位，选择首个发送或接收的数据包的 PID。应用程序通过 SD0PID 寄存器位来配置 DATA0 或 DATA1。 0：DATA0；1：DATA1。
15	**USBAEP**	**USBAEP**：USB 活跃端点(USB active endpoint) 指示在当前配置下，此端点是否为活跃端点。控制器在检测到 USB 复位后会清除所有端点的这一位(除了端点 0)，在收到 SetConfiguration 和 SetInterface 命令后，程序必需根据需要设置此位。
14:11	**Reserved**	保留
10:0	**MPSIZ**	**MPSIZ**：最大包长度(Maximum packet size) 应用程序通过此位配置该端点的最大数据包长度长度数值以字节为单位。

OTG_FS 设备端点 x 中断寄存器(OTG_FS_DIEPINTx)(其中 x 为端点号，x=0…3)

偏移地址：0x908+(端点号×0x20)

复位值：0x0000 0080

此寄存器指示相应端点与 USB 和 AHB 相关的事件状态。具体请参考图 282。当控制器中断寄存器的 IN 端点中断位(OTG_FS_GINTSTS 寄存器的 IEPINT 位)为'1'时，程序必需先读设备所有端点中断寄存器(OTG_FS_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTG_FS_DAINT 和 OTG_FS_GINTST 寄存器的对应位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留																		NAK	保留	PKTDRPSTS				TXFE	INEPNE	INEPNM	ITTXFE	TOC	保留	EPDISD	XFRC
																		rc_w1		rc_w1				r	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1

位	符号	说明
31:14	**Reserved**	保留

13	NAK	**NAK:**NAK 输入 当设备传输或接收 NAK 时，核心会生成此中断。对于等时 IN 端点，当由于 TxFIFO 中无数据而传输零长度数据包时，会生成中断。
12	Reserved	保留
11	PKTDRPSTS	**PKTDRPSTS:**数据包丢失状态 此位向应用程序表示已丢弃 ISOCOUT 数据包。此位没有关联的掩码位，也不会产生中断。
10:8	Reserved	保留
7	TXFE	**TXFE**：发送 FIFO 空(Transmit FIFO empty) 当与此端点相对应的发送 FIFO 为空或者半空时，会产生此中断。控制器 AHB 配置寄存器的 TxFIFO 空级别位(OTG_FS_GAHBCFG 寄存器的 TXFELVL 位)将决定发送 FIFO 在空还是半空状态时产生中断。
6	INEPNE	**INEPNE**：IN 端点 NAK 有效(IN endpoint NAK effective) 写 OTG_FS_DIEPCTLx 寄存器的 CNAK 位将清除 IN 端点的 NAK 状态，同时也清除此位。 此中断意味着控制器检测到 NAK 已经被设置(可以由应用程序设置也可以由控制器设置)，这个中断指示由应用程序设置的 NAK 位已经生效。 这个中断不能保证已经在 USB 线上发送了 NAK 信号。STALL 位的优先级高于 NAK 位。
5	Reserved	保留
4	ITTXFE	**ITTXFE**：当发送 FIFO 空时收到 IN 命令(IN token received when TxFIFO is empty)仅对于非周期性的 IN 端点有效。 指示与此端点相关的发送 FIFO(周期性/非周期性)为空时，收到了 IN 的命令。此中断仅在收到 IN 命令的端点产生。
3	TOC	**TOC**：超时(Timeout condition)仅对控制 IN 端点有效。 指示自该端点收到最后一个 IN 命令以来，控制器监测到了超时状态。
2	Reserved	保留
1	EPDISD	**EPDISD**：端点禁止中断(Endpoint disabled interrupt)此中断指示根据应用程序的请求，此端点已经被禁止。
0	XFRC	**XFRC**：传输完成中断(Transfer completed interrupt) 此中断指示该端点上已配置好的传输，无论在 AHB 方面还是 USB 方面，都已传输完毕。

OTG_FS 设备端点 x 中断寄存器(OTG_FS_DOEPINTx)(其中 x 为端点号，x=0…3)

偏移地址：0xB08+(端点号×0x20)

复位值：0x0000 0080

此寄存器指示相应端点与 USB 和 AHB 相关的事件状态。具体请参考图 282。当控制器中断寄存器的 OUT 端点中断位(OTG_FS_GINTSTS 寄存器的 OEPINT 位)为'1'时，应用程序必需先读设备所有端点中断寄存器(OTG_FS_DAINT)来获得发生事件的端点号，然后再读相应端点的中断寄存器获得详细信息。应用程序必需通过清除此位来清除 OTG_FS_DAINT 和 OTG_FS_GINTST 寄存器的对应位。

31:14	13	12	11:9	8	7:6	5	4	3	2	1	0
保留	NAK	BERR	保留	OUTPKTERR	保留	STSPHSRX	OTEPDIS	STUP	保留	EPDISD	XFRC
	rc_w1	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

位	符号	说明
31:13	Reserved	保留

13	NAK	**NAK:**NAK 输入 当设备传输或接收 NAK 时，核心会生成此中断。对于等时 IN 端点，当由于 TxFIFO 中无数据而传输零长度数据包时，会生成中断。
12	BERR	**BERR:**杂音错误中断 当为终结点接收到杂音时，核心会产生此中断。
11:9	Reserved	保留
8	OUTPKTERR	**OUTPKTERR:**OUT 数据包错误 当核心检测到输出数据包溢出或 CRC 错误时，会断定此中断。此中断只有在启用值时才有效。
7:6	Reserved	保留
5	STSPHSRX	**STSPHSRX:**控制写入状态相位已接收 只有在核心将主机在控制写入传输的数据阶段发送的所有数据传输到系统内存缓冲区之后，才会生成此中断。该中断向应用程序指示主机已从数据阶段切换到控制写入传输的状态阶段。应用程序可以使用此中断在解码数据阶段后确认或暂停状态阶段。
4	OTEPDIS	**OTEPDIS**：当端点禁止时收到了 OUT 命令(OUT token received when endpoint disabled)仅对控制 OUT 端点有效。 指示端点禁止时收到了 OUT 命令，此中断仅在收到 OUT 命令的端点上产生。
3	STUP	**STUP**：SETUP 阶段完成(SETUP phase done)仅对控制 OUT 端点有效。 指示与此控制端点相关的 SETUP 阶段已经完成，当前传输不会再有更多的连续的 SETUP 包。产生此中断后，应用程序可以开始处理收到的 SETUP 数据包。
2	Reserved	保留
1	EPDISD	**EPDISD**：端点禁止中断(Endpoint disabled interrupt)此中断指示根据应用程序的请求，此端点已经被禁止。
0	XFRC	**XFRC**：传输完成中断(Transfer completed interrupt) 此中断指示该端点上已配置好的传输，无论在 AHB 端还是 USB 端，都已传输完毕。

OTG_FS 设备 IN 端点 0 传输长度寄存器(OTG_FS_DIEPTSIZ0)

偏移地址：0x910

复位值：0x0000 0000

应用程序必需在使能端点 0 之前配置此寄存器。一旦通过设备控制端点 0 控制寄存器的端点使能位(OTG_FS_DIEPCTL0 寄存器的 EPENA 位)使能了端点 0，就只有控制器可以修改此寄存器。

当端点使能位被清除之前，应用程序只能读此寄存器。非 0 端点使用端点 1 至端点 15 的寄存器。

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

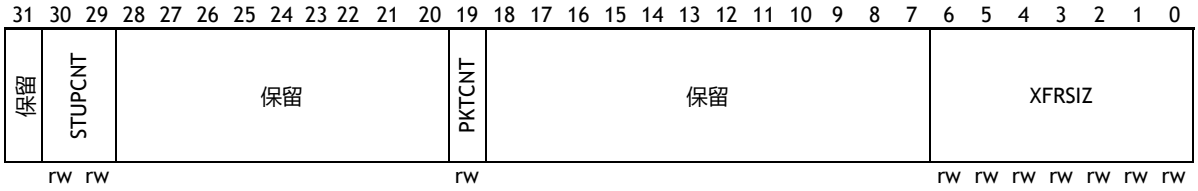
位	符号	说明
31:21	Reserved	保留
20:19	PKTCNT	**PKTCNT**：包数目(Packet count) 指示在端点 0 上要传输”传输长度”数据，所需要的 USB 数据包的数量。 控制器每从发送 FIFO 中读出一个数据包(最大长度的数据包和短包)，此寄存器的值会自动减 1。
18:7	Reserved	保留
6:0	XFRSIZ	**XFRSIZ**：传输长度(Transfer size) 指示端点 0 上要传输的字节数。当传输字节数减为 0 时，控制器会产生中断并通知应用程序。可以在每个包结束后，将此寄存器值设置为端点的最大传输长度。 控制器会在从存储区向发送 FIFO 写数据的时候自动递减此域。

OTG_FS 设备 OUT 端点 0 传输长度寄存器(OTG_FS_DOEPTSIZ0)

偏移地址：0xB10

复位值：0x0000 0000

应用程序必需在使能端点 0 前配置好此寄存器。一旦端点 0 通过设备控制端点 0 控制寄存器的端点使能位(OTG_FS_DOEPCTL0 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。控制器在清除端点使能位之前，应用程序只能读此寄存器。非 0 的端点使用针对端点 1 至端点 15 的寄存器。



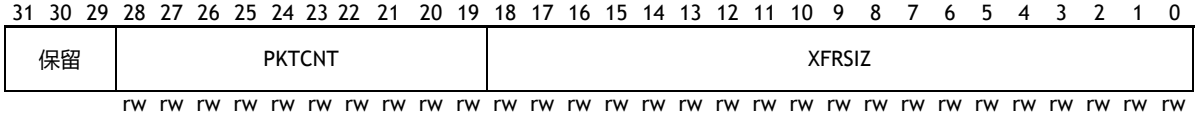
位	符号	说明
31	Reserved	保留
30:29	STUPCNT	**STUPCNT**：SETUP 包数目(SETUP packet count)此位指示端点能够接收的连续的 SETUP 包数目 01：1 个包； 10：2 个包； 11：3 个包。
28:20	Reserved	保留
19	PKTCNT	**PKTCNT**：包数目(Packet count) 当一个包被写入接收 FIFO 后，此位递减，直至减为 0。
18:7	Reserved	保留
6:0	XFRSIZ	**XFRSIZ**：传输长度(Transfer size) 指示端点 0 上的传输字节数。控制器在传输长度为 0 时产生中断并通知应用程序。可以在每个包结束时，设置此寄存器值为端点的最大传输长度，并在每个数据包结束时产生中断。 控制器在数据包从接收 FIFO 中写入存储区时会自动递减此域。

OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DIEPTSIZx)(其中 x 为端点号，x=1…3)

偏移地址：0x910+(端点号×0x20)

复位值：0x0000 0000

应用程序必需在使能端点前配置此寄存器。一旦端点通过设备端点 x 控制寄存器的端点使能位(OTG_FS_DIEPCTLx 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。控制器清除了端点使能位之前，应用程序只能读此寄存器。



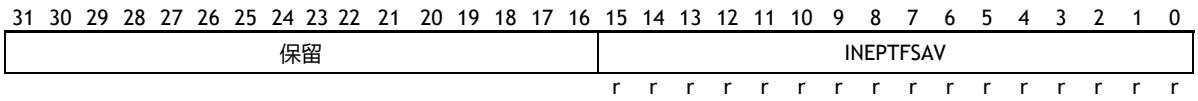
位	符号	说明

31:29	**Reserved**	保留
28:19	**PKTCNT**	**PKTCNT**：包数目(Packet count)指示端点上总共要发送的数据包数。 每从发送 FIFO 中读出一个数据包，自动递减此域。
18:0	**XFRSIZ**	**XFRSIZ**：传输长度 该位指示当前端点的传输长度。控制器会在此位为 0 时产生中断并通知应用程序。可以配置此寄存器为端点的最大传输长度，并在每个数据包结束产生中断。 控制器会在每次从存储区向发送 FIFO 写数据时，自动递减此域。

OTG_FS 设备 IN 端点传输 FIFO 状态寄存器(OTG-FS_DTXFSTSx)(其中 x 为端点号，x=0…3)

偏移地址：0x918+(端点号×0x20)

此寄存器为只读寄存器，保存设备 IN 端点的发送 FIFO 的剩余空间信息。



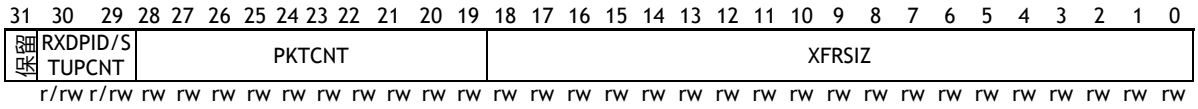
位	符号	说明
31:16	**Reserved**	保留
15:0	**INEPTFSAV**	**INEPTFSAV**：IN 端点发送 FIFO 剩余空间(IN endpoint TxFIFO space avail)指示 IN 端点的发送 FIFO 的剩余空间信息，寄存器值以 32 位的字为单位。 0x0：发送 FIFO 满； 0x1：剩余 1 个字； 0x02：剩余 2 个字； 0xn：剩余 n 个字(其中 0<n<512)； 0x200：剩余 512 个字； 其他：保留。

OTG_FS 设备端点 x 传输长度寄存器(OTG_FS_DOEPTSIZx)(其中 x 为端点号，x=1…3)

偏移地址：0xB10+(端点号×0x20)

复位值：0x0000 0000

应用程序必需在使能端点前，配置此寄存器。一旦端点通过设备端点 x 控制寄存器的端点使能位(OTG_FS_DOEPCTLx 寄存器的 EPENA 位)使能，就只有控制器能修改此寄存器。当控制器清除端点使能位之前，应用程序只能读此寄存器。



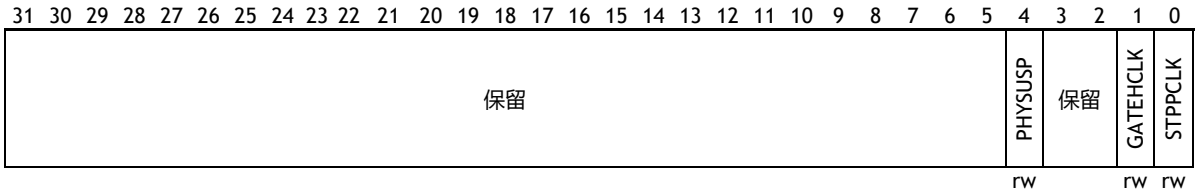
位	符号	说明
31	**Reserved**	保留
30:29	**RXDPID**	**RXDPID**：收到的数据 PID(Received data PID)此位仅对同步 OUT 端点有效。 指示收到的最后一个数据包的数据 PID：00：DATA0； 01：DATA2；10：DATA1；11：MDATA。 STUPCNT：SETUP 包数目(SETUPpacketcount)此位仅对控制 OUT 端点有效。 指示端点收到的连续的 SETUP 包的数量： 01：1 个包；10：2 个包；11：3 个包。
28:19	**PKTCNT**	**PKTCNT**：包数目(Packet count)指示该端点上传输的 USB 包数目。 控制器在每次从存储区向接收 FIFO 中写数据包时，自动递减此域。
18:0	**XFRSIZ**	**XFRSIZ**：传输长度(Transfer size) 此位指示该端点要传输的字节数。控制器在此域为 0 时会产生中断通知应用程序。可以配置此域为端点的最大传输长度，并在每个数据包结束产生中断。 控制器在每次从接收 FIFO 向存储区写数据时，自动递减此数值。

26.16.5 OTG_FS 电源和时钟门控寄存器(OTG_FS_PCGCCTL)

偏移地址：0xE00

复位值：0x0000 0000

此寄存器在设备模式和主机模式下都有效。



位	符号	说明
31:5	Reserved	保留
4	PHYSUSP	**PHYSUSP**：PHY 挂起 指示 PHY 已经挂起。此位在通过应用程序设置 STPPCLK 位(位 0)使 PHY 进入挂起状态时置为'1'。
3:2	Reserved	保留
1	GATEHCLK	**GATEHCLK**：HCLK 门控(Gate HCLK) 应用程序通过设置此位可以控制 HCLK，在 USB 挂起或会话无效时唤醒逻辑模块。应用程序在 USB 唤醒或新的会话发起时清除此位。
0	STPPCLK	**STPPCLK**：停止 PHY 时钟(Stop PHY clock) 当 USB 挂起、或会话无效、或者设备断开时，应用程序可以通过设置此位来停止 PHY 的时钟驱动。当 USB 唤醒或新的会话发起时，应用程序可以清除此位。

26.16.6 OTG_FS 寄存器映像

下表给出了 USBOTG 寄存器映像和复位值。

表 156 OTG_FS 模块的寄存器及其复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	OTG_FS_GOTGCTL	保留												BSVLD	ASVLD	DBCT	CIDSTS	保留				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	保留						SRQ	SRQSCS
	复位值													0	0	0	1					0	0	0	0							0	0
004h	OTG_FS_GOTGINT	保留												DBCDNE	ADTOCHG	HNGDET	保留							HNSSCHG	SRSSCHG	保留					SEDET	保留	
	复位值													0	0	0								0	0						0		
008h	OTG_FS_GAHBCFG	保留																							PTXFELVL	TXFELVL	保留						GINT
	复位值																								0	0							0
00Ch	OTG_FS_GUSBCFG	CTXPKT	FDMOD	FHMOD	保留													NPTXRWEN	TRDT					HNPCAP	SRPCAP	保留					TOCAL		
	复位值	0	0	0														0	0	0	1	0	1	0	0						0	0	0
010h	OTG_FS_GRSTCTL	AHBIDL	保留																TXFNUM							TXFFLSH	RXFFLSH	保留				FCRST	HSRST	CSRST
	复位值	1																	0	0	0	0	0			0	0				0	0	0
014h	OTG_FS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	保留	PTXFE	HCINT	HPRTINT	保留		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	保留		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	保留		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	复位值	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
018h	OTG_FS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	保留	PTXFEM	HCIM	PRTIM	保留		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	保留	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	保留		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	保留
	复位值	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
01Ch	OTG_FS_GRXSTSR (主机模式)	保留												PKTSTS				DPID				BCNT									CHNUM		

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_FS_GRXSTSR (设备模式)	保留							FRMNUM				PKTSTS				DPID		BCNT												EPNUM			
	复位值								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	OTG_FS_GRXSTSPR (主机模式)	保留											PKTSTS				DPID		BCNT												CHNUM			
	复位值												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_FS_GRXSTSPR (设备模式)	保留							PRMNUM				PKTSTS				DPID		BCNT												EPNUM			
	复位值								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	OTG_FS_GRXFSIZ	保留																	RXFD															
	复位值																		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
028h	OTG_FS_GNPTXFSIZ	NPTXFD																	NPTXFSA															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
02Ch	OTG_FS_GNPTXSTS	保留	NPTXQTOP							NPTQXSAV									NPTXFSAV															
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
038h	OTG_FS_GCCFG	保留											SOFOUTEN	VBUSBSEN	VBUSASEN	保留	PWRDWN		保留															
	复位值												0	0	0		0																	
03Ch	OTG_FS_CID	PRODUCT_ID																																
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
100h	OTG_FS_HPTXFSIZ	PTXFSIZ																	PTXSA															
	复位值	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0		0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
104h	OTG_FS_DIEPTXF1	INEPTXFD																	INEPTXSA															
	复位值	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
108h	OTG_FS_DIEPTXF2	INEPTXFD																	INEPTXSA															
	复位值	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
10Ch	OTG_FS_DIEPTXF3	INEPTXFD																	INEPTXSA															
	复位值	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
110h	OTG_FS_DIEPTXF4	INEPTXFD																	INEPTXSA															
	复位值	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
400h	OTG_FS_HCFG	保留																													FSLSS		FSLSPCS	
	复位值																														0		0	0
404h	OTG_FS_HFIR	保留																	FRIVL															
	复位值																		1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
408h	OTG_FS_HFNUM	FTREM																	FRNUM															
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
410h	OTG_FS_HPTXSTS	PTXQTOP								PTXQSAV									PTXFSAVL															
	复位值	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
414h	OTG_FS_HAINT	保留																	HAINT															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
418h	OTG_FS_HAINTMSK	保留																	HAINTM															
	复位值																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
440h	OTG_FS_HPRT	保留													PSPD		PTCTL				PPWR	PLSTS		保留	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	复位值														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
500h	OTG_FS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
520h	OTG_FS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
540h	OTG_FS_HCCHAR2	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
560h	OTG_FS_HCCHAR3	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
580h	OTG_FS_HCCHAR4	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5A0h	OTG_FS_HCCHAR5	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5C0h	OTG_FS_HCCHAR6	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5E0h	OTG_FS_HCCHAR7	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	保留	EPDIR	EPNUM				MPSIZ										
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
508h	OTG_FS_HCINT0	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
528h	OTG_FS_HCINT1	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
548h	OTG_FS_HCINT2	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
568h	OTG_FS_HCINT3	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
588h	OTG_FS_HCINT4	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
5A8h	OTG_FS_HCINT5	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
5C8h	OTG_FS_HCINT6	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
5E8h	OTG_FS_HCINT7	保留																					DTERR	FRMOR	BBERR	TXERR	保留	ACK	NAK	STALL	保留	CHH	XFRC
	复位值																						0	0	0	0		0	0	0		0	0
50Ch	OTG_FS_HCINTMSK0	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
52Ch	OTG_FS_HCINTMSK1	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
54Ch	OTG_FS_HCINTMSK2	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
56Ch	OTG_FS_HCINTMSK3	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
58Ch	OTG_FS_HCINTMSK4	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
5ACh	OTG_FS_HCINTMSK5	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
5CCh	OTG_FS_HCINTMSK6	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
5ECh	OTG_FS_HCINTMSK7	保留																					DTERRM	FRMORM	BBERRM	TXERRM	保留	ACKM	NAKM	STALLM	保留	CHHM	XFRCM
	复位值																						0	0	0	0		0	0	0		0	0
510h	OTG_FS_HCTSIZ0	保	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
530h	OTG_FS_HCTSIZ1	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
550h	OTG_FS_HCTSIZ2	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
570h	OTG_FS_HCTSIZ3	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
590h	OTG_FS_HCTSIZ4	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5B0h	OTG_FS_HCTSIZ5	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5D0h	OTG_FS_HCTSIZ6	保留	DPID		PKTCNT											XFRSIZ																	
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5F0h	OTG_FS_HCTSIZ7	保	DPID		PKTCNT											XFRSIZ																	

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
800h	OTG_FS_DCFG	保留																			PFIVL		DAD							保留	NZLSOHSK	DSPD	
	复位值																				0	0	0	0	0	0	0	0	0		0	0	0
804h	OTG_FS_DCTL	保留																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	复位值																					0	0	0	0	0	0	0	0	0	0	0	0
808h	OTG_FS_DSTS	保留																			FNSOF										保留	EERR	ENUMSPD	SUSPSTS
	复位值																				0	0	0	0	0	0	0	0	0		0	0	0
810h	OTG_FS_DIEPMSK	保留																			NAKM	保留								INEPNEM	INEPNMM	ITTXFEMSK	TOM	保留	EPDM	XFRCM
	复位值																													0	0	0	0		0	0
814h	OTG_FS_DOEPMSK	保留																			NAKMSK	BERRM	保留					OUTPKTERRM	保留	STSPHSRXM	OTEPDM	STUPM	保留	EPDM	SFRCM
	复位值																											0		0	0	0	0	0	0
818h	OTG_FS_DAINT	OEPINT																IEPINT																	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
81Ch	OTG_FS_DAINTMSK	OEPM																IEPM																	
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
828h	OTG_FS_DVBUSDIS	保留																VBUSDT																	
	复位值																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
82Ch	OTG_FS_DVBUSPULSE	保留																			DVBUSP													
	复位值																				0	1	0	1	1	0	1	1	1	0	0	0	
834h	OTG_FS_DIEPEMPMSK	保留																INEPTXFEM																	
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
900h	OTG_FS_DIEPCTL0	EPENA	EPDIS	保留		SNAK	CNAK	TXFNUM				STALL	保留	EPTYP		NAKSTS	保留	USBAEP	保留													MPSIZ		
	复位值	0	0			0	0	0	0	0	0	0		0	0	0		1														0	0
918h	OTG_FS_DTXFSTS0	保留																INEPTFSAV																	
	复位值																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
920h	OTG_FS_DIEPCTL1	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	保留	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留													MPSIZ		
	复位值	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
938h	OTG_FS_DTXFSTS1	保留																INEPTFSAV																	
	复位值																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
940h	OTG_FS_DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	保留	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留		MPSIZ												
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
958h	OTG_FS_DTXFSTS2	保留															INEPTFSAV																
	复位值																0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
960h	OTG_FS_DIEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	保留	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留		MPSIZ												
	复位值	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
978h	OTG_FS_DTXFSTS3	保留															INEPTFSAV																
	复位值																0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
B00h	OTG_FS_DOEPCTL0	EPENA	EPDIS	保留		SNAK	CNAK	保留				STALL	SNPM	EPTYP		NAKSTS	保留	USBAEP	保留													MPSIZ	
	复位值	0	0			0	0					0	0	0	0	0		1														0	0
B20h	OTG_FS_DOEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	保留				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留		MPSIZ												
	复位值	0	0	0	0	0	0					0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
B40h	OTG_FS_DOEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	保留				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留		MPSIZ												
	复位值	0	0	0	0	0	0					0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
B60h	OTG_FS_DOEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	保留				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	保留		MPSIZ												
	复位值	0	0	0	0	0	0					0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
908h	OTG_FS_DIEPINT0	保留																NAK	保留	PKTDRPSTS	保留			TXFE	INEPNE	INEPNM	ITTXFE	TOC	保留	EPDISD	XFRC		
	复位值																	0		0				1	0	0	0	0		0	0		
928h	OTG_FS_DIEPINT1	保留																NAK	保留	PKTDRPSTS	保留			TXFE	INEPNE	INEPNM	ITTXFE	TOC	保留	EPDISD	XFRC		
	复位值																	0		0				1	0		0	0		0	0		
948h	OTG_FS_DIEPINT2	保留																NAK	保留	PKTDRPSTS	保留			TXFE	INEPNE	INEPNM	ITTXFE	TOC	保留	EPDISD	XFRC		

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值																	0		0				1	0		0	0		0	0		
968h	OTG_FS_DIEPINT3	保留																NAK	保留	PKTDRPSTS	保留			TXFE	INEPNE	INEPNM	ITTXFE	TOC	保留	EPDISD	XFRC		
	复位值																	0		0				1	0	0	0	0		0	0		
B08h	OTG_FS_DOEPINT0	保留																NAK	BERR	保留				OUTPKTERR	保留	STSPHSRX	OTEPDIS	STUP	保留	EPDISD	XFRC		
	复位值																	0	0					0		0	0	0		0	0		
B28h	OTG_FS_DOEPINT1	保留																NAK	BERR	保留				OUTPKTERR	保留	STSPHSRX	OTEPDIS	STUP	保留	EPDISD	XFRC		
	复位值																	0	0							0	0	0		0	0		
B48h	OTG_FS_DOEPINT2	保留																NAK	BERR	保留				OUTPKTERR	保留	STSPHSRX	OTEPDIS	STUP	保留	EPDISD	XFRC		
	复位值																	0	0					0		0	0	0		0	0		
B68h	OTG_FS_DOEPINT3	保留																NAK	BERR	保留				OUTPKTERR	保留	STSPHSRX	OTEPDIS	STUP	保留	EPDISD	XFRC		
	复位值																	0	0					0		0	0	0		0	0		
910h	OTG_FS_DIEPTSIZ0	保留																			PKT CNT			保留								XFRSIZ		
	复位值																				0	0										0	0	0	0	0	0	0
930h	OTG_FS_DIEPTSIZ1	保留																PKTCN														XFRSIZ		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
950h	OTG_FS_DIEPTSIZ2	保留																PKTCNT														XFRSIZ		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
970h	OTG_FS_DIEPTSIZ3	保留																PKTCNT														XFRSIZ		
	复位值																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B10h	OTG_FS_DOEPTSIZ0	保留		STUPCNT																				PKTCNT								XFRSIZ		
	复位值			0	0																			0								0	0	0	0	0	0	0
B30h	OTG_FS_DOEPTSIZ1	保留		RXDPID/ STUPCNT														PKTCNT														XFRSIZ		
	复位值			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B50h	OTG_FS_DOEPTSIZ2	保留		RXDPID/ STUPCNT														PKTCNT														XFRSIZ		
	复位值			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B70h	OTG_FS_DOEPTSIZ3	保留		RXDPID/ STUPCNT														PKTCNT														XFRSIZ		

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	复位值		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
E00h	OTG_FS_PCGCCTL	保留																									PHYSUSP	保留		GATEHCLK	STPPCLK		
	复位值																																

请参考表 1，获得这些寄存器的基地址。

26.17 OTG_FS 编程规则

26.17.1 控制器初始化

应用程序必须按照顺序初始化控制器。如果在上电期间，USB 电缆已经连上，控制器中断寄存器的当前操作模式位(OTG_FS_GINTSTS 的 CMOD 位)将指出当前的工作模式。当 A 类插头插入时，OTG_FS 控制器工作在主机模式下，当 B 类插头插入时，控制器工作在设备模式下。

本节介绍上电后 OTG_FS 控制器的初始化。无论在主机模式或是在设备模式下工作，应用程序都必须按照顺序初始化控制器。所有的控制器全局寄存器都必须按照控制器的配置来进行初始化：

1. 配置 AHB 全局配置寄存器(OTG_FS_GAHBCFG)的如下位：
   - 全局的中断屏蔽位 GINT = 1
   - 接收 FIFO 非空位(OTG_FS_GINTSTS 寄存器的 RXFLVL 位)
   - 周期性的发送 FIFO 空级别
2. 配置 OTG_FS_GUSBCFG 寄存器的如下位：
   - HNP 使能位
   - SRP 使能位
   - FS 超时校准位
   - USB 反射时间位
3. 应用程序不能屏蔽 GINTMSK 寄存器的如下位：
   - OTG 中断屏蔽位
   - 模式不匹配中断屏蔽位
4. 应用程序通过读取 OTG_FS_GINTSTS 寄存器的 CMOD 位来判断当前 OTF_FS 控制器处于主机模式还是处于设备模式。

26.17.2 主机模式下的初始化

在主机模式下，应用程序需遵循如下步骤来配置控制器：

1. 将 GINTMSK 寄存器的 HPRTINT 位配置为'1'，使能该中断。
2. 配置 OTG_FS_HCFG 寄存器，选择工作在全速的主机模式下。
3. 配置 OTG_FS_HPRT 寄存器的 PPWR 位为'1'，使能 USB 线上的 VBUS 供电。
4. 等待 OTG_FS_HPRT0 寄存器的 PCDET 中断，该中断表示有 USB 设备接到端口。
5. 配置 OTG_FS_HPRT 寄存器的 PRST 位为'1'，对设备发起复位操作。
6. 等待最少 10ms，保证复位操作完成。
7. 配置 OTG_FS_HPRT 寄存器的 PRST 位为'0'。
8. 等待 OTG_FS_HPRT 寄存器的 PENCHNG 位中断。
9. 读取 OTG_FS_HPRT 寄存器的 PSPD 位，该位指示设备使用全速还是低速进行枚举。
10. 根据已选择的 PHY 时钟 1 来配置 HFIR 寄存器。
11. 配置 OTG_FS_RXFSIZE 寄存器，选择接收 FIFO 的长度。
12. 配置 OTG_FS_NPTXFSIZE 寄存器，选择非周期性发送 FIFO 的长度和起始地址。
13. 配置 OTG_FS_HPTXFSIZ 寄存器，选择周期性发送 FIFO 的长度和起始地址。为了和设备通讯，应用程序必须使能并初始化至少一个通道。

26.17.3 设备模式下的初始化

应用程序必须在上电或者从主机模式切换到设备模式时，遵循如下步骤来初始化控制器，使之工作于设备模式下：

1. 配置 OTG_FS_DCFG 寄存器的如下位：
   - 设备速度
   - 对非零长度的 OUT 包的响应状态
2. 配置 OTG_FS_GINTMSK 寄存器，使能以下中断：
   - USB 复位
   - 枚举完成
   - USB 早期挂起
   - USB 挂起
   - SOF
3. 在 B 类设备模式下，配置 OTG_FS_GCCFG 寄存器的 VBUSBSEN 位使能 VBUS，使 DP 线上拉到 5V。
4. 等待 OTG_FS_GINTSTS 寄存器的 USBRST 位，指示在 USB 线上检测到了持续大约 10ms 的复位信号。

等待OTG_FS_GINTSTS寄存器的ENUMDNE位，它表示USB复位操作的结束。在接收到此中断后，应用程序必须读取 OTG_FS_DSTS 寄存器，获得枚举速度的信息，并按照”枚举完成后的端点初始化”小节实现初始化。

此时，设备已准备好接收 SOF 数据包，并通过端点 0 实现控制传输。

26.17.4 主机模式下的编程规则

通道初始化

在主机与所连接的设备通讯之前，应用程序需要初始化一个或多个通道。可通过如下步骤初始化并使能通道：

1. 配置 GINTMSK 寄存器，使能以下中断：
2. 通道中断
   - 对于 OUT 传输的非周期性发送 FIFO 空(适用于从模式，即配置运行在传输级的流水线上的包数超过 1)
   - 对于 OUT 传输的非周期性发送 FIFO 半空(适用于从模式，即配置运行在传输级的流水线上的包数超过 1)
3. 配置 OTG_FS_HAINTMSK 寄存器，使能选中通道的中断。
4. 配置选中通道的 OTG_FS_HCINTMSK 寄存器，使能在主机通道中断寄存器中与传输相关的中断。
5. 配置选中通道的 OTG_FS_HCTSIZx 寄存器，以字节为单位设置总传输长度，和期望接收到的数据包数，包括短数据包。需要根据初始的数据 PID 号来设置寄存器的 PID 位(将用于首个传输的 OUT 包的数据 PID 号和期望首个接收的 IN 包的数据 PID 号)。
6. 配置选中通道的 OTG_FS_HCCHARx 寄存器，设置设备端点的特性，例如传输类型、速度、方向等。(仅在应用程序准备好传输或接收数据包时，才需要设置通道使能位为'1'，使能通道。)

中止通道

应用程序可以通过设置 OTG_FS_HCCHARx 寄存器的 CHDIS 和 CHENA 位为'1'来中止任何一个通道。这个操作将使 OTG_FS 主机控制器清除已递交的传输请求(如果存在的话)，并产生一个通道中止中断。应用程序需要在重新分配此通道用于其他通讯前，等待 OTG_FS_HCINTx 寄存器的 CHH 位为'1'(指示此通道已中止)。OTG_FS 的主机控制器不能打断已经开始的 USB 总线上的传输。

在中止通道前，应用程序需要确认在非周期性请求队列(中止的是非周期性通道时)、或周期性请求队列(中止的是周期性通道时)中至少有一个剩余空间。在请求队列已满时(执行通道中止操作前)，可以通过写 OTG_FS_HCCHARx 寄存器的 CHDIS 位为'1'并等待 CHENA 位变为'0'，来清除已递交的传输请求。

应用程序需要在以下情况中止通道：

1. 在一个非周期性的 IN 传输或者高带宽的中断 IN 传输时(仅在从模式下)，检测到 OTG_FS_HCINTx 寄存器的 XFRC 位为'1'。
2. 在任意 IN 或者 OUT 通道(仅在从模式下)的 OTG_FS_HCINTx 寄存器中产生了 STALL、TXERR、BBERR 或 DTERR 事件。对于高带宽的中断 IN 传输(仅在从模式下)，一旦应用程序检测到了 DTERR 事件，就必须中止该通道并等待该通道已成功中止的事件。在收到该通道已成功中止的信息前，应用程序需要能接收该通道的其他事件(DTERR、NAK、DATA、TXERR)。
3. 当 OTG_FS_GINTSTS 寄存器的 DISCINT 为'1'时(指示设备已断开)，应用程序需要中止所有已使能的通道。
4. 当应用程序需要在传输正常结束前中止传输。

操作模式

应用程序需要在与所连接的设备通讯前初始化一个通道，本节介绍了针对不同的 USB 传输类型所进行的操作。

写入发送 FIFO

OTG_FS 主机模式控制器会在程序按照 DWORD 方式写数据包时，自动地向周期性/非周期性请求队列中写入请求(OUT 请求)，因此在写入发送 FIFO 之前，必须确保周期性/非周期性请求队列中至少有一个空余位置。应用程序只能以 DWORD 的方式来写入发送 FIFO，如果要写的数据包不是 DWORD 对齐的，需要补齐剩余字节。OTG_FS 主机模式控制器会按照配置好的最大数据包长度和实际传输长度来决定实际发送的数据包长度。

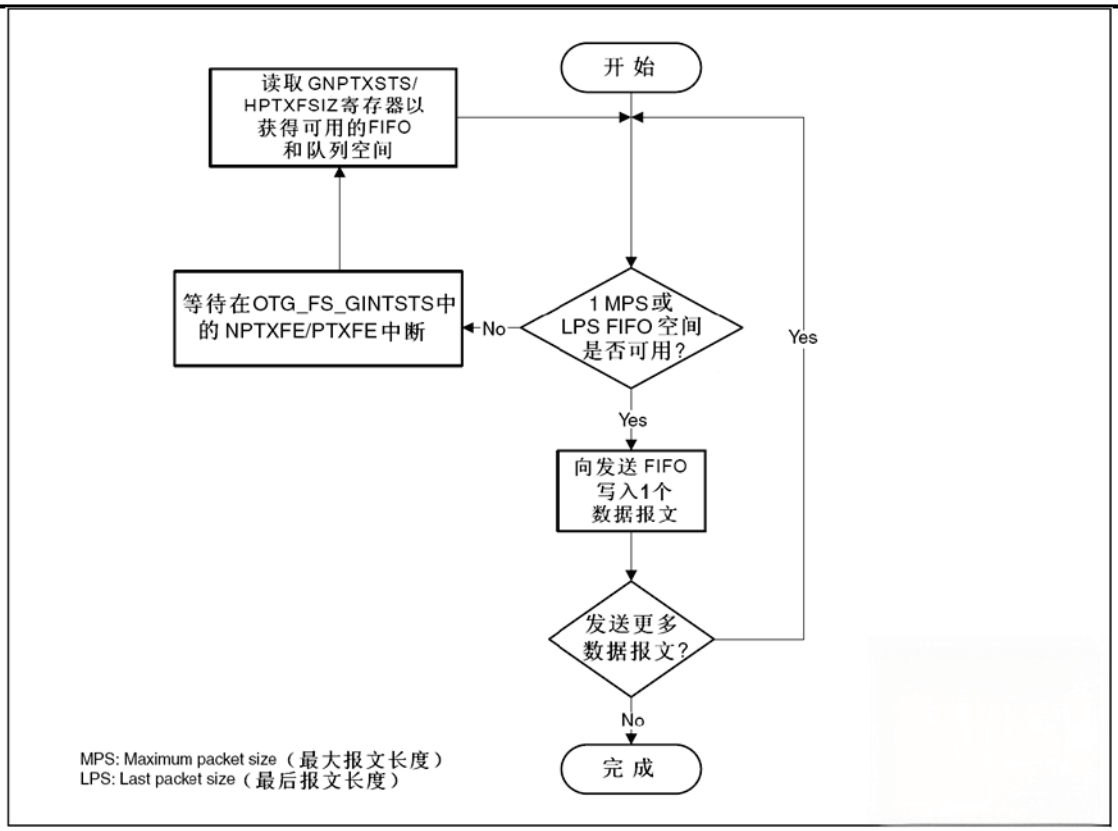


图 284 发送 FIFO 写任务

读取接收 FIFO

只有在收到 IN 的数据包时(0x0010)应用程序才需要读取接收 FIFO。

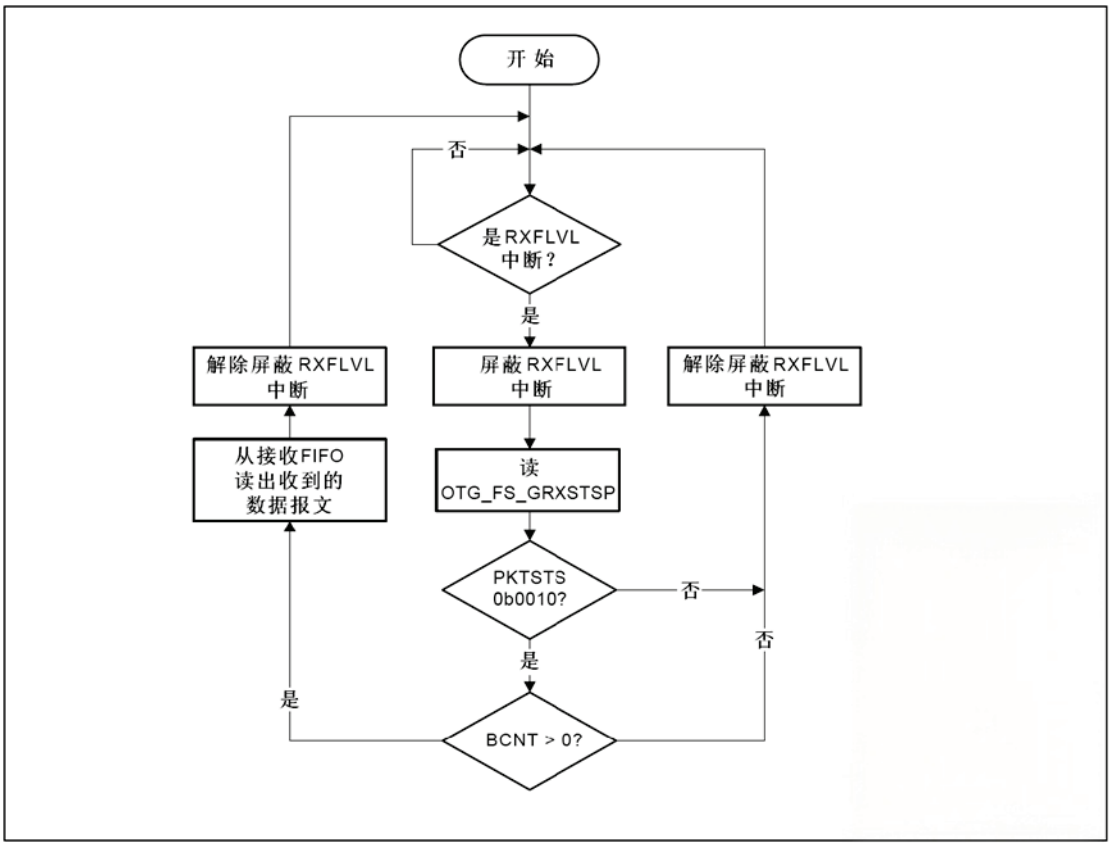


图 285 接收 FIFO 读出任务

块传输和控制传输的 OUT/SETUP

下图给出了典型的块传输或控制传输的 OUT/SETUP 的操作流程。详见通道 1(ch_1)。2 个块传输的 OUT 包需要发送，1 个控制传输的 SETUP 包传输也需要处理。

假设：

- 应用程序需要发送 2 个最大包长度的数据包(传输长度为 1024 字节)
- 非周期性的发送 FIFO 已保存了 2 个数据包(对于全速传输有 128K 字节)
- 非周期性的请求队列深度为 4

普通块传输和控制传输的 OUT/SETUP 处理流程

下图(使用通道 1)所描述的处理流程如下：

a) 初始化通道 1。
b) 写通道 1 的第一个包。
c) 随着最后一个 DWORD 数据的写入，控制器向非周期性请求队列写入一个请求。
d) 当非周期性请求队列非空时，控制器在当前帧内发送一个 OUT 令牌。
e) 将第二个数据包(最后一个)写入通道 1。
f) 在前一个传输正常结束时，控制器产生一个 XFRC 中断。
g) 收到 XFRC 事件，重新安排通道为其他传输服务。
h) 控制非 ACK 响应。

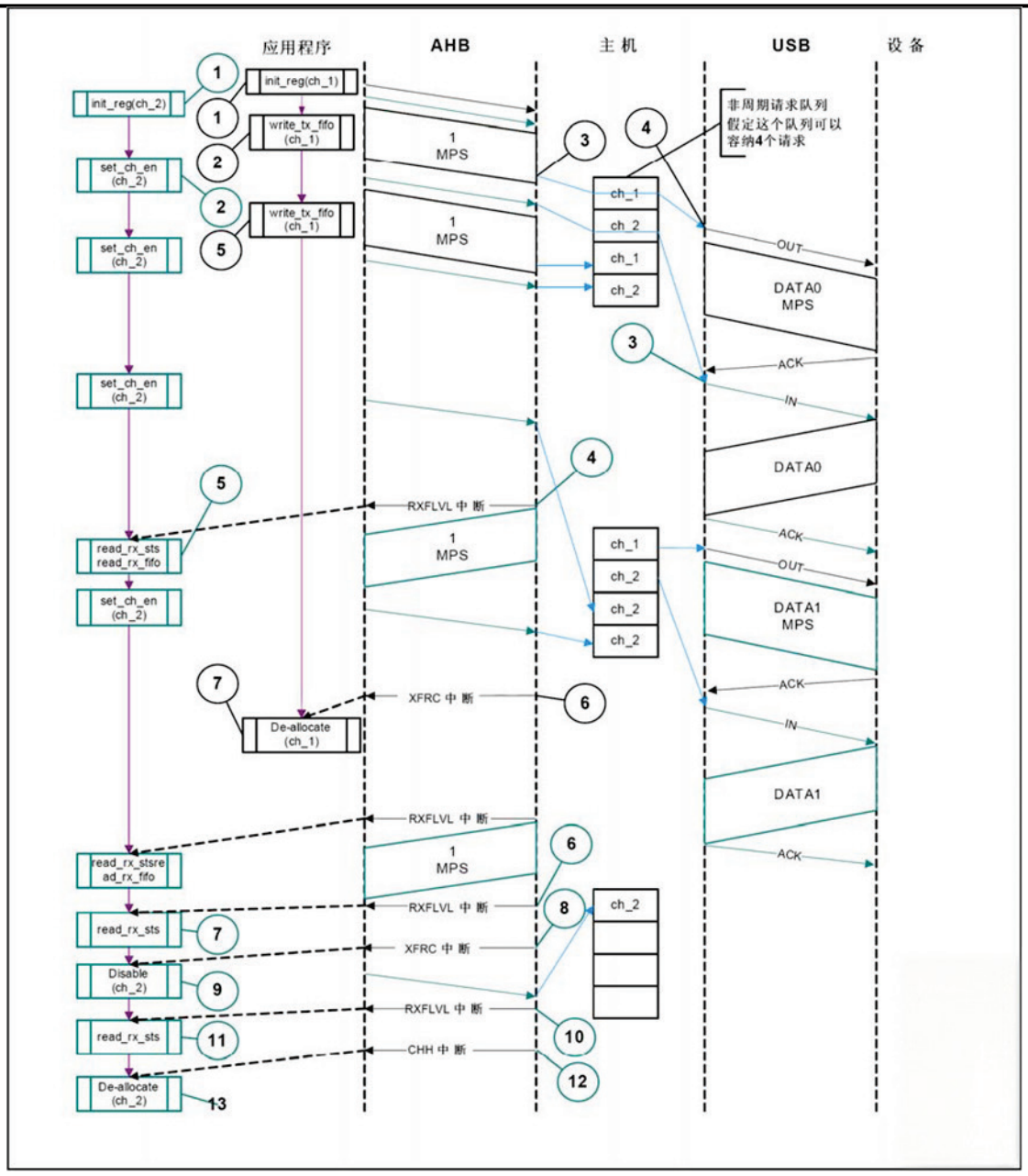


图 286 普通的块(Bulk)/控制(Control)的 OUT/SETUP 和块/控制的 IN 传输过程

下面针对从模式下块传输和控制传输的 OUT/SETUP 流程，列出了与通道相关的中断处理程序代码例程。

块和控制 OUT/SETUP 传输与块和控制 IN 传输的中断处理流程

a) 块和控制 OUT/SETUP 传输

使能(NAK/TXERR/STALL/XFRC)中断
if(XFRC)
{
清除错误计数
屏蔽 ACK 中断
解除通道分配
}
else if(STALL)
{
传输完成=1

```
使能 CHH 中断
中止通道
}
else if(NAK 或 TXERR)
{
重置缓冲区指针
使能 CHH 中断
中止通道
if(TXERR)
{
增加错误计数
使能 ACK 中断
}
else
{
清除错误计数
}
}
else if(CHH)
{
屏蔽 CHH 中断
if(传输完成或(错误计数==3))
{
解除通道分配
}
else
{
重新初始化通道
}
}
else if(ACK)
{
清除错误计数屏蔽 ACK 中断
}
```

应用程序必须在发送 FIFO 和请求队列有空余空间时才能向发送 FIFO 写数据包。可以通过 OTG_FS_GINTSTS 寄存器的 NPTXFE 位来检测发送 FIFO 是否有空余空间。

b) 块和控制的 IN 传输

```
使能(TXERR/XFRC/BBERR/STALL/DTERR)中断
if(XFRC)
{
复位错误计数
使能 CHH 中断
中止通道
屏蔽 ACK
}
else if(TXERR 或 BBERR 或 STALL)
{
使能 CHH 中断
中止通道
```

```
    if(TXERR)
      {
      增加错误计数
      使能 ACK 中断
      }
    }
  else if(CHH)
    {
    屏蔽 CHH
    if(传输完成或(错误计数==3))
      {
      重新分配通道
      }
    else
      {
      重新初始化通道
      }
    }
  else if(ACK)
    {
    复位错误计数器屏蔽 ACK
    }
  else if(DTERR)
    {
    复位错误计数器
    }
```

应用程序必须等到 XFRC 事件发生后，才能在请求队列有剩余空间时写入请求。

块/控制的 IN 传输

典型的块(BULK)和控制 IN 传输的操作流程如下图所示；见通道 2。做如下假设：

- 应用程序需要接收 2 个最大包长度的数据包(传输长度为 1024 字节)。
- 接收 FIFO 可以保存至少 1 个最大包长度的数据包和针对每个数据包的 2 个 DWORD 类型状态。(对于全速通信来说是 72 个字节)。
- 非周期性请求队列深度为 4。

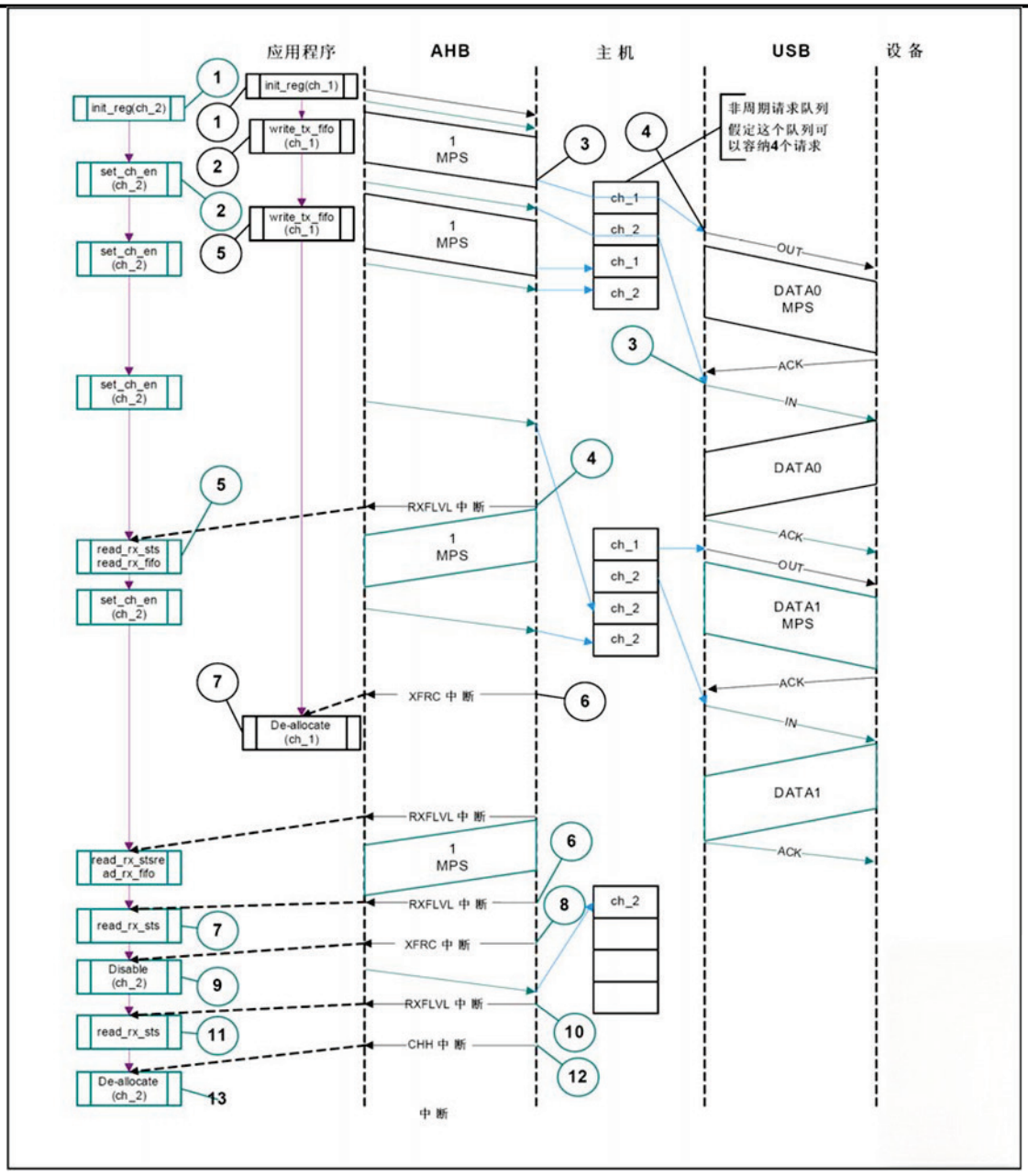


图 287 块(Bulk)/控制(Control)的 IN 传输过程

操作流程如下：

a) 初始化通道 2

b) 设置 HCCHAR2 寄存器的 CHENA 位，向非周期性请求队列写入一个 IN 请求。

c) 控制器在完成当前 OUT 传输后尝试发送 IN 令牌。

d) 在接收到的数据写入接收 FIFO 后，控制器产生 RXFLVL 中断。

e) 在 RXFLVL 中断处理程序中，需要先屏蔽 RXFLVL 中断，读取接收到的状态判定接收到的字节数，并从接收 FIFO 中读取相应数据。完成以上操作后再打开 RXFLVL 中断。

f) 控制器在传输完成的状态存入接收 FIFO 后产生 RXFLVL 中断。

g) 应用程序读取接收到的数据包状态，如果它不是一个 IN 的数据包(即 GRXSTSR 寄存器的 PKTSTS 位≠0x0010)，则不理它。

h) 控制器在接收到的数据包状态被读出后产生 XFRC 中断。

i) 在 XFRC 中断处理程序中，需要配置 OTG_FS_HCCHAR2 寄存器来中止通道，并停止写入更多的请求。控制器会在 OTG_FS_HCCHAR2 寄存器被设置完后向非周期性请求队列写入通道中止请求。

j) 控制器将在中止状态写入接收 FIFO 后产生 RXFLVL 中断。

k) 读出该包状态，但不处理它。

l) 控制器将在中止信息从接收 FIFO 中读出后产生 CHH 中断。

m) 在 CHH 中断处理程序中，应用程序可以重新分配该通道为其他传输服务。

n) 控制非 ACK 响应。

从模式下的控制传输

Setup、数据和状态是控制传输的 3 个阶段，必须按照 3 个独立的传输来对待。Setup、数据 OUT 和状态 OUT 传输的处理流程类似于前面介绍的块 OUT 传输。数据 IN 和状态 IN 传输的处理流程类似于前面介绍的块 IN 传输。在这 3 个阶段中，应用程序都需要设置 OTG_FS_HCCHAR1 寄存器的 EPTYP 位，表明传输类型为控制传输。在 Setup 阶段，应用程序需要设置 OTG_FS_HCTSIZ1 寄存器的 DPID 位，表明为 SETUP 包。

中断 OUT 传输

典型的中断 OUT 传输的操作流程如下图所示。做如下假设：

- 应用程序从奇数帧开始在每个帧中都发送一个最大数据包长度的包(发送长度为 1024 字节)。
- 周期性的 FIFO 中能储存 1 个包(1024 字节)。
- 周期性的请求队列深度为 4。操作流程如下：

a) 初始化通道 1，同时应用程序需要配置 OTG_FS_HCCHAR1 寄存器的 ODDFRM 位，指明从奇数帧开始发送。

b) 将第一个要传输的包写入通道 1。对于高带宽的中断传输，应用程序需要在切换到其他通道前，先将后续需要传输的包写入通道，包数由 MCNT 位)在下一帧时需要传输的包数)指定。

c) 在写完每个包的最后一个 DWORD 时，控制器会将请求写入周期性请求队列。

d) 在下一个奇数帧时，控制器会发送一个 OUT 令牌。

e) 在最后一个包正常传输完毕后，控制器会产生 XFRC 中断。

f) 在 XFRC 中断处理程序中，应用程序可以重新初始化通道，以便通道为其他传输所用。

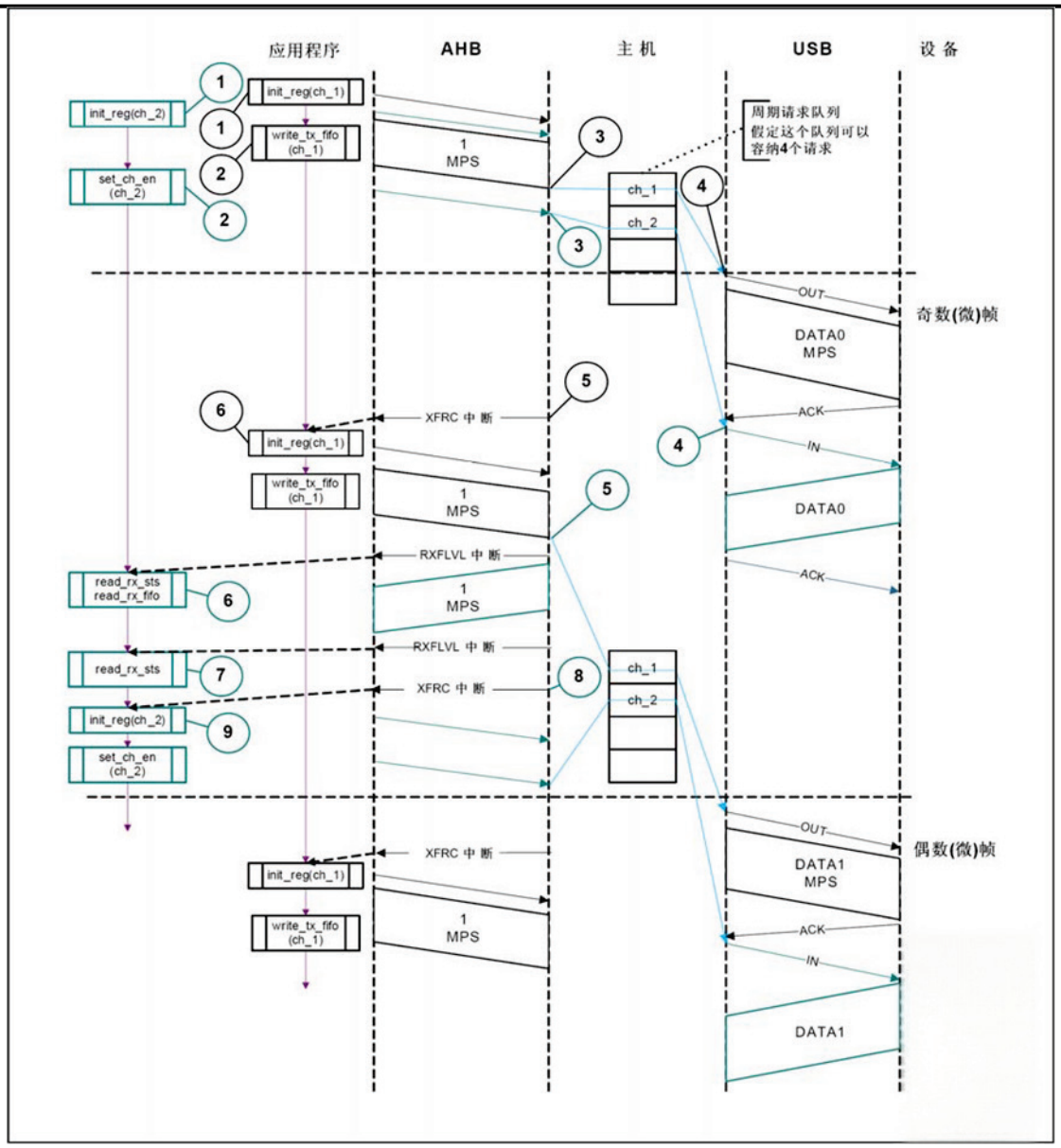


图 288 普通的中断(Interrupt)OUT/IN 传输过程

针对中断 OUT/IN 传输的中断服务程序

a) 中断 OUT

使能(NAK/TXERR/STALL/XFRC/FRMOR)中断

if(XFRC)

{

复位错误计数

屏蔽 ACK 中断

重新分配通道

}

else if(STALL 或 FRMOR)

{

屏蔽 ACK 中断

使能 CHH 中断

中止通道

if(STALL)

{

```
        传输完成 = 1
        }
      }
    else if(NAK 或 TXERR)
      {
      重置缓存区指针
      复位错误计数
      屏蔽 ACK 中断
      使能 CHH 中断
      中止通道
      }
    else if(CHH)
      {
      屏蔽 CHH 中断
      if(传输完成或(错误计数==3))
        {
        重新分配通道
        }
        else
        {
        重新初始化通道(为下一个 b_interval - 1 的帧)
        }
      }
    else if(ACK)
      {
      复位错误计数屏蔽 ACK 中断
      }
```

应用程序需要在切换到其他通道之前，先根据 MCNT 位指定的数据包数，把数据包和请求写入发送 FIFO 和请求队列，此时发送 FIFO 必须有剩余空间，可以通过 OTG_FS_GINTSTS 寄存器的 PTXFE 位来获得发送 FIFO 是否有剩余空间的信息。

b) 中断 IN

```
使能(NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if(XFRC)
  {
  复位错误计数
  屏蔽 ACK
  if(OTG_FS_HCTSIZx.PKTCNT==0)
    {
    重新分配通道
    }
    else
    {
    传输完成 = 1
    使能 CHH 中止通道
    }
  }
else if(STALL 或 FRMOR 或 NAK 或 DTERR 或 BBERR)
  {
  屏蔽 ACK 使能
  CHH
```

```
        中止通道
        if(STALL 或 BBERR)
          {
          复位错误计数器
          传输完成 = 1
          }
        else if(！ FRMOR)
          {
          复位错误计数
          }
        }
      else if(TXERR)
        {
        增加错误计数
        使能 ACK
        使能 CHH
        中止通道
        }
      else if(CHH)
        {
        屏蔽 CHH
        if(传输完成或(错误计数==3))
          {
          重新分配通道
          }
        else
          {
          重新初始化通道(为下一个 b_interval-1 帧)
          }
        }
      else if(ACK)
        {
        复位错误计数屏蔽 ACK
        }
```

应用程序需要在切换到其他通道前，向有剩余空间的请求队列写入请求，直到写入的请求数达到在 MCNT 位指定的数目。

中断 IN 传输

假设：

- 程序需要从奇数帧开始每个帧接收一个最大数据包长度的包(传输长度为 1024 字节)
- 接收 FIFO 可以保存指示 1 个最大数据包长度的包和 2 个 DWORD 类型的状态字(共 1031 字节)
- 周期性请求队列深度为 4

普通的中断 IN 操作

操作流程如下：

a) 初始化通道 2，程序需要写 OTG_FS_HCCHAR2 寄存器的 ODDFRM 位，指定奇数帧。
b) 设置 OTG_FS_HCCHAR2 寄存器的 CHENA 位，控制器会向周期性请求队列写入 IN 请求。对于高带宽的中断传输，程序需要在切换到其他通道前写 OTG_FS_HCCHAR2 寄存器的 MCNT 位(指定在下一个帧期间需要接收的包数目)。
c) 每次程序设置 OTG_FS_HCCHAR2 寄存器 CHENA 位时，控制器都会向周期性请求队列写入一个 IN 请求。
d) 在下一个奇数帧时，控制器会发送一个 IN 令牌。
e) 控制器收到 IN 的数据包并写入 RXFIFO 后，会产生 RXFLVL 中断。
f) 在 RXFLVL 中断处理程序中，应用程序读取接收到的包状态，得知收到的字节数目，并相应地读取接收 FIFO。在读接收 FIFO 前，需要屏蔽 RXFLVF 中断，并在读完完整的包后使能 RXFLVL 中断。
g) 控制器在传输完成状态存入接收 FIFO 后，产生 RXFLVL 中断。程序需要读取这个包状态，并且在状态表示非 IN 的数据包时(GRXSTSR 寄存器的 PKTSTS 位≠0x0010)丢弃这个包。
h) 在读取接收到的包状态后，控制器会产生 XFRC 中断。
i) 在 XFRC 中断处理程序中，应用程序读取 OTG_FS_HCTSIZ2 寄存器的 PKTCNT 位，如果该位不为 0，中止通道，然后重新初始化通道，准备下一次的传输。如果 PKTCNT 位为 0，重新初始化通道，准备下一次的传输。此时应用程序需要复位 OTG_FS_HCCHAR2 寄存器的 ODDFRM 位。

同步 OUT 传输

典型的从模式下的同步 OUT 传输的流程图如下图所示。假设：

- 程序需要从奇数帧开始每个帧发送一个最大数据包长度的包(发送长度为 1024 字节)。
- 周期性发送 FIFO 能储存 1 个最大数据包长度的包(1024 字节)。
- 周期性请求队列深度为 4。操作流程如下：

a) 初始化并使能通道 1，程序需要设置 OTG_FS_HCCHAR1 寄存器的 ODDFRM 位。
b) 将第一个要发送的包写入通道 1。对于高带宽的同步传输，应用程序需要在切换到其他通道前根据 MCNT 位(下一个帧时间内需要发送的最大包数)将后续要发送的数据包都写入通道。
c) 在每个包的最后一个 DWORD 数据写完后，控制器会写一个请求到周期性的请求队列中。
d) 控制器会在下一个奇数帧时发送 OUT 令牌。
e) 当最后一个包正确传输完毕后，控制器会产生 XFRC 中断。
f) 在 XFRC 中断处理程序中，需要重新初始化通道为下一次传输做准备。
g) 控制非 ACK 响应。

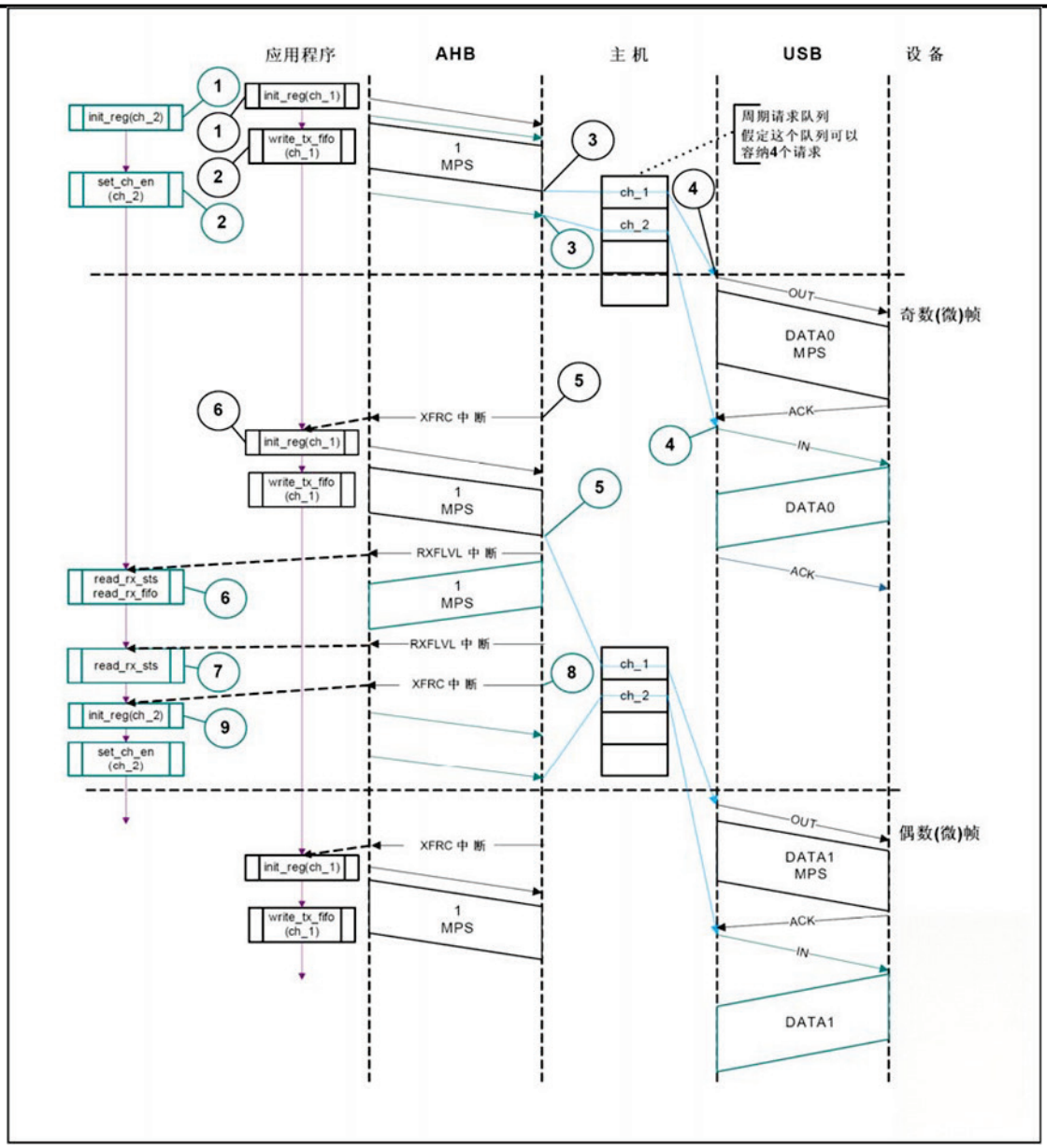


图 289 普通的同步(Isochronous)OUT/IN 传输过程

同步 OUT/IN 传输的中断处理程序

代码例子：同步 OUT：

使能(FRMOR/XFRC)

if(XFRC)

{

重新分配通道

}

else if(FRMOR)

{

使能 CHH

中止通道}

else if(CHH)

{

屏蔽 CHH

重新分配通道

}

代码例子：同步 IN

```
使能(TXERR/XFRC/FRMOR/BBERR)
    if(XFRC 或 FRMOR)
      {
      if(XFRC 并且 OTG_FS_HCTSIZx.PKTCNT==0)
        {
        复位错误计数
        重新分配通道
        }
      else
        {
        使能 CHH
        中止通道
        }
      }
    else if(TXERR 或 BBERR)
      {
      增加错误计数
      使能 CHH
      中止通道
      }
    else if(CHH)
      {
      屏蔽 CHH
      if(传输完成或(错误计数==3))
        {
        重新分配通道
        }
      else
        {
        重新初始化通道
        }
      }
```

同步 IN 传输

假设：

- 程序从奇数帧起，每个帧都需要接收一个最大数据长度的包(传输长度为 1024 字节)。
- 接收 FIFO 至少可以储存一个最大数据长度的包和针对每个包的两个 DWORD 类型的状态字(共 1031 字节)。
- 周期性请求队列深度为 4。操作流程如下：

a) 初始化通道 2，应用程序需要设置 OTG_FS_HCCHAR2 寄存器的 ODDFRM 位。

b) 设置 OTG_FS_HCCHAR2 寄存器的 CHENA 位，控制器会将 IN 请求写入周期性请求队列。对于高带宽的同步传输，应用程序需要在切换到其他通道前写 OTG_FS_HCCHAR2 寄存器的 MCNT 位(下一个帧时间内需要接收的最大包数)。

c) 每次设置 OTG_FS_HCCHAR2 寄存器的 CHENA 位，控制器都会写一个 IN 请求到周期性请求队列。

d) 控制器会在下一个奇数帧时发送 IN 令牌。

e) 当控制器收到 IN 包并写入接收 FIFO 后，会产生 RXFLVL 中断。
f) 在 RXFLVL 中断处理程序中，读取收到的数据包状态，并判定需要接收的字节数，然后读取接收 FIFO。应用程序需要在读取接收 FIFO 前屏蔽 RXFLVL 中断，并在读完后使能。
g) 在控制器将传输完成的状态信息写入接收 FIFO 后，会产生 RXFLVL 中断。此时应用程序需要读取状态，并在判定不是 IN 数据包(OTG_FS_GRXSTSR 寄存器的 PKTST 位≠0x0010)后丢弃这个包。
h) 控制器会在读取接收到的包状态后产生 XFRC 中断。
i) 在 XFRC 中断处理程序中，应用程序需要读取 OTG_FS_HCTSIZ2 寄存器的 PKTCNT 位。如果 PKTCNT 位不为 0，应用程序需要中止通道，并重新初始化通道，为下一次传输做准备。如果 PKTCNT 为 0，则重新初始化通道为下一次传输做准备。此时应用程序需要复位 OTG_FS_HCCHAR2 寄存器的 ODDFRM 位。

设置队列深度

需要小心选择周期性和非周期性请求队列的深度，使之与周期性/非周期性的端点数相匹配。
非周期性请求队列的深度仅影响非周期性的传输效率。请求队列越深(配合足够的 FIFO 空间)，控制器能流水线式的操作越多的非周期性传输。如果请求队列较少，控制器只有在请求队列有剩余空间时才能写入一个新的请求。
周期性请求队列的深度将对周期性传输的调度有较大的影响。一定要按照每个微帧时间内要周期性传输的包数来决定周期性请求队列的深度。在从模式下，应用程序还需要考虑到那些必须写入请求队列的无效请求。因此，如果有 2 个非高带宽的周期性端点，周期性请求队列的深度至少是 4。如果至少支持一个高带宽的周期性端点，请求队列深度必须是 8。如果周期性请求队列的深度小于每个微帧时间内需要周期性传输的包数，就会发生帧溢出。

管理混乱现象

OTG_FS 控制器管理两类混乱现象：包混乱和端口混乱。
设备发送的数据超过主机通道支持的最大数据包长度时会发生包混乱。主机在 EOF2(第二类帧结束信号，非常接近于帧首信号)时还不停的收到设备发送的数据时会发生端口混乱。
当 OTG_FS 控制器检测到包混乱时，将停止往接收缓存区中写数据，并等待包结束信号(EOP)。当检测到 EOP 后，控制器将清除已写入缓存区的数据，并产生一个混乱中断通知应用程序。
当 OTG_FS 控制器检测到端口混乱时，将清除接收 FIFO 并中止端口，同时产生端口无效中断(OTG_FS_CINTSTS 寄存器的 HPRTINT 位和 OTG_FS_HPRT 寄存器的 PENCHNG 位)。应用程序在处理该中断时，需要先读 OTG_FS_HPRT 寄存器的 POCA 位，来排除是由于端口过流造成的无效(这是产生端口无效中断的另一个源)，然后再执行一个软件复位操作。在检测到一个端口混乱事件后，控制器不能再发送任何令牌。

26.17.5 设备模式下的编程规则

USB 复位时的端点初始化

1. 将所有 OUT 端点都设为 NAK 状态。
   - 设置 OTG_FS_DOEPCTLx(所有的 OUT 端点)寄存器的 SNAK 位为'1'
2. 使能以下中断位:
   - 设置 OTG_FS_DAINTMSK 寄存器的 INEP0 位为'1'(控制 IN 端点 0)
   - 设置 OTG_FS_DAINTMSK 寄存器的 OUTEP0 位为'1'(控制 OUT 端点 0)
   - 设置 DOEPMSK 寄存器的 STUP 位为'1'
   - 设置 DOEPMSK 寄存器的 XFRC 位为'1'
   - 设置 DIEPMSK 寄存器的 XFRC 位为'1'
   - 设置 DIEPMSK 寄存器的 TOC 位为'1'
3. 为每个 FIFO 分配 RAM 空间
   - 设置 OTG_FS_GRXFSIZ 寄存器，用以接收控制传输的 OUT 数据和 SETUP 数据。所分配 RAM 的最小值应该是控制端点 0 的 1 个最大数据包的长度+2 个 DWORD 空间(用于控制传输的 OUT 数据包的状态信息)+10 个 DWORD 空间(用于 SETUP 包)。
   - 配置 OTG_FS_GNPTXFSIZ 寄存器(根据所选用的 FIFO 编号)，用于发送控制传输的 IN 数据。所分配 RAM 的最小值应为控制端点 0 的 1 个最大数据包长度。
4. 设置与端点相关的寄存器的以下位，用于控制 OUT 端点 0 接收 SETUP 数据包。
   - 设置 OTG_FS_DOEPTSIZ0 寄存器的 STUPCNT = 3(用于接收 3 个连续的 SETUP 数据包)。

此时，初始化完成，可以开始接收 SETUP 数据包。

枚举完成后的端点初始化

1. 在枚举完成中断(OTG_FS_GINTSTS 寄存器的 ENUMDNE 位)中，读取 OTG_FS_DSTS 寄存器，获得枚举速度的信息。
2. 配置 OTG_FS_DIEPCTL0 寄存器的 MPSIZ 位，设置最大的包长度。这个步骤配置的是控制端点 0。对于控制端点，最大的包长度取决于枚举速度。

此时，设备已经准备好接收 SOF 包了，并且能执行端点 0 上的控制传输。

Set Address 时的端点配置

应用程序在收到 SETUP 包中的 SetAddress 命令后所要进行的操作:

1. 用包含在 SetAddress 命令中的设备地址信息来配置 OTG_FS_DCFG 寄存器。
2. 配置控制器，使之发出状态 IN 包。

Set Configuration/Set Interface 命令时的端点配置

应用程序在收到 SETUP 包中的 Set Configuration 或 Set Interface 命令后要进行的操作:

1. 在收到 Set Configuration 命令时，需要按照新的配置中定义的属性来配置端点寄存器。
2. 在收到 Set Interface 命令时，要配置该命令所涉及到的端点寄存器。
3. 部分端点可能仅在先前的设置中有效，在新的配置或设置中可能已经失效，这些已失效的端点需要重新配置为无效。
4. 设置 OTG_FS_DAINTMSK 寄存器，使能每个有效端点的中断，屏蔽所有无效端点的中断。
5. 为每个 FIFO 分配 RAM 空间。
6. 在配置完所有的端点后，应用程序需要使控制器发送一个状态 IN 包。此时，设备模式下的控制器已准备好发送或接收数据包了。

端点激活

激活一个端点或者将一个已存在的端点配置成新类型的步骤：

1. 配置 OTG_FS_DIEPCTLx 寄存器(对于 IN 或者双向端点)或者 OTG_FS_DOEPCTLx 寄存器(对于 OUT 或者双向端点)的以下位：
   - 最大包长度
   - USB 有效端点位 = '1'
   - 端点起始的数据翻转号(对于中断和块传输类型的端点)
   - 端点的类型
   - 发送 FIFO 号
2. 一旦端点被激活，控制器将解析发送到该端点的令牌，并对于有效的令牌发送有效的握手包。

端点无效

使一个已存在的端点失效的操作：

1. 清除 OTG_FS_DIEPCTLx 寄存器(对于 IN 或者双向端点)或者 OTG_FS_DOEPCTLx 寄存器(对于 OUT 或者双向端点)的 USB 有效端点位。
2. 一旦端点失效，控制器将会忽略发向该端点的令牌，这将导致 USB 超时。

注意： *应用程序需要进行以下的配置，以便设备模式下的控制器来控制通信：GINTMSK 寄存器的 NPTXFEM 和 RXFLVLM 位必须清0'。*

26.17.6 操作流程

SETUP 和 OUT 数据传输

本节描述了 OUT 数据传输和 SETUP 传输的内部数据流及应用程序的操作流程。

- 读操作

在从模式下从接收 FIFO 中读一个包(OUT 和 SETUP 包)的步骤如下：

1. 在 RXFLVL 中断处理程序中，需要读取接收状态弹出寄存器(OTG_FS_GRXSTSP)。
2. 在从接收 FIFO 中读出数据包期间，写入 RXFLVL=0(OTG_FS_GINTSTS 寄存器)来屏蔽 RXFLVL 中断。
3. 如果接收到的包的字节数不为 0，则相应的数据将从数据 FIFO 中弹出，并储存到内存中，如果收到的包的字节数为 0，则没有数据从数据 FIFO 中弹出。
4. 从接收 FIFO 中读出的包状态为以下几种模式之一：
   a) 全局的 OUTNAK 模式：
   PKTSTS=全局的 OUTNAK，BCNT = 0x000，EPNUM = 不关心(0x0)，DPID = 不关心(0x0)。这样的数据表示有一个全局的 OUTNAK。
   b) SETUP 包模式：
   PKTSTS = SETUP，BCNT = 0x008，EPNUM = 控制端点号，DPID = D0。这样的数据表示针对指定端点的 SETUP 包已有效，可以从接收 FIFO 中读取。
   c) SETUP 阶段已完成模式：
   PKTSTS = SETUP 阶段已完成，BCNT = 0，EPNUM = 控制端点号，DPID = 不关心 (0x0)。这样的数据表示一个针对指定端点的 SETUP 阶段已完成，并已经开始数据阶段。在这个信息从接收 FIFO 中弹出后，控制器将产生一个指定的控制 OUT 端点的 SETUP 中断。

d) 数据 OUT 包模式：

PKTSTS=数据 OUT，BCNT = 接收到的 OUT 数据包长度(0≤BCNT≤1024)，EPNUM = 接收到包的端点号，DPID = 实际的数据 PID。

e) 数据阶段已完成模式：

PKTSTS=数据 OUT 阶段已完成，BCNT = 0x0，EPNUM = OUT 端点数据传输的端点 号 ，DPID = 不关心(0x0)。

这样的数据表示一个针对指定 OUT 端点的 OUT 数据阶段已完成。在这个信息从接收 FIFO 中弹出后，控制器将会产生一个针对指定 OUT 端点的传输完成中断。

5. 在读出接收 FIFO 中的数据后，需要再次打开 RXFLVL 中断(OTG_FS_GINTSTS)。
6. 在每次检测到一个 RXFLVL(OTG_FS_GINTSTS)中断后，都需要重复以上五个步骤。读一个空的接收 FIFO 会导致未定义的后果。

下图是以上流程的流程图

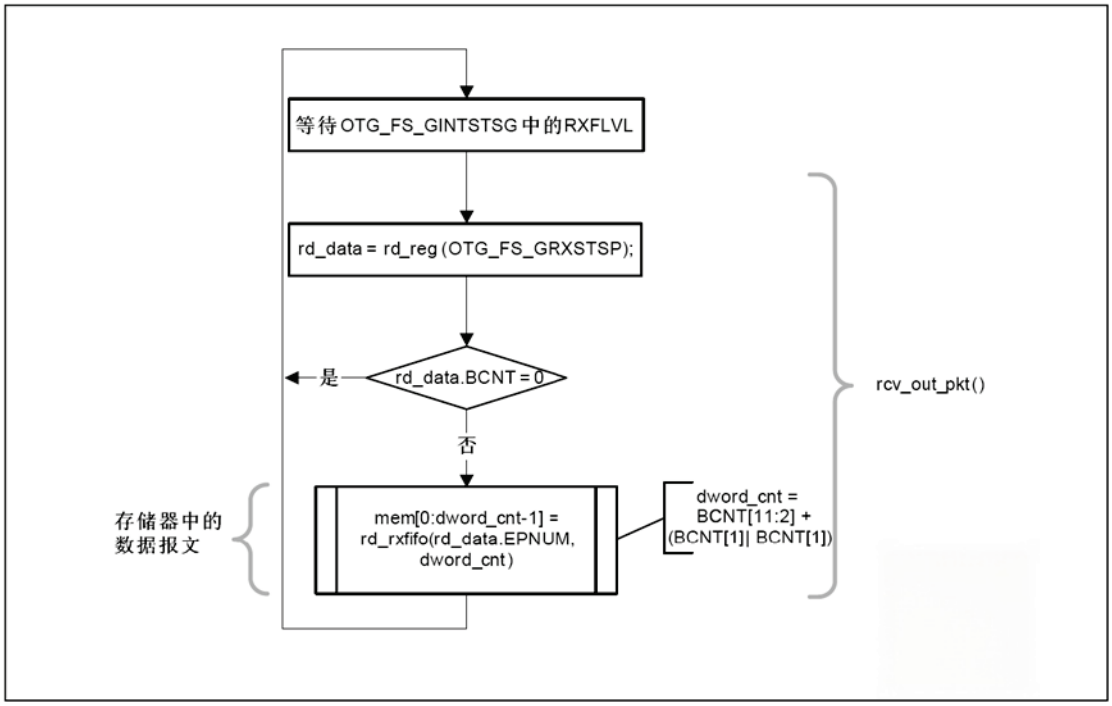


图 290 在从模式下读出接收 FIFO 的数据报文

- SETUP 传输

本节描述了控制器如何控制一个 SETUP 的包，以及应用程序处理 SETUP 传输的过程。

- 对应用程序的要求

1. 为了接收一个 SETUP 包，控制 OUT 端点的 STUPCNT 位(OTG_FS_DOEPTSIZx 寄存器中)必须被配置为一个非 0 值。当应用程序配置 STUPCNT 位为非 0 时，控制器将接收 SETUP 包，并将其写入接收 FIFO 中，此操作与 OTG_FS_DOEPCTLx 寄存器的 NAKSTS 和 EPENA 位的设置无关。每当控制端点收到一个 SETUP 包，STUPCNT 位的值都会自减 1。如果在接收 SETUP 包之前，没有设置一个合适的 STUPCNT 值，控制器仍将接收 SETUP 包，并自减 STUPCNT 位，但应用程序就不能知道在控制传输的 SETUP 阶段到底接收了多少个正确的 SETUP 包。

   – 设置 OTG_FS_DOEPTSIZx 寄存器的 STUPCNT 位 = 3

2. 程序需要为接收 FIFO 分配一些额外的空间，以便接收最多 3 个 SETUP 包。

- 需要分配的空间是10个DWORD。其中3个DWORD空间用于第一个SETUP包，1个DWORD空间用于 SETUP 阶段完成信息，另 6 个 DWORD 空间用于储存控制端点的另两个额外的SETUP 包。
- 一个 SETUP 包需要 3 个 DWORD 空间，其中 8 个字节用于储存 SETUP 数据，4 个字节用于储存 SETUP 状态(SETUP 包模式)。控制器会为接收到的数据保留这些空间。
- FIFO 仅用于 SETUP 包，而不用于数据包。

3. 应用程序需要从接收 FIFO 中读 SETUP 包的 2 个 DWORD 长度的数据。
4. 应用程序需要从接收 FIFO 中读一个 DWORD 的 SETUP 阶段完成信息。

- 内部的数据流程

5. 当收到 SETUP 包，控制器会将收到的数据储存到接收 FIFO 中，此时不会检测接收 FIFO 是否有剩余空间，也不检测该端点的 NAK 或者 STALL 位的状态。
   - 在收到 SETUP 包后，控制器会在内部将控制 IN/OUT 端点的状态位设为 INNAK 和 OUTNAK。
6. 从 USB 线上收到每个 SETUP 包后，都会有 3 个 DOWRD 的数据写入接收 FIFO 中，并且 STUPCNT 位自减 1。
   - 第一个 DWORD 数据包含控制器内部使用的控制信息。
   - 第二个 DWORD 数据包含 SETUP 命令的前 4 个字节。
   - 第三个 DWORD 数据包含 SETUP 命令的后 4 个字节。
7. 当 SETUP 阶段完成，数据 IN/OUT 阶段开始时，控制器会写一个信息(DWORD 类型的 SETUP 阶段完成信息)到接收 FIFO 中，指示 SETUP 阶段的完成。
8. 应用程序通过 AHB 总线读取 SETUP 包。
9. 当应用程序取出接收 FIFO 中 DWORD 类型的 SETUP 阶段完成信息，控制器将产生一个 STUP 中断(OTG_FS_DOEPINTx)，指示 SETUP 包已收完，应用程序可以开始处理接收到的 SETUP 包了。
   - 控制器将清除控制 OUT 端点的端点使能位。

- 应用程序处理流程

1. 配置 OTG_FS_DOEPTSIZx 寄存器
   - STUPCNT = 3
2. 等待 RXFLVL 中断(OTG_FS_GINTSTS)，并读取接收 FIFO 中的数据包。
3. 等待 STUP 中断(OTG_FS_DOEPINTx)，指示一个成功完成的 SETUP 阶段。
   - 在此中断处理程序中，应用程序需要读 OTG_FS_DOEPTSIZx 寄存器，以便知道接收到了多少个 SETUP 包，并对最后一个接收到的 SETUP 包进行处理。

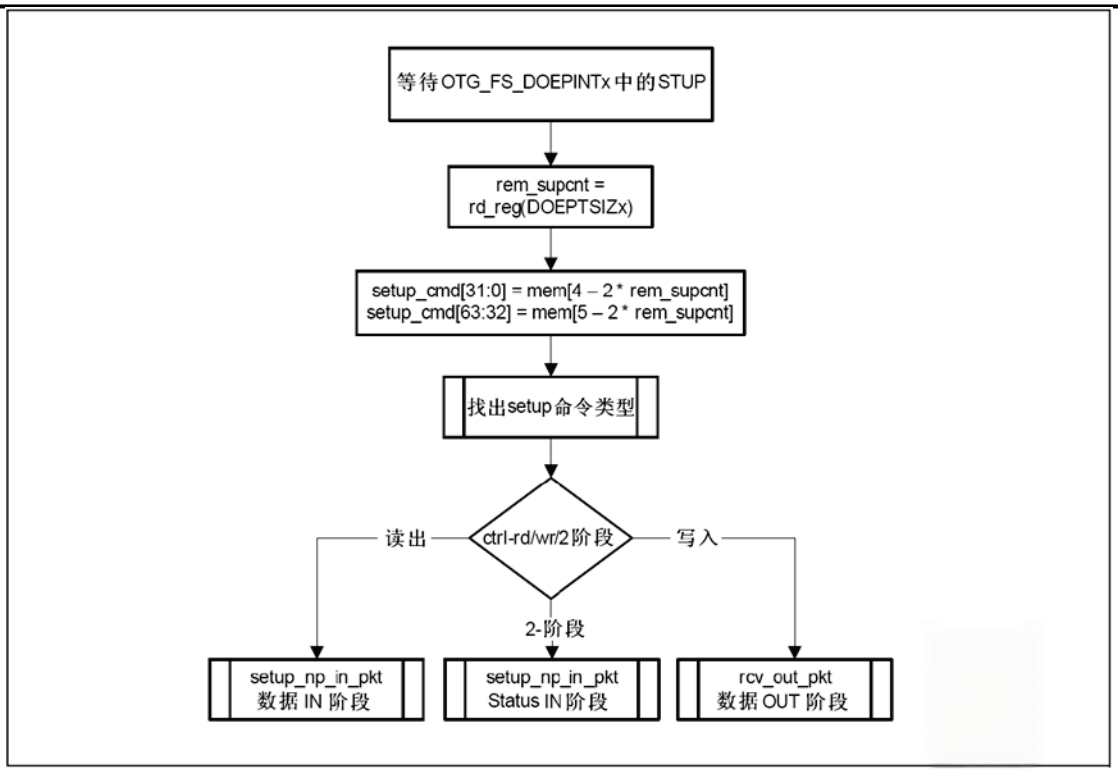


图 291 处理一个 SETUP 数据报文

- 处理多于 3 个连续的 SETUP 包

根据 USB2.0 规范，通常如果出现了错误的 SETUP 包，主机不能向同一个端口发送超过 3 个连续的 SETUP 包。然而，USB2.0 规范并没有限制主机发向同一个端口连续的 SETUP 包的数量。当发生此种情况时，OTG_FS 控制器将产生一个 B2BSTUP 中断(OTG_FS_DOEPINTx)。

- 设置全局的 OUTNAK

内部数据流：

1. 当应用程序设置了一个全局的 OUTNAK 时(OTG_FS_DCTL 寄存器的 SGONAK 位)，除了 SETUP 包，控制器将停止向接收 FIFO 中写入任何数据。不管接收 FIFO 是否有剩余空间，非同步的 OUT 端点将收到 NAK 握手信号，控制器将丢弃同步的 OUT 数据包。
2. 控制器将写全局 OUTNAK 信息到接收 FIFO 中，应用程序需要保证接收 FIFO 中有足够的空间写此信息。
3. 当从接收 FIFO 中读出 DWORD 类型的全局 OUTNAK 信息时，控制器将产生一个 GONAKEFF 中断(OTG_FS_GINTSTS)。
4. 一旦检测到此中断，就可以判断控制器已处于全局 OUTNAK 状态。应用程序可以通过清除 OTG_FS_DCTL 寄存器的 SGONAK 位来清除这个中断。

应用程序处理流程

1. 应用程序需要设置全局 OUTNAK 位，使控制器停止接收任何数据到接收 FIFO 中。
   - OTG_FS_CTL 寄存器的 SGONAK = 1
2. 等待 GONAKEFF 中断(OTG_FS_GINTSTS)，此中断指示控制器将停止接收除了 SETUP 包以外的任何数据。
3. 在设置 OTG_FS_DCTL 寄存器的 SGONAK 位之后，在控制器产生 GONAKEFF 中断之前，应用程序可以接收到有效的 OUT 包。
4. 可以通过写 GINTMSK 寄存器的 GINAKEFFM 位来暂时屏蔽此中断。
   - GINTMSK 寄存器的 GINAKEFFM = 0

5. 一旦需要退出全局 OUTNAK 模式，可以清除 OTG_FS_DCTL 寄存器的 SGONAK 位。此操作也将清除 GONAKEFF(OTG_FS_GINTSTS)中断。
   - OTG_FS_DCTL 寄存器的 CGONAK 位 = 1
6. 如果已经屏蔽了此中断，可以通过以下操作使能中断。
   - GINTMSK 寄存器的 GINAKEFFM = 1

- 中止一个 OUT 端点

需要使用以下流程来中止一个已使能的 OUT 端点：

1. 在中止一个 OUT 端点前，应用程序需要使能全局 OUTNAK 状态。
   - OTG_FS_DCTL 寄存器的 SGONAK = 1
2. 等待 GONAKEFF 中断(OTG_FS_GINTSTS)
3. 对以下位进行编程，中止 OUT 端点
   - OTG_FS_DOEPCTLx 寄存器的 EPDIS=1
   - OTG_FS_DOEPCTLx 寄存器的 SNAK = 1
4. 等待 EPDISD 中断(OTG_FS_DOEPINTx)，此中断指示 OUT 端点已被成功中止。当产生 EPDISD 中断时，控制器将同时清除以下位：
   - OTG_FS_DOEPCTLx 寄存器的 EPDIS=0
   - OTG_FS_DOEPCTLx 寄存器的 EPENA = 0
5. 需要清除全局 OUTNAK 状态，使其他没有被中止的 OUT 端点能正常接收数据。
   - OTG_FS_DCTL 寄存器的 SGONAK = 0

- 普通的非同步 OUT 数据传输

本节描述了一个标准的非同步 OUT 数据的传输过程(控制、块或中断传输)。

对应用程序的要求：

1. 在使能一个 OUT 传输前，需要在存储区中分配一个缓存区用于保存 OUT 传输中收到的所有数据。
2. 对于 OUT 传输，端点的传输长度寄存器中的传输长度位，需要被设置成端点的最大数据包长度的倍数，以 DWORD 类型对齐。
   - 传输长度[EPNUM] = n×(MPSIZ[EPNUM] + 4 - (MPSIZ[EPNUM]模 4))
   - 包数[EPNUM] = n
   - n>0
3. 在每个 OUT 端点中断中，都需要读出端点的传输长度寄存器，以便计算缓存区中的数据长度。收到的数据长度可以比设置的传输长度小。
   - 缓存区中的数据长度 = 初始配置的传输长度 - 控制器更新的传输长度。
   - 接收到的 USB 包数目 = 初始配置的包数目 - 控制器更新的包数目

内部数据流：

1. 应用程序需要配置相关端点寄存器的传输长度位和包数目位，清除 NAK 位，并使能端点以便接收数据。
2. 一旦 NAK 位被清除，控制器将开始接收数据，并在接收 FIFO 有剩余空间时，将数据写入接收 FIFO 中。对于每一个收到的数据包，数据包和其状态信息都将写入接收 FIFO 中。每写入接收 FIFO 一个数据包(最大包长度的包或短包)，端点寄存器的包数目位都将自减 1。
   - 一个带有错误的 CRC 的 OUT 数据包，将被从接收 FIFO 中自动清除。
   - 在发送一个 ACK 信号后，对于因为主机没有收到 ACK 信号时而重新发送的非同步

OUT 数据包，将被控制器丢弃。应用程序不用处理对同一端点的带有同样数据 PID 号的连续 OUT 数据包，这种情况下寄存器的包数位不会自减。

- 如果接收 FIFO 没有剩余空间，同步或非同步的数据包都将被丢弃，并不会被写入接 FIFO。另外，对于非同步的 OUT 数据包，将发送 NAK 握手信号。
- 对于以上三种情况，由于都没有实际的数据写入接收 FIFO 中，端点寄存器的包数目位都不会自减。

3. 当包数目自减到 0，或者端点收到一个短的数据包，控制器会将该端点的状态设置为 NAK。一旦端点处于 NAK 状态，同步和非同步传输的数据包都将被丢弃，而不会写入接收 FIFO 中，同时，对于非同步的 OUT 传输，控制器将返回一个 NAK 的握手包。
4. 在数据被控制器写入接收 FIFO 后，应用程序可以从接收 FIFO 中读取数据，并将之写入其他存储区，每个端点一次只能操作一个包的数据。
5. 每当一个包的数据通过 AHB 总线写入其他存储区后，端点的传输长度寄存器的值都会自动减去已写入的数据长度。
6. OUT 端点的 OUT 传输完成标志会在以下情况时写入接收 FIFO 中：
   - 传输长度寄存器的值为 0，同时包数寄存器的值也为 0。
   - 写入接收 FIFO 的最后一个 OUT 数据包是一个短包(0≤收到的包长度<最大包长度)
7. 当应用程序将此标志(OUT 传输完成标志)从接收 FIFO 中读出后，控制器将产生该端点的传输完成中断，同时，该端点的使能标志被清除。

应用程序处理流程：

1. 配置 OTG_FS_DOEPTSIZx 寄存器，设置传输长度和合适的包数目。
2. 配置 OTG_FS_DOEPCTLx 寄存器，设置端点的特性，同时设置 EPENA 和 CNAK 位为'1'。
   - OTG_FS_DOEPCTLx 寄存器的 EPENA=1
   - OTG_FS_DOEPCTLx 寄存器的 CNAK = 1
3. 等待 RXFLVL(OTG_FS_GINTSTS)中断，并从接收 FIFO 中读出数据包。
   - 根据不同的传输长度，此步骤将重复多次。
4. 等待非同步 OUT 传输正常完成标志的 XFRC(OTG_FS_DOEPINTx)中断。
5. 读 OTG_FS_DOEPTSIZx 寄存器，获得实际接收到的数据长度的信息。

- 普通的同步 OUT 数据传输

本章描述了标准的同步 OUT 传输处理过程。

对应用程序的要求：

1. 所有对于非同步 OUT 传输的要求都适用于同步 OUT 传输。
2. 对于同步 OUT 传输，控制寄存器的传输长度位和包数目位都必须设置为在一个帧内接收的最大数据包长度和数量。同步 OUT 传输不能跨越一个帧。
3. 应用程序需要在一个周期性的帧(OTG_FS_GINTSTS 寄存器的 EOPF 中断)结束前，从接收 FIFO 中读出所有的同步 OUT 数据包(包括数据和状态)。
4. 为准备接收下一个帧的数据，同步 OUT 端点需要在 EOPF(OTG_FS_GINTSTS)后，SOF(OTG_FS_GINTSTS)前被使能。

内部数据流：

1. 除了一些细微的差别，同步 OUT 端点的内部数据流与非同步 OUT 端点的内部数据流大致相同。

2. 在同步 OUT 端点被使能(通过设置端点使能位)，同时 NAK 状态被清除时，应用程序必须设置合适的奇数/偶数帧位。控制器只有在以下情况下，同步 OUT 端点才会在特殊的帧接收数据：
   — EONUM(OTG_FS_DOEPCTLx 寄存器) = SOFFN[0](OTG_FS_DSTS 寄存器)
3. 在从接收 FIFO 中读取完整的同步 OUT 数据包(包括数据和状态)后，控制器会根据从接收 FIFO 中读取的最后一个同步 OUT 数据包来更新 OTG_FS_DOEPTSIZx 寄存器的 RXDPID 位。

应用程序操作流程：

1. 配置 OTG_FS_DOEPTSIZx 寄存器，设置合适的传输长度和包数。
2. 配置 OTG_FS_DOEPCTLx 寄存器，设置端点的特性并使能端点，同时清除 NAK 状态，选择合适的奇数/偶数帧。
   — EPENA=1
   — CNAK=1
   — EONUM=(0：偶数/1：奇数)
3. 等待 RXFLVL 中断(OTG_FS_GINTSTS)，并从接收 FIFO 中读取数据包
   — 根据不同的传输长度，本步骤将重复多次。
4. 等待同步 OUT 传输结束标志产生的 XFRC 中断(OTG_FS_DOEPINTx)。此中断并不意味着存储区中的数据是正确的。
5. 在同步传输中，并不是每次都会产生 XFRC 中断，但能够检测到 OTG_FS_GINTSTS 寄存器的 IISOOXFRM 中断。
6. 读 OTG_FS_DOEPTSIZx 寄存器，以便知道在一个帧内收到了多少有效数据。只有在满足以下任一条件之一时，才能判定收到的数据是有效的。
   在这个帧内收到的 USB 包数目 = 初始化的包数目 - 控制器更新的最终包数目。
   — RXDPID=D0(OTG_FS_DOEPTSIZx)，同时在这个帧内收到的 USB 包数为 1。
   — RXDPID=D1(OTG_FS_DOEPTSIZx)，同时在这个帧内收到的 USB 包数为 2。
   — RXDPID=D2(OTG_FS_DOEPTSIZx)，同时在这个帧内收到的 USB 包数为 3。在这个帧内收到的 USB 包数目 = 初始化的包数目 - 控制器更新的最终包数目。应用程序可以丢弃无效的数据包。

- 不完全的同步 OUT 数据传输

本节描述了同步 OUT 数据包接收不完全时的处理流程。

内部数据流：

1. 对于同步 OUT 端点，XFRC 中断(OTG_FS_DOEPINTx)不是每次都会产生的。在以下情况下，控制器会停止接收同步 OUT 数据包，应用程序将不会检测到 XFRC 中断(OTG_FS_DOEPINTx)。
   — 当接收 FIFO 没有足够空间存放完整的同步 OUT 数据包时，控制器将停止接收。
   — 当收到的同步 OUT 数据包发生了 CRC 错误。
   — 当收到的同步 OUT 令牌是错误的。
   — 当应用程序从接收 FIFO 中读取数据的速度过慢。
2. 当控制器在完成所有的同步 OUT 传输前检测到了帧结束信号，会产生一个不完全的同步 OUT 中断(OTG_FS_GINTSTS 寄存器的 IISOOXFRM)，此中断表示至少有一个同步 OUT 端点没有产生 XFRC 中断(OTG_FS_DOEPINTx)。这种情况下，该端点仍然有效，但在 USB 线上不会有进一步的数据传输。

应用程序处理流程：

1. IISOOXFRM 中断(OTG_FS_GINTSTS)指示在当前帧内至少有一个同步 OUT 端点发生了不完全传输的情况。
2. 应用程序需要判断是否由于没有读空接收 FIFO 才导致发生了不完全传输。应该确保在进一步操作前必须读出接收 FIFO 中所有的同步 OUT 数据(包括数据和状态)。
   — 当应用程序读空接收 FIFO 后，会产生 XFRC 中断(OTG_FS_DOEPINTx)，此时，需要重新使能端点，以便在下一帧时接收同步 OUT 数据。
3. 当检测到 IISOOXFRM 中断(OTG_FS_GINTSTS)，需要读出所有同步 OUT 端点的控制寄存器(OTG_FS_DOEPCTLx)以便确认在当前帧内哪个端点发生了不完全传输。在同时满足以下条件时，表示传输不完整。
   — EONUM 位(OTG_FS_DOEPCTLx)=SOFFN[0](OTG_FS_DSTS)
   — EPENA=1(OTG_FS_DOEPCTLx)
4. 以上的操作必需在检测到 SOF 中断(OTG_FS_GINTSTS)之前完成，以确保当前的帧号没有改变。
5. 对于一个发生了不完全传输的同步 OUT 端点，必需丢弃储存在存储区的数据，并通过设置 OTG_FS_DOEPCTLx 寄存器的 EPDIS 位来无效该端点。
6. 等待 EPDIS 中断(OTG_FS_DOEPINTx)，并重新使能端点，以便在下一个帧开始接收新的数据。
   — 由于控制器需要一定的时间来关闭端点，因此在收到一个错误的同步数据之后，应用程序可能在下一个帧时不能接收数据。

- 中止一个非同步 OUT 端点

本节描述了应用程序如何中止一个非同步端点。

1. 设置控制器进入全局 OUTNAK 响应状态。
2. 无效相应的端点。
   — 通过设置 STALL=1(OTG-FS_DOEPCTL)，而不是设置 SNAK 位(OTG_FS_DOEPCTL)来使端点无效。
3. 当应用程序准备好结束以 STALL 来响应端点时，必需清除 STALL 位(OTG_FS_DOEPCTLx)。
4. 当应用程序收到 SetFeature.Endpoint Halt 命令或者 ClearFeature.Endpoint Halt 命令时，需要设置或者清除端点的 STALL 位，这个设置或者清除的操作必需在控制端点发起状态阶段传输之前完成。

示例

本节列举了一些例子，来说明如何处理基本的传输类型和情况。

- 设备模式下块 OUT 传输

下图给出了从 USB 接收一个单独的块 OUT 数据包到 AHB 的流程以及相关的事件。

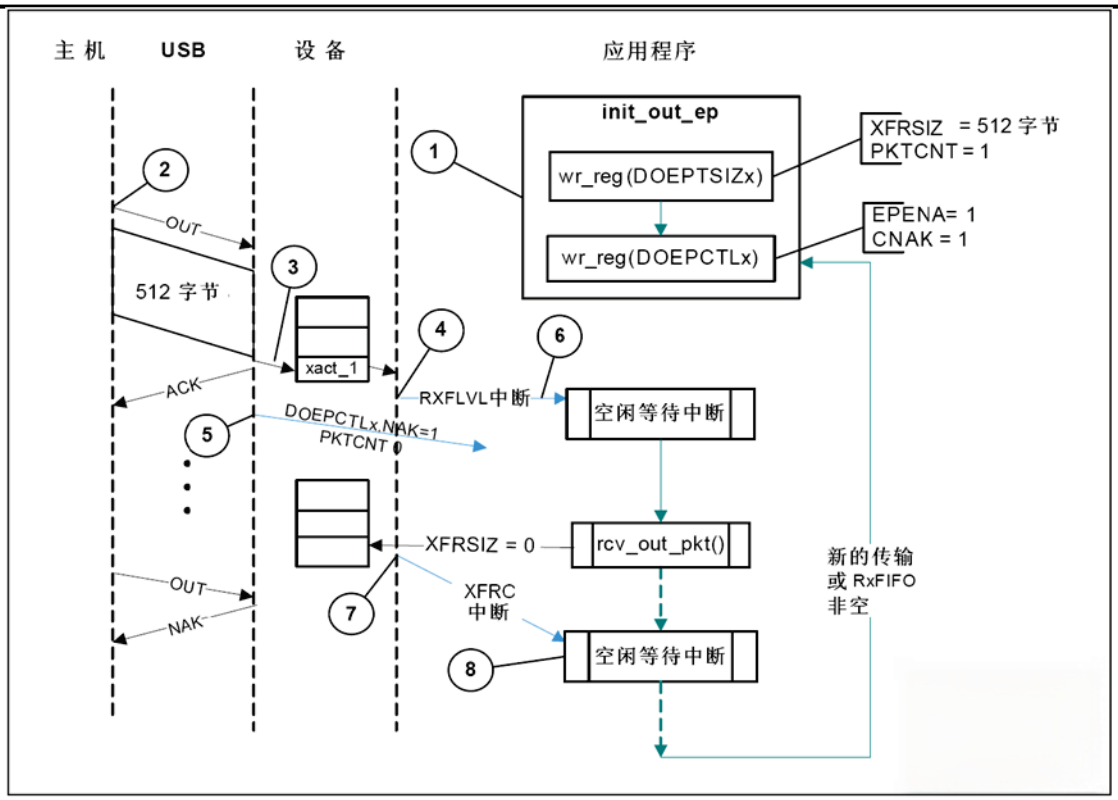


图 292 从模式下块(Bulk)OUT 传输过程

在收到 SetConfiguration/SetInterface 命令后，应用程序通过设置(OTG_FS_DOEPCTLx 中)CNAK=1 和 EPENA=1 来初始化所有的 OUT 端点，并为 OTG_FS_DOEPTSIZx 寄存器设置合理的 XFRSIZ 和 PKTCNT 值。

1. 主机尝试向一个端点发送数据(OUT 令牌)
2. 当接收到 USB 线上的 OUT 令牌，控制器将数据包储存在接收 FIFO 中，此时接收 FIFO 应该有剩余空间。
3. 当控制器将完整的数据包写入接收 FIFO 后，将产生 RXFLVL 中断(OTG_FS_GINTSTS)。
4. 在接收到 PKTCNT 个 USB 数据包之后，控制器将自动设置端点状态的 NAK 位，以防止持续接收更多的数据包。
5. 应用程序需要响应中断，并从接收 FIFO 中读取数据。
6. 当读出了所有的数据(数据长度等于 XFRSIZ)，控制器将产生(OTG_FS_DOEPINTx)XFRC 中断。
7. 应用程序需要响应中断，并通过 XFRC 中断位的来决定整个传输是否已完成。

IN 数据传输

本节叙述了当使能了相应的发送 FIFO 时，如何在设备模式下写数据包到端点的 FIFO 中。

1. 可以选择使用轮询方式或者中断方式
   - 使用轮询方式，需要通过读 OTG_FS_DTXFSTSx 寄存器来监视端点相应的发送 FIFO 状态，以便了解发送 FIFO 是否还有剩余空间。
   - 使用中断方式，需要等待 TXFE 中断(OTG_FS_DIEPINTx)并读 OTG_FS_DTXFSTSx 寄存器来了解发送 FIFO 是否有足够的剩余空间。
   - 如果要写一个单独的非零长度的数据包，发送 FIFO 的剩余空间必需要足够写入整个完整的数据包。
   - 如果要写一个零长度的数据包，则无需查看发送 FIFO 的剩余空间。

2. 无论使用以上哪种方式，当发送 FIFO 有足够剩余空间来写入一个数据包，应用程序需要先写端点控制寄存器，再写数据包到发送 FIFO。通常，除了设置端点的使能位，应用程序都必需通过读-修改-写的操作来设置 OTG_FS_DIEPCTLx 寄存器，以防止寄存器的内容被修改。

如果发送FIFO的剩余空间足够大，可以向同一个端点连续写入多个数据包。对于周期性IN端点，在一个帧内只能写入一个数据包，只有在收到前一个数据包的传输完成标志后，才能写入下一个周期要发送的数据。

- 设置 IN 端点 NAK 状态

内部数据流程：

1. 当设置某个具体的 IN 端点状态为 NAK 时，控制器将无视该端点的发送 FIFO 中是否有要传送的数据，而停止该端点的数据传送，
2. 对于非同步 IN 命令，将以 NAK 握手信号回复。
   - 对于同步 IN 命令，将以零长度的数据包回复。
3. 对于 OTG_FS_DIEPCTLx 寄存器的 SNAK 位，控制器将产生 INEPNE(IN 端点 NAK 有效)中断(OTG_FS_DIEPINTx)。
4. 当接收到此中断，就可以判断该端点已进入 IN 端点 NAK 状态。应用程序可以通过设置 OTG_FS_DIEPCTLx 寄存器的 CNAK 位来清除此中断。

应用程序编程流程：

1. 需要设置 INNAK 位，来停止指定 IN 端点的数据传输活动：
   - OTG_FS_DIEPCTLx 寄存器的 SNAK=1
2. 等待 OTG_FS_DIEPCTLx 寄存器的 INEPNE 中断。此中断表示控制器已停止指定端点的数据传输活动。
3. 控制器在应用程序设置 NAK 位之后，在产生 NAK 有效的中断之前，可以在该端点上发送有效的 IN 数据包。
4. 应用程序可以通过写 DIEPMSK 寄存器的 INEPNEM 位来临时屏蔽此中断。
   - DIEPMSK 寄存器的 INEPNEM=0
5. 通过清除 OTG_FS_DIEPCTLx 寄存器的 NAK 状态位(NAKSTS)来退出端点 NAK 状态模式。同时需要清除 INEPNE 中断(OTG_FS_DIEPINTx)。
   - OTG_FS_DIEPCTLx 寄存器的 CNAK=1
6. 如果应用程序在之前屏蔽了此中断，需要解除屏蔽：
   - DIEPMSK 寄存器的 INEPNEM=1

- IN 端点无效

通过以下步骤，可以使一个已经使能的 IN 端点无效。

应用程序编程流程：

1. 应用程序在无效一个 IN 端点之前，需要先停止从 AHB 写数据包。
2. 应用程序需要设置端点进入 NAK 模式
   - OTG_FS_DIEPCTLx 寄存器的 SNAK=1
3. 等待 OTG_FS_DIEPINTx 寄存器的 INEPNE 中断。
4. 对于需要无效的端点，设置 OTG_FS_DIEPCTLx 寄存器的以下位：
   - OTG_FS_DIEPCTLx 寄存器的 EPDIS=1
   - OTG_FS_DIEPCTLx 寄存器的 SNAK=1

5. 等待 OTG_FS_DIEPINTx 寄存器的 EPDISD 中断，该中断表示控制器已经使指定的端点无效。伴随着该中断，控制器也将清除以下位：
   - OTG_FS_DIEPCTLx 寄存器的 EPENA=0
   - OTG_FS_DIEPCTLx 寄存器的 EPDIS=0
6. 应用程序必需读取周期性 IN 端点的 OTG_FS_DIEPTSIZx 寄存器，以了解该端点已经向 USB 总线发送了多少数据。
7. 应用程序需要通过设置 OTG_FS_GRSTCTL 寄存器的以下位，来清除该端点的发送 FIFO 中的数据：
   - TXFNUM(OTG_FS_GRSTCTL)=端点的发送 FIFO 编号
   - TXFFLSH(OTG_FS_GRSTCTL)=1

应用程序需要查询 OTG_FS_GRSTCTL 寄存器，直到 TXFFLSH 位被控制器清除，这表示刷新操作已经完成。应用程序可以随后重新使能该端点，来发送新的数据。

- 普通的非周期性 IN 数据传输

应用程序需要：

1. 在发起一个 IN 传输前，需要确保所有要通过 IN 传输被发送的数据，都属于同一个缓冲区。
2. 对于 IN 传输，端点传输长度寄存器的传输长度位指示整个要发送的数据长度，包括若干个最大包长度的数据包和一个短包。短包将在传输的最后进行传送。
   - 为了要先发送若干个最大数据包长度的数据包，并在传输的最后发送一个短包：
     传输长度[EPNUM]=x×MPSIZ[EPNUM]+sp
     If(sp>0)
       包数目[EPNUM]=x+1
     else
       包数目[EPNUM]=x
   - 为了要发送一个单独的零长度的数据包：
     传输长度[EPNUM]=0
     包数目[EPNUM]=1
   - 为了要先发送若干个最大数据包长度的数据包，并在传输最后发送一个零长度的数据包，应用程序需要把整个发送过程分割为两部分。第一个部分发送若干个最大数据包长度的数据包，第二个部分再单独发送长度为零的数据包。
     第一部分：传输长度[EPNUM]=x×MPSIZ[EPNUM]；包数目[PENUM]=n
     第二部分：传输　长度[EPNUM]=0；包数目[EPNUM]=1
3. 一旦一个端点开始发送数据，控制器会更新传输长度寄存器的值，在 IN 传输结束时，需要读出传输长度寄存器，以了解发送 FIFO 中有多少数据已通过 USB 总线发出。
4. 仍然留在发送 FIFO 中的数据 = 设置的传输长度 - 控制器更新过的最终传输长度。
   - USB 总线上传输的数据 = (设置的初始包数目 - 控制器更新过的最终包数目)×MPSIZ[EPNUM]
   - 仍然需要发送的数据 = 设置的初始传输长度 - 已通过 USB 发送的数据长度。

内部数据流程：

1. 应用程序需要设置端点控制寄存器的传输长度和包数目位，并使能需要传输数据的端点。
2. 应用程序需要将要传输的数据，写入相应端点的发送 FIFO 中。
3. 每当应用程序写一个数据包到发送 FIFO 中，控制器将自动从传输长度中减去写入的包长度。需要持续写入数据直到该端点的传输长度变为 0。在写数据到 FIFO 后，FIFO 中的数据包数目

位(每个 IN 端点都用 3 位表示包数目，在内部由控制器管理，在任何时候，对于 IN 端点控制器能管理的最大包数为 8)将自动加 1。对于零长度的数据包，会有相应的标志位指明，FIFO 中不需要有任何数据。

4. 在数据被写入发送 FIFO 后，控制器会在收到 IN 命令后读取这些数据。对于每个以 ACK 结束的非同步 IN 数据包传输，该端点的包数目寄存器将自减 1，直到包数目变为 0。包数目寄存器将不会因为超时而自减。
5. 对于零长度数据包(由内部的零长度标志位指示)，控制器将根据 IN 命令发送零长度的数据包，并且包数寄存器自减 1。
6. 如果在包数寄存器已经变为 0，并且 FIFO 中已无要发送的数据时，收到了 IN 命令，控制器将产生”在发送 FIFO 为空时收到了 IN 命令”的中断(ITTXFE)，同时不设置该端点的 NAK 标志位。控制器将以 NAK 握手信号来回应一个非同步端点的传输。
7. 控制器内部复原 FIFO 的指针并且不会产生超时的中断。
8. 如果传输长度为 0，并且包数也为 0，传输完成中断(XFRC)会产生，并且端点的有效标志将被清除。

应用程序流程:

1. 根据传输长度和相应的包数目设置 OTG_FS_DIEPTSIZx 寄存器。
2. 根据端点的特性设置 OTG_FS_DIEPCTLx 寄存器，并设置 CNAK 和 EPENA(端点使能)位。
3. 当传输非零长度的数据包时，应用程序需要查询 OTG_FS_DTXFSTSx 寄存器(此处 x 代表与端点相关的 FIFO 编号)来了解发送 FIFO 是否有剩余空间。应用程序也可以使用 TXFE(OTG_FS_DIEPINTx)中断来决定是否写数据到 FIFO。

- 普通的周期性 IN 传输

本节叙述了一个典型的周期性 IN 传输

应用程序需要:

1. 上一节所描述的普通的非周期性 IN 传输的应用程序需求 1、2、3 和 4 同样适用于本节的周期性 IN 传输，除了 2 有些许不同。
   - 应用程序可以传输多个最大数据包长度的数据包，或传输带有一个短包结尾的多个最大数据包长度的数据包。传输若干个最大数据包长度的数据包，并在传输的最后发送一个短包需要做到:

     传输长度[EPNUM]=x×MPSIZ[EPNUM]+sp

     (此处 x 是≥0 的整数，并且 0≤sp≤MPSIZ[EPNUM])

     If(sp>0)

       包数目[EPNUM]=x+1

     else

       包数目[EPNUM]=x

     MCNT[EPNUM]=包数目[EPNUM]
   - 应用程序不能在传输末尾发送一个零长度的数据包，但可以自动发送一个单独的零长度的数据包。为了要发送一个单独的零长度的数据包，需要设置:

     传输长度[EPNUM]=0 包数目[EPNUM]=1

     MCNT[EPNUM]=包数目[EPNUM]
2. 应用程序在一个时间只能安排一个帧的数据传输。
   - (MCNT-1)×MPSIZ≤XFERSIZ≤MCNT≤MPSIZ
   - PKTCNT=MCNT(OTG_FS_DIEPTSIZx)

- 如果 XFERSIZ<MCNT×MPSIZ，那么传输的最后一个数据包是一个短包。

注意： *MCNT 位在 OTG_FS_DIEPTSIZx 寄存器中，MPSIZ 位在 OTG_FS_DIEPCTLx 寄存器中，PKTCNT 位在 OTG_FS_DIEPTSIZx 寄存器中，XFERSIZ 位在 OTG_FS_DIEPTSIZx 寄存器中。*

3. 所有在一个帧内需要发送的数据都必需在收到 IN 命令之前写入到发送 FIFO 中。需要在一个帧内发送的数据，即使只有一个 DWORD 没有写入发送 FIFO 中，控制器仍然会认为 FIFO 是空的。当发送 FIFO 为空时：
   - 对于同步 IN 端点，会发送一个零长度的数据包。
   - 对于中断 IN 端点，会发送 NAK 握手信号。
4. 对于一个帧内有三个数据包的高带宽 IN 端点，应用程序需要设置端点 FIFO 的长度为 2×max_pkt_size(最大数据包长度)，并在第一个数据包已经发送到 USB 总线后，写入第三个数据包。

内部数据流程：

1. 应用程序需要设置相应端点寄存器的传输长度和包数目位，并使能该传输端点。
2. 应用程序需要将数据写入相应的发送 FIFO 中。
3. 每当应用程序写入一个数据包到发送 FIFO，该端点的传输长度寄存器值将自动减去写入的数据包长度。应用程序需要持续的写入数据直到该端点的传输长度变为 0.
4. 当周期性端点收到 IN 命令时，控制器将自动的发送 FIFO 中的数据。如果 FIFO 中没有当前帧的完整数据包(在指定 FIFO 模式下有完整的包)，控制器会产生”收到 IN 命令时发送 FIFO 为空”的中断。
   - 对于同步 IN 端点，控制器会发送零长度的数据包
   - 对于中断 IN 端点，控制器会发送 NAK 的握手响应
5. 相应端点的包数目寄存器值会在以下情况时，自动减 1：
   - 对于同步端点，当发送了一个零长度或非零长度的数据包
   - 对于中断端点，当一个 ACK 握手信号被传输
   - 当发送长度和包数目都为 0 时，会产生传输结束中断，相应端点的使能会被清除。
6. 在”周期性帧间隔时间”(由 OTG_FS_DCFG 寄存器的 PFIVL 位控制)内，当控制器发现任一在当前帧内应该为空的同步 IN 端点的 FIFO 为非空，会产生 OTG_FS_GINTSTS 寄存器的 IISOIXFR 中断。

应用程序流程：

1. 根据端点特性配置 OTG_FS_DIEPCTLx 寄存器，并设置 CNAK 和 EPENA 位。
2. 将需要在下一个帧时发送的数据写入发送 FIFO 中。
3. 如果产生了 ITTXFE 中断(OTG_FS_DIEPINTx)，表示应用程序没有来得及将要发送的数据写入发送 FIFO 中。
4. 当产生中断的端点在中断产生前已经使能，不用理会此中断；否则，使能该端点，控制器将在收到下一个 IN 命令时发送数据。
5. 当产生 XFRC 中断(OTG_FS_DIEPINTx)，并且没有 ITTXFE 中断(OTG_FS_DIEPINTx)，表示同步 IN 传输已经正常结束。可以读 OTG_FS_DIEPTSIZx 寄存器，传输长度寄存器值和包数目寄存器值都等于 0，指示所有的数据都已通过 USB 传送。
6. 当产生 XFRC 中断(OTG_FS_DIEPINTx)，不管有没有 ITTXFE 中断(OTG-FS_DIEPINTx)，都表示中断 IN 传输已经正常结束。可以读 OTG-FS_DIEPTSIZx 寄存器，传输长度和包数寄存器值都等于 0，指示所有的数据都已通过 USB 传送。

7. 当产生未完成的同步 IN 传输中断(OTG_FS_GINTSTS 的 IISOIXFR)，而没有产生前述的中断，表示在当前帧，控制器少收到至少一个周期性 IN 命令。

- 未完成的同步 IN 数据传输

本节叙述了当发生未完成的同步 IN 传输时，应用程序的处理流程。内部数据流程：

1. 出现以下任一情况时会认为同步 IN 传输未完成：
   - 控制器在至少一个同步 IN 端点上，收到一个错误的同步 IN 命令。此时，应用程序会检测到一个”未完成的同步 IN 传输”中断(OTG_FS_GINTSTS 寄存器的 IISOIXFR)。
   - 应用程序来不及将完整的数据写入发送 FIFO 中，也就是说在写完整的数据包到发送 FIFO 之前，收到了 IN 命令。此时，应用程序会检测到”收到 IN 命令时发送 FIFO 为空”中断(OTG_FS_DIEPINTx)。应用程序可以不理会此中断，因为这将在周期性帧结束的时候，导致一个”未完成的同步 IN 传输”中断(OTG_FS_GINTSTS 的 IISOIXFR)。控制器将发送一个零长度的数据包到 USB 总线来响应收到的 IN 命令。
2. 应用程序应该尽快停止继续向发送 FIFO 写数据。
3. 应用程序应该设置端点的 NAK 位和取消使能位。
4. 控制器会解除端点的使能，清除端点的取消使能位，并产生该端点的”端点取消使能”中断。

应用程序流程：

1. 应用程序在任何同步 IN 端点上收到 OTG_FS_DIEPINTx 的发送 FIFO 空中断时，都可以忽略接收到的 IN 命令，因为这将导致一个未完全的同步 IN 传输中断(OTG_FS_GINTSTS)。
2. “未完成的同步 IN 传输”中断(OTG_FS_GINTSTS)表示至少有一个同步 IN 端点发生了未完成的同步 IN 传输。
3. 应用程序需要读所有同步 IN 端点的端点控制寄存器，以了解是哪个端点发生了未完成的 IN 传输事件。
4. 应用程序必需停止继续向这个端点的发送 FIFO 写数据。
5. 配置 OTG_FS_DIEPCTLx 寄存器的以下位来取消端点的使能：
   - OTG_FS_DIEPCTLx 寄存器的 SNAK=1
   - OTG_FS_DIEPCTLx 寄存器的 EPDIS=1
6. OTG_FS_DIEPINTx 寄存器的”端点取消使能”中断指示控制器取消端点的使能状态。
   - 此时，应用程序需要清除相应的发送 FIFO，或者在重新使能端点之后，覆盖仍存留在 FIFO 中的数据。为了清除数据，应用程序必需使用 OTG_FS_GRSTCTL 寄存器。

- 中止非同步 IN 端点

本节叙述了应用程序如何中止一个非同步端点

应用程序流程：

1. 解除需要中止的 IN 端点的使能状态。设置 STALL 位。
2. 如果端点已经使能，设置 OTG_FS_DIEPCTLx 寄存器的 EPDIS = 1.
   - OTG_FS_DIEPCTLx 的 STALL=1。
   - STALL 位的优先级用于高于 NAK 位。
3. 端点”取消使能中断”(OTG_FS_DIEPINTx)通知应用程序，控制器已经取消了相应端点的使能。
4. 应用程序需要根据端点的传输类型，清除非周期性或周期性发送 FIFO。对于非周期性端点，应用程序必需重新使能另一个不需要中止的非周期性端点，来完成数据传输。
5. 当应用程序准备好结束以 STALL 握手信号来响应端点，必需清除 OTG_FS_DIEPCTLx 寄存器的 STALL 位。

6. 当应用程序根据 SetFeature.Endpoint Halt 命令，或者 ClearFeature.Endpoint Halt 命令来设置或清除 STALL 位，必需在控制端点上建立状态传输阶段之前执行对 STALL 位的设置和清除操作。

特殊情况：中止一个控制 OUT 端点

控制器在主机发送超过 SETUP 包中定义的数量的 IN/OUT 命令时，可以在控制传输的数据传输阶段 STALLIN/OUT 命令。在这种情况下，应用程序必需在控制传输的数据传输阶段，使能 OTG_FS_DIEPINTx 的 ITTXFE 中断和 OTG_FS_DOEPINTx 的 OTEPDIS 中断。当应用程序收到了中断，意味着必需设置相应端点控制寄存器的 STALL 位，并清除中断。

26.17.7 最差情况下的响应时间

当 OTG_FS 控制器工作在设备模式下时，对于任何一个跟在同步 OUT 传输后的命令，都会有一个最坏的响应时间。这个最坏的响应时间根据 AHB 的时钟频率有所不同。

控制器寄存器位于 AHB 时钟域的范围内，并且控制器在这些寄存器值被更新之前，不能接收新的命令。最坏的情况是：对于任何一个跟在同步 OUT 传输之后的命令，由于同步传输不需要握手信号，因此这个命令可能会马上到达。这个最坏的响应时间在 AHB 时钟等于 PHY 时钟频率时，是 7 个 PHY 时钟。当 AHB 时钟更快时，这个响应时间会变快些。

当这种最坏的情况发生时，控制器会以 NAK 来响应块传输和中断传输命令，丢弃同步和 SETUP 命令。对于 SETUP 命令，主机会以 SETUP 超时来处理这种情况，并重发 SETUP 包。对于同步传输，会产生未完全的同步 IN 传输中断(IISOIXFR)和未完全的同步 OUT 传输中断(IISOOXFR)，来通知应用程序有同步的 IN/OUT 命令被丢弃了。

选择 OTG_FS_GUSBCFG 寄存器的 TRDT 值

OTG_FS_GUSBCFG 寄存器的 TRDT 值以 PHY 时钟个数为单位，表示 MAC 从收到一个 IN 命令，到获取 FIFO 状态并从 PFC(包 FIFO 控制器)获取最初的数据，所需要的时间。这个时间包括了 PHY 和 AHB 时钟同步的延迟。最坏的情况是 AHB 时钟频率与 PHY 时钟频率相同，此时，这个延迟是 5 个时钟周期。

一旦 MAC 收到了 IN 命令，这个信息(收到命令)将由 PFC(PFC 以 AHB 时钟驱动)同步到 AHB 时钟。PFC 会从 SPRAM 读取数据，并将之写入双时钟源的缓存区。MAC 再从缓存区中读出数据(4 个深度)。

如果 AHB 时钟频率高于 PHY，可以选择一个较小的 TRDT 数值(OTG_FS_GUSBCFG)。下图中有以下信号：

- tkn_rcvd：从 MAC 发到 PFC 的命令已收到信息
- dynced_tkn_rcvd：从 PCLK 到 HCLK 的双倍的同步 tkn_rcvd 信号
- spr_read：读 SPRAM
- spr_addr：SPRAM 寻址
- spr_rdata：从 SPRAM 读数据
- srcbuf_push：送入源缓存区
- srcbuf_rdata：从源缓存区读数据。MAC 检测到数据。应用程序可以使用以下公式来计算 TRDT 的值：

4×AHB 时钟+1 个 PHY 时钟 = (2 个时钟同步+1 个时钟的存储器寻址+1 个时钟从同步 RAM 获取数据)+1 个 PHY 时钟(下一个 PHY 时钟 MAC 可以采样 2 个时钟的 FIFO 输出)。

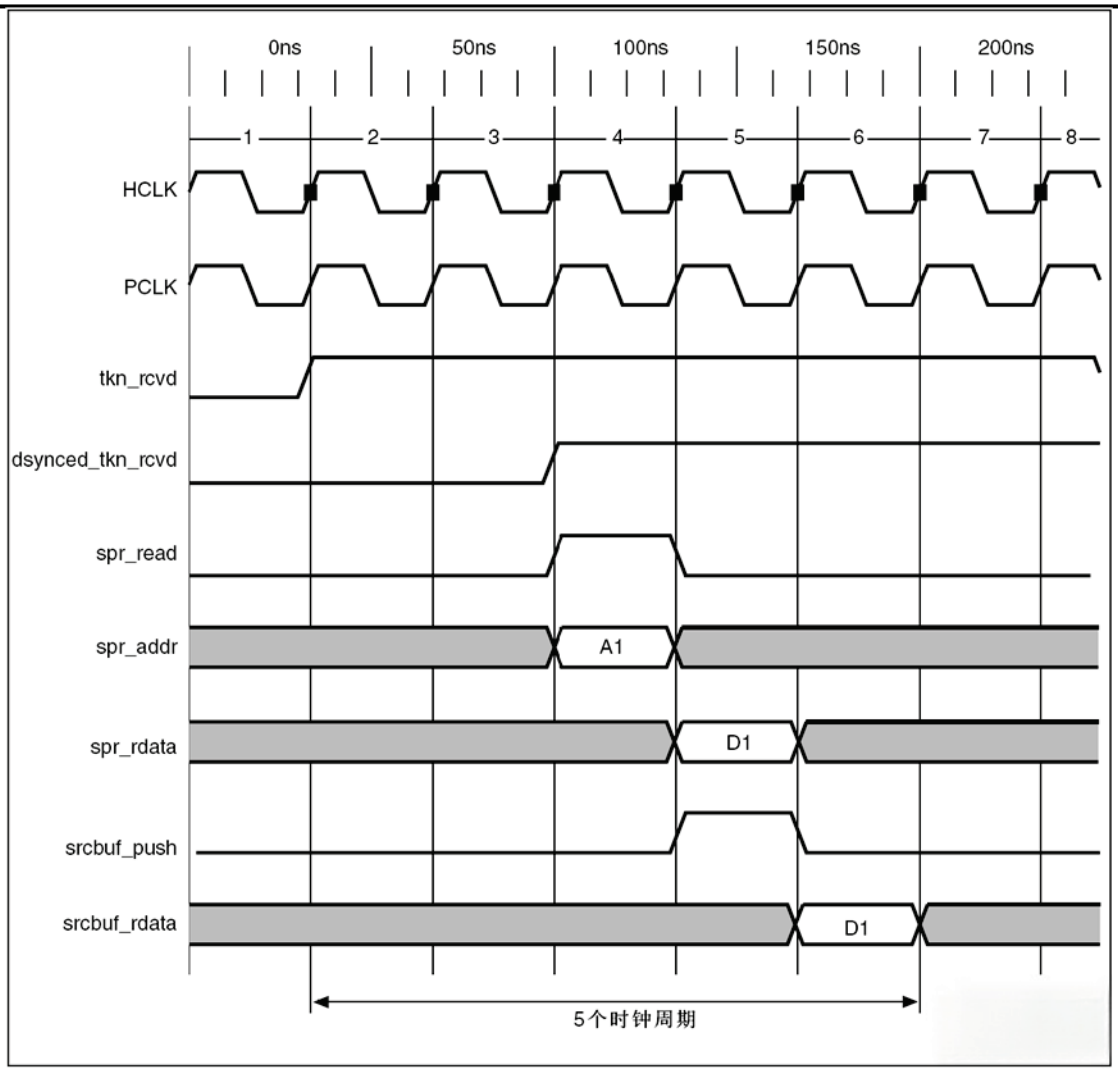


图 293 TRDT 最大时序的情况

26.17.8 OTG 编程规则

OTG_FS 控制器可用于设计一个支持 HNP 和 SRP 协议的 OTG 设备。当控制器发现插入一个 A 类插头，控制器将执行 A 类设备操作，当插入 B 类插头，则执行 B 类操作。在主机模式下，OTG_FS 控制器可以关闭 VBUS 来节省耗电。SRP 协议用于 B 类设备请求 A 类设备打开 VBUS 的供电。设备必需在控制数据线和 VBUS 线上都产生脉冲，但主机可以只识别其中一个的脉冲信号作为 SRP 信号。HNP 协议用于 B 类设备协商和切换到主机角色，协商之后，B 类设备也可以挂起总线，回到设备角色。

A 类设备的会话请求协议

应用程序必需设置控制器 USB 配置寄存器的 SRP 使能位，这会使得 OTG_FS 控制器能在 A 类设备模式下识别的 SRP 请求。

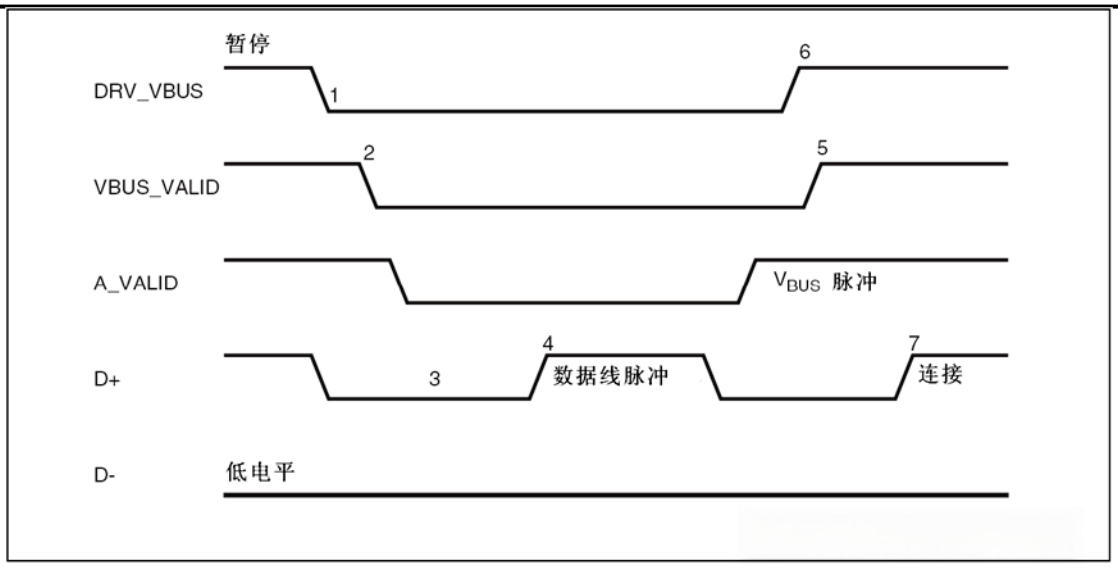


图 294 A 设备的 SRP

DRV_VBUS：送到 PHY 的 VBUS 驱动信号

VBUS_VALID：PHY 的 VBUS 有效信号

A_VALID：送到 PHY 的 A 类设备的 VBUS 电平信号

D+：正向数据线

D-：反向数据线

1. 为了节省耗电，应用程序需要在总线空闲时，设置主机端口控制和状态寄存器的端口挂起位和端口供电位，来挂起总线并关闭对端口的供电。
2. PHY 拉低 VBUS_VALID 信号来指示端口供电已关闭。
3. 当 VBUS 总线供电关闭时，设备必需检测到至少 2ms 的 SE0 信号，才能发起 SRP 请求。
4. 为了发起 SRP 请求，设备需要打开数据线的上拉电阻，持续 5 到 10ms。OTG_FS 控制器将检测数据线上的脉冲信号。
5. 设备需要驱动 VBUS，提供超过 A 类设备会话有效电平(最小 2.0V)的 VBUS 脉冲。OTG_FS 控制器将用中断通知应用程序，检测到了 SRP 请求，并设置控制器全局中断状态寄存器的会话请求检测位(OTG_FS_GINTSTS 的 SRQINT 位)。
6. 应用程序需要响应会话请求检测中断，并通过写主机端口控制和状态寄存器的端口供电位来打开对端口的供电。PHY 会拉高 VBUS_VALID 信号来指示端口供电已被打开。
7. 当 USB 总线恢复供电，设备连接，SRP 过程结束。

B 类设备的会话请求协议

应用程序必需设置控制器 USB 配置寄存器的 SRP 使能位。此位将使能 OTG_FS 控制器作为 B 类设备时的 SRP 功能。SRP 功能使 OTG_FS 控制器可以向主机发起一个会话请求。

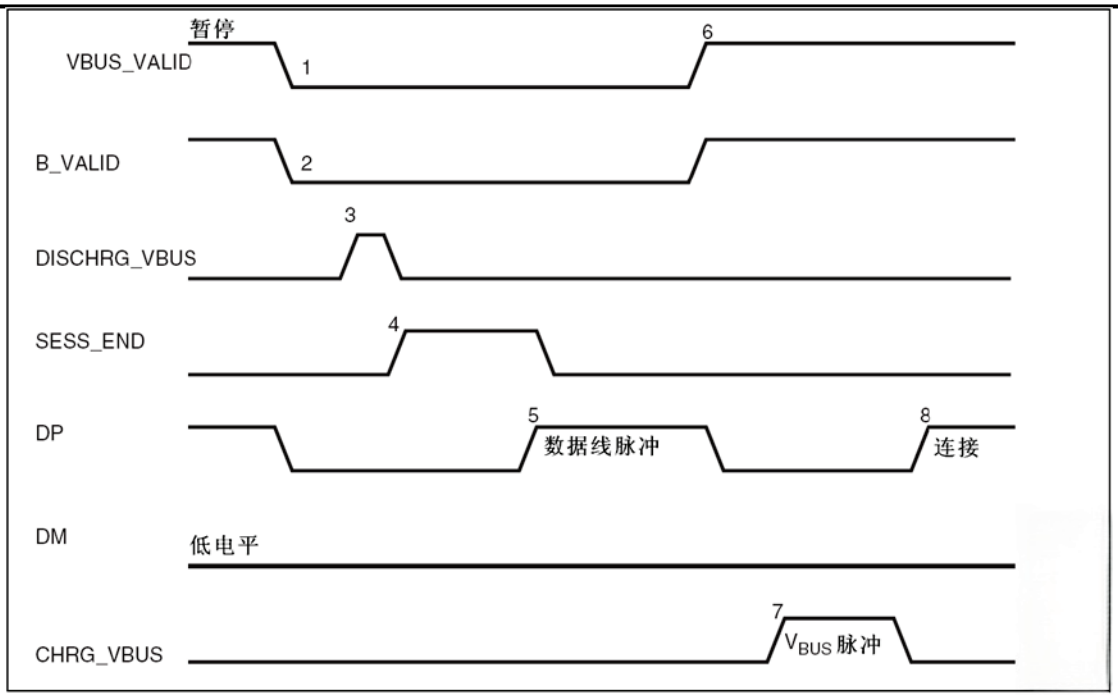


图 295 B 类设备 SRP

VBUS_VALID：来自 PHY 的 VBUS 有效信号

B_VALID：通知 PHY 的 B 类设备有效信号

DISCHRG_VBUS：通知 PHY 的放电信号

SESS_END：通知 PHY 的会话结束信号

CHRG_VBUS：通知 PHY 的驱动 VBUS 的信号

DP：正向的数据线

DM：反向的数据线

1. 为了节省耗电，主机可以在总线空闲的时候挂起和关闭端口供电。
   OTG_FS 控制器会在总线空闲 3ms 后，设置控制器中断寄存器的早期挂起位。紧跟着，OTG_FS 控制器会设置控制器中断寄存器的 USB 挂起位。
   OTG_FS 控制器会通知 PHY 停止 VBUS 的供电。
2. PHY 会指示发向设备的会话停止信号。这是发起 SRP 请求的先决条件。OTG_FS 控制器在发起 SRP 请求之前，需要保持至少 2ms 的 SE0 状态。
   对于 USB1.1 全速收发器，应用程序需要在 BSVLD 位(OTG_FS_GOTGCTL)复位后等待 VBUS 降到 0.2V。这段等待时间由收发器的供应商决定，并且不同的收发器也各不相同。
3. 应用程序通过写 OTG 控制和状态寄存器的会话请求位来发起 SRP 请求。OTG_FS 控制器将先输出数据线的脉冲，再输出 VBUS 的脉冲。
4. 主机可以根据 VBUS 或者数据线的脉冲，识别到 SRP 请求，打开 VBUS 的供电。PHY 将指示 VBUS 已重新供电。
5. OTG_FS 控制器产生 VBUS 脉冲。
   主机将打开 VBUS 的供电，开始一个新的会话，这表示 SRP 请求成功。OTG_FS 控制器会设置 OTG 中断状态寄存器的”会话请求成功状态改变位”来通知应用程序。应用程序可以通过读 OTG 控制和状态寄存器的会话请求成功位来获得这一信息。
6. 当 USB 重新供电，OTG_FS 控制器连接，SRP 请求成功完成。

A 类设备的主机协商协议

HNP 协议用于在 A 类设备和 B 类设备间切换主机角色。应用程序需要设置控制器 USB 配置寄存器的 HNP 使能位，来使能 OTG_FS 控制器作为 A 类设备时的 HNP 功能。

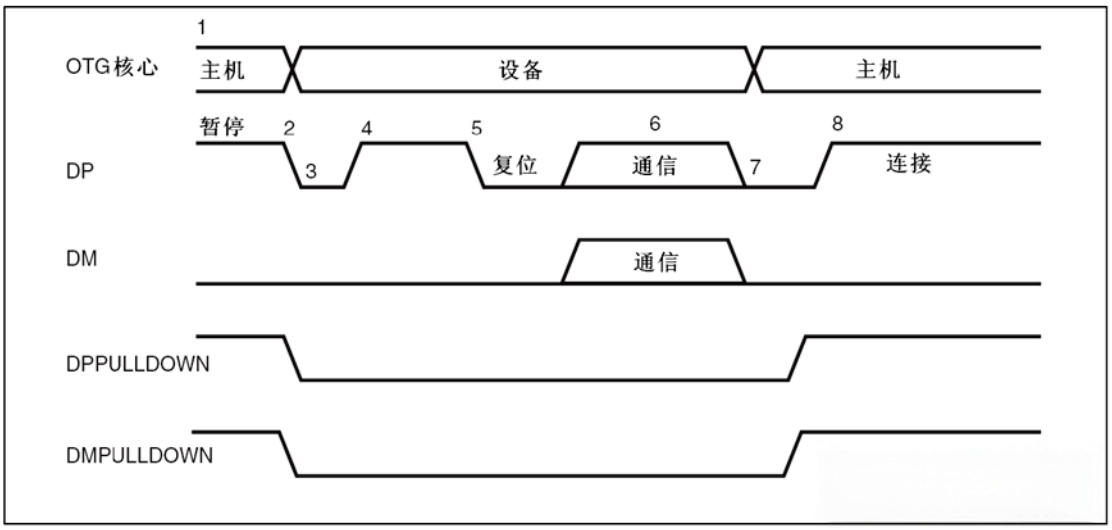


图 296 A 类设备 HNP

DPPULLDOWN：从控制器发向 PHY 的使能/取消使能 PHY 内部 DP 线的下拉的信号 DMPULLDOWN：从控制器发向 PHY 的使能/取消使能 PHY 内部 DM 线的下拉的信号

1. OTG_FS 控制器会发送 SetFeatureb_hnp_enable 命令到 B 类设备，来使能 HNP 功能。B 类设备以 ACK 来响应命令，表示 B 类设备支持 HNP 协议。应用程序需要设置 OTG 控制和状态寄存器的 HNP 使能位，来告知 OTG_FS 控制器，连接上的 B 类设备支持 HNP 协议。
2. 当应用程序不需要再使用到总线，需要设置主机端口控制和状态寄存器的端口挂起位来挂起总线。
3. 当 B 类设备检测到 USB 挂起信号，可以断开连接，指示开始发起 HNP 请求。B 类设备只有在需要执行主机角色的时候才需要发起 HNP 请求，否则就保持总线的挂起状态。
   OTG_FS 控制器会设置 OTG 中断状态寄存器的主机协商已检测到中断位，来通知应用程序检测到 HNP 请求。
   OTG_FS 控制器会解除 DM 和 DP 线的下拉，来执行设备角色。PHY 会使能 DP 线的上拉来指示 B 类设备的接入。
   应用程序必需读 OTG 控制和状态寄存器的当前模式位，来获知当前的工作模式。
4. B 类设备检测到设备的接入，发起 USB 复位，并枚举 OTG_FS 控制。
5. B 类设备一直执行主机角色，发起通信，并在完成操作后，挂起总线。
   OTG_FS 控制器在检测到总线超过 3ms 的空闲后，会设置控制器中断寄存器的早期挂起位，紧跟着会设置控制器中断寄存器的 USB 挂起位。
6. 在协商模式下，OTG_FS 控制器会检测总线的挂起，断开设备，并切换回主机角色。OTG_FS 控制器会使能 PHY 内部 DM 和 DP 线的下拉，指示重新开始执行主机角色。
7. OTG_FS 控制器会设置 OTG 中断状态寄存器的接入 ID 线状态改变中断。应用程序必需读取 OTG 控制和状态寄存器的接入 ID 线状态位来判断 OTG_FS 控制器是否工作在 A 类设备模式下。这标志 HNP 协议的完成。应用程序必需读取 OTG 看哦你看告知和状态寄存器的当前模式位来判断是否工作在主机模式下。
8. B 类设备重新接入，完成 HNP 协议。

B 类设备主机协商协议

HNP 协议用于在 A 类设备和 B 类设备间切换主机角色。应用程序必需设置控制器 USB 配置寄存器的 HNP 使能位，来通知 OTG_FS 控制器在作为 B 类设备时，使能 HNP 功能。

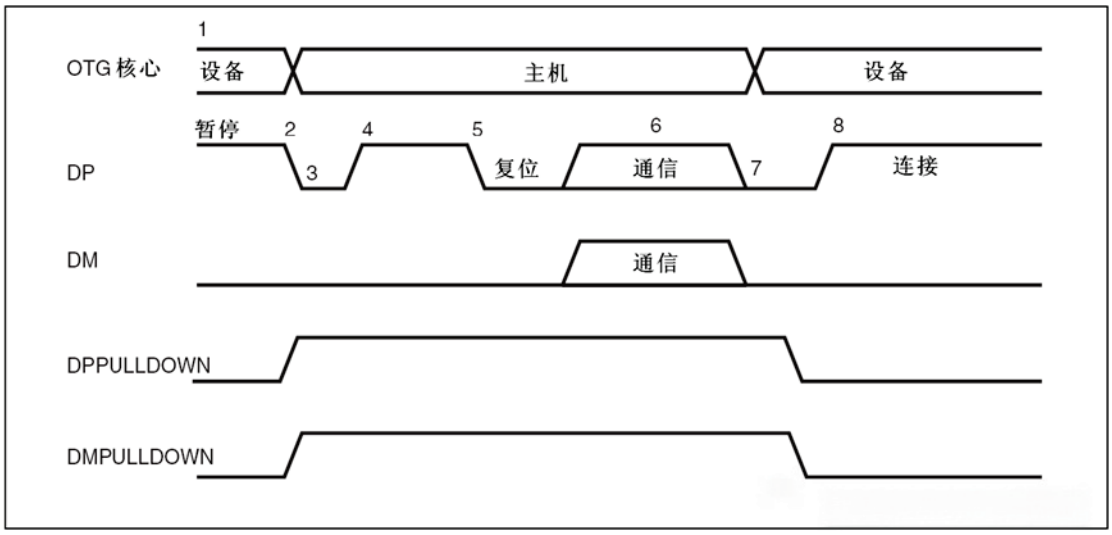


图 297 B 类设备 HNP

DPPULLDOWN：从控制器发向 PHY 的对 PHY 内部的 DP 线的下拉使能/取消使能的信号。

DMPULLDOWN：从控制器发向 PHY 的对 PHY 内部的 DM 线的下拉使能/取消使能的信号。

1. A 类设备会发送 SetFeatureb_hnp_enable 命令来使能 HNP 协议。OTG_FS 控制器需要回复 ACK 响应，以告知支持 HNP 协议。应用程序必需设置 OTG 控制和状态寄存器的设备 HNP 使能位来表示支持 HNP 协议。

   应用程序需要设置 OTG 控制和状态寄存器的 HNP 请求位来通知 OTG_FS 控制器，发起 HNP 请求。
2. 当停止使用总线时，A 类设备会挂起总线。A 类设备设置主机端口控制和状态寄存器的端口挂起位来挂起总线。

   OTG_FS 控制器在查到超过 3ms 的总线空闲时，会设置控制器中断寄存器的早期挂起位，紧跟着会设置控制器中断寄存器的 USB 挂起位。

   OTG_FS 控制器会断开连接，A 类设备将检测到总线上的 SE0 状态，表示开始了 HNP 协议。

   OTG_FS 控制器会使能 DP 和 DM 的下拉，指示开始执行主机角色。

   A 类设备在检测到 SE0 状态超过 3ms 后，会使能 DP 线的上拉电阻。OTG_FS 控制器将认为有设备插入。

   OTG_FS 控制器会设置 OTG 中断状态寄存器的”主机协商成功状态改变”中断，来指示开始 HNP 协议。应用程序必需读 OTG 控制和状态寄存器的主机协商成功位，来获得主机协商是否成功的信息。应用程序必需读控制器中断寄存器的当前模式位，来判断控制器是否工作在主机模式下。
3. 应用程序设置复位位(OTG_FS_HPRT 的 PRST 位)，OTG_FS 控制器执行 USB 复位操作，并枚举 A 类设备。
4. OTG_FS 控制器一直执行主机角色，发起通信，并在需求结束时，通过写主机端口控制和状态寄存器的端口挂起位，来挂起总线。
5. 在协商模式下，当 A 类设备检测到总线挂起，会断开连接，并切换回主机角色。OTG_FS 控制器将取消 DP 和 DM 的下拉指示重新执行设备角色。
6. 应用程序必需读取控制器中断寄存器的当前模式位，以确定是否工作在主机模式下。
7. OTG_FS 控制器恢复连接，结束 HNP 协议。

27 器件电子签名

27.1 存储器容量寄存器

27.1.1 闪存容量寄存器

基地址：0x1FFF F7E0

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
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位	符号	说明
15:0	F_SIZE	**F_SIZE**：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：0x0080=128K 字节

27.2 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 W55MH32 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 F7E8

地址偏移：0x00

只读，其值在出厂时编写

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

位	符号	说明
31:16	U_ID[15:0]	**U_ID[15:0]**：唯一身份标志 15:0 位

地址偏移：0x02

只读，其值在出厂时编写

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位	符号	说明
31:16	U_ID[31:16]	**U_ID[31:16]**：唯一身份标志 31:16 位 这个域的数值也预留作为未来的其它功能。

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
U_ID[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
U_ID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位	符号	说明
31:16	**U_ID[63:32]**	**U_ID[63:32]**：唯一身份标志 31:16 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
U_ID[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
U_ID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位	符号	说明
31:16	**U_ID[95:64]**	**U_ID[95:64]**：唯一身份标志 95:64 位

28 调试支持(DBG)

28.1 概况

W55MH32 使用 Cortex™-M3 内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指(指令断点)或访问数据(数据断点)时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当 W55MH32 微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

支持两种调试接口：

- 串行接口
- JTAG 调试接口

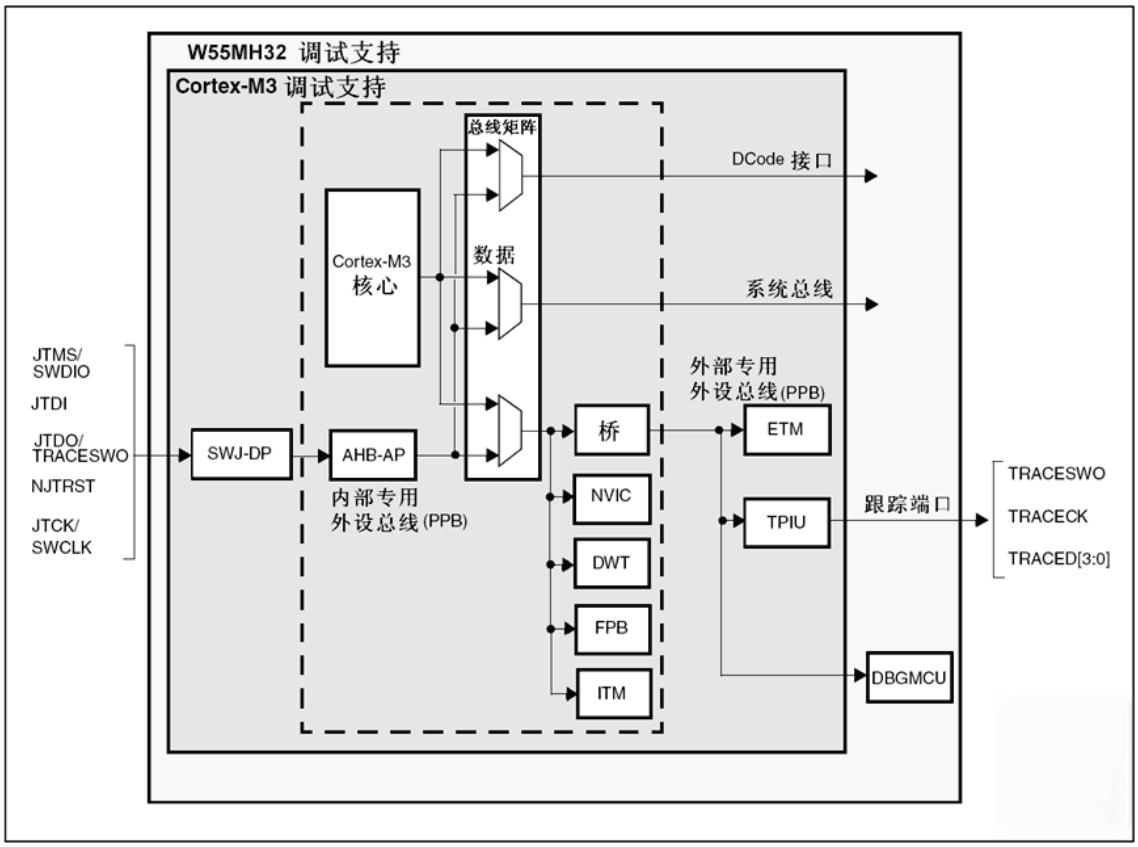


图 298 W55MH32 级别和 Cortex™-M3 级别的调试框图

注意： *Cortex™-M3 内核内含的硬件调试模块是 ARM CoreSight 开发工具集的子集。*

ARM Cortex™-M3 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口
- AHP-AP：AHB 访问端口
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPUI：跟踪单元接口(仅较大封装的芯片支持)

- ETM：嵌入式跟踪微单元(在较大的封装上才有支持此功能的引脚)专用于 W55MH32 的调试特性
- 灵活的调试引脚分配
- MCU 调试盒(支持低电源模式，控制外设时钟等)

注意：更多 ARM Cortex™-M3 内核的调试功能信息，请参考 Cortex™-M3(r1p1 版)技术参考手册(TRM)和 CoreSight 开发工具集(r1p0 版)TRM。

28.2 ARM 参考文献

- Cortex™-M3(r1p1 版)技术参考手册(TRM)
- ARM 调试接口 V5
- ARM CoreSight 开发工具集(r1p0 版)技术参考手册

28.3 SWJ 调试端口(serial wire and JTAG)

W55MH32 内核集成了串行/JTAG 调试接口(SWJ-DP)。这是标准的 ARM CoreSight 调试接口，包括 JTAG-DP 接口(5 个引脚)和 SW-DP 接口(2 个引脚)。

- JTAG 调试接口(JTAG-DP)为 AHP-AP 模块提供 5 针标准 JTAG 接口。
- 串行调试接口(SW-DP)为 AHP-AP 模块提供 2 针(时钟 + 数据)接口。

在 SWJ-DP 接口中，SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

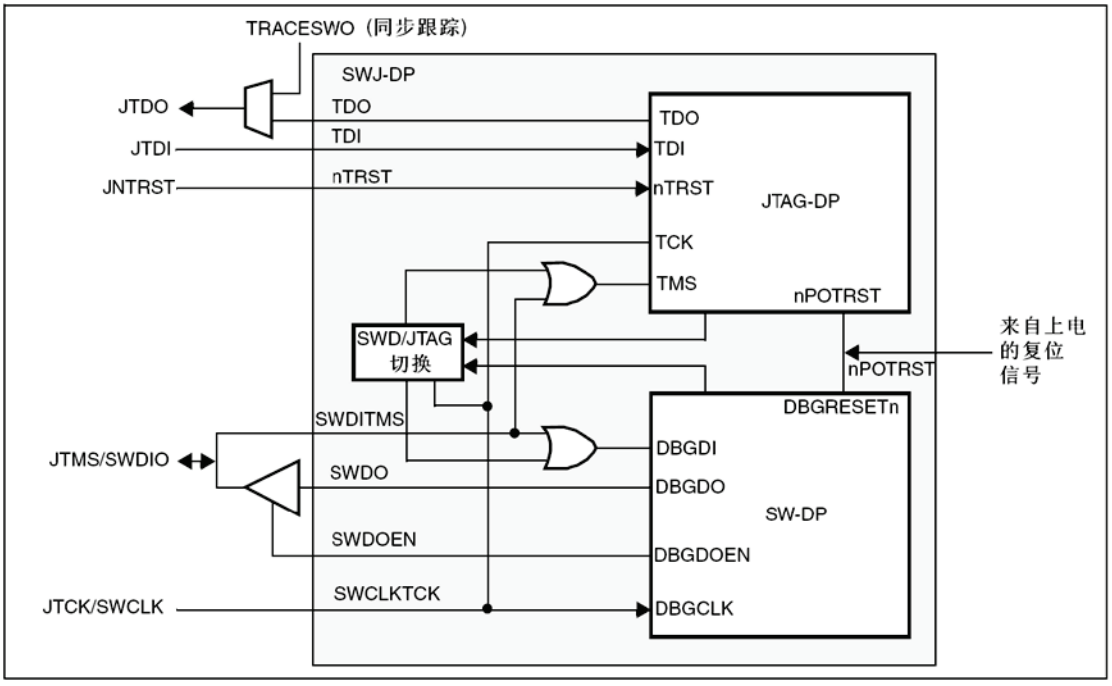


图 299 SWJ 调试端口

上面的图显示异步跟踪输出脚(TRACE SWO)和 TDO 是复用的，因此异步跟踪功能只能在 SW-DP 调试接口上实现，不能在 JTAG-DP 调试接口上实现。

28.3.1 JTAG-DP 和 SW-DP 切换的机制

JTAG 调试接口是默认的调试接口。

如果调试器想要切换到 SW-DP，必须在 TMS/TCK 上输出一指定的 JTAG 序列(分别映射到 SWDIO 和 SWCLK)，该序列禁止 JTAG-DP，并激活 SW-DP。该方法可以只通过 SWCLK 和

SWDIO 两个引脚来激活 SW-DP 接口。指定的序列是：

1. 输出超过 50 个 TCK 周期的 TMS(SWDIO) = 1 信号
2. 输出 16 个 TMS(SWDIO)信号 0111100111100111(MSB)
3. 输出超过 50 个 TCK 周期的 TMS(SWDIO) = 1 信号

28.4 引脚分布和调试端口脚

28.4.1 SWJ 调试端口脚

W55MH32 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表 157 SWJ 调试端口引脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTMS/SWDIO	输入	JTAG 模式选择	输入/输出	串行数据输入/输出	PA13
JTCK/SWCLK	输入	JTAG 时钟	输入	串行时钟	PA14
JTDI	输入	JTAG 数据输入	——	——	PA15
JTDO/TRACESWO	输出	JTAG 数据输出	——	跟踪时为 TRACESWO 信号	PB3
JNTRST	输入	JTAG 模块复位	——	——	PB4

28.4.2 灵活的 SWJ-DP 脚分配

复位(SYSRESETn 或 PORESETn)以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调试器使用的专用引脚(注意，并没有初始化跟踪输出脚，除非调试器对此脚进行定义)。

然而，W55MH32 微控制器可以用复用重映射和调试 I/O 配置寄存器(AFIO_MAPR)寄存器(见 7.4.2 节)来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口。此寄存器被映射到和 Cortex™-M3 系统总线相连接的 APB 桥上。对此寄存器的设置将由用户代码而不是调试器完成。

3 个控制位用来配置 SWJ-DP 接口的引脚，这 3 个位在系统复位时复位。

- AFIO_MAPR(地址是 0x4001 0004)
  - 读：APB，无等待状态
  - 写：APB，如果 AHB-APB 桥的写缓冲器满了，则一个等待状态

位 26:24=SWJ_CFG[2:0]

由软件置位和复位

这 3 位用来设置分配给 SWJ 调试接口的专用引脚数目，目的是在使用不同的调试接口时能释放尽可能多的引脚用作普通 I/O 口。

复位后的初始值是 000(所有引脚都设置为 JTAG-DP 接口专用引脚)，同时只能置位 3 个位中的一个(禁止同时设置一个以上的位)。

表 158 灵活的 SWJ_DP 引脚分配

SWJ-CFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO	PB4/ JNTRST
000	所有的 SWJ 引脚 (JTAG-DP+SW-DP) 复位状态	专用	专用	专用	专用	专用
001	所有的 SWJ 引脚 (JTAG-DP+SW-DP) 除了 JNTRST 引脚	专用	专用	专用	专用	释放
010	JTAG-DP 接口禁止，SW-DP 接口允许	专用	专用	释放		
100	JTAG-DP 接口和 SW-DP 接口都禁止	释放				
其他	禁止					

注意：当 APB 桥的写缓冲区满了的时候，在写 AFIO_MAPR 寄存器时需要多用一个 APB 周期。这是因为 JTAGSW 脚的释放需要 2 个 APB 周期，以保证输入内核的 nTRST 和 TCK 信号的平稳。

- 周期 1：输入 1/0 的 JTAGSW 信号到内核(nTRST,TDI 和 TMS 为 1，TCK 为 0)。
- 周期 2：GPI/O 控制器获得 SWJTAGI/O 引脚的控制信号(如对方向，上拉/下拉，施密特触发等的控制)。

28.4.3 JTAG 脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为他们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/TCK 引脚，因为他们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平，W55MH32 在 JTAG 输入脚上嵌入了内部上拉和下拉。

- JINTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO:内部上拉
- TCK/SWCLK:内部下拉

一旦 JTAGI/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态。

- JNTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入
- JICK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

注意：JTAG IEEE 标准建议对 TDI，TMS 和 nTRST 上拉，而对 TCK 没有特别的建议。但在 W55MH32 中，JTCK 引脚带有下拉。

内嵌的上拉和下拉使芯片不再需要外加外部电阻。

28.4.4 利用串行接口并释放不用的调试脚作为普通 I/O 口

为了利用串行调试接口来释放一些普通 I/O 口，用户软件必须在复位后设置 SWJ_CFG=010，从而释放 PA15，PB3 和 PB4 用做普通 I/O 口。

在调试时，调试器进行以下操作：

- 在系统复位时，所有 SWJ 引脚被分配为专用引脚(JTAG-DP+SW-DP)。
- 在系统复位状态下，调试器发送指定 JTAG 序列，从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试器在复位地址处设置断点
- 释放复位信号，内核停止在复位地址处。
- 从这里开始，所有的调试通信将使用 SW-DP 接口，其他 JTAG 引脚可以由用户代码改配为普通 I/O 口。

注意： *对于用户软件设计，应注意：*

在复位后，这些专用引脚仍然处于带上拉的输入(nTRST,TMS,TDI)，带下拉的输入(TCK)，和输出(TDO)状态，并持续一段时间，直到用户代码释放这些引脚。

当这些引脚被配置成专用引脚时(JTAG 或者 SW 或者 TRACE)，修改相应的普通 I/O 口配置寄存器是无效的。

28.5 W55MH32JTAG TAP 连接

W55MH32 微控制器内部串联了两个 JTAG TAP。边界扫描 TAP 专门用来进行测试(IR 寄存器为 5 比特位宽)和 Cortex™-M3 TAP(IR 寄存器为有 4 比特位宽)。

为了访问 Cortex™-M3 TAP 对芯片进行调试，必须：

1. 首先，必须将 BYPASS 指令移位输入 TMCTAP。
2. 其次，在移位输入 IR 时，每个扫描链包含 9 个比特位(=5+4)，对于不用的 TAP，必须输入 BYPASS 指令
3. 移位输入数据时，不用的 TAP 处于 BYPASS 模式下，因此数据扫描链需要额外添加一位比特位。

注意： *重要：一旦使用了指定的 JTAG 序列选择了串行调试接口，TMC TAP 自动被禁止(JTMS 被强制为高)。*

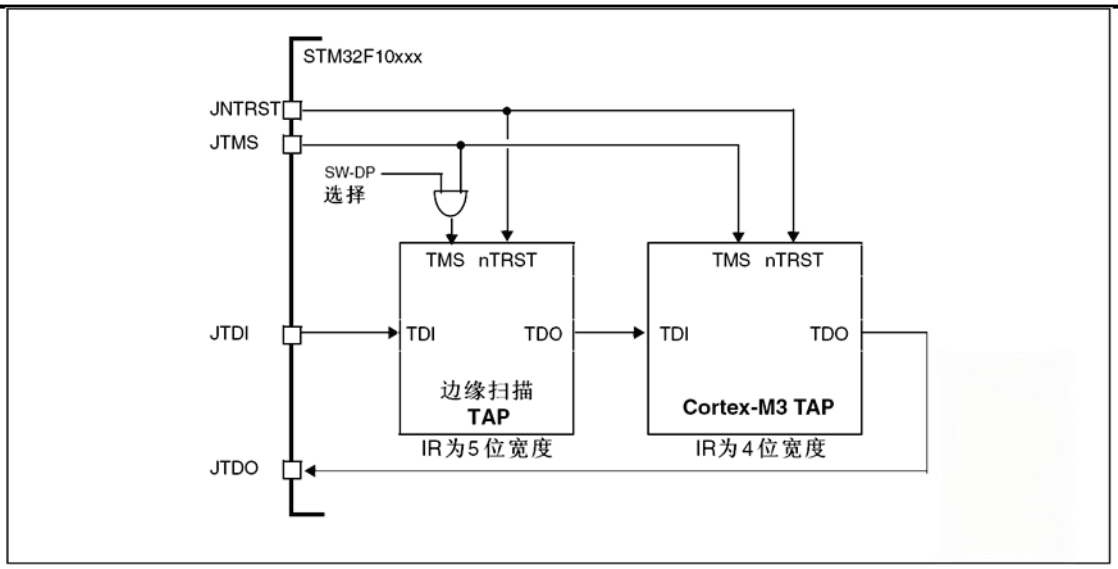


图 300 JTAG TAP 连接

28.6 ID 代码和锁定机制

在 W55MH32 微控制器内部有多个 ID 编码。强烈建议工具设计者使用映射在外部 PPB 存储器上地址为 0xE004 2000 的 MCU DEVICE ID 来锁定调试器。

28.6.1 微控制器设备 ID 编码

微控制器 W55MH32 内含一个 MCUID 编码。这个 ID 定义了 MCU 的部件号和硅片版本。它是 DBG_MCU 的一个组成部分，并且映射到外部 PPB 总线上(见 28.16 节)。使用 JTAG 调试口(4~5 个引脚)或 SW 调试口(2 个引脚)或通过用户代码都可以访问此编码。即使当 MCU 处于系统复位状态下这个编码也可以被访问。

DBGMCU_IDCODE

地址：0xE004 2000 只支持 32 位访问

只读=0xXXXX X410，其中 X 为内容不确定的位

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

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

位	符号	说明
31:16	REV_ID[15:0]	**REV_ID[15:0]**：版本识别该域标识产品的版本 0x1000=版本 A 0x1001=版本 Z
15:12	Reserved	保留
11:0	DEV_ID[11:0]	**DEV_ID[11:0]**：设备识别 这个部分指示了设备编码。 设备编码为 0x414;

28.6.2 边界扫描 TAP

JTAGID 编码

W55MH32 的边界扫描 TAP 集成了 JTAGID 编码：0x0641 4041=版本 A

28.6.3 Cortex-M3 TAP

ARM Cortex-M3 的 TAP 有一个 JTAG ID 编码。这个 ID 编码是 ARM 默认的，且没有被修改过，只能通过 JTAG 调试口访问。此编码是 0x3BA0 0477(对应 Cortex-M3 r1p1)。
调试器/编程工具应该只通过 DEV_ID(11:0)来识别芯片。

28.6.4 Cortex-M3 JEDEC-106 ID 代码

ARM 的 Cortex-M3 有一个 JEDEC-106ID 编码。它位于映射到内部 PPB 总线地址为 0xE00FF000_0xE00F FFFF 的 4KB ROM 表中。

28.7 JTAG 调试端口

标准的 JTAG 状态机是通过一个 4 比特位的指令寄存器(IR)和 5 个数据寄存器(详见 Cortex-M3 r1p1 Technical Reference Manual)实现的。

表 159 JTAG 调试端口数据寄存器

IR(3:0)	数据寄存器	描述
1111	BYPASS[1 比特位]	
1110	IDCODE[32 比特位]	**ID 编码寄存器** 0x3BA0 0477(ARM Cortex-M3 r1p1-01rel0 ID 编码)
1010	DPACC[32 比特位]	**调试接口寄存器** 初始化调试端口，并允许访问调试接口寄存器 -输入数据时： Bits34:3=DATA[31:0]：对应写操作的 32 位数据位 Bits2:1=A[3:2]：调试接口寄存器的 2 位地址值 Bit0=RnW：读操作(1)或写操作(0) -输出数据时： Bits34:3=DATA[31:0]：前一次读操作的 32 位数据结果 Bits2:0=ACK[2:0]：3 比特位的应答 010 = 成功/失败 001 = 等待其他 = 未定义 A(3:2)的定义请参考 0
1011	APACC[35 比特位]	**存取接口寄存器** 初始化存取接口并允许访问存取接口寄存器 -输入数据时： Bits34:3=DATA[31:0]：对应写操作的 32 位数据位 Bits2:1=A[3:2]：2 比特位地址(AP 寄存器的部分地址)Bit0=RnW：读操作(1)或写操作(0) -输出数据时： Bits34:3=DATA[31:0]：前一次读操作的 32 位数据结果 Bits2:0=ACK[2:0]：3 比特位的应答 010 = 成功/失败 001 = 等待其他 = 未定义 关于 AP 寄存器请参考 AHB-AP 章节，这些寄存器的地址由以下部分组成：A[3:2] -移位值 A[3:2] -DPSELECT 寄存器的当前值

1000	ABORT[35 比特位]	**中止寄存器** -Bits31:1 未定义 -Bit0 = DAPABORT：写 1 产生一个 DAP 中止

表 160 由 A[3:2]定义的 32 位调试接口寄存器地址

地址	A(3:2)值	描述
0x0	00	未定义
0x4	01	DPCTRL/STAT 寄存器 -请求一个系统或调试的上电操作 -配置 AP 访问的操作模式 -控制比较，校验操作 -读取一些状态位(溢出，上电响应)
0x8	10	DPSELECT 寄存器 用来选择当前的访问端口和有效的 4 字长寄存器窗口 -Bits31:24：APSEL 选择当前 AP -Bits23:8：未定义 -Bits7:4：APBANKSEL：在当前 AP 上选择 4 字长寄存器窗口 -Bits3:0：未定义
0xC	11	DPRDBUFF 寄存器：用来使调试器获得前一次操作的最终结果(不用再请求一个新的 JTAG-DP 操作)

28.8 SW 调试端口

28.8.1 SW 协议介绍

此同步串行协议使用 2 个引脚：

- SWCLK：从主机到目标的时钟信号
- SWDIO：双向数据信号

协议允许读写 2 个寄存器组(DPACC 和 APACC 寄存器组)。数据位按 LSB 传输。

由于 SWDIO 为双向口，该引脚需有上拉(ARM 建议使用 100KΩ 电阻)。

按协议每次 SWDIO 方向改变时，需插入一个转换时间。在该期间内主机和目标都不驱动此信号线。

转换时间的默认值是 1 个比特，但可以通过配置 SWCLK 频率来调节。

28.8.2 SW 协议序列

每个序列由 3 个阶段组成：

1. 主机发送包请求(8 位)
2. 目标发送确认响应(3 位)
3. 主机或目标发送数据(33 位)

表 161 请求包(8 比特位)

比特位	名称	描述
0	起始	必须为 1
1	APnDP	0：访问 DP1：访问 AP
2	RnW	0：写请求 1：读请求
4:3	A(3:2)	DP 或 AP 寄存器的地址(请参考 0)
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为 1

有关 DPACC 和 APACC 寄存器描述的详细资料，请参考 Cortex-M3 r1p1 技术参考手册。包请求后总是跟一个(默认为 1 位)转换时间，此时主机和目标都不驱动线路。

表 162 ACK 定义(3 比特位)

比特位	名称	描述
0..2	ACK	001：失败 010：等待 100：成功

当 ACK 为失败或等待，或者是一个回复读操作的 ACK，此 ACK 后有一个转换时间。

表 163 传输数据(33 比特位)

比特位	名称	描述
0..31	WDATA/RDATA	写或读的数据
32	Parity	32 位数据的奇偶校验位

读操作的数据传输操作后有一个转换时间。

28.8.3 SW-DP 状态机(Reset,idle states,IDcode)

SW-DP 状态机有一个内部 ID 编码用来识别 SW-DP，它遵守 JEP-106 标准。此 ID 编码是 ARM 默认的编码，值为 0x1BA0 1477(对应于 Cortex-M3 r1p1)。

注意：在调试器读这个 ID 编码之前，SW-DP 的状态机是不工作的。

- SW-DP 状态机将处于 RESET 状态，在上电复位后，或 DP 从 JTAG 切换到 SWD 后，或有超过 50 个周期的高电平。
- 当状态机处于 RESET 状态时，如果有至少 2 个周期的低电平，状态机将切换到 IDLE 状态。
- 当状态机处于 RESET 状态后，必需首先进入 IDLE 状态，并执行一个读 DP-SWID 寄存器的操作。否则，调试器在执行其他传输时，只能获得一个失败的 ACK 响应。

更详细的 SW-DP 状态机资料请参考 Cortex-M3 r1p1 技术参考手册和 Core Sight Design Kit r1p0 技术参考手册。

28.8.4 DP 和 AP 读/写访问

- 对 DP 的读操作没有延迟：调试器将直接获得数据(如果 ACK = 成功)，或者等待(如果 ACK=等待)。

- 对 AP 的读操作具有延迟。即前一次读操作的结果只能在下一次操作时获得。如果下一次的操作不是对 AP 的访问，则必需读 DP-RDBUFF 寄存器来获得上一次读操作的结果。
- DP-CTRL/STAT 寄存器的 READOK 标志位会在每次 AP 读操作和 RDBUFF 读操作后更新，以通知调试器 AP 的读操作是否成功。
- SW-DP 具有写缓冲区(DP 和 AP 都有写缓冲)，这使得其他传输在进行时，仍然可以接受写操作。如果写缓冲区满，调试器将获得一个等待的 ACK 响应。读 IDCODE 寄存器，读 CTRL/STAT 寄存器和写 ABORT 寄存器操作在写缓冲区满时仍被接受。
- 由于 SWCLK 和 HCLK 的异步性，需要在写操作后(在奇偶校验位后)插入 2 个额外的 SWCLK 周期，以确保内部写操作正确完成。这两个额外的时钟周期需要在线路为低时插入(IDLE 状态下)。这个操作步骤在写 CTRL/STAT 寄存器以提出一个上电请求时尤其重要，否则下一个操作(在内核上电后才有效的操作)会立即执行，这将导致失败。

28.8.5 SW-DP 寄存器

当 APnDP=0 时，可以访问以下这些寄存器。

表 164 SW-DP 寄存器

A(3:2)	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	读		IDCODE	固定为 0x1BA0 1477(用于识别 SW-DP)。
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	-请求一个系统或调试的上电操作; -配置 AP 访问的操作模式; -控制比较，校验操作; -读取一些状态位(溢出，上电响应)。
01	读/写	1	WIRE CONTROL	配置串行通信物理层协议(如转换时间长度等)。
10	读		READ RESEND	允许从一个错误的调试传输中恢复数据而不用重复最初的 AP 传输。
10	写		SELECT	选择当前的访问端口和有效的 4 字长寄存器窗口。
11	读/写		READ BUFFER	由于 AP 的访问具有传递性(当前 AP 读操作的结果会在下次 AP 传输时传出),因此这个寄存器非常必要。这个寄存器会从 AP 捕获上一次读操作的数据结果，因此可以获得数据而不必再启动一个新的 AP 传输。

28.8.6 SW-AP 寄存器

当 APnDP=1 时，可以访问以下这些寄存器。AP 寄存器的访问地址由以下两部分组成：

- A[3:2]的值
- DPSELECT 寄存器的当前值

28.9 对于 JTAG-DP 或 SWDP 都有效的 AHB-AP(AHB 访问端口)

功能:

- 系统访问是独立于处理器状态的。
- JTAG-DP 和 SW-DP 都可以访问 AHB-AP
- AHB-AP 是总线矩阵的 AHB 主设备。因此，它可以访问所有的数据总线(Dcode 总线，System 总线，内部和外部 PPB 总线)，只有 ICode 总线除外。
- 支持位寻址的传输
- 旁路 FPB 的 AHB-AP 传输

32 位 AHP-AP 寄存器的地址是 6-位宽(最多 64 个字或 256 个字节)，由以下部分组成:

a)比特位[8:4]=DPSELECT 寄存器的位[7:4]APBANKSEL

b)比特位[3:2]=35 位 SW-DP 包请求中的 A(3:2)。Cortex-M3 的 AHB-AP 有 9 个 32 位的寄存器

表 165 Cortex-M3AHB-AP 寄存器

地址偏移	寄存器名	描述
0x00	AHB-AP Control and Status Word	配置 AHB 接口的传输特性(长度，地址自加模式，当前传输状态，特权模式等)。
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data0	直接访问 4 个相连的字而不用重写访问地址。
0x14	AHB-AP Banked Data1	
0x18	AHB-AP Banked Data2	
0x1C	AHB-AP Banked Data3	
0xF8	AHB-AP Debug ROM Address	调试接口的基地址。
0xFC	AHB-AP ID Register	

更多信息请参考 Cortex-M3r1p1 技术参考手册

28.10 内核调试

通过操作内核调试寄存器可以实行对内核的调试。对这些寄存器的访问通过先进高性能总线(AHB-AP)进行。处理器可以通过内部私有外设总线(PPB)直接访问这些寄存器。

它包括 4 个寄存器。

表 166 内核调试寄存器

寄存器	描述
DHCSR	32 位的调试控制和状态寄存器 此寄存器提供内核状态信息，允许内核进入调试模式，和提供单步功能。
DCRSR	17 位的内核寄存器调试选择寄存器 此寄存器选择需要进行读写操作的内核寄存器。
DCRDR	32 位的内核寄存器调试数据寄存器 此寄存器存放由 DCRSR 选择的内核寄存器读出的或需要写入的数据。

DEMCR	32 位异常调试和监视控制寄存器 此寄存器提供向量传输和监视调试控制功能。TRCENA 位启动 TRACE 功能。

注意：　重要：这些寄存器在系统复位时不复位，仅在上电复位时复位。

更多详细资料请参考 Cortex-M3r1p1 技术参考手册。为了在复位后立即使内核进入调试状态，需要：

- 使能调试和异常监视控制寄存器(Debug and Exception Monitor Control Register)的位 0(VC_CORRESET)。
- 使能调试控制和状态寄存器(Debug Halting Control and Status Register)的位 0(C_DEBUGEN)。

28.11 调试器主机在系统复位下的连接能力

W55MH32 微控制器的复位系统由下列复位源组成：

- POR(上电复位)，在每次上电时发起一次复位
- 内部看门狗复位
- 软件复位
- 外部复位

Cortex-M3 将调试部分的复位(通常是 PORRESETn)和其他复位(SYSRESETn)区分开。因此，当内核处于系统复位状态时，调试器可以连接到内核，配置内核调试寄存器，使能调试允许位，这样操作使内核在系统复位被释放时立即进入调试状态而不执行任何指令。同样的，可以在内核处于复位状态下时配置调试特性。

注意：　强烈建议调试器在系统复位时连接内核(在复位向量处设置断点)。

28.12 FPB(Flash patch breakpoint)

FPB 单元：

- 实现硬件断点
- 用系统区域的代码和数据取代代码区域的代码和数据。此特性可以用来纠正代码区域内的软件错误。

软件补丁功能和硬件断点功能不能同时使用。FPB 由以下部分组成：

- 2 个内容比较器，用来比较代码区域取得的内容并重映射到系统区域的相关地址。
- 6 个指令比较器，用来比较代码区域的指令。这些比较器可用来实现软件补丁或者硬件断点功能。

28.13 DWT(数据观察点触发 data watch point trigger)

DWT 模块由四个比较器组成，它们分别是：

- 一个硬件数据比较器
- 一个 ETM 触发器
- 一个 PC 值取样器
- 一个数据地址取样器

DWT 还可用来获取某些侧面的信息。通过一些计数器可以获得以下数据：

- 时钟周期
- 分支指令
- 存取单元操作

- 睡眠周期
- CPI(每条指令的执行时间)
- 中断开销

28.14 ITM(指令跟踪微单元 instrumentation trace macrocell)

28.14.1 概述

ITM 是一应用驱动的跟踪源，它支持 printf 类的调试手段来跟踪操作系统(OS)和应用事件，并发布判定的系统信息。ITM 以包的形式发布跟踪信息，它由以下部分组成：

- 软件跟踪：软件可以通过直接写 ITM 激发寄存器来发布包信息。
- 硬件跟踪：ITM 会发布由 DWT 产生的信息包。
- 时间戳：时间戳被发布到相应的包上。ITM 包含一个 21 位的计数器以产生时间戳。Cortex-M3 的时钟或串行线观测器(Serial Wire Viewer)的位时钟率给计数器提供时钟。

由 ITM 发送的信息包输出到 TPIU(Trace Port Interface Unit)，TPIU 再添加一些额外的包(参考 TPIU)，然后输出完整的包序列给调试器。

用户在设置或使用 ITM 之前，必需先使能异常调试和监视控制寄存器(Debug Exception and Monitor Control Register)的 TRCEN 位。

28.14.2 时间戳包，同步和溢出包

时间戳包包含了时间戳信息，普通的控制和同步信息。它使用一个 21 位的时间戳计数器(及可能的预分频器)，此计数器在每个时间戳包发放时复位。计数器的时钟可以是 CPU 时钟也可以是 SWV 时钟。

同步包为 0x80_00_00_00_00_00，按 00 00 00 00 00 80 发送给 TPIU(LSB 在前)。同步包是时间戳包的控制信号。

它也在每个 DWT 触发时发送，因此 DWT 必须配置为触发 ITM：必须设置 DWT 控制寄存器(DWT Control Register)的位 0(CYCCNTENA)。此外，也必须设置 ITM 跟踪控制寄存器(Trace Control Register)的位 2(SYNCENA)。

注意： *如果 SYNENA 位没有被置起，DWT 产生给 TPIU 的同步触发，将只发送 TPIU 同步包而不发送 ITM 同步包。*

溢出包是一个特殊的时间戳包，该包指示数据已经被写但是 FIFO 已满。

表 167 主要的 ITM 寄存器

地址	寄存器	描述
@E0000FB0	ITM Lock Access	写入 0xC5AC CE55 允许写其他 ITM 寄存器
@E0000E80	ITM Trace Control	位 31-24=总是 0 位 23=忙 位 22-16=7 位的 ATBID 用以识别跟踪数据源位 15-10=总是 0 位 9:8=时间戳的预分频位 7-5=未定义 位 4=使能 SWV 功能即时间戳计数器使用 SWV 时钟位 3=使能 DWT 的激发功能 位 2=此位必需设为 1 来使能 DWT 的产生同步触发功能，以使 TPIU 能够发送同步包 位 1=时间戳使能 位 0=ITM 的全局使能位
@E0000E40	ITM Trace Privilege	位 3：置'1'使能跟踪端口 31:24

		位 2：置'1'使能跟踪端口 23:16 位 1：置'1'使能跟踪端口 15:8 位 0：置'1'使能跟踪端口 7:0
@E0000E00	ITM Trace Enable	每个比特位使能相应的触发端口产生跟踪
@E0000000-E000007C	Stimulus Port Registers 0-31	向选中的产生跟踪的触发端口(32 个)写 32 位数据

关于配置的例子：

向 TUIU 输出一个简单值：

- 配置 TPIU 并使能 I/IO_TRACEN 以使 MCU 分配 TRACE 的引脚(参见 28.17.2 节-跟踪引脚分配,28.16.3 节-调试 MCU 配置寄存器)；
- 向 ITM Lock Access 寄存器写入 0xC5AC CE55，以允许写其他 ITM 寄存器；
- 向 Trace Control 寄存器写入 0x0001 0005，使能 TPIU 的同步包并使能整个 ITM 功能，寄存器中的 ATBID 为 0x01；
- 向 ITM Trace Enable 寄存器写入 0x1，以使能触发端口 0；
- 向 ITM Trace Privilege 寄存器写入 0x1，关闭对触发端口 7:0 的屏蔽；
- 把需要输出的值写入触发端口 0 寄存器，这个步骤可以通过软件完成(使用 printf 功能)。

28.15 ETM 模块(嵌入式跟踪微单元 Embedded Trace Macrocell)

28.15.1 概述

ETM 可以重现程序的运行过程。使用数据观察点触发模块(DWT)或指令跟踪微单元(ITM)可以跟踪数据的变化，而是用嵌入式跟踪微单元(ETM)可以跟踪指令的执行。

ETM 由内置的资源触发并以包的形式传送信息，软件可以分别配置这些资源，使用触发事件寄存器(0xE004 1008)可以选择触发源。触发事件可以是一个简单事件(例如来自地址比较器的地址匹配)，触发事件也可以是 2 个事件之间的逻辑运算结果。触发源是 DWT 模块中四个比较器之一。

下列事件可以被观测到：

- 时钟周期匹配
- 数据地址匹配

更多有关触发源的信息，请参考第 28.13 节。

ETM 传送的信息包是通过 TPIU(跟踪端口接口单元)输出，TPIU 还需要增加一些额外的信息(详见第 0 节)，然后向调试主机输出完整的包序列。

28.15.2 信号协议和包类型

这部分的说明请参考 ARM IHI 0014N 文档的第 7 章”ETM v3 Singal Protocol”。

28.15.3 主要的 ETM 寄存器

有关寄存器的详细说明，请参考 ARMIHI0014N 文档的第 3 章。

表 168 主要的 ETM 寄存器

地址	寄存器	说明
0xE004 1FB0	ETMLockAccess	写入 0xC5AC CE55 解除对其它 ETM 寄存器的写保护
0xE004 1000	ETMControl	控制 ETM 的操作，例如如何使能跟踪
0xE004 1010	ETMStatus	提供跟踪和触发逻辑当前状态的信息
0xE004 1008	ETMTriggerEvent	定义将要控制触发的事件

0xE004 101C	ETMTraceEnableControl	定义选用的比较器
0xE004 1020	ETMTraceEnableEvent	定义跟踪使能事件
0xE004 1024	ETMTraceStart/Stop	定义触发源使用的跟踪事件，以设置跟踪的启动或停止

28.15.4 配置实例

从 TPIU 输出一个简单的数值：

- 配置 TPIU 并使能 I/IO_TRACEN，从而在调试配置寄存器中分配 TRACEI/O；
- 在 ETMLockAccess 寄存器中写入 0xC5AC CE55，解除对 ITM 寄存器的写保护；
- 在控制寄存器中写入 0x0000 1D1E(配置跟踪)；
- 在触发事件寄存器(Trigger Event)中写入 0x0004 06F(定义触发事件)；
- 在跟踪使能事件寄存器(Trace Enable Event)中写入 0x0000 006F(定义事件的启停)；
- 在跟踪启动/停止寄存器(Trace Start/Stop)中写入 0x0000 0001(跟踪使能)；
- 在 ETM 控制寄存器(Control)中写入 0x0000 191E(配置结束)。

28.16 MCU 调试模块(MCUDBG)

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器、看门狗、I2C 和 bxCAN 的时钟控制
- 对跟踪脚分配的控制

28.16.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。

MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。

内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK(由代码配置的系统时钟)相同的时钟。
- 在停止模式下，调试器必须先置位 DBG_STOP 位。这将激活内部 RC 振荡器，在停止模式下为 FCLK 和 HCLK 提供时钟。

28.16.2 支持定时器、看门狗、bxCAN 和 I2C 的调试

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出 PWM 控制电机时常常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

对于 bxCAN，用户可以选择在断点期间阻止接收寄存器的更新。

对于 I2C，用户可以选择在断点期间阻止 SMBUS 超时。

28.16.3 调试 MCU 配置寄存器

此寄存器允许在调试状态下下配置 MCU。包括：

- 支持低功耗模式
- 支持定时器和看门狗的计数器
- 支持 bxCAN 通信
- 分配跟踪引脚

DBGMCU_CR 寄存器被映射到外部 PPB 总线，基地址为 0xE004 2000。

寄存器由 PORESET 异步复位(不被系统复位所复位)。当内核处于复位状态下时，调试器可写该寄存器。

如果调试器不支持这些特性，用户软件仍可写这些寄存器。

DBGMCU_CR

地址：0xE004 2004 只支持 32 位访问

POR 复位：0x0000 0000(不被系统复位所复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	保留			DBG_CAN2_STOP	DBG_TIM8 STOP	DBG_TIM7 STOPT	DBG_TIM6 STOPT	DBG_TIM5 STOP	DBG_I2C2_SMBUSTIMEOUT
	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
DBG_I2C1_SMBUSTIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	res	rw	rw	rw

位	符号	说明
31	Reserved	保留，始终读为 0。
30:25	DBG_TIMx_STOP	**DBG_TIMx_STOP:**TIMx 计数器在核心停止时停止(x=9..14) 0:即使核心停止，所涉及的定时计数器的时钟也会被输入，输出行为正常。 1:当核心停止时，所涉及的计时器计数器的时钟被停止，输出被禁用(就好像有针对中断事件的紧急停止一样)。
24:22	Reserved	保留，始终读为 0。

21	DBG_CAN2_STOP	**DBG_CAN2_STOP**：当内核进入调试状态时，CAN2 停止运行。 0：CAN2 仍然正常运行； 1：CAN2 的接收寄存器不继续接收数据。
20:17	DBG_TIMx_STOP	**DBG_TIMx_STOP**：当核心停止时停止定时器计数器(x=8..5) 0:当核心停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常； 1:当核心停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出(就好象对某一暂停事件的紧急响应，停止定时器)。
16	DBG_I2C2_SMBUS_TIMEOUT	**DBG_I2C2_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0:与正常模式操作相同； 1:冻结 SMBUS 的超时控制。
15	DBG_I2C1_SMBUS_TIMEOUT	**DBG_I2C1_SMBUS_TIMEOUT**：当核心停止时停止 SMBUS 超时模式。 0:与正常模式操作相同； 1:冻结 SMBUS 的超时控制。
14	DBG_CAN1_STOP	**DBG_CAN1_STOP**：当内核进入调试状态时，CAN1 停止运行。 0：CAN1 仍然正常运行； 1：CAN1 的接收寄存器不继续接收数据。
13:10	DBG_TIMx_STOP	**DBG_TIMx_STOP**：当内核进入调试状态时计数器停止工作 x=4..1。 0：选中定时器的计数器仍然正常工作； 1：选中定时器的计数器停止工作。
9	DBG_WWDG_STOP	**DBG_WWDG_STOP**：当内核进入调试状态时调试窗口看门狗停止工作。 0：窗口看门狗计数器仍然正常工作； 1：窗口看门狗计数器停止工作。
8	DBG_IWDG_STOP	**DBG_IWDG_STOP**：当内核进入调试状态时看门狗停止工作 0：看门狗计数器仍然正常工作； 1：看门狗计数器停止工作。
7:5	TRACE_MODE[1:0]和 TRACE_IOEN	**TRACE_MODE[1:0]和 TRACE_IOEN**：跟踪引脚分配控制 -当 TRACE_IOEN = 0 时：TRACE_MODE = xx：不分配跟踪引脚(默认状态)。 -当 TRACE_IOEN=1 时： TRACE_MODE=00：跟踪引脚使用异步模式； TRACE_MODE=01：跟踪引脚使用同步模式，并且数据长度为 1； TRACE_MODE=10:跟踪引脚使用同步模式，并且数据长度为 2； TRACE_MODE=11:跟踪引脚使用同步模式，并且数据长度为 4。
4:3	Reserved	保留，必须保持为 0。
2	DBG_STANDBY	**DBG_STANDBY**：调试待机模式。 0：(FCLK 关，HCLK 关)整个数字电路部分都断电。 从软件的观点看，退出 STANDBY 模式与复位是一样的(除了一些状态位指示了微控制器刚从 STANDBY 状态退出)。 1：(FCLK 开，HCLK 开)数字电路部分不下电，FCLK 和 HCLK 时钟由内部 RL 振荡器提供时钟。另外，微控制器通过产生系统复位来退出 STANDBY 模式和复位是一样的。
1	DBG_STOP	**DBG_STOP**：调试停止模式。 0：(FCLK 关，HCLK 关)在停止模式时，时钟控制器禁止一切时钟(包括 HCLK 和 FCLK)。当从 STOP 模式退出时，时钟的配置和复位之后的配置一样(微控制器由 8MHz 的内部 RC 振荡器(HIS)提供时钟)。因此，软件必需重新配置时钟控制系统启动 PLL，晶振等。 1：(FCLK 开，HCLK 开)在停止模式时，FCLK 和 HCLK 时钟由内部 RC 振荡器提供。当退出停止模式时，软件必需重新配置时钟系统启动 PLL，晶振等(与配置此比特位为 0 时的操作一样)。
0	DBG_SLEEP	**DBG_SLEEP**：调试睡眠模式 0：(FCLK 开，HCLK 关)在睡眠模式时，FCLK 由原先已配置好的系统时钟提供，HCLK 则关

		闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 1：(FCLK 开，HCLK 开)在睡眠模式时，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

28.17 TPIU(跟踪端口接口单元 Trace Port Interface Unit)

28.17.1 导言

TPIU 在片上数据跟踪和 ITM 之间担当桥梁的作用。
输出的数据流封装成跟踪源 ID，然后被追踪端口分析器(Trace Port Analyzer)采集。
内核嵌入了一个简单的专门为低价调试所设计的 TPIU(由一个特殊版本的 CoreSight TPIU 组成)。

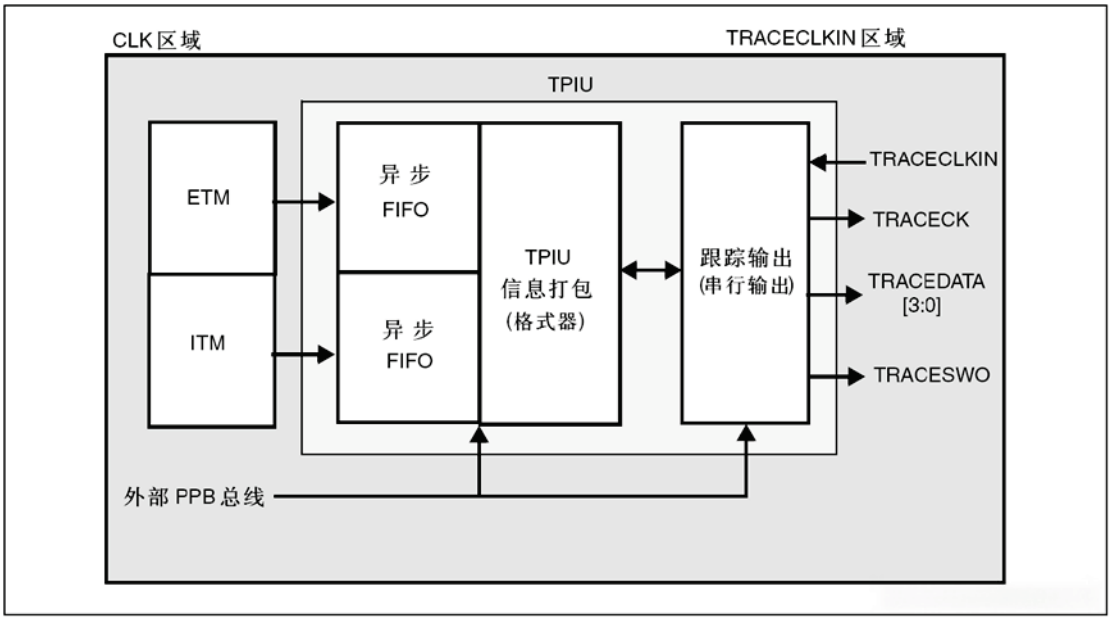


图 301 TPIU 框图

28.17.2 跟踪引脚分配

- 异步模式

异步模式需要 1 个额外的引脚并且存在于所有的封装。此模式仅在串行调试接口有效(不支持 JTAG 调试接口)。

表 169 异步跟踪引脚分配

TPUI 引脚	同步跟踪模式		W55MH32 引脚分配
	类型	描述	
TRACESWO	输出	异步跟踪数据输出	PB3

- 同步模式

同步模式根据跟踪数据长度使用 2 到 6 个额外引脚，并且只存在于大封装芯片里。此外，此模式在 JTAG 调试接口和串行调试接口下都可使用，并提供比异步跟踪更好的数据输出量。

表 170 同步跟踪引脚分配

TPUI 引脚名	同步跟踪模式		W55MH32 引脚分配
	类型	描述	
TRACECK	输出	跟踪时钟	PE2
TRACED[3:0]	输出	同步跟踪数据输出，长度可以是 1,2,或 4	PE[6:3]

TPUI 跟踪引脚分配

这些引脚在默认状态下不是专用引脚。可以通过设置 MCU Debug Component Configuration 寄存器的 TRACE_IOEN 和 TRACE_MODE 位分配这些引脚。必需由调试器完成设置。

此外，由跟踪的配置决定分配的引脚数(异步还是同步)。

- 异步模式：需要 1 个额外引脚
- 同步模式：根据跟踪端口的数据长度(1、2 或 4)决定使用 2 到 5 个额外的引脚
  - TRACECK
  - TRACED(0)如果数据长度配置为 1、2 或 4
  - TRACED(1)如果数据长度配置为 2 或 4
  - TRACED(2)如果数据长度配置为 4
  - TRACED(3)如果数据长度配置为 4

调试器需要设置 Debug MCU Configuration 寄存器的 TRACE_IOEN 和 TRACE_MODE[1:0]位来分配跟踪引脚。默认时，跟踪脚是不分配的。

此寄存器被映射到外部 PPB 并且被 PORESET 所复位(系统复位不复位此寄存器)。调试器可以在系统复位的状态下写该寄存器。

表 171 灵活的跟踪引脚分配

<table>
<tr><th colspan="2">DBGMCU_CR 寄存器</th><th rowspan="2">引脚用途</th><th colspan="6">跟踪引脚分配</th></tr>
<tr><th>TRACE_IOEN</th><th>TRACE_MODE[1:0]</th><th>PB3/ JTDO/ TRACESWO</th><th>PE2/ TRACECK</th><th>PE3/ TRACED[0]</th><th>PE4/ TRACED[1]</th><th>PE5/ TRACED[2]</th><th>PE6/ TRACED[3]</th></tr>
<tr><td>0</td><td>XX</td><td>无跟踪(默认状态)</td><td>释放(1)</td><td colspan="5" rowspan="2">释放(可用作普通 I/O 口)</td></tr>
<tr><td>1</td><td>00</td><td>异步跟踪</td><td>TRACESWO</td></tr>
<tr><td>1</td><td>01</td><td>同步跟踪 1 位</td><td rowspan="2">释放(1)</td><td>TRACECK</td><td>TRACED[0]</td><td colspan="3">释放(可用作普通 I/O 口)</td></tr>
<tr><td>1</td><td>10</td><td>同步跟踪 2 位</td><td>TRACECK</td><td>TRACED[0]</td><td>TRACED[1]</td><td colspan="2">释放(可用作普通 I/O 口)</td></tr>
</table>

1	11	同步跟踪 4 位		TRACECK	TRACE D[0]	TRACE D[1]	TRACE D[2]	TRACE D[3]
注释(1)：使用串行调试接口时，此引脚被释放，使用 JTAG 调试接口时，此引脚用作 JTDO。								

注意： *TUIP 的输入时钟 TRACECLKIN 默认接地。所以在比特位 TRACE_IOEN 被置位后，HCLK 需要两个时钟周期。*

然后，调试器可以通过写 TPIU 的 SPP_R(Selected Pin Protocol)寄存器的 PROTOCOL[1:0]位来配置跟踪模式。

- PROTOCOL=00:跟踪模式(同步)
- PROTOCOL=01 或 10：串行模式(曼彻斯特或 NRZ 编码)。默认状态为 01

然后通过写 TPIU 的 CPSPS_R(Current Sync Port Size)寄存器的位[3:0]来配置跟踪端口的大小。

- 0x1：1 个引脚(默认)
- 0x2：2 个引脚
- 0x8：4 个引脚

28.17.3 TPUI 格式器

协议格式器输出 16 个字节组成的帧：

- 7 个数据字节
- 8 个多用途字节，由以下部分组成：
  - 1 位(LSB)用来区分数据字节(0)和 ID 字节(1)。
  - 7 位(MSB)可以作为数据或跟踪源 ID 的变化。
- 1 个辅助字节，其中的每个位都对应于 8 个多用途字节中的一个：
  - 如果对应的多用途字节是数据字节，那么这个位是数据的比特 0 位。
  - 如果对应字节是 ID 字节，这个位表明 ID 变化何时生效。

注意： *更多信息，请参考 ARM CoreSight Architecture Specification v1.0(ARM IHI 0029B)*

28.17.4 TPUI 帧异步包

TPUI 会产生两种类型的同步包：

- 帧同步包(或全字同步包)

该包为 0x7F_FF_FF_FF(LSB 先发)，这个序列只有在 0x7F 没有被作为 ID 源编码的时候才能使用。

该包在帧之间周期性地输出。

在连续模式里，一旦同步帧被发现，TPA 必须抛弃所有这些帧。

- 半字同步包

该包为：0x7F_FF(LSB 先发)。

它在帧之间或帧内周期性的输出。

这些包只存在于连续模式中，并且使能 TPA 检测 IDLE 模式下的 TRACE 口(无 TRACE 被捕捉)。当被 TPA 检测到时，必须将其抛弃。

28.17.5 同步帧包的发送

由于内核的 TPIU 内没有同步计数寄存器，因此同步的触发只能由 DWT 产生。参 DWT Control Register 寄存器(SYNCTAP[11:10]位)和 DWT Current PC Sampler Cycle Count 寄存器。

TPUI 帧同步包(0x7F_FF_FF_FF)在下列情况时被发送：

- 在每个 TPIU 复位释放后。复位信号同步于 TRACECLKIN 时钟的上升沿释放，这意味着当 DBGMCU_CFG 寄存器的 TRACE_IOEN 位被置位时，同步包就被发送。这种情况下，包 0x7F_FF_FF_FF 后面不跟任何格式的包。
- 在每个 DWT 触发时(假设已事先设置好 DWT)，有以下两种情况：
  - 如果 ITM 的 SYNENA 位=0，只发送字 0x7F_FF_FF_FF，后面不跟任何格式的数据包。
  - 如果 ITM 的 SYNENA 位=1，ITM 同步包将跟在(0x80_00_00_00_00_00)后面，由 TPUI 编排格式(加上跟踪源 ID)。

28.17.6 同步模式

跟踪输出数据的引脚数可以为 4 个，2 个或者 1 个，由 TRACED(3:0)。

配置时钟输出到调试器(TRACECK)TRACECLKIN 在内部被驱动，并仅当使用 TRACE 时和 HCLK 相连接。

注意： *在此类同步模式中，不需要提供稳定的时钟频率。*

TRACE 的 I/O 端口(包括 TRACECK)由 TRACLKIN 的上升沿驱动(等同于 HCLK)。因此，TRACECK 的输出频率等于 HCLK/2。

28.17.7 异步模式

调试模块提供一个低成本的，只使用一个引脚的跟踪数据输出功能，即使用异步输出引脚 TRACESWO。但显然，这样的输出数据带宽是有限的。

TRACESWO 引脚与 JTDO 引脚复用，只在 SW-DP 调试接口有效，因此 W55MH32 的所有封装都提供这种功能。

异步模式需要 TRACECLKIN 引脚有平稳的频率提供。对标准的 UART(NRZ)捕捉机制来说，需要 5%的正确度。曼彻斯特编码可放宽到 10%。

28.17.8 TRACECLKIN 在 W55MH32 内部的连接

TRACECLKIN 输入在 W55MH32 内部与 HCLK 相连接。这意味着在使用异步跟踪模式时，应用程序应限制使用时间帧保证 CPU 频率的稳定。

注意： *重要：当使用异步跟踪功能时需注意：*

W55MH32 微控制器的初时时钟是内部 RC 振荡器，此振荡器在复位状态下的频率与复位后的频率不同。这是由于 RC 校准在复位状态下使用初始值，而这个值在复位释放后会更新。

因此，不应该在系统复位状态下激活跟踪端口分析器(Trace Port Analyzer)的跟踪功能(置位 TRACE_IOEN)。因为在复位状态下的同步帧包的比特宽度与复位后的包不同。

28.17.9 TPIU 寄存器

只有当 Debug Exception and Monitor Control(DEMCR)寄存器的 TRCENA 位被置位时，TPIUAPB 寄存器才可以被读写。否则寄存器读出值为 0(这一位的输出使能 TPIU 的 PCLK)。

表 172 重要的 TPIU 寄存器

地址	寄存器	描述
0xE004 0004	Current port size	跟踪端口的长度：位 0：端口长度为 1 位 1：端口长度为 2 位 2：端口长度为 3，不支持位 3：端口长度为 4 四个比特中只能同时置位一个比特。 默认状态下，端口长度为 1(0x0000 0001)
0xE004 00F0	Selected pin protocol	跟踪端口协议的选择：位 1:0= 00：同步跟踪模式 01：串行输出 - 曼彻斯特编码(默认值)10：串行输出 - NRZ 11：未定义
0xE004 0304	Formatter and flush control	位 31-9：总是 0 位 8 = Trigln：总是 1，指示触发器位 7-4：总是 0 位 3-2：总是 0 位 1 = EnFCont：同步模式下(SelectPinProtocol 寄存器的 Bit1：0 为 00)，此比特位强制为 1，连续模式下格式器被自动使能。异步模式下(SelectPinProtocol 寄存器的 Bit1：0 不为 00)，此比特可以被置位或复位来选择是否使能格式器。 位 0：总是 0 默认值为 0x102 注意：在同步模式下，由于 TRACECTL 信号没有外部引脚，因此格式器会在连续模式下自动使能。这意味着格式器会插入一些控制包来识别跟踪包的源。
0xE004 0300	Formatter and flush status	没有在 Cortex-M3 中使用，读出值始终为 0x0000 0008

28.17.10配置的例子

- 设置 Debug Exception and Monitor Control 寄存器的 TRCENA 位;
- 在 TPIU Current Port Size 寄存器中写入期望值(默认是 0x1，指示端口长度为 1bit);
- 向 TPIU Formatter and Flush Control 寄存器中写入 0x102(默认值);
- 写 TPIU Select Pin Protocol 寄存器，选择同步或异步模式。例如写 0x2 选择 NRZ 编码的异步模式(类似 URAT);
- 向 DBGMCU Control 寄存器写入 0x20(置位 IO_TRACEN)，为异步模式分配 TRACE 的 I/O 口。此时 TPIU 将发出一个同步包(FF_FF_FF_7F);
- 配置 ITM 并且写 ITMStimulus 寄存器输出数据。

28.18 DBG 寄存器地址映象

下列表格归纳了调试寄存器。

表 173 DBG-寄存器和复位值

地址	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	DBGMCU_IDCODE	REV_ID																保留				DEV_ID											
	复位值(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x

地址	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2004	DBGMCU_CR	保留										DBG_CAN2_STOP	DBG_TIM8_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_I2C2_SMBUS_TIMEOU	DBS_I2C1_SMBUS_TIMEOU	DBG_CAN1_STOP	DBG_TIM4_STOP	DBT_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE [1:0]		TRACE_IOEN	保留		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	复位值											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0

(1) 复位值详情参见 28.6.1 节-微控制器设备 ID 编码。

29 文档历史信息

版本	日期	说明
Ver. 1.0.0	2024-12-02	第一版发布
Ver. 1.0.1	2025-07-16	修复图片、描述、排版错误

技术支持: support@wiznet.hk
销售&代理: sales@wiznet.hk
更多信息，请登录 http://www.wiznet.io