RM0367
参考手册

基于超低功耗 STM32L0x3 高级 ARM® 的 32 位 MCU

前言

本参考手册面向应用开发人员，提供有关使用 STM32L0x3 微控制器存储器与外设的完整信息。

STM32L0x3 构成一个微控制器系列，各产品具有不同的存储器大小、封装和外设。

有关订购信息以及器件的机械与电气特性，请参见相应的数据手册。

有关 ARM® Cortex®-M0+ 内核的信息，请参见 *Cortex®-M0+ 技术参考手册*。

相关文档

- Cortex®-M0+ 技术参考手册，可从 www.arm.com 获取。
- STM32L0 系列 Cortex®-M0+ 编程手册 (PM0223)。
- STM32L0x3 数据手册。

目录

1 文档约定 50

1.1 寄存器相关缩写词列表 50

1.2 词汇表 50

1.3 外设可用性 51

1.4 产品类别定义 51

2 系统和存储器概述 54

2.1 系统架构 54

2.1.1 S0：Cortex® 总线 55

2.1.2 S1：DMA 总线 55

2.1.3 总线矩阵 55

AHB/APB 总线桥 55

2.2 存储器组织结构 56

2.2.1 简介 56

2.2.2 存储器映射和寄存器边界地址 57

2.3 嵌入式 SRAM 60

2.4 自举配置 60

存储区交换（仅限第 5 类器件）. 61

物理重映射 61

内部自举程序 61

3 Flash 程序存储器和数据 EEPROM (FLASH) 62

3.1 简介 62

3.2 NVM 主要特性 62

3.3 NVM 功能说明 63

3.3.1 NVM 的构成 63

3.3.2 双存储区自举功能 67

3.3.3 读取 NVM 68

读取协议 68

CPU 频率/工作模式/NVM 读取时间之间的关系 69

数据缓冲 70

3.3.4 写入/擦除 NVM 76

写入/擦除协议 76

解锁/锁定操作 77

NVM 写入/擦除操作的详细说明 79
对半页 Flash 程序存储器执行并行写入操作 85
状态寄存器 88
3.4 存储器保护 89
3.4.1 RDP（读保护） 90
3.4.2 PcROP（专有代码读取保护） 91
3.4.3 防止意外的写入/擦除操作 93
3.4.4 写入/擦除保护管理 94
3.4.5 保护错误 94
写保护错误标志 (WRPERR) 94
读错误 (RDERR) 94
3.5 NVM 中断 95
3.5.1 硬性故障 95
3.6 存储器接口管理 96
3.6.1 操作优先级和流程 96
读取 96
写入/擦除 96
选项字节加载 96
3.6.2 操作序列 97
执行写入操作时请求读取数据 97
执行写入操作时请求取指 97
执行写入操作时请求执行另一写入操作 97
3.6.3 执行读取操作时更改等待周期数 97
3.6.4 掉电 98
3.7 Flash 寄存器说明 98
对寄存器执行读取操作 98
对寄存器执行写入操作 98
3.7.1 访问控制寄存器 (FLASH_ACR) 99
3.7.2 编程和擦除控制寄存器 (FLASH_PECR) 100
3.7.3 掉电密钥寄存器 (FLASH_PDKEYR) 103
3.7.4 PECR 解锁密钥寄存器 (FLASH_PEKEYR) 103
3.7.5 编程和擦除密钥寄存器 (FLASH_PRGKEYR) 104
3.7.6 选项字节解锁密钥寄存器 (FLASH_OPTKEYR) 104
3.7.7 状态寄存器 (FLASH_SR) 105
3.7.8 选项字节寄存器 (FLASH_OPTR) 107
3.7.9 写保护寄存器 1 (FLASH_WRPROT1) 109
3.7.10 写保护寄存器 2 (FLASH_WRPROT2) 110
3.7.11 Flash 寄存器映射 111

3.8 选项字节 112
3.8.1 选项字节说明 112
3.8.2 加载保护标志时发生不匹配 113
3.8.3 通过软件重载选项字节 113

4 循环冗余校验计算单元（CRC） 114
4.1 简介 114
4.2 CRC 主要特性 114
4.3 CRC 功能说明 114
多项式可编程 115
4.4 CRC 寄存器 116
4.4.1 数据寄存器 (CRC_DR) 116
4.4.2 独立数据寄存器 (CRC_IDR) 116
4.4.3 控制寄存器 (CRC_CR) 117
4.4.4 CRC 初始值 (CRC_INIT) 118
4.4.5 CRC 多项式 (CRC_POL) 118
4.4.6 CRC 寄存器映射 119

5 防火墙 (FW) 120
5.1 简介 120
5.2 防火墙主要特性 120
5.3 防火墙功能说明 121
5.3.1 防火墙 AMBA 总线监听 121
5.3.2 功能要求 121
调试注意事项 121
写保护 122
中断管理 122
5.3.3 防火墙段 122
代码段 122
非易失性数据段 122
易失性数据段 122
5.3.4 段访问方式及其属性 123
不同防火墙状态下的段访问 123
段属性 123
5.3.5 防火墙初始化 124
5.3.6 防火墙状态 125
打开防火墙 126
关闭防火墙 126

5.4 防火墙寄存器 127
5.4.1 代码段起始地址寄存器 (FW_CSSA) 127
5.4.2 代码段长度寄存器 (FW_CSL) 127
5.4.3 非易失性数据段起始地址寄存器 (FW_NVDSSA) 128
5.4.4 非易失性数据段长度寄存器 (FW_NVDSL) 128
5.4.5 易失性数据段起始地址寄存器 (FW_VDSSA) 129
5.4.6 易失性数据段长度寄存器 (FW_VDSL) 129
5.4.7 配置寄存器 (FW_CR) 130
5.4.8 防火墙寄存器映射 131

6 电源控制 (PWR) 132
6.1 电源 132
6.1.1 独立 A/D 和 DAC 转换器电源和参考电压 133
在具有多于 64 个引脚和 UFBGA64 的封装上 133
在 64 引脚及以下的封装上（除了 BGA 封装） 133
6.1.2 独立 LCD 供电 134
6.1.3 RTC 和 RTC 备份寄存器 134
RTC 寄存器访问 134
6.1.4 调压器 134
6.1.5 动态电压调节管理 135
范围 1 135
范围 2 和 3 135
6.1.6 动态电压调节配置 136
6.1.7 当 VDD 降至 1.71 V 以下时的调压器和时钟管理 136
6.1.8 修改 VCORE 范围时的调压器和时钟管理 137
6.1.9 当 VDD 介于 1.71 V 到 2.0 V 之间时的电压范围和限制 137
6.2 电源监控器 137
6.2.1 上电复位 (POR)/掉电复位 (PDR) 139
6.2.2 欠压复位 (BOR) 139
6.2.3 可编程电压检测器 (PVD) 140
6.2.4 内部参考电压 (VREFINT) 141
6.3 低功耗模式 142
6.3.1 低功耗模式下时钟的行为 143
睡眠和低功耗睡眠模式 143
停止和待机模式 143
6.3.2 降低系统时钟速度 143
6.3.3 外设时钟门控 143
6.3.4 低功耗运行模式（LP 运行） 144

进入低功耗运行模式 144
退出低功耗运行模式 144
6.3.5 进入低功耗模式 144
6.3.6 退出低功耗模式 145
6.3.7 睡眠模式 145
睡眠模式下的 I/O 状态 145
进入睡眠模式 145
退出睡眠模式 145
6.3.8 低功耗睡眠模式（LP 睡眠） 147
低功耗睡眠模式下的 I/O 状态 147
进入低功耗睡眠模式 147
退出低功耗睡眠模式 147
6.3.9 停止模式 149
低功耗睡眠模式下的 I/O 状态 149
进入停止模式 149
退出停止模式 149
6.3.10 待机模式 151
待机模式下的 I/O 状态 151
进入待机模式 151
退出待机模式 151
调试模式 152
6.3.11 使用 RTC 和比较器从停止和待机模式唤醒器件 152
从停止模式的 RTC 自动唤醒 (AWU) 153
从待机模式的 RTC 自动唤醒 (AWU) 153
从停止模式的比较器自动唤醒 (AWU) 153
6.4 电源控制寄存器 154
6.4.1 PWR 电源控制寄存器 (PWR_CR) 154
6.4.2 PWR 电源控制/状态寄存器 (PWR_CSR) 157
6.4.3 PWR 寄存器映射 159

7 复位和时钟控制 (RCC) 160
7.1 复位 160
7.1.1 系统复位 160
软件复位 160
低功耗管理复位 160
选项字节加载器复位 160
7.1.2 电源复位 161
7.1.3 RTC 和备份寄存器复位 161
7.2 时钟 161

7.2.1 HSE 时钟 164
外部源（HSE 旁路） 165
外部晶振/陶瓷谐振器（HSE 晶振） 165
7.2.2 HSI16 时钟 165
校准 165
7.2.3 MSI 时钟 166
校准 166
7.2.4 HSI48 时钟 166
7.2.5 PLL 167
7.2.6 LSE 时钟 167
外部源（LSE 旁路） 167
7.2.7 LSI 时钟 168
LSI 测量 168
7.2.8 系统时钟 (SYSCLK) 选择 168
7.2.9 系统时钟源频率与电压范围 168
7.2.10 HSE 时钟安全系统(CSS) 169
7.2.11 LSE 时钟安全系统 169
7.2.12 RTC 和 LCD 时钟 169
7.2.13 看门狗时钟 170
7.2.14 时钟输出功能 170
7.2.15 基于 TIM21 的内部/外部时钟测量 170
7.2.16 TIM2/TIM21/TIM22 独立于系统时钟的时钟源 171
7.3 RCC 寄存器 172
7.3.1 时钟控制寄存器 (RCC_CR) 172
7.3.2 内部时钟源校准寄存器 (RCC_ICSCR) 175
7.3.3 时钟恢复 RC 寄存器 (RCC_CRRCR) 176
7.3.4 时钟配置寄存器 (RCC_CFGR) 177
7.3.5 时钟中断使能寄存器 (RCC_CIER) 179
7.3.6 时钟中断标志寄存器 (RCC_CIFR) 181
7.3.7 时钟中断清零寄存器 (RCC_CICR) 182
7.3.8 GPIO 复位寄存器 (RCC_IOPRSTR) 183
7.3.9 AHB 外设复位寄存器 (RCC_AHBRSTR) 184
7.3.10 APB2 外设复位寄存器 (RCC_APB2RSTR) 185
7.3.11 APB1 外设复位寄存器 (RCC_APB1RSTR) 187
7.3.12 GPIO 时钟使能寄存器 (RCC_IOPENR) 189
7.3.13 AHB 外设时钟使能寄存器 (RCC_AHBENR) 190
7.3.14 APB2 外设时钟使能寄存器 (RCC_APB2ENR) 191
7.3.15 APB1 外设时钟使能寄存器 (RCC_APB1ENR) 193

7.3.16 睡眠模式下的 GPIO 时钟使能寄存器 (RCC_IOPSMENR) 196
7.3.17 睡眠模式下的 AHB 外设时钟使能寄存器 (RCC_AHBSMENR) 197
7.3.18 睡眠模式下的 APB2 外设时钟使能寄存器 (RCC_APB2SMENR) 198
7.3.19 睡眠模式下的 APB1 外设时钟使能寄存器 (RCC_APB1SMENR) 199
7.3.20 时钟配置寄存器 (RCC_CCIPR) 202
7.3.21 控制/状态寄存器 (RCC_CSR) 203
7.3.22 RCC 寄存器映射 207

8 时钟恢复系统 (CRS) 210
8.1 简介 210
8.2 CRS 主要特性 210
8.3 CRS 功能说明 211
8.3.1 CRS 框图 211
8.3.2 同步输入 211
8.3.3 频率误差测量 212
8.3.4 频率误差评估和自动微调 213
8.3.5 CRS 初始化和配置 213
RELOAD 值 213
FELIM 值 214
8.4 CRS 低功耗模式 214
8.5 CRS 中断 214
8.6 CRS 寄存器 215
8.6.1 CRS 控制寄存器 (CRS_CR) 215
8.6.2 CRS 配置寄存器 (CRS_CFGR) 216
8.6.3 CRS 中断和状态寄存器 (CRS_ISR) 217
8.6.4 CRS 中断标志清零寄存器 (CRS_ICR) 219
8.6.5 CRS 寄存器映射 220

9 通用 I/O (GPIO) 221
9.1 简介 221
9.2 GPIO 主要特性 221
9.3 GPIO 功能描述 221
9.3.1 通用 I/O (GPIO) 224
9.3.2 I/O 引脚复用功能复用器和映射 224
9.3.3 I/O 端口控制寄存器 225
9.3.4 I/O 端口数据寄存器 225

9.3.5 I/O 数据位操作 225
9.3.6 GPIO 锁定机制 225
9.3.7 I/O 复用功能输入/输出 226
9.3.8 外部中断线/唤醒线 226
9.3.9 输入配置 226
9.3.10 输出配置 227
9.3.11 复用功能配置 227
9.3.12 模拟配置 228
9.3.13 将 HSE 或 LSE 振荡器引脚用作 GPIO 229
9.3.14 在 RTC 电源域中使用 GPIO 引脚 229
9.4 GPIO 寄存器 229
9.4.1 GPIO 端口模式寄存器 (GPIOx_MODER)（x =A..E 和 H） 229
9.4.2 GPIO 端口输出类型寄存器 (GPIOx_OTYPER)（x = A..E 和 H） 230
9.4.3 GPIO 端口输出速度寄存器 (GPIOx_OSPEEDR)（x = A..E 和 H） . . . 230
9.4.4 GPIO 端口上拉/下拉寄存器 (GPIOx_PUPDR)（x = A..E 和 H） 231
9.4.5 GPIO 端口输入寄存器 (GPIOx_OTYPER)（x = A..E 和 H） 231
9.4.6 GPIO 端口输出数据寄存器 (GPIOx_ODR)（x = A..E 和 H） 232
9.4.7 GPIO 端口置位/复位寄存器 (GPIOx_BSRR)（x = A..E 和 H） 232
9.4.8 GPIO 端口配置锁定寄存器 (GPIOx_LCKR)（x = A..E 和 H） 233
9.4.9 GPIO 复用功能低位寄存器 (GPIOx_AFRL)（x = A..E 和 H） 234
9.4.10 GPIO 复用功能高位寄存器 (GPIOx_AFRH)（x = A..E 和 H） 235
9.4.11 GPIO 端口位复位寄存器 (GPIOx_BRR)（x =A..E 和 H） 235
9.4.12 GPIO 寄存器映射 236

10 系统配置控制器 (SYSCFG) 238
10.1 简介 238
10.2 SYSCFG 寄存器 239
10.2.1 SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1) 239
10.2.2 SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2) 240
10.2.3 参考控制和状态寄存器 (SYSCFG_CFGR3) 241
10.2.4 SYSCFG 外部中断配置寄存器 1 (SYSCFG_EXTICR1) 243
10.2.5 SYSCFG 外部中断配置寄存器 2 (SYSCFG_EXTICR2) 243
10.2.6 SYSCFG 外部中断配置寄存器 3 (SYSCFG_EXTICR3) 244
10.2.7 SYSCFG 外部中断配置寄存器 4 (SYSCFG_EXTICR4) 244
10.2.8 SYSCFG 寄存器映射 245

11 直接存储器访问控制器 (DMA) 246

11.1 简介 246
11.2 DMA 主要特性 246
11.3 DMA 功能说明 247
11.3.1 DMA 事务 247
11.3.2 仲裁器 248
11.3.3 DMA 通道 248
数据大小可编程 248
指针递增 248
通道配置流程 249
循环模式 249
存储器到存储器模式 249
11.3.4 可编程数据宽度、数据对齐和字节存储次序 250
解决 AHB 外设无法支持字节或半字写操作的问题 251
11.3.5 错误管理 251
11.3.6 DMA 中断 251
11.3.7 DMA 请求映射 252
DMA 控制器 252
11.4 DMA 寄存器 254
11.4.1 DMA 中断状态寄存器 (DMA_ISR) 254
11.4.2 DMA 中断标志清零寄存器 (DMA_IFCR) 255
11.4.3 DMA 通道 x 配置寄存器 (DMA_CCRx)
（x = 1..7，其中 x 表示通道编号） 256
11.4.4 DMA 通道 x 数据数寄存器 (DMA_CNDTRx)
（x = 1..7，其中 x 表示通道编号） 258
11.4.5 DMA 通道 x 外设地址寄存器 (DMA_CPARx)
（x = 1..7，其中 x 表示通道编号） 258
11.4.6 DMA 通道 x 存储器地址寄存器 (DMA_CMARx)
（x = 1..7，其中 x 表示通道编号） 259
11.4.7 DMA 通道选择寄存器 (DMA_CSELR) 260
11.4.8 DMA 寄存器映射 262

12 嵌套向量中断控制器 (NVIC) 264
12.1 主要特性 264
12.2 SysTick 校准值寄存器 264
12.3 中断和异常向量 264

13 扩展中断和事件控制器 (EXTI) 266
13.1 简介 266

13.2 EXTI 主要特性 266
13.3 EXTI 功能说明 266
13.3.1 EXTI 框图 267
13.3.2 唤醒事件管理 267
13.3.3 外设异步中断 267
13.3.4 硬件中断选择 268
13.3.5 硬件事件选择 268
13.3.6 软件中断/事件选择 268
13.4 EXTI 中断/事件线映射 269
13.5 EXTI 寄存器 271
13.5.1 EXTI 中断屏蔽寄存器 (EXTI_IMR) 271
13.5.2 EXTI 事件屏蔽寄存器 (EXTI_EMR) 271
13.5.3 EXTI 上升沿触发选择寄存器 (EXTI_RTSR) 272
13.5.4 下降沿触发选择寄存器 (EXTI_FTSR) 273
13.5.5 EXTI 软件中断事件寄存器 (EXTI_SWIER) 274
13.5.6 EXTI 挂起寄存器 (EXTI_PR) 275
13.5.7 EXTI 寄存器映射 276

14 模数转换器（ADC） 277
14.1 简介 277
14.2 ADC 主要特性 277
14.3 ADC 引脚和内部信号 278
14.4 ADC 功能说明 279
14.4.1 ADC 调压器 (ADVREGEN) 279
ADC 内部调压器的模拟参考 280
ADVREG 使能序列 280
ADVREG 禁止序列 280
14.4.2 校准 (ADCAL) 280
校准系数强制软件程序 281
14.4.3 ADC 开关控制（ADEN、ADDIS、ADRDY） 282
14.4.4 ADC 时钟（CKMODE、PRESC[3:0]、LFMEN） 283
低频 284
14.4.5 配置 ADC 284
14.4.6 通道选择（CHSEL、SCANDIR） 284
温度传感器、V_{REFINT} 和 LCD_VLCD1 内部通道 285
14.4.7 可编程采样时间 (SMP) 285
14.4.8 单次转换模式 (CONT=0) 285

14.4.9 连续转换模式 (CONT=1) 286
14.4.10 开始转换 (ADSTART) 286
14.4.11 时序 287
14.4.12 停止正在进行的转换 (ADSTP) 288
14.5 外部触发转换和触发极性（EXTSEL，EXTEN） 288
14.5.1 不连续模式 (DISCEN) 289
14.5.2 可编程分辨率 (RES) - 快速转换模式 290
14.5.3 转换结束、采样阶段结束（EOC、EOSMP 标志） 290
14.5.4 转换序列结束（EOSEQ 标志） 291
14.5.5 时序图示例（单次/连续模式硬件/软件触发） 291
14.6 数据管理 293
14.6.1 数据寄存器和数据对齐（ADC_DR、ALIGN） 293
14.6.2 ADC 溢出（OVR、OVRMOD） 293
14.6.3 在不使用 DMA 的情况下管理转换的数据序列 294
14.6.4 在不使用 DMA 且不发生溢出的情况下管理转换的数据 294
14.6.5 使用 DMA 管理转换的数据 294
DMA 单次模式 (DMACFG=0) 295
DMA 循环模式 (DMACFG=1) 295
14.7 低功耗特性 296
14.7.1 等待模式转换 296
14.7.2 自动关闭模式 (AUTOFF) 296
14.8 模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD） 297
14.9 过采样器 299
14.9.1 过采样时支持的 ADC 工作模式 300
14.9.2 模拟看门狗 300
14.9.3 触发模式 301
14.10 温度传感器和内部参考电压 301
主要特性 302
读取温度 302
使用内部参考电压计算实际的 VDDA 电压 303
将电源相关的 ADC 测量值转换为绝对电压值 303
14.11 VLCD 电压监控 303
14.12 ADC 中断 304
14.13 ADC 寄存器 305
14.13.1 ADC 中断和状态寄存器 (ADC_ISR) 305
14.13.2 ADC 中断使能寄存器 (ADC_IER) 306

14.13.3 ADC 控制寄存器 (ADC_CR) 308
14.13.4 ADC 配置寄存器 1 (ADC_CFGR1) 310
14.13.5 ADC 配置寄存器 2 (ADC_CFGR2) 313
14.13.6 ADC 采样时间寄存器 (ADC_SMPR) 314
14.13.7 ADC 看门狗阈值寄存器 (ADC_TR) 315
14.13.8 ADC 通道选择寄存器 (ADC_CHSELR) 316
14.13.9 ADC 数据寄存器 (ADC_DR) 316
14.13.10 ADC 校准系数 (ADC_CALFACT) 317
14.13.11 ADC 通用配置寄存器 (ADC_CCR) 317
14.13.12 ADC 寄存器映射 319

15 数模转换器 (DAC) 321

15.1 简介 321
15.2 DAC1 主要特性 321
15.3 DAC 输出缓冲器使能 323
15.4 DAC 通道使能 323
15.5 单模式功能说明 323
15.5.1 DAC 数据格式 323
15.5.2 DAC 通道转换 324
生成单个LFSR的独立触发 324
生成单个三角波的独立触发 324
15.5.3 DAC 输出电压 325
15.5.4 DAC 触发选择 325
15.6 双模式功能说明 326
15.6.1 DAC 数据格式 326
15.6.2 双模式下的 DAC 通道转换 326
15.6.3 双转换模式说明 326
独立触发（不产生波形） 327
独立触发（生成单个 LFSR） 327
独立触发（生成不同 LFSR） 327
独立触发（生成单个三角波） 328
独立触发（生成不同三角波） 328
同步软件启动 328
同步触发（不产生波形） 328
同步触发（生成单个 LFSR） 329
同步触发（生成不同 LFSR） 329
同步触发（生成单个三角波） 329
同步触发（生成不同三角波） 330

15.6.4 DAC 输出电压 330
15.6.5 DAC 触发选择 330
15.7 生成噪声 330
15.8 生成三角波 331
15.9 DMA 请求 332
DMA 下溢 332
15.10 DAC 寄存器 333
15.10.1 DAC 控制寄存器 (DAC_CR) 333
15.10.2 DAC 软件触发寄存器 (DAC_SWTRIGR) 336
15.10.3 DAC 1 通道 12 位右对齐数据保持寄存器 (DAC_DHR12R1) 337
15.10.4 DAC 1 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L1) 337
15.10.5 DAC 1 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R1) 338
15.10.6 DAC 2 通道 12 位右对齐数据保持寄存器 (DAC_DHR12R2) 338
15.10.7 DAC 2 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L2) 339
15.10.8 DAC 2 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R2) 339
15.10.9 双 DAC 12 位右对齐数据保持寄存器 (DAC_DHR12RD) 340
15.10.10 双 DAC 12 位左对齐数据保持寄存器 (DAC_DHR12LD) 340
15.10.11 双 DAC 8 位右对齐数据保持寄存器 (DAC_DHR8RD) 341
15.10.12 DAC 1 通道 数据输出寄存器 (DAC_DOR1) 341
15.10.13 DAC 2 通道数据输出寄存器 (DAC_DOR2) 342
15.10.14 DAC 状态寄存器 (DAC_SR) 342
15.10.15 DAC 寄存器映射 343

16 比较器 (COMP) 345
16.1 简介 345
16.2 COMP 主要特性 345
16.3 COMP 功能说明 346
16.3.1 COMP 框图 346
16.3.2 COMP 引脚和内部信号 346
16.3.3 COMP 复位和时钟 347
16.3.4 比较器锁定机制 347
16.3.5 功率模式 347
16.4 COMP 中断 347
16.5 COMP 寄存器 348
16.5.1 比较器 1 控制和状态寄存器 (COMP1_CSR) 348
16.5.2 比较器 2 控制和状态寄存器 (COMP2_CSR) 349

16.5.3 COMP 寄存器映射 351

17 液晶显示控制器 (LCD) 352

17.1 简介 352

词汇表 352

17.2 LCD 主要特性 353

17.3 LCD 实现 353

17.4 LCD 功能说明 354

17.4.1 概述 354

17.4.2 频率发生器 355

17.4.3 公用驱动器 356

COM 信号偏置 356

COM 信号占空比 356

8 选 1 多路复用器 358

17.4.4 区段驱动器 359

在 1/4 或 1/8 占空比的情况下 359

闪烁 363

17.4.5 电压发生器和对比度控制 363

LCD 电源 363

LCD 中间电压 363

LCD 驱动器选择 364

外部去耦 365

死区 365

17.4.6 双缓冲存储器 366

17.4.7 COM 和 SEG 多路复用 366

输出引脚与占空比模式 366

小型封装的重映射功能 366

COM 和 SEG 功能与占空比和重映射的关系汇总 367

17.4.8 流程图 371

17.5 LCD 低功耗模式 372

17.6 LCD 中断 372

帧起始 (SOF) 372

更新显示完成 (UDD) 372

17.7 LCD 寄存器 373

17.7.1 LCD 控制寄存器 (LCD_CR) 373

17.7.2 LCD 帧控制寄存器 (LCD_FCR) 374

17.7.3 LCD 状态寄存器 (LCD_SR) 377

17.7.4 LCD 清零寄存器 (LCD_CLR) 379

17.7.5 LCD 显示器存储器 (LCD_RAM) 379
17.7.6 LCD 寄存器映射 380

18 触摸感应控制器（TSC） 382
18.1 简介 382
18.2 TSC 主要特性 382
18.3 TSC 功能说明 383
18.3.1 TSC 框图 383
18.3.2 表面电荷转移采集概述 383
18.3.3 复位和时钟 385
18.3.4 电荷转移采集序列 386
18.3.5 扩频功能 387
18.3.6 最大计数错误 387
18.3.7 采样电容 I/O 和通道 I/O 模式选择 388
18.3.8 采集模式 389
18.3.9 I/O 滞后和模拟开关控制 389
18.4 TSC 低功耗模式 389
18.5 TSC 中断 390
18.6 TSC 寄存器 390
18.6.1 TSC 控制寄存器 (TSC_CR) 390
18.6.2 TSC 中断使能寄存器 (TSC_IER) 392
18.6.3 TSC 中断清零寄存器 (TSC_ICR) 393
18.6.4 TSC 中断状态寄存器 (TSC_ISR) 394
18.6.5 TSC I/O 滞后控制寄存器 (TSC_IOHCR) 394
18.6.6 TSC I/O 模拟开关控制寄存器 (TSC_IOASCR) 395
18.6.7 TSC I/O 采样控制寄存器 (TSC_IOSCR) 395
18.6.8 TSC I/O 通道控制寄存器 (TSC_IOCCRTSC_IOCCR) 396
18.6.9 TSC I/O 组控制状态寄存器 (TSC_IOGCSR) 397
18.6.10 TSC I/O 组 x 计数器寄存器 (TSC_IOGxCR) (x = 1..8) 397
18.6.11 TSC 寄存器映射 398

19 高级加密标准硬件加速器 (AES) 400
19.1 简介 400
19.2 AES 主要特性 400
19.3 AES 功能说明 401
19.4 加密和生成密钥 402

19.5 AES 链接算法 402
19.5.1 电子密码本 (ECB) 402
19.5.2 密码块链接 (CBC) 404
给定消息的挂起模式 405
19.5.3 计数器模式 (CTR) 407
CTR 模式下的挂起模式 408
19.6 数据类型 408
19.7 工作模式 410
19.7.1 模式 1：加密 410
19.7.2 模式 2：密钥生成 411
19.7.3 模式 3：解密 411
19.7.4 模式 4：密钥生成和解密 412
19.8 AES DMA 接口 413
19.9 错误标志 414
19.10 处理时间 414
19.11 AES 中断 414
19.12 AES 寄存器 415
19.12.1 AES 控制寄存器 (AES_CR) 415
19.12.2 AES 状态寄存器 (AES_SR) 417
19.12.3 AES 数据输入寄存器 (AES_DINR) 418
19.12.4 AES 数据输出寄存器 (AES_DOUTR) 418
19.12.5 AES 密钥寄存器 0 (AES_KEYR0) (LSB: key [31:0]) 419
19.12.6 AES 密钥寄存器 1 (AES_KEYR1) (Key[63:32]) 419
19.12.7 AES 密钥寄存器 2 (AES_KEYR2) (Key [95:64]) 420
19.12.8 AES 密钥寄存器 3 (AES_KEYR3) (MSB: key[127:96]) 420
19.12.9 AES 初始化向量寄存器 0 (AES_IVR0)（LSB：IVR[31:0]） 421
19.12.10 AES 初始化向量寄存器 1 (AES_IVR1) (IVR[63:32]) 421
19.12.11 AES 初始化向量寄存器 2 (AES_IVR2) (IVR[95:64]) 422
19.12.12 AES 初始化向量寄存器 3 (AES_IVR3) (MSB: IVR[127:96]) 422
19.12.13 AES 寄存器映射 423

20 随机数发生器 (RNG) 424
20.1 简介 424
20.2 RNG 主要特性 424
20.3 RNG 功能说明 424
20.3.1 操作 425

20.3.2 错误管理 425
如果 CEIS 位的值为 1（时钟错误）. 425
如果 SEIS 位的值为 1（种子错误）. 425
20.4 RNG 寄存器 426
20.4.1 RNG 控制寄存器 (RNG_CR) 426
20.4.2 RNG 状态寄存器 (RNG_SR) 427
20.4.3 RNG 数据寄存器 (RNG_DR) 428
20.4.4 RNG 寄存器映射 428

21 通用定时器 (TIM2/TIM3) 429
21.1 TIM2/TIM3 简介 429
21.2 TIM2/TIM3 主要特性 429
21.3 TIM2/TIM3 功能说明 431
21.3.1 时基单元 431
预分频器说明 431
21.3.2 计数器模式 433
递增计数模式 433
递减计数模式 436
中心对齐模式（递增/递减计数）. 439
21.3.3 时钟选择 443
内部时钟源 (CK_INT) 443
外部时钟源模式 1 444
外部时钟源模式 2 446
21.3.4 捕获/比较通道 447
21.3.5 输入捕获模式 449
21.3.6 PWM 输入模式 450
21.3.7 强制输出模式 451
21.3.8 输出比较模式 451
21.3.9 PWM 模式 452
PWM 边沿对齐模式 453
递减计数配置 454
PWM 中心对齐模式 454
21.3.10 单脉冲模式 456
特殊情况：OCx 快速使能：. 457
21.3.11 发生外部事件时清除 OCxREF 信号 457
21.3.12 编码器接口模式 458
21.3.13 定时器输入异或功能 460
21.3.14 定时器与外部触发同步 460

从模式：复位模式 460
从模式：门控模式 461
从模式：触发模式 462
从模式：外部时钟模式 2 + 触发模式 463
21.3.15 定时器同步 464
将一个定时器用作另一个定时器的预分频器 464
使用一个定时器使能另一个定时器 465
使用一个定时器启动另一个定时器 467
使用一个外部触发同步的启动 2 个定时器 468
21.3.16 调试模式 469
21.4 TIM2/TIM3 寄存器 470
21.4.1 TIMx 控制寄存器 1 (TIMx_CR1) 470
21.4.2 TIMx 控制寄存器 2 (TIMx_CR2) 472
21.4.3 TIMx 从模式控制寄存器 (TIMx_SMCR) 473
21.4.4 TIMx DMA/ 中断使能寄存器 (TIMx_DIER) 475
21.4.5 TIMx 状态寄存器 (TIMx_SR) 476
21.4.6 TIMx 事件生成寄存器 (TIMx_EGR) 478
21.4.7 TIMx 捕获/比较模式寄存器 1 (TIMx_CCMR1) 479
输出比较模式 479
输入捕获模式 481
21.4.8 TIMx 捕获/比较模式寄存器 2 (TIMx_CCMR2) 482
输出比较模式 482
输入捕获模式 483
21.4.9 TIMx 捕获/比较使能寄存器 (TIMx_CCER) 483
21.4.10 TIMx 计数器 (TIMx_CNT) 485
21.4.11 TIMx 预分频器 (TIMx_PSC) 485
21.4.12 TIMx 自动重载寄存器 (TIMx_ARR) 485
21.4.13 TIMx 捕获/比较寄存器 1 (TIMx_CCR1) 486
21.4.14 TIMx 捕获/比较寄存器 2 (TIMx_CCR2) 486
21.4.15 TIMx 捕获/比较寄存器 3 (TIMx_CCR3) 487
21.4.16 TIMx 捕获/比较寄存器 4 (TIMx_CCR4) 487
21.4.17 TIMx DMA 控制寄存器 (TIMx_DCR) 488
21.4.18 TIMx 全传输 DMA 地址 (TIMx_DMAR) 488
DMA 连续传送功能使用方法示例 489
21.4.19 TIM2 选项寄存器 (TIM2_OR) 490
21.4.20 TIM3 选项寄存器 (TIM3_OR) 491
21.5 TIMx 寄存器映射 492

22 通用定时器 (TIM21/22) 494
22.1 简介 494
22.2 TIM21/22 主要特性 494
22.2.1 TIM21/22 主要特性 494
22.3 TIM21/22 功能说明 496
22.3.1 时基单元 496
预分频器说明 496
22.3.2 计数器模式 498
递增计数模式 498
递减计数模式 502
中心对齐模式（递增/递减计数） 504
22.3.3 时钟选择 508
内部时钟源 (CK_INT) 508
外部时钟源模式 2 511
22.3.4 捕获/比较通道 512
22.3.5 输入捕获模式 514
22.3.6 PWM 输入模式 515
22.3.7 强制输出模式 516
22.3.8 输出比较模式 516
22.3.9 PWM 模式 518
PWM 中心对齐模式 519
中心对齐模式使用建议 520
22.3.10 发生外部事件时清除 OCxREF 信号 521
22.3.11 单脉冲模式 522
特殊情况：OCx 快速使能 523
22.3.12 编码器接口模式 523
22.3.13 TIM21/22 外部触发同步 525
从模式：复位模式 525
从模式：门控模式 526
从模式：触发模式 527
22.3.14 定时器同步 (TIM21/22) 528
22.3.15 调试模式 528
22.4 TIM21/22 寄存器 529
22.4.1 TIM21/22 控制寄存器 1 (TIMx_CR1) 529
22.4.2 TIM21/22 控制寄存器 2 (TIMx_CR2) 530
22.4.3 TIM21/22 从模式控制寄存器 (TIMx_SMCR) 531
22.4.4 TIM21/22 中断使能寄存器 (TIMx_DIER) 533
22.4.5 TIM21/22 状态寄存器 (TIMx_SR) 534

22.4.6 TIM21/22 事件生成寄存器 (TIMx_EGR) 536
22.4.7 TIM21/22 捕获/比较模式寄存器 1 (TIMx_CCMR1) 537
输出比较模式 537
输入捕获模式 539
22.4.8 TIM21/22 捕获/比较使能寄存器 (TIMx_CCER) 540
22.4.9 TIM21/22 计数器 (TIMx_CNT) 541
22.4.10 TIM21/22 预分频器 (TIMx_PSC) 541
22.4.11 TIM21/22 自动重载寄存器 (TIMx_ARR) 541
22.4.12 TIM21/22 捕获/比较寄存器 1 (TIMx_CCR1) 542
22.4.13 TIM21/22 捕获/比较寄存器 2 (TIMx_CCR2) 542
22.4.14 TIM21 选项寄存器 (TIM21_OR) 543
22.4.15 TIM22 选项寄存器 (TIM22_OR) 544
22.4.16 TIM21/22 寄存器映射 545

23 基本定时器 (TIM6/7) 547
23.1 简介 547
23.2 TIM6/7 主要特性 547
23.3 TIM6/7 功能说明 548
23.3.1 时基单元 548
预分频器说明 548
23.3.2 计数模式 550
23.3.3 时钟源 553
23.3.4 调试模式 554
23.4 TIM6/7 寄存器 555
23.4.1 TIM6/7 控制寄存器 1 (TIMx_CR1) 555
23.4.2 TIM6/7 控制寄存器 2 (TIMx_CR2) 556
23.4.3 TIM6/7 DMA/中断使能寄存器 (TIMx_DIER) 556
23.4.4 TIM6/7 状态寄存器 (TIMx_SR) 557
23.4.5 TIM6/7 事件产生寄存器 (TIMx_EGR) 557
23.4.6 TIM6/7 计数器 (TIMx_CNT) 557
23.4.7 TIM6/7 预分频器 (TIMx_PSC) 558
23.4.8 TIM6/7 自动重载寄存器 (TIMx_ARR) 558
23.4.9 TIM6/7 寄存器映射 559

24 低功耗定时器 (LPTIM) 560
24.1 简介 560
24.2 LPTIM 主要特性 560

24.3 LPTIM 实现 560
24.4 LPTIM 功能说明 561
24.4.1 LPTIM 框图 561
24.4.2 LPTIM 复位和时钟 561
24.4.3 干扰滤波器 562
24.4.4 预分频器 563
24.4.5 触发多路复用器 563
24.4.6 工作模式 564
24.4.7 超时功能 565
24.4.8 生成波形 566
24.4.9 寄存器更新 567
24.4.10 计数器模式 567
24.4.11 定时器使能 568
24.4.12 编码器模式 568
24.5 LPTIM 中断 569
24.6 LPTIM 寄存器 570
24.6.1 LPTIM 中断和状态寄存器 (LPTIMx_ISR) 570
24.6.2 LPTIM 中断清零寄存器 (LPTIMx_ICR) 571
24.6.3 LPTIM 中断使能寄存器 (LPTIMx_IER) 572
24.6.4 LPTIM 配置寄存器 (LPTIMx_CFGR) 573
24.6.5 LPTIM 控制寄存器 (LPTIMx_CR) 576
24.6.6 LPTIM 比较寄存器 (LPTIMx_CMP) 577
24.6.7 LPTIM 自动重载寄存器 (LPTIMx_ARR) 577
24.6.8 LPTIM 计数器寄存器 (LPTIMx_CNT) 578
24.6.9 LPTIM 寄存器映射 579

25 独立看门狗 (IWDG) 580
25.1 简介 580
25.2 IWDG 主要特性 580
25.3 IWDG 功能说明 580
25.3.1 IWDG 框图 580
25.3.2 窗口选项 581
使能窗口选项时配置 IWDG 581
禁止窗口选项时配置 IWDG 581
25.3.3 硬件看门狗 581
25.3.4 停止和待机模式下的行为 581
25.3.5 寄存器访问保护 581

25.3.6 调试模式 582
25.4 IWDG 寄存器 582
25.4.1 键寄存器 (IWDG_KR) 582
25.4.2 预分频器寄存器 (IWDG_PR) 583
25.4.3 重载寄存器 (IWDG_RLR) 584
25.4.4 状态寄存器 (IWDG_SR) 585
25.4.5 窗口寄存器 (IWDG_WINR) 586
25.4.6 IWDG 寄存器映射 586

26 系统窗口看门狗 (WWDG) 587
26.1 简介 587
26.2 WWDG 主要特性 587
26.3 WWDG 功能说明 587
26.3.1 使能看门狗 588
26.3.2 控制递减计数器 588
26.3.3 看门狗中断高级特性 588
26.3.4 如何设置看门狗超时 589
26.3.5 调试模式 590
26.4 WWDG 寄存器 590
26.4.1 控制寄存器 (WWDG_CR) 590
26.4.2 配置寄存器 (WWDG_CFR) 591
26.4.3 状态寄存器 (WWDG_SR) 591
26.4.4 WWDG 寄存器映射 592

27 实时时钟 (RTC) 593
27.1 简介 593
27.2 RTC 主要特性 593
27.3 RTC 功能说明 594
27.3.1 RTC 框图 594
27.3.2 RTC 控制的 GPIO 595
27.3.3 时钟和预分频器 596
27.3.4 实时时钟和日历 597
27.3.5 可编程闹钟 597
27.3.6 周期性自动唤醒 598
27.3.7 RTC 初始化和配置 598
RTC 寄存器访问 598
RTC 寄存器写保护 598

日历初始化和配置 599
夏令时 599
编程闹钟 599
编程唤醒定时器 600
27.3.8 读取日历 600
当 RTC_CR 寄存器中的 BYPSHAD 控制位清零时 600
当 RTC_CR 寄存器中的 BYPSHAD 控制位置 1 时（旁路影子寄存器） 600
27.3.9 复位 RTC 601
27.3.10 RTC 同步 601
27.3.11 RTC 参考时钟检测 602
27.3.12 RTC 精密数字校准 602
PREDIV_A<3 条件下的校准 603
验证 RTC 校准 603
动态重校准 604
27.3.13 时间戳功能 604
27.3.14 入侵检测 604
RTC 备份寄存器 604
入侵检测初始化 605
出现入侵事件时生成触发输出 605
入侵事件的时间戳 605
对入侵输入的边沿检测 605
对 RTC_TAMPx 输入的带过滤电平检测 606
27.3.15 校准时钟输出 606
27.3.16 闹钟输出 606
闹钟复用功能输出 606
27.4 RTC 低功耗模式 607
27.5 RTC 中断 607
27.6 RTC 寄存器 608
27.6.1 RTC 时间寄存器 (RTC_TR) 608
27.6.2 RTC 日期寄存器 (RTC_DR) 609
27.6.3 RTC 控制寄存器 (RTC_CR) 610
27.6.4 RTC 初始化和状态寄存器 (RTC_ISR) 613
27.6.5 RTC 预分频器寄存器 (RTC_PRER) 615
27.6.6 RTC 唤醒定时器寄存器 (RTC_WUTR) 616
27.6.7 RTC 闹钟 A 寄存器 (RTC_ALRMAR) 617
27.6.8 RTC 闹钟 B 寄存器 (RTC_ALRMBR) 618
27.6.9 RTC 写保护寄存器 (RTC_WPR) 619
27.6.10 RTC 亚秒寄存器 (RTC_SSR) 619

27.6.11 RTC 平移控制寄存器 (RTC_SHIFTR) 620
27.6.12 RTC 时间戳时间寄存器 (RTC_TSTR) 621
27.6.13 RTC 时间戳日期寄存器 (RTC_TSDR) 622
27.6.14 RTC 时间戳亚秒寄存器 (RTC_TSSSR) 622
27.6.15 RTC 校准寄存器 (RTC_CALR) 623
27.6.16 RTC 入侵配置寄存器 (RTC_TAMPCR) 624
27.6.17 RTC 闹钟 A 亚秒寄存器 (RTC_ALRMASSR) 627
27.6.18 RTC 闹钟 B 亚秒寄存器 (RTC_ALRMBSSR) 628
27.6.19 RTC 选项寄存器 (RTC_OR) 628
27.6.20 RTC 备份寄存器 (RTC_BKPxR) 629
27.6.21 RTC 寄存器映射 630

28 内部集成电路 (I2C) 接口 632
28.1 简介 632
28.2 I2C 主要特性 632
28.3 I2C 特性实现 633
28.4 I2C 功能说明 633
28.4.1 I2C1/3 框图 634
28.4.2 I2C2 框图 635
28.4.3 I2C 时钟要求 635
28.4.4 模式选择 636
通信流程 636
28.4.5 I2C 初始化 637
使能和禁止外设 637
噪声滤波器 637
I2C 时序 638
28.4.6 软件复位 641
28.4.7 数据传输 642
接收 642
发送 643
硬件传输管理 643
28.4.8 I2C 从模式 644
I2C 从模式初始化 644
带时钟延长的从模式 (NOSTRETCH = 0) 645
不带时钟延长的从模式 (NOSTRETCH = 1) 645
从器件字节控制模式 645
从发送器 646
从接收器 650

28.4.9 I2C 主模式 652
I2C 主模式初始化 652
主模式通信初始化（地址阶段） 654
主接收器寻址 10 位地址从器件的初始化过程 655
主发送器 656
主接收器 660
28.4.10 I2C_TIMINGR 寄存器配置示例 664
28.4.11 SMBus 特性 665
简介 665
SMBUS 以 I2C 规范第 2.1 版为基础。 666
总线协议 666
地址解析协议 (ARP) 666
接收的命令和数据应答控制 666
主机通知协议 666
SMBus 报警 666
数据包错误校验 667
超时 667
总线空闲检测 668
28.4.12 SMBus 初始化 668
接收的命令和数据应答控制（从模式） 668
特定地址（从模式） 668
数据包错误校验 668
超时检测 669
总线空闲检测 669
28.4.13 SMBus：I2C_TIMEOUTR 寄存器配置示例 670
28.4.14 SMBus 从模式 671
SMBus 从发送器 671
SMBus 从接收器 672
SMBus 主发送器 674
SMBus 主接收器 676
28.4.15 地址匹配时从停止模式唤醒 677
28.4.16 错误条件 677
总线错误 (BERR) 677
仲裁丢失 (ARLO) 678
上溢/下溢错误 (OVR) 678
数据包错误校验错误 (PECERR) 678
超时错误 (TIMEOUT) 678
报警 (ALERT) 679
28.4.17 DMA 请求 679
使用 DMA 进行发送 679
使用 DMA 进行接收 679

28.4.18 调试模式 680
28.5 I2C 低功耗模式 680
28.6 I2C 中断 680
28.7 I2C 寄存器 682
28.7.1 控制寄存器 1 (I2C_CR1) 682
28.7.2 控制寄存器 2 (I2C_CR2) 685
28.7.3 设备自身地址 1 寄存器 (I2C_OAR1) 688
28.7.4 设备自身地址 2 寄存器 (I2C_OAR2) 689
28.7.5 时序寄存器 (I2C_TIMINGR) 690
28.7.6 超时寄存器 (I2C_TIMEOUTR) 691
28.7.7 中断和状态寄存器 (I2C_ISR) 692
28.7.8 中断清零寄存器 (I2C_ICR) 694
28.7.9 PEC 寄存器 (I2C_PECR) 695
28.7.10 接收数据寄存器 (I2C_RXDR) 696
28.7.11 发送数据寄存器 (I2C_TXDR) 696
28.7.12 I2C寄存器映射 697

29 通用同步异步收发器 (USART) 699
29.1 简介 699
29.2 USART 主要特性 699
29.3 USART 扩展特性 700
29.4 USART 实现 700
29.5 USART 功能说明 701
29.5.1 USART 字符说明 703
29.5.2 USART 发送器 705
字符发送 705
单字节通信 706
中断字符 707
空闲字符 707
29.5.3 USART 接收器 708
起始位检测 708
字符接收 709
中断字符 709
空闲字符 709
上溢错误 710
选择合适的过采样方法 710
帧错误 (Framing error) 712
接收期间可配置的停止位 712

29.5.4 USART 波特率生成 713
如何从 USARTx_BRR 寄存器中获取 USARTDIV 713
29.5.5 USART 接收器对时钟偏差的容差 714
29.5.6 USART 自动波特率检测 716
29.5.7 使用 USART 进行多处理器通信 717
空闲线路检测 (WAKE=0) 717
4 位/7 位地址标记检测 (WAKE=1) 718
29.5.8 使用 USART 进行 Modbus 通信 718
Modbus/RTU 718
Modbus/ASCII 719
29.5.9 USART 极性控制 719
偶校验. 719
奇校验. 719
接收时进行奇偶校验检查 719
发送时的奇偶校验生成 719
29.5.10 USART LIN（局域互连网络）模式 720
LIN 发送 720
LIN 接收 720
29.5.11 USART 同步模式 722
29.5.12 USART 单线半双工通信 725
29.5.13 USART 智能卡模式 725
块模式 (T=1) 727
正向约定和反向约定 728
29.5.14 USART IrDA SIR ENDEC 模块 729
IrDA 低功耗模式 730
29.5.15 DMA 模式下的 USART 连续通信 731
使用 DMA 进行发送 731
使用 DMA 进行接收 732
多缓冲区通信中的错误标志和中断生成 733
29.5.16 RS232 硬件流控制和 RS485 驱动器使能（使用 USART） 733
RS232 RTS 流控制. 734
RS232 CTS 流控制. 734
RS485 驱动器使能 735
29.5.17 使用 USART 从停止模式唤醒 735
使用静默模式和停止模式 735
当 USART 时钟源为 HSI 时钟时，确定最大 USART 波特率允许从停止模式正确唤醒 736
29.6 USART 低功耗模式 736
29.7 USART 中断 737
29.8 USART 寄存器 739

29.8.1 控制寄存器 1 (USARTx_CR1) 739
29.8.2 控制寄存器 2 (USARTx_CR2) 742
29.8.3 控制寄存器 3 (USARTx_CR3) 745
29.8.4 波特率寄存器 (USARTx_BRR) 749
29.8.5 保护时间和预分频器寄存器 (USARTx_GTPR) 750
29.8.6 接收器超时寄存器 (USARTx_RTOR) 751
29.8.7 请求寄存器 (USARTx_RQR) 752
29.8.8 中断和状态寄存器 (USARTx_ISR) 753
29.8.9 中断标志清零寄存器 (USARTx_ICR) 757
29.8.10 接收数据寄存器 (USARTx_RDR) 758
29.8.11 发送数据寄存器 (USARTx_TDR) 759
29.8.12 USART 寄存器映射 759

30 低功耗通用异步接收器 (LPUART) 761

30.1 简介 761
30.2 LPUART 主要特性 761
30.3 LPUART 特性实现 762
30.4 LPUART 功能说明 762
30.4.1 LPUART 字符说明 764
30.4.2 LPUART 发送器 766
字符发送 766
单字节通信 767
中断字符 768
空闲字符 768
30.4.3 LPUART 接收器 768
起始位检测 768
字符接收 768
中断字符 769
空闲字符 769
上溢错误 769
选择时钟源 770
帧错误 (Framing error) 770
接收期间可配置的停止位 770
30.4.4 LPUART 波特率生成 771
30.4.5 LPUART 接收器对时钟偏差的容差 771
30.4.6 使用 LPUART 进行多处理器通信 772
空闲线路检测 (WAKE=0) 773
4 位/7 位地址标记检测 (WAKE=1) 773

30.4.7 LPUART 极性控制 774
偶校验. 774
奇校验. 775
接收时进行奇偶校验检查 775
发送时的奇偶校验生成 775
30.4.8 使用 LPUART 单线半双工通信 775
30.4.9 使用 LPUART 在 DMA 模式下进行连续通信 775
使用 DMA 进行发送 776
使用 DMA 进行接收 777
多缓冲区通信中的错误标志和中断生成 777
30.4.10 RS232 硬件流控制和 RS485 驱动器使能（使用 LPUART） 778
RS232 RTS 流控制. 778
RS232 CTS 流控制. 778
RS485 驱动器使能 779
30.4.11 使用 LPUART 从停止模式唤醒 779
使用静默模式和停止模式 780
当 LPUART 时钟源为 HSI 时钟时，确定最大 LPUART 波特率可使从停止模式正确地唤醒. 780
30.5 LPUART 低功耗模式 781
30.6 LPUART 中断 781
30.7 LPUART 寄存器 783
30.7.1 控制寄存器 1 (LPUART_CR1) 783
30.7.2 控制寄存器 2 (LPUART_CR2) 786
30.7.3 控制寄存器 3 (LPUART_CR3) 788
30.7.4 波特率寄存器 (LPUART_BRR) 790
30.7.5 请求寄存器 (LPUART_RQR) 791
30.7.6 中断和状态寄存器 (LPUART_ISR) 791
30.7.7 中断标志清零寄存器 (LPUART_ICR) 794
30.7.8 接收数据寄存器 (LPUART_RDR) 795
30.7.9 发送数据寄存器 (LPUART_TDR) 796
30.7.10 LPUART 寄存器映射 797

31 串行外设接口/集成电路内置音频总线 (SPI/I2S) 798
31.1 简介 798
31.1.1 SPI 主要特性 798
31.1.2 SPI 扩展特性 799
31.1.3 I2S 功能 799
31.2 SPI/I2S 实现 799

31.3 SPI 功能说明 800
31.3.1 概述 800
31.3.2 一个主器件和一个从器件之间的通信 801
全双工通信 801
半双工通信 801
单工通信 802
31.3.3 标准多从器件通信 803
31.3.4 多主器件通信 804
31.3.5 从器件选择 (NSS) 引脚管理 804
31.3.6 通信格式 806
时钟相位和极性控制 806
数据帧格式 807
31.3.7 SPI配置 808
31.3.8 使能 SPI 的步骤 808
31.3.9 数据发送和接收过程 809
接收和发送缓冲区 809
发送缓冲区处理 809
接收缓冲区处理 809
序列处理 809
31.3.10 禁止 SPI 的步骤 811
31.3.11 使用 DMA（直接存储器寻址）进行通信 812
31.3.12 SPI 状态标志 814
发送缓冲区为空 (TXE) 814
接收缓冲区非空 (RXNE) 814
忙标志 (BSY) 814
31.3.13 SPI 错误标志 814
上溢标志 (OVR) 814
模式故障 (MODF) 815
CRC 错误 (CRCERR) 815
TI 模式帧格式错误 (FRE) 815
31.4 SPI 特性 815
31.4.1 TI 模式 815
主模式下的 TI 协议 815
31.4.2 CRC 计算 816
CRC 原理 816
CPU 管理的 CRC 传输 817
DMA 管理的 CRC 传输 817
复位 SPIx_TXCRC 和 SPIx_RXCRC 值 817
31.5 SPI 中断 818

31.6 I²S 功能说明 819
31.6.1 I²S 一般说明 819
31.6.2 支持的音频协议 820
I²S Philips 标准 821
MSB 对齐标准 823
LSB 对齐标准 824
PCM 标准 826
31.6.3 时钟发生器 826
31.6.4 I²S 主模式 829
步骤 829
发送序列 829
接收序列 829
31.6.5 I²S 从模式 830
发送序列 830
接收序列 831
31.6.6 I²S 状态标志 831
忙标志 (BSY) 831
发送缓冲区为空 (TXE) 832
接收缓冲区非空 (RXNE) 832
通道方向 (CHSIDE) 832
31.6.7 I²S 错误标志 832
下溢标志 (UDR) 832
上溢标志 (OVR) 832
帧错误标志 (FRE) 833
31.6.8 I²S 中断 833
31.6.9 DMA 特性 833
31.7 SPI 和 I²S 寄存器 834
31.7.1 SPI 控制寄存器 1 (SPI_CR1)（不用于 I²S 模式） 834
31.7.2 SPI 控制寄存器 2 (SPI_CR2) 836
31.7.3 SPI 状态寄存器 (SPI_SR) 837
31.7.4 SPI 数据寄存器 (SPI_DR) 838
31.7.5 SPI CRC 多项式寄存器 (SPI_CRCPR)（不用于 I²S 模式） 839
31.7.6 SPI RX CRC 寄存器 (SPI_RXCRCR)（不用于 I²S 模式） 839
31.7.7 SPI TX CRC 寄存器 (SPI_TXCRCR)（不用于 I²S 模式） 840
31.7.8 SPI_I²S 配置寄存器 (SPI_I2SCFGR) 841
31.7.9 SPI_I²S 预分频器寄存器 (SPI_I2SPR) 842
31.7.10 SPI 寄存器映射 843

32 通用串行总线全速设备接口 (USB) 844

32.1 简介 844
32.2 USB 主要特性 844
32.3 USB 实现 844
32.4 USB 功能说明 845
32.4.1 USB 模块说明 846
32.5 编程注意事项 847
32.5.1 通用 USB 设备编程 847
32.5.2 系统复位和上电复位 847
USB 复位（RESET 中断）. 847
数据包缓冲区的结构和用法 848
端点初始化 849
IN 数据包（数据传送）. 849
OUT 和 SETUP 数据包（数据接收）. 849
控制传输 850
32.5.3 双缓冲端点 851
32.5.4 同步传输 853
32.5.5 挂起/恢复事件 854
32.6 USB 寄存器 855
32.6.1 通用寄存器 855
USB 控制寄存器 (USB_CNTR) 855
USB 中断状态寄存器 (USB_ISTR). 857
USB 帧数寄存器 (USB_FNR). 859
USB 设备地址 (USB_DADDR). 860
缓冲区表地址 (USB_BTABLE). 860
LPM 控制和状态寄存器 (USB_LPMCSR). 861
电池充电检测器 (USB_BCDR). 861
端点相关寄存器 862
USB 端点 n 寄存器 (USB_EPnR)，n=[0..7]. 863
32.6.2 缓冲区描述符表 866
发送缓冲区地址 n (USB_ADDRn_TX) 866
发送字节数 n (USB_COUNTn_TX) 867
接收缓冲区地址 n (USB_ADDRn_RX) 867
接收字节数 n (USB_COUNTn_RX) 867
32.6.3 USB 寄存器映射 869

33 调试支持 (DBG) 871
33.1 概述 871
33.2 ARM® 参考文档 872

33.3 引脚排列和调试端口引脚 872
33.3.1 SWD 端口引脚 872
33.3.2 SW-DP 引脚分配 872
33.3.3 SWD 引脚上的内部上拉和下拉 872
33.4 ID 代码和锁定机制 873
33.4.1 MCU 器件 ID 代码 873
DBG_IDCODE 873
33.5 SWD 端口 874
33.5.1 SWD 协议简介 874
33.5.2 SWD 协议序列 874
33.5.3 SW-DP 状态机（复位、空闲状态、ID 代码） 875
33.5.4 DP 和 AP 读/写访问 875
33.5.5 SW-DP 寄存器 876
33.5.6 SW-AP 寄存器 876
33.6 内核调试 877
33.7 BPU（断点单元） 878
33.7.1 BPU 功能 878
33.8 DWT（数据观察点） 878
33.8.1 DWT 功能 878
33.8.2 DWT 程序计数器采样寄存器 878
33.9 MCU 调试组件 (DBG) 878
33.9.1 对低功耗模式的调试支持 878
33.9.2 对定时器、看门狗和 I^2C 的调试支持 879
33.9.3 调试 MCU 配置寄存器（DBG_CR） 879
33.9.4 调试 MCU APB1 冻结寄存器 (DBG_APB1_FZ) 880
33.9.5 调试 MCU APB2 冻结寄存器 (DBG_APB2_FZ) 882
33.10 DBG 寄存器映射 883

34 设备电子签名 884
34.1 存储器大小寄存器 884
34.1.1 Flash 大小寄存器 884
34.2 唯一设备 ID 寄存器（96 位） 884

附录 A 代码示例. 886
A.1 简介 886
A.2 NVM/RCC 操作代码示例 886

A.2.1 提高 CPU 频率的准备序列代码 886
A.2.2 降低 CPU 频率的准备序列代码 886
A.2.3 从 PLL 切换到 HSI16 的序列代码 887
A.2.4 切换到 PLL 的序列代码 887
A.3 NVM 操作代码示例 888
A.3.1 解锁数据 EEPROM 和 FLASH_PECR 寄存器的代码示例 888
A.3.2 锁定数据 EEPROM 和 FLASH_PECR 寄存器的代码示例 888
A.3.3 解锁 NVM 程序存储器的代码示例 888
A.3.4 解锁选项字节区域的代码示例 889
A.3.5 写入数据 EEPROM 的代码示例 889
A.3.6 擦除数据 EEPROM 的代码示例 889
A.3.7 编程选项字节的代码示例 890
A.3.8 擦除选项字节的代码示例 890
A.3.9 对 Flash 程序存储器中的单个字进行编程的代码示例 891
A.3.10 对 Flash 程序存储器的半页进行编程的代码示例 892
A.3.11 擦除 Flash 程序存储器中一页的代码示例 893
A.3.12 批量擦除代码示例 894
A.4 时钟控制器 895
A.4.1 HSE 启动序列代码示例 895
A.4.2 PLL 配置修改代码示例 896
A.4.3 MCO 选择代码示例 897
A.5 GPIO 897
A.5.1 锁定机制代码示例 897
A.5.2 复用功能选择序列代码示例 897
A.5.3 模拟 GPIO 配置代码示例 897
A.6 DMA 898
A.6.1 DMA 通道配置序列代码示例 898
A.7 中断和事件 898
A.7.1 NVIC 初始化示例 898
A.7.2 扩展中断选择代码示例 898
A.8 ADC 899
A.8.1 校准代码示例 899
A.8.2 ADC 使能序列代码示例 899
A.8.3 ADC 禁止序列代码示例 900
A.8.4 MCO 时钟选择代码示例 900
A.8.5 单次转换序列代码示例 - 软件触发 900

A.8.6 连续转换序列代码示例 - 软件触发 901
A.8.7 单次转换序列代码示例 - 硬件触发 901
A.8.8 连续转换序列代码示例 - 硬件触发 902
A.8.9 DMA 单触发模式序列代码示例 902
A.8.10 DMA 循环模式序列代码示例 903
A.8.11 等待模式序列代码示例 903
A.8.12 自动关闭和无等待模式序列代码示例 903
A.8.13 自动关闭和等待模式序列代码示例 904
A.8.14 模拟看门狗代码示例 904
A.8.15 过采样代码示例 905
A.8.16 温度配置代码示例 905
A.8.17 温度计算代码示例 905
A.9 DAC 906
A.9.1 独立触发（不产生波形）代码示例 906
A.9.2 独立触发（生成单个三角波）代码示例 906
A.9.3 DMA 初始化代码示例 906
A.10 TSC 代码示例 907
A.10.1 TSC 配置代码示例 907
A.10.2 TSC 中断代码示例 908
A.11 定时器 908
A.11.1 TI2 上升沿递增计数器代码示例 908
A.11.2 每 2 个 ETR 上升沿递增计数器代码示例 908
A.11.3 输入捕捉配置代码示例 909
A.11.4 输入捕捉数据管理代码示例 909
A.11.5 PWM 输入配置代码示例 910
A.11.6 采用 DMA 的 PWM 输入配置代码示例 910
A.11.7 输出比较配置代码示例 911
A.11.8 边沿对齐 PWM 配置示例 911
A.11.9 中心对齐 PWM 配置示例 912
A.11.10 清零 OCxREF 的 ETR 配置代码示例 912
A.11.11 编码器接口代码示例 913
A.11.12 复位模式代码示例 913
A.11.13 门控模式代码示例 914
A.11.14 触发模式代码示例 914
A.11.15 外部时钟模式 2 + 触发模式代码示例 915
A.11.16 单脉冲模式代码示例 915
A.11.17 将一个定时器用作另一个定时器的预分频器的代码示例 916

A.11.18 使用一个定时器使能另一个定时器的代码示例 916
A.11.19 主器件和从器件同步代码示例 917
A.11.20 两个定时器通过一个外部触发信号同步的代码示例 919
A.11.21 DMA 连续传送功能代码示例 920
A.12 低功耗定时器 (LPTIM) 921
A.12.1 脉冲计数器配置代码示例 921
A.13 IWDG 代码示例 921
A.13.1 IWDG 配置代码示例 921
A.13.2 IWDG 配置（带窗口）代码示例 921
A.14 WWDG 代码示例 922
A.14.1 WWDG 配置代码示例 922
A.15 RTC 代码示例 922
A.15.1 RTC 日历配置代码示例 922
A.15.2 RTC 闹钟配置代码示例 923
A.15.3 RTC WUT 配置代码示例 923
A.15.4 RTC 读日历代码示例 924
A.15.5 RTC 校准代码示例 924
A.15.6 RTC 入侵和时间戳配置代码示例 924
A.15.7 RTC 入侵和时间戳代码示例 925
A.15.8 RTC 时钟输出代码示例 925
A.16 I2C 代码示例 925
A.16.1 将 I2C 配置为从模式的代码示例 925
A.16.2 I2C 从发送器代码示例 926
A.16.3 I2C 从接收器代码示例 926
A.16.4 将 I2C 配置为主模式以进行接收的代码示例 926
A.16.5 将 I2C 配置为主模式以进行发送的代码示例 927
A.16.6 I2C 主发送器代码示例 927
A.16.7 I2C 主接收器代码示例 927
A.16.8 将 I2C 配置为主模式以通过 DMA 进行发送的代码示例 927
A.16.9 将 I2C 配置为从模式以通过 DMA 进行接收的代码示例 928
A.17 USART 代码示例 928
A.17.1 USART 发送器配置代码示例 928
A.17.2 USART 发送字节代码示例 928
A.17.3 USART 传输完成代码示例 928
A.17.4 USART 接收器配置代码示例 928
A.17.5 USART 接收字节代码示例 929

A.17.6 USART LIN 模式代码示例 929
A.17.7 USART 同步模式代码示例 929
A.17.8 USART 单线半双工代码示例 930
A.17.9 USART 智能卡模式代码示例 930
A.17.10 USART IrDA 模式代码示例 930
A.17.11 USART DMA 代码示例 931
A.17.12 USART 硬件流控制代码示例 931
A.18 LPUART 代码示例 932
A.18.1 LPUART 接收器配置代码示例 932
A.18.2 LPUART 接收字节代码示例 932
A.19 SPI 代码示例 932
A.19.1 SPI 主器件配置代码示例 932
A.19.2 SPI 从器件配置代码示例 932
A.19.3 SPI 全双工通信代码示例 932
A.19.4 SPI 主器件配置（采用 DMA）代码示例 933
A.19.5 SPI 从器件配置（采用 DMA）代码示例 933
A.19.6 SPI 中断代码示例 933
A.20 DBG 代码示例 933
A.20.1 DBG 读取器件 Id 代码示例 933
A.20.2 LPM 下的 DBG 调试代码示例 933

35 版本历史 934

表格索引

表 1. STM32L0x3 存储器容量 51
表 2. 按类别划分的特性 51
表 3. STM32L0x3 外设寄存器边界地址 57
表 4. 自举模式 60
表 5. NVM 的构成（第 3 类器件） 63
表 6. UFB = 0 时的 NVM 构成（192 KB 第 5 类器件） 64
表 7. Flash 存储器和数据 EEPROM 重映射（192 KB 第 5 类器件） 65
表 8. UFB = 0 时的 NVM 构成（128 KB 第 5 类器件） 65
表 9. Flash 存储器和数据 EEPROM 重映射（128 KB 第 5 类器件） 66
表 10. UFB = 0 时的 NVM 构成（64 KB 第 5 类器件） 66
表 11. 自举引脚和 BFB2 位配置 67
表 12. 主时钟电源电压范围和频率之间的关系 69
表 13. 存储器访问延迟和等待周期数 69
表 14. 内部缓冲区管理 71
表 15. 缓冲区和推测性读取的配置 74
表 16. 所有存储器接口配置的 Dhrystone 性能 75
表 17. NVM 写入/擦除时序 87
表 18. NVM 写入/擦除时长 87
表 19. 保护级别与 RDP 选项字节的内容 91
表 20. FLASH_WRPROTx 寄存器的保护位与 Flash 程序存储器中受保护的地址之间的关系 92
表 21. 存储器访问与模式、保护和 Flash 程序存储器扇区 93
表 22. Flash 中断请求 95
表 23. Flash 接口 - 寄存器映射和复位值 111
表 24. 选项字节格式 112
表 25. 选项字节构成 112
表 26. CRC 寄存器映射和复位值 119
表 27. 不同防火墙状态下的段访问 123
表 28. 段粒度和区域范围 123
表 29. 防火墙寄存器映射与复位值 131
表 30. 性能与 VCORE 范围 135
表 31. 低功耗模式汇总 142
表 32. 立即睡眠 146
表 33. 退出时睡眠 146
表 34. 立即睡眠（低功耗睡眠） 148
表 35. 退出时睡眠（低功耗睡眠） 148
表 36. 停止模式 150
表 37. 待机模式 152
表 38. PWR - 寄存器映射和复位值 159
表 39. 系统时钟源频率 168
表 40. RCC 寄存器映射和复位值 207
表 41. 低功耗模式对 CRS 的作用 214
表 42. 中断控制位 214
表 43. CRS 寄存器映射和复位值 220
表 44. 端口位配置表 223
表 45. GPIO 寄存器映射和复位值 236
表 46. SYSCFG 寄存器映射和复位值 245
表 47. 可编程的数据宽度、字节存储次序（位 PINC = MINC = 1 时） 250
表 48. DMA 中断请求 251

表 49. 各通道 DMA 请求汇总 . . . 253
表 50. DMA 寄存器映射和复位值 . . . 262
表 51. 向量列表 . . . 264
表 52. EXTI 线连接 . . . 270
表 53. 扩展中断/事件控制器寄存器映射和复位值 . . . 276
表 54. ADC 内部信号 . . . 278
表 55. ADC 引脚 . . . 278
表 56. 触发与转换开始之间的延迟 . . . 284
表 57. 配置触发极性 . . . 288
表 58. 外部触发器 . . . 289
表 59. tSAR 与分辨率的对应关系 . . . 290
表 60. 模拟看门狗比较 . . . 298
表 61. 模拟看门狗通道选择 . . . 298
表 62. 最大输出结果与 N 和 M 的对应关系。呈灰色显示的数值表示截断的部分 . . . 300
表 63. ADC 中断 . . . 304
表 64. ADC 寄存器映射和复位值 . . . 319
表 65. DAC 引脚 . . . 322
表 66. 外部触发器 . . . 325
表 67. DAC 寄存器映射 和复位值 . . . 343
表 68. COMP 寄存器映射和复位值 . . . 351
表 69. 实现 . . . 353
表 70. 帧速率计算示例 . . . 355
表 71. 闪烁频率 . . . 363
表 72. 将 VLCD 轨连接到 GPIO 引脚 . . . 365
表 73. 重映射功能 . . . 367
表 74. 低功耗模式下的 LCD 行为 . . . 372
表 75. LCD 中断请求 . . . 372
表 76. LCD 寄存器映射和复位值 . . . 380
表 77. 采集序列汇总 . . . 385
表 78. 扩频偏差与 AHB 时钟频率之间的关系 . . . 387
表 79. I/O 状态（取决于其模式和 IODEF 位值） . . . 388
表 80. 低功耗模式对 TSC 的作用 . . . 389
表 81. 中断控制位 . . . 390
表 82. TSC 寄存器映射和复位值 . . . 398
表 83. 处理时间（时钟周期数） . . . 414
表 84. AES 中断请求 . . . 414
表 85. AES 寄存器映射 . . . 423
表 86. RNG 寄存器映射和复位映射 . . . 428
表 87. 计数方向与编码器信号的关系 . . . 459
表 88. TIM2/TIM3 内部触发连接 . . . 474
表 89. 标准 OCx 通道的输出控制位 . . . 485
表 90. TIM2/3 寄存器映射和复位值 . . . 492
表 91. 计数方向与编码器信号的关系 . . . 524
表 92. TIMx 内部触发连接 . . . 533
表 93. 标准 OCx 通道的输出控制位 . . . 541
表 94. TIM21/22 寄存器映射和复位值 . . . 545
表 95. TIM6/7 寄存器映射和复位值 . . . 559
表 96. STM32L0x3 LPTIM 特性 . . . 560
表 97. 预分频器的分频比 . . . 563
表 98. 编码器计数方案 . . . 568
表 99. LPTIM 外部触发连接 . . . 575
表 100. LPTIM 寄存器映射和复位值 . . . 579

表 101. IWDG 寄存器映射和复位值 586
表 102. WWDG 寄存器映射和复位值 592
表 103. RTC 引脚 PC13 配置 595
表 104. RTC_OUT 映射 596
表 105. 低功耗模式对 RTC 的作用 607
表 106. 中断控制位 607
表 107. RTC 寄存器映射和复位值 630
表 108. STM32L0x3 I2C 特性 633
表 109. 模拟滤波器与数字滤波器对比 637
表 110. I2C-SMBUS 规范数据建立和保持时间 640
表 111. I2C 配置表 644
表 112. I2C-SMBUS 规范时钟时序 654
表 113. fI2CCLK = 8 MHz 时的时序设置示例 664
表 114. fI2CCLK = 16 MHz 时的时序设置示例 665
表 115. SMBus 超时规范 667
表 116. 带 PEC 的 SMBUS 配置 668
表 117. 不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 $t_{TIMEOUT}$ = 25 ms） 670
表 118. 不同 I2CCLK 频率下的 TIMEOUTB 设置示例 670
表 119. 不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 t_{IDLE} = 50 µs） 670
表 120. 低功耗模式 680
表 121. I2C 中断请求 680
表 122. I2C 寄存器映射和复位值 697
表 123. STM32L0x3 USART 特性 700
表 124. 通过采样数据进行噪声检测 712
表 125. 采用 16 倍或 8 倍过采样时，在 f_{CK} = 32 MHz 下编程波特率的误差计算 714
表 126. BRR [3:0] = 0000 时的 USART 接收器容差 715
表 127. BRR[3:0] 不等于 0000 时的 USART 接收器容差 715
表 128. 帧格式 719
表 129. 低功耗模式对 USART 的影响 736
表 130. USART 中断请求 737
表 131. USART 寄存器映射和复位值 759
表 132. fck = 32,768 KHz 时编程的波特率的错误计算 771
表 133. BRR[3:0] 不等于 0000 时的 LPUART 接收器容差 772
表 134. 帧格式 774
表 135. 低功耗模式对 LPUART 的影响 781
表 136. LPUART 中断请求 781
表 137. LPUART 寄存器映射和复位值 797
表 138. STM32L0x3 SPI 实现 799
表 139. SPI 中断请求 818
表 140. 使用标准 8 MHz HSE 时的音频频率精度 828
表 141. I^2S 中断请求 833
表 142. SPI 寄存器映射和复位值 843
表 143. STM32L0x3USB 实现 844
表 144. 双缓冲缓冲区标志定义 851
表 145. 批量双缓冲存储器缓冲区用法 852
表 146. 同步存储器缓冲区的用法 853
表 147. 恢复事件检测 854
表 148. 接收状态编码 865
表 149. 端点类型编码 865
表 150. 端点种类的意义 866
表 151. 发送状态编码 866
表 152. 已分配缓冲区存储器的定义 868

表 153. USB 寄存器映射和复位值 869
表 154. SW 调试端口引脚 872
表 155. REV-ID 值 873
表 156. 数据包请求（8 位） 874
表 157. ACK 响应（3 位） 874
表 158. DATA 传输（33 位） 875
表 159. SW-DP 寄存器 876
表 160. 32 位调试端口寄存器，通过移位值 A[3:2] 进行寻址 877
表 161. 内核调试寄存器 877
表 162. DBG 寄存器映射和复位值 883
表 163. 文档版本历史 934
表 164. 中文文档版本历史 945

图片索引

图 1. 系统架构 54
图 2. 存储器映射 56
图 3. 一个内部缓冲区的结构 70
图 4. 在预取禁止时获取和执行指令的时序 73
图 5. 在预取使能时获取和执行指令的时序 74
图 6. RDP 级别 91
图 7. CRC 计算单元框图 114
图 8. STM32L0x3 防火墙连接示意图. 121
图 9. 防火墙功能状态. 125
图 10. 电源概述 133
图 11. 性能与 VDD 和 VCORE 范围 136
图 12. 电源监控器 138
图 13. 上电复位/掉电复位波形 139
图 14. BOR 阈值 140
图 15. PVD 阈值 141
图 16. 复位电路简化框图. 161
图 17. 时钟树 163
图 18. HSE/LSE 时钟源. 164
图 19. 使用 TIM21 通道 1 输入捕获测量频率 170
图 20. CRS 框图 211
图 21. CRS 计数器行为 212
图 22. I/O 端口位的基本结构 222
图 23. 5 V 容限 I/O 端口位的基本结构. 222
图 24. 输入浮空/上拉/下拉配置 226
图 25. 输出配置 227
图 26. 复用功能配置 228
图 27. 高阻态模拟配置 228
图 28. DMA 框图 247
图 29. DMA 请求映射 252
图 30. 扩展中断和事件控制器 (EXTI) 框图 267
图 31. 扩展中断/事件 GPIO 映射 269
图 32. ADC 框图 279
图 33. ADC 校准 281
图 34. 校准系数强制 281
图 35. 使能/禁止 ADC 282
图 36. ADC 时钟图 283
图 37. 模数转换时间 287
图 38. ADC 转换时序. 287
图 39. 停止正在进行的转换 288
图 40. 单次序列转换，软件触发 291
图 41. 连续序列转换，软件触发 291
图 42. 单次序列转换，硬件触发 292
图 43. 连续序列转换，硬件触发 292
图 44. 数据对齐方式和分辨率（过采样已禁止：OVSE = 0） 293
图 45. 溢出示例 (OVR) 294
图 46. 等待模式转换（连续模式，软件触发） 296
图 47. WAIT=0、AUTOFF=1 时的行为 297
图 48. WAIT=1、AUTOFF=1 时的行为 297

图 49. 模拟看门狗的保护区域 298
图 50. 20 位到 16 位结果的截断过程 299
图 51. 移 5 位并进行舍入的数值示例 299
图 52. 已触发的过采样模式（TOVS 位 = 1） 301
图 53. 温度传感器和 VREFINT 通道方框图 302
图 54. DAC 框图 322
图 55. DAC 单通道模式下的数据寄存器 323
图 56. 关闭触发 (TEN = 0) 时的转换时序图 TEN = 0 324
图 57. DAC 双通道模式下的数据寄存器 326
图 58. DAC LFSR 寄存器计算算法 330
图 59. LFSR 产生波形的 DAC 转换（使能软件触发） 331
图 60. 生成 DAC 三角波 331
图 61. 生成三角波波形的 DAC 转换（使能软件触发） 332
图 62. 比较器 1 和 2 框图 346
图 63. LCD 控制器框图 354
图 64. 1/3 偏置，1/4 占空比 356
图 65. 静态占空比用例 1 357
图 66. 静态占空比用例 2 358
图 67. 1/2 占空比，1/2 偏置 359
图 68. 1/3 占空比，1/3 偏置 360
图 69. 1/4 占空比，1/3 偏置 361
图 70. 1/8 占空比，1/4 偏置 362
图 71. LCD 电压控制 364
图 72. 死区 365
图 73. SEG/COM 多路复用功能示例 370
图 74. 流程图示例 371
图 75. TSC 框图 383
图 76. 表面电荷转移模拟 I/O 组结构 384
图 77. 采样电容电压变化 385
图 78. 电荷转移采集序列 386
图 79. 扩频变化原理 387
图 80. 框图 401
图 81. ECB 加密模式 403
图 82. ECB 解密模式 403
图 83. CBC 模式加密 404
图 84. CBC 模式解密 405
图 85. 挂起模式管理示例 406
图 86. CTR 模式加密 407
图 87. CTR 模式解密 407
图 88. 32 位计数器 + 随机值构成 408
图 89. 根据数据类型构建 128 位块 409
图 90. 根据数据类型构建 128 位块（续） 410
图 91. 模式 1：加密 410
图 92. 模式 2：密钥生成 411
图 93. 模式 3：解密 412
图 94. 模式 4：密钥生成和解密 412
图 95. 输入阶段的 DMA 请求和数据传输 (AES_IN) 413
图 96. 输出阶段的 DMA 请求 (AES_OUT) 413
图 97. 框图 424
图 98. 通用定时器框图 430
图 99. 预分频器分频由 1 变为 2 时的计数器时序图 432
图 100. 预分频器分频由 1 变为 4 时的计数器时序图 432

图 101. 计数器时序图，1 分频内部时钟 433
图 102. 计数器时序图，2 分频内部时钟 434
图 103. 计数器时序图，4 分频内部时钟 434
图 104. 计数器时序图，N 分频内部时钟 435
图 105. 计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载） 435
图 106. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 已预装载） 436
图 107. 计数器时序图，1 分频内部时钟 437
图 108. 计数器时序图，2 分频内部时钟 437
图 109. 计数器时序图，4 分频内部时钟 438
图 110. 计数器时序图，N 分频内部时钟 438
图 111. 计数器时序图，使用重复计数器时更新事件 439
图 112. 计数器时序图，1 分频内部时钟，TIMx_ARR=0x6 440
图 113. 计数器时序图，2 分频内部时钟 441
图 114. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36 441
图 115. 计数器时序图，N 分频内部时钟 442
图 116. 计数器时序图，ARPE=1 时的更新事件（计数器下溢） 442
图 117. 计数器时序图，ARPE=1 时的更新事件（计数器上溢） 443
图 118. 正常模式下的控制电路，1 分频内部时钟 444
图 119. TI2 外部时钟连接示例 444
图 120. 外部时钟模式 1 下的控制电路 445
图 121. 外部触发输入模块 446
图 122. 外部时钟模式 2 下的控制电路 447
图 123. 捕获/比较通道（例如：通道 1 输入阶段） 448
图 124. 捕获/比较通道 1 主电路 448
图 125. 捕获/比较通道的输出阶段（通道 1） 449
图 126. PWM 输入模式时序 451
图 127. 输出比较模式，翻转 OC1。 452
图 128. 边沿对齐模式的 PWM 波形 (ARR=8) 454
图 129. 中心对齐模式 PWM 波形 (ARR=8) 455
图 130. 单脉冲模式示例。 456
图 131. 清除 TIMx 的 OCxREF 458
图 132. 编码器接口模式下的计数器工作示例 459
图 133. TI1FP1 极性反相时的编码器接口模式示例 460
图 134. 复位模式下的控制电路 461
图 135. 门控模式下的控制电路 462
图 136. 触发模式下的控制电路 462
图 137. 外部时钟模式 2 + 触发模式下的控制电路 463
图 138. 主/从定时器示例 464
图 139. 使用定时器 x 的 OC1REF 信号对定时器 y 实施门控控制 465
图 140. 使用定时器 x 的使能信号对定时器 y 实施门控控制 466
图 141. 使用定时器 x 的更新信号对定时器 y 实施触发控制 467
图 142. 使用定时器 x 的使能信号对定时器 y 实施触发控制 468
图 143. 使用定时器 x 的 TI1 输入触发定时器 x 和定时器 y 469
图 144. 通用定时器框图 (TIM21/22) 495
图 145. 预分频器分频由 1 变为 2 时的计数器时序图 497
图 146. 预分频器分频由 1 变为 4 时的计数器时序图 498
图 147. 计数器时序图，1 分频内部时钟 499
图 148. 计数器时序图，2 分频内部时钟 499
图 149. 计数器时序图，4 分频内部时钟 500
图 150. 计数器时序图，N 分频内部时钟 500
图 151. 计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载） 501

图 152. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载） 501
图 153. 计数器时序图，1 分频内部时钟 502
图 154. 计数器时序图，2 分频内部时钟 503
图 155. 计数器时序图，4 分频内部时钟 503
图 156. 计数器时序图，N 分频内部时钟 504
图 157. 计数器时序图，1 分频内部时钟，TIMx_ARR=0x6 505
图 158. 计数器时序图，2 分频内部时钟 506
图 159. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36 506
图 160. 计数器时序图，N 分频内部时钟 507
图 161. 计数器时序图，ARPE=1 时的更新事件（计数器下溢） 507
图 162. 计数器时序图，ARPE=1 时的更新事件（计数器上溢） 508
图 163. 正常模式下的控制电路，1 分频内部时钟 509
图 164. TI2 外部时钟连接示例 509
图 165. 外部时钟模式 1 下的控制电路 510
图 166. 外部触发输入模块 511
图 167. 外部时钟模式 2 下的控制电路 512
图 168. 捕获/比较通道（例如：通道 1 输入阶段） 513
图 169. 捕获/比较通道 1 主电路 513
图 170. 捕获/比较通道的输出阶段（通道 1 和通道 2） 514
图 171. PWM 输入模式时序 516
图 172. 输出比较模式，翻转 OC1 517
图 173. 边沿对齐模式的 PWM 波形 (ARR=8) 519
图 174. 中心对齐模式 PWM 波形 (ARR=8) 520
图 175. 清除 TIMx 的 OCxREF 521
图 176. 单脉冲模式示例 522
图 177. 编码器接口模式下的计数器工作示例 524
图 178. TI1FP1 极性反相时的编码器接口模式示例 525
图 179. 复位模式下的控制电路 526
图 180. 门控模式下的控制电路 527
图 181. 触发模式下的控制电路 528
图 182. 基本定时器框图 547
图 183. 预分频器分频由 1 变为 2 时的计数器时序图 549
图 184. 预分频器分频由 1 变为 4 时的计数器时序图 549
图 185. 计数器时序图，1 分频内部时钟 550
图 186. 计数器时序图，2 分频内部时钟 551
图 187. 计数器时序图，4 分频内部时钟 551
图 188. 计数器时序图，N 分频内部时钟 552
图 189. 计数器时序图，ARPE = 0 时更新事件（TIMx_ARR 未预装载） 552
图 190. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载） 553
图 191. 正常模式下的控制电路，1 分频内部时钟 554
图 192. 低功耗定时器框图 561
图 193. 干扰滤波器时序图 562
图 194. LPTIM 输出波形，单次计数模式配置 564
图 195. LPTIM 输出波形，单次计数模式配置且激活置 1 一次模式（WAVE 位置 1） 564
图 196. LPTIM 输出波形、连续计数模式配置 565
图 197. 生成波形 566
图 198. 编码器模式计数序列 569
图 199. 独立看门狗框图 580
图 200. 看门狗框图 588
图 201. 窗口看门狗时序图 589
图 202. RTC 框图 594
图 203. I2C1/3 框图 634

图 204. I2C2 框图 635
图 205. I2C 总线协议 636
图 206. 建立和保持时序 638
图 207. I2C 初始化流程图 641
图 208. 数据接收 642
图 209. 数据发送 643
图 210. 从器件初始化流程图 646
图 211. I2C 从发送器的传输序列流程图 (NOSTRETCH=0) 647
图 212. I2C 从发送器的传输序列流程图 (NOSTRETCH=1) 648
图 213. I2C 从发送器的传输总线图 649
图 214. 从接收器的传输序列流程图 (NOSTRETCH=0) 650
图 215. 从接收器的传输序列流程图 (NOSTRETCH=1) 651
图 216. I2C 从接收器的传输总线图 651
图 217. 主时钟生成 653
图 218. 主模式初始化流程图 655
图 219. 10 位地址读访问 (HEAD10R=0) 655
图 220. 10 位地址读访问 (HEAD10R=1) 656
图 221. I2C 主发送器的传输序列流程图（N≤255 字节） 657
图 222. I2C 主发送器的传输序列流程图（N>255 字节） 658
图 223. I2C 主发送器的传输总线图 659
图 224. I2C 主接收器的传输序列流程图（N≤255 字节） 661
图 225. I2C 主接收器的传输序列流程图（N>255 字节） 662
图 226. I2C 主接收器的传输总线图 663
图 227. $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 的超时间隔 667
图 228. SMBus 从发送器的传输序列流程图（N 字节 + PEC） 671
图 229. SMBus 从发送器的传输总线图 (SBC=1) 672
图 230. SMBus 从接收器的传输序列流程图（N 字节 + PEC） 673
图 231. SMBus 从接收器的总线传输图 (SBC=1) 674
图 232. SMBus 主发送器的总线传输图 675
图 233. SMBus 主接收器的总线传输图 676
图 234. I2C 中断映射图 681
图 235. USART 框图 702
图 236. 字长编程 704
图 237. 可配置的停止位 705
图 238. 发送时的 TC/TXE 行为 707
图 239. 16 倍或 8 倍过采样时的起始位检测 708
图 240. 16 倍过采样时的数据采样 711
图 241. 8 倍过采样时的数据采样 711
图 242. 使用空闲线路检测时的静默模式 717
图 243. 使用地址标记检测时的静默模式 718
图 244. LIN 模式下的断路检测（11 位断路长度——LBDL 位置 1） 721
图 245. LIN 模式下的断路检测与帧错误检测 722
图 246. USART 同步发送示例 723
图 247. USART 数据时钟时序图 (M 位 = 00) 723
图 248. USART 数据时钟时序图 (M 位 = 01) 724
图 249. RX 数据建立/保持时间 724
图 250. ISO 7816-3 异步协议 726
图 251. 使用 1.5 个停止位检测奇偶校验错误 727
图 252. IrDA SIR ENDEC——框图 730
图 253. IrDA 数据调制 (3/16)——正常模式 731
图 254. 使用 DMA 进行发送 732
图 255. 使用 DMA 进行接收 733

图 256. 2 个 USART 间的硬件流控制 733
图 257. RS232 RTS 流控制 734
图 258. RS232 CTS 流控制 734
图 259. USART 中断映射图 738
图 260. LPUART 框图 763
图 261. 字长编程 765
图 262. 可配置的停止位 766
图 263. 发送时的 TC/TXE 行为 768
图 264. 使用空闲线路检测时的静默模式 773
图 265. 使用地址标记检测时的静默模式 774
图 266. 使用 DMA 进行发送 776
图 267. 使用 DMA 进行接收 777
图 268. 2 个 LPUART 间的硬件流控制 778
图 269. RS232 RTS 流控制 778
图 270. RS232 CTS 流控制 779
图 271. LPUART 中断映射图 782
图 272. SPI 框图 800
图 273. 全双工单个主器件/单个从器件应用 801
图 274. 半双工单个主器件/单个从器件应用 802
图 275. 单工单个主器件/单个从器件应用（主器件为只发送模式/从器件为只接收模式） 802
图 276. 主器件和三个独立的从器件 803
图 277. 多主器件应用 804
图 278. 硬件/软件从器件选择管理 805
图 279. 数据时钟时序图 807
图 280. 主/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 时序（在连续传输的情况下） 810
图 281. 从器件/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 时序（在连续传输的情况下） 811
图 282. 使用 DMA 进行发送 813
图 283. 使用 DMA 进行接收 813
图 284. TI 模式传输 816
图 285. I^2S 框图 819
图 286. I^2S Philips 协议波形（16/32 位全精度，CPOL = 0） 821
图 287. I^2S Philips 标准波形（24 位帧，CPOL = 0） 821
图 288. 发送 0x8EAA33 821
图 289. 接收 0x8EAA33 822
图 290. I^2S Philips 标准（16 位扩展为 32 位数据包帧，CPOL = 0） 822
图 291. 16 位数据帧扩展到 32 位通道帧的示例 822
图 292. MSB 对齐的 16 位或 32 位全精度长度，CPOL = 0 823
图 293. MSB 对齐的 24 位帧长度，CPOL = 0 823
图 294. 扩展为 32 位数据包帧的 MSB 对齐的 16 位，CPOL = 0 823
图 295. LSB 对齐的 16 位或 32 位全精度，CPOL = 0 824
图 296. LSB 对齐的 24 位帧长度，CPOL = 0 824
图 297. 发送 0x3478AE 所需的操作 824
图 298. 接收 0x3478AE 时所需的操作 825
图 299. 扩展为 32 位数据包帧的 LSB 对齐的 16 位，CPOL = 0 825
图 300. 16 位数据帧扩展到 32 位通道帧的示例 825
图 301. PCM 标准波形（16 位） 826
图 302. PCM 标准波形（16 位扩展到 32 位数据包帧） 826
图 303. 音频采样频率定义 827
图 304. I^2S 时钟发生器架构 827
图 305. USB 外设框图 845

图 306. 数据包缓冲区与缓冲区描述表存储单元示例 848
图 307. STM32L0x3 MCU 和 Cortex®-M0+ 级调试支持框图 871

1 文档约定

1.1 寄存器相关缩写词列表

寄存器说明中使用以下缩写词：

读/写 (rw)	软件可以读写这些位。
只读 (r)	软件只能读取这些位。
只写 (w)	软件只能写入该位。读取该位时将返回复位值。
读取/清零 (rc_w1)	软件可以读取该位，也可以通过写入 1 将该位清零。写入“0”对该位的值无影响。
读取/清零 (rc_w0)	软件可以读取该位，也可以通过写入 0 将该位清零。写入“1”对该位的值无影响。
读取/读取清零 (rc_r)	软件可以读取该位。读取该位时，将自动清零。写入“0”对该位的值无影响。
读取/置位 (rs)	软件可以读取该位，也可将其置 1。写入“0”对该位的值无影响。
保留 (Res.)	保留位，必须保持复位值。

1.2 词汇表

本节简要介绍本文档中所用首字母缩略词和缩写词的定义：

- **扇区**：代码区域中的 32 页写保护数据块
- **页**：32 个字表示代码和系统存储区，1 个字表示数据、出厂选项和用户选项区
- **字**：32 位数据。
- **半字**：16 位数据。
- **字节**：8 位数据。
- **IAP（在应用中编程）**：IAP 是指可以在用户程序运行期间对微控制器的 Flash 进行重新编程。
- **ICP（在线编程）**：ICP 是指可以在器件安装于用户应用电路板上时使用 JTAG 协议、SWD 协议或自举程序对微控制器的 Flash 进行编程。
- **选项字节**：存储于 Flash 中的产品配置位。
- **OBL**：选项字节加载器。
- **AHB**：高级高性能总线。
- **NVM**：非易失性存储器。
- **ECC**：误码校正。
- **DMA**：直接存储器访问。
- **MIF**：NVM 接口。
- **PCROP**：专有代码读保护。

1.3 外设可用性

有关各型号产品的外设可用性及数量信息，请参见特殊器件数据手册。

1.4 产品类别定义

表 1 概括介绍了存储器容量和产品线。

当前参考手册介绍了每个产品线特性的超集，有关每个类别的特性列表，请参见*表 2*。

表 1. STM32L0x3存储器容量

存储器容量	第 3 类器件	第 5 类器件(1)
16 KB	-	-
32 KB	STM32L053x STM32L063x (AES)	-
64 KB	STM32L053x STM32L063x (AES)	STM32L073x STM32L083x (AES)
128 KB	-	STM32L073x STM32L083x (AES)
192 KB	-	STM32L073x STM32L083x (AES)

1. 正在开发的产品。

表 2. 按类别划分的特性

特性	第 3 类器件	第 5 类器件
MPU	全功能	全功能
NVM	全功能， 单存储区域	全功能
循环冗余校验计算单元（CRC）	全功能	全功能
防火墙 (FW)	全功能	全功能
电源控制 (PWR)	全功能	全功能
复位和时钟控制 (RCC)	全功能	全功能
时钟恢复系统 (CRS)	全功能	全功能
GPIOA	全功能	全功能
GPIOB	全功能	全功能
GPIOC	全功能	全功能
GPIOD	[2]	全功能
GPIOE	-	全功能

表 2. 按类别划分的特性（续）

特性	第 3 类器件	第 5 类器件
GPIOH	[0:1]	[0:1][9:10]
系统配置控制器 (SYSCFG)	全功能	全功能
直接存储器访问控制器（DMA1）	全功能	全功能
嵌套向量中断控制器 (NVIC)	全功能	全功能
扩展中断和事件控制器 (EXTI)	全功能	全功能
模数转换器 (ADC1)	全功能	全功能
数模转换器 (DAC1)	全功能	全功能
数模转换器 (DAC2)	-	全功能
比较器 (COMP1)	全功能	全功能
比较器 (COMP2)	全功能	全功能
液晶显示控制器（LCD）	8x28 或 4x32	8x48 或 4x52
触摸感应控制器（TSC1）	全功能	全功能
高级加密标准硬件加速器（AES）	全功能	全功能
随机数发生器 (RNG)	全功能	全功能
通用定时器 (TIM2)	全功能	全功能
通用定时器 (TIM3)	-	全功能
通用定时器 (TIM21)	全功能	全功能
通用定时器 (TIM22)	全功能	全功能
基本定时器 (TIM6)	全功能	全功能
基本定时器 (TIM7)	-	全功能
低功耗定时器 1 (LPTIM1)	全功能	全功能
独立看门狗 (IWDG)	全功能	全功能
系统窗口看门狗 (WWDG)	全功能	全功能
实时时钟 (RTC)	全功能	全功能
内部集成电路 (I2C1) 接口	全功能	全功能
内部集成电路 (I2C2) 接口	全功能	全功能
内部集成电路 (I2C3) 接口	-	全功能
通用同步异步收发器 (USART1)	全功能	全功能
通用同步异步收发器 (USART2)	全功能	全功能
通用同步异步收发器 (USART4)	-	全功能
通用同步/异步收发器 (USART5)	-	全功能
低功耗通用异步收发器 (LPUART1)	全功能	全功能

表 2. 按类别划分的特性（续）

特性	第 3 类器件	第 5 类器件
串行外设接口 (SPI1)	全功能	全功能
串行外设接口/集成电路内置音频总线 (SPI2/I2S2)	全功能	全功能
通用串行总线全功能型高速器件接口 (USB)	全功能	全功能
调试支持 (DBG)	全功能	全功能
设备电子签名	全功能	全功能

2 系统和存储器概述

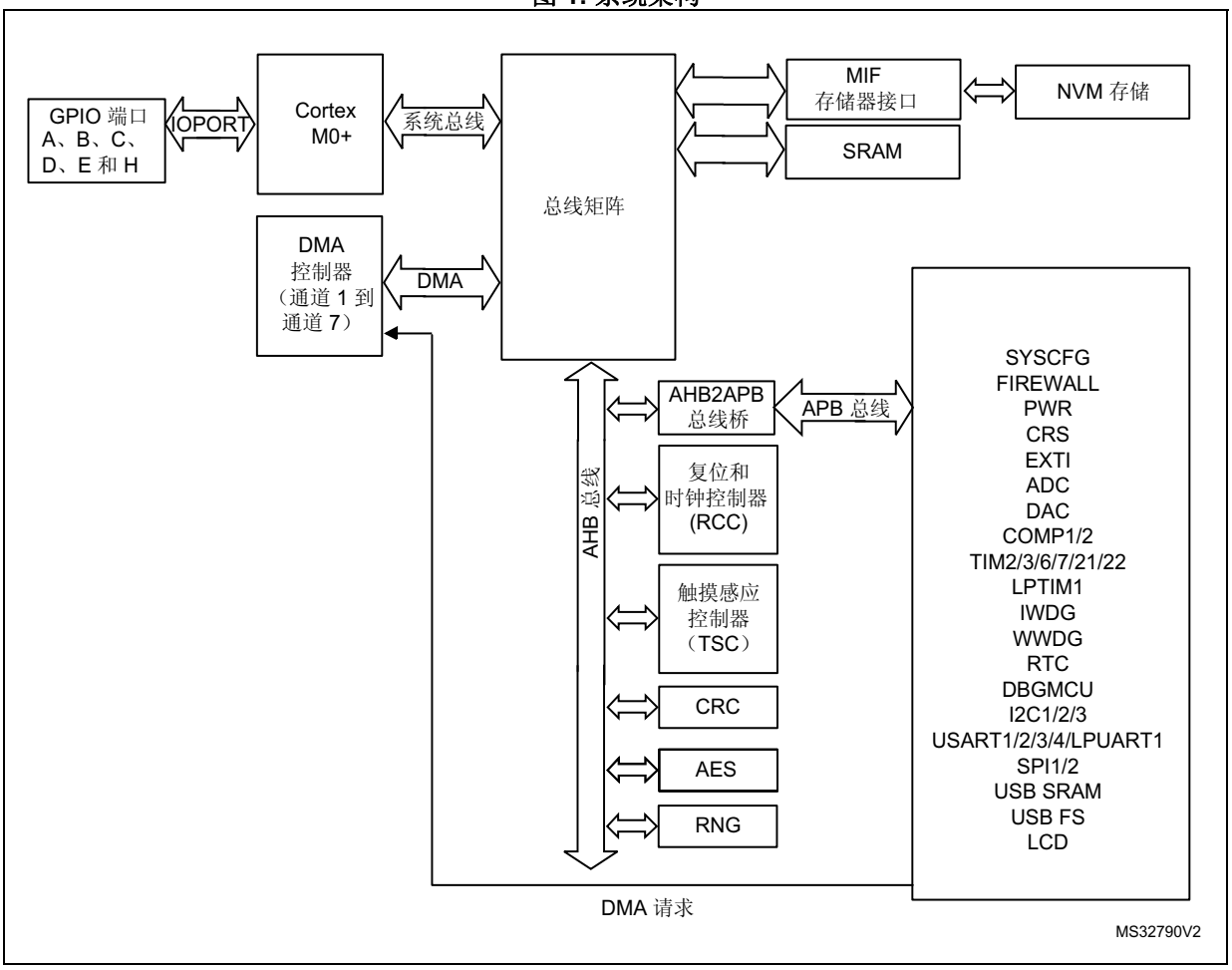
2.1 系统架构

主系统包括：

- 两个主器件：
  - Cortex®-M0+ 内核（AHB-lite 总线）
  - GP-DMA，通用微控制器
- 三个从器件：
  - 内部 SRAM
  - 内部非易失性存储器
  - AHB 到 APB，连接所有 APB 外设

这些外设使用多层 AHB 总线架构互连，如*图 1* 所示。

图 1. 系统架构



1. 有关器件上可用的 GPIO 端口和外设，请参见*表 1：STM32L0x3 存储器容量*、*表 2：按类别划分的特性*和器件数据表。

2.1.1 S0：Cortex® 总线

此总线用于将 Cortex®-M0+ 内核的 DCode/ICode 总线连接到总线矩阵。内核会使用此总线获取指令、获取数据以及访问 AHB/APB 资源。

2.1.2 S1：DMA 总线

此总线用于将 DMA 的 AHB 主接口连接到总线矩阵，而总线矩阵管理着不同主器件对 Flash 和 EEPPROM、SRAM 以及 AHB/APB 外设的访问。

2.1.3 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。总线矩阵由两个主器件（CPU、DMA）和三个从器件（NVM 接口、SRAM、AHB2APB1/2 总线桥）构成。

AHB/APB 总线桥

AHB/APB 总线桥可在 AHB 与 2 条 APB 总线之间实现完全同步的连接。APB1 和 APB2 的最大工作频率高达 32 MHz。

有关连接到此总线桥的外设的地址映射，请参见*第 57 页的第 2.2.2 节：存储器映射和寄存器边界地址*。

每次芯片复位后，所有外设时钟都被关闭（SRAM 和 MIF 除外）。使用外设前，必须在 RCC_AHBENR、RCC_APB2ENR、RCC_APB1ENR 或 RCC_IOPENR 寄存器中使能其时钟。

注：对 APB 寄存器执行 16 位或 8 位访问时，该访问将转换为 32 位访问：总线桥将 16 位或 8 位数据复制后提供给 32 位向量。

2.2 存储器组织结构

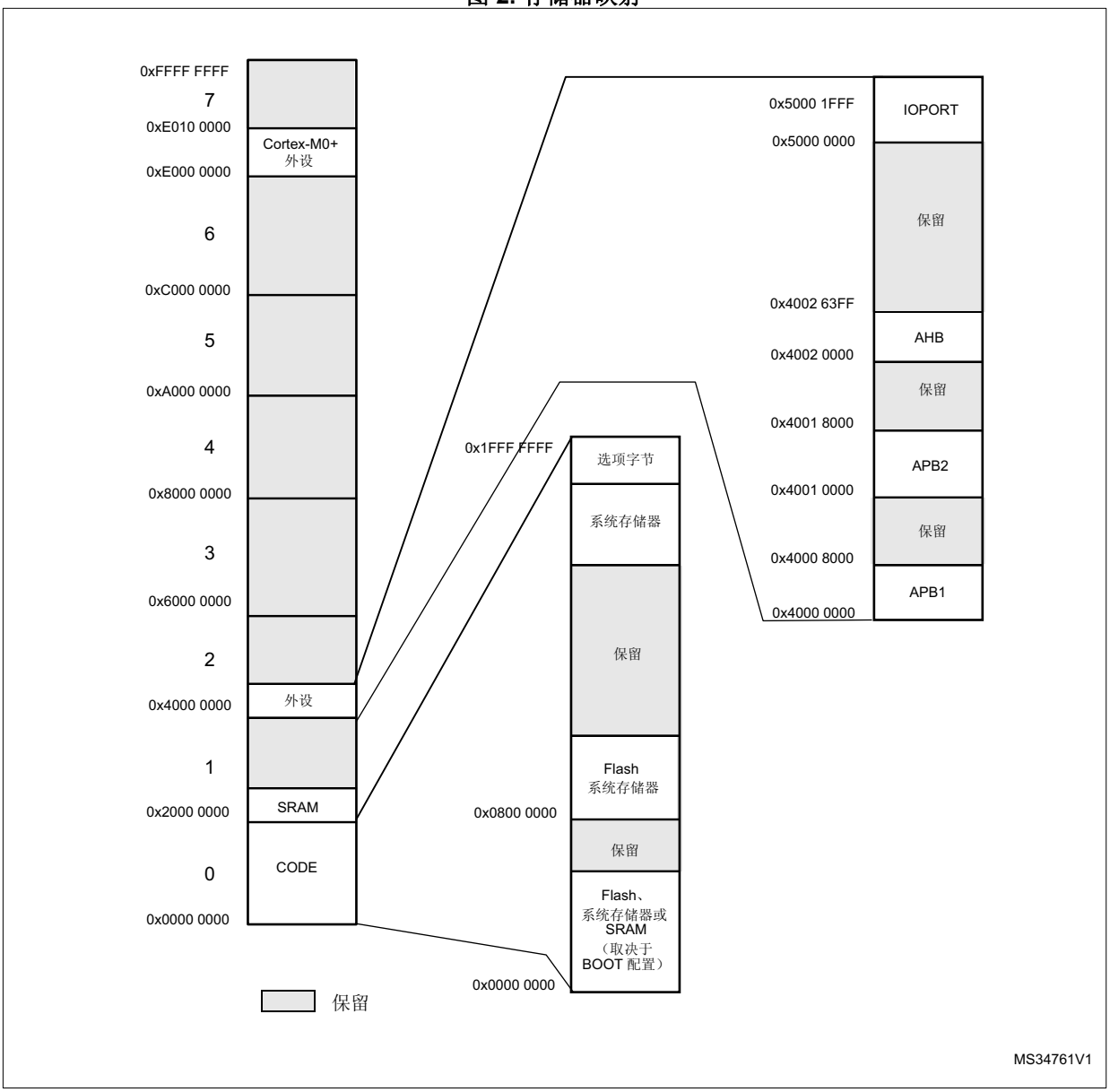
2.2.1 简介

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个线性（即地址连续）的 4 GB 地址空间内。

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

可寻址的存储空间分为 8 个主要块，每个块为 512 MB。

图 2. 存储器映射



未分配给片上存储器和外设的所有存储区域均视为“保留区”。有关可用存储器和寄存器区域的详细映射，请参见*存储器映射和寄存器边界地址*和外设章节。

2.2.2 存储器映射和寄存器边界地址

有关综合存储器映射图，请参见相应器件的数据手册。

下表给出了器件中可用外设的边界地址。

表 3. STM32L0x3 外设寄存器边界地址(1)

总线	边界地址	大小（字节）	外设	外设寄存器映射
IOPORT	0X5000 1C00 - 0X5000 1FFF	1K	GPIOH	*第 9.4.12 节：GPIO 寄存器映射*
	0X5000 1400 - 0X5000 1BFF	2 K	保留	-
	0X5000 1000 - 0X5000 13FF	1K	GPIOE	*第 9.4.12 节：GPIO 寄存器映射*
	0X5000 0C00 - 0X5000 0FFF	1K	GPIOD	*第 9.4.12 节：GPIO 寄存器映射*
	0X5000 0800 - 0X5000 0BFF	1K	GPIO C	*第 9.4.12 节：GPIO 寄存器映射*
	0X5000 0400 - 0X5000 07FF	1K	GPIOB	*第 9.4.12 节：GPIO 寄存器映射*
	0X5000 0000 - 0X5000 03FF	1K	GPIOA	*第 9.4.12 节：GPIO 寄存器映射*
AHB	0X4002 6400 - 0X4002 FFFF	49 K	保留	-
	0X4002 6000 - 0X4002 63FF	1 K	AES（仅限使用 AES 的第 3 类和第 5 类器件）	*第 19.12.13 节：AES 寄存器映射*
	0X4002 5400 - 0X4002 5FFF	3 K	保留	-
	0X4002 5000 - 0X4002 53FF	1 K	RNG：	*第 20.4.4 节：RNG 寄存器映射*
	0X4002 4400 - 0X4002 4FFF	3 K	保留	-
	0X4002 4000 - 0X4002 43FF	1 K	TSC	*第 18.6.11 节：TSC 寄存器映射*
	0X4002 3400 - 0X4002 3FFF	3 K	保留	-
	0X4002 3000 - 0X4002 33FF	1 K	CRC	*第 4.4.6 节：CRC 寄存器映射*
	0X4002 2400 - 0X4002 2FFF	3 K	保留	-
	0X4002 2000 - 0X4002 23FF	1 K	闪存	*第 3.7.11 节：Flash 寄存器映射*
	0X4002 1400 - 0X4002 1FFF	3 K	保留	-
	0X4002 1000 - 0X4002 13FF	1 K	RCC	*第 7.3.22 节：RCC 寄存器映射*
	0X4002 0400 - 0X4002 0FFF	3 K	保留	-
	0X4002 0000 - 0X4002 03FF	1 K	DMA1	*第 11.4.8 节：DMA 寄存器映射*

表 3. STM32L0x3外设寄存器边界地址[1]（续）

总线	边界地址	大小（字节）	外设	外设寄存器映射
APB2	0X4001 5C00 - 0X4001 FFFF	42 K	保留	-
	0X4001 5800 - 0X4001 5BFF	1 K	DBG	*第 33.10 节：DBG 寄存器映射*
	0X4001 3C00 - 0X4001 57FF	7 K	保留	-
	0X4001 3800 - 0X4001 3BFF	1 K	USART1	*第 29.8.12 节：USART 寄存器映射*
	0X4001 3400 - 0X4001 37FF	1 K	保留	-
	0X4001 3000 - 0X4001 33FF	1 K	SPI1	*第 31.7.10 节：SPI 寄存器映射*
	0X4001 2800 - 0X4001 2FFF	2 K	保留	-
	0X4001 2400 - 0X4001 27FF	1 K	ADC1	*第 14.13.12 节：ADC 寄存器映射*
	0X4001 2000 - 0X4001 23FF	1 K	保留	-
	0X4001 1C00 - 0X4001 1FFF	1 K	防火墙	*第 5.4.8 节：防火墙寄存器映射*
	0X4001 1800 - 0X4001 1BFF	1 K	保留	-
	0X4001 1400 - 0X4001 17FF	1 K	TIM22	*第 22.4.16 节：TIM21/22 寄存器映射*
	0X4001 0C000 - 0X4001 13FF	2 K	保留	-
	0X4001 0800 - 0X4001 0BFF	1 K	TIM21	*第 22.4.16 节：TIM21/22 寄存器映射*
	0X4001 0400 - 0X4001 07FF	1 K	EXTI	*第 13.5.7 节：EXTI 寄存器映射*
	0X4001 0000 - 0X4001 03FF	1 K	SYSCFG，COMP	*第 10.2.8 节：SYSCFG 寄存器映射*，*第 16.5.3 节：COMP 寄存器映射*
APB1	0X4000 8000 - 0X4000 FFFF	32 K	保留	-
	0X4000 7C00 - 0X4000 7FFF	1 K	LPTIM1	*第 24.6.9 节：LPTIM 寄存器映射*
	0X4000 7800 - 0X4000 7BFF	1K	I2C3	*第 28.7.12 节：I2C 寄存器映射*
	0X4000 7400 - 0X4000 77FF	1 K	DAC1/2	*第 15.10.15 节：DAC 寄存器映射*
	0X4000 7000 - 0X4000 73FF	1 K	PWR	*第 6.4.3 节：PWR 寄存器映射*
	0X4000 6C00 - 0X4000 6FFF	1 K	CRS	*第 8.6.5 节：CRS 寄存器映射*
	0X4000 6800 - 0X4000 6BFF	1 K	保留	-
	0X4000 6000 - 0X4000 67FF	2 K	USB（SRAM 512x16 位）	-
	0X4000 5C00 - 0X4000 5FFF	1 K	USB FS	*第 32.6.3 节：USB 寄存器映射*

表 3. STM32L0x3外设寄存器边界地址[1]（续）

总线	边界地址	大小（字节）	外设	外设寄存器映射
APB1	0X4000 5800 - 0X4000 5BFF	1 K	I2C2	*第 28.7.12 节：I2C 寄存器映射*
	0X4000 5400 - 0X4000 57FF	1 K	I2C1	*第 28.7.12 节：I2C 寄存器映射*
	0X4000 5000 - 0X4000 53FF	1 K	USART5	*第 29.8.12 节：USART 寄存器映射*
	0X4000 4C00 - 0X4000 4FFF	1 K	USART4	*第 29.8.12 节：USART 寄存器映射*
	0X4000 4800 - 0X4000 4BFF	1 K	LPUART1	*第 30.7.10 节：LPUART 寄存器映射*
	0X4000 4400 - 0X4000 47FF	1 K	USART2	*第 29.8.12 节：USART 寄存器映射*
	0X4000 3C000 - 0X4000 43FF	2 K	保留	-
	0X4000 3800 - 0X4000 3BFF	1 K	SPI2	*第 31.7.10 节：SPI 寄存器映射*
	0X4000 3400 - 0X4000 37FF	1 K	保留	-
	0X4000 3000 - 0X4000 33FF	1 K	IWDG	*第 25.4.6 节：IWDG 寄存器映射*
	0X4000 2C00 - 0X4000 2FFF	1 K	WWDG	*第 26.4.4 节：WWDG 寄存器映射*
	0X4000 2800 - 0X4000 2BFF	1 K	RTC + BKP_REG	*第 27.6.21 节：RTC 寄存器映射*
	0X4000 2400 - 0X4000 27FF	1 K	LCD	*第 17.7.6 节：LCD 寄存器映射*
	0X4000 1800 - 0X4000 23FF	3 K	保留	-
	0X4000 1400 - 0X4000 17FF	1 K	TIMER7	*第 23.4.9 节：TIM6/7 寄存器映射*
	0X4000 1000 - 0X4000 13FF	1 K	TIMER6	*第 23.4.9 节：TIM6/7 寄存器映射*
	0X4000 0800 - 0X4000 0FFF	1 K	保留	-
	0X4000 0400 - 0X4000 07FF	1 K	TIMER3	*第 21.5 节：TIMx 寄存器映射*
	0X4000 0000 - 0X4000 03FF	1 K	TIMER2	*第 21.5 节：TIMx 寄存器映射*
SRAM	0X2000 2000 - 0X3FFF FFFF	约 524 M	保留	-
	0X2000 0000 - 0X2000 4FFF	高达 20 K	SRAM	-

表 3. STM32L0x3外设寄存器边界地址[1]（续）

总线	边界地址	大小（字节）	外设	外设寄存器映射
NVM	0X0800 0000 - 0X0802 FFFF	高达 192 K	Flash 程序存储器	-
	0x0808 0000 - 0x0808 17FF	高达 6 K	数据EEPROM	-
	0x1FF0 0000 - 0x1FF0 1FFF	8 K	系统存储器	-
	0x1FF8 0020 - 0x1FF8 007F	96	出厂选项字节	-
	0x1FF8 0000 - 0x1FF8 001F	32	用户选项字节	-

1. 有关器件上可用的 GPIO 端口和外设，请参见*表 1：STM32L0x3 存储器容量*、*表 2：按类别划分的特性*和器件数据表。对应于不可用 GPIO 端口或外设的存储器区域会保留。

2.3 嵌入式 SRAM

STM32L0x3器件的静态 SRAM 高达 20 KB。

此 RAM 可按字节、半字（16 位）或全字（32 位）访问。此存储器可以最大系统时钟频率进行寻址，无等待状态，因此可由 CPU 和 DMA 访问。

SRAM 起始地址为 0x2000 0000。

若通过启动引脚或 MEM_MODE 选择物理重映射，CPU 可从地址 0x0000 0000 访问 SRAM（参见*第 10.2.1 节：SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1)*）。

2.4 自举配置

在 STM32L0x3 中，可通过 BOOT0 引脚以及用户选项字节中的启动配置位选择三种不同的启动模式，如下表所示。

表 4. 自举模式[1]

自举模式选择		自举模式	自举空间
BOOT1 引脚	BOOT0 引脚		
X	0	Flash 程序存储器	选择 Flash 程序存储器作为启动区域
0	1	系统存储器	选择系统存储器作为启动区域
1	1	嵌入式 SRAM	选择嵌入式 SRAM 作为启动区域

1. BOOT1 的值为 nBOOT1 选项位取反。

复位后，在 SYSCLK 的第四个上升沿锁存启动模式配置。用户可以通过设置 nBOOT1 和 BOOT0 引脚来选择需要的启动模式。

退出待机模式后，还可以对启动模式配置进行重新采样。因此不得在待机模式下修改启动模式配置。该启动延迟结束后，CPU 将从地址 0x0000 0000 获取栈顶值，然后从始于 0x0000 0004 的启动存储器开始执行代码。

根据所选的启动模式，Flash 程序存储器、系统存储器或 SRAM 可通过以下方式进行访问：

- 从 Flash 程序存储器启动：Flash 程序存储器在启动存储空间 (0x0000 0000) 中有别名，但也可从它原来的存储空间 (0x0800 0000) 访问。换句话说，Flash 存储器内容可从地址 0x0000 0000 或 0x0800 0000 开始访问。
- 从系统存储器启动：系统存储器在启动存储空间 (0x0000 0000) 中有别名，但也可从它原来的存储空间 (0x1FF0 0000) 访问。
- 从嵌入式 SRAM 启动：SRAM 在启动存储空间 (0x0000 0000) 中有别名，但也可从它原来的存储空间 (0x2000 0000) 访问。

存储区交换（仅限第 5 类器件）

对于具有两个存储区的器件，存储区交换机制允许 CPU 指向启动存储空间 (0x0000 0000) 中的存储区 1 或存储区 2。Flash 程序和数据 EEPROM 地址会更改（参见*表 8：UFB = 0 时的 NVM 构成（128 KB 第 5 类器件）*、*表 10：UFB = 0 时的 NVM 构成（64 KB 第 5 类器件）*）。

物理重映射

选择了启动引脚和启动位后，应用程序软件可修改代码区域中可访问的存储器。这种修改是通过对 SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1) 中的 MEM_MODE 位进行编程实现的。

内部自举程序

嵌入式启动加载器位于系统存储器中，由 ST 在生产阶段编程。它用于通过以下串行接口重新编程 Flash：

- 对于第 3 类器件：USART1、USART2、SPI1 或 SPI2。
- 对于配有 USB 接口的第 5 类器件：USART1、USART2 或 USB。
- 对于没有 USB 接口的第 5 类器件：USART1、USART2、SPI1、SPI2、I2C1 或 I2C2。

关于启动加载器串行接口对应的 I/O 的详细信息，请参见器件数据表。

更多关于 STM32 启动加载器的详细信息，请参见 AN2606。

3 Flash 程序存储器和数据 EEPROM (FLASH)

3.1 简介

非易失性存储器 (NVM) 由以下几部分构成：

- 多达 192 KB 的 Flash 程序存储器。该区域用于存储应用程序代码。
- 多达 6 KB 的数据 EEPROM
- 信息块：
  - 多达 8 KB 的系统存储器
  - 多达 8x4 字节的用户选项字节
  - 多达 96 字节的出厂选项字节

3.2 NVM 主要特性

NVM 接口具有以下特性：

- 读接口在各个区域按字、半字或字节进行组织
- 在 Flash 中按字或半页进行编程
- 在选项字节区域按字进行编程
- 在数据 EEPROM 中按字、半字或字节进行编程
- 按页执行擦除操作（在 Flash、数据 EEPROM 和选项字节中执行）
- 选项字节加载器
- ECC（错误校正代码）：每个字存储 6 位，从而仅识别并纠正一个错误
- 批量擦除操作
- 读/写保护
- PCROP 保护
- 低功耗模式
- 下列特性仅适用于第 5 类器件：
  - 支持同时读写的双存储区存储器
  - 双存储区自举功能，用于在启动过程中从存储区域 1 或存储区域 2 启动
  - 存储区域交换功能

3.3 NVM 功能说明

3.3.1 NVM 的构成

NVM 为 32 位存储器单元，用于存储代码、数据、自举代码或选项字节。

存储器阵列分为多个页。一个页由 Flash 程序存储器和系统存储器中的 32 个字（即 128 字节）以及数据 EEPROM 和选项字节区域（用户和出厂）中的 1 个单字（即 4 字节）构成。

一个 Flash 扇区由 32 个页（即 4 KB）构成。扇区是写保护的粒度。

表 5. NVM 的构成（第 3 类器件）

NVM	NVM 地址	大小（字节）	名称	说明
Flash 程序存储器(1)	0x0800 0000 - 0x0800 007F	128 字节	第 0 页	扇区 0
	0x0800 0080 - 0x0800 00FF	128 字节	第 1 页	
	-	-	-	
	0x0800 0F80 - 0x0800 0FFF	128 字节	第 31 页	
	. . .	. . .	. . .	. . .
	0x0800 7000 - 0x0800 707F	128 字节	第 224 页	扇区 7
	0x0800 7080 - 0x0800 70FF	128 字节	第 225 页	
	-	-	-	
	0x0800 7F80 - 0x0800 7FFF	128 字节	第 255 页	
	. . .	. . .	. . .	. . .
	0x0800 F000 - 0x0800 F07F	128 字节	第 480 页	扇区 15
	0x0800 F080 - 0x0800 F0FF	128 字节	第 481 页	
	-	-	-	
	0x0800 FF80 - 0x0800 FFFF	128 字节	第 511 页	
数据 EEPROM	0x0808 0000 - 0x0808 07FF	2 KB	-	数据 EEPROM
信息块	0x1FF0 0000 - 0x1FF0 0FFF	4 KB	-	系统存储器
	0x1FF8 0020 - 0x1FF8 007F	96 字节	-	出厂选项
	0x1FF8 0000 - 0x1FF8 001F	32 字节	-	用户选项字节

1. 对于 32 KB 第 3 类器件，Flash 程序存储器分为 256 个页，每页包含 128 字节。

表 6. UFB = 0 时的 NVM 构成（192 KB 第 5 类器件）

NVM	NVM 地址	大小（字节）	名称	说明	
Flash 程序存储器	0x0800 0000 - 0x0800 007F	128 字节	第 0 页	扇区 0	存储区域 1
	0x0800 0080 - 0x0800 00FF	128 字节	第 1 页		
	-	-	-		
	0x0800 0F80 - 0x0800 0FFF	128 字节	第 31 页		
	. . .	. . .	. . .	. . .	
	0x0800 7000 - 0x0800 707F	128 字节	第 224 页	扇区 7	
	0x0800 7080 - 0x0800 70FF	128 字节	第 225 页		
	-	-	-		
	0x0800 7F80 - 0x0800 7FFF	128 字节	第 255 页		
	. . .	. . .	. . .	. . .	
	-	-	-		
	0x0801 7F80 - 0x0801 7FFF	128 字节	第 767 页	扇区 23	
	0x0801 8000 - 0x0801 807F	128 字节	第 768 页	扇区 24	存储区域 2
	. . .	. . .	. . .	. . .	
	0x0802 F000 - 0x0802 F07F	128 字节	第 1504 页	扇区 47	
	0x0802 F080 - 0x0802 F0FF	128 字节	第 1505 页		
	-	-	-		
	0x0802 FF80 - 0x0802 FFFF	128 字节	第 1535 页		
数据 EEPROM	0x0808 0000 - 0x0808 0BFF	6 KB	-	数据 EEPROM 存储区域 1	
	0x0808 0C00 - 0x0808 17FF		-	数据 EEPROM 存储区域 2	
信息块	0x1FF0 0000 - 0x1FF0 1FFF	8 KB	-	系统存储器	
	0x1FF8 0020 - 0x1FF8 007F	96 字节	-	出厂选项	
	0x1FF8 0000 - 0x1FF8 001F	32 字节	-	用户选项字节	

表 7. Flash 存储器和数据 EEPROM 重映射（192 KB 第 5 类器件）

NVM	说明	NVM 地址		重映射地址	
		MEM_MODE = 0，BOOT0= 0 并且 UFB = 0	MEM_MODE = 0，BOOT0= 0 并且 UFB = 1	MEM_MODE = 0，BOOT0= 0 并且 UFB = 0	MEM_MODE = 0，BOOT0= 0 并且 UFB = 1
Flash 程序存储器	存储区域 1	0x0800 0000 - 0x0801 7FFF	0x0801 8000 - 0x0802 FFFF	0x0000 0000 - 0x0001 7FFF	0x0001 8000 - 0x0002 FFFF
	存储区域 2	0x0801 8000 - 0x0802 FFFF	0x08000 0000 - 0x0801 7FFF	0x0001 8000 - 0x0002 FFFF	0x0000 0000 - 0x0001 7FFF
数据 EEPROM	存储区域 1	0x0808 0000 - 0x0808 0BFF	0x0808 0C00 - 0x0808 17FF	0x0008 0000 - 0x0008 0BFF	0x0008 0C00 - 0x0008 17FF
	存储区域 2	0x0808 0C00 - 0x0808 17FF	0x0808 0000 - 0x0008 0BFF	0x0008 0C00 - 0x0008 17FF	0x0008 0000 - 0x0008 0BFF

表 8. UFB = 0 时的 NVM 构成（128 KB 第 5 类器件）

NVM	NVM 地址	大小（字节）	名称	说明	
Flash 程序存储器	0x0800 0000 - 0x0800 007F	128 字节	第 0 页	扇区 0	存储区域 1
	0x0800 0080 - 0x0800 00FF	128 字节	第 1 页		
	-	-	-		
	0x0800 0F80 - 0x0800 0FFF	128 字节	第 31 页		
	. . .	. . .	. . .	. . .	
	0x0800 7000 - 0x0800 707F	128 字节	第 224 页	扇区 7	
	0x0800 7080 - 0x0800 70FF	128 字节	第 225 页		
	-	-	-		
	0x0800 7F80 - 0x0800 7FFF	128 字节	第 255 页		
	. . .	. . .	. . .	. . .	
	0x0800 FF80 - 0x0800 FFFF	128 字节	第 511 页	扇区 15	
	0x0801 0000 - 0x0801 007F	128 字节	第 512 页	扇区 16	存储区域 2
	. . .	. . .	. . .	. . .	
	0x0801 F000 - 0x0801 F07F		第 992 页	扇区 31	
	-	-	-		
	0x0801 FF80 - 0x0801 FFFF	128 字节	第 1023 页		
数据 EEPROM	0x0808 0000 - 0x0808 0BFF	6 KB	-	数据 EEPROM 存储区域 1	
	0x0808 0C00 - 0x0808 17FF		-	数据 EEPROM 存储区域 2	

表 8. UFB = 0 时的 NVM 构成（128 KB 第 5 类器件）（续）

NVM	NVM 地址	大小（字节）	名称	说明
信息块	0x1FF0 0000 - 0x1FF0 1FFF	8 KB	-	系统存储器
	0x1FF8 0020 - 0x1FF8 007F	96 字节	-	出厂选项
	0x1FF8 0000 - 0x1FF8 001F	32 字节		用户选项字节

表 9. Flash 存储器和数据 EEPROM 重映射（128 KB 第 5 类器件）

NVM	说明	NVM 地址		重映射地址	
		MEM_MODE = 0，BOOT0= 0 并且 UFB = 0	MEM_MODE = 0，BOOT0= 0 并且 UFB = 1	MEM_MODE = 0，BOOT0= 0 并且 UFB = 0	MEM_MODE = 0，BOOT0= 0 并且 UFB = 1
Flash 程序存储器	存储区域 1	0x0800 0000 - 0x0800 FFFF	0x0801 0000 - 0x0801 FFFF	0x0000 0000 - 0x0000 FFFF	0x0001 0000 - 0x0001 FFFF
	存储区域 2	0x0801 0000 - 0x0801 FFFF	0x0800 0000 - 0x0800 FFFF	0x0001 0000 - 0x0001 FFFF	0x0000 0000 - 0x0000 FFFF
数据 EEPROM	存储区域 1	0x0808 0000 - 0x0808 0BFF	0x0808 0C00 - 0x0808 17FF	0x0008 0000 - 0x0008 0BFF	0x0008 0C00 - 0x0008 17FF
	存储区域 2	0x0808 0C00 - 0x0808 17FF	0x0808 0000 - 0x0808 0BFF	0x0008 0C00 - 0x0008 17FF	0x0008 0000 - 0x0008 0BFF

表 10. UFB = 0 时的 NVM 构成（64 KB 第 5 类器件）(1)

NVM	NVM 地址	大小（字节）	名称	说明	
Flash 程序存储器	0x0800 0000 - 0x0800 007F	128 字节	第 0 页	扇区 0	存储区域 1
	0x0800 0080 - 0x0800 00FF	128 字节	第 1 页		
	-	-	-		
	0x0800 0F80 - 0x0800 0FFF	128 字节	第 31 页		
	. . .	. . .	. . .	. . .	
	0x0800 F000 - 0x0800 F07F	128 字节	第 480 页	扇区 15	
	-	-	-		
	-	-	-		
	0x0800 FF80 - 0x0800 FFFF	128 字节	第 511 页		
数据 EEPROM	0x0808 0C00 - 0x0808 17FF	3 KB	-	数据 EEPROM 存储区域 2	
信息块	0x1FF0 0000 - 0x1FF0 1FFF	8 KB	-	系统存储器	
	0x1FF8 0020 - 0x1FF8 007F	96 字节	-	出厂选项	
	0x1FF8 0000 - 0x1FF8 001F	32 字节		用户选项字节	

1. 64 KB 第 5 类器件不会进行 Flash 和数据 EEPROM 重映射。

3.3.2 双存储区自举功能

第 5 类器件具有两个 Flash 存储区域：存储区域 1 和存储区域 2。这两个存储区采用了附加的自举机制，允许根据 BFB2 位的状态（FLASH_OPTR 寄存器中的位 23）从存储区域 2 或存储区域 1 进行自举。

- 当 BFB2 位置 1 并且自举引脚配置为从 Flash 自举（BOOT0 = 0 且 BOOT1 = x）时，此器件在地址 0 映射系统存储器。此器件在复位和待机后从系统存储器自举并执行实现双存储区自举机制的嵌入式自举程序代码（时长约为 440 µs）：
  - a) 系统存储器代码首先检查存储区域 2。如果其中包含有效代码（请参见下方注释），则将 SYSCFG_CFGR1 寄存器中的 UFB 位置 1，以在地址 0x0800 0000 映射存储区域 2，跳转到存储区域 2 中的应用程序代码并退出自举程序。
  - b) 如果存储区域 2 中的代码无效，则系统存储器代码会检查存储区域 1 中的代码。如果代码有效（请参见下方注释），则跳转到存储区域 1 中的应用程序（UFB 保持为“0”，以便存储区域 1 继续在地址 0x0800 0000 进行映射）。
  - c) 如果存储区域 1 和存储区域 2 均不包含有效代码（请参见下方注释），则禁止保护级别 2 后会执行正常的自举程序操作。否则，无论存储区域 1 的代码是否有效，系统存储器代码均会跳转到该区域。更多详细信息，请参见*表 11*。
- BFB2 位复位时（默认状态），不执行双存储区自举机制。

注：当位于存储区起始地址处的第一个数据（应为堆栈指针）指向有效地址时（栈顶地址），该代码视为有效。

对于第 5 类器件，可选择 Flash 存储区域 1 和存储区域 2、系统存储器或 SRAM 作为自举区域，如*表 11* 所示。

表 11. 自举引脚和 BFB2 位配置

保护级别	BFB2 位	自举模式选择引脚		自举模式	自举空间
		BOOT1	BOOT0		
0 或 1	0	X	0	用户 Flash	选择用户 Flash 存储区域 1 作为自举区域。
		0	1	系统存储器	在系统存储器中自举来执行自举程序。
		1	1	嵌入式 SRAM	在嵌入式 SRAM 中自举。
	1	X	0	系统存储器	在系统存储器中自举来执行双存储区自举机制。如果存储区域 1 和存储区域 2 均无效，则执行自举程序来更新 Flash。
		0	1	系统存储器	在系统存储器中自举来执行自举程序。
		1	1	嵌入式 SRAM	在嵌入式 SRAM 中自举。
2	0	X	0	用户 Flash	选择用户 Flash 存储区域 1 作为自举区域。
		0	1	用户 Flash	
		1	1	用户 Flash	
	1	X	0	系统存储器	在系统存储器中自举来执行双存储区自举机制。如果存储区域 2 无效，则跳转到存储区域 1。
		0	1	系统存储器	
		1	1	系统存储器	

当进入系统存储器时，可以执行执行自举程序（更新 Flash）或执行双存储区跳转（请参见*表 11*）。

启用保护级别 2 后，始终不会执行自举程序来更新 Flash。

满足下述 a、b 和 c 三个条件时，相当于将自举引脚配置为系统存储器自举模式（BOOT0 = 1 且 BOOT1 = 0）。在这种情况下，禁止保护级别 2 后可执行正常的自举程序操作。

a) BFB2 位置 1。
b) 两存储区都不包含有效代码。
c) 自举引脚按如下方式配置：BOOT0 = 0 且 BOOT1 = x。

当 BFB2 位置 1 且存储区域 1 和/或存储区域 2 包含有效用户应用程序代码时，始终执行双存储区自举模式（自举程序始终跳转到用户代码）。

因此，如果已将 BFB2 位置 1（从存储区域 2 自举），为在禁止保护级别 2 后执行自举程序代码从而更新 Flash，需要执行下列操作：

a) 将 BFB2 位置 0，使 BOOT0 = 1 且 BOOT1 = 0，或者
b) 将地址 0x0801 8000/0x0801 0000（存储区域 2 的基址）和 0x0800 0000（存储区域 1 的基址）编程为 0x0。

3.3.3 读取 NVM

读取协议

为读取 NVM 的内容，请从*表 5* 中选择任意地址。存储器接口的时钟必须处于运行状态。（请参见*第 7.3.13 节：AHB 外设时钟使能寄存器 (RCC_AHBENR)* 中的 MIFEN 位）。

根据时钟频率，可能需要 0 或 1 个等待周期才能读取 NVM。

用户必须设置正确的等待周期数（FLASH_ACR 寄存器中的 LATENCY 位）。未采用任何控制来验证所用的频率或功率相对于等待周期数是否正确。若等待周期数不正确，会导致读取值错误（高频率和 0 个等待周期）或执行代码需要较长的时间（低频率和 1 个等待周期）。

可以按字（4 字节）、半字（2 字节）或字节读取 NVM。

当 NVM 仅具有一个存储区域时，无法在写入/擦除操作过程中进行读取。如果正在执行写入/擦除操作，读取操作将处于等待状态，直至写入/擦除操作完成，请求读取操作的主器件停止，但地址受读保护时除外。在这种情况下，错误将通过硬性故障或存储器接口标志发送到主器件，不会发生停止并且读取操作不会处于等待状态。

具有两个存储区域时（第 5 类器件），对其中一个存储区域执行写入或擦除操作时，可同时对另一个存储区域执行读取操作。

CPU 频率/工作模式/NVM 读取时间之间的关系

此器件（及 NVM）可在多种不同的电源电压范围内工作。对于每个范围，可对一些主时钟频率进行设置。*表 12* 汇总了能够确保 NVM 访问时间正确的电源电压范围和频率之间的关系。

表 12. 主时钟电源电压范围和频率之间的关系

名称	电源电压范围	最高频率（1 个等待周期）	最高频率（无等待周期）
范围 1	1.65 V - 1.95 V	32 MHz	16 MHz
范围 2	1.35 V - 1.65 V	16 MHz	8 MHz
范围 3	1.05 V - 1.35 V	4.2 MHz	4.2 MHz

表 13 给出了读取 NVM 中的一个字的延迟。将完整读取一个字的时间 (Ttotal) 与时钟周期进行对比可以看出，在范围 3 中，无需任何等待周期，同时采用器件允许的最高频率 (4.2 MHz)。Ttotal 是 NVM 返回一个值所需的时间，而非完整读取该值的时间（通过存储器接口由存储器读取到内核）；其余所有时间都丢失。

表 13. 存储器访问延迟和等待周期数

名称	Ttotal	频率	周期	无需等待周期
范围 1	46.1 ns	32 MHz	31.25	1
		16 MHz	62.5	0
范围 2	86.8 ns	16 MHz	62.5	1
		8 MHz	125	0
范围 3	184.6 ns	4 MHz	250	0
		2 MHz	500	0

更改 CPU 频率

复位后，使用的时钟是 MSI (2.1 MHz) 并且在 FLASH_ACR 寄存器中配置为 0 个等待周期。建议采用下列软件序列来根据 CPU 频率调整访问 NVM 时所需的等待周期数。

对 CPU 时钟或等待周期数所做的更改可能需要一段时间才能生效。检查 AHB 预分频系数和时钟源状态值是确保所配置 CPU 时钟频率正确的一种方法。同理，读取 FLASH_ACR 是确保编程的等待周期数有效的一种方法。

提升 CPU 频率（采用同一电压范围）

1. 如有必要，可在 FLASH_ACR 寄存器的 LATENCY 位中编程 1 个等待周期。
2. 通过读取 FLASH_ACR 寄存器，检查新的周期数是否设置成功。当等待周期数发生变化时，存储器接口会更改对 NVM 执行读访问的方式。执行读取操作的过程中无法更改等待周期数，因此存储器接口会等待，直到不对 NVM 执行任何读取操作为止。如果主器件读回 FLASH_ACR 寄存器的内容，会停止此读取操作（同时也会停止请求读取操作的主器件），直到等待周期数确实发生变化。如果用户未读回寄存器，对 NVM 的后续访问可在 0 个等待周期内完成，即使已提高时钟频率也是如此，因此值不正确。
3. 更改复位和时钟控制器 (RCC) 中的 CPU 时钟源和/或 AHB 时钟预分频器。
4. 通过读取复位和时钟控制器 (RCC) 中相应的时钟源状态和/或 AHB 预分频值，检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功。由于有些时钟需要经过一段时间才能提供，因此此项检查至关重要。

有关代码示例，请参见 *A.2.1：提高 CPU 频率的准备序列代码*、*A.2.3：从 PLL 切换到 HSI16 的序列代码*和 *A.2.4：切换到 PLL 的序列代码*。

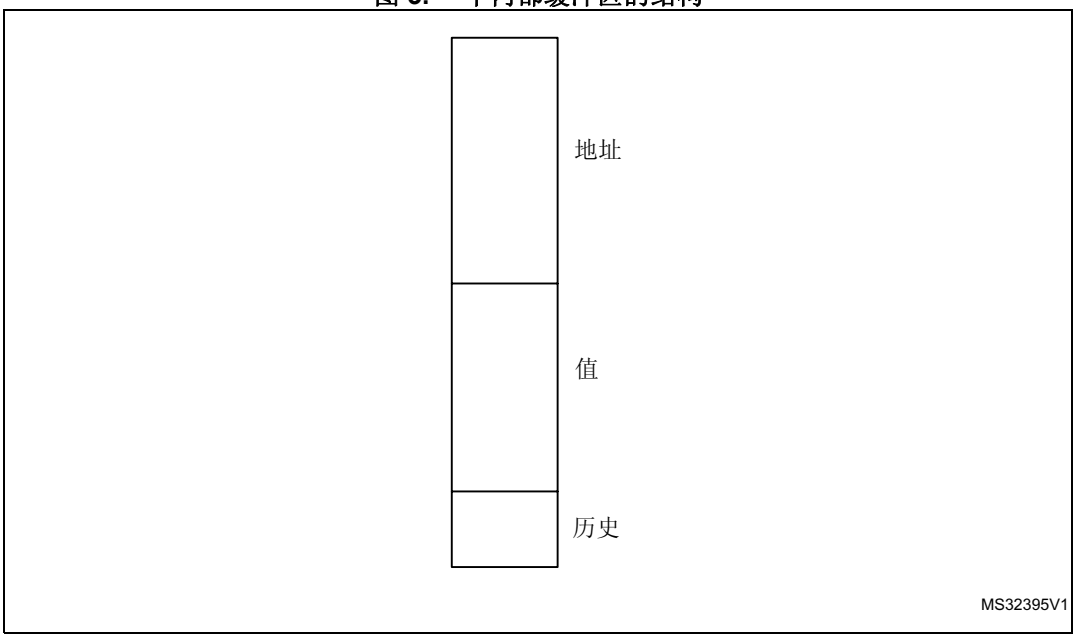
降低 CPU 频率（采用同一电压范围）

1. 更改复位和时钟控制器 (RCC) 中的 CPU 时钟源和/或 AHB 时钟预分频器。
2. 通过读取复位和时钟控制器 (RCC) 中相应的时钟源状态和/或 AHB 预分频值，检查新的 CPU 时钟源和/或新的 CPU 时钟预分频值是否设置成功。
3. 如果需要，可在 FLASH_ACR 寄存器的 LATENCY 位中编程 0 个等待周期。
4. 通过读取 FLASH_ACR 检查新的等待周期数是否设置成功。基于前文介绍的原因，需要读回该寄存器。

数据缓冲

在 NVM 中，六个缓冲区在读取操作中会对性能造成影响（在有些情况下有助于降低功耗），对于取指和数据均是如此。其中一个缓冲区的结构如*图 3* 所示。

图 3. 一个内部缓冲区的结构



每个缓冲区存储 3 类不同的信息：地址、数据和历史。在读取操作过程中，如果找到地址，存储器接口可在不访问 NVM 的情况下返回数据。缓冲区中的数据为 32 位宽（即使主器件仅读取 8 位或 16 位也是如此），因此无论上一次读取操作中所用的大小如何，均可返回值。历史信息用于了解缓冲区的内容是否有效并用于删除原有值（替换为新值）。

这些缓冲区用于存储 NVM 在正常读取操作过程中接收的值并用于推测性读取。禁止推测性读取仅会使主器件请求的数据存储于缓冲区（如果使能，默认状态）。由于值已存在于缓冲区中，无需等待周期，因此可提升性能；存储器中的读取次数会减少且其中的所有组合路径处于稳定状态，因此可降低功耗。

这些缓冲区已进行分组以管理不同的任务。每组缓冲区的数量可由用户所选的配置开始更改（请参见*表 14*）。使用的缓冲区总数始终为 6（如果使能）。历史信息始终由分组进行管理。

存储器接口始终搜索所有缓冲区中是否存在特定地址，而不会检查缓冲区分组以及读取操作的执行对象是取指还是数据。

在复位后或写入/擦除操作更改多个地址后，所有缓冲区将清空并且历史信息将设为 EMPTY。按字、半字或字节编程后，只会清空采用所关注地址的缓冲区。

表 14. 内部缓冲区管理

DISAB_BUF	PREFTEN	PRE_READ	用于取指的缓冲区			用于数据的缓冲区	
			用于跳转的缓冲区	用于预取的的缓冲区	用于最后一个值的缓冲区	用于预读的缓冲区	用于最后一个值的缓冲区
1	-	-	0	0	0	0	0
0	0	0	3	0	1	0	2
0	1	0	2	1	1	0	2
0	0	1	3	0	1	1	1
0	1	1	2	1	1	1	1

如果缓冲区中的值不为空，历史信息会显示从读取或写入该值开始经过的时间。历史信息是包含从最早值到最新值的列表。在给定时刻，分组中仅有一个缓冲区可以具有特定历史值（空值除外）。将一个缓冲区移至最新位置，该组中的其他所有缓冲区向前移动一步，从而保持原有顺序。读取（主器件请求读取缓冲区内容）或写入（从 NVM 中写入新值）缓冲区时，历史信息变为最新位置。当请求存储器中的新地址时，存储器接口始终写入右侧分组中最早的缓冲区，如果存在空缓冲区，将写入空缓冲区。

FLASH_ACR 寄存器的三个配置位用于管理缓冲：

- DISAB_BUF
  该位置 1 会禁止所有缓冲区。该位置 1 时，预取或预读操作无法使能并且例如，如果主器件两次请求同一地址，会对 NVM 执行两次读取操作。
- PRFTEN
  该位置 1（同时 DISAB_BUF 置 0）可使能预取操作。存储器接口未执行任何操作时，会读取获取的最后一个地址后的地址并存储于缓冲区中。
- PRE_READ
  该位置 1（同时 DISAB_BUF 置 0）可使能预读操作。存储器接口未执行任何操作或未执行任何预取操作时，会读取获取的最后一个数据地址后的地址并存储于缓冲区中。

取指和预取

存储器接口取指是对 NVM 执行的一种读取操作，用于执行已读取的操作。存储器接口不检查执行读取操作的主器件或其读取的位置，而仅会验证是否已执行读取操作以执行已经读取的内容。这意味着取指操作：

- 可在所有区域执行。
- 可以任意大小执行（16 位或 32 位）。

存储器接口在缓冲区中存储以下内容：

- 跳转地址，从而在一次循环中，由于已提供跳转地址，仅需要访问一次 NVM。
- 最后读取的地址，从而在对其中 16 位执行取指操作时，另外的 16 位地址已经可用。

为管理取指操作，存储器接口在复位后（DISAB_BUF = 0，PRFTEN = 0 且 PRE_READ = 0）使用 4 个缓冲区。其中 3 个缓冲区用于管理跳转，1 个缓冲区用于存储最后一个获取值。采用这种配置时，4 个用于取指的缓冲区划分为分别带有历史信息的 2 个分组，即用于循环的分组以及用于最后一个获取值的分组。

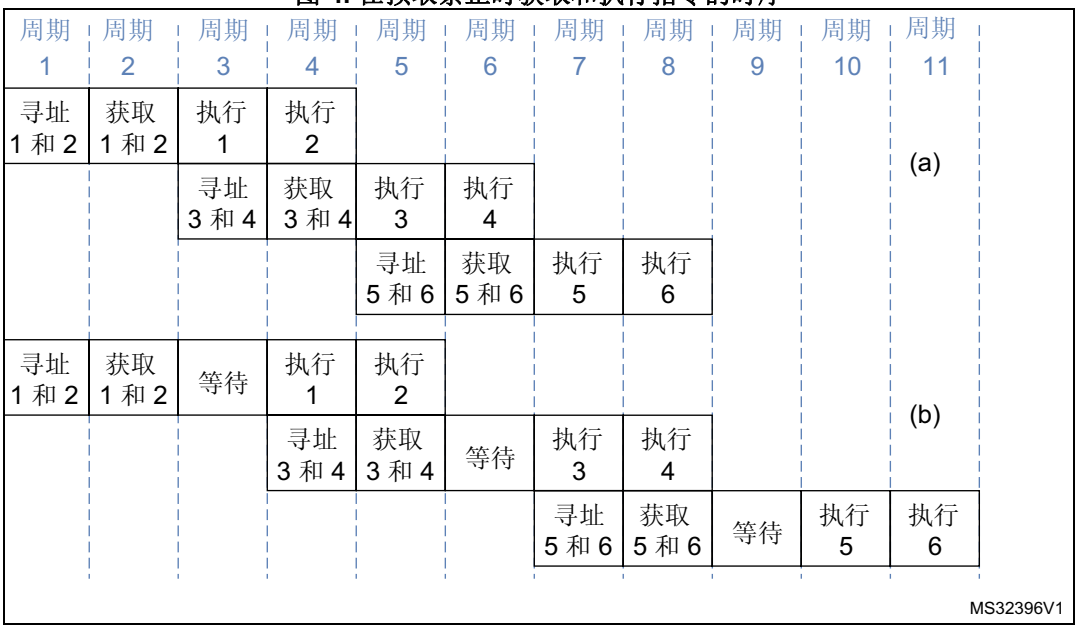
将 PRFTEN 位置 1 以使能预取。预取操作是 NVM 中执行的推测性读取操作，在主器件不请求任何读取操作的情况执行，在此操作中，存储器接口从获取的最后一个地址增加 4（一个字）后的地址读取。该读取操作的优先级较低，如果主器件请求对不同于预取地址的地址执行读取（数据或取指）操作，则此读取操作会中止。预取操作使能后，用于循环的一个缓冲区移到新分组（仅一个缓冲区的分组）以存储预取值：其中 2 个缓冲区继续存储跳转，1 个缓冲区用于预取，而另一个缓冲区用于最后一个值。

存储器接口仅可预取一个地址，因此在未执行取指操作但已完成预取操作时会暂时禁止此功能。执行预取操作后，如果主器件请求预取值，会将预取缓冲区的内容复制到最后一个值缓冲区并使能新的预取操作。相反，如果主器件请求其他地址，预取缓冲区的内容会丢失，并启动 NVM 中的读取操作（如有必要）。该操作完成后，新的预取操作在获取的新地址增加 4 后的地址使能。

预取操作仅在采用 1 个等待周期进行读取时才可提升性能，对于多数线性代码：用户必须对每种情况下使能预取操作的优缺点进行评估。预取操作会增大功耗，原因在于在 NVM 中会执行更多读取操作（并非所有读取操作都由主器件使用）。有关对 Dhrystone 代码执行预取操作的优势，请参见*Dhrystone 性能*部分。

图 4 所示为预取禁止时获取 NVM 中线性代码的时序，其中包括 0 个等待周期 (a) 和 1 个等待周期 (b) 的情况。可将这两个序列与*图 5* 中使能预取操作后的序列进行对比，了解在 0 个和 1 个等待周期时对线性代码执行预取操作的优势。

图 4. 在预取禁止时获取和执行指令的时序



1. (a) 对应于 0 个等待周期。
2. (b) 对应于 1 个等待周期。

图 5 所示为预取使能时，在 0 个等待周期 (a) 和 1 个等待周期 (b) 的情况下获取并执行 NVM 中的指令的时序。预取执行的读取操作以绿色显示。

数据读取和预读

通过存储器接口执行的数据读取操作，对应于除取指操作之外的任何读取操作。主器件将操作常数和操作参数读为数据。DMA 执行的所有读取操作（从一个地址复制到另一地址）均为数据读取操作。不对数据读取的位置（可在 NVM 的每个区域读取）进行任何检查。

复位后（DISAB_BUF = 0，PRFTEN = 0 且 PRE_READ = 0），存储器接口使用同一分组中的 2 个缓冲区存储读为数据的最后两个值。

在一些特殊情况下（例如 DMA 读取 NVM 中大量的连续字），这有助于使能预读（PRE_READ = 1 且 DISAB_BUF = 0）。预读与预取的工作方式完全相同：为在最后一个地址增加 4（一个字）后的地址执行的推测性读取操作。采用这种配置时，会将一个数据缓冲区移到新分组来存储预读值，而第二个缓冲区继续存储最后读取的数据。对于预取操作，预读值在主器件发出请求时在最后读取的值中复制，或在主器件请求其他地址时丢失。

预读操作的优先级低于常规的读取或预取操作：这意味着预读操作仅在未执行其他类型的读取操作时才会启动。请注意，在不适合的情况下执行预读操作可能会产生不利影响：如不会以线性方式执行数据读取的代码中，预读操作会减少用于最后读取的值的缓冲区数（由 2 个减为 1 个），从而增加对 NVM 的访问次数（并会延长读取值所需的时间）。此外，这还会使预取操作产生延迟。Dhrystone 代码是这种情况的一个示例，其结果将在相应章节进行介绍。

对于预取操作，用户必须选择在正确的时刻使能和禁止预读。

图 5. 在预取使能时获取和执行指令的时序

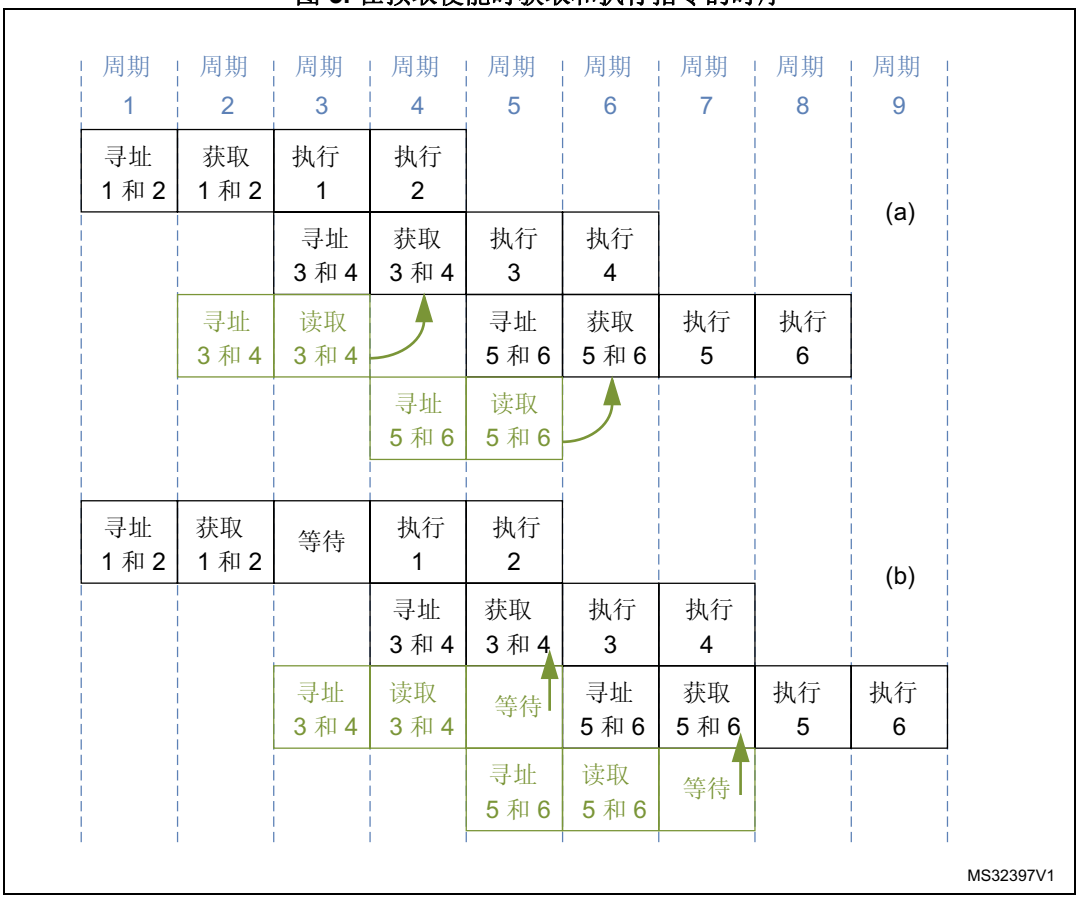


表 15 汇总了可能的配置。

表 15. 缓冲区和推测性读取的配置

DISAB_BUF	PRFTEN	PRE_READ	说明
1	X	X	缓冲区禁止
0	0	0	缓冲区使能：未执行任何推测性读取操作
0	1	0	预取使能：针对取指的推测性读取操作使能
0	0	1	预读使能：针对数据的推测性读取操作使能
0	1	1	预取和预读使能：针对取指和数据的推测性读取操作使能

Dhrystone 性能

Dhrystone 测试用于评估存储器接口的性能。该测试已在所有存储器接口配置下执行。有关结果的汇总，请参见*表 16*。

通用参数包括：

- 矩阵大小为 20 x 20
- 循环执行 1757 次
- ARM® 编译器的版本为 4.1 [固件版本 561]

下列内容为所得结果的相关说明：

表 16. 所有存储器接口配置的 Dhrystone 性能

等待周期数	DISAB_BUF	PRFTEN	PRE_READ	DMIPS 的数量 (x1000)	DMIPS x MHz
0	1	0	0	953	15.25
0	0	0	0	953	15.25
0	0	1	0	953	15.25
0	0	0	1	953	15.25
0	0	1	1	953	15.25
1	1	0	0	677	21.66
1	0	0	0	690	22.08
1	0	1	0	823	26.34
1	0	0	1	691	22.11
1	0	1	1	816	26.11

- 预读操作对于该测试无效：使用预取操作使能会导致存储器接口性能下降，原因在于仅有一个缓冲区用于存储最后读取的数据并且在此代码中，主器件几乎不会以线性方式读取数据。不使用预取使能时，这会证明性能出现极小幅的提升。
- 这些缓冲区（不执行推测性读取操作）采用 1 个等待周期，可带来微弱优势，这种优势在不产生任何成本的情况下可以加以考虑。
- 在 0 个等待周期下，最佳性能（由 ARM® 认证）可能是由于在编译过程中采用不同的代码对齐方式导致的。

3.3.4 写入/擦除 NVM

更改 NVM 内容的方式有多种。存储器接口有助于降低出现意外更改的可能性，并有助于通过硬件实现擦除或写入不同存储器区域所需的所有序列。

写入/擦除协议

为在删除保护后写入/擦除存储器内容，用户需要执行下列操作：

1. 配置待执行的操作；
2. 将正确的数据数量发送到存储器接口，对 NVM 中的一个或多个地址执行写入操作；
3. 等待操作完成。

在等待过程中，用户可以准备下一次操作（极特殊的情况下除外）、写入新配置并开始为下一次写入/擦除操作写入数据。

等待时间取决于操作类型。写入/擦除的持续时间可从 Tprog (3.2 ms) 到 2 x Tglob (3.7 ms) + Tprog (3.2 ms)。存储器接口可配置为仅在一个等待周期的情况下写入一个半页（Flash 程序存储器中的 16 个字）。这可以缩短对大量数据进行编程的时间。

可使用两种不同的协议：单次编程操作和多次编程操作。

单次编程操作

采用该协议时，软件必须将值写入不受保护的 NVM 地址。收到该写入请求后，存储器接口会在一些时钟脉冲内使主器件停止运行（有关详细信息，请参见*表 17*），同时检查保护及之前的值并将新值锁存在 NVM 中。之后，软件可以开始配置下一次操作。该操作将在 FLASH_SR 寄存器的 EOP 位置 1（如果该位在操作启动时置为 0）时完成。*表 19* 汇总了所有操作的操作时间。

多次编程操作（半页）

可以对 Flash 程序存储器中的一个半页（16 个字）执行写入操作。为执行该协议，必须符合下列条件：

- FLASH_SR 寄存器的 PGAERR 位必须为零（之前未出现对齐错误）。
- 第一个地址必须采用半页对齐方式（地址的 6 个低位必须为零）。
- 所有 16 个字必须位于同一半页（16 个字的地址位 7 到位 31 必须相同）。这意味着第一个地址设置该半页，之后的地址必须位于此半页中。写入的数据将按顺序存储于后续地址中。地址增加或更改并不重要（例如，同一地址可使用 16 次），原因在于存储器接口将在内部自动增加地址。
- 仅可写入字（32 位）。

收到第一个地址后，存储器接口会在一些时钟脉冲内使主器件停止运行，同时检查保护及之前的值并将新值锁存在 NVM 中。（更多详细信息，请参见*表 17*）。之后，存储器接口等待第二个地址。不接受任何读取操作：仅执行取指操作，但该操作会使正在执行的写入操作中止。在第二个地址后，存储器接口使内核停止运行一小段时间（短于前一次停止时间），以执行检查并将此地址锁存于 NVM 中，然后等待下一地址。该序列继续执行，直到将所有 16 个字都锁存于 NVM 中。对齐方式或大小不正确将会中止写入操作。如果已将 16 个地址正确锁存，存储器接口会开始执行写入操作。该操作将在 FLASH_SR 寄存器的 EOP 位置 1（如果该位在操作启动时置为 0）时完成。*表 19* 汇总了操作时间。

该协议可通过 RAM 中运行的应用程序代码使用，也可以通过 DMA 使用，DMA 中包含 RAM 中或处于睡眠状态的内核中运行的应用程序代码。

解锁/锁定操作

执行写入/擦除操作前，有必要使能相应操作。用户可以对 Flash 程序存储器、数据 EEPROM 和选项字节区域执行写入操作。

为执行写入/擦除操作，需要解锁 FLASH_PECR 寄存器的 PELOCK 位。该位解锁后（其值为 0），可对同一寄存器的其他位进行修改。当 PELOCK 为 0 时，可在数据 EEPROM 中执行写入/擦除操作。

为对 Flash 程序存储器执行写入/擦除操作，需要解锁 FLASH_PECR 寄存器的 PRGLOCK 位。只有在 PELOCK 为 0 时才能解锁该位。

为对用户选项字节执行写入/擦除操作，需要解锁 FLASH_PECR 寄存器的 OPTLOCK 位。只有在 PELOCK 为 0 时才能解锁该位。PRGLOCK 和 OPTLOCK 之间不存在任何关联：前者可在后者处于锁定状态时解锁，反之亦然。

解锁数据 EEPROM 和 FLASH_PECR 寄存器

复位后，数据 EEPROM 和 FLASH_PECR 寄存器不能以写入模式访问，原因在于 FLASH_PECR 寄存器的 PELOCK 位已置 1。同一解锁序列可同时对二者解除保护。

下列序列用于解锁数据 EEPROM 和 FLASH_PECR 寄存器：

- 将 PEKEY1 = 0x89ABCDEF 写入 FLASH_PEKEYR 寄存器
- 将 PEKEY2 = 0x02030405 写入 FLASH_PEKEYR 寄存器。

有关代码示例，请参见 *A.3.1：解锁数据 EEPROM 和 FLASH_PECR 寄存器的代码示例*。

只要密钥序列错误，就会在下一次复位前锁定 FLASH_PECR 并生成硬性故障。如果主器件尝试在两密钥序列之间写入另一寄存器或使用了错误的密钥，同样会产生上述结果。读访问不会生成错误并且不会使序列中断。在以下四种情况下都会返回硬性故障：

- 在第一次写访问后，输入的 PEKEY1 值不正确。
- 在第二次写访问过程中，输入的 PEKEY1 正确，但 PEKEY2 的值不匹配。
- 尝试对 PEKEYR 写入其他值（请注意：这同样适用于调试器）。
- 尝试写入存储器接口 PEKEY1 和 PEKEY2 之间的其他寄存器。

正确执行操作后，解锁序列会将 FLASH_PECR 寄存器的 PELOCK 位清零。

若要再次锁定 FLASH_PECR 和数据 EEPROM，仅需通过软件将 FLASH_PECR 的 PELOCK 位置 1 即可。再次锁定后，PELOCK 位需要使用新序列返回为 0。

有关代码示例，请参见 *A.3.2：锁定数据 EEPROM 和 FLASH_PECR 寄存器的代码示例*。

解锁 Flash 程序存储器

对 Flash 程序存储器执行写入/擦除操作时采用了附加保护。

复位后，不再能够以写入模式访问 Flash 程序存储器：FLASH_PECR 寄存器的 PRGLOCK 位已置 1。将 PRGLOCK 位清零后，才允许对 Flash 程序存储器执行写访问。

用于解锁 Flash 程序存储器的序列如下：

- 解锁 FLASH_PECR 寄存器（请参见*解锁数据 EEPROM 和 FLASH_PECR 寄存器*部分）。
- 将 PRGKEY1 = 0x8C9DAEBF 写入 FLASH_PRGKEYR 寄存器。
- 将 PRGKEY2 = 0x13141516 写入 FLASH_PRGKEYR 寄存器。

有关代码示例，请参见 *A.3.3：解锁 NVM 程序存储器的代码示例*。

如果在 PELOCK 置 1 的情况下写入这些密钥，则不会产生错误并且 PRGLOCK 将仍为 1。存储器将在重新执行该序列 (PELOCK = 0) 时解锁。

只要密钥序列错误，就会在下一次复位前锁定 FLASH_PECR 的 PRGLOCK 并返回硬性故障。在以下四种情况下都会返回硬性故障：

- 在第一次写访问后，输入的 PRGKEY1 值不正确。
- 在第二次写访问过程中，输入的 PRGKEY1 正确，但 PRGKEY2 的值不匹配。
- 尝试对 PRGKEYR 写入其他值（这同样适用于调试器）。
- 尝试写入存储器接口 PRGKEY1 和 PRGKEY2 之间的其他寄存器。

正确执行操作后，解锁序列会将 PRGLOCK 位清零，因此可对 Flash 程序存储器进行写访问。

若要再次锁定 Flash 程序存储器，仅需通过软件将 FLASH_PECR 中的 PRGLOCK 位置 1 即可。再次锁定后，PRGLOCK 位需要使用新序列返回为 0。如果 PELOCK 返回为 1（锁定），PRGLOCK 也会自动锁定。

解锁选项字节区域

选项字节区域采用了附加写保护。需要解锁 OPTLOCK 才能重新加载或对选项字节区域执行写入/擦除操作。

复位后，选项字节区域不能以写入模式访问：FLASH_PECR 寄存器的 OPTLOCK 位已置 1。将 OPTLOCK 位清零后，才允许对选项字节区域执行写访问。

用于解锁选项字节区域的序列如下：

1. 解锁 FLASH_PECR 寄存器（请参见*解锁数据 EEPROM 和 FLASH_PECR 寄存器*部分）。
2. 将 OPTKEY1 = 0xFBEAD9C8 写入 FLASH_OPTKEYR 寄存器。
3. 将 OPTKEY2 = 0x24252627 写入 FLASH_OPTKEYR 寄存器。

有关代码示例，请参见 *A.3.4：解锁选项字节区域的代码示例*。

如果在 PELOCK = 1 时写入密钥，不会产生错误，OPTLOCK 仍为 1，存储器在重新执行该序列（PELOCK 置为 0）时解锁。

只要密钥序列错误，就会在下一次复位前锁定 FLASH_PECR 的 OPTLOCK 并返回硬性故障。在以下四种情况下都会返回硬性故障：

- 在第一次写访问后，输入的 OPTKEY1 值不正确。
- 在第二次写访问过程中，输入的 OPTKEY1 正确，但 OPTKEY2 的值不匹配。
- 尝试对 OPTKEYR 写入其他值（这同样适用于调试器）。
- 尝试写入存储器接口 OPTKEY1 和 OPTKEY2 之间的其他寄存器。

正确执行操作后，解锁序列会将 OPTLOCK 位清零，因此可对选项字节区域进行写访问。

若要再次锁定选项字节区域，仅需通过软件将 FLASH_PECR 的 OPTLOCK 位置 1 即可。再次锁定后，OPTLOCK 位需要使用新序列返回为 0。如果 PELOCK 返回为 1（锁定），OPTLOCK 也会自动锁定。

在不同操作类型之间进行选择

执行必要的解锁序列后（PELOCK、PRGLOCK 和 OPTLOCK），用户可以使能不同类型的写入和擦除操作，并在 FLASH_PECR 寄存器写入正确配置。涉及的位包括：

- PRG
- DATA
- FIX
- ERASE
- FPRG

NVM 写入/擦除操作的详细说明

本部分详细介绍不同类型的写入和擦除操作，并给出了各项操作所涉及的位。

对数据 EEPROM 执行写入操作

- 目的

  以特定值对数据 EEPROM 中的一个字执行写入操作。
- 大小

  按字节、半字或字写入。
- 地址

  选择数据 EEPROM 中的有效地址。
- 协议

  单次编程操作。
- 请求

  PELOCK = 0，ERASE = 0。
- 错误

  如果 PELOCK = 1 或存储器受到读保护，WRPERR 置 1（并且不执行写入操作）。
- 说明

  该操作用于对数据 EEPROM 的一个字或一个字的一部分执行写入操作。用户必须以正确的大小将正确的值写入正确的地址。存储器接口可在必要时自动执行擦除操作（如果所有位当前均置为 0，则在执行写入操作前无需删除原有内容）。同理，如果写入的数据为 0，则仅执行擦除操作。仅执行写入操作或擦除操作的时长为 Tprog (3.2 ms)；同时执行两操作的时长为 2 x Tprog (6.4 ms)。若将 FIX 标志置 1，则可强制存储器接口每次同时执行擦除和写入操作。
- 时长

  Tprog (3.2 ms) 或 2 x Tprog (6.4 ms)。
- 选项

  将 FIX 位置 1 可强制存储器接口每次同时执行擦除（删除原有内容）和写入操作（写入新数据）操作。这为针对任何数据值和之前数据执行的操作设定了固定时间。

有关代码示例，请参见 *A.3.5：写入数据 EEPROM 的代码示例*。

对数据 EEPROM 执行擦除操作

- **目的**

  删除数据 EEPROM 中的一行。该操作与对数据 EEPROM 写入空字的功能相同，仅出于兼容性目的而提供此功能。

- **大小**

  仅支持按字擦除。

- **地址**

  选择数据 EEPROM 中的有效地址。

- **协议**

  单次编程操作。

- **请求**

  PELOCK = 0，ERASE = 1（可选 DATA = 1）。

- **错误**

  如果 PELOCK = 1 或存储器受到读保护，WRPERR 置 1。

  如果大小不等于一个字，SIZERR 置 1。

- **说明**

  该操作用于删除数据 EEPROM 任一行中的内容。一行仅包含一个字。用户必须以一个字的大小将值写入正确的地址。数据并不重要：仅执行擦除操作（使用非零数据也同样如此）。

- **时长**

  Tprog (3.2 ms)。

有关代码示例，请参见 *A.3.6：擦除数据 EEPROM 的代码示例*。

对选项字节执行写入操作

- **目的**

  将特定值写入选项字节区域的一个字。

- **大小**

  仅支持按字写入。

- **地址**

  选择选项字节区域中的有效地址。

- **协议**

  单次编程操作。

- **请求**

  PELOCK = 0，OPTLOCK = 0 且 ERASE = 0。

- **错误**

  如果 PELOCK = 1 或 OPTLOCK = 1，WRPERR 置 1。
  如果实际读保护级别为 2（在级别 2 无法对选项字节区域执行写入操作），WRPERR 置 1。
  如果大小不等于一个字，SIZERR 置 1。

- 说明

  该操作用于对选项字节区域写入一个字。仅可在级别 0 或级别 1 对选项字节区域执行写入操作。

  用户必须注意，一个字中的高 16 位（位 6 到位 31）必须是低 16 位（位 0 到位 15）的补码：数据高位部分和低位部分不匹配会导致加载选项字节的过程中出错（请参见*第 3.8 节：选项字节*）并强制存储器接口加载默认值。存储器接口在写入时不会检查求得的补码是否正确。用户必须以一个字的大小将所需值写入正确的地址。

  对于数据 EEPROM，存储器接口会在需要时删除之前的内容再执行写入操作。如果待写入的数据为 0，存储器接口不会执无用的写入操作。仅执行写入操作或仅执行擦除操作的时长为 Tprog (3.2 ms)。如果同时执行两操作，时长为 2 x Tprog (6.4 ms)。若将 FIX 标志置 1，则可强制存储器接口每次同时执行擦除和写入操作。

  由于一些配置会更改保护，因此需要密切注意。存储器接口可在批量擦除中更改选项字节写入或强制一些位不得降低保护级别：更多详细信息，请参见*第 3.4.4 节：写入/擦除保护管理*。

- 时长

  Tprog (3.2 ms) 或 2 x Tprog (6.4 ms)。

- 选项

  可将 FIX 位置 1，从而强制存储器接口每次同时执行擦除（删除原有内容）和写入操作（写入新数据）操作。这为针对任何数据值和之前数据执行的操作设定了固定时间。

有关代码示例，请参见 *A.3.7：编程选项字节的代码示例*。

对选项字节执行擦除操作

- 目的

  删除选项字节区域中的一行。该操作执行的功能与将零值写入选项字节相同。仅出于兼容性目的才执行此操作。

- 大小

  仅支持按字擦除。

- 地址

  选择选项字节区域中的有效地址。

- 协议

  单次编程操作。

- 请求

  PELOCK = 0，OPTLOCK = 0 且 ERASE = 1（可选 OPT = 1）。

- 错误

  如果 PELOCK = 1 或 OPTLOCK = 1，WRPERR 置 1。

  如果实际保护级别为 2（在级别 2 无法对选项字节区域执行擦除操作），WRPERR 置 1。

  如果大小不等于一个字，SIZERR 置 1。

- 说明

  该操作用于删除选项字节区域中一行的内容。一行仅包含 1 个字。用户必须以一个字的大小将零写入正确的地址。

  有关更多信息，请参见*对选项字节执行写入操作*一节。

  由于所有位在擦除操作后均置为 0，加载选项字节的过程中会出现不匹配的情况，并且会加载默认值。

- 时长

  Tprog (3.2 ms)。

有关代码示例，请参见 *A.3.8：擦除选项字节的代码示例*。

对 Flash 程序存储器中的单个字进行编程

- **目的**

  将特定值写入 Flash 程序存储器中的一个字。

- **大小**

  仅支持按字写入。

- **地址**

  选择 Flash 程序存储器中的地址。

- **协议**

  单次编程操作。

- **请求**

  PELOCK = 0，PRGLOCK = 0。

- **错误**

  如果 PELOCK = 1 或 PRGLOCK = 1，WRPERR 置 1。

  如果用户尝试写入受写保护的扇区，WRPERR 置 1（请参见 *PcROP（专有代码读取保护）*部分）。

  如果用户尝试将值写入并非为零的一个字，NOTZEROERR 置 1。该错误不会停止第 3 类器件的写入操作，而会停止其他类别器件的写入操作。

  如果大小不等于一个字，SIZERR 置 1。

- **说明**

  该操作用于对 Flash 程序存储器中的一个字执行写入操作。用户必须以一个字的大小将正确的值写入正确的地址。存储器接口在执行写入操作前无法通过擦除操作删除之前的内容。

  如果之前的内容不为空：

  – 第 3 类器件

    NOTZEROERR 置 1。

    写入存储器的实际值是对之前的值与新值执行 OR 运算所得的值（如果之前为 0，存储器接口写入 1）。该操作针对数据和 ECC 执行。读回数据可能不会返回原有值、新值或执行 OR 运算后所得的值。ECC 不再与数据兼容。

  – 其他类别

    NOTZEROERR 置 1。不能对包含非空值的地址写入一个字。

- **时长**

  Tprog (3.2 ms)。

有关代码示例，请参见 *A.3.9：对 Flash 程序存储器中的单个字进行编程的代码示例*。

对 Flash 程序存储器的半页进行编程

- **目的**

  写入 Flash 程序存储器中的一个半页（16 个字）。

- **大小**

  仅支持按字写入。

- **地址**

  选择 Flash 程序存储器中与半页对齐（对于第一个地址而言）并位于由第二个地址为后 15 个地址选择的同一半页中的一个地址。

- **协议**

  多次编程操作。

- **请求**

  PELOCK = 0，PRGLOCK = 0，FPRG = 1 且 PRG = 1。

- **错误**

  如果 PELOCK = 1 或 PRGLOCK = 1，WRPERR 置 1。如果用户尝试写入受写保护的扇区，WRPERR 置 1（请参见 *PcROP（专有代码读取保护）*部分）。

  如果用户尝试将值写入并非为零的一个字，NOTZEROERR 置 1。该错误不会停止第 3 类器件的写入操作，而会停止其他类别器件的写入操作。当接收到所有 16 个地址后，在 Flash 存储器的当前写入阶段前，完成该检查。错误标志仅在执行所有检查后才置 1。

  如果大小不等于一个字，SIZERR 置 1。

  如果第一个地址未与半页对齐并且如果后续任一地址（地址 2 到地址 16）未处于第一个地址确定的半页，PGAERR 置 1。不会执行相关检查来验证地址是否增加或发生变化：该操作由存储器接口自动完成。重要的是第一个地址与半页对齐，而且之后的地址均位于同一半页中。

  如果写入操作因主器件在 NVM 中执行取指操作而中止，FWWERR 置为 1。读为数据不会停止写入操作。

- **说明**

  该操作用于对 Flash 程序存储器中的半页执行写入操作。用户必须以一个字的大小将 16 个所需值写入正确的地址（如多次编程操作中所述）。存储器接口在执行写入操作前无法通过擦除操作删除之前的内容（用户必须在执行写入操作前删除之前的内容）。

  对于单次编程操作，如果之前的内容非空：

  – 第 3 类器件

    NOTZEROERR 置 1。

    写入的值是之前的数据和新数据执行 OR 运算所得的值。这意味着读回写入的地址可能返回一个不同于写入值的值。

  – 其他类别

    NOTZEROERR 置 1。不能对包含非空值的地址写入一个字。

  当半页操作启动后，存储器接口等待 16 个地址/数据，（通过硬性故障）中止并非为取指操作的所有读访问（请参见*取指和预取*）。取指操作会中止半页操作。存储器内容保持不变，FLASH_SR 寄存器的 FWWERR 错误标志将置 1。为完成半页编程操作，应再次写入所需的所有值。

- **时长**

  Tprog (3.2 ms)。

有关代码示例，请参见 *A.3.10：对 Flash 程序存储器的半页进行编程的代码示例*。

对半页 Flash 程序存储器执行并行写入操作

- **目的**

  以并行方式对 Flash 程序存储器中的 2 个半页（每个存储区域 1 个半页）执行写入操作。

- **大小**

  仅支持按字写入。

- **地址**

  对于每个半页，必须在 Flash 程序存储器中选择与半页起始地址对齐的地址。之后的 15 个地址必须指向由第一个地址选择的半页。

  此外，相对于第一个半页的起始地址，第二个半页的地址必须位于不同的存储区域（仅检查第一个地址）。

- **协议**

  多次编程操作。

- **请求**

  PELOCK = 0，PRGLOCK = 0，FPRG = 1，PRG = 1 且 PARALLELBANK = 1。

- **错误**

  该操作产生的错误与对 Flash 程序存储器半页进行编程产生的错误相同。然而，所选的第二个半页与第一个半页位于同一个存储区域时也会生成 PGAERR。

  若在该操作执行过程中检测到错误，会中止整个程序（即两个存储区域）的运行。

- **说明**

  该操作以并行方式对 Flash 程序存储区域中的半页进行编程。这会提升整个 NVM 的初始编程速度。

  可选择从存储区域 1 或存储区域 2 开始。

- **时长**

  Tprog (3.2 ms)。

擦除 Flash 程序存储器中的一页

- **目的**

  删除 Flash 程序存储器中的一页（32 个字）。

- **大小**

  仅支持按字擦除（该操作会以一个字大小删除 Flash 程序存储器的一个页）。

- **地址**

  选择 Flash 程序存储器中的有效地址。

- **协议**

  单次编程操作。

- **请求**

  PELOCK = 0，PRGLOCK = 0，ERASE = 1 且 PRG = 1。

- **错误**

  如果 PELOCK = 1 或 PRGLOCK = 1，WRPERR 置 1。
  如果一行位于受保护的扇区中，则 WRPERR 置 1（请参见 *PcROP（专有代码读取保护）*）。

  如果大小不等于一个字，SIZERR 置 1。

- 说明

  该操作用于删除 Flash 程序存储器一行中的内容。用户必须以一个字的大小将值写入正确的地址。数据并不重要：仅执行擦除操作（使用非零数据也同样如此）。地址无需与页对齐：存储器接口将删除包含地址的页。

- 时长

  Tprog (3.2 ms)。

有关代码示例，请参见 *A.3.11：擦除 Flash 程序存储器中一页的代码示例*。

批量擦除

- 目的

  删除 Flash 程序存储器和数据 EEPROM 中的读保护和写保护。

- 大小

  仅支持按字擦除。

- 地址

  为引发批量擦除操作，需要在实际保护级别为级别 1 时将 0x015500AA 写入选项字节第一个地址（位 31 到位 25 与位 15 到位 9 并不互补，原因在于未使用这些位而且未进行检查）。

- 协议

  单次编程操作。

- 请求

  PELOCK = 0，OPTLOCK = 0，保护级别 = 1，数据较低的半字节必须为 0xAA（级别 0），同时第三个半字节必须为 0x55。

- 错误

  如果 PELOCK = 1 或 OPTLOCK = 1，WRPERR 置 1。

  如果实际保护级别为 2（在级别 2 无法对选项字节区域执行写入操作），WRPERR 置 1。

  如果大小不等于一个字，SIZERR 置 1。

- 说明

  该操作与对用户选项字节区域执行的写入操作类似：当请求的保护级别为 0，但实际保护级别为 1 时，存储器接口在批量擦除操作中会对其进行更改。用户必须以一个字的大小将所需值写入选项字节区域的第一个地址。

  批量擦除操作会删除 Flash 程序存储器和数据 EEPROM 的内容、将保护级别更改为级别 0 并禁止 PcROP (WPRMOD = 0)。写保护位和 BOR_LEVEL 位保持不变。

  与其他所有操作不同，软件在执行批量擦除操作的过程中无法请求执行新的写入操作。为确保批量擦除操作已完成，软件可在执行写入操作前复位 FLASH_SR 寄存器的 EOP 位并检查 EOP 何时置 1（编程结束）。如果未遵循此限制条件，在写入保护级别后，可能会向 Flash 程序存储器和数据 EEPROM 写入错误的值，从而添加可能导致器件无效的意外保护（也会导致不匹配的情况）。

- 时长

  2 x Tprog (6.4 ms) + Tglob (3.7 ms)

有关代码示例，请参见 *A.3.12：批量擦除代码示例*。

时序表

表 17. NVM 写入/擦除时序

操作	锁存第一个地址/数据的延迟（在 AHB 时钟脉冲中）	锁存下一个地址/数据的延迟（在 AHB 时钟脉冲中）
对数据 EEPROM 执行写入操作	18	-
对数据 EEPROM 执行擦除操作	17	-
对选项字节执行写入操作	18	-
对选项字节执行擦除操作	17	-
对 Flash 程序存储器中的单个字进行编程	78	-
对 Flash 程序存储器的半页进行编程	63	6
擦除 Flash 程序存储器中的一页	76	-

表 18. NVM 写入/擦除时长

操作	参数/条件	时长
对数据 EEPROM 执行写入操作	前一数据 = 0 FIX = 0	Tprog (3.2 ms)
	前一数据 /= 0 新数据 = 0 大小 = 字 FIX = 0	Tprog (3.2 ms)
	其他情况	2 x Tprog (6.4 ms)
对数据 EEPROM 执行擦除操作	-	Tprog (3.2 ms)
对选项字节执行写入操作	前一数据 = 0 FIX = 0	Tprog (3.2 ms)
	前一数据 /= 0 新数据 = 0 FIX = 0	Tprog (3.2 ms)
	其他情况	2 x Tprog (6.4 ms)
对选项字节执行擦除操作	-	Tprog (3.2 ms)
对 Flash 程序存储器中的单个字进行编程	-	Tprog (3.2 ms)
对 Flash 程序存储器的半页进行编程	-	Tprog (3.2 ms)
擦除 Flash 程序存储器中的一页	-	Tprog (3.2 ms)
批量擦除	-	2 x Tprog (6.4 ms) + Tglob (3.7 ms)

状态寄存器

FLASH_SR 状态寄存器提供有关存储器接口或 NVM 状态（正在执行的操作）的信息以及已经出现的错误的相关信息。

BSY

此标志由硬件置 1 和复位。存储器接口每次执行写入/擦除操作时，该位均会置 1，指示无法执行其他操作。若请求执行新操作，会发生其他行为：

- 等待读取、等待写入/擦除或等待选项加载：

  如果软件在执行写入/擦除操作 (HVOFF = 0) 的同时请求执行写入操作，存储器接口会使主器件停止运行并在写入/擦除操作完成后立即执行挂起的操作。
- 硬性故障：

  存储器接口等待下一个地址/数据（BSY 已置为 1，但 HVOFF = 0）时，如果软件在半页操作中请求读取数据，存储器接口会发生硬性故障（原因在于无法执行读取操作）并继续等待丢失的地址。
- RDERR 错误：

  如果软件在执行写入/擦除操作 (HVOFF = 0) 的同时请求执行读取操作，但地址受到保护，则存储器接口会将该标志置 1 并继续等待写入/擦除操作结束。
- 写入中止：

  如果软件在存储器接口等待半页操作中的地址/数据时对 NVM 执行取指操作，则会中止写入/擦除操作，FWWERR 标志置 1 并会执行取指操作。

EOP

此标志由硬件置 1，由软件复位。软件在状态寄存器中对该标志写入 1 可将其复位。该位在写入/擦除操作完成后置 1，存储器接口可执行其他操作（或开始执行挂起的操作）。

在执行新的写入/擦除操作前，将该位清零有助于了解实际操作何时完成。请求执行新操作前，在执行批量擦除的过程中等待该标志置 1 至关重要。

HVOFF

该标志由硬件置 1 和复位，是从 NVM 复制的存储器接口信息：用于指示高压稳压器接通 (= 0) 还是断开 (= 1)。

PGAERR

此标志由硬件置 1，由软件复位。它指示出现对齐错误。该标志在下列情况下置 1：

- 半页操作中的第一个地址未与半页对齐（较低的 6 位等于零）。
- 半页操作中半页发生变化（由第一个地址选择的半页操作的地址 2 到 6 不处于同一半页）。

对齐错误会中止写入/擦除操作，并且会生成中断（如果 FLASH_PECR 寄存器中的 ERRIE = 1）。NVM 的内容保持不变。

如果该标志置 1，存储器接口会阻止其他所有半页操作。

若要复位该标志，需通过软件向其写入 1。

SIZERR

此标志由硬件置 1，由软件复位。它指示出现大小错误。该标志在下列情况下置 1：

- 按字节和半字对 Flash 程序存储器和选项字节执行写入操作。
- 按字节或半字对所有区域执行擦除（FLASH_PECR 寄存器中位 ERASE = 1）操作。

大小错误会中止写入/擦除操作，并且会生成中断（如果 FLASH_PECR 寄存器中 ERRIE = 1）。NVM 的内容保持不变。

若要复位该标志，需通过软件向其写入 1。

NOTZEROERR

此标志由硬件置 1，由软件复位。它指示应用程序软件正在尝试对包含非零值的一个或多个 NVM 地址执行写入操作。

除第 3 类器件外，满足该条件时始终会中止修改操作。对于第 3 类器件，非零错误不会中止写入/擦除操作，但值可能会收到损坏。

在按半页执行的写入操作中，将检查第一个地址/值和第二个地址/值之间的所有 16 个字，标志仅在检查完所有字后才置 1。如果标志置 1，则意味着至少有一个字的实际值不为零。

在按字执行的写入操作中，只检查目标字，如果内容不为零，则立即将标志置 1。

如果 FLASH_PECR 寄存器中 ERRIE = 1，则会生成中断。若要复位该标志，需通过应用程序软件将其编程为 1。

注：可以使用应用程序软件通过 FLASH_PECR 寄存器的 NZDISABLE 位禁止非零错误条件通知（即 NOTZEROERR 标志及相关中断）。但是，对于除第 3 类器件之外的所有器件，仍会在内部检查该条件并仍会阻止修改操作

3.4 存储器保护

用户可保护 NVM 部分（Flash 程序存储器、数据 EEPROM 和选项字节区域）免遭意外的写入操作并可防止黑客读取代码（意外的读取操作）。

该器件实现了三类保护。

3.4.1 RDP（读保护）

此类保护旨在保护 NVM 内容免遭意外的读取操作（黑客）。该保护由 FLASH_OPTR 寄存器的 RDPROT 位域进行管理。其值在自举期间从选项字节区域加载，并复制到只读寄存器中。

此类保护分为三个级别，具体定义如下：

- 级别 0：无保护

  当 RDPROT 设为 0xAA 时，保护级别设为级别 0。使能该级别后，如果未使能其他保护，则可以在 Flash 程序存储器、数据 EEPROM 和选项字节区域无限制地执行读写操作。此外，还可以对备份寄存器随意执行读写操作。

- 级别 1：存储器读保护

  当 RDPROT 设为除 0xAA 和 0xCC（这两个值分别用于级别 0 和级别 2）以外的任意值时，保护设为级别 1。该级别是对选项字节执行擦除操作后或者 RDPROT 字段中存在不匹配的情况时的默认保护级别。

  如果器件从 RAM 或系统存储器自举或者连接有调试器（单线），则将在内部保存存储器自举模式。该信息只能通过掉电复位、器件进入待机模式（系统复位或选项加载不足）或者在器件切换到级别 2 时复位。它可以保护 Flash 程序存储器和数据 EEPROM。如果在从 RAM、自举程序或调试程序自举的过程中设为保护级别 1，则需要通过掉电复位或进入待机模式才能执行用户代码。

  使能该级别时：

  – 如果具有调试功能（单线）或者器件从 RAM 或系统存储器自举，则不能对 Flash 程序存储器和数据 EEPROM 执行访问（用于取指和数据的读取操作以及写入操作），也不能对备份寄存器执行读取操作。如果用户尝试读取 Flash 存储器或数据 EEPROM，则会生成硬性故障。其他区域则不存在任何限制：可以对选项字节区域执行读取操作和写入/擦除操作，以及执行系统存储器中的代码或对系统存储器执行读取操作。

  – 当在 Flash 程序存储器中进行自举后，可以执行所有操作。

  – 向第一个选项字节写入将保护级别更改为级别 0 的值（字节 0 必须为 0xAA，字节 2 必须为 0x55）时，将引起批量擦除操作。批量擦除会删除 Flash 程序存储器和数据 EEPROM 的内容，删除第一个选项字节的内容并对其重新执行写入操作以使能级别 0 并禁止 PCROP (WPRMOD = 0)，并删除备份寄存器的内容。

- 级别 2：禁止调试和芯片读保护

  当 RDPROT 设为 0xCC 时，保护设为级别 2。使能该级别时，只能从 Flash 程序存储器自举，并且将禁止调试功能（单线）。此保护级别可防止对选项字节执行写入/擦除操作，并且一经设置无法再更改。应用程序可对 Flash 程序存储器和数据 EEPROM（只能从 Flash 程序存储器自举和执行客户代码）执行写入/擦除操作，并且能够访问备份寄存器。在使能级别 2 的情况下加载选项字节时，将删除 RAM 或系统存储器中与调试和自举相关的旧信息。

图 6 汇总了各保护级别之间的切换方式，*表 19* 给出了在选项字节中读取的值与保护级别之间的关系。

图 6. RDP 级别

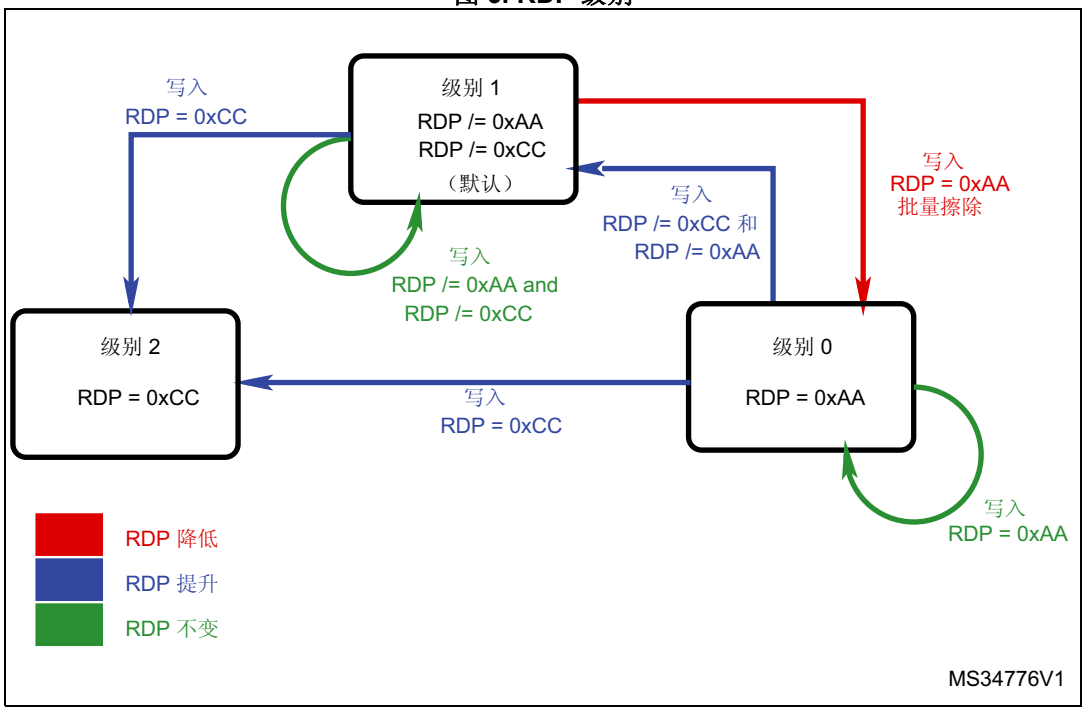


表 19. 保护级别与 RDP 选项字节的内容

RDP 字节值	RDP 补码值	读保护状态
0xAA	0x55	级别 0
0xCC	0x33	级别 2
其他任意值	RDP 字节的补码值	级别 1
任意值	非 RDP 字节的补码值	级别 1

3.4.2 PcROP（专有代码读取保护）

该保护可防止 Flash 程序存储器被黑客代码读取：阻止数据读取操作（而非取指操作）。受保护的代码不得访问受保护区域（包括缓冲池）的数据。

此保护可以防止 Flash 程序存储器被黑客代码读取：阻止数据读取操作（而非取指操作），前提是已根据 PcROP 选项编译原生代码。可通过将 FLASH_OPTR 寄存器中 WPRMOD 位置 1 激活该模式。

保护粒度为扇区（1 个扇区 = 32 页 = 4 KB）。要保护扇区，需将 WRPROT 配置中相应的位设为 0：0 表示受读写保护，1 表示无保护。

表 20 显示了 WRPROT 配置的各个位与 Flash 存储器扇区的地址之间的关系。

对受保护的扇区执行任何数据读访问（请参见*数据读取和预读*）都将触发 FLASH_SR 寄存器中的 RDERR 标志。任何受读保护的扇区同时也受写保护，对其中任一扇区执行任何写访问都将触发 FLASH_SR 寄存器中的 WRPERR 标志。

表 20. FLASH_WRPROTx 寄存器的保护位与 Flash 程序存储器中受保护的地址之间的关系

位	起始地址	结束地址	位	起始地址	结束地址
0	0x0800 0000	0x0800 0FFF	24	0x0801 8000	0x0801 8FFF
1	0x0800 1000	0x0800 1FFF	25	0x0801 9000	0x0801 9FFF
2	0x0800 2000	0x0800 2FFF	26	0x0801 A000	0x0801 AFFF
3	0x0800 3000	0x0800 3FFF	27	0x0801 B000	0x0801 BFFF
4	0x0800 4000	0x0800 4FFF	28	0x0801 C000	0x0801 CFFF
5	0x0800 5000	0x0800 5FFF	29	0x0801 D000	0x0801 DFFF
6	0x0800 6000	0x0800 6FFF	30	0x0801 E000	0x0801 EFFF
7	0x0800 7000	0x0800 7FFF	31	0x0801 F000	0x0801 FFFF
8	0x0800 8000	0x0800 8FFF	32	0x0802 0000	0x0802 0FFF
9	0x0800 9000	0x0800 9FFF	33	0x0802 1000	0x0802 1FFF
10	0x0800 A000	0x0800 AFFF	34	0x0802 2000	0x0802 2FFF
11	0x0800 B000	0x0800 BFFF	35	0x0802 3000	0x0802 3FFF
12	0x0800 C000	0x0800 CFFF	36	0x0802 4000	0x0802 4FFF
13	0x0800 D000	0x0800 DFFF	37	0x0802 5000	0x0802 5FFF
14	0x0800 E000	0x0800 EFFF	38	0x0802 6000	0x0802 6FFF
15	0x0800 F000	0x0800 FFFF	39	0x0802 7000	0x0802 7FFF
16	0x0801 0000	0x0801 0FFF	40	0x0802 8000	0x0802 8FFF
17	0x0801 1000	0x0801 1FFF	41	0x0802 9000	0x0802 9FFF
18	0x0801 2000	0x0801 2FFF	42	0x0802 A000	0x0802 AFFF
19	0x0801 3000	0x0801 3FFF	43	0x0802 B000	0x0802 BFFF
20	0x0801 4000	0x0801 4FFF	44	0x0802 C000	0x0802 CFFF
21	0x0801 5000	0x0801 5FFF	45	0x0802 D000	0x0802 DFFF
22	0x0801 6000	0x0801 6FFF	46	0x0802 E000	0x0802 EFFF
23	0x0801 7000	0x0801 7FFF	47	0x0802 F000	0x0802 FFFF

当 WPRMOD = 1（PcROP 使能）时，无法降低对扇区的保护级别：可以设置新的零（以保护新的扇区），但不能添加新的 1（以移除对扇区的保护）。这对于任何保护级别（RDPROT 配置）均有效。当 WPRMOD 激活时，如果用户尝试复位 WPRMOD 或移除对扇区的保护，则将启动编程但 WPRMOD 或受保护的扇区保持不变。

移除对扇区保护的唯一方式是请求批量擦除（这会将保护级别更改为级别 0 并禁止 PcROP）：禁止 PcROP 后，可随意更改对扇区的保护。

3.4.3 防止意外的写入/擦除操作

存储器接口采用了两种防止意外写入/擦除操作的方式，这两种方式对所有矩阵均有效，但只对 Flash 程序存储区的特定扇区有效。

如*解锁/锁定操作*部分所述，用户可以执行下列操作：

- 只能在 FLASH_PECR 寄存器的 PELOCK = 0 时对数据 EEPROM 执行写入/擦除操作。
- 只能在 FLASH_PECR 寄存器的 PELOCK = 0 且 OPTLOCK = 0 时对选项字节区域执行写入/擦除操作。
- 只能在 FLASH_PECR 寄存器的 PELOCK = 0 且 PRGLOCK = 0 时对 Flash 程序存储器执行写入/擦除操作。

有关设置 PELOCK、PRGLOCK 和 OPTLOCK 的序列，请参见*解锁数据 EEPROM 和 FLASH_PECR 寄存器*、*解锁 Flash 程序存储器*和*解锁选项字节区域*部分。

在 Flash 程序存储器中，可以以扇区为粒度添加额外写保护。禁止 PcROP (WPRMODE = 0) 时，WRPROT 的位用于使能对扇区的写保护。极性与 PcROP 相反：要保护扇区，需要将位置 1；要移除保护，需要将位设为 0。*表 20* 同样适用于写保护。如上文所述，使能 PcROP 时，受读保护的扇区同时也受写入/擦除保护。在级别 0 和级别 1 保护下始终可以更改对扇区的写保护（前提是可以对选项字节执行写入/擦除操作且禁止 PcROP）。

表 21 对保护情况进行了汇总。

表 21. 存储器访问与模式、保护和 Flash 程序存储器扇区

Flash 程序存储器扇区	模式				
	用户（包括在应用程序编程中）禁止调试功能，或未从 RAM 自举，或未从系统存储器自举		用户使能调试功能，或从 RAM 自举，或从系统存储器自举		
RDP	**级别 1 级别 0**	**级别 2**	**级别 0**	**级别 1**	**级别 2**
Flash 程序存储器 (FLASH_PRGLOCK = 1)	R	R	R	受保护（无保护）	NA(1)
Flash 存储器 (FLASH_PRLOCK = 0)	R/W	R/W	R/W	受保护（无保护）	NA(1)
Flash 程序存储器（WRP 页中）	R	R	R	受保护（无保护）	NA(1)
Flash 程序存储器（PCROP 页中）	取指	取指	取指	受保护（无保护）	NA(1)
数据 EEPROM (FLASH_PELOCK = 1)	R	R	R	受保护（无保护）	NA(1)
数据 EEPROM (FLASH_PELOCK = 0)	R/W	R/W	R/W	受保护（无保护）	NA(1)
选项字节 (FLASH_OPTLOCK = 1)	R	R	R	R	NA(1)
选项字节 (FLASH_OPTLOCK = 0)	R/W	R	R/W	R/W	NA(1)

1. NA 表示“不适用”。

3.4.4 写入/擦除保护管理

下文汇总了更改所有之前保护的规则：

- 保护级别为 2 时，无法执行任何保护更改。
- 保护级别为 0 或 1 时，始终能够更改为级别 2，将 xx33xxCC（x 为十六进制数，可以为任意值）写入选项字节的第一个字即可。
- 保护级别为 0 时，能够更改为级别 1，将除了 xx33xxCC（级别 2）或 xx55xxAA（级别 0）以外的任意值写入选项字节的第一个字即可。
- 保护级别为 1 时，可将保护级别降为级别 0，将 xx55xxAA 写入选项字节的第一个字即可。这将引起批量擦除操作，并会删除 PcROP 字段的内容。
- 始终能够使能 PcROP（级别为 2 时除外），将 x0xxx1xx 写入选项字节的第一个字即可。如果该标志在选项字节加载期间发生不匹配的情况，则使能 PcROP。
- 可在请求批量擦除（从级别 1 更改为级别 0）时移除 PcROP。
- PcROP 禁止时，可使用选项字节的第三个字添加（写入 1）或移除（写入 0）对扇区的保护。不匹配问题涉及受写保护的所有扇区（PcROP 禁止时）。
- PcROP 使能时，可以添加对扇区的保护（写入 0），但不能移除。不匹配问题涉及受读写保护的所有扇区（PcROP 使能时）。
- 批量擦除操作不会删除选项字节的第三个字：用户必须向其中写入正确的值。

3.4.5 保护错误

写保护错误标志 (WRPERR)

如果对 Flash 程序存储器和数据 EEPROM 的写保护页执行擦除/编程操作，则 FLASH_SR 寄存器的写保护错误标志位 (WRPERR) 将置 1。

因此，WRPERR 标志会在软件尝试执行下列操作时置 1：

- 对 WRP 页执行写入操作。
- 对系统存储器页或出厂选项字节执行写入操作。
- 在 Flash 程序存储器、数据 EEPROM 或选项字节未通过 PEKEY、PRGKEY 或 OPTKEY 解锁时对它们执行写入操作。
- 在 RDP 选项字节置 1 且器件处于调试模式或从 RAM 或系统存储器自举时对 Flash 程序存储器、数据 EEPROM 或选项字节执行写入操作。

写保护错误会终止写入/擦除操作，并且会生成中断（如果 FLASH_PECR 寄存器中 ERRIE = 1）。

若要复位该标志，需通过软件向其写入 1。

读错误 (RDERR)

如果软件尝试读取受 PcROP 保护的扇区，FLASH_SR 的 RDERR 标志会置 1。总线上接收到的数据为 0。

如果使能错误中断（FLASH_PECR 寄存器的 ERRIE = 1），则会生成中断。

若要复位该标志，需通过软件向其写入 1。

3.5 NVM 中断

若将 FLASH_PECR 寄存器的编程结束中断使能位 (EOPIE) 置 1，可在擦除或编程操作成功结束时生成中断。此时，FLASH_SR 寄存器中的编程结束 (EOP) 位置 1。若要复位该位，需通过软件向其写入 1。

若将 FLASH_PECR 寄存器的错误中断使能位 (ERRIE) 置 1，可在请求编程或擦除操作期间出现错误时生成中断。此时，FLASH_SR 寄存器中会有一个或多个错误标志置 1：

- RDERR（PCROP 读保护错误标志）
- WRPERR（写保护错误标志）
- PGAERR（编程对齐错误标志）
- OPTVERR（选项有效性错误标志）
- SIZERR（大小错误标志）
- FWWERR（写入时取指错误标志）
- NOTZEROERR（写入非零字错误标志）

若要复位错误标志，需通过软件向相应的标志写入 1。

表 22. Flash 中断请求

中断事件	事件标志	使能控制位
操作结束	EOP	EOPIE
错误	RDERR WRPERR PGAERR OPTVERR SIZERR FWWERR NOTZEROERR	ERRIE

3.5.1 硬性故障

硬性故障会在下列位置生成：

- 如果在 RDP 置 1 时尝试读访问，则会在存储器总线上生成。
- 如果接收到读数据请求，则会在存储器总线上生成；之后，存储器接口会在半页写入操作期间（第 1 个地址之后，第 16 个地址之前）等待数据/地址。
- 如果向 PEKEYR、PRGKEYR 或 OPTKEYR 写入的值不正确，则会在寄存器总线上生成。

3.6 存储器接口管理

本部分旨在阐明当正在执行某一操作时请求执行另一操作会出现的情况：不同操作间的协作方式，由存储器接口进行管理。

3.6.1 操作优先级和流程

操作分为三类，流程各不相同：

读取

- 如果当前未执行任何操作且读地址未受保护，则以实际配置无延迟地执行读取操作。
- 如果读地址受保护，则会滤掉读取操作（从不将读取操作请求发送到存储器）并生成错误。
- 如果读地址未受保护但存储器接口繁忙以致无法执行操作，则会将读取操作搁置以待不繁忙时尽快执行。

写入/擦除

- 如果当前未执行任何操作且写地址未受保护，则写入/擦除操作将立即开始执行；经过几个时钟脉冲（请参见*表 17*，期间会阻止总线和主器件）后，存储器接口会继续执行操作，释放总线和主器件。
- 如果地址受保护，则会滤掉写入/擦除操作（从不将写入/擦除操作请求发送到存储器）并会生成错误。
- 如果地址未受保护但未满足一个或多个条件，则会中止操作（中止操作需要较长时间，因为 NVM 和数据 EEPROM 需要返回到默认配置）并生成错误。
- 如果待执行写入/擦除的地址未受保护且满足所有规则，同时存储器接口处于繁忙状态，则会将操作搁置以待不繁忙时尽快执行。

选项字节加载

- 如果正在执行写入/擦除操作，则会等待操作结束时再执行选项字节加载操作：不接受任何其他写入/擦除请求，即使等待期间也是如此。
- 如果当前未执行写入/擦除操作，则会直接执行选项字节加载操作（执行读取操作直至因选项字节请求使系统复位为 0）。

这意味着，选项字节加载操作的优先级高于读取操作和写入/擦除操作。所有其他操作均按照请求的顺序执行。

3.6.2 操作序列

当使用第 5 类器件的“同时读写”功能同时对两个存储区域执行不同操作时，下列说明不适用。

执行写入操作时请求读取数据

主器件在写入操作执行时请求读取数据（请参见*数据读取和预读*）的情况分为以下三种：

1. 如果请求对受保护区域执行读取操作，则 RDERR 标志置 1，写入操作继续执行。
2. 如果写入操作使用*单次编程操作*或*多次编程操作（半页）*，并且所有地址 / 数据已发送至存储器接口，则将读取操作搁置以待写入操作完成后执行。需要强调的是，在读取操作等待执行期间，主器件将停止工作并且不能执行其他任何操作，直到读写操作完成。
3. 如果写入操作使用*多次编程操作（半页）*，并且并未将所有地址/数据发送至存储器接口，则拒绝读取操作请求并生成硬性故障，同时存储器接口继续等待缺失的地址/数据以完成写入操作。

执行写入操作时请求取指

主器件在写入操作执行时请求执行取指操作的情况也分为三种，前两种与读取数据的情况类似（请参见 *1* 和 *2*），最后一种情况有所不同，具体如下：

- 如果写入操作使用*多次编程操作（半页）*，并且并未将所有地址/数据发送至存储器接口，则将写入操作视为从未发生过并将其终止：接受读取操作请求，将请求读取的值发送给主器件。

执行写入操作时请求执行另一写入操作

主器件在写入操作执行时请求执行另一写入操作的情况分为以下几种：

- 如果正在执行的写入操作使用*单次编程操作*或*多次编程操作（半页）*，并且已将所有地址/数据发送至存储器接口，而新的写入操作针对受保护区域，则 WRPERR 标志会置 1，同时继续执行当前的写入操作并删除新的写入操作。
- 如果正在执行的写入操作使用*单次编程操作*或*多次编程操作（半页）*，并且已将所有地址/数据发送至存储器接口，而新的*单次编程操作*或*多次编程操作（半页）*并非针对受保护区域，则将新的写入操作搁置以待当前写入操作完成后执行。需要强调的是，请求第二个写入操作的主器件将停止工作，直到第一个写入操作完成且第二个写入操作已在内部存储地址和数据。
- 正在执行批量擦除操作时禁止请求执行新的写入操作：在批量擦除的所有阶段，将不在内部存储数据，新数据会更改存储为保护的值，进而增加不需要的保护。
- 当第一个写入操作使用*单次编程操作*或*多次编程操作（半页）*，并且已将所有地址 / 数据发送至存储器接口时，可以更改配置以准备新的写入操作。

3.6.3 执行读取操作时更改等待周期数

要更改等待周期数，必须对 FLASH_ACR 寄存器执行写入操作。读写寄存器与读写存储器使用不同的接口。存储器接口正在执行读取操作时不能更改等待周期数，如果请求已发送至寄存器接口，存储器接口不能停止工作。因此，当一个主器件正在读取存储器而另一个主器件更改等待周期数时，将锁定寄存器接口，直到更改生效（读取操作停止）为止。要使更改等待周期数的主器件停止工作，务必读回 FLASH_ACR 寄存器的内容：无法获悉更改等待周期数所需的时钟周期数，因为这取决于客户代码。

3.6.4 掉电

要将 NVM 置于掉电状态，必须执行解锁序列。

下列序列用于解锁 FLASH_ACR 寄存器的 RUN_PD 位：

- 将 PDKEY1 = 0x04152637 写入 FLASH_PDKEYR 寄存器。
- 将 PEKEY2 = 0xFAFBFCFD 写入 FLASH_PDKEYR 寄存器。

必须确保写入的两个密钥不会限制其他读写操作。如果使用的密钥有误，不会生成错误：当写入两个密钥后，RUN_PD 位解锁并可写入 1，以将 NVM 置于掉电模式。

若将 RUN_PD 标志复位为 0（使 NVM 可用），会自动复位序列，同时请求写入两个密钥以重新使能 RUN_PD。

3.7 Flash 寄存器说明

对寄存器执行读取操作

要读取存储器接口的所有内部寄存器，用户必须读取寄存器地址。内容立即可用（读寄存器无需等待周期）。如果用户在修改等待周期数后尝试读取 FLASH_ACR 寄存器，内容将在更改生效后可用（未对 NVM 存储器执行读取操作时，因此等待周期数会发生变化）。

当未选择寄存器时或将错误地址发送至存储器接口时，将发送零值作为应答。不会生成任何错误。

当主器件发送读取 8 位或 16 位内容的请求时，存储器接口会将寄存器内容的相应部分返回到数据输出总线上。例如，如果寄存器内容为 0x12345678 而主器件发送读取第二个字节的请求，输出将为 0x34343434（因为 0x34 是从零开始对字节进行计数时寄存器第二个字节的内容）。类似地，如果主器件发送读取上述寄存器半字零的请求时，输出将为 0x56785678。

对寄存器执行写入操作

在存储器接口的配置寄存器中，位分为以下两种：

- 可直接写入的位
- 需要使用特定序列解锁的位

要了解某位属于哪一种，请参见后续部分中对每个位的详细介绍。

当可以直接对寄存器或密钥寄存器执行写入操作时，用户必须将预期的值写入寄存器地址。如果地址不正确，不会生成错误。如果用户尝试修改只读寄存器，不会生成错误，并且修改操作不起任何作用。可以按字节、半字和字对寄存器执行写入操作。

需要解锁序列时，要给出要使用的正确值。

3.7.1 访问控制寄存器 (FLASH_ACR)

Access control register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_READ	DISAB_BUF	RUN_PD	SLEEP_PD	Res.	PRFTEN	LATENCY
									rw	rw	rw	rw		rw	rw

位 31:7 保留，必须保持复位值

位 6 **PRE_READ**

该位使能预读操作。

0：禁止预读操作

1：使能预读操作 存储器接口存储读取数据操作读取的最后一个地址，并在无任何其他读写操作或预取操作执行的情况下尝试读取下一个地址。

注： 每次（该寄存器中的）DISAB_BUF 位置 1 时，该位都自动复位。

位 5 **DISAB_BUF**

该位禁止读取操作期间用作缓存的缓冲区。这意味着，每次读取操作都将访问 NVM，甚至包括已经读取的地址（例如，上一个地址）。该位复位时，PRFTEN 和 PRE_READ 位也会自动复位。

0：使能缓冲区

1：禁止缓冲区 每次需要一个 NVM 值时，都必须执行一个新的存储器读序列。

位 4 **RUN_PD**

该位确定当器件处于运行模式时，NVM 处于掉电模式还是空闲模式。只有在对 FLASH_PDKEYR 寄存器执行解锁序列后才可以对该位执行写入操作。

*第 3.6.4 节：掉电*中介绍了正确的序列。将该位写为 0 时，密钥会自动丢失，此时需要使用新的解锁序列将该位重新写为 1。

0：当器件处于运行模式时，NVM 处于空闲模式。

1：当器件处于运行模式时，NVM 处于掉电模式。

位 3 **SLEEP_PD**

该位用于在器件处于睡眠模式时，将 Flash 程序存储器和数据 EEPROM 置于掉电模式或空闲模式。

0：当器件处于睡眠模式时，NVM 处于空闲模式。

1：当器件处于睡眠模式时，NVM 处于掉电模式。

位 2 保留，必须保持复位值

位 1 **PRFTEN**

该位使能预取操作。每次（该寄存器中的）DISAB_BUF 位置 1 时，该位都自动复位。若要了解预取操作的工作原理，请参见*取指和预取*部分。

0：禁止预取操作

1：使能预取操作 存储器接口存储获取的最后一个地址，并在无任何其他读写操作正在执行的情况下尝试读取下一个地址。

位 0 **LATENCY**

该位的值指定读取 NVM 时需要 1 个还是 0 个等待周期。用户必须写入与内核频率和工作模式（电源）相关的正确值。可用的正确值可参见*表 13*。不会执行任何检查来验证配置是否正确。

若要提高时钟频率，用户必须先将该位更改为“1”，然后再提高频率。若要降低时钟频率，用户必须先降低频率，然后再将该位更改为“0”。

0：读取 NVM 中的字不需要使用等待周期。

1：读取 NVM 中的字需要使用一个等待周期。

3.7.2 编程和擦除控制寄存器 (FLASH_PECR)

Program and erase control register

偏移地址：0x04

复位值：0x0000 0007

只有在 FLASH_PEKEYR 中完成适当的写入序列以复位 PELOCK 位后，才可以对该寄存器执行写入操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NZDISABLE	Res.	Res.	Res.	Res.	OBL_LAUNCH	ERRIE	EOPIE
								rw					rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PARRALELBANK	Res.	Res.	Res.	Res.	FPRG	ERASE	FIX	Res.	Res.	Res.	DATA	PROG	OPT_LOCK	PRG_LOCK	PE_LOCK
rw					rw	rw	rw				rw	rw	rs	rs	rs

位 31:24 保留，必须保持复位值

位 23 **NZDISABLE**：非零校验通知禁止 (Non-Zero check notification disable)

该位置 1 时，在对程序区域或自举区域执行字或半页编程之前，应用程序软件不会检查之前的 NVM 内容是否为零。因此，NOTZEROERR 标志将始终保持为 0，并且即使满足上述条件也不会生成中断。默认情况下，NZDISABLE 设为 0。仅可在 PELOCK 为 0 时修改该位。

0：禁止错误中断

1：使能错误中断

第 3 类器件不具备该位，行为对应于 NZDISABLE=0 时的情况。

位 22:19 保留，必须保持复位值

位 18 **OBL_LAUNCH**

该位置 1 时，软件请求重载选项字节。选项字节重载不会使正在执行的修改操作停止，但会阻止新的操作。选项字节重载会引发系统复位。

0：选项字节加载完成。

1：选项字节加载即将执行。

注：只有在 OPTLOCK 为 0 时才能修改该位。锁定 OPTLOCK（或其他锁定位）不会将该位复位。

位 17 **ERRIE**：错误中断使能 (Error interrupt enable)

0：禁止错误中断。

1：使能错误中断。

注：只有在 PELOCK 为 0 时才能修改该位。锁定 PELOCK 不会将该位复位；中断保持使能状态。

位 16 **EOPIE**：编程结束中断使能 (End of programming interrupt enable)

0：禁止编程结束中断。

1：使能编程结束中断。

注：只有在 PELOCK 为 0 时才能修改该位。锁定 PELOCK 不会将该位复位；中断保持使能状态。

位 15 **PARALLELBANK**：双区编程模式 (Parallel bank programming mode)

当前未执行任何编程或擦除操作时，可通过软件将该位置 1 或清零。

该位置 1 时，可同时执行 2 个半页编程操作，第一个针对存储区域 1，第 2 个针对存储区域 2。

0：禁止双区模式

1：使能双区模式

仅第 5 类器件具备该位。

位 14:11 保留，必须保持复位值

位 10 **FPRG**：半页编程模式 (Half Page programming mode)

0：禁止半页编程。

1：使能半页编程。

注：该位可在 PELOCK 为 0 时进行修改。PELOCK 位置 1 时，该位复位。

位 9 **ERASE**

0：未请求任何擦除操作。

1：已请求擦除操作。

注：该位可在 PELOCK 为 0 时进行修改。PELOCK 位置 1 时，该位复位。

位 8 **FIX**

0：必要时，可在对数据 EEPROM 和选项字节区域执行编程操作之前自动执行擦除操作。编程时间可以是：

Tprog（编程操作）或 2 * Tprog（擦除操作 + 编程操作）。

1：执行编程操作之前始终会执行初步擦除操作，编程时间为：2 * Tprog。

注：该位可在 PELOCK 为 0 时进行修改。PELOCK 位置 1 时，该位复位。

位 7:5 保留，必须保持复位值

位 4 **DATA**

0：未选择数据 EEPROM。

1：已选择数据 EEPROM。

注：该位可在 PELOCK 为 0 时进行修改。当 PELOCK 置 1 时，该位复位。该位也并非很有用，因为数据 EEPROM 中的页和字的大小相同，但该位可用于区分擦除操作（按页）与按字执行的操作。

位 3 **PROG**

该位用于 Flash 程序存储器中的半页编程操作和页擦除操作。

0：未选择 Flash 程序存储器。

1：已选择 Flash 程序存储器。

注： 该位可在 PELOCK 为 0 时进行修改。PELOCK 位置 1 时，该位复位。

位 2 **OPTLOCK**：选项字节锁定 (Option bytes lock)

该位会阻止对用户选项字节区域和（该寄存器中）OBL_LAUNCH 位执行写入/擦除操作。只能向该位写入 1 以重新锁定。要复位为 0，需在 PELOCK 位为 0 时使用 OPTKEYR 寄存器执行正确的解锁序列（请参见*解锁选项字节区域*）。如果序列不正确，该位将保持锁定状态直到下一次系统复位，并且将生成硬性故障。如果在 PELOCK = 1 时执行序列，该位保持锁定状态，并且不生成硬性故障。解锁密钥如下：

– 第一个密钥：0xFBEAD9C8

– 第二个密钥：0x24252627

0：禁止对选项字节区域执行写入和擦除操作。

1：允许对选项字节区域执行写入和擦除操作。

注： PELOCK 置1 时，该位置 1。

位 1 **PRGLOCK**：程序存储器锁定 (Program memory lock)

该位用于阻止对 Flash 程序存储器执行写入/擦除操作。只能向该位写入 1 以重新锁定。要复位为 0，需在 PELOCK 位为 0 时使用 PRGKEYR 寄存器执行正确的解锁序列（请参见*解锁 Flash 程序存储器*）。如果序列不正确，该位将保持锁定状态直到下一次系统复位，并且将生成硬性故障。如果在 PELOCK = 1 时执行序列，该位保持锁定状态，并且不生成硬性故障。解锁密钥如下：

– 第一个密钥：0x8C9DAEBF

– 第二个密钥：0x13141516

0：禁止对 Flash 程序存储器执行写入和擦除操作。

1：允许对 Flash 程序存储器执行写入和擦除操作。

注： PELOCK 置1 时，该位置 1。

位 0 **PELOCK**：FLASH_PECR 锁定 (FLASH_PECR lock)

该位用于锁定 FLASH_PECR 寄存器。只能向该位写入 1 以重新锁定。要复位为 0，需使用 PEKEYR 寄存器执行正确的解锁序列（请参见*解锁数据 EEPROM 和 FLASH_PECR 寄存器*）。如果序列不正确，该位将保持锁定状态直到下一次系统复位，并且将生成一个硬性故障。解锁密钥如下：

– 第一个密钥：0x89ABCDEF

– 第二个密钥：0x02030405

0：解锁 FLASH_PECR 寄存器；该位可修改，且其他位处于解锁状态。允许执行数据写入/擦除操作。

1：锁定 FLASH_PECR 寄存器，不允许执行写入/擦除操作。

3.7.3 掉电密钥寄存器 (FLASH_PDKEYR)

Power-down key register

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
FLASH_PDKEYR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_PDKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0 该寄存器为只写寄存器。使用由两个按字执行的写入操作构成的序列（第一个写入 0x04152637，第二个写入 0xFAFBFCFD），可以解锁 FLASH_ACR 寄存器的 RUN_PD 位。有关详细信息，请参见*第 3.6.4 节：掉电*。

3.7.4 PECR 解锁密钥寄存器 (FLASH_PEKEYR)

PECR unlock key register

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
FLASH_PEKEYR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_PEKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0 该寄存器为只写寄存器。使用由两个按字执行的写入操作构成的序列（第一个写入 0x89ABCDEF，第二个写入 0x02030405），可以解锁 FLASH_PECR 寄存器。有关详细信息，请参见*解锁数据 EEPROM 和 FLASH_PECR 寄存器*。

3.7.5 编程和擦除密钥寄存器 (FLASH_PRGKEYR)

Program and erase key register

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
FLASH_PRGKEYR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_PRGKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:0 该寄存器为只写寄存器。使用由两个按字执行的写入操作构成的序列（第一个写入 0x8C9DAEBF，第二个写入 0x13141516），可以解锁 Flash 程序存储器。只有在 PELOCK 已解锁时才能执行该序列。有关详细信息，请参见*解锁 Flash 程序存储器*。

3.7.6 选项字节解锁密钥寄存器 (FLASH_OPTKEYR)

Option bytes unlock key register

偏移地址：0x14

复位值：0x0000 0000

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

位 31:0 该寄存器为只写寄存器。使用由两个按字执行的写入操作构成的序列（第一个写入 0xFBEAD9C8，第二个写入 0x24252627），可以解锁选项字节区域和 OBL_LAUNCH 位。只有在 PELOCK 已解锁时才能执行该序列。有关详细信息，请参见*解锁选项字节区域*。

3.7.7 状态寄存器 (FLASH_SR)

Status register

偏移地址：0x018

复位值：0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWWER	NOTZERO ERR
														rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RDERR	Res.	OPTVERR	SIZERR	PGAERR	WRPERR	Res.	Res.	Res.	Res.	READY	ENDHV	EOP	BSY
		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1					r	r	rc_w1	r

位 31:18 保留，必须保持复位值

位 17 **FWWERR**

当中止写入/擦除操作以执行取指操作时，该位由硬件置 1。这不属于真正的错误，但可用于指示写入/擦除操作并未执行。要复位该标志，可写入 1。

0：未中止写入/擦除操作以执行取指操作。

1：已中止写入/擦除操作以执行取指操作。

位 16 **NOTZEROERR**

当 Flash 程序存储器或系统存储器中的程序尝试覆盖非零区域时，该位由硬件置 1。在第 3 类器件中，该标志不会使编程操作停止：读回的值可能不是用户写入的值。要复位该标志，可写入 1。

0：对已擦除区域执行写入操作，或者存储器接口可在执行写入操作之前执行擦除操作。

1：写入操作尝试写入未擦除区域，而且存储器接口不能在执行写入操作之前执行擦除操作。除第 3 类器件外，其余器件都会中止修改操作。对于第 3 类器件，非零错误不会中止写入/擦除操作。

位 15:14 保留，必须保持复位值

位 13 **RDERR**

当用户尝试读取受 PcROP 保护的区域时，该位由硬件置 1。通过写入 1 可将该位清零。

0：未发生读保护错误。

1：发生一个读保护错误。

位 12 保留，必须保持复位值

位 11 **OPTVERR**：选项有效错误 (Option valid error)

当选项字节加载期间一个或多个配置不匹配时，该位由硬件置 1。这意味着，加载的配置可能与用户写入存储器的配置不同。通过写入 1 可将该位清零。

如果加载保护（WPRMOD、RDPROT 和 WRPROT）时出现错误，Flash 程序存储器中的源代码可能无法正确执行。

0：选项字节加载期间未发生错误。

1：选项字节加载期间发生一个或多个错误。

位 10 **SIZERR**：大小错误 (Size error)

当要编程的数据大小不正确时，该位由硬件置 1。通过写入 1 可将该位清零。

0：未发生大小错误。

1：发生一个大小错误。

位 9 **PGAERR**：编程对齐错误 (Programming alignment error)

当发生对齐错误时，该位由硬件置 1：半页操作的第一个字未与半页对齐，或者半页操作中第一个字后面的其中一个字与第一个字不属于同一半页。该位置 1 时，在写入 1 之前必须将其清零，并且不接受任何半页操作。

0：未发生对齐错误。

1：发生了一个对齐错误。

位 8 **WRPERR**：写保护错误 (Write protection error)

当待编程或擦除的地址受写保护时，该位由硬件置 1。通过写入 1 可将该位清零。

0：未发生保护错误。

1：发生了一个保护错误。

位 7:4 保留，必须保持复位值

位 3 **READY**

该位置 1 时，NVM 已准备好执行读取和写入/擦除操作。

0：NVM 未准备就绪。不能对其执行任何读取操作或写入/擦除操作。

1：NVM 已准备就绪。

位 2 **ENDHV**

该位由硬件置 1 和复位。

0：高电压对 NVM 执行写入/擦除操作。

1：高电压关闭，当前未执行写入/擦除操作。

位 1 **EOP**：编程结束 (End of program)

当操作未被中止时，该位在写入/擦除操作结束时由硬件置 1。该位由软件复位（写入 1）。

0：未发生 EOP 事件。

1：发生了 EOP 事件。如果 EOPIE 位置 1，则会生成中断。

位 0 **BSY**：存储器接口繁忙 (Memory interface busy)

正在执行写入/擦除操作。

0：未执行写入/擦除操作。

1：正在执行写入/擦除操作。

3.7.8 选项字节寄存器 (FLASH_OPTR)

Option bytes register

偏移地址：0x1C

复位值：0xX0XX 0XXX，具体取决于选项字节中编程的值。在生产阶段，该寄存器设为 0x8070 00AA。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
nBOOT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BFB2	nRST_STDBY	nRTS_STOP	WDG_SW	BOR_LEV[3:0]			
r								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WPRMOD	RDPROT							
							r	r	r	r	r	r	r	r	r

位 31 **nBOOT1**

该位与 BOOT0 输入引脚搭配使用，用于选择自举源：

– 如果 BOOT0 = 0 且 nBOOT1 = X，则在 Flash 程序存储器中自举。

– 如果 BOOT0 = 1 且 nBOOT1 = 0，则在 RAM 存储器中自举。

– 如果 BOOT0 = 1 且 nBOOT1 = 1，则在系统存储器中自举。

若要更改自举源，需要执行选项字节重载操作。如果该配置在选项字节加载期间发生不匹配，则为其加载 1。

如果器件的保护级别为级别 2，BOOT0 和 nBOOT1 将不起作用：始终强制在 Flash 程序存储器中自举。

位 30:24 保留，必须保持复位值

位 23 **BFB2**：从存储区域 2 自举 (Boot from Bank 2)

该位包含由器件 OPTL 加载的用户选项字节。该位用于从存储区域 2 自举。实际上，此位指示选择的是从系统存储器自举还是从 Flash 程序存储器自举。如果选择的是从系统存储器自举，则会通过软件根据每个存储区域开头前两个字的值跳转到存储区域 1 或存储区域 2。BFB2 置 1 时，用户 Flash 存储器在地址 0 没有别名。不过，可以使用 SYSCFG_CFGR1 寄存器的 MEM_MODE 位使系统 Flash 存储器在地址 0 处有别名。

0：从存储区域 1（第 5 类器件）或用户 Flash 存储器（其他类器件）自举

1：从系统存储器自举

注： 仅第 5 类器件具备该位。

位 22 **nRST_STDBY**

如果该配置在选项字节加载期间发生不匹配，则为其加载 1。

0：进入待机模式时产生复位。

1：不产生复位

位 21 **nRST_STOP**

如果该配置在选项字节加载期间发生不匹配，则为其加载 1。

0：进入停机模式时产生复位。

1：不产生复位

位 20 **WDG_SW**

如果该配置在选项字节加载期间发生不匹配，则为其加载 1。

0：硬件看门狗。

1：软件看门狗。

位 19:16 **BOR_LEV**：欠压复位阈值级别 (Brown out reset threshold level)

这些位复位 1.45 V 到 1.55 V 电压范围的阈值级别（仅限掉电）。在此特殊情况下，V_{DD} 必须高于 V_{BOR0} 才能启动器件 OBL 序列，从而禁止 BOR。PDR 随后将监视掉电。如果禁止 BOR，1.65 V 与 VPDR 阈值之间会存在一个“灰色区域”（这意味着，V_{DD} 可低于最低工作电压 (1.65 V)，到达 VPDR 阈值之前不会复位）。

如果该配置在选项字节加载期间发生不匹配，则为其加载 0x8。

0xxx：**BOR 关闭**。这是 1.45 V - 1.55 V 电压范围的复位阈值级别（仅限掉电）。

在此特殊情况下，V_{DD} 必须高于 BOR 级别 1 才能启动器件 OBL 序列，从而禁止 BOR。PDR 随后将监视掉电。

注： 如果禁止 BOR，1.65 V 与 VPDR 阈值之间会存在一个“灰色区域”（这意味着，V_{DD} 可低于最低工作电压 (1.65 V)，达到 VPDR 阈值之前不会引发复位）

1000：**BOR 级别 1**，是 V_{BOR0} 的复位阈值电压（大约 1.8 V）

1001：**BOR 级别 2**，是 V_{BOR1} 的复位阈值电压（大约 2.0 V）

1010：**BOR 级别 3**，是 V_{BOR2} 的复位阈值电压（大约 2.5 V）

1011：**BOR 级别 4**，是 V_{BOR3} 的复位阈值电压（大约 2.7 V）

1100：**BOR 级别 5**，是 V_{BOR4} 的复位阈值电压（大约 3.0 V）

注： 有关 BOR 级别的确切定义，请参见器件数据手册。

位 15:9 保留，必须保持复位值

位 8 **WPRMOD**

该位用于在 Flash 程序存储器扇区的写保护和读保护之间进行选择。如果该配置在选项字节加载期间发生不匹配，则为其加载 1。

0：禁止 PCROP。WRPROT 位用作对扇区的写保护。

1：使能 PCROP。WRPROT 位用作对扇区的读保护。

位 7:0 **RDPROT**：读保护 (Read protection)

这些位包含选项字节加载期间加载的保护级别。如果该配置在选项字节加载期间发生不匹配，则为其加载 0x00。

0xAA：级别 0

0xCC：级别 2

其他值：级别 1

3.7.9 写保护寄存器 1 (FLASH_WRPROT1)

Write protection register 1

偏移地址：0x20

复位值：0xXXXX XXXX，具体取决于选项字节中编程的值。在生产阶段，该寄存器设为 0x0000 0000。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPROT1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPROT1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:0 **WRPROT1**：写保护 (Write protection)

– 如果 FLASH_OPTR 寄存器中 WPRMOD = 0，则这些位包含 Flash 存储器的写保护配置（每个位保护 4 KB 扇区：第一个位保护第一个扇区，第二个位保护第二页，依次类推）。在这种情况下，1 = 扇区受保护，0 = 无保护。

– 如果 WPRMOD = 1，这些位用于防止执行数据读取操作（请参见*数据读取和预读*），同时还可防止执行写入操作，而且粒度相同，位和扇区的对应关系也相同。读保护不能防止执行取指操作。在这种情况下，1 = 无保护，0 = 扇区受保护。

当 WPRMOD = 0 时，可以无限制地将这些位置 1 或复位，从而更改相关的选项字节。

当 WPRMOD = 1 时，只能提高保护级别，这意味着用户只能添加零，不能添加 1。

批量擦除操作会删除 WPRMOD 位，但不会删除该寄存器的内容。执行批量擦除操作后，用户必须向相关的选项字节写入零以完全移除写保护。

如果该配置在选项字节加载期间发生不匹配，并且 FLASH_OPTR 寄存器中 WPRMOD 的内容为：

1，则为该配置加载 0x0000。

0，则为该配置加载 0xFFFF。

如果加载 FLASH_OPTR 寄存器的 WPRMOD 时发生不匹配（即加载 1），则为寄存器加载 0x0000。

3.7.10 写保护寄存器 2 (FLASH_WRPROT2)

Write protection register 2

偏移地址：0x80

复位值：0x 0000 XXXX，具体取决于选项字节中编程的值。在生产阶段，该寄存器设为 0x0000 0000。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPROT2 [15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16 保留，必须保持复位值

位 15:0 **WRPROT2**：写保护 (Write protection)

– 如果 FLASH_OPTR 寄存器中 WPRMOD = 0，则这些位包含 Flash 存储器的写保护配置（每个位保护 4 KB 扇区：第一个位保护第一个扇区，第二个位保护第二页，依次类推）。在这种情况下，1 = 扇区受保护，0 = 无保护。

– 如果 WPRMOD = 1，这些位用于防止执行数据读取操作（请参见*数据读取和预读*），同时还可防止执行写入操作，而且粒度相同，位和扇区的对应关系也相同。读保护不能防止执行取指操作。在这种情况下，1 = 无保护，0 = 扇区受保护。

当 WPRMOD = 0 时，可以无限制地将这些位置 1 或复位，从而更改相关的选项字节。

当 WPRMOD = 1 时，只能提高保护级别，这意味着用户只能添加零，不能添加 1。

批量擦除操作会删除 WPRMOD 位，但不会删除该寄存器的内容。执行批量擦除操作后，用户必须向相关的选项字节写入零以完全移除写保护。

如果该配置在选项字节加载期间发生不匹配，并且 FLASH_OPTR 寄存器中 WPRMOD 的内容为：

1，则为该配置加载 0x0000。

0，则为该配置加载 0xFFFF。

如果加载 FLASH_OPTR 寄存器的 WPRMOD 时发生不匹配（即加载 1），则为寄存器加载 0x0000。

3.7.11 Flash 寄存器映射

表 23. Flash 接口 - 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_READ	DESAB_8BUF	RUN_PD	SLEEP_PD	Res.	PRFTEN	LATENCY
	0x00000000																										0	0	0	0		0	0
0x004	FLASH_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NZDISABLE	Res.	Res.	Res.	Res.	OBL_LAUNCH	ERRIE	EOPIE	PARRALELBANK	Res.	Res.	Res.	Res.	FPRG	ERASE	FIX	Res.	Res.	Res.	DATA	PRG	OPTLOCK	PRGLOCK	PELOCK
	0x00000007									0					0	0	0	0					0	0	0				0	0	1	1	1
0x008	FLASH_ PDKEYR	PDKEYR[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_ PKEYR	PKEYR[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	FLASH_ PRGKEYR	PRGKEYR[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	FLASH_ OPTKEYR	OPTKEYR[31:0]																															
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWWERR	NOTZEROERR	Res.	Res.	RDERR	Res.	OPTVERR	SIZERR	PGAERR	WRPERR	Res.	Res.	Res.	Res.	READY	ENDHV	EOP	BSY
	0x0000000C															0	0			0		0	0	0	0					1	1	0	0
0x01C	FLASH_OPTR	nBOOT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BFB2	nRST_STBY	nRST_STOP	WDG_SW	BOR_LEV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	WPRMOD	RDPROT[7:0]							
	0xX0XX0XXX	X	X	X						X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X
0x020	FLASH_ WRPROT1	WRPROT1[31:0]																															
	0x0000XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x080	FLASH_ WRPROT2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPROT2[15:0]															
	0xXXX 0000																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

有关寄存器边界地址的信息，请参见*第 57 页上的第 2.2.2 节*。

3.8 选项字节

NVM 上有一个保留区域，其中存储一组用于配置产品的选项字节。有些选项字节在出厂时已写入，其余选项字节由最终用户配置。

最终用户管理的配置存储在选项字节区域（32 个字节）。为将其纳入考量范围，必须执行自举序列。该自举序列在退出待机模式时的上电复位后执行，或者通过软件重新加载选项字节来执行（*第 3.8.3 节：通过软件重载选项字节*）。选项字节在自举操作期间自动加载。它们用于设置 FLASH_OPTR 和 FLASH_WRPROTx 寄存器的内容。

自举期间读取的每个字都按照*表 24* 所述进行解析：低 16 位包含待复制到存储器接口寄存器中的数据；高 16 位包含低 16 位的补码值，用于检查所读取的数据是否正确。如果加载操作期间发生错误（高 16 位不是低 16 位的补码），则寄存器将存储默认值。检查操作通过配置来完成。*第 3.8.2 节*介绍了保护配置不匹配时会出现的情况。

在写入操作器期间，不会执行任何控制来检查字的高位部分是否为低位部分的补码：该检查必须由用户应用程序执行。

表 24. 选项字节格式

31-24	23-16	15-8	7-0
选项字节 1 的补码	选项字节 0 的补码	选项字节 1	选项字节 0

3.8.1 选项字节说明

选项字节可以从*表 25* 所列的存储器位置读取。

表 25. 选项字节构成

地址	[31:16]	[15:0]
0x1FF8 0000	nFLASH_OPTR[15:0]	FLASH_OPTR[15:0]
0x1FF8 0004	nFLASH_OPTR[31:16]	FLASH_OPTR[31:16]
0x1FF8 0008	nFLASH_WRPROT1[15:0]	FLASH_WRPROT1[15:0]
0x1FF8 000C	nFLASH_WRPROT1[31:16]	FLASH_WRPROT1[31:16]
0x1FF8 0010	nFLASH_WRPROT2[15:0]	FLASH_WRPROT2[15:0]

有关各个位的含义，请参见*第 3.7.8 节：选项字节寄存器 (FLASH_OPTR)* 和*第 3.7.9 节：写保护寄存器 1 (FLASH_WRPROT1)*。

3.8.2 加载保护标志时发生不匹配

如果选项字节加载期间发生不匹配的情况，则存储器接口将设置寄存器的默认值。

在选项字节区域，存在三类保护信息：

- **RDPROT**
  该配置设置保护级别。更改该级别会改变对 NVM 和产品的访问权限，如下文所述。默认值为级别 1。可以从级别 1 返回到级别 0，但数据 EEPROM 和 Flash 程序存储器中的所有内容都将被删除（批量擦除）。始终可以更改为级别 2，但加载级别 2 时不能更改保护级别（如果用户在选项字节中写入级别 2 但从未重载选项字节，则存储器接口将继续采用先前的保护级别工作，并且可以重新向选项字节区域写入不同的保护级别）。
- **WPRMOD**
  该标志与 RDPROT 无关，并会在 Flash 程序存储器受读保护或写保护时置 1。该标志置 1 时（读保护），将其复位的唯一方式是请求执行批量擦除操作（同时会返回到级别 0）。这意味着，当器件保护级别为级别 2 时，无法移除读保护。默认值为 1（读保护），该位发生不匹配还会导致 WRPROT 配置恢复默认值。
- **WRPROT**
  该配置设置 Flash 程序存储器的哪些扇区受读保护或写保护。如果禁止读保护 (WPRMOD = 0)，必须将相应的位设为 1 以保护扇区。如果使能读保护 (WPRMOD = 1)，必须将相应的位设为 0 以保护扇区。如果自举期间 WPRMOD 发生不匹配，该配置会加载全零，以便使 Flash 程序存储器的所有扇区均受读保护。如果已正确读取 WPRMOD 但读取 WRPROT 时发生不匹配，则寄存器在 WPRMOD = 1 时加载全零，在WPRMOD = 0 时加载全 1。

因此，保护发生不匹配会对代码的正常执行产生严重影响（如果代码位于 Flash 程序存储器中）：存在读保护时，只能执行取指操作。在 Flash 程序存储器中，有些值在代码执行期间以数据读取的方式（例如，常数）读取；对所有扇区施加读保护可防止执行 Flash 程序存储器中的应用程序代码。

3.8.3 通过软件重载选项字节

通过将 FLASH_PECR 寄存器的 OBL_LAUNCH 标志置 1，可以请求执行选项字节重载操作。只有在 OPTLOCK = 0（且 PELOCK = 0）时才能将该位置 1。将该位置 1 时，将完成当前执行的写入/擦除操作，但不会执行新的写入/擦除或读取操作。

重载选项字节会导致器件复位，但不会掉电。每次更改 NVM 中的选项字节后必须重载选项，以便使更改生效。重载操作可通过将 OBL_LAUNCH 置 1 或通过 V18 域上电（即，上电复位后或退出待机模式后）来实现。

4 循环冗余校验计算单元（CRC）

4.1 简介

CRC（循环冗余校验）计算单元使用多项式发生器从一个 8 位/16 位/32 位的数据字中产生 CRC 码。

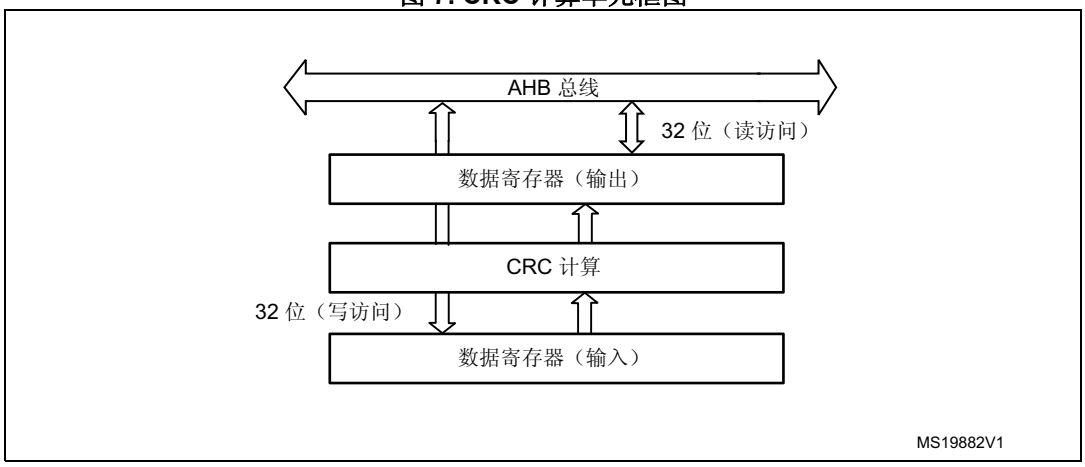
在众多的应用中，基于 CRC 的技术还常用来验证数据传输或存储的完整性。根据功能安全标准的规定，这些技术提供了验证 Flash 完整性的方法。CRC 计算单元有助于在运行期间计算软件的签名，并将该签名与链接时生成并存储在指定存储单元的参考签名加以比较。

4.2 CRC 主要特性

- 位数可编程的（7位、8位、16位和32位）的完全可编程多项式。
- 处理 8 位、16 位、32 位数据大小
- 可编程 CRC 初始值
- 单输入/输出 32 位数据寄存器
- 输入缓冲器可避免计算期间发生总线阻塞
- 对于 32 位数据大小，CRC 计算在 4 个 AHB 时钟周期 (HCLK) 内完成
- 8 位通用寄存器（可用于临时存储）
- I/O 数据的可逆性选项

4.3 CRC 功能说明

图 7. CRC 计算单元框图



CRC 计算单元具有单个 32 位读/写数据寄存器 (CRC_DR)。它用于输入新数据（写访问）和保存之前 CRC 计算的结果（读访问）。

对数据寄存器的每个写操作都会对之前的 CRC 值（存储在 CRC_DR 中）和新值再做一次 CRC 计算。CRC 计算针对整个 32 位数据字或逐个字节完成，具体取决于数据的写入格式。

CRC_DR 寄存器可按字、右对齐半字和右对齐字节进行访问。对于其它寄存器，只允许进行 32 位访问。

计算时间取决于数据宽度：

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

输入缓冲器中可立即写入第二个数据，无需因之前的 CRC 计算而等待任何等待状态。

可动态调整数据大小，从而能最大程度地减少给定字节数的写访问次数。例如，对 5 个字节进行 CRC 计算时，可先写入字，然后写入字节。

输入数据的顺序可反转，以管理各种数据存放方式（双字/单字/字节，大端/小端等等）。可对 8 位、16 位和 32 位数据执行反转操作，具体取决于 CRC_CR 寄存器中的 REV_IN[1:0] 位。

例如，输入数据 0x1A2B3C4D 在 CRC 计算中用作：

按字节执行位反转的 0x58D43CB2

按半字执行位反转的 0xD458B23C

按全字执行位反转的 0xB23CD458

通过将 CRC_CR 寄存器中 REV_OUT 位置 1 也可以将输出数据反转。

该操作按位进行：例如，输出数据 0x11223344 将转换为 0x22CC4488。

使用 CRC_CR 寄存器中的 RESET 控制位可将 CRC 计算器初始化为可编程值（默认值为 0xFFFFFFFF）。

可使用 CRC_INIT 寄存器对 CRC 初始值进行编程。对 CRC_INIT 寄存器进行写访问时会自动初始化 CRC_DR 寄存器。

CRC_IDR 寄存器可用于保存与 CRC 计算相关的临时值。它不受 CRC_CR 寄存器中的 RESET 位影响。

多项式可编程

多项式系数可完全通过 CRC_POL 寄存器进行编程，通过编程 CRC_CR 寄存器中的 POLYSIZE[1:0] 位可将多项式大小配置为 7 位、8 位、16 位或 32 位。不支持偶数多项式。

如果 CRC 数据小于 32 位，可从 CRC_DR 寄存器的最低有效位读取 CRC 的值。

为实现可靠的 CRC 计算，CRC 计算期间不能实时更改多项式的值或大小。因此，如果正在执行 CRC 计算，则在更改多项式前，应用程序必须先复位，或者先执行 CRC_DR 读操作。

默认的多项式值为 CRC-32（以太网）多项式：0x4C11DB7。

4.4 CRC 寄存器

4.4.1 数据寄存器 (CRC_DR)

Data register

偏移地址：0x00

复位值：0xFFFF FFFF

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

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

位 31:0 **DR[31:0]**：数据寄存器位

该寄存器用于向 CRC 计算器写入新数据。

读取寄存器时可读出之前的 CRC 计算结果。

如果数据大小小于 32 位，则最低有效位可用于写入/读取正确值。

4.4.2 独立数据寄存器 (CRC_IDR)

Independent data register

偏移地址：0x04

复位值：0x0000 0000

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

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

位 31:8 保留，必须保持清零。

位 7:0 **IDR[7:0]**：通用的 8 位数据寄存器位

这些位可用作一个字节的临时存储单元。

此寄存器不受 CRC_CR 寄存器中 RESET 位产生的 CRC 复位影响

4.4.3 控制寄存器 (CRC_CR)

Address offset: 0x08

偏移地址：0x08

复位值：0x0000 0000

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

位 31:8 保留，必须保持清零。

位 7 **REV_OUT**：反转输出数据 (Reverse output data)

该位用于控制输出数据位顺序的反转。

0：不影响位顺序

1：位反转输出格式

位 6:5 **REV_IN[1:0]**：反转输入数据 (Reverse input data)

这些位用于控制输入数据位顺序的反转。

00：不影响位顺序

01：按字节执行位反转

10：按半字执行位反转

11：按字执行位反转

位 4:3 **POLYSIZE[1:0]**：多项式大小 (Polynomial size)

这些位用于控制多项式的大小。

00：32 位多项式

01：16 位多项式

10：8 位多项式

11：7 位多项式

位 2:1 保留，必须保持清零。

位 0 **RESET**：RESET 位

此位由软件置 1，用于复位 CRC 计算单元并将数据寄存器设置为存储在 CRC_INIT 寄存器中的值。此位只能置 1，将由硬件自动进行清零

4.4.4 CRC 初始值 (CRC_INIT)

Initial CRC value

偏移地址：0x10

复位值：0xFFFF FFFF

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

位 31:0 **CRC_INIT**：*可编程 CRC 初始值 (Programmable initial CRC value)*
此寄存器用于写入 CRC 初始值。

4.4.5 CRC 多项式 (CRC_POL)

CRC polynomial

偏移地址：0x14

复位值：0x04C11DB7

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

位 31:0 **POL[31:0]**：可编程多项式 (Programmable polynomial)
此寄存器用于写入要用于 CRC 计算的多项式系数。
如果多项式大小小于 32 位，则必须使用最低有效位编程正确值。

4.4.6 CRC 寄存器映射

表 26. CRC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**CRC_IDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x08	**CRC_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	**CRC_INIT**	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	**CRC_POL**	Polynomial coefficients																															
	Reset value	0x04C11DB7																															

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

5 防火墙 (FW)

5.1 简介

防火墙用于保护非易失性存储器上的特殊代码或数据，和/或用于隔离 SRAM 上的易失性数据与其余在保护区域外执行的代码。

5.2 防火墙主要特性

- 防火墙保护的代码（代码段）可能位于以下区域：
  - Flash 程序存储器映射
  - SRAM 存储器，在防火墙配置阶段须将此存储器声明为可执行保护区。
- 被保护数据可位于以下区域：
  - Flash 程序存储器或者 EEPROM 数据存储器（非易失性数据段）
  - SRAM 存储器（易失性数据段）

防火墙打开后，软件可以访问上述保护区。通过使用一种基于“调用门”的机制可打开或关闭防火墙（请参见*打开防火墙*）。

在使能防火墙之前，必须先配置各数据段的起始地址及相应长度（请参见*第 5.3.5 节：防火墙初始化*）。

若防火墙已使能，则任何对这些保护段的非法访问均会产生复位，从而立即终止非法访问。

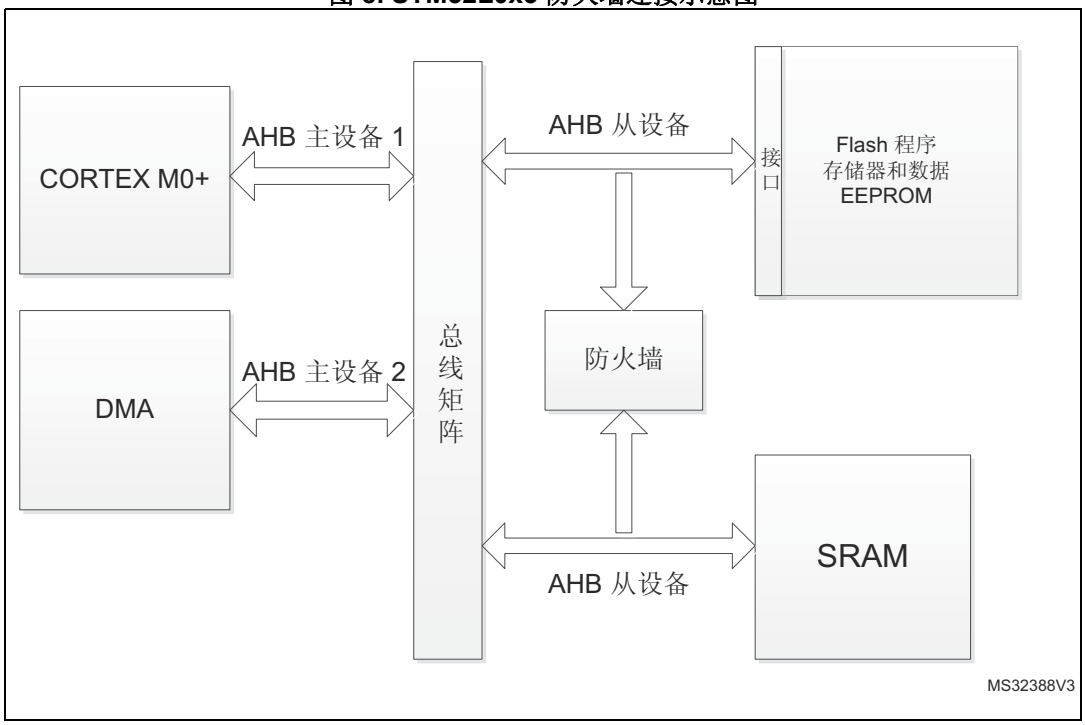
无论防火墙是否开启，任何对保护段的 DMA 访问均不允许。此类访问将视为非法访问并产生复位。

5.3 防火墙功能说明

5.3.1 防火墙 AMBA 总线监听

防火墙外设始终会监听存储器（易失性和非易失性）所连接的 AMBA 总线状态。整体架构图如*图 8* 所示。

图 8. STM32L0x3 防火墙连接示意图



5.3.2 功能要求

为保证防火墙所保护的应用程序代码和数据获得最高级安全等级，需要考虑以下若干事项，以防止产生不必要的防火墙报警（即产生复位）。

调试注意事项

在调试模式中，即使防火墙已开启，调试器仍可访问保护段。为此，在使用防火墙的同时，还需将读保护级别设置为 2 级。

若需要调试，则可按照如下方式进行操作：

- 在最终用户代码开发阶段期间，可开发伪代码，伪代码的 API 必须与被保护代码相同。由于被保护代码最终将应用在实际产品上，因此伪代码在功能实现上和时序要求（必要的话）上应与被保护代码保持一致。
- 在开发阶段，若签订了 NDA 协议，则被保护代码可交给用户端使用，相应软件可以在 0 保护等级下开发。如果产品的 ROP 等级为 2 级，为方便更新代码，需要在用户最终代码的写保护段中嵌入 IAP。

写保护

为提供最高安全等级，须遵守如下说明：

- 必须对使能防火墙的代码部分进行写保护。该激活代码应位于防火墙保护的代码段外。
- 还必须对防火墙保护的代码段进行写保护。
- 必须对包含复位向量的扇区进行写保护。

中断管理

防火墙保护的代码在执行期间不应接收任何中断请求。这取决于用户在执行被保护代码前是否禁止了所有中断源。如果不遵守此要求，则在被保护代码执行期间发现中断请求（防火墙开启的情况下）时，一旦执行中断子程序，防火墙就会立即关闭。在程序返回到被保护代区域时，由于未应用“调用门”序列，将生成防火墙报警，从而产生复位信号。

有关 Flash 程序存储器中的中断向量以及第一个用户扇区的信息如下：

- 如果第一个用户扇区（包含复位向量）受防火墙保护，则应在被保护段以外区域对此 NVIC 向量重新编程。
- 如果第一个用户扇区未受防火墙保护，则可将此中断向量放置在该扇区。

防火墙不产生任何中断。

5.3.3 防火墙段

防火墙用于保护以下三种不同段区域：

代码段

该位于 Flash 程序存储器中。其中应包含在执行时需由防火墙保护的代码。在访问此段时，须使用“调用门”序列打开防火墙。如果通过系统配置寄存器的 FWDIS 位使能了防火墙，而打开防火墙的“调用门”入口序列是错误的（请参见*打开防火墙*），将产生系统复位。使能防火墙前，必须先配置段长度和基地址（请参见*第 5.3.5 节：防火墙初始化*）。

非易失性数据段

该段用于存储防护墙保护的代码段所使用的非易失性数据。在*第 5.3.4 节：段访问方式及其属性*中对该段的访问方式进行了说明。在访问该区域的数据前，必须先打开防火墙。非易失性数据段应位于 Flash 程序存储器或者容量为 2Kb 的 EEPROM 数据存储器中。使能防火墙前，必须先配置段长度和基地址（请参见*第 5.3.5 节：防火墙初始化*）。

易失性数据段

被保护代码所使用的易失性数据必须存储在 SRAM 存储器中。在*第 5.3.4 节：段访问方式及其属性*中对该段的访问方式进行了说明。通过配置易失性数据段，可决定在访问该段之前是否必须开启防火墙。使能防火墙前，必须先配置段长度和基地址以及段选项（请参见*第 5.3.5 节：防火墙初始化*）。

此外，通过设置防火墙寄存器中的两个位，可将易失性数据段定义为可执行段（用于执行代码）或共享段，即位 VDE 用于定义易失性数据段是否为可执行段，位 VDS 用于定义易失性数据段是否为共享段。有关详细信息，请参见*表 27*。

5.3.4 段访问方式及其属性

禁止对被保护段进行任何形式的 DMA 访问，无论防火墙处于何种状态，这种访问都会引起系统复位。

不同防火墙状态下的段访问

上述三种段的访问属性各不相同，相关信息请参见*表 27*。

表 27. 不同防火墙状态下的段访问

段	防火墙开启 允许的访问	防火墙关闭 允许的访问	防火墙禁用 允许的访问
代码段	读取和执行	禁止任何访问。 对段进行任何访问（“调用门”条目除外）均会引起系统复位	允许各种访问操作（取决于代码所在 EEPROM 的保护特性）
非易失性数据段	读取和写入	禁止任何访问	允许各种访问操作（取决于代码所在 EEPROM 的保护特性）
易失性数据段	读取和写入 防火墙配置寄存器的 VDE = 1 且 VDS = 0 时，可进行执行访问	防火墙配置寄存器的 VDS = 0 且 VDE = 0 时，禁止任何访问 VDS = 1 时，无论 VDE 位的值为何，可进行读取/写入/执行访问 若 VDE = 1 且 VDS = 0，但先使用“调用门”条目打开防火墙，则可进行执行访问。	允许各种访问操作

易失性数据段的情况与其他两段的情况略有差异。易失性数据段可以是：

- 共享段（寄存器中 VDS 位）

  这意味着该段内的区域以及数据可在被保护代码和用户代码（在非保护区内执行）之间进行共享。访问共享段不受防火墙限制。

  在这种情况下，VDS 位的优先级高于 VDE 位，VDE 位的值被忽略。这表示易失性数据段可直接执行位于该段的代码，无需在执行前开启防火墙。

- 执行段

  当 FW_CR 寄存器中的 VDS = 0 时，将考虑 VDE 位。若 VDS = 1，请参见上述有关易失性数据共享段的描述。若 VDS = 0 且 VDE = 1，则易失性数据段为可执行段。为避免防火墙引起系统复位，应对易失性数据段应用“调用门”序列打开防火墙，将此序列作为代码执行的入口点。

段属性

系统为各个段分配了特定寄存器来定义防火墙所保护段的大小：CSL 为代码段长度寄存器，NVDSL 为非易失性数据段长度寄存器，VDSL 为易失性数据段长度寄存器。有关上述段的粒度和区域范围信息，请参见*表 28*。

表 28. 段粒度和区域范围

段	粒度	区域范围
代码段	256 字节	最大 64 KB 到 256 字节
非易失性数据段	256 字节	最大 64 KB 到 256 字节
易失性数据段	64 字节	8 KB 到 64 字节

5.3.5 防火墙初始化

防火墙初始化应在执行用户代码之前进行（请参见*写保护*）。

在初始化阶段，需要设置防火墙所保护段的地址和长度。由于防火墙使能位的值只能写入一次，因此在使能防火墙前完成此设置。于是，防火墙使能后，在下一次系统复位前将无法禁止。

防火墙使能后，将不能访问段地址和长度。所有写操作均视为无效操作。

若段长度定义为 0，则该段将不受防火墙保护。这时，对段基地址进行任何操作都不会使防火墙产生复位。

默认情况下，在复位后会禁止防火墙（SYSCFG 寄存器中 FWDIS 位置位）。必须清除此位方可使能防火墙功能。

防火墙初始化过程如下所示：

1. 配置 RCC 以使能防火墙模块的时钟
2. 配置 RCC 以使能系统配置寄存器的时钟
3. 设置各段的基地址和长度（CSSA、CSL、NVDSSA、NVDSL、VDSSA 和 VDSL 寄存器）
4. 设置防火墙配置寄存器（FW_CR 寄存器）
5. 通过清除系统配置寄存器的 FWDIS 位使能防火墙。

防火墙配置寄存器（FW_CR 寄存器）是唯一可以采用动态方式进行管理的寄存器（即使防火墙已使能）：

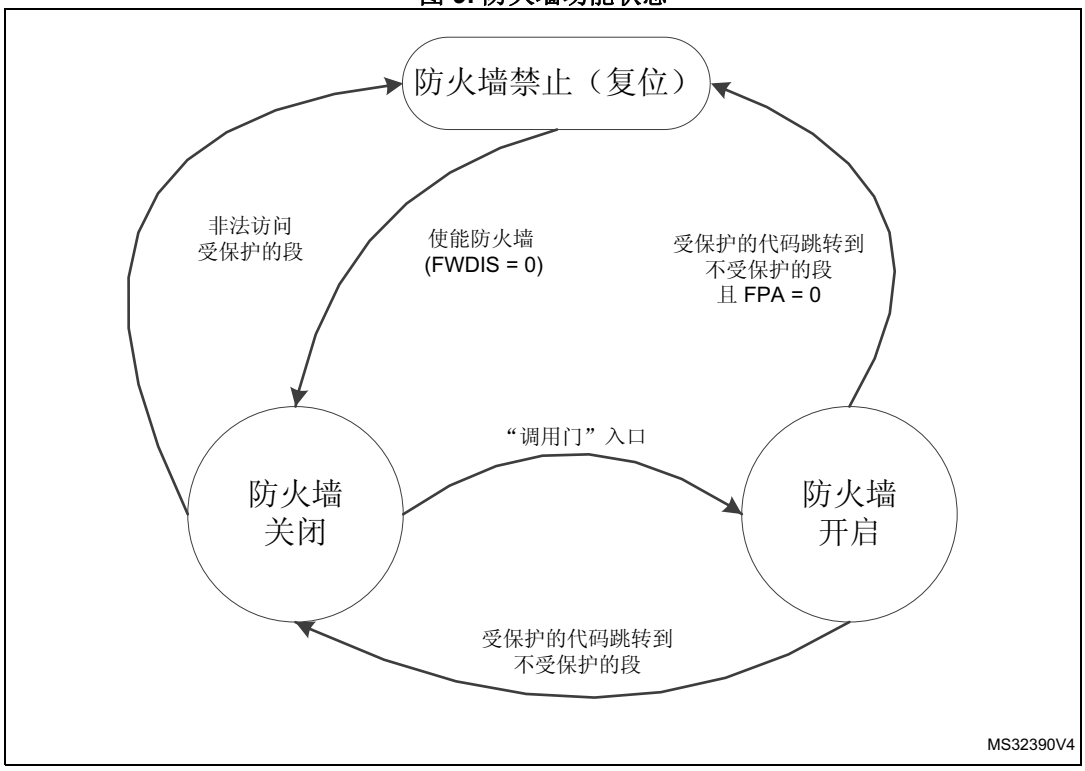
- 若未定义非易失性数据段（NVDSL 寄存器为 0），则无论防火墙是否打开，均可访问该寄存器。
- 若已定义非易失性数据段（NVDSL 寄存器为 1），则只有在防火墙打开的情况下，方可访问该寄存器。

5.3.6 防火墙状态

防火墙的状态分为三种，如*图 9* 所示：

- 禁止：复位后 FWDIS 位默认会置位。防火墙未激活。
- 关闭：防火墙限制对三类段（代码段、非易失性数据段和易失性数据段）的访问。
- 打开：防火墙允许对被保护段进行访问，如*第 5.3.4 节：段访问方式及其属性*所述。

图 9. 防火墙功能状态



打开防火墙

防火墙使能后，立即关闭。这表示对被保护段的大部分访问方式被禁止（请参见*第 5.3.4 节：段访问方式及其属性*）。为打开防火墙以与被保护段进行交互，必须采用以下所述的“调用门”序列。

“调用门”序列

“调用门”序列由 3 个字组成，它占据代码段和声明为执行段 (VDE = 1) 但非共享段 (VDS = 0) 的易失性数据段之基地址的头三个 32 位地址空间。

– 第一个字：空 32 位字，此字不对外开放，目的是防止预取缓冲器访问此字从而导致“调用门”机制开启。

– 第二个和第三个字：这两个特定的 32 位字即为“调用门”，始终处于开放状态。

要打开防火墙，当前执行的代码必须跳转至“调用门”的第二个字，并从该点执行代码。任何中间指令取操作均不得中断第二个字和第三个字的执行过程，否则防火墙不视为打开状态，而是回到关闭状态。若在接收到中间指令取操作后执行第三个字，则将产生系统复位。

防火墙打开后，可按照*第 5.3.4 节：段访问方式及其属性*中的描述访问被保护段。

关闭防火墙

防火墙在使能后会立即关闭（清除系统配置寄存器的 FWDIS 位）。

要关闭防火墙，受保护代码必须：

- 将正确的“防火墙预武装标志”值写入 FW_CR 寄存器中。
- 跳转至防火墙段以外的任何可执行位置。

如果未设置“防火墙预武装标志”位，则在被保护段跳转至非保护段时将产生复位。该控制位属于附加保护方法，可防止任何非法访问在私人信息未清除的情况下关闭防火墙，详情请见下文说明。

出于安全性考虑，建议在执行使用防火墙功能的应用程序后，清理 CPU 寄存器和硬件单元中的私人信息。

5.4 防火墙寄存器

5.4.1 代码段起始地址寄存器 (FW_CSSA)

Code segment start address

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD[23:16]							
								rw							

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

位 31:24 保留，必须保持复位值。

位 23:8 **ADD[23:8]**：代码段起始地址 (Code segment start address)

起始地址的 LSB 位（位 7:0）为保留位，被强制清零以满足 256 字节粒度的要求。

注： 这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 7:0 保留，必须保持复位值。

5.4.2 代码段长度寄存器 (FW_CSL)

Code segment length

偏移地址：0x04

复位值：0x0000 0000

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

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

位 31:22 保留，必须保持复位值。

位 21:8 **LENG[21:8]**：代码段长度 (Code segment length)

LENG[21:8] 设置代码段的大小，以字节表示，且与 256 字节成倍数关系。

段区域定义为从 {ADD[23:8], 0x00} 到 {ADD[23:8]+LENG[21:8], 0x00} - 0x01

注： 如果 LENG[21:8] = 0，则使能防火墙后，将视该段为未定义段，因而不受防火墙保护。这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 7:0 保留，必须保持复位值。

5.4.3 非易失性数据段起始地址寄存器 (FW_NVDSSA)

Non-volatile data segment start address

偏移地址：0x08

复位值：0x0000 0000

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

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

位 31:24 保留，必须保持复位值。

位 23:8 **ADD[23:8]**：非易失性数据段起始地址 (Non-volatile data segment start address)

起始地址的 LSB 位（位 7:0）为保留位，被强制清零以满足 256 字节粒度的要求。

注： 这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 7:0 保留，必须保持复位值。

5.4.4 非易失性数据段长度寄存器 (FW_NVDSL)

Non-volatile data segment length

偏移地址：0x0C

复位值：0x0000 0000

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

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

位 31:22 保留，必须保持复位值。

位 21:8 **LENG[21:8]**：非易失性数据段长度 (Non-volatile data segment length)

LENG[21:8] 设置非易失性数据段的大小，以字节表示，且与 256 字节成倍数关系。

段区域定义为从 {ADD[23:8], 0x00} 到 {ADD[23:8]+LENG[21:8], 0x00} - 0x01

注： 如果 LENG[21:8] = 0，则使能防火墙后，将视该段为未定义段，因而不受防火墙保护。这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 7:0 保留，必须保持复位值。

5.4.5 易失性数据段起始地址寄存器 (FW_VDSSA)

Volatile data segment start address

偏移地址：0x10

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:6 **ADD[15:6]**：易失性数据段起始地址 (Volatile data segment start address)

起始地址的 LSB 位（位 5:0）为保留位，被强制清零以满足 64 字节粒度的要求。

注： 这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 5:0 保留，必须保持复位值。

5.4.6 易失性数据段长度寄存器 (FW_VDSL)

Volatile data segment length

偏移地址：0x14

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:6 **LENG[15:6]**：非易失性数据段长度 (Non-volatile data segment length)

LENG[15:6] 设置非易失性数据段的大小，以字节表示，且与 64 字节成倍数关系。

段区域定义为从 {ADD[15:6], 0x00} 到 {ADD[15:6]+LENG[15:6], 0x00} - 0x01

注： 如果 LENG[15:6] = 0，则使能防火墙后，将视该段为未定义段，因而不受防火墙保护。

这些位只能在使能防火墙之前写入。请参见第 5.3.5 节：防火墙初始化。

位 5:0 保留，必须保持复位值。

5.4.7 配置寄存器 (FW_CR)

Configuration register

偏移地址：0x20

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDE	VDS	FPA
													rw	rw	rw

位 31:3 保留，必须保持复位值。

位 2 **VDE**：易失性数据可执行 (Volatile data execution)

0：若 VDS = 0，则无法执行易失性数据段

1：无论 VDS 位为何值，易失性数据段均声明为可执行段

若 VDS = 1，则该位无意义。此时无论 VDE 位为何值，均可执行易失性数据段。

若 VDS = 1，则无论防火墙是否开启，均可执行代码

若 VDS = 0，则仅当防火墙打开，或者在防火墙关闭情况下应用了“调用门”序列，才可执行代码。

请参见*不同防火墙状态下的段访问*。

位 1 **VDS**：易失性数据可共享 (Volatile data shared)

0：易失性数据段不对外共享，在防火墙关闭时，不会遭受来自非保护可执行代码的攻击。在这种情况下访问该段，防火墙将产生系统复位。

1：可与非保护应用程序代码共享易失性数据段。无论防火墙是否开启，均可访问该段。

请参见*不同防火墙状态下的段访问*。

位 0 **FPA**：防火墙预武装 (Firewall prearm)

0：如果防火墙已打开，则在被保护段以外执行任何代码都将产生系统复位。

1：在被保护段以外执行任何代码都将关闭防火墙。

请参见*关闭防火墙*。

该寄存器受保护的方式与易失性数据段相同（请参见*第 5.3.5 节：防火墙初始化*）。

5.4.8 防火墙寄存器映射

下表提供了防火墙寄存器映射和复位值。

表 29. 防火墙寄存器映射与复位值

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
0x0	FW_CSSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x4	FW_CSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value											0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x8	FW_NVDSSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0xC	FW_NVDSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value											0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x10	FW_VDSSA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD										Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value																	0	0	0	0	0	0	0	0	0	0						
0x14	FW_VDSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG										Res.	Res.	Res.	Res.	Res.	Res.
0x14	FW_VDSL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LENG										Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value																	0	0	0	0	0	0	0	0	0	0						
0x18		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value																																
0x1C		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset Value																																
0x20	FW_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDE	VDS	FPA
	Reset Value																														0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

6 电源控制 (PWR)

6.1 电源

当 BOR 可用时，器件的工作电压 (V_{DD}) 要求介于 1.8 V 到 3.6 V 之间（掉电时下至 1.65 V）。当 BOR 不可用时，器件的工作电压 (V_{DD}) 要求介于 1.65 V 到 3.6 V 之间。

嵌入式线性调压器用于提供内部数字电源，范围从 1.2 V 到 1.8 V。

- 当 BOR 可用时，V_{DD} = 1.8 V（上电时）或 1.65 V（掉电时）到 3.6 V。当 BOR 不可用时，V_{DD} = 1.65 到 3.6 V

  V_{DD} 是 I/O 和内部调压器的外部电源。通过 V_{DD} 引脚从外部提供
- V_{CORE} = 1.2 到 1.8 V

  V_{CORE} 是数字外设、SRAM 和 Flash 的电源。它由内部调压器产生。根据 V_{DD}，可通过软件选择三种 V_{CORE} 范围（请参见*图 11*）。
- 当 BOR 可用时，V_{SSA}、V_{DDA} = 1.8 V（上电时）或 1.65 V（掉电时）到 3.6 V。当 BOR 不可用时，V_{SSA}、V_{DDA} = 1.65 V 到 3.6 V。

  V_{DDA} 是 ADC、DAC、复位模块、RC 振荡器和 PLL 的外部模拟电源。使用 DAC 时，施加到 V_{DDA} 的最小电压为 1.8 V。
- V_{REF+}

  V_{REF+} 是输入参考电压。该电压仅在一些封装上用作外部引脚，否则它会连接到 V_{DDA}。
- V_{LCD} = 2.5 到 3.6 V

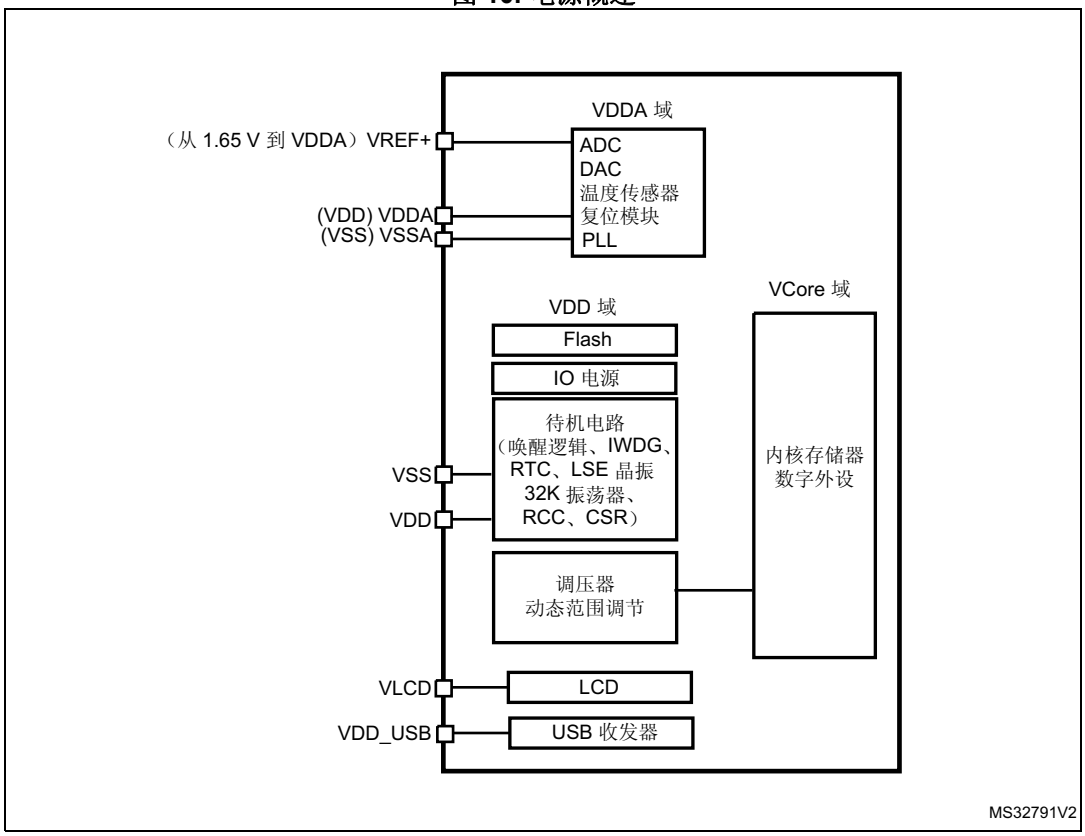
  LCD 控制器可通过 V_{LCD} 引脚外部供电，或由嵌入式升压转换器产生的内部电压内部供电。
- V_{DD_USB} = 3.0 到 3.6 V

  V_{DD_USB} 是为全速收发器供电的专用独立 USB 电源。如果将 PA11 和 PA12 引脚配置为 USB 复用功能，则可在这些引脚上提供该电源。

注： *V_{DD_USB} 值与 V_{DD} 和 V_{DDA} 无关。但是，V_{DD_USB} 电源必须最后提供给器件并且最先关闭。当关闭三个电源时，如果 V_{DD_USB} 在短时间内保持激活并且 V_{DDA}/V_{DDIO} 低于功能范围，则器件不会损坏。*

当 V_{DD_USB} 关闭时，器件仍能工作。

图 10. 电源概述



1. V_{DDA} 和 V_{SSA} 必须分别连接到 V_{DD} 和 V_{SS}。
2. 根据使用的工作电压范围，某些外设可能只提供有限的功能或性能。
3. 仅在 TFBGA64 封装上提供 V_{REF+}。

6.1.1 独立 A/D 和 DAC 转换器电源和参考电压

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

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

在具有多于 64 个引脚和 UFBGA64 的封装上

为确保低电压输入和输出上的更好精度，用户可将 V_{REF+} 连接至一个独立的，低于 V_{DD} 的外部参考电压源。对于模拟输入 (ADC) 或输出 (DAC) 信号，V_{REF+} 为最高电压，以满量程值表示。

对于 DAC：

$1.8\ V \leq V_{REF+} < V_{DDA}$

对于 ADC：

$1.65\ V \leq V_{REF+} < V_{DDA}$

在 64 引脚及以下的封装上（除了 BGA 封装）

V_{REF+} 引脚不可用。它内部连接到 ADC 电压源 (V_{DDA})。

6.1.2 独立 LCD 供电

V_{LCD} 引脚用于控制玻璃 LCD 的对比度。可用两种方法使用这一引脚：

- 它可从外部电路接收所需的最大电压，由微控制器通过区段和公用线供给玻璃 LCD。
- 还可用它连接外部电容，微控制器将该电容用于内部的升压转换器。通过软件控制此升压转换器，以向玻璃 LCD 的区段和公用线提供所需电压。

向区段和公用线提供的电压定义了玻璃 LCD 像素的对比度。当配置帧间死区时，可降低此对比度。

- 当向 V_{LCD} 引脚提供外部电压时，电压范围应当为 2.5 V 到 3.6 V。它不取决于 V_{DD}。
- 当 LCD 基于内部升压转换器时，V_{LCD} 引脚应连接到电容（有关详细信息，请参见产品数据手册）。

6.1.3 RTC 和 RTC 备份寄存器

实时时钟 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供一个日历时钟、两个可编程闹钟中断，以及一个具有中断功能的可编程的周期唤醒标志。RTC 包含 5 个备份数据寄存器（20 字节）。发生入侵检测事件时，将复位这些备份寄存器。有关详细信息，请参见*实时时钟 (RTC)* 部分。

RTC 寄存器访问

复位后，RTC 寄存器（RTC 寄存器和 RTC 备份寄存器）将受到保护，以防止可能的意外写访问。要使能对 RTC 寄存器的访问，请按以下步骤操作：

1. 将 RCC_APB1ENR 寄存器中的 PWREN 位置 1，使能电源接口时钟。
2. 将 PWR_CR 寄存器中的 DBP 位置 1（请参见*第 6.4.1 节*）。
3. 通过 RCC_RCC 寄存器中的 RTCSEL[1:0] 位选择 RTC 时钟源。
4. 通过对 RCC_CSR 寄存器中的 RTCEN 位进行编程，使能 RTC 时钟。

6.1.4 调压器

嵌入式线性调压器为待机电路以外的所有数字电路供电。调压器的输出电压 (V_{CORE}) 可通过软件编程为 1.2 - 1.8 V（典型值）内的三个不同的范围（请参见*第 6.1.5 节*）。

调压器在复位后始终处于使能状态。根据应用模式的不同，可采用三种不同的模式工作：主 (MR) 模式、低功耗 (LPR) 模式和掉电模式。

- 在运行模式下，调压器处于主 (MR) 模式并为 V_{CORE} 域（内核、存储器和数字外设）提供全功率。
- 在低功耗运行模式下，调压器处于低功耗 (LPR) 模式并为 V_{CORE} 域提供低功率，保留寄存器和内部 SRAM 中的内容。
- 在睡眠模式下，调压器处于主 (MR) 模式并为 V_{CORE} 域提供全功率，保留寄存器和内部 SRAM 中的内容。
- 在低功耗睡眠模式下，调压器处于低功耗 (LPR) 模式并为 V_{CORE} 域提供低功率，保留寄存器和内部 SRAM 中的内容。
- 在停止模式下，调压器为 V_{CORE} 域提供低功率，保留寄存器和内部 SRAM 中的内容。
- 在待机模式下，调压器掉电。除待机电路外，寄存器和 SRAM 的内容都将丢失。

6.1.5 动态电压调节管理

动态电压调节是一种功率管理技术，可根据情况增加或减少用于数字外设的电压 (V_{CORE})。

用于增加 V_{CORE} 的动态电压调节称为过电压。它可提高设备性能。有关器件工作状态与 CPU 性能关系的介绍，请参见*图 11*；有关 ADC 时钟频率与动态范围的关系，请参见数据手册的电气特性部分。

用于减少 V_{CORE} 的动态电压调节称为欠电压。执行欠电压调节可以节省电能，特别是在电能来自电池并因此受限的笔记本电脑和其他移动设备中。

范围 1

范围 1 是“高性能”范围。

只要 V_{DD} 输入电压超过 1.71 V，电压调节器就会输出 1.8 V 电压（典型值）。可在此范围内执行 Flash 编程和擦除操作。

仅当器件在范围 1 中工作时，时钟恢复系统 (CRS) 才可用（请参见*第 8 节：时钟恢复系统 (CRS)*）。

当 V_{DD} 低于 2.0 V 时，从初始到最终状态的 CPU 频率变化必须遵循以下条件：

- $f_{CPU\,最终} < 4xf_{CPU\,初始}$。
- 此外，两次变化间必须有 5 µs 的延迟。例如，要从 4.2 切换到 32 MHz，首先从 4.2 切换到 16 MHz，等待 5 µs，然后从 16 切换到 32 MHz。

范围 2 和 3

也可对调压器进行编程，以输出经调整的 1.5 V（典型值，范围 2）或 1.2 V（典型值，范围 3），而对 V_{DD}（1.65 到 3.6 V）没有任何限制。

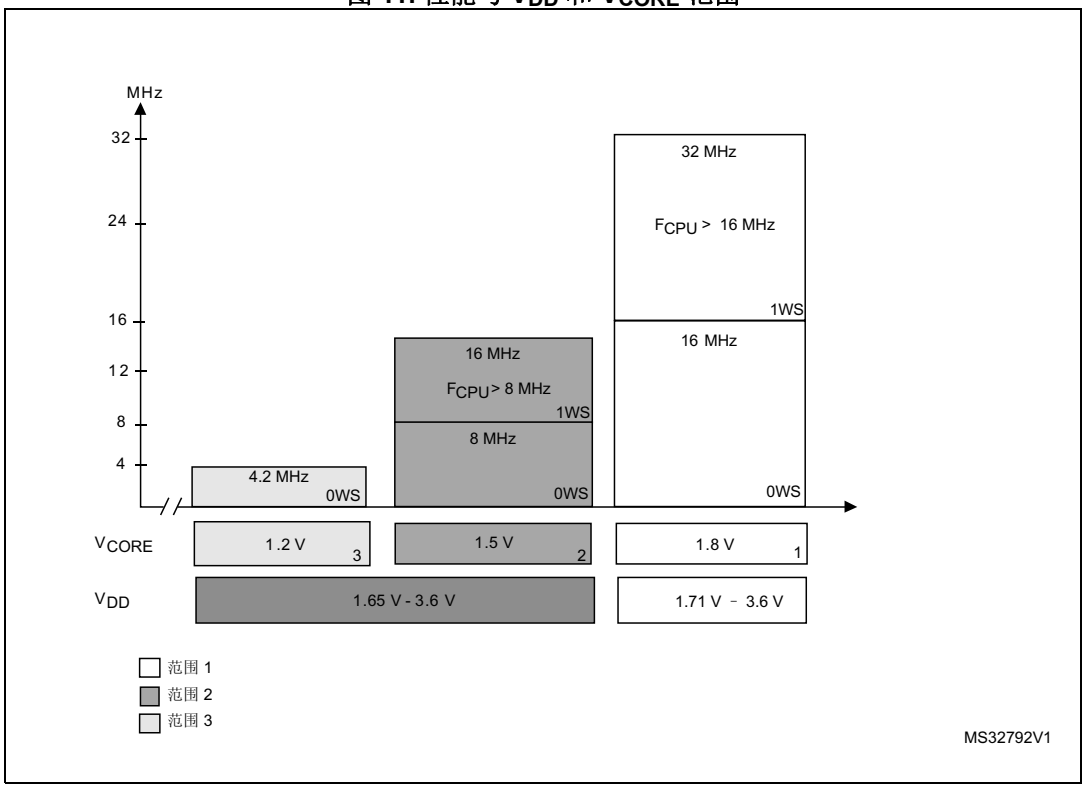
- 在 1.5 V 下，Flash 应可工作，但具有中等读取访问时间。这是“中等性能”范围。仍可对 Flash 进行编程和擦除操作。
- 在 1.2 V 下，Flash 应可工作，但具有缓慢读取访问时间。这是“低性能”范围。在这些条件下，不能对 Flash 进行编程和擦除操作。

有关各范围下性能的详细信息，请参见*表 30*。

表 30. 性能与 V_{CORE} 范围

CPU 性能	功耗性能	V_{CORE} 范围	典型值 (V)	最大频率 (MHz)		V_{DD} 范围
				1 WS	0 WS	
高	低	1	1.8	32	16	1.71 - 3.6
中	中	2	1.5	16	8	1.65 - 3.6
低	高	3	1.2	4.2	4.2	

图 11. 性能与 V_{DD} 和 V_{CORE} 范围



6.1.6 动态电压调节配置

要编程调压器范围，需要按照以下顺序操作：

1. 检查 V_{DD} 以确定允许哪些范围（请参见*图 11：性能与 VDD 和 VCORE 范围*）。
2. 轮询 PWR_CSR 中的 VOSF 位。等到该位复位为 0。
3. 通过设置 PWR_CR 寄存器中的 VOS[1:0] 位，配置电压调节范围。
4. 轮询 PWR_CSR 寄存器中的 VOSF 位。等到该位复位为 0。

注：在电压调节配置期间，系统时钟停止，直到调压器稳定 (VOSF=0)。如果需要中断的关键响应时间，则应用开发过程中必须考虑这一点，具体取决于使用的外设（定时器、通信、……）。

6.1.7 当 V_{DD} 降至 1.71 V 以下时的调压器和时钟管理

当选择 V_{CORE} 范围 1 并且 V_{DD} 降至 1.71 V 以下时，应用程序必须重新配置系统。

要重新配置系统，需要执行以下三步操作：

1. 检测 V_{DD} 电压是否低于 1.71 V：
   使用 PVD 监视 V_{DD} 电压，当电压低于所选电平时生成中断。要检测 1.71 V 电压限制，应用程序可以通过软件选择 PVD 阈值 2（典型值 2.26 V）。有关 PVD 的详细信息，请参见*第 6.2.3 节*。
2. 根据将在下一步选择的电压范围调整时钟频率：
   低于 1.71 V 时，系统时钟频率限制为 16 MHz（针对范围 2）和 4.2 MHz（针对范围 3）。

3. 选择所需电压范围：
   请注意，当 V_{DD} 低于 1.71 V 时，只能选择范围 2 或范围 3。

注：当选择 V_{CORE} 范围 2 或范围 3 并且 V_{DD} 降至 1.71 V 以下时，无需重新配置系统。

6.1.8 修改 V_{CORE} 范围时的调压器和时钟管理

当 V_{DD} 高于 1.71 V 时，可以选择 3 个电压范围中的任意一个。

- 当电压范围高于目标电压范围时（例如，从范围 1 到范围 2）：
  a) 根据将在下一步选择的更低电压范围调整时钟频率。
  b) 选择所需电压范围。
- 当电压范围低于目标电压范围时（例如，从范围 3 到范围 1）：
  a) 选择所需电压范围。
  b) 需要时调整时钟频率。

当 V_{DD} 低于 1.71 V 时，只能选择范围 2 和范围 3：

- 从范围 2 到范围 3
  a) 根据电压范围 3 调整时钟频率。
  b) 选择电压范围3。
- 从范围 3 到范围 2
  a) 选择电压范围2。
  b) 需要时调整时钟频率。

6.1.9 当 V_{DD} 介于 1.71 V 到 2.0 V 之间时的电压范围和限制

STM32L0x3 调压器基于针对超低功耗设计的架构。它不使用任何外部电容器。这种调压器对负载的快速变化很敏感。在这种情况下，输出电压会在短时间内降低。考虑到核心电压必须高于 1.65 V 以确保 32 MHz 运行，这种现象对极低的 V_{DD} 电压（如 1.71 V V_{DD} 最小值）至关重要。

要保证 V_{DD} =1.8 V±5%（具有 1 个等待周期）和 V_{CORE} 范围 1 时的 32 MHz 运行，必须对运行模式下的 CPU 频率进行管理以避免单触发中超过比率 4 的变化。两次变化间必须有 5 µs 的延迟。从低功耗模式唤醒时没有限制。

6.2 电源监控器

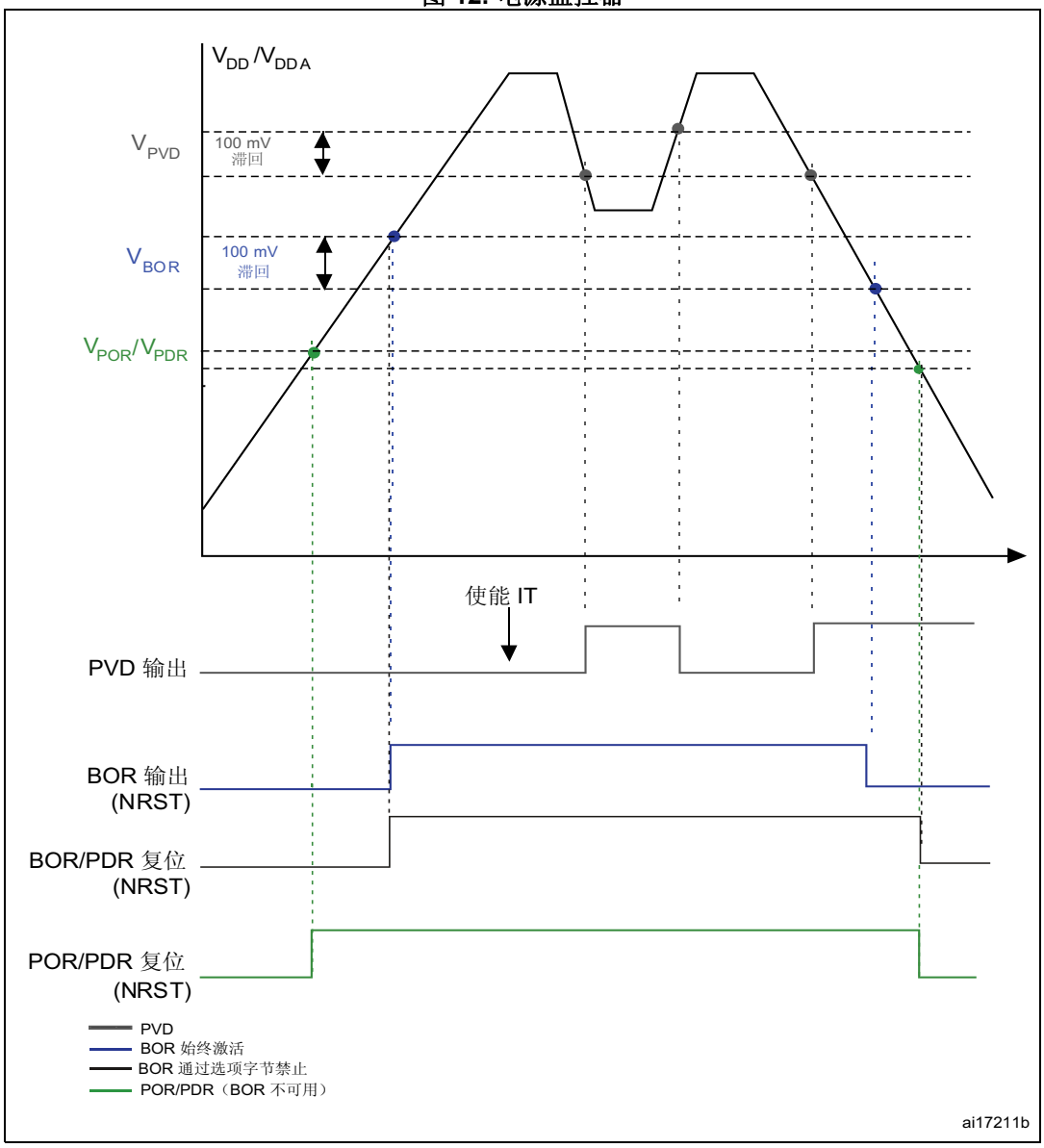
该器件具有一个集成的零功率上电复位 (POR)/掉电复位 (PDR) 电路，与欠压复位 (BOR) 电路耦合。对于在 1.8 和 3.6 V 之间工作的器件，BOR 在上电时始终激活，以确保从 1.8 V 开始正常工作。达到 1.8 V BOR 阈值之后，选项字节加载过程启动，以确认或修改默认阈值，或者永久禁止 BOR（在这种情况下，掉电时的 V_{DD} 最小值为 1.65 V）。对于在 1.65 V 和 3.6 V 之间工作的器件，永久禁止 BOR。因此，上电时的启动时间通常可以减少到 1 ms。

可以通过选项字节配置五个 BOR 阈值，从 1.65 到 3 V。要减少停止模式下的功耗，可以自动关闭内部参考电压 V_{REFINT}。当 V_{DD} 低于指定阈值 V_{POR}、V_{PDR} 或 V_{BOR} 时，器件无需任何外部复位电路便可保持复位模式。

该器件还有一个嵌入式可编程电压检测器 (PVD)，用于监视 V_{DD}/V_{DDA} 电源并将其与 V_{PVD} 阈值进行比较。可由软件选择 1.85 V 和 3.05 V 之间的 7 个不同的 PVD 电平，步长为 200 mV。当 V_{DD}/V_{DDA} 低于 V_{PVD} 阈值和/或 V_{DD}/V_{DDA} 高于 V_{PVD} 阈值时，将产生中断。随后，中断服务程序会生成一条警告消息并且/或者使 MCU 进入安全状态。PVD 由软件使能。

图 12 中说明了不同的电源监控器（POR、PDR、BOR、PVD）。

图 12. 电源监控器



1. 所有器件上都有 PVD，它可由软件使能或禁止。
2. BOR 仅在工作电压从 1.8 到 3.6 V 的器件上可用，除非被选项字节禁止，否则它会掩盖 POR/PDR 阈值。
3. 如果通过选项字节禁止 BOR，当 V_{DD} 低于 PDR 电平时，会发生复位。
4. 工作于1.65 到 3.6 V 的器件没有 BOR，当 V_{DD} 高于 POR 电平时，会释放复位，当 V_{DD} 低于 PDR 电平时，会发生复位。

6.2.1 上电复位 (POR)/掉电复位 (PDR)

本器件内部集成有 POR/PDR 电路，可以从 1.5 V 开始工作。

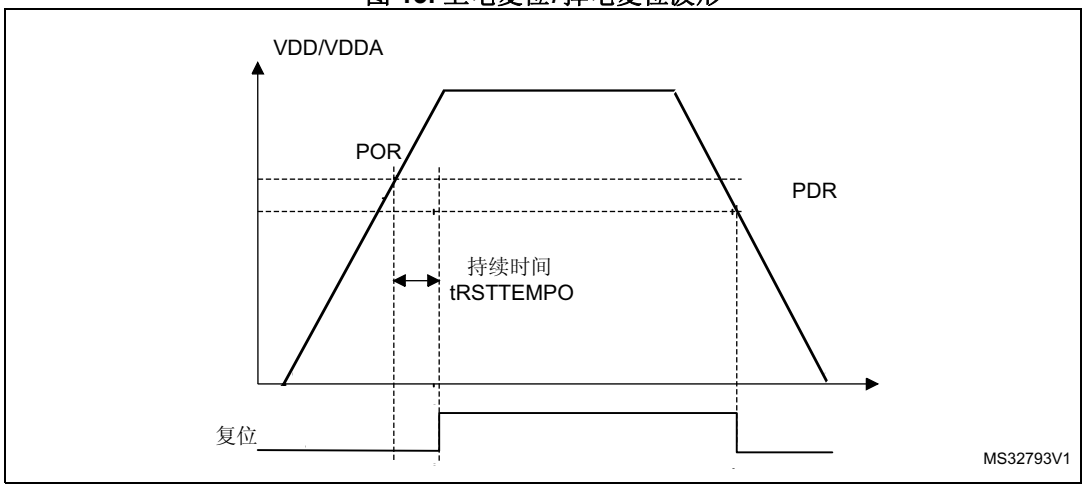
上电期间，当 V_{DD}/V_{DDA} 低于指定阈值 V_{POR} 时，器件无需外部复位电路便可保持复位模式。POR 功能始终使能，POR 阈值为 1.5 V。

掉电期间，当电源电压 (V_{DD}) 低于 V_{PDR} 阈值时，PDR 将使器件保持复位状态。PDR 功能始终使能，PDR 阈值为1.5 V。

只有在禁止 BOR 时才使用 POR 和 PDR（请参见*第 6.2.2 节：欠压复位 (BOR)*）。要确保最小工作电压 (1.65 V)，BOR 应配置为 BOR 级别 1。当禁止 BOR 时，最小工作电压 (1.65 V) 与 V_{POR}/V_{PDR} 阈值之间存在一个“盲区”。这意味着，V_{DD} 可低于 1.65 V 并且器件不会复位，直至达到 V_{PDR} 阈值。

有关上电/掉电复位阈值的相关详细信息，请参见数据手册的电气特性部分。

图 13. 上电复位/掉电复位波形



6.2.2 欠压复位 (BOR)

上电期间，欠压复位 (BOR) 将使器件保持复位状态，直到电源电压达到指定的 V_{BOR} 阈值。

对于工作于 1.65 到 3.6 V 间的器件，BOR 选项不可用，电源供电由 POR/PDR 监控。由于 POR/PDR 阈值为 1.5 V，因此在V_{POR}/V_{PDR} 阈值和最低产品工作电压 1.65 V 之间存在一个“盲区”。

对于工作在 1.8 到 3.6 V 之间的器件，BOR 在上电时始终激活，其阈值为 1.8 V。

随后，当释放系统复位时，可通过加载选项字节重新配置或禁止 BOR 级别。

若 BOR 电平在开机时保持在最低电平 1.8 V，在掉电时为 1.65 V，则 BOR 完全管理系统复位，产品工作电压处于安全范围内。

当通过选项字节禁止 BOR 选项时，掉电复位由 PDR 控制，在 1.65 V 和 V_{PDR} 之间存在“盲区”。

V_{BOR} 通过器件选项字节进行配置。默认情况下，激活 4 级阈值。可以选择 5 个可编程 V_{BOR} 阈值。

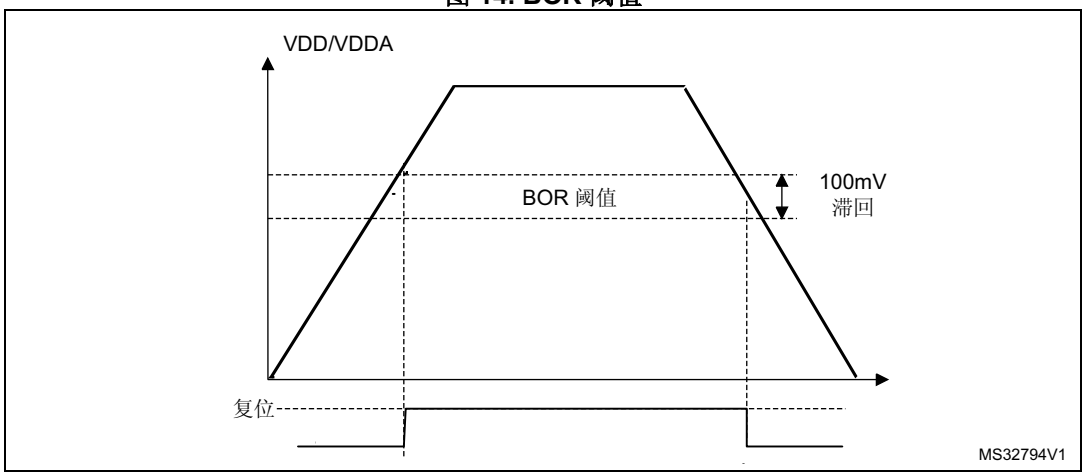
- BOR 级别 1 (V_{BOR0})：1.69 V 到 1.80 V 电压范围的复位阈值级别
- BOR 级别 2 (V_{BOR1})：1.94 V 到 2.1 V 电压范围的复位阈值级别
- BOR 级别 3 (V_{BOR2})：2.3 V 到 2.49 V 电压范围的复位阈值级别
- BOR 级别 4 (V_{BOR3})：2.54 V 到 2.74 V 电压范围的复位阈值级别
- BOR 级别 5 (V_{BOR4})：2.77 V 到 3.0 V 电压范围的复位阈值级别

当电源电压 (V_{DD}) 降至所选 V_{BOR} 阈值以下时，将使器件复位。当 V_{DD} 高于 V_{BOR} 上限时，释放器件复位，系统可以启动。

通过对器件选项字节进行编程可以禁止 BOR。要禁止 BOR 功能，V_{DD} 必须高于 V_{BOR0}，以启动器件选项字节编程序列。POR 和 PDR 将在随后监视上电和掉电（请参见*第 6.2.1 节：上电复位 (POR)/掉电复位 (PDR)*）。

BOR 阈值滞回电压约为 100 mV（电源电压的上升沿与下降沿之间）。

图 14. BOR 阈值



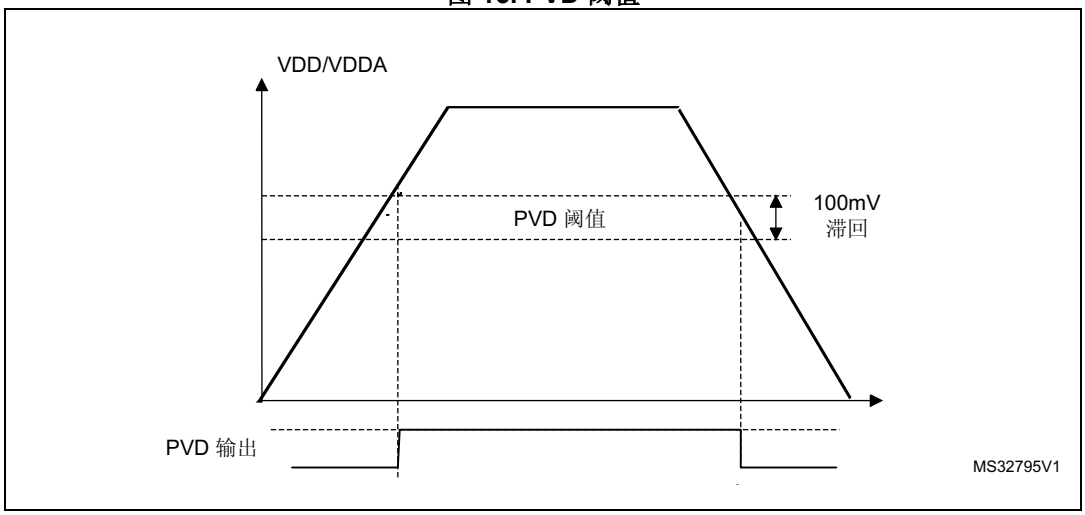
6.2.3 可编程电压检测器 (PVD)

可以使用 PVD 监视 V_{DD} 电源，方法是将其与 PWR_CR 中的 PLS[2:0] 位所选的阈值进行比较（请参见*第 6.4.1 节*）。

PVD 可以使用外部输入模拟电压 (PVD_IN)，该电压在内部与 VREFINT 进行比较。当 PLS[2:0] = 111 时，必须在模拟模式下配置 PVD_IN (PB7)。通过设置 PVDE 位来使能 PVD。

PWR_CSR 中提供了 PVDO 标志位（请参见*第 6.4.2 节*），用于指示 V_{DD} 是大于还是小于 PVD 阈值。该事件内部连接到 EXTI 线 16，如果通过 EXTI 寄存器使能，则可以产生中断。当 V_{DD} 降至 PVD 阈值以下以及/或者当 V_{DD} 升至 PVD 阈值以上时，可以产生 PVD 输出中断，具体取决于 EXTI 线 16 上升沿/下降沿的配置。该功能的用处之一就是可以在中断服务程序中执行紧急关闭系统的任务。

图 15. PVD 阈值



6.2.4 内部参考电压 (V_{REFINT})

内部参考电压 (V_{REFINT}) 为模拟外设提供稳定的电压。通过内部参考电压 (V_{REFINT}) 管理的功能包括 BOR、PVD、ADC、HSI48、LCD 和比较器。当使用这些功能之一时，始终使能内部参考电压 (V_{REFINT})。

内部参考电压消耗不可忽略，尤其是在停止和待机模式下。为降低功耗，可以设置 PWR_CR 寄存器中的 ULP 位（超低功耗）以禁止内部参考电压。不过，在这种情况下，当退出停止/待机模式时，在内部参考电压启动期间（最长 3 ms）通过内部参考电压管理的功能不可靠。

要减少唤醒时间，器件可以退出停止/待机模式，而无需等待内部参考电压启动时间。为此，在进入停止/待机模式之前，在 PWR_CR 寄存器中设置 FWU 位（快速唤醒）。

如果 ULP 位置 1，则进入停止/待机模式之前使能的功能在这些模式下将被禁止，仅当内部参考电压启动时间结束后才重新使能，无论 FWU 值如何。PWR_CSR 寄存器中的 VREFINTRDYF 标志指示内部参考电压已就绪。

当器件在 NRST 脉冲上退出低功耗模式时，它不会等待内部参考电压启动（即使 ULP = 1 且 FWU=0）。必要时，应用程序应检查 VREFINTRDYF 标志。

注：当 LCD 激活（LCD_CR 的 LCDEN 位置 1）时，需要 VREFINT。因此 ULP 位必须复位。

6.3 低功耗模式

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

器件有五个低功耗模式：

- 低功耗运行模式：调压器处于低功耗模式，时钟频率受限，运行的外设数受限（请参见*第 6.3.4 节*）
- 睡眠模式：Cortex®-M0+ 内核停止，外设保持运行（请参见*第 6.3.7 节*）
- 低功耗睡眠模式：Cortex®-M0+ 内核停止，时钟频率受限，运行的外设数受限，调压器处于低功耗模式，Flash 停止（请参见*第 6.3.8 节*）
- 停止模式：所有时钟停止，调压器运行，调压器处于低功耗模式（请参见*第 6.3.9 节*）
- 待机模式：V_{CORE} 域断电（请参见*第 6.3.10 节*）

此外，可通过下列方法之一降低运行模式下的功耗：

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

表 31. 低功耗模式汇总

<table>
<tr><th>模式名称</th><th>进入</th><th>唤醒</th><th>对 V_{CORE} 域时钟的影响</th><th>对 V_{DD} 域时钟的影响</th><th>调压器</th></tr>
<tr><td>低功耗运行</td><td>LPSDSR 和 LPRUN 位 + 时钟设置</td><td>强制调压器处于主调压器模式 (1.8 V)</td><td>无</td><td>无</td><td>处于低功耗模式</td></tr>
<tr><td rowspan="2">睡眠（立即睡眠或退出时睡眠）</td><td>WFI 或从 ISR 返回</td><td>任意中断</td><td rowspan="2">CPU CLK 关闭对其它时钟或模拟时钟源无影响</td><td rowspan="2">无</td><td rowspan="2">启动</td></tr>
<tr><td>WFE</td><td>唤醒事件</td></tr>
<tr><td rowspan="2">低功耗睡眠（立即睡眠或退出时睡眠）</td><td>LPSDSR 位 + WFI 或从 ISR 返回</td><td>任意中断</td><td rowspan="2">CPU CLK 关闭对其它时钟或模拟时钟源无影响，Flash CLK 关闭</td><td rowspan="2">无</td><td rowspan="2">处于低功耗模式</td></tr>
<tr><td>LPSDSR 位 + WFE</td><td>唤醒事件</td></tr>
<tr><td>停止</td><td>PDDS、LPSDSR 位 + SLEEPDEEP 位 + WFI，从 ISR 返回或 WFE</td><td>任意 EXTI 线（在 EXTI 寄存器中配置，内部线和外部线）</td><td rowspan="2">所有 V_{CORE} 域时钟都关闭</td><td rowspan="2">HSI16(1)、HSE 和 MSI 振荡器关闭</td><td>在低功耗模式下打开（取决于 PWR_CR）</td></tr>
<tr><td>待机</td><td>PDDS 位 + SLEEPDEEP 位 + WFI，从 ISR 返回或 WFE</td><td>WKUP 引脚上升沿、RTC 闹钟（闹钟 A 或闹钟 B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位、IWDG 复位</td><td>关闭</td></tr>
</table>

1. 如果在*时钟控制寄存器 (RCC_CR)* 中设置 HSI16KERON，HSI16 可在停止模式下运行。

6.3.1 低功耗模式下时钟的行为

可通过软件禁止 APB 外设和 DMA 时钟。

睡眠和低功耗睡眠模式

在睡眠和低功耗睡眠模式下 CPU 时钟停止。睡眠期间可通过软件停止存储器接口（Flash 和 RAM 接口）时钟和所有外设时钟。在低功耗睡眠模式下，存储器接口时钟停止，RAM 处于掉电状态。当连接到 AHB-APB 总线桥时钟的所有外设时钟均被禁止时，睡眠/低功耗睡眠模式期间将通过硬件禁止 AHB-APB 总线桥时钟。

停止和待机模式

在停止和待机模式下系统时钟和所有高速时钟都停止：

- 禁止 PLL
- 禁止内部 RC 16 MHz (HSI16) 振荡器
- 禁止外部 1-24 MHz (HSE) 振荡器
- 禁止内部 65 kHz - 4.2 MHz (MSI) 振荡器

通过中断退出此模式（停止模式）时，可以选择内部 MSI 或 HSI16 作为系统时钟。对于这两种振荡器，退出停止模式时保持其相应配置（范围和微调）。

通过复位退出此模式（待机模式）时，可以选择 MSI 振荡器作为系统时钟。范围和微调值复位为默认值 2.1 MHz。

如果正在执行 Flash 编程操作或正在访问 APB 域，停止/待机模式的进入将延迟到 Flash 或 APB 访问完成后执行。

6.3.2 降低系统时钟速度

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（SYSCLK、HCLK、PCLK1 和 PCLK2）速度。进入睡眠模式之前，也可以使用这些预分频器降低外设速度。

有关详细信息，请参见*第 7.3.4 节：时钟配置寄存器 (RCC_CFGR)*。

6.3.3 外设时钟门控

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

要进一步降低睡眠模式的功耗，可在执行 WFI 或 WFE 指令之前禁止外设时钟。

外设时钟门控由 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) 进行控制（参见*第 7.3.13 节：AHB 外设时钟使能寄存器 (RCC_AHBENR)*、*第 7.3.15 节：APB1 外设时钟使能寄存器 (RCC_APB1ENR)* 和*第 7.3.14 节：APB2 外设时钟使能寄存器 (RCC_APB2ENR)*）。

在睡眠模式下，复位 RCC_AHBLPENR 和 RCC_APBxLPENR 寄存器（x 可以是 1 或 2）中的对应位可以自动禁止外设时钟。

6.3.4 低功耗运行模式（LP 运行）

当系统处于停止模式时，要进一步降低功耗，可以在低功耗模式下配置调压器。在此模式下，系统频率不应超过 f_MSI 范围 1。

有关调压器和外设工作条件的详细信息，请参见产品数据手册。

*注：为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍 (7*RTCLCK) 时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。*

这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。

仅当 V_{CORE} 处于范围 2 时，才能进入低功耗运行模式。此外，选择低功耗运行模式时，不能使用动态电压调节。选择低功耗运行模式时，只允许停止和睡眠模式以及在低功耗模式下配置的调压器。

注：在低功耗运行模式下，所有 I/O 引脚的状态与运行模式下相同。

进入低功耗运行模式

要进入低功耗运行模式，请执行下列操作：

1. 必须使用 RCC_APBxENR 和 RCC_AHBENR 寄存器使能或禁止每个数字 IP 时钟。
2. 系统时钟的频率必须降低到不超过 f_MSI 范围 1 的频率。
3. 通过软件强制调压器处于低功耗模式（LPRUN 和 LPSDSR 位置 1）。

退出低功耗运行模式

要退出低功耗运行模式，请执行下列操作：

1. 通过软件强制调压器处于主调压器模式。
2. 如果需要，打开 Flash。
3. 可以增加时钟系统的频率。

6.3.5 进入低功耗模式

通过执行 WFI（等待中断）或 WFE（等待事件）指令，或者从 ISR 返回时 Cortex®-M0+ 系统时钟寄存器中的 SLEEPONEXIT 位置 1 时，即可进入低功耗模式（低功耗运行模式除外）。

仅当没有中断和事件挂起时，才会执行通过 WFI 或 WFE 进入低功耗模式。

6.3.6 退出低功耗模式

微控制器根据模式的进入方式退出睡眠和停止模式：

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

    通过以下操作实现：在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能 Cortex®-M0+ 系统控制寄存器中的 SEVONPEND 位。当微控制器从 WFE 恢复时，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）。
  - 事件：

    通过以下操作实现：配置一个外部或内部 EXTI 线为事件模式。当 CPU 从 WFE 恢复时，因为对应事件线的挂起位没有被置 1，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

6.3.7 睡眠模式

睡眠模式下的 I/O 状态

在睡眠模式下，所有 I/O 引脚的状态与运行模式下相同。

进入睡眠模式

根据*第 6.3.5 节：进入低功耗模式*进入睡眠模式。

有关如何进入睡眠模式的详细信息，请参见*表 32：立即睡眠*和*表 33：退出时睡眠*。

退出睡眠模式

根据*第 6.3.6 节：退出低功耗模式*退出睡眠模式。

有关如何退出睡眠模式的详细信息，请参见*表 32：立即睡眠*和*表 33：退出时睡眠*。

表 32. 立即睡眠

立即睡眠模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – SLEEPDEEP = 0 及 – 没有中断（对于 WFI）或事件（对于 WFE）挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
	从 ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
退出模式	如果使用 WFI 或从 ISR 返回进入： 中断：请参见*表 51：向量列表* 如果使用 WFE 进入且 SVONPEND = 0 唤醒事件：请参见*第 13.3.2 节：唤醒事件管理* 如果使用 WFE 进入且 SVONPEND = 1 中断事件（当在 NVIC 中禁止时，请参见*表 51：向量列表*）或唤醒事件（请参见*第 13.3.2 节：唤醒事件管理*）
唤醒延迟	无

表 33. 退出时睡眠

退出时睡眠	说明
进入模式	WFI（等待中断），且： – SLEEPDEEP = 0 及 – 没有中断（对于 WFI）或事件（对于 WFE）挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
	从 ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
退出模式	中断：请参见*表 51：向量列表*
唤醒延迟	无

6.3.8 低功耗睡眠模式（LP 睡眠）

低功耗睡眠模式下的 I/O 状态

在低功耗睡眠模式下，所有 I/O 引脚的状态与运行模式下相同。

进入低功耗睡眠模式

要进入低功耗睡眠模式，请执行下列操作：

1. 可以使用 FLASH_ACR 寄存器中的控制位 SLEEP_PD 关闭 Flash。这会降低功耗，但增加了唤醒时间。
2. 必须使用 RCC_APBxENR 和 RCC_AHBENR 寄存器使能或禁止每个数字 IP 时钟。
3. 必须降低系统时钟的频率。
4. 通过软件强制调压器处于低功耗模式（LPSDSR 位置 1）。
5. 执行*第 6.3.5 节：进入低功耗模式*中所述的步骤。

有关如何进入低功耗睡眠模式的详细信息，请参见*表 34：立即睡眠（低功耗睡眠）*和*表 35：退出时睡眠（低功耗睡眠）*。

在低功耗睡眠模式下，可以关闭 Flash，RAM 存储器保持可用。

在此模式下，系统频率不应超过 f_MSI 范围 1。

有关调压器和外设操作条件的详细信息，请参见产品数据手册。

仅当 V_{CORE} 处于范围 2 时，才能进入低功耗睡眠模式。

*注：为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍 (7*RTCLCK) 时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。*

这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。

退出低功耗睡眠模式

根据*第 6.3.6 节：退出低功耗模式*退出低功耗睡眠模式。

通过发出中断或唤醒事件退出低功耗睡眠模式时，在主调压器模式下配置调压器，Flash 关闭（如有必要），并且可以增加系统时钟。

当调压器在低功耗模式下工作时，将器件从低功耗睡眠模式唤醒将需要额外的启动延时。

有关如何退出低功耗睡眠模式的详细信息，请参见*表 34：立即睡眠（低功耗睡眠）*和*表 35：退出时睡眠（低功耗睡眠）*。

表 34. 立即睡眠（低功耗睡眠）

立即睡眠模式	说明
进入模式	调压器处于低功耗模式且 Flash 关闭 WFI（等待中断）或 WFE（等待事件），且： – SLEEPDEEP = 0 及 – 没有中断（对于 WFI）或事件（对于 WFE）挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
	从 ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
退出模式	调压器处于主调压器模式且 Flash 打开 如果使用 WFI 或从 ISR 返回进入： 中断：请参见*表 51：向量列表* 如果使用 WFE 进入且 SEVONPEND = 0 唤醒事件：请参见*第 13.3.2 节：唤醒事件管理* 如果使用 WFE 进入且 SVONPEND = 1 中断事件（当在 NVIC 中禁止时，请参见*表 51：向量列表）*或唤醒事件（请参见*第 13.3.2 节：唤醒事件管理*）
唤醒延迟	调压器从低功耗模式唤醒的时间

表 35. 退出时睡眠（低功耗睡眠）

退出时睡眠	说明
进入模式	WFI（等待中断），且： – SLEEPDEEP = 0 及 – 没有中断（对于 WFI）或事件（对于 WFE）挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
	从 ISR 返回，且： – SLEEPDEEP = 0 及 – SLEEPONEXIT = 1 及 – 没有中断挂起 请参见 Cortex®-M0+ 系统控制寄存器（请参见 PM0223 编程手册）。
退出模式	中断：请参见*表 51：向量列表*。
唤醒延迟	调压器从低功耗模式唤醒的时间

6.3.9 停止模式

停止模式基于 Cortex®-M0+ 深度睡眠模式与外设时钟门控。调压器既可以配置为正常模式，也可以配置为低功耗模式。在停止模式下，V_{CORE} 域中的所有时钟都会停止，PLL、MSI、HSI16 和 HSE RC 振荡器也被禁止。内部 SRAM 和寄存器内容将保留。

要使停止模式下的功耗最低，内部 Flash 也进入低功耗模式。Flash 处于掉电模式时，将器件从停止模式唤醒将需要额外的启动延时。

要使停止模式下的功耗最低，可在进入停止模式前关闭 V_{REFINT}、BOR、PVD 和温度传感器。退出停止模式后，可以使用 PWR_CR 寄存器中的 ULP 位通过软件重新打开它们。

低功耗睡眠模式下的 I/O 状态

在停止模式下，所有 I/O 引脚的状态与运行模式下相同。

进入停止模式

有关如何进入停止模式的详细信息，请参见*第 6.3.5 节：进入低功耗模式*和*表 36*。

如果应用程序需要在进入停止模式之前禁止外部时钟，必须首先禁止 HSEON 位并将系统时钟切换到 HSI16。

否则，如果在进入停止模式前 HSEON 位保持使能并且可以移除外部时钟（外部振荡器），则必须使能时钟安全系统 (CSS) 功能以检测任何外部振荡器故障并避免进入停止模式时出现故障行为。

要进一步降低停止模式的功耗，可将内部调压器设置为低功耗模式。通过 PWR_CR 寄存器中的 LPSDSR 位进行配置（请参见*第 6.4.1 节*）。

如果正在执行 Flash 编程操作或正在访问 APB 域，停止模式的进入将延迟到 Flash 或 APB 访问完成后执行。

在停止模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第 25 节：独立看门狗 (IWDG)* 中的*第 25.3 节：IWDG 功能说明*。
- 实时时钟 (RTC)：通过 RCC_CSR 寄存器中的 RTCEN 位进行配置（请参见*第 7.3.21 节*）。
- 内部 RC 振荡器 (LSI RC)：通过 RCC_CSR 寄存器中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC)：通过 RCC_CSR 寄存器中的 LSEON 位进行配置。

在停止模式下，ADC、DAC 或 LCD 也会产生功耗，除非在进入停止模式前将其禁止。要禁止这些转换器，必须将 ADC_CR2 寄存器中的 ADON 位和 DAC_CR 寄存器中的 ENx 位都清零。

退出停止模式

有关如何退出停止模式的详细信息，请参见*第 6.3.6 节：退出低功耗模式*和*表 36*。

通过发出中断或唤醒事件退出停止模式时，将根据 RCC_CFGR 寄存器中的 STOPWUCK 位选择 MSI 或 HSI16 RC 振荡器作为系统时钟。

当调压器在低功耗模式下工作时，将器件从停止模式唤醒将需要额外的延时。在停止模式下一直开启内部调压器虽然可以缩短启动时间，但功耗却增大。

表 36. 停止模式

停止模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – 没有中断（对于 WFI）或事件（对于 WFE）挂起。 – 将 Cortex®-M0+ 系统控制寄存器中的 SLEEPDEEP 位置 1 – 电源控制寄存器 (PWR_CR) 中的 PDDS 位 = 0 – 电源控制/状态寄存器 (PWR_CSR) 中的 WUF 位 = 0 – 通过配置 RCC_CFGR 寄存器中的 STOPWUCK 位退出停止模式时，选择 MSI 或 HSI16 RC 振荡器作为系统时钟。 *注：要进入停止模式，所有 EXTI 线挂起位（在第 13.5.6 节：EXTI 挂起寄存器 (EXTI_PR) 中）、所有外设中断挂起位、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒、RTC 入侵和 RTC 时间戳标志位必须复位。否则将忽略进入停止模式这一过程，继续执行程序。*
	从 ISR 返回，且： – 没有中断挂起。 – 将 Cortex®-M0+ 系统控制寄存器中的 SLEEPDEEP 位置 1 – SLEEPONEXIT = 1 – 电源控制寄存器 (PWR_CR) 中的 PDDS 位 = 0 – 电源控制/状态寄存器 (PWR_CSR) 中的 WUF 位 = 0 – 通过配置 RCC_CFGR 寄存器中的 STOPWUCK 位退出停止模式时，选择 MSI 或 HSI16 RC 振荡器作为系统时钟。 *注：要进入停止模式，所有 EXTI 线挂起位（在第 13.5.6 节：EXTI 挂起寄存器 (EXTI_PR) 中）、所有外设中断挂起位、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒、RTC 入侵和 RTC 时间戳标志位必须复位。否则将忽略进入停止模式这一过程，继续执行程序。*
退出模式	如果使用 WFI 或从 ISR 返回进入： 任何配置为中断模式的 EXTI 线（必须在 NVIC 中使能对应的 EXTI 中断向量）。请参见*表 51：向量列表。* 如果使用 WFE 进入且 SEVONPEND = 0 任何配置为事件模式的 EXTI 线。请参见*第 267 页上的第 13.3.2 节：唤醒事件管理。* 如果使用 WFE 进入且 SEVONPEND = 1 – 任何配置为事件模式的 EXTI 线（即使在 NVIC 中禁止对应的 EXTI 中断）。中断源可以是外部中断或具有唤醒功能的外设（请参见*表 51：向量列表）。* – 唤醒事件（请参见*第 267 页上的第 13.3.2 节：唤醒事件管理*）
唤醒延迟	MSI 或 HSI16 RC 唤醒时间 + 调压器从低功耗模式唤醒的时间 + FLASH 唤醒时间

6.3.10 待机模式

待机模式下可达到最低功耗。待机模式基于 Cortex®-M0+ 深度睡眠模式，其中调压器被禁止。因此 V_{CORE} 域断电。PLL、MSI、HSI16 振荡器和 HSE 振荡器也将关闭。除 RTC 寄存器、RTC 备份寄存器和待机电路外，SRAM 和寄存器内容都将丢失（参见*图 10*）。

待机模式下的 I/O 状态

在待机模式下，除以下各部分以外，所有 I/O 引脚都处于高阻态：

- 复位引脚
- 唤醒引脚（WKUP1、WKUP2、WKUP3）
- 以下 I/O 上的 RTC 功能（入侵、时间戳、RTC 闹钟输出、RTC 时钟校准输出）：
  - 类别 3：PC13、PA0
  - 类别 5：PC13、PA0、PE6

进入待机模式

有关如何进入待机模式的详细信息，请参见*第 6.3.5 节：进入低功耗模式*和*表 37*。

在待机模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)：IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。请参见*第 580 页上的第 25.3 节：IWDG 功能说明*。
- 实时时钟 (RTC)：通过 RCC_CSR 寄存器中的 RTCEN 位进行配置（请参见*第 7.3.21 节*）。
- 内部 RC 振荡器 (LSI RC)：通过 RCC_CSR 寄存器中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC)：通过 RCC_CSR 寄存器中的 LSEON 位进行配置。

退出待机模式

检测到外部复位（NRST 引脚）、IWDG 复位、WKUP 引脚（WUKP1、WKUP2 或 WKUP3）上升沿、RTC 闹钟、入侵事件或时间戳事件时，微控制器退出待机模式。从待机模式唤醒后，除 *PWR 电源控制/状态寄存器 (PWR_CSR)* 外，所有寄存器都将复位。

从待机模式唤醒后，程序将按照复位（启动引脚采样、复位向量已获取等）后的方式重新执行。PWR_CSR 寄存器（请参见*第 6.4.2 节*）中的 SBF 状态标志指示 MCU 已处于待机模式。

有关如何退出待机模式的详细信息，请参见*第 6.3.6 节：退出低功耗模式*和*表 37*。

表 37. 待机模式

待机模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： – Cortex®-M0+ 系统控制寄存器中的 SLEEPDEEP 位 = 1 – 电源控制寄存器 (PWR_CR) 中的 PDDS 位 = 1 – 没有中断（对于 WFI）或事件（对于 WFE）挂起。 – 电源控制/状态寄存器 (PWR_CSR) 中的 WUF 位 = 0 – 将与所选唤醒源（RTC 闹钟 A、RTC 闹钟 B、RTC 唤醒、入侵或时间戳标志）对应的 RTC 标志清零
	从 ISR 返回，且： – Cortex®-M0+ 系统控制寄存器中的 SLEEPDEEP 位 = 1 – SLEEPONEXIT = 1 – 电源控制寄存器 (PWR_CR) 中的 PDDS 位 = 1 – 没有中断挂起 – 电源控制/状态寄存器 (PWR_CSR) 中的 WUF 位 = 0 – 将与所选唤醒源（RTC 闹钟 A、RTC 闹钟 B、RTC 唤醒、入侵或时间戳标志）对应的 RTC 标志清零
退出模式	WKUP 引脚上升沿、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 唤醒事件、RTC 入侵事件、RTC 时间戳事件、NRST 引脚外部复位 和 IWDG 复位。
唤醒延迟	复位阶段

调试模式

默认情况下，如果使用调试功能时应用程序将 MCU 置于停止模式或待机模式，调试连接将中断。这是因为 Cortex®-M0+ 内核时钟停止了。

不过，通过设置 DBG_CR 寄存器中的一些配置位，即使 MCU 进入低功耗模式，仍可使用软件对其进行调试。有关详细信息，请参见*第 33.9.1 节：对低功耗模式的调试支持*。

6.3.11 使用 RTC 和比较器从停止和待机模式唤醒器件

可以通过 RTC 闹钟事件、RTC 唤醒事件、入侵事件、时间戳事件或比较器事件将 MCU 从低功耗模式唤醒，而不依赖于外部中断（自动唤醒模式）。

这些 RTC 复用功能可将系统从停止和待机低功耗模式唤醒，而比较器事件只能将系统从停止模式唤醒。

通过使用 RTC 闹钟或 RTC 唤醒事件，无需依赖外部中断即可将系统从低功耗模式唤醒（自动唤醒模式）。

RTC 提供了可编程时基，便于定期从停止或待机模式唤醒器件。为此，通过对 RCC_CSR 寄存器（请参见*第 7.3.21 节*）中的 RTCSEL[1:0] 位进行编程，可以选择三个复用 RTC 时钟源中的其中两个：

- 低功耗 32.768 kHz 外部晶振 (LSE OSC)
  此时钟源提供的时基非常精确，功耗也非常低（典型条件下附加功耗小于 1 μA ）
- 低功耗内部 RC 振荡器 (LSI RC)
  此时钟源的优势在于可以节省 32.768 kHz 晶振的成本。此内部 RC 振荡器非常省电。

从停止模式的 RTC 自动唤醒 (AWU)

- 要通过 RTC 闹钟事件从停止模式唤醒，必须：
  a) 将 EXTI 线 17 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 闹钟中断
  c) 配置 RTC 以生成 RTC 闹钟
- 要通过 RTC 入侵事件或时间戳事件从停止模式唤醒，必须：
  a) 将 EXTI 线 19 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 时间戳中断，或者使能 RTC_TCR 寄存器中的 RTC 入侵中断
  c) 配置 RTC 以检测入侵事件或时间戳事件
- 要通过 RTC 唤醒事件从停止模式唤醒，必须：
  a) 将 EXTI 线 20 配置为检测外部信号的上升沿（中断或事件模式）
  b) 使能 RTC_CR 寄存器中的 RTC 唤醒中断
  c) 配置 RTC 以生成 RTC 唤醒事件

从待机模式的 RTC 自动唤醒 (AWU)

- 要通过 RTC 闹钟事件从待机模式唤醒，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 闹钟中断
  b) 配置 RTC 以生成 RTC 闹钟
- 要通过 RTC 入侵事件或时间戳事件从停止模式唤醒，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 时间戳中断，或者使能 RTC_TCR 寄存器中的 RTC 入侵中断
  b) 配置 RTC 以检测入侵事件或时间戳事件
- 要通过 RTC 唤醒事件从停止模式唤醒，必须：
  a) 使能 RTC_CR 寄存器中的 RTC 唤醒中断
  b) 配置 RTC 以生成 RTC 唤醒事件

从停止模式的比较器自动唤醒 (AWU)

- 要通过比较器 1 或比较器 2 唤醒事件从停止模式唤醒，必须：
  a) 将 EXTI21（硬连线到比较器 1）和 EXTI22（硬连线到比较器 2）（中断或事件模式）配置为检测选定沿（下降、上升或下降和上升）
  b) 配置比较器以生成事件

6.4 电源控制寄存器

必须按半字（16 位）或字（32 位）访问外设寄存器。

6.4.1 PWR 电源控制寄存器 (PWR_CR)

PWR power control register

偏移地址：0x00

复位值：0x0000 1000（通过从待机模式唤醒进行复位）

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPRUN	DS_EE_KOFF	VOS[1:0]		FWU	ULP	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPSDSR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

位 31:15 保留，始终读为 0。

位 14 **LPRUN**：低功耗运行模式 (Low-power run mode)

当 LPRUN 位与 LPSDSR 位一起置 1 时，调压器从主模式切换到低功耗模式。否则，它将保持在主模式。当 LPRUN 复位时，调压器返回到在主模式下工作。

如果该位置 1（LPSDSR 位也置 1）且 CPU 进入睡眠或深度睡眠模式（LP 睡眠或停止模式），则当 CPU 从这些模式唤醒时，它将进入运行模式，但 LPRUN 位会置 1。要重新进入低功耗运行模式，必须执行复位并将 LPRUN 位重新置 1。

当 MCU 处于低功耗运行模式时，禁止复位 LPS DSR。LPSDSR 用作进入低功率模式的前置条件，向系统指示进入低功率模式时将选择调压器的哪个配置。在 LPRUN 位置 1 前 LPSDSR 位必须置 1。仅当 LPRUN 位 = 0 时才能复位 LPSDSR。

0：低功耗运行模式下调压器进入主模式

1：低功耗运行模式下调压器进入低功耗模式

位 13 **DS_EE_KOFF**：深度睡眠模式，非易失性存储器保持关闭 (Deepsleep mode with non-volatile memory kept off)

当进入低功耗模式（仅停止或待机）时，如果 FLASH_ACR 寄存器（请参见*第 3.7.1 节：访问控制寄存器 (FLASH_ACR)*）中的 DS_EE_KOFF 和 RUN_PD 位均置 1，则从深度睡眠模式退出时，不会唤醒非易失性存储器（Flash 程序存储器和数据 EEPROM）。

0：从深度睡眠模式退出时唤醒 NVM，即使 RUN_PD 位已置 1

1：从深度睡眠模式退出时不唤醒 NVM（如果 RUN_PD 位置 1）

位 12:11 **VOS[1:0]**：选择电压调节范围 (Voltage scaling range selection)

这些位用于选择内部调压器电压范围。

在通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位复位电源接口之前，这些位必须设置为 ‘10’ 并且必须相应地配置系统频率。

00：禁止（位保持不变，并保留以前的值，没有发生电压变化）

01：1.8 V（范围 1）

10：1.5 V（范围 2）

11：1.2 V（范围 3）

位 10 **FWU**：快速唤醒 (Fast wakeup)

此位与 ULP 位结合使用。

如果 ULP = 0，则忽略 FWU

如果 ULP = 1 且 FWU = 1：从低功耗模式退出时忽略 V_{REFINT} 启动时间。当 V_{REFINT} 重新就绪时，通过 PWR_CSR 寄存器中的 VREFINTRDYF 标志加以指示。

如果 ULP = 1 且 FWU = 0：仅当 V_{REFINT} 就绪（其启动时间后）时，才会从低功耗模式退出。通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位不会复位该位。

0：仅当 V_{REFINT} 就绪时，才会退出低功耗模式

1：退出低功耗模式时忽略 V_{REFINT} 启动时间

位 9 **ULP**：超低功耗模式 (Ultra-low-power mode)

置 1 时，V_{REFINT} 在低功耗模式下关闭。通过复位 RCC_APB1RSTR 寄存器中的 PWRRST 位不会复位该位。当该位置 1 时，寄存器 LCD_CR 的 LCDEN 位不能置 1。

0：V_{REFINT} 在低功耗模式下打开

1：V_{REFINT} 在低功耗模式下关闭

位 8 **DBP**：禁止备份写保护 (Disable backup write protection)

在复位状态下，RTC、RTC 备份寄存器和 RCC CSR 寄存器均受到保护，以防止寄生写访问。必须将此位置 1 才能使能对这些寄存器的写访问。

0：禁止访问 RTC、RTC 备份寄存器和 RCC CSR 寄存器

1：允许访问 RTC、RTC 备份寄存器和 RCC CSR 寄存器

注：　如果被 2、4、8 或 16 分频的 HSE 用作 RTC 时钟，该位必须保持置 1。

位 7:5 **PLS[2:0]**：PVD 电平选择 (PVD level selection)

这些位由软件写入，用于选择电源电压检测器检测的电压阈值：

000：1.9 V

001：2.1 V

010：2.3 V

011：2.5 V

100：2.7 V

101：2.9 V

110：3.1 V

111：外部输入模拟电压（内部与 V_{REFINT} 进行比较）

当 PLS[2:0] = 111 时，PVD_IN 输入 (PB7) 必须配置为模拟输入。

注：　有关详细信息，请参见数据手册的电气特性。

位 4 **PVDE**：使能电源电压检测器 (Power voltage detector enable)

此位由软件置 1 和清零。

0：禁止 PVD

1：使能 PVD

位 3 **CSBF**：将待机标志清零 (Clear standby flag)

此位始终读为 0。

0：不起作用

1：将 SBF 待机标志位清零（写）。

位 2 **CWUF**：将唤醒标志清零 (Clear wakeup flag)

此位始终读为 0。

0：不起作用

1：2 个系统时钟周期后将 WUF 唤醒标志清零

位 1 **PDDS**：掉电深度睡眠 (Power-down deepsleep)

此位由软件置 1 和清零。

0：器件在 CPU 进入深度睡眠时进入停止模式。调压器处于低功耗模式。

1：器件在 CPU 进入深度睡眠时进入待机模式。

位 0 **LPSDSR**：低功耗深度睡眠/睡眠/低功率运行 (Low-power deepsleep/Sleep/Low-power run)

– DeepSleep/Sleep 模式

如果该位置 1，当 CPU 进入睡眠或深度睡眠模式时稳压器切换到低功耗模式。CPU 从这些模式退出时，调压器返回到主模式。

– 低功耗运行模式

如果该位置 1，当 LPRUN 位置 1 时稳压器切换到低功耗模式。LPRUN 位复位时，调压器返回到主模式。

此位由软件置 1 和清零。

0：深度睡眠/睡眠/低功率运行模式下调压器打开

1：深度睡眠/睡眠/低功率运行模式下调压器进入低功耗模式

6.4.2 PWR 电源控制/状态寄存器 (PWR_CSR)

PWR power control/status register

偏移地址：0x04

复位值：0x0000 0008（不通过从待机模式唤醒进行复位）

与标准的 APB 读操作相比，读取此寄存器需要更多的 APB 周期。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EWUP 3	EWUP 2	EWUP 1	Res.	Res.	REG LPF	VOSF	VREFIN TRDYF	PVDO	SBF	WUF
					rw	rw	rw			r	r	r	r	r	r

位 31:11 保留，必须保持复位值。

位 10 **EWUP3**：使能 WKUP 引脚 3 (Enable WKUP pin 3)

此位由软件置 1 和清零。

0：WKUP 引脚 3 用于通用 I/O。WKUP 引脚 3 上的事件不会将器件从待机模式唤醒。

1：WKUP 引脚 3 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚 3 出现上升沿时从待机模式唤醒系统）。

注： 此位通过系统复位进行复位。

位 9 **EWUP2**：使能 WKUP 引脚 2 (Enable WKUP pin 2)

此位由软件置 1 和清零。

0：WKUP 引脚 2 用于通用 I/O。WKUP 引脚 2 上的事件不会将器件从待机模式唤醒。

1：WKUP 引脚 2 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚 2 出现上升沿时从待机模式唤醒系统）。

注： 此位通过系统复位进行复位。

位 8 **EWUP1**：使能 WKUP 引脚 1 (Enable WKUP pin 1)

此位由软件置 1 和清零。

0：WKUP 引脚 1 用于通用 I/O。WKUP 引脚 1 上的事件不会将器件从待机模式唤醒。

1：WKUP 引脚 1 用于从待机模式唤醒器件并被强制配置成输入下拉（WKUP 引脚 1 出现上升沿时从待机模式唤醒系统）。

注： 此位通过系统复位进行复位。

位 7:6 保留，必须保持复位值。

位 5 **REGLPF**：调压器 LP 标志 (Regulator LP flag)

MCU 处于低功耗运行模式时，此位由硬件置 1。

当 MCU 从低功耗运行模式退出时，该位保持为 1，直到调压器在主模式下准备就绪。建议轮询此位以等待调压器主模式。当调压器准备就绪后，此位由硬件复位。

0：调压器在主模式下准备就绪

1：调压器处于低功耗模式

位 4 **VOSF**：电压调节选择标志 (Voltage Scaling select flag)

电压范围变更后，内部调压器要准备好需要一定的延时。VOSF 位指示调压器已达到通过 PWR_CR 寄存器的 VOS 位定义的电压电平。

当 PWR_CR 寄存器中的 VOS[1:0] 变化时，该位复位。

调压器准备好后，该位置 1。

0：调压器在所选电压范围内准备就绪

1：调压器输出电压更改为所需 VOS 电平。

位 3 **VREFINTRDYF**：内部参考电压 (V_{REFINT}) 就绪标志 (Internal voltage reference (V_{REFINT}) ready flag)

该位指示内部参考电压 V_{REFINT} 的状态。

0：V_{REFINT} 关闭

1：V_{REFINT} 就绪

位 2 **PVDO**：PVD 输出 (PVD output)

此位通过硬件置 1 和清零。仅当通过 PVDE 位使能 PVD 时此位才有效。

0：V_{DD} 高于 PLS[2:0] 位选择的 PVD 阈值。

1：V_{DD} 低于 PLS[2:0] 位选择的 PVD 阈值。

注：PVD 在进入待机模式时停止。因此，进入待机模式或执行复位后，此位等于 0，直到 PVDE 位置 1。

位 1 **SBF**：待机标志 (Standby flag)

此位由硬件置 1，清零则只能通过 POR/PDR（上电复位/掉电复位）或将 *PWR 电源控制寄存器 (PWR_CR)* 中的 CSBF 位置 1 来实现。

0：器件未进入待机模式

1：器件已进入待机模式

位 0 **WUF**：唤醒标志 (Wakeup flag)

该位通过硬件置 1，并且只能通过系统复位或通过将 *PWR 电源控制寄存器 (PWR_CR)* 中的 CWUF 位置 1 清零

0：未发生唤醒事件

1：收到唤醒事件，可能来自 WKUP 引脚、RTC 闹钟（闹钟 A 和闹钟 B）、RTC 入侵事件、RTC 时间戳事件或 RTC 唤醒事件。

注：如果使能 WKUP 引脚（将 EWUPx (x=1, 2, 3) 位置 1）时 WKUP 引脚已为高电平，系统将检测到另一唤醒事件。

6.4.3 PWR 寄存器映射

下表对 PWR 寄存器进行了汇总。

表 38. PWR - 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPRUN	DS_EE_KOFF	VOS [1:0]		FWU	ULP	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPSDSR
	Reset value																		0		1	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWUP3	EWUP2	EWUP1	Res.	Res.	REGLPF	VOSF	VREFINTRDYF	PVDO	SBF	WUF
	Reset value																						0	0	0			0	0	1	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

7 复位和时钟控制 (RCC)

7.1 复位

共有三种类型的复位，分别为系统复位、电源复位和 RTC 域复位。

7.1.1 系统复位

系统复位会将除 RTC、RTC 备份寄存器和控制/状态寄存器（RCC_CR 和 RCC_CSR）以外的所有寄存器设为其复位值。

只要发生以下事件之一，就会产生系统复位：

- NRST 引脚低电平（外部复位）
- 窗口看门狗计数结束（WWDG 复位）
- 独立看门狗计数结束（IWDG 复位）
- 软件复位（SW 复位）（请参见*软件复位*）
- 低功耗管理复位（请参见*低功耗管理复位*）
- 选项字节加载器复位（请参见*选项字节加载器复位*）
- 退出待机模式
- 防火墙保护（请参见*第 5 节：防火墙 (FW)*）

可通过查看控制/状态寄存器 (RCC_CSR) 中的复位标志确定复位源（请参见*第 7.3.21 节*）。

软件复位

要对器件进行软件复位，必须将 Cortex®-M0+ 寄存器（应用中断和复位控制寄存器）中的 SYSRESETREQ 位置 1。更多详细信息，请参见 ARM® Cortex®-M0+ 技术参考手册。

低功耗管理复位

产生低功耗管理复位的方式有两种：

- 进入待机模式时产生复位：

  可通过清零用户选项字节中的 nRST_STDBY 位来使能此类复位。在这种情况下，只要成功执行待机模式进入序列，器件就会进行复位，而非进入待机模式。
- 进入停止模式时产生复位：

  可通过清零用户选项字节中的 nRST_STOP 位来使能此类复位。启用该位后，只要成功执行停止模式进入序列，器件就将进行复位，而非进入停止模式。

选项字节加载器复位

将 FLASH_PECR 寄存器中的 OBL_LAUNCH 位（位 18）置 1 时会产生选项字节加载器复位。该位用于通过软件启动选项字节加载。

有关用户选项字节的详细信息，请参见*第 3 节：Flash 程序存储器和数据 EEPROM (FLASH)*。

7.1.2 电源复位

只要发生以下事件之一，就会产生电源复位：

- 上电/掉电复位（POR/PDR 复位）
- BOR 复位

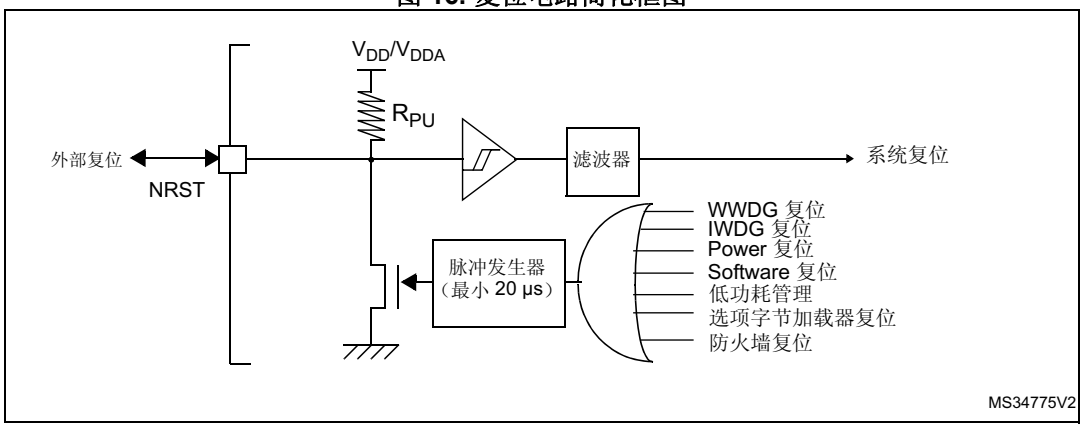
电源复位会将包含 RTC 域在内的所有寄存器设置为其复位值（请参见*图 16*）。

这些源均作用于 NRST 引脚，该引脚在复位过程中始终保持低电平。RESET 复位入口向量在存储器映射中固定在地址 0x0000_0004。有关详细信息，请参见*表 51：向量列表*。

提供给器件的系统复位信号在 NRST 引脚输出（退出待机复位除外，此复位信号不是在 NRST 引脚上输出，但会产生系统复位）。

脉冲发生器可确保每个内部复位源的复位脉冲都至少持续 20 µs。对于外部复位，在 NRST 引脚处于低电平时产生复位脉冲。

图 16. 复位电路简化框图



7.1.3 RTC 和备份寄存器复位

仅当发生以下事件之一时，RTC 外设、RTC 时钟源选择（RCC_CSR 中）以及备份寄存器才会复位：

- 软件复位，通过将 RCC_CSR 寄存器中的 RTCRST 位置 1 触发（请参见*第 7.3.21 节*）。
- 电源复位 (BOR/POR/PDR)。

7.2 时钟

可以使用四种不同的时钟源来驱动系统时钟 (SYSCLK)：

- HSI16（高速内部）振荡器时钟
- HSE（高速外部）振荡器时钟
- PLL 时钟
- MSI（多速内部）振荡器时钟

  从电源复位、系统或 RTC 域复位启动后，以及从待机模式唤醒后，会使用工作频率为 2.1MHz 的 MSI 作为系统时钟源。

  HSI16、4 分频的 HSI16 或者以任何可行频率工作的 MSI 均可用于从停止模式唤醒。

器件具有两个次级时钟源：

- 37 kHz 低速内部 RC (LSI RC)，该 RC 用于驱动独立看门狗，也可选择驱动用于从停止/待机模式自动唤醒的 RTC、LCD 以及 LPTIMER。
- 32.768 kHz 低速外部晶振（LSE 晶振），该晶振可选择驱动实时时钟 (RTCCLK)、LPTIMER、LCD 以及 USART。

对于每个时钟源来说，在未使用时都可单独打开或者关闭，以降低功耗。

可通过多个预分频器配置 AHB 频率以及两个 APB（APB1 和 APB2）域。AHB、APB1 和 APB2 域的最大频率为 32 MHz，具体取决于器件的电压范围。有关详细信息，请参见 *第 6.1.5 节：动态电压调节管理*。

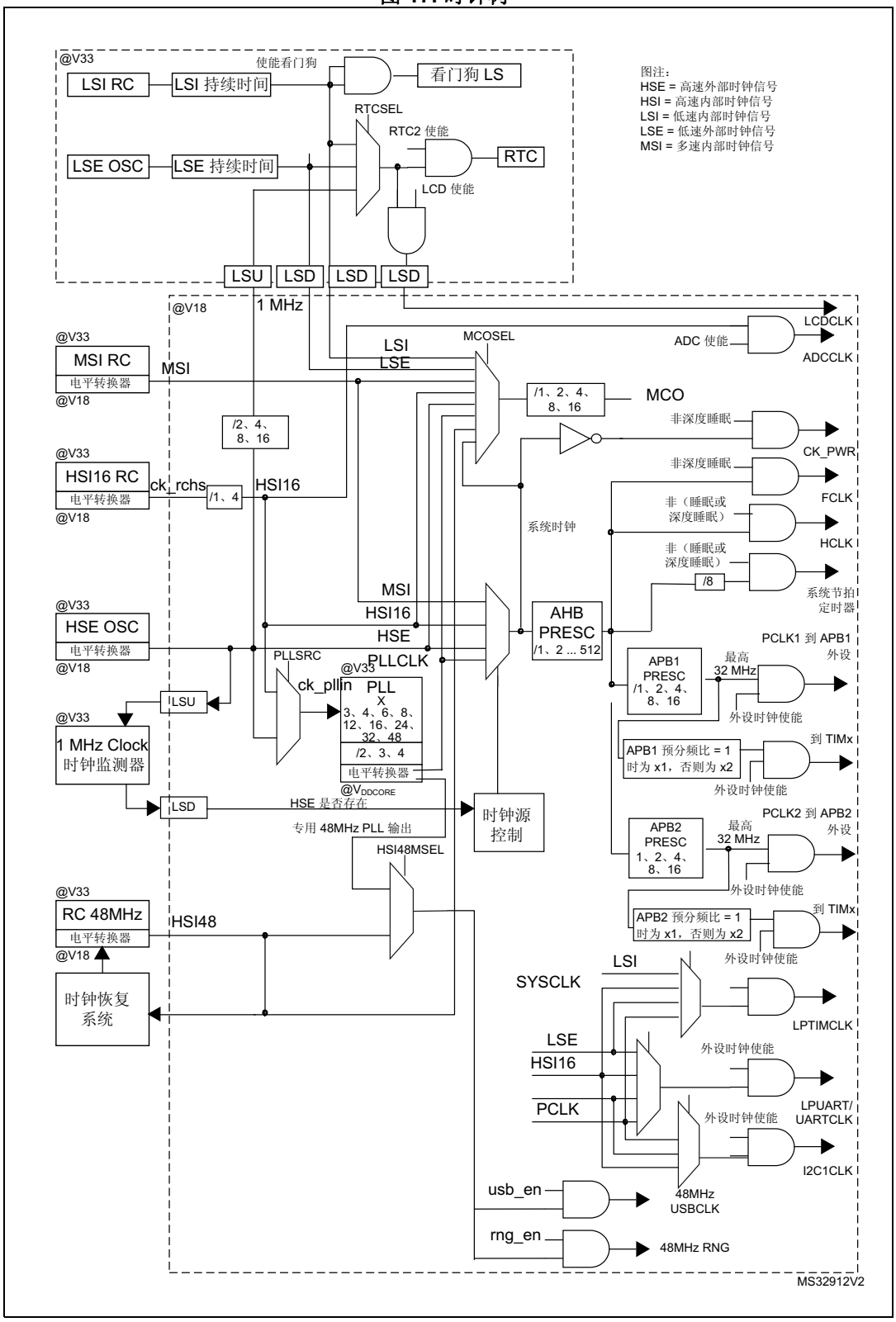
除以下时钟外，所有外设时钟均由系统时钟 (SYSCLK) 提供：

- 由以下两个时钟源之一提供的 48 MHz USB 和 RNG 时钟：
  - PLL VCO 时钟。
  - RC48 时钟 (HSI48)
- ADC 可由 APB 时钟或 HSI16 时钟提供。
- 由以下四个时钟源之一提供的 LPUART1 和 USART1/2 时钟（由软件选择）：
  - 系统时钟
  - HSI16 时钟
  - LSE 时钟
  - APB 时钟 (PCLK)
- 由以下三个时钟源之一提供的 I2C1 时钟（由软件选择）：
  - 系统时钟
  - HSI16 时钟
  - APB 时钟 (PCLK)
- 由以下四个时钟源之一提供的 LPTIMER 时钟（由软件选择）：
  - HSI16 时钟
  - LSE 时钟
  - LSI 时钟
  - APB 时钟 (PCLK)
- 由以下时钟源提供的 RTC/LCD 时钟：
  - LSE 时钟，
  - LSI 时钟，
  - 1 MHz HSE_RTC（HSE 经过可编程预分频器分频获得）。
- 始终为 LSI 时钟的 IWDG 时钟。

系统时钟 (SYSCLK) 频率必须大于或等于 RTC/LCD 时钟频率。

RCC 向 Cortex® 系统定时器 (SysTick) 外部时钟馈送 8 分频的 AHB 时钟 (HCLK)。SysTick 可使用此时钟作为时钟源，也可使用 Cortex® 时钟 (HCLK) 作为时钟源，具体可在 SysTick 控制和状态寄存器中配置。

图 17. 时钟树



1. 有关内部和外部时钟源特性的所有详细信息，请参见器件数据手册的“电气特性”部分。

定时器时钟频率由硬件自动设置为固定值。分为两种情况：

1. 如果 APB 预分频器为 1，定时器时钟频率等于 APB 域的频率。
2. 否则，等于 APB 域的频率的两倍 (×2)。

f_{CLK} 充当 Cortex®-M0+ 的自由运行时钟。更多详细信息，请参见*第 33 节：调试支持 (DBG)*。

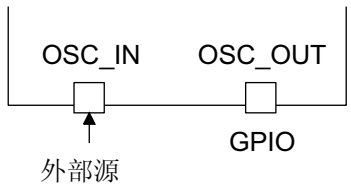
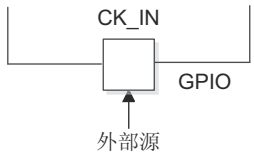
7.2.1 HSE 时钟

高速外部时钟信号 (HSE) 有 2 个时钟源：

- HSE 外部晶振/陶瓷谐振器
- HSE 用户外部时钟

谐振器和负载电容必须尽可能地靠近振荡器的引脚，以尽量减小输出失真和起振稳定时间。负载电容值必须根据所选振荡器的不同做适当调整。

图 18. HSE/LSE 时钟源

时钟源	硬件配置
用于第 3 类和第 5 类器件的外部时钟	 MSv31915V1
外部时钟	 MSv36151V1

外部源（HSE 旁路）

在此模式下，必须提供外部时钟源。最高频率不超过 32 MHz。通过将 RCC_CR 寄存器中的 HSEBYP 和 HSEON 位置 1 来选择此模式*（请参见第 7.3.1 节：时钟控制寄存器 (RCC_CR)）*。必须使用占空比约为 50% 的外部时钟信号来驱动 OSC_IN 引脚，同时 OSC_OUT 引脚应保持为高阻态 (hi-Z)（请参见*图 18*）。外部时钟信号可以是方波、正弦波或三角波。为最大程度地降低功耗，建议使用方波信号。

外部晶振/陶瓷谐振器（HSE 晶振）

1 MHz 到 24 MHz 外部振荡器的优点是精度非常高。

相关的硬件配置如*图 18*所示。有关详细信息，请参见*数据手册*的电气特性部分。

RCC_CR 寄存器的 HSERDY 标志（请参见*第 7.3.1 节*）指示 HSE 振荡器是否稳定。在启动时，硬件将该位置 1 后，HSE 时钟才可以使用。如果在 *RCC_CR 寄存器*中使能中断，则可产生中断。

HSE 晶振可通过 *RCC_CR 寄存器*中的 HSEON 位打开或关闭。

有关代码示例，请参见*A.4.1：HSE 启动序列代码示例*。

7.2.2 HSI16 时钟

HSI16 时钟信号是从 16 MHz 内部 RC 振荡器生成的。该时钟信号可直接用作系统时钟或 PLL 输入。

从低功耗停止模式唤醒后，可使用 HSI16 时钟。与使用 MSI 时钟唤醒相比，使用 HSI16 时钟的唤醒时间更短。

HSI16 RC 振荡器的优点是成本较低（无需使用外部元件）。此外，其启动速度也要比 HSE 晶振快，但即使校准后，其精度也不及外部晶振或陶瓷谐振器。

校准

因为生产工艺不同，不同芯片的 RC 振荡器频率也不同，因此 ST 会对每个器件进行出厂校准，使器件在环境温度 T_A= 25°C 时达到 1% 的精度。

复位后，工厂校准值将加载到内部时钟源校准寄存器 (RCC_ICSCR) 的 HSI16CAL[7:0] 位中*（请参见第 7.3.2 节）*。

如果应用受到电压或温度变化影响，则这可能也会影响到 RC 振荡器的速度。用户可通过 RCC_ICSCR 寄存器 中的 HSI16TRIM[4:0] 位对应用中的 HSI16 频率进行微调。有关如何测量 HSI16 频率变化的更多详细信息，请参见*第 7.2.15 节：基于 TIM21 的内部/外部时钟测量*。

RCC_CR 寄存器中的 HSI16RDY 标志指示 HSI16 振荡器是否稳定。在启动时，硬件将该位置 1 后，HSI16 才可以使用。

HSI16 RC 振荡器可通过 RCC_CR 寄存器中的 HSI16ON 位打开或关闭。

7.2.3 MSI 时钟

MSI 时钟信号是从内部 RC 振荡器生成的。频率范围可通过软件使用 RCC_ICSCR 寄存器中的 MSIRANGE[2:0] 位进行调整（请参见*第 7.3.2 节：内部时钟源校准寄存器 (RCC_ICSCR)*）。有七个频率范围可用：65.536 kHz、131.072 kHz、262.144 kHz、524.288 kHz、1.048 MHz、2.097 MHz（默认值）和4.194 MHz。

在从复位重启、从待机唤醒后，始终会将 MSI 时钟用作系统时钟。从停止模式唤醒后，可选择 MSI 时钟作为系统时钟，而不使用 HSI16（或 HSI16/4）作为系统时钟。

当器件在复位后重启或者从待机模式唤醒时，MSI 频率会设为其默认值。从停止模式唤醒后，MSI 频率保持不变。

MSI RC 振荡器的优势在于可提供一个低成本（无外部元件）低功耗的时钟源。此振荡器用作低功耗模式下的唤醒时钟，以降低功耗。

RCC_CR 寄存器中的 MSIRDY 标志指示 MSI RC 是否稳定。在启动时，硬件将该位置 1 后，MSI RC 输出时钟才可以使用。

MSI RC 可通过 RCC_CR 寄存器*中的 MSION 位打开或关闭（请参见第 7.3.1 节*）。

MSI RC 还可作为备份时钟源（辅助时钟）使用，以防 HSE 晶振发生故障。请参见*第 169 页上的第 7.2.10 节：HSE 时钟安全系统(CSS)*。

校准

因为生产工艺不同，不同芯片的 MSI RC 振荡器频率也不同，因此 ST 会对每个器件进行出厂校准，是器件在环境温度 T_A 为 25°C 时达到 1% 的精度。

复位后，工厂校准值将加载到 RCC_ICSCR 寄存器的 MSICAL[7:0] 位中。如果应用受到电压或温度变化影响，则这可能也会影响到 RC 振荡器的速度。用户可通过 RCC_ICSCR 寄存器中的 MSITRIM[7:0] 位对 MSI 频率进行微调。有关如何测量 MSI 频率变化的更多详细信息，请参见*第 7.2.15 节：基于 TIM21 的内部/外部时钟测量*。

7.2.4 HSI48 时钟

HSI48 时钟信号由内部 48 MHz RC 振荡器生成，可直接用于 USB 和随机数发生器 (RNG)。

内部 48MHz RC 振荡器主要用于通过特殊的时钟恢复系统 (CRS) 电路向 USB 外设提供高精度时钟。CRS 可使用 USB SOF 信号、LSE 或外部信号自动而快速地实时调整振荡器频率。系统进入停止或待机模式后，CRS 会立即被禁止。未使用 CRS 时，HSI48 RC 振荡器以其默认频率运行，具体默认频率值取决于生产工艺。

有关如何配置和使用 CRS 外设的更多详细信息，请参见*第 8 节：时钟恢复系统 (CRS)*。

HSI48 要求使能与 48 MHz RC 相关的 VREFINT 及其缓冲区（请参见*第 10.2.3 节：参考控制和状态寄存器 (SYSCFG_CFGR3)*中的 ENREF_HSI48）。

时钟恢复 RC 寄存器 (RCC_CRRCR) 中的 HSI48RDY 标志指示 HSI48 RC 是否稳定。在启动时，硬件将该位置 1 后，HSI48 才可以使用。

HSI48 RC 可通过时钟恢复 RC 寄存器 (RCC_CRRCR) 中的 HSI48ON 位打开或关闭。

7.2.5 PLL

内部 PLL 的时钟可由 HSI16 RC 或 HSE 时钟提供。它会驱动系统时钟，并可生成供 USB 外设使用的 48 MHz 时钟（请参见*图 17* 和*第 7.3.1 节：时钟控制寄存器 (RCC_CR)*）。

PLL 输入时钟频率必须介于 2 MHz 和 24 MHz 之间。

所需频率通过 PLL 中内置的倍频系数和输出分频系数获得：

- 如果 USB 使用 PLL 作为时钟源，则 PLL VCO 时钟（由 PLL 倍频系数定义）必须编程为输出 96 MHz 的频率 (USBCLK = PLLVCO/2)。
- 系统时钟由 PLL VCO 经过输出分频系数分频后获得。

注：应用软件必须正确设置 PLL 倍频系数，以免
乘积在范围 1 内时，PLL VCO 时钟频率超出 96 MHz；

乘积在范围 2 内时，PLL VCO 时钟频率超出 48 MHz；

乘积在范围 3 内时，PLL VCO 时钟频率超出 24 MHz。

应用软件还必须正确设置输出分频系数，以免 SYSCLK 时钟频率超出 32 MHz。

PLL 的最小输入时钟频率为 2 MHz（使用 HSE 作为 PLL 时钟源的情况）。

必须在使能 PLL 之前配置 PLL（选择源时钟、倍频系数和输出分频系数）。PLL 使能后，这些参数不可更改。

要修改 PLL 配置，请执行以下步骤操作：

1. 通过将 PLLON 位设置为 0 禁止 PLL。
2. 等待直到 PLLRDY 清零。PLL 现已完全停止。
3. 根据需要更改参数。
4. 通过将 PLLON 位置 1 再次使能 PLL。

如果已在 RCC_CIER 寄存器中使能中断，则 PLL 就绪时便可产生中断（请参见*第 7.3.5 节*）。

有关代码示例，请参见*A.4.2：PLL 配置修改代码示例*。

7.2.6 LSE 时钟

LSE 晶振是 32.768 kHz 低速外部晶振或陶瓷谐振器。可作为实时时钟 (RTC) 的时钟源来提供时钟/日历或其它定时功能，具有功耗低且精度高的优点。

LSE 晶振可通过 RCC_CSR 寄存器中的 LSEON 位打开和关闭（请参见*第 7.3.21 节*）。

通过 RCC_CSR 寄存器中的 LSEDRV[1:0] 位，可在运行时更改晶振驱动强度，以实现稳健性、短启动时间和低功耗之间的最佳平衡。可在不同驱动水平之间动态更改驱动能力，但到达低驱动模式的情况除外。此时，只能通过上电复位或 RTC 复位切换到另一模式。

RCC_CSR 寄存器中的 LSERDY 标志指示 LSE 晶振是否稳定。在启动时，硬件将该位置 1 后，LSE 晶振输出时钟信号才可以使用。如果在 RCC_CIER 寄存器中使能中断，则可产生中断（请参见*第 7.3.5 节*）。

外部源（LSE 旁路）

在此模式下，必须提供外部时钟源。最高频率不超过 1 MHz。通过将 RCC_CSR 寄存器中的 LSEBYP 和 LSEON 位置 1 来选择此模式（请参见*第 7.3.1 节*）。必须使用占空比约为 50% 的外部时钟信号（方波、正弦波或三角波）来驱动 OSC32_IN 引脚，同时 OSC32_OUT 引脚应保持为高阻态 (Hi-Z)（请参见*图 18*）。

7.2.7 LSI 时钟

LSI RC 可作为低功耗时钟源在停止和待机模式下保持运行，供独立看门狗 (IWDG) 使用。时钟频率在 37 kHz 左右。

LSI RC 振荡器可通过 RCC_CSR 寄存器中的 LSION 位打开或关闭（请参见*第 7.3.21 节*）。

RCC_CSR 中的 LSIRDY 标志指示低速内部振荡器是否稳定。在启动时，硬件将该位置 1 后，此时钟才可以使用。如果在 RCC_CIER 中使能中断，则可产生中断（请参见*第 7.3.5 节*）。

从 IWDG 激活时候起，LSI 振荡器不能通过 LSION=0 停止。LSI 振荡器通过系统复位停止（通过硬件使用 FLASH_OPTR 寄存器中的 WDG_SW 选项位使能 IWDG 的情况除外）。如果 IWDG 已通过软件使能，则必须在系统复位后再次使能 LSI 振荡器，以确保 IWDG 和/或 RTC 正常工作。

LSI 测量

可通过测量 LSI 振荡器的频散来获得准确的 RTC 时基和/或实现精度水平可以接受的 IWDG 超时（LSI 用作这些外设的时钟源时除外）。有关详细信息，请参见数据手册的电气特性部分。有关如何测量 LSI 频率的更多详细信息，请参见*第 7.2.15 节：基于 TIM21 的内部/外部时钟测量*。

7.2.8 系统时钟 (SYSCLK) 选择

可以使用四种不同的时钟源来驱动系统时钟 (SYSCLK)：

- HSI16 振荡器
- HSE 振荡器
- PLL
- MSI 振荡器时钟（复位后的默认时钟）

在直接使用或者通过 PLL 使用某一时钟源作为系统时钟时，该时钟源无法停止。

只有在目标时钟源已就绪时（时钟在启动延迟或 PLL 锁相后稳定时），才可从一个时钟源切换到另一个。如果选择尚未就绪的时钟源，则切换在该时钟源就绪时才会进行。RCC_CR 寄存器中的状态位指示哪个（些）时钟已就绪，以及当前哪个时钟正充当系统时钟。

7.2.9 系统时钟源频率与电压范围

下表依照产品电压范围列出了不同时钟源的最大频率。

表 39. 系统时钟源频率

产品电压范围	时钟频率			
	MSI	HSI16	HSE	PLL
范围 1 (1.8 V)	4.2 MHz	16 MHz	HSE 32 MHz（外部时钟） 或 24 MHz（晶振）	32 MHz （PLLVCO 最大值 = 96 MHz）
范围 2 (1.5 V)	4.2 MHz	16 MHz	16 MHz	16 MHz （PLLVCO 最大值 = 48 MHz）
范围 3 (1.2 V)	4.2 MHz	NA	8 MHz	4 MHz （PLLVCO 最大值 = 24 MHz）

7.2.10 HSE 时钟安全系统(CSS)

时钟安全系统可通过软件在 HSE 上激活。激活后，时钟监测器将在 HSE 振荡器启动延迟后使能，并在此振荡器停止时被关闭。

如果检测到 HSE 时钟故障，则会自动禁止该振荡器，并且会产生 CSSHSEI 中断（时钟安全系统中断）以通知软件振荡器发生故障，从而使 MCU 执行救援操作。CSSHSEI 与 Cortex®-M0+ NMI（不可屏蔽中断）异常向量相链接。

注：当 CSSHSE 使能后，如果 HSE 时钟发生故障，则会产生 CSSHSE 中断，并会自动产生 NMI。NMI 将无限期执行，除非将 CSSHSE 中断挂起位清零。因此，NMI 中断服务程序 (ISR) 必须将 RCC_CICR 寄存器中的 CSSHSEC 位置 1，以清除 CSSHSE 中断。

如果直接或间接使用 HSE 振荡器作为系统时钟（间接是指它用作 PLL 输入时钟，PLL 时钟用作系统时钟），检测到故障时会导致系统时钟切换并禁止 HSE 振荡器。当故障发生时，如果 HSE 振荡器时钟为正在用作系统时钟的 PLL 时钟输入，则也会禁止该 PLL。

发生 HSE 故障时，系统时钟可切换为 MSI 或内部 16 MHz HSI 时钟，具体取决于 RCC_CFGR 寄存器中 STOPWUCK 位的值。

7.2.11 LSE 时钟安全系统

时钟安全系统可通过软件在 LSE 上激活。可通过写入 RCC_CSR 寄存器中的 CSSLSEON 位来完成此操作。可通过硬件复位、RTC 软件复位、或在检测到 LSE 时钟故障后禁止该位。LSE 和 LSI 时钟使能（LSEON 和 LSION 置 1）并就绪（通过硬件将 LSERDY 和 LSIRDY 位置 1）后，以及通过 RTCSEL 位选择 RTC 时钟后，必须写入 CSSLSEON 位。

LSE CSS 适用于所有模式：运行、睡眠、停止和待机。

如果在外部 32 kHz 振荡器上检测到故障，则不会再向 RTC 提供 LSE 时钟，但寄存器的内容保持不变。

在待机模式下，会产生唤醒。在任何其它模式下，可发送中断来唤醒软件（请参见*第 7.3.5 节*）。

之后，软件必须将 CSSLSEON 位复位，并通过将 LSEON 位复位的方式停用出现故障的 32 kHz 振荡器。软件可通过 RTCSEL 位更改 RTC 时钟源（LSI、HSE 或无时钟），也可采取任何必要措施来确保应用的安全。

7.2.12 RTC 和 LCD 时钟

RTC 和 LCD 使用同一时钟源，可以是 LSE、LSI 或 HSE 1 MHz 时钟（HSE 经过可编程预分频器分频获得）。选择方式是编程 RCC_CSR 寄存器中的 RTCSEL[1:0] 位（请参见*第 7.3.21 节*）和 RCC_CR 寄存器中的 RTCPRE[1:0] 位（请参见*第 7.3.1 节*）。

一旦选定 RTC 和 LCD 时钟源后，只能通过将 RCC_CSR 寄存器中的 RTCRST 位置 1 的方式或者通过 POR 来修改。

如果使用 LSE 或 LSI 作为 RTC 时钟源，RTC 会继续在停止和待机两种低功耗模式下工作，并可用作唤醒源。但是，如果使用 HSE 作为 RTC 时钟源，则 RTC 不能在停止和待机两种低功耗模式下使用。如果使用 LSE 或 LSI 作为 RTC 时钟源，则 LCD 可在停止低功耗模式下使用。

如果 RTC 的时钟由 LSE 提供，则 RTC 在系统复位后仍可获得时钟并保持正常工作。

*注：为了能够在 APB1 时钟频率低于 RTC 时钟频率的七倍 (7*RTCLCK) 时读取 RTC 日历寄存器，软件必须分两次读取日历时间和日期寄存器。*

这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。

7.2.13 看门狗时钟

如果独立看门狗 (IWDG) 已通过硬件选项或软件访问的方式启动，则 LSI 振荡器将强制打开且不可禁止。在 LSI 振荡器稳定后，时钟将提供给 IWDG。

如果已通过软件使能 IWDG，则系统复位后会禁止 LSI。之后，必须再次使能 LSI 振荡器，以确保 IWDG 正常工作。

7.2.14 时钟输出功能

微控制器时钟输出 (MCO) 功能允许使用可配置的预分频值（1、2、4、8 或 16）将时钟输出到外部 MCO 引脚上。必须在复用功能模式下对相应 GPIO 端口的配置寄存器进行编程。可选择以下 7 种时钟信号之一作为 MCO 时钟：

- SYSCLK
- HSI16
- HSI48
- MSI
- HSE
- PLL
- LSI
- LSE

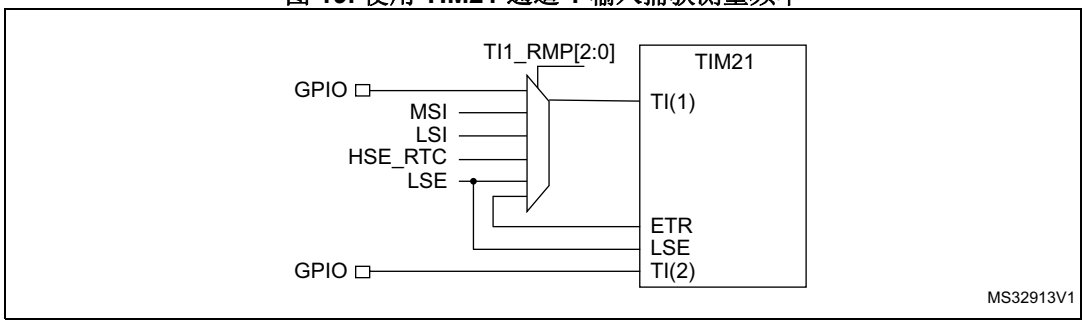
信号选择由 RCC_CFGR 寄存器中的 MCOSEL[3:0] 位控制（请参见*第 7.3.20 节*）。

有关代码示例，请参见*A.4.3：MCO 选择代码示例*。

7.2.15 基于 TIM21 的内部/外部时钟测量

所有板上时钟源发生器的频率都可通过对 TIM21 通道 1 的输入捕获进行间接测量，如*图 19* 所示。

图 19. 使用 TIM21 通道 1 输入捕获测量频率



TIM21 具有一个输入复用器，用于选择使用哪一 I/O 或内部时钟触发输入捕获。此选择通过 TIM21_OR 寄存器的 TI1_RMP [2:0] 位执行。

将 LSE 连接到通道 1 输入捕获的主要目的是为了能够精确测量 HSI16 和 MSI 系统时钟（为此，应将 HSI16 或 MSI 用作系统时钟源）。借助 LSE 信号连续边沿之间的 HSI16（或 MSI）时钟计数数量，即可对内部时钟周期进行测量。利用 LSE 的高精度（通常为几十 ppm），用户能以同一分辨率测定时钟频率，并可通过对时钟源进行微调来补偿由生产工艺造成的和/或与温度和电压相关的频率偏差。

MSI 和 HSI16 振荡器均设有针对此目的的专用校准位，且支持用户访问。

其基本原理是基于相对的测量（例如，HSI16/LSE 比）：因此，精度与两个时钟源之比紧密相关。比率越大，测量效果越好。

但是，如果 MSI 时钟频率较低（通常低于 1 MHz），测量精度会不够好。在这种情况下，建议：

- 累加连续多次捕获的结果
- 使用定时器的输入捕获预分频值（最多每 8 个周期进行 1 次捕获）
- 使用频率为 512 Hz 的 RTC_OUT 信号（如果 RTC 的时钟由 LSE 提供）作为通道 1 输入捕获的输入。这样可提高测量精度

也可使用 TIM21 测量 LSI、MSI 或 HSE_RTC：此方法对于不含晶振的应用非常有用。超低功耗 LSI 振荡器具有较大的生产工艺偏差：通过测量该振荡器与 HSI16 时钟源的函数关系，可以借助 HSI16 精度测定其频率。如果 HSE_RTC 频率（HSE 经过可编程预分频器分频获得）相对较高 (1 MHz)，相对频率测量就不会非常精确。这种做法的主要目的是获得外部晶振频率的粗略指示。此功能非常实用，可借此测定谐波频率或分谐波频率（–50/+100% 偏差），从而满足 IEC 60730/IEC 61335 标准的要求。

7.2.16 TIM2/TIM21/TIM22 独立于系统时钟的时钟源

在一些使用 32.768 kHz 时钟作为 RTC 时基的应用中，与系统时钟完全无关的时基会非常有用。使用这类时基时，无需考虑处理器状态（处理器可处于停止状态，也可以低速、中速或全速执行操作）即可安排任务。

为此，LSE 时钟会在内部重定向到 3 个定时器的 ETR 输入，这些输入用作附加的时钟源。这样便可提供多达三个独立时基（使用自动重载功能），其中有 1 个或 2 个附加的比较通道用于小数事件。例如，可将 TIM21 自动重载中断编程为 1 秒节拍中断，外加主节拍 250 ms 后发生的附加中断。

注：在该配置中，外部时钟 (LSE) 与 APB 时钟之比不得小于 2。如果应用使用的 APB 时钟频率小于 LSE 时钟频率的二倍（通常 LSE = 32.768 kHz，因此 LSE 的二倍 = 65.536 kHz），则必须使用定时器的外部触发预分频器功能：此功能最多可将 ETR 时钟进行 8 分频。

7.3 RCC 寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

7.3.1 时钟控制寄存器 (RCC_CR)

Clock control register

偏移地址：0x00

系统复位值：0b0000 0000 00XX 0X00 0000 0011 0000 0000，其中 X 未定义

上电复位值：0x0000 0300

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLLON	Res.	Res.	RTCPRE[1:0]		CSSHS EON	HSE BYP	HSE RDY	HSE ON
						r	rw			rw	rw	rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MSI RDY	MSION	Res.	Res.	HSI16 OUTEN	HSI16 DIVF	HSI16 DIVEN	HSI16 RDYF	HSI16K ERON	HSI16 ON
						r	rw			rw	r	rw	r	rw	rw

位 31:26 保留，必须保持复位值。

位 25 **PLLRDY**：PLL 时钟就绪标志 (HSE clock ready flag)

该位由硬件置 1，用于指示 PLL 已锁定。

0：PLL 未锁定

1：PLL 已锁定

位 24 **PLLON**：PLL 使能位 (PLL enable bit)

此位由软件置 1 和清零，以使能 PLL。

当进入停止或待机模式时由硬件清零。如果 PLL 时钟用作系统时钟或被选作系统时钟，该位不能复位。

0：PLL 关闭

1：PLL 开启

位 23:22 保留，必须保持复位值。

位 21:20 **RTCPRE[1:0]** RTC/LCD 预分频器 (RTC/LCD prescaler)

这些位由软件置 1 和清零，以从 HSE 获取 1 MHz 时钟。如果 HSE 已使能 (HSEON = 1)，则不能修改该预分频器。这些位通过上电复位来复位。该值不能通过系统复位进行修改。

00: HSE 进行 2 分频可得到 RTC/LCD 时钟

01: HSE 进行 4 分频可得到 RTC/LCD 时钟

10: HSE 进行 8 分频可得到 RTC/LCD 时钟

11: HSE 进行 16 分频可得到 RTC/LCD 时钟

位 19 **CSSHSEON**：HSE 上的时钟安全系统使能位 (Clock security system on HSE enable bit)

该位由软件置 1，以使能 HSE 上的时钟安全系统 (CSS)。该位为“只置 1”（通过系统复位禁止）。当 CSSHSEON 置 1 时，时钟监测器将在 HSE 振荡器就绪时由硬件使能，并在检出振荡器故障时由硬件禁止。

0：时钟安全系统关闭（时钟监测器关闭）

1：时钟安全系统开启（如果 HSE 振荡器稳定，则时钟监测器打开；如果不稳定，则关闭）

位 18 **HSEBYP**：HSE 时钟旁路位 (HSE clock bypass bit)

此位由软件置 1 和清零，用于用外部时钟旁路振荡器。外部时钟必须通过 HSEON 位使能才能为器件使用。

HSEBYP 只有在 HSE 振荡器已禁止的情况下才可写入。该位通过上电复位进行复位。其值不能通过系统复位进行修改。

0：不旁路 HSE 振荡器

1：外部时钟旁路 HSE 振荡器

位 17 **HSERDY**：HSE 时钟就绪标志 (HSE clock ready flag)

该位由硬件置 1，用于指示 HSE 振荡器已稳定。在将 HSEON 位清零后，HSERDY 将在 6 个 HSE 振荡器时钟周期后转为低电平。

0：HSE 振荡器未就绪

1：HSE 振荡器已就绪

位 16 **HSEON**：HSE 时钟使能位 (HSE clock enable bit)

此位由软件置 1 和清零。

由硬件清零，用于在进入停止或待机模式时停止 HSE 振荡器。如果 HSE 振荡器直接或间接用作系统时钟，则该位不可复位。

0：HSE 振荡器关闭

1：HSE 振荡器开启

位 15:10 保留，必须保持复位值。

位 9 **MSIRDY**：MSI 时钟就绪标志 (MSI clock ready flag)

该位由硬件置 1，用于指示 MSI 振荡器已稳定。

0：MSI 振荡器未就绪

1：MSI 振荡器已就绪

注： 在将 MSION 位清零后，MSIRDY 将在 6 个 MSI 时钟周期后转为低电平。

位 8 **MSION**：MSI 时钟使能位 (MSI clock enable bit)

此位由软件置 1 和清零。

由硬件置 1，用于在退出停止或待机模式时或者在直接或间接用作系统时钟的 HSE 振荡器发生故障时强制 MSI 振荡器打开。如果 MSI 用作系统时钟，则该位不可清零。

0：MSI 振荡器关闭

1：MSI 振荡器开启

位 7:6 保留，必须保持复位值。

位 5 **HSI16OUTEN**：16 MHz 高速内部时钟输出使能 (16 MHz high-speed internal clock output enable)

此位由软件置 1 和清零。该位置 1 时，若 *TIM2 选项寄存器 (TIM2_OR)* 中的 ETR_RMP 设为 011，则 TIM2 ETR 输入会连接至 16 MHz HSI 输出时钟 (HSI16)。可随时通过应用写入该位。

0：禁止 HSI16 输出时钟

1：使能 HSI16 输出时钟

位 4 **HSI16DIVF**：HSI16 分频器标志 (HSI16 divider flag)

该位由硬件置 1 和复位。由于写入 HSI16DIVEN 不会直接对频率起作用，因此该标志指示的是 HSI16 分频器的状态。

0：16 MHz HSI 时钟未分频

1：16 MHz HSI 时钟进行了 4 分频

位 3 **HSI16DIVEN**：HSI16 分频器使能位 (HSI16 divider enable bit)

该位通过软件置 1 和复位，用于使能/禁止对 16 MHz HSI 进行 4 分频。可随时写入该位。

0：未请求对 16 MHz HSI 进行分频

1：请求对 16 MHz HSI 进行 4 分频

位 2 **HSI16RDYF**：内部高速时钟就绪标志 (Internal high-speed clock ready flag)

该位由硬件置 1，用于指示 HSI 16 MHz 振荡器已稳定。仅当 HSI 16 MHz 振荡器通过 HSI16KERON 或 IP 请求使能时，该位才会被置“1”。在将 HSI16ON 位清零后，HSI16RDY 将在 6 个 HSI16 时钟周期后转为低电平。

0：HSI 16 MHz 振荡器未就绪

1：HSI 16 MHz 振荡器已就绪

位 1 **HSI16KERON**：一些 IP 内核的高速内部时钟使能位 (High-speed internal clock enable bit for some IP kernels)

该位由软件置 1 和复位，即使在停止模式下也可强制 HSI 16 MHz 振荡器开启，从而可快速用作 USART 或 I2C1 的内核时钟。该位对 HSI16ON 的值没有任何影响。

0：未强制 HSI 16 MHz 振荡器开启

1：即使在停止模式下也强制 HSI 16 MHz 振荡器开启

位 0 **HSI16ON**：16 MHz 高速内部时钟使能 (16 MHz high-speed internal clock enable)

此位由软件置 1 和清零。如果 16 MHz HSI 直接或间接用作系统时钟，则该位不可清零。

0：HSI16 振荡器关闭

1：HSI16 振荡器开启

7.3.2 内部时钟源校准寄存器 (RCC_ICSCR)

Internal clock sources calibration register

偏移地址：0x04

复位值：0x00XX B0XX，其中 X 未定义。

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSITRIM[7:0]								MSICAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIRANGE[2:0]			HSI16TRIM[4:0]					HSI16CAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

位 31:24 **MSITRIM[7:0]**：MSI 时钟微调 (MSI clock trimming)

这些位由软件置 1，用于调整 MSI 校准。

通过这些位，可在 MSICAL[7:0] 位基础上实现可由用户编程的微调值。可通过编程这些位来补偿电压和温度的差异，使内部 MSI RC 的频率更为准确。

位 23:16 **MSICAL[7:0]**：MSI 时钟校准 (MSI clock calibration)

这些位在启动时自动初始化。

位 15:13 **MSIRANGE[2:0]**：MSI 时钟范围 (MSI clock ranges)

这些位由软件设置，用于选择 MSI.7 的频率范围，可选频率范围包括：

000：范围 0，65.536 kHz 左右
001：范围 1，131.072 kHz 左右
010：范围 2，262.144 kHz 左右
011：范围 3，524.288 kHz 左右
100：范围 4，1.048 MHz 左右
101：范围 5，2.097 MHz 左右（复位值）
110：范围 6，4.194 MHz 左右
111：不允许

位 12:8 **HSI16TRIM[4:0]**：高速内部时钟微调 (High speed internal clock trimming)

通过这些位，可在 HSI16CAL[7:0] 位基础上实现可由用户编程的微调值。可通过编程这些位来补偿电压和温度的差异，使内部 HSI16 RC 的频率更为准确。

位 7:0 **HSI16CAL[7:0]**：内部高速时钟校准 (Internal high speed clock calibration)

这些位在启动时自动初始化。

7.3.3 时钟恢复 RC 寄存器 (RCC_CRRCR)

Clock recovery RC register

地址：0x08

复位值：0x0000 XX00

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSI48CAL[7:0]								Res.	Res.	Res.	Res.	Res.	HSI48 DIV6EN	HSI48RDY	HSI48ON
r	r	r	r	r	r	r	r						rw	r	rw

位 31:16 保留，必须保持复位值。

位 15:8 **HSI48CAL[7:0]**：48 MHz HSI时钟校准 (48 MHz HSI clock calibration)

这些位为只读。这些位在系统复位过程中通过加载选项字节由硬件置 1。

位 7:3 保留，必须保持复位值。

位 2 **HSI48DIV6EN**：48 MHz HSI 时钟 6 分频输出使能 (48 MHz HSI clock divided by 6 output enable)

此位由软件置 1 和清零。该位置 1 时，HSI48/6 时钟会传送到 TIM3。

0：HSI48/6 输出传送未使能

1：HSI48/6 输出传送已使能

位 1 **HSI48RDY**：48MHz HSI 时钟就绪标志 (48MHz HSI clock ready flag)

该位由硬件置 1，用于指示 48 MHz RC 振荡器已稳定。HSION 复位后，需要经过 6 个 48 MHz RC 振荡器时钟周期才能变为低电平。

0：48 MHz HSI 时钟未就绪

1：48 MHz HSI 时钟已就绪

位 0 **HSI48ON**：48MHz HSI 时钟使能位 (48MHz HSI clock enable bit)

此位由软件置 1 和清零。

0：48 MHz HSI 时钟关闭

1：48 MHz HSI 时钟开启

7.3.4 时钟配置寄存器 (RCC_CFGR)

Clock configuration register

偏移地址：0x0C

复位值：0x0000 0000

访问：0 ≤ 等待周期 ≤ 2，按字、半字和字节访问

只有在时钟源切换期间进行访问时才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				PLLDIV[1:0]		PLLMUL[3:0]				Res.	PLL SRC
	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
STOP WUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

位 31 保留，必须保持复位值。

位 30:28 **MCOPRE[2:0]**：微控制器时钟输出预分频器 (Microcontroller clock output prescaler)

这些位由软件置 1 和清零。

强烈建议在 MCO 输出使能之前更改此预分频器。

000：MCO 1 分频
001：MCO 2 分频
010：MCO 4 分频
011：MCO 8 分频
100：MCO 16 分频
其它值：不允许使用

位 27:24 **MCOSEL[3:0]**：微控制器时钟输出选择 (Microcontroller clock output selection)

这些位由软件置 1 和清零。

0000：禁止 MCO 输出，MCO 上无时钟
0001：选择 SYSCLK 时钟
0010：选择 HSI16 振荡器时钟
0011：选择 MSI 振荡器时钟
0100：选择 HSE 振荡器时钟
0101：选择 PLL 时钟
0110：选择 LSI 振荡器时钟
0111：选择 LSE 振荡器时钟
1000：选择 HSI48 振荡器时钟
其它值：保留

注： 该时钟输出在启动时或 MCO 时钟源切换期间可能包含一些被截断的周期。

位 23:22 **PLLDIV[1:0]**：PLL 输出分频 (PLL output division)

这些位由软件置 1 和清零，用于控制 PLL VCO 时钟的 PLL 输出时钟分频。这些位只能在 PLL 已禁止时写入。

00：不允许
01：PLL 时钟输出 = PLLVCO / 2
10：PLL 时钟输出 = PLLVCO / 3
11：PLL 时钟输出 = PLLVCO / 4

位 21:18 **PLLMUL[3:0]**：PLL 倍频系数 (PLL multiplication factor)

这些位由软件写入，用于定义 PLL 倍频系数，以生成 PLL VCO 时钟。这些位只能在 PLL 已禁止时写入。

0000：PLLVCO = PLL 时钟输入 x 3
0001：PLLVCO = PLL 时钟输入 x 4
0010：PLLVCO = PLL 时钟输入 x 6
0011：PLLVCO = PLL 时钟输入 x 8
0100：PLLVCO = PLL 时钟输入 x 12
0101：PLLVCO = PLL 时钟输入 x 16
0110：PLLVCO = PLL 时钟输入 x 24
0111：PLLVCO = PLL 时钟输入 x 32
1000：PLLVCO = PLL 时钟输入 x 48
其它值：不允许使用

小心： 当乘积在范围 1 内时，PLL VCO 时钟频率不得超过 96 MHz；当乘积在范围 2 内时，PLL VCO 时钟频率不得超过 48 MHz；当乘积在范围 3 内时，PLL VCO 时钟频率不得超过 24 MHz。

位 17 保留，必须保持复位值。

位 16 **PLLSRC**：PLL 输入时钟源 (PLL entry clock source)

此位由软件置 1 和清零，用以选择 PLL 时钟源。该位只能在 PLL 已禁止时写入。

0：选择 HSI16 振荡器时钟作为 PLL 输入时钟
1：选择 HSE 振荡器时钟作为 PLL 输入时钟

注： PLL 最小输入时钟频率为2 MHz。

位 15 **STOPWUCK**：从停止模式唤醒时钟选择 (Wake-up from Stop clock selection)

此位由软件置 1 和清零，用以选择从停止模式唤醒时钟。

0：选择内部 64 KHz 到 4 MHz (MSI) 振荡器作为从停止模式唤醒时钟
1：选择内部 16 MHz (HSI16) 振荡器作为从停止模式唤醒时钟（如果 HSI16DIVEN=1，则选择 HSI16/4）

位 14 保留，必须保持复位值。

位 13:11 **PPRE2[2:0]**：APB 高速预分频器 (APB2) (APB high-speed prescaler (APB2))

这些位由软件置 1 和清零，用于控制 APB 高速时钟 (PCLK2) 的分频系数。

0xx：HCLK 不分频
100：HCLK 2 分频
101：HCLK 4 分频
110：HCLK 8 分频
111：HCLK 16 分频

位 10:8 **PPRE1[2:0]**：APB 低速预分频器 (APB1) (APB low-speed prescaler (APB1))

这些位由软件置 1 和清零，用于控制 APB 低速时钟 (PCLK1) 的分频系数。

0xx：HCLK 不分频
100：HCLK 2 分频
101：HCLK 4 分频
110：HCLK 8 分频
111：HCLK 16 分频

位 7:4 **HPRE[3:0]**：AHB 预分频器 (AHB prescaler)

这些位由软件置 1 和清零，用于控制 APB 时钟的分频系数。

小心： 软件必须根据器件电压范围正确设置这些位，以确保系统频率不会超过允许的最大频率（更多详细信息，请参见 PWR 一章中的“动态电压调节管理”部分）。在对这些位执行写操作之后以及减小电压范围之前，必须读取该寄存器以确保其中的值为新值。

0xxx：SYSCLK 不分频
1000：SYSCLK 2 分频
1001：SYSCLK 4 分频
1010：SYSCLK 8 分频
1011：SYSCLK 16 分频
1100：SYSCLK 64 分频
1101：SYSCLK 128 分频
1110：SYSCLK 256 分频
1111：SYSCLK 512 分频

位 3:2 **SWS[1:0]**：系统时钟切换状态 (System clock switch status)

这些位由硬件置 1 和清零，用于指示用作系统时钟的时钟源。

00：MSI 振荡器用作系统时钟
01：HSI16 振荡器用作系统时钟
10：HSE 振荡器用作系统时钟
11：PLL 用作系统时钟

位 1:0 **SW[1:0]**：系统时钟切换 (System clock switch)

这些位由软件置 1 和清零，用于选择 SYSCLK 源。

由硬件置 1，用于在退出待机模式时或者在直接或间接用作系统时钟的 HSE 振荡器发生故障时强制 MSI 的选择（如果时钟安全系统已使能）。

00：MSI 振荡器用作系统时钟
01：HSI16 振荡器用作系统时钟
10：HSE 振荡器用作系统时钟
11：PLL 用作系统时钟

7.3.5 时钟中断使能寄存器 (RCC_CIER)

Clock interrupt enable register

地址：0x10

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSS LSE	HSI48 RDYIE	MSI RDYIE	PLL RDYIE	HSE RDYIE	HSI16 RDYIE	LSE RDYIE	LSI RDYIE
								r	r	r	r	r	r	r	r

位 31:8 保留，必须保持复位值。

位 7 **CSSLSE**：LSE CSS 中断标志 (LSE CSS interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由外部 32 kHz 振荡器上的时钟安全系统引起的中断。

0：禁止 LSE CSS 中断

1：使能 LSE CSS 中断

位 6 **HSI48RDYIE**：HSI48 就绪中断标志 (HSI48 ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 HSI48 振荡器稳定所引起的中断。

0：禁止 HSI48 就绪中断

1：使能 HSI48 就绪中断

位 5 **MSIRDYIE**：MSI 就绪中断标志 (MSI ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 MSI 振荡器稳定所引起的中断。

0：禁止 MSI 就绪中断

1：使能 MSI 就绪中断

位 4 **PLLRDYIE**：PLL 就绪中断标志 (PLL ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 PLL 锁定引起的中断。

0：禁止 PLL 锁定中断

1：使能 PLL 锁定中断

位 3 **HSERDYIE**：HSE 就绪中断标志 (HSE ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 HSE 振荡器稳定所引起的中断。

0：禁止 HSE 就绪中断

1：使能 HSE 就绪中断

位 2 **HSI16RDYIE**：HSI16 就绪中断标志 (HSI16 ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 HSI16 振荡器稳定所引起的中断。

0：禁止 HSI16 就绪中断

1：使能 HSI16 就绪中断

位 1 **LSERDYIE**：LSE 就绪中断标志 (LSE ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 LSE 振荡器稳定所引起的中断。

0：禁止 LSE 就绪中断

1：使能 LSE 就绪中断

位 0 **LSIRDYIE**：LSI 就绪中断标志 (LSI ready interrupt flag)

该位由软件置 1 和复位，用于使能/禁止由 LSI 振荡器稳定所引起的中断。

0：禁止 LSI 就绪中断

1：使能 LSI 就绪中断

7.3.6 时钟中断标志寄存器 (RCC_CIFR)

Clock interrupt flag register

地址：0x14

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSS HSEF	CSS LSEF	HSI48 RDYF	MSI RDYF	PLL RDYF	HSE RDYF	HSI16 RDYF	LSE RDYF	LSI RDYF
							r	r	r	r	r	r	r	r	r

位 31:9 保留，必须保持复位值。

位 8 **CSSHSEF**：时钟安全系统中断标志 (Clock Security System Interrupt flag)

该位通过软件写入 CSSHSEC 位复位。如果 HSE 时钟出现故障，则由硬件置 1。

0：当前未因 HSE 时钟故障而引起时钟安全中断

1：因 HSE 时钟故障而引起时钟安全中断

位 7 **CSSLSEF**：LSE 时钟安全系统中断标志 (LSE Clock Security System Interrupt flag)

该位通过软件写入 CSSLSEC 位复位。如果 LSE 时钟出现故障且 CSSLSE 置 1，则由硬件置 1。

0：未检测到 LSE 时钟故障

1：检测到 LSE 时钟故障

位 6 **HSI48RDYF**：HSI48 就绪中断标志 (HSI48 ready interrupt flag)

该位通过软件写入 HSI48RDYC 位复位。当 CSS 稳定且 HSI48RDYIE 置 1 时，该位由硬件置 1。

0：当前未因 HSI48 时钟故障而引起时钟就绪中断

1：因 HSI48 时钟故障而引起时钟就绪中断

位 5 **MSIRDYF**：MSI 就绪中断标志 (MSI ready interrupt flag)

该位通过软件写入 MSIRDYC 位复位。当 MSI 时钟稳定且 MSIRDYIE 置 1 时，该位由硬件置 1。

0：当前未因 MSI 时钟故障而引起时钟就绪中断

1：因 MSI 时钟故障而引起时钟就绪中断

位 4 **PLLRDYF**：PLL 就绪中断标志 (PLL ready interrupt flag)

该位通过软件写入 PLLRDYC 位复位。当 PLL 时钟稳定且 PLLRDYIE 置 1 时，该位由硬件置 1。

0：当前未因 PLL 时钟故障而引起时钟就绪中断

1：因 PLL 时钟故障而引起时钟就绪中断

位 3 **HSERDYF**：HSE 就绪中断标志 (HSE ready interrupt flag)

该位通过软件写入 HSERDYC 位复位。当 HSE 时钟稳定且 HSERDYIE 置 1 时，该位由硬件置 1。

0：当前未因 HSE 时钟故障而引起时钟就绪中断

1：因 HSE 时钟故障而引起时钟就绪中断

位 2 **HSI16RDYF**：HSI16 就绪中断标志 (HSI16 ready interrupt flag)

该位通过软件写入 HSI16RDYC 位复位。当 HSE 时钟稳定且 HSI16RDYIE 置 1 时，该位由硬件置 1。

0：当前未因 HSI16 时钟故障而引起时钟就绪中断

1：因 HSI16 时钟故障而引起时钟就绪中断

位 1 **LSERDYF**：LSE 就绪中断标志 (LSE ready interrupt flag)

该位通过软件写入 LSERDYC 位复位。当 LSE 时钟稳定且 LSERDYIE 置 1 时，该位由硬件置 1。

0：当前未因 LSE 时钟故障而引起时钟就绪中断

1：因 LSE 时钟故障而引起时钟就绪中断

位 0 **LSIRDYF**：LSI 就绪中断标志 (LSI ready interrupt flag)

该位通过软件写入 LSIRDYC 位复位。当 LSI 时钟稳定且 LSIRDYIE 置 1 时，该位由硬件置 1。

0：当前未因 LSI 时钟故障而引起时钟就绪中断

1：因 LSI 时钟故障而引起时钟就绪中断

7.3.7 时钟中断清零寄存器 (RCC_CICR)

Clock interrupt clear register

地址：0x18

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSS HSEC	CSS LSEC	HSI48 RDYC	MSI RDYC	PLL RDYC	HSE RDYC	HSI16 RDYC	LSE RDYIC	LSI RDYC
							w	w	w	w	w	w	w	w	w

位 31:9 保留，必须保持复位值。

位 8 **CSSHSEC**：时钟安全系统中断清零 (Clock Security System Interrupt clear)

该位由软件置 1，用于将 CSSHSEF 标志清零。该位由硬件复位。

0：不起作用

1：将 CSSHSEF 标志清零

位 7 **CSSLSEC**：LSE 时钟安全系统中断清零 (LSE Clock Security System Interrupt clear)

该位由软件置 1，用于将 CSSLSEF 标志清零。该位由硬件复位。

0：不起作用

1：将 CSSLSEF 标志清零

位 6 **HSI48RDYC**：HSI48 就绪中断清零 (HSI48 ready Interrupt clear)

该位由软件置 1，用于将 HSI48RDYF 标志清零。该位由硬件复位。

0：不起作用

1：将 HSI48RDYF 标志清零

位 5 **MSIRDYC**：MSI 就绪中断清零 (MSI ready Interrupt clear)
该位由软件置 1，用于将 MSIRDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 MSIRDYF 标志清零

位 4 **PLLRDYC**：PLL 就绪中断清零 (PLL ready Interrupt clear)
该位由软件置 1，用于将 PLLRDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 PLLRDYF 标志清零

位 3 **HSERDYC**：HSE 就绪中断清零 (HSE ready Interrupt clear)
该位由软件置 1，用于将 HSERDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 HSERDYF 标志清零

位 2 **HSI16RDYC**：HSI16 就绪中断清零 (HSI16 ready Interrupt clear)
该位由软件置 1，用于将 HSI16RDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 HSI16RDYF 标志清零

位 1 **LSERDYC**：LSE 就绪中断清零 (LSE ready Interrupt clear)
该位由软件置 1，用于将 LSERDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 LSERDYF 标志清零

位 0 **LSIRDYC**：LSI 就绪中断清零 (LSI ready Interrupt clear)
该位由软件置 1，用于将 LSIRDYF 标志清零。该位由硬件复位。
0：不起作用
1：将 LSIRDYF 标志清零

7.3.8 GPIO 复位寄存器 (RCC_IOPRSTR)

GPIO reset register

地址：0x1C

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPH RST	Res.	Res.	IOPER ST	IOPD RST	IOPC RST	IOPB RST	IOPA RST
								rw			rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7 **IOPHRST**：I/O 端口 H 复位 (I/O port H reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 H

位 6:5 保留，必须保持复位值。

位 4 **IOPERST**：I/O 端口 E 复位 (I/O port E reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 E

位 3 **IOPDRST**：I/O 端口 D 复位 (IO port D reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 D

位 2 **IOPCRST**：I/O 端口 C 复位 (IO port C reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 C

位 1 **IOPBRST**：I/O 端口 B 复位 (I/O port B reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 B

位 0 **IOPARST**：I/O 端口 A 复位 (I/O port A reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I/O 端口 A

7.3.9 AHB 外设复位寄存器 (RCC_AHBRSTR)

AHB peripheral reset register

偏移地址：0x20

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPRST	Res.	Res.	Res.	RNGRST	Res.	Res.	Res.	TSCRST
							rw				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	MIFRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMARST
			rw				rw								rw

位 31:25 保留，必须保持复位值。

位 24 **CRYPTRST**：加密模块复位 (Crypto module reset)
该位由软件置 1 和复位。
0：不起作用
1：复位 CRYPTO 模块

位 23:21 保留，必须保持复位值。

位 20 **RNGRST**：随机数发生器模块复位 (Random Number Generator module reset)
该位由软件置 1 和复位。
0：不起作用
1：复位 RNG 模块

位 19:17 保留，必须保持复位值。

位 16 **TSCRST**：触摸感应复位 (Touch Sensing reset)
该位由软件置 1 和复位。
0：不起作用
1：复位触摸感应模块

位 15:13 保留，必须保持复位值。

位 12 **CRCRST**：测试集成模块复位 (Test integration module reset)
该位由软件置 1 和复位。
0：不起作用
1：复位测试集成模块

位 11:9 保留，必须保持复位值。

位 8 **MIFRST**：存储器接口复位 (Memory interface reset)
该位由软件置 1 和复位。
仅当 E2 处于 I_{DDQ} 模式时，才能激活此复位。
0：不起作用
1：复位存储器接口

位 7:1 保留，必须保持复位值。

位 0 **DMARST**：DMA 复位 (DMA reset)
该位由软件置 1 和复位。
0：不起作用
1：复位 DMA

7.3.10 APB2 外设复位寄存器 (RCC_APB2RSTR)

APB2 peripheral reset register

偏移地址：0x24

复位值：0x00000 0000

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	Res.	Res.	ADC RST	Res.	Res.	Res.	TIM22 RST	Res.	Res.	TIM21 RST	Res.	SYSCF GRST
	rw		rw			rw				rw			rw		rw

位 31:23 保留，必须保持复位值。

位 22 **DBGRST**：DBG 复位 (DBG reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 DBG

位 21:15 保留，必须保持复位值。

位 14 **USART1RST**：USART1 复位 (USART1 reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 USART1

位 13 保留，必须保持复位值。

位 12 **SPI1RST**：SPI 1 复位 (SPI 1 reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 SPI 1

位 11:10 保留，必须保持复位值。

位 9 **ADCRST**：ADC 接口复位 (ADC interface reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 ADC 接口

位 8:6 保留，必须保持复位值。

位 5 **TIM22RST**：TIM22 定时器复位 (TIM22 timer reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 TIM22 定时器

位 4:3 保留，必须保持复位值。

位 2 **TIM21RST**：TIM21 定时器复位 (TIM21 timer reset)

此位由软件置 1 和清零。

0：不起作用

1：复位 TIM21 定时器

位 1 保留，必须保持复位值。

位 0 **SYSCFGRST**：系统配置控制器复位 (System configuration controller reset)

此位由软件置 1 和清零。

0：不起作用

1：复位系统配置控制器

7.3.11 APB1 外设复位寄存器 (RCC_APB1RSTR)

APB1 peripheral reset register

偏移地址：0x28

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 RST	I2C3 RST	DACR ST	PWR RST	CRS RST	Res.	Res.	Res.	USBRST	I2C2 RST	I2C1 RST	USART5 RST	USART4 RST	LPUART1 RST	USART2 RST	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 RST	Res.	Res.	WWDG RST	Res.	LCDRST	Res.	Res.	Res.	TIM7 RST	TIM6 RST	Res.	Res.	TIM3 RST	TIM2 RST
	rw			rw		rw				rw	rw			rw	rw

位 31 **LPTIM1RST**：低功耗定时器复位 (Low-power timer reset)
此位由软件置 1 和清零。
0：不起作用
1：复位低功耗定时器

位 30 **I2C3RST**：I2C3 复位 (I2C3 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I2C3

位 29 **DACRST**：DAC 接口复位 (DAC interface reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 DAC 接口

位 28 **PWRRST**：电源接口复位 (Power interface reset)
此位由软件置 1 和清零。
0：不起作用
1：复位电源接口

位 27 **CRSRST**：时钟恢复系统复位 (Clock recovery system reset)
此位由软件置 1 和清零。
0：不起作用
1：复位时钟恢复系统

位 26:24 保留，必须保持复位值。

位 23 **USBRST**：USB 复位 (USB reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 USB

位 22 **I2C2RST**：I2C2 复位 (I2C2 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I2C2

位 21 **I2C1RST**：I2C1 复位 (I2C1 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 I2C1

位 20 **USART5RST**：USART5 复位 (USART5 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 USART5

位 19 **USART4RST**：UART4 复位 (UART4 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 USART4

位 18 **LPUART1RST**：LPUART1 复位 (USART1 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 LPUART1

位 17 **USART2RST**：USART2 复位 (USART2 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 USART2

位 16:15 保留，必须保持复位值。

位 14 **SPI2RST**：SPI2 复位 (SPI2 reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 SPI2

位 13:12 保留，必须保持复位值。

位 11 **WWDGRST**：窗口看门狗复位 (Window watchdog reset)
此位由软件置 1 和清零。
0：不起作用
1：复位窗口看门狗

位 10 保留，必须保持复位值。

位 9 **LCDRST**：LCD 复位 (LCD reset)
此位由软件置 1 和清零。
0：不起作用
1：复位 LCD

位 8:6 保留，必须保持复位值。

位 5 **TIM7RST**：定时器 7 复位 (Timer 7 reset)
由软件置 1 和清零。
0：不起作用
1：复位定时器 7

位 4 **TIM6RST**：定时器 6 复位 (Timer 6 reset)
由软件置 1 和清零。
0：不起作用
1：复位定时器 6

位 3:2 保留，必须保持复位值。

位 1 **TIM3RST**：定时器 3 复位 (Timer 3 reset)
由软件置 1 和清零。
0：不起作用
1：复位定时器 3

位 0 **TIM2RST**：定时器 2 复位 (Timer 2 reset)
由软件置 1 和清零。
0：不起作用
1：复位定时器 2

7.3.12 GPIO 时钟使能寄存器 (RCC_IOPENR)

GPIO clock enable register

地址：0x2C

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPHEN	Res.	Res.	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN
								rw			rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7 **IOPHEN**：I/O 端口 H 时钟使能位 (I/O port H clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 H 时钟
1：使能端口 H 时钟

位 6:45 保留，必须保持复位值。

位 4 **IOPEEN**：I/O 端口 E 时钟使能位 (IO port E clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 E 时钟
1：使能端口 E 时钟

位 3 **IOPDEN**：I/O 端口 D 时钟使能位 (I/O port D clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 D 时钟
1：使能端口 D 时钟

位 2 **IOPCEN**：IO 端口 C 时钟使能位 (IO port C clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 C 时钟
1：使能端口 C 时钟

位 1 **IOPBEN**：IO 端口 B 时钟使能位 (IO port B clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 B 时钟
1：使能端口 B 时钟

位 0 **IOPAEN**：IO 端口 A 时钟使能位 (IO port A clock enable bit)
此位由软件置 1 和清零。
0：禁止端口 A 时钟
1：使能端口 A 时钟

7.3.13 AHB 外设时钟使能寄存器 (RCC_AHBENR)

AHB peripheral clock enable register

偏移地址：0x30

复位值：0x0000 0100

访问：无等待周期，按字、半字和字节访问

当外设时钟未激活时，外设寄存器值不能通过软件读取，返回值始终为 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPEN	Res.	Res.	Res.	RNGEN	Res.	Res.	Res.	TSCEN
							rw				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	MIFEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAEN
			rw				rw								rw

位 31:25 保留，必须保持复位值。

位 24 **CRYPEN**：加密时钟使能位 (Crypto clock enable bit)
该位由软件置 1 和复位。
0：禁止加密时钟
1：使能加密时钟

位 23:21 保留，必须保持复位值。

位 20 **RNGEN**：随机数发生器时钟使能位 (Random Number Generator clock enable bit)
该位由软件置 1 和复位。
0：禁止 RNG 时钟
1：使能 RNG 时钟

位 19:17 保留，必须保持复位值。

位 16 **TSCEN**：触摸感应时钟使能位 (Touch Sensing Clock enable bit)
该位由软件置 1 和复位。
0：禁止触摸感应时钟
1：使能触摸感应时钟

位 15:13 保留，必须保持复位值。

位 12 **CRCEN**：CRC 时钟使能位 (CRC clock enable bit)
该位由软件置 1 和复位。
0：禁止测试集成模块时钟
1：使能测试集成模块时钟

位 11:9 保留，必须保持复位值。

位 8 **MIFEN**：NVM 接口时钟使能位 (DAC interface clock enable bit)
该位由软件置 1 和复位。
仅当 NVM 处于掉电模式时，才可激活此复位。
0：禁止 NVM 接口时钟
1：使能 NVM 接口时钟

位 7:1 保留，必须保持复位值。

位 0 **DMAEN**：DMA 时钟使能位 (DMA clock enable bit)
该位由软件置 1 和复位。
0：禁止 DMA 时钟
1：使能 DMA 时钟

7.3.14 APB2 外设时钟使能寄存器 (RCC_APB2ENR)

APB2 peripheral clock enable register

地址：0x34

复位值：0x0000 0000

访问：按字、半字和字节访问。

无等待周期，正在访问 APB2 域中的外设时请求访问的情况除外。在这种情况下，会插入等待周期，直到对 APB2 外设的访问结束。

注：当外设时钟未激活时，外设寄存器值不能通过软件读取，返回值始终为 0x0。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI1 EN	Res.	Res.	ADC EN	Res.	FWEN	Res.	TIM22 EN	Res.	Res.	TIM21 EN	Res.	SYSCF EN
	rw		rw			rw		rs		rw			rw		rw

位 31:23 保留，必须保持复位值。

位 22 **DBGEN**：DBG 时钟使能位 (CAN clock enable bit)
此位由软件置 1 和清零。
0：禁止 DBG 时钟
1：使能 DBG 时钟

位 21:15 保留，必须保持复位值。

位 14 **USART1EN**：USART1 时钟使能位 (USART1 clock enable bit)
此位由软件置 1 和清零。
0：禁止 USART1 时钟
1：使能 USART1 时钟

位 13 保留，必须保持复位值。

位 12 **SPI1EN**：SPI1 时钟使能位 (SPI1 clock enable bit)
此位由软件置 1 和清零。
0：禁止 SPI1 时钟
1：使能 SPI1 时钟

位 11:10 保留，必须保持复位值。

位 9 **ADCEN**：ADC 时钟使能位 (ADC clock enable bit)
此位由软件置 1 和清零。
0：禁止 ADC 时钟
1：使能 ADC 时钟

位 8 保留，必须保持复位值。

位 7 **FWEN**：防火墙时钟使能位 (Firewall clock enable bit)
该位通过软件置 1，通过硬件复位。软件只能将该位编程为 1。写入 0 不起作用。
0：禁止防火墙时钟
1：使能防火墙时钟

位 6 保留，必须保持复位值。

位 5 **TIM22EN**：TIM22 定时器时钟使能位 (TIM22 timer clock enable bit)
此位由软件置 1 和清零。
0：禁止 TIM22 时钟
1：使能 TIM22 时钟

位 4:3 保留，必须保持复位值。

位 2 **TIM21EN**：TIM21 定时器时钟使能位 (TIM21 timer clock enable bit)
此位由软件置 1 和清零。
0：禁止 TIM21 时钟
1：使能 TIM21 时钟

位 1 保留，必须保持复位值。

位 0 **SYSCFGEN**：系统配置控制器时钟使能位 (System configuration controller clock enable bit)
此位由软件置 1 和清零。
0：禁止系统配置控制器时钟
1：使能系统配置控制器时钟

7.3.15 APB1 外设时钟使能寄存器 (RCC_APB1ENR)

APB1 peripheral clock enable register

地址：0x38

复位值：0x0000 0000

访问：按字、半字和字节访问。

无等待周期，正在访问 APB1 域中的外设时请求访问的情况除外。在这种情况下，会插入等待周期，直到对 APB1 外设的访问结束。

注：当外设时钟未激活时，外设寄存器值不能通过软件读取，返回值始终为 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1EN	I2C3EN	DACEN	PWREN	CRSEN	Res.	Res.	Res.	USBEN	I2C2EN	I2C1EN	USART5EN	USART4EN	LPUART1EN	USART2EN	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	LCDEN	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
	rw			rw		rw				rw	rw			rw	rw

位 31 **LPTIM1EN**：低功耗定时器时钟使能位 (Low-power timer clock enable bit)
此位由软件置 1 和清零。
0：禁止低功耗定时器时钟
1：使能低功耗定时器时钟

位 30 **I2C3EN**：I2C3 时钟使能位 (I2C3 clock enable bit)
此位由软件置 1 和清零。
0：禁止 I2C3 时钟
1：使能 I2C3 时钟

位 29 **DACEN**：DAC 接口时钟使能位 (DAC interface clock enable bit)
此位由软件置 1 和清零。
0：禁止 DAC 接口时钟
1：使能 DAC 接口时钟

位 28 **PWREN**：电源接口时钟使能位 (Power interface clock enable bit)
此位由软件置 1 和清零。
0：禁止电源接口时钟
1：使能电源接口时钟

位 27 **CRSEN**：时钟恢复系统时钟使能位 (Clock recovery system clock enable bit)
此位由软件置 1 和清零。
0：禁止时钟恢复系统时钟
1：使能时钟恢复系统时钟

位 26:24 保留，必须保持复位值。

位 23 **USBEN**：USB 时钟使能位 (USB clock enable bit)
此位由软件置 1 和清零。
0：禁止 USB 时钟
1：使能 USB 时钟

位 22 **I2C2EN**：I2C2 时钟使能位 (I2C2 clock enable bit)
此位由软件置 1 和清零。
0：禁止 I2C2 时钟
1：使能 I2C2 时钟

位 21 **I2C1EN**：I2C1 时钟使能位 (I2C1 clock enable bit)
此位由软件置 1 和清零。
0：禁止 I2C1 时钟
1：使能 I2C1 时钟

位 20 **USART5EN**：USART5 时钟使能位 (USART5 clock enable bit)
此位由软件置 1 和清零。
0：禁止 USART5 时钟
1：使能 USART5 时钟

位 19 **USART4EN**：USART4 时钟使能位 (USART4 clock enable bit)
此位由软件置 1 和清零。
0：禁止 USART4 时钟
1：使能 USART4 时钟

位 18 **LPUART1EN**：LPUART1 时钟使能位 (LPUART1 clock enable bit)
此位由软件置 1 和清零。
0：禁止 LPUART1 时钟
1：使能 LPUART1 时钟

位 17 **USART2EN**：USART2 时钟使能位 (USART2 clock enable bit)
此位由软件置 1 和清零。
0：禁止 USART2 时钟
1：使能 USART2 时钟

位 16:15 保留，必须保持复位值。

位 14 **SPI2EN**：SPI2 时钟使能位 (SPI2 clock enable bit)
此位由软件置 1 和清零。
0：禁止 SPI2 时钟
1：使能 SPI2 时钟

位 13:12 保留，必须保持复位值。

位 11 **WWDGEN**：窗口看门狗时钟使能位 (Window watchdog clock enable bit)
此位由软件置 1 和清零。
0：禁止窗口看门狗时钟
1：使能窗口看门狗时钟

位 10 保留，必须保持复位值。

位 9 **LCDEN**：LCD 时钟使能位 (LCD clock enable bit)

此位由软件置 1 和清零。

0：禁止 LCD 时钟

1：使能 LCD 时钟

位 8:6 保留，必须保持复位值。

位 5 **TIM7EN**：定时器 7 时钟使能位 (Timer 7 clock enable bit)

由软件置 1 和清零。

0：禁止定时器 7 时钟

1：使能定时器 7 时钟

位 4 **TIM6EN**：定时器 6 时钟使能位 (Timer 6 clock enable bit)

由软件置 1 和清零。

0：禁止定时器 6 时钟

1：使能定时器 6 时钟

位 3:2 保留，必须保持复位值。

位 1 **TIM3EN**：定时器 3 时钟使能位 (Timer 3 clock enable bit)

由软件置 1 和清零。

0：禁止定时器 3 时钟

1：使能定时器 3 时钟

位 0 **TIM2EN**：定时器 2 时钟使能位 (Timer 2 clock enable bit)

由软件置 1 和清零。

0：禁止定时器 2 时钟

1：使能定时器 2 时钟

7.3.16 睡眠模式下的 GPIO 时钟使能寄存器 (RCC_IOPSMENR)

GPIO clock enable in Sleep mode register

地址：0x3C

复位值：可用 GPIO 端口对应的位置 1

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPHS MEN	Res.	Res.	IOPES MEN	IOPDS MEN	IOPCS MEN	IOPBS MEN	IOPAS MEN
								rw			rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7 **IOPHSMEN**：睡眠模式期间的端口 H 时钟使能位 (Port H clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 H 时钟
1：睡眠模式下使能端口 H 时钟（如果已通过 IOPHEN 使能）

位 6:5 保留，必须保持复位值。

位 4 **IOPESMEN**：睡眠模式期间的端口 E 时钟使能位 (Port E clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 E 时钟
1：睡眠模式下使能端口 E 时钟（如果已通过 IOPDEN 使能）

位 3 **IOPDSMEN**：睡眠模式期间的端口 D 时钟使能位 (Port D clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 D 时钟
1：睡眠模式下使能端口 D 时钟（如果已通过 IOPDEN 使能）

位 2 **IOPCSMEN**：睡眠模式期间的端口 C 时钟使能位 (Port C clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 C 时钟
1：睡眠模式下使能端口 C 时钟（如果已通过 IOPCEN 使能）

位 1 **IOPBSMEN**：睡眠模式期间的端口 B 时钟使能位 (Port B clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 B 时钟
1：睡眠模式下使能端口 B 时钟（如果已通过 IOPBEN 使能）

位 0 **IOPASMEN**：睡眠模式期间的端口 A 时钟使能位 (Port A clock enable during Sleep mode bit)
此位由软件置 1 和清零。
0：睡眠模式下禁止端口 A 时钟
1：睡眠模式下使能端口 A 时钟（如果已通过 IOPAEN 使能）

7.3.17 睡眠模式下的 AHB 外设时钟使能寄存器 (RCC_AHBSMENR)

AHB peripheral clock enable in Sleep mode register

register

地址：0x40

复位值：可用外设对应的位置 1

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPSMEN	Res.	Res.	Res.	RNGSMEN	Res.	Res.	Res.	TSCSMEN
							rw				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAMSMEN	MIFSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMASMEN
			rw			rw	rw								rw

位 31:25 保留，必须保持复位值。

位 24 **CRYPTSMEN**：睡眠模式期间的加密时钟使能位 (Crypto clock enable during Sleep mode bit)

该位由软件置 1 和复位。

0：睡眠模式下禁止加密时钟

1：睡眠模式下使能加密时钟

位 23:21 保留，必须保持复位值。

位 20 **RNGSMEN**：睡眠模式期间的随机数发生器时钟使能位 (Random Number Generator clock enable during Sleep mode bit)

该位由软件置 1 和复位。

0：睡眠模式下禁止 RNG 时钟

1：睡眠模式下使能 RNG 时钟（如果已通过 RNGEN 使能）

位 19:17 保留，必须保持复位值。

位 16 **TSCSMEN**：睡眠模式期间的触摸感应时钟使能位 (Touch Sensing clock enable during Sleep mode bit)

该位由软件置 1 和复位。

0：睡眠模式下禁止触摸感应时钟

1：睡眠模式下使能触摸感应时钟（如果已通过 TSCEN 使能）

位 15:13 保留，必须保持复位值。

位 12 **CRCSMEN**：睡眠模式期间的 CRC 时钟使能位 (CRC clock enable during Sleep mode bit)

该位由软件置 1 和复位。

0：睡眠模式下禁止测试集成模块时钟

1：睡眠模式下使能测试集成模块时钟（如果已通过 CRCEN 使能）

位 11:10 保留，必须保持复位值。

位 9 **SRAMSMEN**：睡眠模式期间的 SRAM 接口时钟使能位 (SRAM interface clock enable during Sleep mode bit)

该位由软件置 1 和复位。

0：睡眠模式下禁止 NVM 接口时钟

1：睡眠模式下使能 NVM 接口时钟

位 8 **MIFSMEN**：睡眠模式期间的 NVM 接口时钟使能位 (NVM interface clock enable during Sleep mode bit)

该位由软件置 1 和复位。
0：睡眠模式下禁止 NVM 接口时钟
1：睡眠模式下使能 NVM 接口时钟

位 7:1 保留，必须保持复位值。

位 0 **DMASMEN**：睡眠模式期间的 DMA 时钟使能位 (DMA clock enable during Sleep mode bit)

该位由软件置 1 和复位。
0：睡眠模式下禁止 DMA 时钟
1：睡眠模式下使能 DMA 时钟

7.3.18 睡眠模式下的 APB2 外设时钟使能寄存器 (RCC_APB2SMENR)

APB2 peripheral clock enable in Sleep mode register

地址：0x44

复位值：可用外设对应的位置 1。

访问：无等待周期，按字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 SMEN	Res.	SPI1 SMEN	Res.	Res.	ADC SMEN	Res.	Res.	Res.	TIM22 SMEN	Res.	Res.	TIM21 SMEN	Res.	SYSCF SMEN
	rw		rw			rw				rw			rw		rw

位 31:23 保留，必须保持复位值。

位 22 **DBGSMEN**：睡眠模式期间的 DBG 时钟使能位 (DBG clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 DBG 时钟
1：睡眠模式下使能 DBG 时钟（如果已通过 DBGEN 使能）

位 21:15 保留，必须保持复位值。

位 14 **USART1SMEN**：睡眠模式期间的 USART1 时钟使能位 (USART1 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 USART1 时钟
1：睡眠模式下使能 USART1 时钟（如果已通过 USART1EN 使能）

位 13 保留，必须保持复位值。

位 12 **SPI1SMEN**：睡眠模式期间的 SPI1 时钟使能位 (SPI1 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 SPI1 时钟
1：睡眠模式下使能 SPI1 时钟（如果已通过 SPI1EN 使能）

位 11:10 保留，必须保持复位值。

位 9 **ADCSMEN**：睡眠模式期间的 ADC 时钟使能位 (ADC clock enable during Sleep mode bit)

此位由软件置 1 和清零。

0：睡眠模式下禁止 ADC 时钟

1：睡眠模式下使能 ADC 时钟（如果已通过 ADCEN 使能）

位 8:6 保留，必须保持复位值。

位 5 **TIM22SMEN**：睡眠模式期间的 TIM22 定时器时钟使能位 (TIM22 timer clock enable during Sleep mode bit)

此位由软件置 1 和清零。

0：睡眠模式下禁止 TIM22 时钟

1：睡眠模式下使能 TIM22 时钟（如果已通过 TIM22EN 使能）

位 4:3 保留，必须保持复位值。

位 2 **TIM21SMEN**：睡眠模式期间的 TIM21 定时器时钟使能位 (TIM21 timer clock enable during Sleep mode bit)

此位由软件置 1 和清零。

0：睡眠模式下禁止 TIM21 时钟

1：睡眠模式下使能 TIM21 时钟（如果已通过 TIM21EN 使能）

位 1 保留，必须保持复位值。

位 0 **SYSCFGSMEN**：睡眠模式期间的系统配置控制器时钟使能位 (System configuration controller clock enable during Sleep mode bit)

此位由软件置 1 和清零。

0：睡眠模式下禁止系统配置控制器时钟

1：睡眠模式下使能系统配置控制器时钟

7.3.19 睡眠模式下的 APB1 外设时钟使能寄存器 (RCC_APB1SMENR)

APB1 peripheral clock enable in Sleep mode register

地址：0x48

复位值：可用外设对应的位置 1

注： *访问：无等待周期，按字、半字和字节访问*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 SMEN	I2C3 SMEN	DACS MEN	PWR SMEN	CRSS MEN	Res.	Res.	Res.	USBS MEN	I2C2 SMEN	I2C1 SMEN	USART5 SMEN	USART4 SMEN	LPUART1 SMEN	USART2 SMEN	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 SMEN	Res.	Res.	WWDG SMEN	Res.	LCDSM EN	Res.	Res.		TIM7 SMEN	TIM6 SMEN	Res.	Res.	TIM3 SMEN	TIM2 SMEN
	rw			rw		rw				rw	rw			rw	rw

位 31 **LPTIM1SMEN**：睡眠模式期间的低功耗定时器时钟使能位 (Low-power timer clock enable during Sleep mode bit)

此位由软件置 1 和清零。

0：睡眠模式下禁止低功耗定时器时钟

1：睡眠模式下使能低功耗定时器时钟（如果已通过 LPTIM1EN 使能）

位 30 **I2C3SMEN**：睡眠模式期间的 I2C3 时钟使能位 (I2C3 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 I2C3 时钟
1：睡眠模式下使能 I2C3 时钟（如果已通过 I2C3EN 使能）

位 29 **DACSMEN**：睡眠模式期间的 DAC 接口时钟使能位 (DAC interface clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 DAC 接口时钟
1：睡眠模式下使能 DAC 时钟（如果已通过 DACEN 使能）

位 28 **PWRSMEN**：睡眠模式期间的电源接口时钟使能位 (Power interface clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止电源接口时钟
1：睡眠模式下使能电源接口时钟（如果已通过 PWREN 使能）

位 27 **CRSSMEN**：睡眠模式期间的时钟恢复系统时钟使能位 (Clock recovery system clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止时钟恢复系统时钟
1：睡眠模式下使能时钟恢复系统时钟（如果已通过 CRSEN 使能）

位 26:24 保留，必须保持复位值。

位 23 **USBSMEN**：睡眠模式期间的 USB 时钟使能位 (USB clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 USB 时钟
1：睡眠模式下使能 USB 时钟（如果已通过 USBEN 使能）

位 22 **I2C2SMEN**：睡眠模式期间的 I2C2 时钟使能位 (I2C2 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 I2C2 时钟
1：睡眠模式下使能 I2C2 时钟（如果已通过 I2C2EN 使能）

位 21 **I2C1SMEN**：睡眠模式期间的 I2C1 时钟使能位 (I2C1 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 I2C1 时钟
1：睡眠模式下使能 I2C1 时钟（如果已通过 I2C1EN 使能）

位 20 **USART5SMEN**：睡眠模式期间的 USART5 时钟使能位 (USART5 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 USART5 时钟
1：睡眠模式下使能 USART5 时钟（如果已通过 USART5EN 使能）

位 19 **USART4SMEN**：睡眠模式期间的 USART4 时钟使能位 (USART4 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 USART4 时钟
1：睡眠模式下使能 USART4 时钟（如果已通过 USART4EN 使能）

位 18 **LPUART1SMEN**：睡眠模式期间的 LPUART1 时钟使能位 (LPUART1 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 LPUART1 时钟
1：睡眠模式下使能 LPUART1 时钟（如果已通过 LPUART1EN 使能）

位 17 **USART2SMEN**：睡眠模式期间的 USART2 时钟使能位 (USART2 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 USART2 时钟
1：睡眠模式下使能 USART2 时钟（如果已通过 USART2EN 使能）

位 16:15 保留，必须保持复位值。

位 14 **SPI2SMEN**：睡眠模式期间的 SPI2 时钟使能位 (SPI2 clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 SPI2 时钟
1：睡眠模式下使能 SPI2 时钟（如果已通过 SPI2SEN 使能）

位 13:12 保留，必须保持复位值。

位 11 **WWDGSMEN**：睡眠模式期间的窗口看门狗时钟使能位 (Window watchdog clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止窗口看门狗时钟
1：睡眠模式下使能窗口看门狗时钟（如果已通过 WWDGEN 使能）

位 10 保留，必须保持复位值。

位 9 **LCDSMEN**：睡眠模式期间的 LCD 时钟使能位 (LCD clock enable during Sleep mode bit)

此位由软件置 1 和清零。
0：睡眠模式下禁止 LCD 时钟
1：睡眠模式下使能 LCD 时钟（如果已通过 LCDEN 使能）

位 8:6 保留，必须保持复位值。

位 5 **TIM7SMEN**：睡眠模式期间的定时器 7 时钟使能位 (Timer 7 clock enable during Sleep mode bit)

由软件置 1 和清零。
0：睡眠模式下禁止定时器 7 时钟
1：睡眠模式下使能定时器 7 时钟（如果已通过 TIM7EN 使能）

位 4 **TIM6SMEN**：睡眠模式期间的定时器 6 时钟使能位 (Timer 6 clock enable during Sleep mode bit)

由软件置 1 和清零。
0：睡眠模式下禁止定时器 6 时钟
1：睡眠模式下使能定时器 6 时钟（如果已通过 TIM6EN 使能）

位 3:2 保留，必须保持复位值。

位 1 **TIM3SMEN**：睡眠模式期间的定时器 3 时钟使能位 (Timer3 clock enable during Sleep mode bit)

由软件置 1 和清零。
0：睡眠模式下禁止定时器 3 时钟
1：睡眠模式下使能定时器 3 时钟（如果已通过 TIM3EN 使能）

位 0 **TIM2SMEN**：睡眠模式期间的定时器 2 时钟使能位 (Timer2 clock enable during Sleep mode bit)

由软件置 1 和清零。
0：睡眠模式下禁止定时器 2 时钟
1：睡眠模式下使能定时器 2 时钟（如果已通过 TIM2EN 使能）

7.3.20 时钟配置寄存器 (RCC_CCIPR)

Clock configuration register

地址：0x4C

复位值：0x0000 0000

访问：无等待周期，按字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	HSI48SEL	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1 SEL1	LPTIM1 SEL0	I2C3 SEL1	I2C3 SEL0
					rw							rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1 SEL1	I2C1 SEL0	LPUART1 SEL1	LPUART1 SEL0	Res.	Res.	Res.	Res.	Res.	Res.	USART2 SEL1	USART2 SEL0	USART1 SEL1	USART1 SEL0
		rw	rw	rw	rw							rw	rw	rw	rw

位 31:27 保留，必须保持复位值。

位 26 **HSI48SEL**：48 MHz HSI48 时钟源选择位 (48 MHz HSI48 clock source selection bit)
此位由软件置 1 和清零，用以选择 USB 和 RNG 的 HSI48 时钟源。
0：选择PLL USB 时钟作为 HSI48 时钟
1：选择 RC48 时钟作为 HSI48 时钟

位 25:20 保留，必须保持复位值。

位 19:18 **LPTIM1SEL**：低功耗定时器时钟源选择位 (Low-power Timer clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 LP 定时器时钟
01：选择 LSI 时钟作为 LP 定时器时钟
10：选择 HSI16 时钟作为 LP 定时器时钟
11：选择 LSE 时钟作为 LP 定时器时钟

位 17:16 **I2C3SEL**：I2C3 时钟源选择位 (I2C3 clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 I2C3 时钟
01：选择系统时钟作为 I2C3 时钟
10：选择 HSI16 时钟作为 I2C3 时钟
11：未使用

位 15:14 保留，必须保持复位值。

位 13:12 **I2C1SEL**：I2C1 时钟源选择位 (I2C1 clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 I2C1 时钟
01：选择系统时钟作为 I2C1 时钟
10：选择 HSI16 时钟作为 I2C1 时钟
11：未使用

位 11:10 **LPUART1SEL**：LPUART1 时钟源选择位 (LPUART1 clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 LPUART1 时钟
01：选择系统时钟作为 LPUART1 时钟
10：选择 HSI16 时钟作为 LPUART1 时钟
11：选择 LSE 时钟作为 LPUART1 时钟

位 9:4 保留，必须保持复位值。

位 3:2 **USART2SEL：**USART2 时钟源选择位 (USART2 clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 USART2 时钟
01：选择系统时钟作为 USART2 时钟
10：选择 HSI16 时钟作为 USART2 时钟
11：选择 LSE 时钟作为 USART2 时钟

位 1:0 **USART1SEL：**USART1 时钟源选择位 (USART1 clock source selection bits)
此位由软件置 1 和清零。
00：选择 APB 时钟作为 USART1 时钟
01：选择系统时钟作为 USART1 时钟
10：选择 HSI16 时钟作为 USART1 时钟
11：选择 LSE 时钟作为 USART1 时钟

7.3.21 控制/状态寄存器 (RCC_CSR)

Control/status register

地址：0x50

上电复位值：0x0C00 0000

访问：0 ≤ 等待周期 ≤ 3，按字、半字和字节访问

连续访问该寄存器时，则插入等待周期。

注：RCC 控制和状态寄存器 (RCC_CSR) 中的 LSEON、LSEBYP、RTCSEL、LSEDRV 和 RTCEN 位在 RTC 域中。由于这些位在复位后受写保护，因此必须将电源控制寄存器 (PWR_CR) 中的 DBP 位置 1 才能修改这些位。有关详细信息，请参见第 6.1.3 节：RTC 和 RTC 备份寄存器。只有 RTC 域复位后，这些位才能复位（请参见第 6.1.3 节）。内部或外部复位对这些位不会有任何影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	OBL RS TF	FW RSTF	RMVF	Res.	Res.	Res.	RTC RST	RTC EN	RTCSEL[1:0]	
r	r	r	r	r	r	r	r	rt_w				rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CSSLS ED	CSSLS EON	LSEDRV[1:0]		LSE BYP	LSERDY	LSEON	Res.	Res.	Res.	Res.	Res.	Res.	LSI RDY	LSION
	r	rw	rw		rw	r	rw							r	rw

位 31 **LPWRRSTF**：低功耗复位标志 (Low-power reset flag)

发生低功耗管理复位时，由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生低功耗管理复位
1：发生低功耗管理复位
有关低功耗管理复位的详细信息，请参见*低功耗管理复位*一节。

位 30 **WWDGRSTF**：窗口看门狗复位标志 (Window watchdog reset flag)

发生窗口看门狗复位时，该位由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生窗口看门狗复位
1：发生窗口看门狗复位

位 29 **IWDGRSTF**：独立看门狗复位标志 (Independent watchdog reset flag)

V_{DD} 域发生独立看门狗复位时，该位由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生看门狗复位
1：发生看门狗复位

位 28 **SFTRSTF**：软件复位标志 (Software reset flag)

发生软件复位时，该位由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生软件复位
1：发生软件复位

位 27 **PORRSTF**：POR/PDR 复位标志 (POR/PDR reset flag)

发生 POR/PDR 复位时，该位由硬件置 1。
通过写入 RMVF 位清零。
0：未发生 POR/PDR 复位
1：发生 POR/PDR 复位

位 26 **PINRSTF**：引脚复位标志 (PIN reset flag)

发生来自 NRST 引脚的复位时，该位由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生来自 NRST 引脚的复位
1：发生来自 NRST 引脚的复位

位 25 **OBLRSTF**：选项字节加载复位标志 (Options bytes loading reset flag)

发生 OBL 复位时，该位由硬件置 1。
通过写入 RMVF 位或者通过 POR 将该位清零。
0：未发生 OBL 复位
1：发生 OBL 复位

位 24 **FWRSTF**：防火墙复位标志 (Firewall reset flag)

防火墙产生复位时，该位由硬件置 1。通过写入 RMVF 位或者通过上电复位将该位清零。
0：未发生防火墙复位
1：发生了防火墙复位

位 23 **RMVF**：清除复位标志 (Remove reset flag)

该位由软件置 1，用于清除复位标志。
0：不起作用
1：清除复位标志

位 22:20 保留，必须保持复位值。

位 19 **RTCRST**：RTC 软件复位位 (RTC software reset bit)

此位由软件置 1 和清零。

0：复位未激活

1：复位 RTC 外设、其时钟源选择和备份寄存器。

位 18 **RTCEN**：RTC 时钟使能位 (RTC clock enable bit)

此位由软件置 1 和清零。

通过将 RTCRST 位置 1 或者通过 POR 将该位复位。

0：禁止 RTC 时钟

1：使能 RTC 时钟

位 17:16 **RTCSEL[1:0]**：RTC 和 LCD 时钟源选择位 (RTC and LCD clock source selection bits)

这些位由软件置 1，用于选择 RTC 的时钟源。

一旦选择 RTC 和 LCD 时钟源后，便不能进行切换，直到 RTCRST 置 1 或者发生上电复位。唯一的例外情况是选择了 LSE 振荡器时钟、LSE 时钟停止并且被 CSSHSE 检测到，此时可切换时钟。

00：无时钟

01：LSE 振荡器时钟用作 RTC/LCD 时钟

10：LSI 振荡器时钟用作 RTC/LCD 时钟

11：由可编程预分频器分频的 HSE 振荡器时钟（通过 RCC 时钟控制寄存器 (RCC_CR) 中的 RTCPRE[1:0] 位选择）用作 RTC/LCD 时钟

如果 LSE 或 LSI 用作 RTC 时钟源，RTC 会继续在停止和待机两种低功耗模式下工作，并可用作唤醒源。但是，如果使用 HSE 时钟作为 RTC 时钟源，则 RTC 不能在停止和待机低功耗模式下使用。

位 15 保留，必须保持复位值。

位 14 **CSSLSED**：LSE 上的 CSS 故障检测标志 (CSS on LSE failure detection flag)

该位由硬件置 1，用于指示外部 32 kHz 振荡器 (LSE) 上的时钟安全系统何时检测到故障。

该位通过上电复位或 RTC 软件复位（RTCRST 位）清零。

0：LSE（32 kHz 振荡器）上未检测到故障

1：LSE（32 kHz 振荡器）上检测到故障

位 13 **CSSLSEON**：LSE 上的 CSS 使能位 (CSS on LSE enable bit)

该位由软件置 1，用于使能 LSE（32 kHz 振荡器）上的时钟安全系统。

LSE 和 LSI 振荡器使能（LSEON 和 LSION 位使能）并就绪（LSERDY 和 LSIRDY 标志由硬件置 1）后，以及选择 RTCSEL 位后，必须使能 CSSLSEON。

该位一旦使能便不能禁止，检测到 LSE 故障 (CSSLSED =1) 后除外。此时，软件必须禁止 CSSLSEON 位。

该位通过上电复位和 RTC 软件复位（RTCRST 位）复位。

0：LSE（32 kHz 振荡器）上的 CSS 关闭

1：LSE（32 kHz 振荡器）上的 CSS 开启

位 12-11 **LSEDRV**：LSE 振荡器驱动能力位 (LSE oscillator Driving capability bits)

这些位由软件置 1，用于选择 LSE 振荡器的驱动能力。

这些位通过上电复位或 RTC 复位清零。一旦写入“00”后，LSEDRV 的内容便不能通过软件进行更改。

00：最低驱动

01：中低驱动

10：中高驱动

11：最高驱动

位 10 **LSEBYP**：外部低速振荡器旁路位 (External low-speed oscillator bypass bit)

此位由软件置 1 和清零，用于在调试模式下旁路振荡器。只有在禁止 LSE 振荡器后才能写入该位。

通过将 RTCRST 位置 1 或者通过 POR 将该位复位。

0：不旁路 LSE 振荡器

1：旁路 LSE 振荡器

位 9 **LSERDY**：外部低速振荡器就绪位 (External low-speed oscillator ready bit)

该位由硬件置 1 和清零，用于指示 LSE 振荡器已稳定。在将 LSEON 位清零后，LSERDY 将在 6 个 LSE 振荡器时钟周期后转为低电平。

通过将 RTCRST 位置 1 或者通过 POR 将该位复位。

0：外部 32 kHz 振荡器未就绪

1：外部 32 kHz 振荡器已就绪

位 8 **LSEON**：外部低速振荡器使能位 (External low-speed oscillator enable bit)

此位由软件置 1 和清零。

通过将 RTCRST 位置 1 或者通过 POR 将该位复位。

0：LSE 振荡器关闭

1：LSE 振荡器开启

位 7:3 保留，必须保持复位值。

位 1 **LSIRDY**：内部低速振荡器就绪位 (Internal low-speed oscillator ready bit)

该位由硬件置 1 和清零，用于指示 LSI 振荡器已稳定。在将 LSION 位清零后，LSIRDY 将在 3 个 LSI 振荡器时钟周期后转为低电平。

该位通过系统复位进行复位。

0：LSI 振荡器未就绪

1：LSI 振荡器已就绪

位 0 **LSION**：内部低速振荡器使能位 (Internal low-speed oscillator enable bit)

此位由软件置 1 和清零。

该位通过系统复位进行复位。

0：LSI 振荡器关闭

1：LSI 振荡器开启

7.3.22 RCC 寄存器映射

下表列出了 RCC 寄存器映射和复位值。

表 40. RCC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLL ON	Res.	Res.	RTC PRE [1:0]		CSSLSEON	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.	Res.	Res.	Res.	MSIRDY	MSION	Res.	Res.	HSI16OUTEN	HSI16DIVF	HSI16DIVEN	HSI16RDYF	HSI16KERON	HSI16ON
	Reset value							0	0			X	X	0	X	0	0							1	1			0	0	0	0	0	0
0x04	RCC_ICSCR	MSITRIM[7:0]								MSICAL[7:0]								MSIRANGE[2:0]			HSI16TRIM[4:0]					HSI16CAL[7:0]							
	Reset value	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	1	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x
0x08	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[7:0]								Res.	Res.	Res.	Res.	Res.	HSI48DIV6EN	HSI48RDY	HSI48ON
	Reset value																	0	0	0	0	0	0	0	0						0	0	0
0x0C	RCC_CFGR	Res.	MCOPRE [2:0]			Res.	MCOSEL [3:0]			PLL DIV [1:0]		PLLMUL[3:0]				Res.	PLLSRC	STOPWUCK	Res.	PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value		0	0	0		0	0	0	0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSLSE	HSI48RDYIE	MSIRDYIE	PLLRDYIE	HSERDYIE	HSI16RDYIE	LSERDYIE	LSIRDYIE
	Reset value																									0	0	0	0	0	0	0	0
0x14	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSHSEF	CSSLSEF	HSI48RDYF	MSIRDYF	PLLRDYF	HSERDYF	HSI16RDYF	LSERDYF	LSIRDYF
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSHSEC	CSSLSEC	HSI48RDYC	MSIRDYC	PLLRDYC	HSERDYC	HSI16RDYC	LSERDYC	LSIRDYC
	Reset value																								0	0	0	0	0	0	0	0	0
0x1C	RCC_IOPRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPHRST	Res.	Res.	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST
	Reset value																									0			0	0	0	0	0
0x20	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPRST	Res.	Res.	Res.	RNGRST	Res.	Res.	Res.	TSCRST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	MIFRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMARST
	Reset value								0				0				0				0				0								0

表 40. RCC 寄存器映射和复位值（续）

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
0x24	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	Res.	SPI1RST	Res.	Res.	ADCRST	Res.	Res.	Res.	TM12RST	Res.	Res.	TIM21RST	Res.	SYSCFGRST
	Reset value										0								0		0			0				0			0		0
0x28	RCC_APB1RSTR	LPTIM1RST	I2C3RST	DACRST	PWRRST	CRSRST	Res.	Res.	Res.	USBRST	I2C2RST	I2C1RST	USART5RST	USART4RST	LPUART1RST	USART2RST	Res.	Res.	SPI2RST	Res.	Res.	WWDRST	Res.	LCDRST	Res.	Res.	Res.	TIM7RST	TIM6RST	Res.	Res.	TIM3RST	TIM2RST
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0			0			0		0				0	0			0	0
0x2C	RCC_IOPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPHEN	Res.	Res.	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN
	Reset value																									0			0	0	0	0	0
0x30	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPEN	Res.	Res.	Res.	RNGEN	Res.	Res.	Res.	TSCEN	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	MIFEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAEN
	Reset value								0				0				0				0				1								0
0x34	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1EN	Res.	SPI1EN	Res.	Res.	ADCEN	Res.	FWEN	Res.	TIM22EN	Res.	Res.	TIM21EN	Res.	SYSCFGEN
	Reset value										0								0		0			0		0		0			0		0
0x38	RCC_APB1ENR	LPTIM1EN	I2C3EN	DACEN	PWREN	CRSEN	Res.	Res.	Res.	USBEN	I2C2EN	I2C1EN	USART5EN	USART4EN	LPUART1EN	USART2EN	Res.	Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	LCDEN	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0			0			0		0				0	0			0	0
0x3C	RCC_IOPSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOPHSMEN	Res.	Res.	IOPESMEN	IOPDSMEN	IOPCSMEN	IOPBSMEN	IOPASMEN
	Reset value																									1			1	1	1	1	1
0x40	RCC_AHBSM ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRYPSMEN	Res.	Res.	Res.	RNGSMEN	Res.	Res.	Res.	TSCSMEN	Res.	Res.	Res.	CRCSMEN	Res.	Res.	SRAMSMEN	MIFSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMASMEN
	Reset value								1				1				1				1			1	1								1
0x44	RCC_APB2SM ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SMEN	Res.	SPI1SMEN	Res.	Res.	ADCSMEN	Res.	Res.	Res.	TIM22SMEN	Res.	Res.	TIM21SMEN	Res.	SYSCFGSMEN
	Reset value										1								1		1			1				1			1		1

表 40. RCC 寄存器映射和复位值（续）

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	RCC_APB1 SMENR	LPTIM1SMEN	I2C3SMEN	DACSMEN	PWRSMEN	CRSSMEN	Res.	Res.	Res.	USBSMEN	I2C2SMEN	I2C1SMEN	USART5SMEN	USART4SMEN	LPUART1SMEN	USART2SMEN	Res.	Res.	SPI2SMEN	Res.	Res.	WWDGSMEN	Res.	LCDSMEN	Res.	Res.	Res.	TIM7SMEN	TIM6SMEN	Res.	Res.	TIM3SMEN	TIM2SMEN
	Reset value	1	1	1	1	1				1	1	1	1	1	1	1			1			1		1				1	1			1	1
0x4C	RCC_CCIPR	Res.	Res.	Res.	Res.	Res.	HSI48SEL	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SEL1	LPTIM1SEL0	I2C3SEL1	I2C3SEL0	Res.	Res.	I2C1SEL1	I2C1SEL0	LPUART1SEL1	LPUART1SEL0	Res.	Res.	Res.	Res.	Res.	Res.	USART2SEL1	USART2SEL0	USART1SEL1	USART1SEL0
	Reset value						0							0	0	0	0			0	0	0	0							0	0	0	0
0x50	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	FWRSTF	RMVF	Res.	Res.	Res.	RTCRST	RTCEN	RTC SEL [1:0]		Res.	CSSLSED	CSSLSEON	LSE DRV [1:0]		LSEBYP	LSERDY	LSEON	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0	0	0				0	0	0	0		0	0	0	0	0	0	0							0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

8 时钟恢复系统 (CRS)

8.1 简介

时钟恢复系统 (CRS) 是一个作用于内部精细粒度可调 RC 振荡器 HSI48 的高级数字控制器。CRS 提供一种十分强大的方法，可通过与可选同步信号进行比较来评估振荡器输出频率。它能够根据测得的频率误差值自动调整振荡器微调，同时还可以进行手动微调。

CRS 非常适合为 USB 外设提供精密时钟。在这种情况下，同步信号可由 USB 总线上的帧起始 (SOF) 数据包信号提供，USB 主机以 1 ms 的时间间隔精确发送该信号。

同步信号也可由 LSE 振荡器输出或外部引脚提供，或者由用户软件生成。

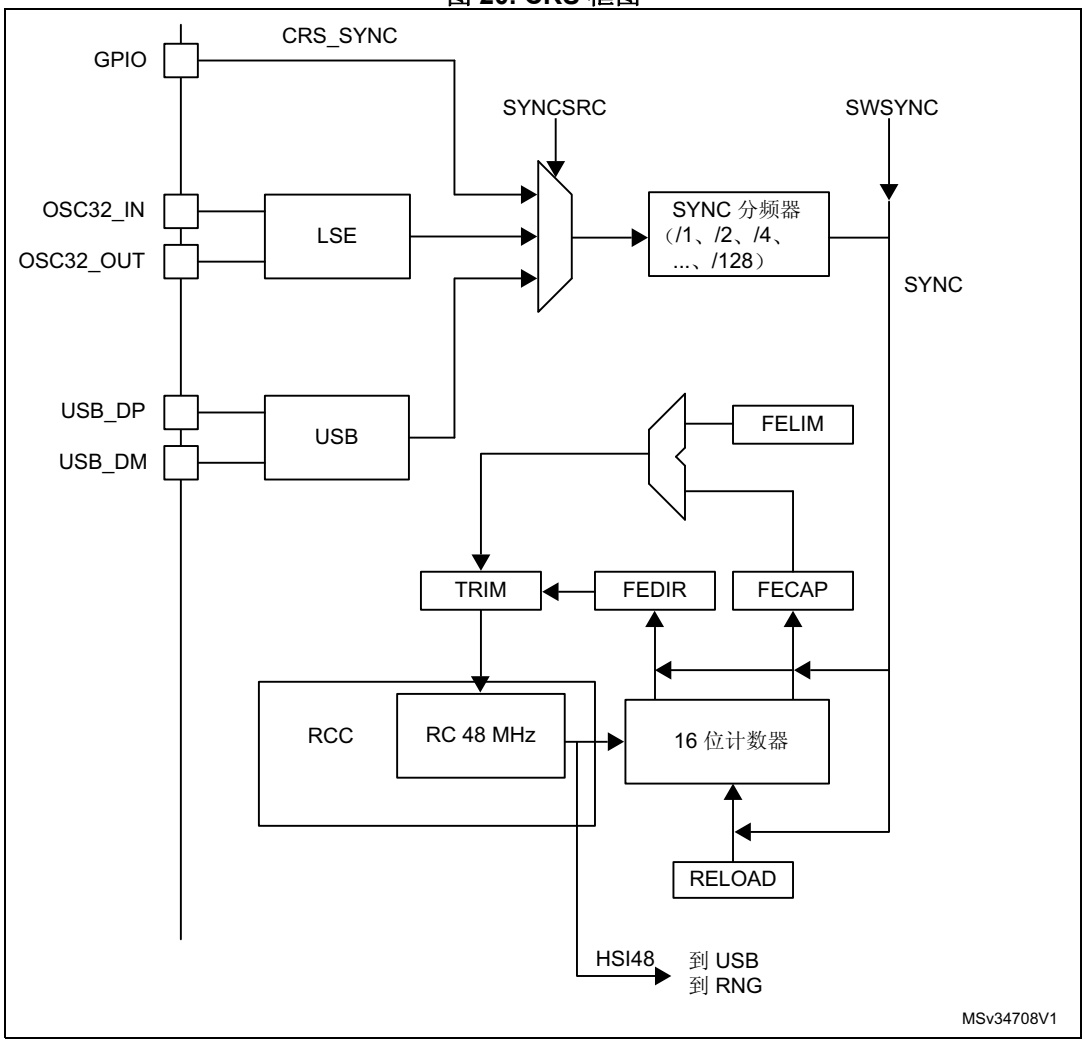
8.2 CRS 主要特性

- 具有可编程预分频器和极性的可选同步源：
  - 外部引脚
  - LSE 振荡器输出
  - USB SOF 数据包接收
- 可由软件生成同步脉冲
- 振荡器自动微调功能，无需 CPU 操作
- 手动控制选项，可加快启动融合
- 16 位频率误差计数器，用于自动误差值捕捉和重载
- 可编程限值，用于自动频率误差值评估和状态报告
- 可屏蔽中断/事件：
  - 预期同步 (ESYNC)
  - 同步正常 (SYNCOK)
  - 同步警告 (SYNCWARN)
  - 同步或微调错误 (ERR)

8.3 CRS 功能说明

8.3.1 CRS 框图

图 20. CRS 框图



8.3.2 同步输入

CRS 同步 (SYNC) 源可通过 CRS_CFGR 寄存器选择，它可以是来自外部 CRS_SYNC 引脚的信号、LSE 时钟或 USB SOF 信号。为使 SYNC 输入更稳定，可使用一个简单的数字滤波器（三分之二多数表决，通过 HSI48 时钟采样）来滤除任何干扰。该信号源还具有可配置极性，可随后由可编程二进制预分频器分频，以获得合理频率范围内的同步信号（通常为 1 kHz 左右）。

有关 CRS 同步源配置的更多信息，请参见*第 8.6.2 节：CRS 配置寄存器 (CRS_CFGR)*。

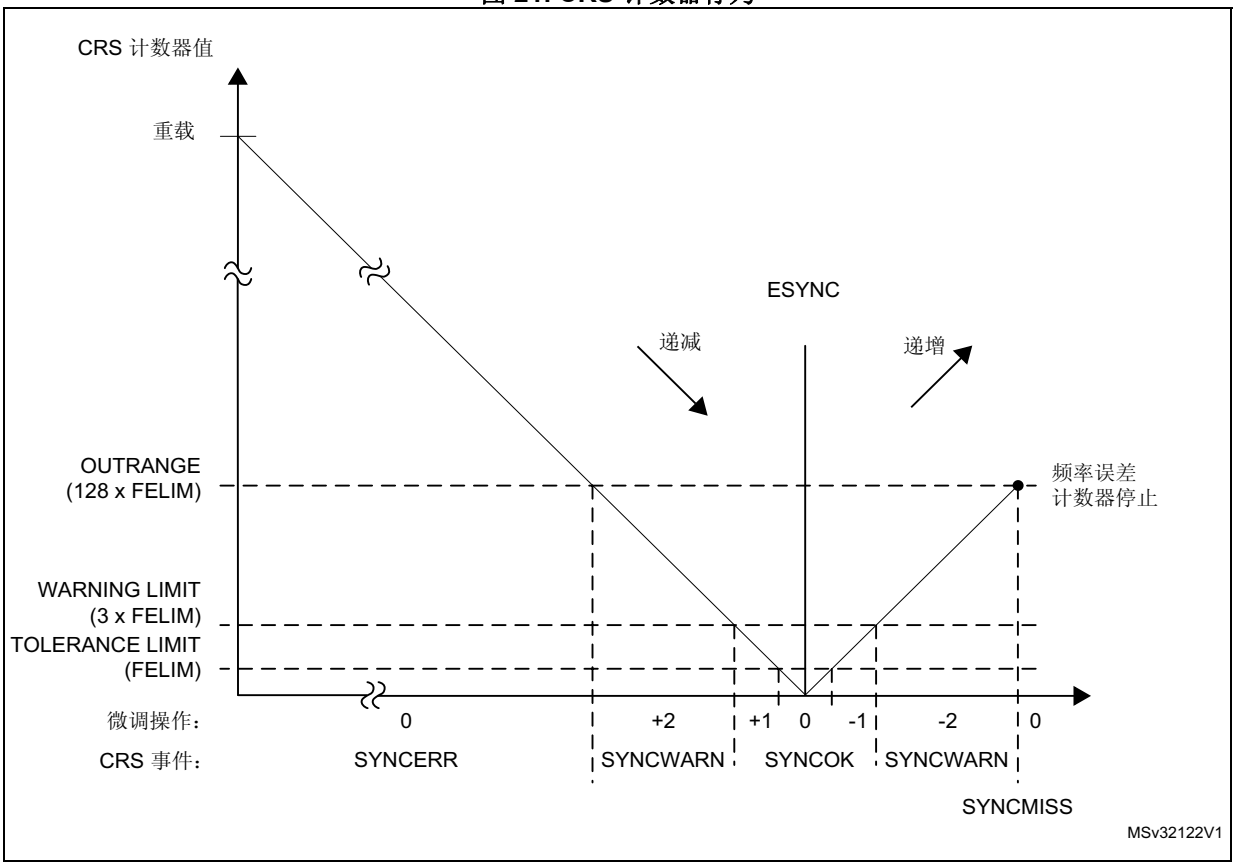
还可通过用软件将 CRS_CR 寄存器中的 SWSYNC 位置 1 的方法来生成同步事件。

8.3.3 频率误差测量

频率误差计数器是一个 16 位递减/递增计数器，会在每个 SYNC 事件发生时重载 RELOAD 值。它开始递减计数，直至达到零值，此时会生成 ESYNC（预期同步）事件。随后它将递增计数到 OUTRANGE 限值，这种情况下最终将停止计数（如果未接收到 SYNC 事件）并生成 SYNCMISS 事件。OUTRANGE 限值定义为频率误差限值（CRS_CFGR 寄存器的 FELIM 字段）乘以 128。

当检测到 SYNC 事件时，频率误差计数器的实际值及其方向存储在 CRS_ISR 寄存器的 FECAP（频率误差捕捉）字段和 FEDIR（频率误差方向）位中。当在递减计数阶段（达到零值之前）期间检测到 SYNC 事件时，意味着实际频率小于目标频率（因此，TRIM 值应递增）；当在递增计数期间检测到 SYNC 事件时，意味着实际频率大于目标频率（因此，TRIM 值应递减）。

图 21. CRS 计数器行为



8.3.4 频率误差评估和自动微调

测得的频率误差通过将频率值与一组限值进行比较进行估算：

- TOLERANCE LIMIT，在 CRS_CFGR 寄存器的 FELIM 字段中直接给出
- WARNING LIMIT，定义为 3 * FELIM 值
- OUTRANGE（误差限值），定义为 128 * FELIM 值

该比较结果用于生成状态指示以及控制自动微调，自动微调通过将 CRS_CR 寄存器中的 AUTOTRIMEN 位置 1 来使能。

- 当频率误差低于容差限值时，意味着 TRIM 字段中的实际微调值是最优值，因此无需任何微调操作。
  - 指示 SYNCOK 状态
  - AUTOTRIM 模式下的 TRIM 值无变化
- 当频率误差低于警告限值但高于或等于容差限值时，意味着需要某种微调操作，但只需一步微调便可达到最优 TRIM 值。
  - 指示 SYNCOK 状态
  - 在 AUTOTRIM 模式下通过一步微调来调整 TRIM 值
- 当频率误差高于或等于警告限值但低于误差限值时，意味着需要更强的微调操作，并且存在下一周期无法达到最优 TRIM 值的风险。
  - 指示 SYNCWARN 状态
  - 在 AUTOTRIM 模式下通过两步微调来调整 TRIM 值
- 当频率误差高于或等于误差限值时，意味着频率超出微调范围。当 SYNC 输入不干净或某个 SYNC 脉冲丢失时（例如，当一个 USB SOF 被破坏时）。也会发生这种情况。
  - 指示 SYNCERR 或 SYNCMISS 状态
  - AUTOTRIM 模式下的 TRIM 值无变化

注： *如果 TRIM 字段的实际值十分接近其限值，则自动微调会强制其上溢或下溢，之后 TRIM 值会恰好设置为限值，此时将指示 TRIMOVF 状态。*

在 AUTOTRIM 模式（CRS_CR 寄存器中的 AUTOTRIMEN 位置 1）下，CRS_CR 的 TRIM 字段由硬件调整并且是只读的。

8.3.5 CRS 初始化和配置

RELOAD 值

RELOAD 值应根据预分频后目标频率与同步源频率之比进行选择。该值随后会减 1，以在零值时达到预期同步。具体公式如下：

$RELOAD = (f_{TARGET} / f_{SYNC}) -1$

RELOAD 字段的复位值对应于 48 MHz 的目标频率和 1 kHz 的同步信号频率（来自 USB 的 SOF 信号）。

FELIM 值

FELIM 值的选择与 HSI48 振荡器的特性及其典型微调步长紧密相关。最优值对应于微调步长的一半，以 HSI48 振荡器时钟节拍数表示。可使用以下公式：

$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$

结果应始终舍入为最接近的整数值以获得最佳微调响应。如果应用中不需要频繁的微调操作，可通过稍微增大 FELIM 值来增加微调滞后。

FELIM 字段的复位值对应于 $(f_{TARGET} / f_{SYNC}) = 48000$ 以及 0.14% 的典型微调步长。

注意： 错误配置 RELOAD 和 FELIM 字段无法实现硬件保护，从而导致不定微调响应。预期工作模式需要正确设置 RELOAD 值（根据同步源频率），该值还大于 128 * FELIM 值（OUTRANGE 限值）。

8.4 CRS 低功耗模式

表 41. 低功耗模式对 CRS 的作用

模式	说明
睡眠	无影响。 CRS 中断可使器件退出睡眠模式。
停止	CRS 寄存器被冻结。 CRS 停止工作，直到退出停止或待机模式且 HSI48 振荡器重启。
待机	

8.5 CRS 中断

表 42. 中断控制位

中断事件	事件标志	使能控制位	清零标志位
预期同步	ESYNCF	ESYNCIE	ESYNCC
同步正常	SYNCOKF	SYNCOKIE	SYNCOKC
同步警告	SYNCWARNF	SYNCWARNIE	SYNCWARNC
同步或微调错误 （TRIMOVF、SYNCMISS 和 SYNCERR）	ERRF	ERRIE	ERRC

8.6 CRS 寄存器

有关寄存器说明中使用的缩写，请参见参考手册中的*第 50 页的第 1.1 节*。

外设寄存器可按字（32 位）进行访问。

8.6.1 CRS 控制寄存器 (CRS_CR)

CRS control register

偏移地址：0x00

复位值：0x0000 2000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
		rw	rw	rw	rw	rw	rw	rt_w	rw	rw		rw	rw	rw	rw

位 31:14 保留，必须保持复位值。

位 13:8 **TRIM[5:0]**：HSI48 振荡器平滑微调 (HSI48 oscillator smooth trimming)

这些位为 HSI48 振荡器提供用户可编程的微调值。可通过编程使其适应电压和温度的差异，使 HSI48 的频率更为准确。

默认值为 32，该值对应于微调间隔的中间值。连续两个 TRIM 步之间的微调步长约为 67 kHz。较高的 TRIM 值对应于较高的输出频率。

当 AUTOTRIMEN 位置 1 时，该字段由硬件控制并且是只读的。

位 7 **SWSYNC**：生成软件 SYNC 事件 (Generate software SYNC event)

此位由软件置 1 以生成软件 SYNC 事件。它由硬件自动清零。

0：不执行任何操作

1：生成软件 SYNC 事件。

位 6 **AUTOTRIMEN**：自动微调使能 (Automatic trimming enable)

此位根据在两个 SYNC 事件间测得的频率误差使能 TRIM 位的自动硬件调整。如果此位置 1，则 TRIM 位是只读的。TRIM 值可通过硬件一次调整一步或两步，具体取决于测得的频率误差值。更多详细信息，请参见*第 8.3.4 节：频率误差评估和自动微调*。

0：禁止自动微调，TRIM 位可由用户调整。

1：使能自动微调，TRIM 位是只读的且由硬件控制。

位 5 **CEN**：频率误差计数器使能 (Frequency error counter enable)

此位为频率误差计数器使能振荡器时钟。

0：禁止频率误差计数器

1：使能频率误差计数器

当此位置 1 时，CRS_CFGR 寄存器被写保护，无法修改。

位 4 保留，必须保持复位值。

位 3 **ESYNCIE**：预期 SYNC 中断使能 (Expected SYNC interrupt enable)

0：禁止预期 SYNC (ESYNCF) 中断

1：使能预期 SYNC (ESYNCF) 中断

位 2 **ERRIE**：同步或微调错误中断使能 (Synchronization or trimming error interrupt enable)
0：禁止同步或微调错误 (ERRF) 中断
1：使能同步或微调错误 (ERRF) 中断

位 1 **SYNCWARNIE**：SYNC 警告中断使能 (SYNC warning interrupt enable)
0：禁止 SYNC 警告 (SYNCWARNF) 中断
1：使能 SYNC 警告 (SYNCWARNF) 中断

位 0 **SYNCOKIE**：SYNC 事件正常中断使能 (SYNC event OK interrupt enable)
0：禁止 SYNC 事件正常 (SYNCOKF) 中断
1：使能 SYNC 事件正常 (SYNCOKF) 中断

8.6.2 CRS 配置寄存器 (CRS_CFGR)

CRS configuration register

只有禁止频率误差计数器时，才可写入该寄存器（CRS_CR 中的 CEN 位清零）。当计数器使能时，该寄存器被写保护。

偏移地址：0x04

复位值：0x2022 BB7F

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

位 31 **SYNCPOL**：SYNC 极性选择 (SYNC polarity selection)
此位由软件置 1 和清零，以选择 SYNC 信号源的输入极性。
0：SYNC 在上升沿有效（默认）
1：SYNC 在下降沿有效

位 30 保留，必须保持复位值。

位 29:28 **SYNCSRC[1:0]**：SYNC 信号源选择 (SYNC signal source selection)
这些位由软件置 1 和清零，以选择 SYNC 信号源。
00：GPIO 选择为 SYNC 信号源
01：LSE 选择为 SYNC 信号源
10：USB SOF 选择为 SYNC 信号源（默认）
11：保留

注：当使用 USB LPM（链路层电源管理）且器件处于休眠模式时，周期性 USB SOF 不会由主机产生。因此，不会向 CRS 提供任何 SYNC 信号来校准运行状态下的 HSI48。要确保从休眠模式唤醒后所需的时钟精度，GPIO 上的 LSE 或参考时钟应用作 SYNC 信号。

位 27 保留，必须保持复位值。

位 26:24 **SYNCDIV[2:0]**：SYNC 分频器 (SYNC divider)

这些位由软件置 1 和清零，以控制 SYNC 信号的分频系数。

000：SYNC 不进行分频（默认）
001：SYNC 2 分频
010：SYNC 4 分频
011：SYNC 8 分频
100：SYNC 16 分频
101：SYNC 32 分频
110：SYNC 64 分频
111：SYNC 128 分频

位 23:16 **FELIM[7:0]**：频率误差限值 (Frequency error limit)

FELIM 包含用于评估 CRS_ISR 寄存器的 FECAP[15:0] 位中锁存的已捕捉频率误差值的值。有关 FECAP 评估的更多详细信息，请参见*第 8.3.4 节：频率误差评估和自动微调*。

位 15:0 **RELOAD[15:0]**：计数器重载值 (Counter reload value)

RELOAD 是每次发生 SYNC 事件时要装载到频率误差计数器中的值。
有关计数器行为的更多详细信息，请参见*第 8.3.3 节：频率误差测量*。

8.6.3 CRS 中断和状态寄存器 (CRS_ISR)

CRS interrupt and status register

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
FECAP[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
FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
r					r	r	r					r	r	r	r

位 31:16 **FECAP[15:0]**：频率误差捕捉 (Frequency error capture)

FECAP 是最后一次 SYNC 事件发生时锁存的频率误差计数器值。
有关 FECAP 使用的更多详细信息，请参见*第 8.3.4 节：频率误差评估和自动微调*。

位 15 **FEDIR**：频率误差方向 (Frequency error direction)

FEDIR 是最后一次 SYNC 事件发生时锁存的频率误差计数器的计数方向。它显示实际频率低于目标频率还是高于目标频率。
0：递增计数方向，实际频率高于目标频率。
1：递减计数方向，实际频率低于目标频率。

位 14:11 保留，必须保持复位值。

位 10 **TRIMOVF**：微调上溢或下溢 (Trimming overflow or underflow)

当自动微调尝试使 TRIM 值上溢或下溢时，该标志由硬件置 1。如果 CRS_CR 寄存器中的 ERRIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 ERRC 位置 1。
0：未发出微调错误信号
1：已发出微调错误信号

位 9 **SYNCMISS**：SYNC 丢失 (SYNC missed)

当频率误差计数器达到值 FELIM * 128 且未检测到 SYNC 时，该标志由硬件置 1，这意味着 SYNC 脉冲丢失或频率误差过大（内部频率过高）而无法通过调整 TRIM 值来补偿，应采取其它措施。此时，频率误差计数器停止（等待下一个 SYNC），并且会在 CRS_CR 寄存器中的 ERRIE 位置 1 时生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 ERRC 位置 1。

0：未发出 SYNC 丢失错误信号

1：已发出 SYNC 丢失错误信号

位 8 **SYNCERR**：SYNC 错误 (SYNC error)

当 SYNC 脉冲在 ESYNC 事件之前达到且所测得的频率误差大于或等于 FELIM * 128 时，该标志由硬件置 1。这意味着频率误差过大（内部频率过低）而无法通过调整 TRIM 值进行补偿，应采取其它措施。如果 CRS_CR 寄存器中的 ERRIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 ERRC 位置 1。

0：未发出 SYNC 错误信号

1：已发出 SYNC 错误信号

位 7:4 保留，必须保持复位值。

位 3 **ESYNCF**：预期 SYNC 标志 (Expected SYNC flag)

当频率误差计数器达到零值时，此标志由硬件置 1。如果 CRS_CR 寄存器中的 ESYNCIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 ESYNCC 位置 1。

0：未发出预期 SYNC 信号

1：已发出预期 SYNC 信号

位 2 **ERRF**：错误标志 (Error flag)

发生同步或微调错误时，该标志由硬件置 1。它是 TRIMOVF、SYNCMISS 和 SYNCERR 的逻辑或运算的结果。如果 CRS_CR 寄存器中的 ERRIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 ERRC 位置 1，这将清零 TRIMOVF、SYNCMISS 和 SYNCERR 位。

0：未发出同步或微调错误信号

1：已发出同步或微调错误信号

位 1 **SYNCWARNF**：SYNC 警告标志 (SYNC warning flag)

当所测得的频率误差大于或等于 FELIM * 3 但小于 FELIM * 128 时，该标志由硬件置 1。这意味着要补偿频率误差，必须通过两步或多步来调整 TRIM 值。如果 CRS_CR 寄存器中的 SYNCWARNIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 SYNCWARNC 位置 1。

0：未发出 SYNC 警告信号

1：已发出 SYNC 警告信号

位 0 **SYNCOKF**：SYNC 事件正常标志 (SYNC event OK flag)

当所测得的频率误差小于 FELIM * 3 时，该标志由硬件置 1。这意味着无需调整 TRIM 值或只需一步微调便足以补偿频率误差。如果 CRS_CR 寄存器中的 SYNCOKIE 位置 1，将生成中断。此位用软件清零，方法是将 CRS_ICR 寄存器中的 SYNCOKC 位置 1。

0：未发出 SYNC 事件正常信号

1：已发出 SYNC 事件正常信号

8.6.4 CRS 中断标志清零寄存器 (CRS_ICR)

CRS interrupt flag clear register

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
												rw	rw	rw	rw

位 31:4 保留，必须保持复位值

位 3 **ESYNCC**：预期 SYNC 清零标志 (Expected SYNC clear flag)

将 1 写入此位时，CRS_ISR 寄存器中 ESYNCF 标志将清零。

位 2 **ERRC**：错误清零标志 (Error clear flag)

将 1 写入此位将清零 TRIMOVF、SYNCMISS 和 SYNCERR 位，进而会清零 CRS_ISR 寄存器中的 ERRF 标志。

位 1 **SYNCWARNC**：SYNC 警告清零标志 (SYNC warning clear flag)

将 1 写入此位时，CRS_ISR 寄存器中 SYNCWARNF 标志将清零。

位 0 **SYNCOKC**：SYNC 事件正常清零标志 (SYNC event OK clear flag)

将 1 写入此位时，CRS_ISR 寄存器中 SYNCOKF 标志将清零。

8.6.5 CRS 寄存器映射

表 43. CRS 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value																			1	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**CRS_CFGR**	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	**CRS_ISR**	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	**CRS_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

9 通用 I/O (GPIO)

9.1 简介

每个通用 I/O 端口包括 4 个 32 位配置寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR 和 GPIOx_PUPDR）、2 个 32 位数据寄存器（GPIOx_IDR 和 GPIOx_ODR）和 1 个 32 位置位/复位寄存器 (GPIOx_BSRR)。此外，所有 GPIO 都包括 1 个 32 位锁定寄存器 (GPIOx_LCKR) 和 2 个 32 位复用功能选择寄存器（GPIOx_AFRH 和 GPIOx_AFRL）。

9.2 GPIO 主要特性

- 输出状态：推挽或开漏 + 上拉/下拉
- 从输出数据寄存器 (GPIOx_ODR) 或外设（复用功能输出）输出数据
- 可为每个 I/O 选择不同的速度
- 输入状态：浮空、上拉/下拉、模拟
- 将数据输入到输入数据寄存器 (GPIOx_IDR) 或外设（复用功能输入）
- 置位和复位寄存器 (GPIOx_BSRR)，对 GPIOx_ODR 具有按位写权限
- 锁定机制 (GPIOx_LCKR)，可冻结 I/O 端口配置
- 模拟功能
- 复用功能选择寄存器
- 快速翻转，每次翻转最快只需要两个时钟周期
- 引脚复用非常灵活，允许将 I/O 引脚用作 GPIO 或多种外设功能中的一种

9.3 GPIO 功能描述

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

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

每个 I/O 端口位均可自由编程，但 I/O 端口寄存器必须按 32 位字、半字或字节进行访问。GPIOx_BSRR 寄存器和 GPIOx_BRR 寄存器旨在实现对 GPIOx_ODR 寄存器进行原子读取/修改访问。这样便可确保在读取和修改访问之间发生中断请求也不会有问题。

图 22 和 *图 23* 分别显示了标准和 5 V 容限 I/O 端口位的基本结构。*表 44* 给出了可能的端口位配置方案。

图 22. I/O 端口位的基本结构

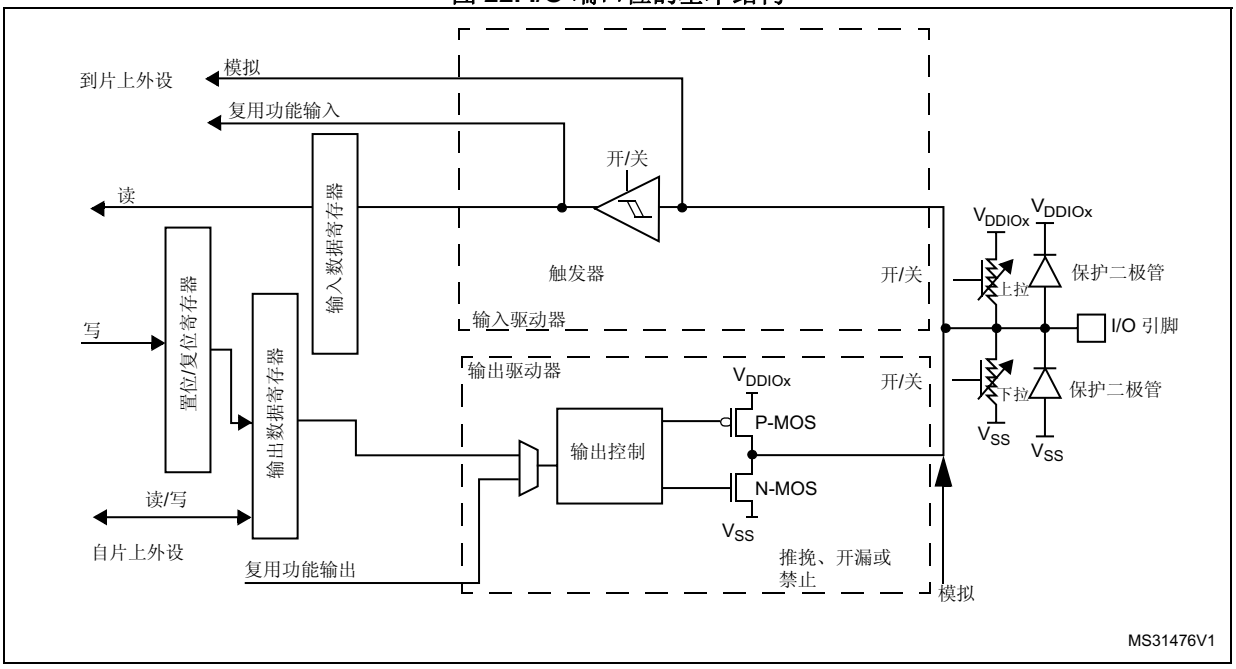
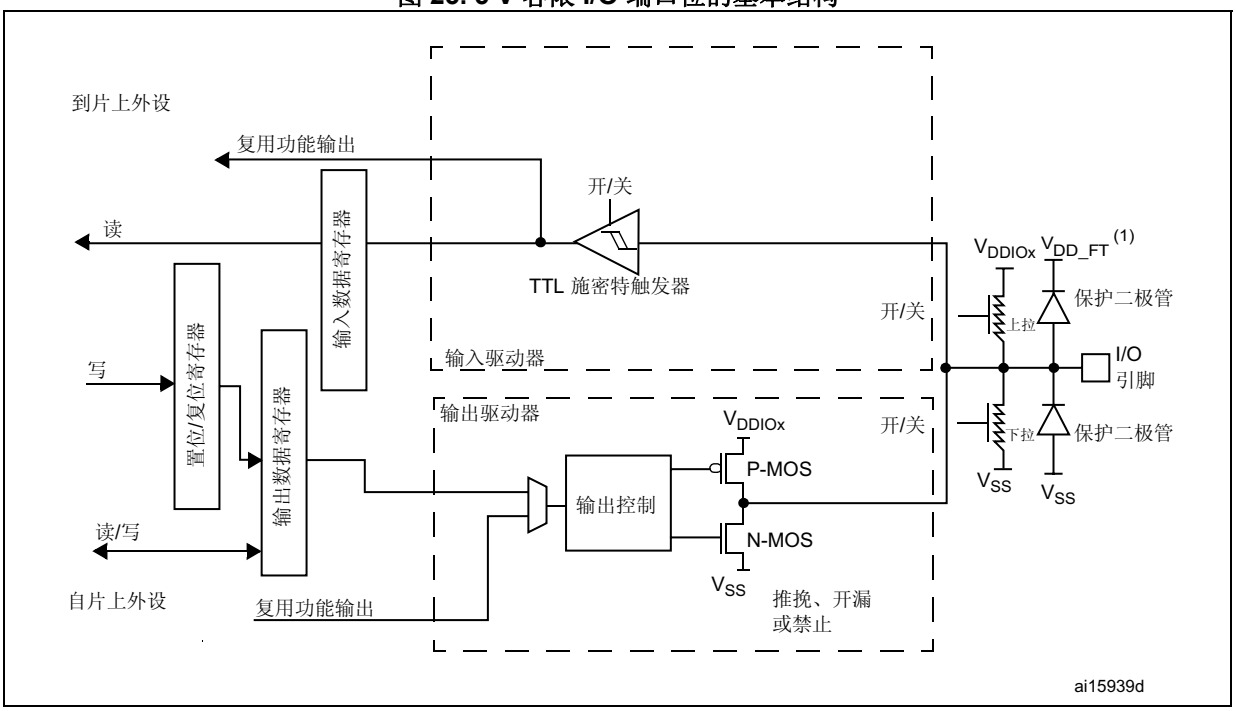


图 23. 5 V 容限 I/O 端口位的基本结构



1. V_{DD_FT} 是和 5 V 容限 I/O 相关的电位，与 V_{DD} 不同。

表 44. 端口位配置表[1]

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O 配置	
01	0	SPEED [1:0]		0	0	GP 输出	PP
	0			0	1	GP 输出	PP + PU
	0			1	0	GP 输出	PP + PD
	0			1	1	保留	
	1			0	0	GP 输出	OD
	1			0	1	GP 输出	OD + PU
	1			1	0	GP 输出	OD + PD
	1			1	1	保留（GP 输出 OD）	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	保留	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	保留	
00	x	x	x	0	0	输入	浮空
	x	x	x	0	1	输入	PU
	x	x	x	1	0	输入	PD
	x	x	x	1	1	保留（输入浮空）	
11	x	x	x	0	0	输入/输出	模拟
	x	x	x	0	1	保留	
	x	x	x	1	0		
	x	x	x	1	1		

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

9.3.1 通用 I/O (GPIO)

在复位期间及复位刚刚完成后，复用功能尚未激活，大多数 I/O 端口被配置为模拟模式。

复位后，调试引脚处于复用功能上拉/下拉状态：

- PA14：SWCLK 处于下拉状态
- PA13：SWDIO 处于上拉状态

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

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

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

9.3.2 I/O 引脚复用功能复用器和映射

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

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

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

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

要在指定配置下使用 I/O，用户必须按照以下步骤操作：

- **调试功能：**每个器件复位后，立即将这些引脚分配为可由调试主机使用的复用功能引脚
- **GPIO：**在 GPIOx_MODER 寄存器中将所需 I/O 配置为输出、输入或模拟。
- **外设复用功能：**
  - 在 GPIOx_AFRL 或 GPIOx_AFRH 寄存器中，将 I/O 连接到所需的 AFx。
  - 通过 GPIOx_OTYPER、GPIOx_PUPDR 和 GPIOx_OSPEEDER 寄存器，分别选择类型、上拉/下拉以及输出速度。
  - 在 GPIOx_MODER 寄存器中将所需 I/O 配置为复用功能。
- **其它功能：**
  - 对于 ADC、DAC 和 COMP，在 GPIOx_MODER 寄存器中将所需 I/O 配置为模拟模式，并在 ADC、DAC 和 COMP 寄存器中配置所需功能。
  - 对于 RTC、WKUPx 和振荡器的其它功能，在相关的 RTC、PWR 和 RCC 寄存器中配置所需功能。这些功能优先于标准 GPIO 寄存器中的配置。

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

9.3.3 I/O 端口控制寄存器

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

9.3.4 I/O 端口数据寄存器

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

有关寄存器说明的详细信息，请参见*第 9.4.5 节：GPIO 端口输入寄存器 (GPIOx_OTYPER)（x = A..E 和 H）*和*第 9.4.6 节：GPIO 端口输出数据寄存器 (GPIOx_ODR)（x = A..E 和 H）*。

9.3.5 I/O 数据位操作

置位复位寄存器 (GPIOx_BSRR) 是一个 32 位寄存器，它允许应用程序在输出数据寄存器 (GPIOx_ODR) 中对各个单独的数据位执行置位和复位操作。置位复位寄存器的大小是 GPIOx_ODR 的二倍。

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

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

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

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

9.3.6 GPIO 锁定机制

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

要对 GPIOx_LCKR 寄存器执行写操作，必须应用特定的写/读序列。当正确的 LOCK 序列应用到此寄存器的第 16 位后，会使用 LCKR[15:0] 的值来锁定 I/O 的配置（在写序列期间，LCKR[15:0] 的值必须相同）。将 LOCK 序列应用到某个端口位后，在执行下一次 MCU 复位或外设复位之前，将无法对该端口位的值进行修改。每个 GPIOx_LCKR 位都会冻结控制寄存器（GPIOx_MODER、GPIOx_OTYPER、GPIOx_OSPEEDR、GPIOx_PUPDR、GPIOx_AFRL 和 GPIOx_AFRH）中的对应位。

LOCK 序列（参见*第 9.4.8 节：GPIO 端口配置锁定寄存器 (GPIOx_LCKR)（x = A..E 和 H）*）只能通过对 GPIOx_LCKR 寄存器进行字（32 位长）访问的方式来执行，因为 GPIOx_LCKR 的第 16 位必须与 [15:0] 位同时置位。

有关代码示例，请参见 *A.5.1：锁定机制代码示例*。

有关详细信息，请参见*第 9.4.8 节：GPIO 端口配置锁定寄存器 (GPIOx_LCKR)（x = A..E 和 H）*中的 LCKR 寄存器说明。

9.3.7 I/O 复用功能输入/输出

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

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

要了解在每个 GPIO 引脚上复用了哪些功能，请参见器件数据手册。

有关代码示例，请参见 *A.5.2：复用功能选择序列代码示例*。

9.3.8 外部中断线/唤醒线

所有端口都具有外部中断功能。要使用外部中断线，必须将端口配置为输入模式。请参见 *第 13 节：扩展中断和事件控制器 (EXTI)* 和 *第 13.3.2 节：唤醒事件管理*。

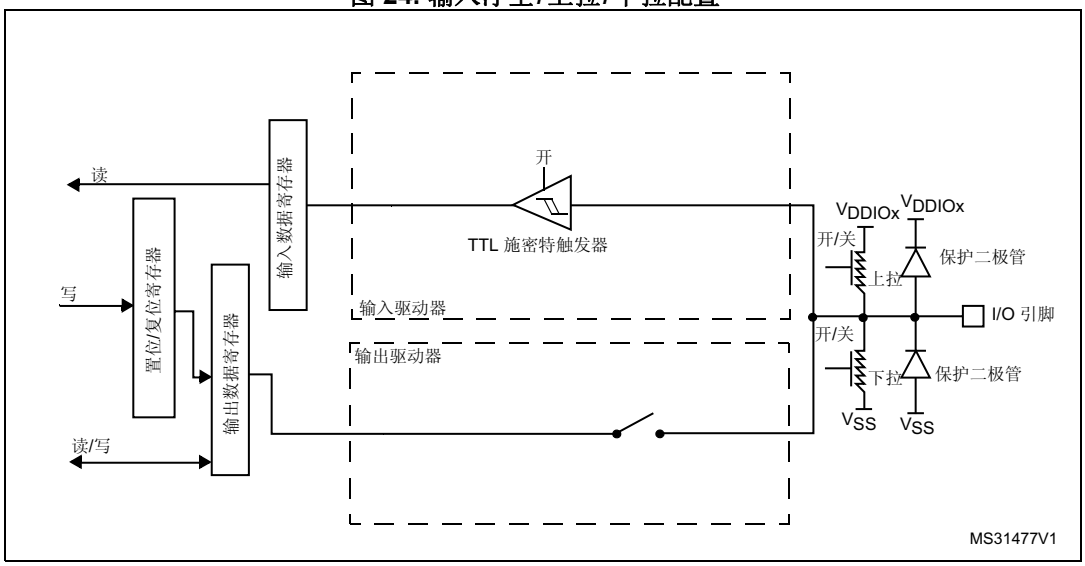
9.3.9 输入配置

对 I/O 端口进行编程作为输入时：

- 输出缓冲器被禁止
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

图 24 说明了 I/O 端口位的输入配置。

图 24. 输入浮空/上拉/下拉配置



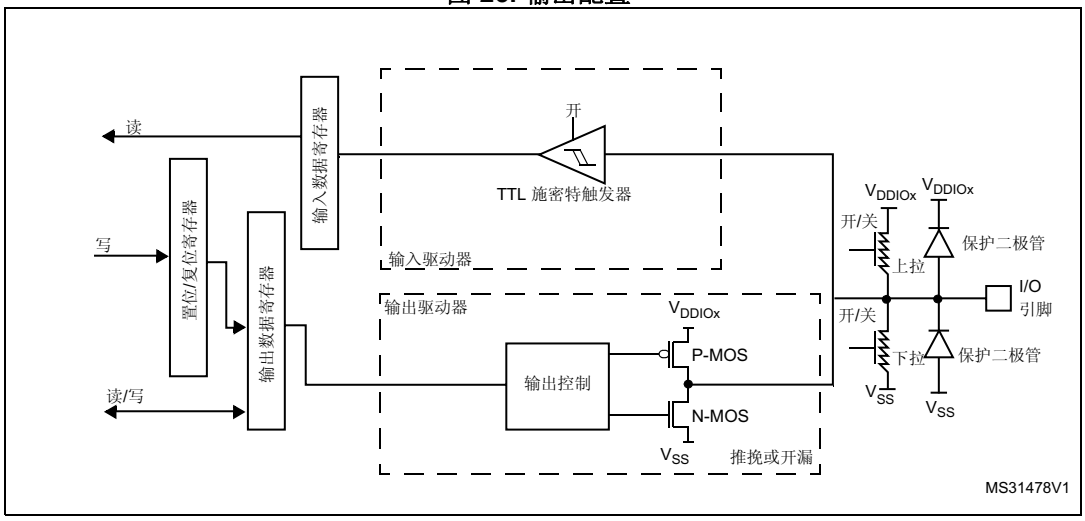
9.3.10 输出配置

对 I/O 端口进行编程作为输出时：

- 输出缓冲器被打开：
  - 开漏模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”会使端口保持高阻态 (Hi-Z)（P-MOS 始终不激活）
  - 推挽模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”可激活 P-MOS
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态
- 对输出数据寄存器的读访问可获取最后的写入值

图 25 说明了 I/O 端口位的输出配置。

图 25. 输出配置



9.3.11 复用功能配置

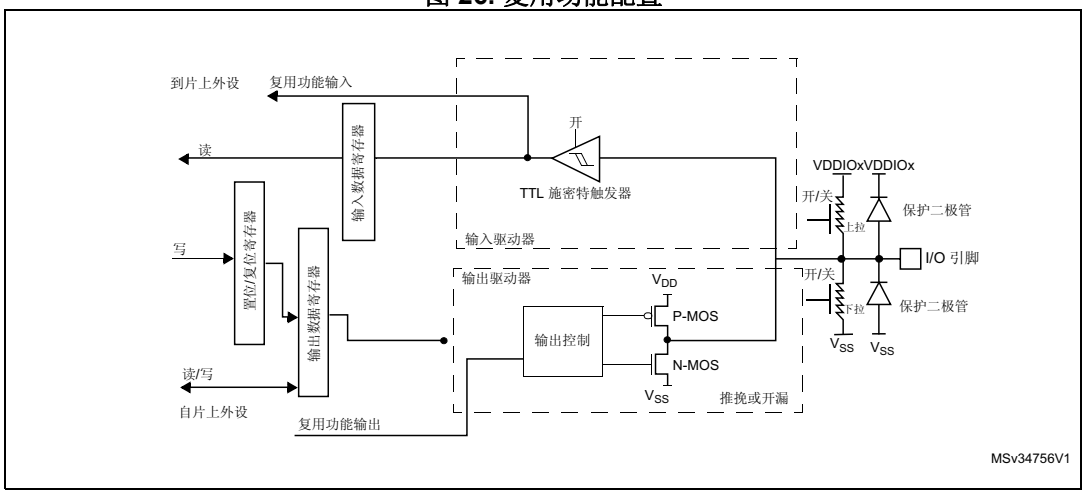
对 I/O 端口进行编程作为复用功能时：

- 可将输出缓冲器配置为开漏或推挽模式
- 输出缓冲器由来自外设的信号驱动（发送器使能和数据）
- 施密特触发器输入被打开
- 根据 GPIOx_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

注： *当所选复用功能为 LCD 功能时，不应用上述复用功能配置。在这种情况下，编程为复用功能输出的 I/O 按模拟配置中所述进行配置。*

图 26 说明了 I/O 端口位的复用功能配置。

图 26. 复用功能配置



9.3.12 模拟配置

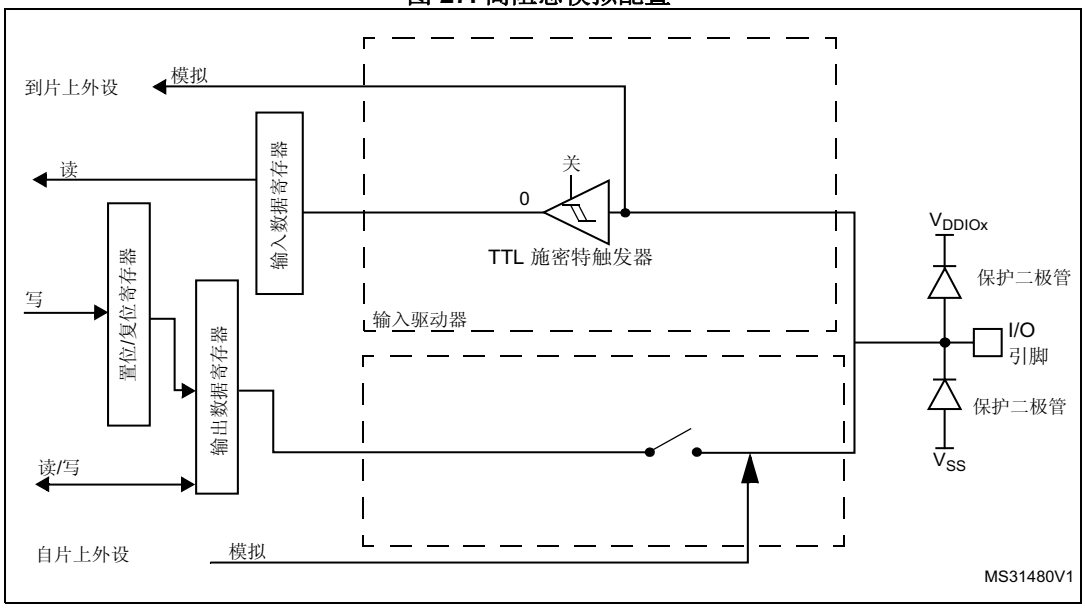
对 I/O 端口进行编程作为模拟配置时：

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

有关代码示例，请参见 *A.5.3：模拟 GPIO 配置代码示例*。

图 27 说明了 I/O 端口位的高阻态模拟输入配置。

图 27. 高阻态模拟配置



9.3.13 将 HSE 或 LSE 振荡器引脚用作 GPIO

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

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

将振荡器配置为用户外部时钟模式时，仅为时钟输入保留 OSC_IN 或 OSC32_IN 引脚，OSC_OUT 或 OSC32_OUT 引脚仍可用作常规 GPIO。

9.3.14 在 RTC 电源域中使用 GPIO 引脚

当内核电源域掉电时（器件进入待机模式时），PC13/PC14/PC15 GPIO 功能会丢失。在这种情况下，如果不被配置成 RTC 相关复用功能，则会设置成模拟输入模式。

有关 RTC 的 I/O 控制的详细信息，请参见*第 593 页上的第 27.2 节：RTC 主要特性*。

9.4 GPIO 寄存器

本节对 GPIO 寄存器进行了详细介绍。

有关寄存器位、寄存器偏移地址和复位值的汇总，请参见*表 45*。

可按字、半字或字节模式写入外设寄存器。

9.4.1 GPIO 端口模式寄存器 (GPIOx_MODER)（x =A..E 和 H）

GPIO port mode register

偏移地址：0x00

复位值：

- 0xEBFF FCFF（端口 A）
- 0xFFFF FFFF（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y+1:2y **MODEy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)

这些位通过软件写入，用于配置 I/O 模式。

00：输入模式
01：通用输出模式
10：复用功能模式
11：输入模式（复位状态）

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

GPIO port output type register

偏移地址：0x04

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **OTy**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 输出类型。
0：推挽输出（复位状态）
1：开漏输出

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

GPIO port output speed register

偏移地址：0x08

复位值：

- 0x0C00 0000（端口 A）
- 0x0000 0000（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y+1:2y **OSPEEDy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 输出速度。
00：低速
01：中速
10：高速
11：超高速

注： 关于每种速度的频率规范以及电源和负载条件，请参见器件数据手册。

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

GPIO port pull-up/pull-down register

偏移地址：0x0C

复位值：

- 0x2400 0000（端口 A）
- 0x0000 0000（其它端口）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y+1:2y **PUPDy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)
这些位通过软件写入，用于配置 I/O 上拉或下拉。
00：无上拉或下拉
01：上拉
10：下拉
11：保留

9.4.5 GPIO 端口输入寄存器 (GPIOx_OTYPER)（x = A..E 和 H）

GPIO port input data register

偏移地址：0x10

复位值：0x0000 XXXX（其中 X 表示未定义）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16 保留，必须保持复位值。

位 15:0 **IDy**：端口输入数据位 (Port input data bit) (y = 0..15)
这些位为只读。它们包含相应 I/O 端口的输入值。

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

GPIO port output data register

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:0 **ODy**：端口输出数据位 (Port output data bit) (y = 0..15)
这些位可通过软件读取和写入。
注： 对于原子置位/复位，通过写入 GPIOx_BSRR 或 GPIOx_BRR 寄存器，可分别置位和/或复位 OD 位（x = A..E 和 H）

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

GPIO port bit set/reset register

偏移地址：0x18

复位值：0x0000 0000

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

位 31:16 **BRy**：端口 x 复位位 y (Port x reset bit y) (y = 0..15)
这些位为只写。若读取这些位则返回 0x0000。
0：不会对相应的 ODx 位执行任何操作
1：复位相应的 ODx 位
注： 如果同时对 BSx 和 BRx 置位，则 BSx 的优先级更高。

位 15:0 **BSy**：端口 x 置位位 y (Port x set bit y) (y= 0..15)
这些位为只写。若读取这些位则返回 0x0000。
0：不会对相应的 ODx 位执行任何操作
1：将相应的 ODx 位置 1

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

GPIO port configuration lock register

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

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

每个锁定位冻结一个特定的配置寄存器（控制寄存器和复用功能寄存器）。

偏移地址：0x1C

复位值：0x0000 0000

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

位 31:17 保留，必须保持复位值。

位 16 LCKK：锁定键 (Lock key)

可随时读取此位。可使用锁定键写序列对其进行修改。

0：端口配置锁定键未激活

1：端口配置锁定键已激活。直到下一次 MCU 复位或外设复位时，才锁定 GPIOx_LCKR 寄存器。

锁定键写序列：

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

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

注：在锁定键写序列期间，不能更改 LCK[15:0] 的值。

锁定序列中的任何错误都将中止锁定操作。

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

位 15:0 **LCKy**：端口 x 锁定位 y (Port x lock bit y) (y= 0..15)

这些位都是读/写位，但只能在 LCKK 位等于“0”时执行写操作。

0：端口配置未锁定

1：端口配置已锁定

9.4.9 GPIO 复用功能低位寄存器 (GPIOx_AFRL)（x = A..E 和 H）

GPIO alternate function low register

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **AFSELy[3:0]**：端口 x 引脚 y 的复用功能选择 (Alternate function selection for port x pin y) (y = 0..7)

这些位通过软件写入，用于配置复用功能 I/O。

AFSELy 选择：

0000：AF0
0001：AF1
0010：AF2
0011：AF3
0100：AF4
0101：AF5
0110：AF6
0111：AF7
1000：保留
1001：保留
1010：保留
1011：保留
1100：保留
1101：保留
1110：保留
1111：保留

9.4.10 GPIO 复用功能高位寄存器 (GPIOx_AFRH)（x = A..E 和 H）

GPIO alternate function high register

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **AFSELy[3:0]**：端口 x 引脚 y 的复用功能选择 (Alternate function selection for port x pin y) (y = 8..15)

这些位通过软件写入，用于配置复用功能 I/O。

AFSELy 选择：

0000：AF0
0001：AF1
0010：AF2
0011：AF3
0100：AF4
0101：AF5
0110：AF6
0111：AF7
1000：保留
1001：保留
1010：保留
1011：保留
1100：保留
1101：保留
1110：保留
1111：保留

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

GPIO port bit reset register

偏移地址：0x28
复位值：0x0000 0000

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

位 15:0 **BRy**：端口 x 复位位 y (Port x Reset bit y) (y = 0..15)

这些位为只写。读取这些位可返回值 0x0000

0：不会对相应的 ODx 位执行任何操作
1：复位相应的 ODx 位

9.4.12 GPIO 寄存器映射

下表提供了 GPIO 寄存器映射和复位值。

表 45. GPIO 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_MODER**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
0x00	**GPIOx_MODER** (where x = **B..E, H**)	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER** (where x = **A..E,H**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOA_OSPEEDR**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_OSPEEDR** (where x = **B..E,H**)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = **A..E,H**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = **A..E,H**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = **A..E,H**)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = **A..E,H**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = **A..E,H**)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	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	**GPIOx_AFRH** (where x = **A..E,H**)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 45. GPIO 寄存器映射和复位值（续）

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
0x28	**GPIOx_BRR** (where x = **A..E,H**)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

10 系统配置控制器 (SYSCFG)

10.1 简介

器件配有一组配置寄存器。系统配置控制器的主要用途如下：

- 重映射存储器
- 将有些触发源重映射到定时器输入捕捉通道
- 管理内部边沿检测器复用的外部中断线
- 使能专用功能，例如输入捕捉复用或振荡器引脚重映射
- I2C Fm+ 模式管理
- LCD 电源轨去耦
- 防火墙管理
- 温度传感器和内部参考电压管理（包括比较器、48 MHz HSI 和 ADC 用途）。

Cortex®-M0+ 可在 *eventin* 输入信号发生转换时从 WFE（等待事件）唤醒。为支持多处理器环境中的信号量管理，内核还会在 SEV 指令执行过程中在信号输出 EVENTOUT 上输出事件。

在 STM32L0x3 器件中，事件输入可由外部中断线或 RTC 报警中断生成。此外，还可以选择哪个输出引脚与 Cortex®-M0+ 的 EVENTOUT 信号相连。EVENTOUT 复用由 GPIO 复用功能进行管理（请参见*第 9.4.9 节：GPIO 复用功能低位寄存器 (GPIOx_AFRL)（x = A..E 和 H）*和*第 9.4.10 节：GPIO 复用功能高位寄存器 (GPIOx_AFRH)（x = A..E 和 H）*）。

注： *EVENTOUT 并非映射到所有 GPIO 上（例如 PC13、PC14 和 PC15）。*

10.2 SYSCFG 寄存器

外设寄存器必须按字（32 位）进行访问。

10.2.1 SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1)

SYSCFG memory remap register

此寄存器用于与存储器重映射相关的特定配置：

注：此寄存器并非通过 RCC_APB2RSTR 寄存器中的 SYSCFGRST 位复位。

偏移地址：0x00

复位值：0x000x 000x（x 为自举配置选择的存储器模式）。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BOOT_MODE		Res.	Res.	Res.	Res.	UFB	Res.	MEM_MODE	
						r	r					rw		rw	rw

位 31:10 保留，必须保持复位值

位 9:8 **BOOT_MODE**：自举引脚选择的自举模式状态位 (Boot mode selected by the boot pins status bits)

这些位为只读。这些位指示 BOOT 引脚选择的自举模式。位 9 对应于 FLASH_OPTR 寄存器中 nBOOT1 位的补码。其值在选项字节中定义。位 8 对应于 BOOT0 引脚上采样的值（请参见*第 60 页上的第 2.4 节：自举配置*）。

00：主 Flash 自举模式

01：系统 Flash 自举模式

10：保留

11：嵌入式 SRAM 自举模式

位 7:4 保留，必须保持复位值

位 3 **UFB**：用户存储区域交换 (User bank swapping)

此位仅在第 5 类器件上可用，在其它类器件上保留。

此位由软件置 1 和清零。它用于控制存储区域 1/2 映射（请参见*表 8：UFB = 0 时的 NVM 构成（128 KB 第 5 类器件）*和*表 10：UFB = 0 时的 NVM 构成（64 KB 第 5 类器件）*）。

0：Flash 程序存储器存储区域 1 映射到 0x0800 0000（如果 MEM_MODE=00，则在 0x0000 0000 有别名），数据 EEPROM 存储区域 1 映射到 0x0808 0000（如果 MEM_MODE=00，则在 0x0008 0000 有别名）

1：Flash 程序存储器存储区域 2 映射到 0x0800 0000（如果 MEM_MODE=00，则在 0x0000 0000 有别名），数据 EEPROM 存储区域 2 映射到 0x0808 0000（如果 MEM_MODE=00，则在 0x0008 0000 有别名）

位 2 保留，必须保持复位值

位 1:0 **MEM_MODE**：存储器映射选择位 (Memory mapping selection bits）

这些位由软件置 1 和清零。此位控制在地址 0x0000 0000 的存储器内部映射。复位后，这些位将指示 BOOT 引脚选择的存储器映射（请参见*第 60 页上的第 2.4 节：自举配置*）。

00：将主 Flash 映射到地址 0x0000 0000

01：将系统 Flash 映射到地址 0x0000 0000

10：保留

11：将 SRAM 映射到地址 0x0000 0000。

10.2.2 SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)

SYSCFG peripheral mode configuration register

偏移地址：0x04

复位值：0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	LCD_CAPA[4:0]					FWDIS
	rw	rw	rw							rw	rw	rw	rw	rw	rw

位 31:15 保留，必须保持复位值

位 14 **I2C3 FMP**：I2C3 Fm+ 驱动能力使能位 (I2C3 Fm+ drive capability enable bit)
此位由软件置 1 和清零。此位置 1 时，在通过 IOPORT 控制寄存器 AF 选择位选择的 I2C3 引脚 PC0、PC1、PA8 和 PB4 上使能 Fm+ 模式。

位 13 **I2C2 FMP**：I2C2 Fm+ 驱动能力使能位 (I2C2 Fm+ drive capability enable bit)
此位由软件置 1 和清零。此位置 1 时，在通过 IOPORT 控制寄存器 AF 选择位选择的 I2C2 引脚 PB13 和 PB14 上使能 Fm+ 模式。

位 12 **I2C1 FMP**：I2C1 Fm+ 驱动能力使能位 (I2C1 Fm+ drive capability enable bit)
此位由软件置 1 和清零。此位置 1 时，在通过 IOPORT 控制寄存器 AF 选择位选择的 I2C1 引脚上使能 Fm+ 模式。此位与 I2C_PBx_FMP 位进行逻辑或运算。

位 11 **I2C PB9 FMP**：PB9 上的 Fm+ 驱动能力使能位 (Fm+ drive capability on PB9 enable bit)
此位由软件置 1 和清零。此位置 1 时，强制在 PB9 上使能 Fm+ 驱动能力。

位 10 **I2C PB8 FMP**：PB8 上的 Fm+ 驱动能力使能位 (Fm+ drive capability on PB8 enable bit)
此位由软件置 1 和清零。此位置 1 时，强制在 PB8 上使能 Fm+ 驱动能力。

位 9 **I2C PB7 FMP**：PB7 上的 Fm+ 驱动能力使能位 (Fm+ drive capability on PB7 enable bit)
此位由软件置 1 和清零。此位置 1 时，强制在 PB7 上使能 Fm+ 驱动能力。

位 8 **I2C PB6 FMP**：PB6 上的 Fm+ 驱动能力使能位 (Fm+ drive capability on PB6 enable bit)
此位由软件置 1 和清零。此位置 1 时，强制在 PB6 上使能 Fm+ 驱动能力。

位 7:4 保留，必须保持复位值

位 5:1 **LCD_CAPA[4:0]**：去耦电容连接 (Decoupling capacitance connection)（有关器件功能的详细信息，请参见数据手册）

这些位由软件置 1 和清零。它们控制内部 V_{LCD} 电源轨与专用 I/O（LCD_VLCD1、LCD_VLCD2、LCD_VLCD3）之间的连接，以执行可选去耦。

位 1 控制 VLCDrail1 与 PB2/LCD_VLCD1 的连接

0：VLCDrail1 不连接到 PB2/LCD_VLCD1

1：VLCDrail1 连接到 PB2/LCD_VLCD1

位 2 控制 VLCDrail2 与 PB12 的连接

位 3 控制 VLCDrail3 与 PB0 的连接

位 4 控制 VLCDrail2 与 PE11 的连接

位 5 控制 VLCDrail3 与 PE12 的连接

有关详细信息，请参见*外部去耦*一节。

位 0 **FWDIS**：防火墙禁止位 (Firewall disable bit)

此位默认置 1（复位后）。此位由软件清零，以根据防火墙配置对存储器段访问施加保护。清零后，此位不能通过软件置 1。只能通过系统复位将此位置 1。

0：使能防火墙访问

1：禁止防火墙访问

注： 此位不能通过 APB 复位置 1。只能通过系统复位将其置 1。

10.2.3 参考控制和状态寄存器 (SYSCFG_CFGR3)

Reference control and status register

SYSCFG_CFGR3 寄存器为参考控制/状态寄存器。此寄存器包含与 VREFINT 和温度传感器相关的所有位/标志。

偏移地址：0x20

系统复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REF_ LOCK	VREFINT _RDYF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ENREF_ HSI48	ENBUF_VR EFINT_ COMP	Res.	Res.	ENBUF_ SENSOR _ADC	ENBUF_ VREFINT _ ADC	Res.	Res.	SEL_VREF _OUT		Res.	Res.	Res.	Res.
		rw	rw			rw	rw			rw	rw				

位 31 **REF_LOCK：SYSCFG_CFGR3 锁定位**(SYSCFG_CFGR3 lock bit)

此位由软件置 1，通过硬件系统复位清零。它将锁定参考控制/状态寄存器 SYSCFG_CFGR3[31:0] 的全部内容。

0：SYSCFG_CFGR3[31:0] 位可进行读/写

1：SYSCFG_CFGR3[31:0] 位为只读

位 30 **VREFINT_RDYF**：VREFINT 就绪标志 (VREFINT ready flag)

此位为只读。它指示内部参考电压 (VREFINT) 的状态。置 1 时，它指示 VREFINT 可供 BOR、PVD 和 LCD 使用。

0：VREFINT 关闭

1：VREFINT 就绪

位 29:14 保留，必须保持复位值

位 13 **ENREF_HSI48**：HSI48 振荡器的 VREFINT 参考使能位 (VREFINT reference for HSI48 oscillator enable bit)

此位由软件置 1 和清零（REF_LOCK 置 1 时除外）。

0：用于生成 HSI48 振荡器的 VREFINT 参考的缓冲区关闭。

1：用于生成 HSI48 振荡器的 VREFINT 参考的缓冲区开启。

位 12 **ENBUF_VREFINT_COMP**：比较器 2 的 VREFINT 参考使能位 (VREFINT reference for comparator 2 enable bit)

此位由软件置 1 和清零（REF_LOCK 置 1 时除外）。

0：用于生成比较器 2 的 VREFINT 参考的缓冲区关闭。

1：用于生成比较器 2 的 VREFINT 参考的缓冲区开启。

位 11:10 保留，必须保持复位值

位 9 **ENBUF_SENSOR_ADC**：ADC 的温度传感器参考使能位 (Temperature sensor reference for ADC enable bit)

此位由软件置 1 和清零（REF_LOCK 置 1 时除外）。此位置 1 时，会自动使能 VREFINT。

0：用于生成 ADC 的温度传感器参考的缓冲区关闭。

1：用于生成 ADC 的温度传感器参考的缓冲区开启。

位 8 **ENBUF_VREFINT_ADC**：ADC 的 VREFINT 参考使能位 (VREFINT reference for ADC enable bit)

此位由软件置 1 和清零（REF_LOCK 置 1 时除外）。

0：用于生成 ADC 的 VREFINT 参考的缓冲区关闭。

1：用于生成 ADC 的 VREFINT 参考的缓冲区开启。

位 7:6 保留，必须保持复位值

位 5:4 **SEL_VREF_OUT**：VREFINT_ADC 连接位 (VREFINT_ADC connection bit)

这些位由软件置 1 和清零（REF_LOCK 置 1 时除外）。这些位选择 ENBUF_VREFINT_ADC 置 1 时将哪个引脚连接到 VREFINT_ADC。

00：不连接任何引脚

01：连接 PB0

10：连接 PB1

11：连接 PB0 和 PB1

位 3:0 保留，必须保持复位值

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

SYSCFG external interrupt configuration register 1

偏移地址：0x08

复位值：0x0000

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

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 0 到 3）(EXTI x configuration (x = 0 to 3))

这些位通过软件写入，以选择 EXTIx 外部中断的源输入。0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PH[x]（仅限 PH[1:0] 和 PH[10:9]）
其他配置保留

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

SYSCFG external interrupt configuration register 2

偏移地址：0x0C

复位值：0x0000

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

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

这些位通过软件写入，以选择 EXTIx 外部中断的源输入。
0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
其他配置保留

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

SYSCFG external interrupt configuration register 3

偏移地址：0x10

复位值：0x0000

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

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 8 到 11）(EXTI x configuration (x = 8 to 11))

这些位通过软件写入，以选择 EXTIx 外部中断的源输入。

0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
0101：PH[x]（仅限 PH[1:0] 和 PH[10:9]）
其他配置保留。

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

SYSCFG external interrupt configuration register 4

偏移地址：0x14

复位值：0x0000

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

位 15:0 **EXTIx[3:0]**：EXTI x 配置（x = 12 到 15）(EXTI x configuration (x = 12 to 15))

这些位通过软件写入，以选择 EXTIx 外部中断的源输入。

0000：PA[x] 引脚
0001：PB[x] 引脚
0010：PC[x] 引脚
0011：PD[x] 引脚
0100：PE[x] 引脚
其他配置保留。

10.2.8 SYSCFG 寄存器映射

下表列出了 SYSCFG 寄存器映射和复位值。

表 46. SYSCFG 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_MODE		Res.	Res.	Res.	Res.	UFB	Res.	MEM_MODE	
	Reset value																							x	x					x		x	x
0x04	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	CAPA[4:0]					FWDISEN
	Reset value																		0	0	0	0	0	0	0			0	0	0	0	0	1
0x08	SYSCFG_EXTICR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	COMP1_CTRL	请参见*第 16 节：比较器 (COMP)*。																															
0x1C	COMP2_CTRL																																
0x20	SYSCFG_CFGR3	REF_LOCK	VREFINT_RDYF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENREF_HSI48	ENBUF_VREFINT_COMP	Res.	Res.	ENBUF_SENSOR_ADC	ENBUF_VREFINT_ADC	Res.	Res.	SEL_VREF_OUT		Res.	Res.	Res.	Res.
	Reset value	0	0																	0	0			0	0			0	0				

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

11 直接存储器访问控制器 (DMA)

11.1 简介

直接存储器访问 (DMA) 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据。这样节省的 CPU 资源可供其他操作使用。

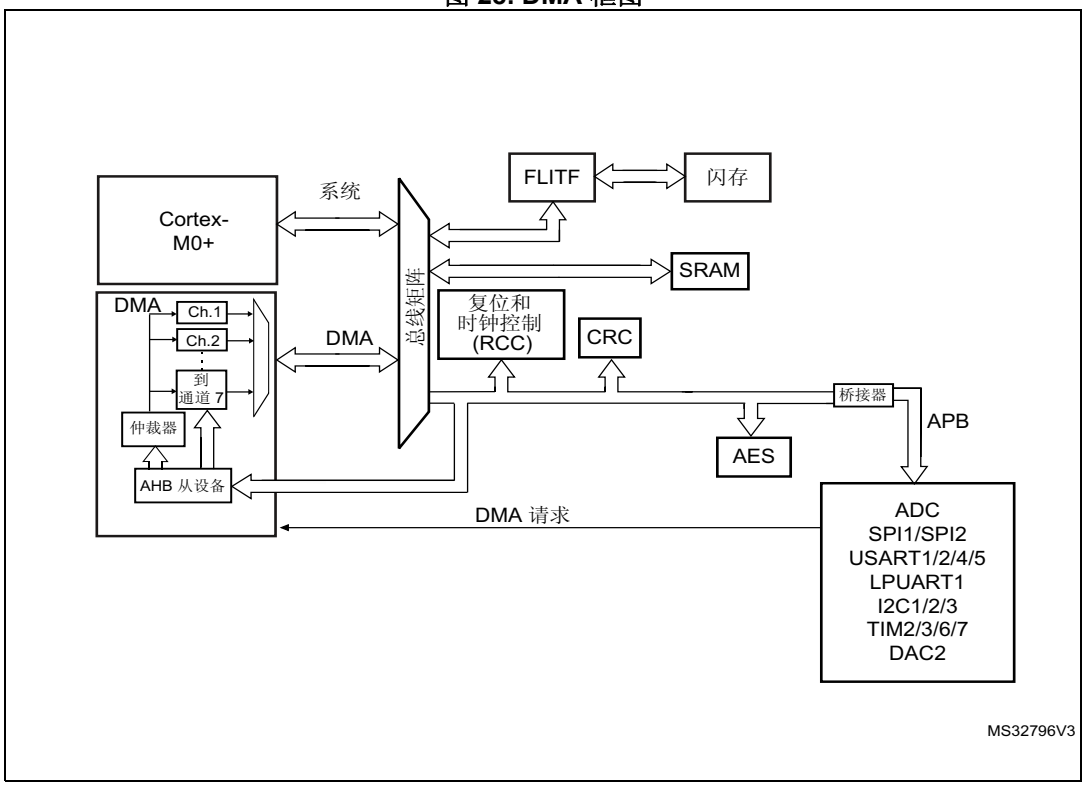
DMA 控制器共有 7 个通道，每个通道都专门管理来自一个或多个外设的存储器访问请求。它具有一个仲裁器，用于处理 DMA 请求间的优先级。

11.2 DMA 主要特性

- 多达 7 个可单独配置通道（请求）
- 每个通道都与专用的硬件 DMA 请求相连，同时还支持软件触发。此配置由软件执行。
- DMA 通道请求的优先级可用软件编程（4 个级别：*非常高*、*高*、*中*和*低*），在软件优先级相同的情况下可以通过硬件决定优先级（例如，请求 1 的优先级高于请求 2）。
- 独立的源和目标传输大小（字节、半字、字），模拟打包和解包。源 / 目标地址必须与数据大小匹配。
- 支持循环缓冲区管理。
- 3 个事件标志（DMA 半传输、DMA 传输完成和 DMA 传输错误），在单个中断请求中针对每个通道一起进行逻辑或运算。
- 存储器到存储器的传输。
- 外设到存储器、存储器到外设以及外设到外设的传输。
- 访问作为源和目标的 Flash、SRAM、APB 和 AHB 外设。
- 可编程的待传输数据数目：最大 65535。

框图如下图所示。

图 28. DMA 框图



11.3 DMA 功能说明

DMA 控制器通过与 Cortex®-M0+ 内核共用系统总线来执行直接存储器传输。当 DMA 与 CPU 访问目标（存储器或外设）相同时，DMA 请求会将 CPU 对系统总线的访问停止数个总线周期。总线矩阵执行循环调度，从而确保 CPU 至少可使用一半的系统总线带宽（无论对于存储器还是外设）。

11.3.1 DMA 事务

在发生事件后，外设会向 DMA 控制器发送请求信号。DMA 控制器根据通道优先级处理该请求。只要 DMA 控制器访问外设，DMA 控制器就会向外设发送确认信号。外设获得 DMA 控制器的确认信号后，便会立即释放其请求。一旦外设使请求失效，DMA 控制器就会释放确认信号。如有更多请求，外设可以启动下一个事务。

概括起来，每个 DMA 传输包含三项操作：

- 从外设数据寄存器或通过内部当前外设/存储器地址寄存器寻址的存储器单元中加载数据。用于首次传输的起始地址是在 DMA_CPARx 寄存器或 DMA_CMARx 寄存器中编程的外设/存储器基址
- 将加载的数据存储到外设数据寄存器或通过内部当前外设/存储器地址寄存器寻址的存储器单元。用于首次传输的起始地址是在 DMA_CPARx 寄存器或 DMA_CMARx 寄存器中编程的外设/存储器基址
- DMA_CNDTRx 寄存器向后递减，该寄存器中包含仍需执行的事务数。

11.3.2 仲裁器

仲裁器根据通道请求的优先级管理通道请求，并启动外设/存储器访问序列。

优先级管理分为两个阶段：

- 软件：每个通道优先级都可以在 DMA_CCRx 寄存器中配置。分为四个级别：
  - 非常高优先级
  - 高优先级
  - 中优先级
  - 低优先级
- 硬件：如果两个请求具有相同的软件优先级，则编号低的通道优先于编号高的通道。例如，通道 2 的优先级高于通道 4。

11.3.3 DMA 通道

各个通道均能处理外设寄存器（位于固定地址中）与存储器单元之间的 DMA 传输。待传输数据的数目可编程（最大 65535）。每个事务完成后，包含要传输的数据项总数的寄存器都会递减。

数据大小可编程

外设和存储器的传输数据大小可通过 DMA_CCRx 寄存器的 PSIZE 位和 MSIZE 位完全进行编程。

指针递增

根据 DMA_CCRx 寄存器的 PINC 位和 MINC 位的状态，可以选择使外设和存储器指针在每次传输后自动向后递增。如果使能了递增模式，则根据所选的数据大小，下次传输的地址即为前一次传输的地址加上 1、2 或 4。首次传输的地址可在 DMA_CPARx/DMA_CMARx 寄存器中进行编程。在传输过程中，这些寄存器将保持初始编程的值。当前传输的地址（位于当前内部外设或存储器地址寄存器中）无法通过软件获得。

如果将通道配置为非循环模式，则在最后一次传输完成后（即待传输的数据项数目达到零后），将不处理任何 DMA 请求。若需要将待传输数据项的新数目重新加载至 DMA_CNDTRx 寄存器中，则必须禁止 DMA 通道。

注：如果禁止 DMA 通道，则将不复位相关 DMA 寄存器。在通道配置阶段，DMA 通道寄存器（DMA_CCRx、DMA_CPARx 和 DMA_CMARx）仍保留初始编程值。

在循环模式下，最后一次传输完成后，DMA_CNDTRx 寄存器将自动重新加载初始编程值。当前的内部地址寄存器重新加载 DMA_CPARx/DMA_CMARx 寄存器中的基址值。

通道配置流程

配置 DMA 通道 x（其中 x 为通道编号）时应遵循下列顺序：

1. 在 DMA_CPARx 寄存器中设置外设寄存器地址。外设事件发生后，数据会从此地址移动到存储器或从存储器移动到此地址。
2. 在 DMA_CMARx 寄存器中设置存储器地址。外设事件发生后，将从此存储器读取数据或将数据写入此存储器中。
3. 在 DMA_CNDTRx 寄存器中配置待传输的总数据数。在每次外设事件发生后，此值都会递减。
4. 使用 DMA_CCRx 寄存器中的 PL[1:0] 位来配置通道优先级。
5. 在 DMA_CCRx 寄存器中配置数据传输方向、循环模式、外设和存储器递增模式、外设和存储器数据大小以及中断方式（数据传输一半产生中断还是数据完全传输后产生中断）
6. 将 DMA_CCRx 寄存器中的 ENABLE 位置 1 以激活通道。

有关代码示例，请参见 *A.6.1：DMA 通道配置序列代码示例*。

通道使能后，即可响应与该通道相连的外设所发出的所有 DMA 请求。

在数据传输一半后，传输一半标志 (HTIF) 便会置 1，如果传输一半中断使能位 (HTIE) 置 1，还会生成中断。传输结束时，传输完成标志 (TCIF) 便会置 1，如果传输完成中断使能位 (TCIE) 置 1，还会生成中断。

循环模式

循环模式可用于处理循环缓冲区和连续数据流（例如 ADC 扫描模式）。可以使用 DMA_CCRx 寄存器中的 CIRC 位使能此特性。如果循环模式已激活，待传输数据的数目将在通道配置阶段自动重新加载为初始编程值，并继续响应 DMA 请求。

存储器到存储器模式

DMA 通道在没有外设请求触发的情况下同样可以工作。此模式称为“存储器到存储器”模式。

如果 DMA_CCRx 寄存器中的 MEM2MEM 位置 1，则只要通过软件将 DMA_CCRx 寄存器中的使能位 (EN) 置 1，通道就会启动传输。DMA_CNDTRx 寄存器达到零后，传输停止。存储器到存储器模式不能与循环模式同时使用。

11.3.4 可编程数据宽度、数据对齐和字节存储次序

如果 PSIZE 与 MSIZE 不相等，则 DMA 将按照*表 47：可编程的数据宽度、字节存储次序（位 PINC = MINC = 1 时）*所述方式进行数据对齐。

表 47. 可编程的数据宽度、字节存储次序（位 PINC = MINC = 1 时）

源端口宽度	目标端口宽度	要传输的数据项的数目 (NDT)	源内容：地址/数据	传输操作	目标内容：地址/数据
8	8	4	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3	1：读取 B0[7:0] @0x0，然后写入 B0[7:0] @0x0 2：读取 B1[7:0] @0x1，然后写入 B1[7:0] @0x1 3：读取 B2[7:0] @0x2，然后写入 B2[7:0] @0x2 4：读取 B3[7:0] @0x3，然后写入 B3[7:0] @0x3	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3
8	16	4	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3	1：读取 B0[7:0] @0x0，然后写入 00B0[15:0] @0x0 2：读取 B1[7:0] @0x1，然后写入 00B1[15:0] @0x2 3：读取 B3[7:0] @0x2，然后写入 00B2[15:0] @0x4 4：读取 B4[7:0] @0x3，然后写入 00B3[15:0] @0x6	@0x0/00B0 @0x2/00B1 @0x4/00B2 @0x6/00B3
8	32	4	@0x0/B0 @0x1/B1 @0x2/B2 @0x3/B3	1：读取 B0[7:0] @0x0，然后写入 000000B0[31:0] @0x0 2：读取 B1[7:0] @0x1，然后写入 000000B1[31:0] @0x4 3：读取 B3[7:0] @0x2，然后写入 000000B2[31:0] @0x8 4：读取 B4[7:0] @0x3，然后写入 000000B3[31:0] @0xC	@0x0/000000B0 @0x4/000000B1 @0x8/000000B2 @0xC/000000B3
16	8	4	@0x0/B1B0 @0x2/B3B2 @0x4/B5B4 @0x6/B7B6	1：读取 B1B0[15:0] @0x0，然后写入 B0[7:0] @0x0 2：读取 B3B2[15:0] @0x2，然后写入 B2[7:0] @0x1 3：读取 B5B4[15:0] @0x4，然后写入 B4[7:0] @0x2 4：读取 B7B6[15:0] @0x6，然后写入 B6[7:0] @0x3	@0x0/B0 @0x1/B2 @0x2/B4 @0x3/B6
16	16	4	@0x0/B1B0 @0x2/B3B2 @0x4/B5B4 @0x6/B7B6	1：读取 B1B0[15:0] @0x0，然后写入 B1B0[15:0] @0x0 2：读取 B3B2[15:0] @0x2，然后写入 B3B2[15:0] @0x2 3：读取 B5B4[15:0] @0x4，然后写入 B5B4[15:0] @0x4 4：读取 B7B6[15:0] @0x6，然后写入 B7B6[15:0] @0x6	@0x0/B1B0 @0x2/B3B2 @0x4/B5B4 @0x6/B7B6
16	32	4	@0x0/B1B0 @0x2/B3B2 @0x4/B5B4 @0x6/B7B6	1：读取 B1B0[15:0] @0x0，然后写入 0000B1B0[31:0] @0x0 2：读取 B3B2[15:0] @0x2，然后写入 0000B3B2[31:0] @0x4 3：读取 B5B4[15:0] @0x4，然后写入 0000B5B4[31:0] @0x8 4：读取 B7B6[15:0] @0x6，然后写入 0000B7B6[31:0] @0xC	@0x0/0000B1B0 @0x4/0000B3B2 @0x8/0000B5B4 @0xC/0000B7B6
32	8	4	@0x0/B3B2B1B0 @0x4/B7B6B5B4 @0x8/BBBAB9B8 @0xC/BFBEBDBC	1：读取 B3B2B1B0[31:0] @0x0，然后写入 B0[7:0] @0x0 2：读取 B7B6B5B4[31:0] @0x4，然后写入 B4[7:0] @0x1 3：读取 BBBAB9B8[31:0] @0x8，然后写入 B8[7:0] @0x2 4：读取 BFBEBDBC[31:0] @0xC，然后写入 BC[7:0] @0x3	@0x0/B0 @0x1/B4 @0x2/B8 @0x3/BC
32	16	4	@0x0/B3B2B1B0 @0x4/B7B6B5B4 @0x8/BBBAB9B8 @0xC/BFBEBDBC	1：读取 B3B2B1B0[31:0] @0x0，然后写入 B1B0[15:0] @0x0 2：读取 B7B6B5B4[31:0] @0x4，然后写入 B5B4[15:0] @0x2 3：读取 BBBAB9B8[31:0] @0x8，然后写入 B9B8[15:0] @0x4 4：读取 BFBEBDBC[31:0] @0xC，然后写入 BDBC[15:0] @0x6	@0x0/B1B0 @0x2/B5B4 @0x4/B9B8 @0x6/BDBC
32	32	4	@0x0/B3B2B1B0 @0x4/B7B6B5B4 @0x8/BBBAB9B8 @0xC/BFBEBDBC	1：读取 B3B2B1B0[31:0] @0x0，然后写入 B3B2B1B0[31:0] @0x0 2：读取 B7B6B5B4[31:0] @0x4，然后写入 B7B6B5B4[31:0] @0x4 3：读取 BBBAB9B8[31:0] @0x8，然后写入 BBBAB9B8[31:0] @0x8 4：读取 BFBEBDBC[31:0] @0xC，然后写入 BFBEBDBC[31:0] @0xC	@0x0/B3B2B1B0 @0x4/B7B6B5B4 @0x8/BBBAB9B8 @0xC/BFBEBDBC

解决 AHB 外设无法支持字节或半字写操作的问题

如果 DMA 初始化了 AHB 字节或半字写操作，则会将数据复制到 HWDATA[31:0] 总线的未使用通道上。因此，若所用的 AHB 从外设不支持字节或半字写操作（即外设未使用 HSIZE）*且*不会产生任何错误，则 DMA 会对 HWDATA 的 32 个位执行写操作，如下列两个示例所示：

- 要写入半字“0xABCD”，则 DMA 会将 HWDATA 总线设为“0xABCDABCD”，同时将 HSIZE 设为 HalfWord
- 要写入字节“0xAB”，则 DMA 会将 HWDATA 总线设为“0xABABABAB”，同时将 HSIZE 设为 Byte

假设 AHB/APB 桥为 AHB 32 位从外设，且该外设未设置数据的 HSIZE，则任何 AHB 字节或半字操作都将转换为 32 位 APB 操作，转换方式如下所示：

- 将 AHB 字节写操作转化为 APB 字写操作，如向 0x0（或 0x1、0x2、0x3）写入数据“0xB0”转化为向 0x0 写入数据“0xB0B0B0B0”
- 将 AHB 半字写操作转化为 APB 字写操作，如向 0x0（或 0x2）写入数据“0xB1B0”转化为向 0x0 写入数据“0xB1B0B1B0”

例如，若用户希望对 APB 备份寄存器（与 32 位地址边界对齐的16 位寄存器）执行写操作，则必须将存储器源大小 (MSIZE) 配置为“16 位”，同时将外设目标大小 (PSIZE) 配置为“32 位”。

11.3.5 错误管理

当对保留的地址空间执行读取操作时，将生成 DMA 传输错误。若在 DMA 读或写访问过程中生成了 DMA 传输错误，则会将相应的通道配置寄存器 (DMA_CCRx) 的 EN 位进行硬件清零，从而自动禁止出错的通道。如果 DMA_CCRx 寄存器中的传输错误中断使能位 (TEIE) 置 1，则 DMA_IFR 寄存器中该通道的传输错误中断标志 (TEIF) 将置 1 并生成中断。

11.3.6 DMA 中断

对于每个 DMA 通道，在发生“半传输”、“传输完成”或“传输错误”时都可以产生中断。可以使用单独的中断使能位以提高灵活性。

表 48. DMA 中断请求

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

11.3.7 DMA 请求映射

DMA 控制器

来自外设（TIM2/6、ADC、DAC、SPI1/2、I2C1/2、AES（仅限具有 AES 的 第 3 类和第 5 类器件）、USART1/2 以及 LPUART1）的各硬件请求可通过 DMA 通道选择寄存器映射到 DMA 通道（1 到 7）。在一个通道上，一次只能响应一个请求。请参见*图 29：DMA 请求映射*。

外设 DMA 请求可对相应外设的寄存器的 DMA 控制位进行编程，从而单独进行激活或取消激活。

图 29. DMA 请求映射

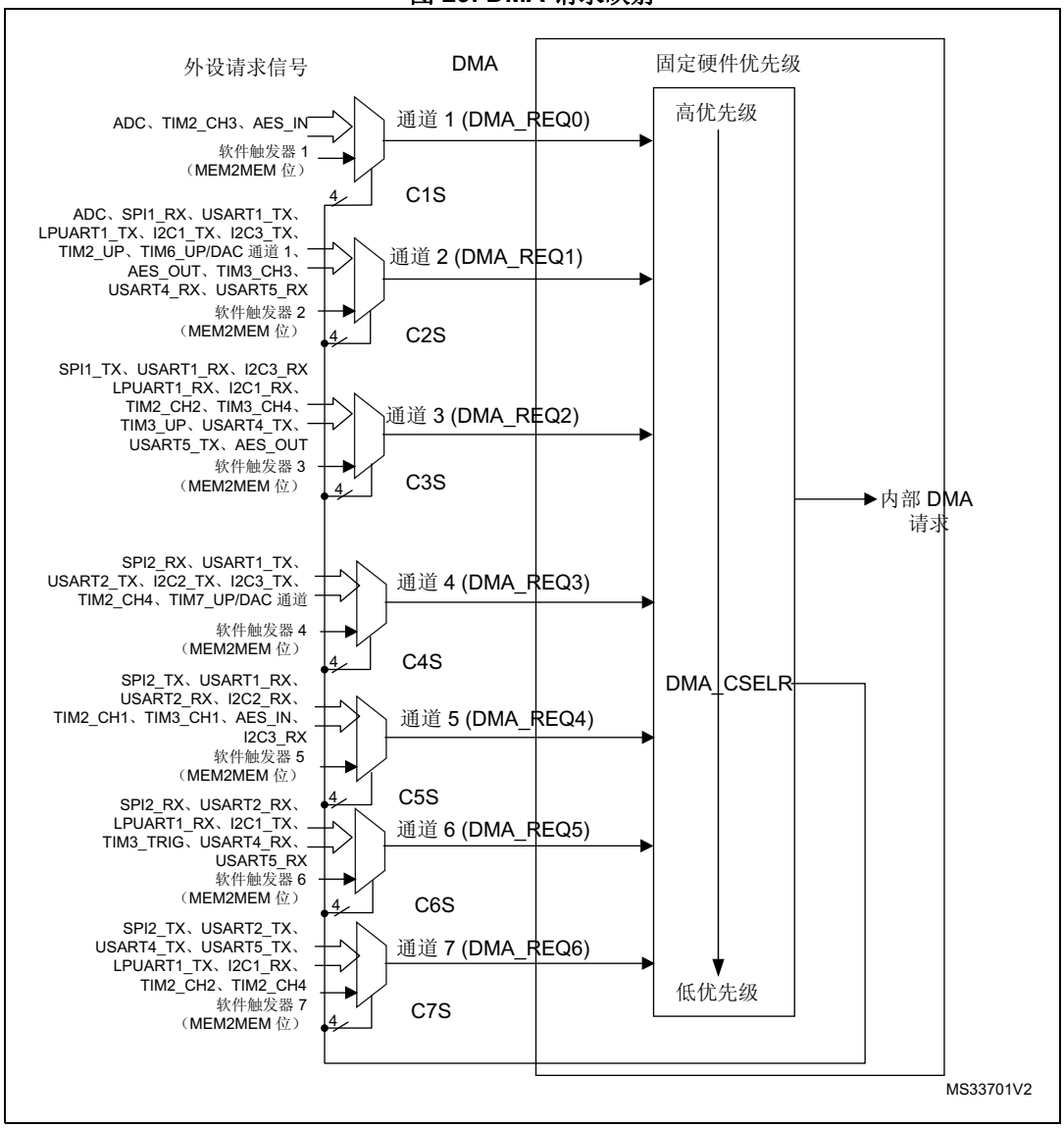


表 49 列出了各通道的 DMA 请求。

表 49. 各通道 DMA 请求汇总

请求编号	外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
0	**ADC**	ADC	ADC	-	-		-	-
1	**SPI1**	-	SPI1_RX	SPI1_TX	-	-	-	-
2	**SPI2**	-	-	-	SPI2_RX	SPI2_TX	SPI2_RX	SPI2_TX
3	**USART1**	-	USART1_TX	USART1_RX	USART1_TX	USART1_RX	-	-
4	**USART2**	-	-	-	USART2_TX	USART2_RX	USART2_RX	USART2_TX
5	**LPUART1**	-	LPUART1_TX	LPUART1_RX	-	-	LPUART1_RX	LPUART1_TX
6	**I2C1**	-	I2C1_TX	I2C1_RX	-	--	I2C1_TX	I2C1_RX
7	**I2C2**	-	-	-	I2C2_TX	I2C2_RX	-	-
8	**TIM2**	TIM2_CH3	TIM2_UP	TIM2_CH2	TIM2_CH4	TIM2_CH1		TIM2_CH2 TIM2_CH4
9	**TIM6_UP/ DAC_ channel1**	-	TIM6/DAC_ channel1	-	-	-	-	-
10	**TIM3**	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	TIM3_CH1	TIM3_TRIG	-
11	**AES**(1)	AES_IN	AES_OUT	AES_OUT		AES_IN		
12	**USART4**	-	USART4_RX	USART4_TX	-	-	USART4_RX	USART4_TX
13	**USART5**	-	USART5_RX	USART5_TX	-	-	USART5_RX	USART5_TX
14	**I2C3**	-	I2C3_TX	I2C3_RX	I2C3_TX	I2C3_RX	-	-
15	**TIM7_UO/ DAC_ channel2**	-	-	-	TIM7/DAC_ channel2	-	-	-

1. 仅限具有 AES 的第 3 类和第 5 类器件。

11.4 DMA 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器可支持字节（8 位）、半字（16 位）或字（32 位）访问。

11.4.1 DMA 中断状态寄存器 (DMA_ISR)

DMA interrupt status register

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
Res.	Res.	Res.	Res.	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 31:28 保留，必须保持复位值。

位 27、23、19、15、11、7、3 **TEIFx**：通道 x 传输错误标志 (Channel x transfer error flag) (x = 1..7)
此位由硬件置 1，由软件清零，软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可。
0：通道 x 上无传输错误 (TE)
1：通道 x 上出现传输错误 (TE)

位 26、22、18、14、10、6、2 **HTIFx**：通道 x 半传输标志 (Channel x half transfer flag) (x = 1..7)
此位由硬件置 1，由软件清零，软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可。
0：通道 x 上无半传输 (HT) 事件
1：通道 x 上发生半传输 (HT) 事件

位 25、21、17、13、9、5、1 **TCIFx**：通道 x 传输完成标志 (Channel x transfer complete flag) (x = 1..7)
此位由硬件置 1，由软件清零，软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可。
0：通道 x 上无传输完成 (TC) 事件
1：通道 x 上发生传输完成 (TC) 事件

位 24、20、16、12、8、4、0 **GIFx**：通道 x 全局中断标志 (Channel x global interrupt flag) (x = 1..7)
此位由硬件置 1，由软件清零，软件只需将 1 写入 DMA_IFCR 寄存器的相应位即可。
0：通道 x 上无 TE、HT 或 TC 事件
1：通道 x 上发生 TE、HT 或 TC 事件

11.4.2 DMA 中断标志清零寄存器 (DMA_IFCR)

DMA interrupt flag clear register

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w

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

位 31:28 保留，必须保持复位值。

位 27、23、19、15、11、7、3 **CTEIFx：**通道 x 传输错误清零 (Channel x transfer error clear) (x = 1..7)
此位由软件置 1 和清零。
0：无影响
1：清除 DMA_ISR 寄存器中相应的 TEIF 标志

位 26、22、18、14、10、6、2 **CHTIFx：**通道 x 半传输清零 (Channel x half transfer clear) (x = 1..7)
此位由软件置 1 和清零。
0：无影响
1：清除 DMA_ISR 寄存器中相应的 HTIF 标志

位 25、21、17、13、9、5、1 **CTCIFx：**通道 x 传输完成清零 (Channel x transfer complete clear) (x = 1..7)
此位由软件置 1 和清零。
0：无影响
1：清除 DMA_ISR 寄存器中相应的 TCIF 标志

位 24、20、16、12、8、4、0 **CGIFx：**通道 x 全局中断清零 (Channel x global interrupt clear) (x = 1..7)
此位由软件置 1 和清零。
0：无影响
1：清除 DMA_ISR 寄存器中的 GIF、TEIF、HTIF 和 TCIF 标志

11.4.3 DMA 通道 x 配置寄存器 (DMA_CCRx)（x = 1..7 ，其中 x 表示通道编号）

DMA channel x configuration register

偏移地址：0x08 + 0d20 × (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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	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 31:15 保留，必须保持复位值。

位 14 **MEM2MEM**：存储器到存储器模式 (Memory to memory mode)
此位由软件置 1 和清零。
0：禁止存储器到存储器模式
1：使能存储器到存储器模式

位 13:12 **PL[1:0]**：通道优先级 (Channel priority level)
这些位由软件置 1 和清零。
00：低
01：中
10：高
11：非常高

位 11:10 **MSIZE[1:0]**：存储器大小 (Memory size)
这些位由软件置 1 和清零。
00：8 位
01：16 位
10：32 位
11：保留

位 9:8 **PSIZE[1:0]**：外设大小 (Peripheral size)
这些位由软件置 1 和清零。
00：8 位
01：16 位
10：32 位
11：保留

位 7 **MINC**：存储器递增模式 (Memory increment mode)
此位由软件置 1 和清零。
0：禁止存储器递增模式
1：使能存储器递增模式

位 6 **PINC**：外设递增模式 (Peripheral increment mode)
此位由软件置 1 和清零。
0：禁止外设递增模式
1：使能外设递增模式

位 5 **CIRC**：循环模式 (Circular mode)

此位由软件置 1 和清零。

0：禁止循环模式
1：使能循环模式

位 4 **DIR**：数据传输方向 (Data transfer direction)

此位由软件置 1 和清零。

0：从外设读取
1：从存储器读取

位 3 **TEIE**：传输错误中断使能 (Transfer error interrupt enable)

此位由软件置 1 和清零。

0：禁止 TE 中断
1：使能 TE 中断

位 2 **HTIE**：半传输中断使能 (Half transfer interrupt enable)

此位由软件置 1 和清零。

0：禁止 HT 中断
1：使能 HT 中断

位 1 **TCIE**：传输完成中断使能 (Transfer complete interrupt enable)

此位由软件置 1 和清零。

0：禁止 TC 中断
1：使能 TC 中断

位 0 **EN**：通道使能 (Channel enable)

此位由软件置 1 和清零。

0：禁止通道
1：使能通道

11.4.4 DMA 通道 x 数据数寄存器 (DMA_CNDTRx)（x = 1..7，其中 x 表示通道编号）

DMA channel x number of data register

偏移地址：0x0C + 0d20 × (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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

位 31:16 保留，必须保持复位值。

位 15:0 **NDT[15:0]**：待传输的数据数 (Number of data to transfer)

待传输的数据数（0 到 65535）。只有在通道禁止时才能写入此寄存器。通道使能后，该寄存器为只读，用于指示待传输的剩余字节数。每次 DMA 传输后，此寄存器将递减。

传输完成后，该寄存器的值可以保持为零，若已将通道配置为自动重载模式，则自动重新加载先前编程的值。

若该寄存器的值为零，则无论通道是否使能，都将不会处理任何事务。

11.4.5 DMA 通道 x 外设地址寄存器 (DMA_CPARx)（x = 1..7，其中 x 表示通道编号）

DMA channel x peripheral address register

偏移地址：0x10 + 0d20 × (通道编号 – 1)

复位值：0x0000 0000

若通道已使能，则*不得*对该寄存器执行写操作。

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

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

位 31:0 **PA[31:0]**：外设地址 (Peripheral address)

读/写数据的外设数据寄存器的基址。

若 PSIZE 为 01（16 位），则忽略 PA[0] 位。访问将自动对齐到半字地址。

若 PSIZE 为 10（32 位），则忽略 PA[1:0]。访问将自动对齐到字地址。

11.4.6 DMA 通道 x 存储器地址寄存器 (DMA_CMARx)（x = 1..7，其中 x 表示通道编号）

DMA channel x memory address register

偏移地址：0x14 + 0d20 × (通道编号 – 1)

复位值：0x0000 0000

若通道已使能，则*不得*对该寄存器执行写操作。

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

位 31:0 **MA[31:0]**：存储器地址 (Memory Address)

读/写数据的存储区基址。

若 MSIZE 为 01（16 位），则忽略 MA[0] 位。访问将自动对齐到半字地址。

若 MSIZE 为 10（32 位），则忽略 MA[1:0]。访问将自动对齐到字地址。

11.4.7 DMA 通道选择寄存器 (DMA_CSELR)

DMA channel selection register

偏移地址：0xA8

复位值：0x0000 0000

该寄存器用于管理 DMA 通道的重映射（请参见*图 29*）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	C7S [3:0]				C6S [3:0]				C5S [3:0]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C4S [3:0]				C3S [3:0]				C2S [3:0]				C1S [3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:24 **C7S[3:0]**：DMA 通道 7 选择 (DMA channel 7 selection)

0010：DMA 通道 7 重映射到 SPI2_TX
0100：DMA 通道 7 重映射到 USART2_TX
0101：DMA 通道 7 重映射到 LPUART1_TX
0110：DMA 通道 7 重映射到 I2C1_RX
1000：DMA 通道 7 重映射到 TIM2_CH2/TIM2_CH4
1100：DMA 通道 7 重映射到 USART4_TX
1101：DMA 通道 7 重映射到 USART5_TX
其他配置：DMA 通道 7 不进行重映射

位 23:20 **C6S[3:0]**：DMA 通道 6 选择 (DMA channel 6 selection)

0010：DMA 通道 6 重映射到 SPI2_RX
0100：DMA 通道 6 重映射到 USART2_RX
0101：DMA 通道 6 重映射到 LPUART1_RX
0110：DMA 通道 6 重映射到 I2C1_TX
1010：DMA 通道 6 重映射到 TIM3_TRIG
1100：DMA 通道 6 重映射到 USART4_RX
1101：DMA 通道 6 重映射到 USART5_RX
其他配置：DMA 通道 6 不进行重映射

位 19:16 **C5S[3:0]**：DMA 通道 5 选择 (DMA channel 5 selection)

0010：DMA 通道 5 重映射到 SPI2_TX
0011：DMA 通道 5 重映射到 USART1_RX
0100：DMA 通道 5 重映射到 USART2_RX
0111：DMA 通道 5 重映射到 I2C2_RX
1000：DMA 通道 5 重映射到 TIM2_CH1
1010：DMA 通道 5 重映射到 TIM3_CH1
1011：DMA 通道 5 重映射到 AES_IN
1110：DMA 通道 5 重映射到 I2C3_RX
其他配置：DMA 通道 5 不进行重映射

位 15:12 **C4S[3:0]**：DMA 通道 4 选择 (DMA channel 4 selection)
0010：DMA 通道 4 重映射到 SPI2_RX
0011：DMA 通道 4 重映射到 USART1_TX
0100：DMA 通道 4 重映射到 USART2_TX
0111：DMA 通道 4 重映射到 I2C2_TX
1000：DMA 通道 4 重映射到 TIM2_CH4
1110：DMA 通道 4 重映射到 I2C3_TX
1111：DMA 通道 4 重映射到 TIM7_UP/DAC 通道 2
其他配置：DMA 通道 4 不进行重映射

位 11:8 **C3S[3:0]**：DMA 通道 3 选择 (DMA channel 3 selection)
0001：DMA 通道 3 重映射到 SPI1_TX
0011：DMA 通道 3 重映射到 USART1_RX
0101：DMA 通道 3 重映射到 LPUART1_RX
0110：DMA 通道 3 重映射到 I2C1_RX
1000：DMA 通道 3 重映射到 TIM2_CH2
1010：DMA 通道 3 重映射到 TIM4_CH4/TIM4_UP
1011：DMA 通道 3 重映射到 AES_OUT
1100：DMA 通道 3 重映射到 USART4_TX
1101：DMA 通道 3 重映射到 USART4_TX
1110：DMA 通道 3 重映射到 I2C3_RX
其他配置：DMA 通道 3 不进行重映射

位 7:4 **C2S[3:0]**：DMA 通道 2 选择 (DMA channel 2 selection)
0000：DMA 通道 2 重映射到 ADC
0001：DMA 通道 2 重映射到 SPI1_RX
0011：DMA 通道 2 重映射到 USART1_TX
0101：DMA 通道 2 重映射到 LPUART1_TX
0110：DMA 通道 2 重映射到 I2C1_TX
1000：DMA 通道 2 重映射到 TIM2_UP
1001：DMA 通道 2 重映射到 TIM6_UP/DAC 通道 1
1010：DMA 通道 2 重映射到 TIM3_CH3
1011：DMA 通道 2 重映射到 AES_OUT
1100：DMA 通道 2 重映射到 USART4_RX
1101：DMA 通道 2 重映射到 USART5_RX
1110：DMA 通道 2 重映射到 I2C3_TX
其他配置：DMA 通道 2 不进行重映射

位 3:0 **C1S[3:0]**：DMA 通道 1 选择 (DMA channel 1 selection)
0000：DMA 通道 1 重映射到 ADC
1000：DMA 通道 1 重映射到 TIM2_CH3
1011：DMA 通道 1 重映射到 AES_IN
其他配置：DMA 通道 1 不进行重映射

11.4.8 DMA 寄存器映射

下表列出了 DMA 寄存器映射和复位值。

表 50. DMA 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DMA_ISR	Res.	Res.	Res.	Res.	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
	Reset value					0	0	0	0	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	DMA_IFCR	Res.	Res.	Res.	Res.	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
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	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	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 50. DMA 寄存器映射和复位值（续）

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
0x4C	**DMA_CPAR4**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**DMA_CMAR4**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	Reserved																																
0x58	**DMA_CCR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**DMA_CNDTR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**DMA_CPAR5**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	**DMA_CMAR5**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	**DMA_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**DMA_CNDTR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	**DMA_CPAR6**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	**DMA_CMAR6**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	**DMA_CCR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]		MSIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**DMA_CNDTR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																
0x0A8	**DMA_CSELR**	Res.	Res.	Res.	Res.	C7S[3:0]				C6S[3:0]				C5S[3:0]				C4S[3:0]				C3S[3:0]				C2S[3:0]				C1S[3:0]			
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

12 嵌套向量中断控制器 (NVIC)

12.1 主要特性

- 具有多达 39 个可屏蔽中断通道（请参见*表 51*），不包括 Cortex®-M0+ 的 16 根中断线。
- 4 个可编程优先级（使用了 2 位中断优先级）
- 低延迟异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

NVIC 和处理器内核接口紧密配合，可以实现低延迟的中断处理和晚到中断的高效处理。

包括内核异常在内的所有中断均通过 NVIC 进行管理。关于异常和 NVIC 编程的更多信息，请参见 STM32L0 系列 Cortex®-M0+ 编程手册 (PM0223)。

有关代码示例，请参见 *A.7.1：NVIC 初始化示例*。

12.2 SysTick 校准值寄存器

SysTick 校准值设置为 4000。当 SysTick 时钟设置为 4 MHz（最大 HCLK/8），会产生 1 ms 时间基准。

12.3 中断和异常向量

表 51 为 STM32L0x3 器件的向量表。

表 51. 向量列表(1)(2)

位置	优先级	优先级类型	缩略语	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	复位	复位	0x0000_0004
	-2	固定	NMI_Handler	不可屏蔽中断。RCC 时钟安全系统 (CSS) 连接到 NMI 向量。	0x0000_0008
	-1	固定	HardFault_Handler	所有类型的错误	0x0000_000C
	-	-	-	保留	0x0000_0010 - 0x0000_002B
	3	可设置	SVC_Handler	通过 SWI 指令调用的系统服务	0x0000_002C
	-	-	-	保留	0x0000_0030 - 0x0000_0037
	5	可设置	PendSV_Handler	可挂起的系统服务请求	0x0000_0038
	6	可设置	SysTick_Handler	系统节拍定时器	0x0000_003C
0	7	可设置	WWDG	窗口看门狗中断	0x0000_0040
1	8	可设置	PVD	通过 EXTI 线的 PVD 中断	0x0000_0044
2	9	可设置	RTC	RTC 全局中断（通过 EXTI17/19/20 线）和 LSE CSS 中断（通过 EXTI 19 线）	0x0000_0048

表 51. 向量列表[(1)(2)]（续）

位置	优先级	优先级类型	缩略语	说明	地址
3	10	可设置	FLASH	Flash 和数据 EEPROM 全局中断	0x0000_004C
4	11	可设置	RCC_CRS	RCC 和 CRS 全局中断	0x0000_0050
5	12	可设置	EXTI[1:0]	EXTI 线 0 和 1 中断	0x0000_0054
6	13	可设置	EXTI[3:2]	EXTI 线 2 和 3 中断	0x0000_0058
7	14	可设置	EXTI[15:4]	EXTI 线 4 到 15 中断	0x0000_005C
8	15	可设置	TSC	触摸感应控制器中断	0x0000_0060
9	16	可设置	DMA1_Channel1	DMA1 通道 1 全局中断	0x0000_0064
10	17	可设置	DMA1_Channel[3:2]	DMA1 通道 2 和通道 3 中断	0x0000_0068
11	18	可设置	DMA1_Channel[7:4]	DMA1 通道 4 到通道 7 中断	0x0000_006C
12	19	可设置	ADC_COMP	ADC 和比较器中断（通过 EXTI21 和 22）	0x0000_0070
13	20	可设置	LPTIM1	LPTIMER1 中断（通过 EXTI29）	0x0000_0074
14	21	可设置	USART4/USART5	USART4/USART5 全局中断	0x0000_0078
15	22	可设置	TIM2	TIMER2 全局中断	0x0000_007C
16	23	可设置	TIM3	TIMER3 全局中断	0x0000_0080
17	24	可设置	TIM6_DAC	TIMER6 全局中断和 DAC 中断	0x0000_0084
18	25	可设置	TIM7	TIMER7 全局中断	0x0000_0088
19	26	可设置	-	保留	0x0000_008C
20	27	可设置	TIM21	TIMER21 全局中断	0x0000_0090
21	28	可设置	I2C3	I2C3 全局中断	0x0000_0094
22	29	可设置	TIM22	TIMER22 全局中断	0x0000_0098
23	30	可设置	I2C1	I2C1 全局中断（通过 EXTI23）	0x0000_009C
24	31	可设置	I2C2	I2C2 全局中断	0x0000_00A0
25	32	可设置	SPI1	SPI1 全局中断	0x0000_00A4
26	33	可设置	SPI2	SPI2 全局中断	0x0000_00A8
27	34	可设置	USART1	USART1 全局中断（通过 EXTI25）	0x0000_00AC
28	35	可设置	USART2	USART2 全局中断（通过 EXTI26）	0x0000_00B0
29	36	可设置	LPUART1 + AES +RNG	LPUART1 全局中断（通过 EXTI28）+ AES 全局中断 + RNG 全局中断	0x0000_00B4
30	37	可设置	LCD	LCD 全局中断	0x0000_00B8
31	38	可设置	USB	USB 事件中断（通过 EXTI18）	0x0000_00BC

1. 灰色单元格对应于 Cortex®-M0+ 中断。
2. 有关器件上可用的 GPIO 端口和外设，请参阅*表 1：STM32L0x3 存储器容量*、*表 2：按类别划分的特性*和器件数据手册。对应于不可用 GPIO 端口或外设的存储区为保留区。

13 扩展中断和事件控制器 (EXTI)

13.1 简介

扩展中断和事件控制器 (EXTI) 用于管理内外部异步事件/中断，并生成对 CPU/中断控制器的事件请求和对电源控制器的唤醒请求。

EXTI 最多可管理 30 根事件线，这些事件线可将器件从停止模式唤醒。

其中有部分线可配置：对于这类线，可以独立选择有效边沿，并通过状态标志指示中断源。可配置线供 I/O 外部中断和少部分外设使用。有些线无需配置：它们供部分外设使用，以生成从停止模式唤醒的事件或中断。对于这类线，状态标志由外设提供。

每根线均可单独屏蔽生成的中断或事件。

EXTI 控制器还允许通过编程专用的寄存器来在与其相对应的硬件事件线上进行软件仿真事件或者中断。

13.2 EXTI 主要特性

EXTI 的主要特性如下：

- 可生成多达 30 个事件/中断请求
  - 22 条可配置线
  - 7 条无需配置的线
- 单独屏蔽每根事件/中断线
- 可配置上升沿或下降沿（仅限可配置线）
- 专用状态位（仅限可配置线）
- 事件/中断仿真请求（仅限可配置线）

13.3 EXTI 功能说明

对于可配置的中断线，应配置好并使能中断线才能产生中断。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”使能中断请求。当中断线上出现选定信号沿时，便会产生中断请求，对应的挂起位也会置 1。在挂起寄存器的对应位写“1”，将清除该中断请求。

对于无需配置的中断线：默认在中断屏蔽寄存器中使能中断，挂起寄存器中没有对应的挂起位。

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。当事件线上出现选定信号沿时，便会产生事件脉冲，对应的挂起位不会置 1。

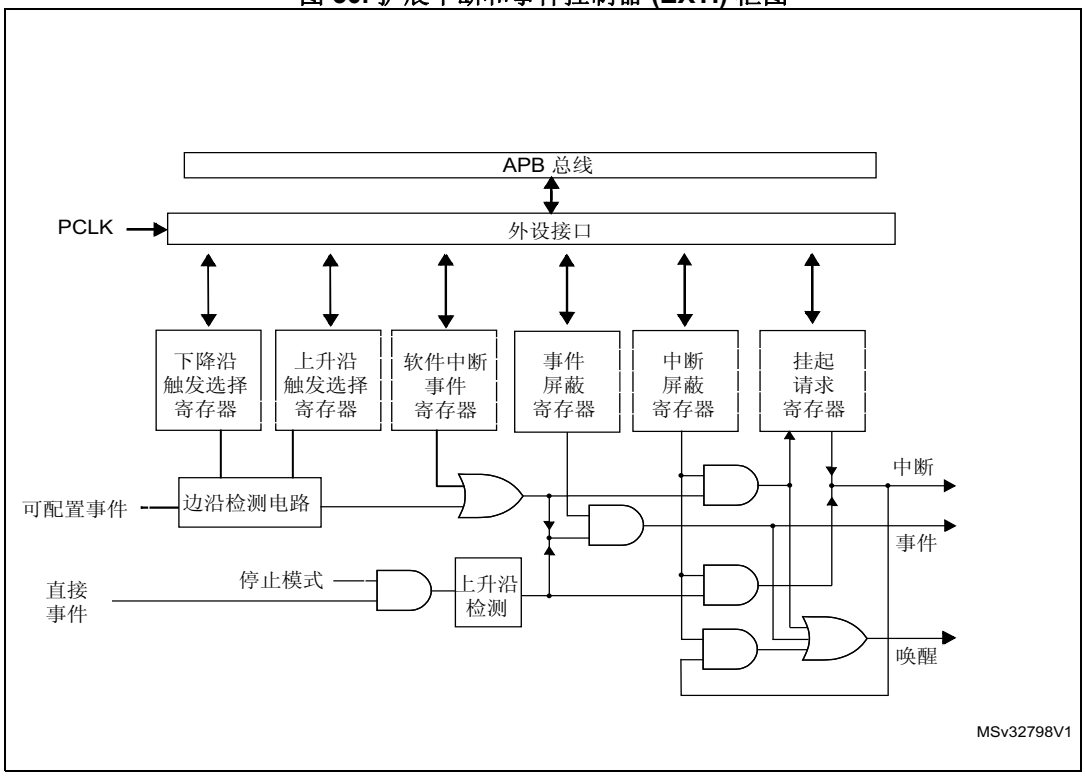
对于可配置线，通过在软件中对软件中断/事件寄存器写“1”，也可以产生中断/事件请求。

注：与无需配置的线相关的中断或事件仅在系统处于停止模式时触发。如果系统仍在运行，EXTI 不会产生任何中断/事件。

13.3.1 EXTI 框图

框图如*图 30* 所示。

图 30. 扩展中断和事件控制器 (EXTI) 框图



13.3.2 唤醒事件管理

STM32L0x3 微控制器能够处理唤醒内核 (WFE) 的外部或内部事件。唤醒事件可通过以下方式产生：

- 在外设的控制寄存器中使能一个中断，但不能在 NVIC 中使能，同时使能 Cortex®-M0+ 系统控制寄存器中的 SEVONPEND 位（请参见 STM32L0 系列 Cortex®-M0+ 编程手册 (PM0223)）。当 MCU 从 WFE 恢复时，需要清除相应外设的中断挂起位和外设 NVIC IRQ 通道挂起位（在 NVIC 中断清除挂起寄存器中）。
- 将一个 EXTI 线配置为事件模式。当 CPU 从 WFE 恢复时，因为对应事件线的挂起位没有被置 1，不必清除相应外设的中断挂起位或 NVIC IRQ 通道挂起位。

13.3.3 外设异步中断

有些外设在系统处于运行模式或停止模式时都能产生事件，从而可将系统从停止模式唤醒。

为实现此目的，外设会同时生成同步（与系统时钟同步，即 APB 时钟）和异步两种事件。此异步事件连接到 EXTI 无需配置的线。

注：少部分具有从停止模式唤醒功能的外设连接到 EXTI 可配置线。对于这类外设，需要配置 EXTI 才能从停止模式唤醒。

13.3.4 硬件中断选择

要将一根线配置为中断源，请执行以下步骤：

1. 配置中断线的屏蔽位 (EXTI_IMR)。
2. 配置中断线的触发选择位（EXTI_RTSR 和 EXTI_FTSR）。
3. 配置用于控制映射到扩展中断控制器 (EXTI) 的 NVIC IRQ 通道的使能位和屏蔽位，从而使任一中断线中的请求可以被正确地响应。

无需配置的线不需要进行任何 EXTI 配置。

有关代码示例，请参见 *A.7.2：扩展中断选择代码示例*。

13.3.5 硬件事件选择

要将一根线配置为事件源，请执行以下步骤：

1. 配置事件线的屏蔽位 (EXTI_EMR)
2. 配置事件线的触发选择位（EXTI_RTSR 和 EXTI_FTSR）

13.3.6 软件中断/事件选择

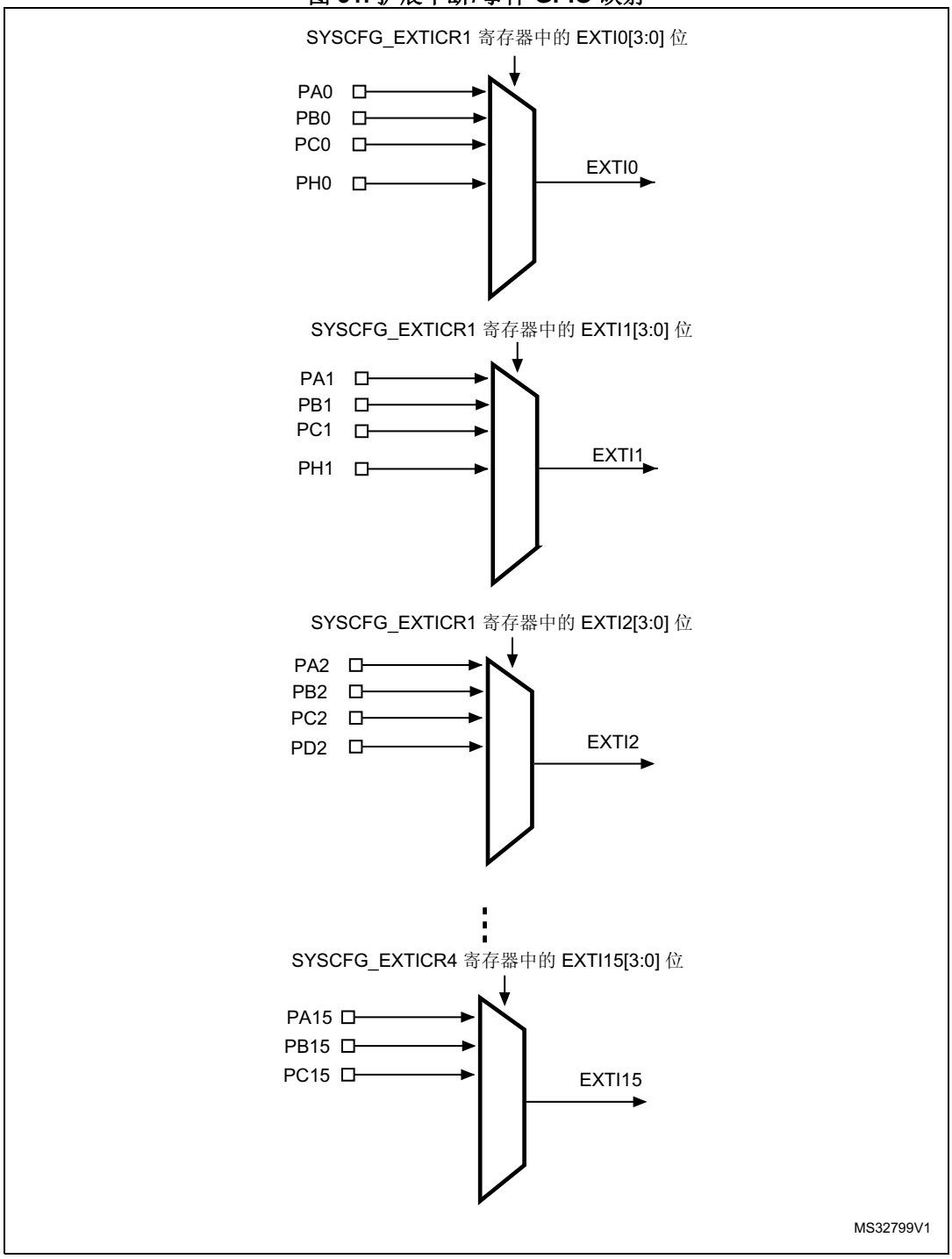
可将任一可配置线配置为软件中断/事件线。要产生软件中断，必须执行以下步骤。

1. 配置中断/事件线的屏蔽位（EXTI_IMR 和 EXTI_EMR）
2. 在软件中断寄存器设置相应的请求位 (EXTI_SWIER)。

13.4 EXTI 中断/事件线映射

STM32L0x3 中提供 30 根中断/事件线。GPIO 与 16 根可配置中断/事件线相连，如*图 31* 所示。

图 31. 扩展中断/事件 GPIO 映射



注： *关于可用 I/O 端口的列表，请参见数据手册。*

*表 52：EXTI 线连接*列出了这 30 根线的连接方式：

表 52. EXTI 线连接

EXTI 线	线源	线类型
0-15	GPIO	可配置
16	PVD	可配置
17	RTC 闹钟	可配置
18	USB 唤醒事件	无需配置
19	RTC 入侵、时间戳或 CSS_LSE	可配置
20	RTC 唤醒定时器	可配置
21	COMP1 输出	可配置
22	COMP2 输出	可配置
23	I2C1 唤醒	无需配置
24	I2C3 唤醒	无需配置
25	USART1 唤醒	无需配置
26	USART2 唤醒	无需配置
27	保留	
28	LPUART1 唤醒	无需配置
29	LPTIM1 唤醒	无需配置

13.5 EXTI 寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

外设寄存器必须按字（32 位）进行访问。

13.5.1 EXTI 中断屏蔽寄存器 (EXTI_IMR)

EXTI interrupt mask register

偏移地址：0x00
复位值：0x3F84 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IM29	IM28	Res.	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
		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
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:28 **IMx**：线 x 上的中断屏蔽 (Interrupt mask on line x)（x = 28 或 29）
0：屏蔽来自线 x 的中断请求
1：开放来自线 x 的中断请求

位 27 保留，必须保持复位值。

位 26:0 **IMx**：线 x 上的中断屏蔽 (Interrupt mask on line x)（x = 0 到 26）
0：屏蔽来自线 x 的中断请求
1：开放来自线 x 的中断请求

13.5.2 EXTI 事件屏蔽寄存器 (EXTI_EMR)

EXTI event mask register

偏移地址：0x04
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EM29	EM28	Res.	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
		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
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29:28 **EMx**：线 x 上的事件屏蔽 (Event mask on line x)（x = 28 或 29）
0：屏蔽来自线 x 的事件请求
1：开放来自线 x 的事件请求

位 27 保留，必须保持复位值。

位 26:0 **EMx**：线 x 上的事件屏蔽 (Event mask on line x)（x = 0 到 26）
0：屏蔽来自线 x 的事件请求
1：开放来自线 x 的事件请求

13.5.3 EXTI 上升沿触发选择寄存器 (EXTI_RTSR)

EXTI rising edge trigger selection register

偏移地址：0x08
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	Res.	RT17	RT16
									rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	RT16	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:23 保留，必须保持复位值。

位 22:19 **RTx**：线 x 的上升沿触发事件配置位 (Rising trigger event configuration bit of line x)（x = 19 到 22）
0：禁止输入线 x 上升沿触发（事件和中断）
1：允许输入线 x 上升沿触发（事件和中断）

位 18 保留，必须保持复位值。

位 17:0 **RTx**：线 x 的上升沿触发事件配置位 (Rising trigger event configuration bit of line x)（x = 0 到 17）
0：禁止输入线 x 上升沿触发（事件和中断）
1：允许输入线 x 上升沿触发（事件和中断）

注：可配置的唤醒线配置为边沿触发时，在这些线上不能出现毛刺信号。

如果在对 EXTI_RTSR 寄存器执行写入操作时可配置的中断线上产生上升沿，挂起位不会被置 1。

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。

13.5.4 下降沿触发选择寄存器 (EXTI_FTSR)

Falling edge trigger selection register

偏移地址：0x0C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	Res.	FT17	FT16
									rw	rw	rw	rw		rw	rw

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

位 31:23 保留，必须保持复位值。

位 22:19 **FTx**：线 x 的下降沿触发事件配置位 (Falling trigger event configuration bit of line x)（x = 19 到 22）
0：禁止输入线 x 下降沿触发（事件和中断）
1：允许输入线 x 下降沿触发（事件和中断）

位 18 保留，必须保持复位值。

位 17:0 **FTx**：线 x 的下降沿触发事件配置位 (Falling trigger event configuration bit of line x)（x = 0 到 17）
0：禁止输入线 x 下降沿触发（事件和中断）
1：允许输入线 x 下降沿触发（事件和中断）

注：可配置的唤醒线配置为边沿触发时，在这些线上不能出现毛刺信号。

如果在对 EXTI_RTSR 寄存器执行写入操作时可配置的中断线上产生下降沿，挂起位不会被置 1。

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。

13.5.5 EXTI 软件中断事件寄存器 (EXTI_SWIER)

EXTI software interrupt event register

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	SWI20	SWI19	Res.	SWI17	SWI16
									rw	rw	rw	rw		rw	rw

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

位 31:23 保留，必须保持复位值。

位 22:19 **SWIx**：线 x 上的软件中断 (Software interrupt on line x)（x = 19 到 22）

当此位为 0 时向其写入 1 会将 EXTI_PR 中相应的挂起位置 1。如果在 EXTI_IMR 和 EXTI_EMR 中允许在此线上产生该中断，则产生中断请求。

通过清除 EXTI_PR 中相应的位（写入 1），可清除此位。

位 18 保留，必须保持复位值。

位 17:0 **SWIx**：线 x 上的软件中断 (Software interrupt on line x)（x = 0 到 17）

当此位为 0 时向其写入 1 会将 EXTI_PR 中相应的挂起位置 1。如果在 EXTI_IMR 和 EXTI_EMR 中允许在此线上产生该中断，则产生中断请求。

通过清除 EXTI_PR 中相应的位（写入 1），可清除此位。

13.5.6 EXTI 挂起寄存器 (EXTI_PR)

EXTI pending register

偏移地址：0x14

复位值：未定义

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF22	PIF21	PIF20	PIF19	Res.	PIF17	PIF16
									rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

位 31:23 保留，必须保持复位值。

位 22:19 **PIFx**：线 x 上的挂起中断标志 (Pending interrupt flag on line x)（x = 19 到 22）

0：未发生触发请求

1：发生了选择的触发请求

当中断线上发生了选择的边沿事件时，此位被置 1。向此位写入 1 可将其清除，也可以通过更改边沿检测器的敏感性将其清除。

位 18 保留，必须保持复位值。

位 17:0 **PIFx**：线 x 上的挂起中断标志 (Pending interrupt flag on line x)（x = 0 到 17）

0：未发生触发请求

1：发生了选择的触发请求

当中断线上发生了选择的边沿事件时，此位被置 1。向此位写入 1 可将其清除，也可以通过更改边沿检测器的敏感性将其清除。

13.5.7 EXTI 寄存器映射

下表列出了 EXTI 寄存器映射和复位值。

表 53. 扩展中断/事件控制器寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_IMR**	Res.	Res.	IM[29:28]		Res.	IM[26:24]			IM[23:0]																							
	Reset value			1	1		1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_EMR**	Res.	Res.	EM[29:28]		Res.	EM[26:24]			EM[23:0]																							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_RTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[22:19]				Res.	RT[17:0]																	
	Reset value										0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[22:19]				Res.	FT[17:0]																	
	Reset value										0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI [22:19]				Res.	SWI[17:0]																	
	Reset value										0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PIF [22:19]				Res.	PIF[17:0]																	
	Reset value										0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

14 模数转换器（ADC）

14.1 简介

12 位 ADC 是逐次趋近型模数转换器。它具有多达 19 个复用通道，可测量来自 16 个外部源和 3 个内部源的信号。各种不同通道的 A/D 转换可在单次、连续、扫描或不连续采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

采用了高效的低功耗模式，在低频下可实现极低的功耗。

内置硬件过采样器，可提高模拟性能，同时还能减轻 CPU 进行相关计算的负担。

14.2 ADC 主要特性

- 高性能
  - 可配置 12 位、10 位、8 位或 6 位分辨率
  - ADC 转换时间：12 位分辨率对应的转换时间为 0.87 µs (1.14 MHz)，10 位分辨率对应的转换时间为 0.81 µs，若降低分辨率，可进一步缩短转换时间。
  - 自校准
  - 可编程采样时间
  - 内置数据一致性，可确保数据对齐
  - 支持 DMA
- 低功耗
  - 应用可降低 PCLK 频率从而以低功耗运行，同时仍可保持最优的 ADC 性能。例如，无论 PCLK 的频率如何，都会保持 1.0 µs 的转换时间
  - 等待模式：防止 ADC 在低频 PCLK 应用中溢出
  - 自动关闭模式：ADC 会自动断电，但正在进行转换时除外。这可大幅降低 ADC 的功耗。
- 模拟输入通道
  - 16 路外部模拟输入
  - 1 条用于内部温度传感器 (V_{SENSE}) 的通道
  - 1 条用于内部参考电压 (V_{REFINT}) 的通道
  - 1 条用于监视外部 VLCD 电源针脚的通道
- 可通过以下方式启动转换过程：
  - 通过软件
  - 通过极性可配置的硬件触发器（来自 TIM2、TIM3、TIM6、TIM21、TIM22 的内部定时器事件，或 GPIO 输入事件）

- 转换模式
  - 可转换单条通道，也可扫描一系列通道。
  - 单次模式会在每次触发时对选定的输入执行一次转换
  - 连续模式可连续转换选定的输入
  - 不连续转换模式
- 在采样结束、转换结束、序列转换结束以及发生模拟看门狗或溢出事件时产生中断
- 模拟看门狗
- 过采样器
  - 16 位数据寄存器
  - 过采样率可在 2x 到 256x 之间进行调整
  - 可编程数据最多可移位 8 位
- ADC 电源要求：1.65 V 到 3.6 V
- ADC 输入范围：$V_{SSA} \leq V_{IN} \leq$ VDDA

图 32 显示了 ADC 的框图。

14.3 ADC 引脚和内部信号

表 54. ADC 内部信号

内部信号名称	信号类型	说明
TRGx	输入	ADC 转换触发信号
V_{SENSE}	输入	内部温度传感器输出电压
V_{REFINT}	输入	内部参考电压输出电压
LCD_VLCD1	输入	1/3 V_{LCD}（LCD 配置为 1/3 偏置时） 1/4 V_{LCD}（LCD 配置为 1/4 偏置或 1/2 偏置时）

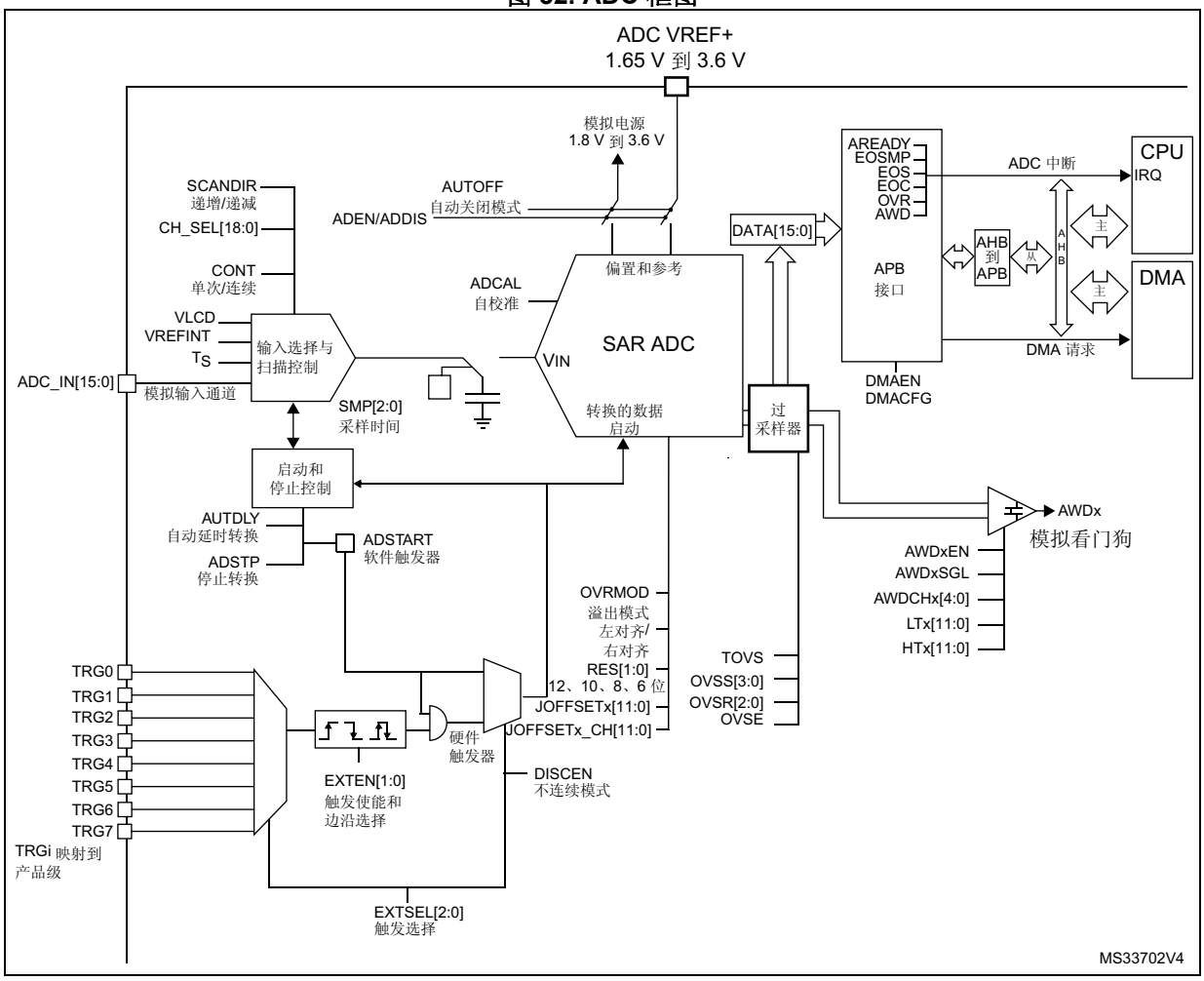
表 55. ADC 引脚

名称	信号类型	备注
V_{DDA}	模拟电源输入	ADC 的模拟电源和正参考电压，$V_{DDA} \geq$ VDD
V_{SSA}	模拟电源地输入	模拟电源接地引脚。电压必须等于 V_{SS}
ADC_IN[15:0]	模拟输入信号	16 个模拟输入通道

14.4 ADC 功能说明

给出了 ADC 的框图，*图 32* 列出了 ADC 的引脚说明。

图 32. ADC 框图



1. 请参见*表 58：外部触发器*，了解 TRGi 映射。

14.4.1 ADC 调压器 (ADVREGEN)

ADC 配有特定的内部调压器，使用 ADC 之前，调压器必须使能并处于稳定状态。

ADC 调压器的稳定时间完全由硬件管理，软件不需要进行处理。

ADC 操作完成后，可禁止 ADC (ADEN=0)。随后，可以禁止 ADC 调压器（请参见 ADC 调压器禁止序列），以节省更多电能。

注： *如果内部调压器已禁止，则会保持内部模拟校准。*

ADC 内部调压器的模拟参考

ADC 内部调压器使用缓冲的内部参考电压副本。如果主调压器处于正常运行模式（MR 模式，器件以运行模式或睡眠模式工作），此缓冲区始终会使能。如果主调压器处于低功耗运行模式（LPR 模式，器件以低功耗运行、低功耗睡眠或停止模式工作），则会禁止参考电压，ADC 无法再使用。

软件必须按照下述程序管理处于低功耗运行模式的 ADC：

1. 确保 ADC 已禁止 (ADEN = 0)。
2. 写入 ADVREFEN = 0。
3. 进入低功耗运行模式
4. 从低功耗运行模式恢复。
5. 检查 REGLPF 是否为 0。
6. 按照 *ADVREG 使能序列*一节中介绍的序列使能 ADC 调压器（ADC_CR 中 ADVREGEN= 1）。
7. 写入 ADC_CR ADEN = 1 并等待，直至 ADC_CR ADRDY = 1。
8. 写入 ADRDY = 1 并将其清零。

ADVREG 使能序列

可通过下列三种方法使能调压器：

- 写入 ADVREGEN=1。
- 通过写入 ADCAL=1 启动校准（ADVREGEN 位将自动置 1）
- 通过写入 ADEN=1 使能 ADC

ADVREG 禁止序列

若要禁止 ADC 调压器，需执行以下序列：

1. 确保 ADC 已禁止 (ADEN=0)
2. 写入 ADVREGEN=0

14.4.2 校准 (ADCAL)

ADC 具有校准功能。校准过程中，ADC 会计算校准系数，ADC 下一次掉电之前，会在内部对 ADC 应用此校准系数。校准过程中，应用不得使用 ADC，必须等待校准完成。

校准应在启动 A/D 转换之前进行。校准可消除偏移误差，由于制造工艺的不同，各芯片的偏移误差也有所不同。

校准通过软件将 ADCAL 位置 1 发起。仅当 ADC 禁止 (ADEN=0) 后，才能发起校准。ADCAL 位在所有校准序列过程中保持为 1。校准完成后，此位会立即由硬件清零。随后，可从 ADC_DR 寄存器中读取校准系数（位 6 到位 0）。

若禁止了 ADC (ADEN=0)、或禁止了 ADC 参考电压 (ADVREGEN = 0)，则会保留内部模拟校准。如果 ADC 工作条件发生变化（V_{DDA} 变化是造成 ADC 偏移变化的主要原因，并会导致温度发生小幅变化），建议重新运行校准周期。

在下列情况下，校准系数会丢失：

- 产品处于待机模式（ADC 掉电）
- ADC 外设复位。

在低功耗运行、低功耗睡眠和停止这三种低功耗模式下，会保留校准系数。

仍可通过软件保存并恢复校准系数，以节省 ADC 重启时间（前提是 ADC 掉电期间的温度和电压保持稳定）。

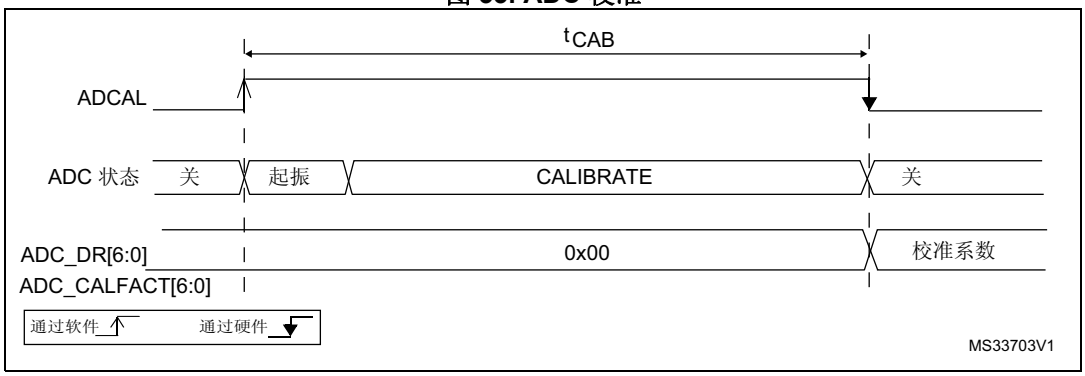
如果 ADC 已使能但未进行转换（ADEN=1 且 ADSTART=0），可写入校准系数。随后，下次转换启动时，校准系数会自动添加到模拟 ADC 中。这一载入过程是透明的，不会对转换的启动造成延迟。

校准软件程序

1. 确保 ADEN=0
2. 将 ADCAL 置 1
3. 等待，直至 ADCAL=0（或直至 EOCAL=1）。如果已通过将 ADC_IER 寄存器中的 EOCALIE 位置 1 来使能中断，可通过中断进行处理
4. 校准系数可从 ADC_DR 或 ADC_CALFACT 寄存器的位 6:0 读取。

有关代码示例，请参见 *A.8.1：校准代码示例*。

图 33. ADC 校准



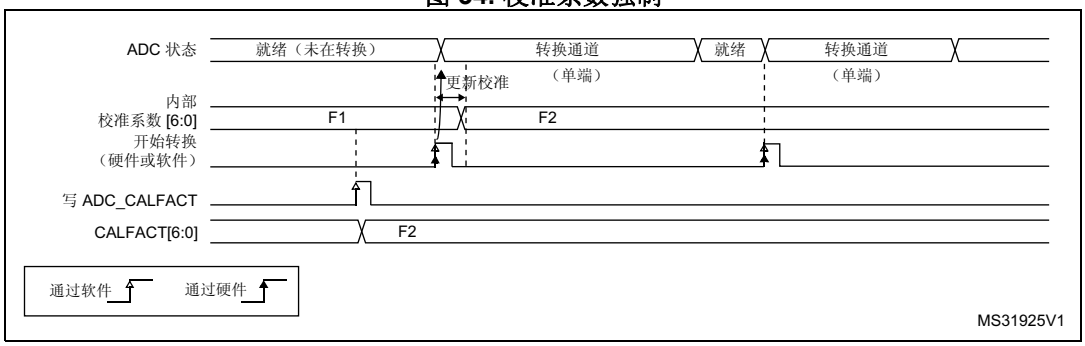
如果 ADC 调压器之前未置 1，则将 ADCAL 置 1 后，ADC 调压器会自动使能（ADVREGEN 位会由硬件自动置 1）。在这种情况下，由于要考虑 ADC 调压器的稳定时间，因此 ADC 校准时间会比较长。

校准结束时，ADC 调压器仍保持使能状态。

校准系数强制软件程序

1. 确保 ADEN= 1 且 ADSTART =0（ADC 启动时没有进行任何转换）
2. 将保存的校准系数写入 ADC_CALFACT
3. 启动新的转换后，将立即使用校准系数。

图 34. 校准系数强制



14.4.3 ADC 开关控制（ADEN、ADDIS、ADRDY）

MCU 上电时，会禁止 ADC 并进入掉电模式 (ADEN=0)。

如*图 35* 所示，ADC 在开始精确转换之前需要一段稳定时间 t_{STAB}。

以下两个控制位可用于使能或禁止 ADC：

- 将 ADEN 置 1 可使能 ADC。ADC 准备就绪后，ADRDY 标志会立即置 1。
- 将 ADDIS 置 1 可禁止 ADC 并使 ADC 进入掉电模式。随后，ADC 被完全禁止后，ADEN 位和 ADDIS 位会自动由硬件清零。

如果 ADC 调压器之前未置 1，则将 ADEN 置 1 后，ADC 调压器会自动使能（ADVREGEN 位会由硬件自动置 1）。在这种情况下，由于要考虑 ADC 调压器的稳定时间，因此 ADC 稳定时间 t_{STAB} 会比较长。

随后可通过将 ADSTART 置 1（请参见*第 288 页上的第 14.5 节：外部触发转换和触发极性（EXTSEL，EXTEN）*）开始进行转换，如果触发器已使能，也可在发生外部触发事件时开始进行转换。

请按照以下流程使能 ADC：

1. 将 ADC_ISR 寄存器中的 ADRDY 位编程为 1，将此位清零。
2. 将 ADC_CR 寄存器中的 ADEN 位置 1。
3. 等待，直至 ADC_ISR 寄存器中的 ADRDY=1（ADRDY 会在 ADC 启动时间后置 1）。如果已通过将 ADC_IER 寄存器中的 ADRDYIE 位置 1 来使能中断，可通过中断进行处理。

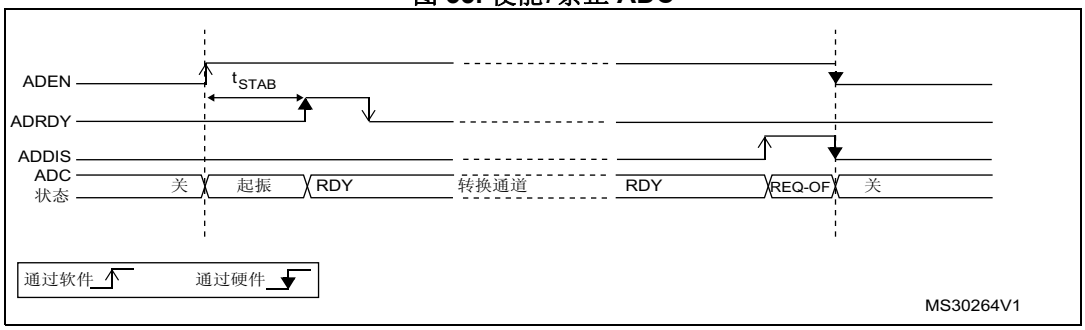
有关代码示例，请参见 *A.8.2：ADC 使能序列代码示例*。

请按照以下流程禁止 ADC：

1. 检查 ADC_CR 寄存器中的 ADSTART 是否为 0，以确保当前未执行任何转换。如有需要，可向 ADC_CR 寄存器中的 ADSTP 位写入 1 并等待此位读取值为 0，以此停止正在进行的转换。
2. 将 ADC_CR 寄存器中的 ADDIS 位置 1。
3. 如果应用要求，可等待 ADC_CR 寄存器中的 ADEN=0，这表明 ADC 已完全禁止（ADEN=0 后，ADDIS 会自动复位）。
4. 将 ADC_ISR 寄存器中的 ADRDY 位编程为 1，将此位清零（可选）。

有关代码示例，请参见 *A.8.3：ADC 禁止序列代码示例*。

图 35. 使能/禁止 ADC

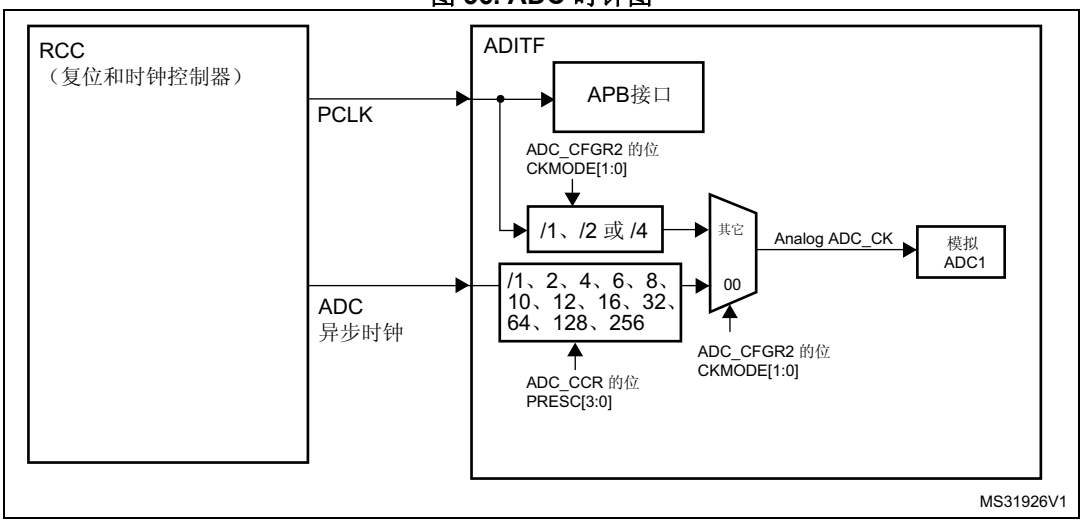


注：在自动关闭模式下 (AUTOFF=1)，上电/掉电阶段是由硬件自动执行的，不会将 ADRDY 标志置 1。

14.4.4 ADC 时钟（CKMODE、PRESC[3:0]、LFMEN）

ADC 采用双时钟域架构，因此，ADC 可由独立于 APB 时钟 (PCLK) 的时钟提供时钟（ADC 异步时钟）。

图 36. ADC 时钟图



1. 请参见*第 160 页上的第 7 节：复位和时钟控制 (RCC)*，了解 PCLK 和 ADC 异步时钟的使能方式。

模拟 ADC 的输入时钟可在两个不同的时钟源之间进行选择（请参见*图 36：ADC 时钟图*，了解 PCLK 和 ADC 异步时钟的使能方式）：

a) ADC 时钟可以是名为“ADC 异步时钟”的特定时钟源，该时钟源独立于 APB 时钟，并与 APB 时钟异步。

   更多有关生成该时钟源的信息，请参见 RCC 部分。

   要选择此时钟图，必须将 ADC_CFGR2 寄存器的 CKMODE[1:0] 位复位。

b) ADC 时钟可由 ADC 总线接口的 APB 时钟除以一个可编程因子（2 或 4，具体根据 CKMODE[1:0] 位而定）来提供。

   要选择此时钟图，ADC_CFGR2 寄存器的 CKMODE[1:0] 位不得为“00”。

有关代码示例，请参见 *A.8.4：MCO 时钟选择代码示例*。

在选项 a) 中，对 ADC_CCR 寄存器中的 PRESC[3:0] 位进行编程时，生成的 ADC 时钟最后会除以预分频系数（1、2、4、6、8、12、16、32、64、128、256）。

选项 a) 的优点在于无论选择哪种 APB 时钟图，都可以达到最大 ADC 时钟频率。

选项 b) 的优势在于绕过了时钟域重新同步。如果 ADC 由定时器触发，并且应用要求 ADC 精确触发（不存在任何不确定性），可使用此选项（否则，重新同步两个时钟域会为触发时刻带来不确定性）。

表 56. 触发与转换开始之间的延迟

ADC 时钟源	CKMODE[1:0]	触发事件与 转换开始之间的延迟
HSI16 MHz 时钟	00	延迟是不确定的（存在抖动）
PCLK 2 分频	01	延迟是确定的（无抖动），等于 4.25 个 ADC 时钟周期
PCLK 4 分频	10	延迟是确定的（无抖动），等于 4.125 个 ADC 时钟周期
PCLK 1 分频	11	延迟是确定的（无抖动），等于 4.5 个 ADC 时钟周期

注意： 选择 CKMODE[1:0]=11（PCLK 1 分频）时，用户必须确保 PCLK 的占空比为 50%。为此，在 RCC 中，用户必须选择占空比为 50% 的系统时钟，并且必须以旁路模式在 RCC 内配置 APB 预分频系数（请参见 RCC 部分）。对于内部源时钟，这只意味着 AHB 和 APB 预分频器不会对时钟进行分频。

低频

如果选择的模拟 ADC 时钟频率低于 3.5 MHz，必须先将 ADC_CCR 寄存器中的 LFMEN 位置 1，从而使能低频模式

14.4.5 配置 ADC

如果 ADC 已禁止，必须通过软件对 ADC_CR 寄存器中的 ADCAL 位和 ADEN 位执行写操作（ADEN 必须为 0）。

只有在 ADC 已使能、并且没有待处理的禁止 ADC 的请求时（ADEN = 1，且 ADDIS = 0），软件才必须只对 ADC_CR 寄存器中的 ADSTART 位和 ADDIS 位执行写操作。

对于 ADC_IER、ADC_CFGRi、ADC_SMPR、ADC_TR、ADC_CHSELR 和 ADC_CCR 寄存器中的其他控制位，只有在 ADC 已使能 (ADEN = 1)，且没有正在进行的转换 (ADSTART = 0)，软件才可以执行写操作。

如果 ADC 已使能（很可能正在进行转换）、并且没有待处理的禁止 ADC 的请求时（ADSTART = 1 且 ADDIS = 0），软件才必须只对 ADC_CR 寄存器中的 ADSTP 位执行写操作。

注： *未采取硬件保护机制来防止软件执行上述规则禁止的写操作。如果发生此类禁止的写访问，ADC 可能会进入未定义状态。要在这种情况下恢复正确的操作，必须禁止 ADC（将 ADEN 以及 ADC_CR 寄存器中的所有位都清零）。*

14.4.6 通道选择（CHSEL、SCANDIR）

复用通道多达 19 条：

- 16 路来自 GPIO 引脚 (ADC_IN0...ADC_IN15) 的模拟输入
- 3 路内部模拟输入（温度传感器、内部参考电压、LCD_VLCD1 通道）

可转换单条通道，也可以自动扫描一系列通道。

待转换通道的顺序必须在 ADC_CHSELR 通道选择寄存器中进行编程：每条模拟输入通道都有专用的选择位 (CHSEL0...CHSEL18)。

要配置通道的扫描顺序，可对 ADC_CFGR1 寄存器中的 SCANDIR 位进行编程：

- SCANDIR=0：正向扫描通道 0 到通道 18
- SCANDIR=1：反向扫描通道 18 到通道 0

温度传感器、V_{REFINT} 和 LCD_VLCD1 内部通道

温度传感器连接至通道 ADC_IN18。内部参考电压 V_{REFINT} 连接至通道 ADC_IN17。LCD_VLCD1 通道连接至通道 ADC_IN16。

14.4.7 可编程采样时间 (SMP)

开始转换之前，ADC 需要在待测量电压源与 ADC 内置采样电容之间建立直接连接。该采样时间必须足以使输入电压源为采样电容充电并将电容保持在输入电压水平。

使用可编程采样时间后，可根据输入电压源的输入电阻调整转换速度。

ADC 会在数个 ADC 时钟周期内对输入电压进行采样，时钟周期数可使用 ADC_SMPR 寄存器中的 SMP[2:0] 位进行修改。

此可编程采样时间是所有通道共用的。如果应用要求，可通过软件在两次转换之间更改和调整此采样时间。

总转换时间的计算公式如下：

t_{CONV} = 采样时间 + 12.5 x ADC 时钟周期

示例：

如果 ADC_CLK = 16 MHz，采样时间为 1.5 个 ADC 时钟周期：

t_{CONV} = 1.5 + 12.5 = 14 个 ADC 时钟周期 = 0.875 µs

ADC 通过将 EOSMP 标志置 1 来指示采样阶段结束。

14.4.8 单次转换模式 (CONT=0)

在单次转换模式下，ADC 会执行单次转换序列，对所有通道进行一次转换。当 ADC_CFGR1 寄存器中的 CONT=0 时，会选择此模式。可通过以下方式开始转换：

- 将 ADC_CR 寄存器中的 ADSTART 位置 1
- 硬件触发事件

在序列中，每次转换完成后：

- 转换后的数据会存储在 16 位 ADC_DR 寄存器中
- EOC（转换结束）标志置 1
- EOCIE 位置 1 时将产生中断

转换序列完成后：

- EOSEQ（序列结束）标志置 1
- EOSEQIE 位置 1 时将产生中断

随后，ADC 会停止工作，直至发生新的外部触发事件或 ADSTART 位再次置 1。

注： *要转换单个通道，可将序列长度编程为 1。*

14.4.9 连续转换模式 (CONT=1)

在连续转换模式下，如果发生软件或硬件触发事件，ADC 会执行转换序列，对所有通道进行一次转换，随后会自动重启并持续执行相同的转换序列。当 ADC_CFGR1 寄存器中的 CONT=1 时，会选择此模式。可通过以下方式开始转换：

- 将 ADC_CR 寄存器中的 ADSTART 位置 1
- 硬件触发事件

在序列中，每次转换完成后：

- 转换后的数据会存储在 16 位 ADC_DR 寄存器中
- EOC（转换结束）标志置 1
- EOCIE 位置 1 时将产生中断

转换序列完成后：

- EOSEQ（序列结束）标志置 1
- EOSEQIE 位置 1 时将产生中断

随后，会立即重启新序列，ADC 会继续重复执行转换序列。

注： *要转换单个通道，可将序列长度编程为 1。*

不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

14.4.10 开始转换 (ADSTART)

软件通过将 ADSTART 置 1 的方式开始进行 ADC 转换。

ADSTART 置 1 后：

- 在 EXTEN = 00 时会立即开始转换（软件触发）
- 在 EXTEN ≠ 00 时，会在所选硬件触发器的下一有效边沿开始转换。

ADSTART 位还用于指示当前是否正在进行 ADC 操作。可以在 ADSTART=0 时重新配置 ADC，从而指示 ADC 处于空闲状态。

ADSTART 位由硬件清零：

- 在使用软件触发的单次模式下（CONT=0，EXTEN=00）
  - 只要转换序列结束 (EOSEQ=1) 就清零
- 在使用软件触发的不连续模式下（CONT=0、DISCEN=1、EXTEN=00）
  - 转换结束时 (EOC=1) 清零
- 在所有情况下（CONT=x、EXTEN=XX）
  - 执行由软件调用的 ADSTP 程序之后（参见*第 288 页上的第 14.4.12 节：停止正在进行的转换 (ADSTP)*）清零

注： *在连续模式下 (CONT=1)，由于序列会自动重新启动，因此，当 EOSEQ 标志置 1 时，ADSTART 位不会清零。*

如果在单次模式下选择了硬件触发（CONT=0，且 EXTEN ≠ 00），则 EOSEQ 置 1 时，ADSTART 不会由硬件清零。这样，便无需通过软件将 ADSTART 再次置 1，并可确保不会错过下一触发事件。

14.4.11 时序

从转换开始到转换结束所经过的时间是配置的采样时间与逐次趋近时间（具体视数据分辨率而定）的总和：

$$t_{ADC} = t_{SMPL} + t_{SAR} = [\ 1.5\ |_{min} + 12.5\ |_{12bit}\ ] \times t_{ADC_CLK}$$

$$t_{ADC} = t_{SMPL} + t_{SAR} = 93.8\ ns\ |_{min} + 781.3\ ns\ |_{12bit} = 0.875\ \mu s\ |_{min}$$（对于 $f_{ADC_CLK} = 16\ MHz$）

图 37. 模数转换时间

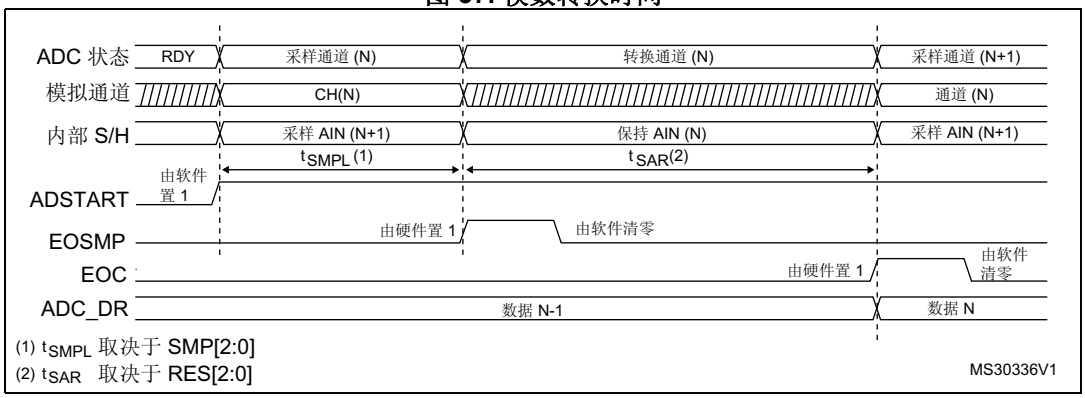
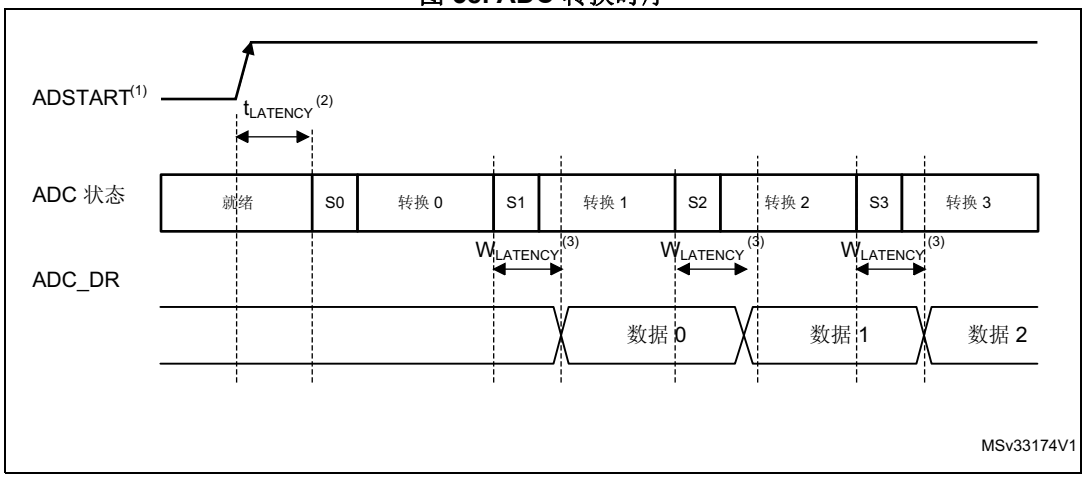


图 38. ADC 转换时序



1. EXTEN =00 或 EXTEN ≠ 00
2. 触发延迟（更多详细信息，请参见数据手册）
3. ADC_DR 寄存器写入延迟（更多详细信息，请参见数据手册）

14.4.12 停止正在进行的转换 (ADSTP)

可通过软件将 ADC_CR 寄存器中的 ADSTP 置 1，停止任何正在进行的转换。

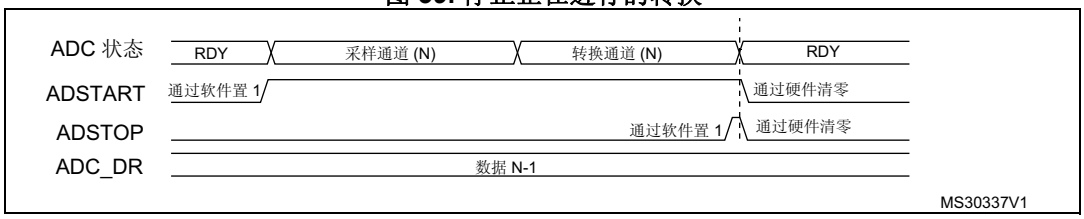
此操作会复位 ADC 操作，ADC 将处空闲状态，准备好进行新操作。

如果 ADSTP 位由软件置 1，则会中止任何正在进行的转换，并会丢弃转换结果（ADC_DR 寄存器不会更新为当前转换结果）。

扫描序列也会中止并会复位（这意味着重启 ADC 将重新开始新的序列）。

此程序完成后，ADSTP 位和 ADSTART 位均会由硬件清零，软件必须等待 ADSTART=0，然后才能开始进行新的转换。

图 39. 停止正在进行的转换



14.5 外部触发转换和触发极性（EXTSEL，EXTEN）

可通过软件或外部事件（定时器捕获事件）触发转换或转换序列。如果 EXTEN[1:0] 控制位不等于“0b00”，那么外部事件能够触发具有所选极性的转换。软件将 ADSTART 位置 1 后，触发选择将立即生效。

在转换进行时发生的硬件触发会被忽略。

如果位 ADSTART=0，则会忽略发生的任何硬件触发。

表 57 提供 EXTEN[1:0] 值与触发极性之间的对应关系。

表 57. 配置触发极性

源	EXTEN[1:0]
禁止触发检测	00
在上升沿检测	01
在下降沿检测	10
在上升沿和下降沿均检测	11

注： *仅当 ADC 未进行转换 (ADSTART= 0) 时，才可以更改外部触发的极性。*

EXTSEL[2:0] 控制位用于选择 8 个可能的事件中哪一事件可触发转换。

表 58 给出了可用于常规转换的外部触发。

可将 ADC_CR 寄存器中的 ADSTART 位置 1，从而生成软件源触发事件。

表 58. 外部触发器

名称	源	EXTSEL[2:0]
TRG0	TIM6_TRGO	000
TRG1	TIM21_CH2	001
TRG2	TIM2_TRGO	010
TRG3	TIM2_CH4	011
TRG4	TIM22_TRGO	100
TRG5(1)	TIM2_CH3	101
TRG6	TIM3_TRGO	110
TRG7	EXTI 线 11	111

1. 在第 3 类器件以外的其他所有类别器件上可用。

注：仅当 ADC 未进行转换 (ADSTART= 0) 时，才可以更改触发选择。

14.5.1 不连续模式 (DISCEN)

可将 ADC_CFGR1 寄存器中的 DISCEN 位置 1 来使能此模式。

在该模式下 (DISCEN=1)，需要通过硬件或软件触发事件开始序列中定义的各个转换。相反，如果 DISCEN=0，单个硬件或软件触发事件会连续启动序列中定义的所有转换。

示例：

- DISCEN=1，待转换通道 = 0、3、7、10
  - 第一次触发：转换通道 0 并生成 EOC 事件
  - 第二次触发：转换通道 3 并生成 EOC 事件
  - 第三次触发：转换通道 7 并生成 EOC 事件
  - 第四次触发：转换通道 10 并同时生成 EOC 和 EOSEQ 事件
  - 第五次触发：转换通道 0 并生成 EOC 事件
  - 第六次触发：转换通道 3 并生成 EOC 事件
  - ...
- DISCEN=0，待转换通道 = 0、3、7、10
  - 第一次触发：转换整个序列：通道 0、然后是通道 3、7 和 10。每次转换都会生成 EOC 事件，最后一次转换还会生成 EOSEQ 事件。
  - 任何后续触发事件都将重启整个序列。

注：不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

14.5.2 可编程分辨率 (RES) - 快速转换模式

可通过降低 ADC 分辨率缩短转换时间 (t_{SAR})。

通过对 ADC_CFGR1 寄存器中的 RES[1:0] 位进行编程，可将分辨率配置为 12 位、10 位、8 位或 6 位。对于不要求使用高数据精度的应用，分辨率越低，转换时间越短。

注： *RES[1:0] 位必须在 ADEN 位复位后才能进行更改。*

转换结果的宽度始终为 12 位，任何未使用的 LSB 位都会读为零。

降低分辨率可缩短逐次趋近步骤所需的转换时间，如*表 59* 所示。

表 59. tSAR 与分辨率的对应关系

RES[1:0] 位	t_{SAR}（ADC 时钟周期）	t_{SAR} (ns)，f_{ADC} = 16 MHz	$t_{SMPL\ (min)}$（ADC 时钟周期）	t_{CONV}（ADC 时钟周期）（使用最短 t_{SMPL}）	t_{CONV} (µs)，f_{ADC} = 16 MHz
12	12.5	781 ns	1.5	14	875 ns
10	11.5	719 ns	1.5	13	812 ns
8	9.5	594 ns	1.5	11	688 ns
6	7.5	469 ns	1.5	9	562 ns

14.5.3 转换结束、采样阶段结束（EOC、EOSMP 标志）

ADC 指示每个转换结束 (EOC) 事件。

新的转换数据结果出现在 ADC_DR 寄存器中后，ADC 会立即将 ADC_ISR 寄存器中的 EOC 标志置 1。如果 ADC_IER 寄存器中的 EOCIE 位置 1，可产生中断。EOC 标志可通过由软件向其写入 1 或读取 ADC_DR 寄存器的方式来清零。

ADC 还通过将 ADC_ISR 寄存器中的 EOSMP 标志置 1 来指示采样阶段结束。EOSMP 标志可通过由软件向其写入 1 来清零。如果 ADC_IER 寄存器中的 EOSMPIE 位置 1，可产生中断。

此中断用于使处理与转换进行同步。通常情况下，可在转换阶段的隐藏时间中访问模拟复用器，这样在下次采样开始时便可放置好复用器。

注： *由于采样结束与转换结束之间只留有非常短的时间，因此建议使用轮询或 WFE 指令，而不建议使用中断和 WFI 指令。*

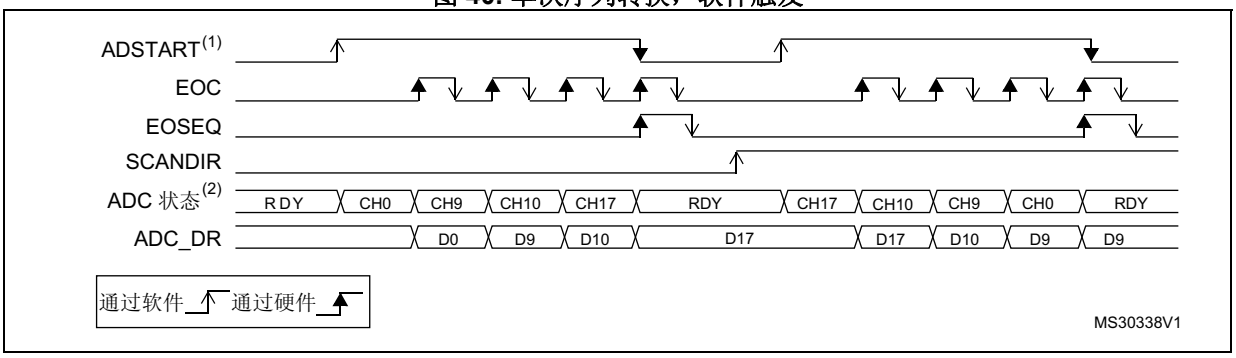
14.5.4 转换序列结束（EOSEQ 标志）

每次序列 (EOSEQ) 事件结束时，ADC 都会通知应用。

转换序列的上一数据结果出现在 ADC_DR 寄存器中时，ADC 会立即将 ADC_ISR 寄存器中的 EOSEQ 标志置 1。如果 ADC_IER 寄存器中的 EOSEQIE 位置 1，可产生中断。EOSEQ 标志可通过由软件向其写入 1 的方式来清零。

14.5.5 时序图示例（单次/连续模式硬件/软件触发）

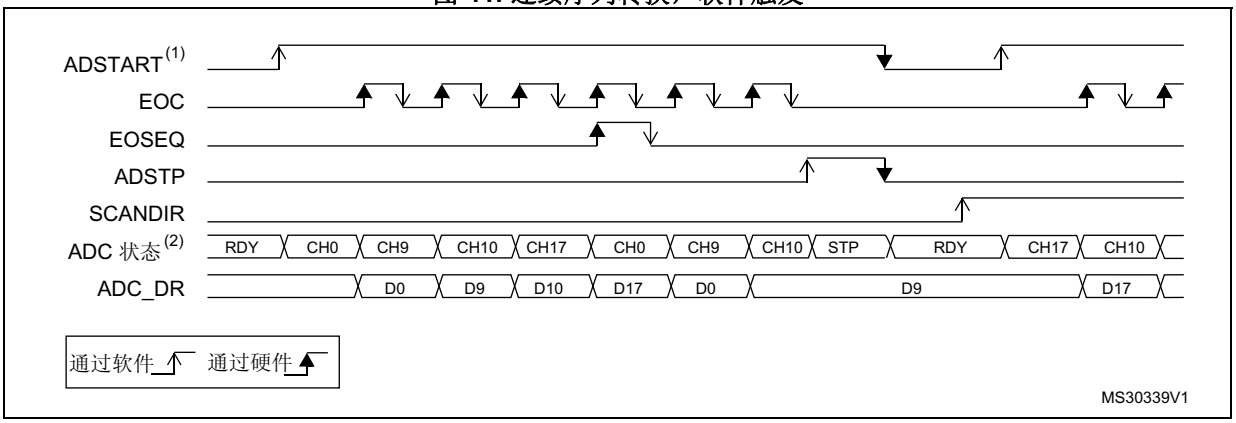
图 40. 单次序列转换，软件触发



1. EXTEN=00，CONT=0
2. CHSEL=0x20601，WAIT=0，AUTOFF=0

有关代码示例，请参见 *A.8.5：单次转换序列代码示例 - 软件触发*。

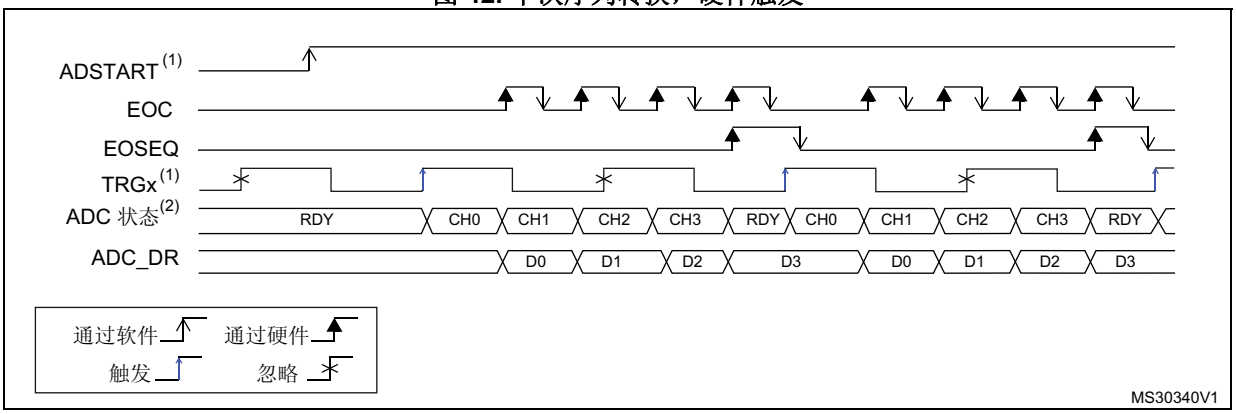
图 41. 连续序列转换，软件触发



1. EXTEN=00，CONT=1，
2. CHSEL=0x20601，WAIT=0，AUTOFF=0

有关代码示例，请参见 *A.8.6：连续转换序列代码示例 - 软件触发*。

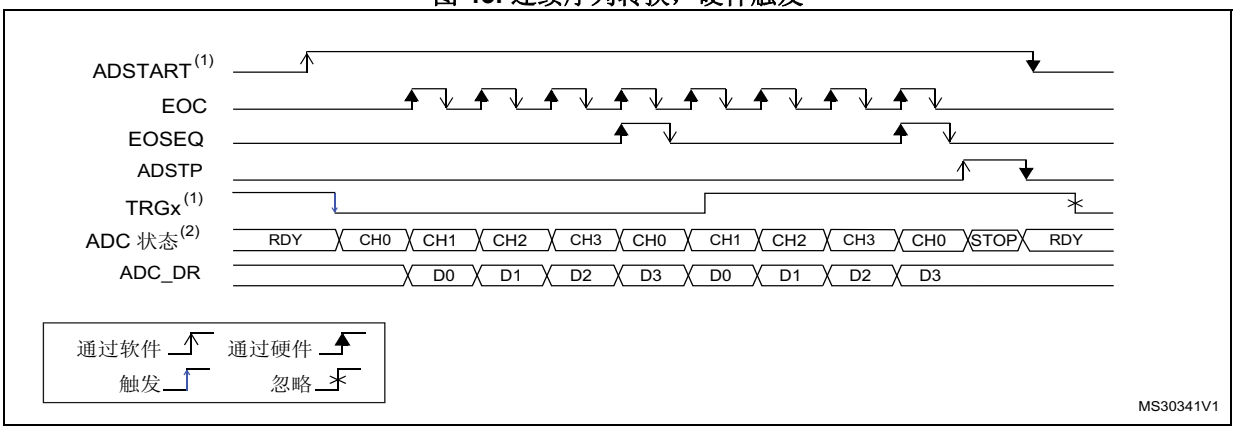
图 42. 单次序列转换，硬件触发



1. EXTSEL=TRGx（过频），EXTEN=01（上升沿），CONT=0
2. CHSEL=0xF，SCANDIR=0，WAIT=0，AUTOFF=0

有关代码示例，请参见 *A.8.7：单次转换序列代码示例 - 硬件触发*。

图 43. 连续序列转换，硬件触发



1. EXTSEL=TRGx，EXTEN=10（下降沿），CONT=1
2. CHSEL=0xF，SCANDIR=0，WAIT=0，AUTOFF=0

有关代码示例，请参见 *A.8.8：连续转换序列代码示例 - 硬件触发*。

14.6 数据管理

14.6.1 数据寄存器和数据对齐（ADC_DR、ALIGN）

每次转换结束时（发生 EOC 事件时），转换后数据的结果都会存储在宽度为 16 位的 ADC_DR 数据寄存器中。

ADC_DR 的格式取决于配置的数据对齐方式和分辨率。

ADC_CFGR1 寄存器中的 ALIGN 位用于选择转换后存储的数据的对齐方式。数据可右对齐 (ALIGN=0) 或左对齐(ALIGN=1)，如*图 44* 所示。

图 44. 数据对齐方式和分辨率（过采样已禁止：OVSE = 0）

ALIGN	RES	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0	0x0				DR[11:0]											
	0x1	0x00						DR[9:0]									
	0x2	0x00								DR[7:0]							
	0x3	0x00										DR[5:0]					
1	0x0	DR[11:0]												0x0			
	0x1	DR[9:0]										0x00					
	0x2	DR[7:0]								0x00							
	0x3	0x00								DR[5:0]						0x0	

MS30342V1

14.6.2 ADC 溢出（OVR、OVRMOD）

如果转换后的数据未由 CPU 或 DMA 及时读取，在新转换生成数据之前，会由溢出标志 (OVR) 指示数据溢出事件。

如果新转换完成时 EOC 标志仍为“1”，则 ADC_ISR 寄存器中的 OVR 标志会置 1。如果 ADC_IER 寄存器中的 OVRIE 位置 1，可产生中断。

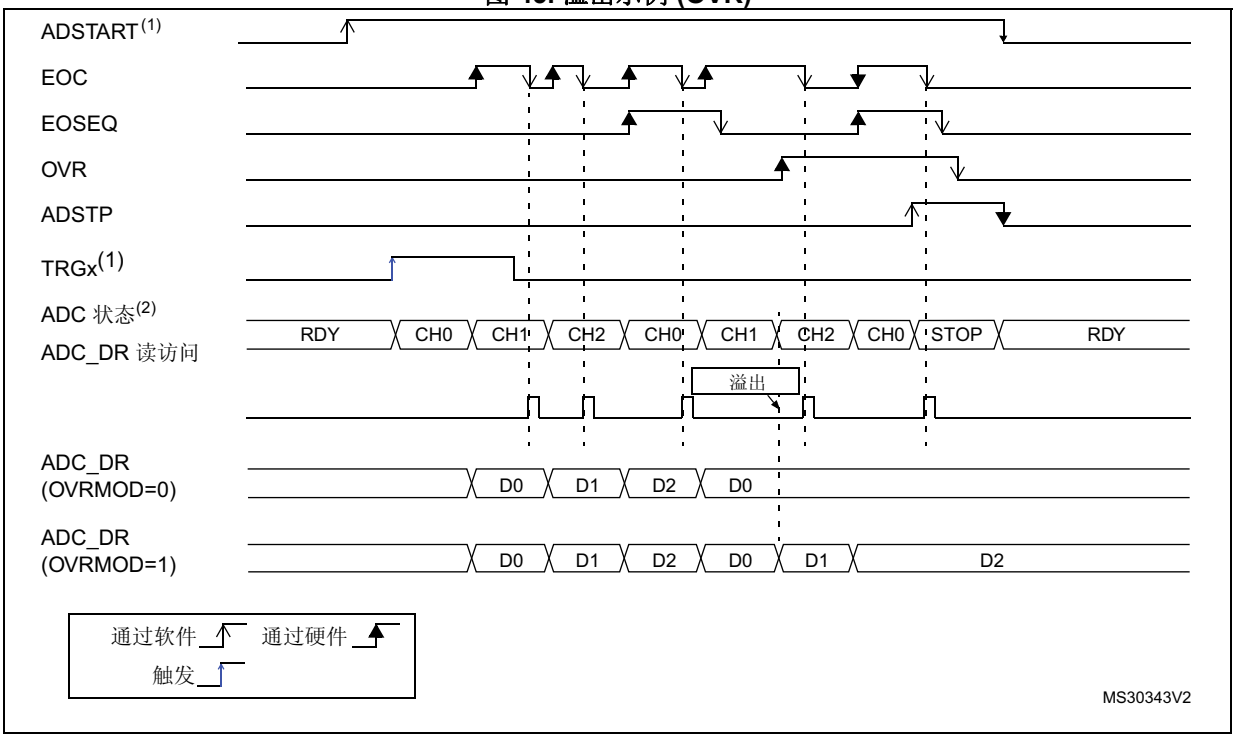
如果发生溢出情况，ADC 会保持工作状态并可继续进行转换，除非通过软件将 ADC_CR 寄存器中的 ADSTP 位置 1，从而停止并复位序列。

OVR 标志可通过由软件向其写入 1 的方式来清零。

可对 ADC_CFGR1 寄存器中的 OVRMOD 位进行编程，从而配置发生溢出事件时是要保留数据还是要覆盖数据：

- OVRMOD=0
  - 溢出事件会保留数据寄存器的数据，防止其被覆盖：会保留原数据，并会丢弃新的转换结果。如果 OVR 保持为 1，可继续进行转换，但会丢弃所得的数据。
- OVRMOD=1
  - 数据寄存器会由上一次转换结果覆盖，之前未读取的数据会丢失。如果 OVR 保持为 1，可继续进行转换，ADC_DR 寄存器始终包含最新转换得出的数据。

图 45. 溢出示例 (OVR)



14.6.3 在不使用 DMA 的情况下管理转换的数据序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，软件必须使用 EOC 标志及其相关中断来处理各个数据结果。每次转换完成时，都会将 ADC_ISR 寄存器中的 EOC 位置 1，并可读取 ADC_DR 寄存器。ADC_CFGR1 寄存器中的 OVRMOD 位应配置为 0，以便将溢出事件作为错误进行管理。

14.6.4 在不使用 DMA 且不发生溢出的情况下管理转换的数据

使 ADC 在转换一条或多条通道时无需在每次转换后都读取数据可能会很有用。在这种情况下，OVRMOD 位必须配置为 1，OVR 标志应被软件忽略。如果 OVRMOD=1，溢出事件不会阻止 ADC 继续进行转换，ADC_DR 寄存器始终包含最新的转换数据。

14.6.5 使用 DMA 管理转换的数据

由于转换得出的所有通道值都存储在单个数据寄存器中，因此，在转换多条通道时使用 DMA 可提高效率。这样可以避免存储在 ADC_DR 寄存器中的转换数据结果丢失。

若 DMA 模式已使能（ADC_CFGR1 寄存器中的 DMAEN 位置 1），则会在每个通道转换后生成 DMA 请求。这样便可将转换后的数据从 ADC_DR 寄存器传输到由软件选择的目标位置。

注： *ADC 校准阶段完成后，必须将 ADC_CFGR1 寄存器中的 DMAEN 位置 1。*

尽管如此，如果因 DMA 无法及时处理 DMA 传输请求而发生溢出 (OVR=1)，ADC 会停止生成 DMA 请求，新转换对应的数据不会通过 DMA 进行传输。这意味着可将传输到 RAM 的所有数据都视为有效数据。

根据 OVRMOD 位的配置，可保留或覆盖数据（请参见*第 293 页上的第 14.6.2 节：ADC 溢出（OVR、OVRMOD）*）。

DMA 传输请求会禁止，直至软件将 OVR 位清零。

根据应用用途的不同，推荐使用两种不同的 DMA 模式，并使用 ADC_CFGR1 寄存器中的 DMACFG 位配置相应的模式：

- DMA 单次模式 (DMACFG=0)。
  如果通过编程将 DMA 设置为传输固定数目的数据字，应选择此模式。
- DMA 循环模式 (DMACFG=1)
  如果通过编程将 DMA 设置为循环模式或双缓冲区模式，应选择此模式。

DMA 单次模式 (DMACFG=0)

在该模式下，每次出现新的转换数据字时，ADC 都会生成 DMA 传输请求，DMA 到达最后一个 DMA 传输操作时（发生 DMA_EOT 中断，请参见*第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*），即使转换已再次开始，ADC 也会停止生成 DMA 请求。

有关代码示例，请参见 *A.8.9：DMA 单触发模式序列代码示例*。

DMA 传输完成后（在 DMA 控制器中配置的所有传输操作均已完成后）：

- ADC 数据寄存器的内容会冻结。
- 任何正在进行的转换都会中止，其部分结果会被丢弃
- 不会将任何新的 DMA 请求发送到 DMA 控制器。如果仍存在已开始的转换，这样可避免生成溢出错误。
- 扫描序列会停止并复位
- DMA 会停止

DMA 循环模式 (DMACFG=1)

在该模式下，每次数据寄存器中出现新的转换数据字时，ADC 都会生成 DMA 传输请求，即使 DMA 已到达最后一次 DMA 传输操作也不例外。这样可将 DMA 配置为循环模式，从而处理连续的模拟输入数据流。

有关代码示例，请参见 *A.8.10：DMA 循环模式序列代码示例*。

14.7 低功耗特性

14.7.1 等待模式转换

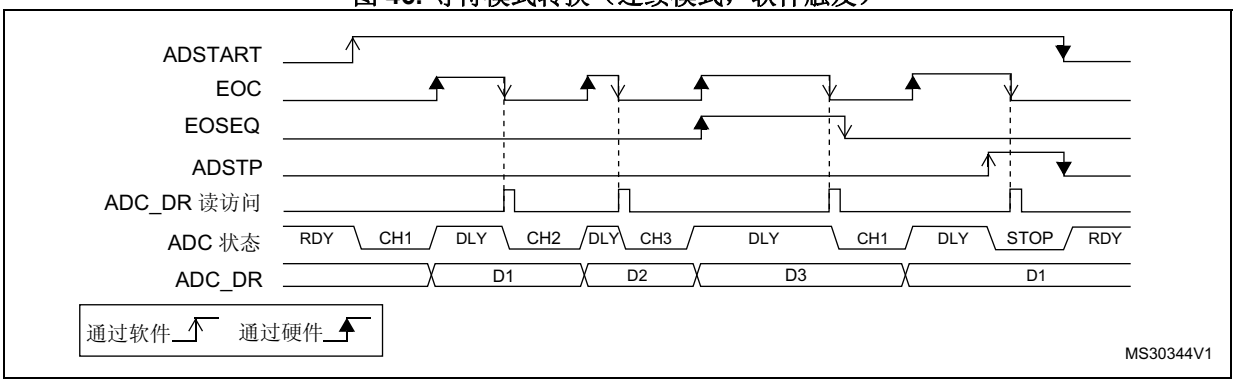
等待模式转换可用于简化软件，并可优化采用低频时钟的应用（此类应用可能存在 ADC 溢出的风险）的性能。

如果 ADC_CFGR1 寄存器中的 WAIT 位置 1，则仅当之前的数据已进行处理、ADC_DR 寄存器已读取或者 EOC 位已清零后，才会开始新的转换。

通过这种方式，可自动调整 ADC 的速度，使其适应系统读取数据的速度。

注： *转换进行时发生的或在读访问之前的等待时间内发生的硬件触发会被忽略。*

图 46. 等待模式转换（连续模式，软件触发）



1. EXTEN=00，CONT=1
2. CHSEL=0x3，SCANDIR=0，WAIT=1，AUTOFF=0

有关代码示例，请参见 *A.8.11：等待模式序列代码示例*。

14.7.2 自动关闭模式 (AUTOFF)

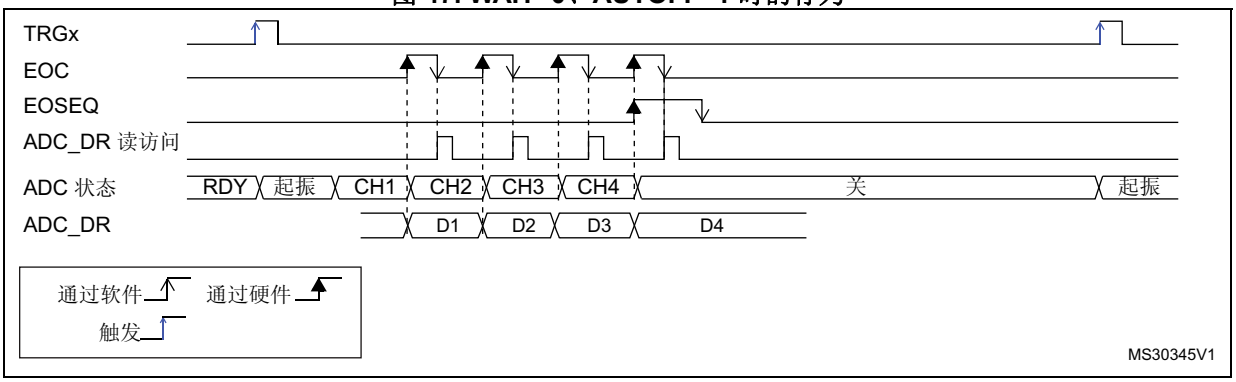
ADC 具有自动电源管理功能，也称为自动关闭模式，将 ADC_CFGR1 寄存器中的 AUTOFF 位置 1 可使能此模式。

如果 AUTOFF=1，ADC 始终会在未进行转换时关闭，并会在转换开始后自动唤醒（通过软件或硬件触发）。在启动转换的触发事件和 ADC 的采样时间之间，会自动插入启动时间。随后，转换序列完成后，ADC 会自动禁止。

如果应用需要进行的转换次数相对较少，或者为了证明开关 ADC 额外使用的功率和时间合理而将转换请求的间隔时间设定得足够长（例如采用低频硬件触发），使用自动关闭模式可显著降低应用的功耗。

对于采用低频时钟的应用，可将自动关闭模式与等待模式转换 (WAIT=1) 结合使用，如果 ADC 在等待过程中自动掉电、并会在应用读取 ADC_DR 寄存器后立即重启，这种组合可显著降低功耗（请参见*图 48：WAIT=1、AUTOFF=1 时的行为*）。

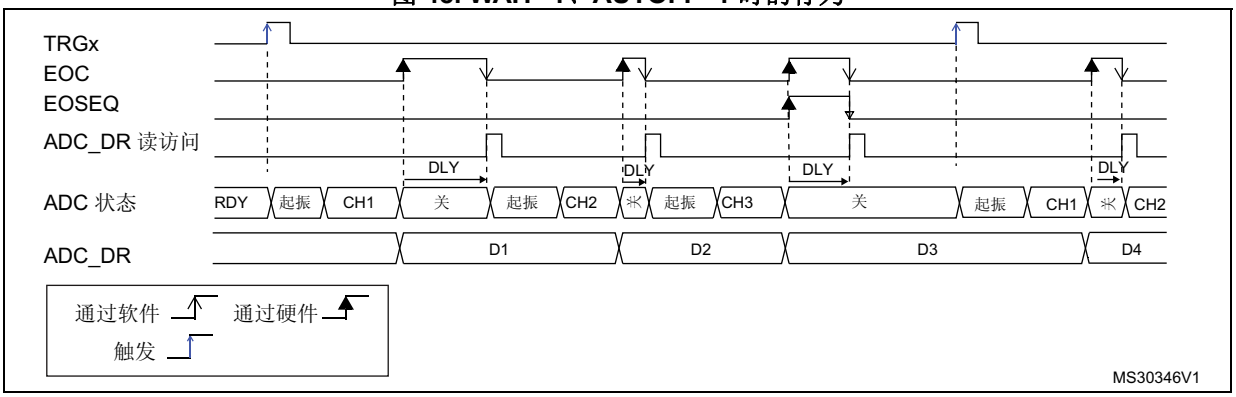
图 47. WAIT=0、AUTOFF=1 时的行为



1. EXTSEL=TRGx，EXTEN=01（上升沿），CONT=x，ADSTART=1，CHSEL=0xF，SCANDIR=0，WAIT=1，AUTOFF=1

有关代码示例，请参见 *A.8.12：自动关闭和无等待模式序列代码示例*。

图 48. WAIT=1、AUTOFF=1 时的行为



1. EXTSEL=TRGx，EXTEN=01（上升沿），CONT=x，ADSTART=1，CHSEL=0xF，SCANDIR=0，WAIT=1，AUTOFF=1

有关代码示例，请参见 *A.8.13：自动关闭和等待模式序列代码示例*。

14.8 模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD）

将 ADC_CFGR1 寄存器中的 AWDEN 位置 1，可使能 AWD 模拟看门狗功能。此功能用于监控一条选定的通道或所有已使能的通道（请参见*表 61：模拟看门狗通道选择*）是否仍处于所配置的电压范围（窗口）内，如*图 49* 所示。

如果 ADC 转换的模拟电压低于阈值下限或高于阈值上限，则会将 AWD 模拟看门狗状态位置 1。这些阈值在 ADC_HTR 和 ADC_LTR 16 位寄存器的 12 个最低有效位中进行编程。可以通过将 ADC_IER 寄存器中的 AWDIE 位置 1 来使能中断。

AWD 标志可通过由软件向其写入 1 的方式来清零。

如果转换的数据分辨率小于 12 位（取决于 DRES[1:0] 位），由于始终会在内部对完整的 12 位原始转换数据进行比较（左对齐），因此编程阈值的 LSB 必须保持清零状态。

有关代码示例，请参见 *A.8.14：模拟看门狗代码示例*。

表 60 介绍了如何对所有可能的分辨率进行比较。

表 60. 模拟看门狗比较

分辨率位 RES[1:0]	模拟看门狗比较对象：		注释
	原始转换数据，左对齐(1)	阈值	
00：12 位	DATA[11:0]	LT[11:0] 和 HT[11:0]	-
01：10 位	DATA[11:2]，00	LT[11:0] 和 HT[11:0]	用户必须将 LT1[1:0] 和 HT1[1:0] 配置为“00”
10：8 位	DATA[11:4]，0000	LT[11:0] 和 HT[11:0]	用户必须将 LT1[3:0] 和 HT1[3:0] 配置为“0000”
11：6 位	DATA[11:6]，000000	LT[11:0] 和 HT[11:0]	用户必须将 LT1[5:0] 和 HT1[5:0] 配置为“000000”

1. 进行任何对齐计算之前，会对原始转换数据进行看门狗比较。

表 61 介绍了如何配置 ADC_CFGR1 寄存器中的 AWDSGL 位和 AWDEN 位，以使能一条或多条通道上的模拟看门狗。

图 49. 模拟看门狗的保护区域

表 61. 模拟看门狗通道选择

模拟看门狗保护的通道	AWDSGL 位	AWDEN 位
无	x	0
所有通道	0	1
单(1)通道	1	1

1. 通过 AWDCH[4:0] 位选择

14.9 过采样器

过采样单元会进行数据预处理，以减轻 CPU 的负担。过采样单元可处理多个转换，并计算多个转换结果的平均值，得到数据宽度增大（高达 16 位）的单个数据。

它提供的结果采用以下形式，其中的 N 和 M 可以进行调整：

$$\text{结果} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{转换}(t_n)$$

允许通过硬件执行以下功能：计算平均值、降低数据速率、改进 SNR 以及基本滤波。

过采样率 N 通过 ADC_CFGR2 寄存器中的 OVFS[2:0] 位进行定义，它的范围是 2x 到 256x。分频系数 M 由向右移位构成，最多可移 8 位。它可通过 ADC_CFGR2 寄存器中的 OVSS[3:0] 位进行配置。

有关代码示例，请参见 *A.8.15：过采样代码示例*。

求和单元可得出多达 20 位（256 x 12 位）的结果，结果会先右移。随后，会将结果的高位截断，只留下 16 个最低有效位，这些位使用移位后剩下的最低有效位四舍五入为最接近的数值，然后将最终得到的结果传输到 ADC_DR 数据寄存器中。

注： *如果移位后得到的中间结果超过 16 位，则会截断结果的高位。*

图 50. 20 位到 16 位结果的截断过程

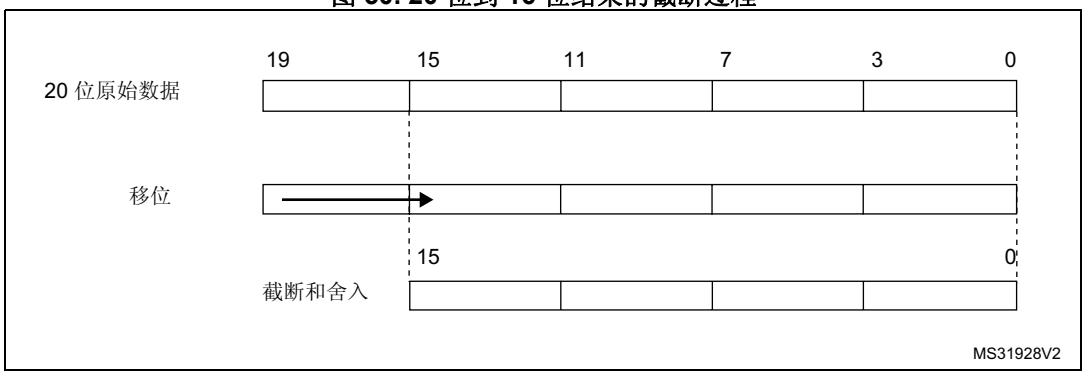
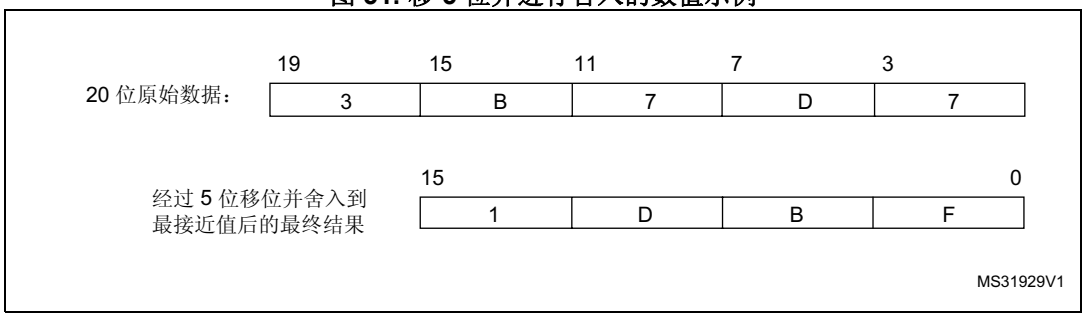


图 51 介绍了从原始的 20 位累加数据到最终 16 位结果的数值处理过程。

图 51. 移 5 位并进行舍入的数值示例



以下*表 62* 列出了原始转换数据等于 0xFFF 时对应的各种 N 和 M 组合的数据格式。

表 62. 最大输出结果与 N 和 M 的对应关系。呈灰色显示的数值表示截断的部分

过采样率	最大原始数据	不移位 OVSS = 0000	移 1 位 OVSS = 0001	移 2 位 OVSS = 0010	移 3 位 OVSS = 0011	移 4 位 OVSS = 0100	移 5 位 OVSS = 0101	移 6 位 OVSS = 0110	移 7 位 OVSS = 0111	移 8 位 OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

与标准转换模式相比，过采样模式下的转换时序不会发生变化：整个过采样序列中，采样时间保持相等。每完成 N 次转换都会提供新数据，等效延迟等于 N x tADC = N x (t_{SMPL} + t_{SAR})。各标志的情况如下：

- 每个采样阶段后都会将采样阶段结束标志 (EOSMP) 置 1
- 如果过采样结果可用，每完成 N 次转换都会发生转换结束事件 (EOC)
- 过采样数据序列完成后（即 N x 序列长度次转换之后），会发生序列结束事件 (EOCSEQ)

14.9.1 过采样时支持的 ADC 工作模式

在过采样模式下，大部分 ADC 工作模式都可用：

- 单次或连续模式转换、向前或向后扫描序列
- 可由软件或触发器启动 ADC 转换
- ADC 在转换过程中停止（中止）
- 通过 CPU 或 DMA 在支持溢出检测的情况下读取数据
- 低功耗模式（WAIT、AUTOFF）
- 可编程分辨率：在这种情况下，会按照与 12 位转换相同的方式对分辨率降低的转换值（根据 ADC_CFGR1 寄存器中的 RES[1:0] 位）进行累加、截断、四舍五入和移位。

注：处理过采样数据时，不可使用对齐模式。ADC_CFGR1 中的 ALIGN 位会被忽略，且数据始终采用右对齐格式。

14.9.2 模拟看门狗

可以使用模拟看门狗功能（AWDSGL 位和 AWDEN 位），但存在以下区别：

- 会忽略 RES[1:0] 位，始终会使用完整的 12 位值 HT[11:0] 和 LT[11:0] 进行比较。
- 会比较 16 位过采样结果 ADC_DR[15:4] 的 12 个最高有效位

注：移位位数较大时必须多加留意，因为这样会缩小比较范围。例如，如果过采样结果移了 4 位，得到 12 位右对齐数据，那么只能对 8 个数据位执行有效的模拟看门狗比较。比较操作会在 ADC_DR[11:4] 与 HT[0:7]/LT[0:7] 之间进行，并且 HT[11:8]/LT[11:8] 必须保持复位状态。

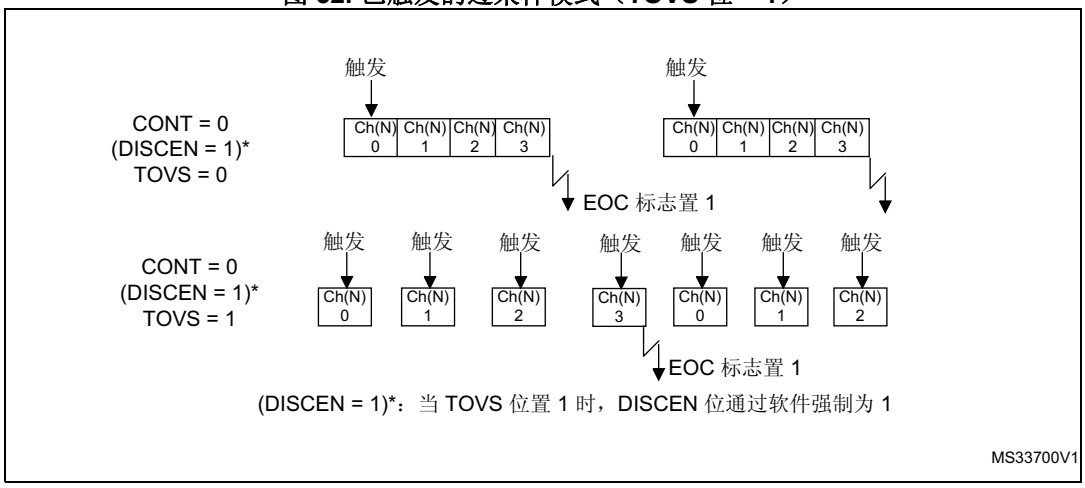
14.9.3 触发模式

平均值计算单元还可用于基本滤波，虽然它不是效率非常高的滤波器（衰减缓慢、停止频段衰减受限），但可用作陷波滤波器，用于抑制恒定的寄生频率（通常来自电源或切换模式电源）。为此，可使用 ADC_CFGR2 中的 TOVS 位使能特定的不连续模式，以获得由用户定义、且与转换时间本身无关的过采样频率。

图 52 显示了在不连续模式下如何响应触发从而开始转换。

如果 TOVS 位置 1，则会忽略 DISCEN 位，并将该位视为 1。

图 52. 已触发的过采样模式（TOVS 位 = 1）



14.10 温度传感器和内部参考电压

温度传感器可用于测量器件的结温 (T_J)。温度传感器在内部连接到 ADC_IN18 输入通道，该通道用于将传感器输出电压转换为数字值。温度传感器模拟引脚的采样时间必须大于数据手册中指定的最小 T_{S_temp} 值。不使用时可将传感器置于掉电模式。

内部参考电压 (V_{REFINT}) 为 ADC 和比较器提供了一个稳定的（带隙基准）电压输出。V_{REFINT} 内部连接到 ADC_IN17 输入通道。V_{REFINT} 的精确电压由 ST 在生产测试期间对每部分单独测量，并存储于系统存储区。访问模式为只读。

图 53 显示的是温度传感器、内部参考电压与 ADC 之间连接的方框图。

必须将 TSEN 位置 1 才能使能 ADC_IN18（温度传感器）的转换，必须将 VREFEN 位置 1 才能使能 ADC_IN17 (V_{REFINT}) 的转换。

温度传感器的输出电压随温度线性变化。由于工艺不同，该线的偏移量取决于各个芯片（芯片之间的温度变化可达 45 °C）。

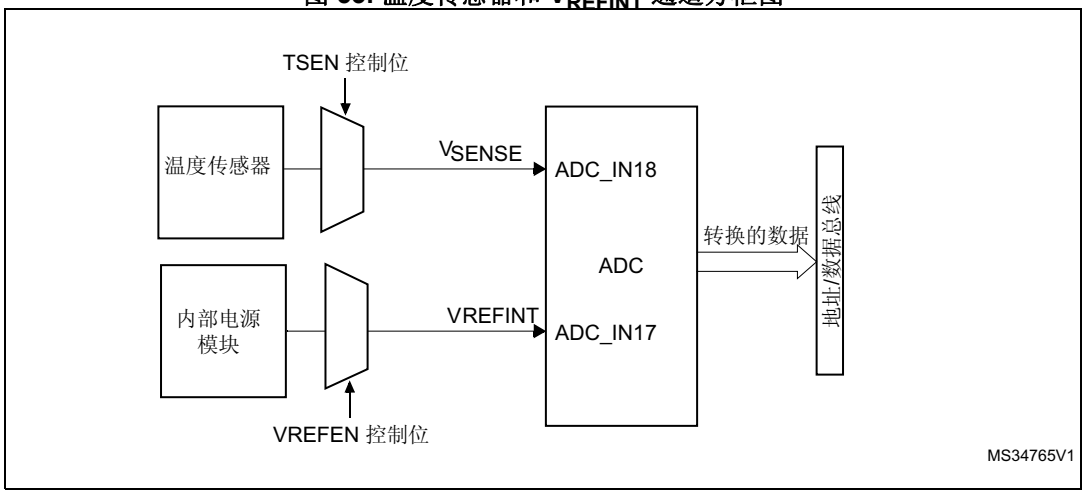
未校准的内部温度传感器更适用于对温度变量而非绝对温度进行测量的应用。为提高温度传感器测量的准确性，ST 在生产过程中将校准值存储在每个器件的系统存储器中。

在制造过程中，会将温度传感器的校准数据和内部参考电压存储在系统存储区。随后，用户应用可读取这些数据，并使用这些数据提高温度传感器或内部参考的准确性。其他相关信息，请参见数据手册。

主要特性

- 支持的温度范围：–40 °C 到 125 °C
- 线性度：最高 ±2 °C，精度取决于校准情况

图 53. 温度传感器和 V_{REFINT} 通道方框图



读取温度

1. 选择 ADC_IN18 输入通道
2. 选择器件数据手册中规定的合适的采样时间 (T_{S_temp})。
3. 将 ADC_CCR 寄存器中的 TSEN 位置 1，将温度传感器从掉电模式中唤醒，并等待其稳定时间 (t_{START})
   有关代码示例，请参见 *A.8.16：温度配置代码示例*。
4. 将 ADC_CR 寄存器中的 ADSTART 位置 1（或通过外部触发），开始 ADC 转换
5. 读取 ADC_DR 寄存器中生成的VSENSE 数据
6. 使用以下公式计算温度：

$$\text{Temperature (in °C)} = \frac{130\ °C - 30\ °C}{TS_CAL2 - TS_CAL1} \times (TS_DATA - TS_CAL1) + 30\ °C$$

$$\text{Temperature (in °C)} = \frac{V_{30} - V_{SENSE}}{Avg_Slope} + 30\ °C$$

其中：

- TS_CAL2 是在 130°C 下获得的温度传感器校准值
- TS_CAL1 是在 30°C 下获得的温度传感器校准值
- TS_DATA 是由 ADC 转换得到的实际温度传感器输出值
  更多关于 TS_CAL1 和 TS_CAL2 校准点的信息，请参见相应的器件数据手册。

有关代码示例，请参见 *A.8.17：温度计算代码示例*。

注： *传感器从掉电模式中唤醒需要一个启动时间，启动时间过后其才能输出正确的 V_{SENSE}。ADC 在上电后同样需要一个启动时间，因此，为尽可能缩短延迟时间，应同时将 ADEN 和 TSEN 位置 1。*

使用内部参考电压计算实际的 VDDA 电压

施加给微控制器的 V_{DDA} 电源电压可能会有变化，或无法获得准确值。在制造过程中由 ADC 在 V_{DDA} = 3 V 的条件下获得的内置内部参考电压 (VREFINT) 及其校准数据可用于评估实际的 VDDA 电压水平。

以下公式可求得为器件供电的实际的 V_{DDA} 电压：

$$V_{DDA} = 3\ V \times VREFINT_CAL / VREFINT_DATA$$

其中：

- VREFINT_CAL 是 VREFINT 校准值
- VREFINT_DATA 是由 ADC 转换得到的实际 VREFINT 输出值

将电源相关的 ADC 测量值转换为绝对电压值

ADC 用于提供对应于模拟电源与施加给转换通道的电压之比的数字值。对于大部分应用用例，需要将该比值转换成与 V_{DDA} 无关的电压。对于 V_{DDA} 已知、ADC 转换值进行了右对齐的应用，可使用以下公式得到该绝对值：

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL_SCALE} \times ADC_DATA_x$$

对于 V_{DDA} 值未知的应用，必须使用内部参考电压，V_{DDA} 可替换为*使用内部参考电压计算实际的 VDDA 电压*部分提供的表达式，从而得出以下公式：

$$V_{CHANNELx} = \frac{3\ V \times VREFINT_CAL \times ADC_DATA_x}{VREFINT_DATA \times FULL_SCALE}$$

其中：

- VREFINT_CAL 是 VREFINT 校准值
- ADC_DATA_x 是由 ADC 在通道 x 上测得的值（右对齐）
- VREFINT_DATA 是由 ADC 转换得到的实际 VREFINT 输出值
- full_SCALE 是 ADC 输出的最大数字值。例如，如果分辨率为 12 位，该值为 2^{12} - 1 = 4095，如果分辨率为 8 位，该值为 2^8 - 1 = 255。

注：如果执行 ADC 测量时使用的是输出格式而非 12 位右对齐格式，那么必须先将所有参数转换为兼容格式，然后再进行计算。

14.11 VLCD 电压监控

ADC_CCR 寄存器中的 VLCDEN 位用于测量 V_{LCD} 引脚上的 LCD 电源电压。由于 V_{LCD} 电压可能高于 V_{DDA}，为确保 ADC 正常工作，V_{LCD} 引脚需要内部连接到桥接分配器。VLCDEN 位置 1 时，会自动使能此桥，以将 LCD_VLCD1 连接到 ADC1_IN16 输入通道。因此，转换得出的数字值是 V_{LCD} 电压的三分之一（LCD 配置为 1/3 偏置时）或 V_{LCD} 电压的四分之一（LCD 配置为 1/4 偏置或 1/2 偏置时）。为防止电池出现意外的电能消耗，建议仅在必要时使能桥接分配器，即执行 ADC 转换。

14.12 ADC 中断

发生下列任一事件均可生成中断：

- 校准结束（EOCAL 标志）
- ADC 就绪后，ADC 上电（ADRDY 标志）
- 任何转换结束（EOC 标志）
- 转换序列结束（EOSEQ 标志）
- 进行模拟看门狗检测时（AWD 标志）
- 采样阶段结束时（EOSMP 标志）
- 发生数据溢出时（OVR 标志）

可以使用单独的中断使能位以提高灵活性。

表 63. ADC 中断

中断事件	事件标志	使能控制位
校准结束	EOCAL	EOCALIE
ADC 就绪	ADRDY	ADRDYIE
转换结束	EOC	EOCIE
转换序列结束	EOSEQ	EOSEQIE
模拟看门狗状态位置 1	AWD	AWDIE
采样阶段结束	EOSMP	EOSMPIE
上溢	OVR	OVRIE

14.13 ADC 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

14.13.1 ADC 中断和状态寄存器 (ADC_ISR)

ADC interrupt and status register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EOCAL	Res.	Res.	Res.	AWD	Res.	Res.	OVR	EOSEQ	EOC	EOSMP	ADRDY
				r_w1				r_w1			r_w1	r_w1	rc_w1	r_w1	r_w1

位 31:12 保留，必须保持复位值。

位 11 **EOCAL**：校准结束标志 (End Of Calibration flag)

校准完成时，该位由硬件置 1。通过软件写入 1 可将该位清零。

0：校准未完成

1：校准已完成

位 10:8 保留，必须保持复位值。

位 7 **AWD**：模拟看门狗标志 (Analog watchdog flag)

当转换电压超过在 ADC_LTR 和 ADC_HTR 寄存器中编程的值时，硬件会将该位置 1。通过软件写入 1 可将该位清零。

0：未发生模拟看门狗事件（或标志事件已通过软件确认并清零）。

1：发生模拟看门狗事件

位 6:5 保留，必须保持复位值。

位 4 **OVR**：ADC 溢出 (ADC overrun)

该位在发生溢出事件时由硬件置 1，这意味着在 EOC 标志已置 1 时，新转换已完成。通过软件写入 1 可将该位清零。

0：未发生溢出事件（或标志事件已通过软件确认并清零）。

1：发生溢出

位 3 **EOSEQ**：序列结束标志 (End of sequence flag)

在由 CHSEL 位选择的一系列通道转换结束时，会通过硬件将该位置 1。通过软件写入 1 可将该位清零。

0：转换序列未完成（或标志事件已通过软件确认并清零）。

1：转换序列已完成

位 2 **EOC**：转换结束标志 (End of conversion flag)

当通道的每次转换结束，新数据结果出现在 ADC_DR 寄存器时，会通过硬件将该位置 1。通过软件向该位写入 1，或读取 ADC_DR 寄存器都可将该位清零。

0：通道转换未完成（或标志事件已通过软件确认并清零）。

1：通道转换已完成

位 1 **EOSMP**：采样结束标志 (End of sampling flag)

在转换过程中，当采样阶段结束时该位由硬件置 1。通过软件将该位编程为“1”，可将该位清零。

0：采样阶段未结束（或标志事件已通过软件确认并清零）。

1：采样阶段已结束

位 0 **ADRDY**：ADC 就绪（ADC ready）

ADC 使能后（位 ADEN=1）以及 ADC 达到准备好接收转换请求的状态时，会通过硬件将该位置 1。通过软件写入 1 可将该位清零。

0：ADC 未准备好开始转换（或标志事件已通过软件确认并清零）。

1：ADC 已准备好开始转换

14.13.2 ADC 中断使能寄存器 (ADC_IER)

ADC interrupt enable register

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EOCALIE	Res.	Res.	Res.	AWDIE	Res.	Res.	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
				rw				rw			rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值。

位 11 **EOCALIE**：校准结束中断使能 (End of calibration interrupt enable)

此位由软件置 1 和清零，用于使能/禁止校准结束中断。

0：禁止校准结束中断

1：使能校准结束中断

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 10:8 保留，必须保持复位值。

位 7 **AWDIE**：模拟看门狗中断使能 (Analog watchdog interrupt enable)

此位由软件置 1 和清零，用于使能/禁止模拟看门狗中断。

0：禁止模拟看门狗中断

1：使能模拟看门狗中断

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 6:5 保留，必须保持复位值。

位 4 **OVRIE**：溢出中断使能 (Overrun interrupt enable)

此位由软件置 1 和清零，用于使能/禁止溢出中断。

0：禁止溢出中断

1：使能溢出中断 OVR 位置 1 时产生中断。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 3 **EOSEQIE**：转换序列结束中断使能 (End of conversion sequence interrupt enable)

此位由软件置 1 和清零，用于使能/禁止转换序列结束中断。

0：禁止 EOSEQ 中断

1：使能 EOSEQ 中断 EOSEQ 位置 1 时产生中断。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 2 **EOCIE**：转换结束中断使能 (End of conversion interrupt enable)

此位由软件置 1 和清零，用于使能/禁止转换结束中断。

0：禁止 EOC 中断

1：使能 EOC 中断 EOC 位置 1 时产生中断。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 1 **EOSMPIE**：采样结束中断使能 (End of sampling interrupt enable)

此位由软件置 1 和清零，用于使能/禁止采样阶段结束中断。

0：禁止 EOSMP 中断。

1：使能 EOSMP 中断。EOSMP 位置 1 时产生中断。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 0 **ADRDYIE**：ADC 就绪中断使能 (ADC ready interrupt enable)

此位由软件置 1 和清零，用于使能/禁止 ADC 就绪中断。

0：禁止 ADRDY 中断。

1：使能 ADRDY 中断。ADRDY 位置 1 时产生中断。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

14.13.3 ADC 控制寄存器 (ADC_CR)

ADC control register

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
ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

位 31 **ADCAL**：ADC 校准 (ADC calibration)

该位由软件置 1，用于开始 ADC 校准。

校准完成后，该位由硬件清零。

0：校准已完成

1：写入 1 可校准 ADC。读取值为 1 表示正在进行校准。

注：仅当 ADC 已禁止时（ADCAL=0、ADSTART=0、ADSTP=0、ADDIS=0 且 ADEN=0），才允许通过软件将 ADCAL 位置 1。

注：仅当 ADEN=1 且 ADSTART=0（ADC 已使能，当前未进行任何转换）时，才允许通过软件对 ADC_CALFACT 执行写操作来更新校准系数。

位 30:29 保留，必须保持复位值。

位 28 **ADVREGEN**：ADC 调压器使能 (ADC Voltage Regulator Enable)

该位通过以下方式置 1：

- 通过软件置 1，从而使能 ADC 内部调压器。
- 启动校准（将 ADCAL 置 1）时，或者使能 ADC 时（将 ADEN 置 1）通过硬件置 1

通过软件将该位清零可禁止调压器（仅当 ADEN=0 时才能将其清零）。

0：禁止 ADC 调压器

1：使能 ADC 调压器

注：仅当 ADC 已禁止时（ADCAL=0、ADSTART=0、ADSTP=0、ADDIS=0 且 ADEN=0），才能通过软件对该位域进行编程

位 27:5 保留，必须保持复位值。

位 4 **ADSTP**：ADC 停止转换命令 (ADC stop conversion command)

该位由软件置 1，用于停止和丢弃正在进行的转换（ADSTP 命令）。

当转换已有效丢弃、并且 ADC 已准备好接收新的开始转换命令时，会通过硬件将该位清零。

0：当前未执行 ADC 停止转换命令

1：写入 1 可停止 ADC。读取值为 1 表示正在执行 ADSTP 命令。

注：仅当 ADSTART=1 且 ADDIS=0 时（ADC 已使能、可能正在进行转换、并且没有任何待处理的禁止 ADC 的请求），才允许通过软件将 ADSTP 置 1。

位 3 保留，必须保持复位值。

位 2 **ADSTART**：ADC 开始转换命令 (ADC start conversion command)

此位由软件置 1，用于开始 ADC 转换。根据 EXTEN [1:0] 配置位的值，可以立即开始转换（软件触发配置），也可以在发生硬件触发事件后开始转换（硬件触发配置）。

该位通过硬件清零：

– 在单次转换模式下（CONT=0、DISCEN=0），如果选择了软件触发 (EXTEN=00)：出现转换序列结束 (EOSEQ) 标志时清零。

– 在不连续转换模式下（CONT=0、DISCEN=1），如果选择了软件触发 (EXTEN=00)：出现转换结束 (EOC) 标志时清零。

– 在所有其他情况下：执行完 ADSTP 命令后，由硬件将 ADSTP 位清零的同时清零。

0：当前未进行 ADC 转换。

1：写入 1 可开始 ADC。读取值为 1 表示 ADC 正在工作，可能正在进行转换。

注：仅当 ADEN=1 且 ADDIS=0 时（ADC 已使能，并且没有任何待处理的禁止 ADC 的请求），才允许通过软件将 ADSTART 置 1。

位 1 **ADDIS**：ADC 禁止命令 (ADC disable command)

该位通过软件置 1，用于禁止 ADC（ADDIS 命令）并使其进入掉电状态（OFF 状态）。

ADC 已有效禁止后，会立即通过硬件将该位清零（此时也会通过硬件将 ADEN 清零）。

0：当前未执行 ADDIS 命令

1：写入 1 可禁止 ADC。读取值为 1 表示正在执行 ADDIS 命令。

注：仅当 ADEN=1 且 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件将 ADDIS 置 1。

位 0 **ADEN**：ADC 使能命令 (ADC enable command)

该位通过软件置 1，用于使能 ADC。ADRDY 标志置 1 后，ADC 将立即准备好运行。

如果 ADC 已禁止，则执行 ADDIS 命令后，将通过硬件对该位清零。

0：禁止 ADC（OFF 状态）

1：写入 1 来使能 ADC。

注：仅当 ADC_CR 寄存器的所有位均为 0 时（ADCAL=0、ADSTP=0、ADSTART=0、ADDIS=0 且 ADEN=0），才允许通过软件将 ADEN 位置 1。

14.13.4 ADC 配置寄存器 1 (ADC_CFGR1)

ADC configuration register 1

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWDCH[4:0]					Res.	Res.	AWDEN	AWDSGL	Res.	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
rw	rw	rw	rw	rw			rw			rw	rw		rw	rw	rw

位 31 保留，必须保持复位值。

位 30:26 **AWDCH[4:0]**：模拟看门狗通道选择 (Analog watchdog channel selection)

这些位由软件置 1 和清零。它们用于选择由模拟看门狗监控的输入通道。

00000：通过 AWD 监控 ADC 模拟输入通道 0

00001：通过 AWD 监控 ADC 模拟输入通道 1

.....

10010：通过 AWD 监控 ADC 模拟输入通道 18

其他值：保留，不得使用

注： 由 AWDCH[4:0] 位选择的通道也必须设置到 CHSELR 寄存器中。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

位 25:24 保留，必须保持复位值。

位 23 **AWDEN**：模拟看门狗使能 (Analog watchdog enable)

此位由软件置 1 和清零。

0：禁止模拟看门狗

1：使能模拟看门狗

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 22 **AWDSGL**：在单一通道或所有通道上使能看门狗 (Enable the watchdog on a single channel or on all channels)

此位由软件置 1 和清零，用于使能由 AWDCH[4:0] 位确定的通道或所有通道上的模拟看门狗。

0：在所有通道上使能模拟看门狗

1：在单一通道上使能模拟看门狗

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 21:17 保留，必须保持复位值。

位 16 **DISCEN**：不连续模式 (Discontinuous mode)

此位由软件置 1 和清零，用于使能/禁止不连续模式。

0：禁止不连续模式

1：使能不连续模式

注：　不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 15 **AUTOFF**：自动关闭模式 (Auto-off mode)

此位由软件置 1 和清零，用于使能/禁止自动关闭模式。

0：禁止自动关闭模式

1：使能自动关闭模式

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 14 **WAIT**：等待转换模式 (Wait conversion mode)

此位由软件置 1 和清零，用于使能/禁止等待转换模式。

0：等待转换模式关闭

1：等待转换模式开启

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 13 **CONT**：单次/连续转换模式 (Single/continuous conversion mode)

此位由软件置 1 和清零。该位置 1 时，转换将持续进行，直到该位清零。

0：单次转换模式

1：连续转换模式

注：　不能同时使能不连续模式和连续模式：禁止同时将 DISCEN 和 CONT 位置 1。

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 12 **OVRMOD**：溢出管理模式 (Overrun management mode)

该位通过软件进行置 1 和清零，并用于配置数据溢出的管理方式。

0：如果检测到溢出，ADC_DR 寄存器会保留原有数据。

1：如果检测到溢出，ADC_DR 寄存器会被上一转换结果覆盖。

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 11:10 **EXTEN[1:0]**：外部触发使能和极性选择 (External trigger enable and polarity selection)

这些位由软件置 1 和清零，用于选择外部触发极性并使能触发。

00：禁止硬件触发检测（可通过软件开始转换）

01：在上升沿执行硬件触发检测

10：在下降沿执行硬件触发检测

11：在上升沿和下降沿都执行硬件触发检测

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

位 9 保留，必须保持复位值。

位 8:6 **EXTSEL[2:0]**：外部触发选择 (External trigger selection)

这些位可选择用于触发转换开始的外部事件（有关详情，请参见*表 58：外部触发器*）：

000：TRG0
001：TRG1
010：TRG2
011：TRG3
100：TRG4
101：TRG5
110：TRG6
111：TRG7

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

位 5 **ALIGN**：数据对齐 (Data alignment)

此位由软件置 1 和清零，用于选择右对齐或左对齐。请参见*第 293 页上的图 44：数据对齐方式和分辨率（过采样已禁止：OVSE = 0）*。

0：右对齐
1：左对齐

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 4:3 **RES[1:0]**：数据分辨率 (Data resolution)

通过软件写入这些位可选择转换的分辨率。

00：12 位
01：10 位
10：8 位
11：6 位

注：　仅当 ADEN=0 时，才允许通过软件对这些位执行写操作。

位 2 **SCANDIR**：扫描序列方向 (Scan sequence direction)

此位由软件置 1 和清零，用于选择扫描序列中各通道的方向。

0：向前扫描（CHSEL0 到 CHSEL18）
1：向后扫描（CHSEL18 到 CHSEL0）

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 1 **DMACFG**：直接存储器访问配置 (Direct memory access configuration)

此位由软件置 1 和清零，用于在 DMA 两种工作模式之间进行选择，仅当 DMAEN=1 时，该位才有效。

0：选择 DMA 单次模式
1：选择 DMA 循环模式

更多详细信息，请参见*第 294 页上的第 14.6.5 节：使用 DMA 管理转换的数据*

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 0 **DMAEN**：直接存储器访问使能 (Direct memory access enable)

此位由软件置 1 和清零，用于使能 DMA 请求的生成。这样便可使用 DMA 控制器自动管理转换的数据。有关详细信息，请参见*第 294 页上的第 14.6.5 节：使用 DMA 管理转换的数据*。

0：禁止 DMA
1：使能 DMA

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

14.13.5 ADC 配置寄存器 2 (ADC_CFGR2)

ADC configuration register 2

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
CKMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw

位 31:30 **CKMODE[1:0]**：ADC 时钟模式 (ADC clock mode)

此位由软件置 1 和清零，用于定义为模拟 ADC 提供时钟的方式：

00：ADCCLK（异步时钟模式），在产品级生成（请参见 RCC 部分）

01：PCLK/2（同步时钟模式）

10：PCLK/4（同步时钟模式）

11：PCLK（同步时钟模式）。仅当 PCLK 的时钟占空比为 50% 时，才必须使能此配置（必须绕过 RCC 中配置的 APB 预分频系数，并且系统时钟占空比必须在 50% 以内）。

在所有异步时钟模式下，从定时器触发到转换开始的延迟过程中，不存在抖动。

注：　仅当 ADC 已禁止时（ADCAL=0、ADSTART=0、ADSTP=0、ADDIS=0 且 ADEN=0），才允许通过软件对这些位执行写操作。

位 29:10　保留，必须保持复位值。

位 9 **TOVS**：已触发过采样 (Triggered Oversampling)

此位由软件置 1 和清零。

0：会在触发后连续完成某一通道的所有过采样转换

1：某一通道的每个过采样转换都需要触发

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 8:5 **OVSS[3:0]**：过采样移位 (Oversampling shift)

此位由软件置 1 和清零。

0000：不发生移位

0001：移 1 位

0010：移 2 位

0011：移 3 位

0100：移 4 位

0101：移 5 位

0110：移 6 位

0111：移 7 位

1000：移 8 位

其他代码保留

注：　仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 4:2 **OVSR[2:0]**：过采样率 (Oversampling ratio)

该位域定义过采样率的数值。

000：2x
001：4x
010：8x
011：16x
100：32x
101：64x
110：128x
111：256x

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 1 保留，必须保持复位值。

位 0 **OVSE**：过采样器使能 (Oversampler Enable)

此位由软件置 1 和清零。

0：禁止过采样器
1：使能过采样器

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

14.13.6 ADC 采样时间寄存器 (ADC_SMPR)

ADC sampling time register

偏移地址：0x14

复位值：0x0000 0000

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

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

位 31:3 保留，必须保持复位值。

位 2:0 **SMP[2:0]**：采样时间选择 (Sampling time selection)

这些位由软件写入，用于选择应用于所有通道的采样时间。

000：1.5 个 ADC 时钟周期
001：3.5 个 ADC 时钟周期
010：7.5 个 ADC 时钟周期
011：12.5 个 ADC 时钟周期
100：19.5 个 ADC 时钟周期
101：39.5 个 ADC 时钟周期
110：79.5 个 ADC 时钟周期
111：160.5 个 ADC 时钟周期

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

14.13.7 ADC 看门狗阈值寄存器 (ADC_TR)

ADC watchdog threshold register

偏移地址：0x20

复位值：0x0FFF 0000

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

位 31:28 保留，必须保持复位值。

位 27:16 **HT[11:0]**：*模拟看门狗阈值上限 (Analog watchdog higher threshold)*

这些位由软件写入，用于定义模拟看门狗的阈值上限。请参见*第 297 页上的第 14.8 节：模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD）*。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

位 15:12 保留，必须保持复位值。

位 11:0 **LT[11:0]**：模拟看门狗阈值下限 (Analog watchdog lower threshold)

这些位由软件写入，用于定义模拟看门狗的阈值下限。

请参见*第 297 页上的第 14.8 节：模拟窗口看门狗（AWDEN、AWDSGL、AWDCH、AWD_HTR/LTR、AWD）*。

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

14.13.8 ADC 通道选择寄存器 (ADC_CHSELR)

ADC channel selection register

偏移地址：0x28

复位值：0x0000 0000

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

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

位 31:19 保留，必须保持复位值。

位 18:0 **CHSELx**：通道 x 选择 (Channel-x selection)

这些位由软件写入，用于定义将哪些通道纳入要转换的通道序列。

0：未选择输入通道 x 进行转换

1：已选择输入通道 x 进行转换

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对这些位执行写操作。

14.13.9 ADC 数据寄存器 (ADC_DR)

ADC data register

偏移地址：0x40

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **DATA[15:0]**：转换后的数据 (Converted data)

这些位为只读。其中包含上一转换通道的转换结果。数据可以采用左对齐和右对齐，如*第 293 页上的图 44：数据对齐方式和分辨率（过采样已禁止：OVSE = 0）*所示。

校准完成后，DATA[6:0] 会立即包含校准系数。

14.13.10 ADC 校准系数 (ADC_CALFACT)

ADC Calibration factor

偏移地址：0xB4

复位值：0x0000 0000

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

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

位 31:7 保留，必须保持复位值。

位 6:0 **CALFACT[6:0]**：校准系数 (Calibration Factor)

这些位可由硬件或软件写入。

– 单端输入校准完成后，会立即由硬件更新为校准系数。

– 软件可向这些位写入新的校准系数。如果新校准系数不同于当前存储于模拟 ADC 中的校准系数，启动新的单端校准后，会立即应用新校准系数。

– 校准完成后，DATA[6:0] 会立即包含校准系数。

注： 仅当 ADEN=1 且 ADSTART=0 时（ADC 已使能、当前未执行任何校准和转换），才允许通过软件对这些位执行写操作。

14.13.11 ADC 通用配置寄存器 (ADC_CCR)

ADC common configuration register

偏移地址：0x308

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LFMEN	VLCD EN	TS EN	VREF EN	PRESC[3:0]				Res.	Res.
						rw	rw	rw	rw	rw	rw	rw	rw		

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

位 31:26 保留，必须保持复位值。

位 25 **LFMEN**：低频模式使能 (Low Frequency Mode enable)

此位由软件置 1 和清零，用于使能/禁止低频模式。

如果用户选择的 ADC 时钟频率低于 3.5 MHz，则必须使能该模式

0：禁止低频模式

1：使能低频模式

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 24 **VLCDEN**：VLCD 使能 (V_{LCD} enable)

此位由软件置 1 和清零，用于使能/禁止 V_{LCD} 读取电路。

0：禁止 V_{LCD} 读取电路
1：使能 V_{LCD} 读取电路

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 23 **TSEN**：温度传感器使能 (Temperature sensor enable)

此位由软件置 1 和清零，用于使能/禁止温度传感器。

0：禁止温度传感器
1：使能温度传感器

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 22 **VREFEN**：VREFINT 使能 (V_{REFINT} enable)

此位由软件置 1 和清零，用于使能/禁止 V_{REFINT} 通道。

0：禁止 V_{REFINT}
1：使能 V_{REFINT}

注： 仅当 ADSTART=0 时（这可确保当前未进行任何转换），才允许通过软件对此位执行写操作。

位 21:18 **PRESC[3:0]**：ADC 预分频系数 (ADC prescaler)

此位由软件置 1 和清零，用于选择 ADC 的时钟频率。该时钟为所有 ADC 所共用。

0000：输入 ADC 时钟未分频
0001：输入 ADC 时钟 2 分频
0010：输入 ADC 时钟 4 分频
0011：输入 ADC 时钟 6 分频
0100：输入 ADC 时钟 8 分频
0101：输入 ADC 时钟 10 分频
0110：输入 ADC 时钟 12 分频
0111：输入 ADC 时钟 16 分频
1000：输入 ADC 时钟 32 分频
1001：输入 ADC 时钟 64 分频
1010：输入 ADC 时钟 128 分频
1011：输入 ADC 时钟 256 分频
其他：保留

注： 仅当 ADC 已禁止时（ADCAL=0、ADSTART=0、ADSTP=0、ADDIS=0 且 ADEN=0），才允许通过软件对这些位执行写操作。

位 17:0 保留，必须保持复位值。

14.13.12 ADC 寄存器映射

下表对 ADC 寄存器进行了汇总。

表 64. ADC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOCAL.	Res.	Res.	Res.	AWD	Res.	Res.	OVR	EOSEQ	EOC	EOSMP	ADRDY
	Reset value																					0				0			0	0	0	0	0
0x04	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOCALIE	Res.	Res.	Res.	AWDIE	Res.	Res.	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																					0				0			0	0	0	0	0
0x08	ADC_CR	ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
	Reset value	0			0																								0		0	0	0
0x0C	ADC_CFGR1	Res.	AWDCH[4:0]					Res.	Res.	AWDEN	AWDSGL	Res.	Res.	Res.	Res.	Res.	DISCEN	AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL [2:0]			ALIGN	RES [1:0]		SCANDIR	DMACFG	DMAEN
	Reset value		0	0	0	0	0			0	0						0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x10	ADC_CFGR2	CKMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS [3:0]				OVSR [2:0]			Res.	OVSE
	Reset value	0	0																					0	0	0	0	0	0	0	0		0
0x14	ADC_SMPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMP [2:0]		
	Reset value																														0	0	0
0x18	Reserved	Reserved																															
0x1C	Reserved	Reserved																															
0x20	ADC_TR	Res.	Res.	Res.	Res.	HT[11:0]												Res.	Res.	Res.	Res.	LT[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	Reserved	Reserved																															
0x28	ADC_CHSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL18	CHSEL17	CHSEL16	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C 0x30 0x34 0x38 0x3C	Reserved	Reserved																															
0x40	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44 ... 0xB0	Reserved	Reserved																															
0xB4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0xB8 ... 0x304	Reserved	Reserved																															

表 64. ADC 寄存器映射和复位值（续）

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
0x308	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	LFMEN	VLCDEN	TSEN	VREFEN	PRESC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0	0	0	0	0	0	0																		

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

15 数模转换器 (DAC)

15.1 简介

DAC 模块是 12 位电压输出数模转换器。DAC 可以按 8 位或 12 位模式进行配置，并且可与 DMA 控制器配合使用。在 12 位模式下，数据可以采用左对齐或右对齐。可使用输入参考电压 VREF+（与 ADC 共享）。可以有选择地缓冲输出以实现更高的电流驱动。

15.2 DAC1 主要特性

器件集成了两个 DAC 转换器，各具有一个输出通道：DAC_OUT1 和 DAC_OUT2。

DAC1 主要特性如下：

- 一个数据保持寄存器
- 12 位模式下数据采用左对齐或右对齐
- 同步更新功能
- 生成噪声波
- 生成三角波
- 双 DAC 通道，支持单独或同时转换
- DMA 功能（包括下溢检测）
- 通过外部触发信号进行转换
- 输入参考电压 V_{REF+}

图 54 所示为 DAC 通道的框图，*表 65* 则给出了引脚说明。

图 54. DAC 框图

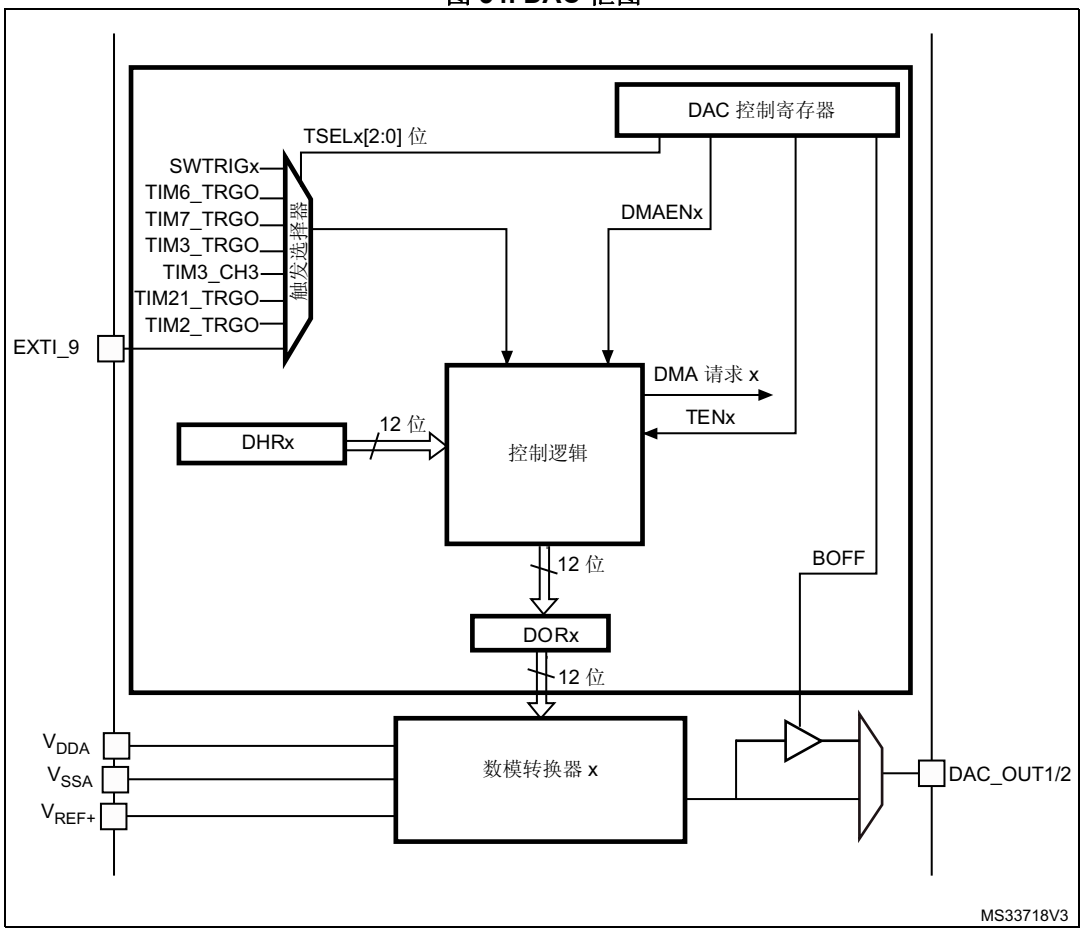


表 65. DAC 引脚

名称	信号类型	备注
V_{DDA}	模拟电源输入	模拟电源
V_{SSA}	模拟电源地输入	模拟电源地
V_{REF+}	正模拟参考电压输入	DAC1 的高端 / 正极参考电压
DAC_OUT1/2	模拟输出信号	DAC 通道 x 模拟输出

注： *使能 DAC_Channelx 后，相应 GPIO 引脚（PA4 或 PA5）将自动连接到模拟转换器输出 (DAC_OUTx)。为了避免寄生电流消耗，应首先将 PA4/PA5 引脚配置为模拟模式 (AIN)。*

15.3 DAC 输出缓冲器使能

DAC 集成了两个输出缓冲器，可用来降低输出阻抗并在不增加外部运算放大器的情况下直接驱动外部负载。

通过 DAC_CR 寄存器中的 BOFF1 位，可使能或禁止 DAC 通道输出缓冲器。

15.4 DAC 通道使能

将 DAC_CR 寄存器中的相应 ENx 位置 1，即可使能对应 DAC 通道。经过一段启动时间 t_{WAKEUP} 后，各 DAC 通道被真正使能。

注： *ENx 位只会使能模拟 DAC 通道 x 宏单元。即使 ENx 位复位，DAC 通道 x 数字接口仍处于使能状态。*

15.5 单模式功能说明

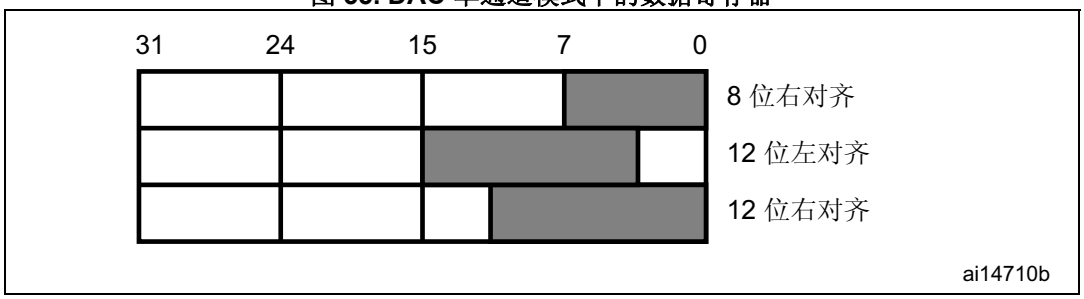
15.5.1 DAC 数据格式

根据所选配置模式，数据必须按如下方式写入指定寄存器：

- 存在三种可能：
  - 8 位右对齐：软件必须将数据加载到 DAC_DHR8Rx [7:0] 位（存储到 DHRx[11:4] 位）
  - 12 位左对齐：软件必须将数据加载到 DAC_DHR12Lx [15:4] 位（存储到 DHRx[11:0] 位）
  - 12 位右对齐：软件必须将数据加载到 DAC_DHR12Rx [11:0] 位（存储到 DHRx[11:0] 位）

根据加载的 DAC_DHRyyyx 寄存器，用户写入的数据将移位并存储到相应的 DHRx（数据保持寄存器 x，即内部非存储器映射寄存器）。之后，DHRx 寄存器将被自动加载，或者通过软件或外部事件触发加载到 DORx 寄存器。

图 55. DAC 单通道模式下的数据寄存器



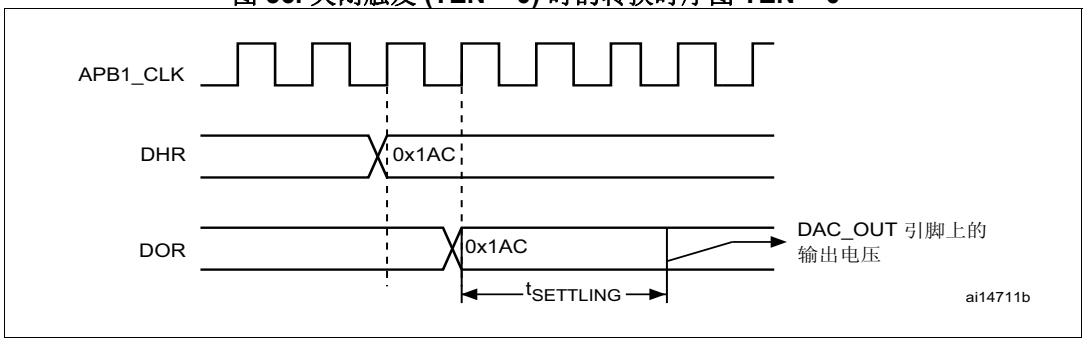
15.5.2 DAC 通道转换

DAC_DORx 无法直接写入，任何数据都必须通过加载 DAC_DHRx 寄存器（写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx）才能传输到 DAC 通道 x。

如果未选择硬件触发（DAC_CR 寄存器中的 TENx 位复位），那么经过一个 APB1 时钟周期后，DAC_DHRx 寄存器中存储的数据将自动转移到 DAC_DORx 寄存器。但是，如果选择硬件触发（置位 DAC_CR 寄存器中的 TENx 位）且触发条件到来，将在三个 PCLK1 时钟周期后进行转移。

当 DAC_DORx 加载了 DAC_DHRx 内容时，模拟输出电压将在一段时间 $t_{SETTLING}$ 后可用，具体时间取决于电源电压和模拟输出负载。

图 56. 关闭触发 (TEN = 0) 时的转换时序图 TEN = 0



生成单个LFSR的独立触发

要将 DAC 配置为此转换模式（请参见*第 15.7 节：生成噪声*），需要遵循以下顺序：

1. 将 DAC 通道触发使能位 TENx 置 1。
2. 通过设置 TSELx[2:0] 位配置触发源。
3. 将 DAC 通道的 WAVEx[1:0] 位配置为“01”，并在 MAMPx[3:0] 位中配置相同的 LFSR 掩码值
4. 将 DAC 通道数据加载到所需 DAC_DHRx 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

DAC 通道 x 触发信号到达时，LFSRx 计数器内容（使用相同的掩码）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。LFSRx 计数器随即更新。

有关代码示例，请参见 *A.9.1：独立触发（不产生波形）代码示例*。

生成单个三角波的独立触发

要将 DAC 配置为此转换模式（请参见*第 15.8 节：生成三角波*），需要遵循以下顺序：

1. 将 DAC 通道 x 触发使能位 TENx 置 1。
2. 通过设置 TSELx[2:0] 位配置触发源。
3. 将 DAC 通道 x 的 WAVEx[1:0] 位配置为“1x”，并在 MAMPx[3:0] 位中配置相同的最大振幅值
4. 将 DAC 通道 x 数据加载到所需 DAC_DHRx 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

DAC 通道 x 触发信号到达时，DAC 通道 x 三角波计数器内容（使用相同的三角波振幅）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。DAC 通道 x 三角波计数器随即更新。

有关代码示例，请参见 *A.9.2：独立触发（生成单个三角波）代码示例*。

15.5.3 DAC 输出电压

经过线性转换后，数字输入会转换为 0 到 V_{REF+} 之间的输出电压。

各 DAC 通道引脚的模拟输出电压通过以下公式确定：

$$DACoutput = V_{REF+} \times \frac{DOR}{4096}$$

15.5.4 DAC 触发选择

如果 TENx 控制位置 1，可通过外部事件（定时计数器、外部中断线）触发转换。TSELx[2:0] 控制位将决定哪个可能事件将触发转换，如*表 66* 所示。

表 66. 外部触发器

源	类型	TSEL[2:0]
TIM6 TRGO 事件	片上定时器的内部信号	000
TIM3 TRGO 事件		001
TIM3 CH3 事件		010
TIM21 TRGO 事件		011
TIM2 TRGO 事件		100
TIM7 TRGO 事件		101
EXTI 线 9	外部引脚	110
SWTRIG	软件控制位	111

每当 DAC 接口在所选定时器 TRGO 输出或所选外部中断线 9 上检测到上升沿时，DAC_DHRx 寄存器中存储的最后一个数据即会转移到 DAC_DORx 寄存器中。发生触发后再经过三个 APB1 周期，DAC_DORx 寄存器将会得到更新。

如果选择软件触发，一旦 SWTRIG 位置 1，转换即会开始。DAC_DHRx 寄存器内容加载到 DAC_DORx 寄存器中后，SWTRIG 即由硬件复位。

注：ENx 位置 1 时，无法更改 TSELx[2:0] 位。如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DORx 寄存器。

15.6 双模式功能说明

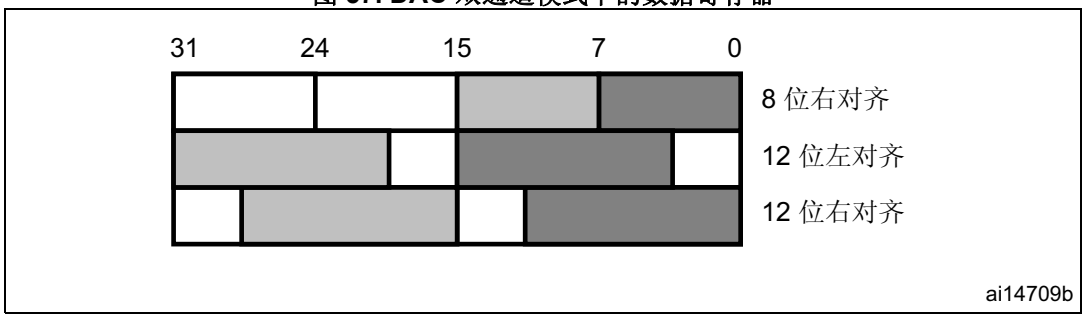
15.6.1 DAC 数据格式

在 DAC 双通道模式下，有三种可能的方式：

- 8 位右对齐：将 DAC 1 通道的数据加载到 DAC_DHR8RD [7:0] 位（存储到 DHR1[11:4] 位），将 DAC 2 通道的数据加载到 DAC_DHR8RD [15:8] 位（存储到 DHR2[11:4] 位）
- 12 位左对齐：将 DAC 1 通道的数据加载到 DAC_DHR12RD [15:4] 位（存储到 DHR1[11:0] 位），将 DAC 2 通道的数据加载到 DAC_DHR12RD [31:20] 位（存储到 DHR2[11:0] 位）
- 12 位右对齐：将 DAC 1 通道的数据加载到 DAC_DHR12RD [11:0] 位（存储到 DHR1[11:0] 位），将 DAC 2 通道的数据加载到 DAC_DHR12RD [27:16] 位（存储到 DHR2[11:0] 位）

根据加载的 DAC_DHRyyyD 寄存器，用户写入的数据将移位并存储到 DHR1 和 DHR2（数据保持寄存器，即内部非存储器映射寄存器）。之后，DHR1 和 DHR2 寄存器将被自动加载，或者通过软件或外部事件触发分别被加载到 DOR1 和 DOR2 寄存器。

图 57. DAC 双通道模式下的数据寄存器



15.6.2 双模式下的 DAC 通道转换

双模式下的 DAC 通道转换执行方式与单模式下基本相同（请参见*第 15.5.2 节*），只是必须通过写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12LD 或 DAC_DHR12RD 加载数据。

15.6.3 双转换模式说明

为了在同时需要两个 DAC 通道的应用中有效利用总线带宽，DAC 模块有三个双寄存器可供操作：DHR8RD、DHR12RD 和 DHR12LD。这样，只需一个寄存器访问即可同时驱动两个 DAC 通道。

通过两个 DAC 通道和这三个双寄存器可以实现 11 种转换模式。但如果需要，所有这些转换模式也都可以通过单独的 DHRx 寄存器来实现。

下面几段内容将介绍所有这些模式。

有关为 DAC 转换计时的 AHB 总线（APB 或 APB1）的详细信息，请参见*第 15.5.2 节：DAC 通道转换*。

独立触发（不产生波形）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
3. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 1 通道触发信号到达时，DHR1 寄存器的内容转移到 DAC_DOR1（三个 APB 时钟周期之后）。

DAC 2 通道触发信号到达时，DHR2 寄存器的内容转移到 DAC_DOR2（三个 APB 时钟周期之后）。

独立触发（生成单个 LFSR）

要将 DAC 配置为此转换模式（请参见*第 15.7 节：生成噪声*），需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
3. 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMPx[3:0] 位中配置相同的 LFSR 掩码值
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB 时钟周期之后）。LFSR2 计数器随即更新。

独立触发（生成不同 LFSR）

要将 DAC 配置为此转换模式（请参见*第 15.7 节：生成噪声*），需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为不同的值，以配置不同的触发源
3. 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的 LFSR 掩码值
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

DAC 通道 1 触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0] 配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB 时钟周期之后）。LFSR1 计数器随即更新。

DAC 通道 2 触发信号到达时，LFSR2 计数器内容（使用 MAMP2[3:0] 配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB 时钟周期之后）。LFSR2 计数器随即更新。

独立触发（生成单个三角波）

要将 DAC 配置为此转换模式（请参见*第 15.8 节：生成三角波*），需要遵循以下顺序：

1. 将 DAC 通道 x 触发使能位 TENx 置 1。
2. 通过在 TSELx[2:0] 位中设置不同的值，以配置不同的触发源
3. 将 DAC 通道 x 的 WAVEx[1:0] 位配置为“1x”，并在 MAMPx[3:0] 位中配置相同的最大振幅值
4. 将 DAC 通道 x 数据加载到所需 DAC_DHRx 寄存器

有关为 DAC 转换计时的 AHB 总线（APB 或 APB1）的详细信息，请参见 *第 15.5.2 节：DAC 通道转换*。

DAC 通道 x 触发信号到达时，DAC 通道 x 三角波计数器内容（使用相同的三角波振幅）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。DAC 通道 x 三角波计数器随即更新。

独立触发（生成不同三角波）

要将 DAC 配置为此转换模式（请参见*第 15.8 节：生成三角波*），需要遵循以下顺序：

1. 将 DAC 通道 x 触发使能位 TENx 置 1。
2. 通过在 TSELx[2:0] 位中设置不同的值，以配置不同的触发源
3. 将 DAC 通道 x 的 WAVEx[1:0] 位设置为“1x”，并在 MAMPx[3:0] 位中设置不同的最大振幅值
4. 将 DAC 通道 x 数据加载到所需 DAC_DHRx 寄存器

DAC 通道 x 触发信号到达时，DAC 通道 x 三角波计数器内容（使用 MAMPx[3:0] 配置的三角波振幅）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。DAC 通道 x 三角波计数器随即更新。

同步软件启动

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

该配置用一个 APB 时钟周期。

同步触发（不产生波形）

要将 DAC 配置为此转换模式，需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
3. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

当触发信号到达时，DHR1 和 DHR2 寄存器内容将分别转移到 DAC_DOR1 和 DAC_DOR2 中（三个 APB 时钟周期之后）。

同步触发（生成单个 LFSR）

要将 DAC 配置为此转换模式（请参见*第 15.7 节：生成噪声*），需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
3. 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMPx[3:0] 位中配置相同的 LFSR 掩码值
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用相同的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB 时钟周期之后）。LFSR1 计数器随即更新。同时，LFSR2 计数器内容（使用相同的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB 时钟周期之后）。LFSR2 计数器随即更新。

同步触发（生成不同 LFSR）

要将 DAC 配置为此转换模式（请参见*第 15.7 节：生成噪声*），需要遵循以下顺序：

1. 将两个 DAC 通道触发使能位 TEN1 和 TEN2 置 1
2. 将 TSEL1[2:0] 和 TSEL2[2:0] 设置为相同的值，以便为两个 DAC 通道配置相同的触发源
3. 将两个 DAC 通道的 WAVEx[1:0] 设置为“01”，并在 MAMP1[3:0] 和 MAMP2[3:0] 位中设置不同的 LFSR 掩码值
4. 将 DAC 双通道数据加载到所需 DHR 寄存器（DAC_DHR12RD、DAC_DHR12LD 或 DAC_DHR8RD）

触发信号到达时，LFSR1 计数器内容（使用 MAMP1[3:0] 配置的掩码）与 DHR1 寄存器内容相加，所得总和转移到 DAC_DOR1 中（三个 APB 时钟周期之后）。LFSR1 计数器随即更新。

同时，LFSR2 计数器内容（使用 MAMP2[3:0] 配置的掩码）与 DHR2 寄存器内容相加，所得总和转移到 DAC_DOR2 中（三个 APB 时钟周期之后）。LFSR2 计数器随即更新。

同步触发（生成单个三角波）

要将 DAC 配置为此转换模式（请参见*第 15.8 节：生成三角波*），需要遵循以下顺序：

1. 将 DAC 通道 x 触发使能位 TEN1x 置 1。
2. 在 TSELx[2:0] 位中设置相同的值，以便为两个 DAC 通道配置相同的触发源。
3. 将 DAC 通道 x 的 WAVEx[1:0] 位配置为“1x”，并使用 MAMPx[3:0] 位配置相同的最大振幅值
4. 将 DAC 通道 x 数据加载到所需 DAC_DHRx 寄存器。

触发信号到达时，DAC 通道 x 三角波计数器内容（使用相同的三角波振幅）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。DAC 通道 x 三角波计数器随即更新。

同步触发（生成不同三角波）

要将 DAC 配置为此转换模式（请参见*第 15.8 节：生成三角波*），需要遵循以下顺序：

1. 将 DAC 通道 x 触发使能位 TENx 置 1。
2. 在 TSELx[2:0] 位中设置相同的值，以便为 DAC 通道 x 配置相同的触发源。
3. 将 DAC 通道 x 的 WAVEx[1:0] 位配置为“1x”，并在 MAMPx[3:0] 位中设置不同的最大振幅值。
4. 将 DAC 通道 x 数据加载到所需 DAC_DHRx 寄存器。

触发信号到达时，DAC 通道 x 三角波计数器内容（使用 MAMPx[3:0] 配置的三角波振幅）与 DHRx 寄存器内容相加，所得总和转移到 DAC_DORx 中（三个 APB 时钟周期之后）。DAC 通道 x 三角波计数器随即更新。

15.6.4 DAC 输出电压

参见*第 15.5.3 节：DAC 输出电压*。

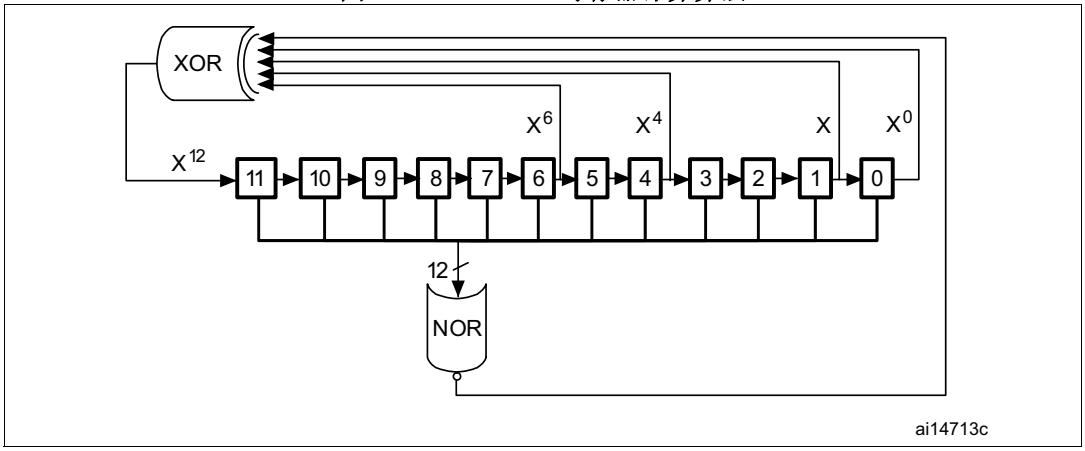
15.6.5 DAC 触发选择

请参见*第 15.5.4 节：DAC 触发选择*。

15.7 生成噪声

为了生成可变振幅的伪噪声，可使用 LFSR（线性反馈移位寄存器）。将 WAVEx[1:0] 置为“01”即可选择生成噪声。LFSR 中的预加载值为 0xAAA。在每次发生触发事件后，经过三个 APB 时钟周期，该寄存器会依照特定的计算算法完成更新。

图 58. DAC LFSR 寄存器计算算法

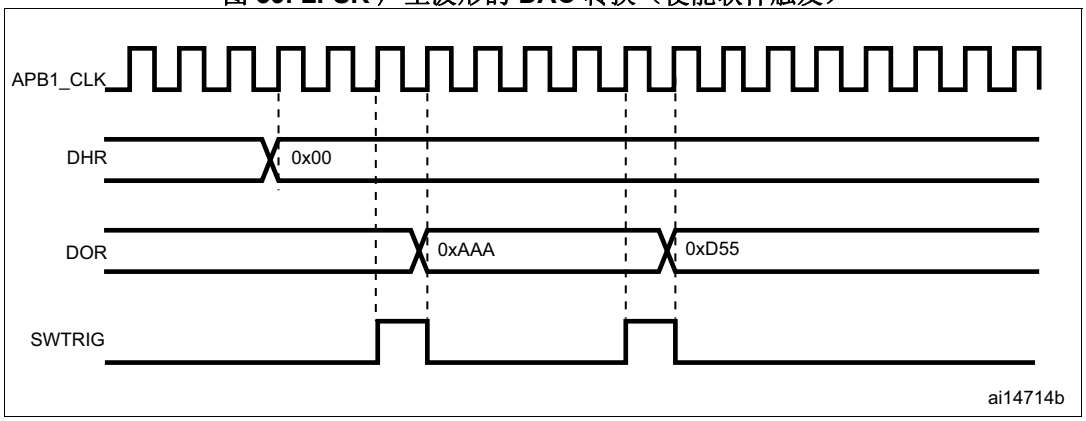


LFSR 值可以通过 DAC_CR 寄存器中的 MAMPx[3:0] 位来部分或完全屏蔽，在不发生溢出的情况下，该值将与 DAC_DHRx 的内容相加，然后存储到 DAC_DORx 寄存器中。

如果 LFSR 为 0x0000，将向其注入“1”（防锁定机制）。

可以通过复位 WAVEx[1:0] 位来将 LFSR 波形产生功能关闭。

图 59. LFSR 产生波形的 DAC 转换（使能软件触发）



注： *要生成三角波，必须通过将 DAC_CR 寄存器中的 TENx 位置 1 来使能 DAC 触发。*

15.8 生成三角波

可以在直流电流或慢变信号上叠加一个小幅三角波。将 WAVEx[1:0] 置为“10”即可选择 DAC 生成三角波。振幅通过 DAC_CR 寄存器中的 MAMPx[3:0] 位进行配置。每次发生触发事件后，经过三个 APB 时钟周期，内部三角波计数器将会递增。在不发生溢出的情况下，该计数器的值将与 DAC_DHRx 寄存器内容相加，所得总和将存储到 DAC_DORx 寄存器中。只要小于 MAMPx[3:0] 位定义的最大振幅，三角波计数器就会一直递增。一旦达到配置的振幅，计数器将递减至零，然后再递增，以此类推。

可以通过复位 WAVEx[1:0] 位来将三角波产生功能关闭。

图 60. 生成 DAC 三角波

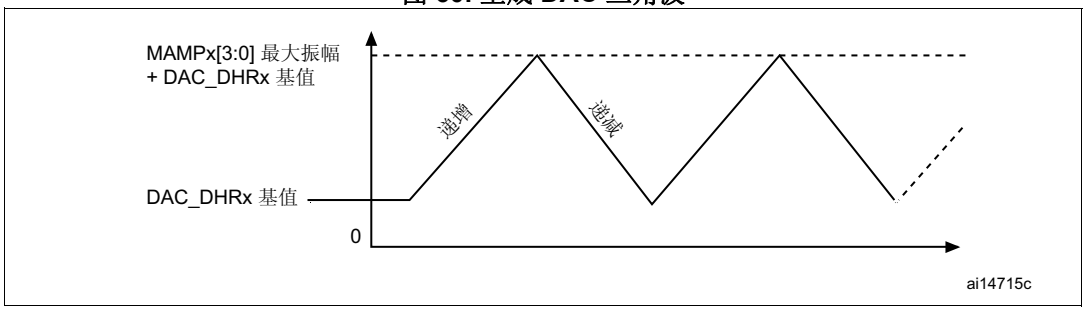
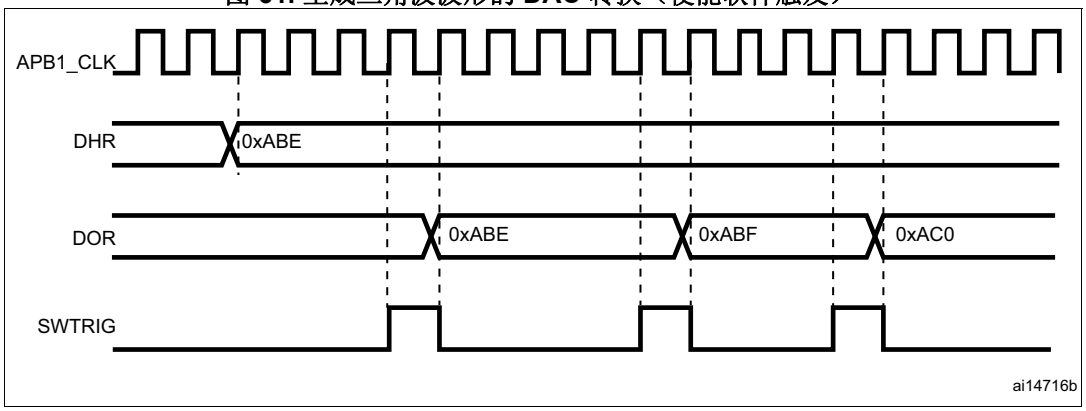


图 61. 生成三角波波形的 DAC 转换（使能软件触发）



注： *要生成三角波，必须通过将 DAC_CR 寄存器中的 TENx 位置 1 来使能 DAC 触发。MAMPx[3:0] 位必须在使能 DAC 之前进行配置，否则将无法更改。*

15.9 DMA 请求

每个 DAC 通道都具有 DMA 功能。两个 DMA 通道用于处理 DAC 通道的 DMA 请求。

当 DMAENx 位置 1 时，如果发生外部触发（而不是软件触发），则将产生 DAC DMA 请求。DAC_DHRx 寄存器的值随后转移到 DAC_DORx 寄存器。

在双通道模式下，如果两个 DMAENx 位均置 1，则将产生两个 DMA 请求。如果只需要一个 DMA 请求，用户应仅将相应 DMAENx 位置 1。这样，应用程序可以在双通道模式下通过一个 DMA 请求和一个特定 DMA 通道来管理两个 DAC 通道。

DMA 下溢

DAC DMA 请求没有缓冲队列。这样，如果第二个外部触发到达时尚未收到第一个外部触发的应答，将不会发出新的请求，并且 DAC_SR 寄存器中的 DAM 通道下溢标志 DMAUDRx 将置 1，以报告这一错误状况。DMA 数据传输随即禁止，并且不再处理其他 DMA 请求。DAC 通道仍将继续转换旧有数据。

软件应通过写入“1”来将 DMAUDRx 标志清零，将所用 DMA 数据流的 DMAEN 位清零，并重新初始化 DMA 和 DAC 通道，以便正确地重新开始 DMA 传输。软件应修改 DAC 触发转换频率或减轻 DMA 工作负载，以避免再次发生 DMA 下溢。最后，可通过使能 DMA 数据传输和转换触发来继续完成 DAC 转换。

对于各 DAC 通道，如果使能 DAC_CR 寄存器中相应的 DMAUDRIEx 位，还将产生中断。

有关代码示例，请参见 *A.9.3：DMA 初始化代码示例*。

15.10 DAC 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器必须按字（32 位）进行访问。

15.10.1 DAC 控制寄存器 (DAC_CR)

DAC control register

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
Res.	Res.	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:30 保留，必须保持复位值。

位 29 **DMAUDRIE2**：DAC 2 通道 DMA 下溢中断使能 (DAC channel2 DMA underrun interrupt enable)
此位由软件置 1 和清零。
0：禁止 DAC 2 通道 DMA 下溢中断
1：使能 DAC 2 通道 DMA 下溢中断
注： 此位仅在双模式下可用，在单模式下保留。

位 28 **DMAEN2**：DAC 2 通道 DMA 使能 (DAC channel2 DMA enable)
此位由软件置 1 和清零。
0：禁止 DAC 2 通道 DMA 模式
1：使能 DAC 2 通道 DMA 模式
注： 此位仅在双模式下可用，在单模式下保留。

位 27:24 **MAMP2[3:0]**：DAC 2 通道掩码/振幅选择器 (DAC channel2 mask/amplitude selector)
这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。
0000：不屏蔽 LFSR 的位 0/三角波振幅等于 1
0001：不屏蔽 LFSR 的位 [1:0]/三角波振幅等于 3
0010：不屏蔽 LFSR 的位 [2:0]/三角波振幅等于 7
0011：不屏蔽 LFSR 的位 [3:0]/三角波振幅等于 15
0100：不屏蔽 LFSR 的位 [4:0]/三角波振幅等于 31
0101：不屏蔽 LFSR 的位 [5:0]/三角波振幅等于 63
0110：不屏蔽 LFSR 的位 [6:0]/三角波振幅等于 127
0111：不屏蔽 LFSR 的位 [7:0]/三角波振幅等于 255
1000：不屏蔽 LFSR 的位 [8:0]/三角波振幅等于 511
1001：不屏蔽 LFSR 的位 [9:0]/三角波振幅等于 1023
1010：不屏蔽 LFSR 的位 [10:0]/三角波振幅等于 2047
≥1011：不屏蔽 LFSR 的位 [11:0]/三角波振幅等于 4095
注： 仅当支持波形生成时，这些位才在双模式下可用。否则，它们被保留，并必须保持复位值。

位 23:22 **WAVE2[1:0]**：DAC 2 通道噪声/三角波生成使能 (DAC channel2 noise/triangle wave generation enable)

这些位由软件置 1 或清零。

00：禁止生成波
01：使能生成噪声波
1x：使能生成三角波

注：只在位 TEN2 = 1（使能 DAC 2 通道触发）时使用
仅当支持波形生成时，这些位才在双模式下可用。否则，它们被保留，并必须保持复位值。

位 21:19 **TSEL2[2:0]**：DAC 2 通道触发器选择 (DAC channel2 trigger selection)

这些位用于选择 DAC 2 通道的外部触发事件

000：定时器 6 TRGO 事件
001：定时器 3 TRGO 事件
010：定时器 3 CH3 事件
011：定时器 21 TRGO 事件
100：定时器 2 TRGO 事件
101：定时器 7 TRGO 事件
110：EXTI 线 9
111：软件触发

注：只在位 TEN2 = 1（使能 DAC 2 通道触发）时使用。
这些位仅在双模式下可用，在单模式下保留。

位 18 **TEN2**：DAC 2 通道触发使能 (DAC channel2 trigger enable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道触发

0：禁止 DAC 2 通道触发，写入 DAC_DHRx 寄存器的数据在一个 APB1 时钟周期之后转移到 DAC_DOR2 寄存器
1：使能 DAC 2 通道触发，DAC_DHRx 寄存器的数据在三个 APB1 时钟周期之后转移到 DAC_DOR2 寄存器

注：如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR2 寄存器。
注：此位仅在双模式下可用，在单模式下保留。

位 17 **BOFF2**：DAC 2 通道输出缓冲器禁止 (DAC channel2 output buffer disable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道输出缓冲器。

0：使能 DAC 2 通道输出缓冲器
1：禁止 DAC 2 通道输出缓冲器

注：此位仅在双模式下可用，在单模式下保留。

位 16 **EN2**：DAC 2 通道使能 (DAC channel2 enable)

此位由软件置 1 和清零，以使能/禁止 DAC 2 通道。

0：禁止 DAC 2 通道
1：使能 DAC 2 通道

注：此位仅在双模式下可用，在单模式下保留。

位 15:14 保留，必须保持复位值。

位 13 **DMAUDRIE1**：DAC 1 通道 DMA 下溢中断使能 (DAC channel1 DMA Underrun Interrupt enable)

此位由软件置 1 和清零。

0：禁止 DAC 1 通道 DMA 下溢中断
1：使能 DAC 1 通道 DMA 下溢中断

位 12 **DMAEN1**：DAC 1 通道 DMA 使能 (DAC channel1 DMA enable)

此位由软件置 1 和清零。

0：禁止 DAC 1 通道 DMA 模式
1：使能 DAC 1 通道 DMA 模式

位 11:8 **MAMP1[3:0]**：DAC 1 通道掩码/振幅选择器 (DAC channel1 mask/amplitude selector)

这些位由软件写入，用于在生成噪声波模式下选择掩码，或者在生成三角波模式下选择振幅。

0000：不屏蔽 LFSR 的位 0/三角波振幅等于 1
0001：不屏蔽 LFSR 的位 [1:0]/三角波振幅等于 3
0010：不屏蔽 LFSR 的位 [2:0]/三角波振幅等于 7
0011：不屏蔽 LFSR 的位 [3:0]/三角波振幅等于 15
0100：不屏蔽 LFSR 的位 [4:0]/三角波振幅等于 31
0101：不屏蔽 LFSR 的位 [5:0]/三角波振幅等于 63
0110：不屏蔽 LFSR 的位 [6:0]/三角波振幅等于 127
0111：不屏蔽 LFSR 的位 [7:0]/三角波振幅等于 255
1000：不屏蔽 LFSR 的位 [8:0]/三角波振幅等于 511
1001：不屏蔽 LFSR 的位 [9:0]/三角波振幅等于 1023
1010：不屏蔽 LFSR 的位 [10:0]/三角波振幅等于 2047
≥ 1011：不屏蔽 LFSR 的位 [11:0]/三角波振幅等于 4095

位 7:6 **WAVE1[1:0]**：DAC 1 通道噪声/三角波生成使能 (DAC channel1 noise/triangle wave generation enable)

这些位由软件置 1 和清零。

00：禁止生成波
01：使能生成噪声波
1x：使能生成三角波

注：　只在位 TEN1 = 1（使能 DAC 1 通道触发）时使用。

位 5:3 **TSEL1[2:0]**：DAC 1 通道触发器选择 (DAC channel1 trigger selection)

这些位用于选择 DAC 1 通道的外部触发事件。

000：定时器 6 TRGO 事件
001：定时器 3 TRGO 事件
010：定时器 3 CH3 事件
011：定时器 21 TRGO 事件
100：定时器 2 TRGO 事件
101：定时器 7 TRGO 事件
110：EXTI 线 9
111：软件触发

注：　只在位 TEN1 = 1（使能 DAC 1 通道触发）时使用。

位 2 **TEN1**：DAC 1 通道触发使能 (DAC channel1 trigger enable)

此位由软件置 1 和清零，以使能/禁止 DAC 1 通道触发。

0：禁止 DAC 1 通道触发，写入 DAC_DHRx 寄存器的数据在一个 APB1 时钟周期之后转移到 DAC_DOR1 寄存器

1：使能 DAC 1 通道触发，DAC_DHRx 寄存器的数据在三个 APB1 时钟周期之后转移到 DAC_DOR1 寄存器

注：　如果选择软件触发，DAC_DHRx 寄存器的内容只需一个 APB1 时钟周期即可转移到 DAC_DOR1 寄存器。

位 1 **BOFF1**：DAC 1 通道输出缓冲器禁止 (DAC channel1 output buffer disable)

此位由软件置 1 和清零，以使能/禁止 DAC 1 通道输出缓冲器。

0：使能 DAC 1 通道输出缓冲器

1：禁止 DAC 1 通道输出缓冲器

位 0 **EN1**：DAC 1 通道使能 (DAC channel1 enable)

此位由软件置 1 和清零，以使能/禁止 DAC 1 通道。

0：禁止 DAC 1 通道

1：使能 DAC 1 通道

15.10.2 DAC 软件触发寄存器 (DAC_SWTRIGR)

DAC software trigger register

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

位 31:2 保留，必须保持复位值。

位 1 **SWTRIG2**：DAC 2 通道软件触发 (DAC channel2 software trigger)

此位由软件置 1 和清零，以使能/禁止软件触发。

0：禁止软件触发

1：使能软件触发

注：　一旦 DAC_DHR2 寄存器值加载到 DAC_DOR2 寄存器中，该位即会由硬件清零（一个 APB1 时钟周期之后）。

此位仅在双模式下可用，在单模式下保留。

位 0 **SWTRIG1**：DAC 1 通道软件触发 (DAC channel1 software trigger)

此位由软件置 1 和清零，以使能/禁止软件触发。

0：禁止软件触发

1：使能软件触发

注：　一旦 DAC_DHR1 寄存器值加载到 DAC_DOR1 寄存器中，该位即会由硬件清零（一个 APB1 时钟周期之后）。

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

DAC channel1 12-bit right-aligned data holding register

偏移地址：0x08

复位值：0x0000 0000

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

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

位 31:12 保留，必须保持复位值。

位 11:0 **DACC1DHR[11:0]**：DAC 1 通道 12 位右对齐数据 (DAC channel1 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

15.10.4 DAC 1 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L1)

DAC channel1 12-bit left-aligned data holding register

偏移地址：0x0C

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:4 **DACC1DHR[11:0]**：DAC 1 通道 12 位左对齐数据 (DAC channel1 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

15.10.5 DAC 1 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R1)

DAC channel1 8-bit right-aligned data holding register

偏移地址：0x10

复位值：0x0000 0000

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

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

位 31:8 保留，必须保持复位值。

位 7:0 **DACC1DHR[7:0]**：DAC 1 通道 8 位右对齐数据 (DAC channel1 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 8 位数据。

15.10.6 DAC 2 通道 12 位右对齐数据保持寄存器 (DAC_DHR12R2)

DAC channel2 12-bit right-aligned data holding register

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:12 保留，必须保持复位值。

位 11:0 **DACC2DHR[11:0]**：DAC 2 通道 12 位右对齐数据 (DAC channel2 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

15.10.7 DAC 2 通道 12 位左对齐数据保持寄存器 (DAC_DHR12L2)

DAC channel2 12-bit left-aligned data holding register

偏移地址：0x18

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:16 保留，必须保持复位值。

位 15:4 **DACC2DHR[11:0]**：DAC 2 通道 12 位左对齐数据 (DAC channel2 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

15.10.8 DAC 2 通道 8 位右对齐数据保持寄存器 (DAC_DHR8R2)

DAC channel2 8-bit right-aligned data holding register

偏移地址：0x1C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值。

位 7:0 **DACC2DHR[7:0]**：DAC 2 通道 8 位右对齐数据 (DAC channel2 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 8 位数据。

15.10.9 双 DAC 12 位右对齐数据保持寄存器 (DAC_DHR12RD)

Dual DAC 12-bit right-aligned data holding register

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:16 **DACC2DHR[11:0]**：DAC 2 通道 12 位右对齐数据 (DAC channel2 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 15:12 保留，必须保持复位值。

位 11:0 **DACC1DHR[11:0]**：DAC 1 通道 12 位右对齐数据 (DAC channel1 12-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

15.10.10 双 DAC 12 位左对齐数据保持寄存器 (DAC_DHR12LD)

Dual DAC 12-bit left-aligned data holding register

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:20 **DACC2DHR[11:0]**：DAC 2 通道 12 位左对齐数据 (DAC channel2 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 12 位数据。

位 19:16 保留，必须保持复位值。

位 15:4 **DACC1DHR[11:0]**：DAC 1 通道 12 位左对齐数据 (DAC channel1 12-bit left-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 12 位数据。

位 3:0 保留，必须保持复位值。

15.10.11 双 DAC 8 位右对齐数据保持寄存器 (DAC_DHR8RD)

Dual DAC 8-bit right-aligned data holding register

偏移地址：0x28

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值。

位 15:8 **DACC2DHR[7:0]**：DAC 2 通道 8 位右对齐数据 (DAC channel2 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 2 通道指定 8 位数据。

位 7:0 **DACC1DHR[7:0]**：DAC 1 通道 8 位右对齐数据 (DAC channel1 8-bit right-aligned data)
这些位由软件写入，用于为 DAC 1 通道指定 8 位数据。

15.10.12 DAC 1 通道 数据输出寄存器 (DAC_DOR1)

DAC channel1 data output register

偏移地址：0x2C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

位 31:12 保留，必须保持复位值。

位 11:0 **DACC1DOR[11:0]**：DAC 1 通道数据输出 (DAC channel1 data output)
这些位为只读，其中包含 DAC 1 通道的数据输出。

15.10.13 DAC 2 通道数据输出寄存器 (DAC_DOR2)

DAC channel2 data output register

偏移地址：0x30
复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

位 31:12 保留，必须保持复位值。

位 11:0 **DACC2DOR[11:0]**：DAC 2 通道数据输出 (DAC channel2 data output)
这些位为只读，其中包含 DAC 2 通道的数据输出。

15.10.14 DAC 状态寄存器 (DAC_SR)

DAC status register

偏移地址：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rc_w1													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rc_w1													

位 31:30 保留，必须保持复位值。

位 29 **DMAUDR2**：DAC 2 通道 DMA 下溢标志 (DAC channel2 DMA underrun flag)
此位由硬件置 1，由软件清零（写入 1）。
0：DAC 2 通道未发生 DMA 下溢错误状况
1：DAC 2 通道发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动 DAC 2 通道转换）
注： 此位仅在双模式下可用，在单模式下保留。

位 28:14 保留，必须保持复位值。

位 13 **DMAUDR1**：DAC 1 通道 DMA 下溢标志 (DAC channel1 DMA underrun flag)
此位由硬件置 1，由软件清零（写入 1）。
0：DAC 1 通道未发生 DMA 下溢错误状况
1：DAC 1 通道发生 DMA 下溢错误状况（当前所选触发源以高于 DMA 服务能力的频率驱动 DAC 1 通道转换）

位 12:0 保留，必须保持复位值。

15.10.15 DAC 寄存器映射

表 67 汇总了 DAC 寄存器。

表 67. DAC 寄存器映射 和复位值

偏移	寄存器名称	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**	Res.	Res.	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
	Reset value			0	0	0	0	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**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	**DAC_ DHR12R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**DAC_ DHR12L1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	**DAC_ DHR8R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**DAC_ DHR12R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**DAC_ DHR12L2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	**DAC_ DHR8R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20	**DAC_ DHR12RD**	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					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]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	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**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**DAC_DOR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	**DAC_DOR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

表 67. DAC 寄存器映射（续）和复位值（续）

偏移	寄存器名称	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
0x34	DAC_SR	Res.	Res.	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0																0													

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

16 比较器 (COMP)

16.1 简介

STM32L0x3 器件内置 两个 超低功耗比较器 COMP1 和 COMP2，这两个比较器可分别用作独立器件（I/O 上提供了全部接口），也可与定时器结合使用。

这两个比较器可用于多种功能，包括：

- 在模拟信号的触发下从低功耗模式唤醒
- 模拟信号调理
- 与 DAC 和定时器的 PWM 输出结合使用时，构成逐周期电流控制环路

16.2 COMP 主要特性

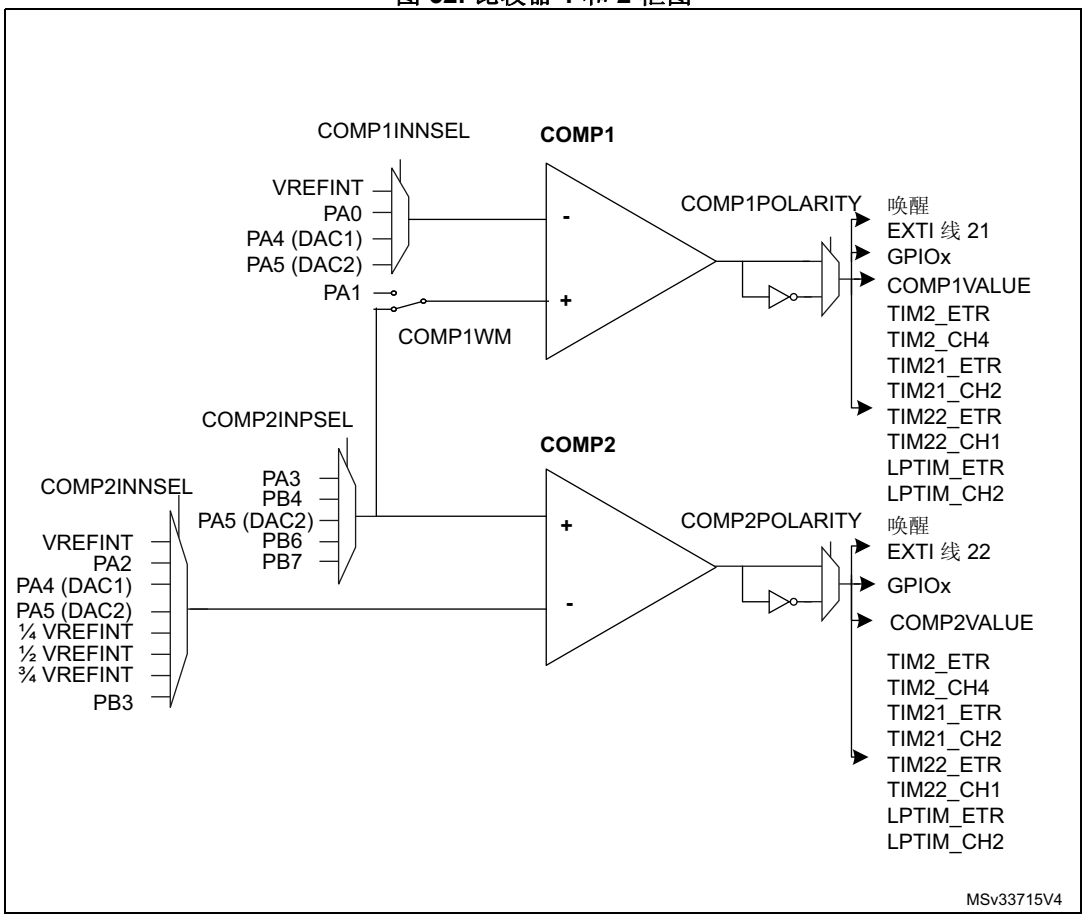
- COMP1 比较器具有超低功耗
- COMP2 比较器具有轨到轨输入以及快速或慢速两种模式
- 每个比较器都具有正输入和可配置的负输入，用于灵活选择电压：
  - I/O 引脚
  - DAC
  - 通过调节器（缓冲分压器）提供的内部参考电压和三个因数分压值（1/4、1/2、3/4）
- 可编程的速度/功耗（仅 COMP2）
- 输出可以重定向到用于触发以下事件的 I/O 或定时器输入：
  - 捕捉事件
- COMP1 和 COMP2 可以组合构成窗口比较器。每个比较器都可产生中断，用于使器件从睡眠模式和停止模式唤醒（通过 EXIT 控制器）

16.3 COMP 功能说明

16.3.1 COMP 框图

比较器的框图如*图 62：比较器 1 和 2 框图*所示。

图 62. 比较器 1 和 2 框图



16.3.2 COMP 引脚和内部信号

用作比较器输入的 I/O 必须在 GPIO 寄存器中配置为模拟模式。

比较器输出可以使用数据手册的“复用功能映射”表中给出的复用功能通道连接到 I/O。

输出也可以在内部重定向到用于以下用途的各种定时器输入：

- 用于时序测量的输入捕捉

可以在内部和外部同时对比较器输出进行重定向。

16.3.3 COMP 复位和时钟

时钟控制器提供的 COMP 时钟与 PCLK（APB 时钟）同步。

RCC 控制器中不单独提供其时钟使能控制位。COMP 和 SYSCFG 共用复位和时钟使能位。

注： ***重要提示：****极性选择逻辑和到端口的输出重定向独立于 PCLK 时钟。因此，即使在停止模式下比较器仍能正常工作。*

16.3.4 比较器锁定机制

这两个比较器可用于过流或热保护等安全用途。对于具有特定功能安全要求的应用，必须保证在发生意外寄存器访问或程序计数器损坏时，不能更改比较器编程。

为此，可以对比较器控制和状态寄存器进行写保护（只读）。

一旦编程完成，COMPxLOCK 位便会设置为 1。这将导致整个 COMPx_CSR 寄存器变成只读，包括 COMPxLOCK 位。

只能通过 MCU 复位来复位写保护。

16.3.5 功率模式

对于给定的应用，可调节 COMP2 功耗与传播延迟以获得最佳平衡。

可以对 COMP2_CSR 寄存器中的 COMP2_SPEED 位进行编程，以提供更高速度/功耗或更低速度/功耗。

16.4 COMP 中断

比较器输出从内部连接到扩展中断和事件控制器。每个比较器都有其各自的 EXTI 线，能够产生中断或事件。该机制还可用于退出低功耗模式。

更多详细信息，请参见中断和事件部分。

16.5 COMP 寄存器

16.5.1 比较器 1 控制和状态寄存器 (COMP1_CSR)

Comparator 1 control and status register

COMP1_CSR 为比较器 1 控制/状态寄存器。此寄存器包含与比较器 1 相关的所有位/标志。

偏移地址：0x18

系统复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP1 LOCK	COMP1 VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1 POLARITY	Res.	Res.	COMP1 LPTIMIN1	Res.	Res.	Res.	COMP1 WM	Res.	Res.	COMP1INN SEL		Res.	Res.	Res.	COMP1 EN
rw			rw				rw			rw	rw				rw

位 31 **COMP1LOCK**：COMP1_CSR 寄存器锁定位 (COMP1_CSR register lock bit)

此位由软件置 1，通过硬件系统复位清零。它将锁定比较器 1 控制寄存器 (COMP1_CSR[31:0]) 的全部内容

0：比较器 1 的 COMP1_CSR[31:0] 可读/写

1：比较器 1 的 COMP1_CSR[31:0] 只读

位 30 **COMP1VALUE**：比较器 1 输出状态位 (Comparator 1 output status bit)

此位为只读。它反映当前比较器 1 输出的状态（受到 COMP1POLARITY 位的影响）。

位 29:16 保留，必须保持复位值

位 15 **COMP1POLARITY**：比较器 1 极性选择位 (Comparator 1 polarity selection bit)

此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它使比较器 1 的极性反相。

0：比较器 1 输出值不反相

1：比较器 1 输出值反相

位 14:13 保留，必须保持复位值

位 12 **COMP1LPTIMIN1**：比较器 1 LPTIM 输入传播位 (Comparator 1 LPTIM input propagation bit)

此位由软件置 1 和清零（假定 COMP1LOCK 未置 1）。它将 COMP1VALUE 发送到 LPTIM 输入 1。

0：对比较器 1 输出进行门控

1：将比较器 1 输出发送到 LPTIM 输入 1

位 11:9 保留，必须保持复位值

位 8 **COMP1WM**：比较器 1 窗口模式选择位 (Comparator 1 window mode selection bit)

此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它选择比较器 1 窗口模式，其中两个比较器的正输入连接在一起。

0：比较器 1 的正输入连接到 PA1。

1：比较器 1 的正输入与比较器 2 的正输入短接（请参见 COMP1_CSR）。

位 7:6 保留，必须保持复位值

位 5:4 **COMP1INNSEL**：比较器 1 负输入连接配置位 (Comparator 1 Input Minus connection configuration bit)

这些位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 1 的负输入

00：VREFINT

01：PA0

10：DAC1/PA4

11：DAC2/PA5

位 3:1 保留，必须保持复位值

位 0 **COMP1EN**：比较器 1 使能位 (Comparator 1 enable bit)

此位由软件置 1 和清零（COMP1LOCK 置 1 时除外）。开关比较器 1

0：关闭比较器 1。

1：开启比较器 1。

16.5.2 比较器 2 控制和状态寄存器 (COMP2_CSR)

Comparator 2 control and status register

COMP2_CSR 为比较器 2 控制/状态寄存器。此寄存器包含与比较器 2 相关的所有位/标志。

偏移地址：0x1C

系统复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2 LOCK	COMP2 VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2 POLARITY	Res.	COMP2 LPTIMIN1	COMP2 LPTIMIN2	Res.	COMP2INPSEL			Res.	COMP2INNSEL			COMP2 SPEED	Res.	Res.	COMP2 EN
rw		rw	rw		rw	rw	rw		rw	rw	rw	rw			rw

位 31 **COMP2LOCK**：COMP2_CSR 寄存器锁定位 (COMP2_CSR register lock bit)

此位由软件置 1，通过硬件系统复位清零。它将锁定比较器 2 控制寄存器 (COMP2_CSR[31:0]) 的全部内容

0：比较器 1 的 COMP2_CSR[31:0] 可读/写

1：比较器 1 的 COMP2_CSR[31:0] 只读

位 30 **COMP2VALUE**：比较器 2 输出状态位 (Comparator 2 output status bit)

此位为只读。它反映当前比较器 2 输出的状态（考虑了 COMP2POLARITY 位的影响）。

位 29:16 保留，必须保持复位值

位 15 **COMP2POLARITY**：比较器 2 极性选择位 (Comparator 2 polarity selection bit)

此位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它使比较器 2 的极性反相。

0：比较器 2 输出值不反相

1：比较器 2 输出值反相

位 14 保留，必须保持复位值

位 13 **COMP2LPTIMIN1**：比较器 2 LPTIM 输入 1 传播位 (Comparator 2 LPTIM input 1 propagation bit)

此位由软件置 1 和清零（假定 COMP2LOCK 未置 1）。它将 COMP2VALUE 发送到 LPTIM 输入 1。

0：对比较器 2 输出进行门控

1：将比较器 2 输出发送到 LPTIM 输入 1

注： COMP2LPTIMIN1 和 COMP2LPTIMIN2 不能同时设置为“1”。

位 12 **COMP2LPTIMIN2**：比较器 2 LPTIM 输入 2 传播位 (Comparator 2 LPTIM input 2 propagation bit)

此位由软件置 1 和清零（假定 COMP2LOCK 未置 1）。它将 COMP2VALUE 发送到 LPTIM 输入 2。

0：对比较器 2 输出进行门控

1：将比较器 2 输出发送到 LPTIM 输入 2

注： COMP2LPTIMIN1 和 COMP2LPTIMIN2 不能同时设置为“1”。

位 11 保留，必须保持复位值

位 10:8 **COMP2INNSEL**：比较器 2 正输入连接配置位 (Comparator 2 Input Plus connection configuration bit)

这些位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 2 的正输入

000：PA3

001：PB4

010：PB5

011：PB6

100：PB7

其它值：保留

位 7 保留，必须保持复位值

位 6:4 **COMP2INNSEL**：比较器 2 负输入连接配置位 (Comparator 2 Input Minus connection configuration bit)

这些位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它们选择哪个输入连接到比较器 2 的负输入。

000：VREFINT

001：PA2

010：DAC/PA4

011：DAC2/PA5

100：1/4 VREFINT

101：1/2 VREFINT

110：3/4 VREFINT

111：PB3

位 3 **COMP2SPEED**：比较器 2 功率模式选择位 (Comparator 2 power mode selection bit)

此位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。它选择比较器 2 功率模式。

0：慢速

1：快速

位 2 保留，必须保持复位值

位 0 **COMP2EN**：比较器 2 使能位 (Comparator 2 enable bit)

此位由软件置 1 和清零（COMP2LOCK 置 1 时除外）。开关比较器 2。

0：关闭比较器 2。

1：开启比较器 2。

16.5.3 COMP 寄存器映射

下表对比较器寄存器进行了汇总。

比较器寄存器与 SYSCFG 外设寄存器基址相同。

表 68. COMP 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	**COMP1_CSR**	COMP1LOCK	COMP1VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP1POLARITY	Res.	Res.	COMP1LPTIMIN1	Res.	Res.	Res.	COMP1WM	Res.	Res.	COMP1INNSEL		Res.	Res.	Res.	COMP1EN
	Reset value	0	0															0			0				0			0	0				0
0x1C	**COMP2_CSR**	COMP2LOCK	COMP2VALUE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP2POLARITY	Res.	COMP2LPTIMIN1	COMP2LPTIMIN2	Res.	COMP2INPSEL			Res.	COMP2INNSEL			COMP2SPEED	Res.	Res.	COMP2EN
	Reset value	0	0															0		0	0		0	0	0		0	0	0	0			0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

17 液晶显示控制器 (LCD)

17.1 简介

LCD 控制器是一款适用于单色无源液晶显示器 (LCD) 的数字控制器/驱动器，最多具有 8 个公用端子和 52[a] 个区段端子，用以驱动 208 (4x52) 或 384 (8x48) 个 LCD 图像元素（像素）。端子的确切数量取决于数据手册中所述的器件引脚。

LCD 由若干区段（像素或完整符号）组成，这些区段均可点亮或熄灭。每个区段都包含一层在两根电极之间对齐的液晶分子。当向液晶施加高于阈值电压的电压时，相应的区段可见。区段电压必须为交流，以避免液晶中出现电泳效应（这将影响显示效果）。之后，必须在区段两端生成波形以避免出现直流 (DC)。

词汇表

LCD：（液晶显示器）无源显示面板，带有直接引向区段的端子。

公用：连接到多个区段（52 个区段）的电气连接端子。

偏置：驱动 LCD 时使用的电压等级。它定义为 1/(用于驱动 LCD 显示器的电压等级数 - 1)。

区段：最小可视单元（一个条或点，用于帮助在 LCD 显示器上创建字符）。

升压电路：对比度控制器电路

占空比：定义为 1 /(给定 LCD 显示器上的公用端子数) 的数字。

帧：写入区段的波形的一个周期。

帧速率：每秒帧数，即，每秒激励 LCD 区段的次数。

a. 关于产品配置，请参见*表 69：实现*。

17.2 LCD 主要特性

- 高度灵活的帧速率控制。
- 支持静态、1/2、1/3、1/4 和 1/8 占空比。
- 支持静态、1/2、1/3 和 1/4 偏置。
- 双缓冲存储器，允许通过应用固件随时更新 LCD_RAM 寄存器中的数据，而不影响所显示数据的完整性。
  - 多达 16 × 32 位寄存器的 LCD 数据 RAM，其中包含像素信息（激活/未激活）。
- 可通过软件选择的 LCD 输出电压（对比度）：V_{LCDmin} 到 V_{LCDmax}。
- 无需外部模拟元件：
  - 内置的升压转换器可生成高于 V_{DD} 的内部 V_{LCD} 电压
  - 可通过软件在外部和内部 V_{LCD} 电压源之间进行选择。如果选择外部电压源，将禁止内部升压电路以降低功耗
  - 内置的电阻网络可生成中间 V_{LCD} 电压
  - 可通过软件配置电阻网络的结构以调整功耗，从而匹配 LCD 面板所需的电容电荷
- 可以通过两种不同的方法调整对比度：
  - 当使用内部升压转换器时，软件可以在 V_{LCDmin} 和 V_{LCDmax} 之间调整 V_{LCD}。
  - 可编程的帧间死区（最多 8 个相位周期）。
- 完全支持低功耗模式：LCD 控制器可在睡眠、低功耗运行、低功耗睡眠和停止模式下进行显示，也可以完全禁止以降低功耗。
- 内置反相功能以降低功耗和 EMI（电磁干扰）。
- 帧起始中断，在更新 LCD 数据 RAM 时同步软件。
- 闪烁功能：
  - 可以设置 1、2、3、4、8 个或所有像素以可配置的频率闪烁
  - 可通过软件将闪烁频率调整到大约 0.5 Hz、1 Hz、2 Hz 或 4 Hz。
- 使用的 LCD 区段和公共引脚应配置为 GPIO 复用功能，未使用的区段和公共引脚可用于通用 I/O 或其他外设复用功能。
- V_{LCD} 轨去耦功能

注：当 LCD 依赖于内部升压转换器时，VLCD 引脚应通过一个电容连接到 V_{SS}。电容典型值为 1 µF（有关详细信息，请参见产品数据手册中的 C_{EXT} 值）。

注：如果不使用 LCD 外设，则 VLCD 引脚应连接到 V_{DDA}。

17.3 LCD 实现

表 69. 实现

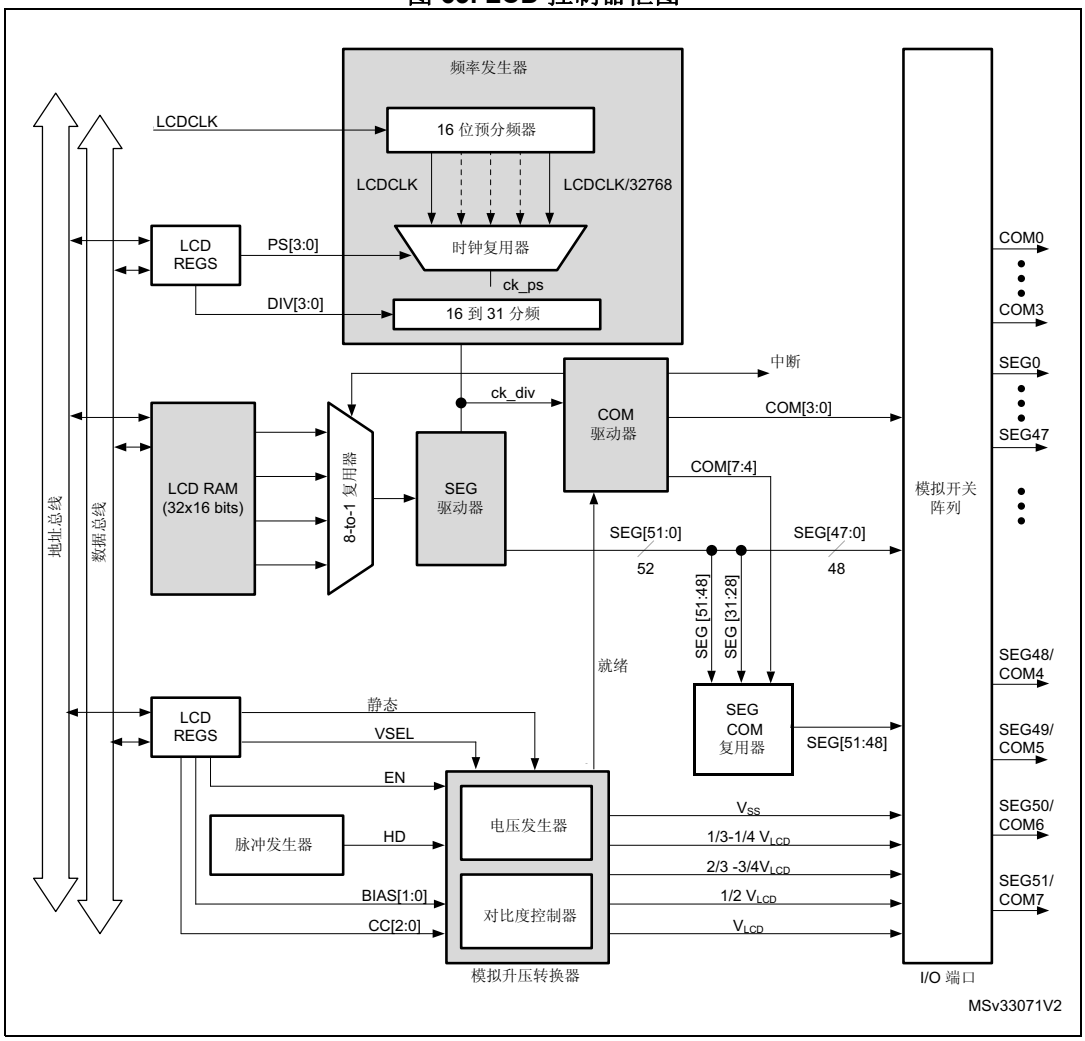
产品	区段端子
第 5 类器件	52 个区段
第 3 类器件	32 个区段

17.4 LCD 功能说明

17.4.1 概述

LCD 控制器具有五个主要模块（请参见*图 63*）：

图 63. LCD 控制器框图



注： *LCDCLK 与 RTCCLK 相同。请参考本手册的 RCC 部分中的 RTC/LCD 时钟说明。*

凭借频率发生器，可以通过 LCD 输入时钟频率（LCDCLK，可从 32 kHz 直至 1 MHz）获得各种 LCD 帧速率。

可以使用 3 种不同的时钟源来提供 LCD 时钟 (LCDCLK/RTCCLK)：

- 32 kHz 低速外部 RC (LSE)
- 32 kHz 低速内部 RC (LSI)
- 2、4、8 或 16 分频的高速外部 RC (HSE) 以获得 1 MHz 时钟

17.4.2 频率发生器

此时钟源必须稳定以获得精确的 LCD时序，并因此最大限度地减少 LCD 区段间的 DC 电压偏移。输入时钟 LCDCLK 可被从 1 到 2^{15}x 31 的任意值分频（请参见*第 374 页上的第 17.7.2 节：LCD 帧控制寄存器 (LCD_FCR)*）。频率发生器包含一个预分频器（16 位纹波计数器）和一个 16 到 31 时钟分频器。LCD_FCR 寄存器中的 PS[3:0] 位选择被 $2^{PS[3:0]}$ 分频的 LCDCLK。如果需要更精细的分辨率，LCD_FCR 寄存器中的 DIV[3:0] 位可用于进一步对时钟进行 16 到 31 分频。通过这种方式可以粗调频率，然后通过计数器线性调整时钟以进行微调。频率发生器模块的输出是 f_{ck_div}，它构成整个 LCD 控制器的时基。ck_div 频率与 LCD 相位频率相等，而不等于帧频率（它们仅在静态占空比的情况下相等）。通过将 f_{ck_div} 除以激活的公用端子数（或者将该频率与占空比相乘），可获得帧频率 (f_{frame})。因此，频率发生器的输入时钟频率 (f_{LCDCLK}) 与其输出时钟频率 f_{ck_div} 之间的关系为：

$$f_{ckdiv} = \frac{f_{LCDCLK}}{2^{PS} \times \langle 16 + DIV \rangle}$$

$$f_{frame} = f_{ckdiv} \times duty$$

这使得频率发生器非常灵活。*表 70* 中给出了帧速率计算示例。

表 70. 帧速率计算示例

LCDCLK	PS[3:0]	DIV[3:0]	比值	占空比	f_{frame}
32.768 kHz	3	1	136	1/8	30.12 Hz
32.768 kHz	4	1	272	1/4	30.12 Hz
32.768 kHz	4	6	352	1/3	31.03 Hz
32.768 kHz	5	1	544	1/2	30.12 Hz
32.768 kHz	6	1	1088	静态	30.12 Hz
32.768 kHz	1	4	40	1/8	102.40 Hz
32.768 kHz	2	4	80	1/4	102.40 Hz
32.768 kHz	2	11	108	1/3	101.14 Hz
32.768 kHz	3	4	160	1/2	102.40 Hz
32.768 kHz	4	4	320	静态	102.40 Hz
1.00 MHz	6	3	1216	1/8	102.80 Hz
1.00 MHz	7	3	2432	1/4	102.80 Hz
1.00 MHz	7	10	3328	1/3	100.16 Hz
1.00 MHz	8	3	4864	1/2	102.80 Hz
1.00 MHz	9	3	9728	静态	102.80 Hz

选择的帧频率必须在大约 30 Hz 到 100 Hz 的范围内，并在功耗和可接受的刷新速率之间进行平衡。此外，可通过一个专用闪烁预分频器选择闪烁频率。此频率定义为：

$f_{BLINK} = f_{ck_div}/2^{(BLINKF + 3)}$,

其中，BLINKF[2:0] = 0、1、2、... 、7

可实现的闪烁频率为 0.5 Hz、1 Hz、2 Hz 或 4 Hz。

17.4.3 公用驱动器

公用信号由公用驱动器模块产生（请参见*图 63*）。

COM 信号偏置

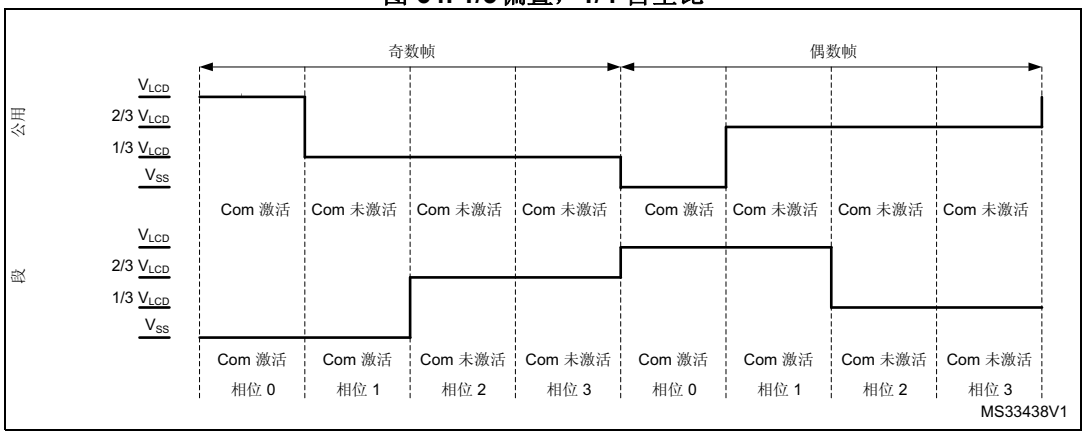
各个 COM 信号的波形相同，但相位不同。这些信号仅在周期的相应相位才具有最大振幅 V_{LCD} 或 V_{SS}，而在其它相位的信号幅值为：

- 1/4 V_{LCD} 或 3/4 V_{LCD}（1/4 偏置时）
- 1/3 V_{LCD} 或 2/3 V_{LCD}（1/3 偏置时）
- 以及 1/2 V_{LCD}（1/2 偏置时）。

可以通过 LCD_CR 寄存器中的 BIAS 位在 1/2、1/3 和 1/4 偏置模式之间进行选择。

当某个像素对应的公用线和区段线在相同相位期间激活时，该像素激活，这意味着在此相位期间公用线和区段线之间的电压差最大。对公用信号进行反相以减少 EMI。如*图 64* 所示，通过反相，每个奇数周期结束时的平均电压为 1/2 V_{LCD}。

图 64. 1/3偏置，1/4 占空比



在 1/2 偏置 (BIAS = 01) 的情况下，VLCD 引脚在奇数帧和偶数帧的节点 b 上产生的中间电压等于 1/2 V_{LCD}（请参见*图 67*）。

COM 信号占空比

根据 LCD_CR 寄存器中的 DUTY[2:0] 位，使用以下占空比产生 COM 信号：静态占空比（见*图 66*）、1/2 占空比（见*图 67*）、1/3 占空比（见*图 68*）、1/4 占空比（见*图 69*）或 1/8 占空比（见*图 70*）。

奇数帧的相位 *n* 期间 COM[*n*] *n*[0 到 7] 激活，即 COM 引脚被驱动到 V_{LCD}。

偶数帧的相位 *n* 期间 COM 引脚被驱动到 V_{SS}。

在 1/3 或 1/4 偏置的情况下：

- 在 n 以外的相位期间 COM[*n*] 未激活，因此 COM 引脚被驱动到 1/3 (1/4) V_{LCD}（奇数帧期间）和 2/3 (3/4) V_{LCD}（偶数帧期间）

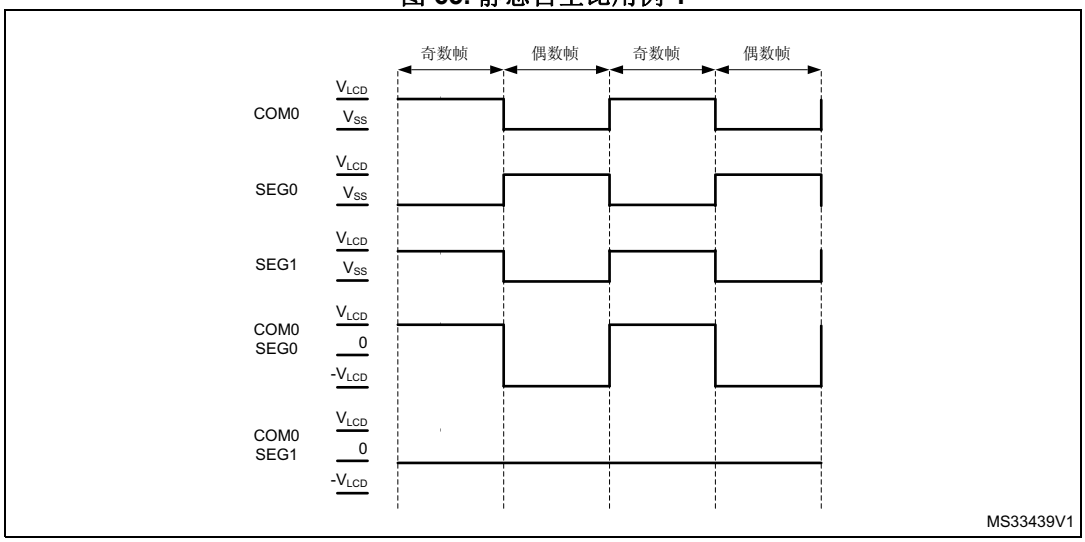
在 1/2 偏置的情况下：

- 如果在 n 以外的相位期间 COM[*n*] 未激活，COM 引脚始终驱动到 1/2 V_{LCD}（奇数帧和偶数帧）。

当选择静态占空比时，区段线不多路复用，这意味着每个区段输出对应于一个像素。通过这种方式，最多只能驱动 51 个像素。COM[7:1] 未使用，并被驱动到 V_{SS}，与此同时，COM[0] 始终激活。

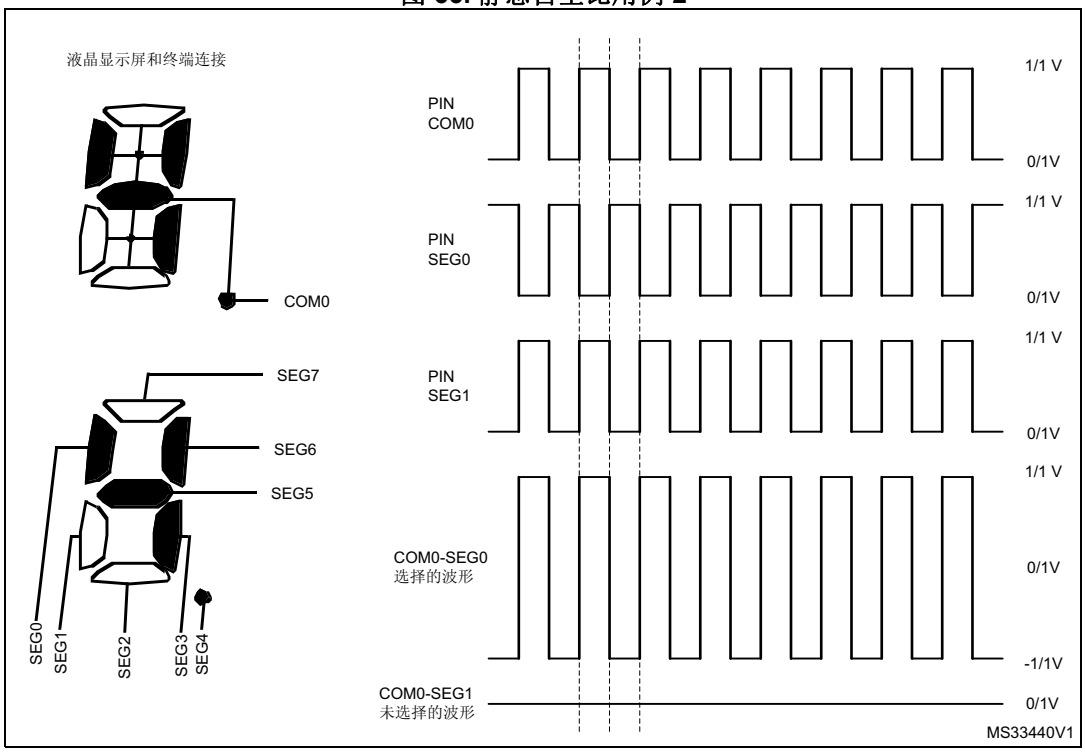
当 LCD_CR 寄存器中的 LCDEN 位复位时，所有公用线均下拉到 V_{SS}，并且 LCD_RS 寄存器中的 ENS 标志变为 0。静态占空比意味着 COM[0] 始终激活，并且只有两个电压电平用于区段和公用线：V_{LCD} 和 V_{SS}。如果像素对应的 SEG 线的电压与 COM 线的电压相反，则像素激活；如果电压相等，则像素未激活。通过这种方式，LCD 可获得最大对比度（请参见*图 65*、*图 66*）。在*图 65* 中，像素 0 激活而像素 1 未激活。

图 65. 静态占空比用例 1



每个帧中只有一个相位，因此 f_{frame} 等于 f_{LCD}。如果选择 1/4 占空比，一个帧中有四个相位，其中，COM[0] 在相位 0 期间激活，COM[1] 在相位 1 期间激活，COM[2] 在相位 2 期间激活，COM[3] 在相位 3 期间激活。

图 66. 静态占空比用例 2

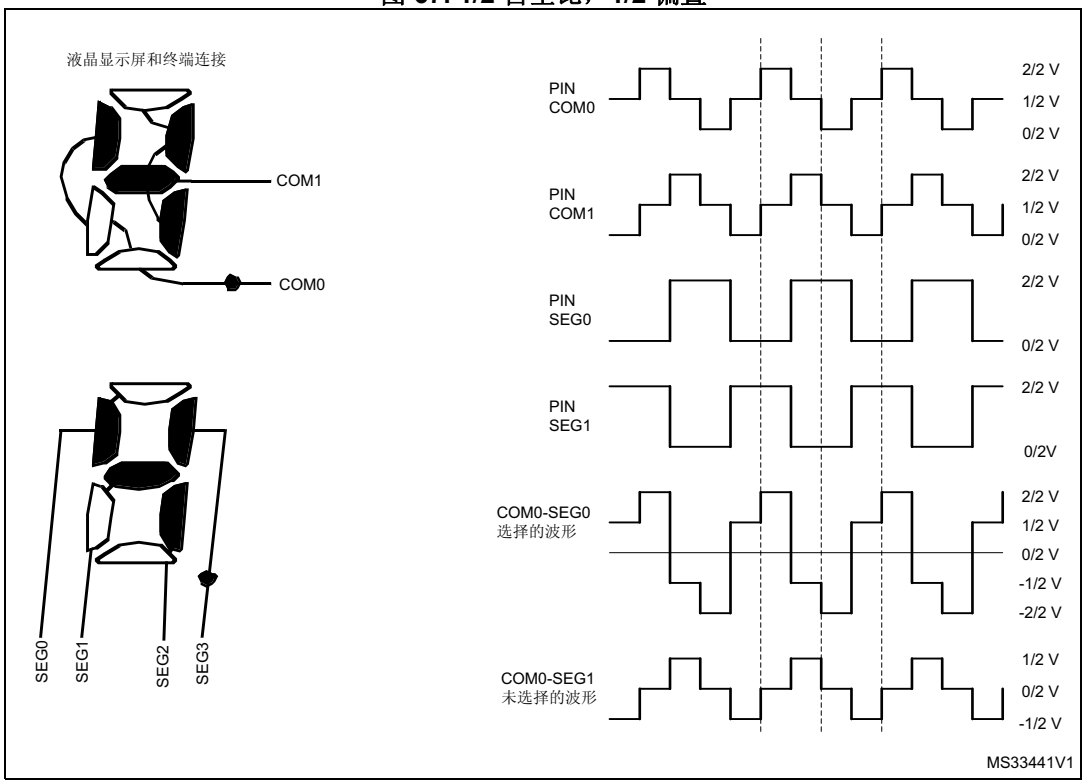


在此模式下，区段端子多路复用并且每个端子控制四个像素。仅当像素对应的 SEG 和 COM 线在相同相位期间激活时，该像素才激活。在 1/4 占空比的情况下，要禁用连接到 COM[0] 的像素 0，当 COM[0] 激活时，SEG[0] 需要在相位 0 期间处于未激活状态。要激活连接到 COM[1] 的像素 0，当 COM[1] 激活时，SEG[0] 需要在相位 1 期间处于激活状态（请参见*图 69*）。要激活连接到 COM[0] 的像素 0 到 51，当 COM[0] 激活时，SEG[0:51] 需要在相位 0 期间处于激活状态。这些注意事项同样适用于其它像素。

8 选 1 多路复用器

当 COM[0] 激活时，公用驱动器模块也将驱动*图 63* 中所示的 8 选 1 多路复用器，以便选择前两个 RAM 寄存器存储单元的内容。当 *COM[7]* 激活时，8 选 1 多路复用器的输出是最后两个 RAM 存储单元的内容。

图 67. 1/2 占空比，1/2 偏置



17.4.4 区段驱动器

区段驱动器模块根据像素数据（来自各相位中由公用驱动器模块驱动的 8 选 1 多路复用器）来控制 SEG 线。

在 1/4 或 1/8 占空比的情况下

当 COM[0] 激活时，与连接到 COM[0]（前两个 LCD_RAM 存储单元的内容）的像素相关的像素信息（激活 / 未激活）通过 8 选 1 多路复用器。

在奇数帧的相位 0 期间，SEG[*n*] 引脚 *n* [0 到 51] 被驱动到 V_{SS}（指示当 COM[0] 激活时像素 *n* 激活）。

在偶数帧的相位 0 期间，SEG[*n*] 引脚被驱动到 V_{LCD}。如果像素 *n* 未激活，则 SEG[*n*] 引脚被驱动到 2/3 (2/4) V_{LCD}（奇数帧期间）或 1/3 (2/4) V_{LCD}（偶数帧期间）（V_{LCD} 焊盘中的当前转换）（请参见*图 64*）。

在 1/2 偏置的情况下，如果像素未激活，SEG[*n*] 引脚被驱动到 V_{LCD}（奇数帧期间）和 V_{SS}（偶数帧期间）。

当禁止 LCD 控制器（LCD_CR 寄存器中的 LCDEN 位清零）时，SEG 线下拉到 V_{SS}。

图 68. 1/3 占空比，1/3 偏置

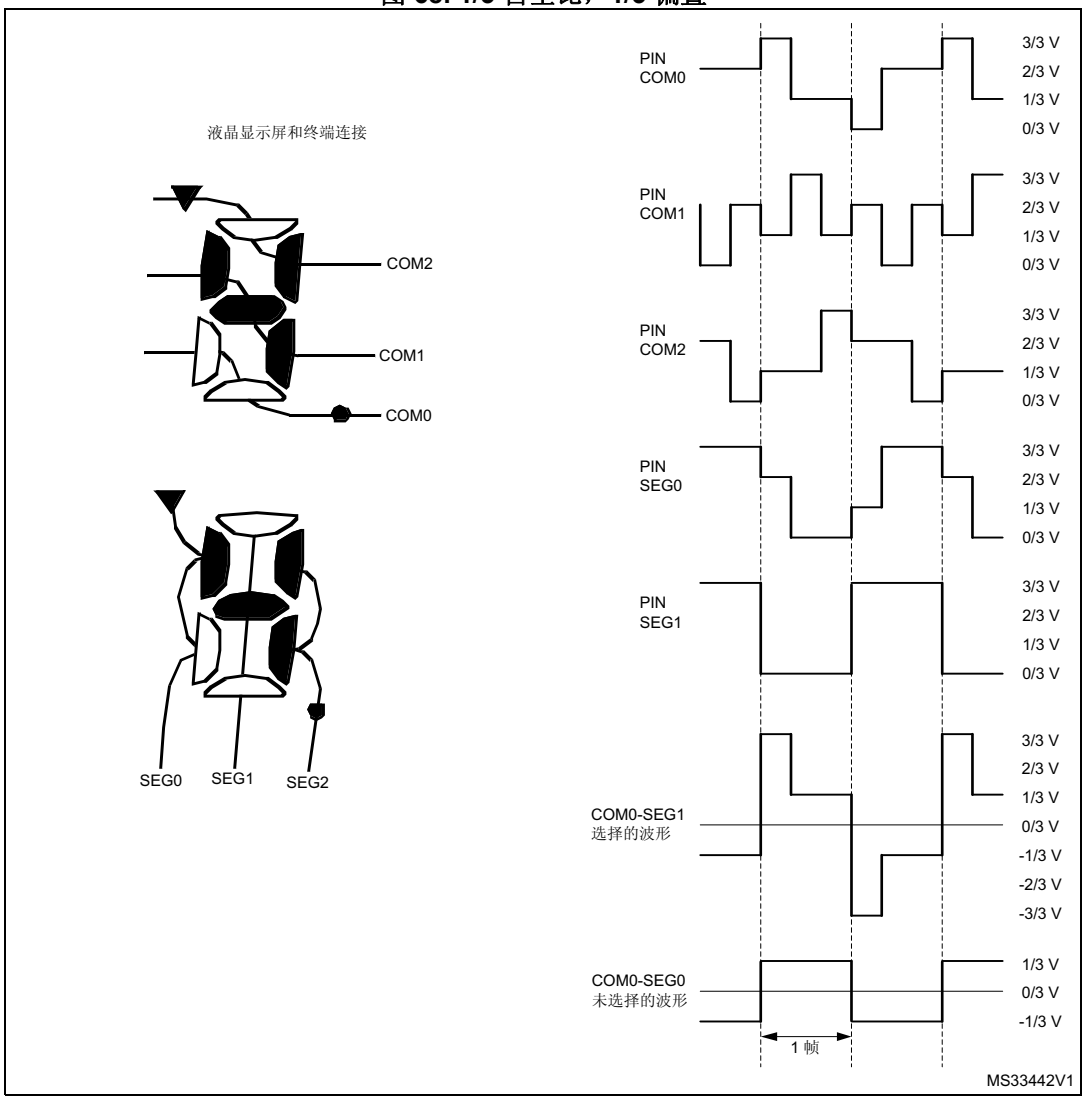


图 69. 1/4 占空比，1/3 偏置

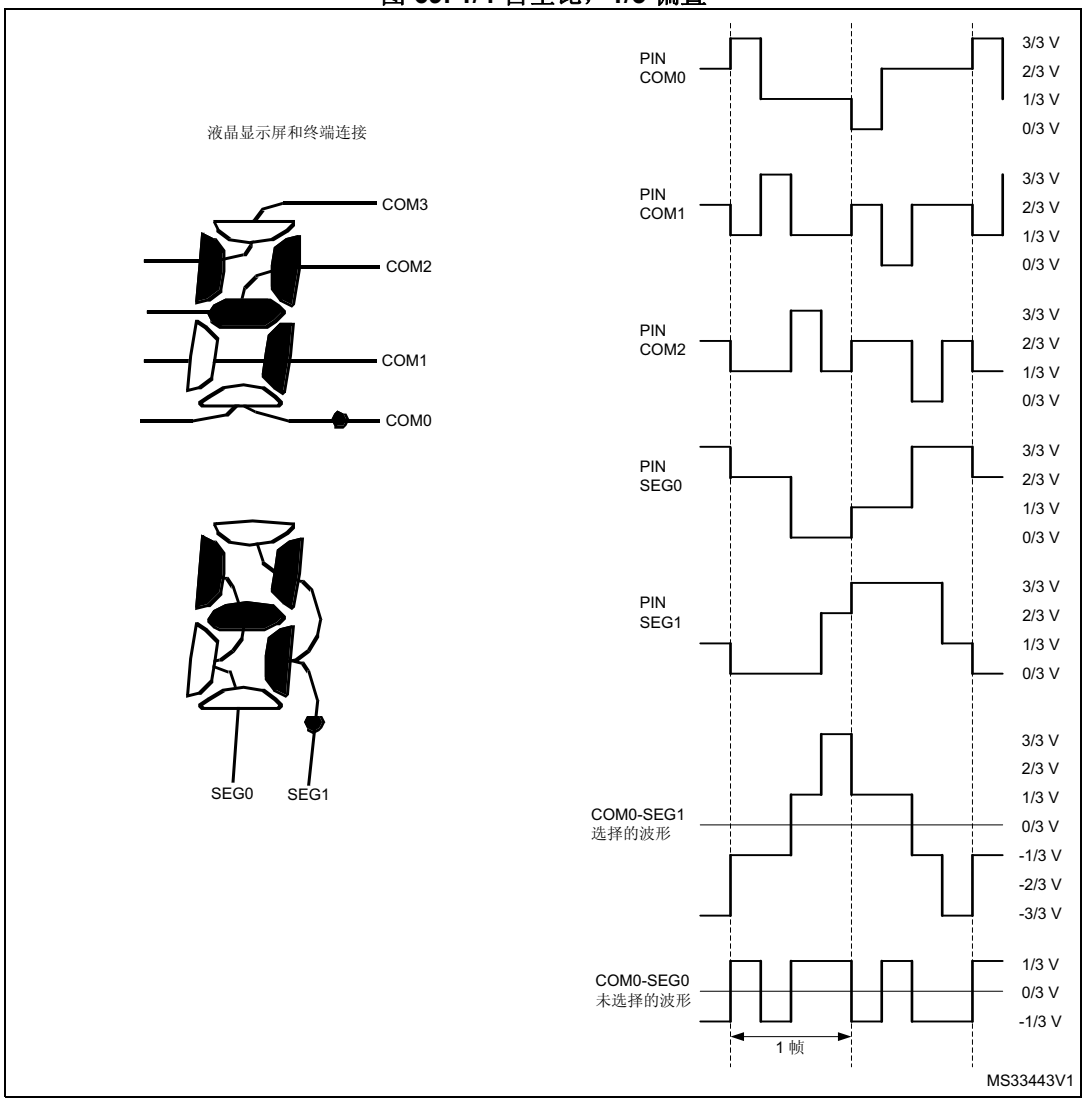
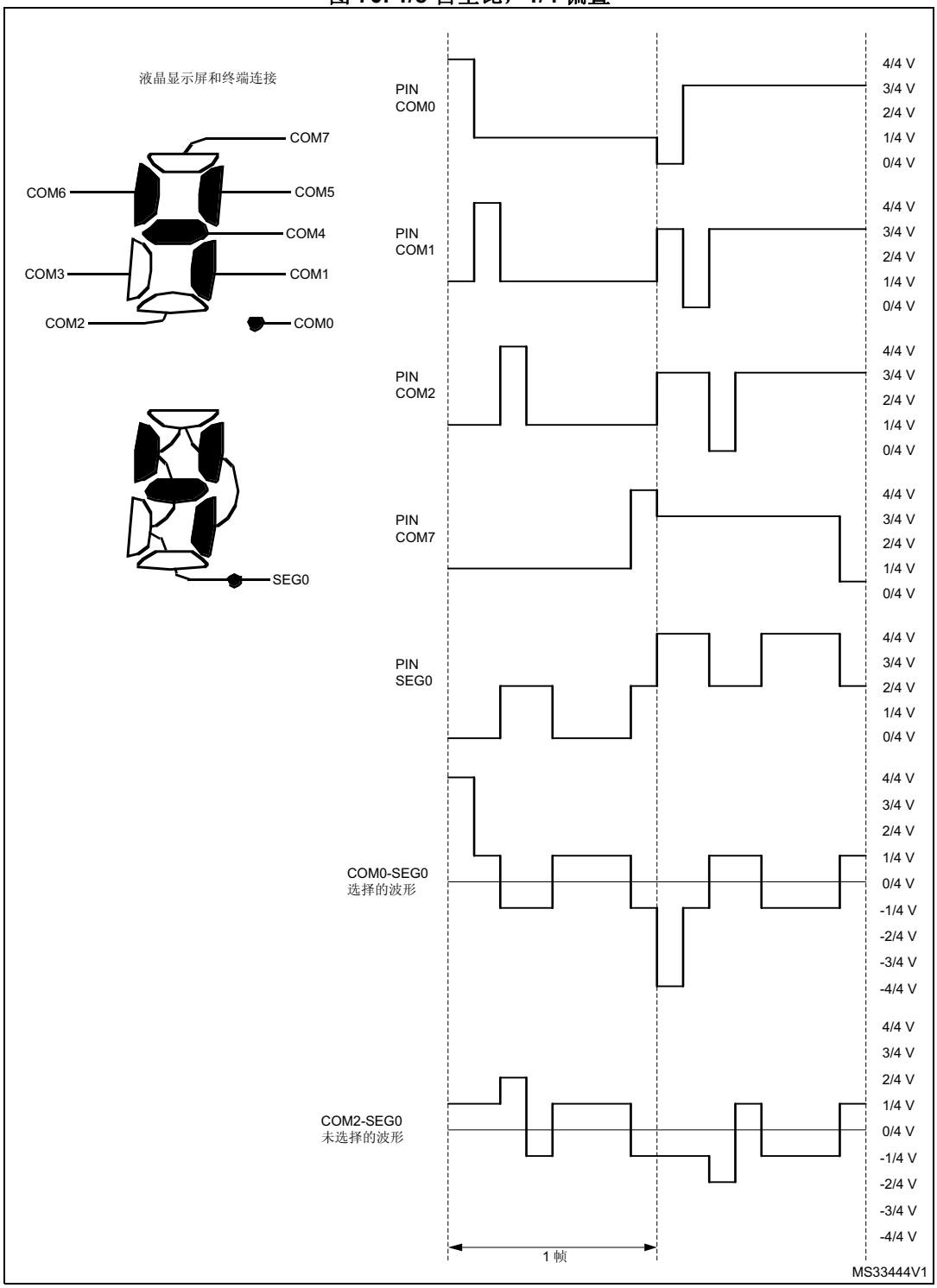


图 70. 1/8 占空比，1/4 偏置



闪烁

区段驱动器还实现了一个可编程的闪烁功能，允许一些像素以特定频率连续地接通。可通过 LCD_FCR 寄存器中的 BLINK[1:0] 位配置闪烁模式，从而使 1、2、4、8 或所有像素能够同时闪烁（请参见*第 17.7.2 节：LCD 帧控制寄存器 (LCD_FCR)*）。可使用 LCD_FCR 寄存器中 BLINKF[2:0] 位从八个不同值中选择闪烁频率。

表 71 给出了不同闪烁频率的示例（以 ck_div 频率的函数形式）。

表 71. 闪烁频率

BLINKF[2:0] 位			ck_div 频率（LCDCLK 频率为 32.768 kHz）			
			32 Hz	64 Hz	128 Hz	256 Hz
0	0	0	4.0 Hz	N/A	N/A	N/A
0	0	1	2.0 Hz	4.0 Hz	N/A	N/A
0	1	0	1.0 Hz	2.0 Hz	4.0 Hz	N/A
0	1	1	0.5 Hz	1.0 Hz	2.0 Hz	4.0 Hz
1	0	0	0.25 Hz	0.5 Hz	1.0 Hz	2.0 Hz
1	0	1	N/A	0.25 Hz	0.5 Hz	1.0 Hz
1	1	0	N/A	N/A	0.25 Hz	0.5 Hz
1	1	1	N/A	N/A	N/A	0.25 Hz

17.4.5 电压发生器和对比度控制

LCD 电源

LCD 电源可能来自内部升压转换器，或来自施加在 VLCD 引脚上的外部电压。可以使用 LCD_CR 寄存器中的 VSEL 位选择内部或外部电压源。如果选择外部电压源，将禁止内部升压电路（升压转换器）以降低功耗。

当选择升压转换器作为 V_{LCD} 源时，可以通过 LCD_FCR（请参见*第 17.7.2 节*）寄存器中的位 CC[2:0]（对比度控制）在多个值（从 V_{LCDmin} 到 V_{LCDmax}）中选择 V_{LCD} 值。V_{LCD} 的新值在每次新帧开始时生效。

当选择外部电源作为 V_{LCD} 源时，必须在 V_{LCDmin} 到 V_{LCDmax} 的范围内选择 V_{LCD} 电压（请参见数据手册）。然后可以通过编程帧间死区来控制对比度（请参见*第 365 页的“死区”*）。

LCD 中间电压

LCD 中间电压通过内部电阻分压器网络生成，如*图 71* 所示。

LCD 电压发生器发出介于 V_{SS} 和 V_{LCD} 之间的中间电压：

- 1/3 V_{LCD} 和 2/3 V_{LCD}（1/3 偏置时）
- 1/4 V_{LCD}、2/4 V_{LCD} 和 3/4 V_{LCD}（1/4 偏置时）
- 仅 1/2 V_{LCD}（1/2 偏置时）。

LCD 驱动器选择

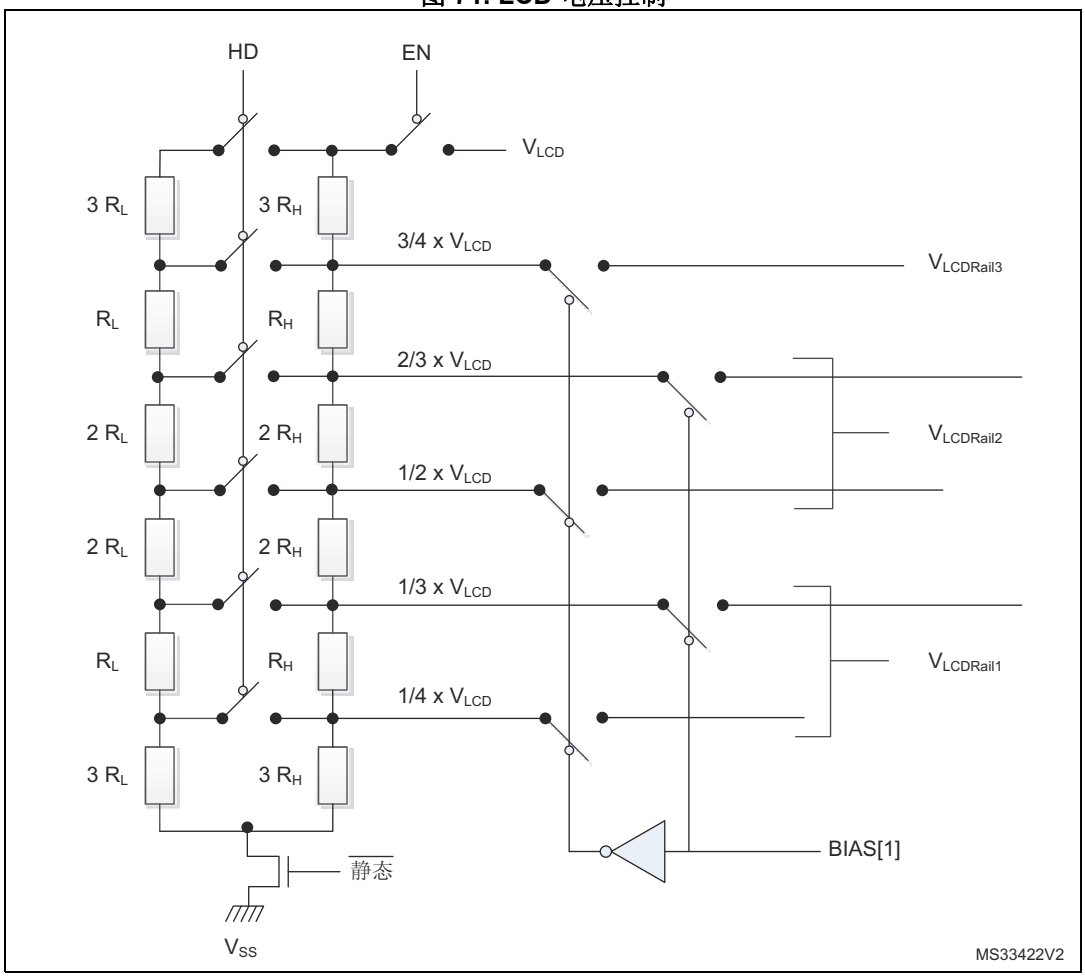
两个电阻网络（一个具有低值电阻 (R_L)，一个具有高值电阻 (R_H)）分别用于在转换过程中增加电流和在静态状态下降低功耗。

EN 开关遵循下述规则（请参见*图 71*）：

- 如果 LCD_CR 寄存器中的 LCDEN 位置 1，EN 开关闭合。
- 当 LCD_CR 寄存器中的 LCDEN 位清零时，EN 开关在偶数帧结束时打开，以避免中间电压不为 0（考虑整个奇数帧加偶数帧）。

当公用线和区段线的电平变化时，LCD_FCR 寄存器中的 PON[2:0]（脉冲打开持续时间）位配置通过 HD（高驱动）接通 R_L 的时间（请参见*图 71*）。较短的驱动时间有助于降低功耗，但内部电阻较高的显示器可能需要更长的驱动时间才能达到令人满意的对比度。

图 71. LCD 电压控制



1. R_{LN} 和 R_{HN} 分别是低值电阻网络和高值电阻网络。

可通过 LCD_FCR 配置寄存器中的 HD 位随时接通 R_{LN} 分频器（请参见*第 17.7.2 节*）。

HD 开关遵循下述规则：

- 如果 LCD_FCR 寄存器中的 HD 位和 PON[2:0] 位复位，则 HD 开关打开。
- 如果 LCD_FCR 寄存器中的 HD 位复位，并且 LCD_FCR 中的 PON[2:0] 位不为 00，则在 PON[2:0] 位中定义的脉冲数期间 HD 开关闭合。
- 如果 LCD_FCR 寄存器中的 HD 位为 1，则 HD 开关常闭。

LCDEN 位激活后，LCD_SR 寄存器中的 RDY 位置 1，以指示电压电平稳定并且 LCD 控制器可以开始工作。

外部去耦

具有 V_{LCD} 轨去耦功能（请参见器件数据手册）的器件允许在 LCD_VLCD1、LCD_VLCD2 和 LCD_VLCD3 上提供的 VLCD 中间电压轨上添加去耦电容，用于稳定用途（请参见*图 71*）。当施加到像素的电压为交流时，可能会观察到尖峰。在这种情况下，这些去耦电容将有助于获得稳定电压，从而提高对比度。

出于功耗考虑，此功能特别有用，因为它允许在 LCD_FCR 寄存器中选择较低的 PON[2:0] 值。

要将*表 72* 中所述的 V_{LCD} 轨连接到专用的 GPIO，配置 SYSCFG_CFGR2 寄存器的 LCD_CAPA[4:0] 位（请参见*第 10.2.2 节：SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)*）。

表 72. 将 VLCD 轨连接到 GPIO 引脚

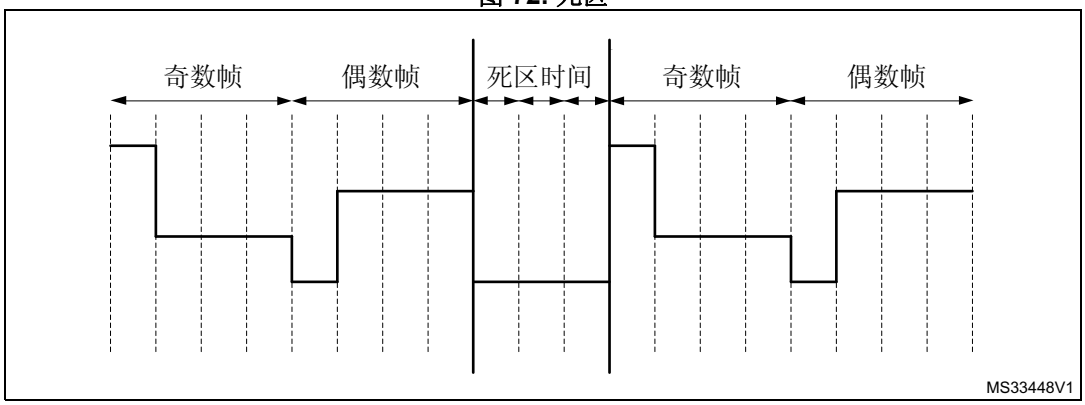
	偏置			引脚（通过 LCD_CAPA[4:0] 位选择）
	1/2	1/3	1/4	
VLCD 轨 1	未使用	未使用	3/4 V_{LCD}	PB0 或 PE12
VLCD 轨 2	1/2 V_{LCD}	2/3 V_{LCD}	1/2 V_{LCD}	PB2
VLCD 轨 3	未使用	1/3 V_{LCD}	1/4 V_{LCD}	PB12 或 PE11

为了确保有效，必须根据 LCD 玻璃和 PCB 电容调整这些去耦电容的值。作为参考，用户可以将去耦电容值设置为 LCD 电容值的 10 倍左右。

死区

除了使用 CC[2:0] 位外，还可以通过编程帧间死区来控制对比度。在死区内，COM 和 SEG 值置于 V_{SS}。LCD_FCR 寄存器中的 DEAD[2:0] 位可用于编程一段最长为八个相位周期的时间。此死区可降低对比度，而无需修改帧速率。

图 72. 死区



17.4.6 双缓冲存储器

使用此双缓冲存储器，LCD 控制器可保证所显示信息连贯，而无需使用中断来控制 LCD_RAM 修改。

应用软件可以通过 APB 接口访问第一缓冲级 (LCD_RAM)。一旦它修改了 LCD_RAM，便会将 LCD_SR 寄存器中的 UDR 标志置 1。该 UDR 标志（更新显示请求）会请求将更新的信息移到第二缓冲级 (LCD_DISPLAY)。

该操作与帧同步进行（在下一帧开始时），直到更新完成，LCD_RAM 处于写保护状态并且 UDR 标志一直保持高电平状态。一旦更新完成，另一个标志（UDD - 更新显示完成）便会置 1 并产生一个中断（如果 LCD_FCR 寄存器中的 UDDIE 位置 1）。

在最坏的情况下，更新 LCD_DISPLAY 所花费的时间为一个奇数帧和一个偶数帧。

使能显示器之前 (LCDEN = 1)，不会发生更新（UDR = 1 且 UDD = 0）

17.4.7 COM 和 SEG 多路复用

输出引脚与占空比模式

输出引脚最多包含：

- SEG[51:0]
- COM[3:0]

根据占空比配置的不同，COM 和 SEG 输出引脚会具有不同的功能：

- 在静态、1/2、1/3、1/4 占空比模式下，有多达 52 个 SEG 引脚，并分别有 1、2、3 和 4 个 COM 引脚
- 在 1/8 占空比模式下 (DUTY[2:0] = 100)，第 5 类器件和第 3 类器件分别在 SEG[51:48] 和 SEG[31:28] 引脚上提供 COM[7:4] 输出。这样可减少可用区段数。

小型封装的重映射功能

此外，可以通过将 LCD_CR 寄存器中 MUX_SEG 位置 1 重新映射 4 个区段。使用外部引脚较少的小型器件时，该功能特别有用。当 MUX_SEG 位置 1 时，输出引脚 SEG [51:48] 与 SEG [31:28] 具有相同功能。

仅第 5 类器件提供该功能。

COM 和 SEG 功能与占空比和重映射的关系汇总

表 73 给出了多路复用 COM 和 SEG 功能的所有可能方式。*图 73* 给出了与外部引脚的信号连接示例。

表 73. 重映射功能(1)

配置位		QFP64/ BGA64 (2)	BGA100/ LQFP100	输出引脚	功能
占空比	MUX_SEG				
1/8	0/1	-	48x8	SEG[51:48]/SEG[31:28]/COM[7:4]	COM[7:4]
				COM[3:0]	COM[3:0]
				SEG[47:0]	SEG[47:0]
	0/1	28x8	-	SEG[51:48]/SEG[31:28]/COM[7:4]	COM[7:4]
				COM[3:0]	COM[3:0]
				SEG[27:0]	SEG[27:0]
1/4	0	-	52x4	COM[3:0]	COM[3:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[51:48]
				SEG[47:0]	SEG[47:0]
	1		48x4	COM[3:0]	COM[3:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[47:32]	SEG[47:32]
				SEG[31:28]	未使用
				SEG[27:0]	SEG[27:0]
	0	28x4	-	COM[3:0]	COM[3:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
				SEG[27:0]	SEG[27:0]
	1	32x4		COM[3:0]	COM[3:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[27:0]	SEG[27:0]

表 73. 重映射功能[1]（续）

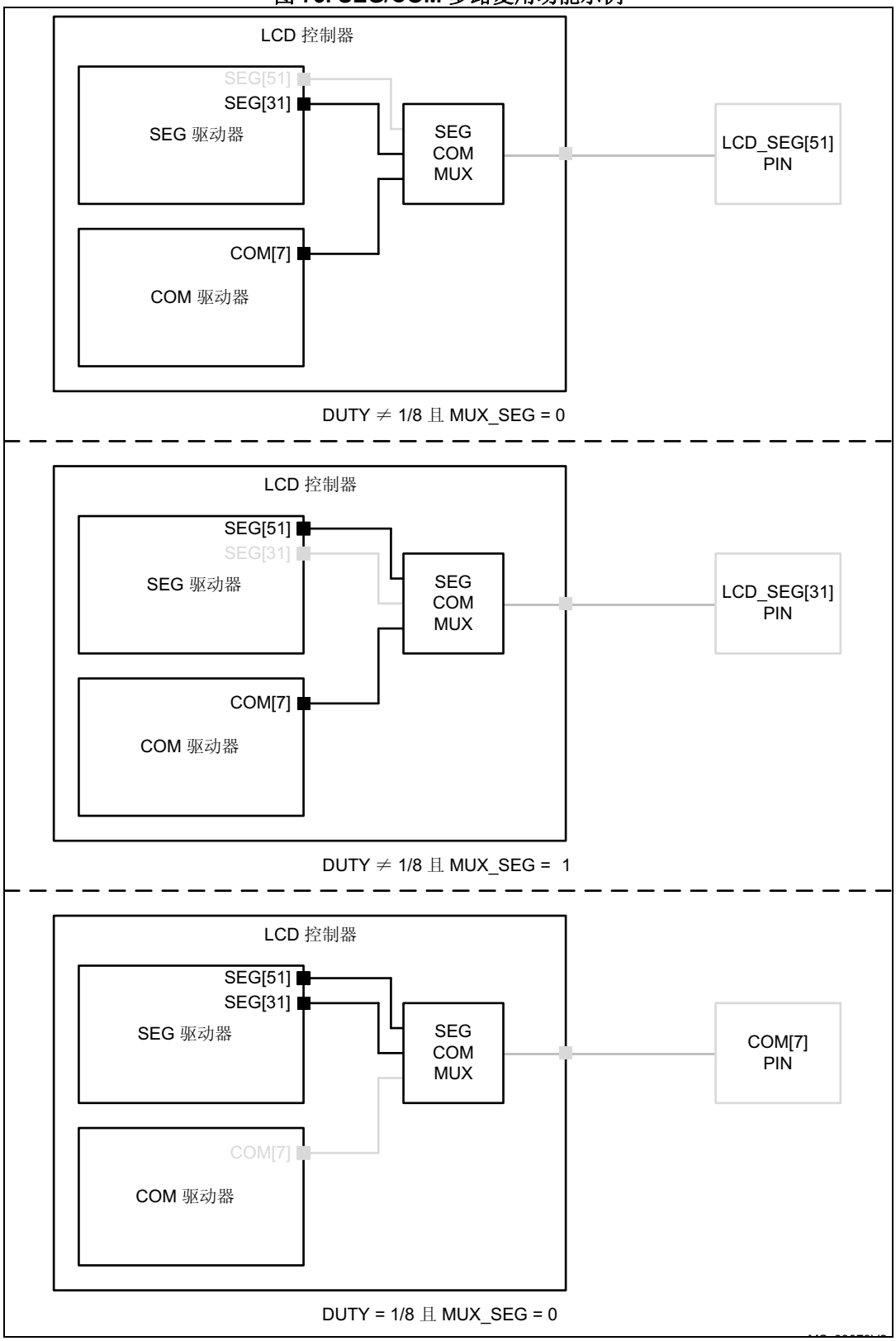
配置位		QFP64/ BGA64 [2]	BGA100/ LQFP100	输出引脚	功能
占空比	MUX_SEG				
1/3	0	-	52x3	COM3	未使用
				COM[2:0]	COM[2:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[51:48]
				SEG[47:0]	SEG[47:0]
	1		48x3	COM3	未使用
				COM[2:0]	COM[2:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[47:32]	SEG[47:32]
				SEG[31:28]	未使用
				SEG[27:0]	SEG[27:0]
	0	28x3	-	COM3	未使用
				COM[2:0]	COM[2:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
				SEG[31:0]	SEG[31:0]
	1	32x3		COM3	未使用
				COM[2:0]	COM[2:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[27:0]	SEG[27:0]
1/2	0	-	52x2	COM[3:2]	未使用
				COM[1:0]	COM[1:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[51:48]
				SEG[47:0]	SEG[47:0]
	1		48x2	COM[3:2]	未使用
				COM[1:0]	COM[1:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[47:32]	SEG[47:32]
				SEG[31:28]	未使用
				SEG[27:0]	SEG[27:0]

表 73. 重映射功能[1]（续）

配置位		QFP64/ BGA64 [2]	BGA100/ LQFP100	输出引脚	功能
占空比	MUX_SEG				
1/2	0	28x2	-	COM[3:2]	未使用
				COM[1:0]	COM[1:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
				SEG[27:0]	SEG[27:0]
	1	32x2		COM[3:2]	未使用
				COM[1:0]	COM[1:0]
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[27:0]	SEG[27:0]
STATIC	0	-	52x1	COM[3:1]	未使用
				COM0	COM0
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[51:48]
				SEG[47:0]	SEG[47:0]
	1		48x1	COM[3:1]	未使用
				COM0	COM0
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[47:32]	SEG[47:32]
				SEG[31:28]	未使用
				SEG[27:0]	SEG[27:0]
	0	28x1	-	COM[3:1]	未使用
				COM0	COM0
				SEG[51:48]/SEG[31:28]/COM[7:4]	未使用
				SEG[27:0]	SEG[27:0]
	1	32x1		COM[3:1]	未使用
				COM0	COM0
				SEG[51:48]/SEG[31:28]/COM[7:4]	SEG[31:28]
				SEG[27:0]	SEG[27:0]

1. 此表仅适用于 5 类器件。
2. BGA64 上不提供 SEG21。

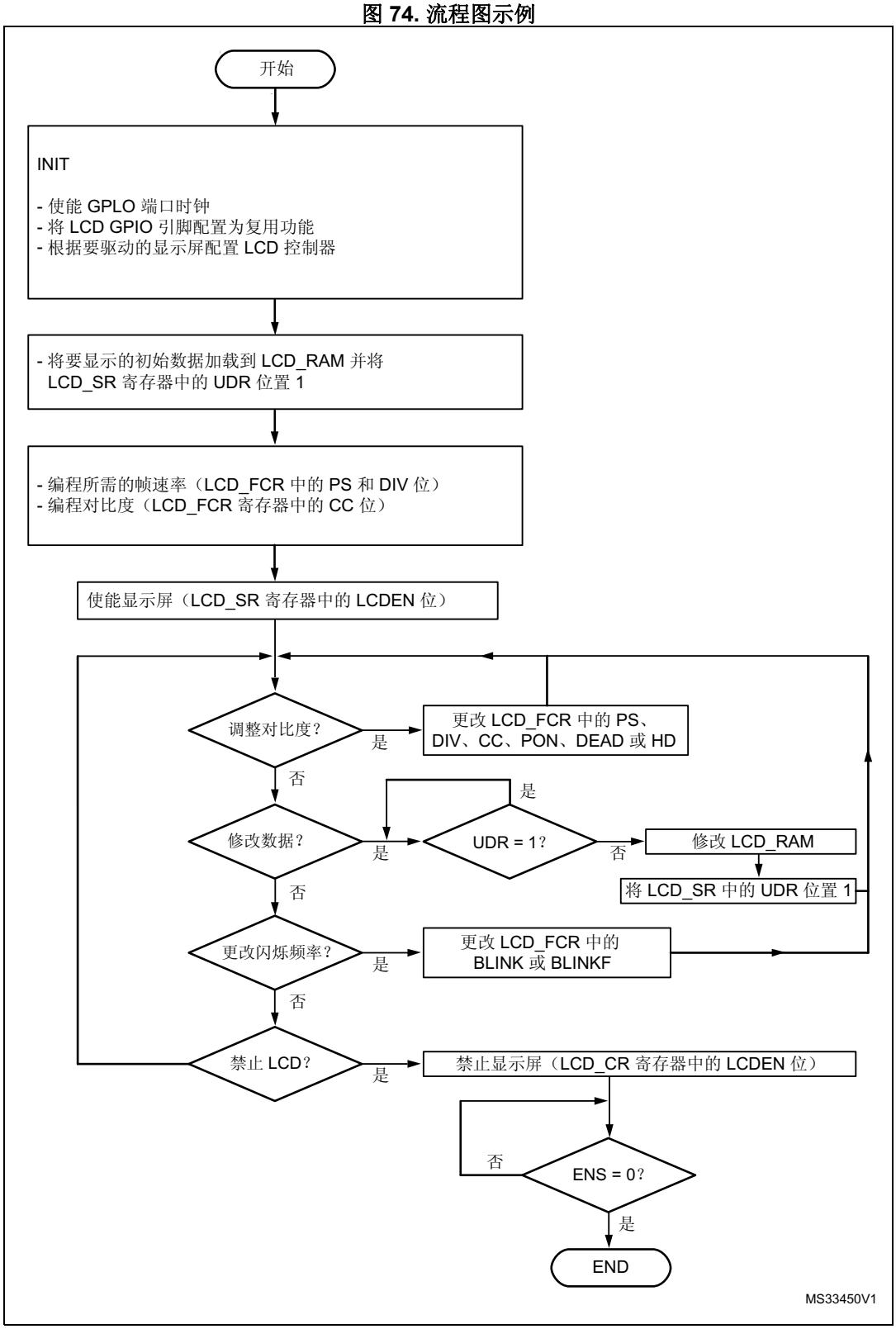
图 73. SEG/COM 多路复用功能示例



1. 此表仅适用于 5 类器件。

17.4.8 流程图

图 74. 流程图示例



17.5 LCD 低功耗模式

LCD 控制器可在停止模式下进行显示，也可以完全禁止以降低功耗。

表 74. 低功耗模式下的 LCD 行为

模式	说明
停止	LCD 仍处于激活状态
待机	LCD 未处于激活状态

17.6 LCD 中断

下表列出了 LCD 中断请求。

表 75. LCD 中断请求

中断事件	事件标志	事件标志/中断清除方法	中断使能控制位
帧起始 (SOF)	SOF	写入 SOFC = 1	SOFIE
更新显示完成 (UDD)	UDD	写入 UDDC = 1	UDDIE

帧起始 (SOF)

如果 SOFIE（帧起始中断使能）位置 1，则执行 LCD 帧起始中断（请参见*第 17.7.2 节：LCD 帧控制寄存器 (LCD_FCR)*）。执行相应的中断处理向量时在 LCD_CLR 寄存器中将 SOFC 位写为 1，可将 SOF 清除。

更新显示完成 (UDD)

如果 UDDIE（更新显示完成中断使能）位置 1，则执行 LCD 更新显示中断（请参见*第 17.7.2 节：LCD 帧控制寄存器 (LCD_FCR)*）。执行相应的中断处理向量时在 LCD_CLR 寄存器中将 UDDC 位写为 1，可将 UDD 清除。

根据产品实现的不同，上述所有中断事件既可以共享同一个中断向量（LCD 全局中断），也可以分组到 2 个不同的中断向量上（LCD SOF 中断和 LCD UDD 中断）。有关详细信息，请参见*表 51：向量列表*。

要使能 LCD 中断，需按照以下顺序操作：

1. 配置 NVIC 中的 LCD IRQ 通道并将其使能
2. 配置 LCD 以生成中断

17.7 LCD 寄存器

外设寄存器必须按字（32 位）进行访问。

17.7.1 LCD 控制寄存器 (LCD_CR)

LCD control register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUX_SEG	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:8 保留，必须保持复位值

位 7 **MUX_SEG**：多路复用区段使能 (Mux segment enable)

此位用于使能 SEG 引脚重映射。四个 SEG 引脚可与 SEG[31:28] 多路复用。请参见*第 17.4.7 节*。

0：禁止 SEG 引脚多路复用

1：SEG[31:28] 与 SEG[43:40] 多路复用

位 6:5 **BIAS[1:0]**：偏置选择器 (Bias selector)

这些位决定使用的偏置。禁止使用值 11。

00：偏置 1/4

01：偏置 1/2

10：偏置 1/3

11：保留

位 4:2 **DUTY[2:0]**：占空比选择 (Duty selection)

这些位决定占空比。禁止使用值 101、110 和 111。

000：静态占空比

001：1/2 占空比

010：1/3 占空比

011：1/4 占空比

100：1/8 占空比

101：保留

110：保留

111：保留

位 1 **VSEL**：电压源选择 (Voltage source selection)
VSEL 位决定 LCD 的电压源。
0：内部源（电压升压转换器）
1：外部源（VLCD 引脚）

位 0 **LCDEN**：LCD 控制器使能 (LCD controller enable)
通过软件将此位置 1 可使能 LCD 控制器/驱动器。通过软件将此位清零可在下一帧开始时关闭 LCD。当 LCD 禁用时，所有 COM 和 SEG 引脚都被驱动到 V_{SS}。当此位置 1 时，必须在 PWR_CR 中复位 ULP 位。
0：禁止 LCD 控制器
1：使能 LCD 控制器

注：当使能 LCD 时（LCD_SR 中的 ENS 位置 1），VSEL、MUX_SEGBIAS 和 DUTY 位处于写保护状态。

17.7.2 LCD 帧控制寄存器 (LCD_FCR)

LCD frame control register

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PS[3:0]				DIV[3:0]				BLINK[1:0]	
						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
BLINKF[2:0]			CC[2:0]			DEAD[2:0]			PON[2:0]			UDDIE	Res.	SOFIE	HD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

位 31:26 保留，必须保持复位值

位 25:22 **PS[3:0]**：PS 16 位预分频器 (PS 16-bit prescaler)
这些位通过软件写入，用于定义 PS 16 位预分频器的分频系数。
ck_ps = LCDCLK/(2)。请参见*第 17.4.2 节*。
0000：ck_ps = LCDCLK
0001：ck_ps = LCDCLK/2
0002：ck_ps = LCDCLK/4
...
1111：ck_ps = LCDCLK/32768

位 21:18 **DIV[3:0]**：DIV 时钟分频器 (DIV clock divider)
这些位通过软件写入，用于定义 DIV 分频器的分频系数。请参见*第 17.4.2 节*。
0000：ck_div = ck_ps/16
0001：ck_div = ck_ps/17
0002：ck_div = ck_ps/18
...
1111：ck_div = ck_ps/31

位 17:16 **BLINK[1:0]**：闪烁模式选择 (Blink mode selection)

00：禁止闪烁
01：在 SEG[0]、COM[0] 上启用闪烁（1 个像素）
10：在 SEG[0]、所有 COM 上启用闪烁（最多 8 个像素，取决于编程的占空比）
11：在所有 SEG 和所有 COM 上启用闪烁（所有像素）

位 15:13 **BLINKF[2:0]**：闪烁频率选择 (Blink frequency selection)

000：$f_{LCD}/8$
001：$f_{LCD}/16$
010：$f_{LCD}/32$
011：$f_{LCD}/64$
100：$f_{LCD}/128$
101：$f_{LCD}/256$
110：$f_{LCD}/512$
111：$f_{LCD}/1024$

位 12:10 **CC[2:0]**：对比度控制 (Contrast control)

这些位指定 V_{LCD} 最大电压之一（与 V_{DD} 无关）。范围为 2.60 V 至 3.51 V。

000：V_{LCD0}
001：V_{LCD1}
010：V_{LCD2}
011：V_{LCD3}
100：V_{LCD4}
101：V_{LCD5}
110：V_{LCD6}
111：V_{LCD7}

有关 V_{LCDx} 的值，请参见产品数据手册。

位 9:7 **DEAD[2:0]**：死区持续时间 (Dead time duration)

这些位通过软件写入，用于配置帧间死区的长度。在死区内，COM 和 SEG 电压电平保持为 0 V 以降低对比度，而无需修改帧速率。

000：无死区
001：1 个相位周期死区
010：2 个相位周期死区
.....
111：7 个相位周期死区

位 6:4 **PON[2:0]**：脉冲打开持续时间 (Pulse ON duration)

这些位通过软件写入，用于根据 ck_ps 脉冲定义脉冲持续时间。较短的脉冲有助于降低功耗，但内部电阻较高的显示器可能需要更长的脉冲才能达到令人满意的对比度。

请注意，脉冲长度永远不会超过预分频 LCD 时钟周期的一半。

000:0
001：1/ck_ps
010：2/ck_ps
011：3/ck_ps
100：4/ck_ps
101：5/ck_ps
110：6/ck_ps
111：7/ck_ps

PON 持续时间示例（LCDCLK = 32.768 kHz 且 PS = 0x03）：

000：0 µs
001：244 µs
010：488 µs
011：782 µs
100：976 µs
101：1.22 ms
110：1.46 ms
111：1.71 ms

位 3 **UDDIE**：更新显示完成中断使能 (Update display done interrupt enable)

此位由软件置 1 和清零。

0：禁止 LCD 更新显示完成中断
1：使能 LCD 更新显示完成中断

位 2 保留，必须保持复位值

位 1 **SOFIE**：帧起始中断使能 (Start of frame interrupt enable)

此位由软件置 1 和清零。

0：禁止 LCD 帧起始中断
1：使能 LCD 帧起始中断

位 0 **HD**：高驱动使能 (High drive enable)

此位通过软件写入，用于使能低电阻分频器。内部电阻较高的显示器可能需要更长的驱动时间才能达到令人满意的对比度。在容许额外功耗的情况下，此位非常有用。

0: 禁止永久高驱动
1: 使能永久高驱动 当 HD=1 时，必须将 PON 位编程为 001。

注：可以随时更新此寄存器中的数据，但仅在下一帧开始时才应用新值（能够立即影响器件行为的 UDDIE 和 SOFIE 除外）。

CC[2:0] 位的新值也会立即应用，但由于电压发生器的缘故，其对器件的作用会在下一帧开始时生效。

读取该寄存器可获得在寄存器中写入的最后一个值，而不是用于显示当前帧的配置。

17.7.3 LCD 状态寄存器 (LCD_SR)

LCD status register

偏移地址：0x08

复位值：0x0000 0020

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCRSF	RDY	UDD	UDR	SOF	ENS
										r	r	r	rs	r	r

位 31:6 保留，必须保持复位值

位 5 **FCRSF**：LCD 帧控制寄存器同步标志 (LCD Frame Control Register Synchronization flag)

每次在 LCDCLK 域中更新 LCD_FCR 寄存器时，此位都会由硬件置 1。对 LCD_FCR 寄存器执行写操作时，此位将由硬件清零。

0：LCD 帧控制寄存器尚未同步

1：LCD 帧控制寄存器已同步

位 4 **RDY**：就绪标志 (Ready flag)

此位通过硬件置 1 和清零。它指示升压转换器的状态。

0：未就绪

1：升压转换器已使能且准备好提供正确的电压。

位 3 **UDD**：更新显示完成 (Update Display Done)

此位由硬件置 1，通过向 LCD_CLR 寄存器中的 UDDC 写入 1 进行清零。位置 1 的优先级高于位清零。

0：无事件

1：更新显示请求完成。如果 LCD_FCR 寄存器中的 UDDIE 位置 1，将产生 UDD 中断。

注： 如果器件处于停止模式（未提供 PCLK），即使 UDDIE = 1，UDD 也不会产生中断。
如果显示器未使能，将永远不会发生 UDD 中断。

位 2 **UDR**：更新显示请求 (Update display request)

每次软件修改 LCD_RAM 时，必须将 UDR 位置 1 以将更新的数据传送到第二级缓冲器。UDR 位在更新结束之前保持置 1，在此期间 LCD_RAM 处于写保护状态。

0：无影响

1：更新显示请求

注：当禁止显示器时，针对所有 LCD_DISPLAY 存储单元执行更新。当使能显示器时，仅针对公共线激活（取决于 DUTY）的存储单元执行更新。例如，如果 DUTY = 1/2，将仅更新 COM0 和 COM1 的 LCD_DISPLAY。

注：当此位已为 1 时，写入 0 或写入 1 都无影响。此位只能由硬件清零。仅当 LCDEN=1 时，此位才能清零

位 1 **SOF**：帧起始标志 (Start of frame flag)

此位在新的帧开始时由硬件置 1，同时更新显示数据。此位通过向 LCD_CLR 寄存器中的 SOFC 位写入 1 进行清零。位清零的优先级高于位置 1。

0：无事件

1：发生帧起始事件。如果 SOFIE 位置 1，则产生 LCD 帧起始中断。

ENS：LCD 使能状态 (LCD enabled status)

位 0 此位通过硬件置 1 和清零。它指示 LCD 控制器状态。

0：禁止 LCD 控制器

1：使能 LCD 控制器

注：当 LCD_CR 寄存器中的 LCDEN 位从 0 变为 1 时，ENS 位立即置 1。禁用时，此位反映 LCD 的实际状态，因此在最后显示的帧结束时变为 0。

17.7.4 LCD 清零寄存器 (LCD_CLR)

LCD clear register

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDDC	Res.	SOFC	Res.
												w		w	

位 31:4 保留，必须保持复位值

位 3 **UDDC**：更新显示完成清零 (Update display done clear)
此位由软件写入，用于将 LCD_SR 寄存器中的 UDD 标志清零。
0：无影响
1：将 UDD 标志清零

位 2 保留，必须保持复位值

位 1 **SOFC**：帧起始标志清零 (Start of frame flag clear)
此位由软件写入，用于将 LCD_SR 寄存器中的 SOF 标志清零。
0：无影响
1：将 SOF 标志清零

位 0 保留，必须保持复位值

17.7.5 LCD 显示器存储器 (LCD_RAM)

LCD display memory

偏移地址：0x14 到 0x50

复位值：0x0000 0000

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

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

位 31:0 SEGMENT_DATA[31:0]
每个位对应于 LCD 显示器的一个像素。
0：像素未激活
1：像素激活

17.7.6 LCD 寄存器映射

下表对 LCD 寄存器进行了汇总。

表 76. LCD 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LCD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUX_SEG	BIAS[1:0]		DUTY [2:0]			VSEL	LCDEN
	Reset value																								0	0	0	0	0	0	0	0	0
0x04	LCD_FCR	Res.	Res.	Res.	Res.	Res.	Res.	PS[3:0]				DIV[3:0]				BLINK[1:0]		BLINKF[2:0]			CC [2:0]			DEAD [2:0]			PON [2:0]			UDDIE	Res.	SOFIE	HD
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x08	LCD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCRSF	RDY	UDD	UDR	SOF	ENS
	Reset value																											1	0	0	0	0	0
0x0C	LCD_CLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDDC	Res.	SOFC	Res.
	Reset value																													0		0	
0x14	LCD_RAM (COM0)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S51	S50	S49	S48	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	LCD_RAM (COM1)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S51	S50	S49	S48	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	LCD_RAM (COM2)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	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		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S51	S50	S49	S48	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 76. LCD 寄存器映射和复位值（续）

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
0x2C	LCD_RAM (COM3)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S51	S50	S49	S48	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	LCD_RAM (COM4)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	LCD_RAM (COM5)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	LCD_RAM (COM6)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	LCD_RAM (COM7)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	S47	S46	S45	S44	S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址表，请参见第 57 页的第 2.2.2 节。

18 触摸感应控制器（TSC）

18.1 简介

触摸感应控制器提供了一种简单的解决方案，可用于向任何应用添加电容感应功能。电容感应技术能够检测电极附近的手指，该电极受电介质（例如玻璃、塑料）保护以防直接接触。由手指（或任何导电物质）产生的电容变化可基于表面电荷转移采集原理，使用已证实有效的实现方法进行测量。

触摸感应控制器受 STMTouch 触摸感应固件库完全支持，该固件库可供免费使用，用于在最终应用中可靠地实现触摸感应功能。

18.2 TSC 主要特性

触摸感应控制器具有以下主要特性：

- 采用经验证的稳定的表面电荷转移采集原理
- 支持多达 24 个电容感应通道
- 可以并行采集多达 8 个电容感应通道，因此具备极佳的响应时间
- 具有扩频功能，可以提高系统在噪声环境中的稳定性
- 对电荷转移采集序列完全采用硬件管理
- 可编程电荷转移频率
- 可编程采样电容 I/O 引脚
- 可编程通道 I/O 引脚
- 可编程最大计数值，从而避免通道出现故障时采集时间过长
- 具有中断功能的专用采集结束和最大计数错误标志
- 一个采样电容可用于多达 3 个电容感应通道，从而可减少系统组件
- 与接近、触键、线性和旋转触摸传感器实现方案兼容
- 设计为与 STMTouch 触摸感应固件库配合使用

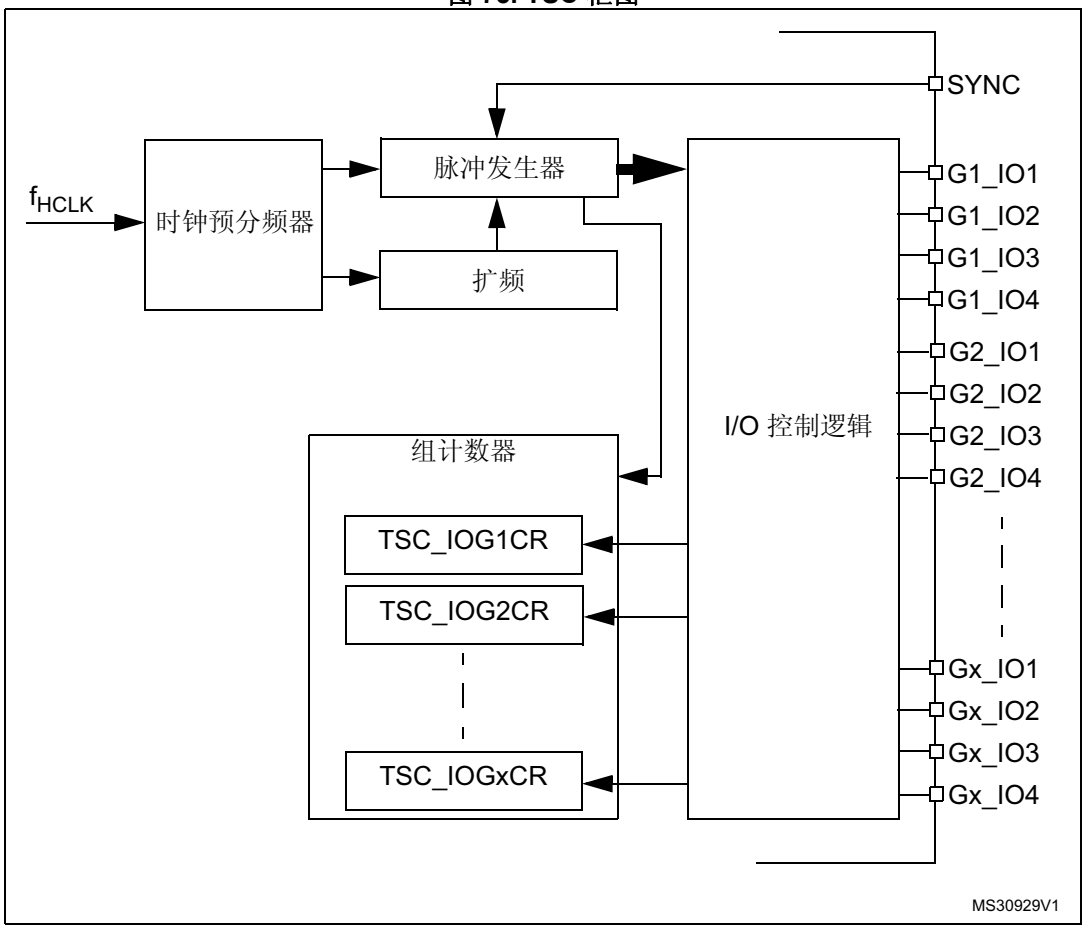
注：电容感应通道的数量取决于封装尺寸并受 I/O 可用性的限制。

18.3 TSC 功能说明

18.3.1 TSC 框图

触摸感应控制器的框图如*图 75：TSC 框图*所示。

图 75. TSC 框图



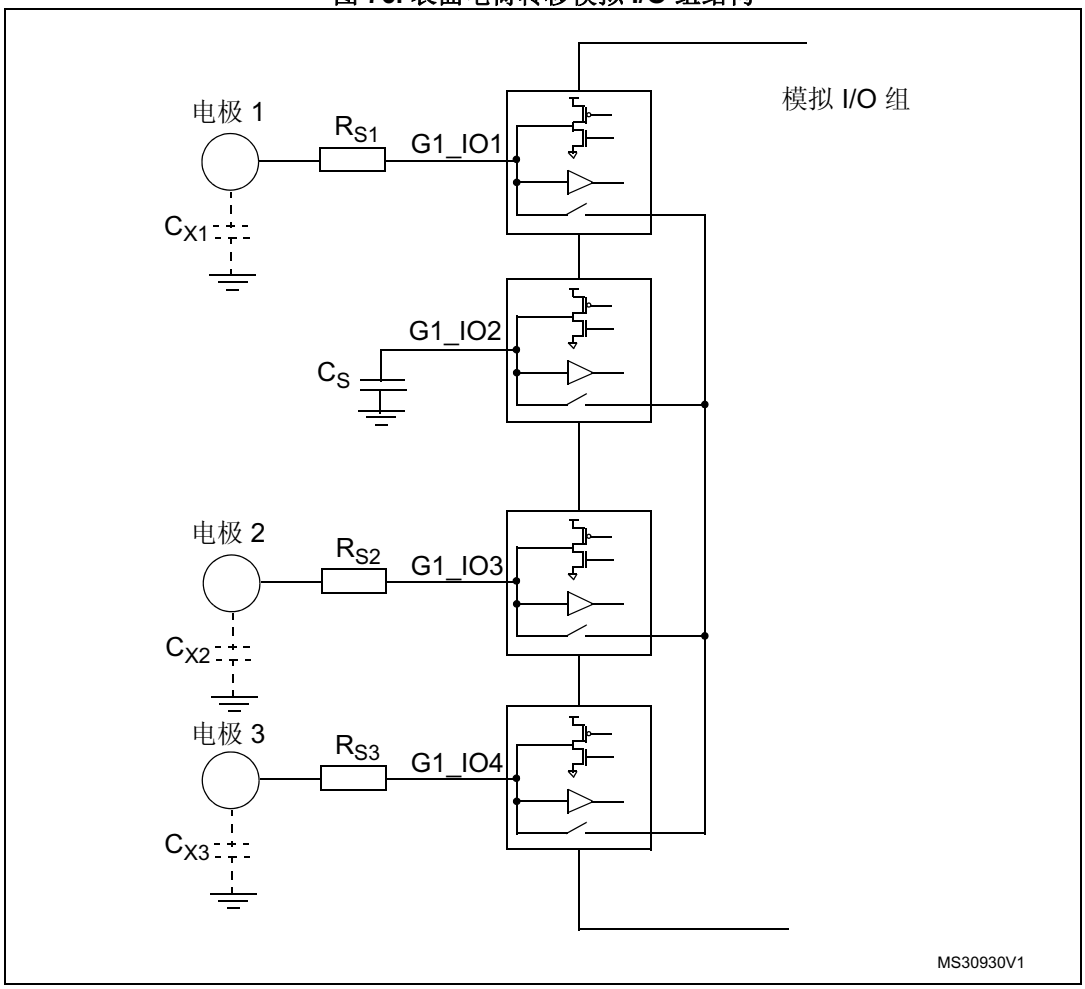
18.3.2 表面电荷转移采集概述

表面电荷转移采集是用于测量电容的一种成熟、稳定且有效的方式。它使用最少数量的外部组件对单端电极类型进行操作。该采集围绕模拟 I/O 组设计，后者由四个 GPIO 组成。（请参见*图 76*）。提供多个模拟 I/O 组，以便可以同时采集多个电容感应通道并支持更多的电容感应通道。在同一个模拟 I/O 组内，电容感应通道的采集操作是连续的。

其中一个 GPIO 专门用于采样电容 C_S。每个模拟 I/O 组在同一时间必须仅使能一个采样电容 I/O。

其余的 GPIO 专用于电极，通常称为通道。对于一些特定需要（如接近检测），每个模拟 I/O 组也可以同时使能多个通道。

图 76. 表面电荷转移模拟 I/O 组结构



注： *Gx_IOy，其中 x 是模拟 I/O 组编号，y 是所选组内的 GPIO 编号。*

表面电荷转移采集原理包括对电极电容 (C_X) 进行充电并将一部分累积电荷转移到采样电容 (C_S)。重复该过程，直到 C_S 两端的电压达到给定的阈值（在本例中为 V_{IH}）。达到阈值所需的电荷转移次数直接表示电极电容的大小。

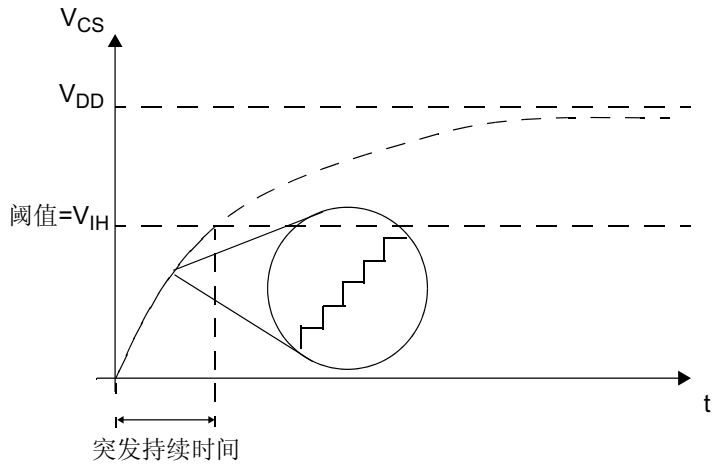
表 77 详细说明了电容感应通道 1 的电荷转移采集序列。重复状态 3 到 7，直到 C_S 两端的电压达到给定的阈值。同一序列适用于其他通道的采集。电极串联电阻 R_S 提高了解决方案的 ESD 抗扰性。

表 77. 采集序列汇总

状态	G1_IO1（电极）	G1_IO2（采样）	G1_IO3（电极）	G1_IO4（电极）	状态说明
#1	输入浮空且模拟开关闭合	开漏输出处于低电平，且模拟开关闭合	输入浮空且模拟开关闭合		将所有 C_X 和 C_S 放电
#2	输入浮空				死区时间
#3	推挽输出处于高电平	输入浮空			将 C_{X1} 充电
#4	输入浮空				死区时间
#5	输入浮空且模拟开关闭合		输入浮空		将电荷从 C_{X1} 转移到 C_S
#6	输入浮空				死区时间
#7	输入浮空				测量 C_S 电压

下图详细描述了采样电容 C_S 的电压随时间的变化情况：

图 77. 采样电容电压变化



18.3.3 复位和时钟

TSC 时钟源是 AHB 时钟 (HCLK)。两个可编程预分频器用于产生脉冲发生器和扩频内部时钟：

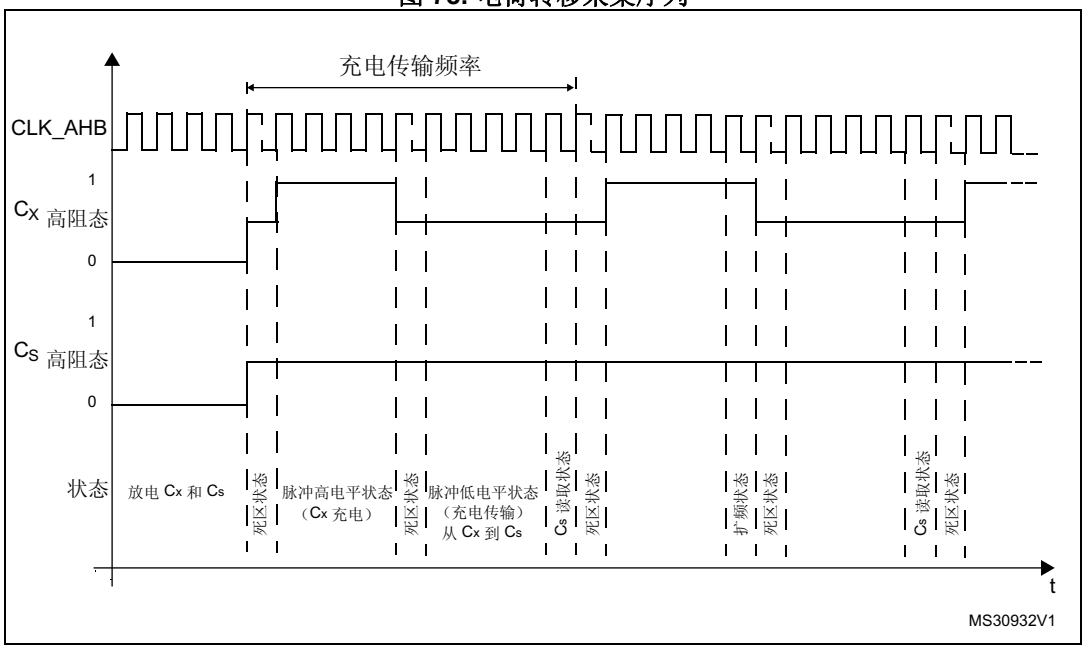
- 脉冲发生器时钟 (PGCLK) 使用 TSC_CR 寄存器的 PGPSC[2:0] 位进行定义
- 扩频时钟 (SSCLK) 使用 TSC_CR 寄存器的 SSPSC 位进行定义

复位和时钟控制器 (RCC) 提供用于使能触摸感应控制器时钟及复位该外设的专用位。有关详细信息，请参见*第 7 节：复位和时钟控制 (RCC)*。

18.3.4 电荷转移采集序列

图 78 详细描述了电荷转移采集序列的示例。

图 78. 电荷转移采集序列



为提高灵活性，可完全配置电荷转移频率。可以使用 TSC_CR 寄存器中的 CTPH[3:0] 和 CTPL[3:0] 位定义脉冲高电平状态（为 C_X 充电）和脉冲低电平状态（将电荷从 C_X 转移到 C_S）的持续时间。脉冲高电平和低电平状态持续时间的标准范围为 500 ns 到 2 µs。为了确保正确测量电极电容，必须设置脉冲高电平状态持续时间以确保 C_X 始终充分充电。

在脉冲高电平和低电平状态之间插入死区时间（在此期间，采样电容 I/O 和通道 I/O 处于输入浮空状态），以确保最优的电荷转移采集序列。死区时间状态的持续时间是 HCLK 的 2 个周期。

如果已启用扩频功能，在脉冲高电平状态结束时，将增加可变数量的 SSCLK 时钟周期。

在脉冲低电平状态的结束时，会读取采样电容 I/O，确定 C_S 两端的电压是否达到给定阈值，读取操作的持续时间是 HCLK 的一个周期。

注： *禁止采用以下 TSC 控制寄存器配置：*

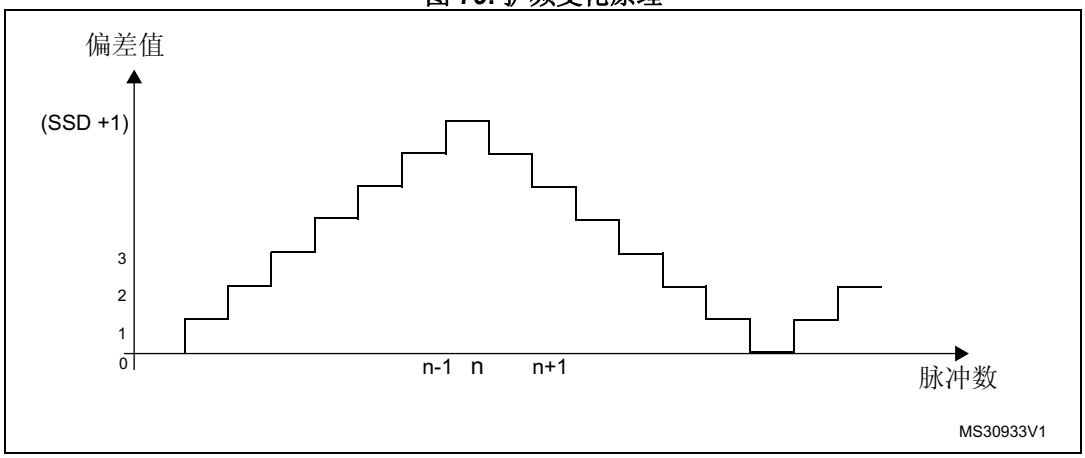
- *PGPSC 位设置为 “0” 且 CTPL 位设置为 “0”*
- *PGPSC 位设置为 “0” 且 CTPL 位设置为 “1”*
- *PGPSC 位设置为 “1” 且 CTPL 位设置为 “0”*

18.3.5 扩频功能

扩频功能用于使电荷转移频率发生变化。这样做是为了改善噪声环境下电荷转移采集的稳定性，并减少由此产生的辐射。最大频率变化范围为标称电荷转移周期的 10% 到 50%。例如，对于 250 kHz (4 µs) 的标称电荷转移频率，典型的扩频偏差为 10% (400 ns)，这将导致出现 227 kHz 的最低电荷转移频率。

实际上，扩频包括使用以下所示的原理将可变数量的 SSCLK 周期添加到脉冲高电平状态：

图 79. 扩频变化原理



下表详细列出了不同 HCLK 设置对应的最大频率偏差：

表 78. 扩频偏差与 AHB 时钟频率之间的关系

f_{HCLK}	扩频步长	最大扩频偏差
24 MHz	41.6 ns	10666.6 ns
32 MHz	27.7 ns	7111.1 ns

可以使用 TSC_CR 寄存器中的 SSE 位禁止/使用扩频功能。也可以根据器件 HCLK 时钟频率和所选电荷转移频率使用 TSC_CR 寄存器中的 SSPSC 和 SSD[6:0] 配置频率偏差。

18.3.6 最大计数错误

最大计数错误可防止因电容感应通道发生故障而导致采集时间较长。它包括为模拟 I/O 组计数器指定最大计数值。最大计数值可使用 TSC_CR 寄存器中的 MCV[2:0] 位进行指定。采集组计数器达到此最大值后，正在进行的采集操作将停止，并且采集结束标志（EOAF 位）和最大计数错误标志（MCEF 位）都会置 1。如果相应的采集结束（EOAIE 位）或/和最大计数错误（MCEIE 位）中断使能位都置 1，还可以产生中断。

18.3.7 采样电容 I/O 和通道 I/O 模式选择

要实现由触摸感应控制器控制 GPIO，必须通过标准 GPIO 寄存器和 GPIOxAFR 寄存器使能相应的复用功能。

GPIO 模式由 TSC 控制，并使用 TSC_IOSCR 和 TSC_IOCCR 寄存器进行定义。

当未执行任何采集操作时，由触摸感应控制器控制的所有 I/O 都处于默认状态。当执行采集操作时，仅未使用的 I/O（未定义为采样电容 I/O 或通道 I/O）处于默认状态。TSC_CR 寄存器中的 IODEF 位定义处于默认状态的 I/O 的配置。下表根据 I/O 模式对其配置进行了汇总。

表 79. I/O 状态（取决于其模式和 IODEF 位值）

IODEF 位	采集状态	未使用的 I/O 模式	电极 I/O 模式	采样电容 I/O 模式
0 （推挽输出处于低电平）	未执行	推挽输出处于低电平	推挽输出处于低电平	推挽输出处于低电平
0 （推挽输出处于低电平）	正在执行	推挽输出处于低电平	-	-
1 （输入浮空）	未执行	输入浮空	输入浮空	输入浮空
1 （输入浮空）	正在执行	输入浮空	-	-

未使用的 I/O 模式

未使用的 I/O 对应于由 TSC 外设控制的 GPIO，但未定义为电极 I/O 或采样电容 I/O。

采样电容 I/O 模式

要实现由 TSC 外设控制采样电容 I/O，必须首先将相应的 GPIO 设置为复用输出开漏模式，然后必须将 TSC_IOSCR 寄存器中相应的 Gx_IOy 位置 1。

每个模拟 I/O 组在同一时间必须仅使能一个采样电容。

通道 I/O 模式

要实现由 TSC 外设控制通道 I/O，必须首先将相应的 GPIO 设置为复用输出推挽模式，然后必须将 TSC_IOCCR 寄存器中的相应 Gx_IOy 位置 1。

对于要求更高等效电极表面的接近检测或为了加快采集过程，可以使能属于同一模拟 I/O 组的多个通道并同时进行采集。

注：在采集阶段，即使未使能 TSC 外设复用功能，只要 TSC_IOSCR 或 TSC_IOCCR 位置 1，相应的 GPIO 模拟开关就由触摸感应控制器自动进行控制。

18.3.8 采集模式

触摸感应控制器具有两种采集模式：

- 正常采集模式：TSC_CR 寄存器中的 START 位置 1 后，采集会立即开始。
- 同步采集模式：通过将 TSC_CR 寄存器中的 START 位置 1 来使能采集，但仅当检测到下降沿或上升沿并且 SYNC 输入引脚处于高电平时，采集才会开始。此模式可用于将电容感应通道采集与外部信号同步，而不会额外增加 CPU 负载。

TSC_IOGCSR 寄存器中的 GxE 位用于指定使能哪个模拟 I/O 组（相对计数器正在计数）。已禁止模拟 I/O 组的 C_S 电压不会受到监视，而且该组不会参与触发采集结束标志。但是，如果已禁止模拟 I/O 组包含一些通道，这些通道将被脉冲化。

已使能的模拟 I/O 组的 C_S 电压达到给定阈值时，TSC_IOGCSR 寄存器中相应的 GxS 位置 1。当已使能的所有模拟 I/O 组的采集完成时（已使能的所有模拟 I/O 组的所有 GxS 位都置 1），TSC_ISR 寄存器中的 EOAF 标志会置 1。如果 TSC_IER 寄存器中的 EOAIE 位置 1，则会产生中断请求。

如果检测到最大计数错误，正在进行的采集将会停止，并且 TSC_ISR 寄存器中的 EOAF 和 MCEF 标志都置 1。如果相应位（TSCIER 寄存器的 EOAIE 和 MCEIE 位）置 1，可以针对这两种事件产生中断请求。注意，当检测到最大计数错误时，已使能模拟 I/O 组中的其余 GxS 位不会置 1。

要清除中断标志，TSC_ICR 寄存器中相应的 EOAIC 和 MCEIC 位必须置 1。

当开始新的采集时，模拟 I/O 组计数器清零。采集完成时，这些计数器会更新为在相应通道上产生的电荷转移周期数。

有关代码示例，请参见 *A.10.1：TSC 配置代码示例*。

18.3.9 I/O 滞后和模拟开关控制

为提高灵活性，触摸感应控制器还可用于控制每个 Gx_IOy 的施密特触发器滞后和模拟开关。在触摸感应控制器使能后，无论 I/O 控制模式是由标准 GPIO 寄存器还是由其他外设进行控制，此控制均可用。该控制还可用于执行不同的采集序列或用于其他用途。

为提高系统的抗扰性，必须通过复位 TSC_IOHCR 寄存器中相应的 Gx_IOy 位来禁止由 TSC 控制的 GPIO 施密特触发器滞后。

18.4 TSC 低功耗模式

表 80. 低功耗模式对 TSC 的作用

模式	说明
睡眠	无任何作用 TSC 中断可使器件退出睡眠模式。
停止	TSC 寄存器被冻结 TSC 停止其操作，直到退出停止或待机模式。
待机	

18.5 TSC 中断

表 81. 中断控制位

中断事件	使能控制位	事件标志	标志清零位	退出 睡眠模式	退出 停止模式	退出 待机模式
采集结束	EOAIE	EOAIF	EOAIC	是	否	否
最大计数错误	MCEIE	MCEIF	MCEIC	是	否	否

有关代码示例，请参见 *A.10.2：TSC 中断代码示例*。

18.6 TSC 寄存器

有关寄存器说明中使用的缩写，请参见参考手册中的*第 50 页的第 1.1 节*。

外设寄存器可按字（32 位）进行访问。

18.6.1 TSC 控制寄存器 (TSC_CR)

TSC control register

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
CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE
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
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 **CTPH[3:0]**：电荷转移脉冲处于高电平 (Charge transfer pulse high)

这些位由软件置 1 和清零。它们定义电荷转移脉冲处于高电平状态的持续时间（C_X 充电）。

0000：1x t_{PGCLK}
0001：2x t_{PGCLK}
...
1111：16x t_{PGCLK}

注： 正在进行采集时不得修改这些位。

位 27:24 **CTPL[3:0]**：电荷转移脉冲处于低电平 (Charge transfer pulse low)

这些位由软件置 1 和清零。它们定义电荷转移脉冲处于低电平状态的持续时间（将电荷从 C_X 转移到 C_S）。

0000：1x t_{PGCLK}
0001：2x t_{PGCLK}
...
1111：16x t_{PGCLK}

注： 正在进行采集时不得修改这些位。

注： 有些配置禁止采用。更多详细信息，请参见第 18.3.4 节：电荷转移采集序列。

位 23:17 **SSD[6:0]**：扩频偏差 (Spread spectrum deviation)

这些位由软件置 1 和清零。它们定义扩频偏差，扩频偏差是由将可变数量的 SSCLK 时钟周期添加到电荷转移脉冲高电平状态产生的。

0000000：1x t_{SSCLK}
0000001：2x t_{SSCLK}
...
1111111：128x t_{SSCLK}

注： 正在进行采集时不得修改这些位。

位 16 **SSE**：扩频使能 (Spread spectrum enable)

此位由软件置 1 和清零，用于使能/禁止扩频功能。

0：禁止扩频
1：使能扩频

注： 正在进行采集时不得修改该位。

位 15 **SSPSC**：扩频预分频器 (Spread spectrum prescaler)

该位由软件置 1 和清零。它选择用于生成扩频时钟 (SSCLK) 的 AHB 时钟分频器。

0：f_{HCLK}
1：f_{HCLK} /2

注： 正在进行采集时不得修改该位。

位 14:12 **PGPSC[2:0]**：脉冲发生器预分频器 (pulse generator prescaler)

这些位由软件置 1 和清零。它们选择用于生成脉冲发生器时钟 (PGCLK) 的 AHB 时钟分频器。

000：f_{HCLK}
001：f_{HCLK} /2
010：f_{HCLK} /4
011：f_{HCLK} /8
100：f_{HCLK} /16
101：f_{HCLK} /32
110：f_{HCLK} /64
111：f_{HCLK} /128

注： 正在进行采集时不得修改这些位。

注： 有些配置禁止采用。更多详细信息，请参见第 18.3.4 节：电荷转移采集序列。

位 11:8 保留，必须保持复位值。

位 7:5 **MCV[2:0]**：最大计数值 (Max count value)

这些位由软件置 1 和清零。它们定义在产生最大计数错误之前可以生成的电荷转移脉冲的最大数量。

000：255
001：511
010：1023
011：2047
100：4095
101：8191
110：16383
111：保留

注： 正在进行采集时不得修改这些位。

位 4 **IODEF**：I/O 默认模式 (I/O Default mode)

该位由软件置 1 和清零。它定义当未执行任何采集时所有 TSC I/O 的配置。当存在正在执行的采集时，它定义所有未使用 I/O（未定义为采样电容 I/O 或通道 I/O）的配置

0：强制 I/O 处于推挽输出低电平状态
1：I/O 处于输入浮空状态

注： 正在进行采集时不得修改该位。

位 3 **SYNCPOL**：同步引脚极性 (Synchronization pin polarity)

此位由软件置 1 和清零，以选择同步输入引脚的极性。

0：仅下降沿

1：上升沿和高电平

位 2 **AM**：采集模式 (Acquisition mode)

此位由软件置 1 和清零，用以选择采集模式。

0：正常采集模式（START 位置 1 后，采集立即开始）

1：同步采集模式（如果 START 位置 1，并在 SYNC 输入引脚上检测到选定的信号，则采集开始）

注： 正在进行采集时不得修改该位。

位 1 **START**：启动新的采集 (Start a new acquisition)

此位由软件置 1，用于启动新的采集。采集完成后，此位会由硬件立即清零，也可由软件将此位清零以取消正在执行的采集。

0：不启动采集

1：启动新的采集

位 0 **TSCE**：触摸感应控制器使能 (Touch sensing controller enable)

此位由软件置 1 和清零，用于使能/禁止触摸感应控制器。

0：禁止触摸感应控制器

1：使能触摸感应控制器

注： 禁止触摸感应控制器后，TSC 寄存器设置不起任何作用。

18.6.2 TSC 中断使能寄存器 (TSC_IER)

TSC interrupt enable register

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
														rw	rw

位 31:2 保留，必须保持复位值。

位 1 **MCEIE**：最大计数错误中断使能 (Max count error interrupt enable)

该位由软件置 1 和清零，用于使能/禁止最大计数错误中断。

0：禁止最大计数错误中断

1：使能最大计数错误中断

位 0 **EOAIE**：采集结束中断使能 (End of acquisition interrupt enable)

该位由软件置 1 和清零，用于使能/禁止采集结束中断。

0：禁止采集结束中断

1：使能采集结束中断

18.6.3 TSC 中断清零寄存器 (TSC_ICR)

TSC interrupt clear register

偏移地址：0x08

复位值：0x0000 0000

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

位 31:2 保留，必须保持复位值。

位 1 **MCEIC**：最大计数错误中断清零 (Max count error interrupt clear)

该位由软件置 1，用于清除最大计数错误标志。该标志复位后，该位由硬件清零。写入“0”不会起任何作用。

0：无任何作用

1：将 TSC_ISR 寄存器中相应的 MCEF 清零

位 0 **EOAIC**：采集结束中断清零 (End of acquisition interrupt clear)

该位由软件置 1，用于清除采集结束标志。该标志复位后，该位由硬件清零。写入“0”不会起任何作用。

0：无任何作用

1：将 TSC_ISR 寄存器的相应 EOAF 清零

18.6.4 TSC 中断状态寄存器 (TSC_ISR)

TSC interrupt status register

偏移地址：0x0C

复位值：0x0000 0000

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

位 31:2 保留，必须保持复位值。

位 1 **MCEF**：最大计数错误标志 (Max count error flag)

模拟 I/O 组计数器达到指定的最大计数值时，该位由硬件置 1。由软件向 TSC_ICR 寄存器的 MCEIC 位写入 1，可将此位清零。

0：未检测到最大计数错误 (MCE)
1：检测到最大计数错误 (MCE)

位 0 **EOAF**：采集结束标志 (End of acquisition flag)

当已使能的所有组的采集完成时（已使能的所有模拟 I/O 组的所有 GxS 位都置 1，或检测到最大计数错误），该位由硬件置 1。由软件向 TSC_ICR 寄存器的 EOAIC 位写入 1，可将此位清零。

0：采集正在执行或未启动
1：采集完成

18.6.5 TSC I/O 滞后控制寄存器 (TSC_IOHCR)

TSC I/O hysteresis control register

偏移地址：0x10

复位值：0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **Gx_IOy**：Gx_IOy 施密特触发器滞后模式 (Gx_IOy Schmitt trigger hysteresis mode)

这些位由软件置 1 和清零，用于使能/禁止 Gx_IOy 施密特触发器滞后。

0：禁止 Gx_IOy 施密特触发器滞后
1：使能 Gx_IOy 施密特触发器滞后

注：这些位控制 I/O 施密特触发器滞后，无论 I/O 控制模式如何（即使由标准 GPIO 寄存器进行控制也是如此）。

18.6.6 TSC I/O 模拟开关控制寄存器 (TSC_IOASCR)

TSC I/O analog switch control register

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **Gx_IOy**：Gx_IOy 模拟开关使能 (Gx_IOy analog switch enable)

这些位由软件置 1 和清零，用于使能/禁止 Gx_IOy 模拟开关。

0：禁止 Gx_IOy 模拟开关（断开）

1：使能 Gx_IOy 模拟开关（闭合）

注：这些位控制 I/O 模拟开关，无论 I/O 控制模式如何（即使由标准 GPIO 寄存器进行控制也是如此）。

18.6.7 TSC I/O 采样控制寄存器 (TSC_IOSCR)

TSC I/O sampling control register

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **Gx_IOy**：Gx_IOy 采样模式 (Gx_IOy sampling mode)

这些位由软件置 1 和清零，用于将 Gx_IOy 配置为采样电容 I/O。每个模拟 I/O 组必须仅有一个 I/O 定义为采样电容。

0：Gx_IOy 未使用

1：Gx_IOy 用作采样电容

注：正在进行采集时不得修改这些位。

在采集阶段，即使未使能 TSC 外设复用功能，只要 TSC_IOSCR 位置 1，相应的 GPIO 模拟开关就由触摸感应控制器自动进行控制。

18.6.8 TSC I/O 通道控制寄存器 (TSC_IOCCRTSC_IOCCR)

TSC I/O channel control register

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
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
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **Gx_IOy**：Gx_IOy 通道模式 (Gx_IOy channel mode)

这些位由软件置 1 和清零，用于将 Gx_IOy 配置为通道 I/O。

0：Gx_IOy 未使用

1：Gx_IOy 用作通道

注： 正在进行采集时不得修改这些位。

在采集阶段，即使未使能 TSC 外设复用功能，只要 TSC_IOCCR 位置 1，相应的 GPIO 模拟开关就由触摸感应控制器自动进行控制。

18.6.9 TSC I/O 组控制状态寄存器 (TSC_IOGCSR)

TSC I/O group control status register

偏移地址：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
								rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值。

位 23:16 **GxS**：模拟 I/O 组 x 状态 (Analog I/O group x status)

当相应已使能模拟 I/O 组 x 上的采集完成时，这些位由硬件置 1。当新的采集开始时，这些位由硬件清零。

0：模拟 I/O 组 x 上的采集正在执行或未启动

1：模拟 I/O 组 x 上的采集已完成

注： 检测到最大计数错误后，已使能模拟 I/O 组的其余 GxS 位不置 1。

位 15:8 保留，必须保持复位值。

位 7:0 **GxE**：模拟 I/O 组 x 使能 (Analog I/O group x enable)

这些位由软件置 1 和清零，用于使能/禁止相应模拟 I/O 组 x 上的采集（计数器正在计数）。

0：禁止模拟 I/O 组 x 上的采集

1：使能模拟 I/O 组 x 上的采集

18.6.10 TSC I/O 组 x 计数器寄存器 (TSC_IOGxCR) (x = 1..8)

TSC I/O group x counter register

偏移地址：0x30 + 0x04 × 模拟 I/O 组编号

复位值：0x0000 0000

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

位 31:14 保留，必须保持复位值。

位 13:0 **CNT[13:0]**：计数器值 (Counter value)

这些位表示为完成采集（C_S 两端电压已达到阈值）在模拟 I/O 组 x 上产生的电荷转移周期数。

18.6.11 TSC 寄存器映射

表 82. TSC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	TSC_CR	CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE	SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV [2:0]			IODEF	SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x0004	TSC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
	Reset value																															0	0
0x0008	TSC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
	Reset value																															0	0
0x000C	TSC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
	Reset value																															0	0
0x0010	TSC_IOHCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0014	Reserved																																
0x0018	TSC_IOASCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	Reserved																																
0x0020	TSC_IOSCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	Reserved																																
0x0028	TSC_IOCCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																
0x0030	TSC_IOGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x0034	TSC_IOG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 82. TSC 寄存器映射和复位值（续）

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
0x0038	**TSC_IOG2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	**TSC_IOG3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	**TSC_IOG4CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**TSC_IOG5CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**TSC_IOG6CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	**TSC_IOG7CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	**TSC_IOG8CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

19 高级加密标准硬件加速器 (AES)

AES 仅可用于带 AES 外设的器件（请参见*表 2：按类别划分的特性*）。

19.1 简介

AES 硬件加速器可用于通过 AES 算法对数据进行加密和解密。加密处理器完全兼容下列标准：

- 联邦信息处理标准出版物（FIPS PUB 197，2001 年 11 月 26 日）规定的高级加密标准 (AES)

加速器会使用长度为 128 位的密钥对 128 位块进行加密和解密，还可执行密钥生成。为了最大限度地减少使用相同密钥处理多个数据块时 CPU 或 DMA 执行的写操作，会将加密或解密密钥存储在内部寄存器中。

默认情况下，会选择电子密码本 (ECB) 模式。硬件还支持密码块链接 (CBC) 或计数器 (CTR) 模式链接算法。

AES 支持对传入和传出数据进行 DMA 传输（需要 2 个 DMA 通道）。

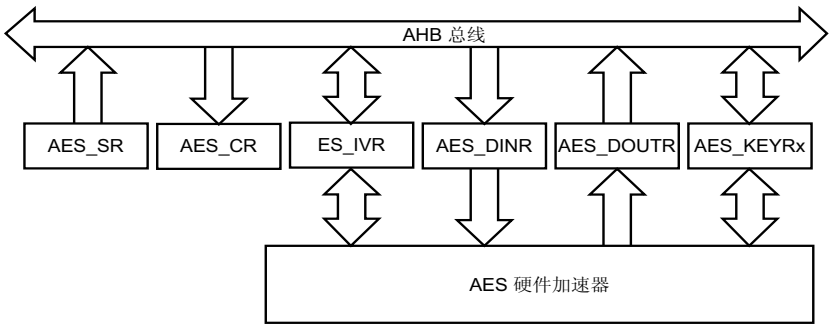
19.2 AES 主要特性

- 使用 AES Rijndael 模块密码算法进行加密/解密
- AES 加密/解密算法符合 NIST FIPS 197 标准
- 内部 128 位寄存器用于存储加密或生成密钥（4 个 32 位寄存器）
- 支持电子密码本 (ECB)、密码块链接 (CBC) 和计数器模式 (CTR)
- 密钥调度器
- 解密密钥生成
- 128 位数据块处理
- 128 位密钥长度
- 213 个时钟周期可加密或解密一个 128 位数据块（包括输入和输出阶段）
- 1 个 32 位输入缓冲器和 1 个 32 位输出缓冲器。
- 寄存器访问仅支持 32 位数据宽度。
- 一个 128 位寄存器用于初始化向量（AES 配置为 CBC 模式时）或 32 位计数器初始化（选择 CRT 模式时）。
- 采用自动数据流控制，支持直接存储器访问 (DMA)（使用 2 个通道，分别用于传入数据和传出数据）

19.3 AES 功能说明

图 80 显示了 AES 加速器的框图。

图 80. 框图



AES 加速器使用长度为 128 位的密钥处理 128 位数据块（4 个字），如果选择 CBC 或 CTR 链接模式，还会处理初始化向量。

它提供 4 种工作模式：

- 模式 1：使用 AES_KEYRx 寄存器中存储的加密密钥进行加密。
- 模式 2：在使能 AES 之前存储在内部 AES_KEYRx 寄存器的加密密钥，经过密钥生成处理后，生成的密钥仍存储于此寄存器中。此模式与选择 AES 链接模式无关。
- 模式 3：使用 AES_KEYRx 寄存器中存储的给定（预计算）解密密钥进行解密。
- 模式 4：使用 AES_KEYRx 寄存器中存储的加密密钥进行密钥生成和解密（如果 AES 配置为计数器模式来执行链接算法，则不会使用此模式）。

通过对 AES_CR 寄存器中的 MODE[1:0] 位进行编程来选择工作模式。仅当 AES 已禁止（AES_CR 寄存器中的位 EN=0）时，才允许切换模式。使能 AES 之前，必须存储 KEY 寄存器 (AES_KEYRx)。

要选择将 ECB、CBC 或 CTR 中的哪一种模式用于加密解决方案，必须在 AES 已禁止（AES_CR 寄存器中的位 En=0）时写入 AES_CR 寄存器的 CHMOD[1:0] 位和 AES_IVR 寄存器（仅用于 CBC 和 CTR 链接模式）。

使能（位 EN=1）后，AES 处于输入阶段，等待软件针对模式 1、模式 3 或模式 4 将输入数据写入 AES_DINR（4 个字）。根据所选模式，数据对应于明文消息或者是密文消息。连续两次对 AES_DINR 寄存器进行写操作时为了在两次数据交替之间，向 AES 处理器发送密钥，将自动插入一个等待周期。

对于模式 2，会在将 AES_CR 寄存器中的 EN 位置 1 后立即开始密钥生成过程。在使能 AES 之前，需要将加密的密钥加载到 AES_KEYRx 寄存器中。密钥生成过程结束时（CFF 标志置 1），会将生成密钥存储在 AES_KEYRx 寄存器中，并由硬件禁止 AES 。在该模式下，不得在 AES 已使能的情况下读取 AES_KEYRx 寄存器，应等到 CCF 标志由硬件置 1 后才能读取。

计算阶段完成后，会立即将 AES_SR 寄存器中的状态标志 CCF（计算完成标志）置 1。如果 AES_CR 寄存器中的 CCFIE=1，可产生中断。随后，软件可从 AES_DOUTR 寄存器（对于模式 1、模式 3、模式 4）或 AES_KEYRx 寄存器（若选择模式 2）中读回数据。

如果 AES_CR 寄存器中的 DMAOUTEN = 1，标志 CCF 没有任何意义，因为读取 AES_DOUTR 寄存器是通过 DMA 自动管理的，在计算过程结束时没有任何软件操作。

在模式 1、模式 3 或模式 4 下，操作以输出阶段结束，在输出阶段，软件会从 AES_DOUTR 寄存器连续读取 4 个输出数据字。在模式 2（密钥生成模式）下，会自动将数据存储在 AES_KEYRx 寄存器中，并由硬件禁止 AES。随后，软件可在使能 AES 之前选择模式 3（解密模式），开始使用该生成密钥进行解密。

在输入和输出阶段，软件必须连续读取或写入数据字节（处于模式 2 时除外），但 AES 允许每个读取或写入操作之间存在延迟（示例：如果在此时为另一中断提供服务）。

如果检测到未预期的读取或写入操作，则会将 AES_SR 寄存器中的 RDERR 和 WRERR 标志置 1。如果 AES_CR 寄存器中的 ERRIE 位置 1，可产生中断。检测到错误后，AES 不会禁止，而是会继续照常处理。

还可以使用通用 DMA 写入输入字以及读取输出字（请参见*图 95* 和*图 96*）。

可随时通过将 AES_xCR 寄存器中的 EN 位置 0 来重新初始化 AES。随后，可通过将 EN 置 1 重新启动 AES，此时，AES 会等待第一个输入数据字节写入（处于模式 2 下时除外，此时在 EN 位置 1 后，会立即从 AES_KEYRx 寄存器中存储的值开始密钥生成过程）。

19.4 加密和生成密钥

AES_KEYRx 寄存器用于存储加密或解密密钥。这四个寄存器使用小端模式配置进行组织：必须将密钥的 32 位 LSB 载入寄存器 AES_KEYR0 中，以此类推，必须将 128 位密钥的 32 位 MSB 载入 AES_KEYR3。

AES 禁止时（AES_CR 寄存器中的 EN = 0），必须将加密或解密密钥存储在这些寄存器中。密钥的字节顺序是固定的。

在模式 2（密钥生成）中，需要将加密密钥载入到 AES_KEYRx 中。然后必须使能 AES。计算阶段结束时，生成密钥会自动存储在 AES_KEYRx 寄存器中，并会覆盖之前的加密密钥。生成密钥可用时，将由硬件禁止 AES。当软件需要将 AES 切换为模式 3（解密模式）时，如果 AES_KEYRx 寄存器的内容与生成密钥（之前通过模式 2 计算）一致，则不需要写入 AES_KEYRx 寄存器。

在模式 4（密钥生成 + 解密）中，AES_KEYRx 寄存器只包含加密密钥。生成密钥会在内部计算，不会对这些寄存器执行任何写操作。

19.5 AES 链接算法

AES 硬件支持三种算法，可在 AES 禁止（位 EN = 0）的情况下通过 AES_CR 寄存器中的 CHMOD[1:0] 位进行选择：

- 电子密码本 (ECB)
- 密码块连接 (CBC)
- 计数器模式 (CTR)

19.5.1 电子密码本 (ECB)

此模式为默认模式。此模式不使用 AES_IVR 寄存器，也不会进行链接操作。消息会划分到各个块中，并对各个块分别进行加密。

图 81 和*图 82* 分别介绍了用于加密和解密的电子密码本算法的原理。

图 81. ECB 加密模式

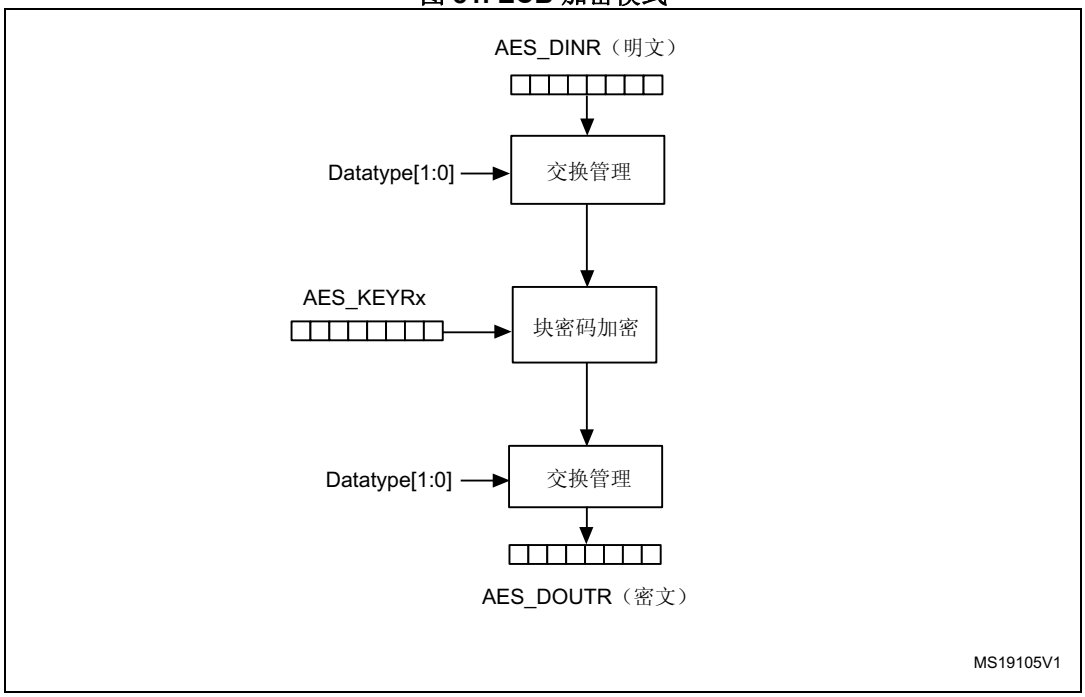
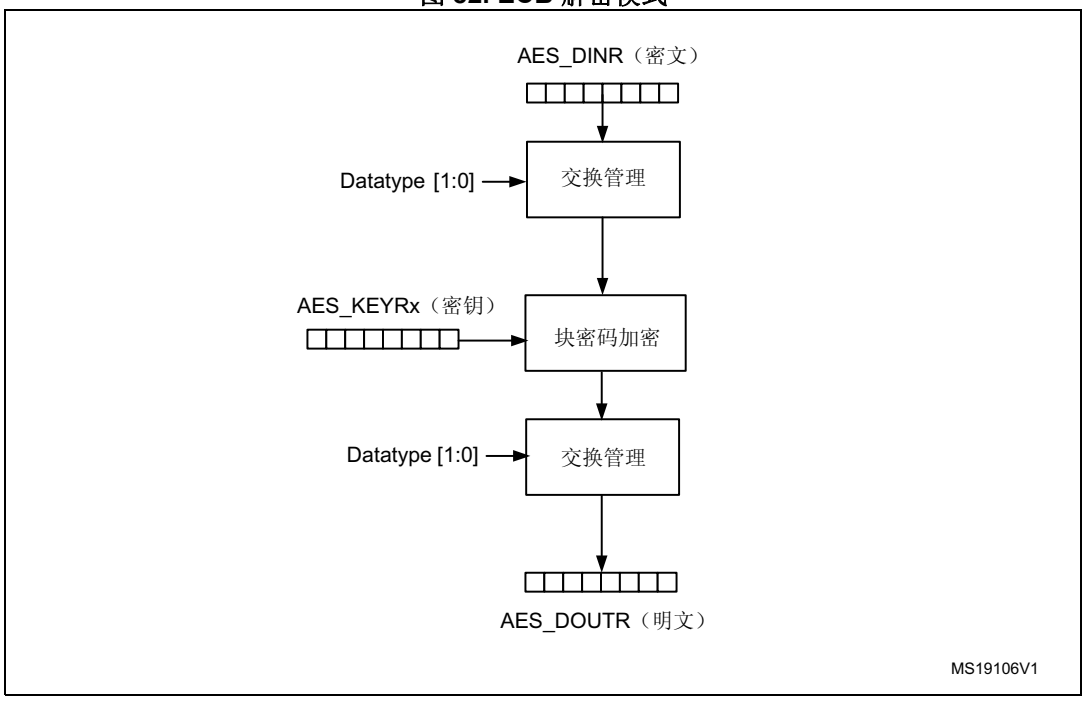


图 82. ECB 解密模式



19.5.2 密码块链接 (CBC)

在密码块链接 (CBC) 模式下，会先将每个明文块与之前的密码文本块进行异或运算，然后再进行加密。为了确保每条消息都是唯一的，会在第一个块处理过程中使用初始化向量 (AES_IVRx)。

在加密模式中交换管理块之后以及在解密模式中交换管理块之前，会对初始化向量进行异或运算（请参见*图 83* 和*图 84*）。

图 83. CBC 模式加密

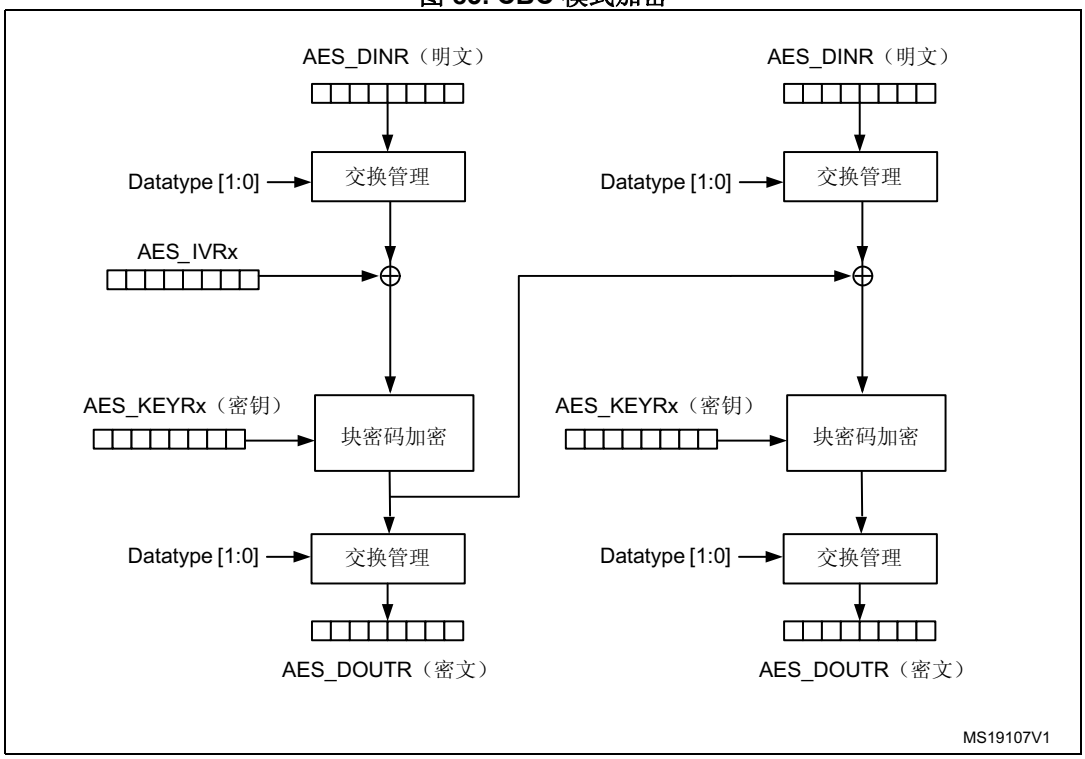
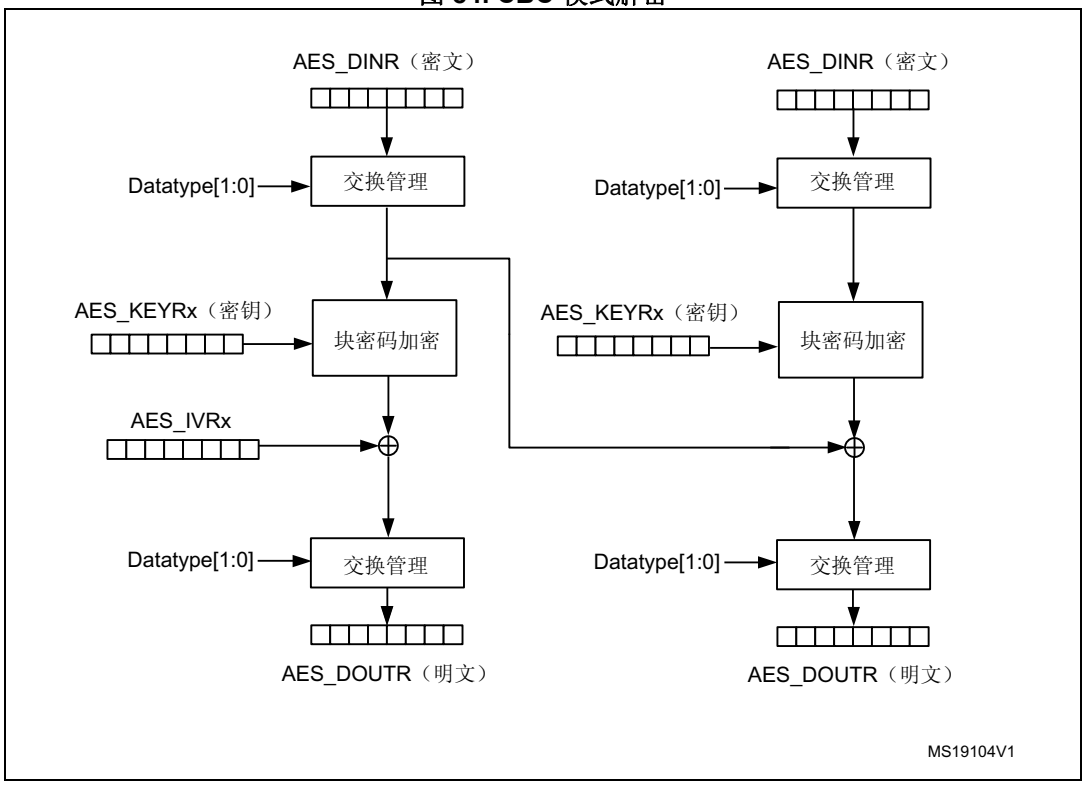


图 84. CBC 模式解密



注： *AES 使能后，读取 AES_IVR 会返回值 0x00000000。*

给定消息的挂起模式

如果需要处理另一条优先级较高的消息，可以将消息挂起。这一优先级最高的消息发送完成时，可在加密或解密模式下恢复挂起的消息。仅当数据传输通过 CPU 访问 AES_DOUTR 和 AES_DINR 寄存器的方式完成时，才可使用此功能。如果 DMA 控制器正在管理数据传输，则不建议使用此功能。

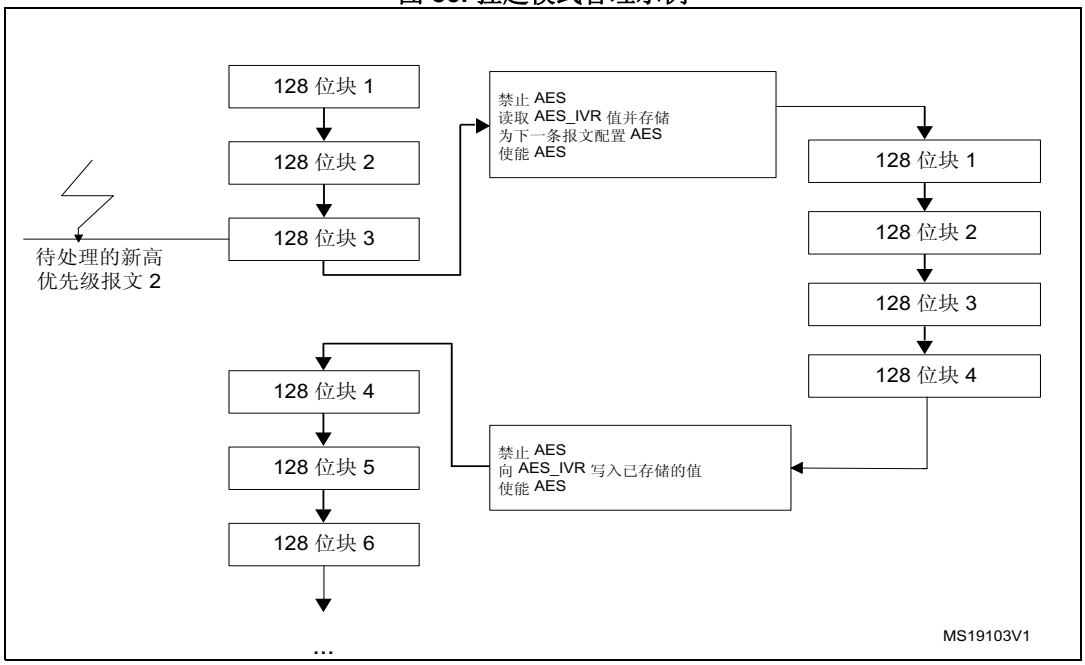
为了实现正确操作，必须在块处理结束时将消息挂起（第四次读取 AES_DOUTR 寄存器之后，以及下一次对 AES_DINR 寄存器（对应于要处理的下一个块的输入）进行写访问之前）。

应将 AES_CR 寄存器中的 EN 位置 0，以禁止 AES。软件需要读取 AES_IVRx，这些寄存器包含用于在消息中断之前进行链接异或运算的最新值。需要恢复已中断的消息时（AES 已禁止时），要立即将该值写回 AES_IVRx 寄存器进行存储，以便重新使用。需要注意的是，此过程不会破坏链接运算，并且在 AES 再次使能以发送下一个 128 位数据块之时，可立即恢复消息处理。

无论采用哪种 AES 配置（加密或解密模式），此行为都有效。

在*图 85* 的示例中，为了发送优先级更高、长度更短的消息 2 而将消息 1 挂起。128 位块处理结束时，将禁止 AES。为了不破坏链接操作，将 AES_IVR 寄存器读回进行存储，以便于后面当消息恢复时，该值可被重新写回到寄存器中。随后，会将 AES 配置为发送消息 2，并使能 AES 开始处理。消息 2 处理完成时，必须再次禁止 AES，并将之前在消息 1 中断时存储的值载入到 AES_IVRx 寄存器中。之后，一旦 AES 使能以恢复消息 1 时，软件需要立即从块 4 对应的输入值重新开始操作。

图 85. 挂起模式管理示例



19.5.3 计数器模式 (CTR)

在计数器模式下，除了随机值之外，还会使用 32 位计数器与密码文本或明文进行异或运算（请参见图 86 和图 87）。

图 86. CTR 模式加密

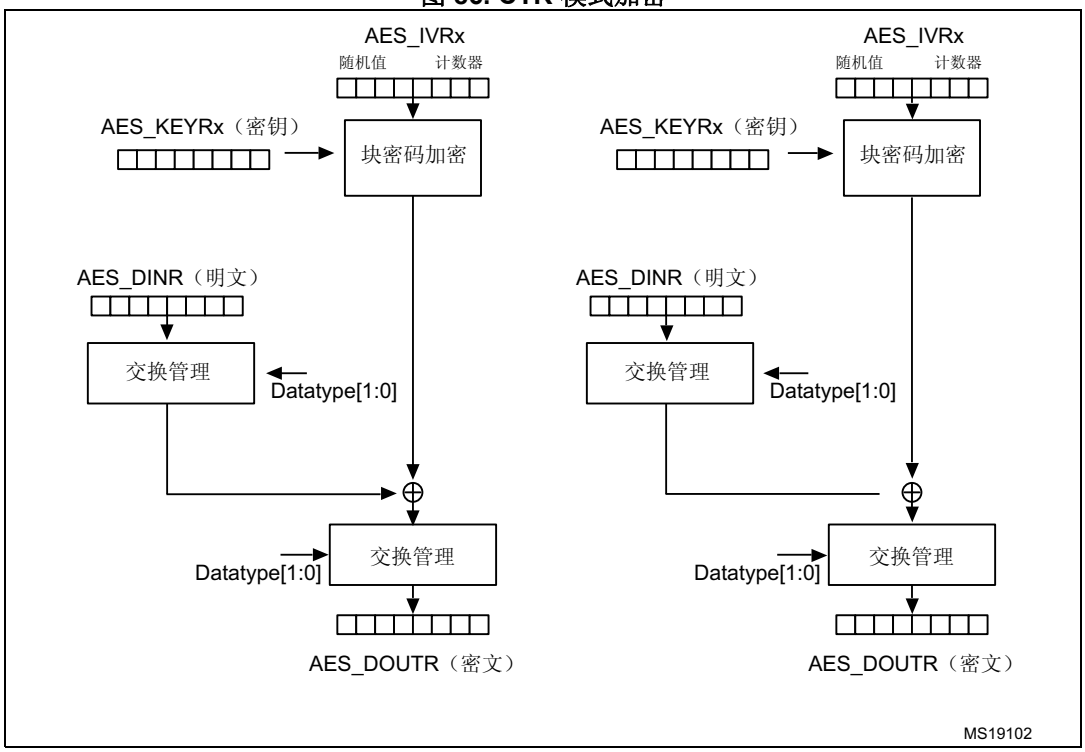
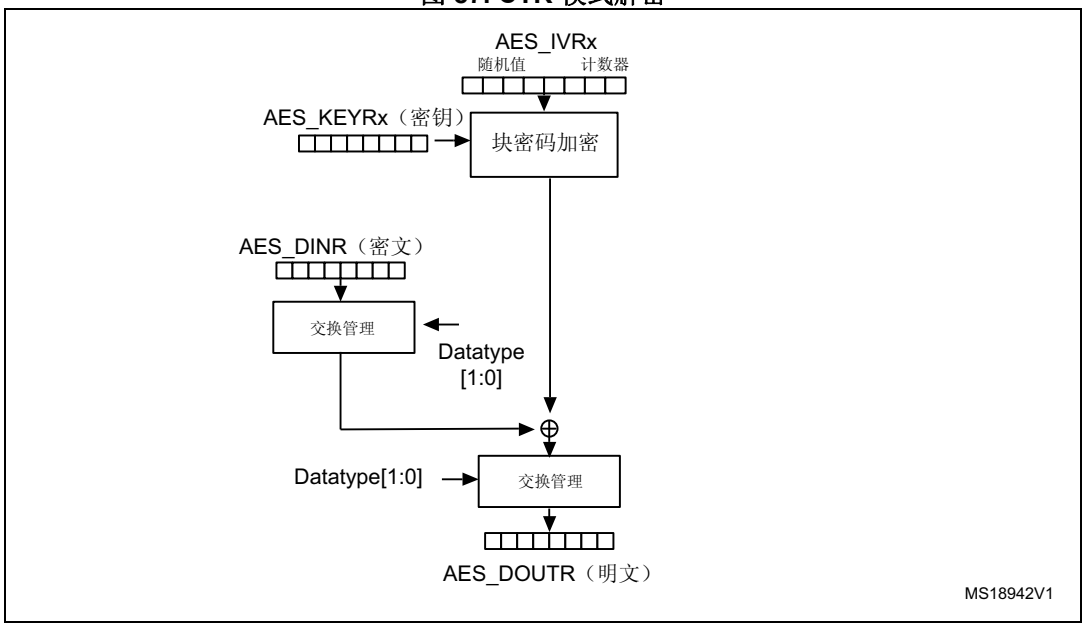
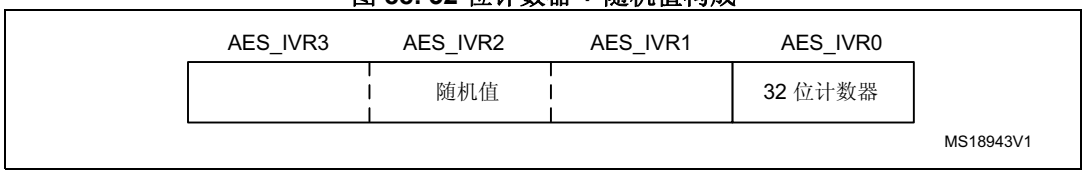


图 87. CTR 模式解密



可通过 AES_IVRx 寄存器获取随机值和 32 位计数器，其构成如下面的*图 88* 所示：

图 88. 32 位计数器 + 随机值构成



在计数器模式下，对于要处理的每个模块，计数器都会在初始值的基础上递增，以确保该序列是长时间内不会重复的唯一序列。该计数器属于 32 位计数器，这意味会对 AES 禁止时保存的初始化值保留随机消息。只有 128 位初始化向量的 32 位 LSB 才能代表计数器。与 CBC 模式（仅在处理第一个数据库块时使用一次 AES_IVRx 寄存器）相反，在计数器模式下，处理每个数据块时都会使用 AES_IVRx 寄存器。

在计数器模式下，密钥生成+解密模式不适用。

注： *仅当 AES 已禁止（位 EN = 0）时才可向 AES_IVRx 寄存器执行写入，这可确保良好的 AES 行为。*

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

在 AES 已禁止的情况下读取此寄存器会返回最新计数器值（用于管理挂起模式）。

在 CTR 模式下，密钥生成+解密没有任何意义。因此不允许将 AES_CR 寄存器中的 MODE[1:0] 设为 11，如果尝试对该配置进行设置，则会强制设为 MODE[1:0] = 10（对应于 CTR 模式解密）。设置后，会使用 AES 处理器的加密模块对消息进行解密，如图 87 所示。

CTR 模式下的挂起模式

该模式与 CBC 模式相似，可以中断消息、发送优先级更高的消息，并可恢复已中断的消息。更多关于挂起模式功能的详细信息，请参见*图 85* 和*第 19.5.2 章*。

19.6 数据类型

通过将数据写入到 AES_DINR 寄存器，数据将一次性输入到 AES 处理器 32 位（字）中。AES 可处理 128 位数据块。必须对 AES_DINR 或 AES_DOUTR 寄存器进行四次读取或写入操作才能处理 128 位数据块（MSB 优先）。

系统存储器结构采用小端模式：无论使用何种数据类型（位、字节、16 位半字、32 位字），最低有效数据均占用最低地址位置。

因此，将数据输入 AES 处理器之前，必须对要从系统存储器写入 AES_DINR 的数据执行位、字节或半字交换操作，并对要从 AES_DOUTR 寄存器读出到系统存储器的 AES 数据执行相同的交换操作，具体取决于加密或解密的数据类型。

AES_CR 寄存器中的 DATATYPE 位提供的不同交换模式将在发送到 AES 处理器之前应用到 AES_DINR 寄存器，并会应用到从处理器传出的数据的 AES_DOUTR 寄存器（请参见*图 89*）。

注： *交换操作仅与 AES_DOUTR 和 AES_DINR 寄存器有关。AES_KEYRx 和 AES_IVRx 寄存器对所选交换模式不敏感，它们有固定的小端配置（请参见第 19.4 节和第 19.12 节）。*

图 89. 根据数据类型构建 128 位块

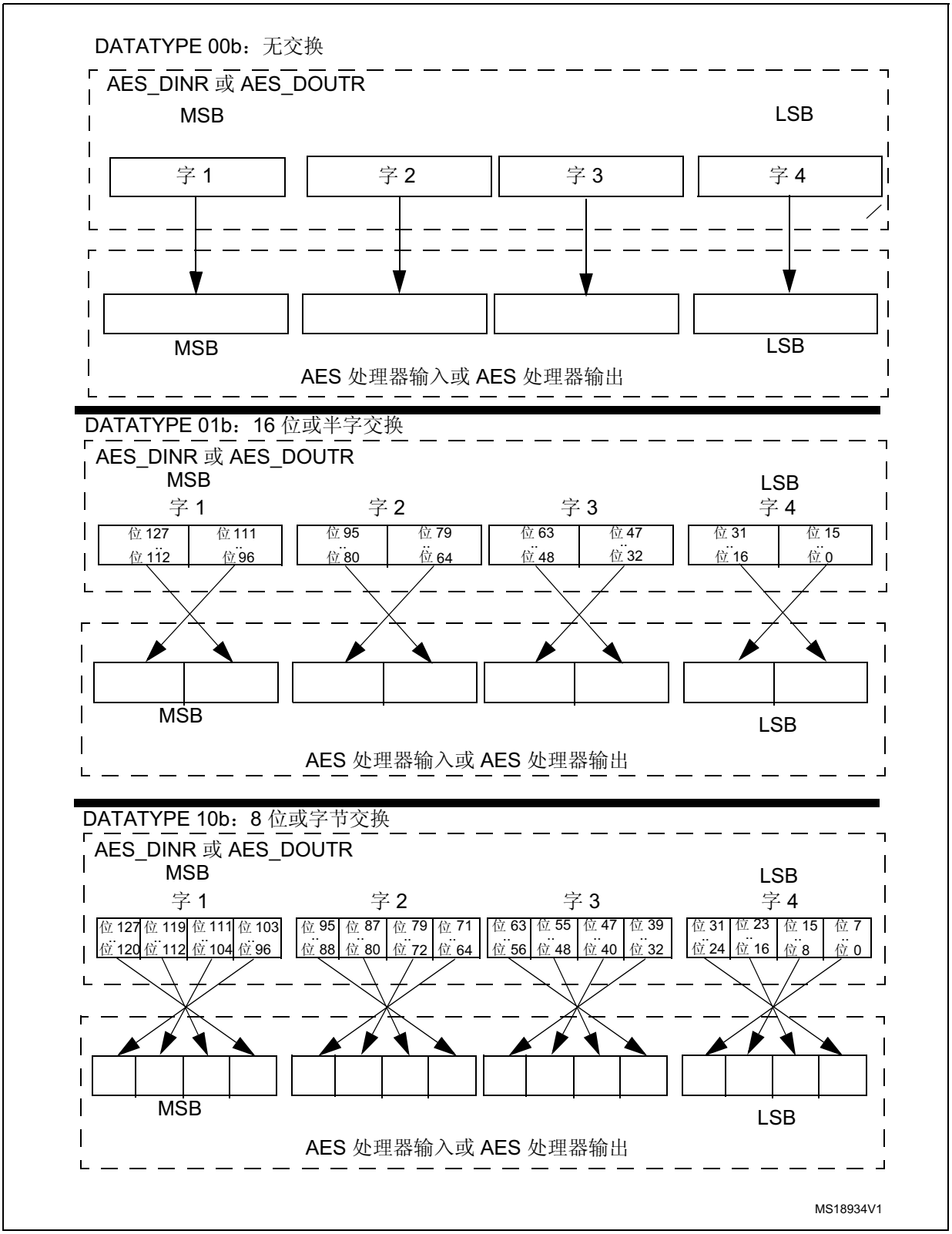
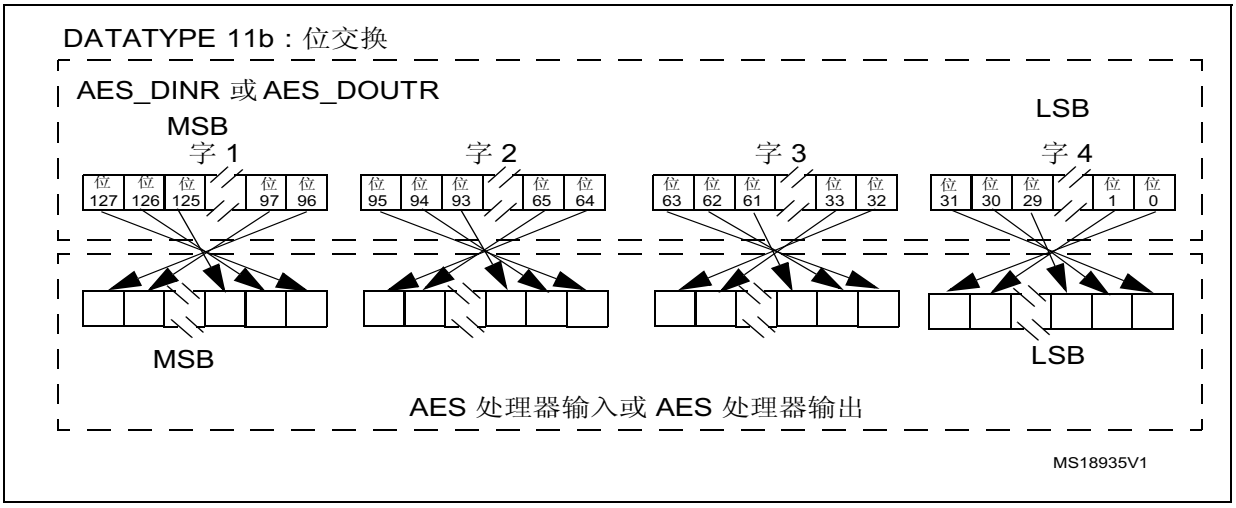


图 90. 根据数据类型构建 128 位块（续）

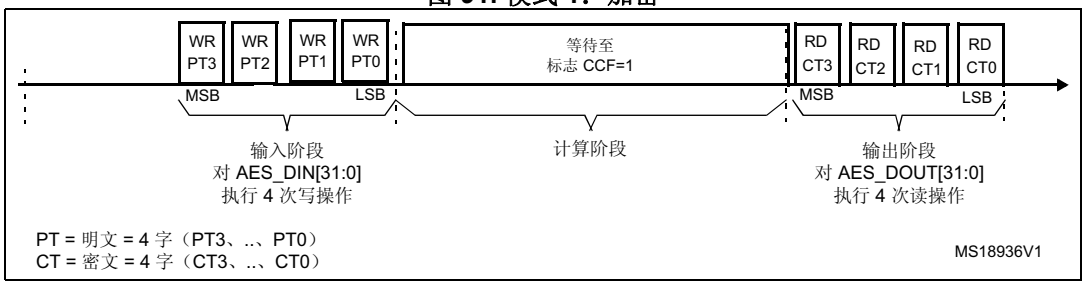


19.7 工作模式

19.7.1 模式 1：加密

1. 通过将 AES_CR 寄存器中的 EN 位置 0 禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=00 来配置模式 1，并通过对 CHMOD[1:0] 位进行编程来选择需要执行哪种类型的链接模式。
3. 如果选择 CTR 或 CBC 模式，则写入 AES_KEYRx 寄存器（128 位加密密钥）和 AES_IVRx 寄存器。如果选择 EBC 模式，则不使用 AES_IVRx 寄存器。
4. 通过将 AES_CR 寄存器中的 EN 位置 1 使能 AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入明文（MSB 优先），如*第 410 页的图 91：模式 1：加密*所示。
6. 等待 AES_SR 寄存器中的 CCF 标志置 1。
7. 对 AES_DOUTR 寄存器执行 4 次读取操作，以获取密文（MSB 优先），如*第 410 页的图 91：模式 1：加密*所示。
8. 重复执行步骤 5、6、7，处理使用相同加密密钥的所有块。

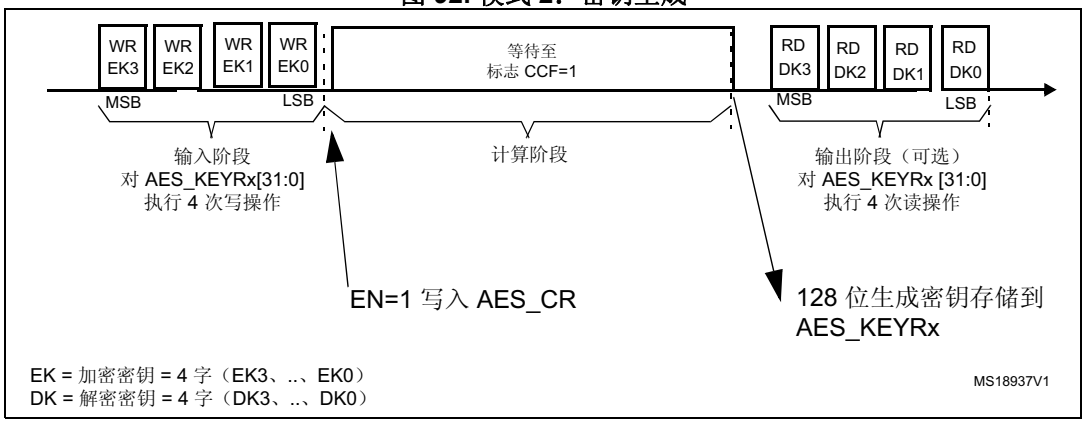
图 91. 模式 1：加密



19.7.2 模式 2：密钥生成

1. 通过将 AES_CR 寄存器中的 EN 位置 0 禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=01 配置模式 2。请注意，CHMOD[1:0] 位在这种情况下没有意义，因为该密钥生成模式与所选链接算法无关。
3. 将加密密钥写入到 AES_KEYRx 寄存器中，以获取生成密钥。写入到 AES_IVRx 中不起作用。
4. 通过将 AES_CR 寄存器中的 EN 位置 1 使能 AES。
5. 等待 AES_SR 寄存器中的 CCF 标志置 1。
6. 生成密钥会自动放入 AES_KEYRx 寄存器。如有需要，可读取 AES_KEYRx 寄存器获得解密密钥。AES 由硬件禁止。要重新开始计算生成密钥，请重复步骤 3、4、5 和 6。

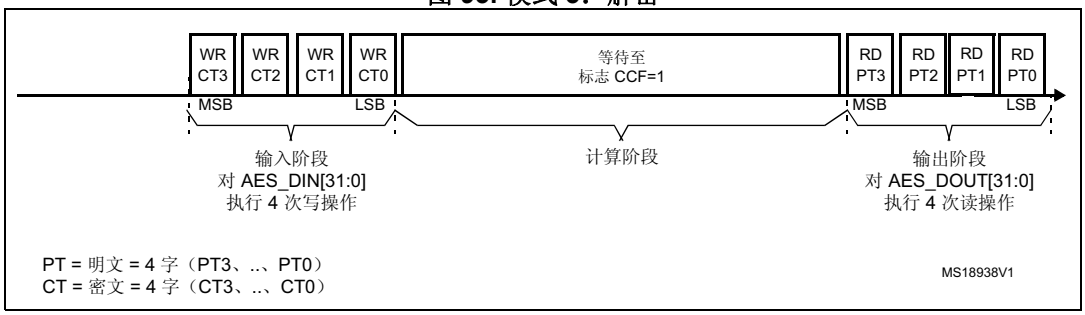
图 92. 模式 2：密钥生成



19.7.3 模式 3：解密

1. 通过将 AES_CR 寄存器中的 EN 位置 0 禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[3:0]=10 来配置模式 3，并通过对 CHMOD[1:0] 位进行编程来选择需要执行哪种类型的链接模式。
3. 将解密密钥写入 AES_KEYRx 寄存器（如果已使用模式 2：密钥生成将生成密钥存储到 AES_KEYRx 寄存器中，则可略过此步）。如果选择 CTR 或 CBC 模式，则写入 AES_IVRx 寄存器。如果选择 EBC 模式，则不使用 AES_IVRx 寄存器。
4. 通过将 AES_CR 寄存器中的 EN 位置 1 使能 AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如*第 412 页的图 93：模式 3：解密*所示。
6. 等待 AES_SR 寄存器中的 CCF 标志置 1。
7. 对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如*第 412 页的图 93：模式 3：解密*所示。
8. 重复执行步骤 5、6、7，处理使用相同加密密钥（存储在 AES_KEYRx 寄存器中）的块。

图 93. 模式 3：解密

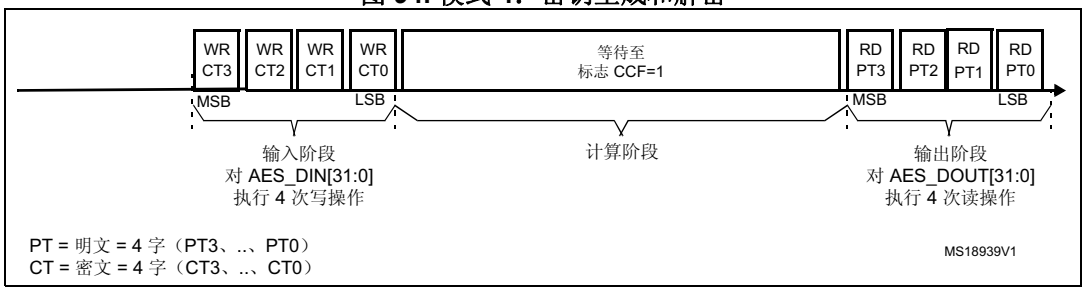


19.7.4 模式 4：密钥生成和解密

1. 通过将 AES_CR 寄存器中的 EN 位置 0 禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=11 配置模式 4。如果 AES 配置为 CTR 模式，则禁止使用此模式。如果软件写入 MODE[1:0] = 11 和 CHMOD[1:0] = 10，则会强制进入 CTR 解密模式。
3. 将加密密钥写入 AES_KEYRx 寄存器。如果选择 CBC 模式，则写入 AES_IVRx 寄存器。
4. 通过将 AES_CR 寄存器中的 EN 位置 1 使能 AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如*第 412 页的图 94：模式 4：密钥生成和解密*所示。
6. 等待 AES_SR 寄存器中的 CCF 标志置 1。
7. 对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如*第 412 页的图 94：模式 4：密钥生成和解密*所示。
8. 重复执行步骤 5、6、7，处理所有使用相同加密密钥的块。

注： *AES_KEYRx 寄存器包含处理过程中的所有加密密钥，不会有生成密钥存储在这些寄存器中。以加密密钥开始的生成密钥会存储在 AES 内部，不会在 AES_KEYRx 寄存器中存储副本。*

图 94. 模式 4：密钥生成和解密



19.8 AES DMA 接口

AES 加速器提供连接 DMA 控制器的接口。

必须对 DMA 进行配置才能传输字。

可将 AES 关联到两个不同的 DMA 请求通道：

- 输入的 DMA 请求通道：如果将 AES_CR 寄存器中的 DMAINEN 位置 1，每次 AES 要将字写入 AES_DINR 寄存器时，都会在 INPUT 阶段发起 DMA 请求 (AES_IN)。DMA 通道必须配置为存储器到外设模式，数据大小为 32 位。
- 输出的 DMA 请求通道：如果将 DMAOUTEN 位置 1，每次 AES 要从 AES_DOUTR 寄存器读取字时，都会在 OUTPUT 阶段发起 DMA 请求 (AES_OUT)。DMA 通道必须配置为外设到存储器模式，数据大小为 32 位。

会为每个阶段使能四个 DMA 请求，*图 95* 和 *图 96* 对此进行了介绍。

DMA 请求会一直产生，赶到 AES 被禁用。因此，128 位数据块处理结束时的数据输出阶段之后，AES 会自动切换到下一个数据块的新数据输入阶段（如果存在）。

注： *对于模式 2（密钥生成），可通过软件使用 CPU 访问 AES_KEYRx 寄存器。不会为此提供 DMA 通道。因此，AES_CR 寄存器中的 DMAINEN 位和 DMAOUTEN 位在该模式下不起作用。*

当 DMAOUTEN = 1 时，CCF 标志不起作用，这种情况下软件不需要读取该标志。如果应用需要禁止 AES 来取消 DMA 管理并为数据输入或数据输出阶段使用 CPU 访问，此位可能保持为 1，必须由软件清零。

图 95. 输入阶段的 DMA 请求和数据传输 (AES_IN)

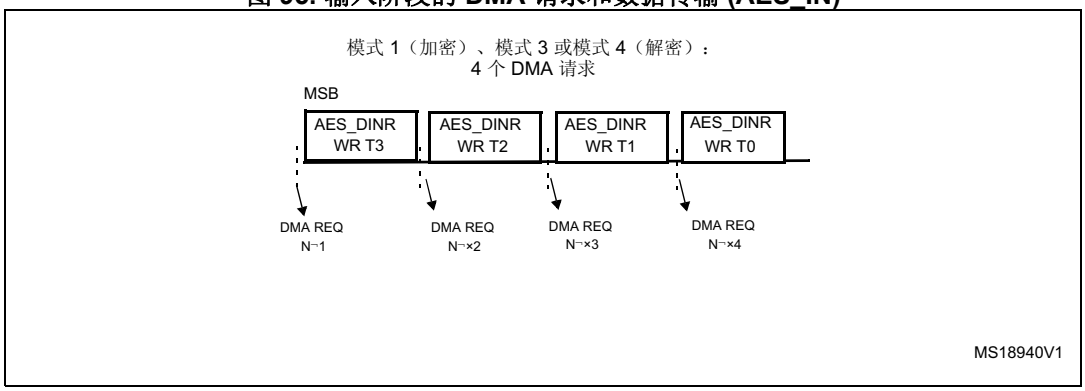
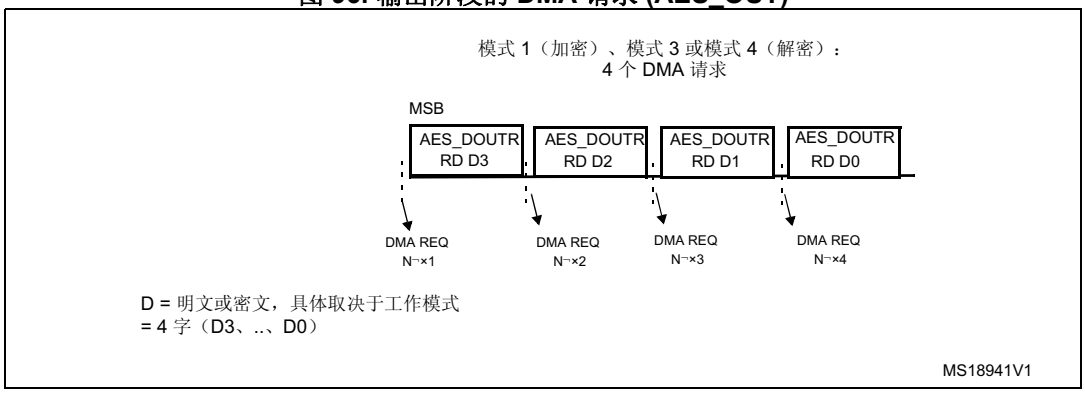


图 96. 输出阶段的 DMA 请求 (AES_OUT)



19.9 错误标志

如果在计算阶段或输入阶段检测到未预期的读取操作，则会将 AES_SR 寄存器中的 RDERR 标志置 1。

如果在输出阶段或计算阶段检测到未预期的写入操作，则会将 AES_SR 寄存器中的 WRERR 标志置 1。

可通过将 AES_CR 寄存器中的相应位置 1 的方式将标志清零（CCFC 位可将 CCF 标志清零，ERRC 位可将 WERR 和 RDERR 标志清零）。

如果之前已将 AES_CR 寄存器中的 ERRIE 位置 1，则当其中一个错误标志置 1 时，可生成中断。

如果检测到错误，则 AES 不会被硬件禁止并继续照常进行处理。

19.10 处理时间

下表列出了每种工作模式下处理 128 位块需要的时间。

表 83. 处理时间（时钟周期数）

工作模式	输入阶段	计算阶段	输出阶段	总计
模式 1：加密	8	202	4	214
模式 2：密钥生成	-	80	-	80
模式 3：解密	8	202	4	214
模式 4：密钥生成 + 解密	8	276	4	288

19.11 AES 中断

表 84. AES 中断请求

中断事件	事件标志	使能控制位	退出等待状态
AES 计算完成标志	CCF	CCFIE	是
AES 读取错误标志	RDERR	ERRIE	是
AES 写入错误标志	WRERR	ERRIE	是

19.12 AES 寄存器

19.12.1 AES 控制寄存器 (AES_CR)

AES control register

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															
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			DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:13 保留，读为 0

位 12 **DMAOUTEN**：数据输出阶段使能 DMA 管理 (Enable DMA management of data output phase)

0：禁止 DMA（在数据输出阶段）

1：使能 DMA（在数据输出阶段）

如果 DMAOUTEN 位置 1，则会在模式 1、3 或 4 下为输出数据阶段生成 DMA 请求。此位在模式 2（密钥生成）下不起作用。

位 11 **DMAOUTEN**：数据输入阶段使能 DMA 管理 (Enable DMA management of data input phase)

0：禁止 DMA（在数据输入阶段）

1：使能 DMA（在数据输入阶段）

如果 DMAINEN 位置 1，则会在模式 1、3 或 4 下为输入数据阶段生成 DMA 请求。此位在模式 2（密钥生成）下不起作用。

位 10 **ERRIE**：错误中断使能 (Error interrupt enable)

如果 RDERR 或 WRERR 中至少有一个标志置 1，则会生成中断。

0：禁止错误中断

1：使能错误中断

位 9 **CCFIE**：CCF 标志中断使能 (CCF flag interrupt enable)

如果 CCF 标志置 1，则会生成中断。

0：禁止 CCF 中断

1：使能 CCF 中断

位 8 **ERRC**：错误清零 (Error clear)

向该位写入 1 会将 RDERR 和 WRERR 标志清零。

此位始终读为 0。

位 7 **CCFC**：计算完成标志清零 (Computation Complete Flag Clear)

向该位写入 1 时会将 CCF 标志清零。

此位始终读为 0。

位 6:5 **CHMOD[1:0]**：AES 链接模式 (AES chaining mode)

00：电子密码本 (ECB)
01：密码块链接 (CBC)
10：计数器模式 (CTR)
11：保留。
仅可在 AES 已禁用的情况下更改 AES 链接模式。禁止在 AES 已使能的情况下写入这些位，以免出现不可预期的 AES 行为。

位 4:3 **MODE[1:0]**：AES 工作模式 (AES operating mode)

00：模式 1：加密
01：模式 2：密钥生成
10：模式 3：解密
11：模式 4：密钥生成 + 解密
仅可在 AES 已禁用的情况下更改工作模式。禁止在 AES 已使能的情况下写入这些位，以免出现不可预期的 AES 行为。
如果选择 CTR 模式，则禁用模式 4。但是，如果软件尝试为该 CTR 模式配置设置模式 4，则会强制进入模式 3。

位 2:1 **DATATYPE[1:0]**：数据类型选择 (Data type selection)（用于数据进/出加密块）

00：32 位数据。无交换。
01：16 位数据或半字。在字中交换每个半字。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0x56AB7643。
10：8 位数据或字节。在字中交换所有字节。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0xAB564376。
11：位数据在字中交换所有位。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0xD56AC26E。
仅可在 AES 已禁用的情况下更改数据类型选择。禁止在 AES 已使能的情况下写入这些位，以免出现不可预期的 AES 行为。

位 0 **EN**：AES 使能 (USART enable)

0：禁止 AES
1：使能 AES
可随时通过将此位清零的方式重新初始化 AES：EN 置 1 后，AES 准备好开始处理新块。
如果在模式 2（密钥生成）下完成 AES 计算，则由硬件清零此位。

19.12.2 AES 状态寄存器 (AES_SR)

AES status register

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															
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													WRERR	RDERR	CCF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:3 保留，读为 0

位 2 **WRERR**：写入错误标志 (Write error flag)

如果检测到 AES_DINR 寄存器发生了未预期的写入操作（计算阶段或数据输出阶段），则由硬件将此位置 1。如果 AES_CR 寄存器中的 ERRIE 位之前已置 1，将生成中断。此标志对 AES 无影响，即使 WRERR 置 1，AES 也会继续运行。

通过将 AES_CR 寄存器中的 ERRC 位置 1，此位由软件清零。

0：未检测到写入错误

1：检测到写入错误

位 1 **RDERR**：读取错误标志 (Read error flag)

如果检测到 AES_DOUTR 寄存器发生了未预期的读取操作（计算阶段或数据输入阶段），则会通过硬件将此位置 1。如果 AES_CR 寄存器中的 ERRIE 位之前已置 1，将生成中断。此标志对 AES 无影响，即使 RDERR 置 1，AES 也会继续运行。

通过将 AES_CR 寄存器中的 ERRC 位置 1，此位由软件清零。

0：未检测到读取错误

1：检测到读取错误

位 0 **CCF**：计算完成标志 (Computation complete flag)

当计算完成时，该位由硬件置 1。如果 AES_CR 寄存器中的 CCFIE 位之前已置 1，将生成中断。

通过向 AES_CR 寄存器中的 CCFC 位写入 1，此位由软件清零。

0：计算未完成

1：计算已完成

注： 该位仅在 DMAOUTEN = 0 时才有效。DMA_EN = 1 时，该位可能保持为 1。

19.12.3 AES 数据输入寄存器 (AES_DINR)

AES data input register

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
DINR[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
DINR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **DINR[31:0]**：数据输入寄存器 (Data Input Register)

输入阶段必须对该寄存器执行 4 次写入操作：

– 在模式 1（加密）下，必须从 MSB 到 LSB 写入 4 个代表明文的字。

– 在模式 2（密钥生成）下，不会使用该寄存器，因为该模式仅与从 AES_KEYRx 寄存器开始的生成密钥计算有关。

– 在模式 3（解密）和模式 4（密钥生成+解密）下，必须从 MSB 到 LSB 写入 4 个代表密文的字。

注： 该寄存器必须通过 32 位数据宽度进行访问。

19.12.4 AES 数据输出寄存器 (AES_DOUTR)

AES data output register

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
DOUTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:0 **DOUTR[31:0]**：数据输出寄存器 (Data output register)。

该寄存器为只读。

将 CCF 标志（计算完成标志）置 1 后，对该数据寄存器读取 4 次可得到 128 位输出结果：

- 在模式 1（加密）下，必须从 MSB 到 LSB 读取 4 个代表密码文本的字。
- 在模式 2（密钥生成）下，不需要读取该寄存器，因为生成密钥位于 AES_KEYRx 寄存器中。
- 在模式 3（加密）和模式 4（密钥生成+解密）下，读取的 4 个字代表从 MSB 到 LSB的明文。

注： 该寄存器必须通过 32 位数据宽度进行访问。

19.12.5 AES 密钥寄存器 0 (AES_KEYR0) (LSB: key [31:0])

AES key register 0

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
KEYR0[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
KEYR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **KEYR0[31:0]**：AES 密钥寄存器 (AES key register) (LSB key [31:0])

必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。

在模式 1（加密）、模式 2（密钥生成）和模式 4（密钥生成 + 解密）下，要写入的值代表从 LSB 开始的加密密钥，也就是 Key [31:0]。

在模式 3（解密）下，要写入的值代表从 LSB 开始的解密密钥，也就是 Key [31:0]。如果在该解密模式下向寄存器写入加密密钥，则在 AES 使能前读取该寄存器将返回加密值。在 CCF 标志置 1 之后读取该寄存器将返回生成密钥。

在 AES 已使能时读取该寄存器会返回不可预测的值。

注： 在模式 4（生成密钥 + 解密）下，该寄存器不包含生成密钥，但始终包含加密密钥值。

19.12.6 AES 密钥寄存器 1 (AES_KEYR1) (Key[63:32])

AES key register 1

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR1[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
KEYR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **KEYR1[31:0]**：AES 密钥寄存器 (AES key register) (key [63:32])

请参见 AES_KEYR0 的说明。

19.12.7 AES 密钥寄存器 2 (AES_KEYR2) (Key [95:64])

AES key register 2

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR2[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
KEYR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **KEYR2[31:0]**：AES 密钥寄存器 (AES key register) (key [95:64])
请参见 AES_KEYR0 的说明。

19.12.8 AES 密钥寄存器 3 (AES_KEYR3) (MSB: key[127:96])

AES key register 3

偏移地址：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR3[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
KEYR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **KEYR3[31:0]**：AES 密钥寄存器 (AES key register) (MSB key [127:96])
请参见 AES_KEYR0 的说明。

19.12.9 AES 初始化向量寄存器 0 (AES_IVR0)（LSB：IVR[31:0]）

AES initialization vector register 0

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR0[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
IVR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **IVR0[31:0]**：初始化向量寄存器 (initialization vector register) (LSB IVR [31:0])

必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。

下列情况下，寄存器值没有意义：

- 选择了 EBC 模式（电子密码本）。
- 在密钥生成的基础上额外选择了 CTR 或 CBC 模式。

在 CTR 模式（计数器模式）下，该寄存器包含 32 位计数器值。

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

19.12.10 AES 初始化向量寄存器 1 (AES_IVR1) (IVR[63:32])

AES initialization vector register 1

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR1[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
IVR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **IVR1[31:0]**：初始化向量寄存器 (Initialization Vector Register) (IVR [63:32])

必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。

下列情况下，寄存器值没有意义：

- 选择了 EBC 模式（电子密码本）。
- 在密钥生成或密钥生成+解密模式的基础上额外选择了 CTR 或 CBC 模式。

在 CTR 模式（计数器模式）下，该寄存器包含随机值。

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

19.12.11 AES 初始化向量寄存器 2 (AES_IVR2) (IVR[95:64])

AES initialization vector register 2

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR2[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
IVR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **IVR2[31:0]**：初始化向量寄存器 (Initialization Vector Register) (IVR [95:64])

必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。

下列情况下，寄存器值没有意义：

- 选择了 EBC 模式（电子密码本）。
- 在密钥生成或密钥生成+解密模式的基础上额外选择了 CTR 或 CBC 模式。

在 CTR 模式（计数器模式）下，该寄存器包含随机值。

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

19.12.12 AES 初始化向量寄存器 3 (AES_IVR3) (MSB: IVR[127:96])

AES initialization vector register 3

偏移地址：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR3[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
IVR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:0 **IVR3[31:0]**：初始化向量寄存器 (initialization vector register) (MSB IVR [127:96])

必须在将 AES_CR 寄存器中的 EN 位置 1 之前写入该寄存器。

下列情况下，寄存器值没有意义：

- 选择了 EBC 模式（电子密码本）。
- 在密钥生成或密钥生成+解密模式的基础上额外选择了 CTR 或 CBC 模式。

在 CTR 模式（计数器模式）下，该寄存器包含随机值。

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

19.12.13 AES 寄存器映射

表 85. AES 寄存器映射

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**AES_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0004	**AES_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRERR	RDERR	CCF
	Reset value																														0	0	0
0x0008	**AES_DINR**	AES_DINR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	**AES_DOUTR**	AES_DOUTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	**AES_KEYR0**	AES_KEYR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**AES_KEYR1**	AES_KEYR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**AES_KEYR2**	AES_KEYR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	**AES_KEYR3**	AES_KEYR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	**AES_IVR0**	AES_IVR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	**AES_IVR1**	AES_IVR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	**AES_IVR2**	AES_IVR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**AES_IVR3**	AES_IVR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

20 随机数发生器 (RNG)

20.1 简介

RNG 处理器是一个以连续模拟噪声为基础的随机数发生器，在主机读数时提供一个 32 位的随机数。

RNG 已通过 FIPS PUB 140-2（2001 年 10 月 10 日）测试，成功率达 99%。

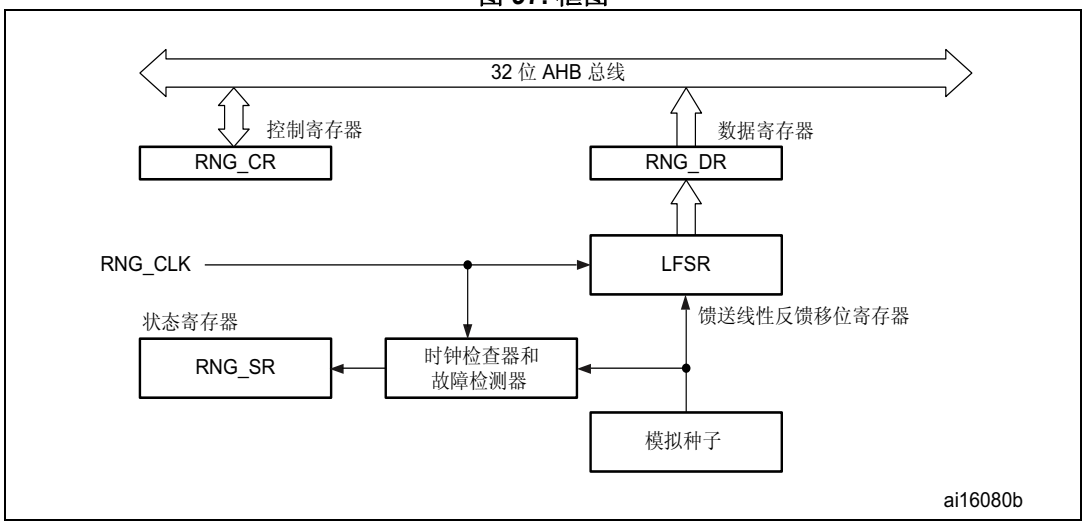
20.2 RNG 主要特性

- 提供由模拟量发生器产生的 32 位随机数
- 两个连续随机数的间隔为 40 个 RNG_CLK 时钟信号周期
- 通过监视 RNG 熵来标识异常行为（产生稳定值，或产生稳定的值序列）
- 可被禁止以降低功耗

20.3 RNG 功能说明

图 97 显示了 RNG 框图。

图 97. 框图



1. 有关 RNG时钟 (RNG_CLK) 源的更多详细信息，请参见*第 7 节：复位和时钟控制 (RCC)*。

随机数发生器采用模拟电路实现。此电路产生馈入线性反馈移位寄存器 (RNG_LFSR) 的种子，用于生成 32 位随机数。

该模拟电路由几个环形振荡器组成，振荡器的输出进行异或运算以产生种子。RNG_LFSR 由专用时钟 (RNG_CLK) 按恒定频率提供时钟信息，因此随机数质量与 HCLK 频率无关。当将大量种子引入 RNG_LFSR 后，RNG_LFSR 的内容会传入数据寄存器 (RNG_DR)。

同时，系统会监视模拟种子和专用时钟 RNG_CLK。状态位（RNG_SR 寄存器中）指示何时在种子上出现异常序列，或指示何时 RNG_CLK 时钟频率过低。检测到错误时生成中断。

20.3.1 操作

要运行 RNG，请按以下步骤操作：

1. 如果需要，使能中断（为此，将 RNG_CR 寄存器中的 IE 位置 1）。准备好随机数时或出现错误时生成中断。
2. 通过将 RNG_CR 寄存器中的 RNGEN 位置 1 使能随机数产生。这会激活模拟部分、RNG_LFSR 和错误检测器。
3. 每次中断时，检查确认未出现错误（RNG_SR 寄存器中的 SEIS 和 CEIS 位应为 0），并且随机数已准备就绪（RNG_SR 寄存器中的 DRDY 位为 1）。然后即可读取 RNG_DR 寄存器中的内容。

按照 FIPS PUB（联邦信息处理标准出版物）140-2 的要求，将 RNGEN 位置 1 后产生的第一个随机数不应使用，但应保存起来，与产生的下一个随机数进行比较。随后产生的每个随机数都需要与产生的上一个随机数进行比较。如果任何一对进行比较的数字相等，则测试失败（连续随机数发生器测试）。

20.3.2 错误管理

如果 CEIS 位的值为 1（时钟错误）

出现时钟错误时，RNG 无法再产生随机数，因为 RNG_CLK 时钟不正确。检查时钟控制器是否正确配置，是否可提供 RNG 时钟，然后将 CEIS 位清零。当 CECS 位为 0 时，RNG 可正常工作。时钟错误对产生的上一个随机数没有影响，因此 RNG_DR 寄存器内容可以使用。

如果 SEIS 位的值为 1（种子错误）

出现种子错误时，只要 SECS 位为 1，就会中断随机数产生。如果 RNG_DR 寄存器中有可用随机数，不能使用该随机数，因为它可能没有足够的熵。

应执行以下操作：将 SEIS 位清零，然后将 RNGEN 位清零并置 1，以便重新初始化和重新启动 RNG。

20.4 RNG 寄存器

RNG 与控制寄存器、数据寄存器和状态寄存器相关联。必须通过字（32 位）进行访问。

20.4.1 RNG 控制寄存器 (RNG_CR)

RNG control register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IE	RNGEN	Res.	Res.
												rw	rw		

位 31:4 保留，必须保持复位值

位 3 **IE**：中断使能 (Interrupt enable)

0：禁止 RNG 中断

1：使能 RNG 中断。只要 RNG_SR 寄存器中 DRDY=1 或 SEIS=1 或 CEIS=1，就会挂起中断。

位 2 **RNGEN**：随机数发生器使能 (Random number generator enable)

0：禁止随机数发生器

1：使能随机数发生器。

位 1:0 保留，必须保持复位值

20.4.2 RNG 状态寄存器 (RNG_SR)

RNG status register

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

位 31:7 保留，必须保持复位值

位 6 **SEIS**：种子错误中断状态 (Seed error interrupt status)

此位与 SECS 同时设置，通过向其写入 0 来清零。

0：未检测到错误序列

1：检测到以下错误序列之一：

– 超过 64 个连续位具有相同值（0 或 1）

– 超过 32 个连续交替的 0 和 1 (0101010101...01)

如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

位 5 **CEIS**：时钟错误中断状态 (Clock error interrupt status)

此位与 CECS 同时设置，通过向其写入 0 来清零。

0：正确检测到 RNG_CLK 时钟

1：未正确检测到 RNG_CLK 时钟 ($f_{RNG_CLK} < f_{HCLK}/16$)

如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

位 4:3 保留，必须保持复位值

位 2 **SECS**：种子错误当前状态 (Seed error current status)

0：目前未检测到错误序列。如果 SEIS 位置 1，则意味着已检测到错误序列并已恢复正常。

1：检测到以下错误序列之一：

– 超过 64 个连续位具有相同值（0 或 1）

– 超过 32 个连续交替的 0 和 1 (0101010101...01)

位 1 **CECS**：时钟错误当前状态 (Clock error current status)

0：正确检测到 RNG_CLK 时钟。如果 CEIS 位置 1，则意味着已检测到时钟错误并已恢复正常。

1：未正确检测到 RNG_CLK 时钟 ($f_{RNG_CLK} < f_{HCLK}/16$)。

位 0 **DRDY**：数据就绪 (Data ready)

0：RNG_DR 寄存器尚未有效，无可用随机数据

1：RNG_DR 寄存器包含有效随机数据

注： 如果 RNG_CR 寄存器中 IE = 1，则会挂起中断。

读取 RNG_DR 寄存器后，此位恢复到 0，直到计算出新的有效值。

20.4.3 RNG 数据寄存器 (RNG_DR)

RNG data register

偏移地址：0x08

复位值：0x0000 0000

RNG_DR 寄存器是只读寄存器，在读取时提供 32 位随机数值。读取后，此寄存器在最多 40 个 RNG_CLK 时钟周期后，提供新的随机数值。在读取 RNDATA 值之前，软件必须检查 DRDY 位是否已置 1。

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

位 31:0 **RNDATA：**随机数据 (Random data)

32 位随机数据。

20.4.4 RNG 寄存器映射

表 86 给出了 RNG 寄存器映射和复位值。

表 86. RNG 寄存器映射和复位映射

偏移	寄存器名称 复位值	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	**RNG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IE	RNGEN	Res.	Res.
	Reset value																													0	0		
0x04	**RNG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x08	**RNG_DR**	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

21 通用定时器 (TIM2/TIM3)

21.1 TIM2/TIM3 简介

通用定时器包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

它们可用于多种用途，包括测量输入信号的脉冲宽度（*输入捕获*）或生成输出波形（*输出比较和 PWM*）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

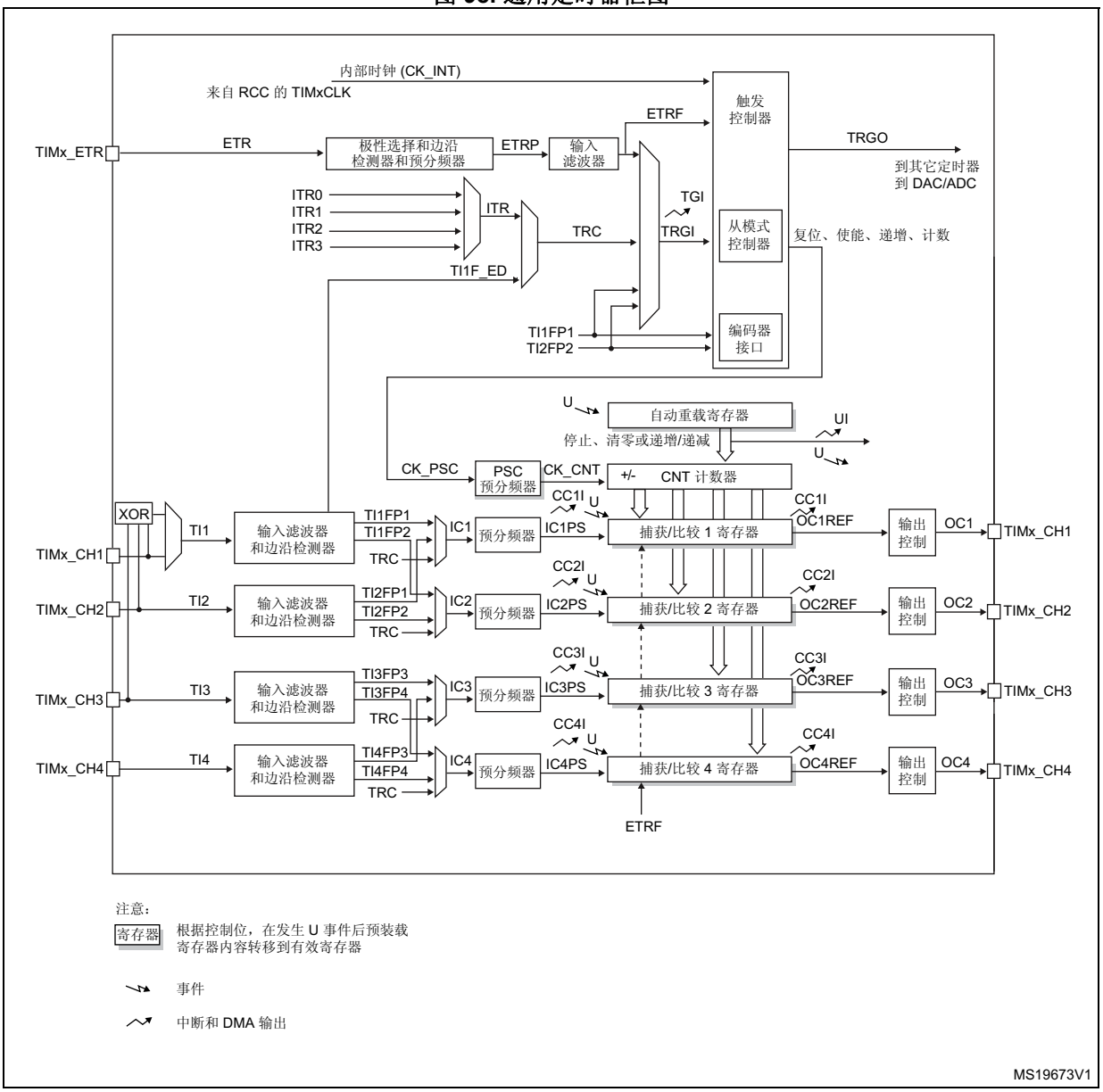
这些定时器彼此完全独立，不共享任何资源。如*第 21.3.15 节*中所述，它们可以同步操作。

21.2 TIM2/TIM3 主要特性

通用 TIMx 定时器具有以下特性：

- 16 位 (TIM2/3) 递增、递减、递增/递减自动重载计数器。
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 到 65535 之间。
- 多达 4 个独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿和中心对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器且可实现多个定时器互连的同步电路。
- 发生如下事件时生成中断/DMA 请求：
  - 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持定位用增量（正交）编码器和霍尔传感器电路
- 外部时钟触发输入或逐周期电流管理

图 98. 通用定时器框图


内部时钟 (CK_INT)
来自 RCC 的 TIMxCLK
TIMx_ETR
ETR
极性选择和边沿检测器和预分频器
ETRP
输入滤波器
ETRF
触发控制器
TRGO
到其它定时器到 DAC/ADC
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
从模式控制器
复位、使能、递增、计数
TI1F_ED
TI1FP1
TI2FP2
编码器接口
U
自动重载寄存器
UI
停止、清零或递增/递减
CK_PSC
PSC 预分频器
CK_CNT
+/-
CNT 计数器
XOR
TI1
输入滤波器和边沿检测器
TI1FP1
TI1FP2
IC1
预分频器
IC1PS
CC1I
捕获/比较 1 寄存器
CC1I
OC1REF
输出控制
OC1
TIMx_CH1
TIMx_CH2
TI2
TI2FP1
TI2FP2
IC2
IC2PS
CC2I
捕获/比较 2 寄存器
OC2REF
OC2
TIMx_CH3
TI3
TI3FP3
TI3FP4
IC3
IC3PS
CC3I
捕获/比较 3 寄存器
OC3REF
OC3
TIMx_CH4
TI4
TI4FP3
TI4FP4
IC4
IC4PS
CC4I
捕获/比较 4 寄存器
OC4REF
OC4
ETRF
注意：
寄存器
根据控制位，在发生 U 事件后预装载寄存器内容转移到有效寄存器
事件
中断和 DMA 输出
MS19673V1

21.3 TIM2/TIM3 功能说明

21.3.1 时基单元

可编程定时器的主要模块由一个 16 位及其相关的自动重载寄存器组成。计数器可递增计数、递减计数或同时递增和递减计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)：
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 (UEV) 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值（或者在递减计数时达到下溢值）并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

请注意，实际的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 16 位/32 位寄存器（TIMx_PSC 寄存器）所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 99 和*图 21.3.2* 以一些示例说明在预分频比实时变化时计数器的行为：

图 99. 预分频器分频由 1 变为 2 时的计数器时序图

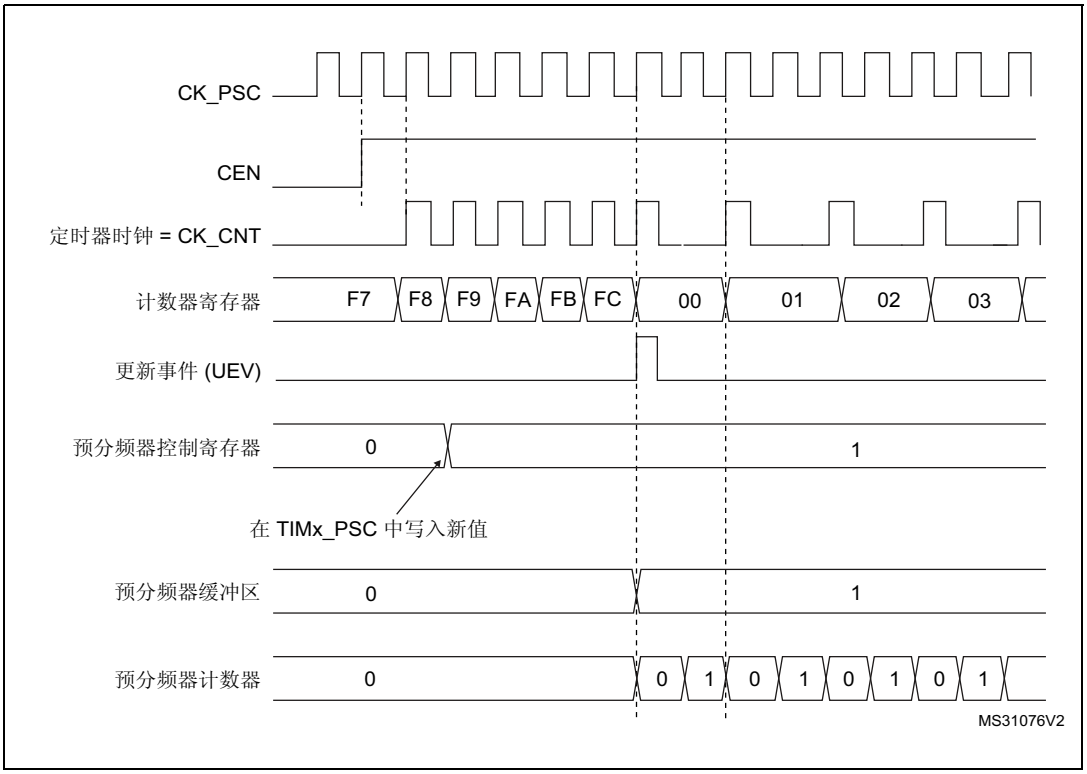
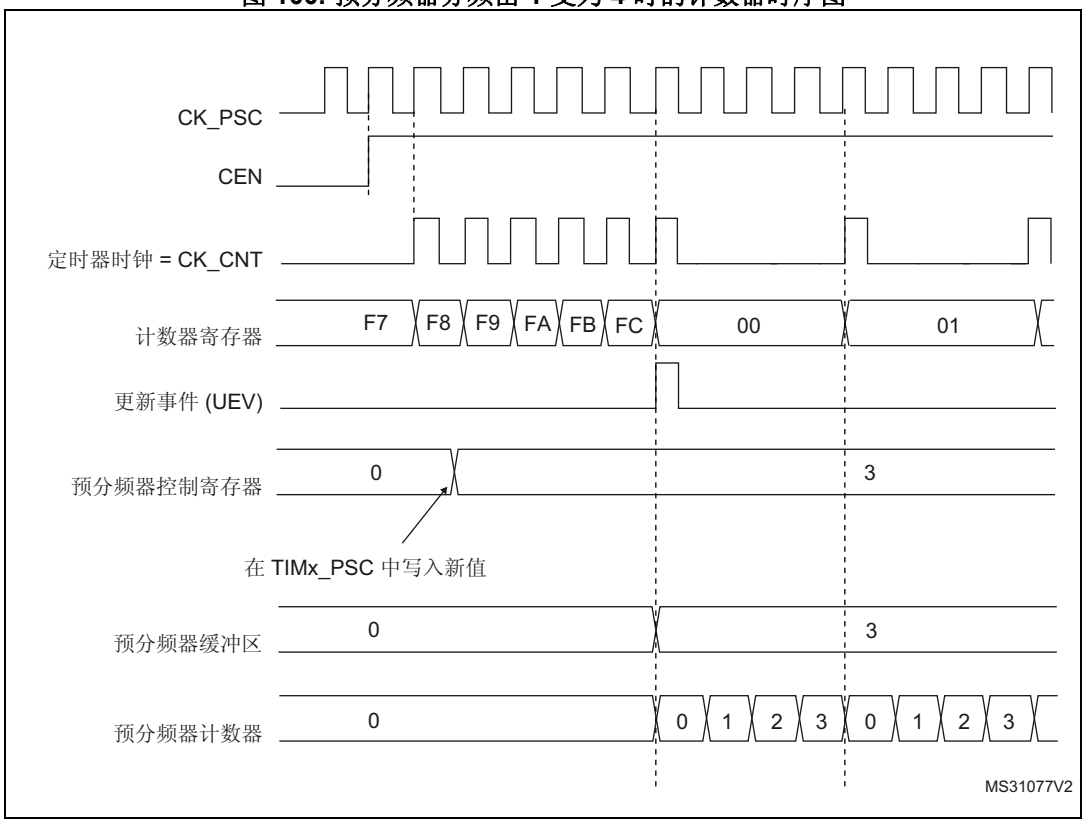


图 100. 预分频器分频由 1 变为 4 时的计数器时序图



21.3.2 计数器模式

递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 使用预装载值 (TIMx_ARR) 更新自动重载影子寄存器

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 101. 计数器时序图，1 分频内部时钟

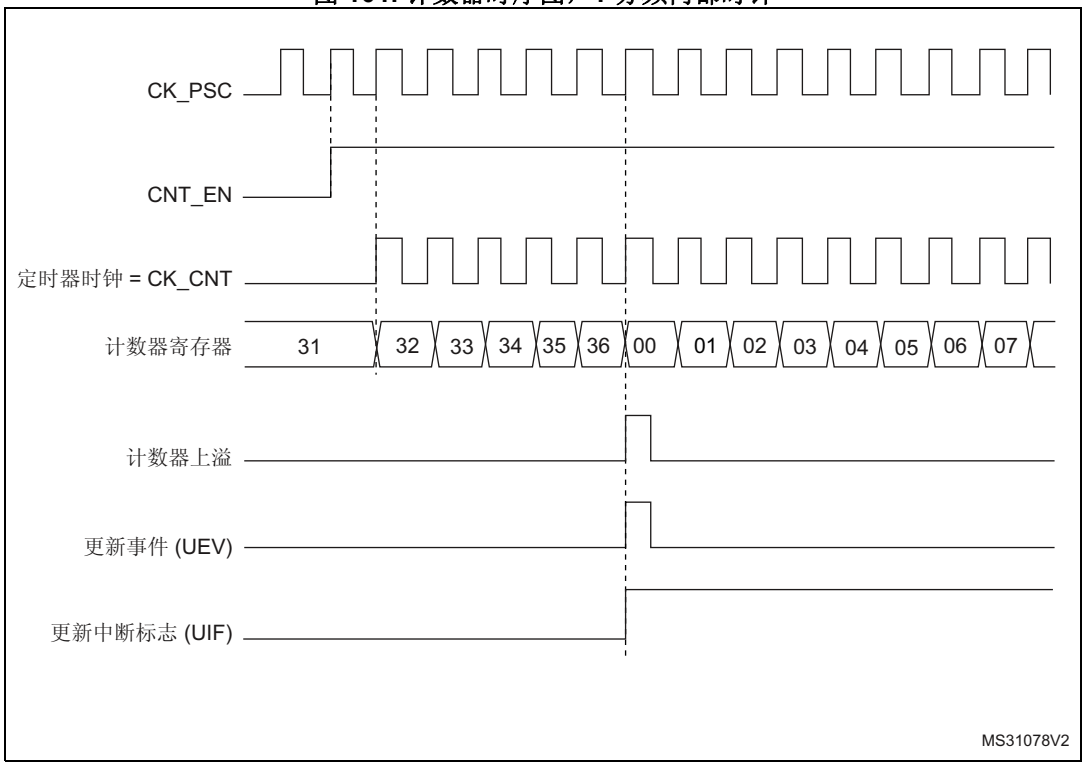


图 102. 计数器时序图，2 分频内部时钟

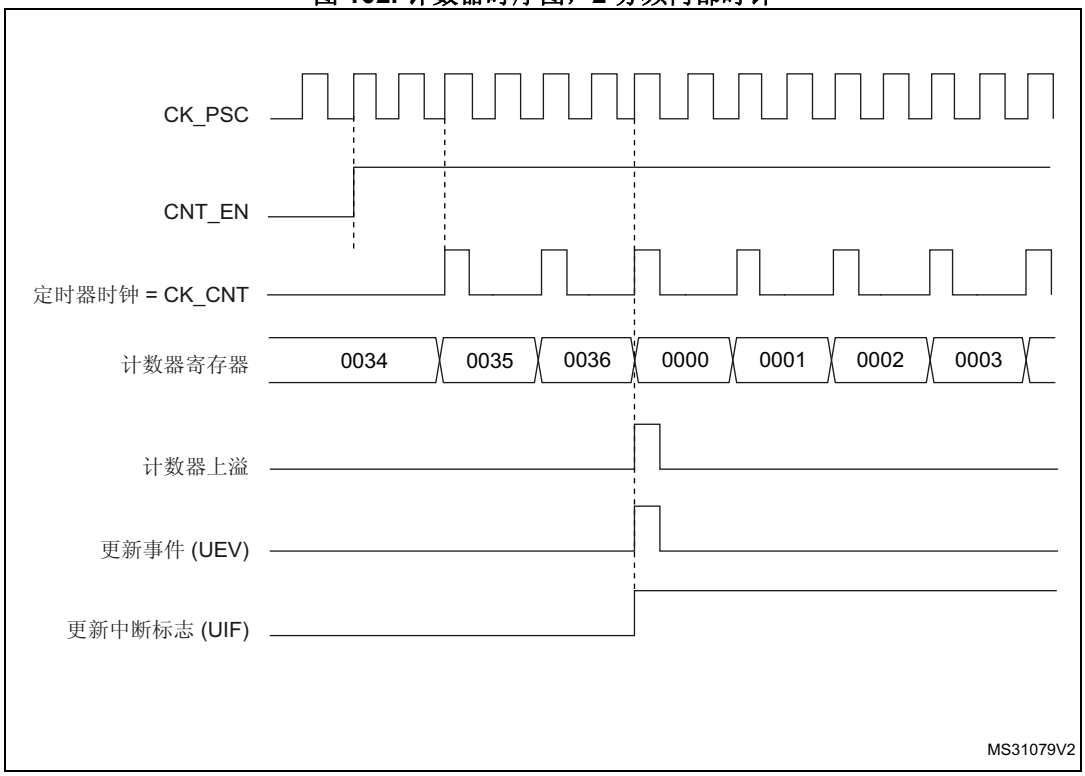


图 103. 计数器时序图，4 分频内部时钟

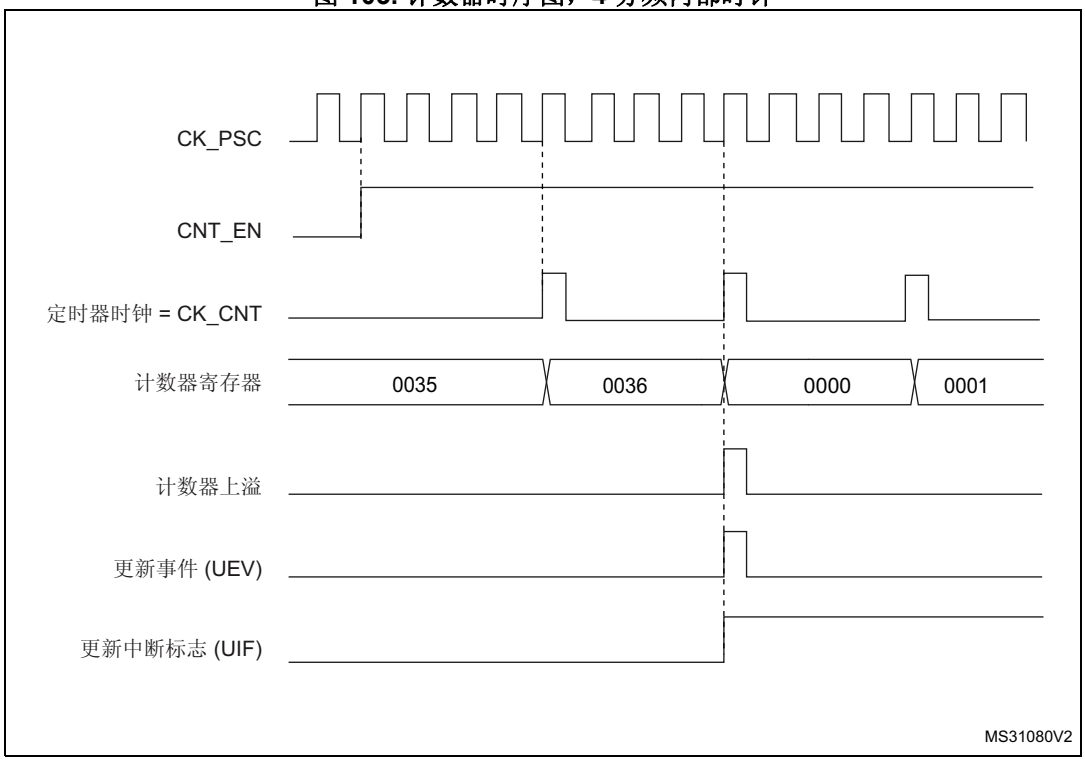


图 104. 计数器时序图，N 分频内部时钟

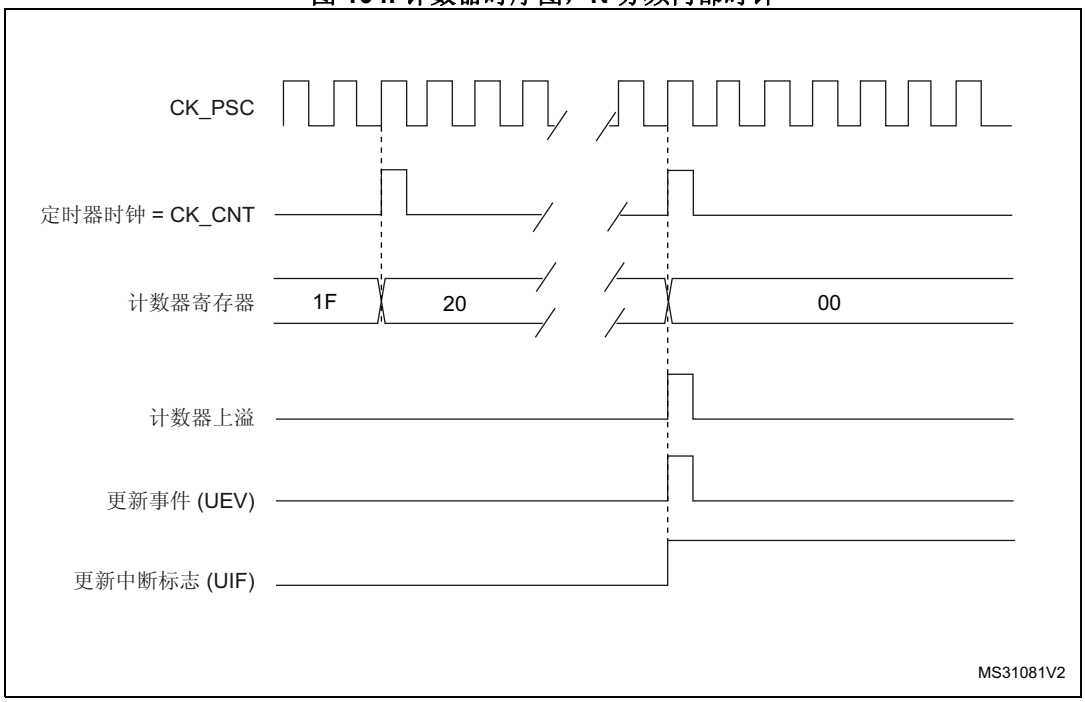


图 105. 计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载）

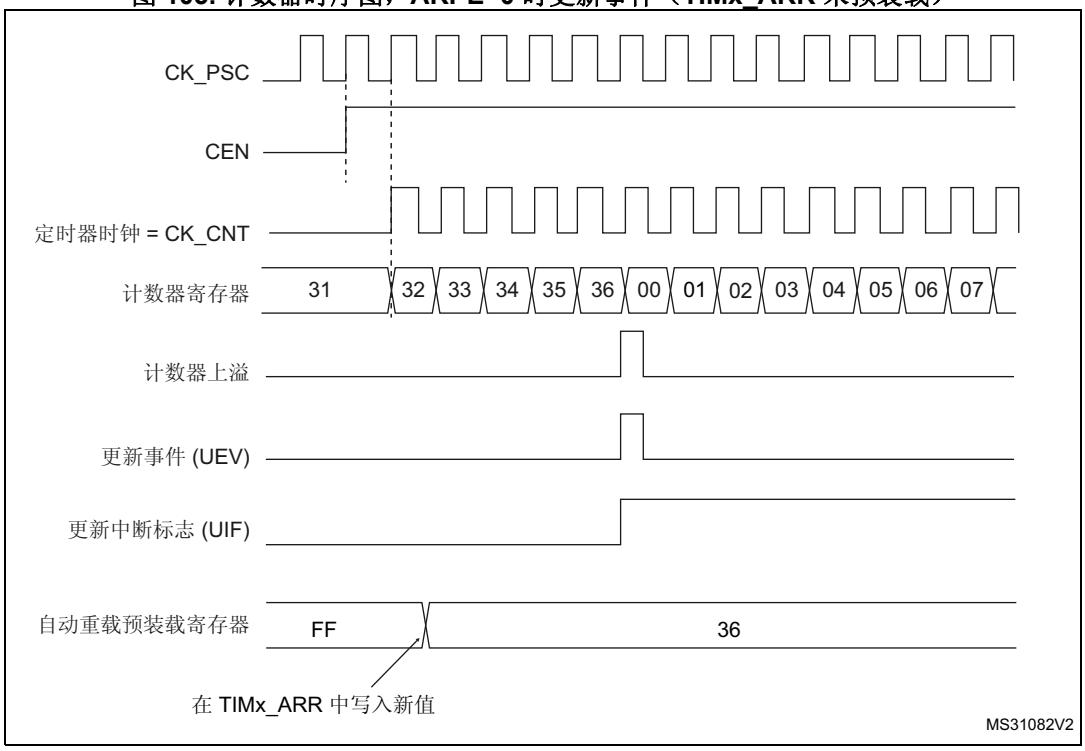
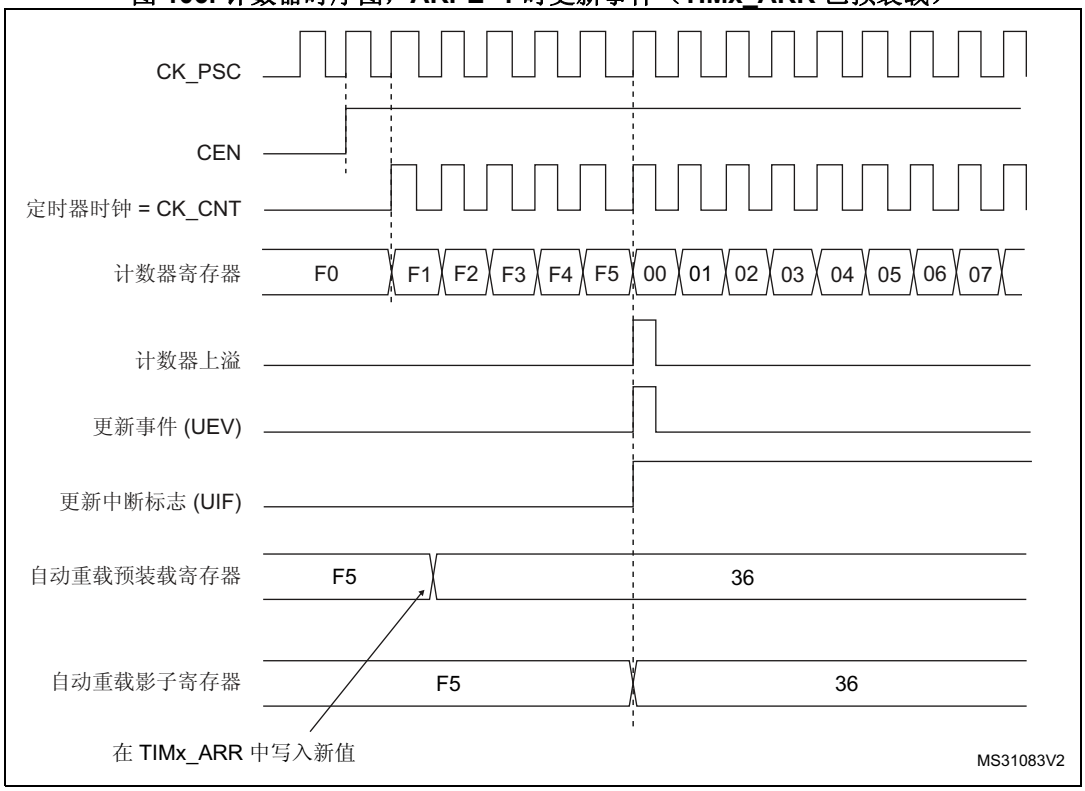


图 106. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 已预装载）



递减计数模式

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

每次发生计数器下溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，ARR 寄存器更新在计数器重载之前被更新，因此下一个周期就是预期的值。

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 107. 计数器时序图，1 分频内部时钟

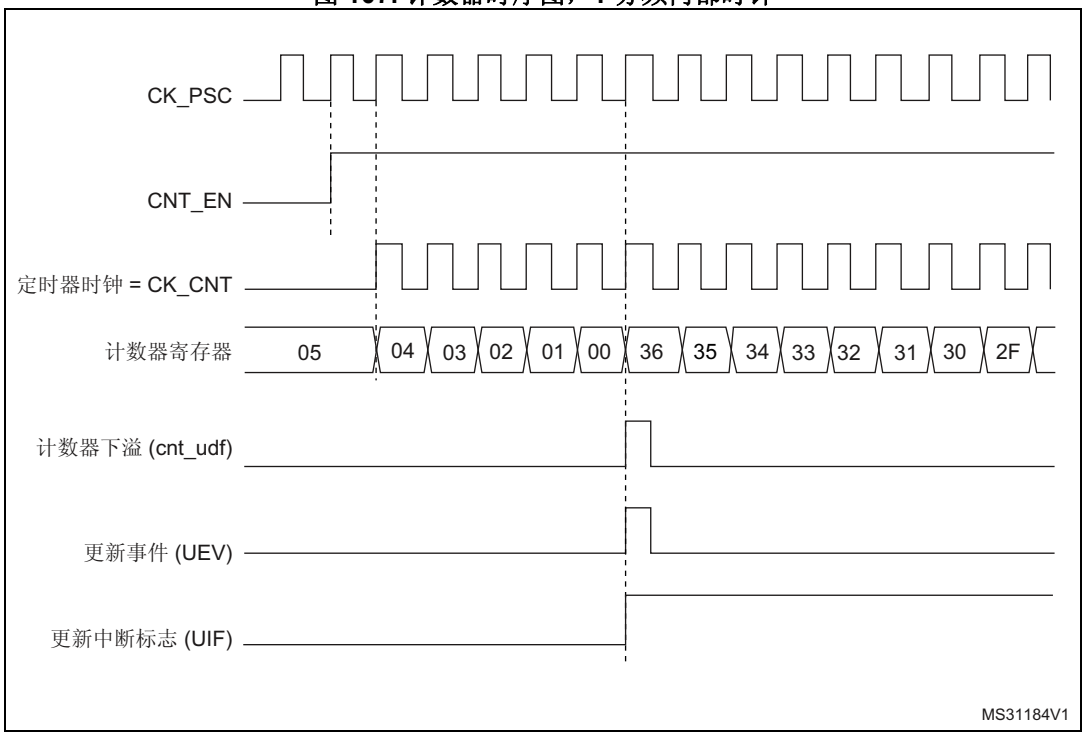


图 108. 计数器时序图，2 分频内部时钟

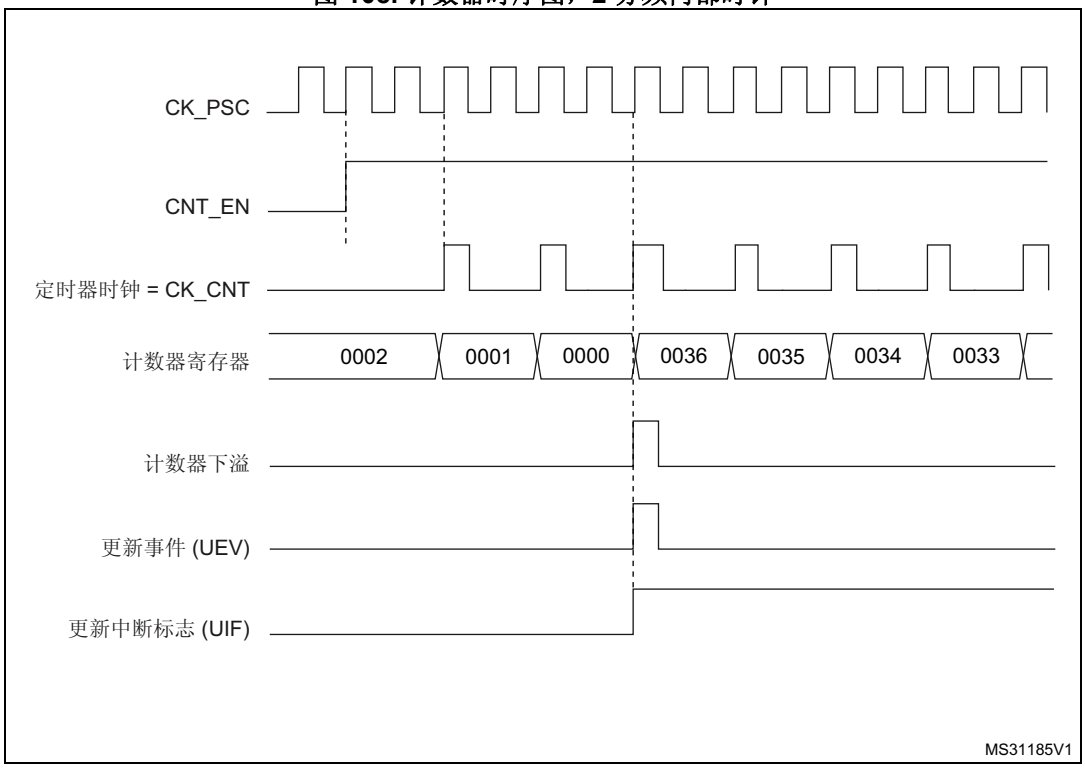


图 109. 计数器时序图，4 分频内部时钟

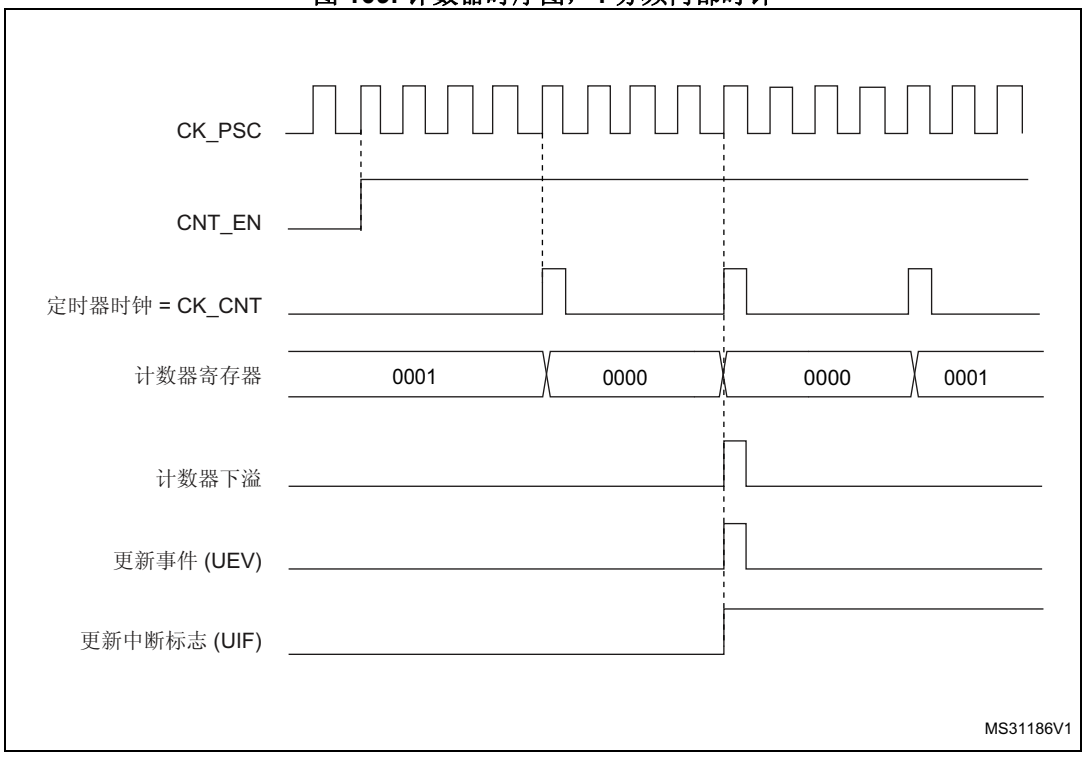


图 110. 计数器时序图，N 分频内部时钟

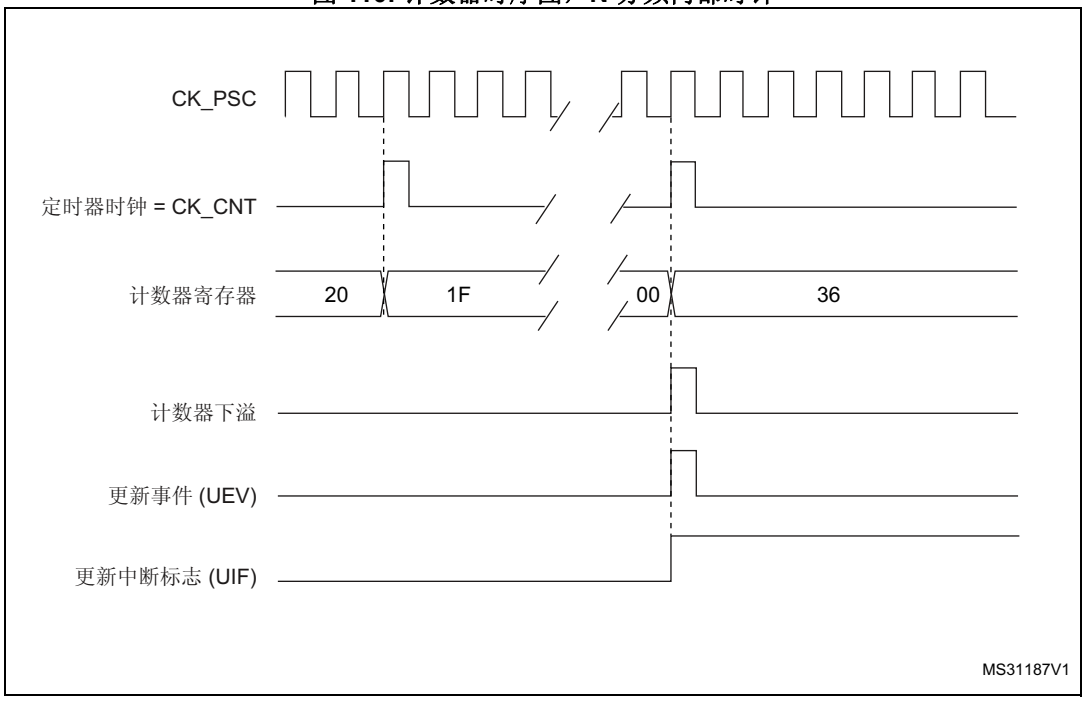
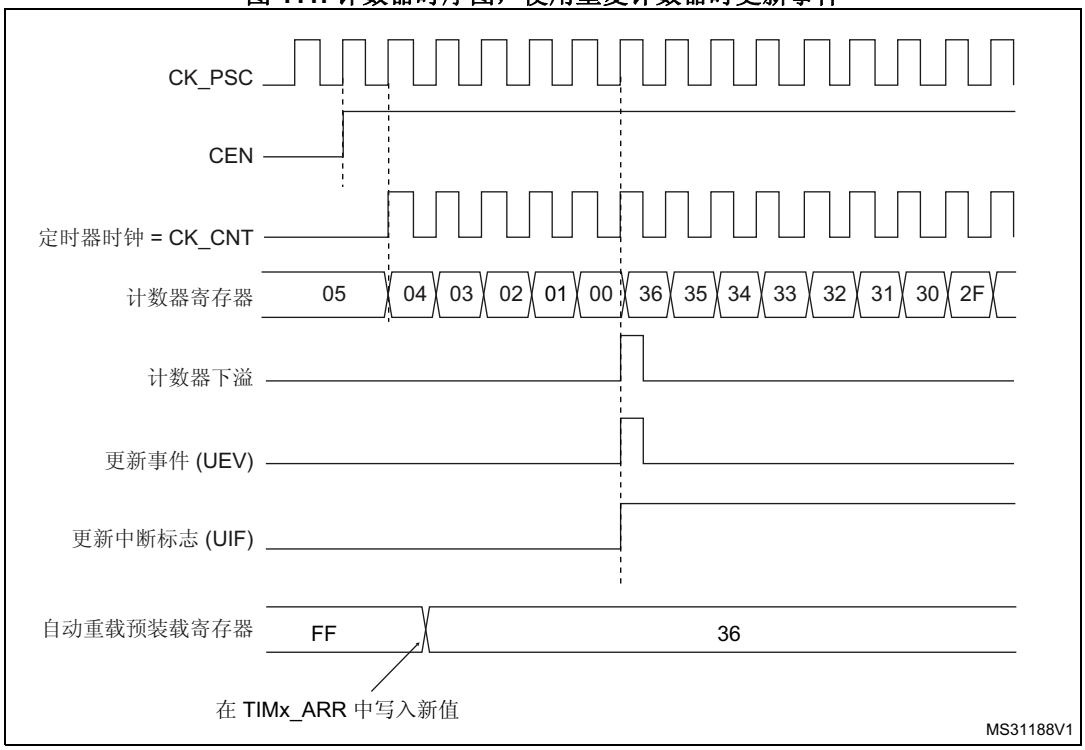


图 111. 计数器时序图，使用重复计数器时更新事件



中心对齐模式（递增/递减计数）

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS =“01”）、计数器递增计数（中心对齐模式 2，CMS =“10”）以及计数器递增/递减计数（中心对齐模式 3，CMS =“11”）。

此模式下无法写入方向位（TIMx_CR1 寄存器中的 DIR 位）。而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

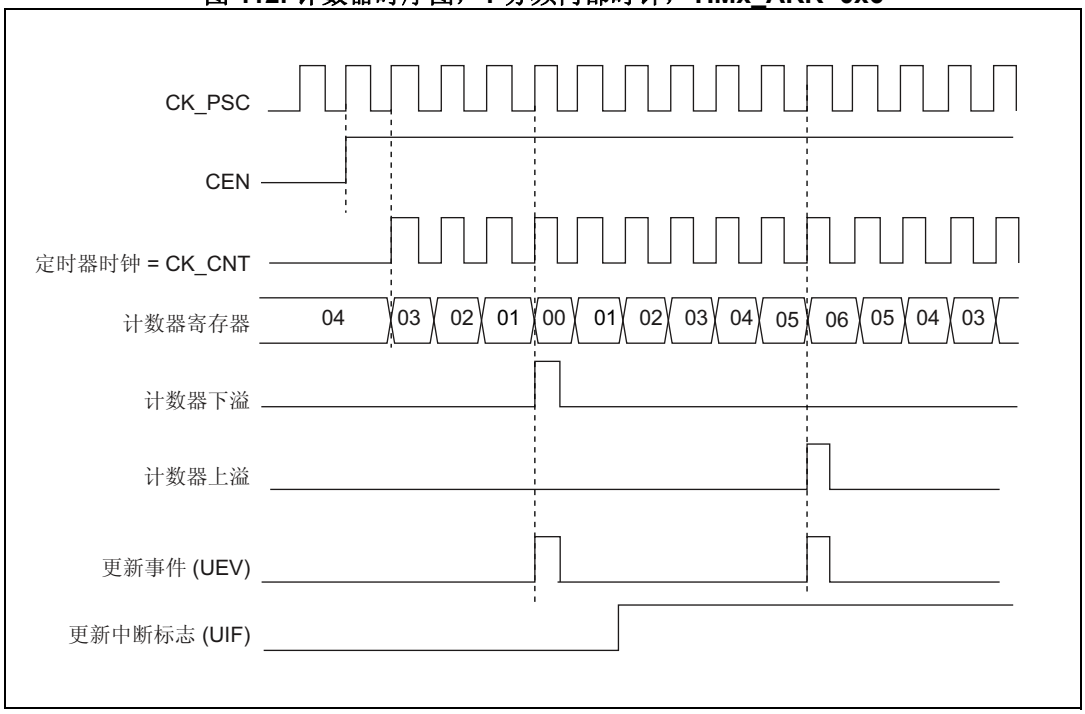
此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则 ARR 寄存器在计数器重载之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。

图 112. 计数器时序图，1 分频内部时钟，TIMx_ARR=0x6



1. 此处使用中心对齐模式 1（有关详细信息，请参见*第 470 页的第 21.4.1 节：TIMx 控制寄存器 1 (TIMx_CR1)*）。

图 113. 计数器时序图，2 分频内部时钟

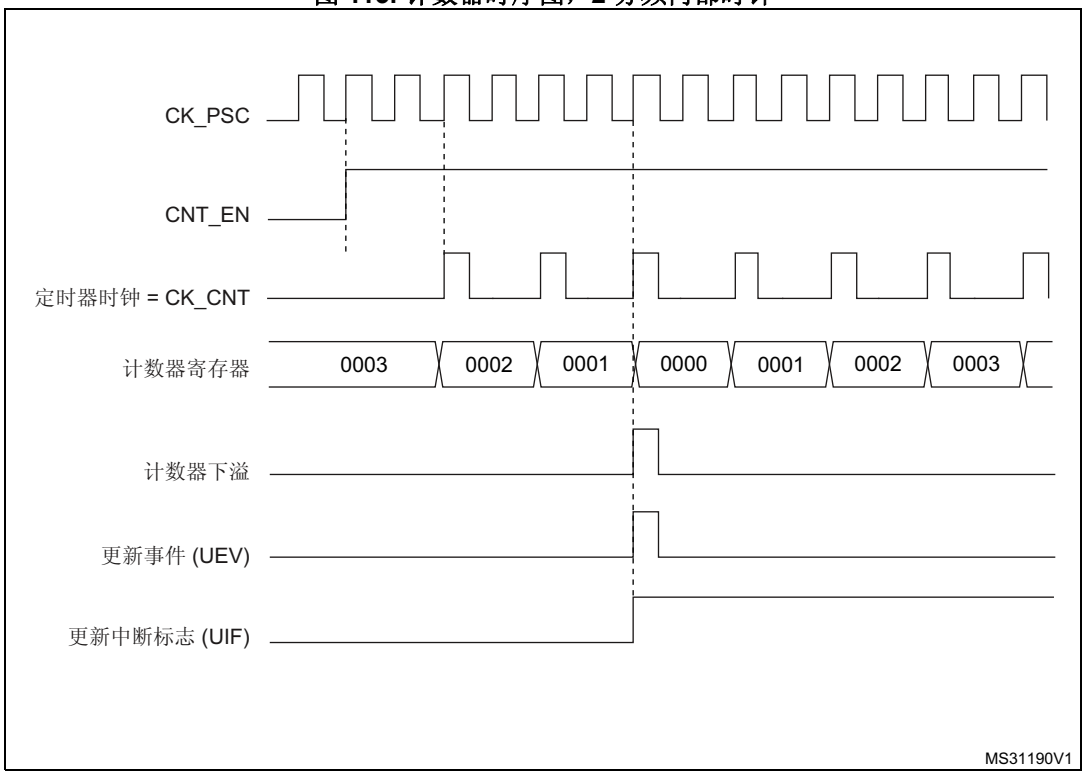
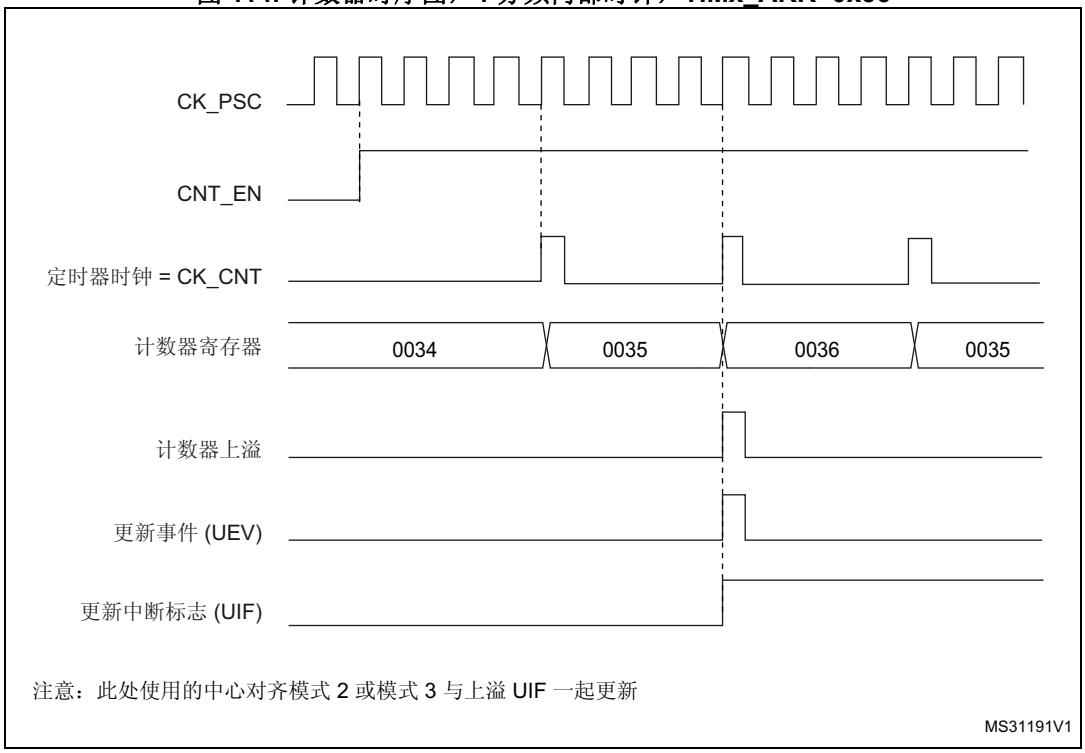


图 114. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36



1. 中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 115. 计数器时序图，N 分频内部时钟

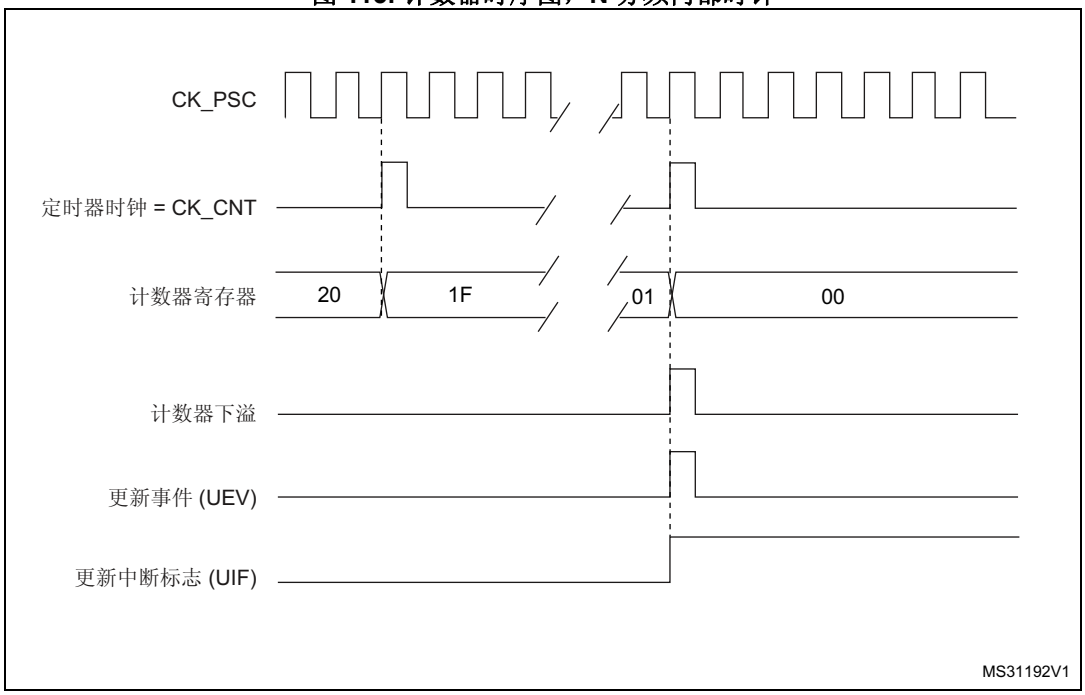


图 116. 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

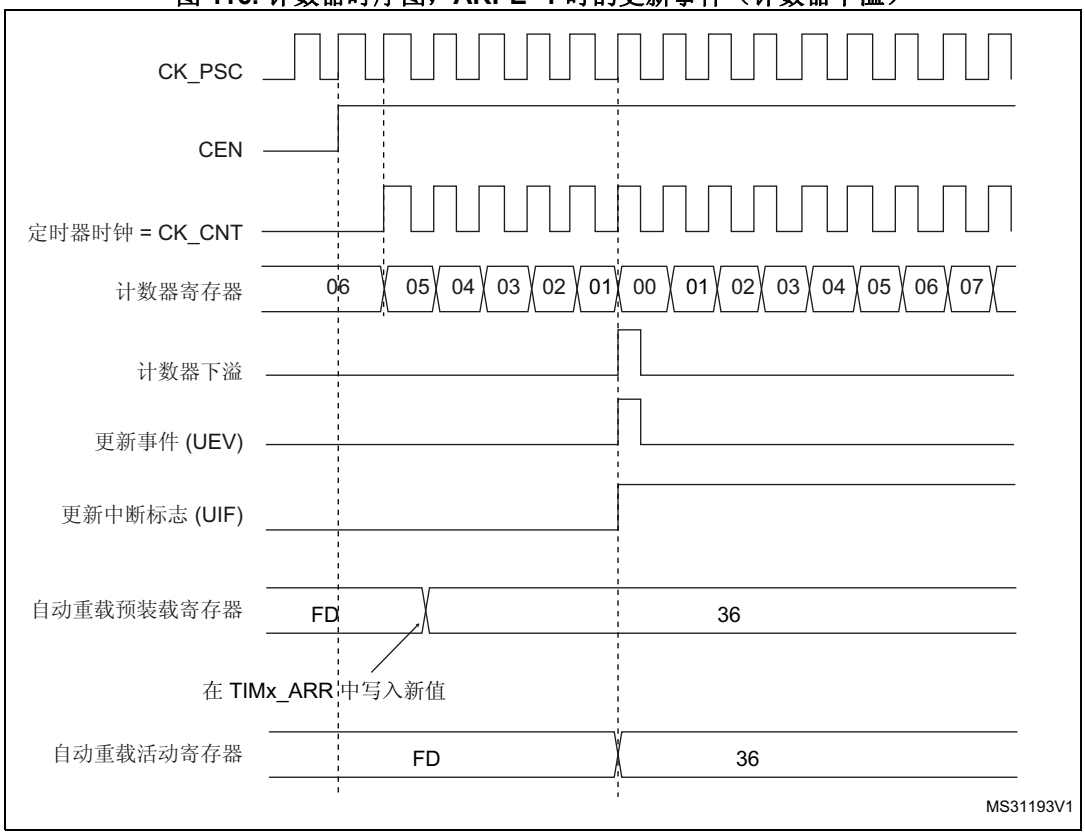
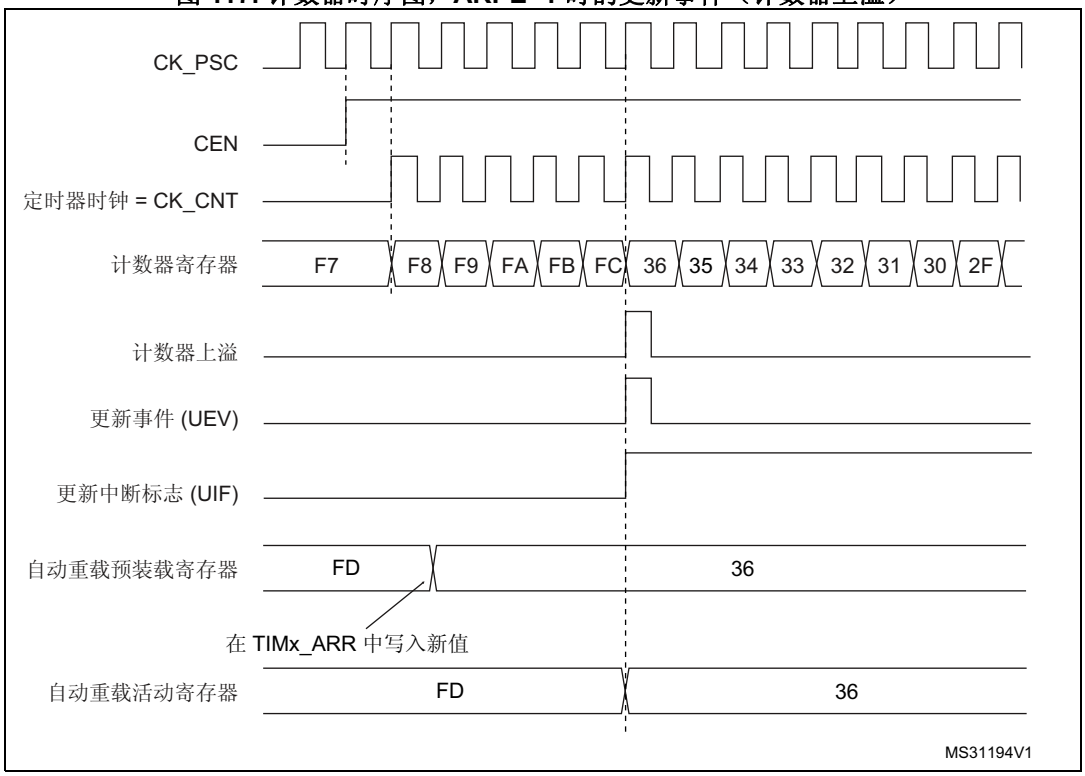


图 117. 计数器时序图，ARPE=1 时的更新事件（计数器上溢）



21.3.3 时钟选择

计数器时钟可由下列时钟源提供：

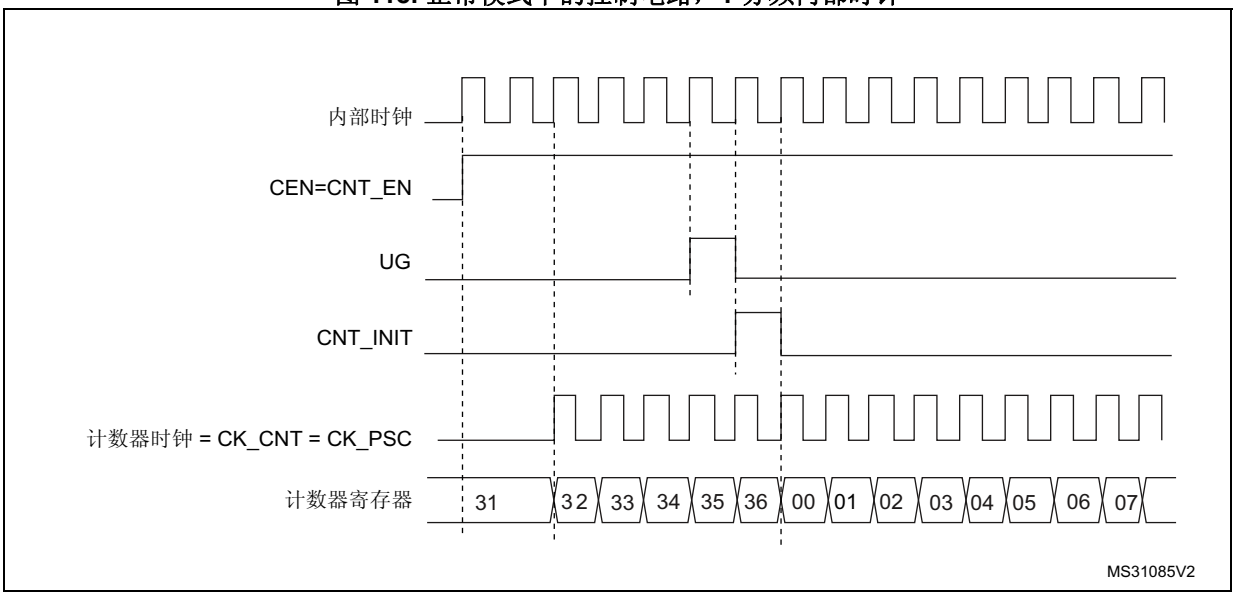
- 内部时钟 (CK_INT)
- 外部时钟模式 1：外部输入引脚 (TIx)
- 外部时钟模式 2：外部触发输入 (ETR)
- 内部触发输入 (ITRx)：将一个定时器用作另一个定时器的预分频器。更多详细信息，请参见*第 464 页的将一个定时器用作另一个定时器的预分频器*一节。

内部时钟源 (CK_INT)

如果禁止从模式控制器（TIMx_SMCR 寄存器中 SMS=000），则 CEN 位、DIR 位（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（UG 除外，仍自动清零）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 118 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

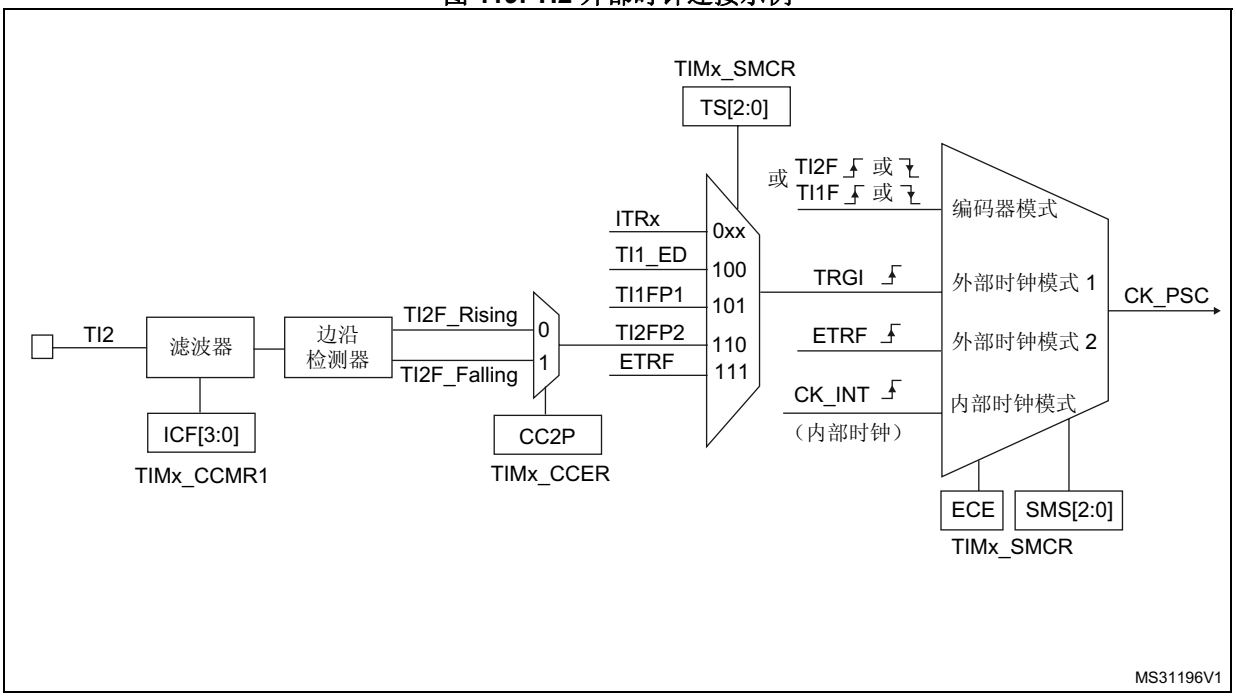
图 118. 正常模式下的控制电路，1 分频内部时钟



外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=111 时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

图 119. TI2 外部时钟连接示例



例如，要使递增计数器在 TI2 输入出现上升沿时计数，请执行以下步骤：

1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S=“01”来配置通道 2，使其能够检测 TI2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0] 位来配置输入滤波带宽（如果不需要任何滤波器，请保持 IC2F=0000）。

注： *由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

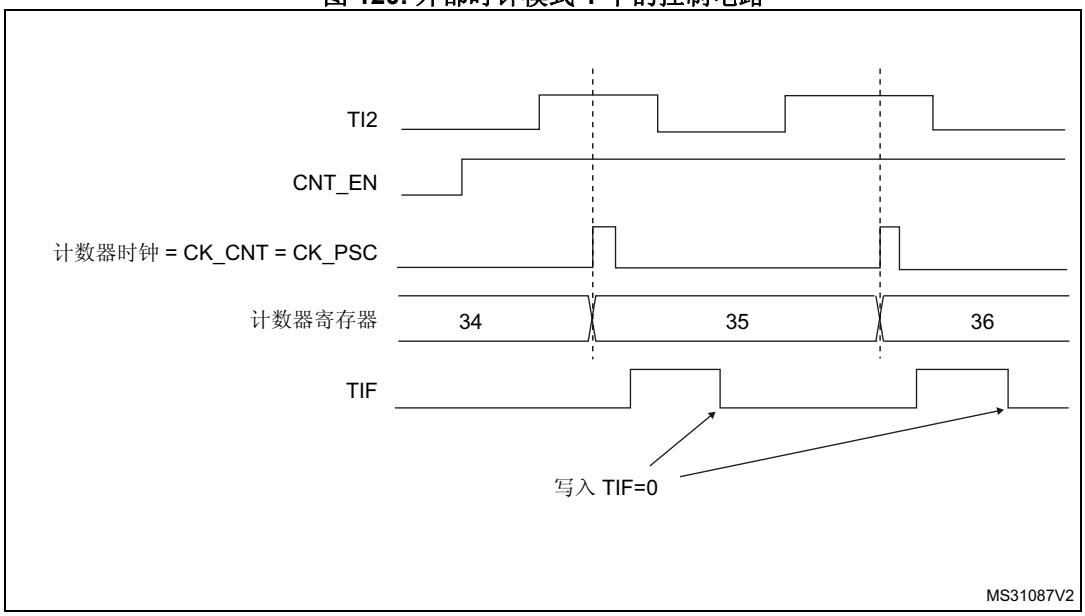
3. 通过在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=0 来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 TI2 作为输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

有关代码示例，请参见 *A.11.1：TI2 上升沿递增计数器代码示例*。

当 TI2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。

TI2 的上升沿与实际计数器时钟之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 120. 外部时钟模式 1 下的控制电路



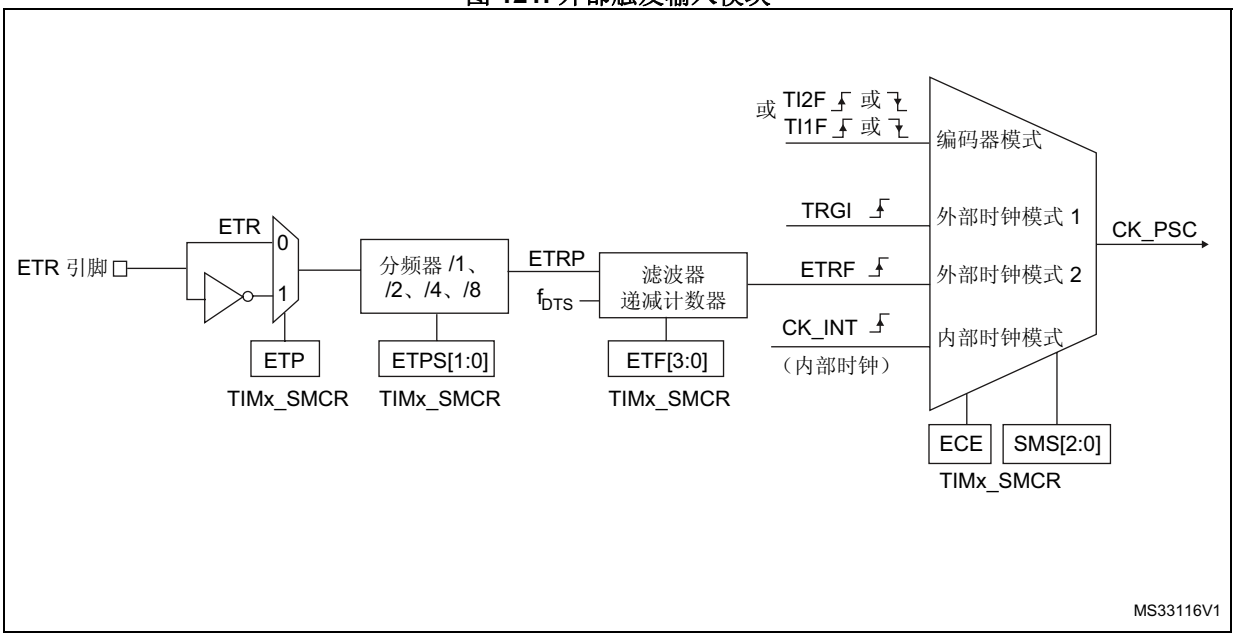
外部时钟源模式 2

通过在 TIMx_SMCR 寄存器中写入 ECE=1 可选择此模式。

计数器可在外部触发输入 ETR 出现上升沿或下降沿时计数。

图 121 简要介绍了外部触发输入模块。

图 121. 外部触发输入模块



例如，要使递增计数器在 ETR 每出现 2 个上升沿时计数，请执行以下步骤：

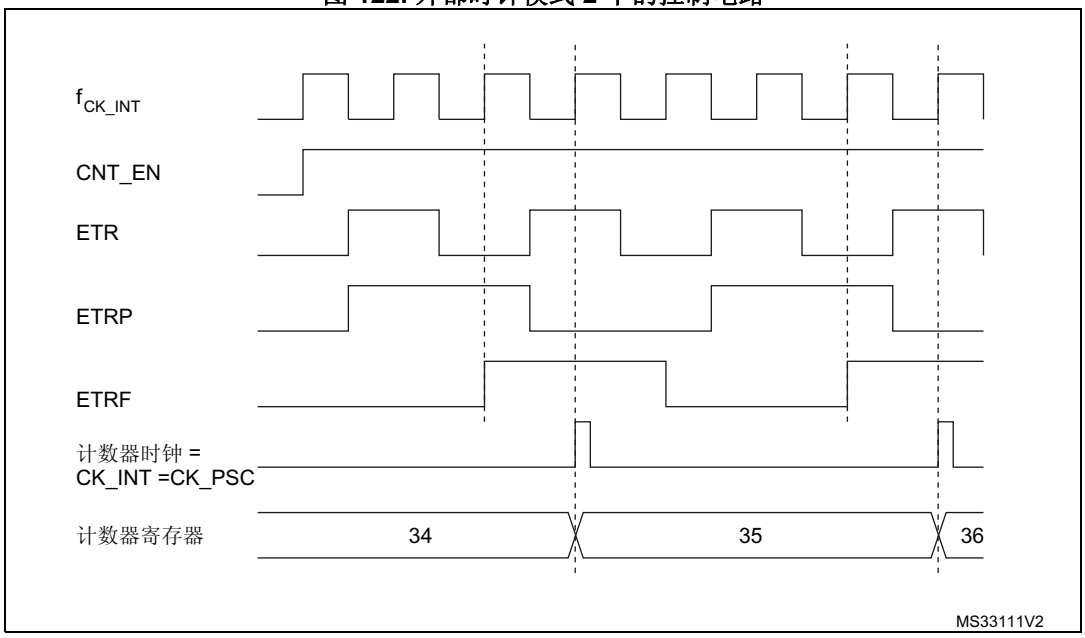
1. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
2. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器。
3. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测。
4. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
5. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

有关代码示例，请参见 *A.11.2：每 2 个 ETR 上升沿递增计数器代码示例*。

ETR 每出现 2 个上升沿，计数器计数一次。

ETR 的上升沿与实际计数器时钟之间的延迟是由于 ETRP 信号的重新同步电路引起的。

图 122. 外部时钟模式 2 下的控制电路



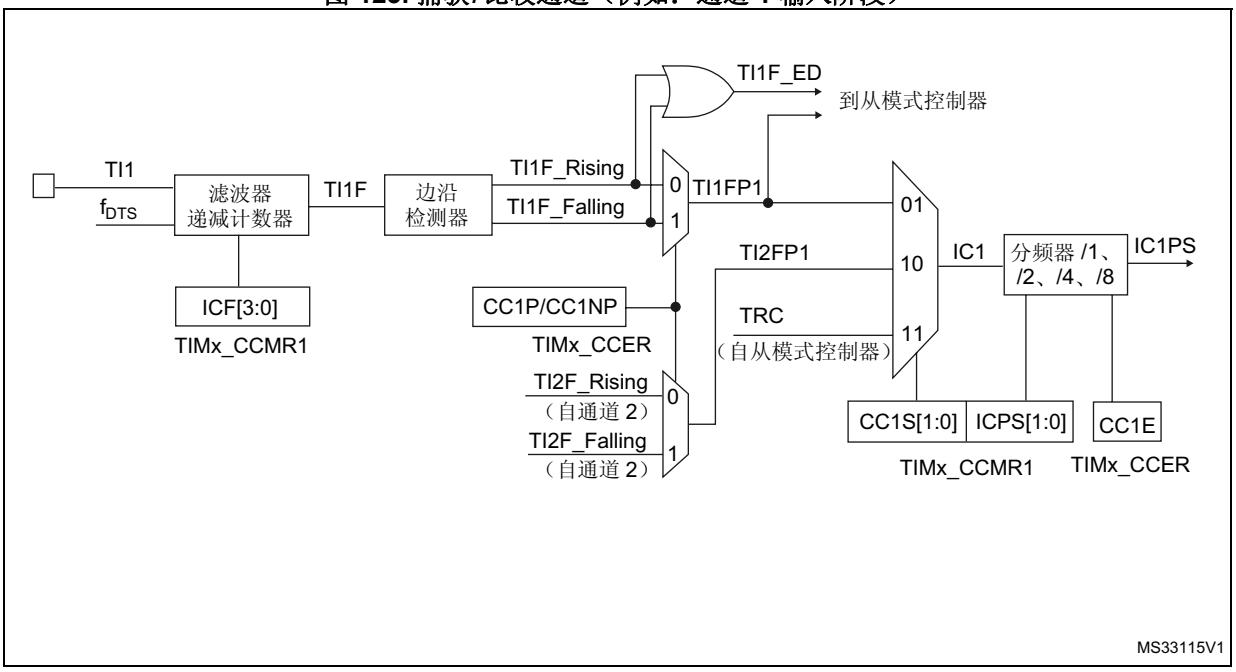
21.3.4 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

下图简要介绍了一路捕获/比较通道。

输入阶段对相应的 TIx 输入进行采样，生成一个滤波后的信号 TIxF。然后，带有极性选择功能的边沿检测器生成一个信号 (TIxFPx)，该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频 (ICxPS)，而后再进入捕获寄存器。

图 123. 捕获/比较通道（例如：通道 1 输入阶段）



输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 124. 捕获/比较通道 1 主电路

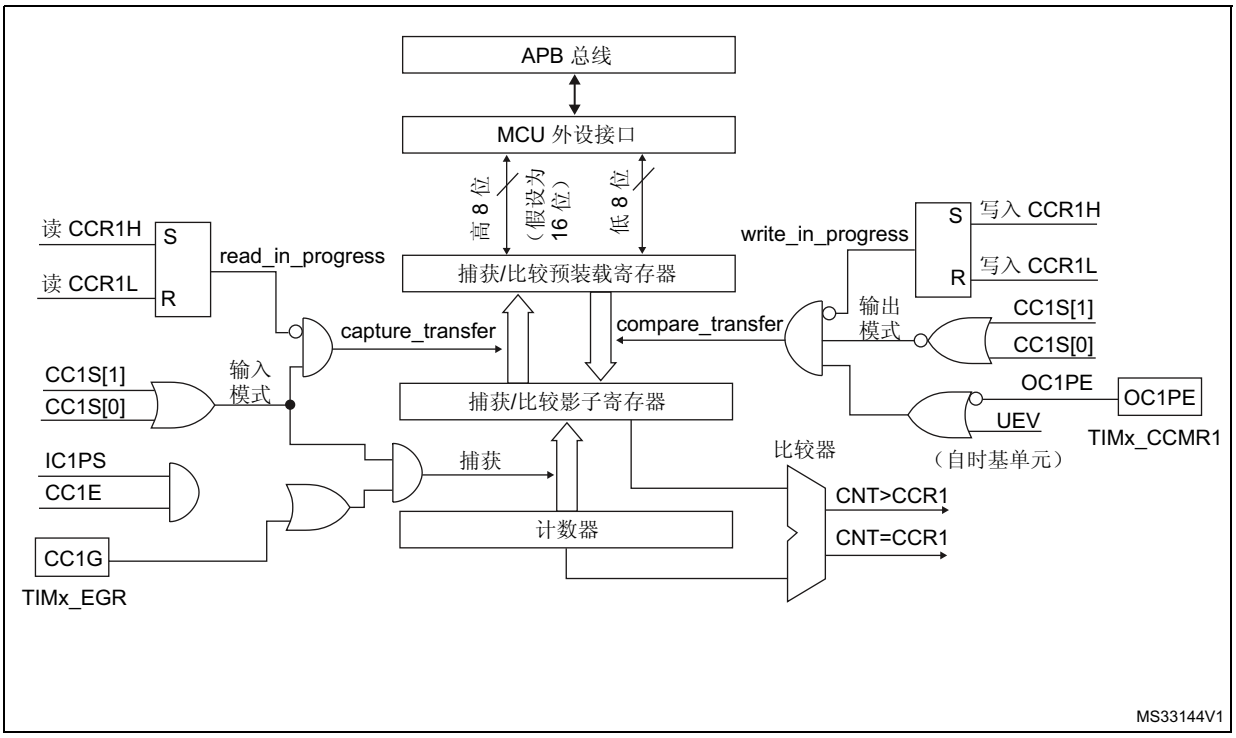
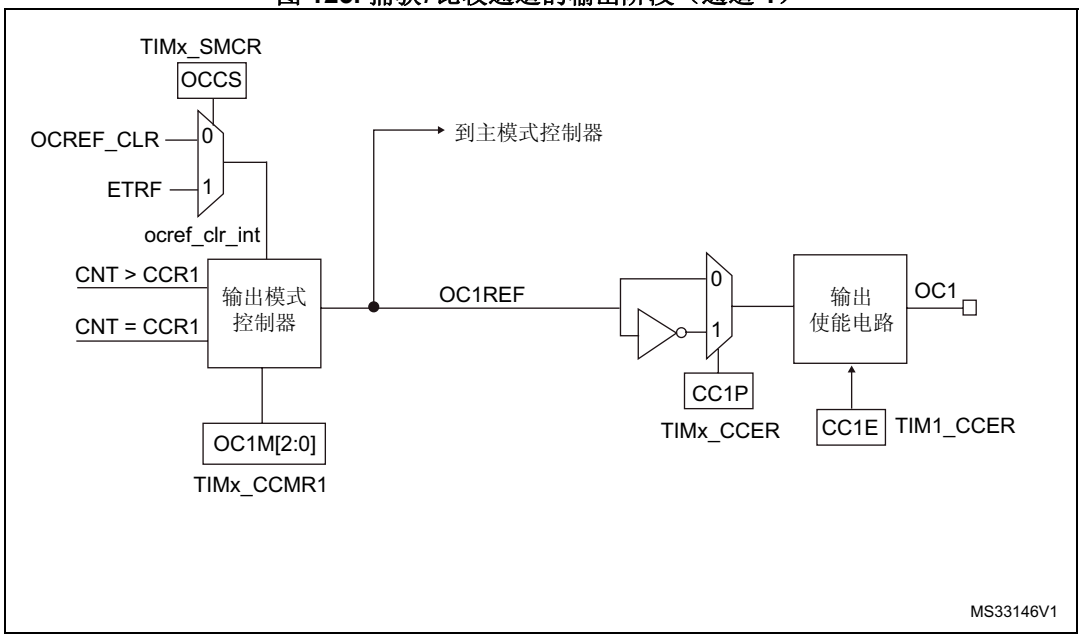


图 125. 捕获/比较通道的输出阶段（通道 1）



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。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

21.3.5 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器 (TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件方法向 CCxIF 写入 0 来给 CCxIF 清零，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入 0 后会将其清零。

以下示例说明了如何在 TI1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 选择有效输入：TIMx_CCR1 必须连接到 TI1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01。只要 CC1S 不等于 00，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
2. 根据连接到定时器的信号，对所需的输入滤波带宽进行编程（如果输入为 TIx 之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号边沿变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波带宽设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 f_{DTS} 频率采样）后，可以确认 TI1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入 0011。

3. 通过向 TIMx_CCER 寄存器中的 CC1P 位和 CC1NP 位写入 0，选择 TI1 通道的有效转换边沿（本例中为上升沿）。
4. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PS 位写入 00）。
5. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
6. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求，并且/或者通过将该寄存器中的 CC1DE 位置 1 来使能 DMA 请求。

有关代码示例，请参见 *A.11.3：输入捕捉配置代码示例*。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。
- 根据 CC1DE 位生成 DMA 请求。

有关代码示例，请参见 *A.11.4：输入捕捉数据管理代码示例*。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

注： *通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断和/或 DMA 请求。*

21.3.6 PWM 输入模式

此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

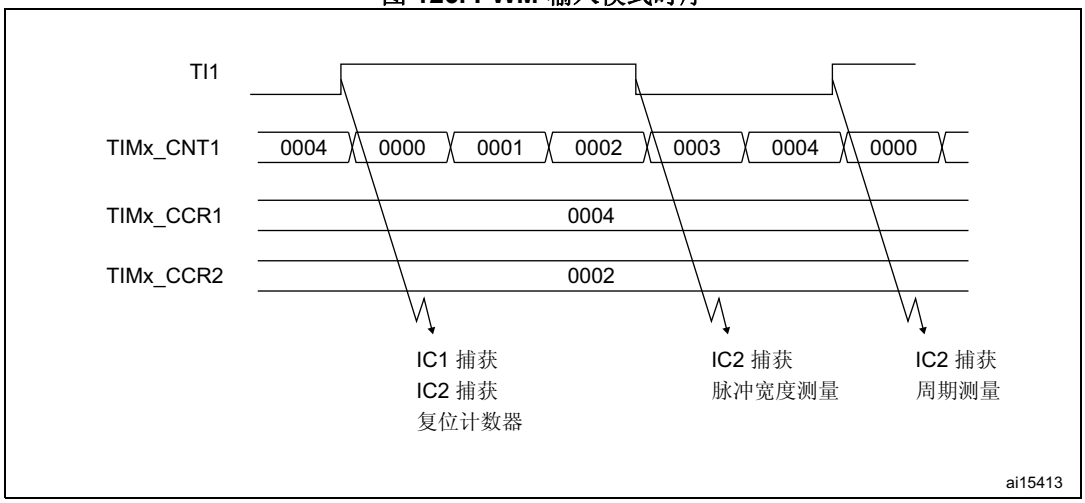
- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤对应用于 TI1 的 PWM 的周期（位于 TIMx_CCR1 寄存器中）和占空比（位于 TIMx_CCR2 寄存器中）进行测量（取决于 CK_INT 频率和预分频器的值）：

1. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入 01（选择 TI1）。
2. 选择 TI1FP1 的有效极性（同时用于 TIMx_CCR1 中的捕获和计数器清零）：向 CC1P 位和 CC1NP 位写入“0”（上升沿有效）。
3. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 写入 10（选择 TI1）。
4. 选择 TI1FP2 的有效极性（用于 TIMx_CCR2 中的捕获）：向 CC2P 位写入“1”，向 CC2NP 位写入“0”（下降沿有效）。
5. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入 101（选择 TI1FP1）。
6. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入 100。
7. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

有关代码示例，请参见 *A.11.5：PWM 输入配置代码示例*。

图 126. PWM 输入模式时序



21.3.7 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位 = 00）下，可直接由软件将每个输出比较信号（OCxREF 和 OCx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号 (OCxREF/OCx) 强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入 101。OCxREF 进而强制设置为高电平（OCxREF 始终为高电平有效），同时 OCx 获取 CCxP 极性位的相反值。

例如：CCxP=0（OCx 高电平有效）=> OCx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 100，可将 OCxREF 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可发送相应的中断和 DMA 请求。输出比较模式一节对此进行了介绍。

21.3.8 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

- 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx_CCER 寄存器中的 CCxP 位）定义。匹配时，输出引脚既可保持其电平 (OCXM=000)，也可设置为有效电平 (OCXM=001)、无效电平 (OCXM=010) 或进行翻转 (OCxM=011)。
- 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
- 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。
- 如果相应使能位（TIMx_DIER 寄存器的 CCxDE 位，TIMx_CR2 寄存器的 CCDS 位，用来选择 DMA 请求）置 1，将发送 DMA 请求。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

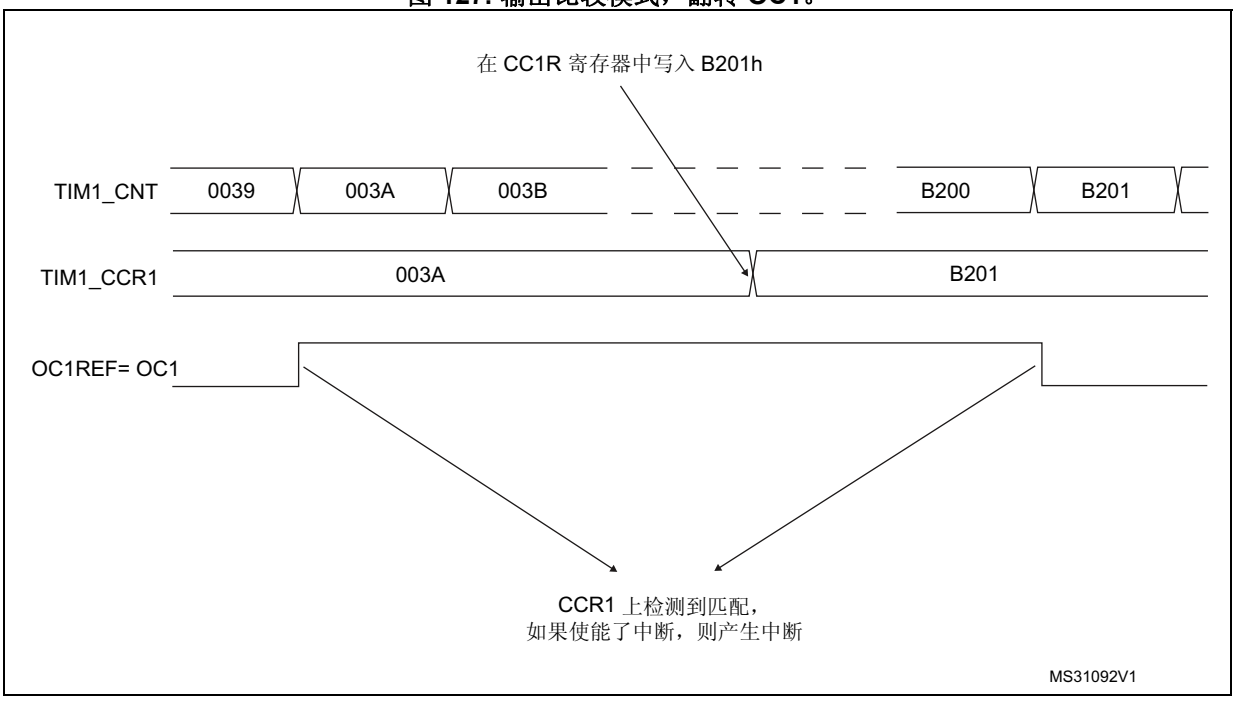
步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断和/或 DMA 请求，将 CCxIE 位和/或 CCxDE 位置 1。
4. 选择输出模式。例如，当 CNT 与 CCRx 匹配、未使用预装载 CCRx 并且 OCx 使能且为高电平有效时，必须写入 OCxM=011、OCxPE=0、CCxP=0 和 CCxE=1 来翻转 OCx 输出引脚。
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

有关代码示例，请参见 *A.11.7：输出比较配置代码示例*。

可随时通过软件更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预装载寄存器（OCxPE=0，否则仅当发生下一个更新事件 UEV 时，才会更新 TIMx_CCRx 影子寄存器）。*图 127* 给出了一个示例。

图 127. 输出比较模式，翻转 OC1。



21.3.9 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入 110（PWM 模式 1）或 111（PWM 模式 2），可以独立选择各通道（每个 OCx 输出对应一个 PWM）的 PWM 模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器（在递增计数或中心对齐模式下）。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

OCx 极性可通过软件来编程（使用 TIMx_CCER 寄存器的 CCxP 位）。可将其编程为高电平有效或低电平有效。OCx 输出通过将 TIMx_CCER 寄存器中的 CCxE 位置 1 来使能。有关详细信息，请参见 TIMx_CCERx 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 总是与 TIMx_CCRx 进行比较，以确定是 TIMx_CCRx ≤ TIMx_CNT 还是 TIMx_CNT ≤ TIMx_CCRx（取决于计数器计数方向）。不过，为符合 OCREF_CLR 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 OCREF），OCREF 信号仅在以下情况下变为有效状态：

- 比较结果发生改变，或
- 输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）从“冻结”配置（不进行比较，OCxM=“000”）切换为任一 PWM 模式（OCxM=“110”或“111”）。

定时器运行期间，可以通过软件强制 PWM 输出。

根据 TIMx_CR1 寄存器中的 CMS 位状态，定时器能够产生边沿对齐模式或中心对齐模式的 PWM 信号。

PWM 边沿对齐模式

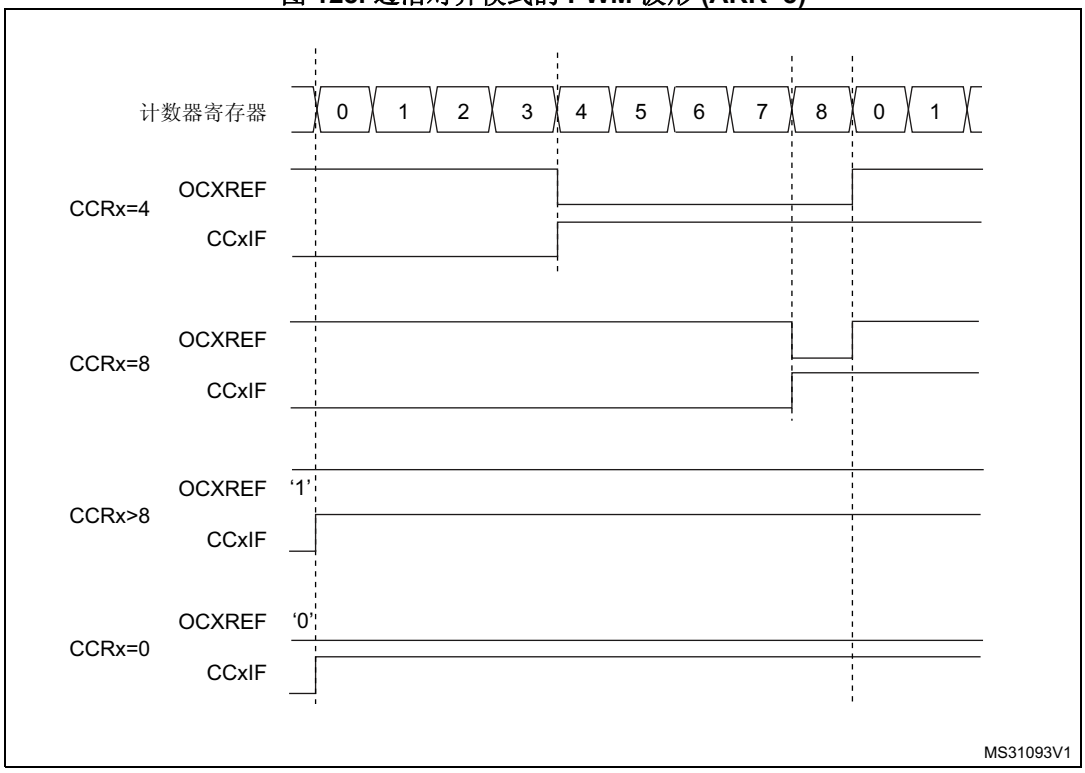
递增计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。请参见*第 433 页的递增计数模式*一节。

以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 OCxREF 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 OCxREF 保持为“1”。如果比较值为 0，则 OCxREF 保持为“0”。*图 128* 举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

有关代码示例，请参见 *A.11.8：边沿对齐 PWM 配置示例*。

图 128. 边沿对齐模式的 PWM 波形 (ARR=8)



递减计数配置

当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。请参见*第 436 页的递减计数模式*一节。

在 PWM 模式 1 下，只要 TIMx_CNT>TIMx_CCRx，参考信号 OCxREF 便为低电平，否则为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，则 OCxREF 保持为“1”。此模式下不可能产生 0% 的 PWM 波形。

PWM 中心对齐模式

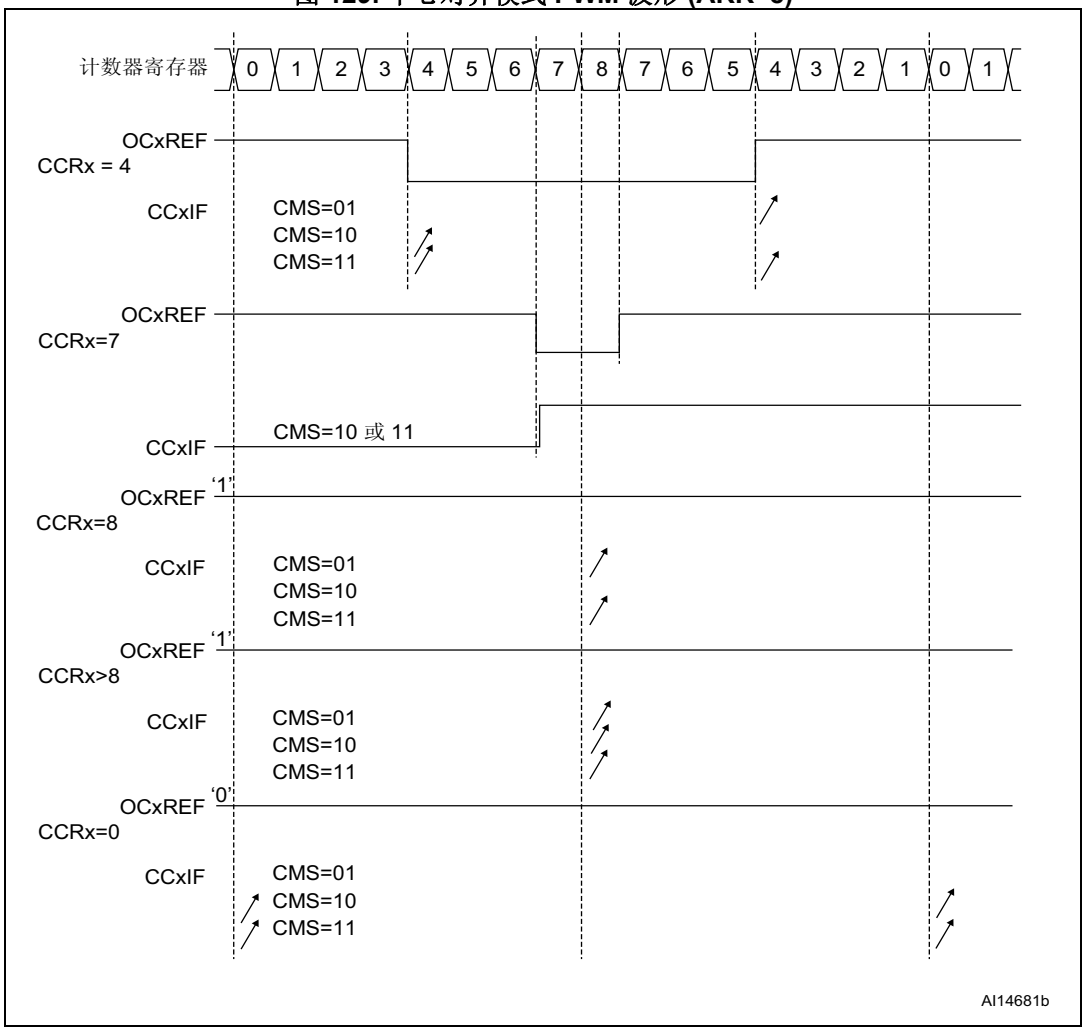
当 TIMx_CR1 寄存器中的 CMS 位不为“00”时（其余所有配置对 OCxREF/OCx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位 (DIR) 由硬件更新，不得通过软件更改。请参见*第 439 页的中心对齐模式（递增/递减计数）*一节。

图 129 显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8，
- PWM 模式为 PWM 模式 1，
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

有关代码示例，请参见 *A.11.9：中心对齐 PWM 配置示例*。

图 129. 中心对齐模式 PWM 波形 (ARR=8)



中心对齐模式使用建议：

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。
- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：
  - 如果写入计数器中的值大于自动重载值 (TIMx_CNT>TIMx_ARR)，计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
  - 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。
- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

21.3.10 单脉冲模式

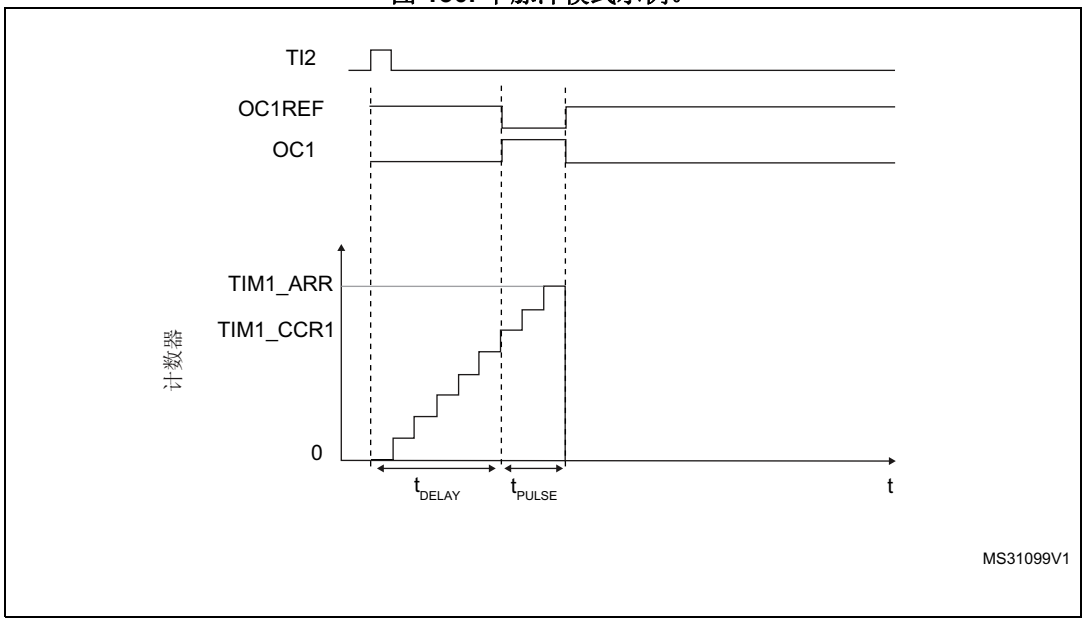
单脉冲模式 (OPM) 是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

- 递增计数时：CNT<CCRx≤ ARR（特别注意，0<CCRx），
- 递减计数时：CNT>CCRx。

图 130. 单脉冲模式示例。



例如，用户希望达到这样的效果：在 TI2 输入引脚检测到上升沿时，经过 t_{DELAY} 的延迟，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

使用 TI2FP2 作为触发 1：

1. 在 TIMx_CCMR1 寄存器中写入 CC2S=01，将 TI2FP2 映射到 TI2。
2. 在 TIMx_CCER 寄存器中写入 CC2P=0 和 CC2NP=“0”，使 TI2FP2 能够检测上升沿。
3. 在 TIMx_SMCR 寄存器中写入 TS=110，将 TI2FP2 配置为从模式控制器的触发 (TRGI)。
4. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），使用 TI2FP2 启动计数器。

有关代码示例，请参见 *A.11.16：单脉冲模式代码示例*。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由写入 TIMx_CCR1 寄存器的值定义。
- t_{PULSE} 由自动重载值与比较值之差 (TIMx_ARR - TIMx_CCR1+1) 来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=111，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入 1，使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 TI2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位置“0”时，即选择重复模式。

特殊情况：OCx 快速使能：

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（t_{DELAY} 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 OCxRef（和 OCx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用。

有关代码示例，请参见 *A.11.16：单脉冲模式代码示例*。

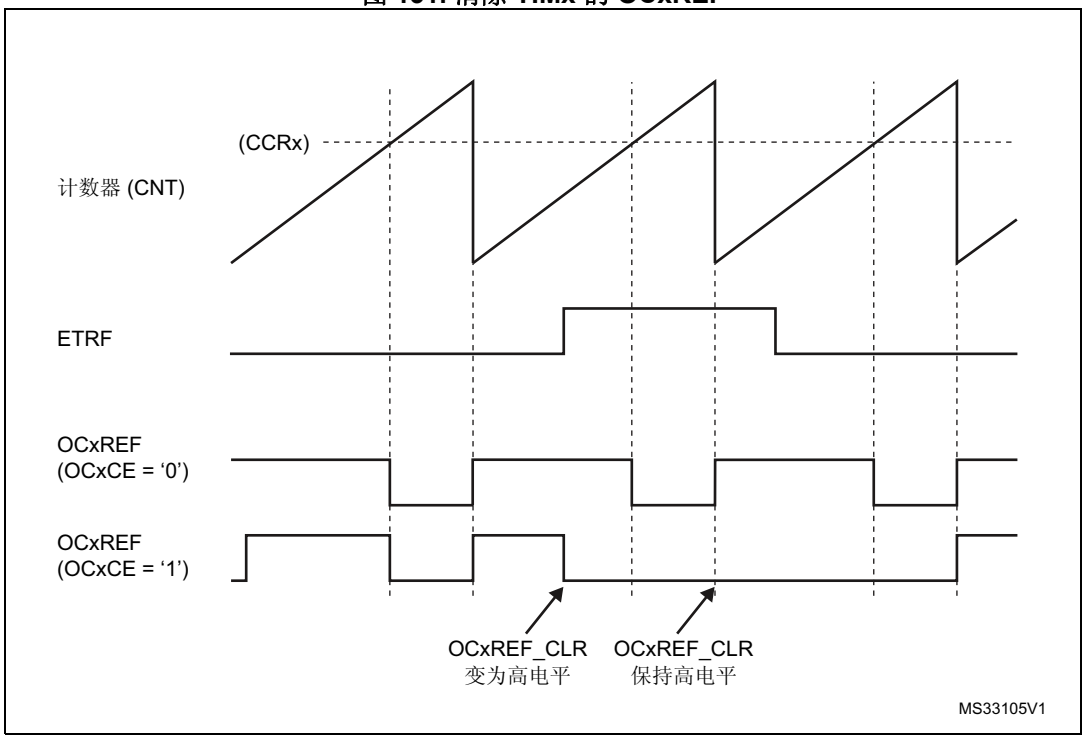
21.3.11 发生外部事件时清除 OCxREF 信号

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0] 位清为 00。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位清为 0。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可根据应用需要进行配置。

有关代码示例，请参见 *A.11.10：清零 OCxREF 的 ETR 配置代码示例*。

图 131 对比了不同使能位 OCxCE 值的情况，显示了当 ETRF 输入变为高电平时 OCxREF 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

图 131. 清除 TIMx 的 OCxREF



1. 如果 PWM 的占空比为 100% (CCRx>ARR)，则下次计数器上溢时会再次使能 OCxREF。

21.3.12 编码器接口模式

选择编码器接口模式时，如果计数器仅在 TI2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=001；如果计数器仅在 TI1 边沿处计数，写入 SMS=010；如果计数器在 TI1 和 TI2 边沿处均计数，则写入 SMS=011。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 TI1 和 TI2 极性。CC1NP 和 CC2NP 必须保持清零。如果需要，还可对输入滤波器进行编程。

TI1 和 TI2 两个输入用于连接增量编码器。请参见*表 87*。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 TI1FP1 或 TI2FP2 上的每次有效信号转换提供。TI1FP1 和 TI2FP2 是进行输入滤波器和极性选择后 TI1 和 TI2 的信号，如果不进行滤波和反相，则 TI1FP1=TI1，TI2FP2=TI2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（TI1 或 TI2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 TI1 或 TI2 边沿处计数，还是同时在 TI1 和 TI2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、触发输出功能继续正常工作。

在此模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 TI1 和 TI2 不同时切换）。

表 87. 计数方向与编码器信号的关系

有效边沿	相反信号的电平（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 处计数	高	递减	递增	不计数	不计数
	低	递增	递减	不计数	不计数
仅在 TI2 处计数	高	不计数	不计数	递增	递减
	低	不计数	不计数	递减	递增
在 TI1 和 TI2 处均计数	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

图 132 以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S=01（TIMx_CCMR1 寄存器，TI1FP1 映射到 TI1 上）
- CC2S=01（TIMx_CCMR2 寄存器，TI2FP2 映射到 TI2 上）
- CC1P=0，CC1NP = “0”（TIMx_CCER 寄存器，TI1FP1 未反相，TI1FP1=TI1）
- CC2P=0，CC2NP = “0”（TIMx_CCER 寄存器，TI2FP2 未反相，TI2FP2=TI2）
- SMS=011（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿均有效）
- CEN = 1（TIMx_CR1 寄存器，使能计数器）

有关代码示例，请参见 *A.11.11：编码器接口代码示例*。

图 132. 编码器接口模式下的计数器工作示例

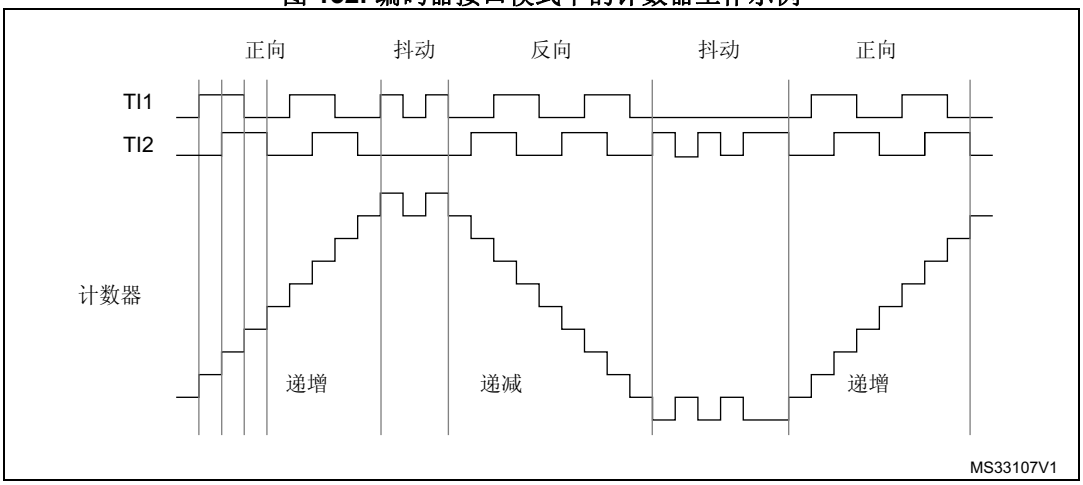
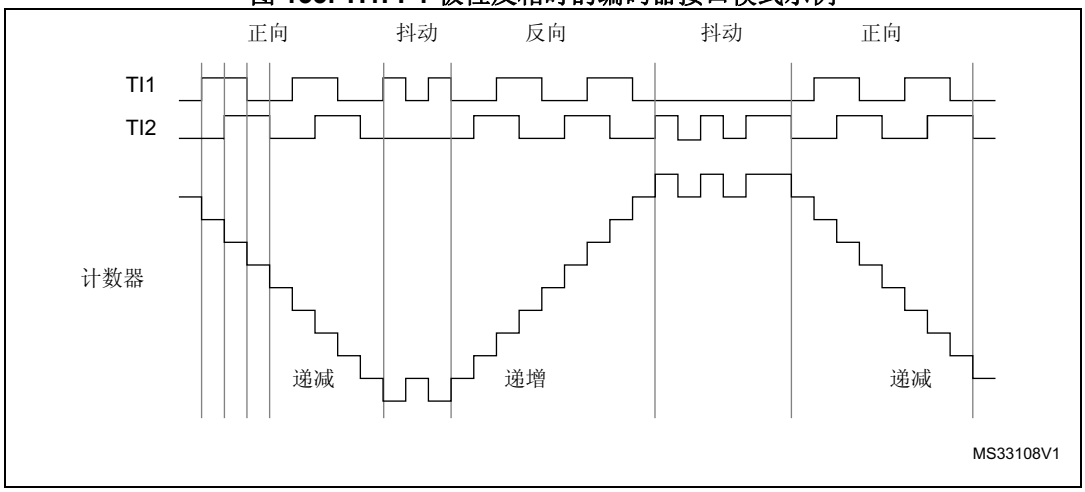


图 133 说明了 TI1FP1 极性反相时的计数器行为示例（除 CC1P=1 外，其它配置与上例相同）。

图 133. TI1FP1 极性反相时的编码器接口模式示例



定时器配置为编码器接口模式时，会提供传感器当前位置的相关信息。使用另一个配置为捕获模式的定时器测量两个编码器事件之间的周期，可获得动态信息（速度、加速度和减速度）。指示机械零位的编码器输出即可用于此目的。根据两个事件之间的时间间隔，还可定期读取计数器。如果可能，可以将计数器值锁存到第三个输入捕获寄存器来实现此目的（捕获信号必须为周期性信号，可以由另一个定时器产生）； 此外，还可以通过实时时钟产生的 DMA 请求读取计数器值。

21.3.13 定时器输入异或功能

借助 TIMx_CR2 寄存器中的 TI1S 位，可将通道 1 的输入滤波器连接到异或门的输出，从而将 TIMx_CH1 到 TIMx_CH3 这三个输入引脚组合在一起。

异或输出可与触发或输入捕获等所有定时器输入功能配合使用。

21.3.14 定时器与外部触发同步

TIMx 定时器可与外部触发以下列模式实现同步：复位模式、门控模式和触发模式。

从模式：复位模式

当触发输入信号发生变化时，计数器及其预分频器可重新初始化。此外，如果 TIMx_CR1 寄存器中的 URS 位处于低电平，则会生成更新事件 UEV。然后，所有预装载寄存器（TIMx_ARR 和 TIMx_CCRx）都将更新。

在以下示例中，TI1 输入上出现上升沿时，递增计数器清零：

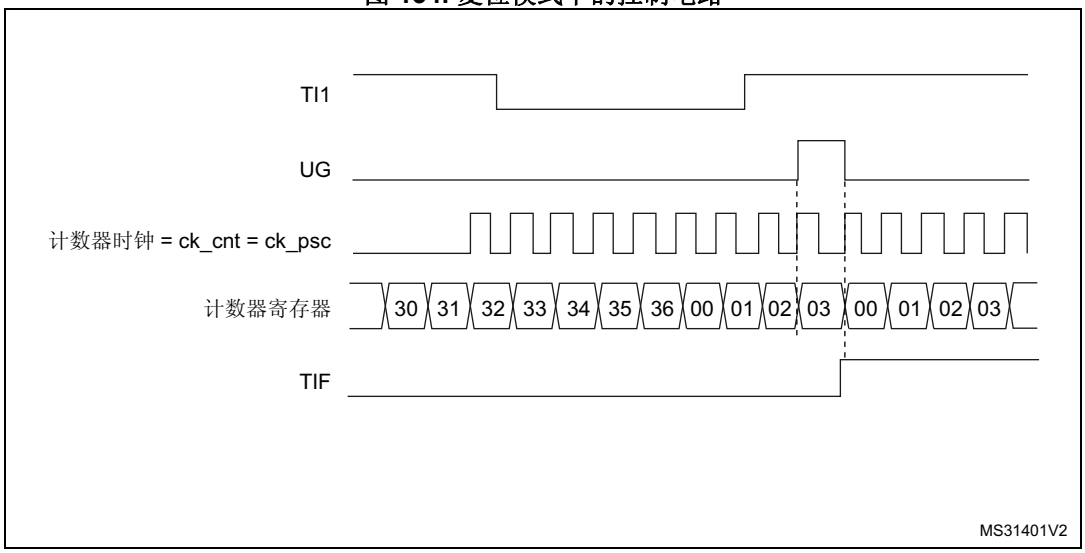
- 将通道 1 配置为检测 TI1 的上升沿。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = 01。在 TIMx_CCER 寄存器中写入 CC1P=0 和 CC1NP=0，验证极性（仅检测上升沿）。
- 在 TIMx_SMCR 寄存器中写入 SMS=100，将定时器配置为复位模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
- 在 TIMx_CR1 寄存器中写入 CEN=1，启动计数器。

有关代码示例，请参见 *A.11.12：复位模式代码示例*。

计数器开始根据内部时钟计数，然后正常运转，直到出现 TI1 上升沿。当 TI1 出现上升沿时，计数器清零，然后重新从 0 开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）置 1，使能中断或 DMA 后，还可发送中断或 DMA 请求（取决于 TIMx_DIER 寄存器中的 TIE 和 TDE 位）。

下图显示了自动重载寄存器 TIMx_ARR=0x36 时的相关行为。TI1 的上升沿与实际计数器复位之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 134. 复位模式下的控制电路



从模式：门控模式

输入信号的电平可用来使能计数器。

在以下示例中，递增计数器仅在 TI1 输入为低电平时计数：

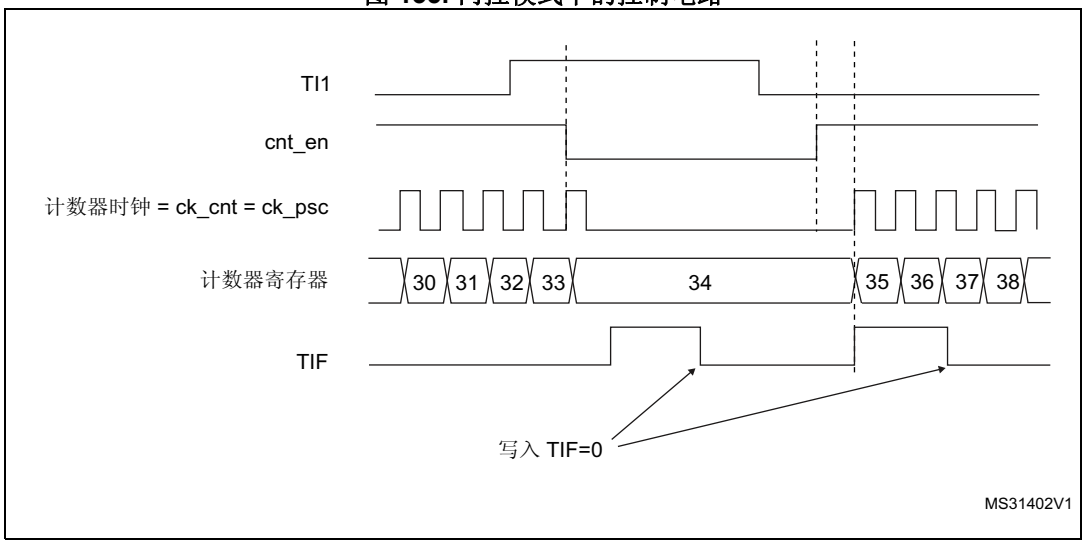
1. 将通道 1 配置为检测 TI1 上的低电平。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC1F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S=01。在 TIMx_CCER 寄存器中写入 CC1P=1 和 CC1NP=0，以确定极性（仅检测低电平）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=101，将定时器配置为门控模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。
3. 在 TIMx_CR1 寄存器中写入 CEN=1，使能计数器（在门控模式下，如果 CEN=0，则无论触发输入电平如何，计数器都不启动）。

有关代码示例，请参见 *A.11.13：门控模式代码示例*。

只要 TI1 为低电平，计数器就开始根据内部时钟计数，直到 TI1 变为高电平时停止计数。计数器启动或停止时，TIMx_SR 寄存器中的 TIF 标志都会置 1。

TI1 的上升沿与实际计数器停止之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 135. 门控模式下的控制电路



1. 由于门控模式作用于电平而非边沿，因此在门控模式下，“CCxP=CCxNP=1”（同时检测上升沿和下降沿）不发挥任何作用。

从模式：触发模式

所选输入上发生某一事件时可以启动计数器。

在以下示例中，TI2 输入上出现上升沿时，递增计数器启动：

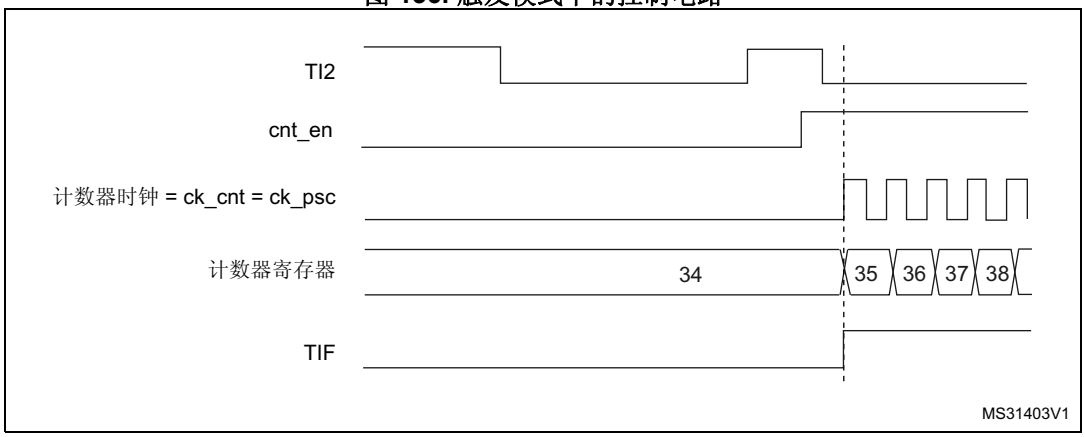
1. 将通道 2 配置为检测 TI2 上的上升沿。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC2F=0000）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC2S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC2S=01。在 TIMx_CCER 寄存器中写入 CC2P=1 和 CC2NP=0，以确定极性（仅检测低电平）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=110，选择 TI2 作为输入源。

有关代码示例，请参见 *A.11.14：触发模式代码示例*。

当 TI2 出现上升沿时，计数器开始根据内部时钟计数，并且 TIF 标志置 1。

TI2 的上升沿与实际计数器启动之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 136. 触发模式下的控制电路



从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可与另一种从模式（外部时钟模式 1 和编码器模式除外）结合使用。这种情况下，ETR 信号用作外部时钟输入，在复位模式、门控模式或触发模式下工作时，可选择另一个输入作为触发输入。不建议通过 TIMx_SMCR 寄存器中的 TS 位来选择 ETR 作为 TRGI。

在以下示例中，只要 TI1 出现上升沿，递增计数器即会在 ETR 信号的每个上升沿处递增：

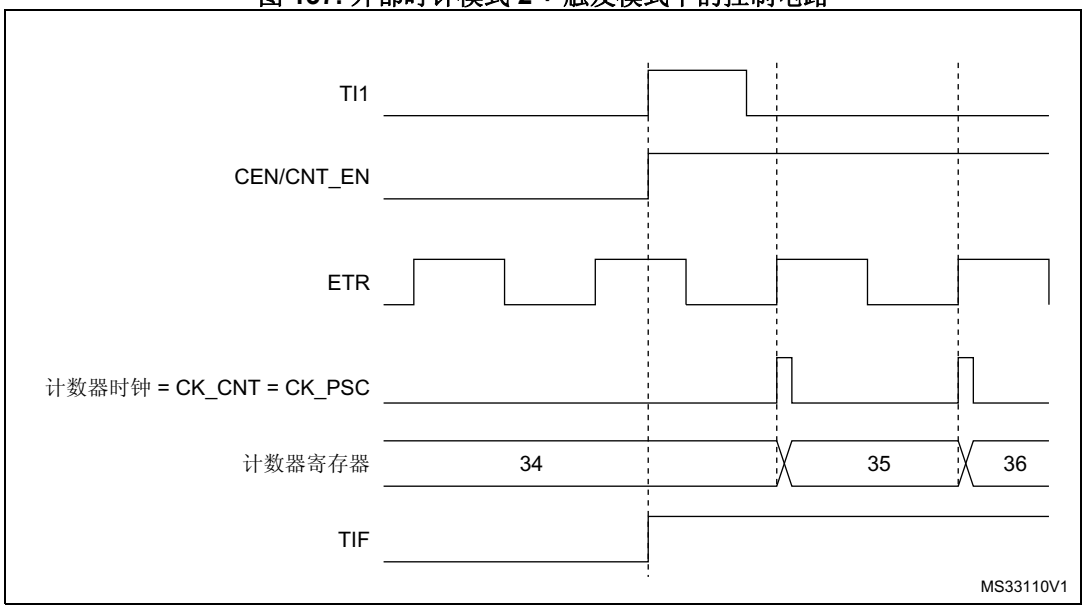
1. 通过对 TIMx_SMCR 寄存器进行如下编程，配置外部触发输入电路：
   - ETF = 0000：无滤波器
   - ETPS = 00：禁止预分频器
   - ETP = 0：检测 ETR 的上升沿，并写入 ECE=1，以使能外部时钟模式 2。
2. 如下配置通道 1，以检测 TI 的上升沿：
   - IC1F=0000：无滤波器。
   - 由于捕获预分频器不用于触发操作，因此无需对其进行配置。
   - TIMx_CCMR1 寄存器中 CC1S=01，只选择输入捕获源。
   - TIMx_CCER 寄存器中 CC1P=0 且 CC1NP= 0，以确定极性（仅检测上升沿）。
3. 在 TIMx_SMCR 寄存器中写入 SMS=110，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=101，选择 TI1 作为输入源。

有关代码示例，请参见 *A.11.15：外部时钟模式 2 + 触发模式代码示例*。

TI1 出现上升沿时将使能计数器并且 TIF 标志置 1。然后计数器在 ETR 出现上升沿时计数。

ETR 信号的上升沿与实际计数器复位之间的延迟是由于 ETRP 输入的重新同步电路引起的。

图 137. 外部时钟模式 2 + 触发模式下的控制电路



21.3.15 定时器同步

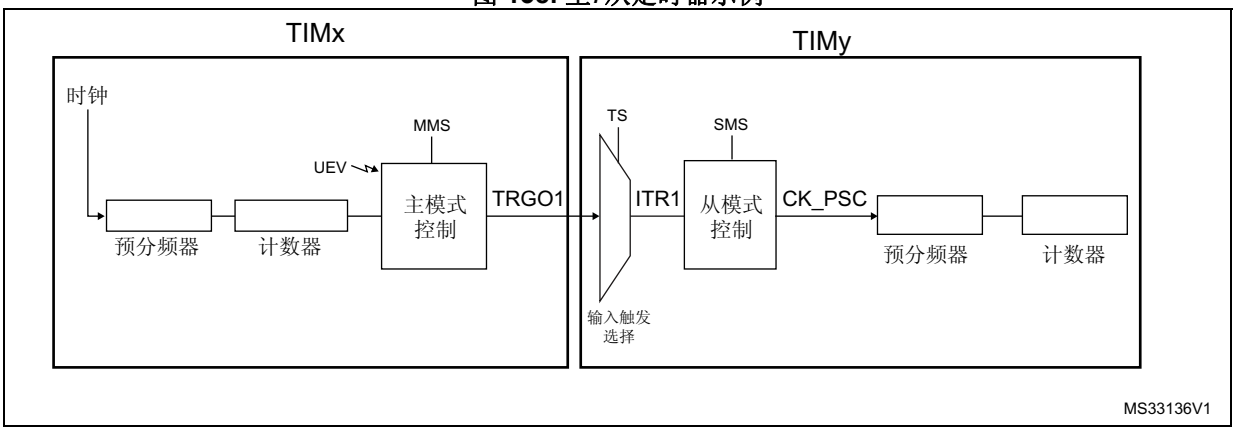
TIMx 定时器从内部连接在一起，以实现定时器同步或链接。当某个定时器配置为主模式时，可对另一个配置为从模式的定时器的计数器执行复位、启动、停止操作或为其提供时钟。

*图 138：主/从定时器示例*简要介绍了触发选择和主模式选择框图。

注： *必须先使能从定时器的时钟，才能从主定时器接收事件；从主定时器接收触发信号时，不得实时更改从定时器的时钟。*

将一个定时器用作另一个定时器的预分频器

图 138. 主/从定时器示例



例如，可以将定时器 x 配置为定时器 y 的预分频器。请参见*图 138*。为此，需遵循以下步骤：

1. 将定时器 x 配置为主模式，以便每次发生更新事件 UEV 时都输出一个周期性触发信号。如果在 TIMx_CR2 寄存器中写入 MMS=010，则每次生成更新事件时，TRGO1 都会输出一个上升沿。
2. 要将定时器 x 的 TRGO1 输出连接到定时器 y，必须将定时器 y 配置为从模式，使用 ITR1 作为内部触发。通过 TIMy_SMCR 寄存器中的 TS 位（写入 TS=000）可对此进行选择。
3. 然后将从模式控制器设为外部时钟模式 1（在 TIMy_SMCR 寄存器中写入 SMS=111）。这样一来，定时器 y 的时钟将由定时器 x 周期性触发信号的上升沿（与定时器 x 的计数器上溢对应）提供。
4. 最后必须通过将这两个定时器的相应 CEN 位（TIMx_CR1 寄存器）置 1 同时使能二者。

有关代码示例，请参见 *A.11.17：将一个定时器用作另一个定时器的预分频器的代码示例*。

注： *如果选择定时器 x 的 OCx 信号作为触发输出 (MMS=1xx)，该信号的上升沿将用于驱动定时器 y 的计数器。*

使用一个定时器使能另一个定时器

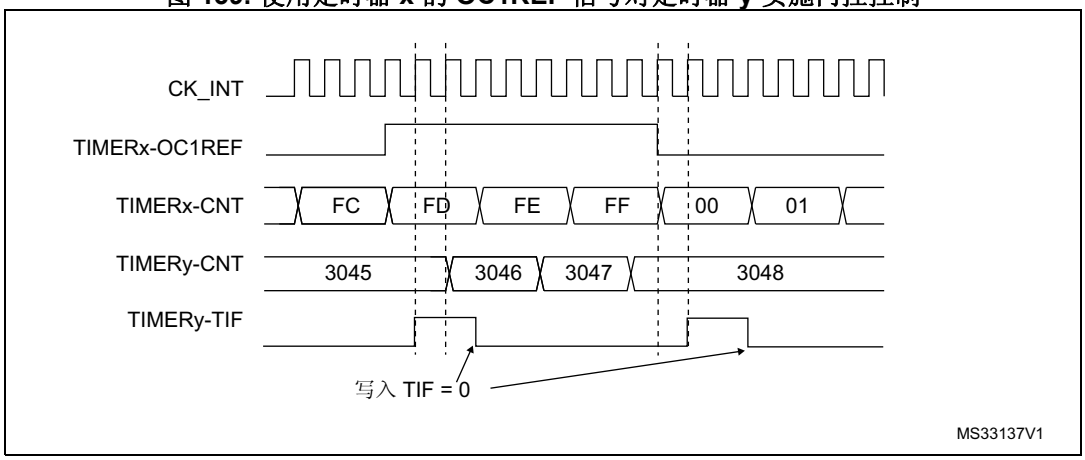
本例中通过定时器 x 的输出比较 1 来使能定时器 y。相关连接图，请参见*图 138*。仅当定时器 x 的 OC1REF 为高电平时，定时器 y 才根据分频后的内部时钟进行计数。两个计数器的时钟频率都基于 CK_INT 通过预分频器执行 3 分频 ($f_{CK_CNT} = f_{CK_INT}/3$)。

1. 将定时器 x 配置为主模式，发送其输出比较 1 参考信号 (OC1REF) 作为触发输出（TIMx_CR2 寄存器中的 MMS=100）。
2. 配置定时器 x 的 OC1REF 波形（TIMx_CCMR1 寄存器）。
3. 配置定时器 y 以接收来自定时器 x 的输入触发（TIMy_SMCR 寄存器中的 TS=000）。
4. 将定时器 y 配置为门控模式（TIMy_SMCR 寄存器中的 SMS=101）。
5. 通过向 CEN 位（TIMy_CR1 寄存器）写入“1”使能定时器 y。
6. 通过向 CEN 位（TIMx_CR1 寄存器）写入“1”启动定时器 x。

有关代码示例，请参见 *A.11.18：使用一个定时器使能另一个定时器的代码示例*。

注： *计数器 2 的时钟与计数器 1 不同步，此模式仅影响定时器 y 的计数器使能信号。*

图 139. 使用定时器 x 的 OC1REF 信号对定时器 y 实施门控控制



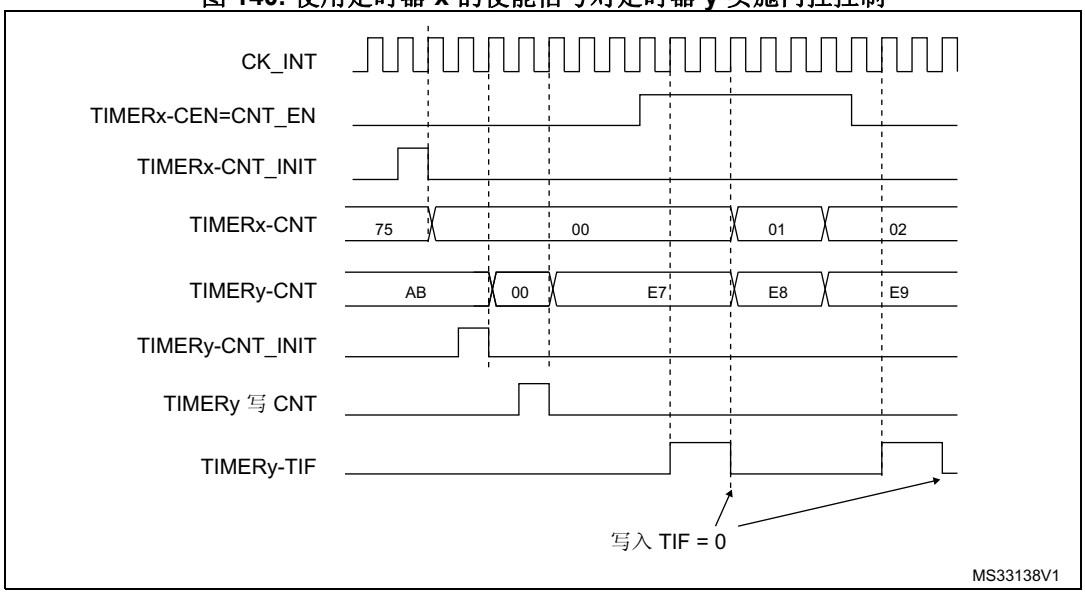
在*图 139* 的示例中，定时器 y 的计数器和预分频器在启动前未进行初始化。因此从各自的当前值开始计数。启动定时器 x 之前，通过复位这两个定时器可以从指定值开始计数。这样便可以在定时器计数器中写入所需的任意值。两个定时器都可通过软件使用 TIMx_EGR 寄存器中的 UG 位轻松复位。

在下一示例中，定时器 x 与定时器 y 同步。定时器 x 为主模式，从 0 开始计数。定时器 y 为从模式，从 0xE7 开始计数。两个定时器的预分频比相同。在 TIMy_CR1 寄存器中通过向 CEN 位写入“0”来禁止定时器 x 时，定时器 y 将停止：

1. 将定时器 x 配置为主模式，发送其输出比较 1 参考信号 (OC1REF) 作为触发输出（TIMx_CR2 寄存器中的 MMS=100）。
2. 配置定时器 x 的 OC1REF 波形（TIMx_CCMR1 寄存器）。
3. 配置定时器 y 以接收来自定时器 x 的输入触发（TIMy_SMCR 寄存器中的 TS=000）。
4. 将定时器 y 配置为门控模式（TIMy_SMCR 寄存器中的 SMS=101）。
5. 通过向 UG 位（TIMx_EGR 寄存器）写入“1”复位定时器 x。
6. 通过向 UG 位（TIMy_EGR 寄存器）写入“1”复位定时器 y。
7. 通过在定时器 y 的计数器 (TIMy_CNTL) 中写入“0xE7”使定时器 y 初始化为 0xE7。
8. 通过向 CEN 位（TIMy_CR1 寄存器）写入“1”使能定时器 y。
9. 通过向 CEN 位（TIMx_CR1 寄存器）写入“1”启动定时器 x。
10. 通过向 CEN 位（TIMx_CR1 寄存器）写入“0”停止定时器 x。

有关代码示例，请参见 *A.11.19：主器件和从器件同步代码示例*。

图 140. 使用定时器 x 的使能信号对定时器 y 实施门控控制

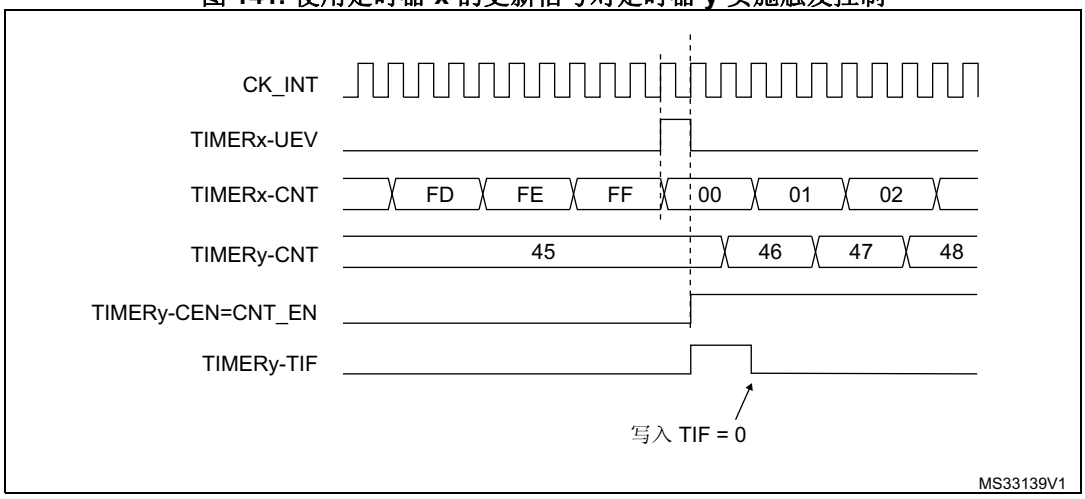


使用一个定时器启动另一个定时器

本例中使用定时器 x 的更新事件使能定时器 y。相关连接图，请参见*图 138*。只要定时器 x 生成更新事件，定时器 y 便根据分频后的内部时钟从当前值（可以不为 0）开始计数。定时器 y 收到触发信号时，其 CEN 位自动置 1，并且计数器开始计数，直到向 TIM2_CR1 寄存器的 CEN 位写入“0”后停止计数。两个计数器的时钟频率都基于 CK_INT 通过预分频器执行 3 分频 ($f_{CK_CNT} = f_{CK_INT}/3$)。

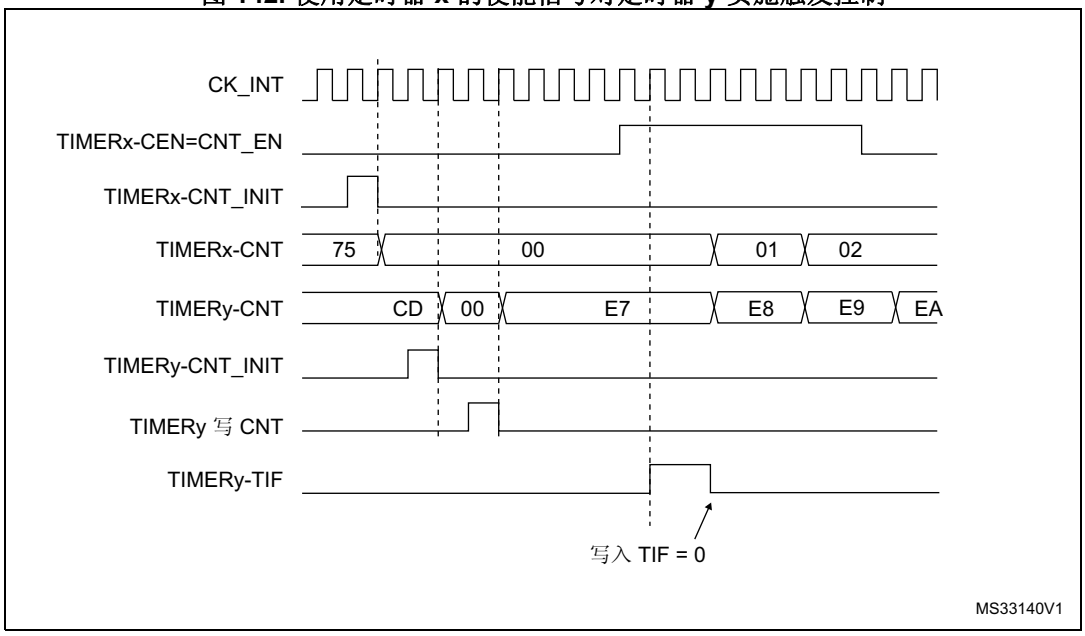
1. 将定时器 x 配置为主模式，发送其更新事件 (UEV) 作为触发输出（TIMx_CR2 寄存器中的 MMS=010）。
2. 配置定时器 x 的周期（TIMx_ARR 寄存器）。
3. 配置定时器 y 以接收来自定时器 x 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
4. 将定时器 y 配置为触发模式（TIM2_SMCR 寄存器中的 SMS=110）。
5. 通过向 CEN 位（TIMx_CR1 寄存器）写入“1”启动定时器 x。

图 141. 使用定时器 x 的更新信号对定时器 y 实施触发控制



如上述示例所示，用户可以在开始计数之前初始化两个计数器。*图 142* 显示了与*图 141* 具有相同配置，只不过处于触发模式（TIMy_SMCR 寄存器中的 SMS=110）而非门控模式的计数行为。

图 142. 使用定时器 x 的使能信号对定时器 y 实施触发控制



使用一个外部触发同步的启动 2 个定时器

本例中，定时器 x 的 TI1 输入出现上升沿时使能定时器 x，使能定时器 x 的同时使能定时器 y。相关连接图，请参见*图 138*。要确保两个计数器对齐，定时器 x 必须配置为主/从模式（对应的 TI1 为从，对应的定时器 y 为主）：

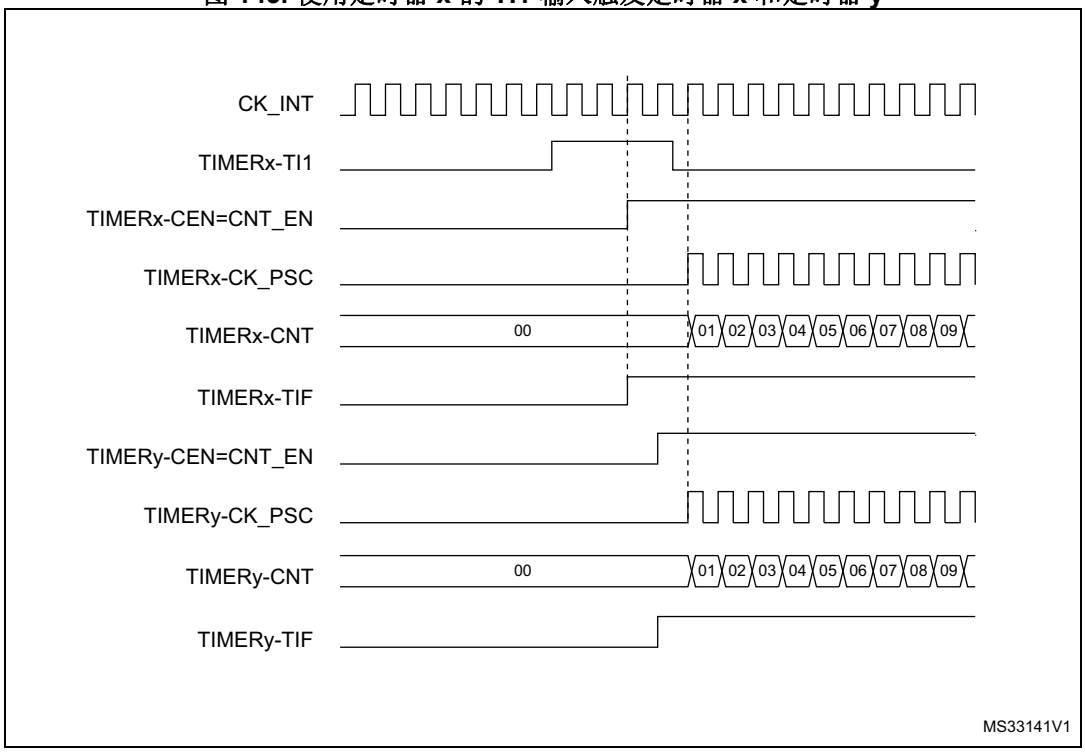
1. 将定时器 x 配置为主模式，发送其使能信号作为触发输出（TIMx_CR2 寄存器中的 MMS=001）。
2. 将定时器 x 配置为从模式以接收来自 TI1 的输入触发（TIMx_SMCR 寄存器中的 TS=100）。
3. 将定时器 x 配置为触发模式（TIMx_SMCR 寄存器中的 SMS=110）。
4. 通过写入 MSM=1（TIMx_SMCR 寄存器）将定时器 x 配置为主/从模式。
5. 配置定时器 y 以接收来自定时器 x 的输入触发（TIM2_SMCR 寄存器中的 TS=000）。
6. 将定时器 y 配置为触发模式（TIM2_SMCR 寄存器中的 SMS=110）。

有关代码示例，请参见 *A.11.20：两个定时器通过一个外部触发信号同步的代码示例*。

当 TI1（定时器 x）出现上升沿时，两个计数器开始根据内部时钟同步计数，并且两个 TIF 标志都置 1。

注： *本例中，两个定时器都在启动之前进行了初始化（通过将各自的 UG 位置 1）。两个计数器都从 0 开始计数，但可以通过对任意一个计数器寄存器 (TIMx_CNT) 进行写操作，在二者之间轻松插入一个偏移量。可注意到主/从模式在定时器 x 的 CNT_EN 与 CK_PSC 之间产生了延迟。*

图 143. 使用定时器 x 的 TI1 输入触发定时器 x 和定时器 y



21.3.16 调试模式

当微控制器进入调试模式时（Cortex®-M0+ 内核停止），TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见 *第 31.6.2 节：支持的音频协议*。

21.4 TIM2/TIM3 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

32 位外设寄存器必须按字（32 位）写入数据。所有其它外设寄存器则必须按半字（16 位）或字（32 位）写入数据。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

21.4.1 TIMx 控制寄存器 1 (TIMx_CR1)

TIMx control register 1

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
Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位[15:10] 保留，必须保持复位值。

位 9:8 **CKD**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与数字滤波器所使用的采样时钟（ETR、TIx）之间的分频比，

00：$t_{DTS} = t_{CK_INT}$
01：$t_{DTS} = 2 \times t_{CK_INT}$
10：$t_{DTS} = 4 \times t_{CK_INT}$
11：保留

位 7 **ARPE**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲
1：TIMx_ARR 寄存器进行缓冲

位 6:5 **CMS**：中心对齐模式选择 (Center-aligned mode selection)

00：边沿对齐模式。计数器根据方向位 (DIR) 递增计数或递减计数。
01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。
11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

注： 只要计数器处于使能状态 (CEN=1)，就不得从边沿对齐模式切换为中心对齐模式。

位 4 **DIR**：方向 (Direction)

0：计数器递增计数
1：计数器递减计数

注： 当定时器配置为中心对齐模式或编码器模式时，该位为只读状态。

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数
1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能时，所有以下事件都会产生更新中断或 DMA 请求。此类事件包括：

– 计数器上溢/下溢
– 将 UG 位置 1
– 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。

0：使能 UEV。更新 (UEV) 事件可通过以下事件之一产生：

– 计数器上溢/下溢
– 将 UG 位置 1
– 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。

1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器
1：使能计数器

注： 只有事先通过软件将 CEN 位置 1，才可以使用外部时钟、门控模式和编码器模式。而触发模式可通过硬件自动将 CEN 位置 1。

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

21.4.2 TIMx 控制寄存器 2 (TIMx_CR2)

TIMx control register 2

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

位 15:8 保留，必须保持复位值。

位 7 **TI1S**：TI1 选择 (TI1 selection)

0：TIMx_CH1 引脚连接到 TI1 输入

1：TIMx_CH1、CH2 和 CH3 引脚连接到 TI1 输入（异或组合）

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位可选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：

000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。

001：**使能**——计数器使能信号 CNT_EN 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号由 CEN 控制位与门控模式下的触发输入的逻辑或运算组合而成。

当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明）。

010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。

011：**比较脉冲**——一旦发生输入捕获或比较匹配事件，当 CC1IF 标志被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(TRGO)

100：**比较**——OC1REF 信号用作触发输出 (TRGO)

101：**比较**——OC2REF 信号用作触发输出 (TRGO)

110：**比较**——OC3REF 信号用作触发输出 (TRGO)

111：**比较**——OC4REF 信号用作触发输出 (TRGO)

注： 必须先使能从定时器或 ADC 的时钟，才能从主定时器接收事件；并且从主定时器接收触发信号时，不得实时更改从定时器或 ADC 的时钟。

位 3 **CCDS**：捕获/比较 DMA 选择 (Capture/compare DMA selection)

0：发生 CCx 事件时发送 CCx DMA 请求

1：发生更新事件时发送 CCx DMA 请求

位 2:0 保留，必须保持复位值。

21.4.3 TIMx 从模式控制寄存器 (TIMx_SMCR)

TIMx slave mode control register

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]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 15 **ETP**：外部触发极性 (External trigger polarity)

此位可选择将 ETR 还是 $\overline{ETR}$ 用于触发操作

0：ETR 未反相，高电平或上升沿有效

1：ETR 反相，低电平或下降沿有效

位 14 **ECE**：外部时钟使能 (External clock enable)

此位可使能外部时钟模式 2。

0：禁止外部时钟模式 2

1：使能外部时钟模式 2。计数器时钟由 ETRF 信号的任意有效边沿提供。

1：将 ECE 位置 1 与选择外部时钟模式 1 并将 TRGI 连接到 ETRF（SMS=111 且 TS=111）具有相同效果。

2：外部时钟模式 2 可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下 TRGI 不得连接 ETRF（TS 位不得为 111）。

3：如果同时使能外部时钟模式 1 和外部时钟模式 2，则外部时钟输入为 ETRF。

位 13:12 **ETPS**：外部触发预分频器 (External trigger prescaler)

外部触发信号 ETRP 频率不得超过 CK_INT 频率的 1/4。可通过使能预分频器来降低 ETRP 频率。这种方法在输入快速外部时钟时非常有用。

00：预分频器关闭

01：2 分频 ETRP 频率

10：4 分频 ETRP 频率

11：8 分频 ETRP 频率

位 11:8 **ETF[3:0]**：外部触发滤波器 (External trigger filter)

此位域可定义 ETRP 信号的采样频率和适用于 ETRP 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样

0001： $f_{SAMPLING}=f_{CK_INT}$，N=2

0010： $f_{SAMPLING}=f_{CK_INT}$，N=4

0011： $f_{SAMPLING}=f_{CK_INT}$，N=8

0100： $f_{SAMPLING}=f_{DTS}/2$，N=6

0101： $f_{SAMPLING}=f_{DTS}/2$，N=8

0110： $f_{SAMPLING}=f_{DTS}/4$，N=6

0111： $f_{SAMPLING}=f_{DTS}/4$，N=8

1000： $f_{SAMPLING}=f_{DTS}/8$，N=6

1001： $f_{SAMPLING}=f_{DTS}/8$，N=8

1010： $f_{SAMPLING}=f_{DTS}/16$，N=5

1011： $f_{SAMPLING}=f_{DTS}/16$，N=6

1100： $f_{SAMPLING}=f_{DTS}/16$，N=8

1101： $f_{SAMPLING}=f_{DTS}/32$，N=5

1110： $f_{SAMPLING}=f_{DTS}/32$，N=6

1111： $f_{SAMPLING}=f_{DTS}/32$，N=8

位 7 **MSM：**主/从模式 (Master/slave mode)

0：不执行任何操作

1：当前定时器的触发输入事件 (TRGI) 的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 TRGO）。此设置适用于由单个外部事件对多个定时器进行同步的情况。

位 6:4 **TS：**触发选择 (Trigger selection)

此位域可选择将要用于同步计数器的触发输入。

000：内部触发 0 (ITR0)。

001：内部触发 1 (ITR1)。

010：内部触发 2 (ITR2)。

011：保留。

100：TI1 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)

有关各定时器 ITRx 含义的详细信息，请参见*第 474 页的表 88：TIM2/TIM3 内部触发连接*。

注：这些位只能在未使用的情况下（例如，SMS=000 时）进行更改，以避免转换时出现错误的边沿检测。

位 3 保留，必须保持为“1”。

位 2:0 **SMS：**从模式选择 (Slave mode selection)

选择外部信号时，触发信号 (TRGI) 的有效边沿与外部输入上所选的极性相关（请参见输入控制寄存器和控制寄存器说明）。

000：禁止从模式——如果 CEN = “1”，预分频器时钟直接由内部时钟提供。

001：编码器模式 1——计数器根据 TI1FP2 电平在 TI2FP1 边沿递增/递减计数。

010：编码器模式 2——计数器根据 TI2FP1 电平在 TI1FP2 边沿递增/递减计数。

011：编码器模式 3——计数器在 TI1FP1 和 TI2FP2 的边沿计数，计数的方向取决于另外一个输入的电平。

100：复位模式——在出现所选触发输入 (TRGI) 上升沿时，重新初始化计数器并生成一个寄存器更新事件。

101：门控模式——触发输入 (TRGI) 为高电平时使能计数器时钟。只要触发输入变为低电平，计数器立即停止计数（但不复位）。计数器的启动和停止都被控制。

110：触发模式——触发信号 TRGI 出现上升沿时启动计数器（但不复位）。只控制计数器的启动。

111：外部时钟模式 1——由所选触发信号 (TRGI) 的上升沿提供计数器时钟。

注：如果将 TI1F_ED 选作触发输入 (TS=100)，则不得使用门控模式。实际上，TI1F 每次转换时，TI1F_ED 都输出 1 个脉冲，而门控模式检查的则是触发信号的电平。

必须先使能从定时器的时钟，才能从主定时器接收事件；并且从主定时器接收触发信号时，不得实时更改从定时器的时钟。

表 88. TIM2/TIM3 内部触发连接

从 TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)
TIM2	TIM21	TIM22	TIM3
TIM3	TIM2	TIM22	TIM21

21.4.4 TIMx DMA/中断使能寄存器 (TIMx_DIER)

TIMx DMA/Interrupt enable register

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
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位 15 保留，必须保持复位值。

位 14 **TDE**：触发 DMA 请求使能 (Trigger DMA request enable)
0：禁止触发 DMA 请求。
1：使能触发 DMA 请求。

位 13 保留，始终读为 0

位 12 **CC4DE**：捕获/比较 4 DMA 请求使能 (Capture/Compare 4 DMA request enable)
0：禁止 CC4 DMA 请求。
1：使能 CC4 DMA 请求。

位 11 **CC3DE**：捕获/比较 3 DMA 请求使能 (Capture/Compare 3 DMA request enable)
0：禁止 CC3 DMA 请求。
1：使能 CC3 DMA 请求。

位 10 **CC2DE**：捕获/比较 2 DMA 请求使能 (Capture/Compare 2 DMA request enable)
0：禁止 CC2 DMA 请求。
1：使能 CC2 DMA 请求。

位 9 **CC1DE**：捕获/比较 1 DMA 请求使能 (Capture/Compare 1 DMA request enable)
0：禁止 CC1 DMA 请求。
1：使能 CC1 DMA 请求。

位 8 **UDE**：更新 DMA 请求使能 (Update DMA request enable)
0：禁止更新 DMA 请求。
1：使能更新 DMA 请求。

位 7 保留，必须保持复位值。

位 6 **TIE**：触发中断使能 (Trigger interrupt enable)
0：禁止触发中断。
1：使能触发中断。

位 5 保留，必须保持复位值。

位 4 **CC4IE**：捕获/比较 4 中断使能 (Capture/Compare 4 interrupt enable)
0：禁止 CC4 中断。
1：使能 CC4 中断。

位 3 **CC3IE**：捕获/比较 3 中断使能 (Capture/Compare 3 interrupt enable)
0：禁止 CC3 中断
1：使能 CC3 中断

位 2 **CC2IE**：捕获/比较 2 中断使能 (Capture/Compare 2 interrupt enable)
0：禁止 CC2 中断
1：使能 CC2 中断

位 1 **CC1IE**：捕获/比较 1 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC1 中断
1：使能 CC1 中断

位 0 **UIE**：更新中断使能 (Update interrupt enable)
0：禁止更新中断
1：使能更新中断

21.4.5 TIMx 状态寄存器 (TIMx_SR)

TIMx status register

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
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15:13 保留，必须保持复位值。

位 12 **CC4OF**：捕获/比较 4 重复捕获标志 (Capture/Compare 4 overcapture flag)
请参见 CC1OF 说明

位 11 **CC3OF**：捕获/比较 3 重复捕获标志 (Capture/Compare 3 overcapture flag)
请参见 CC1OF 说明

位 10 **CC2OF**：捕获/比较 2 重复捕获标志 (Capture/compare 2 overcapture flag)
请参见 CC1OF 说明

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)
仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。
0：未检测到重复捕获
1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8:7 保留，必须保持复位值。

位 6 **TIF**：触发中断标志 (Trigger interrupt flag)
在除门控模式以外的所有模式下，当使能从模式控制器后在 TRGI 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。
0：未发生触发事件
1：触发中断挂起

位 5 保留，必须保持复位值。

位 4 **CC4IF**：捕获/比较 4 中断标志 (Capture/Compare 4 interrupt flag)
请参见 CC1IF 说明

位 3 **CC3IF**：捕获/比较 3 中断标志 (Capture/Compare 3 interrupt flag)
请参见 CC1IF 说明

位 2 **CC2IF**：捕获/比较 2 中断标志 (Capture/Compare 2 interrupt flag)
请参见 CC1IF 说明

位 1 **CC1IF**：捕获/比较 1 中断标志 (Capture/compare 1 interrupt flag)

如果通道 CC1 配置为输出：

当计数器与比较值匹配时，此标志由硬件置 1，中心对齐模式下除外（请参见 TIMx_CR1 寄存器中的 CMS 位说明）。但需要通过软件清零。

0：不匹配

1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢（递增计数模式和增减计数模式下）或下溢（递减计数模式下）时变为高电平。

如果通道 CC1 配置为输入：

此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。

0：未发生输入捕获事件

1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）

位 0 **UIF**：更新中断标志 (Update interrupt flag)

″ 该位在发生更新事件时通过硬件置 1。但需要通过软件清零。
0：未发生更新。
1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

″ 上溢或下溢并且当 TIMx_CR1 寄存器中 UDIS = 0 时。

″ TIMx_CR1 寄存器中的 URS = 0 且 UDIS = 0，并且由软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

TIMx_CR1 寄存器中的 URS=0 且 UDIS=0，并且 CNT 由触发事件重新初始化时（参见同步控制寄存器说明）。

21.4.6 TIMx 事件生成寄存器 (TIMx_EGR)

TIMx event generation register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位 15:7 保留，必须保持复位值。

位 6 **TG**：触发生成 (Trigger generation)
此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断或 DMA 传输事件。

位 5 保留，必须保持复位值。

位 4 **CC4G**：捕获/比较 4 生成 (Capture/Compare 4 generation)
请参见 CC1G 说明

位 3 **CC3G**：捕获/比较 3 生成 (Capture/Compare 3 generation)
请参见 CC1G 说明

位 2 **CC2G**：捕获/比较 2 生成 (Capture/compare 2 generation)
请参见 CC1G 说明

位 1 **CC1G**：捕获/比较 1 生成 (Capture/Compare 1 generation)
此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器当前值。使能时，CC1IF 标志置 1 并发送相应的中断或 DMA 请求。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)
该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作
1：重新初始化计数器并生成寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。如果选择中心对齐模式或 DIR=0（递增计数），计数器将清零；如果 DIR=1（递减计数），计数器将使用自动重载值 (TIMx_ARR)。

21.4.7 TIMx 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIMx capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。此寄存器的所有其它位在输入模式和输出模式下的功能均不同。对于任一给定位，OCxx 用于说明通道配置为输出时该位对应的功能，ICxx 则用于说明通道配置为输入时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

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 15 **OC2CE**：输出比较 2 清零使能 (Output Compare 2 clear enable)

位 14:12 **OC2M[2:0]**：输出比较 2 模式 (Output Compare 2 mode)

位 11 **OC2PE**：输出比较 2 预装载使能 (Output Compare 2 preload enable)

位 10 **OC2FE**：输出比较 2 快速使能 (Output Compare 2 fast enable)

位 9:8 **CC2S[1:0]**：捕获/比较 2 选择 (Capture/Compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，IC2 映射到 TI2 上

10：CC2 通道配置为输入，IC2 映射到 TI1 上

11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7 **OC1CE**：输出比较 1 清零使能 (Output Compare 1 clear enable)

OC1CE：输出比较 1 清零使能 (Output Compare 1 Clear Enable)

0：OC1Ref 不受 ETRF 输入影响

1：ETRF 输入上检测到高电平时，OC1Ref 立即清零

位 6:4 **OC1M**：输出比较 1 模式 (Output Compare 1 mode)

这些位定义提供 OC1 和 OC1N 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 和 OC1N 的有效电平则取决于 CC1P 位和 CC1NP 位。

000：冻结——输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。

010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。

011：翻转——TIMx_CNT=TIMx_CCR1 时，OC1REF 发生翻转。

100：强制变为无效电平——OC1REF 强制变为低电平。

101：强制变为有效电平——OC1REF 强制变为高电平。

110：PWM 模式 1——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态 (OC1REF=0)，否则为有效状态 (OC1REF=1)。

111：PWM 模式 2——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

注：在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。

位 3 **OC1PE**：输出比较 1 预装载使能 (Output Compare 1 preload enable)

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中。

注：只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。

位 2 **OC1FE**：输出比较 1 快速使能 (Output Compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期。

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OCFE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 Selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出。

01：CC1 通道配置为输入，IC1 映射到 TI1 上。

10：CC1 通道配置为输入，IC1 映射到 TI2 上。

11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

输入捕获模式

位 15:12 **IC2F**：输入捕获 2 滤波器 (Input capture 2 filter)

位 11:10 **IC2PSC[1:0]**：输入捕获 2 预分频器 (Input capture 2 prescaler)

位 9:8 **CC2S**：捕获/比较 2 选择 (Capture/Compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC2 通道配置为输出。
01：CC2 通道配置为输入，IC2 映射到 TI2 上。
10：CC2 通道配置为输入，IC2 映射到 TI1 上。
11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7:4 **IC1F**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样
0001：$f_{SAMPLING}=f_{CK_INT}$，N=2
0010：$f_{SAMPLING}=f_{CK_INT}$，N=4
0011：$f_{SAMPLING}=f_{CK_INT}$，N=8
0100：$f_{SAMPLING}=f_{DTS}/2$，N=6
0101：$f_{SAMPLING}=f_{DTS}/2$，N=8
0110：$f_{SAMPLING}=f_{DTS}/4$，N=6
0111：$f_{SAMPLING}=f_{DTS}/4$，N=8
1000：$f_{SAMPLING}=f_{DTS}/8$，N=6
1001：$f_{SAMPLING}=f_{DTS}/8$，N=8
1010：$f_{SAMPLING}=f_{DTS}/16$，N=5
1011：$f_{SAMPLING}=f_{DTS}/16$，N=6
1100：$f_{SAMPLING}=f_{DTS}/16$，N=8
1101：$f_{SAMPLING}=f_{DTS}/32$，N=5
1110：$f_{SAMPLING}=f_{DTS}/32$，N=6
1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。
只要 CC1E=0（TIMx_CCER 寄存器），预分频器便立即复位。
00：无预分频器，捕获输入上每检测到一个边沿便执行捕获
01：每发生 2 个事件便执行一次捕获
10：每发生 4 个事件便执行一次捕获
11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 Selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC1 通道配置为输出
01：CC1 通道配置为输入，IC1 映射到 TI1 上
10：CC1 通道配置为输入，IC1 映射到 TI2 上
11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

21.4.8 TIMx 捕获/比较模式寄存器 2 (TIMx_CCMR2)

TIMx capture/compare mode register 2

偏移地址：0x1C

复位值：0x0000

请参见上述 CCMR1 寄存器说明。

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 15 OC4CE：输出比较 4 清零使能 (Output compare 4 clear enable)

位 14:12 **OC4M**：输出比较 4 模式 (Output compare 4 mode)

位 11 **OC4PE**：输出比较 4 预装载使能 (Output compare 4 preload enable)

位 10 **OC4FE**：输出比较 4 快速使能 (Output compare 4 fast enable)

位 9:8 **CC4S**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC4 通道配置为输出
01：CC4 通道配置为输入，IC4 映射到 TI4 上
10：CC4 通道配置为输入，IC4 映射到 TI3 上
11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据。

位 7 **OC3CE**：输出比较 3 清零使能 (Output compare 3 clear enable)

位 6:4 **OC3M**：输出比较 3 模式 (Output compare 3 mode)

位 3 **OC3PE**：输出比较 3 预装载使能 (Output compare 3 preload enable)

位 2 **OC3FE**：输出比较 3 快速使能 (Output compare 3 fast enable)

位 1:0 **CC3S**：捕获/比较 3 选择 (Capture/compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。
00：CC3 通道配置为输出
01：CC3 通道配置为输入，IC3 映射到 TI3 上
10：CC3 通道配置为输入，IC3 映射到 TI4 上
11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据。

输入捕获模式

位 15:12 **IC4F**：输入捕获 4 滤波器 (Input capture 4 filter)

位 11:10 **IC4PSC**：输入捕获 4 预分频器 (Input capture 4 prescaler)

位 9:8 **CC4S**：捕获/比较 4 选择 (Capture/Compare 4 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC4 通道配置为输出

01：CC4 通道配置为输入，IC4 映射到 TI4 上

10：CC4 通道配置为输入，IC4 映射到 TI3 上

11：CC4 通道配置为输入，IC4 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC4E = 0），才可向 CC4S 位写入数据。

位 7:4 **IC3F**：输入捕获 3 滤波器 (Input capture 3 filter)

位 3:2 **IC3PSC**：输入捕获 3 预分频器 (Input capture 3 prescaler)

位 1:0 **CC3S**：捕获/比较 3 选择 (Capture/compare 3 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC3 通道配置为输出

01：CC3 通道配置为输入，IC3 映射到 TI3 上

10：CC3 通道配置为输入，IC3 映射到 TI4 上

11：CC3 通道配置为输入，IC3 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC3E = 0），才可向 CC3S 位写入数据。

21.4.9 TIMx 捕获/比较使能寄存器 (TIMx_CCER)

TIMx capture/compare enable register

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	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

位 15 **CC4NP**：*捕获/比较 4 输出极性 (Capture/Compare 4 output Polarity)*

请参见 CC1NP 说明

位 14 保留，必须保持复位值。

位 13 **CC4P**：*捕获/比较 4 输出极性 (Capture/Compare 4 output Polarity)*

请参见 CC1P 说明

位 12 **CC4E**：*捕获/比较 4 输出使能 (Capture/Compare 4 output enable)*

请参见 CC1E 说明

位 11 **CC3NP**：*捕获/比较 3 输出极性 (Capture/Compare 3 output Polarity)*

请参见 CC1NP 说明

位 10 保留，必须保持复位值。

位 9 **CC3P**：*捕获/比较 3 输出极性 (Capture/Compare 3 output polarity)*

请参见 CC1P 说明

位 8 **CC3E**：*捕获/比较 3 输出使能 (Capture/Compare 3 output enable)*
请参见 CC1E 说明

位 7 **CC2NP**：*捕获/比较 2 输出极性 (Capture/Compare 2 output Polarity)*
请参见 CC1NP 说明

位 6 保留，必须保持复位值。

位 5 **CC2P**：*捕获/比较 2 输出极性 (Capture/Compare 2 output polarity)*
请参见 CC1P 说明

位 4 **CC2E**：*捕获/比较 2 输出使能 (Capture/Compare 2 output enable)*
请参见 CC1E 说明

位 3 **CC1NP**：*捕获/比较 1 输出极性 (Capture/Compare 1 output Polarity)*
CC1 通道配置为输出：
在这种情况下，CC1NP 必须保持清零。
CC1 通道配置为输入：
此位与 CC1P 配合使用，用以定义 TI1FP1/TI2FP1 的极性。请参见 CC1P 说明。

位 2 保留，必须保持复位值。

位 1 **CC1P**：*捕获/比较 1 输出极性 (Capture/Compare 1 output Polarity)*
CC1 通道配置为输出：
0：OC1 高电平有效
1：OC1 低电平有效
CC1 通道配置为输入：
CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。
00：未反相/上升沿触发
电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。
01：反相/下降沿触发
电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。
10：保留，不使用此配置。
11：未反相/上升沿和下降沿均触发
电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

位 0 **CC1E**：*捕获/比较 1 输出使能 (Capture/Compare 1 output enable)*。
CC1 通道配置为输出：
0：关闭——OC1 未激活
1：开启——在相应输出引脚上输出 OC1 信号
CC1 通道配置为输入：
此位决定了是否可以实际将计数器值捕获到输入捕获/比较寄存器 1 (TIMx_CCR1) 中。
0：禁止捕获
1：使能捕获

表 89. 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：与标准 OCx 通道相连的外部 I/O 引脚的状态取决于通道 OCx 的状态以及 GPIO 寄存器。

21.4.10 TIMx 计数器 (TIMx_CNT)

TIMx counter

偏移地址：0x24

复位值：0x0000

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

位 15:0 **CNT[15:0]**：低计数器值 (Low counter value)

21.4.11 TIMx 预分频器 (TIMx_PSC)

TIMx prescaler

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 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

21.4.12 TIMx 自动重载寄存器 (TIMx_ARR)

TIMx auto-reload register

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 15:0 **ARR[15:0]**：低自动重载值 (Low Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的更多详细信息，请参见*第 431 页的第 21.3.1 节：时基单元*。

当自动重载值为空时，计数器不工作。

21.4.13 TIMx 捕获/比较寄存器 1 (TIMx_CCR1)

TIMx capture/compare register 1

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 15:0 **CCR1[15:0]**：捕获/比较 1 值的低 16 位 (Low Capture/Compare 1 value)

如果通道 CC1 配置为输出：
CCR1 是捕获/比较寄存器 1 的预装载值。
如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 1）。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。

如果通道 CC1 配置为输入：
CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

21.4.14 TIMx 捕获/比较寄存器 2 (TIMx_CCR2)

TIMx capture/compare register 2

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 15:0 **CCR2[15:0]**：捕获/比较 2 值的低 16 位 (Low Capture/Compare 2 value)

如果通道 CC2 配置为输出：
CCR2 是捕获/比较寄存器 2 的预装载值。
如果没有通过 TIMx_CCMR2 寄存器中的 OC2PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 2）。
实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC2 输出上发出信号的值。
如果通道 CC2 配置为输入：
CCR2 为上一个输入捕获 2 事件 (IC2) 发生时的计数器值。

21.4.15 TIMx 捕获/比较寄存器 3 (TIMx_CCR3)

TIMx capture/compare register 3

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 15:0 **CCR3[15:0]**：捕获/比较 3 值的低 16 位 (Low Capture/Compare value)

如果通道 CC3 配置为输出：

CCR3 是捕获/比较寄存器 3 的预装载值。

如果没有通过 TIMx_CCMR3 寄存器中的 OC3PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 3）。

实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC3 输出上发出信号的值。

如果通道 CC3 配置为输入：

CCR3 为上一个输入捕获 3 事件 (IC3) 发生时的计数器值。

21.4.16 TIMx 捕获/比较寄存器 4 (TIMx_CCR4)

TIMx capture/compare register 4

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 15:0 **CCR4[15:0]**：捕获/比较 4 值的低 16 位 (Low Capture/Compare value)

1. 如果 CC4 通道配置为输出（CC4S 位）：
   CCR4 是捕获/比较寄存器 4 的预装载值。
   如果没有通过 TIMx_CCMR4 寄存器中的 OC4PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 4）。
   实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC4 输出上发出信号的值。
2. 如果 CC4 通道配置为输入（TIMx_CCMR4 寄存器中的 CC4S 位）：
   CCR4 为上一个输入捕获 4 事件 (IC4) 发生时的计数器值。

21.4.17 TIMx DMA 控制寄存器 (TIMx_DCR)

TIMx DMA control register

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
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

位 15:13 保留，必须保持复位值。

位 12:8 **DBL[4:0]**：DMA 连续传送长度 (DMA burst length)

该 5 位向量定义了 DMA 的传送次数（当对 TIMx_DMAR 寄存器进行读或写时，定时器进行一次连续传送）。
00000：1 次传送，
00001：2 次传送，
00010：3 次传送，
...
10001：18 次传送。

位 7:5 保留，必须保持复位值。

位 4:0 **DBA[4:0]**：DMA 基址 (DMA base address)

该 5 位向量定义 DMA 传输的基址（通过 TIMx_DMAR 地址进行读/写访问时）。DBA 定义为从 TIMx_CR1 寄存器地址开始计算的偏移量。
示例：
00000：TIMx_CR1，
00001：TIMx_CR2，
00010：TIMx_SMCR，
...

示例：以下面的传送为例：DBL = 7 次传送且 DBA = TIMx_CR1。这种情况下将向/从自 TIMx_CR1 地址开始的 7 个寄存器传输数据。

21.4.18 TIMx 全传输 DMA 地址 (TIMx_DMAR)

TIMx DMA address for full transfer

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 15:0 **DMAB[15:0]**：DMA 连续传送寄存器 (DMA register for burst accesses)

对 DMAR 寄存器执行读或写操作将访问位于如下地址的寄存器

(TIMx_CR1 地址) + (DBA + DMA 索引) x 4

其中 TIMx_CR1 地址为控制寄存器 1 的地址，DBA 为 TIMx_DCR 寄存器中配置的 DMA 基址，DMA 索引由 DMA 传输自动控制，其范围介于 0 到 DBL（TIMx_DCR 寄存器中配置的 DBL）之间。

DMA 连续传送功能使用方法示例

本例中，定时器 DMA 连续传送功能用于将 CCRx 寄存器（x = 2、3、4）的内容更新为通过 DMA 传输到 CCRx 寄存器中的多个半字。

具体操作步骤如下：

1. 将相应的 DMA 通道配置如下：
   - DMA 通道外设地址为 DMAR 寄存器地址
   - DMA 通道存储器地址为包含要通过 DMA 传输到 CCRx 寄存器的数据的 RAM 缓冲区地址。
   - 要传输的数据量 = 3（参见下文注释）。
   - 禁止循环模式。
2. 通过将 DBA 和 DBL 位域配置如下来配置 DCR 寄存器：
   DBL = 3 次传输，DBA = 0xE。
3. 使能 TIMx 更新 DMA 请求（DIER 寄存器中的 UDE 位置 1）。
4. 使能 TIMx
5. 使能 DMA 通道

有关代码示例，请参见 *A.11.21：DMA 连续传送功能代码示例*。

注： *本例适用于每个 CCRx 寄存器只更新一次的情况。如果每个 CCRx 寄存器要更新两次，则要传输的数据量应为 6。下面以包含 data1、data2、data3、data4、data5 和 data6 的 RAM 缓冲区为例。数据将按照如下方式传输到 CCRx 寄存器：在第一个更新 DMA 请求期间，data1 传输到 CCR2，data2 传输到 CCR3，data3 传输到 CCR4；在第二个更新 DMA 请求期间，data4 传输到 CCR2，data5 传输到 CCR3，data6 传输到 CCR4。*

21.4.19 TIM2 选项寄存器 (TIM2_OR)

TIM2 option register

偏移地址：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP		ETR_RMP		
											rw	rw	rw	rw	rw

位 15:5 保留，必须保持复位值。

位 4:3 **TI4_RMP**：内部触发（TI4 连接到 TIM2_CH4）重映射 (Internal trigger (TI4 connected to TIM2_CH4) remap)

此位由软件置 1 和清零。

01：TIM2 TI4 输入连接到 COMP2_OUT

10：TIM2 TI4 输入连接到 COMP1_OUT

其它：TIM2 TI4 输入连接到经过或运算的 GPIO。请参见器件数据手册中的复用功能映射表。

位 2:0 **ETR_RMP**：Timer2 ETR 重映射 (Timer2 ETR remap)

此位由软件置 1 和清零。

111：TIM2 ETR 输入连接到 COMP1_OUT

110：TIM2 ETR 输入连接到 COMP2_OUT

101：TIM2 ETR 输入连接到 LSE

100：TIM2 ETR 输入连接到 HSI48（请参见下方注释）

011：当*时钟控制寄存器 (RCC_CR)* 中的 HSI16OUTEN 位置 1 时，TIM2 ETR 输入连接到 HSI16（第 3 类器件除外）

其它：TIM2 ETR 输入连接到经过或运算的 GPIO。请参见器件数据手册中的复用功能映射表。

注：当 TIM2 ETR 通过 HSI48 提供时钟时，由于最大系统频率为 32 MHz，因此此 ETR 必须在内部预分频到 TIMER2。

21.4.20 TIM3 选项寄存器 (TIM3_OR)

TIM3 option register

偏移地址：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI_RMP			ETR_RMP	
											rw	rw	rw	rw	rw

位 15:5 保留，必须保持复位值。

位 4 **TI_RMP**：Timer3 重映射到 PC9 (Timer3 remapping on PC9)
此位由软件置 1 和清零。
1：选择 TIM3_CH4
0：选择 USB_OE

位 3 **TI_RMP**：Timer3 重映射到 PB5 (Timer3 remap on PB5)
此位由软件置 1 和清零。
1：选择 TIM3_CH2
0：选择 TIM22_CH2

位 2 **TI_RMP**：Timer3 TI 重映射 (Timer3 TI remap)
此位由软件置 1 和清零。
1：TIM3_TI1 输入连接到 PE3、PA6、PC6 或 PB4
0：TIM3_TI1 输入连接到 USB_SOF

位 1:0 **ETR_RMP**：Timer3 ETR 重映射 (Timer3 ETR remap)
这些位由软件置 1 和清零。
10：如果 HSI48DIV6EN 位置 1，则 TIM3_ETR 输入连接到 6 分频的 HSI48
（请参见*第 7.3.3 节：时钟恢复 RC 寄存器 (RCC_CRRCR)*）
其它配置：TIM3_ETR 输入连接到 PE2 或 PD2

21.5 TIMx 寄存器映射

TIMx 寄存器按照下表所述方式映射：

表 90. TIM2/3 寄存器映射和复位值

偏移	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	CKD [1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value									0	0	0	0	0			
0x08	**TIMx_SMCR**	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value		0		0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value										0		0	0	0	0	0
0x18	**TIMx_CCMR1 Output Compare mode**	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2 Output Compare mode**	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value	0		0	0	0		0	0	0		0	0	0		0	0
0x24	**TIMx_CNT**	CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	PSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 90. TIM2/3 寄存器映射和复位值（续）

偏移	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIMx_ARR**	ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Res.																
0x34	**TIMx_CCR1**	CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Res.																
0x48	**TIMx_DCR**	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	DMAB[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM2_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI4_RMP		ETR_RMP	
	Reset value													0	0	0	0
0x50	**TIM3_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI_RMP			ETR_RMP	
	Reset value												0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

22 通用定时器 (TIM21/22)

22.1 简介

TIM21/22 通用定时器包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

它们可用于多种用途，包括测量输入信号的脉冲宽度（输入捕获），或者生成输出波形（输出比较、PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，可将脉冲宽度和波形周期从几微秒调制到几毫秒。

TIM21/22 定时器彼此完全独立，不共享任何资源。如*第 22.3.14 节*中所述，它们可以同步操作。

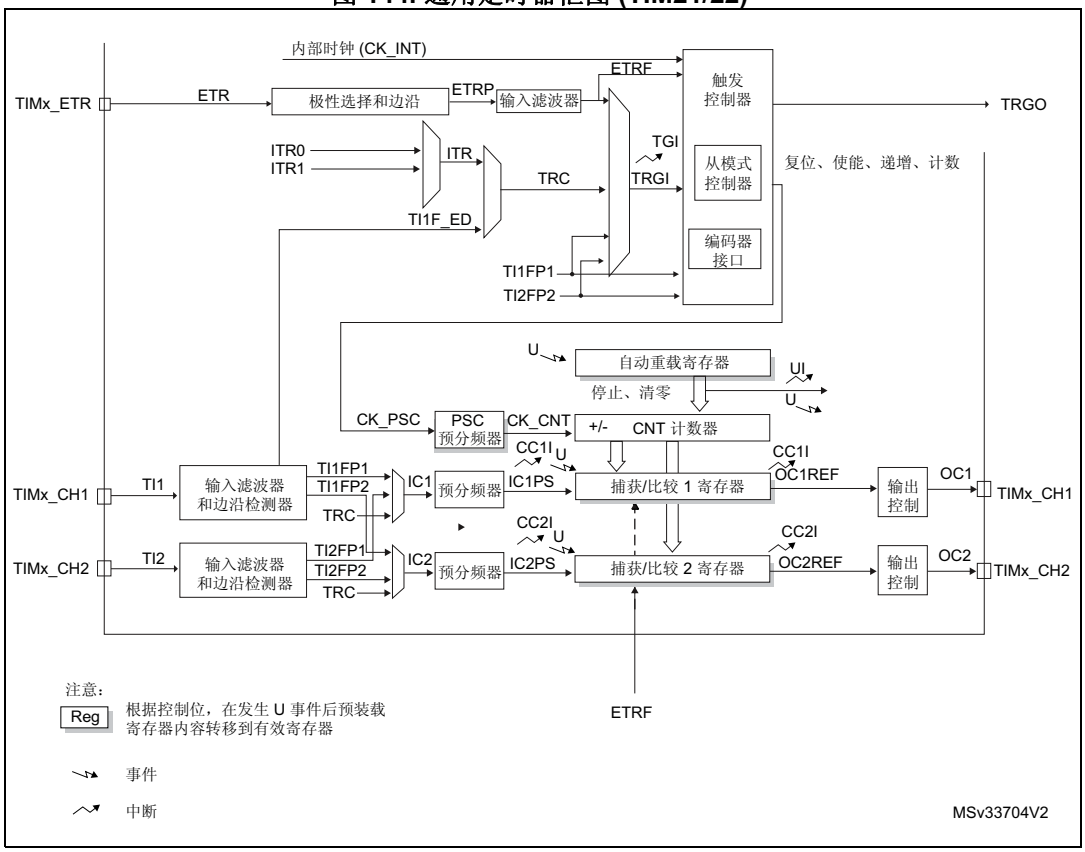
22.2 TIM21/22 主要特性

22.2.1 TIM21/22 主要特性

TIM21/22 通用定时器具有以下特性：

- 16 位递增、递减、递增/递减自动重载计数器
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65535 之间
- 多达 2 个独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿和中心对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器且可实现多个定时器互连的同步电路
- 发生如下事件时生成中断：
  - 更新：计数器上溢/下溢、计数器初始化（通过软件或内部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部触发计数）
  - 输入捕获
  - 输出比较

图 144. 通用定时器框图 (TIM21/22)



22.3 TIM21/22 功能说明

22.3.1 时基单元

定时器的主要模块由一个 16 位计数器及其相关的自动重载寄存器组成。计数器可递增计数、递减计数或同时递增和递减计数，也可递减计数或同时递增和递减计数。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。对自动重载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 (UEV) 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器（有关计数器使能的更多详细信息，另请参见从模式控制器的相关说明）。

注意，计数器将在 TIMx_CR1 寄存器的 CEN 位置 1 时刻的一个时钟周期后开始计数。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于该控制寄存器具有缓冲功能，因此预分频器可实现实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 146 和 *图 147* 以一些示例说明在预分频比实时变化时计数器的行为。

图 145. 预分频器分频由 1 变为 2 时的计数器时序图

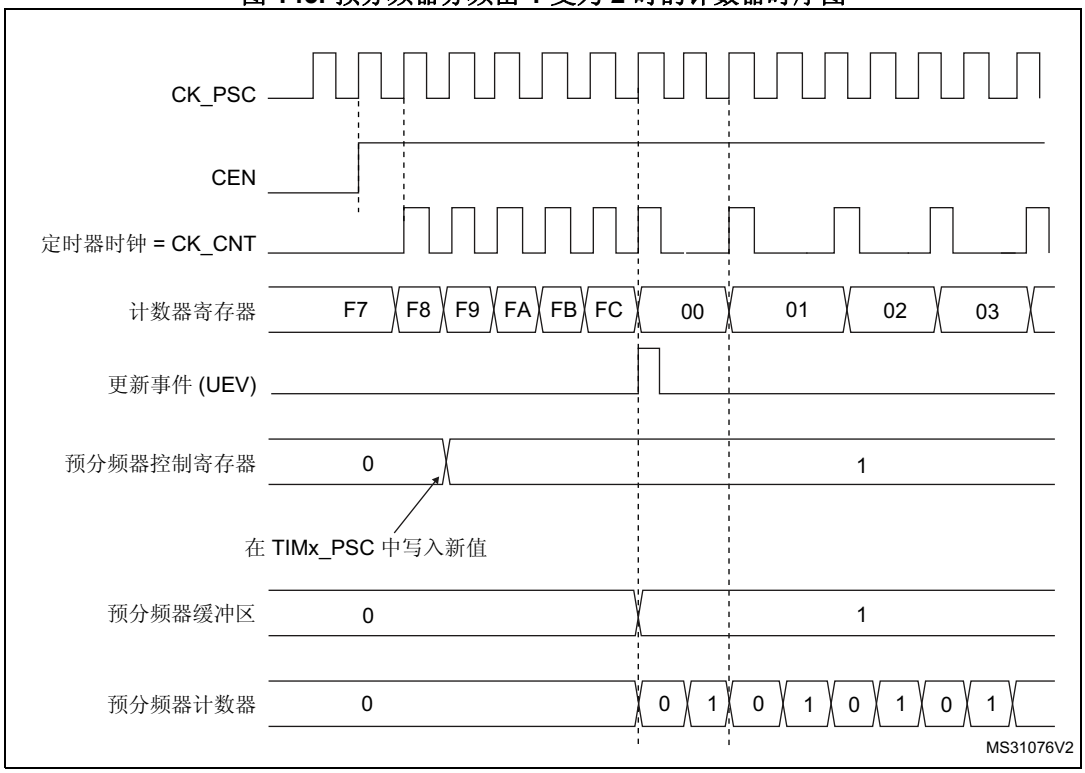
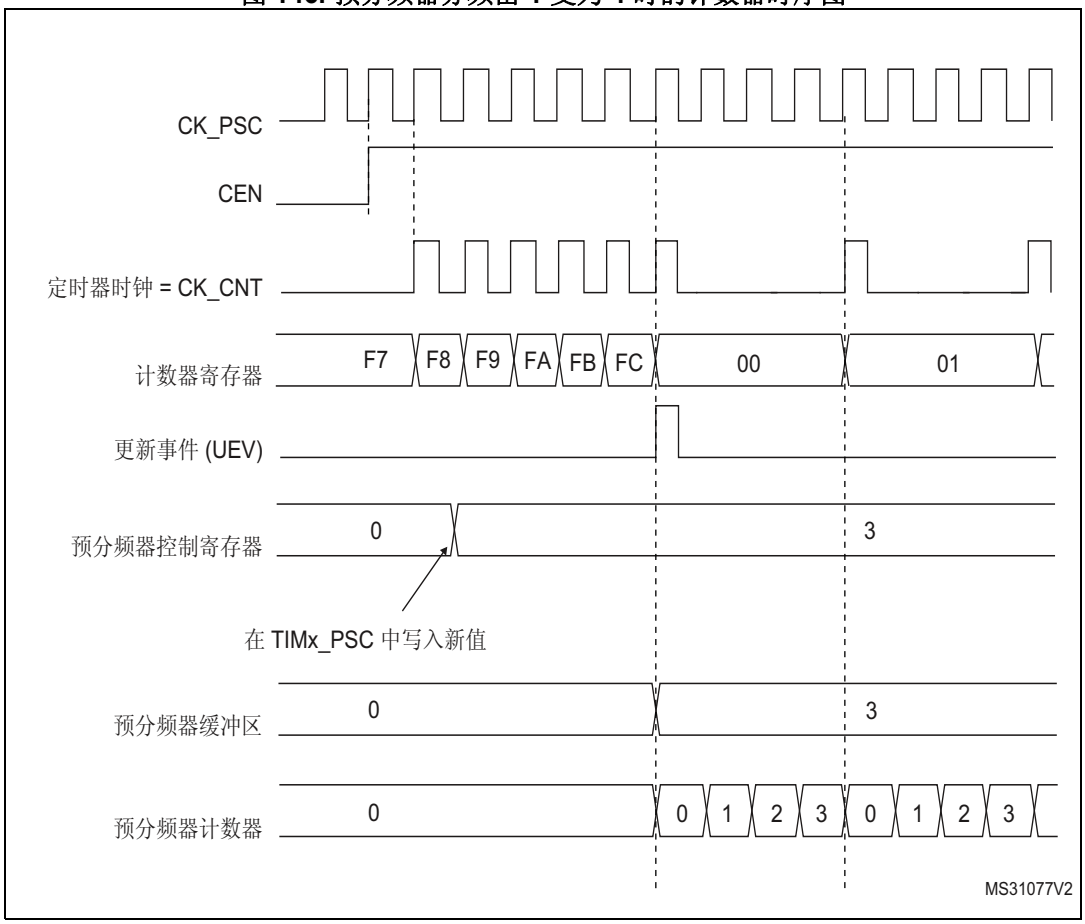


图 146. 预分频器分频由 1 变为 4 时的计数器时序图



22.3.2 计数器模式

递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用 TIM21/22 上的从模式控制器）也会生成更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 自动重载影子寄存器将以预装载值 (TIMx_ARR) 进行更新，
- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 147. 计数器时序图，1 分频内部时钟

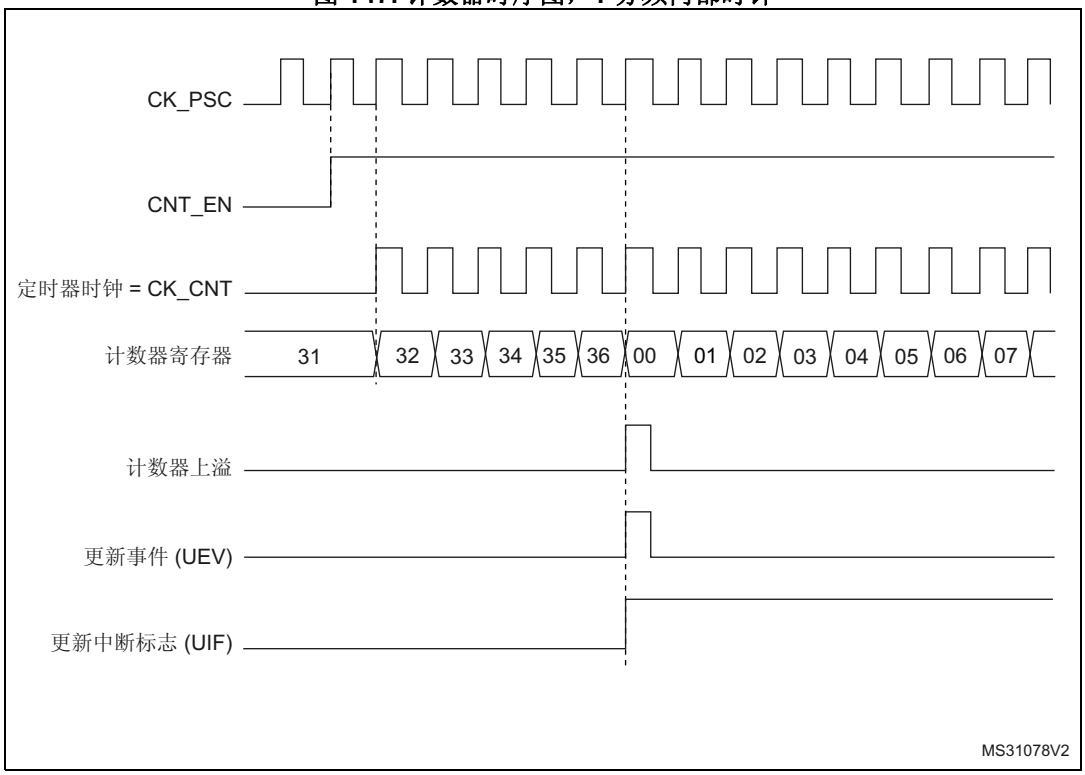


图 148. 计数器时序图，2 分频内部时钟

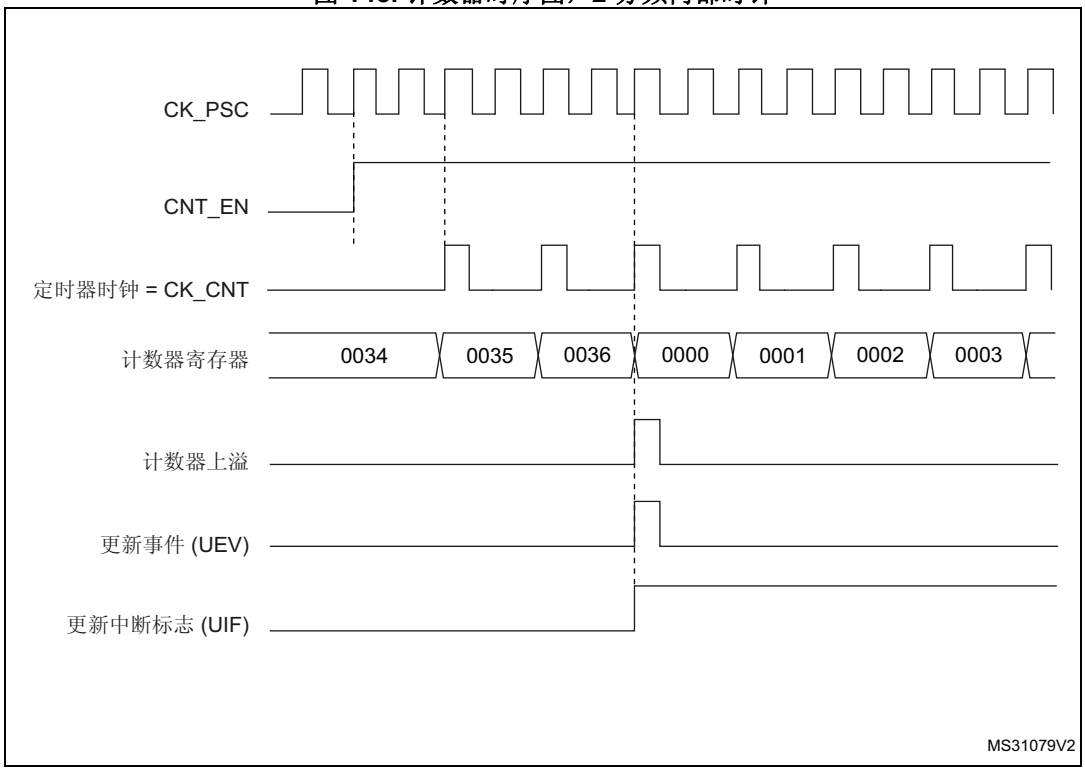


图 149. 计数器时序图，4 分频内部时钟

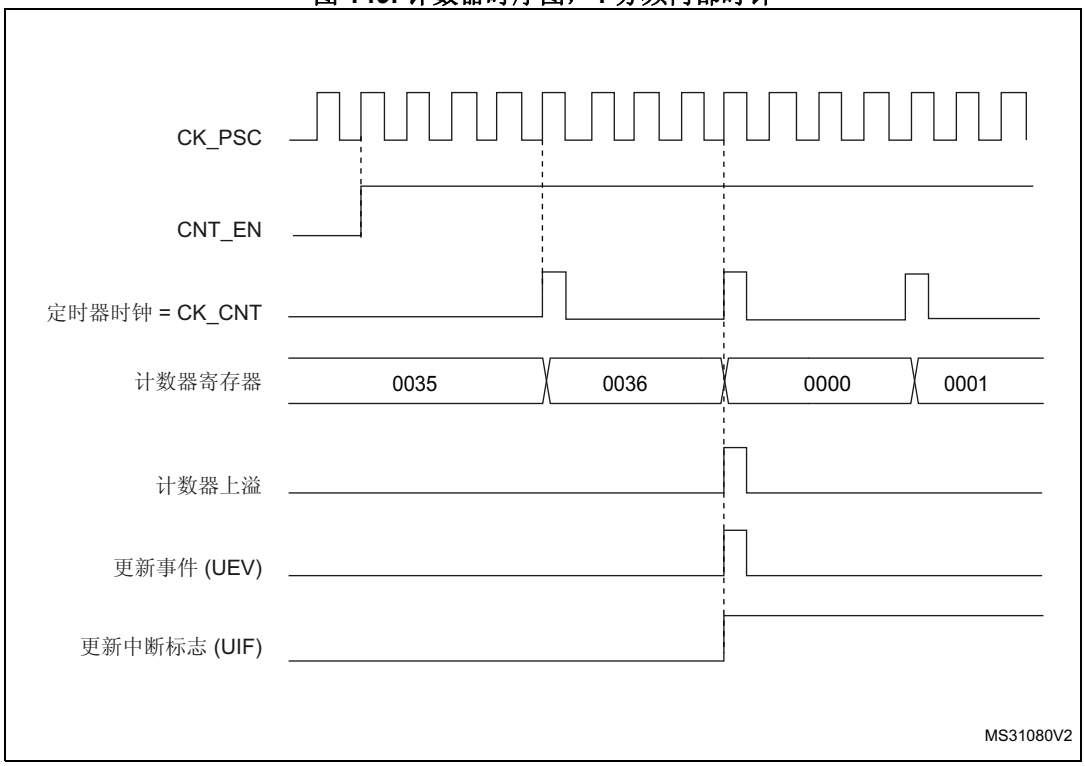


图 150. 计数器时序图，N 分频内部时钟

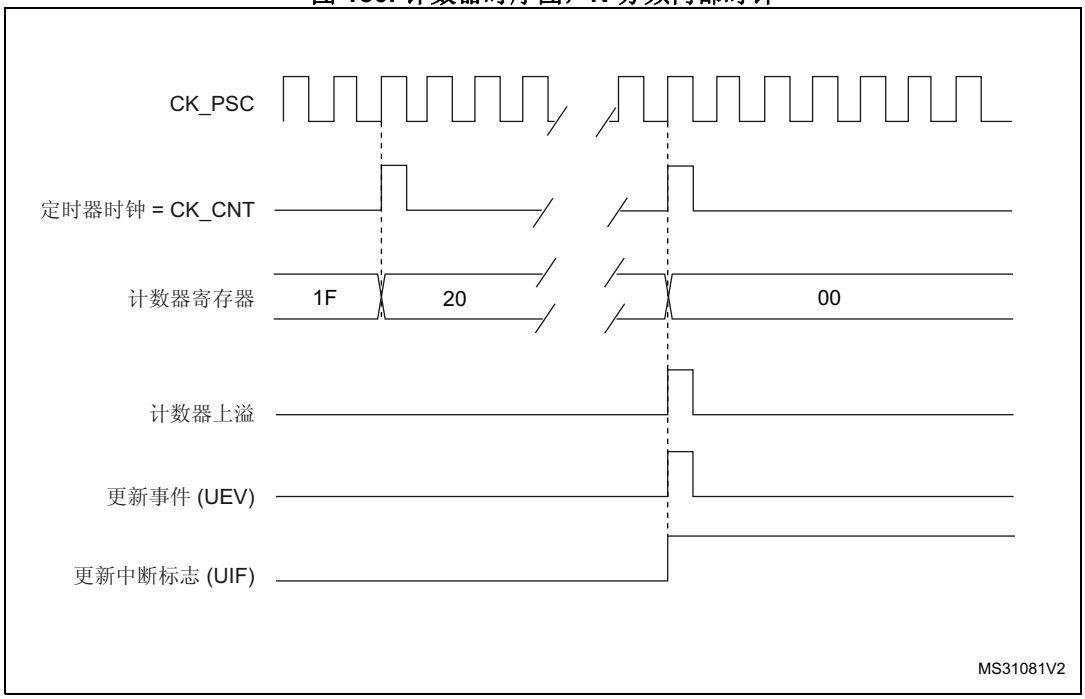


图 151. 计数器时序图，ARPE=0 时更新事件（TIMx_ARR 未预装载）

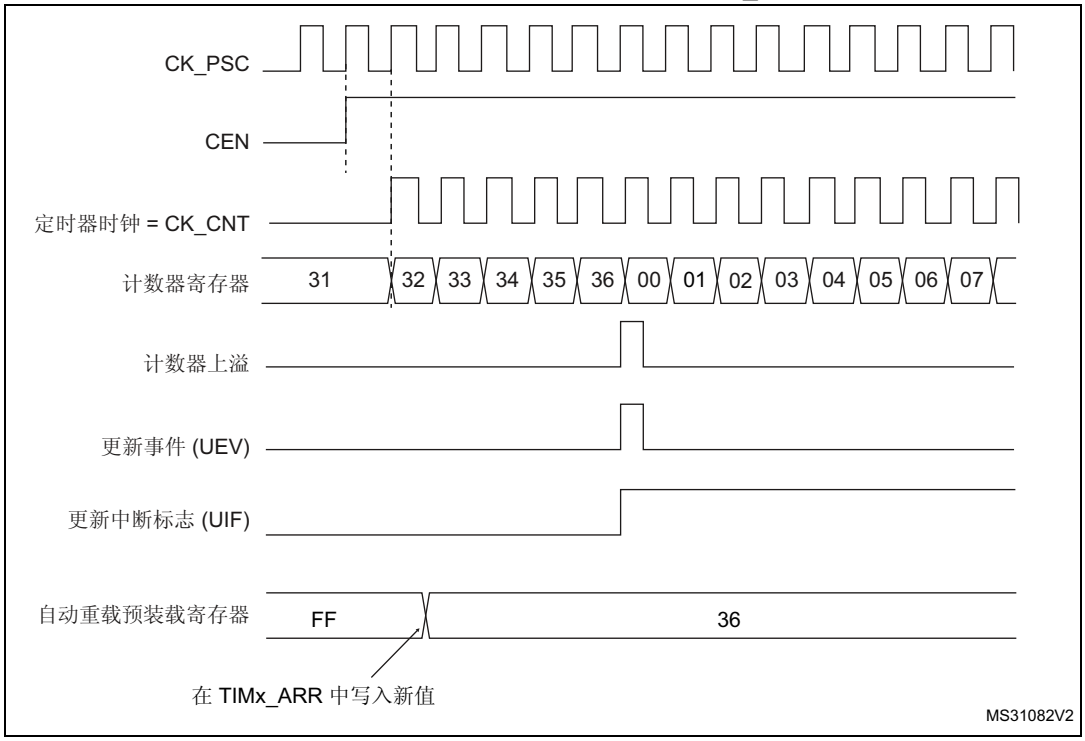
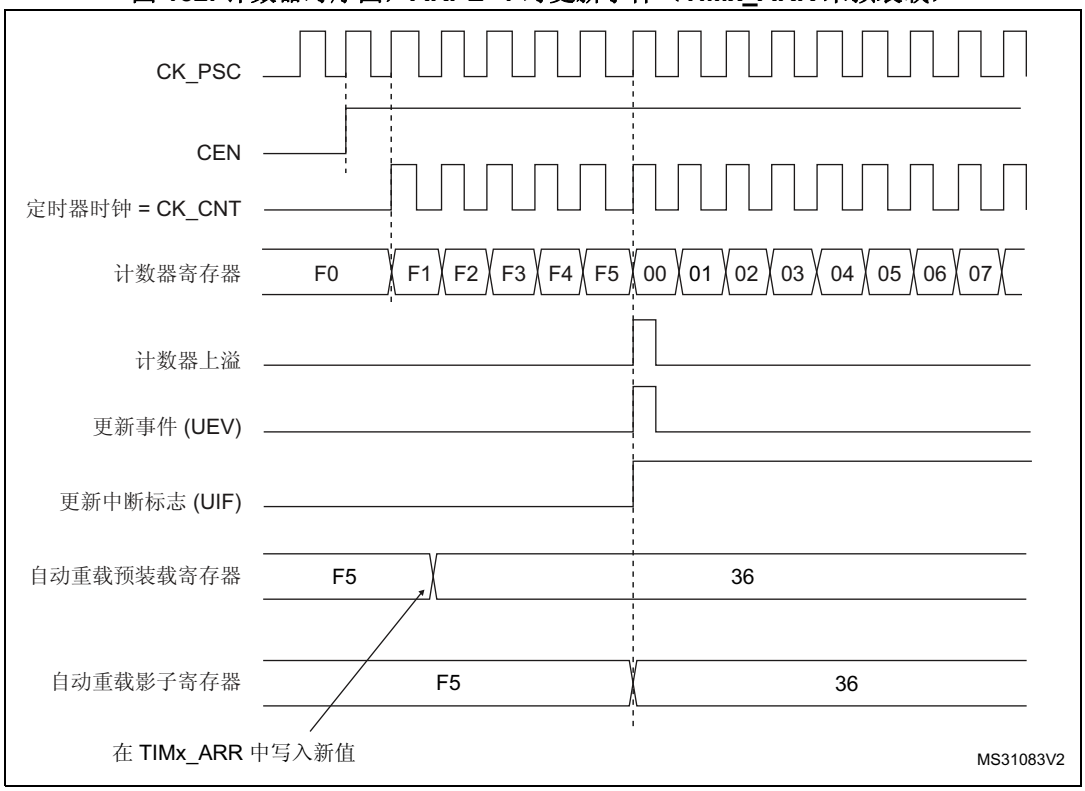


图 152. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载）



递减计数模式

在递减计数模式下，计数器从自动重载值（TIMx_ARR 寄存器的内容）开始递减计数到 0，然后重新从自动重载值开始计数并生成计数器下溢事件。

每次发生计数器下溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器会重新从当前自动重载值开始计数，而预分频器计数器则重新从 0 开始计数（但预分频比保持不变）。

此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，ARR 寄存器更新在计数器重载之前被更新，因此下一个周期就是预期的值。

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 153. 计数器时序图，1 分频内部时钟

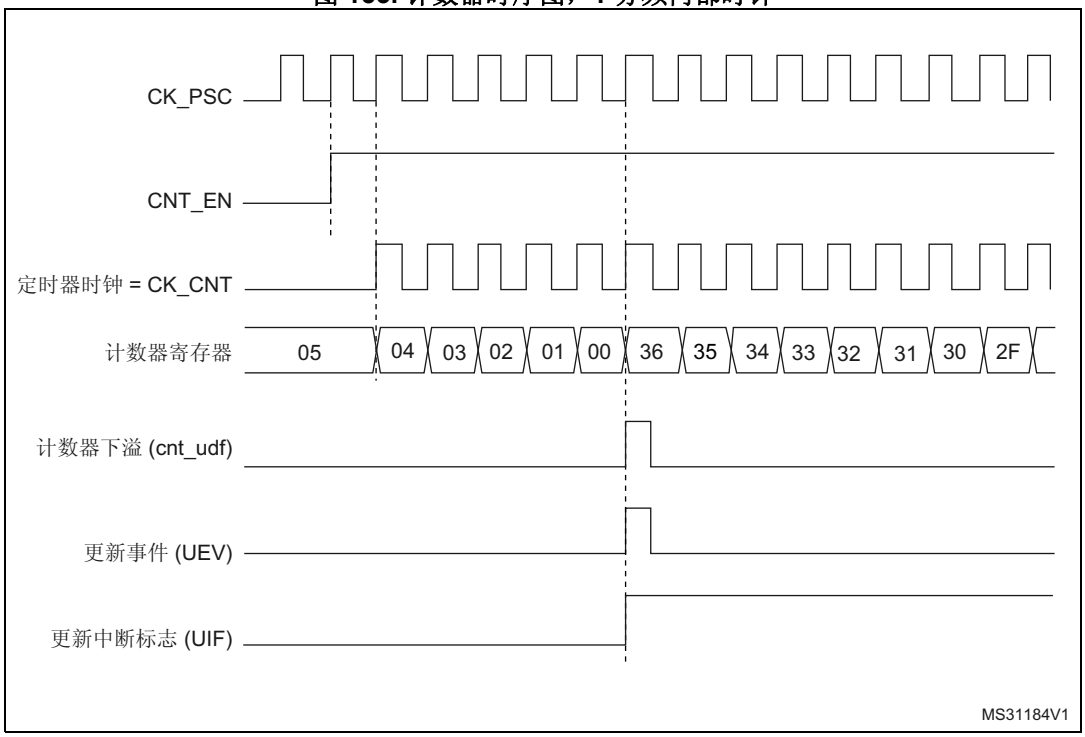


图 154. 计数器时序图，2 分频内部时钟

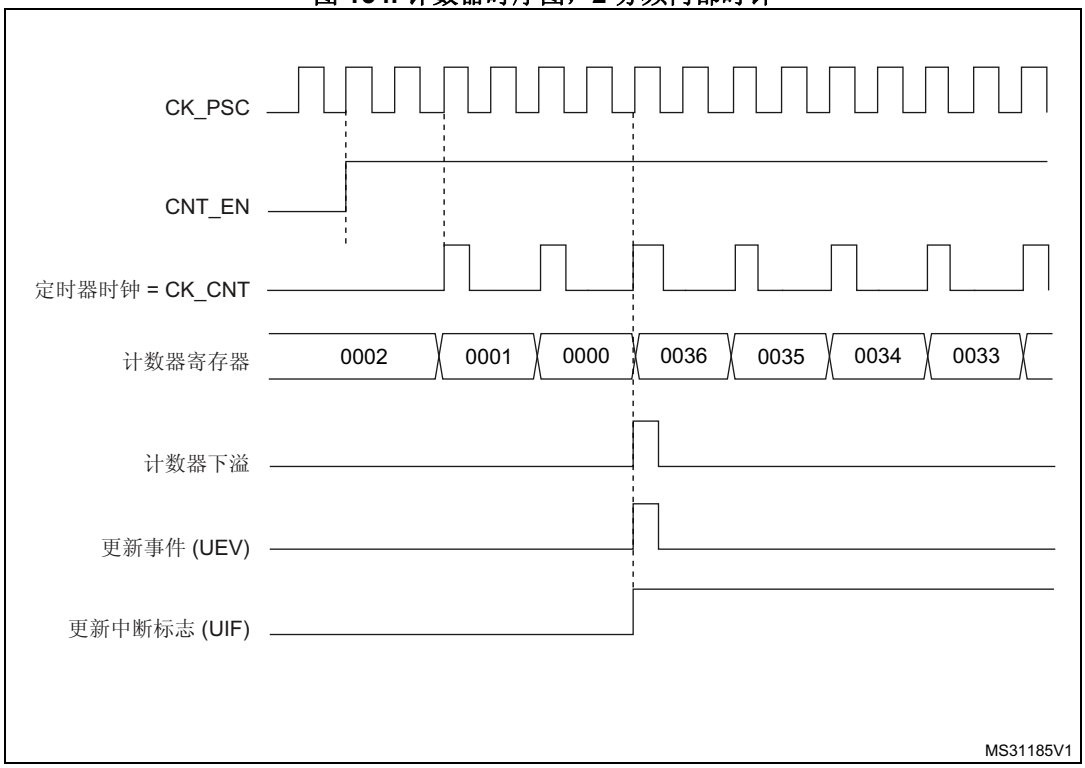


图 155. 计数器时序图，4 分频内部时钟

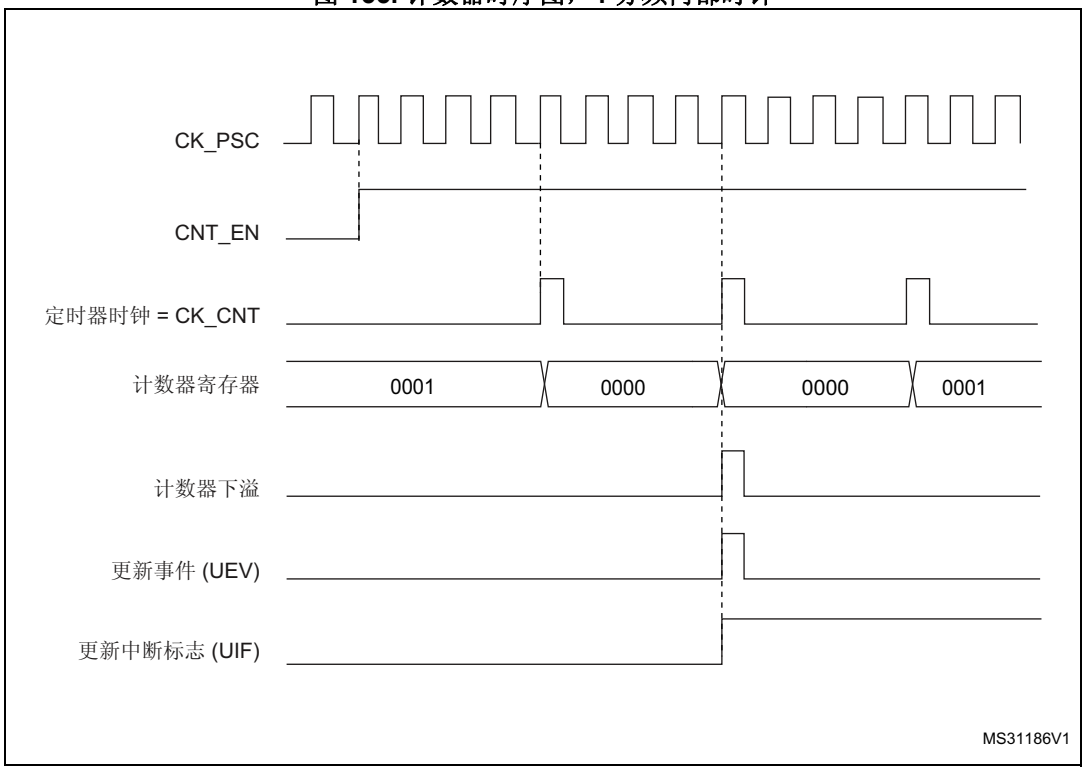
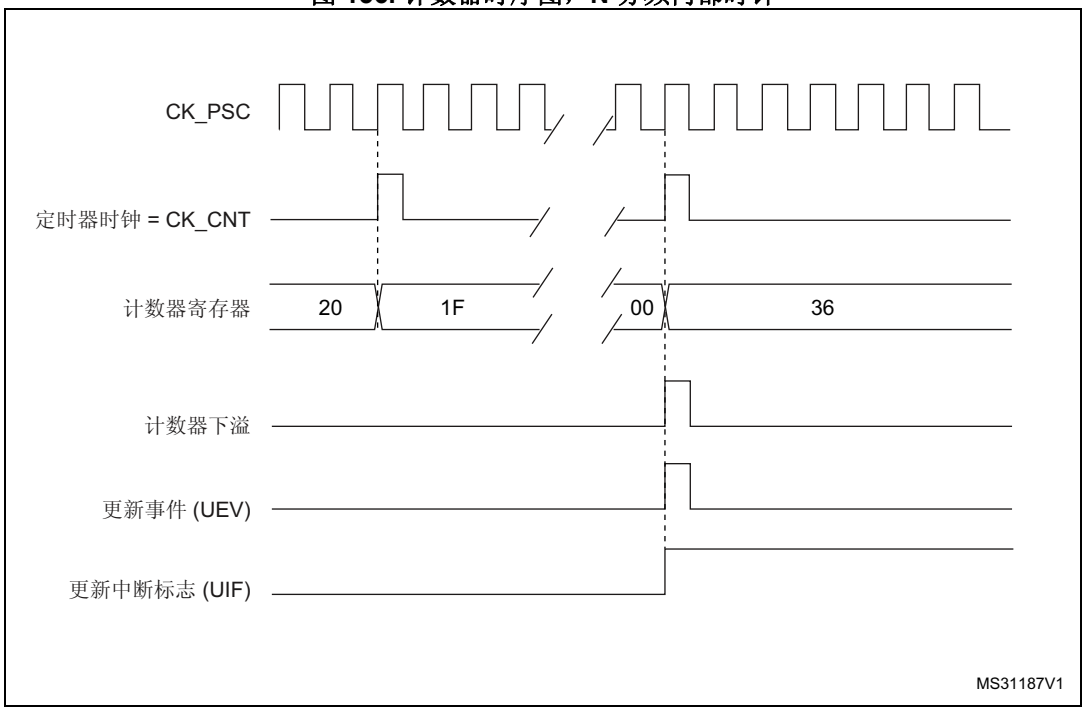


图 156. 计数器时序图，N 分频内部时钟



中心对齐模式（递增/递减计数）

在中心对齐模式下，计数器从 0 开始计数到自动重载值（TIMx_ARR 寄存器的内容）– 1，生成计数器上溢事件；然后从自动重载值开始向下计数到 1 并生成计数器下溢事件。之后从 0 开始重新计数。

当 TIMx_CR1 寄存器中的 CMS 位不为“00”时，中心对齐模式有效。将通道配置为输出模式时，其输出比较中断标志将在以下模式下置 1，即：计数器递减计数（中心对齐模式 1，CMS =“01”）、计数器递增计数（中心对齐模式 2，CMS =“10”）以及计数器递增/递减计数（中心对齐模式 3，CMS =“11”）。

此模式下无法写入方向位（TIMx_CR1 寄存器中的 DIR 位）。而是由硬件更新并指示当前计数器方向。

每次发生计数器上溢和下溢时都会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以生成更新事件。这种情况下，计数器以及预分频器计数器将重新从 0 开始计数。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 更新事件。这可避免向预装载寄存器写入新值时更新影子寄存器。在 UDIS 位写入 0 之前不会产生任何更新事件。不过，计数器仍会根据当前自动重载值进行递增和递减计数。

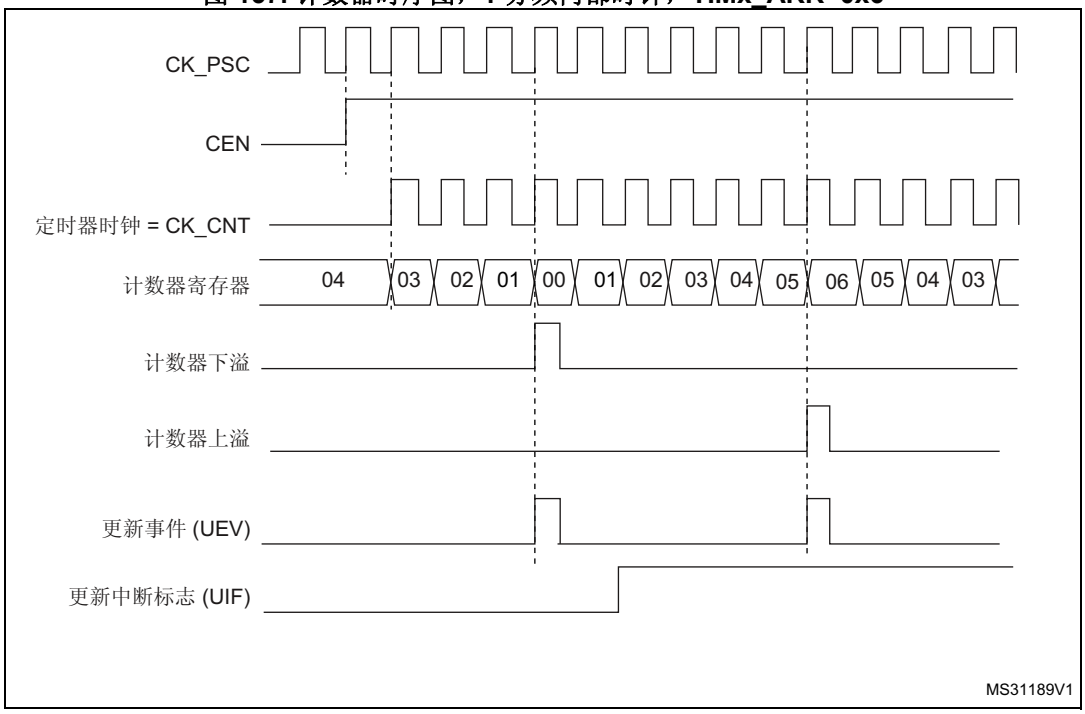
此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。这样一来，如果在发生捕获事件时将计数器清零，将不会同时产生更新中断和捕获中断。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）。
- 自动重载活动寄存器将以预装载值（TIMx_ARR 寄存器的内容）进行更新。注意，如果更新操作是由计数器上溢触发的，则 ARR 寄存器在计数器重载之前更新，因此，下一个计数周期就是我们所希望的新的周期长度（计数器被重载新的值）。

以下各图以一些示例说明不同时钟频率下计数器的行为。

图 157. 计数器时序图，1 分频内部时钟，TIMx_ARR=0x6



1. 此处使用中心对齐模式 1（有关详细信息，请参见*第 529 页的第 22.4.1 节：TIM21/22 控制寄存器 1 (TIMx_CR1)*）。

图 158. 计数器时序图，2 分频内部时钟

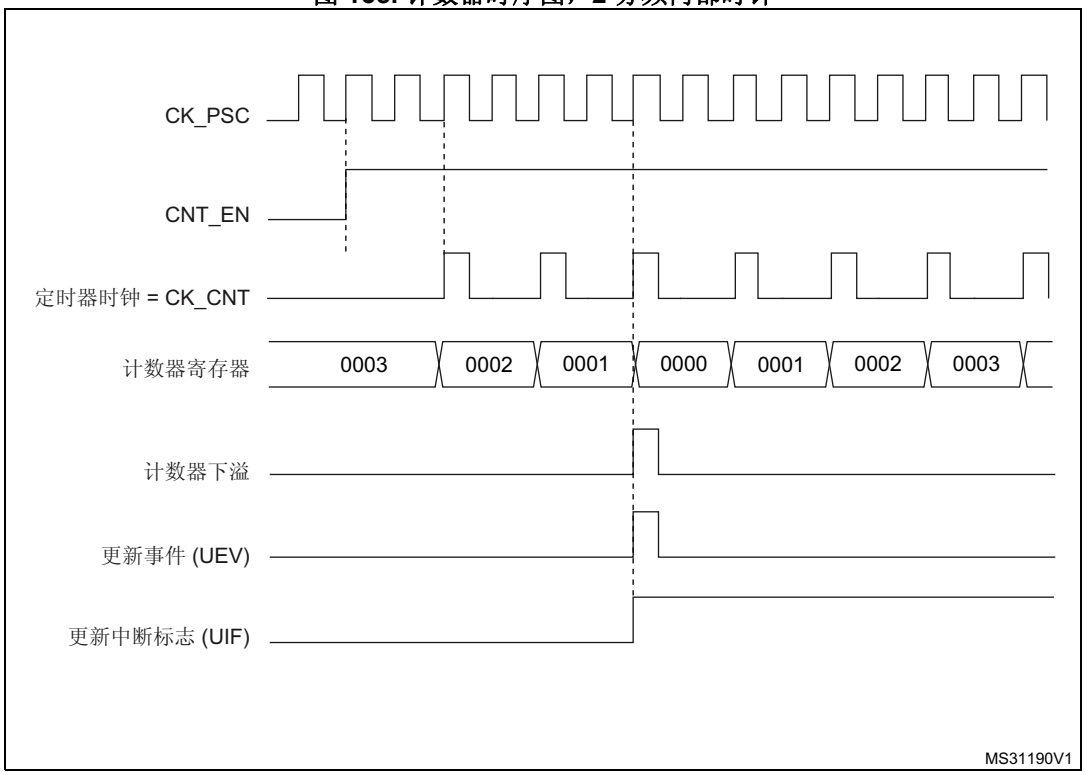
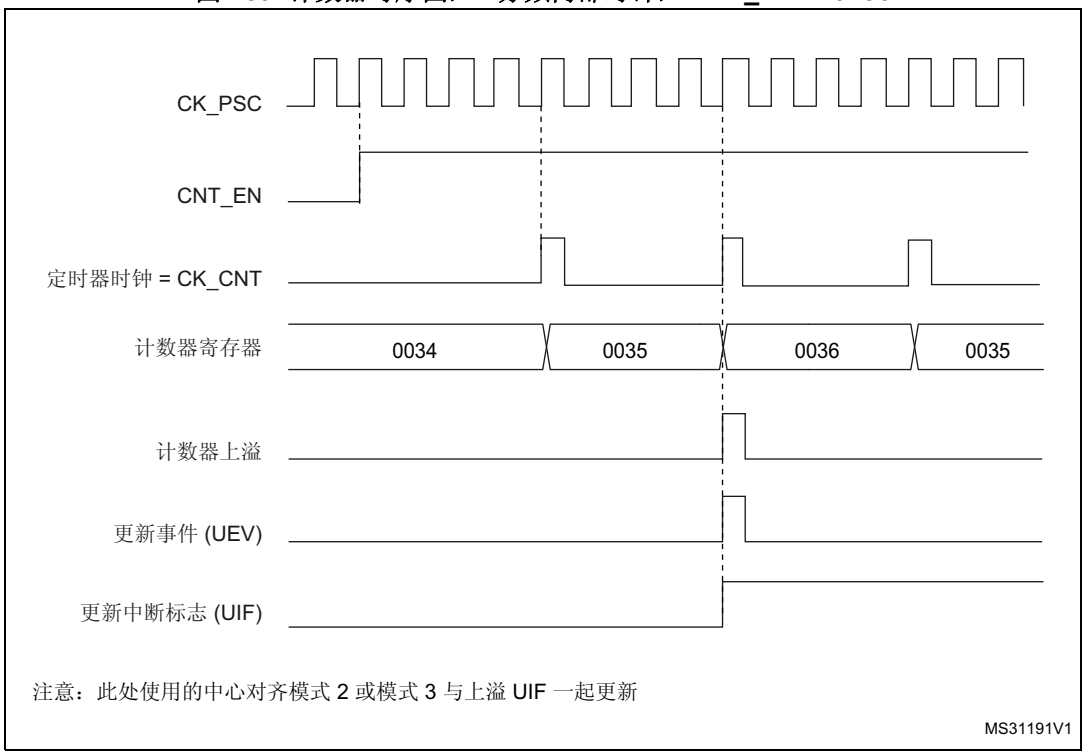


图 159. 计数器时序图，4 分频内部时钟，TIMx_ARR=0x36



1. 中心对齐模式 2 或模式 3 与上溢 UIF 结合使用。

图 160. 计数器时序图，N 分频内部时钟

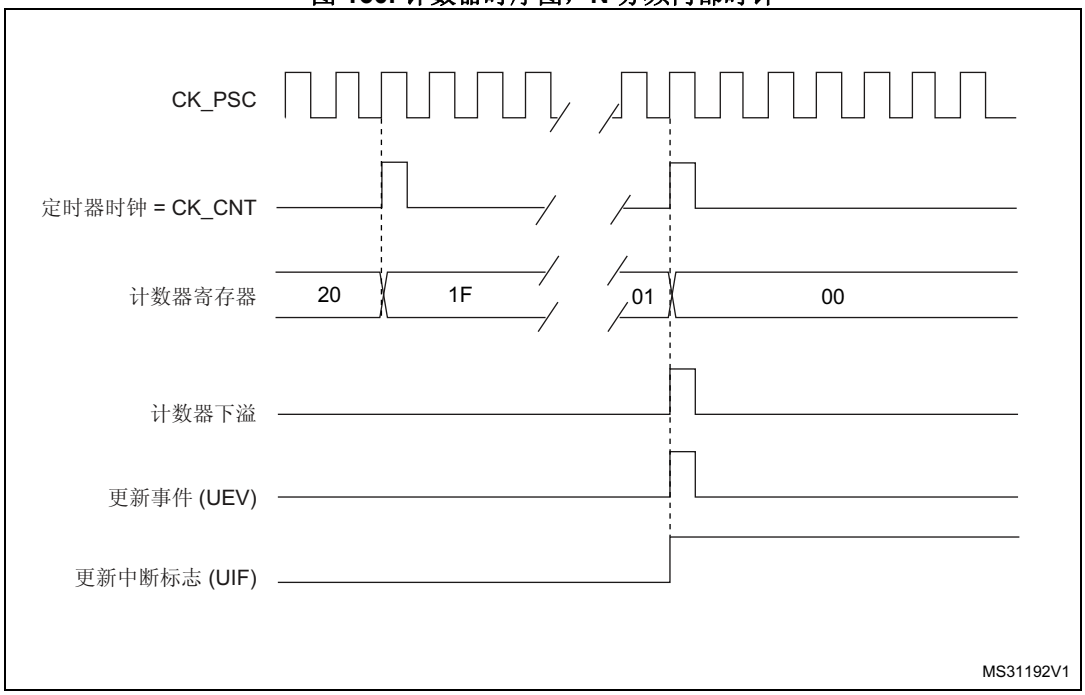


图 161. 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

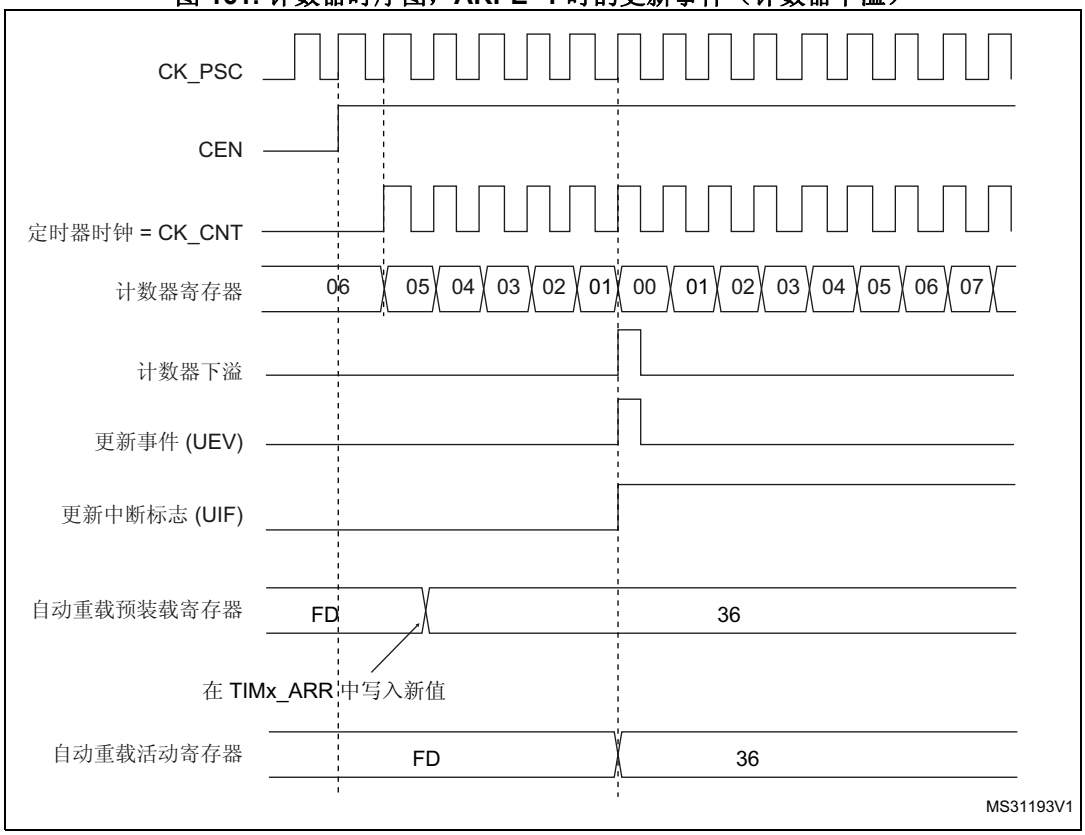
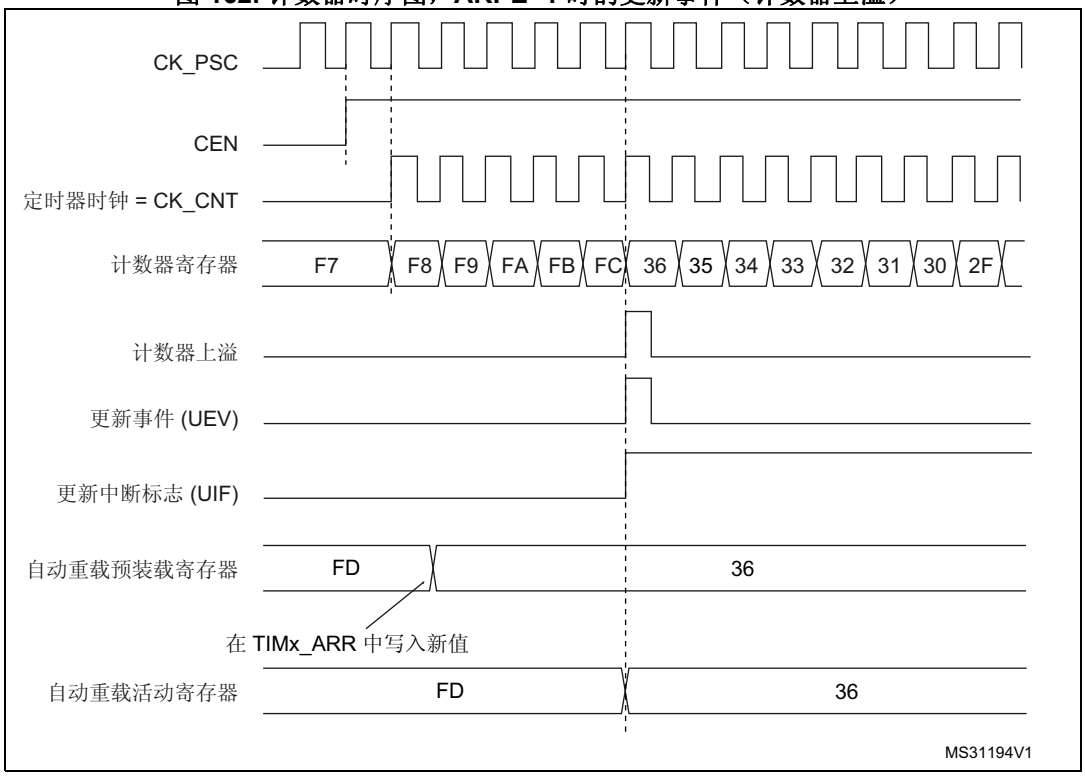


图 162. 计数器时序图，ARPE=1 时的更新事件（计数器上溢）



22.3.3 时钟选择

计数器时钟可由下列时钟源提供：

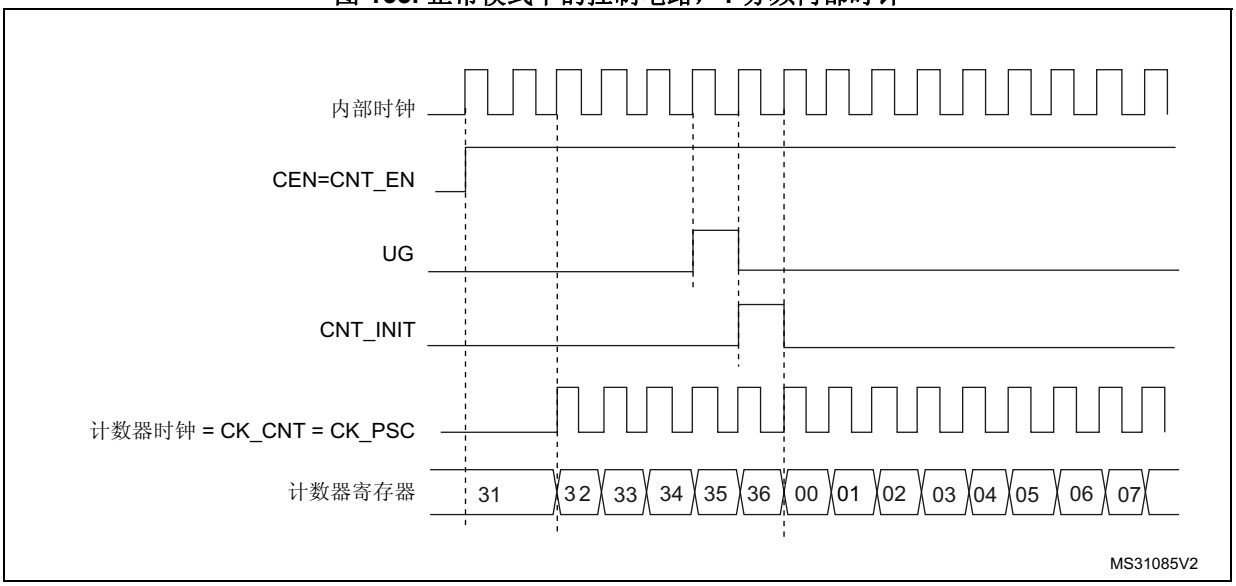
- 内部时钟 (CK_INT)
- 外部时钟模式 1：外部输入引脚 (TIx)
- 外部时钟模式 2：外部触发输入（ETR 内部连接到 LSE）
- 内部触发输入 (ITRx)：连接来自其它计数器的触发输出。更多详细信息，请参见*将一个定时器用作另一个定时器的预分频器*一节。

内部时钟源 (CK_INT)

在禁止从模式控制器后选择内部时钟源（SMS=“000”）。然后，将 TIMx_CR1 寄存器中的 CEN 位和 TIMx_EGR 寄存器中的 UG 位用作控制位，并且只能通过软件对其进行更改（保持清零状态的 UG 除外）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 163 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

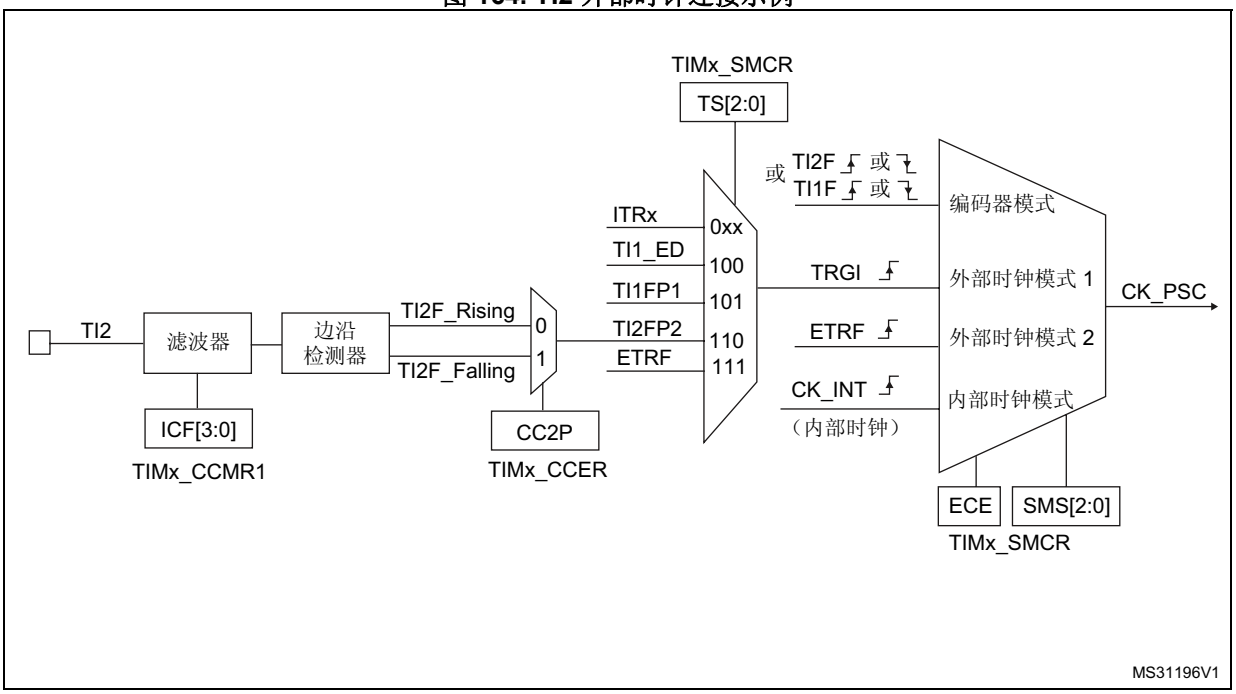
图 163. 正常模式下的控制电路，1 分频内部时钟



外部时钟源模式 1

当 TIMx_SMCR 寄存器中的 SMS=“111”时，可选择此模式。计数器可在选定的输入信号上出现上升沿或下降沿时计数。

图 164. TI2 外部时钟连接示例



例如，要使递增计数器在 TI2 输入出现上升沿时计数，请执行以下步骤：

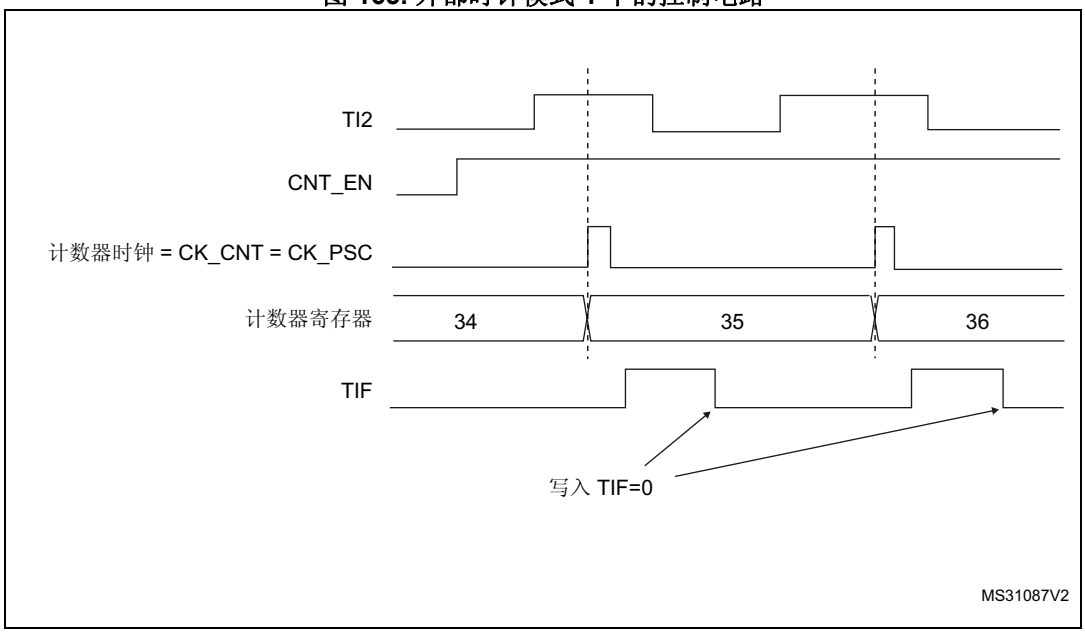
1. 通过在 TIMx_CCMR1 寄存器中写入 CC2S = “01” 来配置通道 2，使其能够检测 TI2 输入的上升沿。
2. 通过在 TIMx_CCMR1 寄存器中写入 IC2F[3:0] 位来配置输入滤波带宽（如果不需要任何滤波器，请保持 IC2F= “0000” ）。
3. 通过在 TIMx_CCER 寄存器中写入 CC2P= “0” 和 CC2NP= “0” 来选择上升沿极性。
4. 通过在 TIMx_SMCR 寄存器中写入 SMS= “111” ，使定时器在外部时钟模式 1 下工作。
5. 通过在 TIMx_SMCR 寄存器中写入 TS= “110” 来选择 TI2 作为触发输入源。
6. 通过在 TIMx_CR1 寄存器中写入 CEN= “1” 来使能计数器。

有关代码示例，请参见 *A.11.1：TI2 上升沿递增计数器代码示例*。

注： *由于捕获预分频器不用于触发操作，因此无需对其进行配置。*

当 TI2 出现上升沿时，计数器便会计数一次并且 TIF 标志置 1。
TI2 的上升沿与实际计数器时钟之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 165. 外部时钟模式 1 下的控制电路



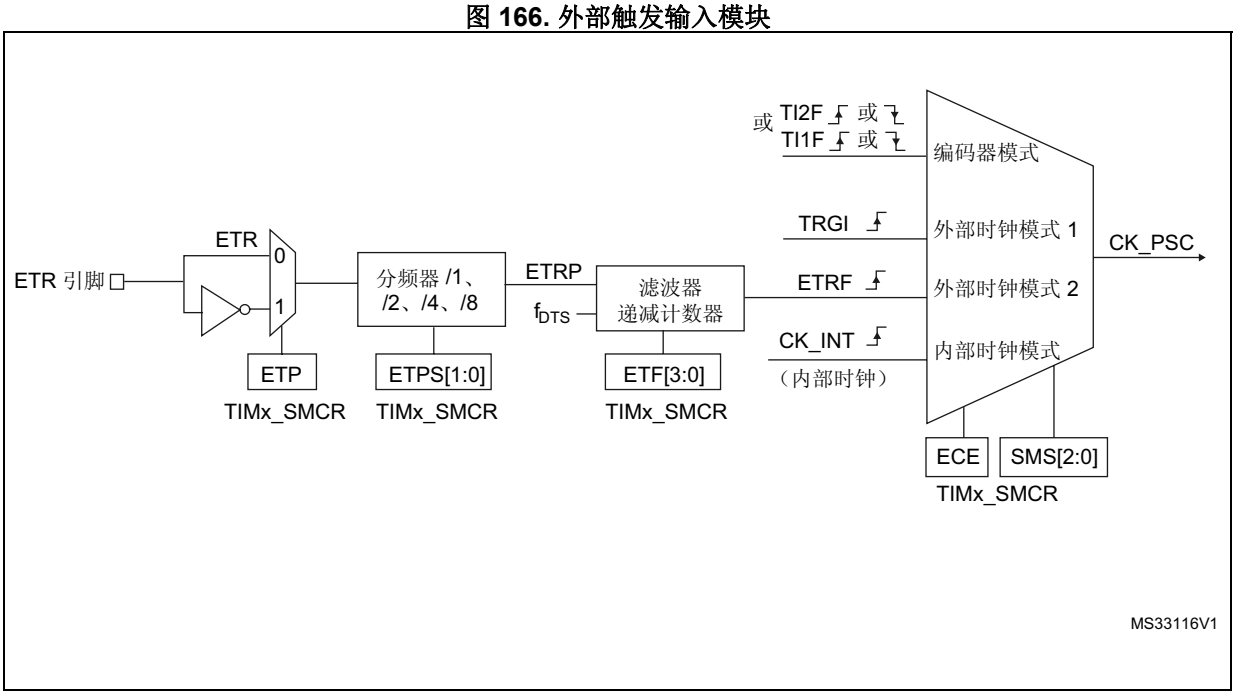
外部时钟源模式 2

通过在 TIMx_SMCR 寄存器中写入 ECE=1 可选择此模式。

计数器可在外部触发输入 ETR 出现上升沿或下降沿时计数。

图 166 简要介绍了外部触发输入模块。

图 166. 外部触发输入模块



例如，要使递增计数器在 ETR 每出现 2 个上升沿时计数，请执行以下步骤：

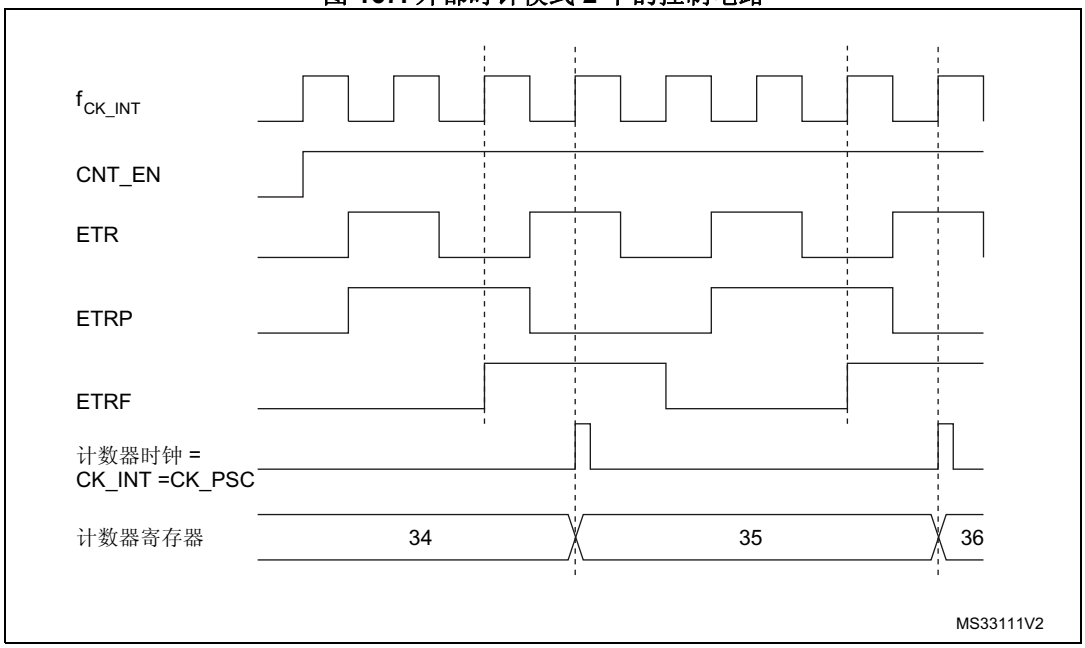
1. 由于此例中不需滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
2. 通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来设置预分频器。
3. 通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测。
4. 通过在 TIMx_SMCR 寄存器中写入 ECE=1 来使能外部时钟模式 2。
5. 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。

有关代码示例，请参见 *A.11.2：每 2 个 ETR 上升沿递增计数器代码示例*。

ETR 每出现 2 个上升沿，计数器计数一次。

ETR 的上升沿与实际计数器时钟之间的延迟是由于 ETRP 信号的重新同步电路引起的。

图 167. 外部时钟模式 2 下的控制电路



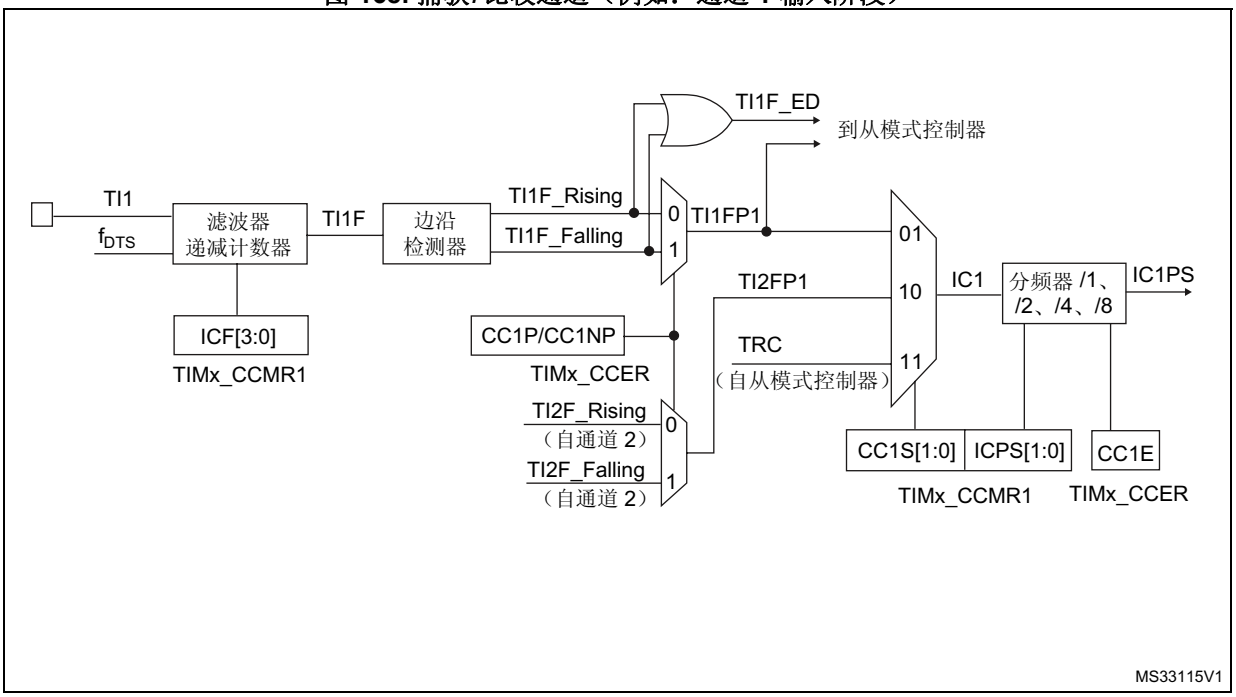
22.3.4 捕获/比较通道

每个捕获/比较通道均围绕一个捕获/比较寄存器（包括一个影子寄存器）、一个捕获输入阶段（数字滤波、多路复用和预分频器）和一个输出阶段（比较器和输出控制）构建而成。

图 168 到*图 170* 概括介绍了一个捕获/比较通道。

输入阶段对相应的 TIx 输入进行采样，生成一个滤波后的信号 TIxF。然后，带有极性选择功能的边沿检测器生成一个信号 (TIxFPx)，该信号可用作从模式控制器的触发输入，也可用作捕获命令。该信号先进行预分频 (ICxPS)，而后再进入捕获寄存器。

图 168. 捕获/比较通道（例如：通道 1 输入阶段）



输出阶段生成一个中间波形作为基准：OCxRef（高电平有效）。链的末端决定最终输出信号的极性。

图 169. 捕获/比较通道 1 主电路

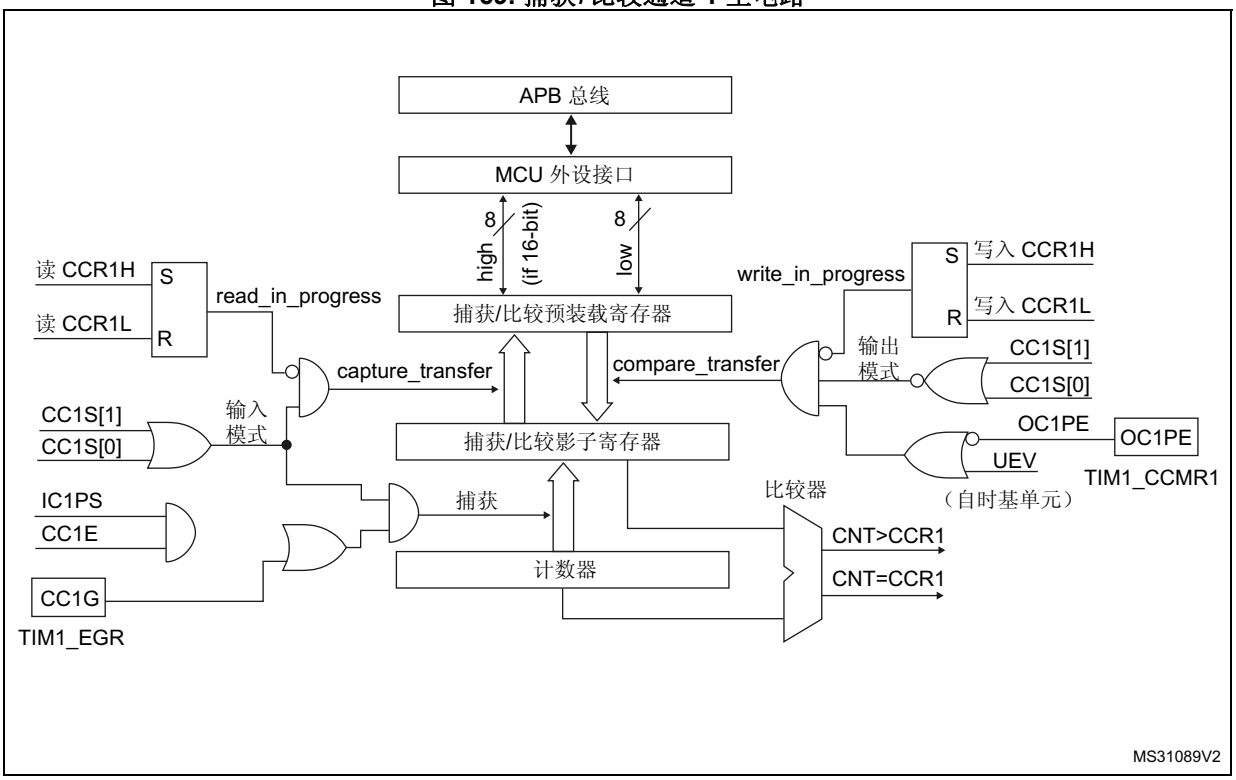
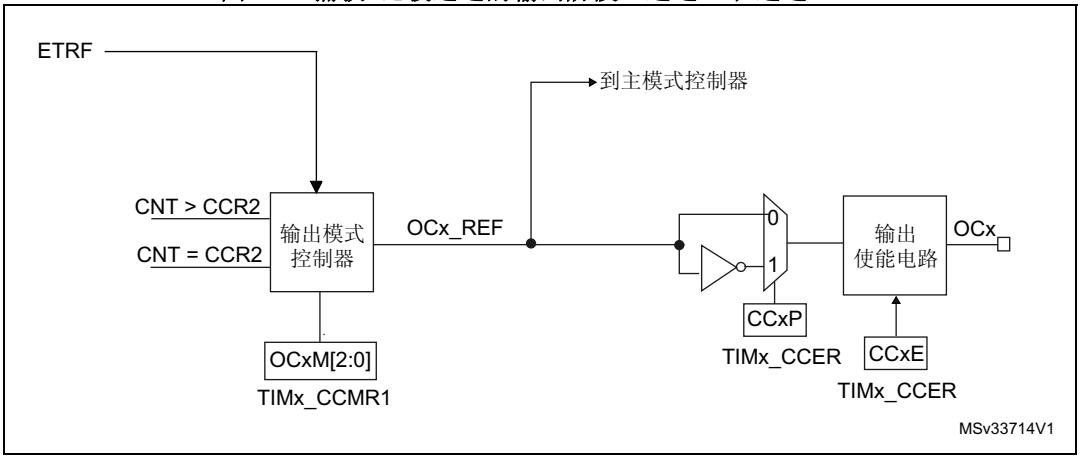


图 170. 捕获/比较通道的输出阶段（通道 1 和通道 2）



捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。始终可通过读写操作访问预装载寄存器。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

22.3.5 输入捕获模式

在输入捕获模式下，当相应的 ICx 信号检测到跳变沿后，将使用捕获/比较寄存器 (TIMx_CCRx) 来锁存计数器的值。发生捕获事件时，会将相应的 CCXIF 标志（TIMx_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。如果发生捕获事件时 CCxIF 标志已处于高位，则会将重复捕获标志 CCxOF（TIMx_SR 寄存器）置 1。可通过软件将 CCxIF 清零，方法是：向 CCxIF 写入“0”，或读取存储在 TIMx_CCRx 寄存器中的已捕获数据。向 CCxOF 写入“0”后会将其清零。

以下示例说明了如何在 TI1 输入出现上升沿时将计数器的值捕获到 TIMx_CCR1 中。具体操作步骤如下：

1. 选择有效输入：TIMx_CCR1 必须连接到 TI1 输入，因此向 TIMx_CCMR1 寄存器中的 CC1S 位写入“01”。只要 CC1S 不等于“00”，就会将通道配置为输入模式，并且 TIMx_CCR1 寄存器将处于只读状态。
2. 根据连接到定时器的信号，对所需的输入滤波带宽进行编程（如果输入为 TIx 之一，则对 TIMx_CCMRx 寄存器中的 ICxF 位进行编程）。假设信号边沿变化时，输入信号最多在 5 个内部时钟周期内发生抖动。因此，我们必须将滤波带宽设置为大于 5 个内部时钟周期。在检测到 8 个具有新电平的连续采样（以 f_{DTS} 频率采样）后，可以确认 TI1 上的跳变沿。然后向 TIMx_CCMR1 寄存器中的 IC1F 位写入“0011”。
3. 通过在 TIMx_CCER 寄存器中将 CC1P 位和 CC1NP 位编程为“00”，选择 TI1 上的有效转换边沿（本例中为上升沿）。
4. 对输入预分频器进行编程。在本例中，我们希望每次有效转换时都执行捕获操作，因此需要禁止预分频器（向 TIMx_CCMR1 寄存器中的 IC1PS 位写入“00”）。
5. 通过将 TIMx_CCER 寄存器中的 CC1E 位置 1，允许将计数器的值捕获到捕获寄存器中。
6. 如果需要，可通过将 TIMx_DIER 寄存器中的 CC1IE 位置 1 来使能相关中断请求。

有关代码示例，请参见 *A.11.3：输入捕捉配置代码示例*。

发生输入捕获时：

- 发生有效跳变沿时，TIMx_CCR1 寄存器会获取计数器的值。
- 将 CC1IF 标志置 1（中断标志）。如果至少发生了两次连续捕获，但 CC1IF 标志未被清零，这样 CC1OF 捕获溢出标志会被置 1。
- 根据 CC1IE 位生成中断。

有关代码示例，请参见 *A.11.4：输入捕捉数据管理代码示例*。

要处理重复捕获，建议在读出捕获溢出标志之前读取数据。这样可避免丢失在读取捕获溢出标志之后与读取数据之前可能出现的重复捕获信息。

注： *通过软件将 TIMx_EGR 寄存器中的相应 CCxG 位置 1 可生成 IC 中断请求。*

22.3.6 PWM 输入模式

此模式是输入捕获模式的一个特例。其实现步骤与输入捕获模式基本相同，仅存在以下不同之处：

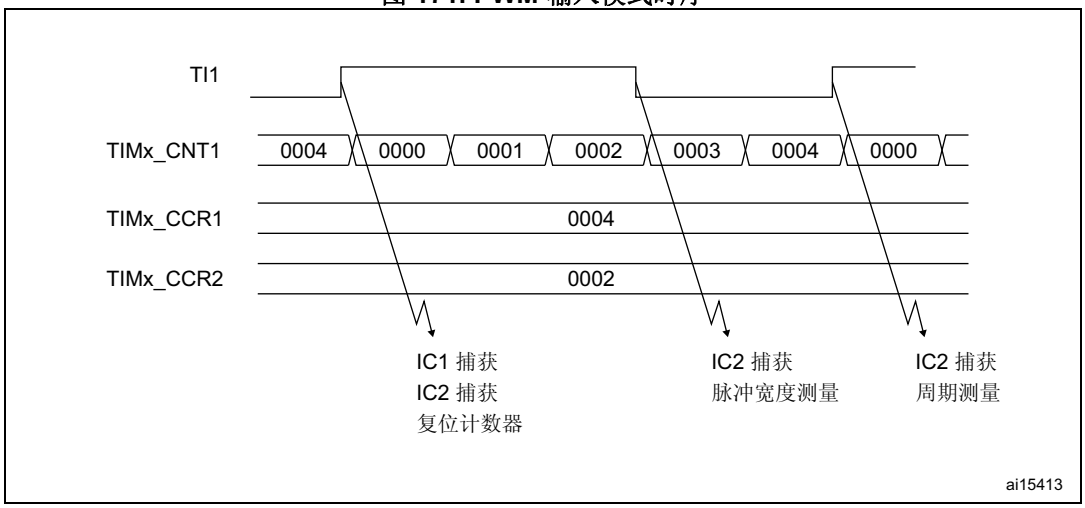
- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这两个 ICx 信号在边沿处有效，但极性相反。
- 选择两个 TIxFP 信号之一作为触发输入，并将从模式控制器配置为复位模式。

例如，可通过以下步骤对应用于 TI1 的 PWM 的周期（位于 TIMx_CCR1 寄存器中）和占空比（位于 TIMx_CCR2 寄存器中）进行测量（取决于 CK_INT 频率和预分频器的值）：

1. 选择 TIMx_CCR1 的有效输入：向 TIMx_CCMR1 寄存器中的 CC1S 位写入“01”（选择 TI1）。
2. 选择 TI1FP1 的有效极性（同时用于 TIMx_CCR1 中的捕获和计数器清零）：将 CC1P 位和 CC1NP 位编程为“00”（上升沿有效）。
3. 选择 TIMx_CCR2 的有效输入：向 TIMx_CCMR1 寄存器中的 CC2S 位写入“10”（选择 TI1）。
4. 选择 TI1FP2 的有效极性（用于 TIMx_CCR2 中的捕获）：将 CC2P 位和 CC2NP 位编程为“11”（下降沿有效）。
5. 选择有效触发输入：向 TIMx_SMCR 寄存器中的 TS 位写入“101”（选择 TI1FP1）。
6. 将从模式控制器配置为复位模式：向 TIMx_SMCR 寄存器中的 SMS 位写入“100”。
7. 使能捕获：向 TIMx_CCER 寄存器中的 CC1E 位和 CC2E 位写入“1”。

有关代码示例，请参见 *A.11.5：PWM 输入配置代码示例*。

图 171. PWM 输入模式时序



1. PWM 输入模式只能与 TIMx_CH1/TIMx_CH2 信号配合使用，因为只有 TI1FP1 和 TI2FP2 与从模式控制器相连。

22.3.7 强制输出模式

在输出模式（TIMx_CCMRx 寄存器中的 CCxS 位 = ‘00’）下，可直接由软件将每个输出比较信号（OCxREF 和 OCx）强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的任何比较结果。

要将输出比较信号 (OCXREF/OCx) 强制设置为有效电平，只需向相应 TIMx_CCMRx 寄存器中的 OCxM 位写入“101”。OCXREF 进而强制设置为高电平（OCxREF 始终为高电平有效），同时 OCx 获取 CCxP 极性位的相反值。

例如：CCxP=“0”（OCx 高电平有效）=> 将 OCx 强制设置为高电平。

通过向 TIMx_CCMRx 寄存器中的 OCxM 位写入“100”，可将 OCxREF 信号强制设置为低电平。

无论如何，TIMx_CCRx 影子寄存器与计数器之间的比较仍会执行，而且允许将标志置 1。因此可相应发送中断请求。下面的输出比较模式一节对此进行了介绍。

22.3.8 输出比较模式

此功能用于控制输出波形，或指示已经过某一时间段。

当捕获/比较寄存器与计数器之间相匹配时，输出比较功能：

1. 将为相应的输出引脚分配一个可编程值，该值由输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx_CCER 寄存器中的 CCxP 位）定义。匹配时，输出引脚既可保持其电平（OCXM = “000”），也可设置为有效电平（OCxM = “001”）、无效电平（OCxM = “010”）或进行翻转（OCxM = “011”）。
2. 将中断状态寄存器中的标志置 1（TIMx_SR 寄存器中的 CCxIF 位）。
3. 如果相应中断使能位（TIMx_DIER 寄存器中的 CCXIE 位）置 1，将生成中断。

使用 TIMx_CCMRx 寄存器中的 OCxPE 位，可将 TIMx_CCRx 寄存器配置为带或不带预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出毫无影响。同步的精度可以达到计数器的一个计数周期。输出比较模式也可用于输出单脉冲（在单脉冲模式下）。

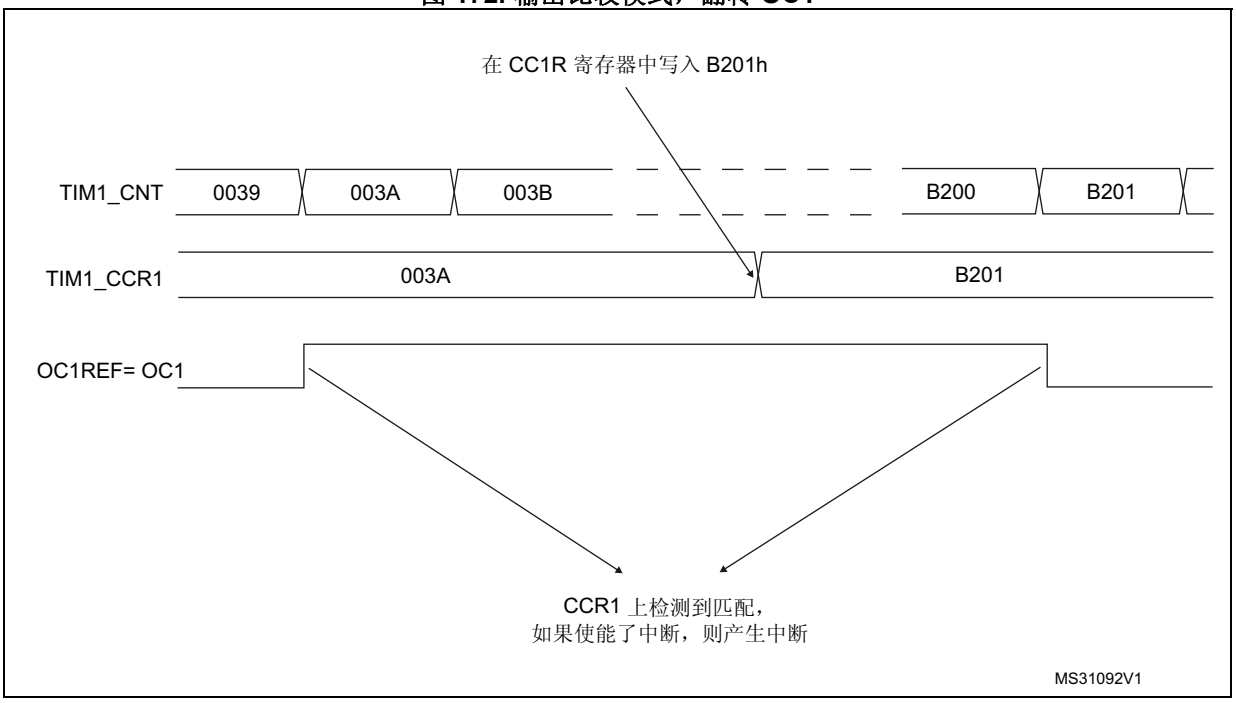
步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 在 TIMx_ARR 和 TIMx_CCRx 寄存器中写入所需数据。
3. 如果要生成中断请求，则需将 CCxIE 位置 1。
4. 选择输出模式。例如：
   - 当 CNT 与 CCRx 匹配时，写入 OCxM = “011” 以翻转 OCx 输出引脚
   - 写入 OCxPE = “0” 以禁止预装载寄存器
   - 写入 CCxP = “0” 以选择高电平有效极性
   - 写入 CCxE = “1” 以使能输出
5. 通过将 TIMx_CR1 寄存器中的 CEN 位置 1 来使能计数器。

有关代码示例，请参见 *A.11.7：输出比较配置代码示例*。

可通过软件随时更新 TIMx_CCRx 寄存器以控制输出波形，前提是未使能预加载寄存器（OCxPE= “0”，否则 TIMx_CCRx 影子寄存器仅在下一更新事件 UEV 发生时进行更新）。*图 172* 给出了一个示例。

图 172. 输出比较模式，翻转 OC1



22.3.9 PWM 模式

脉冲宽度调制模式可以生成一个信号，该信号频率由 TIMx_ARR 寄存器值决定，其占空比则由 TIMx_CCRx 寄存器值决定。

各通道可以独立选择 PWM 模式（每个 OCx 输出对应一个 PWM），只需对 TIMx_CCMRx 寄存器中的 OCxM 位执行写操作。TIMER20 和 TIMER21 仅提供边沿对齐模式。必须通过将 TIMx_CCMRx 寄存器中的 OCxPE 位置 1 使能相应预装载寄存器，最后通过将 TIMx_CR1 寄存器中的 ARPE 位置 1 使能自动重载预装载寄存器（在递增计数或中心对齐模式下）。

由于只有在发生更新事件时预装载寄存器才会传送到影子寄存器，因此启动计数器之前，必须通过将 TIMx_EGR 寄存器中的 UG 位置 1 来初始化所有寄存器。

OCx 极性可通过软件来编程（使用 TIMx_CCER 寄存器的 CCxP 位）。可将其编程为高电平有效或低电平有效。OCx 输出通过将 TIMx_CCER 寄存器中的 CCxE 位置 1 来使能。有关详细信息，请参见 TIMx_CCERx 寄存器说明。

在 PWM 模式（1 或 2）下，TIMx_CNT 始终与 TIMx_CCRx 进行比较，以确定是 TIMx_CNT ≤ TIMx_CCRx。

因为计数器采用递增方式计数，所以定时器能够在边沿对齐模式下生成 PWM。

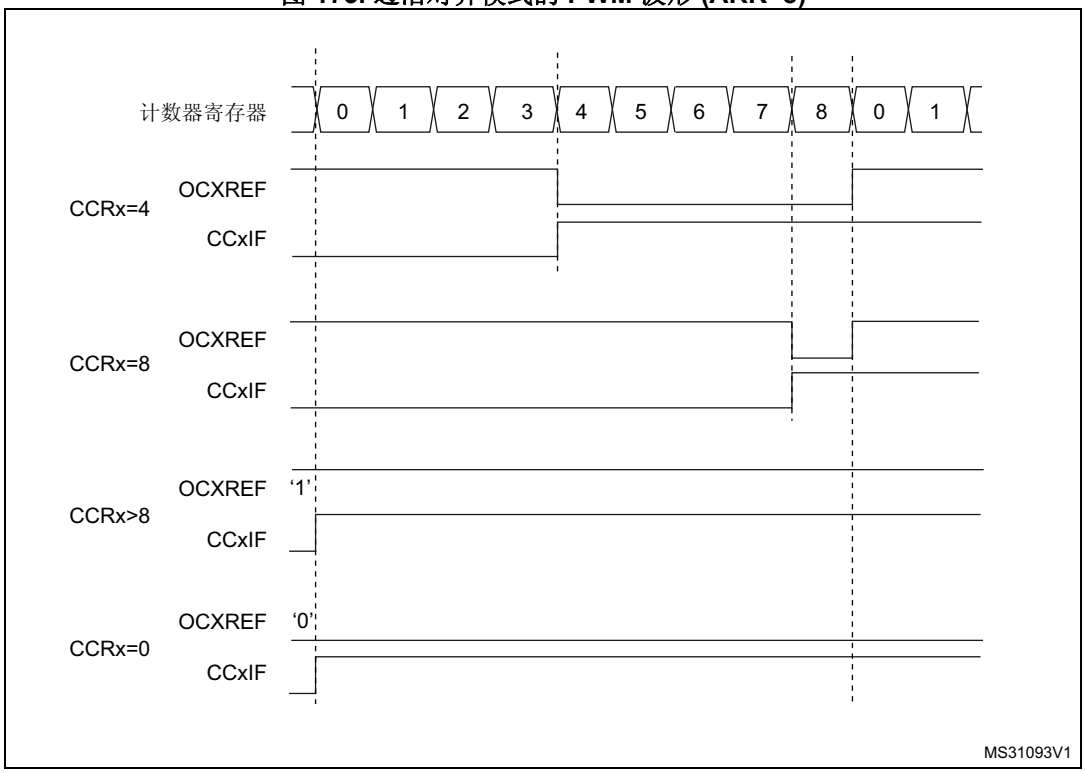
- 递增计数配置

  当 TIMx_CR1 寄存器中的 DIR 位为低时执行递增计数。请参见*第 498 页的递增计数模式*。

  以下以 PWM 模式 1 为例。只要 TIMx_CNT < TIMx_CCRx，PWM 参考信号 OCxREF 便为高电平，否则为低电平。如果 TIMx_CCRx 中的比较值大于自动重载值（TIMx_ARR 中），则 OCxREF 保持为“1”。如果比较值为 0，则 OCxRef 保持为“0”。*图 173* 举例介绍边沿对齐模式的一些 PWM 波形 (TIMx_ARR=8)。

有关代码示例，请参见 *A.11.8：边沿对齐 PWM 配置示例*。

图 173. 边沿对齐模式的 PWM 波形 (ARR=8)



- 递减计数配置

  当 TIMx_CR1 寄存器中的 DIR 位为高时执行递减计数。请参见*第 502 页的递减计数模式*。

  在 PWM 模式 1 下，只要 TIMx_CNT > TIMx_CCRx，参考信号 OCxRef 即为低电平，否则其为高电平。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重载值，则 OCxREF 保持为“1”。此模式下不可能产生 0% 的 PWM 波形。

PWM 中心对齐模式

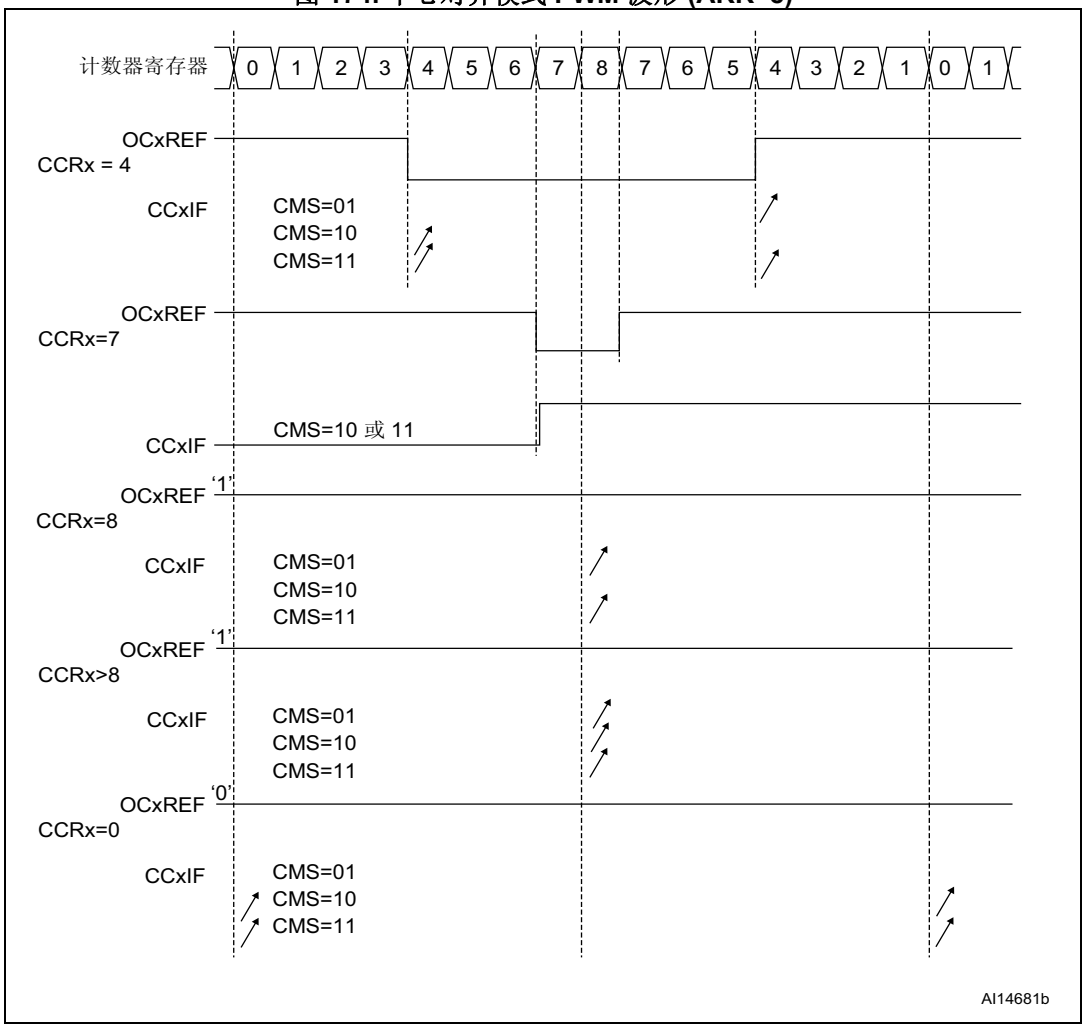
当 TIMx_CR1 寄存器中的 CMS 位不为“00”（其余所有配置对 OCxRef/OCx 信号具有相同的作用），中心对齐模式生效。根据 CMS 位的配置，可以在计数器递增计数、递减计数或同时递增和递减计数时将比较标志置 1。TIMx_CR1 寄存器中的方向位 (DIR) 由硬件更新，不得通过软件更改。请参见*第 504 页的中心对齐模式（递增/递减计数）*。

图 174 显示了中心对齐模式的 PWM 波形，在此例中：

- TIMx_ARR=8，
- PWM 模式为 PWM 模式 1，
- 在根据 TIMx_CR1 寄存器中 CMS=01 而选择的中心对齐模式 1 下，当计数器递减计数时，比较标志置 1。

有关代码示例，请参见 *A.11.9：中心对齐 PWM 配置示例*。

图 174. 中心对齐模式 PWM 波形 (ARR=8)



中心对齐模式使用建议

- 启动中心对齐模式时将使用当前的递增/递减计数配置。这意味着计数器将根据写入 TIMx_CR1 寄存器中 DIR 位的值进行递增或递减计数。此外，不得同时通过软件修改 DIR 和 CMS 位。
- 不建议在运行中心对齐模式时对计数器执行写操作，否则将发生意想不到的结果。尤其是：
  - 如果写入计数器中的值大于自动重载值 (TIMx_CNT>TIMx_ARR)，计数方向不会更新。例如，如果计数器之前递增计数，则继续递增计数。
  - 如果向计数器写入 0 或 TIMx_ARR 的值，计数方向会更新，但不生成更新事件 UEV。
- 使用中心对齐模式最为保险的方法是：在启动计数器前通过软件生成更新（将 TIMx_EGR 寄存器中的 UG 位置 1），并且不要在计数器运行过程中对其执行写操作。

22.3.10 发生外部事件时清除 OCxREF 信号

对于给定通道，在 ETRF 输入施加高电平（相应 TIMx_CCMRx 寄存器中的 OCxCE 使能位置“1”），可使 OCxREF 信号变为低电平。OCxREF 信号将保持低电平，直到发生下一更新事件 (UEV)。

此功能仅能用于输出比较模式和 PWM 模式，而不适用于强制输出模式。

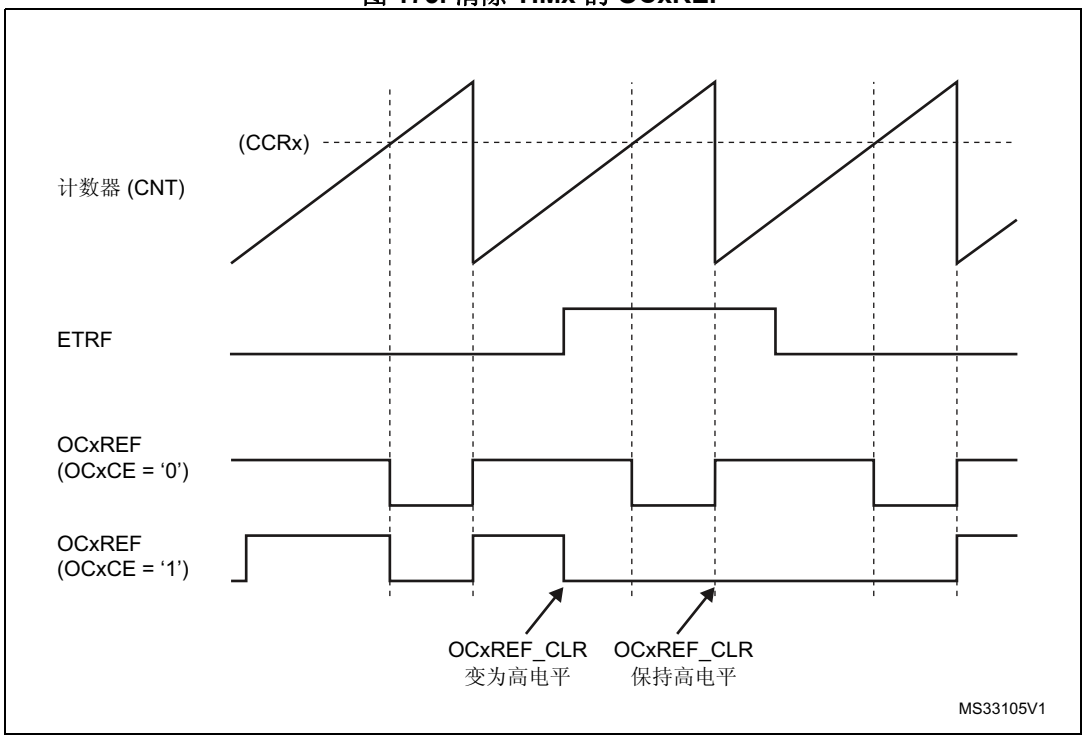
例如，ETR 信号可以连接到比较器的输出，用于控制电流。此时，ETR 必须如下配置：

1. 必须关闭外部触发预分频器：TIMx_SMCR 寄存器中的 ETPS[1:0] 位置“00”。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE 位置“0”。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可根据用户需要进行配置。

有关代码示例，请参见 *A.11.10：清零 OCxREF 的 ETR 配置代码示例*。

图 175 对比了使能位 OCxCE 在不同值下的情况，显示了当 ETRF 输入变为高电平时 OCxREF 信号的行为。在本例中，定时器 TIMx 编程为 PWM 模式。

图 175. 清除 TIMx 的 OCxREF



注： *如果 PWM 的占空比为 100% (CCRx>ARR)，则下次计数器溢出时会再次使能 OCxREF。*

22.3.11 单脉冲模式

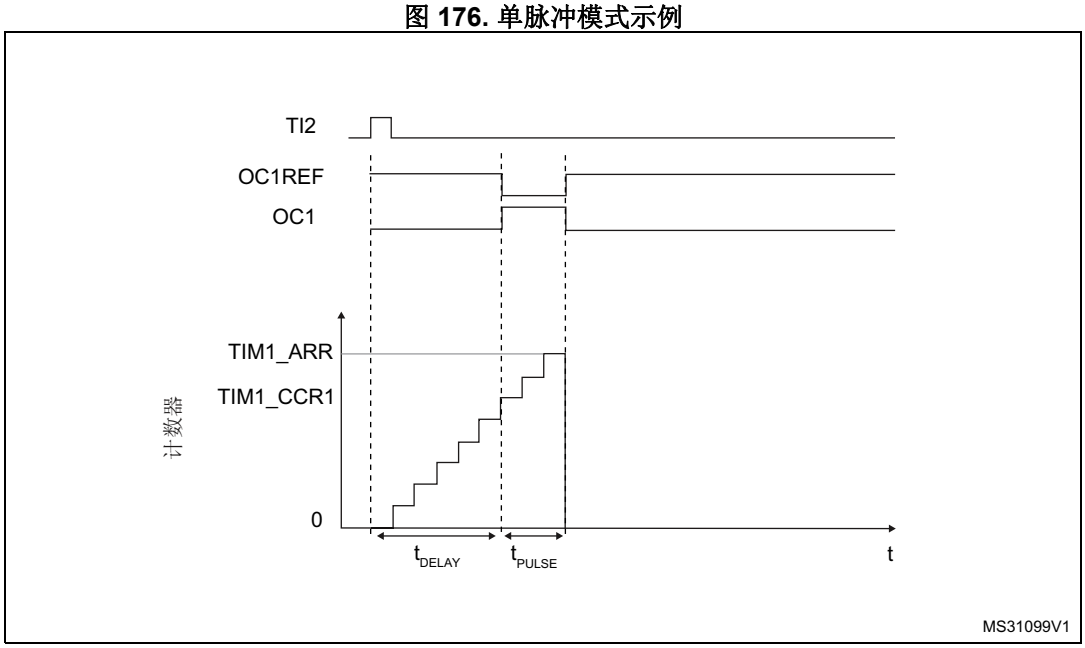
单脉冲模式 (OPM) 是上述模式的一个特例。在这种模式下，计数器可以在一个激励信号的触发下启动，并可在一段可编程的延时后产生一个脉宽可编程的脉冲。

可以通过从模式控制器启动计数器。可以在输出比较模式或 PWM 模式下生成波形。将 TIMx_CR1 寄存器中的 OPM 位置 1，即可选择单脉冲模式。这样，发生下一更新事件 UEV 时，计数器将自动停止。

只有当比较值与计数器初始值不同时，才能正确产生一个脉冲。启动前（定时器等待触发时），必须进行如下配置：

CNT < CCRx ≤ ARR（特别注意，0 < CCRx）

图 176. 单脉冲模式示例



例如，用户希望达到这样的效果：在 TI2 输入引脚检测到上升沿时，经过 t_{DELAY} 的延迟，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

使用 TI2FP2 作为触发 1：

1. 在 TIMx_CCMR1 寄存器中写入 CC2S=“01”，以将 TI2FP2 映射到 TI2。
2. 在 TIMx_CCER 寄存器中写入 CC2P=“0”和 CC2NP=“0”，使 TI2FP2 能够检测上升沿。
3. 在 TIMx_SMCR 寄存器中写入 TS=“110”，以将 TI2FP2 配置为从模式控制器的触发 (TRGI)。
4. 在 TIMx_SMCR 寄存器中写入 SMS=“110”（触发模式），以使用 TI2FP2 启动计数器。

OPM 波形通过对比较寄存器执行写操作来定义（考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由写入 TIMx_CCR1 寄存器的值定义。
- t_{PULSE} 由自动重载值与比较值之差 (TIMx_ARR - TIMx_CCR1+1) 来定义。
- 假设希望产生这样的波形：信号在发生比较匹配时从“0”变为“1”，在计数器达到自动重载值时由“1”变为“0”。为此，应在 TIMx_CCMR1 寄存器中写入 OC1M=“111”，以使能 PWM 模式 2。如果需要，可选择在 TIMx_CCMR1 寄存器的 OC1PE 和 TIMx_CR1 寄存器的 ARPE 中写入“1”，以使能预装载寄存器。这种情况下，必须在 TIMx_CCR1 寄存器中写入比较值并在 TIMx_ARR 寄存器中写入自动重载值，通过将 UG 位置 1 来产生更新，然后等待 TI2 上的外部触发事件。本例中，CC1P 的值为“0”。

在本例中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应为低。

有关代码示例，请参见 *A.11.16：单脉冲模式代码示例*。

由于仅需要 1 个脉冲（单脉冲模式），因此应向 TIMx_CR1 寄存器的 OPM 位写入“1”，以便在发生下一更新事件（计数器从自动重载值返回到 0）时使计数器停止计数。TIMx_CR1 寄存器中的 OPM 位置“0”时，即选择重复模式。

特殊情况：OCx 快速使能

在单脉冲模式下，TIx 输入的边沿检测会将 CEN 位置 1，表示使能计数器。然后，在计数器值与比较值之间发生比较时，将切换输出。但是，完成这些操作需要多个时钟周期，这会限制可能的最小延迟（t_{DELAY} 最小值）。

如果要输出延迟时间最短的波形，可以将 TIMx_CCMRx 寄存器中的 OCxFE 位置 1。这样会强制 OCxRef（和 OCx）对激励信号做出响应，而不再考虑比较的结果。其新电平与发生比较匹配时相同。仅当通道配置为 PWM1 或 PWM2 模式时，OCxFE 才会起作用。

有关代码示例，请参见 *A.11.6：采用 DMA 的 PWM 输入配置代码示例*。

22.3.12 编码器接口模式

选择编码器接口模式时，如果计数器仅在 TI2 边沿处计数，在 TIMx_SMCR 寄存器中写入 SMS=001；如果计数器仅在 TI1 边沿处计数，写入 SMS=010；如果计数器在 TI1 和 TI2 边沿处均计数，则写入 SMS=011。

通过编程 TIMx_CCER 寄存器的 CC1P 和 CC2P 位，选择 TI1 和 TI2 极性。CC1NP 和 CC2NP 必须保持清零。如果需要，还可对输入滤波器进行编程。CC1NP 和 CC2NP 必须保持低电平。

TI1 和 TI2 两个输入用于连接增量编码器。请参见*表 91*。如果使能计数器（在 TIMx_CR1 寄存器的 CEN 位中写入“1”），则计数器的时钟由 TI1FP1 或 TI2FP2 上的每次有效信号转换提供。TI1FP1 和 TI2FP2 是进行输入滤波器和极性选择后 TI1 和 TI2 的信号，如果不进行滤波和反相，则 TI1FP1=TI1，TI2FP2=TI2。将根据两个输入的信号转换序列，产生计数脉冲和方向信号。根据该信号转换序列，计数器相应递增或递减计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应修改。任何输入（TI1 或 TI2）发生信号转换时，都会计算 DIR 位，无论计数器是仅在 TI1 或 TI2 边沿处计数，还是同时在 TI1 和 TI2 处计数。

编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 TIMx_ARR 寄存器中的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 TIMx_ARR。同样，捕获、比较、预分频器、触发输出功能继续正常工作。

在此模式下，计数器会根据正交编码器的速度和方向自动进行修改，因此，其内容始终表示编码器的位置。计数方向对应于所连传感器的旋转方向。下表汇总了可能的组合（假设 TI1 和 TI2 不同时切换）。

表 91. 计数方向与编码器信号的关系

有效边沿	相反信号的电平（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 处计数	高	递减	递增	不计数	不计数
	低	递增	递减	不计数	不计数
仅在 TI2 处计数	高	不计数	不计数	递增	递减
	低	不计数	不计数	递减	递增
在 TI1 和 TI2 处均计数	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

外部增量编码器可直接与 MCU 相连，无需外部接口逻辑。不过，通常使用比较器将编码器的差分输出转换为数字信号。这样大幅提高了抗噪声性能。用于指示机械零位的第三个编码器输出可与外部中断输入相连，用以触发计数器复位。

图 177 以计数器工作为例，说明了计数信号的生成和方向控制。同时也说明了选择双边沿时如何对输入抖动进行补偿。将传感器靠近其中一个切换点放置时可能出现这种情况。本例中假设配置如下：

- CC1S = 01（TIMx_CCMR1 寄存器，TI1FP1 映射到 TI1 上）
- CC2S = 01（TIMx_CCMR2 寄存器，TI2FP2 映射到 TI2 上）
- CC1P 和 CC1NP = “0”（TIMx_CCER 寄存器，TI1FP1 未反相，TI1FP1=TI1）
- CC2P 和 CC2NP = “0”（TIMx_CCER 寄存器，TI2FP2 未反相，TI2FP2= TI2）
- SMS = 011（TIMx_SMCR 寄存器，两个输入在上升沿和下降沿均有效）
- CEN = 1（TIMx_CR1 寄存器，使能计数器）

有关代码示例，请参见 *A.11.11：编码器接口代码示例*。

图 177. 编码器接口模式下的计数器工作示例

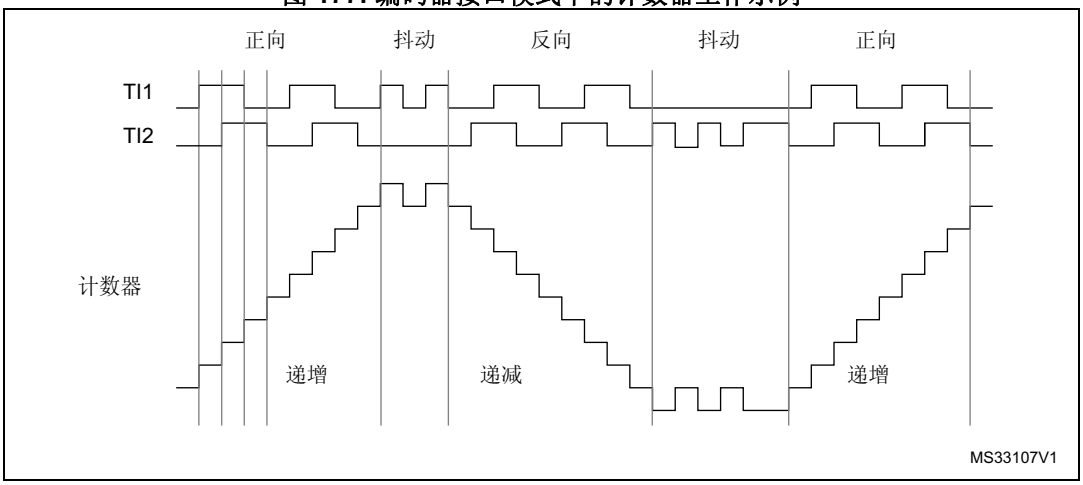
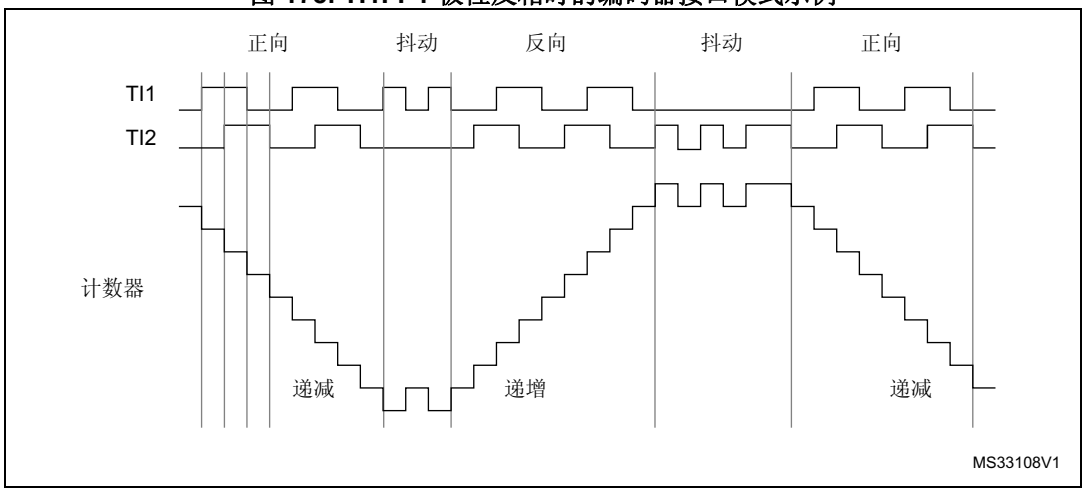


图 178 说明了 TI1FP1 极性反相时的计数器行为示例（除 CC1P=1 外，其它配置与上例相同）。

图 178. TI1FP1 极性反相时的编码器接口模式示例



定时器配置为编码器接口模式时，会提供传感器当前位置的相关信息。使用另一个配置为捕获模式的定时器测量两个编码器事件之间的周期，可获得动态信息（速度、加速度和减速度）。指示机械零位的编码器输出即可用于此目的。根据两个事件之间的时间间隔，还可定期读取计数器。如果可能，可以将计数器值锁存到第三个输入捕获寄存器来实现此目的（捕获信号必须为周期性信号，可以由另一个定时器产生）； 此外，还可以通过实时时钟产生的 DMA 请求读取计数器值。

22.3.13 TIM21/22 外部触发同步

TIM21/22 定时器可与外部触发以下列模式实现同步：复位模式、门控模式和触发模式。

从模式：复位模式

当触发输入信号发生变化时，计数器及其预分频器可重新初始化。此外，如果 TIMx_CR1 寄存器中的 URS 位处于低电平，则会生成更新事件 UEV。然后，所有预装载寄存器（TIMx_ARR 和 TIMx_CCRx）都将更新。

在以下示例中，TI1 输入上出现上升沿时，递增计数器清零：

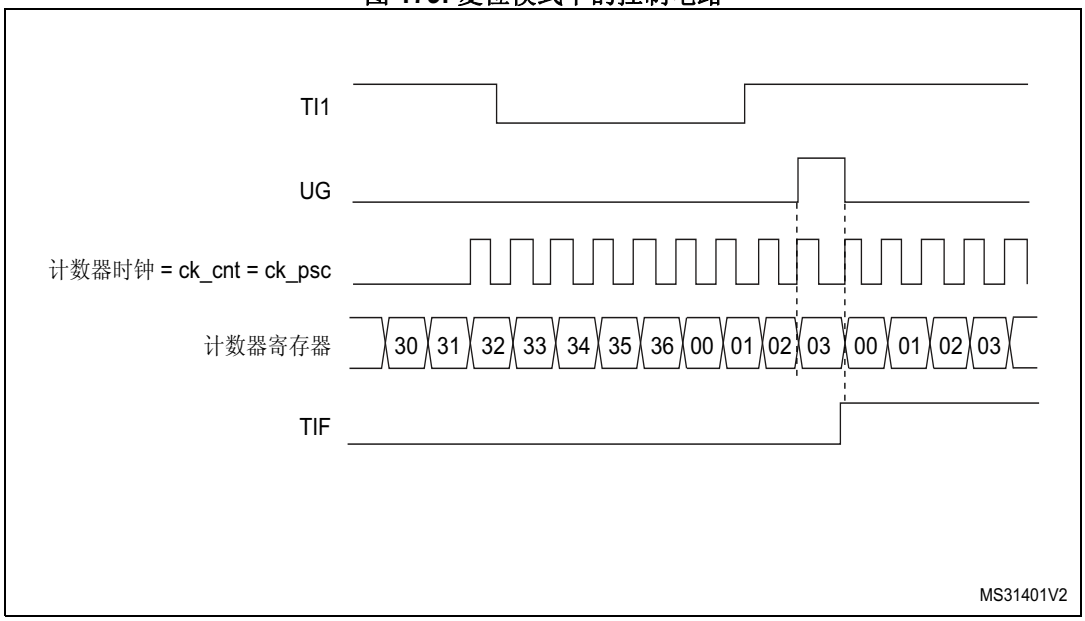
1. 将通道 1 配置为检测 TI1 的上升沿。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC1F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S =“01”。将 TIMx_CCER 寄存器中的 CC1P 和 CC1NP 编程为“00”，以验证极性（仅检测上升沿）。
2. 在 TIMx_SMCR 寄存器中写入 SMS=“100”，将定时器配置为复位模式。在 TIMx_SMCR 寄存器中写入 TS=“101”，选择 TI1 作为输入源。
3. 通过在 TIMx_CR1 寄存器中写入 CEN=“1”来启动动计数器。

有关代码示例，请参见 *A.11.12：复位模式代码示例*。

计数器开始根据内部时钟计数，然后正常运转，直到出现 TI1 上升沿。当 TI1 出现上升沿时，计数器清零，然后重新从 0 开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）置 1，使能中断，还可发送中断请求（取决于 TIMx_DIER 寄存器中的 TIE 位）。

下图显示了自动重载寄存器 TIMx_ARR=0x36 时的相关行为。TI1 的上升沿与实际计数器复位之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 179. 复位模式下的控制电路



从模式：门控模式

输入信号的电平可用来使能计数器。

在以下示例中，递增计数器仅在 TI1 输入为低电平时计数：

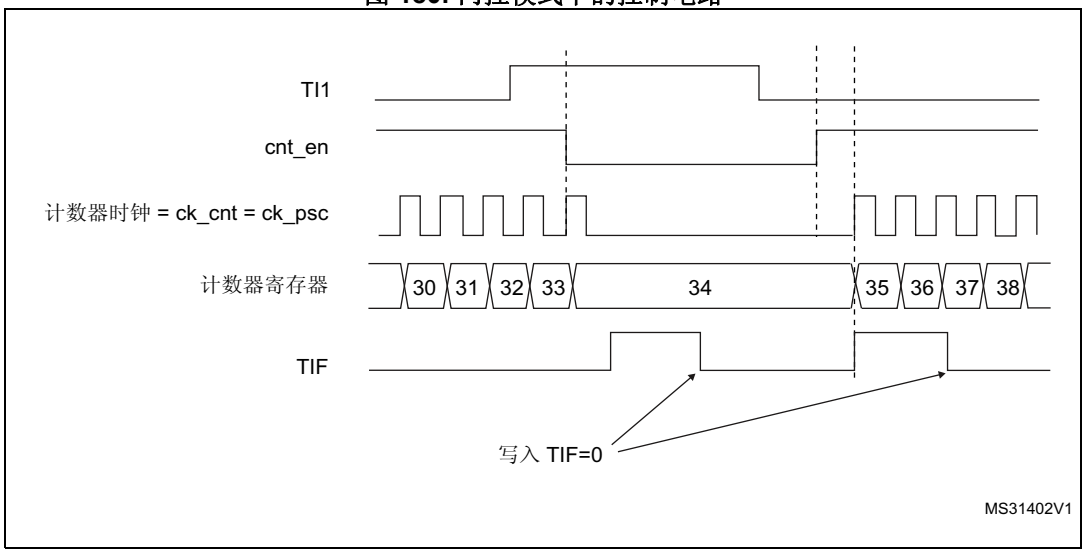
1. 将通道 1 配置为检测 TI1 上的低电平。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC1F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC1S = “01”。将 TIMx_CCER 寄存器中的 CC1P 和 CC1NP 分别编程为“1”和“0”，以确定极性（仅检测低电平）。
2. 在 TIMx_SMCR 寄存器中写入 SMS= “101”，将定时器配置为门控模式。在 TIMx_SMCR 寄存器中写入 TS= “101”，选择 TI1 作为输入源。
3. 在 TIMx_CR1 寄存器中写入 CEN= “1”，使能计数器（在门控模式下，如果 CEN= “0”，则不论触发输入电平如何，计数器都不启动）。

有关代码示例，请参见 *A.11.13：门控模式代码示例*。

只要 TI1 为低电平，计数器就开始根据内部时钟计数，直到 TI1 变为高电平时停止计数。计数器启动或停止时，TIMx_SR 寄存器中的 TIF 标志都会置 1。

TI1 的上升沿与实际计数器停止之间的延迟是由于 TI1 输入的重新同步电路引起的。

图 180. 门控模式下的控制电路



从模式：触发模式

所选输入上发生某一事件时可以启动计数器。

在以下示例中，TI2 输入上出现上升沿时，递增计数器启动：

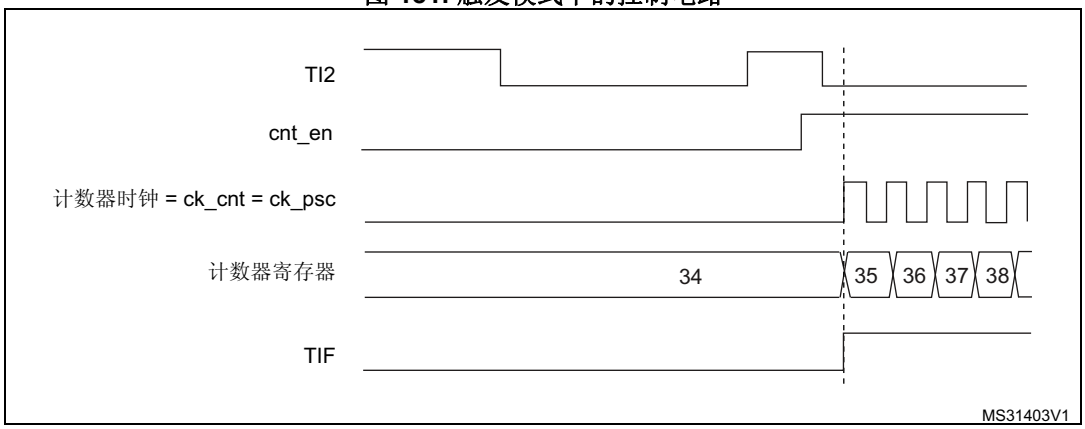
1. 通过对 TIMx_SMCR 寄存器进行如下编程，配置外部触发输入电路：
   - ETF = 0000：无滤波器
   - ETPS = 00：禁止预分频器
   - ETP = 0：检测 ETR 的上升沿，并写入 ECE=1，以使能外部时钟模式 2。
2. 将通道 2 配置为检测 TI2 上的上升沿。配置输入滤波带宽（本例中不需要任何滤波器，因此保持 IC2F=“0000”）。由于捕获预分频器不用于触发操作，因此无需对其进行配置。CC2S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中的 CC2S=“01”。将 TIMx_CCER 寄存器中的 CC2P 和 CC2NP 分别编程为“1”和“0”，以确定极性（仅检测低电平）。
3. 在 TIMx_SMCR 寄存器中写入 SMS=“110”，将定时器配置为触发模式。在 TIMx_SMCR 寄存器中写入 TS=“110”，选择 TI2 作为输入源。

有关代码示例，请参见 *A.11.14：触发模式代码示例*。

当 TI2 出现上升沿时，计数器开始根据内部时钟计数，并且 TIF 标志置 1。

TI2 的上升沿与实际计数器启动之间的延迟是由于 TI2 输入的重新同步电路引起的。

图 181. 触发模式下的控制电路



22.3.14 定时器同步 (TIM21/22)

定时器从内部连接在一起，以实现定时器同步或链接。有关详细信息，请参见*第 464 页的第 21.3.15 节：定时器同步*。

22.3.15 调试模式

当微控制器进入调试模式（Cortex®-M0+ 内核停止）时，TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.9.2 节：对定时器、看门狗和 I^2C 的调试支持*。

22.4 TIM21/22 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器的写访问仅支持半字（16 位）或字（32 位）。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

22.4.1 TIM21/22 控制寄存器 1 (TIMx_CR1)

TIM21/22 control register 1

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
Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:10 保留，必须保持复位值。

位 9:8 **CKD**：时钟分频 (Clock division)

此位域指示定时器时钟 (CK_INT) 频率与数字滤波器所使用的采样时钟 (TIx) 之间的分频比

00：$t_{DTS} = t_{CK_INT}$
01：$t_{DTS} = 2 \times t_{CK_INT}$
10：$t_{DTS} = 4 \times t_{CK_INT}$
11：保留

位 7 **ARPE**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲。
1：TIMx_ARR 寄存器进行缓冲。

位 6:5 **CMS[1:0]**：中心对齐模式选择 (Center-aligned mode selection)

00：边沿对齐模式。计数器根据方向位 (DIR) 递增计数或递减计数。

01：中心对齐模式 1。计数器交替进行递增计数和递减计数。仅当计数器递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

10：中心对齐模式 2。计数器交替进行递增计数和递减计数。仅当计数器递增计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志才置 1。

11：中心对齐模式 3。计数器交替进行递增计数和递减计数。当计数器递增计数或递减计数时，配置为输出的通道（TIMx_CCMRx 寄存器中的 CxS=00）的输出比较中断标志都会置 1。

注： 只要计数器处于使能状态 (CEN=1)，就不得从边沿对齐模式切换为中心对齐模式。

位 4 **DIR**：方向 (Direction)

0：计数器递增计数
1：计数器递减计数

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数
1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。

0：使能后，所有以下事件都会生成更新中断：
- 计数器上溢
- 将 UG 位置 1

1：使能后，只有计数器上溢会生成更新中断。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止更新事件 (UEV) 生成。

0：使能 UEV。UEV 可通过以下事件之一生成：

– 计数器上溢

– 将 UG 位置 1

然后更新影子寄存器的值。

1：禁止 UEV。不会生成 UEV，各影子寄存器的值（ARR、PSC 和 CCRx）保持不变。如果 UG 位置 1，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器

1：使能计数器

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

22.4.2 TIM21/22 控制寄存器 2 (TIMx_CR2)

TIM21/22 control register 2

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

位 15:7 保留，必须保持复位值。

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位可选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：

000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入生成（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。

001：**使能**——计数器使能信号 CNT_EN 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号由 CEN 控制位与门控模式下的触发输入的逻辑或运算组合而成。

当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中 MSM 位的说明）。

010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。

011：**比较脉冲**——一旦发生输入捕获或比较匹配事件，当 CC1IF 标志被置 1 时（即使已为高电平），触发输出都会发送一个正脉冲。(TRGO)

100：**比较**——OC1REF 信号用作触发输出 (TRGO)

101：**比较**——OC2REF 信号用作触发输出 (TRGO)

110：保留

111：保留

位 3:0 保留，必须保持复位值。

22.4.3 TIM21/22 从模式控制寄存器 (TIMx_SMCR)

TIM21/22 slave mode control register

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]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位 15 **ETP**：外部触发极性 (External trigger polarity)

此位可选择将 ETR 还是 $\overline{ETR}$ 用于触发操作

0：ETR 未反相，高电平或上升沿有效。

1：ETR 反相，低电平或下降沿有效。

位 14 **ECE**：外部时钟使能 (External clock enable)

此位可使能外部时钟模式 2。

0：禁止外部时钟模式 2

1：使能外部时钟模式 2。计数器时钟由 ETRF 信号的任意有效边沿提供。

注：将 ECE 位置 1 与选择外部时钟模式 1 并将 TRGI 连接到 ETRF（SMS=111 且 TS=111）具有相同效果。

外部时钟模式 2 可以和以下从模式同时使用：复位模式、门控模式和触发模式。不过此类情况下 TRGI 不得连接 ETRF（TS 位不得为 111）。

如果同时使能外部时钟模式 1 和外部时钟模式 2，则外部时钟输入为 ETRF。

位 13:12 **ETPS[1:0]**：外部触发预分频器 (External trigger prescaler)

外部触发信号 ETRP 频率不得超过 TIMxCLK 频率的 1/4。可通过使能预分频器来降低 ETRP 频率。这种方法在输入快速外部时钟时非常有用。

00：预分频器关闭

01：2 分频 ETRP 频率

10：4 分频 ETRP 频率

11：8 分频 ETRP 频率

位 11:8 **ETF[3:0]**：外部触发滤波器 (External trigger filter)

此位域可定义 ETRP 信号的采样频率和适用于 ETRP 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样

0001：$f_{SAMPLING}=f_{CK_INT}$，N=2

0010：$f_{SAMPLING}=f_{CK_INT}$，N=4

0011：$f_{SAMPLING}=f_{CK_INT}$，N=8

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

位 7 **MSM**：主/从模式 (Master/slave mode)

0：不执行任何操作

1：当前定时器的触发输入事件 (TRGI) 的动作被推迟，以使当前定时器与其从定时器实现完美同步（通过 TRGO）。此设置适用于要在发生单个外部事件时对多个定时器进行同步的情况。

位 6:4 **TS**：触发选择 (Trigger selection)

此位域可选择将要用于同步计数器的触发输入。

000：内部触发 0 (ITR0)

001：内部触发 1 (ITR1)

010：保留

011：保留

100：TI1 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。

有关各定时器的 ITRx 含义的更多详细信息，请参见*第 533 页的表 92：TIMx 内部触发连接*。

注：这些位只能在未使用的情况下（例如，SMS=“000”时）进行更改，以避免转换时出现错误的边沿检测。

位 3 保留，必须保持复位值。

位 2:0 **SMS**：从模式选择 (Slave mode selection)

选择外部信号时，触发信号 (TRGI) 的有效边沿与外部输入上所选的极性相关（请参见输入控制寄存器和控制寄存器说明）。

000：禁止从模式——如果 CEN =“1”，预分频器时钟直接由内部时钟提供

001：编码器模式 1

010：编码器模式 2

011：编码器模式 3

100：复位模式——在出现所选触发输入 (TRGI) 上升沿时，重新初始化计数器并生成一个寄存器更新事件

101：门控模式——触发输入 (TRGI) 为高电平时使能计数器时钟。只要触发输入变为低电平，计数器立即停止计数（但不复位）。计数器的启动和停止都是受控的

110：触发模式——触发信号 TRGI 出现上升沿时启动计数器（但不复位）。只控制计数器的启动

111：外部时钟模式 1

注： 如果将 TI1F_ED 选作触发输入（TS=“100”），则不得使用门控模式。实际上，TI1F 每次转换时，TI1F_ED 都输出 1 个脉冲，而门控模式检查的则是触发信号的电平。

表 92. TIMx 内部触发连接(1)

从 TIM	ITR0 (TS = 000)	ITR1 (TS = 001)
TIM21	TIM2	TIM22
TIM22	TIM21	TIM2

1. 当产品中不存在定时器时，相应的触发 ITRx 不可用。

22.4.4 TIM21/22 中断使能寄存器 (TIMx_DIER)

TIM21/22 Interrupt enable register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

位 15:7 保留，必须保持复位值。

位 6 **TIE**：触发中断使能 (Trigger interrupt enable)

0：禁止触发中断。

1：使能触发中断。

位 5:3 保留，必须保持复位值。

位 2 **CC2IE**：捕获/比较 2 中断使能 (Capture/Compare 2 interrupt enable)
0：禁止 CC2 中断。
1：使能 CC2 中断。

位 1 **CC1IE**：捕获/比较 1 中断使能 (Capture/Compare 1 interrupt enable)
0：禁止 CC1 中断。
1：使能 CC1 中断。

位 0 **UIE**：更新中断使能 (Update interrupt enable)
0：禁止更新中断。
1：使能更新中断。

22.4.5 TIM21/22 状态寄存器 (TIMx_SR)

TIM21/22 status register

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
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

位 15:11 保留，必须保持复位值。

位 10 **CC2OF**：捕获/比较 2 重复捕获标志 (Capture/compare 2 overcapture flag)
请参见 CC1OF 说明

位 9 **CC1OF**：捕获/比较 1 重复捕获标志 (Capture/Compare 1 overcapture flag)
仅当将相应通道配置为输入捕获模式时，此标志位才会由硬件置 1。通过软件写入“0”可将该位清零。
0：未检测到重复捕获。
1：TIMx_CCR1 寄存器中已捕获到计数器值且 CC1IF 标志已置 1。

位 8:7 保留，必须保持复位值。

位 6 **TIF**：触发中断标志 (Trigger interrupt flag)
在除门控模式以外的所有模式下，当使能从模式控制器后在 TRGI 输入上检测到有效边沿时，该标志将由硬件置 1。选择门控模式时，该标志将在计数器启动或停止时置 1。但需要通过软件清零。
0：未发生触发事件。
1：触发中断挂起。

位 5:3 保留，必须保持复位值。

位 2 **CC2IF**：捕获/比较 2 中断标志 (Capture/Compare 2 interrupt flag)

请参见 CC1IF 说明

位 1 **CC1IF**：捕获/比较 1 中断标志 (Capture/compare 1 interrupt flag)

如果通道 CC1 配置为输出：

当计数器与比较值匹配时，此标志由硬件置 1。但需要通过软件清零。

0：不匹配。

1：TIMx_CNT 计数器的值与 TIMx_CCR1 寄存器的值匹配。当 TIMx_CCR1 的值大于 TIMx_ARR 的值时，CC1IF 位将在计数器发生上溢时变为高电平。

如果通道 CC1 配置为输入：

此位将在发生捕获事件时由硬件置 1。通过软件或读取 TIMx_CCR1 寄存器将该位清零。

0：未发生输入捕获事件。

1：TIMx_CCR1 寄存器中已捕获到计数器值（IC1 上已检测到与所选极性匹配的边沿）。

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– 上溢且当 TIMx_CR1 寄存器中 UDIS =“0”时。
– 当由于 TIMx_CR1 寄存器中 URS =“0”且 UDIS =“0”而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。
– 当由于 TIMx_CR1 寄存器中 URS =“0”且 UDIS =“0”而通过触发事件（请参见同步控制寄存器说明）重新初始化 CNT 时。

22.4.6 TIM21/22 事件生成寄存器 (TIMx_EGR)

TIM21/22 event generation register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

位 15:7 保留，必须保持复位值。

位 6 **TG**：触发生成 (Trigger generation)
此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：TIMx_SR 寄存器中的 TIF 标志置 1。使能后可发生相关中断

位 5:3 保留，必须保持复位值。

位 2 **CC2G**：捕获/比较 2 生成 (Capture/compare 2 generation)
请参见 CC1G 说明

位 1 **CC1G**：捕获/比较 1 生成 (Capture/Compare 1 generation)
此位由软件置 1 以生成事件，并由硬件自动清零。
0：不执行任何操作
1：通道 1 上生成捕获/比较事件：
如果通道 CC1 配置为输出：
使能后，CC1IF 标志置 1 并发送相应中断。
如果通道 CC1 配置为输入：
TIMx_CCR1 寄存器中将捕获到计数器的当前值。使能后，CC1IF 标志置 1 并发送相应中断。如果 CC1IF 标志已为高电平，CC1OF 标志将置 1。

位 0 **UG**：更新生成 (Update generation)
该位可通过软件置 1，并由硬件自动清零。
0：不执行任何操作
1：重新初始化计数器并生成寄存器更新事件。预分频器计数器也将清零，但预分频比不受影响。计数器清零。

22.4.7 TIM21/22 捕获/比较模式寄存器 1 (TIMx_CCMR1)

TIM21/22 capture/compare mode register 1

偏移地址：0x18

复位值：0x0000

这些通道可用于输入（捕获模式）或输出（比较模式）模式。通道方向通过配置相应的 CCxS 位进行定义。该寄存器的所有其它位在输入模式和输出模式下的功能不同。对于任一给定位，OCxx 用于说明通道配置为输出模式时该位对应的功能，ICxx 则用于说明通道配置为输入模式时该位对应的功能。因此，必须注意同一个位在输入阶段和输出阶段具有不同的含义。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	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

输出比较模式

位 15 保留，必须保持复位值。

位 14:12 **OC2M[2:0]**：输出比较 2 模式 (Output Compare 2 mode)

位 11 **OC2PE**：输出比较 2 预装载使能 (Output Compare 2 preload enable)

位 10 **OC2FE**：输出比较 2 快速使能 (Output Compare 2 fast enable)

位 9:8 **CC2S[1:0]**：捕获/比较 2 选择 (Capture/Compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，IC2 映射到 TI2 上

10：CC2 通道配置为输入，IC2 映射到 TI1 上

11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7 保留，必须保持复位值。

位 6:4 **OC1M**：输出比较 1 模式 (Output Compare 1 mode)

这些位定义提供 OC1 和 OC1N 的输出参考信号 OC1REF 的行为。OC1REF 为高电平有效，而 OC1 和 OC1N 的有效电平则分别取决于 CC1P 位和 CC1NP 位。

000：冻结——输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 进行比较不会对输出造成任何影响。（该模式用于生成时基）。

001：将通道 1 设置为匹配时输出有效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为高电平。

010：将通道 1 设置为匹配时输出无效电平。当计数器 TIMx_CNT 与捕获/比较寄存器 1 (TIMx_CCR1) 匹配时，OC1REF 信号强制变为低电平。

011：翻转——TIMx_CNT=TIMx_CCR1 时，OC1REF 发生翻转

100：强制变为无效电平——OC1REF 强制变为低电平

101：强制变为有效电平——OC1REF 强制变为高电平

110：PWM 模式 1——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为无效状态（OC1REF=“0”），否则为有效状态（OC1REF=“1”）

111：PWM 模式 2——在递增计数模式下，只要 TIMx_CNT<TIMx_CCR1，通道 1 便为无效状态，否则为有效状态。在递减计数模式下，只要 TIMx_CNT>TIMx_CCR1，通道 1 便为有效状态，否则为无效状态。

注：在 PWM 模式 1 或 PWM 模式 2 下，仅当比较结果发生改变或输出比较模式由“冻结”模式切换到“PWM”模式时，OCREF 电平才会发生更改。

位 3 **OC1PE**：输出比较 1 预装载使能 (Output Compare 1 preload enable)

0：禁止与 TIMx_CCR1 相关的预装载寄存器。可随时向 TIMx_CCR1 写入数据，写入后将立即使用新值。

1：使能与 TIMx_CCR1 相关的预装载寄存器。可读/写访问预装载寄存器。TIMx_CCR1 预装载值在每次生成更新事件时都会装载到活动寄存器中

注：只有单脉冲模式下才可在未验证预装载寄存器的情况下使用 PWM 模式（TIMx_CR1 寄存器中的 OPM 位置 1）。其它情况下则无法保证该行为。

位 2 **OC1FE**：输出比较 1 快速使能 (Output Compare 1 fast enable)

此位用于加快触发输入事件对 CC 输出的影响。

0：即使触发开启，CC1 也将根据计数器和 CCR1 值正常工作。触发输入出现边沿时，激活 CC1 输出的最短延迟时间为 5 个时钟周期

1：触发输入上出现有效边沿相当于 CC1 输出上的比较匹配。随后，无论比较结果如何，OC 都设置为比较电平。采样触发输入和激活 CC1 输出的延迟时间缩短为 3 个时钟周期。仅当通道配置为 PWM1 或 PWM2 模式时，OC1FE 才会起作用。

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 Selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，IC1 映射到 TI1 上

10：CC1 通道配置为输入，IC1 映射到 TI2 上

11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注：仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

输入捕获模式

位 15:12 **IC2F**：输入捕获 2 滤波器 (Input capture 2 filter)

位 11:10 **IC2PSC[1:0]**：输入捕获 2 预分频器 (Input capture 2 prescaler)

位 9:8 **CC2S**：捕获/比较 2 选择 (Capture/Compare 2 selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC2 通道配置为输出

01：CC2 通道配置为输入，IC2 映射到 TI2 上

10：CC2 通道配置为输入，IC2 映射到 TI1 上

11：CC2 通道配置为输入，IC2 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC2E = 0），才可向 CC2S 位写入数据。

位 7:4 **IC1F**：输入捕获 1 滤波器 (Input capture 1 filter)

此位域可定义 TI1 输入的采样频率和适用于 TI1 的数字滤波器带宽。数字滤波器由事件计数器组成，每 N 个连续事件才视为一个有效输出边沿：

0000：无滤波器，按 f_{DTS} 频率进行采样

0001：$f_{SAMPLING}=f_{CK_INT}$，N=2

0010：$f_{SAMPLING}=f_{CK_INT}$，N=4

0011：$f_{SAMPLING}=f_{CK_INT}$，N=8

0100：$f_{SAMPLING}=f_{DTS}/2$，N=6

0101：$f_{SAMPLING}=f_{DTS}/2$，N=8

0110：$f_{SAMPLING}=f_{DTS}/4$，N=6

0111：$f_{SAMPLING}=f_{DTS}/4$，N=8

1000：$f_{SAMPLING}=f_{DTS}/8$，N=6

1001：$f_{SAMPLING}=f_{DTS}/8$，N=8

1010：$f_{SAMPLING}=f_{DTS}/16$，N=5

1011：$f_{SAMPLING}=f_{DTS}/16$，N=6

1100：$f_{SAMPLING}=f_{DTS}/16$，N=8

1101：$f_{SAMPLING}=f_{DTS}/32$，N=5

1110：$f_{SAMPLING}=f_{DTS}/32$，N=6

1111：$f_{SAMPLING}=f_{DTS}/32$，N=8

位 3:2 **IC1PSC**：输入捕获 1 预分频器 (Input capture 1 prescaler)

此位域定义 CC1 输入 (IC1) 的预分频比。

只要 CC1E=“0”（TIMx_CCER 寄存器），预分频器便立即复位。

00：无预分频器，捕获输入上每检测到一个边沿便执行捕获

01：每发生 2 个事件便执行一次捕获

10：每发生 4 个事件便执行一次捕获

11：每发生 8 个事件便执行一次捕获

位 1:0 **CC1S**：捕获/比较 1 选择 (Capture/Compare 1 Selection)

此位域定义通道方向（输入/输出）以及所使用的输入。

00：CC1 通道配置为输出

01：CC1 通道配置为输入，IC1 映射到 TI1 上

10：CC1 通道配置为输入，IC1 映射到 TI2 上

11：CC1 通道配置为输入，IC1 映射到 TRC 上。此模式仅在通过 TS 位（TIMx_SMCR 寄存器）选择内部触发输入时有效

注： 仅当通道关闭时（TIMx_CCER 中的 CC1E = 0），才可向 CC1S 位写入数据。

22.4.8 TIM21/22 捕获/比较使能寄存器 (TIMx_CCER)

TIM21/22 capture/compare enable register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

位 15:8 保留，必须保持复位值。

位 7 **CC2NP**：捕获/比较 2 输出极性 (Capture/Compare 2 output Polarity)
请参见 CC1NP 说明

位 6 保留，必须保持复位值。

位 5 **CC2P**：捕获/比较 2 输出极性 (Capture/Compare 2 output Polarity)
请参见 CC1P 说明

位 4 **CC2E**：捕获/比较 2 输出使能 (Capture/Compare 2 output enable)
请参见 CC1E 说明

位 3 **CC1NP**：捕获/比较 1 互补输出极性 (Capture/Compare 1 complementary output Polarity)
CC1 通道配置为输出：CC1NP 必须保持清零
CC1 通道配置为输入：CC1NP 与 CC1P 配合使用可定义 TI1FP1/TI2FP1 的极性（请参见 CC1P 说明）。

位 2 保留，必须保持复位值。

位 1 **CC1P**：捕获/比较 1 输出极性 (Capture/Compare 1 output Polarity)。
CC1 通道配置为输出：
0：OC1 高电平有效。
1：OC1 低电平有效。
CC1 通道配置为输入：
CC1NP/CC1P 位可针对触发或捕获操作选择 TI1FP1 和 TI2FP1 的极性。
00：未反相/上升沿触发
电路对 TIxFP1 上升沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式或编码器模式下执行触发操作）。
01：反相/下降沿触发
电路对 TIxFP1 下降沿敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 反相（在门控模式或编码器模式下执行触发操作）。
10：保留，不使用此配置。

注： 11：未反相/上升沿和下降沿均触发
电路对 TIxFP1 上升沿和下降沿都敏感（在复位模式、外部时钟模式或触发模式下执行捕获或触发操作），TIxFP1 未反相（在门控模式下执行触发操作）。编码器模式下不得使用此配置。

位 0 **CC1E**：捕获/比较 1 输出使能 (Capture/Compare 1 output enable)。
CC1 通道配置为输出：
0：关闭——OC1 未激活。
1：开启——在相应输出引脚上输出 OC1 信号。
CC1 通道配置为输入：
此位决定了是否可以实际将计数器值捕获到输入捕获/比较寄存器 1 (TIMx_CCR1) 中。
0：禁止捕获。
1：使能捕获。

表 93. 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：与标准 OCx 通道相连的外部 I/O 引脚的状态取决于通道 OCx 的状态以及 GPIO 寄存器。

22.4.9 TIM21/22 计数器 (TIMx_CNT)

TIM21/22 counter

偏移地址：0x24

复位值：0x0000 0000

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

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

22.4.10 TIM21/22 预分频器 (TIMx_PSC)

TIM21/22 prescaler

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 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

22.4.11 TIM21/22 自动重载寄存器 (TIMx_ARR)

TIM21/22 auto-reload register

偏移地址：0x2C

复位值：0x0000 0000

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 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的更多详细信息，请参见*第 496 页的第 22.3.1 节：时基单元*。

当自动重载值为空时，计数器不工作。

22.4.12 TIM21/22 捕获/比较寄存器 1 (TIMx_CCR1)

TIM21/22 capture/compare register 1

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 15:0 **CCR1[15:0]**：捕获/比较 1 值 (Capture/Compare 1 value)

如果通道 CC1 配置为输出：

CCR1 是捕获/比较寄存器 1 的预装载值。

如果没有通过 TIMx_CCMR1 寄存器中的 OC1PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 1）。

实际捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC1 输出上发出信号的值。

如果通道 CC1 配置为输入：

CCR1 为上一个输入捕获 1 事件 (IC1) 发生时的计数器值。

22.4.13 TIM21/22 捕获/比较寄存器 2 (TIMx_CCR2)

TIM21/22 capture/compare register 2

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 15:0 **CCR2[15:0]**：捕获/比较 2 值 (Capture/Compare 2 value)

如果通道 CC2 配置为输出：

CCR2 是捕获/比较寄存器 2 的预装载值。

如果没有通过 TIMx_CCMR2 寄存器中的 OC2PE 位来使能预装载功能，则该值立刻生效；否则只在发生更新事件时生效（拷贝到实际起作用的捕获/比较寄存器 2）。

活动捕获/比较寄存器中包含要与计数器 TIMx_CNT 进行比较并在 OC2 输出上发出信号的值。

如果通道 CC2 配置为输入：

CCR2 为上一个输入捕获 2 事件 (IC2) 发生时的计数器值。

22.4.14 TIM21 选项寄存器 (TIM21_OR)

TIM21 option register

偏移地址：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2_RMP	TI1_RMP			ETR_RMP	
										rw	rw	rw	rw	rw	rw

位 15:6 保留，必须保持复位值。

位 5 **TI2_RMP**：Timer21 TI2（连接到 TIM21_CH1）重映射 (Timer21 TI2 (connected to TIM21_CH1) remap)

此位由软件置 1 和清零。

0：TIM21 TI2 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

1：TIM21 TI2 输入连接到 COMP2_OUT。

位 4:2 **TI1_RMP**：Timer21 TI1（连接到 TIM21_CH1）重映射 (Timer21 TI1 (connected to TIM21_CH1) remap)

此位由软件置 1 和清零。

000：TIM21 TI1 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

001：TIM21 TI1 输入连接到 RTC WAKEUP 中断

010：TIM21 TI1 输入连接到 HSE_RTC 时钟

011：TIM21 TI1 输入连接到 MSI 时钟

100：TIM21 TI1 输入连接到 LSE 时钟

101：TIM21 TI1 输入连接到 LSI 时钟

110：TIM21 TI1 输入连接到 COMP1_OUT

111：TIM21 TI1 输入连接到 MCO 时钟

位 1:0 **ETR_RMP**：Timer21 ETR 重映射 (Timer21 ETR remap)

此位由软件置 1 和清零。

00：TIM21 ETR 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

01：TIM21 ETR 输入连接到 COMP2_OUT

10：TIM21 ETR 输入连接到 COMP1_OUT

11：TIM21 ETR 输入连接到 LSE 时钟

22.4.15 TIM22 选项寄存器 (TIM22_OR)

TIM22 option register

偏移地址：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP		ETR_RMP	
												rw	rw	rw	rw

位 15:4 保留，必须保持复位值。

位 3:2 **TI1_RMP**：Timer22 TI1（连接到 TIM22_CH1）重映射 (Timer22 TI1 (connected to TIM22_CH1) remap)

此位由软件置 1 和清零。

00：TIM22 TI1 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

01：TIM22 TI1 输入连接到 COMP2_OUT

10：TIM22 TI1 输入连接到 COMP1_OUT

11：TIM22 TI1 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

位 1:0 **ETR_RMP**：Timer22 ETR 重映射 (Timer22 ETR remap)

此位由软件置 1 和清零。

00：TIM22 ETR 输入连接到 GPIO。请参见器件数据手册中的备用功能映射表。

01：TIM22 ETR 输入连接到 COMP2_OUT

10：TIM22 ETR 输入连接到 COMP1_OUT

11：TIM22 ETR 输入连接到 LSE 时钟

22.4.16 TIM21/22 寄存器映射

下表提供了 TIM21/22 寄存器映射和复位值。

表 94. TIM21/22 寄存器映射和复位值

偏移	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value							0	0	0			0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value										0	0	0				
0x08	**TIMx_SMCR**	ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value										0				0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value						0	0			0				0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value										0				0	0	0
0x18	**TIMx_CCMR1 Output Compare mode**	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Res.																
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value									0		0	0	0		0	0
0x24	**TIMx_CNT**	CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 94. TIM21/22 寄存器映射和复位值（续）

偏移	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**TIMx_PSC**	PSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Res.																
0x34	**TIMx_CCR1**	CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Res.																
0x38	**TIMx_CCR2**	CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TIM21_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2_RMP	TI1_RMP			ETR_RMP	
	Reset value											0	0	0	0	0	0
0x50	**TIM22_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1_RMP		ETR_RMP	
	Reset value													0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

23 基本定时器 (TIM6/7)

23.1 简介

基本定时器 TIM6 和 TIM7 包含一个 16 位自动重载计数器，该计数器由可编程预分频器驱动。

此类定时器不仅可用作通用定时器以生成时基，还可以专门用于驱动数模转换器 (DAC)。实际上，此类定时器内部连接到 DAC 并能够通过其触发输出驱动 DAC。

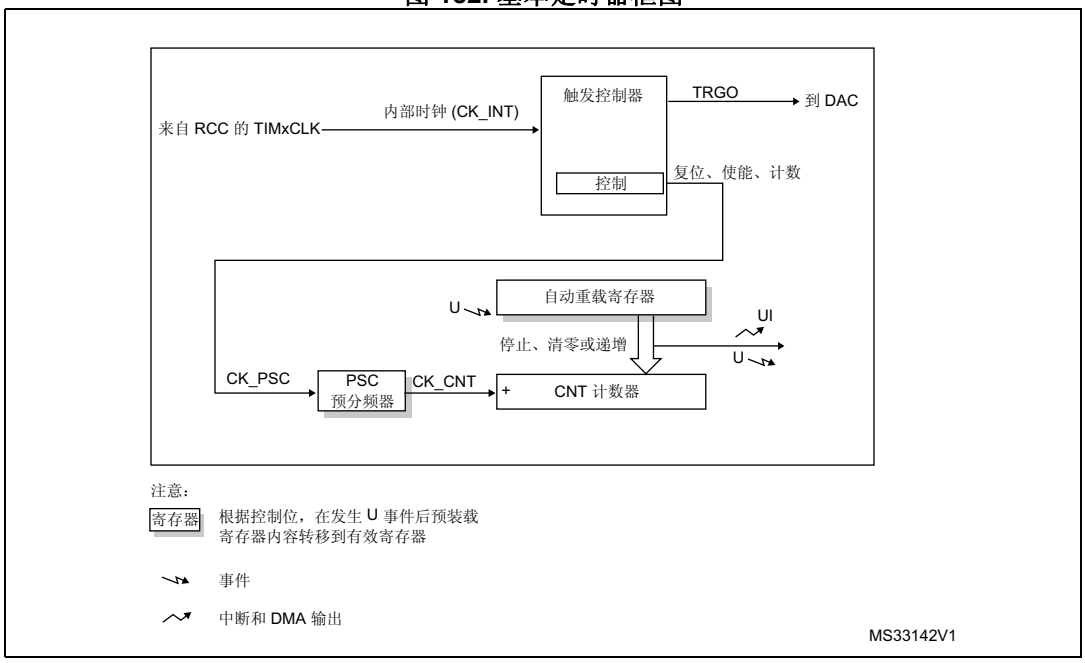
这些定时器彼此完全独立，不共享任何资源。

23.2 TIM6/7 主要特性

基本定时器 (TIM6/TIM7) 的特性包括：

- 16 位自动重载递增计数器
- 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于 1 和 65536 之间
- 用于触发 DAC 的同步电路
- 发生如下更新事件时会生成中断/DMA 请求：计数器上溢

图 182. 基本定时器框图



23.3 TIM6/7 功能说明

23.3.1 时基单元

可编程定时器的主要模块由一个 16 位递增计数器及其相关的自动重载寄存器组成。计数器的时钟可通过预分频器进行分频。

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

时基单元包括：

- 计数器寄存器 (TIMx_CNT)
- 预分频器寄存器 (TIMx_PSC)
- 自动重载寄存器 (TIMx_ARR)

自动重载寄存器是预装载的。每次尝试对自动重载寄存器执行读写操作时，都会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器，也可以在每次发生更新事件 UEV 时传送到影子寄存器，这取决于 TIMx_CR1 寄存器中的自动重载预装载使能位 (ARPE)。当计数器达到上溢值并且 TIMx_CR1 寄存器中的 UDIS 位为 0 时，将发送更新事件。该更新事件也可由软件产生。下文将针对各配置的更新事件的产生进行详细介绍。

计数器由预分频器输出 CK_CNT 提供时钟，仅当 TIMx_CR1 寄存器中的计数器启动位 (CEN) 置 1 时，才会启动计数器。

请注意，实际的计数器使能信号 CNT_EN 在 CEN 置 1 的一个时钟周期后被置 1。

预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 和 65536 之间。该预分频器基于 TIMx_PSC 寄存器中的 16 位寄存器所控制的 16 位计数器。由于 TIMx_PSC 控制寄存器有缓冲，因此可对预分频器进行实时更改。而新的预分频比将在下一更新事件发生时被采用。

图 183 和 *图 184* 以一些示例说明在预分频比实时变化时计数器的行为。

图 183. 预分频器分频由 1 变为 2 时的计数器时序图

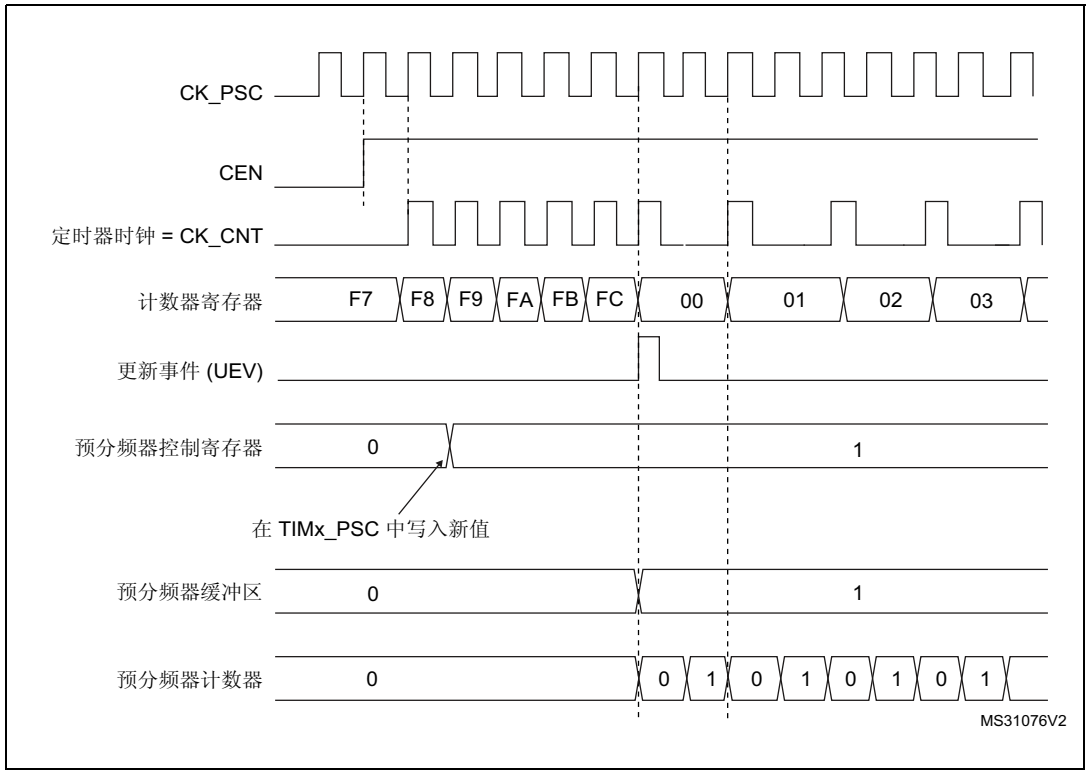
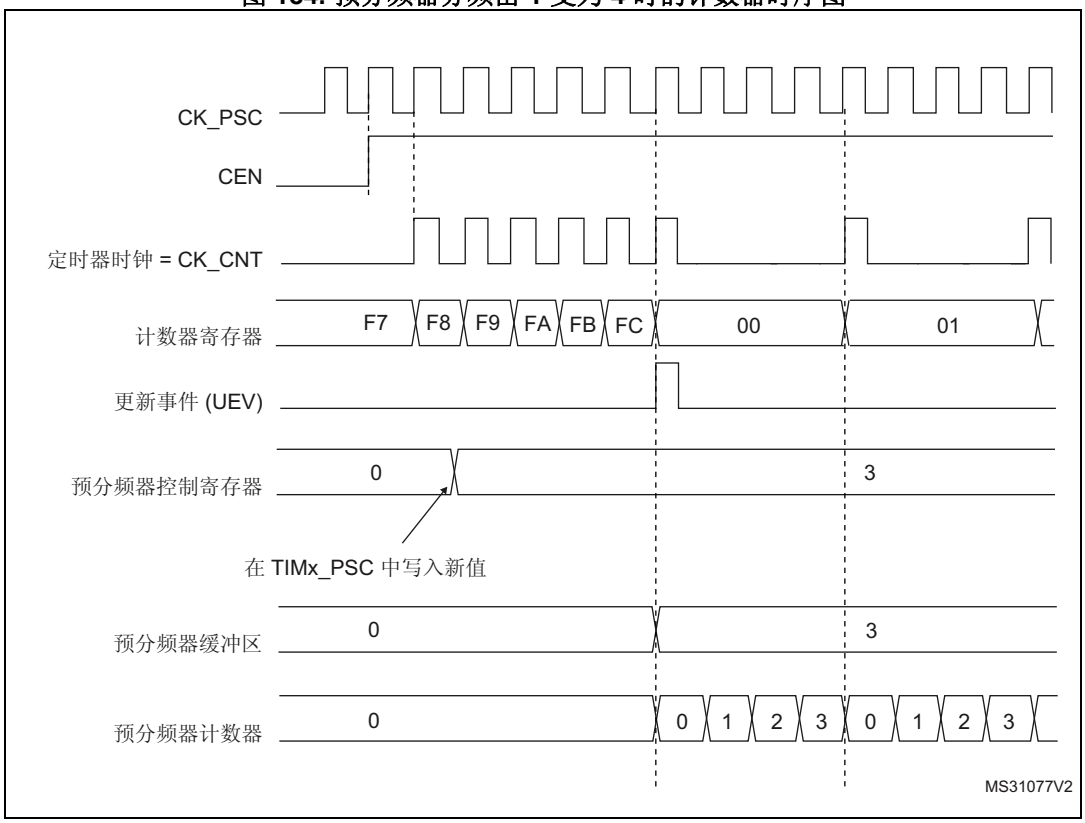


图 184. 预分频器分频由 1 变为 4 时的计数器时序图



23.3.2 计数模式

计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

每次发生计数器上溢时会生成更新事件，或将 TIMx_EGR 寄存器中的 UG 位置 1（通过软件或使用从模式控制器）也可以产生更新事件。

通过软件将 TIMx_CR1 寄存器中的 UDIS 位置 1 可禁止 UEV 事件。这可避免向预装载寄存器写入新值时更新影子寄存器。这样，直到 UDIS 位中写入 0 前便不会生成任何更新事件，但计数器和预分频器计数器都会重新从 0 开始计数（而预分频比保持不变）。此外，如果 TIMx_CR1 寄存器中的 URS 位（更新请求选择）已置 1，则将 UG 位置 1 会生成更新事件 UEV，但不会将 UIF 标志置 1（因此，不会发送任何中断或 DMA 请求）。

发生更新事件时，将更新所有寄存器且将更新标志（TIMx_SR 寄存器中的 UIF 位）置 1（取决于 URS 位）：

- 预分频器的缓冲区中将重新装载预装载值（TIMx_PSC 寄存器的内容）
- 使用预装载值 (TIMx_ARR) 更新自动重载影子寄存器

以下各图以一些示例说明当 TIMx_ARR=0x36 时不同时钟频率下计数器的行为。

图 185. 计数器时序图，1 分频内部时钟

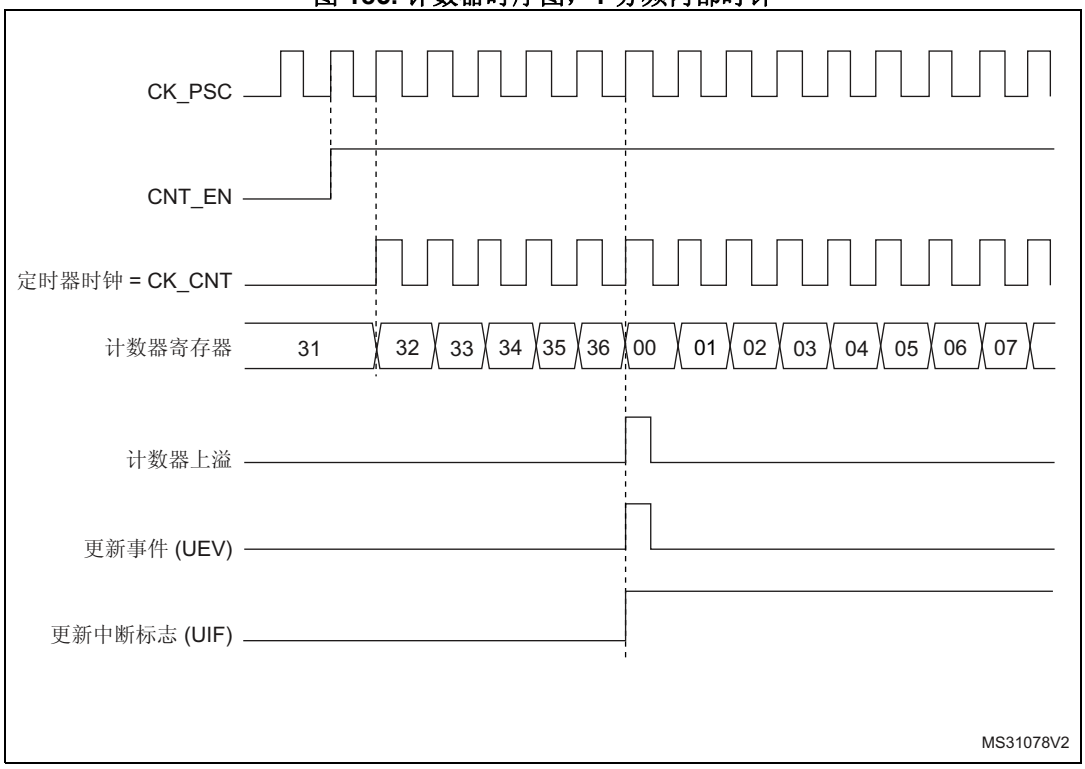


图 186. 计数器时序图，2 分频内部时钟

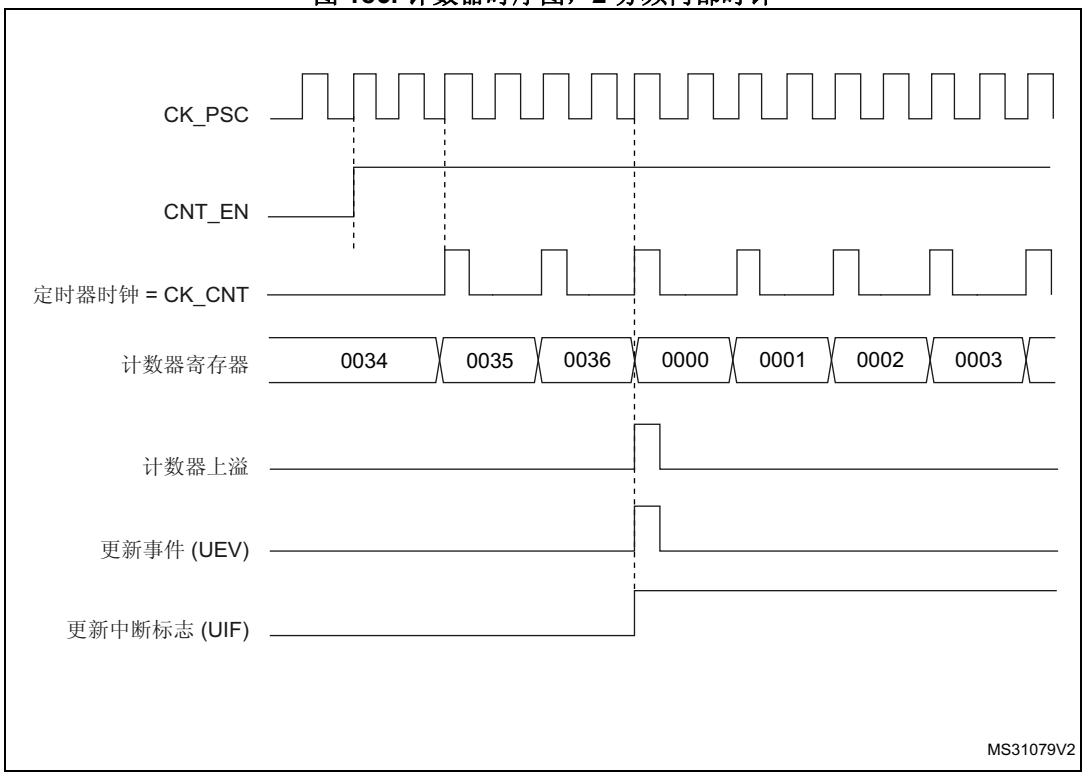


图 187. 计数器时序图，4 分频内部时钟

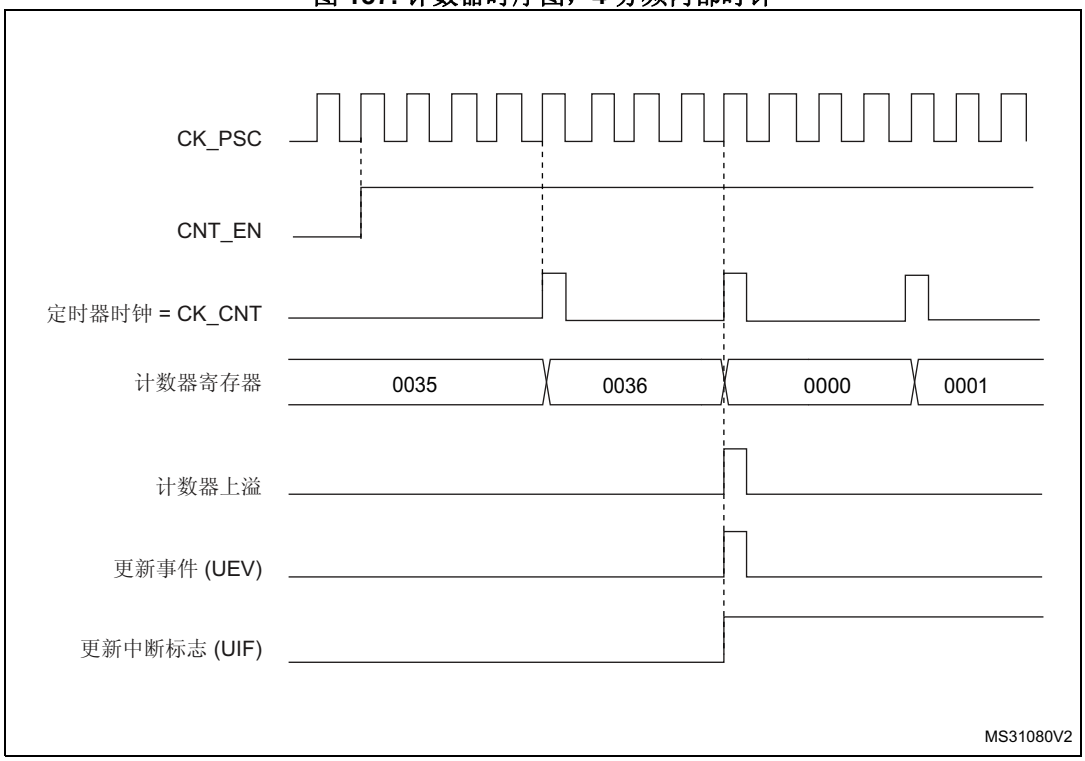


图 188. 计数器时序图，N 分频内部时钟

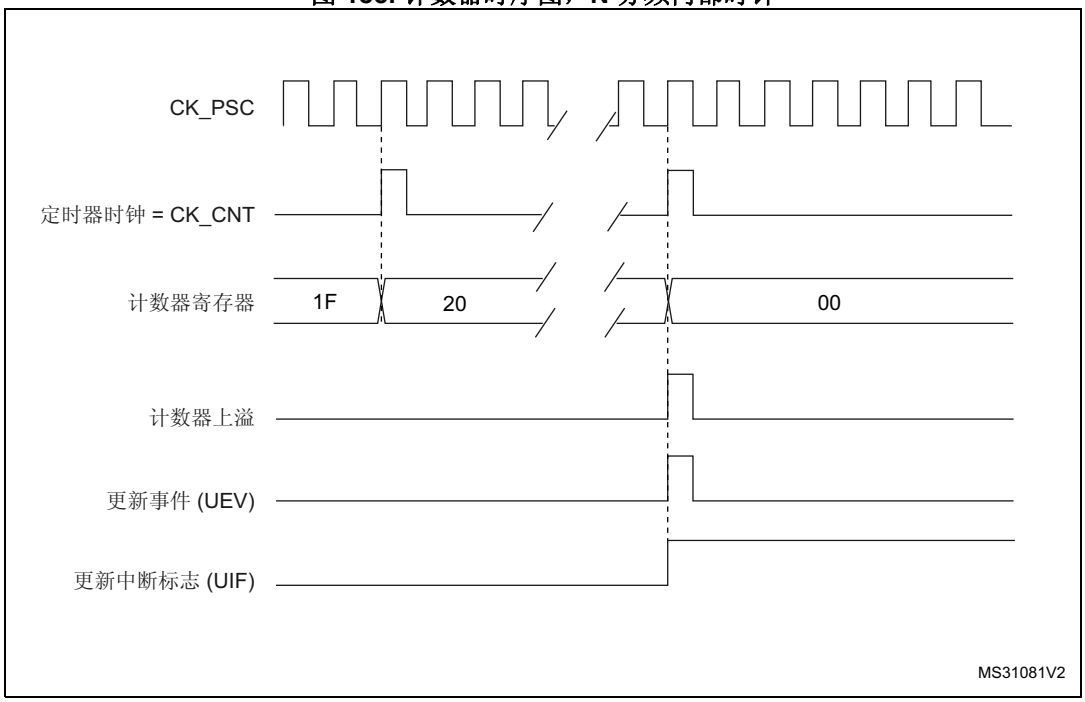


图 189. 计数器时序图，ARPE = 0 时更新事件（TIMx_ARR 未预装载）

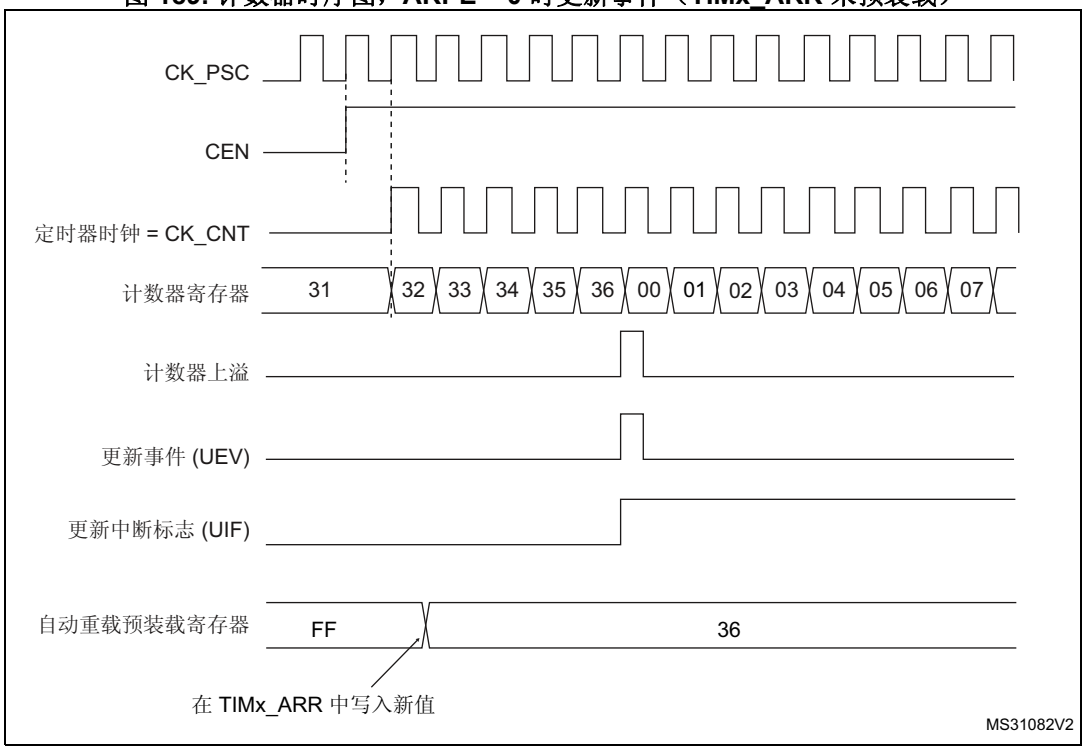
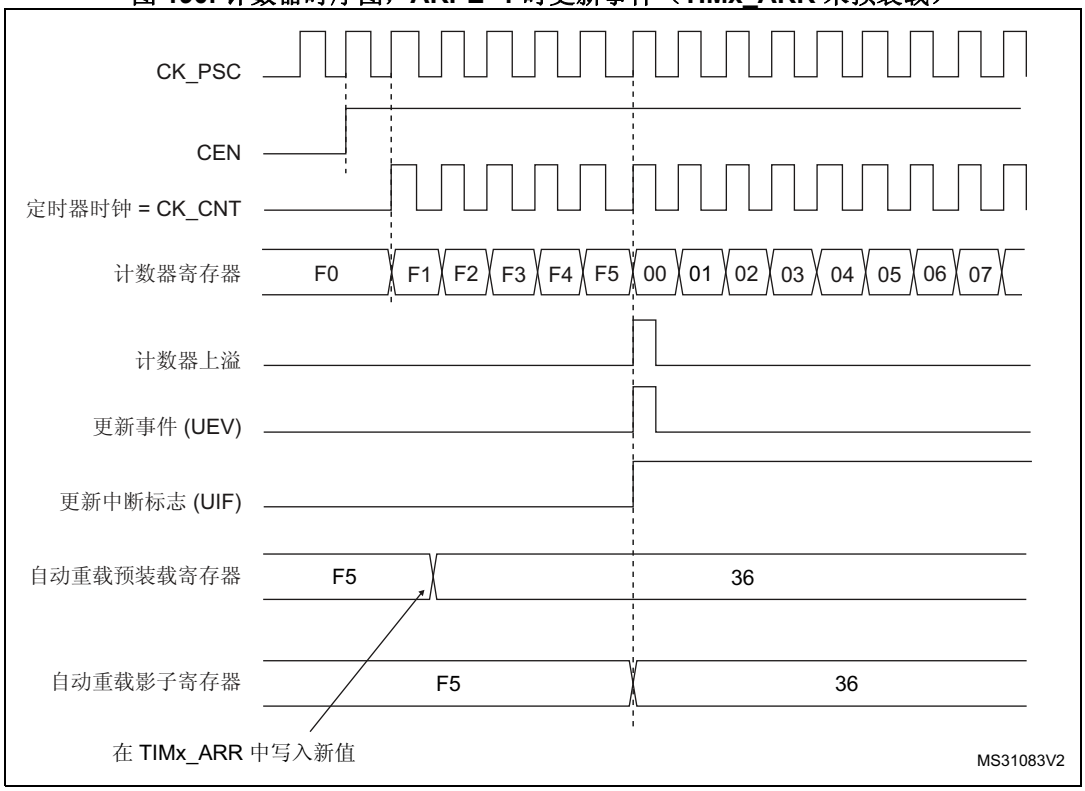


图 190. 计数器时序图，ARPE=1 时更新事件（TIMx_ARR 未预装载）



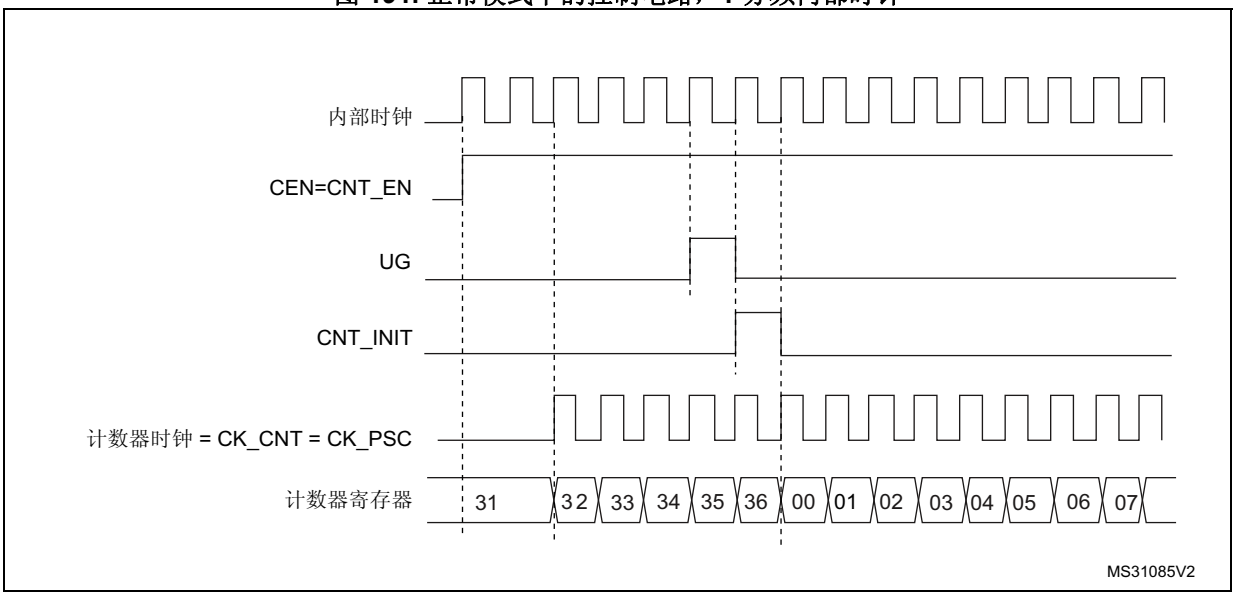
23.3.3 时钟源

计数器时钟由内部时钟 (CK_INT) 源提供。

CEN（TIMx_CR1 寄存器中）和 UG 位（TIMx_EGR 寄存器中）为实际控制位，并且只能通过软件进行更改（保持自动清零的 UG 除外）。当对 CEN 位写入 1 时，预分频器的时钟就由内部时钟 CK_INT 提供。

图 191 显示了正常模式下控制电路与递增计数器的行为（没有预分频的情况下）。

图 191. 正常模式下的控制电路，1 分频内部时钟



23.3.4 调试模式

当微控制器进入调试模式时（Cortex®-M0+ 内核停止），TIMx 计数器会根据 DBG 模块中的 DBG_TIMx_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见 *第 33.9.2 节：对定时器、看门狗和 I²C 的调试支持*。

23.4 TIM6/7 寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节：寄存器相关缩写词列表*。

外设寄存器的写访问仅支持半字（16 位）或字（32 位）。而读访问可支持字节（8 位）、半字（16 位）或字（32 位）。

23.4.1 TIM6/7 控制寄存器 1 (TIMx_CR1)

TIM6/7 control register 1

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

位 15:8 保留，必须保持复位值。

位 7 **ARPE**：自动重载预装载使能 (Auto-reload preload enable)

0：TIMx_ARR 寄存器不进行缓冲。
1：TIMx_ARR 寄存器进行缓冲。

位 6:4 保留，必须保持复位值。

位 3 **OPM**：单脉冲模式 (One-pulse mode)

0：计数器在发生更新事件时不会停止计数
1：计数器在发生下一更新事件时停止计数（将 CEN 位清零）。

位 2 **URS**：更新请求源 (Update request source)

此位由软件置 1 和清零，用以选择 UEV 事件源。
0：使能时，所有以下事件都会产生更新中断或 DMA 请求。此类事件包括：
- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

1：使能时，只有计数器上溢/下溢会生成更新中断或 DMA 请求。

位 1 **UDIS**：更新禁止 (Update disable)

此位由软件置 1 和清零，用以使能/禁止 UEV 事件生成。
0：使能 UEV。更新 (UEV) 事件可通过以下事件之一产生：
- 计数器上溢/下溢
- 将 UG 位置 1
- 通过从模式控制器生成的更新事件

然后更新影子寄存器的值。
1：禁止 UEV。不会生成更新事件，各影子寄存器的值（ARR 和 PSC）保持不变。但如果将 UG 位置 1，或者从从模式控制器接收到硬件复位，则会重新初始化计数器和预分频器。

位 0 **CEN**：计数器使能 (Counter enable)

0：禁止计数器
1：使能计数器

注：只有事先通过软件将 CEN 位置 1，才可以使用门控模式。而触发模式可通过硬件自动将 CEN 位置 1。

在单脉冲模式下，当发生更新事件时会自动将 CEN 位清零。

23.4.2 TIM6/7 控制寄存器 2 (TIMx_CR2)

TIM6/7 control register 2

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

位 15:7 保留，必须保持复位值。

位 6:4 **MMS**：主模式选择 (Master mode selection)

这些位用于选择主模式下将要发送到从定时器以实现同步的信息 (TRGO)。这些位的组合如下：

000：**复位**——TIMx_EGR 寄存器中的 UG 位用作触发输出 (TRGO)。如果复位由触发输入产生（从模式控制器配置为复位模式），则 TRGO 上的信号相比实际复位会有延迟。

001：**使能**——计数器使能信号 CNT_EN 用作触发输出 (TRGO)。该触发输出可用于同时启动多个定时器，或者控制在一段时间内使能从定时器。计数器使能信号由 CEN 控制位与门控模式下的触发输入的逻辑或运算组合而成。

当计数器使能信号由触发输入控制时，TRGO 上会存在延迟，选择主/从模式时除外（请参见 TIMx_SMCR 寄存器中对 MSM 位的说明）。

010：**更新**——选择更新事件作为触发输出 (TRGO)。例如，主定时器可用作从定时器的预分频器。

位 3:0 保留，必须保持复位值。

23.4.3 TIM6/7 DMA/中断使能寄存器 (TIMx_DIER)

TIM6/7 DMA/Interrupt enable register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

位 15:9 保留，必须保持复位值。

位 8 **UDE**：更新 DMA 请求使能 (Update DMA request enable)

0：禁止更新 DMA 请求。
1：使能更新 DMA 请求。

位 7:1 保留，必须保持复位值。

位 0 **UIE**：更新中断使能 (Update interrupt enable)

0：禁止更新中断。
1：使能更新中断。

23.4.4 TIM6/7 状态寄存器 (TIMx_SR)

TIM6/7 status register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

位 15:1 保留，必须保持复位值。

位 0 **UIF**：更新中断标志 (Update interrupt flag)

该位在发生更新事件时通过硬件置 1。但需要通过软件清零。

0：未发生更新。

1：更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1：

– 重复计数器值上溢或下溢并且 TIMx_CR1 寄存器中的 UDIS=0 时。

– 当由于 TIMx_CR1 寄存器中 URS = 0 且 UDIS = 0 而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。

23.4.5 TIM6/7 事件产生寄存器 (TIMx_EGR)

TIM6/7 event generation register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

位 15:1 保留，必须保持复位值。

位 0 **UG**：更新生成 (Update generation)

该位可通过软件置 1，并由硬件自动清零。

0：不执行任何操作。

1：重新初始化定时器计数器并生成寄存器更新事件。请注意，预分频器计数器也将清零（但预分频比不受影响）。

23.4.6 TIM6/7 计数器 (TIMx_CNT)

TIM6/7 counter

偏移地址：0x24

复位值：0x0000

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

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

23.4.7 TIM6/7 预分频器 (TIMx_PSC)

TIM6/7 prescaler

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 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)
计数器时钟频率 CK_CNT 等于 f_{CK_PSC} / (PSC[15:0] + 1)。
PSC 包含在每次发生更新事件时要装载到实际预分频器寄存器的值。

23.4.8 TIM6/7 自动重载寄存器 (TIMx_ARR)

TIM6/7 auto-reload register

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 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)
ARR 为要装载到实际自动重载寄存器的值。
有关 ARR 更新和行为的详细信息，请参见*第 548 页的第 23.3.1 节：时基单元*。
当自动重载值为空时，计数器不工作。

23.4.9 TIM6/7 寄存器映射

TIMx 寄存器可映射为 16 位可寻址寄存器，如下表所述：

表 95. TIM6/7 寄存器映射和复位值

偏移	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value									0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value										0	0	0				
0x08	Res.																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																0
0x18	Res.																
0x1C	Res.																
0x20	Res.																
0x24	**TIMx_CNT**	CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	PSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

24 低功耗定时器 (LPTIM)

24.1 简介

LPTIM 是一个 16 位定时器，可从降低功耗的最终发展中受益。由于 LPTIM 的时钟源具有多样性，因此 LPTIM 能够在所有电源模式（待机模式除外）下保持运行状态。即使没有内部时钟源，LPTIM 也能运行，鉴于这一点，可将其用作“脉冲计数器”，这种脉冲计数器在某些应用中十分有用。此外，LPTIM 还能将系统从低功耗模式唤醒，因此非常适合实现“超时功能”，而且功耗极低。

LPTIM 引入了一个灵活的时钟方案，该方案能够提供所需的功能和性能，同时还能最大程度地降低功耗。

24.2 LPTIM 主要特性

- 16 位递增计数器
- 3 位预分频器，可采用 8 种分频系数（1、2、4、8、16、32、64 和 128）
- 可选时钟
  - 内部时钟源：LSE、LSI、HSI16 或 APB 时钟
  - ULPTIM 输入的外部时钟源（在没有 LP 振荡器运行的情况下工作，由脉冲计数器应用使用）
- 16 位 ARR 自动重载寄存器
- 16 位比较寄存器
- 连续/单触发模式
- 可选软件/硬件输入触发
- 可编程数字干扰滤波器
- 可配置输出：脉冲和 PWM
- 可配置 I/O 极性
- 编码器模式

24.3 LPTIM 实现

表 96 介绍了 STM32L0x3 器件上的 LPTIM 实现。

表 96. STM32L0x3 LPTIM 特性

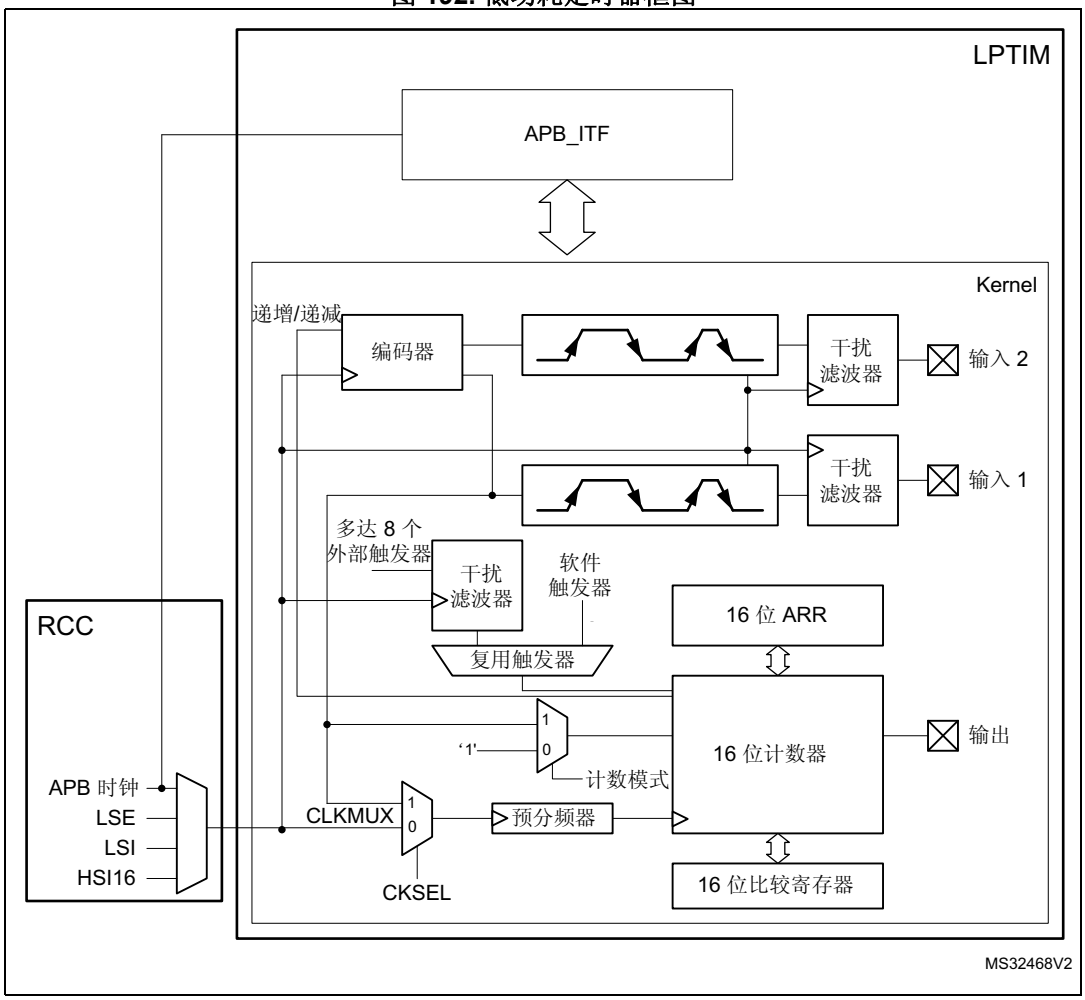
LPTIM 模式/特性(1)	LPTIM1
编码器模式	X

1. X = 支持。

24.4 LPTIM 功能说明

24.4.1 LPTIM 框图

图 192. 低功耗定时器框图



24.4.2 LPTIM 复位和时钟

LPTIM 可通过多个时钟源提供时钟。它可以由内部时钟信号提供时钟，内部时钟信号可通过时钟树控制器 (RCC) 在 APB、LSI、LSE 或 HSI16 时钟源中进行选择。此外，LPTIM 还可通过注入到其外部 Input1 上的外部时钟信号提供时钟。当通过外部时钟源提供时钟时，LPTIM 可以在下述两种可能配置中的其中一种配置下运行：

- 第一种配置是，LPTIM 通过外部信号提供时钟，但同时通过 APB 或 LSE、LSI 和 HSI16 等任何其他内置振荡器为 LPTIM 提供内部时钟信号。
- 第二种配置是，LPTIM 仅由外部时钟源通过外部 Input1 提供时钟。此配置可在进入低功耗模式后所有内置振荡器关闭时，用于实现超时功能或脉冲计数器功能。

对 CKSEL 和 COUNTMODE 位进行编程，可控制 LPTIM 使用外部时钟源还是内部时钟源。

当使用外部时钟源时，可使用 CKPOL 位选择外部时钟信号的有效边沿。如果上升沿和下降沿均为有效边沿，则还应提供内部时钟信号（第一种配置）。在这种情况下，内部时钟信号频率应至少为外部时钟信号频率的五倍。

24.4.3 干扰滤波器

LPTIM 输入（外部或内部）由数字滤波器保护，避免任何毛刺和噪声干扰在 LPTIM 内部传播，从而防止产生意外计数或触发。

在激活数字滤波器之前，首先应向 LPTIM 提供内部时钟源，这是保证滤波器正常工作的必要条件。

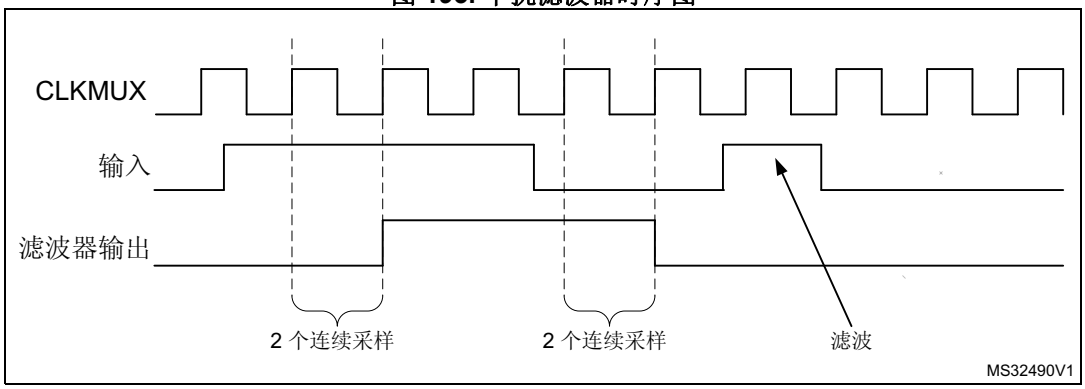
数字滤波器分为两组：

- 第一组数字滤波器保护 LPTIM 外部输入。数字滤波器的敏感性由 CKFLT 位控制。
- 第二组数字滤波器保护 LPTIM 内部触发输入。数字滤波器的敏感性由 TRGFLT 位控制。

注：数字滤波器的敏感性以组为单位进行控制。无法单独配置同一组内各个数字滤波器的敏感性。

滤波器的敏感性会影响相同的连续采样的数量，在其中一个 LPTIM 输入上检测到此类连续采样时，才能将某信号电平变化视为有效切换。*图 193* 给出了编程 2 个连续采样时，干扰滤波器行为的示例。

图 193. 干扰滤波器时序图



注：不提供内部时钟信号时，必须通过将 CKFLT 和 TRGFLT 位设为 0 来停用数字滤波器。在这种情况下，可使用外部模拟滤波器来防止 LPTIM 外部输入产生干扰。

24.4.4 预分频器

LPTIM 16 位计数器前面要有一个可配置的 2 次幂预分频器。预分频器的分频比由 PRESC[2:0] 3 位域进行控制。下表列出了所有可能的分频比：

表 97. 预分频器的分频比

编程	分频系数
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

24.4.5 触发多路复用器

LPTIM 计数器可通过软件启动，也可以在 8 个触发输入之一上检测到有效边沿后启动。

TRIGEN[1:0] 用于确定 LPTIM 触发源：

- TRIGEN[1:0] 等于“00”时，LPTIM 计数器会在通过软件将 CNTSTRT 位或 SNGSTRT 位其中之一置 1 后立即启动。
- TRIGEN[1:0] 的其余三个可能的值用于配置触发输入使用的有效边沿。LPTIM 计数器会在检测到有效边沿后立即启动。

TRIGEN[1:0] 不等于“00”时，TRIGSEL[2:0] 用于选择使用 8 个触发输入中的哪一个来启动计数器。

外部触发信号视为 LPTIM 的异步信号。因此，检测到触发信号后，由于同步问题，需要延迟两个计数器时钟周期，定时器才能开始运行。

如果在定时器已启动时发生新的触发事件，则此事件将被忽略（除非已使能超时功能）。

注： *必须使能定时器，才能将 SNGSTRT/CNTSTRT 位置 1。当定时器禁止时，对这些位执行的任何写操作都将被硬件丢弃。*

24.4.6 工作模式

LPTIM 支持以下两种工作模式：

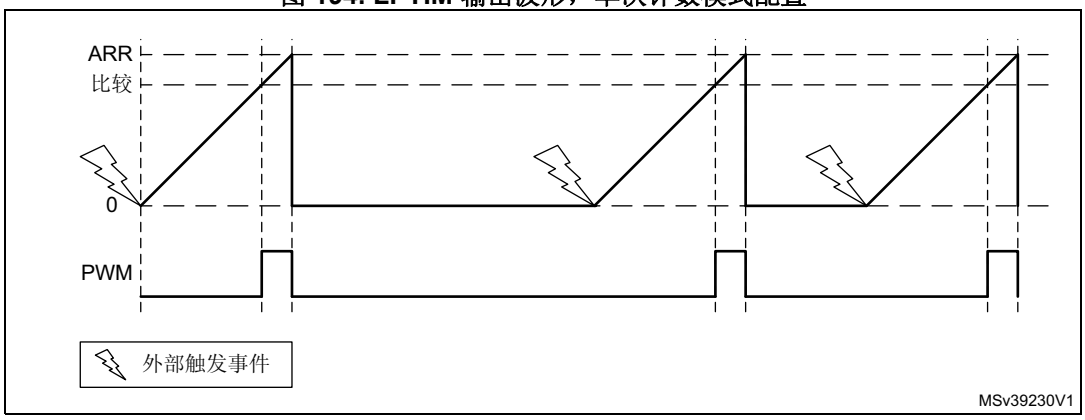
- 连续模式：定时器自由运行，由触发事件启动并且直到被禁止才会停止
- 单触发模式：定时器由触发事件启动，当达到 ARR 值时停止。

新的触发事件将重新启动定时器。从计数器启动到计数器达到 ARR，这段时间内发生的任何触发事件均将被丢弃。

要使能单触发计数，必须将 SNGSTRT 位置 1。

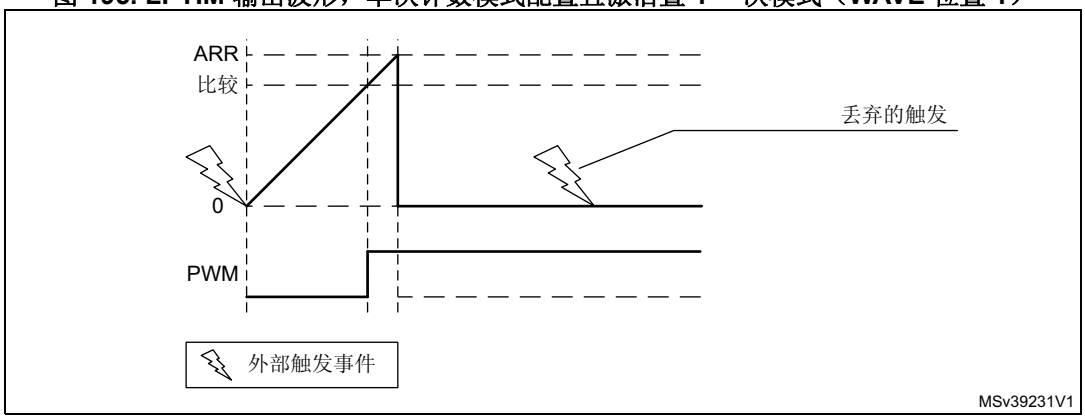
选择外部触发时，在 SNGSTRT 位置 1 后以及计数器寄存器停止后（包含零值）到达的每个外部触发事件都将为计数器启动新的单触发计数周期，如*图 194* 所示。

图 194. LPTIM 输出波形，单次计数模式配置



应注意，当 LPTIMx_CFGR 寄存器中的 WAVE 位域置 1 时，将激活置 1 一次模式。在这种情况下，计数器仅会在第一个触发事件后启动一次，任何后续触发事件都将被丢弃，如*图 194* 所示。

图 195. LPTIM 输出波形，单次计数模式配置且激活置 1 一次模式（WAVE 位置 1）



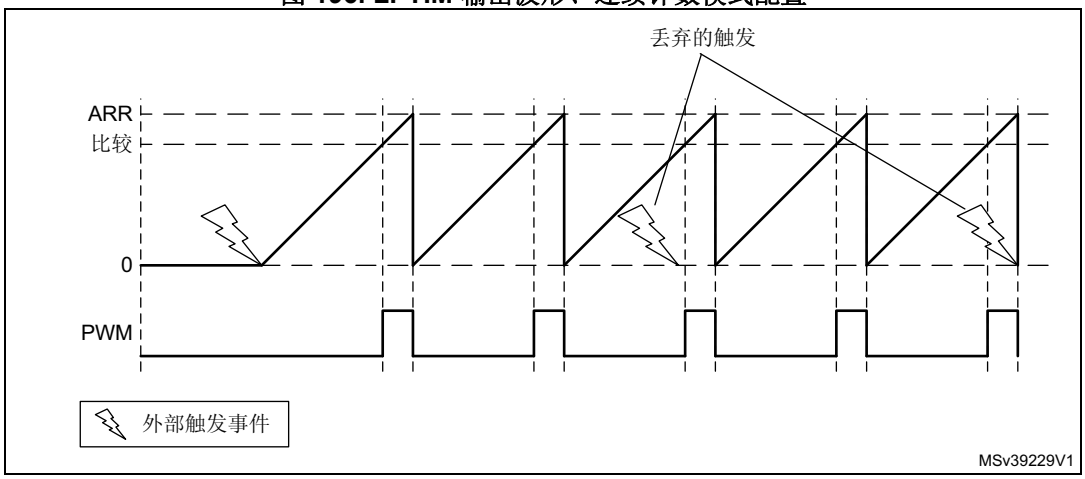
若通过软件启动（TRIGEN[1:0] = “00”），将 SNGSTRT 置 1 会使计数器进行单触发计数。

要使能连续计数，必须将 CNTSTRT 位置 1。

若选择外部触发，则在 CNTSTRT 置 1 后到达的外部触发事件将启动计数器进行连续计数。任何后续的外部触发事件都将被丢弃，如*图 196* 所示。

若通过软件启动（TRIGEN[1:0] = “00”），将 CNTSTRT 置 1 会使计数器开始连续计数。

图 196. LPTIM 输出波形、连续计数模式配置



SNGSTRT 和 CNTSTRT 位只能在定时器使能时（ENABLE 位置 1）置 1。可以“实时的”从单触发模式切换为连续模式。

如果之前选择的是连续模式，则将 SNGSTRT 置 1 会使 LPTIM 切换为单触发模式。计数器（激活时）将在达到 ARR 后立即停止。

如果之前选择的是单触发模式，则将 CNTSTRT 置 1 会使 LPTIM 切换为连续模式。计数器（激活时）将在达到 ARR 后立即重新启动。

24.4.7 超时功能

若在一个选定的触发输入上检测到有效边沿，则可用于复位 LPTIM 计数器。该功能通过 TIMOUT 位进行控制。

第一个触发事件将启动定时器，任何后续的触发事件将复位计数器，且定时器将重新启动。

可实现低功耗超时功能。超时值对应于比较值；如果在预期的时间帧内未发生触发事件，MCU 将由比较匹配事件唤醒。

24.4.8 生成波形

两个 16 位寄存器，LPTIMx_ARR（自动重载寄存器）和 LPTIMx_CMP（比较寄存器）用于在 LPTIM 输出上生成多个不同的波形

定时器可生成以下波形：

- PWM 模式：若 LPTIMx_CMP 寄存器与 LPTIMx_CNT 寄存器匹配，则会立即将 LPTIM 输出置 1。若 LPTIMx_ARR 寄存器与 LPTIMx_CNT 寄存器匹配，则会立即将 LPTIM 输出复位。
- 单脉冲模式：对于第一个脉冲，输出波形与 PWM 模式输出波形类似，随后输出将永久复位。
- 置 1 一次模式：除输出保持最后一个信号电平外（取决于配置的输出极性），输出波形与单脉冲模式输出波形类似。

上述模式要求 LPTIMx_ARR 寄存器的值严格大于 LPTIMx_CMP 寄存器的值。

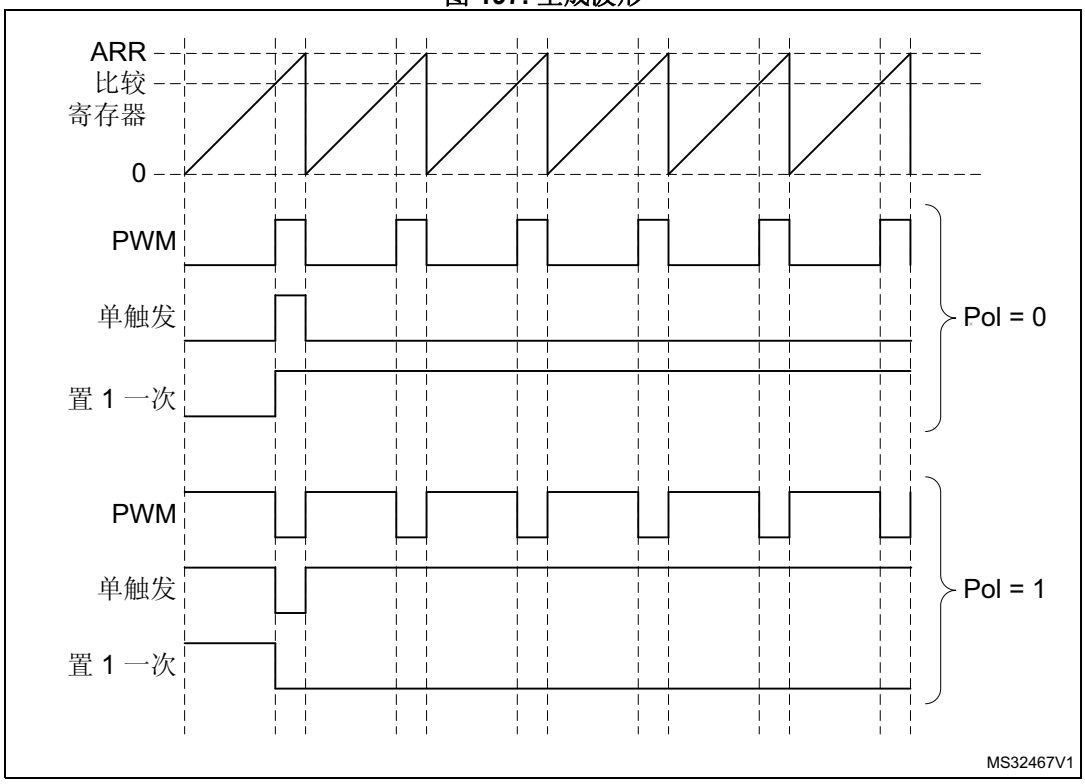
LPTIM 输出波形可通过 WAVE 位配置，具体如下：

- 若将 WAVE 位复位为 0，则会强制 LPTIM 生成 PWM 波形或单脉冲波形，具体取决于将哪个位（CNTSTRT 或 SNGSTRT）置 1。
- 若将 WAVE 位置 1，则会强制 LPTIM 生成置 1 一次模式波形。

WAVPOL 位控制 LPTIM 输出极性。更改立即生效，因此输出默认值将在极性重新配置后立即更改，甚至会在定时器使能前进行更改。

生成的信号的频率高达 LPTIM 时钟频率 2 分频。*图 197* 给出了可能在 LPTIM 输出上生成的三种波形。此外，此图还显示了通过 WAVPOL 位更改极性所产生的效果。

图 197. 生成波形



24.4.9 寄存器更新

LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器在 APB 总线写操作后会立即更新，若定时器已启动，也可在当前周期结束时更新。

PRELOAD 位控制 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器的更新方式：

- 当 PRELOAD 位复位为 0 时，LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器会在写访问后立即更新。
- 当 PRELOAD 位置 1 时，若定时器已启动，LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器会在当前周期结束时更新。

APB 总线和 LPTIM 使用的时钟不同，因此在 APB 写操作后，需要经过一定的延迟，写入值才能用于计数器比较器。在此延迟期间，必须避免向这些寄存器执行其他写操作。

LPTIMx_ISR 寄存器中的 ARROK 标志和 CMPOK 标志分别指示 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器的写操作已完成。

向 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器执行写操作后，只有在前一次写操作完成后，才能对同一寄存器执行新的写操作。在 ARROK 标志或 CMPOK 标志置 1 前执行连续的写操作将造成无法预知的结果。

24.4.10 计数器模式

LPTIM 计数器可用于对 LPTIM Input1 上的外部事件进行计数，也可用于对内部时钟周期进行计数。CKSEL 位和 COUNTMODE 位用于控制将使用哪些源更新计数器。

若使用 LPTIM 对 Input1 上的外部事件进行计数，计数器可在上升沿、下降沿或两种边沿进行更新，具体取决于写入 CKPOL[1:0] 位的值。

根据 CKSEL 和 COUNTMODE 值，可选择以下计数模式：

- CKSEL = 0：LPTIM 由内部时钟源提供时钟
  - COUNTMODE = 0

    当 LPTIM 由内部时钟源提供时钟，且 LPTIM 计数器根据在 LPTIM 外部 Input1 上检测到的有效边沿进行更新时，不得对提供给 LPTIM 的内部时钟进行预分频（PRESC[2:0] = “000”）。
  - COUNTMODE = 1

    LPTIM 外部 Input1 通过提供给 LPTIM 的内部时钟采样。因此，为了不丢失任何事件，外部 Input1 信号变化的频率决不应超过提供给 LPTIM 的内部时钟的频率。
- CKSEL = 1：LPTIM 由外部时钟源提供时钟

  COUNTMODE 值不相关。

  在这种配置下，LPTIM 无需内部时钟源（已使能干扰滤波器时除外）。注入到 LPTIM 外部 Input1 的信号用作 LPTIM 的系统时钟。此配置适合未使能任何内置振荡器的工作模式。

  对于这种配置，LPTIM 计数器可以在 input1 时钟信号的上升沿或下降沿进行更新，但不可在上升沿和下降沿均更新。

  由于注入到 LPTIM 外部 Input1 的信号也可用于 LPTIM 的时钟，计数器递增计数前存在一些初始延时（使能 PTIM 后）。更确切地说，LPTIM 外部 Input1 的前五个有效边沿将丢失（使能 PTIM 后）。

有关代码示例，请参见 *A.12.1：脉冲计数器配置代码示例*。

24.4.11 定时器使能

LPTIMx_CR 寄存器的 ENABLE 位用于使能/禁止 LPTIM。将 ENABLE 位置 1 后，需要延迟两个计数器时钟周期，才能真正使能 LPTIM。

LPTIMx_CFGR 和 LPTIMx_IER 寄存器必须在禁止 LPTIM 后才能修改。

24.4.12 编码器模式

此模式用于处理来自正交编码器的信号，此正交编码器用于检测旋转元件的角度位置。编码器接口模式就相当于带有方向选择的外部时钟。这意味着，计数器仅在 0 到 LPTIMx_ARR 寄存器中编程的自动重载值之间进行连续计数（根据具体方向，从 0 递增计数到 ARR，或从 ARR 递减计数到 0）。因此，在启动前必须先配置 LPTIMx_ARR。通过两个外部输入信号 Input1 和 Input2 生成时钟信号作为 LPTIM 计数器时钟。这两个信号间的相位确定计数方向。

仅当 LPTIM 由内部时钟源提供时钟时才可使用编码器模式。Input1 和 Input2 输入上的信号频率不得超过 LPTIM 内部时钟频率 4 分频。必须满足此条件才能确保 LPTIM 正常工作。

方向变化由 LPTIMx_ISR 寄存器中的两个递减和递增标志指示。此外，如果通过 LPTIMx_IER 寄存器使能，还可为两种方向变化事件产生中断。

要激活编码器模式，必须将 ENC 位置 1。LPTIM 必须首先配置为连续模式。

当编码器模式激活时，LPTIM 计数器按照增量编码器的速度和方向自动修改。因此，其内容始终代表编码器的位置。计数方向由递增和递减标志指示，对应于所连传感器的旋转方向。

根据使用 CKPOL[1:0] 位配置的边沿敏感性，可得几种不同的计数方案。下表汇总了可能的组合（假设 Input1 和 Input2 不同时切换）。

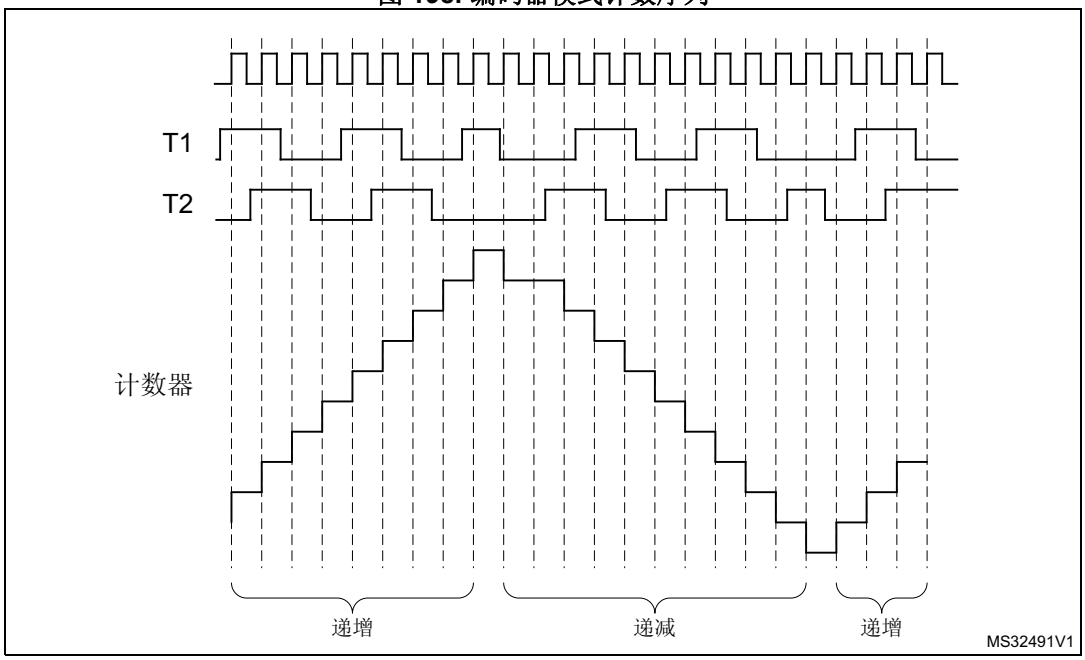
表 98. 编码器计数方案

有效边沿	相反信号的电平（Input1 对应 Input2，Input2 对应 Input1）	Input1 信号		Input2 信号	
		上升	下降	上升	下降
上升沿	高	递减	不计数	递增	不计数
	低	递增	不计数	递减	不计数
下降沿	高	不计数	递增	不计数	递减
	低	不计数	递减	不计数	递增
两种边沿	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

下图所示为编码器模式下配置了两种边沿敏感性的计数序列。

注意： 在此模式下，LPTIM 必须由内部时钟源提供时钟，因此 CKSEL 位必须保持其复位值 0。另外，预分频器分频比必须等于其复位值 1（PRESC[2:0] 位必须为“000”）。

图 198. 编码器模式计数序列



24.5 LPTIM 中断

若以下事件通过 LPTIMx_IER 寄存器使能，则这些事件会生成中断/唤醒事件：

- 与设定的数值比较后匹配
- 自动重载匹配（编码器模式下无论哪种方向）
- 外部触发事件
- 自动重载寄存器写操作完成
- 比较寄存器写操作完成
- 方向变化（编码器模式），可编程（递增/递减/同时递增和递减）。

注：只要 LPTIMx_IER 寄存器（中断使能寄存器）中的位在 LPTIMx_ISR 寄存器（状态寄存器）中相应标志置 1 后置 1，就不会触发中断。

24.6 LPTIM 寄存器

24.6.1 LPTIM 中断和状态寄存器 (LPTIMx_ISR)

LPTIM interrupt and status register

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
									r	r	r	r	r	r	r

位 31:7 保留，必须保持复位值。

位 6 **DOWN**：计数方向从递增变为递减 (Counter direction change up to down)
在编码器模式下，由硬件将 DOWN 位置 1 时，会通知应用计数方向由递增变为递减。

位 5 **UP**：计数方向从递减变为递增 (Counter direction change down to up)
在编码器模式下，由硬件将 UP 位置 1 时，会通知应用计数方向由递减变为递增。

位 4 **ARROK**：自动重载寄存器更新成功 (Autoreload register update OK)
由硬件将 ARROK 置 1 时，会通知应用 LPTIMx_ARR 寄存器的 APB 总线写操作已成功完成。如果这样，便可启动新的写操作。

位 3 **CMPOK**：比较寄存器更新成功 (Compare register update OK)
由硬件将 CMPOK 置 1 时，会通知应用 LPTIMx_CMP 寄存器的 APB 总线写操作已成功完成。如果这样，便可启动新的写操作。

位 2 **EXTTRIG**：外部触发边沿事件 (External trigger edge event)
由硬件将 EXTTRIG 置 1 时，会通知应用所选的外部触发输入上产生有效边沿。如果由于定时器已启动而忽略触发事件，则不会将此标志置 1。

位 1 **ARRM**：自动重载匹配 (Autoreload match)
由硬件将 ARRM 置 1 时，会通知应用 LPTIMx_CNT 寄存器的值已达到 LPTIMx_ARR 寄存器的值。

位 0 **CMPM**：比较匹配 (Compare match)
由硬件将 CMPM 位置 1 时，会通知应用 LPTIMx_CNT 寄存器的值已达到 LPTIMx_CMP 寄存器的值。

24.6.2 LPTIM 中断清零寄存器 (LPTIMx_ICR)

LPTIM interrupt clear register

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
									w	w	w	w	w	w	w

位 31:7 保留，必须保持复位值。

位 6 **DOWNCF**：方向变为递减清零标志 (Direction change to down Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 DOWN 标志将清零

位 5 **UPCF**：方向变为递增清零标志 (Direction change to UP Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 UP 标志将清零

位 4 **ARROKCF**：自动重载寄存器更新成功清零标志 (Autoreload register update OK Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 ARROK 标志将清零

位 3 **CMPOKCF**：比较寄存器更新成功清零标志 (Compare register update OK Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 CMPOK 标志将清零

位 2 **EXTTRIGCF**：外部触发有效边沿清零标志 (External trigger valid edge Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 EXTTRIG 标志将清零

位 1 **ARRMCF**：自动重载匹配清零标志 (Autoreload match Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 ARRM 标志将清零

位 0 **CMPMCF**：比较匹配清零标志 (compare match Clear Flag)
将 1 写入此位时，LPT_ISR 寄存器中的 CMP 标志将清零

24.6.3 LPTIM 中断使能寄存器 (LPTIMx_IER)

LPTIM interrupt enable register

偏移地址：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
									rw	rw	rw	rw	rw	rw	rw

位 31:7 保留，必须保持复位值。

位 6 **DOWNIE**：方向变为递减中断使能 (Direction change to down Interrupt Enable)
- 0：禁止 DOWN 中断
- 1：使能 DOWN 中断

位 5 **UPIE**：方向变为递增中断使能 (Direction change to UP Interrupt Enable)
- 0：禁止 UP 中断
- 1：使能 UP 中断

位 4 **ARROKIE**：自动重载寄存器更新成功中断使能 (Autoreload register update OK Interrupt Enable)
- 0：禁止 ARROK 中断
- 1：使能 ARROK 中断

位 3 **CMPOKIE**：比较寄存器更新成功中断使能 (Compare register update OK Interrupt Enable)
- 0：禁止 CMPOK 中断
- 1：使能 CMPOK 中断

位 2 **EXTTRIGIE**：外部触发有效边沿中断使能 (External trigger valid edge Interrupt Enable)
- 0：禁止 EXTTRIG 中断
- 1：使能 EXTTRIG 中断

位 1 **ARRMIE**：自动重载匹配中断使能 (Autoreload match Interrupt Enable)
- 0：禁止 ARRM 中断
- 1：使能 ARRM 中断

位 0 **CMPMIE**：比较匹配中断使能 (Compare match Interrupt Enable)
- 0：禁止 CMPM 中断
- 1：使能 CMPM 中断

注意： 必须在 LPTIM 禁止已时（ENABLE 位复位为 0）才能修改 LPTIMx_IER 寄存器。

24.6.4 LPTIM 配置寄存器 (LPTIMx_CFGR)

LPTIM configuration register

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

位 31:25 保留，必须保持复位值。

位 24 **ENC**：编码器模式使能 (Encoder mode enable)

ENC 位控制编码器模式

0：禁止编码器模式

1：使能编码器模式

位 23 **COUNTMODE**：计数器模式使能 (counter mode enabled)

COUNTMODE 位用于选择 LPTIM 使用哪个时钟源来为计数器提供时钟：

0：计数器在每个内部时钟脉冲后递增

1：计数器在 LPTIM 外部 Input1 上的每个有效时钟脉冲后递增

位 22 **PRELOAD**：寄存器更新模式 (Registers update mode)

PRELOAD 位控制 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器的更新方式

0：寄存器在每次 APB 总线写访问后更新

1：寄存器在当前 LPTIM 周期结束时更新

位 21 **WAVPOL**：波形极性 (Waveform shape polarity)

WAVEPOL 位控制输出极性

0：LPTIM 输出反映 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器之间的比较结果。

1：LPTIM 输出反映与 LPTIMx_ARR 寄存器和 LPTIMx_CMP 寄存器之间的比较结果相反的值

位 20 **WAVE**：波形 (Waveform shape)

WAVE 位控制输出波形

0： 停用置 1 一次模式和 PWM/ 单脉冲波形（具体取决于 OPMODE 位）

1： 激活置 1 一次模式

位 19 **TIMOUT**：超时使能 (Timeout enable)

TIMOUT 位控制超时功能

0：定时器已启动时到达的触发事件将被忽略

1：定时器已启动时到达的触发事件将复位并重新启动计数器

位 18:17 **TRIGEN**：触发使能和极性 (Trigger enable and polarity)

TRIGEN 位控制 LPTIM 计数器是否由外部触发信号启动。如果已选择由外部触发信号启动，触发有效边沿的配置有以下三种：

00：软件触发（由软件启动计数）

01：上升沿为有效边沿

10：下降沿为有效边沿

11：上升沿和下降沿均为有效边沿

位 16 保留，必须保持复位值。

位 15:13 **TRIGSEL**：触发源选择器 (Trigger selector)

TRIGSEL 位用于选择作为 LPTIM 触发事件的触发源，可用触发源包括以下 8 种：

000: ext_trig0
001: ext_trig1
010: ext_trig2
011: ext_trig3
100: ext_trig4
101: ext_trig5
110: ext_trig6
111: ext_trig7

有关各定时器的 ITRx 含义的更多详细信息，请参见*第 575 页的表 99：LPTIM 外部触发连接*。

位 12 保留，必须保持复位值。

位 11:9 **PRESC**：时钟预分频器 (Clock prescaler)

PRESC 位配置预分频器的分频系数。分频系数可从以下分频系数中选择：

000:/1
001:/2
010:/4
011:/8
100:/16
101:/32
110:/64
111:/128

位 8 保留，必须保持复位值。

位 7:6 **TRGFLT**：触发信号的可配置数字滤波器 (Configurable digital filter for trigger)

TRGFLT 值用于设置连续相同采样的数量，若在内部触发信号电平发生变化时检测到此类连续采样，才会将此电平变化视为有效电平切换。必须存在内部时钟源才能使用此功能

00：任何触发信号有效电平变化均视为有效触发
01：触发信号有效电平变化必须至少稳定 2 个时钟周期，才能将其视为有效触发。
10：触发信号有效电平变化必须至少稳定 4 个时钟周期，才能将其视为有效触发。
11：触发信号有效电平变化必须至少稳定 8 个时钟周期，才能将其视为有效触发。

位 5 保留，必须保持复位值。

位 4:3 **CKFLT**：外部时钟的可配置数字滤波器 (Configurable digital filter for external clock)

CKFLT 值用于设置连续相同采样的数量，若在外部时钟信号电平发生变化时检测到此类连续采样，才会将此电平变化视为有效电平切换。必须存在内部时钟源才能使用此功能

00：任何外部时钟信号电平变化均视为有效切换
01：外部时钟信号电平变化必须至少稳定 2 个时钟周期，才能将其视为有效切换。
10：外部时钟信号电平变化必须至少稳定 4 个时钟周期，才能将其视为有效切换。
11：外部时钟信号电平变化必须至少稳定 8 个时钟周期，才能将其视为有效切换。

位 2:1 **CKPOL**：时钟极性 (Clock Polarity)

如果 LPTIM 由外部时钟源提供时钟：

当 LPTIM 由外部时钟源提供时钟时，CKPOL 位用于配置计数所使用的有效边沿：

00：上升沿为用于计数的有效边沿
01：下降沿为用于计数的有效边沿
10：上升沿和下降沿均为有效边沿 当外部时钟信号的上升沿和下降沿均视为有效边沿时，LPTIM 还必须由内部时钟源提供时钟，且内部时钟源频率至少等于外部时钟频率的四倍。
11：不允许

如果将 LPTIM 配置为编码器模式（ENC 位置 1）：

00：编码器子模式 1 激活
01：编码器子模式 2 激活
10：编码器子模式 3 激活

有关编码器模式子模式的更多详细信息，请参见*第 24.4.12 节：编码器模式*。

位 0 **CKSEL**：时钟选择器 (Clock selector)

CKSEL 位选择 LPTIM 将使用的时钟源：

0：LPTIM 由内部时钟源（APB 时钟或任意内置振荡器）提供时钟
1：LPTIM 由外部时钟源通过 LPTIM 外部 Input1 提供时钟

注意： 必须在 LPTIM 已禁止时（ENABLE 位复位为 0）才能修改 LPTIMx_CFGR 寄存器

表 99. LPTIM 外部触发连接

TRIGSEL	外部触发信号
ext_trig0	PB6 或 PC3
ext_trig1	RTC 闹钟 A
ext_trig2	RTC 闹钟 B
ext_trig3	RTC_TAMP1 输入检测
ext_trig4	RTC_TAMP2 输入检测
ext_trig5	RTC_TAMP3 输入检测
ext_trig6	COMP1_OUT
ext_trig7	COMP2_OUT

24.6.5 LPTIM 控制寄存器 (LPTIMx_CR)

LPTIM control register

偏移地址：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT STRT	SNG STRT	ENA BLE
													rw	rw	rw

位 31:3 保留，必须保持复位值。

位 2 **CNTSTRT**：定时器以连续模式启动 (Timer start in continuous mode)

此位通过软件置 1，通过硬件清零。

若通过软件启动（TRIGEN[1:0] =“00”），将此位置 1 会使 LPTIM 以连续模式启动。

如果禁止软件启动（TRIGEN[1:0] 不等于“00”），将此位置 1 会使定时器在检测到外部触发信号后立即以连续模式启动。

如果在进行单脉冲模式计数时将此位置 1，则在 LPTIMx_ARR 寄存器和 LPTIMx_CNT 寄存器下一次匹配时定时器不会停止，LPTIM 计数器将继续以连续模式计数。

只有在使能 LPTIM 时才能将此位置 1。此位由硬件自动复位。

位 1 **SNGSTRT**：LPTIM 以单次模式启动 (LPTIM start in Single mode)

此位通过软件置 1，通过硬件清零。

若通过软件启动（TRIGEN[1:0] =“00”），将此位置 1 会使 LPTIM 以单脉冲模式启动。

如果禁止软件启动（TRIGEN[1:0] 不等于“00”），将此位置 1 会使 LPTIM 在检测到外部触发信号后立即以单脉冲模式启动。

如果在 LPTIM 处于连续计数模式时将此位置 1，LPTIM 将在 LPTIMx_ARR 寄存器和 LPTIMx_CNT 寄存器下一次匹配时停止。

只有在使能 LPTIM 时才能将此位置 1。此位由硬件自动复位。

位 0 **ENABLE**：LPTIM 使能 (LPTIM enable)

ENABLE 位由软件置 1 和清零。

0：禁止 LPTIM

1：使能 LPTIM

24.6.6 LPTIM 比较寄存器 (LPTIMx_CMP)

LPTIM compare register

偏移地址：0x14

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **CMP**：比较值 (Compare value)

CMP 为 LPTIM 所使用的比较值。

必须在 LPTIM 已使能时（ENABLE 位置 1）才能修改 LPTIMx_CMP 寄存器的内容。

24.6.7 LPTIM 自动重载寄存器 (LPTIMx_ARR)

LPTIM autoreload register

偏移地址：0x18

复位值：0x0000 0001

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **ARR**：自动重载值 (Auto reload value)

ARR 为 LPTIM 的自动重载值。

此值必须严格大于 CMP[15:0] 的值。

必须在 LPTIM 已使能时（ENABLE 位置 1）才能修改 LPTIMx_ARR 寄存器的内容。

24.6.8 LPTIM 计数器寄存器 (LPTIMx_CNT)

LPTIM counter register

偏移地址：0x1C

复位值：0x0000 0000

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **CNT**：计数器值 (Counter value)

当 LPTIM 通过异步时钟运行时，读取 LPTIMx_CNT 寄存器会返回不可靠的值。因此在这种情况下，必须连续执行读访问两次，并验证两次返回的值是否相同。

应注意的是，为了实现可靠的 LPTIM_CNT 寄存器读访问，必须执行两次连续的读访问并进行比较。当两次连续读访问的值相等时，可认为读访问可靠。

24.6.9 LPTIM 寄存器映射

下表对 LPTIM 寄存器进行了汇总。

表 100. LPTIM 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPTIMx_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																										0	0	0	0	0	0	0
0x04	LPTIMx_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF	UPCF	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																										0	0	0	0	0	0	0
0x08	LPTIMx_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE	UPIE	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																										0	0	0	0	0	0	0
0x0C	LPTIMx_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNTMODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0	0	0	0	0	0
0x10	LPTIMx_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTSTRT	SNGSTRT	ENABLE
	Reset value																														0	0	0
0x14	LPTIMx_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	LPTIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	LPTIMx_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

25 独立看门狗 (IWDG)

25.1 简介

此器件具有一个嵌入式看门狗外设，具有安全性高、定时准确及使用灵活的优点。此独立看门狗外设可用于检测并解决由软件错误导致的故障，并在计数器达到给定的超时值时触发系统复位。

独立看门狗 (IWDG) 由其专用低速时钟 (LSI) 驱动，因此即便在主时钟发生故障时仍然保持工作状态。

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。有关窗口看门狗的详细信息，请参见*第 587 页的第 26 节*。

25.2 IWDG 主要特性

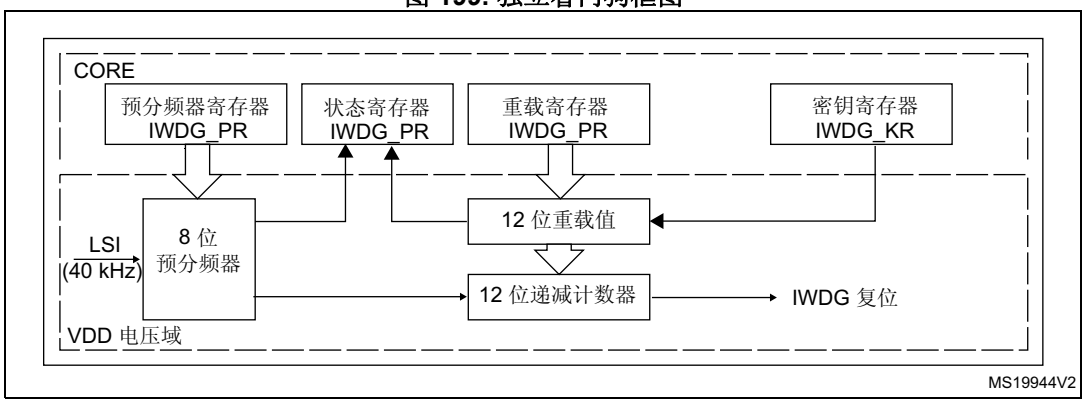
- 自由运行递减计数器
- 时钟由独立 RC 振荡器提供（可在待机和停止模式下运行）
- 复位条件
  - 当递减计数器值小于 0x000 时复位（如果看门狗已激活）
  - 在窗口之外重载递减计数器时复位（如果看门狗已激活）

25.3 IWDG 功能说明

25.3.1 IWDG 框图

图 199 给出了独立看门狗模块的功能框图。

图 199. 独立看门狗框图



注： *看门狗功能由 CORE 电压域供电，在停止模式和待机模式下仍能工作。*

通过向键寄存器 (IWDG_KR) 中写入值 0x0000 CCCC 来启动独立看门狗时，计数器开始从复位值 0xFFF 递减计数。当计数器计数到终值 (0x000) 时会产生一个复位信号（IWDG 复位）。

任何时候将键值 0x0000 AAAA 写到 IWDG_KR 寄存器中，IWDG_RLR 的值就会被重载到计数器，从而避免产生看门狗复位。

25.3.2 窗口选项

通过在 IWDG_WINR 寄存器中设置合适的窗口，IWDG 也可以用作窗口看门狗。

当计数器值大于窗口寄存器 (IWDG_WINR) 中存储的值时，如果执行重载操作，则会产生复位。

IWDG_WINR 的默认值为 0x0000 0FFF，因此，如果不更新此默认值，将禁止窗口选项。

窗口值一经更改，便执行重载操作，以便将递减计数器复位为 IWDG_RLR 值，并方便计算周期数以生成下一次重载。

使能窗口选项时配置 IWDG

1. 通过在 IWDG_KR 寄存器中写入 0x0000 CCCC 来使能 IWDG。
2. 通过在 IWDG_KR 寄存器中写入 0x0000 5555 来使能寄存器访问。
3. 通过将 IWDG_PR 编程为 0~7 中的数值来配置 IWDG 预分频器。
4. 写重载寄存器 (IWDG_RLR)。
5. 等待寄存器更新 (IWDG_SR = 0x0000 0000)。
6. 写窗口寄存器 IWDG_WINR，将自动刷新计数器值为 IWDG_RLR 的值。

注：当 IWDG_SR 设置为 0x0000 0000 时，写入窗口值允许刷新计数器值为 RLR 的值。

有关代码示例，请参见 *A.13.2：IWDG 配置（带窗口）代码示例*。

禁止窗口选项时配置 IWDG

不使用窗口选项时，可按以下步骤配置 IWDG：

1. 通过在 IWDG_KR 寄存器中写入 0x0000 CCCC 来使能 IWDG。
2. 通过在 IWDG_KR 寄存器中写入 0x0000 5555 来使能寄存器访问。
3. 通过将 IWDG_PR 编程为 0~7 中的数值来配置 IWDG 预分频器。
4. 写重载寄存器 (IWDG_RLR)。
5. 等待寄存器更新 (IWDG_SR = 0x0000 0000)。
6. 刷新计数器值为 IWDG_RLR 的值 (IWDG_KR = 0x0000 AAAA)

有关代码示例，请参见 *A.13.1：IWDG 配置代码示例*。

25.3.3 硬件看门狗

如果通过器件选项位使能“硬件看门狗”功能，上电时将自动使能看门狗；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，或者在窗口内部重载了递减计数器，则系统会产生复位。

25.3.4 停止和待机模式下的行为

一旦运行，IWDG 便无法停止。

25.3.5 寄存器访问保护

IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器具有写访问保护。若要修改寄存器，必须首先对 IWDG_KR 寄存器写入代码 0x0000 5555。而写入其他值则会破坏该序列，从而使寄存器访问保护再次生效。这意味着重装载操作（即写入 0x0000 AAAA）也会启动写保护功能。
状态寄存器指示预分频值、递减计数器重载值或窗口值是否正在被更新。

25.3.6 调试模式

当微控制器进入调试模式时（内核停止），IWDG 计数器会根据 DBG 模块中的 DBG_IWDG_STOP 配置位选择继续正常工作或者停止工作。

25.4 IWDG 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

25.4.1 键寄存器 (IWDG_KR)

Key register

偏移地址：0x00

复位值：0x0000 0000（待机模式时复位）

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

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

位 31:16 保留，必须保持复位值。

位 15:0 **KEY[15:0]：**键值 (Key value)（只能写，读为 0x0000）

必须每隔一段时间便通过软件对这些位写入键值 0xAAAA，否则当计数器计数到 0 时，看门狗会产生复位。

写入键值 0x5555 可使能对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的访问（请参见*第 25.3.5 节：寄存器访问保护*）。

写入键值 CCCCh 可启动看门狗（选中硬件看门狗选项的情况除外）。

25.4.2 预分频器寄存器 (IWDG_PR)

Prescaler register

偏移地址：0x04

复位值：0x0000 0000

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

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

位 31:3 保留，必须保持复位值。

位 2:0 **PR[2:0]**：预分频系数 (Prescaler divider)

这些位受写访问保护，请参见*第 25.3.5 节：寄存器访问保护*。通过软件设置这些位来选择计数器时钟的预分频因子。若要更改预分频器的分频系数，IWDG_SR 的 PVU 位必须为 0。

000：4 分频
001：8 分频
010：16 分频
011：32 分频
100：64 分频
101：128 分频
110：256 分频
111：256 分频

注：读取该寄存器会返回 VDD 电压域的预分频器值。如果正在对该寄存器执行写操作，则读取的值可能不是最新的/有效的。因此，只有在 IWDG_SR 寄存器中的 PVU 位为 0 时，从寄存器读取的值才有效。

25.4.3 重载寄存器 (IWDG_RLR)

Reload register

偏移地址：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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

位 31:12 保留，必须保持复位值。

位 11:0 **RL[11:0]**：看门狗计数器重载值 (Watchdog counter reload value)

这些位受写访问保护，请参见*第 25.3.5 节*。这个值由软件设置，每次对 IWDG_KR 寄存器写入值 0xAAAA 时，这个值就会重装载到看门狗计数器中。之后，看门狗计数器便从该装载的值开始递减计数。超时周期由该值和时钟预分频器共同决定。有关超时信息，请参见数据手册。

若要更改重载值，IWDG_SR 中的 RVU 位必须为 0。

注：读取该寄存器会返回 VDD 电压域的重载值。如果正在对该寄存器执行写操作，则读取的值可能不是最新的/有效的。因此，只有在 IWDG_SR 寄存器中的 RVU 位为 0 时，从寄存器读取的值才有效。

25.4.4 状态寄存器 (IWDG_SR)

Status register

偏移地址：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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
													r	r	r

位 31:3 保留，必须保持复位值。

位 2 **WVU**：看门狗计数器窗口值更新 (Watchdog counter window value update)
该位由硬件置 1 以指示窗口值正在更新。当在 V_{DD} 电压域下完成重载值更新操作后（需要多达 5 个 RC 40 kHz 周期），会通过硬件将该位复位。
窗口值只有在 WVU 位为 0 时才可更新。
此位只有在通用“窗口”= 1 时才生成

位 1 **RVU**：看门狗计数器重载值更新 (Watchdog counter reload value update)
该位由硬件置 1 以指示重载值正在更新。当在 V_{DD} 电压域下完成重载值更新操作后（需要多达 5 个 RC 40 kHz 周期），会通过硬件将该位复位。
重载值只有在 RVU 位为 0 时才可更新。

位 0 **PVU**：看门狗预分频器值更新 (Watchdog prescaler value update)
该位由硬件置 1 以指示预分频器值正在更新。当在 V_{DD} 电压域下完成预分频器值更新操作后（需要多达 5 个 RC 40 kHz 周期），会通过硬件将该位复位。
预分频器值只有在 PVU 位为 0 时才可更新。

注：如果应用使用多个重载值、预分频器值或窗口值，则必须等到 RVU 位被复位后才能更改重载值，等到 PVU 位被复位后才能更改预分频器值，而且必须等到 WVU 位被复位后才能更改窗口值。但是，在更新预分频器和/或重载/窗口值之后，则无需等到 RVU、PVU 或 WVU 复位后再继续执行代码（进入低功耗模式时除外）。

25.4.5 窗口寄存器 (IWDG_WINR)

Window register

偏移地址：0x10

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

位 31:12 保留，必须保持复位值。

位 11:0 **WIN[11:0]**：看门狗计数器窗口值 (Watchdog counter window value)

这些位受写访问保护，请参见*第 25.3.5 节*。这些位包含用于与递减计数器进行比较的窗口值上限。

为防止发生复位，当递减计数器的值低于窗口寄存器值

且大于 0x0 时必须重载

若要更改重载值，IWDG_SR 中的 WVU 位必须为 0。

注：读取该寄存器会返回 V_{DD} 电压域的重载值。如果正在对该寄存器执行写操作，则读取的值可能无效。因此，只有在 IWDG_SR 寄存器中的 WVU 位为 0 时，从寄存器读取的值才有效。

25.4.6 IWDG 寄存器映射

下表提供了 IWDG 寄存器映射和复位值。

表 101. IWDG 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WVU	RVU	PVU
	Reset value																														0	0	0
0x10	**IWDG_WINR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1

有关寄存器边界地址的信息，请参见*第 2.2.2 节：存储器映射和寄存器边界地址*。

26 系统窗口看门狗 (WWDG)

26.1 简介

系统窗口看门狗 (WWDG) 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。除非程序在 T6 位变成 0 前刷新递减计数器的值，否则看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。如果在递减计数器达到窗口寄存器值之前刷新控制寄存器中的 7 位递减计数器值，也会产生 MCU 复位。这意味着必须在限定的时间窗口内刷新计数器。

WWDG 时钟由 APB1 时钟经预分频后提供，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。

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。

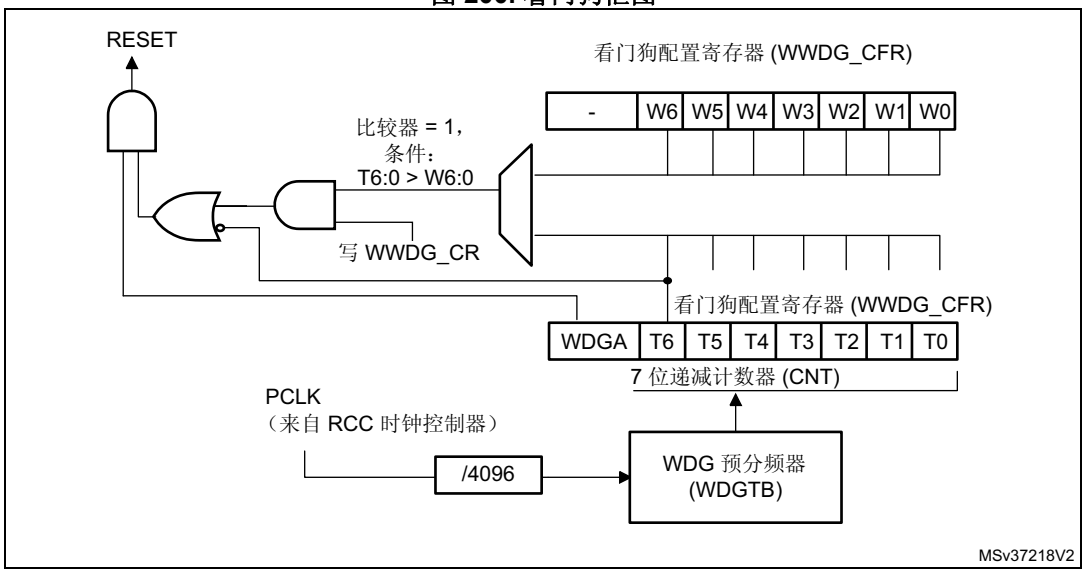
26.2 WWDG 主要特性

- 可编程的自由运行递减计数器
- 复位条件
  - 当递减计数器值小于 0x40 时复位（如果看门狗已激活）
  - 在窗口之外重载递减计数器时复位（如果看门狗已激活）（请参见*图 201*）
- 提前唤醒中断 (EWI)：当递减计数器等于 0x40 时触发（如果已使能且看门狗已激活）

26.3 WWDG 功能说明

如果激活看门狗（WWDG_CR 寄存器中的 WDGA 位置 1），则当 7 位递减计数器（T[6:0] 位）从 0x40 递减到 0x3F（T6 已清零）时会引发复位。当计数器值大于窗口寄存器中所存储的值时，如果软件重载计数器，则会产生复位。

图 200. 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。只有当计数器值低于窗口寄存器值且高于 0x3F 时，才能执行此操作。要存储到 WWDG_CR 寄存器中的值必须介于 0xFF 和 0xC0 之间。

26.3.1 使能看门狗

在系统复位后，看门狗总是处于关闭状态。可通过设置 WWDG_CR 寄存器中的 WDGA 位来使能看门狗，之后除非执行复位操作，否则不能再次关闭。

26.3.2 控制递减计数器

递减计数器处于自由运行状态，即使禁止看门狗，递减计数器仍继续递减计数。当使能看门狗时，必须将 T6 位置 1，以防止立即复位。

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，预分频器的状态是未知的（请参见*图 201*）。配置寄存器 (WWDG_CFR) 包含窗口的上限：为防止发生复位，当递减计数器的值低于窗口寄存器值且大于 0x3F 时必须重载。*图 201* 介绍了窗口看门狗的工作过程。

注： 可使用 T6 位产生软件复位（将 WDGA 位置 1 并将 T6 位清零）。

26.3.3 看门狗中断高级特性

如果在产生实际复位之前必须执行特定的安全操作或数据记录，则可使用提前唤醒中断 (EWI)。通过设置 WWDG_CFR 寄存器中的 EWI 位使能 EWI 中断。当递减计数器的值为 0x40 时，将生成 EWI 中断。在复位器件之前，可以使用相应的中断服务程序 (ISR) 来触发特定操作（例如通信或数据记录）。

在某些应用中，可以使用 EWI 中断来管理软件系统检查和/或系统恢复/功能退化，而不会生成 WWDG 复位。在这种情况下，相应的中断服务程序 (ISR) 可用来重载 WWDG 计数器以避免 WWDG 复位，然后再触发所需操作。

通过将 0 写入 WWDG_SR 寄存器中的 EWIF 位来清除 EWI 中断。

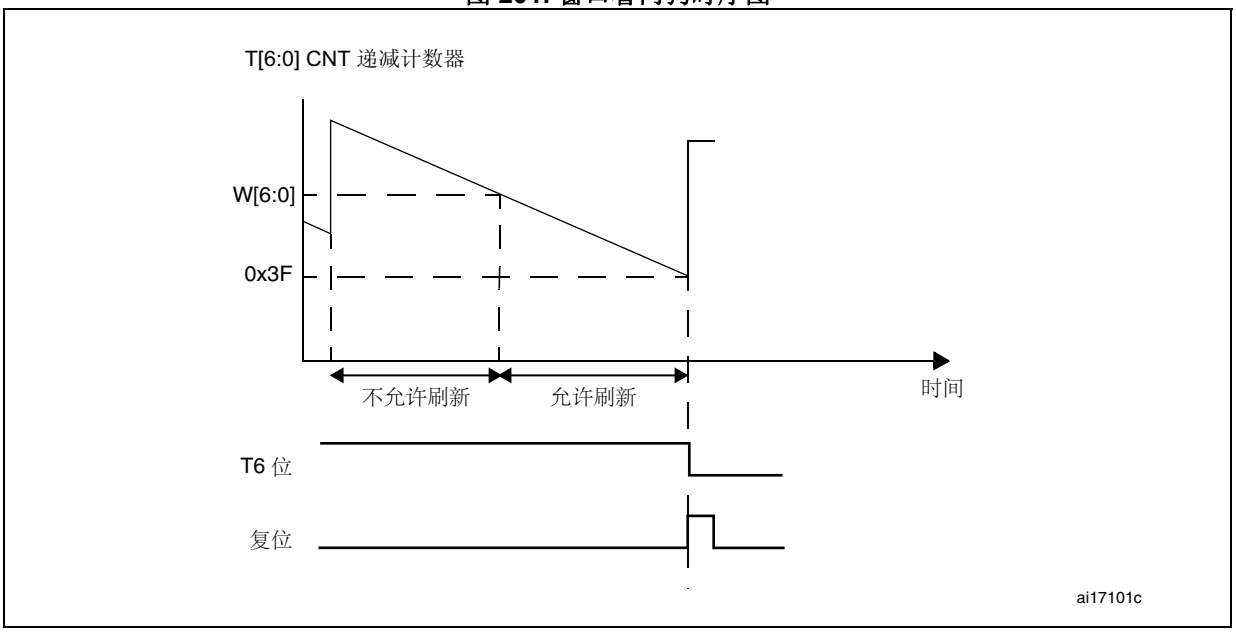
注： 当由于在更高优先级任务中有系统锁定而无法使用 EWI 中断时，最终会产生 WWDG 复位。

26.3.4 如何设置看门狗超时

可以使用*图 201* 中的公式来计算 WWDG 超时。

警告： 写入 **WWDG_CR** 寄存器时，始终将 **1** 写入 **T6** 位，以避免生成立即复位。

图 201. 窗口看门狗时序图



超时值的计算公式如下：

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \qquad (ms)$$

其中：

t_{WWDG}：WWDG 超时

t_{PCLK}：APB1 时钟周期，以 ms 为测量单位

4096：对应于内部分频器的值

例如，假设 APB1 频率等于 32 MHz，将 WDGTB[1:0] 设置为 3，将 T[5:0] 设置为 63：

$$t_{WWDG} = 1/32000 \times 4096 \times 2^{3} \times (63 + 1) = 65.54 \text{ ms}$$

有关 t_{WWDG} 的最小值和最大值，请参见数据手册。

有关代码示例，请参见 *A.14.1：WWDG 配置代码示例*。

26.3.5 调试模式

当微控制器进入调试模式（Cortex®-**M0+** 内核停止）时，WWDG 计数器会根据 DBG 模块中的 DBG_WWDG_STOP 配置位选择继续正常工作或者停止工作。有关详细信息，请参见*第 33.9.2 节：对定时器、看门狗和 I^2C 的调试支持*。

26.4 WWDG 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

26.4.1 控制寄存器 (WWDG_CR)

Control register

偏移地址：0x00

复位值：0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw						

位 31:8 保留，必须保持复位值。

位 7 **WDGA**：激活位 (Activation bit)
此位由软件置 1，只有复位后才由硬件清零。当 WDGA = 1 时，看门狗可产生复位。
0：禁止看门狗
1：使能看门狗

位 6:0 **T[6:0]**：7 位计数器（MSB 到 LSB）
这些位用来存储看门狗计数器的值。它每隔 (4096 x $2^{WDGTB[1:0]}$) PCLK 个周期递减一次。当它从 0x40 递减到 0x3F（T6 清零）时会产生复位。

26.4.2 配置寄存器 (WWDG_CFR)

Configuration register

偏移地址：0x04

复位值：0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		W[6:0]						
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:10 保留，必须保持复位值。

位 9 **EWI**：提前唤醒中断 (Early wakeup interrupt)
置 1 后，只要计数器值达到 0x40 就会产生中断。此中断只有在复位后才由硬件清零。

位 8:7 **WDGTB[1:0]**：定时器时基 (Timer base)
可按如下方式修改预分频器的时基：
00：CK 计数器时钟 (PCLK div 4096) 分频器 1
01：CK 计数器时钟 (PCLK div 4096) 分频器 2
10：CK 计数器时钟 (PCLK div 4096) 分频器 4
11：CK 计数器时钟 (PCLK div 4096) 分频器 8

位 6:0 **W[6:0]**：7 位窗口值 (7-bit window value)
这些位包含用于与递减计数器进行比较的窗口值。

26.4.3 状态寄存器 (WWDG_SR)

Status register

偏移地址：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

位 31:1 保留，必须保持复位值。

位 0 **EWIF**：提前唤醒中断标志 (Early wakeup interrupt flag)
当计数器值达到 0x40 时此位由硬件置 1。它必须由软件通过写入 0 来清零。写入 1 不起作用。如果不使能中断，此位也会被置 1。

26.4.4 WWDG 寄存器映射

下表提供了 WWDG 寄存器映射和复位值。

表 102. WWDG 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**WWDG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

有关寄存器边界地址的信息，请参见*第 2.2.2 节：存储器映射和寄存器边界地址*。

27 实时时钟 (RTC)

27.1 简介

实时时钟 (RTC) 提供用于管理所有低功耗模式的自动唤醒单元。

实时时钟 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供具有可编程闹钟中断功能的日历时钟/日历。

RTC 还包含具有中断功能的周期性可编程唤醒标志。

两个 32 位寄存器包含二进码十进数格式 (BCD) 的秒、分钟、小时（12 或 24 小时制）、星期几、日期、月份和年份。此外，还可提供二进制格式的亚秒值。

系统可以自动将月份的天数补偿为 28、29（闰年）、30 和 31 天。并且还可以进行夏令时补偿。

其它 32 位寄存器还包含可编程的闹钟亚秒、秒、分钟、小时、星期几和日期。

此外，还可以使用数字校准功能对晶振精度的偏差进行补偿。

RTC域复位后，所有 RTC 寄存器都会受到保护，以防止可能的非正常写访问。

无论器件状态如何（运行模式、低功耗模式或处于复位状态），只要电源电压保持在工作范围内，RTC 便不会停止工作。

27.2 RTC 主要特性

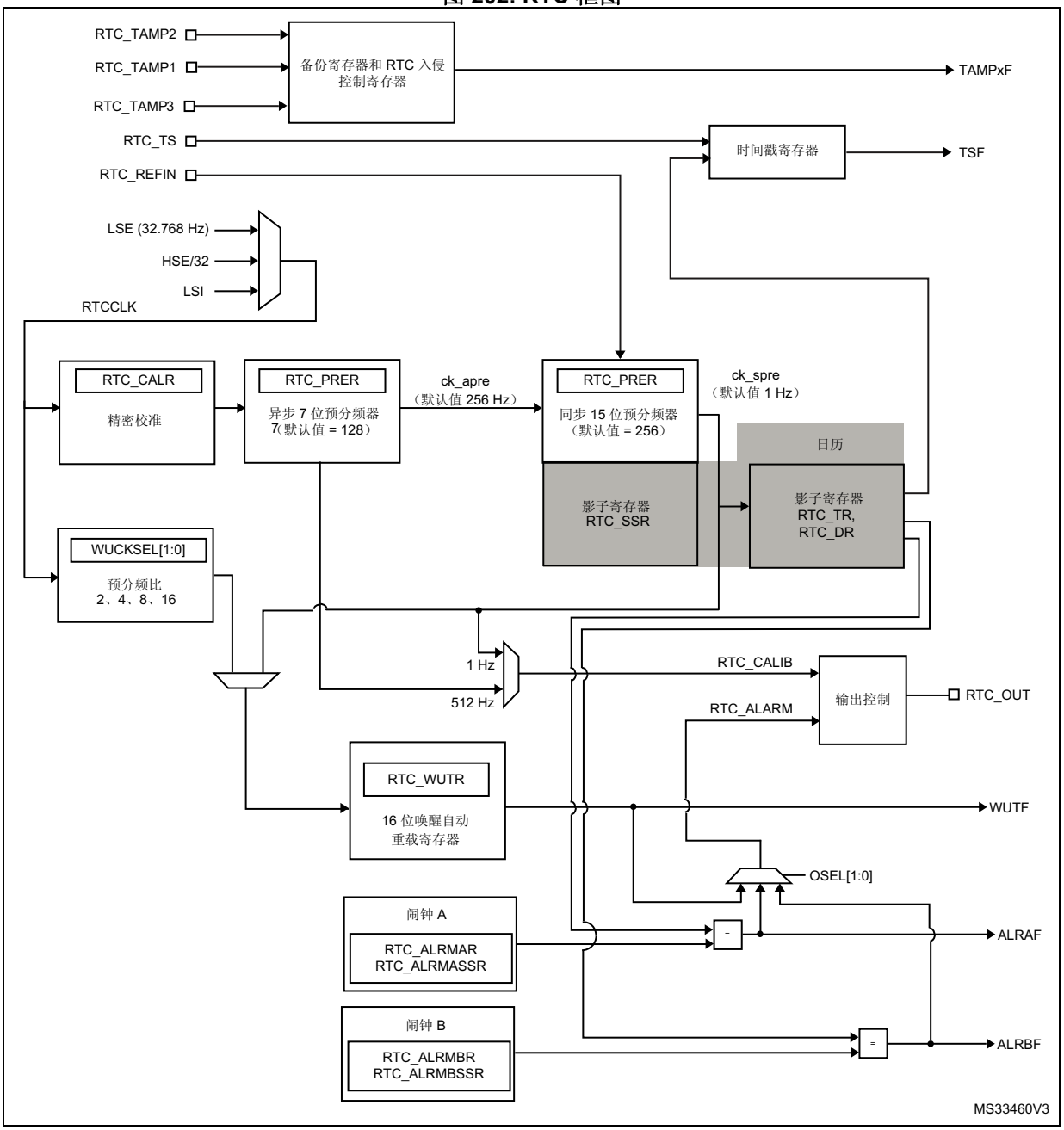
RTC 单元的主要特性如下（参见*图 202：RTC 框图*）：

- 包含亚秒、秒、分钟、小时（12/24 小时制）、星期几、日期、月份和年份的日历。
- 软件可编程的夏令时补偿。
- 具有中断功能的可编程闹钟。可通过任意日历字段的组合触发闹钟。
- 自动唤醒单元，可周期性地生成标志以触发自动唤醒中断。
- 参考时钟检测：可使用更加精确的第二时钟源（50 Hz 或 60 Hz）来提高日历的精确度。
- 利用亚秒级移位特性与外部时钟实现精确同步。
- 数字校准电路（周期性计数器调整）：精度为 0.95 ppm，在数秒钟的校准窗口中获得
- 用于事件保存的时间戳功能
- 带可配置过滤器和内部上拉的入侵检测事件
- 可屏蔽中断/事件：
  - 闹钟 A
  - 闹钟 B
  - 唤醒中断 (Wakeup interrupt)
  - 时间戳
  - 入侵检测
- 5 个备份寄存器。

27.3 RTC 功能说明

27.3.1 RTC 框图

图 202. RTC 框图



1. 仅第 5 类器件具备 RTC_TAMP3。

RTC 包含：

- 两个闹钟
- 最多三个入侵事件
- 5 个 32 位备份寄存器
- 复用功能输出 RTC_OUT，可选择以下两个输出之一：
  - RTC_CALIB：512 Hz 或 1 Hz 时钟输出（LSE 频率为 32.768 kHz）。可通过将 RTC_CR 寄存器中的 COE 位置 1 来使能此输出。
  - RTC_ALARM：可通过配置 RTC_CR 寄存器中的 OSEL[1:0] 位使能此输出，可选择闹钟 A、闹钟 B 或唤醒输出。
- 复用功能输入：
  - RTC_TS：时间戳事件
  - RTC_TAMP1：tamper1 事件检测
  - RTC_TAMP2：tamper2 事件检测
  - RTC_TAMP3：tamper3 事件检测（仅第 5 类器件）
  - RTC_REFIN：50 Hz 或 60 Hz 参考时钟输入

27.3.2 RTC 控制的 GPIO

RTC_OUT、RTC_TS 和 RTC_TAMP1 映射到同一引脚 (PC13)。无论 PC13 的 GPIO 配置如何，PC13 引脚的配置都由 RTC 控制。

输出机制遵循*表 103* 中所示的优先级顺序。

表 103. RTC 引脚 PC13 配置(1)

PC13 引脚配置和函数	OSEL[1:0] 位（RTC_ALARM 输出使能）	COE 位（RTC_CALIB 输出使能）	RTC_OUT _RMP 位	RTC_ALARM _TYPE 位	TAMP1E 位（RTC_TAMP1 输入使能）	TSE位（RTC_TS 输入使能）
RTC_ALARM 输出 OD	01 或 10 或 11	无关	0	0	无关	无关
			1			
RTC_ALARM 输出 PP	01 或 10 或 11	无关	0	1	无关	无关
			1			
RTC_CALIB 输出 PP	00	1	0	无关	无关	无关
RTC_TAMP1 输入浮空	00	0	无关	无关	1	0
	00	1	1			
	01 或 10 或 11	0				
RTC_TS 和 RTC_TAMP1 输入浮空	00	0	无关	无关	1	1
	00	1	1			
	01 或 10 或 11	0				

表 103. RTC 引脚 PC13 配置[1]（续）

PC13 引脚配置和函数	OSEL[1:0] 位（RTC_ALARM 输出使能）	COE 位（RTC_CALIB 输出使能）	RTC_OUT_RMP 位	RTC_ALARM_TYPE 位	TAMP1E 位（RTC_TAMP1 输入使能）	TSE 位（RTC_TS 输入使能）
RTC_TS 输入浮空	00	0	无关	无关	0	1
	00	1	1			
	01 或 10 或 11	0				
唤醒引脚或标准 GPIO	00	0	无关	无关	0	0
	00	1	1			
	01 或 10 或 11	0				

1. OD：开漏；PP：推挽。

此外，可借助 RTC_OUT_RMP 位将 RTC_OUT 重映射到 PB14 引脚上。在这种情况下，必须将 PB14 GPIO 寄存器配置为类型正确的复用功能。重映射功能如*表 104* 所示。

表 104. RTC_OUT 映射

OSEL[1:0] 位（RTC_ALARM 输出使能）	COE 位（RTC_CALIB 输出使能）	RTC_OUT_RMP 位	RTC_OUT 映射到 PC13	RTC_OUT 映射到 PB14
00	0	0	-	-
00	1		RTC_CALIB	-
01 或 10 或 11	无关		RTC_ALARM	-
00	0	1	-	-
00	1		-	RTC_CALIB
01 或 10 或 11	0		-	RTC_ALARM
01 或 10 或 11	1		RTC_ALARM	RTC_CALIB

27.3.3 时钟和预分频器

RTC 时钟源 (RTCCLK) 通过时钟控制器从 LSE 时钟、LSI 振荡器时钟以及 HSE 时钟三者中选择。有关 RTC 时钟源配置的更多信息，请参见*第 7 节：复位和时钟控制 (RCC)*。

可编程的预分频器阶段可生成 1 Hz 的时钟，用于更新日历。为最大程度地降低功耗，预分频器分为 2 个可编程的预分频器（参见*图 202：RTC 框图*）：

- 一个通过 RTC_PRER 寄存器的 PREDIV_A 位配置的 7 位异步预分频器。
- 一个通过 RTC_PRER 寄存器的 PREDIV_S 位配置的 15 位同步预分频器。

注： *使用两个预分频器时，推荐将异步预分频器配置为较高的值，以最大程度降低功耗。*

要使用频率为 32.768 kHz 的 LSE 获得频率为 1 Hz 的内部时钟 (ck_spre)，需要将异步预分频系数设置为 128，并将同步预分频系数设置为 256。

分频系数的最小值为 1，最大值为 2^{22}。

这对应于约为 4 MHz 的最大输入频率。

f_{ck_apre} 可根据以下公式得出：

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

ck_apre 时钟用于为二进制 RTC_SSR 亚秒递减计数器提供时钟。当该计数器计数到 0 时，会使用 PREDIV_S 的内容重载 RTC_SSR。

f_{ck_spre} 可根据以下公式得出：

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

ck_spre 时钟既可以用于更新日历，也可以用作 16 位唤醒自动重载定时器的时基。为获得较短的超时周期，还可以将 16 位唤醒自动重载定时器与经可编程的 4 位异步预分频器分频的 RTCCLK 一同运行（有关详细信息，请参见*第 27.3.6 节：周期性自动唤醒*）。

27.3.4 实时时钟和日历

RTC 日历时间和日期寄存器可通过与 PCLK（APB 时钟）同步的影子寄存器来访问。这些时间和日期寄存器也可以直接访问，这样可避免等待同步的持续时间。

- RTC_SSR 对应于亚秒
- RTC_TR 对应于时间
- RTC_DR 对应于日期

每隔两个 RTCCLK 周期，便将当前日历值复制到影子寄存器，并将 RTC_ISR 寄存器的 RSF 位置 1（请参见*第 27.6.4 节：RTC 初始化和状态寄存器 (RTC_ISR)*）。在停机和待机模式下不会执行复制操作。退出这两种模式时，影子寄存器会在最长 2 个 RTCCLK 周期后进行更新。

当应用读取日历寄存器时，它会访问影子寄存器的内容。也可以通过将 RTC_CR 寄存器的 BYPSHAD 控制位置 1 来直接访问日历寄存器。默认情况下，该位被清零，用户访问影子寄存器。

在 BYPSHAD=0 模式下读取 RTC_SSR、RTC_TR 或 RTC_DR 寄存器时，APB 时钟频率 (f_{APB}) 必须至少为 RTC 时钟频率 (f_{RTCCLK}) 的 7 倍。

影子寄存器通过系统复位来复位。

27.3.5 可编程闹钟

RTC 单元提供两个可编程闹钟，即闹钟 A 和闹钟 B。以下说明针对闹钟 A，但同样适用于闹钟 B。

可通过 RTC_CR 寄存器中的 ALRAE 位来使能可编程闹钟功能。如果日历亚秒、秒、分钟、小时、日期或日与闹钟寄存器 RTC_ALRMASSR 和 RTC_ALRMAR 中编程的值相匹配，则 ALRAF 标志会被置为 1。可通过 RTC_ALRMAR 寄存器的 MSKx 位以及 RTC_ALRMASSR 寄存器的 MASKSSx 位单独选择各日历字段。可通过 RTC_CR 寄存器中的 ALRAIE 位来使能闹钟中断。

注意： 如果选择秒字段（RTC_ALRMAR 中的 MSK1 位复位），则 RTC_PRER 寄存器中设置的同步预分频器分频系数必须至少为 3，才能确保闹钟正确地运行。

闹钟 A 和闹钟 B（如果已通过 RTC_CR 寄存器中的位 OSEL[0:1] 使能）可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 输出极性。

27.3.6 周期性自动唤醒

周期性唤醒标志由 16 位可编程自动重载递减计数器生成。唤醒定时器范围可扩展至 17 位。

可通过 RTC_CR 寄存器中的 WUTE 位来使能此唤醒功能。

唤醒定时器的时钟输入可以是：

- 2、4、8 或 16 分频的 RTC 时钟 (RTCCLK)。

  当 RTCCLK 为 LSE (32.768 kHz) 时，可配置的唤醒中断周期介于 122 µs 和 32 s 之间，且分辨率低至 61 µs。
- ck_spre（通常为 1 Hz 内部时钟）。

  当 ck_spre 频率为 1 Hz 时，可得到的唤醒时间为 1s 到 36h 左右，分辨率为 1 秒。这一较大的可编程时间范围分为两部分：
  - WUCKSEL [2:1] = 10 时为 1s 到 18h
  - WUCKSEL [2:1] = 11 时约为 18h 到 36h。在后一种情况下，会将 2^{16} 添加到 16 位计数器当前值。完成初始化序列后（请参见*第 600 页的编程唤醒定时器*），定时器开始递减计数。在低功耗模式下使能唤醒功能时，递减计数保持有效。此外，当计数器计数到 0 时，RTC_ISR 寄存器的 WUTF 标志会置 1，并且唤醒寄存器会使用其重载值（RTC_WUTR 寄存器值）动重载。

之后必须用软件清零 WUTF 标志。

通过将 RTC_CR2 寄存器中的 WUTIE 位置 1 来使能周期性唤醒中断时，它会使器件退出低功耗模式。

如果已通过 RTC_CR 寄存器的位 OSEL[1:0] 使能周期性唤醒标志，则该标志可连接到 RTC_ALARM 输出。可通过 RTC_CR 寄存器的 POL 位配置 RTC_ALARM 输出极性。

系统复位以及低功耗模式（睡眠、停机和待机）对唤醒定时器没有任何影响。

27.3.7 RTC 初始化和配置

RTC 寄存器访问

RTC 寄存器为 32 位寄存器。除了当 BYPSHAD=0 时对日历影子寄存器执行的读访问之外，APB 接口会在访问 RTC 寄存器时引入 2 个等待周期。

RTC 寄存器写保护

系统复位后，可通过对 PWR_CR 寄存器中的 DBP 位清零来保护 RTC 寄存器以防止非正常的写访问（请参见电源控制部分）。必须将 DBP 位置 1 才能使能 RTC 寄存器的写访问。

RTC 域复位后，所有 RTC 寄存器均受到写保护。通过向写保护寄存器 (RTC_WPR) 写入一个密钥来使能对 RTC 寄存器的写操作。

要解锁所有 RTC 寄存器（RTC_TAMPCR、RTC_BKPxR、RTC_OR 和 RTC_ISR[13:8] 除外）的写保护，需要执行以下步骤：

1. 将“0xCA”写入 RTC_WPR 寄存器。
2. 将“0x53”写入 RTC_WPR 寄存器。

写入一个错误的关键字会再次激活写保护。

保护机制不受系统复位影响。

日历初始化和配置

要编程包括时间格式和预分频器配置在内的初始时间和日期日历值，需按照以下顺序操作：

1. 将 RTC_ISR 寄存器中的 INIT 位置 1 以进入初始化模式。在此模式下，日历计数器将停止工作并且其值可更新。
2. 轮询 RTC_ISR 寄存器中的 INITF 位。当 INITF 置 1 时进入初始化阶段模式。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 要为日历计数器生成 1 Hz 时钟，应编程 RTC_PRER 寄存器中的两个预分频系数。
4. 在影子寄存器（RTC_TR 和 RTC_DR）中加载初始时间和日期值，然后通过 RTC_CR 寄存器中的 FMT 位配置时间格式（12 或 24 小时制）。
5. 通过清零 INIT 位退出初始化模式。随后，自动加载实际日历计数器值，在 4 个 RTCCLK 时钟周期后重新开始计数。

当初始化序列完成之后，日历开始计数。

注： *系统复位后，应用可读取 RTC_ISR 寄存器中的 INITS 标志，以检查日历是否已初始化。如果该标志为 0，表明日历尚未初始化，因为年份字段设置为其RTC 域复位时的默认值 (0x00)。*

要在初始化之后读取日历，必须首先用软件检查 RTC_ISR 寄存器的 RSF 标志是否置 1。

有关代码示例，请参见 *A.15.1：RTC 日历配置代码示例*。

夏令时

可通过 RTC_CR 寄存器的 SUB1H、ADD1H 和 BKP 位管理夏令时。

利用 SUB1H 或 ADD1H，软件只需单次操作便可在日历中减去或增加一个小时，无需执行整个初始化步骤。

此外，软件还可以使用 BKP 位来记录是否曾经执行过此操作。

编程闹钟

要对可编程的闹钟进行编程或更新，必须执行类似的步骤。以下步骤针对闹钟 A，但同样适用于闹钟 B。

1. 将 RTC_CR 中的 ALRAE 位清零以禁止闹钟 A。
2. 编程闹钟 A 寄存器 (RTC_ALRMASSR/RTC_ALRMAR)。
3. 将 RTC_CR 寄存器中的 ALRAE 位置 1 以再次使能闹钟 A。

注： *约 2 个 RTCCLK 时钟周期（由于时钟同步）后，将执行对 RTC_CR 寄存器的更改。*

有关代码示例，请参见 *A.15.2：RTC 闹钟配置代码示例*。

编程唤醒定时器

要配置或更改唤醒定时器的自动重载值（RTC_WUTR 中的 WUT[15:0]），需要按照以下顺序操作：

1. 清零 RTC_CR 中的 WUTE 以禁止唤醒定时器。
2. 轮询 RTC_ISR 中的 WUTWF，直到该位置 1，以确保可以访问唤醒自动重载定时器和 WUCKSEL[2:0] 位。大约需要 2 个 RTCCLK 时钟周期（由于时钟同步）。
3. 编程唤醒自动重载值 WUT[15:0]，并选择唤醒时钟（RTC_CR 中的 WUCKSEL[2:0] 位）。将 RTC_CR 寄存器中的 WUTE 位置 1 以再次使能定时器。唤醒定时器重新开始递减计数。由于时钟同步，WUTWF 位会在 WUTE 清零达 2 个 RTCCLK 时钟周期后清零。

有关代码示例，请参见 *A.15.3：RTC WUT 配置代码示例*。

27.3.8 读取日历

当 RTC_CR 寄存器中的 BYPSHAD 控制位清零时

要正确读取 RTC 日历寄存器（RTC_SSR、RTC_TR 和 RTC_DR），APB1 时钟频率 (f_{PCLK}) 必须等于或大于 RTC 时钟频率 (f_{RTCCLK}) 的七倍。这可以确保同步机制行为的安全性。

如果 APB1 时钟频率低于 RTC 时钟频率的七倍，则软件必须读取日历时间寄存器和日期寄存器两次。这样，当两次读取的 RTC_TR 结果相同时，才能确保数据正确。否则必须执行第三次读访问。任何情况下，APB1 的时钟频率都不能低于 RTC 的时钟频率。

每次将日历寄存器中的值复制到 RTC_SSR、RTC_TR 和 RTC_DR 影子寄存器时，RTC_ISR 寄存器中的 RSF 位都会置 1。每两个 RTCCLK 周期执行一次复制。为确保这 3 个值一致，读取 RTC_SSR 或 RTC_TR 时会锁定高阶日历影子寄存器中的值，直到读取 RTC_DR。为避免软件对日历执行读访问的时间间隔小于 2 个 RTCCLK 周期：第一次读取日历之后必须通过软件将 RSF 清零，并且软件必须等待到 RSF 置 1 之后才可再次读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

从低功耗模式（停机模式或待机模式）唤醒之后，必须通过软件将 RSF 清零。之后，软件必须等待至 RSF 再次置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

RSF 位必须在唤醒之后而不是进入低功耗模式之前进行清零。

系统复位之后，软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。实际上，系统复位会将影子寄存器复位为其默认值。

初始化之后（请参见*第 599 页的日历初始化和配置*），软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

同步之后（请参见*第 27.3.10 节：RTC 同步*），软件必须等待至 RSF 置 1 之后才可以读取 RTC_SSR、RTC_TR 和 RTC_DR 寄存器。

有关代码示例，请参见 *A.15.4：RTC 读日历代码示例*。

当 RTC_CR 寄存器中的 BYPSHAD 控制位置 1 时（旁路影子寄存器）

读取日历寄存器时会直接从日历计数器获取值，这样便无需等待至 RSF 位置 1。这对于从低功耗模式（停机模式或待机模式）退出后的情况特别有用，因为影子寄存器在这些模式下不更新。

当 BYPSHAD 位置 1 时，如果在对寄存器的两次读访问之间出现 RTCCLK 沿，则不同寄存器的结果彼此可能不一致。此外，如果在读操作期间出现 RTCCLK 沿，则可能导致其中一个寄存器的值不正确。软件必须分两次读取所有寄存器，然后将两次结果加以比较来确认数据是否一致和正确。此外，软件也可以只比较两次读取日历寄存器得到的结果的最低位。

注： *当 BYPSHAD=1 时，读取日历寄存器的指令需要一个额外的 APB 周期才能完成。*

27.3.9 复位 RTC

日历影子寄存器（RTC_SSR、RTC_TR 和 RTC_DR）以及 RTC 状态寄存器 (RTC_ISR) 的某些位通过所有可用的系统复位源复位为各自的默认值。

相反，以下寄存器则通过RTC域复位来复位为各自的默认值并且不受系统复位的影响：RTC 当前日历寄存器、RTC 控制寄存器 (RTC_CR)、预分频器寄存器 (RTC_PRER)、RTC 校准寄存器 (RTC_CALR)、RTC 移位寄存器 (RTC_SHIFTR)、RTC 时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）、RTC 入侵和复用功能配置寄存器 (RTC_TAMPCR)、RTC 备份寄存器 (RTC_BKPxR)、唤醒定时器寄存器 (RTC_WUTR)、闹钟 A 和闹钟 B 寄存器（RTC_ALRMASSR/RTC_ALRMAR 和 RTC_ALRMBSSR/RTC_ALRMBR）以及选项寄存器 (RTC_OR)。

此外，当由 LSE 提供时钟时，如果复位源并非 RTC 域复位源（有关不受系统复位影响的 RTC 时钟源列表的详细信息，请参见“复位和时钟控制器”的 *RTC 时钟*），则 RTC 将在系统复位时保持运行状态。发生RTC域复位时，RTC 会停止工作，并且所有 RTC 寄存器都会设置为各自的复位值。

27.3.10 RTC 同步

RTC 可与高精度的远程时钟同步。在读取亚秒字段后（RTC_SSR 或 RTC_TSSSR），即可计算远程时钟的时间与 RTC 之间的精准偏差。之后，可使用 RTC_SHIFTR 对 RTC 的时钟进行零点几秒的“平移”，经过调整后可消除此偏差。

RTC_SSR 包含同步预分频器计数器的值。这样，便可计算分辨率低至 1/(PREDIV_S + 1) 秒的 RTC 的准确时间。因此，可通过增大同步预分频器的值 (PREDIV_S[14:0]) 来提高分辨率。将 PREDIV_S 设置为 0x7FFF 时，可得到允许的最大分辨率（30.52 µs，时钟频率为 32768 Hz）。

但是，提高 PREDIV_S 意味着必须降低 PREDIV_A 才能将同步预分频器的输出维持在 1 Hz。这样，异步预分频器的输出频率会增大，RTC 的动态功耗也会相应增加。

可以使用 RTC 平移控制寄存器 (RTC_SHIFTR) 对 RTC 进行微调。可以用大小为 1/(PREDIV_S + 1) 秒的分辨率对 RTC_SHIFTR 进行写操作，将时钟平移（延迟或提前）最长 1 秒。在这种平移操作中，会将 SUBFS[14:0] 值加到同步预分频器计数器 SS[15:0] 中：这将使时钟产生延迟。如果同时将 ADD1S 位置 1，则会增加一秒，与此同时减去的时间为零点几秒，因此将使时钟提前。

注意： 初始化平移操作前，用户必须检查确认 SS[15] = 0，以确保不会发生上溢。

对 RTC_SHIFTR 寄存器执行写操作以启动平移操作时，硬件会将 SHPF 标志置 1 以指示平移操作挂起。完成平移操作时，硬件会将将该位清零。

注意： 该同步功能与参考时钟检测功能不兼容：当 REFCKON=1 时，固件不能对 RTC_SHIFTR 执行写操作。

27.3.11 RTC 参考时钟检测

RTC 日历更新可与参考时钟 RTC_REFIN（通常为市电频率，50 Hz 或 60 Hz）同步。RTC_REFIN 参考时钟的精度应高于 32.768 kHz LSE 时钟。使能 RTC_REFIN 检测时（将 RTC_CR 的 REFCKON 位置 1），日历仍由 LSE 提供时钟，而 RTC_REFIN 用于补偿不准确的日历更新频率 (1 Hz)。

每个 1 Hz 时钟边沿都与最近的 RTC_REFIN 时钟边沿进行比较（如果在给定的时间窗口内发现一个边沿）。在大多数情况下，两个时钟边沿恰好对齐。当 1 Hz 时钟由于 LSE 时钟不精确而发生偏离时，RTC 会稍微偏移 1 Hz 时钟，以便后续的 1 Hz 时钟边沿能够对齐。利用这种机制，可使日历像参考时钟一样精确。

RTC 使用 32.768 kHz 石英产生的 256 Hz 时钟 (ck_apre) 检测是否存在参考时钟源。大约在日历每次更新时（每 1 秒钟），便会在时间窗口期间执行一次检测。检测到第一个参考时钟边沿时，该窗口等于 7 个 ck_apre 周期。随后的日历更新使用长度为 3 个 ck_apre 周期的较小窗口。

每次在窗口中检测到参考时钟时，都会行强制输出 ck_apre 时钟的异步预分频器进行重载。当参考时钟与 1 Hz 时钟对齐时，此操作不起作用，因为预分频器会在同一时刻重载。当时钟不对齐时，重载操作会微调后续的 1 Hz 时钟边沿，使其与参考时钟对齐。

如果参考时钟停止（在 3 个 ck_apre 窗口内未出现参考时钟边沿），日历将仅根据 LSE 时钟进行连续更新。RTC 随后使用 ck_spre 边沿上居中的大检测窗口（7 个 ck_apre 周期）等待参考时钟。

使能 RTC_REFIN 检测后，必须将 PREDIV_A 和 PREDIV_S 设置为各自的默认值：

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

注： *RTC_REFIN 时钟检测在待机模式下不可用。*

27.3.12 RTC 精密数字校准

RTC 频率可采用约 0.954 ppm 的分辨率进行数字校准，校准范围为 -487.1 ppm 到 +488.5 ppm。使用一系列微调（增加和/或减少单独的 RTCCLK 脉冲）进行频率校正。这些微调的分布非常均匀，因此 RTC 的校准效果相当好，即使在短时间内持续观察也是如此。

当输入频率为 32768 Hz 时，精密数字校准的周期约为 2^{20} 个 RTCCLK 脉冲或 32 秒。此周期由一个通过 RTCCLK 提供时钟信号的 20 位计数器 cal_cnt[19:0] 维持。

精密数字校准寄存器 (RTC_CALR) 可指定 32 秒周期内要减少的 RTCCLK 时钟周期数：

- 将位 CALM[0] 置 1 时，32 秒周期内将只

减少 1 个脉冲。

- 将 CALM[1] 置 1 时，将减少两个周期。
- 将 CALM[2] 置 1 时，将减少 4 个周期
- 依此类推，将 CALM[8] 置 1 时，将减少 256 个时钟。

注： *CALM[8:0] (RTC_CALR) 可指定 32 秒周期内要减少的 RTCCLK 脉冲数。将位 CALM[0] 置 1 时，32 秒周期内将只减少 1 个脉冲（当 cal_cnt[19:0] = 0x80000 时）；将 CALM[1] 置 1 时，将减少 2 个周期（当 cal_cnt = 0x40000 和 0xC0000 时）；将 CALM[2] 置 1 时，将减少 4 个周期（当 cal_cnt = 0x20000/0x60000/0xA0000/0xE0000 时）；依此类推，将 CALM[8] 置 1 时，将减少 256 个时钟（当 cal_cnt = 0xXX800 时）。*

使用适当分辨率时，CALM 可使 RTC 频率减少最多 487.1 ppm，而 CALP 可用于使频率增加 488.5 ppm。将 CALP 置“1”，可每隔 2^{11} 个 RTCCLK 周期有效插入一个额外的 RTCCLK 脉冲，这意味着每 32 秒周期可增加 512 个时钟。

与 CALM 和 CALP 配合使用时，可在 32 秒周期内增加一个范围为 -511 到 +512 RTCCLK 周期的偏差，对应的校准范围为 -487.1 ppm 到 +488.5 ppm，分辨率约为 0.954 ppm。

若输入频率 (F_{RTCCLK}) 已知，可通过以下公式计算有效校准频率 (F_{CAL})：

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

PREDIV_A<3 条件下的校准

当异步预分频器值（RTC_PRER 寄存器中的 PREDIV_A 位）小于 3 时，不能将 CALP 位置 1。如果 CALP 已置 1 并且 PREDIV_A 位的值小于 3，则会忽略 CALP，即假定 CALP 等于 0 而执行校准。

要在 PREDIV_A 小于 3 的条件下执行校准，应降低同步预分频器值 (PREDIV_S) 以便每秒内可加速 8 个 RTCCLK 时钟周期，这意味着每 32 秒可增加 256 个时钟周期。因此，仅使用 CALM 位，可在每 32 秒内有效增加 255 到 256 个时钟脉冲（对应的校准范围为 243.3 ppm 到 244.1 ppm）。

在标称 RTCCLK 频率 32768 Hz 下，当 PREDIV_A 等于 1 时（分频系数为 2），应将 PREDIV_S 设置为 16379 而不是 16383（少 4）。唯一相关的其它情况是，当 PREDIV_A 等于 0 时，应将 PREDIV_S 设置为 32759 而不是 32767（少 8）。

如果以这种方式减少 PREDIV_S，则采用以下公式计算校准

输入时钟的有效频率：

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

在这种情况下，如果 RTCCLK 恰好为 32768.00 Hz，则当 CALM[7:0] 等于 0x100 时（CALM 范围的中值），说明设置正确。

验证 RTC 校准

通过测量 RTCCLK 的精确频率，计算正确的 CALM 和 CALP 值以确保 RTC 精度。此外，还为应用提供了一个可选的 1 Hz 输出，用来测量和验证 RTC 精度。

如果在有限的间隔内测量 RTC 的精确频率，则会导致测量期间产生最多 2 个 RTCCLK 时钟周期的测量误差，具体取决于数字校准周期与测量周期的对齐方式。

但是，如果测量周期与校准周期的长度相同，则可以消除此测量误差。在这种情况下，观测到的唯一误差是由数字校准的分辨率导致的误差。

- 默认情况下，校准周期为 32 秒。

在此模式下，测量整个 32 秒内 1 Hz 输出的精度，可确保测量误差在 0.477 ppm 内（32 秒内为 0.5 个 RTCCLK 周期，受校准分辨率限制）。

- 可将 RTC_CALR 寄存器的 CALW16 位置 1，以强制 16 秒的校准周期。

此时，可在 16 秒内测量 RTC 精度，产生的最大误差为 0.954 ppm（16 秒内为 0.5 个 RTCCLK 周期）。但是，由于校准分辨率降低，长期的 RTC 精度也会降到 0.954 ppm：将 CALW16 置 1 时，CALM[0] 位将始终保持为 0。

- 可将 RTC_CALR 寄存器的 CALW8 位置 1，以强制 8 秒的校准周期。

此时，可在 8 秒内测量 RTC 精度，产生的最大误差为 1.907 ppm（8 秒内为 0.5 个 RTCCLK 周期）。长期的 RTC 精度也会降到 1.907 ppm：将 CALW8 置 1 时，CALM[1:0] 位将始终保持为 00。

动态重校准

当 RTC_ISR/INITF=0 时，可动态更新校准寄存器 (RTC_CALR)，具体步骤如下：

1. 轮询 RTC_ISR/RECALPF（重新校准挂起标志）。
2. 如果该标志为 0，则可以根据需要向 RTC_CALR 写入新值。随后 RECALPF 位会被自动置为 1。
3. 新校准设置将在对 RTC_CALR 执行写操作之后的三个 ck_apre 周期内生效。

有关代码示例，请参见 *A.15.5：RTC 校准代码示例*。

27.3.13 时间戳功能

将 RTC_CR 寄存器的 TSE 位置 1 可使能时间戳。

当在 RTC_TS 引脚上检测到时间戳事件时，日历会保存到时间戳寄存器（RTC_TSSSR、RTC_TSTR 和 RTC_TSDR）中。

发生时间戳事件时，RTC_ISR 寄存器中的时间戳标志位 (TSF) 将置 1。

通过将 RTC_CR 寄存器中的 TSIE 位置 1，可在发生入侵检测事件时生成中断。

如果在时间戳标志 (TSF) 已置 1 的条件下检测到新的时间戳事件，则时间戳上溢标志 (TSOVF) 将置 1，而时间戳寄存器（RTC_TSTR 和 RTC_TSDR）将保持上一事件的结果。

注： *在发生由同步过程引发的时间戳事件后，TSF 在 2 个 ck_apre 周期置为 1。*

TSOVF 的产生中不存在延迟。也就是说如果两个时间戳事件接连发生，TSOVF 可能为“1”而TSF 为“0”。因此，建议只在检测到 TSF 为“1”后再轮询 TSOVF。

注意： 如果在 TSF 位清零后紧接着发生时间戳事件，则 TSF 和 TSOVF 位都将置 1。为防止在时间戳事件发生的同时屏蔽该事件，除非已将 TSF 位读为“1”，否则应用程序不得将“0”写入 TSF 位。

此外，入侵事件可能导致时间戳被记录。有关 TAMPTS 控制位的说明，请参见*第 27.6.16 节：RTC 入侵配置寄存器 (RTC_TAMPCR)*。

27.3.14 入侵检测

RTC_TAMPx 输入事件既可配置为边沿检测，也可配置为带过滤的电平检测。

入侵检测可配置用于以下目的：

- 擦除 RTC 备份寄存器（默认配置）
- 生成中断，能够从停止模式和待机模式唤醒
- 为低功耗定时器生成硬件触发

RTC 备份寄存器

备份寄存器 (RTC_BKPxR) 不会通过系统复位来复位，也不会在器件从待机模式唤醒时复位。

备份寄存器在发生入侵检测事件（请参见*第 27.6.20 节：RTC 备份寄存器 (RTC_BKPxR)* 和*第 605 页的入侵检测初始化*）或者在将 Flash 的读保护从级别 1 更改为级别 0 时复位，TAMPxNOERASE 位置 1 时或者 RTC_TAMPCR 寄存器中的 TAMPxMF 置 1 时除外。

入侵检测初始化

可通过将 RTC_TAMPCR 寄存器中相应的 TAMPxE 位置 1 来使能各输入。

每个 RTC_TAMPx 入侵检测输入分别与 RTC_ISR 寄存器中的标志 TAMPxF 相关联。

将 TAMPxMF 清零时：

TAMPxF 标志将在引脚上出现入侵事件后使能，并存在下述延迟：

- 当 TAMPFLT 不为 0x0 时（带过滤的电平检测），延迟为 3 个 ck_apre 周期
- 当 TAMPTS=1 时（入侵事件的时间戳），延迟为 3 个 ck_apre 周期
- 当 TAMPFLT=0x0（边沿检测）且 TAMPTS=0 时，无延迟

在此期间，只要 TAMPxF 置 1，就无法检测到同一引脚上出现的新入侵。

将 TAMPxMF 置 1 时：

在上述延迟期间以及在 2.5 个 ck_rtc 附加周期内，无法检测到同一引脚上出现的新入侵。

通过将 RTC_TAMPCR 寄存器中的 TAMPIE 位置 1，可在发生入侵检测事件时（当 TAMPxF 置 1 时）生成中断。当一个或多个 TAMPxMF 置 1 时，不允许将 TAMPIE 置 1。

将 TAMPIE 清零时，可通过将 RTC_TAMPCR 寄存器中相应的 TAMPxIE 位置 1 来分别使能各个入侵引脚事件中断。当相应的 TAMPxMF 置 1 时，不允许将 TAMPxIE 置 1。

出现入侵事件时生成触发输出

低功耗定时器可将入侵事件检测用作触发输入。

将 RTC_TAMPCR 寄存器中的 TAMPxMF 位清零时，必须通过软件将 TAMPxF 标志清零以便在同一引脚上检测新入侵。

将 TAMPxMF 位置 1 时，TAMPxF 标志会被屏蔽，并在 RTC_ISR 寄存器中保持清零。此配置允许在停止模式下自动触发低功耗定时器，无需通过系统唤醒来执行 TAMPxF 清零。在这种情况下，备份寄存器不清零。

入侵事件的时间戳

当 TAMPTS 置“1”时，任何入侵事件都会导致时间戳事件发生。在这种情况下，如同发生正常时间戳事件一样，RTC_ISR 中的 TSF 位或 TSOVF 位会置 1。在 TSF 或 TSOVF 置 1 的同时，受影响的入侵标志寄存器 TAMPxF 也会随之置 1。

对入侵输入的边沿检测

如果 TAMPFLT 位设置为“00”，当相应的 TAMPxTRG 位上出现上升沿或下降沿时，RTC_TAMPx 引脚将生成入侵检测事件。选择边沿检测时，会禁止 RTC_TAMPx 输入上的内部上拉电阻。

注意： 使用边沿检测时，建议在使能入侵检测后（通过读取 GPIO 寄存器）以及向备份寄存器中写入敏感值前立即通过软件检查入侵引脚电平，以确保使能入侵事件检测后才出现有效边沿。当 TAMPFLT = “00”且 TAMPxTRG = 0（上升沿检测）时，如果在使能入侵检测前入侵输入已处于高电平，则可通过硬件来检测入侵事件。

检测到入侵事件并清零后，应当在重新编程备份寄存器 (RTC_BKPxR) 之前禁止 RTC_TAMPx 复用功能，然后再重新使能（TAMPxE 置 1）。这可防止应用在 RTC_TAMPx 输入值仍指示入侵检测时，对备份寄存器执行写操作。这相当于对 RTC_TAMPx 复用功能输入的电平检测。

对 RTC_TAMPx 输入的带过滤电平检测

通过将 TAMPFLT 设置为非零值可执行带过滤的电平检测。在 TAMPxTRG 位指定的电平连续出现 2、4 或 8 个（取决于 TAMPFLT 值）采样时生成入侵检测事件。

除非通过将 TAMPPUDIS 置 1 禁止入侵输入，否则在入侵输入的状态被采样前，将通过 I/O 内部上拉电阻对 RTC_TAMPx 输入预充电。预充电的持续时间由 TAMPPRCH 位确定，允许增大 RTC_TAMPx 输入上的电容。

可使用 TAMPFREQ 确定用于电平检测的采样频率，以便使入侵检测延迟与上拉电阻功耗之间达到最佳平衡。

注： *有关上拉电阻的电气特性，请参见数据手册。*

有关代码示例，请参见 *A.15.6：RTC 入侵和时间戳配置代码示例*。

27.3.15 校准时钟输出

将 RTC_CR 寄存器中的 COE 位置 1 时，会在 RTC_CALIB 器件输出上提供一个参考时钟。

如果 RTC_CR 寄存器中的 COSEL 位复位且 PREDIV_A = 0x7F，则 RTC_CALIB 频率为 $f_{RTCCLK/64}$。这相当于 RTCCLK 频率为 32.768 kHz 时，512 Hz 的校准输出。RTC_CALIB 占空比是不规则的：下降沿上存在轻微抖动。因此推荐使用上升沿。

如果 COSEL 置 1 且“PREDIV_S+1”为 256 的非零整数倍（比如：PREDIV_S[7:0] = 0xFF），则 RTC_CALIB 频率为 f_{RTCCLK}/(256 * (PREDIV_A+1))。这相当于 RTCCLK 频率为 32.768 kHz 时，1 Hz 的校准输出，其中预分频器为默认值（PREDIV_A = Ox7F、PREDIV_S = 0xFF）。

注： *选择 RTC_CALIB 或 RTC_ALARM 输出时，RTC_OUT 引脚会自动配置为输出复用功能。*

有关代码示例，请参见 *A.15.7：RTC 入侵和时间戳代码示例*。

27.3.16 闹钟输出

RTC_CR 寄存器中的 OSEL[1:0] 控制位用于激活闹钟复用功能输出 RTC_ALARM，以及选择输出的功能。这些功能可反映 RTC_ISR 寄存器中相应标志的内容。

输出的极性由 RTC_CR 中的 POL 控制位确定，这样当 POL 置 1 时会输出选定标志位的相反值。

闹钟复用功能输出

使用 RTC_OR 寄存器中的控制位 RTC_ALARM_TYPE 可将 RTC_ALARM 引脚配置为输出开漏或输出上拉。

注： *使能 RTC_ALARM 输出之后，其优先级高于 RTC_CALIB（与 COE 位无关，该为必须保持清零）。*

选择 RTC_CALIB 或 RTC_ALARM 输出时，RTC_OUT 引脚会自动配置为输出复用功能。

27.4 RTC 低功耗模式

表 105. 低功耗模式对 RTC 的作用

模式	说明
睡眠	无任何作用 RTC 中断可使器件退出睡眠模式。
停止	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件和 RTC 唤醒会使器件退出停机模式。
待机	当 RTC 时钟源为 LSE 或 LSI 时，RTC 保持工作状态。RTC 闹钟、RTC 入侵事件、RTC 时间戳事件和 RTC 唤醒会使器件退出待机模式。

27.5 RTC 中断

所有 RTC 中断均与 EXTI 控制器相连。请参见*第 13.5 节：EXTI 寄存器*。

要使能 RTC 中断，需执行以下序列：

1. 将对应于 RTC 事件的 EXTI 线配置为中断模式并将其使能，然后选择上升沿有效。
2. 配置 NVIC 中的 RTC IRQ 通道并将其使能。
3. 将 RTC 配置为生成 RTC 中断。

表 106. 中断控制位

中断事件	事件标志	使能控制位	退出睡眠模式	退出停止模式	退出待机模式
闹钟 A	ALRAF	ALRAIE	是	是(1)	是(1)
闹钟 B	ALRBF	ALRBE	是	是(1)	是(1)
RTC_TS 输入（时间戳）	TSF	TSIE	是	是(1)	是(1)
RTC_TAMP1 输入检测	TAMP1F	TAMPIE	是	是(1)	是(1)
RTC_TAMP2 输入检测	TAMP2F	TAMPIE	是	是(1)	是(1)
唤醒定时器中断	WUTF	WUTIE	是	是(1)	是(1)

1. 仅当 RTC 时钟源为 LSE 或 LSI 时，才能从停机和待机模式唤醒。

27.6 RTC 寄存器

有关寄存器说明中使用的缩写，请参见参考手册中的*第 50 页的第 1.1 节*。

外设寄存器可按字（32 位）进行访问。

27.6.1 RTC 时间寄存器 (RTC_TR)

RTC time register

RTC_TR 是日历时间影子寄存器。只能在初始化模式下对该寄存器执行写操作。请参见*第 599 页的日历初始化和配置*和*第 600 页的读取日历*。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x00

RTC域复位值：0x0000 0000

系统复位：当 BYPSHAD = 0 时为 0x0000 0000。当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31-23 保留，必须保持复位值

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)

位 15 保留，必须保持复位值。

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)

位 7 保留，必须保持复位值。

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)

27.6.2 RTC 日期寄存器 (RTC_DR)

RTC date register

RTC_DR 是日历日期影子寄存器。只能在初始化模式下对该寄存器执行写操作。请参见*第 599 页的日历初始化和配置*和*第 600 页的读取日历*。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x04

RTC 域复位值：0x0000 2101

系统复位：当 BYPSHAD = 0 时为 0x0000 2101。当 BYPSHAD = 1 时不受影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值

位 23:20 **YT[3:0]**：年份的十位（BCD 格式）(Year tens in BCD format)

位 19:16 **YU[3:0]**：年份的个位（BCD 格式）(Year units in BCD format)

位 15:13 **WDU[2:0]**：星期几的个位 (Week day units)

000：禁止
001：星期一
...
111：星期日

位 12 **MT**：月份的十位（BCD 格式）(Month tens in BCD format)

位 11:8 **MU**：月份的个位（BCD 格式）(Month units in BCD format)

位 7:6 保留，必须保持复位值。

位 5:4 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 3:0 **DU[3:0]**：日期的个位（BCD 格式）(Date units in BCD format)

27.6.3 RTC 控制寄存器 (RTC_CR)

RTC control register

偏移地址：0x08

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值。

位 23 **COE**：校准输出使能 (Calibration output enable)

该位使能 RTC_CALIB 输出

0：禁止校准输出

1：使能校准输出

位 22:21 **OSEL[1:0]**：输出选择 (Output selection)

这些位用于选择要连接到 RTC_ALARM 输出的标志

00：禁止输出

01：使能闹钟 A 输出

10：使能闹钟 B 输出

11：使能唤醒输出

位 20 **POL**：输出极性 (Output polarity)

该位用于配置 RTC_ALARM 输出的极性

0：当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为高电平

1：当 ALRAF/ALRBF/WUTF 置 1 时（取决于 OSEL[1:0]），该引脚为低电平

位 19 **COSEL**：校准输出选择 (Calibration output selection)

当 COE=1 时，该位可选择 RTC_CALIB 上输出的信号。

0：校准输出为 512 Hz

1：校准输出为 1 Hz

在 RTCCLK 为 32.768 kHz 且预分频器为其默认值（PREDIV_A=127 且 PREDIV_S=255）的条件下，这些频率有效。请参见*第 27.3.15 节：校准时钟输出*。

位 18 **BKP**：备份 (Backup)

用户可对此位执行写操作以记录是否已对夏令时进行更改。

位 17 **SUB1H**：减少 1 小时（冬季时间更改）(Subtract 1 hour (winter time change))

当该位在初始化模式以外的模式下置 1 时，如果当前小时不是 0，则日历时间将减少 1 小时。此位始终读为 0。

当前小时为 0 时，将此位置 1 没有任何作用。

0：无任何作用

1：将当前时间减少 1 小时。这可用于冬季时间更改。

位 16 **ADD1H**：增加 1 小时（夏季时间更改）(Add 1 hour (summer time change))
当该位在初始化模式以外的模式下置 1 时，日历时间将增加 1 小时。此位始终读为 0。
0：无任何作用
1：将当前时间增加 1 小时。这可用于夏季时间更改

位 15 **TSIE**：时间戳中断使能 (Time-stamp interrupt enable)
0：禁止时间戳中断
1：使能时间戳中断

位 14 **WUTIE**：唤醒定时器中断使能 (Wakeup timer interrupt enable)
0：禁止唤醒定时器中断
1：使能唤醒定时器中断

位 13 **ALRBIE**：*闹钟 B 中断使能 (Alarm B interrupt enable)*
0：禁止闹钟 B 中断
1：使能闹钟 B 中断

位 12 **ALRAIE**：闹钟 A 中断使能 (Alarm A interrupt enable)
0：禁止闹钟 A 中断
1：使能闹钟 A 中断

位 11 **TSE**：时间戳使能 (timestamp enable)
0：禁止时间戳
1：使能时间戳

位 10 **WUTE**：唤醒定时器使能 (Wakeup timer enable)
0：禁止唤醒定时器
1：使能唤醒定时器

位 9 **ALRBE**：*闹钟 B 使能 (Alarm B enable)*
0：禁止闹钟 B
1：使能闹钟 B

位 8 **ALRAE**：闹钟 A 使能 (Alarm A enable)
0：禁止闹钟 A
1：使能闹钟 A

位 7 保留，必须保持复位值。

位 6 **FMT**：小时格式 (Hour format)
0：24 小时/天格式
1：AM/PM 小时格式

位 5 **BYPSHAD**：旁路影子寄存器 (Bypass the shadow registers)
0：日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）取自影子寄存器，该影子寄存器每两个 RTCCLK 周期更新一次。
1：日历值（从 RTC_SSR、RTC_TR 和 RTC_DR 读取时）直接取自日历计数器。
注： 如果 APB1 时钟的频率低于 7 倍的 RTCCLK 频率，则必须将 BYPSHAD 置“1”。

位 4 **REFCKON**：RTC_REFIN 参考时钟检测使能（50 Hz 或 60 Hz）(RTC_REFIN reference clock detection enable (50 or 60 Hz))

0：禁止 RTC_REFIN 检测
1：使能 RTC_REFIN 检测

注： PREDIV_S 必须为 0x00FF。

位 3 **TSEDGE**：时间戳事件有效边沿 (Timestamp event active edge)

0：RTC_TS 输入上升沿生成时间戳事件
1：RTC_TS 输入下降沿生成时间戳事件
TSEDGE 发生更改时，必须复位 TSE 以避免将 TSF 意外置 1。

位 2:0 **WUCKSEL[2:0]**：唤醒时钟选择 (Wakeup clock selection)

000：选择 RTC/16 时钟
001：选择 RTC/8 时钟
010：选择 RTC/4 时钟
011：选择 RTC/2 时钟
10x：选择 ck_spre 时钟（通常为 1 Hz）
11x：选择 ck_spre 时钟（通常为 1 Hz）并将 WUT 计数器值增加 2^{16}（见下面的注释）

注： 只能在初始化模式下 (RTC_ISR/INITF = 1) 对该寄存器的位 7、6 和 4 执行写操作。

WUT = 唤醒单元计数器值。当 WUCKSEL[2:1 = 11] 时，WUT = (0x0000 to 0xFFFF) + 0x10000（增加的值）。

只能在 RTC_CR WUTE 位 = 0 且 RTC_ISR WUTWF 位 = 1 时对该寄存器的位 2 到 0 执行写操作。

建议不要在日历小时递增时更改小时，因为这样做会屏蔽日历小时的增量。

ADD1H 和 SUB1H 的更改在下一秒生效。

此寄存器受写保护。第 598 页的 RTC 寄存器写保护中介绍了写访问的过程。

27.6.4 RTC 初始化和状态寄存器 (RTC_ISR)

RTC initialization and status register

此寄存器受写保护（RTC_ISR[13:8] 位除外）。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x0C

RTC 域复位值：0x0000 0007

系统复位：不受影响（INIT、INITF 和 RSF 位除外，它们在复位时被清零）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

位 31:17 保留，必须保持复位值

位 16 **RECALPF**：重新校准挂起标志 (Recalibration pending Flag)
当软件对 RTC_CALR 寄存器执行写操作时，RECALPF 状态标志将自动置“1”，指示 RTC_CALR 寄存器已屏蔽。当采用新的校准设置时，该位恢复为“0”。请参见*动态重校准*。

位 15 **TAMP3F**：RTC_TAMP3 检测标志 (RTC_TAMP3 detection flag)（仅第 5 类器件）
在 RTC_TAMP3 输入上检测到入侵检测事件时，由硬件将此标志置 1。
该标志由软件写零清除

位 14 **TAMP2F**：RTC_TAMP2 检测标志 (RTC_TAMP2 detection flag)
在 RTC_TAMP2 输入上检测到入侵检测事件时，由硬件将此标志置 1。
该标志由软件写零清除

位 13 **TAMP1F**：RTC_TAMP1 检测标志 (RTC_TAMP1 detection flag)
在 RTC_TAMP1 输入上检测到入侵检测事件时，由硬件将此标志置 1。
该标志由软件写零清除

位 12 **TSOVF**：时间戳溢出标志 (Time-stamp overflow flag)
当在 TSF 已置 1 的情况下发生时间戳事件时，由硬件将此标志置 1。
该标志由软件写零清除。建议仅在 TSF 位清零之后再检查并清零 TSOVF 位。否则，如果时间戳事件恰好在清零 TSF 位之前刚刚发生，则溢出事件可能会被漏掉。

位 11 **TSF**：时间戳标志 (Time-stamp flag)
发生时间戳事件时，由硬件将此标志置 1。
该标志由软件写零清除。

位 10 **WUTF**：唤醒定时器标志 (Wakeup timer flag)
当唤醒自动重载计数器计数到 0 时，由硬件将此标志置 1。
该标志由软件写零清除。
软件必须在 WUTF 再次置 1 的 1.5 个 RTCCLK 周期之前将该标志清零。

位 9 **ALRBF**：闹钟 B 标志 (Alarm B flag)
当时间/日期寄存器（RTC_TR 和 RTC_DR）与闹钟 B 寄存器 (RTC_ALRMBR) 匹配时，由硬件将该标志置 1。
该标志由软件写零清除。

位 8 **ALRAF**：闹钟 A 标志 (Alarm A flag)

当时间/日期寄存器（RTC_TR 和 RTC_DR）与闹钟 A 寄存器 (RTC_ALRMAR) 匹配时，由硬件将该标志置 1。

该标志由软件写零清除。

位 7 **INIT**：初始化模式 (Initialization mode)

0：自由运行模式

1：初始化模式，用于编程时间和日期寄存器（RTC_TR 和 RTC_DR）以及预分频器寄存器 (RTC_PRER)。计数器停止计数，当 INIT 被复位后，计数器从新值开始计数。

位 6 **INITF**：初始化标志 (Initialization flag)

当此位置 1 时，RTC 处于初始化状态，此时可更新事件、日期和预分频器寄存器。

0：不允许更新日历寄存器

1：允许更新日历寄存器

位 5 **RSF**：寄存器同步标志 (Registers synchronization flag)

每次将日历寄存器的值复制到影子寄存器（RTC_SSRx、RTC_TRx 和 RTC_DRx）时，都会由硬件将此位置 1。在初始化模式下、平移操作挂起时 (SHPF=1) 或在旁路影子寄存器模式 (BYPSHAD=1) 下，该位由硬件清零。该位还可由软件清零。

在初始化模式下，该位可由软件或硬件清零。

0：日历影子寄存器尚未同步

1：日历影子寄存器已同步

位 4 **INITS**：初始化状态标志 (Initialization status flag)

当日历年份字段不为 0 时（RTC域复位状态），由硬件将该位置 1。

0：日历尚未初始化

1：日历已经初始化

位 3 **SHPF**：平移操作挂起 (Shift operation pending)

0：没有平移操作挂起

1：某个平移操作挂起

只要通过对 RTC_SHIFTR 寄存器执行写操作来启动平移操作，此标志便由硬件置 1。执行完相应的平移操作后，此标志由硬件清零。对 SHPF 位执行写入操作不起作用。

位 2 **WUTWF**：唤醒定时器写标志 (Wakeup timer write flag)

此位在 RTC_CR 中的 WUTE 位清零 2 个 RTCCLK 周期后由硬件置 1，在 WUTE 位置 1 后 2 个 RTCCLK 周期后清零。当 WUTE 位清零且 WUTWF 位置 1 时，可更改唤醒定时器的值。

0：不允许更新唤醒定时器配置

1：允许更新唤醒定时器配置

位 1 **ALRBWF**：闹钟 B 写标志 (Alarm B write flag)

在 RTC_CR 寄存器中的 ALRBE 位置 0 之后，当闹钟 B 的值可更改时，由硬件将该位置 1。该位在初始化模式下由硬件清零。

0：不允许更新闹钟 B

1：允许更新闹钟 B

位 0 **ALRAWF**：闹钟 A 写标志 (Alarm A write flag)

在 RTC_CR 寄存器中的 ALRAE 位置 0 后，当闹钟 A 的值可更改时，由硬件将该位置 1。

该位在初始化模式下由硬件清零。

0：不允许更新闹钟 A

1：允许更新闹钟 A

注： *将 ALRAF、ALRBF、WUTF 和 TSF 位编程为 0 之后 2 个 APB 时钟周期，清零生效。*

27.6.5 RTC 预分频器寄存器 (RTC_PRER)

RTC prescaler register

只能在初始化模式下对该寄存器执行写操作。必须通过两次独立的写访问执行初始化。请参见*第 599 页的日历初始化和配置*。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x10

RTC域复位值：0x007F 00FF

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

位 31:23 保留，必须保持复位值

位 22:16 **PREDIV_A[6:0]**：异步预分频系数 (Asynchronous prescaler factor)
下面是异步分频系数的公式：
ck_apre 频率 = RTCCLK 频率/(PREDIV_A+1)

位 15 保留，必须保持复位值。

位 14:0 **PREDIV_S[14:0]**：同步预分频系数 (Synchronous prescaler factor)
下面是同步分频系数的公式：
ck_spre 频率 = ck_apre 频率/(PREDIV_S+1)

27.6.6 RTC 唤醒定时器寄存器 (RTC_WUTR)

RTC wakeup timer register

仅当 RTC_ISR 中的 WUTWF 置 1 时才可对该寄存器执行写操作。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x14

RTC域复位值：0x0000 FFFF

系统复位：不受影响

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

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

位 31:16 保留，必须保持复位值

位 15:0 **WUT[15:0]**：唤醒自动重载值位 (Wakeup auto-reload value bits)

当使能唤醒定时器时（WUTE 置 1），每 (WUT[15:0] + 1) 个 ck_wut 周期将 WUTF 标志置 1 一次。ck_wut 周期通过 RTC_CR 寄存器的 WUCKSEL[2:0] 位进行选择。

当 WUCKSEL[2] = 1 时，唤醒定时器变为 17 位，WUCKSEL[1] 等效为 WUT[16]，即要重载到定时器的最高有效位。

WUTF 第一次置 1 发生在 WUTE 置 1 之后 (WUT+1) 个 ck_wut 周期。禁止在 WUCKSEL[2:0]=011(RTCCLK/2) 时将 WUT[15:0] 设置为 0x0000。

27.6.7 RTC 闹钟 A 寄存器 (RTC_ALRMAR)

RTC alarm A register

仅当 RTC_ISR 中的 ALRAWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x1C

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **MSK4**：闹钟 A 日期掩码 (Alarm A date mask)
0：如果日期/日匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，日期/日无关

位 30 **WDSEL**：星期几选择 (Week day selection)
0：DU[3:0] 代表日期的个位
1：DU[3:0] 代表星期几 DT[1:0] 为无关位。

位 29:28 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)。

位 27:24 **DU[3:0]**：日期的个位或日（BCD 格式）(Date units or day in BCD format)。

位 23 **MSK3**：闹钟 A 小时掩码 (Alarm A hours mask)
0：如果小时匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，小时无关

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)。

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)。

位 15 **MSK2**：闹钟 A 分钟掩码 (Alarm A minutes mask)
0：如果分钟匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，分钟无关

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)。

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)。

位 7 **MSK1**：闹钟 A 秒掩码 (Alarm A seconds mask)
0：如果秒匹配，则闹钟 A 置 1
1：在闹钟 A 比较中，秒无关

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)。

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)。

27.6.8 RTC 闹钟 B 寄存器 (RTC_ALRMBR)

RTC alarm B register

仅当 RTC_ISR 中的 ALRBWF 置 1 时或在初始化模式下，才可以对该寄存器执行写操作。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x20

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **MSK4**：闹钟 B 日期掩码 (Alarm B date mask)
0：如果日期和日匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，日期和日无关

位 30 **WDSEL**：星期几选择 (Week day selection)
0：DU[3:0] 代表日期的个位
1：DU[3:0] 代表星期几 DT[1:0] 为无关位。

位 29:28 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 27:24 **DU[3:0]**：日期个位或日（BCD 格式）(Date units or day in BCD format)

位 23 **MSK3**：闹钟 B 小时掩码 (Alarm B hours mask)
0：如果小时匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，小时无关

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)

位 15 **MSK2**：闹钟 B 分钟掩码 (Alarm B minutes mask)
0：如果分钟匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，分钟无关

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)

位 7 **MSK1**：闹钟 B 秒掩码 (Alarm B seconds mask)
0：如果秒匹配，则闹钟 B 置 1
1：在闹钟 B 比较中，秒无关

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)

27.6.9 RTC 写保护寄存器 (RTC_WPR)

RTC write protection register

偏移地址：0x24

复位值：0x0000 0000

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

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

位 31:8 保留，必须保持复位值。

位 7:0 **KEY**：写保护关键字 (Write protection key)
可通过软件对该字节执行写操作。
读取该字节时，始终返回 0x00。
有关如何解锁 RTC 寄存器写保护的介绍，请参见*RTC 寄存器写保护*。

27.6.10 RTC 亚秒寄存器 (RTC_SSR)

RTC sub second register

偏移地址：0x28

RTC域复位值：0x0000 0000

系统复位：当 BYPSHAD = 0 时为 0x0000 0000。当 BYPSHAD = 1 时不受影响。

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

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

位 31:16 保留，必须保持复位值

位 15:0 **SS**：亚秒值 (Sub seconds value)
SS[15:0] 是同步预分频器计数器的值。此亚秒值可根据以下公式得出：
亚秒值 = (PREDIV_S - SS) / (PREDIV_S + 1)
注： 仅当执行平移操作之后，SS 才能大于 PREDIV_S。在这种情况下，正确的时间/日期比 RTC_TR/RTC_DR 所指示的时间/日期慢一秒钟。

27.6.11 RTC 平移控制寄存器 (RTC_SHIFTR)

RTC shift control register

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x2C

RTC 域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

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

位 31 **ADD1S**：增加一秒钟 (Add one second)

0：无任何作用

1：对时钟/日历增加一秒钟

此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。

此函数应与 SUBFS 配合使用（请参见下文介绍），以便有效地向原子操作机制的时钟添加亚秒值。

位 30:15 保留，必须保持复位值

位 14:0 **SUBFS**：减少亚秒值 (Subtract a fraction of a second)

此位为只写位且始终读为 0。当平移操作挂起（RTC_ISR 中的 SHPF=1）时，对该位执行写操作无作用。

写入 SUBFS 的值将加到同步预分频器计数器中。由于该计数器递减计数，此操作可有效地从时钟减去（延迟）以下时间：

延迟（秒）= SUBFS / (PREDIV_S + 1)

当 ADD1S 函数与 SUBFS 结合使用时，可有效地将亚秒值增加到时钟（提前时钟），使时钟提前以下时间：

提前（秒）= (1 - (SUBFS / (PREDIV_S + 1)))。

注：对 SUBFS 执行写操作将使 RSF 清零。软件随后会等待至 RSF=1 以确定影子寄存器已更新为平移后的时间。

27.6.12 RTC 时间戳时间寄存器 (RTC_TSTR)

RTC timestamp time register

仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

偏移地址：0x30

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

位 31:23 保留，必须保持复位值

位 22 **PM**：AM/PM 符号 (AM/PM notation)
0：AM 或 24 小时制
1：PM

位 21:20 **HT[1:0]**：小时的十位（BCD 格式）(Hour tens in BCD format)。

位 19:16 **HU[3:0]**：小时的个位（BCD 格式）(Hour units in BCD format)。

位 15 保留，必须保持复位值

位 14:12 **MNT[2:0]**：分钟的十位（BCD 格式）(Minute tens in BCD format)。

位 11:8 **MNU[3:0]**：分钟的个位（BCD 格式）(Minute units in BCD format)。

位 7 保留，必须保持复位值

位 6:4 **ST[2:0]**：秒的十位（BCD 格式）(Second tens in BCD format)。

位 3:0 **SU[3:0]**：秒的个位（BCD 格式）(Second units in BCD format)。

27.6.13 RTC 时间戳日期寄存器 (RTC_TSDR)

RTC timestamp date register

仅当 RTC_ISR 中的 TSF 置 1 时，该寄存器的内容才有效。当 TSF 位复位时，清零该寄存器。

偏移地址：0x34

RTC域复位值：0x0000 0000

系统复位：不受影响

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

位 31:16 保留，必须保持复位值

位 15:13 **WDU[1:0]**：星期几的个位 (Week day units)

位 12 **MT**：月份的十位（BCD 格式）(Month tens in BCD format)

位 11:8 **MU[3:0]**：月份的个位（BCD 格式）(Month units in BCD format)

位 7:6 保留，必须保持复位值

位 5:4 **DT[1:0]**：日期的十位（BCD 格式）(Date tens in BCD format)

位 3:0 **DU[3:0]**：日期的个位（BCD 格式）(Date units in BCD format)

27.6.14 RTC 时间戳亚秒寄存器 (RTC_TSSSR)

RTC time-stamp sub second register

仅当 RTC_ISR/TSF 置 1 时，该寄存器的内容才有效。当 RTC_ISR/TSF 位复位时，清零该寄存器。

偏移地址：0x38

RTC域复位值：0x0000 0000

系统复位：不受影响

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

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

位 31:16 保留，必须保持复位值

位 15:0 **SS**：亚秒值 (Sub seconds value)

当发生时间戳事件时，SS[15:0] 是同步预分频器计数器的值。

27.6.15 RTC 校准寄存器 (RTC_CALR)

RTC calibration register

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x3C

RTC 域复位值：0x0000 0000

系统复位：不受影响

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16 保留，必须保持复位值

位 15 **CALP**：将 RTC 的频率增加 488.5 ppm (Increase frequency of RTC by 488.5 ppm)

0：不增加 RTCCLK 脉冲。

1：每 2^{11} 个脉冲有效插入一个 RTCCLK 脉冲（将频率增加 488.5 ppm）。

此功能应与 CALM 结合使用，后者在高分辨率下会降低日历的频率。如果输入频率为 32768 Hz，则在 32 秒窗口中增加的 RTCCLK 脉冲数按如下公式计算：(512 * CALP) - CALM。请参见*第 27.3.12 节：RTC 精密数字校准*。

位 14 **CALW8**：使用 8 秒校准周期 (Use an 8-second calibration cycle period)

当 CALW8 置“1”时，选择 8 秒校准周期。

注： CALW8=“1”时，CALM[1:0] 将始终保持为“00”。请参见第 27.3.12 节：RTC 精密数字校准。

位 13 **CALW16**：使用 16 秒校准周期 (Use a 16-second calibration cycle period)

当 CALW16 置“1”时，选择 16 秒校准周期。如果 CALW8 = 1，则不能将该位置“1”。

注： 当 CALW16=“1”时，CALM[0] 将始终保持为“0”。请参见第 27.3.12 节：RTC 精密数字校准。

位 12:9 保留，必须保持复位值

位 8:0 **CALM[8:0]**：负校准 (Calibration minus)

在 2^{20} 个 RTCCLK 脉冲内屏蔽 CALM 个脉冲（如果输入频率为 32768 Hz，则为 32 秒）来降低日历的频率。其分辨率为 0.9537 ppm。

要提高日历的频率，则应将此功能与 CALP 结合使用。请参见*第 602 页的第 27.3.12 节：RTC 精密数字校准*。

27.6.16 RTC 入侵配置寄存器 (RTC_TAMPCR)

RTC tamper configuration register

偏移地址：0x40

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 MF	TAMP3 NO ERASE	TAMP3 IE	TAMP2 MF	TAMP2 NO ERASE	TAMP2 IE	TAMP1 MF	TAMP1 NO ERASE	TAMP1 IE
							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
TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMP TS	TAMP3 TRG	TAMP3 E	TAMP2 TRG	TAMP2 E	TAMPI E	TAMP1 TRG	TAMP1 E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:25 保留，必须保持复位值。

位 24 **TAMP3MF**：入侵 3 屏蔽标志 (Tamper 3 mask flag)（仅第 5 类器件）

0：入侵 3 事件生成触发事件，必须通过软件将 TAMP3F 清零来允许下一次入侵事件检测。

1：入侵 3 事件生成触发事件。屏蔽 TAMP3F 并由硬件在内部将其清零。不擦除备份寄存器。

注： TAMP3MF 置 1 时，不得使能入侵 3 中断。

位 23 **TAMP3NOERASE**：入侵 3 不擦除 (Tamper 3 no erase)（仅第 5 类器件）

0：入侵 3 事件擦除备份寄存器。

1：入侵 3 事件不擦除备份寄存器。

位 22 **TAMP3IE**：入侵 3 中断使能 (Tamper 3 interrupt enable)（仅第 5 类器件）

0：TAMPIE = 0 时禁止入侵 3 中断。

1：使能入侵 3 中断。

位 21 **TAMP2MF**：入侵 2 屏蔽标志 (Tamper 2 mask flag)

0：入侵 2 事件生成触发事件，如果要允许下一次入侵事件检测，必须通过软件将 TAMP2F 清零。

1：入侵 2 事件生成触发事件。屏蔽 TAMP2F 并由硬件在内部将其清零。不擦除备份寄存器。

注： TAMP2MF 置 1 时，不得使能入侵 2 中断。

位 20 **TAMP2NOERASE**：入侵 2 不擦除 (Tamper 2 no erase)

0：入侵 2 事件擦除备份寄存器。

1：入侵 2 事件不擦除备份寄存器。

位 19 **TAMP2IE**：入侵 2 中断使能 (Tamper 2 interrupt enable)

0：TAMPIE = 0 时禁止入侵 2 中断。

1：使能入侵 2 中断。

位 18 **TAMP1MF**：入侵 1 屏蔽标志 (Tamper 1 mask flag)

0：入侵 1 事件生成触发事件，必须通过软件将 TAMP1F 清零来允许下一次入侵事件检测。

1：入侵 1 事件生成触发事件。屏蔽 TAMP1F 并由硬件在内部将其清零。不擦除备份寄存器。

注： TAMP1MF 置 1 时，不得使能入侵 1 中断。

位 17 **TAMP1NOERASE**：入侵 1 不擦除 (Tamper 1 no erase)

0：入侵 1 事件擦除备份寄存器。

1：入侵 1 事件不擦除备份寄存器。

位 16 **TAMP1IE**：入侵 1 中断使能 (Tamper 1 interrupt enable)

0：TAMPIE = 0 时禁止入侵 1 中断。

1：使能入侵 1 中断。

位 15 **TAMPPUDIS**：RTC_TAMPx 上拉禁止 (RTC_TAMPx pull-up disable)

该位决定在每次采样之前是否对每个 RTC_TAMPx 引脚都进行预充电。

0：采样之前对 RTC_TAMPx 引脚进行预充电（使能内部上拉）

1：禁止对 RTC_TAMPx 引脚进行预充电。

位 14:13 **TAMPPRCH[1:0]**：RTC_TAMPx 预充电持续时间 (RTC_TAMPx precharge duration)

这些位决定了在每次采样之前激活上拉的持续时间。TAMPPRCH 对每个 RTC_TAMPx 输入都有效。

0x0：1 个 RTCCLK 周期

0x1：2 个 RTCCLK 周期

0x2：4 个 RTCCLK 周期

0x3：8 个 RTCCLK 周期

位 12:11 **TAMPFLT[1:0]**：RTC_TAMPx 过滤器计数 (RTC_TAMPx filter count)

这些位决定了为激活入侵事件所需的指定电平 (TAMP*TRG) 上的连续采样次数。TAMPFLT 对每个 RTC_TAMPx 输入都有效。

0x0：在 RTC_TAMPx 输入转变为有效电平的边沿激活入侵事件（RTC_TAMPx 输入上无内部上拉）。

0x1：在有效电平上连续执行 2 次采样后激活入侵事件。

0x2：在有效电平上连续执行 4 次采样后激活入侵事件。

0x3：在有效电平上连续执行 8 次采样后激活入侵事件。

位 10:8 **TAMPFREQ[2:0]**：入侵采样频率 (Tamper sampling frequency)

决定对每个 RTC_TAMPx 输入进行采样时的频率。

0x0：RTCCLK/32768（RTCCLK = 32768 Hz 时为 1 Hz）

0x1：RTCCLK/16384（RTCCLK = 32768 Hz 时为 2 Hz）

0x2：RTCCLK/8192（RTCCLK = 32768 Hz 时为 4 Hz）

0x3：RTCCLK/4096（RTCCLK = 32768 Hz 时为 8 Hz）

0x4：RTCCLK/2048（RTCCLK = 32768 Hz 时为 16 Hz）

0x5：RTCCLK/1024（RTCCLK = 32768 Hz 时为 32 Hz）

0x6：RTCCLK/512（RTCCLK = 32768 Hz 时为 64 Hz）

0x7：RTCCLK/256（RTCCLK = 32768 Hz 时为 128 Hz）

位 7 **TAMPTS**：发生入侵检测事件时激活时间戳 (Activate timestamp on tamper detection event)

0：发生入侵检测事件时不保存时间戳

1：发生入侵检测事件时保存时间戳

即便 RTC_CR 寄存器中的 TSE=0，TAMPTS 仍有效。

位 6 **TAMP3TRG**：RTC_TAMP3 输入的有效电平 (Active level for RTC_TAMP3 input)（仅第 5 类器件）

如果 TAMPFLT ≠ 00：

0：RTC_TAMP3 输入保持低电平会触发入侵检测事件。

1：RTC_TAMP3 输入保持高电平会触发入侵检测事件。

如果 TAMPFLT = 00：

0：RTC_TAMP3 输入上升沿会触发入侵检测事件。

1：RTC_TAMP3 输入下降沿会触发入侵检测事件。

位 5 **TAMP3E**：RTC_TAMP3 检测使能 (RTC_TAMP3 detection enable)（仅第 5 类器件）

0：禁止 RTC_TAMP3 输入检测

1：使能 RTC_TAMP3 输入检测

位 4 **TAMP2TRG**：RTC_TAMP2 输入的有效电平 (Active level for RTC_TAMP2 input)

如果 TAMPFLT != 00：

0：RTC_TAMP2 输入保持低电平会触发入侵检测事件。

1：RTC_TAMP2 输入保持高电平会触发入侵检测事件。

如果 TAMPFLT = 00：

0：RTC_TAMP2 输入上升沿会触发入侵检测事件。

1：RTC_TAMP2 输入下降沿会触发入侵检测事件。

位 3 **TAMP2E**：RTC_TAMP2 输入检测使能 (RTC_TAMP2 input detection enable)

0：禁止 RTC_TAMP2 检测

1：使能 RTC_TAMP2 检测

位 2 **TAMPIE**：入侵中断使能 (Tamper interrupt enable)

0：禁止入侵中断

1：使能入侵中断

注：　该位使能所有入侵引脚事件的中断，无论 TAMPxIE 电平如何。如果将该位清零，可以通过将 TAMPxIE 置 1 分别使能每个入侵事件中断。

位 1 **TAMP1TRG**：RTC_TAMP1 输入的有效电平 (Active level for RTC_TAMP1 input)

如果 TAMPFLT != 00

0：RTC_TAMP1 输入保持低电平会触发入侵检测事件。

1：RTC_TAMP1 输入保持高电平会触发入侵检测事件。

如果 TAMPFLT = 00：

0：RTC_TAMP1 输入上升沿会触发入侵检测事件。

1：RTC_TAMP1 输入下降沿会触发入侵检测事件。

位 0 **TAMP1E**：RTC_TAMP1 输入检测使能 (RTC_TAMP1 input detection enable)

0：禁止 RTC_TAMP1 检测

1：使能 RTC_TAMP1 检测

注意： 如果 TAMPFLT = 0，则更改 TAMP1TRG 时必须复位 TAMP1E，以避免将 TAMP1F 意外置 1。

27.6.17 RTC 闹钟 A 亚秒寄存器 (RTC_ALRMASSR)

RTC alarm A sub second register

仅当 RTC_CR 中的 ALRAE 复位时或在初始化模式下，才可以对该寄存器执行写操作。

此寄存器受写保护。*第 598 页的 RTC 寄存器写保护*中介绍了写访问的过程。

偏移地址：0x44

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

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

位 31:28 保留，必须保持复位值。

位 27:24 MASKSS[3:0]：屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit)

0：不对闹钟 A 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。
1：在闹钟 A 比较中，SS[14:1] 为无关位。仅比较 SS[0]。
2：在闹钟 A 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。
3：在闹钟 A 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。
...
12：在闹钟 A 比较中，SS[14:12] 为无关位。比较 SS[11:0]。
13：在闹钟 A 比较中，SS[14:13] 为无关位。比较 SS[12:0]。
14：在闹钟 A 比较中，SS[14] 为无关位。比较 SS[13:0]。
15：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。
同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。

位 23:15 保留，必须保持复位值。

位 14:0 SS[14:0]：亚秒值 (Sub seconds value)

该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 A。仅比较位 0 到 MASKSS-1。

27.6.18 RTC 闹钟 B 亚秒寄存器 (RTC_ALRMBSSR)

RTC alarm B sub second register

仅当 RTC_CR 中的 ALRBE 复位时或在初始化模式下，才可以对该寄存器执行写操作。

该寄存器受写保护。*RTC 寄存器写保护*一节中介绍了写访问的过程。

偏移地址：0x48

RTC域复位值：0x0000 0000

系统复位：不受影响

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

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

位 31:28 保留，必须保持复位值。

位 27:24 **MASKSS[3:0]**：屏蔽从此位开始的最高有效位 (Mask the most-significant bits starting at this bit)

0x0：不对闹钟 B 的亚秒进行比较。当秒单元递增时设置闹钟（假定其余字段均匹配）。
0x1：在闹钟 B 比较中，SS[14:1] 为无关位。仅比较 SS[0]。
0x2：在闹钟 B 比较中，SS[14:2] 为无关位。仅比较 SS[1:0]。
0x3：在闹钟 B 比较中，SS[14:3] 为无关位。仅比较 SS[2:0]。
...
0xC：在闹钟 B 比较中，SS[14:12] 为无关位。比较 SS[11:0]。
0xD：在闹钟 B 比较中，SS[14:13] 为无关位。比较 SS[12:0]。
0xE：在闹钟 B 比较中，SS[14] 为无关位。比较 SS[13:0]。
0xF：所有 15 个 SS 位均进行比较，并且必须全部匹配才能激活闹钟。
同步计数器的溢出位（位 15）从不进行比较。仅当执行平移操作之后，此位才不为 0。

位 23:15 保留，必须保持复位值。

位 14:0 **SS[14:0]**：亚秒值 (Sub seconds value)

该值与同步预分频器计数器的内容进行比较以确定是否要激活闹钟 B。仅比较位 0 到 MASKSS-1。

27.6.19 RTC 选项寄存器 (RTC_OR)

RTC option register

偏移地址：0x4C

RTC域复位值：0x0000 0000

系统复位：不受影响

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

位 31:2 保留，必须保持复位值。

位 1 **RTC_OUT_RMP**：RTC_OUT 重映射 (RTC_OUT remap)

将此位置 1 可将 RTC 输出重映射到 PB14，具体如下：

RTC_OUT_RMP = “0”：

如果 OSEL/=“00”：RTC_ALARM 输出到 PC13

如果 OSEL=“00”且 COE =“1”：RTC_CALIB 输出到 PC13

RTC_OUT_RMP = “1”：

如果 OSEL /=“00”且 COE =“0”：RTC_ALARM 输出到 PB14

如果 OSEL =“00”且 COE =“1”：RTC_CALIB 输出到 PB14

如果 OSEL /=“00”且 COE =“1”：RTC_CALIB 输出到 PB14 且 RTC_ALARM 输出到 PC13。

位 0 **RTC_ALARM_TYPE**：PC13 上的 RTC_ALARM 输出类型 (RTC_ALARM output type on PC13)

该位由软件置 1 和清零

0：RTC_ALARM 在映射到 PC13 上时为开漏输出

1：RTC_ALARM 在映射到 PC13 上时为推挽输出

27.6.20 RTC 备份寄存器 (RTC_BKPxR)

RTC backup registers

偏移地址：0x50 到 0x60

RTC 域复位值：0x0000 0000

系统复位：不受影响

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

位 31:0 BKP[31:0]

应用可向/从这些寄存器写入/读取数据。

当器件以低功耗模式工作时，这些寄存器的内容保持有效。

发生入侵检测事件时该寄存器会被复位，并且只要 TAMPxF=1，该寄存器就一直保持复位。或者在禁止闪存读出保护时复位。

27.6.21 RTC 寄存器映射

表 107. RTC 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
	Reset value									0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C	RTC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRBWF	ALRAWF
	Reset value																0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	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	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0

表 107. RTC 寄存器映射和复位值（续）

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
0x38	**RTC_TSSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0
0x40	**RTC_TAMPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP3NOERASE	TAMP3IE	TAMP2MF	TAMP2NOERASE	TAMP2IE	TAMP1MF	TAMP1NOERASE	TAMP1IE	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		TAMPTS	TAMP3-TRG	TAMP3E	TAMP2-TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**RTC_ ALRMASSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ ALRMBSSR**	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RTC_ OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_OUT_RMP	RTC_ALARM_TYPE
	Reset value																															0	0
0x50 to 0x60	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	to **RTC_BKP4R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

28 内部集成电路 (I2C) 接口

28.1 简介

I^2C（内部集成电路）总线接口处理微控制器与串行 I^2C 总线间的通信。它提供多主模式功能，可以控制所有 I^2C 总线特定的序列、协议、仲裁和时序。它支持标准模式 (Sm)、快速模式 (Fm) 和超快速模式 (Fm+)。

它还与 SMBus（系统管理总线）和 PMBus（电源管理总线）兼容。

可使用 DMA 来减轻 CPU 的工作量。

28.2 I2C 主要特性

- 兼容 I^2C 总线规范第 03 版：
  - 从模式和主模式
  - 多主模式功能
  - 标准速度模式（高达 100 kHz）
  - 快速模式（高达 400 kHz）
  - 超快速模式（高达 1 MHz）
  - 7 位和 10 位寻址模式
  - 多个 7 位从地址（2 个从设备地址寄存器，1 个具有可配置的掩码位段）
  - 所有 7 位地址应答模式
  - 广播呼叫
  - 总线上的数据建立和保持时间可软件配置
  - 方便易用的事件管理
  - 可选的时钟延长
  - 软件复位
- 带 DMA 功能的 1 字节缓冲
- 可编程模拟和数字噪声滤波器

还可额外提供以下特性，具体取决于产品实现（请参见*第 28.3 节：I2C 特性实现*）：

- 兼容 SMBus 规范第 2.0 版：
  - 具有 ACK 控制的硬件 PEC（数据包错误校验）生成和验证
  - 命令和数据应答控制
  - 支持地址解析协议 (ARP)
  - 支持主机和从设备
  - SMBus 报警
  - 超时和空闲条件检测
- 兼容 PMBus 第 1.1 版标准
- 独立时钟：选择独立时钟源可使 I2C 通信速度不受 PCLK 时钟频率更改的影响
- 地址匹配时从停止模式唤醒

28.3 I2C 特性实现

本手册介绍了 I2C1 和 I2C3 中实现的所有特性。I2C2 支持的特性较少，但在其余方面与 I2C1/I2C3 完全相同。区别如下表所示。

表 108. STM32L0x3 I2C 特性

I2C特性(1)	I2C1	I2C2	I2C3
独立时钟	X	-	X
SMBus	X	-	X
从停止模式唤醒	X	-	X
Fm+ 模式下的 20 mA 输出驱动器(2)	X	X	X

1. X = 支持。
2. 有关支持该特性的 I/O 的列表，请参见数据手册。

28.4 I2C 功能说明

除了接收和发送数据之外，此接口还可以从串行格式转换为并行格式，反之亦然。中断由软件使能或禁止。该接口通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 I^2C 总线。它可以连接到标准速度（高达 100 kHz）、快速（高达 400 kHz）或超快速（高达 1 MHz）I^2C 总线。

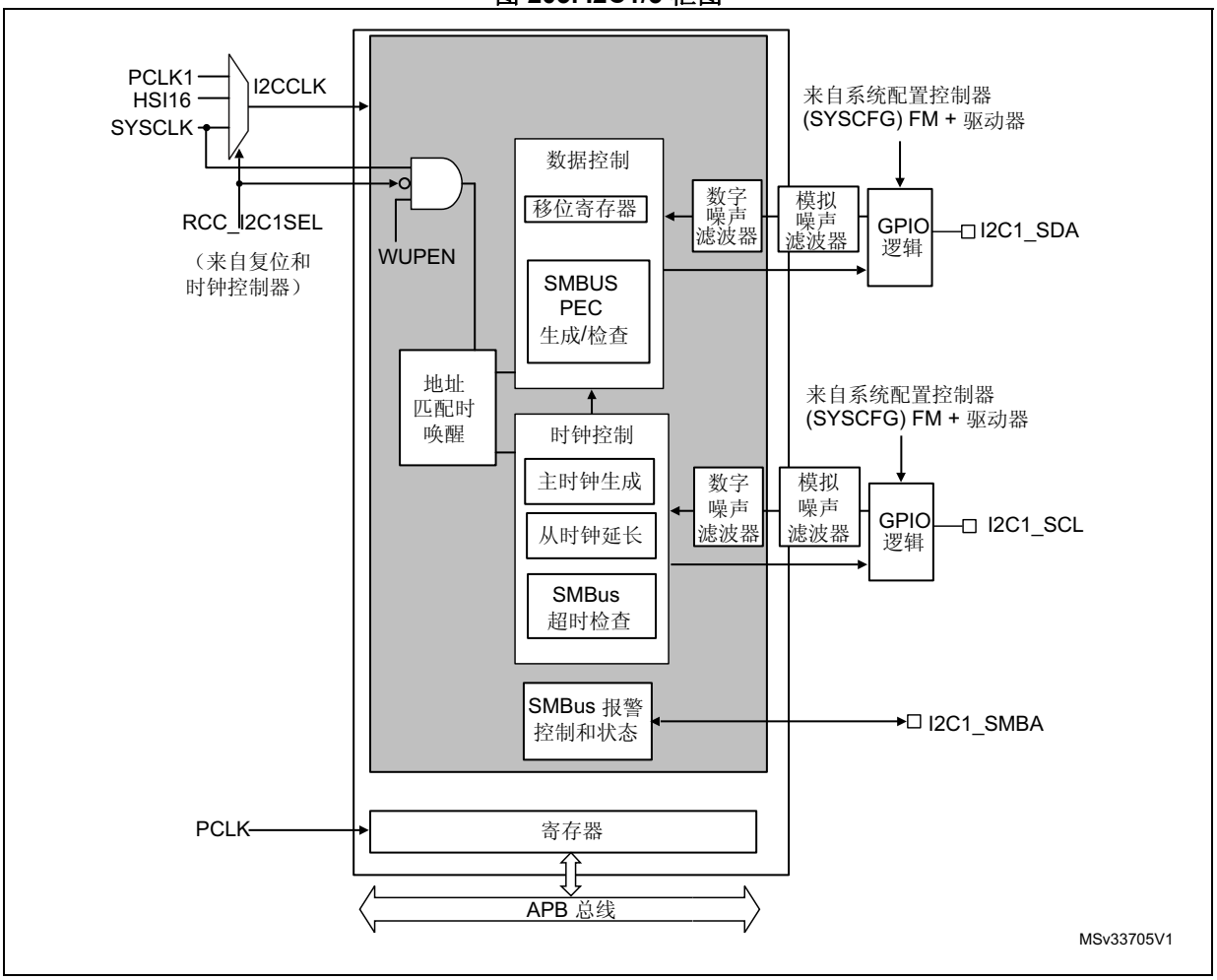
该接口也可通过数据引脚 (SDA) 和时钟引脚 (SCL) 连接到 SMBus。

如果支持 SMBus 功能：还可使用额外的可选 SMBus 报警引脚 (SMBA)。

28.4.1 I2C1/3 框图

I2C1 接口的框图如*图 203* 所示。

图 203. I2C1/3 框图



I2C1/3 的时钟由独立时钟源提供，这使得 I2C 能够独立于 PCLK 频率工作。

该独立时钟源可从以下三种时钟源中任选其一：

- PCLK1：APB1 时钟（默认值）
- HSI16：内部 16 MHz RC 振荡器
- SYSCLK：系统时钟

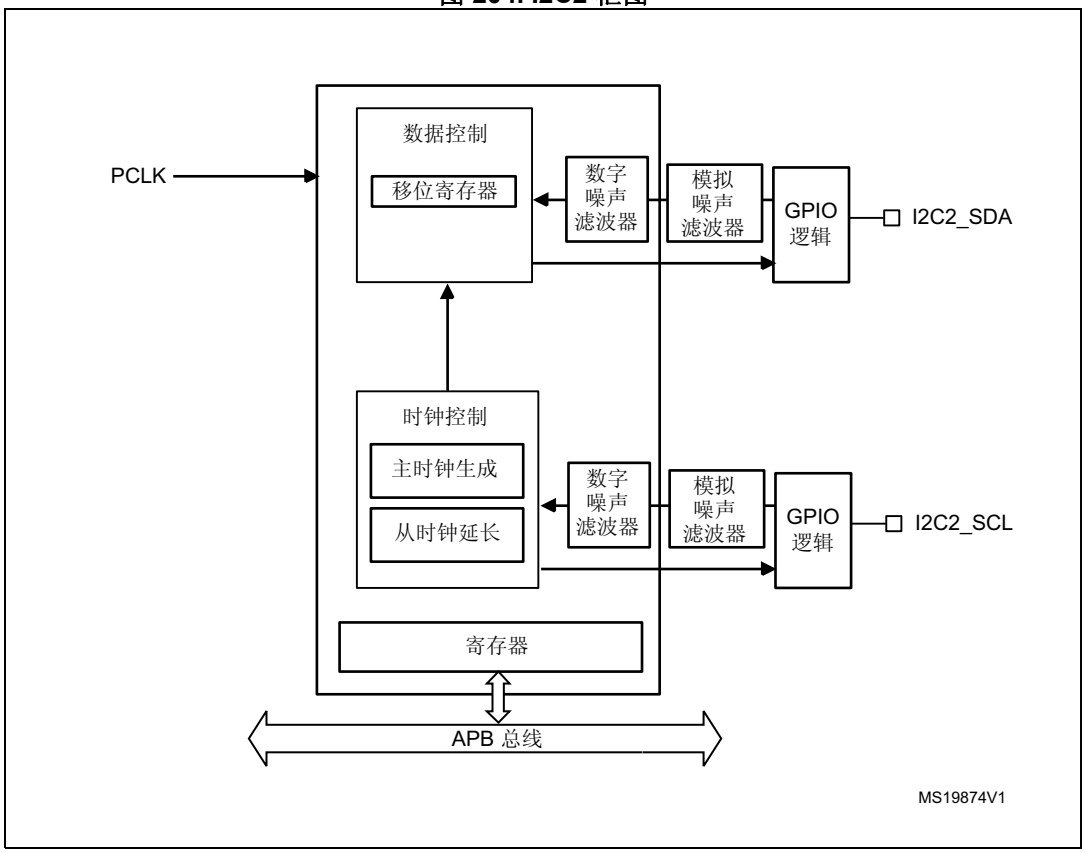
更多详细信息，请参见*第 7 节：复位和时钟控制 (RCC)*。

I2C1/3 I/O 支持 20 mA 输出电流驱动器，用于驱动超快速操作。将 SCL 和 SDA 的驱动能力控制位置 1 可使能该功能，这些位位于*第 10.2.2 节：SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)*。

28.4.2 I2C2 框图

I2C2 接口的框图如*图 204* 所示。

图 204. I2C2 框图



28.4.3 I2C 时钟要求

I2C 内核的时钟由 I2CCLK 提供。

I2CCLK 周期 t_{I2CCLK} 必须遵循以下条件：

$t_{I2CCLK} < (tLOW - tfilters) / 4$ 且 $t_{I2CCLK} < tHIGH$

其中：

t_{LOW}：SCL 低电平时间；t_{HIGH}：SCL 高电平时间

$t_{filters}$：滤波器使能时，该值为模拟滤波器和数字滤波器引入的延时总和。

模拟滤波器延时最大值为 260 ns。数字滤波器延时为 DNF x t_{I2CCLK}。

PCLK 时钟周期 t_{PCLK} 必须遵循以下条件：

$t_{PCLK} < 4/3\ tSCL$

其中，t_{SCL}：SCL 周期

注意： 当 I2C 内核的时钟由 PCLK 提供时，PCLK 必须遵循 t_{I2CCLK} 的条件。

28.4.4 模式选择

该接口在工作时可选用以下四种模式之一：

- 从发送器
- 从接收器
- 主发送器
- 主接收器

默认情况下，它以从模式工作。接口在生成起始位后会自动由从模式切换为主模式，并在出现仲裁丢失或生成停止位时从主模式切换为从模式，从而实现多主模式功能。

通信流程

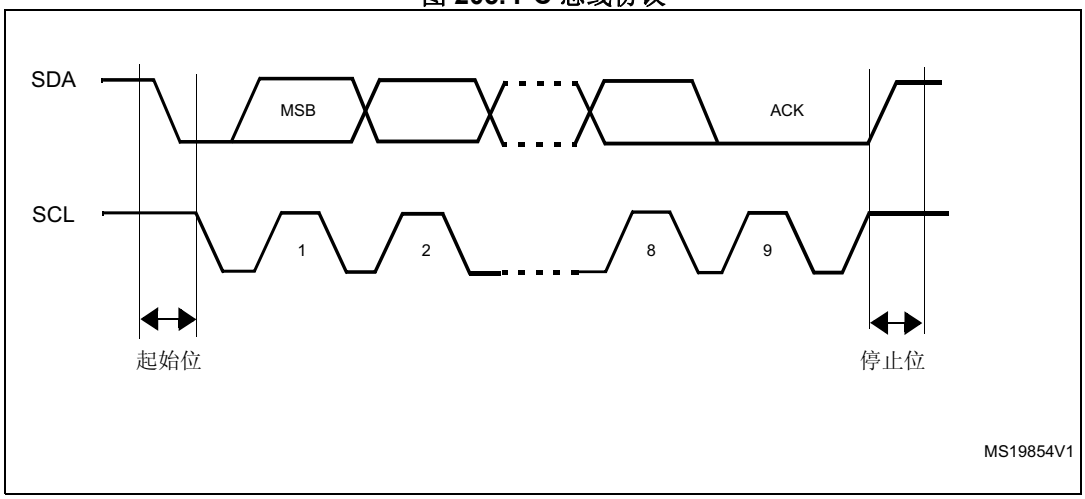
在主模式下，I2C 接口会启动数据传输并生成时钟信号。串行数据传输始终是在出现起始位时开始，在出现停止位时结束。起始位和停止位均在主模式下由软件生成。

在从模式下，该接口能够识别其自身地址（7 或 10 位）以及广播呼叫地址。广播呼叫地址检测可由软件使能或禁止。保留的 SMBus 地址也可由软件使能。

数据和地址均以 8 位字节传输，MSB 在前。起始位后紧随地址字节（7 位地址占据一个字节；10 位地址占据两个字节）。地址始终在主模式下传送。

在字节传输 8 个时钟周期后是第 9 个时钟脉冲，在此期间接收器必须向发送器发送一个应答位。请参见下图。

图 205. I²C 总线协议



应答位可由软件使能或禁止。I2C 接口地址可通过软件进行选择。

28.4.5 I2C 初始化

使能和禁止外设

I2C 外设时钟必须在时钟控制器中进行配置和使能（请参见*第 7 节：复位和时钟控制 (RCC)*）。

然后可通过将 I2C_CR1 寄存器中的 PE 位置 1 使能 I2C。

当禁止 I2C (PE=0) 时，I^2C 将执行软件复位。更多详细信息，请参见*第 28.4.6 节：软件复位*。

噪声滤波器

通过将 I2C_CR1 寄存器中的 PE 位置 1 来使能 I2C 外设之前，如有必要，用户必须配置噪声滤波器。默认情况下，SDA 和 SCL 输入上集成了模拟噪声滤波器。该模拟滤波器符合 I^2C 规范，此规范要求在快速模式和超快速模式下对脉宽在 50ns 以下的脉冲都要抑制。用户可通过将 ANFOFF 位置 1 来禁止该模拟滤波器，和/或通过配置 I2C_CR1 寄存器中的 DNF[3:0] 位来选择数字滤波器。

使能数字滤波器时，SCL 或 SDA 线的电平只有在电平稳定时间超过 DNF x I2CCLK 个周期后才会发生内部变化，从而可抑制的尖峰脉宽在 1 到 15 个 I2CCLK 周期可编程。

表 109. 模拟滤波器与数字滤波器对比

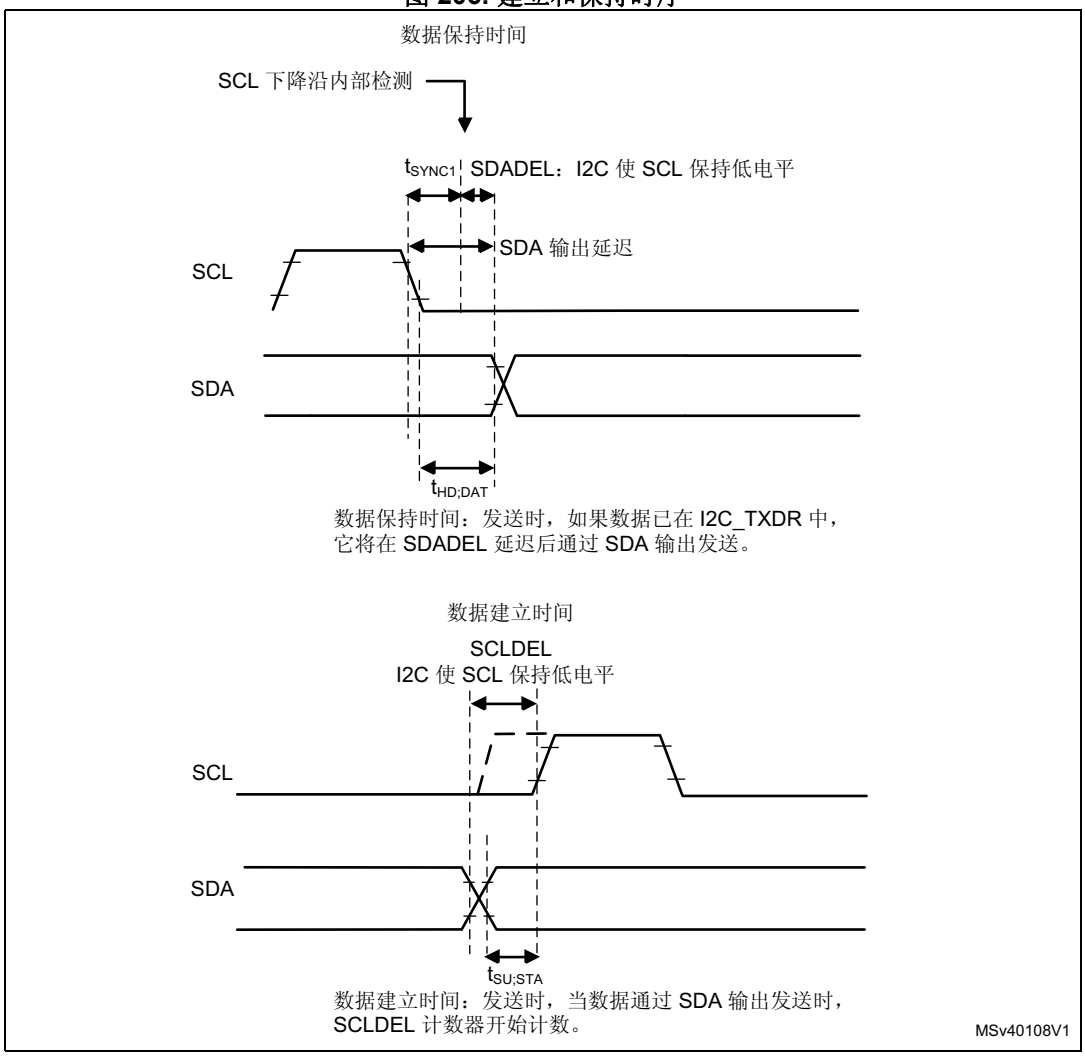
	模拟滤波器	数字滤波器
抑制尖峰的脉冲宽度	≥ 50 ns	长度可编程为 1 到 15 个 I2C 外设时钟
优点	停止模式下仍可用	– 长度可编程：额外的滤波能力与标准要求 – 稳定长度
缺点	随温度、电压和过程变化会发生变化	当使能数字滤波器后，无法在地址匹配时从停止模式唤醒。

注意： 使能 I2C 时，不允许更改滤波器配置。

I2C 时序

必须配置时序，以便保证主模式和从模式下使用正确的数据保持和建立时间。配置方法是编程 I2C_TIMINGR 寄存器中的 PRESC[3:0]、SCLDEL[3:0] 和 SDADEL[3:0] 位。

图 206. 建立和保持时序



- 当内部检测到 SCL 下降沿时，会在发送 SDA 输出之前插入一段延时。该延时为 t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}，其中 t_{PRESC} = (PRESC+1) x t_{I2CCLK}。
  T_{SDADEL} 会影响保持时间 $t_{HD;DAT}$。

SDA 总输出延时为：

t_{SYNC1} + {[SDADEL x (PRESC+1) + 1] x t_{I2CCLK} }

t_{SYNC1} 持续时间取决于以下参数：

- SCL 下降斜率
- 模拟滤波器（使能时）引入的输入延时：$t_{AF(min)} < t_{AF} < t_{AF(max)}$ ns
- 数字滤波器（使能时）引入的输入延时：t_{DNF} = DNF x t_{I2CCLK}
- SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

为了桥接 SCL 下降沿的未定义区域，用户编程 SDADEL 时必须遵循以下条件：

$\{t_{f\ (max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

注：只有使能模拟滤波器时，公式中才包含 $t_{AF(min)}$ / $t_{AF(max)}$。有关 t_{AF} 值的信息，请参见器件数据手册。

标准模式、快速模式和超快速模式下的 $t_{HD;DAT}$ 最大值分别可达 3.45 µs、0.9 µs 和 0.45 µs，但必须小于 $t_{VD;DAT}$ 最大值（差值为跳变时间）。只有器件未延长 SCL 信号的低电平周期 (t_{LOW}) 时，才必须满足该最大值条件。如果时钟延长 SCL，数据必须在建立时间内保持有效，之后才能释放时钟。

SDA 上升沿通常为最坏情况，因此在这种情况下，上述公式变成如下形式：

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$。

注：NOSTRETCH=0 时会违反该条件，这是因为器件会根据 SCLDEL 值来延长 SCL 低电平时间，以保证建立时间。

有关 t_f、t_r、$t_{HD;DAT}$ 和 $t_{VD;DAT}$ 标准值的信息，请参见*表 110：I2C-SMBUS 规范数据建立和保持时间*。

- 在 t_{SDADEL} 延时后，或在因数据未写入 I2C_TXDR 寄存器而导致从器件必须延长时钟的情况下发送 SDA 输出后，SCL 线会在建立时间内保持低电平。该建立时间为 t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}，其中 t_{PRESC} = (PRESC+1) x t_{I2CCLK}。
  t_{SCLDEL} 会影响建立时间 $t_{SU;DAT}$。

为了桥接 SDA 跳变（上升沿通常为最坏情况）的未定义区域，编程 SCLDEL 时必须遵循以下条件：

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 <= SCLDEL$

有关 t_r 和 $t_{SU;DAT}$ 标准值的信息，请参见*表 110：I2C-SMBUS 规范数据建立和保持时间*。

将使用的 SDA 和 SCL 跳变时间值就是应用中的值。使用最大值而非标准值会增加 SDADEL 和 SCLDEL 计算的约束条件，但能够确保任意应用的特性。

注：在发送和接收模式下，对于每个时钟脉冲，检测到 SCL 下降沿后，I2C 主器件或从器件会至少在 [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK} 期间内延长 SCL 低电平时间。在发送模式下，如果 SDADEL 计数器计数结束后数据还未写入 I2C_TXDR，则 I2C 会继续延长 SCL 低电平时间，直到写入下一个数据。随后，会将新数据 MSB 发送到 SDA 输出，SCLDEL 计数器将开始计数，同时会继续延长 SCL 低电平时间以确保提供充足的数据建立时间。

如果从模式下 NOSTRETCH=1，则 SCL 不会延长。因此，编程 SDADEL 时还必须确保提供充足的建立时间。

表 110. I²C-SMBUS 规范数据建立和保持时间

符号	参数	标准模式 (Sm)		快速模式 (Fm)		超快速模式 (Fm+)		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
$t_{HD;DAT}$	数据保持时间	0	-	0	-	0	-	0.3	-	μs
$t_{VD;DAT}$	数据有效时间	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	数据建立时间	250	-	100		50		250		ns
t_r	SDA 和 SCL 信号的上升时间	-	1000		300	-	120	-	1000	
t_f	SDA 和 SCL 信号的下降时间	-	300		300	-	120	-	300	

此外，在主模式下，必须通过编程 I2C_TIMINGR 寄存器中的 PRESC[3:0]、SCLH[7:0] 和 SCLL[7:0] 位来配置 SCL 时钟的高电平和低电平。

- 当内部检测到 SCL 下降沿时，会在释放 SCL 输出之前插入一段延时。该延时为 t_{SCLL} = (SCLL+1) x t_{PRESC}，其中 t_{PRESC} = (PRESC+1) x t_{I2CCLK}。
  t_{SCLL} 会影响 SCL 低电平时间 t_{LOW}。
- 当内部检测到 SCL 上升沿时，会在将 SCL 输出强制为低电平之前插入一段延时。该延时为 t_{SCLH} = (SCLH+1) x t_{PRESC}，其中 t_{PRESC} = (PRESC+1) x t_{I2CCLK}。t_{SCLH} 会影响 SCL 高电平时间 t_{HIGH}。

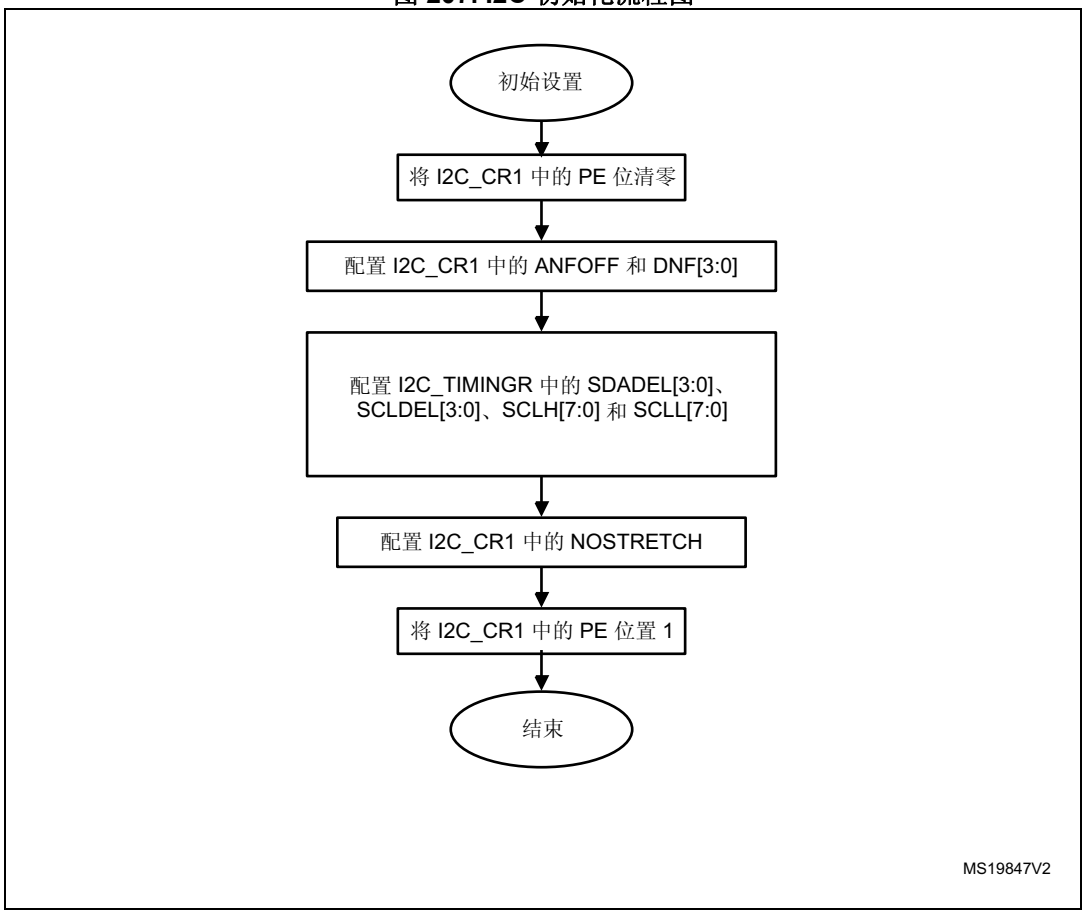
更多详细信息，请参见 *I2C 主模式初始化*。

注意： 使能 I2C 后，不允许更改时序配置。

此外，还必须在使能从设备前，对 NOSTRETCH 进行配置。更多详细信息，请参见 *I2C 从模式初始化*。

注意： 使能 I2C 后，不允许更改 NOSTRETCH 配置。

图 207. I2C 初始化流程图



28.4.6 软件复位

可通过将 I2C_CR1 寄存器中的 PE 位清零来执行软件复位。在这种情况下，I2C 线 SCL 和 SDA 被释放。内部状态机复位，通信控制位和状态位恢复为其复位值。配置寄存器不受影响。

下面列出了受影响的寄存器位：

1. I2C_CR2 寄存器：START、STOP 和 NACK
2. I2C_ISR 寄存器：BUSY、TXE、TXIS、RXNE、ADDR、NACKF、TCR、TC、STOPF、BERR、ARLO 和 OVR

支持 SMBus 功能时还会影响到以下寄存器位：

1. I2C_CR2 寄存器：PECBYTE
2. I2C_ISR 寄存器：PECERR、TIMEOUT 和 ALERT

必须使 PE 保持低电平持续至少 3 个 APB 时钟周期，才能成功执行软件复位。写入以下软件序列可确保这一点：- 写入 PE=0 - 检查 PE=0 - 写入 PE=1

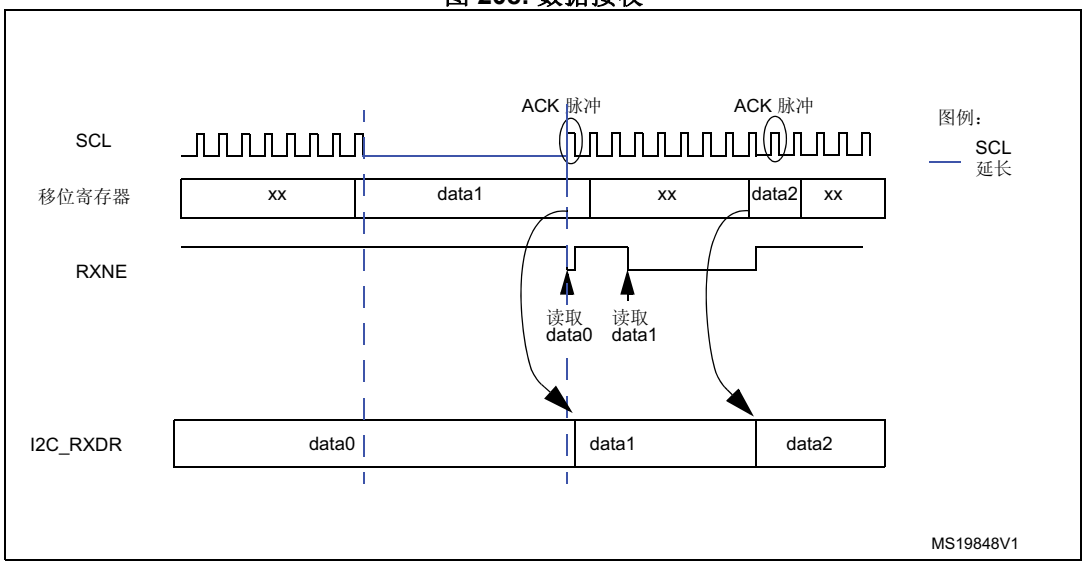
28.4.7 数据传输

数据传输由发送和接收数据寄存器以及移位寄存器来管理。

接收

SDA 输入填充移位寄存器。在第 8 个 SCL 脉冲后（接收到完整的数据字节时），如果 I2C_RXDR 寄存器为空 (RXNE=0)，则移位寄存器的内容会复制到其中。如果 RXNE=1（意味着尚未读取上一次接收到的数据字节），则将延长 SCL 线的低电平时间，直到读取了 I2C_RXDR 为止。在第 8 个和第 9 个 SCL 脉冲之间（应答脉冲之前）插入一段延长的时间。

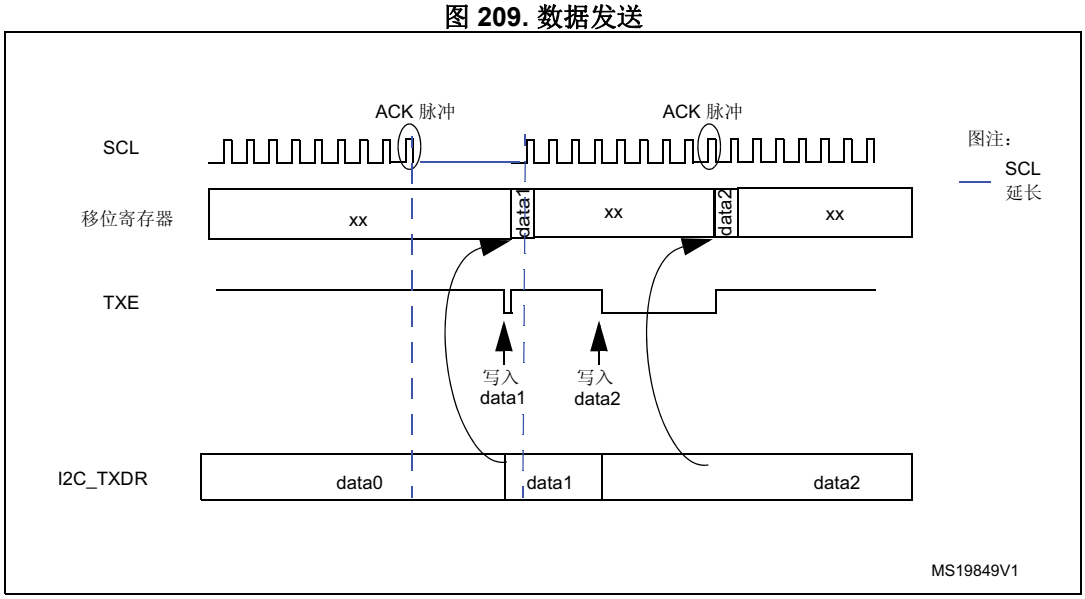
图 208. 数据接收



发送

如果 I2C_TXDR 寄存器不为空 (TXE=0)，则其内容会在第 9 个 SCL 脉冲（应答脉冲）后复制到移位寄存器中。然后移位寄存器的内容会移出到 SDA 线上。如果 TXE=1（意味着 I2C_TXDR 内尚未写入任何数据），则将延长 SCL 线的低电平时间，直到写入了 I2C_TXDR 为止。在第 9 个 SCL 脉冲后进行延长。

图 209. 数据发送



硬件传输管理

I2C 在硬件中内置了字节计数器，以便在下列各种模式下管理字节传输和结束通信：

- 主模式下生成 NACK、STOP 和 ReSTART
- 从接收器模式下控制 ACK 是否发出
- SMBus 模式下生成/校验 PEC

字节计数器通常在主模式下使用。在从模式下，字节计数器默认为禁止状态，但可以通过软件来使能，方法是将 I2C_CR2 寄存器中的 SBC（从字节控制）位置 1。

待传输的字节数在 I2C_CR2 寄存器的 NBYTES[7:0] 位域中进行编程。如果待传输的字节数 (NBYTES) 大于 255，或者接收方希望控制是否对接收到的数据字节进行应答，则必须选择重载模式，方法是将 I2C_CR2 寄存器的 RELOAD 位置 1。在该模式下，完成 NBYTES 中所编程字节数的数据传输之后，TCR 标志将置 1，并且 TCIE 置 1 时将生成中断。只要 TCR 标志置 1，SCL 便会延长。当往 NBYTES 写入一个非零值时，TCR 由软件清零。

在往 NBYTE 中设置最后一次传输的字节数前，必须把 RELOAD 位清零。

当主模式下 RELOAD=0 时，可在以下 2 种模式下使用计数器：

- **自动结束模式**（I2C_CR2 寄存器中的 AUTOEND =“1”）。在该模式下，一旦完成 NBYTES[7:0] 位域中所编程字节数的数据传输，主器件便会自动发送停止位。
- **软件结束模式**（I2C_CR2 寄存器中的 AUTOEND =“0”）在该模式下，一旦完成 NBYTES[7:0] 位域中所编程字节数的数据传输，TC 标志将置 1，并且 TCIE 置 1 时将生成中断。只要 TC 标志置 1，SCL 信号便会延长，需要软件介入操作。当软件把 I2C_CR2 寄存器中的起始位或停止位置 1 时，TC 标志将被清零。当主器件要发送重复起始位时，必须使用该模式。

注意： 当 RELOAD 位置 1 时，AUTOEND 位将不起作用。

表 111. I2C 配置表

功能	SBC 位	RELOAD 位	AUTOEND 位
主 Tx/Rx NBYTES + STOP	x	0	1
主 Tx/Rx + NBYTES + RESTART	x	0	0
从 Tx/Rx 接收的所有字节都要回复应答	0	x	x
具有 ACK 控制的从 Rx	1	1	x

28.4.8 I2C 从模式

I2C 从模式初始化

要在从模式下工作，用户必须至少使能一个从地址。可使用 I2C_OAR1 和 I2C_OAR2 这两个寄存器来编程自身从地址 OA1 和 OA2。

- OA1 既可配置为 7 位寻址模式（默认），也可通过将 I2C_OAR1 寄存器的 OA1MODE 位置 1 配置为 10 位寻址模式。

  通过将 I2C_OAR1 寄存器中的 OA1EN 位置 1 来使能 OA1。
- 如果需要额外的从地址，可配置第 2 个从地址 OA2。将 I2C_OAR2 寄存器的 OA2MSK[2:0] 位置 1 最多可屏蔽 7 个 OA2 LSB。因此，当 OA2MSK 配置为 1 到 6 时，将分别只有 OA2[7:2]、OA2[7:3]、OA2[7:4]、OA2[7:5]、OA2[7:6] 或 OA2[7] 与接收到的地址作比较。只要 OA2MSK 不等于 0， OA2 的地址比较器便会排除 I2C 保留地址 （0000 XXX 和 1111 XXX），这些地址将不会得到应答。如果 OA2MSK=7，接收到的所有 7 位地址（保留地址除外）均得到应答。 OA2 始终为 7 位地址。

  如果这些保留地址在 I2C_OAR1 或 I2C_OAR2 寄存器中进行了编程并且 OA2MSK=0，则它们可以在通过特定使能位使能后得到应答。

  通过将 I2C_OAR2 寄存器中的 OA2EN 位置 1 来使能 OA2。
- 通过将 I2C_CR1 寄存器中的 GCEN 位置 1 来使能广播呼叫地址。

当通过 I2C 的其中一个使能地址来寻址到该 I2C 设备时，ADDR 中断状态标志将置 1，并且 ADDRIE 位置 1 时将生成中断。

默认情况下，从器件使用其时钟延长功能（即必要时延长 SCL 信号的低电平时间）来为软件操作的执行提供时机。如果主器件不支持时钟延长，则必须对 I2C 进行如下配置：将 I2C_CR1 寄存器中 NOSTRETCH 位置 1。

接收到 ADDR 中断后，如果使能多个地址，则用户必须读取 I2C_ISR 寄存器中的 ADDCODE[6:0] 位，以确定是哪个地址匹配。还必须检查 DIR 标志，以获悉传输方向。

带时钟延长的从模式 (NOSTRETCH = 0)

在默认模式下，I2C 从器件会在以下情况下延长 SCL 时钟：

- ADDR 标志置 1 时：接收到的地址与其中一个使能的从地址匹配。通过软件将 ADDRCF 位置 1 以清零 ADRR 标志时，将释放该时钟延展。
- 发送时，前一次数据传输已完成但 I2C_TXDR 寄存器中未写入任何新数据，或者 ADDR 标志清零 (TXE=1) 时未写入第一个数据字节。往 I2C_TXDR 寄存器中写入数据时，将释放该时钟延展。
- 接收时，尚未读取 I2C_RXDR 寄存器但新的数据接收已完成。读取 I2C_RXDR 时，将释放该时钟延展。
- 当从器件字节控制模式和重载模式（SBC=1 且 RELOAD=1）下 TCR = 1 时，这意味着最后一个数据字节已完成传输。通过向 NBYTES[7:0] 字段写入一个非零值以将 TCR 清零时，将释放该时钟延展。
- 在 SCL 下降沿检测之后，I2C 会延长 SCL 的低电平时间（不超过 $[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times t_{I2CCLK}$）。

不带时钟延长的从模式 (NOSTRETCH = 1)

当 I2C_CR1 寄存器中的 NOSTRETCH = 1 时，I2C 从器件不会延长 SCL 信号。

- ADDR 标志置 1 时，不会延长 SCL 时钟。
- 发送时，必须在与发送数据对应的第一个 SCL 脉冲出现之前，向 I2C_TXDR 寄存器写入数据。否则，会发生下溢，I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。当第一次数据发送开始而 STOPF 位仍置 1（尚未清零）时，OVR 标志也将置 1。因此，如果写入下一次传输要发送的第一个数据后才清零上一次传输的 STOPF 标志，则应提供 OVR 状态，甚至对于待发送的第一个数据也是如此。
- 接收时，必须在下一个数据字节的第 9 个 SCL 脉冲（ACK 脉冲）出现之前，从 I2C_RXDR 寄存器读取数据。否则，会发生上溢， I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

从器件字节控制模式

要在从接收模式下实现字节 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从器件字节控制模式。这样符合 SMBus 标准。

要在从接收模式下实现字节 ACK 控制，必须选择重载模式 (RELOAD=1)。要控制每个字节，必须在 ADDR 中断子程序中将 NBYTES 初始化为 0x1，并在每接收一个字节后将 NBYTE 重载为 0x1。接收到字节后，TCR 位将置 1，从而延长 SCL 信号的第 8 个和第 9 个脉冲之间的低电平时间。用户可以从 I2C_RXDR 寄存器中读取数据，然后通过配置 I2C_CR2 寄存器中的 ACK 位来决定是否应答。通过将 NBYTES 编程为非零值来释放 SCL 延长：发送应答或不应答信号，然后可继续接收下一个字节。

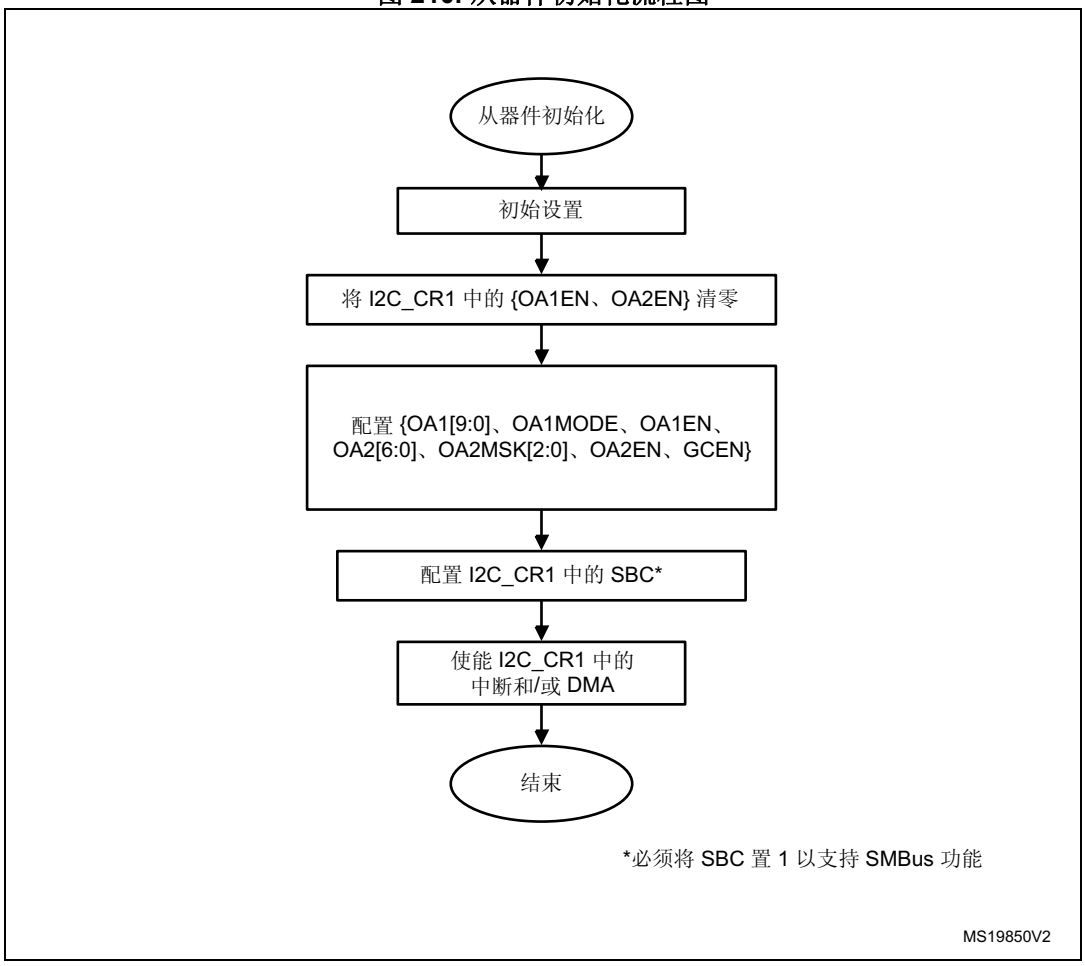
NBYTES 可加载大于 0x1 的值，在这种情况下，接收流在 NBYTES 个数据接收期间是连续的。

注： *SBC 位只能在 I2C 被禁止时、从器件不被寻址时或 ADDR=1 时配置。*

ADDR=1 或 TCR=1 时，可以更改 RELOAD 位的值。

注意： 从器件字节控制模式与 NOSTRETCH 模式不兼容。不允许在 NOSTRETCH=1 时将 SBC 位置 1。

图 210. 从器件初始化流程图



有关代码示例，请参见 *A.16.1：将 I2C 配置为从模式的代码示例*。

从发送器

当 I2C_TXDR 寄存器为空时，将生成发送中断状态 (TXIS)。如果 I2C_CR1 寄存器中的 TXIE 位置 1，将生成中断。

I2C_TXDR 寄存器中写入待发送的下一个数据字节时，TXIS 位将被清零。

接收到 NACK 时，I2C_ISR 寄存器中的 NACKF 位将置 1，如果 I2C_CR1 寄存器中的 NACKIE 位置 1，还将生成中断。从器件自动释放 SCL 和 SDA 线，以使主器件执行停止或重复起始位的发送。收到 NACK 时，TXIS 位不会置 1。

当接收到停止位且 I2C_CR1 寄存器中的 STOPIE 位置 1 时，I2C_ISR 寄存器中的 STOPF 标志将置 1 并且会生成中断。在大多数应用中，SBC 位通常编程为“0”。在这种情况下，如果接收到从地址 (ADDR=1) 时 TXE = 0，用户可以选择发送 I2C_TXDR 寄存器的内容作为第一个数据字节，也可以选择通过将 TXE 位置 1 来刷新 I2C_TXDR 寄存器以编程新的数据字节。

在从器件字节控制模式 (SBC=1) 下，必须在地址匹配中断子程序 (ADDR=1) 中向 NBYTE 写入待发送数据的个数。在这种情况下，传输期间 TXIS 事件的数量对应于 NBYTES 中编程的值。

注意： 如果 NOSTRETCH=1，当 ADDR 标志置 1 时不会延长 SCL 时钟，因此用户无法在 ADDR 子程序中刷新 I2C_TXDR 寄存器的内容，从而编程第一个数据字节。必须在 I2C_TXDR 寄存器中预编程待发送的第一个数据字节：

- 该数据可以是前一个传输消息的最后一个 TXIS 事件中写入的数据。
- 如果该数据字节不是待发送的数据字节，可通过将 TXE 位置 1 来刷新 I2C_TXDR 寄存器，从而编程新的数据字节。必须仅在执行完这些操作后再清零 STOPF 位，以确保在地址应答之后，第一次数据传输开始之前执行这些操作。

  如果第一次数据传输开始时 STOPF 仍置 1，则将生成下溢错误（OVR 标志置 1）。

  如果需要 TXIS 事件（发送中断或发送 DMA 请求），用户必须将 TXE 位和 TXIS 位均置 1，以便生成 TXIS 事件。

图 211. I2C 从发送器的传输序列流程图 (NOSTRETCH=0)

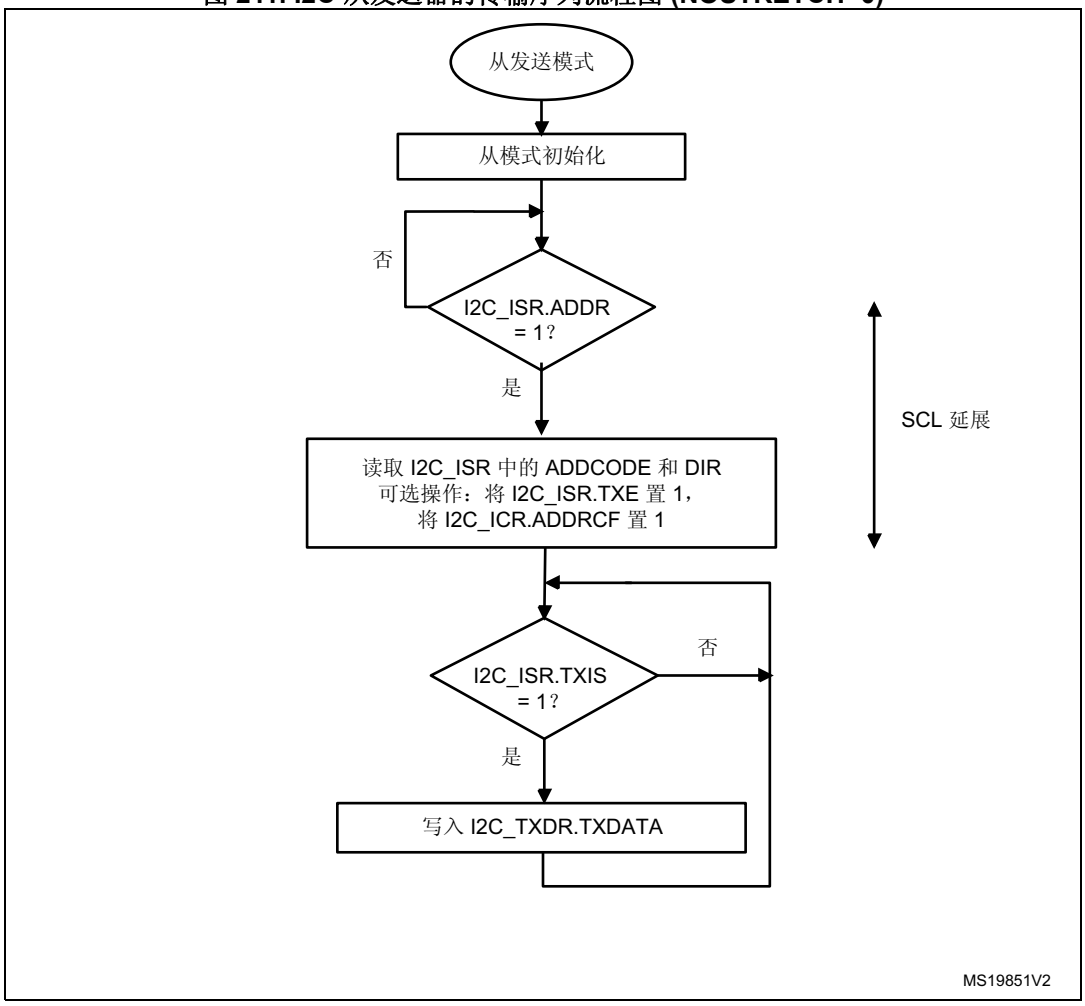


图 212. I2C 从发送器的传输序列流程图 (NOSTRETCH=1)

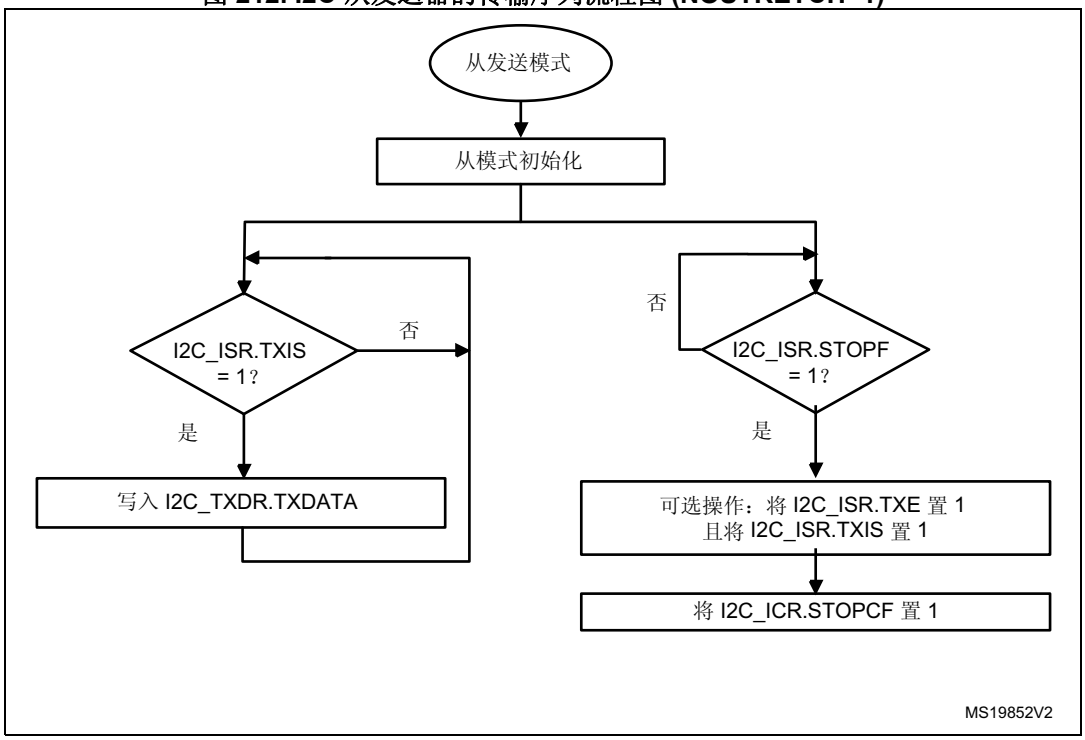
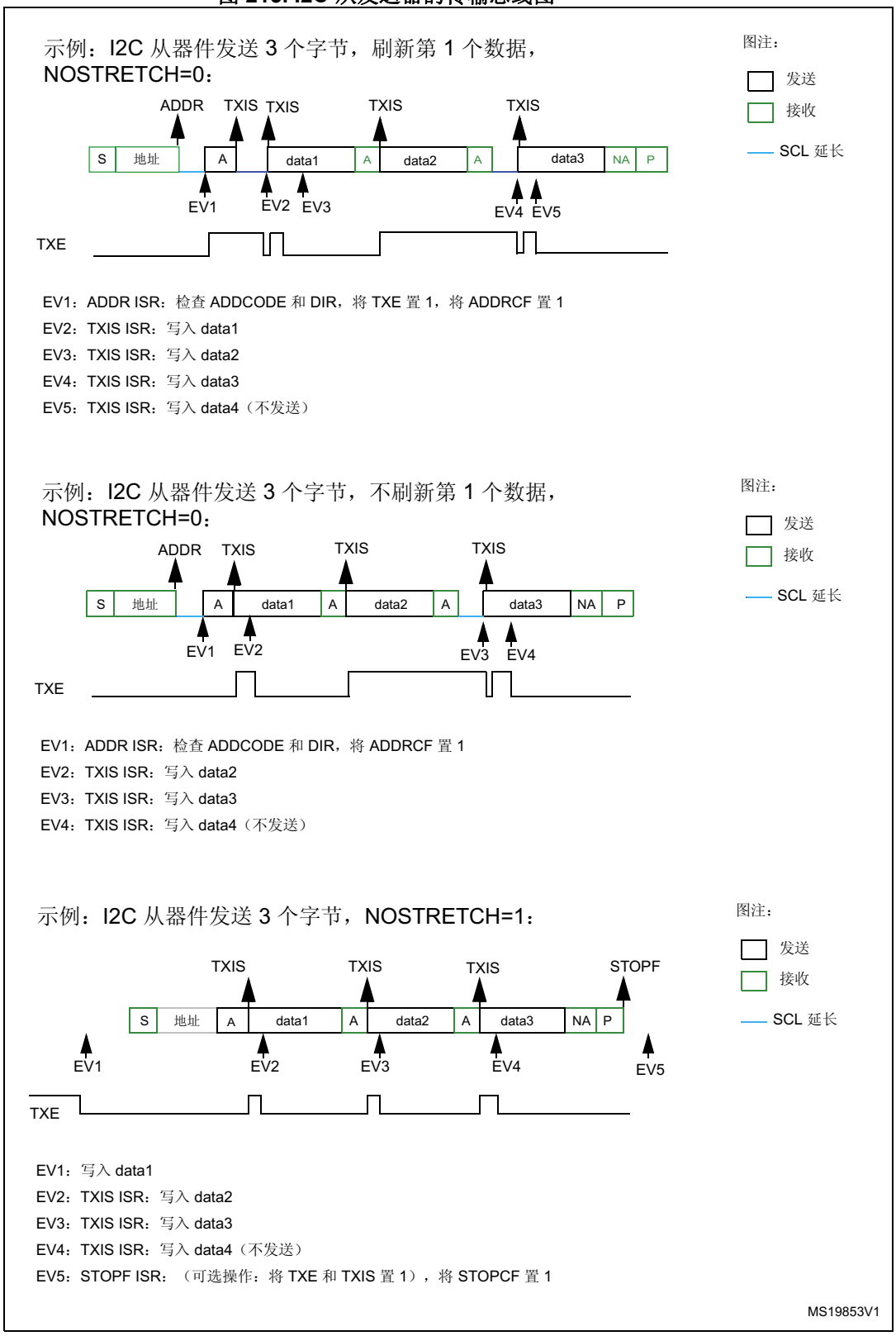


图 213. I2C 从发送器的传输总线图



有关代码示例，请参见 *A.16.2：I2C 从发送器代码示例*。

从接收器

当 I2C_RXDR 满时，I2C_ISR 中的 RXNE 将置 1，如果 I2C_CR1 中的 RXIE 置 1，还将生成中断。读取 I2C_RXDR 时，将清零 RXNE。

接收到停止条件且 I2C_CR1 寄存器中的 STOPIE 置 1 时，I2C_ISR 中的 STOPF 将置 1 并且会生成中断。

图 214. 从接收器的传输序列流程图 (NOSTRETCH=0)

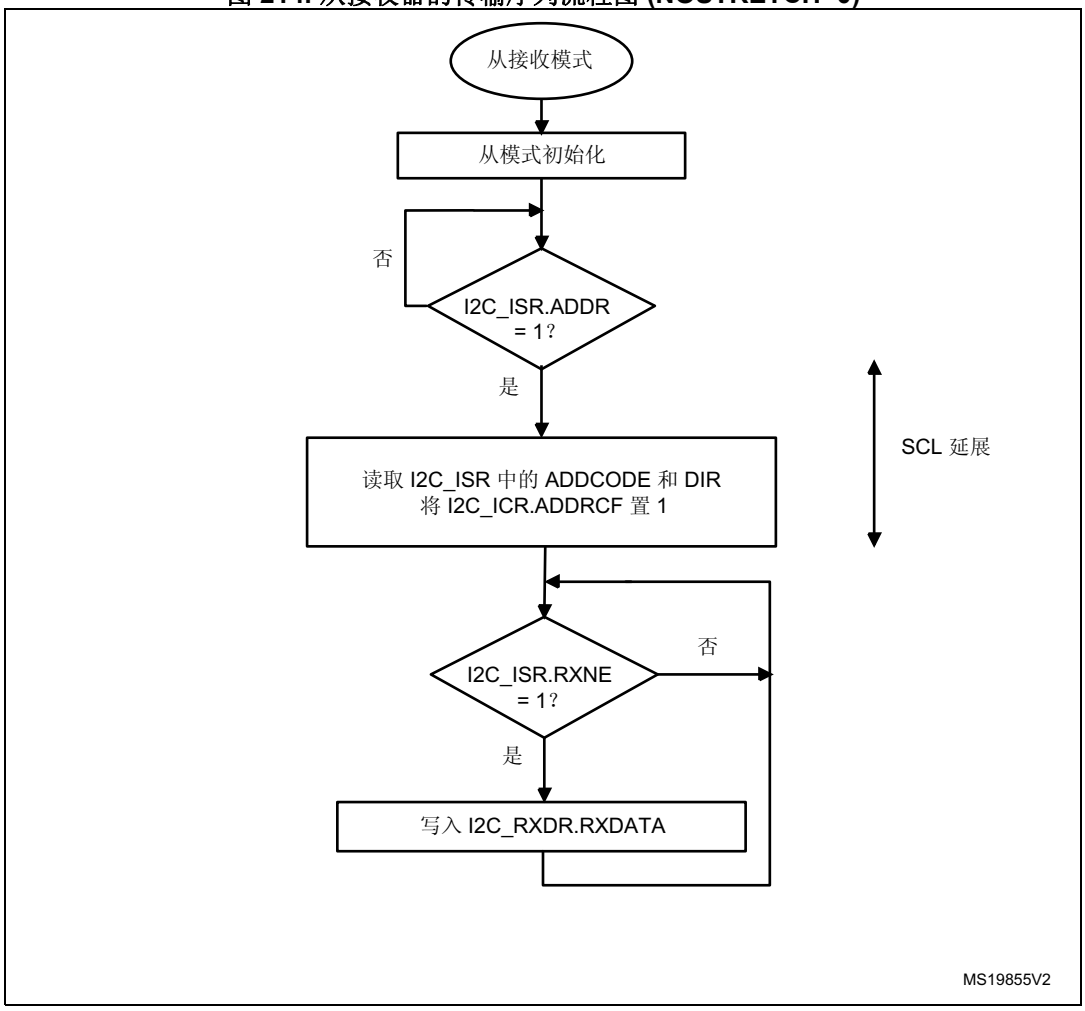


图 215. 从接收器的传输序列流程图 (NOSTRETCH=1)

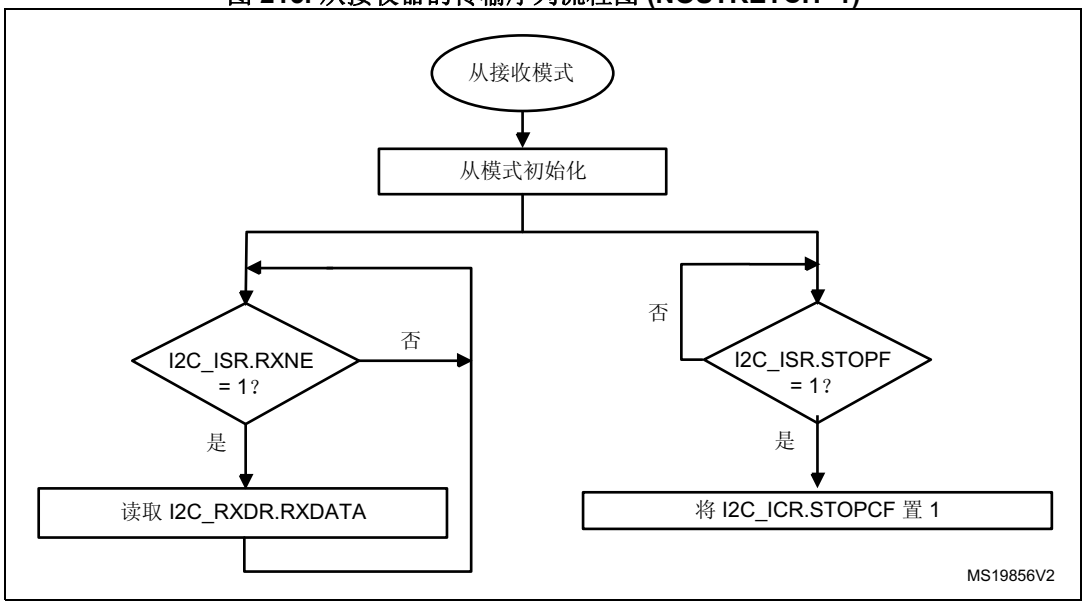
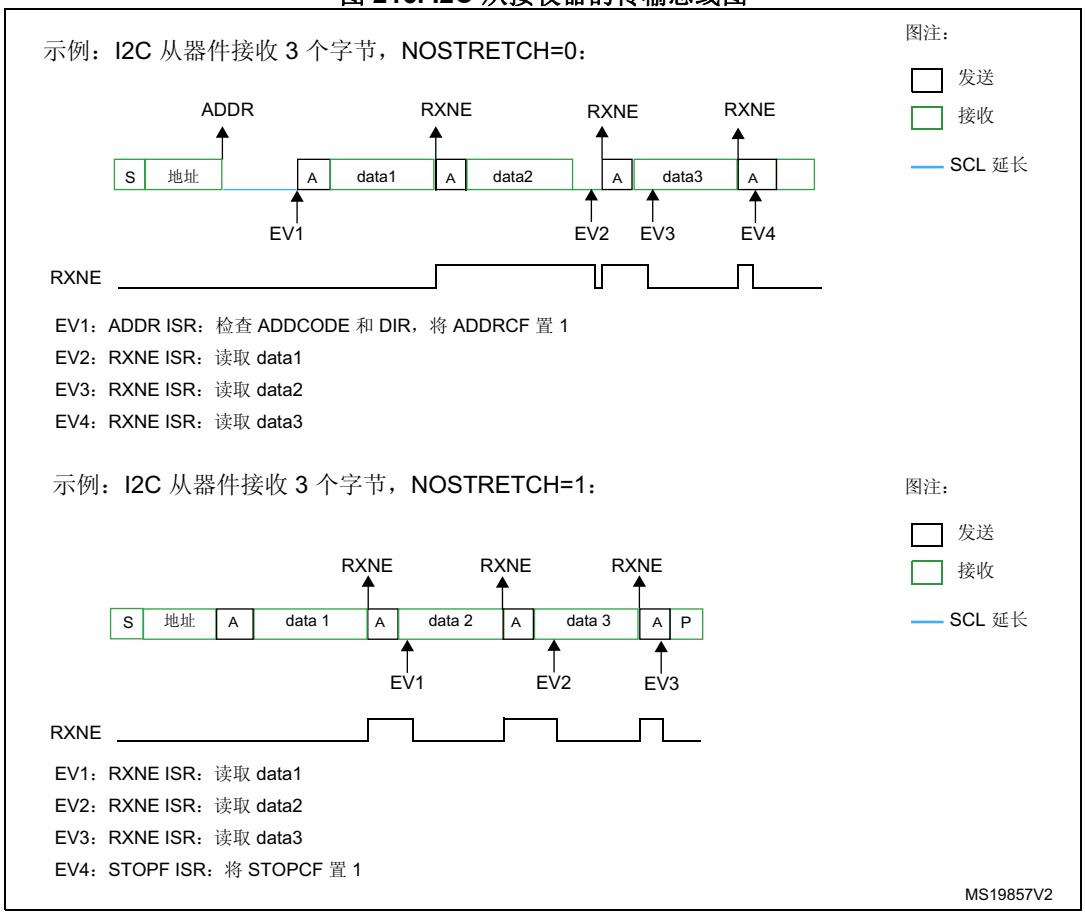


图 216. I2C 从接收器的传输总线图



有关代码示例，请参见 *A.16.3：I2C 从接收器代码示例*。

28.4.9 I2C 主模式

I2C 主模式初始化

使能外设前，必须通过设置 I2C_TIMINGR 寄存器中的 SCLH 和 SCLL 位来配置 I2C 主时钟。

为了支持多主环境和从时钟延长，I2C 实现了时钟同步机制。

为了实现时钟同步，需执行以下操作：

- 使用 SCLL 计数器从 SCL 低电平内部检测开始对时钟的低电平进行计数。
- 使用 SCLH 计数器从 SCL 高电平内部检测开始对时钟的高电平进行计数。

I2C 经过 t_{SYNC1} 延时后检测其自身的 SCL 低电平，该延时取决于 SCL 下降沿、SCL 输入噪声滤波器（模拟 + 数字）以及 SCL 与 I2CxCLK 时钟的同步。一旦 SCLL 计数器达到 I2C_TIMINGR 寄存器的 SCLL[7:0] 位中编程的值，I2C 便会将 SCL 释放为高电平。

I2C 经过 t_{SYNC2} 延时后检测其自身的 SCL 高电平，该延时取决于 SCL 上升沿、SCL 输入噪声滤波器（模拟 + 数字）以及 SCL 与 I2CxCLK 时钟的同步。一旦 SCLH 计数器达到 I2C_TIMINGR 寄存器的 SCLH[7:0] 位中编程的值，I2C 便会使 SCL 变为低电平。

因此，主时钟周期为：

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

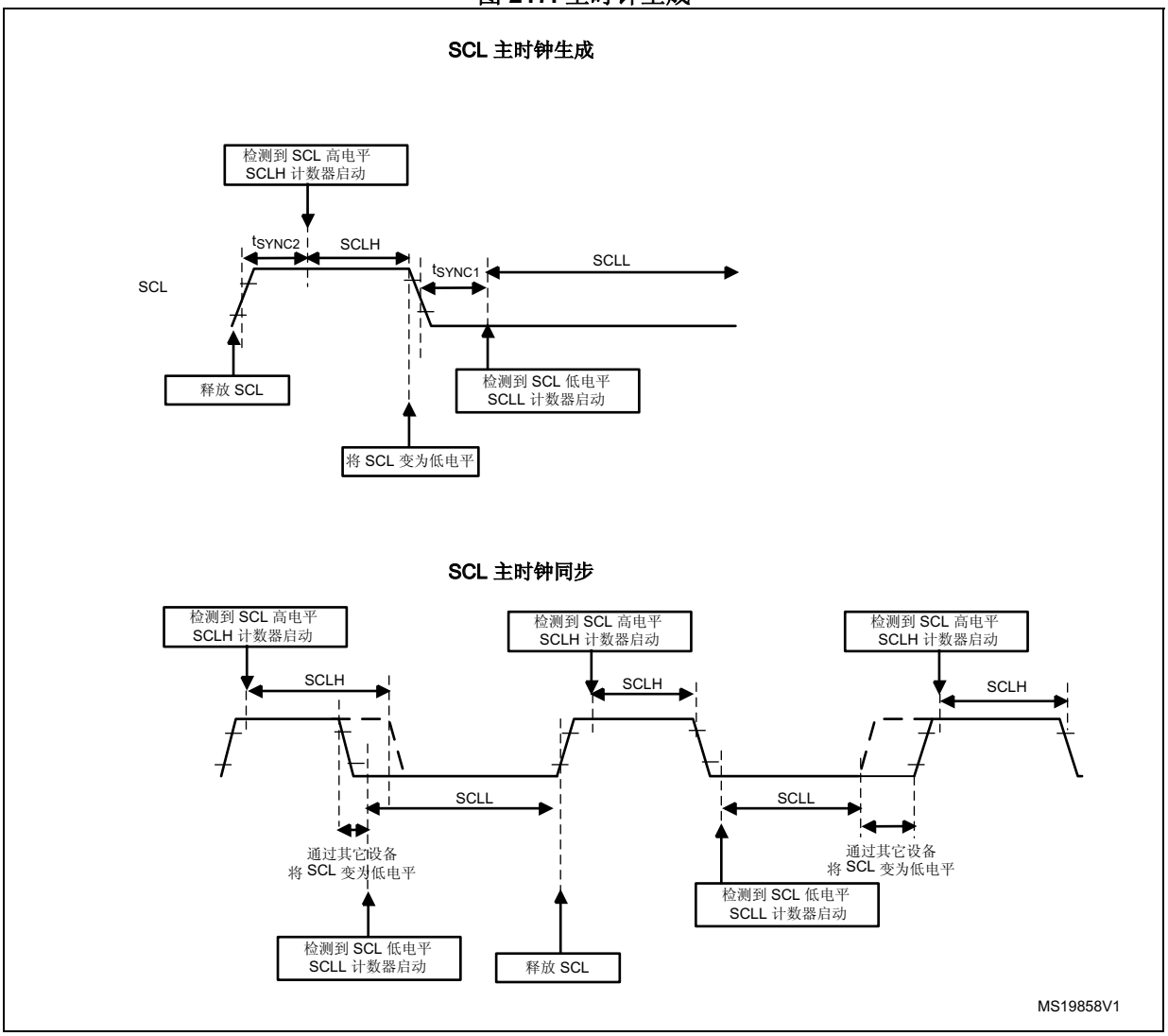
t_{SYNC1} 的持续时间取决于以下参数：

- SCL 下降斜率
- 模拟滤波器（使能时）引入的输入延时。
- 数字滤波器（使能时）引入的输入延时：DNF x t_{I2CCLK}
- SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

t_{SYNC2} 的持续时间取决于以下参数：

- SCL 上升斜率
- 模拟滤波器（使能时）引入的输入延时。
- 数字滤波器（使能时）引入的输入延时：DNF x t_{I2CCLK}
- SCL 与 I2CCLK 时钟建立同步而产生的延时（2 到 3 个 I2CCLK 周期）

图 217. 主时钟生成



注意： 为了符合 I^2C 或 SMBus 规范，主时钟必须遵循下表中给出的时序：

表 112. I²C-SMBUS 规范时钟时序

符号	参数	标准模式 (Sm)		快速模式 (Fm)		超快速模式 (Fm+)		SMBus		单位
		最小值	最大值	最小值	最大值	最小值	最大值	最小值	最大值	
f_{SCL}	SCL 时钟频率		100		400		1000		100	kHz
$t_{HD:STA}$	（重复）起始条件的保持时间	4.0	-	0.6		0.26	-	4.0	-	µs
$t_{SU:STA}$	重复起始条件的建立时间	4.7	-	0.6		0.26	-	4.7	-	µs
$t_{SU:STO}$	停止条件的建立时间	4.0	-	0.6		0.26	-	4.0	-	µs
t_{BUF}	停止条件和起始条件之间的总线空闲时间	4.7	-	1.3		0.5	-	4.7	-	µs
t_{LOW}	SCL 时钟的低电平周期	4.7	-	1.3		0.5	-	4.7	-	µs
t_{HIGH}	SCL 时钟的高电平周期	4.0	-	0.6		0.26	-	4.0	50	µs
t_r	SDA 和 SCL 信号的上升时间	-	1000	-	300		120	-	1000	ns
t_f	SDA 和 SCL 信号的下降时间	-	300	-	300		120	-	300	ns

注： *SCLL 还用于生成 t_{BUF} 和 $t_{SU:STA}$ 时序。*

SCLH 还用于生成 $t_{HD:STA}$ 和 $t_{SU:STO}$ 时序。

有关 I2C_TIMINGR 设置与 I2CCLK 频率的示例，请参见*第 28.4.10 节：I2C_TIMINGR 寄存器配置示例*。

主模式通信初始化（地址阶段）

要发起通信，用户必须在 I2C_CR2 寄存器中为寻址的从器件编程以下参数：

- 寻址模式（7 位或 10 位）：ADD10
- 待发送的从地址：SADD[9:0]
- 传输方向：RD_WRN
- 读取 10 位地址时：HEAD10R 位。必须对 HEAD10R 进行相应配置，以指示传输方向变化时必须发送完整的地址序列，还是只发送地址头。
- 待传输的字节数：NBYTES[7:0]。如果字节数等于或大于 255，则初始化时必须将 NBYTES[7:0] 填充为 0xFF。

然后，用户必须将 I2C_CR2 寄存器中的 START 位置 1。START 位置 1 时，不允许更改上述所有位。

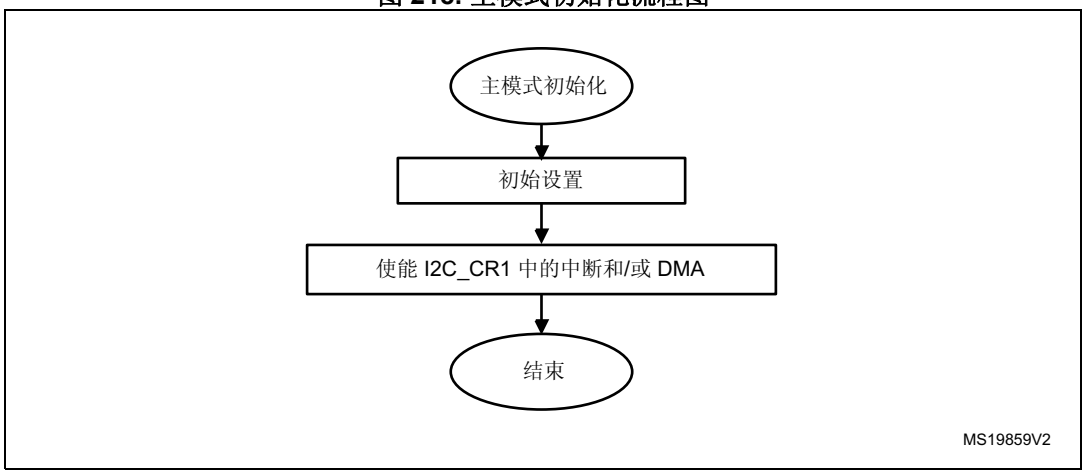
之后，当主器件检测到总线空闲 (BUSY = 0) 时，它会在经过 t_{BUF} 的延时后自动发送起始位，随后发出从器件地址。

仲裁丢失时，主器件将自动切换回从模式，如果作为从器件被寻址，还可对其自身地址进行应答。

注： *无论接收到的应答值为何，只要已在总线上发送从地址，START 位便会由硬件复位。如果仲裁丢失，START 位也会由硬件复位。如果当 START 位置 1 时，I2C 作为从器件 (ADDR=1) 被寻址，则 I2C 将切换为从模式，START 位将在 ADDRCF 位置 1 时清零。*

注： *该步骤同样适用于重复起始位。在这种情况下，BUSY=1。*

图 218. 主模式初始化流程图

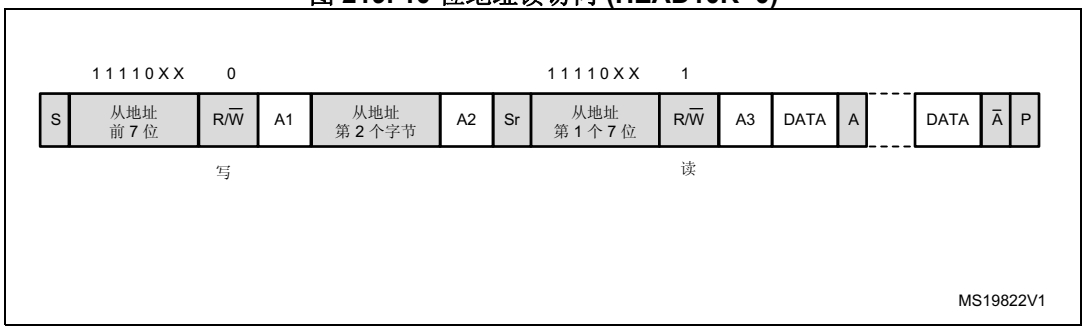


有关代码示例，请参见 *A.16.4：将 I2C 配置为主模式以进行接收的代码示例*和 *A.16.5：将 I2C 配置为主模式以进行发送的代码示例*。

主接收器寻址 10 位地址从器件的初始化过程

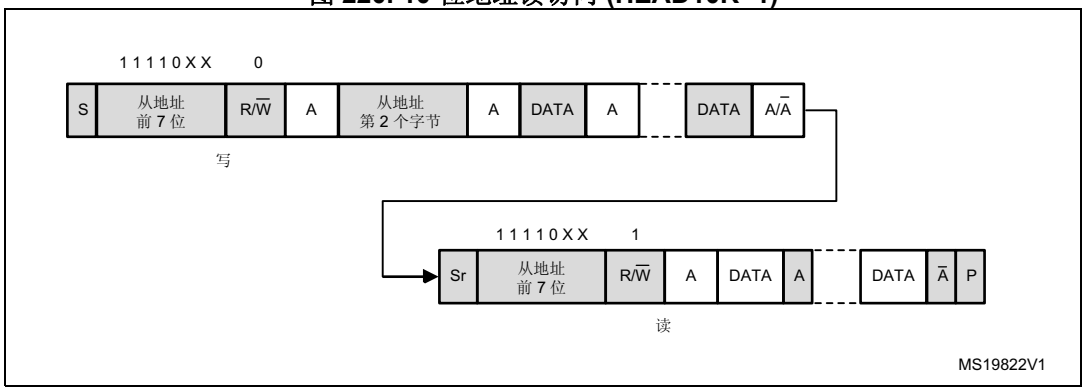
- 如果从地址采用 10 位格式，用户可选择将 I2C_CR2 寄存器中的 HEAD10R 位清零来发送完整的读序列。在这种情况下，主器件会在 START 位置 1 后自动发送以下完整序列：（重复）起始位 + 带写方向的从器件 10 位地址头字节 + 从器件地址第 2 个字节 + 重复起始位 + 带读方向的从器件 10 位地址头字节

图 219. 10 位地址读访问 (HEAD10R=0)



- 如果主器件对 10 位地址从器件进行寻址、向该从器件发送数据、然后再从该从器件读取数据，则必须首先完成主器件发送过程。然后，重复起始位置 1，10 位从地址配置为 HEAD10R=1。在这种情况下，主器件发送以下序列：重复起始位 + 从地址 10 位头读取。

图 220. 10 位地址读访问 (HEAD10R=1)



主发送器

写传输时，在发送完每个字节（即第 9 个 SCL 脉冲（接收到 ACK 时））后，TXIS 标志将置 1。

如果 I2C_CR1 寄存器中的 TXIE 位置 1，TXIS 事件将生成中断。当 I2C_TXDR 寄存器中写入待发送的下一个数据字节时，该标志将被清零。

传输期间的 TXIS 事件的数量对应于 NBYTES[7:0] 中编程的值。如果待发送的数据字节总数大于 255，则必须通过将 I2C_CR2 寄存器中的 RELOAD 位置 1 来选择重载模式。在这种情况下，当 NBYTES 数据传输完成时，TCR 标志将置 1，并且 SCL 线的低电平将被延展，直到 NBYTES[7:0] 被写入非零值。

收到 NACK 时，TXIS 标志不会置 1。

- 当 RELOAD=0 且 NBYTES 数据传输完成时：
  - 在自动结束模式 (AUTOEND=1) 下，将自动发送停止位。
  - 在软件结束模式 (AUTOEND=0) 下，TC 标志将置 1 且 SCL 线的低电平将被延展，以便执行以下软件操作：

    可通过将 I2C_CR2 寄存器中的 START 位置 1 并配置适当的从地址和待传输字节数来请求发送重复起始位。将 START 位置 1 会将 TC 标志清零，并在总线上发送起始位。

    可通过将 I2C_CR2 寄存器中的 STOP 位置 1 来请求停止位。将 STOP 位置 1 会将 TC 标志清零，并在总线上发送停止位。
- 如果接收到 NACK：TXIS 标志不会置 1，并且接收到 NACK 后会自动发送停止位。I2C_ISR 寄存器中的 NACKF 标志置 1，如果 NACKIE 位置 1，还将生成中断。

图 221. I2C 主发送器的传输序列流程图（N≤255 字节）

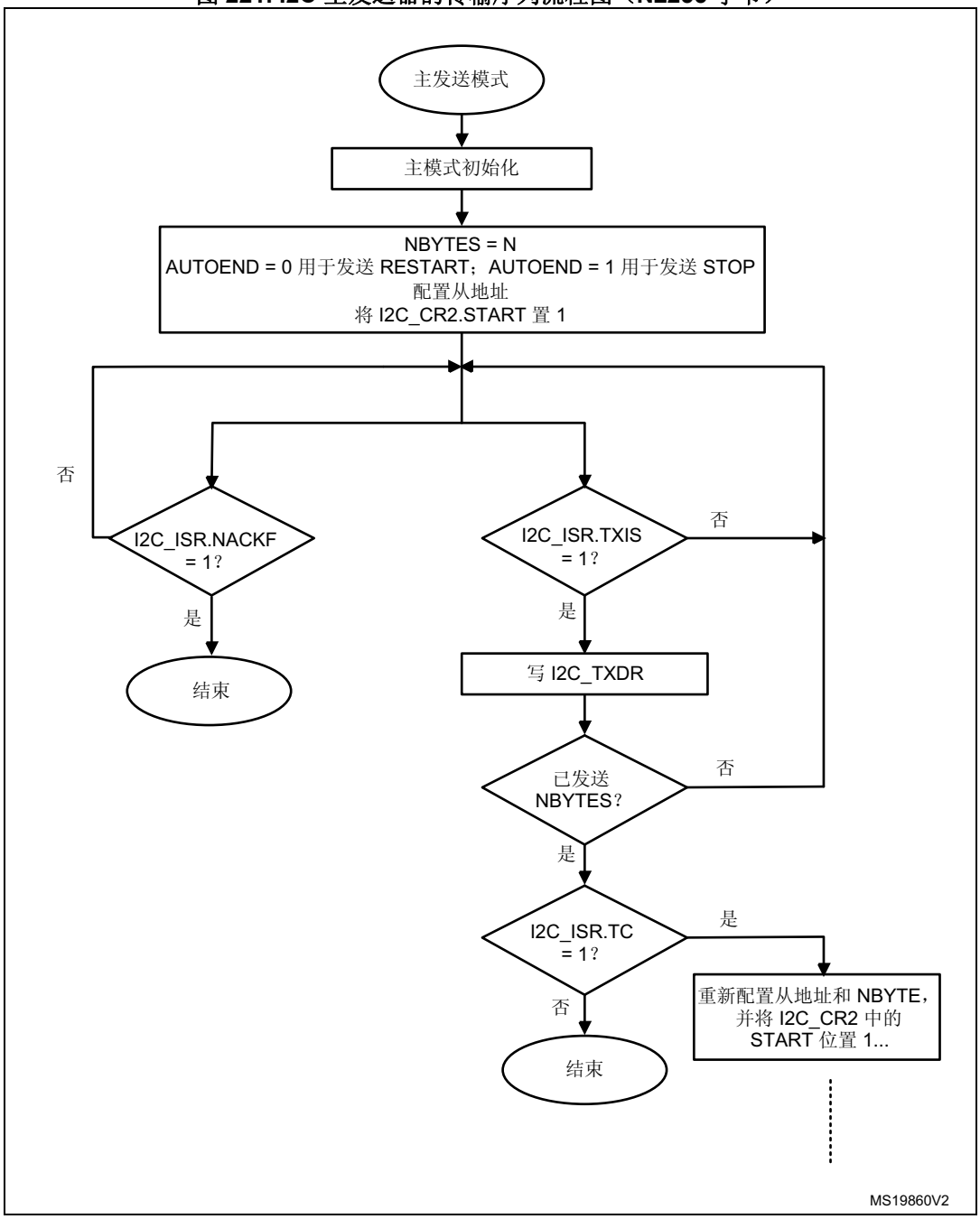


图 222. I2C 主发送器的传输序列流程图（N>255 字节）

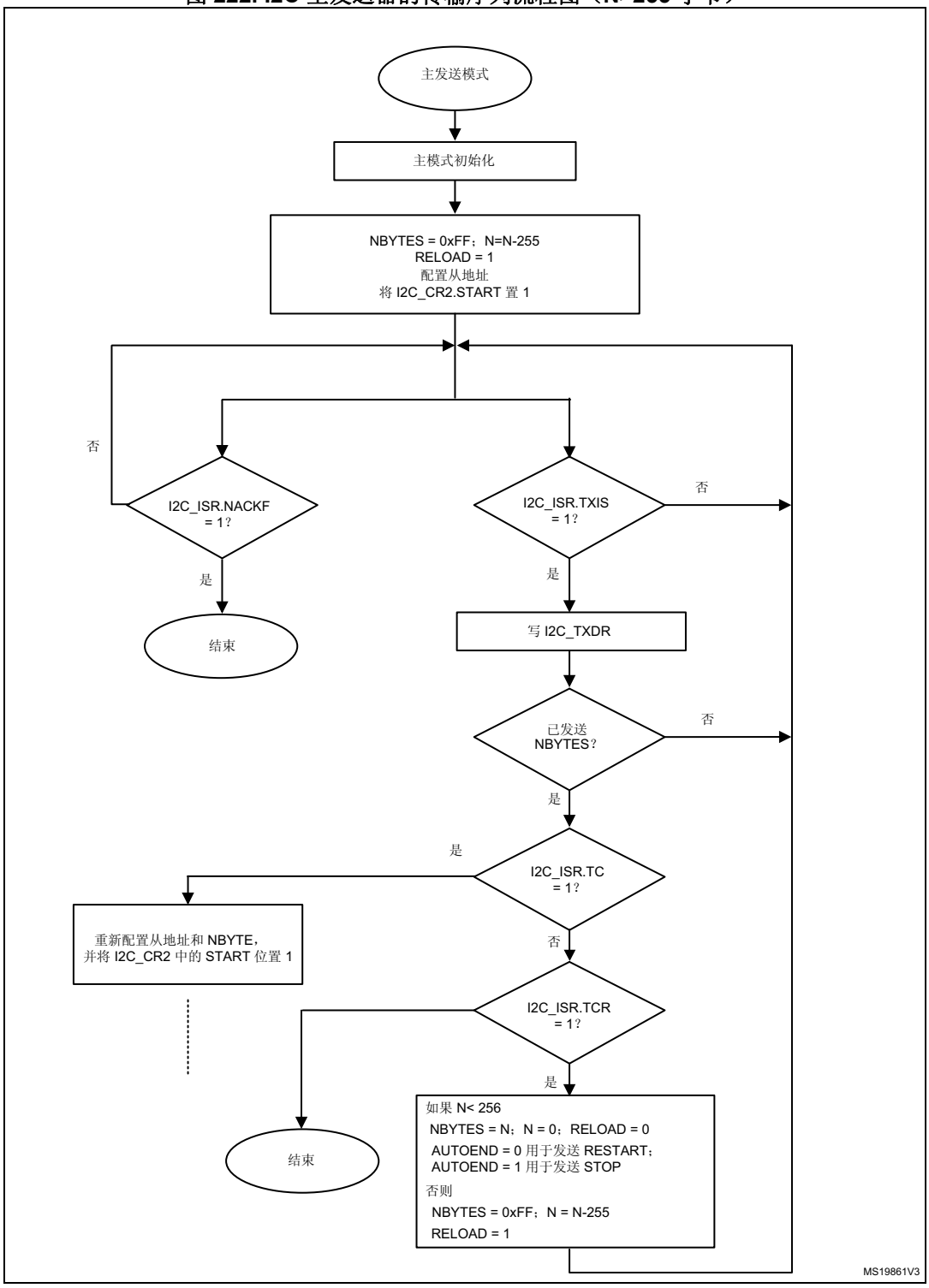
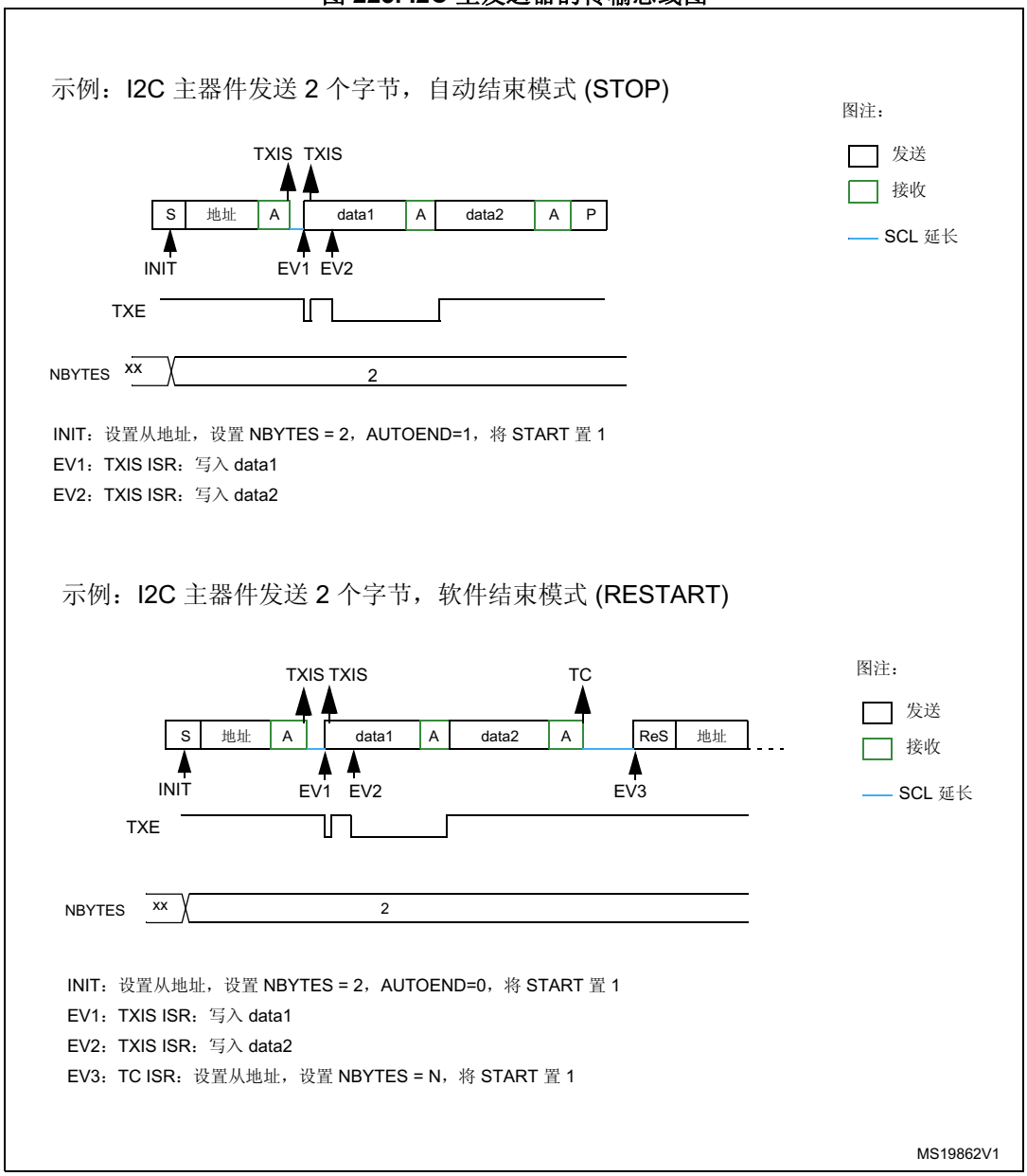


图 223. I2C 主发送器的传输总线图



有关代码示例，请参见 *A.16.6：I2C 主发送器代码示例*。

主接收器

读传输时，在接收到每个字节（即第 8 个 SCL 脉冲）后，RXNE 标志将置 1。如果 I2C_CR1 寄存器中的 RXIE 位置 1，RXNE 事件将生成中断。读取 I2C_RXDR 时，将清零该标志。

如果待接收的数据字节总数大于 255，则必须通过将 I2C_CR2 寄存器中的 RELOAD 位置 1 来选择重载模式。在这种情况下，当 NBYTES[7:0] 数据传输完成时，TCR 标志将置 1，并且 SCL 线的低电平将被延展，直到 NBYTES[7:0] 被写入非零值。

- 当 RELOAD=0 且 NBYTES[7:0] 数据传输完成时：
  - 在自动结束模式 (AUTOEND=1) 下，接收到最后一个字节后，将自动发送 NACK 和停止位。
  - 在软件结束模式 (AUTOEND=0) 下，接收到最后一个字节后，将自动发送 NACK，TC 标志将置 1 且 SCL 线的低电平将被延展，以便执行以下软件操作：

    可通过将 I2C_CR2 寄存器中的 START 位置 1 并配置适当的从地址和待传输字节数来请求发送重复起始位。将 START 位置 1 会将 TC 标志清零，并在总线上发送起始位，后跟从地址。

    可通过将 I2C_CR2 寄存器中的 STOP 位置 1 来请求停止位。将 STOP 位置 1 会将 TC 标志清零，并在总线上发送停止位。

图 224. I2C 主接收器的传输序列流程图（N≤255 字节）

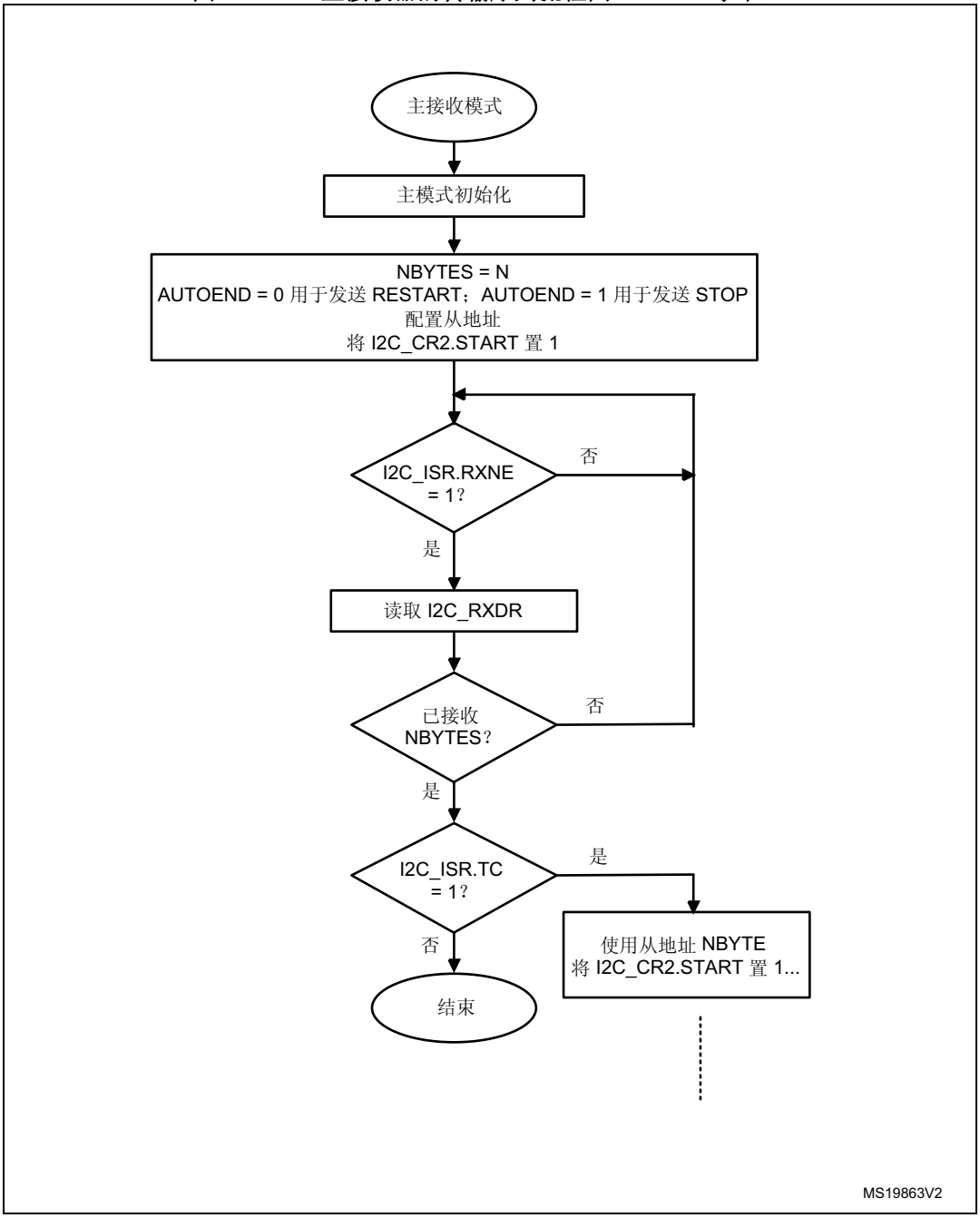


图 225. I2C 主接收器的传输序列流程图（N>255 字节）

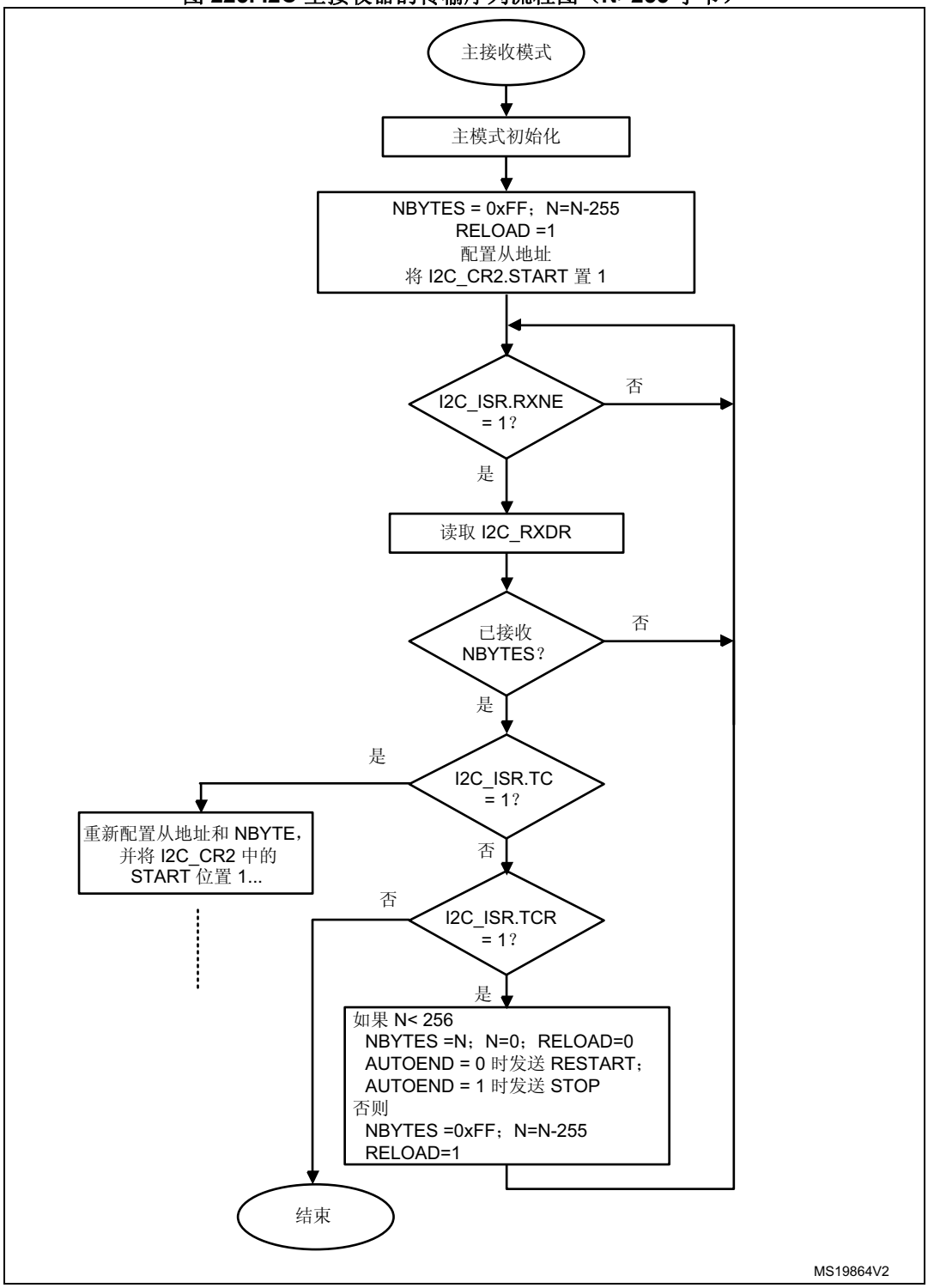
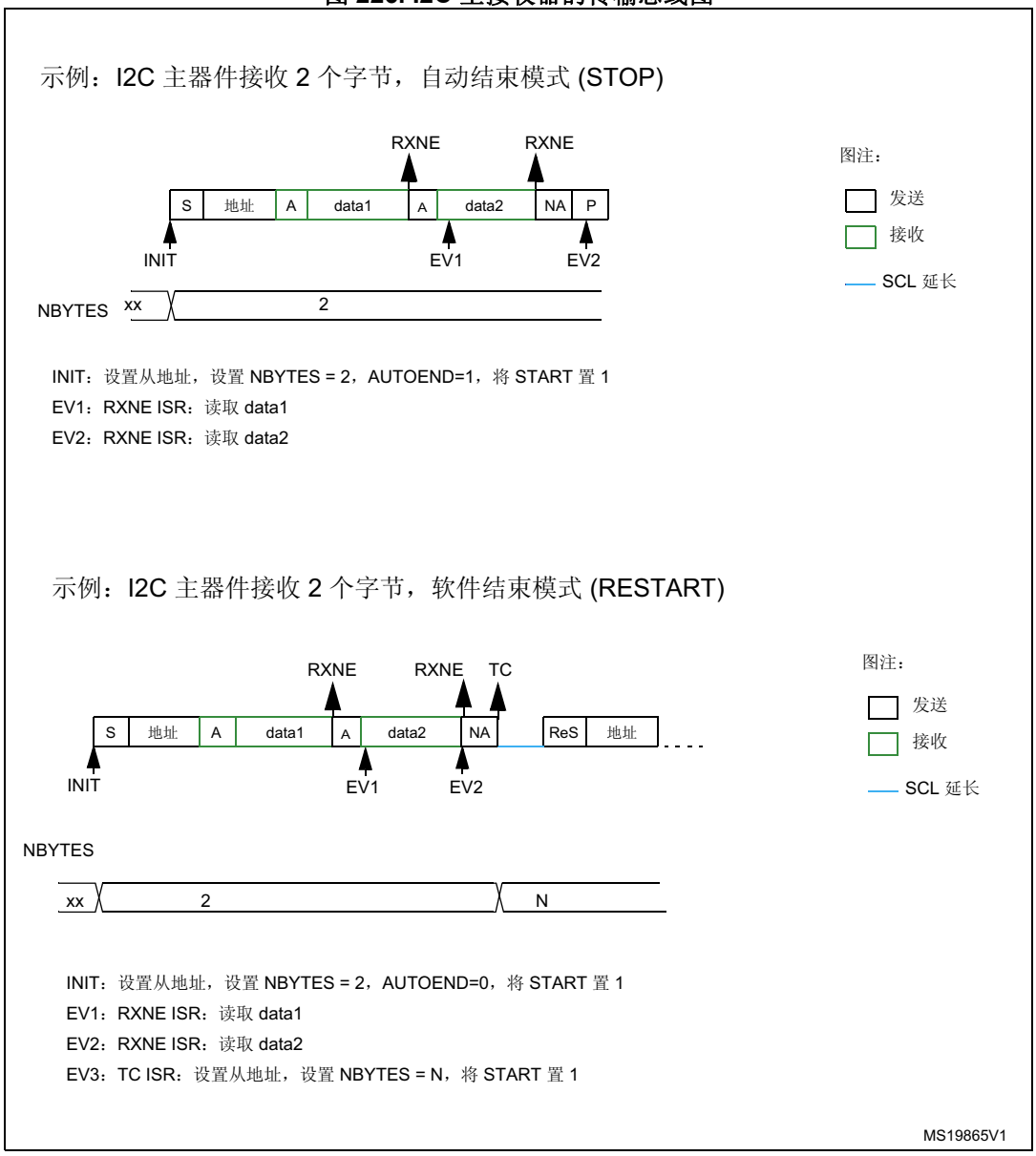


图 226. I2C 主接收器的传输总线图



有关代码示例，请参见 *A.16.7：I2C 主接收器代码示例*。

28.4.10 I2C_TIMINGR 寄存器配置示例

下文各表提供了相应示例，以介绍如何编程 I2C_TIMINGR 才能获得符合 I^2C 规范的时序。要获取更准确的配置值，请参见应用笔记《I^2C 时序配置工具》(AN4235) 及相关软件 STSW-STM32126。

表 113. f_{I2CCLK} = 8 MHz 时的时序设置示例

参数	标准模式 (Sm)		快速模式 (Fm)	超快速模式 (Fm+)
	10 kHz	**100 kHz**	**400 kHz**	**500 kHz**
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
t_{SCLL}	200x250 ns = 50 µs	20x250 ns = 5.0 µs	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
t_{SCLH}	196x250 ns = 49 µs	16x250 ns = 4.0µs	4x125ns = 500ns	4x125 ns = 500 ns
t_{SCL}(1)	约 100 µs(2)	约 10 µs(2)	约 2500 ns(3)	约 2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. 由于 SCL 内部检测存在延时，SCL 周期 t_{SCL} 大于 t_{SCLL} + t_{SCLH}。为 t_{SCL} 提供的值仅用于举例说明。
2. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。t_{SYNC1} + t_{SYNC2} = 1000 ns 时的示例
3. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。t_{SYNC1} + t_{SYNC2} = 750 ns 时的示例
4. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 500 ns。t_{SYNC1} + t_{SYNC2} = 655 ns 时的示例

表 114. f_{I2CCLK} = 16 MHz 时的时序设置示例

参数	标准模式 (Sm)		快速模式 (Fm)	超快速模式 (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	约 100 µs(2)	约 10 µs(2)	约 2500 ns(3)	约 1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. 由于 SCL 内部检测存在延时，SCL 周期 t_{SCL} 大于 t_{SCLL} + t_{SCLH}。为 t_{SCL} 提供的值仅用于举例说明。
2. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 250 ns。t_{SYNC1} + t_{SYNC2} = 1000 ns 时的示例
3. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 250 ns。t_{SYNC1} + t_{SYNC2} = 750 ns 时的示例
4. t_{SYNC1} + t_{SYNC2} 最小值为 4 x t_{I2CCLK} = 250 ns。t_{SYNC1} + t_{SYNC2} = 500 ns 时的示例

28.4.11 SMBus 特性

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

简介

系统管理总线 (SMBus) 是一个双线制接口，各器件可通过它在彼此之间或者与系统的其余部分进行通信。它以 I^2C 的工作原理为基础。SMBus 可针对系统和电源管理相关的任务提供控制总线。

该外设与 SMBUS 规范第 2.0 版兼容 (http://smbus.org)。

系统管理总线规范涉及三类器件。

- 从器件，用于接收或响应命令。
- 主器件，用于发出命令、生成时钟和中止传输。
- 主机，专用的主器件，可提供连接系统 CPU 的主接口。主机必须具有主 - 从器件功能，并且必须支持 SMBus 主机通知协议。系统中只允许存在一个主机。

该外设可配置为主器件或从器件，也可配置为主机。

SMBUS 以 I²C 规范第 2.1 版为基础。

总线协议

任何给定器件都有十一种可用命令协议。器件既可以在这十一种协议中任选其一，也可以使用全部十一种协议进行通信。这十一种协议分别为快速命令、发送字节、接收字节、写入字节、写入字、读取字节、读取字、过程调用、块读取、块写入以及块写入-块读取过程调用。这些协议应通过用户软件实施。

有关这些协议的详细信息，请参见 SMBus 规范第 2.0 版 (http://smbus.org)。

地址解析协议 (ARP)

通过为各个从器件动态分配一个新的唯一地址可解决 SMBus 从地址冲突的问题。为了提供一种机制来针对地址分配隔离各个器件，各器件必须具有唯一的器件标识符 (UDID)。该 128 位数字由软件实现。

该外设支持地址解析协议 (ARP)。通过将 I2C_CR1 寄存器中的 SMBDEN 位置 1 来使能 SMBus 器件默认地址 (0b1100 001)。ARP 命令应通过用户软件实现。

此外，还将在从模式下执行仲裁以支持 ARP。

有关 SMBus 地址解析协议的详细信息，请参见 SMBus 规范第 2.0 版(http://smbus.org)。

接收的命令和数据应答控制

SMBus 接收器必须能够对接收到的每个命令或数据进行否定应答。要在从模式下实现 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从字节控制模式。更多详细信息，请参见*第 645 页的“从器件字节控制模式”*。

主机通知协议

该外设通过将 I2C_CR1 寄存器中的 SMBHEN 位置 1 来支持主机通知协议。在这种情况下，主机将应答 SMBus 主机地址 (0b0001 000)。

使用该协议时，器件作为主器件，而主机作为从器件。

SMBus 报警

器件支持 SMBus ALERT 可选信号。只具备从功能的器件可通过 SMBALERT# 引脚向主机发出信号，指示它想要通信。主机会处理该中断并通过报警响应地址 (0b0001 100) 同时访问所有 SMBALERT# 器件。只有那些将 SMBALERT# 拉到低电平的器件会应答报警响应地址。

如果配置为从器件 (SMBHEN=0)，则通过将 I2C_CR1 寄存器中的 ALERTEN 位置 1 来将 SMBA 引脚拉为低电平。这同时还会使能报警响应地址。

如果配置为主机 (SMBHEN=1)，则当 SMBA 引脚上检测到下降沿且 ALERTEN=1 时，I2C_ISR 寄存器中的 ALERT 标志置 1。如果 I2C_CR1 寄存器中的 ERRIE 位置 1，将生成中断。当 ALERTEN=0 时，即使外部 SMBA 引脚为低电平，ALERT 线也将被视为高电平。

如果无需 SMBus ALERT 引脚，则当 ALERTEN=0 时，SMBA *引脚可用作标准 GPIO*。

数据包错误校验

SMBus 规范中引入了数据包错误校验机制来提高可靠性和通信稳定性。数据包错误校验的实施方式是在每次消息传输结束时附加数据包错误代码 (PEC)。PEC 的计算方式是对所有消息字节（包括地址和读/写位）使用 CRC-8 多项式 $C(x) = x_8 + x^2 + x + 1$。

外设内置了硬件 PEC 计算器，可在接收到的字节与硬件计算的 PEC 不匹配时自动发送否定应答信号。

超时

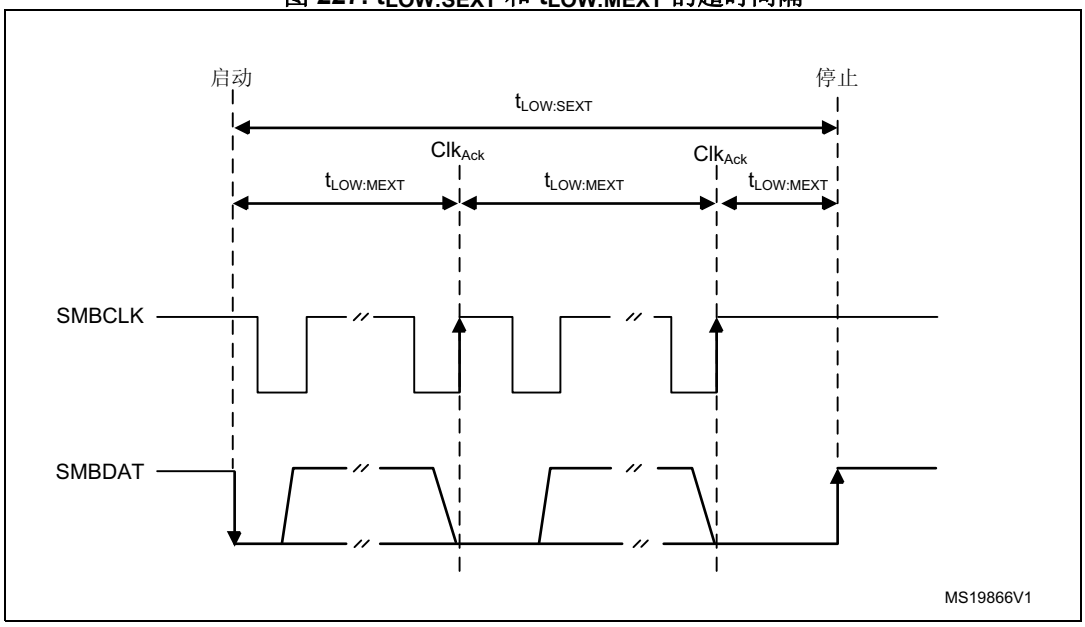
该外设内置了硬件定时器，以便符合 SMBus 规范第 2.0 版中定义的 3 个超时。

表 115. SMBus 超时规范

符号	参数	限值		单位
		最小值	最大值	
$t_{TIMEOUT}$	检测时钟低电平超时	25	35	ms
$t_{LOW:SEXT}$(1)	累积时钟低电平延长时间（从器件）	-	25	ms
$t_{LOW:MEXT}$(2)	累积时钟低电平延长时间（主器件）	-	10	ms

1. $t_{LOW:SEXT}$ 是一段累积时间，即给定从器件在一条消息的最初起始到停止期间时钟信号可延展的时间。其它从器件或主器件也可能延长时钟，进而导致时钟低电平总延长时间超过 $t_{LOW:SEXT}$。因此，测量该参数时该器件应该是全速主器件寻址的唯一器件。
2. $t_{LOW:MEXT}$ 是一段累积时间，即主器件在消息的每个字节（定义为 START 到 ACK、ACK 到 ACK 或 ACK 到 STOP）内时钟信号可延展的时间。从器件或其它主器件也可能延长时钟，进而导致时钟低电平总时间超过 $t_{LOW:MEXT}$（针对给定字节）。因此，测量该参数时该全速主器件只寻址一个从器件。

图 227. $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 的超时间隔



总线空闲检测

如果主器件检测到时钟和数据信号的高电平时间已达 t_{IDLE}（超过 $t_{HIGH,MAX}$），则认为总线空闲（请参见*表 112：I2C-SMBUS 规范时钟时序*）。

该时序参数已考虑如下情况：主器件已动态添加至总线，但可能尚未检测到 SMBCLK 或 SMBDAT 线上的状态转换。在这种情况下，主器件必须等待足够长的时间，以确定当前未进行传输。外设支持硬件总线空闲检测。

28.4.12 SMBus 初始化

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

除了 I2C 初始化之外，还必须进行一些其它的特定初始化，以便执行 SMBus 通信：

接收的命令和数据应答控制（从模式）

SMBus 接收器必须能够对接收到的每个命令或数据进行否定应答。要在从模式下实现 ACK 控制，必须通过将 I2C_CR1 寄存器中的 SBC 位置 1 来使能从字节控制模式。更多详细信息，请参见*第 645 页的“从器件字节控制模式”*。

特定地址（从模式）

必要时应使能特定的 SMBus 地址。更多详细信息，请参见*第 668 页的“总线空闲检测”*。

- 通过将 I2C_CR1 寄存器中的 SMBDEN 位置 1 来使能 SMBus 器件默认地址 (0b1100001)。
- 通过将 I2C_CR1 寄存器中的 SMBHEN 位置 1 来使能 SMBus 主机地址 (0b0001 000)。
- 通过将 I2C_CR1 寄存器中的 ALERTEN 位置 1 来使能报警响应地址 (0b0001100)。

数据包错误校验

通过将 I2C_CR1 寄存器中的 PECEN 位置 1 来使能 PEC 的计算。然后，借助硬件字节计数器（I2C_CR2 寄存器中的 NBYTES[7:0]）来管理 PEC 传输。使能 I2C 之前，必须配置 PECEN 位。

PEC 传输由硬件字节计数器来管理，因此在从模式下连接 SMBus 时必须将 SBC 位置 1。当 PECBYTE 位置 1 且 RELOAD 位清零时，传输完 NBYTES-1 字节的数据后会传输 PEC。如果 RELOAD 置 1，PECBYTE 将不起作用。

注意： 使能 I2C 时，不允许更改 PECEN 配置。

表 116. 带 PEC 的 SMBUS 配置

模式	SBC 位	RELOAD 位	AUTOEND 位	PECBYTE 位
主 Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
主 Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
从 Tx/Rx + PEC	1	0	x	1

超时检测

将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 和 TEXTEN 位置 1 来使能超时检测。定时器必须按如下方式编程：即在 SMBus 规范第 2.0 版规定的时间最大值之前检测出超时情况。

- $t_{TIMEOUT}$ 检查

  要使能 $t_{TIMEOUT}$ 检查，必须将 12 位 TIMEOUTA[11:0] 位编程为定时器重载值，以检查 $t_{TIMEOUT}$ 参数。必须将 TIDLE 位配置为“0”，以检测 SCL 低电平超时。

  然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

  如果 SCL 的低电平持续时间超过 (TIMEOUTA+1) x 2048 x t_{I2CCLK}，I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

  请参见*表 117：不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 $t_{TIMEOUT}$ = 25 ms）*。

注意： TIMEOUTEN 位置 1 时，不允许更改 TIMEOUTA[11:0] 位和 TIDLE 位的配置。

- $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 检查

  必须根据外设配置为主器件还是从器件来配置 TIMEOUTB 定时器，以便为从器件校验 $t_{LOW:SEXT}$，为主器件校验 $t_{LOW:MEXT}$。由于标准只规定了最大值，用户可以为这两个参数选择相同的值。

  然后，通过将 I2C_TIMEOUTR 寄存器中的 TEXTEN 位置 1 来使能定时器。

  如果 SMBus 外设延展 SCL 的累积时间超过 (TIMEOUTB+1) x 2048 x t_{I2CCLK}，并且达到*第 668 页的“总线空闲检测”*一节给出的超时间隔，则 I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

  请参见*表 118：不同 I2CCLK 频率下的 TIMEOUTB 设置示例*。

注意： TEXTEN 位置 1 时，不允许更改 TIMEOUTB 配置。

总线空闲检测

要使能 t_{IDLE} 检查，必须将 12 位 TIMEOUTA[11:0] 字段编程为定时器重载值，以获取 t_{IDLE} 参数。必须将 TIDLE 位配置为“1”，以检测 SCL 和 SDA 高电平超时。

然后，通过将 I2C_TIMEOUTR 寄存器中的 TIMOUTEN 位置 1 来使能定时器。

如果 SCL 和 SDA 线的高电平持续时间超过 (TIMEOUTA+1) x 4 x t_{I2CCLK}，I2C_ISR 寄存器中的 TIMEOUT 标志将置 1。

请参见*表 119：不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 t_{IDLE} = 50 µs）*。

注意： TIMEOUTEN 置 1 时，不允许更改 TIMEOUTA 和 TIDLE 配置。

28.4.13 SMBus：I2C_TIMEOUTR 寄存器配置示例

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

- 将 $t_{TIMEOUT}$ 的最大持续时间配置为 25 ms：

表 117. 不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 $t_{TIMEOUT}$ = 25 ms）

f_{I2CCLK}	TIMEOUTA[11:0] 位	TIDLE 位	TIMEOUTEN 位	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms

- 将 $t_{LOW:SEXT}$ 和 $t_{LOW:MEXT}$ 的最大持续时间配置为 8 ms：

表 118. 不同 I2CCLK 频率下的 TIMEOUTB 设置示例

f_{I2CCLK}	TIMEOUTB[11:0] 位	TEXTEN 位	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
32 MHz	0x7C	1	125 x 2048 x 31.25 ns = 8 ms

- 将 t_{IDLE} 的最大持续时间配置为 50 µs

表 119. 不同 I2CCLK 频率下的 TIMEOUTA 设置示例（最大 t_{IDLE} = 50 µs）

f_{I2CCLK}	TIMEOUTA[11:0] 位	TIDLE 位	TIMEOUTEN 位	t_{TIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
32 MHz	0x18F	1	1	400 x 4 x 31.25 ns = 50 µs

28.4.14 SMBus 从模式

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

除了 I2C 从模式传输管理（请参见*第 28.4.8 节：I2C 从模式*）之外，还提供了一些额外的软件流程图来支持 SMBus。

SMBus 从发送器

在 SMBus 模式下使用 IP 时，必须将 SBC 编程为“1”，以便在完成已编程数据字节数的传输后进行 PEC 传输。当 PECBYTE 位置 1 时，NBYTES[7:0] 中编程的字节数包含 PEC 传输。在这种情况下，总 TXIS 中断数为 NBYTES-1，如果主器件在完成 NBYTES-1 字节的数据传输后请求传输额外的字节，则将自动发送 I2C_PECR 寄存器的内容。

注意： 当 RELOAD 位置 1 时，PECBYTE 位将不起作用。

图 228. SMBus 从发送器的传输序列流程图（N 字节 + PEC）

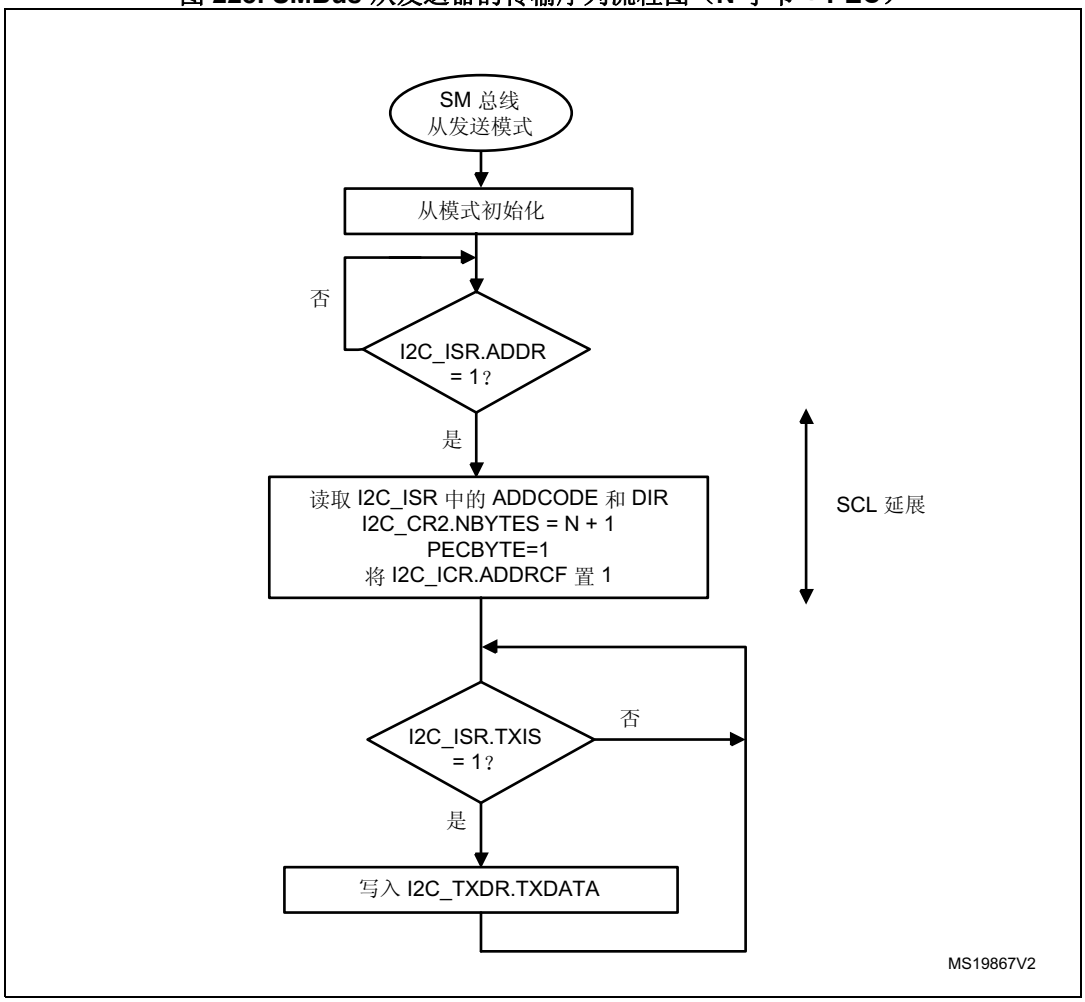
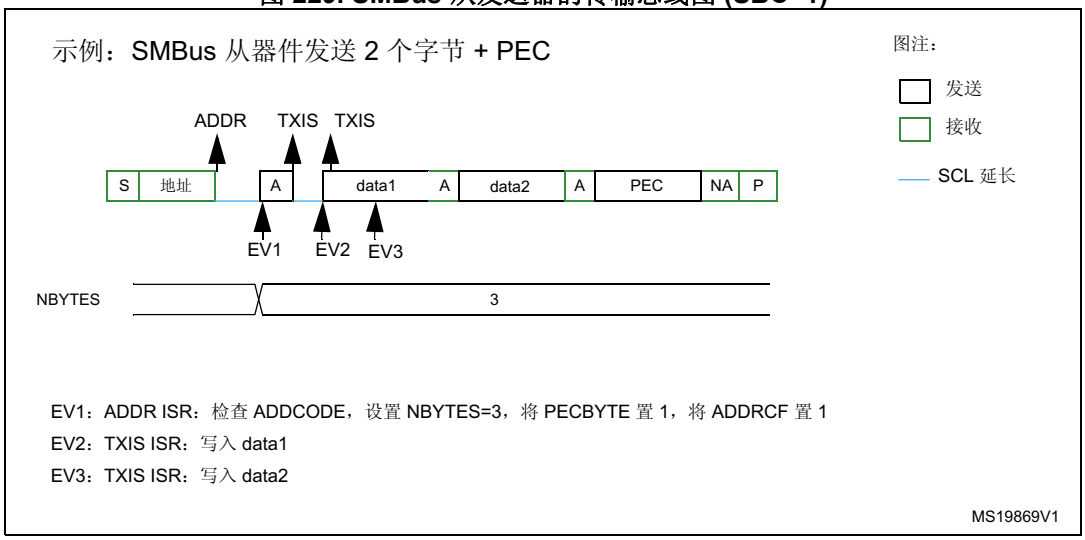


图 229. SMBus 从发送器的传输总线图 (SBC=1)



SMBus 从接收器

在 SMBus 模式下使用 I2C 时，必须将 SBC 编程为“1”，以便在完成已编程数据字节数的传输后进行 PEC 校验。要对每个字节进行 ACK 控制，必须选择重载模式 (RELOAD=1)。更多详细信息，请参见*第 645 页的“从器件字节控制模式”*。

要校验 PEC 字节，必须将 RELOAD 位清零并将 PECBYTE 位置 1。在这种情况下，当接收到 NBYTES-1 字节的数据后，接收的下一个字节将与内部 I2C_PECR 寄存器的内容作比较。如果比较不匹配，则将自动生成 NACK 信号；如果比较匹配，则将自动生成 ACK 信号，而与 ACK 位的值无关。PEC 字节一经接收，便会像任何其它数据一样复制到 I2C_RXDR 寄存器中，并且 RXNE 标志将置 1。

当 PEC 不匹配时，PECERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

如果无需 ACK 软件控制，用户可编程 PECBYTE=1，在同一写操作下，将 NBYTES 编程为连续接收的字节数。接收到 NBYTES-1 字节的数据后，会将接收的下一个字节视为 PEC 进行校验。

注意： 当 RELOAD 位置 1 时，PECBYTE 位将不起作用。

图 230. SMBus 从接收器的传输序列流程图（N 字节 + PEC）

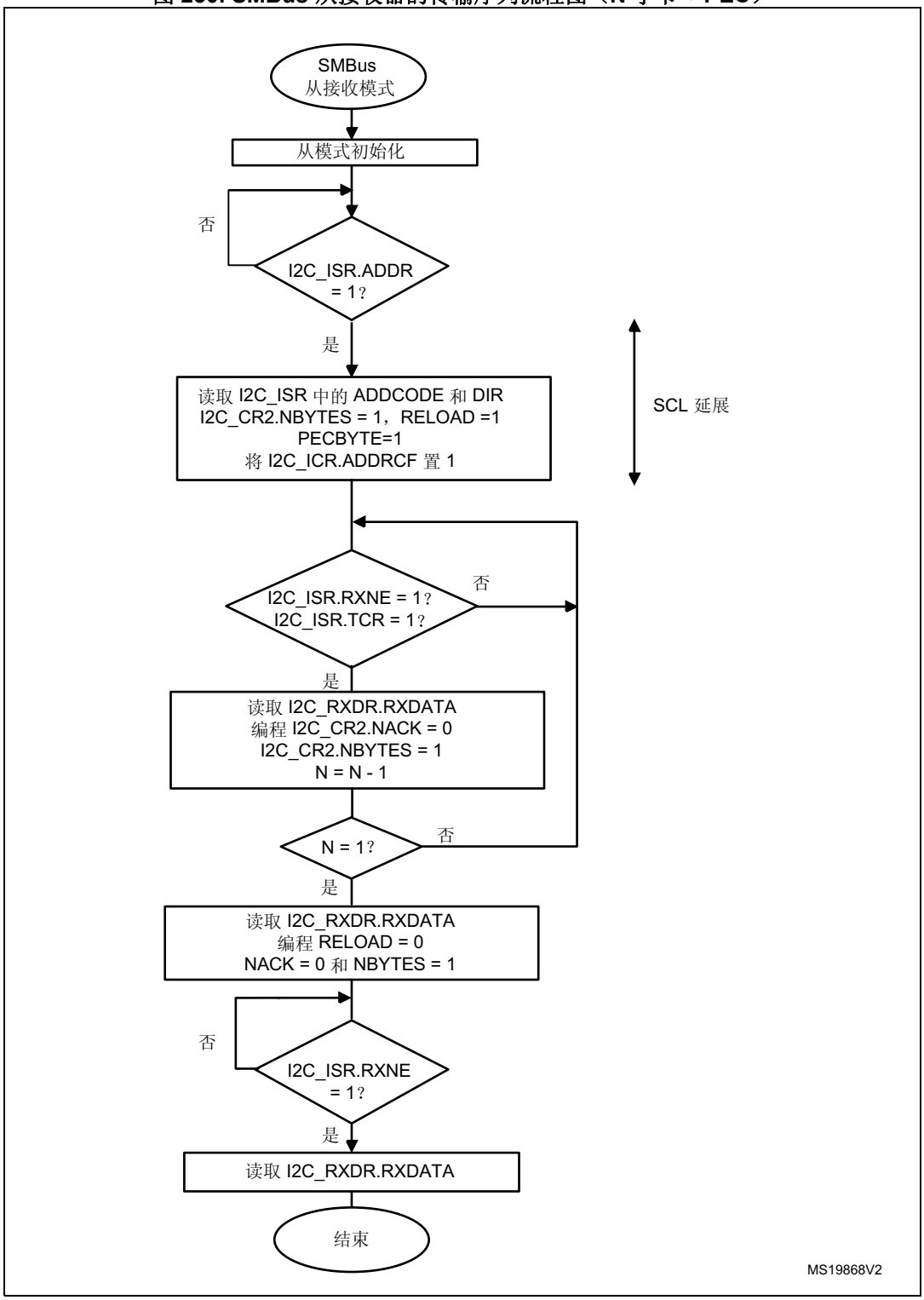
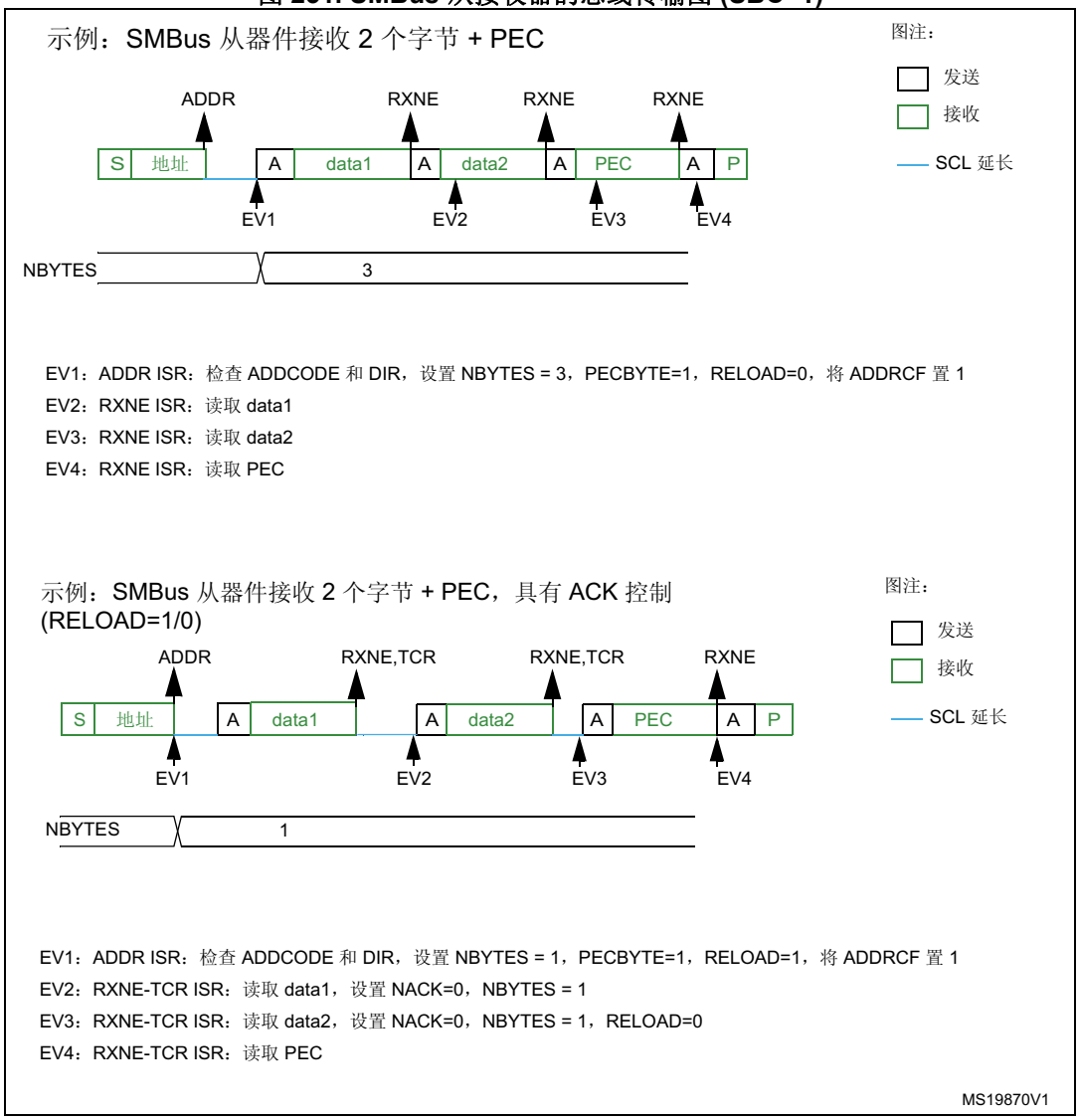


图 231. SMBus 从接收器的总线传输图 (SBC=1)



仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

除了 I2C 主模式传输管理（请参见*第 28.4.9 节：I2C 主模式*）之外，还提供了一些额外的软件流程图来支持 SMBus。

SMBus 主发送器

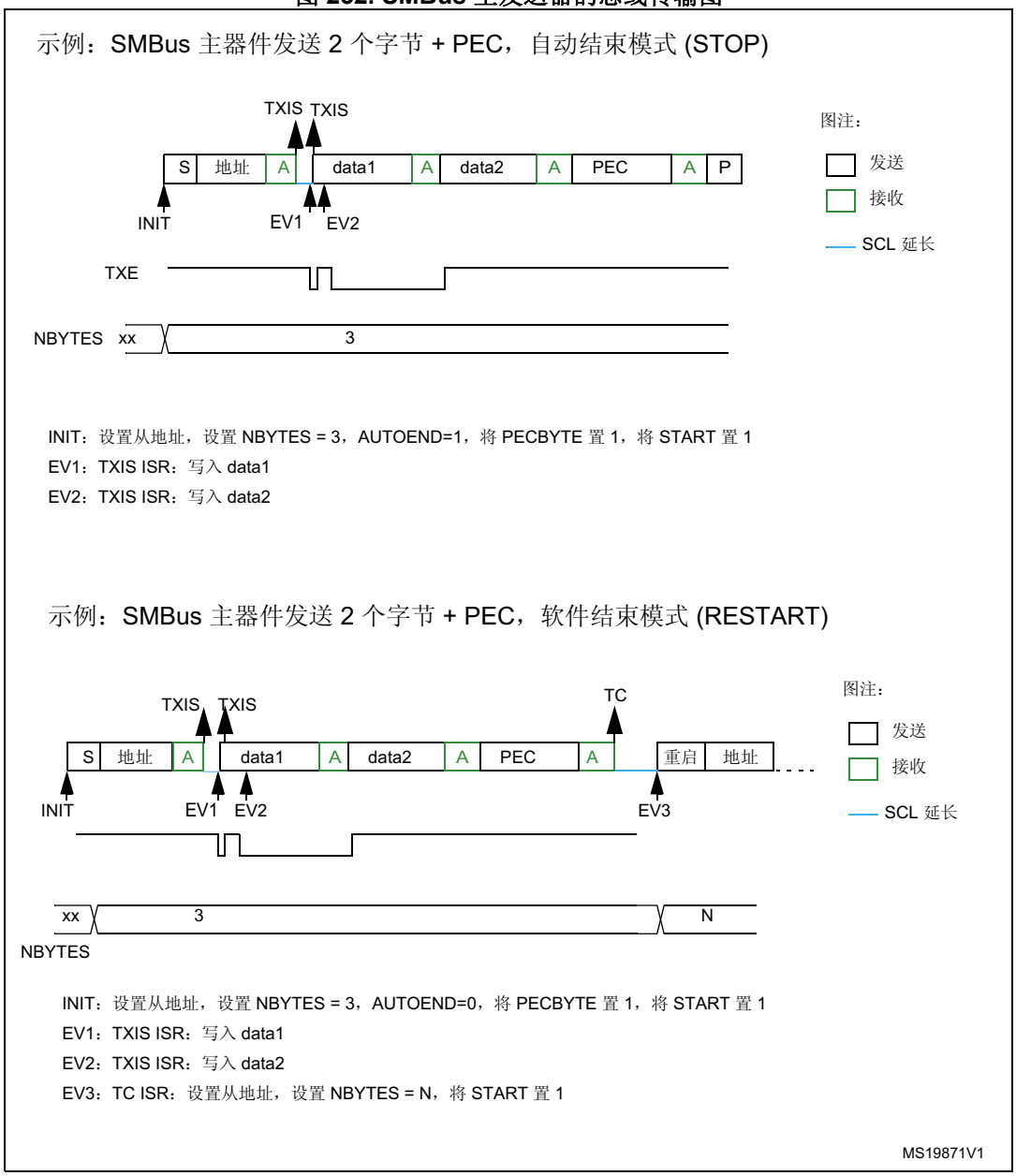
当 SMBus 主器件想要发送 PEC 时，必须在 START 位置 1 前，将 PECBYTE 位置 1 并在 NBYTES[7:0] 字段中设置字节数。在这种情况下，总 TXIS 中断数为 NBYTES-1。因此，如果 PECBYTE 位在 NBYTES=0x1 时置 1，则将自动发送 I2C_PECR 寄存器的内容。

如果 SMBus 主器件想要在 PEC 后发送停止位，则应选择自动结束模式 (AUTOEND=1)。在这种情况下，传输 PEC 后将自动发送停止位。

如果 SMBus 主器件想要在 PEC 后发送重复起始位，则必须选择软件模式 (AUTOEND=0)。在这种情况下，发送 NBYTES-1 字节的数据后，将发送 I2C_PECR 寄存器的内容，TC 标志将在传输完 PEC 之后置 1，SCL 线的低电平时间将延长。必须在 TC 中断子程序中设置重复起始位。

注意： 当 RELOAD 位置 1 时，PECBYTE 位将不起作用。

图 232. SMBus 主发送器的总线传输图



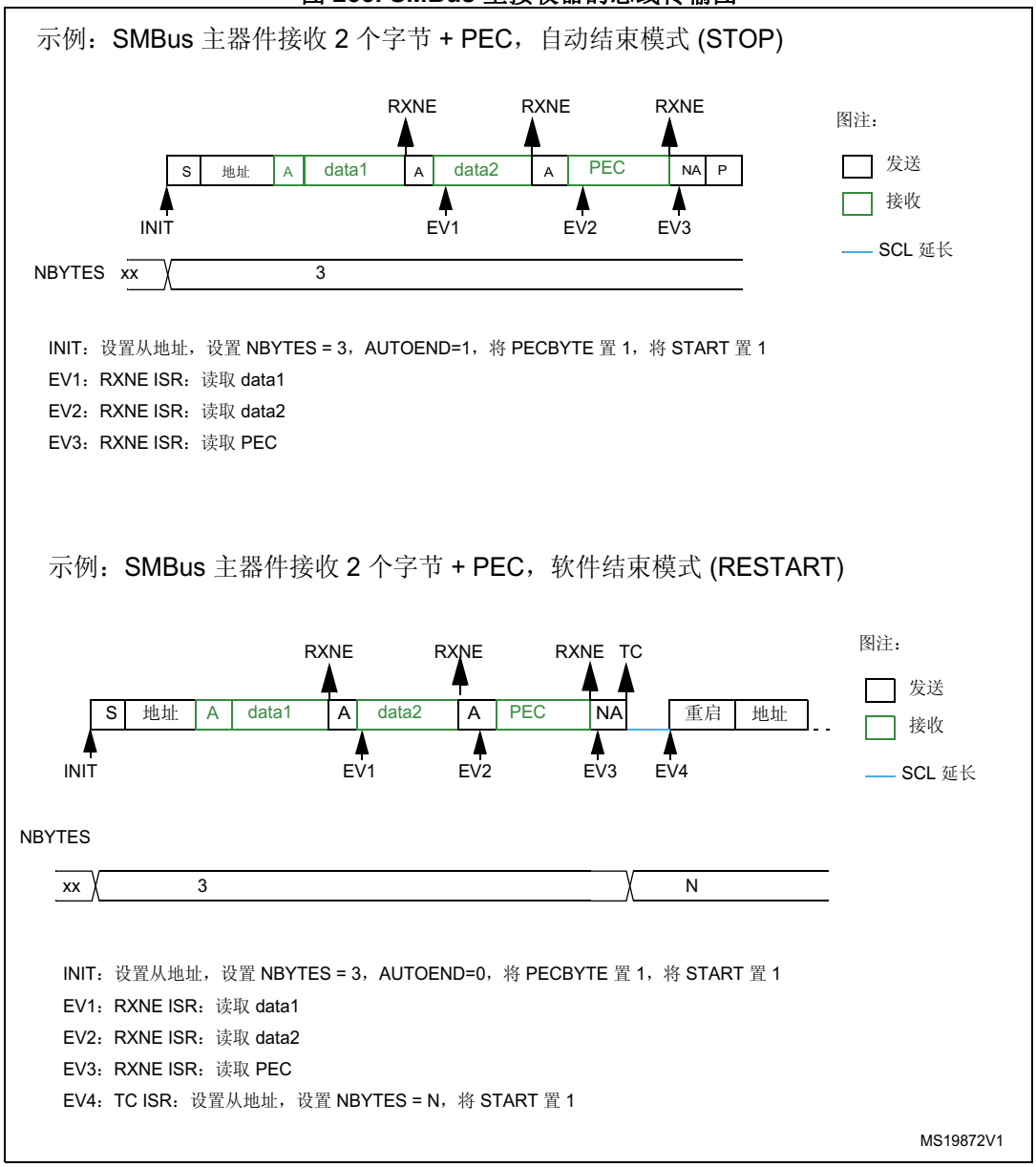
SMBus 主接收器

当 SMBus 主器件想要接收 PEC，并在传输结束后接收 STOP 时，可选择自动结束模式 (AUTOEND=1)。将 START 位置 1 之前，必须将 PECBYTE 位置 1 并设置从地址。在这种情况下，当接收到 NBYTES-1 字节的数据后，将自动使用 I2C_PECR 寄存器的内容对接收的下一个字节进行校验。PEC 字节（其后跟有停止位）将得到 NACK 响应。

当 SMBus 主接收器想要接收 PEC 字节，并且在传输结束后接收重复起始位时，必须选择软件模式 (AUTOEND=0)。将 START 位置 1 之前，必须将 PECBYTE 位置 1 并设置从地址。在这种情况下，当接收到 NBYTES-1 字节的数据后，将自动使用 I2C_PECR 寄存器的内容对接收的下一个字节进行校验。接收到 PEC 字节后，TC 标志将置 1，SCL 线的低电平时间将延长。可以在 TC 中断子程序中设置重复起始位。

注意： 当 RELOAD 位置 1 时，PECBYTE 位将不起作用。

图 233. SMBus 主接收器的总线传输图



28.4.15 地址匹配时从停止模式唤醒

仅当支持从停止模式唤醒功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

被寻址时，I2C 能够将 MCU 从停止模式唤醒（APB 时钟关断）。支持所有寻址模式。

将 I2C_CR1 寄存器中的 WUPEN 位置 1，可以使能从停止模式唤醒功能。对于 I2CCLK，必须选择 HSI 振荡器作为时钟源，以便从停止模式唤醒。

在停止模式中，HSI 关闭。当检测到 START 时，I2C 接口将 HSI 接通，并延长 SCL 低电平持续时间，直到 HSI 唤醒。

HSI 随后用来接收地址。

地址匹配的情况下，MCU 唤醒时间内，I2C 延长 SCL 的低电平持续时间。通过软件清除 ADDR 标志后，此延长操作被释放，传输正常进行。

如果地址不匹配，HSI 再次关断，MCU 不被唤醒。

注： *如果 I2C 时钟是系统时钟，或者 WUPEN = 0，则接收到 START 后，HSI 振荡器不会接通。*

只有 ADDR 中断能够唤醒 MCU。因此，当 I2C 以主器件身份或在 ADDR 标志置 1 后以被寻址从器件身份执行传输时，不要进入停止模式。将 ADDR 中断程序中的 SLEEPDEEP 位清零，然后仅在 STOPF 标志置 1 后将其置 1，可对此进行管理。

注意： 数字滤波器与从停止模式唤醒功能不兼容。如果 DNF 位不等于 0，则将 WUPEN 位置 1 将不起任何作用。

注意： 只有当 I2C 时钟源为 HSI 振荡器时，该功能才可用。

注意： 必须使能时钟延长 (NOSTRETCH=0) 才能确保从停止模式唤醒功能正常工作。

注意： 如果禁止从停止模式唤醒 (WUPEN=0)，则在进入停止模式前必须禁止 I2C 外设 (PE=0)。

28.4.16 错误条件

以下错误条件可能导致通信失败。

总线错误 (BERR)

当检测到起始位或停止位但不位于第 9N 个 SCL 时钟脉冲之后时，会检测到总线错误。当 SDA 边沿出现且 SCL 为高电平时，会检测到起始或停止位。

只有当 I2C 在传输过程中用作主器件或被寻址为从器件时（即未处于从模式下的地址阶段），才会将总线错误标志置 1。

在从模式下误检测到起始位或重复起始位时，I2C 会像接收到正确的起始位一样进入地址识别状态。

检测到总线错误时，I2C_ISR 寄存器中的 BERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

仲裁丢失 (ARLO)

当 SDA 线上发送高电平但在 SCL 上升沿却采样到低电平时，会检测到仲裁丢失。

- 在主模式下，将在地址阶段、数据阶段和数据应答阶段检测到仲裁丢失。在这种情况下，SDA 线和 SCL 线被释放，起始控制位由硬件清零，主器件自动切换为从模式。
- 在从模式下，将在数据阶段和数据应答阶段检测到仲裁丢失。在这种情况下，传输停止，SCL 和 SDA 线被释放。

检测到仲裁丢失时，I2C_ISR 寄存器中的 ARLO 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

上溢/下溢错误 (OVR)

当满足 NOSTRETCH=1 和以下条件时，将在从模式下检测到上溢或下溢错误：

- 在接收过程中接收到一个新的字节，但 RXDR 寄存器的值还未被读取。接收的新字节丢失，自动发送 NACK 来响应新字节。
- 在发送过程中：
  - 当 STOPF=1 且应发送第一个数据字节时。TXE=0 时发送 I2C_TXDR 寄存器的内容，否则发送 0xFF。
  - 应发送一个新字节但尚未向 I2C_TXDR 寄存器写入数据时，将发送 0xFF。

检测到上溢或下溢错误时，I2C_ISR 寄存器中的 OVR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

数据包错误校验错误 (PECERR)

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

当接收到的 PEC 字节与 I2C_PECR 寄存器的内容不匹配时，将检测到 PEC 错误。接收到错误的 PEC 后，将自动发送 NACK。

检测到 PEC 错误时，I2C_ISR 寄存器中的 PECERR 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

超时错误 (TIMEOUT)

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

满足以下任何条件均会出现超时错误：

- TIDLE=0 且 SCL 的低电平持续时间达到 TIMEOUTA[11:0] 位中定义的时间：这用于检测 SMBus 超时。
- TIDLE=1 且 SDA 和 SCL 的高电平持续时间达到 TIMEOUTA[11:0] 位中定义的时间：这用于检测总线空闲情况。
- 主器件的累积时钟低电平延长时间达到了 TIMEOUTB[11:0] 位中定义的时间（SMBus $t_{LOW:MEXT}$ 参数）
- 从器件的累积时钟低电平延长时间达到了 TIMEOUTB[11:0] 位中定义的时间（SMBus $t_{LOW:SEXT}$ 参数）

当在主模式下检测到超时时，将自动发送停止位。

当在从模式下检测到超时时，将自动释放 SDA 和 SCL 线。

检测到超时错误时，I2C_ISR 寄存器中的 TIMEOUT 标志将置 1，如果 I2C_CR1 寄存器中的 ERRIE 位置 1，还将生成中断。

报警 (ALERT)

仅当支持 SMBus 功能时，才涉及本节内容。请参见*第 28.3 节：I2C 特性实现*。

当 I2C 接口配置为主机 (SMBHEN=1)、使能了报警引脚检测 (ALERTEN=1) 并且在 SMBA 引脚上检测到下降沿时，ALERT 标志将置 1。如果 I2C_CR1 寄存器中的 ERRIE 位置 1，将生成中断。

28.4.17 DMA 请求

使用 DMA 进行发送

将 I2C_CR1 寄存器中的 TXDMAEN 位置 1 可以使能 DMA（直接存储器访问）进行发送。当 TXIS 位置 1 时，数据将从由 DMA 外设配置的 SRAM 区（请参见*第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）装载进 I2C_TXDR 寄存器。

只有数据字节采用 DMA 进行传输。

- 在主模式下：初始化、从地址、方向、字节数和起始位均由软件编程（发送的从地址无法通过 DMA 传输）。当所有数据均通过 DMA 传输时，必须在起始位置 1 之前初始化 DMA。传输结束由 NBYTES 计数器来管理。请参见*第 656 页的“主发送器”*。

有关代码示例，请参见 *A.16.8：将 I2C 配置为主模式以通过 DMA 进行发送的代码示例*。

- 在从模式下：
  - 当 NOSTRETCH=0 时，如果所有数据均通过 DMA 传输，则必须在地址匹配事件之前（或清零 ADDR 之前在 ADDR 中断子程序中）初始化 DMA。
  - 当 NOSTRETCH=1 时，必须在地址匹配事件之前初始化 DMA。
- 支持 SMBus 时：PEC 传输由 NBYTES 计数器管理。请参见*第 671 页的“SMBus 从发送器”*和*第 674 页的“SMBus 主发送器”*。

注：如果使用 DMA 进行发送，则无需使能 TXIE 位。

使用 DMA 进行接收

将 I2C_CR1 寄存器中的 RXDMAEN 位置 1 可以使能 DMA（直接存储器访问）进行接收。当 RXNE 位置 1 时，数据将从 I2C_RXDR 寄存器装载进由 DMA 外设配置的 SRAM 区（请参见*第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）。只有数据字节（包括 PEC）采用 DMA 进行传输。

- 在主模式下：初始化、从地址、方向、字节数和起始位均由软件编程。当所有数据均通过 DMA 传输时，必须在起始位置 1 之前初始化 DMA。传输结束由 NBYTES 计数器来管理。有关代码示例，请参见 *A.16.9：将 I2C 配置为从模式以通过 DMA 进行接收的代码示例*。
- 在从模式下，当 NOSTRETCH=0 时，如果所有数据均通过 DMA 传输，则必须在地址匹配事件之前（或清零 ADDR 标志之前在 ADDR 中断子程序中）初始化 DMA。
- 如果支持 SMBus（请参见*第 28.3 节：I2C 特性实现*）：PEC 传输由 NBYTES 计数器管理。请参见*第 672 页的“SMBus 从接收器”*和*第 676 页的“SMBus 主接收器”*。

注：如果使用 DMA 进行接收，则无需使能 RXIE 位。

28.4.18 调试模式

当微控制器进入调试模式时（内核停止），SMBus 超时定时器会根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位选择继续正常工作还是停止工作。

28.5 I2C 低功耗模式

表 120. 低功耗模式

模式	说明
睡眠	对 I2C 通信无影响 I2C 中断可使器件退出睡眠模式。
停止	I2C 模块的寄存器内容仍被保持。
待机	I2C 外设掉电，退出待机模式后必须重新初始化。

28.6 I2C 中断

下表给出了 I2C 中断请求列表。

表 121. I2C 中断请求

中断事件	事件标志	事件标志/ 中断清除方法	中断使能控制位
接收缓冲区非空 (Receive buffer not empty)	RXNE	读取 I2C_RXDR 寄存器	RXIE
发送缓冲区中断状态	TXIS	写入 I2C_TXDR 寄存器	TXIE
停止位检测中断标志	STOPF	写入 STOPCF=1	STOPIE
传输完成等待重载	TCR	写入 I2C_CR2 （NBYTES[7:0] ≠ 0）	TCIE
传输完成	TC	写入 START=1 或 STOP=1	
地址匹配	ADDR	写入 ADDRCF=1	ADDRIE
接收到 NACK 应答	NACKF	写入 NACKCF=1	NACKIE
总线错误 (Bus error)	BERR	写入 BERRCF=1	ERRIE
仲裁丢失	ARLO	写入 ARLOCF=1	
上溢/下溢 (Overrun/Underrun)	OVR	写入 OVRCF=1	
PEC 错误	PECERR	写入 PECERRCF=1	
超时/t_{LOW} 错误	TIMEOUT	写入 TIMEOUTCF=1	
SMBus 报警	ALERT	写入 ALERTCF=1	

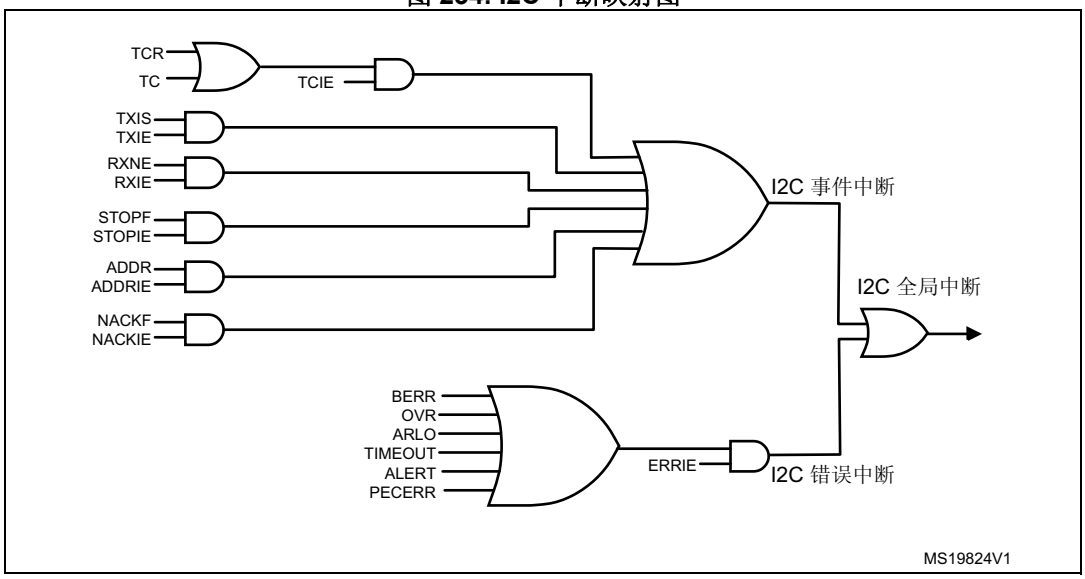
根据产品实现的不同，上述所有中断既可以共享同一个中断向量（I2C 全局中断），也可以分配到 2 个不同的中断向量上（I2C 事件中断和 I2C 错误中断）。有关详细信息，请参见*表 51：向量列表*。

要使能 I2C 中断，需按照以下顺序操作：

1. 配置 NVIC 中的 I2C IRQ 通道并将其使能。
2. 配置 I2C 以生成中断。

I2C 唤醒事件连接到 EXTI 控制器（请参见*第 13.5 节：EXTI 寄存器*）。

图 234. I2C 中断映射图



28.7 I2C 寄存器

有关寄存器说明中使用的缩写，请参见第 50 页的第 1.1 节。

外设寄存器按字（32 位）进行访问。

28.7.1 控制寄存器 1 (I2C_CR1)

Control register 1

偏移地址：0x00

复位值：0x0000 0000

访问：无等待周期，正在对该寄存器执行写访问时出现另一个写访问的情况除外。在这种情况下，会在第二个写访问中插入等待周期，直到前一个写访问完成为止。第二个写访问的延时最长可达 2 x PCLK1 + 6 x I2CCLK。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 保留，必须保持复位值。

位 23 **PECEN**：PEC 使能 (PEC enable)

0：禁止 PEC 计算

1：使能 PEC 计算

注：　如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 22 **ALERTEN**：SMBus 报警使能 (SMBus alert enable)

从机模式 (SMBHEN=0)：

0：将 SMBA 引脚释放为高电平并禁止报警响应地址头：0001100x 后跟 NACK。

1：将 SMBA 引脚驱动为低电平并使能报警响应地址头：0001100x 后跟 ACK。

主机模式 (SMBHEN=1)：

0：不支持 SMBus 报警引脚 (SMBA)。

1：支持 SMBus 报警引脚 (SMBA)。

注：　当 ALERTEN=0 时，SMBA 引脚可用作标准 GPIO。

如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现.。

位 21 **SMBDEN**：SMBus 器件默认地址使能 (SMBus Device Default address enable)

0：禁止器件默认地址。不对地址 0b1100001x 应答。

1：使能器件默认地址。对地址 0b1100001x 应答。

注：　如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 20 **SMBHEN**：SMBus 主机地址使能 (SMBus Host address enable)

0：禁止主机地址。不对地址 0b0001000x 应答。

1：使能主机地址。对地址 0b0001000x 应答。

注： 如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 19 **GCEN**：广播呼叫使能 (General call enable)

0：禁止广播呼叫。不对地址 0b00000000 应答。

1：使能广播呼叫。对地址 0b00000000 应答。

位 18 **WUPEN**：从停止模式唤醒使能 (Wakeup from Stop mode enable)

0：禁止从停止模式唤醒。

1：使能从停止模式唤醒。

注： 如果不支持从停止模式唤醒功能，该位保留并由硬件强制清零。请参见第 28.3 节：I2C 特性实现。

注： 只有当 DNF = “0000”时，WUPEN 才能置 1

位 17 **NOSTRETCH**：时钟延长禁止 (Clock stretching disable)

该位用于在从模式下禁止时钟延长。它在主模式下必须保持清零。

0：使能时钟延长

1：禁止时钟延长

注： 该位只能在 I2C 禁止时 (PE = 0) 编程。

位 16 **SBC**：从设备模式下的字节控制 (Slave byte control)

该位用于在从设备模式下使能硬件字节控制。

0：禁止从设备模式下的字节控制

1：使能从设备模式下的字节控制

位 15 **RXDMAEN**：DMA 接收请求使能 (DMA reception requests enable)

0：禁止 DMA 接收请求

1：使能 DMA 接收请求

位 14 **TXDMAEN**：DMA 发送请求使能 (DMA transmission requests enable)

0：禁止 DMA 发送请求

1：使能 DMA 发送请求

位 13 保留，必须保持复位值。

位 12 **ANFOFF**：模拟噪声滤波器关闭 (Analog noise filter OFF)

0：使能模拟噪声滤波器

1：禁止模拟噪声滤波器

注： 该位只能在 I2C 禁止时 (PE = 0) 编程。

位 11:8 **DNF[3:0]**：数字噪声滤波器 (Digital noise filter)

这些位用于配置 SDA 和 SCL 输入端的数字噪声滤波器。数字滤波器可滤除脉宽 DNF[3:0] * t_{I2CCLK} 以下的尖峰

0000：禁止数字滤波器

0001：使能数字滤波器，可滤除的噪声尖峰脉宽可达 1 t_{I2CCLK}

...

1111：使能数字滤波器，可滤除的噪声尖峰脉宽可达 15 t_{I2CCLK}

注： 如果模拟滤波器也已使能，数字滤波将叠加在模拟滤波之上。

该滤波器只能在 I2C 禁止时 (PE = 0) 编程。

位 7 **ERRIE**：错误中断使能 (Error interrupts enable)

0：禁止错误检测中断
1：使能错误检测中断

注：以下任一错误均会生成中断：
仲裁丢失 (ARLO)
总线错误检测 (BERR)
上溢/ 下溢 (OVR)
超时检测 (TIMEOUT)
PEC 错误检测 (PECERR)
报警引脚事件检测 (ALERT)

位 6 **TCIE**：传输完成中断使能 (Transfer complete interrupt enable)

0：禁止传输完成中断
1：使能传输完成中断

注：以下任一事件均会生成中断：
传输完成 (TC)
传输完成等待重载 (TCR)

位 5 **STOPIE**：停止位检测中断使能 (STOP detection Interrupt enable)

0：禁止停止位检测 (STOPF) 中断
1：使能停止位检测 (STOPF) 中断

位 4 **NACKIE**：接收到否定应答中断使能 (Not acknowledge received Interrupt enable)

0：禁止接收到否定应答 (NACKF) 中断
1：使能接收到否定应答 (NACKF) 中断

位 3 **ADDRIE**：地址匹配中断使能（仅从模式）(Address match Interrupt enable (slave only))

0：禁止地址匹配 (ADDR) 中断
1：使能地址匹配 (ADDR) 中断

位 2 **RXIE**：RX 中断使能 (RX Interrupt enable)

0：禁止接收 (RXNE) 中断
1：使能接收 (RXNE) 中断

位 1 **TXIE**：TX 中断使能 (TX Interrupt enable)

0：禁止发送 (TXIS) 中断
1：使能发送 (TXIS) 中断

位 0 **PE**：外设使能 (Peripheral enable)

0：禁止外设
1：使能外设

注：当 PE=0 时，将释放 I2C SCL 线和 SDA 线。内部状态机和状态位均恢复为复位值。清零时，PE 必须保持低电平状态至少 3 个 APB 时钟周期。

28.7.2 控制寄存器 2 (I2C_CR2)

Control register 2

偏移地址：0x04

复位值：0x0000 0000

访问：无等待周期，正在对该寄存器执行写访问时出现另一个写访问的情况除外。在这种情况下，会在第二个写访问中插入等待周期，直到前一个写访问完成为止。第二个写访问的延时最长可达 2 x PCLK1 + 6 x I2CCLK。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_W RN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

位 31:27 保留，必须保持复位值。

位 26 **PECBYTE**：数据包错误校验字节 (Packet error checking byte)

此位由软件置 1，并可在 PEC 传输完成时、接收到停止条件或匹配地址时、或者 PE=0 时由硬件清零。

0：不传输 PEC。

1：请求 PEC 发送/接收。

注：向该位写入“0”不起作用。

当 RELOAD 置 1 时，该位不起作用。

当 SBC=0 时，该位在从模式下不起作用。

如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 25 **AUTOEND**：自动结束模式（主模式）(Automatic end mode (master mode))

此位由软件置 1 和清零。

0：软件结束模式：当 NBYTES 数据传输完成时，TC 标志将置 1,SCL 的低电平时间将延长直到相应软件操作结束。

1：自动结束模式：当 NBYTES 数据传输完成时，将自动发送停止位。

注：在从模式下，或当 RELOAD 位置 1 时，该位不起作用。

位 24 **RELOAD**：NBYTES 重载模式 (NBYTES reload mode)

此位由软件置 1 和清零。

0：传输 NBYTES 数据（后跟停止位或重复起始位）之后即完成传输。

1：传输 NBYTES 数据之后未完成传输（将重载 NBYTES）。当 NBYTES 数据传输完成时，TCR 标志将置 1,SCL 的低电平时间将延长直到相应软件操作结束。

位 23:16 **NBYTES[7:0]**：字节数 (Number of bytes)

在此设置待发送/接收的字节数。在从模式下，当 SBC=0 时，该字段为无关字段。

注： START 位置 1 时，不允许更改这些位。

位 15 **NACK**：NACK 生成（从模式）(NACK generation (slave mode))

此位由软件置 1，并可在发送 NACK 时、接收到停止条件或匹配地址时、或者 PE=0 时由硬件清零。

0：在当前接收的字节后发送 ACK。

1：在当前接收的字节后发送 NACK。

注： 向该位写入“0”不起作用。

该位仅在从模式下使用：在主接收器模式下，无论 NACK 位的值为何，最后一个字节（后跟停止位或重复起始位）后都将自动生成 NACK。

当从接收器 NOSTRETCH 模式下发生上溢时，无论 NACK 位的值为何，都将自动生成 NACK。

使能硬件 PEC 校验时 (PECBYTE=1)，PEC 应答值与 NACK 值无关。

位 14 **STOP**：停止位生成（主模式）(Stop generation (master mode))

该位由软件置 1，并可在检测到停止位时或 PE = 0 时由硬件清零。

在主模式下：

0：不生成停止位。

1：在当前字节传输完成后生成停止位。

注： 向该位写入“0”不起作用。

位 13 **START**：起始位生成 (Start generation)

该位由软件置 1，并可在发送起始位（后跟地址序列）之后、发生仲裁丢失时、出现超时错误时、或者 PE = 0 时由硬件清零。它也可由软件清零，方法是向 I2C_ICR 寄存器中的 ADDRCF 位写入“1”。

0：不生成起始位。

1：生成重复起始/起始位:

– 如果 I2C 已处于主模式下且 AUTOEND = 0，则将该位置 1 会在 NBYTES 传输结束后且 RELOAD=0 的情况下生成重复起始位。

– 否则，将该位置 1 会在总线释放后立即生成起始位。

注： 向该位写入“0”不起作用。

即使总线繁忙或 I2C 处于从模式，也可将 START 位置 1。

当 RELOAD 置 1 时，该位不起作用。

位 12 **HEAD10R**：读方向传输时，只发送 10 位地址的前 7 位地址头字节（主接收器模式）(10-bit address header only read direction (master receiver mode))

0：主器件发送完整的 10 位从地址读序列：起始位 + 带写方向的 2 字节 10 位地址 + 重复起始位 + 带读方向的 10 位地址的前 7 位。

1：主器件只发送 10 位地址的前 7 位,后跟读方向。

注： START 位置 1 时，不允许更改此位。

位 11 **ADD10**：10 位寻址模式（主模式）(10-bit addressing mode (master mode))

0：主器件工作在 7 位寻址模式下

1：主器件工作在 10 位寻址模式下

注： START 位置 1 时，不允许更改此位。

位 10 **RD_WRN**：传输方向（主模式）(Transfer direction (master mode))

0：主器件请求写传输。

1：主器件请求读传输。

注： START 位置 1 时，不允许更改此位。

位 9:8 **SADD[9:8]**：从设备地址的第 8 到第 9 位（主模式）(Slave address bit 9:8 (master mode))

在 7 位寻址模式 (ADD10 = 0) 下：

这些位无意义

在 10 位寻址模式 (ADD10 = 1) 下：

这些位应写入待发送从地址的第 8 和第 9 位

注： START 位置 1 时，不允许更改这些位。

位 7:1 **SADD[7:1]**：从地址位 7:1（主模式）(Slave address bit 7:1 (master mode))

在 7 位寻址模式 (ADD10 = 0) 下：

这些位应写入待发送的 7 位从地址

在 10 位寻址模式 (ADD10 = 1) 下：

这些位应写入待发送从地址的第 1 到第 7 位。

注： START 位置 1 时，不允许更改这些位。

位 0 **SADD0**：从地址位 0（主模式）(Slave address bit 0 (master mode))

在 7 位寻址模式 (ADD10 = 0) 下：

该位无意义

在 10 位寻址模式 (ADD10 = 1) 下：

该位应写入待发送从地址的第 0 位

注： START 位置 1 时，不允许更改这些位。

28.7.3 设备自身地址 1 寄存器 (I2C_OAR1)

Own address 1 register

偏移地址：0x08

复位值：0x0000 0000

访问：无等待周期，正在对该寄存器执行写访问时出现另一个写访问的情况除外。在这种情况下，会在第二个写访问中插入等待周期，直到前一个写访问完成为止。第二个写访问的延时最长可达 2 x PCLK1 + 6 x I2CCLK。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw

位 31:16 保留，必须保持复位值。

位 15 **OA1EN**：设备自身地址 1 使能 (Own Address 1 enable)

0：禁止设备自身地址 1。不对接收的从地址 OA1 应答。

1：使能设备自身地址 1。对接收的从地址 OA1 应答。

位 14:11 保留，必须保持复位值。

位 10 **OA1MODE**：设备自身地址 1 10 位模式 (Own Address 1 10-bit mode)

0：设备自身地址 1 为 7 位地址。

1：设备自身地址 1 为 10 位地址。

注： 仅可在 OA1EN=0 时写入该位。

位 9:8 **OA1[9:8]**：接口地址 (Interface address)

7 位寻址模式：无关

10 位寻址模式：地址位 9:8

注： 仅可在 OA1EN=0 时写入这些位。

位 7:1 **OA1[7:1]**：接口地址 (Interface address)

地址位 7:1

注： 仅可在 OA1EN=0 时写入这些位。

位 0 **OA1[0]**：接口地址 (Interface address)

7 位寻址模式：无关

10 位寻址模式：地址位 0

注： 仅可在 OA1EN=0 时写入该位。

28.7.4 设备自身地址 2 寄存器 (I2C_OAR2)

Own address 2 register

偏移地址：0x0C

复位值：0x0000 0000

访问：无等待周期，正在对该寄存器执行写访问时出现另一个写访问的情况除外。在这种情况下，会在第二个写访问中插入等待周期，直到前一个写访问完成为止。第二个写访问的延时最长可达 2 x PCLK1 + 6 x I2CCLK。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw			rw							

位 31:16 保留，必须保持复位值。

位 15 **OA2EN**：设备自身地址 2 使能 (Own Address 2 enable)

0：禁止设备自身地址 2。不对接收的从地址 OA2 应答。
1：使能设备自身地址 2。对接收的从地址 OA2 应答。

位 14:11 保留，必须保持复位值。

位 10:8 **OA2MSK[2:0]**：设备自身地址 2 屏蔽位 (Own Address 2 masks)

000：无屏蔽
001：OA2[1] 被屏蔽，为无关位。仅比较 OA2[7:2]。
010：OA2[2:1] 被屏蔽，为无关位。仅比较 OA2[7:3]。
011：OA2[3:1] 被屏蔽，为无关位。仅比较 OA2[7:4]。
100：OA2[4:1] 被屏蔽，为无关位。仅比较 OA2[7:5]。
101：OA2[5:1] 被屏蔽，为无关位。仅比较 OA2[7:6]。
110：OA2[6:1] 被屏蔽，为无关位。仅比较 OA2[7]。
111：OA2[7:1] 被屏蔽，为无关位。不进行比较，对接收到的全部 7 位地址（保留位除外）应答。

注：仅可在 OA2EN=0 时写入这些位。
只要 OA2MSK 不等于 0，即使比较匹配，也不会对保留的 I2C 地址（0b0000xxx 和 0b1111xxx）应答。

位 7:1 **OA2[7:1]**：接口地址 (Interface address)

地址位 7:1

注：仅可在 OA2EN=0 时写入这些位。

位 0 保留，必须保持复位值。

28.7.5 时序寄存器 (I2C_TIMINGR)

Timing register

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw								rw				rw			

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

位 31:28 **PRESC[3:0]**：时序预分频因子 (Timing prescaler)

该字段用于对 I2CCLK 进行预分频，以生成用于数据建立和保持计数器（请参见*第 638 页的 “I2C 时序”*）以及 SCL 高电平和低电平计数器（请参见*第 652 页的 “I2C 主模式初始化”*）的时钟周期 t_{PRESC}。

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

位 27:24 保留，必须保持复位值。

位 23:20 **SCLDEL[3:0]**：数据建立时间 (Data setup time)

该字段用于在 SDA 边沿和 SCL 上升沿之间生成延时 t_{SCLDEL}。在主模式和从模式下，如果 NOSTRETCH = 0，则 SCL 线的低电平时间将在 t_{SCLDEL} 期间延长。

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

注： t_{SCLDEL} 用于生成 $t_{SU:DAT}$ 时序。

位 19:16 **SDADEL[3:0]**：数据保持时间 (Data hold time)

该字段用于在 SCL 下降沿和 SDA 边沿之间生成延时 t_{SDADEL}。在主模式和从模式下，如果 NOSTRETCH = 0，则 SCL 线的低电平时间将在 t_{SDADEL} 期间延长。

t_{SDADEL}= SDADEL x t_{PRESC}

注： SDADEL 用于生成 $t_{HD:DAT}$ 时序。

位 15:8 **SCLH[7:0]**：SCL 高电平周期（主模式）(SCL high period (master mode))

在主模式下，该字段用于生成 SCL 高电平周期。

t_{SCLH} = (SCLH+1) x t_{PRESC}

注： SCLH 还用于生成 $t_{SU:STO}$ 和 $t_{HD:STA}$ 时序。

位 7:0 **SCLL[7:0]**：SCL 低电平周期（主模式）(SCL low period (master mode))

在主模式下，该字段用于生成 SCL 低电平周期。

t_{SCLL} = (SCLL+1) x t_{PRESC}

注： SCLL 还用于生成 t_{BUF} 和 $t_{SU:STA}$ 时序。

注： 该寄存器必须在 I2C 禁止时 (PE = 0) 进行配置。

28.7.6 超时寄存器 (I2C_TIMEOUTR)

Timeout register

偏移地址：0x14

复位值：0x0000 0000

访问：无等待周期，正在对该寄存器执行写访问时出现另一个写访问的情况除外。在这种情况下，会在第二个写访问中插入等待周期，直到前一个写访问完成为止。第二个写访问的延时最长可达 2 x PCLK1 + 6 x I2CCLK。

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

位 31 **TEXTEN**：时钟信号延展超时使能 (Extended clock timeout enable)

0：禁止时钟信号延展超时检测

1：使能时钟信号延展超时检测 当 I2C 接口执行 SCL 延展的累积时间超过 $t_{LOW:EXT}$ 时，将检测到超时错误 (TIMEOUT=1)。

位 30:28 保留，必须保持复位值。

位 27:16 **TIMEOUTB[11:0]**：总线超时 B (Bus timeout B)

该字段用于配置累积时钟延展超时：

在主模式下，将检测主器件的累积时钟低电平延展时间 ($t_{LOW:MEXT}$)

在从模式下，将检测从器件的累积时钟低电平延展时间 ($t_{LOW:SEXT}$)

$t_{LOW:EXT}$= (TIMEOUTB+1) x 2048 x t_{I2CCLK}

注： 仅可在 TEXTEN=0 时写入这些位。

位 15 **TIMOUTEN**：时钟超时使能 (Clock timeout enable)

0：禁止 SCL 超时检测

1：使能 SCL 超时检测：当 SCL 的低电平时间超过 $t_{TIMEOUT}$ (TIDLE=0)，或 SCL 的高电平时间超过 t_{IDLE} (TIDLE=1) 时，将检测到超时错误 (TIMEOUT=1)。

位 14:13 保留，必须保持复位值。

位 12 **TIDLE**：空闲时钟超时检测 (Idle clock timeout detection)

0：TIMEOUTA 用于检测 SCL 低电平超时

1：TIMEOUTA 用于检测 SCL 和 SDA 高电平超时（总线空闲条件）

注： 仅可在 TIMOUTEN=0 时写入该位。

位 11:0 **TIMEOUTA[11:0]**：总线超时 A (Bus Timeout A)

该字段用于配置：

– SCL 低电平超时条件 $t_{TIMEOUT}$（当 TIDLE=0 时）

$t_{TIMEOUT}$= (TIMEOUTA+1) x 2048 x t_{I2CCLK}

– 总线空闲条件，即 SCL 和 SDA 高电平（当 TIDLE=1 时）

t_{IDLE}= (TIMEOUTA+1) x 4 x t_{I2CCLK}

注： 仅可在 TIMOUTEN=0 时写入这些位。

注： 如果不支持 SMBus 功能，该寄存器保留，并由硬件强制为“0x00000000”。请参见第 28.3 节：I2C 特性实现。

28.7.7 中断和状态寄存器 (I2C_ISR)

Interrupt and status register

偏移地址：0x18

复位值：0x0000 0001

访问：无等待周期

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r							r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

位 31:24 保留，必须保持复位值。

位 23:17 **ADDCODE[6:0]**：地址匹配代码（从模式）(Address match code (Slave mode))
发生地址匹配事件时 (ADDR = 1)，这些位更新为接收到的地址。
在 10 位地址的情况下，ADDCODE 提供 10 位地址的头字节，后跟地址的 2 个 MSB。

位 16 **DIR**：传输方向（从模式）(Transfer direction (Slave mode))
该标志在发生地址匹配事件时 (ADDR=1) 更新。
0：写传输，从器件进入接收器模式。
1：读传输，从器件进入发送器模式。

位 15 **BUSY**：总线繁忙 (Bus busy)
该标志用于指示总线上正在进行通信。当检测到起始位时，该位由硬件置 1。当检测到停止位或 PE = 0 时，该位由硬件清零。

位 14 保留，必须保持复位值。

位 13 **ALERT**：SMBus 报警 (SMBus alert)
当 SMBHEN=1（SMBus 主机配置）、ALERTEN=1 且在 SMBA 引脚上检测到 SMBALERT 事件（下降沿）时，该标志由硬件置 1。该位由软件清零，方法是将 ALERTCF 位置 1。
注：当 PE=0 时，该位由硬件清零。
如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 12 **TIMEOUT**：超时或 t_{LOW} 检测标志 (Timeout or t_{LOW} detection flag)
发生超时或延长时钟超时时，该标志由硬件置 1。该位由软件清零，方法是将 TIMEOUTCF 位置 1。
注：当 PE=0 时，该位由硬件清零。
如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 11 **PECERR**：接收期间的 PEC 错误 (PEC Error in reception)
当接收到的 PEC 与 PEC 寄存器的内容不匹配时，该标志由硬件置 1。接收到错误的 PEC 后，将自动发送 NACK。该标志由软件清零，方法是将 PECCF 位置 1。
注：当 PE=0 时，该位由硬件清零。
如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 10 **OVR**：上溢/下溢（从模式）(Overrun/Underrun (slave mode))

在从模式下且 NOSTRETCH=1 时，如果发生上溢/下溢错误，该标志由硬件置 1。该标志由软件清零，方法是将 OVRCF 位置 1。

注： 当 PE=0 时，该位由硬件清零。

位 9 **ARLO**：仲裁丢失 (Arbitration lost)

发生仲裁丢失时，该标志由硬件置 1。该标志由软件清零，方法是将 ARLOCF 位置 1。

注： 当 PE=0 时，该位由硬件清零。

位 8 **BERR**：总线错误 (Bus error)

当检测到错位的起始位或停止位，而外设也参与传输时，该标志由硬件置 1。在从模式下的地址阶段，该标志不会置 1。该标志由软件清零，方法是将 *BERRCF 位置 1*。

注： 当 PE=0 时，该位由硬件清零。

位 7 **TCR**：传输完成等待重载 (Transfer Complete Reload)

当 RELOAD=1 且 NBYTES 数据传输完成时，该标志由硬件置 1。当 NBYTES 写入一个非零值时，该标志由软件清零。

注： 当 PE=0 时，该位由硬件清零。

该标志单独用于主模式，SBC 位置 1 时单独用于从模式。

位 6 **TC**：传输完成（主模式）(Transfer Complete (master mode))

当 RELOAD=0、AUTOEND=0 且 NBYTES 数据传输完成时，该标志由硬件置 1。当 START 位或 STOP 位置 1 时，该标志由软件清零。

注： 当 PE=0 时，该位由硬件清零。

位 5 **STOPF**：停止位检测标志 (Stop detection flag)

当在总线上检测到停止位，且外设也参与本次传输时，该标志由硬件置 1：

– 外设作为主器件，该位置位的前提是外设已经已经发出停止位。

– 外设作为从器件，该位置位的前提条件是此次传输的寻址对象就是该外设。

该标志由软件清零，方法是将 STOPCF 位置 1。

注： 当 PE=0 时，该位由硬件清零。

位 4 **NACKF**：接收到否定应答标志 (Not Acknowledge received flag)

传输完字节后接收到 NACK 时，该标志由硬件置 1。该标志由软件清零，方法是将 NACKCF 位置 1。

注： 当 PE=0 时，该位由硬件清零。

位 3 **ADDR**：地址匹配（从模式）(Address matched (slave mode))

接收到的地址与使能的从设备地址之一匹配时，该位由硬件置 1。该位由软件清零，方法是将 *ADDRCF 位置 1*。

注： 当 PE=0 时，该位由硬件清零。

位 2 **RXNE**：接收数据寄存器不为空（接收器）(Receive data register not empty (receivers))
当接收到的数据已复制到 I2C_RXDR 寄存器且准备就绪可供读取时，该位由硬件置 1。读取 I2C_RXDR 时，将清零该位。
注： 当 PE=0 时，该位由硬件清零。

位 1 **TXIS**：发送中断状态（发送器）(Transmit interrupt status (transmitters))
当 I2C_TXDR 寄存器为空时，该位由硬件置 1，待发送的数据必须写入 I2C_TXDR 寄存器。下一个待发送的数据写入 I2C_TXDR 寄存器时，该位被清零。
该位只能在 NOSTRETCH=1 时由软件写入“1”，以生成 TXIS 事件（TXIE=1 时为中断，TXDMAEN=1 时为 DMA 请求）。
注： 当 PE=0 时，该位由硬件清零。

位 0 **TXE**：发送数据寄存器为空（发送器）(Transmit data register empty (transmitters))
当 I2C_TXDR 寄存器为空时，该位由硬件置 1。下一个待发送的数据写入 I2C_TXDR 寄存器时，该位被清零。
该位可由软件写入“1”，以刷新发送数据寄存器 I2C_TXDR。
注： 当 PE=0 时，该位由硬件置 1。

28.7.8 中断清零寄存器 (I2C_ICR)

Interrupt clear register

偏移地址：0x1C

复位值：0x0000 0000

访问：无等待周期

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

位 31:14 保留，必须保持复位值。

位 13 **ALERTCF**：报警标志清零 (Alert flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 ALERT 标志将清零。
注： 如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 12 **TIMOUTCF**：超时检测标志清零 (Timeout detection flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 TIMEOUT 标志将清零。
注： 如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 11 **PECCF**：PEC 错误标志清零 (PEC Error flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 PECERR 标志将清零。
注： 如果不支持 SMBus 功能，该位保留，并由硬件强制为“0”。请参见第 28.3 节：I2C 特性实现。

位 10 **OVRCF**：上溢/下溢标志清零 (Overrun/Underrun flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 OVR 标志将清零。

位 9 **ARLOCF**：仲裁丢失标志清零 (Arbitration Lost flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 ARLO 标志将清零。

位 8 **BERRCF**：总线错误标志清零 (Bus error flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 BERRF 标志将清零。

位 7:6 保留，必须保持复位值。

位 5 **STOPCF**：停止位检测标志清零 (Stop detection flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 STOPF 标志将清零。

位 4 **NACKCF**：否定应答标志清零 (Not Acknowledge flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 ACKF 标志将清零。

位 3 **ADDRCF**：地址匹配标志清零 (Address Matched flag clear)
将 1 写入此位时，I2C_ISR 寄存器中的 ADDR 标志将清零。将 1 写入此位时，I2C_CR2 寄存器中的 START 位也将清零。

位 2:0 保留，必须保持复位值。

28.7.9 PEC 寄存器 (I2C_PECR)

PEC register

偏移地址：0x20

复位值：0x0000 0000

访问：无等待周期

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

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

位 31:8 保留，必须保持复位值。

位 7:0 **PEC[7:0]**：数据包错误校验寄存器 (Packet error checking register)
当 PECEN=1 时，此字段包含内部 PEC。
当 PE=0 时，PEC 由硬件清零。

注：如果不支持 SMBus 功能，该寄存器保留，并由硬件强制为“0x00000000”。请参见第 28.3 节：I2C 特性实现。

28.7.10 接收数据寄存器 (I2C_RXDR)

Receive data register

偏移地址：0x24

复位值：0x0000 0000

访问：无等待周期

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

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

位 31:8 保留，必须保持复位值。

位 7:0 **RXDATA[7:0]**：8 位接收数据 (8-bit receive data)
从 I^2C 总线接收的数据字节。

28.7.11 发送数据寄存器 (I2C_TXDR)

Transmit data register

偏移地址：0x28

复位值：0x0000 0000

访问：无等待周期

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

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

位 31:8 保留，必须保持复位值。

位 7:0 **TXDATA[7:0]**：8 位发送数据 (8-bit transmit data)
待发送到 I^2C 总线的数据字节。
注： 仅可在 TXE=1 时写入这些位。

28.7.12 I2C 寄存器映射

下表提供了 I2C 寄存器映射和复位值。

表 122. I2C 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.		TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

表 122. I2C 寄存器映射和复位值（续）

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
0x28	**I2C_TXDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

29 通用同步异步收发器 (USART)

29.1 简介

通用同步异步收发器 (USART) 能够灵活地与外部设备进行全双工数据交换，满足外部设备对工业标准 NRZ 异步串行数据格式的要求。USART 通过可编程波特率发生器提供了多种波特率。

它支持同步单向通信、半双工单线通信以及多处理器通信； 还支持 LIN（局域互连网络）、智能卡协议与 IrDA（红外线数据协会）SIR ENDEC 规范，以及调制解调器操作 (CTS/RTS)。

通过配置多个缓冲区使用 DMA（直接存储器访问）可实现高速数据通信。

29.2 USART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 可配置为 16 倍过采样或 8 倍过采样，因而为速度容差与时钟容差的灵活配置提供了可能
- 32 MHz 时钟频率和 8 倍过采样时，通用可编程收发波特率最高为 4 Mbps
- 双时钟域允许：
  - USART 功能和从停止模式唤醒
  - 方便的波特率编程，独立于 PCLK 重新编程
- 自动波特率检测
- 数据字长度可编程（7 位、8 位或 9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位可配置（支持 1 个或 2 个停止位）
- 用于同步通信的同步模式和时钟输出
- 单线半双工通信
- 使用 DMA 实现连续通信
- 使用中央 DMA 在预留的 SRAM 缓冲区中收/发字节
- 发射器和接收器有单独的使能位
- 发送和接收的单独信号极性控制
- Tx/Rx 引脚配置可交换
- 调制解调器和 RS-485 收发器的硬件流控制
- 通信控制/错误检测标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性
- 十四个具有标志位的中断源
- 多处理器通信
  如果地址不匹配，USART 将进入静默模式。
- 从静默模式唤醒（通过空闲线检测或地址标记检测）

29.3 USART 扩展特性

- LIN 主模式同步断路发送功能和 LIN 从模式断路检测功能
  - 当 USART 被配置为 LIN 使用时，可生成 13 位断路和检测 10/11 位断路
- 正常模式下支持 3/16 位持续时间的 IrDA SIR 编解码器
- 智能卡模式
  - 对于 ISO/IEC 7816-3 标准中定义的智能卡，支持 T=0 和 T=1 异步协议
  - 智能卡工作模式下，支持 0.5 或 1.5 个停止位
- 支持 ModBus 通信
  - 超时功能
  - CR/LF 字符识别

29.4 USART 实现

表 123. STM32L0x3 USART 特性

USART 模式/特性(1)	USART1/2	USART4	USART5	LPUART1
调制解调器的硬件流控	X	X	-	X
使用DMA进行连续通信	X	X	X	X
多处理器通信	X	X	X	X
同步模式	X	X	X	-
智能卡模式	X	-	-	-
单线半双工通信	X	X	X	X
Ir SIR ENDEC 模块	X	-	-	-
LIN 模式	X	-	-	-
双时钟域和从停止模式唤醒	X	-	-	X
接收器超时中断	X	-	-	-
Modbus 通信	X	-	-	-
自动波特率检测	X	-	-	-
驱动器使能	X	X	X	X
USART 数据长度	7 位、8 位和 9 位			

1. X = 支持。

29.5 USART 功能说明

任何 USART 双向通信均需要至少两个引脚：接收数据输入引脚 (RX) 和发送数据输出引脚 (TX)：

- **RX**：接收数据输入引脚。

  串行数据输入引脚。过采样技术可区分有效输入数据和噪声，从而用于恢复数据。
- **TX**：发送数据输出引脚。

  如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线和智能卡模式下，该 I/O 用于发送和接收数据。

正常 USART 模式下，通过这些引脚发送和接收串行数据。这些帧包括：

- 发送或接收前保持空闲线路
- 起始位
- 数据字（字长 7 位、8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 0.5 个、1 个、1.5 个、2 个停止位
- USART 接口使用波特率发生器
- 状态寄存器 (USARTx_ISR)
- 接收和发送数据寄存器（USARTx_RDR、USARTx_TDR）
- 波特率寄存器 (USARTx_BRR)
- 智能卡模式下的保护时间寄存器 (USARTx_GTPR)。

有关各个位的定义，请参见*第 739 页上的第 29.8 节：USART 寄存器*。

在同步模式和智能卡模式下连接时需要以下引脚：

- **CK**：时钟输出。该引脚用于输出发送器数据时钟，以便按照 SPI 主模式进行同步发送（起始位和结束位上无时钟脉冲，可通过软件向最后一个数据位发送时钟脉冲）。RX 上可同步接收并行数据。这一点可用于控制带移位寄存器的外设（例如 LCD 驱动器）。时钟相位和极性可通过软件编程。在智能卡模式下，CK 输出可向智能卡提供时钟。

在 RS232 硬件流控制模式下需要以下引脚：

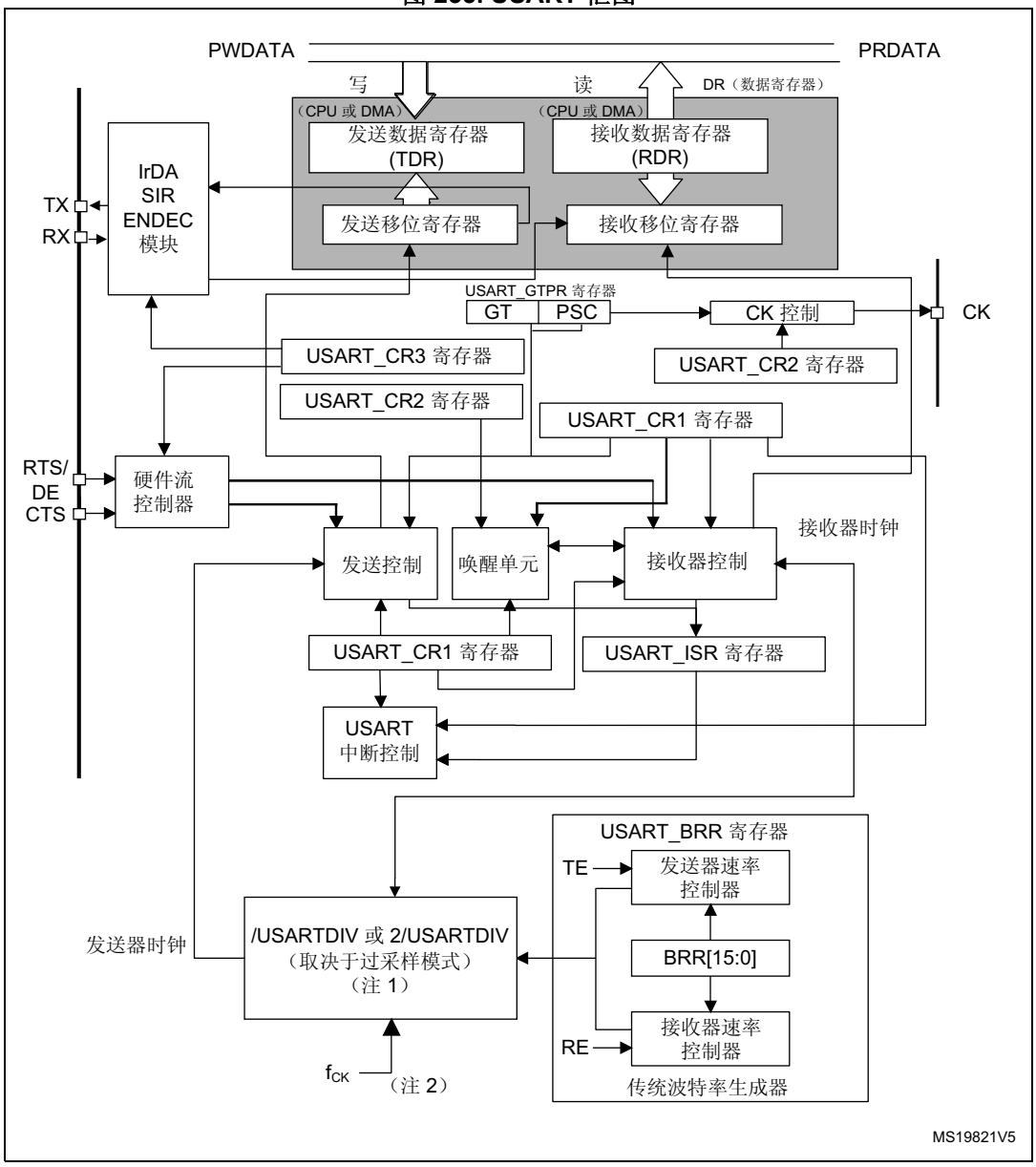
- **CTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **RTS**：“请求以发送”用于指示 USART 已准备好接收数据（低电平时）

在 RS485 硬件控制模式下需要以下引脚：

- **DE**：“驱动器使能”用于激活外部收发器的发送模式。

注： *DE 和 RTS 共用同一个引脚。*

图 235. USART 框图



1. 有关在 USARTx_BRR 寄存器中编码 USARTDIV 的详细信息，请参见*第 29.5.4 节：USART 波特率生成*。
2. f_{CK} 可以是 f_{LSE}、f_{HSI}、f_{PCLK}、f_{SYS}。

29.5.1 USART 字符说明

可通过对 USARTx_CR1 寄存器中的 M[1:0] 位进行编程来选择 7 位、8 位或 9 位的字长（请参见*图 236*）。

- 7 位字符长度：M[1:0] = 10
- 8 位字符长度：M[1:0] = 00
- 9 位字符长度：M[1:0] = 01

注： *7 位数据长度模式下，不支持智能卡模式、LIN 主模式和自动波特率（0x7F 帧和 0x55 帧检测）。只有某些 USART 支持 7 位模式。*

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

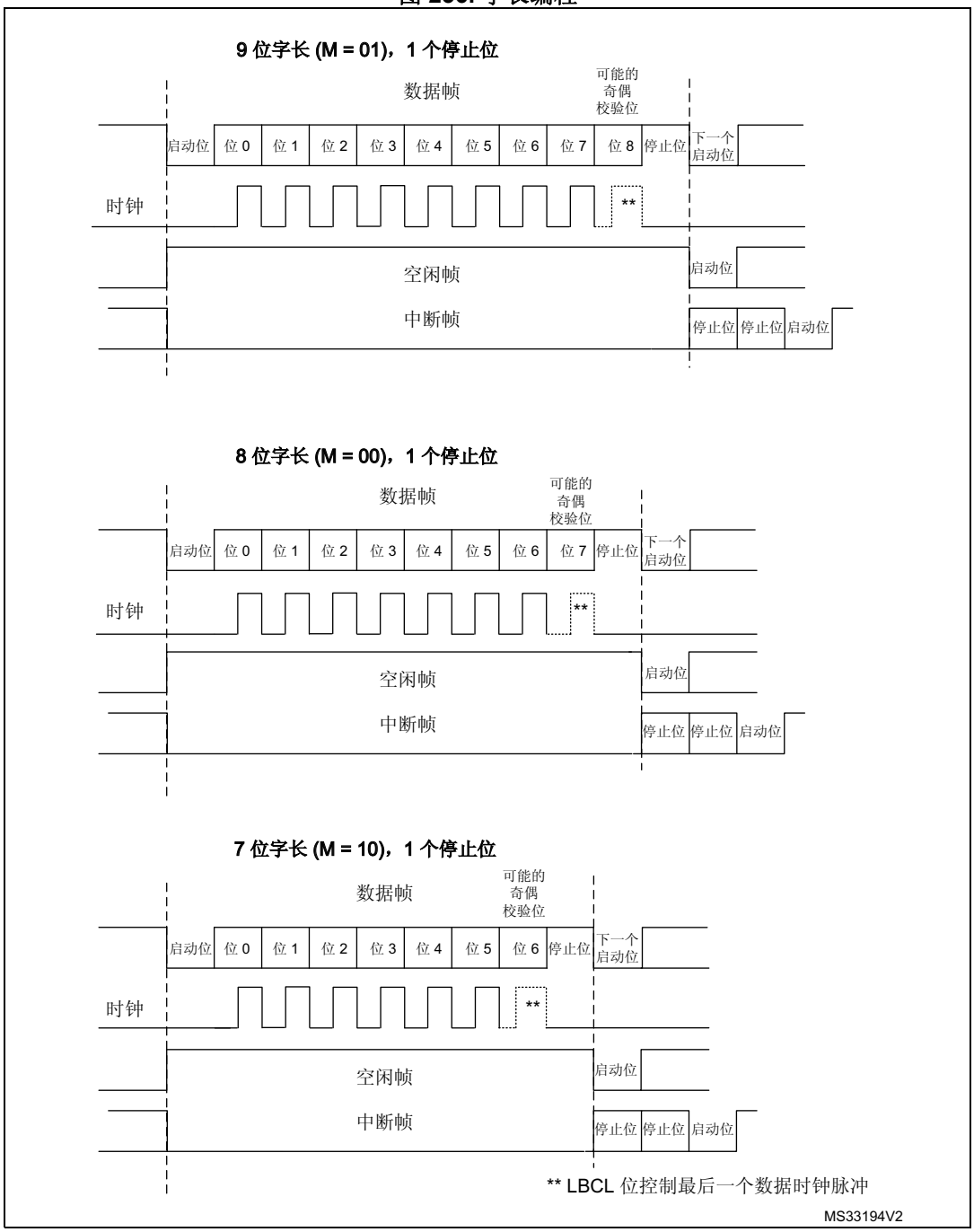
*空闲字符*可理解为整个帧周期内电平均为“1”（停止位的电平也是“1”）。

*停止字符*可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 2 个停止位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接收时钟。

下面给出了各个块的详细信息。

图 236. 字长编程


9 位字长 (M = 01)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
位 7
位 8
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
8 位字长 (M = 00)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
位 7
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
7 位字长 (M = 10)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
** LBCL 位控制最后一个数据时钟脉冲
MS33194V2

29.5.2 USART 发送器

发送器可发送 7 位、8 位或 9 位的数据字，具体取决于 M 位的状态。要激活发送器功能，必须将发送使能位 (TE) 置 1。发送移位寄存器中的数据在 TX 引脚输出，相应的时钟脉冲在 CK 引脚输出。

字符发送

USART 发送期间，首先通过 TX 引脚移出数据的最低有效位（默认配置）。该模式下，USARTx_TDR 寄存器的缓冲区 (TDR) 位于内部总线和发送移位寄存器之间（请参见*图 235*）。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

USART 支持以下停止位：0.5、1、1.5 和 2 个停止位。

注： *向 USARTx_TDR 中写入要发送的数据前，TE 位必须先置 1。*

数据发送期间不应复位 TE 位。发送期间复位 TE 位会冻结波特率计数器，从而将损坏 TX 引脚上的数据。当前传输的数据将会丢失。

使能 TE 位后，将会发送空闲帧。

可配置的停止位

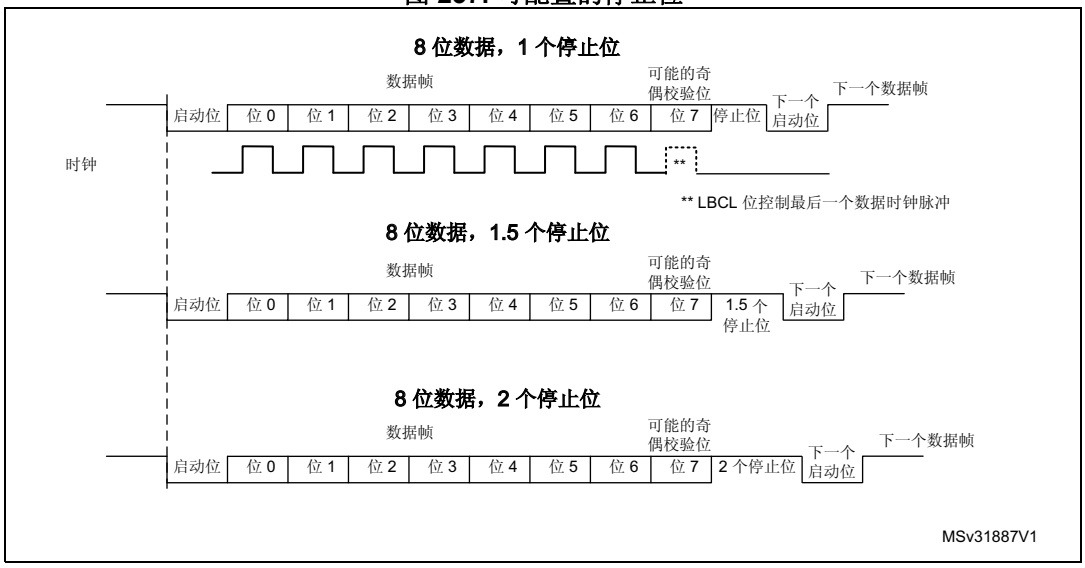
可以在控制寄存器 2 的位 13 和位 12 中编程将随各个字符发送的停止位的数量。

- ***1 个停止位***：这是停止位数量的默认值。
- ***2 个停止位***：正常 USART 模式、单线模式和调制解调器模式支持该值。
- ***1.5 个停止位***：用于智能卡模式。
- ***0.5 个停止位***：在智能卡模式下接收数据时使用。

空闲帧发送将包括停止位。

中断发送是 10 个低电平位（M[1:0] = 00 时）、11 个低电平位（M[1:0] = 01 时）或 9 个低电平位（M[1:0] = 10 时），然后是 2 个停止位（请参见*图 237*）。无法传送长中断（中断长度大于 9/10/11 个低电平位）。

图 237. 可配置的停止位



字符发送步骤

1. 对 USARTx_CR1 中的 M 位进行编程以定义字长。
2. 使用 USARTx_BRR 寄存器选择所需波特率。
3. 对 USARTx_CR2 中的停止位数量进行编程。
4. 通过向 USARTx_CR1 寄存器中的 UE 位写入 1 使能 USART。
5. 如果将进行多缓冲区通信，请选择 USARTx_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 USARTx_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。
7. 在 USARTx_TDR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 USARTx_TDR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 USART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。

有关代码示例，请参见 *A.17.1：USART 发送器配置代码示例*。

单字节通信

始终通过向发送数据寄存器写入数据将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 USARTx_TDR 寄存器移到移位寄存器中且数据发送已开始。
- USARTx_TDR 寄存器为空。
- USARTx_TDR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

有关代码示例，请参见 *A.17.2：USART 发送字节代码示例*。

TXEIE 位置 1 时该标志位会生成中断。

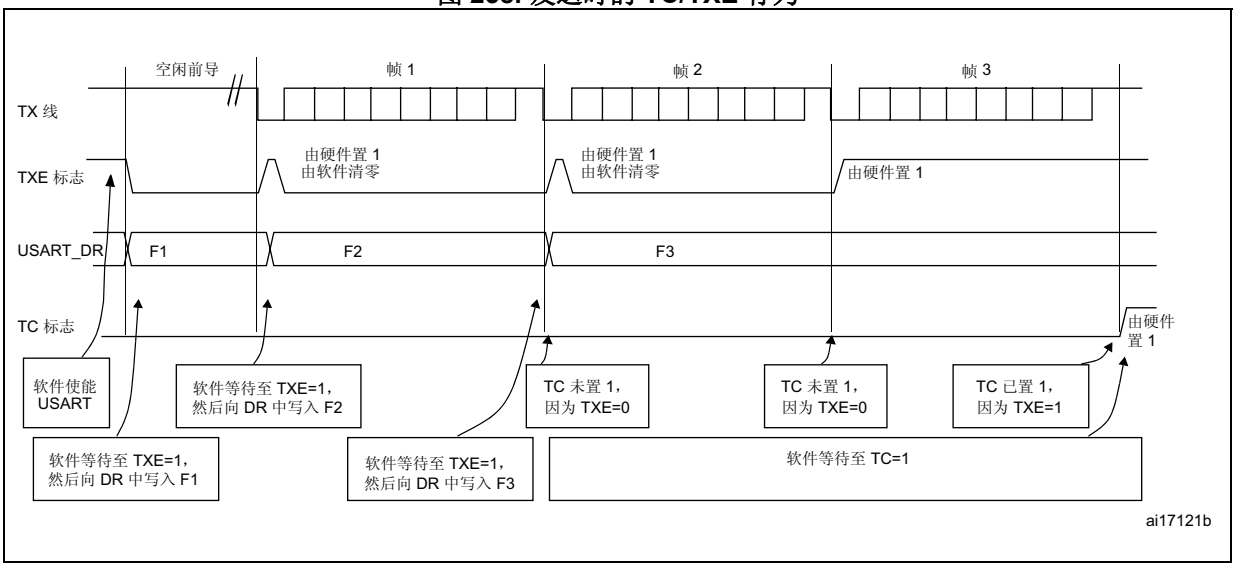
发送时，要传入 USARTx_TDR 寄存器的写指令中存有 TDR 寄存器中的数据，接下来，该数据将在当前进行的发送结束时复制到移位寄存器中。

未发送时，要传入 USARTx_TDR 寄存器的写指令将数据置于移位寄存器中，数据发送开始时，TXE 位置 1。

如果帧已发送（停止位后）且 TXE 位置 1，TC 位将变为高电平。如果 USARTx_CR1 寄存器中的 TCIE 位置 1，将生成中断。

向 USARTx_TDR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 USART 或使微控制器进入低功率模式（请参见*图 238：发送时的 TC/TXE 行为*）。

图 238. 发送时的 TC/TXE 行为



有关代码示例，请参见 *A.17.3：USART 传输完成代码示例*。

中断字符

将 SBKRQ 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位（请参见*图 236*）。

如果将“1”写入 SBKRQ 位，则当前字符发送完成后，将在 TX 线路上发送一个中断字符。通过写操作将 SBKF 位置 1 并在中断字符发送完成时（发送中断字符后的停止位期间），此位由硬件复位。USART 在中断帧末尾的两位持续时间内插入一个逻辑“1”信号 (STOP)，以确保识别下个帧的起始位。

这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

空闲字符

将 TE 位置 1 会驱动 USART 在第一个数据帧之前发送一个空闲帧。

29.5.3 USART 接收器

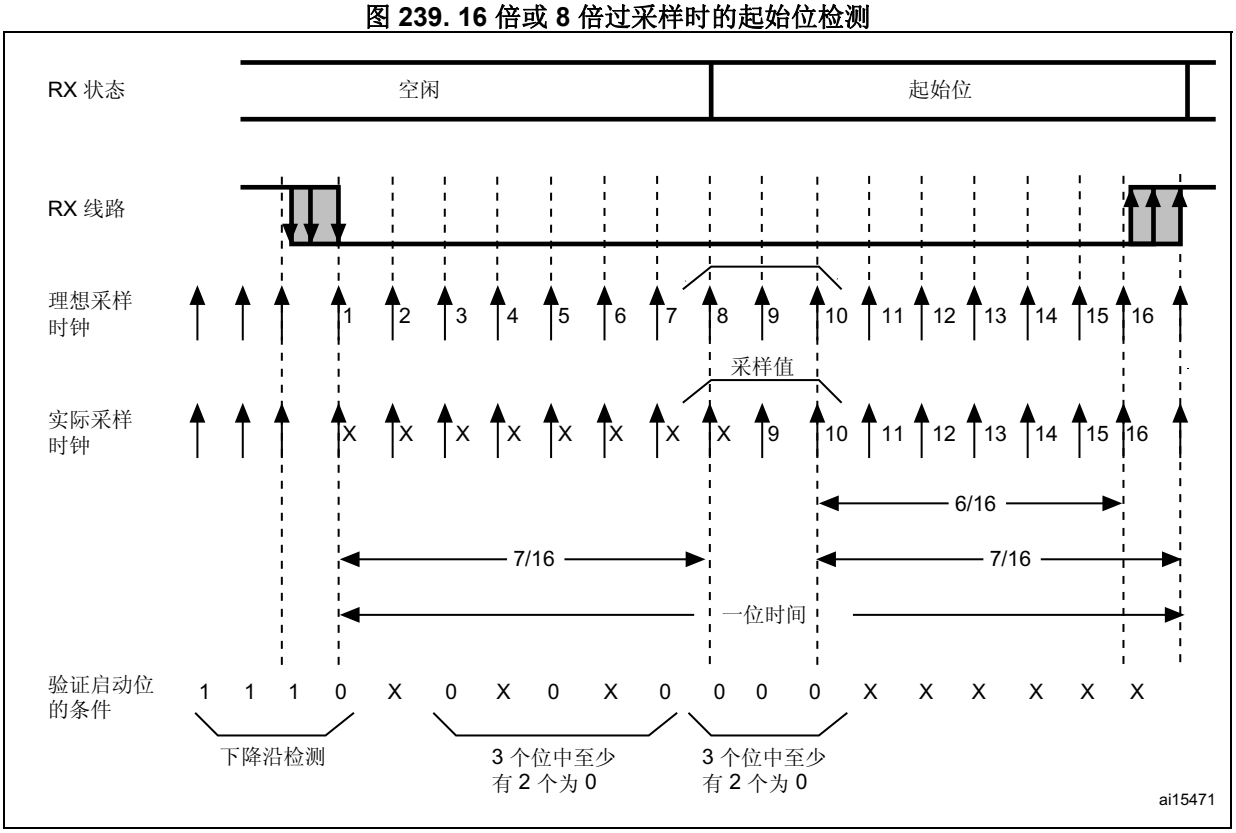
USARTx 可接收 7 位、8 位或 9 位的数据字，具体取决于 USART_CR1 寄存器中的 M 位。

起始位检测

16 倍或 8 倍过采样时，起始位检测序列相同。

在 USART 中，识别出特定序列的采样时会检测起始位。此序列为：1 1 1 0 X 0 X 0X 0X 0 X 0X 0。

图 239. 16 倍或 8 倍过采样时的起始位检测



注： *如果序列不完整，起始位检测将中止，接收器将返回空闲状态（无标志位置 1）等待下降沿。*

如果 3 个采样位均为 0（针对第 3 位、第 5 位和第 7 位进行首次采样时检测到这 3 位均为 0；针对第 8 位、第 9 位和第 10 位进行第二次采样时检测到这 3 位均为 0），可确认起始位（RXNE 标志位置 1，RXNEIE=1 时生成中断）。

满足以下条件时，可验证起始位（RXNE 标志位置 1，如果 RXNEIE=1 则生成中断）但 NF 噪声标志位置 1：

a) 对于两次采样，3 个采样位中有 2 位为 0（针对第 3 位、第 5 位和第 7 位进行采样；针对第 8 位、第 9 位和第 10 位采样）

或

b) 如果其中一次采样时（对第 3 位、第 5 位和第 7 位进行采样或对第 8 位、第 9 位和第 10 位进行采样），3 个采样位中有 2 个为 0。

如果条件 a 或条件 b 都不满足，则启动检测中止，接收器返回空闲状态（无标志位置 1）。

字符接收

USART 接收期间，首先通过 RX 引脚移入数据的最低有效位（默认配置）。该模式下，USARTx_RDR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

字符接收步骤

1. 对 USARTx_CR1 中的 M 位进行编程以定义字长。
2. 使用波特率寄存器 USARTx_BRR 选择所需波特率
3. 对 USARTx_CR2 中的停止位数量进行编程。
4. 通过向 USARTx_CR1 寄存器中的 UE 位写入 1 使能 USART。
5. 如果将进行多缓冲区通信，请选择 USARTx_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 RE 位 USARTx_CR1 置 1。这一操作将使能接收器开始搜索起始位。

有关代码示例，请参见 *A.17.4：USART 接收器配置代码示例*。

接收到字符时：

- RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数据（及其相应的错误标志）。
- 如果 RXNEIE 位置 1，则会生成中断。
- 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。也可使用 RXNE 位将 PE 标志置 1。
- 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对接收数据寄存器执行读操作清零。
- 在单缓冲区模式下，通过软件对 USARTx_RDR 寄存器执行读操作将 RXNE 位清零。也可以通过将 USARTx_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

有关代码示例，请参见 *A.17.5：USART 接收字节代码示例*。

中断字符

接收到中断字符时，USART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位置 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。RXNE 位清零前，数据无法从移位寄存器传送到 RDR 寄存器。

每接收到一个字节后，RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 USARTx_RDR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 位置 1，则会生成中断。
- 通过设置 ICR 寄存器中的 ORECF 位复位 ORE 位。

注： *ORE 位置 1 时表示至少 1 个数据丢失。存在两种可能：*

- 如果 RXNE=1，则最后一个有效数据存储于接收寄存器 RDR 中并且可进行读取；

- 如果 RXNE=0，则表示最后一个有效数据已被读取，因此 RDR 中没有要读取的数据。接收到新（和丢失）数据的同时已读取 RDR 中的最后一个有效数据时，会发生该情况。

选择合适的过采样方法

当支持双时钟域和从停止模式唤醒时，时钟源可以是以下源之一：PCLK（默认值）、LSE、HSI16 或 SYSCLK。否则，USART 时钟源是 PCLK。

选择 LSE 或 HSI16 作为时钟源可允许 USART 在 MCU 处于低功耗模式时接收数据。需要时，USART 可基于所接收的数据和唤醒模式选择来唤醒 MCU，以通过用软件读取 USARTx_RDR 寄存器的方式或通过 DMA 的方式传输已接收数据。

对于其它时钟源，系统必须激活才能进行 USART 通信。

接收器采用不同的用户可配置过采样技术，可以从噪声中提取有效数据。这可在最大通信速度与抗噪声/时钟误差性能之间实现平衡。

可通过编程 USARTx_CR1 寄存器中的 OVER8 位来选择采样方法，且采样时钟可以是波特率时钟的 16 倍或 8 倍（请参见*图 240* 和*图 241*）。

根据应用：

- 选择 8 倍过采样 (OVER8=1) 以获得更高的速度（高达 $f_{CK}/8$）。这种情况下接收器对时钟偏差的最大容差将会降低（请参见*第 714 页上的第 29.5.5 节：USART 接收器对时钟偏差的容差*）
- 选择 16 倍过采样 (OVER8=0) 以增加接收器对时钟偏差的容差。这种情况下，最大速度限制为最高 $f_{CK}/16$，其中 f_{CK} 为时钟源频率。

可通过编程 USARTx_CR3 寄存器中的 ONEBIT 位选择用于评估逻辑电平的方法。有两种选择：

- 在已接收位的中心进行三次采样，从而进行多数表决。这种情况下，如果用于多数表决的 3 次采样结果不相等，NF 位置 1。
- 在已接收位的中心进行单次采样

  根据应用：

  – 在噪声环境下工作时，请选择三次采样的多数表决法 (ONEBIT=0)；在检测到噪声时请拒绝数据（请参见*图 124*），因为这表示采样过程中产生了干扰。

  – 线路无噪声时请选择单次采样法 (ONEBIT=1) 以增加接收器对时钟偏差的容差（请参见*第 714 页上的第 29.5.5 节：USART 接收器对时钟偏差的容差*）。这种情况下 NF 位始终不会置 1。

帧中检测到噪声时：

- 在 RXNE 位的上升沿时 NF 位置 1。
- 无效数据从移位寄存器传送到 USARTx_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，USARTx_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过设置 ICR 寄存器中的 NFCF 位复位 NF 位。

注： *LIN、智能卡和 IrDA 模式下不可采用 8 倍过采样。在这些模式下，OVER8 位由硬件强制清零。*

图 240. 16 倍过采样时的数据采样

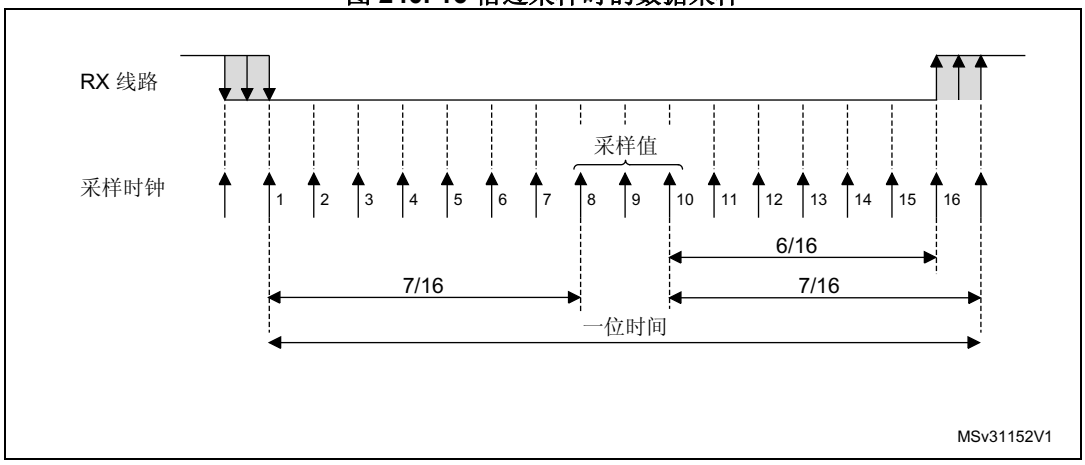


图 241. 8 倍过采样时的数据采样

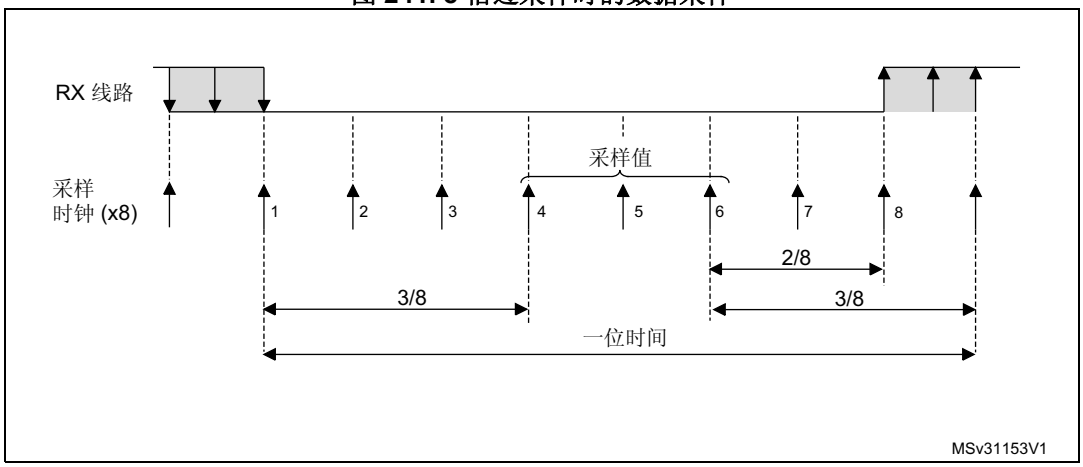


表 124. 通过采样数据进行噪声检测

采样值	NE 状态	接收的位值
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

帧错误 (Framing error)

如果接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，则会检测到帧错误。

检测到帧错误时：

- FE 位由硬件置 1。
- 无效数据从移位寄存器传送到 USARTx_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，USARTx_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过将 1 写入 USARTx_ICR 寄存器中的 FECF 位来复位 FE 位。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 或 2 个（正常模式下），也可以是 0.5 或 1.5 个（智能卡模式下）。

- ***0.5 个停止位（在智能卡模式下接收时）：****不会对 0.5 个停止位进行采样。结果，选择 0.5 个停止位时，无法检测到帧错误和中断帧。*
- ***1 个停止位***：将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
- ***1.5 个停止位（智能卡模式）***：在智能卡模式下发送时，设备必须检查数据是否正确发送。因此必须使能接收器块（USARTx_CR1 寄存器中的 RE =1）并检查停止位，以测试智能卡是否已检测到奇偶校验错误。发生奇偶校验错误时，智能卡会在采样时将数据信号强制为低电平，即 NACK 信号-，该信号被标记为帧错误。之后，FE 标志在 1.5 个停止位的末尾由 RXNE 置 1。在第 16、第 17 和第 18 次采样时对 1.5 个停止位进行采样（停止位采样开始后维持 1 个波特时钟周期）。1.5 个停止位可分为 2 个部分：0.5 个波特时钟周期（未发生任何动作），然后是 1 个正常的停止位周期（一半时间处进行采样）。更多详细信息，请参见*第 725 页上的第 29.5.13 节：USART 智能卡模式*。
- ***2 个停止位***：采样 2 个停止位时在第 8、第 9 和第 10 次采样时对第一个停止位进行采样。如果在第一个停止位期间检测到帧错误，则帧错误标志位将会置 1。发生帧错误时不检测第 2 个停止位。RXNE 标志将在第一个停止位末尾时置 1。

29.5.4 USART 波特率生成

接收器和发送器（Rx 和 Tx）的波特率设置为 USARTx_BRR 寄存器中编程的相同值。

公式 1：适用于标准 USART（包括 SPI 模式）的波特率（OVER8 = 0 或 1）

16 倍过采样时的公式为：

$$\text{Tx/Rx 波特} = \frac{f_{CK}}{\text{USARTDIV}}$$

8 倍过采样时的公式为：

$$\text{Tx/Rx 波特} = \frac{2 \times f_{CK}}{\text{USARTDIV}}$$

公式 2：智能卡、LIN 和 IrDA 模式 (OVER8 = 0) 下的波特率

智能卡、LIN 和 IrDA 模式下只支持 16 倍过采样：

$$\text{Tx/Rx 波特} = \frac{f_{CK}}{\text{USARTDIV}}$$

USARTDIV 是一个存放在 USARTx_BRR 寄存器中的无符号定点数。

- 当 OVER8 = 0 时，BRR = USARTDIV。
- 当 OVER8 = 1 时
  - BRR[2:0] = USARTDIV[3:0]，右移 1 位。
  - BRR[3] 必须保持清零。
  - BRR[15:4] = USARTDIV[15:4]

注：对 USARTx_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。

16 倍或 8 倍过采样时，USARTDIV 必须大于或等于 0d16。

如何从 USARTx_BRR 寄存器中获取 USARTDIV

示例 1

要通过 f_{CK} = 8 MHz 获得 9600 波特率。

- 16 倍过采样时：
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 833d = 0341h
- 8 倍过采样时：
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (1667d = 683h)
  BRR[3:0] = 3h << 1 = 1h
  BRR = 0x681

示例 2

要通过 f_{CK} = 32 MHz 获得 921.6 K 波特率。

- 16 倍过采样时：
  USARTDIV = 32 000 000/921 600
  BRR = USARTDIV = 35d = 23h
- 8 倍过采样时：
  USARTDIV = 2 * 32 000 000/921 600
  USARTDIV = 70d = 46h
  BRR[3:0] = USARTDIV[3:0] >> 1 = 6h >> 1 = 3h
  BRR = 0x43

表 125. 采用 16 倍或 8 倍过采样时，在 f_{CK} = 32 MHz 下编程波特率的误差计算(1)

波特率		16 倍过采样 (OVER8 = 0)			8 倍过采样 (OVER8 = 1)		
序号	所需值	实际值	BRR	误差 % =（计算值 - 所需值）/ 所需波特率	实际值	BRR	误差 %
1	2.4 KBps	2.4 KBps	0x3415	0	2.4 KBps	0x6825	0
2	9.6 KBps	9.6 KBps	0xD05	0	9.6 KBps	0x1A05	0
3	19.2 KBps	19.19 KBps	0x683	0.02	19.2 KBps	0xD02	0
4	38.4 KBps	38.41 KBps	0x341	0.04	38.39 KBps	0x681	0.02
5	57.6 KBps	57.55 KBps	0x22C	0.08	57.6 KBps	0x453	0
6	115.2 KBps	115.1 KBps	0x116	0.08	115.11 KBps	0x226	0.08
7	230.4 KBps	230.21 KBps	0x8B	0.08	230.21 KBps	0x113	0.08
8	460.8 KBps	463.76 KBps	0x045	0.64	460.06 KBps	0x85	0.08
9	921.6 KBps	914.28 KBps	0x23	0.79	927.5 KBps	0x42	0.79
10	2 MBps	2 MBps	0x10	0	2 MBps	0x20	0
12	4MBps	4MBps	NA	NA	4MBps	0x10	0

1. CPU 时钟越低，特定波特率的精度就越低。可达到的波特率的上限可使用这些数据进行确定。

29.5.5 USART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 USART 接收器的容差时，USART 异步接收器才能正常工作。影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

DTRA + DQUANT + DREC + DTCL + DWU < USART 接收器的容差

其中

DWU 是使用从停止模式唤醒时因采样点偏差产生的误差。

M[1:0] = 01 时：

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

M[1:0] = 00 时：

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

M[1:0] = 10 时：

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ 是检测到唤醒事件到时钟（由外设请求）与调压器均就绪的时间。

USART 接收器在*表 126* 和*表 126* 中指定的最大容许偏差下可正确接收数据，具体取决于以下选择：

- 由 USARTx_CR1 寄存器中的 M 位定义的 9 位、10 位或 11 位字符长度
- 由 USARTx_CR1 寄存器中的 OVER8 位定义的 8 倍或 16 倍过采样
- USARTx_BRR 寄存器的 BRR[3:0] 位等于或不等于 0000
- 使用 1 位或 3 位对数据进行采样，取决于 USARTx_CR3 寄存器中 ONEBIT 位的值

表 126. BRR [3:0] = 0000 时的 USART 接收器容差

M 位	OVER8 位 = 0		OVER8 位 = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

表 127. BRR[3:0] 不等于 0000 时的 USART 接收器容差

M 位	OVER8 位 = 0		OVER8 位 = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

注：当接收的帧恰好包含 10 个（M 位 = 00）、11 个（M 位 = 01）或 9 个（M 位 = 10）位持续时间的空闲帧时，表 126 和*表 127* 中指定的数据可能与特例中的数据略微不同。

29.5.6 USART 自动波特率检测

USART 能够根据接收一个字符检测并自动设置 USARTx_BRR 寄存器的值。自动波特率检测在以下两种情况下非常有用：

- 事先不知道系统的通信速度
- 系统正在使用精确度相对较低的时钟源且该机制允许在不测量时钟偏差的情况下获得正确的波特率。

时钟源频率必须与预期通信速度匹配（16 倍过采样时，波特率介于 f_{CK}/65535 与 f_{CK}/16 之间； 8 倍过采样时，波特率介于 f_{CK}/65535 与 f_{CK}/8 之间）。

激活自动波特率检测前，必须先选择自动波特率检测模式。根据不同的字符模式，存在各种检测模式。

这些模式可通过 USARTx_CR2 寄存器中的 ABRMOD[1:0] 字段选择。在这些自动波特率模式下，波特率在同步接收数据期间被多次测量，每次测量的结果都与前一次进行比较。

这些模式包括：

- **模式 0**：以 1 位开头的任意字符。这种情况下，USART 会测量起始位的持续时间（下降沿到上升沿）。
- **模式 1**：以 10xx 位模式开头的任意字符。这种情况下，USART 会测量起始位和第一个数据位的持续时间。测量在下降沿到下降沿期间完成，可在信号斜率较小时确保较高的精度。
- **模式 2**：0x7F 字符帧（可以是 LSB 在前模式下的 0x7F 字符，也可以是 MSB 在前模式下的 0xFE 字符）。这种情况下，先在起始位结束时更新波特率 (BR)，然后在位 6 结束时更新波特率（根据从下降沿到下降沿执行的测量：BR6）。以 BR 对位 0 到位 6 进行采样，而以 BR6 对字符的其它位进行采样。
- **模式 3**：0x55 字符帧。这种情况下，先在起始位结束时更新波特率 (BR)，然后在位 0 结束时更新波特率（根据从下降沿到下降沿执行的测量：BR0），最后在位 6 结束时更新波特率 (BR6)。以 BR 对位 0 进行采样，以 BR0 对位 1 到位 6 进行采样，以 BR6 对字符的其它位进行采样。

  同时，对 RX 线路的每个中间转换执行其它检查。如果 RX 上的转换与接收器（基于根据位 0 计算的波特率的接收器）未充分同步，则生成错误。

激活自动波特率检测前，必须先通过向 USARTx_BRR 寄存器写入非零的波特率值来初始化该寄存器。

通过将 USARTx_CR2 寄存器中的 ABREN 位置 1 来激活自动波特率检测。之后 USART 将等待 RX 线路上的第一个字符。通过将 USARTx_ISR 寄存器中的 ABRF 标志置 1 来指示自动波特率操作完成。如果线路繁忙，则无法保证正确的波特率检测。这种情况下，BRR 值可能会损坏，ABRE 错误标志位将置 1。如果通信速度不在自动波特率检测范围（位持续时间不在 16 个和 65536 个时钟周期（16 倍过采样时）之间，也不在 8 个和 65536 个时钟周期（8 倍过采样时）之间）内，也会出现这种情况。

RXNE 中断将指示操作结束。

稍后，可随时通过复位 ABRF 标志（通过写入 0）重新启动自动波特率检测。

注： *如果在自动波特率操作期间禁止 USART (UE=0)，则可能损坏 BRR 值。*

29.5.7 使用 USART 进行多处理器通信

在多处理器通信中，以下位要保持清零：

- USART_CR2 寄存器中的 LINEN 位，
- USART_CR3 寄存器中的 HDSEL、IREN 和 SCEN 位。

可以通过 USART 进行多处理器通信（多个 USART 连接在一个网络中）。例如，其中一个 USART 可以是主 USART，其 TX 输出与其它 USART 的 RX 输入相连接。其它 USART 为从 USART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 USART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 USART 服务开销。

可通过静默功能将未被寻址的器件置于静默模式下。为了使用静默模式功能，必须将 USARTx_CR1 寄存器中的 MME 位置 1。

在静默模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- USARTx_ISR 寄存器中的 RWU 位置 1。在某些情况下，RWU 可以由硬件或软件通过 USARTx_RQR 寄存器中的 MMRQ 位自动控制。

根据 USARTx_CR1 寄存器中 WAKE 位的设置，USART 可使用以下两种方法进入或退出静默模式：

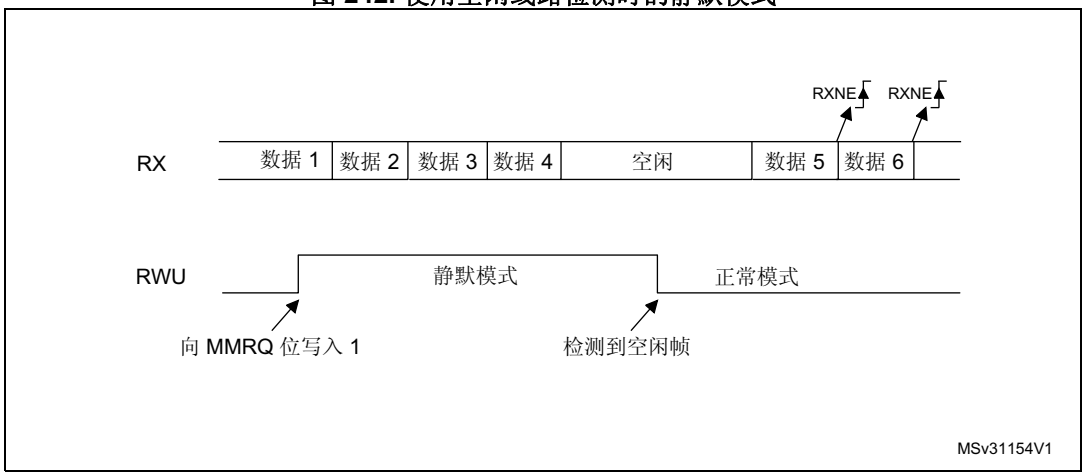
- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

空闲线路检测 (WAKE=0)

当向 MMRQ 位写入 1 且 RWU 位自动置 1 时，USART 进入静默模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 USARTx_ISR 寄存器中的 IDLE 位不会置 1。*图 242* 中给出了使用空闲线路检测时静默模式行为的示例。

图 242. 使用空闲线路检测时的静默模式



注： *如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。*

如果在线路处于空闲状态时激活 USART，在一个 IDLE 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

4 位/7 位地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个或 7 个 LSB 中。7 位或 4 位地址检测通过 ADDM7 位来选择。接收器会将此 4 位/7 位字与其地址进行比较，该接收器的地址在 USARTx_CR2 寄存器的 ADD 位中进行设置。

注：在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。

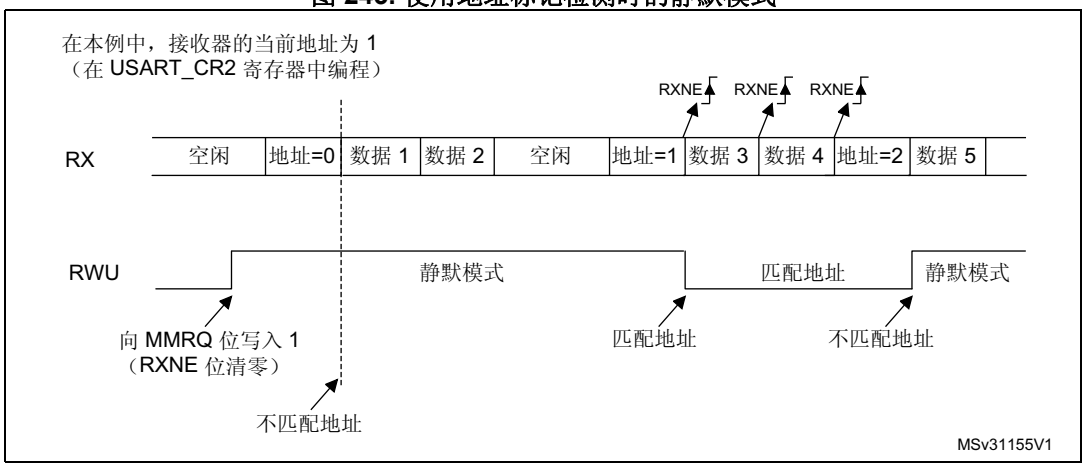
当接收到与其编程地址不匹配的地址字符时，USART 会进入静默模式。此时，RWU 位将由硬件置 1。USART 进入静默模式后，RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当向 MMRQ 位写入 1 时，USART 也会进入静默模式。这种情况下，RWU 位也自动置 1。

当接收到与编程地址匹配的地址字符时，USART 会退出静默模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零，RXNE 位会针对地址字符置 1。

图 243 中给出了使用地址标记检测时静默模式行为的示例。

图 243. 使用地址标记检测时的静默模式



29.5.8 使用 USART 进行 Modbus 通信

USART 为 Modbus/RTU 和 Modbus/ASCII 协议的实现提供基本支持。Modbus/RTU 是一个半双工块传输协议。该协议的控制部分（地址识别、块完整性控制和命令解析）必须用软件实现。

USART 为块结束检测提供基本支持，无需软件开销或其它资源。

Modbus/RTU

在此模式下，一个块的结束通过超过 2 个字符时间的“静默”（空闲线路）来识别。此功能通过可编程的超时功能实现。

超时功能和中断必须分别通过 USARTx_CR2 寄存器中的 RTOEN 位和 USARTx_CR1 寄存器中的 RTOIE 位激活。与 2 个字符时间（例如 22 x 位持续时间）的超时相对应的值必须在 RTO 寄存器中编程。如果在此期间接收线路空闲，则在接收到最后一个停止位后，将生成中断，同时通知软件当前块接收已完成。

Modbus/ASCII

在此模式下，块结束通过特定 (CR/LF) 字符序列识别。USART 通过字符匹配功能管理此机制。

通过在 ADD[7:0] 字段中编程 LF ASCII 码以及激活字符匹配中断 (CMIE=1)，软件可在接收到 LF 时获得通知并检查 DMA 缓存区中的 CR/LF。

29.5.9 USART 极性控制

将 USARTx_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度，*表 128* 中列出了可能的 USART 帧格式。

表 128. 帧格式

M 位	PCE 位	USART 帧(1)
00	0	| SB | 8 位数据 | STB |
00	1	| SB | 7 位数据 | PB | STB |
01	0	| SB | 9 位数据 | STB |
01	1	| SB | 8 位数据 | PB | STB |
10	0	| SB | 7 位数据 | STB |
10	1	| SB | 6 位数据 | PB | STB |

1. 图注：SB：起始位，STB：停止位，P：奇偶校验位。在数据寄存器中，PB 始终位于 MSB 位置（第 9 位、第 8 位或第 7 位，具体取决于 M 位的值）。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择偶校验（USARTx_CR1 寄存器中的 PS 位 = 0）时，校验位为 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择奇校验（USARTx_CR1 寄存器中的 PS 位 = 1）时，校验位为 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 USARTx_ISR 寄存器中的 PE 标志置 1；如果 USARTx_CR1 寄存器中 PEIE 位置 1，则会生成中断。通过软件将 1 写入 USARTx_ICR 寄存器中的 PECF 位来清零 PE 标志。

发送时的奇偶校验生成

如果 USARTx_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

29.5.10 USART LIN（局域互连网络）模式

仅在支持 LIN 模式时才与本节相关。请参见*第 700 页上的第 29.4 节：USART 实现*。

通过将 USARTx_CR2 寄存器中的 LINEN 位置 1 来选择 LIN 模式。在 LIN 模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 STOP[1:0] 和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN、HDSEL 和 IREN 位。

有关代码示例，请参见 *A.17.6：USART LIN 模式代码示例*。

LIN 发送

与正常的 USART 发送相比，在 LIN 主器件中发送时必须采用*第 29.5.2 节：USART 发送器*中 介绍的步骤，同时还具有以下区别：

- M 位清零以配置 8 位字长度。
- LINEN 位置 1 以进入 LIN 模式。此时，将 SBKRQ 位置 1 会发送 13 个“0”位作为断路字符。然后会发送值为“1”的 2 个位以进行下一启动检测。

LIN 接收

使能 LIN 模式后，将激活断路检测电路。该检测完全独立于正常的 USART 接收器。在空闲状态或某个帧期间，只要发生断路即可检测出来。

接收器（USARTx_CR1 寄存器中 RE=1）使能后，电路便开始监测启动信号的 RX 输入。检测起始位的方法与搜索断路字符或数据的方法相同。检测到起始位后，电路会对接下来的位进行采样，方法与数据采样相同（第 8、第 9 和第 10 次采样）。如果 10 个（USARTx_CR2 中 LBDL = 0 时）或 11 个（USARTx_CR2 中 LBDL=1 时）连续位均检测为“0”，且其后跟随分隔符，则 USARTx_ISR 寄存器中的 LBDF 标志将置 1。如果 LBDIE 位=1，则会生成中断。在验证断路前，会对分隔符进行检查，因为它表示 RX 线路已恢复到高电平。

如果在第 10 或第 11 次采样前已对“1”采样，则断路检测电路会取消当前检测，并重新搜索起始位。

如果禁止 LIN 模式 (LINEN=0)，接收器会作为正常的 USART 继续工作，不会再进行断路检测。

如果使能 LIN 模式 (LINEN=1)，只要发生帧错误（例如，在“0”处检测到停止位，这种情况可能出现在任何断路帧中），接收器即会停止，直到断路检测电路接收到“1”（断路字不完整时）或接收到分隔符（检测到断路时）为止。

*第 721 页的图 244：LIN 模式下的断路检测（11 位断路长度——LBDL 位置 1）*中显示了断路检测器状态机和断路标志的行为。

*第 722 页的图 245：LIN 模式下的断路检测与帧错误检测*中列出了断路帧的示例。

图 244. LIN 模式下的断路检测（11 位断路长度——LBDL 位置 1）

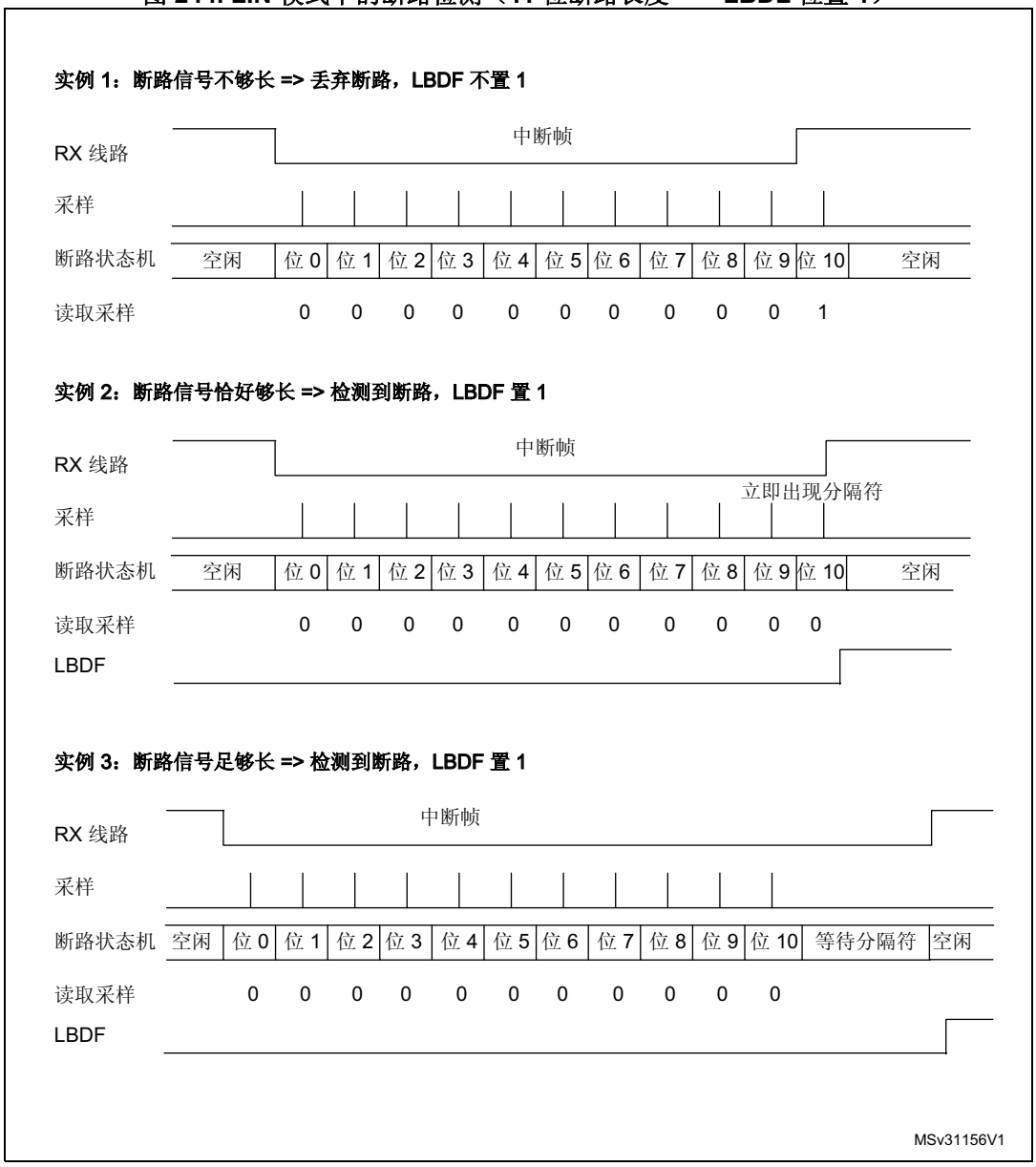
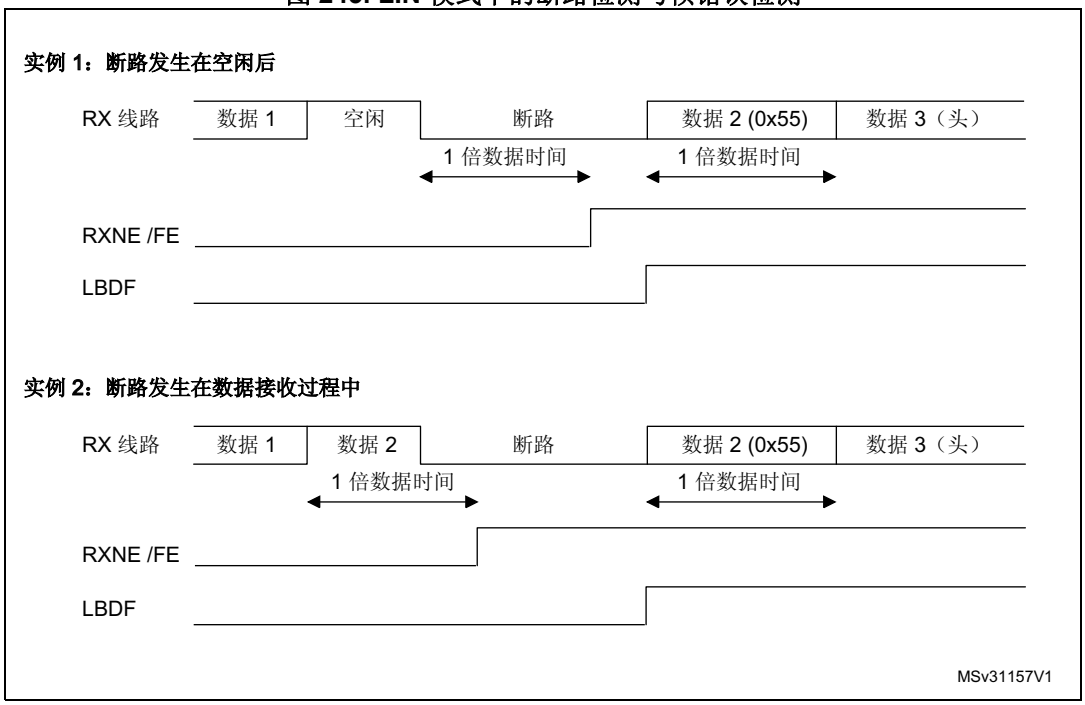


图 245. LIN 模式下的断路检测与帧错误检测



29.5.11 USART 同步模式

通过将 USARTx_CR2 寄存器中的 CLKEN 位写入 1 来选择同步模式。在同步模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位，
- USARTx_CR3 寄存器中的 SCEN、HDSEL 和 IREN 位。

在此模式下，USART 可用于在主模式下控制双向同步串行通信。CK 引脚是 USART 发送器时钟的输出。在起始位或停止位期间，不会向 CK 引脚发送时钟脉冲。在最后一个有效数据位（地址标记）期间，会（也可能不会）生成时钟脉冲，这取决于 USARTx_CR2 寄存器中 LBCL 位的状态。USARTx_CR2 寄存器中的 CPOL 位用于选择时钟极性；USARTx_CR2 寄存器中的 CPHA 位用于选择外部时钟的相位（请参见*图 246*、*图 247* 和*图 248*）。

在空闲状态、报头模式和发送断路期间，外部 CK 时钟处于未激活状态。

USART 发送器在同步模式下的工作方式与异步模式下完全相同。但是由于 CK 与 TX 同步（根据 CPOL 和 CPHA），因此 TX 上的数据是同步的。

在此模式下，USART 接收器的工作方式与异步模式下不同。如果 RE=1，则数据在 CK 上采样（上升或下降沿，取决于 CPOL 和 CPHA），而不会进行任何过采样。此时必须确保建立时间和保持时间（取决于波特率：1/16 位持续时间）。

注： *CK 引脚可与 TX 引脚结合使用。因此，仅当使能发送器 (TE=1) 且正在发送数据时（数据寄存器 USARTx_TDR 已写入），才会提供时钟。这意味着，没有发送数据的情况下无法接收同步数据。*

当 USART 被禁止时 (UE=0)，必须选择 LBCL、CPOL 和 CPHA 位以确保时钟脉冲正常工作。

有关代码示例，请参见 A.17.7：USART 同步模式代码示例。

图 246. USART 同步发送示例

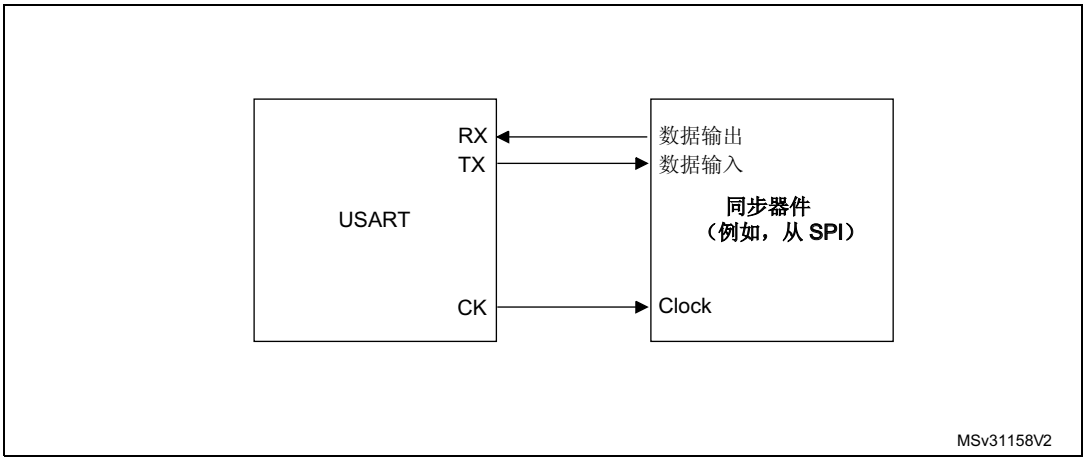


图 247. USART 数据时钟时序图 (M 位 = 00)

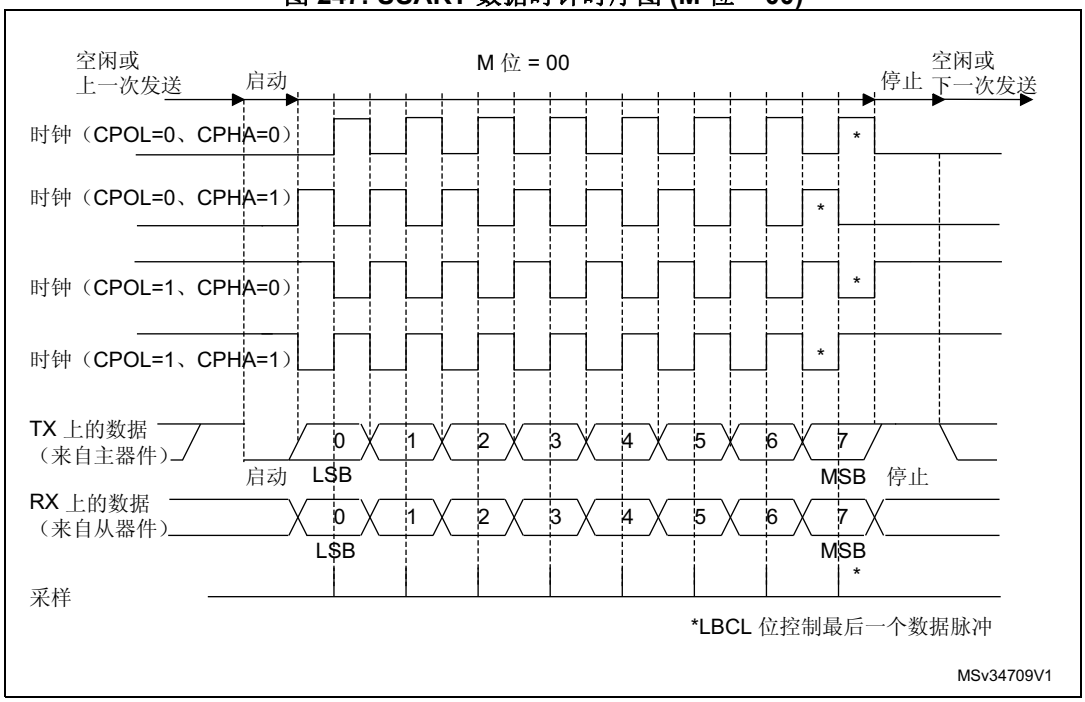


图 248. USART 数据时钟时序图 (M 位 = 01)

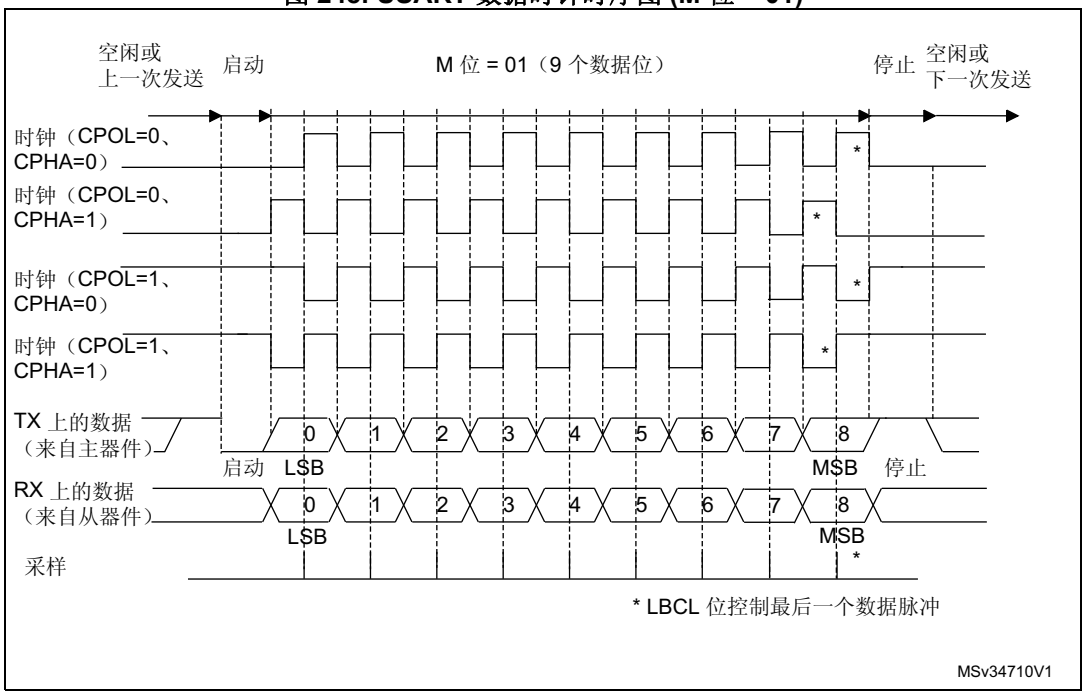
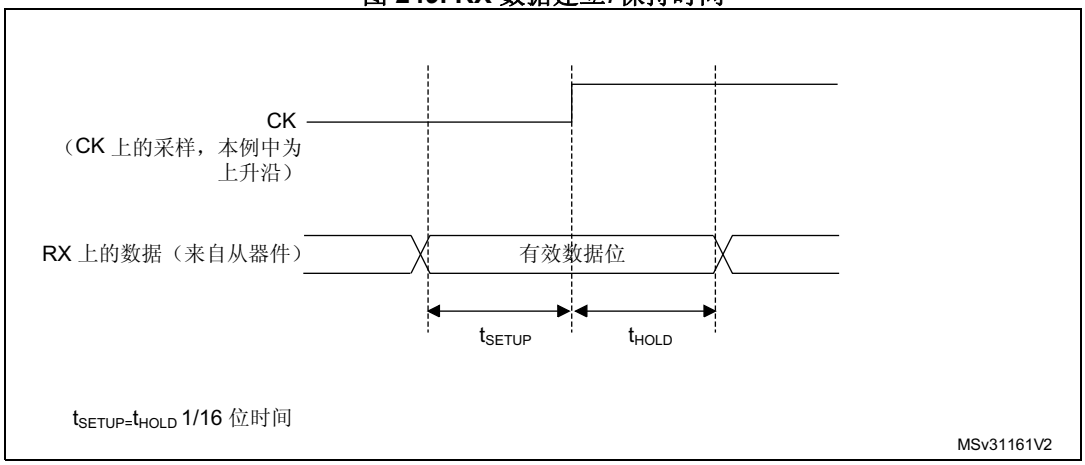


图 249. RX 数据建立/保持时间



注： *在智能卡模式下，CK 的功能有所不同。更多详细信息，请参见第 29.5.13 节：USART 智能卡模式。*

29.5.12 USART 单线半双工通信

通过将 USARTx_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN 和 IREN 位。

USART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用 USARTx_CR3 寄存器中的控制位 HDSEL 可在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时，TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便将 TX 配置为复用功能开漏并外接上拉电阻。

除此之外，通信协议与正常 USART 模式下的通信协议相似。此线路上的任何冲突必须由软件管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

有关代码示例，请参见 *A.17.8：USART 单线半双工代码示例*。

29.5.13 USART 智能卡模式

仅在支持智能卡模式时才涉及本节内容。请参见*第 700 页上的第 29.4 节：USART 实现*。

通过将 USARTx_CR3 寄存器中的 SCEN 位置 1 选择智能卡模式。在智能卡模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN 位，
- USARTx_CR3 寄存器中的 HDSEL 和 IREN 位。

此外，可能需要将 CLKEN 位置 1，以便为智能卡提供时钟。

智能卡接口支持符合 ISO 7816-3 标准的异步协议智能卡。同时支持 T=0（字符模式）和 T=1（块模式）。

USART 应如下所示进行配置：

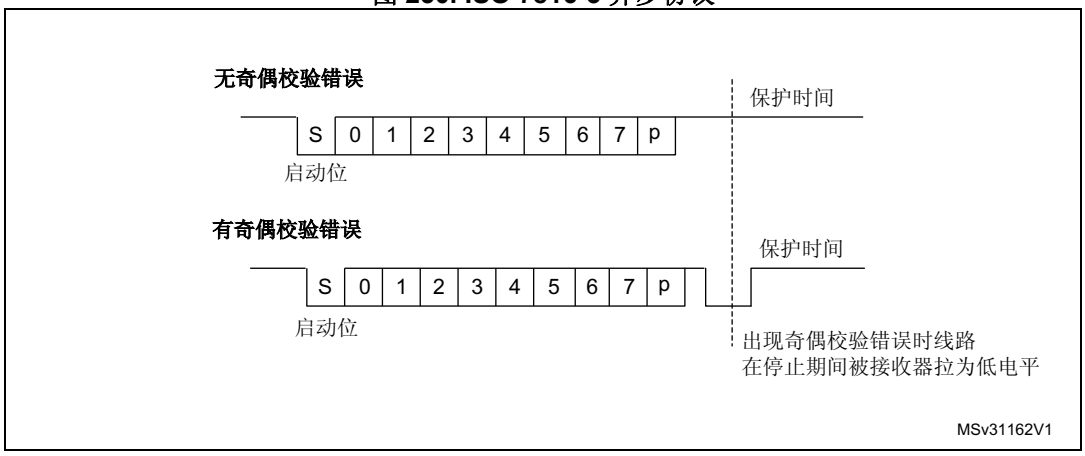
- 8 个位加奇偶校验：其中字长度设置为 8 位，USARTx_CR1 寄存器中 PCE=1
- 1.5 个停止位：其中 USARTx_CR2 寄存器中 STOP=11。接收时也可以选择 0.5 个停止位。

有关代码示例，请参见 *A.17.9：USART 智能卡模式代码示例*。

在 T=0（字符）模式下，奇偶校验错误在保护时间周期内的每个字符结束时指示。

图 250 显示了有奇偶校验错误和无奇偶校验错误时数据线上情况的示例。

图 250. ISO 7816-3 异步协议



连接到智能卡时，USART 的 TX 输出会驱动一条双向线（它也由智能卡驱动）。必须将 TX 引脚配置为开漏引脚。

智能卡模式采用单线半双工通信协议。

- 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个波特时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特时钟周期的延迟。
- 发送时，如果智能卡检测到奇偶校验错误，它会通过将线路驱动为低电平 (NACK) 告知 USART 此状态。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5 个停止位）出现帧错误。USART 可根据协议自动重新发送数据。重试次数在 SCARCNT 位域中编程。如果经过编程次数的重试后，USART 继续收到 NACK，USART 会停止发送并将该错误以帧错误形式发出。可以使用 USARTx_RQR 寄存器中的 TXFRQ 位将 TXE 位置 1。
- 发送时智能卡自动重试：在 USART 检测 NACK 与重复字符的起始位之间插入 2.5 个波特率周期的延迟。接收最后一个重复字符后，立即将 TC 位置 1（无保护时间）。如果软件要重复此操作，必须确保标准要求的最短 2 个波特率周期。
- 如果在接收一个使用 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到 USART 的数据尚未正确接收。如果 NACK 控制位置 1，接收器会向奇偶校验错误发送“NACK”信号；否则不会发送 NACK 信号（将在 T=1 模式下会使用）。如果接收到的字符错误，则不会激活 RXNE/接收 DMA 请求。根据协议规范，智能卡必须重新发送相同的字符。如果经过 SCARCNT 位域中指定的最大重试次数后接收到的字符仍然错误，USART 会停止发送 NACK 信号，并将错误以奇偶校验错误的形式发出。
- 接收时智能卡自动重试：如果 USART 向智能卡发送 NACK 信号，但智能卡不重复字符，则 BUSY 标志将保持置 1。
- 发送时，USART 会在两个连续字符之间插入保护时间（按照保护时间寄存器中编程的值）。由于保护时间在前一个字符的停止位后测量，因此必须将 GT[7:0] 寄存器编程为所需 CGT（字符保护时间，如 7816-3 规范所定义）减去 12（一个字符的持续时间）。
- 通过对保护时间寄存器进行编程，可以延迟 TC 标志的置位。正常工作时，当发送移位寄存器为空且没有新的发送请求出现时，会对 TC 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间，TC 标志被强制为低电平。当保护时间计数器达到设置值时，TC 置位为高电平。

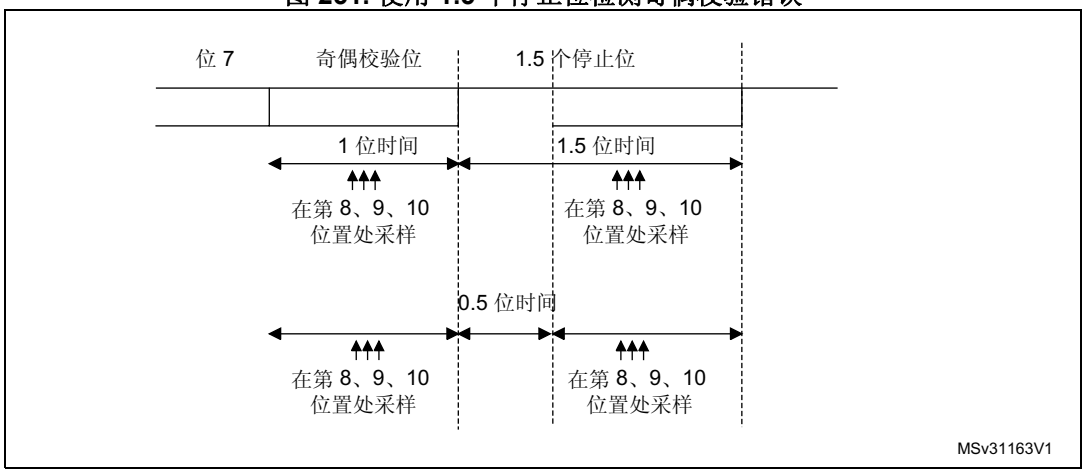
- TC 标志的释放不受智能卡模式的影响。
- 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收块不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持续时间可以是 1 或 2 个时钟周期。
- 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注： *中断字符在智能卡模式下无效。带有帧错误的 0x00 数据被视为数据，而非中断。*

当翻转 TE 位时，不会发送空闲帧。空闲帧（在其它配置中进行了定义）在 ISO 协议中未进行定义。

图 251 详细介绍了 USART 如何对 NACK 信号采样。在本例中，USART 正在发送数据并配置了 1.5 个停止位。USART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

图 251. 使用 1.5 个停止位检测奇偶校验错误



USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式下，CK 仅通过一个 5 位预分频器由内部外设输入时钟提供。分频比在预分频器寄存器 USARTx_GTPR 中进行配置。CK 频率可在 $f_{CK/2}$ 到 $f_{CK/62}$ 之间进行编程，其中 f_{CK} 为外设输入时钟。

块模式 (T=1)

在 T=1（块）模式下，通过将 UART_CR3 寄存器中的 NACK 位清零可停用奇偶校验错误发送。

在块模式下从智能卡请求读操作时，软件必须通过将 USART_CR2 寄存器中的 RTOEN 位置 1 使能接收器超时功能，并将 RTOR 寄存器中的 RTO 位域编程为 BWT（块等待时间）- 11 值。如果此时间段后未从智能卡接收到应答，RTOF 标志将置 1，并生成超时中断（如果 USART_CR1 寄存器中的 RTOIE 位置 1）。如果该时间段后接收到第一个字符，则通过 RXNE 中断发出信号指示。

注： *即使在使用 DMA 模式下的 USART 从块模式下的智能卡读取时，也必须使能 RXNE 中断。同时，只有在接收到第一个字节后才可使能 DMA。*

接收到第一个字符（RXNE 中断）后，为允许自动检查两个连续字符间的最大等待时间，必须将 RTOR 寄存器中的 RTO 位域编程为 CWT（字符等待时间）- 11 值。此时间以波特率时间单位表示。前一个字符结束后，如果智能卡未在小于 CWT 的时间段内发送新字符，USART 将通过 RTOF 标志和中断（当 RTOIE 位置 1 时）向软件指示此情况。

注： *RTO 计数器遵循以下规则开始计数：*

- STOP = 00 时从停止位结束时开始计数。

- STOP = 10 时从第二个停止位结束时开始计数。

- STOP = 11 时从 STOP 位结束后开始计数 1 个位的持续时间。

- STOP = 01 时从 STOP 位开始时开始计数。

按照智能卡协议定义，BWT/CWT 的值从最后一个字符开始（起始位）时定义。必须将 RTO 寄存器分别编程为 BWT -11 或 CWT -11，并考虑最后一个字符本身的长度。

块长度计数器用于对 USART 接收到的所有字符进行计数。当 USART 进行发送 (TXE=0) 时，此计数器复位。块长度由智能卡在块的第三个字节（起始字段）中传达。必须将此值编程到 USARTx_RTOR 寄存器中的 BLEN 字段。使用 DMA 模式时，在块开始之前，必须将此寄存器字段编程为最小值 (0x0)。对于该值，在接收到第四个字符后生成中断。软件必须读取 LEN 字段（第三个字节），其值必须从接收缓存区中读取。

在中断驱动接收模式下，块长度可以由软件或者通过编程 BLEN 值来检查。但是在块开始前，可以编程 BLEN 的最大值 (0xFF)。接收到第三个字符后，将编程实际值。

如果块使用 LRC（纵向冗余校验，1 个结尾字节），则 BLEN=LEN。如果块使用 CRC 机制（2 个结尾字节），则必须编程 BLEN=LEN+1。块总长度（包括起始字段、结尾字段和信息字段）等于 BLEN+4。块结束的信号通过 EOBF 标志和中断（EOBIE 位置 1 时）发送给软件。

如果块长度出现错误，则通过 RTO 中断（字符等待时间上溢）发送块结束信号。

注： *错误检查代码 (LRC/CRC) 必须通过软件计算/验证。*

正向约定和反向约定

智能卡协议定义了两种约定：正向约定和反向约定。

正向约定定义为：LSB 在前，逻辑位值 1 对应于线路的 H 状态，奇偶校验为偶校验。要使用此约定，必须编程以下控制位：MSBFIRST=0，DATAINV=0（默认值）。

反向约定定义为：MSB 在前，逻辑位值 1 对应于信号线路的 L 状态，奇偶校验为偶校验。要使用此约定，必须编程以下控制位：MSBFIRST=1，DATAINV=1。

注： *将逻辑数据值取反（0=H，1=L）时，奇偶校验位将同样取反。*

为识别智能卡约定，智能卡会将初始字符 TS 作为 ATR（复位应答）的第一个字符发送。TS 支持两种格式：LHHL LLL LLH 和 LHHL HHH LLH。

- (H) LHHL LLL LLH 建立反向约定：状态 L 编码为值 1，时间分量 2 传送最高有效位（MSB 在前）。按反向约定解码时，传送的字节等于“3F”。
- (H) LHHL HHH LLH 建立正向约定：状态 H 编码为值 1，时间分量 2 传送最低有效位（LSB 在前）。按正向约定解码时，传送的字节等于“3B”。

在 2 到 10 的九个时间分量中，如果有偶数个位设置为 1，则字符的奇偶校验正确。

由于 USART 不了解智能卡使用哪种约定，因此 USART 需要能够识别任意一种模式并相应操作。模式识别不在硬件中完成，而是通过软件序列完成。此外，假设以正向约定配置 USART（默认）而智能卡以反向约定 (TS = LHHL LLL LLH) 应答，则 USART 接收到的字节将为“03”，奇偶校验将为奇校验。

因此，有两种方法可用于识别 TS 模式：

方法 1

将 USART 编程为标准智能卡模式/正向约定。这种情况下，TS 模式接收会向智能卡生成奇偶校验错误中断和错误信号。

- 奇偶校验错误中断通知软件智能卡未以正向约定正确应答。之后，软件重新将 USART 编程为反向约定
- 为响应错误信号，智能卡会重试同一 TS 字符，此时重新编程后的 USART 将正确接收该字符

或者，为应答奇偶校验错误中断，软件可决定重新编程 USART，同时向智能卡生成新的复位命令，然后再次等待 TS。

方法 2

将 USART 编程为 9 位/无奇偶校验模式，无位反向。在此模式下，按如下方式接收两种 TS 模式中的任一种：

(H) LHHL LLL LLH = 0x103 -> 选择反向约定

(H) LHHL LLL LLH = 0x13B -> 选择正向约定

软件根据这两种模式检查接收到的字符，如果其中任意一种匹配，则相应编程 USART 以接收下一个字符。

如果两种都未被识别，则可能复位智能卡以重新开始协商。

29.5.14 USART IrDA SIR ENDEC 模块

仅在支持 IrDA 模式时才涉及本节内容。请参见*第 700 页上的第 29.4 节：USART 实现*。

通过将 USARTx_CR3 寄存器中的 IREN 位置 1 来选择 IrDA 模式。在 IrDA 模式下，必须将以下位清零：

- USARTx_CR2 寄存器中的 LINEN、STOP 和 CLKEN 位，
- USARTx_CR3 寄存器中的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零 (RZI) 调制方案，它以红外光脉冲表示逻辑 0（参见*图 252*）。

SIR 发送编码器用于调制 USART 发出的非归零 (NRZ) 位流。输出脉冲流会发送到外部输出驱动器和红外线 LED。USART 支持的 SIR ENDEC 比特率最高为 115.2 Kbps。在正常模式下，所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器用于解调由红外探测器发出的归零位流，并将接收到的 NRZ 串行位流输出到 USART。在空闲状态下，解码器输入通常为高电平（标记状态）。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时，会检测到起始位。

- IrDA 是一个半双工通信协议。如果发送器忙（USART 正在向 IrDA 编码器发送数据时），则 IrDA 解码器会忽略 IrDA 接收线上的所有数据；如果接收器忙（USART 正在接收来自 IrDA 解码器的解码数据时），则 IrDA 不会对 USART 发送到 IrDA 的 TX 上的数据进行编码。接收数据时，应避免同时进行发送，因为这样做可能会破坏要发送的数据。
- 0 作为高电平脉冲发送，而 1 作为 0 发送。在正常模式下，脉冲宽度规定为所选位周期的 3/16（参见*图 253*）。
- SIR 解码器用于将兼容 IrDA 的接收信号转换为 USART 的位流。
- SIR 接收逻辑将高电平状态视为逻辑“1”，将低电平脉冲视为逻辑“0”。
- 发送编码器输出的极性与解码器输入相反。SIR 输出在空闲时处于低电平状态。

- IIrDA 规范要求脉冲容忍值要大于 1.41 µs。可接受的脉冲宽度可通过寄存器设置。接收器端的干扰检测逻辑会滤除宽度小于 2 个 PSC 周期的脉冲（PSC 是在 USARTx_GTPR 中编程的预分频器值）。宽度小于 1 个 PSC 周期的脉冲都将被拒绝，但宽度大于 1 个而小于 2 个周期的脉冲可能被接受也可能被拒绝，而宽度大于 2 个周期的脉冲将被接受作为有效脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器能够与低功耗发送器进行通信。
- 在 IrDA 模式下，USARTx_CR2 寄存器中的 STOP 位必须配置为“1 个停止位”。

有关代码示例，请参见 *A.17.10：USART IrDA 模式代码示例*。

IrDA 低功耗模式

发送器

在低功耗模式下，脉冲宽度不再保持为位周期的 3/16。此时的脉冲宽度为低功耗波特率的 3 倍，最小可为 1.42 MHz。

通常此值是 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz)。低功耗模式下的可编程分频器会对系统时钟进行分频，以达到此值。

接收器

在低功耗模式下接收与在正常模式下接收类似。为进行干扰检测，USART 应丢弃持续时间短于 1 个 PSC 周期的脉冲。只有当持续时间大于 2 个 IrDA 低功耗波特时钟周期（USARTx_GTPR 寄存器中的 PSC 值）时，才是有效低电平。

注： *宽度小于两个但大于一个 PSC 周期的脉冲可能被接受，也可能被拒绝。*

接收器的建立时间应由软件进行管理。IrDA 物理层规范规定发送和接收之间至少要经过 10 ms 的延迟（IrDA 是一个半双工协议）。

图 252. IrDA SIR ENDEC——框图

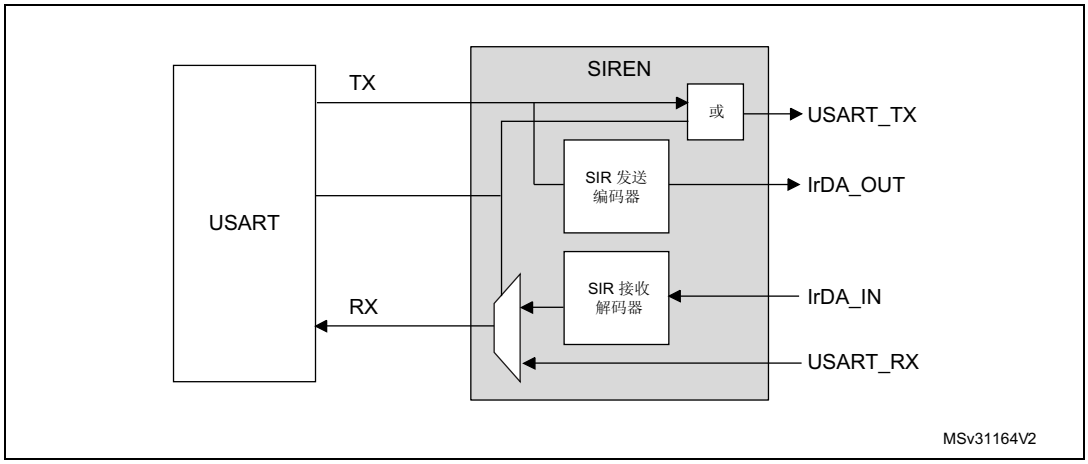
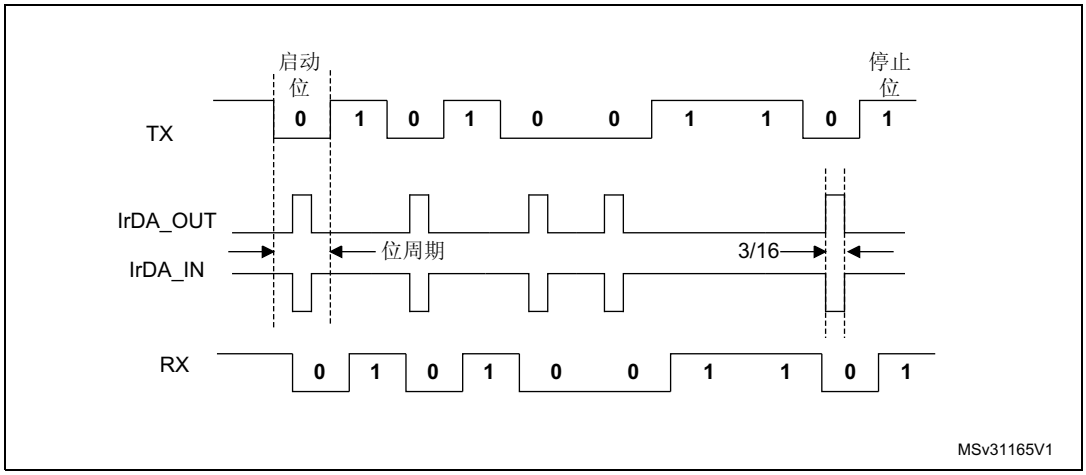


图 253. IrDA 数据调制 (3/16)——正常模式



29.5.15 DMA 模式下的 USART 连续通信

USART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立生成的。

注： *要确定是否支持 DMA 模式，请参见第 700 页上的第 29.4 节：USART 实现。*如果不支持 DMA 模式，请按照*第 29.5.2 节：USART 发送器*或*第 29.5.3 节：USART 接收器*中的*解释说明使用 USART。可以将 USARTx_ISR 寄存器中的 TXE/RXNE 标志清零，从而实现连续通信。*

有关代码示例，请参见 *A.17.11：USART DMA 代码示例*。

使用 DMA 进行发送

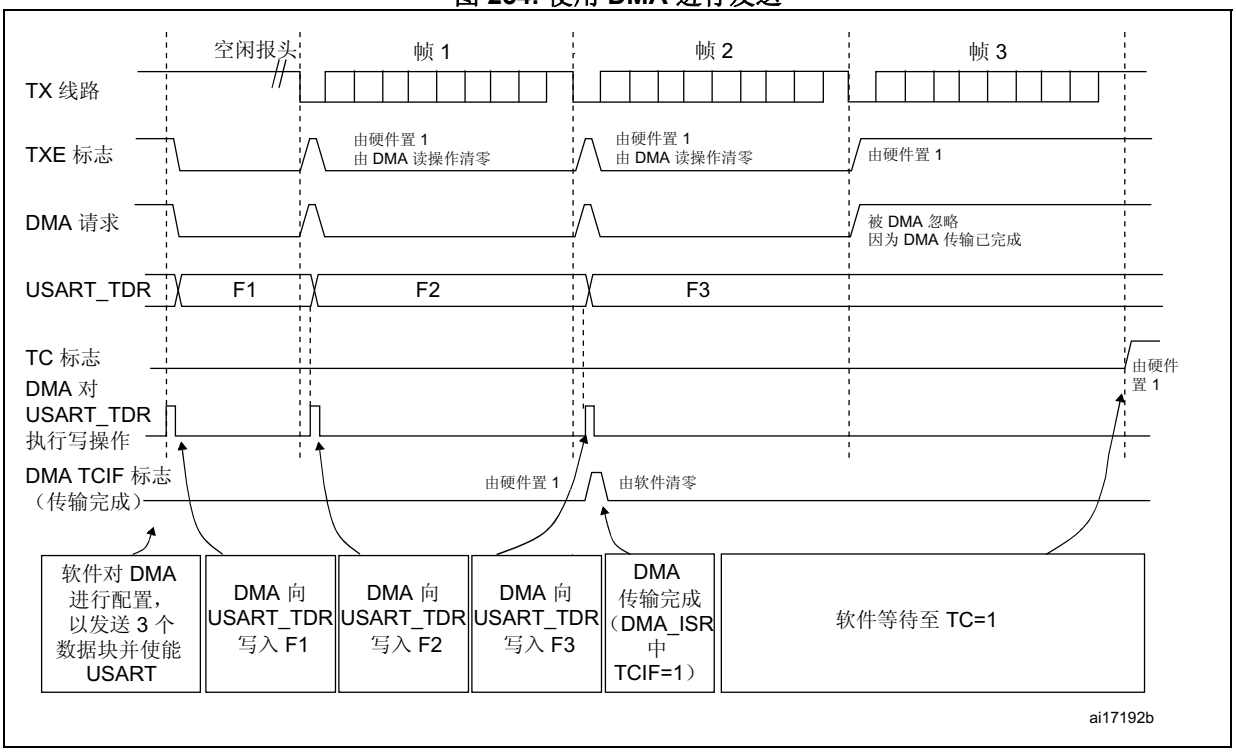
将 USARTx_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 外设，请参见*第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）加载到 USARTx_TDR 寄存器。要映射一个 DMA 通道以进行 USART 发送，请按以下步骤操作（x 表示通道编号）：

1. 在 DMA 控制寄存器中写入 USARTx_TDR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都从这个存储区域加载到 USARTx_TDR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。
6. 通过将 USARTx_ICR 寄存器中的 TCCF 位置 1，将 USARTx_ISR 寄存器中的 TC 标志清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

在发送模式下，DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 USART 通信已完成。在禁止 USART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。TC 标志在所有数据发送期间都保持清零状态，然后在最后一帧发送结束时由硬件置 1。

图 254. 使用 DMA 进行发送



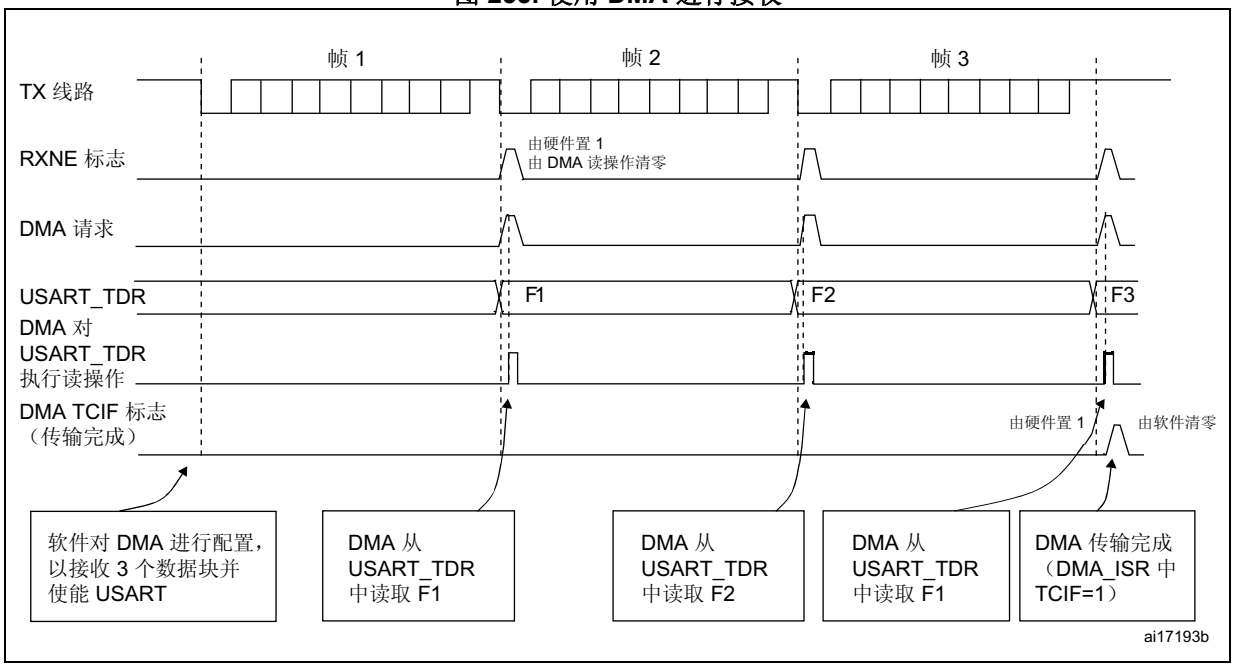
使用 DMA 进行接收

将 USARTx_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 USARTx_RDR 寄存器加载到 SRAM 区域中（通过 DMA 外设，请参见 *第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）。要映射一个 DMA 通道以进行 USART 接收，请按以下步骤操作：

1. 在 DMA 控制寄存器中写入 USARTx_RDR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都从此地址移动到存储器。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都从 USARTx_RDR 寄存器加载到此存储区。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 控制寄存器中配置通道优先级。
5. 根据应用的需求，在完成一半或全部传输后产生中断。
6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

图 255. 使用 DMA 进行接收



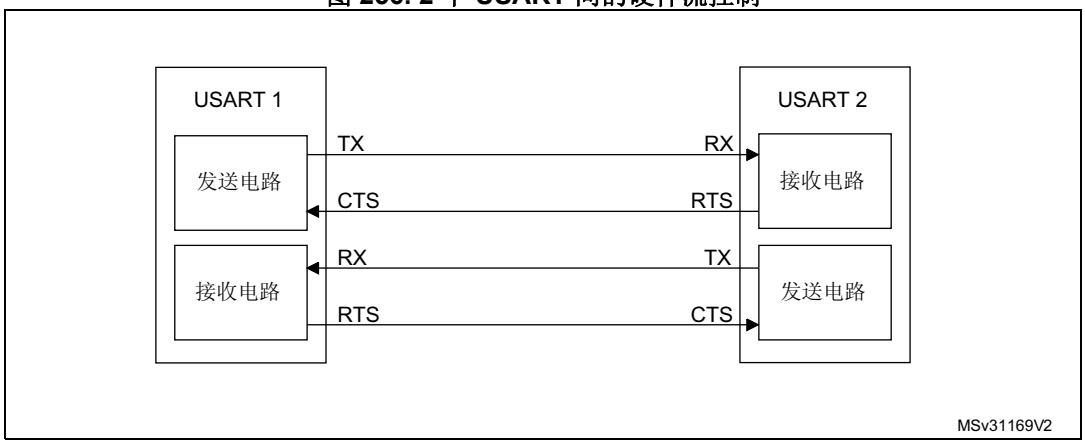
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能标志置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（USARTx_CR3 寄存器中的 EIE 位）；如果该位置 1，在发生其中任何一个错误时，在当前字节后使能中断。

29.5.16 RS232 硬件流控制和 RS485 驱动器使能（使用 USART）

使用 CTS 输入和 RTS 输出可以控制 2 个器件间的串行数据流。*图 256* 显示了在这种模式下如何连接 2 个器件：

图 256. 2 个 USART 间的硬件流控制

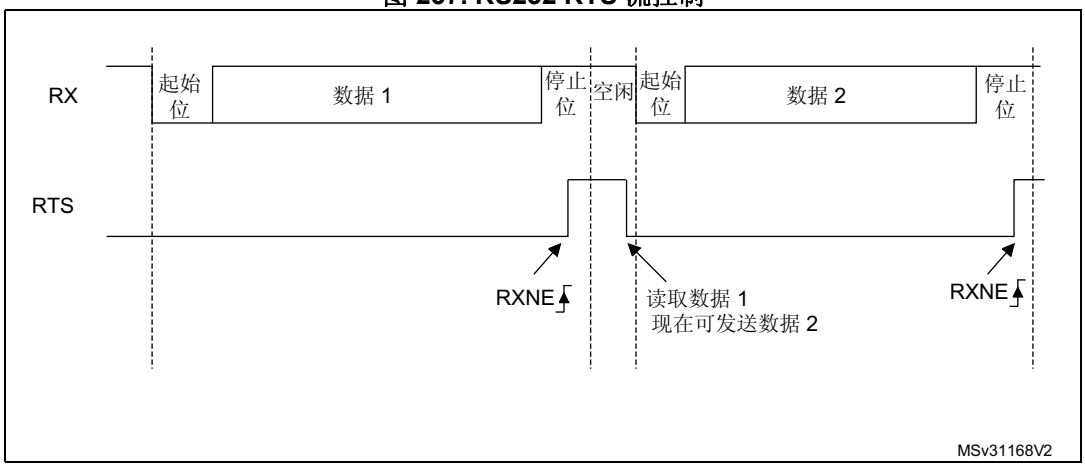


分别向 USARTx_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RS232 RTS 和 CTS 流控制。

RS232 RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 USART 接收器准备好接收新数据，便会将 RTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 RTS 变为无效，表明发送过程会在当前帧结束后停止。*图 257* 显示了在使能 RTS 流控制的情况下进行通信的示例。

图 257. RS232 RTS 流控制

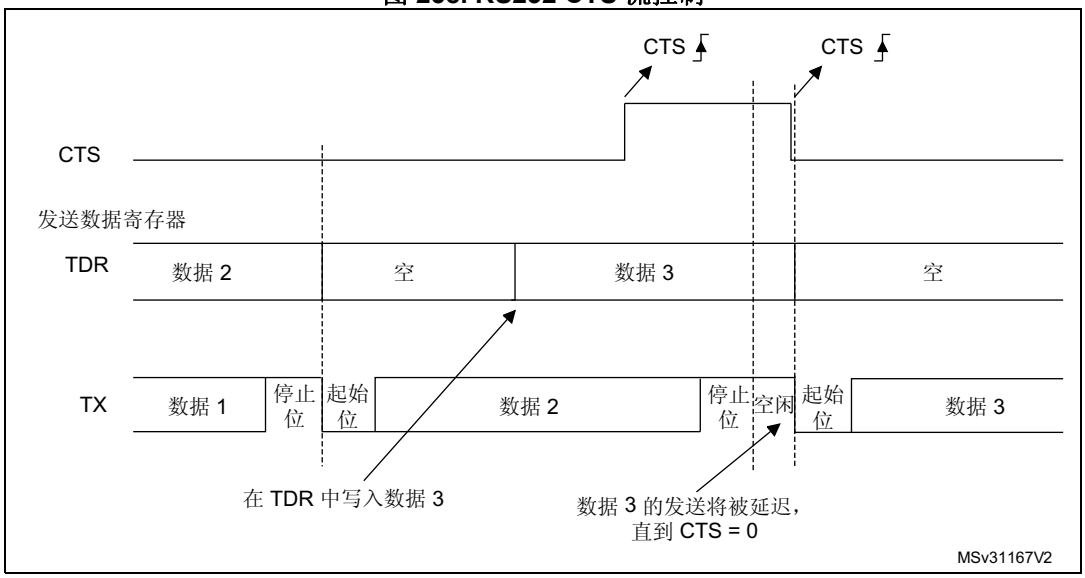


RS232 CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 CTS。如果 CTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 CTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 CTS 发生变化，CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 USARTx_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。*图 258* 显示了在使能 CTS 流控制的情况下进行通信的示例。

图 258. RS232 CTS 流控制



注：为正常运行，必须在当前字符结束前至少 3 个 USART 时钟源周期内使能 CTS。此外还应注意，当脉冲短于 2 个 PCLK 周期时无法将 CTSCF 标志置 1。

有关代码示例，请参见 *A.17.12：USART 硬件流控制代码示例*。

RS485 驱动器使能

驱动器使能功能可通过将 USARTx_CR3 控制寄存器中的 DEM 位置 1 使能。这样用户便可通过 DE（驱动器使能）信号激活外部收发器控制。使能时间为激活 DE 信号与 START 位开始间的时间。可以通过 USARTx_CR1 控制寄存器中的 DEAT [4:0] 位域编程使能时间。禁止时间为发送的消息中最后一个停止位结束与取消激活 DE 信号间的时间。可以通过 USARTx_CR1 控制寄存器中的 DEDT [4:0] 位域编程禁止时间。DE 信号的极性可使用 USARTx_CR3 控制寄存器中的 DEP 位配置。

在 USART 中，DEAT 和 DEDT 以采样时间单位表示（1/8 或 1/16 位持续时间，具体取决于过采样速率）。

29.5.17 使用 USART 从停止模式唤醒

当 UESM 位置 1 且 USART 时钟设置为 HSI 或 LSE 时，USART 能够将 MCU 从停止模式唤醒（请参见复位和时钟控制 (RCC) 部分）。

可使用标准 RXNE 中断将 MCU 从停止模式唤醒。在这种情况下，必须在进入停止模式前将 RXNEIE 位置 1。

也可通过 WUS 位域选择特定中断。

为了能够将 MCU 从停止模式唤醒，必须在进入停止模式前将 USARTx_CR1 控制寄存器中的 UESM 位置 1。

检测到唤醒事件后，WUF 标志会由硬件置 1 并在 WUFIE 位置 1 时生成一个唤醒中断。

注：在进入停止模式前，用户必须确保 USART 未在执行传输。BUSY 标志无法确保运行接收期间始终不进入停止模式。

WUF 标志在检测到唤醒事件时置 1，而与 MCU 处于停止模式还是工作模式无关。

如果在初始化和使能接收器后立即进入停止模式，则必须校验 REACK 位以确保 USART 确实已使能。

当 DMA 用于接收时，它必须在进入停止模式前禁止，并在退出停止模式后重新使能。

从停止模式唤醒功能并非在所有模式下均可用。例如，该功能在 SPI 模式下不起作用，因为 SPI 仅在主模式下工作。

使用静默模式和停止模式

如果 USART 在进入停止模式前处于静默模式：

- 不得使用空闲检测时从静默模式唤醒，因为空闲检测无法在停止模式下工作。
- 如果使用地址匹配时从静默模式唤醒，则从停止模式唤醒的源也必须是地址匹配。如果 RXNE 标志在进入停止模式时置 1，则接口将在地址匹配时和从停止模式唤醒时保持静默模式。
- 如果 USART 配置为在 START 位检测时将 MCU 从停止模式唤醒，WUF 标志将置 1 但 RXNE 标志不置 1。

当 USART 时钟源为 HSI 时钟时，确定最大 USART 波特率允许从停止模式正确唤醒

允许从停止模式正确唤醒的最大波特率取决于：

- 器件数据手册中提供的参数 $t_{WUUSART}$
- *第 29.5.5 节：USART 接收器对时钟偏差的容差*中提供的 USART 接收器的容差。

举例来说：OVER8 = 0，M 位 = 10，ONEBIT = 1，BRR [3:0] = 0000。

在这些条件下，根据*表 126：BRR [3:0] = 0000 时的 USART 接收器容差*，USART 接收器的容差为 4.86 %。

DTRA + DQUANT + DREC + DTCL + DWU < USART 接收器的容差

DWU max = $t_{WUUSART}$ / (11 x Tbit Min)

Tbit Min = $t_{WUUSART}$ / (11 x DWU max)

考虑一种理想情况：参数 DTRA、DQUANT、DREC 和 DTCL 为 0%，DWU 最大值为 4.86 %。实际上，我们至少需要考虑 HSI 不精确的情况。

假设 HSI 不精确度 = 1 %，则 $t_{WUUSART}$ = 8.1 µs（处于停止模式，主调压器处于运行模式，范围 1）：

DWU max = 4.86 % - 1 % = 3.86 %

Tbit min = 8.1 µs / (9 x 3.86 %) = 23.31 µs。

在这些情况下，允许从停止模式正确唤醒的最大波特率为 1/23.31 µs = 42 K 波特率。

29.6 USART 低功耗模式

表 129. 低功耗模式对 USART 的影响

模式	说明
睡眠	无影响。USART 中断可使器件退出睡眠模式。
低功耗运行	无影响。
低功耗睡眠	无影响。USART 中断可使器件退出低功耗睡眠模式。
停止	当 UESM 位置 1 且 USART 时钟设置为 HSI16 或 LSE 时，USART 能够将 MCU 从停止模式唤醒。 可使用标准 RXNE 中断将 MCU 从停止模式唤醒。
待机	USART 掉电，当器件退出待机模式后必须重新初始化 USART。

29.7 USART 中断

表 130. USART 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 中断	CTSIF	CTSIE
发送完成	TC	TCIE
接收数据寄存器不为空（已准备好读取数据）	RXNE	RXNEIE
检测到溢出错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
LIN 断路	LBDF	LBDIE
多缓冲区通信中的噪声标志、溢出错误和帧错误。	NF 或 ORE 或 FE	EIE
字符匹配	CMF	CMIE
接收器超时	RTOF	RTOIE
块结束	EOBF	EOBIE
从停止模式唤醒	WUF(1)	WUFIE

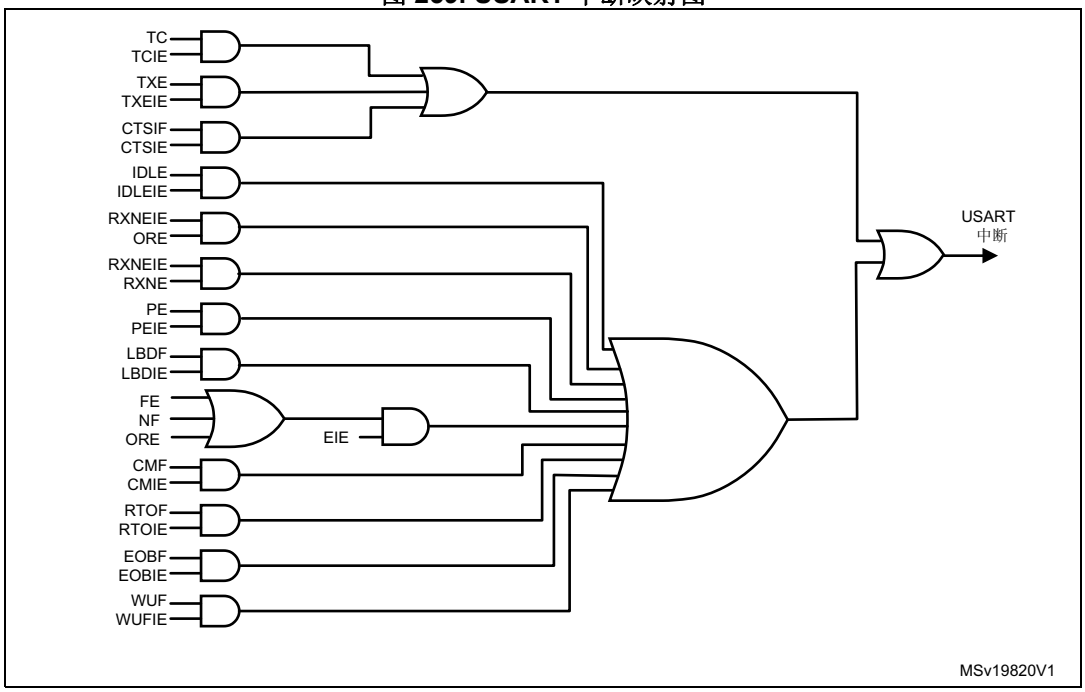
1. WUF 中断仅在停止模式下有效。

USART 中断事件被连接到相同的中断向量（请参见*图 259*）。

- 发送期间：发送完成、清除以发送、发送数据寄存器为空或帧错误（智能卡模式下）中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、LIN 断路检测、噪声标志、帧错误、字符匹配等。

如果相应的使能控制位置 1，则这些事件会生成中断。

图 259. USART 中断映射图



29.8 USART 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

29.8.1 控制寄存器 1 (USARTx_CR1)

Control register 1

偏移地址：0x00

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			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
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:29 保留，必须保持复位值

位 28 **M1**：字长 (Word length)

此位和位 12 (M0) 用于确定字长度。此位由软件置 1 或清零。

M[1:0] = 00：1 个起始位，8 个数据位，n 个停止位

M[1:0] = 01：1 个起始位，9 个数据位，n 个停止位

M[1:0] = 10：1 个起始位，7 个数据位，n 个停止位

只有在禁止 USART (UE=0) 时才能写入此位。

注： 7 位数据长度模式下，不支持智能卡模式、LIN 主模式和自动波特率（0x7F 帧和 0x55 帧检测）。

位 27 **EOBIE**：块结束中断使能 (End of Block interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：USARTx_ISR 寄存器中的 EOBF 标志置 1 时生成 USART 中断

注： 如果 USART 不支持智能卡模式，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 26 **RTOIE**：接收器超时中断使能 (Receiver timeout interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：USARTx_ISR 寄存器中的 RTOF 位置 1 时生成 USART 中断。

注： 如果 USART 不支持接收器超时功能，该位保留并由硬件强制清零。
第 700 页上的第 29.4 节：USART 实现.

位 25:21 **DEAT[4:0]**：驱动器使能使能时间 (Driver Enable assertion time)

该 5 位值用于定义激活 DE（启动器使能）信号与起始位开始间的时间。此时间以采样时间单位表示（1/8 或 1/16 位持续时间，具体取决于过采样速率）。

只有在禁止 USART (UE=0) 后才能写入此位域。

注： 如果不支持驱动器使能功能，该位保留并且必须保持清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 20:16 **DEDT[4:0]**：驱动器使能禁止时间 (Driver Enable de-assertion time)

该 5 位值用于定义发送的消息中最后一个停止位结束与取消激活 DE（驱动器使能）信号间的时间。此时间以采样时间单位表示（1/8 或 1/16 位持续时间，具体取决于过采样速率）。如果在 DEDT 时间内对 USARTx_TDR 寄存器执行写操作，则新数据仅在经过 DEDT 和 DEAT 时间后才会发送。

只有在禁止 USART (UE=0) 后才能写入此位域。

注：　如果不支持驱动器使能功能，该位保留并且必须保持清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 15 **OVER8**：过采样模式 (Oversampling mode)

0：16 倍过采样
1：8 倍过采样
只有在禁止 USART (UE=0) 时才能写入此位。

注：　在 LIN 和 IrDA 模式下，此位必须保持清零。

位 14 **CMIE**：字符匹配中断使能 (Character match interrupt enable)

此位由软件置 1 和清零。
0：禁止中断
1：如果 USARTx_ISR 寄存器中的 CMF 位置 1，则生成 USART 中断。

位 13 **MME**：静默模式使能 (Mute mode enable)

此位用于激活 USART 的静默模式功能，此位置 1 时，USART 可按 WAKE 位定义的方式在活动模式与静默模式之间切换。此位由软件置 1 和清零。
0：接收器永久处于活动模式
1：接收器可在静默模式和活动模式之间切换。

位 12 **M0**：字长 (Word length)

此位和位 28 (M1) 用于确定字长度。此位由软件置 1 或清零。请参见位 28 (M1) 的说明。
只有在禁止 USART (UE=0) 时才能写入此位。

位 11 **WAKE**：接收器唤醒方法 (Receiver wakeup method)

此位用于确定 USART 静默模式的唤醒方法。此位由软件置 1 或清零。
0：空闲线路
1：地址标记
只有在禁止 USART (UE=0) 后才能写入此位域。

位 10 **PCE**：奇偶校验控制使能 (Parity control enable)

该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到 MSB 位置（如果 M=1，则为第 9 位；如果 M=0，则为第 8 位），并对接收到的数据检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1，PCE 在当前字节的后面处于活动状态（在接收和发送时）。
0：禁止奇偶校验控制
1：使能奇偶校验控制
只有在禁止 USART (UE=0) 后才能写入此位域。

位 9 **PS**：奇偶校验选择 (Parity selection)

该位用于在使能奇偶校验生成/检测（PCE 位置 1）时选择奇校验或偶校验。此位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。
0：偶校验
1：奇校验
只有在禁止 USART (UE=0) 后才能写入此位域。

位 8 **PEIE**：PE 中断使能 (PE interrupt enable)

此位由软件置 1 和清零。
0：禁止中断
1：当 USARTx_ISR 寄存器中 PE=1 时，生成 USART 中断

位 7 **TXEIE**：中断使能 (interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USARTx_ISR 寄存器中 TXE=1 时，生成 USART 中断

位 6 **TCIE**：传输完成中断使能 (Transfer complete interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USARTx_ISR 寄存器中的 TC=1 时，生成 USART 中断

位 5 **RXNEIE**：RXNE 中断使能 (RXNE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USARTx_ISR 寄存器中的 ORE=1 或 RXNE=1 时，生成 USART 中断

位 4 **IDLEIE**：IDLE 中断使能 (IDLE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USARTx_ISR 寄存器中的 IDLE=1 时，生成 USART 中断

位 3 **TE**：发送器使能 (Transmitter enable)

该位使能发送器。此位由软件置 1 和清零。

0：禁止发送器

1：使能发送器

注：除了在智能卡模式下以外，传送期间 TE 位上的“0”脉冲（“0”后紧跟的是“1”）会在当前字的后面发送一个报头（空闲线路）。为生成空闲字符，TE 不能立即写入 1。为确保所需的持续时间，软件可轮询 USARTx_ISR 寄存器中的 TEACK 位。

在智能卡模式下，当 TE 置 1 时，在发送开始前存在 1 位的时间延迟。

位 2 **RE**：接收器使能 (Receiver enable)

该位使能接收器。此位由软件置 1 和清零。

0：禁止接收器

1：使能接收器并开始搜索起始位

位 1 **UESM**：停止模式下的 USART 使能 (USART enable in Stop mode)

当此位清零时，USART 无法将 MCU 从停止模式唤醒。

当此位置 1 时，USART 能够将 MCU 从停止模式唤醒，前提是 USART 时钟选择为 HSI16 或 LSE（在 RCC 中）。

此位由软件置 1 和清零。

0：USART 无法将 MCU 从停止模式唤醒。

1：USART 能够将 MCU 从停止模式唤醒。当该功能激活时，USART 的时钟源必须是 HSI16 或 LSE（请参见复位和时钟控制 (RCC) 部分）。

注：建议在进入停止模式前将 UESM 位置 1，并在退出停止模式时将其清零。

如果 USART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

请参见第 700 页上的第 29.4 节：USART 实现。

位 0 **UE**：USART 使能 (USART enable)

此位清零后，USART 预分频器和输出将立即停止，并丢弃当前操作。USART 的配置保留，但 USARTx_ISR 寄存器中的所有状态标志将设置为其默认值。此位由软件置 1 和清零。

0：禁止 USART 预分频器和输出，低功耗模式

1：使能 USART

注：为进入低功耗模式而不在线路上生成错误，之前必须复位 TE 位，并且软件必须等待 USARTx_ISR 寄存器中的 TC 位置 1 后才能复位 UE 位。

UE = 0 时也会复位 DMA 请求，因必须在复位 UE 位前禁止 DMA 通道。

29.8.2 控制寄存器 2 (USARTx_CR2)

Control register 2

偏移地址：0x04

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
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
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

位 31:28 **ADD[7:4]**：USART 节点的地址 (Address of the USART node)

此位域用于指定 USART 节点的地址或要识别的字符代码。

此位域在多处理器通信时于静默模式或停止模式下使用，以通过 7 位地址标记检测进行唤醒。发送器发送字符的 MSB 应为 1。此位域还可用于正常接收和静默模式无效时的字符检测（例如，ModBus 协议中的块结束检测）。这种情况下，接收到的整个字符（8 位）将与 ADD[7:0] 值进行比较，如果匹配，CMF 标志将置 1。

仅在禁止接收 (RE = 0) 或禁止 USART (UE=0) 时才能写入该位域

位 27:24 **ADD[3:0]**：USART 节点的地址 (Address of the USART node)

此位域用于指定 USART 节点的地址或要识别的字符代码。

此位域在多处理器通信时于静默模式或停止模式下使用，以通过地址标记检测进行唤醒。

仅在禁止接收 (RE = 0) 或禁止 USART (UE=0) 时才能写入该位域

位 23 **RTOEN**：接收器超时使能 (Receiver timeout enable)

此位由软件置 1 和清零。

0：禁止接收器超时功能。

1：使能接收器超时功能。

使能此功能后，如果 RX 线路在 RTOR（接收器超时寄存器）中编程的持续时间内处于空闲状态（无接收），则 USARTx_ISR 寄存器中的 RTOF 标志置 1。

注：　如果 USART 不支持接收器超时功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 22:21 **ABRMOD[1:0]**：自动波特率模式 (Auto baud rate mode)

这些位由软件置 1 和清零。

00：通过测量起始位检测波特率。

01：下降沿到下降沿的测量。（接收到的帧必须以一个等于 1 的位开头，即帧 = 10xxxxxx）

10：0x7F 帧检测。

11：0x55 帧检测。

仅在 ABREN = 0 时或禁止 USART (UE=0) 时才能写入该位域。

注：　如果 DATAINV=1 且/或 MSBFIRST=1，这些模式必须与在线路上时相同，例如 MSBFIRST 的 0xAA

如果 USART 不支持自动波特率功能，则此位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 20 **ABREN**：自动波特率使能 (Auto baud rate enable)

此位由软件置 1 和清零。

0：禁止自动波特率检测。

1：使能自动波特率检测。

注：　如果 USART 不支持自动波特率功能，则此位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 19 **MSBFIRST**：最高有效位在前 (Most significant bit first)

此位由软件置 1 和清零。

0：发送/接收数据时位 0 在前，后跟起始位。

1：发送/接收数据时 MSB（位 7/8/9）在前，后跟起始位。

只有在禁止 USART (UE=0) 后才能写入此位域。

位 18 **DATAINV**：二进制数据反向 (Binary data inversion)

此位由软件置 1 和清零。

0：按正/正向逻辑发送/接收数据寄存器中的逻辑数据。（1=H，0=L）

1：按负/反向逻辑发送/接收数据寄存器中的逻辑数据。（1=L，0=H）。奇偶校验位也取反。

只有在禁止 USART (UE=0) 后才能写入此位域。

位 17 **TXINV**：TX 引脚有效电平反向 (TX pin active level inversion)

此位由软件置 1 和清零。

0：TX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲，Gnd = 0/标记）工作

1：对 TX 引脚信号值取反。（V_{DD} = 0/标记，Gnd=1/空闲）。

允许在 TX 线路上使用外部反相器。

只有在禁止 USART (UE=0) 后才能写入此位域。

位 16 **RXINV**：RX 引脚有效电平反向 (RX pin active level inversion)

此位由软件置 1 和清零。

0：RX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲，Gnd = 0/标记）工作

1：对 RX 引脚信号值取反。（V_{DD} = 0/标记，Gnd=1/空闲）。

允许在 RX 线路上使用外部反相器。

只有在禁止 USART (UE=0) 后才能写入此位域。

位 15 **SWAP**：交换 TX/RX 引脚 (Swap TX/RX pins)

此位由软件置 1 和清零。

0：按标准引脚排列定义使用 TX/RX 引脚

1：交换 TX 和 RX 引脚功能。允许在与另一个 USART 的交叉连接时工作。

只有在禁止 USART (UE=0) 后才能写入此位域。

位 14 **LINEN**：LIN 模式使能 (LIN mode enable)

此位由软件置 1 和清零。

0：禁止 LIN 模式

1：使能 LIN 模式

LIN 模式可以使用 USARTx_RQR 寄存器中的 SBKRQ 位发送 LIN 同步断路（13 个低位），并可检测 LIN 同步断路。

只有在禁止 USART (UE=0) 后才能写入此位域。

注：　如果 USART 不支持 LIN 模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 13:12 **STOP[1:0]**：停止位 (STOP bits)

这些位用于编程停止位。

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

只有在禁止 USART (UE=0) 后才能写入此位域。

位 11 **CLKEN**：时钟使能 (Clock enable)

该位允许用户使能 CK 引脚。

0：禁止 CK 引脚

1：使能 CK 引脚

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果既不支持同步模式，也不支持智能卡模式，则该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

注：　为了向智能卡正确提供 CK 时钟，必须按以下步骤操作：

- UE = 0

- SCEN = 1

- GTPR 配置

- CLKEN= 1

- UE = 1

位 10 **CPOL**：时钟极性 (Clock polarity)

该位允许用户在同步模式下选择 CK 引脚上时钟输出的极性。它与 CPHA 位结合使用可获得所需的时钟/数据关系

0：空闲时 CK 引脚为低电平

1：空闲时 CK 引脚为高电平

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持同步模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 9 **CPHA**：时钟相位 (Clock phase)

此位用于在同步模式下选择 CK 引脚上时钟输出的相位。它与 CPOL 位结合使用可获得所需的时钟/数据关系（请参见图 247 和图 248）。

0：从第一个时钟边沿开始采样数据

1：从第二个时钟边沿开始采样数据

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持同步模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 8 **LBCL**：最后一个位时钟脉冲 (Last bit clock pulse)

此位用于在同步模式下选择与发送的最后一个数据位 (MSB) 关联的时钟脉冲是否必须在 CK 引脚上输出。

0：最后一个数据位的时钟脉冲不在 CK 引脚上输出

1：最后一个数据位的时钟脉冲在 CK 引脚上输出

注意：　最后一位为发送的第 7 个、第 8 个或第 9 个数据位，具体取决于 USARTx_CR1 寄存器中 M 位所选择的 7位、8 位或 9 位格式。

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持同步模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 7 保留，必须保持复位值。

位 6 **LBDIE**：LIN 断路检测中断使能 (LIN break detection interrupt enable)

断路中断屏蔽（使用断路分隔符进行断路检测）

0：禁止中断

1：当 USARTx_ISR 寄存器中 LBDF = 1 时，生成中断

注：　如果 LIN 不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 5 **LBDL**：LIN 断路检测长度 (LIN break detection length)

该位用于选择 11 位断路检测或 10 位断路检测。

0：10 位断路检测

1：11 位断路检测

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果 LIN 不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 4 **ADDM7**：7 位地址检测/4 位地址检测 (7-bit Address Detection/4-bit Address Detection)

此位用于选择 4 位地址检测或 7 位地址检测。

0：4 位地址检测

1：7 位地址检测（在 8 位数据模式下）

只有在禁止 USART (UE=0) 时才能写入该位

注：　在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。

位 3:0 保留，必须保持复位值。

注：　使能发送器时不应对这 3 个位（CPOL、CPHA、LBCL）进行写操作。

29.8.3 控制寄存器 3 (USARTx_CR3)

Control register 3

偏移地址：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS		SCARCNT2:0]			Res.
									rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	V	V	rw	rw	rw	rw

位 31:24 保留，必须保持复位值。

位 23 保留，必须保持复位值。

位 22 **WUFIE**：从停止模式唤醒中断使能 (Wakeup from Stop mode interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 USARTx_ISR 寄存器中的 WUF=1 时，生成 USART 中断

注：　WUFIE 必须在进入停止模式前置 1。

WUF 中断仅在停止模式下有效。

如果 USART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 21:20 **WUS[1:0]**：从停止模式唤醒中断标志选择 (Wakeup from Stop mode interrupt flag selection)

该位域用于指定激活 WUF（从停止模式唤醒标志）的事件。

00：WUF 在地址匹配时激活（按 ADD[7:0] 和 ADDM7 所定义）

01：保留。

10：WuF 在起始位检测时激活

11：WUF 在 RXNE 时激活。

只有在禁止 USART (UE=0) 后才能写入此位域。

注：　如果 USART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19:17 **SCARCNT[2:0]**：智能卡自动重试计数 (Smartcard auto-retry count)

此位域用于指定智能卡模式下发送和接收的重试次数。

在发送模式下，此位域用于指定生成发送错误（FE 位置 1）前自动重新发送的重试次数。

在接收模式下，此位域用于指定生成接收错误（RXNE 位和 PE 位置 1）前错误接收尝试的次数。

只有在禁止 USART (UE=0) 时才能编程此位域。

使能 USART (UE=1) 时，此位域只能写入 0x0，以停止重新发送。

0x0：禁止重新发送——发送模式下不会自动重新发送。

0x1 到 0x7：自动重新发送的尝试次数（发出错误信号前）

注：　如果不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 16 保留，必须保持复位值。

位 15 **DEP**：驱动器使能极性选择 (Driver enable polarity selection)

0：DE 信号高电平有效。

1：DE 信号低电平有效。

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持驱动器使能功能，该位保留并且必须保持清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 14 **DEM**：驱动器使能模式 (Driver enable mode)

此位用于通过 DE 信号激活外部收发器控制。

0：禁止 DE 功能。

1：使能 DE 功能。DE 信号在 RTS 引脚上输出。

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持驱动器使能功能，该位保留并且必须保持清零。第 700 页上的第 29.4 节：USART 实现.

位 13 **DDRE**：接收出错时的 DMA 禁止 (DMA Disable on Reception Error)

0：接收出错时不禁止 DMA。相应的错误标志置 1，但 RXNE 保持为 0 以防止上溢。因此，将不使能 DMA 请求，从而不会传送错误数据（无 DMA 请求），但会传送接收到的下一个正确数据（用于智能卡模式）。

1：接收出错后禁止 DMA。相应的错误标志以及 RXNE 均置 1。屏蔽 DMA 请求，直到错误标志清零。这意味着软件必须首先禁止 DMA 请求 (DMAR = 0) 或者将 RXNE 清零，然后再将错误标志清零。

只有在禁止 USART (UE=0) 时才能写入此位。

注：　接收错误包括：奇偶校验错误、帧错误或噪声错误。

位 12 **OVRDIS**：上溢禁止 (Overrun Disable)

此位用于禁止接收上溢检测。

0：接收新数据前未读取已接收的数据时，上溢错误标志 ORE 置 1。

1：禁止上溢功能。如果在 RXNE 标志仍置 1 时接收到新数据，则 ORE 标志不会置 1，且新接收的数据会覆盖 USARTx_RDR 寄存器之前的内容。

只有在禁止 USART (UE=0) 时才能写入此位。

注： 此控制位用于检查通信流而不会读取数据。

位 11 **ONEBIT**：一个采样位方法使能 (One sample bit method enable)

该位允许用户选择采样方法。选择一个采样位方法后，将禁止噪声检测标志 (NF)。

0：三个采样位方法

1：一个采样位方法

只有在禁止 USART (UE=0) 时才能写入此位。

注： ONEBIT 功能仅适用于数据位，不适用于起始位。

位 10 **CTSIE**：CTS 中断使能 (CTS interrupt enable)

0：禁止中断

1：当 USARTx_ISR 寄存器中的 CTSIF=1 时，生成中断

注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 9 **CTSE**：CTS 使能 (CTS enable)

0：禁止 CTS 硬件流控制

1：使能 CTS 模式，仅当 CTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 CTS 输入无效，会在停止之前完成发送。如果使 CTS 无效时数据已写入数据寄存器，则将延迟发送，直到 CTS 有效。

只有在禁止 USART (UE=0) 时才能写入该位

注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 8 **RTSE**：RTS 使能 (RTS enable)

0：禁止 RTS 硬件流控制

1：使能 RTS 输出，仅当接收缓冲区中有空间时才会请求数据。发送完当前字符后应停止发送数据。可以接收数据时使 RTS 输出有效（拉至 0）。

只有在禁止 USART (UE=0) 时才能写入此位。

注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 7 **DMAT**：DMA 使能发送器 (DMA enable transmitter)

此位由软件置 1/复位。

1：针对发送使能 DMA 模式

0：针对发送禁止 DMA 模式

位 6 **DMAR**：DMA 使能接收器 (DMA enable receiver)

此位由软件置 1/复位。

1：针对接收使能 DMA 模式

0：针对接收禁止 DMA 模式

位 5 **SCEN**：智能卡模式使能 (Smartcard mode enable)

该位用于使能智能卡模式。

0：禁止智能卡模式

1：使能智能卡模式

只有在禁止 USART (UE=0) 后才能写入此位域。

注：　如果 USART 不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 4 **NACK**：智能卡 NACK 使能 (Smartcard NACK enable)

0：出现奇偶校验错误时禁止 NACK 发送

1：出现奇偶校验错误时使能 NACK 发送

只有在禁止 USART (UE=0) 后才能写入此位域。

注：　如果 USART 不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 3 **HDSEL**：半双工选择 (Half-duplex selection)

选择单线半双工模式

0：未选择半双工模式

1：选择半双工模式

只有在禁止 USART (UE=0) 时才能写入此位。

位 2 **IRLP**：IrDA 低功耗模式 (IrDA low-power)

该位用于选择正常模式和低功耗 IrDA 模式

0：正常模式

1：低功耗模式

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持 IrDA 模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 1 **IREN**：IrDA 模式使能 (IrDA mode enable)

此位由软件置 1 和清零。

0：禁止 IrDA

1：使能 IrDA

只有在禁止 USART (UE=0) 时才能写入此位。

注：　如果不支持 IrDA 模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 0 **EIE**：错误中断使能 (Error interrupt enable)

如果发生帧错误、上溢错误或出现噪声标志（USARTx_ISR 寄存器中 FE = 1 或 ORE = 1 或 NF = 1），则需要使用错误中断使能位来使能中断生成。

0：禁止中断

USARTx_ISR 寄存器中的 FE=1 或 ORE=1 或 NF=1 时生成中断。

29.8.4 波特率寄存器 (USARTx_BRR)

Baud rate register

只有在禁止 USART (UE=0) 时才能写入此寄存器。在自动波特率检测模式下，此位由硬件自动更新。

偏移地址：0x0C

复位值：0x0000

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

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

位 31:16 保留，必须保持复位值。

位 15:4 **BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

位 3:0 **BRR[3:0]**

当 OVER8 = 0 时，BRR[3:0] = USARTDIV[3:0]。
当 OVER8 = 1 时：
BRR[2:0] = USARTDIV[3:0]，右移 1 位。
BRR[3] 必须保持清零。

29.8.5 保护时间和预分频器寄存器 (USARTx_GTPR)

Guard time and prescaler register

偏移地址：0x10

复位值：0x0000

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

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

位 31:16 保留，必须保持复位值

位 15:8 **GT[7:0]**：保护时间值 (Guard time value)

此位域用于编程保护时间值（以波特时钟周期数为单位）。

该位用于智能卡模式。经过此保护时间后，发送完成标志置 1。

只有在禁止 USART (UE=0) 后才能写入此位域。

注： 如果不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 7:0 **PSC[7:0]**：预分频器值 (Prescaler value)

在 IrDA 低功耗和正常的 IrDA 模式下：

PSC[7:0] = IrDA 正常和低功耗波特率

用于编程预分频器，进行 USART 源时钟分频以获得低功耗频率：

使用寄存器中给出的值（8 个有效位）对源时钟进行分频：

00000000：保留 - 不编程此值

00000001：源时钟 1 分频

00000010：源时钟 2 分频

...

在智能卡模式下：

PSC[4:0]：预分频器值 (Prescaler value)

用于编程预分频器，进行 USART 源时钟分频以提供智能卡时钟。

将寄存器中给出的值（5 个有效位）乘以 2 得出源时钟频率的分频系数：

00000：保留 - 不编程此值

00001：源时钟 2 分频

00010：源时钟 4 分频

00011：源时钟 6 分频

...

只有在禁止 USART (UE=0) 后才能写入此位域。

注： 如果使用智能卡模式，则位 [7:5] 必须保持清零。

不支持智能卡和 IrDA 模式时，该位域保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

29.8.6 接收器超时寄存器 (USARTx_RTOR)

Receiver timeout register

偏移地址：0x14

复位值：0x0000

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

位 31:24 **BLEN[7:0]**：块长度 (Block Length)

此位域用于提供智能卡 T=1 接收下的块长度。其值等于信息字符数 + 结尾字段的长度 (1-LEC/2-CRC) - 1。

例如：

BLEN = 0 -> 0 个信息字符 + LEC

BLEN = 1 -> 0 个信息字符 + CRC

BLEN = 255 -> 254 个信息字符 + CRC（总共 256 个字符）

在智能卡模式下，块长度计数器在 TXE=0 时复位。

此位域也可用于其它模式。这种情况下，块长度计数器在 RE=0（禁止接收器）和/或 EOBCF 位写入 1 时复位。

注： 块接收开始后可编程此值（使用起始字段中 LEN 字符中的数据）。每个接收到的块只能对此值编程一次。

位 23:0 **RTO[23:0]**：接收器超时值 (Receiver timeout value)

此位域用于提供接收器的超时值（以位持续时间数为单位）。

在标准模式下，如果在接收到最后一个字符后，在 RTO 值对应的时间内未检测到新的起始位，则 RTOF 标志置 1。

在智能卡模式下，此值用于实施 CWT 和 BWT。有关更多详细信息，请参见智能卡章节。

这种情况下，从接收到的最后一个字符的起始位开始执行超时测量。

注： 每个接收到的字符只能对此值编程一次。

注： 可以实时写入 RTOR。如果新值小于或等于计数器的值，RTOF 标志置 1。

如果不支持接收器超时功能，此寄存器保留并由硬件强制为“0x00000000”。请参见第 700 页上的第 29.4 节：USART 实现。

29.8.7 请求寄存器 (USARTx_RQR)

Request register

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

位 31:5 保留，必须保持复位值

位 4 **TXFRQ**：发送数据刷新请求 (Transmit data flush request)

向该位写入 1 时会将 TXE 标志置 1。

这可丢弃发送数据。由于错误 (NACK) 而未发送数据且 USARTx_ISR 寄存器中的 FE 标志有效时，只能在智能卡模式下使用此位。

如果 USART 不支持智能卡模式，该位保留并由硬件强制清零。

请参见*第 700 页上的第 29.4 节：USART 实现*。

位 3 **RXFRQ**：接收数据刷新请求 (Receive data flush request)

向该位写入 1 时会将 RXNE 标志清零。

这可以丢弃数据而不对其执行读取操作，并避免发生上溢。

位 2 **MMRQ**：静默模式请求 (Mute mode request)

向此位写入 1 可将 USART 置于静默模式，并将 RWU 标志置 1。

位 1 **SBKRQ**：发送断路请求 (Send break request)

向此位写入 1 可将 SBKF 标志置 1 并在发送设备可用后立即请求在线路上发送 BREAK。

注：这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

位 0 **ABRRQ**：自动波特率请求 (Auto baud rate request)

向此位写入 1 可复位 USARTx_ISR 寄存器中的 ABRF 标志，并请求对下一个接收到的数据帧进行自动波特率测量。

注：如果 USART 不支持自动波特率功能，则此位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

29.8.8 中断和状态寄存器 (USARTx_ISR)

Interrupt and status register

偏移地址：0x1C

复位值：0x0200 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:25 保留，必须保持复位值。

位 24:23 保留，必须保持复位值。

位 22 **REACK**：接收使能确认标志 (Receive enable acknowledge flag)

USART 采用接收使能值时，通过硬件将此位置 1/复位。

当支持从停止模式唤醒时，REACK 标志可用于验证进入停止模式前 USART 是否准备好接收数据。

位 21 **TEACK**：发送使能确认标志 (Transmit enable acknowledge flag)

USART 采用发送使能值时，通过硬件将此位置 1/复位。

通过写入 TE=0 生成空闲帧请求，然后在 USARTx_CR1 寄存器中写入 TE=1 以遵循 TE=0 最短周期时，可使用此位。

位 20 **WUF**：从停止模式唤醒标志 (Wakeup from Stop mode flag

当检测到唤醒事件时，此位由硬件置 1。事件通过 WUS 位域定义。通过向 USARTx_ICR 寄存器中的 WUCF 写入 1，此位由软件清零。

如果 USARTx_CR3 寄存器中 WUFIE=1，则会生成中断。

注：当 UESM 清零时，WUF 标志也清零。

WUF 中断仅在停止模式下有效。

如果 USART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19 **RWU**：*接收器从静默模式唤醒 (Receiver wakeup from Mute mode)*

该位指示 USART 是否处于静默模式。当识别出唤醒/静默序列时，此位由硬件清零/置 1。静默模式控制序列（地址或 IDLE）通过 USARTx_CR1 寄存器中的 WAKE 位选择。

当选择 IDLE 模式下唤醒时，该位只能通过用软件向 USARTx_RQR 寄存器中的 MMRQ 位写 1 的方式置 1。

0：接收器处于活动模式

1：接收器处于静默模式

位 18 **SBKF**：发送断路标志 (Send break flag)

此位指示已请求发送断路字符。通过将 1 写入 USARTx_RQR 寄存器中的 SBKRQ 位，此位由软件置 1。此位在断路发送的停止位期间由硬件自动复位。

0：不发送断路字符

1：将发送断路字符

位 17 **CMF**：字符匹配标志 (Character match flag)

接收到由 ADD[7:0] 定义的字符后由硬件将此位置 1。通过向 USARTx_ICR 寄存器中的 CMCF 写入 1，此位由软件清零。

如果 USARTx_CR1 寄存器中 CMIE=1，则会生成中断。

0：未检测到字符匹配

1：检测到字符匹配

位 16 **BUSY**：忙标志 (Busy flag)

此位由硬件置 1 和复位。当 RX 线路上正在进行通信（成功检测到起始位）时有效。在接收结束（成功或失败）时复位。

0：USART 处于空闲状态（无接收）

1：正在接收

位 15 **ABRF**：自动波特率标志 (Auto baud rate flag)

已设置自动波特率（RXNE 也将置 1，并在 RXNEIE = 1 时生成中断），或者自动波特率操作未成功完成时，此位由硬件置 1 (ABRE=1)（此时，ABRE、RXNE 和 FE 也置 1）

为请求新的自动波特率检测，通过向 USARTx_RQR 寄存器中的 ABRRQ 写入 1，此位由软件清零。

注：　如果 USART 不支持自动波特率功能，则此位保留并由硬件强制清零。

位 14 **ABRE**：自动波特率错误 (Auto baud rate error)

如果波特率测量失败（波特率超出范围或字符比较失败），此位由硬件置 1。

通过将 1 写入 USARTx_CR3 寄存器中的 ABRRQ 位，此位由软件清零。

注：　如果 USART 不支持自动波特率功能，则此位保留并由硬件强制清零。

位 13 保留，必须保持复位值。

位 12 **EOBF**：块结束标志 (End of block flag)

接收到完整块后，此位由硬件置 1（例如 T=1 智能卡模式）。接收到的字节数（从块的起始处，包括起始字段）等于或大于 BLEN + 4 时执行检测。

如果 USARTx_CR2 寄存器中 EOBIE = 1，则会生成中断。

通过向 USARTx_ICR 寄存器中的 EOBCF 写入 1，此位由软件清零。

0：未达到块结束

1：已达到块结束（字符数）

注：　如果不支持智能卡模式，该位保留并由硬件强制清零。请参见第 700 页上的第 29.4 节：USART 实现。

位 11 **RTOF**：接收器超时 (Receiver timeout)

已经过在 RTOR 寄存器中编程的超时值后，如果无任何通信，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 RTOCF 写入 1，此位由软件清零。

如果 USARTx_CR1 寄存器中 RTOIE=1，则会生成中断。

在智能卡模式下，该超时对应于 CWT 或 BWT 时间。

0：未达到超值值

1：已达到超时值，未接收到任何数据

注：　如果 RTOR 寄存器中编程的时间值将 2 个字符隔开，则 RTOF 不置 1。如果此时间大于该值 + 2 个采样时间（2/16 或 2/8，具体取决于过采样方法），则 RTOF 标志置 1。

即使 RE = 0，计数器仍会计数，但 RTOF 仅在 RE = 1 时置 1。如果 RE 置 1 时已经超时，则 RTOF 将置 1。

如果 USART 不支持接收器超时功能，该位保留并由硬件强制清零。

位 10 **CTS**：CTS 标志 (CTS flag)

此位由硬件置 1/复位。此位是对 CTS 输入引脚的状态取反。

0：CTS 线置 1

1：CTS 线复位

注：　如果不支持硬件流控制功能，该位保留并由硬件强制清零。

位 9 **CTSIF**：CTS 中断标志 (CTS interrupt flag)

如果 CTSE 位置 1，当 CTS 输入切换时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 CTSCF 位写入 1，此位由软件清零。

如果 USARTx_CR3 寄存器中 CTSIE=1，则会生成中断。

0：CTS 状态线上未发生变化

1：CTS 状态线上发生变化

注：　如果不支持硬件流控制功能，该位保留并由硬件强制清零。

位 8 **LBDF**：LIN 断路检测标志 (LIN break detection flag)

检测到 LIN 断路时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 LBDCF 写入 1，此位由软件清零。

如果 USARTx_CR2 寄存器中 LBDIE = 1，则会生成中断。

0：未检测到 LIN 断路

1：检测到 LIN 断路

注：　如果 USART 不支持 LIN 模式，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 7 **TXE**：发送数据寄存器为空 (Transmit data register empty)

当 USARTx_TDR 寄存器的内容已传输到移位寄存器时，此位由硬件置 1。通过对 USARTx_TDR 寄存器执行写入操作将该位清零。

为丢弃数据（仅在智能卡 T=0 模式下出现发送故障时），也可以通过向 USARTx_RQR 寄存器中的 TXFRQ 写入 1 来将 TXE 标志清零。

如果 USARTx_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。

0：数据未传输到移位寄存器

1：数据传输到移位寄存器

注：　单缓冲区发送期间使用该位。

位 6 **TC**：发送完成 (Transmission complete)

如果已完成对包含数据的帧的发送并且 TXE 置 1，则此位由硬件置 1。如果 USARTx_CR1 寄存器中 TCIE=1，则会生成中断。通过向 USARTx_ICR 寄存器中的 TCCF 写入 1 或向 USARTx_TDR 寄存器执行写操作，此位由软件清零。

如果 USARTx_CR1 寄存器中 TCIE=1，则会生成中断。

0：传送未完成

1：传送已完成

注：　如果 TE 位复位且无任何发送正在进行，TC 位会立即置 1。

位 5 **RXNE**：读取数据寄存器不为空 (Read data register not empty)

当 RDR 移位寄存器的内容已传输到 USARTx_RDR 寄存器时，此位由硬件置 1。通过对 USARTx_RDR 寄存器执行读入操作将该位清零。也可以通过将 USARTx_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。

如果 USARTx_CR1 寄存器中 RXNEIE=1，则会生成中断。

0：未接收到数据

1：已准备好读取接收到的数据

位 4 **IDLE**：检测到空闲线路 (IDLE line detected)

检测到空闲线路时，此位由硬件置 1。如果 USARTx_CR1 寄存器中 IDLEIE=1，则会生成中断。通过向 USARTx_ICR 寄存器中的 IDLECF 写入 1，此位由软件清零。

0：未检测到空闲线路

1：检测到空闲线路

注：　直到 RXNE 位已置 1 时（即，当出现新的空闲线路时）IDLE 位才会被再次置 1。
使能静默模式 (MME=1) 后，如果 USART 未静默 (RWU=0)，则 IDLE 置 1，无论是否通过 WAKE 位选择了静默模式。如果 RWU=1，IDLE 不置 1。

位 3 **ORE**：溢出错误 (Overrun error)

在 RXNE = 1 的情况下，当移位寄存器中当前正在接收的数据准备好传输到 RDR 寄存器时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 ORECF 写入 1，此位由软件清零。

如果 USARTx_CR1 寄存器中 RXNEIE=1 或 EIE = 1，则会生成中断。

0：无溢出错误

1：检测到溢出错误

注：当此位置 1 时，RDR 寄存器的内容不会丢失，但移位寄存器会被覆盖。EIE 位置 1 后，如果在多缓冲区通信中 ORE 标志置 1，则会生成中断。

USARTx_CR3 寄存器中的 OVRDIS 位置 1 时，此位将被永久强制清零（无上溢检测）。

位 2 **NF**：START 位噪声检测标志 (START bit Noise detection flag)

当在接收的帧上检测到噪声时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 NFCF 写入 1，此位由软件清零。

0：未检测到噪声

1：检测到噪声

注：该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNE 位出现的时间相同。EIE 位置 1 后，如果在多缓冲区通信中 NF 标志置 1，则会生成中断。

注：当线路无噪声时，可以通过将 ONEBIT 位编程为 1 提高 USART 对偏差的容差来禁止 NF 标志（请参见第 714 页上的第 29.5.5 节：USART 接收器对时钟偏差的容差）。

位 1 **FE**：帧错误 (Framing error)

当检测到去同步化、过度的噪声或中断字符时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 FECF 写入 1，此位由软件清零。

在智能卡模式下发送期间，如果在达到最大发送尝试次数后仍未成功（智能卡向数据帧发送 NACK 信号），则此位置 1。

如果 USARTx_CR1 寄存器中 EIE = 1，则会生成中断。

0：未检测到帧错误

1：检测到帧错误或中断字符

位 0 **PE**：奇偶校验错误 (Parity error)

当在接收器模式下发生奇偶校验错误时，此位由硬件置 1。通过向 USARTx_ICR 寄存器中的 PECF 写入 1，此位由软件清零。

如果 USARTx_CR1 寄存器中 PEIE = 1，则会生成中断。

0：无奇偶校验错误

1：奇偶校验错误

29.8.9 中断标志清零寄存器 (USARTx_ICR)

Interrupt flag clear register

偏移地址：0x20

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											rc_w1			rc_w1	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
			rc_w1	rc_w1		rc_w1	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位 31:21 保留，必须保持复位值。

位 20 **WUCF**：从停止模式唤醒清零标志 (Wakeup from Stop mode clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 WUF 标志将清零。
注： 如果 USART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19:18 保留，必须保持复位值。

位 17 **CMCF**：字符匹配清零标志 (Character match clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 CMF 标志将清零。

位 16:13 保留，必须保持复位值。

位 12 **EOBCF**：块结束清零标志 (End of block clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 EOBF 标志将清零。
注： 如果 USART 不支持智能卡模式，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 11 **RTOCF**：接收器超时清零标志 (Receiver timeout clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 RTOF 标志将清零。
注： 如果 USART 不支持接收器超时功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 10 保留，必须保持复位值。

位 9 **CTSCF**：CTS 清零标志 (CTS clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 CTSIF 标志将清零。
注： 如果不支持硬件流控制功能，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 8 **LBDCF**：LIN 中断检测清零标志 (LIN break detection clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 LBDF 标志将清零。
注： 如果 LIN 不支持智能卡模式，该位保留并由硬件强制清零。
请参见第 700 页上的第 29.4 节：USART 实现。

位 7 保留，必须保持复位值。

位 6 **TCCF**：发送完成清零标志 (Transmission complete clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 TC 标志将清零。

位 5 保留，必须保持复位值。

位 4 **IDLECF**：检测到空闲线路清零标志 (Idle line detected clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 IDLE 标志将清零。

位 3 **ORECF**：上溢错误清零标志 (Overrun error clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 ORE 标志将清零。

位 2 **NCF**：检测到噪声清零标志 (Noise detected clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 NF 标志将清零。

位 1 **FECF**：帧错误清零标志 (Framing error clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 FE 标志将清零。

位 0 **PECF**：奇偶校验错误清零标志 (Parity error clear flag)
将 1 写入此位时，USARTx_ISR 寄存器中 PE 标志将清零。

29.8.10 接收数据寄存器 (USARTx_RDR)

Receive data register

偏移地址：0x24

复位值：未定义

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

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

位 31:9 保留，必须保持复位值。

位 8:0 **RDR[8:0]**：接收数据值 (Receive data value)
包含接收到的数据字符。
RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口（请参见*图 235*）。
在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

29.8.11 发送数据寄存器 (USARTx_TDR)

Transmit data register

偏移地址：0x28

复位值：未定义

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

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

位 31:9 保留，必须保持复位值。

位 8:0 **TDR[8:0]**：发送数据值 (Transmit data value)

包含要发送的数据字符。

TDR 寄存器在内部总线和输出移位寄存器之间提供了并行接口（请参见*图 235*）。

在使能奇偶校验位的情况下（USARTx_CR1 寄存器中的 PCE 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。

注： 只能在 TXE=1 时写入此寄存器。

29.8.12 USART 寄存器映射

下表提供了 USART 寄存器映射和复位值。

表 131. USART 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USARTx_CR1**	Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value				0	0	0	0	0	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	**USARTx_CR2**	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0				
0x08	**USARTx_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS		SCARCNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 131. USART 寄存器映射和复位值（续）

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
0x0C	**USARTx_BRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USARTx_GTPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USARTx_RTOR**	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USARTx_RQR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	**USARTx_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	1	0	0	0	0	0	0
0x20	**USARTx_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	EOBCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
	Reset value												0			0					0	0		0	0		0		0	0	0	0	0
0x24	**USARTx_RDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x28	**USARTx_TDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X

有关寄存器边界地址的信息，请参见*第 56 页的第 2.2 节*。

30 低功耗通用异步接收器 (LPUART)

30.1 简介

低功耗通用异步接收器（LPUART）是一种 UART，允许有限功耗下进行全双工 UART 通信。仅需 32.768 kHz LSE 时钟即可进行高达 9600 baud/s 的 UART 通信。当 LPUART 由与 LSE 时钟不同的时钟源计时，可以达到更高的波特率。

即使当微控制器处于停止模式，能耗极低时，LPUART 也会等待 UART 帧的到来。LPUART 包含所有必要的硬件支持，使在最小功耗下可以进行异步串行通信。

它支持半双工单线通信和调制解调器操作 (CTS/RTS)，还支持多处理器通信。

DMA（直接存储器访问）可用于数据发送/接收。

30.2 LPUART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 使用 32.768 kHz 时钟源时波特率可编程为 300 波特/s 到 9600 波特/s。使用高频时钟源可实现更高的波特率
- 双时钟域允许：
  - UART 功能和从停止模式唤醒
  - 方便的波特率编程，独立于 PCLK 重新编程
- 数据字长度可编程（7 位、8 位或 9 位）
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 停止位可配置（支持 1 个或 2 个停止位）
- 单线半双工通信
- 使用 DMA 实现连续通信
- 使用中央 DMA 在预留的 SRAM 缓冲区中收/发字节。
- 发射器和接收器有单独的使能位
- 发送和接收的单独信号极性控制
- Tx/Rx 引脚配置可交换
- 调制解调器和 RS-485 收发器的硬件流控制
- 传输检测标志：
  - 接收缓冲区已满
  - 发送缓冲区为空 (Transmit buffer empty)
  - BUSY 标志和发送结束标志
- 奇偶校验控制：
  - 发送奇偶校验位
  - 检查接收的数据字节的奇偶性

- 四个错误检测标志：
  - 上溢错误
  - 噪声检测
  - 帧错误
  - 奇偶校验错误
- 十四个具有标志位的中断源
- 多处理器通信

  如果地址不匹配，LPUART 将进入静默模式。
- 从静默模式唤醒（通过空闲线检测或地址标记检测）

30.3 LPUART 特性实现

STM32L0x3 器件内置一个 LPUART。有关 LPUART 支持的特性，请参见*第 29.4 节：USART 实现*。

30.4 LPUART 功能说明

任何 LPUART 双向通信均需要至少两个引脚：接收数据输入引脚 (RX) 和发送数据输出引脚 (TX)：

- **RX**：接收数据输入引脚。

  串行数据输入引脚。
- **TX**：发送数据输出引脚。

  如果关闭发送器，该输出引脚模式由其 I/O 端口配置决定。如果使能了发送器但没有待发送的数据，则 TX 引脚处于高电平。在单线模式下，该 I/O 用于发送和接收数据。

正常 LPUART 模式下，通过这些引脚以帧的形式发送和接收串行数据：

- 发送或接收前保持空闲线路
- 起始位
- 数据字（7 位、8 位或 9 位），最低有效位在前
- 用于指示帧传输已完成的 1 个、2 个停止位
- LPUART 接口使用波特率发生器
- 状态寄存器 (LPUART_ISR)
- 接收和发送数据寄存器（LPUART_RDR、LPUART_TDR）
- 波特率寄存器 (LPUART_BRR)

有关各个位的定义，请参见*第 30.7 节：LPUART 寄存器*。

在 RS232 硬件流控制模式下需要以下引脚：

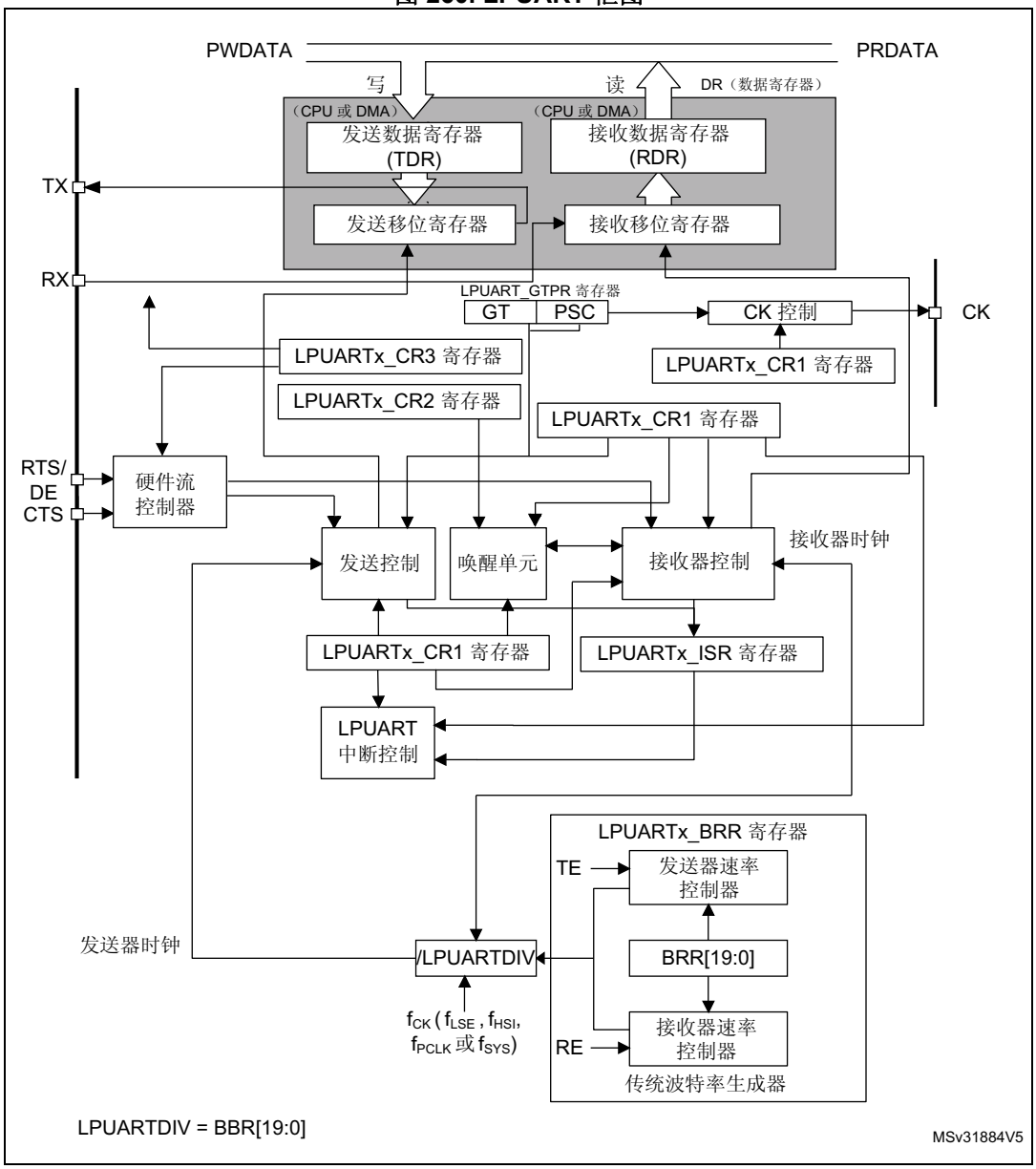
- **CTS**：“清除以发送”用于在当前传输结束时阻止数据发送（高电平时）
- **RTS**：“请求以发送”用于指示 LPUART 已准备好接收数据（低电平时）。

在 RS485 硬件控制模式下需要以下引脚：

- **DE**：“驱动器使能”用于激活外部收发器的发送模式。

注： *DE 和 RTS 共用同一个引脚。*

图 260. LPUART 框图



30.4.1 LPUART 字符说明

可通过对 LPUART_CR1 寄存器中的 M[1:0] 位进行编程来选择 7 位、8 位或 9 位的字长（请参见*图 261*）。

- 7 位字符长度：M[1:0] = 10
- 8 位字符长度：M[1:0] = 00
- 9 位字符长度：M[1:0] = 01

在默认情况下，信号（TX 或 RX）在起始位工作期间处于低电平状态。在停止位工作期间处于高电平状态。

通过极性配置控制，可以单独针对每个信号对这些值取反。

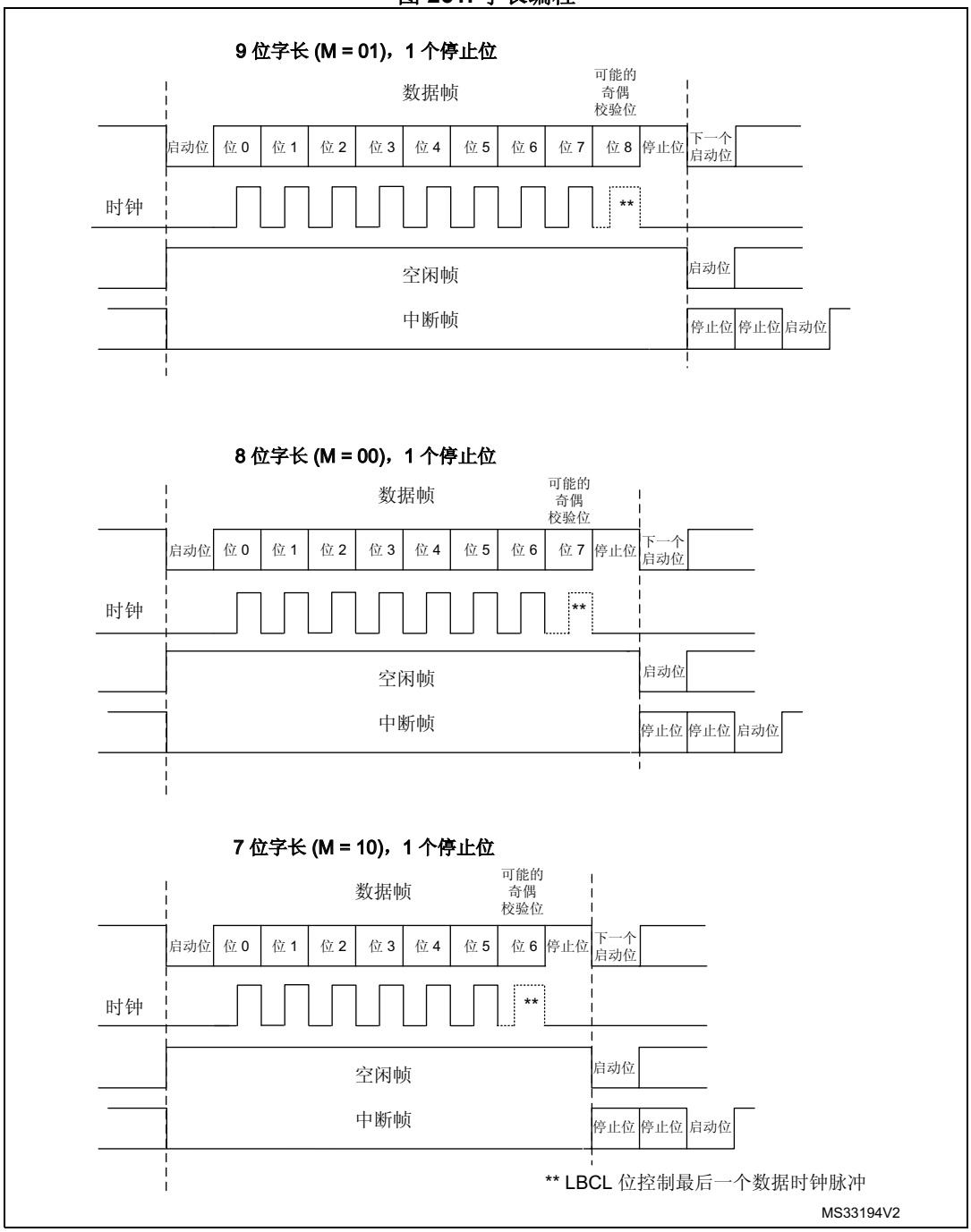
*空闲字符*可理解为整个帧周期内电平均为“1”。（停止位的电平也是“1”）。

*停止字符*可理解为在一个帧周期内接收到的电平均为“0”。发送器在中断帧的末尾插入 2 个停止位。

发送和接收由通用波特率发生器驱动，发送器和接收器的使能位分别置 1 时将生成相应的发送时钟和接收时钟。

下面给出了各个块的详细信息。

图 261. 字长编程


9 位字长 (M = 01)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
位 7
位 8
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
8 位字长 (M = 00)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
位 7
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
7 位字长 (M = 10)，1 个停止位
数据帧
可能的
奇偶
校验位
启动位
位 0
位 1
位 2
位 3
位 4
位 5
位 6
停止位
下一个
启动位
时钟
**
空闲帧
启动位
中断帧
停止位
停止位
启动位
** LBCL 位控制最后一个数据时钟脉冲
MS33194V2

30.4.2 LPUART 发送器

发送器可发送 7 位、8 位或 9 位的数据字，具体取决于 M 位的状态。要激活发送器功能，必须将发送使能位 (TE) 置 1。发送移位寄存器中的数据输出到 TX 引脚。

字符发送

LPUART 发送期间，首先通过 TX 引脚移出数据的最低有效位（默认配置）。该模式下，LPUART_TDR 寄存器的缓冲区 (TDR) 位于内部总线和发送移位寄存器之间（请参见*图 235*）。

每个字符前面都有一个起始位，其逻辑电平在一个位周期内为低电平。字符由可配置数量的停止位终止。

LPUART 支持以下停止位：1 个和 2 个停止位。

注： *向 LPUART_TDR 中写入要发送的数据前，TE 位必须先置 1。*

数据发送期间不应复位 TE 位。发送期间复位 TE 位会冻结波特率计数器，从而将损坏 TX 引脚上的数据。当前传输的数据将会丢失。

使能 TE 位后，将会发送空闲帧。

可配置的停止位

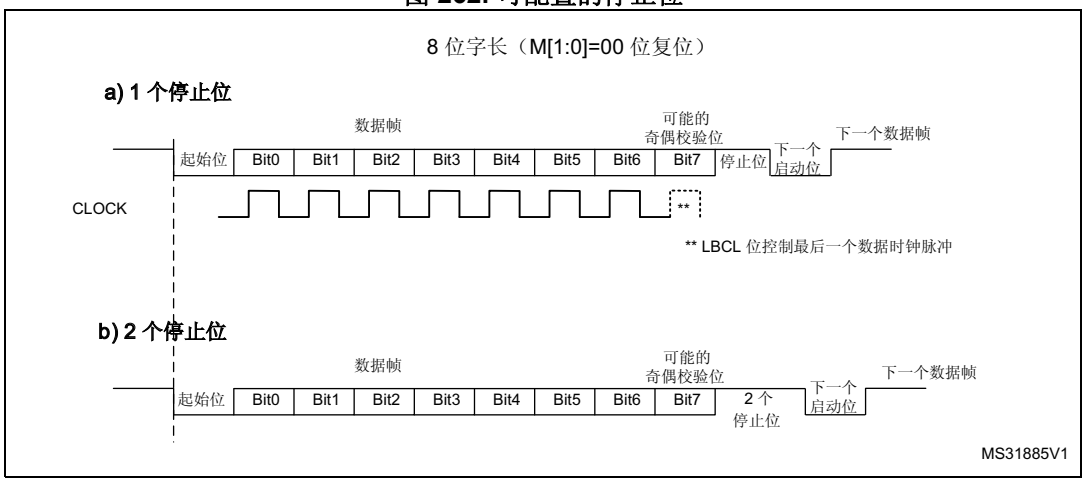
可以在控制寄存器 2 的位 13 和位 12 中编程将随各个字符发送的停止位的数量。

- ***1 个停止位***：这是停止位数量的默认值。
- ***2 个停止位***：正常 LPUART 模式、单线模式和调制解调器模式支持该值。

空闲帧发送将包括停止位。

中断发送是 10 个低电平位（M[1:0] = 00 时）、11 个低电平位（M[1:0] = 01 时）或 9 个低电平位（M[1:0] = 10 时），然后是 2 个停止位。无法传送长中断（中断长度大于 9/10/11 个低电平位）。

图 262. 可配置的停止位



字符发送步骤

1. 对 LPUART_CR1 中的 M 位进行编程以定义字长。
2. 使用 LPUART_BRR 寄存器选择所需波特率。
3. 对 LPUART_CR2 中的停止位数量进行编程。
4. 通过向 LPUART_CR1 寄存器中的 UE 位写入 1 使能 LPUART。
5. 如果将进行多缓冲区通信，请选择 LPUART_CR3 中的 DMA 使能 (DMAT)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 LPUART_CR1 中的 TE 位置 1 以便在首次发送时发送一个空闲帧。
7. 在 LPUART_TDR 寄存器中写入要发送的数据（该操作将清零 TXE 位）。为每个要在单缓冲区模式下发送的数据重复这一步骤。
8. 向 LPUART_TDR 寄存器写入最后一个数据后，等待至 TC=1。这表明最后一个帧的传送已完成。禁止 LPUART 或进入暂停模式时需要此步骤，以避免损坏最后一次发送。

单字节通信

始终通过向发送数据寄存器写入数据将 TXE 位清零。

TXE 位由硬件置 1，它表示：

- 数据已从 LPUART_TDR 寄存器移到移位寄存器中且数据发送已开始。
- LPUART_TDR 寄存器为空。
- LPUART_TDR 寄存器中可写入下一个数据，而不会覆盖前一个数据。

TXEIE 位置 1 时该标志位会生成中断。

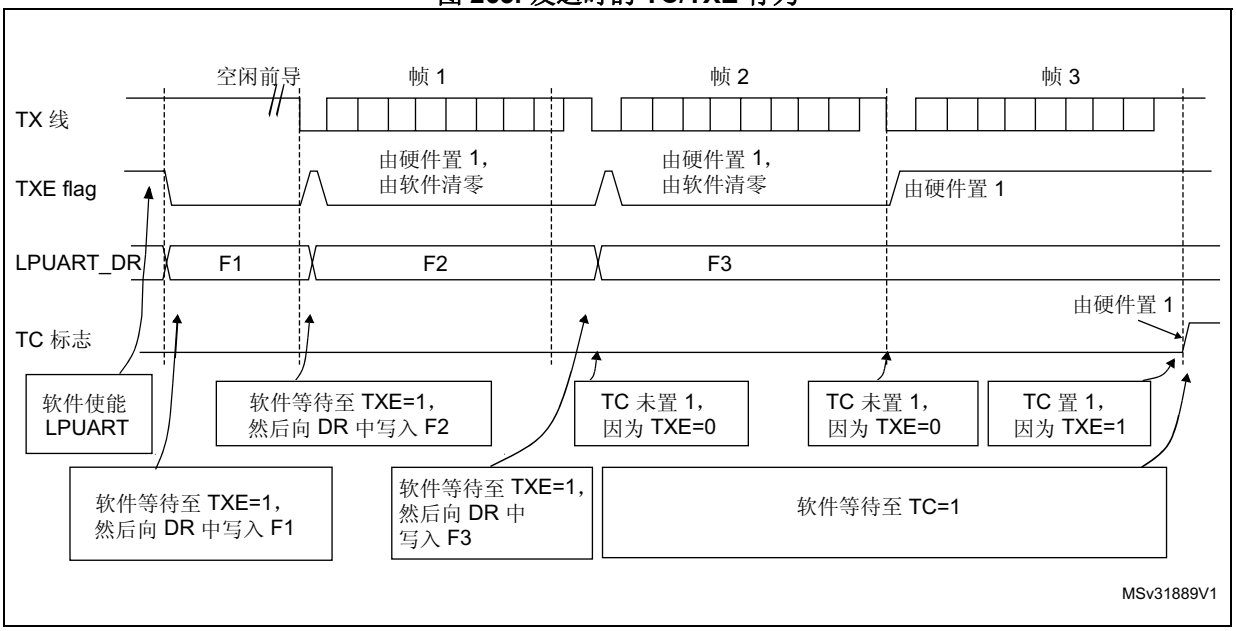
发送时，要传入 LPUART_TDR 寄存器的写指令中存有 TDR 寄存器中的数据，接下来，该数据将在当前进行的发送结束时复制到移位寄存器中。

未发送时，要传入 LPUART_TDR 寄存器的写指令将数据置于移位寄存器中，数据发送开始时，TXE 位置 1。

如果帧已发送（停止位后）且 TXE 位置 1，TC 位将变为高电平。如果 LPUART_CR1 寄存器中的 TCIE 位置 1，将生成中断。

向 LPUART_TDR 寄存器中写入最后一个数据后，必须等待至 TC=1，之后才可禁止 LPUART 或使微控制器进入低功率模式（请参见*图 238：发送时的 TC/TXE 行为*）。

图 263. 发送时的 TC/TXE 行为



中断字符

将 SBKRQ 位置 1 将发送一个中断字符。中断帧的长度取决于 M 位（请参见*图 261*）。

如果将“1”写入 SBKRQ 位，则当前字符发送完成后，将在 TX 线路上发送一个中断字符。通过写操作将 SBKF 位置 1 并在中断字符发送完成时（发送中断字符后的停止位期间），此位由硬件复位。LPUART 在中断帧末尾的两位持续时间内插入一个逻辑“1”信号 (STOP)，以确保识别下个帧的起始位。

这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

空闲字符

将 TE 位置 1 会驱动 LPUART 在第一个数据帧之前发送一个空闲帧。

30.4.3 LPUART 接收器

LPUART 可接收 7 位、8 位或 9 位的数据字，具体取决于 LPUART_CR1 寄存器中的 M 位。

起始位检测

在 LPUART 中，进行 START 位检测时，首先应在 Rx 线路上检测到下降沿，然后在起始位中间采样以确认电平是否仍为“0”。如果起始采样为“1”，则噪声错误标志 (NF) 置 1，START 位将被丢弃，接收器将等待新的 START 位。否则，接收器将继续正常采样所有传入位。

字符接收

LPUART 接收期间，首先通过 RX 引脚移入数据的最低有效位（默认配置）。该模式下，LPUART_RDR 寄存器的缓冲区 (RDR) 位于内部总线和接收移位寄存器之间。

字符接收步骤

1. 对 LPUART_CR1 中的 M 位进行编程以定义字长。
2. 使用波特率寄存器 LPUART_BRR 选择所需波特率。
3. 对 LPUART_CR2 中的停止位数量进行编程。
4. 通过向 LPUART_CR1 寄存器中的 UE 位写入 1 使能 LPUART。
5. 如果将进行多缓冲区通信，请选择 LPUART_CR3 中的 DMA 使能 (DMAR)。按照多缓冲区通信中的解释说明配置 DMA 寄存器。
6. 将 RE 位 LPUART_CR1 置 1。这一操作将使能接收器开始搜索起始位。

接收到字符时

- RXNE 位置 1。这表明移位寄存器的内容已传送到 RDR。也就是说，已接收到并可读取数据（及其相应的错误标志）。
- 如果 RXNEIE 位置 1，则会生成中断。
- 如果接收期间已检测到帧错误、噪声错误或上溢错误，错误标志位可置 1。也可使用 RXNE 位将 PE 标志置 1。
- 在多缓冲区模式下，每接收到一个字节后 RXNE 均置 1，然后通过 DMA 对接收数据寄存器执行读操作清零。
- 在单缓冲区模式下，通过软件对 LPUART_RDR 寄存器执行读操作将 RXNE 位清零。也可以通过将 LPUART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。RXNE 位必须在结束接收下一个字符前清零，以避免发生上溢错误。

中断字符

接收到中断字符时，LPUART 将会按照帧错误对其进行处理。

空闲字符

检测到空闲帧时，处理步骤与接收到数据的情况相同；如果 IDLEIE 位置 1，则会产生中断。

上溢错误

如果在 RXNE 未复位时接收到字符，则会发生上溢错误。RXNE 位清零前，数据无法从移位寄存器传送到 RDR 寄存器。

每接收到一个字节后，RXNE 标志位都将置 1。当 RXNE 标志位是 1 时，如果在接收到下一个数据或尚未处理上一个 DMA 请求时，则会发生上溢错误。发生上溢错误时：

- ORE 位置 1。
- RDR 中的内容不会丢失。对 LPUART_RDR 执行读操作时可使用先前的数据。
- 移位寄存器将被覆盖。之后，上溢期间接收到的任何数据都将丢失。
- 如果 RXNEIE 位置 1 或 EIE 位置 1，则会生成中断。
- 通过设置 ICR 寄存器中的 ORECF 位复位 ORE 位。

注： *ORE 位置 1 时表示至少 1 个数据丢失。存在两种可能：*

- 如果 RXNE=1，则最后一个有效数据存储于接收寄存器 RDR 中并且可进行读取；

- 如果 RXNE=0，则表示最后一个有效数据已被读取，因此 RDR 中没有要读取的数据。接收到新（和丢失）数据的同时已读取 RDR 中的最后一个有效数据时，会发生该情况。

选择时钟源

通过复位和时钟控制系统 (RCC) 选择时钟源。时钟源必须在使能 LPUART（通过将 UE 位置 1）前选择。

必须遵循以下两个条件选择时钟源：

- 可在低功耗模式下使用 LPUART
- 通信速度。

时钟源频率为 f_{CK}。

当支持双时钟域和从停止模式唤醒功能时，时钟源可以是以下其中之一：f_{PCLK}（默认值）、f_{LSE}、f_{HSI} 或 f_{SYS}。否则，LPUART 时钟源是 f_{PCLK}。

选择 f_{LSE} 或 f_{HSI} 作为时钟源可允许 LPUART 在 MCU 处于低功耗模式时接收数据。必要时，LPUART 可基于所接收的数据和唤醒模式的选择来唤醒 MCU，以通过用软件读取 LPUART_RDR 寄存器的方式或通过 DMA 的方式传输已接收数据。

对于其它时钟源，系统必须激活才能进行 LPUART 通信。

时钟源还决定通信速度范围（尤其是最大通信速度）。

接收器在尽可能靠近波特周期中间的位置采样每个传入波特。每个传入波特仅进行一次采样。

注： *不进行数据噪声检测。*

帧错误 (Framing error)

如果接收数据时未在预期时间内识别出停止位，从而出现同步失效或过度的噪声，则会检测到帧错误。

检测到帧错误时：

- FE 位由硬件置 1。
- 无效数据从移位寄存器传送到 LPUART_RDR 寄存器。
- 单字节通信时无中断产生。然而，在 RXNE 位产生中断时，该位出现上升沿。多缓冲区通信时，LPUART_CR3 寄存器中的 EIE 位置 1 时将发出中断。

通过将 1 写入 LPUART_ICR 寄存器中的 FECF 位来复位 FE 位。

接收期间可配置的停止位

可通过控制寄存器 2 中的控制位配置要接收的停止位的数量 - 可以是 1 个或 2 个（正常模式下）。

- ***1 个停止位***：将在第 8、第 9 和第 10 次采样时对 1 个停止位进行采样。
- ***2 个停止位***：将在第 2 个停止位的中间对 2 个停止位进行采样。RXNE 和 FE 标志在此采样期间（例如，在第 2 个停止位期间）置 1。发生帧错误时不检测第 1 个停止位。

30.4.4 LPUART 波特率生成

接收器和发送器（Rx 和 Tx）的波特率设置为 LPUART_BRR 寄存器中编程的相同值。

接收器和发送器（Rx 和 Tx）的波特率设置为 LPUART_BRR 寄存器中编程的相同值。

$$\text{Tx/Rx 波特} = \frac{256 \times f_{CK}}{\text{LPUARTDIV}}$$

LPUARTDIV 在 LPUART_BRR 寄存器中编码。

注： *对 LPUART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。*

禁止在 LPUART_BRR 寄存器中写入小于 0x300 的值。

fck 必须在 [3 x 波特率，4096 x 波特率] 范围内。

LPUART 时钟源为 LSE 时可达到的最大波特率为 9600 波特。当 LPUART 由与 LSE 时钟不同的时钟源驱动时，可以达到更高的波特率。例如，如果 USART 时钟源是系统时钟（最大为 32 MHz），可达到的最大波特率为 10 M 波特。

表 132. f_{ck} = 32,768 KHz 时编程的波特率的错误计算

波特率		f_{CK} = 32,768 KHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % = (计算值 - 所需值) / 所需波特率
1	300 Bps	300 Bps	0x6D3A	0
2	600 Bps	600 Bps	0x369D	0
3	1200 Bps	1200.087 Bps	0x1B4E	0.007
4	2400 Bps	2400.17 Bps	0xDA7	0.007
5	4800 Bps	4801.72 Bps	0x6D3	0.035
6	9600 Bps	9608.94 Bps	0x369	0.093

30.4.5 LPUART 接收器对时钟偏差的容差

仅当总时钟系统偏差小于 LPUART 接收器的容差时，LPUART 异步接收器才能正常工作。影响总偏差的因素包括：

- DTRA：发送器误差引起的偏差（其中还包括发送器本地振荡器的偏差）
- DQUANT：接收器的波特率量化引起的误差
- DREC：接收器本地振荡器的偏差
- DTCL：传输线路引起的偏差（通常是由于收发器所引起，它可能会在低电平到高电平转换时序与高电平到低电平转换时序之间引入不对称）

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{LPUART 接收器容差}$$

其中

DWU 是使用从停止模式唤醒时因采样点偏差产生的误差。

M[1:0] = 01 时：

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

M[1:0] = 00 时：

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

M[1:0] = 10 时：

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ 是检测到唤醒事件到时钟（由外设请求）与调压器均就绪的时间。

LPUART 接收器在*表 133* 中指定的最大容许偏差下可正确接收数据：

- 由 LPUARTx_CR1 寄存器中的 M 位定义的 7 位、8 位或 9 位字符长度
- 1 个或 2 个停止位

表 133. BRR[3:0] 不等于 0000 时的 LPUART 接收器容差

M 位	768 ≤ BRR <1024	1024 ≤ BRR < 2048	2048 ≤ BRR < 4096	4096 ≤ BRR
8 位 (M=00)，1 个停止位	1.82%	2.56%	3.90%	4.42%
9 位 (M=01)，1 个停止位	1.69%	2.33%	2.53%	4.14%
7 位 (M=10)，1 个停止位	2.08%	2.86%	4.35%	4.42%
8 位 (M=00)，2 个停止位	2.08%	2.86%	4.35%	4.42%
9 位 (M=01)，2 个停止位	1.82%	2.56%	3.90%	4.42%
7 位 (M=10)，2 个停止位	2.34%	3.23%	4.92%	4.42%

注：当接收的帧恰好包含 10 个（M 位 = 00）、11 个（M 位 = 01）或 9 个（M 位 = 10）位持续时间的空闲帧时，表 133 中指定的数据可能与特例中的数据略微不同。

30.4.6 使用 LPUART 进行多处理器通信

可以通过 LPUART 进行多处理器通信（多个 LPUART 连接在一个网络中）。例如，其中一个 LPUART 可以是主器件，其 TX 输出与其它 LPUART 的 RX 输入相连接。其它 USART 为从 USART，其各自的 TX 输出在逻辑上通过与运算连在一起，并与主 USART 的 RX 输入相连接。

在多处理器配置中，理想情况下通常只有预期的消息接收方主动接收完整的消息内容，从而减少由所有未被寻址的接收器造成的冗余 LPUART 服务开销。

可通过静默功能将未被寻址的器件置于静默模式下。为了使用静默模式功能，必须将 LPUART_CR1 寄存器中的 MME 位置 1。

在静默模式下：

- 不得将接收状态位置 1。
- 禁止任何接收中断。
- LPUART_ISR 寄存器中的 RWU 位置 1。在某些情况下，RWU 可以由硬件或软件通过 LPUART_RQR 寄存器中的 MMRQ 位自动控制。

根据 LPUART_CR1 寄存器中 WAKE 位的设置，LPUART 可使用以下两种方法进入或退出静音模式：

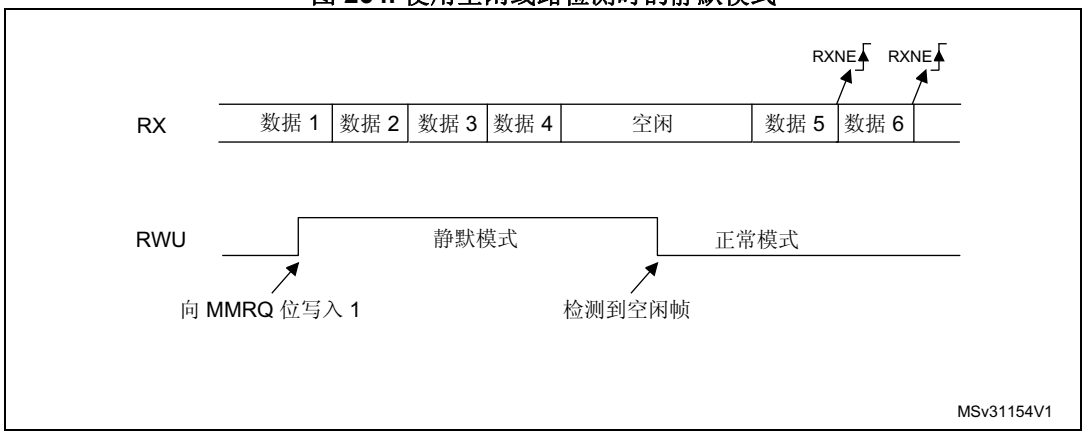
- 如果 WAKE 位被复位，则进行空闲线路检测，
- 如果 WAKE 位置 1，则进行地址标记检测。

空闲线路检测 (WAKE=0)

当向 MMRQ 位写入 1 且 RWU 位自动置 1 时，LPUART 进入静默模式。

当检测到空闲帧时，它会被唤醒。此时 RWU 位会由硬件清零，但 LPUART_ISR 寄存器中的 IDLE 位不会置 1。*图 242* 中给出了使用空闲线路检测时静默模式行为的示例。

图 264. 使用空闲线路检测时的静默模式



注： *如果在 IDLE 字符已经过去时将 MMRQ 位置 1，将不会进入静默模式（RWU 未置 1）。*

如果在线路处于空闲状态时激活 LPUART，在一个 IDLE 帧持续时间后（不只在接收一个字符帧后）会检测到空闲状态。

4 位/7 位地址标记检测 (WAKE=1)

在此模式下，如果字节的 MSB 为 1，则将这些字节识别为地址，否则将其识别为数据。在地址字节中，目标接收器的地址位于 4 个或 7 个 LSB 中。7 位或 4 位地址检测通过 ADDM7 位来选择。接收器会将此 4 位/7 位字与其地址进行比较，该接收器的地址在 LPUART_CR2 寄存器的 ADD 位中进行设置。

注： *在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。*

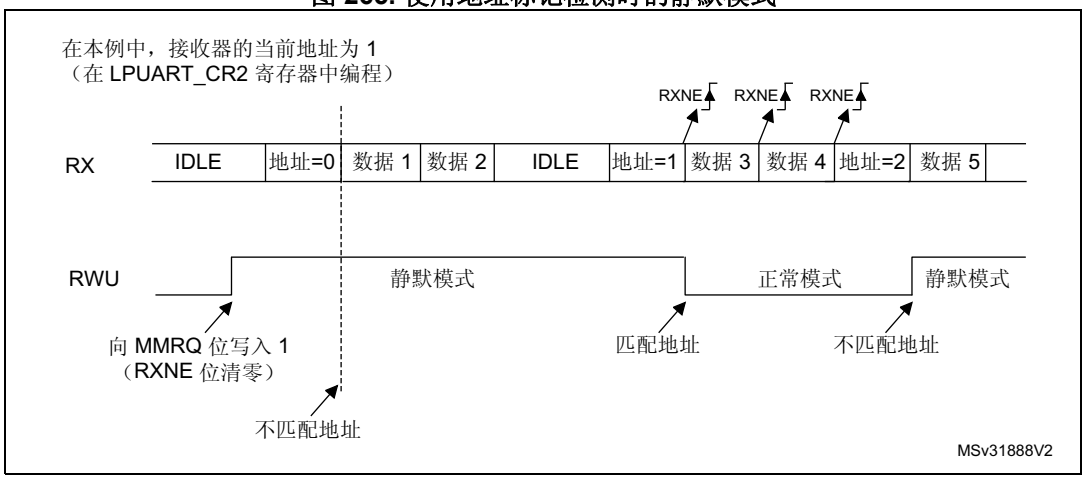
当接收到与其编程地址不匹配的地址字符时，LPUART 会进入静默模式。此时，RWU 位将由硬件置 1。LPUART 进入静默模式后，RXNE 标志不会针对此地址字节置 1，也不会发出中断或 DMA 请求。

当向 MMRQ 位写入 1 时，LPUART 也会进入静默模式。这种情况下，RWU 位也自动置 1。

当接收到与编程地址匹配的地址字符时，LPUART 会退出静默模式。然后 RWU 位被清零，可以开始正常接收后续字节。由于 RWU 位已清零，RXNE 位会针对地址字符置 1。

图 243 中给出了使用地址标记检测时静默模式行为的示例。

图 265. 使用地址标记检测时的静默模式



30.4.7 LPUART 极性控制

将 LPUART_CR1 寄存器中的 PCE 位置 1，可以使能奇偶校验控制（发送时生成奇偶校验位，接收时进行奇偶校验检查）。根据 M 位定义的帧长度，*表 128* 中列出了可能的 LPUART 帧格式。

表 134. 帧格式

M 位	PCE 位	LPUART 帧(1)
00	0	| SB | 8 位数据 | STB |
00	1	| SB | 7 位数据 | PB | STB |
01	0	| SB | 9 位数据 | STB |
01	1	| SB | 8 位数据 | PB | STB |
10	0	| SB | 7 位数据 | STB |
10	1	| SB | 6 位数据 | PB | STB |

1. 图注：SB：起始位，STB：停止位，P：奇偶校验位。
2. 在数据寄存器中，PB 始终位于 MSB 位置（第 9 位、第 8 位或第 7 位，具体取决于 M 位的值）。

偶校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为偶数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择偶校验（LPUART_CR1 寄存器中的 PS 位 = 0）时，校验位为 0。

奇校验

对奇偶校验位进行计算，使帧和奇偶校验位中“1”的数量为奇数（帧由 6 个、7 个或 8 个 LSB 位组成，具体取决于 M 位的值）。

例如，如果数据=00110101 且 4 个位置 1，则在选择奇校验（LPUART_CR1 寄存器中的 PS 位 = 1）时，校验位为 1。

接收时进行奇偶校验检查

如果奇偶校验检查失败，则 LPUART_ISR 寄存器中的 PE 标志置 1；如果 LPUART_CR1 寄存器中 PEIE 位置 1，则会生成中断。通过软件将 1 写入 LPUART_ICR 寄存器中的 PECF 位来清零 PE 标志。

发送时的奇偶校验生成

如果 LPUARTx_CR1 寄存器中的 PCE 位置 1，则在数据寄存器中所写入数据的 MSB 位会进行传送，但是会由奇偶校验位进行更改（如果选择偶校验 (PS=0)，则“1”的数量为偶数；如果选择奇校验 (PS=1)，则“1”的数量为奇数）。

30.4.8 使用 LPUART 单线半双工通信

通过将 LPUART_CR3 寄存器中的 HDSEL 位置 1 来选择单线半双工模式。在此模式下，必须将以下位清零：

- LPUART_CR2 寄存器中的 LINEN 和 CLKEN 位，
- LPUART_CR3 寄存器中的 SCEN 和 IREN 位。

LPUART 可以配置为遵循单线半双工协议，其中 TX 和 RX 线路从内部相连接。使用 LPUART_CR3 中的控制位 HDSEL 可在半双工通信和全双工通信间进行选择。

一旦向 HDSEL 位写入 1：

- TX 和 RX 线路从内部相连接
- 不能再使用 RX 引脚
- 无数据传输时，TX 引脚始终处于释放状态。因此，它在空闲状态或接收过程中用作标准 I/O。这意味着，必须对 I/O 进行配置，以便将 TX 配置为复用功能开漏并外接上拉电阻。

除此之外，通信协议与正常 LPUART 模式下的通信协议相似。此线路上的任何冲突必须由软件管理（例如，使用中央仲裁器）。尤其要注意，发送过程永远不会被硬件封锁，只要数据是在 TE 位置 1 的情况下写入，发送就会持续进行。

注：在 LPUART 中，当进行 1 个停止位配置时，RXNE 标志在停止位中间置 1。

30.4.9 使用 LPUART 在 DMA 模式下进行连续通信

LPUART 能够使用 DMA 进行连续通信。接收缓冲区和发送缓冲区的 DMA 请求是独立生成的。

注：按照第 30.4.3 节中的说明使用 LPUART。可以将 LPUART_ISR 寄存器中的 TXE/ RXNE 标志清零，从而实现连续通信。

使用 DMA 进行发送

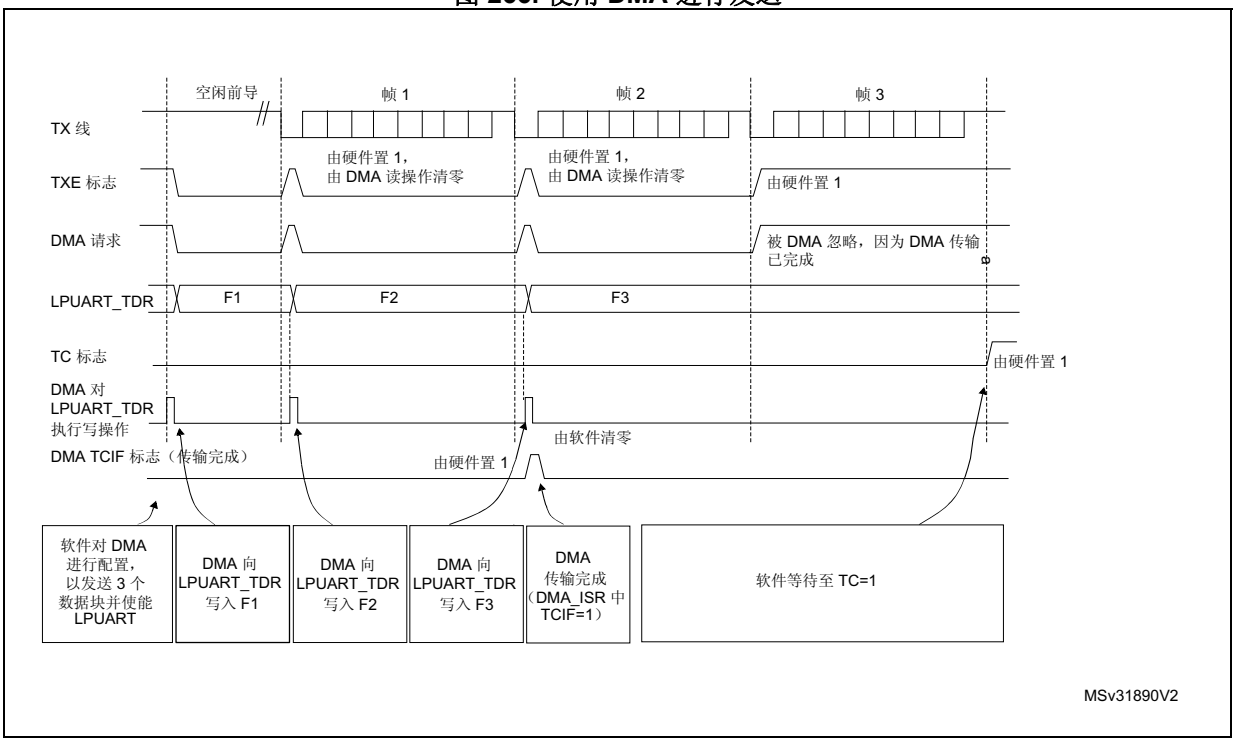
将 LPUART_CR3 寄存器中的 DMAT 位置 1 可以使能 DMA 模式进行发送。当 TXE 位置 1 时，可将数据从 SRAM 区（通过 DMA 外设，请参见*第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）加载到 LPUART_TDR 寄存器。要映射一个 DMA 通道以进行 LPUART 发送，请按以下步骤操作（x 表示通道编号）：

1. 在 DMA 控制寄存器中写入 LPUART_TDR 寄存器地址，将其配置为传输的目标地址。每次发生 TXE 事件后，数据都从存储器移动到此地址。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的源地址。每次发生 TXE 事件后，数据都从这个存储区域加载到 LPUART_TDR 寄存器中。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 寄存器中配置通道优先级
5. 根据应用的需求，在完成一半或全部传输后产生 DMA 中断。
6. 通过将 LPUART_ICR 寄存器中的 TCCF 位置 1，将 LPUART_ISR 寄存器中的 TC 标志清零。
7. 在 DMA 寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

在发送模式下，DMA 对所有要发送的数据执行了写操作（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 TC 标志进行监视，以确保 LPUART 通信已完成。在禁止 LPUART 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须等待直到 TC=1。TC 标志在所有数据发送期间都保持清零状态，然后在最后一帧发送结束时由硬件置 1。

图 266. 使用 DMA 进行发送



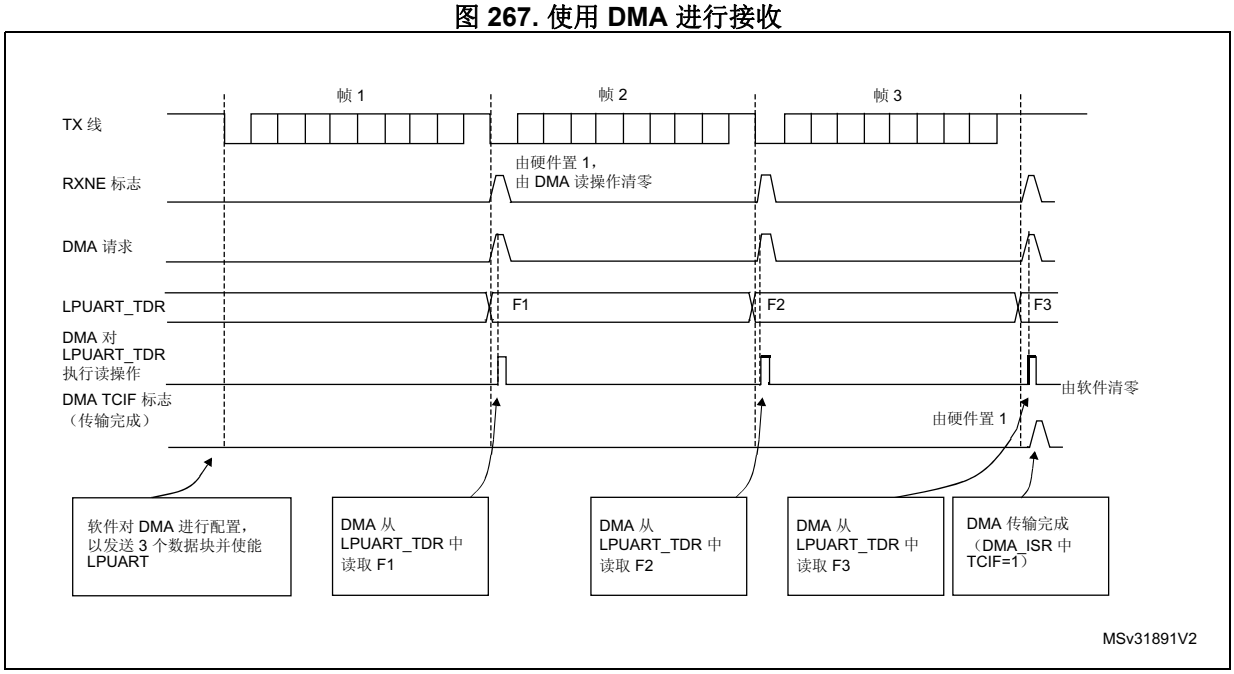
使用 DMA 进行接收

将 LPUART_CR3 寄存器中的 DMAR 位置 1 可以使能 DMA 模式进行接收。接收数据字节时，数据会从 LPUART_RDR 寄存器加载到通过 DMA 外设配置的 SRAM 区域中（请参见 *第 246 页上的第 11 节：直接存储器访问控制器 (DMA)*）。要映射一个 DMA 通道以进行 LPUART 接收，请按以下步骤操作：

1. 在 DMA 控制寄存器中写入 LPUART_RDR 寄存器地址，将其配置为传输的源地址。每次发生 RXNE 事件后，数据都从此地址移动到存储器。
2. 在 DMA 控制寄存器中写入存储器地址，将其配置为传输的目标地址。每次发生 RXNE 事件后，数据都从 LPUART_RDR 寄存器加载到此存储区。
3. 在 DMA 控制寄存器中配置要传输的总字节数。
4. 在 DMA 控制寄存器中配置通道优先级。
5. 根据应用的需求，在完成一半或全部传输后产生中断。
6. 在 DMA 控制寄存器中激活该通道。

当达到在 DMA 控制器中设置的数据传输量时，DMA 控制器会在 DMA 通道的中断向量上产生一个中断。

图 267. 使用 DMA 进行接收



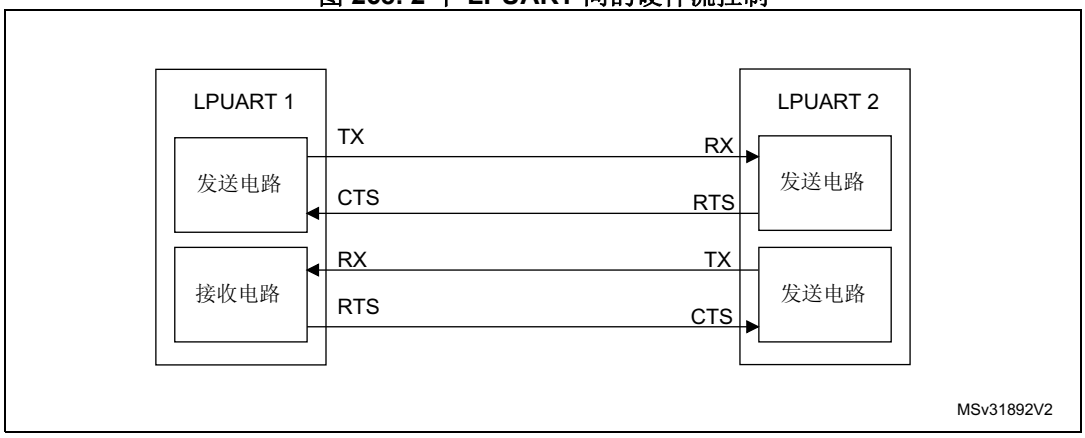
多缓冲区通信中的错误标志和中断生成

在多缓冲区通信中，如果事务中发生任何错误，都会在当前字节后放置错误标志。如果中断使能标志置 1，则会产生中断。在单字节接收过程中，与 RXNE 一同置位的帧错误、上溢错误和噪声标志具有单独的错误标志中断使能位（LPUART_CR3 寄存器中的 EIE 位）；如果该位置 1，在发生其中任何一个错误时，在当前字节后使能中断。

30.4.10 RS232 硬件流控制和 RS485 驱动器使能（使用 LPUART）

使用 CTS 输入和 RTS 输出可以控制 2 个器件间的串行数据流。*图 256* 显示了在这种模式下如何连接 2 个器件：

图 268. 2 个 LPUART 间的硬件流控制

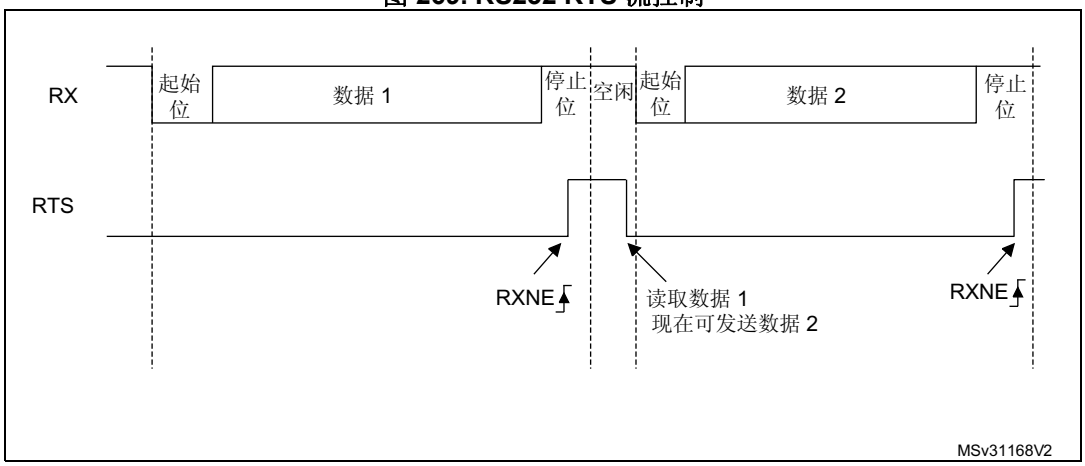


分别向 LPUART_CR3 寄存器中的 RTSE 位和 CTSE 位写入 1，可以分别使能 RS232 RTS 和 CTS 流控制。

RS232 RTS 流控制

如果使能 RTS 流控制 (RTSE=1)，只要 LPUART 接收器准备好接收新数据，便会将 RTS 变为有效（连接到低电平）。当接收寄存器已满时，会将 RTS 变为无效，表明发送过程会在当前帧结束后停止。*图 257* 显示了在使能 RTS 流控制的情况下进行通信的示例。

图 269. RS232 RTS 流控制

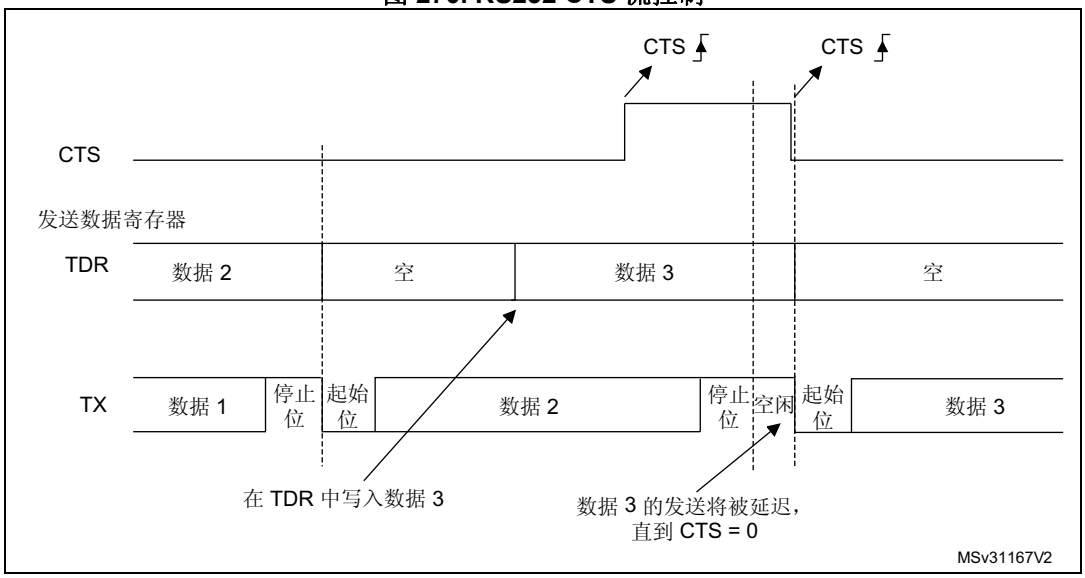


RS232 CTS 流控制

如果使能 CTS 流控制 (CTSE=1)，则发送器会在发送下一帧前检查 CTS。如果 CTS 有效（连接到低电平），则会发送下一数据（假设数据已准备好发送，即 TXE=0）；否则不会进行发送。如果在发送过程中 CTS 变为无效，则当前发送完成之后，发送器停止。

当 CTSE=1 时，只要 CTS 发生变化，CTSIF 状态位便会由硬件自动置 1。这指示接收器是否已准备好进行通信。如果 LPUART_CR3 寄存器中的 CTSIE 位置 1，则会产生中断。*图 258* 显示了在使能 CTS 流控制的情况下进行通信的示例。

图 270. RS232 CTS 流控制



注：为正常运行，必须在当前字符结束前至少 3 个 LPUART 时钟源周期内使能 CTS。此外还应注意，当脉冲短于 2 个 PCLK 周期时无法将 CTSCF 标志置 1。

RS485 驱动器使能

驱动器使能功能可通过将 LPUART_CR3 控制寄存器中的 DEM 位置 1 使能。这样用户便可通过 DE（驱动器使能）信号激活外部收发器控制。使能时间为激活 DE 信号与 START 位开始间的时间。可以通过 LPUART_CR1 控制寄存器中的 DEAT [4:0] 位域编程使能时间。禁止时间为发送的消息中最后一个停止位结束与取消激活 DE 信号间的时间。可以通过 LPUART_CR1 控制寄存器中的 DEDT [4:0] 位域编程使能时间。DE 信号的极性可使用 LPUART_CR3 控制寄存器中的 DEP 位配置。

在 LPUART 中，DEAT 和 DEDT 用 USART 时钟源 (f_{CK}) 周期表示：

- 驱动器使能有效时间 =
  - (1 + (DEAT x P)) x f_{CK}（如果 P <> 0）
  - (1 + DEAT) x f_{CK}（如果 P = 0）
- 驱动器使能禁止时间 =
  - (1 + (DEDT x P)) x f_{CK}（如果 P <> 0）
  - (1 + DEDT) x f_{CK}（如果 P = 0）

P = BRR[14:11]

30.4.11 使用 LPUART 从停止模式唤醒

当 UESM 位置 1 且 LPUART 时钟设置为 HSI 或 LSE 时，LPUART 能够将 MCU 从停止模式唤醒（请参见复位和时钟控制 (RCC) 部分）。

可使用标准 RXNE 中断将 MCU 从停止模式唤醒。在这种情况下，必须在进入停止模式前将 RXNEIE 位置 1。

也可通过 WUS 位域选择特定中断。

为了能够将 MCU 从停止模式唤醒，必须在进入停止模式前将 LPUART_CR1 控制寄存器中的 UESM 位置 1。

检测到唤醒事件后，WUF 标志会由硬件置 1 并在 WUFIE 位置 1 时生成一个唤醒中断。

有关代码示例，请参见 *A.18.1：LPUART 接收器配置代码示例*和 *A.18.2：LPUART 接收字节代码示例*。

注： *在进入停止模式前，用户必须确保 LPUART 未在执行传输。BUSY 标志无法确保运行接收期间始终不进入停止模式。*

WUF 标志在检测到唤醒事件时置 1，而与 MCU 处于停止模式还是工作模式无关。

如果在初始化和使能接收器后立即进入停止模式，则必须校验 REACK 位以确保 LPUART 确实已使能。

当 DMA 用于接收时，它必须在进入停止模式前禁止，并在退出停止模式后重新使能。

从停止模式唤醒功能并非在所有模式下均可用。例如，该功能在 SPI 模式下不起作用，因为 SPI 仅在主模式下工作。

使用静默模式和停止模式

如果 LPUART 在进入停止模式前处于静默模式：

- 不得使用空闲检测时从静默模式唤醒，因为空闲检测无法在停止模式下工作。
- 如果使用地址匹配时从静默模式唤醒，则从停止模式唤醒的源也必须是地址匹配。如果 RXNE 标志在进入停止模式时置 1，则接口将在地址匹配时和从停止模式唤醒时保持静默模式。
- 如果 LPUART 配置为在 START 位检测时将 MCU 从停止模式唤醒，WUF 标志将置 1 但 RXNE 标志不置 1。

当 LPUART 时钟源为 HSI 时钟时，确定最大 LPUART 波特率可使从停止模式正确地唤醒

允许从停止模式正确唤醒的最大波特率取决于：

- 器件数据手册中提供的参数 $t_{WULPUART}$（从停止模式唤醒）
- *第 30.4.5 节：LPUART 接收器对时钟偏差的容差*中提供的 LPUART 接收器的容差。

举例来说：M 位 = 01 个、2 个停止位，BRR ≥ 4096。

在这些条件下，根据*表 133：BRR[3:0] 不等于 0000 时的 LPUART 接收器容差*，LPUART 接收器的容差为 4.42 %。

DTRA + DQUANT + DREC + DTCL + DWU < LPUART 接收器的容差

DWU max = $t_{WULPUART}$ / (11 x Tbit Min)

Tbit Min = $t_{WULPUART}$ / (11 x DWU max)

考虑一种理想情况：参数 DTRA、DQUANT、DREC 和 DTCL 为 0%，则 DWU 最大值为 4.42 %。实际上，我们至少需要考虑 HSI 不精确的情况。

假设 HSI 不精确度 = 1 %，则 $t_{WULPUART}$ = 8.1 µs（处于停止模式，主调压器处于运行模式，范围 1）：

DWU max = 4.42 % - 1 % = 3.42 %

Tbit min = 8.1 µs / (11 x 3.42 %) = 2.5 µs。

在这些情况下，可使从停止模式正确唤醒的最大波特率为 1/ 21.5 µs = 46 K 波特。

30.5 LPUART 低功耗模式

表 135. 低功耗模式对 LPUART 的影响

模式	说明
睡眠	无影响。USART 中断可使器件退出睡眠模式。
低功耗运行	无影响。
低功耗睡眠	无影响。USART 中断可使器件退出低功耗睡眠模式。
停止	当 UESM 位置 1 且 LPUART 时钟设置为 HSI16 或 LSE 时，LPUART 能够将 MCU 从停止模式唤醒。 可使用标准 RXNE 或 WUF 中断将 MCU 从停止模式唤醒。
待机	LPUART 掉电，当器件退出待机模式后必须重新初始化 USART。

30.6 LPUART 中断

表 136. LPUART 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 中断	CTSIF	CTSIE
发送完成	TC	TCIE
接收数据寄存器不为空（已准备好读取数据）	RXNE	RXNEIE
检测到溢出错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
多缓冲区通信中的噪声标志、溢出错误和帧错误	NF 或 ORE 或 FE	EIE
字符匹配	CMF	CMIE
从停止模式唤醒	WUF(1)	WUFIE

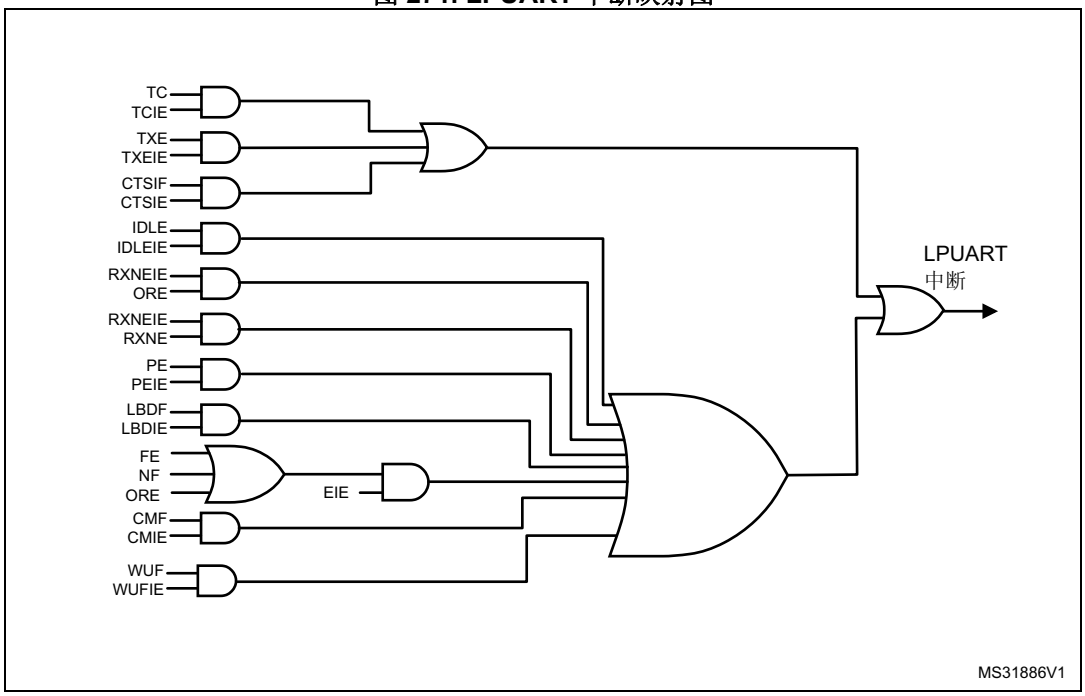
1. WUF 中断仅在停止模式下有效。

LPUART 中断事件被连接到相同的中断向量（请参见*图 259*）。

- 发送期间：发送完成、清除以发送、发送数据寄存器为空或帧错误中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、噪声标志、帧错误、字符匹配等。

如果相应的使能控制位置 1，则这些事件会生成中断。

图 271. LPUART 中断映射图



30.7 LPUART 寄存器

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

30.7.1 控制寄存器 1 (LPUART_CR1)

Control register 1

偏移地址：0x00

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:29 保留，必须保持复位值

位 28 **M1**：字长 (Word length)

此位和位 12 (M0) 用于确定字长度。此位由软件置 1 或清零。

M[1:0] = 00：1 个起始位，8 个数据位，n 个停止位

M[1:0] = 01：1 个起始位，9 个数据位，n 个停止位

M[1:0] = 10：1 个起始位，7 个数据位，n 个停止位

只有在禁止 LPUART (UE=0) 时才能写入此位。

注： 7 位数据长度模式下，不支持智能卡模式、LIN 主模式和自动波特率（0x7F 帧和 0x55 帧检测）。

位 27 保留，必须保持复位值

位 26 保留，必须保持复位值

位 25:21 **DEAT[4:0]**：驱动器使能使能时间 (Driver Enable assertion time)

该 5 位值用于定义激活 DE（启动器使能）信号与起始位开始间的时间。以 UCLK（USART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 20:16 **DEDT[4:0]**：驱动器使能禁止时间 (Driver Enable de-assertion time)

该 5 位值用于定义发送的消息中最后一个停止位结束与取消激活 DE（驱动器使能）信号间的时间。以 UCLK（USART 时钟）时钟周期数表示。有关更多详细信息，请参见 RS485 驱动器使能段落。

如果在 DEDT 时间内对 LPUART_TDR 寄存器执行写操作，则新数据仅在经过 DEDT 和 DEAT 时间后才会发送。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 15 保留，必须保持复位值

位 14 **CMIE**：字符匹配中断使能 (Character match interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：如果 LPUART_ISR 寄存器中的 CMF 位置 1，则生成 LPUART 中断。

位 13 **MME**：静默模式使能 (Mute mode enable)

此位用于激活 LPUART 的静默模式功能。此位置 1 时，LPUART 可按 WAKE 位定义的方式在活动模式与静默模式之间切换。此位由软件置 1 和清零。

0：接收器永久处于活动模式

1：接收器可在静默模式和活动模式之间切换。

位 12 **M0**：字长 (Word length)

此位和位 28 (M1) 用于确定字长度。此位由软件置 1 或清零。请参见位 28 (M1) 的说明。

只有在禁止 LPUART (UE=0) 时才能写入此位。

位 11 **WAKE**：接收器唤醒方法 (Receiver wakeup method)

此位用于确定 LPUART 静默模式的唤醒方法。此位由软件置 1 或清零。

0：空闲线路

1：地址标记

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 10 **PCE**：奇偶校验控制使能 (Parity control enable)

该位选择硬件奇偶校验控制（生成和检测）。使能奇偶校验控制时，计算出的奇偶校验位被插入到 MSB 位置（如果 M=1，则为第 9 位；如果 M=0，则为第 8 位），并对接收到的数据检查奇偶校验位。此位由软件置 1 和清零。一旦该位置 1，PCE 在当前字节的后面处于活动状态（在接收和发送时）。

0：禁止奇偶校验控制

1：使能奇偶校验控制

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 9 **PS**：奇偶校验选择 (Parity selection)

该位用于在使能奇偶校验生成/检测（PCE 位置 1）时选择奇校验或偶校验。此位由软件置 1 和清零。将在当前字节的后面选择奇偶校验。

0：偶校验

1：奇校验

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 8 **PEIE**：PE 中断使能 (PE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 PE=1 时，生成 LPUART 中断

位 7 **TXEIE**：中断使能 (interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 TXE=1 时，生成 LPUART 中断

位 6 **TCIE**：传输完成中断使能 (Transfer complete interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 TC=1 时，生成 LPUART 中断

位 5 **RXNEIE**：RXNE 中断使能 (RXNE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 ORE=1 或 RXNE=1 时，生成 LPUART 中断

位 4 **IDLEIE**：IDLE 中断使能 (IDLE interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 IDLE=1 时，生成 LPUART 中断

位 3 **TE**：发送器使能 (Transmitter enable)

该位使能发送器。此位由软件置 1 和清零。

0：禁止发送器

1：使能发送器

注： 传送期间 TE 位上的“0”脉冲（“0”后紧跟的是“1”）会在当前字的后面发送一个报头（空闲线路）。为生成空闲字符，TE 不能立即写入 1。为确保所需的持续时间，软件可轮询 LPUART_ISR 寄存器中的 TEACK 位。

当 TE 置 1 时，在发送开始前存在 1 位的时间延迟。

位 2 **RE**：接收器使能 (Receiver enable)

该位使能接收器。此位由软件置 1 和清零。

0：禁止接收器

1：使能接收器并开始搜索起始位

位 1 **UESM**：停止模式下的 LPUART 使能 (LPUART enable in Stop mode)

当此位清零时，LPUART 无法将 MCU 从停止模式唤醒。

当此位置 1 时，LPUART 能够将 MCU 从停止模式唤醒，前提是 LPUART 时钟选择为 HSI 或 LSE（在 RCC 中）。

此位由软件置 1 和清零。

0：LPUART 无法将 MCU 从停止模式唤醒。

1：LPUART 能够将 MCU 从停止模式唤醒。当该功能激活时，LPUART 的时钟源必须是 HSI 或 LSE（请参见复位和时钟控制 (RCC) 部分）。

注： 建议在进入停止模式前将 UESM 位置 1，并在退出停止模式时将其清零。

位 0 **UE**：LPUART 使能 (CTS enable)

此位清零后，LPUART 预分频器和输出将立即停止，并丢弃当前操作。LPUART 的配置保留，但 LPUART_ISR 中的所有状态标志将被复位。此位由软件置 1 和清零。

0：禁止 LPUART 预分频器和输出，低功耗模式

1：使能 LPUART

注： 为进入低功耗模式而不在线路上生成错误，之前必须复位 TE 位，并且软件必须等待 LPUART_ISR 寄存器中的 TC 位置 1 后才能复位 UE 位。

UE = 0 时也会复位 DMA 请求，因必须在复位 UE 位前禁止 DMA 通道。

30.7.2 控制寄存器 2 (LPUART_CR2)

Control register 2

偏移地址：0x04

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
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
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

位 31:28 **ADD[7:4]**：LPUART 节点的地址 (Address of the USART node)

此位域用于指定 LPUART 节点的地址或要识别的字符代码。

此位域在多处理器通信时于静默模式或停止模式下使用，以通过 7 位地址标记检测进行唤醒。发送器发送字符的 MSB 应为 1。此位域还可用于正常接收和静默模式无效时的字符检测（例如，Modbus 协议中的块结束检测）。这种情况下，接收到的整个字符（8 位）将与 ADD[7:0] 值进行比较，如果匹配，CMF 标志将置 1。

仅在禁止接收 (RE = 0) 或禁止 LPUART (UE=0) 时才能写入该位域

位 27:24 **ADD[3:0]**：LPUART 节点的地址 (Address of the USART node)

此位域用于指定 LPUART 节点的地址或要识别的字符代码。

此位域在多处理器通信时于静默模式或停止模式下使用，以通过地址标记检测进行唤醒。

仅在禁止接收 (RE = 0) 或禁止 LPUART (UE=0) 时才能写入该位域

位 23:20 保留，必须保持复位值

位 19 **MSBFIRST**：最高有效位在前 (Most significant bit first)

此位由软件置 1 和清零。

0：发送/接收数据时位 0 在前，后跟起始位。

1：发送/接收数据时 MSB（位 7/8/9）在前，后跟起始位。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 18 **DATAINV**：二进制数据反向 (Binary data inversion)

此位由软件置 1 和清零。

0：按正/正向逻辑发送/接收数据寄存器中的逻辑数据。（1=H，0=L）

1：按负/反向逻辑发送/接收数据寄存器中的逻辑数据。（1=L，0=H）。奇偶校验位也取反。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 17 **TXINV**：TX 引脚有效电平反向 (TX pin active level inversion)

此位由软件置 1 和清零。

0：TX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲，Gnd = 0/标记）工作。

1：对 TX 引脚信号值取反。（V_{DD} = 0/标记，Gnd=1/空闲）。

允许在 TX 线路上使用外部反相器。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 16 **RXINV**：RX 引脚有效电平反向 (RX pin active level inversion)

此位由软件置 1 和清零。

0：RX 引脚信号使用标准逻辑电平（V_{DD} = 1/空闲，Gnd = 0/标记）工作。

1：对 RX 引脚信号值取反。（V_{DD} = 0/标记，Gnd=1/空闲）。

允许在 RX 线路上使用外部反相器。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 15 **SWAP**：交换 TX/RX 引脚 (Swap TX/RX pins)

此位由软件置 1 和清零。

0：按标准引脚排列定义使用 TX/RX 引脚

1：交换 TX 和 RX 引脚功能。允许在与另一个 UART 的交叉连接时工作。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 14 保留，必须保持复位值

位 13:12 **STOP[1:0]**：停止位 (STOP bits)

这些位用于编程停止位。

00：1 个停止位

01：保留

10：2 个停止位

11：保留

只有在禁止 LPUART (UE=0) 后才能写入此位域。

位 11:5 保留，必须保持复位值

位 4 **ADDM7**：7 位地址检测/4 位地址检测 (7-bit Address Detection/4-bit Address Detection)

此位用于选择 4 位地址检测或 7 位地址检测。

0：4 位地址检测

1：7 位地址检测（在 8 位数据模式下）

只有在禁止 LPUART (UE=0) 时才能写入该位

注： 在 7 位和 9 位数据模式下，地址检测分别在 6 位和 8 位地址上完成（ADD[5:0] 和 ADD[7:0]）。

位 3:0 保留，必须保持复位值。

30.7.3 控制寄存器 3 (LPUART_CR3)

Control register 3

偏移地址：0x08

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS[2:0]		Res.	Res.	Res.	Res.
									rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HD SEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

位 31:23 保留，必须保持复位值。

位 22 **WUFIE**：从停止模式唤醒中断使能 (Wakeup from Stop mode interrupt enable)

此位由软件置 1 和清零。

0：禁止中断

1：当 LPUART_ISR 寄存器中的 WUF=1 时，生成 LPUART 中断

注： WUFIE 必须在进入停止模式前置 1。

WUF 中断仅在停止模式下有效。

如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 21:20 **WUS[1:0]**：从停止模式唤醒中断标志选择 (Wakeup from Stop mode interrupt flag selection)

该位域用于指定激活 WUF（从停止模式唤醒标志）的事件。

00：WUF 在地址匹配时激活（按 ADD[7:0] 和 ADDM7 所定义）

01：保留。

10：WUF 在起始位检测时激活

11：WUF 在 RXNE 时激活。

只有在禁止 LPUART (UE=0) 后才能写入此位域。

注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19:16 保留，必须保持复位值。

位 15 **DEP**：驱动器使能极性选择 (Driver enable polarity selection)

0：DE 信号高电平有效。

1：DE 信号低电平有效。

只有在禁止 LPUART (UE=0) 时才能写入此位。

位 14 **DEM**：驱动器使能模式 (Driver enable mode)

此位用于通过 DE 信号激活外部收发器控制。

0：禁止 DE 功能。

1：使能 DE 功能。DE 信号在 RTS 引脚上输出。

只有在禁止 LPUART (UE=0) 时才能写入此位。

位 13 **DDRE**：接收出错时的 DMA 禁止 (DMA Disable on Reception Error)

0：接收出错时不禁止 DMA。相应的错误标志置 1，但 RXNE 保持为 0 以防止上溢。因此，将不使能 DMA 请求，从而不会传送错误数据（无 DMA 请求），但会传送接收到的下一个正确数据。

1：接收出错后禁止 DMA。相应的错误标志以及 RXNE 均置 1。屏蔽 DMA 请求，直到错误标志清零。这意味着软件必须首先禁止 DMA 请求 (DMAR = 0) 或者将 RXNE 清零，然后再将错误标志清零。

只有在禁止 LPUART (UE=0) 时才能写入此位。

注： 接收错误包括：奇偶校验错误、帧错误或噪声错误。

位 12 **OVRDIS**：上溢禁止 (Overrun Disable)

此位用于禁止接收上溢检测。

0：接收新数据前未读取已接收的数据时，上溢错误标志 ORE 置 1。

1：禁止上溢功能。如果在 RXNE 标志仍置 1 时接收到新数据，则 ORE 标志不会置 1，且新接收的数据会覆盖 LPUART_RDR 寄存器之前的内容。

只有在禁止 LPUART (UE=0) 时才能写入此位。

注： 此控制位用于检查通信流而不会读取数据。

位 11 保留，必须保持复位值。

位 10 **CTSIE**：CTS 中断使能 (CTS interrupt enable)

0：禁止中断

1：当 LPUART_ISR 寄存器中 CTSIF = 1 时，生成中断

位 9 **CTSE**：CTS 使能 (CTS enable)

0：禁止 CTS 硬件流控制

1：使能 CTS 模式，仅当 CTS 输入有效（连接到 0）时才发送数据。如果在发送数据时使 CTS 输入无效，会在停止之前完成发送。如果使 CTS 无效时数据已写入数据寄存器，则将延迟发送，直到 CTS 有效。

只有在禁止 LPUART (UE=0) 时才能写入该位

位 8 **RTSE**：RTS 使能 (RTS enable)

0：禁止 RTS 硬件流控制

1：使能 RTS 输出，仅当接收缓冲区中有空间时才会请求数据。发送完当前字符后应停止发送数据。可以接收数据时使 RTS 输出有效（拉至 0）。

只有在禁止 LPUART (UE=0) 时才能写入此位。

位 7 **DMAT**：DMA 使能发送器 (DMA enable transmitter)

此位由软件置 1/复位。

1：针对发送使能 DMA 模式

0：针对发送禁止 DMA 模式

位 6 **DMAR**：DMA 使能接收器 (DMA enable receiver)

此位由软件置 1/复位。

1：针对接收使能 DMA 模式

0：针对接收禁止 DMA 模式

位 5:4 保留，必须保持复位值。

位 3 **HDSEL**：半双工选择 (Half-duplex selection)

选择单线半双工模式

0：未选择半双工模式

1：选择半双工模式

只有在禁止 LPUART (UE=0) 时才能写入此位。

位 2:1 保留，必须保持复位值。

位 0 **EIE**：错误中断使能 (Error interrupt enable)

如果发生帧错误、上溢错误或出现噪声标志（LPUART_ISR 寄存器中 FE = 1 或 ORE = 1 或 NF = 1），则需要使用错误中断使能位来使能中断生成。

0：禁止中断

LPUART_ISR 寄存器中的 FE=1 或 ORE=1 或 NF=1 时生成中断。

30.7.4 波特率寄存器 (LPUART_BRR)

Baud rate register

只有在禁止 LPUART (UE=0) 时才能写入此寄存器。

偏移地址：0x0C

复位值：0x0000

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

位 31:20 保留，必须保持复位值。

位 19:0 **BRR[19:0]**

注： *禁止在 LPUART_BRR 寄存器中写入小于 0x300 的值。*

如果 LPUARTx_BRR 必须 > = 0x300 且 LPUART_BRR 为 20 位，则使用高 fck 值生成高波特率时应十分谨慎。fck 必须在 [3 x 波特率，4096 x 波特率] 范围内。

30.7.5 请求寄存器 (LPUART_RQR)

Request register

偏移地址：0x18

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFRQ	MMRQ	SBKRQ	Res.
												w	w	w	

位 31:4 保留，必须保持复位值

位 3 **RXFRQ**：接收数据刷新请求 (Receive data flush request)

向该位写入 1 时会将 RXNE 标志清零。

这可以丢弃数据而不对其执行读取操作，并避免发生上溢。

位 2 **MMRQ**：静默模式请求 (Mute mode request)

向此位写入 1 可将 LPUART 置于静默模式，并将 RWU 标志复位。

位 1 **SBKRQ**：发送断路请求 (Send break request)

向此位写入 1 可将 SBKF 标志置 1 并在发送设备可用后立即请求在线路上发送 BREAK。

注： 这种情况下，应用需要在之前插入的所有数据（包括尚未发送的数据）后发送中断字符，软件应等到 TXE 标志使能后将 SBKRQ 位置 1。

位 0 保留，必须保持复位值

30.7.6 中断和状态寄存器 (LPUART_ISR)

Interrupt & status register

偏移地址：0x1C

复位值：0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
					r	r		r	r	r	r	r	r	r	r

位 31:23 保留，必须保持复位值。

位 22 **REACK**：接收使能确认标志 (Receive enable acknowledge flag)

LPUART 采用接收使能值时，通过硬件将此位置 1/复位。

此位可用于验证 LPUART 是否准备好在进入停止模式前接收数据。

注： 如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 21 **TEACK**：发送使能确认标志 (Transmit enable acknowledge flag)

LPUART 采用发送使能值时，通过硬件将此位置 1/复位。

通过写入 TE=0 生成空闲帧请求，然后在 LPUART_CR1 寄存器中写入 TE=1 以遵循 TE=0 最短周期时，可使用此位。

位 20 **WUF**：从停止模式唤醒标志 (Wakeup from Stop mode flag

当检测到唤醒事件时，此位由硬件置 1。事件通过 WUS 位域定义。通过向 LPUART_ICR 寄存器中的 WUCF 写入 1，此位由软件清零。

如果 LPUART_CR3 寄存器中 WUFIE=1，则会生成中断。

注：　当 UESM 清零时，WUF 标志也清零。

WUF 中断仅在停止模式下有效。

如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19 **RWU**：*接收器从静默模式唤醒 (Receiver wakeup from Mute mode)*

此位指示 LPUART 是否处于静默模式。当识别出唤醒/静默序列时，此位由硬件清零/置 1。静默模式控制序列（地址或 IDLE）通过 LPUART_CR1 寄存器中的 WAKE 位选择。

当选择 IDLE 模式下唤醒时，该位只能通过用软件向 LPUART_RQR 寄存器中的 MMRQ 位写 1 的方式置 1。

0：接收器处于活动模式

1：接收器处于静默模式

注：　如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 18 **SBKF**：发送断路标志 (Send break flag)

此位指示已请求发送断路字符。通过将 1 写入 LPUART_CR3 寄存器中的 SBKRQ 位，此位由软件置 1。此位在断路发送的停止位期间由硬件自动复位。

0：不发送断路字符

1：将发送断路字符

位 17 **CMF**：字符匹配标志 (Character match flag)

接收到由 ADD[7:0] 定义的字符后由硬件将此位置 1。通过向 LPUART_ICR 寄存器中的 CMCF 写入 1，此位由软件清零。

如果 LPUART_CR1 寄存器中 CMIE=1，则会生成中断。

0：未检测到字符匹配

1：检测到字符匹配

位 16 **BUSY**：忙标志 (Busy flag)

此位由硬件置 1 和复位。当 RX 线路上正在进行通信（成功检测到起始位）时有效。在接收结束（成功或失败）时复位。

0：LPUART 处于空闲状态（无接收）

1：正在接收

位 15:11 保留，必须保持复位值。

位 10 **CTS**：CTS 标志 (CTS flag)

此位由硬件置 1/复位。此位是对 CTS 输入引脚的状态取反。

0：CTS 线置 1

1：CTS 线复位

注：　如果不支持硬件流控制功能，该位保留并由硬件强制清零。

位 9 **CTSIF**：CTS 中断标志 (CTS interrupt flag)

如果 CTSE 位置 1，当 CTS 输入切换时，此位由硬件置 1。通过将 1 写入 LPUART_ICR 寄存器中的 CTSCF 位，此位由软件清零。

如果 LPUART_CR3 寄存器中 CTSIE=1，则会生成中断。

0：CTS 状态线上未发生变化

1：CTS 状态线上发生变化

注：　如果不支持硬件流控制功能，该位保留并由硬件强制清零。

位 8 保留，必须保持复位值。

位 7 **TXE**：发送数据寄存器为空 (Transmit data register empty)

当 LPUART_TDR 寄存器的内容已传输到移位寄存器时，此位由硬件置 1。通过对 LPUART_TDR 寄存器执行写入操作将该位清零。

如果 LPUART_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。

0：数据未传输到移位寄存器

1：数据传输到移位寄存器

注： 单缓冲区发送期间使用该位。

位 6 **TC**：发送完成 (Transmission complete)

如果已完成对包含数据的帧的发送并且 TXE 置 1，则此位由硬件置 1。如果 LPUART_CR1 寄存器中 TCIE = 1，则会生成中断。通过向 LPUART_ICR 寄存器中的 TCCF 写入 1 或向 LPUART_TDR 寄存器执行写操作，此位由软件清零。

如果 LPUART_CR1 寄存器中 TCIE = 1，则会生成中断。

0：传送未完成

1：传送已完成

注： 如果 TE 位复位且无任何发送正在进行，TC 位会立即置 1。

位 5 **RXNE**：读取数据寄存器不为空 (Read data register not empty)

当 RDR 移位寄存器的内容已传输到 LPUART_RDR 寄存器时，此位由硬件置 1。通过对 LPUART_RDR 寄存器执行读入操作将该位清零。也可以通过将 LPUART_RQR 寄存器中的 RXFRQ 位置 1 将 RXNE 标志位清零。

如果 LPUART_CR1 寄存器中 RXNEIE = 1，则会生成中断。

0：未接收到数据

1：已准备好读取接收到的数据

位 4 **IDLE**：检测到空闲线路 (IDLE line detected)

检测到空闲线路时，此位由硬件置 1。如果 LPUART_CR1 寄存器中 IDLEIE=1，则会生成中断。通过向 LPUART_ICR 寄存器中的 IDLECF 写入 1，此位由软件清零。

0：未检测到空闲线路

1：检测到空闲线路

注： 直到 RXNE 位已置 1 时（即，当出现新的空闲线路时）IDLE 位才会被再次置 1。

使能静默模式 (MME=1) 后，如果 LPUART 未静默 (RWU=0)，则 IDLE 置 1，无论是否通过 WAKE 位选择了静默模式。如果 RWU=1，IDLE 不置 1。

位 3 **ORE**：溢出错误 (Overrun error)

在 RXNE = 1 的情况下，当移位寄存器中当前正在接收的数据准备好传输到 RDR 寄存器时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 ORECF 写入 1，此位由软件清零。

如果 LPUART_CR1 寄存器中 RXNEIE=1 或 EIE = 1，则会生成中断。

0：无溢出错误

1：检测到溢出错误

注： 当此位置 1 时，RDR 寄存器的内容不会丢失，但移位寄存器会被覆盖。EIE 位置 1 后，如果在多缓冲区通信中 ORE 标志置 1，则会生成中断。

LPUART_CR3 寄存器中的 OVRDIS 位置 1 时，此位将被永久强制清零（无上溢检测）。

位 2 **NF**：START 位噪声检测标志 (START bit Noise detection flag)

当在接收的帧的 START 位上检测到噪声时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 NFCF 写入 1，此位由软件清零。

0：未检测到噪声

1：检测到噪声

注：　该位不会生成中断，因为该位出现的时间与本身生成中断的 RXNE 位出现的时间相同。EIE 位置 1 后，如果在多缓冲区通信中 NF 标志置 1，则会生成中断。

位 1 **FE**：帧错误 (Framing error)

当检测到去同步化、过度的噪声或中断字符时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 FECF 写入 1，此位由软件清零。

如果 LPUART_CR1 寄存器中 EIE = 1，则会生成中断。

0：未检测到帧错误

1：检测到帧错误或中断字符

位 0 **PE**：奇偶校验错误 (Parity error)

当在接收器模式下发生奇偶校验错误时，此位由硬件置 1。通过向 LPUART_ICR 寄存器中的 PECF 写入 1，此位由软件清零。

如果 LPUART_CR1 寄存器中 PEIE = 1，则会生成中断。

0：无奇偶校验错误

1：奇偶校验错误

30.7.7 中断标志清零寄存器 (LPUART_ICR)

Interrupt flag clear register

偏移地址：0x20

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
						w			w		w	w	w	w	w

位 31:21 保留，必须保持复位值。

位 20 **WUCF**：从停止模式唤醒清零标志 (Wakeup from Stop mode clear flag)

将 1 写入此位时，LPUART_ISR 寄存器中 WUF 标志将清零。

注：　如果 LPUART 不支持从停止模式唤醒功能，则此位保留并由硬件强制清零。

位 19:18 保留，必须保持复位值。

位 17 **CMCF**：字符匹配清零标志 (Character match clear flag)

将 1 写入此位时，LPUART_ISR 寄存器中 CMF 标志将清零。

位 16:10 保留，必须保持复位值。

位 9 **CTSCF**：CTS 清零标志 (CTS clear flag)

将 1 写入此位时，LPUART_ISR 寄存器中 CTSIF 标志将清零。

位 8:7 保留，必须保持复位值。

位 6 **TCCF**：发送完成清零标志 (Transmission complete clear flag)

将 1 写入此位时，LPUART_ISR 寄存器中 TC 标志将清零。

位 5 保留，必须保持复位值。

位 4 **IDLECF**：检测到空闲线路清零标志 (Idle line detected clear flag)
将 1 写入此位时，LPUART_ISR 寄存器中 IDLE 标志将清零。

位 3 **ORECF**：上溢错误清零标志 (Overrun error clear flag)
将 1 写入此位时，LPUART_ISR 寄存器中 ORE 标志将清零。

位 2 **NCF**：检测到噪声清零标志 (Noise detected clear flag)
将 1 写入此位时，LPUART_ISR 寄存器中 NF 标志将清零。

位 1 **FECF**：帧错误清零标志 (Framing error clear flag)
将 1 写入此位时，LPUART_ISR 寄存器中 FE 标志将清零。

位 0 **PECF**：奇偶校验错误清零标志 (Parity error clear flag)
将 1 写入此位时，LPUART_ISR 寄存器中 PE 标志将清零。

30.7.8 接收数据寄存器 (LPUART_RDR)

Receive data register

偏移地址：0x24

复位值：未定义

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

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

位 31:9 保留，必须保持复位值。

位 8:0 **RDR[8:0]**：接收数据值 (Receive data value)
包含接收到的数据字符。
RDR 寄存器在输入移位寄存器和内部总线之间提供了并行接口（请参见*图 235*）。
在使能奇偶校验位的情况下进行接收时，从 MSB 位中读取的值为接收到的奇偶校验位。

30.7.9 发送数据寄存器 (LPUART_TDR)

Transmit data register

偏移地址：0x28

复位值：未定义

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

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

位 31:9 保留，必须保持复位值。

位 8:0 **TDR[8:0]**：发送数据值 (Transmit data value)

包含要发送的数据字符。

TDR 寄存器在内部总线和输出移位寄存器之间提供了并行接口（请参见*图 235*）。

在使能奇偶校验位的情况下（LPUART_CR1 寄存器中的 PCE 位被置 1）进行发送时，由于 MSB 的写入值（位 7 或位 8，具体取决于数据长度）会被奇偶校验位所取代，因此该值不起任何作用。

注： 只能在 TXE=1 时写入此寄存器。

30.7.10 LPUART 寄存器映射

下表提供了 LPUART 寄存器映射和复位值。

表 137. LPUART 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1	Res.	Res.	Res.	M1	Res.	Res.	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	Res.	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value				0			0	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	LPUART_CR2	ADD[7:4]				ADD[3:0]				Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS [1:0]		Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																													0	0	0	
0x1C	LPUART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	1	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	Res.	ORECF	NCF	FECF	PECF
	Reset value												0			0								0			0			0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

31 串行外设接口/集成电路内置音频总线 (SPI/I2S)

31.1 简介

SPI/I²S 接口可用于使用 SPI 协议或 I^2S 音频协议与外部器件进行通信。SPI 或 I^2S 模式可通过软件进行选择。器件复位后默认选择 SPI 模式。

串行外设接口 (SPI) 协议支持与外部器件进行半双工、全双工和单工同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟 (SCK)。该接口还能够在多主模式配置下工作。

集成电路内置音频总线 (I^2S) 也是同步串行通信接口。它能够在从模式或主模式下作为接收器或发送器工作

它可满足四种不同音频标准的要求，包括 Philips I^2S 标准、MSB 和 LSB 对齐标准以及 PCM 标准。

31.1.1 SPI 主要特性

- 主模式或从模式操作
- 基于三条线的全双工同步传输
- 基于双线的半双工同步传输，其中一条可作为双向数据线
- 基于双线的单工同步传输，其中一条可作为单向数据线
- 8 位到 16 位传输帧格式选择
- 多主模式功能
- 8 个主模式波特率预分频器，可达 $f_{PCLK}/2$。
- 从模式频率可达 $f_{PCLK}/2$。
- 对于主模式和从模式都可通过硬件或软件进行 NSS 管理：动态切换主/从操作
- 可编程的时钟极性和相位
- 可编程的数据顺序，最先移位 MSB 或 LSB
- 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SPI Motorola模式
- 用于确保可靠通信的硬件 CRC 功能：
  - 在发送模式下可将 CRC 值作为最后一个字节发送
  - 根据收到的最后一个字节自动进行 CRC 错误校验
- 可触发中断的主模式故障和上溢标志
- CRC 错误标志
- 具有 DMA 功能的单字节/字收发缓冲器：发送和接收请求

31.1.2 SPI 扩展特性

- 支持SPI TI模式

31.1.3 I2S 功能

- 半双工通信（仅作为发送器或接收器）
- 主模式或从模式操作
- 8 位可编程线性预分频器，可实现精确的音频采样频率（从 8 kHz 到 192 kHz）
- 数据格式可以是 16 位、24 位或 32 位
- 数据包帧由音频通道固定为 16 位（可容纳 16 位数据帧）或 32 位（可容纳 16 位、24 位、32 位数据帧）
- 可编程的时钟极性（就绪时的电平状态）
- 从发送模式下的下溢标志、接收模式下的上溢标志（主模式和从模式），以及接收和发送模式下的帧错误标志（仅从模式）
- 发送和接收使用同一个 16 位数据寄存器
- 支持的 I^2S 协议：
  - I^2S Philips 标准
  - MSB 对齐标准（左对齐）
  - LSB 对齐标准（右对齐）
  - PCM 标准（在 16 位通道帧或扩展为 32 位通道帧的 16 位数据帧上进行短帧和长帧同步）
- 数据方向始终为 MSB 在前
- 用于发送和接收的 DMA 功能（16 位宽）
- 可输出主时钟以驱动外部音频元件。比率固定为 256 × F_S（其中 F_S 为音频采样频率）

31.2 SPI/I2S 实现

本手册介绍了 SPI1 和 SPI2 中实现的所有特性。

表 138. STM32L0x3 SPI实现

SPI 特性(1)	SPI1	SPI2
硬件 CRC 计算	X	X
I2S 模式	-	X
TI 模式	X	X

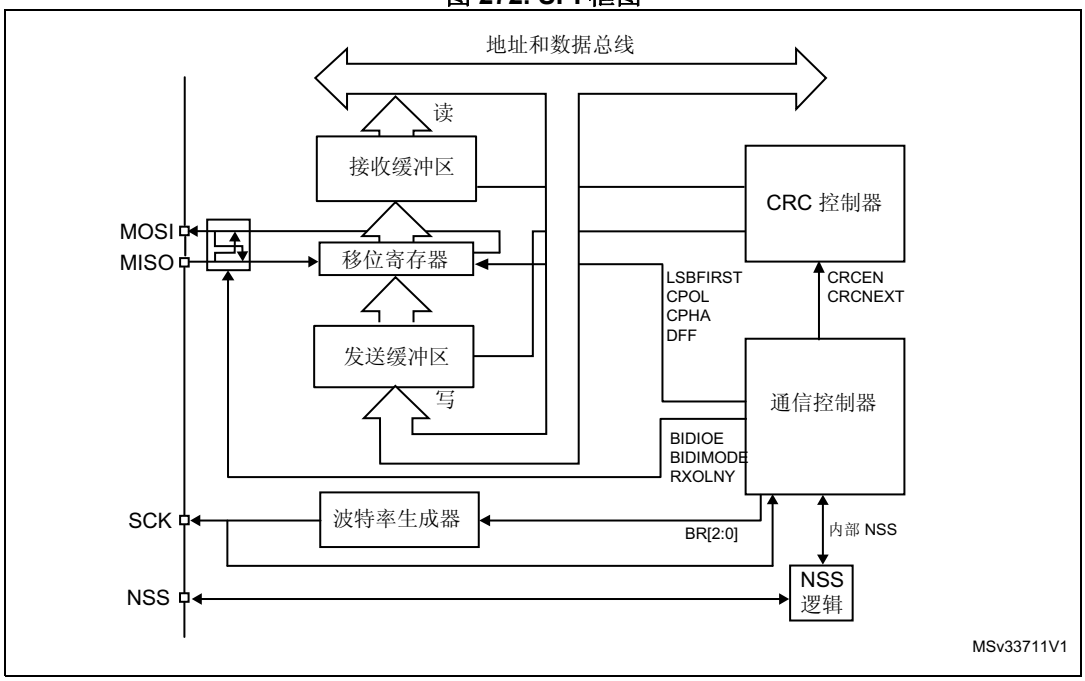
1. X = 支持。

31.3 SPI 功能说明

31.3.1 概述

SPI 支持在 MCU 与外部器件之间进行同步串行通信。应用软件可通过轮询状态标志或使用专用 SPI 中断对通信进行管理。SPI 的主要组件及其交互方式如以下框图所示*图 272*。

图 272. SPI 框图



四个 I/O 引脚专用于与外部器件进行 SPI 通信。

- **MISO**：主输入 / 从输出数据。通常情况下，此引脚用于在从模式下发送数据和在主模式下接收数据。
- **MOSI**：主输出 / 从输入数据。通常情况下，此引脚用于在主模式下发送数据和在从模式下接收数据。
- **SCK**：SPI 主器件的串行时钟输出引脚以及 SPI 从器件的串行时钟输入引脚。
- **NSS**：从器件选择引脚。根据 SPI 和 NSS 设置，该引脚可用于：
  - 选择单个从器件以进行通信
  - 同步数据帧或
  - 检测多个主器件之间是否存在冲突

  详细信息，请参见*第 31.3.4 节：多主器件通信*。

SPI 总线支持一个主器件与一个或多个从器件之间进行通信。该总线至少由两条线构成——一条用于时钟信号，另一条用于同步数据传输。其它信号可以根据 SPI 节点间的数据交换及其从器件选择信号管理进行添加。

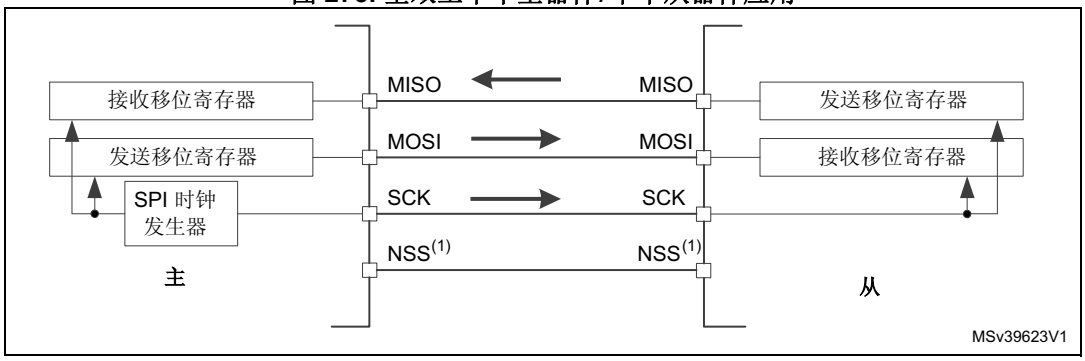
31.3.2 一个主器件和一个从器件之间的通信

SPI 支持 MCU 基于目标器件和应用要求使用不同的配置进行通信。这些配置使用 2 条或 3 条线（通过软件 NSS 管理），也可以使用 3 条或 4 条线（通过硬件 NSS 管理）。通信始终由主器件发起。

全双工通信

默认情况下，SPI 配置为进行全双工通信。在这种配置下，主器件和从器件的移位寄存器通过 MOSI 和 MISO 引脚之间的两条单向线连接。在 SPI 通信过程中，数据随主器件提供的 SCK 时钟边沿同步移位。主器件通过 MOSI 线将待发送的数据发送给从器件，通过 MISO 线从从器件接收数据。当数据帧传输完成时（所有位均移出），主器件和从器件之间即完成信息交换。

图 273. 全双工单个主器件/单个从器件应用

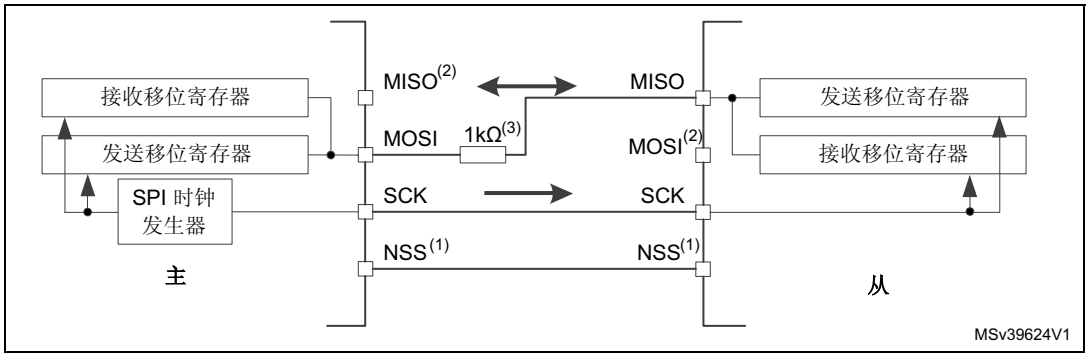


1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见*第 31.3.5 节：从器件选择 (NSS) 引脚管理*。

半双工通信

通过将 SPIx_CR1 寄存器的 BIDIMODE 位置 1，SPI 可采用半双工模式进行通信。在这种配置下，使用一条交叉连接线将主器件和从器件的移位寄存器连接起来。在此通信过程中，数据随 SCK 时钟边沿在移位寄存器之间进行移位，传输方向由主器件和从器件通过各自 SPIx_CR1 寄存器中的 BDIOE 位进行选择。在这种配置下，主器件的 MISO 引脚和从器件的 MOSI 引脚空闲，可在其它应用中用作 GPIO。

图 274. 半双工单个主器件/单个从器件应用



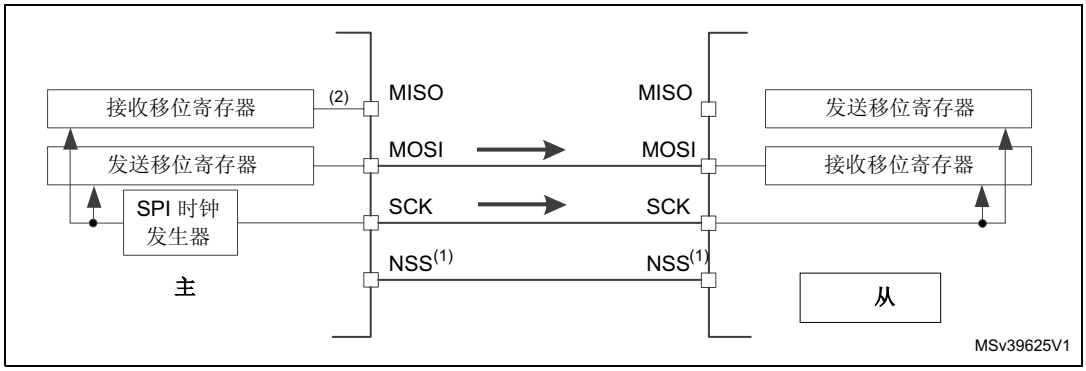
1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见*第 31.3.5 节：从器件选择 (NSS) 引脚管理*。
2. 在这种配置下，主器件的 MISO 引脚和从器件的 MOSI 引脚可用作 GPIO。
3. 当以双向模式工作的两个节点间的通信方向不是同步变化时，会出现临界情况，新发送器访问共用数据线，而前一个发送器仍保持线路上的相反值（值取决于 SPI 配置和通信数据）。两个节点会出现冲突，在共用线上短暂提供相反的输出电平，直到下一个节点也相应地改变其方向设置。建议此模式下在 MISO 和 MOSI 引脚之间插入串行电阻以在这种情况下保护输出并限制电流在二者之间流过。

单工通信

通过 SPIx_CR2 寄存器中的 RXONLY 位将 SPI 设置为只发送模式或只接收模式，可使 SPI 以单工模式进行通信。在这种配置下，仅使用一条线在主器件和从器件的移位寄存器之间进行传输。其余 MISO 和 MOSI 引脚对不用于通信，可用作标准 GPIO。

- **只发送模式 (RXONLY=0)**：配置设置与全双工设置相同。应用必须忽略在未使用的输入引脚上捕获的信息。该引脚可以用作标准 GPIO。
- **只接收模式 (RXONLY=1)**：应用可通过将 RXONLY 位置 1 来禁止 SPI 输出功能。在从器件配置下，MISO 输出被禁止，该引脚可用作 GPIO。当从器件选择信号有效时，从器件继续从 MOSI 引脚接收数据（请参见 *31.3.4：多主器件通信*）。基于数据缓冲区的配置产生接收数据事件。在主器件配置下，MOSI 输出被禁止，该引脚可用作 GPIO。只要 SPI 处于使能状态，便不断生成时钟信号。停止时钟的唯一方式是将 RXONLY 位或 SPE 位清零，直至来自 MISO 引脚的传入模式结束，然后基于相应配置填充数据缓冲区结构。

图 275. 单工单个主器件/单个从器件应用（主器件为只发送模式/从器件为只接收模式）



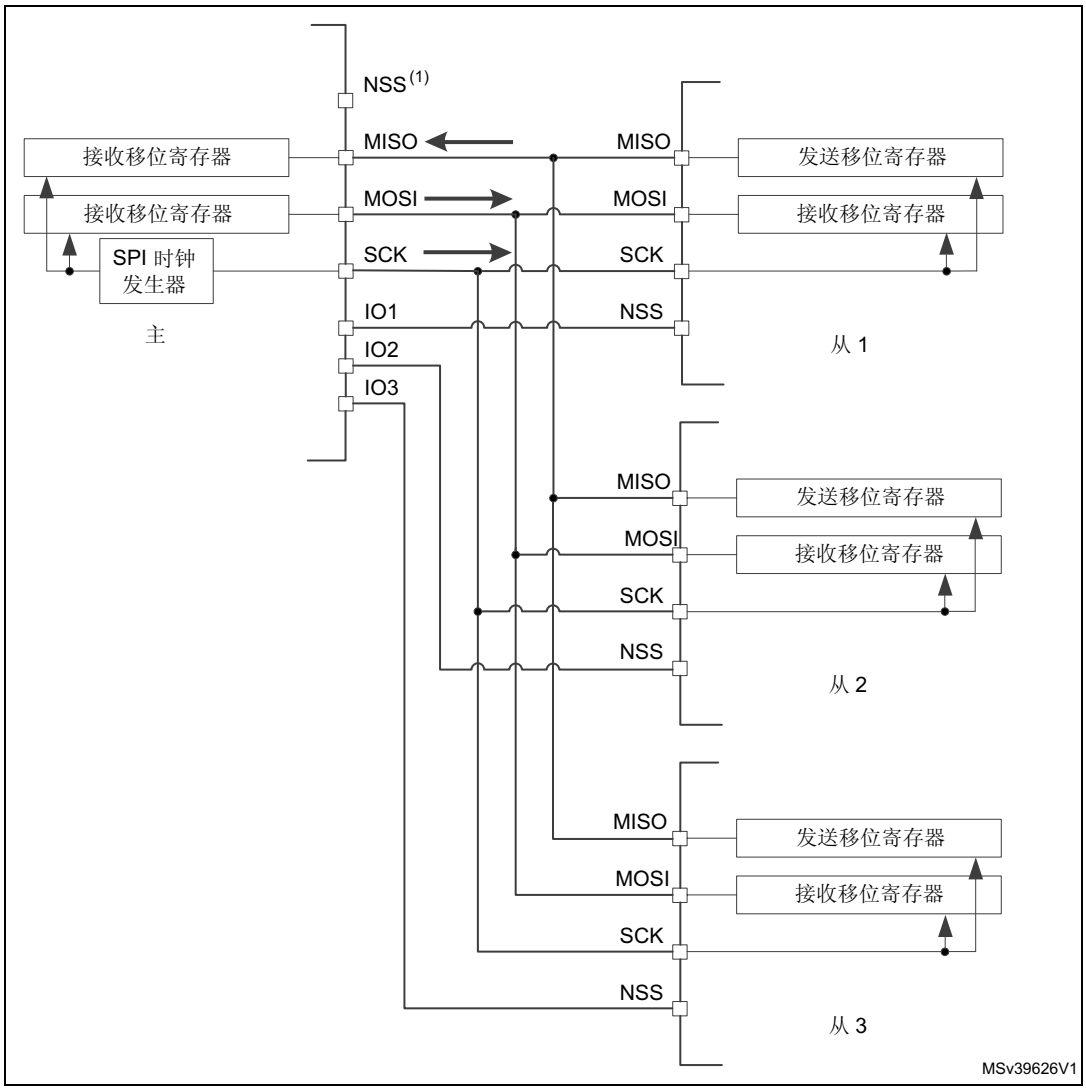
1. NSS 引脚可用于在主器件和从器件之间提供硬件控制流。外设也可选择不使用这些引脚。之后，必须在内部为主器件和从器件处理硬件控制流。有关更多详细信息，请参见*第 31.3.5 节：从器件选择 (NSS) 引脚管理*。
2. 在发送器 Rx 移位寄存器的输入上捕获意外输入信息。标准只发送模式下必须忽略与发送器接收流相关的所有事件（例如 OVF 标志）。
3. 在这种配置下，两个 MISO 引脚均可用作 GPIO。

注：任何单工通信均可以通过半双工通信的一种变型来替换，该变型中设置的数据传输方向不变（在 BDIO 位保持不变的同时，双向模式处于使能状态）。

31.3.3 标准多从器件通信

在具有两个或多个独立从器件的配置下，主器件使用 GPIO 引脚来管理每个从器件的片选线（请参见*图 276*）。主器件必须通过拉低与从器件 NSS 输入相连的 GPIO 的电平来单独选择一个从器件。执行该操作后，便建立了标准主器件与专用从器件之间的通信。

图 276. 主器件和三个独立的从器件



1. 此配置的主器件侧不使用 NSS 引脚。该引脚必须在内部管理 (SSM=1，SSI=1) 以避免任何 MODF 错误。
2. 由于从器件的 MISO 引脚连在一起，所有从器件 MISO 引脚的 GPIO 配置必须设置为复用功能开漏（请参见*第 182 页上的第 7.3.7 节：时钟中断清零寄存器 (RCC_CICR)*）。

31.3.4 多主器件通信

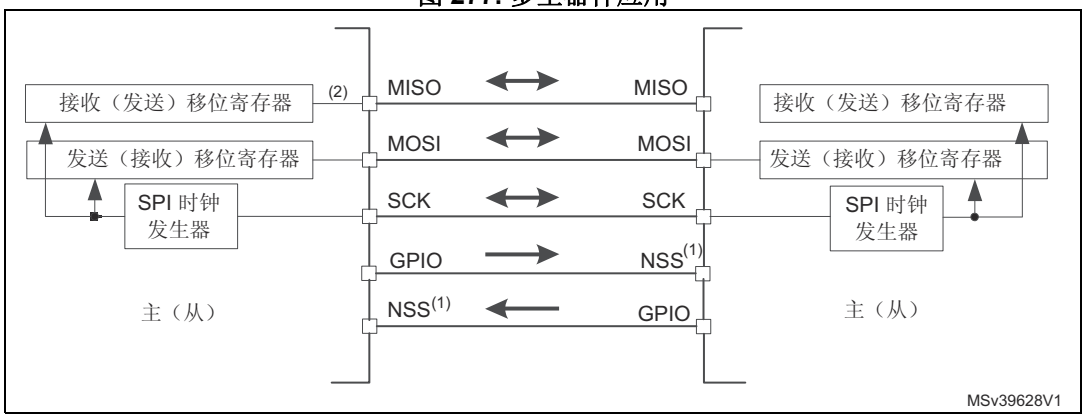
如果 SPI 总线未用于多主功能，用户可使用内置功能来检测尝试同时控制总线的两个节点间是否存在潜在冲突。对于该检测，NSS 引脚配置为硬件输入模式。

由于此时只有一个结点可将其输出施加到公用数据线上，因此无法连接超过两个以此模式工作的 SPI 节点。

当节点无效时，默认情况下均保持从模式。一旦一个节点要接管对总线的控制，它会将自身切换到主模式，然后通过专用 GPIO 引脚向其它节点的从器件选择输入施加有效电平。会话完成后，有效的从器件选择信号将被释放，控制总线的节点会短暂切换回被动从模式，等待下一个会话开始。

如果两个节点同时发出各自的控制请求，则会出现总线冲突（请参见模式故障 MODF 事件）。随后，用户可应用某个简单的仲裁过程（例如，在两个节点上施加不同的预定义超时来推迟下一个尝试）。

图 277. 多主器件应用



1. 在两个节点上，NSS 引脚配置为硬件输入模式。当无效节点配置为从器件时，其有效电平将使能 MISO 线输出控制。

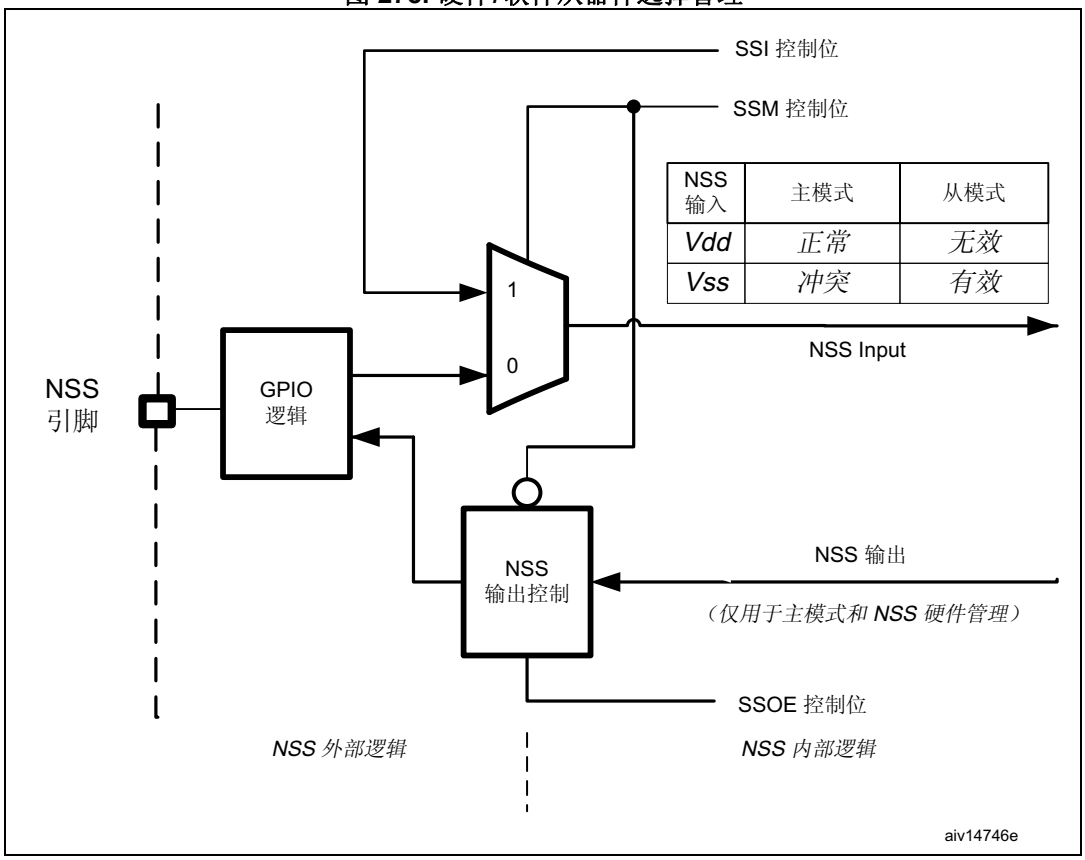
31.3.5 从器件选择 (NSS) 引脚管理

在从模式下，NSS 用作标准的“片选”输入，使从器件与主器件进行通信。在主模式下，NSS 可用作输出或输入。用作输入时，可防止多主模式总线冲突；用作输出时，可驱动单个从器件的从器件选择信号。

可以使用 SPIx_CR1 寄存器中的 SSM 位设置硬件或软件从器件选择管理：

- **软件 NSS 管理 (SSM = 1)**：在这种配置下，由 SPIx_CR1 寄存器中的 SSI 位的值内部驱动从器件选择信息。外部 NSS 引脚空闲，可供其它应用使用。
- **硬件 NSS 管理 (SSM = 0)**：在这种情况下，可行的配置有两种：所用配置取决于 NSS 输出配置（SPIx_CR1 寄存器中的 SSOE 位）。
  - **NSS 输出使能（SSM=0 且 SSOE = 1）**：仅在将 MCU 设置为主器件时才使用该配置。NSS 引脚由硬件管理。只要在主模式下使能 SPI (SPE=1)，NSS 信号便会被驱动为低电平，并且会一直保持低电平状态，直至禁止 SPI (SPE =0)。
  - **NSS 输出禁止（SSM=0 且 SSOE = 0）**：如果微控制器在总线上用作主器件，此配置可实现多主模式功能。如果在该模式下将 NSS 引脚拉至低电平，SPI 将进入主模式故障状态，器件将在从模式下自动进行重新配置。在从模式下，NSS 引脚用作标准的“片选”输入，当 NSS 线为低电平时将选择从器件。

图 278. 硬件/软件从器件选择管理



31.3.6 通信格式

SPI 通信过程中，将同时执行接收和发送操作。串行时钟 (SCK) 对数据线上的信息的移位和采样进行同步。通信格式取决于时钟相位、时钟极性和数据帧格式。为了能够在彼此间进行通信，主器件和从器件必须遵循相同的通信格式。

时钟相位和极性控制

通过 SPIx_CR1 寄存器中的 CPOL 和 CPHA 位，可以用软件选择四种可能的时序关系。CPOL（时钟极性）位控制不传输任何数据时时钟的空闲状态值。此位对主器件和从 器件都有作用。如果复位 CPOL，SCK 引脚在空闲状态处于低电平。如果将 CPOL 置 1，SCK 引脚在空闲状态处于高电平。

如果将 CPHA 位置 1，则会在 SCK 引脚的第二个边沿捕获传输的第一个数据位（如果复位 CPOL 位，则为下降沿；如果将 CPOL 位置 1，则为上升沿）。即，在每次出现该时钟边沿时锁存数据。如果将 CPHA 位复位，则会在 SCK 引脚的第一个边沿捕获传输的第一个数据位（如果将 CPOL 位置 1，则为下降沿；如果将 CPOL 位复位，则为上升沿）。即，在每次出现该时钟边沿时锁存数据。

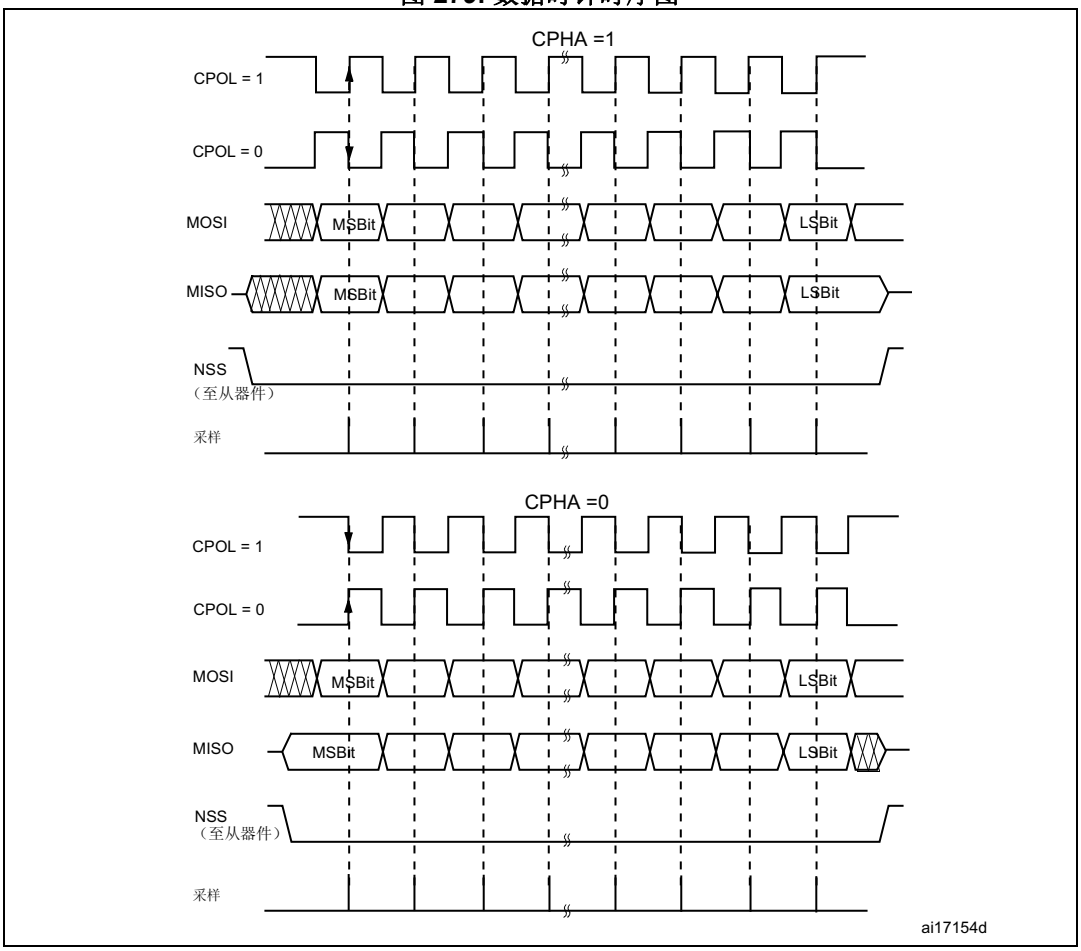
CPOL（时钟极性）和 CPHA（时钟相位）位的组合用于选择数据捕获时钟边沿。

图 279 给出了在 CPHA 和 CPOL 位的四种组合下的 SPI 全双工传输。

注： *在切换 CPOL/CPHA 位之前，必须通过复位 SPE 位来禁止 SPI。*

SCK 的空闲状态必须与 SPIx_CR1 寄存器中选择的极性相对应（如果 CPOL=1，则上拉 SCK；如果 CPOL=0，则下拉 SCK）。

图 279. 数据时钟时序图



注： *数据位的顺序取决于 LSBFIRST 位的设置。*

数据帧格式

SPI 移位寄存器可设置为以 MSB 在前或 LSB 在前的方式移出数据，具体取决于 LSBFIRST 位的值。每个数据帧的长度均为 8 位或 16 位，具体取决于使用 SPI_CR1 寄存器中的 DFF 位编程的数据长度。所选的数据帧格式适用于发送和接收。

31.3.7 SPI配置

主器件和从器件的配置步骤几乎相同。对于具体的模式设置，请遵从相应章节的内容。若要对标准通信进行初始化，请执行以下步骤：

1. 对相应的 GPIO 寄存器执行写操作：将 MOSI、MISO 和 SCK 引脚配置为 GPIO。
2. 对 SPI_CR1 寄存器执行写操作：
   a) 通过 BR[2:0] 位配置串行时钟波特率（*注：*3）。
   b) 配置 CPOL 位和 CPHA 位组合，从数据与时钟四组时序定义中选择一种定义。（*注：*2）。
   c) 通过配置 RXONLY 或 BIDIMODE 和 BIDIOE 来选择单工或半双工模式（RXONLY 和 BIDIMODE 不可同时置 1）。
   d) 配置 LSBFIRST 位以定义帧格式（*注：*2）。
   e) 如果需要 CRC，请配置 CRCEN 和 CRCEN 位（SCK 时钟信号处于空闲状态时）。
   f) 配置 SSM 和 SSI（*注：*2）。
   g) 配置 MSTR 位（在多主模式 NSS 配置下，如果主器件配置为防止发生 MODF 错误，则应避免 NSS 上出现状态冲突）。
   h) 将 DFF 位置 1 以配置数据帧格式（8 位或 16 位）。
3. 对 SPI_CR2 寄存器执行写操作：
   a) 配置 SSOE（*注：*1 和 2）。
   b) 如果需要 TI 协议，请将 FRF 位置 1。
4. 对 SPI_CRCPR 寄存器执行写操作：需要时配置 CRC 多项式。
5. 对相应的 DMA 寄存器执行写操作：如果使用 DMA 数据流，请在 DMA 寄存器中配置 SPI Tx 和 Rx 专用的 DMA 数据流。

注： *(1) 从模式下无需此步骤。*

(2) TI 模式下无需此步骤。

(3) 从模式下无需此步骤，但从器件在 TI 模式下工作时除外。

有关代码示例，请参见 *A.19.1：SPI 主器件配置代码示例*和 *A.19.2：SPI 从器件配置代码示例*。

31.3.8 使能 SPI 的步骤

建议在主器件发送时钟前使能 SPI 从器件。否则，数据传输可能会不正常。从器件的数据寄存器必须包含待发送的数据才能开始与主器件通信（在通信时钟的第一个边沿；如果时钟信号连续，则是在正在进行的通信结束前）。使能 SPI 从器件前，SCK 信号必须稳定为所选极性对应的空闲状态电平。

在全双工（或任何只发送模式）下，使能 SPI 后，主器件开始通信，待发送的数据将写入 Tx 缓冲区。

在任何主器件只接收模式（RXONLY=1 或 BIDIMODE=1 且 BIDIOE=0）下，使能 SPI 后，主器件立即开始通信且时钟立即开始运行。

从器件接收到来自主器件的正确时钟信号时，它将开始通信。在 SPI 主器件启动传输前，从软件必须写入待发送的数据。

有关如何处理 DMA 的详细信息，请参见*第 31.3.11 节：使用 DMA（直接存储器寻址）进行通信*。

31.3.9 数据发送和接收过程

接收和发送缓冲区

在接收过程中，数据收到后，先存储到内部接收缓冲区中；而在发送过程中，先将数据存储到内部发送缓冲区中，然后发送数据。对 SPI_DR 寄存器的读访问将返回接收缓冲值，而对 SPI_DR 寄存器的写访问会将写入的数据存储到发送缓冲区中。

发送缓冲区处理

在第一个位传输期间，数据帧从发送缓冲区加载到移位寄存器。各个位随后从移位寄存器以串行方式移出到专用输出引脚，具体取决于 LSBFIRST 位设置。当数据从发送缓冲区传送到移位寄存器时，TXE 标志（发送缓冲区为空）置 1。该标志表示内部发送缓冲区已准备好加载接下来的数据。如果 SPI_CR2 寄存器中的 TXEIE 位置 1，可产生中断。通过对 SPI_DR 寄存器执行写操作将 TXE 位清零。

如果在前一次帧传输仍在进行时将要发送的下一个数据存储到发送缓冲区，则可保持连续的发送流。如果软件在 TXE 未置 1 时写入发送缓冲区，则等待发送的数据将被覆盖。

接收缓冲区处理

将数据从移位寄存器传输到接收缓冲区时，RXNE 标志（接收缓冲区非空）会在最后一个采样时钟边沿置 1。它表示已准备好从 SPI_DR 寄存器中读取数据。如果 SPI_CR2 寄存器中的 RXNEIE 位置 1，可产生中断。读取 SPI_DR 寄存器即会使 RXNE 位清零。

如果器件未将因发送前一个数据字节而产生的 RXNE 位清零，则在下一个值缓冲时会产生溢出条件。OVR 位置 1 并在 ERRIE 位置 1 时生成一个中断。

另一种管理数据交换的方式是使用 DMA（请参见*第 8.2 节：CRS 主要特性*）。

序列处理

当前数据帧传输正在进行时，BSY 位将置 1。当时钟信号连续运行时，BSY 标志在主器件侧的两个数据帧间保持置 1。不过，在从器件侧，它将在每个数据帧传输之间变为 0 并持续至少一个 SPI 时钟周期。

对于某些配置，可以在最后一次数据传输期间使用 BSY 标志来等待传输完成。

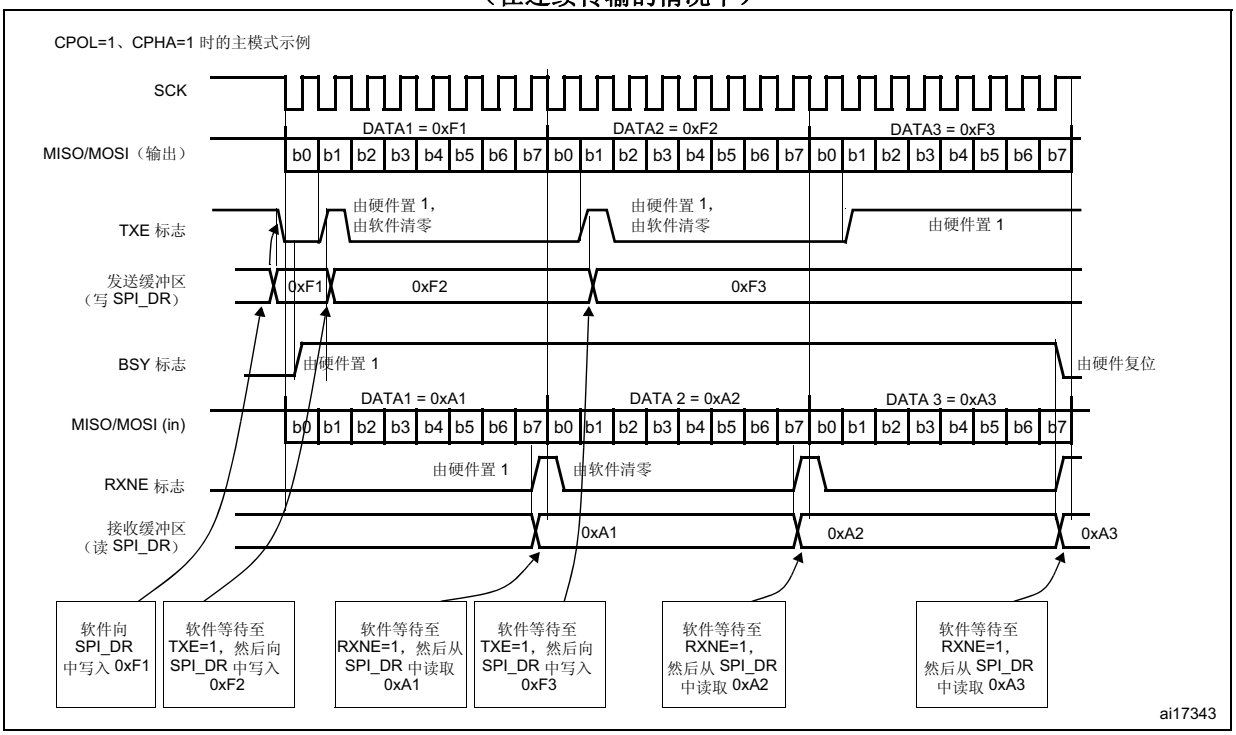
当主器件侧配置只接收模式时，无论是半双工（BIDIMODE=1 且 BIDIOE=0）还是单工配置（BIDIMODE=0 且 RXONLY=1），主器件都会在 SPI 使能后立即启动接收序列。随后，时钟信号由主器件提供，且直至主器件禁止 SPI 或只接收模式才会停止。在此之前，主器件会连续接收数据帧。

当主器件能够以连续模式（SCK 信号连续）提供所有交互时，任何时候都必须根据从器件能力来处理数据流及其内容。必要时，主器件必须降低通信速度，提供较慢的时钟或带有足够延时的单独帧或数据段。请注意，在 SPI 模式下工作时不存在从器件的下溢错误信号，来自从器件的数据始终由主器件处理，即使从器件无法及时正确地准备数据也是如此。从器件最好使用 DMA，尤其是数据帧较短而总线速率较高时。

在多从器件系统中，每个序列必须通过 NSS 脉冲进行控制，从而只选择其中一个从器件进行通信。在单个从器件系统中，不必使用 NSS 来控制从器件。但是，可使用 NSS 脉冲将从器件与每个数据传输序列的开始同步。NSS 可通过软件或硬件进行管理（请参见*第 31.3.4 节：多主器件通信*）。

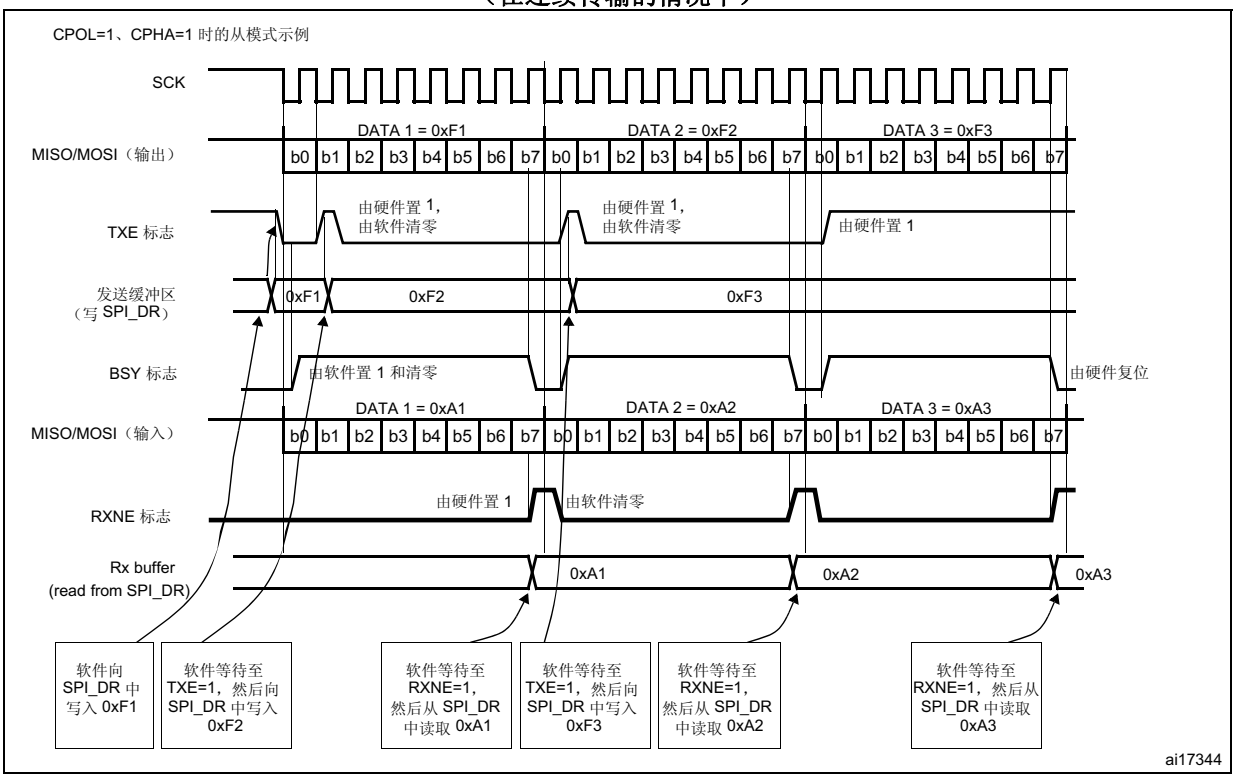
有关主器件/全双工和从器件/全双工模式下的连续传输的说明，请参见*图 280* 和*图 281*。

图 280. 主/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 时序（在连续传输的情况下）



有关代码示例，请参见 *A.19.3：SPI 全双工通信代码示例*。

图 281. 从器件/全双工模式（BIDIMODE=0 且 RXONLY=0）下的 TXE/RXNE/BSY 时序（在连续传输的情况下）



31.3.10 禁止 SPI 的步骤

当禁止 SPI 时，必须按照本段中介绍的禁止步骤进行操作。当外设时钟停止时，在系统进入低功耗模式前做到这一点是十分重要的。否则会损坏正在进行的交互。在某些模式下，禁止步骤是停止所进行的连续通信的唯一方式。

当处于全双工或只发送模式下的主器件停止提供待发送的数据时，可结束任何交互。在这种情况下，时钟在最后一个数据传输后停止。

标准禁止步骤通过轮询 BSY 状态以及 TXE 标志来检查发送会话是否完全结束。还可以在必须识别正在处理的传输是否结束的特定情况下完成这种检查，例如：

- 当 NSS 信号由任意 GPIO 切换管理且主器件必须为从器件提供 NSS 脉冲结束时，或者
- 最后一个数据帧或 CRC 帧传输仍在外设总线中处理时，来自 DMA 的传输数据流完成。

正确的禁止步骤如下（使用只接收模式时除外）：

1. 等待 RXNE=1 以接收最后的数据。
2. 等待至 TXE=1，然后等待至 BSY=0，再关闭 SPI。
3. 读取接收的数据。

注：在不连续通信期间，在对 SPI_DR 寄存器执行写操作与 BSY 位置 1 之间有 2 个 APB 时钟周期的延迟。因此，写入最后的数据后，必须先等待 TXE 位置 1，然后等待 BSY 位清零。

某些只接收模式的正确禁止步骤如下：

1. 当最后一个数据帧正在处理时，通过在特定时间窗口内禁止 SPI (SPE=0) 来中断接收流。
2. 等待至 BSY=0（最后一个数据帧已处理完）。
3. 读取接收的数据。

注： *要停止连续的接收序列，必须在接收最后一个数据帧时遵循特定的时间窗口。该时间窗口在第一个位采样时开始，在最后一个位的传输开始前结束。*

31.3.11 使用 DMA（直接存储器寻址）进行通信

为了以最大速度工作并且为了促进避免上溢所需的数据寄存器读/写过程，SPI 提供了 DMA 功能，该功能采用了简单的请求/应答协议。

将 SPIx_CR2 寄存器中的使能位 TXE 或 RXNE 置 1 时，将请求 DMA 访问。必须向发送缓冲区和接收缓冲区发出单独的请求。

- 在发送过程中，每次 TXE 位置 1 都会发出 DMA 请求。然后，DMA 将对 SPIx_DR 寄存器执行写操作。
- 在接收过程中，每次 RXNE 位置 1 都会发出 DMA 请求。然后，DMA 将对 SPIx_DR 寄存器执行读操作。

有关 DMA 发送和接收波形的说明，请参见*图 282* 和*图 283*。

当 SPI 仅用于发送数据时，可以只使能 SPI Tx DMA 通道。在这种情况下，OVR 标志会置 1，因为未读取接收的数据。当 SPI 仅用于接收数据时，可以只使能 SPI Rx DMA 通道。

在发送模式下，DMA 写入所有要发送的数据（DMA_ISR 寄存器中的 TCIF 标志置 1）后，可以对 BSY 标志进行监视，以确保 SPI 通信已完成。在禁止 SPI 或进入停止模式前必须执行此步骤，以避免损坏最后一次发送。软件必须首先等待 TXE=1，再等待 BSY=0。

通过 DMA 开始通信时，为防止 DMA 通道管理引发错误事件，必须按顺序执行以下步骤：

1. 如果使用 DMA Rx，通过 SPI_CR2 寄存器中的 RXDMAEN 位来使能 DMA 接收缓冲区。
2. 如果使用数据流，通过 DMA 寄存器来使能 Tx 和 Rx 的 DMA 数据流。
3. 如果使用 DMA Tx，通过 SPI_CR2 寄存器中的 TXDMAEN 位来使能 DMA 发送缓冲区。
4. 通过将 SPE 位置 1 使能 SPI。

有关代码示例，请参见 *A.19.4：SPI 主器件配置（采用 DMA）代码示例*和 *A.19.6：SPI 中断代码示例*。

要关闭通信，必须按顺序执行以下步骤：

1. 如果使能了 DMA，通过 DMA 寄存器来禁止 Tx 和 Rx 的 DMA 数据流。
2. 通过后续 SPI 禁止步骤来禁止 SPI。
3. 如果使用 DMA Tx 和/或 DMA Rx，通过将 SPI_CR2 寄存器中的 TXDMAEN 和 RXDMAEN 位清零来禁止 DMA 发送缓冲区和接收缓冲区。

图 282. 使用 DMA 进行发送

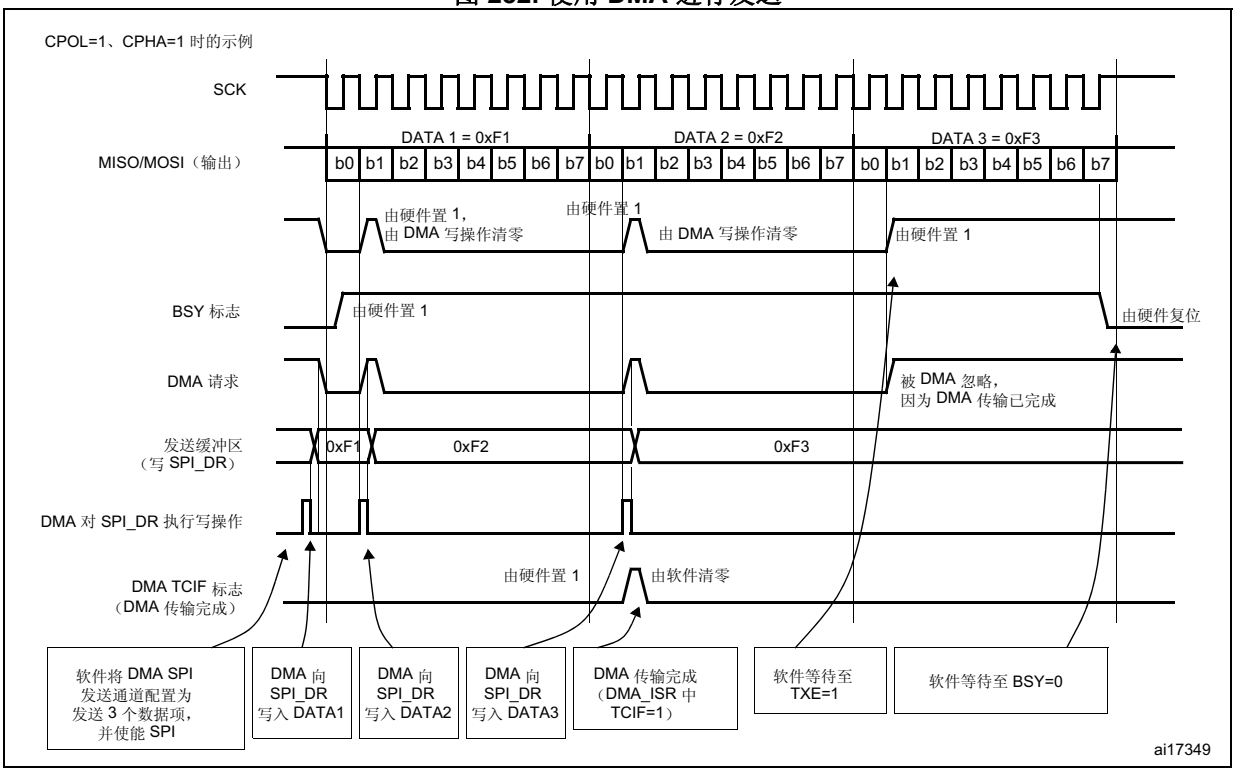
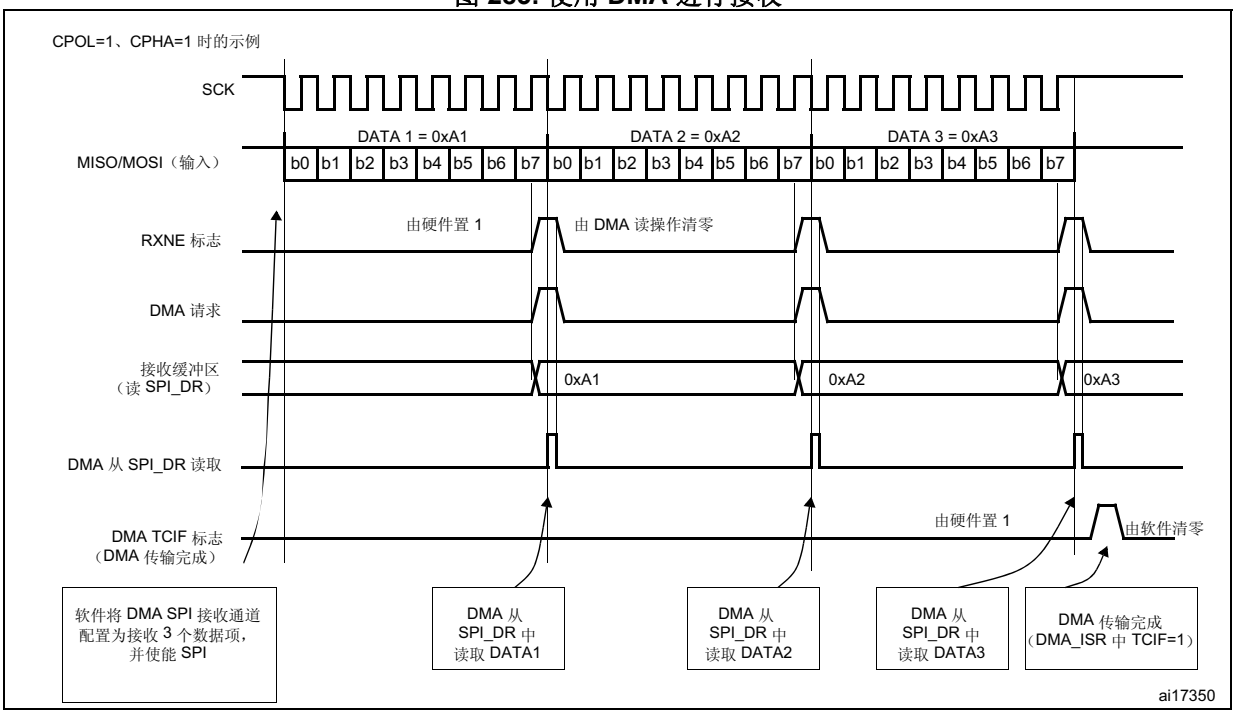


图 283. 使用 DMA 进行接收



31.3.12 SPI 状态标志

应用可通过三种状态标志监视 SPI 总线的状态。

发送缓冲区为空 (TXE)

TXE 标志置 1 时，表示发送缓冲区为空，可以将待发送的下一个数据加载到缓冲区中。对 SPI_DR 寄存器执行写操作时，将清零 TXE 标志。

接收缓冲区非空 (RXNE)

RXNE 标志置 1 时，表示接收缓冲区中存在有效的已接收数据。通过对 SPI_DR 寄存器执行读取操作将该位清零。

忙标志 (BSY)

BSY 标志由硬件置 1 和清零（写入此标志没有任何作用）。

当 BSY 置 1 时，表示 SPI 上正在进行数据传输（SPI 总线繁忙）。在主模式下的双向通信接收模式（MSTR=1 且 BDM=1 且 BDOE=0）有一个例外情况，BSY 标志在接收过程中保持置 0。

在某些模式下，可使用 BSY 标志来检测传输是否结束，从而避免在进入低功耗模式前禁止 SPI 外设时钟时或通过软件管理 NSS 脉冲结束时破坏最后一次传输。

BSY 标志还可用于避免在多主模式系统中发生写冲突。

在以下任意一种条件下，BSY 标志将清零：

- 正确禁止 SPI 时
- 在主模式下检测到故障时（MODF 位置 1）
- 在主模式下，完成了数据发送并且不准备发送任何新数据时
- 在从模式下，BSY 标志在各传输之间的至少一个 SPI 时钟周期内置为“0”时。

注： *建议始终使用 TXE 和 RXNE 标志（而非 BSY 标志）来处理数据发送或接收操作。*

31.3.13 SPI 错误标志

如果以下其中一个错误标志置 1 且已通过将 ERRIE 位置 1 使能了中断，则将生成 SPI 中断。

上溢标志 (OVR)

如果从器件在接收缓冲区中前一个帧的读操作尚未完成时（RXNE 标志置 1）完成下一个数据帧的接收，则会出现溢出条件。

在这种情况下，Rx 缓冲区的内容不会更新为接收的新数据。对 SPI_DR 寄存器执行的读操作将返回先前接收的帧。后续发送的所有其它数据均将丢失。

要将 OVR 位清零，应首先对 SPI_DR 寄存器执行读访问，然后再对 SPI_SR 寄存器执行读访问。

模式故障 (MODF)

当主器件的内部 NSS 信号（NSS 硬件模式下为 NSS 引脚，NSS 软件模式下为 SSI 位）被拉低时，将发生模式故障。这会自动将 MODF 位置 1。主模式故障会在以下几方面影响 SPI 接口：

- 如果 ERRIE 位置 1，MODF 位将置 1，并生成 SPI 中断。
- SPE 位清零。这将关闭器件的所有输出，并禁止 SPI 接口。
- MSTR 位清零，从而强制器件进入从模式。

使用以下软件序列将 MODF 位清零：

1. 在 MODF 位置 1 时，对 SPIx_SR 寄存器执行读或写访问。
2. 然后，对 SPIx_CR1 寄存器执行写操作。

为避免包含多个 MCU 的系统中发生多从模式冲突，必须在 MODF 位清零序列期间将 NSS 引脚拉高。在该清零序列后，可以将 SPE 和 MSTR 位恢复到原始状态。安全起见，硬件不允许在 MODF 位置 1 时将 SPE 和 MSTR 位置 1。在从器件中，MODF 位不可置 1，但由前一次多主模式冲突引起时除外。

CRC 错误 (CRCERR)

当 SPIx_CR1 寄存器中的 CRCEN 位置 1 时，此标志用于验证接收数据的有效性。如果移位寄存器中接收的值与 SPIx_RXCRC 的值不匹配，SPIx_SR 寄存器中的 CRCERR 标志将置 1。该标志由软件清零。

TI 模式帧格式错误 (FRE)

如果 SPI 在从模式下工作，并配置为符合 TI 模式协议，则在通信进行期间出现 NSS 脉冲时，将检测到 TI 模式帧格式错误。出现此错误时，SPIx_SR 寄存器中的 FRE 标志将置 1。发生错误时不会禁止 SPI，但会忽略 NSS 脉冲，并且 SPI 会等待下一个 NSS 脉冲，然后再开始新的传输。由于错误检测可能导致丢失两个数据字节，因此数据可能会损坏。

读取 SPIx_SR 寄存器时，将清零 FRE 标志。如果 ERRIE 位置 1，则检测到 NSS 错误时将生成中断。在这种情况下，由于无法保证数据的一致性，应禁止 SPI，并在重新使能从 SPI 后，由主器件重新发起通信。

31.4 SPI 特性

31.4.1 TI 模式

主模式下的 TI 协议

SPI 接口与 TI 协议兼容。可以使用 SPIx_CR2 寄存器的 FRF 位来配置 SPI，以兼容此协议。

时钟极性和相位都被强制为遵循 TI 协议，和 SPIx_CR1 中的设置无关。NSS 管理也特定于 TI 协议，在这种情况下，无法通过 SPIx_CR1 和 SPIx_CR2 寄存器（SSM、SSI 和 SSOE）来对 NSS 管理进行配置。

在从模式下，SPI 波特率预分频器用于控制在当前传输完成时 MISO 引脚切换为高阻态的时刻（请参见*图 284*）。可以使用任意波特率，因此可以非常灵活地确定此时刻。但是，波特

率通常设置为外部主时钟波特率。MISO 信号变为高阻态的延时 ($t_{release}$) 取决于内部重新同步以及通过 SPIx_CR1 寄存器的 BR[2:0] 位设置的波特率值。具体公式如下：

$$\frac{t_{baud_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud_rate}}{2} + 6 \times t_{pclk}$$

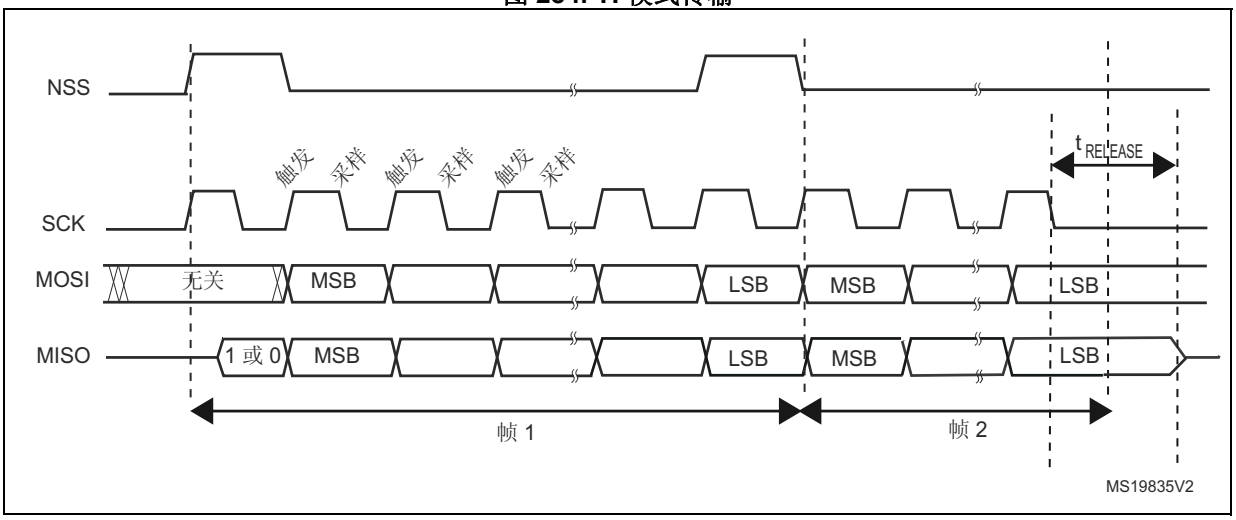
如果从器件在数据帧传输期间检测到错位的 NSS 脉冲，TIFRE 标志将置 1。

此特性不适用于 Motorola SPI 通信（FRF 位为 0）。

注： *要在从器件发送器模式下使用错误中断 (ERRIE = 1) 检测 TI 帧错误，必须通过将 SPI_CR1 寄存器中的 BIDIMODE 和 BIDIOE 置 1 来将 SPI 配置为双线单向模式。当 BIDIMODE 置为 0 时，OVR 将置 1，因为始终不会读取数据寄存器从而始终生成错误中断；而当 BIDIMODE 置 1 时，不会接收数据，也不会将 OVR 置 1。*

图 284: TI 模式传输给出了选择 TI 模式时的 SPI 通信波形。

图 284. TI 模式传输



31.4.2 CRC 计算

为检查发送数据和接收数据的可靠性，使用两个独立的 CRC 计算器（作用于发送和接收数据流）。SPI 提供 CRC8 或 CRC16 计算，具体取决于通过 DFF 位选择的数据格式。CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

CRC 原理

在使能 SPI (SPE = 1) 前，通过将 SPIx_CR1 寄存器中的 CRCEN 位置 1 来使能 CRC 计算。使用值为奇数的可编程多项式对每个位计算 CRC 值。在由 SPIx_CR1 寄存器中的 CPHA 位和 CPOL 位定义的采样时钟边沿进行计算。所计算的 CRC 值在数据块末尾自动进行校验，以及针对由 CPU 或 DMA 管理的传输进行校验。当检测到所接收数据内部计算的 CRC 与发送器发送的 CRC 不匹配时，CRCERR 标志将置 1 以指示数据损坏错误。CRC 计算的正确处理步骤取决于 SPI 配置和所选的传输管理。

注： *多项式值只应为奇数。不支持任何偶数值。*

CPU 管理的 CRC 传输

通信开始后将一直持续到必须发送或接收 SPIx_DR 寄存器中的最后一个数据帧时。之后，SPIx_CR1 寄存器中的 CRCNEXT 位必须置 1，以指示当前处理的数据帧传输后将处理 CRC 帧传输。CRCNEXT 位必须在最后一个数据帧传输结束前置 1。在 CRC 传输期间，CRC 计算将冻结。

与任何其它数据帧一样，接收的 CRC 存储在接收缓冲区内。

CRC 帧的传输通常在数据序列结束时再传输一个数据帧。

接收最后一个 CRC 数据后，将执行自动校验，将接收的值与 SPIx_RXCRC 寄存器中的值进行比较。软件必须校验 SPIx_SR 寄存器中的 CRCERR 标志，以确定数据传输是否损坏。软件通过向 CRCERR 标志写入“0”来将其清零。

接收 CRC 后，CRC 值存储到接收缓冲区中，且必须在 SPIx_DR 寄存器中进行读取，以将 RXNE 标志清零。

DMA 管理的 CRC 传输

当使能的 SPI 通信支持 CRC 通信和 DMA 模式时，在通信结束时会自动发送和接收 CRC（在只接收模式下读取 CRC 数据时除外）。CRCNEXT 位不是一定要通过软件来处理。SPI 发送 DMA 通道计数器必须设置为要发送的数据帧数，其中不包括 CRC 帧。在接收器侧，接收的 CRC 值在传输结束时通过 DMA 自动处理，但用户必须注意刷新 SPI_DR 中接收的 CRC 帧（因为该信息始终加载到其中）。

如果传输过程中出现损坏，则在数据和 CRC 传输结束时，SPIx_SR 寄存器中的 CRCERR 标志将置 1。

复位 SPIx_TXCRC 和 SPIx_RXCRC 值

当使能 CRC 计算时，SPIx_TXCRC 和 SPIx_RXCRC 值自动清零。

当 SPI 配置为从模式并且 CRC 功能已使能时，即使 NSS 引脚上为高电平，也会进行 CRC 计算。例如，在多从模式环境下可能出现这种情况，此时通信主器件会交替寻址从器件。

在禁止从器件（NSS 上为高电平）到使能新的从器件（NSS 上为低电平）的时间内，应在主器件和从器件两端同时将 CRC 值清零，以重新同步主从双方的 CRC 计算。

要将 CRC 清零，请按以下步骤操作：

1. 禁止 SPI
2. 将 CRCEN 位清零
3. 使能 CRCEN 位
4. 使能 SPI

注：当 SPI 处于从模式时，只要 CRCEN 位置 1，无论 SPE 位的值如何，只要存在 SCK 从输入时钟，CRC 计算器就开始工作。为了避免 CRC 计算错误，软件必须仅在时钟稳定（处于稳态）时才能使能 CRC 计算。当 SPI 接口配置为从模式时，NSS 内部信号需要在数据阶段和 CRC 阶段之间保持低电平。

31.5 SPI 中断

在 SPI 通信过程中，中断可由以下事件产生：

- 发送缓冲区准备就绪，可以装载数据
- 接收缓冲区中接收了数据
- 主模式故障
- 上溢错误
- TI 帧格式错误

中断可分别进行使能和禁止。

表 139. SPI 中断请求

中断事件	事件标志	使能控制位
发送缓冲区准备就绪，可以装载数据	TXE	TXEIE
接收缓冲区中接收了数据	RXNE	RXNEIE
主模式故障	MODF	ERRIE
上溢错误	OVR	
CRC 错误	CRCERR	
TI 帧格式错误	FRE	

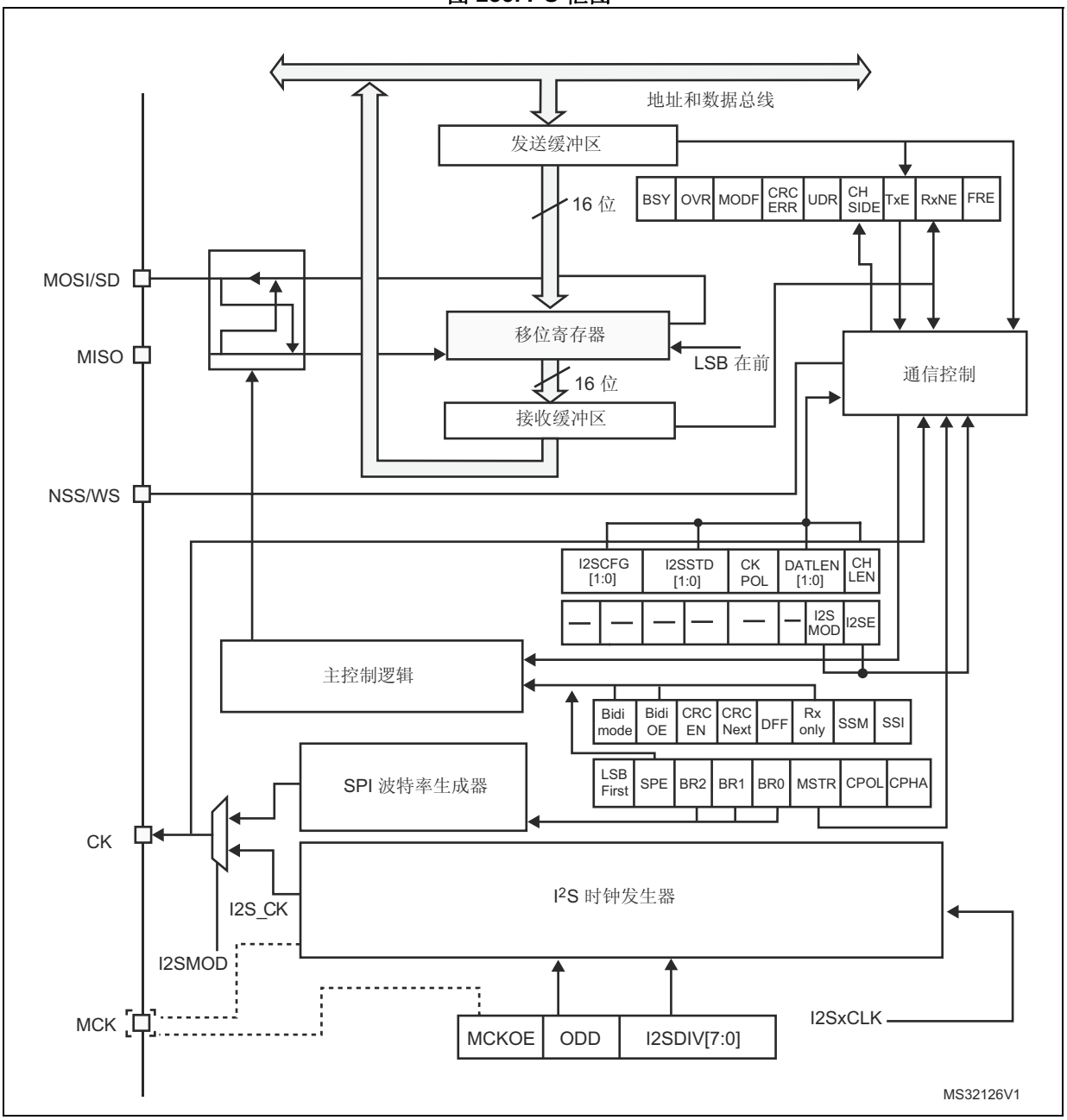
有关代码示例，请参见 *A.19.6：SPI 中断代码示例*。

31.6 I²S 功能说明

31.6.1 I²S 一般说明

I²S 的框图如*图 285* 所示。

图 285. I²S 框图



1. MCK 映射到 MISO 引脚上。

当使能 I²S 功能（将 SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1）后，SPI 可用作音频 I²S 接口。此接口使用几乎与 SPI 相同的引脚、标志和中断。

I^2S 与 SPI 共用以下三个引脚：

- SD：串行数据（映射到 MOSI 引脚），用于发送或接收两个时分复用的数据通道上的数据（仅半双工模式）。
- WS：字选择（映射到 NSS 引脚），是主模式下的数据控制信号输出以及从模式下的数据控制信号输入。
- CK：串行时钟（映射到 SCK 引脚），是主模式下的串行时钟输出以及从模式下的串行时钟输入。

当某些外部音频设备需要使用主时钟输出时，可以使用其它引脚：

- MCK：当 I^2S 配置为主模式（并且 SPIx_I2SPR 寄存器中的 MCKOE 位置 1）时，使用主时钟（单独映射）输出此附加时钟，该时钟以 256 × f_S 的预配置频率生成，其中 f_S 为音频信号采样频率。

I^2S 在主模式下使用自身的时钟发生器生成通信时钟。此时钟发生器也是主时钟输出的源。在 I^2S 模式下可以使用两个额外的寄存器。一个是时钟发生器配置寄存器 SPIx_I2SPR，另一个是通用 I^2S 配置寄存器 SPIx_I2SCFGR（音频标准、从/主模式、数据格式、数据包帧、时钟极性等）。

在 I^2S 模式下不使用 SPIx_CR1 寄存器和所有 CRC 寄存器。同样，也不使用 SPIx_CR2 寄存器中的 SSOE 位以及 SPIx_SR 中的 MODF 和 CRCERR 位。

I^2S 使用相同的 SPI 寄存器 (SPIx_DR) 进行 16 位数据传输。

31.6.2 支持的音频协议

三线总线仅需要处理通常在左右两个通道上时分复用的音频数据。但是，只有一个 16 位寄存器进行发送和接收。所以，需由软件将与每个通道对应的值写入数据寄存器，以及从数据寄存器中读取数据，并通过检查 SPIx_SR 寄存器中的 CHSIDE 位来识别对应的通道。始终先发送左通道数据，而后再发送右通道数据（对于 PCM 协议来说，CHSIDE 没有意义）。

数据和帧格式组合有四种，可采用下列格式发送数据：

- 将 16 位数据封装在 16 位帧中
- 将 16 位数据封装在 32 位帧中
- 将 24 位数据封装在 32 位帧中
- 将 32 位数据封装在 32 位帧中

当使用 32 位数据包中的 16 位数据时，前 16 位 (MSB) 为有效位，16 位 LSB 被强制清零，无需任何软件操作或 DMA 请求（只需一个读/写操作）。

如果应用程序首选 DMA，则 24 位和 32 位数据帧需要对 SPIx_DR 寄存器执行两次 CPU 读取或写入操作，或者需要两次 DMA 操作。对于 24 位数据帧，硬件会在低 8 位填充 8 个 0 扩展到 32 位。

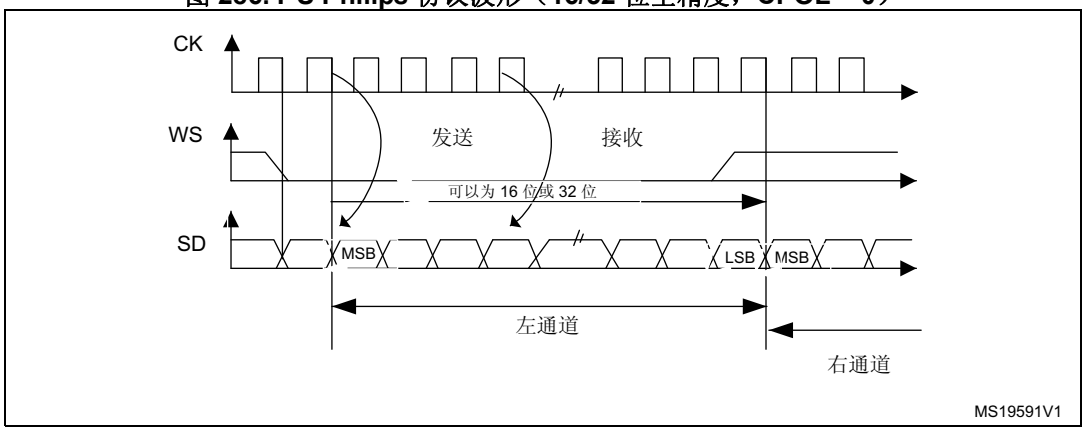
对于所有数据格式和通信标准而言，始终会先发送最高有效位（MSB 优先）。

I^2S 接口支持四种音频标准，可使用 SPIx_I2SCFGR 寄存器中的 I2SSTD[1:0] 和 PCMSYNC 位对其进行配置。

I²S Philips 标准

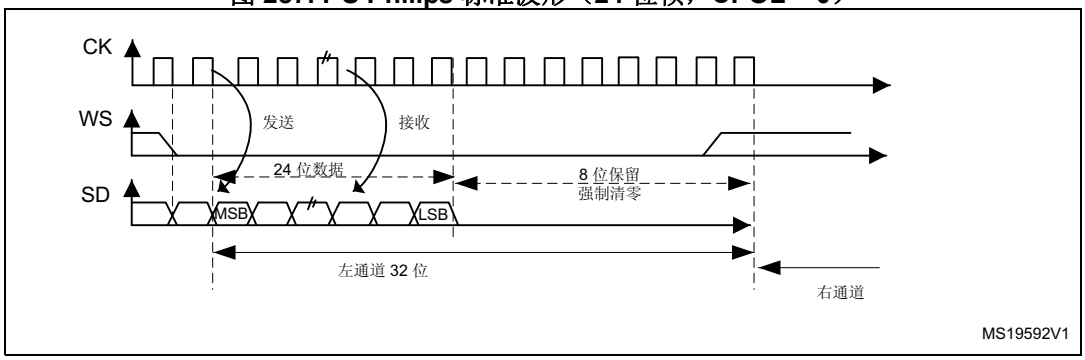
使用 WS 信号来指示当前正在发送的数据所属的通道。该信号从当前通道数据的第一个位 (MSB) 之前的一个时钟开始有效。

图 286. I²S Philips 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 CK 的下降沿变化。

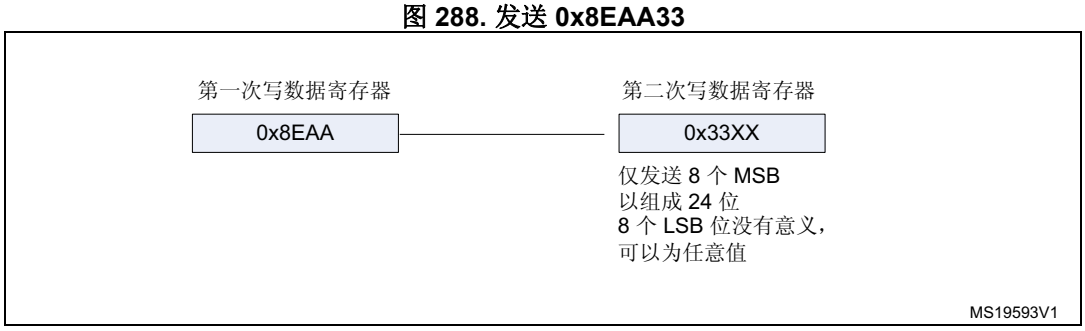
图 287. I²S Philips 标准波形（24 位帧，CPOL = 0）



该模式需要对 SPIx_DR 寄存器执行两次写入或读取操作。

- 在发送模式下：
  如果需要发送 0x8EAA33（24 位）：

图 288. 发送 0x8EAA33



- 在接收模式下：
  如果接收数据 0x8EAA33：

图 289. 接收 0x8EAA33

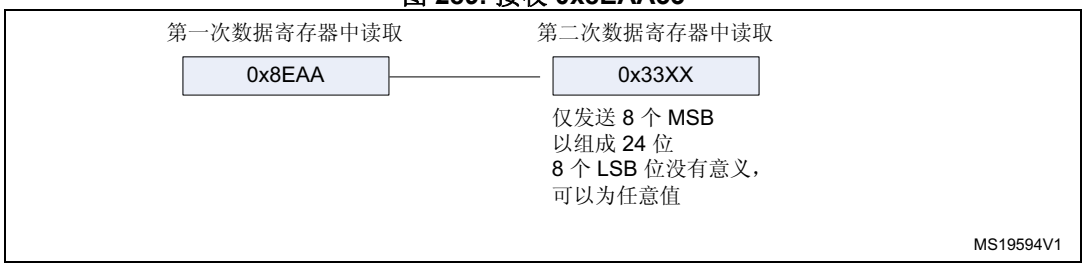
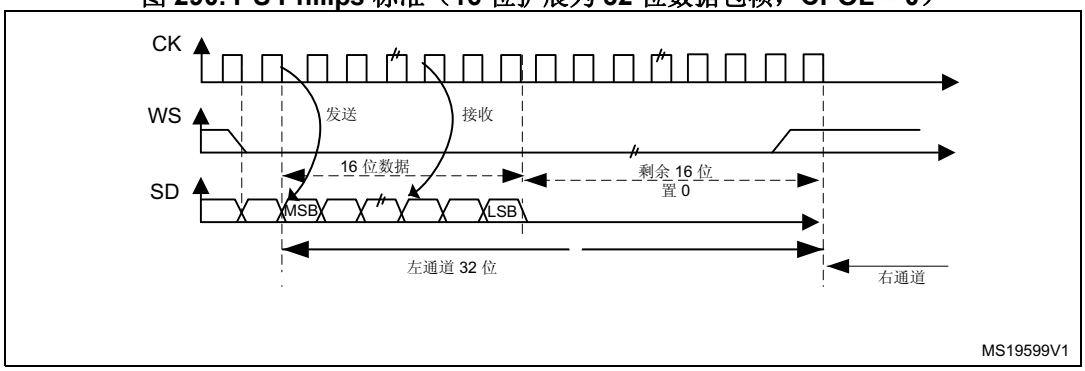


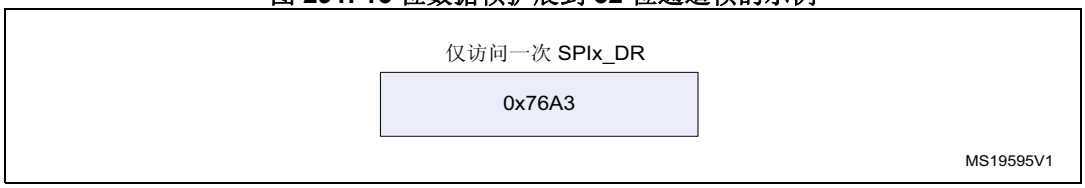
图 290. I²S Philips 标准（16 位扩展为 32 位数据包帧，CPOL = 0）



如果在 I²S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧，则只需要访问一次 SPIx_DR 寄存器。扩展到 32 位中高半字（16 位 MSB）被硬件置为 0x0000。

如果要发送的数据或已接收的数据为 0x76A3（0x76A30000 扩展为 32 位），则需要执行*图 291* 中显示的操作。

图 291. 16 位数据帧扩展到 32 位通道帧的示例



发送时，每次将 MSB 写入 SPIx_DR，TXE 标志就会置 1，并在中断使能的情况下触发中断，以将要发送的新数据加载到 SPIx_DR 寄存器。即使硬件填充的低 16 位 0x0000 还未发送，也会如此，因为低 16 位是由硬件发送。

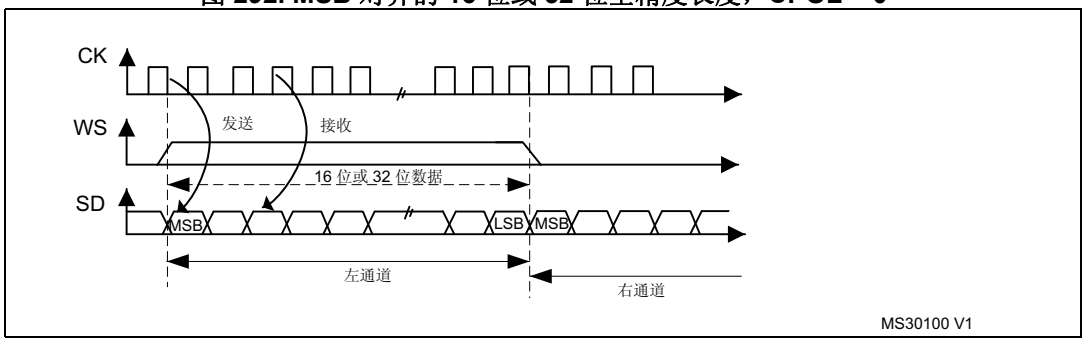
接收时，接收到第一个半字（高 16 位），则硬件将 RXNE 标志置 1，并在中断使能的情况下触发中断。

这样，就延长了两个写入或读取操作之间的时间间隔，从而可防止出现下溢或上溢情况（具体取决于数据传输方向）。

MSB 对齐标准

此标准同时生成 WS 信号和第一个数据位（即 MSBit）。

图 292. MSB 对齐的 16 位或 32 位全精度长度，CPOL = 0



发送方在时钟信号的下降沿改变数据；接收方在上升沿读取数据。

图 293. MSB 对齐的 24 位帧长度，CPOL = 0

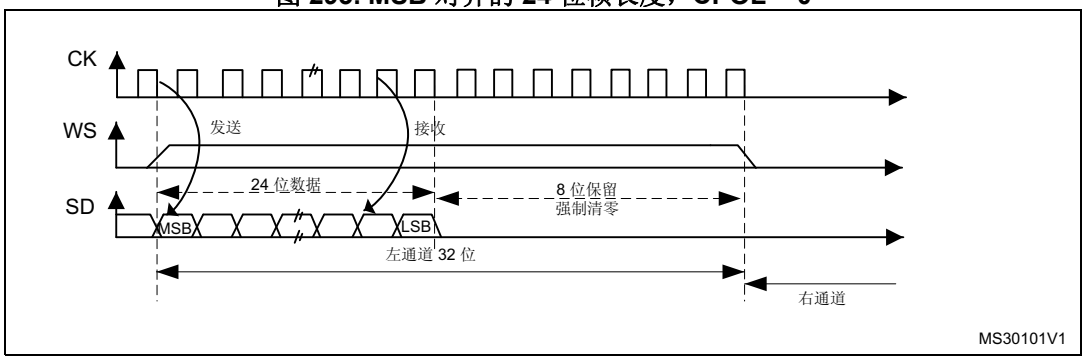
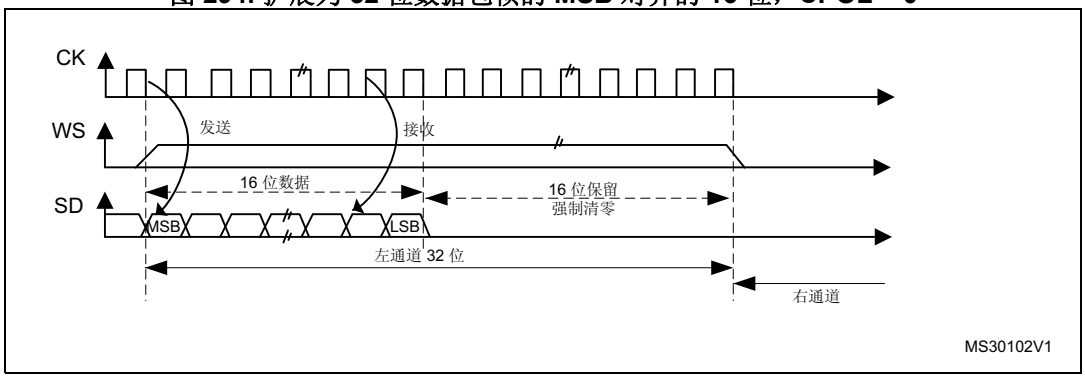


图 294. 扩展为 32 位数据包帧的 MSB 对齐的 16 位，CPOL = 0



LSB 对齐标准

该标准与 MSB 对齐标准类似（对于 16 位和 32 位全精度帧格式，没有任何不同）。

图 295. LSB 对齐的 16 位或 32 位全精度，CPOL = 0

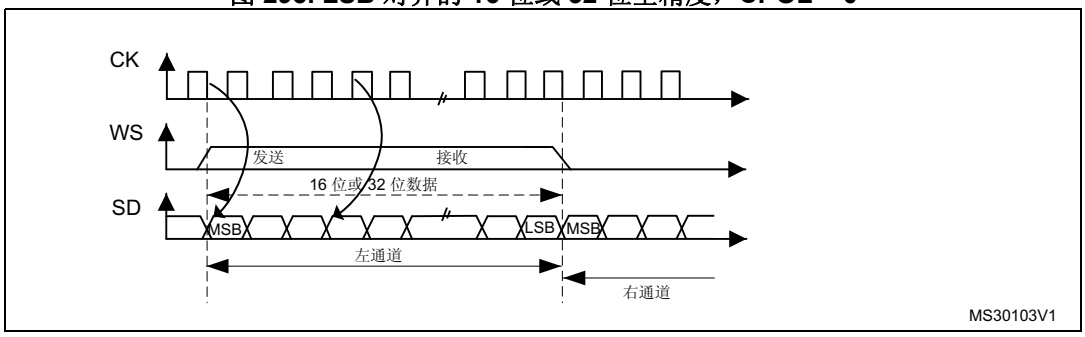
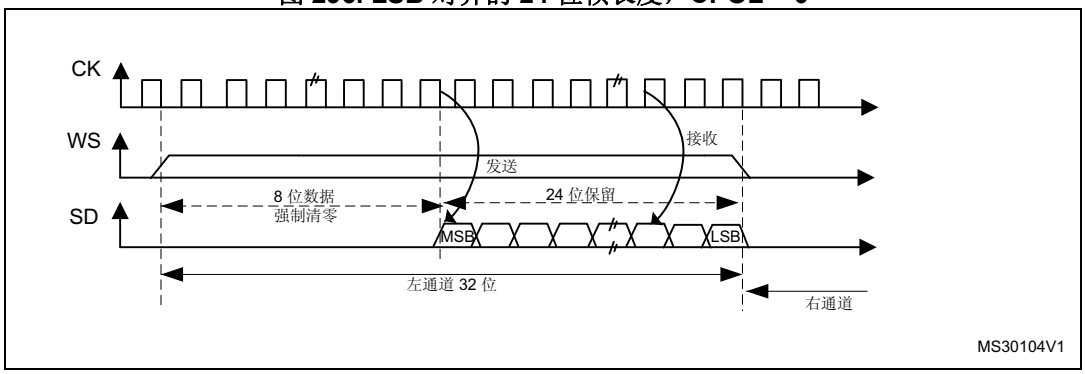


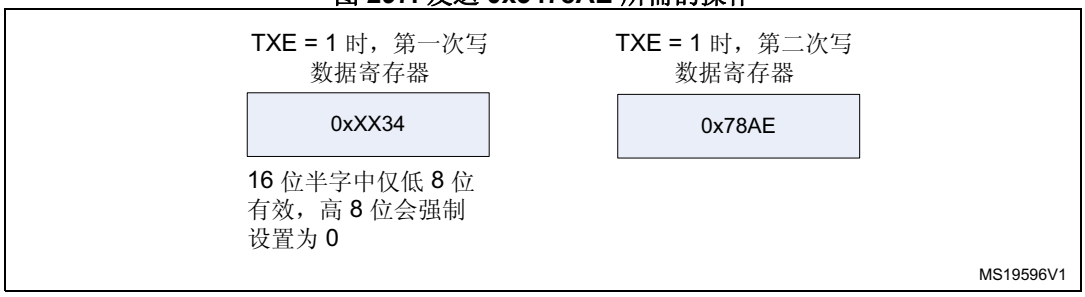
图 296. LSB 对齐的 24 位帧长度，CPOL = 0



- 在发送模式下：

  如果需要发送数据 0x3478AE，则需要通过软件或 DMA 对 SPIx_DR 寄存器执行两次写入操作。下面给出了这些操作。

图 297. 发送 0x3478AE 所需的操作



- 在接收模式下：
  如果接收到数据 0x3478AE，则在每个 RXNE 事件时需要对 SPIx_DR 寄存器执行两次连续的读取操作。

图 298. 接收 0x3478AE 时所需的操作

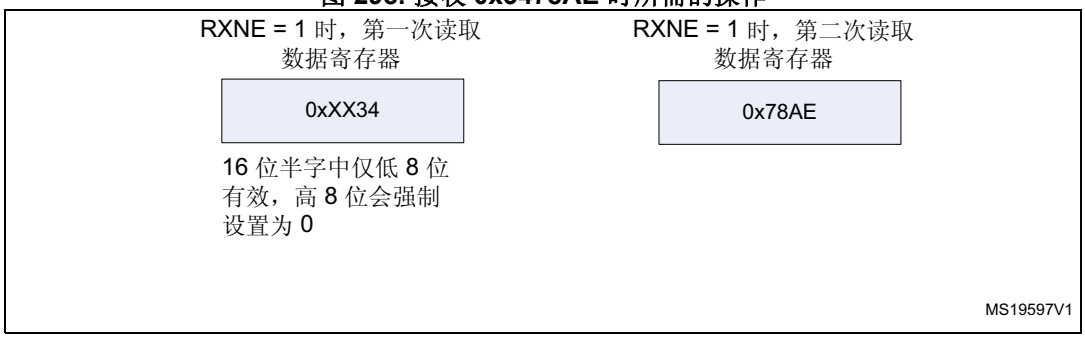
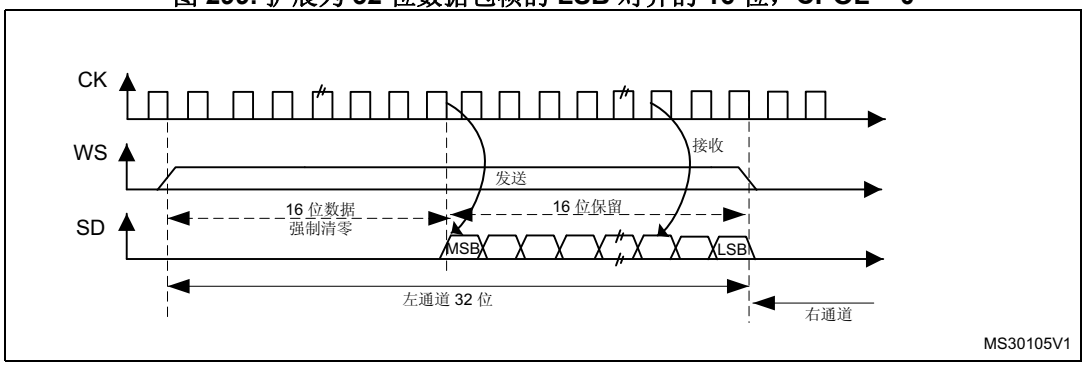


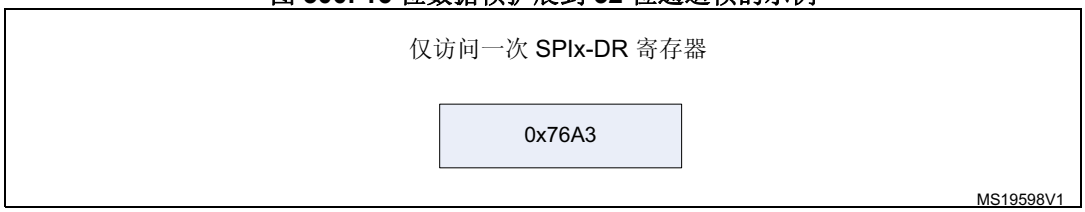
图 299. 扩展为 32 位数据包帧的 LSB 对齐的 16 位，CPOL = 0



如果在 I²S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧，则只需要访问一次 SPIx_DR 寄存器。扩展到 32 位中高半字（16 位 MSB）被硬件置为 0x0000。在这种情况下，其对应于半字 MSB。

如果要发送的数据或已接收的数据为 0x76A3（0x0000 76A3 扩展为 32 位），则需要执行*图 300* 中显示的操作。

图 300. 16 位数据帧扩展到 32 位通道帧的示例



在发送模式下，发生 TXE 事件时，应用程序需要写入要发送的数据（此例中，为 0x76A3）。首先发送 0x000 字段（扩展到 32 位）。有效数据 (0x76A3) 发送到 SD 后，TXE 标志会被再次置 1。

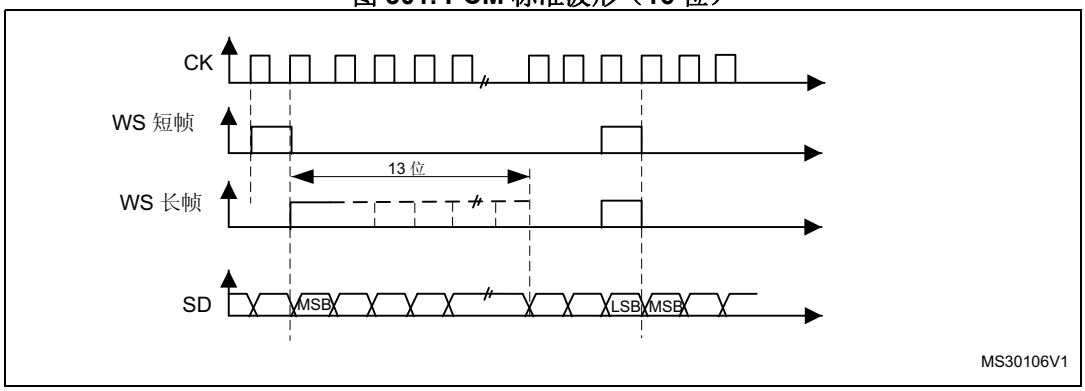
在接收模式下，当接收到有效半字后（而非 0x0000 字段），即会置位 RXNE。

这样，就延长了两个写入或读取操作之间的时间间隔，以防止出现下溢或上溢情况。

PCM 标准

对于 PCM 标准，无需使用通道信息。可使用两种 PCM 模式（短帧和长帧），并且可使用 SPIx_I2SCFGR 寄存器中的 PCMSYNC 位来配置。

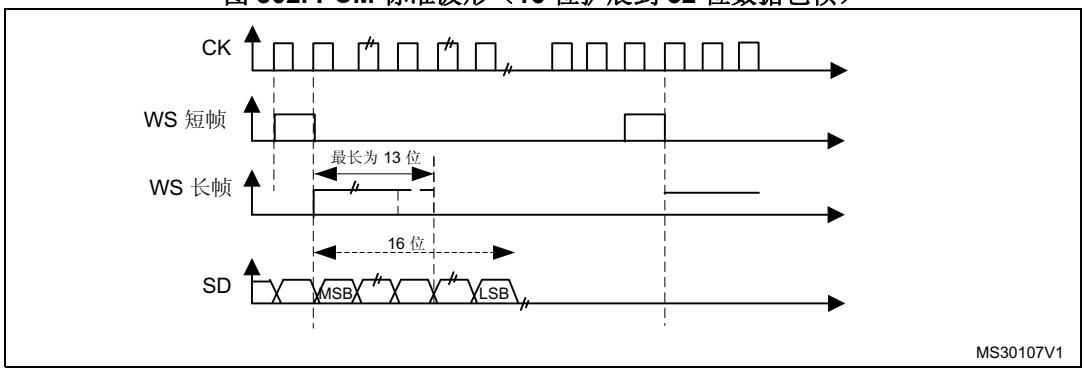
图 301. PCM 标准波形（16 位）



对于长帧同步，在主模式下会将 WS 信号持续 13 个周期。

对于短帧同步，WS 同步信号的持续时间仅为一个周期。

图 302. PCM 标准波形（16 位扩展到 32 位数据包帧）



注：对于两种模式（主/从模式）和两种同步（短/长同步），即使在从模式下，也需要指定两组连续数据（以及两个同步信号）之间位的个数（SPIx_I2SCFGR 寄存器中的 DATLEN 位和 CHLEN 位）。

31.6.3 时钟发生器

I^2S 比特率用来确定 I^2S 数据线上的数据流和 I^2S 时钟信号频率。

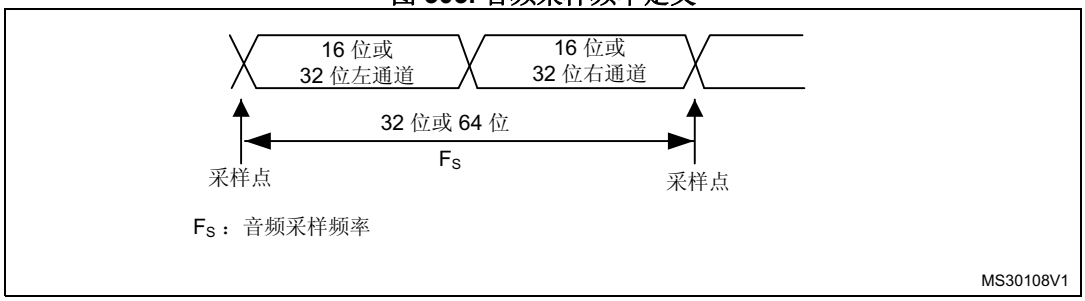
I^2S 比特率 = 每个通道的位数 × 通道数 × 音频采样频率

对于 16 位双通道音频，I^2S 比特率的计算公式如下：

I^2S 比特率 = $16 \times 2 \times f_S$

如果数据包为 32 位宽，则 I^2S 比特率 = $32 \times 2 \times f_S$。

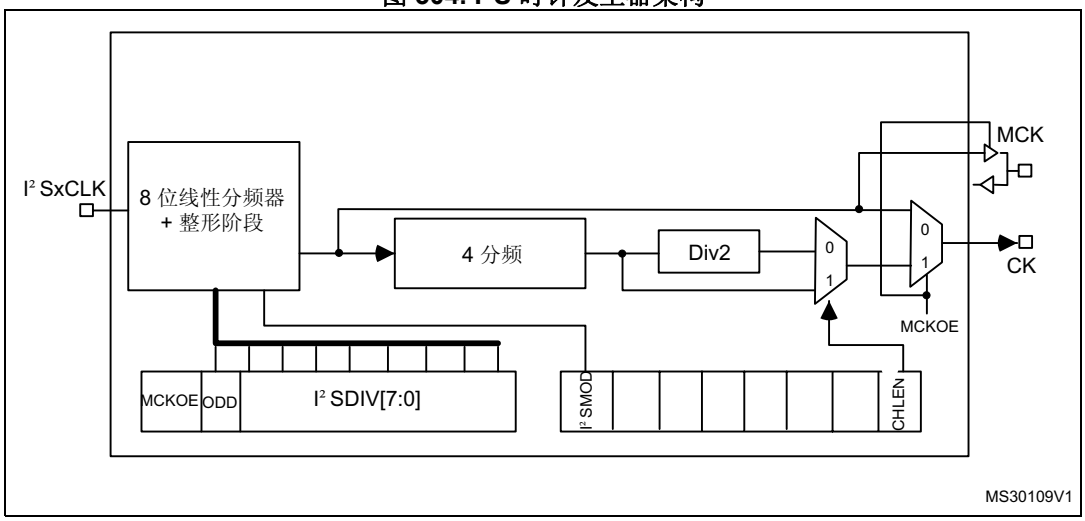
图 303. 音频采样频率定义



配置主模式时，需要正确地对线性分频器进行设置，以便采用所需的音频频率进行通信。

图 304 展示了通信时钟架构。I2Sx 时钟始终为系统时钟。

图 304. I²S 时钟发生器架构



1. 其中 x = 2。

音频采样频率可以是 192 KHz、96 kHz、48 kHz、44.1 kHz、32 kHz、22.05 kHz、16 kHz、11.025 kHz 或 8 kHz（或此范围内的任何其它值）。为达到所需频率，需要根据以下公式对线性分频器进行编程：

输出主时钟（SPIx_I2SPR 寄存器中的 MCKOE 置 1)时：

f_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)]（通道帧宽度为 16 位时）

f_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)]（通道帧宽度为 32 位时）

禁止主时钟输出（MCKOE 位清零）时：

f_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)]（通道帧宽度为 16 位时）

f_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)]（通道帧宽度为 32 位时）

表 140 提供了针对不同时钟配置的示例精度值。

注： *还可以采用其它配置以达到更好的时钟精度。*

表 140. 使用标准 8 MHz HSE 时的音频频率精度

SYSCLK (MHz)	数据长度	I2SDIV	I2SODD	MCLK	目标 fs (Hz)	实际 f_s (kHz)	误差
32	16	5	0	无	96000	100	4.1667%
32	32	2	0	无	96000	100	4.1667%
32	16	10	1	无	48000	47.619	0.7937%
32	32	5	0	无	48000	50	4.1667%
32	16	11	1	无	44100	43.478	1.4098%
32	32	5	1	无	44100	45.454	3.0715%
32	16	15	1	无	32000	32.258	0.8065%
32	32	8	0	无	32000	31.25	2.3430%
32	16	22	1	无	22050	22.222	0.7811%
32	32	11	1	无	22050	21.739	1.4098%
32	16	31	1	无	16000	15.873	0.7937%
32	32	15	1	无	16000	16.129	0.8065%
32	16	45	1	无	11025	10.989	0.3264%
32	32	22	1	无	11025	11.111	0.7811%
32	16	62	1	无	8000	8	0.0000%
32	32	31	1	无	8000	7.936	0.7937%
32	16	2	0	有	32000	31.25	2.3430%
32	32	2	0	有	32000	31.25	2.3430%
32	16	3	0	有	22050	20.833	5.5170%
32	32	3	0	有	22050	20.833	5.5170%
32	16	4	0	有	16000	15.625	2.3428%
32	32	4	0	有	16000	15.625	2.3428%
32	16	5	1	有	11025	11.363	3.0715%
32	32	5	1	有	11025	11.363	3.0715%
32	16	8	0	有	8000	7.812	2.3428%
32	32	8	0	有	8000	7.812	2.3428%

31.6.4 I²S 主模式

I²S 可配置为主模式。这意味着将在 CK 引脚输出串行时钟，在 WS 引脚生成字选信号。主时钟 (MCK) 可以输出，也可以不输出，具体由 SPIx_I2SPR 寄存器中的 MCKOE 位控制。

步骤

1. 设置 SPIx_I2SPR 寄存器的 I2SDIV[7:0] 位，以定义串行时钟波特率，从而达到相应的音频采样频率。SPIx_I2SPR 寄存器的 ODD 位也需要设置。
2. 设置 CKPOL 位，定义时钟在空闲时的电平状态。如果需要为外部 DAC/ADC 音频组件提供主时钟 MCK，则将 SPIx_I2SPR 寄存器的 MCKOE 位置 1（I2SDIV 和 ODD 值应根据 MCK 输出的状态进行计算。有关详细信息，请参见*第 31.6.3 节：时钟发生器*）。
3. 将 SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1 以激活 I²S 功能，通过 I2SSTD[1:0] 和 PCMSYNC 位选择 I²S 标准，通过 DATLEN[1:0] 位选择数据长度并通过配置 CHLEN 位选择每个通道的位数。此外，通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择 I²S 主模式和方向（发送器或接收器）。
4. 如果需要，通过对 SPIx_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。
5. SPIx_I2SCFGR 寄存器的 I2SE 位必须置 1。

WS 和 CK 配置为输出模式。如果 SPIx_I2SPR 的 MCKOE 位置 1，则 MCK 也是输出。

发送序列

将半字写入发送缓冲区后，发送序列随即开始。

假设写入发送缓冲区的第一个数据对应于左通道数据。数据从发送缓冲区传输到移位寄存器时，TXE 置 1，并且必须将对应于右通道的数据写入发送缓冲区。CHSIDE 标志指示将发送的数据对应的通道。TXE 标志置 1 时，CHSIDE 标志有意义，因为该标志在 TXE 变为高电平时进行更新。

一个完整帧表示先进行左通道数据发送再进行右通道数据发送。不存在仅发送左通道的部分帧。

首位发送期间，数据按半字并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后，TXE 标志都将置 1，如果 SPIx_CR2 寄存器的 TXEIE 位置 1，将产生中断。

有关各种 I²S 标准模式的写操作的更多详细信息，请参见*第 31.6.2 节：支持的音频协议*。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送的数据写入 SPIx_DR 寄存器。

要通过将 I2SE 清零来关闭 I²S，必须等待 TXE = 1 且 BSY = 0。

接收序列

此工作模式与发送模式相同，只有第 3 点存在不同（请参见*第 31.6.4 节：I²S 主模式*所述的步骤），即通过 I2SCFG[1:0] 位设置主器件接收模式。

无论数据或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区满时，RXNE 标志即置 1，并且如果 SPIx_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可通过一次或两次接收操作进入接收缓冲区，具体取决于数据和通道长度配置。

读取 SPIx_DR 寄存器即会使 RXNE 位清零。

CHSIDE 在每次接收后进行更新。它由 I²S 单元所产生的 WS 信号触发。

有关各种 I²S 标准模式中读操作的更多详细信息，请参见*第 31.6.2 节：支持的音频协议*。

如果在先前收到的数据尚未读取时又接收到新数据，将产生上溢错误并将 OVR 标志置 1。如果 SPIx_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要关闭 I²S，需要执行特定操作来确保 I²S 正确完成传输周期而不启动新的数据传输。该序列取决于数据和通道长度的配置，以及所选的音频协议模式。在以下情况下：

- 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 LSB 对齐模式 (I2SSTD = 10)
  - a) 等待倒数第二个 RXNE = 1 (n – 1)
  - b) 然后等待 17 个 I²S 时钟周期（使用软件循环）
  - c) 禁止 I²S (I2SE = 0)
- 32 位通道长度上扩展的 16 位数据长度（DATLEN = 00 且 CHLEN = 1），使用 MSB 对齐、I²S 或 PCM 模式（分别为 I2SSTD = 00、I2SSTD = 01 或 I2SSTD = 11）
  - a) 等待最后一个 RXNE
  - b) 然后等待 1 个 I²S 时钟周期（使用软件循环）
  - c) 禁止 I²S (I2SE = 0)
- 对于 DATLEN 和 CHLEN 的所有其它组合，无论通过 I2SSTD 位选择何种音频模式，都将执行以下序列来关闭 I²S：
  - a) 等待倒数第二个 RXNE = 1 (n – 1)
  - b) 然后等待一个 I²S 时钟周期（使用软件循环）
  - c) 禁止 I²S (I2SE = 0)

注： *传输期间，BSY 标志保持低电平。*

31.6.5 I²S 从模式

对于从配置而言，I²S 可配置为发送模式或接收模式。

此工作模式所遵循的规则与 I²S 主模式配置基本相同。在从模式下，I²S 接口不产生时钟。时钟和 WS 信号从 I²S 接口所连接的外部主器件输入。这样，用户便不需要配置时钟。

应遵循如下配置步骤：

1. 将 SPIx_I2SCFGR 寄存器的 I2SMOD 位置 1 以选择 I²S 模式，通过 I2SSTD[1:0] 位选择 I²S 标准，通过 DATLEN[1:0] 位选择数据长度并通过配置 CHLEN 位选择帧中每个通道的位数。此外，通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位选择从器件的模式（发送或接收）。
2. 如果需要，通过对 SPIx_CR2 寄存器执行写操作来选择所有可能的中断源和 DMA 功能。
3. SPIx_I2SCFGR 寄存器的 I2SE 位必须置 1。

发送序列

当外部主器件发送时钟并且通过 NSS_WS 信号请求传输数据时，发送序列开始。必须首先使能从器件，然后外部主器件才能开始通信。主器件开始通信前，还必须加载 I²S 数据寄存器。

对于 I²S、MSB 对齐和 LSB 对齐模式，要写入数据寄存器的第一个数据项对应于左通道的数据。通信开始时，数据从发送缓冲区传输到移位寄存器。TXE 标志随即置 1，以请求将右通道的数据写入 I²S 数据寄存器。

CHSIDE 标志指示将发送的数据对应的通道。与主发送模式相比，在从模式下，CHSIDE 由来自外部主器件的 WS 信号触发。这意味着，从器件需要首先为发送第一个数据做好准备，然后主器件才能产生时钟。WS 置位意味着首先发送左通道数据。

注： *必须要在主器件发出的第一个时钟出现在 CK 线上至少 2 个 PCLK 周期之前置位 I2SE。*

首位发送期间，数据按半字从内部总线并行加载到 16 位移位寄存器中，然后以串行方式移位并输出到 MOSI/SD 引脚（MSB 在前）。每次数据从发送缓冲区传输到移位寄存器后，TXE 标志都将置 1，如果 SPIx_CR2 寄存器的 TXEIE 位置 1，将产生中断。

请注意，仅当 TXE 标志为 1 时，才可以尝试向发送缓冲区写入数据。

有关各种 I^2S 标准模式中写操作的更多详细信息，请参见*第 31.6.2 节：支持的音频协议*。

为确保连续进行音频数据发送，必须在当前数据发送结束前将下一个要发送数据写入 SPIx_DR 寄存器。如果在数据尚未写入 SPIx_DR 寄存器时下一个数据通信的首个时钟边沿到来，下溢标志将置 1 并可能产生中断。通过这种方式，软件可以获知所传输的数据不正确。如果 SPIx_CR2 寄存器的 ERRIE 位置 1，则当 SPIx_SR 寄存器中的 UDR 标志变为 1 时，将产生中断。这种情况下，必须关闭 I^2S 并从左通道开始重新启动数据传输。

要通过将 I2SE 位清零来关闭 I^2S，必须等待 TXE = 1 且 BSY = 0。

接收序列

此工作模式与发送模式相同，只有第 1 点存在不同（请参见*第 31.6.5 节：I^2S 从模式*所述的步骤），即通过 SPIx_I2SCFGR 寄存器的 I2SCFG[1:0] 位设置从器件接收模式。

无论数据长度或通道长度如何，音频数据始终按 16 位数据包进行接收。这意味着，每当接收缓冲区填满时，SPIx_SR 寄存器中的 RXNE 标志即置 1，并且如果 SPIx_CR2 寄存器的 RXNEIE 位置 1，还将产生中断。所接收的右通道或左通道的音频值可能通过一次或两次接收操作进入接收缓冲区，具体取决于数据长度和通道长度配置。

每次接收要从 SPIx_DR 寄存器读取的数据时，CHSIDE 标志都将更新。该标志由外部主器件所管理的外部 WS 线路触发。

读取 SPIx_DR 寄存器即会使 RXNE 位清零。

有关各种 I^2S 标准模式的读操作的更多详细信息，请参见*第 31.6.2 节：支持的音频协议*。

如果在先前收到的数据尚未读取时又接收到新数据，将产生上溢错误，OVR 标志将置 1。如果 SPIx_CR2 寄存器的 ERRIE 位置 1，将产生中断以指示该错误。

要在接收模式下关闭 I^2S，必须在接收到最后一个 RXNE = 1 后立即将 I2SE 清零。

注： *外部主器件应能够通过音频通道以 16 位或 32 位数据包发送/接收数据。*

31.6.6 I^2S 状态标志

应用程序可通过三个状态标志来全面监视 I^2S 总线的状态。

忙标志 (BSY)

BSY 标志由硬件置 1 和清零（写入此标志没有任何作用）。该标志表示 I^2S 通信层的状态。

BSY 置 1 时，表示 I^2S 正在忙于通信。在主接收模式 (I2SCFG = 11) 中，BSY 标志的情况例外，该标志在接收期间仍保持低电平。

如果软件需要禁止 I^2S，可使用 BSY 标志检测传输是否结束。这可以避免损坏最后传输的数据。为此，必须严格遵循下述步骤。

在传输开始时（I^2S 处于主接收器模式时除外），将 BSY 标志置 1。

出现以下情况时，BSY 标志被硬件清零：

- 传输完成时（主发送模式除外，在该模式下通信是连续的）
- 禁止 I^2S 时

当通信连续时：

- 在主发送模式下，BSY 标志在所有传输期间均保持高电平
- 在从模式下，BSY 标志在每次传输之间变为低电平并持续一个 I^2S 时钟周期

注：请勿使用 BSY 标志处理每次数据发送或接收，最好改用 TXE 标志和 RXNE 标志。

发送缓冲区为空 (TXE)

如果此标志置 1，表示发送缓冲区为空，可将要发送的下一个数据加载到其中。发送缓冲区已包含要发送的数据时，TXE 标志复位。禁止 I^2S（I2SE 位复位）时，该标志也会复位。

接收缓冲区非空 (RXNE)

此标志置 1 时，表示接收缓冲区中存在有效的已接收数据。读取 SPIx_DR 寄存器时，该标志复位。

通道方向 (CHSIDE)

在发送模式下，此标志将在 TXE 变为高电平时进行刷新。此标志指示 SD 上要传输的数据所属的通道。如果在从发送模式下发生下溢错误事件，此标志将不可靠，在恢复通信前需要关闭并重新开启 I^2S。

在接收模式下，该标志将在 SPIx_DR 中接收数据时进行刷新。它表示接收的数据所属的通道。请注意，如果发生错误（例如 OVR），此标志将失去意义，应通过禁止并重新使能 I^2S（根据需要更改配置）来复位。

此标志在 PCM 标准中没有意义（短帧和长帧模式）。

当 SPIx_SR 中的 OVR 或 UDR 标志置 1，并且 SPIx_CR2 中的 ERRIE 位也置 1 时，将产生中断。中断源被清除后，可通过读取 SPIx_SR 状态寄存器来将此中断清除。

31.6.7 I^2S 错误标志

I^2S 单元共有三个错误标志。

下溢标志 (UDR)

在从发送模式下，如果在软件尚未将任何值加载到 SPIx_DR 之前出现第一个数据发送时钟，此标志将置 1。SPIx_I2SCFGR 寄存器中的 I2SMOD 位置 1 时，可以使用此标志。如果 SPIx_CR2 寄存器中的 ERRIE 位置 1，可产生中断。
UDR 位通过 SPIx_SR 寄存器上的读操作进行清零。

上溢标志 (OVR)

如果在尚未从 SPIx_DR 寄存器读取上一个数据时又接收到新数据，此标志将置 1。因此，传入的数据将丢失。如果 SPIx_CR2 寄存器中的 ERRIE 位置 1，可产生中断。

这种情况下，将不会用接收到的新数据更新接收缓冲区的内容。对 SPIx_DR 寄存器执行的读操作将返回先前正确接收的数据。主器件后续发送的所有其它半字都将丢失。

要将 OVR 位清零，应首先对 SPIx_DR 寄存器执行读操作，然后再对 SPIx_SR 寄存器进行读访问。

帧错误标志 (FRE)

仅当 I^2S 配置为从模式时，此标志才可由硬件置 1。如果外部主器件没有按照从器件期望的那样改变 WS 线，则此标志将置 1。如果失去同步，要从此状态中恢复并将外部主器件与 I^2S 从器件重新同步，需执行下列步骤：

1. 禁止 I^2S。
2. 在 WS 线上检测到正确的电平时将其重新使能（WS 线在 I^2S 模式下为高电平，在 MSB 对齐、LSB 对齐或 PCM 模式下为低电平）。

主器件与从器件之间的同步失效可能是由于 SCK 通信时钟或 WS 帧同步信号线上存在噪音干扰。如果 ERRIE 位置 1，可产生错误中断。读取状态寄存器时，同步失效标志 (FRE) 由软件清零。

31.6.8 I^2S 中断

表 141 为 I^2S 中断的列表。

表 141. I^2S 中断请求

中断事件	事件标志	使能控制位
发送缓冲区为空	TXE	TXEIE
接收缓冲区非空	RXNE	RXNEIE
上溢错误	OVR	ERRIE
下溢错误	UDR	
帧错误	FRE	

31.6.9 DMA 特性

在 I^2S 模式下，DMA 的工作方式与在 SPI 模式下完全相同。除了由于不存在数据传输保护机制，I^2S 模式下没有 CRC 功能外，没有其它差别。

31.7 SPI 和 I²S 寄存器

外设寄存器可支持半字（16 位）或字（32 位）访问。此外，SPI_DR 可支持 8 位访问。

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

外设寄存器可支持半字（16 位）或字（32 位）访问。

31.7.1 SPI 控制寄存器 1 (SPI_CR1)（不用于 I²S 模式）

SPI control register 1

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
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15 ***BIDIMODE***：双向通信数据模式使能 (Bidirectional data mode enable)

该位通过一条共用的双向数据线来支持半双工通信。双向模式激活时，使 RXONLY 位保持清零状态。

0：选择双线单向通信数据模式

1：选择单线双向数据模式

注： ***该位不适用于 I²S 模式***

位 14 **BIDIOE**：双向模式下的输出使能 (Output enable in bidirectional mode)

该位结合 BIDIMODE 位，用于选择双向模式下的传输方向

0：禁止输出（只接收模式）

1：使能输出（只发送模式）

注： ***该位不适用于 I²S 模式。***

主模式下使用 MOSI 引脚，而从模式下使用 MISO 引脚。

位 13 **CRCEN**：硬件 CRC 计算使能 (Hardware CRC calculation enable)

0：禁止 CRC 计算

1：使能 CRC 计算

注： *为确保正确操作，只应在禁止 SPI（SPE = “0”）时对此位执行写操作。*

不适用于 I²S 模式。

位 12 **CRCNEXT**：下一次 CRC 传输 (CRC transfer next)

0：数据阶段（无 CRC 阶段）

1：下一次传输为 CRC（CRC 阶段）

注： *当 SPI 配置为全双工或只发送器模式时，只要最后一个数据写入 SPI_DR 寄存器，就必须对 CRCNEXT 执行写操作。*

当 SPI 配置为只接收器模式时，必须在接收到倒数第二个数据之后将 CRCNEXT 置 1。

当传输由 DMA 管理时，此位应保持清零状态。

不适用于 I²S 模式。

位 11 **DFF**：数据帧格式 (Data frame format)

0：为发送/接收选择 8 位数据帧格式

1：为发送/接收选择 16 位数据帧格式

注：为确保正确操作，只应在禁止 SPI（SPE = "0"）时对此位执行写操作。

不适用于 I^2S 模式。

位 10 **RXONLY**：只接收模式使能 (Receive only mode enable)

该位用于使能通过一条双向线专门接收数据的单工通信。只接收模式激活时，使 BIDIMODE 位保持清零状态。

此位也适用于多从模式系统，在此类系统中，不会访问特定从器件，也不会损坏访问的从器件的输出。

0：全双工（发送和接收）

1：禁止输出（只接收模式）

注： ***该位不适用于 I^2S 模式***

位 9 **SSM**：软件从器件管理 (Software slave management)

当 SSM 位置 1 时，NSS 引脚输入替换为 SSI 位的值。

0：禁止软件从器件管理

1：使能软件从器件管理

注： ***该位不适用于 I^2S 模式和 SPI TI 模式***

位 8 **SSI**：内部从器件选择 (Internal slave select)

仅当 SSM 位置 1 时，此位才有效。此位的值将作用到 NSS 引脚上，并忽略 NSS 引脚的 IO 值。

注： ***该位不适用于 I^2S 模式和 SPI TI 模式***

位 7 ***LSBFIRST***：帧格式 (Frame format)

0：先发送 MSB

1：先发送 LSB

注：正在通信时不应更改此位。

该位不适用于 I^2S 模式和 SPI TI 模式

位 6 **SPE**：SPI 使能 (SPI enable)

0：禁止外设

1：使能外设

注： ***该位不适用于 I^2S 模式。***

禁止 SPI 时，请按照第 31.3.10 节：禁止 SPI 的步骤中所述的步骤操作。

位 [5:3] **BR[2:0]**：波特率控制 (Baud rate control)

000：$f_{PCLK}/2$

001：$f_{PCLK}/4$

010：$f_{PCLK}/8$

011：$f_{PCLK}/16$

100：$f_{PCLK}/32$

101：$f_{PCLK}/64$

110：$f_{PCLK}/128$

111：$f_{PCLK}/256$

注：正在通信时不应更改这些位。

这些位不适用于 I^2S 模式。

位 2 **MSTR**：主模式选择 (Master selection)
0：从配置
1：主配置
注：　正在通信时不应更改此位。
不适用于 I^2S 模式。

bit1 **CPOL**：时钟极性 (Clock Polarity)
0：空闲状态时，CK 为 0
1：空闲状态时，CK 为 1
注：　正在通信时不应更改此位。
该位不适用于 I^2S 模式和 SPI TI 模式。

位 0 **CPHA**：时钟相位 (Clock phase)
0：从第一个时钟边沿开始采样数据
1：从第二个时钟边沿开始采样数据
注：　正在通信时不应更改此位。
该位不适用于 I^2S 模式和 SPI TI 模式。

31.7.2 SPI 控制寄存器 2 (SPI_CR2)

SPI control register 2

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	FRF	Res.	*SSOE*	TXDMAEN	RXDMAEN
								rw	rw	rw	rw		rw	rw	rw

位 15:8 保留，必须保持复位值。

位 7 **TXEIE**：发送缓冲区空中断使能 (Tx buffer empty interrupt enable)
0：屏蔽 TXE 中断
1：使能 TXE 中断。TXE 标志置 1 时产生中断请求。

位 6 **RXNEIE**：接收缓冲区非空中断使能 (RX buffer not empty interrupt enable)
0：屏蔽 RXNE 中断
1：使能 RXNE 中断 RXNE 标志置 1 时产生中断请求。

位 5 **ERRIE**：错误中断使能 (Error interrupt enable)
此位用于在出现错误条件（SPI 模式下的 OVR、CRCERR、MODF 和 FRE 以及 I^2S 模式下的 UDR、OVR 和 FRE）时控制中断的生成。
0：屏蔽错误中断
1：使能错误中断

位 4 **FRF**：帧格式 (Frame format)
0：SPI Motorola 模式
1：SPI TI 模式
注：　该位不适用于 I^2S 模式。

位 3 Reserved 由硬件强制为零。

位 2 **SSOE**：SS 输出使能 (SS output enable)

0：在主模式下禁止 SS 输出，单元可在多主模式配置下工作

1：在主模式下且使能单元后使能 SS 输出。不能在多主模式环境下工作。

*注： 该位不适用于 **I^2S 模式和 SPI TI 模式。***

位 1 **TXDMAEN**：发送缓冲区 DMA 使能 (Tx buffer DMA enable)

当此位置 1 时，每当 TXE 标志置 1 时，即产生 DMA 请求。

0：禁止发送缓冲区 DMA

1：使能发送缓冲区 DMA

位 0 **RXDMAEN**：接收缓冲区 DMA 使能 (Rx buffer DMA enable)

当此位置 1 时，每当 RXNE 标志置 1 时，即产生 DMA 请求。

0：禁止接收缓冲区 DMA

1：使能接收缓冲区 DMA

31.7.3 SPI 状态寄存器 (SPI_SR)

SPI status register

偏移地址：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

位 15:9 Reserved 由硬件强制为零。

位 8 **FERR**：帧错误 (Frame Error)

0：无帧错误

1：发生帧错误

读取 SPI_SR 寄存器时，该位由硬件置 1，由软件清零。

无论选择哪种音频协议，该位均在 SPI TI 模式或 I2S 模式下使用。它会在从模式下的非预期时间检测 NSS 线或 WS 线的变化，通知外部主器件和从器件之间的同步失效情况。

*位 7 **BSY**：忙标志 (Busy flag)*

0：SPI（或 I2S）不繁忙

1：SPI（或 I2S）忙于通信或者发送缓冲区不为空

此标志由硬件置 1 和清零。

注： BSY 标志必须谨慎使用：请参见第 31.3.12 节：SPI 状态标志和第 31.3.10 节：禁止 SPI 的步骤。

位 6 **OVR**：上溢标志 (Overrun flag)

0：未发生上溢

1：发生上溢

此标志由硬件置 1，由软件序列复位。有关软件序列，请参见*第 31.3.13 节：SPI 错误标志*。

位 5 **MODF**：模式故障 (Mode fault)

0：未发生模式故障

1：发生模式故障

此标志由硬件置 1，由软件序列复位。有关软件序列，请参见*第 815 页的第 31.4 节*。

注： 该位不适用于 I^2S 模式

位 4 **CRCERR**：CRC 错误标志 (CRC error flag)

0：接收到的 CRC 值与 SPI_RXCRCR 值匹配

1：接收到的 CRC 值与 SPI_RXCRCR 值不匹配

此标志由硬件置 1，通过软件写入 0 来清零。

注： 该位不适用于 I^2S 模式。

位 3 **UDR**：下溢标志 (Underrun flag)

0：未发生下溢

1：发生下溢

此标志由硬件置 1，由软件序列复位。有关软件序列，请参见*第 31.6.7 节：I^2S 错误标志*。

注： 该位不适用于 SPI 模式。

位 2 **CHSIDE**：通道 (Channel side)

0：发送或接收左通道信息

1：发送或接收右通道信息

注： 该位不适用于 SPI 模式，在 PCM 模式下没有意义。

位 1 **TXE**：发送缓冲区为空 (Transmit buffer empty)

0：发送缓冲区非空

1：发送缓冲区为空

位 0 **RXNE**：接收缓冲区非空 (Receive buffer not empty)

0：发送缓冲区为空

1：接收缓冲区非空

31.7.4 SPI 数据寄存器 (SPI_DR)

SPI data register

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
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15:0 **DR[15:0]**：数据寄存器 (Data register)

已接收或者要发送的数据。

数据寄存器分为 2 个缓冲区，一个用于写入（发送缓冲区），一个用于读取（接收缓冲区）。对数据寄存器执行写操作时，数据将写入发送缓冲区，从数据寄存器执行读取时，将返回接收缓冲区中的值。

注： 这些注适用于 SPI 模式：

发送或接收的数据为 8 位或 16 位，具体取决于数据帧格式选择位（SPI_CR1 寄存器中的 DFF）。必须在使能 SPI 前进行此项选择，以确保操作正确。

对于 8 位数据帧，缓冲区为 8 位，只有寄存器的 LSB (SPI_DR[7:0]) 用于发送/接收。在接收模式下，寄存器的 MSB (SPI_DR[15:8]) 强制为 0。

对于 16 位数据帧，缓冲区为 16 位，整个寄存器 SPI_DR[15:0] 均用于发送/接收。

31.7.5 SPI CRC 多项式寄存器 (SPI_CRCPR)（不用于 I^2S 模式）

SPI CRC polynomial register

偏移地址：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 15:0 **CRCPOLY[15:0]**：CRC 多项式寄存器 (CRC polynomial register)

此寄存器包含用于 CRC 计算的多项式。

CRC 多项式 (0007h) 是此寄存器的复位值。可根据需要配置另一个多项式。

注： 这些位不适用于 I^2S 模式。

31.7.6 SPI RX CRC 寄存器 (SPI_RXCRCR)（不用于 I^2S 模式）

SPI RX CRC register

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
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0 **RXCRC[15:0]**：接收 CRC 寄存器 (Rx CRC register)

使能 CRC 计算后，RxCRC[15:0] 位将包含后续接收字节在计算后所得到的 CRC 值。当 SPI_CR1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

数据帧格式设置为 8 位数据（SPI_CR1 的 DFF 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。

选择 16 位数据帧格式（SPI_CR1 寄存器的 DFF 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。

注： 当 BSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。这些位不适用于 I^2S 模式。

31.7.7 SPI TX CRC 寄存器 (SPI_TXCRCR)（不用于 I^2S 模式）

SPI TX CRC register

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
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 15:0 **TXCRC[15:0]**：发送 CRC 寄存器 (Tx CRC register)

使能 CRC 计算后，TxCRC[7:0] 位将包含后续发送字节在计算后所得到的 CRC 值。当 SPI_CR1 寄存器中的 CRCEN 位写入 1 时，此寄存器复位。CRC 通过 SPI_CRCPR 寄存器中编程的多项式连续计算。

数据帧格式设置为 8 位数据（SPI_CR1 的 DFF 位清零）时，仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。

选择 16 位数据帧格式（SPI_CR1 寄存器的 DFF 位置 1）时，考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。

注：当 BSY 标志置 1 时，读取此寄存器可能返回一个不正确的值。
这些位不适用于 I^2S 模式。

31.7.8 SPI_I²S 配置寄存器 (SPI_I2SCFGR)

SPI_I²S configuration register

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
Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

位 15:12 保留，必须保持复位值。

位 11 **I2SMOD**：I2S 模式选择 (I2S mode selection)

0：选择 SPI 模式

1：选择 I2S 模式

注： 应在 SPI 或 I²S 禁止时配置此位。

位 10 **I2SE**：I2S 使能 (I2S Enable)

0：禁止 I²S 外设

1：使能 I²S 外设

注： 该位不适用于 SPI 模式。

位 9:8 **I2SCFG**：I2S 配置模式 (I2S configuration mode)

00：从模式 - 发送

01：从模式 - 接收

10：主模式 - 发送

11：主模式 - 接收

注： 应在 I²S 禁止时配置此位。

该位不适用于 SPI 模式。

位 7 **PCMSYNC**：PCM 帧同步 (PCM frame synchronization)

0：短帧同步

1：长帧同步

注： 只有在 I2SSTD = 11（使用 PCM 标准）时，此位才有意义

该位不适用于 SPI 模式。

位 6 保留：由硬件强制为 0

位 5:4 **I2SSTD**：I2S 标准选择 (I2S standard selection)

00：I²S Philips 标准。

01：MSB 对齐标准（左对齐）

10：LSB 对齐标准（右对齐）

11：PCM 标准

有关 I²S 标准的详细信息，请参见*第 820 页的第 31.6.2 节**不适用于 SPI 模式。*

注： 为确保正确运行，应在 I²S 禁止时配置这些位。

位 3 **CKPOL**：空闲状态时钟极性 (Steady state clock polarity)

0：I^2S 时钟空闲状态为低电平

1：I^2S 时钟空闲状态为高电平

注：　为确保正确运行，应在 I^2S 禁止时配置此位。

该位不适用于 SPI 模式。

位 2:1 **DATLEN**：要传输的数据长度 (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：　为确保正确运行，应在 I^2S 禁止时配置这些位。

该位不适用于 SPI 模式。

位 0 **CHLEN**：通道长度（每个音频通道的位数）(Channel length (number of bits per audio channel))

0：16 位

1：32 位

只有在 DATLEN = 00 时，此位的写操作才有意义，否则无论写入何值，通道长度始终由硬件固定为 32 位。*不适用于 SPI 模式。*

注：　为确保正确运行，应在 I^2S 禁止时配置此位。

31.7.9 SPI_I^2S 预分频器寄存器 (SPI_I2SPR)

SPI_I2S prescaler register

偏移地址：0x20

复位值：0000 0010 (0x0002)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
						rw	rw	rw							

位 [15:10] 保留，必须保持复位值。

位 9 **MCKOE**：主时钟输出使能 (Master clock output enable)

0：禁止主时钟输出

1：使能主时钟输出

注：　应在 I^2S 禁止时配置此位。只有在 I^2S 为主模式时，才会使用此位。

该位不适用于 SPI 模式。

位 8 **ODD**：预分频器的奇数因子 (Odd factor for the prescaler)

0：实际分频值为 = I2SDIV *2

1：实际分频值为 = (I2SDIV * 2)+1

请参见*第 826 页的第 31.6.3 节*。*不适用于 SPI 模式。*

注：　应在 I^2S 禁止时配置此位。只有在 I^2S 为主模式时，才会使用此位。

位 7:0 **I2SDIV**：I2S 线性预分频器 (I2S Linear prescaler)

I2SDIV [7:0] = 0 或 I2SDIV [7:0] = 1 为禁用值。

请参见*第 826 页的第 31.6.3 节*。*不适用于 SPI 模式。*

注：　应在 I^2S 禁止时配置这些位。只有在 I^2S 为主模式时，才会使用此位。

31.7.10 SPI 寄存器映射

下表显示了 SPI 寄存器映射和复位值。

表 142. SPI 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	FRF	Res.	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	**SPI_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	**SPI_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPI_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	**SPI_I2SPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

32 通用串行总线全速设备接口 (USB)

32.1 简介

USB 外设在全速 USB 2.0 总线和 APB 总线之间实现了一个接口。

此外设支持 USB 挂起/恢复，这样可以停止设备时钟以降低功耗。

32.2 USB 主要特性

- 符合 USB 2.0 全速规范
- 可配置端点数为 1 个到 8 个
- 最多 1024 字节专用数据包缓冲区存储器 SRAM
- 循环冗余校验 (CRC) 生成/校验、非归零反相 (NRZI) 编码/解码以及位填充
- 同步传输支持
- 双缓冲批量/同步端点支持
- USB 挂起/恢复操作
- 帧锁定时钟脉冲生成
- USB 2.0 链路层电源管理支持
- 电池充电规范第 1.2 版支持
- USB 连接/断开功能（USB_DP 线上可控制的内置上拉电阻）

32.3 USB 实现

表 143 列出了器件中的 USB 实现。

表 143. STM32L0x3USB 实现

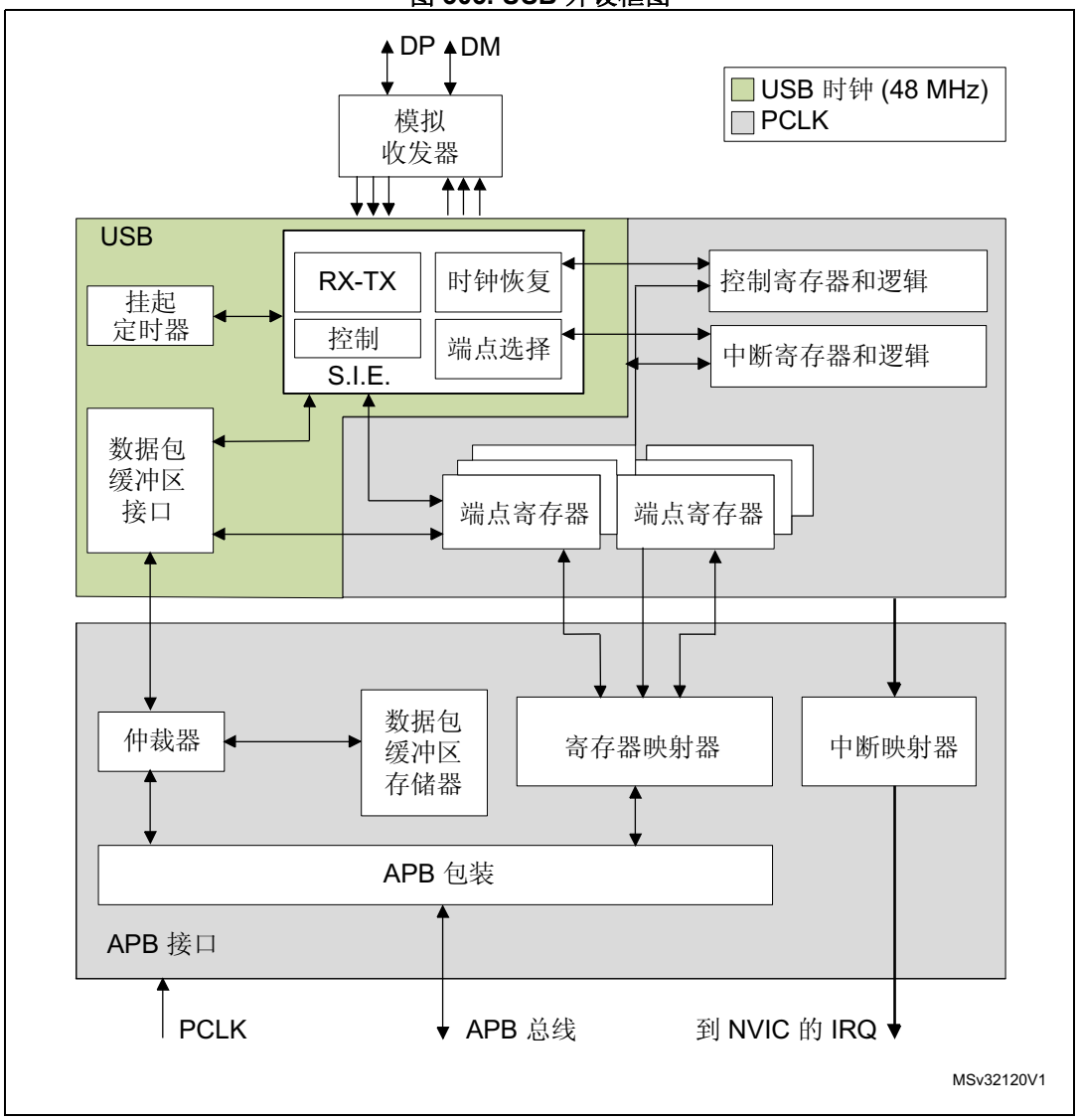
USB 特性(1)	USB
端点数	8
专用数据包缓冲区存储器 SRAM 的大小	1024 字节
专用数据包缓冲区存储器 SRAM 访问模式	2 x 16 位/字
USB 2.0 链路层电源管理 (LPM) 支持	X
电池充电检测 (BCD) 支持	X
USB_DP 线上的内置上拉电阻	X

1. X = 支持

32.4 USB 功能说明

图 305 显示了 USB 外设的框图。

图 305. USB 外设框图



USB 外设在主机 PC 和微控制器实现的功能之间提供符合 USB 规范的连接。主机 PC 和系统存储器之间通过专用数据包缓冲区存储器进行数据传输，此数据包缓冲区存储器通过 USB 外设直接访问。此专用存储器最大为 1024 字节，最多可使用 16 个单向端点或 8 个双向端点。USB 外设与 USB 主机相连，按 USB 标准的要求检测令牌数据包、处理数据发送/接收以及处理握手包。事务格式化通过硬件来执行，其中包括 CRC 生成和校验。

每个端点都与一个缓冲区描述块关联，此描述块指示与端点相关的存储区的位置、大小或必须传送的字节数。当 USB 外设识别到有效功能/端点对的令牌时，将根据需要进行相关的数据传输（如果已配置端点）。USB 外设缓冲的数据装载到内部 16 位寄存器中，并且会对专用缓冲区执行存储器访问。当所有数据传输完成后，必要时，可根据传输方向生成或等待适当的 USB 握手包。

事务结束时将生成端点相关中断，以读取状态寄存器和/或使用不同的中断响应程序。微控制器可确定：

- 必须处理哪个端点
- 出现错误时发生了哪类事务（位填充、格式化、CRC、协议、ACK 丢失和上溢/下溢等）。

为同步传输和高吞吐量批量传输提供特殊支持，实现双缓冲区用法，从而确保当微控制器使用其中一个缓冲区时，始终有另一个缓冲区可供 USB 外设使用。

必要时，可通过写入控制寄存器将单元置于低功耗模式（挂起模式）。此时，可避免产生所有静态功耗，USB 时钟会减速或停止。如果在低功耗模式下检测到 USB 输入上存在活动，则会异步唤醒设备。可以将特殊中断源直接连接到唤醒线，从而使系统立即重启正常的时钟生成操作和/或支持直接时钟启动/停止。

32.4.1 USB 模块说明

USB 外设实现了与 USB 连接相关的所有功能，具体包括以下模块：

- USB 物理接口 (USB PHY)：此模块维持与外部 USB 主机间的电气连接。它包含差分模拟收发器本身、可控制的内置上拉电阻（连接到 USB_DP 线）以及电池充电检测 (BCD) 支持，在相同 USB_DP 和 USB_DM 线上复用。模拟收发器的输出使能控制信号（低电平有效）通过 USB_NOE 从外部提供。它可用于驱动某活动 LED 或提供有关其他电路实际通信方向的信息。
- 串行接口引擎 (SIE)：此模块的功能包括同步模式识别、位填充、CRC 生成和校验、PID 验证/生成以及握手评估。它必须与 USB 收发器连接，并使用数据包缓冲区接口提供的虚拟缓冲区来存储本地数据。此单元还会根据 USB 外设事件（例如，帧起始 (SOF)、USB_Reset 和数据错误）和端点相关事件（例如，数据包发送结束或接收成功）生成相应信号，这些信号随后用于生成中断。
- 定时器：此模块生成帧起始锁定时钟脉冲并在已有 3 ms 未接收到通信数据时检测全局挂起（来自主机）。
- 数据包缓冲区接口：此模块管理本地存储器，能够以灵活的方式实现一组用于发送和接收的缓冲区。它可根据来自 SIE 的请求选择合适的缓冲区，并将这些请求置于端点寄存器指向的存储器地址。在数据包交换过程中，每交换一个字节，地址便会递增一次，这样便可跟踪已交换的字节数，避免超出缓冲区的最大容量。
- 端点相关寄存器：每个端点有一个相关的寄存器，其中包含端点类型及其当前状态。对于单向/单缓冲区端点，可使用单个寄存器实现两个不同的端点。这类寄存器有 8 个，最多支持 16 个单向/单缓冲区端点或 7 个双缓冲区端点，并且支持这两种端点的任意组合。例如，可以将 USB 外设编程为具有 4 个双缓冲区端点和 8 个单缓冲区/单向端点。
- 控制寄存器：这类寄存器包含与整个 USB 外设的状态相关的信息，用于强制执行某些 USB 事件（例如，恢复和掉电）。
- 中断寄存器：这类寄存器包含中断屏蔽和事件记录。它们可用于查询中断原因、中断状态或清除挂起中断的状态。

注： ** 端点 0 始终用于单缓冲区模式下控制传输。*

USB 外设通过 APB 接口连接到 APB 总线，此接口包含以下模块：

- 数据包存储器：此存储器是在物理上包含数据包缓冲区的本地存储器。可通过数据包缓冲区接口使用此存储器，以创建数据结构并且可通过应用软件直接访问。数据包存储器最大为 1024 字节，以 512 个 16 位半字的结构存在。

- 仲裁器：此模块接受来自 APB 总线和 USB 接口的存储器请求。它通过为 APB 访问分配高优先级来解决冲突，同时始终保留一半存储器带宽来完成所有 USB 传输。该时分双工方案实现了虚拟双端口 SRAM，允许在发生 USB 事务时进行存储器访问。此方案还支持任意长度的 APB 传输。
- 寄存器映射器：此模块收集 USB 外设不同字节宽和位宽的寄存器，这些寄存器采用 16 位宽半字组的形式，通过 APB 寻址。
- APB 封装器：此模块为存储器和寄存器提供连接 APB 的接口。它还将整个 USB 外设映射到 APB 地址空间。
- 中断映射器：此模块用于选择可能的 USB 事件如何生成中断以及将中断映射到 NVIC。

32.5 编程注意事项

以下各节将介绍 USB 外设与应用软件之间的预期交互，帮助简化应用软件开发。

32.5.1 通用 USB 设备编程

本部分将介绍应用软件需要执行哪些主要任务才能获得符合 USB 规范的行为。与大多数常规 USB 事件相关的操作均在考虑范围之内，并且有相关段落专门说明双缓冲端点和同步传输这两个特例。除系统复位外，操作始终在下述其中一个 USB 事件的驱动下由 USB 外设发起。

32.5.2 系统复位和上电复位

系统复位和上电复位后，应用软件应执行的第一个操作是为 USB 外设提供所需的全部时钟信号，随后使其复位信号无效，以便能够访问其寄存器。下文将介绍整个初始化序列。

首先，应用软件需要使用设备时钟管理逻辑所提供的相关控制位激活寄存器宏单元时钟并使宏单元特定的复位信号无效。

随后，必须使用 CNTR 寄存器的 PDWN 位接通设备中与 USB 收发器相关的模拟部分，这一步需要特殊处理。此位旨在接通为端口收发器供电的内部参考电压。此电路有一段定义的启动时间（数据手册中指定的 $t_{STARTUP}$），这段时间内未定义 USB 收发器的行为。因此，在将 CNTR 寄存器中的 PDWN 位置 1 后，必须等待这段时间结束时才能消除 USB 部分的复位条件（通过将 CNTR 寄存器中的 FRES 位清零）。清除 ISTR 寄存器会在任何其他宏单元操作使能之前，消除所有意外的挂起中断。

系统复位时，微控制器必须初始化所需的全部寄存器和数据包缓冲区描述表，以便 USB 外设能够正常地产生中断和进行数据传输。对于非特定于任何端点的所有寄存器，必须按照应用软件的需求（使能中断的选择，所选的数据包缓冲区地址等）进行初始化。对于 USB 复位情况，过程将继续进行（请参见下一段）。

USB 复位（RESET 中断）

发生该事件时，USB 外设的状态与系统复位后的初始化状态（如上一段所述）相同：所有端点寄存器中均禁止通信（USB 外设不会响应任何数据包）。作为对 USB 复位事件的响应，必须使能 USB 功能（USB 地址为 0），仅实现默认控制端点（端点地址也为 0）。具体实现方法是：将 USB_DADDR 寄存器的使能功能 (EF) 位置 1 并相应地初始化 EP0R 寄存器及其相关的数据包缓冲区。在 USB 枚举过程中，主机会为此设备分配一个唯一地址（该地址必须写入 USB_DADDR 寄存器的 ADD[6:0] 位）并配置任何其他必要的端点。

当接收到 RESET 中断时，应用软件负责在触发中断的复位序列结束后的 10 ms 内再次使能 USB 功能 0 的默认端点。

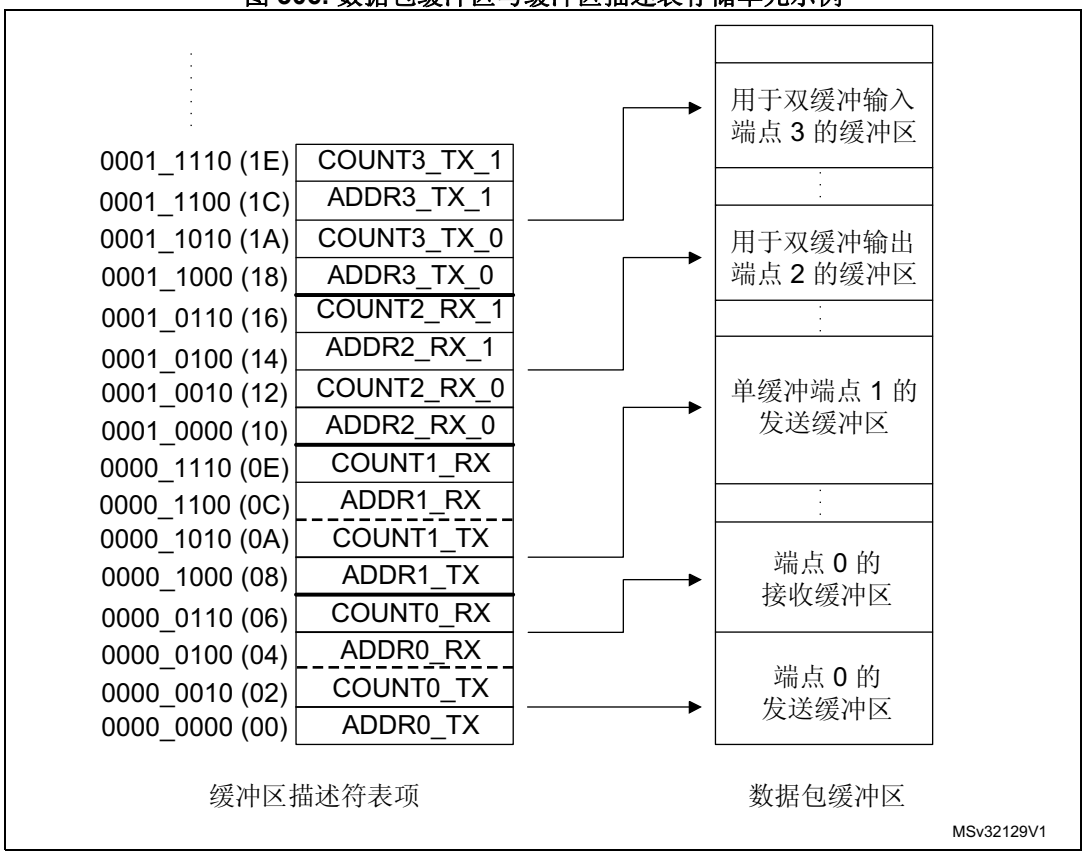
数据包缓冲区的结构和用法

每个双向端点均可从主机接收数据或向主机发送数据。端点接收的数据存储在为该端点保留的专用存储器缓冲区内，而另一个存储器缓冲区则包含该端点要发送的数据。对该存储器的访问通过数据包缓冲区接口模块来执行，该模块会发出存储器访问请求并等待应答。由于数据包缓冲区存储器还必须供微处理器访问，因此需通过仲裁逻辑来处理访问冲突：半个 APB 周期供微控制器访问，剩下的半个周期供 USB 外设访问。通过这种方式，微控制器和 USB 外设可将数据包存储器视为双端口 SRAM 来执行操作，而无需考虑任何冲突，即使微控制器正在执行背靠背访问也如此。USB 外设逻辑使用专用时钟。此专用时钟的频率根据 USB 标准的要求固定为 48 MHz，这可能与用于 APB 总线接口的时钟不同。可采用不同的时钟配置，即 APB 时钟频率可高于或低于 USB 外设的时钟频率。

注： *根据 USB 数据速率和数据包存储器接口要求，必须确保 APB 时钟的频率不低于 10 MHz，以避免发生数据上溢/下溢问题。*

每个端点与两个数据包缓冲区关联（通常，一个用于发送，另一个用于接收）。缓冲区可置于数据包存储器内的任何位置，因为其位置和大小在缓冲区描述表中指定。此描述表也位于数据包存储器中，地址由 USB_BTABLE 寄存器指定。每个表条目与一个端点寄存器关联并且由四个 16 位半字组成，因此表起始地址必须始终与 8 字节边界对齐（USB_BTABLE 寄存器的低三位始终为“000”）。*第 32.6.2 节：缓冲区描述符表*介绍了缓冲区描述符表条目。对于一个既非同步端点也非双缓冲批量端点的单向端点，只需要一个数据包缓冲区（与所支持的传输方向相关的数据包缓冲区）。与不支持的传输方向或未使用端点相关的其他表存储单元可供用户使用。同步端点和双缓冲批量端点的数据包缓冲区处理方式比较特殊（请分别参见*第 32.5.4 节：同步传输*和*第 32.5.3 节：双缓冲端点*）。*图 306* 给出了缓冲区描述表条目和数据包缓冲区之间的关系。

图 306. 数据包缓冲区与缓冲区描述表存储单元示例



在接收或发送期间，每个数据包缓冲区从底部开始使用。USB 外设始终不会更改与所分配存储器缓冲区相邻的存储单元的内容；如果接收的数据包长度大于所分配缓冲区的长度（缓冲区上溢条件），则到达最后一个可用的存储单元后便停止将数据复制到存储器。

端点初始化

初始化端点的第一步是向 ADDRn_TX/ADDRn_RX 寄存器写入相应值，以使 USB 外设获悉要发送的数据已准备就绪，要接收的数据可进行缓冲。必须根据端点类型设置 USB_EPnR 寄存器中的 EP_TYPE 位，最终使用 EP_KIND 位使能所需的任何特殊功能。在发送端，必须通过 USB_EPnR 寄存器中的 STAT_TX 位使能端点，并且必须初始化 COUNTn_TX。接收时，必须将 STAT_RX 位置 1 以使能接收，并且必须通过 BL_SIZE 和 NUM_BLOCK 字段向 COUNTn_RX 写入所分配的缓冲区大小。单向端点（同步端点和双缓冲批量端点除外）只需初始化与所支持方向相关的位和寄存器。使能发送与/或接收后，不得通过应用软件修改寄存器 USB_EPnR 以及存储单元 ADDRn_TX/ADDRn_RX 和 COUNTn_TX/COUNTn_RX（分别地），因为硬件会实时更改其值。数据传输操作完成时（通过 CTR 中断事件通知），可再次访问这些存储单元以重新使能新操作。

IN 数据包（数据传送）

接收 IN 令牌数据包时，如果所接收的地址与已配置的有效端点匹配，则 USB 外设会访问与所寻址端点相关的缓冲区描述符表条目内的 ADDRn_TX 和 COUNTn_TX 存储单元的内容。这些存储单元的内容存储在其内部 16 位寄存器 ADDR 和 COUNT 中（无法通过软件访问）。数据包存储器会被再次访问以读取要发送的第一个字节（请参见*第 848 页的“数据包缓冲区的结构和用法”*），然后会根据 USB_EPnR 位 DTOG_TX 开始发送 DATA0 或 DATA1 PID。当 PID 发送完成时，从缓冲区存储器读取的第一个字节将被装载到输出移位寄存器中以通过 USB 总线发送出去。发送完最后一个数据字节后，将发送计算得出的 CRC。如果寻址的端点无效，将根据 USB_EPnR 寄存器中的 STAT_TX 位发送 NAK 或 STALL 握手包而非数据包。

ADDR 内部寄存器用作当前缓冲区存储单元的指针，而 COUNT 用于对要发送的剩余字节计数。从数据包缓冲区存储器读取的每个半字均通过 USB 总线从最低有效字节开始发送。在发送缓冲区存储器中从 ADDRn_TX 指向的地址开始读取 COUNTn_TX/2 个半字。如果发送的数据包由奇数个字节组成，则仅会使用所访问的最后一个半字的低半部分。

接收到主机的 ACK 后，USB_EPnR 寄存器将通过以下方式更新：DTOG_TX 位翻转、通过设置 STAT_TX=10 (NAK) 将端点置为无效、将 CTR_TX 位置 1。应用软件必须首先识别端点，即请求微控制器关注以检查 USB_ISTR 寄存器中的 EP_ID 和 DIR 位。处理 CTR_TX 事件会开始清零中断位；应用软件随后会准备另一个装满数据的发送缓冲区，用下一次传输期间要发送的字节数更新 COUNTn_TX 表存储单元，最后将 STAT_TX 设置为“11”(VALID) 以重新使能发送。当 STAT_TX 位等于“10”(NAK) 时，寻址到相应端点的任何 IN 请求都得不到应答，并且指示流控制状态：USB 主机将重试事务，直到成功。必须按上述顺序执行操作序列，以避免与触发 CTR 中断的 IN 事务寻址到同一端点且紧随其后的第二个 IN 事务的通知丢失。

OUT 和 SETUP 数据包（数据接收）

USB 外设对这两种令牌的处理方式大致相同，SETUP 数据包不同于 OUT 数据包的处理将在下文有关控制传输的段落中详细说明。接收 OUT/SETUP PID 时，如果地址与有效端点匹配，则 USB 外设会访问与所寻址端点相关的缓冲区描述符表条目内的 ADDRn_RX 和 COUNTn_RX 存储单元的内容。ADDRn_RX 的内容直接存储在其内部寄存器 ADDR 中。而此时 COUNT 复位，BL_SIZE 和 NUM_BLOCK 位域的值（从 COUNTn_RX 内容中读取）用于初始化内部 16 位计数器 BUF_COUNT，该计数器用于检查缓冲区上溢条件（上述所有

内部寄存器均无法通过软件访问）。USB 外设后续接收的数据字节均以半字为单位打包（接收的第一个字节作为最低有效字节存储），随后传输到从内部 ADDR 寄存器包含的地址开始的数据包缓冲区，每次传输字节时 BUF_COUNT 递减、COUNT 递增。当检测到 DATA 数据包结束时，将检验所接收的 CRC 是否正确，只有当接收期间没有错误时，才会将 ACK 握手包发送回发送主机。

当出现 CRC 错误或其他类型的错误（位填充超限、帧错误等）时，数据字节仍将在数据包存储器缓冲区中进行复制，至少到错误检测点为止，但不会发送 ACK 数据包且 USB_ISTR 寄存器中的 ERR 位置 1。不过，这种情况下通常不需要软件操作：USB 外设会从接收错误中恢复，并保持就绪状态等待下一个事务的到来。如果寻址的端点无效，将根据 USB_EPnR 寄存器中的 STAT_RX 位发送 NAK 或 STALL 握手包而非 ACK，并且不会向接收存储器缓冲区中写入任何数据。

接收存储器缓冲区存储单元从 ADDRn_RX 中包含的地址开始写入，写入的字节数与接收的数据包长度（包括 CRC，即数据有效负载长度 + 2）相对应，或者一直写到分配的最后一个存储单元（由 BL_SIZE 和 NUM_BLOCK 定义），以先结束者为准。凭借这种方法，USB 外设写入的数据始终不会超出所分配接收存储器缓冲区的容量。如果数据包有效负载的长度（应用软件实际使用的字节数）大于所分配的缓冲区，则 USB 外设会检测到缓冲区上溢条件。在这种情况下，将发送 STALL 握手而非常规的 ACK 来通知主机出现问题，此时不会产生中断，事务将被视作失败。

当事务正确完成后（通过发送 ACK 握手包），内部 COUNT 寄存器将复制回缓冲区描述表条目内的 COUNTn_RX 存储单元。BL_SIZE 和 NUM_BLOCK 字段不受影响，这两个字段通常无需重写，USB_EPnR 寄存器通过以下方式更新：DTOG_RX 位翻转、通过设置 STAT_RX = “10” (NAK) 将端点置为无效，将 CTR_RX 位置 1。如果事务因错误或缓冲区上溢条件而失败，则上面列出的所有操作均不会发生。应用软件必须首先识别端点，即请求微控制器关注以检查 USB_ISTR 寄存器中的 EP_ID 和 DIR 位。首先确定事务类型（USB_EPnR 寄存器中的 SETUP 位）以处理 CTR_RX 事件；应用软件必须清零中断标志位，并通过读取与正在处理的端点相关的缓冲区描述表条目内的 COUNTn_RX 存储单元获得所接收字节的数量。处理完接收的数据后，应用软件应将 USB_EPnR 中的 STAT_RX 位设置为“11”（有效），从而允许其他事务。当 STAT_RX 位等于“10” (NAK) 时，寻址到相应端点的任何 OUT 请求都得不到应答，并且指示流控制状态：USB 主机将重试事务，直到成功。必须按上述顺序执行操作序列，以避免丢失与触发 CTR 中断的 OUT 事务寻址到同一端点且紧随其后的第二个 IN 事务的通知。

控制传输

控制传输包含一个 SETUP 事务，其后跟有零个或多个数据阶段（全部数据阶段均采用相同的方向），再后面是一个状态阶段（一个零字节反向传输）。SETUP 事务仅由控制端点来处理，并且与 OUT 事务十分相似（数据接收），不同之处在于所寻址端点寄存器的 DTOG_TX 和 DTOG_RX 位的值分别设置为 1 和 0 以初始化控制传输，STAT_TX 和 STAT_RX 均设置为“10” (NAK) 以允许软件决定后续事务必须为 IN 还是 OUT，具体取决于 SETUP 的内容。控制端点必须在每个 CTR_RX 事件时检查 USB_EPnR 寄存器中的 SETUP 位，以区分正常的 OUT 事务与 SETUP 事务。USB 设备可通过解析 SETUP 阶段传输的数据来确定数据阶段的数量和方向，并用于在出现错误时停止事务。为此，在最后一个数据阶段前的所有数据阶段中，应将未使用的方向设置为 STALL，以便在主机过于迅速地改变传输方向时获取 STALL 作为状态阶段。

使能最后一个数据阶段时，应将相反方向设置为 NAK，以便在主机立即改变传输方向（以执行状态阶段）时一直等待控制操作完成。如果控制操作成功完成，软件会将 NAK 更改为 VALID，否则将更改为 STALL。同时，如果状态阶段为 OUT，则应将 STATUS_OUT（USB_EPnR 寄存器中的 EP_KIND）位置 1，以便在执行数据非零的状态事务时生成错误。当处理状态事务时，应用软件会将 STATUS_OUT 位清零，并将 STAT_RX 设置为 VALID（以接受新命令）、将 STAT_TX 设置为 NAK（以在下一次SETUP后立即延迟一段可能的状态阶段）。

由于 USB 规范规定 SETUP 数据包不可通过 ACK 以外的握手包来应答（这最终将中止之前发出的命令而启动新命令），USB 逻辑不允许控制端点通过 NAK 或 STALL 握手包来应答从主机接收的 SETUP 令牌。

当 STAT_RX 位设置为“01”(STALL) 或“10”(NAK) 且接收到 SETUP 令牌时，USB 将接受数据（以执行所需的数据传输）并发回 ACK 握手包。如果该端点之前发出的 CTR_RX 请求尚未得到应用软件的应答（即 CTR_RX 位仍因之前完成的接收操作而置 1），USB 将丢弃 SETUP 事务且不会通过任何握手包进行应答（无论其状态如何均如此），从而模拟接收错误并强制主机再次发送 SETUP 令牌。这样做的目的在于避免与触发 CTR 中断的 IN 事务寻址到同一端点且紧随其后的第二个 IN 事务的通知丢失。

32.5.3 双缓冲端点

USB 标准定义的所有不同端点类型均代表不同的通信模型，并且说明了各类数据传输操作的典型要求。当主机 PC 和 USB 功能之间要传输大量数据时，批量端点类型是最适合的模型。这是因为主机会调度批量事务以填充帧中的所有可用带宽，因此只要 USB 功能就绪处理寻址到它的批量事务，便可最大程度地提高实际传输速率。如果在下一个事务到达时，USB 功能仍忙于上一个事务，则它将通过 NAK 握手进行应答。主机 PC 将再次发出同一事务，直到 USB 功能准备好处理该事务。由于重复发送占用带宽，因此实际传输速率会降低。为此，可为批量端点搭配使用称为“双缓冲”的专用功能。

激活“双缓冲”功能时，数据翻转序列用于选择 USB 外设将使用哪个缓冲区来执行所需的数据传输，使用“发送”和“接收”数据包存储区在每个事务成功完成时管理缓冲区交换，以便在 USB 外设填充一个缓冲区时，始终有另一个完整的缓冲区可供应用软件使用。例如，在指向“接收”双缓冲批量端点的 OUT 事务期间，当一个缓冲区用来自 USB 主机的新数据填充时，另一个缓冲区可供微控制器软件使用（同样适用于“发送”双缓冲批量端点和 IN 事务）。

由于交换缓冲区管理需要使用全部 4 个缓冲区描述表存储单元来管理地址指针和已分配存储区的长度，因此用于实现双缓冲批量端点的 USB_EPnR 寄存器被强制用作单向寄存器。因此，只有一个 STAT 位对必须设置为“00”（禁止）以外的值：将双缓冲批量端点使能用于接收时为 STAT_RX；将双缓冲批量端点使能用于发送时为 STAT_TX。如果需要将双缓冲批量端点使能同时用于发送和接收，则必须使用两个 USB_EPnR 寄存器。

要利用双缓冲功能达到最高传输速率，必须修改前几章中介绍的端点流控制结构，以便只在 USB 外设和应用软件之间发生缓冲区冲突时（而非每次事务成功结束时），将端点状态切换为 NAK。USB 外设当前使用的存储器缓冲区由与端点方向相关的 DTOG 位定义：DTOG_RX（USB_EPnR 寄存器的位 14）用于“接收”双缓冲批量端点；DTOG_TX（USB_EPnR 寄存器的位 6）用于“发送”双缓冲批量端点。为实现新的流控制方案，USB 外设应知道应用软件当前使用哪个数据包缓冲区，以便获悉是否存在任何冲突。由于 USB_EPnR 寄存器中有两个 DTOG 位，但只有一个位可供 USB 外设用于数据和缓冲区排序（由于双缓冲功能所要求的单向限制），因此应用软件可使用另一个位显示当前使用哪个缓冲区。这种新的缓冲区标志称为 SW_BUF。下表给出了对于“发送”和“接收”双缓冲批量端点，USB_EPnR 寄存器位与 DTOG/SW_BUF 定义之间的对应关系。

表 144. 双缓冲缓冲区标志定义

缓冲区标志	“发送”端点	“接收”端点
DTOG	DTOG_TX（USB_EPnR 位 6）	DTOG_RX（USB_EPnR 位 14）
SW_BUF	USB_EPnR 位 14	USB_EPnR 位 6

USB 外设当前使用的存储器缓冲区由 DTOG 缓冲区标志定义，而应用软件当前使用的缓冲区由 SW_BUF 缓冲区标志定义。对于“发送”和“接收”这两种情况，缓冲区标志值与使用的数据包缓冲区之间的关系是相同的，如下表所列。

表 145. 批量双缓冲存储器缓冲区用法

端点类型	DTOG	SW_BUF	USB 外设使用的数据包缓冲区	应用软件使用的数据包缓冲区
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。	ADDRn_TX_1 / COUNTn_TX_1 缓冲区描述表存储单元。
	1	0	ADDRn_TX_1 / COUNTn_TX_1 缓冲区描述表存储单元	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。
	0	0	无(1)	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。
	1	1	无(1)	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 缓冲区描述表存储单元。	ADDRn_RX_1 / COUNTn_RX_1 缓冲区描述表存储单元。
	1	0	ADDRn_RX_1 / COUNTn_RX_1 缓冲区描述表存储单元。	ADDRn_RX_0 / COUNTn_RX_0 缓冲区描述表存储单元。
	0	0	无(1)	ADDRn_RX_0 / COUNTn_RX_0 缓冲区描述表存储单元。
	1	1	无(1)	ADDRn_RX_1 / COUNTn_RX_1 缓冲区描述表存储单元。

1. 端点处于 NAK 状态。

批量端点的双缓冲功能通过以下方式激活：

- 在其 USB_EPnR 寄存器的 EP_TYPE 位域中写入“00”，以将端点定义为批量，
- 将同一寄存器中的 EP_KIND 位设置为“1”(DBL_BUF)。

应用软件负责根据使用的第一个缓冲区初始化 DTOG 和 SW_BUF 位；考虑到这两位具有特殊的仅-位翻转属性，必须执行该操作。如果第一个事务在 DBL_BUF 置 1 后结束，则将触发双缓冲批量端点的特殊流控制，该特殊流控制用于寻址到该端点的所有其他事务，直到 DBL_BUF 保持置 1。每个事务结束时，所寻址端点 USB_EPnR 寄存器的 CTR_RX 或 CTR_TX 位将置 1，具体取决于使能的方向。同时，USB_EPnR 寄存器中受影响的 DTOG 位由硬件位翻转，从而使 USB 外设缓冲区在完全不依赖软件的情况下实现交换。与普通事务以及 DBL_BUF 置 1 后的第一个事务不同，STAT 位对不受事务终止的影响，其值保持为“11”（有效）。不过，当接收到新事务的令牌数据包时，如果检测到 USB 外设和应用软件之间存在缓冲区冲突（此时，DTOG 和 SW_BUF 的值相同，请参见*第 852 页的表 145*），则实际的端点状态将被屏蔽为“10”(NAK)。应用软件通过清零中断标志并开始对完成的事务进行任何所需处理来响应 CTR 事件通知。当应用软件数据包缓冲区使用完毕后，软件将翻转 SW_BUF 位，向该位写入“1”以通知 USB 外设该缓冲区当前可用。通过这种方式，不应答事务的数量仅受应用软件准备事务数据的时间限制：如果准备时间短于在 USB 总线上完成一个事务所需的时间，则将不会发生因流控制引起的重新发送，实际传输速率将仅受主机 PC 的限制。

应用软件可始终覆盖为双缓冲批量端点实现的特殊流控制，向相关 USB_EPnR 寄存器的 STAT 位对写入“11”（有效）以外的明确状态。在这种情况下，USB 外设将始终使用编程的端点状态，而与缓冲区使用情况无关。

32.5.4 同步传输

USB 标准支持需要固定且精确的数据生产/消费频率的全速外设，这种通信定义为“同步”。该数据的典型示例包括：音频采样、压缩视频流以及通常对传输频率精度有严苛要求的任何一种采样数据。当端点在枚举阶段定义为“同步”时，主机将在帧中分配所需带宽并在每个帧中严密地传输一个 IN 或 OUT 帧，具体取决于端点方向。为限制带宽需求，同步通信中不允许重新发送失败的事务，这将导致以下情况：同步事务没有握手阶段，数据包之后不等待或发送任何 ACK 包。出于同样的原因，同步传输不支持数据位翻转，始终使用 DATA0 PID 来启动任何数据包。

端点的同步行为通过将其 USB_EPnR 寄存器中的 EP_TYPE 位设置为“10”来选择；由于没有握手阶段，STAT_RX/STAT_TX 位对的合法值只有“00”（禁止）和“11”（有效），任何其他值都将产生不符合 USB 标准的结果。同步端点通过实现双缓冲来简化应用软件的开发，使用“发送”和“接收”数据包存储区在每个事务成功时管理缓冲区交换，以便在 USB 外设填充一个缓冲区时，始终有另一个完整的缓冲区可供应用软件使用。

根据*表 146*，USB 外设当前使用的存储器缓冲区由与端点方向相关的 DTOG 位定义（DTOG_RX 用于“接收”同步端点，DTOG_TX 用于“发送”同步端点，二者均位于相关的 USB_EPnR 寄存器中）。

表 146. 同步存储器缓冲区的用法

端点类型	DTOG 位值	USB 外设使用的数据包缓冲区	应用软件使用的数据包缓冲区
IN	0	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。	ADDRn_TX_1 / COUNTn_TX_1 缓冲区描述表存储单元。
	1	ADDRn_TX_1 / COUNTn_TX_1 缓冲区描述表存储单元。	ADDRn_TX_0 / COUNTn_TX_0 缓冲区描述表存储单元。
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 缓冲区描述表存储单元。	ADDRn_RX_1 / COUNTn_RX_1 缓冲区描述表存储单元。
	1	ADDRn_RX_1 / COUNTn_RX_1 缓冲区描述表存储单元。	ADDRn_RX_0 / COUNTn_RX_0 缓冲区描述表存储单元。

与双缓冲批量端点一样，用于实现同步端点的 USB_EPnR 寄存器被强制用作单向寄存器。如果需要将同步端点使能同时用于发送和接收，则必须使用两个 USB_EPnR 寄存器。

应用软件负责根据使用的第一个缓冲区初始化 DTOG 位；考虑到这两位具有特殊的仅-位翻转属性，必须执行该操作。每个事务结束时，所寻址端点 USB_EPnR 寄存器的 CTR_RX 或 CTR_TX 位将置 1，具体取决于使能的方向。同时，USB_EPnR 寄存器中受影响的 DTOG 位由硬件同步，从而使缓冲区在完全不依赖软件的情况下实现交换。STAT 位对不受事务完成的影响，因为同步传输缺少握手阶段，没有可用的流控制，端点始终保持“11”（有效）。同步 OUT 传输期间出现的 CRC 错误或缓冲区上溢条件仍被视为正确事务，它们始终会触发 CTR_RX 事件。但是，CRC 错误仍会将 USB_ISTR 寄存器中的 ERR 位置 1 以向软件通知可能的数据损坏。

32.5.5 挂起/恢复事件

USB 标准定义了一个称为 SUSPEND 的特殊外设状态，在此状态下，USB 总线消耗的平均电流不得超过 2.5 mA。这是对总线供电设备的基本要求，而自供电设备无需遵循这一严苛的功耗限制。在挂起模式下，主机 PC 不在 USB 总线上发送任何通信数据并持续 3 ms 以上的时间，以此发送通知：因为在正常操作期间必须每隔 1 ms 发送一次 SOF 数据包，如果 USB 外设检测到 3 个连续的 SOF 数据包缺失，则会将这种情况视为主机 PC 的挂起请求，然后将 USB_ISTR 寄存器中的 SUSP 位设置为“1”，从而引起中断（如果使能）。设备挂起后，可通过所谓的 RESUME 序列恢复其正常操作，该序列可从主机 PC 或直接从外设本身启动，但始终通过主机 PC 终止。挂起的 USB 外设必须仍能够检测到 RESET 序列，将此事件视为正常的 USB 复位事件并对其做出响应。

挂起 USB 外设所使用的实际步骤取决于设备，因为对于不同的设备构成，降低总功耗所需的操作也会有所不同。

下面给出了典型挂起步骤的简单说明，重点关注与 USB 外设的 SUSP 通知对应的应用软件程序中与 USB 相关的方面：

1. 将 USB_CNTR 寄存器中的 FSUSP 位置 1。此操作将激活 USB 外设中的挂起模式。激活挂起模式后，将立即禁止“接收到 SOF 时检查”功能，以避免在 USB 挂起时发出任何其他 SUSP 中断。
2. 消除或降低 USB 外设以外的模块中的任何静态功耗。
3. 将 USB_CNTR 寄存器中的 LP_MODE 位置 1，以消除模拟 USB 收发器中的静态功耗，但使这些收发器能够检测恢复活动。
4. 可选择关闭外部振荡器或器件 PLL 以停止设备内的任何活动。

如果在设备处于 SUSPEND 模式时发生 USB 事件，则必须调用 RESUME 程序来恢复标称时钟和正常的 USB 行为。当唤醒事件是 USB 复位序列时，必须特别注意确保此过程不超过 10 ms（更多详细信息，请参见“通用串行总线规范”）。在 USB 外设挂起时启动恢复或复位序列会将 USB_CNTR 寄存器中的 LP_MODE 位异步清零。即使此事件能够触发 WKUP 中断（如果使能），也必须仔细评估是否使用中断响应程序，因为系统时钟重启会引入较长的延时；为缩短重新激活标称时钟前的延时，建议紧接着挂起程序末尾放置恢复程序，以便在系统时钟重启时，立即执行相应代码。为防止 ESD 放电或任何其他类型的噪声唤醒系统（退出挂起模式是一个异步事件），挂起期间需在数据线上激活合适的模拟滤波器；滤波器的宽度约为 70 ns。

以下是恢复程序应解决的一系列操作：

1. 可选择接通外部振荡器和/或器件 PLL。
2. 将 USB_CNTR 寄存器中的 FSUSP 位清零。
3. 如果必须识别恢复触发事件，则可根据*表 147* 使用 USB_FNR 寄存器中的位 RXDP 和 RXDM，表中还列出了所有情况下的目标软件操作。如果需要，可通过监视上述位的状态（即检查它们何时达到“10”配置，该配置表示空闲总线状态）来检测恢复或复位序列是否结束；此外，当复位序列结束时，USB_ISTR 寄存器中的 RESET 位将置 1，从而发出中断（如果使能），该中断应照常处理。

表 147. 恢复事件检测

[RXDP,RXDM] 状态	唤醒事件	所需的软件恢复操作
“00”	根复位	无
“10”	无（总线上的噪声）	返回挂起模式
“01”	根恢复	无
“11”	不允许（总线上的噪声）	返回挂起模式

设备可能需要退出挂起模式来响应与 USB 协议不是直接相关的特定事件（例如，通过移动鼠标唤醒整个系统）。在这种情况下，可通过如下方式启动恢复序列：将 USB_CNTR 寄存器中的 RESUME 位置 1，然后在一段介于 1 ms 和 15 ms 之间的时间间隔后将其复位为 0（此时间间隔可使用 ESOF 中断来定时，当系统时钟以标称频率运行时，周期为 1 ms）。RESUME 位清零后，恢复序列将由主机 PC 完成，其结束可再次使用 USB_FNR 寄存器中的 RXDP 和 RXDM 位来监视。

注： *RESUME 位仍必须仅在 USB 外设置于挂起模式后（即 USB_CNTR 寄存器中的 FSUSP 位置 1）使用。*

32.6 USB 寄存器

USB 外设寄存器可分为以下几组：

- 通用寄存器：中断和控制寄存器
- 端点寄存器：端点配置和状态
- 缓冲区描述符表：数据包存储器中用于定位数据缓冲区的存储单元

所有寄存器地址均以相对于 USB 外设寄存器基址 0x4000 5C00 的偏移量表示，但缓冲区描述符表的存储单元除外，其起始地址由 USB_BTABLE 寄存器指定。

有关寄存器说明中使用的缩写，请参见*第 50 页的第 1.1 节*。

外设寄存器可按半字（16 位）或字（32 位）访问。

32.6.1 通用寄存器

这类寄存器会影响 USB 外设的一般行为，包括定义工作模式、中断处理、设备地址以及访问主机 PC 所更新的当前帧数。

USB 控制寄存器 (USB_CNTR)

USB control register

偏移地址：0x40

复位值：0x0003

位 15 **CTRM**：正确传输中断屏蔽 (Correct transfer interrupt mask)
0：禁止正确传输 (CTR) 中断。
1：使能 CTR 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 14 **PMAOVRM**：数据包存储区上溢/下溢中断屏蔽 (Packet memory area over / underrun interrupt mask)
0：禁止 PMAOVR 中断。
1：使能 PMAOVR 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 13 **ERRM**：错误中断屏蔽 (Error interrupt mask)
0：禁止 ERR 中断。
1：使能 ERR 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 12 **WKUPM**：唤醒中断屏蔽 (Wakeup interrupt mask)
0：禁止 WKUP 中断。
1：使能 WKUP 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 11 **SUSPM**：挂起模式中断屏蔽 (Suspend mode interrupt mask)

0：禁止挂起模式请求 (SUSP) 中断。

1：使能 SUSP 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 10 **RESETM**：USB 复位中断屏蔽 (USB reset interrupt mask)

0：禁止 RESET 中断。

1：使能 RESET 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 9 **SOFM**：帧起始中断屏蔽 (Start of frame interrupt mask)

0：禁止 SOF 中断。

1：使能 SOF 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 8 **ESOFM**：预期帧起始中断屏蔽 (Expected start of frame interrupt mask)

0：禁止预期帧起始 (ESOF) 中断。

1：使能 ESOF 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

位 7 **L1REQM**：LPM L1 状态请求中断屏蔽 (LPM L1 state request interrupt mask)

0：禁止 LPM L1 状态请求 (L1REQ) 中断。

1：使能 L1REQ 中断，当 USB_ISTR 寄存器中的相应位置 1 时，将生成中断请求。

注：

位 6 保留

位 5 **L1RESUME**：LPM L1 恢复请求 (LPM L1 Resume request)

微控制器可将此位置 1 以向主机发送 LPM L1 恢复信号。信号发送结束后，此位通过硬件清零。

注：

位 4 **RESUME**：恢复请求 (Resume request)

微控制器可将此位置 1 以向主机发送恢复信号。它必须根据 USB 规范激活并持续 1 ms 到 15 ms 的时间，经过这段时间后，主机 PC 便可驱动恢复序列直到结束。

位 3 **FSUSP**：强制挂起 (Force suspend)

软件必须在接收到 SUSP 中断时将此位置 1，当持续 3 ms 未从 USB 外设接收到任何通信数据时，将发出 SUSP 中断。

0：不起作用。

1：进入挂起模式。模拟收发器中的时钟和静态功耗不受影响。如果挂起功耗是一项要求（总线供电设备），则应用软件应将在 FSUSP 后将 LP_MODE 位置 1，下面给出了具体说明。

位 2 **LP_MODE**：低功耗模式 (Low-power mode)

当挂起模式功率限制要求避免所有静态功耗时（为外部上拉电阻供电所需的静态功耗除外），使用此模式。当应用软件准备停止所有系统时钟或降低其频率以满足 USB 挂起状态的功耗要求时，应进入此状态。挂起模式期间的 USB 活动（WKUP 事件）会将此位异步复位（此位也可通过软件复位）。

0：无低功耗模式。

1：进入低功耗模式。

位 1 **PDWN**：掉电 (Power down)

当出于某种原因而需要完全禁止 USB 外设时，此位用于完全关闭所有与 USB 相关的模拟部分。此位置 1 时，USB 外设将与收发器断开，并且无法使用。

0：退出掉电模式。

1：进入掉电模式。

位 0 **FRES**：强制 USB 复位 (Force USB Reset)

0：清除 USB 复位。

1：强制 USB 外设复位，其方式与在 USB 上发出 RESET 信号相同。USB 外设保持 RESET 状态，直到软件将此位清零。使能时，将生成“USB-RESET”中断。

USB 中断状态寄存器 (USB_ISTR)

USB interrupt status register

偏移地址：0x44

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

此寄存器包含所有中断源的状态，允许应用软件确定哪些事件引起了中断请求。

此寄存器的高半部分包含多个单独位，每个位表示一个特定事件。这些位会在发生相关事件时由硬件置 1；如果 USB_CNTR 寄存器中的相应位置 1，将生成通用中断请求。中断程序用于检查每个位，它将执行所有必要的操作并在最终将已处理的位清零。如果其中的任一位未清零，则中断仍被视为处于挂起状态，中断线将再次保持高电平。如果多个位同时置 1，则将仅生成一个中断。

端点事务完成可通过一种不同的方式进行处理，以缩短中断响应延时。端点成功完成一个事务后，硬件会立即将 CTR 位置 1，从而在 USB_CNTR 中的相应位置 1 时生成通用中断请求。端点专用中断状态的激活与 USB_CNTR 寄存器中的 CTRM 位无关。中断状态均保持激活，直到软件将相应 USB_EPnR 寄存器中的挂起位清零为止（CTR 位实际上是一个只读位）。对于与端点相关的中断，软件可使用事务方向 (DIR) 和 EP_ID 只读位来确定哪个端点发出最后一个中断请求以及调用相应的中断服务程序。

用户可通过指定软件在中断服务程序中检查 USB_ISTR 位的顺序来选择同时挂起的多个 USB_ISTR 事件的相对优先级。仅清零与事件相关且已被处理的位。在服务程序结束时，将请求另一个中断来处理剩余中断状态。

为避免意外清零某些位，建议通过加载指令清零相应位，即向所有不得改变的位写入 1，向所有待清零的位写入 0（这些位只能通过软件清零）。由于在读操作和写操作之间，另一个位会被硬件置 1，并将在微处理器有时间处理事件之前由下一次写操作清零，因此应避免读-修改-写周期。

下面详细介绍每个位：

位 15 **CTR**：正确传输 (Correct transfer)

此位由硬件置 1，用于指示端点已成功完成事务；软件可通过 DIR 和 EP_ID 位确定哪个端点请求中断。此位为只读位。

位 14 **PMAOVR**：数据包存储区上溢/下溢 (Packet memory area over / underrun)

当微控制器无法及时响应 USB 存储器请求时，此位将置 1。USB 外设通过以下方式处理此事件：接收期间不发送 ACK 握手包，发送期间在发送的流中强制位填充错误；在这两种情况下，主机都将重试事务。正常操作期间，绝对不应出现 PMAOVR 中断。由于主机重试失败的事务，应用软件将有机会在此中断处理期间加快设备操作，以为下一次事务重试做准备；但这在同步传输期间不会发生（无论如何也不会重试任何同步事务），因此会导致数据丢失。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 13 **ERR**：错误 (Error)

只要出现下列其中一个错误，此标志便会置 1。
NANS：无应答。主机响应已超时。
CRC：循环冗余校验错误。接收的 CRC（令牌或数据中）之一有错误。
BST：位填充错误。在 PID、数据和/或 CRC 中的任一位置上检测到了位填充错误。
FVIO：帧格式违例。接收到了非标准帧（EOP 未处于正确位置、令牌序列错误等）。
由于 USB 外设和 PC 主机在出现错误时采用全透明方式管理重新发送，因此 USB 软件通常可忽略错误。此中断在软件开发阶段会很有用，可用于监视 USB 总线上的传输质量以及向用户指示可能的问题（例如，连接器松动、环境过于嘈杂、USB 电缆中的导线断裂等）。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 12 **WKUP**：唤醒 (Wakeup)

当在挂起模式期间检测到用于唤醒 USB 外设的活动时，此位将由硬件置 1。此事件将异步清零 CTLR 寄存器中的 LP_MODE 位并激活 USB_WAKEUP 线，这可用于通知设备的其余部分（例如，唤醒单元）恢复过程即将开始。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 11 **SUSP**：挂起模式请求 (Suspend mode request)

当持续 3 ms 未接收到通信数据时，此位将由硬件置 1，用于指示来自 USB 总线的挂起模式请求。挂起状态检查将在任何 USB 复位后立即使能，并在挂起模式激活时 (FSUSP=1) 由硬件禁止，直到恢复序列结束。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 10 **RESET**：USB 复位请求 (USB reset request)

当 USB 外设在其输入上检测到有效的 USB RESET 信号时置 1。USB 外设响应 RESET 时，仅会复位其内部协议状态机，从而在 USB_CNTR 寄存器中的 RESETM 使能位置 1 时生成中断。在 RESET 位清零之前，将始终禁止发送和接收。所有配置寄存器均不复位：微控制器必须明确清零这些寄存器（这是为了确保能够安全提供 RESET 中断，以及能够完成 RESET 后紧跟的任何事务）。功能地址和端点寄存器由 USB 复位事件复位。
此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 9 **SOF**：帧起始 (Start of frame)

此位用于指示新的 USB 帧开始，在 SOF 数据包通过 USB 总线到达时置 1。中断服务程序可监视 SOF 事件，以获得 USB 主机的 1 ms 同步事件以及安全读取在接收到 SOF 数据包时更新的 USB_FNR 寄存器（这对于同步应用软件会十分有用）。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 8 **ESOF**：预期帧起始 (Expected start of frame)

当未接收到预期的 SOF 数据包时，此位由硬件置 1。主机每隔 1 ms 发送一个 SOF 数据包，但如果集线器未按预期接收到数据包，则挂起定时器将发出此中断。如果连续生成三个 ESOF 中断（即，三个 SOF 数据包丢失）且两两中断之间没有任何通信数据，则将生成 SUSP 中断。即使在挂起定时器尚未锁定时丢失 SOF 数据包，此位也会置 1。此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 7 **L1REQ**：LPM L1 状态使能 (LPM L1 state request)

当成功接收并应答进入 L1 状态的 LPM 命令时，此位由硬件置 1。此位为读/写位，但只能写入“0”，写入“1”不起作用。

注：

位 6:5 保留

位 4 **DIR**：事务方向 (Direction of transaction)

此位将由硬件根据已生成中断请求的成功事务的方向写入。

如果 DIR 位为 0，则与中断端点相关的 USB_EPnR 寄存器中的 CTR_TX 位置 1。中断事务的类型为 IN（USB 外设将数据发送给主机 PC）。

如果 DIR 位为 1，则与中断端点相关的 USB_EPnR 寄存器中的 CTR_RX 位置 1，或者 CTR_TX 和 CTR_RX 位均置 1。中断事务的类型为 OUT（USB 外设从主机 PC 接收数据），或者两个挂起事务均等待处理。

此信息可供应用软件用于访问与触发事务相关的 USB_EPnR 寄存器的位，因为它表示中断挂起的方向。此位为只读位。

位 3:0 **EP_ID[3:0]**：端点标识符 (Endpoint Identifier)

这些位将由硬件根据已生成中断请求的端点编号写入。如果有多个端点事务挂起，则硬件将写入与优先级最高的端点相关的端点标识符，优先级的定义方式如下：定义两个端点组，同步端点和双缓冲批量端点的优先级最高，其他端点次之。如果来自同一组的多个端点请求一个中断，则 USB_ISTR 寄存器中的 EP_ID 位将按照请求端点寄存器的编号升序分配，即 EP0R 的优先级最高，EP1R 次之，依此类推。应用软件可按照此优先级方案为每个端点分配一个寄存器，以便对同时发生的多个端点请求进行适当排序。这些位为只读。

USB 帧数寄存器 (USB_FNR)

USB frame number register

偏移地址：0x48

复位值：0x0XXX，其中 X 未定义

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 15 **RXDP**：接收数据 + 线状态 (Receive data + line status)

此位可用于观察所接收的数据正上行端口数据线的状态。可以在挂起结束程序期间使用此位来帮助确定唤醒事件。

位 14 **RXDM**：接收数据 - 线状态 (Receive data - line status)

此位可用于观察所接收的数据负上行端口数据线的状态。可以在挂起结束程序期间使用此位来帮助确定唤醒事件。

位 13 **LCK**：锁定 (Locked)

当在 USB 复位状态结束后或 USB 恢复序列结束后接收到至少两个连续的 SOF 数据包时，此位将由硬件置 1。锁定后，帧定时器将保持此状态，直到发生 USB 复位或 USB 挂起事件。

位 12:11 **LSOF[1:0]**：丢失 SOF (Lost SOF)

这些位在生成 ESOF 中断时由硬件写入，用于对丢失的连续 SOF 数据包进行计数。接收到 SOF 数据包时，这些位将被清零。

位 10:0 **FN[10:0]**：帧数 (Frame number)

此位域包含最后接收的 SOF 数据包中的 11 位帧数。主机每发送一个帧，帧数便会递增，帧数对于同步传输十分有用。此位域在生成 SOF 中断时更新。

USB 设备地址 (USB_DADDR)

USB device address

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

位 15:8 保留

位 7 **EF**：使能功能 (Enable function)

此位由软件置 1，用于使能 USB 设备。此设备的地址包含在下面的 ADD[6:0] 位中。如果此位为 0，则无论 USB_EPnR 寄存器的设置为何，都不会处理任何事务。

位 6:0 **ADD[6:0]**：设备地址 (Device Address)

这些位包含枚举过程中由主机 PC 分配的 USB 功能地址。此字段和相关 USB_EPnR 寄存器中的端点地址 (EA) 字段均必须与 USB 令牌中包含的信息匹配，才能处理所需端点的事务。

缓冲区表地址 (USB_BTABLE)

Buffer table address

偏移地址：0x50

复位值：0x0000

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

位 15:3 **BTABLE[15:3]**：缓冲区表 (Buffer table)

这些位包含专用数据包存储器中缓冲区分配表的起始地址。此表描述了各个端点缓冲区的位置和大小，它必须与 8 字节边界对齐（低 3 位始终为 0）。在寻址到此设备的每个事务开始时，USB 外设将读取此表中与所寻址端点相关的元素，以获取端点的缓冲区起始位置和缓冲区大小（请参见*第 848 页的“数据包缓冲区的结构和用法”*）。

位 2:0 保留，由硬件强制清零。

LPM 控制和状态寄存器 (USB_LPMCSR)

LPM control and status register

偏移地址：0x54

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r				r		rw	rw

位 15:8 保留

位 7:4 **BESL[3:0]**：BESL 值 (BESL value)
这些位包含通过最后应答的 LPM 令牌接收的 BESL 值

位 3 **REMWAKE**：bRemoteWake 值 (bRemoteWake value)
此位包含通过最后应答的 LPM 令牌接收的 bRemoteWake 值

位 2 保留

位 1 **LPMACK**：LPM 令牌应答使能 (LPM Token acknowledge enable)
0：有效 LPM 令牌将为 NYET。
1：有效 LPM 令牌将为 ACK。
只有成功完成 LPM 事务后才会返回 NYET/ACK：
EXT 令牌和 LPM 令牌中均无错误（否则返回 ERROR）
接收到有效 bLinkState = 0001B (L1)（否则返回 STALL）

位 0 **LPMEN**：LPM 支持使能 (LPM support enable)
该位由软件置 1，用于在 USB 设备内使能 LPM 支持。如果此位为 0，则不会处理任何 LPM 事务。

电池充电检测器 (USB_BCDR)

Battery charging detector

偏移地址：0x58

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2 DET	SDET	PDET	DC DET	SDEN	PDEN	DCD EN	BCD EN
rw								r	r	r	r	rw	rw	rw	rw

位 15 **DPPU**：DP 上拉控制 (DP pull-up control)

此位由软件置 1，用于使能 DP 线上的内置上拉。必要时，可通过用户软件将此位清零以向主机发出断开信号。

位 14:8 保留

位 7 **PS2DET**：DM 上拉检测状态 (DM pull-up detection status)

此位只在 PD 期间有效，用于指示 DM 电压电平与 V_{LGC} 阈值之间的比较结果。正常情况下，DM 电平应低于此阈值。如果 DM 电平高于此阈值，则意味着 DM 已从外部拉为高电平。原因可能是连接到 PS2 端口（同时上拉 DP 和 DM 线）或某些不遵循 BCD 规范的专有充电器。

0：检测到正常端口（连接到 SDP、ACA、CDP 或 DCP）。

1：检测到 PS2 端口或专有充电器。

位 6 **SDEN**：二次检测 (SD) 状态 (Secondary detection (SD) status)

此位指示 SD 的结果。

0：检测到 CDP。

1：检测到 DCP。

位 5 **PDEN**：初次检测 (PD) 状态 (Primary detection (PD) status)

此位指示 PD 的结果。

0：未检测到 BCD 支持（连接到 SDP 或专有设备）。

1：检测到 BCD 支持（连接到 ACA、CDP 或 DCP）。

位 4 **DCDEN**：数据接触检测 (DCD) 状态 (Data contact detection (DCD) status)

此位指示 DCD 的结果。

0：未检测到数据线接触。

1：检测到数据线接触。

位 3 **SDEN**：二次检测 (SD) 模式使能 (Secondary detection (SD) mode enable)

该位由软件置 1，用于将 BCD 置于 SD 模式。要正确工作，应只选择一个检测模式（DCD、PD、SD 或 OFF）。

位 2 **PDEN**：初次检测 (PD) 模式使能 (Primary detection (PD) mode enable)

该位由软件置 1，用于将 BCD 置于 PD 模式。要正确工作，应只选择一个检测模式（DCD、PD、SD 或 OFF）。

位 1 **DCDEN**：数据接触检测 (DCD) 模式使能 (Data contact detection (DCD) mode enable)

该位由软件置 1，用于将 BCD 置于 DCD 模式。要正确工作，应只选择一个检测模式（DCD、PD、SD 或 OFF）。

位 0 **BCDEN**：电池充电检测器 (BCD) 使能 (Battery charging detector (BCD) enable)

该位由软件置 1，用于在 USB 设备内使能 BCD 支持。使能后，USB PHY 完全由 BCD 控制，无法用于正常通信。BCD 检测完成后，应通过将该位清零使 BCD 置于 OFF 模式，以实现正常的 USB 操作。

端点相关寄存器

这类寄存器的数量因 USB 外设设计要处理的端点数而异。USB 外设最多支持 8 个双向端点。每个 USB 设备必须支持一个控制端点，端点地址（EA 位）必须设置为 0。当使能多个端点编号值相同的端点时，USB 外设将以一种未定义的方式工作。对于每个端点而言，USB_EPnR 寄存器可用于存储端点特定的信息。

USB 端点 n 寄存器 (USB_EPnR)，n=[0..7]

USB endpoint n register

偏移地址：0x00 到 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[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

当从 USB 总线接收到 USB 复位或通过 CTLR 寄存器中的 FRES 位强制 USB 复位时，这些位也会复位；但 CTR_RX 和 CTR_TX 位除外，它们将保持不变，以避免丢失紧跟在 USB 复位事件后的数据包通知。每个端点都有自己的 USB_EPnR 寄存器，其中 *n* 是端点标识符。

由于在读操作和写操作之间，一些位会被硬件置 1，并将在 CPU 有时间检测变化之前由下一次写操作修改，因此应避免这类寄存器上的读-修改-写周期。为此，受该问题影响的所有位都有一个“不变量”值，无需修改时必须使用该值。建议通过加载指令来修改这类寄存器，即向只能通过硬件修改的所有位写入其“不变量”值。

位 15 **CTR_RX：**接收传输正确 (Correct Transfer for reception)

当此端点上成功完成 OUT/SETUP 事务时，此位将由硬件置 1；软件只能将此位清零。如果 USB_CNTR 寄存器中的 CTRM 位相应地置 1，则会生成通用中断条件和端点相关中断条件，后者始终有效。已发生事务的类型（OUT 或 SETUP）可通过下述的 SETUP 位来确定。

以 NAK 或 STALL 握手结束的事务不会将此位置 1，因为在出现协议错误或数据同步不匹配时，实际上并未传输任何数据。

此位为读/写位，但只能写入“0”，写入“1”不起作用。

位 14 **DTOG_RX：**数据翻转，用于接收传输 (Data Toggle, for reception transfers)

如果端点不是同步端点，则此位包含要接收的下一个数据包的数据翻转位的预期值（0=DATA0，1=DATA1）。当 ACK 握手发送给 USB 主机时，硬件会将此位翻转，接着将接收到具有匹配数据 PID 值的数据包；如果端点定义为控制端点，则硬件会在接收到寻址到此端点的 SETUP PID 时将此位清零。

如果端点正在使用双缓冲功能，则此位还用于支持数据包缓冲区交换（请参见*第 32.5.3 节：双缓冲端点*）。

如果端点是同步端点，则此位只用于支持数据包缓冲区交换，因为此类端点不使用数据翻转，仅传输 DATA0 数据包（请参见*第 32.5.4 节：同步传输*）。硬件在数据包接收结束后立即翻转此位，因为同步传输不使用握手。

此位还可通过软件同步以初始化其值（当端点不是控制端点时是强制的）或强制使用特定的数据翻转/数据包缓冲区。当应用软件写入“0”时，DTOG_RX 的值保持不变，而写入“1”时，此位的值将进行翻转。此位为读/写位，但只能通过写入 1 来翻转。

位 13:12 **STAT_RX [1:0]**：状态位，用于接收传输 (Status bits, for reception transfers)

这些位包含有关端点状态的信息，如*第 865 页的表 148：接收状态编码*中所列。这些位可通过软件翻转以初始化其值。当应用软件写入“0”时，此位的值保持不变，而写入“1”时，此位的值将进行翻转。当寻址到此端点的 OUT 或 SETUP（仅控制）事务已正确传输后 (CTR_RX=1)，硬件会将 STAT_RX 位设置为 NAK，因此软件有时间在应答新事务之前准备接收的数据。

双缓冲批量端点实现了一种特殊的事务流控制，可基于缓冲区可用性条件来控制状态（请参见*第 32.5.3 节：双缓冲端点*）。

如果端点定义为同步端点，则其状态只能是“VALID”或“DISABLED”，在这种情况下，硬件无法在事务成功完成后更改端点的状态。如果软件为同步端点将 STAT_RX 位设置为“STALL”或“NAK”，则 USB 外设的行为是不明确的。这些位为读/写位，但只能通过写入 1 来同步。

位 11 **SETUP**：建立事务已完成 (Setup transaction completed)

此位为只读位，将在最后完成的事务为 SETUP 时由硬件置 1。此位仅为控制端点更改其值。它必须在接收事务成功完成（CTR_RX 事件）时进行检查，以确定所发生事务的类型。为防止中断服务程序因接下来传入的令牌而更改 SETUP 位，此位将在 CTR_RX 位为 1 时保持冻结；其状态将在 CTR_RX 为 0 时更改。此位为只读位。

位 10:9 **EP_TYPE[1:0]**：端点类型 (Endpoint type)

这些位配置此端点的行为，如*第 865 页的表 149：端点类型编码*所述。端点 0 必须始终为控制端点，每个 USB 功能必须至少有一个地址为 0 的控制端点，但必要时也可以有其他控制端点。只有控制端点会处理 SETUP 事务，其他类型的端点会忽略 SETUP 事务。SETUP 事务不能通过 NAK 或 STALL 应答。如果控制端点定义为 NAK，则 USB 外设将不会应答，而会在接收到 SETUP 事务时在接收方向上模拟接收错误。如果控制端点在接收方向上定义为 STALL，则 SETUP 数据包仍将被接受，从而传输数据并发出 CTR 中断。OUT 事务的接收以常规方式处理，即使端点为控制端点也如此。

批量端点和中断端点的行为十分相似，唯一的区别在于使用 EP_KIND 配置位可实现的特殊功能。

*第 32.5.4 节：同步传输*中介绍了同步端点的用法。

位 8 **EP_KIND**：端点种类 (Endpoint kind)

此位的含义取决于 EP_TYPE 位所配置的端点类型。*表 150* 汇总了不同的含义。

DBL_BUF：此位通过软件置 1，用于为此批量端点使能双缓冲功能。*第 32.5.3 节：双缓冲端点*中介绍了双缓冲批量端点的用法。

STATUS_OUT：此位通过软件置 1，用于指示将出现状态输出事务：在这种情况下，所有包含非零字节数据的 OUT 事务均通过“STALL”而非“ACK”来应答。此位可用于提高应用软件针对控制传输期间的协议错误的鲁棒性（仅限控制端点）。当 STATUS_OUT 复位时，OUT 事务可以有任何数量的字节（根据需要）。

位 7 **CTR_TX**：发送传输正确 (Correct Transfer for transmission)

当此端点上成功完成 IN 事务时，此位将由硬件置 1；软件只能将此位清零。如果 USB_CNTR 寄存器中的 CTRM 位相应地置 1，则会生成通用中断条件和端点相关中断条件，后者始终有效。

以 NAK 或 STALL 握手结束的事务不会将此位置 1，因为在出现协议错误或数据翻转不匹配时，实际上并未传输任何数据。

此位为读/写位，但只能写入“0”。

位 6 **DTOG_TX**：数据翻转，用于发送传输 (Data Toggle, for transmission transfers)

如果端点不是同步端点，则此位包含要发送的下一个数据包的数据翻转位的预期值（0=DATA0，1=DATA1）。当从 USB 主机接收到 ACK 握手时，硬件会将此位翻转，接着将发送一个数据包。如果端点定义为控制端点，则硬件会在接收到寻址到此端点的 SETUP PID 时将此位置 1。

如果端点正在使用双缓冲功能，则此位还用于支持数据包缓冲区交换（请参见*第 32.5.3 节：双缓冲端点*）。

如果端点是同步端点，则此位用于支持数据包缓冲区交换，因为此类端点不使用数据同步，仅传输 DATA0 数据包（请参见*第 32.5.4 节：同步传输*）。硬件在数据包接收结束后立即翻转此位，因为同步传输不使用握手。

此位还可通过软件翻转以初始化其值（当端点不是控制端点时十分必要）或强制使用特定的数据翻转/数据包缓冲区。当应用软件写入“0”时，DTOG_TX 的值保持不变，而写入“1”时，此位的值将进行同步。此位为读/写位，但只能通过写入 1 来同步。

位 5:4 **STAT_TX [1:0]**：状态位，用于发送传输 (Status bits, for transmission transfers)

这些位包含有关端点状态的信息，如*表 151* 所列。这些位可通过软件翻转以初始化其值。当应用软件写入“0”时，此位的值保持不变，而写入“1”时，此位的值将进行翻转。当寻址到此端点的 IN 或 SETUP（仅控制）事务已正确传输后 (CTR_TX=1)，硬件会将 STAT_TX 位设置为 NAK。然后，它将等待软件准备下一组要发送的数据。

双缓冲批量端点实现了一种特殊的事务流控制，可基于缓冲区可用性来条件控制状态（请参见*第 32.5.3 节：双缓冲端点*）。

如果端点定义为同步端点，则其状态只能是“VALID”或“DISABLED”。因此，硬件无法在事务成功完成后更改端点的状态。如果软件为同步端点将 STAT_TX 位设置为“STALL”或“NAK”，则 USB 外设的行为是不确定的。这些位为读/写位，但只能通过写入 1 来同步。

位 3:0 **EA[3:0]**：端点地址 (Endpoint address)

软件必须在此字段中写入 4 位地址，用于确定定向到此端点的事务。必须在使能相应端点前写入一个值。

表 148. 接收状态编码

STAT_RX[1:0]	意义
00	**DISABLED**：忽略寻址到此端点的所有接收请求。
01	**STALL**：停止端点，所有接收请求导致 STALL 握手。
10	**NAK**：不应答端点，所有接收请求导致 NAK 握手。
11	**VALID**：将此端点使能用于接收。

表 149. 端点类型编码

EP_TYPE[1:0]	意义
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

表 150. 端点种类的意义

EP_TYPE[1:0]		EP_KIND 的意义
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	未使用
11	INTERRUPT	未使用

表 151. 发送状态编码

STAT_TX[1:0]	意义
00	**DISABLED**：忽略寻址到此端点的所有发送请求。
01	**STALL**：停止端点，所有发送请求导致 STALL 握手。
10	**NAK**：不应答端点，所有发送请求导致 NAK 握手。
11	**VALID**：将此端点使能用于发送。

32.6.2 缓冲区描述符表

尽管缓冲区描述符表位于数据包缓冲区存储器内，但其条目可视为额外的寄存器，用于配置在 USB 宏单元和设备之间交换数据所使用的数据包缓冲区的位置和大小。

第一个数据包存储单元位于 0x4000 6000。下文介绍了与 USB_EPnR 寄存器相关的缓冲区描述符表条目。数据包存储器只应按字节（8 位）或半字（16 位）访问。不允许按字（32 位）访问。

有关数据包缓冲区和缓冲区说明符表用法的详细说明，可参见*第 848 页的“数据包缓冲区的结构和用法”*。

发送缓冲区地址 n (USB_ADDRn_TX)

Transmission buffer address n

偏移地址：[USB_BTABLE] + n*8

注： *对于 IN 方向上的双缓冲端点或同步端点，此地址单元用于 USB_ADDRn_TX_0。*
对于 OUT 方向上的双缓冲端点或同步端点，此地址单元用于 USB_ADDRn_RX_0。

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	-

位 15:1 **ADDRn_TX[15:1]**：发送缓冲区地址 (Transmission buffer address)
这些位指向将包含如下内容的数据包缓冲区的起始地址：与 USB_EPnR 寄存器相关的端点将在寻址到它的下一个 IN 令牌中发送的数据。

位 0 必须始终写为 0，因为数据包存储器为半字宽，所有数据包缓冲区必须按半字对齐。

发送字节数 n (USB_COUNTn_TX)

Transmission byte count n

偏移地址：[USB_BTABLE] + n*8 + 2

注： *对于 IN 方向上的双缓冲端点或同步端点，此地址单元用于 USB_COUNTn_TX_0。*
对于 OUT 方向上的双缓冲端点或同步端点，此地址单元用于 USB_COUNTn_RX_0。

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

位 15:10 不使用这些位，因为 USB 规范将数据包大小限制为 1023 字节。USB 外设不会考虑这些位的值。

位 9:0 **COUNTn_TX[9:0]**：发送字节数 (Transmission byte count)
这些位包含与 USB_EPnR 寄存器相关的端点将在寻址到它的下一个 IN 令牌中发送的字节数。

接收缓冲区地址 n (USB_ADDRn_RX)

Reception buffer address n

偏移地址：[USB_BTABLE] + n*8 + 4

注： *对于 OUT 方向上的双缓冲端点或同步端点，此地址单元用于 USB_ADDRn_RX_1。*
对于 IN 方向上的双缓冲端点或同步端点，此地址单元用于 USB_ADDRn_TX_1。

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 15:1 **ADDRn_RX[15:1]**：接收缓冲区地址 (Reception buffer address)
这些位指向将包含如下内容的数据包缓冲区的起始地址：与 USB_EPnR 寄存器相关的端点在寻址到它的下一个 OUT/SETUP 令牌中接收的数据。

位 0 此位必须始终写为 0，因为数据包存储器为半字宽，所有数据包缓冲区必须按半字对齐。

接收字节数 n (USB_COUNTn_RX)

Reception byte count n

偏移地址：[USB_BTABLE] + n*8 + 6

注： *对于 OUT 方向上的双缓冲端点或同步端点，此地址单元用于 USB_COUNTn_RX_1。*
对于 IN 方向上的双缓冲端点或同步端点，此地址单元用于 USB_COUNTn_TX_1。

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

此表存储单元用于存储数据包接收期间需要的两个不同的值。最高有效位包含已分配缓冲区大小的定义，以便进行缓冲区溢出检测；此存储单元的最低有效位由 USB 外设在接收结束时回写，以给出已接收字节的实际数量。由于可用位数受到限制，缓冲区大小使用已分配存储块的数量来表示，其中可通过选择块大小在精细粒度/小缓冲区与粗略粒度/大缓冲区之间进行折衷。已分配缓冲区的大小是端点描述符的一部分，通常在枚举过程中根据其 maxPacketSize 参数值定义（请参见“通用串行总线规范”）。

位 15 **BL_SIZE：**块大小 (Block size)

此位选择用于定义已分配缓冲区的存储器块的大小。

– 如果 BL_SIZE = 0，则存储器块为 2 字节，这是半字宽存储器中允许的最小块。使用此块大小时，所分配缓冲区的大小在 2 字节到 62 字节范围内。

– 如果 BL_SIZE = 1，则存储器块为 32 字节，可达到 USB 规范定义的最大数据包长度。使用此块大小时，理论上所分配缓冲区的大小在 32 字节到 1024 字节范围内，这是 USB 标准规范允许的最大数据包长度。不过，可用的缓冲区大小受可用缓冲区存储器的限制。

位 14:10 **NUM_BLOCK[4:0]：**块数 (Number of blocks)

这些位定义分配到此数据包缓冲区的存储器块的数量。已分配存储器的实际数量取决于*表 152* 中给出的 BL_SIZE 值。

位 9:0 **COUNTn_RX[9:0]：**接收字节数 (Reception byte count)

这些位包含与 USB_EPnR 寄存器相关的端点在寻址到它的最后一个 OUT/SETUP 事务期间接收的字节数。

表 152. 已分配缓冲区存储器的定义

NUM_BLOCK[4:0] 的值	已分配的存储器（当 BL_SIZE=0 时）	已分配的存储器（当 BL_SIZE=1 时）
0 (‘00000)	不允许	32 字节
1 (‘00001)	2 字节	64 字节
2 (‘00010)	4 字节	96 字节
3 (‘00011)	6 字节	128 字节
...	...	...
14 (‘01110)	28 字节	480 字节
15 (‘01111)	30 字节	
16 (‘10000)	32 字节	
...	...	...
29 (‘11101)	58 字节	
30 (‘11110)	60 字节	
31 (‘11111)	62 字节	N/A

32.6.3 USB 寄存器映射

下表提供了 USB 寄存器映射和复位值。

表 153. USB 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_EP0R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USB_EP1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**USB_EP2R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USB_EP3R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USB_EP4R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USB_EP5R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USB_EP6R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**USB_EP7R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	**USB_CNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	L1REQM	Res.	L1RESUME	RESUME	FSUSP	LPMODE	PDWN	FRES
	Reset value																	0	0	0	0	0	0	0	0				0	0	0	1	1
0x44	**USB_ISTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	L1REQ	Res.	Res.	DIR	EP_ID[3:0]			
	Reset value																	0	0	0	0	0	0	0	0				0	0	0	0	0
0x48	**USB_FNR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	**USB_DADDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0

表 153. USB 寄存器映射和复位值（续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	**USB_BTABLE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BTABLE[15:3]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x54	**USB_LPMCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	**USB_BCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

有关寄存器边界地址的信息，请参见*第 57 页的第 2.2.2 节*。

33 调试支持 (DBG)

33.1 概述

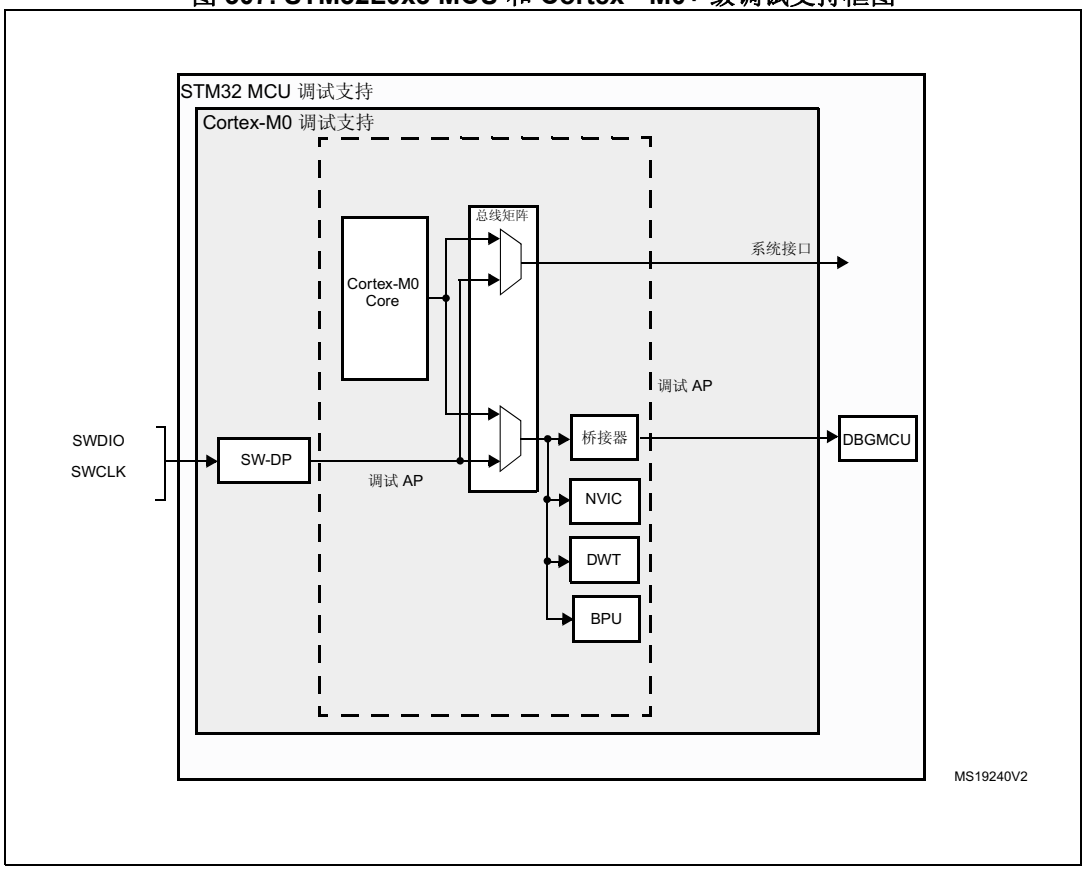
STM32L0x3 器件的内核是 Cortex®-M0+，该内核包含用于高级调试功能的硬件扩展。调试扩展允许内核可以在取指（指令断点）或取访问数据（数据断点）时停止内核。内核停止时，可以查询内核的内部状态和系统的外部状态。查询完成后，将恢复内核和系统并恢复程序执行。

当调试主机与 STM32L0x3 MCU 相连并进行调试时，将使用调试功能。

提供一个调试接口：

- 串行接口

图 307. STM32L0x3 MCU 和 Cortex®-M0+ 级调试支持框图



1. Cortex®-M0+ 内核中内置的调试功能是 ARM® CoreSight 设计套件的一部分。

ARM® Cortex®-M0+ 内核提供集成片上调试支持。它包括：

- SW-DP：串行线
- BPU：断点单元
- DWT：数据观察点触发

它还包括专用于 STM32L0x3 的调试功能：

- 灵活调试引脚分配
- MCU 调试盒（支持低功耗模式和对外设时钟的控制等）

注：有关 ARM® Cortex®-M0+ 内核支持的调试功能的详细信息，请参见 Cortex®-M0+ 技术参考手册（请参见第 33.2 节：ARM® 参考文档）。

33.2 ARM® 参考文档

- Cortex®-M0+技术参考手册 (TRM)
  可从 www.infocenter.arm.com 获取
- ARM® 调试接口 V5
- ARM® CoreSight 设计套件版本 r1p1 技术参考手册

33.3 引脚排列和调试端口引脚

STM32L0x3 MCU 的不同封装有不同的有效引脚数。

33.3.1 SWD 端口引脚

两个引脚被用作 SW-DP 的输出，作为通用 I/O 的复用功能。所有封装都提供这些引脚。

表 154. SW 调试端口引脚

SW-DP 引脚名称	SW 调试端口		引脚分配
	类型	调试分配	
SWDIO	IO	串行线数据输入/输出	PA13
SWCLK	I	串行线时钟	PA14

33.3.2 SW-DP 引脚分配

复位（SYSRESETn 或 PORESETn）后，用于 SW-DP 的引脚将分配为可由调试主机立即使用的专用引脚。

但是，MCU 可以禁止 SWD 端口，进而可释放相关引脚以用作通用 I/O (GPIO)。有关如何禁止 SW-DP 端口引脚的更多详细信息，请参见第 224 页上的第 9.3.2 节：I/O 引脚复用功能复用器和映射。

33.3.3 SWD 引脚上的内部上拉和下拉

用户软件释放 SW I/O 后，GPIO 控制器便会控制这些引脚。GPIO 控制寄存器的复位状态会将 I/O 置于等效的状态：

- SWDIO：输入上拉
- SWCLK：输入下拉

由于内置上拉和下拉电阻，因此无需添加外部电阻。

33.4 ID 代码和锁定机制

MCU 内部有多个 ID 代码。ST 强烈建议工具制造商使用地址 0x40015800 处的 MCU 器件 ID 锁定其调试工具。

33.4.1 MCU 器件 ID 代码

STM32L0x3 产品集成了 MCU ID 代码。此 ID 标识 ST MCU 的料号和晶片版本。

可通过软件调试端口（两个引脚）或用户软件访问此代码。

调试软件/编程工具只能将 DEV_ID[15:0] 用于识别芯片（不得考虑版本 ID）。

有关代码示例，请参见 *A.20.1：DBG 读取器件 Id 代码示例*。

DBG_IDCODE

地址：0x4001 5800

仅支持 32 位访问。只读

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

位 31:16 **REV_ID[15:0]**：版本标识符 (Revision identifier)
此字段指示器件的版本（请参见*表 155：REV-ID 值*）。

位 15:12 保留：读取 0b0110。

位 11:0 **DEV_ID[11:0]**：器件标识符 (Device identifier)
此字段指示器件 ID：
第 3 类器件：0x417
第 5 类器件：0x447

表 155. REV-ID 值

REV_ID	第 3 类器件	第 5 类器件
0x1000	版本 A	
0x1008	版本 Z	-
0x1018	版本 Y	-
0x1038	版本 X	-
0x2000	-	版本 B
0x2008	-	版本 Z

33.5 SWD 端口

33.5.1 SWD 协议简介

此同步串行协议使用两个引脚：

- SWCLK：从主机到目标的时钟
- SWDIO：双向

利用该协议，可以同时读取和写入两组寄存器组（DPACC 寄存器组和 APACC 寄存器组）。

传输数据时，LSB 在前。

对于 SWDIO 双向管理，必须在电路板上对线路进行上拉（ARM® 建议采用 100 kΩ）。这些上拉电阻可在内部配置。无需外部上拉电阻。

每次在协议中更改 SWDIO 的方向时，都会插入转换时间，此时线路即不受主机驱动也不受目标驱动。默认情况下，此转换时间为一位时间，但可以通过配置 SWCLK 频率来调整。

33.5.2 SWD 协议序列

每个序列包括三个阶段：

1. 主机发送的数据包请求（8 位）
2. 目标发送的确认响应（3 位）
3. 主机或目标发送的数据传输阶段（33 位）

表 156. 数据包请求（8 位）

位	名称	说明
0	启动	必须为 1
1	APnDP	0：DP 访问 1：AP 访问
2	RnW	0：写请求 1：读请求
4:3	A[3:2]	DP 或 AP 寄存器的地址字段（请参见*第 877 页的表 160*）
5	奇偶校验	前面几位的单位奇偶校验
6	停止	0
7	驻留	不受主机驱动。由于存在上拉，因此必须由目标读为 1

有关 DPACC 和 APACC 寄存器的详细说明，请参见 Cortex®-M0+ *TRM*。

数据包请求后面始终为转换时间（默认 1 位），此时主机和目标都不会驱动线路。

表 157. ACK 响应（3 位）

位	名称	说明
0..2	ACK	001：FAULT 010：WAIT 100：OK

仅当发生 READ 事务或者接收到 WAIT 或 FAULT 确认时，ACK 响应后才必须是转换时间。

表 158. DATA 传输（33 位）

位	名称	说明
0..31	WDATA 或 RDATA	写入或读取数据
32	奇偶校验	32 个数据位的单奇偶校验

仅当发生 READ 事务时，DATA 传输后才必须是转换时间。

33.5.3 SW-DP 状态机（复位、空闲状态、ID 代码）

SW-DP 的状态机有一个用于标识 SW-DP 的内部 ID 代码。该代码符合 JEP-106 标准。此 ID 代码是默认的 ARM® 代码，设置为 **0x0BB11477**（相当于 Cortex®-M0+）。

注： *请注意，在目标读取此 ID 代码前，SW-DP 状态机是不工作的。*

- 在上电复位后或者线路处于高电平超过 50 个周期后，SW-DP 状态机处于复位状态。
- 如果在复位状态后线路处于低电平至少两个周期，SW-DP 状态机处于空闲状态。
- 复位状态后，该状态机必须首先进入空闲状态，然后对 DP-SW ID CODE 寄存器执行读访问。否则，目标将在另一个事务上发出 FAULT 确认响应。

有关 SW-DP 状态机的更多详细信息，请参见 *Cortex®-M0+ TRM* 和 *CoreSight 设计套件 r1p0 TRM*。

33.5.4 DP 和 AP 读/写访问

- 不延迟对 DP 的读访问：可以立即发送目标响应（如果 ACK=OK），也可以延迟发送目标响应（如果 ACK=WAIT）。
- 延迟对 AP 的读访问。这意味着会在下次传输时返回访问结果。如果要执行的下次访问不是 AP 访问，则必须读取 DP-RDBUFF 寄存器来获取结果。
  每次进行 AP 读访问或 RDBUFF 读请求时都会更新 DP-CTRL/STAT 寄存器的 READOK 标志，以便了解 AP 读访问是否成功。
- SW-DP 有写缓冲区（用于 DP 或 AP 写入），这样即使在其它操作仍未完成时，也可以接受写入操作。如果写缓冲区已满，则目标确认响应为 WAIT。但 IDCODE 读取、CTRL/STAT 读取或 ABORT 写入除外，这几项操作在写缓冲区已满时也会被接受。
- 由于存在异步时钟域 SWCLK 和 HCLK，因此写操作后（奇偶校验位后）还需要两个额外的 SWCLK 周期，以使写入操作在内部生效。应在将线路驱动为低电平时（空闲状态）应用这些周期。
  在写 CTRL/STAT 寄存器以提出一个上电请求时，这一点特别重要。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。

33.5.5 SW-DP 寄存器

当 APnDP=0 时能够访问这些寄存器

表 159. SW-DP 寄存器

A[3:2]	R/W	SELECT 寄存器的 CTRLSEL 位	寄存器	注释
00	读取		IDCODE	制造商代码设置为 Cortex®-M0+ 的默认 ARM® 代码。 **0x0BB11477**（标识 SW-DP）
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	目的： – 请求系统或调试上电 – 配置 AP 访问的传输操作 – 控制比较和验证操作 – 读取一些状态标志（上溢和上电确认）
01	读/写	1	WIRE CONTROL	用于配置物理串行端口协议（如转换时间的持续时间）
10	读取		READ RESEND	允许从已损坏的调试软件传输中恢复读取数据，无需重复执行原始 AP 传输。
10	写		SELECT	用于选择当前访问端口和活动的 4 字寄存器窗口
11	读/写		READ BUFFER	由于已发出 AP 访问，因此该读缓冲区非常有用（在执行下个 AP 事务时提供读取 AP 请求的结果）。 此读取缓冲区捕获 AP 中的数据，显示为前一次读取的结果，无需启动新操作

33.5.6 SW-AP 寄存器

当 APnDP=1 时能够访问这些寄存器

有多个 AP 寄存器，这些寄存器按以下组合进行寻址：

- 移位值 A[3:2]
- DP SELECT 寄存器的当前值

表 160. 32 位调试端口寄存器，通过移位值 A[3:2] 进行寻址

地址	A[3:2] 值	说明
0x0	00	保留，必须保持复位值。
0x4	01	DP CTRL/STAT 寄存器。用于： – 请求系统或调试上电 – 配置 AP 访问的传输操作 – 控制比较和验证操作 – 读取一些状态标志（上溢和上电确认）
0x8	10	DP SELECT 寄存器：用于选择当前访问端口和活动的 4 字寄存器窗口。 – 位 31:24：APSEL：选择当前 AP (select the current AP) – 位 23:8：保留 – 位 7:4：APBANKSEL：在当前 AP 上选择活动的 4 字寄存器窗口 – 位 3:0：保留
0xC	11	DP RDBUFF 寄存器：用于通过调试器在执行一系列操作后获取最后结果（无需请求新的 JTAG-DP 操作）

33.6 内核调试

通过内核调试寄存器调试内核。通过调试访问端口调试访问这些寄存器。它由四个寄存器组成：

表 161. 内核调试寄存器

寄存器	说明
DHCSR	*32 位调试停止控制和状态寄存器* 此寄存器提供有关处理器状态的信息，能够使内核进入调试停止状态并提供处理器步进功能。
DCRSR	*17 位调试内核寄存器选择器寄存器：* 此寄存器选择需要进行读写操作的处理器寄存器。
DCRDR	*32 位调试内核寄存器数据寄存器：* 此寄存器保存在寄存器与 DCRSR（选择器）寄存器选择的处理器之间读取和写入的数据。
DEMCR	*32 位调试异常和监视控制寄存器：* 此寄存器提供向量捕获和调试监视控制。

这些寄存器在系统复位时不复位。它们只能通过上电复位来复位。有关更多详细信息，请参见 Cortex®-M0+ TRM。

为了在复位后立即使内核进入调试停止状态，必须：

- 使能调试和异常监视控制寄存器的位 0 (VC_CORRESET)
- 使能调试停止控制和状态寄存器的位 0 (C_DEBUGEN)

33.7 BPU（断点单元）

Cortex®-M0+ BPU 实现提供四个断点寄存器。BPU 是 ARMv7-M（Cortex®-M3 和 Cortex®-M4）中提供的 Flash 补丁和断点 (FPB) 模块的一部分。

33.7.1 BPU 功能

处理器断点实现了基于 PC 的断点功能。

有关 BPU CoreSight 标识寄存器及其地址和访问类型的更多信息，请参见 ARMv6-M ARM® 和 ARM® CoreSight 组件技术参考手册。

33.8 DWT（数据观察点）

Cortex®-M0+ DWT 实现提供了两个观察点寄存器组。

33.8.1 DWT 功能

处理器观察点实现了数据地址和基于 PC 的观察点功能（即 PC 采样寄存器），并支持比较器地址掩码，如 *ARMv6-M ARM®* 中所述。

33.8.2 DWT 程序计数器采样寄存器

实现数据观察点单元的处理器还实现了 ARMv6-M 可选 *DWT 程序计数器采样寄存器* (DWT_PCSR)。此寄存器允许调试程序定期采样 PC，无需停止处理器。这可提供粗略分析。有关更多信息，请参见 *ARMv6-M ARM®*。

Cortex®-M0+ DWT_PCSR 记录通过条件代码和指令以及未通过条件代码的指令。

33.9 MCU 调试组件 (DBG)

MCU 调试组件帮助调试器为以下各项提供支持：

- 低功耗模式
- 断点期间的定时器、看门狗和 I2C 的时钟控制

33.9.1 对低功耗模式的调试支持

要进入低功耗模式，必须执行指令 WFI 或 WFE。

MCU 支持多个低功耗模式，这些模式可以禁止 CPU 时钟或降低 CPU 功耗。

内核不允许在调试会话期间关闭 FCLK 或 HCLK。由于调试期间需要使用它们进行调试连接，因此其必须保持激活状态。MCU 集成了特殊方法，允许用户在低功耗模式下调试软件。

为此，调试主机首先必须设置一些调试配置寄存器，以更改低功耗模式行为：

- 在休眠模式下，FCLK 和 HCLK 仍有效。因此，此模式对标准调试功能没有任何限制。
- 在停止/待机模式下，调试程序必须事先将 DBG_STOP 位置 1。

这样便可使能内部 RC 振荡器时钟，以在停止模式下为 FCLK 和 HCLK 提供时钟。

当 DBG_STANDBY、DBG_STOP 和 DBG_SLEEP 位中的其中一个置 1 且内部参考电压在低功耗模式下停止（PWR_CR 寄存器中的 ULP 位置 1）时，必须使能快速唤醒（PWR_CR 中的 FWU 位置 1）。

有关代码示例，请参见 *A.20.2：LPM 下的 DBG 调试代码示例*。

33.9.2 对定时器、看门狗和 I²C 的调试支持

断点期间，必须选择定时器和看门狗的计数器的行为方式：

- 在产生断点时，计数器继续计数。例如，当 PWM 控制电机时，通常需要这种方式。
- 在产生断点时，计数器停止计数。用于看门狗时需要这种方式。

对于 I²C，用户可以选择在断点期间阻止 SMBUS 超时。

33.9.3 调试 MCU 配置寄存器（DBG_CR）

当 MCU 处于调试状态下时，DBG_CR 寄存器可配置低功耗模式。当其中一个 DBG_CR 位置 1 时，如果 PWR_CR 中的 ULP 位置 1，则 PWR_CR 的 FWU 位必须置 1。

它映射到地址 0x4001 5804。

此寄存器通过 PORESET（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

如果调试主机不支持这些功能，仍然可以通过用户软件写入这些寄存器。

地址：0x04

仅支持 32 位访问

POR 复位：0x0000 0000（不会被系统复位信号复位）

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
													rw	rw	rw

位 31:3 保留，必须保持复位值。

位 2 **DBG_STANDBY**：调试待机模式 (Debug Standby mode)

0：（FCLK=关闭，HCLK=关闭）将整个数字部分断电。

从软件角度来看，退出待机模式相当于取复位向量（指示 MCU 从待机状态恢复的几个状态位除外）

1：（FCLK=开启，HCLK=开启）在这种情况下，数字部分不断电，FCLK 和 HCLK 由仍保持激活状态的内部 RC 振荡器提供。此外，MCU 会在待机模式期间产生系统复位，这样退出待机模式相当于取复位向量

位 1 **DBG_STOP**：调试停止模式 (Debug Stop mode)

0：（FCLK=关闭，HCLK=关闭）在停机模式下，时钟控制器禁止所有时钟（包括 HCLK 和 FCLK）。当退出停止模式时，时钟配置与 RESET 后的时钟配置相同。因此，软件必须重新编程时钟控制器以使能 PLL 和晶振等。

1：（FCLK=开启，HCLK=开启）在这种情况下，进入停机模式时，FCLK 和 HCLK 由在停机模式下仍保持激活状态的内部 RC 振荡器提供。退出停机模式时，软件必须重新编程时钟控制器以使能 PLL 和晶振等（操作步骤与在 DBG_STOP=0 时相同）

位 0 **DBG_SLEEP**：调试睡眠模式 (Debug Sleep mode)

0：在睡眠模式下，FCLK 由原先在 HCLK 禁止时通过软件配置好的系统时钟驱动。时钟控制器配置不复位，保持先前设置的状态。因此，退出睡眠模式时，软件不需要重新配置时钟控制器。

1：在这种情况下，进入睡眠模式时，将为 HCLK 和 FCLK 馈送相同的时钟（软件先前配置的系统时钟）。

33.9.4 调试 MCU APB1 冻结寄存器 (DBG_APB1_FZ)

当 MCU 处于调试状态下时，DBG_APB1_FZ 寄存器用于配置以下 APB 外设：

- 定时器时钟计数器冻结
- I2C SMBUS 超时冻结
- 系统窗口看门狗和独立看门狗计数器冻结支持。

此寄存器映射到地址 0x4001 5808。

寄存器通过 POR（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

偏移地址：0X08

仅支持 32 位访问。

上电复位 (POR)：0x0000 0000（不会被系统复位信号复位）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIMER_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
rw	rw									rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM6_STOP	Res.	Res.	Res.	DBG_TIM2_STOP
			rw	rw	rw						rw				rw

位 31 **DBG_LPTIMER_STOP**：内核停止时 LPTIM1 计数器停止 (TIM8 counter stopped when core is halted)

0：即使内核停止，仍然馈送 LPTIM1 计数器时钟
1：内核停止时，LPTIM1 计数器时钟停止

位 30 **DBG_I2C3_STOP**：内核停止时停止 I2C3 SMBUS 超时模式 (I2C3 SMBUS timeout mode stopped when core is halted)

0：行为方式与正常模式下相同
1：冻结 I2C3 SMBUS 超时

位 29:22 保留，必须保持复位值。

位 21 **DBG_I2C1_STOP**：内核停止时停止 I2C1 SMBUS 超时模式 (I2C1 SMBUS timeout mode stopped when core is halted)

0：行为方式与正常模式下相同
1：冻结 I2C1 SMBUS 超时

位 20:13 保留，必须保持复位值。

位 12 **DBG_IWDG_STOP**：内核停止时停止调试独立看门狗 (Debug independent watchdog stopped when core is halted)

0：即使内核停止，独立看门狗计数器时钟仍继续工作
1：内核停止时，独立看门狗计数器时钟停止

位 11 **DBG_WWDG_STOP**：内核停止时停止调试窗口看门狗 (Debug window watchdog stopped when core is halted)

0：即使内核停止，窗口看门狗计数器时钟仍继续工作
1：内核停止时，窗口看门狗计数器时钟停止

位 10 **DBG_RTC_STOP**：内核停止时停止调试 RTC (Debug RTC stopped when core is halted)

0：即使内核停止，仍然馈送 RTC 计数器时钟
1：内核停止时，RTC 计数器时钟停止

位 9:6 保留，必须保持复位值。

位 5 **DBG_TIM7_STOP**：内核停止时 TIM7 计数器停止 (TIM7 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM7 计数器时钟
1：内核停止时，TIM7 计数器时钟停止

位 4 **DBG_TIM6_STOP**：内核停止时 TIM6 计数器停止 (TIM6 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM6 计数器时钟
1：内核停止时，TIM6 计数器时钟停止

位 3:2 保留，必须保持复位值。

位 1 **DBG_TIM3_STOP**：内核停止时 TIM3 计数器停止 (TIM3 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM3 计数器时钟
1：内核停止时，TIM3 计数器时钟停止

位 0 **DBG_TIM2_STOP**：内核停止时 TIM2 计数器停止 (TIM2 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM2 计数器时钟
1：内核停止时，TIM2 计数器时钟停止

33.9.5 调试 MCU APB2 冻结寄存器 (DBG_APB2_FZ)

当 MCU 处于调试状态下时，DBG_APB2_FZ 寄存器用于配置 APB 外设的一些特性：

- 定时器时钟计数器冻结。

此寄存器映射到地址 0x4001580C。

它通过 POR（而非系统复位）异步复位。可在系统复位状态下用调试器写入该寄存器。

地址：0x0C

仅支持 32 位访问。

POR：0x0000 0000（不会被系统复位信号复位）

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM22_STOP	Res.	Res.	DBG_TIM21_STOP	Res.	Res.
										rw			rw		

位 31:6 保留，必须保持复位值。

位 5 **DBG_TIM22_STOP**：内核停止时 TIM22 计数器停止 (TIM22 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM22 计数器时钟

1：内核停止时，TIM22 计数器时钟停止

位 4:3 保留，必须保持复位值。

位 2 **DBG_TIM21_STOP**：内核停止时 TIM21 计数器停止 (TIM21 counter stopped when core is halted)

0：即使内核停止，仍然馈送 TIM21 计数器时钟

1：内核停止时，TIM21 计数器时钟停止

位 1:0 保留，必须保持复位值。

33.10 DBG 寄存器映射

下表对调试寄存器进行了汇总。

表 162. DBG 寄存器映射和复位值

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
0x40015800	DBG_ IDCODE	REV_ID																Res.	Res.	Res.	Res.	DEV_ID											
	Reset value(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
0x40015804	DBG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																														0	0	0
0x40015808	DBG_ APB1_FZ	DBG_LPTIMER_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value	0	0									0									0	0	0					0	0			0	0
0x4001580C	DBG_ APB2_FZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM22_STOP	Res.	Res.	DBG_TIM21_STOP	Res.	Res.
	Reset value																											0			0		

1. 复位值与产品有关。有关详细信息，请参见*第 33.4.1 节：MCU 器件 ID 代码*。

34 设备电子签名

除非特别说明，否则本部分适用于所有 STM32L0x3 器件。

电子签名存储在 Flash 模块的系统存储区中，可以使用 JTAG/SWD 或 CPU 对其进行读取。它包含出厂前编程的标识数据，这些标识数据允许用户固件或其它外部设备将其接口与 STM32L0x3 微控制器的特性自动匹配。

34.1 存储器大小寄存器

34.1.1 Flash 大小寄存器

Flash size register

基址：0x1FF8 007C

只读 = 0xXXXX，其中 X 是出厂前编程的

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 15:0 **F_SIZE**：Flash 大小 (Flash memory size)

此字段中存储的值指示器件的 Flash 大小（用 KB 表示）。

示例：0x0040 = 64 KB。

34.2 唯一设备 ID 寄存器（96 位）

唯一设备标识符最适合：

- 用作序列号
- 在对内部 Flash 进行编程前将唯一 ID 与软件加密原语和协议结合使用时用作安全密钥以提高 Flash 中代码的安全性
- 激活安全自举过程等

96 位的唯一设备标识符提供了一个对于任何设备和任何上下文都唯一的参考号码。用户永远不能改变这些位。

96 位的唯一设备标识符也可以以单字节/半字/字等不同方式读取，然后使用自定义算法连接起来。

基址：0x1FF8 0050

偏移地址：0x00

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

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

位 31:0 **U_ID(31:24)**：WAF_NUM[7:0]
晶圆编号（8 位无符号数）
U_ID(23:0)：LOT_NUM[55:32]
批号（ASCII 代码）

偏移地址：0x04

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 63:32 **U_ID(63:32)**：WAF_NUM[31:0]
批号（ASCII 代码）

偏移地址：0x14

只读 = 0xXXXX XXXX，其中 X 是出厂前编程的

95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 95:64 **U_ID(95:64)**：95:64 唯一 ID 位

附录 A 代码示例

A.1 简介

本附录给出了本参考手册中所述序列的代码示例。

这些代码示例摘自 *www.st.com* 上提供的 STM32L0xx 片段固件包 **STM32SnippetsL0**。

这些代码示例使用 CMSIS 头文件 (stm32l0xx.h) 中的外设位和寄存器说明。

如果之前已修改给定寄存器，则应对以 // 开头的代码行取消注释。

A.2 NVM/RCC 操作代码示例

A.2.1 提高 CPU 频率的准备序列代码

```
/* (1) 在 FLASH_ACR 的 Latency 位中设置一个等待周期 */
/* (2) 检查是否设置了延时 */
/* (3) 将时钟切换到 HSI16/4 并禁止 PLL */
/* (4) 将 PLLMUL 设置为 16，以获得 32 MHz 的 CPU 时钟频率 */
/* (5) 使能并接通 PLL */
FLASH->ACR |= FLASH_ACR_LATENCY; /* (1) */
while ((FLASH->ACR & FLASH_ACR_LATENCY) == 0); /* (2) */
SwitchFromPLLtoHSI(); /* (3) */
RCC->CFGR = (RCC->CFGR & (~(uint32_t)RCC_CFGR_PLLMUL))
            | RCC_CFGR_PLLMUL16; /* (4) */
SwitchOnPLL(); /* (5) */
```

A.2.2 降低 CPU 频率的准备序列代码

```
/* (1) 将时钟切换到 HSI16/4 并禁止 PLL */
/* (2) 将 PLLMUL 设置为 4，以获得 8MHz 的 CPU 时钟频率 */
/* (3) 使能并接通 PLL */
/* (4) 在 FLASH_ACR 的 Latency 位中设置一个等待周期 */
/* (5) 检查是否设置了延时 */
SwitchFromPLLtoHSI(); /* (1) */
RCC->CFGR = (RCC->CFGR & (~(uint32_t)RCC_CFGR_PLLMUL))
            | RCC_CFGR_PLLMUL4; /* (2) */
SwitchOnPLL(); /* (3) */
FLASH->ACR &= ~FLASH_ACR_LATENCY; /* (4) */
while ((FLASH->ACR & FLASH_ACR_LATENCY) != 0); /* (5) */
```

A.2.3 从 PLL 切换到 HSI16 的序列代码

```
uint32_t tickstart;
/* (1) 将时钟切换到 HSI16/4 */
/* (2) 等待时钟切换到 HSI16/4 */
/* (3) 通过复位 PLLON 禁止 PLL */
/* (4) 等待 PLLRDY 清零 */
RCC->CFGR = (RCC->CFGR & (~RCC_CFGR_SW)) | RCC_CFGR_SW_HSI; /* (1) */
tickstart = Tick;
while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI) /* (2) */
{
  if ((Tick - tickstart ) > CLOCKSWITCH_TIMEOUT_VALUE)
  {
    /* 管理错误 */
    return;
  }
}
RCC->CR &= ~RCC_CR_PLLON; /* (3) */
tickstart = Tick;
while ((RCC->CR & RCC_CR_PLLRDY) != 0) /* (4) */
{
  if ((Tick - tickstart ) > PLL_TIMEOUT_VALUE)
  {
    /* 管理错误 */
  }
}
```

注： *Tick 是 SysTick ISR 中的全局变量，每毫秒递增一次。*

A.2.4 切换到 PLL 的序列代码

```
uint32_t tickstart;
/* (1) 接通 PLL */
/* (2) 等待 PLL 就绪 */
/* (3) 将时钟切换到 PLL */
/* (4) 等待时钟切换到 PLL */
RCC->CR |= RCC_CR_PLLON; /* (1) */
tickstart = Tick;
while ((RCC->CR & RCC_CR_PLLRDY) == 0) /* (2) */
{
  if ((Tick - tickstart ) > PLL_TIMEOUT_VALUE)
  {
    error = ERROR_PLL_TIMEOUT; /* Report an error */
    return;
  }
}
RCC->CFGR = (RCC->CFGR & (~RCC_CFGR_SW)) | RCC_CFGR_SW_PLL; /* (3) */
```

```
tickstart = Tick;
while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL) /* (4) */
{
  if ((Tick - tickstart ) > CLOCKSWITCH_TIMEOUT_VALUE)
  {
    error = ERROR_CLKSWITCH_TIMEOUT; /* 报告错误 */
    return;
  }
}
```

注： *Tick 是 SysTick ISR 中的全局变量，每毫秒递增一次。*

A.3 NVM 操作代码示例

A.3.1 解锁数据 EEPROM 和 FLASH_PECR 寄存器的代码示例

```
/* (1) 等待至未执行任何操作 */
/* (2) 检查 PELOCK 是否解锁 */
/* (3) 执行解锁序列 */
while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (1) */
{
   /* 为获得稳定实现，在此处添加超时管理 */
}
if ((FLASH->PECR & FLASH_PECR_PELOCK) != 0) /* (2) */
{
  FLASH->PEKEYR = FLASH_PEKEY1; /* (3) */
  FLASH->PEKEYR = FLASH_PEKEY2;
}
```

A.3.2 锁定数据 EEPROM 和 FLASH_PECR 寄存器的代码示例

```
/* (1) 等待至未执行任何操作 */
/* (2) 通过将 PECR 中的 PELOCK 置 1 锁定 NVM */
while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (1) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
FLASH->PECR |= FLASH_PECR_PELOCK; /* (2) */
```

A.3.3 解锁 NVM 程序存储器的代码示例

```
/* (1) 等待至未执行任何操作 */
/* (2) 检查 PELOCK 是否解锁 */
/* (3) 检查 PRGLOCK 是否解锁 */
/* (4) 执行解锁序列 */
while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (1) */
```

```
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
if ((FLASH->PECR & FLASH_PECR_PELOCK) == 0) /* (2) */
{
  if ((FLASH->PECR & FLASH_PECR_PRGLOCK) != 0) /* (3) */
  {
    FLASH->PRGKEYR = FLASH_PRGKEY1; /* (4) */
    FLASH->PRGKEYR = FLASH_PRGKEY2;
  }
}
```

A.3.4 解锁选项字节区域的代码示例

```
/* (1) 等待至未执行任何操作 */
/* (2) 检查 PELOCK 是否解锁 */
/* (3) 检查 OPTLOCK 是否解锁 */
/* (4) 执行解锁序列 */
while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (1) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
if ((FLASH->PECR & FLASH_PECR_PELOCK) == 0) /* (2) */
{
  if ((FLASH->PECR & FLASH_PECR_OPTLOCK) != 0) /* (2) */
  {
    FLASH->OPTKEYR = FLASH_OPTKEY1; /* (3) */
    FLASH->OPTKEYR = FLASH_OPTKEY2;
  }
}
```

A.3.5 写入数据 EEPROM 的代码示例

```
*(uint8_t *)(DATA_E2_ADDR+i) = DATA_BYTE;
*(uint16_t *)(DATA_E2_ADDR+j) = DATA_16B_WORD;
*(uint32_t *)(DATA_E2_ADDR) = DATA_32B_WORD;
```

DATA_E2_ADDR 是数据 EEPROM 区域中的对齐地址。

i 可以是任何整数。

j 必须为偶数。

A.3.6 擦除数据 EEPROM 的代码示例

```
/* (1) 将 FLASH_PECR 寄存器中的 ERASE 和 DATA 位置 1
       以使能页擦除 */
/* (2) 在所需地址写入 32 位字值
       以启动擦除序列 */
```

```
/* (3) 进入等待中断阶段。EOP 检查在 Flash ISR 中完成
       */
/* (6) 将 FLASH_PECR 寄存器中的 ERASE 和 DATA 位复位
       以禁止页擦除 */
FLASH->PECR |= FLASH_PECR_ERASE | FLASH_PECR_DATA; /* (1) */
*(__IO uint32_t *)addr = (uint32_t)0; /* (2) */
__WFI(); /* (3) */
FLASH->PECR &= ~(FLASH_PECR_ERASE | FLASH_PECR_DATA); /* (4) */
```

A.3.7 编程选项字节的代码示例

```
/**
  * 此函数编程 16 位选项字节及其补码字。
  * 无参数
  * 无返回值
  */
__INLINE __RAM_FUNC void OptionByteProg(uint8_t index, uint16_t data)
{
  /* (1) 在选项字节地址写入 32 位字值，
         16 位数据通过其补码值扩展 */
  /* (3) 等待 FLASH_SR 寄存器中的 BSY 位复位 */
  /* (4) 检查 FLASH_SR 寄存器中的 EOP 标志 */
  /* (5) 用软件向 EOP 写入 1，将 EOP 标志清零 */
  *(__IO uint32_t *)(OB_BASE + index) = (uint32_t)((~data << 16) | data);
   /* (1) */
  while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (2) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
  if ((FLASH->SR & FLASH_SR_EOP) != 0) /* (3) */
  {
    FLASH->SR = FLASH_SR_EOP; /* (4) */
  }
  else
  {
    /* 管理错误情况 */
  }
}
```

注： *此函数必须加载到 RAM 中。*

A.3.8 擦除选项字节的代码示例

```
/**
  * 此函数擦除 16 位选项字节及其
    补码字。
  * 无参数
```

```
  * 无返回值
*/
__INLINE __RAM_FUNC void OptionByteErase(uint8_t index)
{
  /* (1) 将 FLASH_PECR 寄存器中的 ERASE 位置 1
         以使能选项字节擦除 */
  /* (2) 在要擦除的选项字节地址处写入 32 位字值，
         以启动擦除序列 */
  /* (3) 等待 FLASH_SR 寄存器中的 BSY 位复位 */
  /* (4) 检查 FLASH_SR 寄存器中的 EOP 标志 */
  /* (5) 用软件向 EOP 写入 1，将 EOP 标志清零 */
  /* (6) 将 FLASH_PECR 寄存器中的 ERASE 和 PROG 位复位
         以禁止页擦除 */
  FLASH->PECR |= FLASH_PECR_ERASE; /* (1) */
  *(__IO uint32_t *)(OB_BASE + index) = 0; /* (2) */
  while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (3) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
  if ((FLASH->SR & FLASH_SR_EOP) != 0) /* (4) */
  {
    FLASH->SR |= FLASH_SR_EOP; /* (5) */
  }
  else
  {
    /* 管理错误情况 */
  }
  FLASH->PECR &= ~(FLASH_PECR_ERASE); /* (6) */
}
```

注： *此函数必须加载到 RAM 中。*

A.3.9 对 Flash 程序存储器中的单个字进行编程的代码示例

```
/* (1) 在所需地址处执行数据写操作（32 位字） */
/* (2) 等待 FLASH_SR 寄存器中的 BSY 位复位 */
/* (3) 检查 FLASH_SR 寄存器中的 EOP 标志 */
/* (4) 用软件向其写入 1，将其清零 */
*(__IO uint32_t*)(flash_addr) = data; /* (1) */
while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (2) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
if ((FLASH->SR & FLASH_SR_EOP) != 0) /* (3) */
{
  FLASH->SR = FLASH_SR_EOP; /* (4) */
```

```
}
else
{
  /* 管理错误情况 */
}
```

A.3.10 对 Flash 程序存储器的半页进行编程的代码示例

```
/**
  * 此函数编程半页。它从 RAM 执行。
  * 连续编程时，编程位 (PROG) 和半页编程位 (FPRG)
  * 在函数开始时置 1，在函数结束时复位，
  * 这两个操作
  * 可在函数之外执行。
  * 此函数等待编程结束，将状态寄存器中的相应
  * 位清零，最后报告一个错误。
  * 参数 flash_addr 是要编程的半页的第一个地址
  * 数据是要编程的 32 位字阵列
  * 无返回值
  */
__RAM_FUNC void FlashHalfPageProg(uint32_t flash_addr, uint32_t *data)
{
  uint8_t i;
  /* (1) 将 FLASH_PECR 寄存器中的 PROG 和 FPRG 位置 1
         以使能半页编程 */
  /* (2) 在所需地址处执行数据写操作（半字） */
  /* (3) 等待 FLASH_SR 寄存器中的 BSY 位复位 */
  /* (4) 检查 FLASH_SR 寄存器中的 EOP 标志 */
  /* (5) 用软件向其写入 1，将其清零 */
  /* (6) 复位 PROG 和 FPRG 位以禁止编程 */
  FLASH->PECR |= FLASH_PECR_PROG | FLASH_PECR_FPRG; /* (1) */
  for (i = 0; i < ((FLASH_PAGE_SIZE/2) * 4); i+=4, data++)
  {
    *(__IO uint32_t*)(flash_addr + i) = *data; /* (2) */
  }
  while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (3) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
  if ((FLASH->SR & FLASH_SR_EOP) != 0) /* (4) */
  {
    FLASH->SR = FLASH_SR_EOP; /* (5) */
  }
  else
  {
```

```
    /* 管理错误情况 */
  }
  FLASH->PECR &= ~(FLASH_PECR_PROG | FLASH_PECR_FPRG); /* (6) */
}
```

注： *此函数必须加载到 RAM 中。*

A.3.11 擦除 Flash 程序存储器中一页的代码示例

```
/**
  * 此函数擦除 Flash 的一页。
  * 连续擦除时，页擦除位 (PER)
  * 在函数开始时置 1，在函数结束时复位，
  * 这两个操作可在函数之外执行。
  * 参数 page_addr 是要擦除的页内的一个地址
  * 无返回值
  */
__INLINE void FlashErase(uint32_t page_addr)
{
  /* (1) 将 FLASH_PECR 寄存器中的 ERASE 和 PROG 位置 1
         以使能页擦除 */
  /* (2) 在所选页的一个地址中写入 32 位字值
         以启动擦除序列 */
  /* (3) 等待 FLASH_SR 寄存器中的 BSY 位复位 */
  /* (4) 检查 FLASH_SR 寄存器中的 EOP 标志 */
  /* (5) 用软件向 EOP 写入 1，将 EOP 标志清零 */
  /* (6) 将 FLASH_PECR 寄存器中的 ERASE 和 PROG 位复位
         以禁止页擦除 */
  FLASH->PECR |= FLASH_PECR_ERASE | FLASH_PECR_PROG; /* (1) */
  *(__IO uint32_t *)page_addr = (uint32_t)0; /* (2) */
  while ((FLASH->SR & FLASH_SR_BSY) != 0) /* (3) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
  if ((FLASH->SR & FLASH_SR_EOP) != 0) /* (4) */
  {
    FLASH->SR = FLASH_SR_EOP; /* (5) */
  }
  else
  {
    /* 管理错误情况 */
  }
  FLASH->PECR &= ~(FLASH_PECR_ERASE | FLASH_PECR_PROG); /* (6) */
}
```

A.3.12 批量擦除代码示例

```
/**
  * 此函数对 Flash 执行批量擦除。
  * 此函数加载到 RAM 中。
  * 无参数
  * 成功时返回值，函数仅在从 RAM
    执行时才返回值
  */
__RAM_FUNC void FlashMassErase(void)
{
  /* (1) 检查读保护是否不为级别 2 */
  /* (2) 检查读保护是否不为级别 1 */
  /* (3) 擦除包含读保护的选项字节 */
  /* (4) 重载选项字节 */
  /* (5) 通过写入 0xAA，将读保护编程为级别 1
         以启动批量擦除 */
  /* (6) 通过将 PELOCK 位置 1 锁定 NVM */
  if ((FLASH->OPTR & 0x000000FF) == 0xCC) /* (1) */
  {
    /* 报告错误并中止 */
    return;
  }
  else if ((FLASH->OPTR & 0x000000FF) == 0xAA) /* (2) */
  {
    OptionByteErase(FLASH_OPTR0); /* (3) */
    FLASH->PECR |= FLASH_PECR_OBL_LAUNCH; /* (4) */
    /* 当执行选项字节重载时，MCU 将复位 */
  }
  OptionByteProg(FLASH_OPTR0, 0x00AA); /* (5) */
  if (*(uint32_t *)(FLASH_MAIN_ADDR ) != (uint32_t)0) /* 通过读取所有页值
  来检查页擦除情况 */
  {
    /* 报告错误 */
  }
  LockNVM(); /* (6) */
  while (1) /* 无限循环 */
  {
  }
}
```

注： *此函数使用 A.3.7：编程选项字节的代码示例和 A.3.8：擦除选项字节的代码示例中的其他两个函数。*

A.4 时钟控制器

A.4.1 HSE 启动序列代码示例

```
/**
  * 此函数使能 HSE 就绪中断，
  * 并启动 HSE 作为外部时钟。
  * 无参数
  * 无返回值
*/
__INLINE void StartHSE(void)
{
  /* 为 RCC 配置 NVIC */
  /* (1) 使能 RCC 中断 */
  /* (2) 为 RCC 设置优先级 */
  NVIC_EnableIRQ(RCC_CRS_IRQn); /* (1) */
  NVIC_SetPriority(RCC_CRS_IRQn,0); /* (2) */

  /* (1) 使能 HSE 就绪中断 */
  /* (2) 使能 CSS
         使能 HSE 并将 HSEBYP 置 1 以使用外部时钟
         代替振荡器
         使能 HSE */
  /* 注：在 RCC_CRS_IRQHandler ISR 中将时钟切换到 HSE */
  RCC->CIER |= RCC_CIER_HSERDYIE; /* (1) */
  RCC->CR |= RCC_CR_CSSHSEON | RCC_CR_HSEBYP | RCC_CR_HSEON; /* (2) */
}

/**
  * 此函数处理 RCC 中断请求
  * 并将系统时钟切换为 HSE。
  * 无参数
  * 无返回值
  */
void RCC_CRS_IRQHandler(void)
{
  /* (1) 检查 HSE 就绪标志 */
  /* (2) 清零 HSE 就绪标志 */
  /* (3) 将系统时钟切换为 HSE */
  if ((RCC->CifR & RCC_CifR_HSERDYF) != 0) /* (1) */
  {
    RCC->CICR |= RCC_CICR_HSERDYC; /* (2) */
    RCC->CFGR = ((RCC->CFGR & (~RCC_CFGR_SW)) | RCC_CFGR_SW_HSE); /* (3) */
  }
  else
```

```
  {
    /* 管理错误 */
  }
}
```

A.4.2 PLL 配置修改代码示例

```
/* (1) 测试 PLL 是否用作系统时钟 */
/* (2) 选择 HSI 作为系统时钟 */
/* (3) 等待 HSI 切换 */
/* (4) 禁止 PLL */
/* (5) 等待 PLLRDY 清零 */
/* (6) 将延时设置为 1 个等待周期 */
/* (7) 将 PLL 倍频比设置为 24，将 PLL 分频比设置为 3 */
/* (8) 使能 PLL */
/* (9) 等待 PLLRDY 置 1 */
/* (10) 选择 PLL 作为系统时钟 */
/* (11) 等待 PLL 接通 */
if ((RCC->CFGR & RCC_CFGR_SWS) == RCC_CFGR_SWS_PLL) /* (1) */
{
  RCC->CFGR = (RCC->CFGR & (uint32_t) (~RCC_CFGR_SW))
              | RCC_CFGR_SW_HSI; /* (2) */
  while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI) /* (3) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
}
RCC->CR &= (uint32_t)(~RCC_CR_PLLON);/* (4) */
while((RCC->CR & RCC_CR_PLLRDY) != 0) /* (5) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
FLASH->ACR |= FLASH_ACR_LATENCY; /* (6) */
RCC->CFGR = RCC->CFGR & (~(RCC_CFGR_PLLMUL| RCC_CFGR_PLLDIV ))
          | (RCC_CFGR_PLLMUL24 | RCC_CFGR_PLLDIV2); /* (7) */
RCC->CR |= RCC_CR_PLLON; /* (8) */
while ((RCC->CR & RCC_CR_PLLRDY) == 0) /* (9) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
RCC->CFGR |= (uint32_t) (RCC_CFGR_SW_PLL); /* (10) */
while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL) /* (11) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
```

A.4.3 MCO 选择代码示例

```
/* (1) 清零 MCO 选择位 */
/* (2) 选择 1/4 系统时钟在 MCO 上输出，不使用预分频器 */
RCC->CFGR &= (uint32_t) RCC_CFGR_MCOSEL; /* (1) */
RCC->CFGR |= RCC_CFGR_MCO_SYSCLK | RCC_CFGR_MCO_PRE_4; /* (2) */
```

A.5 GPIO

A.5.1 锁定机制代码示例

```
/* (1) 向 LCKK 位写 1 并将引脚位置 1 以进行锁定 */
/* (2) 向 LCKK 位写 0 并将引脚位置 1 以进行锁定 */
/* (3) 向 LCKK 位写 1 并将引脚位置 1 以进行锁定 */
/* (4) 读取锁定寄存器 */
/* (5) 检查锁定寄存器（可选） */
GPIOA->LCKR = GPIO_LCKR_LCKK + lock; /* (1) */
GPIOA->LCKR = lock; /* (2) */
GPIOA->LCKR = GPIO_LCKR_LCKK + lock; /* (3) */
GPIOA->LCKR; /* (4) */
if ((GPIOA->LCKR & GPIO_LCKR_LCKK) == 0) /* (5) */
{
  /* 管理错误 */
}
```

A.5.2 复用功能选择序列代码示例

```
/* (1) 使能定时器 2 的外设时钟 */
/* (2) 使能 GPIOA 的外设时钟 */
/* (3) 选择 GPIOA 引脚 0 上的复用功能模式 (10) */
/* (4) 通过在 GPIOA AFRL 寄存器中使能引脚 0 的 AF2 来选择
       PA0 上的 TIM2_CH1 */
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN; /* (1) */
RCC->IOPENR |= RCC_IOPENR_GPIOAEN; /* (2) */
GPIOA->MODER = (GPIOA->MODER & ~(GPIO_MODER_MODE0)) \
             | (GPIO_MODER_MODE0_1); /* (3) */
GPIOA->AFR[0] |= 0x2; /* (4) */
```

A.5.3 模拟 GPIO 配置代码示例

```
/* (1) 使能 GPIOA 的外设时钟 */
/* (2) 选择 GPIOA 引脚 0 上的模拟模式（00 - 默认值） */
RCC->IOPENR |= RCC_IOPENR_GPIOAEN; /* (1) */
GPIOA->MODER &= ~(GPIO_MODER_MODE0); /* (2) */
```

A.6 DMA

A.6.1 DMA 通道配置序列代码示例

```
/* (1) 使能 DMA 的外设时钟 */
/* (2) 将 DMA 通道 1 重映射到 ADC（复位值） */
/* (3) 使能 ADC 上的 DMA 传输 */
/* (4) 配置外设数据寄存器地址 */
/* (5) 配置存储器地址 */
/* (6) 配置通道 1 上要执行的 DMA 传输的次数 */
/* (7) 配置递增、大小和中断 */
/* (8) 使能 DMA 通道 1 */
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
//DMA1_CSELR->CSELR &= (uint32_t)(~DMA_CSELR_C1S); /* (2) */
ADC1->CFGR1 |= ADC_CFGR1_DMAEN; /* (3) */
DMA1_Channel1->CPAR = (uint32_t) (&(ADC1->DR)); /* (4) */
DMA1_Channel1->CMAR = (uint32_t)(ADC_array); /* (5) */
DMA1_Channel1->CNDTR = 3; /* (6) */
DMA1_Channel1->CCR |= DMA_CCR_MINC | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_TEIE | DMA_CCR_TCIE ; /* (7) */
DMA1_Channel1->CCR |= DMA_CCR_EN; /* (8) */

/* 为 DMA 配置 NVIC */
/* (1) 使能 DMA 通道 1 上的中断 */
/* (2) 为 DMA 通道 1 设置优先级 */
NVIC_EnableIRQ(DMA1_Channel1_IRQn); /* (1) */
NVIC_SetPriority(DMA1_Channel1_IRQn,0); /* (2) */
```

A.7 中断和事件

A.7.1 NVIC 初始化示例

```
/* 为 ADC 配置 NVIC */
/* (1) 使能 ADC 中断 */
/* (2) 为 ADC 设置优先级 */
NVIC_EnableIRQ(ADC1_COMP_IRQn); /* (1) */
NVIC_SetPriority(ADC1_COMP_IRQn,0); /* (2) */
```

A.7.2 扩展中断选择代码示例

```
/* (1) 使能 GPIOA 的外设时钟 */
/* (2) 选择 GPIOA 引脚 0 上的输入模式 (00) */
/* (3) 通过在 EXTI0 中写入 0000（复位值）为
       引脚 0 扩展中断选择端口 A */
/* (4) 配置 EXTI_IMR 寄存器中相应的屏蔽位 */
```

```
/* (5) 配置中断线的上升沿触发
       选择位 */
/* (6) 配置中断线的下降沿触发
       选择位 */
RCC->IOPENR |= RCC_IOPENR_GPIOAEN; /* (1) */
GPIOA->MODER = (GPIOA->MODER & ~(GPIO_MODER_MODE0)); /* (2) */
//SYSCFG->EXTICR[0] &= (uint16_t)~SYSCFG_EXTICR1_EXTI0_PA; /* (3) */
EXTI->IMR |= 0x0001; /* (4) */
EXTI->RTSR |= 0x0001; /* (5) */
//EXTI->FTSR |= 0x0001; /* (6) */

/* 为扩展中断配置 NVIC */
/* (7) 使能 EXTI0_1 中断 */
/* (8) 为 EXTI0_1 设置优先级 */
NVIC_EnableIRQ(EXTI0_1_IRQn); /* (7) */
NVIC_SetPriority(EXTI0_1_IRQn,0); /* (8) */
```

A.8 ADC

A.8.1 校准代码示例

```
/* (1) 确保 ADEN = 0 */
/* (2) 清零 ADEN */
/* (3) 设置 ADCAL=1 */
/* (4) 等待 EOCAL=1 */
/* (5) 清零 EOCAL */
if ((ADC1->CR & ADC_CR_ADEN) != 0) /* (1) */
{
  ADC1->CR &= (uint32_t)(~ADC_CR_ADEN); /* (2) */
}
ADC1->CR |= ADC_CR_ADCAL; /* (3) */
while ((ADC1->ISR & ADC_ISR_EOCAL) == 0) /* (4) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
ADC1->ISR |= ADC_ISR_EOCAL; /* (5) */
```

A.8.2 ADC 使能序列代码示例

```
/* (1) 清零 ADRDY 位 */
/* (2) 使能 ADC */
/* (3) 等待 ADC 就绪 */
ADC1->ISR |= ADC_ISR_ADRDY; /* (1) */
ADC1->CR |= ADC_CR_ADEN; /* (2) */
if ((ADC1->CFGR1 & ADC_CFGR1_AUTOFF) == 0)
```

```
{
  while ((ADC1->ISR & ADC_ISR_ADRDY) == 0) /* (3) */
  {
    /* 为获得稳定实现，在此处添加超时管理 */
  }
}
```

A.8.3 ADC 禁止序列代码示例

```
/* (1) 确保当前未进行任何转换 */
/* (2) 停止正在进行的任何转换 */
/* (3) 等待 ADSTP 被硬件复位，即转换停止 */
/* (4) 禁止 ADC */
/* (5) 等待 ADC 完全禁止 */
if ((ADC1->CR & ADC_CR_ADSTART) != 0) /* (1) */
{
  ADC1->CR |= ADC_CR_ADSTP; /* (2) */
}
while ((ADC1->CR & ADC_CR_ADSTP) != 0) /* (3) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
ADC1->CR |= ADC_CR_ADDIS; /* (4) */
while ((ADC1->CR & ADC_CR_ADEN) != 0) /* (5) */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
```

A.8.4 MCO 时钟选择代码示例

```
/* (1) 通过在 CKMODE 中写入 11 来选择 PCLK */
ADC1->CFGR2 |= ADC_CFGR2_CKMODE; /* (1) */
```

A.8.5 单次转换序列代码示例 - 软件触发

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择自动关闭模式 */
/* (3) 为 VRefInt 选择 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk 大于 17.1 us */
/* (5) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_AUTOFF; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL17; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC->CCR |= ADC_CCR_VREFEN; /* (5) */
…
```

```
/* 执行 AD 转换 */
ADC1->CR |= ADC_CR_ADSTART; /* 启动 ADC 转换 */
while ((ADC1->ISR & ADC_ISR_EOC) == 0) /* 等待转换结束 */
{
  /* 为获得稳定实现，在此处添加超时管理 */
}
```

A.8.6 连续转换序列代码示例 - 软件触发

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择连续模式和扫描方向 */
/* (3) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk 大于 5 µs */
/* (5) 使能 EOC 中断、EOSEQ 中断和上溢中断 */
/* (6) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_WAIT |ADC_CFGR1_CONT | ADC_CFGR1_SCANDIR;/* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9 \
             | ADC_CHSELR_CHSEL17; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_EOCIE | ADC_IER_EOSEQIE | ADC_IER_OVRIE; /* (5) */
ADC->CCR |= ADC_CCR_VREFEN; /* (6) */

/* 为 ADC 配置 NVIC */
/* (1) 使能 ADC 中断 */
/* (2) 为 ADC 设置优先级 */
NVIC_EnableIRQ(ADC1_COMP_IRQn); /* (1) */
NVIC_SetPriority(ADC1_COMP_IRQn,0); /* (2) */
```

A.8.7 单次转换序列代码示例 - 硬件触发

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 通过选择 TRG4 (EXTSEL = 100) 来选择下降沿外部触发和 TIM22_TRGO
       外部触发 */
/* (3) 为 VRefInt 选择 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk 大于 5 µs */
/* (5) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_EXTEN_0 | ADC_CFGR1_EXTSEL_2 ; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL17; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC->CCR |= ADC_CCR_VREFEN; /* (5) */
```

注： *之后，必须将 TIM22 配置为定期生成 TRG0 外部触发信号。*

A.8.8 连续转换序列代码示例 - 硬件触发

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择 TIM22_TRGO 外部触发（TRG4，即 EXTSEL = 100）和上升沿外部触发、
       连续模式和扫描方向 */
/* (3) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk 大于 5 µs */
/* (5) 使能 EOC 中断、EOSEQ 中断和上溢中断 */
/* (6) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_EXTEN_0 | ADC_CFGR1_EXTSEL_2 | ADC_CFGR1_CONT \
             | ADC_CFGR1_SCANDIR; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9 \
             | ADC_CHSELR_CHSEL17; /* (3 */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_EOCIE | ADC_IER_EOSEQIE | ADC_IER_OVRIE; /* (5) */
ADC->CCR |= ADC_CCR_VREFEN; /* (6) */
/* 为 ADC 配置 NVIC */
/* (1) 使能 ADC 中断 */
/* (2) 为 ADC 设置优先级 */
NVIC_EnableIRQ(ADC1_COMP_IRQn); /* (1) */
NVIC_SetPriority(ADC1_COMP_IRQn,0); /* (2) */
```

A.8.9 DMA 单触发模式序列代码示例

```
/* (1) 使能 DMA 的外设时钟 */
/* (2) 使能 ADC 上的 DMA 传输 - DMACFG 在单触发模式下保持为 0 */
/* (3) 配置外设数据寄存器地址 */
/* (4) 配置存储器地址 */
/* (5) 配置 DMA 通道 1 上要执行的
       DMA 传输的次数 */
/* (6) 配置递增、大小和中断 */
/* (7) 使能 DMA 通道 1 */
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_DMAEN; /* (2) */
DMA1_Channel1->CPAR = (uint32_t) (&(ADC1->DR)); /* (3) */
DMA1_Channel1->CMAR = (uint32_t)(ADC_array); /* (4) */
DMA1_Channel1->CNDTR = NUMBER_OF_ADC_CHANNEL; /* (5) */
DMA1_Channel1->CCR |= DMA_CCR_MINC | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_TEIE | DMA_CCR_TCIE ; /* (6) */
DMA1_Channel1->CCR |= DMA_CCR_EN; /* (7) */
```

A.8.10 DMA 循环模式序列代码示例

```
/* (1) 使能 DMA 的外设时钟 */
/* (2) 使能 ADC 上的 DMA 传输以及循环模式 */
/* (3) 配置外设数据寄存器地址 */
/* (4) 配置存储器地址 */
/* (5) 配置 DMA 通道 1 上要执行的
       DMA 传输的次数 */
/* (6) 配置递增、大小、中断和循环模式 */
/* (7) 使能 DMA 通道 1 */
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_DMAEN | ADC_CFGR1_DMACFG; /* (2) */
DMA1_Channel1->CPAR = (uint32_t) (&(ADC1->DR)); /* (3) */
DMA1_Channel1->CMAR = (uint32_t)(ADC_array); /* (4) */
DMA1_Channel1->CNDTR = NUMBER_OF_ADC_CHANNEL; /* (5) */
DMA1_Channel1->CCR |= DMA_CCR_MINC | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_TEIE | DMA_CCR_CIRC; /* (6) */
DMA1_Channel1->CCR |= DMA_CCR_EN; /* (7) */
```

A.8.11 等待模式序列代码示例

```
/* (1) 通过在 CKMODE 中写入 11 来选择 PCLK */
/* (2) 选择连续模式和等待模式 */
/* (3) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk
       大于 17.1 µs */
/* (5) 使能上溢中断 */
/* (6) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
ADC1->CFGR2 |= ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_CONT | ADC_CFGR1_WAIT; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9 \
             | ADC_CHSELR_CHSEL17; /* (3 */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_OVRIE; /* (5) */
ADC->CCR |= ADC_CCR_VREFEN; /* (6) */
```

A.8.12 自动关闭和无等待模式序列代码示例

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择 TIM22_TRGO 外部触发和下降沿外部触发、连续模式、
       扫描方向和自动关闭 */
/* (3) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk
       大于 5 µs */
/* (5) 使能 EOC 中断、EOSEQ 中断和上溢中断 */
/* (6) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
```

```
ADC1->CFGR1 |= ADC_CFGR1_EXTEN_0 | ADC_CFGR1_EXTSEL_2 \
             | ADC_CFGR1_SCANDIR | ADC_CFGR1_AUTOFF; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9 \
             | ADC_CHSELR_CHSEL17; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_EOCIE | ADC_IER_EOSEQIE | ADC_IER_OVRIE; /* (5) */
ADC->CCR |= ADC_CCR_VREFEN; /* (6) */
```

A.8.13 自动关闭和等待模式序列代码示例

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择连续模式、等待模式和自动关闭 */
/* (3) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (4) 选择采样模式 111，即 239.5 ADC clk
       大于 5 µs */
/* (5) 使能上溢中断 */
/* (6) 唤醒 VREFINT（仅限温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_CONT | ADC_CFGR1_WAIT | ADC_CFGR1_AUTOFF;/* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9 \
             | ADC_CHSELR_CHSEL17; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_OVRIE; /* (5) */
ADC->CCR |= ADC_CCR_VREFEN; /* (6) */
```

A.8.14 模拟看门狗代码示例

```
/* 定义比出厂值高 15% 的上限值，
   该值可根据应用电源和
   出厂校准电源调整 */
uint16_t vrefint_high = (*VREFINT_CAL_ADDR)* VDD_CALIB / VDD_APPLI * 115 /
100;

/* 定义比出厂值低 15% 的下限值，
   该值可根据应用电源和
   出厂校准电源调整 */
uint16_t vrefint_low = (*VREFINT_CAL_ADDR) * VDD_CALIB / VDD_APPLI * 85 /
100;

/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择连续模式
       并将模拟看门狗配置为仅监视 CH17 */
/* (3) 定义模拟看门狗的范围：16b-MSW 为上限，
       16b-LSW 为下限 */
/* (4) 选择 CHSEL4、CHSEL9 和 CHSEL17 */
/* (5) 选择采样模式 111，即 239.5 ADC clk 大于 5 µs */
```

```
/* (6) 使能 EOC 中断、EOSEQ 中断和模拟看门狗中断 */
/* (7) 唤醒 VREFINT（仅限 VBAT、温度传感器和 VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_CONT \
            | (17<<26) | ADC_CFGR1_AWDEN | ADC_CFGR1_AWDSGL; /* (2) */
ADC1->TR = (vrefint_high << 16) + vrefint_low; /* (3 */
ADC1->CHSELR = ADC_CHSELR_CHSEL4 | ADC_CHSELR_CHSEL9
             | ADC_CHSELR_CHSEL17; /* (4) */
ADC1->SMPR |= ADC_SMPR_SMP_0 | ADC_SMPR_SMP_1 | ADC_SMPR_SMP_2; /* (5) */
ADC1->IER = ADC_IER_EOCIE | ADC_IER_EOSEQIE | ADC_IER_AWDIE; /* (6) */
ADC->CCR |= ADC_CCR_VREFEN; /* (7) */
```

A.8.15 过采样代码示例

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16
       使能 16 倍过采样，移位 1，
       无触发 */
ADC1->CFGR2 = (ADC1->CFGR2 & (~ADC_CFGR2_CKMODE))
            | (ADC_CFGR2_OVSE | ADC_CFGR2_OVSR_1 | ADC_CFGR2_OVSR_0
            | ADC_CFGR2_OVSS_0); /* (1) */
```

A.8.16 温度配置代码示例

```
/* (1) 通过在 CKMODE 中写入 00（复位值）来选择 HSI16 */
/* (2) 选择连续模式 */
/* (3) 为温度传感器选择 CHSEL18 */
/* (4) 选择采样模式 111，即 239.5 ADC clk
       大于 2.2 µs */
/* (5) 唤醒温度传感器（仅限温度传感器和
       VRefInt） */
//ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_CONT; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL18; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP; /* (4) */
ADC->CCR |= ADC_CCR_TSEN; /* (5) */
```

A.8.17 温度计算代码示例

```
/* 温度传感器校准值地址 */
#define TEMP130_CAL_ADDR ((uint16_t*) ((uint32_t) 0x1FF8007E))
#define TEMP30_CAL_ADDR ((uint16_t*) ((uint32_t) 0x1FF8007A))
#define VDD_CALIB ((uint16_t) (300))
#define VDD_APPLI ((uint16_t) (330))
int32_t ComputeTemperature(uint32_t measure)
{
  int32_t temperature;
  temperature = ((measure * VDD_APPLI / VDD_CALIB)
```

```
                - (int32_t) *TEMP30_CAL_ADDR );
  temperature = temperature * (int32_t)(130 - 30);
  temperature = temperature / (int32_t)(*TEMP130_CAL_ADDR -
                                        *TEMP30_CAL_ADDR);
  temperature = temperature + 30;
  return(temperature);
}
```

A.9 DAC

A.9.1 独立触发（不产生波形）代码示例

```
/* (1) 使能 DAC 的外设时钟 */
/* (2) 将 WAVE1 配置为 01，将 LFSR 掩码振幅 (MAMP1)
       配置为 1000 以获得 511 位振幅，
       使能 DAC ch1，
       禁止 ch1 上的缓冲区，
       并通过保持 TSEL1 中为 000 来选择 TIM6 作为触发信号 */
RCC->APB1ENR |= RCC_APB1ENR_DACEN; /* (1) */
DAC->CR |= DAC_CR_WAVE1_0 | DAC_CR_MAMP1_3 \
         | DAC_CR_BOFF1 | DAC_CR_TEN1 | DAC_CR_EN1; /* (2) */
DAC->DHR12R1 = DAC_OUT1_VALUE; /* 初始化 DAC 输出值 */
```

A.9.2 独立触发（生成单个三角波）代码示例

```
/* (1) 使能 DAC 的外设时钟 */
/* (2) 将 WAVE1 配置为 10，将 LFSR 掩码振幅 (MAMP1)
       配置为 1001 以获得 1023 位振幅，
       使能 DAC ch1，
       禁止 ch1 上的缓冲区，
       并通过保持 TSEL1 中为 000 来选择 TIM6 作为触发信号 */
RCC->APB1ENR |= RCC_APB1ENR_DACEN; /* (1) */
DAC->CR |= DAC_CR_WAVE1_1 | DAC_CR_MAMP1_3 | DAC_CR_MAMP1_0 \
         | DAC_CR_BOFF1 | DAC_CR_TEN1 | DAC_CR_EN1; /* (2) */
DAC->DHR12R1 = DAC_OUT1_VALUE; /* 定义通道 1 上的三角波的低电平值 */
```

A.9.3 DMA 初始化代码示例

```
/* (1) 使能 DAC 的外设时钟 */
/* (2) 使能 DAC 通道 1 上的 DMA 传输，
       使能 DMA 下溢 DAC 上的中断，
       使能 DAC 通道 1，使能通道 1 上的触发信号，
       禁止通道 1 上的缓冲区，
       并通过保持 TSEL1 中为 000 来选择 TIM6 作为触发信号 */
RCC->APB1ENR |= RCC_APB1ENR_DACEN; /* (1) */
```

```
DAC->CR |= DAC_CR_DMAUDRIE1 | DAC_CR_DMAEN1 | DAC_CR_BOFF1
          | DAC_CR_TEN1 | DAC_CR_EN1; /* (2) */

/* (1) 使能 DMA 的外设时钟 */
/* (2) 将 DAC 重映射到 DMA 通道 2 上 */
/* (3) 配置外设数据寄存器地址 */
/* (4) 配置存储器地址 */
/* (5) 配置 DMA 通道 x 上要执行的
       DMA 传输的次数 */
/* (6) 配置递增、大小（16 位）、中断、从存储器传输到外设
       以及循环模式 */
/* (7) 使能 DMA 通道 x */
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
DMA1_CSELR->CSELR |= (uint32_t)(9 << 4); /* (2) */
DMA1_Channel2->CPAR = (uint32_t) (&(DAC->DHR12R1)); /* (3) */
DMA1_Channel2->CMAR = (uint32_t)(sin_data); /* (4) */
DMA1_Channel2->CNDTR = SIN_ARRAY_SIZE; /* (5) */
DMA1_Channel2->CCR |= DMA_CCR_MINC | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_DIR | DMA_CCR_TEIE | DMA_CCR_CIRC; /* (6) */
DMA1_Channel2->CCR |= DMA_CCR_EN; /* (7) */

/* 为 DMA 配置 NVIC */
/* (1) 使能 DMA 通道 x 上的中断 */
/* (2) 为 DMA 通道 x 设置优先级 */
NVIC_EnableIRQ(DMA1_Channel2_3_IRQn); /* (1) */
NVIC_SetPriority(DMA1_Channel2_3_IRQn,3); /* (2) */
```

A.10 TSC 代码示例

A.10.1 TSC 配置代码示例

```
/* 配置 TSC */
/* 电荷转移周期约为 8 µs */
/* (1)选择 fPGCLK = fHCLK/32
       设置脉冲高电平 = 2xtPGCLK，主器件
       设置脉冲低电平 = 2xtPGCLK
       设置最大计数值 = 16383 个脉冲
       使能 TSC */
/* (2) 禁止滞后 */
/* (3) 使能采集结束中断 */
/* (4) 使能采样，G2IO4 */
/* (5) 使能通道，G2IO3 */
/* (6) 使能组，G2 */
TSC->CR = TSC_CR_PGPSC_2 | TSC_CR_PGPSC_0 | TSC_CR_CTPH_0 | TSC_CR_CTPL_0
        | TSC_CR_MCV_2 | TSC_CR_MCV_1 | TSC_CR_TSCE; /* (1) */
```

```
TSC->IOHCR &= (uint32_t)(~(TSC_IOHCR_G2_IO3 | TSC_IOHCR_G2_IO4)); /* (2) */
TSC->IER = TSC_IER_EOAIE; /* (3) */
TSC->IOSCR = TSC_IOSCR_G2_IO4; /* (4) */
TSC->IOCCR = TSC_IOCCR_G2_IO3; /* (5) */
TSC->IOGCSR |= TSC_IOGCSR_G2E; /* (6) */
```

A.10.2 TSC 中断代码示例

```
/* 采集结束标志 */
if((TSC->ISR & TSC_ISR_EOAF) == TSC_ISR_EOAF)
{
  TSC->ICR = TSC_ICR_EOAIC; /* 清零标志，TSC_ICR_EOAIC = 1 */
  AcquisitionValue = TSC->IOGXCR[1]; /* 获取 G2 计数器值 */
}
```

A.11 定时器

A.11.1 TI2 上升沿递增计数器代码示例

```
/* (1) 通过在 TIMx_CCMR1 寄存器中写入 CC1S =“01”
       将通道 1 配置为检测 TI1 输入的上升沿，并通过写入
       TIMx_CCMR1 寄存器中的 IC1F[3:0] 位配置输入滤波器
       的持续时间（如果无需滤波器，则保持 IC2F=0000）。*/
/* (2) 通过在 TIMx_CCER 寄存器中写入 CC1P=0 来
       选择上升沿极性
       当它保持复位值时则不必要。*/
/* (3) 通过在 TIMx_SMCR 寄存器中写入 SMS=111，使定时器在外部时钟模式 1 下工作
       通过在 TIMx_SMCR 寄存器中写入 TS=101 来选择
       TI1 作为触发输入源。*/
/* (4) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/
TIMx->CCMR1 |= TIM_CCMR1_IC1F_0 | TIM_CCMR1_IC1F_1
             | TIM_CCMR1_CC1S_0; /* (1) */
//TIMx->CCER &= (uint16_t)(~TIM_CCER_CC1P); /* (2) */
TIMx->SMCR |= TIM_SMCR_SMS | TIM_SMCR_TS_2 | TIM_SMCR_TS_0; /* (3) */
TIMx->CR1 |= TIM_CR1_CEN; /* (4) */
```

A.11.2 每 2 个 ETR 上升沿递增计数器代码示例

```
/* (1) 由于此例中不需要滤波器，因此在 TIMx_SMCR 寄存器中写入 ETF[3:0]=0000。
       保持复位值。
       通过在 TIMx_SMCR 寄存器中写入 ETPS[1:0]=01 来
       设置预分频比。
       通过在 TIMx_SMCR 寄存器中写入 ETP=0 来选择 ETR 引脚的上升沿检测。
       保持复位值。
       通过在 TIMx_SMCR 寄存器中写入 ECE=1 来
       使能外部时钟模式 2。*/
```

```
/* (2) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/
TIMx->SMCR |= TIM_SMCR_ETPS_0 | TIM_SMCR_ECE; /* (1) */
TIMx->CR1 |= TIM_CR1_CEN; /* (2) */
```

A.11.3 输入捕捉配置代码示例

```
/* (1) 选择有效输入 TI1 (CC1S = 01)， 编程输入滤波器
       持续工作 8 个时钟周期 (IC1F = 0011)，
       选择 CC1 的上升沿（CC1P = 0，复位值）和每次有效跳变时的
       预分频比（IC1PS = 00，复位值） */
/* (2) 通过将 CC1E 置 1 使能捕捉 */
/* (3) 使能捕捉/比较中断 */
/* (4) 使能计数器 */
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0 \
             | TIM_CCMR1_IC1F_0 | TIM_CCMR1_IC1F_1; /* (1) */
TIMx->CCER |= TIM_CCER_CC1E; /* (2) */
TIMx->DIER |= TIM_DIER_CC1IE; /* (3) */
TIMx->CR1 |= TIM_CR1_CEN; /* (4) */
```

A.11.4 输入捕捉数据管理代码示例

此代码必须插入到定时器中断子程序中。

```
if ((TIMx->SR & TIM_SR_CC1IF) != 0)
{
  if ((TIMx->SR & TIM_SR_CC1OF) != 0) /* 检查溢出 */
  {
    /* 溢出错误管理 */
    gap = 0; /* 重新初始化 laps 计算 */
    TIMx->SR = ~(TIM_SR_CC1OF | TIM_SR_CC1IF); /* 清零标志 */
    return;
  }
  if (gap == 0) /* 测试是否为第一个上升沿 */
  {
    counter0 = TIMx->CCR1; /* 读取捕捉计数器，这用于清零 CC1ICF */
    gap = 1; /* 指示已检测到第一个上升沿 */
  }
  else
  {
    counter1 = TIMx->CCR1; /* 读取捕捉计数器，这用于清零 CC1ICF */
    if (counter1 > counter0) /* 检查捕捉计数器溢出情况 */
    {
      Counter = counter1 - counter0;
    }
    else
    {
```

```
        Counter = counter1 + 0xFFFF - counter0 + 1;
      }
      counter0 = counter1;
    }
  }
else
{
  /* 管理错误 */
}
```

注：此代码仅管理单个计数器的溢出情况。要管理多个计数器的溢出情况，必须使能并正确管理更新中断 (UIE = 1)。

A.11.5 PWM 输入配置代码示例

```
/* (1) 为 TIMx_CCR1 选择有效输入 TI1 (CC1S = 01),
       为 TIMx_CCR2 选择有效输入 TI1 (CC2S = 10) */
/* (2) 选择 TI1FP1 作为有效触发输入 (TS = 101)
       将从模式配置为复位模式 (SMS = 100) */
/* (3) 通过将 CC1E 和 CC2E 置 1 使能捕捉 */
       选择 CC1 和 CC1N 的上升沿（CC1P = 0 且 CC1NP = 0，复位值），
       选择 CC2 的下降沿 (CC2P = 1)。*/
/* (4) 使能捕捉/比较 1 中断 */
/* (5) 使能计数器 */
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0 | TIM_CCMR1_CC2S_1; /* (1) */
TIMx->SMCR |= TIM_SMCR_TS_2 | TIM_SMCR_TS_0 \
            | TIM_SMCR_SMS_2; /* (2) */
TIMx->CCER |= TIM_CCER_CC1E | TIM_CCER_CC2E | TIM_CCER_CC2P; /* (3) */
TIMx->DIER |= TIM_DIER_CC1IE; /* (4) */
TIMx->CR1 |= TIM_CR1_CEN; /* (5) */
```

A.11.6 采用 DMA 的 PWM 输入配置代码示例

```
/* (1) 使能 DMA 的外设时钟 */
/* (2) 通过在 DMA_CSELR_C5S 和 DMA_CSELR_C7S 中写入 1000
       将 DMA 通道 5 和 7 映射到 TIM2_CH1 和 TIM2_CH2 */
/* (3) 为 DMA 通道 x 配置外设数据寄存器地址 */
/* (4) 为 DMA 通道 x 配置存储器地址 */
/* (5) 配置 DMA 通道 x 上要执行的
       DMA 传输的次数 */
/* (6) 为 DMA 通道 x 配置无递增（复位值）、大小（16 位）、中断、从外设传输到存储器
       和循环模式 */
/* (7) 使能 DMA 通道 x */
```

```
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
DMA1_CSELR->CSELR |= 8 << (4 * (5-1)) | 8 << (4 * (7-1)); /* (2) */
DMA1_Channel5->CPAR = (uint32_t) (&(TIMx->CCR1)); /* (3) */
DMA1_Channel5->CMAR = (uint32_t)(&Period); /* (4) */
DMA1_Channel5->CNDTR = 1; /* (5) */
DMA1_Channel5->CCR |= DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_TEIE | DMA_CCR_CIRC; /* (6) */
DMA1_Channel5->CCR |= DMA_CCR_EN; /* (7) */
/* 为通道 6 重复执行 (3) 到 (6) */
DMA1_Channel7->CPAR = (uint32_t) (&(TIMx->CCR2)); /* (2) */
DMA1_Channel7->CMAR = (uint32_t)(&DutyCycle); /* (3) */
DMA1_Channel7->CNDTR = 1; /* (4) */
DMA1_Channel7->CCR |= DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_0 \
                    | DMA_CCR_TEIE | DMA_CCR_CIRC; /* (5) */
DMA1_Channel7->CCR |= DMA_CCR_EN; /* (6) */

/* 为 DMA 配置 NVIC */
/* (1) 使能 DMA 通道 x 上的中断 */
/* (2) 为 DMA 通道 x 设置优先级 */
NVIC_EnableIRQ(DMA1_Channel4_5_6_7_IRQn); /* (1) */
NVIC_SetPriority(DMA1_Channel4_5_6_7_IRQn,3); /* (2) */
```

A.11.7 输出比较配置代码示例

```
/* (1) 将预分频比设置为 3，因此 APBCLK/4 即为 4 MHz */
/* (2) 设置 ARR = 4000 - 1 */
/* (3) 设置 CCRx = ARR，由于定时器时钟为 4 MHz，因此每 1 ms 发生一个事件 */
/* (4) 选择 OC1 的翻转模式 (OC1M = 011)，禁止 OC1 的预装载寄存器（OC1PE = 0，
       复位值） */
/* (5) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (6) 使能输出 (MOE = 1) */
/* (7) 使能计数器 */
TIMx->PSC |= 3; /* (1) */
TIMx->ARR = 4000 - 1; /* (2) */
TIMx->CCR1 = 4000 - 1; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_0 | TIM_CCMR1_OC1M_1; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->CR1 |= TIM_CR1_CEN; /* (6) */
```

A.11.8 边沿对齐 PWM 配置示例

```
/* (1) 将预分频比设置为 15，因此 APBCLK/16 即为 1 MHz */
/* (2) 设置 ARR = 8，由于定时器时钟为 1 MHz，因此周期为 9 µs */
/* (3) 设置 CCRx = 4，信号的高电平将持续 4 µs */
```

```
/* (4) 选择 OC1 的 PWM 模式 1 (OC1M = 110),
       使能 OC1 的预装载寄存器 (OC1PE = 1) */
/* (5) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (6) 使能输出 (MOE = 1) - 可选 */
/* (7) 使能计数器 (CEN = 1)
       选择边沿对齐模式（CMS = 00，复位值）
       将计数方向选为递增（DIR = 0，复位值） */
/* (8) 强制更新生成 (UG = 1) */
TIMx->PSC = 15; /* (1) */
TIMx->ARR = 8; /* (2) */
TIMx->CCR1 = 4; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1
             | TIM_CCMR1_OC1PE; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->CR1 |= TIM_CR1_CEN; /* (6) */
TIMx->EGR |= TIM_EGR_UG; /* (7) */
```

A.11.9 中心对齐 PWM 配置示例

```
/* (1) 将预分频比设置为 15，因此 APBCLK/16 即为 1 MHz */
/* (2) 设置 ARR = 8，由于定时器时钟为 1 MHz 并且采用中心对齐计数，
       因此周期为 16 µs */
/* (3) 设置 CCRx = 7，信号的高电平将持续 14 µs */
/* (4) 选择 OC1 的 PWM 模式 1 (OC1M = 110),
       使能 OC1 的预装载寄存器（OC1PE = 1，复位值） */
/* (5) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (6) 使能输出 (MOE = 1) */
/* (7) 使能计数器 (CEN = 1)
       选择中心对齐模式 1 (CMS = 01) */
/* (8) 强制更新生成 (UG = 1) */
TIMx->PSC = 15; /* (1) */
TIMx->ARR = 8; /* (2) */
TIMx->CCR1 = 7; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1
             | TIM_CCMR1_OC1PE; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->CR1 |= TIM_CR1_CMS_0 | TIM_CR1_CEN; /* (6) */
TIMx->EGR |= TIM_EGR_UG; /* (7) */
```

A.11.10 清零 OCxREF 的 ETR 配置代码示例

```
/* (1) 将预分频比设置为 15，因此 APBCLK/16 即为 1 MHz */
/* (2) 设置 ARR = 8，由于定时器时钟为 1 MHz，因此周期为 9 µs */
/* (3) 设置 CCRx = 4，信号的高电平将持续 4 µs */
/* (4) 选择 OC1 的 PWM 模式 1 (OC1M = 110),
```

```
        使能 OC1 的预装载寄存器 (OC1PE = 1)
        使能清零 OC1 以清零 ETR (OC1CE = 1)*/
/* (5) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
        使能 OC1 的输出 (CC1E = 1) */
/* (6) 选择 ETR 作为 OCREF 清零源（保留位 = 1）
        选择外部触发预分频器关闭（ETPS = 00，复位值）
        禁止外部时钟模式 2（ECE = 0，复位值）
        选择高电平有效（ETP = 0，复位值） */
/* (7) 使能计数器 (CEN = 1)
        选择边沿对齐模式（CMS = 00，复位值）
        将计数方向选为递增（DIR = 0，复位值） */
/* (8) 强制更新生成 (UG = 1) */

TIMx->PSC = 15; /* (1) */
TIMx->ARR = 8; /* (2) */
TIMx->CCR1 = 4; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1PE \
              | TIM_CCMR1_OC1CE; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->SMCR |= (1<<3); /* (6) */
TIMx->CR1 |= TIM_CR1_CEN; /* (7) */
TIMx->EGR |= TIM_EGR_UG; /* (8) */
```

A.11.11 编码器接口代码示例

```
/* (1) 配置 TI1 上的 TI1FP1 (CC1S = 01)
        配置 TI2 上的 TI1FP2 (CC2S = 01) */
/* (2) 将 TI1FP1 和 TI1FP2 配置为同相（CC1P = CC2P = 0，
        复位值） */
/* (3) 配置两个输入在上升沿和下降沿均有效
        (SMS = 011) */
/* (4) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0 | TIM_CCMR1_CC2S_0; /* (1) */
//TIMx->CCER &= (uint16_t)(~(TIM_CCER_CC21 | TIM_CCER_CC2P); /* (2) */
TIMx->SMCR |= TIM_SMCR_SMS_0 | TIM_SMCR_SMS_1; /* (3) */
TIMx->CR1 |= TIM_CR1_CEN; /* (4) */
```

A.11.12 复位模式代码示例

```
/* (1) 通过在 TIMx_CCMR1 寄存器中写入 CC1S =“01”
        将通道 1 配置为检测 TI1 输入的上升沿，
        并通过写入 TIMx_CCMR1 寄存器中的
        IC1F[3:0] 位
        配置输入滤波器的持续时间（如果无需滤波器，则保持
        IC1F=0000）。*/
```

```
/* (2) 通过在 TIMx_CCER 寄存器中写入 CC1P=0 来选择
       上升沿极性
       当它保持复位值时则不必要。*/
/* (3) 通过在 TIMx_SMCR 寄存器中写入 SMS=100，使定时器在复位模式下工作
       通过在 TIMx_SMCR 寄存器中写入 TS=101 来选择 TI1 作为
       触发输入源。*/
/* (4) 将预分频比设置为 16000-1 以便每 1 ms 递增一次 */
/* (5) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/ */
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0; /* (1) */
//TIMx->CCER &= (uint16_t)(~TIM_CCER_CC1P); /* (2) */
TIMx->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_TS_2 | TIM_SMCR_TS_0; /* (3) */
TIMx->PSC = 15999; /* (4) */
TIMx->CR1 |= TIM_CR1_CEN; /* (5) */
```

A.11.13 门控模式代码示例

```
/* (1) 通过在 TIMx_CCMR1 寄存器中写入 CC1S =“01”将通道 1 配置为
       检测 TI1 输入上的低电平，
       并通过写入 TIMx_CCMR1 寄存器中的
       IC1F[3:0]位
       配置输入滤波器的持续时间（如果无需滤波器，
       则保持 IC1F=0000）。*/
/* (2) 通过在 TIMx_CCER 寄存器中写入 CC1P=1 来选择极性 */
/* (3) 通过在 TIMx_SMCR 寄存器中写入 SMS=101，使定时器在门控模式下工作
       通过在 TIMx_SMCR 寄存器中写入 TS=101 来选择 TI1 作为
       触发输入源。*/
/* (4) 将预分频比设置为 4000-1 以便每 250 µs 递增一次 */
/* (5) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/ */
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0; /* (1) */
TIMx->CCER |= TIM_CCER_CC1P; /* (2) */
TIMx->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_0 \
            | TIM_SMCR_TS_2 | TIM_SMCR_TS_0; /* (3) */
TIMx->PSC = 3999; /* (4) */
TIMx->CR1 |= TIM_CR1_CEN; /* (5) */
```

A.11.14 触发模式代码示例

```
/* (1) 通过在 TIMx_CCMR1 寄存器中写入 CC2S =“01”将通道 2 配置为
       检测 TI2 输入上的上升沿，
       并通过写入 TIMx_CCMR1 寄存器中的
       IC1F[3:0] 位
       配置输入滤波器的持续时间（如果无需滤波器，
       则保持 IC1F=0000）。*/
/* (2) 通过在 TIMx_CCER 寄存器中写入
       CC2P=0（复位值）来选择极性 */
/* (3) 通过在 TIMx_SMCR 寄存器中写入 SMS=110，使定时器在触发模式下工作
```

```
        通过在 TIMx_SMCR 寄存器中写入 TS=110 来选择 TI2
        作为触发输入源。*/
/* (4) 将预分频比设置为 4000-1 以便每 250 µs 递增一次 */
TIMx->CCMR1 |= TIM_CCMR1_CC2S_0; /* (1) */
//TIMx->CCER &= ~TIM_CCER_CC2P; /* (2) */
TIMx->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1 \
            | TIM_SMCR_TS_2 | TIM_SMCR_TS_1; /* (3) */
TIMx->PSC = 3999; /* (4) */
```

A.11.15 外部时钟模式 2 + 触发模式代码示例

```
/* (1) 配置为无输入滤波器（ETF=0000，复位值）
       配置为禁止预分频器（ETPS = 0，复位值）
       选择 ETR 的上升沿检测（ETP = 0，复位值）
       使能外部时钟模式 2 (ECE = 1) */
/* (2) 配置为无输入滤波器（ETF=0000，复位值）
       选择 TI1 的输入捕捉源 (CC1S = 01) */
/* (3) 通过在 TIMx_CCER 寄存器中写入
       CC1P=0（复位值）来选择极性 */
/* (4) 通过在 TIMx_SMCR 寄存器中写入 SMS=110，使定时器在触发模式下工作
       通过在 TIMx_SMCR 寄存器中写入 TS=101 来选择 TI1
       作为触发输入源。*/
TIMx->SMCR |= TIM_SMCR_ECE; /* (1) */
TIMx->CCMR1 |= TIM_CCMR1_CC1S_0; /* (2) */
//TIMx->CCER &= ~TIM_CCER_CC1P; /* (3) */
TIMx->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1 \
            | TIM_SMCR_TS_2 | TIM_SMCR_TS_0; /* (4) */
```

A.11.16 单脉冲模式代码示例

```
/* 通过写入比较寄存器来定义 OPM 波形 */
/* (1) 将预分频比设置为 15，因此 APBCLK/16 即为 1 MHz */
/* (2) 设置 ARR = 7，由于定时器时钟为 1 MHz，因此周期为 8 µs */
/* (3) 设置 CCRx = 5，连续传送将延迟 5 µs（必须 > 0） */
/* (4) 选择 OC1 的 PWM 模式 2 (OC1M = 111)，
       使能 OC1 的预装载寄存器（OC1PE = 1，复位值）
       当 PULSE_WITHOUT_DELAY 置 1 时使能快速使能（无延时） */
/* (5) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (6) 使能输出 (MOE = 1) */
/* (7) 在 TIMx_CR1 寄存器中的 OPM 位写入 1
       以在下一个更新事件时
       停止计数器 (OPM = 1)
       使能自动重载寄存器 (ARPE = 1) */
TIMx->PSC = 15; /* (1) */
```

```
TIMx->ARR = 7; /* (2) */
TIMx->CCR1 = 5; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_0
              | TIM_CCMR1_OC1PE
#if PULSE_WITHOUT_DELAY > 0
              | TIM_CCMR1_OC1FE
#endif
; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->CR1 |= TIM_CR1_OPM | TIM_CR1_ARPE; /* (6) */
```

A.11.17 将一个定时器用作另一个定时器的预分频器的代码示例

```
/* (1) 通过在 TIMx_CR2 中写入 MMS = 010 将更新事件
       选作触发输出 (TRG0)。*/
/* (2) 通过在 TIMy_SMCR 中写入 TS = 000（复位值）使 TIMy 在从模式下工作，
       使用 ITR1 作为内部触发
       通过在 TIMy_SMCR 寄存器中写入 SMS=111，
       使 TIMy 在外部时钟模式 1 下工作。*/
/* (3) 将 TIMx 预分频比设置为 15999 以便每 1 ms 递增一次 */
/* (4) 将 TIMx 自动重载设置为 999 以便每秒溢出一次
       （从而出现一个 UEV） */
/* (5) 将 TIMx 自动重载设置为 24*3600-1 以便每 24 小时
       溢出一次 */
/* (6) 通过在 TIMx_CR1 寄存器中写入 CEN=1 来使能计数器。*/
/* (7) 通过在 TIMy_CR1 寄存器中写入 CEN=1 来使能计数器。*/
TIMx->CR2 |= TIM_CR2_MMS_1; /* (1) */
TIMy->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1 | TIM_SMCR_SMS_0; /* (2) */
TIMx->PSC = 15999; /* (3) */
TIMx->ARR = 999; /* (4) */
TIMy->ARR = (24 * 3600) - 1; /* (5) */
TIMx->CR1 |= TIM_CR1_CEN; /* (6) */
TIMy->CR1 |= TIM_CR1_CEN; /* (7) */
```

A.11.18 使用一个定时器使能另一个定时器的代码示例

```
/* (1) 将定时器 x 配置为主模式，发送其输出比较 1
       参考 (OC1REF)
       信号作为触发输出（TIMx_CR2 寄存器中的 MMS=100）。*/
/* (2) 配置定时器 x 的 OC1REF 波形（TIMx_CCMR1 寄存器）
       当计数器小于捕捉/比较寄存器时，
       通道 1 处于 PWM 模式 1
       （写入 OC1M = 110） */
/* (3) 通过在 TIMy_SMCR 中写入 TS = 000（复位值）使 TIMy 在从模式下工作，
       使用 ITR1 作为内部触发
       通过在 TIMy_SMCR 寄存器中写入 SMS=101，
```

```
        使 TIMy 在门控模式下工作。*/
/* (4) 将 TIMx 预分频比设置为 2 */
/* (5) 将 TIMy 预分频比设置为 2 */
/* (6) 将 TIMx 自动重载设置为 999 以便每 100 ms 溢出一次
        （从而出现一个 UEV） */
/* (7) 将捕捉比较寄存器设置为 0 到 999 间的一个值 */
TIMx->CR2 |= TIM_CR2_MMS_2; /* (1) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; /* (2) */
TIMy->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_0; /* (3) */
TIMx->PSC = 2; /* (4) */
TIMy->PSC = 2; /* (5) */
TIMx->ARR = 999; /* (6) */
TIMx-> CCR1 = 700; /* (7) */

/* 配置从定时器以在每次计数时产生翻转 */
/* (1) 将定时器 2 配置为 PWM 模式 1（写入 OC1M = 110） */
/* (2) 将 TIMx 自动重载设置为 1 */
/* (3) 将捕捉比较寄存器设置为 1 */
TIMy->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; /* (1) */
TIMy->ARR = 1; /* (2) */
TIMy-> CCR1 = 1; /* (3) */
/* 使能 TIMx OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
        使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMx->CCER |= TIM_CCER_CC1E;

/* 使能 TIMy OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
        使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMy->CCER |= TIM_CCER_CC1E;

/* (1) 首先通过在 TIMy_CR1 寄存器中写入 CEN=1
        来使能从计数器。*/
/* (2) 通过在 TIMx_CR1 寄存器中写入 CEN=1
        来使能主计数器。*/
TIMy->CR1 |= TIM_CR1_CEN; /* (1) */
TIMx->CR1 |= TIM_CR1_CEN; /* (2) */
```

A.11.19 主器件和从器件同步代码示例

```
/* (1) 将定时器 x 配置为主模式，发送其使能信号
        作为触发输出（TIMx_CR2 寄存器中的 MMS=001）。*/
/* (2) 将定时器 x 通道 1 的波形（TIMx_CCMR1 寄存器）
        配置为 PWM 模式 1（写入 OC1M = 110） */
```

```
/* (3) 通过在 TIMy_SMCR 中写入 TS = 000（复位值）使 TIMy 在从模式下工作，
       使用 ITR1 作为内部触发
       通过在 TIMy_SMCR 寄存器中写入 SMS=101，
       使 TIMy 在门控模式下工作。*/
/* (4) 将 TIMx 预分频比设置为 2 */
/* (5) 将 TIMy 预分频比设置为 2 */
/* (6) 将 TIMx 自动重载设置为 99 以便每 10 ms 溢出一次
       （从而出现一个 UEV） */
/* (7) 将捕捉比较寄存器设置为 0 到 99 间的一个值 */
TIMx->CR2 |= TIM_CR2_MMS_0; /* (1) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; /* (2) */
TIMy->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_0; /* (3) */
TIMx->PSC = 2; /* (4) */
TIMy->PSC = 2; /* (5) */
TIMx->ARR = 99; /* (6) */
TIMx-> CCR1 = 25; /* (7) */

/* 将从定时器通道 1 与定时器一样配置为 PWM 模式
   以实现同步 */
/* (1) 使定时器 y 在 PWM 模式 1 下工作（写入 OC1M = 110） */
/* (2) 将 TIMx 自动重载设置为 99 */
/* (3) 将捕捉比较寄存器设置为 25 */
TIMy->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; /* (1) */
TIMy->ARR = 99; /* (2) */
TIMy-> CCR1 = 25; /* (3) */

/* 使能 TIMx OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMx->CCER |= TIM_CCER_CC1E;
/* 使能 TIMy OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMy->CCER |= TIM_CCER_CC1E;
/* (1) 通过向 UG 位（TIMx_EGR 寄存器）写入“1”复位定时器 x */
/* (2) 通过向 UG 位（TIMy_EGR 寄存器）写入“1”复位定时器 y */
TIMx->EGR |= TIM_EGR_UG;
TIMy->EGR |= TIM_EGR_UG;
/* (1) 首先通过在 TIMy_CR1 寄存器中写入 CEN=1
       来使能从定时器。
       TIMy 将与主定时器同步启动*/
/* (2) 通过在 TIMx_CR1 寄存器中写入 CEN=1
       来启动主计数器。*/
```

```
TIMy->CR1 |= TIM_CR1_CEN; /* (1) */
TIMx->CR1 |= TIM_CR1_CEN; /* (2) */
```

A.11.20 两个定时器通过一个外部触发信号同步的代码示例

```
/* (1) 将 TIMx 配置为主模式，发送其使能信号
       作为触发输出（TIMx_CR2 寄存器中的 MMS=001）。*/
/* (2) 将 TIMx 配置为从模式，通过在 TIMx_SMCR 中写入 TS = 100
       从 TI1 获得输入触发信号
       通过在 TIMx_SMCR 寄存器中写入 SMS=110
       将 TIMx 配置为触发模式。
       通过在 TIMx_SMCR 中写入 MSM=1 将 TIMx
       配置为主/从模式 */
/* (3) 将 TIMy 配置为从模式，通过在 TIMy_SMCR 中写入 TS = 000（复位值）
       以从定时器 1 获得输入触发信号
       通过在 TIMy_SMCR 寄存器中写入 SMS=110
       将 TIMy 配置为触发模式。*/
/* (4) 通过向 UG 位（TIMx_EGR 寄存器）写入“1”复位定时器 x 计数器 */
/* (5) 通过向 UG 位（TIMy_EGR 寄存器）写入“1”复位定时器 y 计数器 */
TIMx->CR2 |= TIM_CR2_MMS_0; /* (1) */
TIMx->SMCR |= TIM_SMCR_TS_2 | TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1
            | TIM_SMCR_MSM; /* (2) */
TIMy->SMCR |= TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1; /* (3) */
TIMx->EGR |= TIM_EGR_UG; /* (4) */
TIMy->EGR |= TIM_EGR_UG; /* (5) */

/* 将定时器通道 2 与 PWM 一样配置为 PWM 模式 */
/* (1) 将定时器 1 通道 2 的波形（TIM1_CCMR1 寄存器）
       配置为 PWM 模式 1（写入 OC2M = 110） */
/* (2) 将 TIMx 预分频比设置为 2 */
/* (3) 将 TIMx 自动重载设置为 99 以便每 10 ms 溢出一次
       （从而出现一个 UEV） */
/* (4) 将捕捉比较寄存器设置为 0 到 99 间的一个值 */
TIMx->CCMR1 |= TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1; /* (1) */
TIMx->PSC = 2; /* (2) */
TIMx->ARR = 99; /* (3) */
TIMx-> CCR2 = 25; /* (4) */

/* 将从定时器通道 1 与定时器一样配置为 PWM 模式以实现
   同步 */
/* (1) 将定时器 2 配置为 PWM 模式 1（写入 OC1M = 110） */
/* (2) 将 TIMy 预分频比设置为 2 */
/* (3) 将 TIMx 自动重载设置为 99 */
/* (4) 将捕捉比较寄存器设置为 25 */
TIMy->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; /* (1) */
TIMy->PSC = 2; /* (2) */
```

```
TIMy->ARR = 99; /* (2) */
TIMy-> CCR1 = 25; /* (3) */

/* 使能 TIMx OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMx->CCER |= TIM_CCER_CC2E;

/* 使能 TIMy OC1 的输出 */
/* (1) 选择 OC1 的高电平有效极性（CC1P = 0，复位值），
       使能 OC1 的输出 (CC1E = 1) */
/* (2) 使能输出 (MOE = 1) */
TIMy->CCER |= TIM_CCER_CC1E;
```

A.11.21 DMA 连续传送功能代码示例

```
/* 配置 DMA 连续传送功能 */
/* 配置相应的 DMA 通道 */
/* (1) 使能定时器 x 和 DMA 的外设时钟 */
/* (2) 通过在 DMA_CSELR_C2S 中写入 1000 将 DMA 通道 2 重映射到 TIM2_UP */
/* (3) 将 DMA 通道外设地址设置为 DMAR 寄存器地址 */
/* (4) 将 DMA 通道存储器地址设置为
       包含要通过 DMA 传输到 CCRx 的数据的 RAM 中的
       缓冲区地址 */
/* (5) 将数据传输次数设置为 sizeof(Duty_Cycle_Table) */
/* (6) 在 CCR 寄存器中配置 DMA 传输
       通过将 CIRC 位置 1 来使能循环模式（可选）
       将存储器大小设置为 16_bits MSIZE = 01
       将外设大小设置为 32_bits PSIZE = 10
       通过将 MINC 置 1 来使能存储器递增模式
       通过将 DIR 置 1 来设置从存储器读取的数据传输方向 */
/* (7) 将 TIMx_DCR 寄存器配置为 DBL = 3 次传输
       以及 DBA = (@TIMx->CCR2 - @TIMx->CR1) >> 2 = 0xE */
/* (8) 通过将 DIER 寄存器中的 UDE 位置 1 来使能 TIMx 更新
       DMA 请求 */
/* (9) 使能 TIMx */
/* (10)使能 DMA 通道 */
RCC->AHBENR |= RCC_AHBENR_DMA1EN; /* (1) */
DMA1_CSELR->CSELR |= 8 << (4 * (2-1)); /* (2) */
DMA1_Channel2->CPAR = (uint32_t)(&(TIMx->DMAR)); /* (3) */
DMA1_Channel2->CMAR = (uint32_t)(Duty_Cycle_Table); /* (4) */
DMA1_Channel2->CNDTR = 10*3; /* (5) */
DMA1_Channel2->CCR |= DMA_CCR_CIRC | DMA_CCR_MSIZE_0 | DMA_CCR_PSIZE_1
                    | DMA_CCR_MINC | DMA_CCR_DIR; /* (6) */
TIMx->DCR = (3 << 8)
```

```
            + ((((uint32_t)(&TIM2->CCR2)) - ((uint32_t)(&TIM2->CR1))) >> 2)
            ; /* (7) */
TIMx->DIER |= TIM_DIER_UDE; /* (8) */
TIMx->CR1 |= TIM_CR1_CEN; /* (9) */
DMA1_Channel2->CCR |= DMA_CCR_EN; /* (10) */
```

A.12 低功耗定时器 (LPTIM)

A.12.1 脉冲计数器配置代码示例

```
/* (1) 将 LPTimer 配置为外部输入 1 上的计数器 */
/* (2) 使能自动重载匹配中断 */
/* (3) 使能 LPTimer */
/* (4) 将自动重载设置为 4 以便在 10 个脉冲后中断一次，
       因为前 5 个脉冲不会使计数器递增 */
LPTIM1->CFGR |= LPTIM_CFGR_COUNTMODE | LPTIM_CFGR_CKSEL; /* (1) */
LPTIM1->IER |= LPTIM_IER_ARRMIE; /* (2) */
LPTIM1->CR |= LPTIM_CR_ENABLE; /* (3) */
LPTIM1->ARR = 4; /* (4) */
LPTIM1->CR |= LPTIM_CR_CNTSTRT; /* 以连续模式启动计数器 */
```

A.13 IWDG 代码示例

A.13.1 IWDG 配置代码示例

```
/* (1) 激活 IWDG（如果在选项字节中完成则不需要） */
/* (2) 使能对 IWDG 寄存器的写访问 */
/* (3) 将预分频比设置为 8 */
/* (4) 设置重载值以便每 100 ms 翻转一次 */
/* (5) 检查标志是否复位 */
/* (6) 刷新计数器 */
IWDG->KR = IWDG_START; /* (1) */
IWDG->KR = IWDG_WRITE_ACCESS; /* (2) */
IWDG->PR = IWDG_PR_PR_0; /* (3) */
IWDG->RLR = IWDG_RELOAD; /* (4) */
while(IWDG->SR) /* (5) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
IWDG->KR = IWDG_REFRESH; /* (6) */
```

A.13.2 IWDG 配置（带窗口）代码示例

```
/* (1) 激活 IWDG（如果在选项字节中完成则不需要） */
/* (2) 使能对 IWDG 寄存器的写访问 */
```

```
/* (3) 将预分频比设置为 8 */
/* (4) 设置重载值以便每 100 ms 翻转一次 */
/* (5) 检查标志是否复位 */
/* (6) 设置一个 50 ms 的窗口，这将刷新 IWDG */
IWDG->KR = IWDG_START; /* (1) */
IWDG->KR = IWDG_WRITE_ACCESS; /* (2) */
IWDG->PR = IWDG_PR_PR_0; /* (3) */
IWDG->RLR = IWDG_RELOAD; /* (4) */
while(IWDG->SR) /* (5) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
IWDG->WINR = IWDG_RELOAD >> 1; /* (6) */
```

A.14 WWDG 代码示例

A.14.1 WWDG 配置代码示例

```
/* (1) 设置预分频比以便每 16.5 ms 左右翻转一次，
       设置窗口值（约 7.5 ms） */
/* (2) 先刷新 WWDG 再将其激活 */
/* (3) 激活 WWDG */
WWDG->CFR = 0x0060; /* (1) */
WWDG->CR = WWDG_REFRESH; /* (2) */
WWDG->CR |= WWDG_CR_WDGA; /* (3) */
```

A.15 RTC 代码示例

A.15.1 RTC 日历配置代码示例

```
/* (1) 对 RTC 寄存器执行写访问 */
/* (2) 使能初始化阶段 */
/* (3) 等待至可修改 RTC 寄存器值 */
/* (4) 设置预分频比，40 kHz/64 => 625 Hz，625 Hz/625 => 1 Hz */
/* (5) 在 TR 中设置新时间 */
/* (6) 禁止初始化阶段 */
/* (7) 禁止对 RTC 寄存器的写访问 */
RTC->WPR = 0xCA; /* (1) */
RTC->WPR = 0x53; /* (1) */
RTC->ISR = RTC_ISR_INIT; /* (2) */
while((RTC->ISR & RTC_ISR_INITF)!=RTC_ISR_INITF) /* (3) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->PRER = 0x003F0270; /* (4) */
```

```
RTC->TR = RTC_TR_PM | Time; /* (5) */
RTC->ISR =~ RTC_ISR_INIT; /* (6) */
RTC->WPR = 0xFE; /* (7) */
RTC->WPR = 0x64; /* (7) */
```

A.15.2 RTC 闹钟配置代码示例

```
/* (1) 对 RTC 寄存器执行写访问 */
/* (2) 禁止闹钟 A 以对其进行修改 */
/* (3) 等待至可修改闹钟 A 的值 */
/* (4) 修改闹钟 A 掩码以便每 1 Hz 中断一次 */
/* (5) 使能闹钟 A 及闹钟 A 中断 */
/* (6) 禁止写访问 */
RTC->WPR = 0xCA; /* (1) */
RTC->WPR = 0x53; /* (1) */
RTC->CR &=~ RTC_CR_ALRAE; /* (2) */
while((RTC->ISR & RTC_ISR_ALRAWF) != RTC_ISR_ALRAWF) /* (3) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->ALRMAR = RTC_ALRMAR_MSK4 | RTC_ALRMAR_MSK3 | RTC_ALRMAR_MSK2 |
RTC_ALRMAR_MSK1; /* (4) */
RTC->CR = RTC_CR_ALRAIE | RTC_CR_ALRAE; /* (5) */
RTC->WPR = 0xFE; /* (6) */
RTC->WPR = 0x64; /* (6) */
```

A.15.3 RTC WUT 配置代码示例

```
/* (1) 对 RTC 寄存器执行写访问 */
/* (2) 禁止唤醒定时器以对其进行修改 */
/* (3) 等待至可修改唤醒重载值 */
/* (4) 修改唤醒值重载计数器以便每 1 Hz 唤醒一次 */
/* (5) 使能唤醒计数器和唤醒中断 */
/* (6) 禁止写访问 */
RTC->WPR = 0xCA; /* (1) */
RTC->WPR = 0x53; /* (1) */
RTC->CR &=~ RTC_CR_WUTE; /* (2) */
while((RTC->ISR & RTC_ISR_WUTWF) != RTC_ISR_WUTWF) /* (3) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->WUTR = 0x9C0; /* (4) */
RTC->CR = RTC_CR_WUTE | RTC_CR_WUTIE; /* (5) */
RTC->WPR = 0xFE; /* (6) */
RTC->WPR = 0x64; /* (6) */
```

A.15.4 RTC 读日历代码示例

```
if((RTC->ISR & RTC_ISR_RSF) == RTC_ISR_RSF)
{
  TimeToCompute = RTC->TR; /* 获取时间 */
  DateToCompute = RTC->DR; /* 还需要读取日期 */
}
```

A.15.5 RTC 校准代码示例

```
/* (1) 对 RTC 寄存器执行写访问 */
/* (2) 使能初始化阶段 */
/* (3) 等待至可修改 RTC 寄存器值 */
/* (4) 设置预分频比，40 kHz/125 => 320 Hz，320 Hz/320 => 1 Hz */
/* (5) 在 TR 中设置新时间 */
/* (6) 禁止初始化阶段 */
/* (7) 等待至可修改校准寄存器 */
/* (8) 将校准设置为 +20 ppm 左右（25°C 时的标准值） */
/* 注：当 LSE 被选用于 RTC 时钟时，涉及校准 */
/* (9) 禁止对 RTC 寄存器的写访问 */
RTC->WPR = 0xCA; /* (1) */
RTC->WPR = 0x53; /* (1) */
RTC->ISR = RTC_ISR_INIT; /* (2) */
while((RTC->ISR & RTC_ISR_INITF)!=RTC_ISR_INITF) /* (3) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->PRER = (124<<16) | 319; /* (4) */
RTC->TR = RTC_TR_PM | Time; /* (5) */
RTC->ISR &=~ RTC_ISR_INIT; /* (6) */
while((RTC->ISR & RTC_ISR_RECALPF) == RTC_ISR_RECALPF) /* (7) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->CALR = RTC_CALR_CALP | 482; /* (8) */
RTC->WPR = 0xFE; /* (9) */
RTC->WPR = 0x64; /* (9) */
```

A.15.6 RTC 入侵和时间戳配置代码示例

```
/* 入侵配置：
   - 禁止预充电 (PU)
   - RTCCLK/256 入侵采样频率
   - 入侵检测时激活时间戳
   - RTC_TAMP2 (PA0) 的输入上升沿触发检测
   - 入侵中断使能 */
RTC->TAFCR = RTC_TAFCR_TAMPPUDIS | RTC_TAFCR_TAMPFREQ | RTC_TAFCR_TAMPTS
```

```
                | RTC_TAFCR_TAMP2E | RTC_TAFCR_TAMPIE;
```

A.15.7 RTC 入侵和时间戳代码示例

```
/* 检查入侵和时间戳标志 */
if(((RTC->ISR & (RTC_ISR_TAMP2F)) == (RTC_ISR_TAMP2F)) && ((RTC->ISR &
                                    (RTC_ISR_TSF)) == (RTC_ISR_TSF)))
{
  RTC->ISR =~ (RTC_ISR_TAMP2F); /* 清零入侵标志 */
  EXTI->PR = EXTI_PR_PR19; /* 清零 EXTI 线 19 的标志 */
  TimeToCompute = RTC->TSTR; /* 获取时间戳寄存器中的入侵时间 */
  RTC->ISR =~ (RTC_ISR_TSF); /* 清零时间戳标志 */
}
```

A.15.8 RTC 时钟输出代码示例

```
/* (1) 对 RTC 寄存器执行写访问 */
/* (2) 禁止闹钟 A 以对其进行修改 */
/* (3) 等待至可修改闹钟 A 的值 */
/* (4) 修改闹钟 A 掩码以便每 1 Hz 中断一次 */
/* (5) 使能闹钟 A 及闹钟 A 中断，校准输出 (1 Hz)
       使能 */
/* (6) 禁止写访问 */
RTC->WPR = 0xCA; /* (1) */
RTC->WPR = 0x53; /* (1) */
RTC->CR &=~ RTC_CR_ALRAE; /* (2) */
while((RTC->ISR & RTC_ISR_ALRAWF) != RTC_ISR_ALRAWF) /* (3) */
{
  /* 为实现稳定应用，在此处添加超时 */
}
RTC->ALRMAR = RTC_ALRMAR_MSK4 | RTC_ALRMAR_MSK3 | RTC_ALRMAR_MSK2 |
RTC_ALRMAR_MSK1; /* (4) */
RTC->CR = RTC_CR_ALRAIE | RTC_CR_ALRAE | RTC_CR_COE | RTC_CR_COSEL; /*(5) */
RTC->WPR = 0xFE; /* (6) */
RTC->WPR = 0x64; /* (6) */
```

A.16 I2C 代码示例

A.16.1 将 I2C 配置为从模式的代码示例

```
/* (1) 使用 AN4235 xls 文件计算时序寄存器值，
       快速模式 (400 kHz)，I2CCLK = 16 MHz，上升时间 = 100 ns，
       下降时间 = 10 ns */
/* (2) 外设使能，地址匹配中断使能 */
/* (3) 7 位地址 = 0x5A */
/* (4) 使能自身地址 1 */
```

```
I2C1->TIMINGR = (uint32_t)0x00300619; /* (1) */
I2C1->CR1 = I2C_CR1_PE | I2C_CR1_ADDRIE; /* (2) */
I2C1->OAR1 |= (uint32_t)(I2C1_OWN_ADDRESS << 1); /* (3) */
I2C1->OAR1 |= I2C_OAR1_OA1EN; /* (4) */
```

A.16.2 I2C 从发送器代码示例

```
uint32_t I2C_InterruptStatus = I2C1->ISR; /* 获取中断状态 */
/* 检查地址匹配 */
if((I2C_InterruptStatus & I2C_ISR_ADDR) == I2C_ISR_ADDR)
{
  I2C1->ICR |= I2C_ICR_ADDRCF; /* 清零地址匹配标志 */
  /* 检查是否读取了传输方向（从发送器） */
  if((I2C1->ISR & I2C_ISR_DIR) == I2C_ISR_DIR)
  {
    I2C1->CR1 |= I2C_CR1_TXIE; /* 设置发送 IT */
  }
}
else if((I2C_InterruptStatus & I2C_ISR_TXIS) == I2C_ISR_TXIS)
{
  I2C1->CR1 &=~ I2C_CR1_TXIE; /* 禁止发送 IT */
  I2C1->TXDR = I2C_BYTE_TO_SEND; /* 要发送的字节 */
}
```

A.16.3 I2C 从接收器代码示例

```
uint32_t I2C_InterruptStatus = I2C1->ISR; /* 获取中断状态 */
if((I2C_InterruptStatus & I2C_ISR_ADDR) == I2C_ISR_ADDR)
{
  I2C1->ICR |= I2C_ICR_ADDRCF; /* 地址匹配事件 */
}
else if((I2C_InterruptStatus & I2C_ISR_RXNE) == I2C_ISR_RXNE)
{
  /* 读取接收寄存器，将清零 RXNE 标志 */
  if(I2C1->RXDR == I2C_BYTE_TO_SEND)
  {
    /* 过程 */
  }
}
```

A.16.4 将 I2C 配置为主模式以进行接收的代码示例

```
/* (1) 使用 AN4235 xls 文件计算时序寄存器值，
       快速模式 (400 kHz)，I2CCLK = 16 MHz，上升时间 = 100 ns，
       下降时间 = 10 ns */
/* (2) 外设使能，接收中断使能 */
/* (3) 从地址 = 0x5A，读传输，要接收 1 个字节，自动结束 */
```

```
I2C2->TIMINGR = (uint32_t)0x00300619; /* (1) */
I2C2->CR1 = I2C_CR1_PE | I2C_CR1_RXIE; /* (2) */
I2C2->CR2 = I2C_CR2_AUTOEND | (1<<16) | I2C_CR2_RD_WRN |
            (I2C1_OWN_ADDRESS<<1); /* (3) */
```

A.16.5 将 I2C 配置为主模式以进行发送的代码示例

```
/* (1) 使用 AN4235 xls 文件计算时序寄存器值，
       快速模式 (400 kHz)，I2CCLK = 16 MHz，上升时间 = 100 ns，
       下降时间 = 10 ns */
/* (2) 外设使能 */
/* (3) 从地址 = 0x5A，写传输，要发送 1 个字节，自动结束 */
I2C2->TIMINGR = (uint32_t)0x00300619; /* (1) */
I2C2->CR1 = I2C_CR1_PE; /* (2) */
I2C2->CR2 = I2C_CR2_AUTOEND | (1<<16) | (I2C1_OWN_ADDRESS<<1); /* (3) */
```

A.16.6 I2C 主发送器代码示例

```
/* 检查 Tx 是否为空 */
if((I2C2->ISR & I2C_ISR_TXE) == (I2C_ISR_TXE))
{
  I2C2->TXDR = I2C_BYTE_TO_SEND; /* 要发送的字节 */
  I2C2->CR2 |= I2C_CR2_START; /* 启动 */
}
```

A.16.7 I2C 主接收器代码示例

```
if((I2C2->ISR & I2C_ISR_RXNE) == I2C_ISR_RXNE)
{
  /* 读取接收寄存器，将清零 RXNE 标志 */
  if(I2C2->RXDR == I2C_BYTE_TO_SEND)
  {
    /* 过程 */
  }
}
```

A.16.8 将 I2C 配置为主模式以通过 DMA 进行发送的代码示例

```
/* (1) 使用 AN4235 xls 文件计算时序寄存器值，
       快速模式 (400 kHz)，I2CCLK = 16 MHz，上升时间 = 100 ns，
       下降时间 = 10 ns */
/* (2) 外设使能 */
/* (3) 从地址 = 0x5A，写传输，要发送 2 个字节，自动结束 */
I2C2->TIMINGR = (uint32_t)0x00300619; /* (1) */
I2C2->CR1 = I2C_CR1_PE | I2C_CR1_TXDMAEN; /* (2) */
I2C2->CR2 = I2C_CR2_AUTOEND | (SIZE_OF_DATA << 16) |
            (I2C1_OWN_ADDRESS<<1); /* (3) */
```

A.16.9 将 I2C 配置为从模式以通过 DMA 进行接收的代码示例

```
/* (1) 使用 AN4235 xls 文件计算时序寄存器值，
       快速模式 (400 kHz)，I2CCLK = 16 MHz，上升时间 = 100 ns，
       下降时间 = 10 ns */
/* (2) 外设使能，接收 DMA 使能 */
/* (3) 7 位地址 = 0x5A */
/* (4) 使能自身地址 1 */
I2C1->TIMINGR = (uint32_t)0x00300619; /* (1) */
I2C1->CR1 = I2C_CR1_PE | I2C_CR1_RXDMAEN | I2C_CR1_ADDRIE; /* (2) */
I2C1->OAR1 |= (uint32_t)(I2C1_OWN_ADDRESS << 1); /* (3) */
I2C1->OAR1 |= I2C_OAR1_OA1EN; /* (4) */
```

A.17 USART 代码示例

A.17.1 USART 发送器配置代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR1 = USART_CR1_TE | USART_CR1_UE; /* (2) */
```

A.17.2 USART 发送字节代码示例

```
/* 启动 USART 发送 */
USART1->TDR = stringtosend[send++]; /* 将在 TXE 时发起 TC */
```

A.17.3 USART 传输完成代码示例

```
if((USART1->ISR & USART_ISR_TC) == USART_ISR_TC)
{
  if(send == sizeof(stringtosend))
  {
    send=0;
    USART1->ICR = USART_ICR_TCCF; /* 清零传输完成标志 */
  }
  else
  {
    /* 清零传输完成标志并用新字符填充 TDR */
    USART1->TDR = stringtosend[send++];
  }
}
```

A.17.4 USART 接收器配置代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验，接收模式 */
```

```
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR1 = USART_CR1_RXNEIE | USART_CR1_RE | USART_CR1_UE; /* (2) */
```

A.17.5 USART 接收字节代码示例

```
if((USART1->ISR & USART_ISR_RXNE) == USART_ISR_RXNE)
{
  chartoreceive = (uint8_t)(USART1->RDR); /* 接收数据，清零标志 */
}
```

A.17.6 USART LIN 模式代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) LIN 模式 */
/* (3) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验，
       使能接收和发送 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR2 = USART_CR2_LINEN | USART_CR2_LBDIE; /* (2) */
USART1->CR1 = USART_CR1_TE | USART_CR1_RXNEIE | USART_CR1_RE |
              USART_CR1_UE; /* (3) */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)/* 轮询空闲帧发送 */
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.17.7 USART 同步模式代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 同步模式 */
/* CPOL 和 CPHA = 0 => 第一个上升沿 */
/* 最后一个位时钟脉冲 */
/* 采用最高有效位在前的方式进行发送/接收 */
/* (3) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验 */
/* 使能发送，使能接收 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR2 = USART_CR2_MSBFIRST | USART_CR2_CLKEN | USART_CR2_LBCL; /* (2)
*/
USART1->CR1 = USART_CR1_TE | USART_CR1_RXNEIE | USART_CR1_RE |
USART_CR1_UE; /* (3) */
/* 轮询空闲帧发送（无时钟） */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
```

```
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.17.8 USART 单线半双工代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 单线半双工模式 */
/* (3) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验，
       使能接收和发送 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR3 = USART_CR3_HDSEL; /* (2) */
USART1->CR1 = USART_CR1_TE | USART_CR1_RXNEIE | USART_CR1_RE |
              USART_CR1_UE; /* (3) */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)/* 轮询空闲帧发送 */
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.17.9 USART 智能卡模式代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 经 16 分频后的时钟 = 1MHz */
/* (3) 智能卡模式使能 */
/* (4) 1.5 个停止位，时钟使能 */
/* (5) 8 个数据位加奇偶校验，1 个起始位 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->GTPR = 16 >> 1; /* (2) */
USART1->CR3 = USART_CR3_SCEN; /* (3) */
USART1->CR2 = USART_CR2_STOP_1 | USART_CR2_STOP_0 | USART_CR2_CLKEN;/*(4 */
USART1->CR1 = USART_CR1_M | USART_CR1_PCE | USART_CR1_TE |
USART_CR1_UE;/* (5) */
/* 轮询空闲帧发送传输完成（不发送此帧） */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.17.10 USART IrDA 模式代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 进行 24 分频以获得低功耗频率 */
/* (3) 使能 IrDA */
/* (4) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验 */
```

```
USART1->BRR = 160000 / 96; /* (1) */
USART1->GTPR = 24; /* (2) */
USART1->CR3 = USART_CR3_IREN; /* (3) */
USART1->CR1 = USART_CR1_TE | USART_CR1_UE; /* (4) */
/* 轮询空闲帧发送 */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.17.11 USART DMA 代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 在接收和发送时使能 DMA */
/* (3) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验，
       使能接收和发送 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR3 = USART_CR3_DMAT | USART_CR3_DMAR; /* (2) */
USART1->CR1 = USART_CR1_TE | USART_CR1_RE | USART_CR1_UE; /* (3) */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)/* 轮询空闲帧发送 */
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
```

A.17.12 USART 硬件流控制代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 使能 RTS 和 CTS */
/* (3) 8 个数据位，1 个起始位，1 个停止位，无奇偶校验，
       使能接收和发送 */
USART1->BRR = 160000 / 96; /* (1) */
USART1->CR3 = USART_CR3_RTSE | USART_CR3_CTSE; /* (2) */
USART1->CR1 = USART_CR1_TE | USART_CR1_RXNEIE | USART_CR1_RE |
USART_CR1_UE; /* (3) */
while((USART1->ISR & USART_ISR_TC) != USART_ISR_TC)/* 轮询空闲帧发送 */
{
  /* 为实现稳定应用，在此处添加超时 */
}
USART1->ICR = USART_ICR_TCCF;/* 清零 TC 标志 */
USART1->CR1 |= USART_CR1_TCIE;/* 使能 TC 中断 */
```

A.18 LPUART 代码示例

A.18.1 LPUART 接收器配置代码示例

```
/* (1) 16 倍过采样，9600 波特 */
/* (2) 使能 STOP 模式，8 个数据位，1 个起始位，1 个停止位，无奇偶校验，
       接收模式 */
LPUART1->BRR = 0x369; /* (1) */
LPUART1->CR1 = USART_CR1_UESM | USART_CR1_RXNEIE | USART_CR1_RE |
               USART_CR1_UE; /* (2) */
```

A.18.2 LPUART 接收字节代码示例

```
if((LPUART1->ISR & USART_ISR_RXNE) == USART_ISR_RXNE)
{
  chartoreceive = (uint8_t)(LPUART1->RDR); /* 接收数据，清零标志 */
}
```

A.19 SPI 代码示例

A.19.1 SPI 主器件配置代码示例

```
/* (1) 主器件选择，BR: Fpclk/256，
       CPOL 和 CPHA 为零（第一个上升沿） */
/* (2) 使能从器件选择输出，RXNE IT，8 位 Rx FIFO */
/* (3) 使能 SPI1 */
SPI1->CR1 = SPI_CR1_MSTR | SPI_CR1_BR; /* (1) */
SPI1->CR2 = SPI_CR2_SSOE | SPI_CR2_RXNEIE; /* (2) */
SPI1->CR1 |= SPI_CR1_SPE; /* (3) */
```

A.19.2 SPI 从器件配置代码示例

```
/* nSS 硬性，从器件，CPOL 和 CPHA 为零（第一个上升沿） */
/* (1) RXNE IT，8 位 Rx FIFO */
/* (2) 使能 SPI2 */
SPI2->CR2 = SPI_CR2_RXNEIE; /* (1) */
SPI2->CR1 |= SPI_CR1_SPE; /* (2) */
```

A.19.3 SPI 全双工通信代码示例

```
if((SPI1->SR & SPI_SR_TXE) == SPI_SR_TXE) /* 测试 Tx 是否为空 */
{
  /* 将在 TXE 时发起 8 位发送 */
  *(uint8_t *)&(SPI1->DR) = SPI1_DATA;
}
```

A.19.4 SPI 主器件配置（采用 DMA）代码示例

```
/* (1) 主器件选择，BR: Fpclk/256，
       CPOL 和 CPHA 为零（第一个上升沿） */
/* (2) TX 和 RX（采用 DMA），使能从器件选择输出，RXNE IT，
       8 位 Rx FIFO */
/* (3) 使能 SPI1 */
SPI1->CR1 = SPI_CR1_MSTR | SPI_CR1_BR; /* (1) */
SPI1->CR2 = SPI_CR2_TXDMAEN | SPI_CR2_RXDMAEN | SPI_CR2_SSOE;; /* (2) */
SPI1->CR1 |= SPI_CR1_SPE; /* (3) */
```

A.19.5 SPI 从器件配置（采用 DMA）代码示例

```
/* nSS 硬性，从器件，CPOL 和 CPHA 为零（第一个上升沿） */
/* (1) TX 和 RX（采用 DMA），RXNE IT，8 位 Rx FIFO */
/* (2) 使能 SPI2 */
SPI2->CR2 = SPI_CR2_TXDMAEN | SPI_CR2_RXDMAEN; /* (1) */
SPI2->CR1 |= SPI_CR1_SPE; /* (2) */
```

A.19.6 SPI 中断代码示例

```
if((SPI1->SR & SPI_SR_RXNE) == SPI_SR_RXNE)
{
  SPI1_Data = (uint8_t)SPI1->DR; /* 接收数据，清零标志 */
  /* 过程 */
}
```

A.20 DBG 代码示例

A.20.1 DBG 读取器件 Id 代码示例

```
MCU_Id = DBGMCU->IDCODE; /* 读取 MCU Id，32 位访问 */
```

A.20.2 LPM 下的 DBG 调试代码示例

```
DBGMCU->CR |= DBGMCU_CR_DBG_STOP; /* 为了能够在停止模式下调试 */
```

35 版本历史

表 163. 文档版本历史

日期	版本	变更
2014 年 2 月 11 日	1	初始版本。
2014 年 4 月 29 日	2	重新编排了 TSC、RNG、ASE 和通信部分。 **系统和存储器概述** 更新了*第 2.3 节：嵌入式 SRAM*和*第 2.4 节：自举配置*。 **闪存/数据 EEPROM** 更新了*图 6：RDP 级别*。 修改了*第 3.4.2 节：PcROP（专有代码读取保护）*。 **防火墙** 将 EEPROM PROG、PROG EEPROM、PROG 或 PROGRAM MEMORY 重命名为 Flash 程序存储器。 **PWR** 在*图 10：电源概述*下增加了与 VREF+ 相关的注 3。 更新了*第 6.1.1 节：独立 A/D 和 DAC 转换器电源和参考电压*。 **RCC** 更新了*图 16：复位电路简化框图*和*图 17：时钟树* 更新了*第 7.2.4 节：HSI48 时钟*。 将 MCOSEL[2:0] 更改为 MCOSEL[3:0]，*第 7.3.4 节：时钟配置寄存器 (RCC_CFGR)* 中更新了*第 7.2.14 节：时钟输出功能*的定义。 在*第 7.3.9 节：AHB 外设复位寄存器 (RCC_AHBRSTR)* 中，将 TOUCHRST 重命名为 TSCRST。在*第 7.3.17 节：睡眠模式下的 AHB 外设时钟使能寄存器 (RCC_AHBSMENR)* 中，将 TOUCHSMEM 重命名为 TSCSMEM。在*第 7.3.20 节：时钟配置寄存器 (RCC_CCIPR)* 中，将 HSI48MSEL 重命名为 HSI48SEL。 **SYSCFG**： 更新了*第 10.1 节：简介* 将 REF_CTRL 寄存器（*第 10.2.3 节*）重命名为 REF_CFGR3， 将 ENBUFLP_BGAP_COMP 更改为 ENBUF_VREFINT_COMP， 将 ENBUF_BGAP_ADC 更改为 ENBUF_VREFINT_ADC， 将 ENBUF_SENSOR_ADC 更改为 ENBUF_TSENSE_ADC， 将 EN_BGAP 更改为 EN_VREFINT 并更新了相关说明。

表 163. 文档版本历史（续）

日期	版本	变更
2014 年 4 月 28 日（续）	2	**中断** 更新了*表 51：向量列表*。 **ADC：** 删除了整个章节中的 JADSTART。 更新了*第 14.2 节：ADC 主要特性*和*图 32：ADC 框图*中的 ADC 电源要求 将“来自 PMU 的模拟参考”部分重命名为*ADC 内部调压器的模拟参考*一节并更新了内容。修改了*第 14.4.2 节：校准 (ADCAL)*。 在*第 14.10 节：温度传感器和内部参考电压*中，将 110 °C 更改为 130 °C。 更新了*第 14.13.4 节：ADC 配置寄存器 1 (ADC_CFGR1)* 中的 AWDCH[4:0] 和 SCANDIR 说明。 **DAC：** 在*第 15.5.2 节：DAC 通道转换*中增加了*生成单个LFSR 的独立触发*一节和*生成单个三角波的独立触发*一节。 **COMP：** 更新了*图 62：比较器 1 和 2 框图*。 更新了*图 62：比较器 1 和 2 框图*中命名的位。重命名了 COMP1_CSR 和 COMP2_CSR 位。 **LCD** 更新了*第 17.4.7 节：COM 和 SEG 多路复用*中的 COM 和 SEG 输出引脚功能。更新了*第 17.7.1 节：LCD 控制寄存器 (LCD_CR)* 中的注释。 **TIM2/3：** 更新了*第 21.4.19 节：TIM2 选项寄存器 (TIM2_OR)*。 将所有寄存器更改为 16 位长。 **TIM21/22：** 更新了*图 144：通用定时器框图 (TIM21/22)* 。 增加了*第 22.4.2 节：TIM21/22 控制寄存器 2 (TIMx_CR2)*。 **USART：** 更新了*第 29.8.3 节：控制寄存器 3 (USARTx_CR3)* 中的 CTSE 位说明。 **DEBUG** 分别将 DBG_TIM20_STOP 和 DBG_TIM21_STOP 替换为 DBG_TIM21_STOP 和 DBG_TIM22_STOP。 更新了*第 33.4.1 节：MCU 器件 ID 代码*中的 REV_ID。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3	*第 1.3 节：外设可用性*：增加了第 5 类 (STM32L07x/8x) 功能，更新了*表 1：STM32L0x3 存储器容量*并增加了*表 2：按类别划分的特性*。 增加了*附录 A：代码示例*。 **系统和存储器概述** 增加了第 5 类功能。 更新了*表 3：STM32L0x3 外设寄存器边界地址*中的 TIMER 7 寄存器地址。 **闪存/数据 EEPROM** 增加了第 5 类功能。 将 FLASH_WRPROT 替换为 FLASH_WRPROT1 并增加了 FLASH_WRPROT2 寄存器。更新了 BOR_LEV 说明并将 FLASH_OPTCR 寄存器中的 BOOT1 重命名为 nBOOT1。 更新了 FLASH_SR 寄存器中 READY 标志的说明。 **CRC** 更新了*第 4.2 节：CRC 主要特性*和*多项式可编程*一节。 **防火墙** 更新了*第 5.3.5 节：防火墙初始化*。 **PWR** 删除了与 ADC 的 1.8V 最小 V_{DDA} 相关的限制，更新了*第 6.1 节：电源*和*图 10：电源概述*中的 V_{REF+}。更新了*第 6.1.5 节：动态电压调节管理*中的范围 1 说明。 更新了*表 31：低功耗模式汇总*。 增加了*第 6.3.5 节：进入低功耗模式*和*第 6.3.6 节：退出低功耗模式*。 更新了*第 6.3.7 节：睡眠模式*，删除有关模式进入和退出的详细信息，并更新了*表 32：立即睡眠*和*表 33：退出时睡眠*。 更新了*第 6.3.8 节：低功耗睡眠模式（LP 睡眠）*，删除有关模式进入和退出的详细信息，并更新了*表 34：立即睡眠（低功耗睡眠）*和*表 35：退出时睡眠（低功耗睡眠）*。 更新了*第 6.3.9 节：停止模式*，删除有关模式进入和退出的详细信息，并更新了*表 36：停止模式*。 更新了*第 6.3.10 节：待机模式*，删除有关模式进入和退出的详细信息，并更新了*表 37：待机模式*。 更新了*第 6.4.1 节：PWR 电源控制寄存器 (PWR_CR)* 中的 LPRUN 位说明。在*第 6.4.2 节：PWR 电源控制/状态寄存器 (PWR_CSR)* 中增加了 EWUP3 位。更新了*范围 1*一节，指定 CRS 仅在范围 1 内可用。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3（续）	**RCC** 更新了*第 7.2 节：时钟*中的 ADC 时钟。在*第 7.2.10 节：HSE 时钟安全系统(CSS)* 中增加了 HSE 故障。 在*第 7.3.3 节：时钟恢复 RC 寄存器 (RCC_CRRCR)* 中增加了 HSI48DIV6EN 并更新了 HSI48DIV6EN。 *第 7.3.7 节：时钟中断清零寄存器 (RCC_CICR)*：将所有位访问类型更改为“w”，将 USB 位重命名为 UFB 并移动到位 3。 将*第 7.3.13 节：AHB 外设时钟使能寄存器 (RCC_AHBENR)* 中的 TOUCHEN 重命名为 TSCEN。 将*第 7.3.14 节：APB2 外设时钟使能寄存器 (RCC_APB2ENR)* 中的 MIFIEN 重命名为 FWEN，并更新了说明。更新了*第 7.3.21 节：控制/状态寄存器 (RCC_CSR)*。 在*第 7.3.8 节：GPIO 复位寄存器 (RCC_IOPRSTR)* 中增加了 IOPERST，在*第 7.3.12 节：GPIO 时钟使能寄存器 (RCC_IOPENR)* 中增加了 IOPEENR，并在*第 7.3.16 节：睡眠模式下的 GPIO 时钟使能寄存器 (RCC_IOPSMENR)* 中增加了 IOPESMEN。 *第 7.3.11 节：APB1 外设复位寄存器 (RCC_APB1RSTR)*：增加了 USART4RST、USART5RST、TIM3RST、TIM7RST 和 I2C3RST。将 UARTxRST 位重命名为 USARTxRST。 *第 7.3.15 节：APB1 外设时钟使能寄存器 (RCC_APB1ENR)*：增加了 USART4EN、USART5EN、TIM3EN、TIM7EN 和 I2C3EN。将 UARTxEN 位重命名为 USARTxEN。 *第 7.3.19 节：睡眠模式下的 APB1 外设时钟使能寄存器 (RCC_APB1SMENR)*：增加了 USART4SMEN、USART5SMEN、TIM3SMEN、TIM7SMEN 和 I2C3SMEN。将 UARTxSMEN 位重命名为 USARTxSMEN。 在*第 7.3.20 节：时钟配置寄存器 (RCC_CCIPR)* 中增加了 I2C3SEL 位。 **CRS：** 在*第 8.6.2 节：CRS 配置寄存器 (CRS_CFGR)* 寄存器中增加了与 SYNCSRC[1:0] 相关的注释。 **GPIO** 为第 5 类器件增加了端口 E。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3（续）	**SYSCFG** 更新了*图 1：系统架构*以增加 STM32L07/08 外设。 在*第 10.2.1 节：SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1)*中增加了 USB 位。将 REF_CFGR3 替换为 SYSCFG_CFGR3。 在*第 10.2.2 节：SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)*中增加了 I2C3_FMP 位并更新了 CAPA 位。 更新了*表 46：VLCD 轨与 GPIO 引脚的连接* 更新了*第 10.2.4 节：SYSCFG 外部中断配置寄存器 1 (SYSCFG_EXTICR1)*、*第 10.2.5 节：SYSCFG 外部中断配置寄存器 2 (SYSCFG_EXTICR2)*、*第 10.2.6 节：SYSCFG 外部中断配置寄存器 3 (SYSCFG_EXTICR3)* 和*第 10.2.7 节：SYSCFG 外部中断配置寄存器 4 (SYSCFG_EXTICR4)*。 **DMA** 更新了第 5 类器件的 DMA 映射/通道选择。 **中断** 将优先级数从 16 更改为 4。 更新了*表 51：向量列表*和*表 52：EXTI 线连接*，增加第 5 类外设并更新向量 17 和 18。在所有 EXTI 寄存器中增加了位 24。 **ADC** 更新了*图 32：ADC 框图*。 *第 14.4.1 节：ADC 调压器 (ADVREGEN)*：将 REF_CTRL 更改为 REF_CFGR3 并将 ENBUF_EN_VREFINT_ADC 更改为 ENBUF_VREFINT_ADC。 删除了与 1.8 V V_{DDA} 最小值相关的限制。将*第 14.10 节：温度传感器和内部参考电压*中的 V_{DDA}= 3.3 V 更改为 3 V。 将*将电源相关的 ADC 测量值转换为绝对电压值*一节中 $V_{CHANNELx}$ 公式中的 3.3 V 更改为 3 V。 更新了*第 14.11 节：VLCD 电压监控*。 更新了*第 14.13.4 节：ADC 配置寄存器 1 (ADC_CFGR1)* 中 ADC_CFGR1 的 AWDCH 位域定义。 **DAC** 增加了双 DAC 功能和 DAC 通道 2。在*表 66：外部触发器*中增加了 TIM3 和 TIM7 TRGO 事件。 将*第 15.6.4 节：DAC 输出电压*中的 VREF+ 更改为 VDDA。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3（续）	**COMP** 更新了*图 62：比较器 1 和 2 框图*。 在*第 16.5.1 节：比较器 1 控制和状态寄存器 (COMP1_CSR)* 中增加了 COMP1LPTIMIN1。 在*第 16.5.2 节：比较器 2 控制和状态寄存器 (COMP2_CSR)* 中增加了 COMP2LPTIMIN2 和 COMP2LPTIMIN1，并更新了 COMP2INSEL 定义。 **LCD** 更新了 LCD 部分，第 5 类器件最多支持 52 个区段。 更新了*图 63：LCD 控制器框图*。 更新了*第 17.4.7 节：COM 和 SEG 多路复用* **RNG**： 将 PLL48CLK 替换为 RNG_CLK。 在*图 97：框图*下增加了注 1。 **通用定时器 (TIM2/3)** 增加了 TIMER3。 删除了 32 位选项。 更新了*第 21.3.10 节：单脉冲模式*中的序列，将 TI2FP2 用作触发信号 1。 在*第 21.3.15 节：定时器同步*中增加了与从定时器时钟相关的注释。 更新了*第 21.4.2 节：TIMx 控制寄存器 2 (TIMx_CR2)* 中的 MMS 位说明，增加了与从定时器时钟相关的注释。 更新了*第 21.4.3 节：TIMx 从模式控制寄存器 (TIMx_SMCR)* 中的 SMS 位和*表 88：TIM2/TIM3 内部触发连接*，增加了与从定时器时钟相关的注释。 删除了*第 21.4.7 节：TIMx 捕获/比较模式寄存器 1 (TIMx_CCMR1)*/输出比较中与 OC1M 中的 TIMx_BDTR 相关的注和 OC1PE 位说明。 更新了*第 21.4.19 节：TIM2 选项寄存器 (TIM2_OR)* 中的 ETR_RMP 说明。 **通用定时器 (TIM21/22)** 更新了*第 22.3.11 节：单脉冲模式*中的序列，将 TI2FP2 用作触发信号 1。 删除了*第 22.4.7 节：TIM21/22 捕获/比较模式寄存器 1 (TIMx_CCMR1)* 的 IC1F 位说明中的注释。 **基本定时器** 增加了 TIMER7。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3（续）	**LPTIM** 更新了*第 24.6.4 节：LPTIM 配置寄存器 (LPTIMx_CFGR)* 中的 TRIGSEL 说明。在*表 99：LPTIM 外部触发连接*中增加了 ext_trig5。 **WWDG：** 更新了*第 26.3.4 节：如何设置看门狗超时*中的*图 200：看门狗框图*以及超时公式和示例。 **RTC** 为第 5 类器件增加了入侵 3 事件。 更新了*图 202：RTC 框图*中的 WUCKSEL 位。 *第 27.3.5 节：可编程闹钟*：将注意事项中的 MSK0 更改为 MSK1。 **I2C** 更新了*第 28.7.1 节：控制寄存器 1 (I2C_CR1)* 中的 NOSTRECH 定义。 **USART：** 为第 5 类器件增加了 USART4/5。更新了*图 235：USART 框图*。增加了低功耗模式部分。 更新了*单字节通信*一节。 更新了*表 125：采用 16 倍或 8 倍过采样时，在 f_{CK} = 32 MHz 下编程波特率的误差计算*。 更新了*图 252：IrDA SIR ENDEC—— 框图*、*图 254：使用 DMA 进行发送*和*图 255：使用 DMA 进行接收*。 删除了 USARTx_CR3 的 UCESM 位以及使 USART 时钟在停止模式期间保持使能的功能。 更新了 USARTx_ISR 寄存器中 REACK 标志的说明。 **LPUART** 更新了*图 260：LPUART 框图*。增加了低功耗模式部分。删除了*第 30.4.1 节：LPUART 字符说明*中的注释。 更新了*表 132：fck = 32,768 KHz 时编程的波特率的错误计算*。更新了*表 136：LPUART 中断请求*。 更改了*表 137：LPUART 寄存器映射和复位值*中的 LPUARTx_RDR 和 LPUARTx_TDR 复位值。删除了 LPUART_CR3 的 UCESM 位以及使 LPUART 时钟在停止模式期间保持使能的功能。 **SPI** 更新了*表 140：使用标准 8 MHz HSE 时的音频频率精度*。

表 163. 文档版本历史（续）

日期	版本	变更
2015 年 5 月 4 日	3（续）	**DEBUG** 更新了 *DBG_IDCODE* 一节中的 REV_ID 位域。在*第 33.9.4 节：调试 MCU APB1 冻结寄存器 (DBG_APB1_FZ)* 中增加了相应位来支持 I2C3、TIM3 和 TIM7。 更新了***附录 A：代码示例***。
2016 年 2 月 19 日	4	更新了*第 2.3 节：嵌入式 SRAM*。 **Flash 程序存储器和数据 EEPROM** 将第 5 类器件的 NVM 存储器结构表拆分为 2 个表：*UFB = 0 时的 NVM 结构*以及 *Flash 存储器和数据 EEPROM 重映射*。 更新了*表 7：Flash 存储器和数据 EEPROM 重映射（192 KB 第 5 类器件）*和*表 9：Flash 存储器和数据 EEPROM 重映射（128 KB 第 5 类器件）*。更新了*表 10：UFB = 0 时的 NVM 构成（64 KB 第 5 类器件）*：禁止 BOOT0= 0 和 UBS = 1 配置。 将整个章节中的总线错误替换为硬性故障。 更新了*第 3.3.2 节：双存储区自举功能*。 更新了*第 3.4.1 节：RDP（读保护）*中 *1 级存储器读保护*的说明。 更新了*第 3.7.8 节：选项字节寄存器 (FLASH_OPTR)*、*第 3.7.9 节：写保护寄存器 1 (FLASH_WRPROT1)* 和*第 3.7.10 节：写保护寄存器 2 (FLASH_WRPROT2)* 中的复位值。 更新了*第 3.7.8 节：选项字节寄存器 (FLASH_OPTR)* 中的 BFB2 位说明。 **电源控制器 (PWR)** 更新了*第 6.2.4 节：内部参考电压 (VREFINT)*，增加了出现 NRST 脉冲时退出待机模式以及与 LCD 激活相关的注释。 在*表 31：低功耗模式汇总*中增加了与停止模式下的 HSI16 相关的注释。 更新了*第 6.3.5 节：进入低功耗模式*、*表 32：立即睡眠*、*表 33：退出时睡眠*、*表 34：立即睡眠（低功耗睡眠）*、*表 35：退出时睡眠（低功耗睡眠）*、*表 36：停止模式*和*表 37：待机模式*中的进入低功耗模式的条件。 更新了 *PWR 电源控制寄存器 (PWR_CR)* 中的 DS_EE_KOFF 和 ULP 位定义。

表 163. 文档版本历史（续）

日期	版本	变更
2016 年 2 月 19 日	4（续）	**复位和时钟控制 (RCC)** 更新了*第 7.1.2 节：电源复位*和*图 16：复位电路简化框图*。 删除了*第 7.2.4 节：HSI48 时钟*中的 EN_VREFINT。更新了*第 7.2.7 节：LSI 时钟*。在*第 7.2.12 节：RTC 和 LCD 时钟*中增加了通过 LSE 驱动 RTC 的情况。更新了*第 7.2.13 节：看门狗时钟* 修改了*第 7.3.1 节：时钟控制寄存器 (RCC_CR)* 中的 HSI16OUTEN 位定义以及 HSI16KERON 和 HSI16RDYF 访问类型。更新了*第 7.3.3 节：时钟恢复 RC 寄存器 (RCC_CRRCR)* 中的寄存器复位值和 HSIDIV6EN 位。 更新了 *睡眠模式下的 GPIO 时钟使能寄存器 (RCC_IOPSMENR)*、*睡眠模式下的 AHB 外设时钟使能寄存器 (RCC_AHBSMENR)*、*睡眠模式下的 APB2 外设时钟使能寄存器 (RCC_APB2SMENR)* 和*睡眠模式下的 APB1 外设时钟使能寄存器 (RCC_APB1SMENR)* 复位值。 **系统配置控制器 (SYSCFG)** 更新了 *SYSCFG 存储器重映射寄存器 (SYSCFG_CFGR1)* 中的 UFB 位说明。 *SYSCFG 外设模式配置寄存器 (SYSCFG_CFGR2)*： – 更新了复位值。 – 将 SYSCFG_CFGR 寄存器中的 CAPA 位重命名为 LCD_CAPA 并修改了位域说明。 删除了*参考控制和状态寄存器 (SYSCFG_CFGR3)* 中的 EN_VERFINT、VREFINT_COMP_RDYF、VREFINT_ADC_RDYF、SENSOR_ADC_RDYF 和 REF_HSI48_RDYF。 **嵌套向量中断控制器** 删除了*表 51：向量列表*中的 MemManage_Handler、BusFault_Handler、Usagefault _Handler 和 DebugMon_Handler。 更新了 EXTI_IMR 复位值。 **通用 I/O (GPIO)** 更新了*第 9.4.3 节：GPIO 端口输出速度寄存器 (GPIOx_OSPEEDR)（x = A..E 和 H）*中的 OSPEEDy[1:0] 定义。 **模数转换器 (ADC)** 将*图 32：ADC 框图*中的 AUTDLY 替换为 WAIT。 将 t_{ADC} 更改为 t_{CONV}。 更新了*ADC 内部调压器的模拟参考*一节。更新了*第 14.4.3 节：ADC 开关控制（ADEN、ADDIS、ADRDY）*中的 ADC 使能序列。更新了*第 14.4.10 节：开始转换 (ADSTART)* 并更新了*第 14.13.3 节：ADC 控制寄存器 (ADC_CR)* 中的 ADSTART 位说明。更新了*第 14.13.1 节：ADC 中断和状态寄存器 (ADC_ISR)* 中的 EOSMP 位说明。

表 163. 文档版本历史（续）

日期	版本	变更
2016 年 2 月 19 日	4（续）	**液晶显示控制器 (LCD)** 更新了*LCD 中间电压*一节、*LCD 驱动器选择*一节和*外部去耦*一节。 更新了*图 73：SEG/COM 多路复用功能示例* 更新了*LCD 控制寄存器 (LCD_CR)* 中的 LCDEN 位说明。 **触摸感应控制器 (TSC)** 删除了电容感应 GPIO 部分。 在*第 18.3.4 节：电荷转移采集序列*中增加了注。 在*第 18.6.1 节：TSC 控制寄存器 (TSC_CR)* 中的 CTPL 和 PGPSC 位说明中增加了注。 **TIMER2/3** 更新了*TIM2 选项寄存器 (TIM2_OR)* 中的 ETR_RMP 位定义。 **TIMER21/22** 更新了*第 22.4.3 节：TIM21/22 从模式控制寄存器 (TIMx_SMCR)* 中 SMS 位的定义。 将*表 94：TIM21/22 寄存器映射和复位值*限制为 16 位而非 32 位。 **TIMER6/7** 将*表 95：TIM6/7 寄存器映射和复位值*限制为 16 位而非 32 位。 **低功耗定时器 (LPTIM)** 更新了*第 24.4.6 节：工作模式*。 增加了*图 194：LPTIM 输出波形，单次计数模式配置*、*图 195：LPTIM 输出波形，单次计数模式配置且激活置 1 一次模式（WAVE 位置 1）*一节和*图 196：LPTIM 输出波形、连续计数模式配置*一节。 更新了*第 24.6.8 节：LPTIM 计数器寄存器 (LPTIMx_CNT)* 中的 CNT 位域定义。 删除了 LPTIM1_OR 和 LPTIM2_OR。 **实时时钟 (RTC)** 更新了*图 202：RTC 框图*下的注释。 更新了*编程唤醒定时器*一节中的步骤 3。 更新了*第 27.3.9 节：复位 RTC* 中系统复位时的 RTC 行为。 修改了*第 27.6.4 节：RTC 初始化和状态寄存器 (RTC_ISR)* 中的 WUTWF 说明。

表 163. 文档版本历史（续）

日期	版本	变更
2016 年 2 月 19 日	4（续）	**内部集成电路接口 (I2C)** 在*I2C 时序*一节中增加了用于确保建立和保持时间的延长机制的说明并在*第 28.7.5 节：时序寄存器 (I2C_TIMINGR)* 中增加了 SCDEL 位说明。 **通用同步/异步收发器 (USART)** 将 nCTS 替换为 CTS，将 nRTS 替换为 RTS，将 SCLK 替换为 CK。 更新了*块模式 (T=1)* 一节中与 RTO 计数器相关的注释。 将*第 29.5.5 节：USART 接收器对时钟偏差的容差*中的 tWUSTOP 更改为 tWUUSART。 更新了*第 29.8.3 节：控制寄存器 3 (USARTx_CR3)* 中的“ONEBIT”位 11 的说明，即增加了注释。更新了*第 29.8.8 节：中断和状态寄存器 (USARTx_ISR)* 中 RTOF 位的定义。 更新了*第 29.5.10 节：USART LIN（局域互连网络）模式*。增加了*当 USART 时钟源为 HSI 时钟时，确定最大 USART 波特率允许从停止模式正确唤醒*一节。 **低功耗 UART (LPUART)** 将 nCTS 替换为 CTS，将 nRTS 替换为 RTS，将 SCLK 替换为 CK。 更新了*第 30.4.4 节：LPUART 波特率生成*。 增加了*第 29.5.5 节：USART 接收器对时钟偏差的容差*和*当 USART 时钟源为 HSI 时钟时，确定最大 USART 波特率允许从停止模式正确唤醒*一节。 更新了*表 135：低功耗模式对 LPUART 的影响*。 删除了*表 137：LPUART 寄存器映射和复位值*中的 TFQRX。 **SPI/I2S：** 更新了*图 273*、*图 274*、*图 275*和*图 276*。 在*图 273*、*图 274* 和*图 275* 下更新并增加了注。 增加了*第 31.3.4 节：多主器件通信*。 **DEBUG** 更新了*第 33.5.1 节：SWD 协议简介*中的 SWDIO 双向管理。 更新了*第 33.9.1 节：对低功耗模式的调试支持*。 更新了*第 33.9.3 节：调试 MCU 配置寄存器（DBG_CR）*。 在 *DBG_IDCODE*一节中增加了*表 155：REV-ID 值*。

表 163. 文档版本历史（续）

日期	版本	变更
2016 年 2 月 19 日	4（续）	代码示例 更新了*附录 A.3.7：编程选项字节的代码示例*和*附录 A.3.9：对 Flash 程序存储器中的单个字进行编程的代码示例*、*附录 A.3.10：对 Flash 程序存储器的半页进行编程的代码示例*和*附录 A.3.11：擦除 Flash 程序存储器中一页的代码示例*。 更新了*附录 A.8.2：ADC 使能序列代码示例*、*附录 A.8.5：单次转换序列代码示例 - 软件触发*、*附录 A.8.6：连续转换序列代码示例 - 软件触发*、*附录 A.8.7：单次转换序列代码示例 - 硬件触发*、*附录 A.8.8：连续转换序列代码示例 - 硬件触发*、*附录 A.8.11：等待模式序列代码示例*、*附录 A.8.12：自动关闭和无等待模式序列代码示例*、*附录 A.8.13：自动关闭和等待模式序列代码示例*、*附录 A.8.14：模拟看门狗代码示例*和*附录 A.8.16：温度配置代码示例*。 更新了*附录 A.11.4：输入捕捉数据管理代码示例*和*附录 A.11.10：清零 OCxREF 的 ETR 配置代码示例*。 更新了*附录 A.15.1：RTC 日历配置代码示例*、*附录 A.15.5：RTC 校准代码示例*和*附录 A.15.7：RTC 入侵和时间戳代码示例*。 更新了*附录 A.17.3：USART 传输完成代码示例*、*附录 A.17.6：USART LIN 模式代码示例*、*附录 A.17.7：USART 同步模式代码示例*、*附录 A.17.8：USART 单线半双工代码示例*、*附录 A.17.9：USART 智能卡模式代码示例*、*附录 A.17.10：USART IrDA 模式代码示例*、*附录 A.17.11：USART DMA 代码示例*和*附录 A.17.12：USART 硬件流控制代码示例*。

表 164. 中文文档版本历史

日期	版本	变更
2017 年 1 月 30 日	1	中文初始版本。

索引

A

ADC_CALFACT 317
ADC_CCR 317
ADC_CFGR1 310
ADC_CFGR2 313
ADC_CHSELR 316
ADC_CR 308
ADC_DR 316
ADC_IER 306
ADC_ISR 305
ADC_SMPR 314
ADC_TR 315
AES_CR 415
AES_DINR 418
AES_DOUTR 418
AES_IVR0 421
AES_IVR1 421
AES_IVR2 422
AES_IVR3 422
AES_KEYR0 419
AES_KEYR1 419
AES_KEYR2 420
AES_KEYR3 420
AES_SR 417

C

COMP1_CSR 348
COMP2_CSR 349
CRC_CR 117
CRC_DR 116
CRC_IDR 116
CRC_INIT 118
CRC_POL 118
CRS_CFGR 216
CRS_CR 215
CRS_ICR 219
CRS_ISR 217

D

DAC_CR 333
DAC_DHR12L1 337
DAC_DHR12L2 339
DAC_DHR12LD 340
DAC_DHR12R1 337
DAC_DHR12R2 338
DAC_DHR12RD 340

DAC_DHR8R1 338
DAC_DHR8R2 339
DAC_DHR8RD 341
DAC_DOR1 341
DAC_DOR2 342
DAC_SR 342
DAC_SWTRIGR 336
DBG_APB1_FZ 880
DBG_APB2_FZ 882
DBG_CR 879
DBG_IDCODE 873
DBGMCU_CR 879
DMA_CCRx 256
DMA_CMARx 259
DMA_CNDTRx 258
DMA_CPARx 258
DMA_CSELR 260
DMA_IFCR 255
DMA_ISR 254

E

EXTI_EMR 271
EXTI_FTSR 273
EXTI_IMR 271
EXTI_PR 275
EXTI_RTSR 272
EXTI_SWIER 274

F

FLASH_ACR 99
FLASH_CR 104
FLASH_KEYR 100
FLASH_OPTKEYR 103-104
FLASH_OPTR 107
FLASH_PDKEYR 103
FLASH_PECR 100
FLASH_PEKEYR 103
FLASH_PRGKEYR 104
FLASH_SR 103, 105
FLASH_WRPROT1 109
FLASH_WRPROT2 110
FMPI2C_ISR 692
FW_CR 130
FW_CSL 127
FW_CSSA 127
FW_NVDSL 128
FW_NVDSSA 128
FW_VDSL 129
FW_VDSSA 129

G

GPIOx_AFRH 235
GPIOx_AFRL 234
GPIOx_BRR 235
GPIOx_BSRR 232
GPIOx_IDR 231
GPIOx_LCKR 233
GPIOx_MODER 229
GPIOx_ODR 232
GPIOx_OSPEEDR 230
GPIOx_OTYPER 230
GPIOx_PUPDR 231

I

I2C_CR1 682
I2C_CR2 685
I2C_ICR 694
I2C_ISR 692
I2C_OAR1 688
I2C_OAR2 689
I2C_PECR 695
I2C_RXDR 696
I2C_TIMEOUTR 691
I2C_TIMINGR 690
I2C_TXDR 696
I2Cx_CR2 . . . . . 127-130, 685
IWDG_KR 582
IWDG_PR 583
IWDG_RLR 584
IWDG_SR 585
IWDG_WINR 586

L

LCD_CLR 379
LCD_CR 373
LCD_RAM 379
LPTIMx_ARR 577
LPTIMx_CFGR 573
LPTIMx_CMP 577
LPTIMx_CNT 578
LPTIMx_CR 576
LPTIMx_ICR 571
LPTIMx_IER 572
LPTIMx_ISR 570
LPUART_BRR 790
LPUART_CR1 783
LPUART_CR2 786
LPUART_CR3 788
LPUART_ICR 794
LPUART_ISR 791

LPUART_RDR 795
LPUART_RQR 791
LPUART_TDR 796

P

PWR_CR 154
PWR_CSR 157

R

RCC_AHBENR 190
RCC_AHBRSTR 184
RCC_AHBSMENR 197
RCC_APB1ENR 193
RCC_APB1RSTR 187
RCC_APB1SMENR 199
RCC_APB2ENR 191
RCC_APB2RSTR 185
RCC_APB2SMENR 198
RCC_CCIPR 202
RCC_CFGR 177
RCC_CICR 182
RCC_CIER 179
RCC_CIFR 181
RCC_CR 172
RCC_CRRCR 176
RCC_CSR 203
RCC_ICSCR 175
RCC_IOPENR 189
RCC_IOPRSTR 183
RCC_IOPSMENR 196
RNG_CR 426
RNG_DR 428
RNG_SR 427
RTC_ALRMAR 617
RTC_ALRMBR 618
RTC_ALRMBSSR 628
RTC_BKPxR 629
RTC_CALR 623
RTC_CR 610
RTC_DR 609
RTC_ISR 613
RTC_OR 628
RTC_PRER 615
RTC_SHIFTR 620
RTC_SSR 619
RTC_TR 608
RTC_TSDR 622
RTC_TSSSR 622
RTC_TSTR 621
RTC_WPR 619
RTC_WUTR 616

S

SPI_CR1 834
SPI_CR2 836
SPI_CRCPR 839
SPI_DR 838
SPI_I2SCFGR 841
SPI_I2SPR 842
SPI_RXCRCR 839
SPI_SR 837
SPI_TXCRCR 840
SYSCFG_CFGR1 239
SYSCFG_CFGR2 240
SYSCFG_CFGR3 241
SYSCFG_EXTICR1 243
SYSCFG_EXTICR2 243
SYSCFG_EXTICR3 244
SYSCFG_EXTICR4 244

T

TIM2_OR 490
TIM21_OR 543
TIM22_OR 544
TIM3_OR 491
TIMx_ARR 485, 541, 558
TIMx_CCER 483, 540
TIMx_CCMR1 . . . . . 479, 537
TIMx_CCMR2 482
TIMx_CCR1 . . . . . 486, 542
TIMx_CCR2 . . . . . 486, 542
TIMx_CCR3 487
TIMx_CCR4 487
TIMx_CNT 485, 541, 557
TIMx_CR1 . . . . . 470, 529, 555
TIMx_CR2 . . . . . 472, 530, 556
TIMx_DCR 488
TIMx_DIER 475, 533, 556
TIMx_DMAR 488
TIMx_EGR 478, 536, 557
TIMx_PSC 485, 541, 558
TIMx_SMCR 473, 531
TIMx_SR 476, 534, 557
TSC_CR 390
TSC_ICR 393
TSC_IER 392
TSC_IOASCR 395
TSC_IOCCR 396
TSC_IOGCSR 397
TSC_IOGxCR 397
TSC_IOHCR 394
TSC_IOSCR 395
TSC_ISR 394

U

USART_CR1 783
USARTx_BRR 749
USARTx_CR1 739
USARTx_CR2 742
USARTx_CR3 745
USARTx_GTPR 750
USARTx_ICR 757
USARTx_ISR 753
USARTx_RDR 758
USARTx_RQR 752
USARTx_RTOR 751
USARTx_TDR 759
USB_ADDRn_RX 867
USB_ADDRn_TX 866
USB_BCDR 861
USB_BTABLE 860
USB_CNTR 855
USB_COUNTn_RX 867
USB_COUNTn_TX 867
USB_DADDR 860
USB_EPnR 863
USB_FNR 859
USB_ISTR 857
USB_LPMCSR 861

W

WWDG_CFR 591
WWDG_CR 590
WWDG_SR 591

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用，ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。

© 2017 STMicroelectronics - 保留所有权利