

# PY32F07X 系列

## 32 位 ARM® Cortex®-M0+ 微控制器

### 参考手册



Puya Semiconductor (Shanghai) Co., Ltd.

## 目录

<b>1. 寄存器描述中使用的缩写列表</b> .....	<b>24</b>
<b>2. 系统架构框图</b> .....	<b>25</b>
<b>3. 存储器和总线架构</b> .....	<b>26</b>
3.1. 系统架构.....	26
3.2. 存储器结构.....	27
3.2.1. 存储器结构简介.....	27
3.3. 嵌入式 SRAM.....	30
3.4. Boot 模式.....	31
3.4.1. 存储器物理映像.....	31
3.4.2. 内嵌的自举程序.....	31
<b>4. 嵌入式闪存</b> .....	<b>32</b>
4.1. 闪存主要特性.....	32
4.2. 闪存功能介绍.....	32
4.2.1. 闪存结构.....	32
4.2.2. 闪存读操作和访问延迟.....	33
4.2.3. 闪存写操作和擦除操作.....	33
4.3. Flash 选项字节.....	36
4.3.1. Flash 选项字.....	36
4.3.2. Flash 选项字节写.....	38
4.4. Flash 配置字节.....	40
4.4.1. HSI_TRIMMING_FOR_USER.....	41
4.4.2. 温度传感器的校准值.....	41
4.4.3. HSI_4M/8M/16M/22.12M/24M_EPPARA0.....	42
4.4.4. HSI_4M/8M/16M/22.12M/24M_EPPARA1.....	42
4.4.5. HSI_4M/8M/16M/22.12M/24M_EPPARA2.....	42
4.4.6. HSI_4M/8M/16M/22.12M/24M_EPPARA3.....	43
4.4.7. HSI_4M/8M/16M/22.12M/24M_EPPARA4.....	43
4.5. 闪存保护.....	43
4.5.1. 闪存读保护.....	43
4.5.2. 闪存写保护.....	45
4.5.3. 选项字节写保护.....	45
4.6. 闪存中断.....	45
4.7. 闪存寄存器描述.....	45
4.7.1. Flash 访问控制寄存器 (FLASH_ACR).....	45
4.7.2. Flash 密钥寄存器 (FLASH_KEYR).....	46
4.7.3. Flash 选项密钥寄存器 (FLASH_OPTKEYR).....	46
4.7.4. Flash 状态寄存器 (FLASH_SR).....	47

4.7.5.	Flash 控制寄存器 (FLASH_CR) .....	48
4.7.6.	Flash 选项寄存器 (FLASH_OPTR) .....	50
4.7.7.	Flash BORCR 地址寄存器 (FLASH_BORCR) .....	50
4.7.8.	Flash WRP 地址寄存器 (FLASH_WRPR) .....	51
4.7.9.	Flash 睡眠时间配置寄存器 (FLASH_STCR) .....	52
4.7.10.	Flash TS0 寄存器 (FLASH_TS0) .....	53
4.7.11.	Flash TS1 寄存器 (FLASH_TS1) .....	53
4.7.12.	Flash TS2P 寄存器 (FLASH_TS2P) .....	54
4.7.13.	Flash TPS3 寄存器 (FLASH_TPS3) .....	54
4.7.14.	Flash TS3 寄存器 (FLASH_TS3) .....	55
4.7.15.	Flash 页擦写 (PAGE ERASE) TPE 寄存器 (FLASH_PERTPE) .....	55
4.7.16.	Flash SECTOR/MASS ERASE TPE 寄存器 (FLASH_SMERTPE) .....	56
4.7.17.	Flash PROGRAM TPE 寄存器 (FLASH_PRGTPE) .....	56
4.7.18.	Flash PRE-PROGRAM TPE 寄存器 (FLASH_PRETPE) .....	57
<b>5.</b>	<b>电源控制 .....</b>	<b>58</b>
5.1.	电源 .....	58
5.1.1.	电源框图 .....	58
5.2.	电压调节器 .....	59
5.3.	电源监控 .....	59
5.3.1.	上电复位 (POR) / 下电复位 (PDR) / 欠压复位 (BOR) .....	59
5.3.2.	可编程电压检测器 (PVD) .....	60
<b>6.</b>	<b>低功耗控制 .....</b>	<b>61</b>
6.1.	低功耗模式 .....	61
6.1.1.	低功耗模式介绍 .....	61
6.1.2.	低功耗模式开关 .....	62
6.1.3.	各工作模式下的功能 .....	62
6.2.	Sleep 模式 .....	63
6.2.1.	进入 sleep 模式 .....	63
6.2.2.	退出 sleep mode .....	63
6.3.	Stop 模式 .....	64
6.3.1.	进入 Stop 模式 .....	64
6.3.2.	退出 Stop 模式 .....	64
6.4.	降低系统时钟频率 .....	65
6.5.	外设时钟门控 .....	65
6.6.	电源管理寄存器 .....	66
6.6.1.	电源控制寄存器 1 (PWR_CR1) .....	66
6.6.2.	电源控制寄存器 2 (PWR_CR2) .....	67
6.6.3.	电源状态寄存器 (PWR_SR) .....	68
<b>7.</b>	<b>复位 .....</b>	<b>69</b>

7.1.	复位源 .....	69
7.1.1.	电源复位 .....	69
7.1.2.	系统复位 .....	69
7.1.3.	NRST 管脚 (External reset) .....	69
7.1.4.	看门狗复位 .....	69
7.1.5.	软件复位 .....	70
7.1.6.	选项字节加载器复位 .....	70
<b>8.</b>	<b>时钟 .....</b>	<b>71</b>
8.1.	时钟源 .....	71
8.1.1.	外部高速时钟 HSE .....	71
8.1.2.	内部高速时钟 HSI .....	71
8.1.3.	内部低速时钟 LSI .....	71
8.1.4.	HSI10M 时钟 .....	72
8.1.5.	PLL .....	72
8.1.6.	LSE 时钟 .....	72
8.2.	时钟树 .....	73
8.3.	时钟安全系统 (CSS) .....	73
8.3.1.	时钟配置和状态安全 .....	73
8.4.	输出时钟能力 .....	74
8.5.	复位/时钟寄存器 .....	75
8.5.1.	时钟控制寄存器 (RCC_CR) .....	75
8.5.2.	内部时钟源校准寄存器 (RCC_ICSCR) .....	77
8.5.3.	时钟配置寄存器 (RCC_CFGR) .....	78
8.5.4.	PLL 配置寄存器 (RCC_PLLCFGR) .....	80
8.5.5.	外部时钟源控制寄存器 (RCC_ECSCR) .....	80
8.5.6.	时钟中断使能寄存器 (RCC_CIER) .....	82
8.5.7.	时钟中断标志寄存器 (RCC_CIFR) .....	83
8.5.8.	时钟中断清除寄存器 (RCC_CICR) .....	84
8.5.9.	I/O 接口复位寄存器 (RCC_IOPRSTR) .....	85
8.5.10.	AHB 外设复位寄存器 (RCC_AHBSTR) .....	86
8.5.11.	APB 外设复位寄存器 1 (RCC_APBSTR1) .....	87
8.5.12.	APB 外设复位寄存器 2 (RCC_APBSTR2) .....	89
8.5.13.	I/O 接口时钟使能寄存器 (RCC_IOPENR) .....	90
8.5.14.	AHB 外设时钟使能寄存器 (RCC_AHBENR) .....	91
8.5.15.	APB 外设时钟使能寄存器 1 (RCC_APBENR1) .....	92
8.5.16.	APB 外设时钟使能寄存器 2 (RCC_APBENR2) .....	94
8.5.17.	外设独立时钟配置寄存器 (RCC_CCIPR) .....	96
8.5.18.	RTC 域控制寄存器 (RCC_BDCR) .....	97
8.5.19.	控制/状态寄存器 (RCC_CSR) .....	99

<b>9. 时钟校准控制器 (CTC)</b> .....	<b>101</b>
9.1. CTC 简介.....	101
9.2. CTC 主要特性 .....	101
9.3. CTC 功能描述 .....	101
9.3.1. CTC 框图 .....	101
9.3.2. REF 同步脉冲发生器 .....	102
9.3.3. CTC 校准计数器 .....	102
9.3.4. 频率评估和自动校准过程 .....	102
9.3.5. 软件编程指南 .....	103
9.4. CTC 寄存器 .....	104
9.4.1. CTC 控制寄存器 0 (CTC_CTL0) .....	104
9.4.2. CTC 控制寄存器 1 (CTC_CTL1) .....	105
9.4.3. CTC 状态寄存器 (CTC_SR) .....	106
9.4.4. CTC 中断清除寄存器 (CTC_INTC) .....	109
<b>10. 通用 I/O (GPIO)</b> .....	<b>110</b>
10.1. 通用 IO 简介 .....	110
10.2. 通用 IO 功能描述 .....	110
10.3. 通用 IO 功能描述 .....	110
10.3.1. 通用 I/O (GPIO) .....	111
10.3.2. I/O 管脚复用功能多路选择和映射 .....	111
10.3.3. I/O 控制寄存器 .....	112
10.3.4. I/O 数据寄存器 .....	112
10.3.5. I/O 数据按位处理 .....	113
10.3.6. I/O 复用功能输入/输出模式配置 .....	113
10.3.7. 外部中断/唤醒线 .....	113
10.3.8. I/O 输入配置 .....	113
10.3.9. I/O 输出配置 .....	114
10.3.10. 复用功能配置 .....	115
10.3.11. 模拟配置 .....	116
10.3.12. 使用 HSE/LSE 管脚作为 GPIO .....	116
10.4. GPIO 寄存器 .....	117
10.4.1. GPIO 端口模式寄存器 (GPIOx_MODER) (x=A, B, C, F) .....	117
10.4.2. GPIO 端口输出类型寄存器 (GPIOx_OTYPER) (x = A, B, C, F) .....	117
10.4.3. GPIO 端口输出速度寄存器 (GPIOx_OSPEEDR) (x = A, B, C, F) .....	118
10.4.4. GPIO 端口上下拉寄存器 (GPIOx_PUPDR) (x = A, B, C, F) .....	118
10.4.5. GPIO 端口输入数据寄存器 (GPIOx_IDR) (x = A, B, C, F) .....	119
10.4.6. GPIO 端口输出数据寄存器 (GPIOx_ODR) (x = A, B, C, F) .....	119
10.4.7. GPIO 端口位设置/复位寄存器 (GPIOx_BSRR) (x = A, B, C, F) .....	120
10.4.8. GPIO 端口配置锁定寄存器 (GPIOx_LCKR) (x = A, B, C, F) .....	120

10.4.9.	GPIO 复用功能寄存器 (low) (GPIOx_AFRL) (x = A, B, C, F) .....	121
10.4.10.	GPIO 复用功能寄存器 (high) (GPIOx_AFRH) (x = A, B, C, F) .....	122
10.4.11.	GPIO 端口位复位寄存器 (GPIOx_BRR) (x = A, B, C, F) .....	122
<b>11.</b>	<b>系统配置控制器 (SYSCFG) .....</b>	<b>124</b>
11.1.	系统配置寄存器 .....	124
11.1.1.	SYSCFG 配置寄存器 1 (SYSCFG_CFGR1) .....	124
11.1.2.	SYSCFG 配置寄存器 2 (SYSCFG_CFGR2) .....	125
11.1.3.	SYSCFG 配置寄存器 3 (SYSCFG_CFGR3) .....	128
11.1.4.	SYSCFG 配置寄存器 4 (SYSCFG_CFGR4) .....	130
11.1.5.	GPIOA 滤波使能 (PA_ENS) .....	130
11.1.6.	GPIOB 滤波使能 (PB_ENS) .....	131
11.1.7.	GPIOC 滤波使能 (PC_ENS) .....	131
11.1.8.	GPIOF 滤波使能 (PF_ENS) .....	132
11.1.9.	I <sup>2</sup> C 配置寄存器 (SYSCFG_EIIC) .....	132
<b>12.</b>	<b>直接存储器存取 (DMA) .....</b>	<b>134</b>
12.1.	简介 .....	134
12.2.	DMA 主要特性 .....	134
12.3.	DMA 功能描述 .....	135
12.3.1.	DMA 处理 .....	135
12.3.2.	仲裁器 .....	135
12.3.3.	DMA 通道 .....	136
12.3.4.	数据传输宽度/对齐方式/大小端 .....	138
12.3.5.	错误管理 .....	140
12.3.6.	DMA 中断 .....	140
12.3.7.	DMA 外设请求映射 .....	141
<b>12.4.</b>	<b>DMA 寄存器 .....</b>	<b>141</b>
12.4.1.	DMA 中断状态寄存器 (DMA_ISR) .....	141
12.4.2.	DMA 中断标志位清除寄存器 (DMA_IFCR) .....	145
12.4.3.	DMA 通道 1 配置寄存器 (DMA_CCR1) .....	147
12.4.4.	DMA 通道 1 数据传输数量寄存器 (DMA_CNDTR1) .....	149
12.4.5.	DMA 通道 1 外设地址寄存器 (DMA_CPAR1) .....	149
12.4.6.	DMA 通道 1 存储地址寄存器 (DMA_CMAR1) .....	150
12.4.7.	DMA 通道 2 配置寄存器 (DMA_CCR2) .....	150
12.4.8.	DMA 通道 2 数据传输数量寄存器 (DMA_CNDTR2) .....	152
12.4.9.	DMA 通道 2 外设地址寄存器 (DMA_CPAR2) .....	152
12.4.10.	DMA 通道 2 存储地址寄存器 (DMA_CMAR2) .....	153
12.4.11.	DMA 通道 3 配置寄存器 (DMA_CCR3) .....	153
12.4.12.	DMA 通道 3 数据传输数量寄存器 (DMA_CNDTR3) .....	155
12.4.13.	DMA 通道 3 外设地址寄存器 (DMA_CPAR3) .....	155

12.4.14.	DMA 通道 3 存储器地址寄存器 (DMA_CMAR3)	156
12.4.15.	DMA 通道 4 配置寄存器 (DMA_CCR4)	156
12.4.16.	DMA 通道 4 数据传输个数寄存器 (DMA_CNDTR4)	158
12.4.17.	DMA 通道 4 外设地址寄存器 (DMA_CPAR4)	158
12.4.18.	DMA 通道 4 存储器地址寄存器 (DMA_CMAR4)	159
12.4.19.	DMA 通道 5 配置寄存器 (DMA_CCR5)	159
12.4.20.	DMA 通道 5 传输个数寄存器 (DMA_CNDTR5)	161
12.4.21.	DMA 通道 5 外设地址寄存器 (DMA_CPAR5)	161
12.4.22.	DMA 通道 5 存储器地址寄存器 (DMA_CMAR5)	162
12.4.23.	DMA 通道 6 配置寄存器 (DMA_CCR6)	162
12.4.24.	DMA 通道 6 传输个数寄存器 (DMA_CNDTR6)	164
12.4.25.	DMA 通道 6 外设地址寄存器 (DMA_CPAR6)	164
12.4.26.	DMA 通道 6 存储器地址寄存器 (DMA_CMAR6)	165
12.4.27.	DMA 通道 7 配置寄存器 (DMA_CCR7)	165
12.4.28.	DMA 通道 7 传输个数寄存器 (DMA_CNDTR7)	167
12.4.29.	DMA 通道 7 外设地址寄存器 (DMA_CPAR7)	167
12.4.30.	DMA 通道 7 存储器地址寄存器 (DMA_CMAR7)	168
<b>13.</b>	<b>中断和事件</b>	<b>169</b>
<b>13.1.</b>	<b>嵌套向量中断控制器 (NVIC)</b>	<b>169</b>
13.1.1.	主要特性	169
13.1.2.	系统嘀嗒 (SysTick) 校准值寄存器	169
13.1.3.	中断和异常向量	169
<b>13.2.</b>	<b>外部中断/事件控制器 (EXTI)</b>	<b>170</b>
13.2.1.	EXTI 主要特性	170
13.2.2.	EXTI 框图	171
13.2.3.	中断管理	171
13.2.4.	功能描述	171
13.2.5.	硬件中断选择	172
13.2.6.	硬件事件选择	172
13.2.7.	软件中断/事件选择	172
13.2.8.	EXTI 选择器	173
<b>13.3.</b>	<b>EXTI 寄存器</b>	<b>174</b>
13.3.1.	上升沿触发选择寄存器 (EXTI_RTSCR)	174
13.3.2.	下降沿触发选择寄存器 (EXTI_FTSR)	176
13.3.3.	软件中断事件寄存器 (EXTI_SWIER)	178
13.3.4.	挂起寄存器 (EXTI_PR)	181
13.3.5.	外部中断选择寄存器 1 (EXTI_EXTICR1)	184
13.3.6.	外部中断选择寄存器 2 (EXTI_EXTICR2)	185
13.3.7.	外部中断选择寄存器 3 (EXTI_EXTICR3)	186

13.3.8.	外部中断选择寄存器 4 (EXTI_EXTICR4)	187
13.3.9.	中断屏蔽寄存器 (EXTI_IMR)	188
13.3.10.	事件屏蔽寄存器 (EXTI_EMR)	190
<b>14.</b>	<b>循环冗余校验 (CRC)</b>	<b>193</b>
14.1.	简介	193
14.2.	CRC 主要特点	193
14.3.	CRC 功能描述	193
14.3.1.	CRC 框图	193
14.4.	CRC 寄存器	194
14.4.1.	数据寄存器 (CRC_DR)	194
14.4.2.	独立数据寄存器 (CRC_IDR)	194
14.4.3.	控制寄存器 (CRC_CR)	195
<b>15.</b>	<b>数字/模拟转换 (DAC)</b>	<b>196</b>
15.1.	DAC 简介	196
15.2.	DAC 主要特性	196
15.3.	DAC 功能描述	197
15.3.1.	使用 DAC 通道	197
15.3.2.	使用 DAC 输出缓存	197
15.3.3.	DAC 数据格式	197
15.3.4.	DAC 转换	198
15.3.5.	DAC 输出电压	199
15.3.6.	选择 DAC 触发	199
15.3.7.	DMA 功能	199
<b>15.4.</b>	<b>DAC 寄存器</b>	<b>206</b>
15.4.1.	DAC 控制寄存器 (DAC_CR)	206
15.4.2.	DAC 软件触发寄存器 (DAC_SWTRIGR)	209
15.4.3.	DAC 通道 1 的 12 位右对齐数据保持寄存器 (DAC_DHR12R1)	210
15.4.4.	DAC 通道 1 的 12 位左对齐数据保持寄存器 (DAC_DHR12L1)	210
15.4.5.	DAC 通道 1 的 8 位右对齐数据保持寄存器 (DAC_DHR8R1)	211
15.4.6.	DAC 通道 2 的 12 位右对齐数据保持寄存器 (DAC_DHR12R2)	211
15.4.7.	DAC 通道 2 的 12 位左对齐数据保持寄存器 (DAC_DHR12L2)	212
15.4.8.	DAC 通道 2 的 8 位右对齐数据保持寄存器 (DAC_DHR8R2)	212
15.4.9.	双 DAC 的 12 位右对齐数据保持寄存器 (DAC_DHR12RD)	213
15.4.10.	双 DAC 的 12 位左对齐数据保持寄存器 (DAC_DHR12LD)	213
15.4.11.	双 DAC 的 8 位右对齐数据保持寄存器 (DAC_DHR8RD)	214
15.4.12.	DAC 通道 1 数据输出寄存器 (DAC_DOR1)	214
15.4.13.	DAC 通道 2 数据输出寄存器 (DAC_DOR2)	215
15.4.14.	DAC 状态寄存器 (DAC_SR)	215
<b>16.</b>	<b>模拟/数字转换 (ADC)</b>	<b>217</b>



<b>16.1.</b>	<b>简介</b>	217
<b>16.2.</b>	<b>ADC 主要特性</b>	217
<b>16.3.</b>	<b>ADC 功能描述</b>	218
16.3.1.	ADC 框图	218
16.3.2.	校准	218
16.3.3.	ADC 开关控制	219
16.3.4.	ADC 时钟	219
16.3.5.	通道选择	219
16.3.6.	可编程采样时间	220
16.3.7.	可配置的分辨率	220
16.3.8.	单次转换模式	220
16.3.9.	连续转换模式	221
16.3.10.	扫描模式	221
16.3.11.	间断转换模式	221
16.3.12.	注入通道管理	222
16.3.13.	停止进行中的转换 (ADSTP)	223
<b>16.4.</b>	<b>模拟看门狗</b>	223
<b>16.5.</b>	<b>外部触发转换</b>	224
<b>16.6.</b>	<b>数据对齐</b>	225
<b>16.7.</b>	<b>数据过载</b>	225
<b>16.8.</b>	<b>DMA 请求</b>	225
<b>16.9.</b>	<b>温度传感器和内部参考电压</b>	226
<b>16.10.</b>	<b>ADC 中断</b>	227
<b>16.11.</b>	<b>ADC 寄存器</b>	227
16.11.1.	ADC 状态寄存器 (ADC_SR)	227
16.11.2.	ADC 控制寄存器 1 (ADC_CR1)	228
16.11.3.	ADC 控制寄存器 (ADC_CR2)	231
16.11.4.	ADC 采样时间寄存器 1 (ADC_SMPR1)	234
16.11.5.	ADC 采样时间寄存器 2 (ADC_SMPR2)	234
16.11.6.	ADC 采样时间寄存器 3 (ADC_SMPR3)	235
16.11.7.	ADC 注入通道数据偏移寄存器 x (ADC_JOFRx) (x=1..4)	235
16.11.8.	ADC 看门狗高阈值寄存器 (ADC_HTR)	236
16.11.9.	ADC 看门狗低阈值寄存器 (ADC_LTR)	236
16.11.10.	ADC 规则序列寄存器 1 (ADC_SQR1)	237
16.11.11.	ADC 规则序列寄存器 2 (ADC_SQR2)	238
16.11.12.	ADC 规则序列寄存器 3 (ADC_SQR3)	238
16.11.13.	ADC 注入序列寄存器 (ADC_JSQR)	239
16.11.14.	ADC 注入数据寄存器 x (ADC_JDRx) (x= 1..4)	240
16.11.15.	ADC 规则数据寄存器 (ADC_DR)	240

16.11.16. ADC 校准配置和状态寄存器 (ADC_CCSR) .....	241
<b>17. 液晶控制器 (LCD) .....</b>	<b>243</b>
17.1. 简介 .....	243
17.2. LCD 主要特性 .....	243
17.3. LCD 框图 .....	244
17.4. LCD 时钟 .....	244
17.5. LCD 驱动波形 .....	244
17.5.1. 静态驱动波形 .....	245
17.5.2. 1/2DUTY 1/2BIAS 驱动波形 .....	245
17.5.3. 1/8DUTY 1/3BIAS 驱动波形 .....	245
17.6. LCD BIAS 产生电路 .....	246
17.6.1. 内部电阻模式 .....	246
17.6.2. 外部电阻模式 .....	247
17.7. DMA .....	247
17.8. 中断 .....	247
17.9. LCD 显示模式 .....	248
17.9.1. LCD 显示模式 1 (MODE = 1) .....	248
17.9.2. LCD 显示模式 0 (MODE = 0) .....	250
17.10. LCD 寄存器 .....	253
17.10.1. 配置寄存器 0 (LCD_CR0) .....	253
17.10.2. 配置寄存器 1 (LCD_CR1) .....	254
17.10.3. 中断清除寄存器 (LCD_INTCLR) .....	255
17.10.4. 输出配置寄存器 (LCD_POEN0) .....	255
17.10.5. 输出配置寄存器 1 (LCD_POEN1) .....	256
17.10.6. LCD_RAM0~7 .....	257
17.10.7. LCD_RAM8~F .....	258
<b>18. 比较器 (COMP) .....</b>	<b>259</b>
18.1. 简介 .....	259
18.2. COMP 主要特性 .....	259
18.3. COMP 功能描述 .....	260
18.3.1. COMP 框图 .....	260
18.3.2. COMP 管脚和内部信号 .....	261
18.3.3. COMP 复位和时钟 .....	261
18.3.4. 窗口比较器 .....	261
18.3.5. 迟滞 .....	262
18.3.6. 功耗模式 .....	262
18.3.7. 比较器滤波 .....	262
18.3.8. COMP 中断 .....	263
18.4. COMP 寄存器 .....	263

18.4.1.	COMP1 控制和状态寄存器 (COMP1_CSR)	263
18.4.2.	COMP1 滤波寄存器 (COMP1_FR)	265
18.4.3.	COMP2 控制和状态寄存器 (COMP2_CSR)	266
18.4.4.	COMP2 滤波寄存器 (COMP2_FR)	268
18.4.5.	COMP3 控制和状态寄存器 (COMP3_CSR)	268
18.4.6.	COMP3 滤波寄存器 (COMP3_FR)	270
<b>19.</b>	<b>运算放大器 (OPA)</b>	<b>271</b>
19.1.	OPA 简介	271
19.2.	OPA 主要特性	271
19.3.	OPA 功能描述	271
19.3.1.	OPA 框图	271
19.4.	OPA 寄存器	272
19.4.1.	OPA 输出使能寄存器 (OPA_CR0)	272
19.4.2.	OPA 控制寄存器 (OPA_CR1)	272
<b>20.</b>	<b>硬件除法器 (DIV)</b>	<b>273</b>
20.1.	DIV 简介	273
20.2.	DIV 主要特性	273
20.3.	DIV 功能描述	273
20.3.1.	DIV 操作流程	273
20.4.	DIV 寄存器	274
20.4.1.	DIV 被除数寄存器 (DIV_DEND)	274
20.4.2.	DIV 除数寄存器 (DIV_SOR)	274
20.4.3.	DIV 商寄存器 (DIV_QUOT)	274
20.4.4.	DIV 余数寄存器 (DIV_REMD)	275
20.4.5.	DIV 符号寄存器 (DIV_SIGN)	275
20.4.6.	DIV 状态寄存器 (DIV_STAT)	275
<b>21.</b>	<b>高级控制定时器 (TIM1)</b>	<b>277</b>
21.1.	TIM1 简介	277
21.2.	TIM1 主要特性	277
21.3.	TIM1 功能描述	278
21.3.1.	时基单元	278
21.3.2.	计数器模式	279
21.3.3.	重复计数器	287
21.3.4.	时钟源	288
21.3.5.	捕获/比较通道	290
21.3.6.	输入捕获模式	292
21.3.7.	PWM 输入模式	292
21.3.8.	强置输出模式	293
21.3.9.	输出比较模式	293

21.3.10.	PWM 模式.....	294
21.3.11.	互补输出和死区插入 .....	296
21.3.12.	使用刹车功能 .....	298
21.3.13.	在外部事件时清除 OCxREF 信号 .....	299
21.3.14.	六步 PWM 的产生 .....	300
21.3.15.	单脉冲模式.....	301
21.3.16.	编码器接口模式.....	302
21.3.17.	定时器输入异或功能 .....	303
21.3.18.	与霍尔传感器的接口 .....	304
21.3.19.	定时器和外部的触发同步 .....	305
21.3.20.	定时器同步.....	308
21.3.21.	调试模式 .....	308
<b>21.4.</b>	<b>TIM1 寄存器描述 .....</b>	<b>308</b>
21.4.1.	TIM1 控制寄存器 1 (TIM1_CR1) .....	308
21.4.2.	TIM1 控制寄存器 2 (TIM1_CR2) .....	310
21.4.3.	TIM1 从模式控制寄存器 (TIM1_SMCR) .....	312
21.4.4.	TIM1 DMA/中断使能寄存器 (TIM1_DIER) .....	315
21.4.5.	TIM1 状态寄存器 (TIM1_SR) .....	316
21.4.6.	TIM1 事件产生寄存器 (TIM1_EGR) .....	319
21.4.7.	TIM1 捕获/比较模式寄存器 1 (TIM1_CCMR1) .....	321
21.4.8.	TIM1 捕获/比较模式寄存器 2 (TIM1_CCMR2) .....	325
21.4.9.	TIM1 捕获/比较使能寄存器 (TIM1_CCER) .....	326
21.4.10.	TIM1 计数器 (TIM1_CNT) .....	330
21.4.11.	TIM1 预分频器 (TIM1_PSC) .....	330
21.4.12.	TIM1 自动重载寄存器 (TIM1_ARR) .....	330
21.4.13.	TIM1 重复计数器寄存器 (TIM1_RCR) .....	331
21.4.14.	TIM1 捕获/比较寄存器 1 (TIM1_CCR1) .....	332
21.4.15.	TIM1 捕捉/比较寄存器 2 (TIM1_CCR2) .....	332
21.4.16.	TIM1 捕获/比较寄存器 3 (TIM1_CCR3) .....	333
21.4.17.	TIM1 捕捉/比较寄存器 4 (TIM1_CCR4) .....	334
21.4.18.	TIM1 刹车和死区寄存器 (TIM1_BDTR) .....	334
21.4.19.	TIM1 DMA 控制寄存器 (TIM1_DCR) .....	337
21.4.20.	TIM1 连续模式的 DMA 地址 (TIM1_DMAR) .....	338
<b>22.</b>	<b>通用定时器 (TIM2/3) .....</b>	<b>339</b>
22.1.	TIM2/TIM3 简介 .....	339
22.2.	TIM2/3 主要特性 .....	339
22.3.	TIM2/3 功能描述 .....	340
22.3.1.	时基单元 .....	340
22.3.2.	计数器模式.....	341

22.3.3.	时钟源 .....	349
22.3.4.	捕获/比较通道 .....	350
22.3.5.	输入捕获模式 .....	352
22.3.6.	PWM 输入模式 .....	352
22.3.7.	强置输出模式 .....	353
22.3.8.	输出比较模式 .....	353
22.3.9.	PWM 模式 .....	354
22.3.10.	单脉冲模式 .....	356
22.3.11.	编码器接口模式 .....	357
22.3.12.	定时器输入异或功能 .....	359
22.3.13.	定时器和外部触发的同步 .....	359
22.3.14.	定时器同步 .....	362
22.3.15.	调试模式 .....	365
<b>22.4.</b>	<b>寄存器描述 .....</b>	<b>365</b>
22.4.1.	TIM2/3 控制寄存器 1 (TIMx_CR1) .....	365
22.4.2.	TIM2/3 控制寄存器 2 (TIMx_CR2) .....	367
22.4.3.	TIM2/3 从模式控制寄存器 (TIMx_SMCR) .....	369
22.4.4.	TIM2/3 DMA/中断使能寄存器 (TIMx_DIER) .....	371
22.4.5.	TIM2/3 状态寄存器 (TIMx_SR) .....	373
22.4.6.	TIM2/3 事件产生寄存器 (TIMx_EGR) .....	375
22.4.7.	TIM2/3 捕获/比较模式寄存器 1 (TIMx_CCMR1) .....	376
22.4.8.	TIM2/3 捕获/比较模式寄存器 2 (TIMx_CCMR2) .....	380
22.4.9.	TIM2/3 捕获/比较使能寄存器 (TIMx_CCER) .....	382
22.4.10.	TIM2/3 计数器 (TIMx_CNT) .....	384
22.4.11.	TIM2/3 预分频器 (TIMx_PSC) .....	384
22.4.12.	TIM2/3 自动重载寄存器 (TIMx_ARR) .....	385
22.4.13.	TIM2/3 捕获/比较寄存器 1 (TIMx_CCR1) .....	385
22.4.14.	TIM2/3 捕获/比较寄存器 2 (TIMx_CCR2) .....	386
22.4.15.	TIM2/3 捕获/比较寄存器 3 (TIMx_CCR3) .....	387
22.4.16.	TIM2/3 捕获/比较寄存器 4 (TIMx_CCR4) .....	387
22.4.17.	TIM2/3 DMA 控制寄存器 (TIMx_DCR) .....	388
22.4.18.	TIM2/3 连续模式的 DMA 地址 (TIMx_DMAR) .....	389
<b>23.</b>	<b>基本定时器 (TIM6/7) .....</b>	<b>391</b>
23.1.	TIM6 和 TIM7 简介 .....	391
23.2.	TIM6 和 TIM7 的主要特性 .....	391
23.3.	TIM6 和 TIM7 功能描述 .....	391
23.3.1.	时基单元 .....	391
23.3.2.	时钟源 .....	396
23.3.3.	调试模式 .....	396

<b>23.4. TIM6 和 TIM7 寄存器</b> .....	<b>397</b>
23.4.1. TIM6 和 TIM7 控制寄存器 1 (TIMx_CR1) .....	397
23.4.2. TIM6 和 TIM7 控制寄存器 2 (TIMx_CR2) .....	398
23.4.3. TIM6 和 TIM7 DMA/中断使能寄存器 (TIM14_DIER) .....	399
23.4.4. TIM6 和 TIM7 状态寄存器 (TIMx_SR) .....	399
23.4.5. TIM6 和 TIM7 事件产生寄存器 (TIMx_EGR) .....	400
23.4.6. TIM6 和 TIM7 计数器 (TIMx_CNT) .....	400
23.4.7. TIM6 和 TIM7 预分频器 (TIMx_PSC) .....	401
23.4.8. TIM6 和 TIM7 自动重载寄存器 (TIMx_ARR) .....	401
<b>24. 通用定时器 (TIM14)</b> .....	<b>403</b>
<b>24.1. TIM14 简介</b> .....	<b>403</b>
<b>24.2. TIM14 主要特性</b> .....	<b>403</b>
<b>24.3. TIM14 功能描述</b> .....	<b>404</b>
24.3.1. 时基单元 .....	404
24.3.2. 时钟源 .....	408
24.3.3. 捕获/比较通道 .....	408
24.3.4. 输入捕获模式 .....	409
24.3.5. 强置输出模式 .....	410
24.3.6. 输出比较模式 .....	410
24.3.7. PWM 模式 .....	411
24.3.8. 单脉冲模式 .....	412
24.3.9. 定时器同步 .....	413
24.3.10. 调试模式 .....	413
<b>24.4. TIM14 寄存器</b> .....	<b>413</b>
24.4.1. TIM14 控制寄存器 1 (TIM14_CR1) .....	413
24.4.2. TIM14 DMA/中断使能寄存器 (TIM14_DIER) .....	415
24.4.3. TIM14 状态寄存器 (TIM14_SR) .....	415
24.4.4. TIM14 事件产生寄存器 (TIM14_EGR) .....	417
24.4.5. TIM14 捕获/比较模式寄存器 1 (TIM14_CCMR1) .....	417
24.4.6. TIM14 捕获/比较使能寄存器 (TIM14_CCER) .....	420
24.4.7. TIM14 计数器 (TIM14_CNT) .....	421
24.4.8. TIM14 预分频器 (TIM14_PSC) .....	422
24.4.9. TIM14 自动重载寄存器 (TIM14_ARR) .....	422
24.4.10. TIM14 捕获/比较寄存器 1 (TIM14_CCR1) .....	423
24.4.11. TIM14 选项寄存器 (TIMx_OR) .....	424
<b>25. 通用定时器 (TIM15/16/17)</b> .....	<b>425</b>
<b>25.1. 简介</b> .....	<b>425</b>
<b>25.2. TIM15 主要特性</b> .....	<b>425</b>
<b>25.3. TIM16_17 主要特性</b> .....	<b>426</b>

<b>25.4. TIM15_16_17 功能描述</b>	427
25.4.1. 时基单元	427
25.4.2. 计数器模式	428
25.4.3. 重复计数器	431
25.4.4. 时钟源	432
25.4.5. 捕获/比较通道	434
25.4.6. 输入捕获模式	435
25.4.7. PWM 输入模式 (仅 TIM15)	436
25.4.8. 强置输出模式	437
25.4.9. 输出比较模式	437
25.4.10. PWM 模式	438
25.4.11. 互补输出和死区插入	439
25.4.12. 使用刹车功能	440
25.4.13. 单脉冲模式	442
<b>25.5. TIMx 定时器和外部触发的同步 (仅 TIM15)</b>	443
25.5.1. 从模式: 复位模式	444
25.5.2. 从模式: 门控模式	444
25.5.3. 从模式: 触发模式	445
25.5.4. 从模式: 外部时钟模式 2 + 触发模式	446
25.5.5. TIM 和外部的触发同步	446
<b>25.6. 定时器同步 (仅 TIM15)</b>	449
25.6.1. 使用一个定时器作为另一个定时器的预分频器	449
25.6.2. 使用一个定时器使能另一个定时器	450
25.6.3. 使用一个定时器去启动另一个定时器	451
25.6.4. 使用一个外部触发同步地启动 2 个定时器	452
25.6.5. 调试模式	453
<b>25.7. TIM15 寄存器描述</b>	453
25.7.1. TIM15 控制寄存器 1 (TIMx_CR1)	453
25.7.2. TIM15 控制寄存器 2 (TIMx_CR2)	455
25.7.3. TIM15 从模式控制寄存器 (TIMx_SMCR)	456
25.7.4. TIM15 DMA/中断使能寄存器 (TIMx_DIER)	458
25.7.5. TIM15 状态寄存器 (TIMx_SR)	459
25.7.6. TIM15 事件产生寄存器 (TIMx_EGR)	461
25.7.7. TIM15 捕获/比较模式寄存器 1 (TIMx_CCMR1)	462
25.7.8. TIM15 捕获/比较使能寄存器 (TIMx_CCER)	466
25.7.9. TIM15 计数器 (TIMx_CNT)	469
25.7.10. TIM15 预分频器 (TIMx_PSC)	469
25.7.11. TIM15 自动重载寄存器 (TIMx_ARR)	469
25.7.12. TIM15 重复计数器寄存器 (TIMx_RCR)	470

25.7.13.	TIM15 捕获/比较寄存器 1 (TIMx_CCR1)	470
25.7.14.	TIM15 捕捉/比较寄存器 2 (TIMx_CCR2)	471
25.7.15.	TIM15 刹车和死区寄存器 (TIMx_BDTR)	471
25.7.16.	TIM15 DMA 控制寄存器 (TIMx_DCR)	474
25.7.17.	TIM15 连续模式的 DMA 地址 (TIMx_DMAR)	475
<b>25.8.</b>	<b>TIM16_17 寄存器描述</b>	<b>475</b>
25.8.1.	TIM16_17 控制寄存器 1 (TIMx_CR1)	476
25.8.2.	TIM16_17 控制寄存器 2 (TIMx_CR2)	477
25.8.3.	TIM16_17 DMA/中断使能寄存器 (TIMx_DIER)	478
25.8.4.	TIM16_17 状态寄存器 (TIMx_SR)	479
25.8.5.	TIM16_17 事件产生寄存器 (TIMx_EGR)	480
25.8.6.	TIM16_17 捕获/比较模式寄存器 1 (TIMx_CCMR1)	481
25.8.7.	TIM16_17 捕获/比较使能寄存器 (TIMx_CCER)	484
25.8.8.	TIM16_17 计数器 (TIMx_CNT)	487
25.8.9.	TIM16_17 预分频器 (TIMx_PSC)	487
25.8.10.	TIM16_17 自动重载寄存器 (TIMx_ARR)	487
25.8.11.	TIM16_17 重复计数器寄存器 (TIMx_RCR)	488
25.8.12.	TIM16_17 捕获/比较寄存器 1 (TIMx_CCR1)	488
25.8.13.	TIM16_17 刹车和死区寄存器 (TIMx_BDTR)	489
25.8.14.	TIM16_17 DMA 控制寄存器 (TIMx_DCR)	491
25.8.15.	TIM16_17 连续模式的 DMA 地址 (TIMx_DMAR)	493
<b>26.</b>	<b>低功耗定时器 (LPTIM)</b>	<b>494</b>
<b>26.1.</b>	<b>简介</b>	<b>494</b>
<b>26.2.</b>	<b>LPTIM 主要特性</b>	<b>494</b>
<b>26.3.</b>	<b>低功耗定时器 (LPTIM) 功能描述</b>	<b>494</b>
26.3.1.	LPTIM 框图	494
26.3.2.	LPTIM 引脚和内部信号	495
26.3.3.	LPTIM 复位和时钟	495
26.3.4.	预分频器	495
26.3.5.	工作模式	495
26.3.6.	寄存器更新	496
26.3.7.	计数器模式	496
26.3.8.	计数器复位	496
26.3.9.	调试模式 (debug mode)	496
<b>26.4.</b>	<b>LPTIM 低功耗模式</b>	<b>497</b>
<b>26.5.</b>	<b>LPTIM 中断</b>	<b>497</b>
<b>26.6.</b>	<b>LPTIM 寄存器</b>	<b>497</b>
26.6.1.	LPTIM 中断和状态寄存器 (LPTIM_ISR)	497
26.6.2.	LPTIM 中断清除寄存器 (LPTIM_ICR)	498



26.6.3.	LPTIM 中断使能寄存器 (LPTIM_IER)	498
26.6.4.	LPTIM 配置寄存器 (LPTIM_CFGR)	499
26.6.5.	LPTIM 控制寄存器 (LPTIM_CR)	500
26.6.6.	LPTIM 自动重载寄存器 (LPTIM_ARR)	501
26.6.7.	LPTIM 计数寄存器 (LPTIM_CNT)	501
<b>27.</b>	<b>独立看门狗 (IWDG)</b>	<b>503</b>
27.1.	简介	503
27.2.	IWDG 主要特性	503
27.3.	IWDG 功能描述	503
27.3.1.	IWDG 框图	503
27.3.2.	硬件看门狗	504
27.3.3.	寄存器保护	504
27.3.4.	调试模式与 STOP 模式	504
27.4.	IWDG 寄存器	504
27.4.1.	密钥寄存器 (IWDG_KR)	504
27.4.2.	预分频寄存器 (IWDG_PR)	505
27.4.3.	重载寄存器 (IWDG_RLR)	506
27.4.4.	状态寄存器 (IWDG_SR)	506
<b>28.</b>	<b>窗口看门狗 (WWDG)</b>	<b>508</b>
28.1.	简介	508
28.2.	WWDG 主要特性	508
28.3.	WWDG 功能描述	508
28.3.1.	WWDG 架构框图	509
28.3.2.	启动看门狗	509
28.3.3.	控制递减计数器	509
28.3.4.	高级看门狗中断功能	509
28.3.5.	如何编写看门狗超时程序	510
28.3.6.	调试模式	510
28.4.	WWDG 寄存器	510
28.4.1.	控制寄存器 (WWDG_CR)	510
28.4.2.	配置寄存器 (WWDG_CFR)	511
28.4.3.	状态寄存器 (WWDG_SR)	512
<b>29.</b>	<b>实时时钟 (RTC)</b>	<b>513</b>
29.1.	简介	513
29.2.	RTC 主要特性	513
29.3.	RTC 功能描述	513
29.3.1.	总览	513
29.3.2.	复位 RTC 寄存器	514
29.3.3.	读 RTC 寄存器	514

29.3.4.	配置 RTC 寄存器 .....	515
29.3.5.	RTC 标志的设置 .....	515
29.3.6.	RTC 校准 .....	516
<b>29.4.</b>	<b>RTC 寄存器 .....</b>	<b>517</b>
29.4.1.	RTC 控制寄存器 (RTC_CRH) .....	517
29.4.2.	RTC 控制寄存器 (RTC_CRL) .....	518
29.4.3.	RTC 预分频器装载寄存器 (RTC_PRLH) .....	520
29.4.4.	RTC 预分频器分频寄存器 (RTC_PRL) .....	520
29.4.5.	RTC 预分频分频因子寄存器高位 (RTC_DIVH) .....	521
29.4.6.	RTC 预分频分频因子寄存器低位 (RTC_DIVL) .....	521
29.4.7.	RTC 计数寄存器高位 (RTC_CNTH) .....	521
29.4.8.	RTC 计数寄存器低位 (RTC_CNTL) .....	522
29.4.9.	RTC 闹钟寄存器高位 (RTC_ALRH) .....	522
29.4.10.	RTC 闹钟寄存器低位 (RTC_ALRL) .....	523
29.4.11.	RTC 时钟校准及输出配置寄存器 (BKP_RTCCR) .....	523
<b>30.</b>	<b>I<sup>2</sup>C 接口 .....</b>	<b>525</b>
<b>30.1.</b>	<b>介绍 .....</b>	<b>525</b>
<b>30.2.</b>	<b>I<sup>2</sup>C 主要特点 .....</b>	<b>525</b>
<b>30.3.</b>	<b>I<sup>2</sup>C 功能描述 .....</b>	<b>526</b>
30.3.1.	I <sup>2</sup> C 框图 .....	526
30.3.2.	模式选择 .....	526
30.3.3.	I <sup>2</sup> C 初始化 .....	527
30.3.4.	I <sup>2</sup> C 从模式 .....	527
30.3.5.	I <sup>2</sup> C 主模式 .....	529
30.3.6.	错误状态 .....	535
30.3.7.	SDA/SCL 控制 .....	536
30.3.8.	DMA 请求 .....	536
30.3.9.	SMBus .....	537
<b>30.4.</b>	<b>I<sup>2</sup>C 中断 .....</b>	<b>539</b>
<b>30.5.</b>	<b>I<sup>2</sup>C 寄存器 .....</b>	<b>539</b>
30.5.1.	I <sup>2</sup> C 控制寄存器 1 (I2C_CR1) .....	540
30.5.2.	I <sup>2</sup> C 控制寄存器 2 (I2C_CR2) .....	542
30.5.3.	I <sup>2</sup> C 自身地址寄存器 1 (I2C_OAR1) .....	544
30.5.4.	I <sup>2</sup> C 自身地址寄存器 2 (I2C_OAR2) .....	544
30.5.5.	I <sup>2</sup> C 数据寄存器 (I2C_DR) .....	545
30.5.6.	I <sup>2</sup> C 状态寄存器 1 (I2C_SR1) .....	545
30.5.7.	I <sup>2</sup> C 状态寄存器 2 (I2C_SR2) .....	549
30.5.8.	I <sup>2</sup> C 时钟控制寄存器 (I2C_CCR) .....	551
30.5.9.	I <sup>2</sup> C TRISE 寄存器 (I2C_TRISE) .....	552

<b>31. 串行外设接口 (SPI)</b> .....	<b>553</b>
<b>31.1. 简介</b> .....	553
<b>31.2. SPI 主要特征</b> .....	553
31.2.1. SPI 主要特征 .....	553
31.2.2. I <sup>2</sup> S 主要特征.....	554
<b>31.3. SPI 功能描述</b> .....	555
31.3.1. 概述 .....	555
31.3.2. 单主机和单从机通信 .....	555
31.3.3. 多从机通信.....	558
31.3.4. 多主机通信.....	558
31.3.5. 从选择 (NSS) 脚管理 .....	559
31.3.6. 通讯格式 .....	560
31.3.7. SPI 配置.....	561
31.3.8. SPI 使能流程 .....	562
31.3.9. 数据传输和接收流程 .....	562
31.3.10. 状态标志 .....	565
31.3.11. 错误标志 .....	566
31.3.12. SPI 中断.....	567
31.3.13. CRC 计算.....	567
<b>31.4. I<sup>2</sup>S 功能描述</b> .....	569
31.4.1. 概述 .....	569
31.4.2. 音频协议 .....	570
31.4.3. 时钟发生器.....	579
31.4.4. I <sup>2</sup> S 主模式 .....	580
31.4.5. I <sup>2</sup> S 从模式 .....	582
31.4.6. 错误标志位.....	583
31.4.7. I <sup>2</sup> S 中断.....	584
31.4.8. DMA 功能 .....	584
<b>31.5. SPI 和 I<sup>2</sup>S 寄存器</b> .....	584
31.5.1. SPI 控制寄存器 1 (SPI_CR1) (I <sup>2</sup> S 模式下不使用) .....	584
31.5.2. SPI 控制寄存器 2 (SPI_CR2) .....	587
31.5.3. SPI 状态寄存器 (SPI_SR) .....	588
31.5.4. SPI 数据寄存器 (SPI_DR) .....	589
31.5.5. SPI CRC 多项式寄存器 (SPI_CRCPR) .....	590
31.5.6. SPI Rx CRC 寄存器 (SPI_RXCRCR) .....	590
31.5.7. SPI Tx CRC 寄存器 (SPI_TXCRCR) .....	591
31.5.8. SPI_I <sup>2</sup> S 配置寄存器 (SPI_I <sup>2</sup> S_CFGR) .....	591
31.5.9. SPI_I <sup>2</sup> S 预分频寄存器 (SPI_I <sup>2</sup> S_SPR) .....	593
<b>32. 通用同步异步收发器 (USART)</b> .....	<b>595</b>

<b>32.1. 介绍</b> .....	595
<b>32.2. USART 主要特性</b> .....	595
<b>32.3. USART 功能描述</b> .....	596
32.3.1. USART 特征描述 .....	597
32.3.2. 发送器 .....	598
32.3.3. 接收器 .....	600
32.3.4. 分数波特率的产生 .....	604
32.3.5. USART 接收器容忍度 .....	605
32.3.6. USART 自动波特率检测 .....	605
32.3.7. 多处理器通信 .....	606
32.3.8. LIN（局域互联网）模式 .....	608
32.3.9. USART 同步模式 .....	609
32.3.10. 单线半双工通信 .....	611
32.3.11. 智能卡 .....	611
32.3.12. IrDA SIR ENDEC 功能模块 .....	613
32.3.13. 用 DMA 连续通信 .....	614
32.3.14. 硬件流控制 .....	616
<b>32.4. USART 中断请求</b> .....	617
<b>32.5. USART 寄存器</b> .....	618
32.5.1. 状态寄存器（USART_SR） .....	618
32.5.2. 数据寄存器（USART_DR） .....	621
32.5.3. 波特率寄存器（USART_BRR） .....	622
32.5.4. 控制寄存器 1（USART_CR1） .....	622
32.5.5. 控制寄存器 2（USART_CR2） .....	624
32.5.6. 控制寄存器 3（USART_CR3） .....	626
32.5.7. 保护时间和预分频器（USART_GTPR） .....	627
<b>33. CAN2.0 控制器</b> .....	629
<b>33.1. 介绍</b> .....	629
<b>33.2. CAN 主要特性</b> .....	629
<b>33.3. CAN 功能描述</b> .....	630
33.3.1. 模块框图 .....	630
33.3.2. 动作模式 .....	630
33.3.3. 波特率设定 .....	630
33.3.4. 发送缓冲器 .....	631
33.3.5. 接收缓冲器 .....	632
33.3.6. 接收筛选寄存器组 .....	633
33.3.7. LLC 帧格式定义 .....	633
33.3.8. 数据发送 .....	635
33.3.9. 取消数据发送 .....	636

33.3.10.	数据接收 .....	636
33.3.11.	错误处理 .....	636
33.3.12.	节点关闭 .....	637
33.3.13.	仲裁失败位置捕捉 .....	637
33.3.14.	回环模式 .....	637
33.3.15.	静默模式 .....	638
33.3.16.	软件复位功能 .....	638
33.3.17.	时间触发 TTCAN .....	640
33.3.18.	中断 .....	641
<b>33.4.</b>	<b>寄存器说明 .....</b>	<b>642</b>
33.4.1.	节点配置寄存器 (CAN_TSNCR) .....	642
33.4.2.	位时序配置寄存器 (CAN_ACBTR) .....	643
33.4.3.	限制与预分频配置寄存器 (CAN_RLSSP) .....	643
33.4.4.	状态寄存器 (CAN_IFR) .....	644
33.4.5.	中断使能寄存器 (CAN_IER) .....	646
33.4.6.	传输状态寄存器 (CAN_TSR) .....	648
33.4.7.	全局配置寄存器 (CAN_MCR) .....	648
33.4.8.	错误警告寄存器 (CAN_WECR) .....	653
33.4.9.	参考 ID 寄存器 (CAN_REFMSG) .....	654
33.4.10.	TTCAN 配置寄存器 (CAN_TTCR) .....	655
33.4.11.	TTCAN 触发寄存器 (CAN_TTTR) .....	657
33.4.12.	内存状态寄存器 (CAN_SCMS) .....	657
33.4.13.	筛选器组控制寄存器 (CAN_ACFR) .....	658
33.4.14.	筛选器组 code 寄存器 (CAN_ACFC) .....	659
33.4.15.	筛选器组 mask 寄存器 (CAN_ACFM) .....	659
33.4.16.	CAN 接收 BUF 寄存器 (CAN_RBUF) .....	660
33.4.17.	CAN 发送 BUF 寄存器 (CAN_TBUF) .....	660
<b>34.</b>	<b>USB 全速设备接口 (USB) .....</b>	<b>661</b>
34.1.	简介 .....	661
34.2.	USB 主要特征 .....	661
34.3.	USB 设备框图 .....	661
34.4.	功能描述 .....	661
34.4.1.	功能模块描述 .....	662
34.4.2.	系统复位和上电复位 .....	662
34.4.3.	USB 复位状态 .....	662
34.4.4.	USB 挂起/唤醒模式 .....	662
34.4.5.	IN 分组 (用于数据发送) .....	663
34.4.6.	OUT 分组 (用于数据接收) .....	663
34.4.7.	控制传输 .....	664

34.4.8.	同步传输 .....	666
34.4.9.	批量传输 .....	669
34.4.10.	中断传输 .....	672
<b>34.5.</b>	<b>USB 寄存器 .....</b>	<b>672</b>
34.5.1.	USB 控制寄存器 (USB_CR) .....	672
34.5.2.	USB 中断状态寄存器 (USB_INTR) .....	674
34.5.3.	USB 中断使能寄存器 (USB_INTRE) .....	674
34.5.4.	USB 帧寄存器 (USB_FRAME) .....	675
34.5.5.	USB 端点 0 控制寄存器 (USB_EP0CSR) .....	676
34.5.6.	USB IN 端点控制寄存器 (USB_INEPxCSR) .....	676
34.5.7.	USB OUT 端点控制寄存器 (USB_OUTEPxCSR) .....	678
34.5.8.	USB OUT 端点计数寄存器 (USB_OUTCOUNT) .....	679
34.5.9.	USB FIFO 寄存器 (USB_FIFO) .....	679
<b>35.</b>	<b>调试支持 .....</b>	<b>680</b>
<b>35.1.</b>	<b>概况 .....</b>	<b>680</b>
<b>35.2.</b>	<b>引脚分布和调试端口脚 .....</b>	<b>680</b>
35.2.1.	SWD 调试端口 .....	680
35.2.2.	灵活的 SW-DP 脚分配 .....	681
35.2.3.	<b>SWD 脚上的内部上拉和下拉 .....</b>	<b>681</b>
<b>35.3.</b>	<b>ID 代码和锁定机制 .....</b>	<b>681</b>
<b>35.4.</b>	<b>SWD 调试端口 .....</b>	<b>681</b>
35.4.1.	SWD 协议介绍 .....	681
35.4.2.	SWD 协议序列 .....	681
35.4.3.	SW-DP 状态机 (reset, idle states, ID code) .....	682
35.4.4.	DP and AP 读/写访问 .....	682
35.4.5.	SW-DP 寄存器 .....	683
35.4.6.	SW-AP 寄存器 .....	683
<b>35.5.</b>	<b>内核调试 .....</b>	<b>684</b>
<b>35.6.</b>	<b>BPU 断点单元 (Break Point Unit) .....</b>	<b>684</b>
35.6.1.	BPU 功能 .....	684
<b>35.7.</b>	<b>数据观察点 DWT (Data Watchpoint) .....</b>	<b>684</b>
35.7.1.	DWT 功能 .....	684
35.7.2.	DWT 程序计数器样本寄存器 .....	684
<b>35.8.</b>	<b>MCU 调试模块 (DBGMCU) .....</b>	<b>685</b>
35.8.1.	低功耗模式的调试支持 .....	685
35.8.2.	支持定时器、看门狗、CAN 和 I <sup>2</sup> C 的调试 .....	685
<b>35.9.</b>	<b>DBG 寄存器 .....</b>	<b>685</b>
35.9.1.	DBG 设备 ID 代码寄存器 (DBG_IDCODE) .....	685
35.9.2.	调试 MCU 配置寄存器 (DBGMCU_CR) .....	686

---

35.9.3. DBG APB 冻结寄存器 1 (DBG_APB_FZ1) .....	687
35.9.4. DBG APB 冻结寄存器 2 (DBG_APB_FZ2) .....	688
<b>36. 版本历史.....</b>	<b>690</b>

## 1. 寄存器描述中使用的缩写列表

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



## 2. 系统架构框图

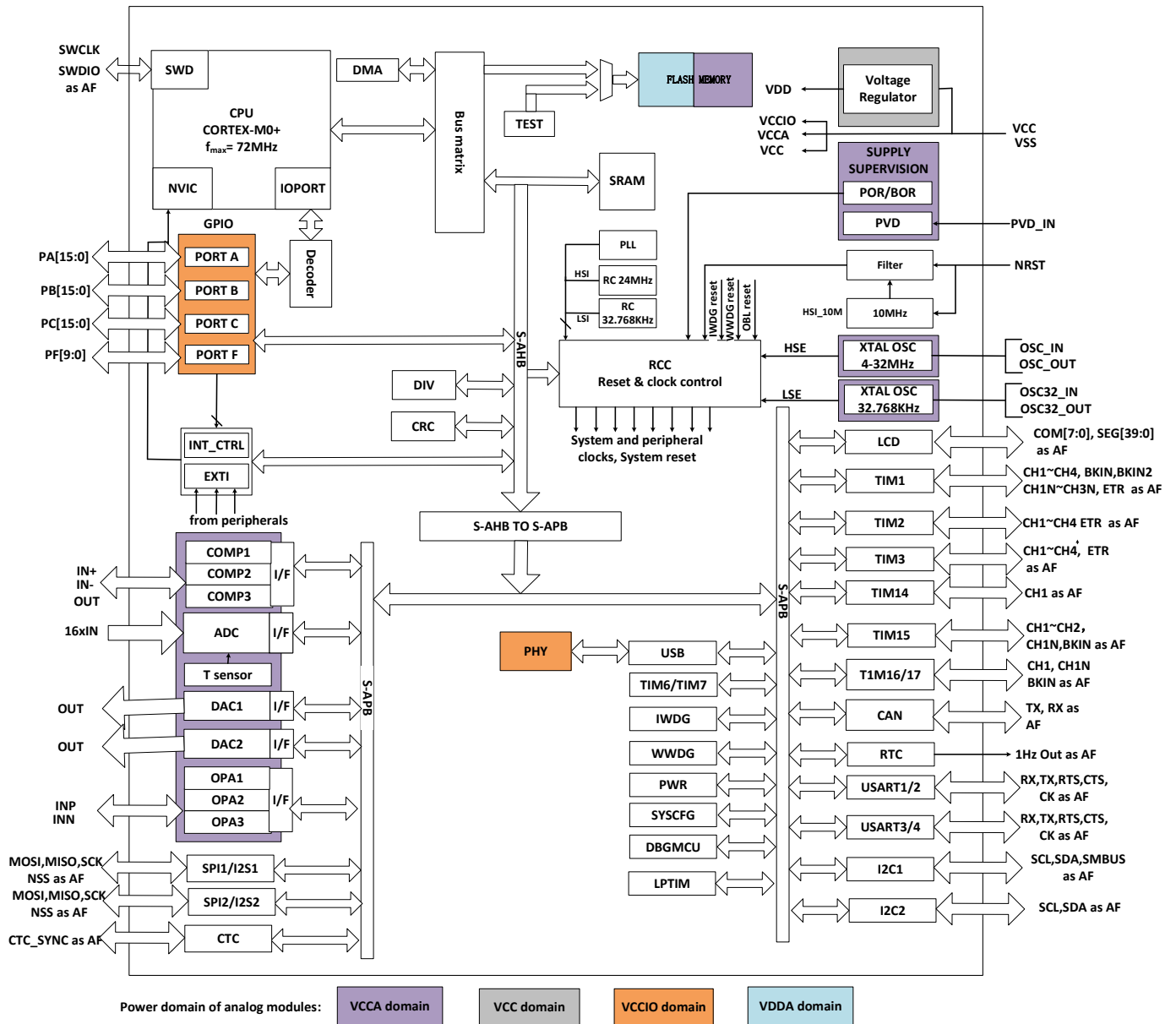


图 2-1 系统架构框图

## 3. 存储器和总线架构

### 3.1. 系统架构

系统由以下部分组成：

- 两个 Master
  - Cortex-M0+
  - 通用 DMA
- 三个 Slave
  - 内部 SRAM
  - 内部 Flash
  - 带 AHB-APB 桥的 AHB

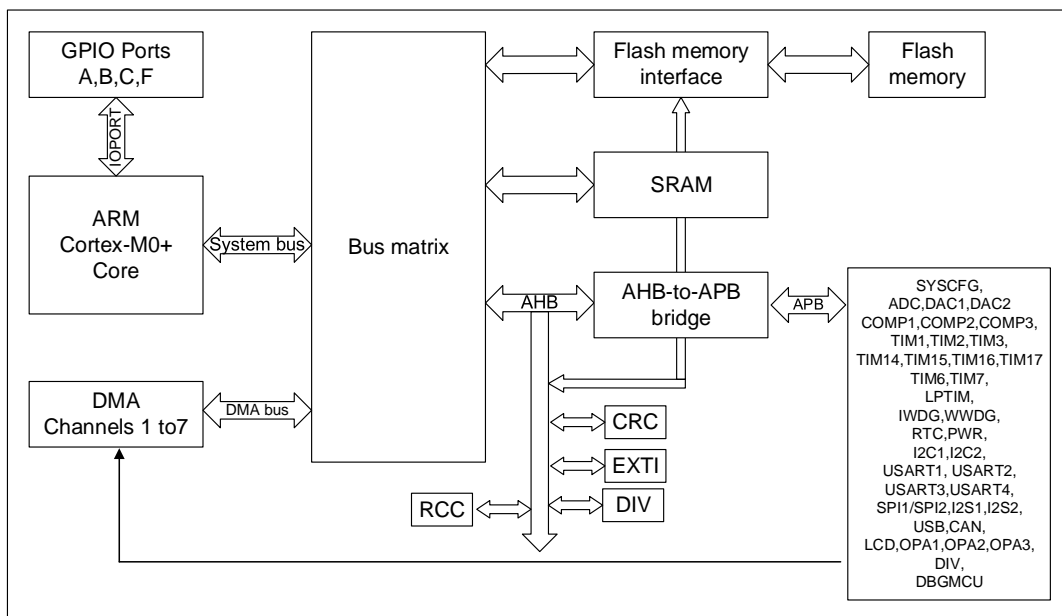


图 3-1 系统架构

#### ■ CPU 系统总线

该总线把 Cortex-M0+ 的系统总线连接到总线矩阵。

#### ■ DMA 总线

该总线把 DMA 的 AHB master 接口连接到总线矩阵，由总线矩阵管理 CPU 和 DMA 对 SRAM、Flash 存储器和 AHB/APB 的外设访问。

#### ■ 总线矩阵

总线矩阵管理在 CPU 总线和 DMA 总线之间的访问仲裁。该仲裁使用 Round Robin 算法。总线矩阵由 Master (CPU、DMA) 和 Slaves (Flash Memory、SRAM 和 AHB-to-APB bridge) 组成。

#### ■ AHB-to-APB Bridge (APB)

AHB-to-APB 桥提供了在 AHB 和 APB 总线之间的同步连接，桥的外设地址映射参考 [表 3-2 外设寄存器地址](#)。

## 3.2. 存储器结构

### 3.2.1. 存储器结构简介

程序存储器、数据存储器、寄存器和 IO 端口被统一编址在一个线性 4 GB 空间。该地址以小端编码形式存在（一个 word 中，最低字节分配在最低地址）。

整个寻址空间被划分成 8 个 512 MB 的 Block 区域。

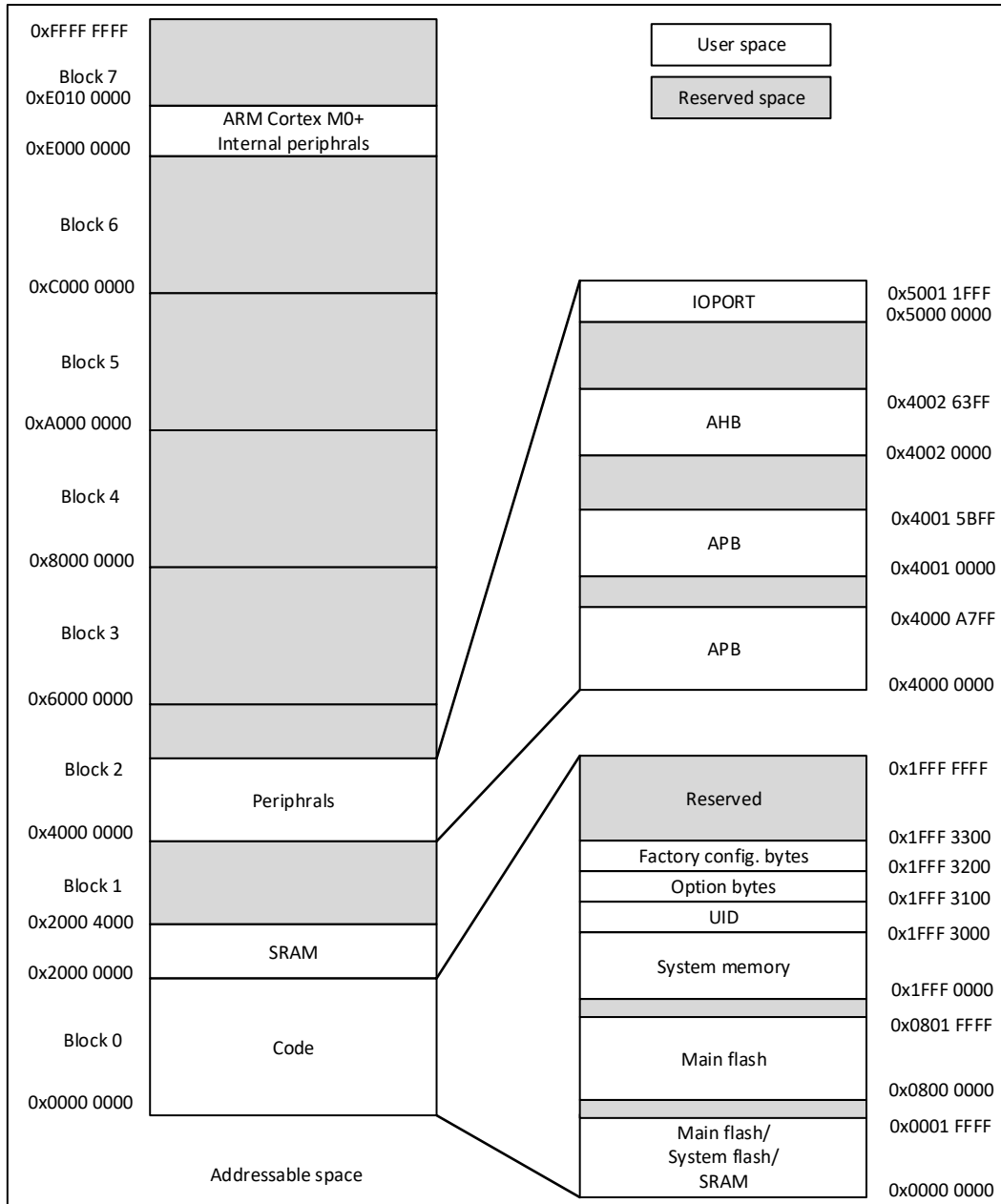


图 3-2 存储器映射

表 3-1 存储器地址

Type	Boundary address	Size	Memory area	Description
SRAM	0x2000 4000-0x3FFF FFFF	-	保留 <sup>(1)</sup>	-
	0x2000 0000-0x2000 3FFF	16 KB	SRAM	SRAM 最大为 16 KB
Code	0x1FFF 3300-0x1FFF FFFF	-	保留	-
	0x1FFF 3200-0x1FFF 32FF	256 Bytes	FT info0 bytes	Factory config.bytes
	0x1FFF 3100-0x1FFF 31FF	256 Bytes	Option bytes	芯片软硬件 option bytes 信息
	0x1FFF 3000-0x1FFF 30FF	256 Bytes	UID bytes	Unique ID
	0x1FFF 0000-0x1FFF 2FFF	12 KB	System memory	存放 boot loader
	0x0802 0000-0x1FFE FFFF	-	保留	-
	0x0800 0000-0x0801 FFFF	128 KB	Main flash memory	-
	0x0002 0000-0x07FF FFFF	-	保留	-
	0x0000 0000-0x0001 FFFF	128 KB	根据 Boot 配置选择： 1) Main flash memory 2) System memory 3) SRAM	-

表 3-2 外设寄存器地址

总线	地址范围	容量	外设
	0xE000 0000-0xE00F FFFF	-	M0+
IOPORT	0x5000 1800-0x5FFF FFFF	-	保留
	0x5000 1400-0x5000 17FF	1 KB	GPIOF
	0x5000 1000-0x5000 13FF	-	保留
	0x5000 0C00-0x5000 0FFF	-	保留
	0x5000 0800-0x5000 0BFF	1 KB	GPIOC
	0x5000 0400-0x5000 07FF	1 KB	GPIOB
	0x5000 0000-0x5000 03FF	1 KB	GPIOA
AHB	0x4002 4000-0x4FFF FFFF	-	保留
	0x4002 3C00-0x4002 3FFF	-	保留
	0x4002 3800-0x4002 3BFF	1 KB	DIV
	0x4002 3400-0x4002 37FF	-	保留
	0x4002 3000-0x4002 33FF	1 KB	CRC
	0x4002 2400-0x4002 2FFF	-	保留

总线	地址范围	容量	外设
	0x4002 2000-0x4002 23FF	1 KB	Flash
	0x4002 1C00-0x4002 1FFF	-	保留
	0x4002 1800-0x4002 1BFF	1 KB	EXTI
	0x4002 1400-0x4002 17FF	1 KB	保留
	0x4002 1000-0x4002 13FF	1 KB	RCC
	0x4002 0400-0x4002 0FFF	-	保留
	0x4002 0000-0x4002 03FF	1 KB	DMA
APB	0x4001 5C00 - 0x4001 FFFF	-	保留
	0x4001 5800 - 0x4001 5BFF	1 KB	DBG
	0x4001 4C00 - 0x4001 57FF	-	保留
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15
	0x4001 3C00 - 0x4001 3FFF	-	保留
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1
	0x4001 3400 - 0x4001 37FF	1 KB	保留
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1
	0x4001 2800 - 0x4001 2BFF	-	保留
	0x4001 2400 - 0x4001 27FF	1 KB	ADC
	0x4001 0400 - 0x4001 23FF	-	保留
	0x4001 0300 - 0x4001 03FF	1 KB	OPA
	0x4001 0200 - 0x4001 02FF		COMP
	0x4001 0000 - 0x4001 01FF		SYSCFG
	0x4000 8000- 0x4000 FFFF	-	保留
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1
	0x4000 7800 - 0x4000 7BFF	-	保留
	0x4000 7400 - 0x4000 77FF	1 KB	DAC
	0x4000 7000 - 0x4000 73FF	1 KB	PWR
	0x4000 6C00 - 0x4000 6FFF	1 KB	CTC
	0x4000 6800 - 0x4000 6BFF	-	保留
	0x4000 6400 - 0x4000 67FF	1 KB	CAN
	0x4000 6000 - 0x4000 63FF	1 KB	USB SRAM

总线	地址范围	容量	外设
	0x4000 5C00 - 0x4000 5FFF	1 KB	USB
	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1
	0x4000 5000 - 0x4000 53FF	-	保留
	0x4000 4C00 - 0x4000 4FFF	1 KB	USART4
	0x4000 4800 - 0x4000 4BFF	1 KB	USART3
	0x4000 4400 - 0x4000 47FF	1 KB	USART2
	0x4000 3C00 - 0x4000 43FF	2 KB	保留
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2/I2S2
	0x4000 3400 - 0x4000 37FF	-	保留
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC
	0x4000 2400 - 0x4000 27FF	1 KB	LCD
	0x4000 2000 - 0x4000 23FF	1 KB	TIM14
	0x4000 1800 - 0x4000 1FFF	-	保留
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6
	0x4000 0800 - 0x4000 0FFF	-	保留
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3
	0x4000 0000 - 0x4000 03FF	1 KB	TIM2

### 3.3. 嵌入式 SRAM

片内最大集成 16 KB SRAM。通过字节、半字（16bit）或全字（32bit）的方式可访问 SRAM。软件对设定范围外空间的读写操作，会产生 HardFault。

Flash 存储器有两个不同的物理区域组成：

- Main flash 区域，128 KB，它包含应用程序和用户数据。软件对设定范围外空间的访问会产生 HardFault。
- Information 区域，14 KB，它包括以下部分：
  - FT info0 Bytes：256 Bytes，用于存放 TS DATA，HSI Clock Re-trimming 数据
  - 选项字节：256 Bytes，用于存放芯片软硬件 Option 的配置信息
  - UID：256 Bytes，用于存放芯片的 UID
  - System memory（系统存储器）：11.75 KB，用于存放 Boot loader。

Flash controller 实现基于 AHB 协议的指令读取和数据访问，它也通过寄存器实现了 Flash 的基本 Program/擦除等操作。

### 3.4. Boot 模式

通过 BOOT0 pin 和 boot 配置位 nBOOT1（存放于选项字节中），可选择三种不同的启动模式，如下表所示：

表 3-3 Boot 配置

Boot mode configuration		Mode
nBOOT1 bit	BOOT0 pin	
X	0	选择 Main flash 作为启动区
1	1	选择 System memory 作为启动区
0	1	选择 SRAM 作为启动区

在该 Startup 延迟后，CPU 从地址 0x0000 0000 取堆栈顶的值，然后从启动存储器的 0x0000 0004 地址开始执行指令。取决于被选择的启动模式，Main flash、System memory 或者 SRAM 按照如下进行访问：

- Boot from main flash: Main flash 跟启动存储器空间的 0x0000 0000 对齐，但是仍然可以按其本来的存储器空间 (0x0800 0000) 进行访问。也就是说，Flash 空间可以从地址 0x0000 0000 或者 0x0800 0000 访问到。
- Boot from system memory: System memory 对齐在启动存储器空间 0x0000 0000，但是仍然可以从它本来的地址空间 0x1FFF 0000 访问到。
- Boot from SRAM: SRAM 对齐在启动存储器空间的 0x0000 0000，但是仍然可以通过 0x2000 0000 地址访问到。

#### 3.4.1. 存储器物理映像

选择启动模式之后，应用软件可以修改在程序空间可被访问的存储器。这个修改通过 SYSCFG\_CFGR1 寄存器的 MEM\_MODE 位选择决定（详见 SYSCFG 章节）。

#### 3.4.2. 内嵌的自举程序

Bootloader 在芯片生产阶段被写入，并存放在 system memory 中。它用来使用下面串行接口对 Flash 存储器的再次写入：

- USART1, PA9/PA10
- USB, PA11/PA12

## 4. 嵌入式闪存

### 4.1. 闪存主要特性

- Main flash block: 最大 128 KB
- Information block: 14 KB
- Page size: 256 Bytes
- Sector size: 8 KB

闪存控制接口电路的主要特征如下:

- 闪存写和擦除
- 选项字节的编程操作
- 读保护
- 写保护

### 4.2. 闪存功能介绍

#### 4.2.1. 闪存结构

Flash 存储器由 64 位宽的存储单元组成, 可以用作程序和数据的存储, Page 大小为 256 Bytes, Sector 大小为 8 KB。

从功能上, Flash 存储器分为 Main flash 和 Information flash, 前者容量最大是 128 KB, 后者容量为 14 KB。

表 4-1 闪存结构及边界地址

Block	扇区 sector	页 Page	Base address	Size
Main flash	Sector 0	Page 0-31	0x0800 0000-0x0800 1FFF	8 KB
	Sector 1	Page 32-63	0x0800 2000-0x0800 3FFF	8 KB
	Sector 2	Page 64-95	0x0800 4000-0x0800 5FFF	8 KB
	...	...	...	...
	Sector 14	Page 448-479	0x0801 C000-0x0801 DFFF	8 KB
	Sector 15	Page 480-511	0x0801 E000-0x0801 FFFF	8 KB
System flash	INFO	Page 0-46	0x1FFF 0000-0x1FFF 2EFF	11.75 KB
保留		Page 47	0x1FFF 2F00-0x1FFF 2FFF	256 Bytes
UID		Page 48	0x1FFF 3000-0x1FFF 30FF	256 Bytes
选项字节		Page 49	0x1FFF 3100-0x1FFF 31FF	256 Bytes
FT info0		Page 50	0x1FFF 3200-0x1FFF 32FF	256 Bytes
保留		Page 51-55	0x1FFF 3300-0x1FFF 37FF	1280 Bytes



## 4.2.2. 闪存读操作和访问延迟

Flash 可以被作为一个通用的存储器空间，被直接寻址访问。通过专门的读控制时序，可以对 Flash 存储器的内容进行读取。取指和数据访问都是通过 AHB 总线进行的。读操作可以被 FLASH\_ACR 寄存器的 Latency 位控制，即读取 Flash 增加一个或者不增加等待状态。

FLASH\_ACR (LATENCY) 位，当为 0，则不增加 Flash 读操作的等待状态。当为 1，Flash 读操作增加 1 个等待状态。当为 2，Flash 读操作增加 3 个等待状态该机制是为了匹配高速的系统时钟和相对低速的 Flash 读取速度，而进行的专门设计。

## 4.2.3. 闪存写操作和擦除操作

通过电路编程 (ICP, In-circuit programming) 或者应用编程 (IAP, In-application programming) 可以对 Flash 进行 Program 操作。

**ICP:** 用来更新整个 Flash 存储器的内容，可以使用 SWD 协议或者系统加载程序 (Boot loader)，把用户应用装入 MCU 中。ICP 提供了快速和有效的设计迭代，避免了不必要的封装和管座 (socketing)。

**IAP:** 可以使用芯片支持的通讯接口，下载要 Program 的数据到 Flash 中。IAP 允许用户在应用运行时，再次 Program flash 存储器。然后，此时 Flash 存储器中已有了之前使用 ICP 编程进去的部分应用程序。

如果在进行闪存写和擦除操作时，发生了复位，则闪存存储器的内容是不被保护的。

在闪存写和擦除操作期间，任何读闪存的操作都会阻塞总线。写或擦除操作结束后，读操作就可以正确的进行。这也就意味着，当正在进写和擦除操作时，不能进行代码和数据的读取。

对于写和擦除操作，必须打开 HSI。

### 4.2.3.1. 闪存解锁

在复位后，Flash 存储器会被保护，防止非预期的（比如电干扰引起的）写和擦除操作。写 FLASH\_CR 寄存器是不被允许的（除了用作重载选项字节的 OBL\_LAUNCH 位）。每次对 Flash 的写和擦除操作，都必须通过写 FLASH\_KEYR 寄存器，产生 Unlock 时序，启用 FLASH\_CR 寄存器的访问。

具体步骤如下：

步骤 1: 向 FLASH\_KEYR 寄存器写入 KEY1=0x4567 0123

步骤 2: 向 FLASH\_KEYR 寄存器写入 KEY2=0xCDEF 89AB

任何错误的时序都会锁住 FLASH\_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，总线错误被发现，并产生 HardFault 中断。这样的错误包括第一个写周期的 KEY1 不匹配，或者 KEY1 匹配，但第二个写周期的 KEY2 不匹配。

FLASH\_CR 寄存器可以通过软件写 FLASH\_CR 寄存器的 LOCK 位被再次锁住。

另外，当 FLASH\_SR 寄存器的 BSY 位被置位时，FLASH\_CR 寄存器不能被写。此时，任何尝试进行写该寄存器 (FLASH\_CR) 的操作会引起 AHB 总线的阻塞，直到 BSY 位被清零。

#### 4.2.3.2. 闪存写操作

Flash 存储器每次以 32 bits word 为单位（进行半字或字节操作会产生 HardFault）进行整个页（Page）的 Program 操作。当 FLASH\_CR 寄存器的 PG 位被置位，CPU 向 Flash 存储器地址空间写 32 bits 数据时，Program 操作开始启动。任何非 32 bits 的写入将导致 HardFault 中断。

如果要 Program 的 Flash 地址空间，是被 FLASH\_WRPR 寄存器设置为保护的区域，则 Program 操作会被忽略掉，同时 FLASH\_SR 寄存器 WRPERR 位会被置位。Program 操作结束后，FLASH\_SR 寄存器的 EOP 位会被置位。

具体 Flash program 的操作步骤如下所示：

1. 检查 FLASH\_SR 寄存器的 BSY 位，判断是否当前没有正在继续的 Flash 操作
2. 如果没有正在进行的 Flash 擦除或者 Program 操作，则软件读出该 Page 的 64 个 Word（如果该 Page 已有数据存放，则进行该步骤，否则跳过该步骤）
3. 向 FLASH\_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH\_CR 寄存器的保护
4. 置位 FLASH\_CR 寄存器的 PG 位和 EOPIE 位
5. 向目标地址进行第 1 到第 63 个 Word 的 Program 操作（只接受 32 bits 的 Program）
6. 置位 FLASH\_CR 寄存器的 PGSTRT
7. 写第 64 个 Word
8. 等待 FLASH\_SR 寄存器的 BSY 位被清零
9. 检查 FLASH\_SR 寄存器的 EOP 标志位（当 Program 操作已经成功，该位被置位），然后软件清零该位
10. 如果不再有 Program 操作，则软件清除 PG 位

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

#### 4.2.3.3. 闪存擦除操作

Flash 存储器可以按照页进行擦除操作，或者进行扇区擦除（Sector erase）和全片擦除（Mass erase）。

##### 页擦除

当某页被 WRP 保护，它是不会擦除的，此时 WRPERR 位被置位。当要进行页擦除时，要进行以下步骤：

1. 检查 FLASH\_SR 寄存器 BSY 位，确认没有正在进行的 Flash 操作
2. 向 FLASH\_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH\_CR 寄存器的保护
3. 置位 FLASH\_CR 寄存器的 PER 位和 EOPIE 位
4. 向该 Page 写任意数据（必须 32 bits 数据）
5. 等待 BSY 位被清零
6. 检查 EOP 标志位被置位
7. 清零 EOP 标志

##### 片擦除

片擦除用来对整片 Main flash 进行擦除操作。另外，当 WRP 被使能，片擦功能无效，不会产生片擦操作，并且 WRPERR 位被置位。

片擦的步骤如下：

1. 检查 BSY 位，确认是否没有正在进行的 Flash 操作
2. 向 FLASH\_KEYR 寄存器依次写 KEY1, KEY2, 解除 FLASH\_CR 寄存器保护
3. 置位 FLASH\_CR 寄存器的 MER 位和 EOPIE 位
4. 向 Flash 的任意 Main flash 空间写任意数据 (32 bits 数据)
5. 等待 BSY 位被清零
6. 检查 EOP 标志位被置位
7. 清零 EOP 标志

#### 扇区擦除

扇区擦除用来对 8 KB 的 Main flash 进行擦除操作。另外，当某个扇区被 WRP 保护，它是不会被擦除的，此时 WRPERR 位被置位。

扇区擦除的步骤如下：

- 检查 BSY 位，确认是否没有正在进行的 Flash 操作
- 向 FLASH\_KEYR 寄存器依次写 KEY1、KEY2, 解除 FLASH\_CR 寄存器保护
- 置位 FLASH\_CR 寄存器的 SER 位和 EOPIE 位
- 向该 Sector 写任意数据
- 等待 BSY 位被清零
- 检查 EOP 标志位被置位
- 清零 EOP 标志

#### 4.2.3.4. 写和擦除时间配置

Flash 的 Program 和擦除的时间需要进行严格的控制，否则会造成操作失败。上电默认情况，硬件设计将 Program 和擦除操作的时间参数，设置成 HSI 为 24 MHz 的参数。当更改了 HSI 输出频率，需要根据下表对 Flash program 和擦除时间控制寄存器进行正确的配置。

表 4-2 写和擦除时间配置

寄存器	4 MHz	8 MHz	16 MHz	22.12 MHz	24 MHz
TS0	0x1E	0x3C	0x78	0xA6	0xB4
TS1	0x48	0x90	0x120	0x18F	0x1B0
TS2P	0x1E	0x3C	0x78	0xA6	0xB4
TPS3	0x120	0x240	0x480	0x639	0x6C0
TS3	0x1E	0x3C	0x78	0xA6	0xB4
PERTPE	0x36B0	0x6D60	0xDAC0	0x12E6C	0x14820
SMERTPE	0x36B0	0x6D60	0xDAC0	0x12E6C	0x14820
PRGTPE	0XFA0	0x1F40	0x3E80	0x5668	0x5DC0
PRETPE	0x320	0x640	0xC80	0x1148	0x12C0

## 4.3. Flash 选项字节

### 4.3.1. Flash 选项字

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

为了数据的安全性，选项字节以正文及反码形式分别存储。

表 4-3 选项字节格式

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

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

- Flash 选项寄存器 (FLASH\_OPTR)
- Flash BORCR 地址寄存器 (FLASH\_BORCR)
- Flash WRP 地址寄存器 (FLASH\_WRPR)

表 4-4 选项字节结构

Word 地址	描述
0x1FFF 3100	Flash 用户选项字节及其补码
0x1FFF 3108	BOR 控制及其补码的选项字节
0x1FFF 3110	保留
0x1FFF 3118	Flash WRP 地址及其补码的选项字节
0x1FFF 3120	保留
0x1FFF 3128	保留
...	保留
...	保留
...	保留
0x1FFF 31F8	保留

#### 4.3.1.1. Flash 用户选项及其补码的选项字节

Flash memory address: 0x1FFF 3100

Production value: 0x2755 D8AA

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~IWDG_STOP	~nBOOT1	~NRST_MODE	~WWDG_SW	~IWDG_SW	Res	Res	Res	~RDP[7: 0]							
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

IWDG_STOP	nBOOT1	NRST_MODE	WWDG_SW	IWDG_SW	Res	Res	Res	RDP[7: 0]							
R	R	R	R	R	-	-	-	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	~IWDG_STOP	R	IWDG_STOP 的反码
30	~ nBOOT1	R	nBOOT1 的反码
29	~ NRST_MODE	R	NRST_MODE 的反码
28	~ WWDG_SW	R	WWDG_SW 的反码
27	~ IWDG_SW	R	IWDG_SW 的反码
26:24	保留	-	保留
23:16	~RDP	R	RDP 的反码
15	IWDG_STOP	R	与设置 IWDG 在 stop 模式下定时器运行状态 0: 冻结定时器 1: 正常运行
14	nBOOT1	R	与 BOOT PIN 一起, 选择芯片启动模式
13	NRST_MODE	R	0: 仅复位输入 1: GPIO 功能
12	WWDG_SW	R	0: 硬件看门狗 1: 软件看门狗
11	IWDG_SW	R	0: 硬件看门狗 1: 软件看门狗
10: 8	保留	-	保留
7: 0	RDP	R	0xAA: Level 0, 读保护无效 非 0xAA: Level 1, 读保护有效

#### 4.3.1.2. BOR 控制及其补码的选项字节

Flash memory address: 0x1FFF 3108

Production value: 0xFFE0 0000

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

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

Bit	Name	R/W	Function
31: 29	~BOR_LEV[2: 0]	R	BOR_LEV[2: 0]的反码

Bit	Name	R/W	Function
28: 22	保留	-	保留
21	~BOR_EN	R	BOR_EN 的反码
20: 16	保留	-	保留
15: 13	BOR_LEV[2: 0]	R	000: BOR 上升阈值为 1.8 V, 下降阈值为 1.7 V 001: BOR 上升阈值为 2.0 V, 下降阈值为 1.9 V 010: BOR 上升阈值为 2.2 V, 下降阈值为 2.1 V 011: BOR 上升阈值为 2.4 V, 下降阈值为 2.3 V 100: BOR 上升阈值为 2.6 V, 下降阈值为 2.5 V 101: BOR 上升阈值为 2.8 V, 下降阈值为 2.7 V 110: BOR 上升阈值为 3.0 V, 下降阈值为 2.9 V 111: BOR 上升阈值为 3.2 V, 下降阈值为 3.1 V
12: 6	保留	-	保留
5	BOR_EN	R	BOR 使能 0: BOR 不使能 1: BOR 使能, BOR_LEV 起作用
4: 0	保留	-	保留

#### 4.3.1.3. Flash WRP 地址及其补码的选项字节

Flash memory address: 0x1FFF 3118

Production value: 0x0000 FFFF

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

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

Bit	Name	R/W	Function
31: 16	~WRP	R	WRP 的反码
15: 0	WRP	R	0: Sector[y]被保护 1: Sector[y]无保护 y=0~15

#### 4.3.2. Flash 选项字节写

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

以下步骤用来解锁该寄存器：

1. 通过 Unlock 时序，Unlock FLASH\_CR 寄存器的写保护
2. 向 FLASH\_OPTKEYR 寄存器，写 OPTKEY1=0x0819 2A3B
3. 向 FLASH\_OPTKEYR 寄存器，写 OPTKEY2=0x4C5D 6E7F

任何错误的时序都会 Lock 住 FLASH\_CR 寄存器，直到下一次复位。在错误的 KEY 时序时，总线错误被发现，并产生 HardFault 中断。

用户选项（Information flash 的选项字节）可以通过软件写 FLASH\_CR 寄存器的 OPTLOCK 位，被保护住，以防止不想要的擦除/Program 操作。

如果软件置位 Lock 位，则 OPTLOCK 位也被自动置位。

#### 4.3.2.1. 修改用户选项字节

选项字节的写操作（Program），跟对 Main flash 的操作不一样。为修改选项字节，需要进行如下步骤：

- 用之前描述的步骤，清零 OPTLOCK 位
- 检查 BSY 位，确认没有正在进行的 Flash 操作
- 向选项字节寄存器 FLASH\_OPTR/FLASH\_BORCR/FLASH\_WRPR 写期望的值（1~3 个 word）
- 置位 OPTSTRT 位
- 向 Main flash 0x4002 2080 地址写任意 32 bit 数据（触发正式的写操作）
- 等待 BSY 位被清零
- 等待 EOP 拉高
- 软件清零 EOP

任何对选项字节的改动，硬件都会先把选项字节对应的整个页擦除掉，然后用 FLASH\_OPTR、FLASH\_BORCR 或者 FLASH\_WRPR 寄存器的值，写到选项字节中。并且，硬件自动计算相应的补码，并把计算值写到选项字节的相应区域。

#### 4.3.2.2. 重新加载选项字节

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

选项字节的装载，在以下两种情况下进行：

1. 当 FLASH\_CR 寄存器中的 OBL\_LAUNCH 位被置位
2. 在上电复位后（POR、BOR）

“装载选项字节”进行的操作是：对 Information memory 区域的选项字节进行读操作，再把读出的数据存储在内部选项寄存器中（FLASH\_OPTR、FLASH\_BORCR 和 FLASH\_WRPR）。这些内部寄存器配置系统，并可以被软件读。置位 OBL\_LAUNCH 位，产生了一个复位，这样选项字节的装载，才能在系统的复位下进行。

每个 Option 位在它相同的双字地址（下一个半字）有相应的补码。在选项字节装载期间，会对 Option bit 和其补码的验证，这能确保装载被正确地进行了。

如果正补码匹配，则选项字节被复制到选项寄存器中。

如果正补码不匹配，则 FLASH\_SR 寄存器的 OPTVERR 状态位被置位。不匹配的位被写入如下值：

- 对于 User option
- BOR\_LEV 写成 000 (最低阈值)
- BOR\_EN 位写成 0 (BOR 不使能)
- NRST\_MODE 位写成 0 (仅复位输入)
- RDP 位写成 0xff (即 level 1)
- 其余不匹配的值都写成 1
- 对于 WRP option, 不匹配的值是缺省值“无保护”

在系统复位后，选项字节的内容被复制到下面的选项寄存器（软件可读可写）：

- FLASH\_OPTTR
- FLASH\_BORCR
- FLASH\_WRPR

这些寄存器也被用来修改选项字节。如果这些寄存器不被用户修改，他们体现了系统 Option 的状态。

## 4.4. Flash 配置字节

芯片内的 Flash 的 Information 区域的部分区间（共 1 个 Page）作为 FT info0 使用。

Page 0 存放供软件读取信息（仅有正码，无反码存放）：

- HSI 频率选择控制值，及对应的 Trimming 值
- 对应 HSI 不同频率的擦写时间配置参数值

表 4-5 FT info0 配置

Page	Word	Address	Contents
0	0	0x1FFF 3200	存放 HSI 4 MHz 频率选择控制及对应的 Trimming 值
	1	0x1FFF 3208	存放 HSI 8 MHz 频率选择控制及对应的 Trimming 值
	2	0x1FFF 3210	存放 HSI 16 MHz 频率选择控制及对应的 Trimming 值
	3	0x1FFF 3218	存放 HSI 22.12 MHz 频率选择控制及对应的 Trimming 值
	4	0x1FFF 3220	存放 HSI 24 MHz 频率选择控制及对应的 Trimming 值
	5	0x1FFF 3228	TS_CAL1 , 30 °C温度传感器的校准值
	6	0x1FFF 3230	TS_CAL2 , 105 °C温度传感器的校准值
	7	0x1FFF 3238	存放 HSI 4 MHz 频率下对应的 FLASH_TS0、FLASH_TS1 寄存器的配置值
	8	0x1FFF 3240	存放 HSI 4 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	9	0x1FFF 3248	存放 HSI 4 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
	10	0x1FFF 3250	存放 HSI 4 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
	11	0x1FFF 3258	存放 HSI 4 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
	12	0x1FFF 3260	存放 HSI 8 MHz 频率下对应的 FLASH_TS0、FLASH_TS1 寄存器的配置值
	13	0x1FFF 3268	存放 HSI 8 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
	14	0x1FFF 3270	存放 HSI 8 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
15	0x1FFF 3278	存放 HSI 8 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值	



16	0x1FFF 3280	存放 HSI 8 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
17	0x1FFF 3288	存放 HSI 16 MHz 频率下对应的 FLASH_TS0、FLASH_TS1 寄存器的配置值
18	0x1FFF 3290	存放 HSI 16 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
19	0x1FFF 3298	存放 HSI 16 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
20	0x1FFF 32A0	存放 HSI 16 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
21	0x1FFF 32A8	存放 HSI 16 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
22	0x1FFF 32B0	存放 HSI 22.12 MHz 频率下对应的 FLASH_TS0、FLASH_TS1 寄存器的配置值
23	0x1FFF 32B8	存放 HSI 22.12 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
24	0x1FFF 32C0	存放 HSI 22.12 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
25	0x1FFF 32C8	存放 HSI 22.12 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
26	0x1FFF 32D0	存放 HSI 22.12 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值
27	0x1FFF 32D8	存放 HSI 24 MHz 频率下对应的 FLASH_TS0、FLASH_TS1 寄存器的配置值
28	0x1FFF 32E0	存放 HSI 24 MHz 频率下对应的 FLASH_TS2P、FLASH_TPS3 寄存器的配置值
29	0x1FFF 32E8	存放 HSI 24 MHz 频率下对应的 FLASH_PERTPE 寄存器的配置值
30	0x1FFF 32F0	存放 HSI 24 MHz 频率下对应的 FLASH_SMERTPE 寄存器的配置值
31	0x1FFF 32F8	存放 HSI 24 MHz 频率下对应的 FLASH_PRGTPE、FLASH_PRETPE 寄存器的配置值

#### 4.4.1. HSI\_TRIMMING\_FOR\_USER

**Address:** 0x1FFF 3200 ~ 0x1FFF 3220

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	HSI_FS[2: 0]		
-	-	-	-	-	-	-	-	-	-	-	-	-	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	HSI_TRIM[12: 0]												
-	-	-	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要从该地址读出数据，再写入 RCC\_ICSCR 寄存器对应的 HSI\_FS[2: 0]和 HSI\_TRIM[12: 0]，以实现 HSI 频率的更改。

#### 4.4.2. 温度传感器的校准值

**Address:** 0x1FFF 3228 (30 °C)、0x1FFF 3230 (105 °C)

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	TSCAL[11: 0]										
-	-	-	-	R										

软件需要从该地址读出数据。

#### 4.4.3. HSI\_4M/8M/16M/22.12M/24M\_EPPARA0

**Address:** 0x1FFF 3238 (4 MHz)、0x1FFF 3260 (8 MHz)、0x1FFF 3288 (16 MHz)、0x1FFF 32B0 (22.12 MHz)、0x1FFF 32D8 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res	Res	Res	Res	Res	Res	Res	TS1[8: 0]										
-	-	-	-	-	-	-	R	R	R	R	R	R	R	R	R		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
TS3[7: 0]							TS0[7: 0]										
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R		

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH\_TS0、FLASH\_TS1、FLASH\_TS3 寄存器，以实现对应 HSI 频率所需的擦写时间的配置。

#### 4.4.4. HSI\_4M/8M/16M/22.12M/24M\_EPPARA1

**Address:** 0x1FFF 3240 (4 MHz)、0x1FFF 3268 (8 MHz)、0x1FFF 3290 (16 MHz)、0x1FFF 32B8 (22.12 MHz)、0x1FFF 32E0 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res	Res	Res	Res	Res	TPS3[10: 0]												
-	-	-	-	-	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	Res	Res	Res	Res	TS2P[7: 0]									
-	-	-	-	-	-	-	-	R	R	R	R	R	R	R	R		

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH\_TS2P、FLASH\_TPS3 寄存器，以实现对应 HSI 频率所需的擦写时间的配置。

#### 4.4.5. HSI\_4M/8M/16M/22.12M/24M\_EPPARA2

**Address:** 0x1FFF 3248 (4 MHz)、0x1FFF 3270 (8 MHz)、0x1FFF 3298 (16 MHz)、0x1FFF 32C0 (22.12 MHz)、0x1FFF 32E8 (24 MHz)

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	PERTPE [16]
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERTPE[15: 0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH\_PERTPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

#### 4.4.6. HSI\_4M/8M/16M/22.12M/24M\_EPPARA3

**Address:** 0x1FFF 3250 (4 MHz) 、0x1FFF 3278 (8 MHz) 、0x1FFF 32A0 (16 MHz) 、0x1FFF 32C8 (22.12 MHz) 、0x1FFF 32F0 (24 MHz)

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	SMERTPE[16]
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMERTPE[15: 0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH\_SMERTPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

#### 4.4.7. HSI\_4M/8M/16M/22.12M/24M\_EPPARA4

**Address:** 0x1FFF 3258 (4 MHz) 、0x1FFF 3280 (8 MHz) 、0x1FFF 32A8 (16 MHz) 、0x1FFF 32D0 (22.12 MHz) 、0x1FFF 32F8 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	PRETPE[11: 0]										
-	-	-	-	-	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
PRGTPE[15: 0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

软件需要根据需要设定的 HSI 时钟频率，选择从相应地址读出数据，再写入 FLASH\_PRGTPE 和 FLASH\_PRETPE 寄存器中，以实现对应 HSI 频率所需的擦写时间的配置。

## 4.5. 闪存保护

对 Flash main memory 的保护包括以下几种机制：

- 读保护 (Read protection, RDP) ，防止来自外部的访问。
- 写保护 (Write protection, WRP) ，以防止不想要的写操作（由于程序存储器指针 PC 的混乱）。写保护的粒度设计为 8 KB。
- 选项字节写保护，专门的解锁设计。

### 4.5.1. 闪存读保护

通过设置选项字节读保护，并进行系统复位 (POR/BOR 或者 OPL 复位) 装载新的选项字节读保护，可以激活读保护功能。RDP 保护 Flash main memory、选项字节、SRAM。

如果通过 SWD 的 Debug 仍在连接时，读保护被设置，（此时访问出错）需要进行上电复位而不是系统复位。

当选项字节读保护和补码成对正确存在于选项字节时，Flash memory 会被保护。

表 4-6 闪存读保护状态

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
除 (0xAA 和 0x55) 组合的任何值		Level 1

无论任何保护级别，System memory 只能读访问，不能进行 Program 和擦除操作。

#### ■ Level 0: 无保护

对 Main flash 的读、Program 和擦操作是可能的，对选项字节也是可以进行任何操作。

#### ■ Level 1: 读保护

当选项字节里的 RDP 及其补码包含任何 (0xAA、0x55) 之外的组合，则 Level 1 读保护生效，Level 1 是缺省的保护级别。

- 用户模式：在用户模式下执行的程序（从 Main flash 启动），可以对 Main flash、选项字节进行所有操作。
- Debug 模式：从 SRAM 启动以及从 System memory mode 启动（Boot loader）：在 Debug 模式，或者当从 SRAM 或者 System memory（Boot loader）启动，Main flash 是不能被访问的。在这些模式下，对 Main flash 读或者写访问将产生一个总线错误，以及产生一个 Hard-Fault 中断。

当已处于 Level 1 (0xAA 之外任何数)，如果要修改为 Level 0 (写 0xAA)，硬件会对 Main flash 进行全片擦除操作。

表 4-7 访问状态与保护级别和执行模式的关系

区域	读保护等级	从 Main Flash (CPU) 启动			调试 / 从 SRAM 执行/ 从 System memory 执行			DMA		
		用户执行操作			读	写	擦除	读	写	擦除
		读	写	擦除						
System memory	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No
选项字节区域	0/1	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
		Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
FT info0	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No
UID	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No

注：

1. 任何对 Level 1 修改为 Level 0，都会触发硬件对 Main flash 的全擦。
2. 对于从 SRAM 或者 System memory 执行程序包括两种情况：一个是 Boot from SRAM 或者 System memory；另一个是从别的存储器 Boot，程序跳转到 SRAM 或者 System memory。

## 4.5.2. 闪存写保护

Flash 可以被设置成写保护，以应对不想要的写操作。定义 WRP 寄存器每 Bit 的控制粒度为 8 KB 的写保护 (WRP) 区域，即 1 个扇区大小。具体参见 WRP 寄存器的描述。

当被 WRP 的区域被激活，则不允许进行擦除或者 Program 操作。相应的，即使只有一个区域被设定为写保护，则 Mass 擦除功能不起作用。

此外，如果尝试对设为写保护的区域进行擦除或者 Program 操作，则 FLASH\_SR 寄存器的写保护错误标识 (WRPERR) 会被置位。

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

## 4.5.3. 选项字节写保护

默认情况下，选项字节是可读，并进行写保护的。为获得对选项字节的擦除或者 Program 访问，需要向 OPTKEYR 寄存器写入正确的序列。

## 4.6. 闪存中断

表 4-8 闪存中断请求

中断事件	事件标志	时间标志/中断清除方法	控制位使能
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE

注：以下事件没有单独的中断标识，但会产生 HardFault：

1. 解锁 Flash memory 的 FLASH\_CR 寄存器的序列错误
2. 解锁 Flash 选项字节的写操作序列错误
3. Flash program 操作未进行 32 位数据的对齐
4. Flash 擦除 (含 page 擦除、sector 擦除和 mass 擦除) 操作未进行 32 位数据对齐
5. 对选项字节寄存器的写操作未进行 32 位数据的对齐

## 4.7. 闪存寄存器描述

### 4.7.1. Flash 访问控制寄存器 (FLASH\_ACR)

Address offset: 0x00

Reset value: 0x0000 0700

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

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1: 0	LATENCY[1: 0]	RW	0	Flash 读操作对应的等待状态： 00: Flash 读操作没有等待状态 (SYSCLK<=24 MHz) 01: Flash 读操作有 1 个等待状态，即每次读 Flash 需要两个系统时钟周期 (24 MHz <SYSCLK<=48 MHz) 10: Flash 读操作有 3 个等待状态，即每次读 Flash 需要四个系统时钟周期 (48 MHz <SYSCLK<=72 MHz) 11: 保留 注：操作 LATENCY 寄存器时，FLASH_ACR 其余高位不允许写 1

#### 4.7.2. Flash 密钥寄存器 (FLASH\_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

所有寄存器位是 Write-Only，读出返回 0。

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

Bit	Name	R/W	Reset Value	Function
31: 0	KEY[31: 0]	W	0	下面的值必须被连续的写入，才能 unlock FLASH_CR 寄存器，并使能了 Flash 的 program/擦除操作 KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

#### 4.7.3. Flash 选项密钥寄存器 (FLASH\_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

所有寄存器位是 Write-Only，读出返回 0。

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

W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Bit	Name	R/W	Reset Value	Function
31: 0	OPTKEY[31: 0]	W	0	下面的值必须被连续的写入，才能 unlock flash 的选项寄存器，并使能选项字节的 program/擦除操作 KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

#### 4.7.4. Flash 状态寄存器 (FLASH\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 17	保留	-	-	保留
16	BSY	R	0	Busy 位 该位表示 Flash 的操作正在进行。该位在 Flash 操作的开始被硬件置位，当操作完成或者错误产生时由硬件清零。
15	OPTVERR	RC_W1	0	Option 和 trimming 位装载错误 当 option 和 trimming bit 及其反码不匹配时，硬件置位该位。装载不匹配的选项字节，被强制成安全值。 软件写 1，清零。
14: 5	保留	-	-	保留
4	WRPERR	RC_W1	0	写保护错误 当要被 program/擦除的地址处于被写保护的 Flash 区域时 (WRP)，硬件置位该位。 软件写 1，清零该位。
3: 1	保留	-	-	保留
0	EOP	RC_W1	0	当 Flash 的 program/擦除操作成功完成，硬件置位。该位仅当如果 FLASH_CR 寄存器的 EOPIE 位使能才会被置位。 软件写 1，清零该位。

#### 4.7.5. Flash 控制寄存器 (FLASH\_CR)

Address offset: 0x14

Reset value: 0xC000 0000

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

Bit	Name	R/W	Reset Value	Function
31	Lock	RS	1	FLASH_CR Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器被 Lock 住。当成功给出 unlock 时序后，该位被硬件清零，unlock 了 FLASH_CR 寄存器。 【软件要在 program/擦除操作完成后，置位该位】 当不成功的 unlock 时序给出，该位仍然保持置位状态，直到下一次系统复位。
30	OPTLOCK	RS	1	选项字节 Lock 位。 软件对该位只能置位。当置位后，FLASH_CR 寄存器中与选项字节有关的位被 Lock 住。当成功给出 unlock 时序后，该位被硬件清零，unlock 了 FLASH_CR 寄存器。 【软件要在 program/擦除操作完成后，置位该位】 当不成功的 unlock 时序给出，该位仍然保持置位状态，直到下一次系统复位。
29: 28	保留	-	-	保留
27	OBL_LAUNCH	RC_W1	0	强制选项字节重装载。 当置位时，该位强制系统进行选项字节的重装载。该位仅当 option byte 装载被完成后被硬件清零。如果 OPT-LOCK 位被置位，该位不能被写。 0: 选项字节重载完成 1: 产生选项字节重载请求，系统产生复位，进行选项字节的重装载。
26	保留	-	-	保留
25	ERRIE	RW	0	错误中断使能位，当 FLASH_SR 寄存器的 WRPERR 位被置位，如果该位使能，则产生中断请求。 0: 无中断产生



Bit	Name	R/W	Reset Value	Function
				1: 有中断产生
24	EOPIE	RW	0	操作结束中断使能 当 FLASH_SR 寄存器的 EOP 位被置位, 该位使能中断的产生。 0: EOP 中断关闭 1: EOP 中断使能
23: 20	保留	-	-	保留
19	PGSTRT	RW	0	Flash main memory 的 program 操作的启动位。 该位启动了 Flash main memory 的 program 操作, 软件置位, 在 FLASH_SR 寄存器的 BSY 位被清零后, 硬件清零该位。
18	保留	-	-	保留
17	OPTSTRT	RW	0	Flash 选项字节修改的启动位 该位启动了对选项字节的修改。软件置位, 在 FLASH_SR 寄存器的 BSY 位被清零后, 硬件清零该位。 注意: 当对 flash 选项字节进行修改时, 硬件自动把整个 256 Bytes 的 page 进行擦除操作, 再进行 program 操作, 其中也包括自动进行补码的写入。
16: 12	保留	-	-	保留
11	SER	RW	0	8 KB 的扇区擦除操作 0: 未选择 Flash 的扇区擦除操作 1: 选择 Flash 的扇区擦除操作 注: Sector 擦除不会对 Flash information memory 起作用。 Sector 擦除对设定为 WRP 的区域不起作用。
10: 3	保留	-	-	保留
2	MER	RW	0	Mass 擦除操作 0: 未选择 Flash 的片擦除操作 1: 选择 Flash 的片擦除操作 注: Mass 擦除不会对 Flash information memory 起作用。当有 WRP 设定时, 片擦除不起作用
1	PER	RW	0	Page 擦除操作 0: 未选择 Flash 的页擦除操作 1: 选择 Flash 的页擦除操作
0	PG	RW	0	Program 操作 0: 未选择 Flash 的 program 操作 1: 选择 Flash 的 program 操作

#### 4.7.6. Flash 选项寄存器 (FLASH\_OPTR)

Address offset: 0x20

Reset value: 0x0000 D8AA。在上电复位 (POR/BOR/OBL\_LAUNCH) 释放后, 从 flash information memory 的选项字节区域读出相应的值, 写入到该寄存器相应的 option bit。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_STOP	nBOOT1	NRST_MODE	WWDG_SW	IWDG_SW	Res			RDP[7: 0]							
RW	RW	RW	RW	RW	-			RW							

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	IWDG_STOP	RW	1	设置 IWDG 在 stop 模式下定时器运行状态 0: 冻结定时器 1: 正常运行
14	nBOOT1	RW	1	与 BOOT PIN 一起, 选择芯片启动模式
13	NRST_MODE	RW	0	0: 仅复位输入 1: GPIO: GPIO 功能
12	WWDG_SW	RW	1	0: 硬件看门狗 1: 软件看门狗
11	IWDG_SW	RW	1	0: 硬件看门狗 1: 软件看门狗
10: 8	保留	-	-	保留
7: 0	RDP	RW	0xAA	0xAA: level 0, 读保护无效 非 0xAA: level 1, 读保护有效

#### 4.7.7. Flash BORCR 地址寄存器 (FLASH\_BORCR)

Address offset: 0x24

Reset value: 0x0000 0000。在上电复位 (POR/BOR/OBL\_LAUNCH) 释放后, 从 flash information memory 的选项字节区域读出相应的值, 写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 13	BOR_LEV[2: 0]	RW	0	000: BOR 上升阈值为 1.8 V, 下降阈值位 1.7 V 001: BOR 上升阈值为 2.0 V, 下降阈值位 1.9 V 010: BOR 上升阈值为 2.2 V, 下降阈值位 2.1 V 011: BOR 上升阈值为 2.4 V, 下降阈值位 2.3 V 100: BOR 上升阈值为 2.6 V, 下降阈值位 2.5 V 101: BOR 上升阈值为 2.8 V, 下降阈值位 2.7 V 110: BOR 上升阈值为 3.0 V, 下降阈值位 2.9 V 111: BOR 上升阈值为 3.2 V, 下降阈值位 3.1 V
12: 6	保留	-	-	保留
5	BOR_EN	RW	0	BOR 使能 0: BOR 不使能 1: BOR 使能, BOR_LEV 起作用
4: 0	保留	-	-	保留

#### 4.7.8. Flash WRP 地址寄存器 (FLASH\_WRP)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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
WRP[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	WRP	RW	1	0: sector 15, 有写保护, 不允许进行 program 和擦除 1: sector 15, 无写保护
14	WRP	RW	1	0: sector 14, 有写保护, 不允许进行 program 和擦除 1: sector 14, 无写保护
13	WRP	RW	1	0: sector 13, 有写保护, 不允许进行 program 和擦除 1: sector 13, 无写保护
12	WRP	RW	1	0: sector 12, 有写保护, 不允许进行 program 和擦除 1: sector 12, 无写保护

Bit	Name	R/W	Reset Value	Function
11	WRP	RW	1	0: sector 11, 有写保护, 不允许进行 program 和擦除 1: sector 11, 无写保护
10	WRP	RW	1	0: sector 10, 有写保护, 不允许进行 program 和擦除 1: sector 10, 无写保护
9	WRP	RW	1	0: sector 9, 有写保护, 不允许进行 program 和擦除 1: sector 9, 无写保护
8	WRP	RW	1	0: sector 8, 有写保护, 不允许进行 program 和擦除 1: sector 8, 有写保护
7	WRP	RW	1	0: sector 7, 有写保护, 不允许进行 program 和擦除 1: sector 7, 无写保护
6	WRP	RW	1	0: sector 6, 有写保护, 不允许进行 program 和擦除 1: sector 6, 无写保护
5	WRP	RW	1	0: sector 5, 有写保护, 不允许进行 program 和擦除 1: sector 5, 无写保护
4	WRP	RW	1	0: sector 4, 有写保护, 不允许进行 program 和擦除 1: sector 4, 无写保护
3	WRP	RW	1	0: sector 3, 有写保护, 不允许进行 program 和擦除 1: sector 3, 无写保护
2	WRP	RW	1	0: sector 2, 有写保护, 不允许进行 program 和擦除 1: sector 2, 无写保护
1	WRP	RW	1	0: sector 1, 有写保护, 不允许进行 program 和擦除 1: sector 1, 无写保护
0	WRP	RW	1	0: sector 0, 有写保护, 不允许进行 program 和擦除 1: sector 0, 无写保护

#### 4.7.9. Flash 睡眠时间配置寄存器 (FLASH\_STCR)

Address offset: 0x90

Reset value: 0x0000 6400

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
SLEEP_TIME[7: 0]								Res	Res	Res	Res	Res	Res	Res	SLEEP_EN
RW	RW	RW	RW	RW	RW	RW	RW	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
15: 8	SLEEP_TIME	RW	0x64	Flash 睡眠时间计数（基于 HSI_10M 时钟的计数器） 当系统时钟选择 LSI 或者 LSE 时，为获得更优化的 Run 模式功耗，可选择使用该寄存器的功能（仅推荐在 LSI 或者 LSE 为系统时钟时，使用该功能）。 当使能该功能时，每半个系统时钟低电平周期内 Flash 处于 Sleep 状态的时间宽度为： $t_{HSI\_10M} * SLEEP\_TIME$ 注： $t_{HSI\_10M}$ 为 HSI_10M 的周期。
7: 1	保留	-	-	保留
0	SLEEP_EN	RW	0	Flash Sleep 使能 1: 使能 Flash sleep 0: 失能 Flash sleep

#### 4.7.10. Flash TS0 寄存器 (FLASH\_TS0)

Address offset: 0x100

Reset value: 0x0000 00B4

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	TS0							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	TS0	RW	0xB4	HSI 输出频率不同，则需要设定如下相应的值 HSI 为 4MHz: 0x1E HSI 为 8MHz: 0x3C HSI 为 16MHz: 0x78 HSI 为 22.12MHz: 0xA6 HSI 为 24MHz: 0xB4

#### 4.7.11. Flash TS1 寄存器 (FLASH\_TS1)

Address offset: 0x104

Reset value: 0x0000 01B0

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	TS1								
-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 9	保留	-	-	保留
8: 0	TS1	RW	0x1B0	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x48 HSI 为 8 MHz: 0x90 HSI 为 16 MHz: 0x120 HSI 为 22.12 MHz: 0x18F HSI 为 24 MHz: 0x1B0

#### 4.7.12. Flash TS2P 寄存器 (FLASH\_TS2P)

Address offset: 0x108

Reset value: 0x0000 00B4

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	TS2P							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	TS2P	RW	0xB4	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x1E HSI 为 8 MHz: 0x3C HSI 为 16 MHz: 0x78 HSI 为 22.12 MHz: 0xA6 HSI 为 24 MHz: 0xB4

#### 4.7.13. Flash TPS3 寄存器 (FLASH\_TPS3)

Address offset: 0x10C

Reset value: 0x0000 06C0

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	TPS3										
-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 11	保留	-	-	保留
10: 0	TPS3	RW	0x6C0	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x120 HSI 为 8 MHz: 0x240 HSI 为 16 MHz: 0x480 HSI 为 22.12 MHz: 0x639 HSI 为 24 MHz: 0x6C0

#### 4.7.14. Flash TS3 寄存器 (FLASH\_TS3)

Address offset: 0x110

Reset value: 0x0000 00B4

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	TS3							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	TS3	RW	0xB4	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x1E HSI 为 8 MHz: 0x3C HSI 为 16 MHz: 0x78 HSI 为 22.12 MHz: 0xA6 HSI 为 24 MHz: 0xB4

#### 4.7.15. Flash 页擦写 (PAGE ERASE) TPE 寄存器 (FLASH\_PERTPE)

Address offset: 0x114

Reset value: 0x0001 1940

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	PERTPE[16]
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERTPE[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 17	保留	-	-	保留
16: 0	PERTPE	RW	0x11940	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x36B0 HSI 为 8 MHz: 0x6D60 HSI 为 16 MHz: 0XDAC0 HSI 为 22.12 MHz: 0x12E6C HSI 为 24 MHz: 0x14820

#### 4.7.16. Flash SECTOR/MASS ERASE TPE 寄存器 (FLASH\_SMERTPE)

Address offset: 0x118

Reset value: 0x0001 1940

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	SMERTPE[16]
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMERTPE[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 17	保留	-	-	保留
16: 0	SMERTPE	RW	0x11940	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x36B0 HSI 为 8 MHz: 0x6D60 HSI 为 16 MHz: 0XDAC0 HSI 为 22.12 MHz: 0x12E6C HSI 为 24 MHz: 0x14820

#### 4.7.17. Flash PROGRAM TPE 寄存器 (FLASH\_PRGTPE)

Address offset: 0x11C

Reset value: 0x0000 A8C0



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
PRGTPE															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PRGTPE	RW	0xA8C0	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0xFA0 HSI 为 8 MHz: 0x1F40 HSI 为 16 MHz: 0x3E80 HSI 为 22.12 MHz: 0x5668 HSI 为 24 MHz: 0x5DC0

#### 4.7.18. Flash PRE-PROGRAM TPE 寄存器 (FLASH\_PRETPE)

Address offset: 0x120

Reset value: 0x0000 12C0

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	PRETPE[13: 0]													
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 14	保留	-	-	保留
13: 0	PRETPE	RW	0x12C0	HSI 输出频率不同, 则需要设定如下相应的值 HSI 为 4 MHz: 0x320 HSI 为 8 MHz: 0x640 HSI 为 16 MHz: 0xC80 HSI 为 22.12 MHz: 0x1148 HSI 为 24 MHz: 0x12C0

## 5. 电源控制

### 5.1. 电源

#### 5.1.1. 电源框图

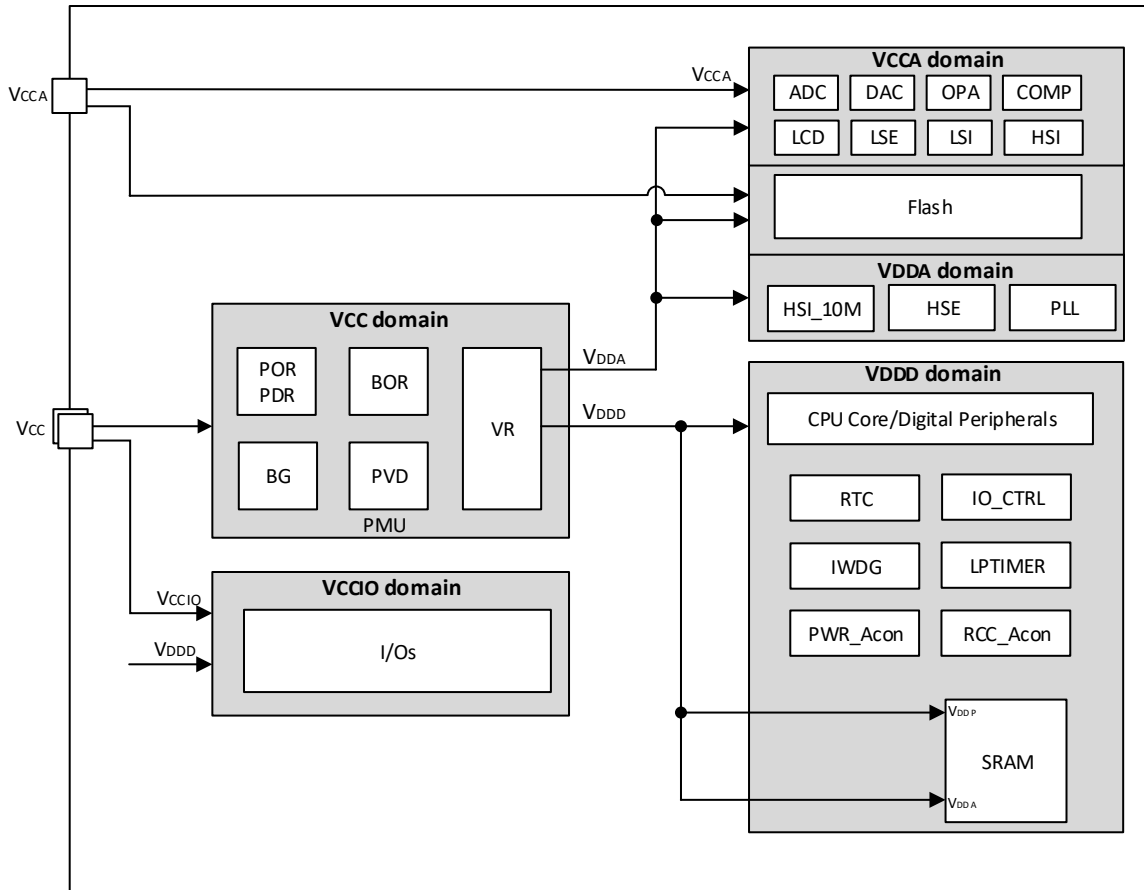


图 5-1 电源框图

表 5-1 电源框图

编号	电源	电源值	描述
1	VCC	1.7 ~ 5.5 V	通过电源管脚为芯片提供电源。
2	VCCA	1.7 ~ 5.5 V	通过电源管脚为芯片模拟电路提供电源。
3	VDDx (VDDD/VDDA)	1.2 V/1.0 V/0.9 V/0.8 V	来自于 VR 的输出，为芯片内部主要逻辑电路、SRAM 供电。当 MR 供电时，输出 1.2 V。当进入 Stop 模式时，根据软件配置，可以由 MR 或者 LPR 供电，并根据软件配置决定 LPR 输出是 1.2 V/1.0 V/0.9 V/0.8 V。

## 5.2. 电压调节器

芯片设计两个电压调节器：

- MR (Main regulator) 在芯片正常运行状态时保持工作。
- LPR (Low power regulator) 在 Stop 模式下，提供更低功耗的选择。

$V_{DDX}$  的电源根据芯片的工作模式，来自于 MR 或 LPR。

在芯片 Run 模式，MR 保持工作，输出 1.2 V 电压，LPR 关闭。

在 Stop 模式，可由软件决定从 MR 或 LPR 供电。同样，由软件决定进入 Stop 后，LPR 供电情况下的  $V_{DDX}$  是 1.2 V/1.0 V/0.9 V/0.8 V。

## 5.3. 电源监控

### 5.3.1. 上电复位 (POR) / 下电复位 (PDR) / 欠压复位 (BOR)

芯片内设计 POR/PDR 模块，放在  $V_{DDX}$  电源域下，为芯片提供上电和下电复位。该模块在各种模式之下都保持工作。

除了 POR/PDR 外，还实现了 BOR (Brown-out reset)。BOR 仅可以通过选项字节进行使能和关闭操作。

当 BOR 被打开时，BOR 的阈值可以通过选项字节进行选择，且上升和下降检测点都可以被单独配置。

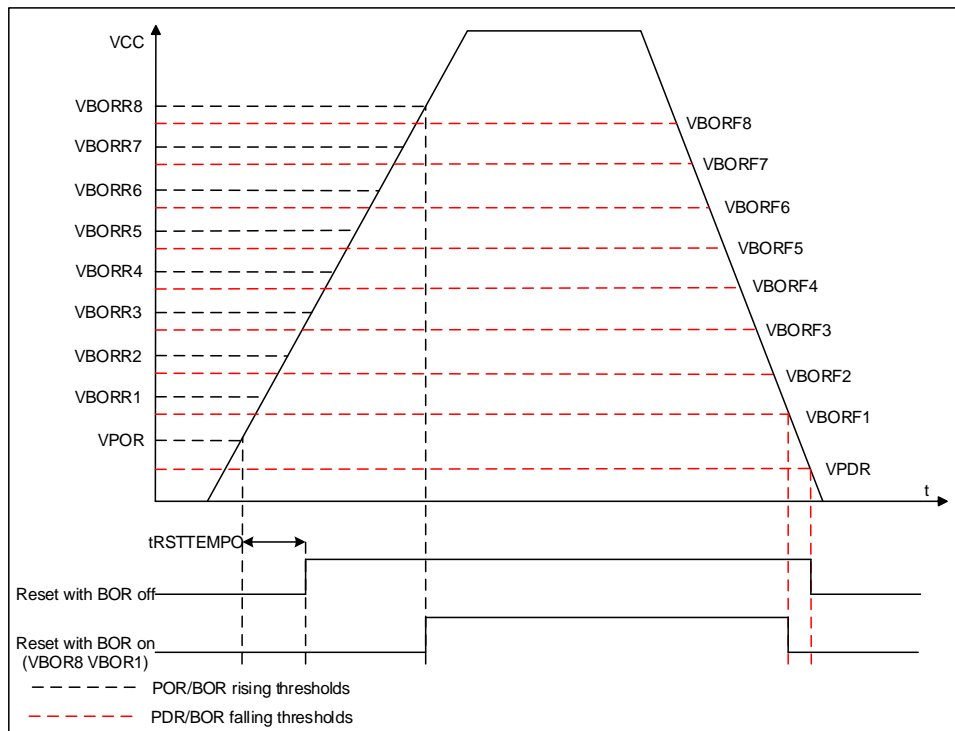


图 5-2 POR/PDR/BOR 阈值

### 5.3.2. 可编程电压检测器 (PVD)

该模块可以用来检测  $V_{CC}$  电源（也可以检测 PB7 引脚的电压），检测点可通过寄存器进行配置。当  $V_{CC}$  高于或者低于 PVD 的检测点时，产生相应的标识。

该事件内部连接到 EXTI 的 line 16，取决于 EXTI line 16 上升/下降沿配置，当  $V_{CC}$  上升超过 PVD 的检测点，或者  $V_{CC}$  降低到 PVD 的检测点以下，产生中断，在中断服务程序中用户可以进行紧急的关闭 (Shutdown) 任务。

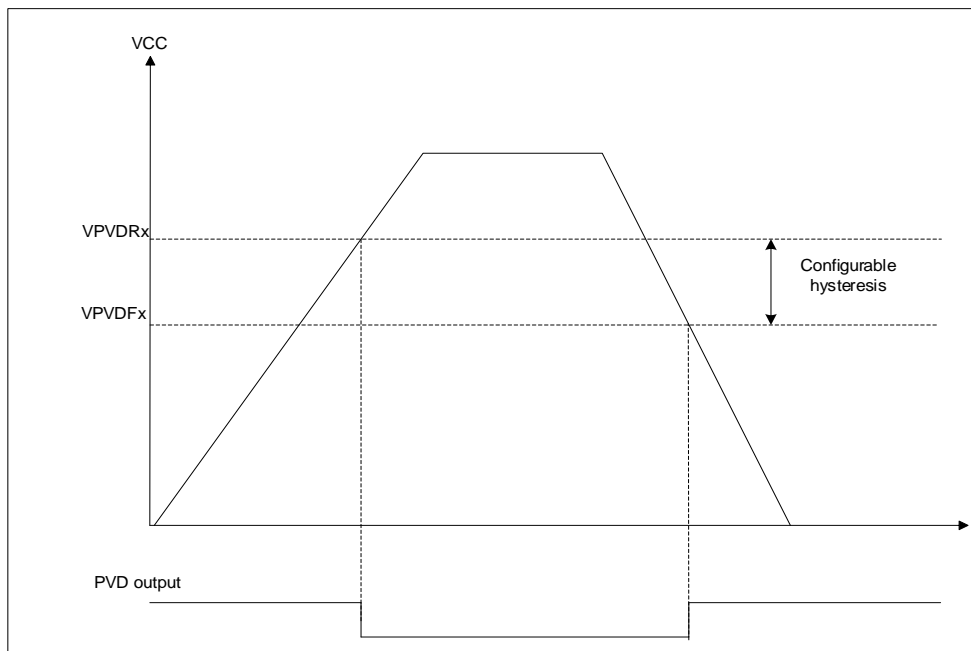


图 5-3 PVD 阈值

## 6. 低功耗控制

默认状态下，芯片在系统或者电源复位之后，进入正常运行 Run 模式。当 CPU 不需要持续工作时，芯片可进入低功耗模式。例如，当等待外部事件时，软件可以在功耗、唤醒时间、唤醒源之间折中选择。

### 6.1. 低功耗模式

#### 6.1.1. 低功耗模式介绍

芯片在正常的 Run 模式之外，有 2 个低功耗模式：

- Sleep mode: CPU 时钟关闭 (NVIC, SysTick 可工作)，外设可以配置为保持工作。（建议只使能必须工作的模块，在模块工作结束后关闭该模块）。
- Stop mode: 该模式下 SRAM 和寄存器的内容保持，HSI、HSE 和 PLL 关闭，VDD 域下大部分模块的时钟都被停掉。

在 Stop 模式，LSI 和 LSE 可以保持工作，RTC、LPTIMER、IWDG 等可以保持工作。具体该模式下各模块的工作情况，参照表 6-2 各工作模式下的功能 (1)。

在 Stop 模式下，对应的 VR 状态可由软件控制，设成 MR 或者 LPR 供电。当 LPR 供电时，芯片功耗大大降低，但唤醒时间较长；当保持 MR 供电的情况，芯片功耗较大，但具备几个周期的快速唤醒能力。

此外，正常 Run 模式下可以通过下述方法降低功耗：

1. 降低系统时钟频率
2. 对于不使用的外设，关掉外设时钟（系统时钟和模块时钟）

综上所述，本项目的低功耗模式转换图如下所述。

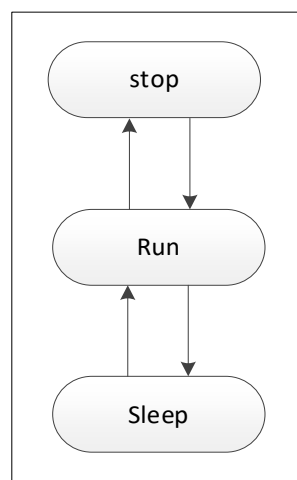


图 6-1 电源模式

### 6.1.2. 低功耗模式开关

表 6-1 低功耗模式开关

模式	进入	唤醒源	唤醒时钟	对时钟的影响	Voltage regulator	
					MR	LPR
Sleep (sleep-now or sleep-on-exit)	WFI or Return from ISR	任何中断	与进入 sleep 之前	CPU 时钟停止, 对其他时钟和时 钟源没影响。	开  ( <sup>1</sup> )	关
	WFE	唤醒事件	一样			
Stop	SLEEPDEEP bit+ WFI or Return from ISR or WFE  注: 系统时钟不能 选择 LSI	任何配置为唤 醒的 EXTI Line (EXTI 寄存器配 置)、 IWDG、 NRST	HSISYS HSI 保持进 入 stop 前 的频率配 置, 不分频	HSI 关闭; HSE 关闭; PLL 关闭; LSI 可选择开或者关; LPTIMER、RTC、IWDG: 由软 件配置是否工作; 低功耗唤醒和部分 RCC 等模块保 持工作; 其余外设模块的时钟关闭。	软件 配置 开关	软件配置开关, 如果开, 输出电 压 1.2 V/1.0 V/0.9 V/0.8 V 可配置

注 1: 软件要配置 VR 的状态为 MR 模式, 才能进入 Sleep 模式。

### 6.1.3. 各工作模式下的功能

表 6-2 各工作模式下的功能<sup>(1)</sup>

外设	Run	Sleep	Stop	
			VR@LPR or VR@MR	唤醒能力
CPU	Y	-	-	-
Flash memory	Y	Y	- <sup>(2)</sup>	-
SRAM	Y	O <sup>(3)</sup>	- <sup>(4)</sup>	-
Brown-out reset (BOR)	Y	Y	O	O
PVD	O	O	O	O
DMA	O	O	-	-
HSI	O	O	-	-
HSE	O	O	-	-
LSI	O	O	O	-
PLL	O	O	-	-
HSE Clock Security System (CSS)	O	O	-	-
RTC	O	O	O	O
USART1	O	O	-	-

USART2	O	O	-	-
I <sup>2</sup> C	O	O	-	-
SPI1	O	O	-	-
ADC	O	O	-	-
COMP1/COMP2/3	O	O	O	O
Temperature sensor	O	O	-	-
Timers (TIM1/TIM3 /TIM14/TIM16/TIM17)	O	O	-	-
LPTIM	O	O	O	O
IWDG	O	O	O	O
WWDG	O	O	-	-
SysTick timer	O	O	-	-
CRC	O	O	-	-
GPIOs	O	O	O	O

1. Y = Yes (使能) ; O = Optional (默认关闭, 可以软件使能) ; - = Not available
2. Flash 不下电, 但无时钟提供, 进入最低功耗状态。
3. SRAM 的时钟可以被开或者关。
4. SRAM 不下电, 但无时钟提供, 进入最低功耗状态。

## 6.2. Sleep 模式

### 6.2.1. 进入 sleep 模式

通过执行 WFI (Wait for interrupt) 或者 WFE (Wait for event) 指令, 进入 Sleep 模式。取决于 Cortex M0+ 的系统控制寄存器的 SLEEPONEXIT 位, 有两种可选的进入 Sleep 模式的机制。

- Sleep-now: 如果 SLEEPONEXIT 位是 0, 则执行 WFI 或者 WFE 后, 立即进入 Sleep 模式。
  - Sleep-on-exit: 如果 SLEEPONEXIT 位是 1, 则当退出低优先级中断 ISR 时, 进入 Sleep 模式。
- 在 Sleep 模式, 所有的 IO 引脚与 Run 模式保持相同的状态。

### 6.2.2. 退出 sleep mode

如果用 WFI 进入 Sleep 模式, 被 NVIC 获得的任何外设中断可以把芯片从 sleep 模式唤醒。

如果用 WFE 进入 Sleep 模式, 当一个事件发生时, 芯片退出 Sleep 模式。Wakeup 事件可以通过以下方式产生:

- 在外设控制寄存器使能中断, 而不是在 NVIC, 并使能 Cortex M0+ 的 SEVONPEND 位。当芯片从 WFE 唤醒后继续执行时, 外设中断 Pending 位和外设 NVIC IRQ 通道 Pending 位 (在 NVIC 的中断清除 Pending 寄存器) 必须被清零。
- 或者, 配置外部或者内部 EXTI line 为事件模式。当 CPU 从 WFE 唤醒后继续执行时, 不必清除外设中断 Pending 位, 或者对应到事件 Line 的 NVIC IRQ 通道 Pending 位没有被置位。

该模式具有最短的 Wakeup 时间, 并且没有在中断进入和退出浪费时间。

表 6-3 Sleep-now

Sleep-now mode	Description
Mode entry	WFI 或者 WFE, 并且: - SLEEPDEEP = 0 且 - SLEEPONEXIT = 0
Mode exit	如果通过 WFI 进入的 sleep 模式, 则退出方式是: 中断。 如果通过 WFE 进入的 sleep 模式, 则退出方式是: 唤醒事件。
Wakeup latency	无

表 6-4 Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI, 并且: - SLEEPDEEP = 0 且 - SLEEPONEXIT = 1
Mode exit	中断
Wakeup latency	无

### 6.3. Stop 模式

Stop 模式是基于 Cortex-M0+ 的 DEEPSLEEP 以及对外设时钟的 Gating, VR 可以被配置成 MR 或者 LPR 供电。在该模式下, HSI、PLL 和 HSE 被关闭, SRAM 和寄存器内容处于保持状态, LSI、LPTIMER、RTC、IWDG 可由软件配置是否工作, 低功耗唤醒和部分 RCC 逻辑等保持工作, 其余  $V_{DDx}$  域的数字模块的时钟输入被关闭。

在 Stop 模式下, 所有的 IO 引脚保持跟 Run 模式相同的状态。

#### 6.3.1. 进入 Stop 模式

为了进一步降低 Stop 模式的功耗, 配置 PWR\_CR1.LPR=1 时, VR 可以进入 LPR 供电。

如果正在进行 Flash 的擦写操作, 则 Stop 模式的进入会被延迟, 直到存储器访问结束 (由软件读 FLASH\_SR 寄存器的 BSY 位判断当前是否已完成擦、写操作)。

如果 APB 总线上的操作正在进行, 则 Stop 模式的进入也会被延迟, 直到 APB 访问结束 (由软件控制)。

如果系统时钟源再进入低功耗模式之前是高速时钟源 (PLL/HSE), 为了保证系统时钟切换成功, 需要软件切换系统时钟源为 HSI。

#### 6.3.2. 退出 Stop 模式

当通过中断或者 Wakeup 事件退出 Stop 模式时, HSI 被选择作为系统时钟。

在 Stop 模式, 如果 VR 处于 LPR 状态, 则从 Stop 模式唤醒有额外的延迟。

在 Stop 模式, 如果 VR 处于 MR 状态, 电流消耗会大, 但唤醒时间会被减少。



表 6-5 Stop 模式

Stop 模式	描述
进入模式	<p>WFI (Wait for interrupt) 或者 WFE (Wait for event) , 并且:</p> <ol style="list-style-type: none"> <li>配置设定: <ul style="list-style-type: none"> <li>通过 PWR_CR1 的 LPR 位, 选择 VR 工作在 MR 或者 LPR 下</li> <li>通过 PWR_CR1 的 VOS 位, 选择 LPR 模式提供 1.2 V, 1.0 V, 0.9 V, 0.8 V</li> <li>通过 PWR_CR1 的 FLS_SLPTIME 配置 FLASH 的唤醒时间</li> </ul> </li> <li>置位 Cortex M0+ 的 SLEEPDEEP 位</li> </ol> <p>注:</p> <ol style="list-style-type: none"> <li>为了进入 Stop 模式, 所有 EXTI line 的 Pending 位 (EXTI_PR 寄存器)、所有外设的中断 Pending 位、RTC alarm 标志位, 必须被复位。否则, 进入 stop 模式的流程将被忽略掉, 程序继续执行。</li> <li>如果应用需要在进入 Stop 模式前关闭 HSE、PLL, 系统时钟源必须先切换到 HSI, 然后清除 HSEON 位。</li> <li>为使芯片功耗变化尽可能均衡, 软件需要遵循逐步关闭的原则: 逐步关闭各个模块的时钟, 选择 HSI 作为系统时钟, 关闭 HSE、PLL。</li> <li>为缩短唤醒时间, 在进入 Stop 模式前, 系统时钟应该配置选择 HSI 高频时钟, RCC_CFGR 寄存器的 HPRE 设为 0, 否则在唤醒后硬件切换时钟会消耗额外的时钟。</li> </ol>
退出模式	<p>如果使用 WFI 进入 Stop 模式:</p> <ul style="list-style-type: none"> <li>任何被配置成中断模式的 EXTI line (相应的 EXTI 中断向量必须被在 NVIC 中使能)</li> </ul> <p>如果使用 WFE 进入 Stop 模式:</p> <ul style="list-style-type: none"> <li>任何被配置成事件模式的 EXTI line</li> <li>CPU SEVONPEND 位置位情况下的中断 pending 位</li> </ul>
唤醒延迟	LPR 到 MR 唤醒时间 + HSI 唤醒时间 + Flash 唤醒时间

## 6.4. 降低系统时钟频率

在 Run 模式下, 系统时钟的频率 (SYSCLK, HCLK, PCLK) 可以通过预分频寄存器配置分频去降低。这些预分频器也可以被用来在进入 Sleep 模式前, 降低外设的频率。

## 6.5. 外设时钟门控

在 Run 模式, 可以在任何时间停止单个外设和存储器的 AHB 时钟 (HCLK) 和 APB 时钟 (PCLK), 以降低功耗。

为了进一步降低在 Sleep 模式的功耗, 外设的时钟可以在执行 WFI 或者 WFE 指令之前被停掉。

## 6.6. 电源管理寄存器

该外设的寄存器可以通过半字或者全字访问。

### 6.6.1. 电源控制寄存器 1 (PWR\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by POR)

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

Bit	Name	R/W	Reset Value	Function
31: 20	保留	-	-	保留
19	HSION_CTRL	RW	0	从 Stop 模式唤醒时, HSI 打开时间控制。 0: 等待 MR 稳定后, 使能 HSI; 1: 唤醒时立刻使能 HSI。
18: 15	保留	-	-	保留
14	LPR	RW	0	Low power regulator 0: Main regulator 工作在 stop 模式 1: Low power regulator 工作在 stop 模式
13: 12	FLS_SLPTIME[1: 0]	RW	2'b00	Stop 模式唤醒时序中, 在 HSI 稳定后, 在 FLASH 操作前需要等待时间。 2'b00: 1 μs 2'b01: 2 μs 2'b10: 3 μs 2'b11: 0 μs 注: 当该寄存器设置为 2'b11 时, 表明唤醒后是从 SRAM 执行程序, 而非 Flash。并且程序保证在唤醒执行程序后不会在 3us 内访问 Flash。
11	保留	-	-	保留
10: 9	VOS[1: 0]	RW	0	电压调节范围选择 00: 进入 Stop 模式后, V <sub>DD</sub> =1.2 V 01: 进入 Stop 模式后, V <sub>DD</sub> =1.0 V 10: 进入 Stop 模式后, V <sub>DD</sub> =0.9 V 11: 进入 Stop 模式后, V <sub>DD</sub> =0.8 V

8	DBP	RW	0	RTC 写保护禁止 在复位后, RTC 处于写保护状态以防意外写入。要访问 RTC 该位必须设置为 1。 0: 禁止访问 RTC 1: 可以访问 RTC
7: 0	保留	-	-	保留

### 6.6.2. 电源控制寄存器 2 (PWR\_CR2)

Address offset: 0x04

Reset value: 0x0000 0500 (reset by POR)

注: 该寄存器是与 PVD 功能相关寄存器。

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

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 9	FLT_TIME[2: 0]	RW	3'b010	数字滤波时间配置 110: 滤波时间大约为 30.7 ms (1024 个 LSI/LSE 时钟) 101: 滤波时间大约为 3.8 ms (128 个 LSI/LSE 时钟) 100: 滤波时间大约为 1.92 ms (64 个 LSI/LSE 时钟) 011: 滤波时间大约为 480 μs (16 个 LSI/LSE 时钟) 010: 滤波时间大约为 120 μs (4 个 LSI/LSE 时钟) 001: 滤波时间大约为 60 μs (2 个 LSI/LSE 时钟) 000: 滤波时间大约为 30 μs (1 个 LSI/LSE 时钟)
8	FLTEN	RW	1	数字滤波功能使能控制 0: 禁止 1: 使能
7	保留	-	-	保留
6: 4	PVDT[2: 0]	RW	000	电压上升沿检测阈值 (下降沿检测阈值相应减小 0.1 V) 及 PVDIN 检测控制。 000: VPVD0 (around 1.8 V) 001: VPVD1 (around 2.0 V) 010: VPVD2 (around 2.2 V) 011: VPVD3 (around 2.4 V) 100: VPVD4 (around 2.6 V)

Bit	Name	R/W	Reset Value	Function
				101: VPVD5 (around 2.8 V) 110: VPVD6 (around 3.0 V) 111: VPVD7 (around 3.2 V)
3	保留	-	-	保留
2	SRCSEL	RW	0	PVD 检测电源选择。 0: V <sub>CC</sub> 1: 检测 PB7 引脚 如果该位置为 1, PB7 上的电压会在内部与 V <sub>REFINT</sub> 进行比较 (包括上升和下降阈值)。这种情况下 PVDT 寄存器的设定无效。
1	保留	-	-	保留
0	PVDE	RW	0	电压检测使能位 0: 电压检测不使能 1: 电压检测使能 如果 SYSCFG_C, G2.PVD_LOCK=1, 则 PVDE 写保护。只有当系统复位后, 写保护才被复位。

### 6.6.3. 电源状态寄存器 (PWR\_SR)

Address offset: 0x14

Reset value: 0x0000 0000 (reset by POR)

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	PVDO	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	R	-	-	-	-	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11	PVDO	R	0	PVD 检测结果输出。 0: 被检测的 V <sub>CC</sub> 或者 PB7 超出 PVD 选择的比较阈值 1: 被检测的 V <sub>CC</sub> 或者 PB7 低于 PVD 选择的比较阈值
10: 0	保留	-	-	保留

## 7. 复位

芯片内设计两种复位，分别是：电源复位和系统复位。

### 7.1. 复位源

#### 7.1.1. 电源复位

电源复位把所有寄存器都复位掉，在以下几种情况下产生：

模拟电路产生的 POR/ BOR，实现对 V<sub>CC</sub> 的检测。

- 当 V<sub>CC</sub> 电压上升到 trigger 值时释放复位；
- 当 V<sub>CC</sub> 电压下降到某一 trigger 值时产生复位。

#### 7.1.2. 系统复位

系统复位把大部分寄存器置成复位值，一些特殊寄存器，如复位标识位寄存器，不会被系统复位。

当产生以下事件时，产生系统复位：

- NRST 管脚的复位
- 窗口看门狗 (WWDG) 复位
- 独立看门狗 (IWDG) 复位
- Cortex-M0+ SYSRESETREQ 软件复位
- 选项字节加载器 (Option byte load, OBL) 复位

#### 7.1.3. NRST 管脚 (External reset)

通过选项字节 (NRST\_MODE 位) 的装载，NRST 管脚可以被配置成下述模式 (具体配置参见选项字节描述)：

- 复位输入

在该模式下，在 NRST 管脚上任何有效的复位信号被传递到内部逻辑，但是芯片内部产生的复位在 NRST 管脚上不输出。

在该配置模式下，GPIO 的 PF2 功能无效。

NRST 管脚输入后，经过去毛刺电路 (去毛刺可配置为禁止) 后产生芯片的外部复位。

- GPIO

在该模式下，该 PIN 可以用作标准的 GPIO，即 PF2。Pin 上的复位功能无效。芯片复位只会由芯片内部产生，并且不能传递到 pin 上。

注：上电复位后，NRST 引脚默认配置为复位输入模式。

#### 7.1.4. 看门狗复位

参见独立看门狗 (IWDG) 和窗口看门狗 (WWDG)。

### 7.1.5. 软件复位

通过置位 ARM M0+的中断和复位控制寄存器的 SYSRESETREQ 位, 可实现软件复位。

### 7.1.6. 选项字节加载器复位

软件通过配置 FLASH\_CR.OBL\_LAUNCH=1, 产生选项字节加载器复位, 从而启动选项字节再次加载。

## 8. 时钟

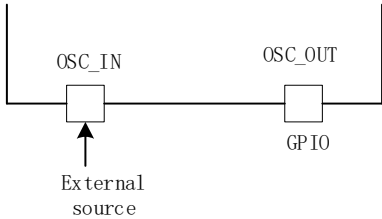
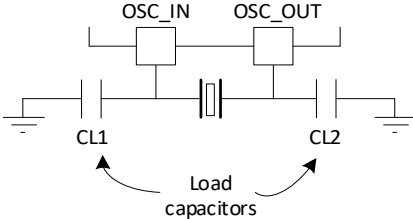
### 8.1. 时钟源

#### 8.1.1. 外部高速时钟 HSE

外部高速时钟 (HSE) 来自两个来源:

- 外部 XTAL OSC+内部起振电路
- 通过 OSC\_IN 结构输入的外部时钟 (HSEBYP=1)

表 8-1 HSE 时钟来源

Clock source	Hardware configuration
外部时钟	
外接晶体	

外部高频 OSC，频率范围 4 ~ 32 MHz。

HSE 时钟的稳定时间由 RCC\_ECSCR.HSE\_STARTUP 寄存器配置。当 HSE 从 OFF 到 ON 时，需要等待稳定时间，稳定后，硬件置位 RCC\_CR.HSERDY 寄存器。当 HSEBYP=1 时，稳定时间比非 bypass 模式减半。

HSE 时钟相关寄存器参考 RCC\_ECSCR。

#### 8.1.2. 内部高速时钟 HSI

内部 RC 振荡器，基准频率可以为 4 MHz、8 MHz、16 MHz、22.12 MHz 和 24 MHz。相较于 XTAL OSC，RC OSC 功耗低，稳定时间短，但精度低。

上电复位后，HSI 校准值需软件加载到 RCC\_ICSCR.HSI\_TRIM 寄存器。系统复位时，该寄存器会随之复位。

从 Stop 模式唤醒后，只能 HSI 作为系统时钟源。

#### 8.1.3. 内部低速时钟 LSI

内部低频 32.768 kHz 时钟。

#### 8.1.4. HSI10M 时钟

该时钟作为低精度时钟，用作 NRST 管脚的滤波计数，以及 Flash 低速 Run 时低功耗处理。

#### 8.1.5. PLL

PLL 模块参考时钟是 HSI 或者 HSE，PLL 输入时钟频率范围要求为 16 ~ 24 MHz，不在此范围不能保证输出时钟频率和稳定性。

PLL 支持 2 倍频或者 3 倍频，在 USB 模块工作时，需要选择 HSI 基准频率为 16 MHz，PLL 倍频系数为 3，从而产生 48 MHz 频率。此时系统时钟若选择 PLL，则最大频率也会限制在 48 MHz。

#### 8.1.6. LSE 时钟

外部 32.768 kHz OSC，用作低功耗时钟。

可以通过配置 LSE\_DRV 在稳定时间和功耗之间做平衡。LSE 稳定时间由 RCC\_ECSCR.LSE\_STARTUP 寄存器配置。

与 HSE 来源类似，LSE 也有两个来源：

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

在 LSE bypass 情况下，稳定时间比非 bypass 模式减半。



## 8.2. 时钟树

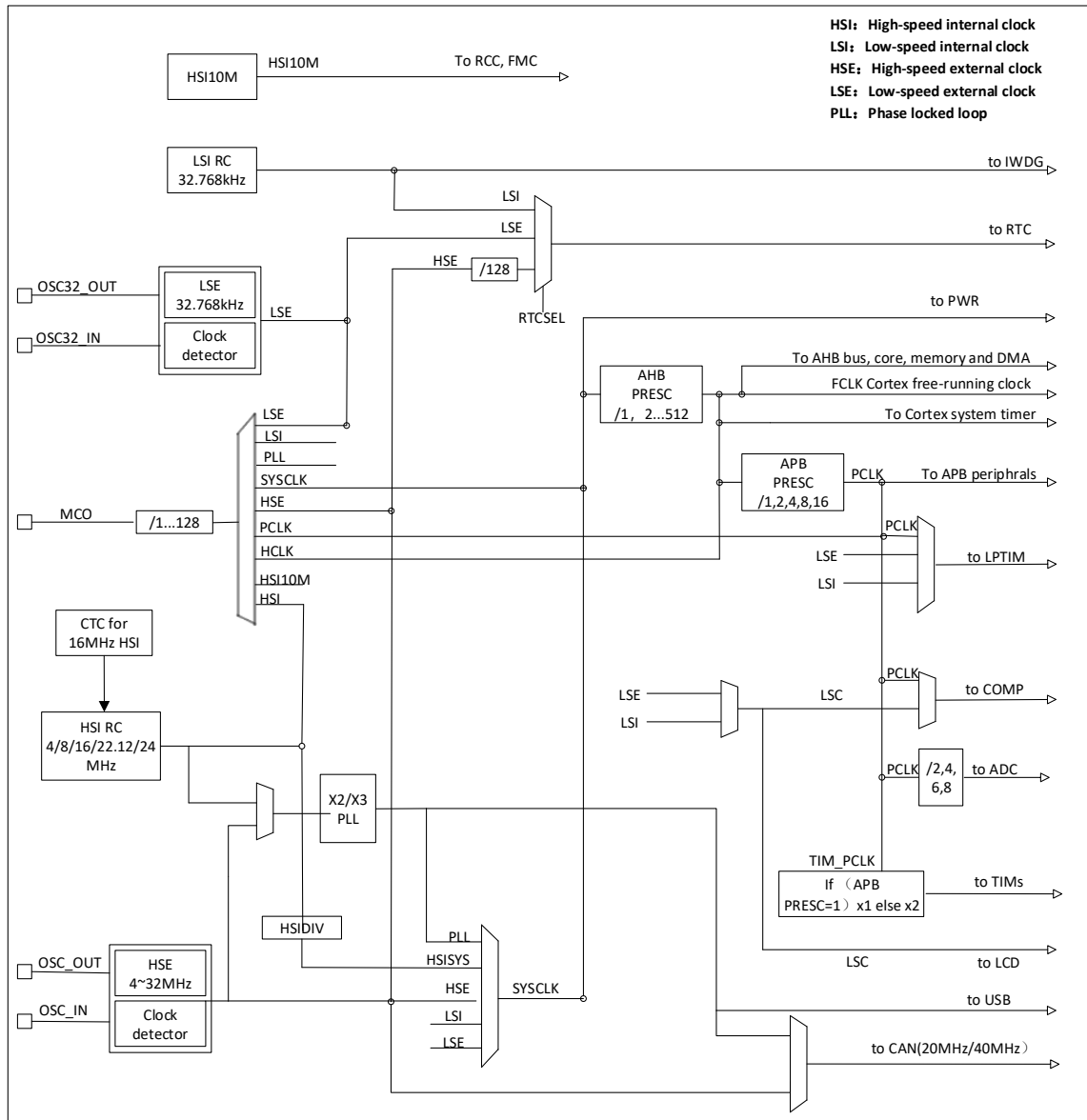


图 8-1 系统时钟结构图

## 8.3. 时钟安全系统 (CSS)

时钟安全主要包括如下方面：

- 时钟配置和状态安全
- 时钟源 HSE 安全
- 时钟源 LSE 安全
- 基于 IWDG 的时钟安全
- 基于 Timer 的时钟安全

### 8.3.1. 时钟配置和状态安全

通过软件周期性地回读时钟配置和状态寄存器，获取系统当前时钟信息，并判断是否与预期一致。

### 8.3.1.1. 时钟源 HSE 监测

通过配置 `RCC_CR.CSSON`，HSE 时钟安全系统可以被软件激活。在这种情况下，HSE 启动后，时钟检测功能被打开。当 HSE 被关闭后，时钟检测功能被关闭。

如果在 HSE 上发现时钟错误，HSE 会被自动关闭，时钟错误事件被送给 TIM1（高级 Timer）和 TIM15/TIM16/TIM17（通用 Timer）的刹车输入端，并产生中断通知软件该错误（时钟安全系统中断，Clock Security System Interrupt CSSI），进而允许 MCU 进行拯救操作。CSSI 被链接到 Cortex-M0+ 的 NMI（非屏蔽中断，Non-maskable interrupt）Exception 向量。

**注：**一旦 CSS 被使能，并且如果 HSE 时钟 Failure，就会产生 CSS 中断，并自动产生一个 NMI。该 NMI 将不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器（`RCC_CICR`）里的 `CSSC` 位来清除 CSS 中断。

如果 HSE 被直接或者间接的用作系统时钟（间接的意思是：它被作为 PLL 的输入端，并且 PLL 被用作系统时钟），时钟 Failure 将导致系统时钟自动切换到 HSI，同时关闭 HSE。如果时钟 Failure 时，HSE 是 PLL 的输入时钟，PLL 也将被关闭。

### 8.3.1.2. 时钟源 LSE 监测

通过配置 `RCC_BDCR.LSECSSON`，LSE 时钟安全系统可以被软件激活。在这种情况下，LSE 启动后，时钟检测功能被打开。当 LSE 被关闭后，时钟检测功能被关闭。

如果在 LSE 上发现时钟错误，LSE 会被自动关闭，时钟错误事件被送给 TIM1（高级 Timer）和 TIM15/TIM16/TIM17（通用 Timer）的刹车输入端，并产生中断通知软件该错误（CSSI），进而允许 MCU 进行拯救操作。CSSI 被链接到 Cortex-M0+ 的 NMI（Non-maskable interrupt）exception 向量。

**注：**一旦 LSECSS 被使能，并且如果 LSE 时钟错误，就会产生 CSS 中断，并自动产生一个 NMI。该 NMI 将不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器（`RCC_CICR`）里的 `CSSC` 位来清除 CSS 中断。

如果 LSE 被用作系统时钟，时钟 Failure 将导致系统时钟自动切换到 LSI，同时关闭 LSE。同时，如果 LPTIM 和 RTC 计数时钟选择 LSE，也会自动切换到 LSI。

## 8.4. 输出时钟能力

为了方便板级应用，节省 BOM 成本，以及 Debug 等的需求，需要芯片提供时钟输出功能。即把下表的 MCO 信号（并分频）通过 GPIO 的复用功能实现时钟输出功能。

表 8-2 输出时钟选择

时钟源	MCO 可输出的时钟源
HSI	√
SYSCLK	√
HSE	√
LSI	√
PLL	√
LSE	√

注意：当对 MCO 时钟源进行切换，以及选择 GPIO AF 功能为 MCO 的起始阶段，MCO 可能会产生毛刺，需要避开该段时间。

## 8.5. 复位/时钟寄存器

该模块的寄存器可以用全字（32 bits）、半字（16 bits）和字节（8 bits）访问。

### 8.5.1. 时钟控制寄存器 (RCC\_CR)

Address offset: 0x00

Reset value: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	PLLRDY	PLLON	Res	ADC_DIV		Res	CSS ON	HSE BYP	HSE RDY	HSE ON
-	-	-	-	-	-	R	RW	-	RW		-	RS	RW	R	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	HSIDIV[2: 0]			HSI RDY	Res	HSION	Res	Res	Res	Res	Res	Res	Res	Res
-	-	RW			R	-	RW	-	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25	PLLRDY	R	0	PLL 时钟准备标志。 硬件置位，表明 PLL 时钟已锁住。 0: PLL 未锁住 1: PLL 已锁住
24	PLLON	RW	0	PLL 使能。 当系统进入 Stop 模式时，硬件会清零该位。当 PLL 时钟作为系统时钟时，该位不能被清零。 0: PLL 关闭 1: PLL 开启
23	保留	-	-	保留
22: 21	ADC_DIV	RW	0	ADC 分频系数 00: 2分频 01: 4分频 10: 6分频 11: 8分频
20	保留	-	-	保留
19	CSSON	RS	0	HSE 时钟安全系统使能。 当该位为1时，如果 HSE OSC ready 则硬件会使能时钟检测模块；如果 HSE 检测失败，则关闭时钟检测模块。 0: 时钟安全系统关闭（时钟检测关闭）

Bit	Name	R/W	Reset Value	Function
				1: 时钟安全系统开启 (如果 HSE 时钟稳定则时钟检测开启, 否则关闭时钟检测)
18	HSEBYP	RW	0	HSE 屏蔽晶振, 选择管脚输入时钟。 该位只有当 HSEON=0时才能写。 0: HSE 晶振不屏蔽, 外部高速时钟选择外部晶振 1: HSE 晶振屏蔽, 外部高速时钟选择外部管脚输入时钟源
17	HSERDY	R	0	HSE 晶振时钟准备标志。 该位由硬件置位表明 HSE 晶振稳定。 0: HSE 晶振没有准备好 1: HSE 晶振准备好了 注: 当 HSEON 清零后, HSERDY 在6个 HSE 时钟周期后清零。
16	HSEON	RW	0	HSE 晶振使能。 当系统进入 Stop 模式时, 硬件会清零该位, 关闭 HSE 晶振。当 HSE 作为系统时钟源时, 该位不能被置0。 0: HSE 晶振关闭 1: HSE 晶振开启
15: 14	保留	-	-	保留
13: 11	HSIDIV	RW	0	HSI 产生 HSISYS 时钟时的分频系数。 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128
10	HSIRDY	R	0	HSI 时钟准备标志。 硬件置位表明 HSI OSC 稳定。该位只有当 HSION=1时才有效。 0: HSI OSC not ready 1: HSI OSC ready
9	保留	-	-	保留
8	HSION	RW	1	HSI 时钟使能。 硬件在进入 stop 模式时, 会根据需要清零该寄存器停止 HSI。 0: HSI OSC 关闭 1: HSI OSC 开启

Bit	Name	R/W	Reset Value	Function
7: 0	保留	-	-	保留

### 8.5.2. 内部时钟源校准寄存器 (RCC\_ICSCR)

Address offset: 0x04

Reset value: 0x00FF 1080

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

Bit	Name	R/W	Reset Value	Function
31: 25	保留	-	-	保留
24: 16	LSI_TRIM	RW	9'h0FF	内部低速时钟频率调整，通过校准，内部低速时钟可以输出32.768 kHz。 校准值保存在 Flash 的如下地址： 32.768 kHz 校准值地址：0x1FFF 0FA4
15: 13	HSI_FS	RW	3'b000	HSI 频率选择： 000: 4 MHz 001: 8 MHz 010: 16 MHz 011: 22.12 MHz 100: 24 MHz >=101: 4 MHz 上电后，默认选择4 MHz，在重载选项字节完成后，硬件切换到8 MHz。
12: 0	HSI_TRIM	RW	13'h1080	时钟频率调整，更改该寄存器的数值可以调整 HSI 的输出频率。寄存器数值每增加1则 HSI 的输出频率增加约 0.2%，总调整范围为4 ~ 24 MHz。 24 MHz/22.12 MHz/16 MHz/8 MHz/4 MHz 对应的校准值保存在 Flash 的如下地址内： 24 MHz 校准值地址：0x1FFF 3220 22.12 MHz 校准值地址：0x1FFF 3218 16 MHz 校准值地址：0x1FFF 3210 8 MHz 校准值地址：0x1FFF 3208 4 MHz 校准值地址：0x1FFF 3200

### 8.5.3. 时钟配置寄存器 (RCC\_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

当时钟源切换时，访问该寄存器有1或者2个时钟的等待周期。

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

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

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30: 28	MCOPRE[2: 0]	RW	0	MCO (microcontroller clock output) 分频系数。软件控制这些位，设置 MCO 输出的分频系数： 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128 在 MCO 输出使能前要配置该分频系数。
27: 24	MCOSEL[3: 0]	RW	0	MCO 选择 0000: no clock, MCO output disabled 0001: SYSCLK 0010: HSI10M 0011: HSI 0100: HSE 0101: PLL CLK 0110: LSI 0111: LSE 1000: HCLK 1001: PCLK 其它: no clock

Bit	Name	R/W	Reset Value	Function
				注：在时钟启动或者切换阶段可能会出现输出时钟不完整的情况。
23: 15	保留	-	-	保留
14: 12	PPRE[2: 0]	RW	0	该位由软件控制。为了产生 PCLK 时钟，它设置 HCLK 的分频系数如下： 0xx: 1 100: 2 101: 4 110: 8 111: 16
11: 8	HPRE[3: 0]	RW	0	AHB 时钟分频系数。 软件控制该位。为了产生 HCLK 时钟，它设置 SYSCLK 的分频系数如下： 0xxx: 1 1000: 2 1001: 4 1010: 8 1011: 16 1100: 64 1101: 128 1110: 256 1111: 512 为了保证系统正常工作，需要根据 VR 电源情况配置合适频率。 注：建议逐级切换分频系数。
7: 6	保留	-	-	保留
5: 3	SWS[2: 0]	R	0	系统时钟切换状态位 这些位由硬件控制，表明当前哪个时钟源被用作系统时钟： 000: HSYSYS 001: HSE 010: PLL CLK 011: LSI 100: LSE 其它：保留
2: 0	SW[2: 0]	RW	0	系统时钟源选择位。 这些位由软件和硬件控制，用来选择系统时钟：

Bit	Name	R/W	Reset Value	Function
				000: HSISYS 001: HSE 010: PLL CLK 011: LSI 100: LSE 其余: 保留 硬件配置为 HSISYS 的情况包括: 1) 系统从 Stop 模式退出 2) 软件配置001 (HSE), 出现 HSE 失效 (HSE 为系统时钟源)

#### 8.5.4. PLL 配置寄存器 (RCC\_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 4	保留	-	-	保留
3: 2	PLLMUL[1: 0]	RW	2'b0	PLL 倍频系数 00: x2 01: x3 11: 保留
1: 0	PLLSRC[1: 0]	RW	0	PLL 时钟源选择。 00: No clock 01: 保留 10: HSI 11: HSE

#### 8.5.5. 外部时钟源控制寄存器 (RCC\_ECSCR)

Address offset: 0x10

Reset value: 0x0003 0003



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	LSE_STARTUP		Res	Res	LSE_DRV	
-	-	-	-	-	-	-	-	-	-	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	HSE_STARTUP		Res	HSE_DRV	
-	-	-	-	-	-	-	-	-	-	-	RW			RW	

Bit	Name	R/W	Reset Value	Function
31: 22	保留	-	-	保留
21: 20	LSE_STARTUP	RW	0	<p>LSE 晶振稳定时间选择。</p> <p>LSEBYP=0:</p> <p>00: 4096个 LSE 时钟周期</p> <p>01: 2048个 LSE 时钟周期</p> <p>10: 8192个 LSE 时钟周期</p> <p>11: 不计稳定时间, 直接输出</p> <p>LSEBYP=1:</p> <p>00: 2048个 LSE 时钟周期</p> <p>01: 1024个 LSE 时钟周期</p> <p>10: 4096个 LSE 时钟周期</p> <p>11: 不计稳定时间, 直接输出</p>
19: 18	保留	-	-	保留
17: 16	LSE_DRV	RW	0x3	<p>低速晶振驱动能力选择。</p> <p>00: 保留;</p> <p>01: 弱驱动能力</p> <p>10: 强驱动能力 (推荐)</p> <p>11: 最强驱动能力</p> <p>注: 需要根据晶振特性、负载电容以及电路板的寄生参数选择适当的驱动能力。驱动能力越大则功耗越大, 驱动能力越弱则功耗越小。</p>
15: 5	保留	-	-	保留
4: 3	HSE_STARTUP	RW	0	<p>HSE 稳定时间选择。</p> <p>HSEBYP=0:</p> <p>00: 4096个 HSE 时钟</p> <p>01: 2048个 HSE 时钟</p> <p>10: 8192个 HSE 时钟</p> <p>11: 不计稳定时间, 直接输出</p> <p>HSEBYP=1:</p> <p>00: 2048个 HSE 时钟</p> <p>01: 1024个 HSE 时钟</p>

Bit	Name	R/W	Reset Value	Function
				10: 4096个 HSE 时钟 11: 不计稳定时间, 直接输出
2	保留	-	-	保留
1: 0	HSE_DRV	RW	0x3	高速晶振驱动能力选择。 00: 保留; 01: 弱驱动能力 10: 强驱动能力 (推荐) 11: 最强驱动能力 注: 需要根据晶振特性、负载电容以及电路板的寄生参数选择适当的驱动能力。驱动能力越大则功耗越大, 驱动能力越弱则功耗越小。

### 8.5.6. 时钟中断使能寄存器 (RCC\_CIER)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 6	保留	-	-	保留
5	PLLRDYIE	RW	0	PLL 准备中断使能。 0: 禁止 1: 使能
4	HSERDYIE	RW	0	HSE 时钟准备中断使能。 0: 禁止 1: 使能
3	HSIRDYIE	RW	0	HSI 时钟准备中断使能。 0: 禁止 1: 使能
2	保留	-	-	保留
1	LSERDYIE	RW	0	LSE 时钟准备中断使能。 0: 禁止 1: 使能

Bit	Name	R/W	Reset Value	Function
0	LSIRDYIE	RW	0	LSI 时钟准备中断使能。 0: 禁止 1: 使能

### 8.5.7. 时钟中断标志寄存器 (RCC\_CIFR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9	LSECSSF	R	0	LSE 时钟安全系统 (CSS) 中断标志。 当硬件检测 LSE OSC 时钟失败时置位该寄存器。 0: LSE 时钟检测失败中断未产生; 1: LSE 时钟检测失败中断产生; 写 LSECSSC 寄存器1清零该位。
8	CSSF	R	0	HSE 时钟安全系统中断标识位。 当硬件检测 HSE OSC 时钟失败时置位该寄存器。 0: HSE 时钟检测失败中断未产生; 1: HSE 时钟检测失败中断产生; 写 CSSC 寄存器1清零该位。
7: 6	保留	-	-	保留
5	PLLRDYF	R	0	PLL 准备中断标志。 当 PLL lock 并且 PLLRDYDIE=1时, 硬件置位该寄存器。 0: PLL lock 中断未产生; 1: PLL lock 中断产生; 写 PLLRDYC 寄存器1清零该位。
4	HSERDYF	R	0	HSE 准备中断标识位 当 HSE 稳定并且 HSERDYIE 使能, 该位由硬件置位。软件通过置位 HSERDYC 位, 清零该位。 0: 无由 HSE 引起的时钟准备中断 1: 有由 HSE 引起的时钟准备中断

Bit	Name	R/W	Reset Value	Function
				写 HSERDYC 寄存器1清零该位
3	HSIRDYF	R	0	HSI 时钟准备中断标志。 HSI 时钟稳定并且 HSIRDYIE=1时, 硬件置位该寄存器。 0: HSI 时钟准备中断未产生; 1: HSI 时钟准备中断产生; 写 HSIRDYC 寄存器1清零该位。
2	保留	-	-	保留
1	LSERDYF	R	0	LSE RDY 时钟准备中断标志。 LSE 时钟稳定并且 LSERDYDIE=1时, 硬件置位该寄存器。 0: LSE RDY 时钟准备中断未产生; 1: LSE RDY 时钟准备中断产生; 写 LSERDYC 寄存器1清零该位。
0	LSIRDYF	R	0	LSI 准备中断标识位 当 LSI 稳定并且 LSIRDYIE 使能, 该位由硬件置位。软件通过置位 LSIRDYC 位, 清零该位。 0: 无由 LSI 引起的时钟准备中断 1: 有由 LSI 引起的时钟准备中断 写 LSIRDYC 寄存器1清零该位。

### 8.5.8. 时钟中断清除寄存器 (RCC\_CICR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9	LSECSSC	W	0	LSE 时钟安全系统 (CSS) 中断标志清零。 0: 无影响 1: 清零 LSECSSF 标志
8	CSSC	W	0	时钟安全中断清零位。 0: 无影响

				1: 清除 CSSF 标志位
7: 6	保留	-	-	保留
5	PLLRDYC	W	0	PLL 准备中断标志清零。 0: 无影响 1: 清零 PLLRDYF 标志
4	HSERDYC	W	0	HSE 准备标志清零。 0: 无影响 1: 清除 HSERDYF 位
3	HSIRDYC	W	0	HSI 准备标志清零。 0: 无影响 1: 清除 HSIRDYF 位
2	保留	-	-	保留
1	LSERDYC	W	0	LSE 准备中断标志清零。 0: 无影响 1: 清零 LSERDYF 标志
0	LSIRDYC	W	0	LSI 准备标志清零。 0: 无影响 1: 清除 LSIRDYF 位

### 8.5.9. I/O 接口复位寄存器 (RCC\_IOPRSTR)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 6	保留	-	-	保留
5	GPIOFRST	RW	0	I/O Port F 复位。 0: 无影响 1: Port F I/O 复位
4: 3	保留	-	-	保留
2	GPIOCRST	RW	0	I/O Port C 复位。 0: 无影响

Bit	Name	R/W	Reset Value	Function
				1: Port C I/O 复位
1	GPIOBRST	RW	0	I/O Port B 复位。 0: 无影响 1: Port B I/O 复位
0	GPIOARST	RW	0	I/O Port A 复位。 0: 无影响 1: Port A I/O 复位

### 8.5.10. AHB 外设复位寄存器 (RCC\_AHBRSTR)

Address offset: 0x28

Reset value: 0x0000 0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件将复位清零。

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

Bit	Name	R/W	Reset Value	Function
31: 25	保留	-	-	保留
24	DIVRST	RW	0	除法器模块复位。 0: 无影响 1: 除法器模块复位
23: 13	保留	-	-	保留
12	CRCRST	RW	0	CRC 模块复位。 0: 无影响 1: CRC 模块复位
11: 9	保留	-	-	保留
8: 1	保留	-	-	保留
0	DMARST	RW	0	DMA 复位。 0: 无影响 1: DMA 模块复位

## 8.5.11. APB 外设复位寄存器 1 (RCC\_APBSTR1)

Address offset: 0x2C

Reset value: 0x0000 0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件将复位清零。

31	30	29	28	27	26	25	24	23	22	21	20
LPTIM RST	OPA RST	DAC RST	PWR RST	CTC RST	Res	CAN RST	Res	USB RST	I2C2 RST	I2C1 RST	Res
RW	RW	RW	RW	RW	-	RW	-	RW	RW	RW	-
19	18	17	16	15	14	13	12	11	10	9	8
USART4 RST	USART3 RST	USART2 RST	Res	Res	SPI2 RST	Res	Res	WWDG RST	RTC APB RST	Res	Res
RW	RW	RW	-	-	RW	-	-	RW	RW	-	-
7	6	5	4	3	2	1	0				
Res	Res	TIM7 RST	TIM6 RST	Res	Res	TIM3 RST	TIM2 RST				
-	-	RW	RW	-	-	RW	RW				

Bit	Name	R/W	Reset Value	Function
31	LPTIMRST	RW	0	LPTIM 模块复位。 0: 无影响 1: 该模块复位
30	OPARST	RW	0	OPA 模块复位。 0: 无影响 1: 该模块复位
29	DACRST	RW	0	DAC 模块复位。 0: 无影响 1: 该模块复位
28	PWRRST	RW	0	Power 接口模块复位。 0: 无影响 1: 该模块复位
27	CTCRST	RW	0	CTC 模块复位。 0: 无影响 1: 该模块复位
26	保留	-	-	保留
25	CANRST	RW	0	CAN 模块复位。 0: 无影响 1: 该模块复位

Bit	Name	R/W	Reset Value	Function
24	保留	-	-	保留
23	USBRST	RW	0	USB 模块复位。 0: 无影响 1: 该模块复位
22	I2C2RST	RW	0	I2C2模块复位。 0: 无影响 1: 该模块复位
21	I2C1RST	RW	0	I2C1模块复位。 0: 无影响 1: 该模块复位
20	保留	-	-	保留
19	USART4RST	RW	0	USART4模块复位。 0: 无影响; 1: 该模块复位
18	USART3RST	RW	0	USART3模块复位。 0: 无影响 1: 该模块复位
17	USART2RST	RW	0	USART2模块复位。 0: 无影响 1: 该模块复位
16: 15	保留	-	-	保留
14	SPI2RST	RW	0	SPI2模块复位。 0: 无影响 1: 该模块复位
13: 12	保留	-	-	保留
11	WWDGRST	RW	0	WWDG 模块复位。 0: 无影响 1: 该模块复位
10	RTCAPBRST	RW	0	RTC 模块 APB 复位。 0: 无影响 1: 该模块复位
9: 6	保留	-	-	保留
5	TIM7RST	RW	0	TIM7模块复位。 0: 无影响 1: 该模块复位
4	TIM6RST	RW	0	TIM6模块复位。 0: 无影响; 1: 该模块复位



Bit	Name	R/W	Reset Value	Function
3: 2	保留	-	-	保留
1	TIM3RST	RW	0	TIM3模块复位。 0: 无影响 1: 该模块复位
0	TIM2RST	RW	0	TIM2模块复位。 0: 无影响 1: 该模块复位

### 8.5.12. APB 外设复位寄存器 2 (RCC\_APBSTR2)

Address offset: 0x30

Reset value: 0x0000 0000

该寄存器由软件置位和清零，软件置位后，该模块维持复位直到软件将复位清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								LCD RST	COMP3 RST	COMP2 RST	COMP1 RST	Res	TIM17 RST	TIM16 RST	TIM15 RST
-								RW	RW	RW	RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM14 RST	USART1 RST	Res	SPI1 RST	TIM1 RST	MCUDBG RST	ADC RST	Res	Res							SYS CFG RST
RW	RW	-	RW	RW	RW	RW		-							RW

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23	LCDRST	RW	0	LCD 模块复位。 0: 无影响 1: 该模块复位
22	COMP3RST	RW	0	COMP3模块复位。 0: 无影响 1: 该模块复位
21	COMP2RST	RW	0	COMP2模块复位。 0: 无影响 1: 该模块复位
20	COMP1RST	RW	0	COMP1模块复位。 0: 无影响 1: 该模块复位
19	保留	-	-	保留
18	TIM17RST	RW	0	TIM17模块复位。

Bit	Name	R/W	Reset Value	Function
				0: 无影响 1: 该模块复位
17	TIM16RST	RW	0	TIM16模块复位。 0: 无影响 1: 该模块复位
16	TIM15RST	RW	0	TIM15模块复位。 0: 无影响 1: 该模块复位
15	TIM14RST	RW	0	TIM14模块复位。 0: 无影响 1: 该模块复位
14	USART1RST	RW	0	USART1模块复位。 0: 无影响 1: 该模块复位
13	保留	-	-	保留
12	SPI1RST	RW	0	SPI1模块复位。 0: 无影响 1: 该模块复位
11	TIM1RST	RW	0	TIM1模块复位。 0: 无影响 1: 该模块复位
10	MCUDBG_RST	RW	0	MCU Debug 模块复位。 0: 无影响 1: 该模块复位
9	ADCRST	RW	0	ADC 模块复位。 0: 无影响 1: 该模块复位
8: 1	保留	-	-	保留
0	SYSCFGRST	RW	0	SYSCFG 模块复位。 0: 无影响 1: 该模块复位

### 8.5.13. I/O 接口时钟使能寄存器 (RCC\_IOPENR)

Address offset: 0x34

Reset value: 0x0000 0000

该寄存器由软件置位和清零。

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	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	GPIOF EN	Res	Res	GPIOC EN	GPIOB EN	GPIOA EN
-	-	-	-	-	-	-	-	-	-	RW	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 6	保留	-	-	保留
5	GPIOFEN	RW	0	I/O Port F 时钟使能。 0: 时钟禁止 1: 时钟使能
4: 3	保留	-	-	保留
2	GPIOCEN	RW	0	I/O Port C 时钟使能。 0: 时钟禁止 1: 时钟使能
1	GPIOBEN	RW	0	I/O Port B 时钟使能。 0: 时钟禁止 1: 时钟使能
0	GPIOAEN	RW	0	I/O Port A 时钟使能。 0: 时钟禁止 1: 时钟使能

#### 8.5.14. AHB 外设时钟使能寄存器 (RCC\_AHBENR)

Address offset: 0x38

Reset value: 0x0000 0300

该寄存器由软件置位和清零。

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

Bit	Name	R/W	Reset Value	Function
31: 25	保留	-	-	保留
24	DIVEN	RW	0	除法器模块时钟使能。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 使能
23: 13	保留	-	-	保留
12	CRCEN	RW	0	CRC 模块时钟使能。 0: 禁止 1: 使能
11: 10	保留	-	-	保留
9	SRAMEN	RW	1	在 Sleep 模式下, SRAM 的时钟使能控制 0: 在 Sleep 模式该模块时钟关闭 1: 在 Sleep 模式该模块时钟使能 注: 该位仅影响 Sleep 模式该模块的时钟使能, 在 Run 模式, 该模块时钟不会关闭
8	FLASHEN	RW	1	在 Sleep 模式下, Flash 的时钟使能控制 0: 在 Sleep 模式该模块时钟关闭 1: 在 Sleep 模式该模块时钟使能 注: 该位仅影响 Sleep 模式该模块的时钟使能, 在 Run 模式, 该模块时钟不会关闭
7: 1	保留	-	-	保留
0	DMAEN	RW	0	DMA 模块时钟使能。 0: 禁止 1: 使能

### 8.5.15. APB 外设时钟使能寄存器 1 (RCC\_APBENR1)

Address offset: 0x3C

Reset value: 0x0000 0000

该寄存器由软件置位和清零。

31	30	29	28	27	26	25	24	23	22	21
LPTIM EN	OPAEN	DACEN	PWR EN	CTCEN	Res	CANEN	Res	USBEN	I2C2EN	I2C1 EN
RW	RW	RW	RW	RW	-	RW	-	RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10
Res	USART4E N	USART3E N	USART2E N	Res	Res	SPI2EN	Res	Res	WWDG EN	TIM- DIV_EN
-	RW	RW	RW	-	-	-	RW	-	-	RW
9	8	7	6	5	4	3	2	1	0	
Res	Res	Res	Res	TIM7EN	TIM6EN	Res	Res	TIM3 EN	TIM2EN	
-	RW	-	-	RW	RW	-	-	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	LPTIMEN	RW	0	LP Timer1模块时钟使能。 0: 禁止 1: 使能
30	OPAEN	RW	0	OPA 模块时钟使能。 0: 禁止 1: 使能
29	DACEN	RW	0	DAC 模块时钟使能。 0: 禁止 1: 使能
28	PWREN	RW	0	低功耗控制模块时钟使能。 0: 禁止 1: 使能
27	CTCEN	RW	0	CTC 模块时钟使能。 0: 禁止 1: 使能
26	保留	-	-	保留
25	CANEN	RW	0	CAN 模块时钟使能。 0: 禁止 1: 使能
24	保留	-	-	保留
23	USBEN	RW	0	USB 模块时钟使能。 0: 禁止 1: 使能
22	I2C2EN	RW	0	I <sup>2</sup> C2模块时钟使能。 0: 禁止 1: 使能
21	I2C1EN	RW	0	I <sup>2</sup> C1模块时钟使能。 0: 禁止 1: 使能
20	保留	-	-	保留
19	USART4EN	RW	0	USART4模块时钟使能。 0: 禁止 1: 使能
18	USART3EN	RW	0	USART3模块时钟使能。 0: 禁止 1: 使能
17	USART2EN	RW	0	USART2模块时钟使能。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 使能
16: 15	保留	-	-	保留
14	SPI2EN	RW	0	SPI2模块时钟使能。 0: 禁止 1: 使能 该寄存器由硬件系统复位清零。
13: 12	保留	-	-	保留
11	WWDGEN	RW	0	Window WDG 模块时钟使能。 0: 禁止 1: 使能 该寄存器由硬件系统复位清零。
10	TIMDIV_EN	RW	0	TIMER PCLK 频率控制。 0: TIMER PCLK 为系统 PCLK*2, 但频率不会超过 HCLK 1: TIMER PCLK 为系统 PCLK*1
9: 6	保留	-	-	保留
5	TIM7EN	RW	0	TIM7模块时钟使能。 0: 禁止 1: 使能
4	TIM6EN	RW	0	TIM6模块时钟使能。 0: 禁止 1: 使能
3: 2	保留	-	-	保留
1	TIM3EN	RW	0	TIM3模块时钟使能。 0: 禁止 1: 使能
0	TIM2EN	RW	0	TIM2模块时钟使能。 0: 禁止 1: 使能

### 8.5.16. APB 外设时钟使能寄存器 2 (RCC\_APBENR2)

Address offset: 0x40

Reset value: 0x0000 0001

该寄存器由软件置位和清零。

31	30	29	28	27	26	25	24	23	22	21
Res	Res	Res	Res	Res	Res	Res	Res	LCDEN	COMP3EN	COMP2EN
-	-	-	-	-	-	-	-	RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10

COMP1EN	Res	TIM17 EN	TIM16 EN	TIM15 EN	TIM14 EN	USART1E N	Res	SPI1EN	TIM1EN	MCUD- BGEN
RW	-	RW	RW	RW	RW	RW	-	RW	RW	RW
<b>9</b>	<b>8</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>	
ADCEN	Res	Res	Res	Res	Res	Res	Res	Res	SYSCFGEN	
RW	-	-	-	-	-	-	-	-	RW	

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23	LCDEN	RW	0	LCD 模块时钟使能。 0: 禁止 1: 使能
22	COMP3EN	RW	0	COMP3模块时钟使能。 0: 禁止 1: 使能
21	COMP2EN	RW	0	COMP2模块时钟使能。 0: 禁止 1: 使能
20	COMP1EN	RW	0	COMP1模块时钟使能。 0: 禁止 1: 使能
19	保留	-	-	保留
18	TIM17EN	RW	0	TIM17模块时钟使能。 0: 禁止 1: 使能
17	TIM16EN	RW	0	TIM16模块时钟使能。 0: 禁止 1: 使能
16	TIM15EN	RW	0	TIM15模块时钟使能。 0: 禁止 1: 使能
15	TIM14EN	RW	0	TIM14模块时钟使能。 0: 禁止 1: 使能
14	USART1EN	RW	0	USART1模块时钟使能。 0: 禁止 1: 使能
13	保留	-	-	保留
12	SPI1EN	RW	0	SPI1模块时钟使能。

Bit	Name	R/W	Reset Value	Function
				0: 禁止 1: 使能
11	TIM1EN	RW	0	TIM1模块时钟使能。 0: 禁止 1: 使能
10	MCUDBGEN	RW	0	MCUDBG 模块时钟使能。 0: 禁止 1: 使能
9	ADCEN	RW	0	ADC 模块时钟使能。 0: 禁止 1: 使能
8: 1	保留	-	-	保留
0	SYSCFGEN	RW	1	SYSCFG 模块时钟使能。 0: 禁止 1: 使能

### 8.5.17. 外设独立时钟配置寄存器 (RCC\_CCIPR)

Address offset: 0x54

Reset value: 0x0000 0000

该寄存器由软件置位和清零。

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	LPTIM1SEL [1: 0]		Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res		Res	COMP3 SEL	COMP2 SEL	COMP1 SEL	PVD SEL	CAN SEL	Res	Res	Res	Res	Res	Res	Res
-	-	-	RW	RW	RW	RW	RW	RW	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 20	保留	-	-	保留
19: 18	LPTIMSEL[1:0]	RW	2'b00	LPTIM1内部时钟源选择。 00: PCLK 01: LSI 10: No clock 11: LSE
17: 11	保留	-	-	保留
10	COMP3SEL	RW	0	COMP3模块时钟源选择。



Bit	Name	R/W	Reset Value	Function
				0: PCLK 1: LSC (RCC_BDCR.LSCOSEL 选择后的时钟) 注: 在使能 COMP3_FR.FLTEN 之前先配置选择 LSC 时钟。
9	COMP2SEL	RW	0	COMP2模块时钟源选择。 0: PCLK 1: LSC (RCC_BDCR.LSCOSEL 选择后的时钟) 注: 在使能 COMP2_FR2.FLTEN 之前先配置选择 LSC 时钟。
8	COMP1SEL	RW	0	COMP1模块时钟源选择。 0: PCLK 1: LSC (RCC_BDCR.LSCOSEL 选择后的时钟) 注: 在使能 COMP1_FR1.FLTEN 之前先配置该寄存器选择时钟。
7	PVDSEL	RW	0	PVD detect 时钟源选择。 0: PCLK 1: LSC (RCC_BDCR.LSCOSEL 选择后的时钟) 注: 当时钟源选择 PCLK 时, 需要将 PWR_CR1.FLTEN 配置为0. 若使能 FLTEN, 必须选择 LSC 时钟, 且在使能 FLTEN 之前先配置选择 LSC 时钟。
6	CANSEL	RW	0	CAN 模块时钟源选择。 0: PLL 1: HSE
5: 0	保留	-	-	保留

### 8.5.18. RTC 域控制寄存器 (RCC\_BDCR)

Address offset: 0x5C

Reset value: 0x0000 0000, reset by POR/BOR

当连续访问该寄存器时,  $0 \leq \text{wait state} \leq 3$ 。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						LSCO SEL	LSC OEN	Res	Res	Res	Res	Res	Res	Res	BDRST
-						RW	RW	RW	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res					RTCSEL[1: 0]		Res	LSE CSS D	LSEC SSON	Res	Res	LSE- BYP	LSE DY	LSEON
RW	-					RW	RW	-	R	RW	-	-	RW	R	RW

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25	LSCOSEL	RW	0	低速时钟选择。 0: LSI 1: LSE
24	LSCOEN	RW	0	低速时钟使能。 0: 禁止 1: 使能
23: 17	保留	-	-	保留
16	BDRST	RW	0	RTC domain 软复位。 0: 无影响 1: 复位
15	RTCEN	RW	0	RTC 时钟使能。软件设置或者清零。 0: 禁止 1: 使能
14: 10	保留	-	-	保留
9: 8	RTCSEL[1: 0]	RW	0	RTC 时钟源选择。 00: No clock 01: LSE 10: LSI 11: HSE 128分频 一旦 RTC 时钟源选择后不能再改变, 除非以下情况: 1. RTC domain 被复位为00 2. 选择 LSE (LSECSSD=1) 但没有 LSE
7	保留	-	-	保留
6	LSECSSD	R	0	CSS 检测 LSE 失败。 该位由硬件置位, 表明 CSS 检测32.768 kHz OSC (LSE) 失败。 0: 未检测到 LSE 失败 1: 检测 LSE 失败
5	LSECSSON	RW	0	CSS 使能 LSE 时钟。 0: 禁止; 1: 使能; 注: 必须 LSEON=1并且 LSEON=1后才能使能 LSECSSON。 一旦使能该位, 不能再把该位禁止, 除非 LSECSSD=1.
4: 3	保留	-	-	保留
2	LSEBYP	RW	0	LSE OSC bypass

Bit	Name	R/W	Reset Value	Function
				0: 没有旁路, 低速外部时钟选择晶振; 1: 旁路, 低速外部时钟选择外部接口输入时钟; 注: 只有当外部32.768 kHz OSC 禁止 (LSEON=0并且 LSERDY=0) 时才能写该位。
1	LSERDY	R	0	LSE OSC ready. 硬件配置该位为1表明 LSE 时钟 ready。
0	LSEON	RW	0	LSE OSC 使能。 0: 禁止 1: 使能

### 8.5.19. 控制/状态寄存器 (RCC\_CSR)

Address offset: 0x60

Reset value: 0x0800 0000

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

当连续访问该寄存器时,  $0 \leq \text{wait state} \leq 3$ 。

31	30	29	28	27	26	25	24	23	22	21
Res	WWDG RSTF	IWDG RSTF	SFT RSTF	PWR RSTF	PIN RSTF	OBL RSTF	Res	RMVF	Res	Res
-	R	R	R	R	R	R	-	RW	-	-
20	19	18	17	16	15	14	13	12	11	
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
-	-	-	-	-	-	-	-	-	-	
10	9	8	7	6	5	4	3	2	1	0
Res	Res	PINRST _FLTDIS	Res	Res	Res	Res	Res	Res	LSI RDY	LSION
-	-	RW	-	-	-	-	-	-	R	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30	WWDGRSTF	R	0	Window WDG 复位标志。 RMVF 置1会清零该位。
29	IWDGRSTF	R	0	IWDG 复位标志。 RMVF 置1会清零该位。
28	SFTRSTF	R	0	软复位标志。 RMVF 置1会清零该位。
27	PWRRSTF	R	1	BOR/POR/PDR 复位标志。 RMVF 置1会清零该位。

Bit	Name	R/W	Reset Value	Function
26	PINRSTF	R	0	外部 NRST 管脚复位标志。 RMVF 置1会清零该位。
25	OBLRSTF	R	0	Option byte loader 复位标志。 RMVF 置1会清零该位。
24	保留	-	-	保留
23	RMVF	RW	0	需通过软件置1来清零[30: 25]的复位标志。
22: 9	保留	-	-	保留
8	PINRST_FLTDIS	RW	0	NRST 滤波宽度40 $\mu$ s 禁止 0: 使能 HSI_10M, 且滤波40 $\mu$ s 宽度功能使能 1: 滤波功能禁止, 同步到系统时钟产生系统复位
7: 2	保留	-	-	保留
1	LSIRDY	R	0	LSI OSC 稳定标志。 0: LSI 未稳定 1: LSI 已稳定
0	LSION	RW	0	LSI OSC 使能。 0: 禁止 1: 使能 硬件开启模拟 LSI 的情况: 1. 硬件 IWDG 使能 2. LSECSS 使能

## 9. 时钟校准控制器 (CTC)

### 9.1. CTC 简介

时钟校准控制器 (CTC) 采用硬件的方式, 自动校准内部配置为 16 MHz 时的 RC 晶振 (HSI), 并将 3 倍频后的 PLL (48 MHz) 作为 USB\_D 模块时钟源, 下文将 HSI 配置为 16 MHz 并经过 3 倍频产生的 PLL 简称为 PLL48M。对于 USB\_D 模块的时钟源, 必须要求 PLL48M 时钟频率在  $48\text{ MHz} \pm 500\text{ ppm}$  的范围内, 但是没有经过校准的内部晶振无法满足这么高的精度。CTC 模块基于外部高精度的参考信号源来校准 HSI 的时钟频率, 通过自动的或手动的调整校准值, 以得到一个精准的 PLL48M 时钟。

### 9.2. CTC 主要特性

- 三个外部参考信号源: GPIO, LSE 时钟, USB\_D\_SOF
- 提供软件参考同步脉冲;
- 硬件自动校准, 无需软件操作;
- 具有参考信号源捕获和重载功能的 16 bits 校准计数器;
- 用于频率评估和自动校准的 8 bits 时钟校准基值;
- 标志位和中断, 用于指示时钟校准的状态: 校准成功状态 (CKOKIF), 警告状态 (CKWARNIF) 和错误状 (ERRIF)。

### 9.3. CTC 功能描述

#### 9.3.1. CTC 框图

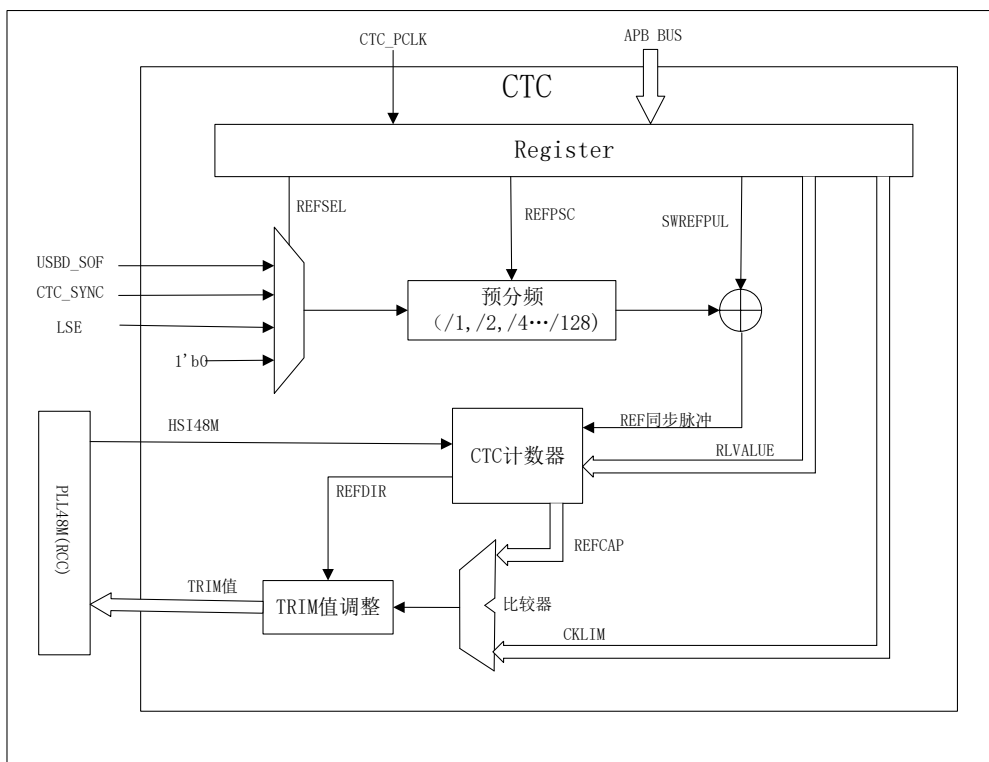


图 9-1 CTC 架构框图

### 9.3.2. REF 同步脉冲发生器

首先，通过设置 CTC\_CTL1 寄存器中的 REFSEL 位来选择参考信号源：GPIO，LSE 时钟输出或者 USBD\_SOF。

然后，可以通过设置 CTC\_CTL1 寄存器中的 REFPOL 位来配置参考信号源同步时的信号极性，通过设置 CTC\_CTL1 寄存器中的 REFPSC 位来产生一个合适的同步时钟频率信号（不大于 48 kHz）。

如果需要使用软件参考脉冲信号，则需要设置 CTC\_CTL0 寄存器中的 SWREFPUL 位为 1。软件参考脉冲信号与外部参考脉冲信号最后进行逻辑‘或’操作。

### 9.3.3. CTC 校准计数器

CTC 时钟校准计数器由 PLL48M 提供时钟。在置位 CTC\_CTL0 寄存器中的 CNTEN 位后，当检测到第一个 REF 同步脉冲信号，计数器开始从 RLVALUE 值（RLVALUE 在 CTC\_CTL1 寄存器中定义）开始向下计数。每次检测到 REF 同步脉冲信号时，计数器重载 RLVALUE 值，同时重新开始向下计数。如果始终检测不到 REF 同步脉冲信号，计数器会向下计数到零，然后再向上计数到  $128 \times \text{CKLIM}$ （CKLIM 在 CTC\_CTL1 中定义），最后停止，直到检测到下一个 REF 同步脉冲信号。一旦检测到 REF 同步脉冲信号，当前 CTC 校准计数器的计数值被捕获存入 CTC\_SR 中的 REFCAP 位，同时，当前计数器的计数方向被存入 CTC\_SR 中的 REFDIR 位。详细内容如下图所示。

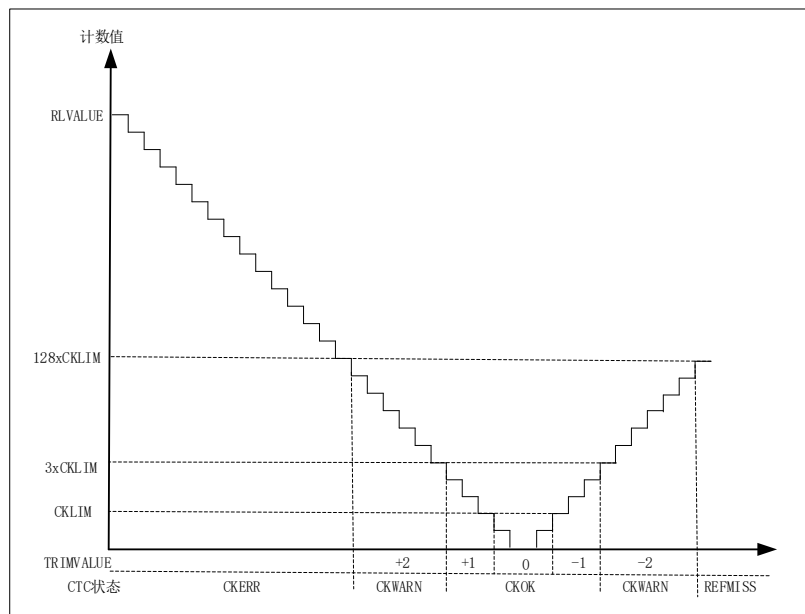


图 9-2 CTC 校准计数器

### 9.3.4. 频率评估和自动校准过程

当 REF 同步脉冲信号出现时，时钟频率评估功能开始执行。如果 REF 同步脉冲信号出现在计数器向下计数的过程中，说明当前时钟频率比期望时钟频率（频率为 48MHz）慢，需要增大 CTC\_CTL0 中的 TRIMVALUE 值（时钟校准值）。如果 REF 同步脉冲信号出现在计数器向上计数的过程中，说明当前时钟频率比期望时钟频率快，需要减小 TRIMVALUE 值。CTC\_SR 中的 CKOKIF 位，CKWARNIF 位，CKERR 位和 REFMISS 位反映了频率评估的状态。

如果 CTC\_CTL0 中的 AUTOTRIM（硬件自动校准模式）位置 1，硬件自动校准模式使能。在这个模式中，如果 REF 同步脉冲信号出现在计数器向下计数的过程中，说明当前时钟频率比期望时钟频率慢，CTC\_CTL0 中的 TRIMVALUE 值会自动增大，来提高当前的时钟频率。反之，如果 REF 同步脉冲信号出现在计数器向上计数的过程中，说明当前时钟频率比期望时钟频率快，TRIMVALUE 值会自动减小，从而减小当前的时钟频率。

- Counter < CKLIM 时，检测到 REF 同步脉冲信号：  
CTC\_SR 中的 CKOKIF 位（时钟校准成功标志位）被置位，同时，如果 CTC\_CTL0 中的 CKOKIE 位（时钟校准完成中断使能位）置 1，将会产生一个中断。如果 CTC\_CTL0 中的 AUTOTRIM 置 1，CTC\_CTL0 中的 TRIMVALUE 值不变。
- CKLIM ≤ Counter < 3 x CKLIM 时，检测到 REF 同步脉冲信号：  
CTC\_SR 中的 CKOKIF 位被置位，同时，如果 CTC\_CTL0 中的 CKOKIE 位置 1，将会产生一个中断。如果 CTC\_CTL0 中的 AUTOTRIM 位置 1，在计数器向下计数过程中，CTC\_CTL0 中的 TRIMVALUE 值将加 1，而在向上计数过程中将减 1。
- 3 x CKLIM ≤ Counter < 128 x CKLIM 时，检测到 REF 同步脉冲信号：  
CTC\_SR 中的 CKWARNIF 位（时钟校准警告中断位）被置位，同时，如果 CTC\_CTL0 中的 CKWARNIE 位（时钟校准警告中断使能位）置 1，将会产生一个中断。如果 CTC\_CTL0 中的 AUTOTRIM 位置 1，在计数器向下计数过程中，CTC\_CTL0 中的 TRIMVALUE 值将加 2，而在向上计数过程中将减 2。
- Counter ≥ 128 x CKLIM，计数器在向下计数过程中，检测到 REF 同步脉冲信号：  
CTC\_SR 中的 CKERR 位（时钟校准错误位）被置位，同时，如果 CTC\_CTL0 中的 ERRIE 位（错误中断使能位）置 1，将会产生一个中断。CTC\_CTL0 中的 TRIMVALUE 值不变。
- Counter = 128 x CKLIM，计数器在向上计数过程中：  
CTC\_SR 中的 REFMIS 位（REF 同步脉冲丢失位）被置位，同时，如果 CTC\_CTL0 中的 ERRIE 位置 1，将会产生一个中断。CTC\_CTL0 中的 TRIMVALUE 值不变。

如果 CTC\_CTL0 中的 TRIMVALUE 的校准值大于 127，将会发生上溢事件，同时，若 TRIMVALUE 的校准值小于 0，将会发生下溢事件。TRIMVALUE 的取值范围为 0~127（上溢事件发生时，TRIMVALUE 值为 127；下溢事件发生时，TRIMVALUE 值为 0）。然后，CTC\_SR 中的 TRIMERR 位（校准值错误位）将会被置位，如果 CTC\_CTL0 中的 ERRIE 位置 1，将会产生一个中断。

### 9.3.5. 软件编程指南

CTC\_CTL1 中 RLVALUE 位和 CKLIM 位是时钟频率评估和硬件自动校准的关键。它们的数值由期望时钟的频率（PLL：48 MHz）和 REF 同步脉冲信号的频率计算得到。理想状态是 REF 同步脉冲信号在 CTC 计数器计数到零时出现，所以 RLVALUE 的值为：

$$RLVALUE = (f_{clock} \div f_{REF}) - 1$$

CKLIM 的值由用户根据时钟的精度来设置，一般建议为步长的一半，所以 CKLIM 的值为：

$$CKLIM = (f_{clock} \div f_{REF}) \times 0.12\% \div 2$$

典型的步长值是 0.12%，f<sub>clock</sub> 是期望时钟的频率（48 MHz），f<sub>REF</sub> 是 REF 同步脉冲信号的频率。

CTC\_CTL0 中 TRIMVALUE 可以在 AUTOTRIM 位为 0 时通过软件写入，但修改 TRIMVALUE 会直接影响 HSI 时钟的频率，因此，不应该随意通过软件修改 TRIMVALUE。建议用户在两次参考信号中间，根据标志位判断修改（详见 频率评估和自动校准过程）；或者若已经存在可靠的值，用户可以直接修改 RCC\_ICSCR 中 HSI\_TRIM 的值。

## 9.4. CTC 寄存器

### 9.4.1. CTC 控制寄存器 0 (CTC\_CTL0)

Address offset: 0x00

Reset value: 0x0000 4000

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	TRIMVALUE[6: 0]							SWR EFP UL	AU- TO- TRIM	CNT EN	Res	ERE- FIE	ERRI E	CKW AR- NIE	CKO KIE	
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14: 8	TRIMVALUE[6: 0]	RW	7'b1000000	PLL48M 校准值。 当 CTC_CTL0 中的 AUTOTRIM 值为 0 时，该位由软件置位和清除，该模式用于软件校准过程。 当 CTC_CTL0 中的 AUTOTRIM 值为 1 时，该位只读，由硬件自动修改，该模式用于硬件校准过程。 TRIMVALUE 的中间值是 64，当 TRIMVALUE 值加 1 时，PLL48M 时钟频率增加大约 48 kHz。当 TRIMVALUE 值减 1 时，PLL48M 时钟频率的减少大约 48 kHz。
7	SWREFPUL	RW	0	软件生成同步参考信号脉冲。 该位由软件置位，并为 CTC 计数器提供一个同步参考脉冲信号。该位由硬件自动清除，读操作时返回 0。 0: 没有影响； 1: 软件产生一个同步参考脉冲信号；
6	AUTOTRIM	RW	0	硬件自动校准模式。 该位由软件置位或清除。当该位置 1 时，硬件自动校准模式使能，通过硬件不断地自动修改 CTC_CTL0 中的



				TRIMVALUE 值, 直到 PLL48M 的时钟频率达到48 MHz。 0: 禁止硬件自动校准模式 1: 使能硬件自动校准模式
5	CNTEN	RW	0	CTC 计数器使能。 该位由软件置位或清除, 用于使能或禁止 CTC 计数器。 当该位置 1 时, 不能修改 CTC_CTL1 的值。 0: 禁止 CTC 计数器 1: 使能 CTC 计数器
4	保留	-	-	保留
3	EREFIE	RW	0	期望参考信号中断使能。 0: 禁止期望参考信号产生中断 1: 使能期望参考信号产生中断
2	ERRIE	RW	0	错误中断使能。 0: 禁止错误中断 1: 使能错误中断
1	CKWARNIE	RW	0	时钟校准警告中断使能。 0: 禁止时钟校准警告中断 1: 使能时钟校准警告中断
0	CKOKIE	RW	0	时钟校准完成中断使能。 0: 禁止时钟校准完成中断 1: 使能时钟校准完成中断

#### 9.4.2. CTC 控制寄存器 1 (CTC\_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

该寄存器只能按字 (32 位) 访问。当 CNTEN 为1时, 不能修改该寄存器的值。

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

Bit	Name	R/W	Reset Value	Function
31	REFPOL	RW	0	参考信号源极性。 该位由软件置位或清除, 用于选择参考信号源的同步极性。

				0: 选择上升沿 1: 选择下降沿
30	保留	-	-	保留
29: 28	REFSEL[1: 0]	RW	2'b10	参考信号源选择。 该位由软件置位或清除, 用于选择参考信号源。 00: 选择 GPIO 输入信号 01: 选择 LSE 时钟 10: 选择 USBD_SOF 信号 11: 保留, 选择 0
27	保留	-	-	保留
26: 24	REFPSC[2: 0]	RW	3'b000	参考信号源预分频。 该位由软件置位或清除。 000: 参考信号不分频 001: 参考信号 2 分频 010: 参考信号 4 分频 011: 参考信号 8 分频 100: 参考信号 16 分频 101: 参考信号 32 分频 110: 参考信号 64 分频 111: 参考信号 128 分频
23: 16	CKLIM[7: 0]	RW	0x22	时钟校准时基限值。 该位由软件置位或清除, 用于定义时钟校准时基限值。该位用于频率评估和自动校准过程。
15: 0	RLVALUE[15: 0]	RW	0xBB7F	CTC 计数器重载值。 该位由软件置位或清除, 用于定义 CTC 计数器的重载值, 当检测到一个同步参考脉冲时, 该值将重载到 CTC 校准计数器中。

### 9.4.3. CTC 状态寄存器 (CTC\_SR)

Address offset: 0x08

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFCAP[15: 0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REF DIR	Res	Res	Res	Res	TRIM ERR	REF MISS	CKE RR	Res	Res	Res	Res	ERE-FIF	ERRI F	CKW AR-NIF	CKO KIF
R	-	-	-	-	R	R	R	-	-	-	-	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 16	REFCAP[15: 0]	R	0	CTC 计数器捕获值。 当检测到一个同步参考脉冲信号时，CTC 校准计数器中的计数值被存入到 REFCAP 位中。
15	REFDIR	R	1	CTC 校准时钟计数方向。 当检测到一个同步参考脉冲信号时，CTC 校准计数器的计数方向被存入 REFDIR 位中。 0: 向上计数 1: 向下计数
14: 11	保留	-	-	保留
10	TRIMERR	R	0	校准值错误位。 当 CTC_CTL0 中的 TRIMVALUE 值发生上溢或下溢时，该位由硬件置位。若 CTC_CTL0 中的 ERRIE 位置 1，则会产生一个中断。通过写 1 到 CTC_INTIC 中的 ERRIC 位，可以将 TRIMERR 位清零。 0: 无校准值错误发生 1: 发生校准值错误
9	REFMISS	R	0	同步参考脉冲信号丢失。 当同步参考脉冲信号丢失时，该位由硬件置位。当 CTC 校准计数器在增计数的过程中计数到 128 x CKLIM 都没有检测到同步参考脉冲信号时，REFMISS 位置位。说明当前时钟太快，无法校准到期望频率值，或者有其他错误产生。通过写 1 到 CTC_INTIC 中的 ERRIC 位，可以将 REFMISS 位清零。 0: 无同步参考脉冲信号丢失 1: 同步参考脉冲信号丢失 注：为防止清除 REFMISS 及同时产生的 ERRIF 后再度置起，可以通过先清除 CNTEN 位，再反复写 ERRIC 位直到标志位不再置起。
8	CKERR	R	0	时钟校准错误位。 当时钟校准错误产生时，该位由硬件置位。当 CTC 校准计数器计数值在减计数的过程中大于或等于 128 x CKLIM，并检测到同步参考脉冲信号时，CKERR 置位，说明当前时钟太慢，无法校准到期望频率值。当 CTC_CTL0 中的 ERRIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTIC 中的 ERRIC 位，可以将 CKERR 位清零。 0: 无时钟校准错误发生

Bit	Name	R/W	Reset Value	Function
				1: 发生时钟校准错误
7: 4	保留	-	-	保留
3	EREFIF	R	0	<p>期望参考中断标志位。</p> <p>当 CTC 校准时钟计数器计数到 0 时, 检测到参考信号, 该位由硬件置位。当 CTC_CTL0 中的 EREFIE 置 1 时, 产生一个中断。通过写 1 到 CTC_INTC 中的 EREFIC 位, 可以将 EREFIF 位清零。</p> <p>0: 无期望参考信号产生 1: 期望参考信号产生</p>
2	ERRIF	R	0	<p>错误中断标志位。</p> <p>当发生一个错误时, 该位由硬件置位。只要有 TRIMERR, REFMISS 或者 CKERR 错误发生时, 该位置位。当 CTC_CTL0 中的 ERRIE 置位时, 产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位, 可以将 ERRIF 位清零。</p> <p>0: 无错误发生 1: 发生错误</p>
1	CKWARNIF	R	0	<p>时钟校准警告中断标志位。</p> <p>当时钟校准警告产生时, 该位由硬件置位。当 CTC 校准计数器计数值大于或等于 <math>3 \times \text{CKLIM}</math> 且小于 <math>128 \times \text{CKLIM}</math>, 并检测到同步参考脉冲信号时, CKWARNIF 置位。这说明当前时钟频率太慢或者太快, 但可以通过校准达到期望频率值。当时钟校准警告产生时, TRIMVALUE 值加 2 或者减 2。当 CTC_CTL0 中的 CKWARNIE 置 1 时, 产生一个中断。通过写 1 到 CTC_INTC 中的 CKWARNIC 位, 可以将 CKWARNIF 位清零。</p> <p>0: 无时钟校准警告发生 1: 有时钟校准警告发生</p>
0	CKOKIF	R	0	<p>时钟校准成功中断标志位。</p> <p>当时钟校准成功时, 该位由硬件置位。若在 CTC 校准计数器计数值小于 <math>3 \times \text{CKLIM}</math> 时, 检测到同步参考脉冲信号, CKOKIF 置位。说明当前时钟频率正常, 可以使用, 不需要通过 TRIMVALUE 值进行时钟校准。当 CTC_CTL0 中的 CKOKIE 置 1 时, 产生一个中断。通过写 1 到 CTC_INTC 中的 CKOKIC 位, 可以将 CKOKIF 位清零。</p> <p>0: 时钟校准未成功 1: 时钟校准成功</p>

#### 9.4.4. CTC 中断清除寄存器 (CTC\_INTC)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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	ERE- FIC	ERR IC	CKWAR NIC	CKOK IC
-	-	-	-	-	-	-	-	-	-	-	-	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 4	保留	-	-	保留
3	EREFIC	W	0	EREFIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_SR 中的 EREFIF 位，写 0 没影响。
2	ERRIC	W	0	ERRIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_SR 中的 ERRIF 位，TRIMERR 位，REFMISS 位和 CKERR 位，写 0 没影响。
1	CKWARNIC	W	0	CKWARNIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_SR 中的 CKWARNIF 位，写 0 没影响。
0	CKOKIC	W	0	CKOKIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_SR 中的 CKOKIF 位，写 0 没影响。

## 10. 通用 I/O (GPIO)

### 10.1. 通用 IO 简介

GPIO 包含 PA[15: 0], PB[15: 0], PC[15: 0]和 PF[9: 0], 每个 GPIO 端口有:

- 4 个 32 位配置寄存器 (GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR、GPIOx\_PUPDR)
- 2 个 32 位数据寄存器 (GPIOx\_IDR 和 GPIOx\_ODR)
- 1 个 32 位位置位/复位寄存器 (GPIOx\_BSRR)
- 1 个 32 位锁定寄存器 (GPIOx\_LCKR)
- 2 个复用功能选择寄存器 (GPIOx\_AFRH 和 GPIOx\_AFLR)
- 1 个 32 位复位寄存器 (GPIOx\_BRR)

### 10.2. 通用 IO 功能描述

- 寄存器支持 Fast IO /AHB 总线读写
- 输出状态: 推挽或者开漏 + 上拉/下拉
- 数据寄存器 (GPIOx\_ODR) 或者外设 (复用功能输出) 数据输出
- 每个 I/O 可进行速度选择
- 输入状态: 浮空, 上拉/下拉, 模拟
- 数据输入送给输入数据寄存器 (GPIOx\_IDR) 或者外设 (复用功能输入)
- 位置位/复位寄存器 (GPIOx\_BSRR), 允许对 GPIOx\_ODR 的位写访问
- 锁定机制 (GPIOx\_LCKR) 会冻结 I/O 口配置功能
- 模拟功能
- 复用功能选择寄存器 (每个 IO 口最多 16 种复用功能)
- 单周期内快速翻转的能力
- 高度灵活的 I/O 多路选择功能, 使得 I/O 口作为 GPIO, 或者作为各种外设接口功能

### 10.3. 通用 IO 功能描述

每个 GPIO 的每个位, 可以通过软件编程, 进行几种模式的配置:

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出, 带上拉或者下拉
- 推挽输出, 带上拉或者下拉
- 带上拉或者下拉的复用功能的推挽
- 带上拉或者下拉的复用的开漏

每个 I/O 口可以自由编程，然而 I/O 端口寄存器必须按全字、半字或者字节被访问。GPIOx\_BSRR 和 GPIOx\_BRR 寄存器允许对任何 GPIOx\_ODR 寄存器的读/更改的独立访问。这样，在读和更改访问之间产生 IRQ 时不会发生危险。

下图给出了一个 I/O 端口（1bit）的基本结构

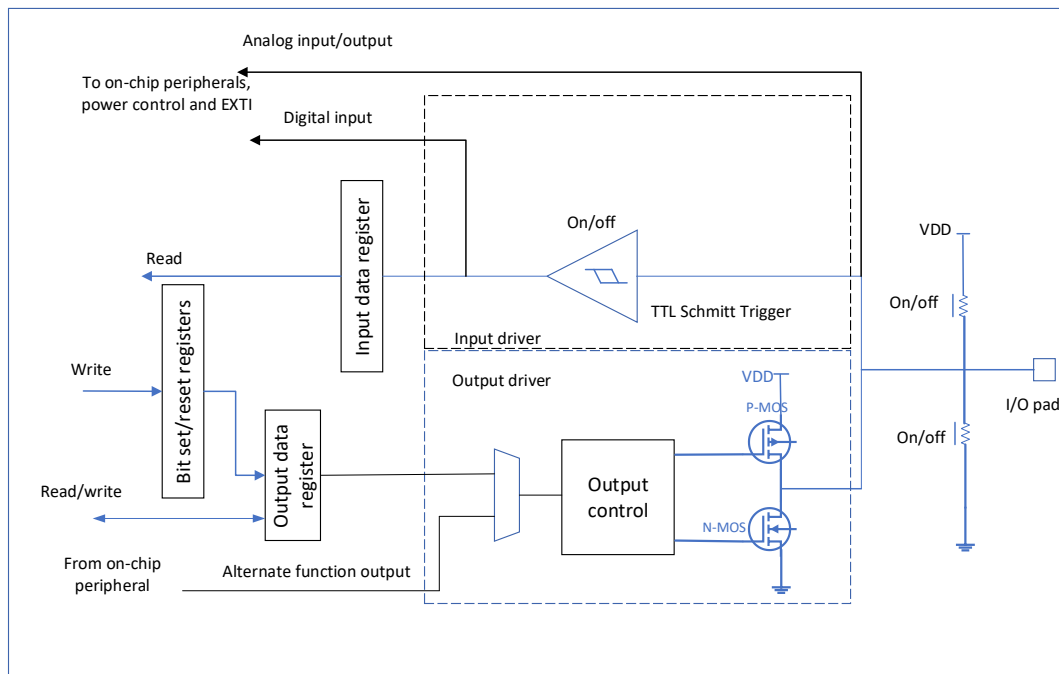


图 10-1 IO 端口位的基本结构

### 10.3.1. 通用 I/O (GPIO)

复位期间和复位后，复用功能未被激活，大多数 IO 被配置为模拟模式。

Debug 引脚默认被置于复用功能上拉或下拉模式：

- PA14-SWCLK：置于下拉模式
- PA13-SWDIO：置于上拉模式

Boot 引脚默认置于输入模式，下拉模式

- PF8-Boot：置于下拉模式

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。可以在推挽模式或开漏模式下使用输出驱动器（只能驱动低电平，高电平为高阻态）。

输入数据寄存器（GPIOx\_IDR）在每个 AHB 时钟会获取 I/O 脚上的电平。

所有的 GPIO 引脚都有内部的弱上拉和弱下拉电阻，可以通过 GPIOx\_PUPDR 寄存器使能或者不使能该功能。

### 10.3.2. I/O 管脚复用功能多路选择和映射

设备 I/O 口通过多路选择器连着板级的外设/模块，一个外设每次可以通过复用功能连着一个 IO 口。这样可以避免同一个 IO 口上的可用外设不会出现冲突。

每个 I/O 口上的多路选择器多达 16 种复用功能输入 (AF0 到 AF15)，可通过寄存器 GPIOx\_AFRL (针对 pin 0 到 7) 和 GPIOx\_AFRH (针对 pin 8 到 15) 来配置。

- 刚复位后，多路选择器默认为 AF0。I/O 口的复用功能模式通过寄存器 GPIOx\_MODER 配置
- 每个脚的复用功能分布请参考数据手册

除了这种灵活的多路选择器架构，每个外设还有复用功能可以分布在不同的 I/O 口上，以便在更小的封装上使用到的外设数量最优化。

用户按照如下说明去配置 IO：

- 调试功能：每次复位后，这些调试功能脚就是默认为调试器立即可用的复用功能脚
- GPIO：在 GPIOx\_MODER 将对应 I/O 口配置为输出、输入或者模拟模式
- 外设复用功能：
  - 寄存器 GPIOx\_AFRL 或者 GPIOx\_AFRH 配置对应的 I/O 为复用功能 x (x=0...15)
  - 寄存器 GPIOx\_OTYPER, GPIOx\_PUPDR 和 GPIOx\_OSPEEDER 分别配置类型，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
  - 总线读写功能：GPIO 模块除了支持 Fast IO 读写 GPIOx 寄存器外，也支持通过 AHB 总线读写寄存器，寄存器访问方式由 SYSCFG 模块的 GPIO\_AHB\_SEL bit 选择。当 GPIO\_AHB\_SEL bit 为 0 时，只能通过 Fast IO 访问 GPIOx 寄存器；当 GPIO\_AHB\_SEL 为 1 时，只有通过 AHB 总线访问 GPIOx 寄存器。
  - AHB 总线访问 GPIO 寄存器的方式，除了支持 CPU 外，还支持 DMA，即 DMA 可以通过 AHB 总线直接访问 GPIO 寄存器。
- 额外功能
  - 无论 IO 口配置成任何模式，ADC 和 COMP 功能均在 ADC 和 COMP 模块的寄存器中使能。当 IO 口用做 ADC 或者 COMP 使用时，需要通过寄存器 GPIOx\_MODER 将该口配置为模拟模式
  - 对于晶振额外功能，在相应的 PWR 与 RCC 模块寄存器里配置各自功能。这些配置比标准的 GPIO 配置具有更高优先级。

### 10.3.3. I/O 控制寄存器

每个 GPIO 口有四个 32 位内存映射控制寄存器 (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR and GPIOx\_PUPDR)，可以配置多达 16 个 I/O 口。寄存器 GPIOx\_MODER 用来选择 I/O 模式 (输入、输出、复用、模拟)。寄存器 GPIOx\_OTYPER 和 GPIOx\_OSPEEDR 用来选择输出类型 (推挽或开漏) 和速度。无论采用哪种 I/O 方向，寄存器 GPIOx\_PUPDR 都用来选择上拉/下拉。

### 10.3.4. I/O 数据寄存器

每个 GPIO 有 2 个 16 位内存映射的数据寄存器：输入和输出数据寄存器 (GPIOx\_IDR 和 GPIOx\_ODR)。寄存器 GPIOx\_ODR 保存了要输出的数据，可读可写。输入数据寄存器 (GPIOx\_IDR) 用来保存 I/O 口上的电平状态，只读。



### 10.3.5. I/O 数据按位处理

置位/复位寄存器 (GPIOx\_BSRR) 是一个 32 位寄存器, 可以将输出数据寄存器 (GPIOx\_ODR) 的每位单独置位和复位。置位/复位寄存器位数是输出寄存器 (GPIOx\_ODR) 的两倍。

GPIOx\_ODR 的每一位对应 GPIOx\_BSRR 的两个控制位: BS (i) 和 BR (i)。位 BS (i) 置 1 可将 GPIOx\_ODR 对应位置 1, 位 BR (i) 置 1 可将 GPIOx\_ODR 对应位清 0。

寄存器 GPIOx\_BSRR 任意位写 0 并不影响寄存器 GPIOx\_ODR 对应的位。如果 GPIOx\_BSRR 对某一位同时清 0 和置 1 操作, 置 1 操作具有优先权。

使用寄存器 GPIOx\_BSRR 改变寄存器 GPIOx\_ODR 的对应位只有一次性的作用, 并不会锁定寄存器 GPIOx\_ODR 的位。寄存器 GPIOx\_ODR 也可以直接访问。寄存器 GPIOx\_BSRR 只是提供一种原子位操作处理方式。

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。

寄存器 GPIOx\_LCKR 通过一系列特殊写时序可以冻结 IO 的控制寄存器, 包括 GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR, GPIOx\_AFRL 和 GPIOx\_AFRH。

一个特殊写/读时序可以操作寄存器 GPIOx\_LCKR。当该寄存器的 Bit16 写入正确的时序, LCKR[15:0] 写入值就可以锁定 I/O (在写入时序过程中, LCKR[15:0] 写入值保持不变)。当在一个端口位上执行了锁定 (LOCK) 程序, 在下次 MCU 或者外设复位之前, 将不能再更改端口位的配置。GPIOx\_LCKR 的每个位冻结控制寄存器 (GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR、GPIOx\_PUPDR、GPIOx\_AFRL and GPIOx\_AFRH) 对应的位。

LOCK 时序只能用字 (32 位长) 访问 GPIOx\_LCKR 寄存器, 因为 GPIOx\_LCKR 位 16 设置的同时也会设置 [15:0] 位。

### 10.3.6. I/O 复用功能输入/输出模式配置

每个 I/O 有两个寄存器可以用来配置复用功能输入/输出模式。用户根据应用需求将复用功能复用到 IO 口上。

使用寄存器 GPIOx\_AFRL 和 GPIOx\_AFRH 可以在每一个 GPIO 口多路选择许多可能的外设功能, 因此应用让每个 I/O 选择其中一种功能。AF 选择信号对于复用功能输入和复用功能输出都是相同的, 对于给定 I/O 的复用功能输入/输出可以选择单独的通道。

### 10.3.7. 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线, 端口必须禁止配置成模拟模式或者晶振管脚, 并且输入触发需要使能。

### 10.3.8. I/O 输入配置

当 I/O 口配置为输入:

- 输出缓冲器不使能
- 施密特触发器输入使能
- 根据寄存器 GPIOx\_PUPDR 配置可使能/不使能上下拉电阻
- 出现在 I/O 脚上的数据在每个 AHB 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

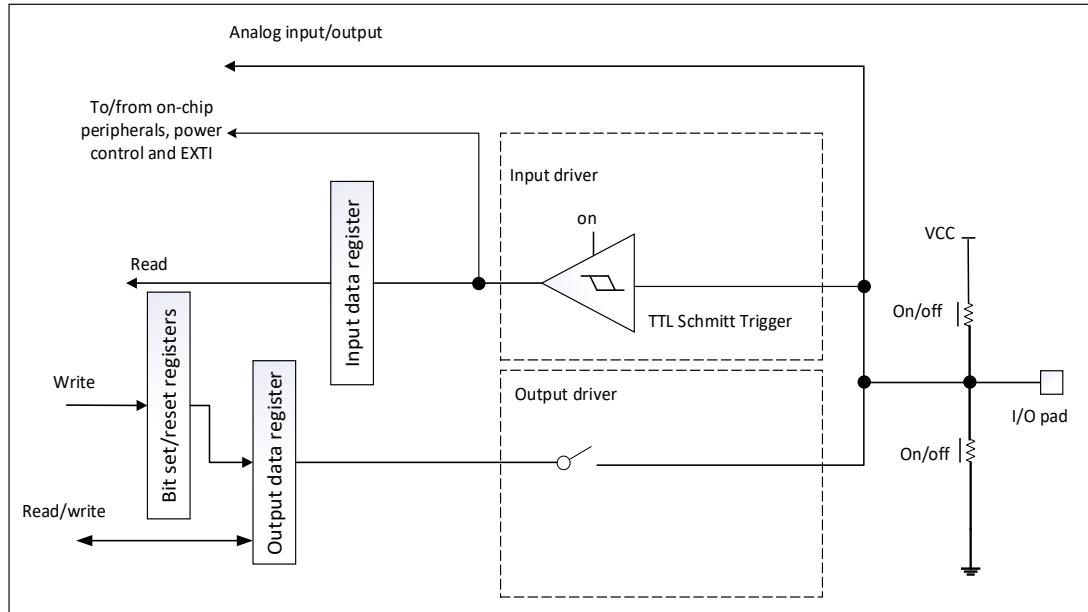


图 10-2 输入浮空/上拉/下拉配置

### 10.3.9. I/O 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
  - 开漏模式：输出寄存器上的 '0' 激活 N-MOS，而输出寄存器上的 '1' 将端口置于高阻状态（P-MOS 从不被激活）。
  - 推挽模式：输出寄存器上的 '0' 激活 N-MOS，而输出寄存器上的 '1' 将激活 P-MOS。
- 施密特触发输入被激活
- 根据寄存器 GPIOx\_PUPDR 配置可使能/不使能上下拉电阻
- 出现在 I/O 脚上的数据在每个 AHB 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态
- 对输出数据寄存器的读访问得到后一次写的值

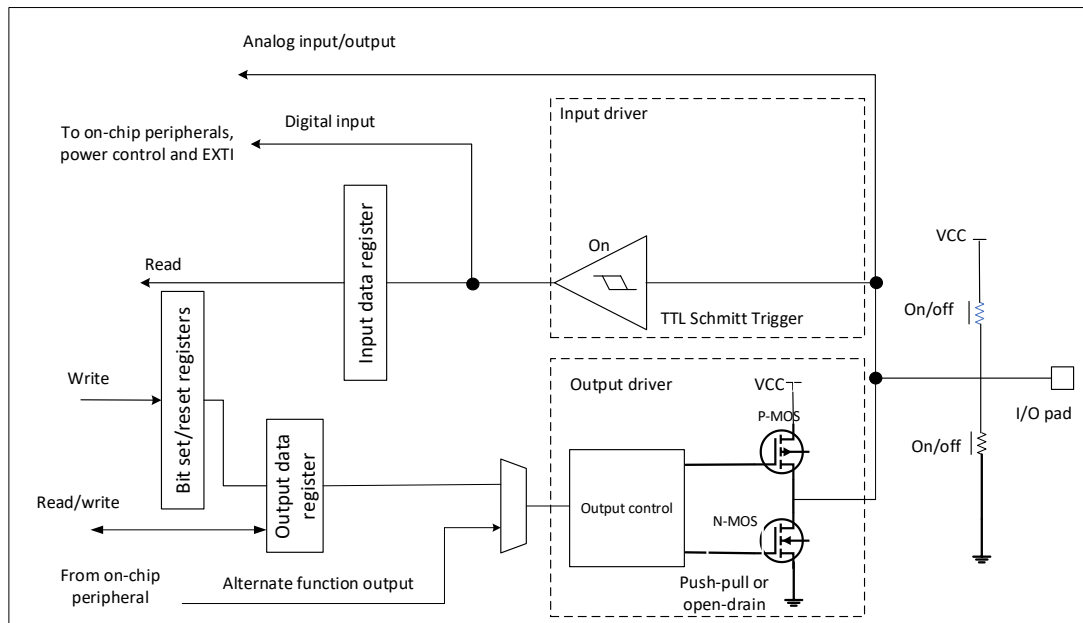


图 10-3 输出配置

### 10.3.10. 复用功能配置

当 I/O 端口被配置为复用功能时：

- 在开漏或推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器（复用功能输出）
- 施密特触发输入被激活
- 根据寄存器 GPIOx\_PUPDR 配置可使能/不使能上下拉电阻
- 在每个 AHB 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器
- 读输入数据寄存器时可得到 I/O 口状态

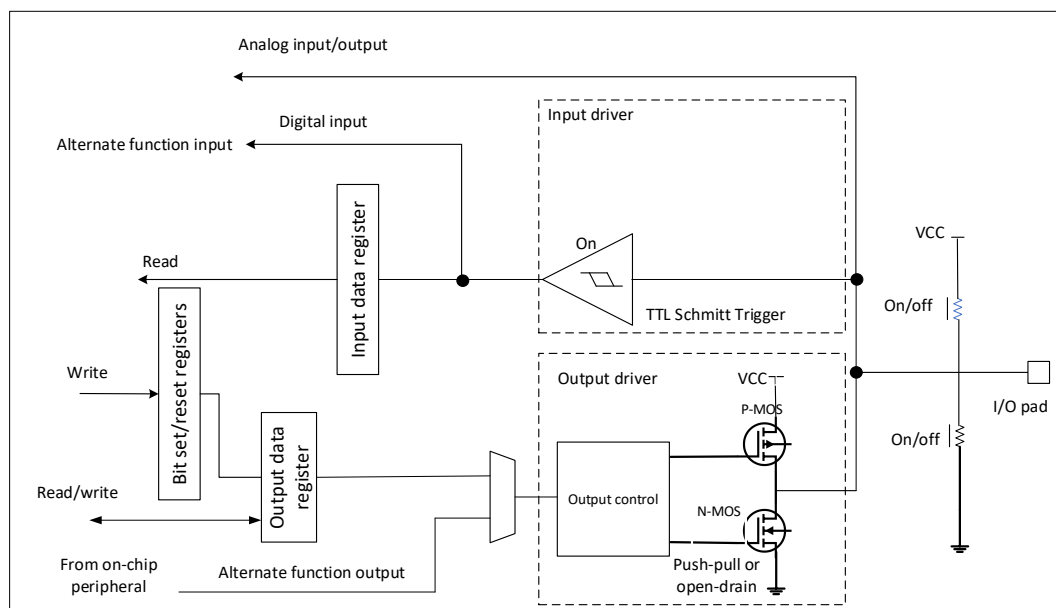


图 10-4 复用功能配置

### 10.3.11. 模拟配置

当 I/O 端口被配置为模拟配置时：

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

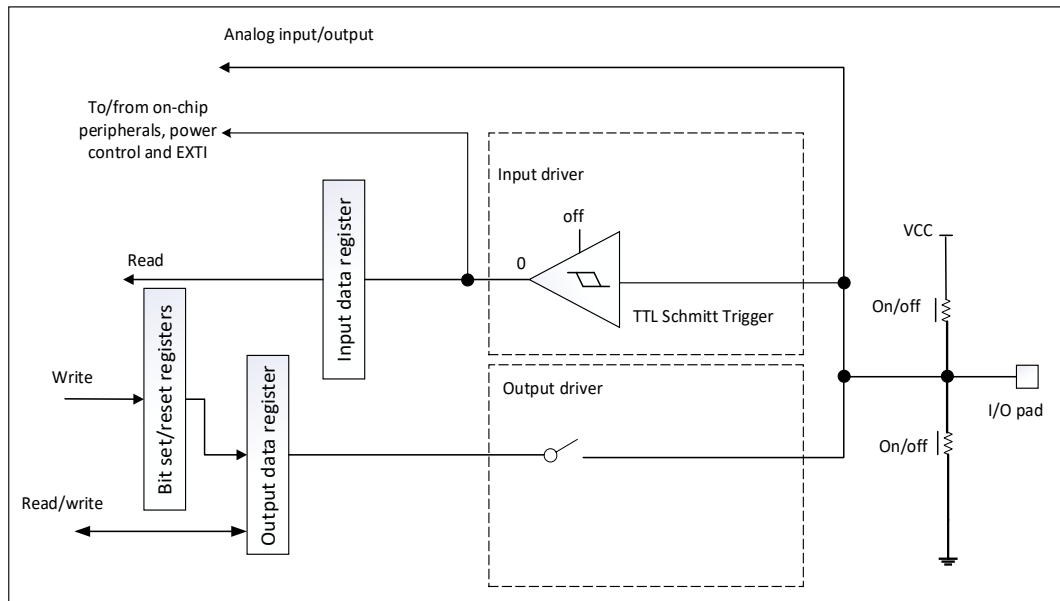


图 10-5 高阻抗模拟配置

### 10.3.12. 使用 HSE/LSE 管脚作为 GPIO

当 HSE 或 LSE 振荡器关闭（复位后的默认），相应的管脚可以当作正常的 GPIO 用。

当 HSE 或 LSE 振荡器打开（RCC\_CSR 寄存器中的 HSEON 或 LSEON 置位），需要软件配置对应的端口为模拟端口。

当振荡器配置为用户外部时钟模式，只有 OSC\_IN 或者 OSC32\_IN 保留给时钟输入，而 OSC\_OUT 或 OSC32\_OUT 脚仍然可以用作正常 GPIO。

## 10.4. GPIO 寄存器

所有 GPIO 相关寄存器都可进行字、半字和字节写操作。

### 10.4.1. GPIO 端口模式寄存器 (GPIOx\_MODER) (x=A, B, C, F)

Address offset: 0x00

Reset value:

- 0xEBFF FFFF (端口 A)
- 0xFFFF FFFF (端口 B)
- 0xFFFF FFFF (端口 C)
- 0x000C FFFF (端口 F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1: 0]		MODE14[1: 0]		MODE13[1: 0]		MODE12[1: 0]		MODE11[1: 0]		MODE10[1: 0]		MODE9[1: 0]		MODE8[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1: 0]		MODE6[1: 0]		MODE5[1: 0]		MODE4[1: 0]		MODE3[1: 0]		MODE2[1: 0]		MODE1[1: 0]		MODE0[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MODEy[1: 0]	RW	0xEBFFFFFF (PA) 0xFFFFFFFF (PB) 0xFFFFFFFF (PC) 0x000CFFFF (PF)	y = 15..0 软件通过这些位配置相应的 I/O 模式 00: 输入模式 01: 通用输出模式 10: 复用功能模式 11: 模拟模式 (reset state)

### 10.4.2. GPIO 端口输出类型寄存器 (GPIOx\_OTYPER) (x = A, B, C, F)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留

15: 0	OT[15: 0]	RW	0	软件配置 I/O 的输出类型 0: 推挽输出 (复位状态) 1: 开漏输出
-------	-----------	----	---	---

### 10.4.3. GPIO 端口输出速度寄存器 (GPIOx\_OSPEEDR) (x = A, B, C, F)

Address offset: 0x08

Reset value: 0x0C00 0000 (端口 A)

Reset value: 0x0000 0000 (其他端口)

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

Bit	Name	R/W	Reset Value	Function
31: 0	OSPEEDy[1: 0]	RW	0x0C000000 (PA) 0x0 (其他)	Y = 15..0 软件配置 IO 口的输出速度 00: 低速 01: 中速 10: 高速 11: 非常高

### 10.4.4. GPIO 端口上下拉寄存器 (GPIOx\_PUPDR) (x = A, B, C, F)

Address offset: 0x0C

Reset value:

- 0x2400 0000 (端口 A)
- 0x0000 0000 (端口 B、C)
- 0x0002 0000 (端口 F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1: 0]		PUPD14[1: 0]		PUPD13[1: 0]		PUPD12[1: 0]		PUPD11[1: 0]		PUPD10[1: 0]		PUPD9[1: 0]		PUPD8[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1: 0]		PUPD6[1: 0]		PUPD5[1: 0]		PUPD4[1: 0]		PUPD3[1: 0]		PUPD2[1: 0]		PUPD1[1: 0]		PUPD0[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PUPDy [1: 0]	RW	0x24000000 (PA) 0x0 (PB) 0x0 (PC) 0x20000 (PF)	Y = 15..0 软件配置 I/O 口上拉或者下拉 00: 无上下拉 01: 上拉 10: 下拉 11: 保留

### 10.4.5. GPIO 端口输入数据寄存器 (GPIOx\_IDR) (x = A, B, C, F)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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

### 10.4.6. GPIO 端口输出数据寄存器 (GPIOx\_ODR) (x = A, B, C, F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ODy	RW	0	y = 15..0 软件可读可写。

				说明: 对 GPIOx_BSRR 或 GPIOx_BRR (x=A,B,C,F) , 可以分别对各个 ODR 位进行独立的设置/清除。
--	--	--	--	---

### 10.4.7. GPIO 端口位设置/复位寄存器 (GPIOx\_BSRR) (x = A, B, C, F)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	BRy	W	0	y = 15..0 软件可写, 读出来返回值是0 0: 对对应的 ODRy 位不产生影响 1: 清除对应的 ODRy 位 注: 如果同时设置 BSy 和 BRy 的对应位, BSy 位起作用
15: 0	BSy	W	0	y = 15..0 软件可写, 读出来返回值是0 0: 对对应的 ODRy 位不产生影响 1: 设置对应的 ODRy 位

### 10.4.8. GPIO 端口配置锁定寄存器 (GPIOx\_LCKR) (x = A, B, C, F)

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

注: 特殊写时序用来写 GPIOx\_LCKR 寄存器。在锁定时序中仅仅只有字访问可以被执行。

每个锁定位冻结一种特定的配置寄存器 (控制和复用功能寄存器)

Address offset: 0x1C

Reset value: 0x0000 0000

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



LCK1	LCK1	LCK1	LCK1	LCK1	LCK1	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK
5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 17	保留	-	-	保留
16	LCKK	RW	0	<p>该位可随时读出，它只能通过锁键写入序列修改</p> <p>0: 端口配置锁键位未激活</p> <p>1: 端口配置锁键位被激活，下次系统复位前 GPIOx_LCKR 寄存器被锁定</p> <p>锁键的写入时序:</p> <p>WR LCKR[16] = '1' + LCKR[15: 0]</p> <p>WR LCKR[16] = '0' + LCKR[15: 0]</p> <p>WR LCKR[16] = '1' + LCKR[15: 0]</p> <p>RD LCKR</p> <p>RD LCKR[16] = '1' (此读操作作为可选操作，但它可确认锁定已激活)</p> <p>注: 在操作锁键的写入时序是，不能改变 LCK[15: 0]的值。锁键时序的任何错误都会终止锁键被激活。对端口的任何一位首次锁键时序之后，读 LCKK 位都是返回1,直接 MCU 复位或者外围复位。</p>
15: 0	LCKy	RW	0	<p>y = 15..0</p> <p>这些位可读可写但只能在 LCKK 位为0时写入。</p> <p>0: 不锁定端口的配置</p> <p>1: 锁定端口配置</p>

#### 10.4.9. GPIO 复用功能寄存器 (low) (GPIOx\_AFRL) (x = A, B, C, F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3: 0]				AFSEL6[3: 0]				AFSEL5[3: 0]				AFSEL4[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3: 0]				AFSEL2[3: 0]				AFSEL1[3: 0]				AFSEL0[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	AFSELy[3: 0] (y= 7 ~ 0)	RW	0	<p>软件可写这些位配置复用功能 I/O</p> <p>AFSELy 选择:</p>

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

#### 10.4.10. GPIO 复用功能寄存器 (high) (GPIOx\_AFRH) (x = A, B, C, F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3: 0]				AFSEL14[3: 0]				AFSEL13[3: 0]				AFSEL12[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3: 0]				AFSEL10[3: 0]				AFSEL9[3: 0]				AFSEL8[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	AFSELY[3: 0] (y= 8 ~ 15)	RW	0	软件可写这些位配置复用功能 I/O AFSELY 选择: 0000: AF0                      1000: AF8 0001: AF1                      1001: AF9 0010: AF2                      1010: AF10 0011: AF3                      1011: AF11 0100: AF4                      1100: AF12 0101: AF5                      1101: AF13 0110: AF6                      1110: AF14 0111: AF7                      1111: AF15

#### 10.4.11. GPIO 端口位复位寄存器 (GPIOx\_BRR) (x = A, B, C, F)

Address offset: 0x28

Reset value: 0x0000 0000

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

BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	BRy	W	0	y = 15..0 这些位软件可写，读出来返回值是0 0: 对对应的 ODRy 位不产生影响 1: 清除对应的 ODRy 位

## 11. 系统配置控制器 (SYSCFG)

芯片内有一套配置寄存器，系统配置控制器的主要目的是：

- I<sup>2</sup>C 类型 IO 滤波使能和关闭
- 根据不同 boot 模式，映射初始程序区
- DMA 外设通道选择控制
- TIMx 级联控制
- PVD Lock 的使能与关闭
- Cortex-M0+ LOCKUP 的使能与关闭
- 所有 GPIO 的噪声滤波器的使能与关闭

### 11.1. 系统配置寄存器

#### 11.1.1. SYSCFG 配置寄存器 1 (SYSCFG\_CFGR1)

该寄存器的 MEM\_MODE 位用作配置存储器地址 0x0000 0000 访问的种类。这两位寄存器用来选择软件的物理重映射，并旁路硬件 BOOT 的选择结果。在上电或者 PIN 复位后，这两位值由硬件采样到的实际 boot 模式配置决定。

**Address offset:** 0x00

**Reset value:** 0x0000 000x (x 是被实际 boot 模式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							GPIO_AHB_SEL	Res					ETR_SRC_TIM3[2: 0]			
-							RW	-					RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res	ETR_SRC_TIM2[2: 0]			Res	ETR_SRC_TIM1[2: 0]			TIM3_IC1_S	TIM2_IC4_S		TIM1_IC1_SR		MEM_MOD			
-	RW			-	RW			RC	RC		C		E			
-	RW			-	RW			RW	RW		RW		RW			

Bit	Name	R/W	Reset Value	Function
31: 25	保留	-	-	保留
24	GPIO_AHB_SEL	RW	0	CPU FASTIO 或 AHB 总线访问 GPIO 寄存器控制。 0: FASTIO 总线访问; 1: AHB 总线访问;
23: 19	保留	-	-	保留
18: 16	ETR_SRC_TIM3[2: 0]	RW	0	TIMER3 ETR 输入源选择。 3' b000: ETR 来源于 GPIO 3' b001: ETR 来源于 COMP1 3' b010: ETR 来源于 COMP2 3' b011: ETR 来源于 ADC

Bit	Name	R/W	Reset Value	Function
				3' b100: ETR 来源于 COMP3;
15	保留	-	-	保留
14: 12	ETR_SRC_TIM2[2: 0]	RW	0	TIMER2 ETR 输入源选择。 3' b000: ETR 来源于 GPIO 3' b001: ETR 来源于 COMP1 3' b010: ETR 来源于 COMP2 3' b011: ETR 来源于 ADC 3' b100: ETR 来源于 COMP3
11	保留	-	-	保留
10: 8	ETR_SRC_TIM1[2: 0]	RW	0	TIMER1 ETR 输入源选择。 3'b000: ETR 来源于 GPIO 3'b001: ETR 来源于 COMP1 3'b010: ETR 来源于 COMP2 3'b011: ETR 来源于 ADC 3'b100: ETR 来源于 COMP3
7: 6	TIM3_IC1_SRC	RW	0	TIM13 CH1输入来源 00: from TIM3_CH1 IO 01: from COMP1 10: from COMP2 11: from COMP3
5: 4	TIM2_IC4_SRC	RW	0	TIM2 CH4输入来源 00: from TIM2_CH4 IO; 01: from COMP1 10: from COMP2 11: from COMP3
3: 2	TIM1_IC1_SRC	RW	0	TIM1 CH1输入来源 00: from TIM1_CH1 IO; 01: from COMP1 10: from COMP2 11: from COMP3
1: 0	MEM_MODE	RW	x	存储区映射选择。 x0: Main Flash memory 映射在 0x0000 0000 01: System Flash memory 映射在 0x0000 0000 11: Embedded SRAM 映射在0x0000 0000

### 11.1.2. SYSCFG 配置寄存器 2 (SYSCFG\_CFGR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21
Res	Res	Res	Res	Res	Res	Res	Res	COMP3_O cref_CLR_ TIM3	COMP3_O cref_CLR_ TIM2	COMP3_O cref_CLR_ TIM1
-	-	-	-	-	-	-	-	-RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10
COMP2_O cref_CLR_ TIM3	COMP2_O cref_CLR_ TIM2	COMP2_O cref_CLR_ TIM1	COMP1_O cref_CLR_ TIM3	COMP1_O cref_CLR_ TIM2	COMP1_O cref_CLR_ TIM1	COMP3 BRK_TIM1 7	COMP2_ BRK_TIM1 7	COMP1_B RK_TIM17	COMP3 BRK_TIM1 6	COMP2_ BRK_TIM1 6
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
9	8	7	6	5	4	3	2	1	0	
COMP1_B RK_TIM16	COMP3 BRK_TIM1 5	COMP2_ BRK_TIM1 5	COMP1_B RK_TIM15	COMP3 BRK_TIM1	COMP2_ BRK_TIM1	COMP1_B RK_TIM1	PVD_LOC K	Res	LOCKUP_LOCK	
RW	RW	RW	RW	RW	RW	RW	RW	-	RW	

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23	COMP3_Ocref_CLR_TIM3	RW	0	1: COMP3输出作为 TIM3 ocref_clr 输入 0: COMP3输出不作为 TIM3 ocref_clr 输入
22	COMP3_Ocref_CLR_TIM2	RW	0	1: COMP3输出作为 TIM2 ocref_clr 输入 0: COMP3输出不作为 TIM2 ocref_clr 输入
21	COMP3_Ocref_CLR_TIM1	RW	0	1: COMP3输出作为 TIM1 ocref_clr 输入 0: COMP3输出不作为 TIM1 ocref_clr 输入
20	COMP2_Ocref_CLR_TIM3	RW	0	1: COMP2输出作为 TIM3 ocref_clr 输入 0: COMP2输出不作为 TIM3 ocref_clr 输入
19	COMP2_Ocref_CLR_TIM2	RW	0	1: COMP2输出作为 TIM2 ocref_clr 输入 0: COMP2输出不作为 TIM2 ocref_clr 输入
18	COMP2_Ocref_CLR_TIM1	RW	0	1: COMP2输出作为 TIM1 ocref_clr 输入 0: COMP2输出不作为 TIM1 ocref_clr 输入
17	COMP1_Ocref_CLR_TIM3	RW	0	1: COMP1输出作为 TIM3 ocref_clr 输入 0: COMP1输出不作为 TIM3 ocref_clr 输入
16	COMP1_Ocref_CLR_TIM2	RW	0	1: COMP1输出作为 TIM2 ocref_clr 输入 0: COMP1输出不作为 TIM2 ocref_clr 输入
15	COMP1_Ocref_CLR_TIM1	RW	0	1: COMP1输出作为 TIM1 ocref_clr 输入 0: COMP1输出不作为 TIM1 ocref_clr 输入
14	COMP3 BRK_TIM17	RW	0	COMP3作为 TIM17 break 输入使能 0: COMP3输出不与 TIM17 break input 相连 1: COMP3输出作为 TIM17 break input

Bit	Name	R/W	Reset Value	Function
13	COMP2_BRK_TIM17	RW	0	COMP2作为 TIM17 break 输入使能 0: COMP2输出不与 TIM17 break input 相连 1: COMP2输出作为 TIM17 break input
12	COMP1_BRK_TIM17	RW	0	COMP1作为 TIM17 break 输入使能 0: COMP1输出不与 TIM17 break input 相连 1: COMP1输出作为 TIM17 break input
11	COMP3 BRK_TIM16	RW	0	COMP3作为 TIM16 break 输入使能。 0: COMP3输出不与 TIM16 break input 相连; 1: COMP3输出作为 TIM16 break input;
10	COMP2_BRK_TIM16	RW	0	COMP2作为 TIM16 break 输入使能。 0: COMP2输出不与 TIM16 break input 相连; 1: COMP2输出作为 TIM16 break input;
9	COMP1_BRK_TIM16	RW	0	COMP1作为 TIM16 break 输入使能。 0: COMP1输出不与 TIM16 break input 相连; 1: COMP1输出作为 TIM16 break input;
8	COMP3 BRK_TIM15	RW	0	COMP3作为 TIM15 break 输入使能。 0: COMP3输出不与 TIM15 break input 相连; 1: COMP3输出作为 TIM15 break input;
7	COMP2_BRK_TIM15	RW	0	COMP2作为 TIM15 break 输入使能。 0: COMP2输出不与 TIM15 break input 相连; 1: COMP2输出作为 TIM15 break input;
6	COMP1_BRK_TIM15	RW	0	COMP1作为 TIM15 break 输入使能。 0: COMP1输出不与 TIM15 break input 相连; 1: COMP1输出作为 TIM15 break input;
5	COMP3 BRK_TIM1	RW	0	COMP3作为 TIM1 break 输入使能。 0: COMP3输出不与 TIM1 break input 相连; 1: COMP3输出作为 TIM1 break input;
4	COMP2_BRK_TIM1	RW	0	COMP2作为 TIM1 break 输入使能。 0: COMP2输出不与 TIM1 break input 相连; 1: COMP2输出作为 TIM1 break input;
3	COMP1_BRK_TIM1	RW	0	COMP1作为 TIM1 break 输入使能。 0: COMP1输出不与 TIM1 break input 相连; 1: COMP1输出作为 TIM1 break input;
2	PVD_LOCK	RW	0	PVD lock 使能位。 0: PVD 中断不与 TIM1/15/16/17 break input 相连, 由 PVDE 和 PLS 配置 PVD 功能; 1: PVD 中断与 TIM1/15/16/17 break input 相连, PVDE 和 PLS 只读;

Bit	Name	R/W	Reset Value	Function
1	保留	-	-	保留
0	LOCKUP_LOCK	RW	0	Cortex-M0+ LOCKUP 位 lock 使能位。 0: Cortex-M0+ LOCKUP 位不与 TIM1/15/16/17 break input 相连 1: Cortex-M0+ LOCKUP 位与 TIM1/15/16/17 break input 相连

### 11.1.3. SYSCFG 配置寄存器 3 (SYSCFG\_CFGR3)

Address offset: 0x08

Reset value: 0x3F3F 3F3F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		DMA4_MAP						Res		DMA3_MAP					
-		RW						-		RW					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		DMA2_MAP						Res		DMA1_MAP					
-		RW						-		RW					

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29: 24	DMA4_MAP	RW	0x3F	DMA1通道4映射。 见 DMA1_MAP 描述。
23: 22	保留	保留	-	保留
21: 16	DMA3_MAP	RW	0x3F	DMA1通道3映射。 见 DMA1_MAP 描述。
15: 14	保留	保留	-	保留
13: 8	DMA2_MAP	RW	0x3F	DMA1通道2映射。 见 DMA1_MAP 描述。
7: 6	保留	保留	-	保留
5: 0	DMA1_MAP	RW	0x3F	DMA1通道1映射。 000000: ADC1 000001: DAC1 000010: DAC2 000011: SPI1_RD 000100: SPI1_WR 000101: SPI2_RD 000110: SPI2_WR 000111: USART1_RD 001000: USART1_WR



Bit	Name	R/W	Reset Value	Function
				001001: USART2_RD
				001010: USART2_WR
				001011: USART3_RD
				001100: USART3_WR
				001101: USART4_RD
				001110: USART4_WR
				001111: I2C1_RD
				010000: I2C1_WR
				010001: I2C2_RD
				010010: I2C2_WR
				010011: TIM1_CH1
				010100: TIM1_CH2
				010101: TIM1_CH3
				010110: TIM1_CH4
				010111: TIM1_COM
				011000: TIM1_TRG
				011001: TIM1_UP
				011010: TIM2_CH1
				011011: TIM2_CH2
				011100: TIM2_CH3
				011101: TIM2_CH4
				011110: TIM2_UP
				011111: TIM2_TRG
				100000: TIM3_CH1
				100001: TIM3_CH2
				100010: TIM3_CH3
				100011: TIM3_CH4
				100100: TIM3_UP
				100101: TIM3_TRG
				100110: TIM6_UP
				100111: TIM7_UP
				101000: TIM15_CH1
				101001: TIM15_CH2
				101010: TIM15_UP
				101011: TIM15_TRG
				101100: TIM15_COM
				101101: TIM16_CH1
				101110: TIM16_UP

Bit	Name	R/W	Reset Value	Function
				101111: TIM17_CH1 110000: TIM17_UP 110001: USB 110010: LCD 其它: 保留

#### 11.1.4. SYSCFG 配置寄存器 4 (SYSCFG\_CFGR4)

Address offset: 0x0C

Reset value: 0x003F 3F3F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										DMA7_MAP					
-										RW					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		DMA6_MAP						Res		DMA5_MAP					
-		RW						-		RW					

Bit	Name	R/W	Reset Value	Function
31: 22	保留	-	-	保留
21: 16	DMA7_MAP	RW	0x3F	DMA1通道7映射。 见 DMA1_MAP 描述。
15: 14	保留	-	-	保留
13: 8	DMA6_MAP	RW	0x3F	DMA1通道6映射。 见 DMA1_MAP 描述。
7: 6	保留	-	-	保留
5: 0	DMA5_MAP	RW	0x3F	DMA1通道5映射。 见 DMA1_MAP 描述。

#### 11.1.5. GPIOA 滤波使能 (PA\_ENS)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PA_ENS[x]	RW	0x0000	PORTA IO 滤波使能。 0: 滤波禁止 1: 滤波使能

### 11.1.6. GPIOB 滤波使能 (PB\_ENS)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PB_ENS[x]	RW	0x0000	PORTB IO 滤波使能。 0: 滤波禁止 1: 滤波使能

### 11.1.7. GPIOC 滤波使能 (PC\_ENS)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PC_ENS[x]	RW	0x0000	PORTC IO 滤波使能。 0: 滤波禁止 1: 滤波使能

### 11.1.8. GPIOF 滤波使能 (PF\_ENS)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9: 0	PF_ENS[x]	RW	0x000	PORTF IO 滤波使能。 0: 滤波禁止 1: 滤波使能

### 11.1.9. I<sup>2</sup>C 配置寄存器 (SYSCFG\_EIIC)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25: 24	PF_EIIC[1: 0]	RW	0	PF 的 EIIC 控制信号。用作 I <sup>2</sup> C 类型 IO Bit0: 控制 PF3 Bit1: 控制 PF4
23: 16	保留	-	-	保留
15: 8	PB_EIIC[7: 0]	RW	0	PB 的 EIIC 控制信号。用作 I <sup>2</sup> C 类型 IO Bit0: 控制 PB6 Bit1: 控制 PB7 Bit2: 控制 PB8

				Bit3: 控制 PB9 Bit4: 控制 PB10 Bit5: 控制 PB11 Bit6: 控制 PB13 Bit7: 控制 PB14
7: 2	保留	-	-	保留
1: 0	PA_EIIC[1: 0]	RW	0	PA 的 EIIC 控制信号。用作 I <sup>2</sup> C 类型 IO Bit0: 控制 PA9 Bit1: 控制 PA10

## 12. 直接存储器存取 (DMA)

### 12.1. 简介

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

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

### 12.2. DMA 主要特性

- 单 AHB Master
- 支持外设到存储器, 存储器到外设, 存储器到存储器和外设到外设的数据传输
- 片上存储器设备, 如 Flash, SRAM, AHB 和 APB 外设, 作为源和目标
- 所有 DMA 通道均可独立配置:
  - 每个通道要么与来自外设的 DMA 请求信号相关联, 要么与存储器到存储器传输中的软件触发器相关联。这个配置是由软件完成的。
  - 请求之间的优先级可通过软件编程实现 (每个通道 4 级: 非常高、高、中、低), 在相等的情况下由硬件可编程 (例如对通道 1 的请求比对通道 2 的请求优先)。
  - 源和目标的传输大小是独立的 (字节, 半字, 字), 模拟打包和拆包。源地址和目标地址必须按数据大小对齐。
  - 可编程传输数据个数: 0 ~ 65535
- 每个通道生成一个中断请求。每个中断请求都是由三个 DMA 事件中的任何一个引起的: 传输完成、半传输或传输错误。

## 12.3. DMA 功能描述

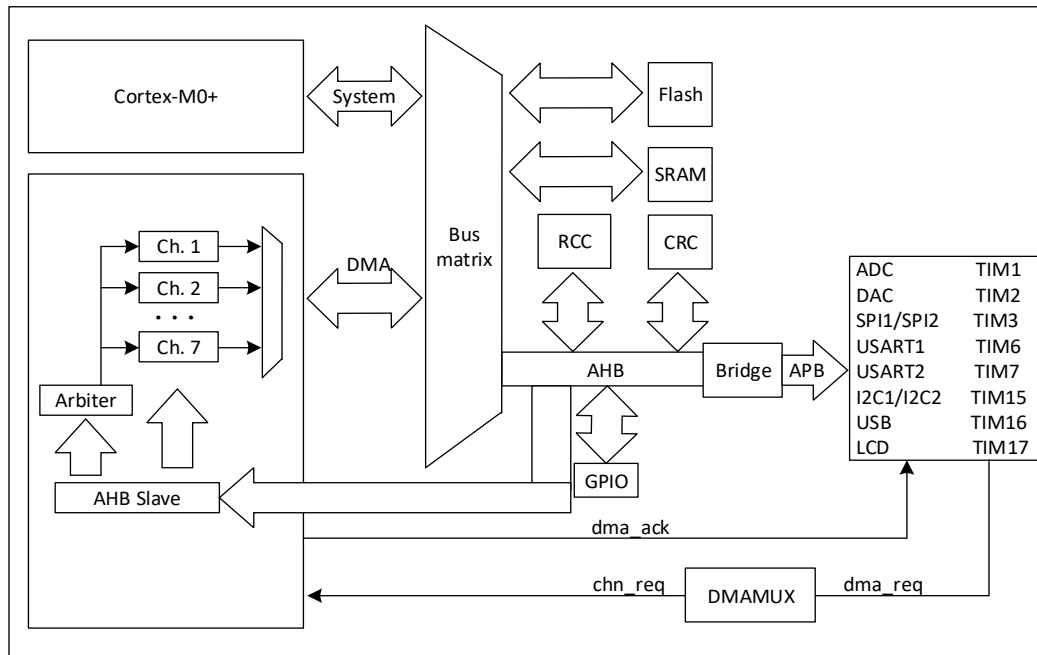


图 12-1 DMA 框图

### 12.3.1. DMA 处理

在完成一个事件后，外设向 DMA 控制器发送一个请求信号。DMA 控制器根据通道的优先级处理请求。当 DMA 控制器开始访问发出请求的外设时，DMA 控制器立即发送给它一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器撤销应答信号。如果有更多的请求时，外设可以启动下一个传送。

总之，每次 DMA 传送由三个操作组成：

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

### 12.3.2. 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。

优先级管理分 2 个阶段：

- 软件：每个通道的优先级可以在 DMA\_CCRx 寄存器中设置，有 4 个等级
  - 最高优先级
  - 高优先级
  - 中等优先级
  - 低优先级

- 硬件：如果 2 个请求有相同的软件优先级，则较低编号的通道比较高编号的通道有较高的优先权。比如，通道 2 优先于通道 4。

### 12.3.3. DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 的传输数据量是可编程的，最大为 65535 字节，该寄存器值在每次数据传输后递减。

#### 传输数据量可编程

外设和存储器的传输量可以通过 DMA\_CCRx 寄存器中的 PSIZE 和 MSIZE 位编程。

#### 地址指针增量

通过设置 DMA\_CCRx 寄存器中的 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。

当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决于所选的数据宽度为 1、2 或 4。第一个传输的地址是存放在 DMA\_CPARx/DMA\_CMARx 寄存器中地址。在传输过程中，这些寄存器保持它们初始的数值，软件不能改变和读出当前正在传输的地址（它在内部的当前外设/存储器地址寄存器中）。

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

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

#### 通道配置过程

按如下步骤配置外设请求时 DMA 通道：

- 在 DMA\_CPARx 寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
- 在 DMA\_CMARx 寄存器中设置数据寄存器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
- 在 DMA\_CNDTRx 寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减。
- 配置 DMA\_CCRx 寄存器如下参数：
  - 通道的优先级。
  - 数据传输方向
  - 循环模式
  - 外设和存储器增量模式
  - 外设和存储器数据大小
  - 中断使能
- 设置 DMA\_CCRx 寄存器的 EN 位，启动该通道。

一旦启动了 DMA 通道，即可响应连到该通道上的外设的 DMA 请求。



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

### 通道状态及禁止通道

处于激活状态的通道  $x$  是一个启用的通道 (读取  $DMA\_CCRx.EN=1$ )。一个活动通道  $x$  是一个必须已经被软件启用的通道 ( $DMA\_CCRx.EN = 1$ )，然后没有发生传输错误 ( $DMA\_ISR.TEIFx = 0$ )。如果出现传输错误，通道会被硬件自动禁用 ( $DMA\_CCRx.EN = 0$ )。

以下 3 种情况可能会发生：

#### 1. 暂停和恢复通道

这对应以下两个动作：

- 1) 激活的通道被软件禁用 (写入  $DMA\_CCRx.EN = 0$ )。
- 2) 软件重新启用通道 ( $DMA\_CCRx.EN=1$ )，但没有重新配置其他通道寄存器 (如  $DMA\_CNDTRx$ 、 $DMA\_CPARx$  和  $DMA\_CMARx$ )；或者软件禁用时未完成的传输挂起了总线。

DMA 硬件不支持这种情况，因此不能保证正确地执行剩余的数据传输。

#### 2. 停止和中止一个通道

如果应用程序不再需要该通道，则可以通过软件禁用该活动通道。通道被停止并中止，但是  $DMA\_CNDTRx$  寄存器内容可能不能正确地反映剩余的数据传输。

#### 3. 中止并重新启动通道

这对应于软件顺序：禁用活动通道，然后重新配置通道并再次启用它。

满足以下条件时，硬件支持：

1. 应用程序保证，当软件禁用通道时，DMA 没有正在进行 (还未完成的) 传输。例如，应用程序可以首先在 DMA 模式下禁用外设，以确保该外设没有挂起的硬件 DMA 请求。
2. 软件必须对同一个  $DMA\_CCRx$  寄存器进行独立的写访问：首先关闭通道，其次，如果需要更改配置，则重新配置通道，以便进行下一次块传输，包括  $DMA\_CCRx$ 。最后再次启用通道。

当发生通道传输错误时，硬件清除  $DMA\_CCRx$  寄存器的 EN 位。在  $DMA\_ISR$  寄存器的  $TEIFx$  位被设置之前，这个 EN 位不能被软件再次设置以重新激活通道  $x$ 。

### DMA 循环模式

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

注：在存储器到存储器模式中不能使用循环模式。在循环模式 (CIRC = 1) 使能通道前，软件必须清除  $DMA\_CCRx$  寄存器的 MEM2MEM 位。当循环模式被激活时，要传输的数据量将在通道配置阶段使用编程的初始值自动重新加载，DMA 请求将继续得到响应。

为了停止循环传输，软件需要在禁用 DMA 通道之前停止外设生成 DMA 请求 (例如退出 ADC 扫描模式)。

在开始/启用一个传输之前，以及在停止一个循环传输之后，软件必须明确地对 DMA\_CNDTRx 值进行编程。

### 存储器到存储器模式

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

当设置了 DMA\_CCRx 寄存器中的 MEM2MEM 位之后，在软件设置了 DMA\_CCRx 寄存器中的 EN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA\_CNDTRx 寄存器变为 0 时，DMA 传输结束。

存储器到存储器模式不能与循环模式同时使用。

### 外设到外设模式

任何 DMA 通道都可以在外设到外设模式下运行：

- 当来自外设的硬件请求被选择触发 DMA 通道时

这个外设是 DMA 启动器，在这个外设和属于另一个存储器映射外设（这个外设没有配置为 DMA 模式）的寄存器之间进行数据传输。

- 当没有选择外设请求并连接到 DMA 通道时

该软件通过设置 DMA\_CCRx 寄存器的 MEM2MEM 位来配置寄存器到寄存器的传输。

### 配置传输方向，指定源/目标

DMA\_CCRx 寄存器的 DIR 位的值设置了传输的方向，因此，它标识了源和目标，而不管源/目标类型（外设或存储器）：

- DIR = 1 通常定义存储器到外设的传输。更一般地，如果 DIR = 1：
  - 源属性由 DMA\_MARx 寄存器、MSIZE[1: 0]字段和 DMA\_CCRx 寄存器的 MINC 位定义。
 

不管它们通常的命名是什么，这些“存储器”寄存器、字段和位被用来在外设到外设模式下定义源外设。
  - 目标属性由 DMA\_PARx 寄存器、DMA\_CCRx 寄存器的 PSIZE[1: 0]字段和 PINC 位定义。
 

不管它们通常的命名，这些“外设”寄存器、字段和位被用来在存储器到存储器模式下定义目标存储器。
- DIR = 0 通常定义一个外设到存储器的传输。更一般地，如果 DIR = 0：
  - 源属性由 DMA\_PARx 寄存器、DMA\_CCRx 寄存器的 PSIZE[1: 0]字段和 PINC 位定义。不管它们通常的命名是什么，这些“外设”寄存器、字段和位被用来在存储器到存储器模式下定义源存储器。
  - 目标属性由 DMA\_MARx 寄存器、DMA\_CCRx 寄存器的 MSIZE[1: 0]字段和 MINC 位定义。
 

不管它们通常的命名，这些“存储器”寄存器、字段和位被用来以外设到外设模式定义目标地外设。

## 12.3.4. 数据传输宽度/对齐方式/大小端

当存储器数据宽度 MSIZE 和外设数据宽度 PSIZE 不同时，DMA 按照下表进行数据对齐：

表 12-1 数据宽度和大小端 (PINC=MINC=1)

源宽度	目标宽度	传输数目	源：地址/数据	传输操作	目标：地址/数据
8	8	4	0x0/B0	1: 在0x0读 B0[7: 0],在0x0写 B0[7: 0]	0x0/B0

			0x1/B1	2: 在0x1读 B1[7: 0],在0x1写 B1[7: 0]	0x1/B1
			0x2/B2	3: 在0x2读 B2[7: 0],在0x2写 B2[7: 0]	0x2/B2
			0x3/B3	4: 在0x3读 B3[7: 0],在0x3写 B3[7: 0]	0x3/B3
8	16	4	0x0/B0	1: 在0x0读 B0[7: 0],在0x0写00B0[7: 0]	0x0/00B0
			0x1/B1	2: 在0x1读 B1[7: 0],在0x2写00B1[7: 0]	0x2/00B1
			0x2/B2	3: 在0x2读 B2[7: 0],在0x4写00B2[7: 0]	0x4/00B2
			0x3/B3	4: 在0x3读 B3[7: 0],在0x6写00B3[7: 0]	0x6/00B3
8	32	4	0x0/B0	1: 在0x0读 B0[7: 0],在0x0写000000B0[31: 0]	0x0/000000B0
			0x1/B1	2: 在0x1读 B1[7: 0],在0x4写000000B1[31: 0]	0x4/000000B1
			0x2/B2	3: 在0x2读 B2[7: 0],在0x8写000000B2[31: 0]	0x8/000000B2
			0x3/B3	4: 在0x3读 B3[7: 0],在0xC 写000000B3[31: 0]	0xC/000000B3
16	8	4	0x0/B1B0	1: 在0x0读 B1B0[15: 0],在0x0写 B0[7: 0]	0x0/B0
			0x2/B3B2	2: 在0x2读 B3B2[15: 0],在0x1写 B2[7: 0]	0x1/B2
			0x4/B5B4	3: 在0x4读 B5B4[15: 0],在0x2写 B4[7: 0]	0x2/B4
			0x6/B7B6	4: 在0x6读 B7B6[15: 0],在0x3写 B6[7: 0]	0x3/B6
16	16	4	0x0/B1B0	1: 在0x0读 B1B0[15: 0],在0x0写 B1B0[15: 0]	0x0/B1B0
			0x2/B3B2	2: 在0x2读 B3B2[15: 0],在0x2写 B3B2[15: 0]	0x2/B3B2
			0x4/B5B4	3: 在0x4读 B5B4[15: 0],在0x4写 B5B4[15: 0]	0x4/B5B4
			0x6/B7B6	4: 在0x6读 B7B6[15: 0],在0x6写 B7B6[15: 0]	0x6/B7B6
16	32	4	0x0/B1B0	1: 在0x0读 B1B0[7: 0],在0x0写0000B1B0[31: 0]	0x0/0000B1B0
			0x2/B3B2	2: 在0x2读 B3B2[7: 0],在0x4写0000B3B2[31: 0]	0x4/0000B3B2
			0x4/B5B4	3: 在0x4读 B5B4[7: 0],在0x8写0000B5B4[31: 0]	0x8/0000B5B4
			0x6/B7B6	4: 在0x6读 B7B6[7: 0],在0xC 写0000B7B6[31: 0]	0xC/0000B7B6
32	8	4	0x0/B3B2B1B0	1: 在0x0读 B3B2B1B0 [31: 0],在0x0写 B0[7: 0]	0x0/B0
			0x4/B7B6B5B4	2: 在0x4读 B7B6B5B4 [31: 0],在0x1写 B4[7: 0]	0x1/B4
			0x8/BBBAB9B8	3: 在0x8读 BBBAB9B8 [31: 0],在0x2写 B8[7: 0]	0x2/B8
			0xC/BFBEBDBC	4: 在0xc 读 BFBEBDBC [31: 0],在0x3写 BC[7: 0]	0x3/BC
32	16	4	0x0/B3B2B1B0	1: 在0x0读 B3B2B1B0 [31: 0],在0x0写 B1B0[7: 0]	0x0/B1B0
			0x4/B7B6B5B4	2: 在0x4读 B7B6B5B4 [31: 0],在0x2写 B5B4[7: 0]	0x2/B5B4
			0x8/BBBAB9B8	3: 在0x8读 BBBAB9B8 [31: 0],在0x4写 B9B8[7: 0]	0x4/B9B8
			0xC/BFBEBDBC	4: 在0xc 读 BFBEBDBC [31: 0],在0x6写 BDBC[7: 0]	0x6/BDBC
32	32	4	0x0/B3B2B1B0	1: 在0x0读 B3B2B1B0 [31: 0],在0x0写 B3B2B1B0[7: 0]	0x0/B3B2B1B0
			0x4/B7B6B5B4	2: 在0x4读 B7B6B5B4 [31: 0],在0x2写 B7B6B5B4[7: 0]	0x4/B7B6B5B4
			0x8/BBBAB9B8	3: 在0x8读 BBBAB9B8 [31: 0],在0x4写 BBBAB9B8[7: 0]	0x8/BBBAB9B8

			0xC/BFBEBDBC	4: 在0xc 读 BFBEBDBC [31: 0],在0x6写 BFBEB- DBC[7: 0]	0xC/BFBEBDBC
--	--	--	--------------	--	--------------

### 操作不支持字节或半字写的 AHB 设备

如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时（即 HSIZE 不适用于该模块），不会发生错误，DMA 将按照下面两个例子写入 32 位 HWDATA 数据：

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

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

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

### 12.3.5. 错误管理

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

DMA\_CCRx 寄存器的 EN 位不能被软件再次设置（通道 x 重新激活），直到 DMA\_ISR 寄存器的 TEIFx 位被清除（通过设置 DMA\_IFCR 寄存器的 CTEIFx 位）。

### 12.3.6. DMA 中断

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

表 12-2 DMA 中断请求

中断事件	事件标志位	使能控制位
传输过半	HTIFx	HTIEx
传输完成	TCIFx	TCIEx
传输错误	TEIFx	TEIEx
传输过半/传输完成/传输错误	GIFx	-

注：

1. 当 DMA\_CNDTRx 寄存器为 1 时，不会置 HTIFx 位，传输完成会置 TCIFx 位。
2. 当传输长度 NDT 为奇数时（大于 1），HTIF 和 TCIF 标志都会产生。内部信号 TCIF 会在 NDT=1 时产生；HTIF 会在 (NDT - (NDT/2 (取整) - 1)) 时产生。如 NDT=5 时，TCIF 在 NDT 减至 1 时产生；HTIF 在 NDT 减至 4 时产生。

3. 当传输长度 NDT 为偶数时 (大于1) , HTIF 和 TCIF 标志都会产生。内部信号 TCIF 会在 NDT=1时产生; HTIF 会在 (NDT- (NDT/2 (取整) -1) ) 时产生。如 NDT=10时, TCIF 在 NDT 减至1时产生; HTIF 在 NDT 减至6时产生。

### 12.3.7. DMA 外设请求映射

DMA 外设请求映射到 DMA 的各个通道, 由 SYSCFG 两个寄存器 (SYSCFG\_CFGR3 和 SYSCFG\_CFGR4) 的 DMAx\_MAP 寄存器位控制, 每个外设请求通过配置可以映射到 7 个通道中的任意一个。

表 12-3 每个通道的 DMA 请求

请求 MUX 输入序号	源	请求 MUX 输入序号	源	请求 MUX 输入序号	源
0	ADC1	17	I2C2_RD	34	TIM3_CH3
1	DAC1	18	I2C2_WR	35	TIM3_CH4
2	DAC2	19	TIM1_CH1	36	TIM3_UP
3	SPI1_RD	20	TIM1_CH2	37	TIM3_TRG
4	SPI1_WR	21	TIM1_CH3	38	TIM6_UP
5	SPI2_RD	22	TIM1_CH4	39	TIM7_UP
6	SPI2_WR	23	TIM1_COM	40	TIM15_CH1
7	USART1_RD	24	TIM1_TRG	41	TIM15_CH2
8	USART1_WR	25	TIM1_UP	42	TIM15_UP
9	USART2_RD	26	TIM2_CH1	43	TIM15_TRG
10	USART2_WR	27	TIM2_CH2	44	TIM15_COM
11	USART3_RD	28	TIM2_CH3	45	TIM16_CH1
12	USART3_WR	29	TIM2_CH4	46	TIM16_UP
13	USART4_RD	30	TIM2_UP	47	TIM17_CH1
14	USART4_WR	31	TIM2_TRG	48	TIM17_UP
15	I2C1_RD	32	TIM3_CH1	49	USB
16	I2C1_WR	33	TIM3_CH2	50	LCD

## 12.4. DMA 寄存器

### 12.4.1. DMA 中断状态寄存器 (DMA\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 28	保留	-	-	保留
27	TEIF7	R	0	通道7传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道7传输错误 (TE)
26	HTIF7	R	0	通道7半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道7发生半传输事件 (HT)
25	TCIF7	R	0	通道7传输完成标志。 0: 无传输完成 (TC) 1: 通道7传输完成 (TC)
24	GIF7	R	0	通道7全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道7发生 TE/HT/TC 事件
23	TEIF6	R	0	通道6传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道6传输错误 (TE)
22	HTIF6	R	0	通道6半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道6发生半传输事件 (HT)
21	TCIF6	R	0	通道6传输完成标志。 0: 无传输完成 (TC) 1: 通道6传输完成 (TC)
20	GIF6	R	0	通道6全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道6发生 TE/HT/TC 事件
19	TEIF5	R	0	通道5传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE)

Bit	Name	R/W	Reset Value	Function
				1: 通道5传输错误 (TE)
18	HTIF5	R	0	通道5半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道5发生半传输事件 (HT)
17	TCIF5	R	0	通道5传输完成标志。 0: 无传输完成 (TC) 1: 通道5传输完成 (TC)
16	GIF5	R	0	通道5全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道5发生 TE/HT/TC 事件
15	TEIF4	R	0	通道4传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道4传输错误 (TE)
14	HTIF4	R	0	通道4半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道4发生半传输事件 (HT)
13	TCIF4	R	0	通道4传输完成标志。 0: 无传输完成 (TC) 1: 通道4传输完成 (TC)
12	GIF4	R	0	通道4全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道4发生 TE/HT/TC 事件
11	TEIF3	R	0	通道3传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道3传输错误 (TE)
10	HTIF3	R	0	通道3半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道3发生半传输事件 (HT)
9	TCIF3	R	0	通道3传输完成标志。 0: 无传输完成 (TC) 1: 通道3传输完成 (TC)

Bit	Name	R/W	Reset Value	Function
8	GIF3	R	0	通道3全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道3发生 TE/HT/TC 事件
7	TEIF2	R	0	通道2传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道2传输错误 (TE)
6	HTIF2	R	0	通道2半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道2发生半传输事件 (HT)
5	TCIF2	R	0	通道2传输完成标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输完成 (TC) 1: 通道2传输完成 (TC)
4	GIF2	R	0	通道2全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道2发生 TE/HT/TC 事件
3	TEIF1	R	0	通道1传输错误标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输错误 (TE) 1: 通道1传输错误 (TE)
2	HTIF1	R	0	通道1半传输标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无半传输事件 (HT) 1: 通道1发生半传输事件 (HT)
1	TCIF1	R	0	通道1传输完成标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无传输完成 (TC) 1: 通道1传输完成 (TC)
0	GIF1	R	0	通道1全局中断标志。 硬件置位, 软件写 DMA_IFCR=1清零。 0: 无 TE/HT/TC 事件 1: 通道1发生 TE/HT/TC 事件



## 12.4.2. DMA 中断标志位清除寄存器 (DMA\_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 28	保留	-	-	保留
27	CTEIF7	W	0	通道7传输错误标志清零。 0: 无影响 1: 清零 TEIF7
26	CHTIF7	W	0	通道7半传输标志清零。 0: 无影响 1: 清零 HTIF7
25	CTCIF7	W	0	通道7传输完成标志清零。 0: 无影响 1: 清零 TCIF7
24	CGIF7	W	0	通道7全局中断标志清零。 0: 无影响 1: 清零通道7的 GIF/TEIF/HTIF/TCIF
23	CTEIF6	W	0	通道6传输错误标志清零。 0: 无影响 1: 清零 TEIF6
22	CHTIF6	W	0	通道6半传输标志清零。 0: 无影响 1: 清零 HTIF6
21	CTCIF6	W	0	通道6传输完成标志清零。 0: 无影响 1: 清零 TCIF6
20	CGIF6	W	0	通道6全局中断标志清零。 0: 无影响 1: 清零通道6的 GIF/TEIF/HTIF/TCIF
19	CTEIF5	W	0	通道5传输错误标志清零。 0: 无影响

Bit	Name	R/W	Reset Value	Function
				1: 清零 TEIF5
18	CHTIF5	W	0	通道5半传输标志清零。 0: 无影响 1: 清零 HTIF5
17	CTCIF5	W	0	通道5传输完成标志清零。 0: 无影响 1: 清零 TCIF5
16	CGIF5	W	0	通道5全局中断标志清零。 0: 无影响 1: 清零通道5的 GIF/TEIF/HTIF/TCIF
15	CTEIF4	W	0	通道4传输错误标志清零。 0: 无影响 1: 清零 TEIF4
14	CHTIF4	W	0	通道4半传输标志清零。 0: 无影响 1: 清零 HTIF4
13	CTCIF4	W	0	通道4传输完成标志清零。 0: 无影响; 1: 清零 TCIF4;
12	CGIF4	W	0	通道4全局中断标志清零。 0: 无影响 1: 清零通道4的 GIF/TEIF/HTIF/TCIF
11	CTEIF3	W	0	通道3传输错误标志清零。 0: 无影响 1: 清零 TEIF3
10	CHTIF3	W	0	通道3半传输标志清零。 0: 无影响 1: 清零 HTIF3
9	CTCIF3	W	0	通道3传输完成标志清零。 0: 无影响 1: 清零 TCIF3
8	CGIF3	W	0	通道3全局中断标志清零。 0: 无影响 1: 清零通道3的 GIF/TEIF/HTIF/TCIF
7	CTEIF2	W	0	通道2传输错误标志清零。 0: 无影响 1: 清零 TEIF2
6	CHTIF2	W	0	通道2半传输标志清零。

Bit	Name	R/W	Reset Value	Function
				0: 无影响 1: 清零 HTIF2
5	CTCIF2	W	0	通道2传输完成标志清零。 0: 无影响 1: 清零 TCIF2
4	CGIF2	W	0	通道2全局中断标志清零。 0: 无影响 1: 清零通道2的 GIF/TEIF/HTIF/TCIF
3	CTEIF1	W	0	通道1传输错误标志清零。 0: 无影响 1: 清零 TEIF1
2	CHTIF1	W	0	通道1半传输标志清零。 0: 无影响 1: 清零 HTIF1
1	CTCIF1	W	0	通道1传输完成标志清零。 0: 无影响 1: 清零 TCIF1
0	CGIF1	W	0	通道1全局中断标志清零。 0: 无影响 1: 清零通道1的 GIF/TEIF/HTIF/TCIF

### 12.4.3. DMA 通道 1 配置寄存器 (DMA\_CCR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道1存储器到存储器模式。 0: 禁止 1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道1优先级配置。 00: 低

Bit	Name	R/W	Reset Value	Function
				01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道1存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道1外设数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道1存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道1外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道1循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道1数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道1传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道1半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道1传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道1使能。 0: 禁止 1: 通道1使能

#### 12.4.4. DMA 通道 1 数据传输数量寄存器 (DMA\_CNDTR1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	<p>通道1数据传输数量。</p> <p>数据传输数量为0~65535。该寄存器只在通道不工作 (DMA_CCR1.EN=0) 时写入。通道使能后该寄存器为只读，表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。</p> <p>数据传输结束后，寄存器的内容或者变为0，或者当该通道配置为循环模式时，寄存器的内容将被自动重新加载为之前配置时的数值。</p> <p>当该寄存器值为0时，即使 DMA 通道开始，都不会传输数据。</p>

#### 12.4.5. DMA 通道 1 外设地址寄存器 (DMA\_CPAR1)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	<p>通道1外设地址。</p> <p>通道1外设数据寄存器的基址，作为数据传输的源或目标。</p>

				当 PSIZE=2'b01, 不使用 PA[0]位。操作自动与半字地址对齐。 当 PSIZE=2'b10, 不使用 PA[1: 0]位。操作自动与字地址对齐。
--	--	--	--	--

### 12.4.6. DMA 通道 1 存储地址寄存器 (DMA\_CMAR1)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道1存储器地址。 通道1存储器地址, 作为数据传输的源或目标。 当 MSIZE=2'b01, 不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10, 不使用 MA[1: 0]位。操作自动与字地址对齐。

### 12.4.7. DMA 通道 2 配置寄存器 (DMA\_CCR2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道2存储器到存储器模式。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道2优先级配置。 00: 低 01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道2存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道2外设数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道2存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道2外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道2循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道2数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道2传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道2半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道2传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道2使能。

Bit	Name	R/W	Reset Value	Function
				0: 禁止 1: 通道1使能

#### 12.4.8. DMA 通道 2 数据传输数量寄存器 (DMA\_CNDTR2)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	<p>通道2数据传输数量。 数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR2.EN=0) 时写入。通道使能后该寄存器为只读, 表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。 数据传输结束后, 寄存器的内容或者变为0, 或者当该通道配置为循环模式时, 寄存器的内容将被自动重新加载为之前配置时的数值。 当该寄存器值为0时, 即使 DMA 通道开始, 都不会传输数据。</p>

#### 12.4.9. DMA 通道 2 外设地址寄存器 (DMA\_CPAR2)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	通道2外设地址。 通道2外设数据寄存器的基址，作为数据传输的源或目标。 当 PSIZE=2'b01，不使用 PA[0]位。操作自动与半字地址对齐。 当 PSIZE=2'b10，不使用 PA[1: 0]位。操作自动与字地址对齐。

### 12.4.10. DMA 通道 2 存储地址寄存器 (DMA\_CMAR2)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道2存储器地址。 通道2存储器地址，作为数据传输的源或目标。 当 MSIZE=2'b01，不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10，不使用 MA[1: 0]位。操作自动与字地址对齐。

### 12.4.11. DMA 通道 3 配置寄存器 (DMA\_CCR3)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道3存储器到存储器模式。 0: 禁止 1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道3优先级配置。 00: 低 01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道3存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道3外设数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道3存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道3外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道3循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道3数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道3传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道3半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道3传输完成中断 (TC) 使能。

Bit	Name	R/W	Reset Value	Function
				0: 禁止 1: TC 中断使能
0	EN	RW	0	通道3使能。 0: 禁止 1: 通道1使能

### 12.4.12. DMA 通道 3 数据传输数量寄存器 (DMA\_CNDTR3)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	通道3数据传输数量。 数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR3.EN=0) 时写入。通道使能后该寄存器为只读, 表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。 数据传输结束后, 寄存器的内容或者变为0, 或者当该通道配置为循环模式时, 寄存器的内容将被自动重新加载为之前配置时的数值。 当该寄存器值为0时, 即使 DMA 通道开始, 都不会传输数据。

### 12.4.13. DMA 通道 3 外设地址寄存器 (DMA\_CPAR3)

Address offset: 0x38

Reset value: 0x0000 0000

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

PA[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	通道3外设地址。 通道3外设数据寄存器的基址，作为数据传输的源或目标。 当 PSIZE=2'b01，不使用 PA[0]位。操作自动与半字地址对齐。 当 PSIZE=2'b10，不使用 PA[1: 0]位。操作自动与字地址对齐。

#### 12.4.14. DMA 通道 3 存储器地址寄存器 (DMA\_CMAR3)

Address offset: 0x3C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道3存储器地址。 通道3存储器地址，作为数据传输的源或目标。 当 MSIZE=2'b01，不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10，不使用 MA[1: 0]位。操作自动与字地址对齐。

#### 12.4.15. DMA 通道 4 配置寄存器 (DMA\_CCR4)

Address offset: 0x44

Reset value: 0x0000 0000

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

Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道存储器到存储器模式。 0: 禁止 1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道优先级配置。 00: 低 01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道外设数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道循环模式。 0: 禁止 1: 循环模式使能;
4	DIR	RW	0	通道数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能

Bit	Name	R/W	Reset Value	Function
2	HTIE	RW	0	通道半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道使能 0: 禁止 1: 通道使能

#### 12.4.16. DMA 通道 4 数据传输个数寄存器 (DMA\_CNDTR4)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	通道数据传输数量。 数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR4.EN=0) 时写入。通道使能后该寄存器为只读, 表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。 数据传输结束后, 寄存器的内容或者变为0, 或者当该通道配置为循环模式时, 寄存器的内容将被自动重新加载为之前配置时的数值。 当该寄存器值为0时, 即使 DMA 通道开始, 都不会传输数据。

#### 12.4.17. DMA 通道 4 外设地址寄存器 (DMA\_CPAR4)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	通道外设地址。 通道外设数据寄存器的基址，作为数据传输的源或目标。 当 PSIZE=2'b01，不使用 PA[0]位。操作自动与半字地址对齐。 当 PSIZE=2'b10，不使用 PA[1: 0]位。操作自动与字地址对齐。

#### 12.4.18. DMA 通道 4 存储器地址寄存器 (DMA\_CMAR4)

Address offset: 0x50

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道存储器地址。 通道存储器地址，作为数据传输的源或目标。 当 MSIZE=2'b01，不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10，不使用 MA[1: 0]位。操作自动与字地址对齐。

#### 12.4.19. DMA 通道 5 配置寄存器 (DMA\_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN

-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道存储器到存储器模式。 0: 禁止 1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道优先级配置。 00: 低 01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道外设数据宽度。 00: 8位; 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道半传输中断 (HT) 使能。 0: 禁止



Bit	Name	R/W	Reset Value	Function
				1: HT 中断使能
1	TCIE	RW	0	通道传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道使能。 0: 禁止 1: 通道使能

### 12.4.20. DMA 通道 5 传输个数寄存器 (DMA\_CNDTR5)

Address offset: 0x5C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	通道数据传输数量。 数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR5.EN=0) 时写入。通道使能后该寄存器为只读, 表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。 数据传输结束后, 寄存器的内容或者变为0, 或者当该通道配置为循环模式时, 寄存器的内容将被自动重新加载为之前配置时的数值。 当该寄存器值为0时, 即使 DMA 通道开始, 都不会传输数据。

### 12.4.21. DMA 通道 5 外设地址寄存器 (DMA\_CPAR5)

Address offset: 0x60

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	通道外设地址。 通道外设数据寄存器的基址，作为数据传输的源或目标。 当 PSIZE=2'b01，不使用 PA[0]位。操作自动与半字地址对齐。 当 PSIZE=2'b10，不使用 PA[1: 0]位。操作自动与字地址对齐。

### 12.4.22. DMA 通道 5 存储器地址寄存器 (DMA\_CMAR5)

Address offset: 0x64

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道存储器地址。 通道存储器地址，作为数据传输的源或目标。 当 MSIZE=2'b01，不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10，不使用 MA[1: 0]位。操作自动与字地址对齐。

### 12.4.23. DMA 通道 6 配置寄存器 (DMA\_CCR6)

Address offset: 0x6C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道存储器到存储器模式。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道优先级配置。 00: 低 01: 中等 10: 高 11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道存储器数据宽度。 00: 8位; 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道外设数据宽度。 00: 8位; 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能
5	CIRC	RW	0	通道循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道使能。

Bit	Name	R/W	Reset Value	Function
				0: 禁止 1: 通道使能

#### 12.4.24. DMA 通道 6 传输个数寄存器 (DMA\_CNDTR6)

Address offset: 0x70

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	通道数据传输数量。 数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR6.EN=0) 时写入。通道使能后该寄存器为只读, 表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。 数据传输结束后, 寄存器的内容或者变为0, 或者当该通道配置为循环模式时, 寄存器的内容将被自动重新加载为之前配置时的数值。 当该寄存器值为0时, 即使 DMA 通道开始, 都不会传输数据。

#### 12.4.25. DMA 通道 6 外设地址寄存器 (DMA\_CPAR6)

Address offset: 0x74

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	通道外设地址。 通道外设数据寄存器的基址, 作为数据传输的源或目标。 当 PSIZE=2'b01, 不使用 PA[0]位。操作自动与半字地址对齐。

Bit	Name	R/W	Reset Value	Function
				当 PSIZE=2'b10, 不使用 PA[1: 0]位。操作自动与字地址对齐。

#### 12.4.26. DMA 通道 6 存储器地址寄存器 (DMA\_CMAR6)

Address offset: 0x78

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道存储器地址。 通道存储器地址, 作为数据传输的源或目标。 当 MSIZE=2'b01, 不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10, 不使用 MA[1: 0]位。操作自动与字地址对齐。

#### 12.4.27. DMA 通道 7 配置寄存器 (DMA\_CCR7)

Address offset: 0x80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1: 0]		MSIZE[1: 0]		PSIZE[1: 0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	MEM2MEM	RW	0	通道存储器到存储器模式。 0: 禁止 1: 存储器到存储器模式使能
13: 12	PL[1: 0]	RW	0	通道优先级配置。 00: 低 01: 中等 10: 高

Bit	Name	R/W	Reset Value	Function
				11: 很高
11: 10	MSIZE[1: 0]	RW	0	通道存储器数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
9: 8	PSIZE[1: 0]	RW	0	通道外设数据宽度。 00: 8位 01: 16位 10: 32位 11: 保留
7	MINC	RW	0	通道存储器地址增量模式。 0: 禁止 1: 存储器地址增量模式使能
6	PINC	RW	0	通道外设地址增量模式。 0: 禁止 1: 外设地址增量模式使能;
5	CIRC	RW	0	通道循环模式。 0: 禁止 1: 循环模式使能
4	DIR	RW	0	通道数据传输方向。 0: 从外设读 1: 从存储器读
3	TEIE	RW	0	通道传输错误中断 (TE) 使能。 0: 禁止 1: TE 中断使能
2	HTIE	RW	0	通道半传输中断 (HT) 使能。 0: 禁止 1: HT 中断使能
1	TCIE	RW	0	通道传输完成中断 (TC) 使能。 0: 禁止 1: TC 中断使能
0	EN	RW	0	通道使能。 0: 禁止 1: 通道使能

### 12.4.28. DMA 通道 7 传输个数寄存器 (DMA\_CNDTR7)

Address offset: 0x84

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	NDT[15: 0]	RW	0	<p>通道数据传输数量。</p> <p>数据传输数量为0~65535.该寄存器只在通道不工作 (DMA_CCR7.EN=0) 时写入。通道使能后该寄存器为只读，表明剩余传输字节数。该寄存器值在每次 DMA 传输后递减。</p> <p>数据传输结束后，寄存器的内容或者变为0，或者当该通道配置为循环模式时，寄存器的内容将被自动重新加载为之前配置时的数值。</p> <p>当该寄存器值为0时，即使 DMA 通道开始，都不会传输数据。</p>

### 12.4.29. DMA 通道 7 外设地址寄存器 (DMA\_CPAR7)

Address offset: 0x88

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	PA[31: 0]	RW	0	<p>通道外设地址。</p> <p>通道外设数据寄存器的基址，作为数据传输的源或目标。</p> <p>当 PSIZE=2'b01，不使用 PA[0]位。操作自动与半字地址对齐。</p> <p>当 PSIZE=2'b10，不使用 PA[1: 0]位。操作自动与字地址对齐。</p>

### 12.4.30. DMA 通道 7 存储器地址寄存器 (DMA\_CMAR7)

Address offset: 0x8C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	MA[31: 0]	RW	0	通道存储器地址。 通道存储器地址，作为数据传输的源或目标。 当 MSIZE=2'b01，不使用 MA[0]位。操作自动与半字地址对齐。 当 MSIZE=2'b10，不使用 MA[1:0]位。操作自动与字地址对齐。



## 13. 中断和事件

### 13.1. 嵌套向量中断控制器 (NVIC)

#### 13.1.1. 主要特性

- 32 个可屏蔽的中断通道 (不包括 CPU 内部中断)
- 4 个可编程的优先级 (2 位中断优先级)
- 低延迟的异常和中断处理
- 功耗管理控制

NVIC 和 CPU 接口是紧密耦合的, 这使得低延迟中断处理和晚到达中断的高效处理成为可能。包括 CPU 的 exception, 所有中断都被 NVIC 管理。

#### 13.1.2. 系统嘀嗒 (SysTick) 校准值寄存器

系统嘀嗒校准值被设为 9000, 通过 SysTick 时钟置为 9MHz (Max fHCLK/8), 给出了 1ms 的参考 time base。

#### 13.1.3. 中断和异常向量

位置	优先级	优先级类型	名称	说明	地址
-	-	-	-	保留	0x0000_0000
-	-3	固定	复位	复位	0x0000_0004
-	-2	固定	NMI_Handler	RCC 时钟安全系统 (CSS) 联接到 NMI 向量	0x0000_0008
-	-1	固定	HardFualt_Handler	所有类型的失效	0x0000_000C
-	3	可设置	SVCAll	通过 SWI 指令的系统服务调用	0x0000_002C
-	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
-	6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	电源电压检测中断 (EXTI line 16)	0x0000_0044
2	9	可设置	RTC	RTC 中断 (combined EXTI lines 19)	0x0000_0048
3	10	可设置	Flash	Flash 全局中断	0x0000_004C
4	11	可设置	RCC	RCC 全局中断	0x0000_0050
5	12	可设置	EXTI0_1	EXTI line[1: 0] interrupt	0x0000_0054
6	13	可设置	EXTI2_3	EXTI line[3: 2] interrupt	0x0000_0058
7	14	可设置	EXTI4_15	EXTI line[15: 4] interrupt	0x0000_005C
8	15	可设置	LCD	LCD 全局中断	0x0000_0060
9	16	可设置	DMA_Channel1	DMA 通道 1 中断	0x0000_0064
10	17	可设置	DMA_Channel2_3	DMA 通道 2 & 3 中断	0x0000_0068

位置	优先级	优先级类型	名称	说明	地址
11	18	可设置	DMA_Channel4_5_6_7	DMA 通道 4 & 5 & 6 & 7中断	0x0000_006C
12	19	可设置	ADC_COMP	ADC and COMP interrupts (COMP combined with EXTI 17 & 18 & 20)	0x0000_0070
13	20	可设置	TIM1_BRK_UP_TRG_COM	TIM1断开、更新、触发和通信中断	0x0000_0074
14	21	可设置	TIM1_CC	TIM1捕获/比较中断	0x0000_0078
15	22	可设置	TIM2	TIM2全局中断	0x0000_007C
16	23	可设置	TIM3	TIM3全局中断	0x0000_0080
17	24	可设置	TIM6/LPTIM1/DAC	TIM6/LPTIM/DAC 全局中断	0x0000_0084
18	25	可设置	TIM7	TIM7全局中断	0x0000_0088
19	26	可设置	TIM14	TIM14全局中断	0x0000_008C
20	27	可设置	TIM15	TIM15全局中断	0x0000_0090
21	28	可设置	TIM16	TIM16全局中断	0x0000_0094
22	29	可设置	TIM17	TIM17全局中断	0x0000_0098
23	30	可设置	I2C1	I2C1全局中断	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	0x0000_00AC
28	35	可设置	USART2	USART2全局中断	0x0000_00B0
29	36	可设置	USART3_4	USART3_4全局中断	0x0000_00B4
30	37	可设置	CAN	CAN 全局中断	0x0000_00B8
31	38	可设置	USB	USB 全局中断	0x0000_00BC

## 13.2. 外部中断/事件控制器 (EXTI)

扩展中断和事件控制器，通过 configurable (可配置) 和 direct (直接事件) 输入 (Lines)，管理着 CPU 和系统唤醒功能，并输出下述请求信号：

- 中断请求，送给 CPU 的 IRQ
- 事件请求，送给 CPU 的事件输入 (RXEV)
- 唤醒请求，送给功耗管理控制模块

EXTI 唤醒请求允许系统从 Stop 模式唤醒，中断请求和事件请求也可以在 Run 模式使用。

EXTI 允许管理多达 22 个可配置/直接事件 Line (20 个可配置事件 Line 和 2 个直接事件 line)。

### 13.2.1. EXTI 主要特性

- 系统可以通过 GPIO 和指定模块 (PVD/COMP/RTC/LPTIM) 输入事件唤醒
- 可配置型事件 (来自 I/O, 或无状态 Pending 位的外设, 产生脉冲的外设)
  - 可选有效触发沿 (上升沿/下降沿)
  - 中断挂起标志位
  - 独立中断和事件产生屏蔽位

- 可软件触发
- Direct 型事件（具有关联标志和中断 Pending 状态位的外设）
  - 固定的上升沿触发
  - 在 EXTI 模块里没有中断 Pending 位
  - 独立中断和事件产生屏蔽位
  - 无软件触发
- 可配置 IO 端口选择

### 13.2.2. EXTI 框图

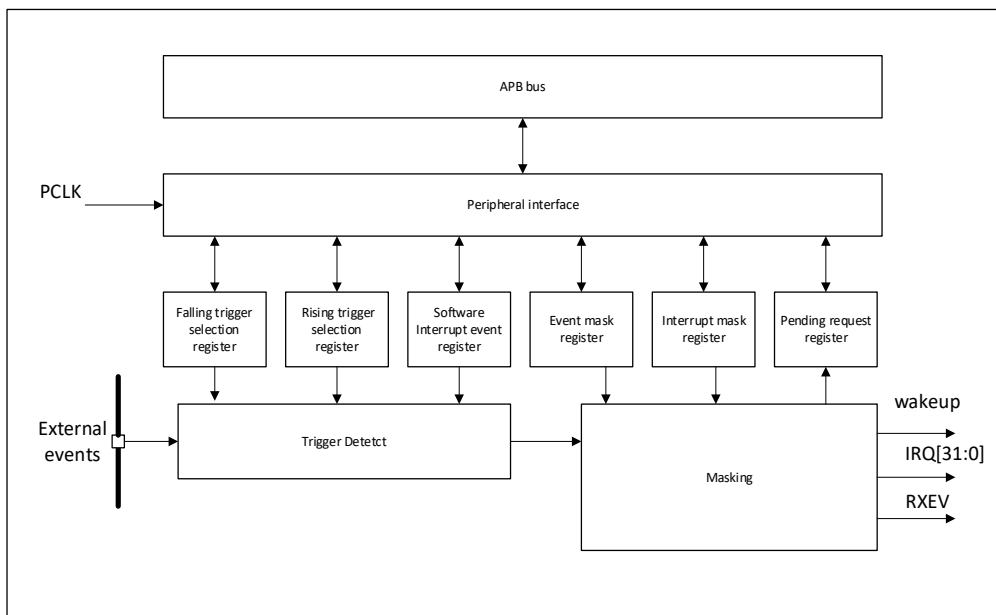


图 13-1 EXTI 框图

### 13.2.3. 中断管理

唤醒事件可以通过以下两种方式之一生成：

#### ■ 中断方式

在外设控制寄存器中启用中断，但在 NVIC（嵌套向量中断控制器）中启用，并在 Cortex-M0 系统控制寄存器中启用 SEVONPEND 位。

当 MCU 从 WFE 状态恢复时，必须清除 EXTI 外设中断挂起位和外设 NVIC IRQ 通道的挂起位（位于 NVIC 中断清除挂起寄存器中）。

#### ■ 事件方式

配置外部或内部 EXTI 线为事件模式。当 CPU 从 WFE 状态恢复时，不需要清除外设中断挂起位或 NVIC IRQ 通道的挂起位，因为与事件线对应的挂起位不会被设置。

### 13.2.4. 功能描述

对于外部中断线，要生成中断，必须配置并启用中断线。这通过编程两个触发寄存器来设置所需的边沿检测，并通过向中断屏蔽寄存器中对应的位写入 '1' 来启用中断请求来完成。当选定的边沿出现在

外部中断线上时，会生成一个中断请求。同时，对应于该中断线的挂起位也会被设置。通过向挂起寄存器中对应的位写入 '1' 可以清除这个请求。

对于内部中断线，活动边沿始终为上升沿，中断默认在中断屏蔽寄存器中启用，并且挂起寄存器中没有对应的挂起位。

为了生成事件，事件线需要进行配置并启用。这通过编程两个触发寄存器来设置所需的边沿检测，并通过向事件屏蔽寄存器中对应的位写入 '1' 来启用事件请求来完成。当选定的边沿出现在事件线上时，会生成一个事件脉冲。对应于该事件线的挂起位不会被设置。

对于外部线，还可以通过软件在软件中断/事件寄存器中写入 '1' 来生成中断或事件请求。

注意：与内部线相关的中断或事件仅在系统处于 Stop 模式时才能触发。如果系统仍在运行，则不会生成任何中断或事件。

### 13.2.5. 硬件中断选择

Direct 类型事件会在 EXTI 模块产生中断，并会产生唤醒系统和 CPU 子系统的事件信号。CPU 在处理该种类型触发事件产生的中断时，要清零外设模块的中断状态位。

要将某条线配置为中断源，请按照以下步骤操作：

- 配置 EXTI\_IMR 寄存器中的对应掩码位：通过设置 EXTI\_IMR（中断屏蔽寄存器）中对应的位来启用该中断线。
- 配置中断线的触发选择位：通过编程 EXTI\_RTSCR（上升沿触发选择寄存器）和 EXTI\_FTSCR（下降沿触发选择寄存器）来设置所需的边沿检测（上升沿、下降沿或两者）。
- 配置 NVIC 中断通道的使能和屏蔽位：配置控制映射到 EXTI 的 NVIC IRQ 通道的使能和屏蔽位，以便来自某条 EXTI 线的中断能够正确地被响应。

### 13.2.6. 硬件事件选择

要将某条线配置为事件源，请按照以下步骤操作：

- 配置 EXTI\_EMR 寄存器中的对应掩码位：通过设置 EXTI\_EMR（事件屏蔽寄存器）中对应的位来启用该事件线。
- 配置事件线的触发选择位：通过编程 EXTI\_RTSCR 和 EXTI\_FTSCR 来设置所需的边沿检测（上升沿、下降沿或两者）。

### 13.2.7. 软件中断/事件选择

任何外部线都可以配置为软件中断/事件线。以下是生成软件中断的步骤：

- 配置对应的掩码位：根据需要配置 EXTI\_IMR（中断屏蔽寄存器）或 EXTI\_EMR（事件屏蔽寄存器）中对应的位。
- 设置软件中断寄存器中的相应位：通过设置 EXTI\_SWIER（软件中断/事件寄存器）中所需的位来生成软件中断

### 13.2.8. EXTI 选择器

GPIO 被用以下方式连接到 16 个外部中断/事件 line 上:

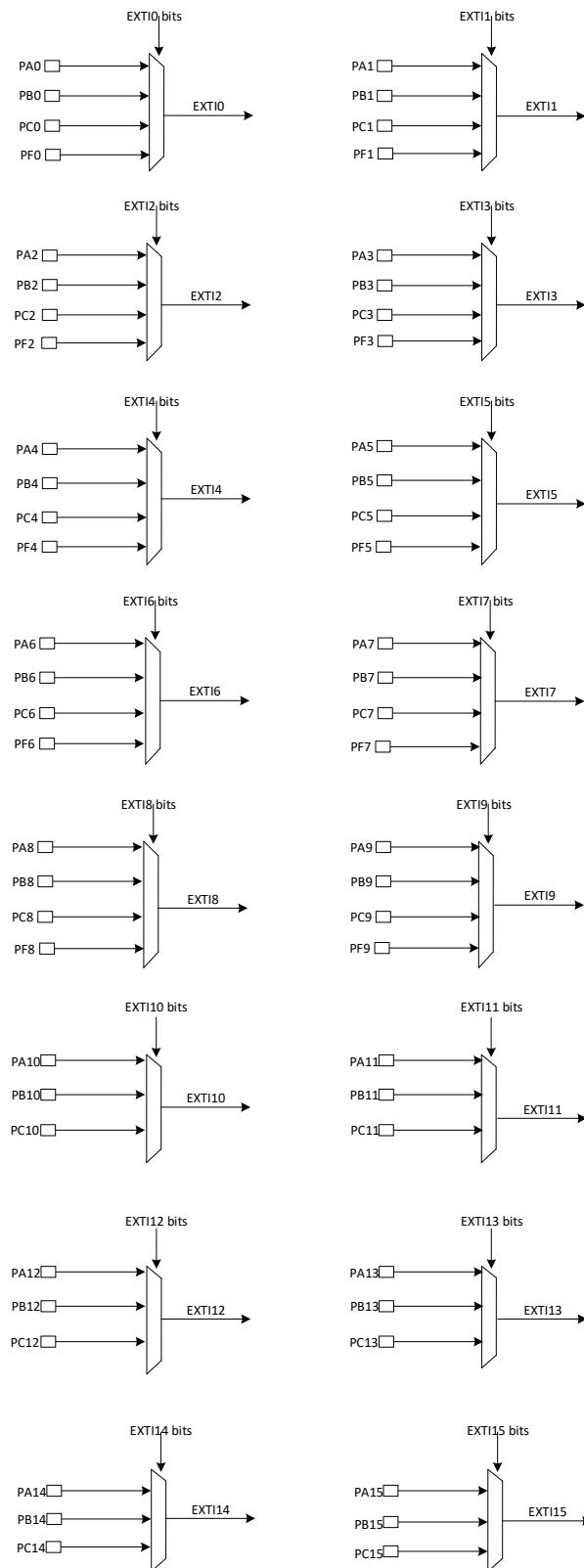


图 13-2 外部中断/事件 GPIO 映像

所有 line 连接内容如下表所示:

EXTI line	Line source	Line type
Line 0-15	GPIO	Configurable
Line 16	PVD output	Configurable
Line 17	COMP 1 output	Configurable
Line 18	COMP 2 output	Configurable
Line 19	RTC	Direct
Line 20	COMP3 output	Configurable
Line 21~28	保留	-
Line 29	LPTIM	Direct
Line 30~31	保留	-

### 13.3. EXTI 寄存器

该外设的寄存器可以用 word (32bit)、half-word (16bit) 和 byte (8bit) 访问。

#### 13.3.1. 上升沿触发选择寄存器 (EXTI\_RTSTR)

Address offset: 0x00

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											RT20	Res	RT18	RT17	RT16
-											RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	保留
20	RT20	RW	0	Configurable 类型 EXTI line20上升沿触发配置。 0: 禁止 1: 使能
19	保留	-	-	保留
18	RT18	RW	0	Configurable 类型 EXTI line18上升沿触发配置。 0: 禁止 1: 使能
17	RT17	RW	0	Configurable 类型 EXTI line17上升沿触发配置。 0: 禁止 1: 使能

Bit	Name	R/W	Reset Value	Function
16	RT16	RW	0	Configurable 类型 EXTI line16上升沿触发配置。 0: 禁止 1: 使能
15	RT15	RW	0	Configurable 类型 EXTI line15上升沿触发配置。 0: 禁止 1: 使能
14	RT14	RW	0	Configurable 类型 EXTI line14上升沿触发配置。 0: 禁止 1: 使能
13	RT13	RW	0	Configurable 类型 EXTI line13上升沿触发配置。 0: 禁止 1: 使能
12	RT12	RW	0	Configurable 类型 EXTI line12上升沿触发配置。 0: 禁止 1: 使能
11	RT11	RW	0	Configurable 类型 EXTI line11上升沿触发配置。 0: 禁止 1: 使能
10	RT10	RW	0	Configurable 类型 EXTI line10上升沿触发配置。 0: 禁止 1: 使能
9	RT9	RW	0	Configurable 类型 EXTI line9上升沿触发配置。 0: 禁止 1: 使能
8	RT8	RW	0	Configurable 类型 EXTI line8上升沿触发配置。 0: 禁止 1: 使能
7	RT7	RW	0	Configurable 类型 EXTI line7上升沿触发配置。 0: 禁止 1: 使能
6	RT6	RW	0	Configurable 类型 EXTI line6上升沿触发配置。 0: 禁止 1: 使能
5	RT5	RW	0	Configurable 类型 EXTI line5上升沿触发配置。 0: 禁止 1: 使能
4	RT4	RW	0	Configurable 类型 EXTI line4上升沿触发配置。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 使能
3	RT3	RW	0	Configurable 类型 EXTI line3上升沿触发配置。 0: 禁止 1: 使能
2	RT2	RW	0	Configurable 类型 EXTI line2上升沿触发配置。 0: 禁止 1: 使能
1	RT1	RW	0	Configurable 类型 EXTI line1上升沿触发配置。 0: 禁止 1: 使能
0	RT0	RW	0	Configurable 类型 EXTI line0上升沿触发配置。 0: 禁止 1: 使能

Configurable line 是边沿触发的，在这些 Line 上不能产生毛刺。如果在写 EXTI\_RTISR 寄存器期间，Configurable line 出现了上升沿，相关的 Pending 位不被置位。

在同一个 Line 上可以同时设置上升和下降沿，在该情况下，两种边沿都会产生触发条件。

### 13.3.2. 下降沿触发选择寄存器 (EXTI\_FTISR)

Address offset: 0x04

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											FT20	Res	FT18	FT17	FT16
-											RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	保留
20	FT20	RW	0	Configurable 类型 EXTI line20下降沿触发配置。 0: 禁止 1: 使能
19	保留	-	-	保留
18	FT18	RW	0	Configurable 类型 EXTI line18下降沿触发配置。 0: 禁止 1: 使能



Bit	Name	R/W	Reset Value	Function
17	FT17	RW	0	Configurable 类型 EXTI line17下降沿触发配置。 0: 禁止 1: 使能
16	FT16	RW	0	Configurable 类型 EXTI line16下降沿触发配置。 0: 禁止 1: 使能
15	FT15	RW	0	Configurable 类型 EXTI line15下降沿触发配置。 0: 禁止 1: 使能
14	FT14	RW	0	Configurable 类型 EXTI line14下降沿触发配置。 0: 禁止 1: 使能
13	FT13	RW	0	Configurable 类型 EXTI line13下降沿触发配置。 0: 禁止 1: 使能
12	FT12	RW	0	Configurable 类型 EXTI line12下降沿触发配置。 0: 禁止 1: 使能
11	FT11	RW	0	Configurable 类型 EXTI line11下降沿触发配置。 0: 禁止 1: 使能
10	FT10	RW	0	Configurable 类型 EXTI line10下降沿触发配置。 0: 禁止 1: 使能
9	FT9	RW	0	Configurable 类型 EXTI line9下降沿触发配置。 0: 禁止 1: 使能
8	FT8	RW	0	Configurable 类型 EXTI line8下降沿触发配置。 0: 禁止 1: 使能
7	FT7	RW	0	Configurable 类型 EXTI line7下降沿触发配置。 0: 禁止 1: 使能
6	FT6	RW	0	Configurable 类型 EXTI line6下降沿触发配置。 0: 禁止 1: 使能
5	FT5	RW	0	Configurable 类型 EXTI line5下降沿触发配置。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				1: 使能
4	FT4	RW	0	Configurable 类型 EXTI line4下降沿触发配置。 0: 禁止 1: 使能
3	FT3	RW	0	Configurable 类型 EXTI line3下降沿触发配置。 0: 禁止 1: 使能
2	FT2	RW	0	Configurable 类型 EXTI line2下降沿触发配置。 0: 禁止 1: 使能
1	FT1	RW	0	Configurable 类型 EXTI line1下降沿触发配置。 0: 禁止 1: 使能
0	FT0	RW	0	Configurable 类型 EXTI line0下降沿触发配置。 0: 禁止 1: 使能

Configurable line 是边沿触发的，在这些 Line 上不能产生毛刺。如果在写 EXTI\_FTSR 寄存器期间，Configurable line 出现了下降沿，相关的 Pending 位不被置位。

在同一个 Line 上可以同时设置上升和下降沿，在该情况下，两种边沿都会产生触发条件。

### 13.3.3. 软件中断事件寄存器 (EXTI\_SWIER)

Address offset: 0x08

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

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	SW2	Res	SW1	SW1	SW1
											0		8	7	6
-	-	-	-	-	-	-	-	-	-	-	RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW1	SW1	SW1	SW1	SW1	SW1	SW	SW	SW	SW	SW	SW4	SW	SW2	SW1	SW0
5	4	3	2	1	0	9	8	7	6	5		3			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	保留
20	SWI20	RW	0	Configurable 类型 EXTI line20软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断

Bit	Name	R/W	Reset Value	Function
				该位由硬件清零，读返回0（硬件清零后）或者配置值（硬件清零前）
19	保留	-	-	保留
18	SWI18	RW	0	Configurable 类型 EXTI line18软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回0（硬件清零后）或者配置值（硬件清零前）
17	SWI17	RW	0	Configurable 类型 EXTI line17软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回0.
16	SWI16	RW	0	Configurable 类型 EXTI line16软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回0（硬件清零后）或者配置值（硬件清零前）
15	SWI15	RW	0	Configurable 类型 EXTI line15软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回0（硬件清零后）或者配置值（硬件清零前）
14	SWI14	RW	0	Configurable 类型 EXTI line14软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回0.
13	SWI13	RW	0	Configurable 类型 EXTI line13软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件自清。读返回0.
12	SWI12	RW	0	Configurable 类型 EXTI line12软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件，进而产生中断 该位由硬件清零，读返回0（硬件清零后）或者配置值（硬件清零前）
11	SWI11	RW	0	Configurable 类型 EXTI line11软件上升沿触发配置。 0: 无影响

Bit	Name	R/W	Reset Value	Function
				1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
10	SWI10	RW	0	Configurable 类型 EXTI line10软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
9	SWI9	RW	0	Configurable 类型 EXTI line9软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件自清。读返回0.
8	SWI8	RW	0	Configurable 类型 EXTI line8软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
7	SWI7	RW	0	Configurable 类型 EXTI line7软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
6	SWI6	RW	0	Configurable 类型 EXTI line6软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
5	SWI5	RW	0	Configurable 类型 EXTI line5软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
4	SWI4	RW	0	Configurable 类型 EXTI line4软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
3	SWI3	RW	0	Configurable 类型 EXTI line3软件上升沿触发配置。

Bit	Name	R/W	Reset Value	Function
				0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
2	SWI2	RW	0	Configurable 类型 EXTI line2软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
1	SWI1	RW	0	Configurable 类型 EXTI line1软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)
0	SWI0	RW	0	Configurable 类型 EXTI line0软件上升沿触发配置。 0: 无影响 1: 产生上升沿触发事件, 进而产生中断 该位由硬件清零, 读返回0 (硬件清零后) 或者配置值 (硬件清零前)

### 13.3.4. 挂起寄存器 (EXTI\_PR)

Address offset: 0x0C

Reset value: 0x0000 0000

仅包含对 configurable 事件的寄存器控制位。

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	PR20	Res	PR18	PR17	PR16
-											RC_W1	-	RC_W1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
RC_W1															

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	-
20	PR20	RC_W1	0	Configurable 类型 EXTI line20事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;

Bit	Name	R/W	Reset Value	Function
19	保留	-	-	-
18	PR18	RC_W1	0	Configurable 类型 EXTI line18事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
17	PR17	RC_W1	0	Configurable 类型 EXTI line17事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
16	PR16	RC_W1	0	Configurable 类型 EXTI line16事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
15	PR15	RC_W1	0	Configurable 类型 EXTI line15事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
14	PR14	RC_W1	0	Configurable 类型 EXTI line14事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
13	PR13	RC_W1	0	Configurable 类型 EXTI line13事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
12	PR12	RC_W1	0	Configurable 类型 EXTI line12事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;

Bit	Name	R/W	Reset Value	Function
11	PR11	RC_W1	0	Configurable 类型 EXTI line11事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
10	PR10	RC_W1	0	Configurable 类型 EXTI line10事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
9	PR9	RC_W1	0	Configurable 类型 EXTI line9事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
8	PR8	RC_W1	0	Configurable 类型 EXTI line8事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
7	PR7	RC_W1	0	Configurable 类型 EXTI line7事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
6	PR6	RC_W1	0	Configurable 类型 EXTI line6事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
5	PR5	RC_W1	0	Configurable 类型 EXTI line5事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
4	PR4	RC_W1	0	Configurable 类型 EXTI line4事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。

Bit	Name	R/W	Reset Value	Function
				0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
3	PR3	RC_W1	0	Configurable 类型 EXTI line3事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
2	PR2	RC_W1	0	Configurable 类型 EXTI line2事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
1	PR1	RC_W1	0	Configurable 类型 EXTI line1事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;
0	PR0	RC_W1	0	Configurable 类型 EXTI line0事件挂起标志。软件或者硬件产生上升沿/下降沿触发事件时, 该位置位。软件写1清零。 0: 未产生事件请求; 1: 产生上升沿/下降沿/软件触发事件请求;

### 13.3.5. 外部中断选择寄存器 1 (EXTI\_EXTICR1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI3[1: 0]		Res	Res	Res	Res	Res	Res	EXTI2[1: 0]	
-	-	-	-	-	-	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	EXTI1[1: 0]		Res	Res	Res	Res	Res	Res	EXTI0[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25: 24	EXTI3[1: 0]	RW	0	EXTI3对应 GPIO port 选择。 2'b00: PA[3] pin 2'b01: PB[3] pin



Bit	Name	R/W	Reset Value	Function
				2'b10: PC[3] pin 2'b11: PF[3] pin
23: 18	保留	-	-	保留
17: 16	EXTI2[1: 0]	RW	0	EXTI2对应 GPIO port 选择。 2'b00: PA[2] pin 2'b01: PB[2] pin 2'b10: PC[2] pin 2'b11: PF[2] pin
15: 10	保留	-	-	保留
9: 8	EXTI1[1: 0]	RW	0	EXTI1对应 GPIO port 选择。 2'b00: PA[1] pin 2'b01: PB[1] pin 2'b10: PC[1] pin 2'b11: PF[1] pin
7: 2	保留	-	-	保留
1: 0	EXTI0[1: 0]	RW	0	EXTI0对应 GPIO port 选择。 2'b00: PA[0] pin 2'b01: PB[0] pin 2'b10: PC[0] pin 2'b11: PF[0] pin

### 13.3.6. 外部中断选择寄存器 2 (EXTI\_EXTICR2)

Address offset: 0x64

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI7[1: 0]		Res	Res	Res	Res	Res	Res	EXTI6[1: 0]	
-	-	-	-	-	-	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	EXTI5[1: 0]		Res	Res	Res	Res	Res	Res	EXTI4[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25: 24	EXTI7[1: 0]	RW	0	EXTI7对应 GPIO port 选择。 2'b00: PA[7] pin 2'b01: PB[7] pin 2'b10: PC[7] pin 2'b11: PF[7] pin

23: 18	保留	-	-	保留
17: 16	EXTI6[1: 0]	RW	0	EXTI6对应 GPIO port 选择。 2'b00: PA[6] pin 2'b01: PB[6] pin 2'b10: PC[6] pin 2'b11: PF[6] pin
15: 10	保留	-	-	保留
9: 8	EXTI5[1: 0]	RW	0	EXTI5对应 GPIO port 选择。 2'b00: PA[5] pin 2'b01: PB[5] pin 2'b10: PC[5] pin 2'b11: PF[5] pin
7: 2	保留	-	-	保留
1: 0	EXTI4[1: 0]	RW	0	EXTI4对应 GPIO port 选择。 2'b00: PA[4] pin 2'b01: PB[4] pin 2'b10: PC[4] pin 2'b11: PF[4] pin

### 13.3.7. 外部中断选择寄存器 3 (EXTI\_EXTICR3)

Address offset: 0x68

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI11[1: 0]		Res	Res	Res	Res	Res	Res	EXTI10[1: 0]	
-	-	-	-	-	-	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	EXTI9[1: 0]		Res	Res	Res	Res	Res	Res	EXTI8[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25: 24	EXTI11[1: 0]	RW	0	EXTI11 对应 GPIO port 选择。 2'b00: PA[11] pin 2'b01: PB[11] pin 2'b10: PC[11] pin 2'b11: 保留
23: 18	保留	-	-	保留
17: 16	EXTI10[1: 0]	RW	0	EXTI10 对应 GPIO port 选择。 2'b00: PA[10] pin

				2'b01: PB[10] pin 2'b10: PC[10] pin 2'b11: 保留
15: 10	保留	-	-	保留
9: 8	EXTI9[1: 0]	RW	0	EXTI11 对应 GPIO port 选择。 2'b00: PA[9] pin 2'b01: PB[9] pin 2'b10: PC[9] pin 2'b11: PF[9] pin
7: 2	保留	-	-	保留
1: 0	EXTI8[1: 0]	RW	0	EXTI8对应 GPIO port 选择。 2'b00: PA[8] pin 2'b01: PB[8] pin 2'b10: PC[8] pin 2'b11: PF[8] pin

### 13.3.8. 外部中断选择寄存器 4 (EXTI\_EXTICR4)

Address offset: 0x6C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI15[1: 0]		Res	Res	Res	Res	Res	Res	EXTI14[1: 0]	
-	-	-	-	-	-	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	EXTI13[1: 0]		Res	Res	Res	Res	Res	Res	EXTI12[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	保留	-	-	保留
25: 24	EXTI15[1: 0]	RW	0	EXTI15 对应 GPIO port 选择。 2'b00: PA[15] pin 2'b01: PB[15] pin 2'b10: PC[15] pin 2'b11: 保留
23: 18	保留	-	-	保留
17: 16	EXTI14[1: 0]	RW	0	EXTI14 对应 GPIO port 选择。 2'b00: PA[14] pin 2'b01: PB[14] pin 2'b10: PC[14] pin 2'b11: 保留

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9: 8	EXTI13[1: 0]	RW	0	EXTI13 对应 GPIO port 选择。 2'b00: PA[13] pin 2'b01: PB[13] pin 2'b10: PC[13] pin 2'b11: 保留
7: 2	保留	-	-	保留
1: 0	EXTI12[1: 0]	RW	0	EXTI12对应 GPIO port 选择。 2'b00: PA[12] pin 2'b01: PB[12] pin 2'b10: PC[12] pin 2'b11: 保留

### 13.3.9. 中断屏蔽寄存器 (EXTI\_IMR)

Address offset: 0x80

Reset value: 0x2008 0000

注意: Direct 类型 line 的中断 mask 位默认为1, 即允许该 line; configurable line 的 mask 位, 默认为0, 即屏蔽该 line。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	IM29	Res	Res	Res	Res	Res	Res	Res	Res	IM20	IM19	IM18	IM17	IM16
		RW									RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29	IM29	RW	1	EXTI line29作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
28: 21	保留	-	-	保留
20	IM20	RW	0	EXTI line20作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
19	IM19	RW	1	EXTI line19作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
18	IM18	RW	0	EXTI line18作为中断唤醒 CPU 屏蔽控制。

Bit	Name	R/W	Reset Value	Function
				0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
17	IM17	RW	0	EXTI line17作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
16	IM16	RW	0	EXTI line16作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
15	IM15	RW	0	EXTI line15作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
14	IM14	RW	0	EXTI line14作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
13	IM13	RW	0	EXTI line13作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
12	IM12	RW	0	EXTI line12作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
11	IM11	RW	0	EXTI line11作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
10	IM10	RW	0	EXTI line10作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
9	IM9	RW	0	EXTI line9作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
8	IM8	RW	0	EXTI line8作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
7	IM7	RW	0	EXTI line7作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
6	IM6	RW	0	EXTI line6作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽

Bit	Name	R/W	Reset Value	Function
5	IM5	RW	0	EXTI line5作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
4	IM4	RW	0	EXTI line4作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
3	IM3	RW	0	EXTI line3作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
2	IM2	RW	0	EXTI line2作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
1	IM1	RW	0	EXTI line1作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽
0	IM0	RW	0	EXTI line0作为中断唤醒 CPU 屏蔽控制。 0: 中断唤醒屏蔽 1: 中断唤醒未屏蔽

### 13.3.10. 事件屏蔽寄存器 (EXTI\_EMR)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	EM2 9	Res	Res	Res	Res	Res	Res	Res	Res	EM2 0	EM1 9	EM1 8	EM1 7	EM1 6
-	-	RW	-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM1 5	EM1 4	EM1 3	EM1 2	EM1 1	EM1 0	EM 9	EM 8	EM 7	EM 6	EM 5	EM4	EM3	EM2	EM1	EM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29	EM29	RW	0	EXTI line29作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
28: 21	保留	-	-	保留
20	EM20	RW	0	EXTI line20作为事件唤醒 CPU 屏蔽控制。

Bit	Name	R/W	Reset Value	Function
				0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
19	EM19	RW	0	EXTI line19作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
18	EM18	RW	0	EXTI line18作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
17	EM17	RW	0	EXTI line17作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
16	EM16	RW	0	EXTI line16作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
15	EM15	RW	0	EXTI line15作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
14	EM14	RW	0	EXTI line14作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
13	EM13	RW	0	EXTI line13作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
12	EM12	RW	0	EXTI line12作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
11	EM11	RW	0	EXTI line11作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
10	EM10	RW	0	EXTI line10作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
9	EM9	RW	0	EXTI line9作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
8	EM8	RW	0	EXTI line8作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽

Bit	Name	R/W	Reset Value	Function
7	EM7	RW	0	EXTI line7作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
6	EM6	RW	0	EXTI line6作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
5	EM5	RW	0	EXTI line5作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
4	EM4	RW	0	EXTI line4作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
3	EM3	RW	0	EXTI line3作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
2	EM2	RW	0	EXTI line2作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
1	EM1	RW	0	EXTI line1作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽
0	EM0	RW	0	EXTI line0作为事件唤醒 CPU 屏蔽控制。 0: 事件唤醒屏蔽 1: 事件唤醒未屏蔽



## 14. 循环冗余校验 (CRC)

### 14.1. 简介

根据生成多项式，CRC 计算单元将输入的 32 位数据，运算产生一个 CRC 结果。

### 14.2. CRC 主要特点

- 使用 CRC-32 (以太网) 多项式:  $0x4C11DB7$   
 $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 支持 32 位数据输入
- 单个输入/输出 32 数据和结果输出共用一个寄存器
- 通用 8 位寄存器 (可用于存放临时数据)
- 计算时间: 32 bits 数据 4 个 AHB 时钟

### 14.3. CRC 功能描述

#### 14.3.1. CRC 框图

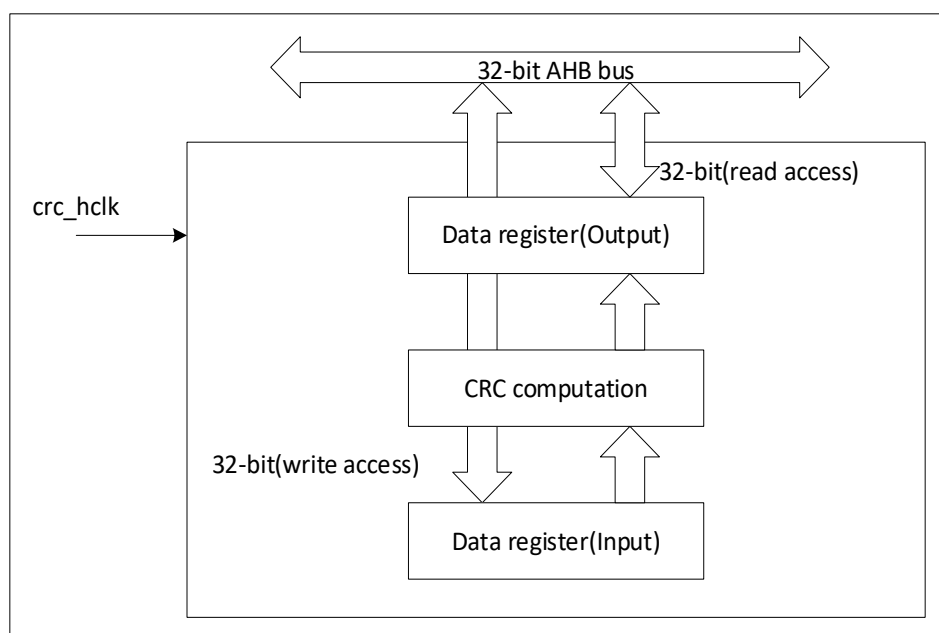


图 14-1 CRC 计算单元框图

CRC 计算单元含有 1 个 32 位数据寄存器 (CRC\_DR) :

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

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

支持配置 CRC 初始值。

可以通过设置寄存器 CRC\_CR 的 RESET 位来重置寄存器 CRC\_DR 为 0xFFFF FFFF。该操作不影响寄存器 CRC\_IDR 内的数据。

## 14.4. CRC 寄存器

### 14.4.1. 数据寄存器 (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 0	DR	RW	0xFFFF FFFF	数据寄存器。 当写新数据时，作为输入寄存器。当被读取时，保持之前 CRC 计算结果。

### 14.4.2. 独立数据寄存器 (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	IDR[7: 0]	RW	0	通用8位数据寄存器位。 可用于临时存放1字节数据。 寄存器 CRC_CR 的 RESET 位产生的 CRC 复位对本寄存器无影响。 注：此寄存器不参与 CRC 计算，可以存放任何数据。

### 14.4.3. 控制寄存器 (CRC\_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	保留	-	-	保留
0	RESET	W	0	软件置位后将会复位 CRC 模块。软件只能写1，由硬件清零。

## 15. 数字/模拟转换 (DAC)

### 15.1. DAC 简介

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

### 15.2. DAC 主要特性

- 2 个 DAC 转换器: 每个转换器对应 1 个输出通道;
- 12 位模式下数据左对齐或者右对齐;
- 同步更新功能;
- 噪声波形生成;
- 三角波形生成;
- 双 DAC 通道同时或者分别转换;
- 每个通道都有 DMA 功能;
- 支持 DMA 下溢错误检测;
- 外部触发转换;

单个 DAC 通道的框图如下图

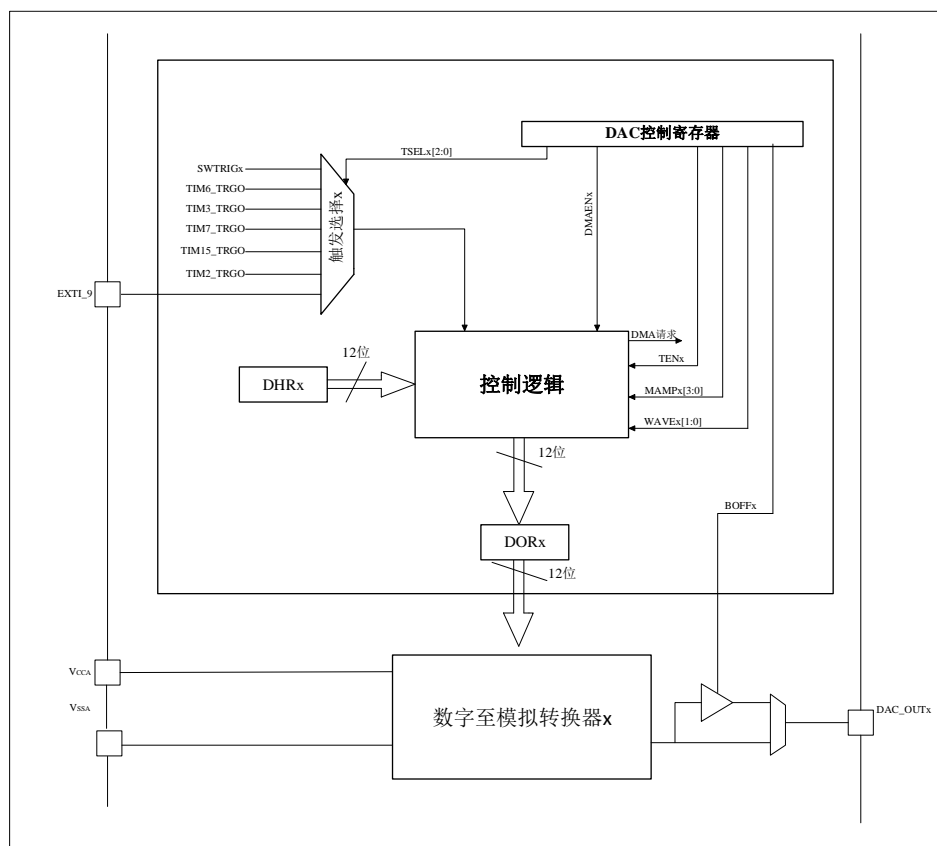


图 15-1 单 DAC 通道框图

表 15-1 DAC 引脚

名称	型号类型	描述
VCCA	输入, 模拟电源	模拟电源
VSSA	输入, 模拟电源地	模拟电源地
DAC_OUTx	模拟输出信号	DACx 模拟输出

注意: 在使能 DAC 模块前, GPIO 口 (PA4对应 DAC 通道1, PA5对应 DAC 通道2) 应配置为模拟模式。

## 15.3. DAC 功能描述

### 15.3.1. 使用 DAC 通道

将 DAC\_CR 寄存器的 ENx 位置'1'即可打开对 DAC 通道 x 的供电。经过一段启动时间 ( $t_{WAKEUP}$ ), DAC 通道 x 即被使能。

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

### 15.3.2. 使用 DAC 输出缓存

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

### 15.3.3. DAC 数据格式

根据选择的配置模式, 数据按照下文所述写入指定的寄存器:

■ 单 DAC 通道 x, 有 3 种情况:

- 8 位数据右对齐: 用户须将数据写入寄存器 DAC\_DHR8Rx[7: 0]位 (实际是存入寄存器 DHRx[11: 4]位)
- 12 位数据左对齐: 用户须将数据写入寄存器 DAC\_DHR12Lx[15: 4]位 (实际是存入寄存器 DHRx[11: 0]位)
- 12 位数据右对齐: 用户须将数据写入寄存器 DAC\_DHR12Rx[11: 0]位 (实际是存入寄存器 DHRx[11: 0]位)

根据对 DAC\_DHRyyyx (yyy 指不同的数据格式和对齐方式) 寄存器的操作, 经过相应的移位后, 写入的数据被转存到 DAC\_DHRx 寄存器中 (DHRx 是内部的数据保存寄存器 x)。随后, DAC\_DHRx 寄存器的内容或被自动地传送到 DAC\_DORx 寄存器, 或通过软件触发或外部事件触发被传送到 DAC\_DORx 寄存器。

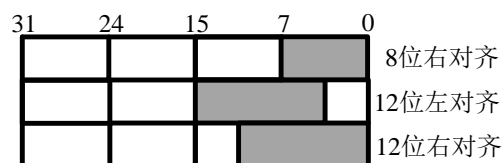


图 15-2 单 DAC 通道模式的数据寄存器

■ 双 DAC 通道，有 3 种情况：

- 8 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC\_DHR8RD[7: 0]位（实际是存入寄存器 DHR1[11: 4]位），将 DAC 通道 2 数据写入寄存器 DAC\_DHR8RD[15: 8]位（实际是存入寄存器 DHR2[11: 4]位）
- 12 位数据左对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC\_DHR12LD[15: 4]位（实际是存入寄存器 DHR1[11: 0]位），将 DAC 通道 2 数据写入寄存器 DAC\_DHR12LD[31: 20]位（实际是存入寄存器 DHR2[11: 0]位）
- 12 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC\_DHR12RD[11: 0]位（实际是存入寄存器 DHR1[11: 0]位），将 DAC 通道 2 数据写入寄存器 DAC\_DHR12RD[27: 16]位（实际是存入寄存器 DHR2[11: 0]位）

根据对 DAC\_DHRyyyx (yyy 指不同的数据格式和对齐方式) 寄存器的操作，经过相应的移位后，写入的数据被转存到 DAC\_DHR1 和 DAC\_DHR2 寄存器中 (DAC\_DHR1 和 DAC\_DHR2 是内部的数据保存寄存器 x)。随后，DAC\_DHR1 和 DAC\_DHR2 的内容通过软件触发或外部事件触发自动地传送到 DAC\_DORx 寄存器。

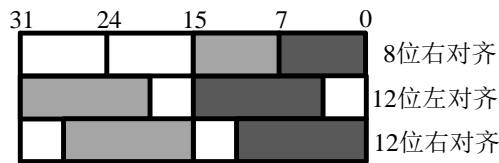


图 15-3 双 DAC 通道模式的数据寄存器

### 15.3.4. DAC 转换

不能直接对寄存器 DAC\_DORx 写入数据，任何输出到 DAC 通道 x 的数据都必须写入 DAC\_DHRx 寄存器（数据实际写入 DAC\_DHR8Rx、DAC\_DHR12Lx、DAC\_DHR12Rx、DAC\_DHR8RD、DAC\_DHR12LD、或者 DAC\_DHR12RD 寄存器）。

如果没有选中硬件触发（寄存器 DAC\_CRx 的 TENx 位置‘0’），存入寄存器 DAC\_DHRx 的数据会在 1 个 APB 时钟周期后自动传至寄存器 DAC\_DORx。如果选中硬件触发（寄存器 DAC\_CR 的 TENx 位置‘1’），数据传输在触发发生以后 3 个 APB 时钟周期后完成。

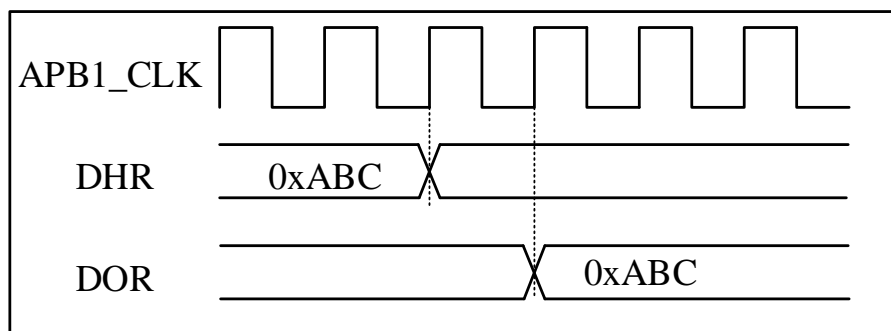


图 15-4 TEN=0时转换的时序框图

### 15.3.5. DAC 输出电压

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

任一 DAC 通道引脚上的输出电压满足下面的关系：

DAC 输出 = VCCA<sub>x</sub> (DOR / 4095) 。

### 15.3.6. 选择 DAC 触发

如果 TEN<sub>x</sub> 位被置 1，DAC 转换可以由某外部事件触发（定时器计数器、外部中断线）。配置控制位 TSEL<sub>x</sub>[2: 0]可以选择 7 个触发事件之一触发 DAC 转换。

表 15-2 外部触发

触发源	类型	TSEL <sub>x</sub> [2: 0]
定时器6 TRGO 事件	来自片上定时器的内部信号	000
定时器3 TRGO 事件		001
定时器7 TRGO 事件		010
定时器15 TRGO 事件		011
定时器2 TRGO 事件		100
EXTI 线路9	外部引脚	110
SWTRIG (软件触发)	软件控制位	111

每次 DAC 接口侦测到来自选中的定时器 TRGO 或者外部中断线 9 的上升沿，最后一次存放在寄存器 DAC\_DHR<sub>x</sub> 中的数据会被传送到寄存器 DAC\_DOR<sub>x</sub> 中。在 3 个 APB 时钟周期后，寄存器 DAC\_DOR<sub>x</sub> 更新为新值。

如果选择软件触发，一旦 SWTRIG 位置‘1’，转换即开始。在数据从 DAC\_DHR<sub>x</sub> 寄存器传送到 DAC\_DOR<sub>x</sub> 寄存器后，SWTRIG 位由硬件自动清‘0’。

注意：

1. 不能在 EN<sub>x</sub> 为‘1’时改变 TSEL<sub>x</sub>[2: 0]位。
2. 如果选择软件触发，数据从寄存器 DAC\_DHR<sub>x</sub> 传送到寄存器 DAC\_DOR<sub>x</sub> 只需要 1 个 APB 时钟周期。
3. Trigger 触发的频率不能超过 1 MHz。

### 15.3.7. DMA 功能

#### 15.3.7.1. DMA 请求

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

如果 DMAEN<sub>x</sub> 位置‘1’，一旦有外部触发（而不是软件触发）发生，则会在 3 个 APB 时钟周期后产生一个 DMA 请求，然后 DAC\_DHR<sub>x</sub> 寄存器的数据被传送到 DAC\_DOR<sub>x</sub> 寄存器。

在双 DAC 模式下（即使用 DAC\_DHR12RD 或者 DAC\_DHR12LD 或者 DAC\_DHR8RD 寄存器），DMAEN<sub>x</sub> 中仅一位应被置位。

15.3.7.2. DMA 下溢检测

DAC 的 DMA 请求没有缓冲队列，因此如果第二个外部触发在接收到对第一个外部触发的应答（第一个请求）之前到达，则不会发出新的 DMA 请求，并将 DAC\_SR 寄存器中的 DMA 下溢标志 DMAUDRx 置“1”，报告错误情况。然后禁用 DMA 数据传输，不再处理任何 DMA 请求。DAC 通道继续转换旧数据。

软件应通过写入“1”来将 DMAUDRx 标志清零，将所用 DMA 通道的 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。同时软件应修改 DAC 触发转换频率以减轻 DMA 工作负载，以避免再次发生 DMA 下溢情况。

对于各 DAC 通道，如果使能 DAC\_CR 寄存器中相应的 DMAUDRIEx 位，还将产生相应的中断。

15.3.7.3. 噪声生成

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

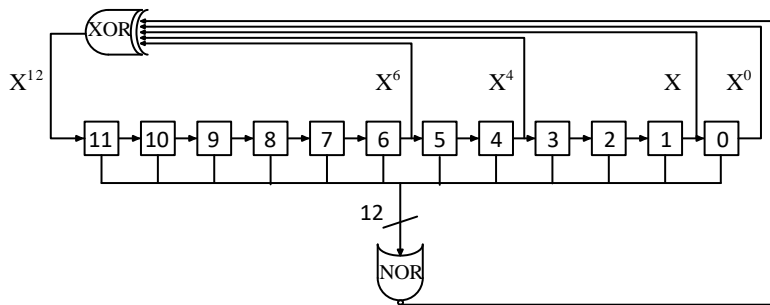


图 15-5 DAC LFSR 寄存器算法

设置 DAC\_CR 寄存器的 MAMPx[3: 0]位可以屏蔽部分或者全部 LFSR 的数据，这样得到的 LFSR 的值与 DAC\_DHRx 的数值相加，如果得到的数值有溢出，则将寄存器所能保留的最大值写入 DAC\_DORx 中，如果得到的数值没有溢出，则将该值写入 DAC\_DORx 中。

如果寄存器 LFSR 值为 0x000，则会注入‘1’（防锁定机制）。

将 WAVEx[1: 0]位置‘0’可以复位 LFSR 波形的生成算法。

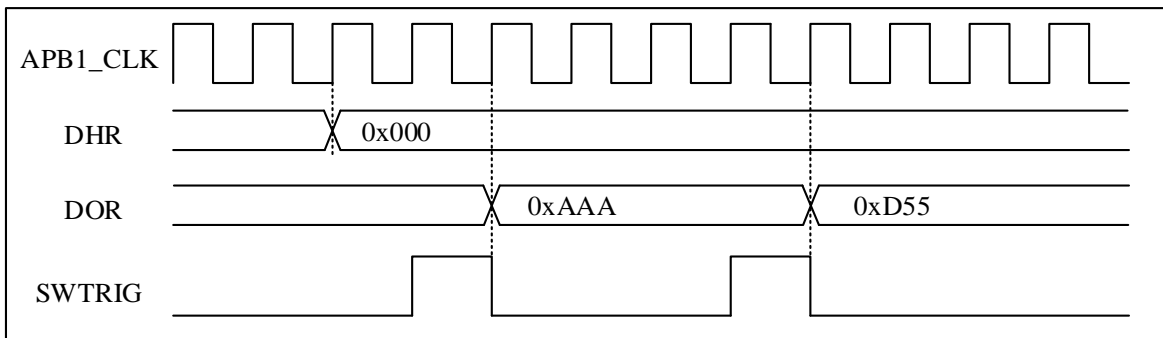


图 15-6 带 LFSR 波形生成的 DAC 转换（使能软件触发）



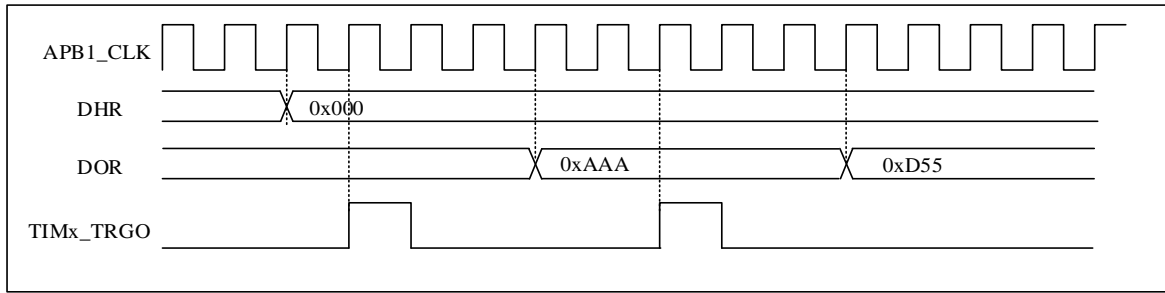


图 15-7 带 LFSR 波形生成的 DAC 转换（使能硬件触发）

- 注意：1. 为了产生噪声，必须使能 DAC 触发，即设 DAC\_CR 寄存器的 TENx 位为‘1’；  
 2. 若 DAC\_DHR 的值发生了改变，则 LFSR 寄存器的值会复位为 0xAAA；  
 3. 若 DAC\_CR.TENx 从“1”降为“0”，则 LFSR 寄存器的值也会复位为 0xAAA。

### 15.3.7.4. 三角波生成

设置 DAC\_CR.WAVEx[1: 0]位为“10”选择 DAC 的三角波生成功能。设置 DAC\_CR 寄存器的 MAMPx[3: 0]位来选择三角波的幅度。内部的三角波计数器每次触发事件之后 3 个 APB 时钟周期后累加 1。计数器的值与 DAC\_DHRx 寄存器的数值相加并丢弃溢出位后写 DAC\_DORx 寄存器。在传入 DAC\_DORx 寄存器的数值小于 DAC\_CR.MAMPx[3: 0]位定义的最大幅度时，三角波计数器逐步累加。一旦达到设置的最大幅度，则计数器开始递减，达到 0 后再开始累加，周而复始。

将 DAC\_CR.WAVEx[1: 0]位置‘0’可以复位三角波的生成。

注意：当基值为寄存器 DAC\_DORx 的最大值时，不论 DAC\_CR.MAMPx 为何值，DAC\_DORx 的输出均为其最大值。

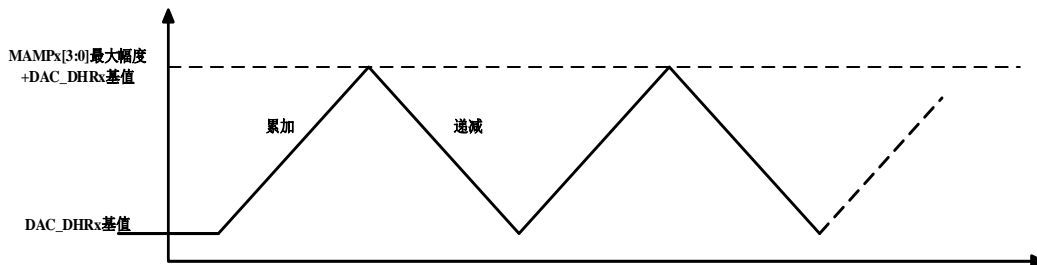


图 15-8 DAC 三角波生成

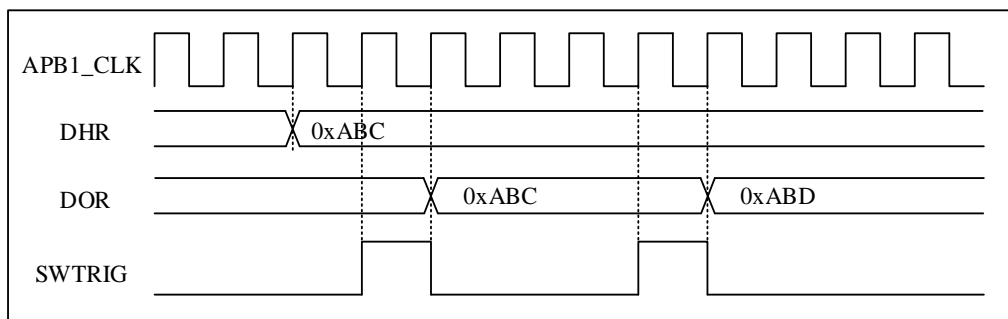


图 15-9 带三角生成的 DAC 转换（使能软件触发）

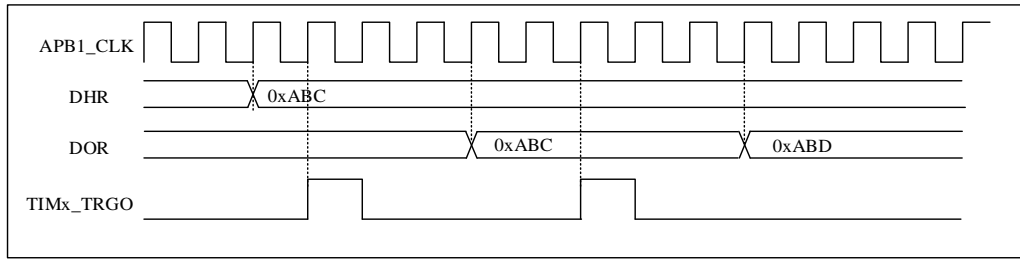


图 15-10 带三角生成的 DAC 转换（使能硬件触发）

- 注意：1. 为了产生三角波，必须使能 DAC 触发，即设 DAC\_CR 寄存器的 TENx 位为‘1’；  
 2. DAC\_CR.MAMP[3: 0]位必须在使能 DAC 之前设置，否则其值不能修改；  
 3. 若 DAC\_DHR 的值发生了改变，三角波计数器复位为 0；  
 4. 若 DAC\_CR.TENx 从“1”降为“0”，三角波计数器也复位为0。

#### 15.3.7.5. 双 DAC 通道转换

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

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

所有模式详述于以下章节。

#### 15.3.7.6. 不使用波形发生器

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生 DAC 通道 1 触发事件时，（延迟 3 个 APB 时钟周期后）寄存器 DAC\_DHR1 的值传入寄存器 DAC\_DOR1。

当发生 DAC 通道 2 触发事件时，（延迟 3 个 APB 时钟周期后）寄存器 DAC\_DHR2 的值传入寄存器 DAC\_DOR2。

#### 15.3.7.7. 使用相同的 LFSR 的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生 DAC 通道 1 触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DAC\_DHR1 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新 LFSR1 计数器。当发生 DAC 通道 2 触发事件时，具有相同屏蔽的 LFSR2 计数器值与 DAC\_DHR2 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 LFSR2 计数器。

#### 15.3.7.8. 使用不同的 LFSR 的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

- 分别设置 2 个 DAC 通道的触发使能位 DAC\_CR.TEN1 和 DAC\_CR.TEN2 为 '1'；
- 通过设置 DAC\_CR.TSEL1[2: 0] 和 DAC\_CR.TSEL2[2: 0] 位为不同值，分别配置 2 个 DAC 通道的不同触发源；
- 设置 2 个 DAC 通道的 DAC\_CR.WAVEx[1: 0] 位为 "01"，并设置 DAC\_CR.MAMPx[3: 0] 为不同的 LFSR 屏蔽值；

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

当发生 DAC 通道 1 触发事件时，按照 DAC\_CR.MAMP1[3: 0] 所设屏蔽的 LFSR1 计数器值与 DAC\_DHR1 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新 LFSR1 计数器。当发生 DAC 通道 2 触发事件时，按照 DAC\_CR.MAMP2[3: 0] 所设屏蔽的 LFSR2 计数器值与 DAC\_DHR2 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 LFSR2 计数器。

#### 15.3.7.9. 产生相同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生 DAC 通道 1 触发事件时，相同的三角波幅值加上 DAC\_DHR1 寄存器的值，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新 DAC 通道 1 三角波计数器。当发生 DAC 通道 2 触发事件时，相同的三角波幅值加上 DAC\_DHR2 寄存器的值，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 DAC 通道 2 三角波计数器。

#### 15.3.7.10. 产生不同三角波的独立触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生 DAC 通道 1 触发事件时，不同的三角波幅值加上 DAC\_DHR1 寄存器的值，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新 DAC 通道 1 三角波计数器。当发生 DAC 通道 2 触发事件时，不同的三角波幅值加上 DAC\_DHR2 寄存器的值，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 DAC 通道 2 三角波计数器。

#### 15.3.7.11. 同时软件启动

按照下列过程设置 DAC 工作在此转换模式：

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

在此配置下，一个 APB 时钟周期后，DHR1 和 DHR2 寄存器的数值即被分别传入 DAC\_DOR1 和 DAC\_DOR2 寄存器。

#### 15.3.7.12. 不使用波形发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生触发事件时，（延迟 3 个 APB 时钟周期后）DAC\_DHR1 和 DAC\_DHR2 寄存器的数值分别传入 DAC\_DOR1 和 DAC\_DOR2 寄存器。

#### 15.3.7.13. 使用相同 LFSR 的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生触发事件时，DAC\_CR.MAMP1[3: 0] 所设屏蔽的 LFSR1 计数器值与 DAC\_DHR1 寄存器的数值相加，（延迟 3 个 APB 时钟周期后）结果传入 DAC\_DOR1 寄存器，然后更新 LFSR1 计数器。

同样，DAC\_CR.MAMP2[3: 0] 所设屏蔽的 LFSR2 计数器值与 DAC\_DHR2 寄存器的数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 LFSR2 计数器。

#### 15.3.7.14. 使用不同 LFSR 的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DAC\_DHR1 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新 LFSR1 计数器。

同时，具有相同屏蔽的 LFSR2 计数器值与 DAC\_DHR2 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新 LFSR2 计数器。

#### 15.3.7.15. 使用相同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生触发事件时，相同的三角波幅值与 DAC\_DHR1 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新三角波计数器。

同时，相同的三角波幅值与 DAC\_DHR2 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新三角波计数器。

#### 15.3.7.16. 使用不同三角波发生器的同时触发

按照下列顺序设置 DAC 工作在此转换模式：

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

当发生触发事件时，DAC\_CR.MAMP1[3: 0] 所设的三角波幅值与 DAC\_DHR1 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR1，然后更新三角波计数器。

同时，MAMP2[3: 0] 所设的三角波幅值与 DAC\_DHR2 寄存器数值相加，（延迟 3 个 APB 时钟周期后）结果传入寄存器 DAC\_DOR2，然后更新三角波计数器。

#### 15.3.7.17. DAC 输出时钟

DAC 输出时钟（DAC\_CLK）为模拟端采样 DAC\_DOR 数据时钟，该时钟只有在 DAC\_CR.ENx 为 "1" 时有效。在选择硬件触发时，检测到硬件触发 Trigger 上升沿后的 5 个 APB 周期置 "1"，在保持 3 个 APB 周期后清 0；在选择软件触发时，检测到软件触发 Trigger 上升沿后的 3 个 APB 周期置 "1"，在保持 3 个 APB 周期后清 0；在选择不触发时，检测到数据变化的 2 个 APB 周期置 "1"，在保持 3 个 apb 周期后清 0。

注意：当 TEN 从 "1" 降为 "0" 后再有 1 个 APB 周期，DAC\_DOR 的数据会更新为 DAC\_DHR 寄存器中的数据。如果 DAC\_DHR 寄存器中的数据有过更新，那么当 DAC\_DOR 更新为 DAC\_DHR 的值时，会产生一个 DAC\_CLK；反之，则不会产生 DAC\_CLK。

## 15.4. DAC 寄存器

必须以字（32 位）的方式操作这些外设寄存器。

### 15.4.1. DAC 控制寄存器 (DAC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	DMAUD RIE2	DMAE N2	MAMP2[3: 0]				WAVE2[1: 0]		TSEL2[2: 0]			TEN2	BOF F2	EN2
-	-	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	DMAUD RIE1	DMAE N1	MAMP1[3: 0]				WAVE1[1: 0]		TSEL1[2: 0]			TEN1	BOF F1	EN1
-	-	RW	RW	RW				RW		RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30	保留	-	-	保留
29	DMAUDRIE2	RW	0	DAC 通道2 DMA 下溢中断使能 该位由软件设置和清除。 0: 不使能 DAC 通道2 DMA 下溢中断; 1: 使能 DAC 通道2 DMA 下溢中断。
28	DMAEN2	RW	0	DAC 通道2 DMA 使能 (DAC channel2 DMA enable) 该位由软件设置和清除。 0: 关闭 DAC 通道2 DMA 模式; 1: 使能 DAC 通道2 DMA 模式。
27: 24	MAMP2[3: 0]	RW	0	DAC 通道2屏蔽/幅值选择器 (DAC channel2 mask/amplitude selector) 由软件设置这些位, 用来在噪声生成模式下选择屏蔽位, 在三角波生成模式下选择波形的幅值。 0000: 不屏蔽 LSFR 位0/三角波幅值等于1; 0001: 不屏蔽 LSFR 位[1: 0]/三角波幅值等于3; 0010: 不屏蔽 LSFR 位[2: 0]/三角波幅值等于7; 0011: 不屏蔽 LSFR 位[3: 0]/三角波幅值等于15; 0100: 不屏蔽 LSFR 位[4: 0]/三角波幅值等于31; 0101: 不屏蔽 LSFR 位[5: 0]/三角波幅值等于63; 0110: 不屏蔽 LSFR 位[6: 0]/三角波幅值等于127; 0111: 不屏蔽 LSFR 位[7: 0]/三角波幅值等于255;

				1000: 不屏蔽 LSFR 位[8: 0]/三角波幅值等于511; 1001: 不屏蔽 LSFR 位[9: 0]/三角波幅值等于1023; 1010: 不屏蔽 LSFR 位[10: 0]/三角波幅值等于2047; ≥1011: 不屏蔽 LSFR 位[11: 0]/三角波幅值等于4095;
23: 22	WAVE2[1: 0]	RW	0	DAC 通道2噪声/三角波生成使能 (DAC channel2 noise/triangle wave generation enable) 该2位由软件设置和清除。 00: 关闭波形发生器; 01: 使能噪声波形发生器; 1x: 使能三角波发生器。
21: 19	TSEL2[2: 0]	RW	0	DAC 通道2触发选择 (DAC channel2 trigger selection) 该3位用于选择 DAC 通道2的外部触发事件。 000: TIM6 TRGO 事件 001: TIM3 TRGO 事件 010: TIM7 TRGO 事件 011: TIM15 TRGO 事件 100: TIM2 TRGO 事件 101: 保留; 110: 外部中断线9 111: 软件触发 注意: 该3位只能在 TEN2 = 1 (DAC 通道2触发使能) 时使用。
18	TEN2	RW	0	DAC 通道2触发使能 (DAC channel2 trigger enable) 该位由软件设置和清除,用来使能/关闭 DAC 通道2的触发。 0: 关闭 DAC 通道2触发, 写入 DAC_DHRx 寄存器的数据在1个 APB 时钟周期后转入 DAC_DOR2寄存器; 1: 使能 DAC 通道2触发, 写入 DAC_DHRx 寄存器的数据在3个 APB 时钟周期后转入 DAC_DOR2寄存器。 注意: 如果选择软件触发, 写入寄存器 DAC_DHRx 的数据只需要一个 APB 始终周期就可以传入寄存器 DAC_DOR2。
17	BOFF2	RW	0	关闭 DAC 通道2输出缓存 (DAC channel2 output buffer disable) 该位由软件设置和清除, 用来使能/关闭 DAC 通道2的输出缓存。 0: 使能 DAC 通道2输出缓存; 1: 关闭 DAC 通道2输出缓存。
16	EN2	RW	0	DAC 通道2使能 (DAC channel2 enable)

				该位由软件设置和清除，用来使能/关闭 DAC 通道2。 0: 关闭 DAC 通道2 1: 使能 DAC 通道2
15	保留	-	-	保留
14	保留	-	-	保留
13	DMAUDRIE1	RW	0	DAC 通道1 DMA 下溢中断使能 该位由软件设置和清除。 0: 不使能 DAC 通道1 DMA 下溢中断 1: 使能 DAC 通道1 DMA 下溢中断
12	DMAEN1	RW	0	DAC 通道1 DMA 使能 (DAC channel1 DMA enable) 该位由软件设置和清除。 0: 关闭 DAC 通道1 DMA 模式 1: 使能 DAC 通道1 DMA 模式
11: 8	MAMP1[3: 0]	RW	0	DAC 通道1屏蔽/幅值选择器 (DAC channel1 mask/amplitude selector) 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的增幅值。 0000: 不屏蔽 LSFR 位0/三角波幅值等于1 0001: 不屏蔽 LSFR 位[1: 0]/三角波幅值等于3 0010: 不屏蔽 LSFR 位[2: 0]/三角波幅值等于7 0011: 不屏蔽 LSFR 位[3: 0]/三角波幅值等于15 0100: 不屏蔽 LSFR 位[4: 0]/三角波幅值等于31 0101: 不屏蔽 LSFR 位[5: 0]/三角波幅值等于63 0110: 不屏蔽 LSFR 位[6: 0]/三角波幅值等于127 0111: 不屏蔽 LSFR 位[7: 0]/三角波幅值等于255 1000: 不屏蔽 LSFR 位[8: 0]/三角波幅值等于511 1001: 不屏蔽 LSFR 位[9: 0]/三角波幅值等于1023 1010: 不屏蔽 LSFR 位[10: 0]/三角波幅值等于2047 ≥1011: 不屏蔽 LSFR 位[11: 0]/三角波幅值等于4095
7: 6	WAVE1[1: 0]	RW	0	DAC 通道1噪声/三角波生成使能 (DAC channel1 noise/triangle wave generation enable) 该2位由软件设置和清除。 00: 关闭波形发生器; 01: 使能噪声波形发生器; 1x: 使能三角波发生器。
5: 3	TSEL1[2: 0]	RW	0	DAC 通道1触发选择 (DAC channel1 trigger selection) 该3位用于选择 DAC 通道1的外部触发事件。 000: TIM6 TRGO 事件 001: TIM3 TRGO 事件



				<p>010: TIM7 TRGO 事件</p> <p>011: TIM15 TRGO 事件</p> <p>100: TIM2 TRGO 事件</p> <p>101: 保留</p> <p>110: 外部中断线9</p> <p>111: 软件触发</p> <p>注意: 该3位只能在 TEN1 = 1 (DAC 通道1触发使能) 时使用。</p>
2	TEN1	RW	0	<p>DAC 通道1触发使能 (DAC channel1 trigger enable)</p> <p>该位由软件设置和清除,用来使能/关闭 DAC 通道1的触发。</p> <p>0: 关闭 DAC 通道1触发, 写入 DAC_DHRx 寄存器的数据在1个 APB 时钟周期后转入 DAC_DOR1寄存器;</p> <p>1: 使能 DAC 通道1触发, 写入 DAC_DHRx 寄存器的数据在3个 APB 时钟周期后转入 DAC_DOR1寄存器。</p> <p>注意: 如果选择软件触发, 写入寄存器 DAC_DHRx 的数据只需要一个 APB 始终周期就可以传入寄存器 DAC_DOR1。</p>
1	BOFF1	RW	0	<p>关闭 DAC 通道1输出缓存 (DAC channel1 output buffer disable)</p> <p>该位由软件设置和清除, 用来使能/关闭 DAC 通道1的输出缓存。</p> <p>0: 使能 DAC 通道1输出缓存</p> <p>1: 关闭 DAC 通道1输出缓存</p>
0	EN1	RW	0	<p>DAC 通道1使能 (DAC channel1 enable)</p> <p>该位由软件设置和清除, 用来使能/关闭 DAC 通道1。</p> <p>0: 关闭 DAC 通道1</p> <p>1: 使能 DAC 通道1</p>

### 15.4.2. DAC 软件触发寄存器 (DAC\_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1	SWTRIG2	RW	0	DAC 通道2软件触发 (DAC channel2 software trigger) 该位由软件设置和清除, 用来使能/关闭软件触发。 0: 关闭 DAC 通道2软件触发 1: 使能 DAC 通道2软件触发 注意: 一旦寄存器 DAC_DHR2的数据传入寄存器 DAC_DOR2, (1个 APB 时钟周期后) 该位由硬件置 '0'
0	SWTRIG1	RW	0	DAC 通道1软件触发 (DAC channel1 software trigger) 该位由软件设置和清除, 用来使能/关闭软件触发。 0: 关闭 DAC 通道1软件触发; 1: 使能 DAC 通道1软件触发。 注意: 一旦寄存器 DAC_DHR1的数据传入寄存器 DAC_DOR1, (1个 APB 时钟周期后) 该位由硬件置 '0'

### 15.4.3. DAC 通道 1 的 12 位右对齐数据保持寄存器 (DAC\_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DAC1DHR[11: 0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	DACC1DHR[11: 0]	RW	0	DAC 通道1的12位右对齐数据 (DAC channel1 12-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道1的12位数据

### 15.4.4. DAC 通道 1 的 12 位左对齐数据保持寄存器 (DAC\_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

DACC1DHR[11: 0]	Res	Res	Res	Res
RW	-			

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 4	DACC1DHR[11: 0]	RW	0	DAC 通道1的12位左对齐数据 (DAC channel1 12-bit left-aligned data ) 该位由软件写入, 表示 DAC 通道1的12位数据。
3: 0	保留	-	-	保留

#### 15.4.5. DAC 通道 1 的 8 位右对齐数据保持寄存器 (DAC\_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DACC1DHR[7: 0]							
-								RW							

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	DACC1DHR[7: 0]	RW	0	DAC 通道1的8位右对齐数据 (DAC channel1 8-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道1的8位数据。

#### 15.4.6. DAC 通道 2 的 12 位右对齐数据保持寄存器 (DAC\_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC2DHR[11: 0]											
-				RW											

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

11: 0	DACC2DHR[11: 0]	RW	0	DAC 通道2的12位右对齐数据 (DAC channel2 12-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道2的12位数据。
-------	-----------------	----	---	--

#### 15.4.7. DAC 通道 2 的 12 位左对齐数据保持寄存器 (DAC\_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11: 0]												Res	Res	Res	Res
RW												-			

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 4	DACC2DHR[11: 0]	RW	0	DAC 通道2的12位左对齐数据 (DAC channel2 12-bit left-aligned data ) 该位由软件写入, 表示 DAC 通道2的12位数据。
3: 0	保留	-	-	保留

#### 15.4.8. DAC 通道 2 的 8 位右对齐数据保持寄存器 (DAC\_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DACC2DHR[7: 0]							
-								RW							

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	DACC2DHR[7: 0]	RW	0	DAC 通道2的8位右对齐数据 (DAC channel2 8-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道2的8位数据。

### 15.4.9. 双 DAC 的 12 位右对齐数据保持寄存器 (DAC\_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	DAC2DHR[11: 0]											
-				RW											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DAC1DHR[11: 0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31: 28	保留	-	-	保留
27: 16	DACC2DHR[11: 0]	RW	0	DAC 通道2的12位右对齐数据 (DAC channel2 12-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道2的12位数据。
15: 12	保留	-	-	保留
11: 0	DACC1DHR[11: 0]	RW	0	DAC 通道1的12位右对齐数据 (DAC channel1 12-bit right-aligned data ) 该位由软件写入, 表示 DAC 通道1的12位数据。

### 15.4.10. 双 DAC 的 12 位左对齐数据保持寄存器 (DAC\_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAC2DHR[11: 0]												Res	Res	Res	Res
RW												-			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC1DHR[11: 0]												Res	Res	Res	Res
RW												-			

Bit	Name	R/W	Reset Value	Function
31: 20	DACC2DHR[11: 0]	RW	0	DAC 通道2的12位左对齐数据 (DAC channel2 12-bit left-aligned data ) 该位由软件写入, 表示 DAC 通道2的12位数据。
19: 16	保留	-	-	保留
15: 4	DACC1DHR[11: 0]	RW	0	DAC 通道1的12位左对齐数据 (DAC channel1 12-bit left-aligned data )

				该位由软件写入，表示 DAC 通道1的12位数据。
3: 0	保留	-	-	保留

### 15.4.11. 双 DAC 的 8 位右对齐数据保持寄存器 (DAC\_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7: 0]								DACC1DHR[7: 0]							
RW								RW							

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 8	DACC2DHR[7: 0]	RW	0	DAC 通道2的8位右对齐数据 (DAC channel2 8-bit right-aligned data ) 该位由软件写入，表示 DAC 通道2的8位数据。
7: 0	DACC1DHR[7: 0]	RW	0	DAC 通道1的8位右对齐数据 (DAC channel1 8-bit right-aligned data ) 该位由软件写入，表示 DAC 通道1的8位数据。

### 15.4.12. DAC 通道 1 数据输出寄存器 (DAC\_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC1DOR[11: 0]											
-				R											

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	DACC1DOR[11: 0]	R	0	DAC 通道1输出数据 (DAC channel1 data output ) 该位只读，表示 DAC 通道1的输出数据。

### 15.4.13. DAC 通道 2 数据输出寄存器 (DAC\_DOR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC2DOR[11: 0]											
-				R											

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	DACC2DOR[11: 0]	R	0	DAC 通道2输出数据 (DAC channel2 data output ) 该位只读, 表示 DAC 通道2的输出数据。

### 15.4.14. DAC 状态寄存器 (DAC\_SR)

Address offset: 0x34

Reset value: 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	-												

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29	DMAUDR2	RC_W1	0	DAC 通道2 DMA underrun flag 该位由硬件置1, 软件写1清0 0: DAC 通道2没有 DMA 下溢情况发生; 1: DAC 通道2有 DMA 下溢情况发生。 注意: 只有在 DMAUDR2置“1”后, 软件才能写“1” 清“0”
28: 14	保留	-	-	保留
13	DMAUDR1	RC_W1	0	DAC 通道1 DMA underrun flag 该位由硬件置1, 软件写1清0 0: DAC 通道1没有 DMA 下溢情况发生;

				1: DAC 通道1有 DMA 下溢情况发生。 注意: 只有在 DMAUDR1置 “1” 后, 软件才能写 “1” 清 “0”
12: 0	保留	-	-	保留



## 16. 模拟/数字转换 (ADC)

### 16.1. 简介

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 24 个通道，可测量 16 个外部和 8 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阈值。

### 16.2. ADC 主要特性

- 高性能
  - 12-bit、10-bit、8-bit 和 6-bit 分辨率可配置
  - ADC 转换时间: 1  $\mu$ s@12-bit (1 Msps)
  - 自校准
  - 可编程的采样时间
  - 可编程的数据对齐模式
  - 规则组支持 DMA
- 模拟输入通道
  - 16 个外部模拟输入通道
  - 1 个内部温度传感器通道 ( $T_{\text{SENSOR}}$ )
  - 1 个内部参考电压通道 ( $V_{\text{REFINT}}$ )
  - 1 个内部参考电压输入通道 ( $V_{\text{REFBUF}}$ )
  - 3 个内部 OPA 输入电压通道
  - 2 个内部 DAC 输入电压通道
- 转换操作启动可以通过
  - 软件启动
  - 硬件启动 (TIM1、TIM2、TIM3、TIM15 或者 GPIO)
- 转换模式
  - 单次模式: 可以转换 1 个单通道
  - 扫描模式: 可以扫描一系列通道
  - 连续模式: 连续转换被选择的通道
  - 间断模式: 每次触发转换子序列通道, 多次触发直到完整序列被转换
- 中断产生
  - 在转换结束
  - 模拟看门狗事件
- 模拟看门狗

## 16.3. ADC 功能描述

### 16.3.1. ADC 框图

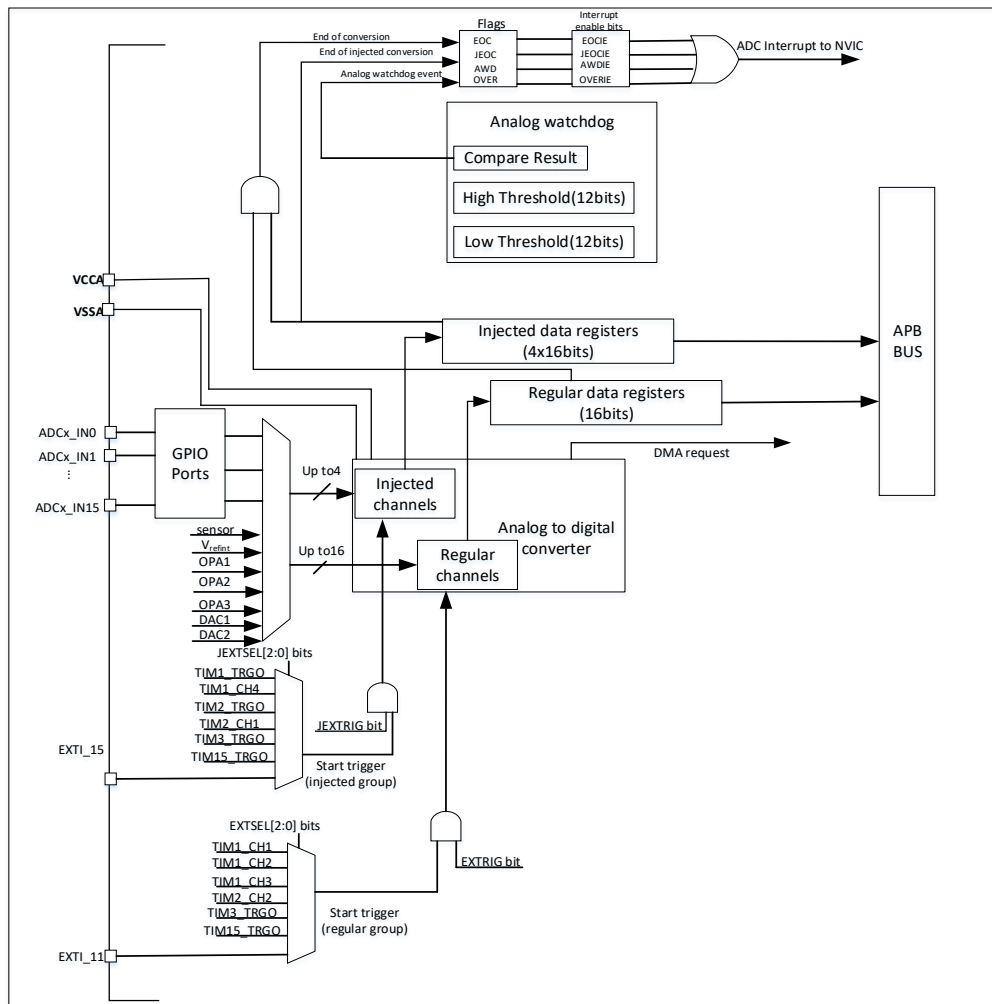


图 16-1 ADC 框图

### 16.3.2. 校准

该 ADC 具有软件校准功能。在校准期间，ADC 计算一个用于 ADC 内部的校准因子（ADC 断电后丢失）。在 ADC 校准期间、未完成校准前，应用不能使用 ADC 模块。

在使用 ADC 转换前，要进行校准操作。校准用于消除芯片和芯片之间的，由于工艺变化引起的偏移误差。

软件设置 `ADC_CR2.CAL=1` 可启动校准，校准只能在 ADC 未使能时 (`ADC_CR2.ADON=0`) 启动，且仅支持选择系统时钟作为 ADC 的时钟。当校准完成后，CAL 被硬件清 0。

当 ADC 的工作条件发生改变时 ( $V_{CCA}$  改变是 ADC offset 偏移的主要因素，温度改变次之)，推荐进行再次校准操作。

校准的软件操作过程：

- 确认 ADON=0
- 设置 CAL=1
- 等待 CAL=0

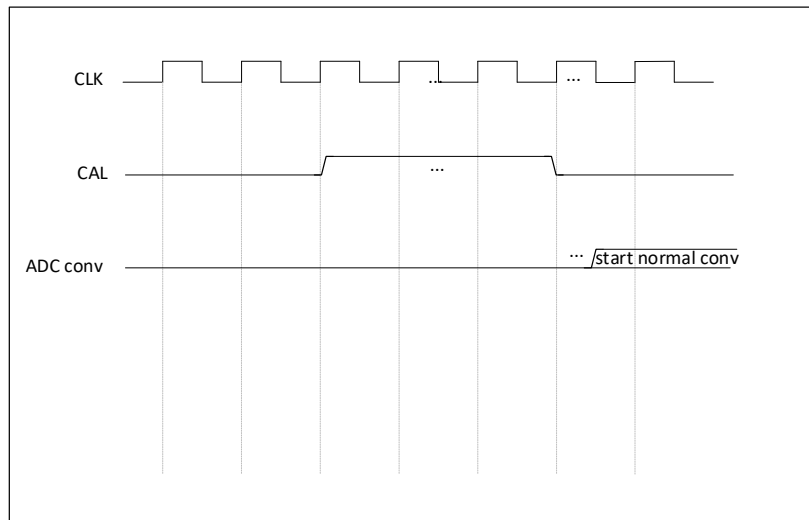


图 16-2 ADC 校准时序图

### 16.3.3. ADC 开关控制

通过设置 ADC\_CR2 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后 ( $t_{\text{STAB}}$ ，不少于  $1 \mu\text{s}$ ) 开始进行转换。

为了省电，当 ADON 为 0 时，ADC 模拟子模块将会进入掉电模式。通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。

### 16.3.4. ADC 时钟

由 RCC 控制提供的 ADCCLK 时钟和 PCLK (APB 时钟) 同步。RCC 控制器 (CLK 控制器) 为 ADC 时钟提供一个专用的可编程预分频器，ADCCLK 时钟分频详见 RCC\_CR.ADC\_DIV[22:21]。

### 16.3.5. 通道选择

有 16 个外部通道和 8 个内部通道，其中内部通道有：

- 温度传感器/ $V_{\text{REFINT}}$  内部通道

温度传感器和通道 ADC\_IN23 相连接，内部参考电压  $V_{\text{REFINT}}$  和 ADC\_IN17 相连接。

- $V_{\text{CCA}}/3$

$V_{\text{CCA}}/3$  和 通道 ADC\_IN18 相连接。

- DAC

DAC1\_VIN 和通道 ADC\_IN19 相连接，DAC2\_VIN 和通道 ADC\_IN20 相连接，

- OPA

OPA1\_VIN 和通道 ADC\_IN21 相连接，OPA2\_VIN 和通道 ADC\_IN22 相连接，OPA3\_VIN 和通道 ADC\_IN16 相连接。（注意：使用通道 16 要保证 SMP16 与 SMP0 设置成相同的值）

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

规则组由多达 16 个转换组成。规则通道和它们的转换顺序在 ADC\_SQRx 寄存器中选择。规则组中转换的总数应写入 ADC\_SQR1 寄存器的 L[3: 0]位中。

注入组由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC\_JSQR 寄存器中选择。注入组里的转换总数目应写入 ADC\_JSQR 寄存器的 JL[1: 0]位中。

如果 ADC\_SQRx 或 ADC\_JSQR 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

### 16.3.6. 可编程采样时间

ADC 使用若干个 ADC\_CLK 周期对输入电压采样，采样周期数可以通过 ADC\_SMPR1、ADC\_SMPR2 和 ADC\_SMPR3 寄存器中的 SMP[2:0]位更改。每个通道可以分别用不同的时间采样。

总转换时间如下计算：

$$t_{\text{CONV}} = \text{采样时间} + 12.5 \text{个周期 (RESSEL=00B)}$$

例如：

当  $f_{\text{ADC}}=16 \text{ MHz}$ ，采样时间为 3.5 周期

$$t_{\text{CONV}} = 3.5 + 12.5 = 16 \text{周期} = 1 \mu\text{s}$$

### 16.3.7. 可配置的分辨率

可通过降低 ADC 分辨率来执行快速转换。ADC\_CR1 寄存器的 RESSEL 位用于选择数据寄存器中可用的位数。每种分辨率的最小转换时间如下：

- 12 位：3.5 + 12.5 = 16 ADCCLK 周期
- 10 位：3.5 + 10.5 = 14 ADCCLK 周期
- 8 位：3.5 + 8.5 = 12 ADCCLK 周期
- 6 位：3.5 + 6.5 = 10 ADCCLK 周期

### 16.3.8. 单次转换模式

单次转换模式下，ADC 只执行一次转换。使能 EXTTRIG/JEXTTRIG 后，外部事件（例如定时器捕获、EXTI 中断、软件触发）触发启动转换（适用于规则通道或注入通道），这时 CONT 位为 0。

一旦选择通道的转换完成：

- 如果一个规则通道被转换：
  - 转换数据被储存在 16 位 ADC\_DR 寄存器中
  - EOC（转换结束）标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在 16 位的 ADC\_JDRx 寄存器中

- JEOC (注入转换结束) 标志被设置
- 如果设置了 JEOCIE 位, 则产生中断。

然后 ADC 停止。

### 16.3.9. 连续转换模式

在连续转换模式中, 当前面 ADC 转换一结束马上就启动另一次转换。使能 EXTTRIG/JEXTTRIG 后, 外部事件 (例如定时器捕获、EXTI 中断、软件触发) 触发启动转换 (适用于规则通道; 注入通道表现为单次转换模式), 此时 CONT 位是 1。

每个转换后:

- 如果一个规则通道被转换:
  - 转换数据被储存在 16 位的 ADC\_DR 寄存器中
  - EOC (转换结束) 标志被设置
  - 如果设置了 EOCIE, 则产生中断。
- 如果一个注入通道被转换:
  - EOC (转换结束标志) 标志置位
  - 转换数据被储存在 16 位的 ADC\_JDRx 寄存器中
  - JEOC (注入转换结束) 标志被设置
  - 如果设置了 JEOCIE 位, 则产生中断。

### 16.3.10. 扫描模式

此模式用来扫描一组模拟通道。

扫描模式可通过设置 ADC\_CR1 寄存器的 SCAN 位来选择。一旦这个位被设置, ADC 扫描所有被 ADC\_SQRx 寄存器 (对规则通道) 或 ADC\_JSQR (对注入通道) 选中的所有通道。在每个组的每个通道上执行单次转换。在每个转换结束时, 同一组的下一个通道被自动转换。如果设置了 CONT 位, 转换不会在选择组的最后一个通道上停止, 而是再次从选择组的第一个通道继续转换。

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

### 16.3.11. 间断转换模式

#### 16.3.11.1. 规则组

此模式通过设置 ADC\_CR1 寄存器上的 DISCEN 位激活。它可以用来执行一个短序列的 n 次转换 ( $n \leq 8$ ), 此转换是 ADC\_SQRx 寄存器所选择的转换序列的一部分。数值 n 由 ADC\_CR1 寄存器的 DISCNUM[2: 0]位给出。

一个外部触发信号可以启动 ADC\_SQRx 寄存器中描述的下一轮 n 次转换, 直到此序列所有的转换完成为止。总的序列长度由 ADC\_SQR1 寄存器的 L[3: 0]定义。

举例:

$n=3$ , 被转换的通道 = 0、1、2、3、6、7、9、10

第一次触发：转换的序列为 0、1、2

第二次触发：转换的序列为 3、6、7

第三次触发：转换的序列为 9、10，并产生 EOC 事件

第四次触发：转换的序列 0、1、2

注意：当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。

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

#### 16.3.11.2. 注入组

此模式通过设置 ADC\_CR1 寄存器的 JDISCEN 位激活。在一个外部触发事件后，该模式按通道顺序逐个转换 ADC\_JSQR 寄存器中选择的序列。

一个外部触发信号可以启动 ADC\_JSQR 寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由 ADC\_JSQR 寄存器的 JL[1: 0]位定义。

例子：

n=1，被转换的通道 = 1、2、3

第一次触发：通道1被转换

第二次触发：通道2被转换

第三次触发：通道3被转换，并且产生 EOC 和 JEOC 事件

第四次触发：通道1被转换

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

不能同时使用自动注入和间断模式。

#### 16.3.12. 注入通道管理

注入通道的外部触发优先级高于规则通道的外部触发，即注入通道的外部触发可以中断正在进行中的规则通道转换。注入通道有两种方式：触发注入和自动注入。

##### 16.3.12.1. 触发注入

为了使用触发注入，ADC\_CR1 寄存器的 JAUTO 位必须清零，并且设置 SCAN 位。

- 通过设置 ADC\_CR2 寄存器的 ADON 位，设置 SWSTART，外部触发启动一组规则通道的转换。
- 如果在规则通道转换期间产生一外部注入触发，当前转换被复位，注入通道序列被以单次扫描方式进行转换。
- 然后，恢复上次被中断的规则组通道转换。如果在注入转换期间产生一规则事件，注入转换不会被中断，但是规则序列不会在注入序列结束后被执行。
- 间断模式不支持触发注入。触发注入期间的规则触发是不响应的。

注：当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为 28 个 ADC 时钟周期（即 2 个具有 1.5 个时钟间隔采样时间的转换），触发之间最小的间隔必须是 29 个 ADC 时钟周期。

### 16.3.12.2. 自动注入

如果设置了 JAUTO 位，在规则组通道之后，注入组通道被自动转换。这可以用来转换在 ADC\_SQRx 和 ADC\_JSQR 寄存器中设置的多至 20 个通道。

在此模式里，必须禁止注入通道的外部触发。

如果除 JAUTO 位外还设置了 CONT 位，规则通道至注入通道的转换序列被连续执行。

注意：禁止同时使用自动注入和间断模式。

### 16.3.13. 停止进行中的转换 (ADSTP)

用软件设置 ADC\_CR1 寄存器中的 ADSTP=1 可以停上当前正在进行的转换，复位 ADC 的操作并让 ADC 进入空闲状态，为下次转换作好准备。

当 ADSTP 由软件设置为 1，任何当前的转换中止且转换结果丢弃（ADC\_DR 寄存器不用当前的转换值进行更新）。

扫描序列也被中止并复位（即重新启动 ADC 时会用新的序列进行转换）

一旦结束该过程 ADSTP 和 SWSTART 位都由硬件清 0。

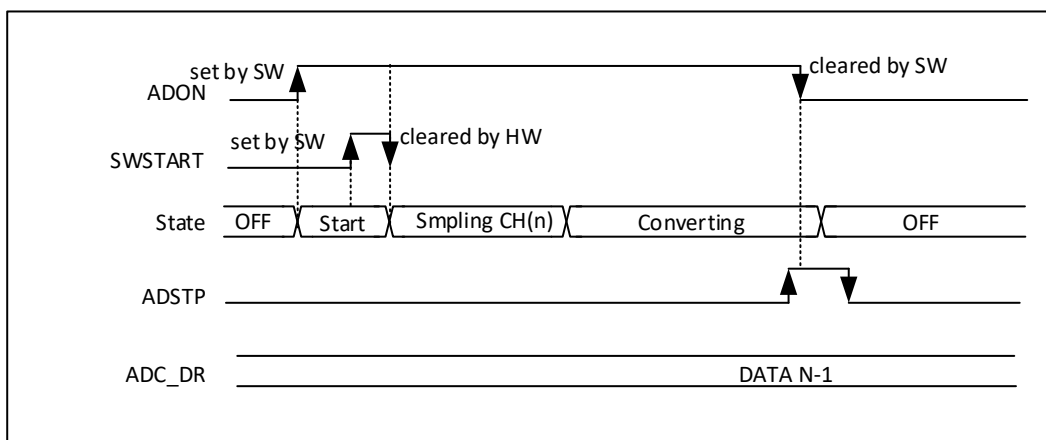


图 16-3 Stop timing

## 16.4. 模拟看门狗

如果 ADC 转换的模拟电压低于阈值下限或高于阈值上限，则 AWD 模拟看门狗状态位会置 1。这些阈值在 ADC\_HTR 和 ADC\_LTR 寄存器的 12 个最低有效位中设置。通过设置 ADC\_CR1 寄存器中的 AWDIE 位产生中断。

阈值与 ADC\_CR2 寄存器中的 ALIGN 位的所选对齐方式无关。阈值比较是在对齐之前完成的（注入通道减去偏移值之前）。

通过配置 ADC\_CR1 寄存器，模拟看门狗可以作用于 1 个或多个通道，具体如下表所示：

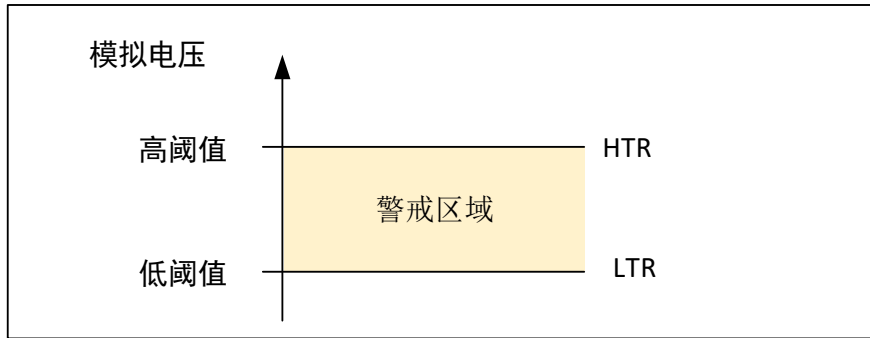


图 16-4 模拟看门狗保护区

表 16-2 模拟看门狗通道选择

模拟看门狗保护通道	ADC_CR1寄存器控制位		
	AWDSGL	AWDEN	JAWDEN
无	X	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有注入和规则通道	0	1	1
单一注入通道	1	0	1
单一规则通道	1	1	0
单一注入或规则通道	1	1	1

## 16.5. 外部触发转换

可以通过外部事件（例如定时器捕获、EXTI 中断）触发转换。如果设置了 EXTTRIG 或 JEXTTRIG 控制位，则外部事件就能够触发转换。EXTSEL[2: 0]和 JEXTSEL2: 0]控制位允许应用程序选择 8 个可能的事件中的某一个，可以触发规则和注入组的采样。

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

下表给出了转换可能的外部触发。软件源触发事件可由设置 ADC\_CR 寄存器中的 ADSTART 位来产生。

表 16-3 ADC 用于规则通道的外部触发

触发源	类型	EXTSEL[2: 0]
定时器1的 CH1输出	片上定时器的内部信号	000
定时器1的 CH2输出		001
定时器1的 CH3输出		010
定时器2的 TRGO 输出		011
定时器3的 TRGO 输出		100
定时器15的 TRGO 输出		101
EXTI 线11	外部管脚	110
SWSTART	软件控制位	111



表 16-4 ADC 用于注入通道的外部触发

触发源	类型	JEXTSEL[2: 0]
定时器1的 TRGO 输出	片上定时器的内部信号	000
定时器1的 CH4输出		001
定时器2的 TRGO 输出		010
定时器2的 CH1输出		011
定时器3的 CH4输出		100
定时器15的 TRGO 输出		101
EXTI 线15输出	外部管脚	110
JSWSTART	软件控制位	111

## 16.6. 数据对齐

在每次转换结束，转换的结果数据被存放到 16 位宽 ADC\_DR 数据寄存器中。ADC\_CR2 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如注入组通道转换的数据值已经减去了在 ADC\_JOFRx 寄存器中定义的偏移量，因此结果可以是一个负值。SEXT 位是扩展的符号值。

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

表 16-5 数据右对齐

注入组

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
------	------	------	------	-----	-----	----	----	----	----	----	----	----	----	----	----

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	-----	-----	----	----	----	----	----	----	----	----	----	----

表 16-6 数据左对齐

注入组

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
------	-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---

## 16.7. 数据过载

ADC 过载标志 (OVER) 是指一个缓冲区过载事件，规则组当转换好的数据未被 CPU 或 DMA 及时读取时，另一个转换数据已经有效时，就发生了 ADC 过载。

## 16.8. DMA 请求

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

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

## 16.9. 温度传感器和内部参考电压

温度传感器可用于测量器件周围的温度 ( $T_A$ )。温度传感器内部连接到 ADC\_IN23 通道，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入推荐采样时间是 17.1us。不使用时，可将传感器置于掉电模式。

温度传感器输出电压随温度线性变化，由于生产过程中的变化，温度变化曲线的偏移在不同芯片上会有不同（最多相差 45 °C）。内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

注意：必须设置 TSVREFE 位激活内部通道：ADC\_IN23（温度传感器）和 ADC\_IN17 ( $V_{REFINT}$ ) 的转换。

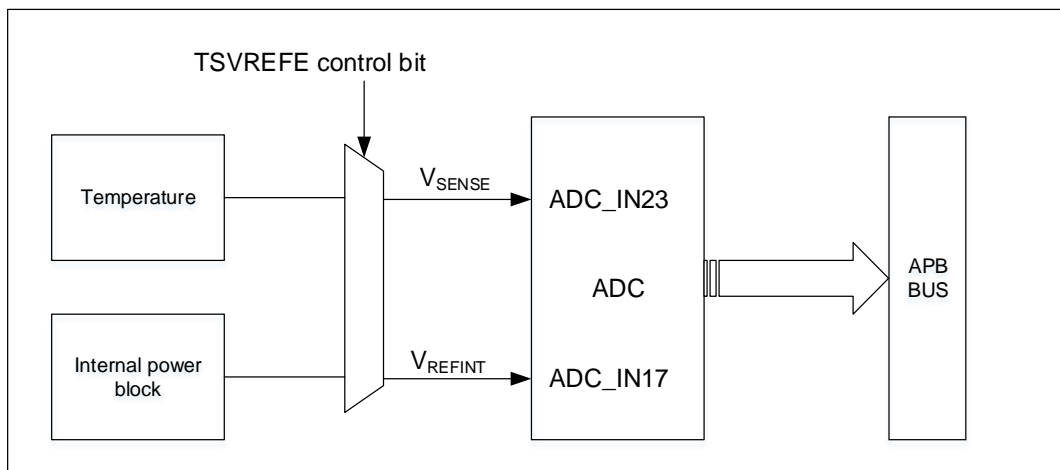


图 16-5 温度传感器通道框图

### 读温度

要使用传感器，请执行以下操作：

1. 选择 ADC\_IN23
2. 选择一个采样时间，该采样时间要大于数据手册中所指定的最低采样时间（SMP23）。
3. 在 ADC\_CR2 寄存器中将 TSVREFE 位置 1，以便将温度传感器从掉电模式中唤醒。
4. 通过将 ADON 位置 1，并使能外部触发，开始 ADC 转换
5. 读取 ADC 数据寄存器中生成的 VSENSE 数据
6. 使用以下公式计算温度：

$$\text{温度 (}^{\circ}\text{C)} = \{ (V_{\text{SENSE}} - V_{30}) / \text{Avg\_Slope} \} + 30^{\circ}\text{C}$$

其中：

$V_{30}$  = 30 °C 时的 VSENSE 值

Avg\_Slope = 温度与 VSENSE 曲线的平均斜率（单位为 mV/°C 或  $\mu\text{V}/^{\circ}\text{C}$ ）

（有关  $V_{30}$  和 Avg\_Slope 实际值的相关信息，请参见数据手册中的电气特性一节。）

注意：传感器从掉电模式中唤醒需要一个建立时间，启动时间过后输出正确水平的 VSENSE。ADC 在上电后同样需要一个建立时间，因此为了缩短延时，应同时设置 ADON 和 TSVREFE。

## 16.10. ADC 中断

规则和注入组转换结束时能产生中断，当模拟看门狗状态位被设置时能产生中断，当规则组转换数据未被及时读取时也能产生中断。它们都有独立的中断使能位。

ADC\_SR 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTRT (注入组通道转换的启动)
- STRT (规则组通道转换的启动)

表 16-7 ADC 中断

中断事件	事件标志	使能控制
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置了模拟看门狗状态位	AWD	AWDIE
溢出标志	OVER	OVERIE

## 16.11. ADC 寄存器

### 16.11.1. ADC 状态寄存器 (ADC\_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										OVER	STRT	JSTRT	JEOC	EOC	AWD
-										RC	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 6	保留	-	-	保留
5	OVER	RC	0	ADC 过载 当过载发生时，硬件置位该位。当 EOC 标志已置起表明一次新的转换已完成。 0: 无过载发生 (DMA 或 CPU 已读取上一次转换结果) 1: 过载已发生
4	STRT	RC_W0	0	规则通道开始状态位 该位由硬件在规则通道转换开始时设置，由软件清除。 0: 规则通道转换未开始 1: 规则通道转换已开始
3	JSTRT	RC_W0	0	注入通道开始状态位

Bit	Name	R/W	Reset Value	Function
				该位由硬件在注入通道组转换开始时设置，由软件清除。 0：注入通道转换未开始 1：注入通道转换已开始
2	JEOC	RC_W0	0	注入通道转换结束状态位 该位由硬件在所有注入通道组转换结束时设置，由软件清除。 0：转换未完成 1：转换完成
1	EOC	RC_W0	0	转换结束状态位 该位由硬件在（规则或注入）通道组转换结束时设置，由软件清除或由读 ADC_DR 时清除。 0：转换未完成 1：转换完成
0	AWD	RC_W0	0	模拟看门狗标志位 该位由硬件在转换的电压值超出了 ADC_LTR 或低于 ADC_HTR 寄存器定义的范围时设置1，由软件清除。 0：没有发生模拟看门狗事件 1：发生模拟看门狗事件

### 16.11.2. ADC 控制寄存器 1 (ADC\_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	OVRIE	Res	AD-STP	Res	RESSEL[1:0]	AWD EN	JAWD EN	Res							
-	RW	-	R_W1	-	RW	RW	RW	-							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]		JDIS CEN	DIS-CEN	JAU TO	AWD SGL	SCA N	JEO CIE	AWDIE	EOCIE	AWDCH[4:0]					
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29	OVRIE	RW	0	OVER FLAG 中断使能 0：禁止过载中断 1：允许过载中断
28	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
27	ADSTP	R_W1	0	ADC 转换停止使能。软件写1置1。当接收到 ADC 停止信号时，硬件置0。 1: 停止 ADC 转换 0: 不停止 ADC 转换
26	保留	-	-	保留
25: 24	RESSEL[1: 0]	RW	0	分辨率 (Resolution) 控制位。通过软件写入这些位可选择转换的分辨率。 00: 12 位 (16 ADCCLK 周期) 01: 10 位 (14 ADCCLK 周期) 10: 8 位 (12 ADCCLK 周期) 11: 6 位 (10 ADCCLK 周期)
23	AWDEN	RW	0	规则通道模拟看门狗使能。在规则通道上开启模拟看门狗。该位由软件写1置1和写0置0。 0: 在规则通道上禁用模拟看门狗 1: 在规则通道上使用模拟看门狗
22	JAWDEN	RW	0	注入通道模拟看门狗使能。在注入通道上开启模拟看门狗。该位由软件写1置1和写0置0。 0: 在注入通道上禁用模拟看门狗 1: 在注入通道上使用模拟看门狗
21: 16	保留	-	-	保留
15: 13	DISCNUM[2: 0]	RW	0	间断模式通道计数。在间断模式下，收到外部触发后，规则通道组的转换通道数。 软件写操作设置这些位。 000: 1个通道 001: 2个通道 ..... 111: 8个通道
12	JDISCEN	RW	0	注入通道间断模式使能。 该位由软件写1置1和写0置0，用于开启或关闭注入通道组上的间断模式。 0: 注入通道组上禁用间断模式 1: 注入通道组上使用间断模式
11	DISCEN	RW	0	规则通道的间断模式使能 该位由软件写1置1和写0置0，用于开启或关闭规则通道组上的间断模式 0: 规则通道组上禁用间断模式 1: 规则通道组上使用间断模式
10	JAUTO	RW	0	自动注入使能

Bit	Name	R/W	Reset Value	Function
				<p>该位由软件写1置1和写0置0，用于开启或关闭规则通道组转换结束后自动进行注入通道组转换</p> <p>0：关闭自动的注入通道组转换</p> <p>1：开启自动的注入通道组转换</p>
9	AWDSGL	RW	0	<p>单一通道看门狗使能。</p> <p>该位由软件写1置1和写0置0，用于开启或关闭由 AWDCH[4: 0]定义的通道上的模拟看门狗。</p> <p>0：在所有的通道上使用模拟看门狗</p> <p>1：在单一通道上使用模拟看门狗</p>
8	SCAN	RW	0	<p>扫描模式使能</p> <p>该位由软件写1置1和写0置0，用于开启或关闭扫描模式。在扫描模式中，由 ADC_SQRx 或 ADC_JSQRx 寄存器选中的通道被转换。</p> <p>0：关闭扫描模式</p> <p>1：使用扫描模式</p>
7	JEOCIE	RW	0	<p>注入通道转换结束中断使能</p> <p>该位由软件写1置1和写0置0。该 bit 使能后，在 JEOC 有效时，产生中断请求。</p> <p>0：禁止 JEOC 中断</p> <p>1：允许 JEOC 中断。</p>
6	AWDIE	RW	0	<p>模拟看门狗中断使能</p> <p>该位由软件写1置1和写0置0。该 bit 使能后，在 AWD 有效时，产生中断请求。</p> <p>0：禁止模拟看门狗中断</p> <p>1：允许模拟看门狗中断</p>
5	EOCIE	RW	0	<p>规则通道转换结束中断使能</p> <p>该位由软件写1置1和写0置0。该 bit 使能后，在 EOC 有效时，产生中断请求。</p> <p>0：禁止 EOC 中断</p> <p>1：允许 EOC 中断。</p>
4: 0	AWDCH[4: 0]	RW	0	<p>模拟看门狗通道选择位</p> <p>该位由软件写设置，用于选择模拟看门狗的输入通道。</p> <p>00000：ADC 模拟输入通道0</p>

Bit	Name	R/W	Reset Value	Function
				00001: ADC 模拟输入通道1 ..... 01111: ADC 模拟输入通道15 10000: OPA3_VIN 10001: V <sub>REFINT</sub> 输入 10010: V <sub>CCA/3</sub> 10011: DAC1_VIN 10100 DAC2_VIN 10101: OPA1_VIN 10110: OPA2_VIN 10111: T <sub>S_VIN</sub> 输入 保留所有其他数值。

### 16.11.3. ADC 控制寄存器 (ADC\_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				VREFBUFF_SEL		VREF BUFFERE	Res	TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2: 0]			Res
-				RW		RW	-	RW	R_W1	R_W1	RW	RW			-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG		JEXTSEL[2: 0]		ALIGN	Res	Res	DMA	Res	Res	Res	Res	RSTCAL	CAL	CONT	ADON
RW				-		-	RW	-				R_W1		RW	RW

Bit	Name	R/W	Reset Value	Function
31: 28	保留	-	-	保留
27: 26	VREFBUFF_SEL	RW	0	V <sub>REFBUF</sub> 电压选择 00: 1.5 V 01: 2.048 V 10: 2.5 V 11: 保留
25	VREF BUFFERE	RW	0	VerfBuffer 使能 软件写0置0, 写1置1。 0: 禁止 VerfBuffer 1: 使能 VerfBuffer
24	保留	-	-	保留
23	TSVREFE	RW	0	温度传感器和 V <sub>REFINT</sub> 使能

Bit	Name	R/W	Reset Value	Function
				通过软件写1置 1 和写0置0, 用于开启或禁止温度传感器和 V <sub>REFINT</sub> 通道。 ADC 中可使能, 。 0: 禁止温度传感器和 V <sub>REFINT</sub> 1: 启用温度传感器和 V <sub>REFINT</sub>
22	SWSTART	R_W1	0	开始转换规则通道使能 通过软件写1置 1启动转换, 启动转换后立即被硬件清除置0。 0: 复位状态 1: 开始转换规则通道
21	JSWSTART	R_W1	0	开始转换注入通道使能 由软件写1置 1启动转换, 启动转换后立即被硬件清除置0。 0: 复位状态 1: 开始转换注入通道
20	EXTTRIG	RW	0	规则通道外部触发使能 该位由软件写1置1和写0置0清除, 用于开启或禁止可以启动规则通道组转换的外部触发信号。 0: 禁止规则通道外部触发 1: 使能规则通道外部触发
19: 17	EXTSEL[2: 0]	RW	0	规则通道外部触发事件选择位。选择启动规则通道组转换的外部事件 ADC 外部触发事件如下: 000: 定时器1的 CH1事件 001: 定时器1的 CH2事件 010: 定时器1的 CH3事件 011: 定时器2的 CH2事件 100: 定时器3的 TRGO 事件 101: 触发 timer15_TRGO 事件 110: EXT11 111: SWSTART
16	保留	-	-	保留
15	JEXTTRIG	RW	0	注入通道外部触发使能 0: 禁止注入通道外部触发 1: 使能注入通道外部触发
14: 12	JEXTSEL[2: 0]	RW	0	注入通道组外部触发事件选择位 ADC 的外部触发事件如下: 000: 定时器1的 TRGO 事件



Bit	Name	R/W	Reset Value	Function
				001: 定时器1的 CH4事件 010: 定时器2的 TRGO 事件 011: 定时器2的 CH1事件 100: 定时器3的 TRGO 事件 101: 定时器15的 TRGO 事件 110: EXTI15 111: JSWSTART
11	ALIGN	RW	0	数据对齐控制位 该位由软件写1置1和写0置0清除。 0: 右对齐 1: 左对齐
10: 9	保留	-	-	保留
8	DMA	RW	0	DMA 使能位 该位由软件写1置1和写0置0清除。 0: 禁止 DMA 模式 1: 使能 DMA 模式
7: 4	保留	-	-	保留
3	RSTCAL	Res	0	校准复位使能位 该位由软件写1置1, 由硬件清除置0。在校准寄存器被初始化后 (即 RSTCAL 置1后), 该位即被清除。 0: 校准寄存器已初始化 1: 初始化校准寄存器 注: 当正在进行转换时, 如果设置 RSTCAL, 清除校准寄存器需要额外的周期。
2	CAL	R_W1	0	校准使能 该位由软件写1置1开始校准, 并在校准失败或者校准成功时由硬件清除。当 ADON 无效时, 且 SWSTART, JSWSTART 无效时; 软件启动校准。 0: 校准完成 1: 使能校准
1	CONT	RW	0	连续转换使能 该位由软件写1置1和写0置0清除。如果设置了此位, 则转换将连续进行直到该位被清除。 0: 单次转换模式 1: 连续转换模式

Bit	Name	R/W	Reset Value	Function
0	ADON	RW	0	开/关 A/D 转换器 ADC 转换器工作使能，当置1后 ADC 转换器唤醒，在转换器上电到开始转换有一个延迟 tSTAB。当置0后 ADC 转换器处于掉电状态。 0: 禁止 ADC 转换，ADC 转换器进入断电模式 1: 使能 ADC 转换器。 注：如果在该寄存器中 SWSTART,JSWSTART 与 ADON 一起被改变，则转换不被触发。这是为了防止触发错误的转换。

#### 16.11.4. ADC 采样时间寄存器 1 (ADC\_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				SMP23[2: 0]			SMP22[2: 0]			SMP21[2: 0]			SMP20[2: 0]		
-				RW			RW			RW			RW		

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	SMPx[2: 0]	RW	0	选择通道 x 的采样时间 通过软件设置这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000: 3.5周期      100: 28.5周期 001: 5.5周期      101: 41.5周期 010: 7.5周期      110: 134.5周期 011: 13.5周期     111: 239.5周期

#### 16.11.5. ADC 采样时间寄存器 2 (ADC\_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SMP19[2: 0]			SMP18[2: 0]			SMP17[2: 0]			SMP16[2: 0]			SMP15[2: 1]	
-		RW			RW			RW			RW			RW	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2: 0]			SMP13[2: 0]			SMP12[2: 0]			SMP11[2: 0]			SMP10[2: 0]		
RW	RW			RW			RW			RW			RW		

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29: 0	SMPx[2: 0]	RW	0	选择通道 x 的采样时间 通过软件设置这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 000: 3.5周期      100: 28.5周期 001: 5.5周期      101: 41.5周期 010: 7.5周期      110: 134.5周期 011: 13.5周期     111: 239.5周期

### 16.11.6. ADC 采样时间寄存器 3 (ADC\_SMPR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res		SMP9[2: 0]			SMP8[2: 0]			SMP7[2: 0]			SMP6[2: 0]			SMP5[2: 1]		
-		RW			RW			RW			RW			RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SMP5[0]	SMP4[2: 0]			SMP3[2: 0]			SMP2[2: 0]			SMP1[2: 0]			SMP0[2: 0]			
RW	RW			RW			RW			RW			RW			

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29: 0	SMPx[2: 0]	RW	0	选择通道 x 的采样时间 通过软件设置这些位用于独立地选择每个通道的采样时间。 在采样周期中通道选择位必须保持不变。 000: 3.5周期      100: 28.5周期 001: 5.5周期      101: 41.5周期 010: 7.5周期      110: 134.5周期 011: 13.5周期     111: 239.5周期

### 16.11.7. ADC 注入通道数据偏移寄存器 x (ADC\_JOFRx) (x=1..4)

Address offset: 0x18-0x24

Reset value: 0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	JOFFSETx[11: 0]	RW	0	注入通道第 x 次转换数据偏移 软件配置这些位的值。当转换注入通道时，这些位定义了用于从原始转换数据中减去的数值。最终转换结果可以在 ADC_JDRx 寄存器中读出。

### 16.11.8. ADC 看门狗高阈值寄存器 (ADC\_HTR)

Address offset: 0x28

Reset value: 0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	HTR[11: 0]	RW	0xFFFF	模拟看门狗高阈值 软件配置这些位的值。这些位定义了模拟看门狗的阈值高限。

### 16.11.9. ADC 看门狗低阈值寄存器 (ADC\_LTR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11: 0	LTR[11: 0]	RW	0x000	模拟看门狗低阈值 软件配置这些位的值。这些位定义了模拟看门狗的阈值下限。

### 16.11.10. ADC 规则序列寄存器 1 (ADC\_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								L[3: 0]				SQ16[4: 1]			
-								RW				RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16[0]		SQ15[4: 0]				SQ14[4: 0]				SQ13[4: 0]					
RW		RW				RW				RW					

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23: 20	L[3: 0]	RW	0	规则通道序列长度 软件配置这些位的值。这些位定义了规则通道转换序列中通道数目。 0000: 1个转换 0001: 2个转换 ..... 1111: 16个转换
19: 15	SQ16[4: 0]	RW	0	软件配置这些位的值。规则序列中的第16个转换, 这些位定义了转换序列中的第16个转换通道的编号 (0~23)。
14: 10	SQ15[4: 0]	RW	0	软件配置这些位的值。规则序列中的第15个转换, 这些位定义了转换序列中的第15个转换通道的编号 (0~23)。
9: 5	SQ14[4: 0]	RW	0	软件配置这些位的值。规则序列中的第14个转换, 这些位定义了转换序列中的第14个转换通道的编号 (0~23)。
4: 0	SQ13[4: 0]	RW	0	软件配置这些位的值。规则序列中的第13个转换, 这些位定义了转换序列中的第13个转换通道的编号 (0~23)。

### 16.11.11. ADC 规则序列寄存器 2 (ADC\_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ12[4: 0]					SQ11[4: 0]					SQ10[4: 1]			
-		RW					RW					RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]		SQ9[4: 0]					SQ8[4: 0]					SQ7[4: 0]			
RW		RW					RW					RW			

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29: 25	SQ12[4: 0]	RW	0	软件配置这些位的值。规则序列中的第12个转换，这些位定义了转换序列中的第12个转换通道的编号 (0~23)。
24: 20	SQ11[4: 0]	RW	0	软件配置这些位的值。规则序列中的第11个转换，这些位定义了转换序列中的第11个转换通道的编号 (0~23)。
19: 15	SQ10[4: 0]	RW	0	软件配置这些位的值。规则序列中的第10个转换，这些位定义了转换序列中的第10个转换通道的编号 (0~23)。
14: 10	SQ9[4: 0]	RW	0	软件配置这些位的值。规则序列中的第9个转换，这些位定义了转换序列中的第9个转换通道的编号 (0~23)。
9: 5	SQ8[4: 0]	RW	0	软件配置这些位的值。规则序列中的第8个转换，这些位定义了转换序列中的第8个转换通道的编号 (0~23)。
4: 0	SQ7[4: 0]	RW	0	软件配置这些位的值。规则序列中的第7个转换，这些位定义了转换序列中的第7个转换通道的编号 (0~23)。

### 16.11.12. ADC 规则序列寄存器 3 (ADC\_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ6[4: 0]					SQ5[4: 0]					SQ4[4: 1]			
-		RW					RW					RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[0]		SQ3[4: 0]					SQ2[4: 0]					SQ1[4: 0]			
RW		RW					RW					RW			

Bit	Name	R/W	Reset Value	Function
31: 30	保留	-	-	保留
29: 25	SQ6[4: 0]	RW	0	软件配置这些位的值。规则序列中的第6个转换，这些位定义了转换序列中的第6个转换通道的编号（0~23）。
24: 20	SQ5[4: 0]	RW	0	软件配置这些位的值。规则序列中的第5个转换，这些位定义了转换序列中的第5个转换通道的编号（0~23）。
19: 15	SQ4[4: 0]	RW	0	软件配置这些位的值。规则序列中的第4个转换，这些位定义了转换序列中的第4个转换通道的编号（0~23）。
14: 10	SQ3[4: 0]	RW	0	软件配置这些位的值。规则序列中的第3个转换，这些位定义了转换序列中的第3个转换通道的编号（0~23）。
9: 5	SQ2[4: 0]	RW	0	软件配置这些位的值。规则序列中的第2个转换，这些位定义了转换序列中的第2个转换通道的编号（0~23）。
4: 0	SQ1[4: 0]	RW	0	软件配置这些位的值。规则序列中的第1个转换，这些位定义了转换序列中的第1个转换通道的编号（0~23）。

### 16.11.13. ADC 注入序列寄存器 (ADC\_JSQR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											JL[1: 0]		JSQ4[4: 1]		
-											RW		RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]		JSQ3[4: 0]				JSQ2[4: 0]				JSQ1[4: 0]					
RW		RW				RW				RW					

Bit	Name	R/W	Reset Value	Function
31: 22	保留	-	-	保留
21: 20	JL[1: 0]	RW	0	注入通道序列长度 软件配置这些位的值。这些位定义了注入通道转换序列中通道数目。 00: 1个转换 01: 2个转换 10: 3个转换 11: 4个转换
19: 15	JSQ4[4: 0]	RW	0	注入序列中的第4个转换 软件配置这些位的值。这些位定义了转换序列中的第4个转换通道的编号（0~23）。 注：不同于规则转换序列，如果 JL[1: 0]的长度小于4，则转换的序列顺序是从（4-JL）开始。

				例如：ADC_JSQR[21: 0] = 10 00011 00011 00111 00010, 意味着扫描转换将按下列通道顺序转换：7、3、3, 而不是2、7、3
14: 10	JSQ3[4: 0]	RW	0	软件配置这些位的值。注入序列中的第3个转换
9: 5	JSQ2[4: 0]	RW	0	软件配置这些位的值。注入序列中的第2个转换
4: 0	JSQ1[4: 0]	RW	0	软件配置这些位的值。注入序列中的第1个转换

### 16.11.14. ADC 注入数据寄存器 x (ADC\_JDRx) (x= 1..4)

Address offset: 0x40-4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	JDATA[15: 0]	R	0	注入通道转换结果 软件可读这些位的值。这些位为只读, 包含了注入通道的转换结果。数据是左对齐或右对齐

### 16.11.15. ADC 规则数据寄存器 (ADC\_DR)

Address offset: 0x50

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	DATA[15: 0]	R	0	规则通道转换结果 软件可读这些位的值。这些位为只读, 包含了规则通道的转换结果。数据是左或右对齐



### 16.11.16. ADC 校准配置和状态寄存器 (ADC\_CCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON.	CAPSUC	OFFSUC	Res												
R	RC_W1	RC_W1	-												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALSET	CALBYP	CALSMPL[1: 0]		CALSEL	Res										
R_W1	R_W1	RW		RW	-										

Bit	Name	R/W	Reset Value	Function
31	CALON	R	0	校准 flag, 标志 ADC 校准正在进行。 1: ADC 校准正在进行 0: ADC 校准已结束或未启动 ADC 校准
30	CAPSUC	RC_W1	0	电容校准状态位。 表示 ADC 电容校准是否成功。硬件置1; 软件写1置0; CALON=0, CALSEL=0,CALSUC=1: 无效状态 CALON=0, CALSEL=0, CALSUC=0: 未进行 CAPs 校准 CALON=0, CALSEL=1, CALSUC =1: ADC CAPs 校准成功 CALON=0, CALSEL=1, CALSUC =0: ADC CAPs 校准失败
29	OFFSUC	RC_W1	0	Offset 校准状态位。 表示 ADC offset 校准是否成功。硬件置1; 软件写1置0; CALON=0, CALSEL=0,OFFSUC=0: ADC OFFSET 校准失败 CALON=0, CALSEL=0, OFFSUC=1: ADC OFFSET 校准成功 CALON=0, CALSEL=1,OFFSUC=1: ADC OFFSET 校准成功 CALON=0, CALSEL=1, OFFSUC=0: ADC OFFSET 校准失败
28: 16	保留	-	-	保留
15	CALSET	RC_W1	0	

Bit	Name	R/W	Reset Value	Function
				校准因子选择。当 CAL 为0时，软件写1置1。 CAL 有效或注入/规则通道 SWSTART， JWSTART 有效时，硬件置0。 1: 设置 CAL_CXIN 数据作为最终的校准数据 0: 关闭 CAL_CXIN 到 CAL_CXOUT 的通路，选择校准电路内部产生的结果。
14	CALBYP	R_W1	0	校准因子旁路。当 CAL 为0时，软件写1置1。 CAL 有效或注入/规则通道 SWSTART， JWSTART 有效时，硬件置0。 1: 校准结果为复位值 0: 校准结果为自校准结果或者校准因子输入值
13: 12	CALSMP[1: 0]	RW	0	校准 sample time seletion 根据以下信息，配置校准采样阶段的时钟周期个数： 00: 1个 ADC 时钟周期 01: 2 个 ADC 时钟周期 10: 4 个 ADC 时钟周期 11: 8 个 ADC 时钟周期 校准时配置 SMP 的周期越长，校准结果更精确，但该配置会带来校准周期延长的问题
11	CALSEL	RW	0	校准内容选择位，用于选择需要校准的内容 1: 校准 OFFSET 以及线性度 0: 只校准 OFFSET
10: 0	保留	-	-	保留

## 17. 液晶控制器 (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 (COM) 和 40 个区段端子 (SEG), 用以驱动 160 (4x40) 或 288 (8x36) 个 LCD 像素。端子的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。

#### 词汇表

- 液晶 (LCD) : 无源显示面板, 带有直接引向区段的端子。
- 公用 (COM) : 连接到多个区段的电气连接端子。
- 偏置 (BIAS) : 驱动 LCD 时使用等级, 定义为  $1/$  (驱动 LCD 显示的电压等级-1)。
- 区段 (SEG) : 最小可视单元 ( LCD 显示器上的最小组成元素, 线条或点)。
- 占空比 (DUTY) :  $1/$ 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。

### 17.2. LCD 主要特性

- 高度灵活的帧速率控制。
- 支持静态、 $1/2$ 、 $1/3$ 、 $1/4$ 、 $1/6$  和  $1/8$  占空比。
- 支持静态、 $1/2$ 、 $1/3$  偏置电压。
- 多达 16 个寄存器的 LCD 数据 RAM。
- 可通过软件配置 LCD 的对比度。
- 2 种驱动波形生成方式
  - 内部电阻分压、外部电阻分压
  - 可通过软件配置内部电阻分压方式的功耗, 从而匹配 LCD 面板所需的电容电荷
- 支持低功耗模式: LCD 控制器可在 Run、Sleep、Stop 模式下进行显示。
- 可配置帧中断。
- 支持 LCD 闪烁功能且可配置多种闪烁频率
- 未使用的 LCD 区段和公共引脚可配置为数字或模拟功能。

### 17.3. LCD 框图

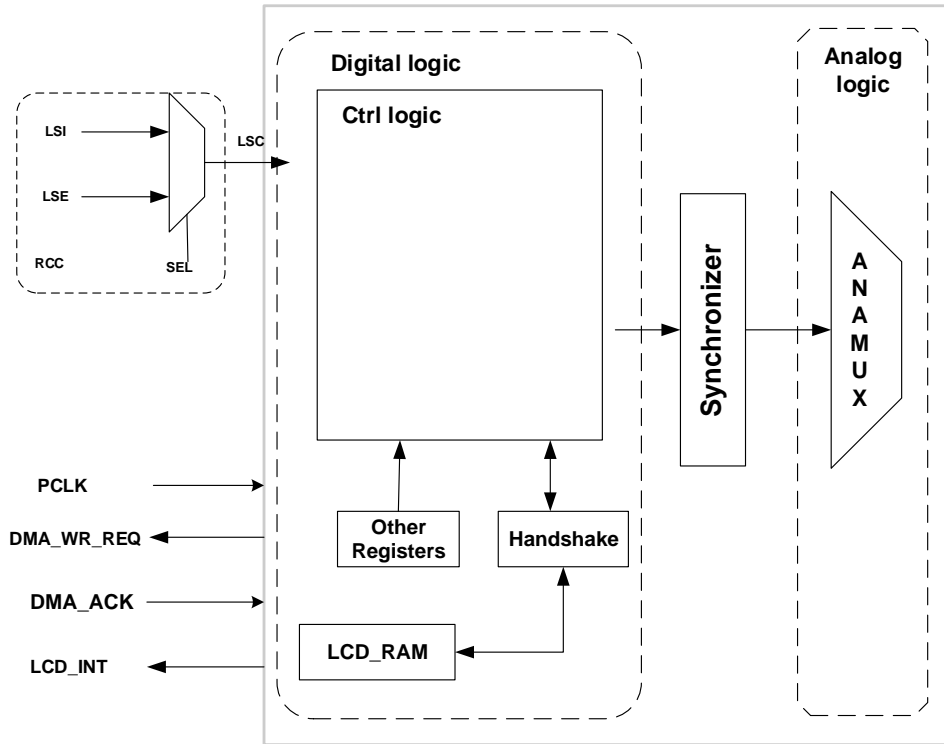


图 17-1 LCD 框图

### 17.4. LCD 时钟

LCD 时钟源可选择为 LSI 或 LSE，可通过 RCC\_BDCR 寄存器的 LSCOSEL 及 LSCOEN 确定输入时钟。

### 17.5. LCD 驱动波形

LCD 支持 5 种占空比 (Duty) 的驱动波形：静态、1/2、1/3、1/4、1/6 和 1/8，由 LCD\_CR0.DUTY 进行设置。

LCD 支持 2 种偏置 (BIAS) 的驱动波形：静态、1/2、1/3，由 LCD\_CR0.BIAS 进行设置。

建议的组合方式如下表所示：

表 17-1 LCD 支持的驱动波形

-	1/2 DUTY	1/3 DUTY	1/4 DUTY	1/6 DUTY	1/8 DUTY
1/2 BIAS	✓	✓	不推荐	不推荐	不推荐
1/3 BIAS	不推荐	不推荐	✓	✓	✓

各模式下的驱动波形如下方所示:

### 17.5.1. 静态驱动波形

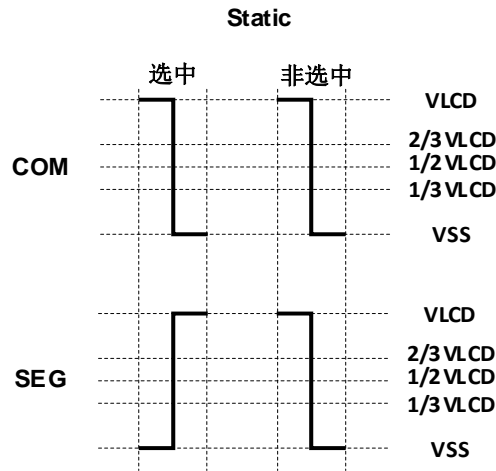


图 17-2 静态驱动波形

### 17.5.2. 1/2DUTY 1/2BIAS 驱动波形

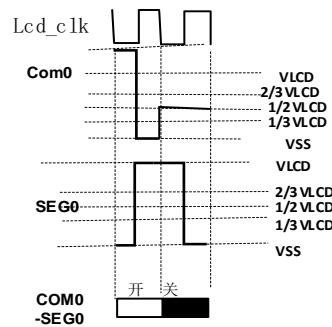


图 17-3 1/2DUTY 1/2BIAS 驱动波形

### 17.5.3. 1/8DUTY 1/3BIAS 驱动波形

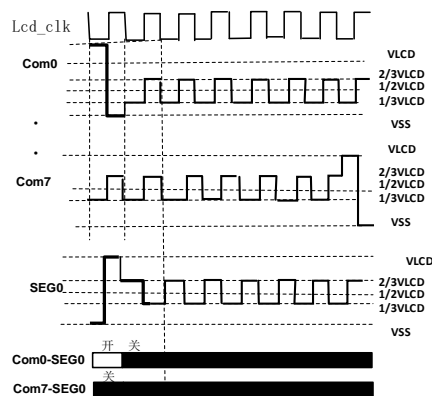


图 17-4 1/8DUTY 1/3 BIAS 驱动波形

## 17.6. LCD BIAS 产生电路

LCD 的 BIAS 电压具有 2 种来源：内部电阻分压、外部电阻分压。当选择内部电阻分压时，芯片会自动切换内部的电路以产生符合 BIAS 和 DUTY 的电压。当选择外部电阻分压时，需要用户在芯片的外围引脚搭建相关电路。

### 17.6.1. 内部电阻模式

内部电阻模式  $V_{LCDH}, V_{LCD1} \sim V_{LCD3}$  可以作为 LCD SEG 输出或者 IO 端口使用。

内部电阻模式，LCD 的驱动电压由 CR0.CONTRAST 控制，如下表所示：

表 17-2 内部电阻模式

CR0.CONTRAST	$V_{LCD}$ (1/3 BIAS)	$V_{LCD}$ (1/2 BIAS)
0	$1.00 * V_{CCA}$	$1.00 * V_{CCA}$
1	$0.95 * V_{CCA}$	$0.92 * V_{CCA}$
2	$0.9 * V_{CCA}$	$0.86 * V_{CCA}$
3	$0.86 * V_{CCA}$	$0.8 * V_{CCA}$
4	$0.82 * V_{CCA}$	$0.75 * V_{CCA}$
5	$0.78 * V_{CCA}$	$0.71 * V_{CCA}$
6	$0.75 * V_{CCA}$	$0.67 * V_{CCA}$
7	$0.72 * V_{CCA}$	$0.63 * V_{CCA}$
8	$0.69 * V_{CCA}$	$0.60 * V_{CCA}$
9	$0.67 * V_{CCA}$	$0.57 * V_{CCA}$
10	$0.64 * V_{CCA}$	$0.55 * V_{CCA}$
11	$0.62 * V_{CCA}$	$0.52 * V_{CCA}$
12	$0.60 * V_{CCA}$	$0.50 * V_{CCA}$
13	$0.58 * V_{CCA}$	$0.48 * V_{CCA}$
14	$0.56 * V_{CCA}$	$0.46 * V_{CCA}$
15	$0.55 * V_{CCA}$	$0.44 * V_{CCA}$

## 17.6.2. 外部电阻模式

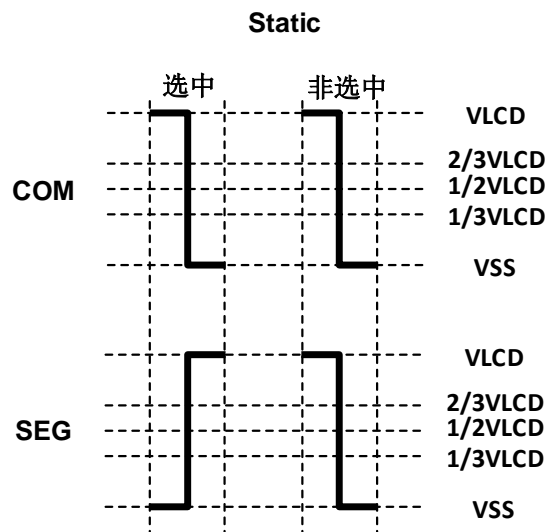


图 17-5 外部电阻模式

注意:

1. Rx 为可调电阻，用于调节 LCD 显示对比度。
2. 根据使用 LCD 屏幕选择合适的电阻 R。

## 17.7. DMA

LCD 支持软件和硬件触发 DMA 数据传输，可以将需要显示的内容从存储内容中自动搬到 LCD 显示 RAM 中。硬件触发使用的是帧中断信号。

DMA 数据传送配置流程:

1. 使能 DMA
2. 选择 LCD DMA
3. 设置传输类型、传输长度、传输方式
4. 设置源起始地址，目标起始地址
5. 设置源地址、目标地址的递增方式
6. 根据需要使能 DMA 中断
7. 使能 LCD DMA 触发

## 17.8. 中断

当 LCD 设置有效时，LCD 中断可以配置为帧数产生中断。

## 17.9. LCD 显示模式

LCD 支持两种显示模式。一种以 COM 为显示单元，同一个 SEG 的所有 COM 段在同一字节中（模式 0）。另外一种为同一个 COM 的不同 SEG 在同一个字节中（模式 1）。

根据 LCD 面板选择合适的显示方式可以简化程序操作。

### 17.9.1. LCD 显示模式 1 (MODE = 1)

#### ■ 1/8 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																								
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0																								
COM0																																		LCDRAM0																						
COM1																																			LCDRAM1																					
COM2																																			LCDRAM2																					
COM3																																			LCDRAM3																					
COM4																																			LCDRAM4																					
COM5																																			LCDRAM5																					
COM6																																			LCDRAM6																					
COM7																																			LCDRAM7																					

图 17-6 1/8 DUTY LCD 显示模式1



■ 1/6 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0			
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0			
COM0																																		LCDRAM0	
COM1																																		LCDRAM1	
COM2																																		LCDRAM2	
COM3																																		LCDRAM3	
COM4																																		LCDRAM4	
COM5																																		LCDRAM5	
																																		LCDRAM6	
																																		LCDRAM7	
																											SEG37	SEG36	SEG35	SEG34	SEG33	SEG32			
COM0																																		LCDRAM8	
COM1																																		LCDRAM9	
COM2																																		LCDRAMA	
COM3																																		LCDRAMB	
COM4																																		LCDRAMC	
COM5																																		LCDRAMD	
																																		LCDRAME	
																																		LCDRAMF	

图 17-7 1/6 DUTY LCD 显示模式1

■ 1/4 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0					
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0					
COM0																																		LCDRAM0			
COM1																																		LCDRAM1			
COM2																																		LCDRAM2			
COM3																																		LCDRAM3			
																																		LCDRAM4			
																																		LCDRAM5			
																																		LCDRAM6			
																																		LCDRAM7			
																											SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32			
COM0																																		LCDRAM8			
COM1																																		LCDRAM9			
COM2																																		LCDRAMA			
COM3																																		LCDRAMB			
																																		LCDRAMC			
																																		LCDRAMD			
																																		LCDRAME			
																																		LCDRAMF			

图 17-8 1/4 DUTY LCD 显示模式1

■ 1/3 DUTY 1/2 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0		
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0		
COM0																																		LCDRAM0
COM1																																		LCDRAM1
COM2																																		LCDRAM2
																																		LCDRAM3
																																		LCDRAM4
																																		LCDRAM5
																																		LCDRAM6
																																		LCDRAM7
																									SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32		
COM0																																		LCDRAM8
COM1																																		LCDRAM9
COM2																																		LCDRAMA
																																		LCDRAMB
																																		LCDRAMC
																																		LCDRAMD
																																		LCDRAME
																																		LCDRAMF

图 17-9 1/3 1/2 DUTY LCD 显示模式1

17.9.2. LCD 显示模式 0 (MODE = 0)

■ 1/8 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0			
COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0			
						SEG3									SEG2								SEG1									SEG0	LCDRAM0	
						SEG7									SEG6								SEG5									SEG4	LCDRAM1	
						SEG11									SEG10								SEG9									SEG8	LCDRAM2	
						SEG15									SEG14								SEG13									SEG12	LCDRAM3	
						SEG19									SEG18								SEG17									SEG16	LCDRAM4	
						SEG23									SEG22								SEG21									SEG20	LCDRAM5	
						SEG27									SEG26								SEG25									SEG24	LCDRAM6	
						SEG31									SEG30								SEG29									SEG28	LCDRAM7	
																																SEG32	LCDRAM8	
																																SEG33	LCDRAM9	
																																SEG34	LCDRAMA	
																																SEG35	LCDRAMB	
																																		LCDRAMC
																																		LCDRAMD
																																		LCDRAME
																																		LCDRAMF

图 17-10 1/8 DUTY LCD 显示模式0

■ 1/6 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0		
		COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0		
							SEG3								SEG2									SEG1							SEG0	LCDRAM0	
							SEG7								SEG6									SEG5							SEG4	LCDRAM1	
							SEG11								SEG10									SEG9							SEG8	LCDRAM2	
							SEG15								SEG14									SEG13							SEG12	LCDRAM3	
							SEG19								SEG18									SEG17							SEG16	LCDRAM4	
							SEG23								SEG22									SEG21							SEG20	LCDRAM5	
							SEG27								SEG26									SEG25							SEG24	LCDRAM6	
							SEG31								SEG30									SEG29							SEG28	LCDRAM7	
																															SEG32	LCDRAM8	
																															SEG33	LCDRAM9	
																															SEG34	LCDRAMA	
																															SEG35	LCDRAMB	
																															SEG36	LCDRAMC	
																															SEG37	LCDRAMD	
																																	LCDRAME
																																	LCDRAMF

图 17-11 1/6 DUTY LCD 显示模式0

■ 1/4 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
				COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0	
							SEG3								SEG2									SEG1							SEG0	LCDRAM0
							SEG7								SEG6									SEG5							SEG4	LCDRAM1
							SEG11								SEG10									SEG9							SEG8	LCDRAM2
							SEG15								SEG14									SEG13							SEG12	LCDRAM3
							SEG19								SEG18									SEG17							SEG16	LCDRAM4
							SEG23								SEG22									SEG21							SEG20	LCDRAM5
							SEG27								SEG26									SEG25							SEG24	LCDRAM6
							SEG31								SEG30									SEG29							SEG28	LCDRAM7
																															SEG32	LCDRAM8
																															SEG33	LCDRAM9
																															SEG34	LCDRAMA
																															SEG35	LCDRAMB
																															SEG36	LCDRAMC
																															SEG37	LCDRAMD
																															SEG38	LCDRAME
																															SEG39	LCDRAMF

图 17-12 1/4 DUTY LCD 显示模式0

■ 1/3 DUTY 1/2 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																						
					COM2	COM1	COM0						COM2	COM1	COM0						COM2	COM1	COM0							COM2	COM1	COM0																					
							SEG3								SEG2																SEG0	LCDRAM0																					
							SEG7								SEG6																SEG4	LCDRAM1																					
							SEG11								SEG10																SEG8	LCDRAM2																					
							SEG15								SEG14																SEG12	LCDRAM3																					
							SEG19								SEG18																SEG16	LCDRAM4																					
							SEG23								SEG22																SEG20	LCDRAM5																					
							SEG27								SEG26																SEG24	LCDRAM6																					
							SEG31								SEG30																SEG28	LCDRAM7																					
																																																			SEG32	LCDRAM8	
																																																				SEG33	LCDRAM9
																																																				SEG34	LCDRAMA
																																																				SEG35	LCDRAMB
																																																				SEG36	LCDRAMC
																																																				SEG37	LCDRAMD
																																																				SEG38	LCDRAME
																																																				SEG39	LCDRAMF

图 17-13 1/3 1/2 DUTY LCD 显示模式0

## 17.10. LCD 寄存器

### 17.10.1. 配置寄存器 0 (LCD\_CR0)

Address offset: 0x00

Reset value: 0x0000 00C2

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
CONTRAST[3: 0]				BSEL[2: 0]			DUTY[2: 0]			BIAS	Res	Res	LCDCLK[1: 0]		EN
RW				RW			RW			RW	-	-	RW		RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 12	CONTRAST[3: 0]	RW	0	LCD 对比度调整 注: 仅当 BIAS 电压来源选择内部电阻分压时有效。 值越大, LCD 波形的幅度越小。 0x0时, LCD 波形幅度最大, 对比度最大; ..... 0xF 时, LCD 波形幅度最小, 对比度最小;
11: 9	BSEL[2: 0]	RW	0	BIAS 电压来源选择 111: Res 110: 内部电阻分压, 大功耗模式 101: Res 100: 内部电阻分压, 小功耗模式 011: Res 010: 内部电阻分压, 中功耗模式 001: Res 000: 外部电阻模式, 需要外部电路配合 注: 具体功耗值见 datasheet
8: 6	DUTY[2: 0]	RW	3'b011	LCD DUTY 配置 000: 静态 001: 1/2 DUTY 010: 1/3 DUTY 011: 1/4 DUTY 100: Res

Bit	Name	R/W	Reset Value	Function
				101: 1/6 DUTY 110: Res 111: 1/8 DUTY
5	BIAS	RW	0	0: 1/3 偏压 (初始值) 1: 1/2 偏压
4: 3	保留	-	-	保留
2: 1	LCDCLK[1: 0]	RW	2'b01	LCD 扫描频率选择 00: 64 Hz 01: 128 Hz 10: 256 Hz 11: 512 Hz 注: LCD 帧频率 = LCD 扫描频率×DUTY
0	EN	RW	0	LCD 使能控制 1: 使能 0: 禁止

### 17.10.2. 配置寄存器 1 (LCD\_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11	INTF	R	0	LCD 中断标志 1: 中断 0: 无中断
10	DMAEN	RW	0	DMA 硬件触发使能 1: 使能 LCD 中断触发 DMA 0: 禁止 LCD 中断触发 DMA
9	IE	RW	0	中断使能 1: 使能 0: 禁止
8	MODE	RW	0	LCD RAM 显示模式选择

Bit	Name	R/W	Reset Value	Function
				0: 模式0 1: 模式1
7	保留	-	-	保留
6	BLINKEN	RW	0	LCD 闪屏配置 1: 使能 0: 禁止
5: 0	BLINKCNT[5:0]	RW	0	闪屏频率与 LCD 中断间隔设置 注: LCD 闪烁频率 = LCD 帧频率 / (BLINKCNT+1) LCD 中断间隔 = (BLINKCNT+1) * (1/LCD 帧频率)

### 17.10.3. 中断清除寄存器 (LCD\_INTCLR)

Address offset: 0x08

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	INTF_CLR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-					R1W0	-									

Bit	Name	R/W	Reset Value	Function
31: 11	保留	-	-	保留
10	INTF_CLR	RC_W0	1	中断标志清除, 写0清除, 写1无效
9: 0	保留	-	-	保留

### 17.10.4. 输出配置寄存器 (LCD\_POEN0)

Address offset: 0x0C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
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
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	Sx	RW	0xFFFFFFFF	SEGx 输出控制位 0: SEG 输出使能

				1: SEG 输出关闭, 可以使用其他功能, 如 IO, 模拟输入输出
--	--	--	--	-------------------------------------

### 17.10.5. 输出配置寄存器 1 (LCD\_POEN1)

Address offset: 0x10

Reset value: 0x1FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	-4	3	2	1	0
Res		MUX	C3	C2	C1	C0	S36_C	S37_C	S38_C	S39_C	S35	S34	S33	S32	
							7	6	5	C4					
-		RW													

Bit	Name	R/W	Reset Value	Function
31: 13	保留	-	-	保留
12	MUX	RW	1	SEG32~SEG35端口功能选择, 详细内容见下表
11: 8	Cx	RW	0xF	COMx 输出控制位 (COM0-COM3) 0: COM 输出使能 1: COM 输出关闭, 可以使用其他功能, 如 IO, 模拟输入输出
7: 4	SxCy[3: 0]	RW	0xF	SEGx/COMy 输出控制位 0: SEG/COM 输出使能 1: SEG/COM 输出关闭, 可以使用其他功能, 如 IO, 模拟输入输出 SEG COM 引脚功能选择由 CR0.DUTY 决定
3: 0	S[3: 0]	RW	0xF	SEGx 输出控制位 0: SEG 输出使能 1: SEG 输出关闭, 可以使用其他功能, 如 IO, 模拟输入输出

表 17-3 COM/SEG 复用选择配置

V <sub>LCDx</sub> SEGxPAD		寄存器如何配置				
		MUX	S<35:32>	BSEL		
V <sub>LCDx</sub> SEGxPAD 选择 GPIO, LCD disable (LCD_ON =0) 时	V <sub>LCDH</sub> SEG35=IO	1	1111	X	X	X
	V <sub>LCDH</sub> SEG34=IO					
	V <sub>LCDH</sub> SEG33=IO					
	V <sub>LCDH</sub> SEG32=IO					
	V <sub>LCDH</sub> SEG35=IO	1	1111	1	1	0



V <sub>LCDx</sub> SEGXPAD 选择 IO, LCD enable (LCD_ON =1) 时, 这时只能选择内部电阻工作模式	小电阻 (大电流) 模式	V <sub>LCDH</sub> SEG34=IO						
		V <sub>LCDH</sub> SEG33=IO						
		V <sub>LCDH</sub> SEG32=IO						
	中电阻 (中电流) 模式	V <sub>LCDH</sub> SEG35=IO	1	1111	0	1	0	
		V <sub>LCDH</sub> SEG34=IO						
		V <sub>LCDH</sub> SEG33=IO						
		V <sub>LCDH</sub> SEG32=IO						
	大电阻 (小电流) 模式	V <sub>LCDH</sub> SEG35=IO	1	1111	1	0	0	
		V <sub>LCDH</sub> SEG34=IO						
		V <sub>LCDH</sub> SEG33=IO						
		V <sub>LCDH</sub> SEG32=IO						
	选择内部电阻工作模式	小电阻 (大电流) 模式	V <sub>LCDH</sub> SEG35=SEG35/IO	0	XXXX	1	1	0
V <sub>LCDH</sub> SEG34=SEG34/IO								
V <sub>LCDH</sub> SEG33=SEG33/IO								
V <sub>LCDH</sub> SEG32=SEG32/IO								
中电阻 (中电流) 模式		V <sub>LCDH</sub> SEG35=SEG35/IO	0	XXXX	0	1	0	
		V <sub>LCDH</sub> SEG34=SEG34/IO						
		V <sub>LCDH</sub> SEG33=SEG33/IO						
		V <sub>LCDH</sub> SEG32=SEG32/IO						
大电阻 (小电流) 模式		V <sub>LCDH</sub> SEG35=SEG35/IO	0	XXXX	1	0	0	
		V <sub>LCDH</sub> SEG34=SEG34/IO						
		V <sub>LCDH</sub> SEG33=SEG33/IO						
		V <sub>LCDH</sub> SEG32=SEG32/IO						
选择外部电阻工作模式, 内部电阻短路	V <sub>LCDH</sub> SEG35=VOUT	0	1111	0	0	0		
	V <sub>LCDH</sub> SEG34=VLCD3							
	V <sub>LCDH</sub> SEG33=VLCD2							
	V <sub>LCDH</sub> SEG32=VLCD1							

### 17.10.6. LCD\_RAM0~7

Address offset: 0x14~0x30

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
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
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	Dx	RW	0	LCD 点输出, 显示参考 LCD 显示模式 0: 对应的 SEG COM 交叉点不亮 1: 对应的 SEG COM 交叉点亮

注: 在配置 LCD\_RAMx 时, 要保证在2个 LCD 时钟 (LSI 或者 LSE) 内完成。且两次写的间隔满足2个 LCD 时钟。

### 17.10.7. LCD\_RAM8~F

Address offset: 0x34~0x50

Reset value: 0x00

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	D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	Dx	RW	0	LCD 点输出, 显示参考 LCD 显示模式 0: 对应的 SEG COM 交叉点不亮; 1: 对应的 SEG COM 交叉点亮;

注: 在配置 LCD\_RAMx 时, 要保证在2个 LCD 时钟 (LSI 或者 LSE) 内完成。且两次写的间隔满足2个 LCD 时钟。

## 18. 比较器 (COMP)

### 18.1. 简介

芯片内集成 3 个通用比较器 (General Purpose Comparators) COMP, 分别是 COMP1, COMP2 和 COMP3。这三个模块可以作为单独的模块, 也可以与 Timer 组合在一起使用。

比较器可以被如下使用:

- 被模拟信号触发, 产生低功耗模式唤醒功能
- 模拟信号调节
- 当与来自 Timer 的 PWM 输出连接时, 构成逐周期 (Cycle by cycle) 的电流控制回路

### 18.2. COMP 主要特性

- 每个比较器有可配置的正或者负输入, 以实现灵活的电压选择:
  - 多路 I/O 引脚
  - 温度传感器的输出
  - 内部参考电压  $V_{REFBUF}$ 、 $V_{CCA}$ 、 $V_{REF1P2}$  通过分压提供的 64 个分数值 (1/64、2/64 ... 64/64)
  - $V_{REFINT}$
  - DAC 输出作为 INP 或者 INM 输入
  - OPA 输出作为 INP 输入
- 迟滞功能可配置
- 可编程的速度和功耗
- 输出可以被连接到 I/O 或者 Timer 的输入作为触发
  - OCREF\_CLR 事件 (逐周期的电流控制)
  - 刹车事件 (用于快速 PWM 关断)
- COMP1 和 COMP2 可以组合成窗口比较器
- 每个 COMP 具有中断产生能力, 用作芯片从低功耗模式 (Sleep 和 Stop 模式) 的唤醒 (通过 EXTI)

## 18.3. COMP 功能描述

### 18.3.1. COMP 框图

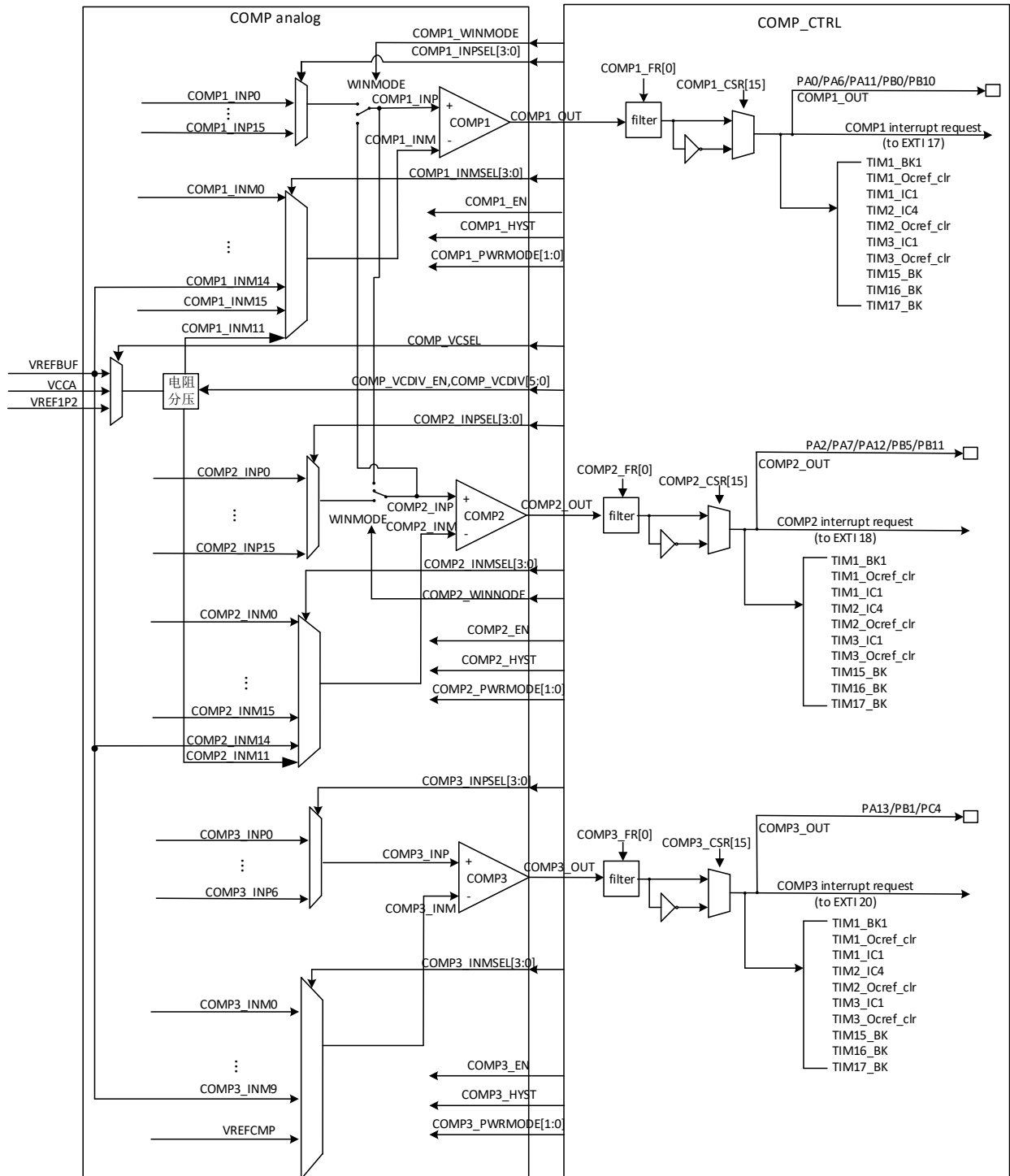


图 18-1 比较器架构框图

### 18.3.2. COMP 管脚和内部信号

用作比较器输入的 I/O，必须在 GPIO 寄存器中被配置成模拟模式。

比较器输出可以通过在 GPIO 的复用功能通道 (Alternate function) 连接到 I/O 引脚。

输出也可以在内部连接到各种 Timer 的输入，达到以下目的：

- 连接刹车输入时，可以紧急关断 PWM 信号
- 使用 OCREF\_CLR 输入进行逐周期电流控制
- 时序测量的输入捕获

### 18.3.3. COMP 复位和时钟

COMP 模块有两个时钟源：

- PCLK (APB clock)，用于给配置寄存器提供时钟
- COMP 时钟，用于模拟比较器输出后的电路（模拟输出的锁存电路、滤毛刺电路等）的时钟，可选择为 PCLK 或者 LSI 或者 LSE。当需要在 Stop 模式下工作时，选择 LSI 或者 LSE。

注意：

1. PCLK 和 COMP CLK 由 RCC\_APBENR2 控制 bit 同时使能，该使能关闭则 PCLK 和 COMP CLK 关闭，若使能则 PCLK 和 COMP CLK 同时开启。
2. 进入 Stop 模式前，建议先配置 COMP CLK 为 LSI 或 LSE，然后进入 Stop 模式。
3. COMP 模块的复位信号包含 APB 复位源和 COMP 模块软件复位源：
  - 1) PCLK 时钟域复位，用于 COMP 寄存器的复位
  - 2) COMP CLK 时钟域复位，用于模拟比较器输出后电路（模拟输出的锁存电路、滤毛刺电路等）的复位。

### 18.3.4. 窗口比较器

窗口比较器的作用是监测模拟电压是否在低和高阈值范围内。

可以使用两个比较器创建窗口比较器。被监测的模拟电压同时连接到两个比较器的非反相 (+端) 输入，高阈值和低阈值分别连接到两个比较器的反相输入端 (-端)。

通过使能 WINMODE 位，可以将两个比较器的非反相 (+输入端) 连接到一起，起到节省一个 I/O 引脚的作用。

注意：两个 COMP 的 WINMODE 模式不能同时使能。

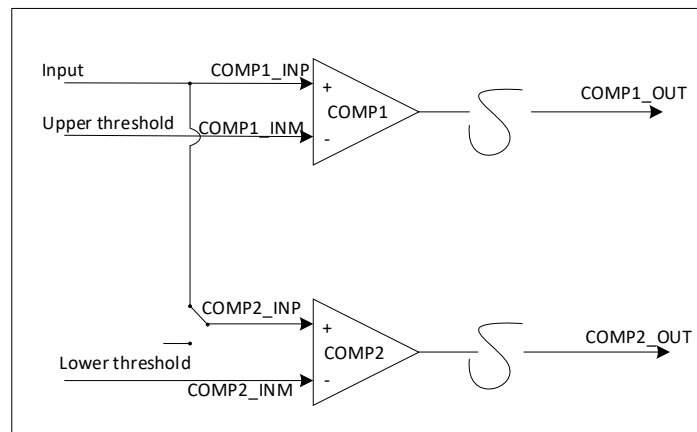


图 18-2 窗口比较器

### 18.3.5. 迟滞

为避免在噪声信号情况产生假的输出转换，比较器可以使能带有迟滞的功能（COMP1、COMP2 和 COMP3 有独立的迟滞功能使能信号 COMP1\_HYST,COMP2\_HYST 和 COMP3\_HYST）。

### 18.3.6. 功耗模式

比较器的功耗和传输延迟可以通过 COMPx\_CSR 寄存器的 PWRMODE[1: 0]位来选择不同模式，以实现在特定应用的最适合的 trade-off。可选的模式包括高速模式和中速模式两种，相对而言高速模式下的功耗更大，传输延迟也更小。注意，当进入 stop 之前，如果选择 PWR\_CR2 寄存器 LPR=1（即选择用 Low power regulator 供电），则需要首先设定 COMP 在中速模式（PWRMODE=01）。

### 18.3.7. 比较器滤波

可以通过设定 COMP\_FR 寄存器，使能 COMP 的输出滤波功能及相应的滤波宽度。注意该设定应在 COMP\_EN 使能前完成。

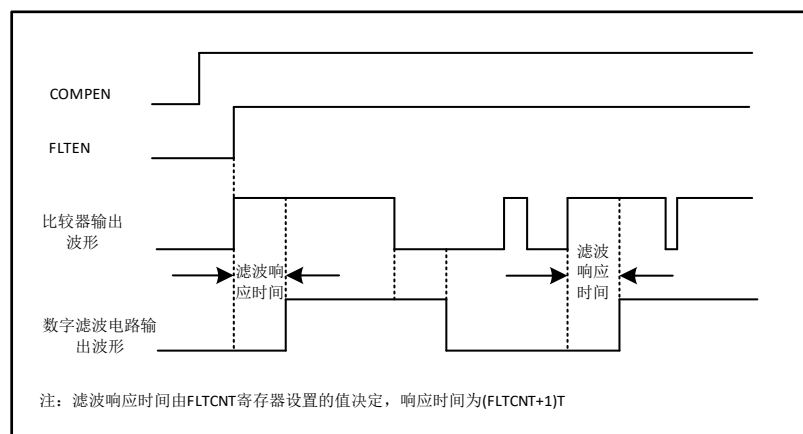


图 18-3 比较器滤波

### 18.3.8. COMP 中断

比较器输出在芯片内部连接到 EXTI 控制器。每个比较器有单独的 EXTI 线 (17 和 18 和 20)，并能够产生中断或者事件。相同的机制被用作从低功耗的唤醒。

## 18.4. COMP 寄存器

### 18.4.1. COMP1 控制和状态寄存器 (COMP1\_CSR)

Address offset: 0x0200\_0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_OUT	Res		COMP_VCSEL[1: 0]		COMP_VCDIV[5: 0]						PWRMODE[1: 0]	Res	COMP1_HYST	
-	R	-		RW		RW						RW	-	RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO-LARITY		Res		WINMODE	Res	INPSEL[3: 0]				INNSEL[3: 0]				Res	COMP1_EN
RW		-		RW	-	RW				RW				-	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30	COMP_OUT	R	0	COMP1输出状态 该位只读，它反映了 COMP1在经过极性选择的输出电平。
29: 28	保留	-	-	保留
27: 26	COMP_VCSEL	RW	0	COMP1,COMP2,COMP3参考电压 VREF 选择 00: 分压不使能 01: V <sub>CCA</sub> 10: V <sub>REFBUF</sub> 11: V <sub>REF1P2</sub>
25: 20	COMP_VCDIV[5: 0]	RW	100000	分压选择 00_0000: 1/64 V <sub>REFBUF</sub> 或 V <sub>CCA</sub> 或 V <sub>REF1P2</sub> 00_0001: 2/64 V <sub>REFBUF</sub> 或 V <sub>CCA</sub> 或 V <sub>REF1P2</sub> 00_0010: 3/64 V <sub>REFBUF</sub> 或 V <sub>CCA</sub> 或 V <sub>REF1P2</sub> ... 11_1110: 63/64 V <sub>REFBUF</sub> 或 V <sub>CCA</sub> 或 V <sub>REF1P2</sub> 11_1111: V <sub>REFBUF</sub> 或 V <sub>CCA</sub> 或 V <sub>REF1P2</sub>
19: 18	PWRMODE[1: 0]	RW	0	COMP1功耗模式选择 选择了功耗和由此而来的 COMP1的速度 00: 高速模式

Bit	Name	R/W	Reset Value	Function
				01: 中速模式 10: 保留 11: 保留
17	保留	-	-	保留
16	COMP1_HYST	RW	0	COMP1迟滞功能使能控制 0: 无迟滞 1: 迟滞电压约20 mV
15	POLARITY	RW	0	COMP1输出极性选择 软件可读可写 0: 不反相 1: 反相
14: 12	保留	-	-	保留
11	WINMODE	RW	0	COMP1非反相的输出选择 (窗口模式) 软件可读可写 0: 信号被 INPSEL[3: 0]选择 1: COMP2的 COMP2_INP 信号 注意两个 COMP 的 WINMODE 模式不能同时使能。
10	保留	-	-	保留
9: 6	INPSEL[3: 0]	RW	0	COMP1非反相输入的信号选择 0000: COMP1_INP0 from OPA1输出 0001: COMP1_INP1 from PC1 0010: COMP1_INP2 from PC2 0011: COMP1_INP3 from PC3 0100: COMP1_INP4 from PA0 0101: COMP1_INP5 from PA1 0110: COMP1_INP6 from PA2 0111: COMP1_INP7 from PA3 1000: COMP1_INP8 from PA4 1001: COMP1_INP9 from PA5 1010: COMP1_INP10 from PA6 1011: COMP1_INP11 from PA7 1100: COMP1_INP12 from PB4 1101: COMP1_INP13 from PB5 1110: COMP1_INP14 from PB6 1111: COMP1_INP15 from DAC1_VIN
5: 2	INNSEL[3: 0]	RW	0	COMP1反相输入的信号选择 0000: COMP1_INM0 from PA0 0001: COMP1_INM1 from PA1 0010: COMP1_INM2 from PA2



Bit	Name	R/W	Reset Value	Function
				0011: COMP1_INM3 from PA3 0100: COMP1_INM4 from PA4 0101: COMP1_INM5 from PA5 0110: COMP1_INM6 from PA6 0111: COMP1_INM7 from PA7 1000: COMP1_INM8 from PA6 1001: COMP1_INM9 from PC5 1010: COMP1_INM10 from DAC1_VIN 1011: COMP1_INM11 from V <sub>REFCOMP</sub> 1100: COMP1_INM12 from TS_VIN (温度传感器电压) 1101: COMP1_INM13 from V <sub>REFINT</sub> (需要使能 ADC 模块的 ADC_CR2.TSVREFE 寄存器) 1110: COMP1_INM14 from V <sub>REFBUF</sub> 1111: COMP1_INM15 from PC0
1	保留	-	-	保留
0	COMP1_EN	RW	0	COMP1使能位 软件可读可写 (如果没有被锁定) 0: 不使能 1: 使能

### 18.4.2. COMP1 滤波寄存器 (COMP1\_FR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN1
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT1[15:0]	RW	0	比较器1采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时, 结果一致输出。 采样计数周期=FLTCNT[15: 0]
15: 1	保留	-	-	保留
0	FLTEN1	RW	0	比较器1数字滤波功能配置 0: 禁止数字滤波功能

				1: 使能数字滤波功能 注: 该位必须在 COMP1_EN 为0时置位
--	--	--	--	--

### 18.4.3. COMP2 控制和状态寄存器 (COMP2\_CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_OUT	Res											PWR-MODE[1: 0]	Res	Res
-	R	-											RW	-	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res	Res	WINMODE	Res	INPSEL[3: 0]			INNSEL[3: 0]			COMP2_HYST	COMP2_EN			
RW	-	-	RW	-	RW										

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30	COMP_OUT	R	0	COMP2输出状态 该位只读, 它反映了 COMP2在经过极性选择的输出电平。
29: 20	保留	-	-	保留
19: 18	PWRMODE[1: 0]	RW	0	COMP2功耗模式选择 选择了功耗和由此而来的 COMP2的速度 00: 高速模式 01: 中速模式 10: 保留 11: 保留
17: 16	保留	-	-	保留
15	POLARITY	RW	0	COMP2输出极性选择 软件可读可写 0: 不反相 1: 反相
14: 12	保留	-	-	保留
11	WINMODE	RW	0	COMP2不反相的输出选择 软件可读可写 0: 信号被 INPSEL[1: 0]选择 1: COMP1的 COMP1_INP 信号

Bit	Name	R/W	Reset Value	Function
				注意两个 COMP 的 WINMODE 模式不能同时使能。
10	保留	-	-	保留
9: 6	INPSEL[3: 0]	RW	0	COMP2不反相输入的信号选择, 软件可读可写 0000: COMP2_INP0 from PA0 0001: COMP2_INP1 from PA1 0010: COMP2_INP2 from PA2 0011: COMP2_INP3 from PA3 0100: COMP2_INP4 from PA4 0101: COMP2_INP5 from PA5 0110: COMP2_INP6 from PB1 0111: COMP2_INP7 from PB2 1000: COMP2_INP8 from PB10 1001: COMP2_INP9 from OPA2输出 1010: COMP2_INP10 from PB13 1011: COMP2_INP11 from PB14 1100: COMP2_INP12 from PB4 1101: COMP2_INP13 from PB6 1110: COMP2_INP14 from PB7 1111: COMP2_INP15 from DAC2_VIN
5: 2	INNSEL[3: 0]	RW	0	COMP2不反相输入的信号选择 0000: COMP2_INM0 from PC0 0001: COMP2_INM1 from PC1 0010: COMP2_INM2 from PC2 0011: COMP2_INM3 from PC3 0100: COMP2_INM4 from PA0 0101: COMP2_INM5 from PA1 0110: COMP1_INM6 from PB0 0111: COMP1_INM7 from PB1 1000: COMP2_INM8 from PB2 1001: COMP2_INM9 from PB3 1010: COMP2_INM10 from DAC2_VIN 1011: COMP2_INM11 from V <sub>REFCOMP</sub> 1100: COMP2_INM12 from T <sub>S_VIN</sub> (温度传感器电压) 1101: COMP2_INM13 from V <sub>REFINT</sub> (需要使能 ADC 模块的 ADC_CR2.TSVREFE 寄存器) 1110: COMP2_INM14 from V <sub>REFBUF</sub> 1111: COMP2_INM15 from PB12
1	COMP2_HYST	RW	0	COMP2迟滞功能使能控制 0: 无迟滞

Bit	Name	R/W	Reset Value	Function
				1: 迟滞电压约20 mV
0	COMP2_EN	RW	0	COMP2使能位 软件可读可写 0: 不使能 1: 使能

#### 18.4.4. COMP2 滤波寄存器 (COMP2\_FR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT2[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN2
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT2[15: 0]	RW	0	比较器2采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时，结果一致输出。 采样计数周期=FLTCNT[15: 0]
15: 1	保留	-	-	保留
0	FLTEN2	RW	0	比较器2数字滤波功能配置 0: 禁止数字滤波功能 1: 使能数字滤波功能 注: 该位必须在 COMP2_EN 为0时置位

#### 18.4.5. COMP3 控制和状态寄存器 (COMP3\_CSR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_OUT	Res										PWR-MODE[1: 0]		Res	
-	R	-										RW		-	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res				INPSEL[3: 0]				INNSEL[3: 0]				COMP3_HYST	COMP3_EN	
RW	-				RW				RW				RW	RW	

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30	COMP_OUT	R	0	COMP3输出状态 该位只读，它反映了 COMP3在经过极性选择的输出电平。
29: 20	保留	-	-	保留
19: 18	PWRMODE[1: 0]	RW	0	COMP3功耗模式选择 选择了功耗和由此而来的 COMP3的速度 00: 高速模式 01: 中速模式 10: 保留 11: 保留
17: 16	保留	-	-	保留
15	POLARITY	RW	0	COMP3输出极性选择 软件可读可写 0: 不反相 1: 反相
14: 10	保留	-	-	保留
9: 6	INPSEL[3: 0]	RW	0	COMP3不反相输入的信号选择，软件可读可写 0000: COMP3_INP0 from PA5 0001: COMP3_INP1 from PB1 0010: COMP3_INP2 from PB11 0011: COMP3_INP3 from PB14 0100: COMP3_INP4 from OPA3输出 0101: COMP3_INP5 from DAC1_VIN 0110: COMP3_INP6 from DAC2_VIN 其它: 保留
5: 2	INNSEL[3: 0]	RW	0	COMP3不反相输入的信号选择 0000: COMP3_INM0 from PA5 0001: COMP3_INM1 from PB1 0010: COMP3_INM2 from PB11 0011: COMP3_INM3 from PB14 0100: COMP3_INM4 from PC7 0101: COMP3_INM5 from DAC1_VIN 0110: COMP3_INM6 from DAC2_VIN 0111: COMP3_INM7 from TS_VIN (温度传感器电压) 1000: COMP3_INM8 from V <sub>REFINT</sub> (需要使能 ADC 模块的 ADC_CR2.TSVREFE 寄存器) 1001: COMP3_INM9 from V <sub>REFBUF</sub>

Bit	Name	R/W	Reset Value	Function
				1010: COMP3_INM10 from V <sub>REFCOMP</sub> 其它: 保留
1	COMP3_HYST	RW	0	COMP3迟滞功能使能控制 0: 无迟滞 1: 迟滞电压约20 mV
0	COMP3_EN	RW	0	COMP3使能位 软件可读可写 0: 不使能 1: 使能

#### 18.4.6. COMP3 滤波寄存器 (COMP3\_FR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT3[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN3
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT3[15: 0]	RW	0	比较器3采样滤波计数器 采样时钟为 APB 或 LSI 或 LSE。滤波计数值可配置。采样次数达到滤波计数值时, 结果一致输出。 采样计数周期=FLTCNT[15: 0]
15: 1	保留	-	-	保留
0	FLTEN3	RW	0	比较器3数字滤波功能配置 0: 禁止数字滤波功能 1: 使能数字滤波功能 注: 该位必须在 COMP3_EN 为0时置位

## 19. 运算放大器 (OPA)

### 19.1. OPA 简介

OPA 模块适用于简易放大器应用。内部 3 个运放可以使用外部电阻进行级联。OPA 输入范围是  $0 \sim V_{CCA}$ , 输出范围是  $0.1 \sim V_{CCA} - 0.2 V$ 。

### 19.2. OPA 主要特性

- 3 个独立配置运放
- OPA 的输入范围是 0 到  $V_{CCA}$ , 输出范围是  $0.1 \sim V_{CCA} - 0.2 V$  可编程增益
- 可配置为以下模式
  - 通用运放模式

### 19.3. OPA 功能描述

3 个 OPA 可以通过使用外部元件组成放大器来放大小信号模拟输入信号, 输出为放大后的信号。

#### 19.3.1. OPA 框图

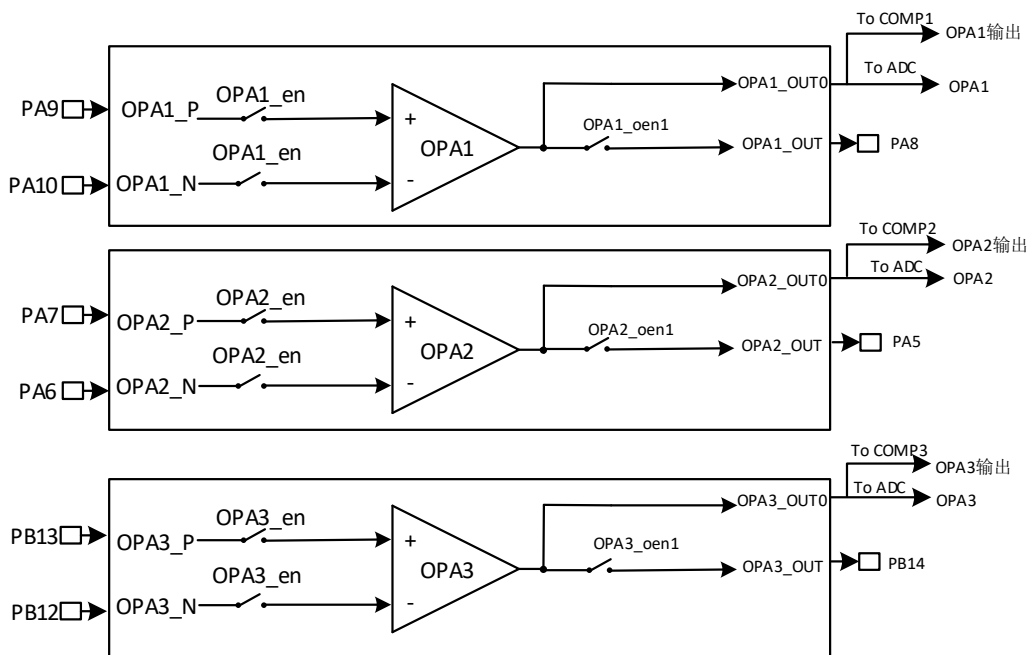


图 19-1 OPA 架构框图

## 19.4. OPA 寄存器

### 19.4.1. OPA 输出使能寄存器 (OPA\_CR0)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				OP3OEN1	Res				OP2OEN1	Res				OP1OEN1	Res
-				RW	-				RW	-				RW	-

Bit	Name	R/W	Reset Value	Function
31: 12	保留	-	-	保留
11	OPA3OEN1	RW	0	OPA3输出1使能
10: 7	保留	-	-	保留
6	OPA2OEN1	RW	0	OPA2输出1使能
5: 2	保留	-	-	保留
1	OPA1OEN1	RW	0	OPA1输出1使能
0	保留	-	-	保留

### 19.4.2. OPA 控制寄存器 (OPA\_CR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								EN3	EN2	EN1	Res				
-								RW	RW	RW	-				

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7	EN3	RW	0	OPA3使能
6	EN2	RW	0	OPA2使能
5	EN1	RW	0	OPA1使能
4: 0	保留	-	-	保留



## 20. 硬件除法器 (DIV)

### 20.1. DIV 简介

DIV (Divider) 是一个 32 位有符号/无符号整数硬件除法器。

### 20.2. DIV 主要特性

- 支持32位除法
- 当前除法未运行完毕时，寄存器中的数据不可改变
- 可配置有符号/无符号整数除法计算
- 32 位被除数，32 位除数
- 输出 32 位商和 32 位余数
- 除数为零警告标志位，除法运算结束标志位
- 8 个时钟周期完成一次除法运算
- 写除数寄存器触发除法运算电路
- 写完除数后，读商寄存器和余数等寄存器时，需要先等待计算完成标志 DIV\_END
- 除数为0时，商和余数结果为0

### 20.3. DIV 功能描述

#### 20.3.1. DIV 操作流程

1. 在 RCC 系统时钟控制器里打开硬件除法器的模块时钟使能位。
2. 配置寄存器 DIV\_SIGN 设置有符号/无符号除法运算。
3. 配置寄存器 DIV\_DEND 设置被除数。
4. 配置寄存器 DIV\_SOR 设置除数，除法运算开始。
5. 查询寄存器 DIV\_STAT 中的运算结束标志位 DIV\_END，DIV\_END 为 1 标志运算结束。读寄存器 DIV\_QUOT 得到商，读寄存器 HDIV\_REMD 得到余数。
6. 当除数为零时，除法运算立即结束，商和余数同时置为 0，同时除数为零警告标志位 DIV\_ZERO 被置起。

7. 在除法运算结束之前，读寄存器 DIV\_QUOT/DIV\_REMD 时，CPU 将读出上一次计算值

举例：计算一个无符号除法，被除数为 1917887483 (0x7250A3FB)，除数为 9597 (0x257D)

步骤一，配置寄存器 DIV\_SIGN 为 0，即无符号除法运算

步骤二，配置寄存器 DIV\_DEND 为 0x7250A3FB，即设置被除数

步骤三，配置寄存器 DIV\_SOR 为 0x257D，即设置除数，计算开始

步骤四，查询寄存器 DIV\_STAT 运算结束标志位 DIV\_END，DIV\_END 为 1 标志，运算结束。

读寄存器 DIV\_QUOT 得到商 199842 (0x30CA2)

读寄存器 DIV\_REMD 得到余数 3809 (0xEE1)

## 20.4. DIV 寄存器

### 20.4.1. DIV 被除数寄存器 (DIV\_DEND)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_DEND	RW	0	被除数

### 20.4.2. DIV 除数寄存器 (DIV\_SOR)

Address offset: 0x04

Reset value: 0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_SOR	RW	1	除数 (写此寄存器自动触发除法运算)

### 20.4.3. DIV 商寄存器 (DIV\_QUOT)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_QUOT	RW	0	存储除法器计算得到的商

#### 20.4.4. DIV 余数寄存器 (DIV\_REMD)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_REMD	RW	0	存储除法器计算得到的余数

#### 20.4.5. DIV 符号寄存器 (DIV\_SIGN)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	保留	-	-	保留
0	SIGN	RW	0	符号选择 0: 无符号除法运算 1: 有符号除法运算

#### 20.4.6. DIV 状态寄存器 (DIV\_STAT)

Address offset: 0x14

Reset value: 0x0000 0000

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

Res	DIV_ZERO	DIV_END
-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1	DIV_ZERO	R	0	除数为零警告标志位 0: 除数不为零 1: 除数为零
0	DIV_END	R	0	除法运算结束标志位 0: 运算进行中 1: 运算结束

## 21. 高级控制定时器（TIM1）

### 21.1. TIM1简介

高级定时器（TIM1）由 16 位被可编程预分频器驱动的自动重载计数器组成。它可以被用作各种场景，包括：输入信号（输入捕获）的脉冲长度测量，或者产生输出波形（输出比较、输出 PWM、带死区插入的互补 PWM）。

脉冲长度和波形周期可以使用定时器分频器和 RCC 时钟控制分频器，实现从微秒到毫秒的调制。高级定时器（TIM1）和通用（TIMx）定时器是完全独立的，不共享任何资源。他们可以同步操作。

### 21.2. TIM1主要特性

- 16-bit 向上、向下或者向上向下的自动重载计数器
- 16-bit 可以实时修改的可编程预分频器，允许对计数器的时钟频率进行 1 到 65536 的分频
- 多达 4 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 产生（边沿或者中央对齐模式）
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 重复计数器，在计数指定周期数后，才更新定时器的寄存器
- 刹车输入可以将定时器的输出信号置于用户可选的安全配置中
- 中断/DMA 产生在以下事件
  - 更新：计数器向上、向下溢出，计数器初始化（通过软件或者内外部触发）
  - 触发事件
  - 输入捕获
  - 输出比较
  - 刹车输入
- 支持增量式的（正交）编码器和为定位用的霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

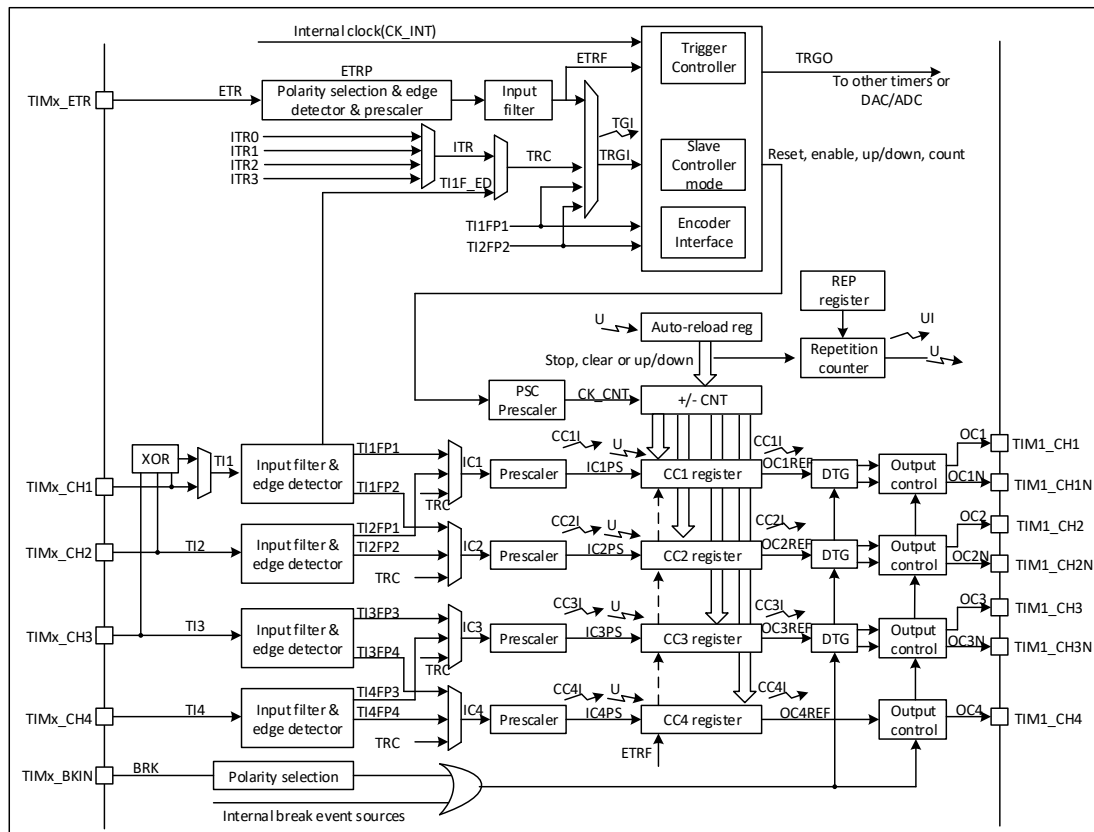


图 21-1 高级控制定时器架构框图

## 21.3. TIM1 功能描述

### 21.3.1. 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动重载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动重载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包括：

- 计数器寄存器（TIMx\_CNT）
- 预分频寄存器（TIMx\_PSC）
- 自动重载寄存器（TIMx\_ARR）
- 重复计数寄存器（TIMx\_RCR）

自动重载寄存器是预先装载的，写或者读自动重载寄存器将访问预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动重载预装载使能位（ARPE）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢出（向下计数器时的下溢条件）并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIMx\_CR1 寄存器中的计数器使能位（CEN）时，CK\_CNT 才有效。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

## 预分频器描述

预分频器可以将计数器的时钟按 1 到 65536 之间的任意值分频。它是一个基于（在 TIMx\_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下次更新事件到来时被采用。

图 21-2 和图 21-3 给出了在预分频器运行时，更改计数器参数的例子。

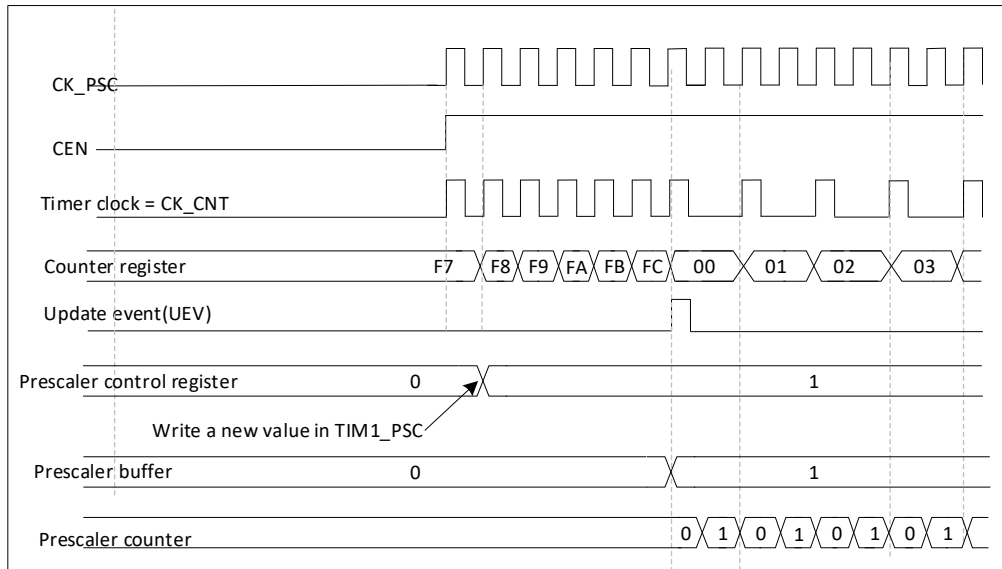


图 21-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

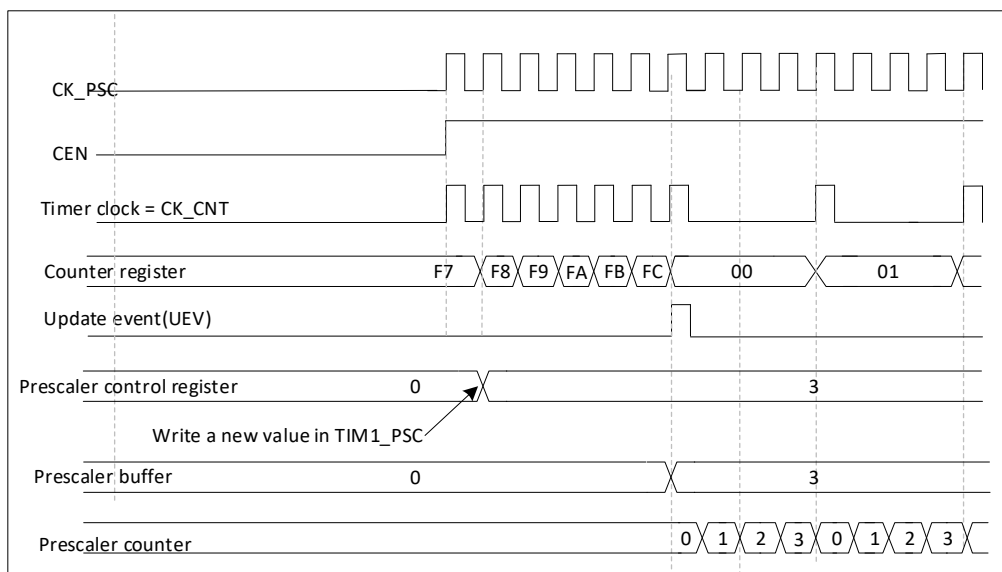


图 21-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 21.3.2. 计数器模式

#### 向上计数模式

向上计数模式，是从 0 到自动重载值的计数器，然后又从 0 重新开始计数，并产生一个计数的溢出事件。

如果重复计数器被使用，则在向上计数器重复了重复计数寄存器（TIMx\_RCR）中设定的次数后，将产生更新事件（UEV），否则每次计数器上溢都会产生更新事件。

在 TIMx\_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 URS 位）设置更新标志位（TIMx\_SR 寄存器中的 UIF 位）。

- 重复计数器被重新加载为 TIMx\_RCR 寄存器的内容。
- 自动重载影子寄存器被重新置入预装载寄存器的值（TIMx\_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIMx\_PSC 寄存器的内容）。

下图给出一些例子，当 TIMx\_ARR=0x36 时计数器在不同时钟频率下的动作。

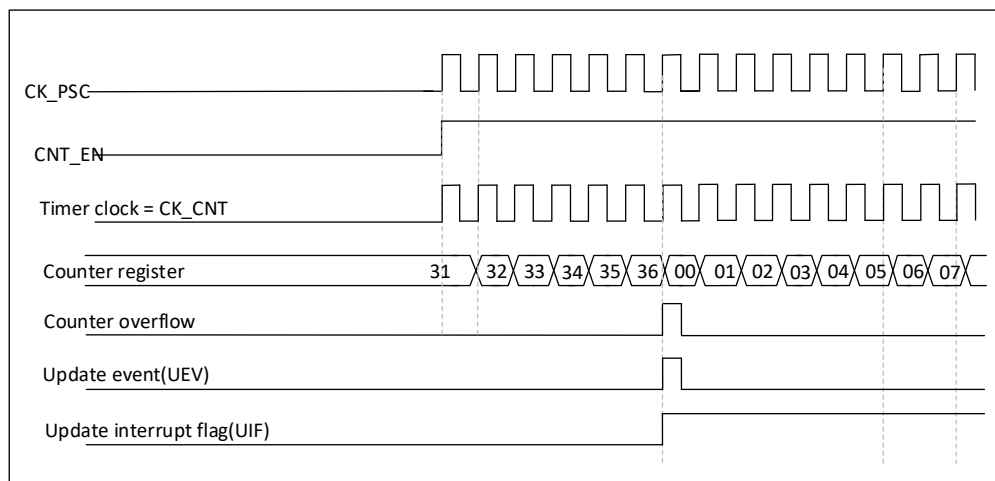


图 21-4 计数器时序图，内部时钟分频因子为1

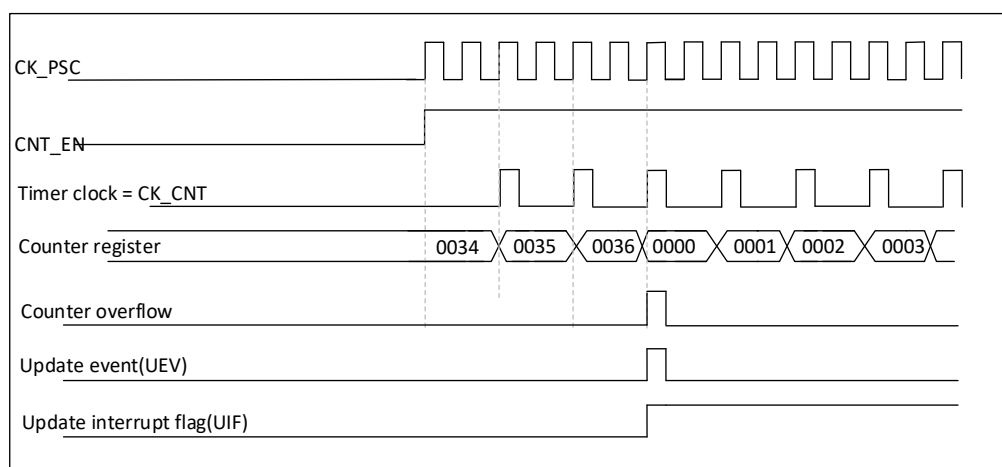


图 21-5 计数器时序图，内部时钟分频因子为 2



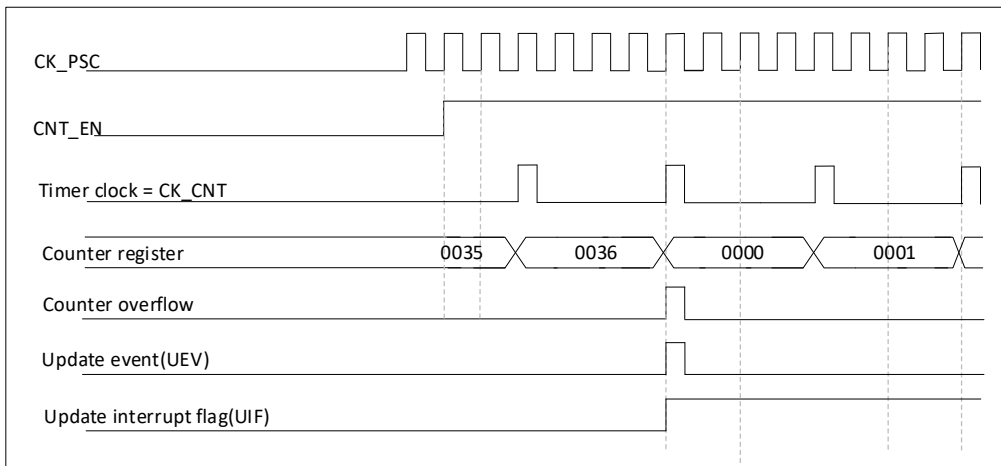


图 21-6 计数器时序图，内部时钟分频因子为 4

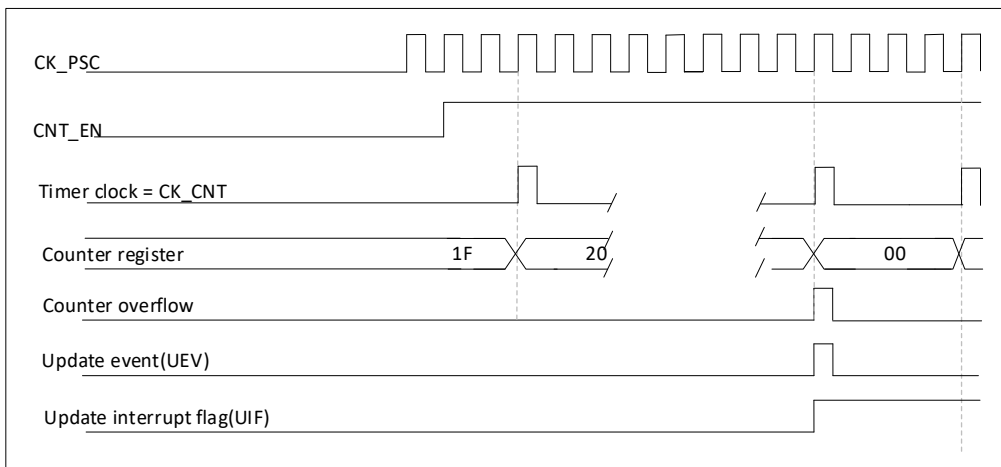


图 21-7 计数器时序图，内部时钟分频因子为 N

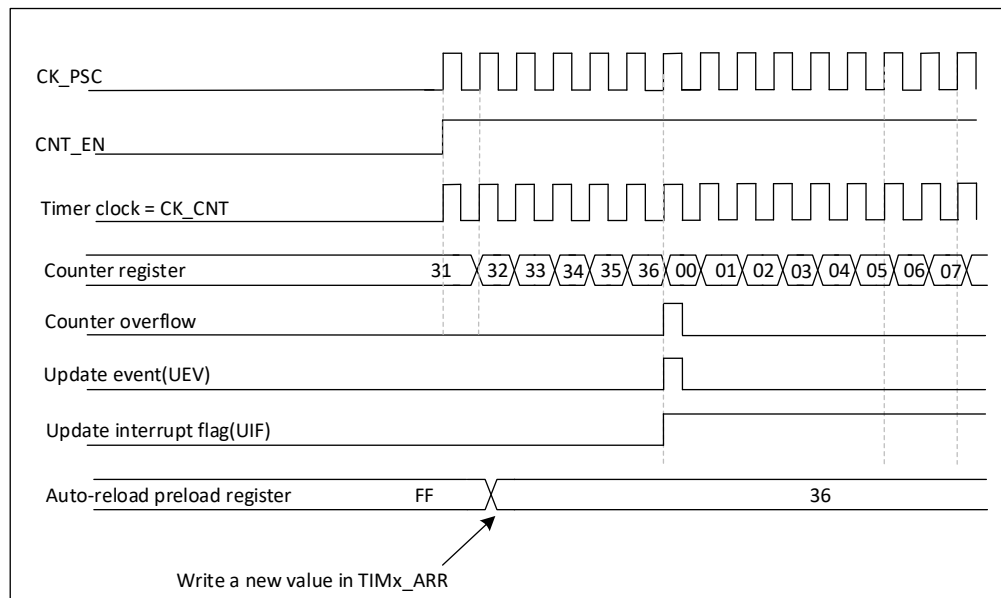


图 21-8 计数器时序图，当 ARPE=0 时的更新事件 (TIMx\_ARR 没有预装入)

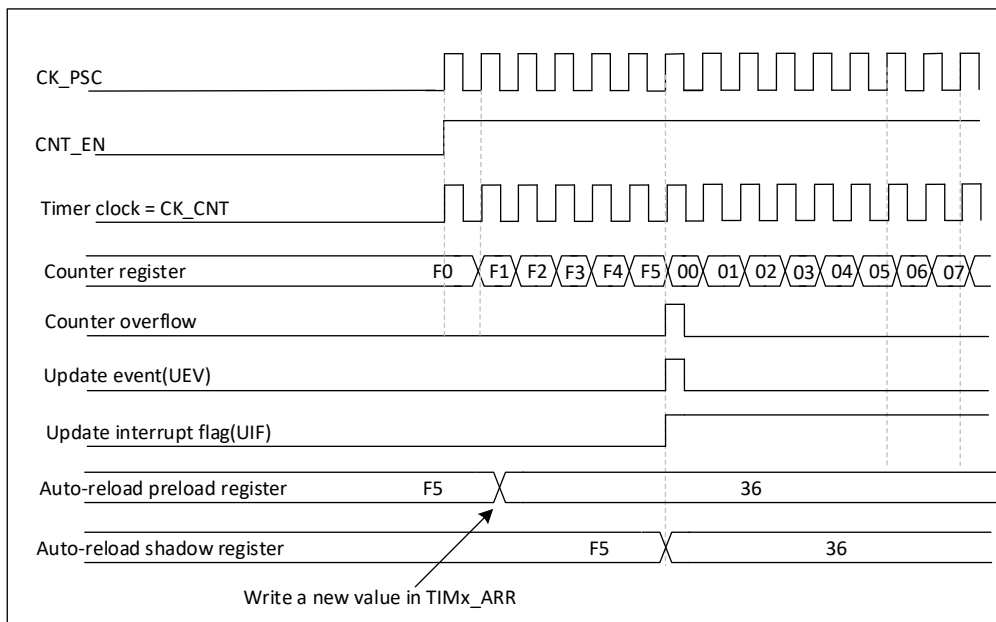


图 21-9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx\_ARR）

### 向下计数模式

向下计数模式，从自动重载的值开始向下计数到 0，然后重新开始从自动重载的值向下计数，并产生一个向下溢出事件。

如果使用了重复计数器，当向下计数重复了重复计数寄存器（TIMx\_RCR）中设定的次数后，将产生更新事件（UEV），否则每次计数器下溢时才产生更新事件。

在 TIMx\_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始（但预分频系数不变）。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx\_SR 寄存器中的 UIF 位）也被设置。

- 重复计数器被重置为 TIMx\_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载的值（TIMx\_PSC 寄存器的值）
- 当前的自动重载寄存器被更新为预装载值（TIMx\_ARR 寄存器中的内容）

注：自动重载寄存器在计数器重载入之前被更新，因此下一个周期将是预期的值。

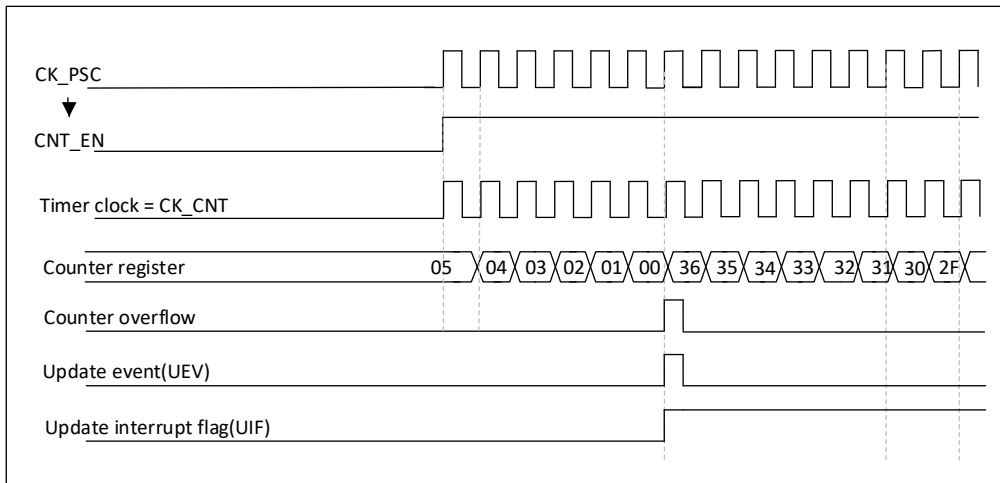


图 21-10 计数器时序图，内部时钟分频因子为 1

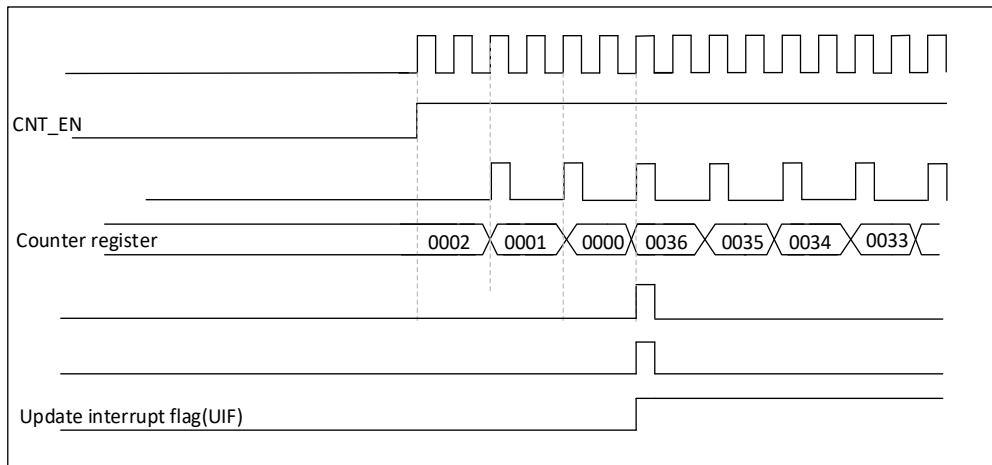


图 21-11 计数器时序图，内部时钟分频因子为 2

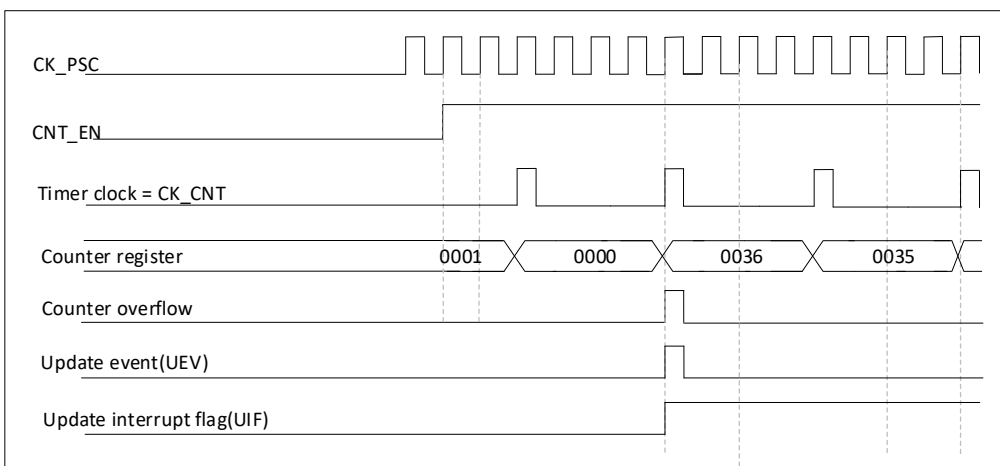


图 21-12 计数器时序图，内部时钟分频因子为 4

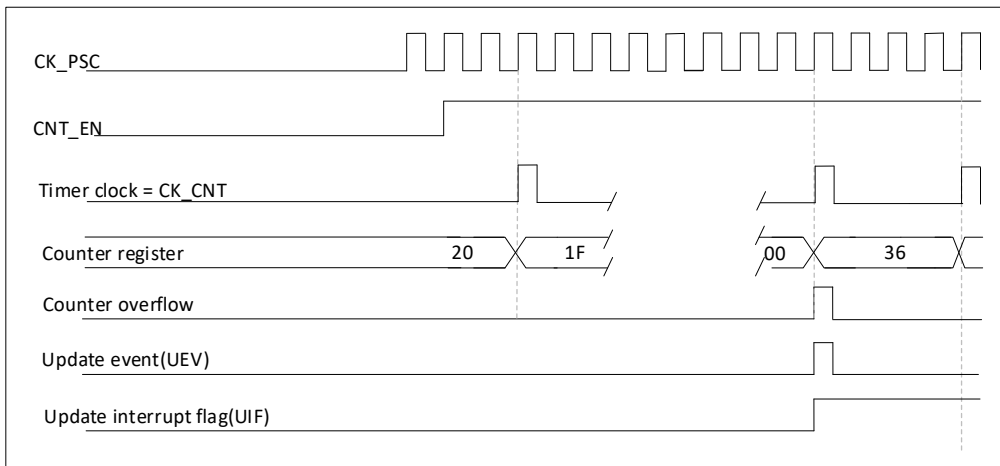


图 21-13 计数器时序图，内部时钟分频因子为 N

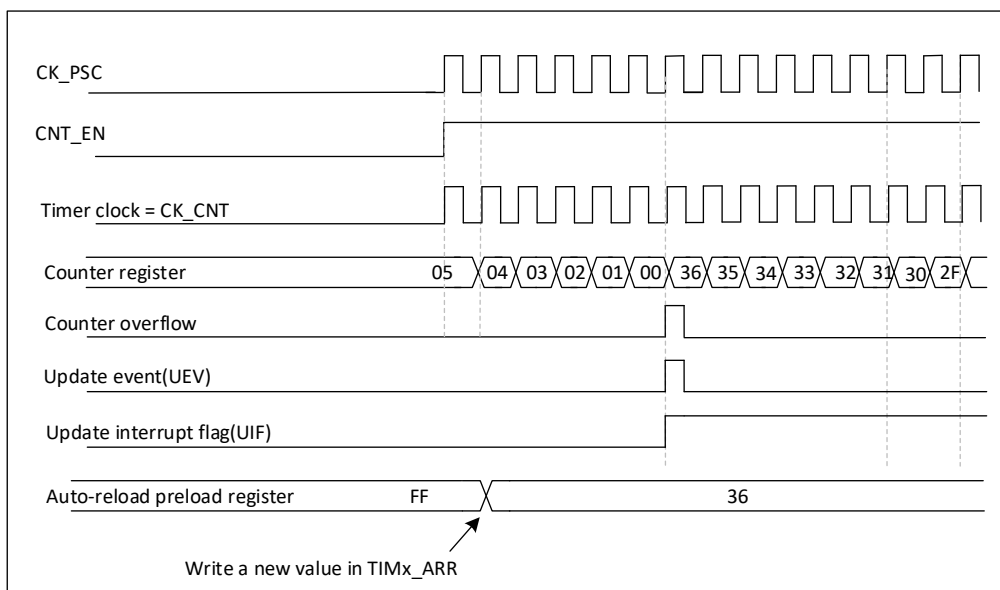


图 21-14 计数器时序图，当没有使用周期计数器时的更新事件

### 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值（TIMx\_ARR 寄存器）-1，产生一个计数器上溢事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

中央对齐模式在 TIMx\_CR1 寄存器中的 CMS 不等于 0 时有效。通道在配置成输出模式时，输出比较中断标志将被置位，当：向下计数时（中央对齐模式 1，CMS="01"），向上计数时（中央对齐模式 2，CMS="10"）向上向下计数（中央对齐模式 3，CMS="11"）。

在此模式下，不能写入 TIMx\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）设置 TIMx\_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx\_SR 寄存器中的 UIF 位）也被设置。

- 重复计数器被重置为 TIMx\_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载（TIMx\_PSC 寄存器）的值。
- 当前的自动重载寄存器被更新为预装载值（TIMx\_ARR 寄存器中的内

注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）

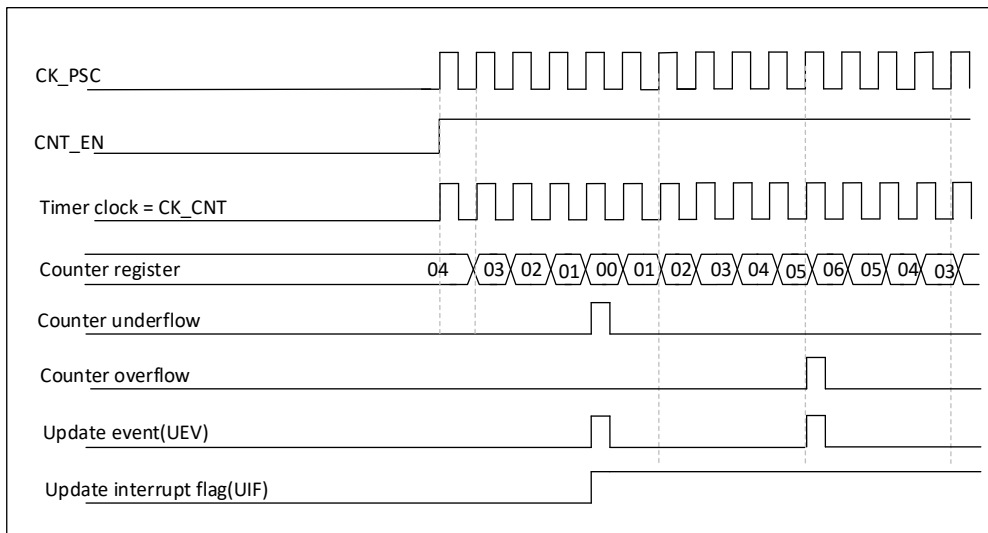


图 21-15 计数器时序图，内部时钟分频因子为 1，TIMx\_ARR = 0x6

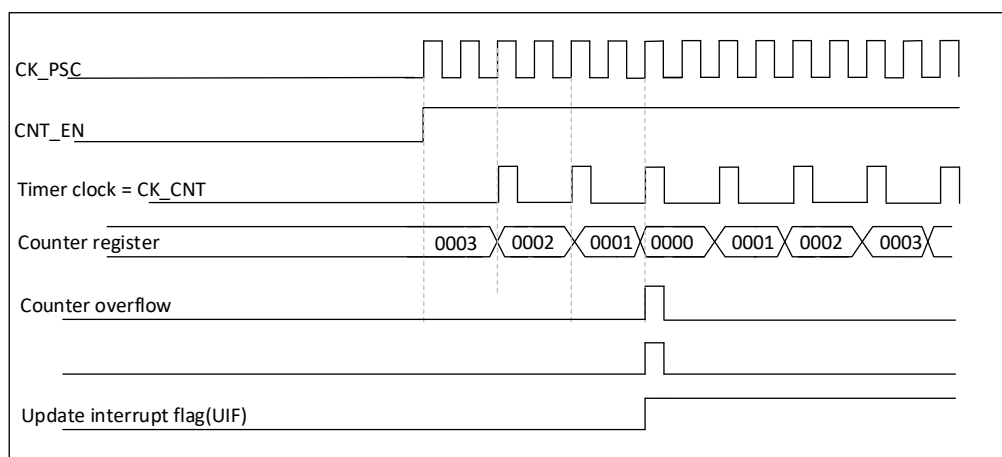


图 21-16 计数器时序图，内部时钟分频因子为 2，TIMx\_ARR=0x36

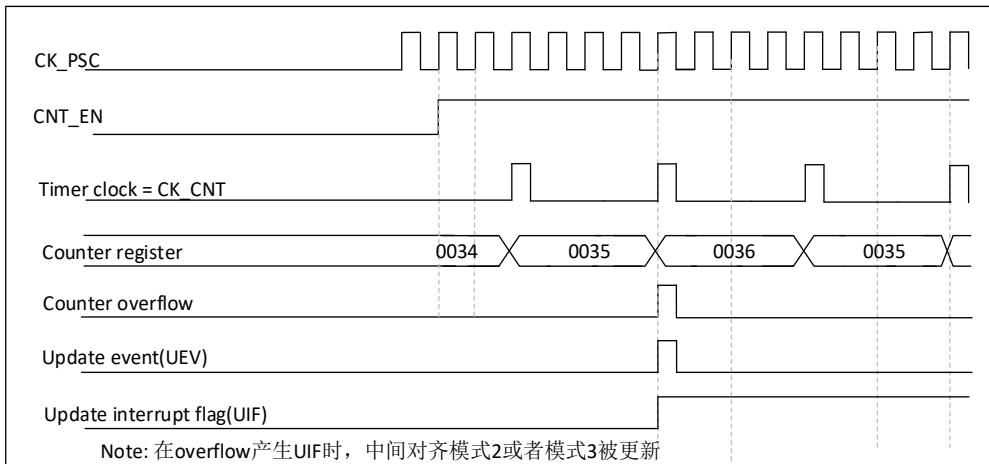


图 21-17 计数器时序图，内部时钟分频因子为 4，TIMx\_ARR=0x36

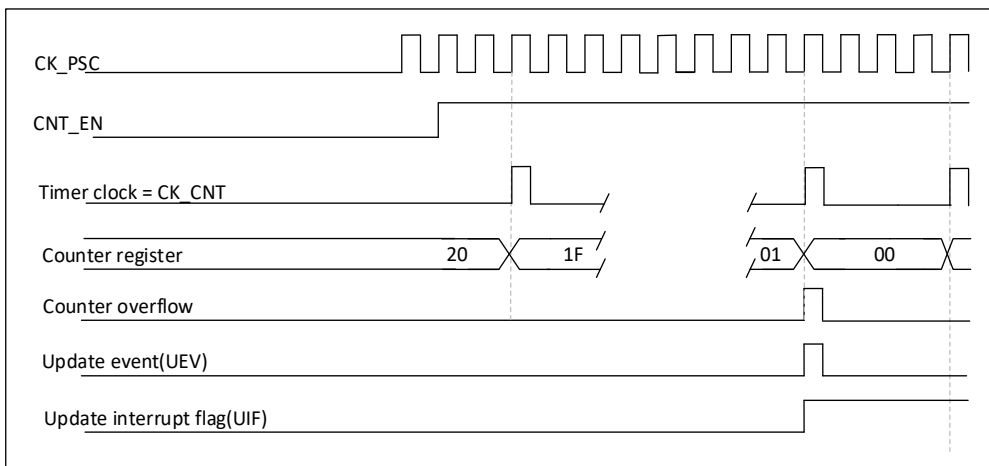


图 21-18 计数器时序图，内部时钟分频因子为 N

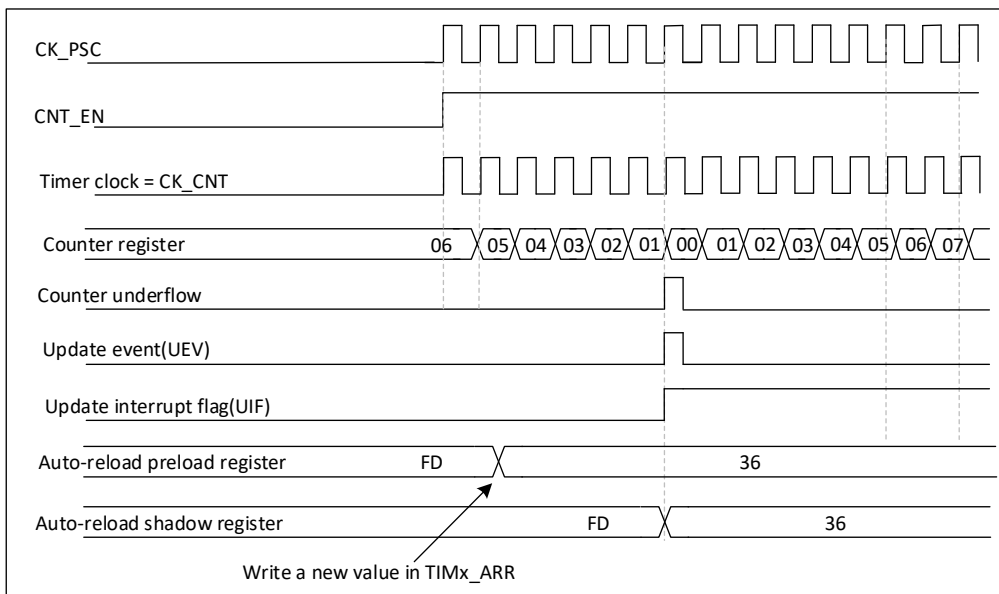


图 21-19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

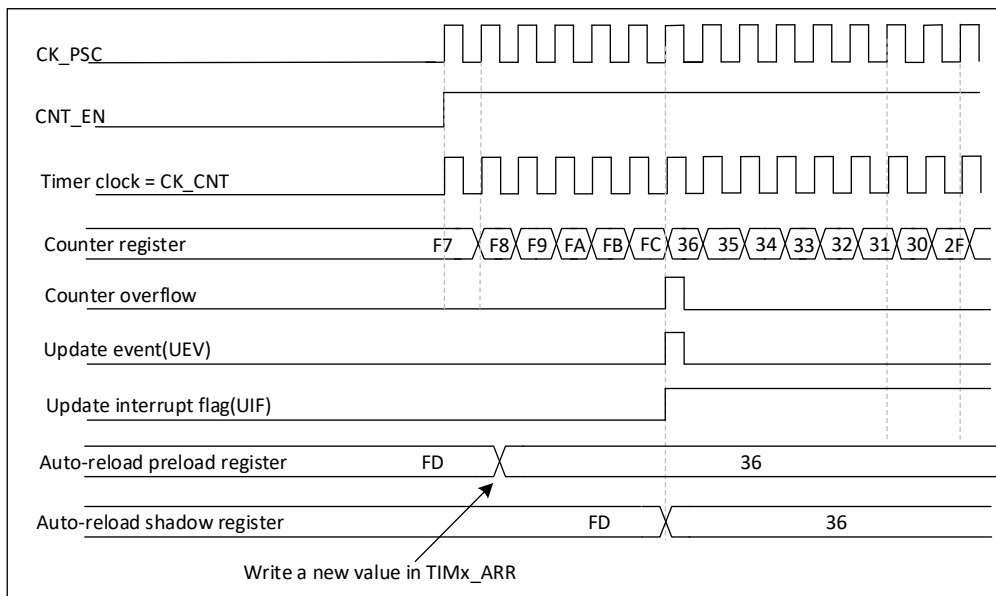


图 21-20 计数器时序图, ARPE=1 时的更新事件 (计数器上溢)

### 21.3.3. 重复计数器

时基单元描述了关于计数器向上、向下溢出的更新事件 (UEV) 是如何产生的。它实际上仅当重复计数器计数到零才产生。这个特性对产生 PWM 信号非常有用。

这意味着在每  $N+1$  次计数上溢或下溢时, 数据被从预装载寄存器传送到影子寄存器 (TIMx\_ARR 自动重载入寄存器, TIMx\_PSC 预装载寄存器, 还有在比较模式下的捕获/比较寄存器 TIMx\_CCRx),  $N$  是 TIMx\_RCR 重复计数寄存器中的值。

重复计数器在下述任何一条件成立时递减:

- 向上计数模式下每次计数器溢出时
- 向下计数模式下每次计数下溢时
- 中央对齐模式下每次上溢和每次下溢时。

中央对齐模式中, 虽然这样限制了 PWM 的最大循环周期位 128, 但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下, 因为波形是对称的, 如果每个 PWM 周期中仅刷新一次比较寄存器, 则最大的分辨率为  $2xTck$ 。

重复计数器是自动加载的, 重复速率是由 TIMx\_RCR 寄存器的值定义。当更新事件由软件产生 (通过设置 TIMx\_EGR 中的 UG 位) 或者通过硬件的从模式控制器产生, 则无论重复计数器的值是多少, 立即发生更新事件, 并且 TIMx\_RCR 寄存器中的内容被重载如到重复计数器。

在中央对齐模式下, 对于 RCR 的奇数值, 取决于当 RCR 寄存器被写入和当计数器开始, 出现上溢、或者下溢, 则更新事件产生。如果在启动计数器之前写 RCR, 在上溢时产生更新事件。例如, 对于 RCR = 3 时, 更新事件被产生在第 4 个上溢或者下溢事件 (取决于 RCR 被写入的值)。

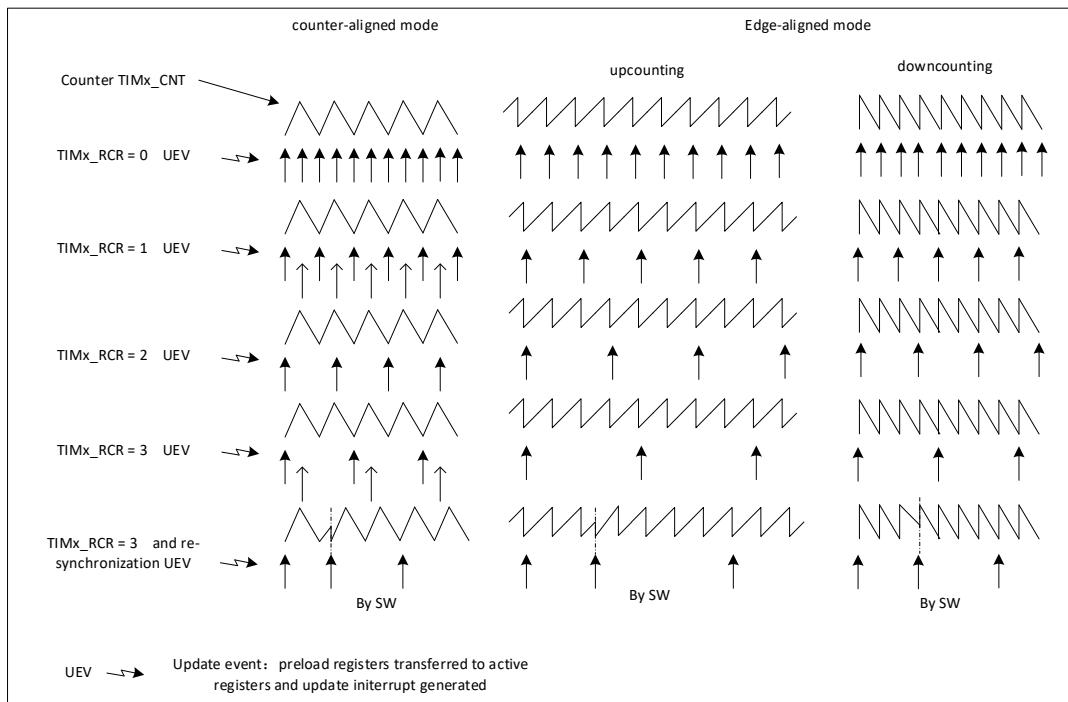


图 21-21 不同模式下更新速率的例子，及 TIMx\_RCR 的寄存器设置

### 21.3.4. 时钟源

计数器的时钟可以由以下时钟源提供：

- 内部时钟 (CK\_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入 (ITRx)：使用一个定时器作为另一个定时器的预分频器。例如，可以配置一个定时器 Timer1 作为另一个定时器 Timer2 的预分频器。

#### 内部时钟源 (CK\_INT)

如果从模式控制器被禁止，则 CEN、DIR (TIMx\_CR1 寄存器) 和 UG 位 (TIMx\_EGR 寄存器) 是事实上的控制位，并且只能被软件修改。只要 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作



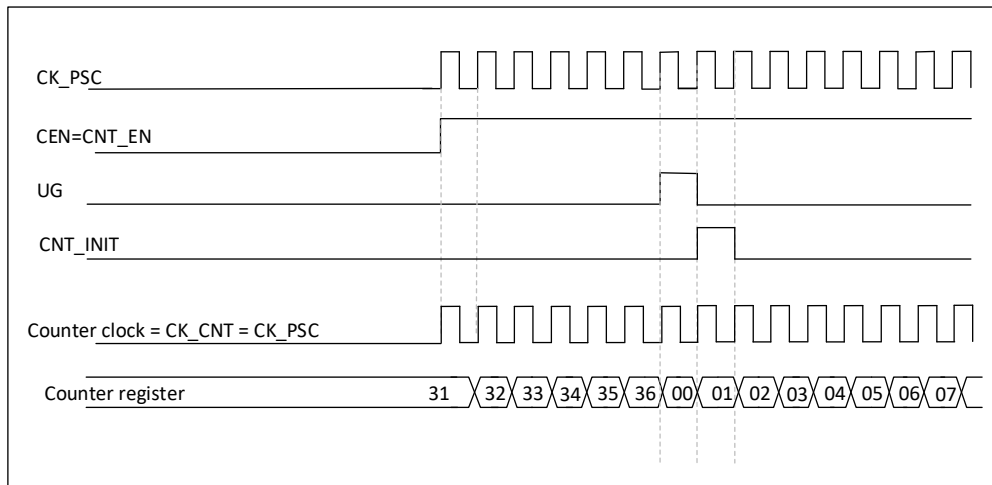


图 21-22 一般模式下的控制电路，内部时钟分频因子为 1

**外部时钟源模式1**

当 TIMx\_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

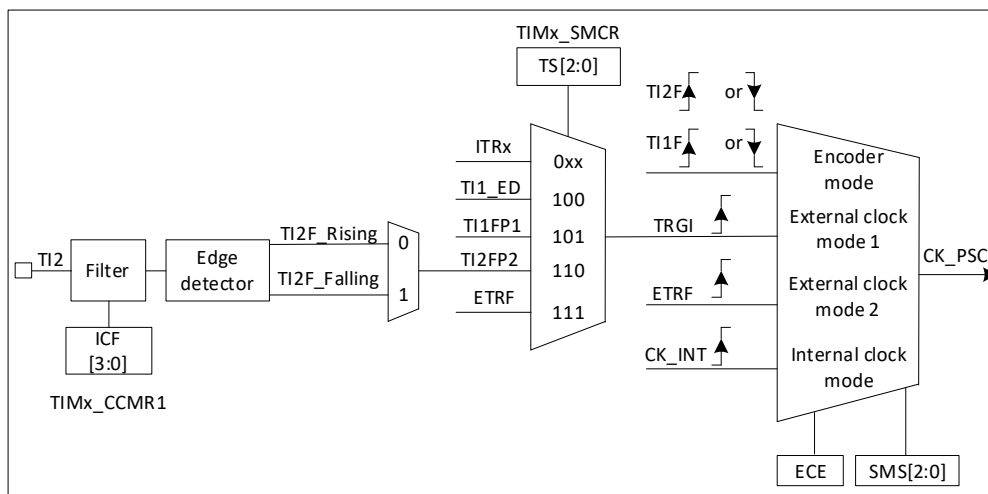


图 21-23 TI2 外部时钟连接例子

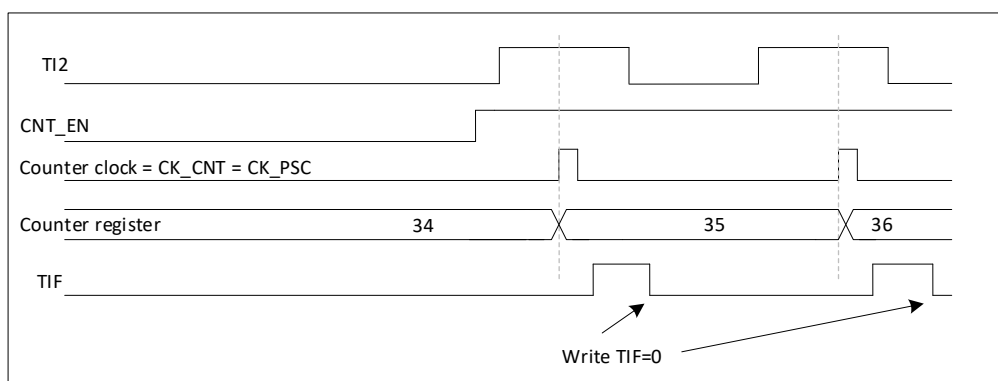


图 21-24 外部时钟模式 1 下的控制电路

**外部时钟源模式2**

通过写 TIMx\_SMCR 寄存器的 ECE 为 1，选定此模式。计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

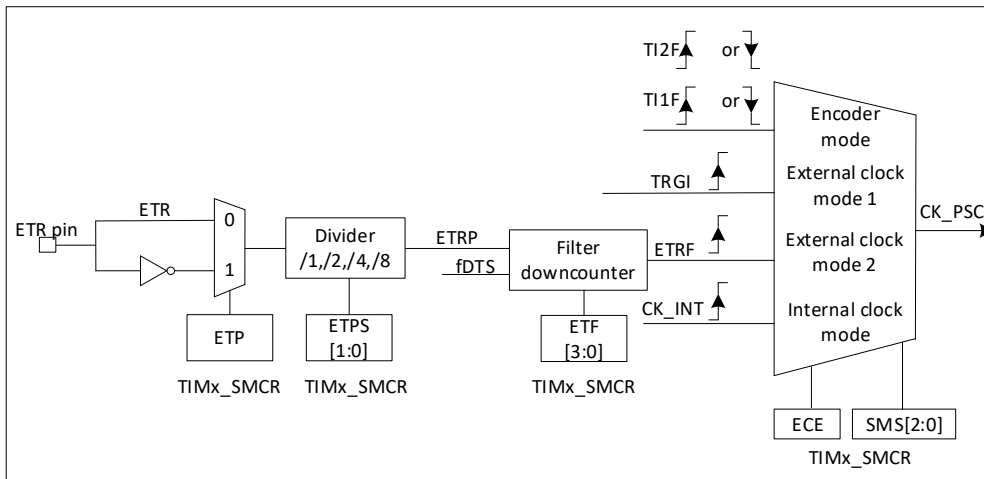


图 21-25 TI2外部触发输入框图

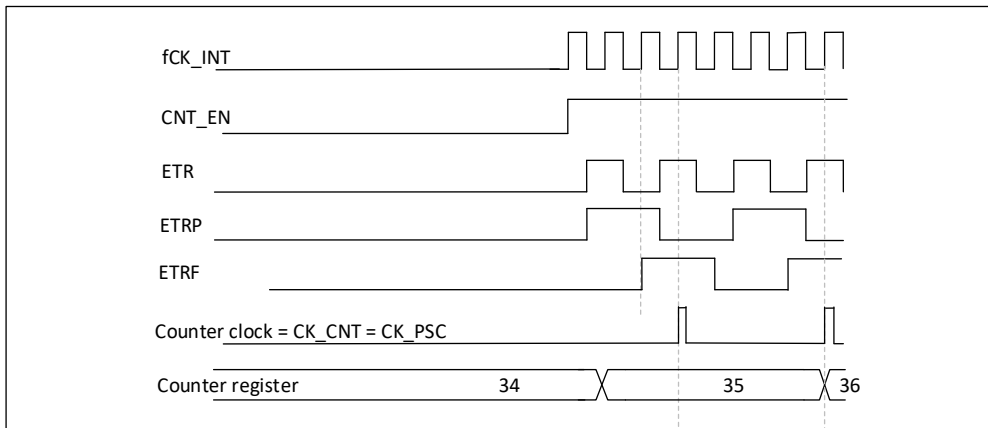


图 21-26 外部时钟模式 2 下的控制电路

### 21.3.5. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（输入滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

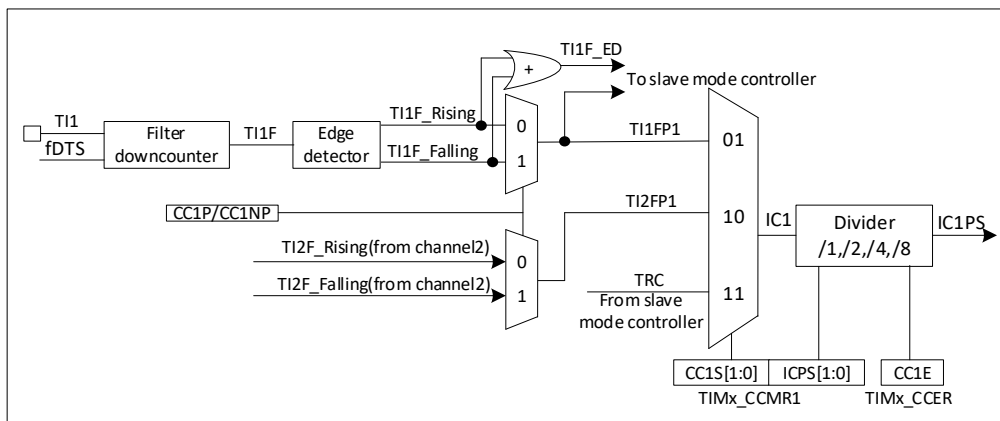


图 21-27 捕获/比较通道 (如: 通道 1 输入部分)

输出部分产生一个中间波形 OCxREF（高有效）作为基准，链的末端决定最终输出信号的极性。

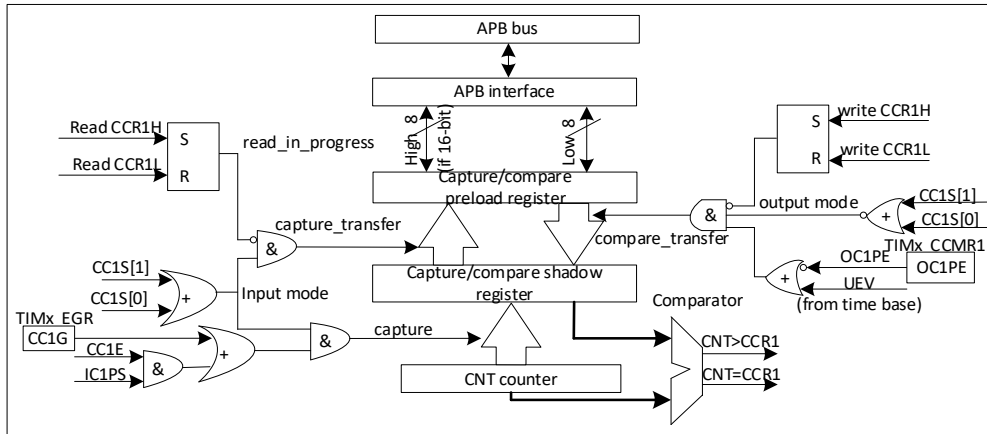


图 21-28 捕获/比较通道 1 的主电路

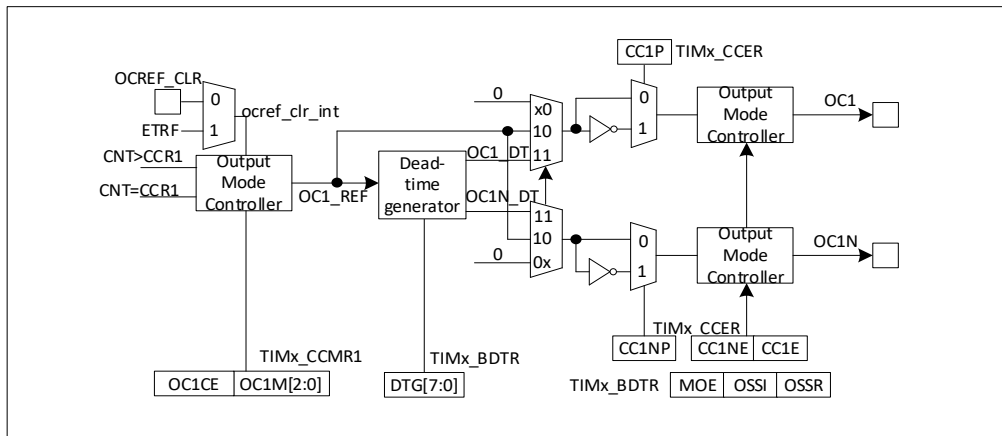


图 21-29 捕获/比较通道的输出部分（通道 1 至 3）

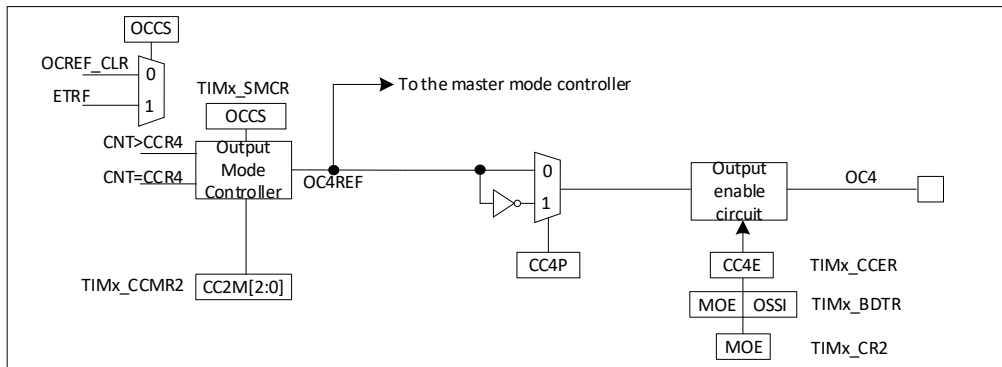


图 21-30 捕获/比较通道的输出部分（通道 4）

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

### 21.3.6. 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器中。当发生捕获事件时，相应的 CCxIF 标志 (TIMx\_SR 寄存器) 被置 1，如果中断和 DMA 操作被打开，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，则重复捕获标志 CCxOF (TIMx\_SR 寄存器) 被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx\_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx\_CCR1 寄存器中，步骤如下：

1. 选择有效输入端：TIMx\_CCMR1 必须连接到 TI1 输入，所以写入 TIMx\_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx\_CCR1 寄存器变为只读。
2. 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 TIx 时，输入滤波器控制位是 TIMx\_CCMRx 寄存器中的 ICxF 位）。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以（以 fDTS 频率）连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx\_CCMR1 寄存器中写入 IC1F=0011。
3. 选择 TI1 通道的有效转换边沿，在 TIMx\_CCER 寄存器中写入 CC1P=0（上升沿）
4. 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx\_CCMR1 寄存器的 IC1PS=00）。
5. 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置 TIMx\_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx\_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

### 21.3.7. PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射到同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，当需要测量输入到 TI1 上的 PWM 信号的长度 (TIMx\_CCR1 寄存器) 和占空比 (TIMx\_CCR2 寄存器) 时，具体步骤如下（取决于 CK\_INT 的频率和预分频器的值）

- 选择 TIMx\_CCR1 的有效输入：置 TIMx\_CCMR1 寄存器的 CC1S=01（选中 TI1）。
- 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx\_CCR1 中和清除计数器）：置 CC1P=0（上升沿有效）。
- 选择 TIMx\_CCR2 的有效输入：置 TIMx\_CCMR1 寄存器的 CC2S=10（选中 TI1）。
- 选择 TI1FP2 的有效极性（捕获数据到 TIMx\_CCR2）：置 CC2P=1（下降沿有效）。
- 选择有效的触发输入信号：置 TIMx\_SMCR 寄存器中的 TS=101（选择 TI1FP1）。
- 配置从模式控制器为复位模式：置 TIMx\_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx\_CCER 寄存器中 CC1E=1 且 CC2E=1。

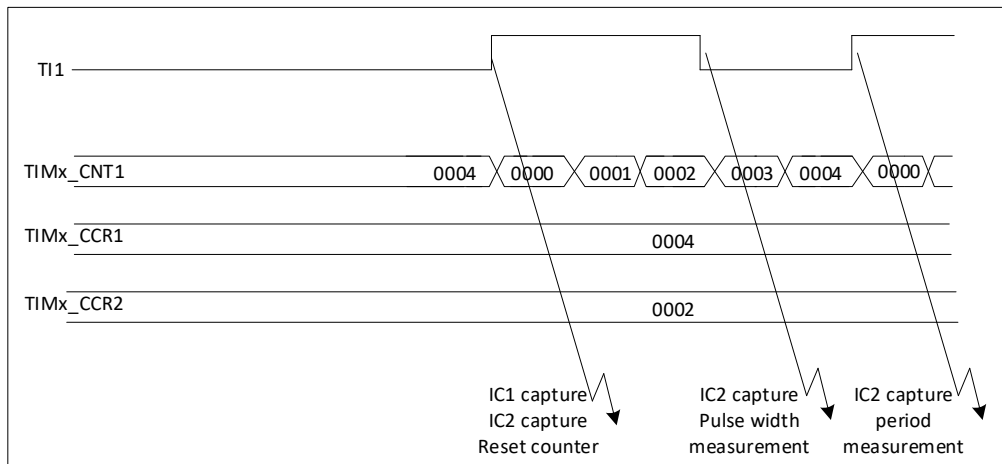


图 21-31 PWM 输入模式时序

### 21.3.8. 强置输出模式

在输出模式（TIMx\_CCMRx 寄存器中 CCxS=00）下，输出比较信号（OCxREF 和相应的 OCx/OCxN）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。置 TIMx\_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0（OCx 高电平有效），则 OCx 被强置为高电平。置 TIMx\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

### 21.3.9. 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TIMx\_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx\_CCER 寄存器中的 CCxP 位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxM=000）、被设置成有效电平（OCxM=001）、被设置成无效电平（OCxM=010）或进行翻转（OCxM=011）。

- 设置中断状态寄存器中的标志位 (TIMx\_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx\_DIER 寄存器中的 CCxIE 位)，则产生一个中断。
- 若设置了相应的使能位 (TIMx\_DIER 寄存器中的 CCxDE 位, TIMx\_CR2 寄存器中的 CCDS 位选择 DMA 请求功能)，则产生一个 DMA 请求。

TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式 (在单脉冲模式下) 也能用来输出一个单脉冲。

输出比较模式的配置步骤:

1. 选择计数器时钟 (内部, 外部, 预分频器)。
2. 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
3. 如果要产生一个中断请求, 设置 CCxIE 位。
4. 选择输出模式, 例如:
  - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚, 设置 OCxM=011
  - 置 OCxPE = 0 禁用预装载寄存器
  - 置 CCxP = 0 选择极性为高电平有效
  - 置 CCxE = 1 使能输出
5. 设置 TIMx\_CR1 寄存器的 CEN 位启动计数器

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形, 条件是未使用预装载寄存器 (OCxPE='0', 否则 TIMx\_CCRx 的影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

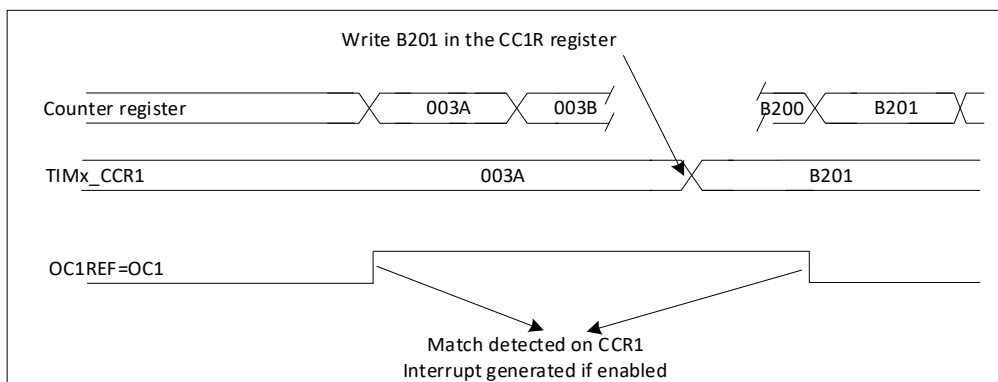


图 21-32 输出比较模式, 翻转 OC1

### 21.3.10. PWM 模式

脉冲宽度调制模式可以允许产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入 “110” (PWM 模式 1) 或 “111” (PWM 模式 2), 能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器, 最后还要设置 TIMx\_CR1 寄存器的 ARPE 位, (在向上计数或中心对称模式中) 使能自动重载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过 (TIMx\_CCER 和 TIMx\_BDTR 寄存器中) CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx\_CCER 寄存器的描述。

在 PWM 模式 (模式 1 或模式 2) 下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，(依据计数器的计数方向) 以确定是否符合  $TIMx\_CCRx \leq TIMx\_CNT$  或者  $TIMx\_CNT \leq TIMx\_CCRx$ 。

根据 TIMx\_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

## PWM 边沿对齐模式

### ■ 向上计数配置

当 TIMx\_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看下面是一个 PWM 模式 1 的例子。当  $TIMx\_CNT < TIMx\_CCRx$  时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自动重装载值 (TIMx\_ARR)，则 OCxREF 保持为 '1'。如果比较值为 0，则 OCxREF 保持为 '0'。下图为 TIMx\_ARR=8 时边沿对齐的 PWM 波形实例。

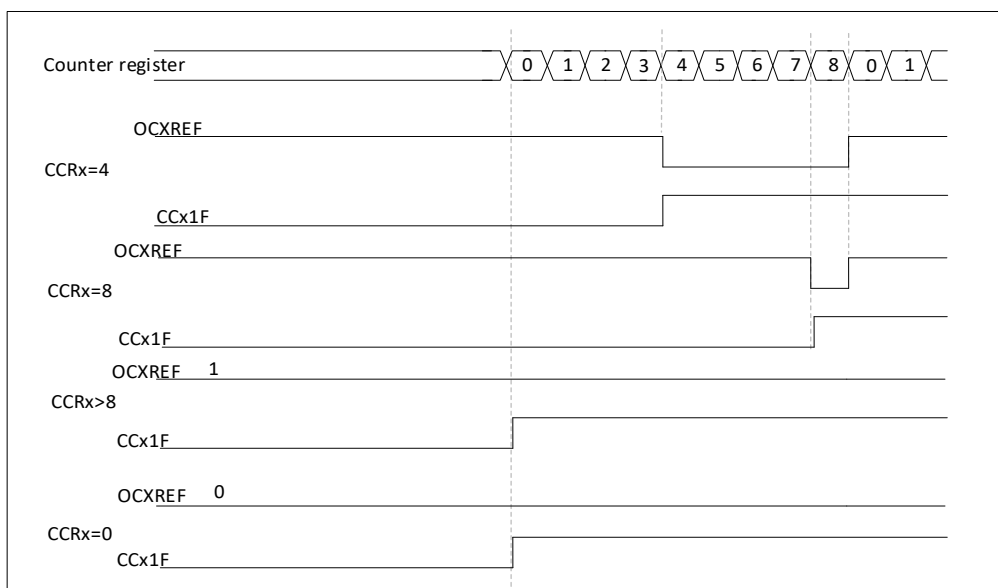


图 21-33 边沿对齐方式 PWM 输出，向上 (ARR=8)

### ■ 向下计数的配置

当 TIMx\_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当  $TIMx\_CNT > TIMx\_CCRx$  时参考信号 OCxREF 为低，否则为高。如果 TIMx\_CCRx 中的比较值大于 TIMx\_ARR 中的自动重装载值，则 OCxREF 保持为 '1'。该模式下不能产生 0% 的 PWM 波形。

### PWM 中央对齐模式

当 TIMx\_CR1 寄存器中的 CMS 位不为 '00' 时为中央对齐模式 (所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx\_CR1 寄存器中的计数方向位 (DIR) 由硬件更新，不要用软件修改它。

下图给出一些中央对齐的 PWM 波形的例子

- $TIMx\_ARR = 8$
- PWM 模式1
- $TIMx\_CR1$ 寄存器的  $CMS=01$ ，在中央对齐模式下，当计数器向下计数时设置比较标志

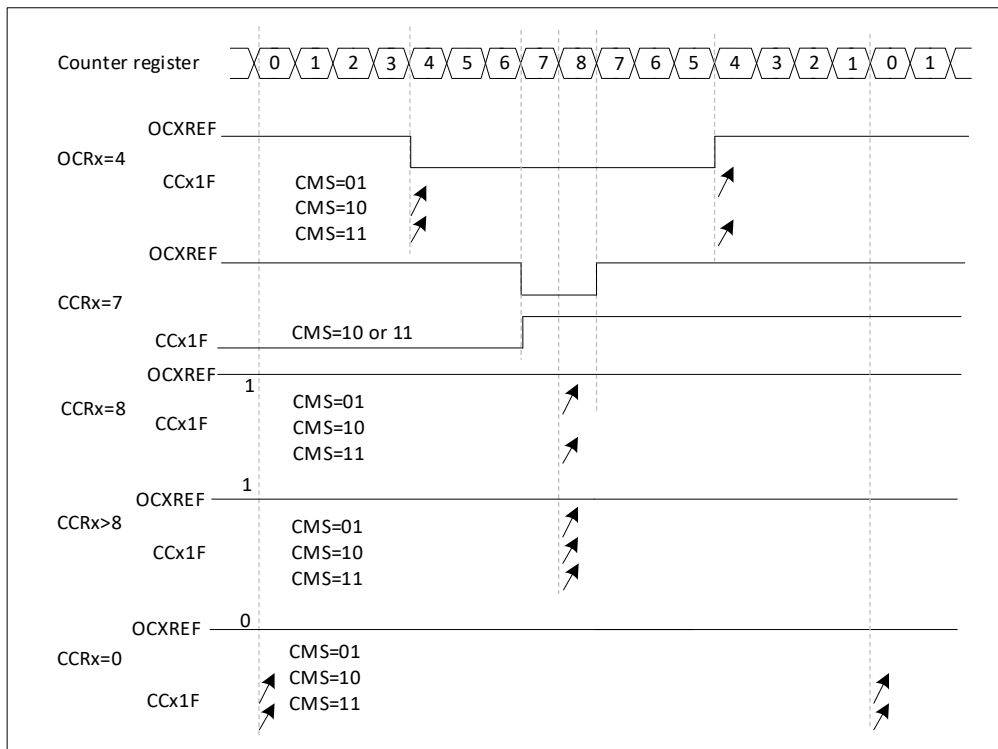


图 21-34 中央对齐的 PWM 波形 (APR=8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于  $TIMx\_CR1$  寄存器中  $DIR$  位的当前值。此外，软件不能同时修改  $DIR$  和  $CMS$  位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：— 如果写入计数器的值大于自动重加载的值 ( $TIMx\_CNT > TIMx\_ARR$ )，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。— 如果将 0 或者  $TIMx\_ARR$  的值写入计数器，方向被更新，但不产生更新事件  $UEV$ 。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置  $TIMx\_EGR$  位中的  $UG$  位），并且不要在计数进行过程中修改计数器的值。

### 21.3.11. 互补输出和死区插入

高级控制定时器 ( $TIM1$ ) 能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性（电平转换的延时、电源开关的延时等）来调整死区时间。

配置  $TIMx\_CCER$  寄存器中的  $CCxP$  和  $CCxNP$  位，可以为每一个输出独立地选择极性（主输出  $OCx$  或互补输出  $OCxN$ ）。

互补信号  $OCx$  和  $OCxN$  通过下列控制位的组合进行控制： $TIMx\_CCER$  寄存器的  $CCxE$  和  $CCxNE$  位， $TIMx\_BDTR$  和  $TIMx\_CR2$  寄存器中的  $MOE$ 、 $OISx$ 、 $OISxN$ 、 $OSSI$  和  $OSSR$  位，详见表21-2带刹车功能的互补输出通道  $OCx$  和  $OCxN$  的控制位。特别的是，在转换到  $IDLE$  状态时 ( $MOE$  下降到0) 死区被激活。



同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 8 位的死区发生器 DTG[7:0]。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度（OCx 或者 OCxN），则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。（假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

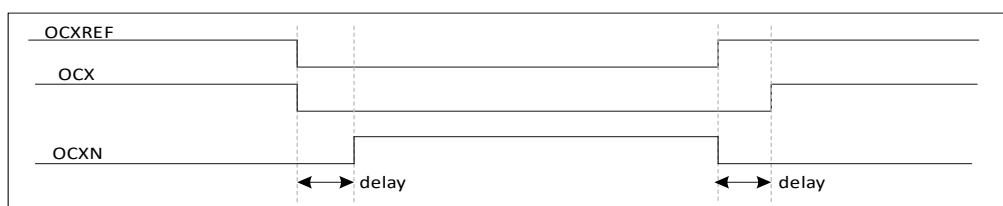


图 21-35 带死区插入的互补输出

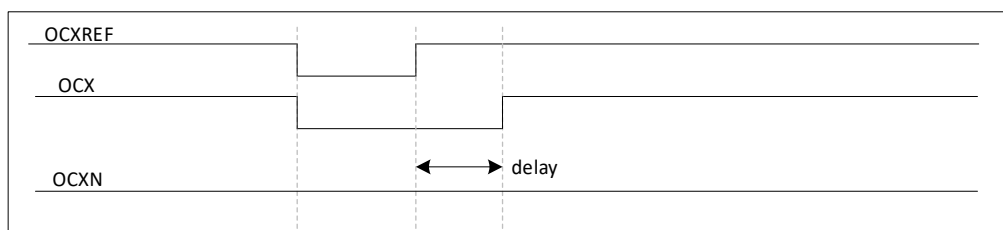


图 21-36 死区波形延迟大于负脉冲

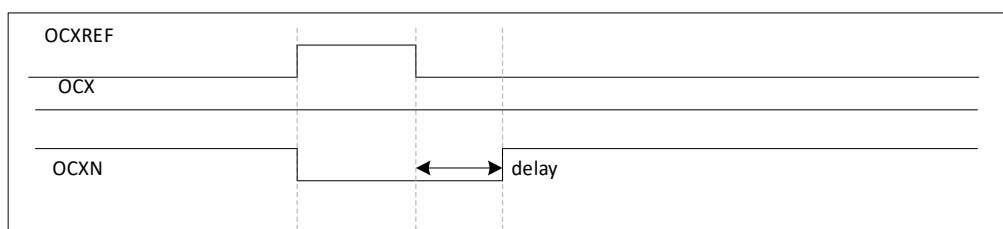


图 21-37 死区波形延迟大于正脉冲

每一个通道的死区延时都是相同的，是由 TIMx\_BDTR 寄存器中的 DTG 位编程配置。

### 重定向 OCxREF 到 OCx 或 OCxN

在输出模式下（强置、输出比较或 PWM），通过配置 TIMx\_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形（例如 PWM 或者静态有效电平）。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注：当只使能 OCxN（CCxE=0，CCxNE=1）时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时（CCxE=CCxNE=1），当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

### 21.3.12. 使用刹车功能

当使用刹车功能时，依据额外的控制位，输出使能信号和无效电平信号都会被修改。无论什么情况下，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。

刹车源既可以是刹车输入引脚，或者以下内部源：

- CPU LOCKUP 输出
- PVD 输出
- 由 CSS 监测产生的时钟 failure 事件
- 来自比较器的输出

系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx\_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有1个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（在 TIMx\_BDTR 寄存器中）之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时（空指令）才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时（在刹车输入端出现选定的电平），有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者释放对 GPIO 的控制（由 OSSI 位选择）。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx\_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些（大约2个 ck\_tim 的时钟周期）。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 如果设置了 TIMx\_DIER 寄存器中的 BIE 位，当刹车状态标志（TIMx\_SR 寄存器中的 BIF 位）为'1'时，则产生一个中断。
- 如果设置了 TIMx\_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能同时（自动地或者通过软件）设置 MOE。同时，状态标志 BIF 不能被清除。

刹车可以由 BRK 输入产生，它的有效极性是可编程的，且由 TIMx\_BDTR 寄存器中的 BG 位开启。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数（死区长度，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性）。用户可以通过 TIMx\_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

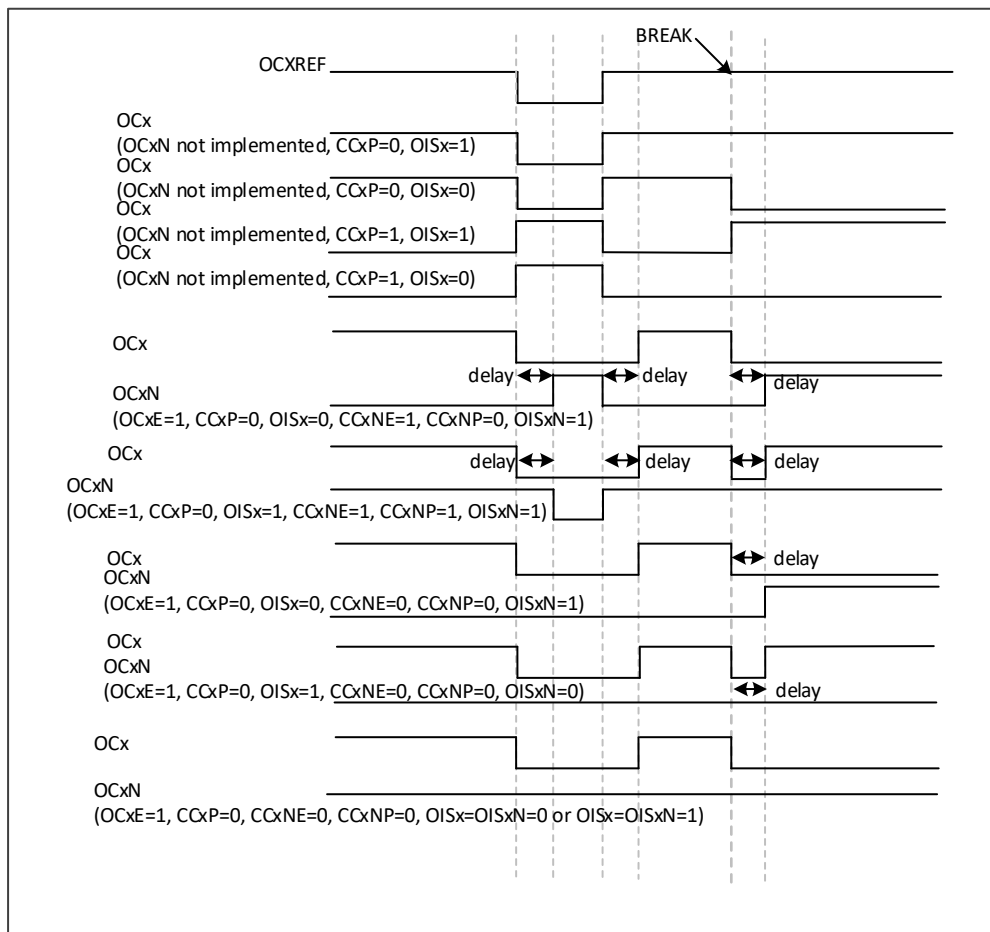


图 21-38 响应刹车的输出

### 21.3.13. 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx\_CCMRx 寄存器中对应的 OCxCE 位为1，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低电平，直到下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强制模式。

而 OCREF\_CLR\_INPUT 可以通过配置 TIMx\_SMCR 寄存器中的 OCCS 位，在 OCREF\_CLR 和 ETRF (ETR 滤波后) 之间选择。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx\_SMCR 寄存器中的 ETPS[1:0]=00。
2. 必须禁止外部时钟模式2：TIMx\_SMCR 寄存器中的 ECE=0。
3. 外部触发极性 (ETP) 和外部触发滤波器 (ETF) 可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM 模式。

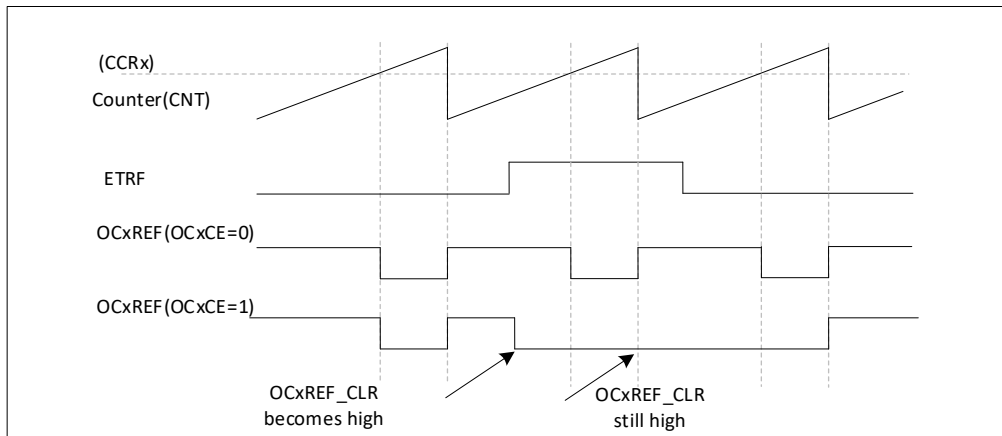


图 21-39 清除 TIM1 的 OCxREF

### 21.3.14. 六步 PWM 的产生

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样就可以预先设置好下一步骤配置，并在同一个时刻同时修改所有通道的配置。COM 可以通过设置 TIMx\_EGR 寄存器的 COM 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位 (TIMx\_SR 寄存器中的 COMIF 位)，这时如果已设置了 TIMx\_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx\_DIER 寄存器的 COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

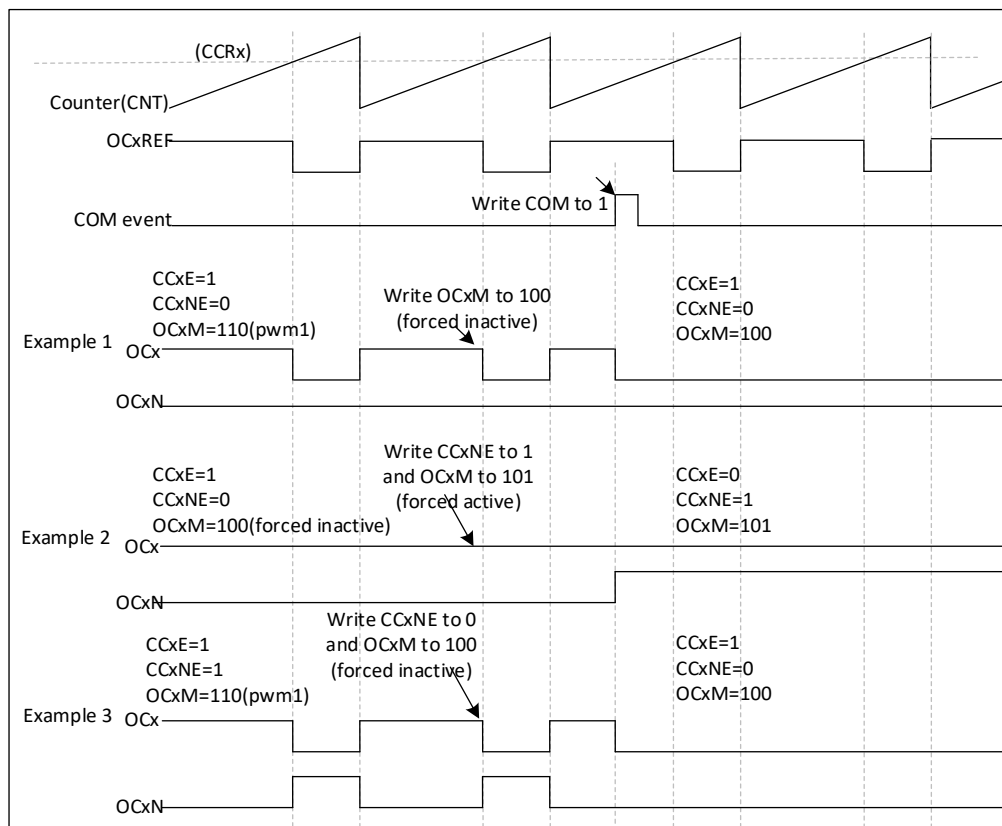


图 21-40 六步产生，COM 的例子 (OSSR=1)

### 21.3.15. 单脉冲模式

单脉冲模式 (OPM) 是之前所述众多模式中的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可被程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器的 OPM 位将选择单脉冲模式，这样可以使计数器自动地在产生下一次计数器溢出时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$ （特别地， $0 < CCRx$ ）
- 向下计数方式：计数器  $CNT > CCRx$

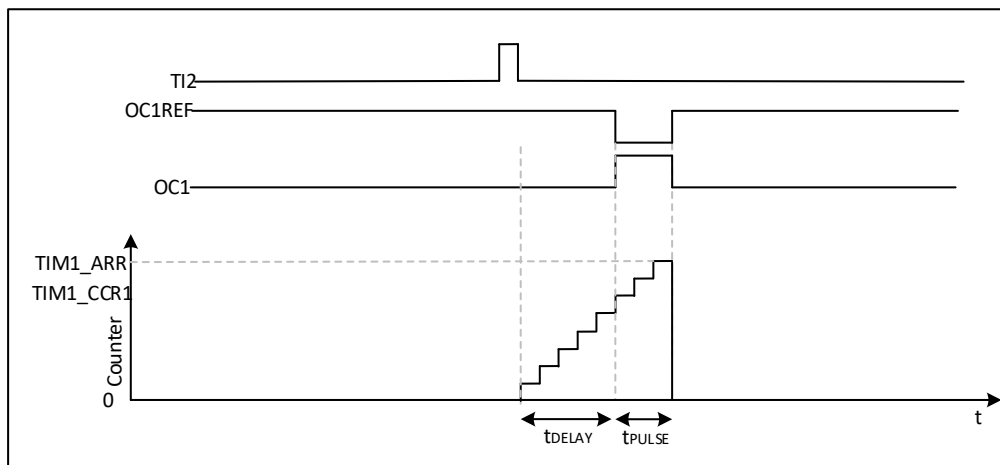


图 21-41 单脉冲模式的例子

例如，当需要在从 TI2 输入脚上检测到一个上升沿开始，延迟  $t_{DELAY}$  之后，在 OC1 上产生一个长度为  $t_{PULSE}$  的正脉冲。

使用 TI2FP2 作为触发：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx\_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发 (TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110 (触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）

- $t_{DELAY}$  由 TIMx\_CCR1 寄存器中的值定义。
- $t_{PULSE}$  由自动重载值和比较值之间的差值定义 ( $TIMx\_ARR - TIMx\_CCR1$ )。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要可选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE=1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动重载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM=1，在下一个更新事件（当计数器从自动重载值翻转到 0）时停止计数。

**特殊情况：OCx 快速使能：**

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx\_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF（和 OCx）直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

**21.3.16. 编码器接口模式**

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx\_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx\_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 21-1，假定计数器已经启动（TIMx\_CR1 寄存器中的 CEN=1），则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx\_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端（TI1 或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx\_ARR 寄存器的自动重载值之间连续计数（根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数）。所以在开始计数之前必须配置 TIMx\_ARR；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 21-1 计数方向与编码器信号的关系

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
Counting on TI2 only	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S='01' (TIMx\_CCMR1寄存器, TI1FP1映射到 TI1)
- CC2S='01' (TIMx\_CCMR2寄存器, TI2FP2映射到 TI2)
- CC1P='0' (TIMx\_CCER 寄存器, TI1FP1不反相, TI1FP1=TI1)
- CC2P='0' (TIMx\_CCER 寄存器, TI2FP2不反相, TI2FP2=TI2)
- SMS='011' (TIMx\_SMCR 寄存器, 所有的输入均在上升沿和下降沿有效)
- CEN='1' (TIMx\_CR1寄存器, 计数器使能)

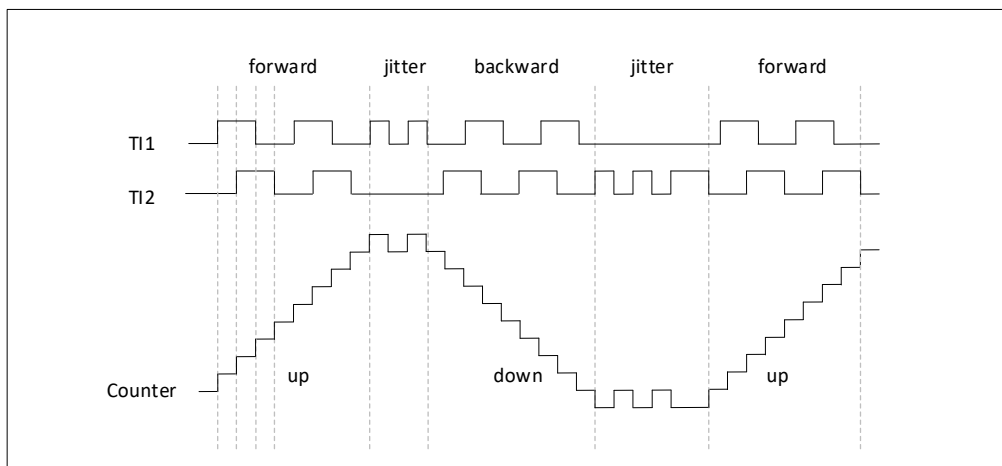


图 21-42 编码器模式下的计数器操作实例

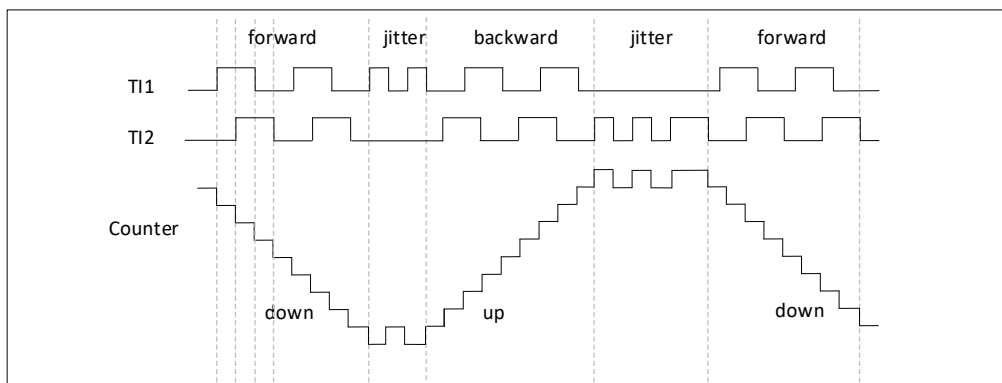


图 21-43 TI1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前的位置信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度、加速度、减速度）。指示机械零点的编码器输出可被用作此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的，并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 21.3.17. 定时器输入异或功能

TIM\_CR2寄存器的 TI1S 位，允许通道1的输入滤波器连接到一个异或门的输出端，异或门的3个输入端为 TIMx\_CH1、TIMx\_CH2和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。

### 21.3.18. 与霍尔传感器的接口

使用高级定时器 (TIM1) 产生 PWM 信号驱动马达时, 可以使用另一个通用 timer (TIM3) 作为“接口定时器”来连接霍尔传感器。3个定时器输入脚 (CC1、CC2、CC3) 通过一个异或门连接到 TI1输入通道 (通过设置 TIMx\_CR2寄存器中的 TI1S 位来选择), “接口定时器”捕获这个信号。

从模式控制器被配置到复位模式, 从输入是 TI1F\_ED。每当3个输入之一变化时, 计数器重新从0开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

接口定时器上的捕获/比较通道1配置为捕获模式, 捕获信号为 TRC (见图21-27)。捕获值反映了两个输入变化间的时间延迟, 给出了马达速度的信息。

接口定时器可以用来在输出模式产生一个脉冲, 这个脉冲可以 (通过触发一个 COM 事件) 用于改变高级定时器 TIM1各个通道的属性, 而高级定时器产生 PWM 信号驱动马达。因此接口定时器通道必须编程为一个指定的延迟 (输出比较或 PWM 模式) 之后产生一个正脉冲, 这个脉冲通过 TRGO 输出被送到高级定时器 TIM1。

举例: 霍尔输入连接到 TIMx 定时器, 要求每次任一霍尔输入上发生变化之后的一个指定的时刻, 改变高级控制定时器 TIMx 的 PWM 配置。

- 置 TIMx\_CR2寄存器的 TI1S 位为'1', 配置三个定时器输入逻辑或到 TI1输入。
- 时基编程: 置 TIMx\_ARR 为其最大值 (计数器必须通过 TI1的变化清零)。设置预分频器得到一个最大的计数器周期, 它长于传感器上的两次变化的时间间隔。
- 设置通道1为捕获模式 (选中 TRC): 置 TIMx\_CCMR1寄存器中 CC1S=01, 如果需要, 还可以设置数字滤波器。
- 设置通道2为 PWM2模式, 并具有要求的延时: 置 TIMx\_CCMR1寄存器中的 OC2M=111和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发输出: 置 TIMx\_CR2寄存器中的 MMS=101。

在高级控制寄存器 TIM1中, 正确的 ITR 输入必须是触发器输入, 定时器被编程为产生 PWM 信号, 捕获/比较控制信号为预装载的 (TIMx\_CR2寄存器中 CCPC=1), 同时触发输入控制 COM 事件 (TIMx\_CR2寄存器中 CCUS=1)。在一次 COM 事件后, 写入下一步的 PWM 控制位 (CCxE、OCxM), 这可以在处理 OC2REF 上升沿的中断子程序里实现。



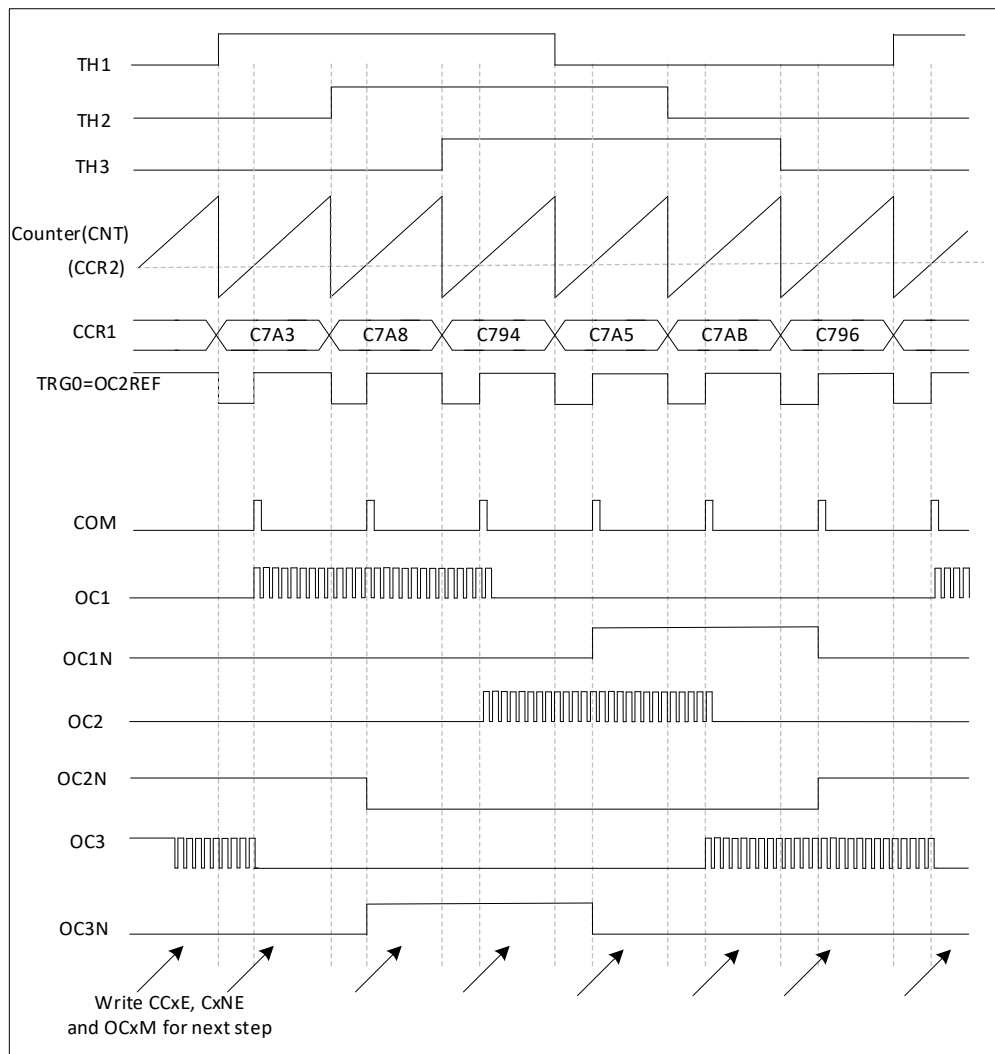


图 21-44 霍尔传感器接口的实例

### 21.3.19. 定时器和外部的触发同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测 TI1 的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志 (TIMx\_SR 寄存器中的 TIF 位) 被设置，根据 TIMx\_DIER 寄存器中 TIE (中断使能) 位和 TDE (DMA 使能) 位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重载寄存器 TIMx\_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

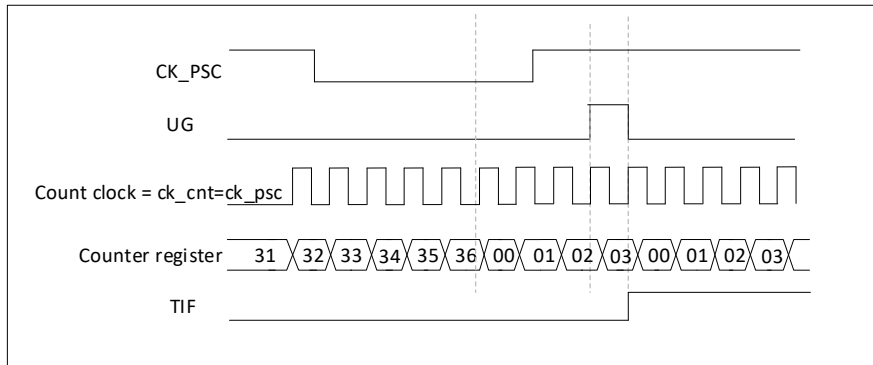


图 21-45 复位模式下的控制电路

### 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1 以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标志。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

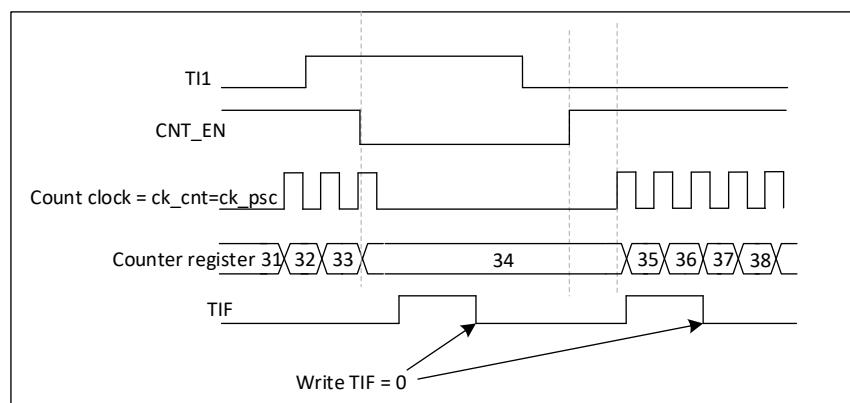


图 21-46 门控模式下的控制电路

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

1. 配置通道2检测 TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC2S=01。置 TIMx\_CCER 寄存器中 CC2P=1以确定极性（只检测低电平）。
2. 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx\_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。TI2上升沿和计数器启动计数之间的延时，取决于 TI2输入端的重同步电路。

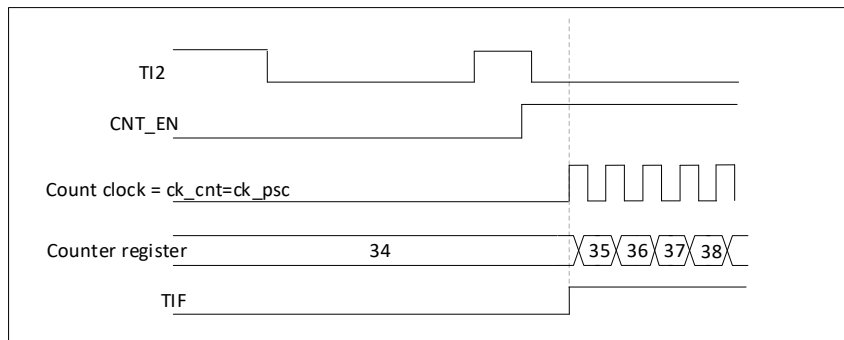


图 21-47 门控模式下的控制电路

#### 从模式：外部时钟模式 2 + 触发模式

外部时钟模式2可以与另一种从模式（外部时钟模式1和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx\_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

- 通过 TIMx\_SMCR 寄存器配置外部触发输入电路：

- ETF=0000：没有滤波
- ETPS=00：不用预分频器
- ETP=0：检测 ETR 的上升沿，置 ECE=1使能外部时钟模式2。

- 按如下配置通道1，检测 TI 的上升沿：

- IC1F=0000：没有滤波
- 触发操作中不使用捕获预分频器，不需要配置
- 置 TIMx\_CCMR1寄存器中 CC1S=01，选择输入捕获源
- 置 TIMx\_CCER 寄存器中 CC1P=0以确定极性（只检测上升沿）

- 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。

当 TI1上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

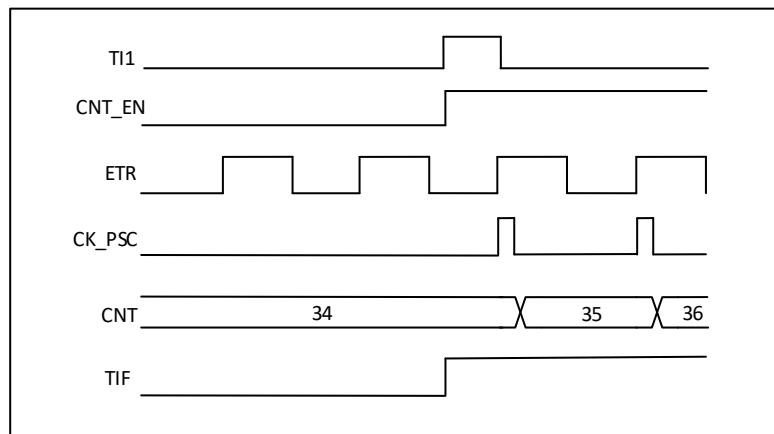


图 21-48 外部时钟模式 2 + 触发模式下的控制电路

### 21.3.20. 定时器同步

TIM 定时器在内部相连，用于 timer 的同步或者链接功能。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止等操作。详情参考22.3.14节定时器同步描述。

### 21.3.21. 调试模式

当芯片进入调试模式时，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器可以继续正常工作或者停止工作。

## 21.4. TIM1寄存器描述

### 21.4.1. TIM1 控制寄存器 1 (TIM1\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD[1: 0]	ARPE	CMS[1: 0]	DIR	OPM	URS	UDIS	CEN		
-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9: 8	CKD[1: 0]	RW	00	时钟分频因子 这2位定义在定时器时钟 (CK_INT) 频率，死区时间和由死区发生器与数字滤波器 (ETR,Tix) 所用的采样时钟之间的分频比例 00: tDTS = tCK_INT

Bit	Name	R/W	Reset Value	Function
				01: $tDTS = 2 \times tCK\_INT$ 10: $tDTS = 4 \times tCK\_INT$ 11: 保留, 不要使用这个配置
7	ARPE	RW	0	自动重载预装载允许位 0: TIMx_ARR 寄存器没有缓冲 1: TIMx_ARR 寄存器被装入缓冲器
6: 5	CMS[1: 0]	RW	00	选择中央对齐模式 00: 边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数。 01: 中央对齐模式1。计数器交替地向上和向下计数。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 只在计数器向下计数时会置位。 10: 中央对齐模式2。计数器交替地向上和向下计数。。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 只在计数器向上计数时会置位。 11: 中央对齐模式3。计数器交替地向上和向下计数。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 在计数器向上和向下计数时会置位。 注: 在计数器开启时 (CEN=1), 不允许从边沿对齐模式转换到中央对齐模式。
4	DIR	RW	0	计数方向 0: 计数器向上计数 1: 计数器向下计数 注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读
3	OPM	RW	0	单脉冲模式 0: 在发生更新事件时, 计数器不停止 1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求

Bit	Name	R/W	Reset Value	Function
1	UDIS	RW	0	<p>禁止更新 (Update disable)</p> <p>软件通过该位允许/禁止 UEV 事件的产生</p> <p>0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>具有缓存的寄存器被装入它们的预装载值。</p> <p>1: 禁止 UEV。不产生更新事件, 影子寄存器 (ARR,PSC,CCRx) 保持它们的值。</p> <p>如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。</p>
0	CEN	RW	0	<p>使能计数器</p> <p>0: 禁止计数器</p> <p>1: 开启计数器</p> <p>注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。</p>

#### 21.4.2. TIM1 控制寄存器 2 (TIM1\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	OIS	OIS3	OIS	OIS2	OIS	OIS1	OIS	T11	MMS[2: 0]			CC	CCU	Re	CCP
s	4	N	3	N	2	N	1	S				DS	S	s	C
-	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	OIS4	RW	0	输出空闲状态4 (OC4输出)。参见 OIS1位。
13	OIS3N	RW	0	输出空闲状态3 (OC3N 输出)。参见 OIS1N 位
12	OIS3	RW	0	输出空闲状态3 (OC3输出)。参见 OIS1位。
11	OIS2N	RW	0	输出空闲状态2 (OC2N 输出)。参见 OIS1N 位。
10	OIS2	RW	0	输出空闲状态2 (OC2输出)。参见 OIS1位
9	OIS1N	RW	0	输出空闲状态1 (OC1N 输出)。

Bit	Name	R/W	Reset Value	Function
				<p>0: 当 MOE=0时, 死区后 OC1N=0</p> <p>1: 当 MOE=0时, 死区后 OC1N=1</p> <p>注: 已经设置了 LOCK (TIMx_BKR 寄存器) 级别1、2或3后, 该位不能被修改。</p>
8	OIS1	RW	0	<p>输出空闲状态1 (OC1输出)。</p> <p>0: 当 MOE=0时, 如果实现了 OC1N, 则死区后 OC1=0</p> <p>1: 当 MOE=0时, 如果实现了 OC1N, 则死区后 OC1=1</p> <p>注: 已经设置了 LOCK (TIMx_BKR 寄存器) 级别1、2或3后, 该位不能被修改。</p>
7	TI1S	RW	0	<p>TI1选择</p> <p>0: TIMx_CH1管脚连到 TI1输入。</p> <p>1: TIMx_CH1、TIMx_CH2和 TIMx_CH3管脚经异或后连到 TI1输入。</p>
6: 4	MMS[2: 0]	RW	000	<p>主模式选择</p> <p>这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下:</p> <p>000: 复位 - TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果触发输入 (复位模式下的从模式控制器) 产生复位, 则 TRGO 上的信号相对实际的复位会有一个延迟。</p> <p>001: 允许 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要在同一时间启动多个定时器或控制从定时器的一个窗口。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时, TRGO 上会有一个延迟, 除非选择了主/从模式 (见 TIMx_SMCR 寄存器中 MSM 位的描述)。</p> <p>010: 更新 - 更新事件被选为触发输入 (TRGO)。例如, 一个主定时器的时钟可以被用作一个从定时器的预分频器。</p> <p>011: 比较脉冲 - 一旦发生一次捕获或一次比较成功时, 当要设置 CC1IF 标志时 (即是它已经为高), 触发输出送出一个正脉冲 (TRGO)。</p> <p>100: 比较 - OC1REF 信号被用于作为触发输出 (TRGO)。</p> <p>101: 比较 - OC2REF 信号被用于作为触发输出 (TRGO)。</p> <p>110: 比较 - OC3REF 信号被用于作为触发输出 (TRGO)。</p>

Bit	Name	R/W	Reset Value	Function
				111: 比较 - OC4REF 信号被用于作为触发输出 (TRGO)。 注意: 1.从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号, 并在接收时不要改变。 2.若主从定时器不在同一总线上, 主模式应该配置为能被从定时器采到的宽度。
3	CCDS	RW	0	捕获/比较的 DMA 选择 0: 当发生 CCx 事件时, 送出 CCx 的 DMA 请求。 1: 当发生更新事件时, 送出 CCx 的 DMA 请求。
2	CCUS	RW	0	捕获/比较控制更新选择 0: 如果捕获/比较控制位是预装载的 (CCPC=1), 只能通过设置 COM 位更新它们。 1: 如果捕获/比较控制位是预装载的 (CCPC=1), 可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注: 该位只对具有互补输出的通道起作用。
1	Res	-	0	保留, 始终读为0。
0	CCPC	RW	0	捕获/比较预装载控制位 0: CCxE, CCxNE 和 OCxM 位不是预装载的。 1: CCxE, CCxNE 和 OCxM 位是预装载的; 设置该位后, 它们只在设置了 COM 位后被更新。 注: 该位只对具有互补输出的通道起作用。

### 21.4.3. TIM1 从模式控制寄存器 (TIM1\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
ETP	ECE	ETPS[1: 0]		ETF[3: 0]			MSM	TS[2: 0]		OCCS	SMS[2: 0]				
RW	RW	RW		RW			RW	RW		RW	RW				

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	ETP	RW	0	外部触发极性。该位选择是否 ETR 或者 ETR 的反向被用作触发操作。 0: ETR 不进行反向, 高电平或者上升沿有效



Bit	Name	R/W	Reset Value	Function
				1: ETR 反向, 低电平或者下降沿有效
14	ECE	RW	0	<p>外部时钟使能位。该位启用外部时钟模式2</p> <p>0: 禁止外部时钟模式2;</p> <p>1: 使能外部时钟模式2。计数器由 ETRF 信号上的任意有效边沿驱动。</p> <p>注1: 设置 ECE 位与选择外部时钟模式1并将 TRGI 连到 ETRF (SMS=111和 TS=111) 具有相同功效。</p> <p>注2: 下述从模式可以与外部时钟模式2同时使用: 复位模式, 门控模式和触发模式; 但是, 这时 TRGI 不能连到 ETRF (TS 位不能是 '111' )。</p> <p>注3: 外部时钟模式1和外部时钟模式2同时被使能时, 外部时钟的输入是 ETRF。</p>
13: 12	ETPS[1: 0]	RW	00	<p>外部触发预分频器。外部触发输入信号 ETR 频率必须至多是 TIM1CLK 频率的1/4。可以使能预分频器, 以降低 ETRP 的频率。</p> <p>00: 预分频器关闭</p> <p>01: ETRP 频率的2分频</p> <p>10: ETRP 频率的4分频</p> <p>11: ETRP 频率的8分频</p>
11: 8	ETF[3: 0]	RW	0000	<p>外部触发滤波。这些位定义采样 ETRP 信号的频率和应用在 ETRP 的数字滤波长度。这个数字滤波由一个事件计数器组成, 在改计数器里, N 个连续的事件被需要使输出的边沿有效。</p> <p>0000: 没有滤波器, 在 fDTS 下采样</p> <p>0001: <math>f_{\text{SAMPLING}}=f_{\text{CK\_INT}}, N=2</math></p> <p>0010: <math>f_{\text{SAMPLING}}=f_{\text{CK\_INT}}, N=4</math></p> <p>0011: <math>f_{\text{SAMPLING}}=f_{\text{CK\_INT}}, N=8</math></p> <p>0100: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/2}, N=6</math></p> <p>0101: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/2}, N=8</math></p> <p>0110: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/4}, N=6</math></p> <p>0111: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/4}, N=8</math></p> <p>1000: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/8}, N=6</math></p> <p>1001: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/8}, N=8</math></p> <p>1010: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/16}, N=5</math></p> <p>1011: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/16}, N=6</math></p> <p>1100: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/16}, N=8</math></p> <p>1101: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/32}, N=5</math></p> <p>1110: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/32}, N=6</math></p> <p>1111: <math>f_{\text{SAMPLING}}=f_{\text{DTS}/32}, N=8</math></p>
7	MSM	RW	0	主/从模式

Bit	Name	R/W	Reset Value	Function
				<p>0: 无作用</p> <p>1: 触发输入 (TRGI) 上的事件被延迟了, 以允许在当前定时器 (通过 TRGO) 与它的当前定时器和从定时器间的同步 (通过 TRGO)。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的</p>
6: 4	TS[2: 0]	RW	000	<p>触发选择, 这3位选择用于同步计数器的触发输入。</p> <p>000: TIM15 (ITR0)</p> <p>001: TIM2 (ITR1)</p> <p>010: TIM3 (ITR2)</p> <p>011: TIM17 (ITR3)</p> <p>100: TI1的边沿检测器 (TI1F_ED)</p> <p>101: 滤波后的定时器输入1 (TI1FP1)</p> <p>110: 滤波后的定时器输入2 (TI2FP2)</p> <p>111: 外部触发输入 (ETRF)</p> <p>注: 为避免在信号转变时产生错误的边沿检测, 必须在未使用这些位 (比如 SMS=000) 时修改它们</p>
3	OCCS	RW	0	<p>OCCS 清除选择位。该位用于选择 OCCS 的清除源。</p> <p>0: OCCS_CLR_INT 连接到 OCCS_CLR 输入</p> <p>1: OCCS_CLR_INT 连接到 ETRF</p>
2: 0	SMS[2: 0]	RW	000	<p>从模式选择。当选择了外部信号, 触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 (见输入控制寄存器和控制寄存器的说明)</p> <p>000: 关闭从模式</p> <p>如果 CEN=1, 则预分频器直接由内部时钟驱动。</p> <p>001: 编码器模式1</p> <p>根据 TI1FP1的电平, 计数器在 TI2FP2的边沿向上/下计数。</p> <p>010: 编码器模式2</p> <p>根据 TI2FP2的电平, 计数器在 TI1FP1的边沿向上/下计数。</p> <p>011: 编码器模式3</p> <p>编码器模式1和模式2的综合</p> <p>100: 复位模式</p> <p>选中的触发输入 (TRGI) 的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。</p> <p>101: 门控模式</p> <p>当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的。</p> <p>110: 触发模式</p>

Bit	Name	R/W	Reset Value	Function
				<p>计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。</p> <p>111：外部时钟模式1</p> <p>选中的触发输入（TRGI）的上升沿驱动计数器。</p> <p>注：如果 TI1F_ED 被选为触发输入（TS=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。</p> <p>注：在编码器模式下，不要使用 UEV 作为 TRGO 输出信号，（即 MMS 不能配置为010）</p>

## TIM1 内部触发连接

Slave TIM	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM1	TIM15_TRGO	TIM2_TRGO	TIM3_TRGO	TIM17_OC

## 21.4.4. TIM1 DMA/中断使能寄存器 (TIM1\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	TD	COM	CC4D	CC3D	CC2D	CC1D	UD	BI	TI	COMI	CC4I	CC3I	CC2I	CC1I	UI
s	E	DE	E	E	E	E	E	E	E	E	E	E	E	E	E
RW															

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	TDE	RW	0	<p>TDE: 允许触发 DMA 请求</p> <p>0: 禁止触发 DMA 请求</p> <p>1: 允许触发 DMA 请求</p>
13	COMDE	RW	0	<p>COMDE: 允许 COM 的 DMA 请求</p> <p>0: 禁止 COM 的 DMA 请求</p> <p>1: 允许 COM 的 DMA 请求</p>
12	CC4DE	RW	0	<p>CC4DE: 允许捕获/比较4的 DMA 请求</p> <p>0: 禁止捕获/比较4的 DMA 请求</p> <p>1: 允许捕获/比较4的 DMA 请求</p>
11	CC3DE	RW	0	<p>CC3DE: 允许捕获/比较3的 DMA 请求</p> <p>0: 禁止捕获/比较3的 DMA 请求</p>

Bit	Name	R/W	Reset Value	Function
				1: 允许捕获/比较3的 DMA 请求
10	CC2DE	RW	0	CC2DE: 允许捕获/比较2的 DMA 请求 0: 禁止捕获/比较2的 DMA 请求 1: 允许捕获/比较2的 DMA 请求
9	CC1DE	RW	0	CC1DE: 允许捕获/比较1的 DMA 请求 0: 禁止捕获/比较1的 DMA 请求 1: 允许捕获/比较1的 DMA 请求
8	UDE	RW	0	UDE: 允许更新的 DMA 请求 0: 禁止更新的 DMA 请求 1: 允许更新的 DMA 请求
7	BIE	RW	0	BIE: 允许刹车中断 0: 禁止刹车中断 1: 允许刹车中断
6	TIE	RW	0	TIE: 允许触发中断 0: 禁止触发中断 1: 允许触发中断
5	COMIE	RW	0	COMIE: 允许 COM 中断 0: 禁止 COM 中断 1: 允许 COM 中断
4	CC4IE	RW	0	CC4IE: 允许捕获/比较4中断 0: 禁止捕获/比较4中断 1: 允许捕获/比较4中断
3	CC3IE	RW	0	CC3IE: 允许捕获/比较3中断 0: 禁止捕获/比较3中断 1: 允许捕获/比较3中断
2	CC2IE	RW	0	CC2IE: 允许捕获/比较2中断 0: 禁止捕获/比较2中断 1: 允许捕获/比较2中断
1	CC1IE	RW	0	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断 1: 允许捕获/比较1中断
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

#### 21.4.5. TIM1 状态寄存器 (TIM1\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			Res					ic4i	ic3i	ic2if	ic1if	ic4i	ic3ir	ic2ir	ic1i
								RC_W0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4O	CC3O	CC2O	CC1O	Re	BI	TIF	COMI	CC4I	CC	CC2	CC1I	UI
			F	F	F	F	s	F		F	F	3IF	IF	F	F
-			RC_W0				-		RC_W0						

Bit	Name	R/W	Reset Value	Function
31: 13	保留	-	-	保留
23	IC4IF	RC_W0	0	下降沿捕获4标志 参见 IC1IF 描述。
22	IC3IF	RC_W0	0	下降沿捕获3标志 参见 IC1IF 描述。
21	IC2IF	RC_W0	0	下降沿捕获2标志 参见 IC1IF 描述。
20	IC1IF	RC_W0	0	下降沿捕获1标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件，该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0'。 0: 无下降沿捕获产生； 1: 发生下降沿捕获事件。
19	IC4IR	RC_W0	0	上升沿捕获4标志 参见 IC1IR 描述。
18	IC3IR	RC_W0	0	上升沿捕获3标志 参见 IC1IR 描述。
17	IC2IR	RC_W0	0	上升沿捕获2标志 参见 IC1IR 描述。
16	IC1IR	RC_W0	0	上升沿捕获1标志 仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0'。 0: 无上升沿捕获产生； 1: 发生上升沿捕获事件。
15: 13	保留	-	-	保留
12	CC4OF	RC_W0	0	捕获/比较4 过捕获标记 参见 CC1OF 描述
11	CC3OF	RC_W0	0	捕获/比较3 过捕获标记 参见 CC1OF 描述

Bit	Name	R/W	Reset Value	Function
10	CC2OF	RC_W0	0	捕获/比较2 过捕获标记 参见 CC1OF 描述
9	CC1OF	RC_W0	0	捕获/比较1 过捕获标记 仅当相应的通道被配置为输入捕获时, 该标记可由硬件置1。写0可清除该位。 0: 无过捕获产生; 1: CC1OF 置1时, 当 CC1IF 为1时计数器的值被捕获到 TIMx_CCR1寄存器。
8	保留	-	-	保留
7	BIF	RC_W0	0	刹车中断标记 一旦刹车输入有效, 由硬件对该位置1。如果刹车输入无效, 则该位可由软件清0。 0: 无刹车事件产生; 1: 刹车输入上检测到有效电平。
6	TIF	RC_W0	0	触发器中断标记 当发生触发事件 (当从模式控制器处于除门控模式外的其它模式时,在 TRGI 输入端检测到有效边沿, 或或门控模式下的任一边沿) 时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生; 1: 触发器中断等待响应
5	COMIF	RC_W0	0	COM 中断标记 一旦产生 COM 事件 (当 CCxE、CCxNE、OCxM 已被更新) 该位由硬件置1。它由软件清0。 0: 无 COM 事件产生; 1: COM 中断等待响应
4	CC4IF	RC_W0	0	捕获/比较4 中断标记 参考 CC1IF 描述
3	CC3IF	RC_W0	0	捕获/比较3 中断标记 参考 CC1IF 描述
2	CC2IF	RC_W0	0	捕获/比较2 中断标记 参考 CC1IF 描述
1	CC1IF	RC_W0	0	捕获/比较1 中断标记 如果通道 CC1配置为输出模式: 当计数器值与比较值匹配时该位由硬件置1。 但在中心对称模式下需要判断 CMS; 当比较值大于重载值时, 会在上溢 (向上计数模式、中央对齐计数模式) 或者下溢 (向下计数模式) 时置1。它由软件清0。 0: 无匹配发生;

Bit	Name	R/W	Reset Value	Function
				1: TIMx_CNT 的值与 TIMx_CCR1的值匹配。 如果通道 CC1配置为输入模式： 当捕获事件发生时该位由硬件置1，它由软件清0或通过读 TIMx_CCR1清0。 0: 无输入捕获产生； 1: 输入捕获产生并且计数器值已装入 TIMx_CCR1 (在 IC1上检测到与所选极性相同的边沿)。 注：当 CEN 打开，该位也会被置位。
0	UIF	RC_W0	0	更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0: 无更新事件产生； 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1。 - 若 TIMx_CR1寄存器的 UDIS=0，当 REP_CNT=0时产生更新事件（重复向下计数器上溢或下溢时）； - 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1时产生更新事件（软件对 CNT 重新初始化）； - 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 CNT 被触发事件重初始化时产生更新事件。（参考从模式控制寄存器（TIMx_SMCR））

#### 21.4.6. TIM1 事件产生寄存器 (TIM1\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	-	-	-	-	-	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7	BG	W	0	产生刹车事件 该位由软件置1，用于产生一个刹车事件，由硬件自动清0。 0: 无动作；

Bit	Name	R/W	Reset Value	Function
				1: 产生一个刹车事件。此时 MOE=0、BIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
6	TG	W	0	产生触发事件 该位由软件置1, 用于产生一个触发事件, 由硬件自动清0。 0: 无动作; 1: TIMx_SR 寄存器的 TIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
5	COMG	W	0	捕获/比较事件, 产生控制更新 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 当 CCPC=1, 允许更新 CCxE、CCxNE、OCxM 位。 注: 该位只对有互补输出的通道有效。
4	CC4G	W	0	产生捕获/比较4事件 参考 CC1G 描述
3	CC3G	W	0	产生捕获/比较3事件 参考 CC1G 描述
2	CC2G	W	0	产生捕获/比较2事件 参考 CC1G 描述
1	CC1G	W	0	产生捕获/比较1事件 该位由软件置1, 用于产生一个捕获/比较事件, 由硬件自动清0。 0: 无动作; 1: 在通道 CC1上产生一个捕获/比较事件: 若通道 CC1配置为输出: 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。 若通道 CC1配置为输入: 当前的计数器值捕获至 TIMx_CCR1寄存器, 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。若 CC1IF 已经为1, 则设置 CC1OF=1。
0	UG	W	0	产生更新事件。该位由软件置1, 硬件自动清0。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意: 预分频器的计数器也被清0 (但是预分频系数不变)。若在中心对称模式下或 DIR=0 (向上计数) 则计数器被清0, 若 DIR=1 (向下计数) 则计数器装载 TIMx_ARR 的值。



### 21.4.7. TIM1 捕获/比较模式寄存器 1 (TIM1\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

输出比较模式:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2: 0]			OC2PE	CO2FE	CC2S[1: 0]		OC1CE	OC1M[2: 0]			OC1PE	OC1FE	CC1S[1: 0]	
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	OC2CE	RW	0	输出比较2清0使能
14: 12	OC2M[2: 0]	RW	000	输出比较2模式选择
11	OC2PE	RW	0	输出比较2预装载使能
10	OC2FE	RW	0	输出比较2快速使能
9: 8	CC2S[1: 0]	RW	00	捕获/比较2选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00: CC2通道被配置为输出； 01: CC2通道被配置为输入，IC2映射在 TI2上； 10: CC2通道被配置为输入，IC2映射在 TI1上； 11: CC2通道被配置为输入，IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIMx_SMCR 寄存器的 TS 位选择）。 注: CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E=0）才是可写的。
7	OC1CE	RW	0	输出比较1清0使能 0: OC1REF 不受 ETRF 输入的影响； 1: 一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
6: 4	OC1M[2: 0]	RW	00	输出比较1模式 该3位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000: 冻结。输出比较寄存器 TIMx_CCR1与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用；

Bit	Name	R/W	Reset Value	Function
				<p>001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时, 强制 OC1REF 为高。</p> <p>010 : 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时, 强制 OC1REF 为低。</p> <p>011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。</p> <p>100: 强制为无效电平。强制 OC1REF 为低。</p> <p>101: 强制为有效电平。强制 OC1REF 为高。</p> <p>110: PWM 模式1 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为有效电平, 否则为无效电平; 在向下计数时, 一旦 TIMx_CNT&gt;TIMx_CCR1时通道 1 为无效电平 (OC1REF=0), 否则为有效电平 (OC1REF=1)。</p> <p>111: PWM 模式2 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为无效电平, 否则为有效电平; 在向下计数时, 一旦 TIMx_CNT&gt;TIMx_CCR1时通道1为有效电平, 否则为无效电平。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 在 PWM 模式1或 PWM 模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较1预装载使能</p> <p>0: 禁止 TIMx_CCR1寄存器的预装载功能, 可随时写入 TIMx_CCR1寄存器, 且新值马上起作用。</p> <p>1: 开启 TIMx_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p>
2	OC1FE	RW	0	<p>输出比较1 快速使能</p> <p>该位用于加快 CC 输出对触发器输入事件的响应。</p>

Bit	Name	R/W	Reset Value	Function
				<p>0: 根据计数器与 CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活 CC1输出的最小延时为5个时钟周期。</p> <p>1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1输出间的延时被缩短为3个时钟周期。</p> <p>OCFE 的只在通道被配置成 PWM1或 PWM2模式时起作用。</p>
1: 0	CC1S[1: 0]	RW	00	<p>捕获/比较1 选择。</p> <p>这2位定义通道的方向 (输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC1映射在 TI1上;</p> <p>10: CC1通道被配置为输入, IC1映射在 TI2上;</p> <p>11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。</p> <p>注: CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E=0) 才是可写的。</p>

**输入捕获模式:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3: 0]				IC2PSC[1: 0]		CC2S[1: 0]		IC1F[3: 0]				IC1PSC[1: 0]		CC1S[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 12	IC2F	RW	0000	输入捕获2滤波器
11: 10	IC2PSC[1: 0]	RW	00	捕获/比较2预分频器
9: 8	CC2S[1: 0]	RW	0	<p>捕获/比较2选择。</p> <p>这2位定义通道的方向 (输入/输出), 及输入脚的选择:</p> <p>00: CC2通道被配置为输出;</p> <p>01: CC2通道被配置为输入, IC2映射在 TI2上;</p> <p>10: CC2通道被配置为输入, IC2映射在 TI1上;</p>

Bit	Name	R/W	Reset Value	Function
				11: CC2通道被配置为输入, IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E=0) 才是可写的。
7: 4	IC1F[3: 0]	RW	0000	输入捕获1滤波器 这几位定义了 TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成, 它记录到 N 个事件后会产生一个输出的跳变: 0000: 没有滤波器, 在 fDTS 下采样 0001: fSAMPLING=fCK_INT, N=2 0010: fSAMPLING=fCK_INT, N=4 0011: fSAMPLING=fCK_INT, N=8 0100: fSAMPLING=fDTS/2, N=6 0101: fSAMPLING=fDTS/2, N=8 0110: fSAMPLING=fDTS/4, N=6 0111: fSAMPLING=fDTS/4, N=8 1000: fSAMPLING=fDTS/8, N=6 1001: fSAMPLING=fDTS/8, N=8 1010: fSAMPLING=fDTS/16, N=5 1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	捕获/比较1预分频器 00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获; 01: 每2个事件触发一次捕获; 10: 每4个事件触发一次捕获; 11: 每8个事件触发一次捕获。
1: 0	CC1S[1: 0]	RW	00	CC1S[1: 0]: 捕获/比较1选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC1映射在 TI1上; 10: CC1通道被配置为输入, IC1映射在 TI2上; 11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。

Bit	Name	R/W	Reset Value	Function
				注: CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E=0) 才是可写的。

### 21.4.8. TIM1 捕获/比较模式寄存器 2 (TIM1\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

输出比较模式:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4C E	OC4M[2: 0]			OC4P E	CO4F E	CC4S[1 : 0]		OC3C E	OC3M[2: 0]			OC3P E	OC 3FE	CC3S[1: 0]	
IC4F[3: 0]				IC4PSC[1: 0]			IC3F[3: 0]			IC3PSC[1: 0]					
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	OC4CE	RW	0	输出比较4清0使能
14: 12	OC4M[2: 0]	RW	000	输出比较4模式
11	OC4PE	RW	0	输出比较4预装载使能
10	OC4FE	RW	0	输出比较4快速使能
9: 8	CC4S[1: 0]	RW	00	捕获/比较4选择。 该位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC4通道被配置为输出; 01: CC4通道被配置为输入, IC4映射在 TI4上; 10: CC4通道被配置为输入, IC4映射在 TI3上; 11: CC4通道被配置为输入, IC4映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E=0) 才是可写的。
7	OC3CE	RW	0	输出比较3清0使能
6: 4	OC3M[2: 0]	RW	00	输出比较3模式
3	OC3PE	RW	0	输出比较3预装载使能
2	OC3FE	RW	0	输出比较3快速使能
1: 0	CC3S[1: 0]	RW	00	捕获/比较3选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择:

Bit	Name	R/W	Reset Value	Function
				00: CC3通道被配置为输出; 01: CC3通道被配置为输入, IC3映射在 TI3上; 10: CC3通道被配置为输入, IC3映射在 TI4上; 11: CC3通道被配置为输入, IC3映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E=0) 才是可写的。

**输入捕获模式:**

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 12	IC4F[3: 0]	RW	0000	输入捕获4滤波器
11: 10	IC4PSC[1: 0]	RW	00	捕获/比较4预分频器
9: 8	CC4S[1: 0]	RW	00	捕获/比较4选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC4通道被配置为输出; 01: CC4通道被配置为输入, IC4映射在 TI4上; 10: CC4通道被配置为输入, IC4映射在 TI3上; 11: CC4通道被配置为输入, IC4映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E=0) 才是可写的。
7: 4	IC3F[3: 0]	RW	0000	捕获/比较3滤波器
3: 2	IC3PSC[1: 0]	RW	00	捕获/比较3预分频器
1: 0	CC3S[1: 0]	RW	00	捕获/比较3选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC3通道被配置为输出; 01: CC3通道被配置为输入, IC3映射在 TI3上; 10: CC3通道被配置为输入, IC3映射在 TI4上; 11: CC3通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E=0) 才是可写的。

**21.4.9. TIM1 捕获/比较使能寄存器 (TIM1\_CCER)**

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re s	Re s	CC4 P	CC4 E	CC3 NP	CC3 NE	CC3 P	CC3 E	CC2 NP	CC2 NE	CC2 P	CC2 E	CC1 NP	CC1 NE	CC1 P	CC1 E
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 14	保留	-	0	保留, 一直为0
13	CC4P	RW	0	捕获/比较4输出极性。参考 CC1P 的描述。
12	CC4E	RW	0	捕获/比较4输出使能。参考 CC1E 的描述。
11	CC3NP	RW	0	捕获/比较3互补输出极性。参考 CC1NP 的描述。
10	CC3NE	RW	0	捕获/比较3互补输出使能。参考 CC1NE 的描述。
9	CC3P	RW	0	捕获/比较3输出极性。参考 CC1P 的描述。
8	CC3E	RW	0	捕获/比较3输出使能。参考 CC1E 的描述。
7	CC2NP	RW	0	捕获/比较2互补输出极性。参考 CC1NP 的描述。
6	CC2NE	RW	0	捕获/比较2互补输出使能。参考 CC1NE 的描述。
5	CC2P	RW	0	捕获/比较2输出极性。参考 CC1P 的描述。
4	CC2E	RW	0	捕获/比较2输出使能。参考 CC1E 的描述。
3	CC1NP	RW	0	捕获/比较1互补输出极性 0: OC1N 高电平有效 1: OC1N 低电平有效 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LCKK 位) 设为3或2且 CC1S=00 (通道配置为输出) 则该位不能被修改。
2	CC1NE	RW	0	捕获/比较1互补输出使能 0: OC1N 禁止输出 1: OC1N 信号输出到对应的输出引脚  当 CC1通道配置为输出时, OC1N 输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定, 见下表  对于互补输出通道, 这一位是预载的。如果 TIMx_CR2 寄存器中的 CCPC 位被设置, 那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。
1	CC1P	RW	0	捕获/比较1输出极性 CC1通道配置为输出:

Bit	Name	R/W	Reset Value	Function
				<p>0: OC1高电平有效 1: OC1低电平有效</p> <p>CC1通道配置为输入: CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1和 TI2FP1的极性。</p> <p>00: 不反相/上升沿: TIxFP1上升沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ; TIxFP1不反相 (门控模式、编码器模式) 。</p> <p>01: 反相/下降沿: TIxFP1下降沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ; TIxFP1反相 (门控模式、编码器模式) 。</p> <p>10: 保留, 不要使用这个配置。</p> <p>11: 不反相/双沿 TIxFP1上升和下降沿都有效 (捕获、复位模式下触发、外部时钟或触发模式下) ; TIxFP1不反相 (门控模式) 。这个配置不能应用于编码器模式下。</p> <p>注: 1.对于互补输出通道, 这一位是预载的。如果 TIMx_CR2寄存器中的 CCPC 位被设置, 那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。 2.一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LCCK 位) 设为3或2, 则该位不能被修改</p>
0	CC1E	RW	0	<p>捕获/比较1输出使能</p> <p>0: 捕获模式关闭/OC1禁止输出 1: 捕获模式开启/OC1信号输出到对应的输出引脚</p> <p>当 CC1通道配置为输出时, OC1输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定, 见下表</p> <p>注: 对于互补输出通道, 这一位是预载的。如果 TIMx_CR2寄存器中的 CCPC 位被设置, 那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。</p>



表21-2具有中断功能的互补 OCx 和 OCxN 通道的输出控制

控制位					输出状态	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	x	0	0	0	输出禁止 (与定时器断开) OCx=0, OCx_EN=0	输出禁止 (与定时器断开) OCxN=0, OCxN_EN=0
		0	0	1	输出禁止 (与定时器断开) OCx=0, OCx_EN=0	OCxREF+极性, OCxN=OCREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF+极性, OCx=OCREF xor CCxP, OCx_EN=1	输出禁止 (与定时器断开) OCxN=0, OCxN_EN=0
		0	1	1	OCxREF+极性+死区, , OCx_EN=1	OCxREF+极性+死区, OCxN_EN=1
		1	0	0	输出禁止 (与定时器断开) OCx=CCxP, OCx_EN=0	输出禁止 (与定时器断开) OCxN=CCxNP, OCxN_EN=0
		1	0	1	关闭状态 (输出使能且为无效电 平) OCx=CCxP, OCx_EN=1	OCxREF+极性, OCxN=OCREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+极性, OCx=OCREF xor CCxP, OCx_EN=1	关闭状态 (输出使能且为无效电平) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF+极性+死区, , OCx_EN=1	OCxREF+极性+死区, OCxN_EN=1
0	x	0	0	0	输出禁止 (与定时器断开)	
		0	0	1		
		0	1	0		
		0	1	1		
		1	0	0	关闭状态 (输出使能且为无效电平) 异步的: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCx_EN=1; 若时钟存在: 经过一个死区时间后 OCx=OISx, OCxN=OISxN, 假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平	
		1	0	1		
		1	1	0		
		1	1	1		

### 21.4.10. TIM1 计数器 (TIM1\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CNT[15: 0]	RW	0	计数器的值

### 21.4.11. TIM1 预分频器 (TIM1\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PSC[15: 0]	RW	0	预分频器的值 计数器的时钟频率 (CK_CNT) 等于 fCK_PSC/ ( PSC[15: 0]+1) 。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的 值; 更新事件包括计数器 被 TIM_EGR 的 UG 位清0或被工作在复位模式的从控制器 清0。

### 21.4.12. TIM1 自动重载寄存器 (TIM1\_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Res	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															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ARR[15: 0]	RW	0xFFFF	自动重载的值 ARR 包含了将要装载入实际的自动重载寄存器的值。 当自动重载的值为空时，计数器不工作。

### 21.4.13. TIM1 重复计数器寄存器 (TIM1\_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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	REP[7: 0]							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7: 0	REP[7: 0]	RW	0	周期计数器的值 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1) 对应着： <ul style="list-style-type: none"> <li>- 在边沿对齐模式下，PWM 周期的数目；</li> <li>- 在中心对称模式下，PWM 半周期的数目；</li> </ul>

### 21.4.14. TIM1 捕获/比较寄存器 1 (TIM1\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
CCR1[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CCR1[15: 0]	RW	0	<p>捕获/比较1的值</p> <p>若 CC1通道配置为输出： CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC1PE 位）中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较1寄存器中。</p> <p>当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1端口上产生输出信号。</p> <p>若 CC1通道配置为输入： CCR1包含了由上一次输入捕获1事件（IC1）传输的计数器值。</p>

### 21.4.15. TIM1 捕捉/比较寄存器 2 (TIM1\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
CCR2[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留

15: 0	CCR2[15: 0]	RW	0	<p>捕获/比较通道2的值</p> <p>若 CC2通道配置为输出： CCR2包含了装入当前捕获/比较2寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC2PE 位）中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较2寄存器中。</p> <p>当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2端口上产生输出信号。</p> <p>若 CC2通道配置为输入： CCR2包含了由上一次输入捕获2事件（IC2）传输的计数器值。</p>
-------	-------------	----	---	--

### 21.4.16. TIM1 捕获/比较寄存器 3 (TIM1\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
CCR3[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CCR3[15: 0]	RW	0	<p>捕获/比较3的值</p> <p>若 CC3通道配置为输出： CCR3包含了装入当前捕获/比较3寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR2寄存器（OC3PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较3寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC 端口上输出信号。</p> <p>若 CC3通道配置为输入： CCR3包含了由上一次输入捕获3事件（IC3）传输的计数器值。</p>

**21.4.17. TIM1 捕捉/比较寄存器 4 (TIM1\_CCR4)****Address offset:** 0x40**Reset value:** 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
CCR4[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CCR4[15: 0]	RW	0	<p>捕获/比较4的值</p> <p>若 CC4通道配置为输出： CCR4包含了装入当前捕获/比较4寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR2寄存器（OC4PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较4寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC 端口上输出信号。</p> <p>若 CC4通道配置为输入： CCR4包含了由上一次输入捕获4事件（IC4）传输的计数器值。</p>

**21.4.18. TIM1 刹车和死区寄存器 (TIM1\_BDTR)****Address offset:** 0x44**Reset value:** 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1: 0]		DTG[7: 0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	MOE	RW	0	<p>主输出使能</p> <p>一旦刹车输入有效, 该位被硬件异步清0。根据 AOE 位的值, 可由软件清0或自动置1。它仅对配置为输出通道有效。</p> <p>0: 禁止 OC 和 OCN 输出或强制为空闲状态;</p> <p>1: 如果设置了相应的使能位 (TIMx_CCER 寄存器的 CcxE、CcxNE 位), 则开启 OC 和 OCN 输出。</p>
14	AOE	RW	0	<p>自动输出使能</p> <p>0: MOE 只能被软件置1;</p> <p>1: MOE 能被软件置1或在下一个更新事件自动置1 (如果刹车输入无效)。</p> <p>注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。</p>
13	BKP	RW	0	<p>刹车输入极性</p> <p>0: 刹车输入低电平有效;</p> <p>1: 刹车输入高电平有效。</p> <p>注:</p> <p>1、一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。</p> <p>2、任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。</p>
12	BKE	RW	0	<p>刹车功能使能</p> <p>0: 禁止刹车功能);</p> <p>1: 开启刹车功能。</p> <p>注:</p> <p>1、一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。</p> <p>2、任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。</p>
11	OSSR	RW	0	<p>运行模式下“关闭状态”选择</p> <p>该位用于当 MOE=1且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。</p> <p>参考 OC/OCN 使能的详细说明 (捕获/比较使能寄存器 (TIMx_CCER) )。</p> <p>0: 禁止 OC/OCN 输出 (释放对 GPIO 的控制);</p> <p>1: 一旦 CCxE=1或 CCxNE=1, 开启 OC/OCN 输出并输出无效电平。</p> <p>OC/OCN 使能输出信号=1。</p>

Bit	Name	R/W	Reset Value	Function
				注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下“关闭状态”选择 该位用于当 MOE=0且通道设为输出时。 参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器（TIMx_CCER））。 0：禁止 OC/OCN 输出（释放对 GPIO 的控制）； 1：，一旦 CCxE=1或 CCxNE=1，OC/OCN 输出 其 空闲电 平。 注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为2，则该位不能被修改。
9: 8	LOCK[1: 0]	RW	00	锁定设置 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别1，不能写入 TIMx_BDTR 寄存器的 DTG/BKE/BKP/AOE 位、TIMx_CR2寄存器的 OISx/OISxN 位； 10：锁定级别2，不能写入锁定级别1中的各位，也不能写入 CC 极性位（一旦相关通道通过 CCxS 位设为输出，TIMx_CCER 寄存器的 CCxP/CCNxP 位）以及 OSSR/OSSI 位； 11：锁定级别3，不能写入锁定级别2中的各位，也不能写入 CC 控制位（一旦相关通道通过 CCxS 位设为输出，TIMx_CCMRx 寄存器的 OCxM/OCxPE 位）； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。
7: 0	DTG[7: 0]	RW	0000 0000	死区发生器设置 这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： DTG[7: 5]=0xx => DT=DTG[7: 0] × Tdtg, Tdtg = TDTs; DTG[7: 5]=10x => DT= (64+DTG[5: 0]) × Tdtg, Tdtg = 2 × TDTs; DTG[7: 5]=110 => DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 8 × TDTs; DTG[7: 5]=111 => DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 16 × TDTs; 例：若 TDTs = 125ns (8MHZ)，可能的死区时间为：0到15875ns，若步长时间为125ns；



Bit	Name	R/W	Reset Value	Function
				16us 到31750ns, 若步长时间为250ns; 32us 到63us, 若步长时间为1us; 64us 到126us, 若步长时间为2us; 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1、2或3, 则这些位不能被修改。

#### 21.4.19. TIM1 DMA 控制寄存器 (TIM1\_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 13	保留	-	-	保留
12: 8	DBL[4: 0]	RW	0 0000	DMA 连续传送长度 这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器的地址进行读或写时, 定时器则进行一次连续传送), 即: 定义被传送的次数: 00000: 1次传输 00001: 2次传输 00010: 3次传输 ..... ..... 10001: 18次传输
7: 5	保留	RW	0	保留, 始终读为0
4: 0	DBA[4: 0]	RW	0 0000	DBA[4: 0]: DMA 基地址 这些位定义了 DMA 在连续模式下的基地址 (当对 TIMx_DMAR 寄存器的地址进行读或写时), DBA 定义为从 TIMx_CR1寄存器所在地址开始的偏移量: 00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR,

				.....
--	--	--	--	-------

### 21.4.20. TIM1 连续模式的 DMA 地址 (TIM1\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	DMAB[15: 0]	RW	0	<p>DMA 连续传送寄存器</p> <p>对 TIMx_DMAR 寄存器的读或写会导致对以下地址的寄存器的存取操作:</p> <p>TIMx_CR1地址 + (DBA + DMA 指针) x4, 其中:</p> <p>“TIMx_CR1地址” 是控制寄存器1的地址;</p> <p>“DBA” 是 TIMx_DCR 寄存器中定义的基地址;</p> <p>“DMA 指针” 是由 DMA 自动控制的偏移量, 它取决于 TIMx_DCR 寄存器中定义的 DBL。</p>

## 22. 通用定时器 (TIM2/3)

### 22.1. TIM2/TIM3简介

TIM2 通用定时器是由可编程预分频器驱动的 32 位自动重载计数器构成。

TIM3 通用定时器是由可编程预分频器驱动的 16 位自动重载计数器构成。

适合多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微妙到几个毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。他们可以一起同步操作。

### 22.2. TIM2/3主要特性

通用 TIM2/3 定时器功能包括：

- 16 位 (TIM3) ,32 位 (TIM2) 向上、向下、向上向下自动重载计数器
- 16 位可编程预分频器，计数器时钟频率的分频系数为 1~65536 之间的任意数值
- 4 个独立通道
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿或中央对齐模式）
  - 单脉冲模式输出
- 可以使用外部信号同步控制定时器和内部相连的其他定时器的同步电路
- 如下事件发生时产生中断/DMA
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

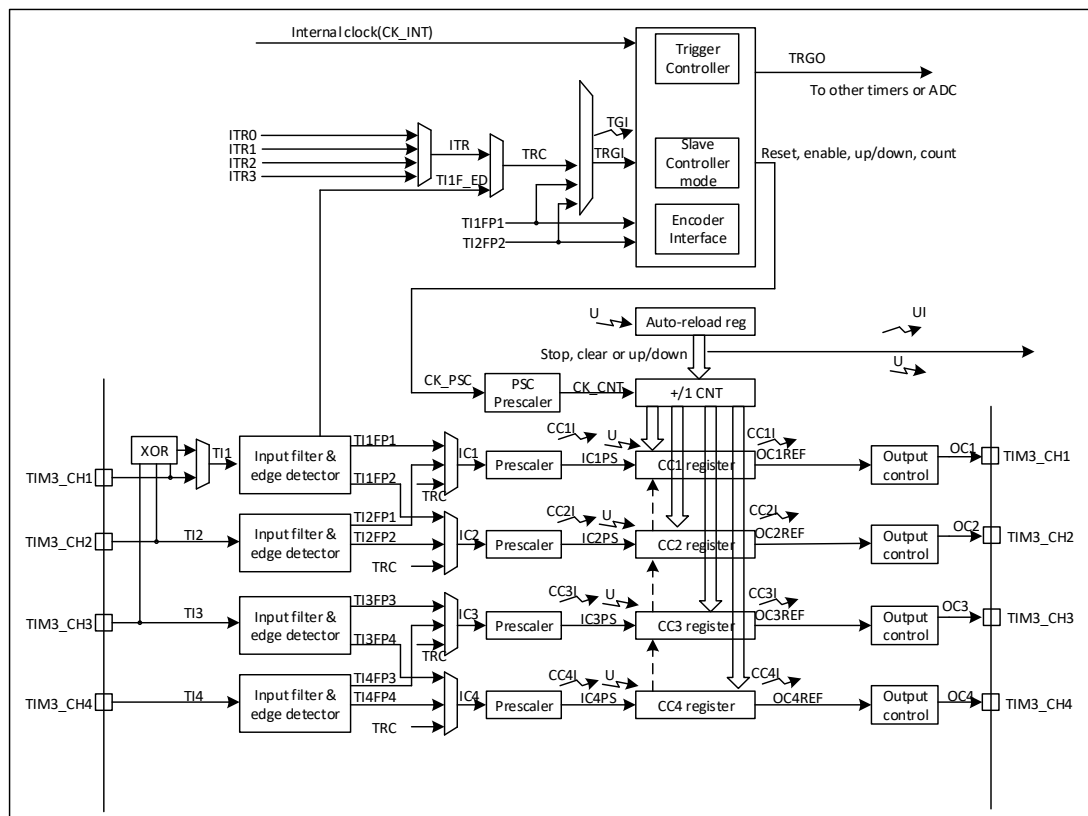


图 22-1 通用定时器架构图

## 22.3. TIM2/3功能描述

### 22.3.1. 时基单元

TIM2/3 定时器的主要部分是一个 16 位 (TIM3) 或者 32 位 (TIM2) 计数器和与其相关的自动重载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动重载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包括：

- 计数器寄存器 (TIMx\_CNT)
- 预分频寄存器 (TIMx\_PSC)
- 自动重载寄存器 (TIMx\_ARR)

自动重载寄存器是预先装载的，写或者读自动重载寄存器将访问预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动重载预装载使能位 (ARPE) 的设置，预装载寄存器的内容被一直或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢出 (向下计数器时的下溢条件) 并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIMx\_CR1 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

#### 预分频器描述

预分频器可以将计数器的时钟按 1 到 65536 之间的任意值分频。它是一个基于（在 TIMx\_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下次更新事件到来时被采用。

下图给出了在预分频器运行时，更改计数器参数的例子。

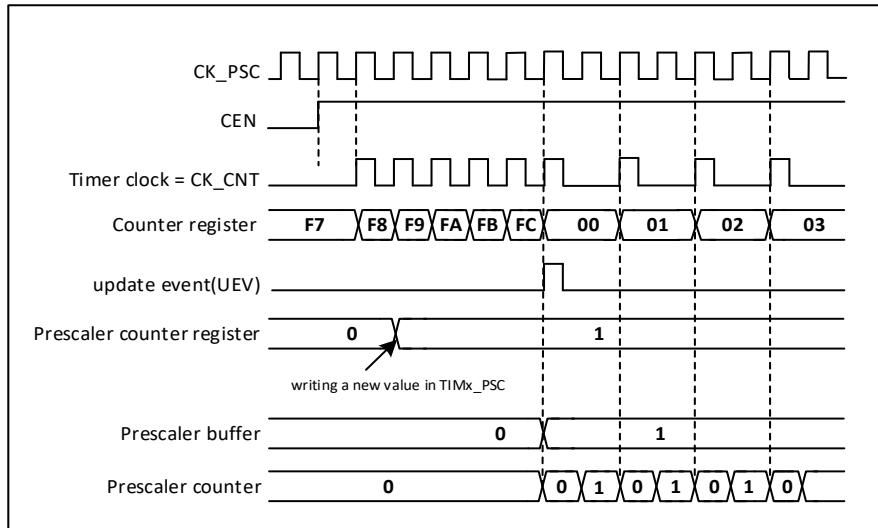


图 22-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

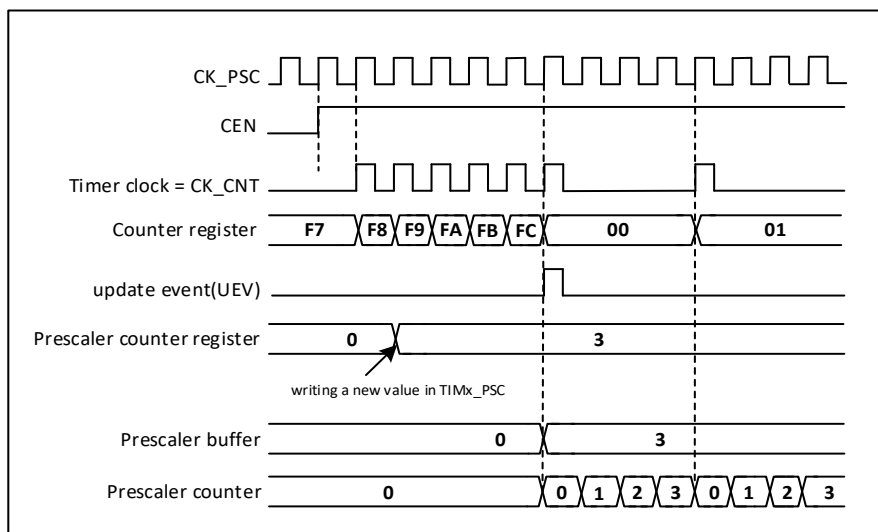


图 22-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

## 22.3.2. 计数器模式

### 22.3.2.1. 向上计数模式

向上计数模式，是从 0 计数到自动重载值的计数器，然后又从 0 重新开始计数，并产生一个计数的溢出事件。

每次计数上溢时可以产生更新事件，在 TIMx\_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。但是在应该产生更新事件

时，计数器仍会被清'0'，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求源），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时（依据 URS 位）设置更新标志位（TIMx\_SR 寄存器中的 UIF 位）。

- 自动重载影子寄存器被重新置入预装载寄存器的值（TIMx\_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIMx\_PSC 寄存器的内容）。

下图给出一些例子，当 TIMx\_ARR=0x36时计数器在不同时钟频率下的动作。

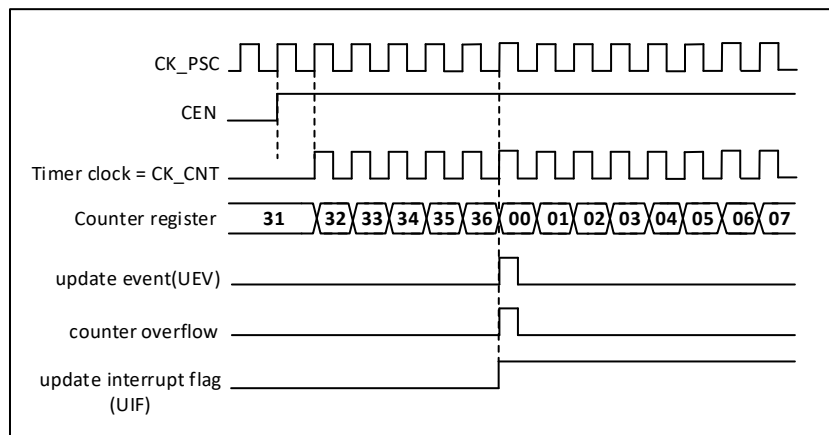


图 22-4 计数器时序图，内部时钟分频因子为 1

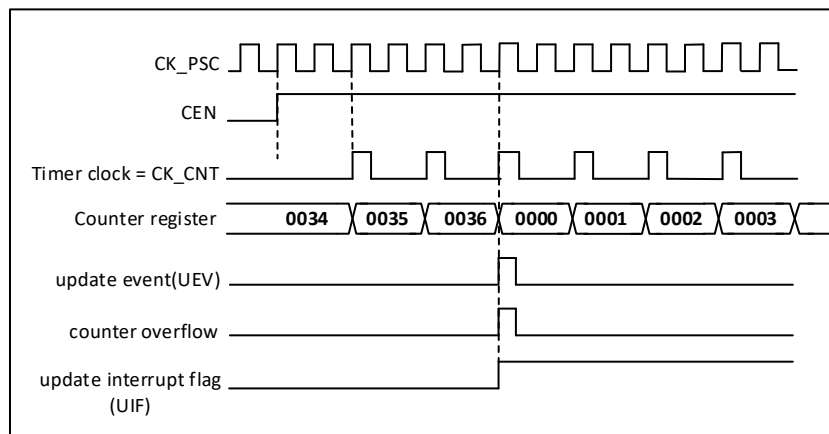


图 22-5 计数器时序图，内部时钟分频因子为2

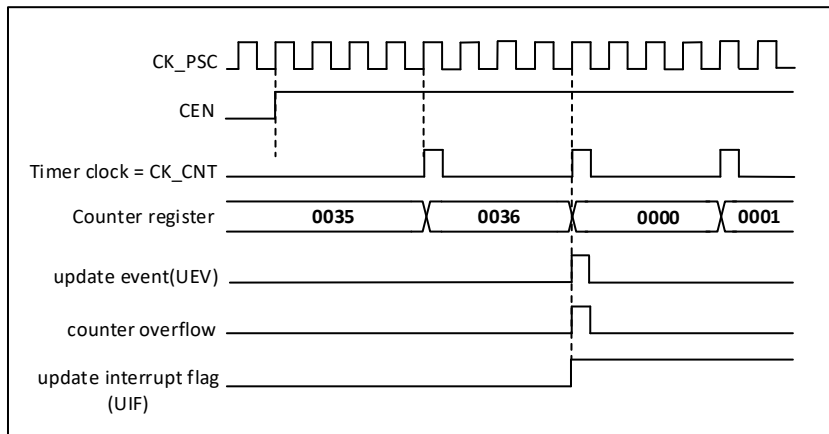


图 22-6 计数器时序图，内部时钟分频因子为 4

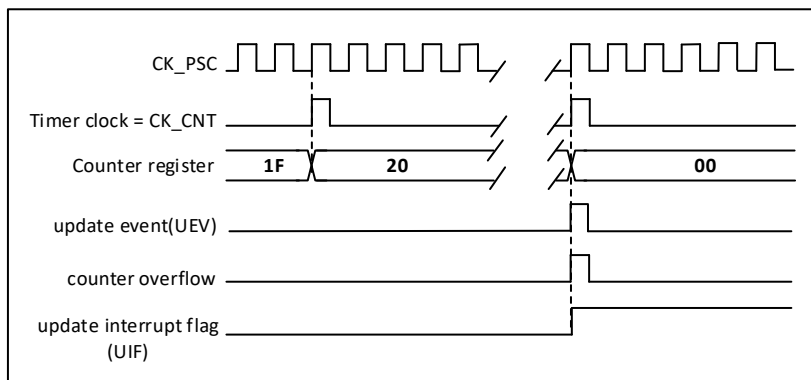


图 22-7 计数器时序图，内部时钟分频因子为 N

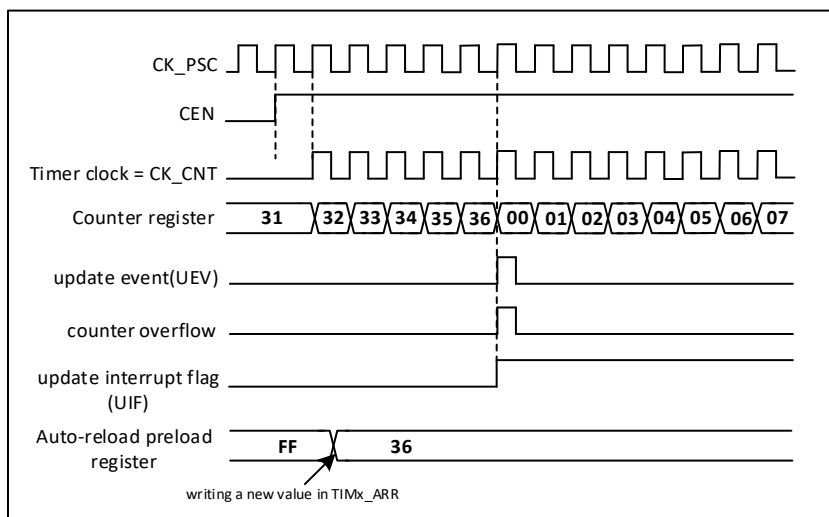


图 22-8 计数器时序图，当 ARPE=0 时的更新事件 (TIMx\_ARR 没有预装入)

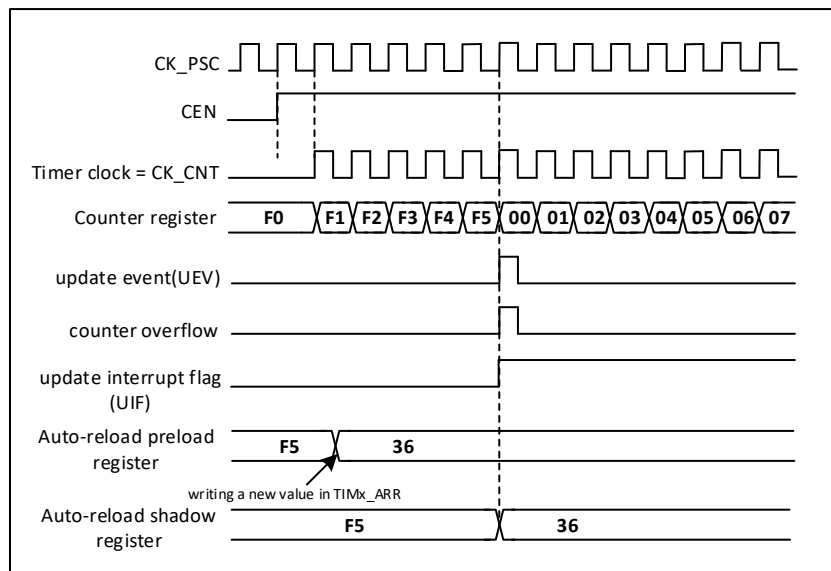


图 22-9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx\_ARR）

### 22.3.2.2. 向下计数模式

向下计数模式，从自动重载的值开始向下计数到 0，然后重新开始从自动重载的值向下计数，并产生一个向下溢出事件。

每次计数下溢时可以产生更新事件，在 TIMx\_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器的 UDIS 位可以禁止更新事件。这样可以避免向预装载寄存器中写入新值时更新寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器被清零（但预分频系数不变）。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求源），设置 UG 位将产生一个更新事件 UEV，但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有以下的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx\_SR 寄存器中的 UIF 位）也被设置。

- 预分频器的缓存器被加载为预装载的值（TIMx\_PSC 寄存器的值）。
- 当前的自动重载寄存器被更新为预装载值（TIMx\_ARR 寄存器中的内容）。

注：自动重载寄存器在计数器重载入之前被更新，因此下一个周期将是预期的值。



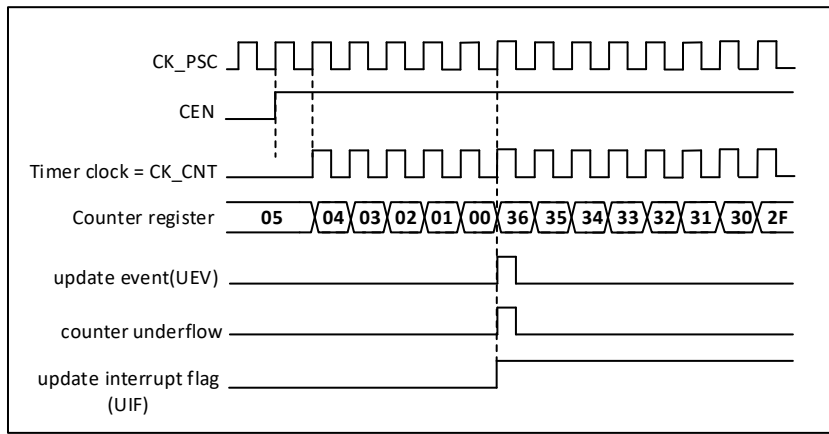


图 22-10 计数器时序图, 内部时钟分频因子为 1

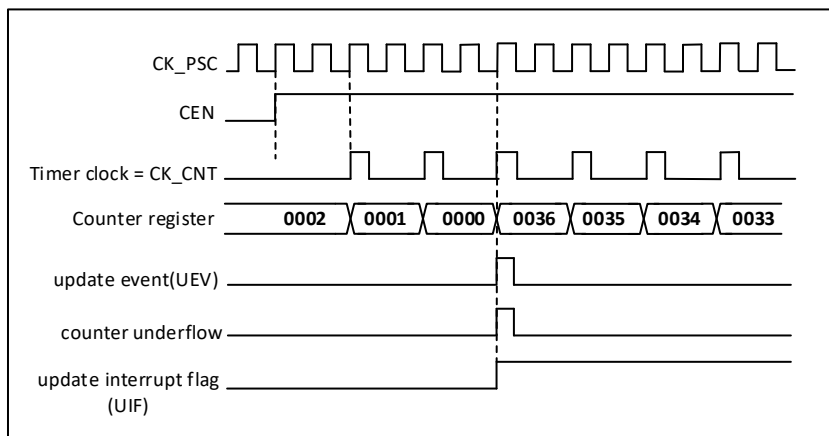


图 22-11 计数器时序图, 内部时钟分频因子为 2

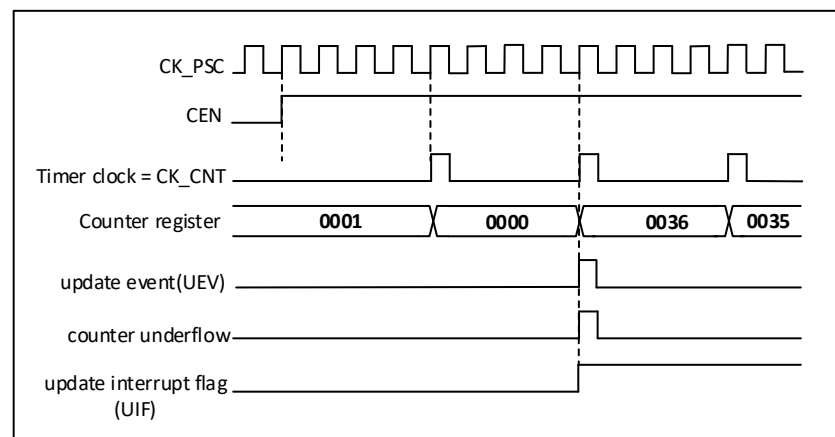


图 22-12 计数器时序图, 内部时钟分频因子为 4

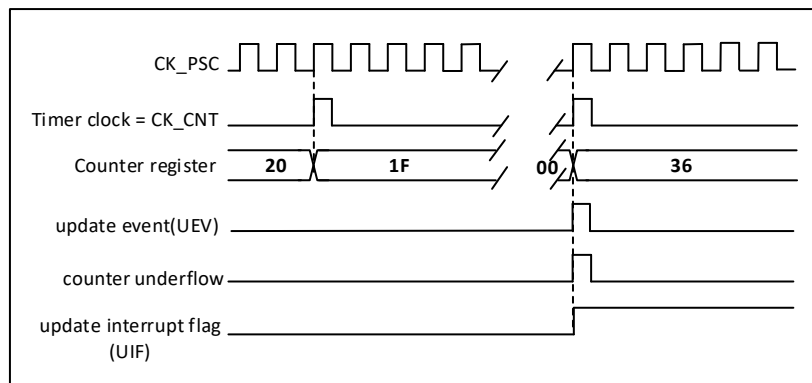


图 22-13 计数器时序图，内部时钟分频因子为 N

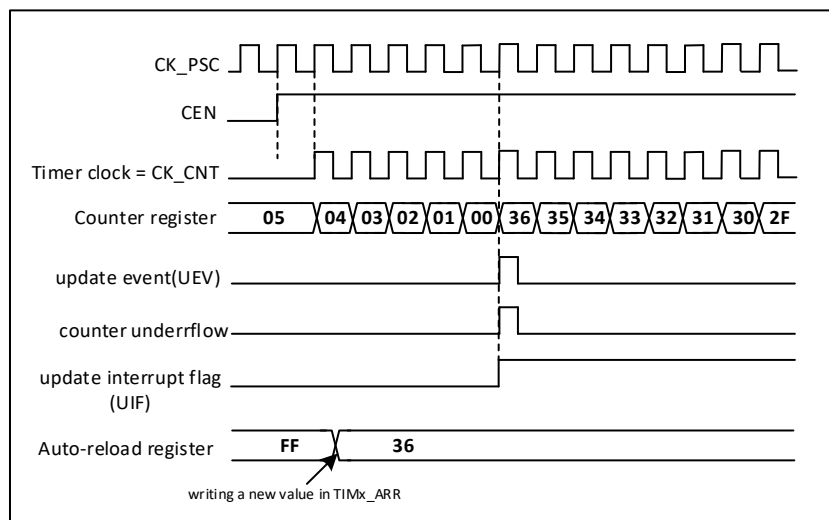


图 22-14 计数器时序图，当没有使用周期计数器时的更新事件

### 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值（TIMx\_ARR 寄存器）-1，产生一个计数器溢出事件，然后向下计数到 1，并产生一个计数器下溢事件，然后再从 0 开始重新计数。

中央对齐模式在 TIMx\_CR1 寄存器中的 CMS 不等于 0 时有效。通道在配置成输出模式时，输出比较中断标志将被置位，当：向下计数时（中央对齐模式 1，CMS="01"），向上计数时（中央对齐模式 2，CMS="10"）向上向下计数（中央对齐模式 3，CMS="11"）。

在此模式下，不能写入 TIMx\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）设置 TIMx\_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx\_SR 寄存器中的 UIF 位）也被设置。

- 预分频器的缓存器被加载为预装载（TIMx\_PSC 寄存器）的值。
- 当前的自动重载寄存器被更新为预装载值（TIMx\_ARR）

注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。

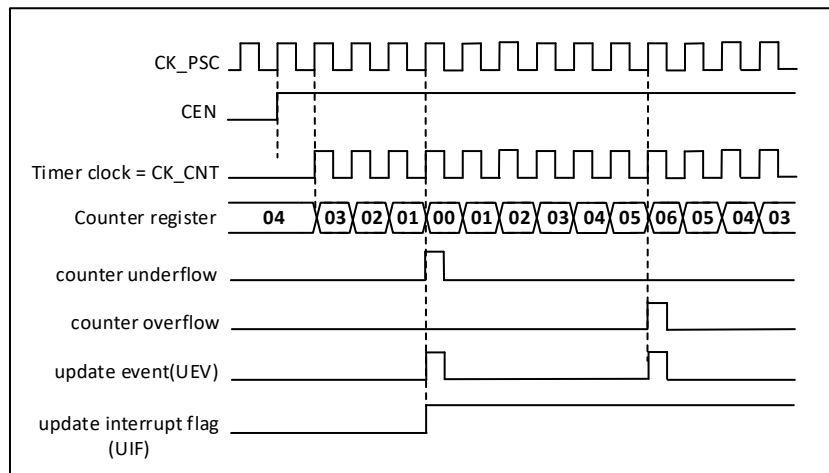


图 22-15 计数器时序图，内部时钟分频因子为 1，TIMx\_ARR = 0x6

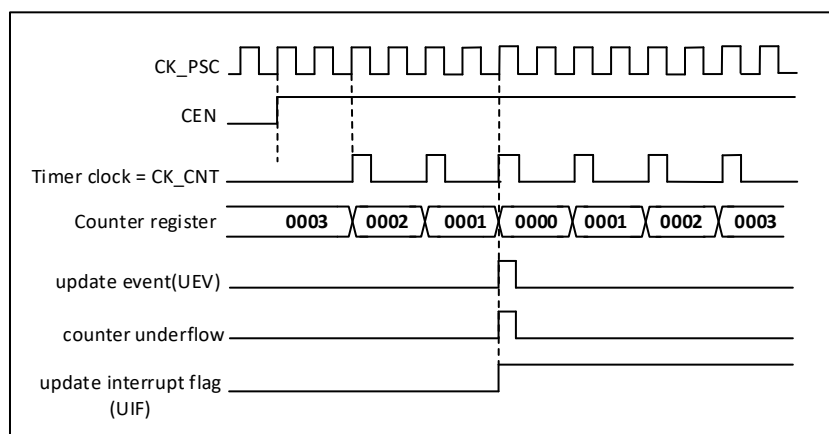


图 22-16 计数器时序图，内部时钟分频因子为 2

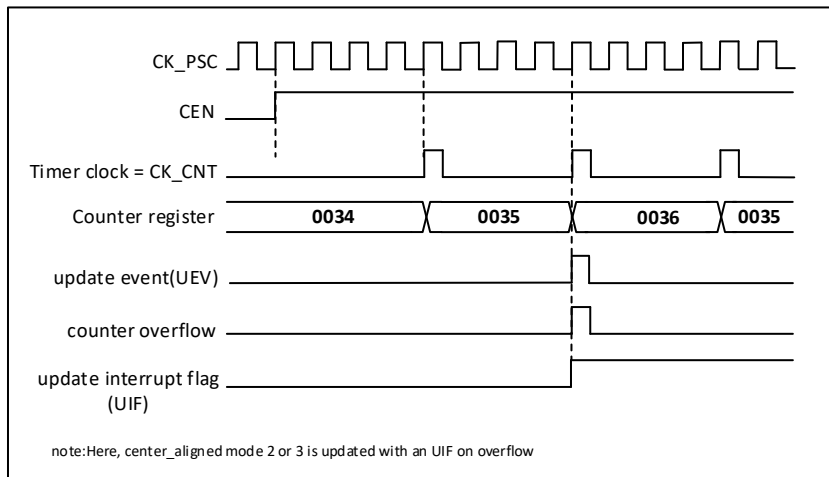


图 22-17 计数器时序图，内部时钟分频因子为4，TIMx\_ARR=0x36

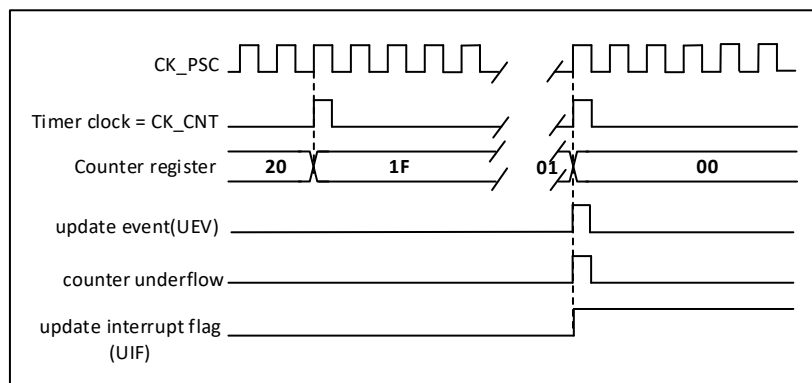


图 22-18 计数器时序图，内部时钟分频因子为 N

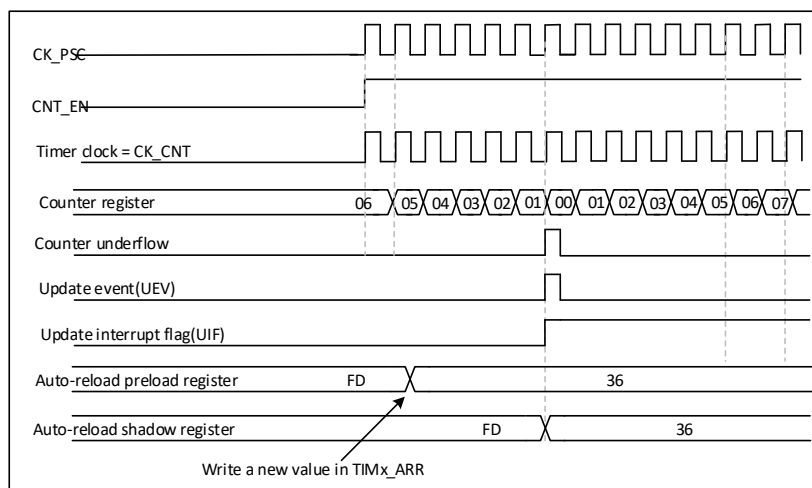


图 22-19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

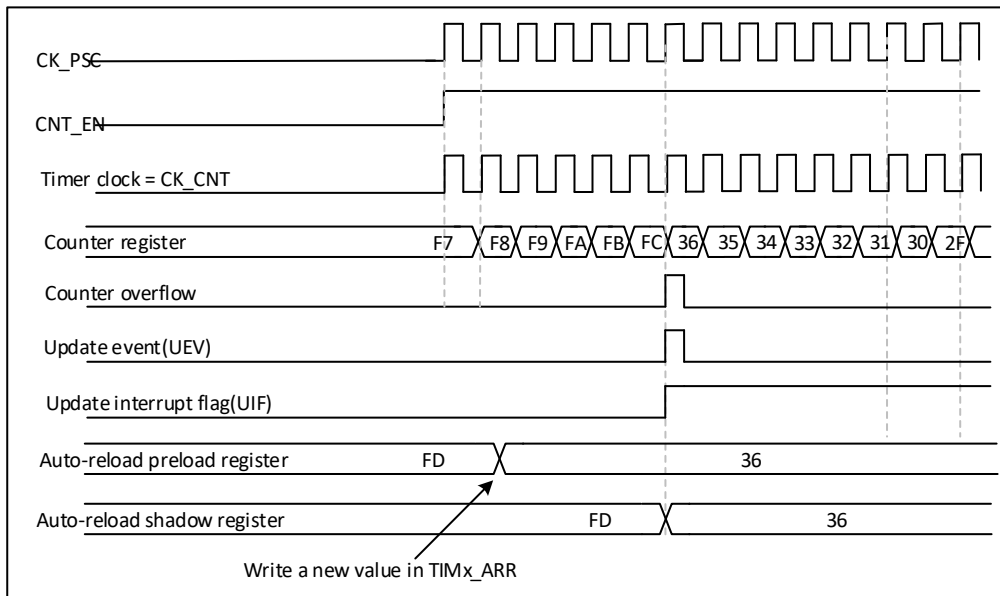


图 22-20 计数器时序图, ARPE=1 时的更新事件 (计数器上溢)

### 22.3.3. 时钟源

计数器的时钟可以由以下时钟源提供:

- 内部时钟 (CK\_INT)
- 外部时钟模式 1: 外部输入引脚
- 内部触发输入 (ITRx): 使用一个定时器作为另一个定时器的预分频器。例如, 可以配置一个定时器 Timer1 作为另一个定时器 Timer3 的预分频器。

#### 内部时钟源 (CK\_INT)

如果从模式控制器被禁止, 则 CEN、DIR (TIMx\_CR1 寄存器) 和 UG 位 (TIMx\_EGR 寄存器) 是事实上的控制位, 并且只能被软件修改。只要 CEN 位被写成 1, 预分频器的时钟就由内部时钟 CK\_INT 提供。

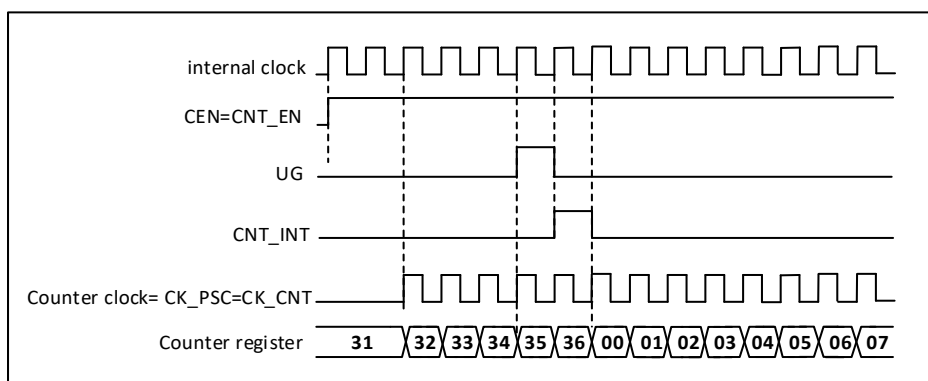


图 22-21 一般模式下的控制电路, 内部时钟分频因子为 1

#### 外部时钟源模式 1

当 TIMx\_SMCR 寄存器的 SMS=111 时, 此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

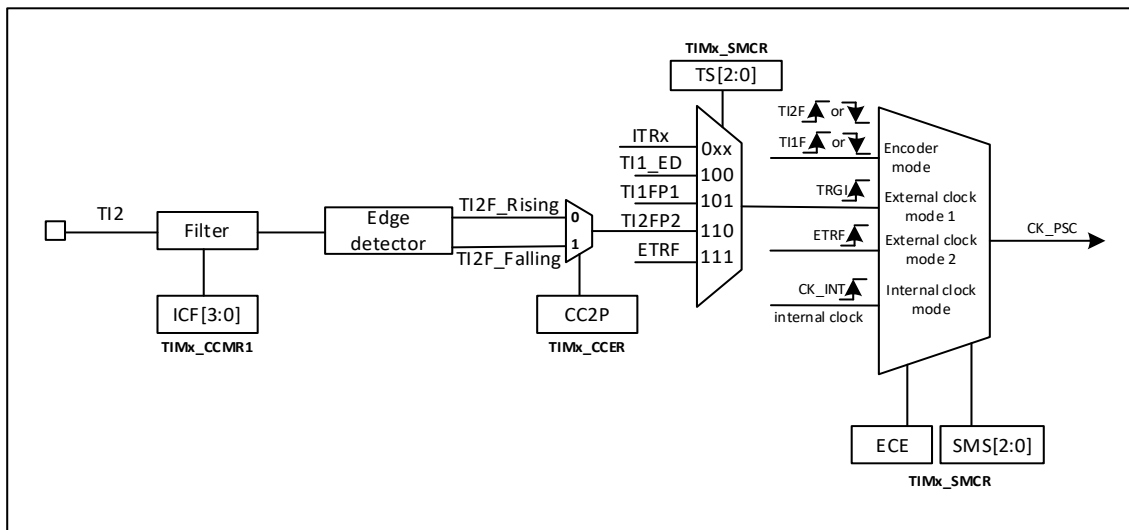


图 22-22 T12 外部时钟连接示例

例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置 TIMx\_CCMR1 寄存器 CC2S=01，配置通道2检测 T12 输入的上升沿
2. 配置 TIMx\_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2F=0000）
3. 配置 TIMx\_CCER 寄存器的 CC2P=0，选定上升沿极性
4. 配置 TIMx\_SMCR 寄存器的 SMS=111，选择定时器外部时钟模式1
5. 配置 TIMx\_SMCR 寄存器中的 TS=110，选定 T12 作为触发输入源
6. 设置 TIMx\_CR1 寄存器的 CEN=1，启动计数器

注：捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 T12，计数器计数一次，且 TIF 标志被设置。

在 T12 的上升沿和计数器实际时钟之间的延时取决于在 T12 输入端的重新同步电路。

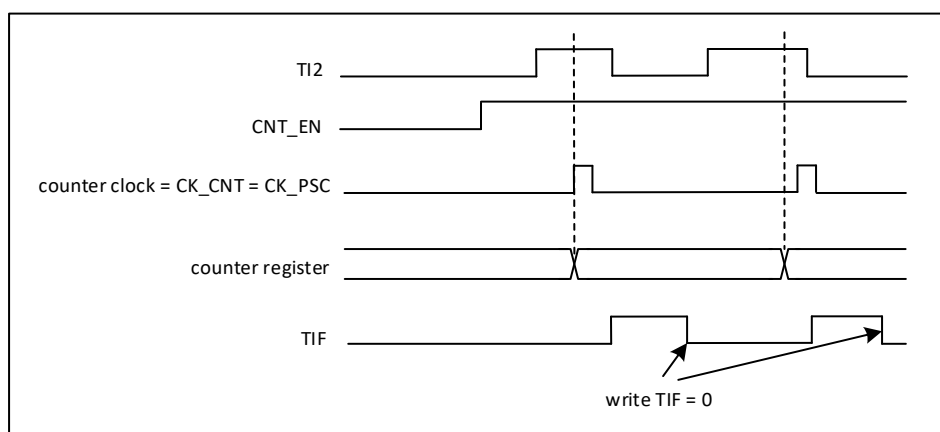


图 22-23 Control circuit in external clock mode 1

#### 22.3.4. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（输入滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

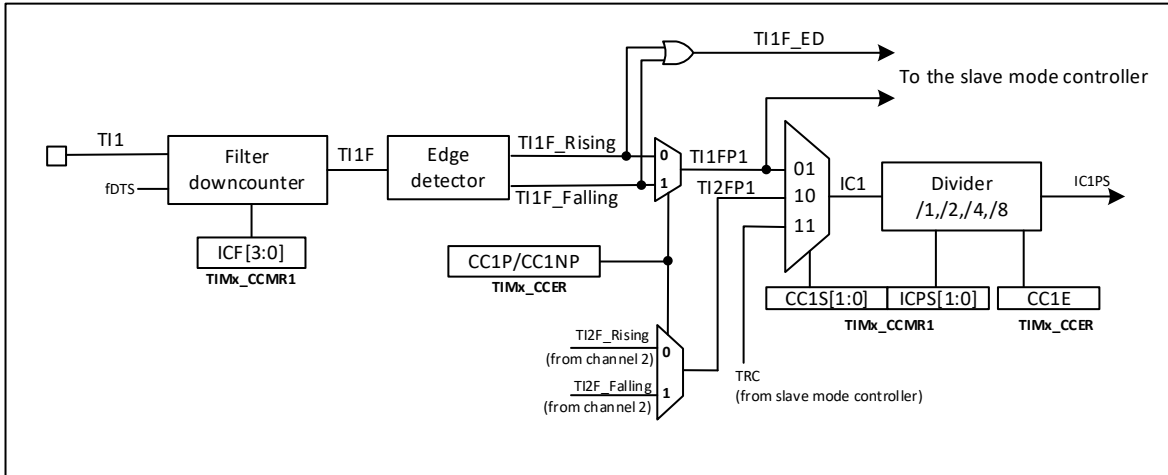


图 22-24 捕获/比较通道 (如: 通道 1 输入部分)

输出部分产生一个中间波形 OCxREF (高有效) 作为基准，链的末端决定最终输出信号的极性。

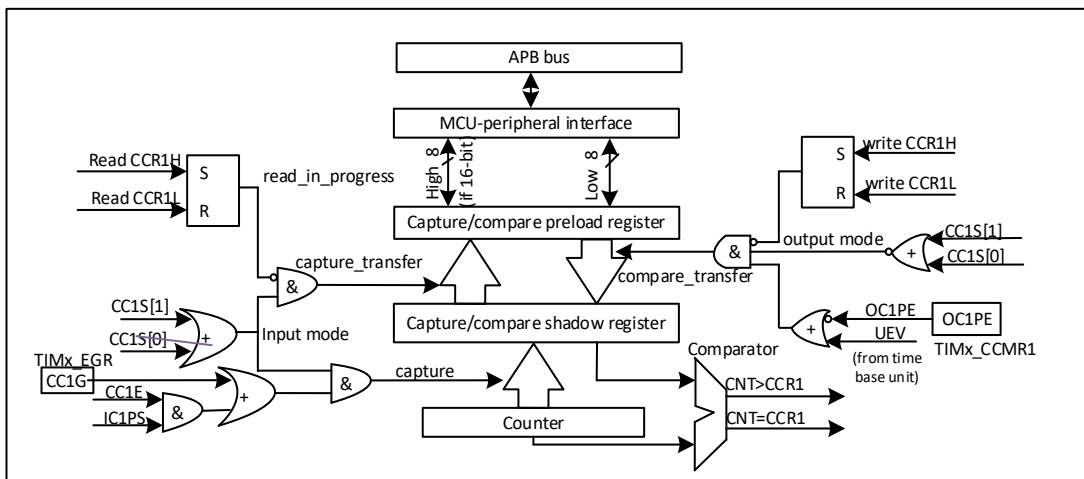


图 22-25 捕获/比较通道 1 的主电路

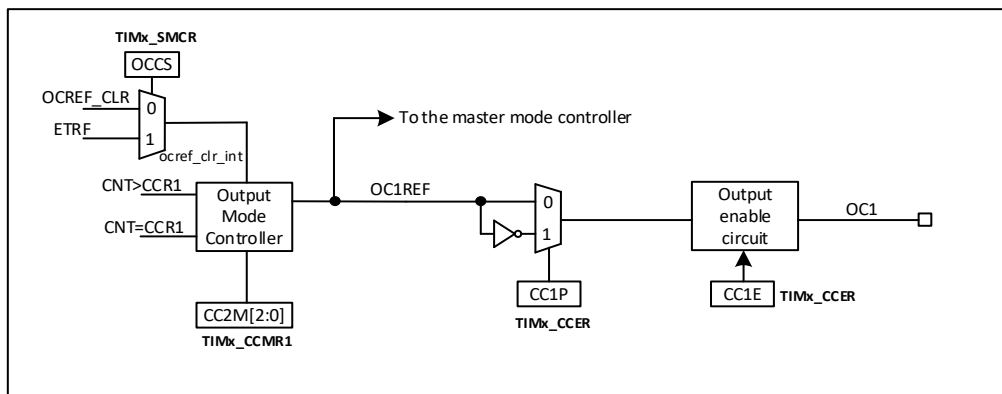


图 22-26 捕获/比较通道的输出部分 (通道1)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

### 22.3.5. 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器中。当发生捕获事件时，相应的 CCxIF 标志 (TIMx\_SR 寄存器) 被置1，如果中断和 DMA 操作被打开，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，则重复捕获标志 CCxOF (TIMx\_SR 寄存器) 被置1。写 CCxIF=0可清除 CCxIF，或读取存储在 TIMx\_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0可清除 CCxOF。

以下例子说明如何在 TI1输入的上升沿时捕获计数器的值到 TIMx\_CCR1寄存器中，步骤如下：

- 选择有效输入端：TIMx\_CCR1必须连接到 TI1输入，所以写入 TIMx\_CCR1寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx\_CCR1寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 Tix 时，输入滤波器控制位是 TIMx\_CCMRx 寄存器中的 ICxF 位）。假设输入信号在最多5个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于5个时钟周期；因此我们可以（以 fDTS 频率）连续采样8次，以确认在 TI1上一次真实的边沿变换，即在 TIMx\_CCMR1寄存器中写入 IC1F=0011。
- 选择 TI1通道的有效转换边沿，在 TIMx\_CCER 寄存器中写入 CC1P=0（上升沿）（和 CC1NP=0）
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx\_CCMR1寄存器的 IC1PS=00）。
- 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx\_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx\_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少2个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

### 22.3.6. PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射到同一个 Tix 输入。
- 这2个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TixFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，当需要测量输入到 TI1上的 PWM 信号的长度 (TIMx\_CCR1寄存器) 和占空比 (TIMx\_CCR2寄存器) 时，具体步骤如下（取决于 CK\_INT 的频率和预分频器的值）



- 选择 TIMx\_CCR1 的有效输入：置 TIMx\_CCMR1 寄存器的 CC1S=01（选中 TI1）。
- 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx\_CCR1 中和清除计数器）：置 CC1P=0（上升沿有效）。
- 选择 TIMx\_CCR2 的有效输入：置 TIMx\_CCMR1 寄存器的 CC2S=10（选中 TI1）。
- 选择 TI1FP2 的有效极性（捕获数据到 TIMx\_CCR2）：置 CC2P=1（下降沿有效）。
- 选择有效的触发输入信号：置 TIMx\_SMCR 寄存器中的 TS=101（选择 TI1FP1）。
- 配置从模式控制器为复位模式：置 TIMx\_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx\_CCER 寄存器中 CC1E=1 且 CC2E=1。

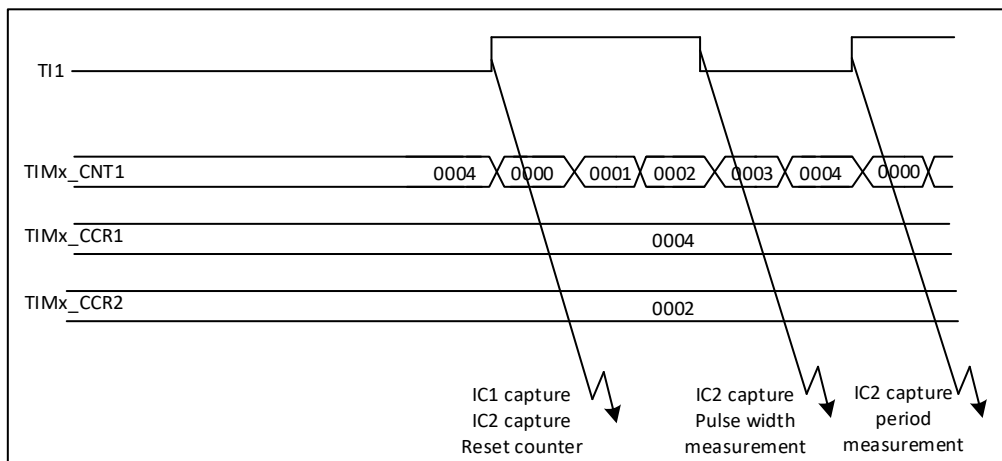


图 22-27 PWM 输入模式时序

### 22.3.7. 强置输出模式

在输出模式（TIMx\_CCMRx 寄存器中 CCxS=00）下，输出比较信号（OCxREF 和相应的 OCx）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。置 TIMx\_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0（OCx 高电平有效），则 OCx 被强置为高电平。置 TIMx\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下文的输出比较模式一节中介绍。

### 22.3.8. 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TIMx\_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx\_CCER 寄存器中的 CCxP 位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxM=000）、被设置成有效电平（OCxM=001）、被设置成无效电平（OCxM=010）或进行翻转（OCxM=011）。
- 设置中断状态寄存器中的标志位（TIMx\_SR 寄存器中的 CcxIF 位）。
- 若设置了相应的中断屏蔽（TIMx\_DIER 寄存器中的 CCxIE 位），则产生一个中断。
- 若设置了相应的使能位（TIMx\_DIER 寄存器中的 CCxDE 位，TIMx\_CR2 寄存器中的 CCDS 位选择 DMA 请求功能），则产生一个 DMA 请求。

TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟（内部，外部，预分频器）。
2. 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
  - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
  - 置 OCxPE = 0禁用预装载寄存器
  - 置 CCxP = 0选择极性为高电平有效
  - 置 CCxE = 1使能输出
5. 设置 TIMx\_CR1寄存器的 CEN 位启动计数器

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPE='0'，否则 TIMx\_CCRx 的影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

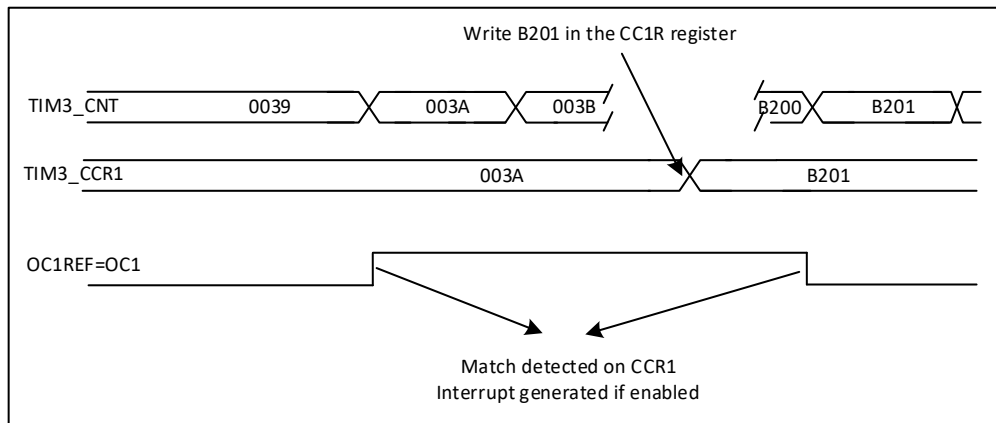


图 22-28 输出比较模式，翻转 OC1

### 22.3.9. PWM 模式

脉宽调制模式可以允许产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入“110”（PWM 模式1）或“111”（PWM 模式2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx\_CR1寄存器的 ARPE 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过（TIMx\_CCER 和 TIMx\_BDTR 寄存器中）CcxE、CcxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx\_CCER 寄存器的描述。

在 PWM 模式（模式1或模式2）下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，（依据计数器的计数方向）以确定是否符合  $TIMx\_CCRx \leq TIMx\_CNT$  或者  $TIMx\_CNT \leq TIMx\_CCRx$ 。

根据 TIMx\_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

## PWM 边沿对齐模式

### ■ 向上计数配置

当 TIMx\_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看下面是一个 PWM 模式1的例子。当  $TIMx\_CNT < TIMx\_CCRx$  时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自动重装载值 (TIMx\_ARR)，则 OCxREF 保持为'1'。如果比较值为0，则 OCxREF 保持为'0'。下图为 TIMx\_ARR=8时边沿对齐的 PWM 波形实例。

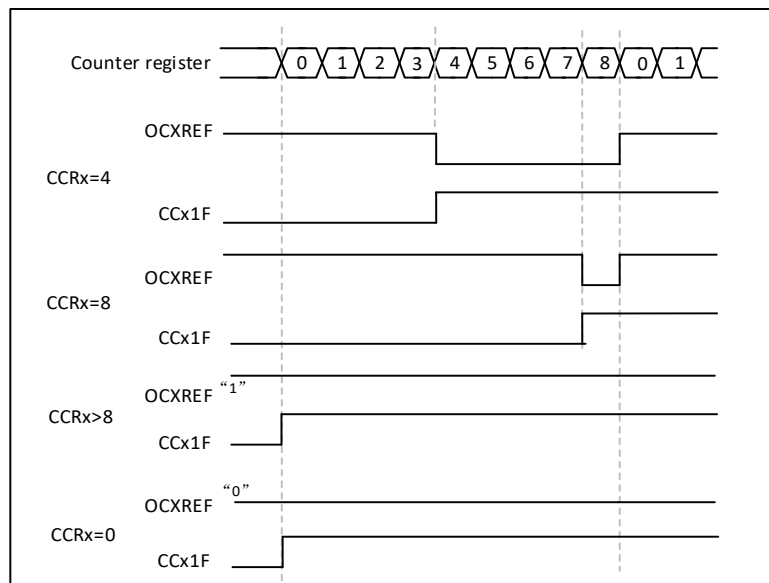


图 22-29 Edge-aligned PWM waveforms (ARR=8)

### 向下计数配置

当 TIMx\_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式1，当  $TIMx\_CNT > TIMx\_CCRx$  时参考信号 OCxREF 为低，否则为高。如果 TIMx\_CCRx 中的比较值大于 TIMx\_ARR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生0%的 PWM 波形。

### PWM 中央对齐模式

当 TIMx\_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式（所有其他的配置对 OCxREF/OCx 信号都有相同的作用）。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置1、在计数器向下计数时被置1、或在计数器向上和向下计数时被置1。TIMx\_CR1 寄存器中的计数方向位 (DIR) 由硬件更新，不要用软件修改它。

下图给出一些中央对齐的 PWM 波形的例子

- TIMx\_ARR = 8
- PWM 模式1
- TIMx\_CR1 寄存器的 CMS=01，在中央对齐模式下，当计数器向下计数时设置比较标志

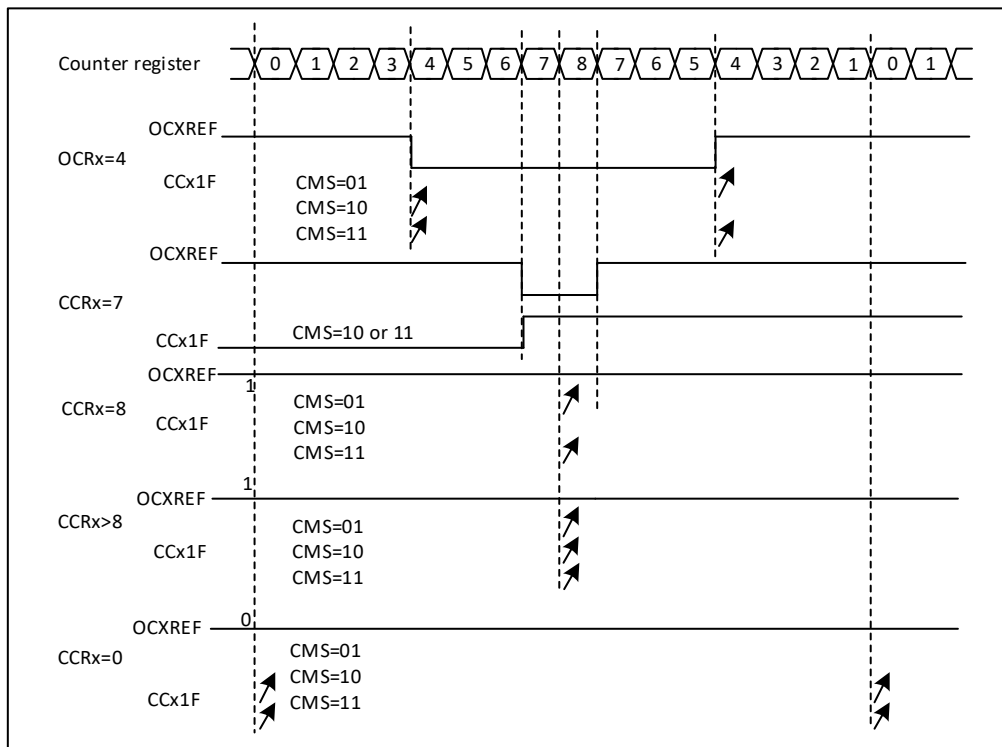


图 22-30 中央对齐的 PWM 波形 (APR=8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx\_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值 ( $TIMx\_CNT > TIMx\_ARR$ )，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx\_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置 TIMx\_EGR 位中的 UG 位），并且不要在计数进行过程中修改计数器的值。

### 22.3.10. 单脉冲模式

单脉冲模式 (OPM) 是之前所述众多模式中的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可被程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器的 OPM 位将选择单脉冲模式，这样可以使计数器自动的在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$ （特别地， $0 < CCRx$ ）
- 向下计数方式：计数器  $CNT > CCRx$

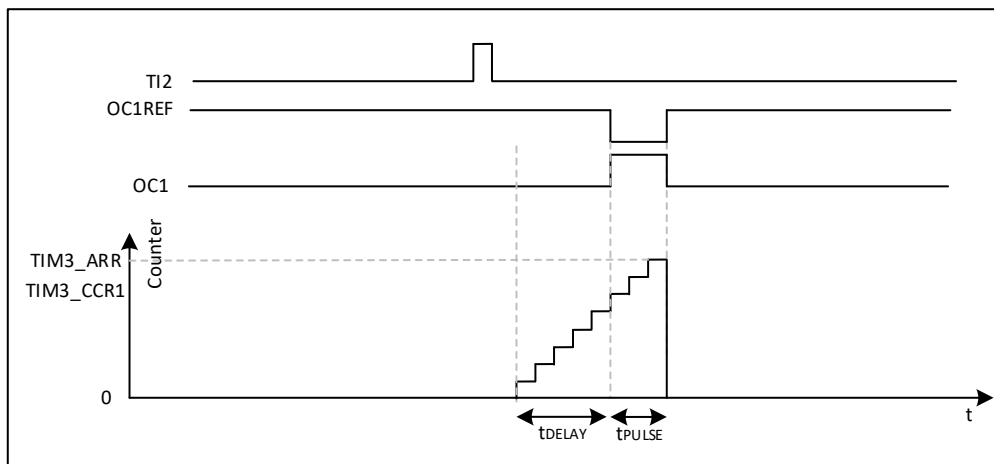


图 22-31 单脉冲模式的例子

例如，当需要在从 TI2 输入脚上检测到一个上升沿开始，延迟  $t_{\text{DELAY}}$  之后，在 OC1 上产生一个长度为  $t_{\text{PULSE}}$  的正脉冲。

使用 TI2FP2 作为触发 1：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx\_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发 (TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110 (触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)

- $t_{\text{DELAY}}$  由 TIMx\_CCR1 寄存器中的值定义。
- $t_{\text{PULSE}}$  由自动重载值和比较值之间的差值定义 (TIMx\_ARR - TIMx\_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE=1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动重载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM=1，在下一个更新事件 (当计数器从自动重载值翻转到 0) 时停止计数。

#### 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时  $t_{\text{DELAY}}$ 。

如果要以最小延时输出波形，可以设置 TIMx\_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF (和 OCx) 直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

### 22.3.11. 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx\_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx\_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1和 TI2极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1和 TI2被用来作为增量编码器的接口。参看表22-1，假定计数器已经启动（TIMx\_CR1寄存器中的 CEN=1），则计数器由每次在 TI1FP1或 TI2FP2上的有效跳变驱动。TI1FP1和 TI2FP2是 TI1和 TI2在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx\_CR1寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1计数、依靠 TI2计数或者同时依靠 TI1和 TI2计数，在任一输入端（TI1或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在0到 TIMx\_ARR寄存器的自动重载值之间连续计数（根据方向，或是0到 ARR 计数，或是 ARR 到0计数）。所以在开始计数之前必须配置 TIMx\_ARR；同样，捕获器、比较器、预分频器、触发输出特性等仍工作如常。编码器模式和外部时钟模式2不兼容，因此不能同时操作。在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1和 TI2不同时变换。

表 22-1 计数方向与编码器信号的关系

有效边沿	相对信号的电平（TI1FP1的相对信号为 TI2，TI2FP2的相对信号为 TI1）	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在 TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 TI1和 TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差分输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S='01'（TIMx\_CCMR1寄存器，TI1FP1映射到 IC1）
- CC2S='01'（TIMx\_CCMR2寄存器，TI2FP2映射到 IC2）
- CC1P='0'（TIMx\_CCER 寄存器，TI1FP1不反相，TI1FP1=TI1）
- CC2P='0'（TIMx\_CCER 寄存器，TI2FP2不反相，TI2FP2=TI2）
- SMS='011'（TIMx\_SMCR 寄存器，所有的输入均在上升沿和下降沿有效）
- CEN='1'（TIMx\_CR1寄存器，计数器使能）

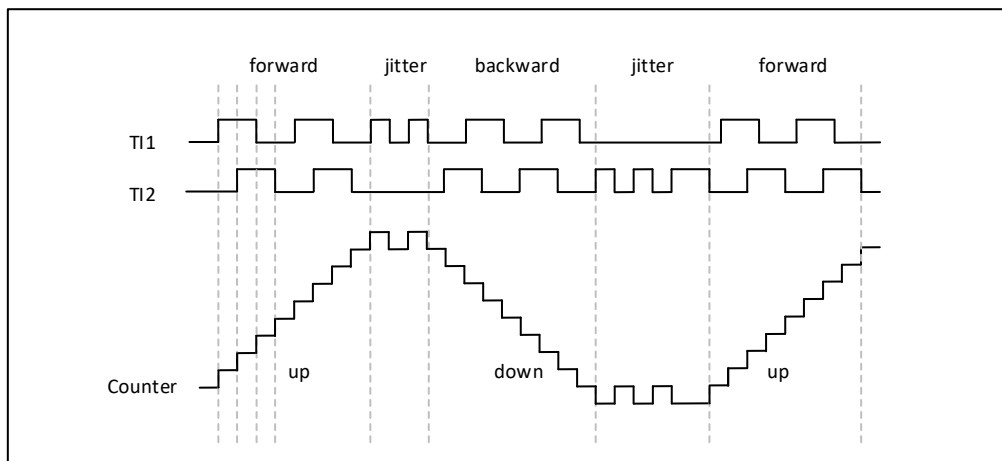


图 22-32 编码器模式下的计数器操作实例

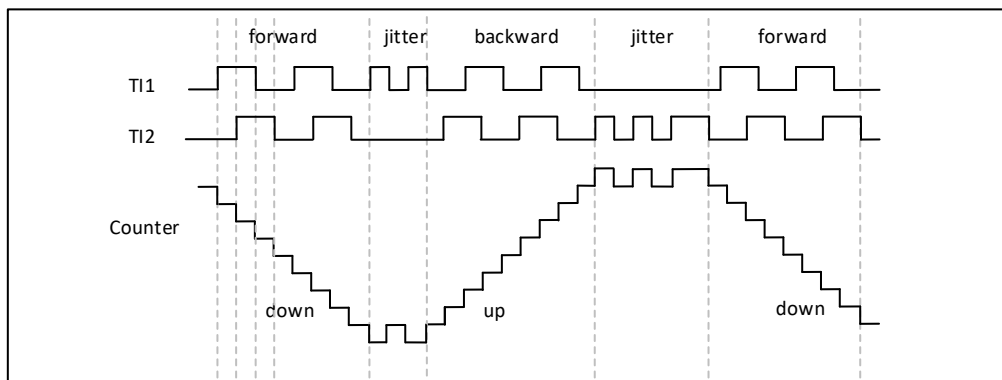


图 22-33 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前的位置信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度、加速度、减速度）。指示机械零点的编码器输出可被用作此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的，并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 22.3.12. 定时器输入异或功能

TIMx\_CR2寄存器的 TI1S 位，允许通道1的输入滤波器连接到一个异或门的输出端，异或门的3个输入端为 TIMx\_CH1、TIMx\_CH2和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。

一个应用例子是霍尔传感器接口。

### 22.3.13. 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

在以下的例子中，TI1输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测 TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0（和 CC1NP=0）以确定极性（只检测上升沿）。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1出现一个上升沿；此时，计数器被清零然后从0重新开始计数。同时，触发标志（TIMx\_SR 寄存器中的 TIF 位）被设置，根据 TIMx\_DIER 寄存器中 TIE（中断使能）位和 TDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重载寄存器 TIMx\_ARR=0x36时的动作。在 TI1上升沿和计数器的实际复位之间的延时取决于 TI1输入端的重同步电路。

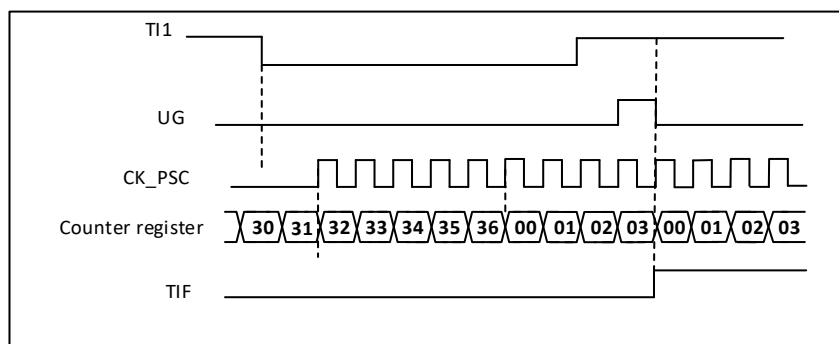


图 22-34 复位模式下的控制电路

#### Slave mode: Gated mode

按照选中的输入端的电平使能计数器。

在如下的例子中，计数器只在 TI1为低时向上计数：

- 配置通道1以检测 TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1（和 CC1NP=0）以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1为低，计数器开始依据内部时钟计数，一旦 TI1变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标志。

TI1上升沿和计数器实际停止之间的延时取决于 TI1输入端的重同步电路。



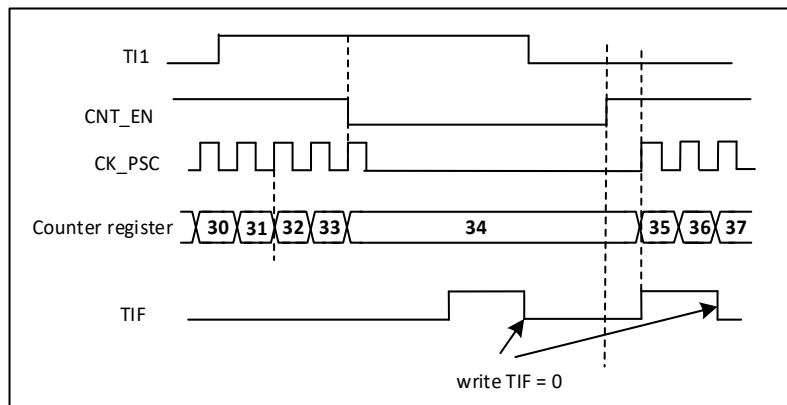


图 22-35 门控模式下的控制电路

### 从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2输入的上升沿开始向上计数：

- 配置通道2检测 TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC2S=01。置 TIMx\_CCER 寄存器中 CC2P=1（和 CC2NP=0）以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx\_SMCR 寄存器中 TS=110，选择 TI2作为输入源。

当 TI2出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。TI2上升沿和计数器启动计数之间的延时，取决于 TI2输入端的重同步电路。

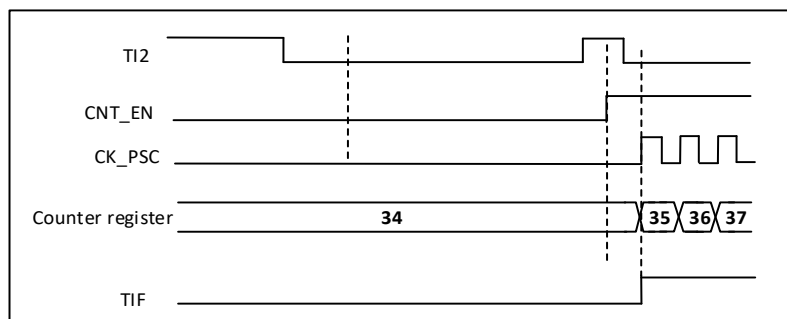


图 22-36 触发器模式下的控制电路

### 从模式：外部时钟模式 2 + 触发模式

外部时钟模式2可以与另一种从模式（外部时钟模式1和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx\_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

- 通过 TIMx\_SMCR 寄存器配置外部触发输入电路：
  - ETF=0000：没有滤波
  - ETPS=00：不用预分频器
  - ETP=0：检测 ETR 的上升沿，置 ECE=1使能外部时钟模式2。
- 按如下配置通道1，检测 TI1的上升沿：
  - IC1F=0000：没有滤波

- 触发操作中不使用捕获预分频器，不需要配置
- 置 TIMx\_CCMR1 寄存器中 CC1S=01，选择输入捕获源
- 置 TIMx\_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）
- 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

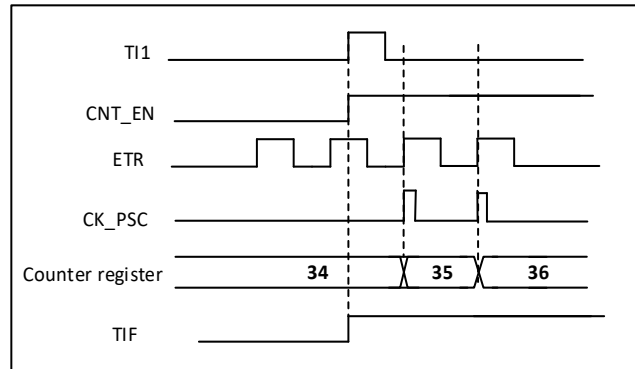


图 22-37 外部时钟模式 2 + 触发模式下的控制电路

#### 22.3.14. 定时器同步

所有定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

##### 使用一个定时器作为另一个的预分频器

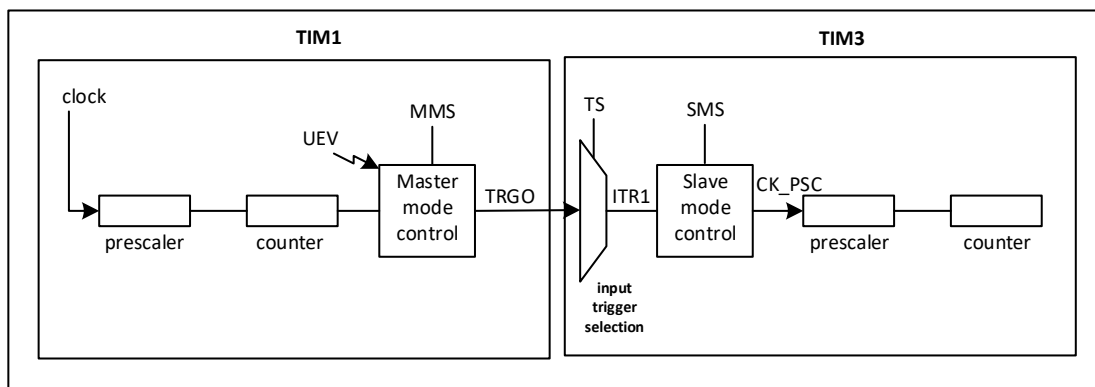


图 22-38 主/从定时器的例子

如：可以配置定时器1作为定时器3的预分频器。进行下述操作：

- 配置定时器1为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1\_CR2 寄存器的 MMS='010' 时，每当产生一个更新事件时在 TRGO 上输出一个上升沿信号。
- 连接定时器1的 TRGO 输出至定时器3，设置 TIM3\_SMCR 寄存器的 TS='000'，配置定时器3为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式1 (TIM3\_SMCR 寄存器的 SMS=111)；这样定时器3即可由定时器1周期性的上升沿（即定时器1的计数器溢出）信号驱动。

- 最后，必须设置相应（TIM3\_CR1寄存器）的CEN位分别启动两个定时器，确保先启动Timer3，后启动Timer1。

注：如果OCx已被选中为定时器1的触发输出（MMS=1xx），它的上升沿用于驱动定时器3的计数器。

### 使用一个定时器去使能另一个定时器

在这个例子中，定时器3的使能由定时器1的输出比较控制。参考上图的连接。只当定时器1的OC1REF为高时，定时器3才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对CK\_INT除以3（ $f_{CK\_CNT}=f_{CK\_INT}/3$ ）得到。

- 配置定时器1为主模式，送出它的输出比较参考信号（OC1REF）为触发输出（TIM1\_CR2寄存器的MMS=100）
- 配置定时器1的OC1REF波形（TIM1\_CCMR1寄存器）
- 配置定时器3从定时器1获得输入触发（TIM3\_SMCR寄存器的TS=000）
- 配置定时器3为门控模式（TIM3\_SMCR寄存器的SMS=101）
- 置TIM3\_CR1寄存器的CEN=1以使能定时器3
- 置TIM1\_CR1寄存器的CEN=1以启动定时器1

注：定时器3的时钟不与定时器1的时钟同步，这个模式只影响定时器3计数器的使能信号。

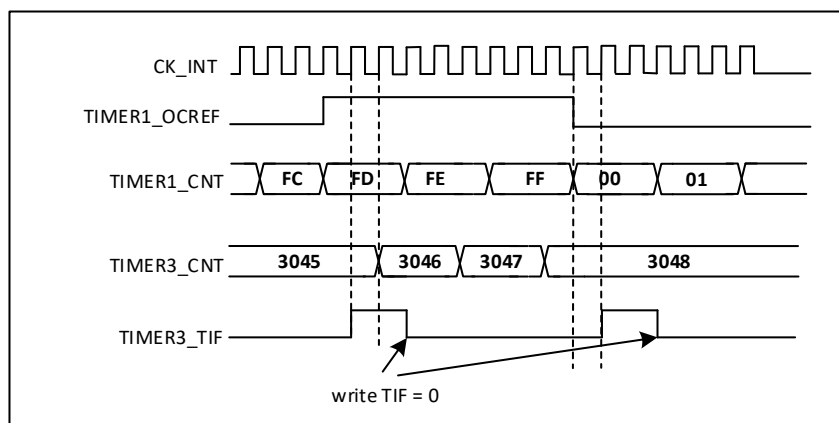


图 22-39 定时器 1 的 OC1REF 控制定时器 3

在上图的例子中，在定时器3启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器1之前复位2个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写TIMx\_EGR寄存器的UG位即可复位定时器。

在下一个例子中，需要同步定时器1和定时器2。定时器1是主模式并从0开始，定时器2是从模式并从0xE7开始；2个定时器的预分频器系数相同。写'0'到TIM1\_CR1的CEN位将禁止定时器1，定时器2随即停止。

- 配置定时器1为主模式，送出定时器使能信号（CNT\_EN）做为触发输出（TIM1\_CR2寄存器MMS=001的）。
- 配置定时器1的OC1REF波形（TIM1\_CCMR1寄存器）。
- 配置定时器2从定时器1获得输入触发（TIM2\_SMCR寄存器的TS=000）
- 配置定时器2为门控模式（TIM2\_SMCR寄存器的SMS=101）
- 置TIM1\_EGR寄存器的UG='1'，复位定时器1。
- 置TIM2\_EGR寄存器的UG='1'，复位定时器2。
- 写'0xE7'至定时器2的计数器（TIM2\_CNT），初始化它为0xE7。
- 置TIM2\_CR1寄存器的CEN='1'以使能定时器2。

- 置 TIM1\_CR1寄存器的 CEN='1'以启动定时器1。
- 置 TIM1\_CR1寄存器的 CEN='0'以停止定时器1。

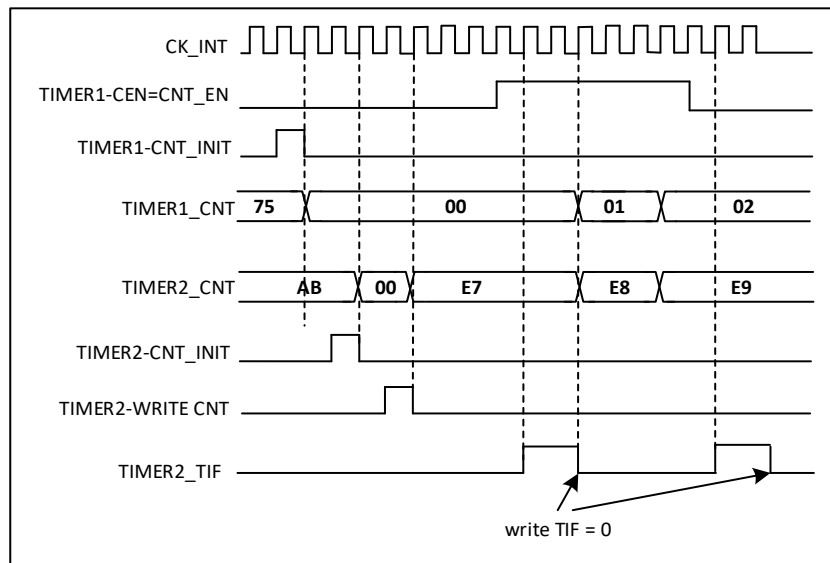


图 22-40 通过使能定时器 1 可以控制定时器 2

### 使用一个定时器去启动另一个定时器

在这个例子中，使用定时器1的更新事件使能定时器3。参考图22-38的连接。一旦定时器1产生更新事件，定时器3即从它当前的数值（可以是非0）按照分频的内部时钟开始计数。在收到触发信号时，定时器3的 CEN 位被自动地置'1'，同时计数器开始计数直到写'0'到 TIM3\_CR1寄存器的 CEN 位。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以3 ( $f_{CK\_CNT}=f_{CK\_INT}/3$ )。

- 配置定时器1为主模式，送出它的更新事件（UEV）做为触发输出（TIM1\_CR2寄存器的 MMS=010）
- 配置定时器1的周期（TIM1\_ARR 寄存器）
- 配置定时器3从定时器1获得输入触发（TIM3\_SMCR 寄存器的 TS=000）
- 配置定时器3为触发模式（TIM3\_SMCR 寄存器的 SMS=110）
- 置 TIM1\_CR1寄存器的 CEN=1以启动定时器1

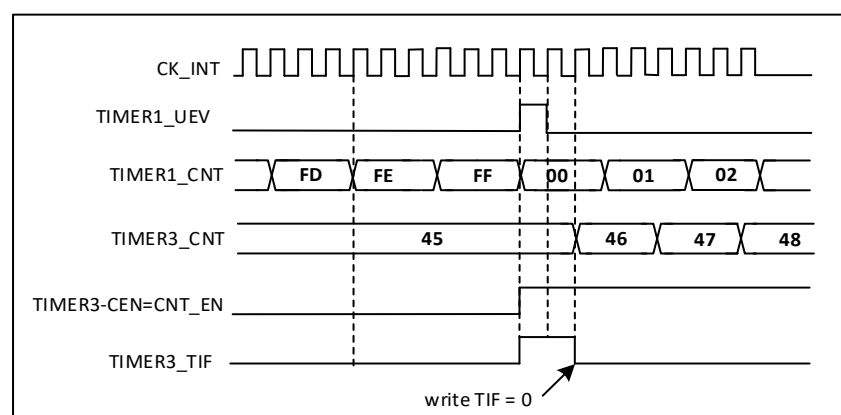


图 22-41 使用定时器 1 的更新触发定时器 3

在上一个例子中，可以在启动计数之前初始化两个计数器。下图显示在与上图相同配置情况下，使用触发模式而不是门控模式（TIM3\_SMCR 寄存器的 SMS=110）的动作。

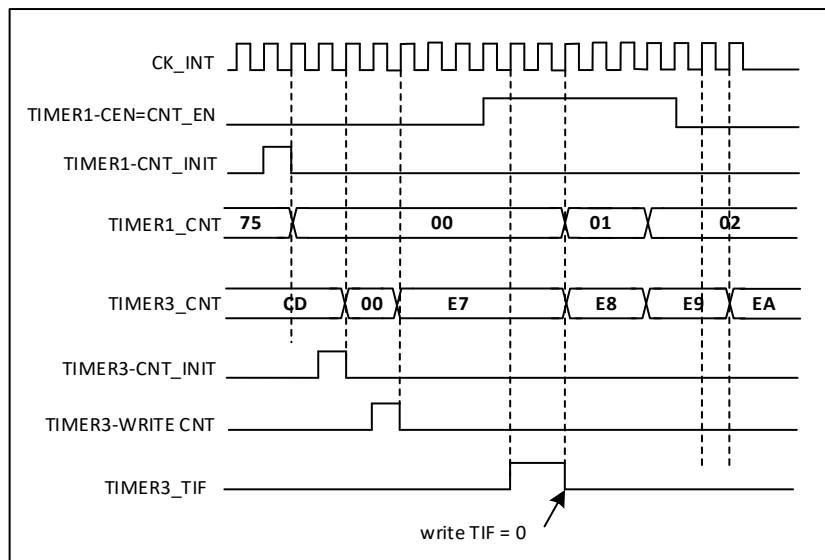


图 22-42 利用定时器 1 的使能触发定时器 3

### 使用一个外部触发同步地启动2个定时器

这个例子中当定时器1的TI1输入上升时使能定时器1，使能定时器1的同时使能定时器3，参见图22-38的连接方式。为保证计数器的对齐，定时器1必须配置为主/从模式（对应TI1为从，对应定时器3为主）：

- 配置定时器1为主模式，送出它的使能作为触发输出（TIM1\_CR2寄存器的MMS='001'）。
- 配置定时器1为从模式，从TI1获得输入触发（TIM1\_SMCR寄存器的TS='100'）。
- 配置定时器1为触发模式（TIM1\_SMCR寄存器的SMS='110'）。
- 配置定时器1为主/从模式，TIM1\_SMCR寄存器的MSM='1'。
- 配置定时器3从定时器1获得输入触发（TIM3\_SMCR寄存器的TS=000）
- 配置定时器3为触发模式（TIM3\_SMCR寄存器的SMS='110'）。

当定时器1的TI1上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个TIF标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化（设置相应的UG位），两个计数器都从0开始，但可以通过写入任意一个计数器寄存器（TIMx\_CNT）在定时器间插入一个偏移。下图中能看到主/从模式下在定时器1的CNT\_EN和CK\_PSC之间有个延迟。

### 22.3.15. 调试模式

当芯片进入调试模式时，根据DBG模块中DBG\_TIMx\_STOP的设置，TIMx计数器可以继续正常工作或者停止工作。

## 22.4. 寄存器描述

TIM2寄存器基地址：0x4000 0000

TIM3寄存器基地址：0x4000 0400

### 22.4.1. TIM2/3 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD[1: 0]		ARPE	CMS[1: 0]		DIR	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9: 8	CKD[1: 0]	RW	00	<p>时钟分频因子</p> <p>这2位定义在定时器时钟 (CK_INT) 频率, 死区时间和由死区发生器与数字滤波器 (ETR,Tix) 所用的采样时钟之间的分频比例</p> <p>00: <math>tDTS = tCK\_INT</math></p> <p>01: <math>tDTS = 2 \times tCK\_INT</math></p> <p>10: <math>tDTS = 4 \times tCK\_INT</math></p> <p>11: 保留, 不要使用这个配置</p>
7	ARPE	RW	0	<p>自动重载预装载允许位</p> <p>0: TIMx_ARR 寄存器没有缓冲</p> <p>1: TIMx_ARR 寄存器被装入缓冲器</p>
6: 5	CMS[1: 0]	RW	00	<p>选择中央对齐模式</p> <p>00: 边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数。</p> <p>01: 中央对齐模式1。计数器交替地向上和向下计数。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 只在计数器向下计数时会置位。</p> <p>10: 中央对齐模式2。计数器交替地向上和向下计数。。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 只在计数器向上计数时会置位。</p> <p>11: 中央对齐模式3。计数器交替地向上和向下计数。配置为输出通道 (TIMx_CCMRx 寄存器中 CCxS=00) 的输出比较中断标志位, 在计数器向上和向下计数时会置位。</p> <p>注: 在计数器开启时 (CEN=1), 不允许从边沿对齐模式转换到中央对齐模式。</p>
4	DIR	RW	0	<p>方向</p> <p>0: 计数器向上计数</p> <p>1: 计数器向下计数</p> <p>注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读</p>

Bit	Name	R/W	Reset Value	Function
3	OPM	RW	0	单脉冲模式 0: 在发生更新事件时, 计数器不停止 1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。 1: 禁止 UEV。不产生更新事件, 影子寄存器 (ARR,PSC,CCR <sub>x</sub> ) 保持它们的值。 如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。
0	CEN	RW	0	允许计数器 0: 禁止计数器 1: 开启计数器 注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

## 22.4.2. TIM2/3 控制寄存器 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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	TI1S	MMS[2: 0]		CCDS	Res	Res	Res

-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	-	-	-
---	---	---	---	---	---	---	---	----	----	----	----	----	---	---	---

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7	TI1S	RW	0	TI1选择 0: TIMx_管脚连到 TI1输入。 1: TIMx_CH1、TIMx_CH2和 TIMx_CH3管脚经异或后连到 TI1输入。
6: 4	MMS[2: 0]	RW	000	主模式选择 这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下: 000: 复位 - TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果触发输入 (复位模式下的从模式控制器) 产生复位, 则 TRGO 上的信号相对实际的复位会有一个延迟。 001: 允许 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要在同一时间启动多个定时器或控制从定时器的一个窗口。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时, TRGO 上会有一个延迟, 除非选择了主/从模式 (见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010: 更新 - 更新事件被选为触发输入 (TRGO)。例如, 一个主定时器的时钟可以被用作一个从定时器的预分频器。 011: 比较脉冲 - 一旦发生一次捕获或一次比较成功时, 当要设置 CC1IF 标志时 (即是它已经为高), 触发输出送出一个正脉冲 (TRGO)。 100: 比较 - OC1REF 信号被用于作为触发输出 (TRGO)。 101: 比较 - OC2REF 信号被用于作为触发输出 (TRGO)。 110: 比较 - OC3REF 信号被用于作为触发输出 (TRGO)。 111: 比较 - OC4REF 信号被用于作为触发输出 (TRGO)。
3	CCDS	RW	0	捕获/比较的 DMA 选择 0: 当发生 CCx 事件时, 送出 CCx 的 DMA 请求。 1: 当发生更新事件时, 送出 CCx 的 DMA 请求。
2: 0	保留	-	-	保留



### 22.4.3. TIM2/3 从模式控制寄存器 (TIMx\_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
ETP	ECE	ETPS[1: 0]		ETF[3: 0]				MSM	TS[2: 0]			OCCS	SMS[2: 0]		
RW	RW	RW		RW				RW	RW			RW	RW		

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	ETP	RW	0	外部触发极性 (External trigger polarity) 该位选择是用 ETR 还是 ETR 的反相来作为触发操作 0: ETR 不反相, 高电平或上升沿有效; 1: ETR 被反相, 低电平或下降沿有效。
14	ECE	RW	0	外部时钟使能位 (External clock enable) 该位启用外部时钟模式2 0: 禁止外部时钟模式2; 1: 使能外部时钟模式2。计数器由 ETRF 信号上的任意有效边沿驱动。 注1: 设置 ECE 位与选择外部时钟模式1并将 TRGI 连到 ETRF (SMS=111和 TS=111) 具有相同功效。 注2: 下述从模式可以与外部时钟模式2同时使用: 复位模式, 门控模式和触发模式; 但是, 这时 TRGI 不能连到 ETRF (TS 位不能是 '111')。 注3: 外部时钟模式1和外部时钟模式2同时被使能时, 外部时钟的输入是 ETRF。
13: 12	ETPS[1: 0]	RW	0	外部触发预分频器。外部触发输入信号 ETR 频率必须至多是 TIMxCLK 频率的1/4。可以使能预分频器, 以降低 ETRP 的频率。 00: 预分频器关闭 01: ETRP 频率的2分频 10: ETRP 频率的4分频 11: ETRP 频率的8分频
11: 8	ETF[3: 0]	RW	0	外部触发滤波 (External trigger filter) 这些位定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上, 数字滤波器是一个事件计数器, 它记录到 N 个事件后会产生一个输出的跳变。 0000: 无滤波器, 以 fDTS 采样

Bit	Name	R/W	Reset Value	Function
				0001: 采样频率 $f_{\text{SAMPLING}}=f_{\text{CK\_INT}}$ , $N=2$ 0010: 采样频率 $f_{\text{SAMPLING}}=f_{\text{CK\_INT}}$ , $N=4$ 0011: 采样频率 $f_{\text{SAMPLING}}=f_{\text{CK\_INT}}$ , $N=8$ 0100: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/2$ , $N=6$ 0101: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/2$ , $N=8$ 0110: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/4$ , $N=6$ 0111: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/4$ , $N=8$ 1000: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/8$ , $N=6$ 1001: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/8$ , $N=8$ 1010: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=5$ 1011: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=6$ 1100: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=8$ 1101: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=5$ 1110: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=6$ 1111: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=8$
7	MSM	RW	0	主/从模式 (Master/slave mode) 0: 无作用; 1: 触发输入 (TRGI) 上的事件被延迟了, 以允许在当前定时器 (通过 TRGO) 与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
6: 4	TS	RW	0	触发选择 (Trigger selection) 这3位选择用于同步计数器的触发输入。 000: 内部触发0 (ITR0) <span style="float:right">100: TI1的边沿检测器 (TI1F_ED)</span> 001: 内部触发1 (ITR1) <span style="float:right">101: 滤波后的定时器输入1 (TI1FP1)</span> 010: 内部触发2 (ITR2) <span style="float:right">110: 滤波后的定时器输入2 (TI2FP2)</span> 011: 内部触发3 (ITR3) <span style="float:right">111: 外部触发输入 (ETRF)</span> 更多有关 ITRx 的细节, 参见表22-1。 注: 这些位只能在未用到 (如 SMS=000) 时被改变, 以避免在改变时产生错误的边沿检测。
3	OCCS	RW	0	OCREF 清除选择位 (OCREF clear selection) 这位选择了 OCREF 的清除源。 0: OCREF_CLR_INPUT 连接 OCREF_CLR 输入 1: OCREF_CLR_INPUT 连接 ETRF
2: 0	SMS	RW	0	从模式选择 (Slave mode selection)

Bit	Name	R/W	Reset Value	Function
				<p>当选择了外部信号，触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 (见输入控制寄存器和控制寄存器的说明)</p> <p>000: 关闭从模式 - 如果 CEN=1, 则预分频器直接由内部时钟驱动。</p> <p>001: 编码器模式1 - 根据 TI1FP1的电平, 计数器在 TI2FP2的边沿向上/下计数。</p> <p>010: 编码器模式2 - 根据 TI2FP2的电平, 计数器在 TI1FP1的边沿向上/下计数。</p> <p>011: 编码器模式3 - 根据另一个信号的输入电平, 计数器在 TI1FP1和 TI2FP2的边沿向上/下计数。</p> <p>100: 复位模式 - 选中的触发输入 (TRGI) 的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。</p> <p>101: 门控模式 - 当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的。</p> <p>110: 触发模式 - 计数器在触发输入 TRGI 的上升沿启动 (但不复位), 只有计数器的启动是受控的。</p> <p>111: 外部时钟模式1 - 选中的触发输入 (TRGI) 的上升沿驱动计数器。</p> <p>注: 如果 TI1F_EN 被选为触发输入 (TS=100) 时, 不要使用门控模式。这是因为, TI1F_ED 在每次 TI1F 变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。</p>

表 22-2 TIMx 内部触发连接

从定时器	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM2	TIM1	TIM15	TIM3	TIM14_OC
TIM3	TIM1	TIM2	TIM15	TIM14_OC

#### 22.4.4. TIM2/3 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Re s	TD E	Re s	CC4D E	CC3D E	CC2D E	CC1 DE	UD E	R es	TIE	Re s	CC4I E	CC3I E	CC2IE	CC 1IE	UIE
-	RW	-	RW				-	RW	-	RW					

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	TDE	RW	0	TDE: 允许触发 DMA 请求 0: 禁止触发 DMA 请求 1: 允许触发 DMA 请求
13	保留	-	-	保留
12	CC4DE	RW	0	CC4DE: 允许捕获/比较4的 DMA 请求 0: 禁止捕获/比较4的 DMA 请求 1: 允许捕获/比较4的 DMA 请求
11	CC3DE	RW	0	CC3DE: 允许捕获/比较3的 DMA 请求 0: 禁止捕获/比较3的 DMA 请求 1: 允许捕获/比较3的 DMA 请求
10	CC2DE	RW	0	CC2DE: 允许捕获/比较2的 DMA 请求 0: 禁止捕获/比较2的 DMA 请求 1: 允许捕获/比较2的 DMA 请求
9	CC1DE	RW	0	CC1DE: 允许捕获/比较1的 DMA 请求 0: 禁止捕获/比较1的 DMA 请求 1: 允许捕获/比较1的 DMA 请求
8	UDE	RW	0	UDE: 允许更新的 DMA 请求 0: 禁止更新的 DMA 请求 1: 允许更新的 DMA 请求
7	保留	-	-	保留
6	TIE	RW	0	TIE: 允许触发中断 0: 禁止触发中断 1: 允许触发中断
5	保留	-	-	保留
4	CC4IE	RW	0	CC4IE: 允许捕获/比较4中断 0: 禁止捕获/比较4中断 1: 允许捕获/比较4中断
3	CC3IE	RW	0	CC3IE: 允许捕获/比较3中断 0: 禁止捕获/比较3中断 1: 允许捕获/比较3中断
2	CC2IE	RW	0	CC2IE: 允许捕获/比较2中断 0: 禁止捕获/比较2中断 1: 允许捕获/比较2中断
1	CC1IE	RW	0	CC1IE: 允许捕获/比较1中断

Bit	Name	R/W	Reset Value	Function
				0: 禁止捕获/比较1中断 1: 允许捕获/比较1中断
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

### 22.4.5. TIM2/3 状态寄存器 (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			Res	Res	Res	Res	Res	IC4IF	IC3IF	IC2IF	IC1IF	IC4IR	IC3IR	IC2IR	IC1IR
-			-	-	-	-	-	RC_W0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4OF	CC3OF	CC2OF	CC1OF	Res		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
-			RC_W0				-		RC_W0	-					

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23	IC4IF	RC_W0	0	下降沿捕获4标志 参见 IC1IF 描述。
22	IC3IF	RC_W0	0	下降沿捕获3标志 参见 IC1IF 描述。
21	IC2IF	RC_W0	0	下降沿捕获2标志 参见 IC1IF 描述。
20	IC1IF	RC_W0	0	下降沿捕获1标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件, 该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0'。 0: 无下降沿捕获产生; 1: 发生下降沿捕获事件。
19	IC4IR	RC_W0	0	上升沿捕获4标志 参见 IC1IR 描述。
18	IC3IR	RC_W0	0	上升沿捕获3标志 参见 IC1IR 描述。
17	IC2IR	RC_W0	0	上升沿捕获2标志

Bit	Name	R/W	Reset Value	Function
				参见 IC1IR 描述。
16	IC1IR	RC_W0	0	上升沿捕获1标志 仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0'。 0: 无上升沿捕获产生； 1: 发生上升沿捕获事件。
15: 13	保留	-	-	保留
12	CC4OF	RC_W0	0	捕获/比较4重复捕获标记 (Capture/Compare 4 overcapture flag) 参见 CC1OF 描述。
11	CC3OF	RC_W0	0	捕获/比较3重复捕获标记 (Capture/Compare 3 overcapture flag) 参见 CC1OF 描述。
10	CC2OF	RC_W0	0	捕获/比较2重复捕获标记 (Capture/Compare 2 overcapture flag) 参见 CC1OF 描述。
9	CC1OF	RC_W0	0	捕获/比较1重复捕获标记 (Capture/Compare 1 overcapture flag) 仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。 0: 无重复捕获产生； 1: 计数器的值被捕获到 TIMx_CCR1寄存器时，CC1IF 的状态已经为 '1'。
8: 7	保留	-	-	保留
6	TIF	RC_W0	0	触发器中断标记 (Trigger interrupt flag) 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿）时由硬件对该位置 '1'。它由软件清 '0'。 0: 无触发器事件产生； 1: 触发中断等待响应。
5	保留	-	-	保留
4	CC4IF	RC_W0	0	捕获/比较4中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	RC_W0	0	捕获/比较3中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	RC_W0	0	捕获/比较2中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	捕获/比较1中断标记 (Capture/Compare 1 interrupt flag)

Bit	Name	R/W	Reset Value	Function
				<p>如果通道 CC1配置为输出模式： 当计数器值与比较值匹配后一个时钟周期，该位由硬件置1，但在中心对称模式下除外（参考 TIMx_CR1寄存器的 CMS 位）。它由软件清 '0' 。</p> <p>0: 无匹配发生； 1: TIMx_CNT 的值与 TIMx_CCR1的值匹配。</p> <p>当 TIMx_CCR1的内容大于 TIMx_APR 的内容时，在向上或向上/下计数模式时计数器上溢条件时，或向下计数模式时的计数器下溢条件时，CC1IF 位变高</p> <p>如果通道 CC1配置为输入模式： 当捕获事件发生时该位由硬件置 '1'，它由软件清 '0' 或通过读 TIMx_CCR1清 '0' 。</p> <p>0: 无输入捕获产生； 1: 计数器值已被捕获（拷贝）至 TIMx_CCR1（在 IC1上检测到与所选极性相同的边沿）。</p>
0	UIF	RC_W0	0	<p>更新中断标记（Update interrupt flag） 当产生更新事件时该位由硬件置 '1'。它由软件清 '0' 。</p> <p>0: 无更新事件产生； 1: 更新中断等待响应。当寄存器被更新时该位由硬件置 '1'：</p> <ul style="list-style-type: none"> <li>- 若 TIMx_CR1寄存器的 UDIS=0，当重复计数器数值上溢或下溢时（重复计数器=0时产生更新事件）。</li> <li>- 若 TIMx_CR1寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1时产生更新事件，通过软件对计数器 CNT 重新初始化时。</li> <li>- 若 TIMx_CR1寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。（参考 TIM2/3从模式控制寄存器（TIMx_SMCR））。</li> </ul>

#### 22.4.6. TIM2/3 事件产生寄存器 (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	-	-	-	-	-	-	W	-	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 7	保留	-	-	保留
6	TG	W	0	产生触发事件 该位由软件置1, 用于产生一个触发事件, 由硬件自动清0。 0: 无动作; 1: TIMx_SR 寄存器的 TIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
5	保留	-	-	保留
4	CC4G	W	0	产生捕获/比较4事件 参考 CC1G 描述
3	CC3G	W	0	产生捕获/比较3事件 参考 CC1G 描述
2	CC2G	W	0	产生捕获/比较2事件 参考 CC1G 描述
1	CC1G	W	0	产生捕获/比较1事件 该位由软件置1, 用于产生一个捕获/比较事件, 由硬件自动清0。 0: 无动作; 1: 在通道 CC1上产生一个捕获/比较事件: 若通道 CC1配置为输出: 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。 若通道 CC1配置为输入: 当前的计数器值捕获至 TIMx_CCR1寄存器, 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。若 CC1IF 已经为1, 则设置 CC1OF=1。
0	UG	W	0	产生更新事件 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意预分频器的计数器也被清0 (但是预分频系数不变)。若在中心对称模式下或 DIR=0 (向上计数) 则计数器被清0, 若 DIR=1 (向下计数) 则计数器取 TIMx_ARR 的值。

#### 22.4.7. TIM2/3 捕获/比较模式寄存器 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

Output compare mode:



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Res																			
-																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
OC2CE		OC2M[2: 0]		OC2PE		CO2FE		CC2S[1: 0]		OC1CE		OC1M[2: 0]		OC1PE		OC1FE		CC1S[1: 0]	
IC2F[3: 0]				IC2PSC[1: 0]				0]		IC1F[3: 0]				IC1PSC[1: 0]				0]	
RW																			

### Output compare mode

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	OC2CE	RW	0	输出比较2清0使能
14: 12	OC2M[2: 0]	RW	000	输出比较2模式选择
11	OC2PE	RW	0	输出比较2预装载使能
10	OC2FE	RW	0	输出比较2快速使能
9: 8	CC2S[1: 0]	RW	00	捕获/比较2选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00: CC2通道被配置为输出； 01: CC2通道被配置为输入，IC2映射在 TI2上； 10: CC2通道被配置为输入，IC2映射在 TI1上； 11: CC2通道被配置为输入，IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIMx_SMCR 寄存器的 TS 位选择）。 注：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E=0）才是可写的。
7	OC1CE	RW	0	输出比较1清0使能 0: OC1REF 不受 ETRF 输入的影响； 1: 一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
6: 4	OC1M[2: 0]	RW	00	输出比较1模式 该位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 000: 冻结。输出比较寄存器 TIMx_CCR1与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时，强制 OC1REF 为高。 010: 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时，强制 OC1REF 为低。

Bit	Name	R/W	Reset Value	Function
				<p>011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。</p> <p>100: 强制为无效电平。强制 OC1REF 为低。</p> <p>101: 强制为有效电平。强制 OC1REF 为高。</p> <p>110: PWM 模式1 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为有效电平, 否则为无效电平; 在向下计数时, 一旦 TIMx_CNT&gt;TIMx_CCR1时通道 1 为无效电平 (OC1REF=0), 否则为有效电平 (OC1REF=1)。</p> <p>111: PWM 模式2 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为无效电平, 否则为有效电平; 在向下计数时, 一旦 TIMx_CNT&gt;TIMx_CCR1时通道1为有效电平, 否则为无效电平。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 在 PWM 模式1或 PWM 模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较1预装载使能</p> <p>0: 禁止 TIMx_CCR1寄存器的预装载功能, 可随时写入 TIMx_CCR1寄存器, 且新值马上起作用。</p> <p>1: 开启 TIMx_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 仅在单脉冲模式下, 可以在未确认预装载寄存器情况下使用 PWM 模式, 否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较1 快速使能</p> <p>该位用于加快 CC 输出对触发器输入事件的响应。</p> <p>0: 根据计数器与 CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活 CC1输出的最小延时为5个时钟周期。</p> <p>1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1输出间的延时被缩短为3个时钟周期。</p>

Bit	Name	R/W	Reset Value	Function
				OCFE 的只在通道被配置成 PWM1 或 PWM2 模式时起作用。
1: 0	CC1S[1: 0]	RW	00	<p>捕获/比较1 选择。</p> <p>这2位定义通道的方向（输入/输出），及输入脚的选择：</p> <p>00: CC1通道被配置为输出；</p> <p>01: CC1通道被配置为输入，IC1映射在 TI1上；</p> <p>10: CC1通道被配置为输入，IC1映射在 TI2上；</p> <p>11: CC1通道被配置为输入，IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。</p> <p>注：CC1S 仅在通道关闭时（TIMx_CCER 寄存器的 CC1E=0）才是可写的。</p>

### 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 12	IC2F[3: 0]	RW	0000	输入捕获2滤波器
11: 10	IC2PSC[1: 0]	RW	00	捕获/比较2预分频器
9: 8	CC2S[1: 0]	RW	0	<p>捕获/比较2选择。</p> <p>这2位定义通道的方向（输入/输出），及输入脚的选择：</p> <p>00: CC2通道被配置为输出；</p> <p>01: CC2通道被配置为输入，IC2映射在 TI2上；</p> <p>10: CC2通道被配置为输入，IC2映射在 TI1上；</p> <p>11: CC2通道被配置为输入，IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。</p> <p>注：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E=0）才是可写的。</p>
7: 4	IC1F[3: 0]	RW	0000	<p>输入捕获1滤波器</p> <p>这几位定义了 TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变：</p> <p>0000: 无滤波器，以 fDTS 采样</p> <p>0001: 采样频率 fSAMPLING=fCK_INT, N=2</p> <p>0010: 采样频率 fSAMPLING=fCK_INT, N=4</p> <p>0011: 采样频率 fSAMPLING=fCK_INT, N=8</p> <p>0100: 采样频率 fSAMPLING=fDTS/2, N=6</p> <p>0101: 采样频率 fSAMPLING=fDTS/2, N=8</p> <p>0110: 采样频率 fSAMPLING=fDTS/4, N=6</p>

Bit	Name	R/W	Reset Value	Function
				0111: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/4$ , $N=8$ 1000: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/8$ , $N=6$ 1001: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/8$ , $N=8$ 1010: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=5$ 1011: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=6$ 1100: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/16$ , $N=8$ 1101: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=5$ 1110: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=6$ 1111: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=8$
3: 2	IC1PSC[1: 0]	RW	00	捕获/比较1预分频器 00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获; 01: 每2个事件触发一次捕获; 10: 每4个事件触发一次捕获; 11: 每8个事件触发一次捕获。
1: 0	CC1S[1: 0]	RW	00	CC1S[1: 0]: 捕获/比较1选择。 这2位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC1映射在 TI1上; 10: CC1通道被配置为输入, IC1映射在 TI2上; 11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E=0) 才是可写的。

### 22.4.8. TIM2/3 捕获/比较模式寄存器 2 (TIMx\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2: 0]		OC4PE	CO4FE	CC4S[1: 0]		OC3CE	OC3M[2: 0]		OC3PE	OC3FE	CC3S[1: 0]			
IC4F[3: 0]			IC4PSC[1: 0]		0]	IC3F[3: 0]			IC3PSC[1: 0]						
RW															

## 输出比较模式

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	OC4CE	RW	0	输出比较4清0使能
14: 12	OC4M[2: 0]	RW	000	输出比较4模式
11	OC4PE	RW	0	输出比较4预装载使能
10	OC4FE	RW	0	输出比较4快速使能
9: 8	CC4S[1: 0]	RW	00	捕获/比较4选择。 该位定义通道的方向（输入/输出），及输入脚的选择： 00: CC4通道被配置为输出； 01: CC4通道被配置为输入，IC4映射在 TI4上； 10: CC4通道被配置为输入，IC4映射在 TI3上； 11: CC4通道被配置为输入，IC4映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIMx_SMCR 寄存器的 TS 位选择）。 注：CC4S 仅在通道关闭时（TIMx_CCER 寄存器的 CC4E=0）才是可写的。
7	OC3CE	RW	0	输出比较3清0使能
6: 4	OC3M[2: 0]	RW	00	输出比较3模式
3	OC3PE	RW	0	输出比较3预装载使能
2	OC3FE	RW	0	输出比较3快速使能
1: 0	CC3S[1: 0]	RW	00	捕获/比较3选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00: CC3通道被配置为输出； 01: CC3通道被配置为输入，IC3映射在 TI3上； 10: CC3通道被配置为输入，IC3映射在 TI4上； 11: CC3通道被配置为输入，IC3映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 （由 TIMx_SMCR 寄存器的 TS 位选择）。 注：CC3S 仅在通道关闭时（TIMx_CCER 寄存器的 CC3E=0）才是可写的。

## 输入捕获模式

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 12	IC4F[3: 0]	RW	0000	输入捕获4滤波器
11: 10	IC4PSC[1: 0]	RW	00	捕获/比较4预分频器
9: 8	CC4S[1: 0]	RW	0	捕获/比较4选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00: CC4通道被配置为输出；

Bit	Name	R/W	Reset Value	Function
				01: CC4通道被配置为输入, IC4映射在 TI4上; 10: CC4通道被配置为输入, IC4映射在 TI3上; 11: CC4通道被配置为输入, IC4映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC4E=0) 才是可写的。
7: 4	IC3F[3: 0]	RW	0000	输入捕获3滤波器
3: 2	IC3PSC[1: 0]	RW	00	捕获/比较3预分频器
1: 0	CC3S[1: 0]	RW	00	捕获/比较3选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC3通道被配置为输出; 01: CC3通道被配置为输入, IC3映射在 TI3上; 10: CC3通道被配置为输入, IC3映射在 TI3上; 11: CC3通道被配置为输入, IC3映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC3E=0) 才是可写的。

### 22.4.9. TIM2/3 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4N	Re	CC4	CC4	CC3N	Re	CC3	CC3	CC2N	Re	CC2	CC2	CC1N	Re	CC1	CC1
P	s	P	E	P	s	P	E	P	s	P	E	P	s	P	E
RW	-	RW	RW	RW	-	RW	RW	RW	-	RW	RW	RW	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15	CC4NP	RW	0	捕获/比较4互补输出极性。参考 CC1NP 的描述。
14	保留	-	-	保留
13	CC4P	RW	0	捕获/比较4输出极性。参考 CC1P 的描述。
12	CC4E	-	0	捕获/比较4输出使能。参考 CC1E 的描述。
11	CC3NP	RW	0	捕获/比较3互补输出极性。参考 CC1NP 的描述。

Bit	Name	R/W	Reset Value	Function
10	保留	-	-	保留
9	CC3P	RW	0	捕获/比较3输出极性。参考 CC1P 的描述。
8	CC3E	RW	0	捕获/比较3输出使能。参考 CC1E 的描述。
7	CC2NP	RW	0	捕获/比较2互补输出极性。参考 CC1NP 的描述。
6	保留	-	-	保留
5	CC2P	RW	0	捕获/比较2输出极性。参考 CC1P 的描述。
4	CC2E	RW	0	捕获/比较2输出使能。参考 CC1E 的描述。
3	CC1NP	RW	0	捕获/比较1互补输出极性 0: OC1N 高电平有效 1: OC1N 低电平有效 这位是和 CC1P 联合使用定义 TI1FP1/TI2FP1极性, 参考 CC1P 描述
2	保留	-	-	保留
1	CC1P	RW	0	捕获/比较1输出极性 CC1通道配置为输出: 0: OC1高电平有效 1: OC1低电平有效 CC1通道配置为输入: CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1和 TI2FP1的极性。 00: 不反相/上升沿: TIxFP1上升沿有效 (捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1不反相 (门控模式、编码器模式)。 01: 反相/下降沿: TIxFP1下降沿有效 (捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1反相 (门控模式、编码器模式)。 10: 保留, 不要使用这个配置。 11: 不反相/双沿 TIxFP1上升和下降沿都有效 (捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1不反相 (门控模式)。这个配置不能应用于编码器模式下。
0	CC1E	RW	0	捕获/比较1输出使能 CC1通道配置为输出: 0: 关闭 - OC1禁止输出, 1: 开启 - OC1信号输出到对应的输出引脚, CC1 通道配置为输入:

Bit	Name	R/W	Reset Value	Function
				该位决定了计数器的值是否能捕获入 TIMx_CCR1寄存器。 0: 捕获禁止; 0: 捕获使能

#### 标准 OCx 通道的输出控制

CcxE 位	OCx output State
0	输出禁止 (OCx=0,OCx_EN=0)
1	OCx=OCxREF+Polarity,OCx_EN=1

### 22.4.10. TIM2/3 计数器 (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31: 16] (仅 TIM2)															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	CNT[31: 16]	RW	0	计数器的值 (仅 TIM2)
15: 0	CNT[15: 0]	RW	0	计数器的值

### 22.4.11. TIM2/3 预分频器 (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PSC[15: 0]	RW	0	预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 fCK_PSC/ ( PSC[15: 0]+1) 。





				<p>如果在 TIMx_CCMR1寄存器 (OC1PE 位) 中未选择预装载特性, 其始终装入当前寄存器中。</p> <p>否则, 只有当更新事件发生时, 此预装载值才装入当前捕获/比较1寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值, 并且在 OC1端口上输出信号。</p> <p>若 CC1通道配置为输入:</p> <p>CCR1包含了由上一次输入捕获1事件 (IC1) 传输的计数器值。</p>
--	--	--	--	---

#### 22.4.14. TIM2/3 捕获/比较寄存器 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31: 16] (仅 TIM2)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR2[31: 16]	RW	0	捕获/比较2的值 (仅 TIM2)
15: 0	CCR2[15: 0]	RW	0	<p>捕获/比较2的值</p> <p>若 CC2通道配置为输出:</p> <p>CCR2包含了装入当前捕获/比较2寄存器的值 (预装载值)。</p> <p>如果在 TIMx_CCMR1寄存器 (OC2PE 位) 中未选择预装载特性, 其始终装入当前寄存器中。</p> <p>否则, 只有当更新事件发生时, 此预装载值才装入当前捕获/比较2寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值, 并且在 OC 端口上输出信号。</p> <p>若 CC2通道配置为输入:</p> <p>CCR2包含了由上一次输入捕获2事件 (IC2) 传输的计数器值。</p>

### 22.4.15. TIM2/3 捕获/比较寄存器 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31: 16] (仅 TIM2)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR3[31: 16]	RW	0	捕获/比较3的值 (仅 TIM2)
15: 0	CCR3[15: 0]	RW	0	<p>捕获/比较3的值</p> <p><b>若 CC3通道配置为输出:</b></p> <p>CCR3包含了装入当前捕获/比较3寄存器的值 (预装载值)。</p> <p>如果在 TIMx_CCMR2寄存器 (OC3PE 位) 中未选择预装载特性, 其始终装入当前寄存器中。</p> <p>否则, 只有当更新事件发生时, 此预装载值才装入当前捕获/比较3寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值, 并且在 OC 端口上输出信号。</p> <p><b>若 CC3通道配置为输入:</b></p> <p>CCR3包含了由上一次输入捕获3事件 (IC3) 传输的计数器值。</p>

### 22.4.16. TIM2/3 捕获/比较寄存器 4 (TIMx\_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31: 16] (仅 TIM2)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR4[31: 16]	RW	0	捕获/比较4的值 (仅 TIM2)
15: 0	CCR4[15: 0]	RW	0	<p>捕获/比较4的值</p> <p><b>若 CC4通道配置为输出:</b></p> <p>CCR4包含了装入当前捕获/比较4寄存器的值 (预装载值)。</p> <p>如果在 TIMx_CCMR2寄存器 (OC4PE 位) 中未选择预装载特性, 其始终装入当前寄存器中。</p> <p>否则, 只有当更新事件发生时, 此预装载值才装入当前捕获/比较4寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值, 并且在 OC 端口上输出信号。</p> <p><b>若 CC4通道配置为输入:</b></p> <p>CCR4包含了由上一次输入捕获4事件 (IC4) 传输的计数器值。</p>

### 22.4.17. TIM2/3 DMA 控制寄存器 (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 13	保留	-	-	保留
12: 8	DBL[4: 0]	RW	0 0000	<p>DMA 连续传送长度</p> <p>这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器的地址进行读或写时, 定时器则进行一次连续传送), 即: 定义被传送的次数:</p> <p>00000: 1次传输</p> <p>00001: 2次传输</p> <p>00010: 3次传输</p> <p>.....</p> <p>.....</p> <p>10001: 18次传输</p> <p>例: 我们考虑这样的传输: DBL=7, DBA=TIMx_CR1</p>

Bit	Name	R/W	Reset Value	Function
				<p>- 如果 DBL=7, DBA=TIMx_CR1表示待传输数据的地址, 那么传输的地址由下式给出:            (TIMx_CR1的地址) + DBA + (DMA 索引), 其中            DMA 索引 = DBL</p> <p>其中 (TIMx_CR1的地址) + DBA 再加上7, 给出了将要写入或者读出数据的地址, 这样数据的传输将发生在从地址 (TIMx_CR1的地址) + DBA 开始的7个寄存器。            根据 DMA 数据长度的设置, 可能发生以下情况:</p> <p>- 如果设置数据为半字 (16位), 那么数据就会传输给全部7个寄存器。</p> <p>- 如果设置数据为字节, 数据仍然会传输给全部7个寄存器: 第一个寄存器包含第一个 MSB 字节, 第二个寄存器包含第一个 LSB 字节, 以此类推。因此对于定时器, 用户必须指定由 DMA 传输的数据宽度。</p>
7: 5	保留	-	-	保留
4: 0	DBA[4: 0]	RW	0 0000	<p>DBA[4: 0]: DMA 基地址</p> <p>这些位定义了 DMA 在连续模式下的基地址 (当对 TIMx_DMAR 寄存器的地址进行读或写时), DBA 定义为从 TIMx_CR1寄存器所在地址开始的偏移量:</p> <p>00000: TIMx_CR1,            00001: TIMx_CR2,            00010: TIMx_SMCR,            .....</p>

#### 22.4.18. TIM2/3 连续模式的 DMA 地址 (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DMAB[31: 0]	RW	0	<p>DMA 连续传送寄存器 (DMA register for burst accesses)</p> <p>对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作:</p> <p><math>TIMx\_CR1地址 + (DBA + DMA 索引) \times 4</math>, 其中:</p> <p>“TIMx_CR1地址” 是控制寄存器1 (TIMx_CR1) 所在的地址;</p> <p>“DBA” 是 TIMx_DCR 寄存器中定义的基地址;</p> <p>“DMA 索引” 是由 DMA 自动控制的偏移量, 它取决于 TIMx_DCR 寄存器中定义的 DBL。</p>

## 23. 基本定时器 (TIM6/7)

### 23.1. TIM6和 TIM7简介

基本定时器 TIM6 和 TIM7 各包含一个 16 位自动重载计数器，由各自的可编程预分频器驱动。

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

TIM6 和 TIM7 都是完全独立的，没有互相共享任何资源。

### 23.2. TIM6和 TIM7的主要特性

- 16 位自动重载向上计数器
- 16 位可编程 (可以实时修改) 预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 触发 DAC 的同步电路
- 在更新事件发生时产生中断/DMA

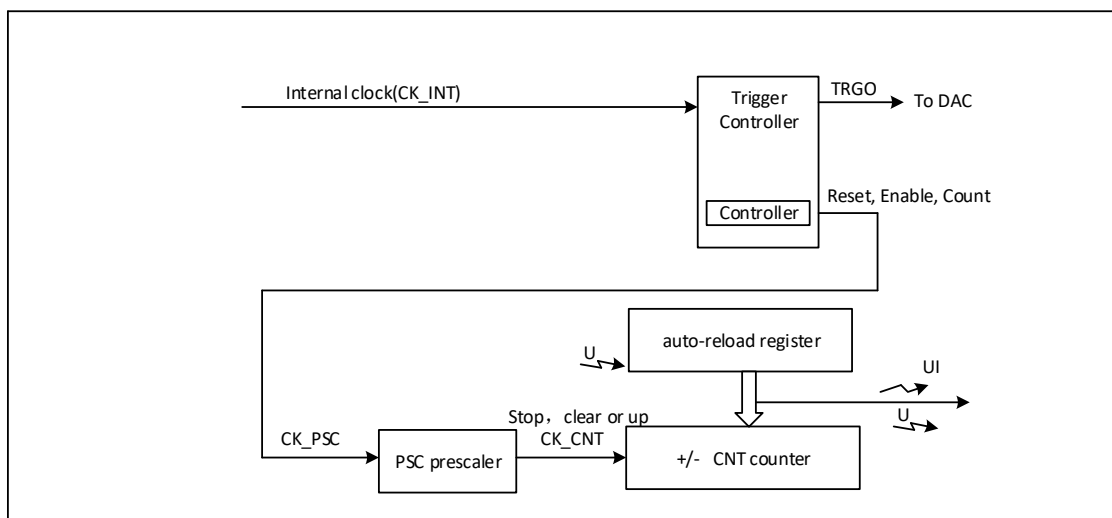


图 23-1 基本定时器架构框图

### 23.3. TIM6和 TIM7功能描述

#### 23.3.1. 时基单元

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

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

时基单元包括：

- 计数器寄存器 (TIMx\_CNT)

- 预分频寄存器 (TIMx\_PSC)
- 自动重载寄存器 (TIMx\_ARR)

自动重载寄存器是预先装载的，写或者读自动重载寄存器将访问预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动重载预装载使能位 (ARPE) 的设置，预装载寄存器的内容一直或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢出并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIM14\_CR1 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

### 预分频器描述：

预分频器可以将计数器的时钟按 1 到 65536 之间的任意值分频。它是一个基于（在 TIMx\_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下次更新事件到来时被采用。

下图给出了在计数器运行时，更改预分频器参数的例子。

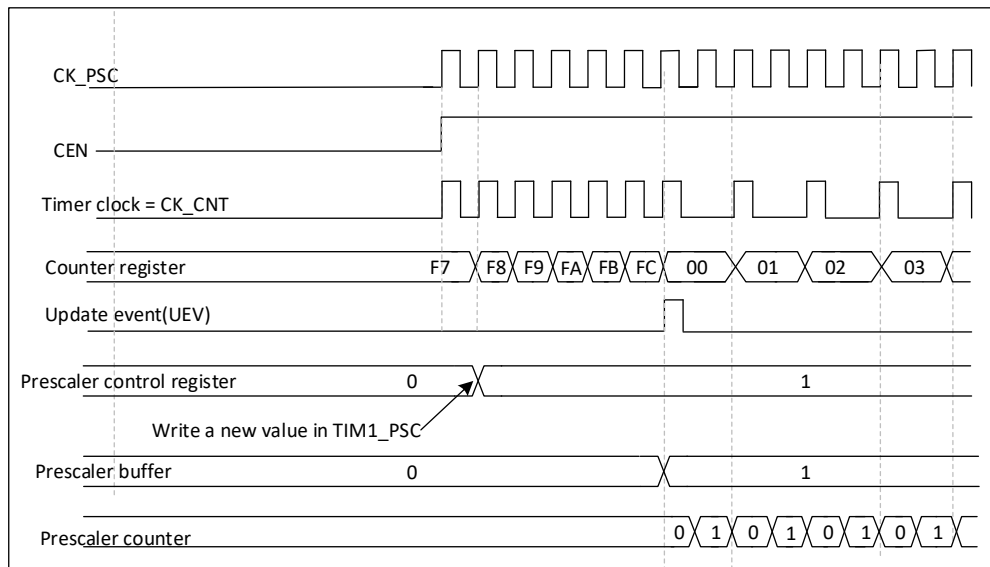


图 23-2 当预分频器的参数从 1 变到 2 时，计数器的时序图



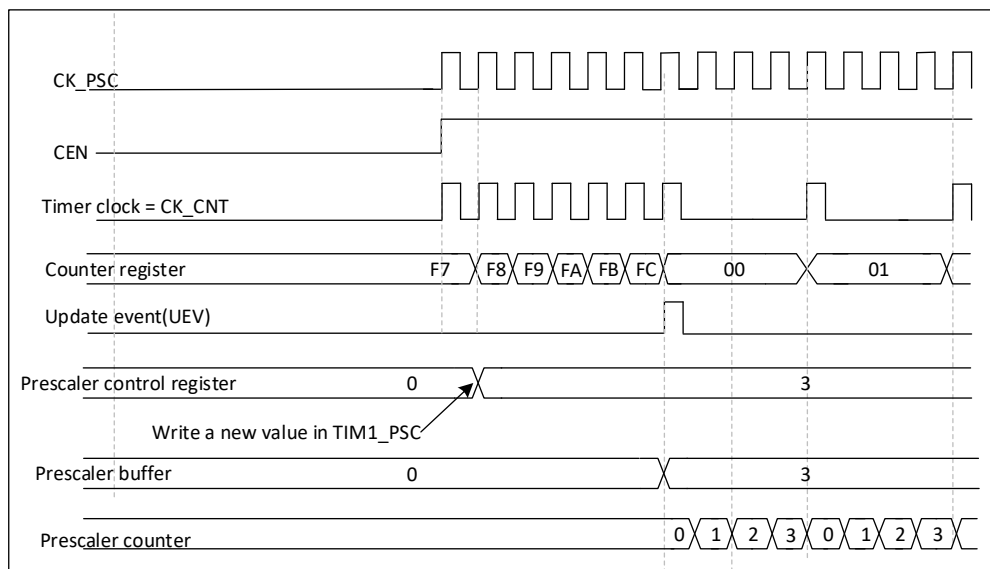


图 23-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 向上计数模式

计数器从 0 计数到自动重载值（TIMx\_ARR 寄存器的值），然后又从 0 重新开始计数，并产生一个计数器上溢事件。

在 TIMx\_EGR 寄存器中（通过软件方式）设置 UG 位也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。虽然如此，但是计数器依旧从 0 开始，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 URS 位）设置更新标志位（TIMx\_SR 寄存器中的 UIF 位）。

- 自动重载影子寄存器被重新置入预装载寄存器的值（TIMx\_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIMx\_PSC 寄存器的内容）。

下面的例程显示了几个在不同频率下的计数器行为，当 TIMx\_ARR=0X36。

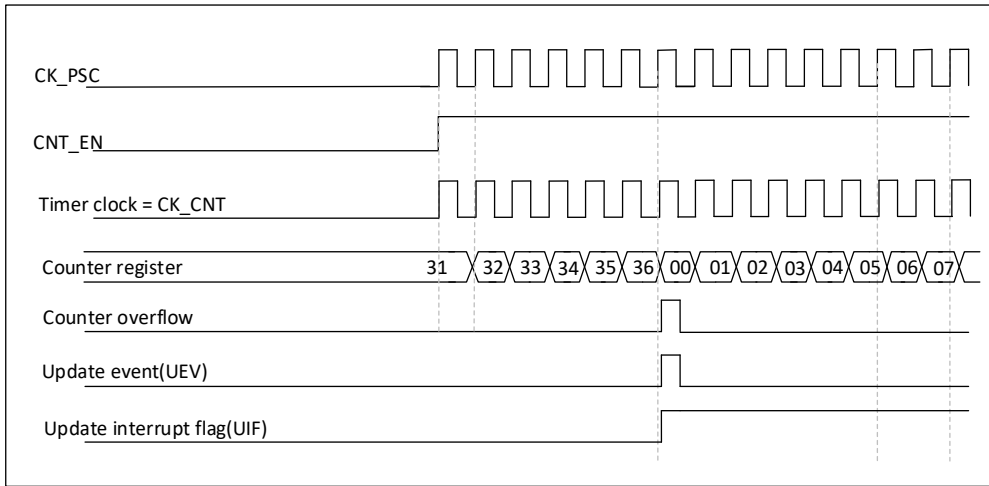


图 23-4 计数器时序图，内部时钟分频因子为 1

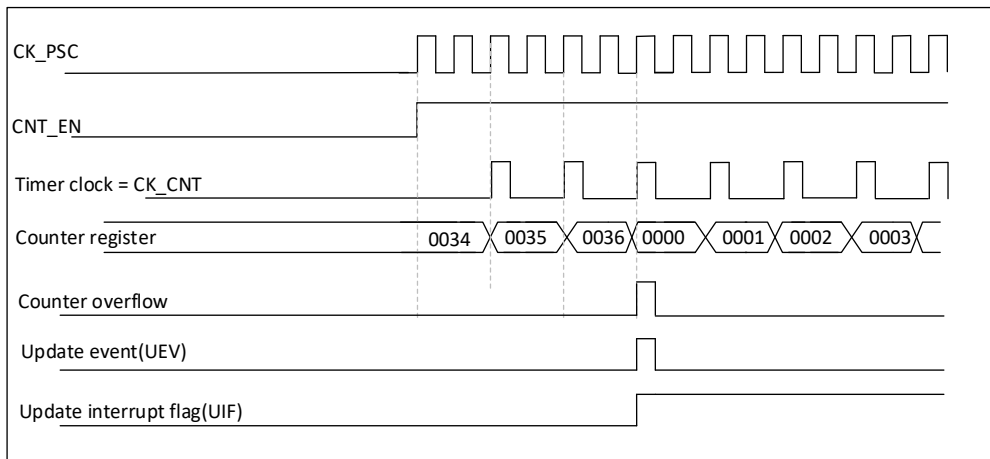


图 23-5 计数器时序图，内部时钟分频因子为 2

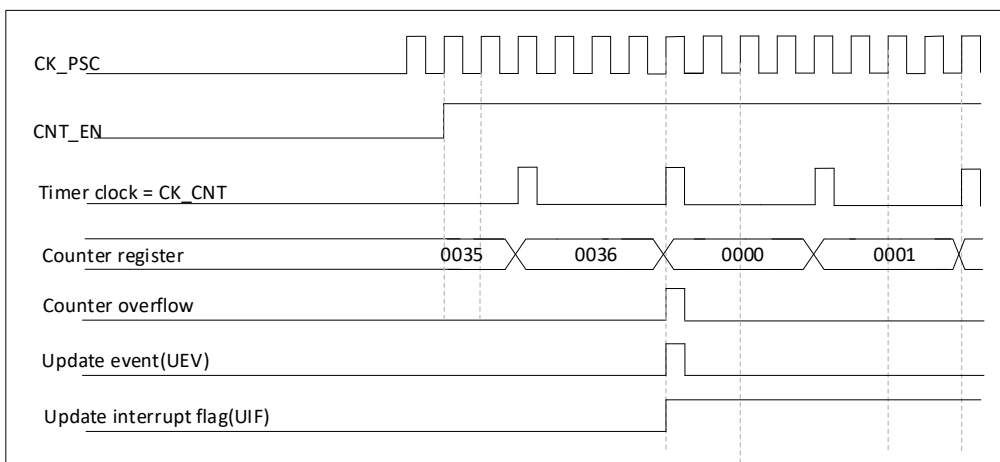


图 23-6 计数器时序图，内部时钟分频因子为 4

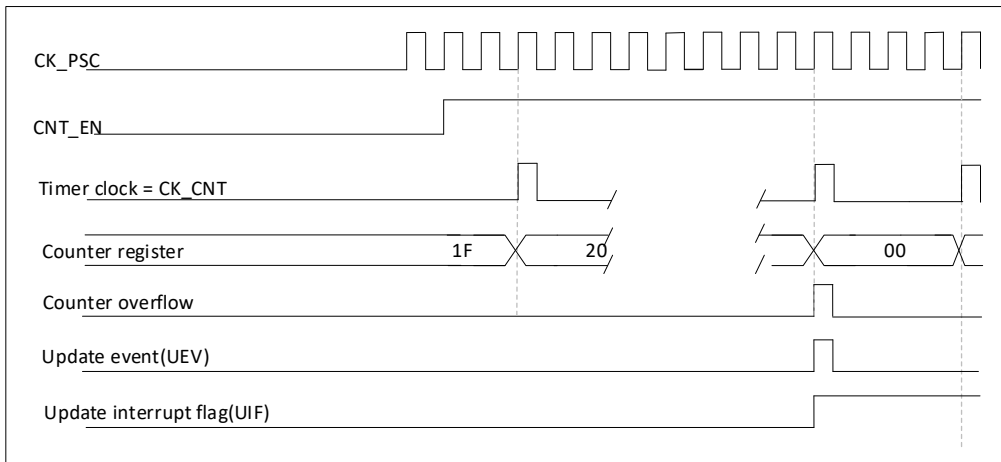


图 23-7 计数器时序图，内部时钟分频因子为 N

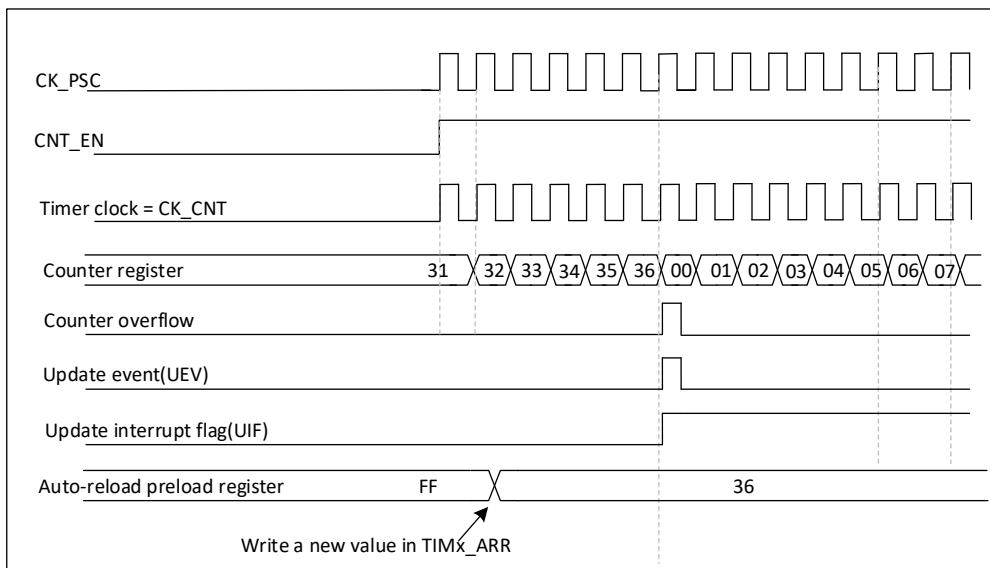


图 23-8 计数器时序图，当 ARPE=0 时的更新事件 (TIMx\_ARR 没有预装入)

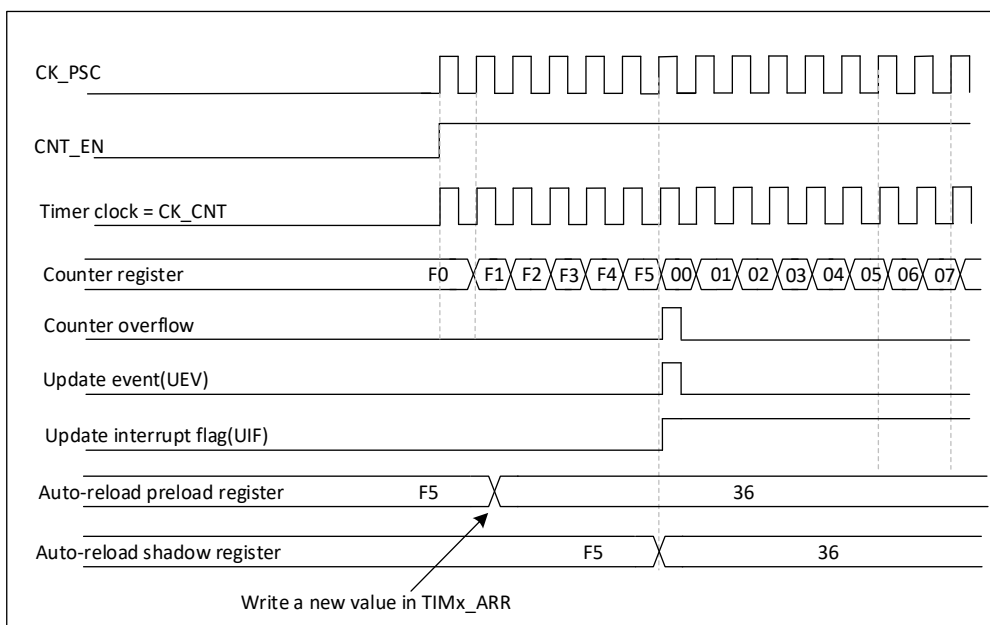


图 23-9 计数器时序图，当 ARPE=1 时的更新事件 (预装入了 TIMx\_ARR)

### 23.3.2. 时钟源

计数器的时钟由内部时钟 (CK\_INT) 提供。TIMx\_CR1 寄存器的 CEN 位和 TIMx\_EGR 寄存器的 UG 位是实际的控制位 (除了 UG 位被自动清除外)，只能通过软件改变他们。一旦置 CEN 位为 1，内部时钟即向分频器提供时钟。

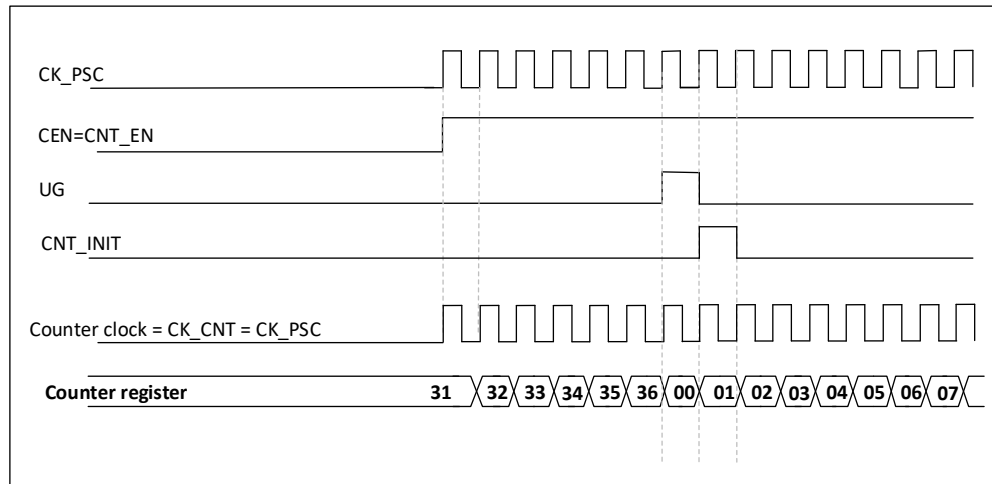


图 23-10 一般模式下的控制电路，内部时钟分频因子为 1

### 23.3.3. 调试模式

当芯片进入调试模式 (M0+停止)，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。

## 23.4. TIM6和 TIM7寄存器

### 23.4.1. TIM6 和 TIM7 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7	ARPE	RW	0	自动重装载预装载允许位 0: TIMx_ARR 寄存器没有缓冲 1: TIMx_ARR 寄存器被装入缓冲器
6: 4	保留	-	-	保留
3	OPM	RW	0	单脉冲模式 (One pulse mode) 0: 在发生更新事件时, 计数器不停止; 1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的生产 0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。

Bit	Name	R/W	Reset Value	Function
				1: 禁止 UEV。不产生更新事件，影子寄存器 (ARR,PSC,CCRx) 保持它们的值。 如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	RW	0	允许计数器 0: 禁止计数器 1: 开启计数器 注：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

### 23.4.2. TIM6 和 TIM7 控制寄存器 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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	MMS[2: 0]	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	RW	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 7	保留	-	-	保留
6: 4	MMS[2: 0]	RW	0	主模式选择 这3位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下： 000: 复位 - TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果是触发输入产生的复位 (从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。 001: 使能 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式 (见 TIMx_SMCR 寄存器中 MSM 位的描述)。

Bit	Name	R/W	Reset Value	Function
				010: 更新 - 更新事件被选为触发输入 (TRGO)。例如, 一个主定时器的时钟可以被用作一个从定时器的预分频器。 注意: 从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号, 并在接收时不要改变。
3: 0	保留	-	-	保留

### 23.4.3. TIM6 和 TIM7 DMA/中断使能寄存器 (TIM14\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 9	保留	-	-	保留
8	UDE	RW	0	UDE: 允许更新的 DMA 请求 0: 禁止更新的 DMA 请求 1: 允许更新的 DMA 请求
7: 1	保留	-	-	保留
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

### 23.4.4. TIM6 和 TIM7 状态寄存器 (TIMx\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 1	保留	-	-	保留
0	UIF	RC_W0	0	更新中断标记，当产生更新事件时该位由硬件置1。它由软件清0。 0: 无更新事件产生； 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1： - 若 TIMx_CR1寄存器的 UDIS=0，产生更新事件上溢； - 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1时产生更新事件（软件对 CNT 重新初始化）；

### 23.4.5. TIM6 和 TIM7 事件产生寄存器 (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	保留	-	-	保留
0	UG	W	0	产生更新事件，该位由软件置1，由硬件自动清0。 0: 无动作； 1: 重新初始化计数器，并产生一个寄存器的更新事件。注意预分频器的计数器也被清0（但是预分频系数不变）。

### 23.4.6. TIM6 和 TIM7 计数器 (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CNT[15: 0]	RW	0	计数器的值

### 23.4.7. TIM6 和 TIM7 预分频器 (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PSC[15: 0]	RW	0	<p>预分频器的值</p> <p>计数器的时钟频率 (CK_CNT) 等于 <math>f_{CK\_PSC} / (PSC[15: 0] + 1)</math>。</p> <p>PSC 包含了当更新事件产生时装入当前预分频器寄存器的值; 更新事件包括计数器被 TIM_EGR 的 UG 位清0或被工作在复位模式的从控制器清0。</p>

### 23.4.8. TIM6 和 TIM7 自动重载寄存器 (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ARR[15: 0]	RW	FFFF	<p>自动重载的值</p> <p>ARR 包含了将要装载入实际的自动重载寄存器的值。</p>

---

				详细参考23.3.1：时基单元有关 ARR 的更新和动作。 当自动重载的值为空时，计数器不工作。
--	--	--	--	---

## 24. 通用定时器 (TIM14)

### 24.1. TIM14简介

通用定时器 TIM14 由可编程预分频器驱动的 16 位自动重载计数器构成。

它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

通用定时器 (TIMx) 都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见 TIM2/3 的同步章节。

### 24.2. TIM14主要特性

- 16位自动重载向上计数器
- 16位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为1 ~ 65536之间的任意数值
- 1个独立通道，作为：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿对齐模式）

如下事件发生时产生中断

- 更新：计数器向上溢出，计数器初始化（通过软件）
- 输入捕获
- 输出比较

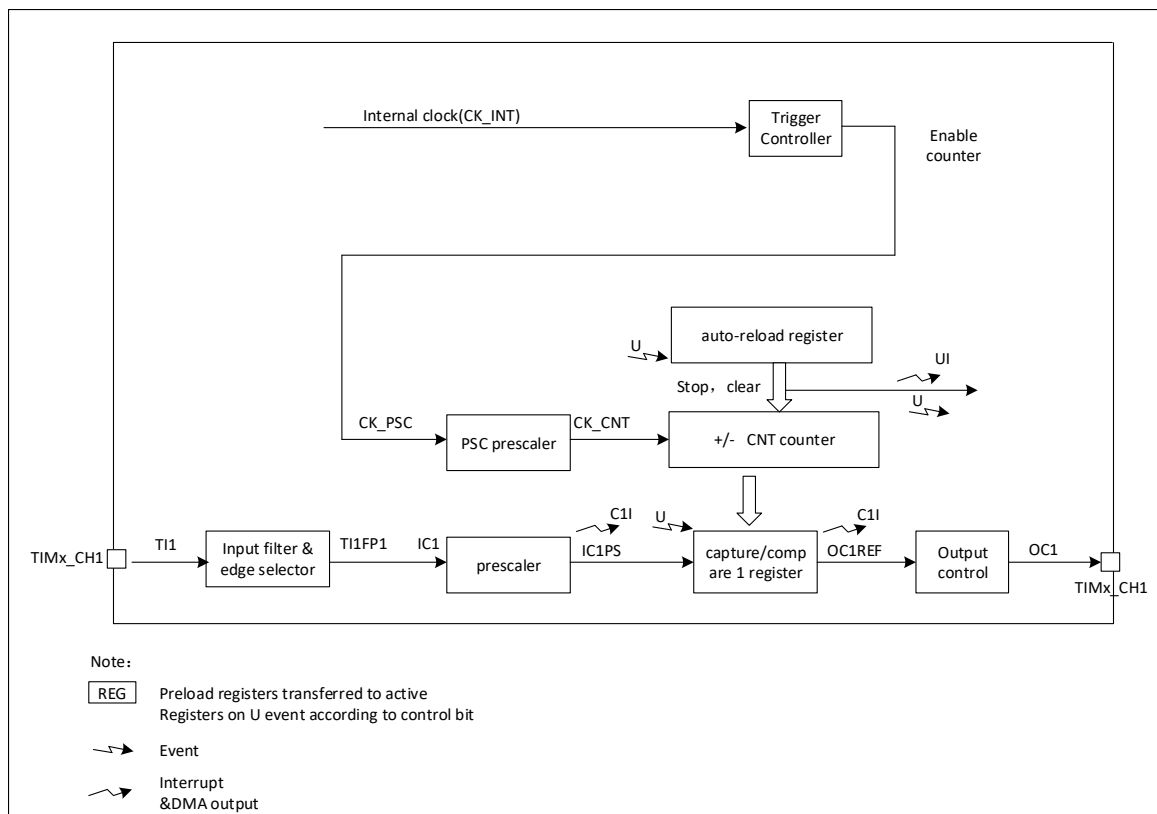


图 24-1 TIM14架构框图

## 24.3. TIM14功能描述

### 24.3.1. 时基单元

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

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

时基单元包括：

- 计数器寄存器 (TIM14\_CNT)
- 预分频寄存器 (TIM14\_PSC)
- 自动重载寄存器 (TIM14\_ARR)

自动重载寄存器是预先装载的，写或者读自动重载寄存器将访问预装载寄存器。根据在 TIM14\_CR1 寄存器中的自动重载预装载使能位 (ARPE) 的设置，预装载寄存器的内容一直或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢出并当 TIM14\_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIM14\_CR1 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。

注意，在设置了 TIM14\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

#### 预分频器描述：

预分频器可以将计数器的时钟按 1 到 65536 之间的任意值分频。它是一个基于（在 TIM14\_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下次更新事件到来时被采用。

下图给出了在计数器运行时，更改预分频器参数的例子。

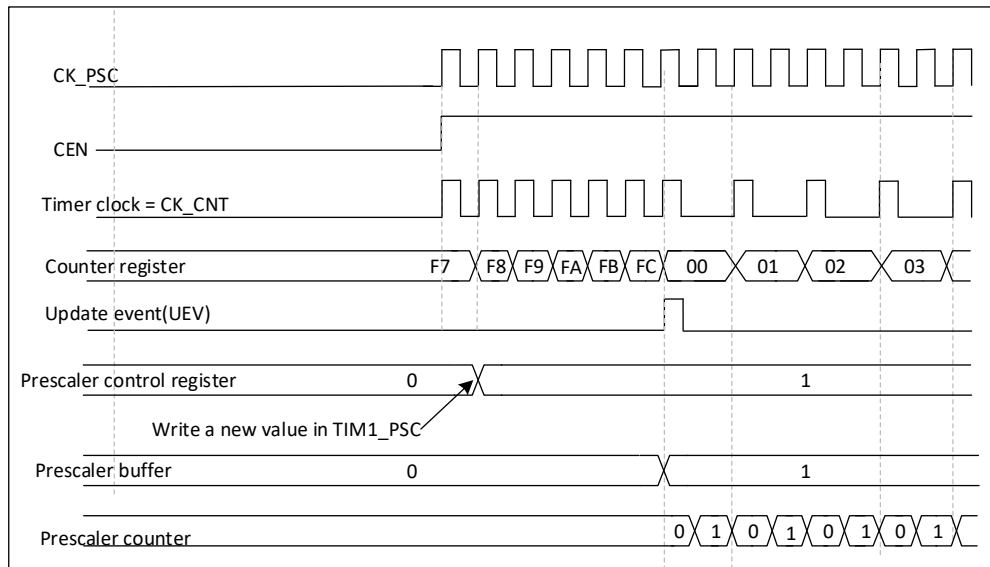


图 24-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

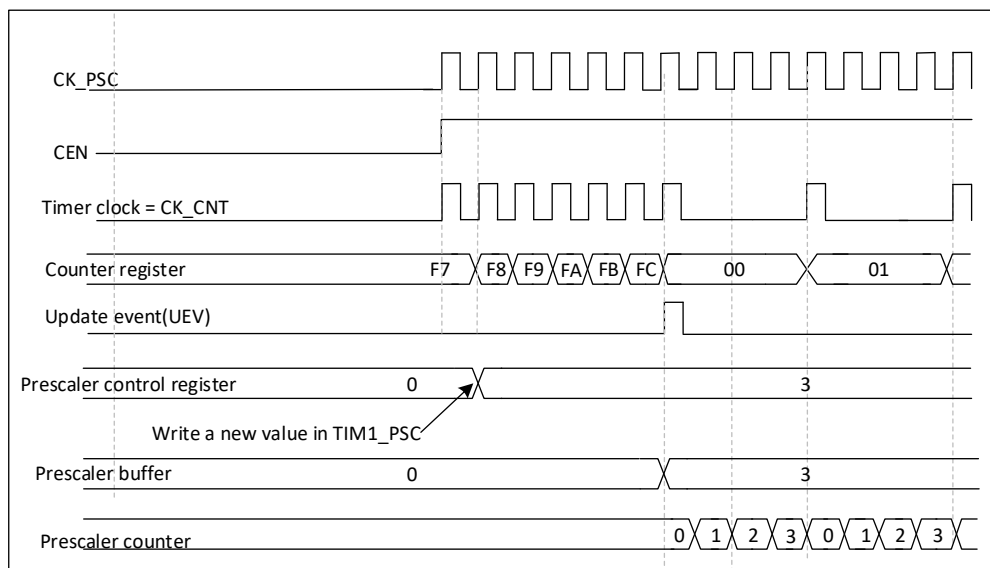


图 24-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 向上计数模式

计数器从 0 计数到自动重载值（TIM14\_ARR 寄存器的值），然后又从 0 重新开始计数，并产生一个计数器上溢事件。

在 TIM14\_EGR 寄存器中（通过软件方式）设置 UG 位也同样可以产生一个更新事件。

设置 TIM14\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不会产生更新事件。虽然如此，但是计数器依旧从 0 开始，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TIM14\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标

志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 URS 位）设置更新标志位（TIM14\_SR 寄存器中的 UIF 位）。

- 自动重载影子寄存器被重新置入预装载寄存器的值（TIM14\_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIM14\_PSC 寄存器的内容）。

下面的例程显示了几个在不同频率下的计数器行为，当 TIMx\_ARR=0X36.

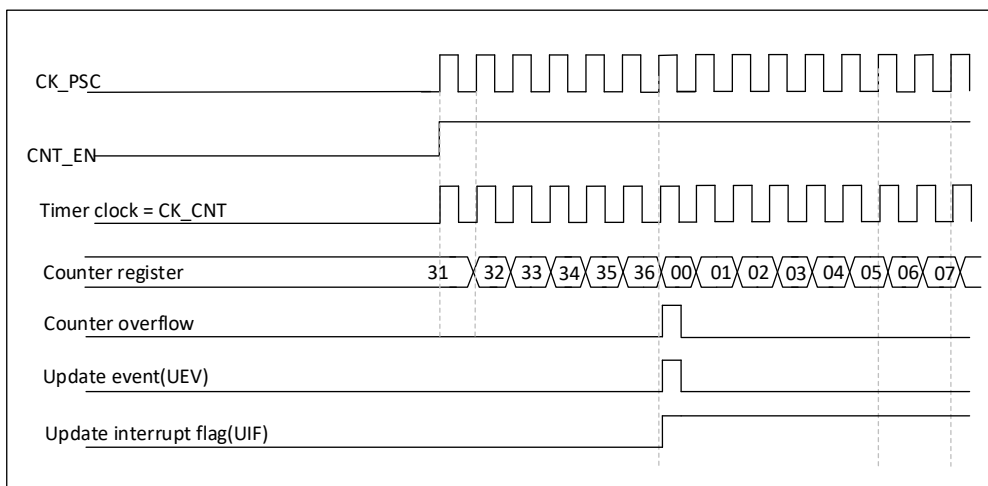


图 24-4 计数器时序图，内部时钟分频因子为 1

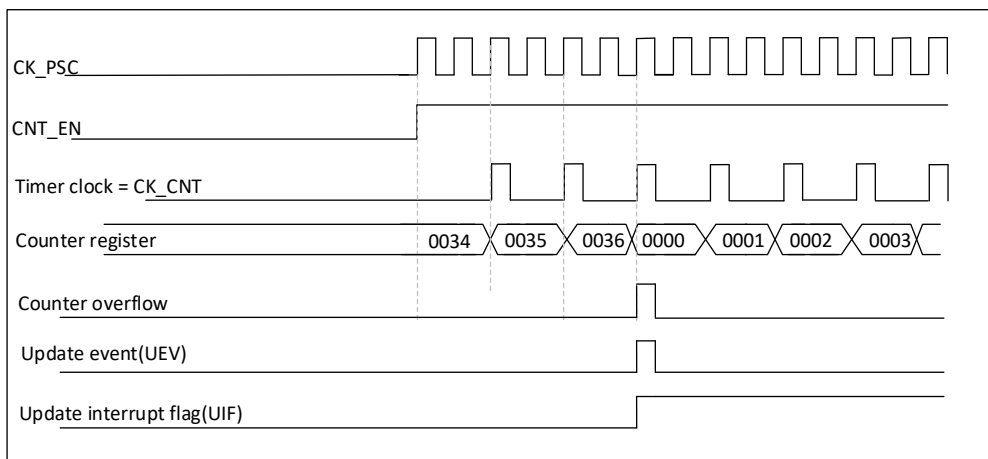


图 24-5 计数器时序图，内部时钟分频因子为 2

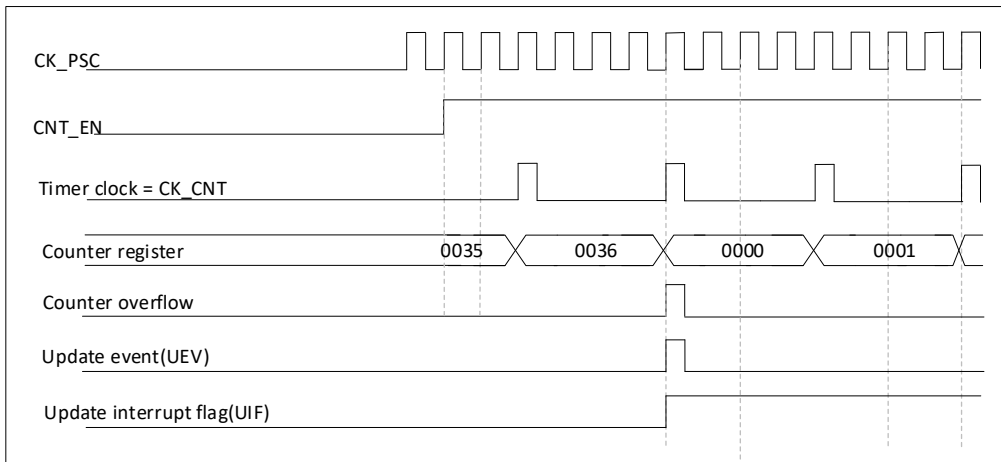


图 24-6 计数器时序图，内部时钟分频因子为 4

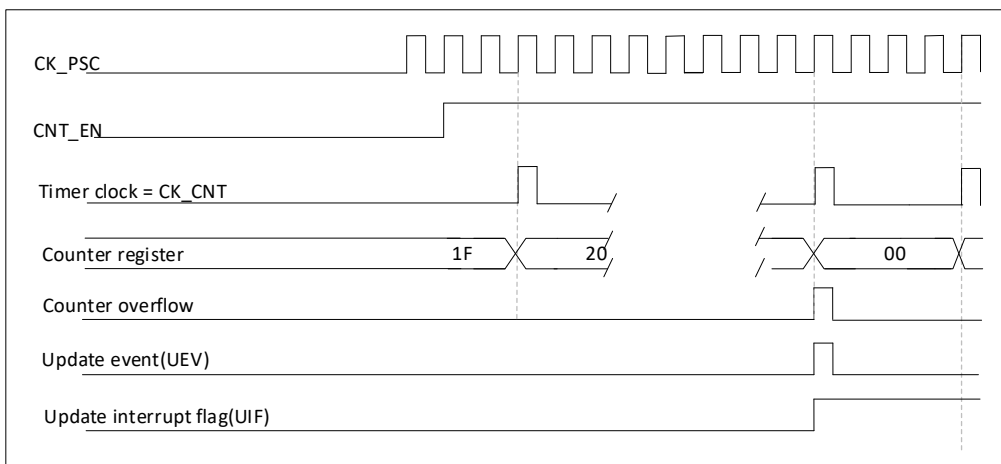


图 24-7 计数器时序图，内部时钟分频因子为 N

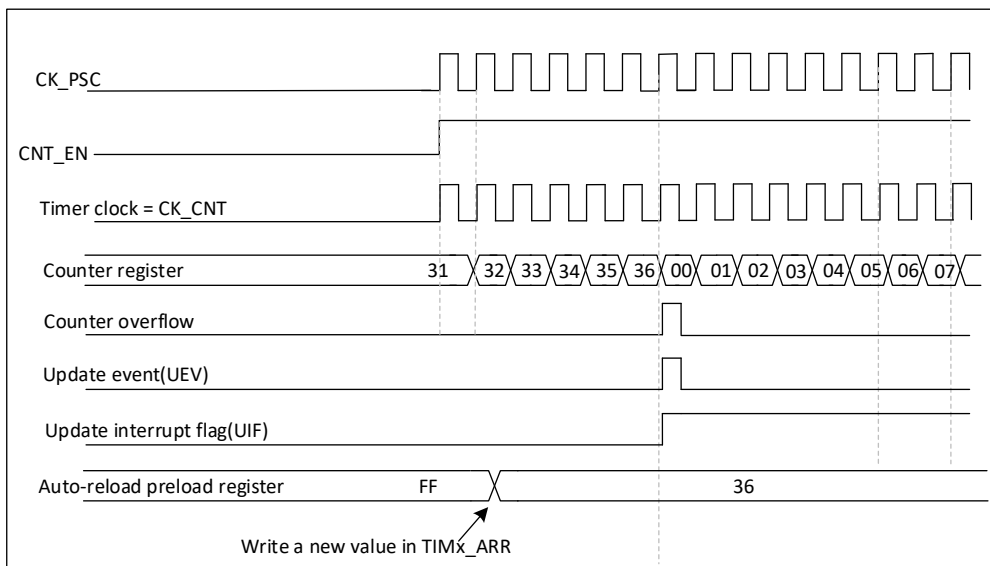


图 24-8 计数器时序图，当 ARPE=0 时的更新事件 (TIMx\_ARR 没有预装入)

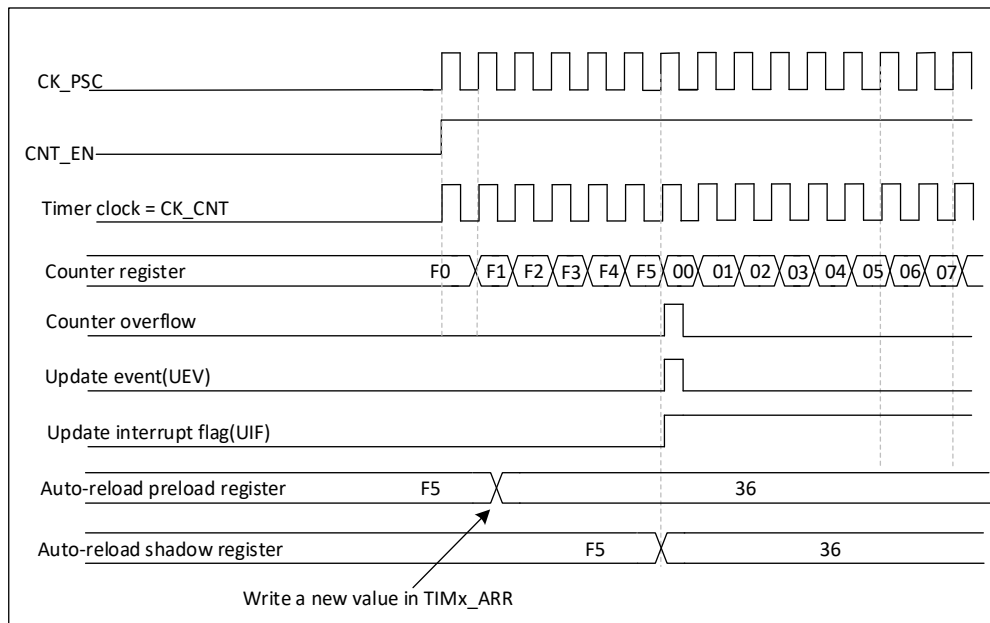


图 24-9 计数器时序图, 当 ARPE=1 时的更新事件 (预装入了 TIMx\_ARR)

### 24.3.2. 时钟源

计数器的时钟由内部时钟 (CK\_INT) 提供。TIMx\_CR1 寄存器的 CEN 位和 TIM14\_EGR 寄存器的 UG 位是实际的控制位 (除了 UG 位被自动清除外), 只能通过软件改变他们。一旦置 CEN 位为 1, 内部时钟即向分频器提供时钟。

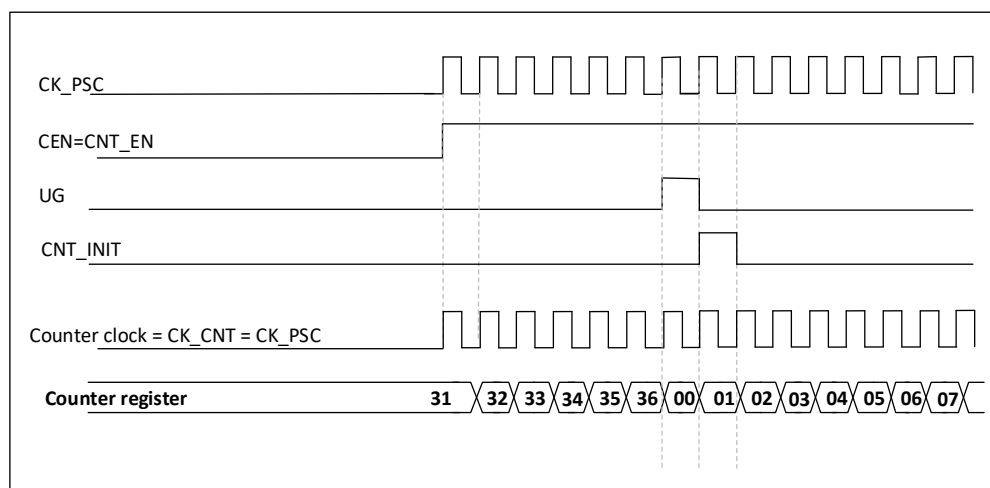


图 24-10 一般模式下的控制电路, 内部时钟分频因子为 1

### 24.3.3. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器 (包含影子寄存器), 包括捕获的输入部分 (数字滤波、多路复用和预分频器), 和输出部分 (比较器和输出控制)。



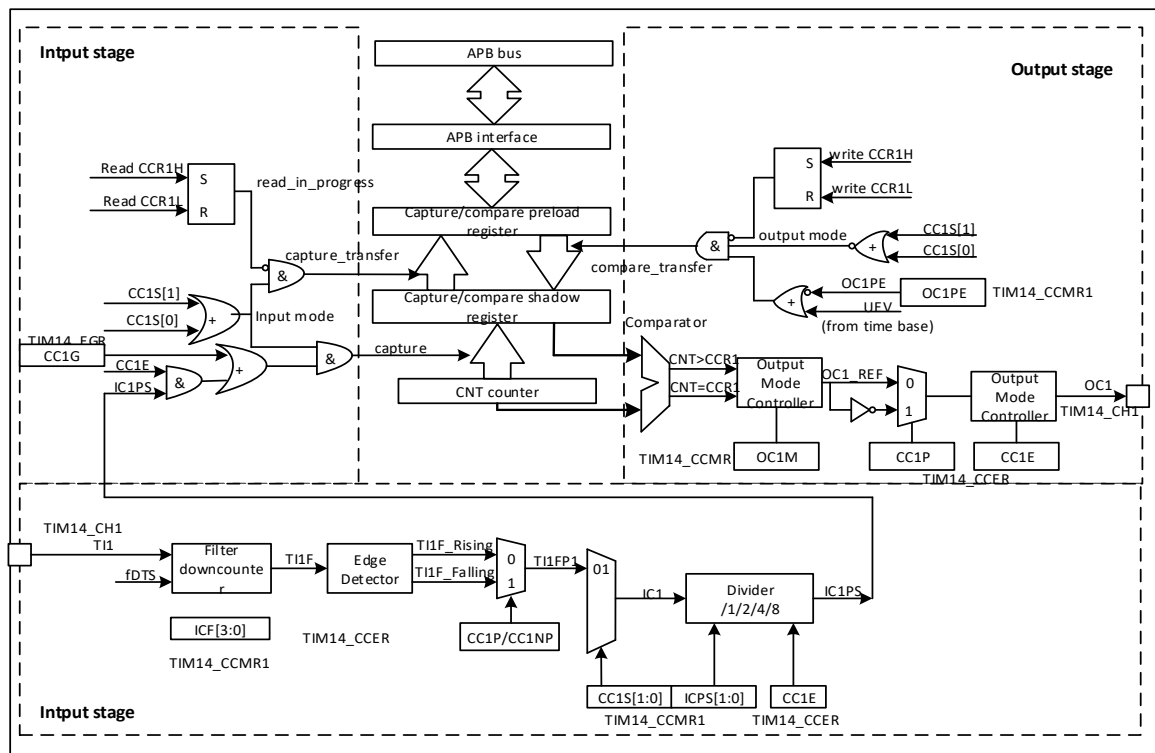


图 24-11 TIM14 捕获/比较通道图

输入部分对相应的  $Ti_x$  输入信号采样，并产生一个滤波后的信号  $Ti_xF$ 。然后，一个带极性选择的边缘检测器产生一个信号 ( $Ti_xFP_x$ )，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 ( $IC_xPS$ )。

输出部分产生一个中间波形（高有效）作为基准，链的末端决定最终输出信号的极性。

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

#### 24.3.4. 输入捕获模式

在输入捕获模式下，当检测到  $IC_x$  信号相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器 ( $TIM_x\_CCRx$ ) 中。当捕获事件发生时，相应的  $CC_xIF$  标志 ( $TIM14\_SR$  寄存器) 被置 1。如果捕获事件发生时， $CC_xIF$  标志已经为高，那么重复捕获标志  $CC_xOF$  ( $TIM_x\_SR$  寄存器) 被置 1。写  $CC_xIF$  可清除  $CC_xIF$ ，或读取存储中的捕获数据也可清除  $CC_xIF$ 。写  $CC_xOF=0$  可清除  $CC_xOF$ 。

以下例子说明如何在  $Ti1$  输入的上升沿时捕获计数器的值到  $TIM14\_CCR1$  寄存器中，步骤如下：

- 选择有效输出端：  $TIM14\_CCR1$  必须连接到  $Ti1$  输入，所以写入  $TIM14\_CCMR1$  寄存器中的  $CC1S=01$ ，只要  $CC1S$  不为  $00$  时，通道被配置为输入，并且  $TIM14\_CCR1$  寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的宽度（即输入为  $Ti_x$  时，输入滤波器控制位是  $TIM14\_CCMR_x$  寄存器中的  $IC_xF$  位）。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们必须配置滤波器的宽度长于 5 个时钟周期。因此，我们可（以  $fDTS$  频率）连续采样 8 次，已确认在  $Ti1$  上一次真是的边沿变化，然后再  $TIM14\_CCMR1$  寄存器中写入  $IC1F=0011$ 。
- 选择  $Ti1$  通道的有效转换边沿，在  $TIM_x\_CCER$  寄存器中写入  $CC1P=0$ （上升沿）（和  $CC1NP=0$ ）

- 配置输入预分频器。在这个例子中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx\_CCMR1 寄存器的 IC1PS=00）。
- 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。  
如果需要，通过设置 TIMx\_DIER 寄存器中的 CC1E 位允许相关中断请求

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIM14\_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少2个连续捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断请求。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：输入捕获中断请求能通过软件设置在 TIMx\_EGR 中相应的 CCxG 位来产生。

### 24.3.5. 强置输出模式

在该模式下（TIM14\_CCMRx 寄存器中 CCxS bits = 00）下，输出比较信号（OCxREF 和相应的 OCx）能够直接由软件置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

写 TIM14\_CCMRx 寄存器中相应的 OCxM=101，即可强制输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CCxP 极性位相反的值。

例如：CCxP=0（OCx 高电平有效），则 OCx 被强置为高电平。

置 TIM14\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIM14\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。这将会在下面的输出比较模式一节中介绍。

### 24.3.6. 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较工作做如下操作：

- 将输出比较模式（TIM14\_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx\_CCER 寄存器中的 CCxP 位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxM=000）、被设置成有效电平（OCxM=001）、被设置成无效电平（OCxM=010）或进行翻转（OCxM=011）。
- 设置中断状态寄存器中的标志位（TIMx\_SR 寄存器中的 CcxIF 位）。
- 若设置了相应的中断屏蔽（TIM14\_DIER 寄存器中的 CCxIE 位），则产生一个中断。

可通过配置 TIM14\_CCMRx 中的 OCxPE 位选择 TIM14\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPE= '0'，否则 TIMx\_CCRx 的影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

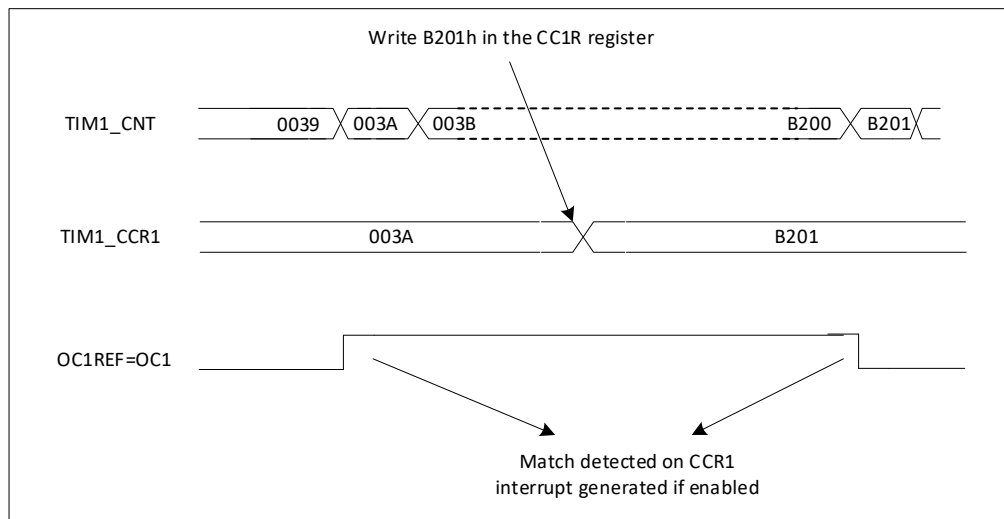


图 24-12 输出比较模式，翻转 OC1

### 24.3.7. PWM 模式

脉冲宽度调节模式可以产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIM14\_CCMRx 寄存器中的 OCxM 位写入 “110” (PWM 模式1) 或 “111” (PWM 模式2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TIM14\_CCMRx 寄存器 OCxPE 位以使能相应的预装载寄存器，最后还要设置 TIM14\_CR1 寄存器中的 ARPE 位 (在向上计数或中心对称模式中) 使能自动重载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIM14\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIM14\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或者低电平有效。TIM14\_CCER 寄存器中的 CcxE 位控制 OCx 输出使能。

在 PWM 模式 (模式1或者模式2)，TIM14\_CNT 和 TIM14\_CCRx 始终在进行比较，以确定是否符合  $TIM14\_CNT \leq TIM14\_CCRx$ 。

定时器能够产生边沿对齐模式的 PWM 信号。

#### PWM 边沿对齐模式

下面是一个 PWM 模式1的例子。当  $TIM14\_CNT < TIMx\_CCRx$  时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIM14\_CCRx 中的比较值大于自动重载值 (TIM14\_ARR)，则 OCxREF 保持为 '1'。如果比较值为 0，则 OCxREF 保持为 '0'。

下图为 TIMx\_ARR=8 时边沿对齐的 PWM 波形实例。

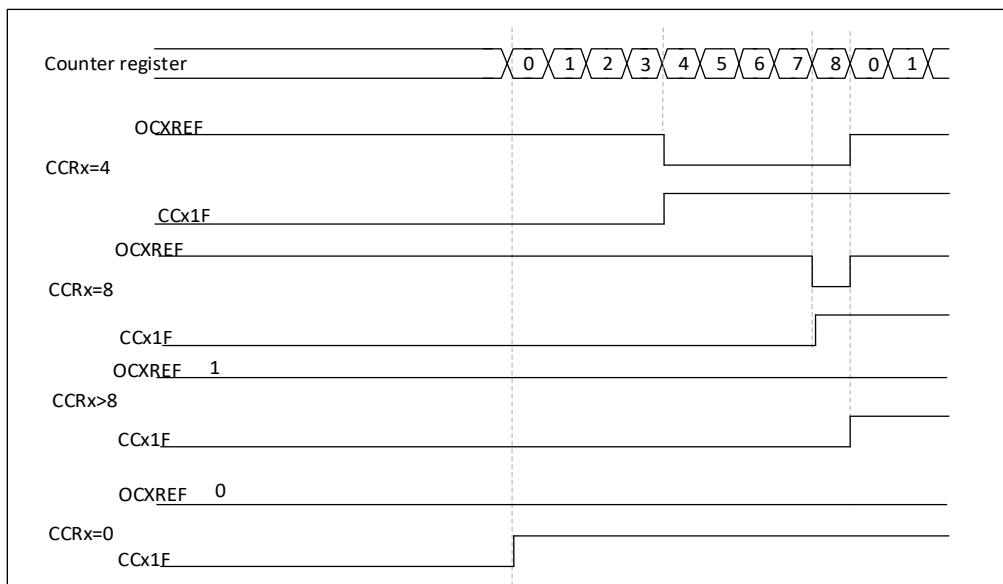


图 24-13 边沿对齐的 PWM 波形 (ARR=8)

### 24.3.8. 单脉冲模式

单脉冲模式 (OPM) 是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以使计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

向上计数方式：计数器  $CNT < CCRx \leq ARR$ （特别地， $0 < CCRx$ ）。

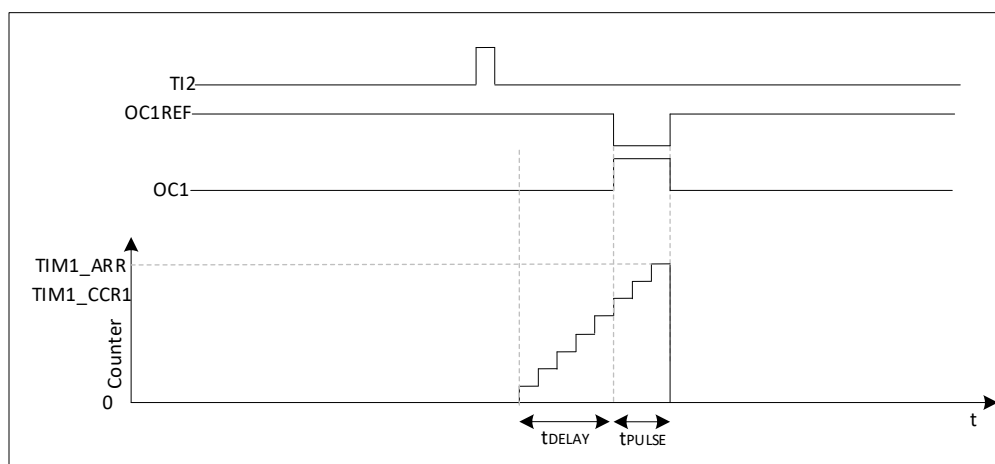


图19-14单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟  $t_{DELAY}$  之后，在 OC1 上产生一个长度为  $t_{PULSE}$  的正脉冲。

假定 TI2FP2 作为触发：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。

- 置 TIMx\_SMCR 寄存器中的 TS=110, TI2FP2作为从模式控制器的触发 (TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110 (触发模式), TI2FP2被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)

- $t_{DELAY}$  由 TIMx\_CCR1寄存器中的值定义。
- $t_{PULSE}$  由自动重载值和比较值之间的差值定义 (TIMx\_ARR - TIMx\_CCR1)。
- 假定当发生比较匹配时要产生从0到1的波形, 当计数器达到预装载值时要产生一个从1到0的波形; 首先要置 TIMx\_CCMR1寄存器的 OC1M=111, 进入 PWM 模式2; 根据需要选择地使能预装载寄存器: 置 TIMx\_CCMR1中的 OC1PE=1和 TIMx\_CR1寄存器中的 ARPE; 然后在 TIMx\_CCR1寄存器中填写比较值, 在 TIMx\_ARR 寄存器中填写自动重载值, 设置 UG 位来产生一个更新事件, 然后等待在 TI2上的一个外部触发事件。本例中, CC1P=0。

在这个例子中, TIMx\_CR1寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲, 所以必须设置 TIMx\_CR1寄存器中的 OPM=1, 在下一个更新事件 (当计数器从自动重载值翻转到0) 时停止计数。当 OPM=0时, 重复模式被选中。

### 24.3.9. 定时器同步

所有 TIMx 定时器在内部相连, 用于定时器同步或链接。当一个定时器处于主模式时, 它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

### 24.3.10. 调试模式

当芯片进入调试模式 (M0+停止), 根据 DBG 模块中 DBG\_TIMx\_STOP 的设置, TIMx 计数器或者继续正常操作, 或者停止。

## 24.4. TIM14寄存器

### 24.4.1. TIM14 控制寄存器 1 (TIM14\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9: 8	CKD[1: 0]	RW	00	时钟分频因子, 这2位定义在定时器时钟 (CK_INT) 频率, 所用的采样时钟之间的分频比例 00: $t_{DTS} = t_{CK\_INT}$

Bit	Name	R/W	Reset Value	Function
				01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: 保留, 不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位 0: TIM14_ARR 寄存器没有缓冲 1: TIM14_ARR 寄存器被装入缓冲器
6: 4	保留	-	-	保留
3	OPM	RW	0	单脉冲模式 (One pulse mode) 0: 在发生更新事件时, 计数器不停止; 1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求: - 计数器溢出/下溢 - 设置 UG 位 1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生: - 计数器溢出/下溢 - 设置 UG 位 被缓存的寄存器被装入它们的预装载值。 1: 禁止 UEV。不产生更新事件, 影子寄存器 (ARR,PSC,CCRx) 保持它们的值。 如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。
0	CEN	RW	0	允许计数器 0: 禁止计数器 1: 开启计数器 注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

### 24.4.2. TIM14 DMA/中断使能寄存器 (TIM14\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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													CC1IE	UIE	
-													RW	RW	

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1	CC1IE	RW	0	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断 1: 允许捕获/比较1中断
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

### 24.4.3. TIM14 状态寄存器 (TIM14\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											IC1IF	Res	Res	Res	IC1IR
-											RC_W0	-	-	-	RC_W0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res						CC1F	UIF	
-						RC_W0	-						RC_W0	RC_W0	

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	保留
20	IC1IF	RC_W0	0	下降沿捕获1标志 仅当相应的通道被配置为输入捕获时且由下降沿触发捕获事件, 该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0'。 0: 无下降沿捕获产生; 1: 发生下降沿捕获事件。

Bit	Name	R/W	Reset Value	Function
19: 17	保留	-	-	保留
16	IC1IR	RC_W0	0	<p>上升沿捕获1标志</p> <p>仅当相应的通道被配置为输入捕获时且由上升沿触发捕获事件，该标记可由硬件置1。它由软件清 '0' 或通过读 TIMx_CCR1清 '0' 。</p> <p>0: 无上升沿捕获产生；</p> <p>1: 发生上升沿捕获事件。</p>
15: 10	保留	-	-	保留
9	CC1OF	RC_W0	0	<p>捕获/比较1 过捕获标记</p> <p>仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。</p> <p>0: 无过捕获产生；</p> <p>1: CC1IF 置1时，计数器的值已经被捕获到 TIM14_CCR1寄存器。</p>
8: 2	保留	-	-	保留
1	CC1IF	RC_W0	0	<p>捕获/比较1 中断标记</p> <p>如果通道 CC1配置为输出模式： 当计数器值与比较值匹配后一个时钟周期时，该位由硬件置1，它由软件清0。</p> <p>0: 无匹配发生；</p> <p>1: TIM14_CNT 的值与 TIM14_CCR1的值匹配。</p> <p>如果通道 CC1配置为输入模式： 当捕获事件发生时该位由硬件置1，它由软件清0或通过读 TIM14_CCR1清0。</p> <p>0: 无输入捕获产生；</p> <p>1: 输入捕获产生并且计数器值已装入 TIM14_CCR1 (在 IC1上检测到与所选极性相同的边沿) 。</p>
0	UIF	RC_W0	0	<p>更新中断标记，当产生更新事件时该位由硬件置1。它由软件清0。</p> <p>0: 无更新事件产生；</p> <p>1: 更新事件等待响应。当寄存器被更新时该位由硬件置1：</p> <ul style="list-style-type: none"> <li>- 若 TIMx_CR1寄存器的 UDIS=0，产生更新事件上溢；</li> <li>- 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1时产生更新事件 (软件对 CNT 重新初始化) ；</li> </ul>



#### 24.4.4. TIM14 事件产生寄存器 (TIM14\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													CC1G	UG	
-													W	W	

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1	CC1G	W	0	产生捕获/比较1事件, 该位由软件置1, 用于产生一个捕获/比较事件, 由硬件自动清0。 0: 无动作; 1: 在通道 CC1上产生一个捕获/比较事件: 若通道 CC1配置为输出: 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断请求。 若通道 CC1配置为输入: 当前的计数器值捕获至 TIM14_CCR1寄存器, 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断请求。 若 CC1IF 已经为1, 则设置 CC1OF=1。
0	UG	W	0	产生更新事件, 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 重新初始化计数器, 并产生一个寄存器的更新事件。注意预分频器的计数器也被清0 (但是预分频系数不变)。

#### 24.4.5. TIM14 捕获/比较模式寄存器 1 (TIM14\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

输出比较模式:

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	OC1M[2: 0]			OC1PE	Res	CC1S[1: 0]	

-	-	RW	RW	RW	RW	-	RW	RW
---	---	----	----	----	----	---	----	----

Bit	Name	R/W	Reset Value	Function
31: 7	保留	-	-	保留
6: 4	OC1M[2: 0]	RW	00	<p>输出比较1模式</p> <p>该位定义了输出参考信号 OC1REF 的动作, 而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效, 而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。</p> <p>000: 冻结。输出比较寄存器 TIMx_CCR1与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用;</p> <p>001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时, 强制 OC1REF 为高。</p> <p>010: 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时, 强制 OC1REF 为低。</p> <p>011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。</p> <p>100: 强制为无效电平。强制 OC1REF 为低。</p> <p>101: 强制为有效电平。强制 OC1REF 为高。</p> <p>110: PWM 模式1 -</p> <p>在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为有效电平, 否则为无效电平; 在向下计数时, 一旦 TIMx_CNT&gt;TIMx_CCR1 时通道 1 为无效电平 (OC1REF=0), 否则为有效电平 (OC1REF=1)。</p> <p>111: PWM 模式2</p> <p>一旦 TIMx_CNT&lt;TIMx_CCR1时, 通道1为无效电平, 否则为有效电平。</p> <p>注: 在 PWM 模式1或 PWM 模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较1预装载使能</p> <p>0: 禁止 TIM14_CCR1寄存器的预装载功能, 可随时写入 TIM14_CCR1寄存器, 且新值马上起作用。</p> <p>1: 开启 TIM14_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIM14_CCR1的预装载值在更新事件到来时被载入当前寄存器中。</p>

Bit	Name	R/W	Reset Value	Function
2	Res	-	-	保留
1: 0	CC1S[1: 0]	RW	00	捕获/比较1 选择。 这2位定义通道的方向（输入/输出），及输入脚的选择： 00: CC1通道被配置为输出； 01: CC1通道被配置为输入，IC1映射在 TI1上； 10: 保留； 11: 保留。 注：CC1S 仅在通道关闭时（TIM14_CCER 寄存器的 CC1E=0）才是可写的。

### 输入捕获模式：

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								IC1F[3: 0]				IC1PSC[1: 0]		CC1S[1: 0]		
-								RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	保留	-	-	保留
7: 4	IC1F[3: 0]	RW	0000	输入捕获1滤波器 这几位定义了 TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： 0000: 无滤波器，以 fDTS 采样 0001: 采样频率 fSAMPLING=fCK_INT, N=2 0010: 采样频率 fSAMPLING=fCK_INT, N=4 0011: 采样频率 fSAMPLING=fCK_INT, N=8 0100: 采样频率 fSAMPLING=fDTS/2, N=6 0101: 采样频率 fSAMPLING=fDTS/2, N=8 0110: 采样频率 fSAMPLING=fDTS/4, N=6 0111: 采样频率 fSAMPLING=fDTS/4, N=8 1000: 采样频率 fSAMPLING=fDTS/8, N=6 1001: 采样频率 fSAMPLING=fDTS/8, N=8 1010: 采样频率 fSAMPLING=fDTS/16, N=5 1011: 采样频率 fSAMPLING=fDTS/16, N=6 1100: 采样频率 fSAMPLING=fDTS/16, N=8 1101: 采样频率 fSAMPLING=fDTS/32, N=5 1110: 采样频率 fSAMPLING=fDTS/32, N=6

				1111: 采样频率 $f_{\text{SAMPLING}}=f_{\text{DTS}}/32$ , $N=8$
3: 2	IC1PSC[1: 0]	RW	00	<p>捕获/比较1预分频器</p> <p>这2位定义了 CC1输入 (IC1) 的预分频系数。一旦 <math>\text{CC1E}=0</math> (TIMx_CCER 寄存器中), 则预分频器复位。</p> <p>00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获;</p> <p>01: 每2个事件触发一次捕获;</p> <p>10: 每4个事件触发一次捕获;</p> <p>11: 每8个事件触发一次捕获。</p>
1: 0	CC1S[1: 0]	RW	00	<p>CC1S[1: 0]: 捕获/比较1选择。</p> <p>这2位定义通道的方向 (输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC1映射在 TI1上;</p> <p>10: 保留</p> <p>11: 保留</p> <p>注: CC1S 仅在通道关闭时 (TIM14_CCER 寄存器的 <math>\text{CC1E}=0</math>) 才是可写的。</p>

#### 24.4.6. TIM14 捕获/比较使能寄存器 (TIM14\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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	CC1NP	Res	CC1P	CC1E
												RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31: 4	保留	-	-	保留
3	CC1NP	RW	0	<p>捕获/比较1互补输出极性</p> <p>CC1通道配置成输出: CC1NP 必须保持0.</p> <p>CC1通道配置成输入: CC1NP 和 CC1P 联合使用来定义 TI1FP1极性 (参考 CC1P 描述)</p>
2	保留	-	-	保留
1	CC1P	RW	0	<p>捕获/比较1输出极性</p> <p>CC1通道配置为输出: 0: OC1高电平有效;</p>

				<p>1: OC1低电平有效。</p> <p>CC1通道配置为输入： CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1和 TI2FP1的极性。</p> <p>00: 不反相/上升沿： TIxFP1上升沿有效（捕获模式下触发）；</p> <p>01: 反相/下降沿： TIxFP1下降沿有效（捕获模式下触发）；</p> <p>10: 保留，不要使用这个配置。</p> <p>11: 不反相/双沿 TIxFP1上升和下降沿都有效（捕获模式下触发）；</p>
0	CC1E	RW	0	<p>捕获/比较1输出使能</p> <p><b>CC1通道配置为输出：</b></p> <p>0: 关闭 - OC1禁止输出</p> <p>1: 开启 - OC1信号输出到对应的输出引脚 <b>CC1</b></p> <p><b>通道配置为输入：</b></p> <p>该位决定了计数器的值是否能捕获入 TIMx_CCR1寄存器。</p> <p>0: 捕获禁止</p> <p>1: 捕获使能</p>

CcxE 位	OCx output State
0	输出禁止 (OCx=0,OCx_EN=0)
1	OCx=OCxREF+Polarity,OCx_EN=1

#### 24.4.7. TIM14 计数器 (TIM14\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CNT[15: 0]	RW	0	计数器的值

### 24.4.8. TIM14 预分频器 (TIM14\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PSC[15: 0]	RW	0	<p>预分频器的值</p> <p>计数器的时钟频率 (CK_CNT) 等于 fCK_PSC/ ( PSC[15: 0]+1) 。</p> <p>PSC 包含了当更新事件产生时装入当前预分频器寄存器的值; 更新事件包括计数器</p> <p>被 TIM_EGR 的 UG 位清0或被工作在复位模式的从控制器清0。</p>

### 24.4.9. TIM14 自动重载寄存器 (TIM14\_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ARR[15: 0]	RW	FFFF	<p>自动重载的值</p> <p>ARR 包含了将要装载入实际的自动重载寄存器的值。</p> <p>详细参考24.3.1: 时基单元有关 ARR 的更新和动作。</p> <p>当自动重载的值为空时, 计数器不工作。</p>

## 24.4.10. TIM14 捕获/比较寄存器 1 (TIM14\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
CCR1[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CCR1[15: 0]	RW	0	<p>捕获/比较1的值</p> <p>若 CC1通道配置为输出： CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC1PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较1寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC1端口上输出信号。</p> <p>若 CC1通道配置为输入： CCR1包含了由上一次输入捕获1事件（IC1）传输的计数器值。</p>

### 24.4.11. TIM14 选项寄存器 (TIMx\_OR)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														TI1_RMP	
-														RW	

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1: 0	TI1_RMP	RW	0	定时器输入1 重映射 通过软件置位和清零。 00: TIM14通道1连接到 GPIO,具体参考数据手册的复用功能。 01: TIM14通道1 连接到 RTCCLK. 10: TIM14通道1 连接到 HSE/32时钟 11: TIM14通道1 连接到 MCU 时钟输出 (MCO) .这个配置是通过 RCC_CFG 寄存器的 MCO[2: 0]的设置来决定的。



## 25. 通用定时器 (TIM15/16/17)

### 25.1. 简介

通用定时器 (TIM15\_16\_17) 由16位被可编程预分频器驱动的自动重载计数器组成, 可应用于各种场景, 包括: 输入信号 (输入捕获) 的脉冲长度测量, 或者产生输出波形 (输出比较、输出 PWM、带死区插入的互补 PWM) 。

脉冲长度和波形周期可以使用定时器分频器和 RCC 时钟控制分频器, 从微秒到毫秒的调制。通用定时器 (TIM15\_16\_17) 和通用 (TIMx) timer 是完全独立的, 不共享任何资源。TIM15\_16\_17和其它 TIMER 可以同步起来。

### 25.2. TIM15主要特性

- 16bit 向上的自动重载计数器
- 16bit 可编程预分频器, 允许对计数器的时钟频率进行 1 到 65536 的分频
- 多达 2 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 产生 (边沿对齐模式)
  - 单脉冲模式输出
- 死区时间可编程的互补输出 (只有通道 1)
- 使用外部信号控制定时器和定时器互连的同步电路
- 重复计数器, 在计数指定周期数后, 才更新定时器的寄存器
- 刹车输入可以将定时器的输出信号置为复位状态或者已知状态
- 中断/DMA 产生在以下事件
  - 更新: 计数器向上溢出, 计数器初始化 (通过软件或者内外部触发)
  - 触发事件
  - 输入捕获
  - 输出比较
  - 刹车输入

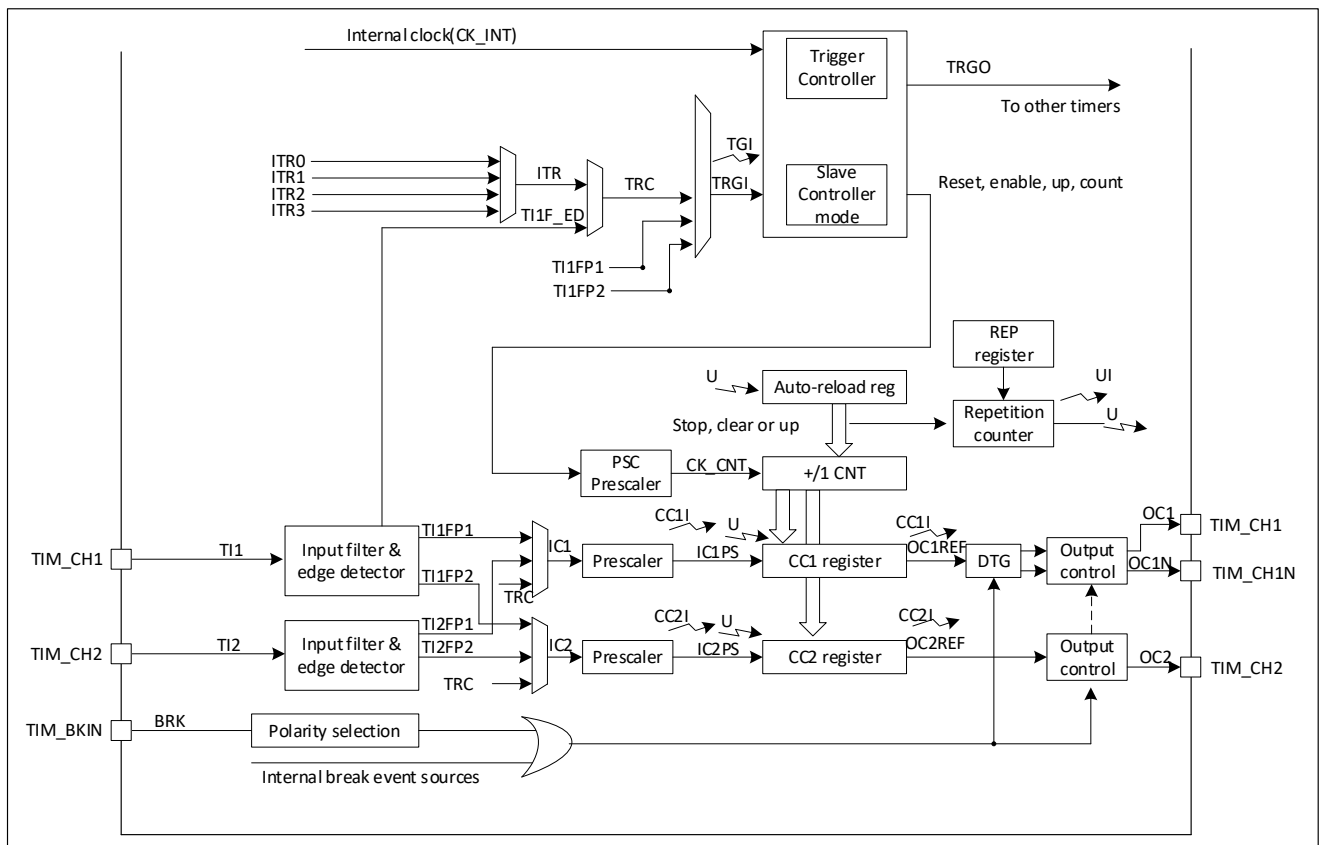


图 25-1 通用控制定时器 (TMI15) 架构框图

### 25.3. TIM16\_17 主要特性

- 16bit 向上的自动重装计数器
- 16bit 可编程预分频器，允许对计数器的时钟频率进行 1 到 65536 的分频
- 1 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 产生 (边缘对齐模式)
  - 单脉冲模式输出
- 死区时间可编程的互补输出
- 重复计数器，在计数指定周期数后，才更新定时器的寄存器
- 刹车输入可以将定时器的输出信号置为复位状态或者已知状态
- 中断/DMA 产生在以下事件
  - 更新：计数器向上溢出
  - 输入捕获
  - 输出比较
  - 刹车输入

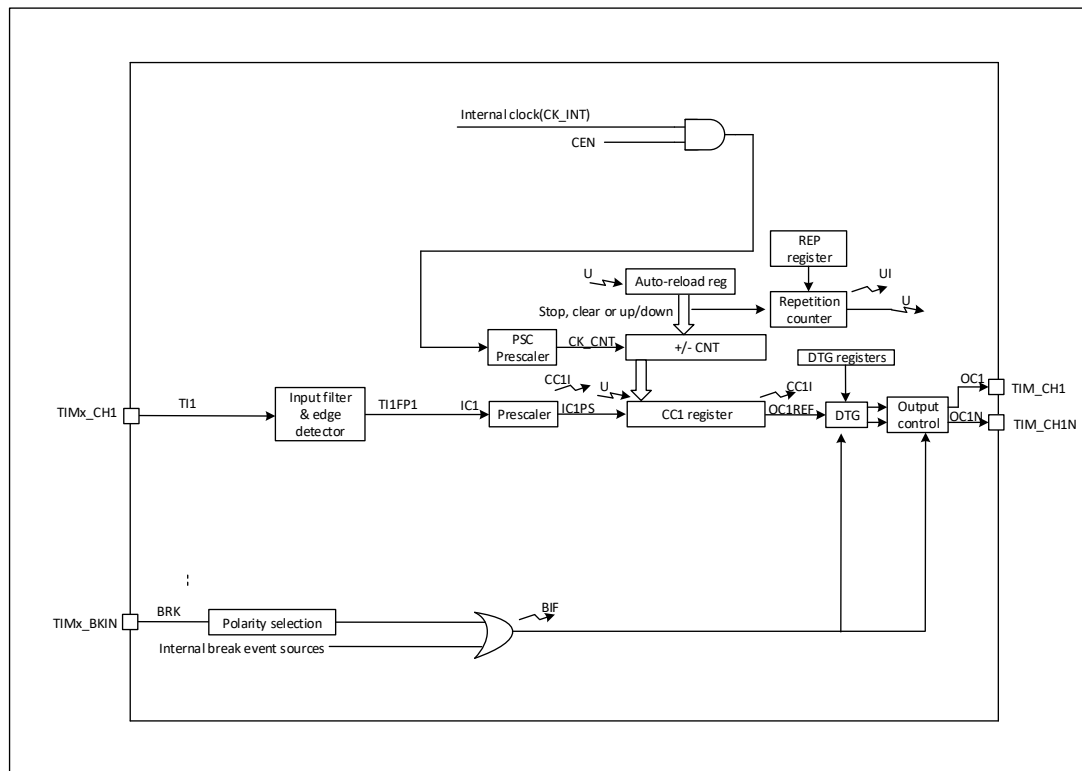


图 25-2 通用控制定时器 (TIM16\_17) 架构框图

## 25.4. TIM15\_16\_17功能描述

### 25.4.1. 时基单元

可编程通用控制定时器的主要部分是一个16位计数器和与其相关的自动重载寄存器。这个计数器可以向上计数。此计数器时钟由预分频器分频得到。

计数器、自动重载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包括：

- 计数器寄存器 (TIMx\_CNT)
- 预分频寄存器 (TIMx\_PSC)
- 自动重载寄存器 (TIMx\_ARR)
- 重复计数寄存器 (TIMx\_RCR)

自动重载寄存器是预先装载的，写或者读自动重载寄存器将访问预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动重载预装载使能位 (ARPE) 的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到上溢并当 TIMx\_CR1 寄存器中的 UDIS 位等于0时，产生更新事件。更新事件也可以由软件产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIMx\_CR1 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

## 预分频器描述

预分频器可以将计数器的时钟按1到65536之间的任意值分频。它是一个基于（在TIMx\_PSC寄存器中的）16位寄存器控制的16位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下次更新事件到来时被采用。

图 xx 和图 xx 给出了在预分频器运行时，更改计数器参数的例子。

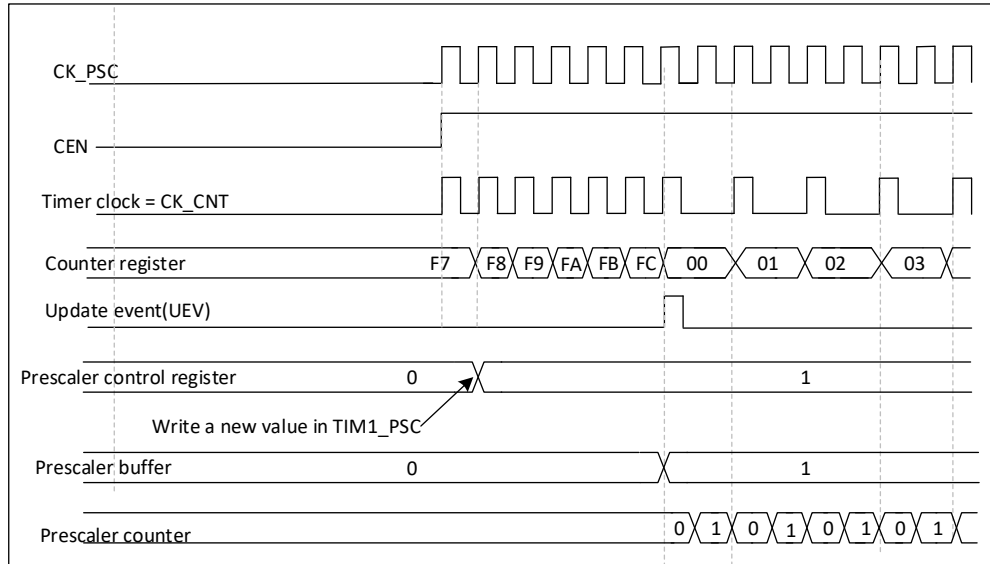


图 25-3 当预分频器的参数从 1 变到 2 时，计数器的时序图

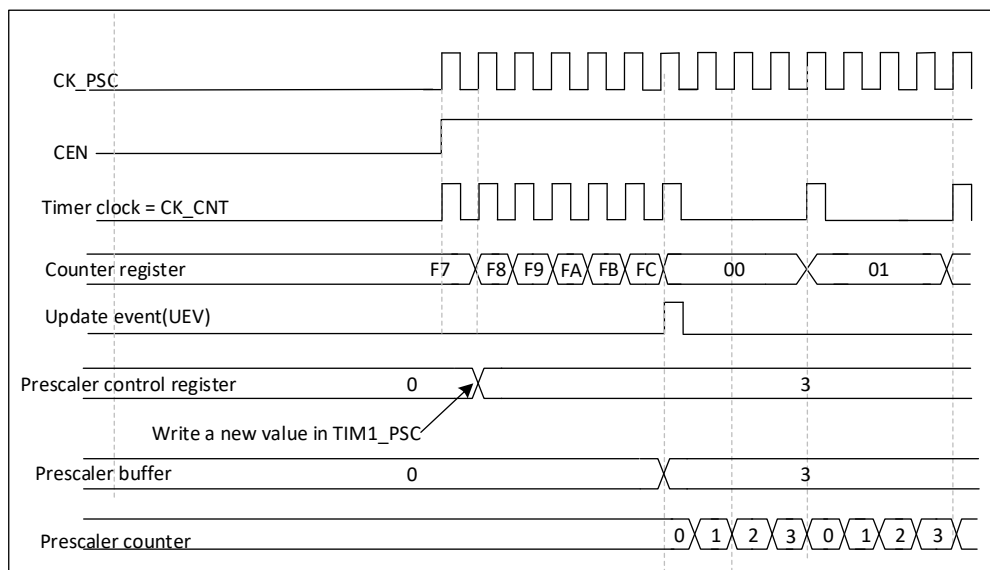


图 25-4 当预分频器的参数从 1 变到 4 时，计数器的时序图

## 25.4.2. 计数器模式

### 向上计数模式

向上计数模式，是从0到自动重载值的计数器，然后又从0重新开始计数，并产生一个计数的溢出事件。

如果重复计数器被使用，则在上溢次数达到所配置的重复计数寄存器的值加一（即TIMx\_RCR+1）后，产生更新事件。否则，在每个计数溢出时，产生更新事件。

在TIMx\_EGR寄存器中（通过软件方式或者使用从模式控制器）设置UG位也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样也可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清'0'，同时预分频器的计数也被清0（但预分频器的数值不变）。此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时（依据 URS 位）设置更新标志位（TIMx\_SR 寄存器中的 UIF 位）。

- 重复计数器被重新加载为 TIMx\_RCR 寄存器的内容。
- 自动重载影子寄存器被重新置入预装载寄存器的值（TIMx\_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIMx\_PSC 寄存器的内容）。

下图给出一些例子，当 TIMx\_ARR=0x36时计数器在不同时钟频率下的动作。

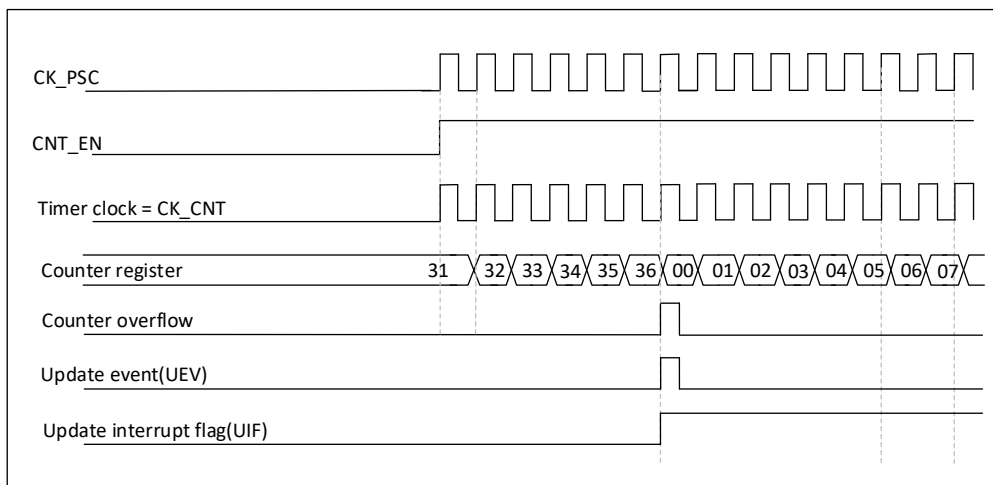


图 25-5 计数器时序图，内部时钟分频因子为 1

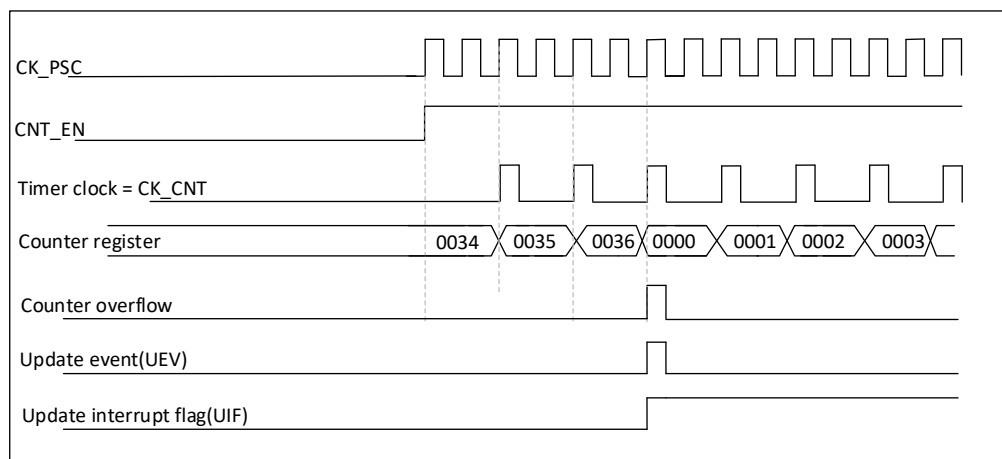


图 25-6 计数器时序图，内部时钟分频因子为 2

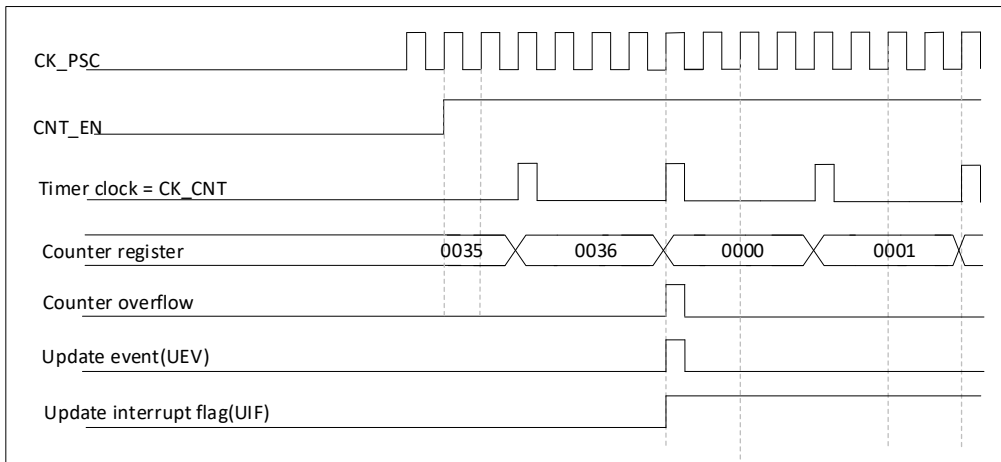


图 25-7 计数器时序图，内部时钟分频因子为 4

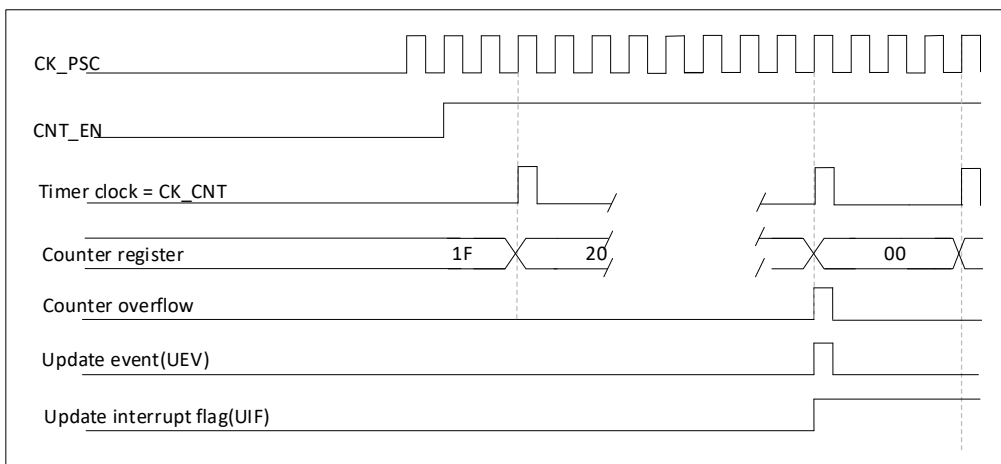


图 25-8 计数器时序图，内部时钟分频因子为 N

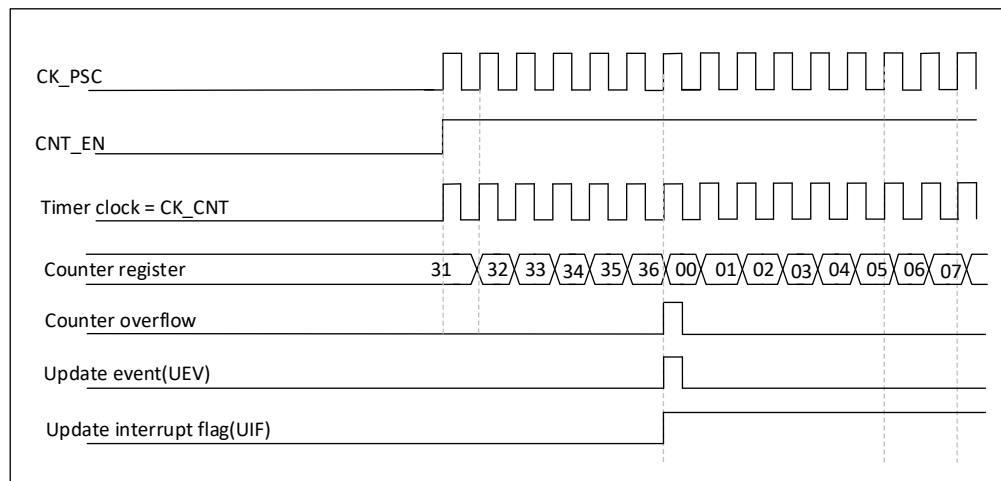


图 25-9 计数器时序图，当 ARPE=0 时的更新事件 (TIMx\_ARR 没有预装入)

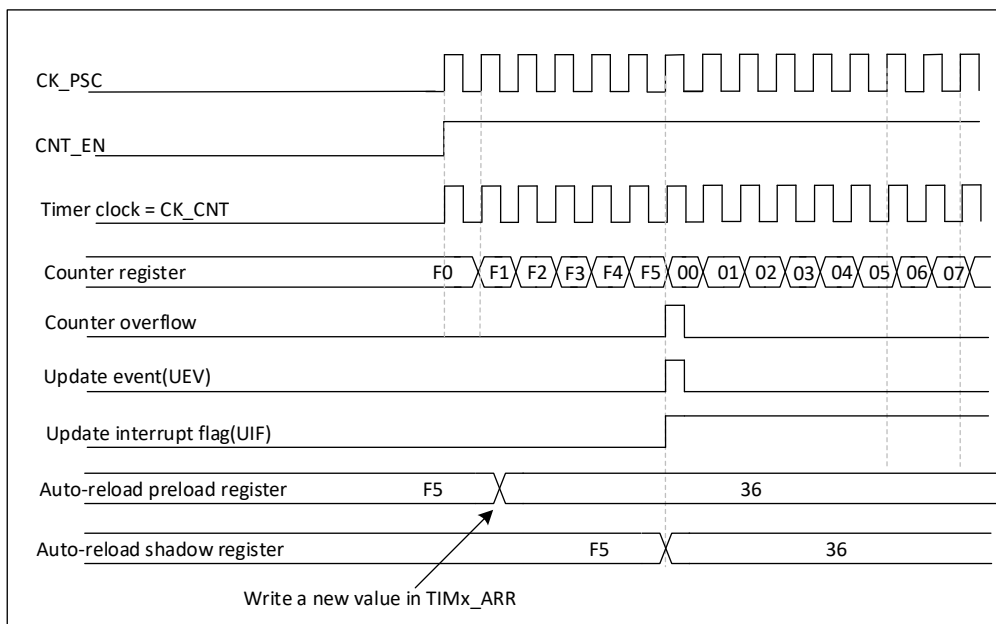


图 25-10 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx\_ARR）

### 25.4.3. 重复计数器

时基单元描述了关于计数器向上溢出的更新事件如何产生的。它实际上仅当重复计数器计数到零才产生。这也是当产生 PWM 信号时很有用的。

这意味着在每 N+1 次计数上溢时，数据被从预装载寄存器传送到影子寄存器（TIMx\_ARR 自动重载入寄存器，TIMx\_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx\_CCRx），N 是 TIMx\_RCR 重复计数器寄存器中的值。

重复计数器在下述条件成立时递减：

- 向上计数模式下每次计数上溢时

重复计数器是自动加载的，重复速率是由 TIMx\_RCR 寄存器的值定义。当更新事件由软件产生（通过设置 TIMx\_EGR 中的 UG 位）或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 TIMx\_RCR 寄存器中的内容被重载如到重复计数器。

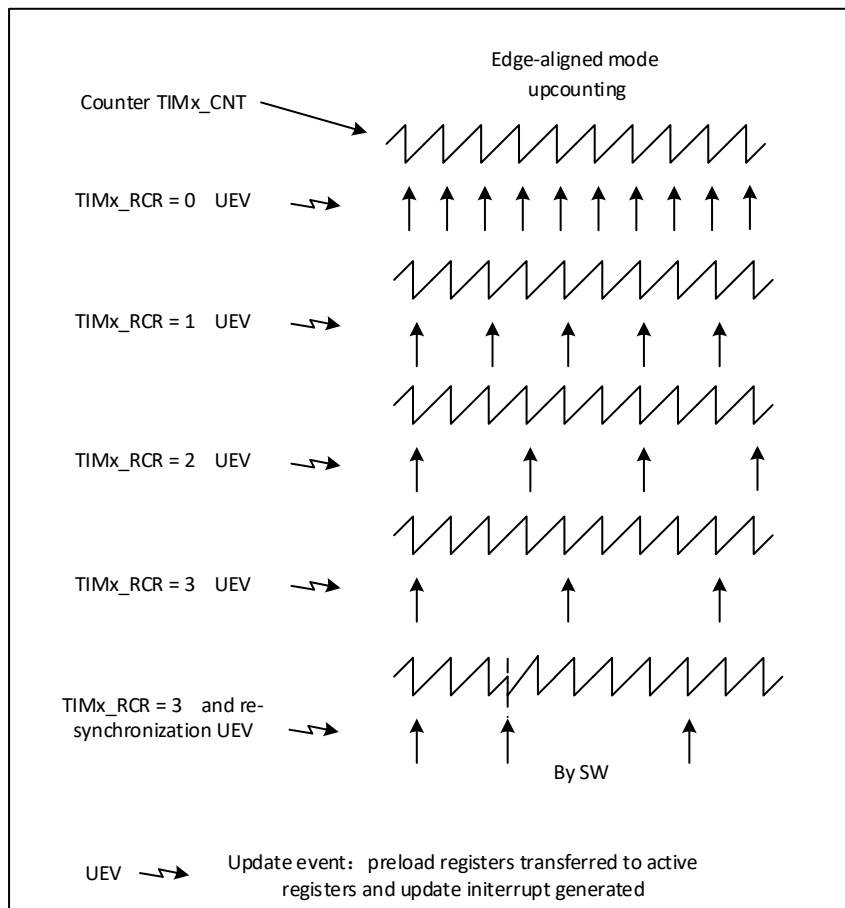


图 25-11 不同模式下更新速率的例子，及 TIMx\_RCR 的寄存器设置

#### 25.4.4. 时钟源

计数器的时钟可以由以下时钟源提供：

- 内部时钟 (CK\_INT)
- 外部时钟模式 1：外部输入引脚 (仅 TMI15)
- 内部触发输入 (ITRx)：使用一个定时器作为另一个定时器的预分频器。例如，可以配置一个定时器 Timer1 作为另一个定时器 Timer2 的预分频器。(仅 TMI15)

##### 内部时钟源 (CK\_INT)

如果从模式控制器被禁止，则 CEN、DIR (TIMx\_CR1寄存器) 和 UG 位 (TIMx\_EGR 寄存器) 是事实上的控制位，并且只能被软件修改。只要 CEN 位被写成1，预分频器的时钟就由内部时钟 CK\_INT 提供。



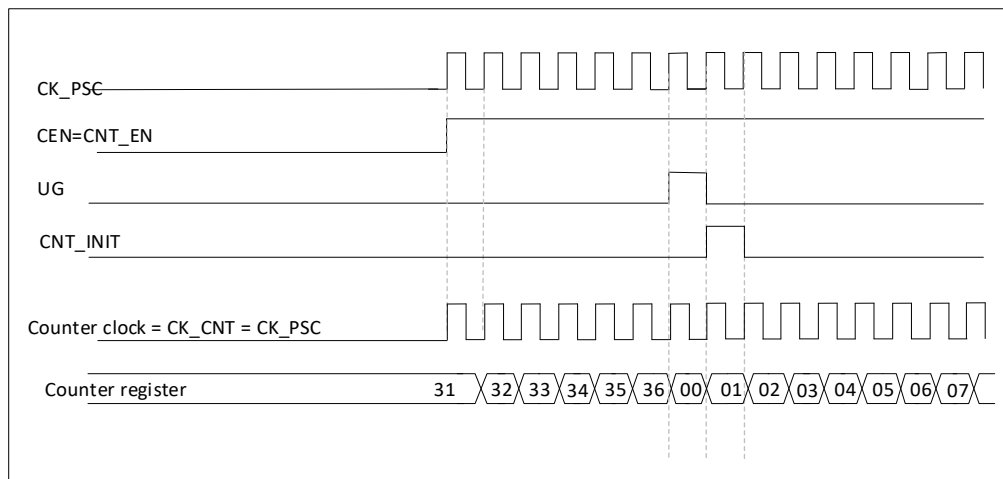


图 25-12 一般模式下的控制电路，内部时钟分频因子为 1

### 外部时钟源模式1 (仅 TIM15)

当 TIMx\_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

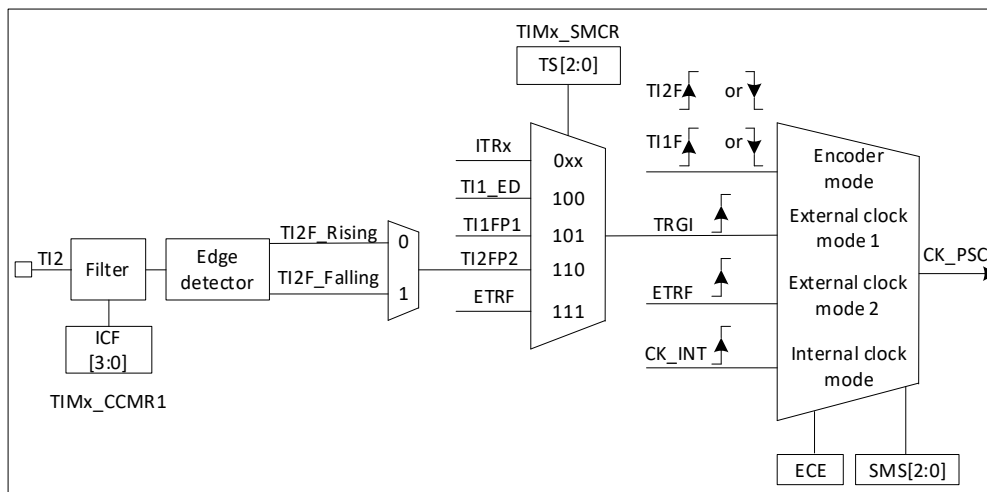


图 25-13 TI2 外部时钟连接例子

例如，要配置计数器在 T12输入端的上升沿向上计数，使用下列步骤：

- 1.配置 TIMx\_CCMR1寄存器 CC2S=01，使得通道2检测 TI2输入端的上升沿；
- 2.配置 TIMx\_CCMR1寄存器的 IC2F[3: 0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2F=0000）；

- 3.配置 TIMx\_CCER 寄存器的 CC2P=0，选定上升沿极性；
- 4.配置 TIMx\_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式1；
- 5.配置 TIMx\_SMCR 寄存器中的 TS=110，选定 TI2作为触发输入源；
- 6.设置 TIMx\_CR1寄存器的 CEN=1，启动计数器。

注：捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2的上升沿和计数器实际时钟之间的延时，取决于在 TI2输入端的重新同步电路。

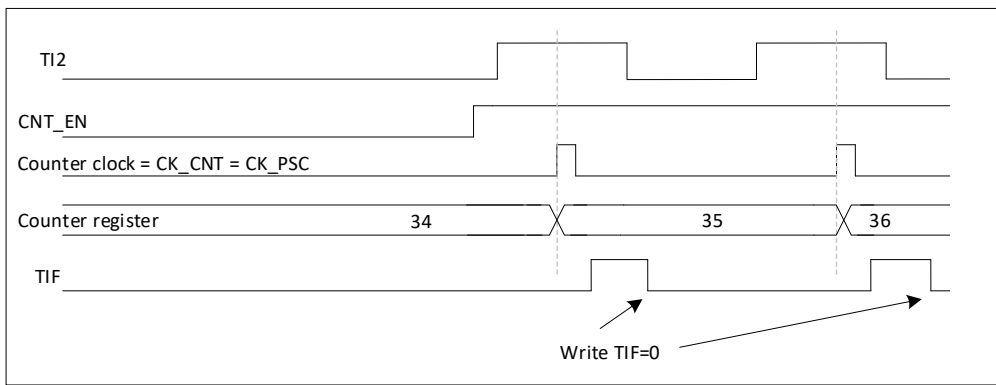


图 25-14 外部时钟模式1下的控制电路

### 25.4.5. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

输入部分对相应的  $TIx$  输入信号采样，并产生一个滤波后的信号  $TIxF$ 。然后，一个带极性选择的边缘监测器产生一个信号 ( $TIxFPx$ )，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 ( $ICxPS$ )。

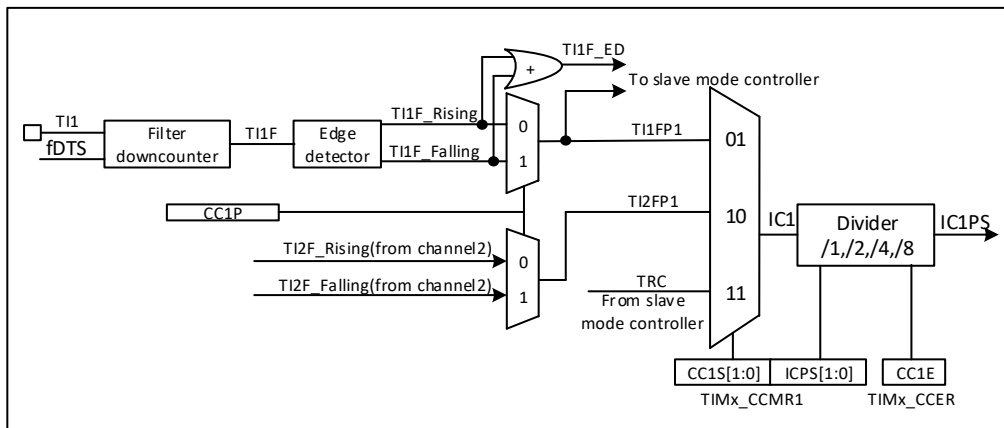


图 25-15 捕获/比较通道 (如: 通道 1 输入部分)

输出部分产生一个中间波形  $OCxREF$  (高有效) 作为基准，链的末端决定最终输出信号的极性。

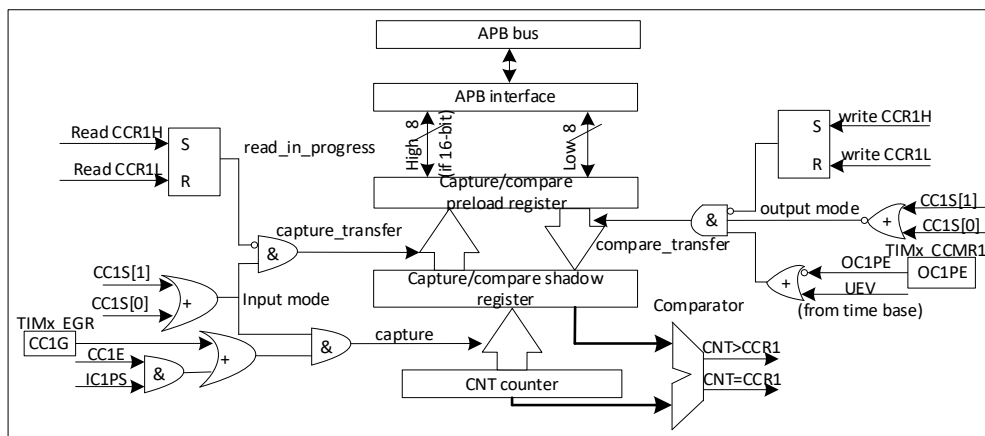


图 25-16 捕获/比较通道 1 的主电路

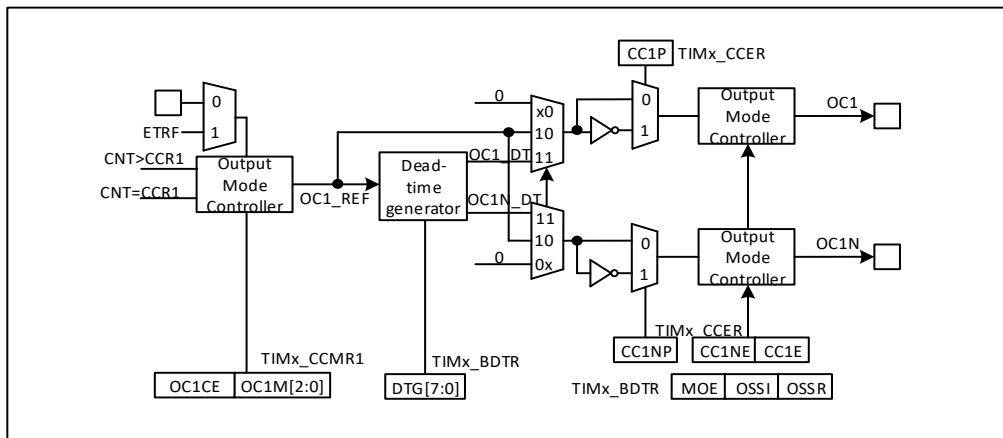


图 25-17 捕获/比较通道的输出部分（通道 1 至 3）

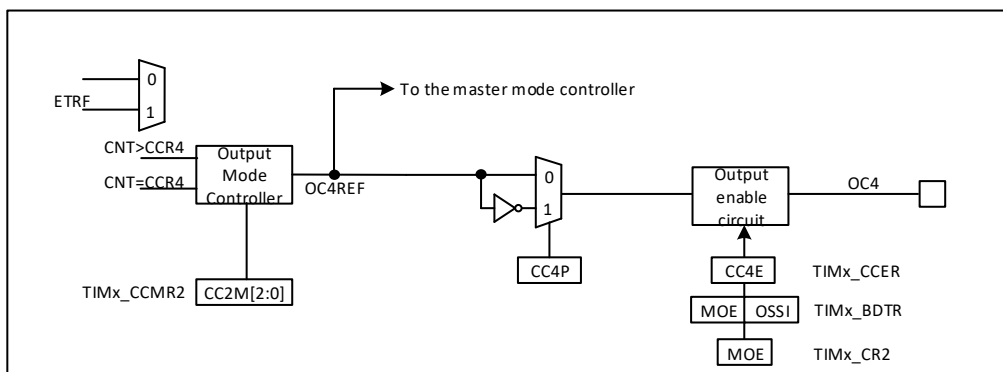


图 25-18 捕获/比较通道的输出部分（通道 4）

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

### 25.4.6. 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器中。当发生捕获事件时，相应的 CCxIF 标志（TIMx\_SR 寄存器）被置 1，如果中断或者 DMA 操作被打开，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，则重复捕获标志 CCxOF（TIMx\_SR 寄存器）被置 1。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx\_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx\_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx\_CCMR1 必须连接到 TI1 输入，所以写入 TIMx\_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为 '00'，通道被配置为输入，并且 TIMx\_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 TIx 时，输入滤波器控制位是 TIMx\_CCMRx 寄存器中的 ICxF 位）。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以（以 fDTS 频率）连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx\_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx\_CCER 寄存器中写入 CC1P=0（上升沿）
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx\_CCMR1 寄存器的 IC1PS=00）。

- 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置 TIMx\_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx\_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少2个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

#### 25.4.7. PWM 输入模式 (仅 TIM15)

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射到同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，当需要测量输入到 TI1 上的 PWM 信号的长度（TIMx\_CCR1 寄存器）和占空比（TIMx\_CCR2 寄存器）时，具体步骤如下（取决于 CK\_INT 的频率和预分频器的值）

- 选择 TIMx\_CCR1 的有效输入：置 TIMx\_CCMR1 寄存器的 CC1S=01（选中 TI1）。
- 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx\_CCR1 中和清除计数器）：置 CC1P=0（上升沿有效）。
- 选择 TIMx\_CCR2 的有效输入：置 TIMx\_CCMR1 寄存器的 CC2S=10（选中 TI1）。
- 选择 TI1FP2 的有效极性（捕获数据到 TIMx\_CCR2）：置 CC2P=1（下降沿有效）。
- 选择有效的触发输入信号：置 TIMx\_SMCR 寄存器中的 TS=101（选择 TI1FP1）。
- 配置从模式控制器为复位模式：置 TIMx\_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx\_CCER 寄存器中 CC1E=1 且 CC2E=1。

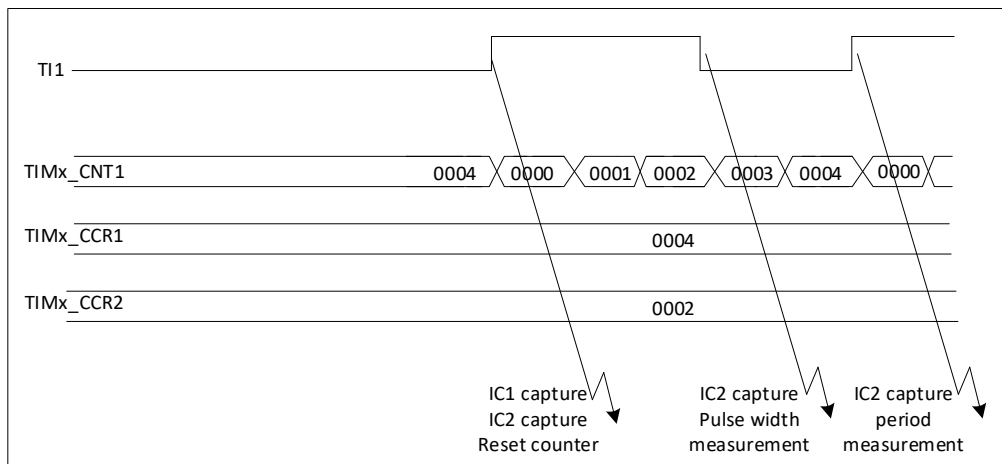


图 25-19 PWM 输入模式时序

### 25.4.8. 强置输出模式

在输出模式 (TIMx\_CCMRx 寄存器中 CCxS=00) 下, 输出比较信号 (OCxREF 和相应的 OCx/OCxN) 能够直接由软件强置为有效或无效状态, 而不依赖于输出比较寄存器和计数器间的比较结果。置 TIMx\_CCMRx 寄存器中相应的 OCxM=101, 即可强置输出比较信号 (OCxREF/OCx) 为有效状态。这样 OCxREF 被强置为高电平 (OCxREF 始终为高电平有效), 同时 OCx 得到 CCxP 极性相反的信号。

例如: CCxP=0 (OCx 高电平有效), 则 OCx 被强置为高电平。置 TIMx\_CCMRx 寄存器中的 OCxM=100, 可强置 OCxREF 信号为低。

该模式下, 在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行, 相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下文的输出比较模式一节中介绍。

### 25.4.9. 输出比较模式

此项功能是用来控制一个输出波形, 或者指示一段给定的时间已经到时。当计数器与捕获/比较寄存器的内容相同时, 输出比较功能做如下操作:

- 将输出比较模式 (TIMx\_CCMRx 寄存器中的 OCxM 位) 和输出极性 (TIMx\_CCER 寄存器中的 CCxP 位) 定义的值输出到对应的引脚上。在比较匹配时, 输出引脚可以保持它的电平 (OCxM=000)、被设置成有效电平 (OCxM=001)、被设置成无效电平 (OCxM=010) 或进行翻转 (OCxM=011)。
- 设置中断状态寄存器中的标志位 (TIMx\_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx\_DIER 寄存器中的 CCxIE 位), 则产生一个中断。
- 若设置了相应的使能位 (TIMx\_DIER 寄存器中的 CCxDE 位, TIMx\_CR2 寄存器中的 CCDS 位选择 DMA 请求功能), 则产生一个 DMA 请求。

TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。在输出比较模式下, 更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式 (在单脉冲模式下) 也能用来输出一个单脉冲。

输出比较模式的配置步骤:

1. 选择计数器时钟 (内部, 外部, 预分频器)。
2. 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
3. 如果要产生一个中断请求, 设置 CCxIE 位。

## 4. 选择输出模式，例如：

- 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
- 置 OCxPE = 0禁用预装载寄存器
- 置 CCxP = 0选择极性为高电平有效
- 置 CcxE = 1使能输出

## 5. 设置 TIMx\_CR1寄存器的 CEN 位启动计数器

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器 (OCxPE='0'，否则 TIMx\_CCRx 的影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

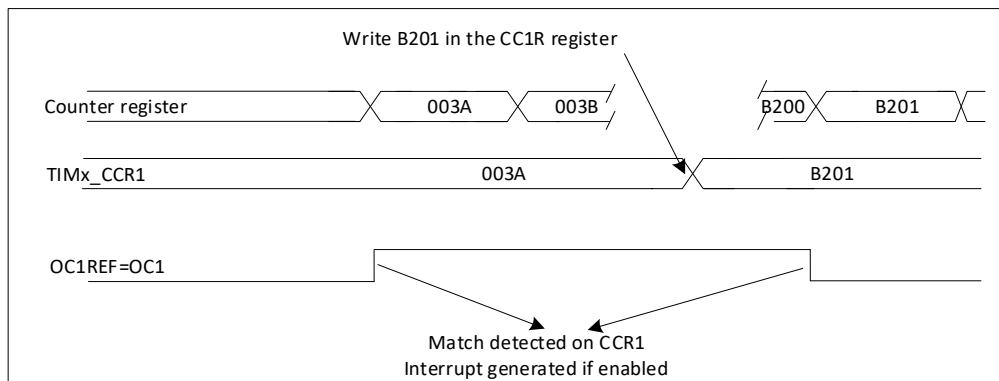


图 25-20 输出比较模式，翻转 OC1

## 25.4.10. PWM 模式

脉冲宽度调制模式可以允许产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入 "110" (PWM 模式1) 或 "111" (PWM 模式2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx\_CR1寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过 (TIMx\_CCER 和 TIMx\_BDTR 寄存器中) CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx\_CCER 寄存器的描述。

在 PWM 模式 (模式1或模式2) 下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，(依据计数器的计数方向) 以确定是否符合  $TIMx\_CCRx \leq TIMx\_CNT$  或者  $TIMx\_CNT \leq TIMx\_CCRx$ 。

根据 TIMx\_CR1寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

### PWM 边沿对齐模式

#### ■ 向上计数配置

当 TIMx\_CR1寄存器中的 DIR 位为低的时候执行向上计数。参看下面是一个 PWM 模式1的例子。当  $TIMx\_CNT < TIMx\_CCRx$  时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自

动重载值 (TIMx\_ARR)，则 OCxREF 保持为'1'。如果比较值为0，则 OCxREF 保持为'0'。下图为 TIMx\_ARR=8时边沿对齐的 PWM 波形实例。

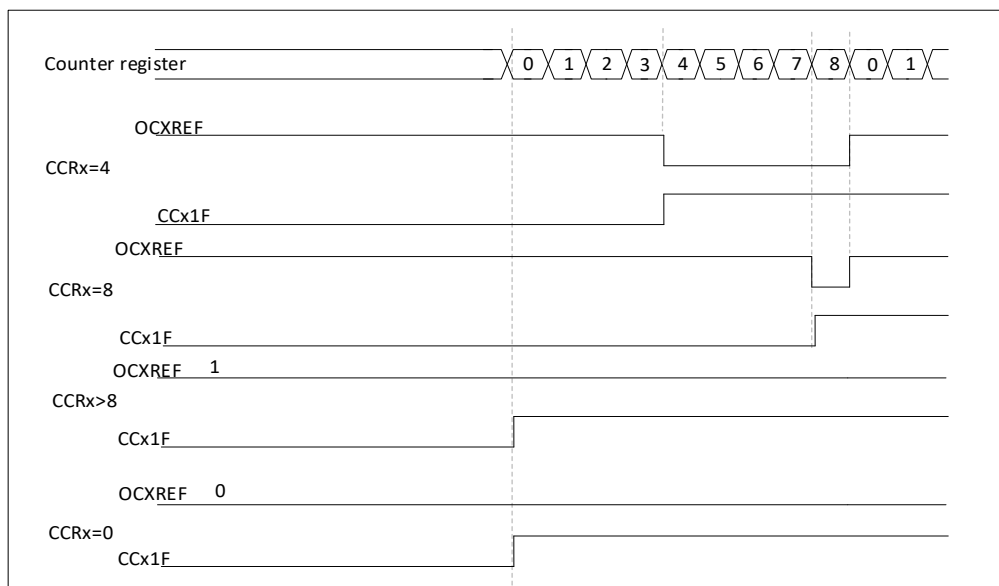


图 25-21 边沿对齐方式 PWM 输出，向上 (ARR=8)

#### 25.4.11. 互补输出和死区插入

通用控制定时器 (TIM15\_16\_17) 能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性（电平转换的延时、电源开关的延时等）来调整死区时间。

配置 TIMx\_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性（主输出 OCx 或互补输出 OCxN）。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TIMx\_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx\_BDTR 和 TIMx\_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见 TIMx\_CCER 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLE 状态时 (MOE 下降到0) 死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 8 位的死区发生器 DTG[7:0]。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度 (OCx 或者 OCxN)，则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。（假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

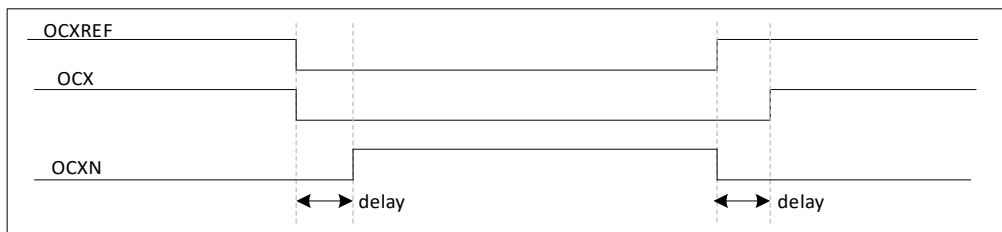


图 25-22 带死区插入的互补输出

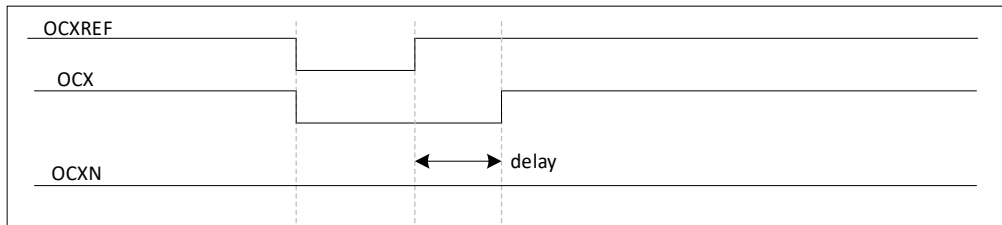


图 25-23 死区波形延迟大于负脉冲

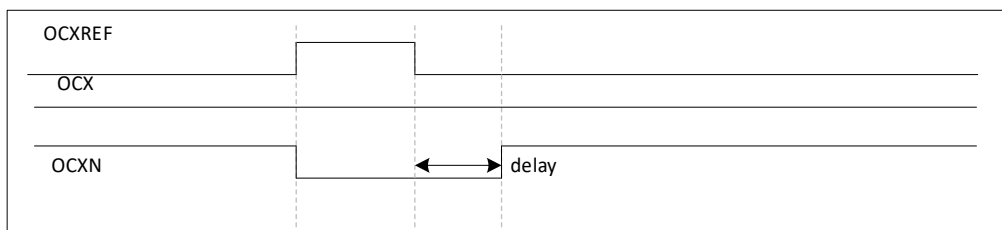


图 25-24 死区波形延迟大于正脉冲

每一个通道的死区延时都是相同的，是由 TIMx\_BDTR 寄存器中的 DTG 位编程配置。

### 重定向 OCxREF 到 OCx 或 OCxN

在输出模式下（强置、输出比较或 PWM），通过配置 TIMx\_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形（例如 PWM 或者静态有效电平）。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注：当只使能 OCxN (CCxE=0, CCxNE=1) 时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时 (CCxE=CCxNE=1)，当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

### 25.4.12. 使用刹车功能

当使用刹车功能时，依据额外的控制位（TIMx\_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx\_CR2 寄存器中的 OISx 和 OISxN 位），输出使能信号和无效电平信号都会被修改。无论什么情况下，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。

刹车源既可以是刹车输入引脚，或者以下内部源：

- 内核 LOCKUP 输出
- PVD 输出
- 由 CSS 监测产生的时钟 failure 事件
- 来自比较器的输出



系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx\_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有1个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（在 TIMx\_BDTR 寄存器中）之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时（空指令）才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时（在刹车输入端出现选定的电平），有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态（由 OSSI 位选择）。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx\_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些（大约 2 个 ck\_tim 的时钟周期）。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 如果设置了 TIMx\_DIER 寄存器中的 BIE 位，当刹车状态标志（TIMx\_SR 寄存器中的 BIF 位）为'1'时，则产生一个中断。如果设置了 TIMx\_DIER 寄存器中的 BDE 位，则产生一个 DMA 请求。
- 如果设置了 TIMx\_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能同时（自动地或者通过软件）设置 MOE。同时，状态标志 BIF 不能被清除。

刹车可以由 BRK 输入产生，它的有效极性是可编程的，且由 TIMx\_BDTR 寄存器中的 BKE 位开启。刹车也可以通过软件设置 TIMx\_EGR 寄存器中的 BG 位来产生。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数（死区长度，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性）。用户可以通过 TIMx\_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

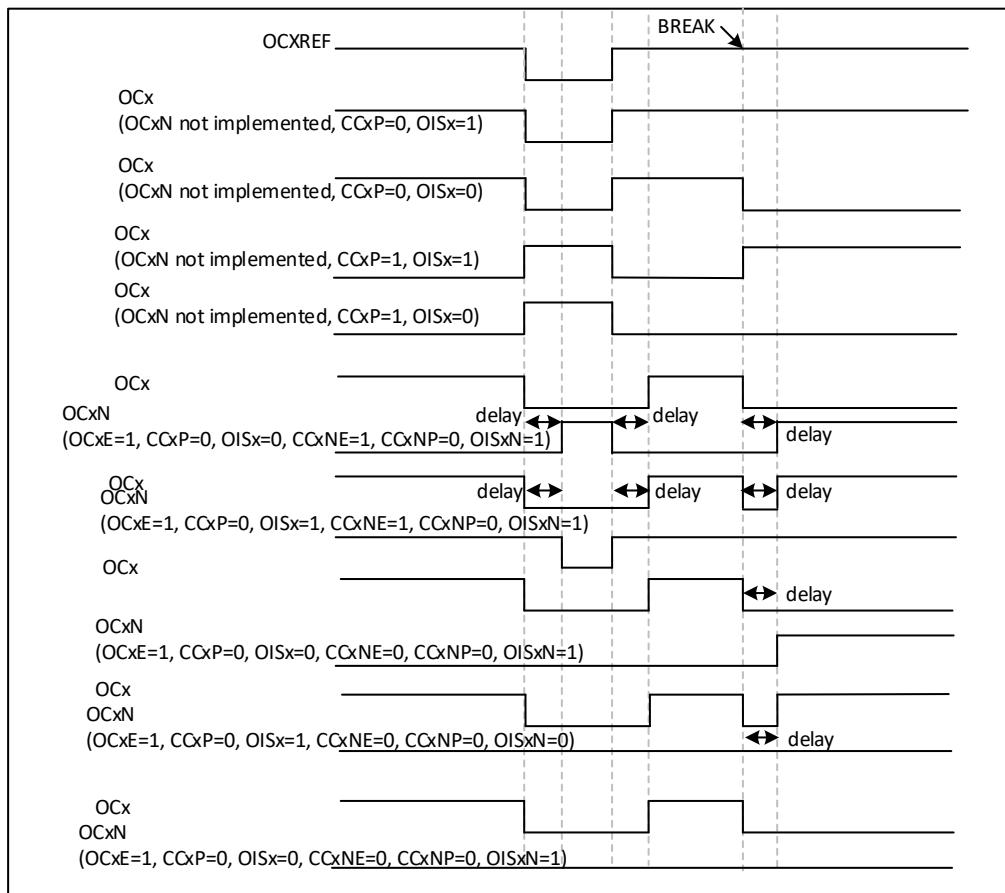


图 25-25 响应刹车的输出

### 25.4.13. 单脉冲模式

单脉冲模式 (OPM) 是之前所述众多模式中的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽可被程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器的 OPM 位将选择单脉冲模式，这样可以使计数器自动的在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$ （特别地， $0 < CCRx$ ）

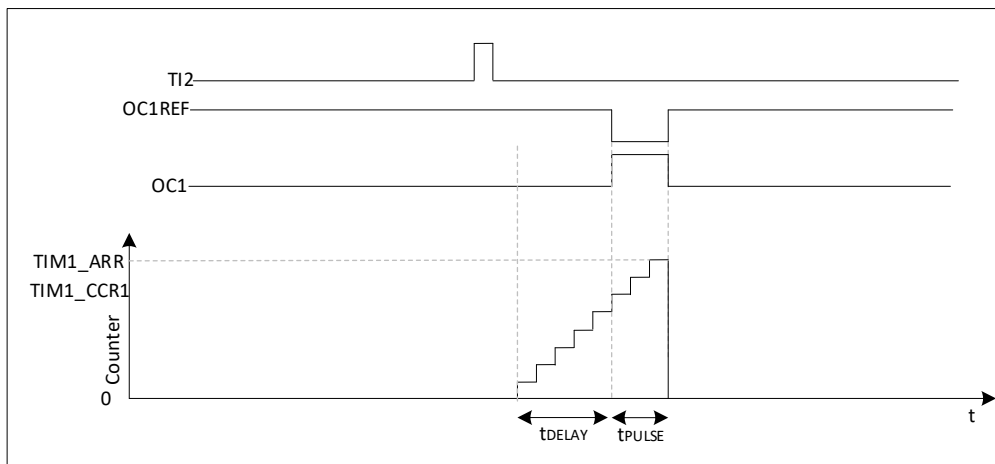


图 25-26 单脉冲模式的例子

例如，当需要在从 TI2 输入脚上检测到一个上升沿开始，延迟  $t_{\text{DELAY}}$  之后，在 OC1 上产生一个长度为  $t_{\text{PULSE}}$  的正脉冲。

使用 TI2FP2 作为触发 1：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx\_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发 (TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110 (触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)

- $t_{\text{DELAY}}$  由 TIMx\_CCR1 寄存器中的值定义。
- $t_{\text{PULSE}}$  由自动重载值和比较值之间的差值定义 (TIMx\_ARR - TIMx\_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE=1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动重载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM=1，在下一个更新事件 (当计数器从自动重载值翻转到 0) 时停止计数。

#### 特殊情况：OCx 快速使能：

在单脉冲模式下，在 Tix 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时  $t_{\text{DELAY}}$ 。

如果要以最小延时输出波形，可以设置 TIMx\_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF (和 OCx) 直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

## 25.5. TIMx 定时器和外部触发的同步 (仅 TIM15)

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

### 25.5.1. 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器 (TIMx\_ARR, TIMx\_CCRx) 都被更新了。

在以下的例子中，TI1输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测 TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0以确定极性（只检测上升沿）。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1出现一个上升沿；此时，计数器被清零然后从0重新开始计数。同时，触发标志 (TIMx\_SR 寄存器中的 TIF 位) 被设置，根据 TIMx\_DIER 寄存器中 TIE (中断使能) 位和 TDE (DMA 使能) 位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重载寄存器 TIMx\_ARR=0x36时的动作。在 TI1上升沿和计数器的实际复位之间的延时取决于 TI1输入端的重同步电路。

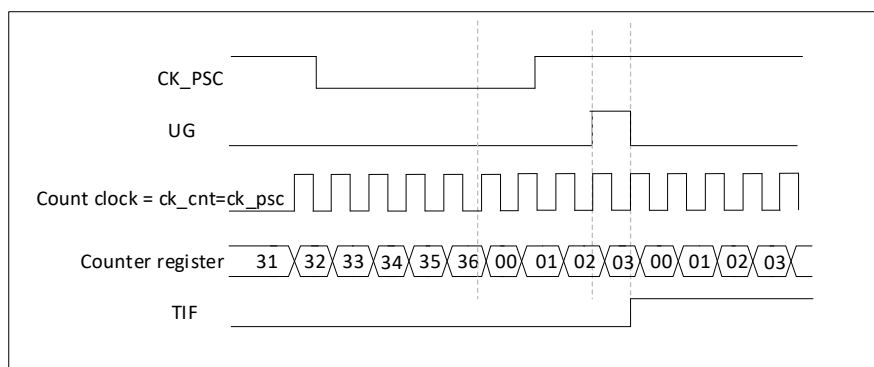


图 25-27 复位模式下的控制电路

### 25.5.2. 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1为低时向上计数：

- 配置通道1以检测 TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标志。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

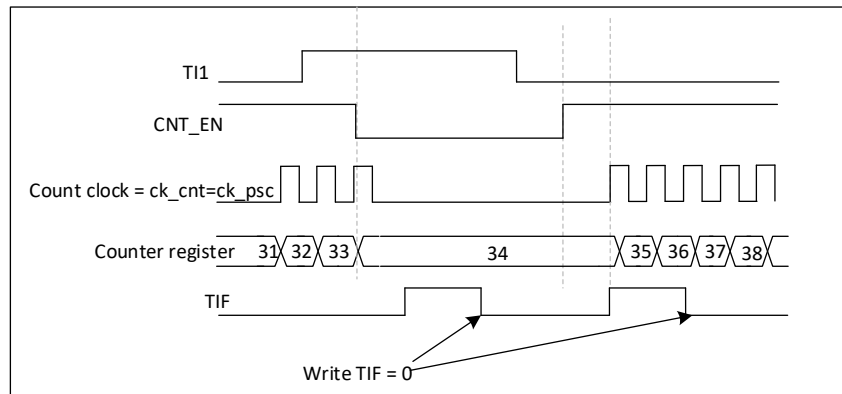


图 25-28 门控模式下的控制电路

### 25.5.3. 从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道2检测 TI2 的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC2S=01。置 TIMx\_CCER 寄存器中 CC2P=1 以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx\_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

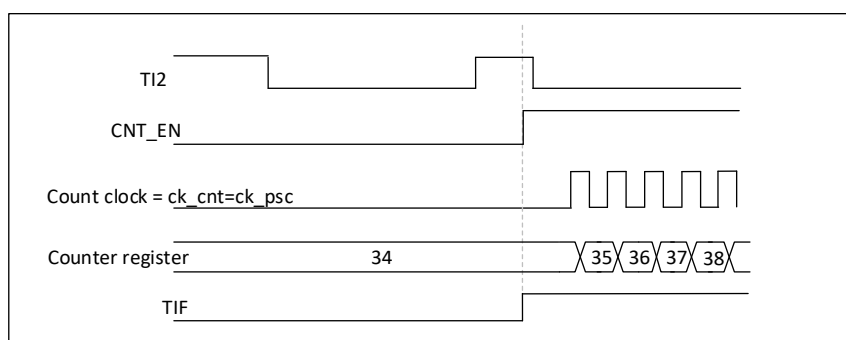


图 25-29 触发器模式下的控制电路

### 25.5.4. 从模式：外部时钟模式 2 + 触发模式

外部时钟模式2可以与另一种从模式（外部时钟模式1和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx\_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx\_SMCR 寄存器配置外部触发输入电路：

- ETF=0000：没有滤波
- ETPS=00：不用预分频器
- ETP=0：检测 ETR 的上升沿，置 ECE=1使能外部时钟模式2。

2. 按如下配置通道1，检测 TI 的上升沿：

- IC1F=0000：没有滤波
- 触发操作中不使用捕获预分频器，不需要配置
- 置 TIMx\_CCMR1寄存器中 CC1S=01，选择输入捕获源
- 置 TIMx\_CCER 寄存器中 CC1P=0以确定极性（只检测上升沿）

3. 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。

当 TI1上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

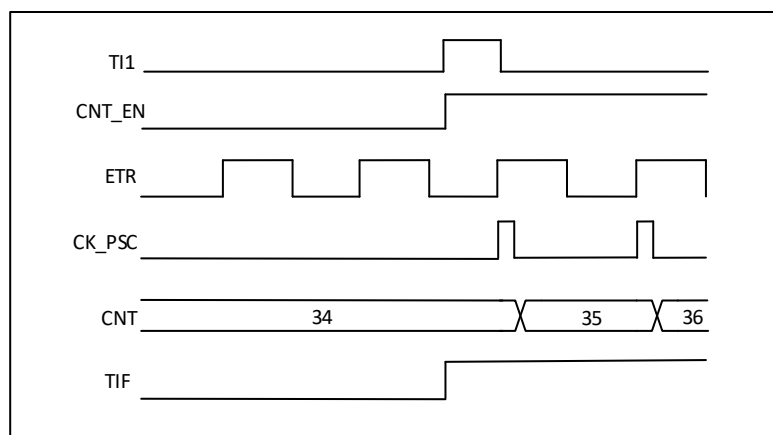


图 25-30 外部时钟模式2 + 触发模式下的控制电路

### 25.5.5. TIM 和外部的触发同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

在以下的例子中，TI1输入端的上升沿导致向上计数器被清零：

- 配置通道1以检测 TI1的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0以确定极性（只检测上升沿）。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1出现一个上升沿；此时，计数器被清零然后从0重新开始计数。同时，触发标志（TIMx\_SR 寄存器中的 TIF 位）被设置，根据 TIMx\_DIER 寄存器中 TIE（中断使能）位和 TDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重载寄存器 TIMx\_ARR=0x36时的动作。在 TI1上升沿和计数器的实际复位之间的延时取决于 TI1输入端的重同步电路。

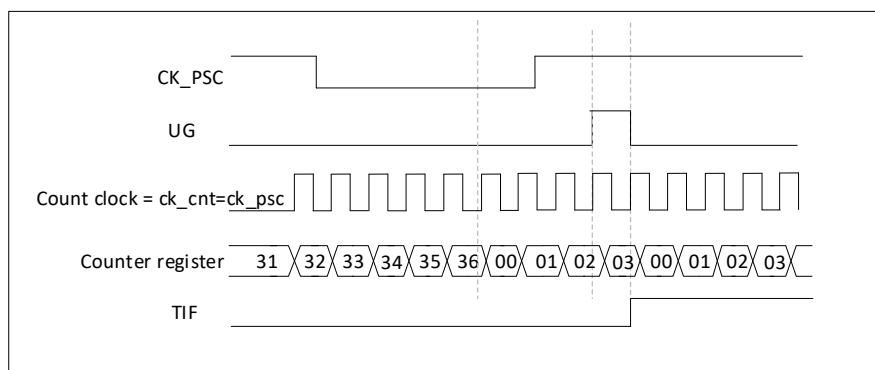


图 25-31 复位模式下的控制电路

### 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1为低时向上计数：

- 配置通道1以检测 TI1上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1作为输入源。
- 置 TIMx\_CR1寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1为低，计数器开始依据内部时钟计数，一旦 TI1变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标志。

TI1上升沿和计数器实际停止之间的延时取决于 TI1输入端的重同步电路。

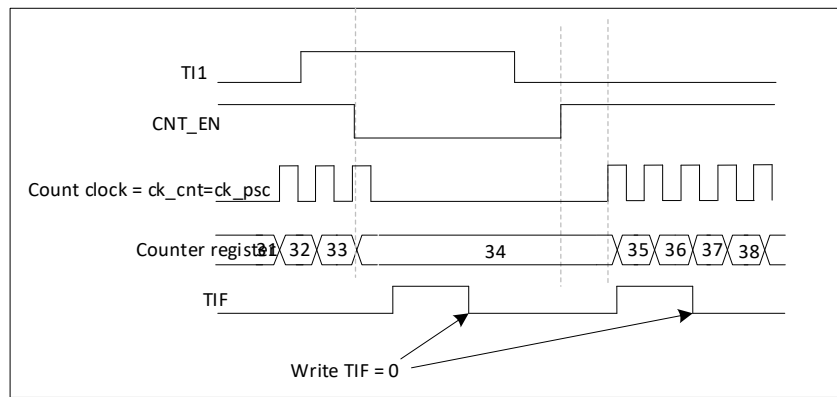


图 25-32 门控模式下的控制电路

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2输入的上升沿开始向上计数：

- c) 配置通道2检测 TI2的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx\_CCMR1寄存器中 CC2S=01。置 TIMx\_CCER 寄存器中 CC2P=1以确定极性（只检测低电平）。
- d) 置 TIMx\_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx\_SMCR 寄存器中 TS=110，选择 TI2作为输入源。

当 TI2出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。TI2上升沿和计数器启动计数之间的延时，取决于 TI2输入端的重同步电路。

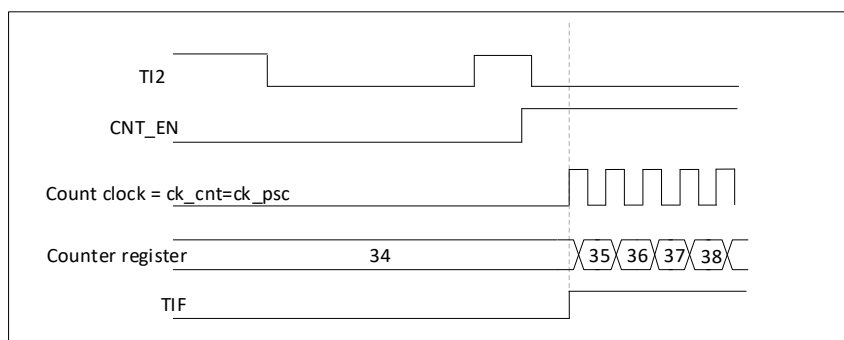


图 25-33 门控模式下的控制电路

### 从模式：外部时钟模式 2 + 触发模式

外部时钟模式2可以与另一种从模式（外部时钟模式1和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx\_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

- 通过 TIMx\_SMCR 寄存器配置外部触发输入电路：
  - ETF=0000：没有滤波
  - ETPS=00：不用预分频器
  - ETP=0：检测 ETR 的上升沿，置 ECE=1使能外部时钟模式2。
- 按如下配置通道1，检测 TI 的上升沿：
  - IC1F=0000：没有滤波
  - 触发操作中不使用捕获预分频器，不需要配置



- 置 TIMx\_CCMR1 寄存器中 CC1S=01, 选择输入捕获源
- 置 TIMx\_CCER 寄存器中 CC1P=0 以确定极性 (只检测上升沿)
- 置 TIMx\_SMCR 寄存器中 SMS=110, 配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS=101, 选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时, TIF 标志被设置, 计数器开始在 ETR 的上升沿计数。ETR 信号的上升沿和计数器实际复位间的延时, 取决于 ETRP 输入端的重同步电路。

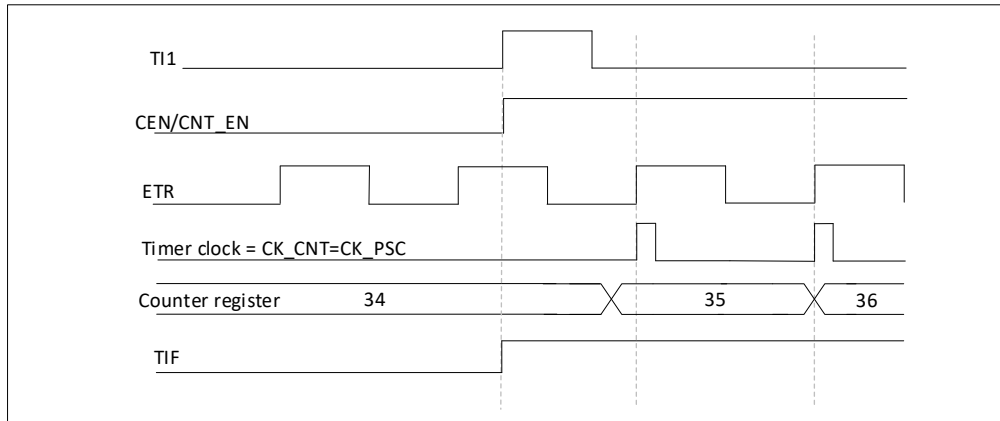


图 25-34 外部时钟模式 2 + 触发模式下的控制电路

## 25.6. 定时器同步 (仅 TIM15)

所有 TIMx 定时器在内部相连, 用于定时器同步或链接。当一个定时器处于主模式时, 它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

### 25.6.1. 使用一个定时器作为另一个定时器的预分频器

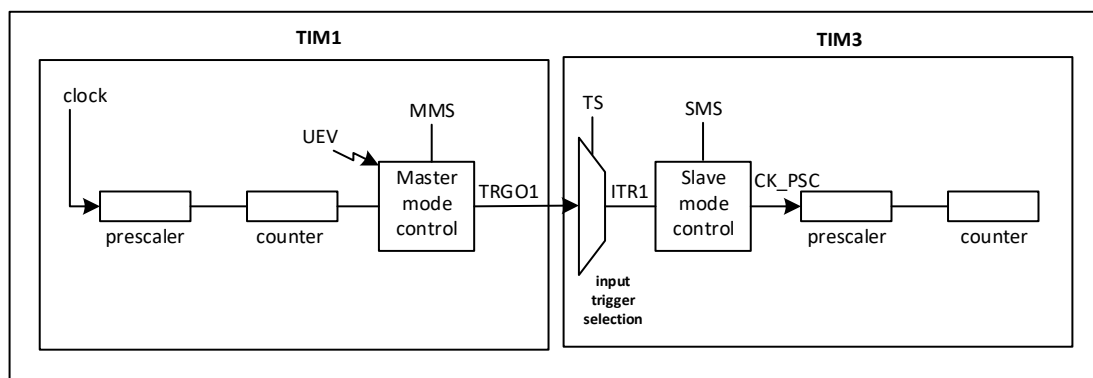


图 25-35 主/从定时器的例子

如: 可以配置定时器1作为定时器2的预分频器。参考图4-48, 进行下述操作:

- 配置定时器1为主模式, 它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM15\_16\_17\_CR2 寄存器的 MMS= '010' 时, 每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。

- 连接定时器1的 TRGO1输出至定时器2，设置 TIM2\_SMCR 寄存器的 TS= '000' ，配置定时器2为使用 ITR1作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式1 (TIM2\_SMCR 寄存器的 SMS=111) ；这样定时器2即可由定时器1周期性的上升沿 (即定时器1的计数器溢出) 信号驱动。
- 最后，必须设置相应 (TIMx\_CR1寄存器) 的 CEN 位分别启动两个定时器。

注：如果 OCx 已被选中为定时器1的触发输出 (MMS=1xx) ，它的上升沿用于驱动定时器2的计数器。

### 25.6.2. 使用一个定时器使能另一个定时器

在这个例子中，定时器2的使能由定时器1的输出比较控制。参考图4-48的连接。只当定时器1的 OC1REF 为高时，定时器2才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以3 ( $f_{CK\_CNT}=f_{CK\_INT}/3$ ) 得到。

- 配置定时器1为主模式，送出它的输出比较参考信号 (OC1REF) 为触发输出 (TIM15\_16\_17\_CR2寄存器的 MMS=100)
- 配置定时器1的 OC1REF 波形 (TIM15\_16\_17\_CCMR1寄存器)
- 配置定时器2从定时器1获得输入触发 (TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器2为门控模式 (TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM2\_CR1寄存器的 CEN=1以使能定时器2
- 置 TIM15\_16\_17\_CR1寄存器的 CEN=1以启动定时器1

注：定时器2的时钟不与定时器1的时钟同步，这个模式只影响定时器2计数器的使能信号。

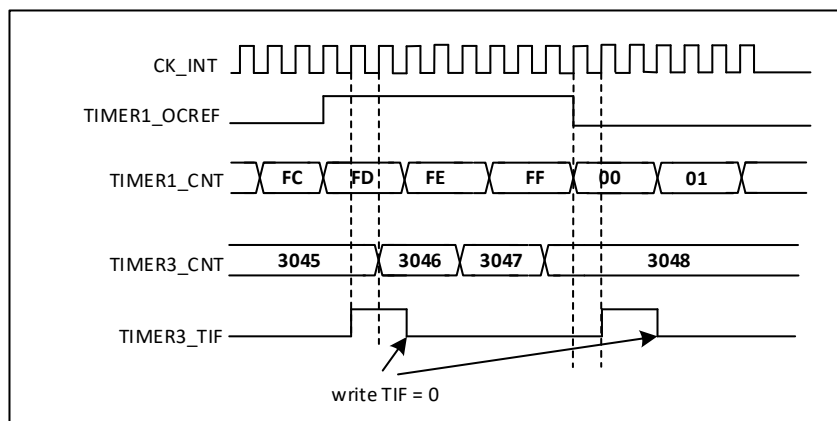


图 25-36 定时器1的 OC1REF 控制定时器2

在图4-47的例子中，在定时器2启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器1之前复位2个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx\_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器1和定时器2。定时器1是主模式并从0开始，定时器2是从模式并从0xE7开始；2个定时器的预分频器系数相同。写 '0' 到 TIM15\_16\_17\_CR1 的 CEN 位将禁止定时器1，定时器2随即停止。

- 配置定时器1为主模式，送出输出比较1参考信号 (OC1REF) 做为触发输出 (TIM15\_16\_17\_CR2寄存器的 MMS=100) 。

- 配置定时器1的 OC1REF 波形 (TIM15\_16\_17\_CCMR1寄存器)。
- 配置定时器2从定时器1获得输入触发 (TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器2为门控模式 (TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM15\_16\_17\_EGR 寄存器的 UG= '1' , 复位定时器1。
- 置 TIM2\_EGR 寄存器的 UG= '1' , 复位定时器2。
- 写 '0xE7' 至定时器2的计数器 (TIM2\_CNT) , 初始化它为0xE7。
- 置 TIM2\_CR1寄存器的 CEN= '1' 以使能定时器2。
- 置 TIM15\_16\_17\_CR1寄存器的 CEN= '1' 以启动定时器1。
- 置 TIM15\_16\_17\_CR1寄存器的 CEN= '0' 以停止定时器1。

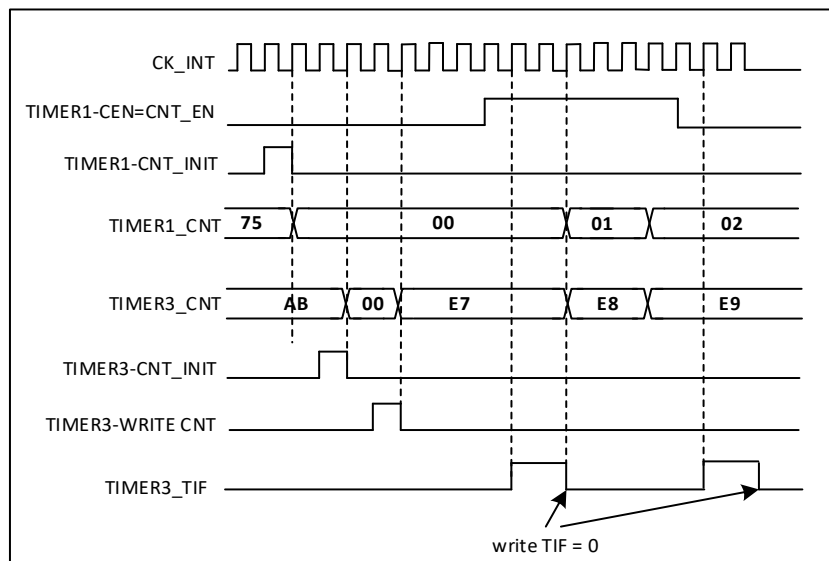


图 25-37 通过使能定时器1可以控制定时器2

### 25.6.3. 使用一个定时器去启动另一个定时器

在这个例子中，使用定时器1的更新事件使能定时器2。参考图4-47的连接。一旦定时器1产生更新事件，定时器2即从它当前的数值（可以是非0）按照分频的内部时钟开始计数。在收到触发信号时，定时器2的 CEN 位被自动地置 '1'，同时计数器开始计数直到写 '0' 到 TIM2\_CR1寄存器的 CEN 位。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以3 ( $f_{CK\_CNT}=f_{CK\_INT}/3$ )。

- 配置定时器1为主模式，送出它的更新事件 (UEV) 做为触发输出 (TIM15\_16\_17\_CR2寄存器的 MMS=010)。
- 配置定时器1的周期 (TIM15\_16\_17\_ARR 寄存器)。
- 配置定时器2从定时器1获得输入触发 (TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器2为触发模式 (TIM2\_SMCR 寄存器的 SMS=110)
- 置 TIM15\_16\_17\_CR1寄存器的 CEN=1以启动定时器1。

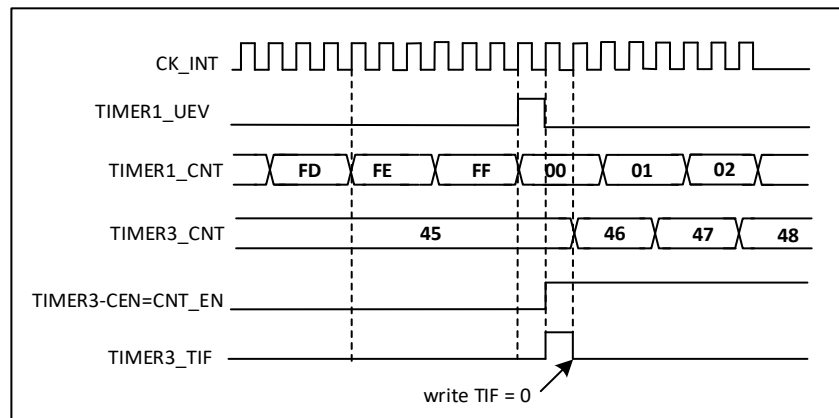


图 25-38 使用定时器1的更新触发定时器2

在上一个例子中，可以在启动计数之前初始化两个计数器。显示在与0相同配置情况下，使用触发模式而不是门控模式（TIM2\_SMCR寄存器的SMS=110）的动作。

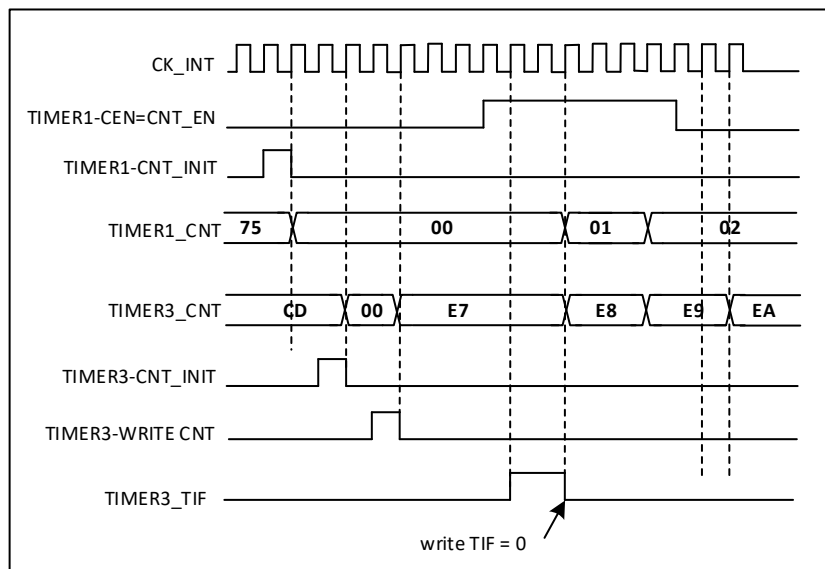


图 25-39 利用定时器1的使能触发定时器2

#### 25.6.4. 使用一个外部触发同步地启动 2 个定时器

这个例子中当定时器1的 TI1输入上升时使能定时器1，使能定时器1的同时使能定时器2，参见图25-40。保证计数器的对齐，定时器1必须配置为主/从模式（对应 TI1为从，对应定时器2为主）：

- 配置定时器1为主模式，送出它的使能做为触发输出（TIM15\_16\_17\_CR2寄存器的MMS=001）。
- 配置定时器1为从模式，从 TI1获得输入触发（TIM15\_16\_17\_SMCR寄存器的TS=100）。
- 配置定时器1为触发模式（TIM15\_16\_17\_SMCR寄存器的SMS=110）。
- 配置定时器1为主/从模式，TIM15\_16\_17\_SMCR寄存器的MSM=1。
- 配置定时器2从定时器1获得输入触发（TIM2\_SMCR寄存器的TS=000）
- 配置定时器2为触发模式（TIM2\_SMCR寄存器的SMS=110）。

当定时器1的 TI1上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化（设置相应的 UG 位），两个计数器都从0开始，但可以通过写入任意一个计数器寄存器（TIMx\_CNT）在定时器间插入一个偏移。下图中能看到主/从模式下在定时器1的 CNT\_EN 和 CK\_PSC 之间有个延迟。

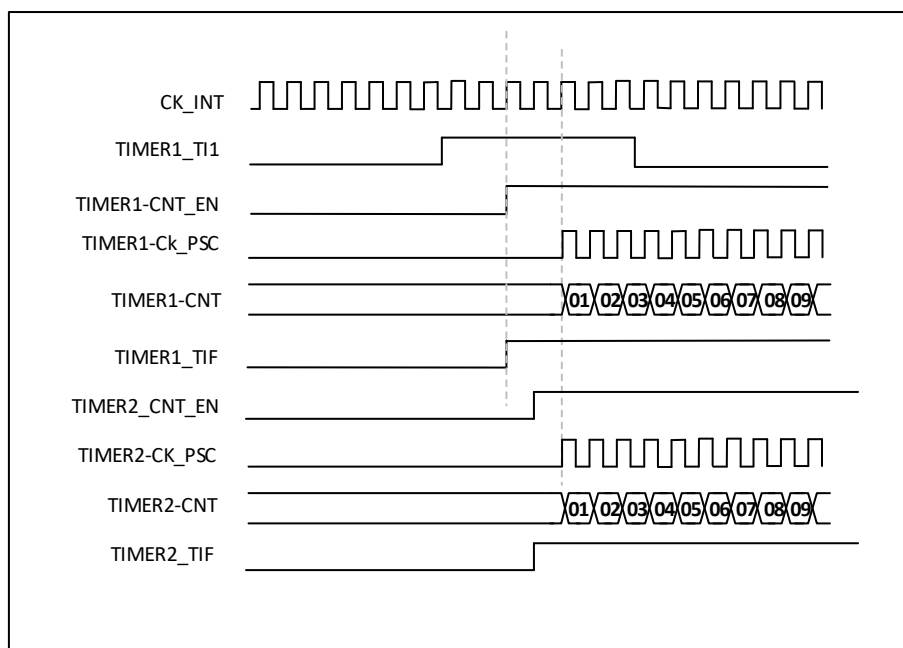


图 25-40 使用定时器1的 TI1输入触发定时器1和定时器2

### 25.6.5. 调试模式

当芯片进入调试模式时，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器可以继续正常工作或者停止工作。

## 25.7. TIM15寄存器描述

0x4001 4800 - 0x4001 4BFF TIM17

0x4001 4400 - 0x4001 47FF TIM16

0x4001 4000 - 0x4001 43FF TIM15

### 25.7.1. TIM15 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9: 8	CKD[1: 0]	RW	00	<p>时钟分频因子</p> <p>这2位定义在定时器时钟 (CK_INT) 频率, 死区时间和由死区发生器与数字滤波器 (ETR,Tix) 所用的采样时钟之间的分频比例</p> <p>00: tDTS = tCK_INT</p> <p>01: tDTS = 2 x tCK_INT</p> <p>10: tDTS = 4 x tCK_INT</p> <p>11: 保留, 不要使用这个配置</p>
7	ARPE	RW	0	<p>自动重装载预装载允许位</p> <p>0: TIMx_ARR 寄存器没有缓冲</p> <p>1: TIMx_ARR 寄存器被装入缓冲器</p>
6: 4	保留	-	-	保留
3	OPM	RW	0	<p>单脉冲模式</p> <p>0: 在发生更新事件时, 计数器不停止</p> <p>1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。</p>
2	URS	RW	0	<p>更新请求源</p> <p>软件通过该位选择 UEV 事件的源</p> <p>0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求</p>
1	UDIS	RW	0	<p>禁止更新</p> <p>软件通过该位允许/禁止 UEV 事件的产生</p> <p>0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>被缓存的寄存器被装入它们的预装载值。</p> <p>1: 禁止 UEV。不产生更新事件, 影子寄存器 (ARR,PSC,CCRx) 保持它们的值。</p> <p>如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。</p>

0	CEN	RW	0	<p>允许计数器</p> <p>0: 禁止计数器</p> <p>1: 开启计数器</p> <p>注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。</p>
---	-----	----	---	--

## 25.7.2. TIM15 控制寄存器 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					OIS2	OIS1N	OIS1	Res	MMS[2: 0]			CCDS	CCUS	Res	CCPC
-					RW	RW	RW	-	RW	RW	RW	RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
15: 11	保留	-	-	保留
10	OIS2	RW	0	输出空闲状态2 (OC2输出)。参见 OIS1位
9	OIS1N	RW	0	<p>输出空闲状态1 (OC1N 输出)。</p> <p>0: 当 MOE=0时, 死区后 OC1N=0</p> <p>1: 当 MOE=0时, 死区后 OC1N=1</p> <p>注: 已经设置了 LOCK (TIMx_BDTR 寄存器) 级别1、2或3后, 该位不能被修改。</p>
8	OIS1	RW	0	<p>输出空闲状态1 (OC1输出)。</p> <p>0: 当 MOE=0时, 如果实现了 OC1N, 则死区后 OC1=0</p> <p>1: 当 MOE=0时, 如果实现了 OC1N, 则死区后 OC1=1</p> <p>注: 已经设置了 LOCK (TIMx_BDTR 寄存器) 级别1、2或3后, 该位不能被修改。</p>
7	保留	-	-	保留
6: 4	MMS[2: 0]	RW	000	<p>主模式选择</p> <p>这3位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下:</p> <p>000: 复位 - TIMx_EGR 寄存器的 UG 位被用于作为触发输出 (TRGO)。如果触发输入 (复位模式下的从模式控制器) 产生复位, 则 TRGO 上的信号相对实际的复位会有一个延迟。</p> <p>001: 允许 - 计数器使能信号 CNT_EN 被用于作为触发输出 (TRGO)。有时需要</p>

Bit	Name	R/W	Reset Value	Function
				<p>在同一时间启动多个定时器或控制从定时器的一个窗口。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式（见 TIMx_SMCR 寄存器中 MSM 位的描述）。</p> <p>010: 更新 - 更新事件被选为触发输入（TRGO）。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。</p> <p>011: 比较脉冲 - 一旦发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时（即是它已经为高），触发输出送出一个正脉冲（TRGO）。</p> <p>100: 比较 - OC1REF 信号被用于作为触发输出（TRGO）。</p> <p>101: 比较 - OC2REF 信号被用于作为触发输出（TRGO）。</p> <p>注意： 1. 从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。 2. 若主从定时器不在同一总线上，主模式应该配置为能被从定时器采到的宽度。</p>
3	CCDS	RW	0	<p>捕获/比较的 DMA 选择</p> <p>0: 当发生 CCx 事件时，送出 CCx 的 DMA 请求。 1: 当发生更新事件时，送出 CCx 的 DMA 请求。</p>
2	CCUS	RW	0	<p>捕获/比较控制更新选择</p> <p>0: 如果捕获/比较控制位是预装载的（CCPC=1），只能通过设置 COM 位更新它们。 1: 如果捕获/比较控制位是预装载的（CCPC=1），可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注：该位只对具有互补输出的通道起作用。</p>
1	保留	-	-	保留
0	CCPC	RW	0	<p>捕获/比较预装载控制位</p> <p>0: CCxE, CCxNE 和 OCxM 位不是预装载的。 1: CCxE, CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新。 注：该位只对具有互补输出的通道起作用。</p>

### 25.7.3. TIM15 从模式控制寄存器 (TIMx\_SMCR)

Address offset: 0x08



Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								MSM	TS[2: 0]			Res	SMS[2: 0]		
-								RW	RW			-	RW		

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7	MSM	RW	0	主/从模式 0: 无作用 1: 触发输入 (TRGI) 上的事件被延迟了, 以允许在当前定时器 (通过 TRGO) 与它的当前定时器和从定时器间的同步 (通过 TRGO) 。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的
6: 4	TS[2: 0]	RW	000	触发选择, 这3位选择用于同步计数器的触发输入。 000: TIM2 (ITR0) 001: TIM3 (ITR1) 010: TIM16 (ITR2) 011: TIM17 (ITR3) 100: TI1的边沿检测器 (TI1F_ED) 101: 滤波后的定时器输入1 (TI1FP1) 110: 滤波后的定时器输入2 (TI2FP2) 111: 外部触发输入 (ETRF) 注: 为避免在信号转变时产生错误的边沿检测, 必须在未使用这些位时修改它们
3	保留	-	-	保留
2: 0	SMS[2: 0]	RW	000	从模式选择。当选择了外部信号, 触发信号 (TRGI) 的有效边沿与选中的外部输入极性相关 (见输入控制寄存器和控制寄存器的说明) 000: 关闭从模式 如果 CEN=1, 则预分频器直接由内部时钟驱动。 100: 复位模式 选中的触发输入 (TRGI) 的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。 101: 门控模式 当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的。 110: 触发模式 计数器在触发输入 TRGI 的上升沿启动 (但不复位), 只有计数器的启动是受控的。

				<p>111: 外部时钟模式1</p> <p>选中的触发输入 (TRGI) 的上升沿驱动计数器。</p> <p>注: 如果 TI1F_EN 被选为触发输入 (TS=100) 时, 不要使用门控模式。这是因为, TI1F_ED 在每次 TI1F 变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。</p> <p>注: 在编码器模式下, 不要使用 uev 作为 trgo 输出信号, (即 mms 不能配置为010)</p>
--	--	--	--	--

## TIM15 内部触发连接

Slave TIM	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM15	TIM2_TRGO	TIM3_TRGO	TIM16 OC1	TIM17 OC

## 25.7.4. TIM15 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	COMDE	Res		CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res		CC2IE	CC1IE	UIE
-	RW	RW	-	-	RW						-	-	RW		

Bit	Name	R/W	Reset Value	Function
15	保留	-	-	保留
14	TDE	RW	0	TDE: 允许触发 DMA 请求 0: 禁止触发 DMA 请求 1: 允许触发 DMA 请求
13	COMDE	RW	0	COMDE: 允许 COM 的 DMA 请求 0: 禁止 COM 的 DMA 请求 1: 允许 COM 的 DMA 请求
11: 12	保留	-	-	保留
10	CC2DE	RW	0	CC2DE: 允许捕获/比较2的 DMA 请求 0: 禁止捕获/比较2的 DMA 请求 1: 允许捕获/比较2的 DMA 请求
9	CC1DE	RW	0	CC1DE: 允许捕获/比较1的 DMA 请求 0: 禁止捕获/比较1的 DMA 请求 1: 允许捕获/比较1的 DMA 请求
8	UDE	RW	0	UDE: 允许更新的 DMA 请求 0: 禁止更新的 DMA 请求 1: 允许更新的 DMA 请求

7	BIE	RW	0	BIE: 允许刹车中断 0: 禁止刹车中断 1: 允许刹车中断
6	TIE	RW	0	TIE: 允许触发中断 0: 禁止触发中断 1: 允许触发中断
5	COMIE	RW	0	COMIE: 允许 COM 中断 0: 禁止 COM 中断 1: 允许 COM 中断
4: 3	保留	-	-	保留
2	CC2IE	RW	0	CC2IE: 允许捕获/比较2中断 0: 禁止捕获/比较2中断 1: 允许捕获/比较2中断
1	CC1IE	RW	0	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断 1: 允许捕获/比较1中断
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

### 25.7.5. TIM15 状态寄存器 (TIMx\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	CC2OF	CC1OF	Res	BIF	TIF	COMIF	Res	Res	CC2IF	CC1IF	UIF
-	-	-	-	-	RC_W0		-	RC_W0		-	-	RC_W0			

Bit	Name	R/W	Reset Value	Function
15: 11	保留	-	-	保留
10	CC2OF	RC_W0	0	捕获/比较2 过捕获标记 参见 CC1OF 描述
9	CC1OF	RC_W0	0	捕获/比较1 过捕获标记 仅当相应的通道被配置为输入捕获时，该标记可由硬件置1。写0可清除该位。 0: 无过捕获产生； 1: CC1OF 置1时，计数器的值已经被捕获到 TIMx_CCR1 寄存器。
8	保留	-	-	保留
7	BIF	RC_W0	0	刹车中断标记

Bit	Name	R/W	Reset Value	Function
				一旦刹车输入有效, 由硬件对该位置1。如果刹车输入无效, 则该位可由软件清0。 0: 无刹车事件产生; 1: 刹车输入上检测到有效电平。
6	TIF	RC_W0	0	触发器中断标记 当发生触发事件 (当从模式控制器处于除门控模式外的其它模式时,在 TRGI 输入端检测到有效边沿, 或或门控模式下的任一边沿) 时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生; 1: 触发器中断等待响应
5	COMIF	RC_W0	0	COM 中断标记 一旦产生 COM 事件 (当 CcxE、CcxNE、OCxM 已被更新) 该位由硬件置1。它由软件清0。 0: 无 COM 事件产生; 1: COM 中断等待响应
4: 3	保留	-	-	保留
2	CC2IF	RC_W0	0	捕获/比较2 中断标记 参考 CC1IF 描述
1	CC1IF	RC_W0	0	捕获/比较1 中断标记 如果通道 CC1配置为输出模式: 当计数器值与比较值匹配时该位由硬件置1, 但在中心对称模式下除外 (参考 TIMx_CR1寄存器的 CMS 位)。它由软件清0。 0: 无匹配发生; 1: TIMx_CNT 的值与 TIMx_CCR1的值匹配。 如果通道 CC1配置为输入模式: 当捕获事件发生时该位由硬件置1, 它由软件清0或通过读 TIMx_CCR1清0。 0: 无输入捕获产生; 1: 输入捕获产生并且计数器值已装入 TIMx_CCR1 (在 IC1上检测到与所选极性相同的边沿)。 注: 当 CEN 打开, 该位也会被置位。
0	UIF	RC_W0	0	更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0: 无更新事件产生; 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1:

Bit	Name	R/W	Reset Value	Function
				<ul style="list-style-type: none"> <li>- 若 TIMx_CR1寄存器的 UDIS=0, 当 REP_CNT=0时产生更新事件计数器上溢时) ;</li> <li>- 若 TIMx_CR1寄存器的 UDIS=0、URS=0, 当 TIMx_EGR 寄存器的 UG=1时产生更新事件 (软件对 CNT 重新初始化) ;</li> <li>- 若 TIMx_CR1寄存器的 UDIS=0、URS=0, 当 CNT 被触发事件重初始化时产生更新事件。(参考从模式控制寄存器 (TIMx_SMCR) )</li> </ul>

### 25.7.6. TIM15 事件产生寄存器 (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	Res	Res	CC2G	CC1G	UG
-	-	-	-	-	-	-	-	W	W	W	-	-	W	W	W

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7	BG	W	0	产生刹车事件 该位由软件置1, 用于产生一个刹车事件, 由硬件自动清0。 0: 无动作; 1: 产生一个刹车事件。此时 MOE=0、BIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
6	TG	W	0	产生触发事件 该位由软件置1, 用于产生一个触发事件, 由硬件自动清0。 0: 无动作; 1: TIMx_SR 寄存器的 TIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
5	COMG	W	0	捕获/比较事件, 产生控制更新 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 当 CCPC=1, 允许更新 CcxE、CcxNE、OCxM 位。 注: 该位只对有互补输出的通道有效。
4: 3	保留	-	-	保留
2	CC2G	W	0	产生捕获/比较2事件 参考 CC1G 描述
1	CC1G	W	0	产生捕获/比较1事件

Bit	Name	R/W	Reset Value	Function
				<p>该位由软件置1，用于产生一个捕获/比较事件，由硬件自动清0。</p> <p>0：无动作；</p> <p>1：在通道 CC1上产生一个捕获/比较事件：</p> <p>若通道 CC1配置为输出：</p> <p>设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。</p> <p>若通道 CC1配置为输入：</p> <p>当前的计数器值捕获至 TIMx_CCR1寄存器，设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为1，则设置 CC1OF=1。</p>
0	UG	W	0	<p>产生更新事件。该位由软件置1，硬件自动清0。</p> <p>0：无动作；</p> <p>1：重新初始化计数器，并产生一个更新事件。注意：预分频器的计数器也被清0（但是预分频系数不变）。</p>

### 25.7.7. TIM15 捕获/比较模式寄存器 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	OC2M[2: 0]			OC2PE	CO2FE	CC2S[1: 0]		Res	OC1M[2: 0]			OC1PE	OC1FE	CC1S[1: 0]	
IC2F[3: 0]				IC2PSC[1: 0]		0]		IC1F[3: 0]			IC1PSC[1: 0]		0]		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

#### 输出比较模式

Output compare mode:

Bit	Name	R/W	Reset Value	Function
15	保留	-	-	保留
14: 12	OC2M[2: 0]	RW	0	输出比较2模式选择
11	OC2PE	RW	0	输出比较2预装载使能
10	OC2FE	RW	0	输出比较2快速使能
9: 8	CC2S[1: 0]	RW	00	<p>捕获/比较2选择。</p> <p>该位定义通道的方向（输入/输出），及输入脚的选择：</p> <p>00：CC2通道被配置为输出；</p> <p>01：CC2通道被配置为输入，IC2映射在 TI2上；</p> <p>10：CC2通道被配置为输入，IC2映射在 TI1上；</p> <p>11：CC2通道被配置为输入，IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时</p>

Bit	Name	R/W	Reset Value	Function
				(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E=0) 才是可写的。
7	保留	-	-	保留
6: 4	OC1M[2: 0]	RW	00	<p>输出比较1模式</p> <p>该3位定义了输出参考信号 OC1REF 的动作, 而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效, 而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。</p> <p>000: 冻结。输出比较寄存器 TIMx_CCR1与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用;</p> <p>001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时, 强制 OC1REF 为高。</p> <p>010 : 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时, 强制 OC1REF 为低。</p> <p>011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。</p> <p>100: 强制为无效电平。强制 OC1REF 为低。</p> <p>101: 强制为有效电平。强制 OC1REF 为高。</p> <p>110: PWM 模式1 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为有效电平, 否则为无效电平 ;</p> <p>111: PWM 模式2 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为无效电平, 否则为有效电平;</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 在 PWM 模式1或 PWM 模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较1预装载使能</p> <p>0: 禁止 TIMx_CCR1寄存器的预装载功能, 可随时写入 TIMx_CCR1寄存器, 且新值马上起作用。</p> <p>1: 开启 TIMx_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。</p>

Bit	Name	R/W	Reset Value	Function
				<p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 仅在单脉冲模式下, 可以在未确认预装载寄存器情况下使用 PWM 模式, 否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较1 快速使能</p> <p>该位用于加快 CC 输出对触发器输入事件的响应。</p> <p>0: 根据计数器与 CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活 CC1输出的最小延时为5个时钟周期。</p> <p>1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1输出间的延时被缩短为3个时钟周期。</p> <p>OCFE 的只在通道被配置成 PWM1或 PWM2模式时起作用。</p>
1: 0	CC1S[1: 0]	RW	00	<p>捕获/比较1 选择。</p> <p>这2位定义通道的方向 (输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC1映射在 TI1上;</p> <p>10: CC1通道被配置为输入, IC1映射在 TI2上;</p> <p>11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。</p> <p>注: CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E=0) 才是可写的。</p>

## 输入捕获模式

### Input Capture mode:

Bit	Name	R/W	Reset Value	Function
15: 12	IC2F	RW	0	输入捕获2滤波器
11: 10	IC2PSC[1: 0]	RW	0	捕获/比较2预分频器
9: 8	CC2S[1: 0]	RW	0	<p>捕获/比较2选择。</p> <p>这2位定义通道的方向 (输入/输出), 及输入脚的选择:</p> <p>00: CC2通道被配置为输出;</p> <p>01: CC2通道被配置为输入, IC2映射在 TI2上;</p> <p>10: CC2通道被配置为输入, IC2映射在 TI1上;</p> <p>11: CC2通道被配置为输入, IC2映射在 TRC 上。此模式仅工作在内部触发器输入被选中时</p>



Bit	Name	R/W	Reset Value	Function
				(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC2S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC2E=0) 才是可写的。
7: 4	IC1F[3: 0]	RW	0000	输入捕获1滤波器 这几位定义了 TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成, 它记录到 N 个事件后会产生一个输出的跳变: 0000: 没有滤波器, 在 fDTS 下采样 0001: fSAMPLING=fCK_INT, N=2 0010: fSAMPLING=fCK_INT, N=4 0011: fSAMPLING=fCK_INT, N=8 0100: fSAMPLING=fDTS/2, N=6 0101: fSAMPLING=fDTS/2, N=8 0110: fSAMPLING=fDTS/4, N=6 0111: fSAMPLING=fDTS/4, N=8 1000: fSAMPLING=fDTS/8, N=6 1001: fSAMPLING=fDTS/8, N=8 1010: fSAMPLING=fDTS/16, N=5 1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	捕获/比较1预分频器 00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获; 01: 每2个事件触发一次捕获; 10: 每4个事件触发一次捕获; 11: 每8个事件触发一次捕获。
1: 0	CC1S[1: 0]	RW	00	CC1S[1: 0]: 捕获/比较1选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC1映射在 TI1上; 10: CC1通道被配置为输入, IC1映射在 TI2上; 11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC1S 仅在通道关闭时 (TIMx_CCER 寄存器的 CC1E=0) 才是可写的。

## 25.7.8. TIM15 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								CC2NP	Res	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
-								RW	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7	CC2NP	RW	0	捕获/比较2互补输出极性。参考 CC1NP 的描述。
6	保留	-	-	保留
5	CC2P	RW	0	捕获/比较2输出极性。参考 CC1P 的描述。
4	CC2E	RW	0	捕获/比较2输出使能。参考 CC1E 的描述。
3	CC1NP	RW	0	捕获/比较1互补输出极性 0: OC1N 高电平有效 1: OC1N 低电平有效 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为3或2且 CC1S=00 (通道配置为输出) 则该位不能被修改。
2	CC1NE	RW	0	捕获/比较1互补输出使能 0: OC1N 禁止输出 1: OC1N 信号输出到对应的输出引脚 当 CC1通道配置为输出时, OC1N 输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定, 见下表
1	CC1P	RW	0	捕获/比较1输出极性 CC1通道配置为输出: 0: OC1高电平有效 1: OC1低电平有效 CC1通道配置为输入: CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1和 TI2FP1的极性。 00: 不反相/上升沿: TIxFP1上升沿有效 (捕获、复位模式下触发、外部时钟或触发模式下); TIxFP1不反相 (门控模式、编码器模式)。 01: 反相/下降沿: TIxFP1下降沿有效 (捕获、复位模式下触发、外部时钟或触发模式下);

Bit	Name	R/W	Reset Value	Function
				<p>TIxFP1反相（门控模式、编码器模式）。</p> <p>10: 保留，不要使用这个配置。</p> <p>11: 不反相/双沿</p> <p>TIxFP1上升和下降沿都有效（捕获、复位模式下触发、外部时钟或触发模式下）；</p> <p>TIxFP1不反相（门控模式）。这个配置不能应用于编码器模式下。</p> <p>注：</p> <p>1.对于互补输出通道，这一位是预载的。如果 TIMx_CR2寄存器中的 CCPC 位被设置，那么 CC1P 的实际有效位只有在 com 事件发生时才会加载预载值。</p> <p>2.一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为3或2，则该位不能被修改</p>
0	CC1E	RW	0	<p>捕获/比较1输出使能</p> <p>0: 捕获模式关闭/OC1禁止输出</p> <p>1: 捕获模式开启/OC1信号输出到对应的输出引脚</p> <p>当 CC1通道配置为输出时，OC1输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定，见下表</p> <p>注：</p> <p>对于互补输出通道，这一位是预载的。如果 TIMx_CR2寄存器中的 CCPC 位被设置，那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。</p>

表25-1具有中断功能的互补 OCx 和 OCxN 通道的输出控制

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	X	0	0	0	输出禁止（与定时器断开），OCx=0，OCx_EN=0	输出禁止（与定时器断开），OCxN=0，OCxN_EN=0
		0	0	1	输出禁止（与定时器断开），OCx=0，OCx_EN=0	OCxREF + Polarity OCxN=OCxREF 异或 CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCREF 异或 CCxP, OCx_EN=1	输出禁止（与定时器断开），OCxN=0，OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	OCREF 的互补 (not OCREF) + Polarity + dead-time OCxN_EN=1

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
		1	0	0	输出禁止 (与定时器断开) , OCx=CCxP, OCx_EN=0	输出禁止 (与定时器断 开) , OCxN=CCxNP, OCxN_EN=0
		1	0	1	输出禁止 (与定时器断开) , OCx=CCxP, OCx_EN=1	OCxREF+Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+Polarity OCx=OCxREF xor CCxP, OCx_EN=1	关闭状态 (输出使能且为 无效电平) , OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF+Polarity + dead-time OCx_EN=1	OCREF 的互补 (not OCREF) + polarity + dead-time OCN_EN=1
0	X	0	0	0	输出禁止 (与定时器断开)	
		0	0	1		
		0	1	0		
		0	1	1		
		1	0	0	关闭状态 (输出使能且为无效电平) 异步的: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 若时钟存在: 经过一个死区时间后, 假设 OISx 与 OISxN 并不都对 应 OCx 和 OCxN 的有效电平, OCx=OISx 和 OCxN=OISxN	
		1	0	1		
		1	1	0		
		1	1	1		

如果一个通道的2个输出都没有使用 (CCxE = CCxNE = 0) , 那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

注: 引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态, 取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

### 25.7.9. TIM15 计数器 (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CNT[15: 0]	RW	0	计数器的值

### 25.7.10. TIM15 预分频器 (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	PSC[15: 0]	RW	0	预分频器的值 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清0或被工作在复位模式的从控制器清0。

### 25.7.11. TIM15 自动重载寄存器 (TIMx\_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
15: 0	ARR[15: 0]	RW	FFFF	自动重载的值 ARR 包含了将要装载入实际的自动重载寄存器的值。 当自动重载的值为空时，计数器不工作。

### 25.7.12. TIM15 重复计数器寄存器 (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7: 0	REP[7: 0]	RW	0	<p>周期计数器的值</p> <p>开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。</p> <p>每次向下计数 REP_CNT 达到0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件U_RC发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。</p> <p>这意味着在 PWM 模式中，(REP+1) 对应着：</p> <ul style="list-style-type: none"> <li>- 在边沿对齐模式下，PWM 周期的数目；</li> </ul>

### 25.7.13. TIM15 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15: 0	CCR1[15: 0]	RW/RO	0	<p>捕获/比较1的值</p> <p>若 CC1通道配置为输出： CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC1PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较1寄存器中。</p>

				<p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC1端口上输出信号。</p> <p>若 CC1通道配置为输入： CCR1包含了由上一次输入捕获1事件（IC1）传输的计数器值。</p>
--	--	--	--	---

### 25.7.14. TIM15 捕捉/比较寄存器 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15: 0	CCR2[15: 0]	RW/RO	0	<p>捕获/比较2的值</p> <p>若 CC2通道配置为输出： CCR2包含了装入当前捕获/比较2寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC2PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较2寄存器中。</p> <p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC 端口上输出信号。</p> <p>若 CC2通道配置为输入： CCR2包含了由上一次输入捕获2事件（IC2）传输的计数器值。</p>

### 25.7.15. TIM15 刹车和死区寄存器 (TIMx\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1: 0]		DTG[7: 0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

注释：根据锁定设置，AOE、BKP、BKE、OSSI、OSSR 和 DTG[7: 0]位均可被写保护，有必要在第一次写入 TIMx\_BDTR 寄存器时对它们进行配置。

Bit	Name	R/W	Reset Value	Function
15	MOE	RW	0	主输出使能

Bit	Name	R/W	Reset Value	Function
				一旦刹车输入有效, 该位被硬件异步清0。根据 AOE 位的值, 可由软件清0或自动置1。它仅对配置为输出通道有效。 0: 禁止 OC 和 OCN 输出或强制为空闲状态; 1: 如果设置了相应的使能位 (TIMx_CCER 寄存器的 CCxE、CCxNE 位), 则开启 OC 和 OCN 输出。
14	AOE	RW	0	自动输出使能 0: MOE 只能被软件置1; 1: MOE 能被软件置1或在下一个更新事件自动置1 (如果刹车输入无效)。 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。
13	BKP	RW	0	刹车输入极性 0: 刹车输入低电平有效; 1: 刹车输入高电平有效。 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。 任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
12	BKE	RW	0	刹车功能使能 0: 禁止刹车输入; 1: 开启刹车输入。 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1, 则该位不能被修改。 任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
11	OSSR	RW	0	运行模式下“关闭状态”选择 该位用于当 MOE=1且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明 (捕获/比较使能寄存器 (TIMx_CCER) )。 0: 当定时器不工作时, 禁止 OC/OCN 输出 (OC/OCN 使能输出信号=0) ; 1: 当定时器不工作时, 一旦 CCxE=1或 CCxNE=1, 开启 OC/OCN 输出并输出无效电平。 OC/OCN 使能输出信号=1。 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为2, 则该位不能被修改。



Bit	Name	R/W	Reset Value	Function
10	OSSI	RW	0	<p>空闲模式下“关闭状态”选择</p> <p>该位用于当 MOE=0且通道设为输出时。</p> <p>参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器（TIMx_CCER））。</p> <p>0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）；</p> <p>1：当定时器不工作时，一旦 CCxE=1或 CcxNE=1，OC/OCN 首先输出其空闲电平。</p> <p>OC/OCN 使能输出信号=1。</p> <p>注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为2，则该位不能被修改。</p>
9: 8	LOCK[1: 0]	RW	00	<p>锁定设置</p> <p>该位为防止软件错误而提供写保护。</p> <p>00：锁定关闭，寄存器无写保护；</p> <p>01：锁定级别1，不能写入 TIMx_BDTR 寄存器的 DTG/BKE/BKP/AOE 位、TIMx_CR2寄存器的 OISx/OISxN 位；</p> <p>10：锁定级别2，不能写入锁定级别1中的各位，也不能写入 CC 极性位（一旦相关通道通过 CCxS 位设为输出，TIMx_CCER 寄存器的 CCxP/CCNxP 位）以及 OSSR/OSSI 位；</p> <p>11：锁定级别3，不能写入锁定级别2中的各位，也不能写入 CC 控制位（一旦相关通道通过 CCxS 位设为输出，TIMx_CCMRx 寄存器的 OCxM/OCxPE 位）；</p> <p>注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。</p>
7: 0	DTG[7: 0]	RW	0000 0000	<p>死区发生器设置</p> <p>这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间：</p> <p>DTG[7: 5]=0xx =&gt; DT=DTG[7: 0] × Tdtg, Tdtg = TDTS;</p> <p>DTG[7: 5]=10x =&gt; DT= (64+DTG[5: 0]) × Tdtg, Tdtg = 2 × TDTS;</p> <p>DTG[7: 5]=110 =&gt; DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 8 × TDTS;</p> <p>DTG[7: 5]=111 =&gt; DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 16 × TDTS;</p> <p>例：若 TDTS = 125ns (8MHZ)，可能的死区时间为：</p>

Bit	Name	R/W	Reset Value	Function
				0到15875ns, 若步长时间为125ns; 16us 到31750ns, 若步长时间为250ns; 32us 到63us, 若步长时间为1us; 64us 到126us, 若步长时间为2us; 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1、2或3, 则这些位不能被修改。

### 25.7.16. TIM15 DMA 控制寄存器 (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4: 0]					Res			DBA[4: 0]				
-	-	-	RW	RW	RW	RW	RW	-	-	-	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	保留	-	-	保留
12: 8	DBL[4: 0]	RW	0 0000	DMA 连续传送长度 这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器的地址进行读或写时, 定时器则进行一次连续传送), 即: 定义被传送的字节数目: 00000: 1次传输 00001: 2次传输 00010: 3次传输 ..... ..... 10001: 18次传输 例: 我们考虑这样的传输: DBL=7, DBA=TIM2_CR1 - 如果 DBL=7, DBA=TIM2_CR1表示待传输数据的地址, 那么传输的地址由下式给出: (TIMx_CR1的地址) + DBA + (DMA 索引), 其中 DMA 索引 = DBL 其中 (TIMx_CR1的地址) + DBA 再加上7, 给出了将要写入或者读出数据的地址, 这样数据的传输将发生在从地址 (TIMx_CR1的地址) + DBA 开始的7个寄存器。 根据 DMA 数据长度的设置, 可能发生以下情况: - 如果设置数据为半字 (16位), 那么数据就会传输给全部7个寄存器。

Bit	Name	R/W	Reset Value	Function
				- 如果设置数据为字节，数据仍然会传输给全部7个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
7: 5	保留	-	-	保留
4: 0	DBA[4: 0]	RW	0 0000	DBA[4: 0]: DMA 基地址 这些位定义了 DMA 在连续模式下的基地址（当对 TIMx_DMAR 寄存器的地址进行读或写时），DBA 定义为从 TIMx_CR1寄存器所在地址开始的偏移量： 00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR, .....

### 25.7.17. TIM15 连续模式的 DMA 地址 (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DMAB[31: 0]	RW	0	DMA 连续传送寄存器 对 TIMx_DMAR 寄存器的读或写会导致对以下地址的寄存器的存取操作： TIMx_CR1地址 + (DBA + DMA 指针) x4，其中： “TIMx_CR1地址” 是控制寄存器1的地址； “DBA” 是 TIMx_DCR 寄存器中定义的基地址； “DMA 指针” 是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

注：在使用 DMA 连续传输功能时，必须将 DMA 中对应通道的 CNDTR 寄存器的值与 TIMx\_DCR 寄存器中 DBL 的值对应起来，否则该将不能正常使用。

## 25.8. TIM16\_17寄存器描述

0x4001 4800 - 0x4001 4BFF TIM17

0x4001 4400 - 0x4001 47FF TIM16

0x4001 4000 - 0x4001 43FF TIM15

## 25.8.1. TIM16\_17 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9: 8	CKD[1: 0]	RW	00	时钟分频因子 这2位定义在定时器时钟 (CK_INT) 频率, 死区时间和由死区发生器与数字滤波器 (ETR,Tix) 所用的采样时钟之间的分频比例 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: 保留, 不要使用这个配置
7	ARPE	RW	0	自动重载预装载允许位 0: TIMx_ARR 寄存器没有缓冲 1: TIMx_ARR 寄存器被装入缓冲器
6: 4	保留	-	-	保留
3	OPM	RW	0	单脉冲模式 0: 在发生更新事件时, 计数器不停止 1: 在发生下一次更新事件 (清除 CEN 位) 时, 计数器停止。
2	URS	RW	0	更新请求源 软件通过该位选择 UEV 事件的源 0: 如果允许产生更新中断或 DMA 请求, 则下述任一事件产生一个更新中断或 DMA 请求: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 1: 如果允许产生更新中断或 DMA 请求, 则只有计数器溢出/下溢产生一个更新中断或 DMA 请求
1	UDIS	RW	0	禁止更新 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新 (UEV) 事件由下述任一事件产生:

Bit	Name	R/W	Reset Value	Function
				<ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> 被缓存的寄存器被装入它们的预装载值。 1: 禁止 UEV。不产生更新事件，影子寄存器 (ARR,PSC,CCR <sub>x</sub> ) 保持它们的值。 如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	CEN	RW	0	允许计数器 0: 禁止计数器 1: 开启计数器 注: 在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。

### 25.8.2. TIM16\_17 控制寄存器 2 (TIMx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						OIS1N	OIS1	Res				CCDS	CCUS	Res	CCPC
-						RW	RW	-				Rw	RW	-	RW

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9	OIS1N	RW	0	输出空闲状态1 (OC1N 输出)。 0: 当 MOE=0时，死区后 OC1N=0 1: 当 MOE=0时，死区后 OC1N=1 注: 已经设置了 LOCK (TIMx_BDTR 寄存器) 级别1、2或3后，该位不能被修改。
8	OIS1	RW	0	输出空闲状态1 (OC1输出)。 0: 当 MOE=0时，如果实现了 OC1N，则死区后 OC1=0 1: 当 MOE=0时，如果实现了 OC1N，则死区后 OC1=1 注: 已经设置了 LOCK (TIMx_BDTR 寄存器) 级别1、2或3后，该位不能被修改。
7: 4	保留	-	-	保留
3	CCDS	RW	0	捕获/比较的 DMA 选择 0: 当发生 CCx 事件时，送出 CCx 的 DMA 请求。 1: 当发生更新事件时，送出 CCx 的 DMA 请求。
2	CCUS	RW	0	捕获/比较控制更新选择

Bit	Name	R/W	Reset Value	Function
				0: 如果捕获/比较控制位是预装载的 (CCPC=1), 只能通过设置 COM 位更新它们。 1: 如果捕获/比较控制位是预装载的 (CCPC=1), 可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注: 该位只对具有互补输出的通道起作用。
1	保留	-	-	保留
0	CCPC	RW	0	捕获/比较预装载控制位 0: CCxE, CCxNE 和 OCxM 位不是预装载的。 1: CCxE, CCxNE 和 OCxM 位是预装载的; 设置该位后, 它们只在设置了 COM 位后被更新。 注: 该位只对具有互补输出的通道起作用。

### 25.8.3. TIM16\_17 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1DE	UDE	BIE	Res	COMIE	Res	Res	Res	CC1IE	UIE
-						RW	RW	RW	-	RW	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9	CC1DE	RW	0	CC1DE: 允许捕获/比较1的 DMA 请求 0: 禁止捕获/比较1的 DMA 请求 1: 允许捕获/比较1的 DMA 请求
8	UDE	RW	0	UDE: 允许更新的 DMA 请求 0: 禁止更新的 DMA 请求 1: 允许更新的 DMA 请求
7	BIE	RW	0	BIE: 允许刹车中断 0: 禁止刹车中断 1: 允许刹车中断
6	保留	-	-	保留
5	COMIE	RW	0	COMIE: 允许 COM 中断 0: 禁止 COM 中断 1: 允许 COM 中断
4: 2	保留	-	-	保留
1	CC1IE	RW	0	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断

				1: 允许捕获/比较1中断
0	UIE	RW	0	UIE: 允许更新中断 0: 禁止更新中断 1: 允许更新中断

#### 25.8.4. TIM16\_17 状态寄存器 (TIMx\_SR)

Address offset: 0x010

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res	BIF	Res	COMIF	Res			CC1F	UIF
-						RC_W0	-	RC_W0	-	RC_W0	-			RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9	CC1OF	RC_W0	0	捕获/比较1 过捕获标记 仅当相应的通道被配置为输入捕获时, 该标记可由硬件置1。写0可清除该位。 0: 无过捕获产生; 1: CC1OF 置1时, 计数器的值已经被捕获到TIMx_CCR1寄存器。
8	保留	-	-	保留
7	BIF	RC_W0	0	刹车中断标记 一旦刹车输入有效, 由硬件对该位置1。如果刹车输入无效, 则该位可由软件清0。 0: 无刹车事件产生; 1: 刹车输入上检测到有效电平。
6	保留	-	-	保留
5	COMIF	RC_W0	0	COM 中断标记 一旦产生 COM 事件 (当 CCxE、CCxNE、OCxM 已被更新) 该位由硬件置1。它由软件清0。 0: 无 COM 事件产生; 1: COM 中断等待响应
4: 2	保留	-	-	保留
1	CC1F	RC_W0	0	捕获/比较1 中断标记 如果通道 CC1 配置为输出模式: 当计数器值与比较值匹配时该位由硬件置1, 但在中心对称模式下除外 (参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清0。 0: 无匹配发生; 1: TIMx_CNT 的值与 TIMx_CCR1 的值匹配。

Bit	Name	R/W	Reset Value	Function
				如果通道 CC1配置为输入模式： 当捕获事件发生时该位由硬件置1，它由软件清0或通过读 TIMx_CCR1清0。 0: 无输入捕获产生； 1: 输入捕获产生并且计数器值已装入 TIMx_CCR1（在 IC1上检测到与所选极性相同的边沿）。
0	UIF	RC_W0	0	更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0: 无更新事件产生； 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1： - 若 TIMx_CR1寄存器的 UDIS=0，当 REP_CNT=0时产生更新事件（计数器上溢时）； - 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1时产生更新事件（软件对 CNT 重新初始化）； - 若 TIMx_CR1寄存器的 UDIS=0、URS=0，当 CNT 被触发事件重初始化时产生更新事件。（参考从模式控制寄存器（TIMx_SMCR））

### 25.8.5. TIM16\_17 事件产生寄存器 (TIMx\_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	Res	COMG	Res	Res	Res	CC1G	UG
-	-	-	--	-	-	-	-	W	-	W	-	-	-	W	W

Bit	Name	R/W	Reset Value	Function
15: 8	Res	-	-	保留
7	BG	W	0	产生刹车事件 该位由软件置1，用于产生一个刹车事件，由硬件自动清0。 0: 无动作； 1: 产生一个刹车事件。此时 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
6	Res	-	-	保留
5	COMG	W	0	捕获/比较事件，产生控制更新 该位由软件置1，由硬件自动清0。



Bit	Name	R/W	Reset Value	Function
				0: 无动作; 1: 当 CCPC=1, 允许更新 CCxE、CCxNE、OCxM 位。 注: 该位只对有互补输出的通道有效。
4: 2	Res	-	-	保留
1	CC1G	W	0	产生捕获/比较1事件 该位由软件置1, 用于产生一个捕获/比较事件, 由硬件自动清0。 0: 无动作; 1: 在通道 CC1上产生一个捕获/比较事件: 若通道 CC1配置为输出: 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。 若通道 CC1配置为输入: 当前的计数器值捕获至 TIMx_CCR1寄存器, 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。若 CC1IF 已经为1, 则设置 CC1OF=1。
0	UG	W	0	产生更新事件。该位由软件置1, 硬件自动清0。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意: 预分频器的计数器也被清0 (但是预分频系数不变)。若在中心对称模式下或 DIR=0 (向上计数) 则计数器被清0, 若 DIR=1 (向下计数) 则计数器装载 TIMx_ARR 的值。

### 25.8.6. TIM16\_17 捕获/比较模式寄存器 1 (TIMx\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						Res	OC1M[2: 0]				OC1PE	OC1FE	CC1S[1: 0]		
Res							IC1F[3: 0]				IC1PSC[1: 0]		0]		
-						-	RW	RW	RW	RW	RW	RW	RW	RW	RW

输出比较模式:

Bit	Name	R/W	Reset Value	Function
15: 7	保留	-	-	保留
6: 4	OC1M[2: 0]	RW	00	输出比较1模式 该3位定义了输出参考信号 OC1REF 的动作, 而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效, 而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。

Bit	Name	R/W	Reset Value	Function
				<p>000: 冻结。输出比较寄存器 TIMx_CCR1与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用;</p> <p>001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器1 (TIMx_CCR1) 相同时, 强制 OC1REF 为高。</p> <p>010: 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获 / 比较寄存器 1 (TIMx_CCR1) 相同时, 强制 OC1REF 为低。</p> <p>011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。</p> <p>100: 强制为无效电平。强制 OC1REF 为低。</p> <p>101: 强制为有效电平。强制 OC1REF 为高。</p> <p>110: PWM 模式1 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为有效电平, 否则为无效电平。</p> <p>111: PWM 模式2 - 在向上计数时, 一旦 TIMx_CNT&lt;TIMx_CCR1时通道1为无效电平, 否则为有效电平。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 在 PWM 模式1或 PWM 模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时, OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较1预装载使能</p> <p>0: 禁止 TIMx_CCR1寄存器的预装载功能, 可随时写入 TIMx_CCR1寄存器, 且新值马上起作用。</p> <p>1: 开启 TIMx_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIMx_CCR1的预装载值在更新事件到来时被载入当前寄存器中。</p> <p>注1: 一旦 LOCK 级别设为3 (TIMx_BDTR 寄存器中的 LOCK 位) 并且 CC1S=00 (该通道配置成输出) 则该位不能被修改。</p> <p>注2: 仅在单脉冲模式下, 可以在未确认预装载寄存器情况下使用 PWM 模式, 否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较1 快速使能</p> <p>该位用于加快 CC 输出对触发器输入事件的响应。</p>

Bit	Name	R/W	Reset Value	Function
				<p>0: 根据计数器与 CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活 CC1输出的最小延时为5个时钟周期。</p> <p>1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1输出间的延时被缩短为3个时钟周期。</p> <p>OCFE 的只在通道被配置成 PWM1或 PWM2模式时起作用。</p>
1: 0	CC1S[1: 0]	RW	00	<p>捕获/比较1 选择。</p> <p>这2位定义通道的方向(输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC1映射在 TI1上;</p> <p>10: CC1通道被配置为输入, IC1映射在 TI2上;</p> <p>11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。</p> <p>注: CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0) 才是可写的。</p>

**输入捕获模式:**

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7: 4	IC1F[3: 0]	RW	0000	<p>输入捕获1滤波器</p> <p>这几位定义了 TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成, 它记录到 N 个事件后会产生一个输出的跳变:</p> <p>0000: 没有滤波器, 在 fDTS 下采样</p> <p>0001: fSAMPLING=fCK_INT, N=2</p> <p>0010: fSAMPLING=fCK_INT, N=4</p> <p>0011: fSAMPLING=fCK_INT, N=8</p> <p>0100: fSAMPLING=fDTS/2, N=6</p> <p>0101: fSAMPLING=fDTS/2, N=8</p> <p>0110: fSAMPLING=fDTS/4, N=6</p> <p>0111: fSAMPLING=fDTS/4, N=8</p> <p>1000: fSAMPLING=fDTS/8, N=6</p> <p>1001: fSAMPLING=fDTS/8, N=8</p> <p>1010: fSAMPLING=fDTS/16, N=5</p> <p>1011: fSAMPLING=fDTS/16, N=6</p>

Bit	Name	R/W	Reset Value	Function
				1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	捕获/比较1预分频器 00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获; 01: 每2个事件触发一次捕获; 10: 每4个事件触发一次捕获; 11: 每8个事件触发一次捕获。
1: 0	CC1S[1: 0]	RW	00	CC1S[1: 0]: 捕获/比较1选择。 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC1映射在 TI1上; 10: CC1通道被配置为输入, IC1映射在 TI2上; 11: CC1通道被配置为输入, IC1映射在 TRC 上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的 TS 位选择)。

### 25.8.7. TIM16\_17 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

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	CC1NP	CC1NE	CC1P	CC1E
-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 4	保留	-	-	保留
3	CC1NP	RW	0	捕获/比较1互补输出极性 0: OC1N 高电平有效 1: OC1N 低电平有效 注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LCCK 位) 设为3或2且 CC1S=00 (通道配置为输出) 则该位不能被修改。
2	CC1NE	RW	0	捕获/比较1互补输出使能 0: OC1N 禁止输出 1: OC1N 信号输出到对应的输出引脚

Bit	Name	R/W	Reset Value	Function
				当 CC1通道配置为输出时, OC1N 输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定, 见下表
1	CC1P	RW	0	<p>捕获/比较1输出极性</p> <p>CC1通道配置为输出:</p> <p>0: OC1高电平有效</p> <p>1: OC1低电平有效</p> <p>CC1通道配置为输入:</p> <p>CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1和 TI2FP1的极性。</p> <p>00: 不反相/上升沿:</p> <p>TIxFP1上升沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ;</p> <p>TIxFP1不反相 (门控模式、编码器模式) 。</p> <p>01: 反相/下降沿:</p> <p>TIxFP1下降沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ;</p> <p>TIxFP1反相 (门控模式、编码器模式) 。</p> <p>10: 保留, 不要使用这个配置。</p> <p>11: 不反相/双沿</p> <p>TIxFP1上升和下降沿都有效 (捕获、复位模式下触发、外部时钟或触发模式下) ;</p> <p>TIxFP1不反相 (门控模式) 。这个配置不能应用于编码器模式下。</p> <p>注:</p> <p>一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为 3或2, 则该位不能被修改</p>
0	CC1E	RW	0	<p>捕获/比较1输出使能</p> <p>0: 捕获模式关闭/OC1禁止输出</p> <p>1: 捕获模式开启/OC1信号输出到对应的输出引脚</p> <p>当 CC1通道配置为输出时, OC1输出电平由 MOE、OSSI、OSSR、OIS1、OIS1N、CC1E 和 CC1NE 位共同决定, 见下表</p> <p>注:</p> <p>对于互补输出通道, 这一位是预载的。如果 TIMx_CR2寄存器中的 CCPC 位被设置, 那么 CC1E 的实际有效位只有在 com 事件发生时才会加载预载值。</p>

表 具有中断功能的互补 OCx 和 OCxN 通道的输出控制

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	X	0	0	0	输出禁止 (与定时器断开), OCx=0, OCx_EN=0	输出禁止 (与定时器断开), OCxN=0, OCxN_EN=0
		0	0	1	输出禁止 (与定时器断开), OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF 异或 CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCREF 异或 CCxP, OCx_EN=1	输出禁止 (与定时器断开), OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	OCREF 的互补 (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	输出禁止 (与定时器断开), OCx=CCxP, OCx_EN=0	输出禁止 (与定时器断开), OCxN=CCxNP, OCxN_EN=0
		1	0	1	输出禁止 (与定时器断开), OCx=CCxP, OCx_EN=1	OCxREF+Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF+Polarity OCx=OCxREF xor CCxP, OCx_EN=1	关闭状态 (输出使能且为 无效电平), OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF+Polarity + dead-time OCx_EN=1	OCREF 的互补 (not OCREF) + polarity + dead-time OCN_EN=1
0	X	0	0	0	输出禁止 (与定时器断开)	
		0	0	1		
		0	1	0		
		0	1	1		
		1	0	0	关闭状态 (输出使能且为无效电平) 异步的: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 若时钟存在: 经过一个死区时间后, 假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平, OCx=OISx 和 OCxN=OISxN	
		1	0	1		
		1	1	0		
		1	1	1		

如果一个通道的2个输出都没有使用 (CCxE = CCxNE = 0), 那么 OISx, OISxN, CCxP 和 CCxNP 都必须清零。

注: 引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态, 取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

### 25.8.8. TIM16\_17 计数器 (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CNT[15: 0]	RW	0	计数器的值

### 25.8.9. TIM16\_17 预分频器 (TIMx\_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	PSC[15: 0]	RW	0	预分频器的值 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 PSC 包含了当更新事件产生时装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清0或被工作在复位模式的从控制器清0。

### 25.8.10. TIM16\_17 自动重载寄存器 (TIMx\_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
15: 0	ARR[15: 0]	RW	FFFF	自动重载的值 ARR 包含了将要装载入实际的自动重载寄存器的值。 当自动重载的值为空时，计数器不工作。

### 25.8.11. TIM16\_17 重复计数器寄存器 (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7: 0	REP[7: 0]	RW	0	<p>周期计数器的值</p> <p>开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）；如允许产生更新中断，则会同时影响产生更新中断的速率。</p> <p>每次向下计数 REP_CNT 达到0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件U_RC发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。</p> <p>这意味着在 PWM 模式中，(REP+1) 对应着：</p> <ul style="list-style-type: none"> <li>- 在边沿对齐模式下，PWM 周期的数目；</li> </ul>

### 25.8.12. TIM16\_17 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15: 0	CCR1[15: 0]	RW/RO	0	<p>捕获/比较1的值</p> <p>若 CC1通道配置为输出： CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。</p> <p>如果在 TIMx_CCMR1寄存器（OC1PE 位）中未选择预装载特性，其始终装入当前寄存器中。</p> <p>否则，只有当更新事件发生时，此预装载值才装入当前捕获/比较1寄存器中。</p>



				<p>当前捕获/比较寄存器包含了与计数器 TIMx_CNT 比较的值，并且在 OC1端口上输出信号。</p> <p>若 CC1通道配置为输入： CCR1包含了由上一次输入捕获1事件 (IC1) 传输的计数器值。</p>
--	--	--	--	---

### 25.8.13. TIM16\_17 刹车和死区寄存器 (TIMx\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7: 0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

注释：根据锁定设置，AOE、BKP、BKE、OSSI、OSSR 和 DTG[7: 0]位均可被写保护，有必要在第一次写入

TIMx\_BDTR 寄存器时对它们进行配置。

Bit	Name	R/W	Reset Value	Function
15	MOE	RW	0	<p>主输出使能</p> <p>一旦刹车输入有效，该位被硬件异步清0。根据 AOE 位的值，可由软件清0或自动置1。它仅对配置为输出通道有效。</p> <p>0: 禁止 OC 和 OCN 输出或强制为空闲状态； 1: 如果设置了相应的使能位 (TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出。</p>
14	AOE	RW	0	<p>自动输出使能</p> <p>0: MOE 只能被软件置1； 1: MOE 能被软件置1或在下一个更新事件自动置1 (如果刹车输入无效)。</p> <p>注：一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1，则该位不能被修改。</p>
13	BKP	RW	0	<p>刹车输入极性</p> <p>0: 刹车输入低电平有效； 1: 刹车输入高电平有效。</p> <p>注：一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1，则该位不能被修改。</p> <p>任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。</p>
12	BKE	RW	0	<p>刹车功能使能</p> <p>0: 禁止刹车输入； 1: 开启刹车输入。</p>

Bit	Name	R/W	Reset Value	Function
				注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为1，则该位不能被修改。 任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
11	OSSR	RW	0	运行模式下“关闭状态”选择 该位用于当 MOE=1且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器（TIMx_CCER））。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1：当定时器不工作时，一旦 CCxE=1或 CCxNE=1，开启 OC/OCN 输出并输出无效电平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下“关闭状态”选择 该位用于当 MOE=0且通道设为输出时。 参考 OC/OCN 使能的详细说明（捕获/比较使能寄存器（TIMx_CCER））。 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）； 1：当定时器不工作时，一旦 CCxE=1或 CCxNE=1，OC/OCN 首先输出其空闲电平。 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为2，则该位不能被修改。
9: 8	LOCK[1: 0]	RW	00	锁定设置 该位为防止软件错误而提供写保护。 00：锁定关闭，寄存器无写保护； 01：锁定级别1，不能写入 TIMx_BDTR 寄存器的 DTG/BKE/BKP/AOE 位、TIMx_CR2寄存器的 OISx/OISxN 位； 10：锁定级别2，不能写入锁定级别1中的各位，也不能写入 CC 极性位（一旦相关通道通过 CCxS 位设为输出，TIMx_CCER 寄存器的 CCxP/CCNxP 位）以及 OSSR/OSSI 位；

Bit	Name	R/W	Reset Value	Function
				11: 锁定级别3, 不能写入锁定级别2中的各位, 也不能写入 CC 控制位 (一旦相关通道通过 CCxS 位设为输出, TIMx_CCMRx 寄存器的 OCxM/OCxPE 位); 注: 在系统复位后, 只能写一次 LOCK 位, 一旦写入 TIMx_BDTR 寄存器, 则其内容冻结直至复位。
7: 0	DTG[7: 0]	RW	0000 0000	<p>死区发生器设置</p> <p>这些位定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间:</p> <p>DTG[7: 5]=0xx =&gt; DT=DTG[7: 0] × Tdtg, Tdtg = TDTS;</p> <p>DTG[7: 5]=10x =&gt; DT= (64+DTG[5: 0]) × Tdtg, Tdtg = 2 × TDTS;</p> <p>DTG[7: 5]=110 =&gt; DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 8 × TDTS;</p> <p>DTG[7: 5]=111 =&gt; DT= (32+DTG[4: 0]) × Tdtg, Tdtg = 16 × TDTS;</p> <p>例: 若 TDTS = 125ns (8MHZ), 可能的死区时间为: 0到15875ns, 若步长时间为125ns; 16us 到31750ns, 若步长时间为250ns; 32us 到63us, 若步长时间为1us; 64us 到126us, 若步长时间为2us;</p> <p>注: 一旦 LOCK 级别 (TIMx_BDTR 寄存器中的 LOCK 位) 设为1、2或3, 则这些位不能被修改。</p>

#### 25.8.14. TIM16\_17 DMA 控制寄存器 (TIMx\_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4: 0]					Res			DBA[4: 0]				
-	-	-	RW	RW	RW	RW	RW	-	-	-	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	保留	-	-	保留
12: 8	DBL[4: 0]	RW	0 0000	DMA 连续传送长度 这些位定义了 DMA 在连续模式下的传送长度 (当对 TIMx_DMAR 寄存器的地址进行读或写

Bit	Name	R/W	Reset Value	Function
				<p>时, 定时器则进行一次连续传送), 即: 定义被传送的字节数目:</p> <p>00000: 1次传输  00001: 2次传输  00010: 3次传输  .....  .....  10001: 18次传输</p> <p>例: 我们考虑这样的传输: DBL=7, DBA=TIM2_CR1</p> <p>- 如果 DBL=7, DBA=TIM2_CR1表示待传输数据的地址, 那么传输的地址由下式给出:</p> <p>(TIMx_CR1的地址) + DBA + (DMA 索引), 其中  DMA 索引 = DBL</p> <p>其中 (TIMx_CR1的地址) + DBA 再加上7, 给出了将要写入或者读出数据的地址, 这样数据的传输将发生在从地址 (TIMx_CR1的地址) + DBA 开始的7个寄存器。</p> <p>根据 DMA 数据长度的设置, 可能发生以下情况:</p> <p>- 如果设置数据为半字 (16位), 那么数据就会传输给全部7个寄存器。</p> <p>- 如果设置数据为字节, 数据仍然会传输给全部7个寄存器: 第一个寄存器包含第一个 MSB 字节, 第二个寄存器包含第一个 LSB 字节, 以此类推。因此对于定时器, 用户必须指定由 DMA 传输的数据宽度。</p>
7: 5	保留	-	-	保留
4: 0	DBA[4: 0]	RW	0 0000	<p>DBA[4: 0]: DMA 基地址</p> <p>这些位定义了 DMA 在连续模式下的基地址 (当对 TIMx_DMAR 寄存器的地址进行读或写时), DBA 定义为从 TIMx_CR1寄存器所在地址开始的偏移量:</p> <p>00000: TIMx_CR1,  00001: TIMx_CR2,  00010: TIMx_SMCR,  .....</p>

### 25.8.15. TIM16\_17 连续模式的 DMA 地址 (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DMAB[31: 0]	RW	0	DMA 连续传送寄存器 对 TIMx_DMAR 寄存器的读或写会导致对以下地址的寄存器的存取操作： $TIMx\_CR1地址 + (DBA + DMA 指针) \times 4$ ，其中： “TIMx_CR1地址” 是控制寄存器1的地址； “DBA” 是 TIMx_DCR 寄存器中定义的基地址； “DMA 指针” 是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

注：在使用 DMA 连续传输功能时，必须将 DMA 中对应通道的 CNDTR 寄存器的值与 TIMx\_DCR 寄存器中 DBL 的值对应起来，否则该将不能正常使用。

## 26. 低功耗定时器 (LPTIM)

### 26.1. 简介

LPTIM 是一款 16 位定时器。LPTIM 将系统从低功耗模式中唤醒的能力使得它适合于实现低功耗应用。

LPTIM 可以使用高频时钟 (PCLK) 和低频时钟 (LSI/LSE) 计数, 可提供所需的功能和性能, 同时将功耗降至最低。

### 26.2. LPTIM 主要特性

- 16 位向上计数器
- 3 位预分频器, 具有 8 个可能的分频因子 (1、2、4、8、16、32、64、128)
- 可选时钟
  - 内部时钟源: LSE, LSI 或 APB 时钟 (PCLK)
- 16 位 ARR 可重载寄存器
- 连续/单次模式

### 26.3. 低功耗定时器 (LPTIM) 功能描述

#### 26.3.1. LPTIM 框图

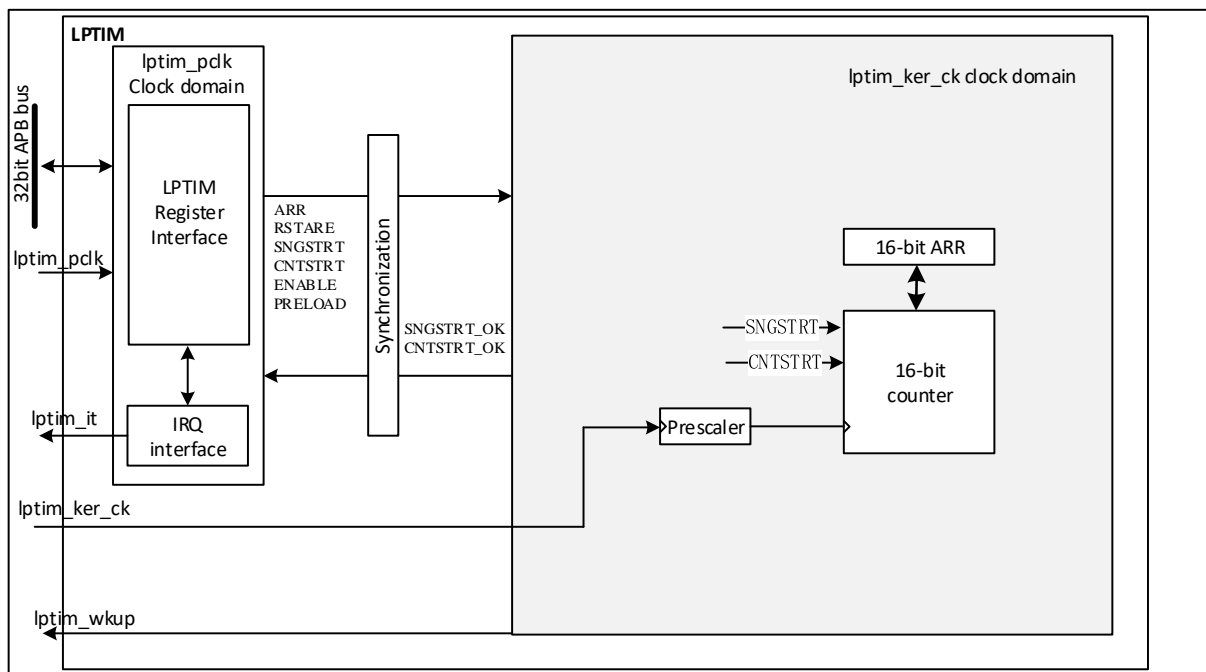


图 26-1 低功耗定时器框图

### 26.3.2. LPTIM 引脚和内部信号

名字	信号类型	说明
lptim_pclk	输入	LPTIM APB 时钟
lptim_ker_ck	输入	LPTIM 内部时钟源
lptim_it	输出	LPTIM 全局中断
lptim_wkup	输出	LPTIM 唤醒事件

### 26.3.3. LPTIM 复位和时钟

LPTIM 可以使用多个时钟源进行计时。

通过 RCC 模块，可以使用内部时钟信号对其进行时钟控制（该时钟信号可以在 PCLK、LSI、LSE 源中进行选择）。

### 26.3.4. 预分频器

LPTIM 16 位计数器，由一个可配置的 2 次方预分频器控制驱动。预分频器分频比由 PRESC[2: 0] 控制。

下表列出了所有情况：

表 26-26 预分频系数

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

### 26.3.5. 工作模式

LPTIM 具有两种如下工作模式。

- **单次模式：** 定时器从一个触发事件（写 SNGSTRT 寄存器）开始，当达到 ARR 值时停止。

要使能单次计数，SNGSTRT 位必须置 1。

一个新的触发事件将重新启动计时器。在计数器启动之后，并到达 ARR 之前的任何触发事件都将被忽略。

- **连续模式：** 计时器自由运行，从触发事件（写 CNTSTRT 寄存器）开始运行，直到计时器被禁止才停止。

要使能连续计数，LPTIM\_CR.CNTSTRT 位必须置 1。

设置 LPTIM\_CR.CNTSTRT 将启动计数器进行连续计数。

可以实时从单次模式转变为连续模式：

- 如果之前选择了连续模式，则置位 LPTIM\_CR.SNGSTRT 会将 LPTIM 切换到单次模式。计数器（如果激活）将在达到 ARR 后立即停止。
- 如果先前选择了单次模式，则置位 LPTIM\_CR.CNTSTRT 会将 LPTIM 切换到连续模式。计数器（如果激活）一到达 ARR 就会重新启动。

### 26.3.6. 寄存器更新

PRELOAD 位控制 LPTIM\_ARR 寄存器的更新方式：

- 当 PRELOAD 位被复位为“0”：LPTIM\_ARR 寄存器在任何写访问后立即更新。
- 当 PRELOAD 位被设为“1”时：如果定时器已经启动，则 LPTIM\_ARR 将在下一个 LPTIM 更新事件同步进行更新。

LPTIM APB 接口和 LPTIM Kernel 逻辑使用不同的时钟，因此在 APB 写入,和写入的值被应用到计数器比较器时，存在一定的延迟。在此延迟周期内，必须避免对这些寄存器进行任何额外的写操作。

LPTIM\_ISR 寄存器中的 ARROK 标志，指示对 LPTIM\_ARR 寄存器的写操作是否完成。

对 LPTIM\_ARR 寄存器进行写操作后，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 标志位置位之前的任何连续写入都将导致不可预测的结果。

### 26.3.7. 计数器模式

LPTIM 仅支持内部时钟计数模式。

LPTIM\_CR 寄存器中的 ENABLE 位用于使能/不使能 LPTIM 内核逻辑。置位 ENABLE 位后，硬件会延迟 3 个计数器时钟才使能 LPTIM。

仅当 LPTIM 禁用时，才能修改 LPTIM\_CFGR 和 LPTIM\_IER 寄存器。

### 26.3.8. 计数器复位

为了将 LPTIM\_CNT 寄存器的内容复位，提供复位机制：

#### 同步复位机制：

同步复位由 LPTIM\_CR.CONURST 位控制。将 COUNTRST 位置 1 后，复位信号送给 LPTIM 的 Kernel 时钟域。所以需要注意在复位起作用前，LPTIM Kernel 时钟域的逻辑电路已过去几个时钟周期了。这将使从复位触发到复位生效，LPTIM 计数器多计了额外几个数。

由于 COUNTRST 是 APB 时钟域的，而 LPTIM 计数器位于 LPTIM Kernel 时钟域，所以当向 COUNTRST 位写入 1 时，需要 Kernel 时钟的 3 个时钟周期的延迟来同步来自 APB 时钟域的复位信号。

#### 异步复位机制：

异步复位由位于 LPTIM\_CR 寄存器中的 RSTARE 位控制。当该位置 1 时，对 LPTIM\_CNT 寄存器的任何读访问都会将其内容复位为零。应在不提供 LPTIM 内核时钟的时间范围内触发异步复位。

应注意的是，为了实现可靠的 LPTIM\_CNT 寄存器内容读取，必须执行两次连续的读访问并进行比较。当两次读访问的值相等时，可认为读访问可靠。然而，当使能了异步复位时，不可能两次读取 LPTIM\_CNT 寄存器。

### 26.3.9. 调试模式 (debug mode)



当芯片进入调试模式，LPTIM 计数器会根据 DBG 模块中的 DBG\_LPTIM\_STOP 配置位选择继续正常工作或者停止工作。

## 26.4. LPTIM 低功耗模式

表 26-26 LPTIM 不同低功耗模式的区别

模式	描述
Sleep	功能无影响。 LPTIM 中断（使能后）会退出 sleep 模式。
Stop	LSE/LSI 时钟存在时功能无影响。 LPTIM 中断（使能后）会退出 stop 模式。

## 26.5. LPTIM 中断

如果下列事件在 LPTIM\_IER 寄存器内使能，则这些事件将生成中断/唤醒事件：

- 自动重新加载匹配

注意：如果在 LPTIM\_ISR 寄存器（状态寄存器）中的相应标志置 1 后，LPTIM\_IER 寄存器（中断使能寄存器）中的相应位被置 1，则不产生中断。

表 26-26 LPTIM 中断事件

中断事件	描述
自动重载匹配	当计数器寄存器的内容（LPTIM_CNT）与自动重新加载寄存器的内容匹配（LPTIM_ARR），中断标志置位
自动重载寄存器更新 OK	当对 LPTIM_ARR 寄存器的写操作完成，中断标志被置位

## 26.6. LPTIM 寄存器

### 26.6.1. LPTIM 中断和状态寄存器（LPTIM\_ISR）

地址偏移：0x000

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	ARROK	Res	Res	ARRM	Res
-	-	-	-	-	-	-	-	-	-	-	R	-	-	R	-

Bit	Name	R/W	Reset Value	Function
31: 5	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
4	ARROK	R	0	自动重载寄存器更新 OK。 ARROK 由硬件设置，以通知应用程序 APB 总线对 LPTIM_ARR 的写操作已成功完成。向 LPTIM_ICR.ARROKCF 写入1可清除 ARROK 标志。
3: 2	保留	-	-	保留
1	ARRM	R	0	自动重载匹配 ARRM 由硬件设置，通知应用程序 LPTIM_CNT 寄存器值匹配 LPTIM_ARR 寄存器的值。向 LPTIM_ICR 寄存器的 ARRMCF 位写入1可清除 ARRM 标志
0	保留	-	-	保留

### 26.6.2. LPTIM 中断清除寄存器 (LPTIM\_ICR)

地址偏移: 0x004

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	ARROKCF	Res	Res	ARRMCF	Res
-	-	-	-	-	-	-	-	-	-	-	RW	-	-	RW	-

Bit	Name	R/W	Reset Value	Function
31: 5	保留	-	-	保留
4	ARROKCF	W	0	自动重载寄存器更新 OK 清除标志 向该位写入1可清除 LPTIM_ISR 寄存器中的 ARROK 标志
3: 2	保留	-	-	保留
1	ARRMCF	W	0	自动重载匹配清除标志 向该位写入1可清除 LPTIM_ISR 寄存器中的 ARRM 标志
0	保留	-	-	保留

### 26.6.3. LPTIM 中断使能寄存器 (LPTIM\_IER)

地址偏移: 0x008

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	ARROKIE	Res	Res	ARRMIE	Res

-	-	-	-	-	-	-	-	-	-	-	RW	-	-	RW	-
---	---	---	---	---	---	---	---	---	---	---	----	---	---	----	---

Bit	Name	R/W	Reset Value	Function
31: 5	保留	-	-	保留
4	ARROKIE	RW	0	自动重载寄存器更新 OK 中断使能。 0: ARROK 中断禁用 1: ARROK 中断使能
3: 2	保留	-	-	保留
1	ARRMIE	RW	0	自动重载匹配中断使能 0: ARRM 中断禁用 1: ARRM 中断使能
0	保留	-	-	保留

#### 26.6.4. LPTIM 配置寄存器 (LPTIM\_CFGR)

地址偏移: 0x00C

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	PRE-LOAD	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	RW	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	PRESC[2: 0]			Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	RW	RW	RW	-	-	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 23	保留	-	-	保留
22	PRELOAD	RW	0	寄存器更新模式 预加载位控制 LPTIM_ARR 寄存器更新模式 0: 每次 APB 总线写访问后更新寄存器 1: 寄存器在当前 LPTIM 周期结束时更新
21: 12	保留	-	-	保留
11: 9	PRESC[2: 0]	RW	0	时钟预分频器 PRESC 位配置预分频器分频系数。它可以是下列分频中的一个因素: 000: /1 001: /2 010: /4 011: /8 100: /16

				101: /32 110: /64 111: /128
8: 0	保留	-	-	保留

### 26.6.5. LPTIM 控制寄存器 (LPTIM\_CR)

地址偏移: 0x010

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	R STARE	COUNT RST	CNT STRT	SNG STRT	EN ABLE
-	-	-	-	-	-	-	-	-	-	-	RW	RS	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 5	保留	-	-	保留
4	RSTARE	RW	0	读取后复位使能 此位由软件置1和清0。当 RSTARE 设置为“1”时，对 LPTIM_CNT 的任何读取访问寄存器将异步重置 LPTIM_CNT 寄存器内容。
3	COUNTRST	RS	0	计数器复位。 该位由软件置1，硬件清0。设置为“1”时，此位将触发 LPTIM_CNT 计数寄存器同步复位。由于此复位的同步特性，它只需要在同步延迟3个 LPTIM 内核时钟周期之后释放（LPTIM 内核时钟可能是与 APB 时钟不同）。 注：在 COUNTRST 已被硬件清除为“0”之前，软件绝不能将其设置为“1”。因此，软件在尝试将其设置为“1”之前，应检查 COUNTRST 位是否已清零为“0”
2	CNTSTRT	RW	0	定时器启动连续模式。 该位由软件置位，该位置1将启动连续模式下的 LPTIM。 如果在进行单次计数模式计数时该位被置1，则定时器不会在下一个 LPTIM_ARR 和 LPTIM_CNT 寄存器匹配的脉冲模式计数时停止。LPTIM 计数器保持在连续模式下计数。 注：仅当 LPTIM 使能时，此位才能置1。它将由硬件自动重置。
1	SNGSTRT	RW	0	LPTIM 启动单次模式。 该位由软件置位，由硬件清零。该位置1将以单脉冲模式启动 LPTIM。

Bit	Name	R/W	Reset Value	Function
				注: 仅当 LPTIM 使能时, 此位才能置1。它将由硬件自动复位。
0	ENABLE	RW	0	LPTIM 使能位,由软件设置和清零 0: LPTIM 禁用 1: LPTIM 使能

### 26.6.6. LPTIM 自动重载寄存器 (LPTIM\_ARR)

地址偏移: 0x018

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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ARR	RW	0x0001	自动重新加载值 ARR 是 LPTIM 的自动重载值 当 LPTIM 使能后才能更新该寄存器

### 26.6.7. LPTIM 计数寄存器 (LPTIM\_CNT)

地址偏移: 0x01C

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	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	CNT	R	0	计数器值 当 LPTIM 以异步时钟运行时, 读取 LPTIM_CNT 寄存器可能返回不可靠的值。因此在这种情况下, 有必要执行两次

				连续的读访问并验证返回的两个值是否相同。当两次连续读取访问的值相等时，可以认为读取访问是可靠的。
--	--	--	--	--

## 27. 独立看门狗 (IWDG)

### 27.1. 简介

芯片内集成了一个独立看门狗定时器 (Independent watchdog) (简称 IWDG)，该模块具有高安全级别、时序精确及灵活使用的特点。IWDG 发现并解决由于软件失效造成的功能混乱，并在计数器达到指定的超时值时触发系统复位。

IWDG 由 LSI 提供时钟，这样即使系统时钟关闭，也能保持工作。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场景。

### 27.2. IWDG 主要特性

- 自由运行递减计数器
- 时钟由独立的 RC 振荡器提供 (可在 Stop 模式下工作)
- 支持硬件或者软件模式启动 IWDG 后，RCC 模块自动使能 LSI 作为 IWDG 时钟。
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

### 27.3. IWDG 功能描述

#### 27.3.1. IWDG 框图

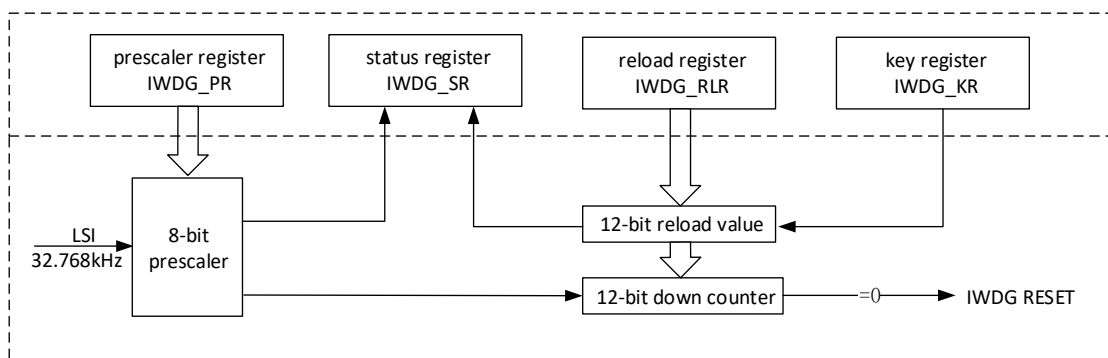


图 27-1 IWDG 框图

注：看门狗功能处于  $V_{DD}$  供电区，即在停机和待机模式时仍能正常工作。

在键寄存器 (IWDG\_KR) 中写入 0xCCCC，开始启用独立看门狗；此时计数器开始从其复位值 0xFFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号 (IWDG\_RESET)。

无论何时，只要键寄存器 IWDG\_KR 中被写入 0xAAAA，IWDG\_RLR 中的值就会被重新加载到计数器中从而避免产生 IWDG 复位。

表 27-1 看门狗超时时间 32.768 kHz 的输入时钟 (LSI)

预分频系数	PR[2: 0]位	最短时间 (ms) RL[11: 0]=0x000	最长时间 (ms) RL[11: 0]=0xFFFF
/4	0	0.125	511.875

/8	1	0.25	1023.75
/16	2	0.5	2047.5
/32	3	1	4095
/64	4	2	8190
/128	5	4	16380
/256	(6或7)	8	32760

注：这些时间是按照32.768 kHz 时钟给出。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。通过对 LSI 进行校准可获得相对精确的看门狗超时时间。

### 27.3.2. 硬件看门狗

如果上电装载的选项字节 (option bytes) 设置了打开硬件看门狗，则 IWDG 上电被自动使能，并且如果在计数器计数到 0 之前，IWDG\_KR 的 KEY 寄存器没被软件改写，则产生复位信号。

### 27.3.3. 寄存器保护

对寄存器 IWDG 预分频、IWDG 重装载的写访问是被保护的。为了修改他们，用户必须先向 IWDG\_KR 的 KEY 寄存器写 0x0000 5555。对这些寄存器的写其他数将破坏时序，如写 0x0000AAAA 加载，寄存器将被再次保护。

如果预分频寄存器、重装载寄存器的值正在更新，状态寄存器是会体现出来的。

### 27.3.4. 调试模式与 STOP 模式

本功能为系统支持调试模式时才存在。

如果 CPU 进入调试模式，IWDG 继续计数还是冻结定时器，取决于 DBGMCU 模块 DBG\_IWDG\_STOP 的配置。

如果 CPU 进入 STOP 低功耗模式，在 Flash 中的选项字节 (Option byte) 中有 IWDG\_STOP 位，IWDG\_STOP 位可以控制 CPU 在进入 STOP 低功耗模式时，IWDG 继续正常计数还是冻结定时器。

选项字节中关于 IWDG\_STOP 的配置如下：

设置 IWDG 在 STOP 模式下定时器运行状态：

0: 冻结定时器

1: 正常运行

## 27.4. IWDG 寄存器

可以用半字 (16 位) 或字 (32 位) 的方式操作这些外设寄存器。

### 27.4.1. 密钥寄存器 (IWDG\_KR)

Address offset: 0x00

Reset value: 0x0000 0000



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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	KEY[15: 0]	W	0x00	<p>Key 值。</p> <p>软件必须以一定的时间间隔向该寄存器写入0xAAAA，否则，当计数器计数到0时，看门狗会产生复位。</p> <p>0x5555：表示允许访问 IWDG_PR、IWDG_RLR 寄存器；</p> <p>0xCCCC：表示启动 IWDG（如果选择了硬件看门狗则不受此命令字限制）。</p>

### 27.4.2. 预分频寄存器 (IWDG\_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 3	保留	-	-	保留
2: 0	PR[2: 0]	RW	0	<p>预分频值。</p> <p>通过配置该寄存器选择计数器时钟的预分频值。</p> <p>要改变该寄存器，IWDG_SR 寄存器的 PVU 必须为0。</p> <p>000：4分频；</p> <p>001：8分频；</p> <p>010：16分频；</p> <p>011：32分频；</p> <p>100：64分频；</p> <p>101：128分频；</p> <p>110：256分频；</p> <p>111：256分频；</p>

### 27.4.3. 重装载寄存器 (IWDG\_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	保留
11: 0	RL[11: 0]	RW	0xFFFF	IWDG 计数器重装载值。 当向 IWDG_KR 寄存器写入 0xAAAA 时, RL 值会传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此 RL 值和时钟预分频值来计算。 只有当 IWDG_SR.RVU=0 时, 才能对寄存器进行修改。 此外需要注意, 当向 RLR 寄存器写入重装载值后需要等待三个 LSI 时钟才能读出。

### 27.4.4. 状态寄存器 (IWDG\_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-	-	保留
1	RVU	R	0	看门狗计数器重装值更新。 必须要软硬件启动 IWDG 后才可让 RVU 正常工作 该位由硬件置1, 表明重装载值正在更新。当重装载值更新结束后, 此位由硬件清零。
0	PVU	R	0	看门狗预分频值更新。 必须要软硬件启动 IWDG 后才可让 PVU 正常工作

				该位由硬件置1，表明预分频值正在更新。当预分频值更新结束后，此位由硬件清零。
--	--	--	--	--

注：在更新 IWDG\_PR.PR、IWDG\_RLR.RL 前，要分别等待 IWDG\_SR.PVU、IWDG\_SR.RVU 为0。但在更新 IWDG\_PR.PR、IWDG\_RLR.RL 后，不必再等待 IWDG\_SR.PVU、IWDG\_SR.RVU 为0，可继续执行下面的代码。

## 28. 窗口看门狗 (WWDG)

### 28.1. 简介

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 APB 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。

### 28.2. 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)

- 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)

### 28.3. WWDG 功能描述

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。

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。

### 28.3.1. WWDG 架构框图

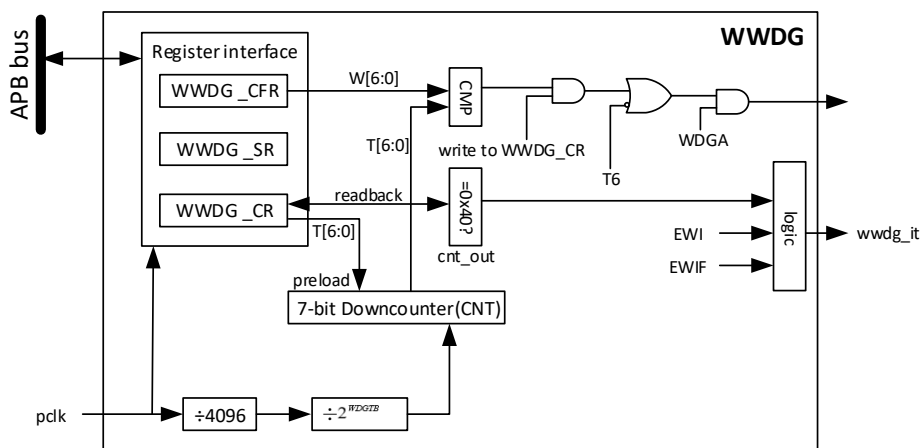


图 28-1 窗口看门狗架构框图

### 28.3.2. 启动看门狗

在系统复位后，看门狗总是处于关闭状态，设置 WWDG\_CR 寄存器的 WDGA 位或者选项字节 (option byte) 都能够开启看门狗，之后除非执行复位操作，否则不能再次关闭。

在选项字节 (option byte) 中有 WWDG\_SW 寄存器位，也可启动看门狗，其值为：

0：硬件看门狗

1：软件看门狗

硬件启动与软件启动只要设置了其中一种，便会启动看门狗。

### 28.3.3. 控制递减计数器

递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。

T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG\_CR 寄存器时，预分频值是未知的。配置寄存器 (WWDG\_CFR) 中的窗口寄存器有上限值，要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于 0x3F 时被重新装载，图 28-2 描述了窗口寄存器的工作过程。

另一个重装计数器的方法是利用早期唤醒中断 (EWI)。设置 WWDG\_CFR 寄存器中的 WEI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序 (ISR) 可以用来重载计数器以防止 WWDG 复位。在 WWDG\_SR 寄存器中写'0'可以清除该中断。

注：可以用 T6 位产生一个软件复位 (设置 WDGA 位为'1'，T6 位为'0')。

### 28.3.4. 高级看门狗中断功能

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）可用于重载 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

### 28.3.5. 如何编写看门狗超时程序

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。

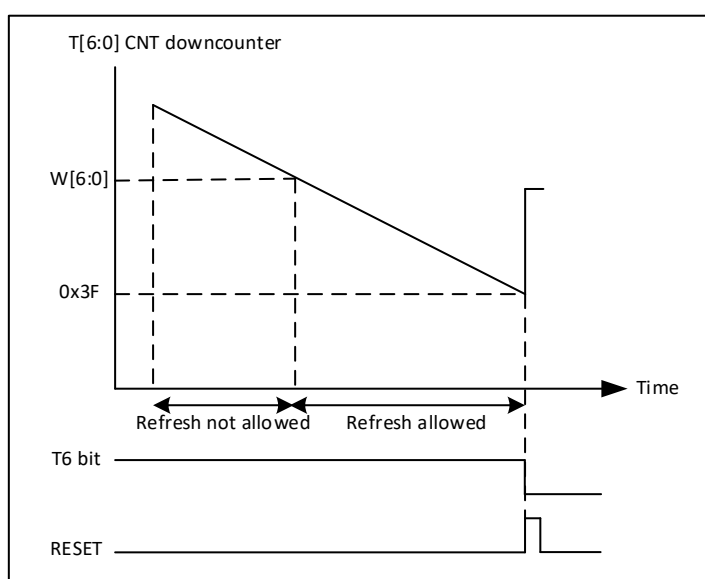


图 28-2 窗口看门狗时序图

计算 WWDG 超时值的公式如下：

$$t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WWDGTB}[1:0]} \times (\text{T}[5:0] + 1) \quad (\text{ms})$$

其中：

1.  $t_{\text{WWDG}}$ ：WWDG 超时
2.  $t_{\text{PCLK}}$ ：APB 时钟周期，以 ms 为测量单位
3. 4096：对应于内部分频器的值

### 28.3.6. 调试模式

当微控制器进入调试模式时（Cortex-M0+核心停止），根据 DBGMCU 模块中的 DBG\_WWDG\_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见调试模式章节。

## 28.4. WWDG 寄存器

### 28.4.1. 控制寄存器 (WWDG\_CR)

Address offset: 0x00

Reset value: 0x0000 007F

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

Bit	Name	R/W	Reset Value	Function
31: 8	保留	-	-	保留
7	WDGA	RS	0	WDGA: 激活位 (Activation bit)。 此位由软件置'1', 但仅能由硬件在复位后清'0'。当WDGA=1时, 看门狗可以产生复位。 0: 禁止; 1: 使能;
6: 0	T[6: 0]	RW	7F	7位计数器 (MSB 至 LSB)。 该寄存器用来存储看门狗的计数值。每 (4096x2 <sup>WDGTB</sup> ) 个 PCLK 周期减1.当计数值从 40h 变为3Fh 时 (T[6]变为0), 产生看门狗复位。

### 28.4.2. 配置寄存器 (WWDG\_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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]		T[6: 0]						
-	-	-	-	-	-	RS	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9	EWI	RS	0	提前唤醒中断。 该位置1, 则当计数器值达到40 h 时, 即产生中断。 此位只能由硬件在复位后清除。
8: 7	WDGTB[1: 0]	RW	0	时基 (Timer base)。 预分频器的时基设置如下: 00: CK 计数器时钟 (PCLK 除以4096) 除以1

				01: CK 计数器时钟 (PCLK 除以4096) 除以2 10: CK 计数器时钟 (PCLK 除以4096) 除以4 11: CK 计数器时钟 (PCLK 除以4096) 除以8
6: 0	W[6: 0]	RW	7F	7位窗口值。 该寄存器包含了用来与递减计数器进行比较用的窗口值。

### 28.4.3. 状态寄存器 (WWDG\_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EWIF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 1	保留	-	-	保留
0	EWIF	RC_W0	0	提前唤醒中断标志。 当计数器值达到40h, 此位由硬件置1.软件写0清零, 写1无效。 当中断未被使能时, 该位也置1.



## 29. 实时时钟 (RTC)

### 29.1. 简介

实时时钟 (Real time clock) 是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

### 29.2. RTC 主要特性

- 可编程的预分频系数：分频系数最高为  $2^{20}$
- 32 位的可编程计数器，可用于较长时间段的测量
- 2 个单独的时钟：用于 APB 接口的 PCLK 和 RTC 时钟 (RTC 时钟的频率必须小于 PCLK 时钟频率的四分之一以上)
  - 可以选择以下三种 RTC 的时钟源：
    - HSE 时钟除以 128
    - LSI 振荡器时钟
    - LSE 振荡器时钟
- 2 个独立的复位类型：
  - APB 接口由系统复位；
  - RTC 核心 (预分频器、闹钟、计数器和分频器) 只能由备份域复位。
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断
  - 秒中断，用来产生一个可编程的周期性中断信号 (最长可达 1 秒)
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态

### 29.3. RTC 功能描述

#### 29.3.1. 总览

RTC 由两个主要部分组成 (参见下面框图)。第一部分 (APB 接口) 用来和 APB 总线相连。此单元还包含一组 16 位寄存器 (RTC\_CRL 和 RTC\_CRH, 实际分散在两个地址的寄存器。本章中将所有此类寄存器组的描述增加后缀 x, 如 RTC\_CRx), 可通过 APB 总线对其进行读写操作。APB 接口由 APB 总线时钟驱动, 用来连接 APB 总线。

另一部分 (RTC 核心) 由一组可编程计数器组成, 分成两个主要模块。

第一个模块是 RTC 的预分频模块, 它可编程产生最长为 1 秒的 RTC 时间基准 TR\_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器 (RTC 预分频器)。如果在 RTC\_CRH 寄存器中设置了相应的允许位, 则在每个 TR\_CLK 周期中 RTC 产生一个中断 (秒中断)。

第二个模块是一个 32 位的可编程计数器, 可被初始化为当前的系统时间。系统时间按 TR\_CLK 周期累加并与存储在 RTC\_ALRx 寄存器中的可编程时间相比较, 如果 RTC\_CRH 控制寄存器中设置了相应允许位, 比较匹配时将产生一个闹钟中断。

所有中断都可以作为 Stop 模式的唤醒信号，如下图所示，在 Stop 模式下，RTC 对应中断作为唤醒需要预先配置对应的中断使能位以及 EXTI line19 对应寄存器。

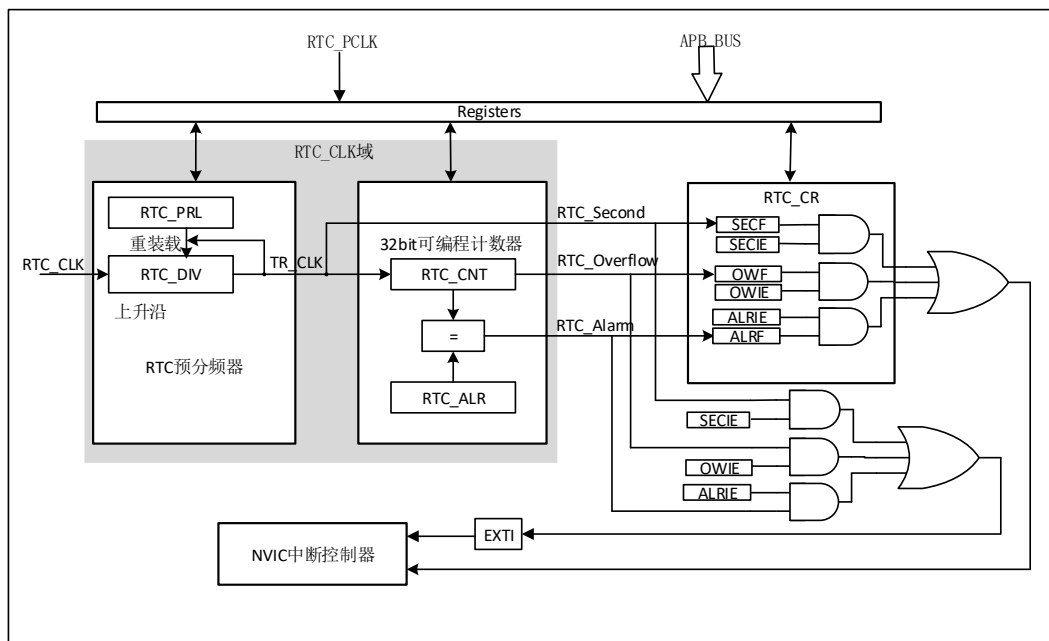


图 29-1 RTC 框图

PWR\_CR1 寄存器的 DBP 位用来控制对 RTC 寄存器的写保护禁止。默认情况 DBP=0，不能对 RTC 寄存器进行写访问，软件置位 DBP 后，才能实现对 RTC 寄存器的写操作。

### 29.3.2. 复位 RTC 寄存器

RTC\_PRLx, RTC\_ALRx, RTC\_CNTx 和 RTC\_DIVx 寄存器仅能通过 RTC 模块软复位或电源复位信号复位。其他系统寄存器 (RTC\_CRx) 由系统复位或电源复位进行异步复位。

注意，当不能对 RTC 模块产生复位的复位源，不仅不能复位 RTC 模块，也不能复位送给 RTC 模块的时钟源使能信号，也不能复位送给 RTC 模块的时钟源选择控制。

### 29.3.3. 读 RTC 寄存器

RTC 核完全独立于 RTC APB 接口。软件通过 APB 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在 RTC 时钟的上升沿同步到 RTC APB 时钟后的信号有效时更新。RTC 标志也是如此的。

这意味着，如果 APB 接口曾经被关闭，而读操作又是在刚刚重新开启 APB 之后，则在第一次的内部寄存器更新之前，从 APB 上读出的 RTC 寄存器数值可能被破坏了（通常读到 0）。下述几种情况下能够发生这种情形：

- 1) 发生系统复位或电源复位
- 2) 系统刚从 Stop 模式唤醒（这种情况计数值只是不会更新，因为 CPU 不工作，系统时钟停止，但 RTC 正常计数，计数值不会同步到 V<sub>DDD</sub> 区）

所有以上情况中，APB 接口被禁止时（复位、无时钟），RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器时，RTC 的 APB 接口曾经处于禁止状态，则软件首先必须等待 RTC\_CRL 寄存器中的 RSF 位（寄存器同步标志）被硬件置'1'。

说明:

- CPU 可读的寄存器包括 RTC\_CRx, RTC\_CNTx, RTC\_DIVx 和 RTC\_ALRx;
- RTC\_CRx 寄存器为 RTC\_PCLK 域, CPU 任何时候读能读到稳定值;
- RTC\_CNTx 和 RTC\_DIVx 来源于 RTC\_CLK 域。在 RTC 工作后, RTC\_DIVx 寄存器在每个 RTC\_CLK 的上升沿更新; RTC\_CNTx 和来源于 RTC\_CLK 时钟域的标志位同样采用跟 RTC\_DIVx 寄存器同样的更新信号, 虽然这样不是每次更新时 RTC\_CNTx 的值都改变;
- RSF 实现在 RTC\_PCLK 域, 在 RTC\_CLK 同步到 RTC\_PCLK 后的脉冲信号有效时置位;
- RSF 仅控制 RTC\_CNTx 和 RTC\_DIVx 的读取时机 (硬件不会控制)

#### 29.3.4. 配置 RTC 寄存器

必须通过置位 RTC\_CRL 寄存器的 CNF 位, 进入配置模式, 才能写入 RTC\_PRLx、RTC\_CNTx、RTC\_ALRx、BKP\_RTCCR 寄存器。

另外, 对 RTC 任何寄存器的写操作, 都必须在前一次写操作结束后进行。可以通过查询 RTC\_CRL 寄存器中的 RTOFF 状态位, 判断 RTC 寄存器是否处于更新中。仅当 RTOFF 状态位是'1'时, 才可以写入 RTC 寄存器。

**配置过程:**

- 1) 查询 RTOFF 位, 直到 RTOFF 的值变为'1';
- 2) 置 CNF 值为1, 进入配置模式;
- 3) 对一个或多个 RTC 寄存器进行写操作;
- 4) 清除 CNF 标志位, 退出配置模式;
- 5) 查询 RTOFF, 直至 RTOFF 位变为'1'以确认写操作已经完成。

注: 仅当 CNF 标志位被清除时, 写操作才能进行, 这个过程至少需要3个 RTC\_CLK 周期。(在清除 CNF 标志位后3个 RTC\_CLK 不能重新启动配置, 否则会出现配置错误 (此时通过 RTOFF=0控制) )

说明:

- 在此过程中, CPU 写寄存器时 RTOFF=1;
- CPU 的写周期为从 CNF=1到 CNF=0, 配置其他寄存器在这两个操作之间;
- 先将 CNF 写1后将 CNF 清零, 这个操作清零 RTOFF; 只写 CNF=0或者写 CNF=1后不清零不会清零 RTOFF;
- 先将 CNF 写1后将 CNF 清零, 这个操作将缓存寄存器值写入 RTC\_CLK 域寄存器;
- RTOFF 实现在 RTC\_PCLK 域;

#### 29.3.5. RTC 标志的设置

在每一个 RTC\_CLK 时钟周期, 更改 RTC 计数器之前, 硬件置位 RTC 秒标志 (SECF)。在计数器到达 0x0000 之前的最后一个 RTC 时钟周期, RTC 溢出标志 (OWF) 被置位。

在计数器的值到达闹钟寄存器的值加 1 (ALR+1) 之前的 RTC 时钟周期, 置位 RTC\_Alarm 和 RTC 闹钟标志 (ALRF)。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志同步:

- (1) 使用 RTC 闹钟中断, 并在中断处理程序中修改 RTC 闹钟和/或 RTC 计数器。
- (2) 等待 RTC 控制寄存器中的 SECF 位被设置, 再更改 RTC 闹钟和/或 RTC 计数器。

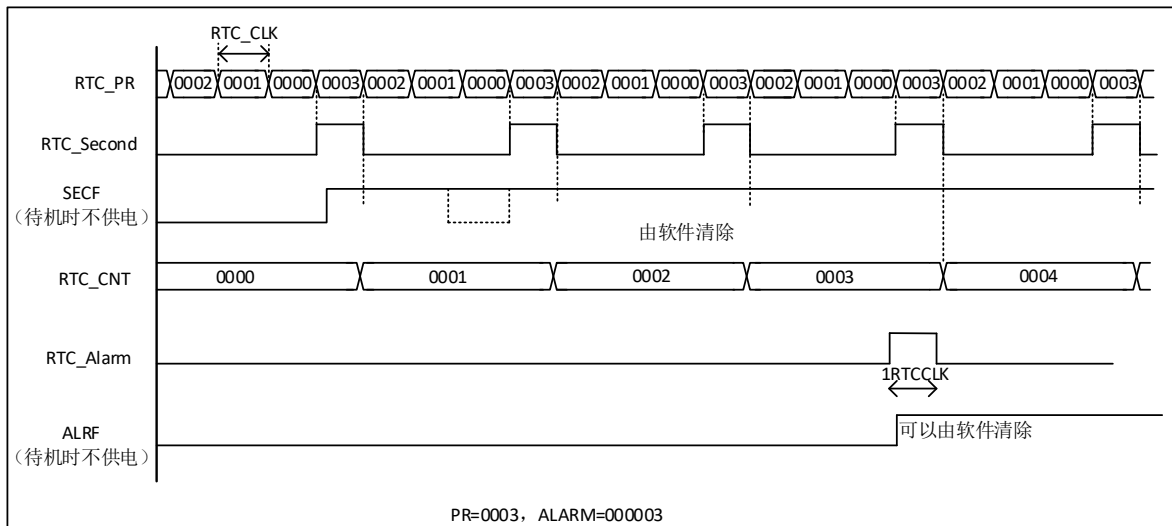


图 29-2 RTC 秒和闹钟波形图示例, PR=0003, ALR=00003

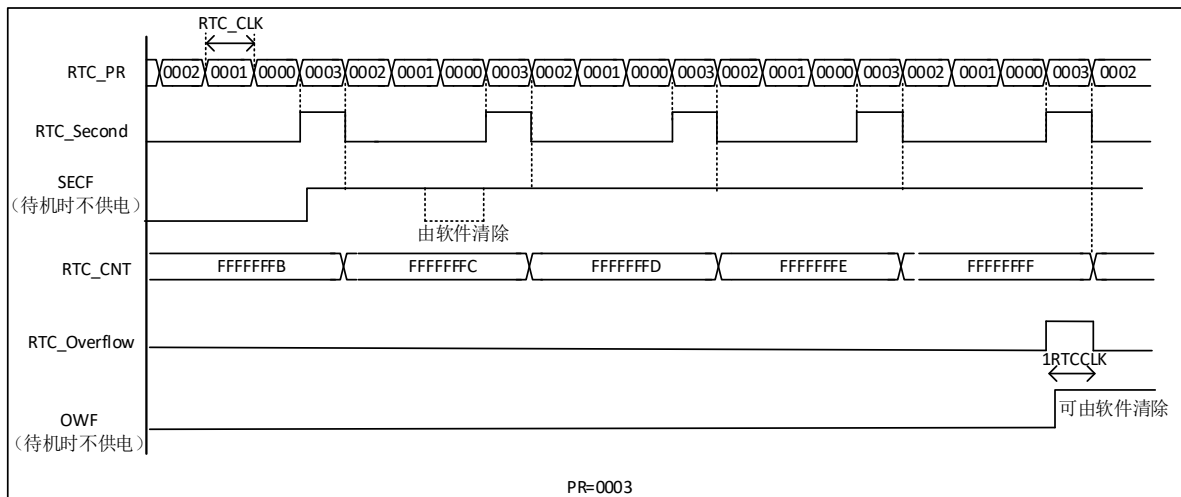


图 29-3 RTC 溢出波形图示例, PR=0003

### 29.3.6. RTC 校准

为了测量目的, RTC 时钟的 64 分频可以输出到 IO 引脚上 (PF5)。该功能是通过置位 CCO bit (BKP\_RTCCR 寄存器) 实现的。

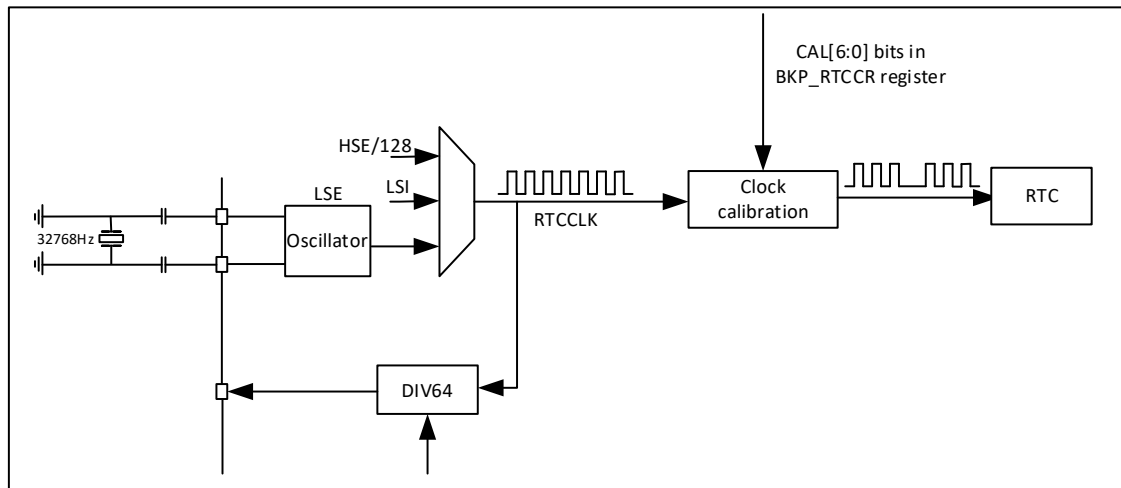


图 29-4 RTC 校准图

## 29.4. RTC 寄存器

### 29.4.1. RTC 控制寄存器 (RTC\_CRH)

Address offset: 0x00

Reset value: 0x0000

该寄存器由系统复位复位。

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	OWIE	ALR IE	SEC IE
-	-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 3	保留	-	-	保留
2	OWIE	RW	0	溢出中断允许位 0: 不允许溢出中断 1: 允许溢出中断
1	ALRIE	RW	0	闹钟中断允许位 0: 不允许闹钟中断 1: 允许闹钟中断
0	SECIE	RW	0	秒中断允许位 0: 不允许秒中断 1: 允许秒中断

这些位用于屏蔽中断请求。注意：在复位后，所有中断是未使能的，所以在初始化后，写 RTC 寄存器以确保没有正在挂起的中断请求是可能的。但是当外设正在完成前一次的写操作（RTOFF=0）时，是不能写 RTC\_CRH 寄存器的。

该控制寄存器控制着 RTC 的功能。某些位必须使用专门的配置流程才能进行写操作。

## 29.4.2. RTC 控制寄存器 (RTC\_CRL)

Address offset: 0x04

Reset value: 0x0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										RTOFF	CNF	RSF	OWF	ALRF	SECF
-										R	RW	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 6	保留	-	-	保留
5	RTOFF	R	1	RTC 操作关闭 (RTC operation OFF)，该位只读。 RTC 模块利用该位来指示对其寄存器进行的最后一次操作的状态（指示操作是否完成）。 若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。 0: 上一次对 RTC 寄存器的写操作仍在进行 1: 上一次对 RTC 寄存器的写操作已经完成
4	CNF	RW	0	配置标志 (Configuration flag) 此位必须由软件置'1'以进入配置模式，从而允许向 RTC_CNTx、RTC_ALRx 或 RTC_PRLx 寄存器写入新值。 只有当此位在被置'1'，并重新由软件清'0'后，才会执行写操作。 0: 退出配置模式 (开始更新 RTC 寄存器) 1: 进入配置模式
3	RSF	RC_W0	0	寄存器同步标志 (Registers synchronized flag) 当 RTC_CNTx 寄存器和 RTC_DIVx 寄存器更新时，硬件置'1'该位，软件清零该位。 在 APB 复位后，或 APB 时钟停止后，此位必须由软件清'0'。

				<p>要进行任何的读操作之前，用户程序必须等待该位被硬件置'1'，以确保 RTC_CNTx、RTC_ALRx 或 RTC_PRLx 已经被同步。</p> <p>0: 寄存器尚未被同步 1: 寄存器已经被同步</p>
2	OWF	RC_W0	0	<p>溢出标志 (Overflow flag)</p> <p>当32位可编程计数器溢出时，此位由硬件置'1'。如果 RTC_CRH 寄存器中 OWIE=1，则产生中断。此位只能由软件清'0'，写'1'无效。</p> <p>0: 无溢出； 1: 32位可编程计数器溢出</p>
1	ALRF	RC_W0	0	<p>闹钟标志 (Alarm flag)</p> <p>当32位可编程计数器达到 RTC_ALRx 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清'0'，写'1'无效。</p> <p>0: 无闹钟 1: 有闹钟</p>
0	SECF	RC_W0	0	<p>秒标志 (Second flag)</p> <p>当32位可编程预分频器溢出时，此位由硬件置'1'，同时 RTC 计数器加1。</p> <p>因此，此标志为分辨率可编程的 RTC 计数器提供一个周期性的信号 (通常为1秒)。如果 RTC_CRH 寄存器中 SECIE=1，则产生中断。此位只能由软件清除，写'1'无效。</p> <p>0: 秒标志条件不成立 1: 秒标志条件成立</p>

RTC 的功能是被该控制寄存器控制的。当外设正在继续上一次写操作时 (RTOFF=0)，是不能写 RTC\_CRx 寄存器的。

注:

- 任何标志位都将保持挂起状态，直到被软件复位，表示所请求的中断已经被接受。
- 在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作 (指从缓存寄存器向 RTC\_CLK 域寄存器的写操作)。
- 当 APB 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新 (无法同步)。
- OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零。

### 29.4.3. RTC 预分频器装载寄存器 (RTC\_PRLH)

PRL 寄存器保持 RTC 预分频器周期性的计数值。该寄存器是被 RTC\_CRL 寄存器的 RTOFF 位写保护的，只有 RTOFF=1，才允许 CPU 进行写操作（写入到 buffer 寄存器）。

Address offset: 0x08

Reset value: 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	PRL[19: 16]			
-	-	-	-	-	-	-	-	-	-	-	-	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 4	保留	-	-	保留
3: 0	PRL[19: 16]	W	0	RTC 预分频装载值高位 (RTC prescaler reload value high) 根据以下公式, 这些位用来定义计数器的时钟频率: $f_{TR\_CLK} = f_{RTC\_CLK} / (PRL[19: 0] + 1)$ 注: 不推荐使用0值, 否则无法正确的产生 RTC 中断和标志位

### 29.4.4. RTC 预分频器分频寄存器 (RTC\_PRL)

Address offset: 0x0C

Reset value: 0x8000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	PRL[15: 0]	W	0x8000	RTC 预分频装载值高位 (RTC prescaler reload value high) 根据以下公式, 这些位用来定义计数器的时钟频率: $f_{TR\_CLK} = f_{RTC\_CLK} / (PRL[19: 0] + 1)$



Bit	Name	R/W	Reset Value	Function
				注：不推荐使用0值，否则无法正确的产生 RTC 中断和标志位

### 29.4.5. RTC 预分频分频因子寄存器高位 (RTC\_DIVH)

在每个 TR\_CLK 周期，RTC\_PRLx 寄存器的值被重装载到 RTC 预分频计数器里。用户可以通过读取 RTC\_DIVx 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。

该寄存器只读属性，当 RTC\_PRLx 或者 RTC\_CNTx 寄存器的值发生任何变化，该寄存器值将由硬件重新装载。

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 4	保留	-	-	保留
3: 0	RTC_DIV[19: 16]	R	0	RTC 时钟分频器高位。

### 29.4.6. RTC 预分频分频因子寄存器低位 (RTC\_DIVL)

Address offset: 0x14

Reset value: 0x0000 8000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	DIV[15: 0]	R	0x8000	RTC 时钟分频器低位

### 29.4.7. RTC 计数寄存器高位 (RTC\_CNTH)

RTC 模块有个32 bits 可编程的计数器，该寄存器通过两个16bit 的寄存器访问，计数基于预分频器产生的 TR\_CLK 时间基准为参考进行计数。

RTC\_CNTx 寄存器保持该计数器的计数值。寄存器是被写保护的，仅当 RTOFF=1时才能进行写操作。对高16bit 的 RTC\_CNTH 或者低16bit 的 RTC\_CNTL 寄存器进行写操作，直接装载到相应的可编程计数器里，并重装载 RTC 预分频器。当读操作发生，返回计数器的当前值（系统时间）。

**Address offset:** 0x18

**Reset value:** 0x0000

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
RTC_CNT[31: 16]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	RTC_CNT[31: 16]	RW	0x0000	RTC 计数器的高16bit 当读 RTC_CNTH 寄存器时，返回 RTC 计数器寄存器的当前值的高16bit。只有进入配置模式才能对该寄存器进行写操作。

#### 29.4.8. RTC 计数寄存器低位 (RTC\_CNTL)

**Address offset:** 0x1C

**Reset value:** 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
RTC_CNT[15: 0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	RTC_CNT[15: 0]	RW	0x0000	RTC 计数器低16bit 当读 RTC_CNTL 寄存器时，返回 RTC 计数器寄存器当前值的低16bit。只有进入配置模式才能对该寄存器进行写操作。

#### 29.4.9. RTC 闹钟寄存器高位 (RTC\_ALRH)

当可编程计数器（计数）达到存储在 RTC\_ALRx 寄存器的32bit 值时，并产生闹钟中断请求。该寄存器是被 RTOFF 位写保护的，只有 RTOFF=1，才允许写访问。

Address offset: 0x20

Reset value: 0xFFFF

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
RTC_ALR[31: 16]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ALR[31: 16]	RW	0xFFFF	RTC 闹钟寄存器高16bit 软件可写闹钟时间的高16bit。写该寄存器必须进入配置模式。

#### 29.4.10. RTC 闹钟寄存器高位 (RTC\_ALRL)

Address offset: 0x24

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 0	ALR[15: 0]	RW	0xFFFF	RTC 闹钟寄存器低16bit 软件可写闹钟时间的低16bit。写该寄存器必须进入配置模式。

#### 29.4.11. RTC 时钟校准及输出配置寄存器 (BKP\_RTCCR)

Address offset: 0x2C

Reset value: 0x0000 0000 (仅能被 por 及 bdcr 软复位复位)

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	ASOS	ASOE	CCO	CAL[6: 0]						

-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

Bit	Name	R/W	Reset Value	Function
31: 10	保留	-	-	保留
9	ASOS	RW	0	秒/闹钟脉冲输出选择位 当 ASOE 位被置位, ASOS 位可以被用作选择 Pin 上输出是 RTC 秒脉冲还是闹钟信号 0: RTC 闹钟脉冲信号 1: RTC 秒脉冲信号
8	ASOE	RW	0	秒/闹钟脉冲输出使能位 当置位该位, 则由 ASOS 位决定 RTC_OUT 引脚上输出是 RTC 秒脉冲还是闹钟脉冲信号。
7	CCO	RW	0	校准时钟输出位 当 ASOE 没有置位时, 可以置位 CCO 输出 RTC 时钟的64分频 0: 无作用 1: pin 上输出 RTC 时钟的64分频
6: 0	CAL[6: 0]	RW	0	校准值 该值显示了每 $2^{20}$ 个时钟可忽略的时钟脉冲个数, 这允许 RTC 进行校准, 以 $1000000/2^{20}$ PPM 的步长减慢时钟。 RTC_CLK 可以被减慢的从0到121PPM。

## 30. I<sup>2</sup>C 接口

### 30.1. 介绍

I<sup>2</sup>C（内部集成电路）总线接口连接微控制器和串行 I<sup>2</sup>C 总线。它提供多主机功能，可以控制所有 I<sup>2</sup>C 总线特定的序列、协议、仲裁和时序。它支持标准（Sm）、快速（Fm）两种模式。

根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

### 30.2. I<sup>2</sup>C 主要特点

- 从模式和主模式
- 多主机功能
- 支持不同通讯速度
  - 标准模式（Sm）：高达 100 kHz
  - 快速模式（Fm）：高达 400 kHz
- 作为主机
  - 时钟产生
  - 起始和停止的产生
- 作为从机
  - 可编程的 I<sup>2</sup>C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止条件的检测
- 7 位/10 位寻址模式
- 支持广播呼叫（General call）功能
- 状态标志位
  - 发送/接收模式标志位
  - 字节传输完成标志位
  - I<sup>2</sup>C 总线忙标志位
- 错误标志位
  - 主机仲裁丢失
  - 地址/数据传输后的 ACK 失败
  - 起始和停止错误
  - 过载（Overrun）/欠载（Underrun）（时钟拉长功能禁止）
- 可选的时钟拉长功能
- 具备 DMA 能力的单字节缓冲
- 软件复位
- 模拟噪声滤波功能
- 可配置的 PEC（数据包错误校验）生成和验证
  - PEC 值可以在 Tx 模式下的最后一个字节发送

- 接收最后字节做 PEC 错误检查
- 兼容 SMBus
  - 25 ms 时钟低超时延时
  - 10 ms 主设备累积时钟低扩展时间
  - 25 ms 从设备累积时钟低扩展时间
  - 带 ACK 控制的硬件 PEC 生成和验证
  - 支持地址分辨协议 (ARP)

## 30.3. I<sup>2</sup>C 功能描述

### 30.3.1. I<sup>2</sup>C 框图

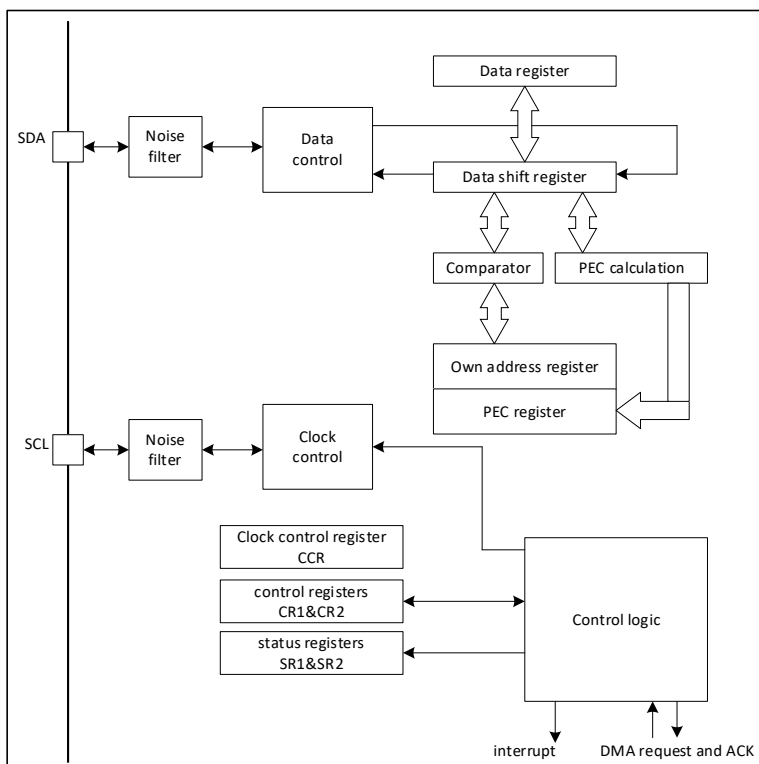


图 30-1 I<sup>2</sup>C 框图

### 30.3.2. 模式选择

I<sup>2</sup>C 支持以下四种模式：

- 从发送器模式 (Slave transmitter)
- 从接收器模式 (Slave receiver)
- 主发送器模式 (Master transmitter)
- 主接收器模式 (Master receiver)

默认情况下，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。

### 30.3.2.1. 通信流程

作为主机，I<sup>2</sup>C 接口启动数据传输，并产生时钟信号。串行数据的传输总是以起始条件开始，并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

作为从机，I<sup>2</sup>C 接口能识别自己的地址（7 位/10 位）和广播呼叫（general call）地址。软件能够控制开启或禁止对广播呼叫（general call）地址的识别。

数据和地址按 8 位（字节）进行传输，高位在前。跟在起始条件后的 1 个字节是地址。地址只在主模式下发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收方必须回送一个应答位（ACK）给发送方。参见下图。

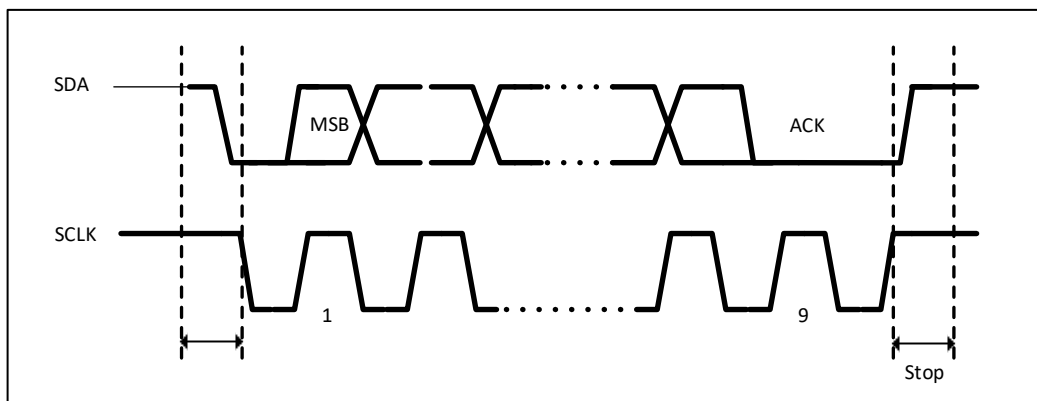


图 30-2 I<sup>2</sup>C 总线协议

软件可启用或不启用应答（ACK）位。软件也可以选择 I<sup>2</sup>C 接口地址（7 位、10 位或广播呼叫地址）。

### 30.3.3. I<sup>2</sup>C 初始化

#### 30.3.3.1. 使能/关闭 I<sup>2</sup>C 模块

I<sup>2</sup>C 的时钟模块先要通过 RCC\_APBENR1 寄存器的 I2C\_EN 位打开，然后通过设定 I2C\_CR1 的 PE 位使能 I<sup>2</sup>C 模块。

#### 30.3.3.2. I<sup>2</sup>C 时序设定

数据信号（SDA）的保持和建立时间要求满足 I<sup>2</sup>C 标准协议，需要进行 I<sup>2</sup>C 时序的设定。这是通过写 I2C\_CCR 和 I2C\_TRISE 寄存器实现的。

### 30.3.4. I<sup>2</sup>C 从模式

默认情况下，I<sup>2</sup>C 接口总是工作在从模式。从从模式切换到主模式，需要产生一个起始条件。

为了产生正确的时序，必须在 I2C\_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：4 MHz
- 快速模式下为：8 MHz

一旦检测到起始条件，在 SDA 线上接收到的地址，被送到移位寄存器，并与 I<sup>2</sup>C 的地址 OAR1 或者广播呼叫地址（如果 ENGC=1）相比较。

#### 头段或地址不匹配:

I<sup>2</sup>C 接口将其忽略并等待另一个起始条件。

#### 地址匹配:

I<sup>2</sup>C 接口产生以下时序:

- 如果 ACK 被软件置'1'，则产生一个应答脉冲
- 硬件置位 ADDR 位，如果设置了 ITEVTEN 位，则产生中断

在从模式下 TRA 位指示当前是处于接收器模式还是发送器模式。

#### 30.3.4.1. 从发送器

在接收到地址并清除 ADDR 位后，（如果地址字节的最低位是1）从机将数据（字节）从 DR 寄存器，经由内部移位寄存器发送到 SDA 上。

从机拉低 SCL，直到 ADDR 位被清除，并且待发送数据已写入 DR 寄存器。

当收到应答脉冲时：TxE 位被硬件置位，如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 位被置位，但在下一个数据发送结束之前，没有新数据写入到 I2C\_DR 寄存器，则 BTF 位被置位。从机拉低 SCL，直到 BTF 位被软件清零（读 I2C\_SR1 之后，再写入 I2C\_DR 寄存器）。

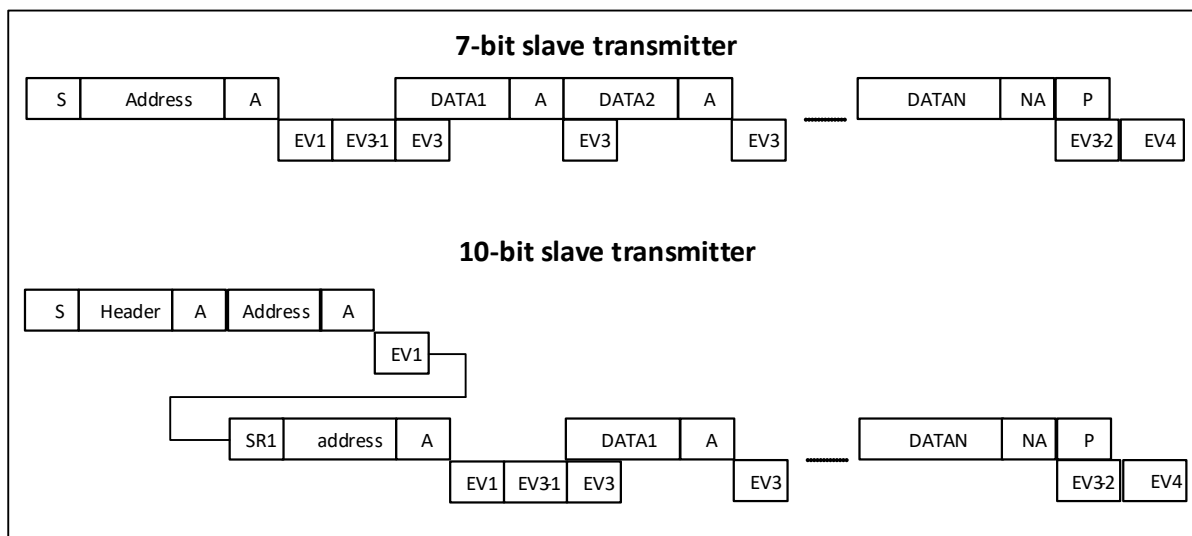


图 30-3 从发送器的传送序列图

**Legend:** S= Start (起始条件) , Sr= Repeated Start (重复的起始条件) , P= Stop (停止条件) , A= Acknowledge (响应) , NA= Non-acknowledge (不响应) , EVx= Event (ITEVFEN= 1时产生中断)

**EV1:** ADDR=1, 通过先读 SR1寄存器, 再读 SR2寄存器清零 ADDR 位

**EV3-1:** TxE=1, 移位寄存器为空, 数据寄存器为空, 向 DR 寄存器写 Data1

**EV3:** TxE=1, 移位寄存器不为空, 数据寄存器为空, 向 DR 寄存器写 (Data2) 清零 TxE

**EV3-2:** AF=1; 软件向 AF 位写0清零该位

**EV4:** STOPF=1, 通过先读 SR1寄存器, 后写 CR1寄存器实现对该位的清零。

#### 注:

- 1 ) EV1和 EV3\_1事件拉低 SCL，直到对应的软件序列结束。
- 2 ) EV3的软件序列必须在当前字节传输结束之前完成



### 30.3.4.2. 从接收器

在接收到地址并清除 ADDR 后，（如果地址字节的最低位是0）从机将通过内部移位寄存器把从 SDA 线接收到的字节存进 DR 寄存器。I<sup>2</sup>C 接口在接收到每个字节后都执行下列操作：

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 RxNE 被置位，并且在接收新的数据结束之前，DR 寄存器未被读出，则 BTF 位被置位，在清除 BTF（读出 I2C\_SR1之后再 I2C\_DR 寄存器）之前，从机一直拉低 SCL。（见下图）。

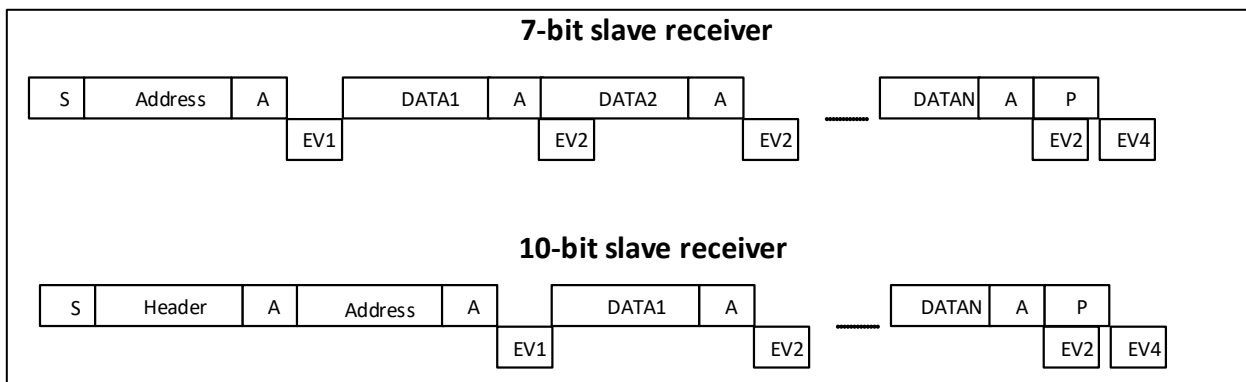


图 30-4 从接收器的传送序列图

**Legend:** S= Start, Sr= Repeated Start, P= Stop, A= Acknowledged, EVx= Event (ITEVFEN= 1时产生中断)

**EV1:** ADDR=1，通过先读 SR1寄存器，后读 SR2寄存器，实现 ADDR 的清零

**EV2:** RxNE=1，读 DR 寄存器清零该位

**EV4:** STOPF=1，通过先读 SR1寄存器，后写 CR1寄存器实现对该位的清零。

**注:**

- EV1 时间拉低 SCL，直到相应软件序列的结束。
- EV2软件序列必须在当前字节传输完成前即完成。
- 当用户检查 SR1寄存器内容后，应该对每个发现置位的标志位，进行完整的清除序列。比如 ADDR 和 STOPF 标志位，需要用以下序列：

如果 ADDR=1，先读 SR1，再读 SR2；如果 STOPF=1，先读 SR1，再写 CR1。这样做的目的是确保如果 ADDR 和 STOPF 两位都被发现置位，都能被清除掉。

### 30.3.4.3. 关闭通信

在传输完最后一个数据字节后，主机产生一个停止条件，从机检测到该条件时：

- 硬件置位 STOPF，如果设置了 ITEVTEN 位，则产生一个中断。
- 通过先读 SR1，后写 CR1，实现对 STOPF 位的清零。

### 30.3.5. I<sup>2</sup>C 主模式

在主模式时，I<sup>2</sup>C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始，并以停止条件结束。

当通过 START 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

- 在 I2C\_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置 I2C\_CCR 寄存器
- 配置 I2C\_TRISE 寄存器
- 编程 I2C\_CR1 寄存器启动外设
- 置 I2C\_CR1 寄存器中的 START 位为 1，产生起始条件

I<sup>2</sup>C 模块的输入时钟频率必须至少是：

- 标准模式下为：4 MHz
- 快速模式下为：8 MHz

#### 30.3.5.1. 主机产生时钟

CCR 寄存器以上升沿或者下降沿，产生 SCL 的高电平和低电平。由于从机可能拉长 SCL 信号，在 SCL 上升沿产生后，主机在 TRISE 寄存器编程的时间到达时，检查来自总线的 SCL 信号。

- 如果 SCL 是低电平，意味着从机正在拉长 SCL 总线，高电平计数器停止计数，直到 SCL 被检测到高电平。这是为了确保 SCL 参数的最小高电平时间。
- 如果 SCL 是高电平，高电平计数器保持计数。

实际上，即使从机不拉长 SCL，从 SCL 上升沿产生，到 SCL 上升沿被发现，这样的反馈环路也是要花费些时间的。这个回路的时间与 SCL 的上升时间（SCL 的 VIH 数据检测）有关系，再加上 SCL 输入路径的模拟噪声滤波，以及芯片内部由于用 APB 时钟进行的 SCL 同步。反馈回路的最大时间编程在 TRISE 寄存器中，所以无论 SCL 上升时间如何，SCL 的频率保持稳定。

#### 30.3.5.2. 开始条件

当 BUSY=0 时，设置 START=1，I<sup>2</sup>C 接口将产生一个起始条件，并切换至主模式（MSL 被置位）。

注：在主模式下设置 START 位，将在当前字节传输完后，由硬件产生一个重新开始条件。

一旦发出起始条件：

- SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。

主机读 I2C\_SR1 寄存器，再把从机地址写入 I2C\_DR 寄存器。

#### 30.3.5.3. 从机地址发送

主机将从机地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头端序列产生以下事件：

— ADDR10 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

然后主设备等待一次读 I2C\_SR1 寄存器，跟着第二个地址字节写入 DR 寄存器。

ADDR 位被硬件置位，如果设置了 TEVFEN 位，则产生一个中断。

随后主设备等待一次读 I2C\_SR1 寄存器，跟着读 I2C\_SR2 寄存器。

- 在 7 位地址模式时，送出一个地址字节。

该地址字节一旦被送出，

— ADDR 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

随后主机读 I2C\_SR1 寄存器，跟着读 I2C\_SR2 寄存器。

根据送出从机地址的最低位，主机决定进入发送器模式，还是进入接收器模式。

- 在 7 位地址模式时，

— 要进入发送器模式，主设备发送从地址时置最低位为 '0' 。

– 要进入接收器模式，主设备发送从地址时置最低位为 ‘1’ 。

TRA 位指示主设备是在接收器模式还是发送器模式。

• 在10位地址模式时，

– 要进入发送器模式，主设备先送头字节（11110xx0），然后送 LSB 位等于 0 的从地址。（头段字节中的 xx 是 10 位地址中的最高 2 位）。

– 要进入接收器模式，主设备先送头字节（11110xx0），然后送 LSB 位等于 0 的从地址。然后再重新发送一个开始条件，后面跟着头字节（11110xx1）。

（头字节中的 xx 是 10 位地址中的最高 2 位）。TRA 位指示主设备是在接收器模式还是发送器模式。

#### 30.3.5.4. 主机发送

在发送了地址和清除了 ADDR 位后，主设备通过内部移位寄存器将字节从 DR 寄存器发送到 SDA 线上。

主设备等待，直到第一个数据字节被写入 I2C\_DR 寄存器（参见 EV8\_1）。

当收到 ACK 脉冲时，TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 被置位，且在上一次数据发送结束之前，没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位。在清除 BTF（读 I2C\_SR1 之后，再写 I2C\_DR 寄存器）之前，I<sup>2</sup>C 接口将保持 SCL 为低电平。

#### 关闭通信

在 DR 寄存器中写入最后一个字节后，通过设置 STOP 位产生一个停止条件（见图的 EV8\_2），然后 I<sup>2</sup>C 接口将自动回到从模式（MSL 位清除）。

注：当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8\_2 事件时。

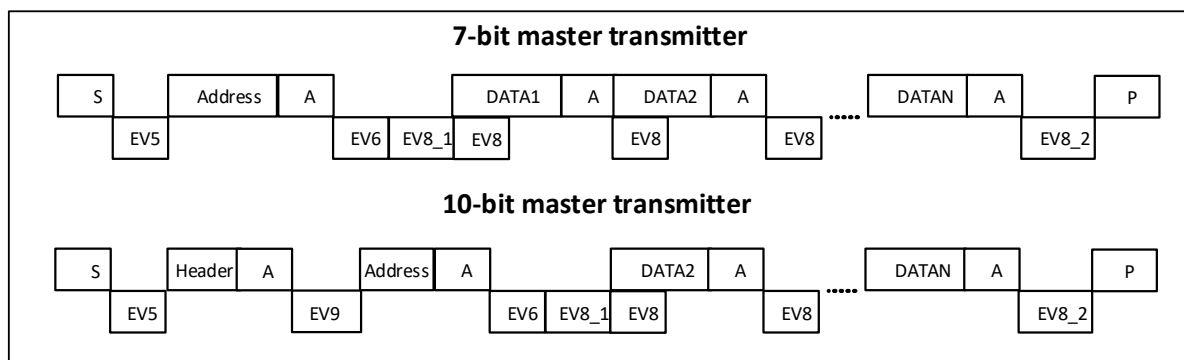


图 30-5 主发送器传送序列图

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (ITEVFEN= 1 时产生中断)

EV5: SB=1, 读 SR1 然后将地址写入 DR 寄存器将清除该事件

EV6: ADDR=1, 读 SR1 然后读 SR2 清除该事件

EV8\_1: TXE=1, 移位寄存器空

EV8: TXE=1, 写 DR 寄存器清除该事件

EV8\_2: TXE=1, BTF=1, 产生停止条件时由硬件清除

EV9: ADDR10=1, 读 SR1 然后写 DR 寄存器清除该事件

注：

- 1) 发生 EV5, EV6, EV8\_1 和 EV8\_2 事件时，拉长 SCL 的低电平，直到相应的软件序列执行结束
- 2) EV8 软件序列必须在当前字节发送完成前执行完毕。若 EV8 的软件序列不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 TxE。

### 30.3.5.5. 主接收器

在发送地址和清除 ADDR 之后，I<sup>2</sup>C 接口进入主接收器模式。在此模式下，I<sup>2</sup>C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DR 寄存器。在每个字节后，I<sup>2</sup>C 接口依次执行以下操作：

- 如果 ACK 位被置位，发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断。

如果 RxNE 位被置位，并且在接收新数据结束前，DR 寄存器中的数据没有被读走，硬件将设置 BTF=1，在清除 BTF 之前 I<sup>2</sup>C 接口将保持 SCL 为低电平；读出 I2C\_SR1 之后再读出 I2C\_DR 寄存器将清除 BTF 位。

#### 关闭通信

##### 方法1：该方法的应用场景是：当 I<sup>2</sup>C 被设成应用程序中最高优先级的中断

主机在从从机接收到最后一个字节后，发送一个 NACK。接收到 NACK 后，从机释放对 SCL 和 SDA 线的控制。主机就可以发送一个停止/重新开始条件。

- 1) 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后（在倒数第二个 RxNE 事件之后）必须清除 ACK 位。
- 2) 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后（在倒数第二个 RxNE 事件之后）设置停止/开始位。
- 3) 当接收单个字节时，关闭应答和停止条件的产生位要刚好在 EV6 之后（EV6\_1 时，清除 ADDR 之后）。

在产生了停止条件后，I<sup>2</sup>C 接口自动回到从模式（MSL 位被清除）。

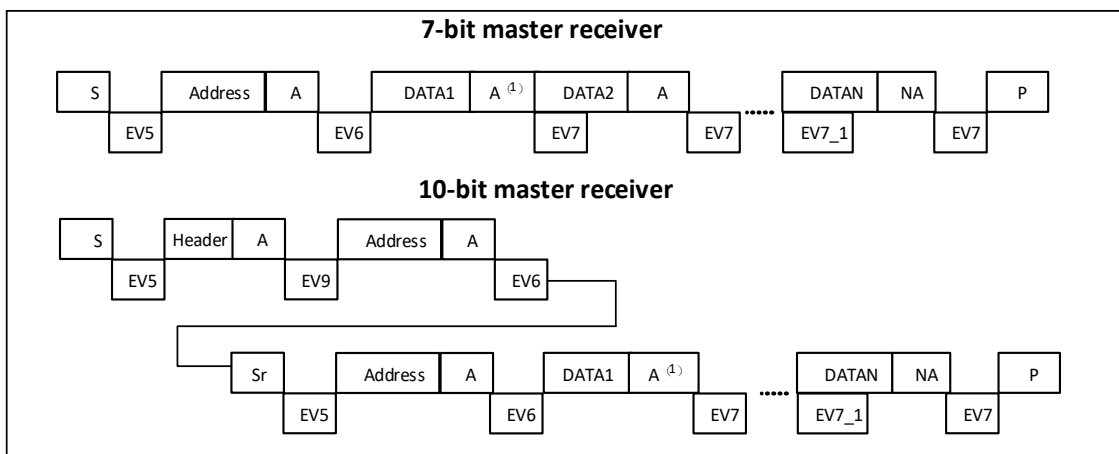


图 30-6 方法1：主模式发送时的时序

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (ITEVFEN= 1时产生中断)

EV5: SB=1, 读 SR1, 再写 DR 寄存器, 该位被清零

EV6: ADDR=1, 读 SR1, 再读 SR2, 该位被清零

EV6\_1: 无相关的标志事件, 仅用作1个字节的接收

EV7: RxNE=1, 读 DR 寄存器, 该位被清零

EV7\_1: RxNE=1, 读 DR 寄存器, 写 ACK=0并置位 STOP

EV9: ADDR10=1, 读 SR1然后写 DR 寄存器清除该事件

- 1) 如果是单个字节接收, 则上述标注为 (1) 的地方会是 NA
- 2) EV5, EV6事件, 拉长 SCL 的低电平, 直到相应的软件序列执行结束

- 3) EV7软件序列必须在当前字节发送完成前执行完毕。若 EV7的软件序列不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 RXNE。
- 4) EV6\_1或者 EV7\_1的软件序列必须在当前字节传输的 ACK 之前完成。

**方法2：这个方法的应用场景是：I<sup>2</sup>C 的中断在应用中不是最高优先级，或者使用查询方式**

用这个方法，DataN-2没有被读，因此在 DataN-1之后，通讯被拉长 (RxNE 和 BTF 都被置位)。然后，在读 DR 寄存器的 DataN-2前，清 ACK 位，以确保在 DataN ACK 之前被清掉。在此之后，在读 DataN-2之后，置位 STOP/START 位，并读 DataN-1。在 RxNE 置位后，读 DataN

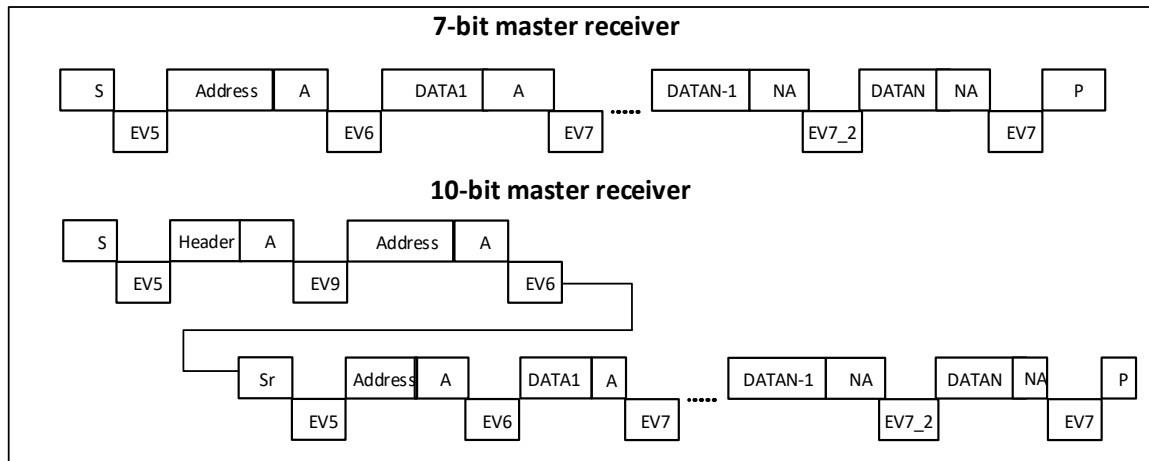


图 30-7 方法2：N>2时主模式发送时的时序

**Legend:** S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (ITEVFEN= 1时产生中断)

**EV5:** SB=1, 先读 SR1寄存器, 再写 DR 寄存器, 清零该位

**EV6:** ADDR, 先读 SR1, 再读 SR2, 清零该位

**EV7:** RxNE=1, 读 DR 寄存器清零该位

**EV7\_2:** BTF=1, DataN-2存在 DR 寄存器中, DataN-1存在移位寄存器中, 写 ACK=0, 读 DR 寄存器中的 DataN-2。置位 STOP, 读 DataN-1

**EV9:** ADDR10=1, 读 SR1然后写 DR 寄存器清除该事件

**注:**

- EV5, EV6事件, 拉长 SCL 的低电平, 直到相应的软件序列执行结束
- EV7软件序列必须在当前字节发送完成前执行完毕。若 EV7的软件序列不能在当前传输的字节结束前完成，则推荐使用 BTF 代替 RXNE。

■ **当3个字节要被读走:**

- RxNE=1 => 不处理 (DataN-2没被读)。
- 接收到 DataN-1
- BTF=1,移位和数据寄存器都是满的: DR 寄存器存放了 DataN-2, shift 寄存器存放了 DataN-1 => SCL 拉低: 总线上没有其他要被接收的数据
- 清零 ACK 位
- 读 DR 寄存器中的 DataN-2 => 这将启动移位寄存器对 DataN 的接收
- DataN 接收完成 (发送 NACK 条件)
- 写 START 或者 STOP 位
- 读 DataN-1

- RxNE=1
- 读 DataN

以上流程是针对  $N > 2$  的描述。1个字节和2个字节的接收，要用不同的处理方式，参见以下描述：

#### ■ 2个字节接收的情况

- 置位 POS 和 ACK 位
- 等待 ADDR 置位
- 清零 ADDR 位
- 清零 ACK 位
- 等待 BTF 被置位
- 写 STOP 位
- 读 DR 两次

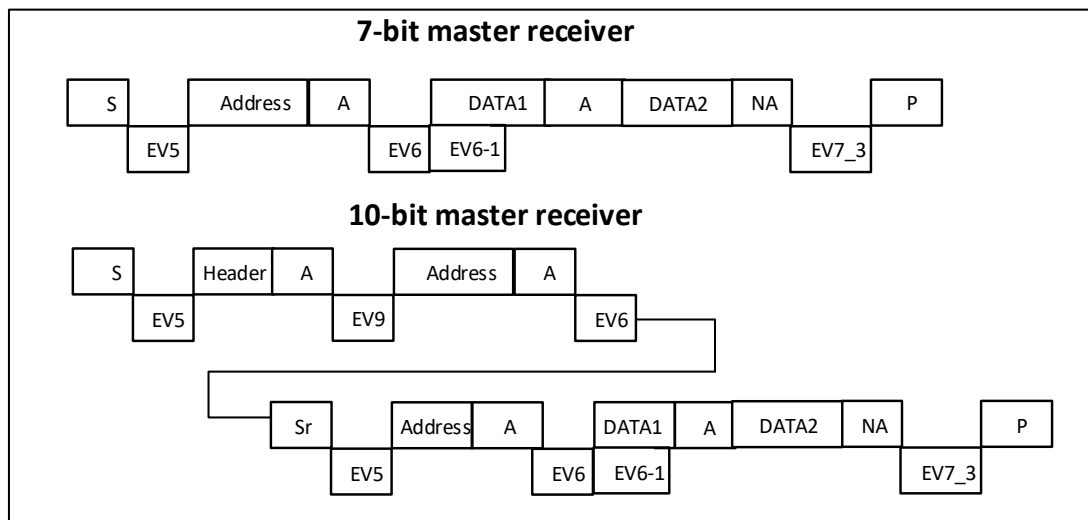


图 30-8 方法2: N=2时主模式发送时的时序

**Legend:** S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (ITEVFEN= 1时产生中断)

**EV5:** SB=1, 先读 SR1寄存器, 再写 DR 寄存器, 清零该位

**EV6:** ADDR=1, 先读 SR1寄存器, 后读 SR2寄存器, 清零 ADDR 位

**EV6\_1:** 无相关的标志位事件。在 EV6后, 也就是地址被清零后, ACK 应该被清零

**EV7\_3:** BTF=1, 写 STOP=1, 之后读两次 DR (Data1和 Data2)

**EV9:** ADDR10=1, 读 SR1然后写 DR 寄存器清除该事件**注:**

- EV5, EV6事件, 拉长 SCL 的低电平, 直到相应的软件序列执行结束
- EV6\_1的软件序列必须在当前字节传输的 ACK 之前完成

#### ■ 单个字节接收的情况

- 在 ADDR 事件里, 清零 ACK 位
- 清零 ADDR
- 写 STOP 或者 START 位
- 在 RxNE 标志置位后, 读数据

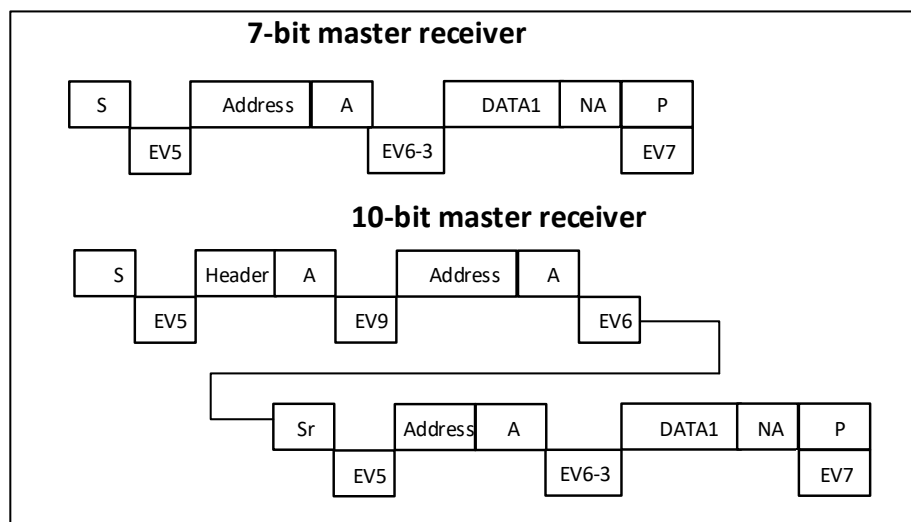


图 30-9 方法2: N=1时主模式发送时的时序

**Legend:** S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (当 ITEVFEN= 1时产生中断)

**EV5:** SB=1, 先读 SR1寄存器, 再写 DR 寄存器, 清零该位

**EV6\_3:** ADDR=1, 写 ACK=0。先读 SR1寄存器, 后读 SR2寄存器, 清零 ADDR 位。在 ADDR 被清零后, 写 STOP=1

**EV7:** RxNE=1, 读 DR 寄存器清零该位

**EV9:** ADDR10=1, 读 SR1然后写 DR 寄存器清除该事件

**注:**

EV5, EV6, EV8\_1和 EV8\_2事件会拉长 SCL 的低电平, 直到相应的软件序列执行结束。

### 30.3.6. 错误状态

#### 30.3.6.1. 总线错误 (BERR)

在一个地址或数据字节传输期间, 当 I<sup>2</sup>C 接口检测到一个外部的停止或起始条件则产生总线错误。此时:

- BERR 位被置位为'1'; 如果设置了 ITERREN 位, 则产生一个中断;
- 在从模式: 数据被丢弃, 硬件释放总线:
  - 如果是错误的起始条件, slave 认为是一个重新开始, 并等待地址或停止条件
  - 如果是错误的停止条件, slave 按正常的停止条件操作, 同时硬件释放总线
- 在主模式: 硬件不释放总线, 同时不影响当前的传输状态。此时由软件决定是否要中止当前的传输。

#### 30.3.6.2. 应答失败 (AF)

当接口检测到一个无应答位时, 产生应答错误。此时:

- AF 位被置位, 如果设置了 ITERREN 位, 则产生一个中断
- 当发送器接收到一个 NACK 时, 必须复位通讯:
  - 如果是处于从模式, 硬件释放总线。
  - 如果是处于主模式, 软件必须生成一个停止条件或者重新开始。

#### 30.3.6.3. 仲裁丢失 (ARLO)

当 I<sup>2</sup>C 接口检测到仲裁丢失时产生仲裁丢失错误, 此时:

- ARLO 位被硬件置位, 如果设置了 ITERREN 位, 则产生一个中断

- I<sup>2</sup>C 接口自动回到从模式 (MSL 位被清除)。当 I<sup>2</sup>C 接口丢失了仲裁, 则它无法在同一个传输中响应它的从地址, 但它可以在接收到主机发送重新开始条件之后响应
- 硬件释放总线

#### 30.3.6.4. 过载 overrun/欠载 underrun (OVR)

在从模式下, 如果禁止时钟延长, I<sup>2</sup>C 接口正在接收数据时, 当它已经接收到一个字节 (RxNE=1), 但在 DR 寄存器中前一个字节数据还没有被读出, 则发生过载错误。

此时:

- 最后接收的数据被丢弃
- 在过载错误时, 软件应清除 RxNE 位, 发送器应该重新发送最后一次发送的字节

在从模式下, 如果禁止时钟延长, I<sup>2</sup>C 接口正在发送数据时, 在下一个字节的时钟到达之前, 新的数据还未写入 DR 寄存器 (TxNE=1), 则发生欠载错误。此时:

- 在 DR 寄存器中的前一个字节将被重复发出
- 用户应该确定在发生欠载运行错误时, 接收端应丢弃重复接收到的数据。发送端应按 I<sup>2</sup>C 总线标准在规定的更新 DR 寄存器

在发送第一个字节时, 必须在清除 ADDR 之后且在第一个 SCL 上升沿之前写入 DR 寄存器; 如果不能做到这点, 则接收方应该丢弃第一个数据。

#### 30.3.7. SDA/SCL 控制

- 如果允许时钟延长:
  - 发送器模式: 如果 TxNE=1且 BTF=1: I<sup>2</sup>C 接口在传输前保持时钟线为低, 以等待软件读取 SR1, 然后把数据写进数据寄存器 (DR 和移位寄存器都是空的)。
  - 接收器模式: 如果 RxNE=1且 BTF=1: I<sup>2</sup>C 接口在接收到数据字节后保持时钟线为低, 以等待软件读 SR1, 然后读数据寄存器 DR (DR 和移位寄存器都是满的)。
- 如果在从模式中禁止时钟延长:
  - 如果 RxNE=1, 在接收到下个字节前 DR 还没有被读出, 则发生过载。接收到的最后一个字节丢失。
  - 如果 TxNE=1, 在必须发送下个字节之前却没有新数据写进 DR, 则发生欠载。相同的字节将重复发出。
  - 硬件未实现对重复写冲突的控制。

#### 30.3.8. DMA 请求

DMA 请求 (当被使能时) 仅用于数据传输。发送时数据寄存器变空, 或接收时数据寄存器变满, 则产生 DMA 请求。DMA 必须在当前字节传输结束之前被初始化和使能。DMAEN 位 (I2C\_CR2寄存器中) 必须在 ADDR 事件发生前使能。

在主机或者从机模式, 当时钟延长功能使能, DMAEN 位可以在清零 ADDR 之前的 ADDR 事件期间置位。DMA 请求必须在当前字节传输完成之前响应。当 DMA 传输数据长度达到 DMA 设定的值时, DMA 向 I<sup>2</sup>C 发送 EOT (传输结束), 并产生传输完成中断 (如果中断使能位有效):

- 主机发送: 在 EOT 中断服务程序中, 需禁止 DMA 请求, 然后在等到 BTF 事件后, 置位停止条件。
- 主机接收: 当要接收的数据数目大于或等于 2 时, DMA 发送一个硬件信号 EOT\_1, 它对应 DMA 传输 (字节数 - 1)。如果在 I2C\_CR2寄存器中设置了 LAST 位, 硬件在发送完 EOT\_1后的下一个字节, 将自动发送 NACK。在中断允许的情况下, 用户可以在 DMA 传输完成的中断服务程序中产生一个停止条件。



### 30.3.8.1. DMA 发送

通过置位 I2C\_CR2寄存器中的 DMAEN 位，可以使能 DMA 模式。只要 TxE 位被置位，数据将由 DMA 从预置的存储区，装载进 I2C\_DR 寄存器。为 I<sup>2</sup>C 分配一个 DMA 通道，须执行以下步骤（x 是通道号）：

1. 在 DMA\_CPARx 寄存器中设置 I2C\_DR 寄存器地址。数据将在每个 TxE 事件后，从存储器传送至这个地址。
2. 在 DMA\_CMARx 寄存器中设置存储器地址。数据在每个 TxE 事件后从这个存储区传送至 I2C\_DR。
3. 在 DMA\_CNDTRx 寄存器中设置所需的传输字节数。在每个 TxE 事件后，此值将被递减。
4. 利用 DMA\_CCRx 寄存器中的 PL[0: 1]位配置通道优先级。
5. 设置 DMA\_CCRx 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置 DMA\_CCRx 寄存器上的 EN 位激活通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I<sup>2</sup>C 接口发送一个传输结束的 EOT/EOT\_1信号。在中断允许的情况下，将产生一个 DMA 中断。

注：如果使用 DMA 进行发送时，不要设置 I2C\_CR2寄存器的 ITBUFEN 位。

### 30.3.8.2. DMA 接收

通过设置 I2C\_CR2寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2C\_DR 寄存器的数据传送到设置的存储区（参考 DMA 说明）。设置 DMA 通道进行 I<sup>2</sup>C 接收，须执行以下步骤（x 是通道号）：

1. 在 DMA\_CPARx 寄存器中设置 I2C\_DR 寄存器的地址。数据将在每次 RxNE 事件后从此地址传送到存储区。
2. 在 DMA\_CMARx 寄存器中设置存储区地址。数据将在每次 RxNE 事件后从 I2C\_DR 寄存器传送到此存储区。
3. 在 DMA\_CNDTRx 寄存器中设置所需的传输字节数。在每个 RxNE 事件后，此值将被递减。
4. 用 DMA\_CCRx 寄存器中的 PL[0: 1]配置通道优先级。
5. 清除 DMA\_CCRx 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置 DMA\_CCRx 寄存器中的 EN 位激活该通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I<sup>2</sup>C 接口发送一个传输结束的 EOT/EOT\_1信号。在中断允许的情况下，将产生一个 DMA 中断。

注：如果使用 DMA 进行接收时，不要设置 I2C\_CR2寄存器的 ITBUFEN 位。

## 30.3.9. SMBus

系统管理总线（SMBus）是一个两线接口。通过它，各设备之间以及设备与系统的其他部分之间可以互相通信。它基于 I<sup>2</sup>C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线（SMBus）标准涉及三类设备。从器件，接收或响应命令的设备。主器件，用来发布命令，产生时钟和终止发送的设备。主机，是一种专用的主器件，它提供与系统 CPU 的主接口。主机必须具有主-从器件功能，并且必须支持 SMBus 通报协议。在一个系统里只允许有一个主机。

本项目的 I<sup>2</sup>C1模块支持 SMBus/PMbus 功能

**30.3.9.1. SMBus 和 I<sup>2</sup>C 之间的相似点**

- 1) 2 条线的总线协议 (1 个时钟, 1 个数据) + 可选的 SMBus 提醒线
- 2) 主-从通信, 主设备提供时钟
- 3) 多主机功能
- 4) SMBus 数据格式类似于 I<sup>2</sup>C 的 7 位地址格式

**30.3.9.2. SMBus 和 I<sup>2</sup>C 之间的不同点**

下表列出来 SMBus 和 I<sup>2</sup>C 的不同点:

表 30-1 SMBus 与 I<sup>2</sup>C 的比较

SMBus	I <sup>2</sup> C
最大传输速度100 kHz	最大传输速度400 kHz
最小传输速度10 kHz	无最小传输速度
35 ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 V <sub>DD</sub> 决定
不同的地址类型 (保留、动态等)	7位、10位和广播呼叫从地址类型
不同的总线协议 (快速命令、处理呼叫等)	无总线协议

**30.3.9.3. SMBus 应用用途**

利用系统管理总线, 设备可提供制造商信息, 告诉系统它的型号/部件号, 保存暂停事件的状态, 报告不同类型的错误, 接收控制参数, 和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

**30.3.9.4. 地址解析协议 (ARP)**

SMBus 从地址冲突可以通过给每个从设备动态分配一个新的唯一地址来解决。

ARP 有以下属性:

- 地址分配利用标准 SMBus 物理层仲裁机制
- 当设备维持供电期间, 分配的地址仍保持不变, 允许设备在断电时保留其地址。
- 在地址分配后, 没有额外的 SMBus 的打包开销 (也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)。
- 任何一个 SMBus 主设备可以遍历总线。

**30.3.9.5. SMBus 提醒模式 (ALERT)**

SMBus 提醒是一个带中断线的可选信号, 用于那些希望扩展他们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL 和 SDA 信号一样, 是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。

单一的从设备可以通过 SMBALERT 发信号给主机表示它希望进行通信, 这可通过设置 I2C\_CR1 寄存器上的 ALERT 位实现。主机处理该中断并通过提醒响应地址 ARA (Alert Response Address, 地址值为 0001100x) 访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2C\_SR1 寄存器中的 SMBALERT 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上, 第八个位可以是 0 或 1。

如果多个设备把 SMBALERT 拉低，最高优先级设备（最小的地址）将在地址传输期间通过标准仲裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。没有执行 SMBALERT 信号的主机可以定期访问 ARA。有关 SMBus 提醒模式的更多详细资料，请参考 2.0 版的 SMBus 规范。

#### 30.3.9.6. 总线协议

SMBus 技术规范支持9个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考2.0版的 SMBus 规范。这些协议由用户的软件来执行。

#### 30.3.9.7. 超时错误 (TIMEOUT)

在定时规范上 I<sup>2</sup>C 和 SMBus 之间有很多差别。

SMBus 定义一个时钟低超时，35ms 的超时。SMBus 规定 TLOW: SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW: MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范。

I2C\_SR1 中的状态标志 Timeout 表明了这个特征的状态。

#### 30.3.9.8. PMBus

PMbus 是基于 SMBus 而来的，传输逻辑与 SMBus 完全一致，区别在于 PMbus 定义了一些与电源管理相关的功能（由软件完成）。

## 30.4. I<sup>2</sup>C 中断

表 30-30 I<sup>2</sup>C 中断请求

中断事件	事件标志	开启控制位
起始位已发送 (Master)	SB	ITEVTEN
地址已发送 (Master) 或 地址匹配 (Slave)	ADDR	
10位头段已发送	ADDR10	
已收到停止 (Slave)	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVTEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失 (Master)	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

## 30.5. I<sup>2</sup>C 寄存器

寄存器可以 half-word 或者 word 访问。

30.5.1. I<sup>2</sup>C 控制寄存器1 (I2C\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRS T	Re s	ALER T	PE C	PO S	AC K	STO P	STAR T	NO STRETC H	ENG C	ENPE C	ENAR P	SMB TYP E	Re s	SMBU S	P E
RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	RW	

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	软件复位。 当被置位时，I <sup>2</sup> C 处于复位状态。在复位释放前，要确保 I <sup>2</sup> C 的引脚被释放，总线是空闲状态。 0: I <sup>2</sup> C 模块不处于复位状态 1: I <sup>2</sup> C 模块处于复位状态 注：该位可以用于错误或锁住状态时重新初始化 I <sup>2</sup> C。如 BUSY 位为1，在总线上又没有检测到停止条件时。
14	保留	-	-	保留
13	ALERT	RW	0	SMBus 提醒。 0: 释放 SMBAlert 引脚使其变高。提醒响应地址头后跟 NACK 1: 驱动 SMBAlert 引脚使其变低。提醒响应地址头后跟 ACK PE=0时，由硬件清除。
12	PEC	RW	0	数据包出错检查 软件可置位和清零该位，硬件可以在以下情况下清零该位：当传送 PEC 后，起始条件、或者停止条件、或者当 PE=0时； 0: 无 PEC 传输 1: PEC 传输（在发送或接收模式） 注：仲裁丢失时，PEC 的计算失效。
11	POS	RW	0	ACK/PEC 位置（用于数据接收），软件可置位/清零该寄存器，或 PE=0时由硬件清零。 0: ACK 位控制当前移位寄存器内正在接收的字节的 (N) ACK。PEC 位表明当前移位寄存器内的字节是 PEC

Bit	Name	R/W	Reset Value	Function
				<p>1: ACK 位控制在移位寄存器里接收的下一个字节的 (N) ACK。PEC 位表明在移位寄存器里接收的下一个字节是 PEC</p> <p>注: POS 位只能用在2字节的接收配置中, 必须在接收数据之前配置。</p> <p>为了 NACK 第2个字节, 必须在清除 ADDR 之后清除 ACK 位。</p> <p>为了检测第2个字节的 PEC, 必须在配置了 POS 位之后, ADDR 拉长事件时设置 PEC 位。</p>
10	ACK	RW	0	<p>应答使能。软件可置位/清零该寄存器, 或 PE=0时由硬件清零。</p> <p>0: 无应答返回</p> <p>1: 在接收到一个字节后返回一个应答。(匹配的地址或数据)</p>
9	STOP	RW	0	<p>停止条件产生, 软件可以置位/清零该寄存器, 或者当检测到停止条件时, 由硬件清除; 当检测到超时错误时, 硬件置位。</p> <p>在主模式下:</p> <p>0: 无停止条件产生</p> <p>1: 在当前字节传输或在当前起始条件发出后产生停止条件</p> <p>在从模式下:</p> <p>0: 无停止条件产生</p> <p>1: 在当前字节传输后释放 SCL 和 SDA 线</p>
8	START	RW	0	<p>起始条件产生。</p> <p>软件可置位/清零该寄存器, 或当起始条件发出后或 PE=0时由硬件清零。</p> <p>主模式:</p> <p>0: 无起始条件产生</p> <p>1: 重复产生起始条件</p> <p>从模式:</p> <p>0: 无起始条件产生</p> <p>1: 当总线空闲时, 产生起始条件 (并由硬件自动切换到主机模式)</p>
7	NOSTRETCH	RW	0	<p>禁止时钟延长 (从机)。</p> <p>当 ADDR 或 BTF 标志被置位时, 该位用于从机禁止时钟延长, 直到被软件复位。</p> <p>0: 允许时钟延长</p> <p>1: 禁止时钟延长</p>

Bit	Name	R/W	Reset Value	Function
6	ENGC	RW	0	广播呼叫使能。 0: 禁止广播呼叫。以 NACK 响应地址00h 1: 允许广播呼叫。以 ACK 响应地址00h
5	ENPEC	RW	0	PEC 使能。 0: 禁止 PEC 计算 1: 开启 PEC 计算
4	ENARP	RW	0	ARP 使能。 0: 禁止 ARP; 1: 使能 ARP; 如果 SMBTYPE=0, 使用 SMBus 设备的默认地址; 如果 SMBTYPE=1, 使用 SMBus 的主地址。
3	SMBTYPE	RW	0	SMBus 类型。 0: SMBus 设备 1: SMBus 主机
2	保留	-	-	保留
1	SMBUS	RW	0	SMBus 模式。 0: I <sup>2</sup> C 模式 1: SMBus 模式
0	PE	RW	0	I <sup>2</sup> C 模块使能。 0: 禁止 1: I <sup>2</sup> C 使能 注: 如果清除该位时通讯正在进行, 在当前通讯结束后, I <sup>2</sup> C 模块被禁用并返回空闲状态。 由于在通讯结束后 PE=0, 所有的位被清除。 在主模式下, 通讯结束之前, 绝不能清除该位。

### 30.5.2. I<sup>2</sup>C 控制寄存器 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	LAST	DMAEN	IT- BUFEN	ITEV- TEN	ITER- REN	Res	Res	FREQ[5: 0]					
-	-	-	RW	RW	RW	RW	RW	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	保留	-	-	保留
12	LAST	RW	0	DMA 最后一次传输。 0: 下一次 DMA 的 EOT 不是最后的传输 1: 下一次 DMA 的 EOT 是最后的传输

Bit	Name	R/W	Reset Value	Function
				注：该位在主接收模式使用，使在最后一次接收数据时可以产生一个 NACK。
11	DMAEN	RW	0	DMA 请求使能。 0: 禁止 DMA 请求; 1: 当 TxE=1或 RxNE=1时, 允许 DMA 请求。
10	ITBUFEN	RW	0	缓冲器中断使能。 0: 当 TxE=1或 RxNE=1时, 不产生中断 1: 当 TxE=1或 RxNE=1时, 产生事件中断 (不管 DMAEN 是何值)
9	ITEVTEN	RW	0	事件中断使能。 0: 禁止 1: 允许事件中断 在下列条件下, 将产生该中断: 1) SB=1 (主模式) 2) ADDR=1 (主/从模式) 3) STOPF=1 (从模式) 4) BTF=1, 但没有 TxE 或 RxNE 事件 5) 如果 ITBUFFEN=1, TxE 事件为1 6) 如果 ITBUFEN=1, RxNE 事件为1
8	ITERREN	RW	0	出错中断使能。 0: 禁止出错中断; 1: 允许出错中断; 在下列条件下, 将产生该中断: — BERR=1 — ARLO=1 — AF=1 — OVR=1 — PECERR=1 — TIMEOUT=1 — SMBAlert=1
7: 6	保留	-	-	保留
5: 0	FREQ	RW	0	I <sup>2</sup> C 模块时钟频率。 必须用 APB 时钟频率的值配置该寄存器, 以产生与 I2C 协议兼容的数据 setup 和 hold 时间。 最小允许可设定的频率是4MHz (标准模式, 即 100k)、8MHz (400k), 最大频率是芯片最高的 APB 时钟频率。 000000: 禁止 000001: 禁止

Bit	Name	R/W	Reset Value	Function
				000100: 4 MHz ..... 100100: 36 MHz ..... 110010: 50 MHz 大于100100: 禁止。

### 30.5.3. I<sup>2</sup>C 自身地址寄存器 1 (I2C\_OAR1)

Address offset: 0x08

Reset value: 0x0000 4000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Res					ADD[9: 8]		ADD[7: 1]						ADDR0	
RW	-					RW									

Bit	Name	R/W	Reset Value	Function
15	ADDMODE	RW	0	寻址模式 (从模式)。 0: 7位从地址 (不响应10位地址) 1: 10位从地址 (不响应7位地址)
14:10	保留	-	-	保留
9:8	ADD[9:8]	RW	0	接口地址。 7位地址模式该寄存器无关。 10位地址模式位地址的9~8位。
7:1	ADD[7:1]	RW	0	接口地址的7~1位。
0	ADDR0	RW	0	接口地址。 7位地址模式该寄存器无效。 10位地址模式的地址0位。

### 30.5.4. I<sup>2</sup>C 自身地址寄存器 2 (I2C\_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	OA2MSK[2: 0]			ADD2[7: 1]							ENDUAL
-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	保留	-	-	保留
7:1	ADD2[7:1]	RW	0	接口地址的7~1位。



				双地址模式下地址的7~1位。
0	ENDUAL	RW	0	双地址模式使能位。 0: 在7位地址模式下, 只有 OAR1被识别 1: 在7位地址模式下, OAR1和 OAR2都被识别

### 30.5.5. I<sup>2</sup>C 数据寄存器 (I2C\_DR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7: 0	DR[7: 0]	RW	0	<p>8位数据寄存器, 芯片内部实际是两个独立的 buffer 共用一个地址, 分别用于存放接收到的数据 (RX_DR)、放置要发送到总线的数据 (TX_DR)。</p> <p><b>发送器模式:</b> 当写一个字节至 DR 寄存器时 (实际写到 TX_DR), 自动启动数据传输。一旦传输开始 (TxE=1), 如果能及时把下一个需传输的数据写入 DR 寄存器, I<sup>2</sup>C 模块将保持连续的数据流。</p> <p><b>接收器模式:</b> 接收到的字节被拷贝到 DR 寄存器 (实际是 RX_DR) (RxNE=1)。在接收到下一个字节 (RxNE=1) 之前读出数据寄存器, 即可实现连续的数据接收。</p> <p>注:</p> <ol style="list-style-type: none"> <li>1. 在从模式下, 地址不会被写进数据寄存器 DR</li> <li>2. 硬件不处理写冲突 (如果 TxE=0, 仍能写入数据寄存器)</li> <li>3. 如果在处理 ACK 脉冲时发生 ARLO 事件, 接收到的字节不会被复制到数据寄存器里, 因此不能读到</li> </ol>

### 30.5.6. I<sup>2</sup>C 状态寄存器 (I2C\_SR1)

Address offset: 0x14

Reset value: 0x0000 0000

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

SMBA LERT	TIMEO UT	Res	PECE RR	OV R	AF	AR LO	BER R	Tx E	RxN E	Re s	STOP F	ADD1 0	BT F	ADD R	S B	
RC_W0		-	RC_W0					R	R	-	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15	SMBALERT	RC_W0	0	<p>SMBus 提醒状态。</p> <p><b>SMBus 主机模式:</b></p> <p>0: 无 SMBus 提醒</p> <p>1: 在引脚产生 SMBAlert 提醒事件</p> <p><b>SMBus 从机模式:</b></p> <p>0: 没有 SMBAlert 响应地址头序列</p> <p>1: 收到 SMBAlert 响应地址头序列直到 SMBAlert 变低 该位由软件写0或 PE=0时由硬件清除。</p>
14	TIMEOUT	RC_W0	0	<p>超时或 Tlow 错误。</p> <p>0: 无超时错误;</p> <p>1: SCL 为低的持续时间已达到25ms (超时); 或者主机低电平累计时钟扩展时间超过10ms (Tlow: mext); 或从设备低电平累积时钟扩展时间超过25 ms (Tlow: sext)。</p> <p>当在从模式下设置该位: 从设备复位通讯, 硬件释放总线;</p> <p>当在主模式下设置该位: 硬件发出停止条件;</p> <p>该位由软件写0清除, 或当 PE=0时由硬件清除。</p> <p>注: 该功能仅在 SMBUS 模式有作用。</p>
13	保留	-	-	保留
12	PECERR	RC_W0	0	<p>在接收时发生 PEC 错误。</p> <p>0: 无 PEC 错误, 接收到 PEC 后返回 ACK (如果 ACK=1)</p> <p>1: 有 PEC 错误, 接收到 PEC 后返回 NACK (不管 ACK 为何值)</p> <p>该位由软件写0清除, 或当 PE=0时由硬件清除。</p>
11	OVR	RC_W0	0	<p>过载/欠载标志。</p> <p>0: 无过载/欠载;</p> <p>1: 出现过载/欠载。</p> <p>当 NOSTRETCH=1时, 在从模式下该位被硬件置位;</p> <p>在接收模式中当收到一个新的字节时 (包括 ACK 应答脉冲), 数据寄存器里的内容还未被读出, 则新接收的字节将丢失。</p> <p>在发送模式中当要发送一个新的字节时, 却没有新的数据写入数据寄存器, 同样的字节将被发送两次。</p>

Bit	Name	R/W	Reset Value	Function
				该位由软件写0清除，或当 PE=0时由硬件清除。 注：如果数据寄存器的写操作发生时间非常接近 SCL 的上升沿，发送的数据是不确定的，并发生保持时间错误。
10	AF	RC_W0	0	应答失败标志。 0：没有应答失败； 1：应答失败。 当没有返回应答时，硬件将置位该寄存器。 该位由软件写0清除，或当 PE=0时由硬件清除。
9	ARLO	RC_W0	0	仲裁丢失（主模式）。 0：没有检测到仲裁丢失； 1：检测到仲裁丢失。 当接口失去对总线的控制给另一个主机时，硬件将置位该寄存器。 该位由软件写0清除，或在 PE=0时由硬件清除。 在 ARLO 事件之后，I <sup>2</sup> C 接口自动切换回从模式 (MSL=0)。
8	BERR	RC_W0	0	总线出错标志。 0：无起始或者停止条件出错； 1：起始或者停止条件出错。 当接口检测到错误的起始或者停止条件，硬件将该位置1。 该位由软件写0清除，或者在 PE=0时由硬件清除。
7	TxE	R	0	数据寄存器为空（发送时）标志。 0：数据寄存器非空； 1：数据寄存器为空。 在发送数据时，数据寄存器为空时该位被置1，在发送地址阶段不设置该位。 软件写数据到 DR 寄存器可清除该位，或在发生一个起始或停止条件后，或当 PE=0时由硬件自动清除。 如果收到一个 NACK，或下一个要发送的字节时 PEC (PEC=1)，该位不被置位。 注：在写入第1个要发送的数据后，或设置了 BTF 时写入数据，都不能清除 TxE 位，因为此时数据寄存器为空。
6	RxNE	R	0	数据寄存器非空（接收时）标志。 0：数据寄存器为空； 1：数据寄存器非空。 在接收时，当数据寄存器不为空，置位该寄存器。在接收地址阶段，该寄存器不置位。

Bit	Name	R/W	Reset Value	Function
				软件对数据寄存器的读写操作会清除该寄存器，或当 PE=0 时由硬件清除。 注：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为此时数据寄存器仍为满。
5	保留	-	-	保留
4	STOPF	R	0	停止条件检测位（从模式）。 0：没有检测到停止位； 1：检测到停止条件。 在一个应答之后（如果 ACK=1），当从设备在总线上检测到停止条件时，硬件将该位置1。 软件读取 I2C_SR1 寄存器后，对 I2C_CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。
3	ADD10	R	0	10位地址头序列已发送（主模式）。 0：没有 ADD10 事件发生。 1：主设备已经将第一个地址字节发送出去。 在10位地址模式下，当主设备将已第一个字节发送出去时，硬件将该位置1。 软件读取 I2C_SR1 寄存器后，对 I2C_DR 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 主：收到 NACK 后，该寄存器不被置位。
2	BTF	R	0	字节传输结束标志位。 0：字节传输未完成 1：字节传输成功结束 在下列情况下硬件将置位该寄存器（当从模式，NOSTRETCH=0 时；主模式，与 NOSTRETCH 无关）： <ul style="list-style-type: none"> <li>✓ 接收时，当收到一个新字节（包括 ACK 脉冲）且数据寄存器还未被读取（RxNE=1）。</li> <li>✓ 发送时，当一个新数据应该被发送，且数据寄存器还未被写入新的数据（TxE=1）。</li> </ul> 软件读取 I2C_SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。 注： 在收到一个 NACK 后，BTF 位不会被置位。
1	ADDR	R	0	地址已被发送（主模式）/地址匹配（从模式）。 软件读取 I2C_SR2 寄存器后，再读 I2C_SR1 寄存器将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。 地址匹配（从机）：

Bit	Name	R/W	Reset Value	Function
				0: 地址不匹配或没有收到地址; 1: 收到的地址匹配。 当收到的从地址与 OAR 寄存器或广播呼叫地址匹配, 或 SMBus 设备默认地址、或 SMBus 主机识别除 SMBus 提醒时, 硬件将置位该位。 注: 在 slave 模式下, 推荐进行完整的清零序列, 即在 ADDR 被置位后, 先读 SR1 寄存器, 再读 SR2 寄存器。 地址已发送 (主机) : 0: 地址发送没有结束; 1: 地址发送结束。 10位地址时, 当收到地址的第二个字节的 ACK 后置位。 7位地址时, 当收到 ACK 后置位。 注: 在收到 NACK 后, 该寄存器不会被置位。
0	SB	R	0	起始位标志 (主模式) 。 0: 未发送起始条件; 1: 起始条件已发送; —当发送起始条件时, 置位该寄存器。 —软件读取 I2C_SR1 寄存器后, 对数据寄存器的读或写操作将清除该位; 或当 PE=0 时, 由硬件清除。

### 30.5.7. I<sup>2</sup>C 状态寄存器 2 (I2C\_SR2)

Address offset: 0x18

Reset value: 0x0000 0000

注: 即使 ADDR 标志位在读 I2C\_SR1 寄存器后被置位, 在读 I2C\_SR1 之后再读 I2C\_SR2 寄存器, 也会清零 ADDR 标志位。因此, 仅在发现 I2C\_SR1 寄存器的 ADDR 位被置位或者 STOPF 位被清零时, I2C\_SR2 寄存器才必须被读。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7: 0]								DUALF	SMBHOST	SMBDE- FAULT	GEN- CALL	Res	TRA	BUSY	MSL
R	R	R	R	R	R	R	R	R	R	R	R	-	R	R	R

Bit	Name	R/W	Reset Value	Function
15: 8	PEC	R	0	数据包出错检测寄存器。 当 ENPEC=1 时, 该寄存器存放内部的 PEC 的值。
7	DUALF	R	0	双地址标志 (从模式) 。 0: 接收到的地址与 OAR1 匹配; 1: 接收到的地址与 OAR2 匹配。

Bit	Name	R/W	Reset Value	Function
				当产生一个停止条件或一个重复的起始条件时，或 PE=0时，硬件清除该寄存器。
6	SMBHOST	R	0	接收到 SMBus 主机头序列（从模式）标志。 0：未收到 SMBus 主机的地址； 1：当 SMBTYPE=1且 ENARP=1时，收到 SMBus 主机地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0时，硬件清除该寄存器。
5	SMBDEFAULT	R	0	SMBus 从设备默认地址（从模式）。 0：未收到 SMBus 设备的默认地址； 1：当 ENARP=1时，收到 SMBus 设备的默认地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0时，硬件清除该寄存器。
4	GENCALL	R	0	广播呼叫地址（从模式）。 0：未收到广播呼叫地址； 1：当 ENGC=1时，收到广播呼叫的地址。 当产生一个停止条件或一个重复的起始条件时，或 PE=0时，硬件清除该寄存器。
3	保留	-	-	保留
2	TRA	R	0	发送/接收标志。 0：接收到数据 1：数据已发送 在整个地址传输阶段的结尾，该寄存器根据地址字节的 R/W 位来设定。 当检测到停止条件（STOPF=1），或者重复的起始条件、或者总线仲裁丢失（ARLO=1），或当 PE=0时，硬件清除该寄存器。
1	BUSY	R	0	总线忙标志。 0：在总线上无数据通讯 1：在总线上正在进行数据通讯 当检测到 SDA 或 SCL 为低电平时，硬件置位。 当检测到一个停止条件时，硬件清零。 该寄存器指示当前正在进行的总线通讯，当接口被禁用（PE=0）时该信息仍然被更新。
0	MSL	R	0	主从模式。 0：从模式 1：主模式 —当接口处于主模式（SB=1）时，硬件置位；

Bit	Name	R/W	Reset Value	Function
				—当总线上检测到一个停止条件 (STOPF=1)、仲裁丢失 (ARLO=1)、或当 PE=0时, 硬件清零。

### 30.5.8. I<sup>2</sup>C 时钟控制寄存器 (I2C\_CCR)

Address offset: 0x1C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Res	Res	CCR[11: 0]											
RW	RW	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	I <sup>2</sup> C 主模式选择。 0: 标准模式 1: 快速模式
14	DUTY	RW	0	快速模式时的占空比。 0: 快速模式下: $T_{low}/T_{high}=2$ 1: 快速模式下: $T_{low}/T_{high}=16/9$
13: 12	保留	-	-	保留
11: 0	CCR[11: 0]	RW	0	快速/标准模式下的时钟控制分频系数 (主模式)。 该分频系数用于设置主模式下的 SCL 时钟。 1. 标准模式: <ul style="list-style-type: none"> <li>■ <math>T_{high}=CCR \times T_{pclk}</math></li> <li>■ <math>T_{low}=CCR \times T_{pclk}</math></li> </ul> 2. 快速模式: <ul style="list-style-type: none"> <li>■ DUTY=0:  <math>T_{high}=CCR \times T_{pclk}</math>  <math>T_{low}=2 \times CCR \times T_{pclk}</math> </li> <li>■ DUTY=1 (为达到400KHz):  <math>T_{high}=9 \times CCR \times T_{pclk}</math>  <math>T_{low}=16 \times CCR \times T_{pclk}</math> </li> </ul> 注: <ul style="list-style-type: none"> <li>■ 允许设定的最小值为 0x04, 在快速 DUTY 模式下允许的最小值为 0x01</li> <li>■ <math>T_{high}=t_{r(SCL)} + t_w(SCLH)</math></li> <li>■ <math>T_{low}=t_{r(SCL)} + t_w(SCLL)</math></li> <li>■ 这些延时没有过滤器</li> <li>■ 只有当 PE=0时才能配置该寄存器;</li> </ul>

### 30.5.9. I<sup>2</sup>C TRISE 寄存器 (I2C\_TRISE)

Address offset: 0x20

Reset value: 0x0000 0002

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	TRISE[5: 0]					
-	-	-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 6	保留	-	-	保留
5: 0	TRISE	RW	0x0002	<p>在快速/标准模式下的最大上升时间（主模式）。这些位应该提供在主机模式下，SCL 反馈回路的最大持续时间。这样做的目的是无论 SCL 上升沿持续时间多少，SCL 都能保持一个稳定的频率。这些位必须设置为 I<sup>2</sup>C 总线规范里给出的最大的 SCL 上升时间，增长步幅为1。</p> <p>例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CR2寄存器中 FREQ[5: 0] 中的值等于0x08，Tpclk=125ns，则 TRISE 中配置为0x09 (1000ns/125ns = 8 + 1 = 9)。</p> <p>滤波器的值也可以加到 TRISE 内。</p> <p>如果结果不为整数，则将整数部分写入 TRISE，以确保 t<sub>HIGH</sub> 参数。</p> <p>注：当 PE=0时才能设置该寄存器。</p>



## 31. 串行外设接口 (SPI)

本项目设计实现了 2 个 SPI 模块，两者的功能完全一样。

### 31.1. 简介

SPI 接口可以配置为支持 SPI 协议或者支持 I<sup>2</sup>S 音频协议。SPI 接口默认工作在 SPI 方式，可以通过软件把功能从 SPI 模式切换到 I<sup>2</sup>S 模式。

串行外设接口 (SPI) 允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟 (SCK)。接口还能以多主配置方式工作。它可用于多种用途，包括使用一条双向数据线的双线单工同步传输，还可使用 CRC 校验的可靠通信。

I<sup>2</sup>S 也是一种 3 引脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦 I<sup>2</sup>S 标准，MSB 和 LSB 对齐标准，以及 PCM 标准。它在半双工通讯中，可以工作在主和从 2 种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

### 31.2. SPI 主要特征

#### 31.2.1. SPI 主要特征

- 支持 SPI 主机模式和 SPI 从机模式
- 3 线全双工同步传输
- 2 线半双工同步传输 (有双向数据线)
- 2 线单工同步传输 (无双向数据线)
- 8 位或者 16 位传输帧选择
- 支持多主模式
- 8 个主模式波特率预分频系数 (最大为  $f_{PCLK}/2$ )
- 从模式频率 (最大为  $f_{PCLK}/4$ )
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持可靠通讯硬件 CRC
  - 在发送模式下，CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 中断标志：主模式故障、过载、CRC 错误
- 2 个具备 DMA 能力的深度为 4，宽度为 16 位 (当数据帧设置为 8 位时，宽度为 8 位) 的 Rx 和 Tx FIFO

### 31.2.2. I<sup>2</sup>S 主要特征

- 半双工通讯 (仅发送或接收)
- 主或者从操作
- 8 位线性可编程预分频器, 获得精确的音频采样频率 (8 kHz ~ 96 kHz)
- 数据格式可以是 16 位, 24 位或者 32 位
- 音频信道固定数据包格式为 16 位 (16 位数据帧) 或 32 位 (16 位、24 位 32 位数据帧)
- 可编程的时钟极性 (稳定态)
- 从发送模式下的下溢标志位和主/从接收模式下的溢出标志位
- 16 位数据寄存器用来发送和接收, 在通道两端各存在一个寄存器
- 支持 I<sup>2</sup>S 协议:
  - I<sup>2</sup>S 飞利浦标准
  - MSB 对齐标准 (左对齐)
  - LSB 对齐标准 (右对齐)
  - PCM 标准 (16 位通道帧上带长帧或者短帧同步或者 16 位数据帧扩展为 32 位通道帧)
- 数据方向总是 MSB 在前
- 发送和接收都有 DMA 能力 (16 位宽)
- 主时钟可以输出到外部音频设备, 比率固定为  $256 \times f_s$  ( $f_s$  为音频采样频率)

## 31.3. SPI 功能描述

### 31.3.1. 概述

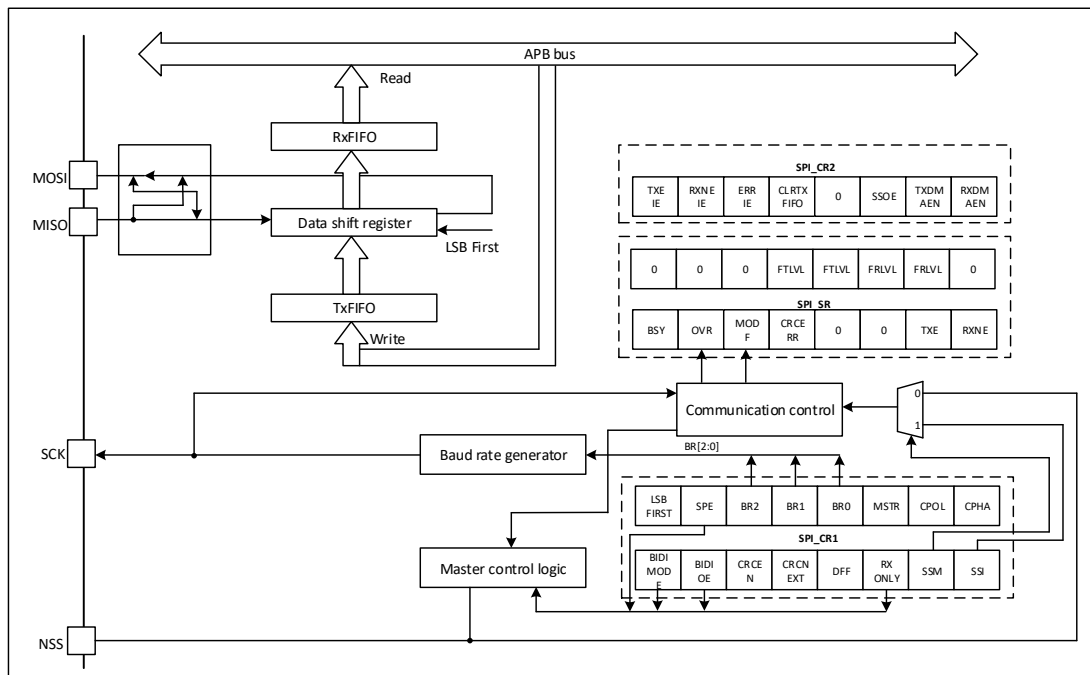


图 31-1 SPI 框图

SPI 通过 4 个引脚与外部器件相连：

**MISO**：主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。

**MOSI**：主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。

**SCK**：串口时钟，作为主设备的输出，从设备的输入。

**NSS**：从设备选择。取决于 SPI 和 NSS 的设置，该引脚可以用作：

- 选择要通讯的从机
- 同步数据帧
- 检测多主机间的冲突

SPI 总线允许在一个主机和一个或者多个从机之间的通讯。总线由至少两根线组成：一个是时钟，另一个是用于同步传输的数据。根据应用场景，可以选择增加另外一根数据线和从机 NSS 信号。

### 31.3.2. 单主机和单从机通信

针对不同的应用场景，SPI 可以使用几种不同的配置进行通讯。这些配置使用 2 线、3 线（软件 NSS）或者 4 线（硬件 NSS）。通讯始终由主机启动。

#### 31.3.2.1. 全双工通信

默认情况，SPI 被配置成全双工通讯。在这种配置下，主机和从机的移位寄存器，在 MOSI 和 MISO 之间，使用两个单向的线连到一起。在 SPI 通讯期间，数据在主机提供的时钟沿同步移位。主机通过 MOSI 发送数据，从 MISO 接收来自从机的数据。当数据帧传输完成（所有位均移出），主机和从机之间完成信息交换。

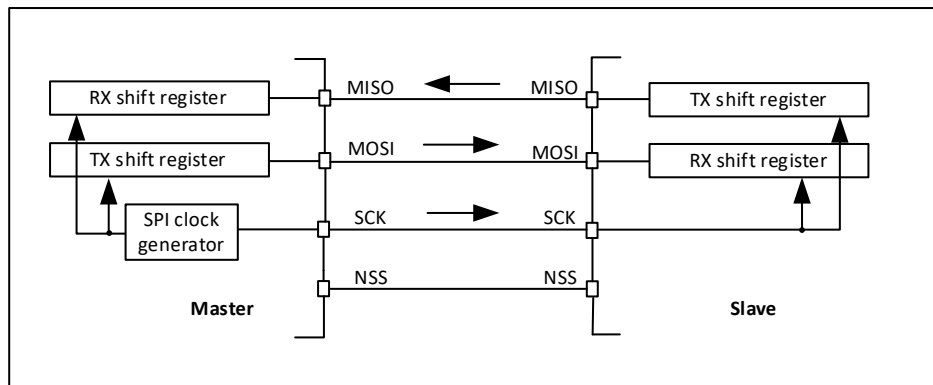


图 31-2 全双工单主机/单从机应用

### 31.3.2.2. 半双工通信

通过设定 BIDIMODE 位（SPI\_CR1 寄存器），SPI 可以工作在半双工模式。在这种配置下，用 1 根数据线完成主机和从机 移位寄存器的连接。在通讯过程中，在 SCK 的时钟沿，数据在两个移位寄存器之间以 BIDIOE（SPI\_CR1 寄存器）选择的方向，同步移位。在该配置下，主机的 MISO 和从机的 MOSI 被释放作为通用端口给其他应用使用。

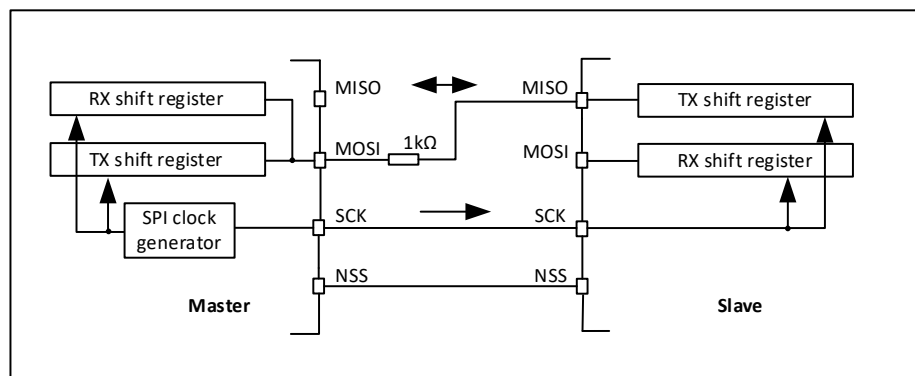


图 31-3 半双工单主机/单从机应用

NSS 可以被使用在主机和从机之间进行硬件控制流。可选的，NSS 也可以不使用，然后该流程就要内部处理。

在该配置下，主机的 MISO 引脚和从机的 MOSI 引脚可以用作 GPIO。

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。

### 31.3.2.3. 单工通信

通过使用 RXONLY（SPI\_CR1 寄存器），设定 SPI 在只发送模式或者只接收模式，使 SPI 工作在单工模式下。在这个配置下，在主机和从机的移位寄存器之间只使用 1 根线。另一对 MISO 和 MOSI 不被使用，可以被用作标准的 GPIO。

- 只发送模式 (RXONLY=0) : 配置与全双工相同。应用忽略在未使用的端口上的信息。这个端口可以被用作标准的 GPIO。
- 只接收模式 (RXONLY=1) : 通过置位 RXONLY, 应用可以禁止 SPI 输出功能。在从机配置, MISO 输出被禁能, 该端口被用作 GPIO。当他的从机 NSS 信号有效时, 从机继续从 MOSI 接收数据。接收到的数据事件是否发生取决于数据缓冲区的配置。在主机配置下, MOSI 输出被禁止, 该端口可以用作 GPIO。只要 SPI 被使用, 时钟信号就被连续的产生。停止时钟的唯一方法是清零 RXONLY 或者 SPE, 直到来自 MISO 的输入完成。并然后基于相应配置填入数据缓冲区。

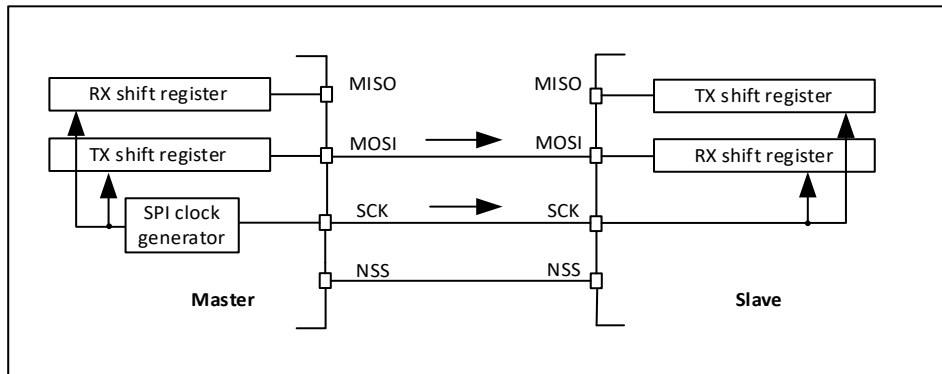


图 31-4 单工单从机/单主机应用

(主设备处于仅发送模式/从设备处于仅接收模式)

(1) 在主机和从机之间可以使用 NSS 进行硬件控制流。可选的, NSS 也可以不使用。然后该流程就要内部处理。

(2) 在 Rx 移位寄存器的输入上捕获意外的输入信息。在标准的只发送模式下, 所有与传输接收相关的事件都被必须被忽略。

(3) 在该配置下, 两边的 MISO 引脚都被用作 GPIO。

通过用传输方向的设定 (在 BIDIOE 位未发生改变时, 双向模式被使能), 任何单工通讯可以被半双工通讯代替。

### 31.3.3. 多从机通信

在具有两个或者更多个独立从机的配置里，主机为每个从机，使用 GPIO 来管理 NSS。主机必须通过拉低连接从机 NSS 来选择某个从机。当完成这个，标准的主机和专门的从机通讯就被建立了。

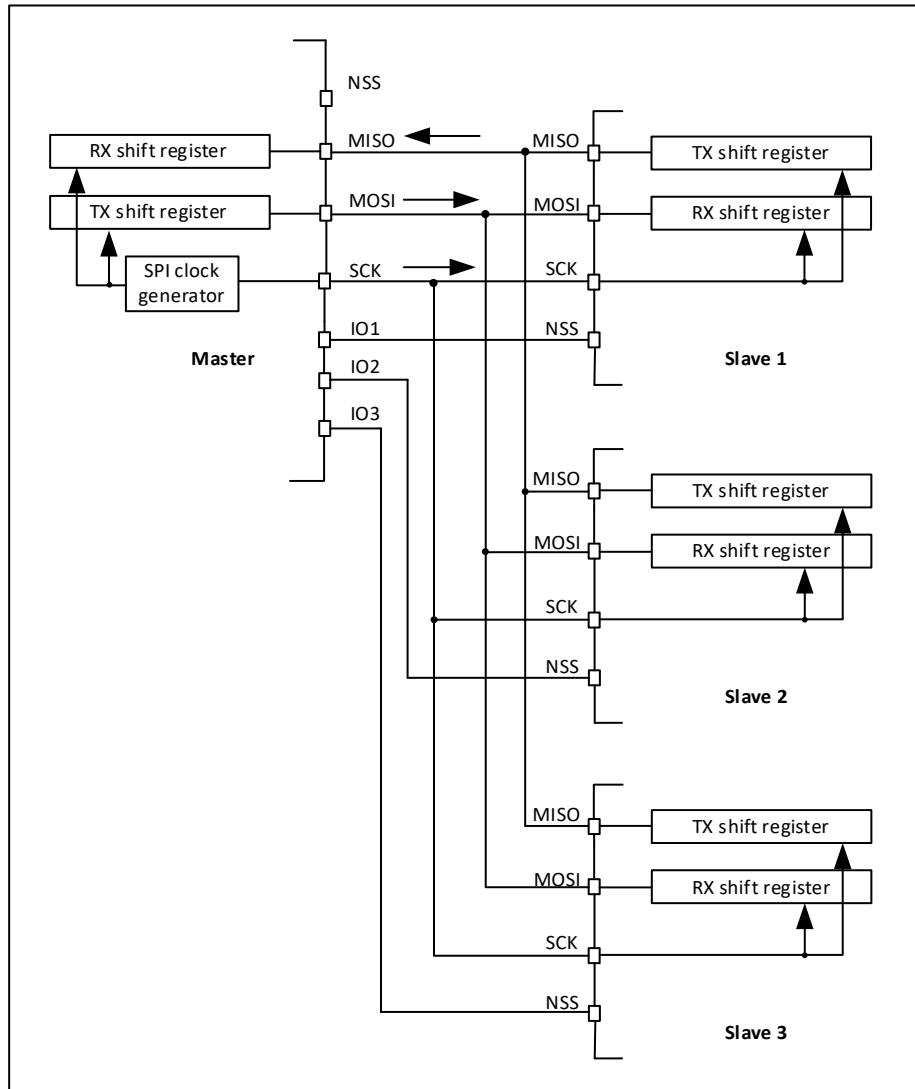


图 31-5 主机与三个独立的从机通信

这种配置下主机不使用 NSS 引脚。必须通过 SSM=1, SSI=1 来防止任何 MODF 错误。

由于从机的 MISO 连接到一起，所有从机必须把他们 MISO 的 GPIO 配置作为开漏模式。

### 31.3.4. 多主机通信

除非 SPI 总线不是被设计成具备多主机功能，否则用户可以使用其内嵌功能，该功能可以发现两个试图同时控制总线的节点存在的潜在冲突。当需要用到这种检测时，要使用配置为硬件输入模式的 NSS 引脚。

在这种模式下有两个以上 SPI 节点工作的连接是不可能的，因为单次只有一个节点可以在公共数据线上传输。

当节点无效，缺省下两个都保持从机模式。一旦节点要控制总线，它自己切换到主机模式，然后通过专用 GPIO 引脚向其他节点的从器件 NSS 引脚输入施加有效电平。在该进程完成后，有效的从机选择信号被释放，控制总线的节点暂时返回被动从模式，并等待新的进程开始。

如果两个节点在同一时间都给出控制请求，总线冲突事件就会产生（查看 MODF 事件）。然后用户可以应用一些简单的仲裁过程（例如：在两个节点上施加不同的预定义超时来推迟下一次尝试）

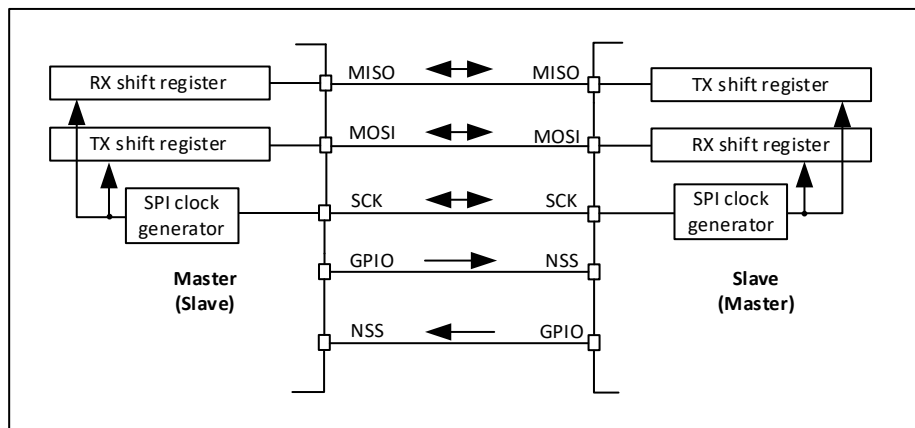


图 31-6 多主机应用

NSS 在两个节点都被配置成硬件输入模式。当被动节点被配置成从机时，其有效电平将使能 MISO 输出控制。

### 31.3.5. 从选择 (NSS) 脚管理

在从机模式，NSS 作为标准的片选输入，使从机能与主机通讯。在主机模式，NSS 既可以作为输出又可以作为输入。当作为输入时，它可以防止多主机的总线冲突，当作为输出时，它可以驱动单个从机的从机选择信号。

通过 SPI\_CR1 寄存器的 SSM 位，可以选择硬件或者软件从机 NSS 管理：

- 软件 NSS 管理 (SSM=1)：在这个配置下，从机选择信号被内部的 SSI 位 (SPI\_CR1 寄存器) 驱动。外部 NSS 引脚被释放给其他应用使用。
- 硬件 NSS 管理 (SSM=0)：在这个情况下，有几个可能的配置。
  - 1) NSS 输出使能 (SSM=0, SSOE=1)：这个配置仅在作为主机时使用。硬件管理 NSS 引脚。当 SPI 在主机模式被使能 (SPE=1)，NSS 信号就被拉低并保持低电平，直到 SPI 被禁止 (SPE=0)。在多主机应用中，SPI 不能进行这种 NSS 配置。
  - 2) NSS 输出禁止 (SSM=0, SSOE=0)：如果 MCU 在总线上作为主机，这个配置可实现多主机功能。如果 NSS 引脚此时被拉低，SPI 进入主机故障状态，芯片自动被重配置为从机模式。在从机模式，NSS 引脚作为标准的片选输入，当 NSS 为低时，从机被选中。

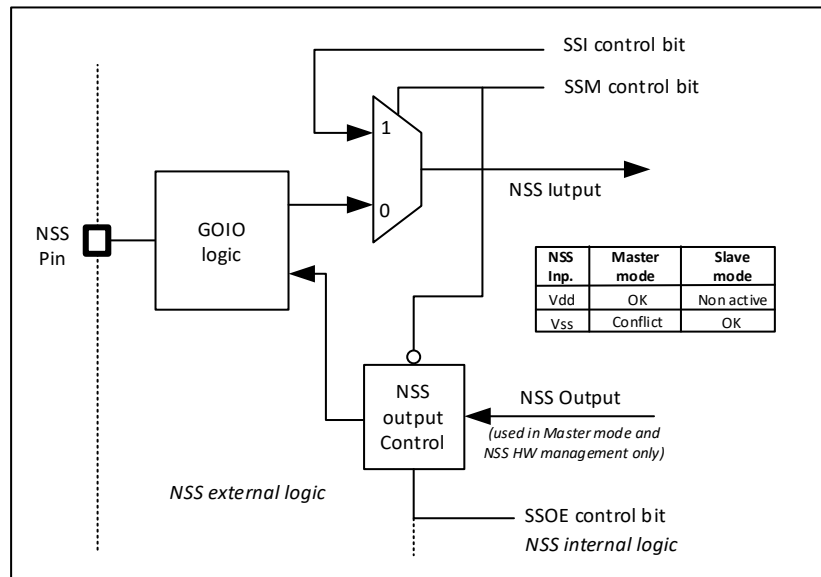


图 31-7 硬件/软件从机 NSS 管理

### 31.3.6. 通讯格式

在 SPI 通讯期间，接收和发送操作同时进行。SCK（serial clock）将数据线上的信息移位和采样操作同步。通讯格式取决于时钟相位、时钟极性和数据帧格式。为了能够进行通讯，主机和从机必须遵循相同的通讯格式。

#### 31.3.6.1. 时钟相位 和极性控制

通过 CPOL 和 CPHA 位（SPI\_CR1 寄存器），软件可以配置 4 种可能的时序。CPOL（clock polarity）控制当没有数据传输时的时钟空闲状态。该位对主机和从机都有影响。如果 CPOL 被复位，SCK 引脚在空闲状态处于低电平。如果 CPOL 被置位，SCK 引脚在空闲状态处于高电平。

如果 CPHA 被置位，SCK 的第二个边沿捕获传输的第一个数据位（如果 CPOL 被复位，是下降沿，否则是上升沿）。在每次出现该时钟边沿时锁存数据。如果 CPHA 被复位，SCK 的第一个边沿捕获第一个传输的数据位（如果 CPOL 被置位，是下降沿，否则是上升沿）。在每次出现该时钟边沿时锁存数据。

CPOL 和 CPHA 的组合用于选择数据捕获时钟边沿。

在 CPOL/CPHA 切换之前，SPI 必须被禁止（SPE=0）。

SCK 的空闲状态必须与 SPI\_CR1 寄存器选择的极性相对应（如果 CPOL=1，则上拉 SCK；如果 CPOL=0，则下拉 SCK）。



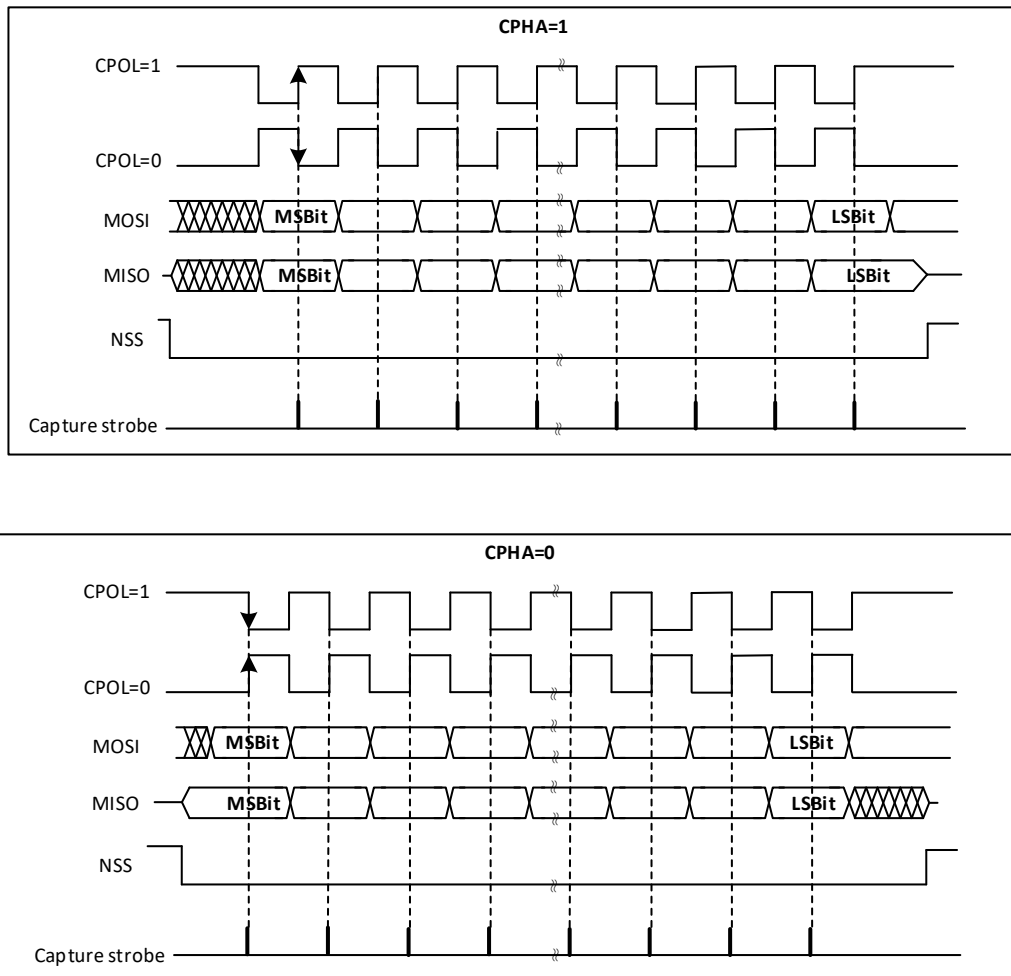


图 31-8 数据时钟时序图

数据位的顺序取决于 LSBFIRST 位的设定。

### 31.3.6.2. 数据帧格式

通过 LSBFIRST 位 (SPI\_CR1 寄存器)，SPI 移位寄存器可以设定为 MSB-FIRST 或者 LSB-FIRST。通过使用 DFF 位 (SPI\_CR1 寄存器)，选择数据帧的位数。可选择为 8 位或者 16 位长度，该设置对于发送和接收都适用。

### 31.3.7. SPI 配置

对于主机和从机，SPI 的配置流程几乎一样。对于具体的模式建立，遵循专门的章节介绍。当进行标准的通讯，进行以下步骤：

- 1) 写相关的 GPIO 寄存器：配置 MOSI、MISO 和 SCK 引脚
- 2) 写 SPI\_CR1 寄存器
  - 通过 BR[2: 0]配置时钟波特率 (从机模式不需要)
  - 配置 CPOL 和 CPHA 定义数据传输和串行时钟之间的关系 (四种关系中的一种)
  - 通过 RXONLY 或者 BIDIMODE 和 BIDIOE (RXONLY 和 BIDIMODE 不能同时有效)，选择单工或者半双工模式
  - 配置 LSBFIRST 以定义帧格式
  - 配置 SSM 和 SSI

- 配置 MSTR 位 (在多主机 NSS 配置中, 如果主机被配置防止 MODF 错误, 要避免 NSS 的冲突状态)
  - 配置 DFF, 选择数据帧位数
- 3) 写 SPI\_CR2 寄存器
    - 配置 SSOE (从机模式不需要)
  - 4) 写 SPI\_CRCPR 寄存器: 需要时配置 CRC 多项式。
  - 5) 写相应的 DMA 寄存器: 使能 TXDMAEN 或者 RXDMAEN, 配置 SYSCFG\_CFGR3.DMAx\_MAP 和 SYSCFG\_CFGR4.DMAx\_MAP 寄存器选择 SPI 对应的 DMA 通道。

### 31.3.8. SPI 使能流程

建议在主机发送时钟之前使能 SPI 从机。如果不这样处理, 数据传输可能不正常。从机的数据寄存器在开始与主机通讯之前, 必须包含要被发送的数据 (或者在通讯时钟的第一个沿, 或者如果时钟信号是连续的情况, 要在正在进行的通讯结束之前)。使能 SPI 从机前, SCK 信号必须稳定为所选极性对应的空闲状态电平。

全双工模式 (或者只发送模式), 当 SPI 被使能并且 TXFIFO 不空, 或者向 TXFIFO 进行下一个写操作, 主机开始通讯。

在任何主机只接收模式 (RXONLY=1, 或者 BIDIMODE=1 且 BIDIOE=0) 下, 主机在使能 SPI 后开始通信, 立即输出时钟。

对于 DMA 处理, 遵从具体的章节内容。

### 31.3.9. 数据传输和接收流程

#### 31.3.9.1. RXFIFO 和 TXFIFO

SPI 所有数据通讯都通过 FIFO, 深度为 4, 宽度为 16 位/8 位。该特性使 SPI 能够以连续数据流进行工作, 并防止由于 CPU 来不及处理数据导致的通讯溢出问题。发送和接收有独立的 FIFO, 叫做 TXFIFO 和 RXFIFO。这些 FIFO 被用在所有的 SPI 模式。

FIFO 的处理取决于多种参数, 包括: 数据交换模式 (全双工、半双工)、数据帧格式。

读 SPI\_DR 寄存器会得到最早存放在 RXFIFO 中还未被读走的数据结果。写 SPI\_DR 寄存器, 会在 FIFO 发送队列的最后位置, 存入被写的数据。读写访问必须与数据宽度对齐。FTLVL[1: 0]和 FRLVL[1: 0]位显示了两个 FIFO 当前的占用级别。

对 SPI\_DR 寄存器的读访问必须通过 RXNE 事件管理。当有数据存储在 RXFIFO 时, 该事件被触发。当 RXNE 被清零, RXFIFO 就被认为是空的。

相似地, 要发送的数据帧的写访问通过 TXE 事件进行管理。当 TXFIFO 的占用级别小于或者等于总容量的一半时, 该事件就会被触发。否则, TXE 被清零, 并且 TXFIFO 被认为是满的。

用这样的方式, RXFIFO 和 TXFIFO 都可以存 4 个数据帧。

TXE 和 RXNE 事件可以通过查询或者中断方式处理。

另一种管理数据交换的方式是使用 DMA

当 RXFIFO 满时，如果下一个数据被接收，则发生上溢事件。上溢事件可以通过查询或者中断的方式处理。

BSY 位被置 1 表示当前正在处理数据帧。当时钟信号连续运行时，在主器件中，BSY 标志在数据帧之间保持置 1 状态，但在从器件中，BSY 标志在数据帧传输之间变为低电平并持续最短的一段时间（一个 SPI 时钟）。

某些应用场景下，当向 TXFIFO 中写入数据，可以通过设置 CLRTXFIFO 位，清空 TXFIFO 数据，从而重新往 TXFIFO 里写新的数据重新进行通信。

### 31.3.9.2. 序列处理

一些数据帧可以通过单序列传递来完成一条信息。使能发送后，当主机的 TXFIFO 里有任何数据，序列开始，只要主器件的 TXFIFO 中存在数据便一直继续。时钟信号被主机连续的提供，直到 TXFIFO 为空，然后时钟信号停止，等待其他的数据。

在只接收模式，即半双工 (BIDIMODE=1, BIDIOE=0) 或者单工模式 (BIDIMODE=0, RXONLY=1)，在 SPI 被使能并且只接收模式被激活的时候，主机就立即开始接收。主机一直会提供时钟并连续地接收数据，直到主机停止 SPI 或者关闭只接收 模式。

当主机能够以连续的模式 (SCK 信号是连续的) 提供所有通讯，主机必须要考虑从机处理数据流的能力。当有必要时，主机必须降低通讯速度，并提供更慢的时钟，或者带有足够延时的单独帧或数据段。要注意的是，对于主机或者从机来说，没有下溢错误信号，来自于从机的数据通常被主机交互和处理（即使从机不能及时准备好数据）。对于从机，更好的方法是使用 DMA，尤其当数据帧小，且波特率高的情况。

每个序列都必须被 NSS 脉冲进行控制，同时多从机系统选择要进行通讯的其中的一个从机。在一个单从机系统，没有必要用 NSS 去控制从机，但此时也最好提供 NSS 脉冲，使从系统与每个数据序列的开头同步。NSS 可以由软件和硬件两种方式管理。

当 BSY 被置位，它显示了正在进行的数据帧交互。当专门的帧交互完成时，RXNE 标志置位。最后一位被采样，并且整个数据帧存储到 RXFIFO 中。

### 31.3.9.3. 关闭 SPI 流程

必须按照特定的关闭 流程关闭 SPI。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。

全双工或者只发送模式下，主机停止提供待发送的数据时，可结束任何事务。在这种情况下，在最后的数据交互后，时钟被停止。在这些模式下，SPI 被关闭之前，用户必须使用标准的关闭 流程。SPI 在主模式传输时，如果在数据帧处理的过程中，或者 TXFIFO 中有待传输的数据时，此时关闭 SPI，则 SPI 的状态是不可预测的。

当主机处在只接收模式，停止连续时钟的唯一方法是禁止外设 (SPE=0)。这必须在最后一个数据帧传输内的特定时间段，即第一位采样与最后一位传输开始之间完成（以便接收全部数量的预期数据帧并防止在最后一个有效数据帧后读取任何其他的“空”数据）。如果不能做到这一点，则主机会一直产生时钟。该模式下，要进行专门的 SPI 关闭流程。

当 SPI 被关闭后，已接收到但未读走的数据存放在 RXFIFO 中，这些数据必须在下一次 SPI 使能后进行处理，才能开始新的序列。为防止有未读的数据，要确保当 SPI 被关闭时，RXFIFO 是空的（使用正确的关闭流程，或者通过用软件复位以初始化所有的 SPI 寄存器）。

标准的关闭流程是基于 BSY 状态，并查看 FTLVL[1: 0]，以确保传输彻底完成。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：

1. 当 NSS 信号被软件管理，主机要向从机提供正确的 NSS 脉冲。或者
2. 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 或 FIFO 的传输数据流完成。

正确的关闭流程是（只接收模式除外）：

- 等待 FTLVL[1: 0]=00（没有数据要发送）
- 等待 BSY=0（最后的数据被处理完成）
- 关闭 SPI（SPE=0）
- 读数据，直到 FRLVL[1: 0]=00（读所有接收到的数据）

对于特定只接收模式，正确的关闭流程是：

- 在最后一个数据帧传输过程中，通过关闭 SPI（SPE=0），打断接收流程
- 等待 BSY=0（最后的数据帧已被处理）
- 读数据，直到 FRLVL[1: 0]=00（读所有接收到的数据）

#### 31.3.9.4. 使用 DMA 通讯

为了以最大的速度工作，并加快数据的读/写过程，以避免溢出，SPI 提供了 DMA 功能，该功能采用了简单的请求/应答协议。

当 TXE 或者 RXNE 被置位，会产生 DMA 请求。Tx 和 Rx 缓冲区有独立的请求。

1. 发送时，每次 TXE 置为1，则产生 DMA 请求。然后 DMA 会向 SPI\_DR 寄存器写入数据。
2. 接收时，每次 RXNE 置为1，则产生 DMA 请求。然后 DMA 读 SPI\_DR 寄存器的数据。

当 SPI 被仅用作发送数据，可以只使能 SPI Tx DMA 通道。在这个情况下，因为被接收到的数据没有被读走，OVR 标志位被置位。当 SPI 被仅用作接收数据，可以使能 SPI Rx DMA 通道。

在发送时，当 DMA 已经写入了所有要发送的数据（DMA\_ISR 寄存器的 TCIF 标志位被置位），可以通过检测 BSY 标志来确保 SPI 通讯已完成。这是用来避免当关闭 SPI 或者进入停止低功耗模式时，最后的传输被破坏。软件必须先等待 FTLVL[1: 0]=00，然后再等待 BSY=0。

当开始使用 DMA 通讯时，为避免 DMA 通道发生错误事件，必须遵从以下步骤：

1. 使能 DMA Rx 缓冲区（SPI\_CR2 的 RXDMAEN 位）（如果 Rx DMA 被使用）
2. 配置 SYSCFG\_CFGR3.DMAx\_MAP 和 SYSCFG\_CFGR4.DMAx\_MAP 寄存器，选择 SPI 使用的 DMA 通道；并配置 DMA 的寄存器（参考 DMA 模块的描述）
3. 使能 DMA Tx 缓冲区（在 SPI\_CR2 寄存器的 TXDMAEN 位）（如果 Tx DMA 被使用）
4. 通过 SPE 位使能 SPI

强制关闭通讯，必须遵从以下步骤：

- 如果使能了 DMA，通过操作 DMA 寄存器来关闭 DMA 模块（参考 DMA 模块的描述）
- 通过 SPI 关闭流程关闭 SPI
- 通过清除 TXDMAEN 和 RXDMAEN（SPI\_CR2 寄存器），关闭 DMA Tx 和 Rx 缓冲区（如果 DMA Tx 和 Rx 被使用）

### 31.3.9.5. 通讯时序

本节介绍一些典型的时序，这些时序对于查询、中断或者 DMA 都是有效的。为了简化，假定  $LSBFIRST=0$ ,  $CPOL=0$ ,  $CPHA=1$ 。不提供完整的 DMA 操作配置。

- 激活 NSS 并使能 SPI 后，从机开始控制 MISO；当 NSS 被释放或者 SPI 关闭时从机失去对 MISO 的控制。必须为从机提供充足的时间，以便在传输开始前准备好主机专用的数据。  
在主机端，仅在 SPI 使能后，SPI 外设才会控制 MOSI 和 SCK 信号（也包括 NSS 信号）。如果 SPI 被关闭，SPI 外设就从 GPIO 断开，在这些线上的电平值取决于 GPIO 的设置。
- 在主机端，如果通讯是连续的，则 BSY 在帧之间保持有效。在从机端，BSY 信号在数据帧之间通常会保持至少一个时钟周期的低电平状态。
- 只有当 TXFIFO 是满的，TXE 信号才被清零。
- 在 TXDMAEN 位被置位后，DMA 仲裁过程即开始。在 TXEIE 被置位后，产生 TXE 中断。当 TXE 信号有效时，开始向 TxFIFO 传输数据，直到 TxFIFO 变满，或者 DMA 传输完成。
- 如果所有要被发送的数据能装进 TxFIFO，在 SPI 总线传输前 DMA Tx TCIF 标志就会被拉高。这个标志在 SPI 交互完成之前，一直都为高。

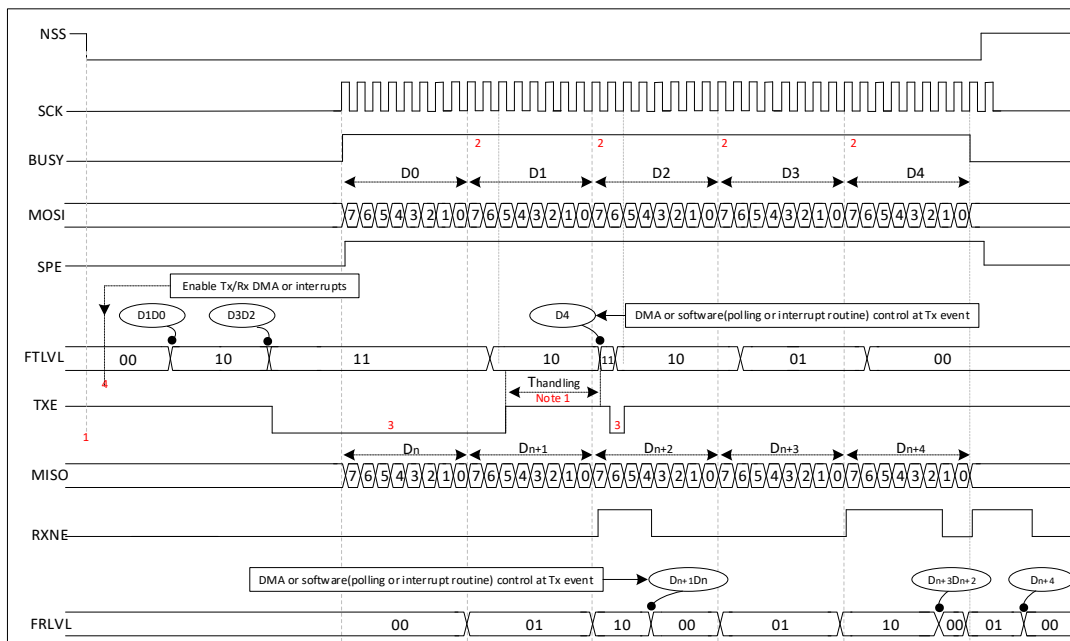


图 31-9 主机全双工通讯时序（数据宽度=8，FRXTH=0）

### 31.3.10. 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

#### 31.3.10.1. 发送缓冲空标志（TXE）

当 TXFIFO 有足够的空间存储要发送的数据时，TXE 标志位被置位。TXE 标志位与 TXFIFO 占用级别有关。该标志位变高并保持高电平，直到 TXFIFO 占用级别小于等于 1/2 FIFO 深度才会被硬件清零。如果 TXEIE（SPI\_CR2）被置位，则会产生中断请求。当 TXFIFO 占用级别大于 1/2，该位被自动清零。

### 31.3.10.2. 接收缓冲非空标志 (RXNE)

RXNE 标志位被置位:

- 当接收到数据, RXNE 置位。  
当上述条件不再成立, 则 RXNE 被硬件自动清零。

### 31.3.10.3. 忙标志 (BSY)

BSY 标志由硬件设置与清除 (写入此位无效果)。

当它被设置为'1'时, 表明 SPI 正忙于通信, 但有一个例外: 在主模式的双向接收模式下 (MSTR=1、BDM=1 并且 BDOE=0), 在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停止低功耗模式 (或关闭设备时钟) 之前, 可以使用 BSY 标志检测传输是否结束, 这样可以避免破坏最后一次传输, 因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主机系统中避免写冲突。

除了主模式的双向接收模式 (MSTR=1、BDM=1 并且 BDOE=0), 当传输开始时, BSY 标志被置'1'。

以下情况该标志将被清除为'0':

- 正确关闭 SPI
- 主机模式, 当产生 MODF=1
- 主机模式, 当传输完成, 不再有效数据要发送
- 从机模式, 在每个数据传输之间, BSY 标志置为0, 并保持至少一个 SPI 时钟周期

注意: 不要使用 BSY 标志处理每个数据发送和接收。使用 TXE 和 RXNE 更合适。

## 31.3.11. 错误标志

### 31.3.11.1. 主模式故障 (MODF)

主模式失效 (MODF) 仅发生在: 当 NSS 作为输入信号 (SSOE=0), NSS 引脚在硬件模式管理下, 主机的 NSS 脚被拉低; 或者在 NSS 引脚在软件模式管理下, SSI 位被置为'0'时。此时, MODF 位被自动置位。

主模式故障对 SPI 设备有以下影响:

- MODF 位被置为'1', 如果设置了 ERRIE 位, 则产生 SPI 中断;
- SPE 位被清为'0'。这将停止一切输出, 并且关闭 SPI 接口;
- MSTR 位被清为'0', 从而强制设备进入从模式。

下面的步骤用于清除 MODF 位:

1. 当 MODF 位被置为'1'时, 执行一次对 SPI\_SR 寄存器的读或写操作;
2. 然后写 SPI\_CR1 寄存器。

在有多个 MCU 的系统中, 为了避免出现多个从设备的冲突, 必须先拉高该主设备的 NSS 脚, 再对 MODF 位进行清零。在完成清零之后, SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑, 当 MODF 位为'1'时, 硬件不允许设置 SPE 和 MSTR 位。

通常配置下，从机的 MODF 位不能被置为'1'。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从机模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

#### 31.3.11.2. 上溢标志 (OVR)

当数据被主机或者从机接收，并且 RXFIFO 没有足够的空间接收到的数据时，产生上溢的情况。如果软件或者 DMA 没有足够的时间读走以前接收到的数据 (RXFIFO 中存放)，该情况就会发生。

当上溢情况发生，新收到的数据不会覆盖以前存放在 RXFIFO 的数据。接收到的新数据被忽略，并且接下来发送的所有数据将被丢弃。

依次读出 SPI\_DR 寄存器后可将 SPI\_SR 寄存器的 OVR 位清除。

#### 31.3.11.3. 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\_RXCRCR 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。

### 31.3.12. SPI 中断

表 31-1 SPI 中断请求

中断事件	事件标志	使能控制位
TXFIFO 等待被装载	TXE	TXEIE
数据接收到 RXFIFO 中	RXNE	RXNEIE
主模式故障事件	MODF	ERRIE
上溢错误	OVR	ERRIE
CRC 协议错误	CRCERR	ERRIE

### 31.3.13. CRC 计算

CRC 校验用于保证通信的可靠性，数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。SPI 提供独立于帧数据长度的 CRC8 或 CRC16 计算，可固定为 8 位或 16 位。

#### 31.3.13.1. CRC 原理

在 SPI 启用 (SPE = 1) 之前，通过设置 SPIx\_CR1 寄存器中的 CRCEN 位来启用 CRC 计算。CRC 值是使用每个位上的奇数可编程多项式计算的。该计算在由 SPIx\_CR1 寄存器中的 CPHA 和 CPOL 位定义的采样时钟边沿上进行处理。计算出的 CRC 值在数据块结束时自动检查，并且由 CPU 或 DMA 管理其传输。当检测到根据接收数据内部计算的 CRC 与发送器发送的 CRC 不匹配时，CRCERR 标志将置位以指示数据错误。CRC 计算的正确处理步骤取决于 SPI 配置和选择的传输管理方式。

注：多项式值只应为奇数，不支持任何偶数值。

### 31.3.13.2. 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 寄存器中最后一个数据帧。然后必须在 SPIx\_CR1 寄存器中设置 CRCNEXT 位，用于指示在当前处理的数据帧后开始传输 CRC 帧。CRCNEXT 位必须在最后一个数据帧传输结束之前置位。在 CRC 传输期间停止 CRC 计算。

接收到的 CRC 数据以字节或字的形式存储在 RXFIFO 中。这就是为什么仅在 CRC 模式下，必须将接收缓冲区视为单个 16 位缓冲区，用于一次仅接收一个数据帧。CRC 格式的传输通常在数据传输结束时再传输一个数据帧。但是，当设置通过 16 位 CRC 校验的 8 位数据帧时，需要另外两帧才能发送完整的 CRC。

当接收到最后一个 CRC 数据时，将执行自动校验，比较接收到的值和 SPIx\_RXCRC 寄存器。软件必须检查 SPIx\_SR 寄存器中的 CRCERR 标志以确定数据传输是否出错。软件通过向其写入“0”来清除 CRCERR 标志。接收 CRC 后，CRC 值存储在 RXFIFO 中，必须读 SPIx\_DR 寄存器以清除 RXNE 标志。

### 31.3.13.3. DMA 管理的 CRC 传输

当使能 SPI，并开启带 CRC 的 DMA 模式进行通信时，通信结束时会自动发送和接收 CRC（在仅接收模式下读取 CRC 数据除外），CRCNEXT 位不必由软件处理。SPI 传输 DMA 通道的计数器必须设置为要传输的数据帧数，其中不包括 CRC 帧。在接收端，接收到的 CRC 值在传输结束时由 DMA 自动处理，但用户必须注意从 RXFIFO 中清除接收到的 CRC 信息，因为它总是存放到 RXFIFO 中。在全双工模式下，接收 DMA 通道的计数器可以设置为要接收的包括 CRC 在内的数据帧的数量。

在只接收模式下，DMA 接收通道计数器应该只包含传输的数据量，而不包括 CRC 计算。然后基于 DMA 的完整传输，因为它在此模式下作为单个缓冲区工作，所以所有 CRC 值必须由软件从 FIFO 读回。在数据和 CRC 传输结束时，如果在传输过程中出错，SPIx\_SR 寄存器中的 CRCERR 标志会被置位。



## 31.4. I<sup>2</sup>S 功能描述

### 31.4.1. 概述

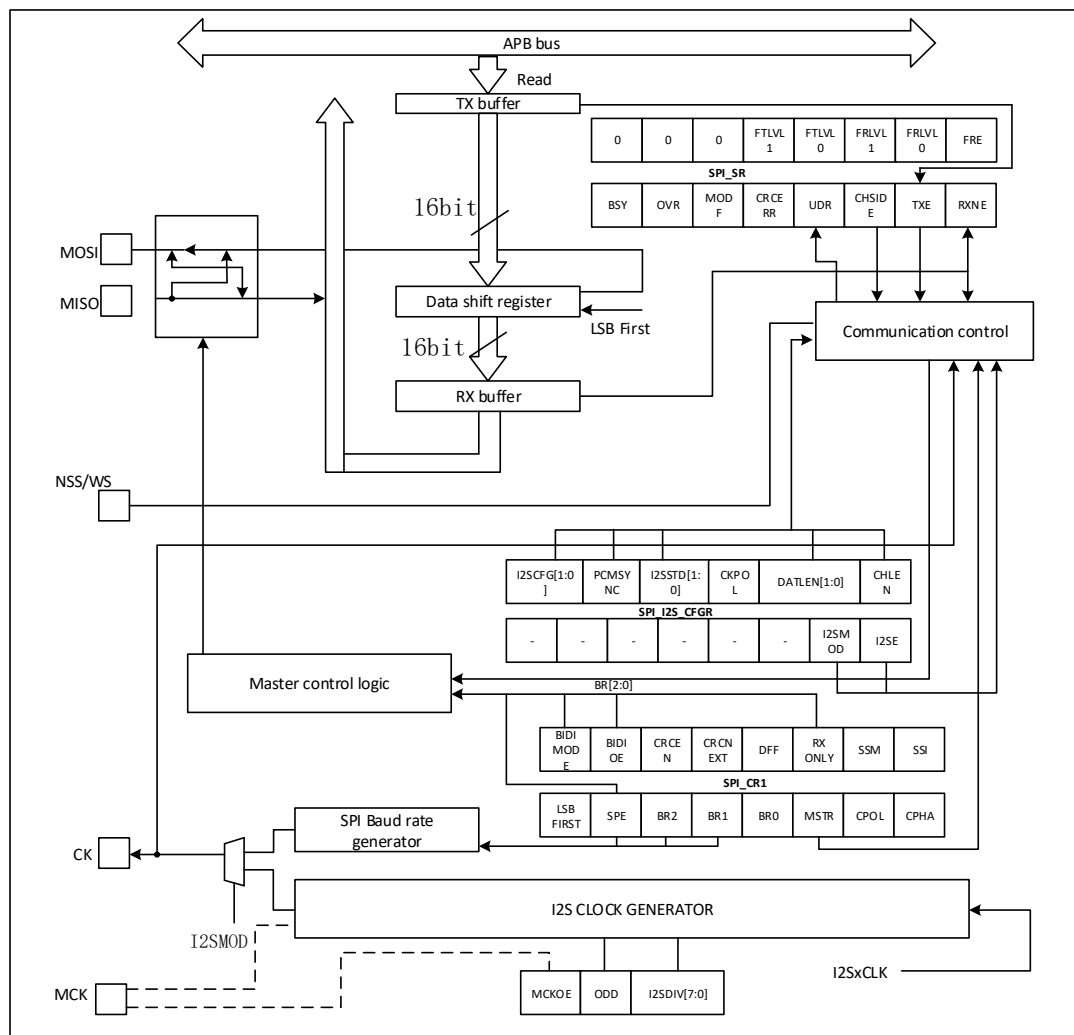


图 31-10 IIS 框图

通过将寄存器 SPI\_I2SCFGR 的 I2SMOD 位置为 '1'，即可使能 I2S 功能。此时，可以把 SPI 模块用作 I2S 音频接口。I2S 接口与 SPI 接口使用大致相同的引脚、标志和中断。

I2S 与 SPI 共用 3 个引脚：

- SD：串行数据（映射至 MOSI 引脚），用来发送和接收 2 路时分复用通道的数据；
- WS：字选（映射至 NSS 引脚），主模式下作为数据控制信号输出，从模式下作为输入；
- CK：串行时钟（映射至 SCK 引脚），主模式下作为时钟信号输出，从模式下作为输入。

在某些外部音频设备需要主时钟时，可以另有一个附加引脚输出时钟：

- MCK：主时钟（映射至 MISO 引脚），在 I2S 配置为主模式，寄存器 SPI\_I2SPR 的 MCKOE 位为 '1' 时，作为输出额外的时钟信号引脚使用。输出时钟信号的频率预先设置为  $256 \times F_s$ ，其中  $F_s$  是音频信号的采样频率。

设置成主模式时，I2S 使用自身的时钟发生器来产生通信用的时钟信号。这个时钟发生器也是主时钟输出的时钟源。I2S 模式下有 2 个额外的寄存器，一个是与时钟发生器配置相关的寄存器

SPI\_I2SPR, 另一个是 I2S 通用配置寄存器 SPI\_I2SCFGR (可设置音频标准、从/主模式、数据格式、数据包帧、时钟极性等参数)。

在 I2S 模式下不使用寄存器 SPI\_CR1 和所有的 CRC 寄存器。同样, I2S 模式下也不使用寄存器 SPI\_CR2 的 SSOE 位, 和寄存器 SPI\_SR 的 MODF 位和 CRCERR 位。

I2S 使用与 SPI 相同的寄存器 SPI\_DR 用作 16 位宽模式数据传输。

### 31.4.2. 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 位寄存器用作发送或接收。因此, 需由软件将与每个通道对应的值写入数据寄存器, 或者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。左声道总是先于右声道发送数据 (CHSIDE 位在 PCM 协议下无意义)。有四种可用的数据和包帧组合。可以通过以下四种数据格式发送数据:

- 16位数据打包进16位帧
- 16位数据打包进32位帧
- 24位数据打包进32位帧
- 32位数据打包进32位帧

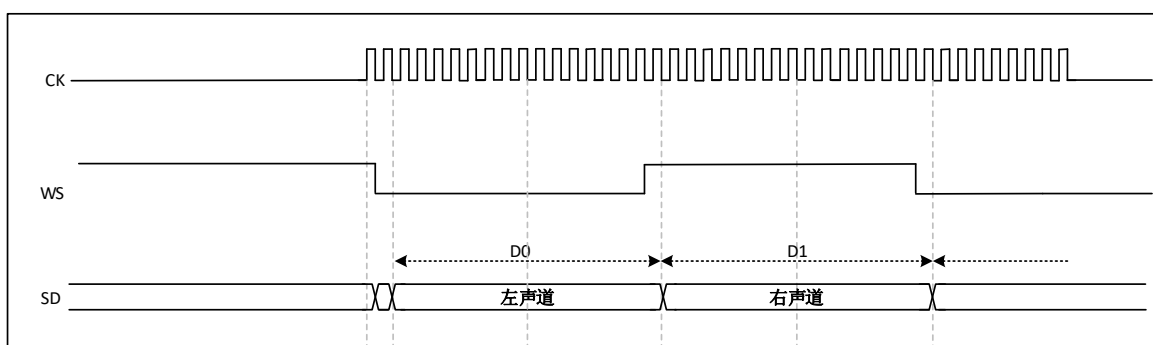
在使用 16 位数据扩展到 32 位帧时, 前 16 位 (MSB) 是有意义的数字, 后 16 位 (LSB) 被强制为 0, 该操作不需要软件干预, 也不需要 DMA 请求 (仅需要一次读/写操作)。24 位和 32 位数据帧需要 CPU 对寄存器 SPI\_DR 进行 2 次读或写操作, 在使用 DMA 时, 需要 2 次 DMA 传输。对于 24 位数据, 扩展到 32 位后, 最低 8 位由硬件置 0。对于所有的数据格式和通讯标准, 总是先发送最高位 (MSB)。

I2S 接口支持四种音频标准, 可以通过设置寄存器 SPI\_I2SCFGR 的 I2SSTD[1: 0]位和 PCMSYNC 位来选择。

#### 31.4.2.1. I2S 飞利浦协议

在此标准下, 引脚 WS 用来指示正在发送的数据属于哪个声道。在发送第一位数据 (MSB) 有效之前, 存在 1 个时钟周期。

16/32 位全精度时序图如下图所示, 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。



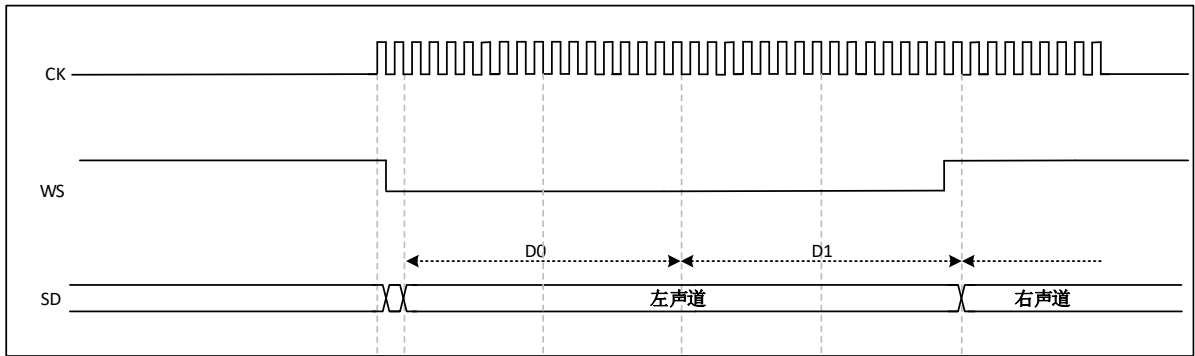


图 31-11 I2S 飞利浦协议波形 (16/32位全精度, CKPOL = 0)

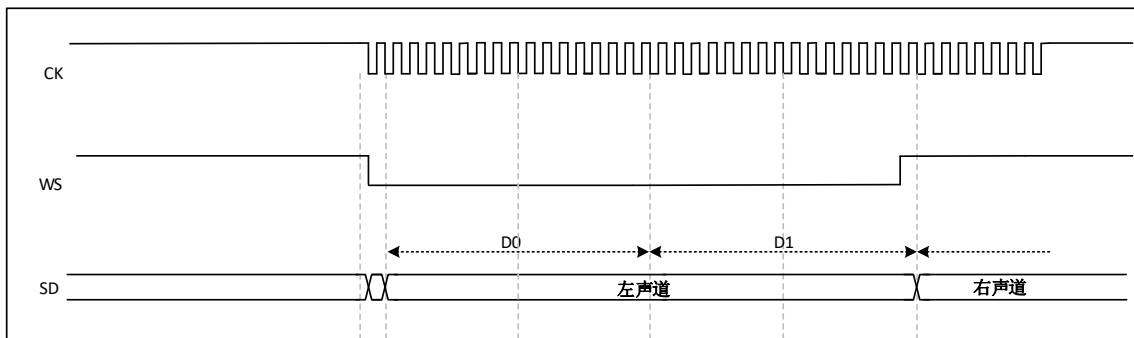
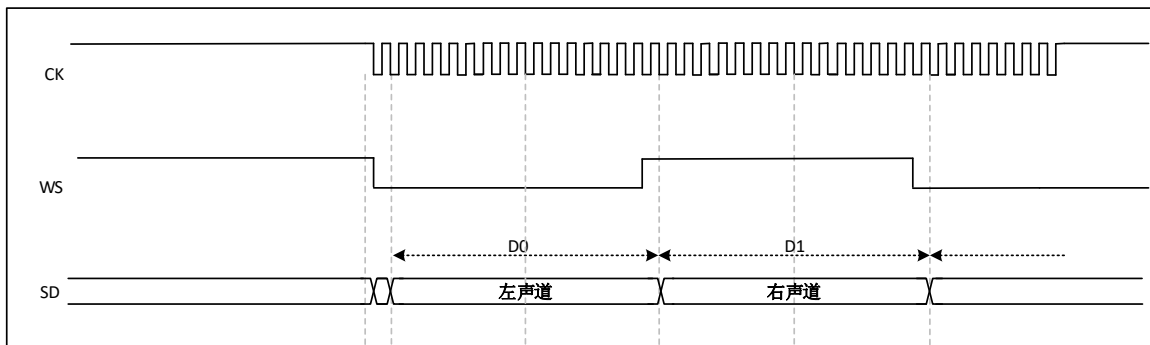


图 31-12 I2S 飞利浦协议波形 (16/32位全精度, CKPOL = 0)

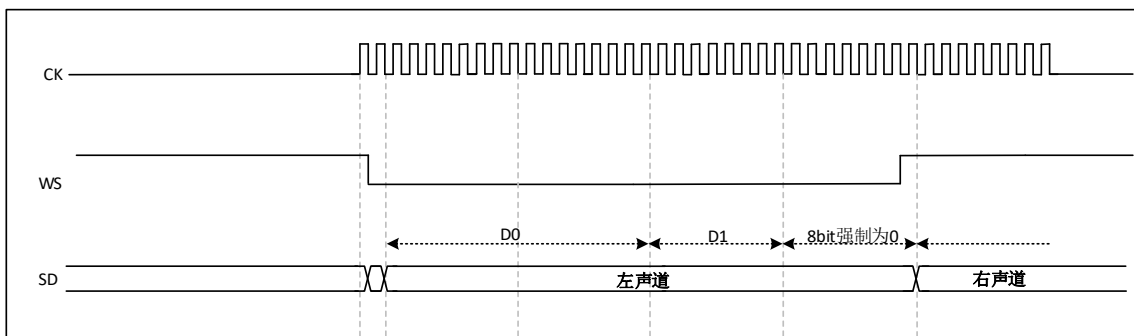


图 31-13 I2S 飞利浦协议标准波形 (24位帧, CKPOL = 0)

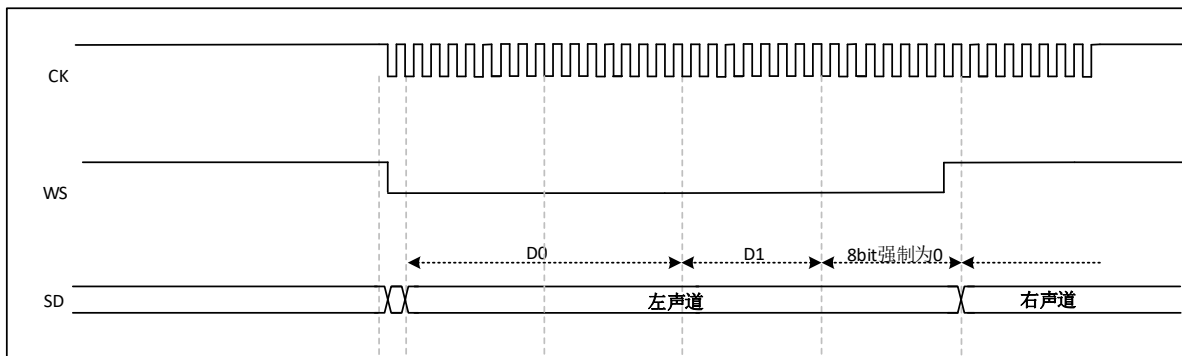


图 31-14 I2S 飞利浦协议标准波形 (24位帧, CKPOL = 1)

此模式需要对寄存器 SPI\_DR 进行 2 次读或写操作。

- 在发送模式下: 如果需要发送 0x8EAA33 (24位), 第一次写入 0x8EAA, 第二次写入 0x33xx, 低8位无意义
- 在接收模式下: 如果接收 0x8EAA33 (24位), 第一次接收 0x8EAA, 第二次读出 0x3300, 只高8位为有意义的数据, 低8位始终为 0

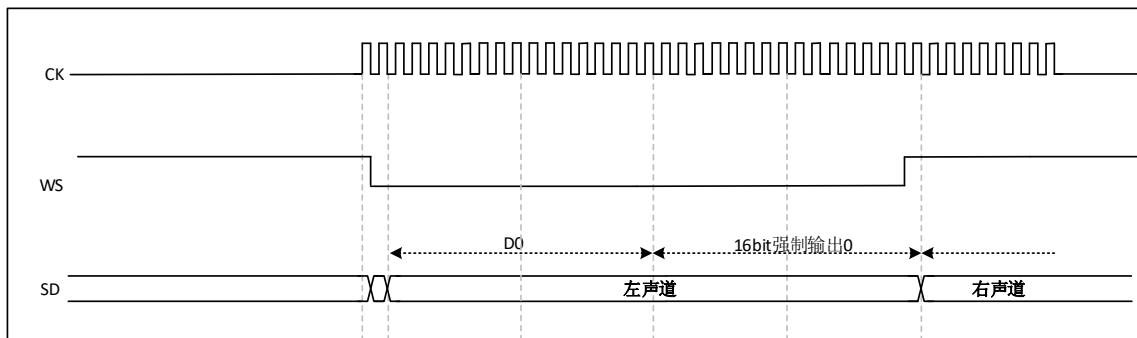


图 31-15 I2S 飞利浦协议标准波形 (16位扩展至32位包帧, CPOL = 0)

在 I2S 配置阶段, 如果选择将 16 位数据扩展到 32 位声道帧, 只需要访问一次寄存器 SPI\_DR。用来扩展到 32 位的低 16 位被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3 (扩展到 32 位是 0x76A30000), 只需要操作一次 SPI\_DR, 写入数据 0x76A3。

在发送时需要将 MSB 写入寄存器 SPI\_DR: 标志位 TXE 置 1, 表示可以写入新的数据, 如果允许了相应的中断, 则可以产生中断。发送是由硬件完成的, 即使还未发送出后 16 位的 0x0000, 也会设置 TXE 并产生相应的中断。接收时, 每次收到高 16 位半字 (MSB) 后, 标志位 RXNE 置 1, 如果允许了相应的中断, 则可以产生中断。

这样, 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)。

#### 31.4.2.2. MSB 对齐标准

在此标准下, WS 信号和第一个数据位, 即最高位 (MSB) 同时产生。

16/32 位全精度时序图如下图所示, 发送方在时钟信号的下降沿改变数据; 接收方是在上升沿读取数据。

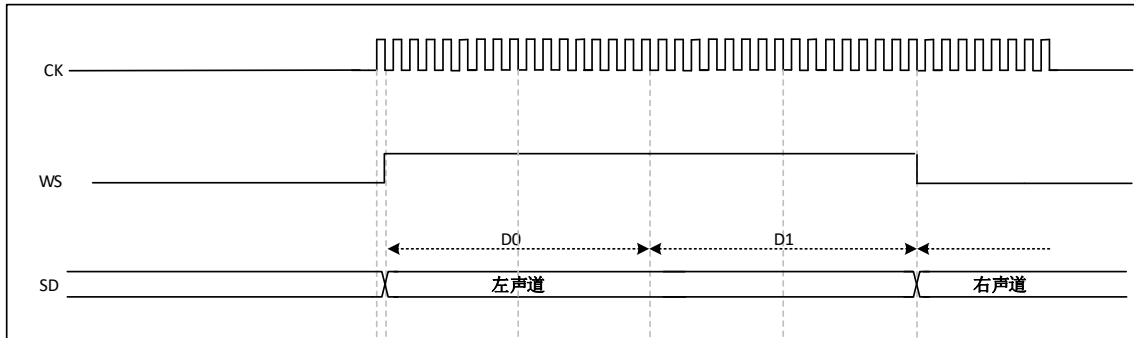
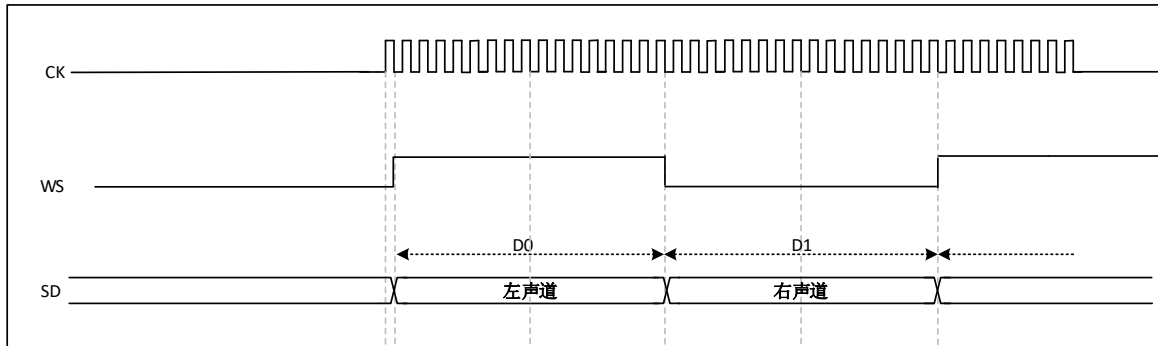


图 31-16 MSB 对齐16位或32位全精度, CPKOL = 0

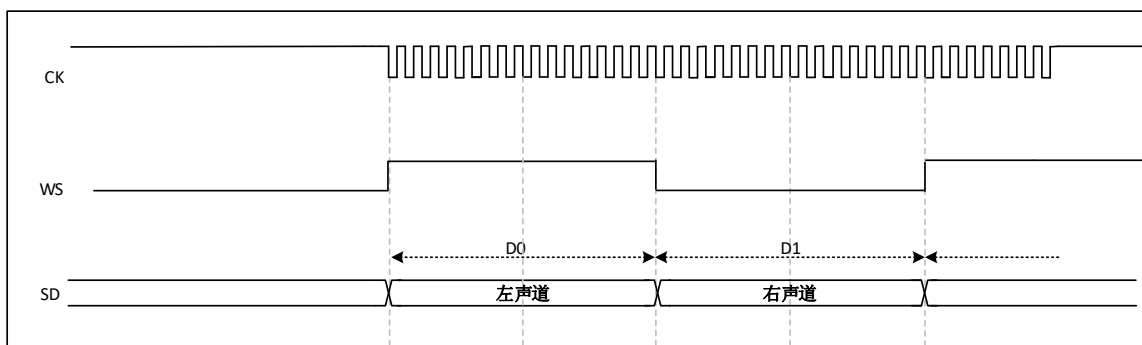
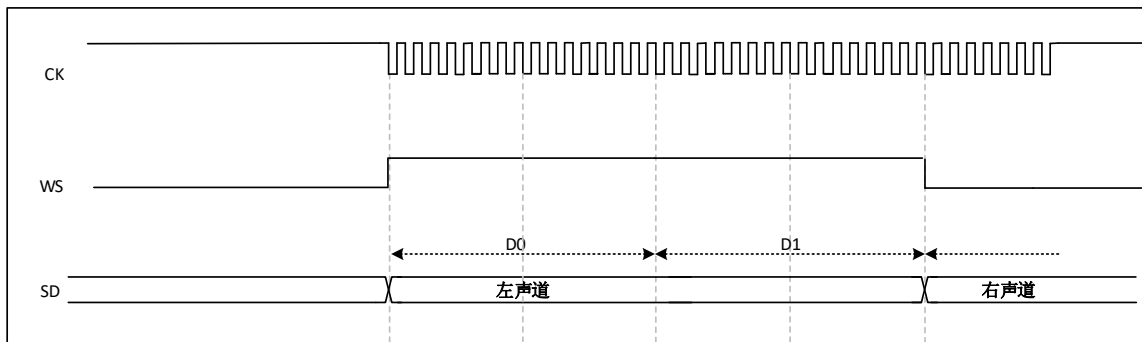


图 31-17 MSB 对齐16位或32位全精度, CPKOL = 1

24 位帧时序图如下:

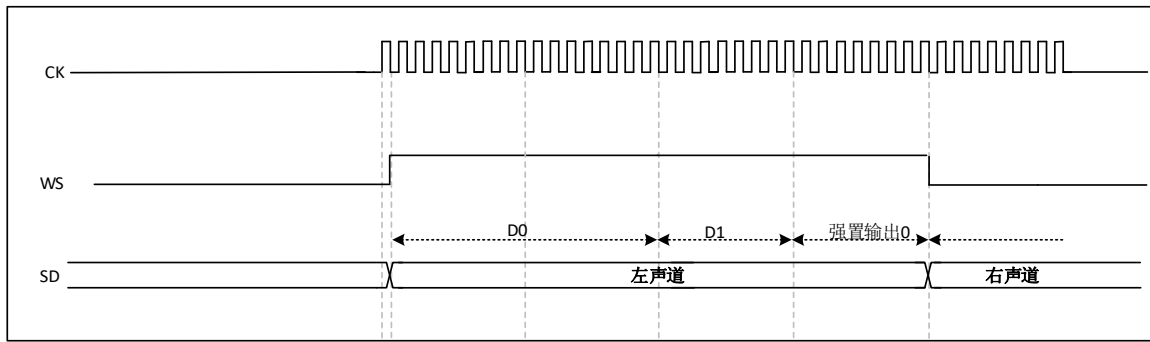


图 31-18 MSB 对齐24位帧, CPKOL = 0

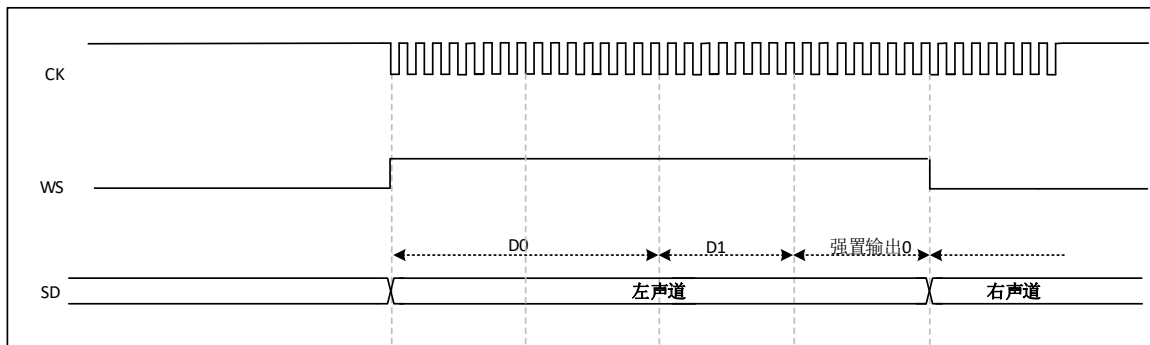


图 31-19 MSB 对齐24位帧, CPKOL = 1

下图为 16 位扩展到 32 位声道帧的时序:

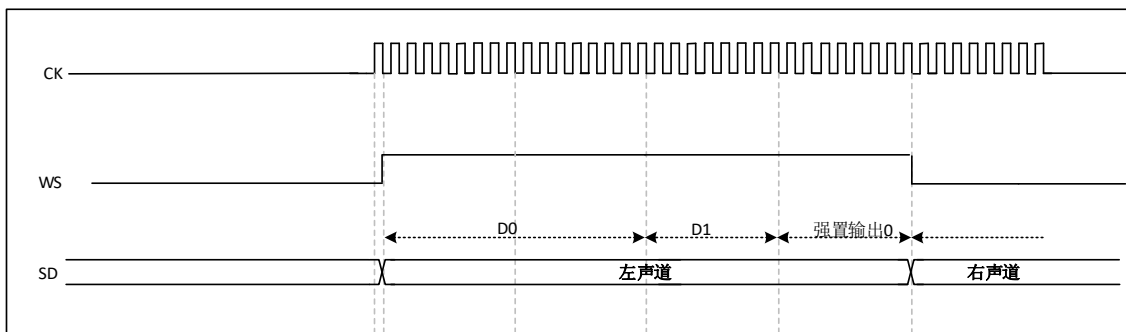


图 31-20 MSB 对齐标准, 16位扩展到32位, CPKOL = 0

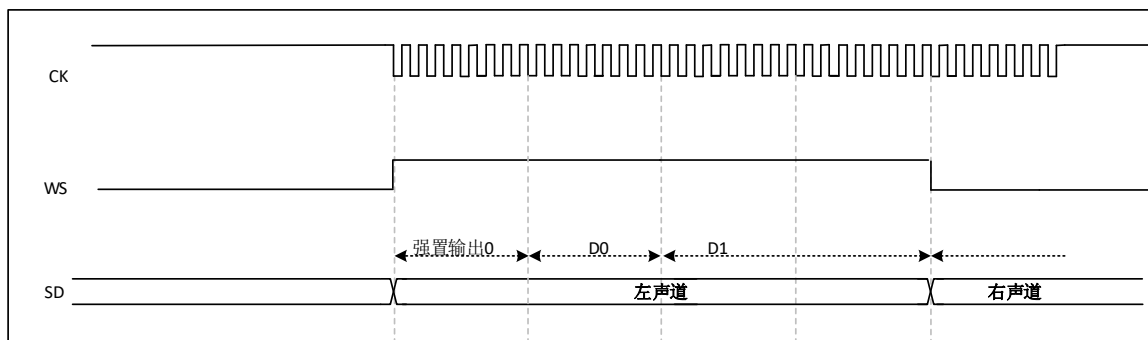


图 31-21 MSB 对齐标准, 16位扩展到32位, CPKOL = 1

一旦有效数据开始从 SD 引脚送出, 即发生下一次 TXE 事件。在接收时, 一旦接收到有效数据 (而不是 0x0000 部分), 即发生 RXNE 事件。

### 31.4.2.3. LSB 对齐标准

此标准与 MSB 对齐标准类似 (在 16 位或 32 位全精度帧格式下无区别)。

16/32 位全精度时序图如下图所示，发送方在时钟信号的下降沿改变数据；接收方是在上升沿读取数据。

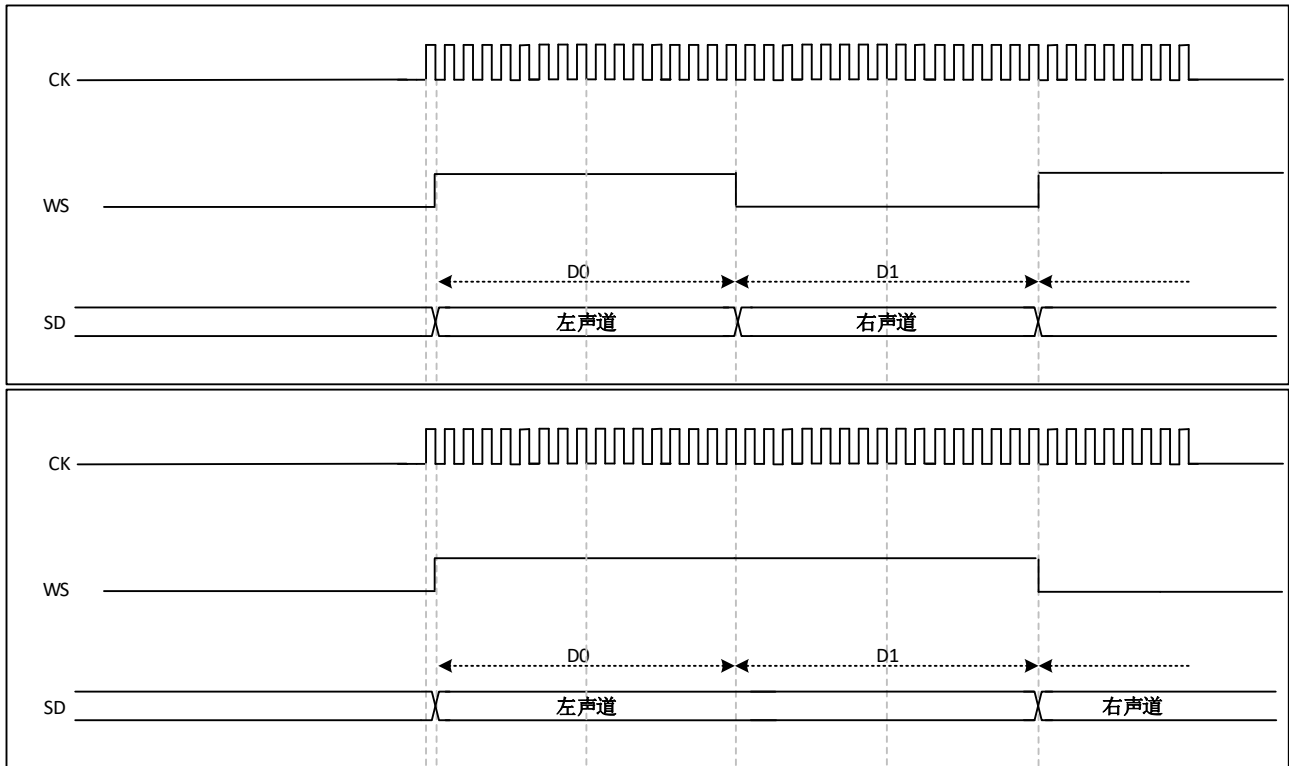


图 31-22 LSB 对齐16位或32位全精度，CKPOL = 0

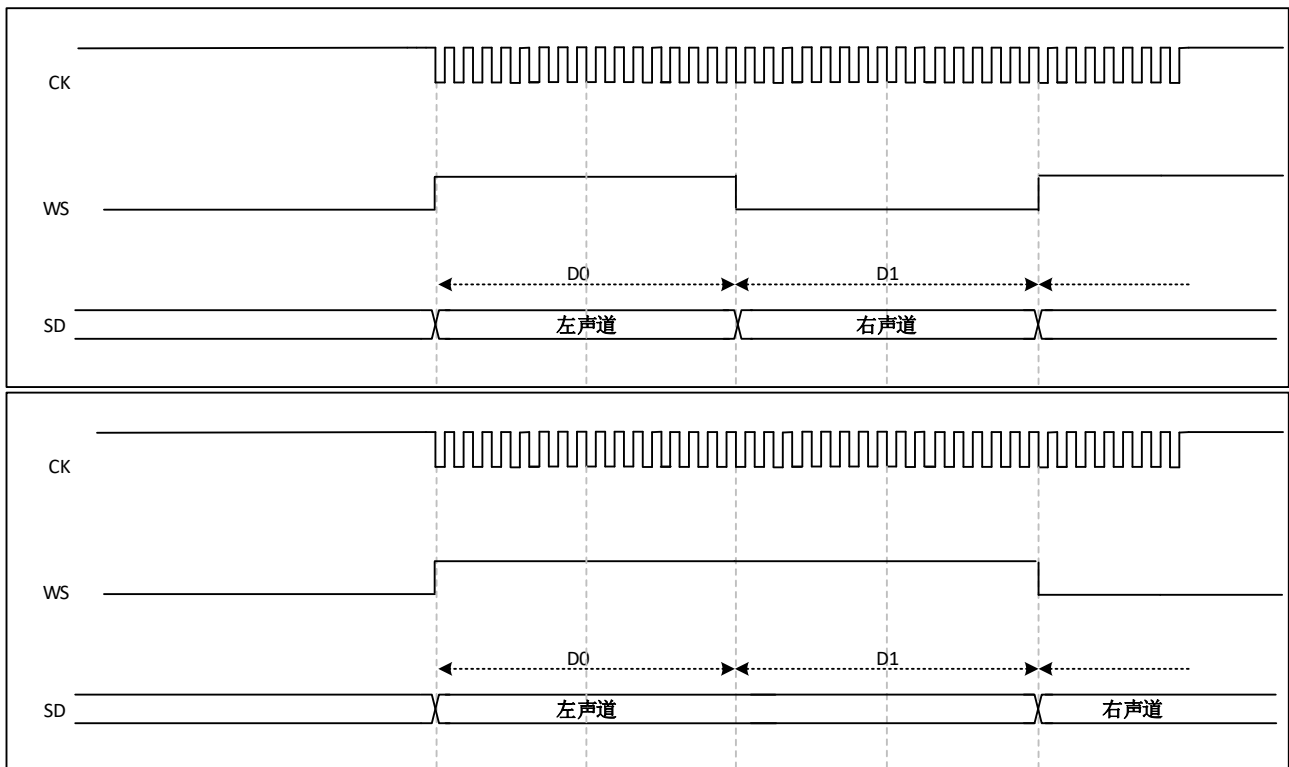


图 31-23 LSB 对齐16位或32位全精度，CKPOL = 0

24 位帧时序图如下：

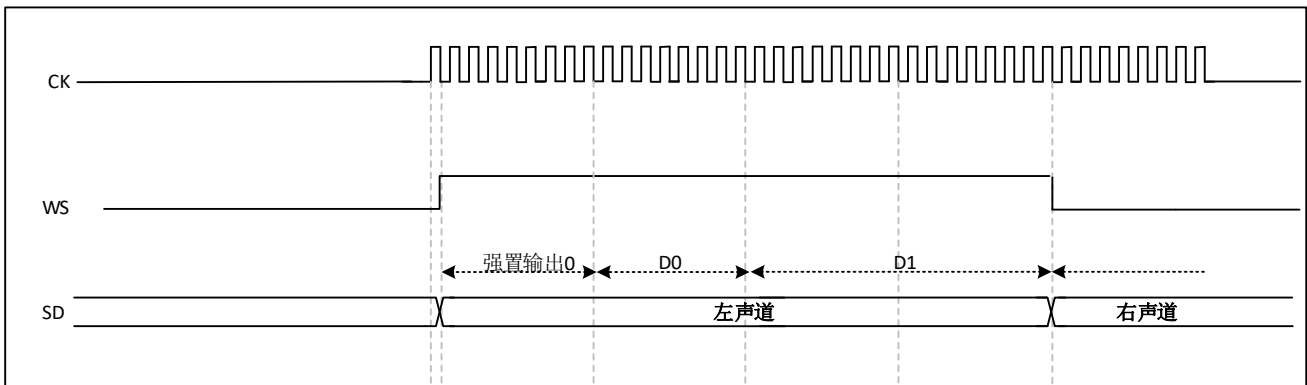


图 31-24 LSB 对齐24位数据, CKPOL = 0

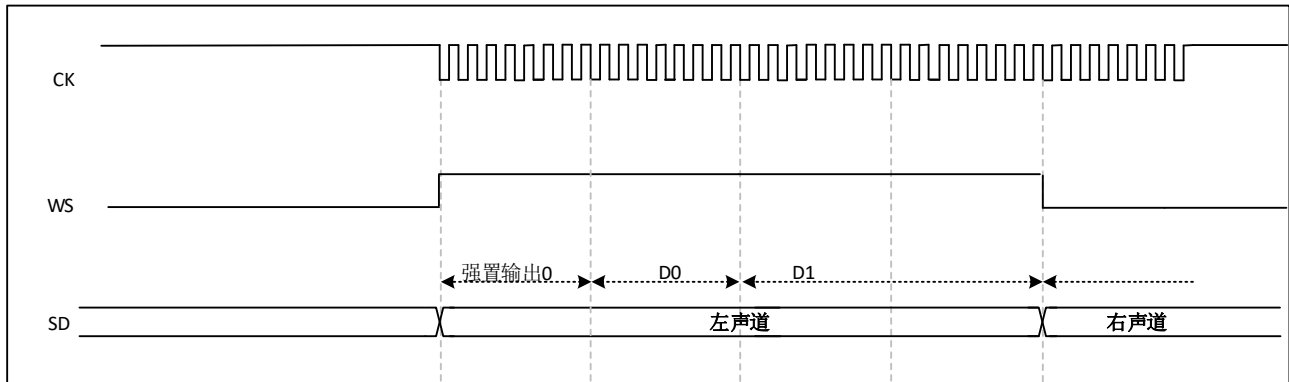


图 31-25 LSB 对齐24位数据, CKPOL = 1

■ 在发送模式下

如果要发送数据 0x3478AE，需要通过软件或者 DMA 对寄存器 SPI\_DR 进行 2 次写操作。第一次写入数据寄存器 0xXX34，第二次写入数据寄存器 0x78AE。

■ 在接收模式下

如果要接收数据 0x3478AE，需要在 2 个连续的 RXNE 事件发生时，分别对寄存器 SPI\_DR 进行 1 次读操作。第一次读出 0x0034，只有低 8 位有意义；第二次读出 0x78AE。

下图为16位扩展到32位声道帧的时序：

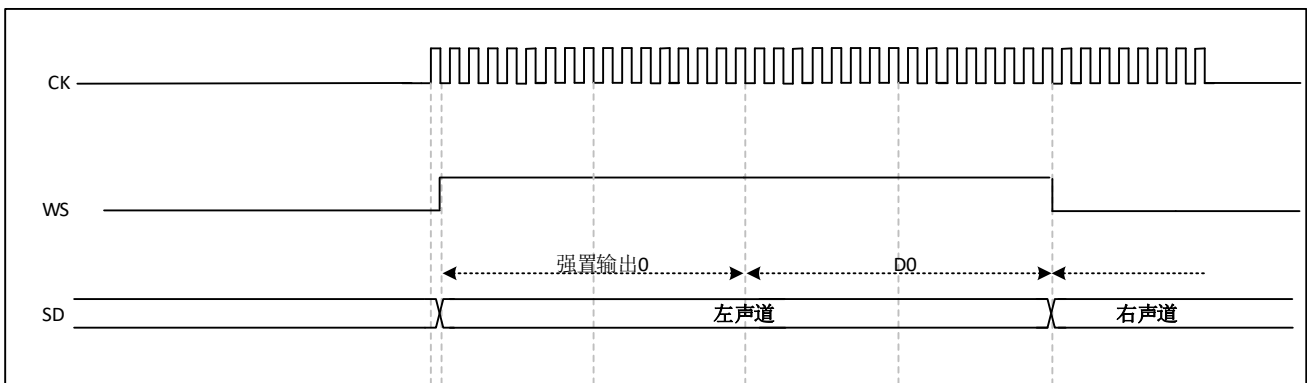


图 31-26 LSB 对齐16位数据扩展到32位包帧, CKPOL = 0



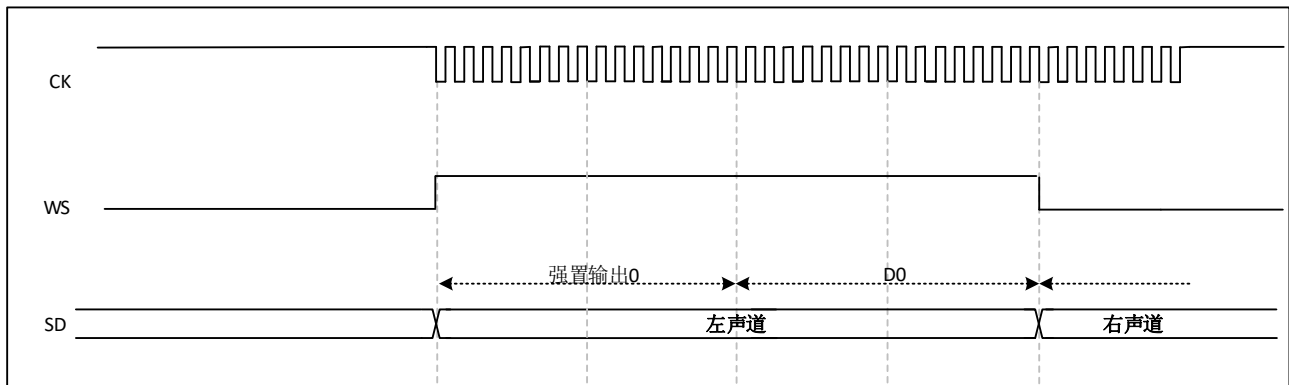


图 31-27 LSB 对齐16位数据扩展到32位包帧, CKPOL =1

在 I2S 配置阶段, 如果选择将 16 位数据扩展到 32 声道帧, 只需要访问一次寄存器 SPI\_DR。此时, 扩展到 32 位后的高半字 (16 位 MSB) 被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3 (扩展到 32 位是 0x0000 76A3), 只需要操作一次 SPI\_DR 寄存器, 写入 0x76A3。

在发送时, 发生 TXE 事件时, 用户需要写入待发送的数据 (即 0x76A3)。用来扩展到 32 位的 0x0000, 部分由硬件首先发送出去, 一旦有效数据开始从 SD 引脚送出, 即发生下一次 TXE 事件。

在接收时, 一旦接收到有效数据 (而不是 0x0000 部分), 即发生 RXNE 事件。

这样, 在 2 次读和写之间有更多的时间, 可以防止下溢或者上溢的情况发生。

#### 31.4.2.4. PCM 标准

在 PCM 标准下, 不存在声道选择的信息。PCM 标准有 2 种可用的帧结构, 短帧或者长帧, 可以通过设置寄存器 SPI\_I2SCFGR 的 PCMSYNC 位来选择。

在 PCM 模式下, 输出信号 (WS 和 SD) 在 CK 信号的上升沿采样。输入信号 (WS 和 SD) 在 CK 信号的下降沿捕获。

请注意, CK 和 WS 在主模式下配置为输出。

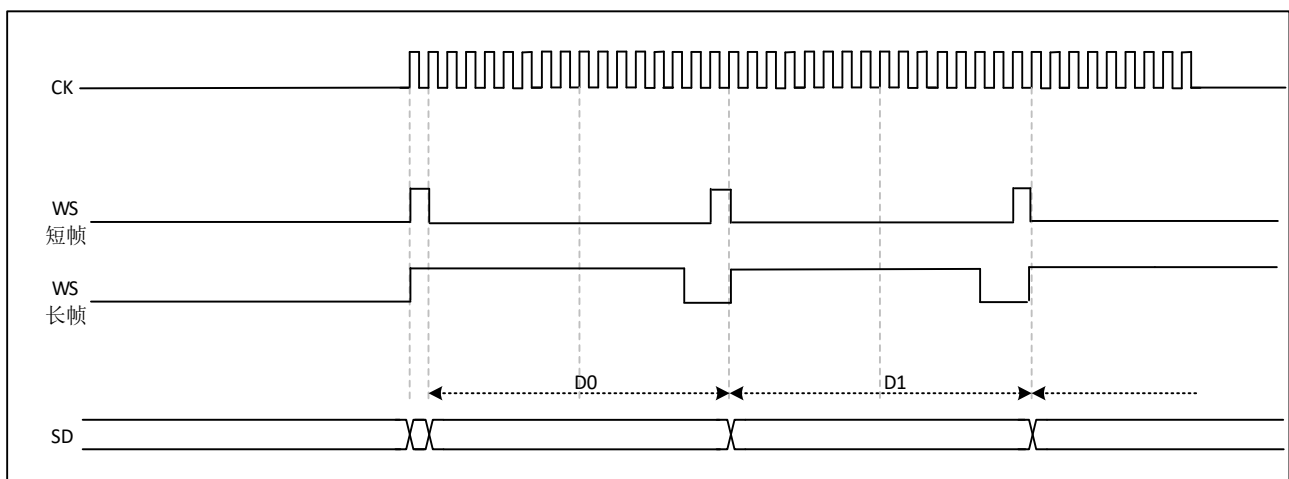


图 31-28 PCM 标准波形 (16位, CKPOL=0)

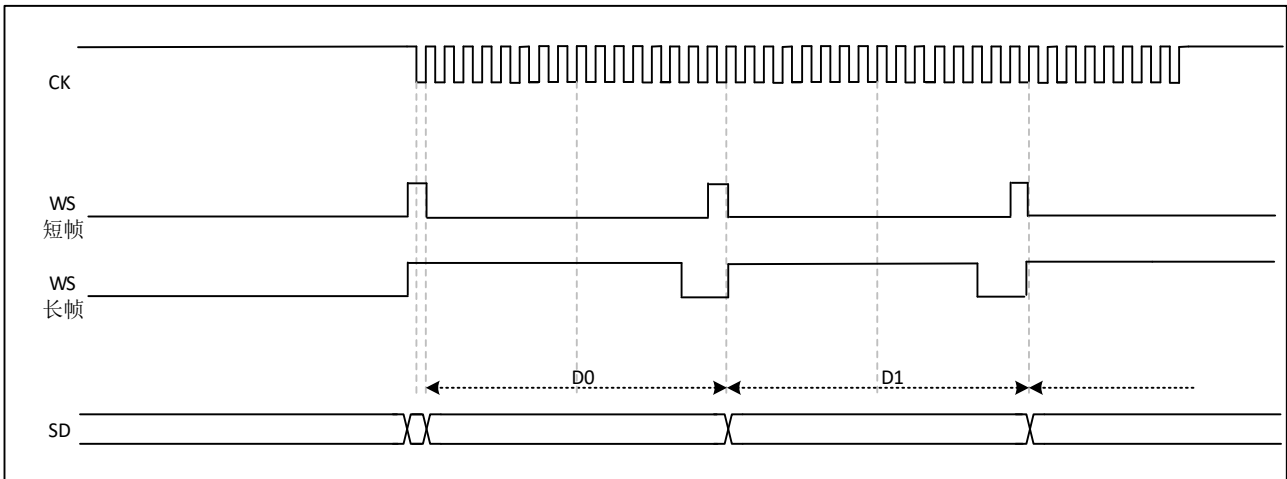


图 31-29 PCM 标准波形 (16位, CKPOL=1)

对于长帧, 主模式下, 用来同步的 WS 信号有效的的时间固定为 13 个周期。

对于短帧, 用来同步的 WS 信号长度只有 1 个周期。

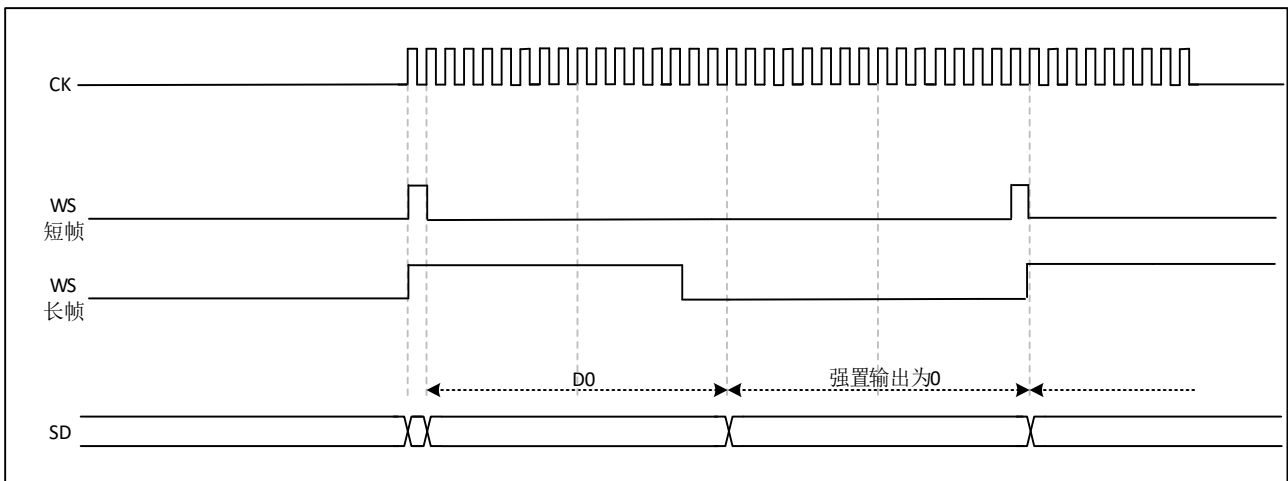


图 31-30 PCM 标准波形 (16位扩展到32位包帧, CKPOL=0)

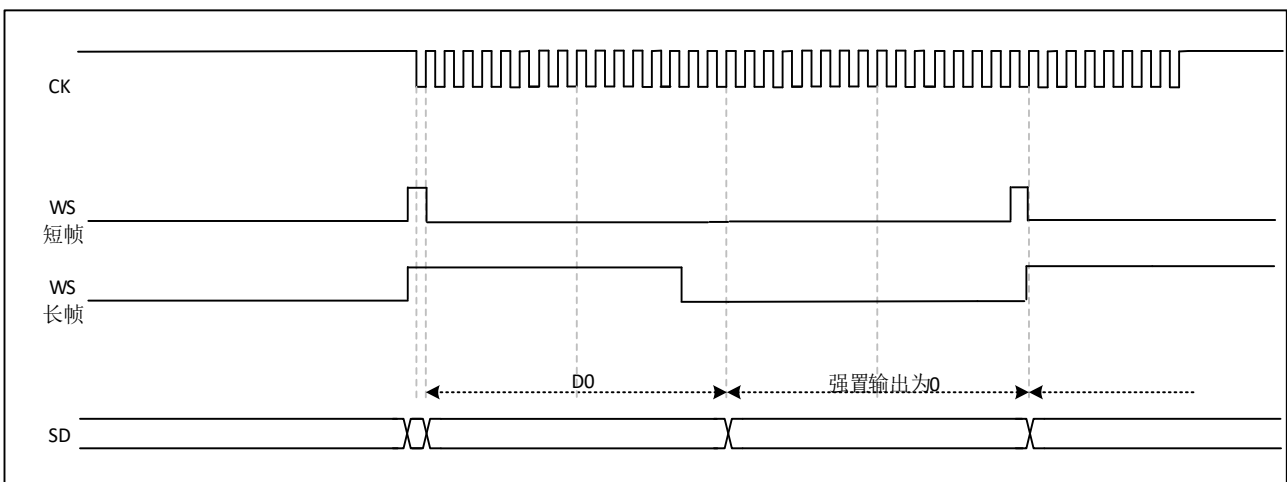


图 31-31 PCM 标准波形 (16位扩展到32位包帧, CKPOL=1)

注意：无论哪种模式（主或从）、哪种同步方式（短帧或长帧），连续的 2 帧数据之间和 2 个同步信号之间的时间差，（即使是从模式）需要通过设置 SPI\_I2SCFGR 寄存器的 DATLEN 位和 CHLEN 位来确定。

### 31.4.3. 时钟发生器

I2S 的比特率即确定了在 I2S 数据线上的数据流和 I2S 的时钟信号频率。即

I2S 比特率 = 每个声道的比特数 × 声道数目 × 音频采样频率

对于一个具有左右声道和 16 位音频信号，I2S 比特率计算如下：

$$\text{I2S 比特率} = 16 \times 2 \times f_s;$$

如果包长为 32 位，则有：I2S 比特率 = 32 × 2 ×  $f_s$

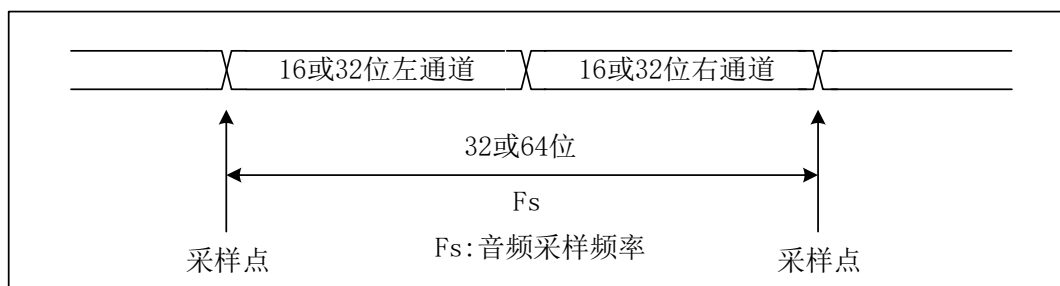


图 31-32 音频采样定义

在主模式下，为了获得需要的音频频率，需要正确地对线性分频器进行设置。

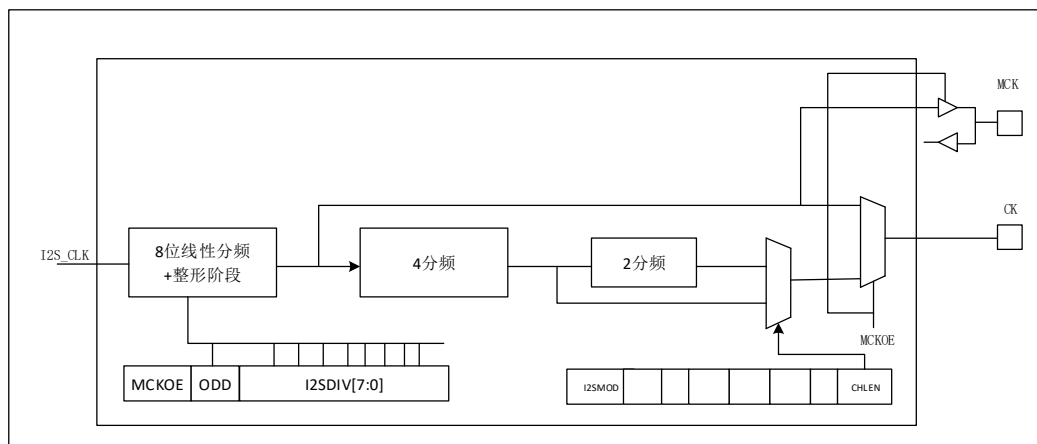


图 31-33 I2S 时钟发生器结构

音频的采样频率可以是 96 kHz、48 kHz、44.1 kHz、32 kHz、22.05 kHz、16 kHz、11.025 kHz 或者 8 kHz（或任何此范围内的数值）。为了获得需要的频率，需按照以下公式设置线性分频器而获得，当需要生成主时钟时（寄存器 SPI\_I2SPR 的 MCKOE 位为 '1'）：

声道的帧长为 16 位时， $F_s = I2SxCLK / [ (16 \times 2) * ( (2 * I2SDIV) + ODD) * 8 ]$

声道的帧长为 32 位时， $F_s = I2SxCLK / [ (32 \times 2) * ( (2 * I2SDIV) + ODD) * 4 ]$

当关闭主时钟时（MCKOE 位为 '0'）：

声道的帧长为 16 位时， $F_s = I2SxCLK / [ (16 \times 2) * ( (2 * I2SDIV) + ODD) ]$

声道的帧长为 32 位时， $F_s = I2SxCLK / [ (32 \times 2) * ( (2 * I2SDIV) + ODD) ]$

使用标准的 8 MHz HSE 时钟得到精确的音频频率，见下表：

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	期望 fs (Hz)	实际 fs (Hz)		误差	
	16位	32位	16位	32位			16位	32位	16位	32位
72	11	6	1	0	无	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	无	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	无	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	无	32000	32142.86	32142.86	0.45%	0.45%
72	51	25	0	1	无	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	无	16000	15957.45	16071.43	0.27%	0.45%
72	102	51	0	0	无	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	无	8000	8007.117	7978.723	0.09%	0.27%
72	2	2	0	0	有	96000	70312.5	70312.5	26.76%	26.76%
72	3	3	0	0	有	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	有	44100	46875	46875	6.29%	6.29%
72	4	4	1	1	有	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	有	22050	21634.62	21634.62	1.88%	1.88%
72	9	9	0	0	有	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	有	11025	10817.31	10817.31	1.88%	1.88%
72	17	17	1	1	有	8000	8035.714	8035.714	0.45%	0.45%

#### 31.4.4. I<sup>2</sup>S 主模式

设置 I<sup>2</sup>S 工作在主模式，串行时钟由引脚 CK 输出，字选信号由引脚 WS 产生。可以通过设置寄存器 SPI\_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟（MCK）。

步骤如下：

1. 设置寄存器 SPI\_I2SPR 的 I2SDIV[7:0]定义与音频采样频率相符的串行时钟波特率。同时也要定义寄存器 SPI\_I2SPR 的 ODD 位。
2. 设置 CKPOL 位定义通信用时钟在空闲时的电平状态。如果需要向外部的 DAC/ADC 音频器件提供主时钟 MCK，将寄存器 SPI\_I2SPR 的 MCKOE 位置为 1，并按照不同的 MCK 输出状态，计算 I2SDIV 和 ODD 的值。
3. 设置寄存器 SPI\_I2SCFGR 的 I2SMOD 位为 1 以激活 I2S 功能，设置 I2SSTD[1:0]和 PCMSYNC 位选择所用的 I<sup>2</sup>S 标准，设置 CHLEN 选择每个声道的数据位数。还要设置寄存器 SPI\_I2SCFGR 的 I2SCFG[1:0]选择 I<sup>2</sup>S 主模式和方向。
4. 如果需要，可以通过设置寄存器 SPI\_CR2 来打开所需的 interrupt 功能和 DMA 功能。
5. 必须将寄存器 SPI\_I2SCFGR 的 I2SE 位置为 '1'。
6. 引脚 WS 和 CK 需要配置为输出模式。如果寄存器 SPI\_I2SPR 的 MCKOE 位为 '1'，引脚 MCK 也要配置成输出模式。

##### 1) 发送流程

当写入 1 个半字（16 位）的数据至发送缓存，发送流程开始。

假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时，标志位 TXE 置 '1'，这时，要把对应右声道的数据写入发送缓存。标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。标志位 CHSIDE 的值在 TXE 为 '1' 时更新，因此它在 TXE 为 '1' 时有意义。在先左声道后右声道的数据都传输完成后，才能被认为是一个完整的数据帧。不可以只传输部分数据帧，如仅有左声道的数据。

当发出第一位数据的同时，半字数据被并行地传送到 16 位移位寄存器，然后后面的位依次按高位在前的顺序从引脚 MOSI/SD 发出。每次数据从发送缓存移至移位寄存器时，标志位 TXE 置为 '1'，如果寄存器 SPI\_CR2 的 TXEIE 位为 '1'，则产生中断。

写入数据的操作取决于所选择的 I2S 标准。为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI\_DR 写入下一个要传输的数据。

建议在要关闭 I2S 功能时，等待标志位 TXE=1 及 BSY=0，再将 I2SE 位清 '0'。

## 2) 接收流程

接收流程的配置步骤除了第 3 点外，与发送流程的一致（参见前述的“发送流程”），需要通过配置 I2SCFG[1:0]来选择主接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收。即每次填满接收缓存后，标志位 RXNE 置 '1'，如果寄存器 SPI\_CR2 的 RXNEIE 位为 '1'，则产生中断。根据配置的数据和声道长度，收到左声道或右声道的数据会需要 1 次或者 2 次把数据传送到接收缓存的过程。对寄存器 SPI\_DR 进行读操作即可清除 RXNE 标志位。每次接收以后即更新 CHSIDE。它的值取决于 I2S 单元产生的 WS 信号。读取数据的操作取决于所选择的 I2S 标准。

如果前一个接收到的数据还没有被读取，又接收到新数据，即发生上溢，标志位 OVR 被置为 '1'，如果寄存器 SPI\_CR2 的 ERRIE 位为 '1'，则产生中断，表示发生了错误。

若要关闭 I2S 功能，需要执行特别的操作，以保证 I2S 模块可以正常地完成传输周期而不会开始新的数据传输。操作过程与数据配置和通道长度、以及音频协议的模式相关：

- 16位数据扩展到32位通道长度 (DATLEN=00并且 CHLEN=1)，使用 LSB (低位) 对齐模式 (I2SSTD=10)
  - a) 等待倒数第二个 RXNE=1 (n-1)；
  - b) 等待17个 I2S 时钟周期 (使用软件延迟)；
  - c) 关闭 I2S (I2SE=0)。
- 16位数据扩展到32位通道长度 (DATLEN=00并且 CHLEN=1)，使用 MSB (高位) 对齐、I2S 或 PCM 模式 (分别为 I2SSTD=00, I2SSTD=01或 I2SSTD=11)
  - a) 等待最后一个 RXNE=1；
  - b) 等待1个 I2S 时钟周期 (使用软件延迟)；
  - c) 关闭 I2S (I2SE=0)。
- 所有 DATLEN 和 CHLEN 的其它组合，I2SSTD 选择的任意音频模式，使用下述方式关闭 I2S：
  - a) 等待倒数第二个 RXNE=1 (n-1)；
  - b) 等待一个 I2S 时钟周期 (使用软件延迟)；
  - c) 关闭 I2S (I2SE=0)。

注：在传输期间 BSY 标志始终为低。

### 31.4.5. I2S 从模式

在从模式下，I2S 可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要 I2S 接口提供时钟。时钟信号和 WS 信号都由外部主 I2S 设备提供，连接到相应的引脚上。因此用户无需配置时钟。

1. 设置寄存器 SPI\_I2SCFGR 的 I2SMOD 位激活 I2S 功能；设置 I2SSTD[1: 0]来选择所用的 I2S 标准；设置 DATLEN[1: 0]选择数据的比特数；设置 CHLEN 选择每个声道的数据位数。设置寄存器 SPI\_I2SCFGR 的 I2SCFG[1: 0]选择 I2S 从模式的数据方向。
2. 根据需要，设置寄存器 SPI\_CR2 打开所需的 interrupt 功能和 DMA 功能。
3. 必须将寄存器 SPI\_I2SCFGR 的 I2SE 位置 '1' 。

#### 1) 发送流程

当外部主设备发送时钟信号，并且当 NSS\_WS 信号请求传输数据时，发送流程开始。必须先使能从设备，并且写入 I2S 数据寄存器之后，外部主设备才能开始通信。

对于 I2S 的 MSB 对齐和 LSB 对齐模式，第一个写入数据寄存器的数据项对应左声道的数据。当开始通信时，数据从发送缓冲器传送到移位寄存器，然后标志位 TXE 置为 '1'；这时，要把对应右声道的数据项写入 I2S 数据寄存器。

标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。与主模式的发送流程相比，在从模式中，CHSIDE 取决于来自外部主 I2S 的 WS 信号。这意味着从 I2S 在接收到主机生成的时钟信号之前，就要准备好第一个要发送的数据。WS 信号为 '1' 表示先发送左声道。

**注：**设置 I2SE 位为 '1' 的时间，应当比 CK 引脚上的主 I2S 时钟信号早至少 2 个 PCLK 时钟周期。当发出第一位数据的时候，半字数据并行地通过 I2S 内部总线传输至 16 位移位寄存器，然后其它位依次按高位在前的顺序从引脚 MOSI/SD 发出。每次数据从发送缓冲器传送到移位寄存器时，标志位 TXE 置 '1'，如果寄存器 SPI\_CR2 的 TXEIE 位为 '1'，则产生中断。注意，在对发送缓冲器写入数据前，要确认标志位 TXE 为 '1'。写入数据的操作取决于所选中的 I2S 标准。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPI\_DR 写入下一个要传输的数据。如果在代表下一个数据传输的第一个时钟边沿到达之前，新的数据仍然没有写入寄存器 SPI\_DR，下溢标志位会置 '1'，并可能产生中断；它指示软件发送数据错误。如果寄存器 SPI\_CR2 的 ERRIE 位为 '1'，在寄存器 SPI\_SR 的标志位 UDR 为高时，就会产生中断。建议在这时关闭 I2S，然后重新从左声道开始发送数据。

建议在清除 I2SE 位关闭 I2S 之前，先等待 TXE=1 并且 BSY=0。

#### 2) 接收流程

配置步骤除了第 1 点外，与发送流程一致。需要通过配置 I2SCFG[1: 0]来选择从接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收，即每次填满接收缓存，标志位 RXNE 置 '1'，如果寄存器 SPI\_CR2 的 RXNEIE 位为 '1'，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要 1 次或者 2 次传输数据至接收缓冲器的过程。每次接收到数据（将要从 SPI\_DR 读出）以后即更新 CHSIDE，它对应 I2S 单元产生的 WS 信号。读取 SPI\_DR 寄存器，将清除 RXNE 位。读取数据的操作取决于所选中的 I2S 标准，详见 5.2 节。

有 3 个状态标志位供用户监控 I2S 总线的状态。

#### 31.4.5.1. 忙标志位 (BSY)

BSY 标志由硬件设置与清除（写入此位无效果），该标志位指示 I2S 通信层的状态。该位为 '1' 时表明 I2S 通讯正在进行中，但有一个例外：主接收模式 (I2SCFG=11) 下，在接收期间 BSY 标志始终为低。在软件要关闭 I2S 模块之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。当传输开始时，BSY 标志被置为 '1'，除非 I2S 模块处于主接收模式。

下述情况时，该标志位被清除：

当传输结束时（除了主发送模式，这种模式下通信是连续的）；

当关闭 I2S 模块时。

当通信是连续的时候：

在主发送模式时，整个传输期间，BSY 标志始终为高；

在从模式时，BSY 标志在每次传输之间变为低电平并持续一个 I2S 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

#### 31.4.5.2. 发送缓存空标志位 (TXE)

该标志位为 '1' 表示发送缓冲器为空，可以对发送缓冲器写入新的待发送数据。在发送缓冲器中已有数据时，标志位清 '0'。在 I2S 被关闭时 (I2SE 位为 '0')，该标志位也为 '0'。

#### 31.4.5.3. 接收缓存非空标志位 (RXNE)

该标志位置 '1' 表示在接收缓存里有接收到的有效数据。在读取寄存器 SPI\_DR 时，该位清 '0'。

#### 31.4.5.4. 声道标志位 (CHSIDE)

在发送模式下，该标志位在 TXE 为高时刷新，指示从 SD 引脚上发送的数据所在的声道。如果在从发送模式下发生了下溢错误，该标志位的值无效，在重新开始通讯前需要把 I2S 关闭再打开。在接收模式下，该标志位在寄存器 SPI\_DR 接收到数据时刷新，指示接收到的数据所在的声道。如果发生错误（如上溢 OVR），该标志位无意义，需要将 I2S 关闭再打开（同时，如果必要修改 I2S 的配置）。

在 PCM 标准下，无论短帧格式还是长帧格式，这个标志位都没有意义。

如果寄存器 SPI\_SR 的标志位 OVR 或 UDR 为 '1'，且寄存器 SPI\_CR2 的 ERRIE 位为 '1'，则会产生中断，而后可以通过读寄存器 SPI\_SR 来清除中断标志。

### 31.4.6. 错误标志位

I2S 单元有 2 个错误标志位。

#### 31.4.6.1. 下溢标志位 (UDR)

在从发送模式下，如果数据传输的第一个时钟边沿到达时，新的数据仍然没有写入 SPI\_DR 寄存器，该标志位会被置 '1'。在寄存器 SPI\_I2SCFGR 的 I2SMOD 位置 '1' 后，该标志位才有效。如果寄存器 SPI\_CR2 的 ERRIE 位为 '1'，就会产生中断。通过对寄存器 SPI\_SR 进行读操作来清除该标志位。

### 31.4.6.2. 上溢标志位 (OVR)

如果还没有读出前一个接收到的数据时，又接收到新的数据，即产生上溢，该标志位置‘1’，如果寄存器 SPI\_CR2 的 ERRIE 位为‘1’，则产生中断指示发生了错误。这时，接收缓存的内容，不会刷新为从发送设备送来的新数据。对寄存器 SPI\_DR 的读操作返回最后一个正确接收到的数据。其他所有在上溢发生后由发送设备发出的 16 位数据都会丢失。通过先读寄存器 SPI\_SR 再读寄存器 SPI\_DR，来清除该标志位。

### 31.4.7. I2S 中断

I2S 中断请求

中断事件	事件标志位	使能标志位
发送缓冲器空标志位	TXE	TXEIE
接收缓冲器非空标志位	RXNE	RXNEIE
上溢标志位	OVR	ERRIE
下溢标志位	UDR	ERRIE

### 31.4.8. DMA 功能

除了 CRC 功能外，同 SPI。因为在 I2S 模式下没有数据传输保护系统。

## 31.5. SPI 和 I<sup>2</sup>S 寄存器

SPI 对应的寄存器可以进行 16-位和 32-位访问，DR 寄存器支持 32-位、16-位和 8-位访问。

### 31.5.1. SPI 控制寄存器 1 (SPI\_CR1) (I<sup>2</sup>S 模式下不使用)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BI-DIMODE	BIDIOE	CRCE	CRCNEXT	DF	RXONLY	SSM	SSI	LSBFIRST	SPEN	BR[2:0]			MSTR	CPOL	CPHA
RW															

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	双向数据模式使能 0: 选择“双线双向”模式; 1: 选择“单线双向”模式。 注: I <sup>2</sup> S 模式下不使用。
14	BIDIOE	RW	0	双向模式下的输出使能和 BIDIMODE 位一起决定在“单线双向”模式下数据的输出方向



Bit	Name	R/W	Reset Value	Function
				0: 输出禁止 (只收模式) ; 1: 输出使能 (只发模式) 。 这个“单线”数据线在主设备端为 MOSI 引脚, 在从设备端为 MISO 引脚。 注: I <sup>2</sup> S 模式下不使用。
13	CRCEN	RW	0	硬件 CRC 校验使能 0: 禁止 CRC 计算; 1: 启动 CRC 计算。 注: 只有在禁止 SPI 时 (SPE=0) , 才能写该 位, 否则出错。 注: I <sup>2</sup> S 模式下不使用。
12	CRCNEXT	RW	0	下一个发送 CRC 0: 下一个发送的值来自发送缓冲区。 1: 下一个发送的值来自发送 CRC 寄存器。 注: 在 SPI_DR 寄存器写入最后一个数据后应马 上设置该位。 注: I <sup>2</sup> S 模式下不使用。
11	DFF	RW	0	数据帧格式 0: 使用8位数据帧格式进行发送/接收; 1: 使用16位数据帧格式进行发送/接收。 注: 只有当 SPI 禁止 (SPE=0) 时, 才能写该 位, 否则出错。 注: I <sup>2</sup> S 模式下不使用。
10	RXONLY	RW	0	只接收 该位和 BIDIMODE 位一起决定在“双线单向”模 式下的传输方向。在多个从设备的配置中, 在未 被访问的从设备上该位置1, 使得只有被访问的从 设备才有输出, 因而不会造成数据线上有数据冲 突。 0: 全双工 (发送和接收) 1: 禁止输出 (只接收模式) 注: I <sup>2</sup> S 模式下不使用。
9	SSM	RW	0	软件从设备管理 当 SSM 置位, NSS 引脚上的电平由 SSI 位的值 决定。 0: 禁止软件从设备管理 1: 使能软件从设备管理 注: I <sup>2</sup> S 模式下不使用。
8	SSI	RW	0	内部从设备选择 (Internal slave select)

Bit	Name	R/W	Reset Value	Function
				该寄存器只有当 SSM=1 时才有效。该寄存器决定了 NSS 上的电平，在 NSS 引脚上的 I/O 操作无效。 注：I <sup>2</sup> S 模式下不使用。
7	LSBFIRST	RW	0	帧格式 0：先发送 MSB 1：先发送 LSB 通讯进行时不能改变该寄存器的值。 注：I <sup>2</sup> S 模式下不使用。
6	SPE	RW	0	SPI 使能 0：禁止 SPI 1：使能 SPI 注：I <sup>2</sup> S 模式下不使用。
5: 3	BR[2: 0]	RW	0	波特率控制 000: f <sub>PCLK</sub> /2 001: f <sub>PCLK</sub> /4 010: f <sub>PCLK</sub> /8 011: f <sub>PCLK</sub> /16 100: f <sub>PCLK</sub> /32 101: f <sub>PCLK</sub> /64 110: f <sub>PCLK</sub> /128 111: f <sub>PCLK</sub> /256 通讯进行时不能改变该寄存器的值。 注：I <sup>2</sup> S 模式下不使用。从机模式下，最快波特率仅支持 f <sub>PCLK</sub> /4。
2	MSTR	RW	0	主设备选择 0：配置为从设备 1：配置为主设备 通讯进行时不能改变该寄存器的值。 注：I <sup>2</sup> S 模式下不使用。
1	CPOL	RW	0	时钟极性 0：空闲状态时，SCK 保持低电平 1：空闲状态时，SCK 保持高电平 通讯进行时不能改变该寄存器的值。 注：I <sup>2</sup> S 模式下不使用。
0	CPHA	RW	0	时钟相位 0：数据采样从第一个时钟边沿开始 1：数据采样从第二个时钟边沿开始 通讯进行时不能改变该寄存器的值。

Bit	Name	R/W	Reset Value	Function
				注：I <sup>2</sup> S 模式下不使用。

### 31.5.2. SPI 控制寄存器 2 (SPI\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								TXEIE	RXNEIE	ERRIE	CLRTXFIFO	Res	SSOE	TXDMAEN	RXDMAEN
-								RW	RW	RW	RW	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 8	保留	-	-	保留
7	TXEIE	RW	0	发送缓冲区空中断使能 0: 禁止 TXE 中断 1: 使能 TXE 中断。TXE=1时产生中断请求。
6	RXNEIE	RW	0	接收缓冲区非空中断使能 0: 禁止 RXNE 中断 1: 使能 RXNE 中断。RXNE=1时产生中断请求。
5	ERRIE	RW	0	错误中断使能 当错误 (CRCERR、OVR、MODF) 产生时, 该位控制是否产生中断 0: 禁止错误中断 1: 使能错误中断。
4	CLRTXFIFO	RW	0	清空 TXFIFO 软件置位, 硬件复位 0: 没作用 1: 清空 TXFIFO 注: 只有当 SPI 禁止 (SPE=0) 时, 才能写该位, 否则无效。
3	Res	-	-	保留
2	SSOE	RW	0	SS 输出使能 0: 禁止在主模式下 SS 输出, 该设备可以工作在多主设备模式 1: 开启主模式下 SS 输出, 该设备不能工作在多主设备模式。 注: I <sup>2</sup> S 模式下不使用。
1	TXDMAEN	RW	0	发送缓冲区 DMA 使能

Bit	Name	R/W	Reset Value	Function
				当该位被设置时, TXE 标志一旦被置位就发出 DMA 请求 0: 禁止发送缓冲区 DMA 1: 使能发送缓冲区 DMA。
0	RXDMAEN	RW	0	接收缓冲区 DMA 使能 当该位被设置时, RXNE 标志一旦被置位就发出 DMA 请求 0: 禁止接收缓冲区 DMA 1: 使能接收缓冲区 DMA。

### 31.5.3. SPI 状态寄存器 (SPI\_SR)

Address offset: 0x08

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			FTLVL [1: 0]		FRLVL [1: 0]		Res	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
-			R	R	R	R	-	R	R	R	RC_W0	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15: 13	保留	-	-	保留
12: 11	FTLVL	R	0	FIFO 发送 level。硬件置位, 硬件清零 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full (当 FIFO 阈值大于1/2, 即认为是满) 注意: 该位在 I <sup>2</sup> S 模式下不使用。
10: 9	FRLVL	R	0	FIFO 接收 level。硬件置位, 硬件清零 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO 满 注意: 这些位在 I <sup>2</sup> S 模式和带 CRC 校验的 SPI 仅接收模式下不使用。
8	保留	-	-	保留
7	BSY	R	0	忙标志。 0: SPI 不忙 1: SPI 处于通讯, 或者发送缓冲非空。 该位由硬件置位或者复位。
6	OVR	R	0	溢出标志

Bit	Name	R/W	Reset Value	Function
				0: 没有出现溢出错误 1: 出现溢出错误 该位由硬件置位, 由软件序列复位。(上溢和下溢序列不同)。
5	MODF	R	0	模式错误 0: 没有出现模式错误 1: 出现模式错误 该位由硬件置位, 由软件序列复位。 注: I <sup>2</sup> S 模式下不使用。
4	CRCERR	RC_W0	0	CRC 错误标志 (CRC error flag) 0: 收到的 CRC 值和 SPI_RXCRCR 寄存器中的值匹配 1: 收到的 CRC 值和 SPI_RXCRCR 寄存器中的值不匹配 该位由硬件置位, 由软件写 '0' 而复位。 注: I <sup>2</sup> S 模式下不使用。
3	UDR	R	0	下溢标志位 (Underrun flag) 0: 未发生下溢 1: 发生下溢 该标志位由硬件置 '1', 由一个软件序列清 '0'。 注: 在 SPI 模式下不使用。
2	CHSIDE	R	0	声道 0: 需要传输或者接收左声道; 1: 需要传输或者接收右声道。 注: 在 SPI 模式下不使用。在 PCM 模式下无意义。
1	TXE	R	1	发送缓冲空。 0: 发送缓冲非空 1: 发送缓冲为空
0	RXNE	R	0	接收缓冲非空。 0: 接收缓冲非空 1: 接收缓冲为空

#### 31.5.4. SPI 数据寄存器 (SPI\_DR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DR[15: 0]	RW	0	数据寄存器。

				<p>要发送或者接收到的数据。</p> <p>数据寄存器作为 RxFIFO 和 TxFIFO 的接口。当要读数据，实际访问 RxFIFO，而要写数据，实际访问 TxFIFO。</p> <p>注意：数据始终是右对齐的。写入寄存器时忽略未使用的位，读取寄存器时读取为零。Rx 阈值设置必须始终与当前使用的读取访问相对应。</p>
--	--	--	--	--

### 31.5.5. SPI CRC 多项式寄存器 (SPI\_CRCPR)

Address offset: 0x10

Reset value: 0x0000 0007

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

Bit	Name	R/W	Reset Value	Function
15: 0	CRCPOLY[15: 0]	RW	0x7	<p>CRC 多项式寄存器 (CRC polynomial register)</p> <p>该寄存器包含了 CRC 计算时用到的多项式。其复位值为0x0007，根据应用可以设置其他数值。</p> <p>注：在 I<sup>2</sup>S 模式下不使用。</p> <p>注：多项式值只能是奇数，不支持偶数值。</p>

### 31.5.6. SPI Rx CRC 寄存器 (SPI\_RXCRCR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	RxCRC[15: 0]	R	0	<p>接收 CRC 寄存器</p> <p>在启用 CRC 计算时，RXCR[15: 0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CR1 的 CRCEN 位写入 '1' 时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。当数据帧格式被设置为8位时，仅低8位参与计算，并且按照 CRC8的方法进行；当数据帧格式</p>

Bit	Name	R/W	Reset Value	Function
				为16位时，寄存器中的所有16位都参与计算，并且按照 CRC16的标准。 注：当 BSY 标志为 '1' 时读该寄存器，将可能读到不正确的数值。 注：在 I2S 模式下不使用。

### 31.5.7. SPI Tx CRC 寄存器 (SPI\_TXCRCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	TxCRC[15: 0]	R	0	发送 CRC 寄存器 在启用 CRC 计算时，TXCRC[15: 0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPI_CR1中的 CRCEN 位写入 '1' 时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。 当数据帧格式被设置为8位时，仅低8位参与计算，并且按照 CRC8的方法进行；当数据帧格式为16位时，寄存器中的所有16个位都参与计算，并且按照 CRC16的标准。 注：当 BSY 标志为 '1' 时读该寄存器，将可能读到不正确的数值。 注：在 I2S 模式下不使用。

### 31.5.8. SPI\_I2S 配置寄存器 (SPI\_I2S\_CFGR)

Address offset: 0x1C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				I2SMOD	I2SE	I2SCFG	PCMSYNC	Res	I2SSTD	CKPOL	DATLEN	CHLEN			
-				RW	RW	RW	RW	-	RW	RW	RW	RW			

Bit	Name	R/W	Reset Value	Function
15: 12	保留	-	-	保留
11	I2SMOD	RW	0	I2S 模式选择

Bit	Name	R/W	Reset Value	Function
				0: 选择 SPI 模式 1: 选择 I2S 模式 注: 该位只有在关闭了 SPI 或者 I2S 时才能设置。
10	I2SE	RW	0	I2S 使能 0: 关闭 I2S; 1: I2S 使能。 注: 在 SPI 模式下不使用。
9: 8	I2SCFG	RW	0	I2S 模式设置 00: 从设备发送 01: 从设备接收 10: 主设备发送 11: 主设备接收 注: 该位只有在关闭了 I <sup>2</sup> S 时才能设置。 在 SPI 模式下不使用。
7	PCMSYNC	RW	0	PCM 帧同步 0: 短帧同步 1: 长帧同步 注: 该位只在 I2SSTD = 11 (使用 PCM 标准) 时有意义。 在 SPI 模式下不使用。
6	保留	-	-	保留
5: 4	I2SSTD	RW	0	I2S 标准选择 00: I2S 飞利浦标准 01: 高字节对齐标准 (左对齐) 10: 低字节对齐标准 (右对齐) 11: PCM 标准 注: 为了正确操作, 只有在关闭了 I2S 时才能设置该位。 在 SPI 模式下不使用。
3	CKPOL	RW	0	静止态时钟极性 0: I2S 时钟静止态为低电平 1: I2S 时钟静止态为高电平 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。 CKPOL 位不影响接收或发送 SD 和 WS 信号所使用的 CK 边沿灵敏度
2: 1	DATLEN	RW	0	待传输数据长度



Bit	Name	R/W	Reset Value	Function
				00: 16位数据长度 01: 24位数据长度 10: 32位数据长度 11: 不允许 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
0	CHLEN	RW	0	声道长度 (每个音频声道的数据位数) 0: 16位宽 1: 32位宽 只有在 DATLEN = 00 时该位的写操作才有意义, 否则声道长度都由硬件固定为32位。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。

### 31.5.9. SPI\_I2S 预分频寄存器 (SPI\_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						MCKOE	ODD	I2SDIV							
-						RW	RW	RW							

Bit	Name	R/W	Reset Value	Function
15: 10	保留	-	-	保留
9	MCKOE	RW	0	主设备时钟输出使能 0: 关闭主设备时钟输出 1: 主设备时钟输出使能 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。 在 SPI 模式下不使用。
8	ODD	RW	0	奇系数预分频 0: 实际分频系数 = I2SDIV * 2 1: 实际分频系数 = (I2SDIV * 2) + 1 注: 为了正确操作, 该位只有在关闭了 I <sup>2</sup> S 时才能设置。仅在 I <sup>2</sup> S 主设备模式下使用该位。 在 SPI 模式下不使用。
7: 0	I2SDIV	RW	0x2	I <sup>2</sup> S 线性预分频

Bit	Name	R/W	Reset Value	Function
				<p>禁止设置 I2SDIV [7: 0] = 0或者 I2SDIV [7: 0] = 1</p> <p>注：为了正确操作，该位只有在关闭了 I<sup>2</sup>S 时才能设置。仅在 I<sup>2</sup>S 主设备模式下使用该位。在 SPI 模式下不使用。</p>

## 32. 通用同步异步收发器 (USART)

### 32.1. 介绍

通用同步异步收发器 (USART) 提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用小数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信也支持 LIN (局部互联网), 智能卡协议和 IrDA (红外数据组织) ENDEC 规范, 以及调制解调器 (CTS/RTS) 操作。它还允许多处理器通信。

使用多缓冲器配置的 DMA 方式, 可以实现高速数据通信。

### 32.2. USART 主要特性

- 2 个支持全功能的 USART (USART1 和 USART2), 两个不支持 LIN, SCEN, IRDA 的 USART (USART3 和 USART4)
- 全双工异步通信
- 发送和接收共用的可编程波特率, 最高达 4.5 Mbit/s
- 可配置的数据字长度 (8 位或者 9 位)
- 可配置的停止位— 支持 0.5, 1, 1.5 或 2 个停止位
- 发送方为同步传输提供时钟
- 单线半双工通信
- 单独的发送器和接收器使能位
- 奇偶校验控制
  - 发送校验位
  - 对接收数据进行校验
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- Lin 主发送同步断开符的能力以及 Lin 从检测断开符的能力
  - 当 USART 硬件配置成 LIN 时, 生成13位断开符, 检测10/11断开符
- IRDA SIR 编码器解码器
  - 在正常模式下支持3/16位的持续时间
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3标准里定义的异步智能卡协议
  - 智能卡用到的0.5和1.5个停止位
- 四个错误检测标志
  - 溢出错误
  - 噪音错误
  - 帧错误
  - 校验错误
- 10 个带标志的中断源

- CTS 改变
  - LIN 断开符检测
  - 发送数据寄存器为空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪音错误
  - 校验错误
  - 多处理器通信。如果地址不匹配，则进入静默模式
  - 从静默模式中唤醒（通过空闲总线检测或地址标志检测）
- 两种唤醒接收器的方式：地址位（MSB,第 9 位），总线空闲

### 32.3. USART 功能描述

USART 接口通过三个引脚与其他设备连接在一起。任何 USART 双向通信至少需要两个脚：接收数据输入（RX）和发送数据输出（TX）。

**RX:** 接收数据串行输。通过过采样技术来区别数据和噪音，从而恢复数据。

**TX:** 发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同时用于数据的发送和接收。

- 1) 总线在发送或接收前应处于空闲状态
- 2) 一个起始位
- 3) 一个数据字（8或9位），最低有效位在前
- 4) 0.5, 1, 1.5, 2个的停止位，由此表明数据帧的结束
- 5) 使用分数波特率发生器：12位整数和4位小数的表示方法
- 6) 一个状态寄存器（USART\_SR）
- 7) 数据寄存器（USART\_DR）
- 8) 一个波特率寄存器（USART\_BRR），12位的整数和4位小数
- 9) 一个智能卡模式下的保护时间寄存器（USART\_GTPR）

在同步模式中需要下列引脚：

#### CK: 发送器时钟输出。

此引脚输出用于同步传输的时钟，（在 Start 位和 Stop 位上没有时钟脉冲，软件可选地，可以在最后一个数据位送出一个时钟脉冲）。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备（例如 LCD 驱动器）。时钟相位和极性都是软件可编程的。

下列引脚在硬件流控模式中需要：

- **nCTS:** 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- **nRTS:** 发送请求，若是低电平，表明 USART 准备好接收数据。

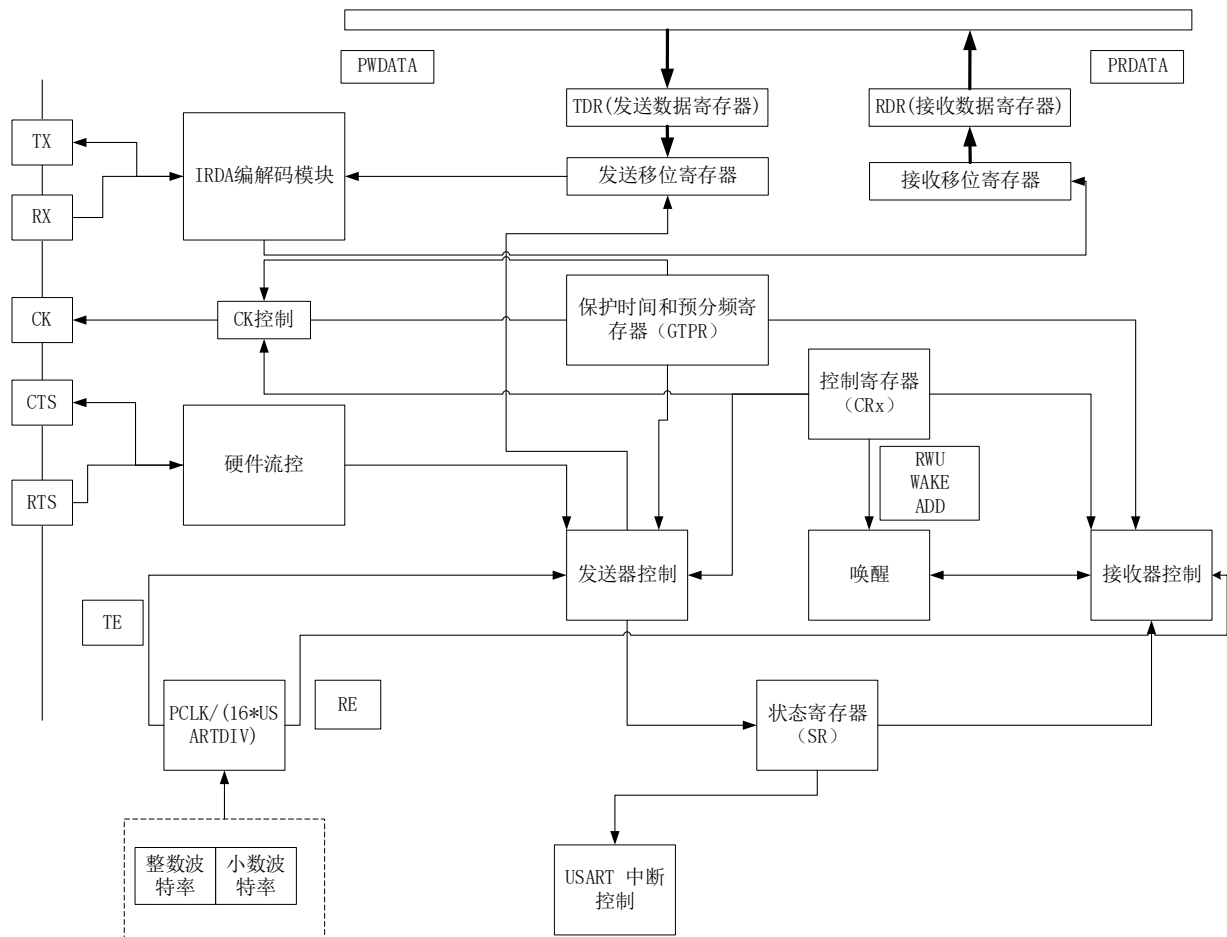


图32-1 USART 框图

### 32.3.1. USART 特征描述

字长可以通过编程 USART\_CR1 寄存器中的 M 位，选择成 8 或 9 位。在起始位期间，TX 脚处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由'1'组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位（'1'的位数也包括了停止位的位数）。

断开符号被视为在一个帧周期内全部收到'0'（包括停止位期间，也是'0'）。在断开符号结束时，发送器再插入 1 或 2 个停止位（'1'）来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。

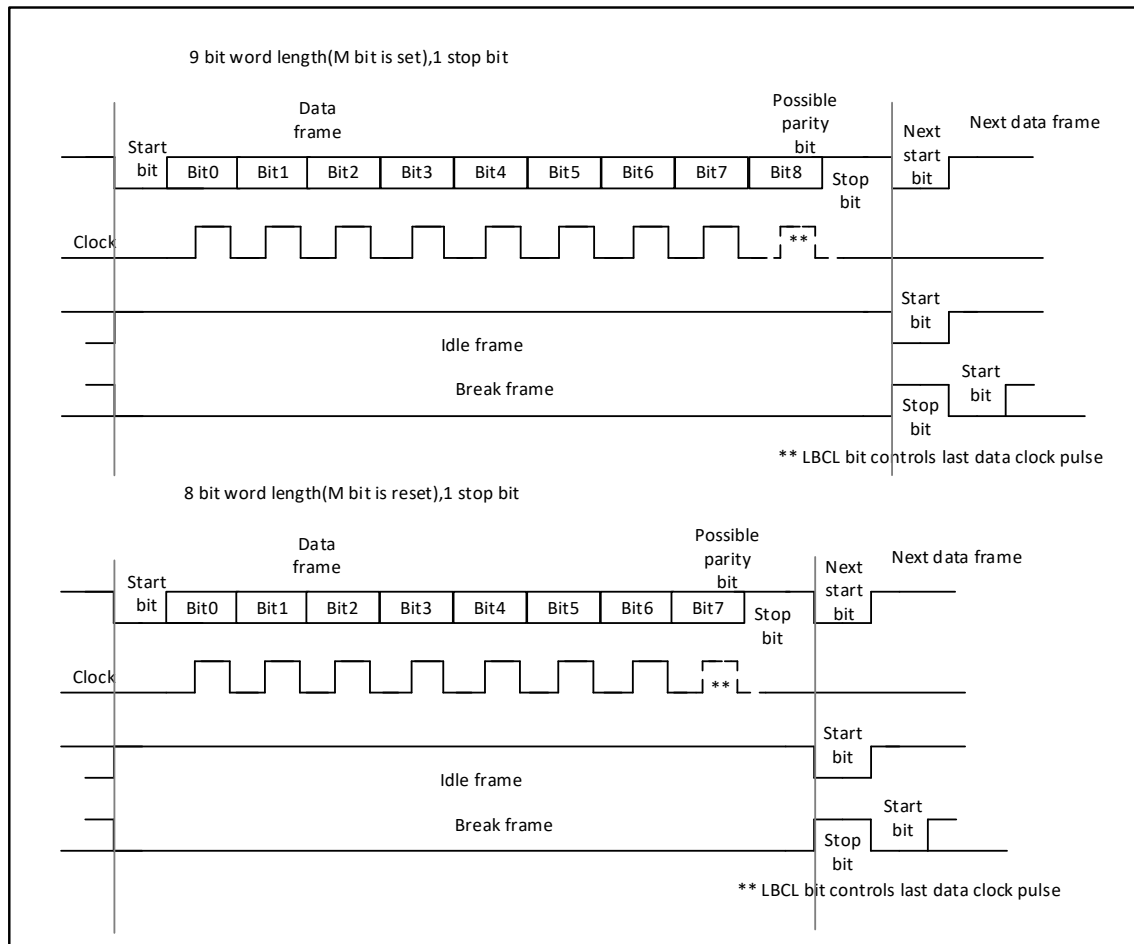


图 32-2 字长设置

### 32.3.2. 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位 (TE) 被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

#### 32.3.2.1. 字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式 USART\_DR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

注：

在数据传输期间不能复位 TE 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

TE 位被激活后将发送一个空闲帧。

#### 32.3.2.2. 可配置的停止位

随每个字符发送的停止位的位数可以通过控制寄存器 2 的位 13、12 进行编程。

- 1) 1 个停止位：停止位位数的默认值。
- 2) 2 个停止位：可用于常规 USART 模式、单线模式以及调制解调器模式。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位（当  $m=0$  时）；或者 11 位低电平，后跟停止位（ $m=1$  时）。不可能传输更长的断开帧（长度大于 10 或者 11 位）。

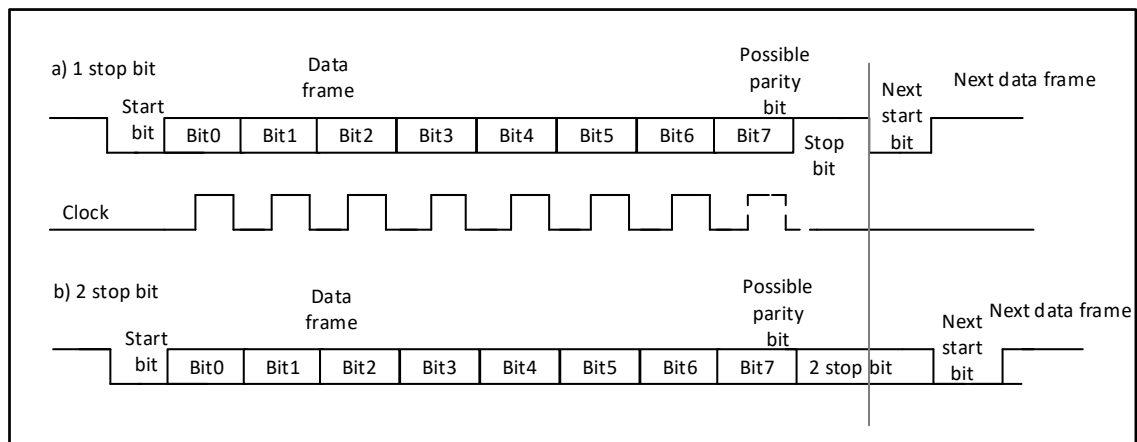


图 32-3 配置停止位

配置步骤：

- 1) 通过在 USART\_CR1 寄存器上置位 UE 位来激活 USART
- 2) 编程 USART\_CR1 的 M 位来定义字长。
- 3) 在 USART\_CR2. STOP 中编程停止位的位数。
- 4) 如果采用多缓冲器通信，配置 USART\_CR3 中的 DMA 使能位 (DMAT)。按多缓冲器通信中的描述配置 DMA 寄存器。
- 5) 利用 USART\_BRR 寄存器选择要求的波特率。
- 6) 设置 USART\_CR1 中的 TE 位，发送一个空闲帧作为第一次数据发送。
- 7) 把要发送的数据写进 USART\_DR 寄存器（此动作清除 TXE 位）。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 7。
- 8) 在 USART\_DR 寄存器中写入最后一个数据字后，要等待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次传输。

### 32.3.2.3. 单字节通信

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空
- 下一个数据可以被写进 USART\_DR 寄存器而不会覆盖先前的数据

如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USART\_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USART\_DR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE 位立即被置起。

当一帧发送完成时（停止位发送后）并且设置了 TXE 位，TC 位被置起，如果 USART\_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。

在 USART\_DR 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式（详见下图）之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1. 读一次 USART\_SR 寄存器；
2. 写一次 USART\_DR 寄存器。

注：TC 位也可以通过软件对它写 '0' 来清除。此清零方式只推荐在多缓冲器通信模式下使用。

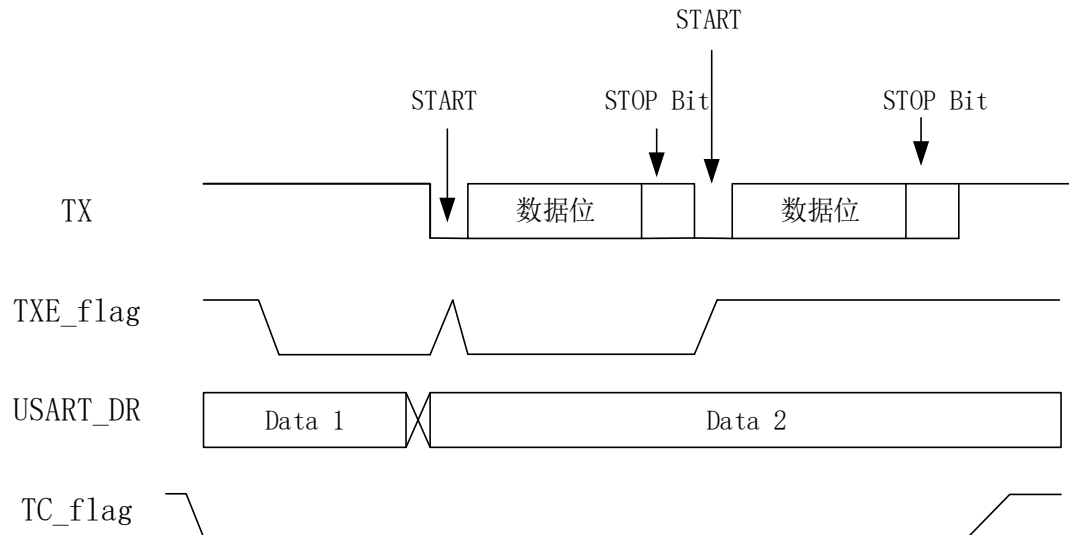


图 32-4 USART 发送时 TC/TXE 的变化

#### 32.3.2.4. 断开符号

设置 SBK 可发送一个断开符号。断开帧长度取决 M 位。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开符号发送完成时（在断开符号的停止位时）SBK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑 '1'，以保证能识别下一帧的起始位。

注意：如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置起。

#### 32.3.2.5. 空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

### 32.3.3. 接收器

USART 可以根据 USART\_CR1 的 M 位接收 8 位或 9 位的数据字。

#### 32.3.3.1. 开始位检测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。该序列为：1 1 1 0 X 0 X 0 X 0 0 0 0



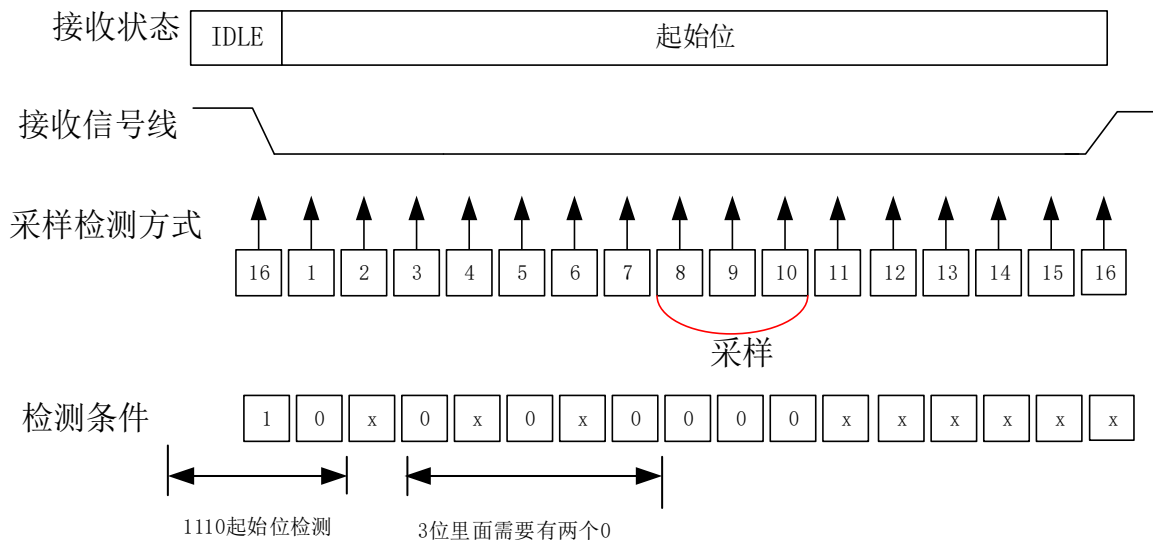


图 32-5 开始位检测

如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态（不设置标志位）等待下降沿。如果3个采样点都为'0'（在第3、5、7位的第一次采样，和在第8、9、10位的第二次采样都为'0'），则确认收到起始位，这时设置 RXNE 标志位，如果 RXNEIE=1，则产生中断。

如果两次3个采样点上仅有2个是'0'（第3、5、7位的采样点和第8、9、10位的采样点），那么起始位仍然是有效的，但是会设置 NE 噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态（不设置标志位）。

如果有一次3个采样点上仅有2个是'0'（第3、5、7位的采样点或第8、9、10位的采样点），那么起始位仍然是有效的，但是会设置 NE 噪声标志位。

### 32.3.3.2. 字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USART\_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 USART\_CR1寄存器的 UE 置1来激活 USART。
2. 编程 USART\_CR1的 M 位定义字长
3. 在 USART\_CR2. STOP 中编写停止位的个数
4. 如果需多缓冲器通信，选择 USART\_CR3中的 DMA 使能位 (DMAR)。按多缓冲器通信所要求的配置 DMA 寄存器。
5. 利用波特率寄存器 USART\_BRR 选择希望的波特率。
6. 设置 USART\_CR1的 RE 位。激活接收器，使它开始寻找起始位。

当一个字符被接收到时：

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出（包括与之有关的错误标志）。
- 如果 RXNEIE 位被设置，产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起
- 在多缓冲器通信时，RXNE 在每个字节接收后被置起，并由 DMA 对数据寄存器的读操作而清零。

- 在单缓冲器模式里，由软件读 USART\_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

注意：在接收数据时，RE 位不应该被复位。如果 RE 位在接收时被清零，当前字节的接收被丢失。

### 32.3.3.3. 断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

### 32.3.3.4. 空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

### 32.3.3.5. 溢出错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当 RXNE 位被清零后才能从移位寄存器转移到 RDR 寄存器。RXNE 标记是接收到每个字节后被置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RXNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：ORE 位被置位。

- RDR 内容将不会丢失。读 USART\_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USART\_SR 和 USART\_DR 寄存器的读操作，可复位 ORE 位

注意：当 ORE 位置位时，表明至少有 1 个数据已经丢失。有两种可能性：

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。
- 如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（也就是丢失的）数据时，此种情况可能发生。在读序列期间（在 USART\_SR 寄存器读访问和 USART\_DR 读访问之间）接收到新的数据，此种情况也可能发生。

### 32.3.3.6. 噪音错误

使用过采样技术（同步模式除外），通过区别有效输入数据和噪音来进行数据恢复。

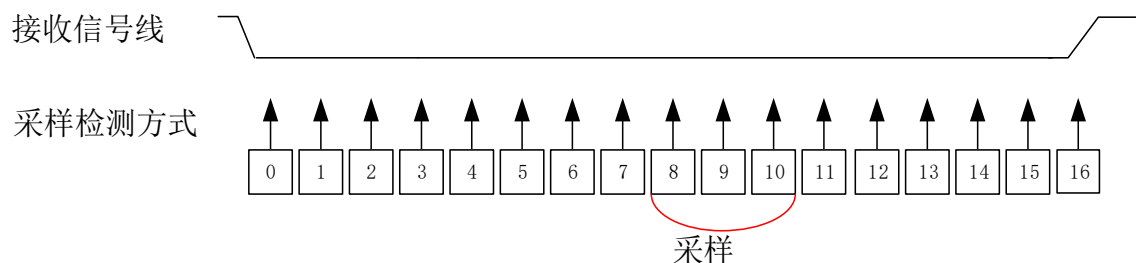


图 32-6 检测噪声的数据采样

表 32-1 检测噪声的数据采样

采样值	NE 状态	接收的位值	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志
- 无效数据从移位寄存器传送到 USART\_DR 寄存器
- 在单个字节通信情况下，没有中断产生。然而，因为 NE 标志位和 RXNE 标志位是同时被设置，RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USART\_CR3 寄存器中 EIE 位，将产生一个中断
- 先读出 USART\_SR，再读出 USART\_DR 寄存器，将清除 NE 标志位

#### 32.3.3.7. 帧错误

当以下情况发生时检测到帧错误：

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。

当帧错误被检测到时：

- FE 位被硬件置起
- 无效数据从移位寄存器传送到 USART\_DR 寄存器
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 位同时置起，后者将产生中断。在多缓冲器通信情况下，如果 USART\_CR3 寄存器中 EIE 位被置位的话，将产生中断
- 顺序执行对 USART\_SR 和 USART\_DR 寄存器的读操作，可复位 FE 位

#### 32.3.3.8. 接收期间的可配置的停止位

被接收的停止位的个数可以通过控制寄存器2的控制位来配置，在正常模式时，可以是1或2个，在智能卡模式里可能是0.5或1.5个。

- 0.5 个停止位（智能卡模式中的接收）：不对 0.5 个停止位进行采样。因此，如果选择 0.5 个停止位则不能检测帧错误和断开帧
- 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行
- 1.5 个停止位（智能卡模式）：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活（USART\_CR1 寄存器中的 RE =1），并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样 NACK 信号时，即总线上停止位对应的时间时，拉低数据线，以此表示出现了帧错误。FE 在 1.5 个停止位结束时和 RXNE 一起被置起。对 1.5 个停止位的采样是在第 16，第 17 和第 18 采样点进行的。1.5 个的停止位可以被分成 2 部分：一个是 0.5 个时钟周期，期间不做任何事情。随后是 1 个时钟周期的停止位，在这段时间的中点处采样。

- 2个停止位：对2个停止位的采样是在第一停止位的第8，第9和第10个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时RXNE标志将被设置。

### 32.3.4. 分数波特率的产生

分数波特率的产生接收器和发送器的波特率在USARTDIV的整数和小数寄存器中的值应设置成相同。

$$Tx / Rx \text{ 波特率} = f_{CK} / (16 * USARTDIV)$$

这里的 $f_{CK}$ 是给外设的时钟USARTDIV是一个无符号的定点数。这12位的值设置在USART\_BRR寄存器。

注：在写入USART\_BRR之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

#### 如何从USART\_BRR寄存器值得到USARTDIV

##### 例1：

如果 DIV\_Mantissa = 27, DIV\_Fraction = 12 (USART\_BRR=0x1BC),  
则：

$$\text{Mantissa (USARTDIV)} = 27$$

$$\text{Fraction (USARTDIV)} = 12/16 = 0.75$$

所以 USARTDIV = 27.75

##### 例2：

要求 USARTDIV = 25.62,

就有：

$$\text{DIV\_Fraction} = 16 * 0.62 = 9.92$$

最接近的整数是：10 = 0x0A

$$\text{DIV\_Mantissa} = \text{mantissa} (25.620) = 25 = 0x19$$

于是, USART\_BRR = 0x19A

##### 例3：

要求 USARTDIV = 50.99

就有：

$$\text{DIV\_Fraction} = 16 * 0.99 = 15.84$$

最接近的整数是：16 = 0x10 => DIV\_frac[3: 0]溢出 => 进位必须加到小数部分

$$\text{DIV\_Mantissa} = \text{mantissa} (50.990 + \text{进位}) = 51 = 0x33$$

于是：USART\_BRR = 0x330, USARTDIV=51

波特率		f <sub>PCLK</sub> =36 MHz			f <sub>PCLK</sub> =72 MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差 (%)	实际	置于波特率寄存器中的值	误差 (%)
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%

6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%

注：CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。

### 32.3.5. USART 接收器容忍度

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化（包括发送器端振荡器的变化）
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化（通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成）。

需要满足： $DTRA + DQUANT + DREC + DTCL < USART$  接收器的容忍度。

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USART\_CR1 寄存器的 M 位定义的 10 或 11 位字符长度

是否使用分数波特率产生表 32-2 DIV\_Fraction 为 0 时 USART 接收器的容忍度

M bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

表 32-3 DIV\_Fraction 非 0 时 USART 接收器容忍度

M bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

### 32.3.6. USART 自动波特率检测

USART 能够基于一个字符的接收，检测并自动设定 USARTx\_BRR 寄存器的值。自动波特率检测在以下场景是有用处的：

- 系统通讯速率提前未确认
  - 系统正在使用相对低精度的时钟源，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。时钟源频率必须与被期望的通讯速率兼容。（必须选择 16 倍过采样，并从 fCK/65535 和 fCK/16 之间选择）
- 在激活自动波特率检测之前，自动波特率检测模式必须被选择（通过 USARTx\_CR3 寄存器的

ABRM0D[1:0] 位被选择）。基于不同的字符模式，有各种模式。

在这些自动波特率模式下，波特率在同步数据接收期间会被测量几次，每次测量都会跟之前进行对比。

这些模式是：

**MODE 0**：任何以 1 开始的字符。在这种情况下，USART 测量起始位的宽度（下降沿到上升沿）

**MODE 1:** 任何以10xx 位开始的字符。在这种情况下，USART 测量起始位和第一个数据位的宽度。该测量在下降沿到下降沿之间进行，以确保在慢速信号上升情况下更好的精度。

另一个对每个 RX 的 transition 的检查也会并行进行。如果 RX 上的 transition 没有与接收器充分的同步（接收器基于 bit 0计算的波特率），则会产生错误。

在激活自动波特率检测之前，USARTx\_BRR 寄存器必须通过写一个 non-zero 波特率值进行初始化。

通过置位 USARTx\_CR3寄存器的 ABREN 位，可以激活自动波特率检测功能。USART 将等待 RX 上的第一个字符。置位 USARTx\_ISR 寄存器的 ABRF 标志，显示了自动波特率检测的完成。如果通讯线上是有噪声的，则自动波特率检测的正确进行不能被保证。在这种情况下，BRR 的值可能被破坏，ABRE 错误标志位将被置位。如果通讯速度与自动波特率检测范围不兼容（位宽不在16和65535个时钟周期之间@16位过采样），ABRE 错误也会发生。

RXNE 中断将显示了操作的完成。在以后的任何时刻，自动波特率检测可能通过复位 ABRF 标志（通过写 0）再次启动。

注：如果在自动波特期间，清零 UE，BRR 值可能被破坏。

### 32.3.7. 多处理器通信

通过 USART 可以实现多处理器通信（将几个 USART 连在一个网络里）。例如某个 USART 设备可以是主，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；USART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据，这样就可以减少由未被寻址的接收器的参与带来的多余的 USART 服务开销。

未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USARTx\_CR1寄存器中的 RWU 位被置1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

根据 USARTx\_CR1寄存器中的 WAKE 位状态，USARTx 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

#### 32.3.7.1. 空闲总线检测 (WAKE=0)

当 RWU 位被写1时，USART 进入静默模式。当检测到一空闲帧时，它被唤醒。然后 RWU 被硬件清零，但是 USART\_SR 寄存器中的 IDLE 位并不置起。RWU 还可以被软件写0。下图给出利用空闲总线检测来唤醒和进入静默模式的一个例子

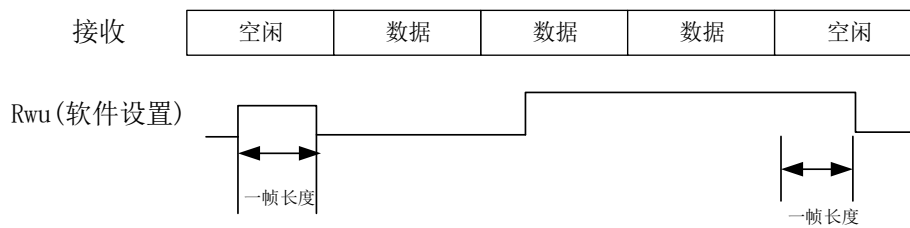


图 32-7 利用空闲总线检测的静默模式

### 32.3.7.2. 地址标记 (Address mark) 检测 (WAKE=1)

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USART\_CR2 寄存器的 ADD。

如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。此时，硬件设置 RWU 位。

接收该字节既不会设置 RXNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时 (USART\_SR 的 RXNE=0)，RWU 位可以被写 0 或 1。否则，该次写操作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

在本例中地址为 3，

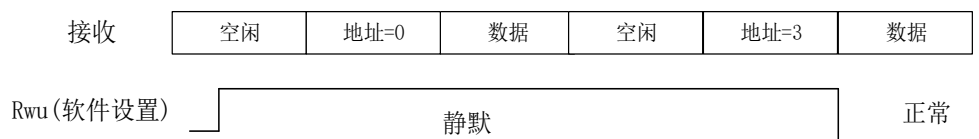


图 32-8 利用地址标记检测的静默模式

### 32.3.7.3. 校验控制

设置 USART\_CR1 寄存器上的 PCE 位，可以使能奇偶控制（发送时生成一个奇偶位，接收时进行奇偶校验）。根据 M 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 32-4 帧格式

M bit	PCE bit	USART fram
0	0	SB—8 bit data—STB
0	1	SB—7 bit data—PB—STB
1	0	SB—9 bit data—STB
1	1	SB—8 bit data—PB—STB

在用地址标记唤醒设备时，地址的匹配只考虑到数据的 MSB 位，而不用关心校验位。（MSB 是数据位中最后发出的，后面紧跟校验位或者停止位）

### 32.3.7.4. 偶校验

校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中‘1’的个数为偶数。

例如：数据=00110101，有 4 个‘1’，如果选择偶校验（在 USARTx\_CR1 中的 PS = 0），校验位将是‘0’。

### 32.3.7.5. 奇校验

此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中‘1’的个数为奇数。

例如：数据=00110101，有 4 个‘1’，如果选择奇校验（在 USARTx\_CR1 中的 PS = 1），校验位将是‘1’。

### 32.3.7.6. 传输模式

如果 USARTx\_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去（如果选择偶校验偶数个‘1’，如果选择奇校验奇数个‘1’）。如果奇偶校验失败，USART\_SR 寄存器中的 PE 标志被置‘1’，并且如果 USART\_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

### 32.3.8. LIN (局域网) 模式

配置 USART\_CR2.LINEN=1 选择 LIN 模式。在 LIN 模式下，下列位必须保持为 0：

- USART\_CR2 寄存器的 CLKEN;
- USART\_CR3 寄存器的 STOP/SCEN/HDSEL/IREN.

#### 32.3.8.1. 发送

与一般 USART 发送相比，LIN 发送有如下区别：

M=0，数据长度为 8bit；

需要配置 USART\_CR2.LINEN=1。置位 SBK 后将发送 13bit 0 作为 break 帧。然后发送一位“1”，以允许对下一个 start 位检测。

#### 32.3.8.2. 接收

当 LIN 模式被使能时，断开符号检测电路被激活。该检测完全独立于 USART 接收器。断开符号只要一出现就能检测到，不管是在总线空闲时还是在发送某数据帧其间，数据帧还未完成，又插入了断开符号的发送。

当接收器被激活时 (USART\_CR1 的 RE=1)，电路监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8, 9, 10 个过采样时钟点上进行采样。如果 10 个 (当 USART\_CR2 的 LBDL = 0) 或 11 个 (当 USART\_CR2 的 LBDL = 1) 连续位都是 0，并且又跟着一个定界符，USART\_SR 的 LBD 标志被设置。如果 LBDIE 位=1，中断产生。在确认断开符号前，要检查定界符，因为它意味 RX 线已经回到高电平。

如果在第 10 或 11 个采样点之前采样到了 1，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如正常 USART 那样工作，不需要考虑检测断开符号。

如果 LIN 模式没有被激活 (LINEN=0)，接收器仍然正常工作于 USART 模式，不会进行断开检测。

如果 LIN 模式被激活 (LINEN=1)，只要一发生帧错误 (也就是停止位检测到‘0’，这种情况出现在断开帧)，接收器就停止，直到断开符号检测电路接收到一个 1 (这种情况发生于断开符号没有完整的发出来)，或一个定界符 (这种情况发生于已经检测到一个完整的断开符号)。

本情况：断开帧长度不够：需要舍弃该帧，不设置 LBD

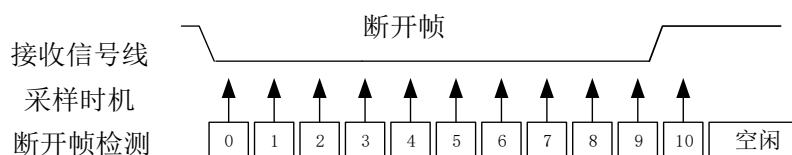


图 32-9 LIN 模式下的断开帧长度不够的情况



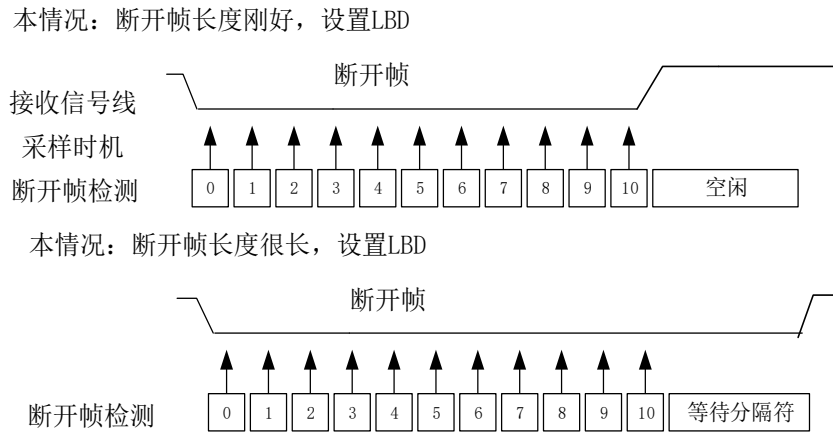
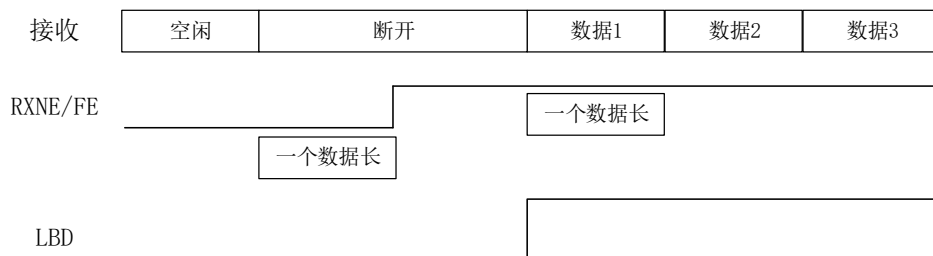


图 32-10 LIN 模式下的断开帧长度够的情况

断开发生在空闲后



断开发生在正在接收数据时

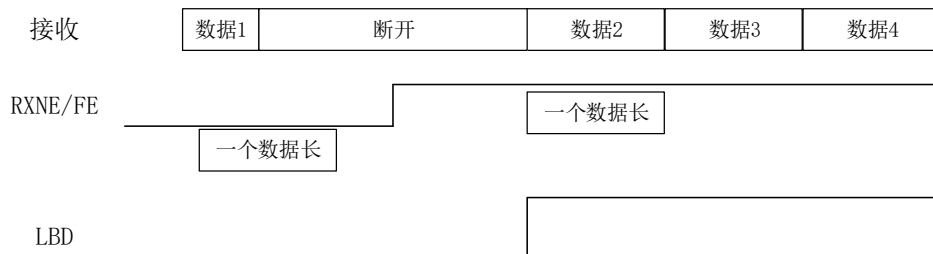


图 32-11 LIN 模式下的断开检测与帧错误的检测

### 32.3.9. USART 同步模式

通过写 USART\_CR2寄存器的 CLKEN 位为1，选择同步模式。在同步模式里，下列位必须保持清零状态：

USART\_CR2寄存器中的 LINEN 位

USART\_CR3寄存器中的 SCEN,HDSEL 和 IREN 位

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USART\_CR2寄存器中 LBCL 位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART\_CR2寄存器的 CPOL 位允许用户选择时钟极性，USART\_CR2寄存器上的 CPHA 位允许用户选择外部时钟的相位。

在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 CK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的（根据 CPOL 和 CPHA），所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 CK 上采样（根据 CPOL 和 CPHA 决定在上升沿还是下降沿），不需要任何的过采样。但必须考虑建立时间和持续时间（取决于波特率，1/16位时间）。

注意：

CK 脚同 TX 脚一起联合作。因而，只有在使能了发送器（TE = 1），并且发送数据时（写入数据至 USART\_DR 寄存器）才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。

LBCL,CPOL 和 CPHA 位的正确配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变。

建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。

USART 只支持主模式：它不能来自其他设备的输入时钟接收或发送数据（CK 永远是输出）。

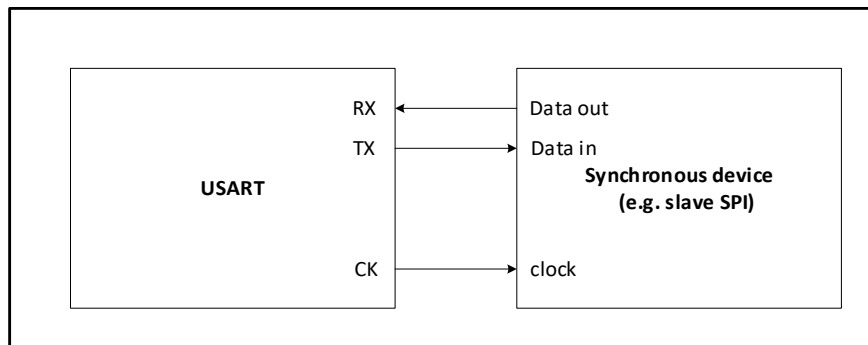


图 32-12 USART 同步传输的例子

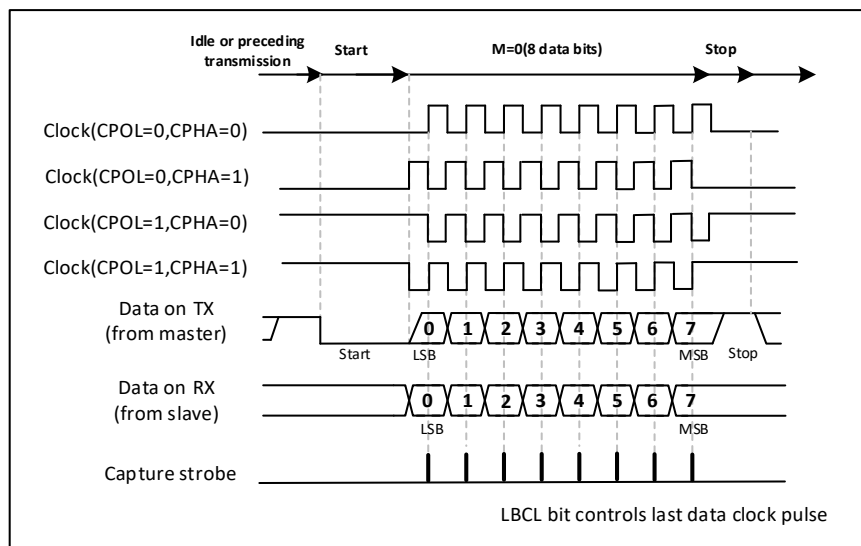


图 32-13 USART 数据时钟时序示例 (M=0)

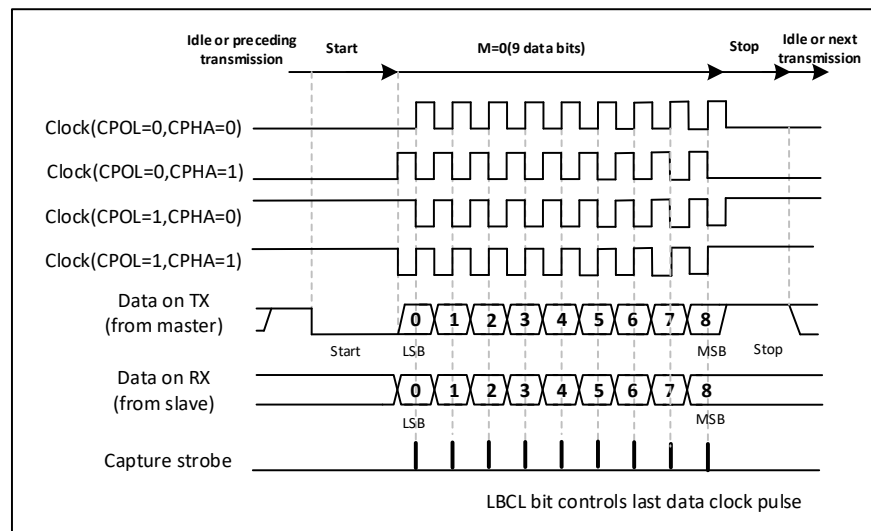


图 32-14 USART 数据时钟时序示例 (M=1)

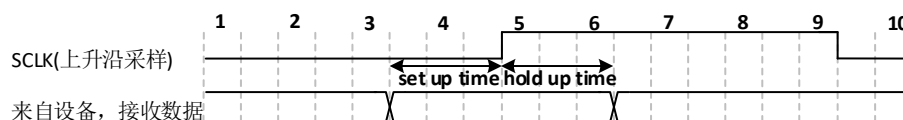


图 32-15 RX 数据采样/保持时间

### 32.3.10. 单线半双工通信

单线半双工模式通过设置 USARTx\_CR3寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USARTx\_CR2寄存器的 CLKEN 位
- USART\_CR3寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位“HALF DUPLEX SEL” (USARTx\_CR3中的 HDSEL 位) 选择半双工和全双工通信。

当 HDSEL 为‘1’时

- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入（或开漏的输出高）。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突（例如通过使用一个中央仲裁器）。特别是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

### 32.3.11. 智能卡

设置 USART\_CR3寄存器的 SCEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USART\_CR2寄存器的 LINEN 位
- USART\_CR3寄存器的 HDSEL 位和 IREN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

该接口符合 ISO7816-3标准，支持智能卡异步协议。USART 应该被设置为：

1. 8位数据位加校验位：此时 USART\_CR1寄存器中 M=1、PCE=1

2. 发送和接收时为1.5个停止位：即 USART\_CR2寄存器的 STOP=11

注：也可以在接收时选择0.5个停止位，但为了避免在2种配置间转换，建议在发送和接收时使用1.5个停止位。

下图给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

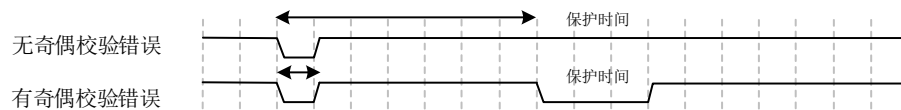


图 32-16 ISO7816-3异步协议

当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线。为了做到这点，SW\_RX 必须和 TX 连接到相同的 I/O 口。在发送开始位和数据字节期间，发送器的输出使能位 TX\_EN 被置起，在发送停止位期间被释放（弱上拉），因此在发现校验错误的情况下接收器可以将数据线拉低。如果 TX\_EN 不被使用，在停止位期间 TX 被拉到高电平：这样的话，只要 TX 配置成开漏，接收器也可以驱动这根线。

智能卡是一个单线半双工通信协议

- 1) 从发送移位寄存器把数据发送出去，要被延时最小1/2波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟1/2波特时钟。
- 2) 如果在接收一个设置为0.5或1.5个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后（即停止位结束时），发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确地接收到。此 NACK 信号（拉低发送线一个波特时钟周期）在发送端将产生一个帧错误（发送端被配置成1.5个停止位）。应用程序可以根据协议处理重新发送数据。如果设置了 NACK 控制位，发生校验错误时接收器会给出一个 NACK 信号；否则就不会发送 NACK。
- 3) TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC 被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC 在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。
- 4) TC 标志的撤销不受智能卡模式的影响。
- 5) 如果发送器检测到一个帧错误（收到接收器的 NACK 信号），发送器的接收功能模块不会把 NACK 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是1或2波特时钟周期。
- 6) 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意：

- 1) 断开符号在智能卡模式里没有意义。一个带帧错误的00h 数据将被当成数据而不是断开符号。
- 2) 当来回切换 TE 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

下图详述了 USART 是如何采样 NACK 信号的。在这个例子里，USART 正在发送数据，并且被配置成1.5个停止位。为了检查数据的完整性和 NACK 信号，USART 的接收功能块被激活。

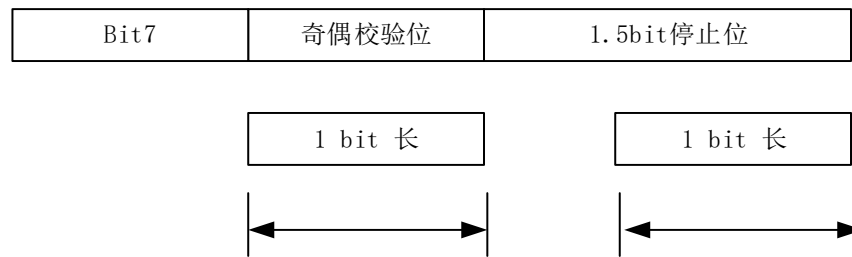


图 32-17 使用1.5停止位检测奇偶校验错

USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式里，CK 不和通信直接关联，而是先通过一个5位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 USART\_GTPR 中配置。CK 频率可以从  $f_{CK}/2$  到  $f_{CK}/62$ ，这里的  $f_{CK}$  是外设输入时钟。

### 32.3.12. IrDA SIR ENDEC 功能模块

通过设置 USART\_CR3寄存器的 IREN 位选择 IrDA 模式。在 IRDA 模式里，下列位必须保持清零：

- USART\_CR2寄存器的 LINEN,STOP 和 CLKEN 位
- USART\_CR3寄存器的 SCEN 和 HDSEL 位。

#### 32.3.12.1. IrDA 正常模式

IrDA SIR 物理层规定使用反相归零调制方案 (RZI)，该方案用一个红外光脉冲代表逻辑'0'。SIR 发送编码器对从 USART 输出的 NRZ (非归零) 比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外 LED。USART 为 SIR ENDEC 最高只支持到115.2Kbps 速率。在正常模式里，脉冲宽度规定为一个位周期的  $3/16$ 。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高 (标记状态 marking state)。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙 (也就是 USART 正在送数据给 IrDA 编码器)，IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙 (也就是 USART 正在接收从 IrDA 解码器来的解码数据)，从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。
- SIR 发送逻辑把'0'作为高脉冲发送，把'1'作为低电平发送。脉冲的宽度规定为正常模式时位周期的  $3/16$ 。
- SIR 接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于  $1.41\mu s$ 。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于2个 PSC 周期的脉冲 (PSC 是在 IrDA 低功耗波特率寄存器 USART\_GTPR 中编程的预分频值)。宽度小于1个 PSC 周期的脉冲一定被滤除掉，但是那些宽度大于1个而小于2个 PSC 周期的脉冲可能被接收或滤除，那些宽度大于2个周期的将被视为一个有效的脉冲。当  $PSC=0$  时，IrDA 编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。
- 在 IrDA 模式里，USART\_CR2寄存器上的 STOP 位必须配置成1个停止位。

### 32.3.12.2. IrDA 低功耗模式

#### 发送器:

在低功耗模式，脉冲宽度不再持续3/16个位周期。取而代之，脉冲的宽度是低功耗波特率的3倍，它最小可以是1.42MHz。通常这个值是1.8432MHz ( $1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$ )。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

#### 接收器:

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于1个 PSC 的脉冲。只有持续时间大于2个周期的 IrDA 低功耗波特率时钟 (USART\_GTPR 中的 PSC) 的低电平信号才被接受为有效的信号。

#### 注意:

1. 宽度小于2个大于1个 PSC 周期的脉冲可能会也可能不会被滤除。
2. 接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有10ms 的延时 (IrDA 是一个半双工协议)。

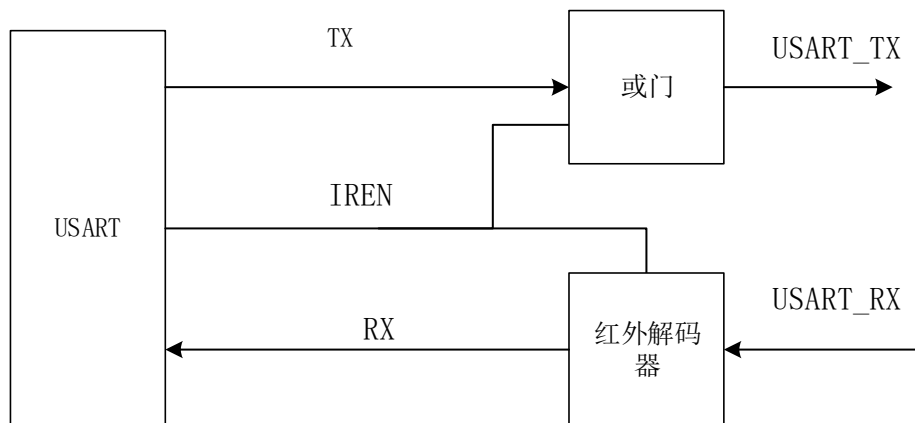


图 32-18 IrDA SIR ENDEC 框图

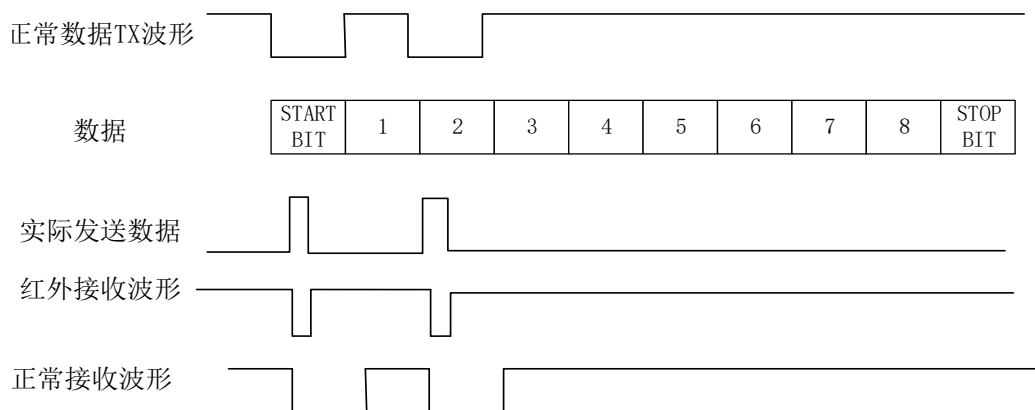


图 32-19 IrDA 数据调制 (3/16) -普通模式

### 32.3.13. 用 DMA 连续通信

USART 可以利用 DMA 连续通信。Rx 缓冲器和 Tx 缓冲器的 DMA 请求是分别产生的。

#### 32.3.13.1. 利用 DMA 发送

使用 DMA 进行发送，可以通过设置 USART\_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为 '1' 时，DMA 就从指定的 SRAM 区传送数据到 USART\_DR 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下：

- 1) 在 DMA 控制寄存器上将 USART\_DR 寄存器地址配置成 DMA 传输的目的地址。在每个 TXE 事件后，数据将被传送到这个地址。
- 2) 在 DMA 控制寄存器上将存储器地址配置成 DMA 传输的源地址。在每个 TXE 事件后，将从此存储器区读出数据并传送到 USART\_DR 寄存器。
- 3) 在 DMA 控制寄存器中配置要传输的总的字节数。
- 4) 在 DMA 寄存器上配置通道优先级。
- 5) 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- 6) 通过写0，清 TC 位。
- 7) 在 DMA 寄存器上激活该通道。

当传输完成 DMA 控制器指定的数据量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。

在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 DMA\_ISR 寄存器的 TCIF 标志；监视 USARTx\_SR 寄存器的 TC 标志可以确认 USART 通信是否结束，这样可以在关闭 USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等待 TC=1。

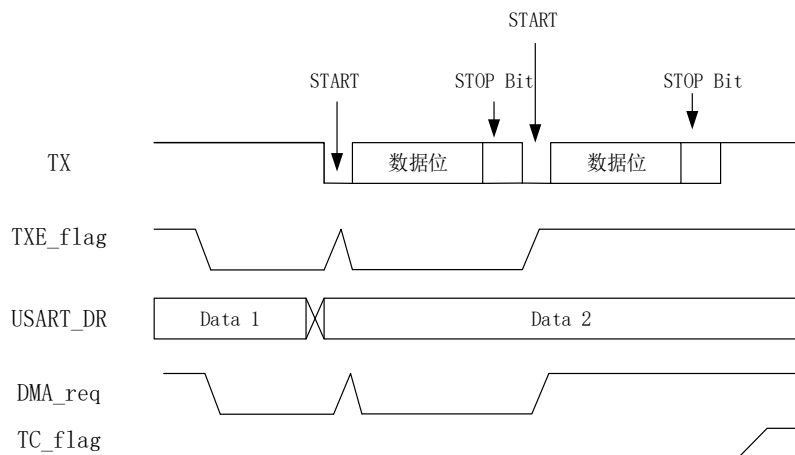


图 32-20 利用 DMA 发送

### 32.3.13.2. 利用 DMA 接收

可以通过设置 USART\_CR3寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就把数据从 USART\_DR 寄存器传送到指定的 SRAM 区（参考 DMA 相关说明）。为 USART 的接收分配一个 DMA 通道的步骤如下：

- 1) 通过 DMA 控制寄存器把 USART\_DR 寄存器地址配置成传输的源地址。在每个 RXNE 事件后，将从此地址读出数据并传输到存储器。
- 2) 通过 DMA 控制寄存器把存储器地址配置成传输的目的地址。在每个 RXNE 事件后，数据将从 USART\_DR 传输到此存储器区。
- 3) 在 DMA 控制寄存器中配置要传输的总的字节数。
- 4) 在 DMA 寄存器上配置通道优先级。
- 5) 根据应用程序的要求配置在传输完成一半还是全部完成时产生 DMA 中断。
- 6) 在 DMA 控制寄存器上激活该通道。

当接收完成 DMA 控制器指定的传输量时，DMA 控制器在该 DMA 通道的中断矢量上产生一中断。

### 32.3.13.3. 多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和 RXNE 一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了，会在当前字节传输结束后，产生中断。

### 32.3.14. 硬件流控制

利用 nCTS 输入和 nRTS 输出可以控制2个设备间的串行数据流。下图表明在这个模式里如何连接2个设备。

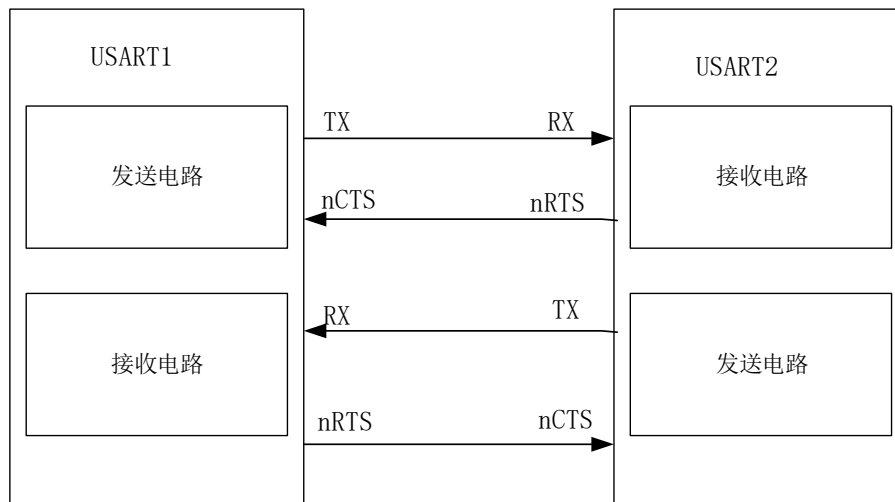


图 32-21 两个 USART 间的硬件流控制

#### 32.3.14.1. RTS 流控制

如果 RTS 流控制被使能 (RTSE=1)，只要 USART 接收器准备好接收新的数据，nRTS 就变成有效（接低电平）。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。

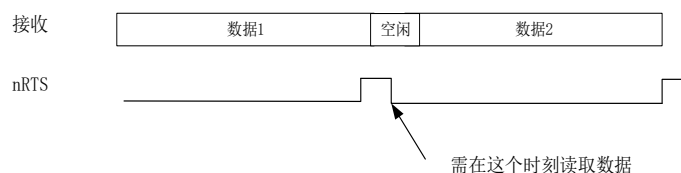


图 32-22 RTS 流控制

#### 32.3.14.2. CTS 流控制

如果 CTS 流控制被使能 (CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效（被拉成低电平），则下一个数据被发送（假设那个数据是准备发送的，也就是 TXE=0），否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1时，只要 nCTS 输入一变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART\_CT3寄存器的 CTSIE 位，则产生中断。



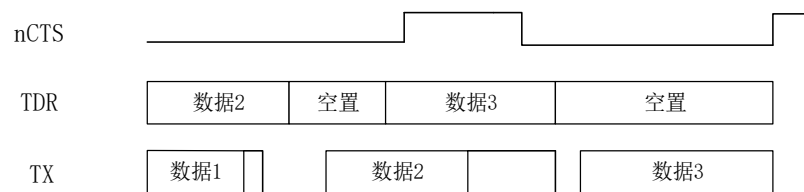


图 32-23 CTS 流控制

### 32.4. USART 中断请求

序号	中断事件	事件标志	使能位	发送/接收
1	发送数据寄存器空	TXE	TXEIE	发送
2	CTS (Clear to Send) 中断	CTSIF	CTSIE	发送
3	发送完成	TC	TCIE	发送
4	接收寄存器非空 (读数据准备好)	RXNE	RXNEIE	接收
5	Overrun 错误	ORE		接收
6	空闲帧	IDLE	IDLEIE	接收
7	奇偶校验错误	PE	PEIE	接收
8	断开标志	LBD	LBDIE	接收
9	多处理器通讯时, 噪声、overrun 和帧错误	NE/ORE/FE	EIE	接收

注：所有 USART 中断共用同一个中断向量。

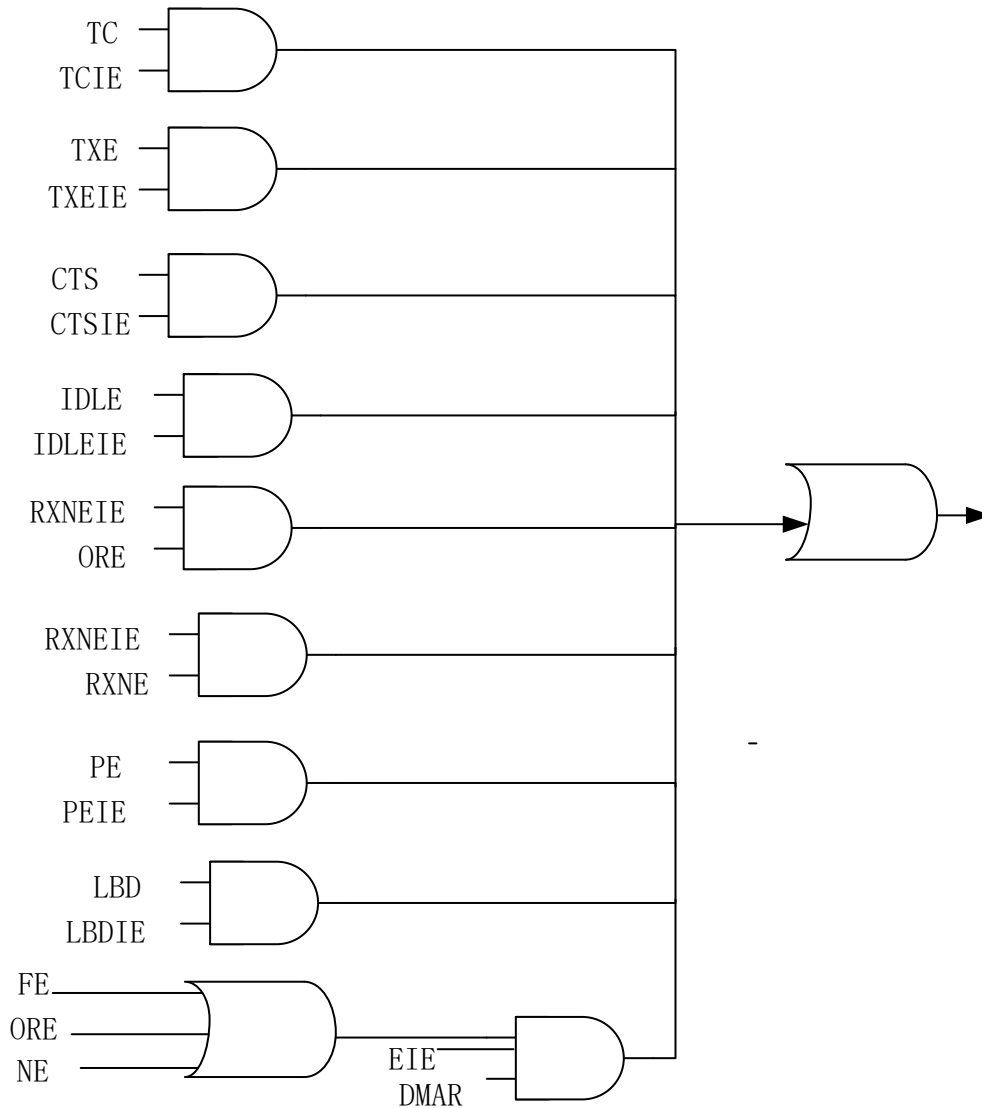


图 32-24 USART 中断映像图

## 32.5. USART 寄存器

### 32.5.1. 状态寄存器 (USART\_SR)

Address offset: 0x00

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		ABRRQ	ABRE	ABRF	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE	
-		W	R		RC_W0		R	RC_W0		R					

Bit	Name	R/W	Reset Value	Function
31: 13	保留	-	-	保留

Bit	Name	R/W	Reset Value	Function
12	ABRRQ	W	0	自动波特率请求。 该位写1会复位 ABRF 标志位，并且请求下一帧的自动波特率检测。
11	ABRE	R	0	自动波特率错误标志。 当自动波特率检测出错（波特率超出范围或者字符比较错误）时，硬件置位该寄存器。 软件通过写1到 ABRRQ 寄存器清零该位。
10	ABRF	R	0	自动波特率检测标志。 当自动波特率设置（同时设置 RXNE=1，当中断使能后产生中断），或者自动波特率检测操作出错（ABRE=1，RXNE=1，FE=1）时该位由硬件置1。 软件通过写1到 USART_RQR 寄存器的 ABRRQ 位清零该位。
9	CTS	RC_W0	0	CTS 标志。 当 CTS 输入 toggle，别 CTSE=1时，该寄存器为1.软件写0清零。当 CTSIE=1时，产生 CTS 中断。 0: CTS line 值未改变 1: CTS line 值改变
8	LBD	RC_W0	0	LBD: LIN 断开检测标志 (LIN break detection flag) 当探测到 LIN 断开时，该位由硬件置 '1'，由软件清 '0'（向该位写0）。如果 USART_CR3 中的 LBDIE = 1，则产生中断。 0: 没有检测到 LIN 断开； 1: 检测到 LIN 断开。 注意：若 LBDIE=1，当 LBD 为 '1' 时要产生中断
7	TXE	R	1	传输寄存器空标志。 当 USART_DR 寄存器数据传送到移位寄存器，硬件置位该寄存器。当 TXEIE=1时，产生中断。 写 USART_DR 寄存器会清零该位。 0: 数据未传送到移位寄存器 1: 数据传送到移位寄存器
6	TC	RC_W0	1	传送完成标志。 传送数据帧完成后，且 TXE=1，则硬件置位该寄存器。TCIE=1时产生中断。

Bit	Name	R/W	Reset Value	Function
				<p>软件先读 USART_SR 寄存器然后写 USART_DR 寄存器会清零该位（针对多处理器通讯）。软件同时可以写0清零。</p> <p>0: 传送未完成 1: 传送完成</p>
5	RXNE	RC_W0	0	<p>读数据寄存器不空标志。</p> <p>当移位寄存器值传送到 USART_DR 寄存器，硬件置位该寄存器。</p> <p>软件读 USART_DR 寄存器清零该位。</p> <p>当 RXNEIE=1时，产生中断。</p> <p>0: 未收到数据 1: 接收数据准备好</p>
4	IDLE	R	0	<p>空闲标志。</p> <p>检测 IDLE line，硬件置位该寄存器。当 IDLEIE=1时产生中断。</p> <p>软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。</p> <p>0: 未检测到 IDLE line 1: 检测到 IDLE line</p>
3	ORE	R	0	<p>Overrun 错误标志。</p> <p>当 RXNE=1时，在移位寄存器中接收到的数据正准备转移到 RDR 寄存器时，硬件设置该位。</p> <p>软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。</p> <p>当 RXNEIE=1时，产生中断。</p> <p>0: 未产生 Overrun 错误 1: 产生 Overrun 错误</p> <p>注：该寄存器置位时，RDR 寄存器内容不会丢失，但移位寄存器内容被覆盖。</p> <p>当 EIE=1时，产生 ORE 中断。</p>
2	NE	R	0	<p>噪声错误标志。</p> <p>在数据帧接收到噪声时，硬件置位该寄存器。</p> <p>软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。</p> <p>0: 未检测到噪声错误 1: 检测到噪声错误</p> <p>注：当 RXNE 与 NE 同时产生时，NE=1时不产生中断，而在设置 RXNE 标志时产生中断。在多</p>

Bit	Name	R/W	Reset Value	Function
				缓冲器通讯模式下, 当 EIE=1 时 NE=1 会产生中断。
1	FE	R	0	<p>帧错误标志。</p> <p>当检测到不同步、过多的噪声或中止字符时, 由硬件设置此位。</p> <p>软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。</p> <p>0: 未检测到帧错误 1: 检测到帧错误或 break 字符</p> <p>注: 当 RXNE 与 FE 同时产生时, FE=1 时不产生中断, 而在设置 RXNE 标志时产生中断。如果当前传输的数据既产生了帧错误, 又产生了过载错误, 硬件还是会继续该数据的传输, 并且只设置 ORE 标志位。在多缓冲器通讯模式下, 当 EIE=1 时 FE=1 会产生中断。</p>
0	PE	R	0	<p>校验值错误。</p> <p>当接收时校验值错误时, 硬件置位该寄存器。</p> <p>软件先读 USART_SR 寄存器后读 USART_DR 寄存器可以清零该位。但软件在清该位前必须等待 RXNE=1。</p> <p>当 PEIE 时, 产生中断。</p> <p>0: 未产生奇偶校验错误 1: 产生奇偶校验错误</p>

### 32.5.2. 数据寄存器 (USART\_DR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 9	保留	-	-	保留
8: 0	DR[8: 0]	RW	0x0	接收/发送数据寄存器。

				<p>取决于读还是写操作，前者是接收到的数据，后者是发送的数据。</p> <p>DR 寄存器物理上由两个寄存器组成（一个是发送的 TDR，一个是接收的 RDR），所以 DR 寄存器实现了读和写的两个功能。</p> <p>TDR 寄存器在内部总线和输出移位寄存器之间提供了并行的接口，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。</p> <p>当奇偶校验使能打开进行发送操作时，写 MSB 位（bit7或者 bit8）是无效的，因为已被校验位代替了。</p> <p>当奇偶校验使能打开进行接收操作时，读出的 MSB 位是被接收到的校验位。</p>
--	--	--	--	--

### 32.5.3. 波特率寄存器 (USART\_BRR)

Address offset: 0x08

Reset value: 0x0000 0000

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

在自动波特率检测模式，硬件更新该寄存器。

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	-	保留
15: 4	DIV_Mantissa[11: 0]	RW	0	12bit 整数
3: 0	DIV_Fraction[3: 0]	RW	0	4bit 小数

### 32.5.4. 控制寄存器 1 (USART\_CR1)

Address offset: 0x0C

Reset value: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
-	-	RW													

Bit	Name	R/W	Reset Value	Function
31: 14	保留	-	-	保留
13	UE	RW	0	USART 使能。当该位清零后，USART 模块会立即停止当前操作。该位由软件置位和清零。 0: USART prescaler 和 output 禁止, low-power 模式 1: USART 使能 软件需要等待 USART_SR.TC 置位后, 才能清零 UE 位, 进入低功耗模式; 同时, 在清零 UE 位之前 DMA 通道需要禁止。
12	M	RW	0	0: 1 bit 起始位, 8 bits 数据为, n bits 停止位 1: 1 bit 起始位, 9 bits 数据位, n bits 停止位
11	WAKE	RW	0	接收唤醒方式。 从 mute 模式唤醒方式。由软件置位或者清零。 0: Idle line 唤醒 1: 地址唤醒
10	PCE	RW	0	奇偶校验控制。 0: 奇偶校验禁止 1: 奇偶校验使能 奇偶校验位: 9bit 的第9位; 8bit 的第8位。
9	PS	RW	0	奇偶校验选择。由软件置位和清零。 0: 偶校验 1: 奇校验
8	PEIE	RW	0	PE 中断使能。由软件置位和清零。 0: 禁止 1: PE 中断使能
7	TXEIE	RW	0	TXE 中断使能。由软件置位和清零。 0: 禁止 1: TXE 中断使能
6	TCIE	RW	0	传送结束中断使能。由软件置位和清零。 0: 禁止 1: TC 中断使能
5	RXNEIE	RW	0	RXNE 中断使能; 由软件置位和清零。 0: 禁止 1: ORE 或者 RXNE 中断使能
4	IDLEIE	RW	0	IDLE 中断使能。由软件置位和清零。 0: 禁止 1: IDLE 中断使能
3	TE	RW	0	传送使能。

Bit	Name	R/W	Reset Value	Function
				0: 传送禁止 1: 传送使能
2	RE	RW	0	接收使能。 0: 接收禁止 1: 接收使能, 开始检测 start 位
1	RWU	RW	0	接收唤醒。 该位表明 USART 是否为 mute 模式。 当接收到 mute 模式序列, 该寄存器置位; 如果接收到唤醒序列, 该寄存器清零。具体哪种唤醒序列 (地址或者 IDLE) 由寄存器 USART_CR1.WAKEbit 控制。 0: 接收器为工作模式 1: 接收器为静默模式 注1: 在设置该位进入 mute 模式前, USART 要已经先接收了一个数据字节, 否则在 mute 模式下, 不能被 idle 总线检测唤醒。 注2: 当配置成地址标记检测唤醒 (WAKE=1), 在 RXNE 被置位时, 不能用软件修改 RWU 位。
0	SBK	RW	0	发送 break 帧。 软件置位该寄存器, 发送 break 字节。Break 帧的 stop 位发送后, 硬件清零该寄存器。 0: 不发送 break 字节 1: 发送 break 字节

### 32.5.5. 控制寄存器 2 (USART\_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res	LBDIE	LBDL	Res	ADD[3:0]			
-	RW							-	RW	RW	-	RW			

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14	LINEN	RW	0	LIN 模式使能。 软件置位和清零。



Bit	Name	R/W	Reset Value	Function
				0: LIN 模式; 1: 使能 LIN 模式; LIN 模式下, 通过使能 SBK 位, 发送 LIN 同步 breaks (13低位), 以及检测 Lin 同步断开符。
13: 12	STOP[1: 0]	RW	0	Stop 位配置。 00: 1 stop bit; 01: 0.5stop; 10: 2 stop 位; 11: 1.5stop
11	CLKEN	RW	0	CK pin 使能。 0: 禁止; 1: CK pin 使能; 不支持同步模式时, 该位保留。
10	CPOL	RW	0	时钟极性。 同步模式, CK pin 输出时钟极性。 0: 传输窗外, CK pin 为稳定低值; 1: 传输窗外, CK pin 为稳定高值;
9	CPHA	RW	0	该位在同步模式下用于选择 CK pin 输出时钟的相位。它与 CPOL 位一起工作, 以产生所需的时钟/数据关系。 0: 第一个时钟传输是首个数据捕获沿; 1: 第二个时钟传输是首个数据捕获沿;
8	LBCL	RW	0	最后一位数据的时钟脉冲是否在 CK pin 输出。 0: 最后一位数据的时钟脉冲不在 CK pin 输出; 1: 最后一位数据的时钟脉冲在 CK pin 输出;
7	保留	-	-	保留
6	LBDIE	RW	0	LIN break 中断使能。 0: 禁止; 1: 中断产生; 通过这去控制 USART_SR 寄存器中的 LBD,使得 LBD 为1, 产生中断
5	LBDL	RW	0	LIN break 检测长度。 0: 10bits 检测 break; 1: 11bits 检测 break;
4	保留	-	-	保留
3: 0	ADD[3: 0]	RW	4'b0	USART 地址。 该寄存器用于多处理器 mute 模式, 用作4bit 地址唤醒时的地址。

### 32.5.6. 控制寄存器 3 (USART\_CR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	ABR-		ABR	OVE	CTSI	CTS	RTS	DMA	DMA	SC	NAC	HDSE	IRL	IRE	EI
s	MOD[1: 0]		EN	R8	E	E	E	T	R	EN	K	L	P	N	E
-	RW														

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14: 13	ABRMOD[1: 0]	RW	2'b0	自动波特率检测模式。 00: 从 start 位开始测量波特率 01: 下降沿到下降沿测量 10: 保留 11: 保留 当 ABREN=0或者 UE=0时, 该寄存器只写。
12	ABREN	RW	0	自动波特率使能。 0: 禁止 1: 自动波特率使能
11	OVER8	RW	0	过采样模式。 0: 16倍过采样 1: 8倍过采样 该位仅在 UE=0时可被写。
10	CTSIE	RW	0	CTS 中断使能。 0: 禁止; 1: CTSIF 中断使能;
9	CTSE	RW	0	CTS 使能。 0: CTS 硬件流控制禁止; 1: CTS 模式使能。当有当 CTS 输入为0时, 才会传输数据。此时, 当数据写入数据寄存器后, 要等待 CTS 有效后才会启动传输。
8	RTSE	RW	0	RTS 使能。 0: RTS 硬件流控制禁止; 1: RTS 输出使能, 只有当接收 buffer 未滿时才会请求下一个数据。当前数据发送完成后, 发送

Bit	Name	R/W	Reset Value	Function
				操作暂停。如果可以接收数据了，将 RTS 置为有效 (0)。
7	DMAT	RW	0	发送时使能 DMA。 0: 禁止; 1: 传送时使能 DMA;
6	DMAR	RW	0	接收时使能 DMA。 0: 禁止; 1: 接收时使能 DMA;
5	SCEN	RW	0	智能卡模式使能。 0: 禁止; 1: 使能;
4	NACK	RW	0	智能卡 NACK 使能。 0: 奇偶校验错误时发送 NACK 禁止; 1: 奇偶校验错误时发送 NACK 使能;
3	HDSEL	RW	0	半双工选择。 0: 非半双工模式; 1: 半双工模式选择;
2	IRLP	RW	0	IrDA 低功耗。 0: normal 模式; 1: IrDA 低功耗模式;
1	IREN	RW	0	IrDA 模式使能。 软件使能和清零该寄存器。 0: IrDA 禁止; 1: IrDA 使能;
0	EIE	RW	0	错误中断使能。 0: 禁止; 1: 帧错误 FE、overrun 错误 ORE、噪声 NE 中断使能。

### 32.5.7. 保护时间和预分频器 (USART\_GTPR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	0	保留
15: 8	GT[7: 0]	RW	0	保护时间值 (Guard time value) 。 该域定义了以波特时钟为单位的保护时间。在智能卡模式, 需要此功能。当保护时间过去后, 才会设置发送完成标志。
7: 0	PSC[7: 0]	RW	0	<p>预分频器值 (Prescaler value) 。</p> <ul style="list-style-type: none"> <li>■ 在红外 (IrDA) 低功耗模式下: PSC[7: 0]=红外低功耗波特率。 对系统时钟分频以获得低功耗模式下的频率: 源时钟被寄存器中的值 (仅有8位有效) 分频 00000000: 保留 – 不要写入该值; 00000001: 对源时钟1分频; 00000010: 对源时钟2分频; .....</li> <li>■ 在红外 (IrDA) 的正常模式下: PSC 只能设置为00000001。</li> <li>■ 在智能卡模式下: PSC[4: 0]: 预分频值 对系统时钟进行分频, 给智能卡提供时钟。 寄存器中给出的值 (低5位有效) 乘以2后, 作为对源时钟的分频因子。 00000: 保留 – 不要写入该值; 00001: 对源时钟进行2分频; 00010: 对源时钟进行4分频; 00011: 对源时钟进行6分频; .....</li> </ul> <p>注意: 位[7: 5]在智能卡模式下没有意义。</p>

## 33. CAN2.0 控制器

### 33.1. 介绍

CAN (Controller Area Network) 总线是一种可以在无主机情况下实现微处理器或者设备之间相互通信的总线标准。

CAN 控制器遵循 CAN 总线 CAN2.0B 协议。

CAN 总线控制器可以处理总线上的数据收发，在本产品中，CAN 控制器具有 12 组筛选器。筛选器用于为应用程序选择要接收的消息。

CAN 控制器中应用程序可通过 1 个高优先级的主发送缓冲器 (Primary Transmit Buffer, 以下简称 PTB) 和 3 个辅发送缓冲器 (Secondary Transmit Buffer, 以下简称 STB) 将发送数据送至总线，由发送调度器决定邮箱发送顺序。通过 3 个接收缓冲器 (Receive Buffer, 以下简称 RB) 获取总线数据。3 个 STB 以及 3 个 RB 可以理解为一个 3 级 FIFO，FIFO 完全由硬件控制。

CAN 总线控制器同时也可以支持时间触发 CAN 通信 (Time-trigger communication)。

### 33.2. CAN 主要特性

- 完全支持 CAN2.0B 协议。
- CAN2.0支持最高通信波特率1 Mbit/s
- 支持1 ~ 1/32的波特率预分频，灵活配置波特率。
- 3 个接收缓冲器
  - FIFO 方式
  - 错误或者不被接收的数据不会覆盖存储的消息
- 1 个高优先主发送缓冲器 PTB
- 3 个副发送缓冲器 STB
  - FIFO 方式
  - 优先级仲裁方式
- 12组独立的筛选器
  - 支持 11 位标准 ID 和 29 位扩展 ID
  - 可编程 ID CODE 位以及 MASK 位
- PTB/STB 均支持单次发送模式
- 支持静默模式
- 支持回环模式
- 支持捕捉传输的错误种类以及定位仲裁失败位置
- 可编程的错误警告值
- 支持 ISO11898-4 规定时间触发 CAN 以及接收时间戳

### 33.3. CAN 功能描述

#### 33.3.1. 模块框图

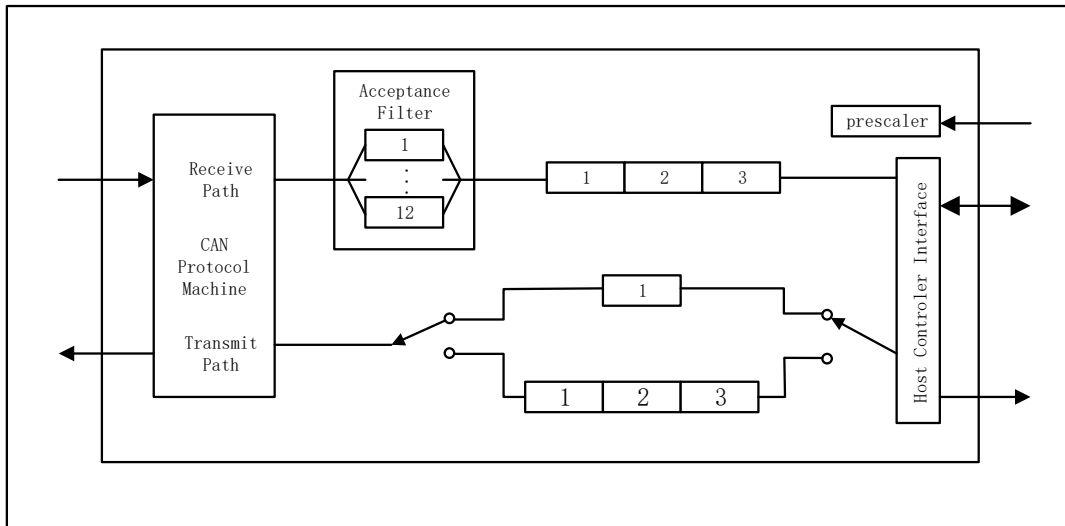


图 33-1 CAN 模块框图

#### 33.3.2. 动作模式

CAN 控制器存在两个操作模式，复位模式 (CAN\_MCR.RESET=1) 和动作模式 (CAN\_MCR.RESET=0)。模块初始化时，首先应该在复位模式中设定只能在复位模式下操作的寄存器（详见软件复位功能章节），然后退出复位模式，在动作模式中操作其余寄存器。

#### 33.3.3. 波特率设定

CAN 通信使用时钟 CAN\_clk 的时钟源为外部高速振荡器 HSE 或 PLL，使用 CAN 模块之前，需要在 RCC 章节设定 CAN 通信时钟。

下图给出 CAN 位时间定义图，虚线上部分为 CAN 协议规定的位时间，虚线下部分为本 CAN 控制器 CAN-CTRL 定义的位时间。其中 segment1 和 segment2 可以通过寄存器 CAN\_ACBTR 设定。CAN\_ACBTR 寄存器只能在 CAN\_MCR.RESET=1 即 CAN 软件复位时设定。

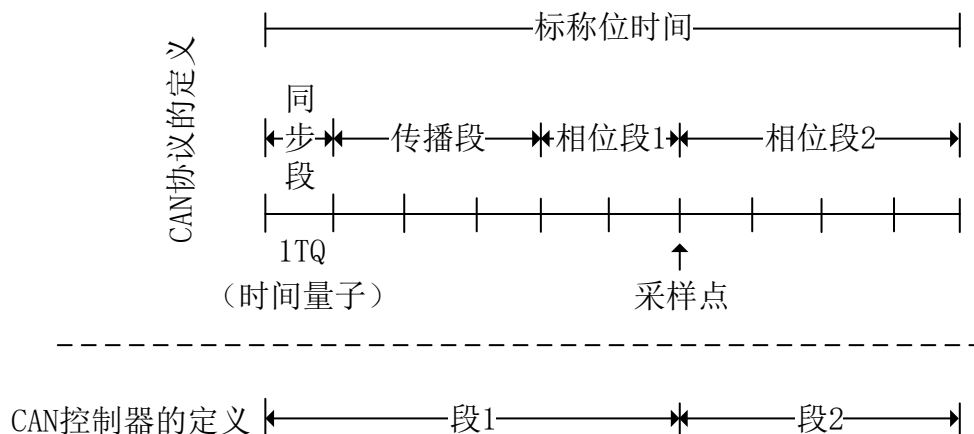


图 33-2 CAN 位时间定义

TQ 计算方法请参考以下公式，其中 PRESC 通过 CAN\_RLSSP 寄存器的 PRESC 位设定。  
 $f_{can\_clk}$ 为 CAN 通信时钟频率。

$$TQ = \frac{S\_PRESC+1}{f_{can\_clk}}$$

位时间计算方法请参考以下公式，其中 segment1 和 segment2 通过 CAN\_ACBTR 寄存器的 AC\_SEG\_1 位和 AC\_SEG\_2 位设定。

$$BT = t_{s\_segment1} + t_{s\_segment2} = ( (AC\_SEG\_1 + 2) + (AC\_SEG\_2 + 1) ) \times TQ$$

表 33-1 CAN 时间设定规则

位	设定范围			规则
CAN_ACBTR寄存器的AC_SEG_1位	[0..511]	CAN2.0 bits	(slow)	SEG_1 ≥ SEG_2 + 1
CAN_ACBTR寄存器的AC_SEG_2位	[0..127]	CAN2.0 bits	(slow)	-
CAN_ACBTR寄存器的AC_SJW位	[0..127]	CAN2.0 bits	(slow)	-

表 33-2 20 MHz 通信时钟时波特率设定建议

Bit Rate [Mbit/s]	PSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN 通信时钟]
0.25 (仲裁)	80	1	80	64	16	16	-
0.5 (仲裁)	80	1	40	32	8	8	-
0.5	80	1	40	32	8	8	-
1	80	1	20	16	4	4	16

表 33-3 40 MHz 通信时钟时波特率设定建议

Bit Rate[Mbit/s]	PSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN 通信时钟]
0.25 (仲裁)	80	2	80	64	16	16	-
0.5 (仲裁)	80	1	80	64	16	16	-
0.5	80	2	40	32	8	8	-
1	80	1	40	32	8	8	32

#### 33.3.4. 发送缓冲器

CAN 控制器提供两种发送缓冲器用于发送数据，主发送数据缓冲器 PTB 和副发送缓冲器 STB。PTB 具有最高的优先级，但只能缓冲一帧数据。STB 优先级比 PTB 低，但可以缓冲 3 帧数据，且 STB 内 3 帧数据可以工作在 FIFO 模式或者优先级仲裁模式。STB 中的 3 帧数据可以通过 TCMD 寄存器的 TSALL 位设定为 1 全部发送，在 FIFO 模式下，最先写入的数据先发送，在优先级模式下，ID 小的数据先发送。

PTB 中的数据具有最高优先级，所以 PTB 发送能推迟 STB 发送，但是已经赢得仲裁并开始发送的 STB 不能够被 PTB 发送推迟。

PTB 和 STB 可以通过 TBUF 寄存器进行访问。通过 TCMD 寄存器的 TBSEL 位选择 PTB 或者 STB，TBSEL=0，选择 PTB，TBSEL=1，选择 STB。通过 TCTRL 寄存器的 TSNEXT 位选择 STB 中的下一个 SLOT。对应关系如下图所示：

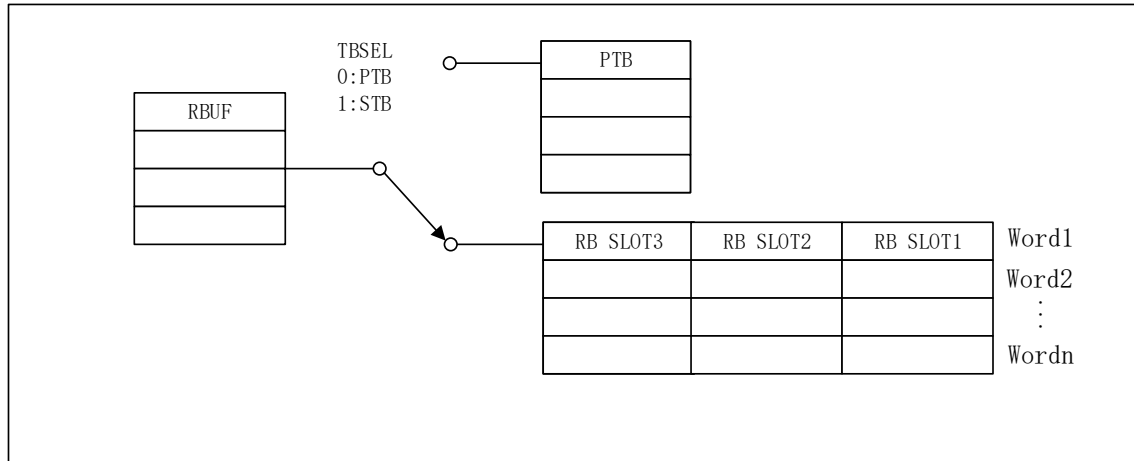


图 33-3 CAN TBUF 寄存器写发送缓冲器和示意图

### 33.3.5. 接收缓冲器

CAN 控制器 提供 3 个 SLOT 的接收缓冲器用于存储接收到的数据，这 3 个 SLOT 的接收缓冲器工作在 FIFO 模式。RB SLOT 通过 RBUF 寄存器来读取接收到的数据，总是最先读取最早接收到的数据，并通过 RCTRL 寄存器的 RREL 设置为 1 释放已经读取的 RB SLOT，并指向下一个 RB SLOT。

通过 RBUF 读取 RB SLOT 示意图如下。

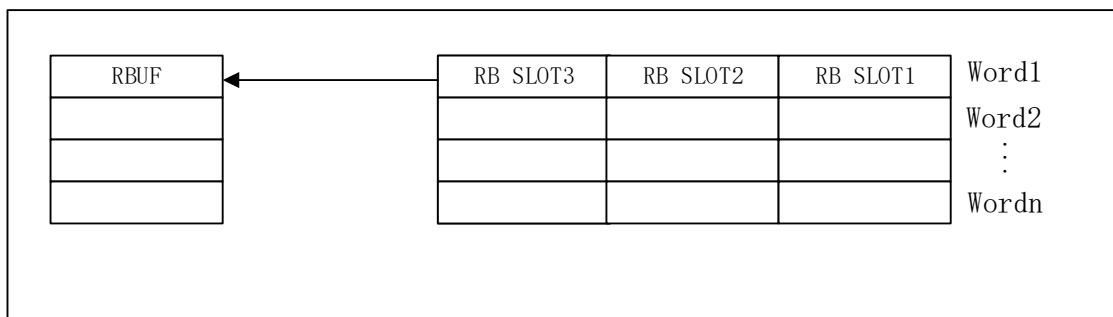


图 33-4 CAN RBUF 寄存器读接收缓冲器示意图



### 33.3.6. 接收筛选寄存器组

CAN 控制器 提供 12 组 32 位筛选器用于过滤接收到的数据从而降低 CPU 负荷，筛选器可以支持标准格式 11 位 ID 或者扩展格式 29 位 ID。每组筛选器由 3 个 CODE 寄存器（参考 筛选器组 code 寄存器）和 3 个 MASK 寄存器（参考 筛选器组 mask 寄存器）组成，CODE 寄存器用于比较接收到的 CAN 帧（帧格式请参考 LLC 帧格式定义），而 MASK 寄存器作为 CODE 寄存器的掩码，对应的 MASK 位为 1 时，则忽略 CODE 寄存器中的对应 bit 位。

接收到的数据只要通过 12 组筛选器的任意一组，则被接收，接收到的数据存储在 RB 中，否则数据不被接收，也不被存储。

每组筛选器通过 CAN\_ACFCR 寄存器的 AE\_n 位使能或者禁止。ID CODE 和 ID MASK 通过 CAN\_ACFC 寄存器和 CAN\_ACFM 配置。筛选器通过 CAN\_ACFCR 寄存器的 ACFADR 位选择。ID CODE 和 ID MASK 通过 ACF 寄存器访问且只能在 CAN\_MCR.RESET=1 即 CAN 软件复位时设定。ACF 寄存访问筛选寄存器组的方式请参考下图。

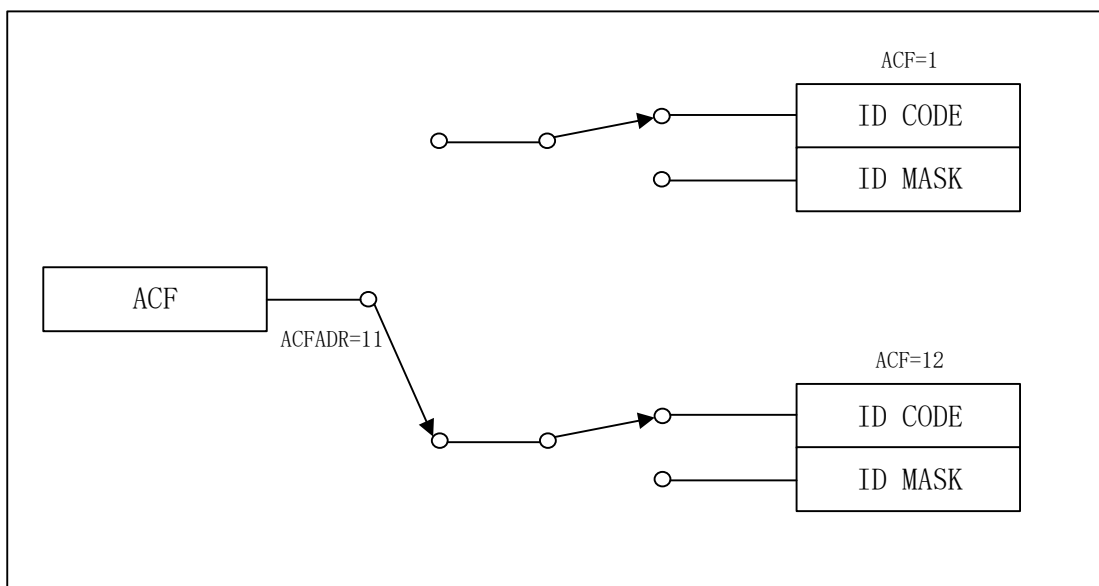


图 33-5 CAN ACF 寄存器访问筛选器组示意图

### 33.3.7. LLC 帧格式定义

下表显示了包含时间戳的逻辑链路控制帧（LLC）的定义。这个统一的定义用于发送帧（存储在 TBUF 中）、接收帧（从 RBUF 读取）和配置接收过滤器（ACFC 和 ACFM）。

偏移地址	寄存器																
0x00	CAN_ID	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res	Res	Res	ID[28: 16]												
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		ID[15: 0]															
0x04	CAN_FORMAT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

偏移地址	寄存器																	
		Res	Res	Res	LBF	Res	KOER[2: 0]				Res	Res	Res	RMF.	Res	Res	FDF.	IDE
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		Res	Res	Res	Res	Res								DLC[3: 0]				
0x08	CAN_TYPE	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
		HANDLE[7: 0]								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																
0x18	CAN_TTCAN	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	Res
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		CYCLE_TIME[15: 0]																
0x1c	CAN_DATA1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
		Data[31: 16]																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		Data[15: 0]																
0x20	CAN_DATA2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
		Data[31: 16]																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		Data[15: 0]																

位	描述
ID	帧标识符
DLC	<p>数据长度代码</p> <p>DLC 定义了帧内的有效载荷字节数。</p> <p>经典 CAN 2.0B 帧的 DLC 长度为 4 位 (DLC (3: 0))。</p> <p>对于经典 CAN 2.0B, 可以选择传输 DLC 没有意义的远程帧。位 RMF 的解释提供了更多细节。</p>
IDE	<p>ID 格式</p> <p>0 – 标准格式: ID (28: 18)</p> <p>1 – 扩展格式: ID (28: 0)</p>
FDF	该位应当设置为0从而支持 CAN 2.0 frame
RMF	远程帧

位	描述
	<p>0 – 数据帧</p> <p>1 – 远程帧</p> <p>远程帧携带零字节有效载荷。DLC 的值按原样传输，但对帧大小没有影响。因此，远程帧的 DLC 值可能携带一些编码信息。</p> <p>只有经典的 CAN 2.0B 帧可以是远程帧。</p>
KOER	<p>错误类型</p> <p>接收帧的 KOER 与寄存器 EALCAP 中的 KOER 位具有相同的含义。如果 RBALL=1，则接收帧的 KOER 变得有意义。</p> <p>请注意，如果 RBALL=1，一般情况下会禁用过滤器组。</p>
LBF	<p>回环帧</p> <p>如果激活了回环模式并且 CAN 控制器已接收到自己的传输帧，则接收帧的 LBF 设置为 1。如果 LBME=1 而网络中的其他节点也进行传输，这将很有用。</p>
HANDLE	<p>帧识别 HANDLE</p> <p>句柄的目的是使用 TSTAT 识别帧。MAC 帧中不使用 HANDLE。建议主机应用程序将软件计数器的值写入 HANDLE。这样的软件计数器可以随着要传输的每个新帧而增加，并且应该翻转。</p> <p>注意：为避免在模拟过程中出现未初始化的内存问题，主机应用程序应始终为要传输的每一帧定义 HANDLE。</p>
CYCLE_TIME	<p>周期时间 (TTCAN 的时间戳)</p> <p>CYCLE_TIME 将仅存储接收到的帧。这是帧 SOF 处的循环时间。参考消息的循环时间始终为 0。</p>
Data	<p>帧的有效载荷数据。</p> <p>经典 CAN 2.0B 最多 8 个字节，RBUF 中未使用的有效载荷数据字确实包含需要忽略的垃圾数据。</p>

### 33.3.8. 数据发送

在开始发送前必须保证 PTB 或者 STB 中至少有一帧数据已被装载，PTB 发送过程中 TPE 被锁定，STB 的填充情况可以通过 TSSTAT 位确认。发送数据设定步骤如下：

1. 设定 TBSEL 从 PTB 和 STB 中选择发送 BUF
2. 通过 TBUF 寄存器写需要发送的数据。
3. 如果选择的是 STB，设置 TSNEXT=1 以完成 STB SLOT 的装载。
4. 发送使能
  - PTB 发送使用 TPE
  - STB 发送使用 TSALL 或者 TSONE
5. 发送完成状态确认
  - PTB 发送完成使用 TPIF，TPIE 用于使能 TPIF

- STB 采用 TSONE 发送完成时使用 TSIF, TSIE 用于使能 TSIF
- STB 采用 TSALL 发送完成时使用 TSIF, 此时需要设定的全部 STB SLOT 数据发送完成后, TSIF 才置位, TSIE 用于使能 TSIF

### 33.3.9. 取消数据发送

可以通过 TPA 或者 TSA 取消已请求但还没有被执行的数据发送。取消数据发送会出现以下几种情况:

#### 1. 仲裁中

节点仲裁失败, 则取消数据发送。

- 节点仲裁成功, 则继续发送。

#### 2. 数据发送中

- 成功发送数据且收到 ACK, 对应的标志和状态正常置位。数据发送不取消。
- 成功发送数据但没有收到 ACK, 数据发送取消, 错误计数器增加。
- TSALL=1 设定的发送数据, 正在发送的 STB SLOT 数据正常发送, 没有开始发送的 STB SLOT 被取消。

取消数据发送的结果有以下两种情况。

1. TPA 释放 PTB, 且使 TPE=0。
2. TSA 释放一个 STB SLOT 或者全部 STB SLOT 取决是 TSONE 还是 TSALL 使能的发送。

### 33.3.10. 数据接收

接收筛选器组可以过滤掉不需要的接收数据, 减少中断的发生和 RB 的读取, 从而降低 CPU 负荷。接收数据设定步骤如下:

1. 设定筛选器组。
2. 设定 RFIE, RAFIE 和 AFWL。
3. 等待 RFIF 或者 RAFIF。
4. 通过 RBUF 从 RB FIFO 中读取最早接收到的数据。
5. 设置 RREL=1, 选择下一个 RB SLOT。
6. 重复 4, 5 直到通过 RSTAT 确认 RB 为空。

### 33.3.11. 错误处理

CAN 控制器一方面可以自动处理部分错误, 比如自动重发数据或者丢弃接收到含有错误的帧, 另一方面通过中断将错误向 CPU 报告。

CAN 节点有以下三种错误状态:

- 错误主动: 节点检测到错误时自动发送主动错误标志。
- 错误被动: 节点检测到错误时自动发送被动错误标志。
- 节点关闭: 关闭状态下此节点不再影响整个 CAN 网络。

CAN 控制器提供 TECNT 和 RECNT 两个计数器用于计数错误。TECNT 和 RECNT 计数器按照 CAN 协议规定的规则进行增减。另外提供可编程的 CAN 错误警告 LIMIT 寄存器用于产生错误中断通知 CPU。

CAN 通信过程中有以下 5 种错误类型, 错误类型可以通过 EALCAP 寄存器的 KOER 位识别。

- 位错误
- 形式错误
- 填充错误
- 应答错误
- CRC 错误

### 33.3.12. 节点关闭

当发送错误数大于 255 时，CAN 节点自动进入节点关闭状态而不参与 CAN 通信，直到返回到错误主动状态。可以通过 CAN\_MCR 寄存器的 BUSOFF 位确认 CAN 节点关闭状态。BUSOFF 被置位的同时 EIF 中断产生。

CAN 从节点关闭状态恢复到错误主动状态有以下两种方法：

- 上电复位
- 接收到连续 128 个 11 位的隐性位序列（恢复序列）

节点关闭状态下，TECNT 值保持不变，RECNT 用于计数恢复序列。从节点关闭状态恢复后，TECNT 和 RECNT 被复位为 0。

### 33.3.13. 仲裁失败位置捕捉

CAN 控制器能够精确捕捉到仲裁失败位的位置并反映到 WECR.ALC 寄存器中。WECR.ALC 寄存器中保存着最近一次仲裁失败位的位置，如果节点赢得仲裁，则 WECR.ALC 位不更新。ALC 值定义如下：

SOF 位后，第一个 ID 数据位 ALC 为 0，第二个 ID 数据位 ALC 为 1，依次类推。因为仲裁只发生在仲裁场内，所以 ALC 的最大值为 31。比如一个标准格式远程帧和一个扩展帧仲裁，扩展帧在 IDE 位失败，则 ALC=12。

### 33.3.14. 回环模式

CAN 控制器支持以下两种回环模式：

- 内部回环
- 外部回环

两种回环模式都可以接收自己发出的数据帧，主要用于测试用途。

内部回环模式，模块内部将接收数据线连接到发送数据线，并且发送数据不输出。内部回环模式下，节点会生成自应答信号以避免 ACK 错误。

外部回环模式保持和收发器的连接因此发送的数据仍能出现在 CAN 总线上，在收发器的帮助下，CAN 能收到自己发送的数据。外部回环模式可以通过 RCTRL 寄存器的 SACK 位来决定是否生成自应答信号，SACK=0 时，不生成自应答信号，SACK=1 时，生成自应答信号。

外部回环模式，SACK=0 时，会出现以下两种情况：

- 其它节点也收到本节点发送的数据帧并发送应答信号，该情况下本节点能够成功收发数据。
- 如果没有其它节点返回应答信号，则会产生应答错误，会重新发送数据并增加错误计数器。此时推荐采用单次发送模式。

从回环模式返回到正常模式时，除了清除模式位以外，还需要软件复位 CAN 控制器。

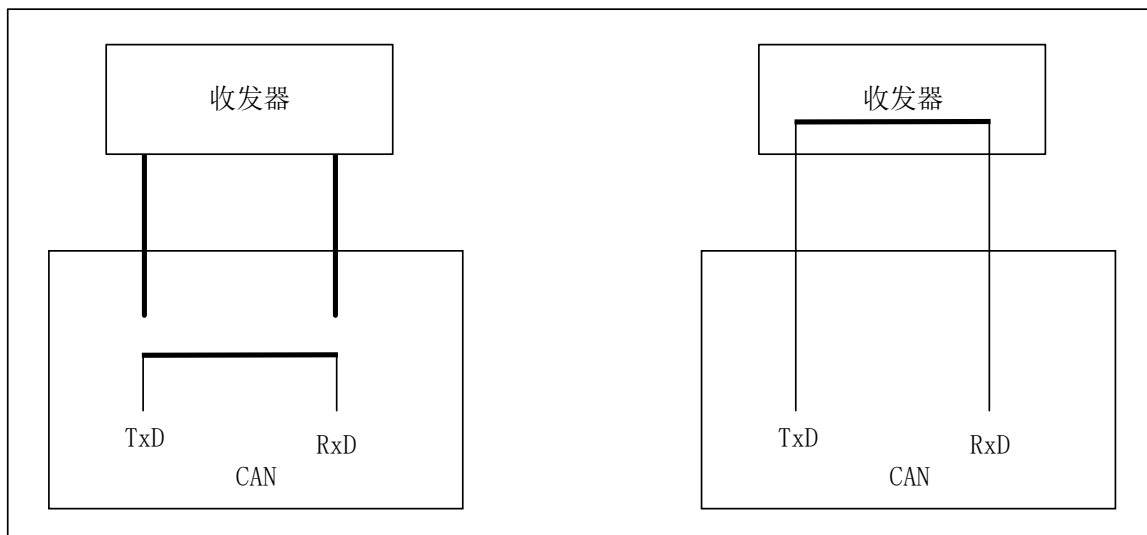


图 33-6 CAN 内部回环 LBMI 和外部回环 LBME 示意图

### 33.3.15. 静默模式

静默模式可以用来监控 CAN 网络数据。在静默模式下，可以从 CAN 总线接收数据，不向总线发送任何数据。将 CAN\_MCR 寄存器中的 LOM 置 1，使 CAN 总线控制器进入静默模式，将其清 0 可以离开静默模式。

外部回环模式和静默模式组合成外部回环静默模式，此时 CAN 可以认为一个安静的接收者，但在有必要的时候可以发送数据。外部回环静默模式下，帧包含自应答信号允许被发送，但是该节点不会产生错误标志和过载帧。

### 33.3.16. 软件复位功能

通过设定寄存器 CAN\_MCR 寄存器的 RESET 位为 1,实现软件复位功能，软件复位功能的复位范围如下表所示。

表 33-4 软件复位范围表

寄存器位名	软件复位	备注	寄存器位名	软件复位	备注
ACFADR	否	-	F_SEG_1	是	只能在软件复位时可写
ACODE	否	只能在软件复位时可写	F_SEG_2	是	只能在软件复位时可写
AE_x	否	-	F_SJW	是	只能在软件复位时可写
AFWL	否	-	KOER	是	-
AIF	是	-	LBME	是	-
ALC	是	-	LBMI	是	-
ALIE	否	-	RACTIVE	是	接收立即停止，并不生成 ACK
ALIF	是	-	RAFIE	否	-
AMASK	否	只能在软件复位时可写	RAFIF	是	-

寄存器位名	软件复位	备注	寄存器位名	软件复位	备注
BEIE	否	-	RBALL	是	-
BEIF	是	-	RBUF	是	RB 被标记为空, 数值不定
BUSOFF	否	通过写1清除	RECNT	否	通过 BUSOFF 写1清零
EIE	否	-	REF_ID	否	-
EIF	否	-	REF_IDE	否	-
EPASS	否	-	RFIE	否	-
EPIE	否	-	RFIF	是	-
EPIF	是	-	RIE	否	-
EWARN	否	-	RIF	是	-
EWL	是	-	ROIE	否	-
			ROIF	是	-
F_FRESC	否	只能在软件复位时可写	ROM	否	
ROV	是	-	TSMODE	否	
RREL	是	-	TSNEXT	是	-
RSTAT	是		TSONE	是	-
SACK	是	-	TPIE	否	-
SELMASK	否	-	TPIF	是	-
S_PRESC	否	只能在软件复位时可以写	TPSS	是	-
S_SEG_1	否	只能在软件复位时可以写	TSFF	是	所有 STB SLOT 被标记为空
S_SEG_2	否	只能在软件复位时可以写	TSIE	否	-
SJW	否	只能在软件复位时可以写	TSIF	是	-
SSPOFF	是	-	TSSS	是	-
TACTIVE	是	发送立即停止	TSSTAT	是	所有 STB SLOT 被标记为空
TBE	是	-	TTEN	是	-
TBF	否	-	TTIF	是	-
TBPTR	否	-	TTIE	否	-
TBSEL	是	-	TTPTR	否	-
TBUF	是	STB 被标记为空, 指向 PTB	TTTBM	否	-
TDCEN	是	-	TTYPE	否	-
TECNT	否	可通过 BUSOFF=1清除	TT_TRIG	否	-
TEIF	是	-	TT_WTRIG	否	-
TPA	是	-	T_PRESC	否	-
TPE	是	-	WTIE	否	-
TSA	是	-	WTIF	是	
TSALL	是	-			

### 33.3.17. 时间触发 TTCAN

CAN-CTRL 为 ISO11898-4 规定的时间触发通信方式提供部分 (level 1) 硬件支持。本章节从以下 5 个部分介绍 TTCAN 功能。

#### 33.3.17.1. TTCAN 模式下的 TBUF 行为

##### TTTBM=1

TTTBM=1 时, PTB 和 STB SLOT 一样组成 TB SLOT, 通过 TBPTR 寄存器指定发送 BUF, 其中 TBPTR=0 时, 指向 PTB, TBPTR=1 是指向 STB SLOT1, 依次类推。主机可以通过 TBE 和 TBF 寄存器来标记发送 BUF SLOT。此时 TBSEL 和 TSNEXT 寄存器无任何意义从而可以被忽略。

TTTBM=1 时, PTB 不具有任何特殊的属性, 和 STB SLOT 一样, 传送完成标志也采用 TSIF。

TTCAN 模式时, 发送 BUF 没有 FIFO 模式和优先级仲裁模式, 同时也只有一个选定的 SLOT 可以发送数据。

TTCAN 模式下, 传输开始需要采用时间触发方式, TPE, TSONE, TSALL 和 TPA 被固定为 0 且被忽略。

##### TTTBM=0

TTTBM=0 时, 组合使用事件驱动通信和接收时间戳功能。在该模式下, PTB 和 STB 的功能和 TTEN=0 时一致, 因此 PTB 始终具有最高的优先级, 而 STB 可以工作在 FIFO 模式或者仲裁模式。

#### 33.3.17.2. TTCAN 功能

上电后, Time Master 需要根据 ISO 11898-4 协议进行初始化。一个 CAN 网络中, 最多可以有 8 个潜在的 Time Master。每一个 Time Master 都具有自己的参考消息 ID (ID 最后 3 位)。这些潜在的 Time Master 根据自己的优先级发送各自的参考消息。TTEN=1 后, 16 位的计数器开始工作, 当参考消息被成功接收或者 Time Master 成功发送参考消息时, CAN 控制器将 Sync\_Mark 拷贝给 Ref\_Mark, Ref\_Mark 将 cycle time 设置为 0。成功接收参考消息置位 RIF 标志而成功发送参考消息置位 TPIF 标志或者 TSIF 标志。此时主机需要准备下一个动作的触发条件。

触发条件可以是接收触发。该触发仅触发中断可用于检测期待的消息有没有被收到。触发条件也可以是发送触发。该触发开始发送通过 TTPTR 寄存器指定的 TBUF SLOT 里的数据。如果选定的 TBUF SLOT 被标记为空, 则不开始发送, 但置位中断标志。

#### 33.3.17.3. TTCAN 时序

CAN 控制器支持 ISO11898-4 level 1。包含的一个 16 位计数器工作在 AC\_PRESC, AC\_SEG\_1, AC\_SEG\_2 定义的位时间下。如果 TTEN=1, 则有一个额外的预分频器 T\_PRESC。

一帧数据的 SOF 时, 计数器的值为 Sync\_Mark。如果该帧数据为参考消息, 则将 Sync\_Mark 拷贝给 Ref\_Mark。cycle time 等于计数器的值减去 Ref\_Mark。该时间用作接收消息的时间戳或者发送消息的触发时间基准。

#### 33.3.17.4. TTCAN 触发方式

通过 TTYPE 寄存器定义 TTCAN 的触发方式, TTPTR 寄存器指定发送 SLOT, 而 TT\_TRIG 指定触发器的 cycle time。

包含以下五种触发方式:

- 立即触发



- 时间触发
- 单次发送触发
- 发送开始触发
- 发送停止触发

除了立即触发方式外，所有的触发器都使用 TTIF 标志。TTTBM=1 时，只支持时间触发方式。

#### 立即触发

通过写 TT\_TRIG 的高位（不在意写入的值），启动触发器。此模式下，TTPTR 选定的 TBUF SLOT 内的数据会立即发送。TTIF 不置位。

#### 时间触发

时间触发方式仅通过置位 TTIF 标志产生中断，并无其他功能。如果一个节点期待在特定的时间窗口内收到期待的数据，则可以使用时间触发方式。如果 TT\_TRIG 值小于实际的 cycle time，则 TEIF 置位且无其它动作。

#### 单次发送触发

单次发送触发方式用于在执行时间窗口内发送数据。

通过 TEW 位设定 ISO11898-4 规定的最多 16 个 cycle time 的 Tick，设定范围为 1~16。如果在规定的发送使能时间窗口内数据没有开始发送，则帧被丢弃。对应的发送 BUF SLOT 被标记为空，并且置位 AIF，对应的发送 BUF 内的数据不会被改写，因为可以通过置位 TPF 再次发送。

如果 TT\_TRIG 值小于实际的 cycle time，则 TEIF 置位且无其它动作。

#### 发送开始触发

发送开始触发方式用于仲裁时间窗口内，参与仲裁。如果 TTPTR 寄存器指定的消息没有被成功发送，可以使用发送停止触发来停止该发送。

如果 TT\_TRIG 值小于实际的 cycle time，则 TEIF 置位且无其它动作。

#### 发送停止触发

发送停止触发方式用于停止通过发送开始触发方式已经开始的发送。如果发送被停止，则发送帧被舍弃，置位 AIF 并将选定的 TBUF SLOT 标记为空，但 TBUF SLOT 内的数据不会被改写，可以通过置位 TBF 就可以再次发送。

如果 TT\_TRIG 值小于实际的 cycle time 则 TEIF 置位且执行停止。

### 33.3.17.5. TTCAN 触发看门时间

TTCAN 触发看门时间功能类似于看门狗功能，在 TTTBM=1 时使用。用来看门从上次成功接收到参考消息开始的时间。参考消息可以在周期 cycle time 中或者一个事件后被接收，应用程序应该根据具体情况设定合适的看门时间。

如果 cycle count 等于 TT\_WTRIG，则置位 WTIF。通过 WTIE 写 0，关闭看门触发。如果 TT\_WTRIG 比实际的 cycle time 小，则 TEIF 置位。

### 33.3.18. 中断

中断标志	描述
RIF	接收中断
ROIF	接收上溢中断
RFIF	接收 BUF 满中断

RAFIF	接收 BUF 将满中断
TPIF	PTB 发送中断
TSIF	STB 发送中断
EIF	错误中断
AIF	取消发送中断
EPIE	错误被动中断
ALIF	仲裁失败中断
BEIF	总线错误中断
WTIF	触发看门中断
TEIF	触发错误中断
TTIF	时间触发中断

## 33.4. 寄存器说明

### 33.4.1. 节点配置寄存器 (CAN\_TSNCR)

Address offset: 0x00

Reset value: 0x0201 0801

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	ROP	CES
														RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[15: 0]															
R															

Bit	Name	R/W	Reset Value	Function
31: 18	保留	-	-	保留
17	ROP	RW	0	限制操作 0:限制操作不使能 1:限制操作使能 当传输处于活动状态时, 不能更改 ROP。如果启用 ROP, 则无法启动数据帧发送。
16	CES	RW	1	CAN 错误信号 0:禁用错误信号 1:启用错误信号 如果 CES=1, 则使用错误标志发出错误信号并且错误计数器递增。此行为等同于 ISO 11898-1: 2015 (经典 CAN) 中的定义。 否则, 如果 CES=0, 则错误将导致协议异常事件, 并且不会修改错误计数器。

15: 0	Version [15: 0]	R	0x2050	CAN-CTRL 版本。VER_1 保存主要版本，VER_0 保存次要版本。例如：版本 5x15N00S00 由 VER_1=5 和 VER_0=15=0x0f 表示。
-------	--------------------	---	--------	--

### 33.4.2. 位时序配置寄存器 (CAN\_ACBTR)

Address offset: 0x04

Reset value: 0x0505 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	AC_SJW[6: 0]							Res	AC_SEG_2[6: 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
Res	Res	Res	Res	Res	Res	Res	AC_SEG_1[8: 0]								
-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	保留	-	-	保留
30: 24	AC_SJW	RW	0x05	同步跳转宽度 同步跳转宽度 $t_{SJW} = (AC\_SJW + 1) \cdot TQ$ 是缩短或延长重新同步位时间的最大时间。
23	保留	-	-	保留
22: 16	AC_SEG_2	RW	0x05	位时序段 2 时间 $t_{SEG\_2} = (AC\_SEG\_2 + 1) \cdot TQ$ 从采样点到位结束
15: 9	保留	-	-	保留
8: 0	AC_SEG_1	RW	0x08	位时序段 1 采样点将在位时间开始后设置为 $t_{SEG\_1} = (AC\_SEG\_1 + 2) \cdot TQ$ 。

### 33.4.3. 限制与预分频配置寄存器 (CAN\_RLSSP)

Address offset: 0x10

Reset value: 0x7700 0000

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

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

Bit	Name	R/W	Reset Value	Function
30: 28	RETLIM	RW	0x07	<p>自动重发次数限制</p> <p>111: 无限制 (仅受发送错误计数器 TECNT 限制)</p> <p>110: 7 次尝试</p> <p>...</p> <p>000: 1 次尝试 (不重传)</p> <p>如果传输过程中出现任何错误, CAN 节点能够在总线空闲时自动重试。可以使用 RETLIM 限制重传尝试的次数。</p> <p>RETLIM 可以随时更新, 但在更新后, 寄存器的内容需要几个 clock 的时间同步到 CAN 协议机器, 在此期间, REALIM 不能再次写入</p>
27	保留	-	-	保留
26: 24	REALIM	RW	0x07	<p>重新仲裁次数限制</p> <p>111: 无限制</p> <p>110: 7 次尝试</p> <p>...</p> <p>000: 1 次尝试 (无自动仲裁)</p> <p>如果两个或多个 CAN 节点尝试同时传输帧, 则较低优先级的帧会失去仲裁并静默后退, 而不会中断较高优先级的帧。CAN 节点能够在总线空闲时自动重试。使用 REALIM 可以限制重新仲裁尝试的次数。</p> <p>REALIM 可以随时更新, 但在更新后, 寄存器的内容需要几个 clock 的时间同步到 CAN 协议机器, 在此期间, REALIM 不能再次写入。</p>
23: 5	保留	-	-	保留
4: 0	PRESC	RW	0	<p>预分频器</p> <p>对模块输入时钟进行 (PRESC+1) 分频得到 TQ</p>

### 33.4.4. 状态寄存器 (CAN\_IFR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EWARN	EPASS	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
R	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	WTIF	TEIF	TTIF	EPIF	ALIF	BEIF	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	EWARN	R	0	到达设定的 ERROR WARNING LIMIT (Error WARNING limit reached) 0: RECNT 或者 TECNT 小于 EWL 设定值 1: RECNT 或者 TECNT 大于等于 EWL 设定值
30	EPASS	R	0	错误被动 (Error Passive mode active) 0: 节点是主动错误节点 1: 节点时被动错误节点
29: 14	保留	-	-	保留
13	WTIF	RW	0	TTCAN: 观察触发中断标志 如果周期计数达到 TT_WTRIG 定义的限制且 WTIE 已设置, 则 WTIF 将被置位。
12	TEIF	RW	0	TTCAN: 触发错误中断标志 设置 TEIF 的条件在 TTCAN 章节描述; 没有启用或禁用 TEIF 处理的位
11	TTIF	RW	0	时间触发中断标志 当 CYCLECOUNT 值=TT_TRIG 设定值且 TTIE=1时, TTIF 置位。如果 TT_TRIG 没有更新, 则 TTIF 只置位1次, 下一个计数周期, 即使 CYCLECOUNT 值=TT_TRIG 也不会置位。 通过应用程序写1清零该标志位。
10	EPIF	RW	0	错误被动中断标志。 如果 EPASS 改变并且 EPIE=1, EPIF 将置位。 通过应用程序写1清零该标志位。
9	ALIF	RW	0	仲裁失败中断标志 (Arbitration Lost Interrupt Flag) 0: 仲裁成功 1: 仲裁失败 通过应用程序写1清零该标志位。
8	BEIF	RW	0	总线错误中断标志 (Bus Error Interrupt Flag) 0: 无总线错误 1: 总线错误 通过应用程序写1清零该标志位。
7	RIF	RW	0	接收中断标志 (Receive Interrupt Flag) 0: 未收到数据帧 1: 接收到有效的数据帧或者远程帧 通过应用程序写1清零该标志位。
6	ROIF	RW	0	接收上溢中断标志 (Receive Overrun Interrupt Flag) 0: 无 RB 被覆盖 (oveRWrite) 1: RB 至少有一个被覆盖

Bit	Name	R/W	Reset Value	Function
				上溢时 ROIF 和 RFIF 同时置1。通过应用程序写1清零该标志位。
5	RFIF	RW	0	接收 BUF 满中断标志 (RB Full Interrupt Flag) 0: RB FIFO 未 1: RB FIFO 满 通过应用程序写1清零该标志位。
4	RAFIF	RW	0	接收 BUF 将满中断标志 (RB Almost Full Interrupt Flag) 0: 被填充的 RB SLOT 数目小于 AFWL 设定值 1: 被填充的 RB SLOT 数目大于等于 AFWL 设定值 通过应用程序写1清零该标志位。
3	TPIF	RW	0	PTB 发送中断标志 (Transmission Primary Interrupt Flag) 0: 没有 PTB 发送完成 1: 请求的 PTB 发送成功完成通过应用程序写1清零该标志位。 注意: TTCAN 模式时, TPIF 无效, 仅适用 TSIF 标志
2	TSIF	RW	0	STB 发送中断标志 (Transmission Secondary Interrupt Flag) 0: 没有 STB 发送完成 1: 请求的 STB 发送成功完成通过应用程序写1清零该标志位。 注意: TTCAN 模式时, TPIF 无效, 仅使用 TSIF 标志
1	EIF	RW	0	错误中断标志 (Error Interrupt Flag) 0: BUSOFF 位未发生变化, 或者错误计数器的值与 ERROR warning limit 设定值的相对关系未发生变化。 1: BUSOFF 位发生变化, 或者错误计数器的值与 ERROR warning limit 设定值的相对关系发生变化。比如错误计数器的值从小于设定值变为大于设定值, 或者从大于设定值变为小于设定值。 通过应用程序写1清零该标志位。
0	AIF	RW	0	取消发送中断标志 (Abort Interrupt Flag) 0: 未取消发送数据 1: 通过 TPA 或 TSA 请求的发送消息被成功取消。 通过应用程序写1清零该标志位。

### 33.4.5. 中断使能寄存器 (CAN\_IER)

Address offset: 0x18

Reset value: 0x0004 68FE

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
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<b>15</b>	<b>14</b>	<b>13</b>	<b>12</b>	<b>11</b>	<b>10</b>	<b>9</b>	<b>8</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
Res	Res	WTIE	Res	TTIE	EPIE	ALIE	BEIE	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	Res
-	-	RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 14	保留	-	-	保留
13	WTIE	RW	1	触发看门中断使能 (Watch Trigger Interrupt Enable) 0: 禁止 1: 使能
12	保留	-	-	保留
11	TTIE	RW	1	时间触发中断使能 (Time Trigger Interrupt Enable) 0: 禁止 1: 使能
10	EPIE	RW	0	到达设定的 ERROR WARNING LIMIT (Error WARNING limit reached) 0: RECNT 或者 TECNT 小于 EWL 设定值 1: RECNT 或者 TECNT 大于等于 EWL 设定值
9	ALIE	RW	0	仲裁失败中断使能 (Arbitration Lost Interrupt Enable) 0: 禁止 1: 使能
8	BEIE	RW	0	总线错误中断使能 (Bus Error Interrupt Enable) 0: 禁止 1: 使能
7	RIE	RW	1	接收中断使能 (Receive Interrupt Enable ) 0: 禁止 1: 使能
6	ROIE	RW	1	接收上溢中断使能 (Receive Overrun Interrupt Enable ) 0: 禁止 1: 使能
5	RFIE	RW	1	接收 BUF 满中断使能 (RB Full Interrupt Enable) 0: 禁止 1: 使能
4	RAFIE	RW	1	接收 BUF 将满中断使能 (RB Almost Full Interrupt Enable) 0: 禁止 1: 使能
3	TPIE	RW	1	PTB 发送中断使能 (Transmission Primary Interrupt Enable)

Bit	Name	R/W	Reset Value	Function
				0: 禁止 1: 使能
2	TSIE	RW	1	STB 发送中断使能 (Transmission Secondary Interrupt Enable) 0: 禁止 1: 使能
1	EIE	RW	1	错误中断使能 (Error Interrupt Enable) 0: 禁止 1: 使能
0	保留	-	-	保留

### 33.4.6. 传输状态寄存器 (CAN\_TSR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	TSTAT_H[2: 0]			HANDLE_H[7: 0]							
-	-	-	-	-	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	Res	TSTAT_L[2: 0]			HANDLE_L[7: 0]							
-	-	-	-	-	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 27	保留	-	-	保留
26: 24	TSTAT_H	R	0	已经完成传输的发送帧状态码
23: 16	HANDLE_H	R	0	已经完成传输的发送帧 handle 值
15: 11	保留	-	-	保留
10: 8	TSTAT_L	R	0	当前正在传输的发送帧状态码
7: 0	HANDLE_L	R	0	当前正在传输的发送帧 handle 值

### 33.4.7. 全局配置寄存器 (CAN\_MCR)

Address offset: 0x28

Reset value: 0x0090 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SACK	ROM	ROV	RREL	RBALL	Res	RSTAT[1: 0]		Res	TSNEXT	TSMODE	TTTBM	Res	TSFF	TSSTAT[1: 0]	
RW	RW	R	RW	RW	-	R	R	RW				-	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RE-SET	LBME	LBMI	Res	Res	Res	Res	BUSOFF



RW	-	-	-	-	RW
----	---	---	---	---	----

Bit	Name	R/W	Reset Value	Function
31	SACK	RW	0	自应答 (Self-ACKnowledge) 0: 无自应答 1: LBME=1时, 使能自应答功能
30	ROM	RW	0	接收 BUF 上溢模式设定位 (Receive buffer Overflow Mode) 0: 最早接收到的数据被覆盖 1: 新接收到的数据不被存储
29	ROV	R	0	接收 BUF 上溢标志位 (Receive buffer OVerflow) 0: 无上溢 1: 上溢, 最少有一个数据丢失 通过写 RREL 为1清零。
28	RREL	RW	0	释放接收 BUF (Receive buffer RElease) 0: 不释放 1: 表示该接收 BUF 已经被读取过, RBUF 寄存器指向下一个 RB SLOT。
27	RBALL	RW	0	接收 BUF 数据存储素所有的数据帧 (Receive Buffer stores ALL data frames) 0: 正常模式 1: 存储所有的数据包括有错误的的数据。
25: 24	RSTAT	R	0	接收缓冲区状态 00 - 空 01 -> 空且 < 几乎已满 (AFWL) 10 - 几乎已满 (可通过 AFWL 编程设定阈值) 但未满且无溢出 11 - 已满 (溢出时保持置位)
23	保留	-	-	保留
22	TSNEXT	RW	0	下一个 STB (Transmit buffer Secondary NEXT) 0: 无动作 1: 当前 STB SLOT 已填充, 指向下一个 SLOT 应用程序将 TBUF 中的数据写完后, 通过置位 TSNEXT 位标识当前 STB SLOT 已经被填充, 从而硬件将 TBUF 指向下一个 STB SLOT。 被 TSNEXT 位标识的 STB SLOT 中的数据可以通过 TSONE 或者 TSALL 位发送。该位通过应用程序写1, 硬件清零。 所有的 STB SLOT 被填满后, TSNEXT 保持为1直到有 STB SLOT 被释放。

Bit	Name	R/W	Reset Value	Function
				注意: TTCAN 模式时此位固定为0。
21	TSMODE	RW	0	STB 发送模式 (Transmit buffer Secondary operation MODE) 0: FIFO 模式 1: 优先级模式 FIFO 模式根据数据帧写入的先后顺序发送。 优先级模式根据 ID 自动判断, ID 越小, 优先级越高。无论何种模式, PTB 具有最高的优先级。 注意: TSMODE 位只能在 STB 空时设定。
20	TTTBM	RW	1	TTCAN BUF 模式 (TTCAN Transmit Buffer Mode) TTEN=0时, TTTBM 被忽略。 0: TSMODE 决定, PTB 和 STB 1: 通过 TBPTR 和 TTPTR 设定 TTCAN 模式时, 只需要接收时间戳功能时, 此位可以设置为0, 通过 TSMODE 决定使用 PTB 还是 STB。 注意: TSMODE 位只能在 STB 空时设定。
19	保留	-	-	保留
18	TSFF	R	0	TTEN=0 or TTTBM=0: STB 满标志 (Transmit Secondary buffer Full Flag) 0: STB SLOT 没有被全部填充 1: STB SLOT 被全部填充 TTEN=1 and TTTBM=1: TB 满标志 (Transmit buffer Full Flag) 0: TBPTR 选择的发送 BUF 没有被全部填充 1: TBPTR 选择的发送 BUF 被全部填充
17: 16	TSSTAT	R	0	STB 状态 (Transmission Secondary Status bits) TTEN=0 或 TTEN=1 & TTTBM=0 00: STB 空 01: STB 小于等于半满 10: STB 大于半满 11: STB 满 TTEN=1 且 TTTBM=1 00: PTB 和 STB 空 01: PTB 和 STB 非满 10: 保留 11: PTB 和 STB 满
15	TBSEL	RW	0	发送 BUF 选择位 (Transmit Buffer Select) 0: PTB 1: STB

Bit	Name	R/W	Reset Value	Function
				当 TTEN=1&TTTBM=1时, TBSEL 被复位成复位值。 注意: 写 TBUF 寄存器或者 TSNEXT 位时, 此位需要保持定值。
14	LOM	RW	0	静默模式使能位 (Listen Only Mode) 0: 禁止静默模式 1: 使能静默模式 LOM=1&LBME=0时禁止发送。 LOM=1&LBME=1时禁止应答相应接收到的帧以及错误帧, 但可以发送数据。 注意: 通信中禁止设定该位。
13	STBY	RW	0	收发器待机模式 0 - 禁用 1 - 启用 该寄存器位连接到输出信号 stby, 可用于控制收发器的待机模式。 如果 TPE=1、TSONE=1 或 TSALL=1, 则 STBY 不能设置为 1。 如果主机将 STBY 设置为 0, 那么在主机请求新的传输之前, 主机需要等待收发器启动所需的时间。
12	TPE	RW	0	PTB 发送使能位 (Transmit Primary Enable) 0: 禁止 PTB 发送 1: 使能 PTB 发送 此位使能后, PTB 中的 Mailbox 将在下一个可以发送的位置被发送。已经开始的 STB 发送将继续, 但是下一个等待的 STB 发送会被延迟到 PTB 发送完成后再进行。 该位写1后将保持为1直到 PTB 发送完成或者通过 TPA 取消发送。软件不能通过写0清除该位。 以下情况 TPE 被硬件复位成复位值: RESET=1 BUSOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1
11	TPA	RW	0	PTB 发送取消位 (Transmit Primary Abort) 0: 不取消 1: 取消已经通过 TPE 置1请求但还未开始的 PTB 发送 该位软件写1但是通过硬件清零。通过写1可以清零 TPE 位, 因此和 TPE 同时写1。以下情况 TPE 被硬件复位成复位值: RESET=1 BUSOFF=1

Bit	Name	R/W	Reset Value	Function
				TTEN=1&TTTBM=1
10	TSONE	RW	0	<p>发送一帧 STB 数据设定位 (Transmit Secondary ONE frame)</p> <p>0: 不发送</p> <p>1: 发送一帧 STB 数据</p> <p>FIFO 模式中, 发送最早写入的数据, 优先级模式里发送最高优先级的数据</p> <p>该位写1后将保持为1直到 STB 发送完成或者通过 TSA 取消发送。软件不能通过写0清除该位。</p> <p>以下情况 TSONE 被硬件复位成复位值:</p> <p>RESET=1</p> <p>BUFOFF=1</p> <p>LOM=1&amp;LBME=0</p> <p>TTEN=1&amp;TTTBM=1</p>
9	TSALL	RW	0	<p>发送所有的 STB 数据设定位 (Transmit Secondary ALL frame)</p> <p>0: 不发送</p> <p>1: 发送 STB 中所有的数据</p> <p>该位写1后将保持为1直到 STB 发送完成或者通过 TSA 取消发送。软件不能通过写0清除该位。</p> <p>以下情况 TSALL 被硬件复位成复位值:</p> <p>RESET=1</p> <p>BUSOFF=1</p> <p>LOM=1&amp;LBME=0</p> <p>TTEN=1&amp;TTTBM=1</p>
8	TSA	RW	0	<p>STB 发送取消位 (Transmit Secondary Abort)</p> <p>0: 不取消</p> <p>1: 取消已经通过 TSONE 或者 TSALL 置1请求但还未开始的 STB 发送</p> <p>该位通过软件写1但是通过硬件清零。写1可以清零 TSONE 或者 TSALL 位。以下情况 TSA 被硬件复位成复位值:</p> <p>RESET=1</p> <p>BUSOFF=1</p>
7	RESET	RW	1	<p>复位请求位</p> <p>0: 不请求局部复位</p> <p>1: 请求局部复位</p> <p>部分寄存器只能在 RESET=1时进行写操作, 具体请参考软件复位功能, 当该节点进入 BUS OFF 状态时, 硬件自动将</p>

Bit	Name	R/W	Reset Value	Function
				RESET 位置1。请注意，当 RESET=0后需要11个 CAN bit times 该节点才能参与通信。
6	LBME	RW	0	外部回环模式使能位 0: 禁止外部回环模式 1: 使能外部回环模式 注意：通信中禁止设定该位。
5	LBMI	RW	0	内部回环模式使能位 0: 禁止内部回环模式 1: 使能内部回环模式 注意：通信中禁止设定该位。
4: 1	保留	-	-	保留
0	BUSOFF	RW	0	总线关闭状态 0: 总线有效状态 1: 总线关闭状态 注意：写1可以清除 TECNT 和 RECNT 寄存器，一般只用于调试用途

### 33.4.8. 错误警告寄存器 (CAN\_WECR)

Address offset: 0x2C

Reset value: 0x0000 001B

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

Bit	Name	R/W	Reset Value	Function
31: 24	TECNT	R	0	发送错误计数器 (Transmit Error Count) 发送错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255为最大值。
23: 16	RECNT	R	0	接收错误计数器 (Receive Error Count) 接收错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255为最大值。
15: 13	KOER	RW	0	错误类别 (Kind Of Error) 000: 无错误 001: 位错误 010: 形式错误

Bit	Name	R/W	Reset Value	Function
				011: 填充错误 100: 应答错误 101: CRC 错误 110: 其他错误 111: 保留 有错误时 KOER 位更新, 正常发送接收时 KOER 位保持不变。
12: 8	ALC	R	0	仲裁失败位置捕捉 (Arbitration Lost Capture) 仲裁失败时 ALC 记录一帧数据中仲裁失败时的位置。
7: 4	AFWL	RW	0x1	接收 BUF 将满 Warning Limit (receive buffer Almost Full Warning Limit) 设定值范围为1~3。 AFWL=0无意义, 当做 AFWL=1处理。
3: 0	EWL	RW	0xB	Error Warning Limit 编程值 (Programmable Error Warning Limit) Error Waring Limit= (EWL+1) *8。 该寄存器设定值影响 EIF 标志。

### 33.4.9. 参考 ID 寄存器 (CAN\_REFMSG)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31	REF_IDE	RW	0	参考消息的 IDE 位 (REFerence message IDE bit) 0: 标准格式 1: 扩展格式
30: 29	保留	-	-	保留
28: 0	REF_ID	RW	0	参考消息的 ID 位 (REFerence message IDentifier) REF_IDE=0: REF_ID[10: 0]有效 REF_IDE=1: REF_ID[28: 0]有效 REF_ID 用于检测参考消息, 适用于发送和接收。 检测到参考消息后, 当前帧的 Sync_Mark 则变成 Ref_Mark。

				REF_ID[2: 0]固定为0, 并不检查其值, 这样最多可以支持8个潜在的 time master。 当 REF_MSG 的最高字节写操作后, 则需要等待6个 CAN 时钟周期以完成 REF_MSG 向 CAN 时钟域的传递。
--	--	--	--	--

### 33.4.10. TTCAN 配置寄存器 (CAN\_TTCR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	T_PRESC[1: 0]		TTEN	TBE	TBF	TBPTR[5: 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
TEW[3: 0]				Res	TTYPE[2: 0]			Res	Res	TTPTR[5: 0]					
RW	RW	RW	RW	-	RW	RW	RW	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 27	保留	-	-	保留
26: 25	T_PRESC	RW	0	TTCAN 计数器预分频 00b: ACBTR 寄存器设定的位时间的1分频 01b: ACBTR 寄存器设定的位时间的2分频 10b: ACBTR 寄存器设定的位时间的4分频 11b: ACBTR 寄存器设定的位时间的8分频 TTCAN 时基是由 PRESC、AC_SEG_1 和 AC_SEG_2 定义的 CAN 位时间。 使用 T_PRESC 定义了额外的预缩放因子 1、2、4 或 8。 T_PRESC 只能在 TTEN=0 时修改, 也可以修改 T_PRESC 并同时设置 TTEN 与一次写访问。
24	TTEN	RW	0	TTCAN 使能 (Time Trigger Enable) 0: 禁止 1: 使能 TTCAN, 计数器开始计数。
23	TBE	RW	0	设置 TB 为空 (set TB slot to “empty”) 0: 无操作 1: 被 TBPTR 选择的 SLOT 被标记为空 当 SLOT 被标记为空并且 TSFF=0时, TBE 自动复位为0。 如果设定此位为1时, 被选定的 SLOT 中存在数据正在发送状态则 TBE=1, 则等到发送完成、发送错误或者发送取消后 TBE 复位为0。 TBE 优先级高于 TBF。
22	TBF	RW	0	设置 TB 已填充 (set TB slot to “Filled”)

Bit	Name	R/W	Reset Value	Function
				0: 无操作 1: 被 TBPTR 选择的 SLOT 被标记为已填充 当 SLOT 被标记为已填充并且 TSFF=1时, TBE 自动复位为0。
21: 16	TBPTR	RW	0	TB SLOT 指针 (Pointer to a TB message slot) 000: 指向 PTB 001: 指向 STB SLOT1 010: 指向 STB SLOT2 011: 指向 STB SLOT3 其他: 无效设置 被指向的 TB SLOT 可以通过 TBUF 进行读写访问, 并且可以通过 TBE 和 TBF 来标记是否已经被填充。 TTCAN 模式下, TBSEL 和 TSNEXT 寄存器无效 注意: 仅可以在 TSFF=0时对该位进行写操作
15: 12	TEW	RW	0	发送使能窗口 (Transmit Enable Window) 用于 TTCAN 的单次发送触发模式 (Single Shot Transmit Trigger), 可以设定 TEW+1个 cycle time 的窗口, 发送仅在此窗口内被允许。
11	保留	-	-	保留
10: 8	TTYPE	RW	0	触发类型 (Trigger Type) 000: 立即触发 (Immediate Trigger for immediate transmission) 001: 时间触发 (Time Trigger for receive triggers) 010: 单次发送触发 (Single Shot Transmit Trigger for exclusive time windows) 011: 发送开始触发 (Transmit Start Trigger for merged arbitrating time windows) 100: 发送停止触发 (Transmit Stop Trigger for merged arbitrating time windows) 其他: 保留 触发时间通过 TT_TRIG 寄存器设定, TB Slot 通过 TTPTR 选择。
7: 6	保留	-	-	保留
5: 0	TTPTR	RW	0	发送触发器 TB slot 指针 (Transmit Trigger TB slot Pointer) 000: 指向 PTB 001: 指向 STB SLOT1 010: 指向 STB SLOT2 011: 指向 STB SLOT3



Bit	Name	R/W	Reset Value	Function
				其他：设定禁止 如果指向的 TB SLOT 被标记为空，当到达触发时间后，TEIF 置位。

### 33.4.11. TTCAN 触发寄存器 (CAN\_TTTR)

Address offset: 0x38

Reset value: 0x0202 0003

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

Bit	Name	R/W	Reset Value	Function
31: 16	TT_WTRIG	RW	0xffff	看门触发器的 cycle time 当 TT_WTRIG 的最高字节操作后，TT_WTRIG 值开始向 CAN 时钟域传递，因此如果 BYTE 操作，需先写低字节再写高字节。
15: 0	TT_TRIG	RW	0	触发时间 (Trigger Time) 用于指定触发器的 cycle time，对于发送触发器来说发送 SOF 时间大约是 TT_TRIG 设定值+1 当 TT_WTRIG 的最高字节操作后，TT_WTRIG 值开始向 CAN 时钟域传递，因此如果 BYTE 操作，需先写低字节再写高字节。

### 33.4.12. 内存状态寄存器 (CAN\_SCMS)

Address offset: 0x3C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 29	保留	-	-	保留
28: 27	HELOC	R	0x0	主机端内存错误位置

				<p>00 – 从主机端访问时没有错误</p> <p>01 – 在 TBUF 中从主机端访问时出错</p> <p>10 – 在 RBUF 中从主机端访问时出错</p> <p>11 – 在 ACF 中从主机端访问时出错</p> <p>在从主机端读取访问期间，HELOC 将根据每个新错误进行更新。</p> <p>这已经足够了，因为从 CAN 端读取访问期间的读取错误将由 ACFA、TXS 和 TXB 发出信号。</p> <p>HELOC 只会在出现错误的情况下更新，但不会在由更正的单位错误引起的警告的情况下更新。</p>
26	TXB	R	0x0	<p>传输块</p> <p>0 – 正常运行</p> <p>1 – 传输受阻</p> <p>如果 MDEIF 或 MAEIF 在 CAN 协议机器正在读取数据进行传输时由于错误而被设置，则传输将立即被阻止。</p> <p>如果设置了 SEIF，那么传输也会立即被阻止。</p> <p>如果 TXB=1，则错误计数器被冻结。</p> <p>如果 RESET=1，则 TXB 被复位。</p>
25	TXS	R	0x0	<p>传输停止</p> <p>0 – 正常运行</p> <p>1 – 传输停止</p> <p>如果在优先排序机器访问内存时由于错误而设置了 MDEIF 或 MAEIF，则停止任何新的传输。如果有一个活动传输，这将在停止之前完成，但如果在此传输期间发生错误，则不会开始重新传输。</p> <p>如果 RESET=1，则 TXS 被重置。</p>
24	ACFA	RW	0x0	<p>筛选器使能</p> <p>0 – ACF 正常运行</p> <p>1 – ACF 禁用：接受所有接收到的帧</p> <p>如果由于 ACF 的地址范围错误而设置 MDEIF 或 MAEIF，则设置 ACFA。然后接受过滤被禁用，所有的帧都将被接受。</p> <p>ACFA 可以像中断标志一样通过向其写入 1 来复位。但是由于 ACFA 将在接收仍处于活动状态时设置，因此需要在接收完成后重新设置它，例如 RIF 已设置。</p> <p>如果 RESET=1，ACFA 也会被重置。</p>
23: 0	保留	-	-	保留

### 33.4.13. 筛选器组控制寄存器 (CAN\_ACFCR)

Address offset: 0x44

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	AE_11	AE_10	AE_9	AE_8	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_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	Res	Res	Res	Res	Res	Res	Res	Res	ACFADR			
-	-	-	-	-	-	-	-	-	-	-	-	RW			

Bit	Name	R/W	Reset Value	Function
31: 28	保留	-	-	保留
27: 16	AE_x	RW	0x1	ACF 使能 1 – 使能 0 – 禁止 每个验收滤波器 (ACFC / ACFM) 可以单独启用或禁用。 硬件复位后, 默认情况下仅启用过滤器编号 0。
15: 4	保留	-	-	保留
3: 0	ACFADR	RW	0x0	筛选器地址 ACFADR 指向特定的验收过滤器。可以使用寄存器 ACFC 和 ACFM 访问所选过滤器。 ACFADR>ACF_NUMBER-1 的值是无意义的, 并自动被视为值 ACF_NUMBER-1

#### 33.4.14. 筛选器组 code 寄存器 (CAN\_ACFC)

Address offset: 0x48

Reset value: 0xFFFF XXXX

0x00	CAN_ID
0x04	CAN_FORMAT
0x08	CAN_TYPE

注: 上述寄存器的控制含义请参考33.3.7 LLC 帧格式定义。

#### 33.4.15. 筛选器组 mask 寄存器 (CAN\_ACFM)

Address offset: 0x58

Reset value: 0xFFFF XXXX

0x00	CAN_ID
0x04	CAN_FORMAT
0x08	CAN_TYPE

注: 上述寄存器的控制含义请参考33.3.7 LLC 帧格式定义。

### 33.4.16. CAN 接收 BUF 寄存器 (CAN\_RBUF)

**Address offset:** 0x70

**Reset value:** 0xFFFF XXXX

注：上述寄存器的控制含义请参考33.3.7 LLC 帧格式定义。

### 33.4.17. CAN 发送 BUF 寄存器 (CAN\_TBUF)

**Address offset:** 0x94

**Reset value:** 0xFFFF XXXX

注：上述寄存器的控制含义请参考33.3.7 LLC 帧格式定义。

## 34. USB 全速设备接口 (USBD)

### 34.1. 简介

USBD 外设实现了 USB2.0 全速总线和 APB 总线间的接口。

USBD 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

### 34.2. USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 6 个 USB 端点 (端点 0 ~端点 5)
- 专用的 1024 字节的数据包缓存存储
- CRC (循环冗余校验) 生成/校验, 反向不归零 (NRZI) 编码/解码和位填充
- 支持控制传输/同步传输/批量传输/中断传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/恢复操作
- 帧锁定时钟脉冲生成

### 34.3. USB 设备框图

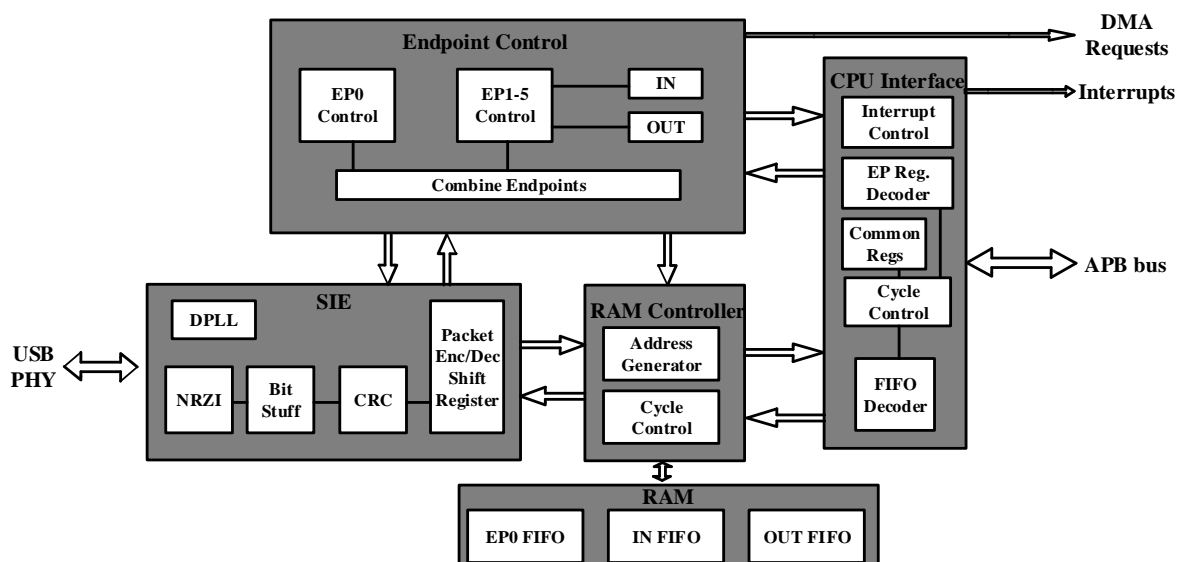


图 34-1 USB 设备框图

### 34.4. 功能描述

USB 模块为 PC 主机和微控制器之间提供了符合《USB 规范》的通信连接。PC 主机和微控制器之间的数据传输是通过一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 模块直接访问。这块专用数据缓冲区的大小是固定的，端点 0 最大使用 64 字节缓冲区，EP1 最大使用 512 字节缓冲区，EP2、EP3、EP4 都是使用 128 字节缓冲区，EP5 使用 64 字节缓冲区，同一个端点的 IN 和 OUT 缓冲区是

共享的。USB 模块同 PC 主机通信，根据《USB 规范》实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。

当一有效的令牌分组被 USB 模块识别时，相关的数据传输随之发生。USB 模块通过一个内部的寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输都完成后，如果需要，就根据传输的方向，发送或接收适当的握手分组。

USB 模块使用固定的时钟，此时钟被 USB 标准定义为 48 MHz。

#### 34.4.1. 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口引擎 (SIE)：SIE 处理 NRZI 编码/解码、位填充/解填充和 CRC 生成/检查。它从输入的 48 MHz 时钟里产生一个 12 MHz 的 USB 时钟，当从 USB 主机接收到数据时，数据流就会同步到该时钟下。它会为传输的包 (Packets) 产生头文件 (Headers) 并在接收到数据包时进行解码。
- 端点控制器：使用两种状态控制器，一个用于端点 0 上的控制传输，另一个用于端点 1—5 上的批量、中断或同步传输。
- CPU 接口：CPU 接口允许访问每个端点的控制/状态寄存器和 FIFO。当数据包被成功传输或接收，或者当主机进入 SUSPEND 模式或者从 SUSPEND 模式恢复时，它会产生一个中断给 CPU。
- RAM 控制器：用于控制缓存 CPU 和 USB 之间数据包的同步单端 RAM。

#### 34.4.2. 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。

#### 34.4.3. USB 复位状态

发生 USB 复位时，USB 模块进入以下描述的状态：

- USB\_CR 寄存器的 ADD[6: 0]位置 0；
- USB\_FRAME 寄存器的 INDEX 位置 0；
- 将所有的 FIFO 中的数据清空；
- 清空所有控制/状态寄存器；
- 使能除了 SUSPEND 以外的所有中断；
- 产生一个 USB 的复位中断。

#### 34.4.4. USB 挂起/唤醒模式

当 USB 总线在 3 ms 内都是空闲状态，并且 USB\_CR 寄存器中的 Enable\_Suspend 位已被置 1 时，USB 模块将进入 SUSPEND 模式。如果 USB\_INTRE.EN\_Suspend 位被置位，则会产生一个 SUSPEND 中断。

当 USB 进入 SUSPEND 模式后，内部 12 MHz 的数据时钟将会被停止以此来降低功耗。同时，输入的 48MHz 系统时钟需要一直保持，以便 USB 模块可以检测 USB 总线上的信号。如果系统时钟停止了的话，USB 模块将无法检测到 USB 总线上的信号，用户将不得不重启系统时钟。

当 USB 模块处于 SUSPEND 模式时，检测到 USB 总线上的 K 状态时可以把设备唤醒，从而退出 SUSPEND 模式并产生一个 RESUME 中断。当然应用程序也可以设置 USB\_CR.Resume 位来强制设备离开 SUSPEND 模式，在这种情况下，应用程序应该在 10 毫秒（最多 15 毫秒）后清除该位。至此 USB 模块完全退出 SUSPEND 模式并且不会产生 RESUME 中断。

#### 34.4.5. IN 分组（用于数据发送）

IN 端点 1-5 的 FIFO 大小分别为 512 Bytes、128 Bytes、128 Bytes、128 Bytes、64 Bytes，但 IN 传输的最大包大小是可以配置的，由写入每个端点的 InMaxP (USB\_INEPxCSR) 寄存器决定。

由于每个要发送的数据包都被加载到 IN FIFO 中，所以需要设置 USB\_INEPxCSR 寄存器中的 InPktRdy 位。如果设置了 USB\_INEPxCSR 寄存器中的 AutoSet 位，则在 FIFO 加载最大数据包时，InPktRdy 位会自动设置。对于小于最大值的数据包，InPktRdy 是需要手动设置的（由应用程序设置）。

当 InPktRdy 位被设置时，USB\_INEPxCSR 寄存器中的 FIFONotEmpty 位也会被设置，接着这个数据包就准备发送了。

当数据包被成功发送时，USB\_INEPxCSR.InPktRdy 位和 USB\_INEPxCSR.FIFONotEmpty 位被清除，如果相应的中断被使能了，那么就会产生相应的 IN 端点的中断。然后下一个数据包就能够被加载到 FIFO 中。

##### 34.4.5.1. IN 数据包双包缓存功能

如果 IN 端点的 FIFO 大小至少是这个端点最大包大小的两倍（取决于 USB\_INEPxCSR.InMaxP 的大小），那么两个数据包可以都缓存在 FIFO 中。

当第一个数据包加载到 FIFO 中后，USB\_INEPxCSR.InPktRdy 会被先设置再立即清除，生成相应的 IN 端点中断。然后第二个数据包就可以加载到 FIFO 中，同时 USB\_INEPxCSR.InPktRdy 将再次设置，这样两个包就都可以发送了。

当第一个数据包被成功发送时，USB\_INEPxCSR.InPktRdy 位的清除以及相应 IN 端点中断的产生表明另一个数据包现在可以加载到 FIFO 中。此时 USB\_INEPxCSR.FIFONotEmpty 位的状态表示可能加载了多少包。如果设置了 USB\_INEPxCSR.FIFONotEmpty 位，那么在 FIFO 中就会有另一个数据包，并且只会有一个数据包被加载。如果 USB\_INEPxCSR.FIFONotEmpty 位是复位状态，则表明 FIFO 中没有数据包，可以加载两个数据包。

#### 34.4.6. OUT 分组（用于数据接收）

OUT 端点 1-5 的 FIFO 大小分别为 512 Bytes、128 Bytes、128 Bytes、128 Bytes、64 Bytes，但 OUT 传输的最大包大小是可编程的，由写入每个端点的 OutMaxP (USB\_OUTEPxCSR) 寄存器决定。

当一个数据包被接收到并放置在 OUT FIFO 中时，USB\_OUTEPxCSR 中的 OutPktRdy 位和 FIFOFull 位将被设置，并生成适当的 OUT 端点中断，以表明一个数据包现在可以从 FIFO 读取了。在数据包被读取后，需要清除 USB\_OUTEPxCSR.OutPktRdy 位，以便接收更多的数据包。如果设置了

USB\_OUTEPxCSR 中的 AutoClear 位，并且从 FIFO 中读走了最大的数据包，则 OutPktRdy 位将被自动清除。FIFOFull 位也被清除。对于小于最大包大小的数据包，USB\_OUTEPxCSR.OutPktRdy 必须手动清除（即由应用程序清除）。

#### 34.4.6.1. OUT 包双包缓存功能

如果 OUT 端点 FIFO 的大小至少是该端点最大数据包大小的两倍（在 USB\_OUTEPxCSR.OutMaxP 寄存器中设置），则 OUT FIFO 可以缓存两个数据包。

当第一个要接收的数据包被加载到 OUT FIFO 时，USB\_OUTEPxCSR 中的 OutPktRdy 位被设置，相应的 OUT 端点中断生成，以表示数据包现在可以从 FIFO 中读取。

注意：USB\_OUTEPxCSR 中的 FIFOFull 位在这一点上没有设置，它只有在第二个数据包被接收并加载到 OUT FIFO 时才设置。

读走第一个数据包后，需要清除 USB\_OUTEPxCSR.OutPktRdy 位，以便接收更多的数据包。如果设置了 USB\_OUTEPxCSR 中的 AutoClear 位，并且从 FIFO 中读走了最大大小的数据包，则 OutPktRdy 位将被自动清除。对于小于最大大小的数据包，USB\_OUTEPxCSR.OutPktRdy 必须手动清除（即由应用程序清除）。

如果清除 USB\_OUTEPxCSR.OutPktRdy 时 USB\_OUTEPxCSR.FIFOFull 位设置为 1,将首先清除 USB\_OUTEPxCSR.FIFOFull 位。然后 USB\_OUTEPxCSR.OutPktRdy 被再次置位，以表明在 FIFO 中有另一个数据包等待被读走。

### 34.4.7. 控制传输

端点 0 是 USB 的控制端点。因此，驱动端点 0 所需的例程比驱动其他端点所需的例程更复杂。

应用程序可以通过端点 0 来接收处理所有的标准设备请求。这些在《USB 总线规范》中有描述。标准设备请求可以分为三类：零数据请求、写请求和读请求。本节将介绍应用程序处理不同类型的设备请求时必须执行的事件序列。

注意：与任何标准设备请求相关联的 Setup 令牌包应该包括一个 8 字节的命令。任何包含命令字符超过 8 字节的 Setup 包将被自动拒绝。

#### 34.4.7.1. 零数据请求

零数据请求的所有信息都包含在 8 字节命令中，不需要传输额外的数据。零数据标准设备请求的例子有：SET\_FEATURE, CLEAR\_FEATURE, SET\_ADDRESS, SET\_CONFIGURATION, SET\_INTERFACE（参考《USB 总线规范》）。

与所有请求一样，当应用程序接收到一个端点 0 中断时，事件序列将开始。

USB\_EP0CSR.OutPktRdy 位也将被置位。8 字节的命令应该从端点 0 FIFO 读取，解码并采取适当的操作。例如，如果命令是 SET\_ADDRESS，则应该将命令中包含的 7 位地址值写入 USB\_CR.ADD 寄存器。

然后应用程序应该设置 USB\_EP0CSR.ServicedOutPktRdy 位（表明该命令已经从 FIFO 读取）和设置 USB\_EP0CSR.DataEnd 位（表明该请求不需要进一步的数据）。

当主机移动到 USB 协议中规定的请求的状态阶段时，USB 模块将生成第二个端点 0 中断，以表明请求已经完成。应用程序不需要进一步的操作；第二次中断只是确认请求成功完成。



如果该命令是一个无法识别的命令，或者由于某些其他原因不能执行，那么当它被解码后，应用程序应该设置 USB\_EP0CSR.ServicedOutPktRdy 位和 USB\_EP0CSR.SendStall 位。当主机移动到请求的状态阶段时，设备将发送一个 STALL 来告诉主机这个请求没有被执行。USB 模块将生成第二个端点 0 中断，并设置 USB\_EP0CSR.SentStall 位。

如果主机在设置 USB\_EP0CSR.DataEnd 位之后发送更多的数据，那么设备将发送一个 STALL 握手包。USB 设备会生成一个端点 0 中断，并设置 USB\_EP0CSR.SentStall 位。

#### 34.4.7.2. 写请求

写请求包括一个（或多个）额外的数据包，这些数据包在 8 字节命令之后从主机发送过来。一个写标准设备请求的例子是：SET\_DESCRIPTOR（参考《USB 总线规范》）。

与所有请求一样，当应用程序接收到一个端点 0 中断时，事件序列将开始。

USB\_EP0CSR.OutPktRdy 位也将被置位。8 字节的命令应该从端点 0 FIFO 读取并解码。

与零数据请求一样，应用程序应该设置 USB\_EP0CSR.ServicedOutPktRdy 位（表明该命令已经从 FIFO 读取），但在这种情况下不应该设置 USB\_EP0CSR.DataEnd 位（表明需要更多的数据）。

当第二个端点 0 中断被接收时，应该读取 USB\_EP0CSR 寄存器来检查端点状态。

USB\_EP0CSR.OutPktRdy 位应该被设置，表示已经接收到一个数据包。然后应该读取 USB\_EP0CSR.COUNT0 寄存器，以确定这个数据包的大小。数据包可以从端点 0 FIFO 读取。

如果与请求相关联的数据长度大于端点 0 的最大数据包大小，则将发送进一步的数据包。在这种情况下，应设置 USB\_EP0CSR.ServicedOutPktRdy 位，但不应该设置 USB\_EP0CSR.DataEnd 位。

当接收到所有预期的数据包时，应该设置 USB\_EP0CSR.ServicedOutPktRdy 位和 USB\_EP0CSR.DataEnd 位（表示不需要更多的数据）。

当主机移动到请求的状态阶段时，USB 设备将生成另一个端点 0 中断，以表明请求已经完成。应用程序不需要进一步的操作，中断只是请求成功完成的确认。

如果该命令是一个无法识别的命令，或者由于某些其他原因不能执行，那么当它被解码后，应用程序应该设置 USB\_EP0CSR.ServicedOutPktRdy 位和 USB\_EP0CSR.SendStall 位。当主机发送更多的数据时，USB 设备将发送一个 STALL 握手包来告诉主机这个请求没有被执行。USB 设备将生成一个端点 0 中断，并且 USB\_EP0CSR.SentStall 位将会被设置。

如果主机在设置 USB\_EP0CSR.DataEnd 后发送了更多的数据，那么 USB 设备将发送一个 STALL 握手包。一个端点 0 中断将会生成，并将 USB\_EP0CSR.SentStall 位置位。

#### 34.4.7.3. 读请求

在 8 字节的命令之后，读请求有一个（或多个）数据包从设备发送到主机。标准设备请求的例子有：GET\_CONFIGURATION, GET\_INTERFACE, GET\_DESCRIPTOR, GET\_STATUS, SYNCH\_FRAME（参考《USB 总线规范》）。

与所有请求一样，当应用程序接收到一个端点 0 中断时，事件序列将开始。

USB\_EP0CSR.OutPktRdy 位也将被置位。8 字节的命令应该从端点 0 FIFO 读取并解码。然后应该将 USB\_EP0CSR 寄存器中的 ServicedOutPktRdy 位置位（表示命令已经从 FIFO 读取）。

发送给主机的数据应该被写入到端点 0 的 FIFO。如果要发送的数据大于端点 0 的最大包大小，则应该只将最大包大小写入 FIFO。然后寄存器 USB\_EP0CSR 中的 InPktRdy 位被置位（表示在 FIFO 中有

一个数据包要发送)。当数据包被发送到主机, 另一个端点 0 中断将产生, 下一个数据包可以写入 FIFO。

当最后一个数据包被写入 FIFO 时, USB\_EP0CSR 寄存器中的 InPktRdy 位和 DataEnd 位应该被置位 (表示在这个数据包之后没有更多的数据)。

当主机移动到请求的状态阶段时, USB 设备将生成另一个端点 0 中断, 以表明请求已经完成。应用程序不需要进一步的操作;中断只是请求成功完成的确认。

如果该命令是一个无法识别的命令, 或者由于某些其他原因不能执行, 那么当它被解码后, 应该设置 USB\_EP0CSR.ServicedOutPktRdy 位和 USB\_EP0CSR.SendStall 位设置。当主机请求数据时, USB 设备将发送一个 STALL 握手包来告诉主机这个请求没有被执行。然后一个端点 0 中断将会产生, 并 USB\_EP0CSR.SentStall 位将被置位。

如果主机在设置 USB\_EP0CSR.DataEnd 之后请求更多的数据, 那么 USB 设备将发送一个 STALL 握手包。一个端点 0 中断将会产生, 并且 USB\_EP0CSR.SentStall 位将会被置位。

#### 34.4.7.4. 错误处理

USB 设备在以下情况下会自动检测协议错误, 并向主机发送 STALL 握手包。

- 主机在写请求的 OUT DATA 阶段发送的数据比命令中指定的要多。当设置了 USB\_EP0CSR.DataEnd 位后主机发送 OUT 令牌时, 将检测到此条件。
- 主机在读请求的 IN DATA 阶段请求比命令中指定的数据更多的数据。当在设置了 USB\_EP0CSR.DataEnd 位后发送 IN 令牌时, 将检测到此条件。
- 主机发送的 OUT 数据包中大于命令中指定的数据字节。
- 主机在读请求的状态阶段发送一个非零长度的 DATA1 数据包。

当 USB 设备发送了 STALL 握手包后, 它设置 USB\_EP0CSR.SentStall 位并产生一个中断。当应用程序接收到设置了 USB\_EP0CSR.SentStall 位的端点 0 中断时, 应该中止当前的传输, 清除 USB\_EP0CSR.SentStall 位, 并返回到初始状态。

如果主机在所有数据传输结束之前进入状态阶段, 或者在完成当前传输完成前发送一个新的 SETUP 令牌包, 从而提前结束传输, 那么 USB\_EP0CSR.SetupEnd 位将被设置, 并生成一个端点 0 中断。当应用程序接收到一个设置了 USB\_EP0CSR.SetupEnd 位的端点 0 中断时, 它应该中止当前传输, 设置 USB\_EP0CSR.ServicedSetupEnd 位, 并返回到初始状态。如果 USB\_EP0CSR.OutPktRdy 位被设置, 这表明主机已经发送了另一个 SETUP 令牌包, 然后应用程序应该处理这个命令。

如果应用程序希望终止当前传输, 因为它不能处理该命令或有其他内部错误, 那么它应该设置 USB\_EP0CSR.SendStall 位。然后 USB 设备将发送一个 STALL 握手包给主机, 设置 USB\_EP0CSR.SentStall 位并生成一个端点 0 中断。

#### 34.4.8. 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式: 同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”, USB 主机则会为每个帧分配固定的带宽, 并且保证每个帧正好传送一个 IN 事务或者 OUT 事务 (由端点传输方向确定分组类型)。为了满足带宽要求, 同步传输中没有出错重传; 这也就

意味着，同步传输在发送或接收数据包之后，无握手协议，即不会发送 ACK 握手包。同样，同步传输只传送 PID 为 DATA0 的数据包，而不会用到数据翻转机制。

#### 34.4.8.1. 同步读传输

同步读传输用于从 USB 模块向主机传输周期性数据。三个可选特性可用于同步 IN 端点：

##### ■ 双包缓存功能

当写入 USB\_INEPxCSR.InMaxP 寄存器的值小于或等于分配给端点的 FIFO 大小的一半时，自动启用双包缓存。当启用时，最多可以存储两个数据包在 FIFO 等待传输到主机。

##### ■ DMA 功能

如果端点启用了 DMA，那么只要端点能够在其 FIFO 中接收另一个数据包，就会生成一个 DMA 请求。这个特性允许在没有处理器干预的情况下将数据包加载到 FIFO 中。然而，这个特性对于同步端点并不是特别有用，因为传输的数据包通常不是最大包大小，并且需要在每个数据包之后访问 USB\_INEPxCSR 寄存器来检查 Underrun 错误。

##### ■ AutoSet 功能

当 AutoSet 功能被启用时，USB\_INEPxCSR.InPktRdy 位将在一个 USB\_INEPxCSR.InMaxP 字节的数据包加载到 FIFO 时被自动设置。然而，这个特性对于同步端点并不是特别有用，因为传输的数据包通常不是最大包大小，并且需要在每个数据包之后访问 USB\_INEPxCSR 寄存器来检查 Underrun 错误。

在使用一个同步 IN 端点之前，USB\_INEPxCSR.InMaxP 寄存器必须被写入该端点的最大包大小（以字节为单位）。这个值应该与端点的标准端点描述符的字段相同。此外，USB\_INTRE 寄存器中相关的中断使能位应该设置为 1，并且 USB\_INEPxCSR 寄存器中部分位应该设置为如下所示：

表 34-1 USB\_INEPxCSR 寄存器

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	1	1	0/1	0

一个同步端点不支持数据重传，所以如果要避免数据出错，那么发送给主机的数据必须在接收到 IN 令牌之前加载到 FIFO 中。主机将每帧发送一个 IN 令牌，但帧内的时间是可以变化的。如果一个 IN 令牌在一个帧的末尾被接收，然后在下一个帧的开始进行传输，那么这将有很少的时间来重新加载 FIFO。由于这个原因，在同步 IN 端点中通常是需要同步缓存的。

AutoSet 功能可用于同步 IN 端点，但是除非来自源的数据以绝对一致的速率到达，并与主机的帧时钟同步，否则发送给主机的数据包的大小将不得不逐帧增加或减少，以匹配源数据速率。这意味着实际的包大小并不总是 USB\_INEPxCSR.InMaxP 大小，这使得 AutoSet 功能无用。

当一个数据包被发送到主机时就会产生一个中断，应用程序可以使用这个中断来将下一个数据包加载到 FIFO 中，并在 USB\_INEPxCSR 寄存器中设置 InPktRdy 位。由于中断几乎可以在一个帧内的任何时间发生，这取决于主机何时安排例程，可能会导致 FIFO 加载请求的不规则计时。如果端点的数据源来自一些外部硬件，在加载 FIFO 之前等待每一帧结束可能会更方便，因为这将最小化对额外缓冲的需求。上述操作可以通过使用 SOF 中断或来自 USB 设备的 SOF\_PULSE 信号来触发下一个数据包的加载来实现。当收到一个 SOF 包时，USB 设备就会生成一个 SOF\_PULSE 信号（USB 还保留了一

个外部帧计数器，因此当 SOF 包丢失时，它仍然可以生成 SOF\_PULSE) 并且生成一个 SOF 中断。该中断/信号可以用来设置 USB\_INEPxCSR 中的 InPktRdy 位，并检查数据溢出/缺失。

如果端点在收到 IN 令牌包时 FIFO 中没有数据，它将发送一个空数据包给主机，并在 USB\_INEPxCSR 寄存器中设置 UnderRun 位。这表明该应用程序为主机提供的数据不够快。由应用程序决定如何处理这个错误情况。

如果应用程序在每帧加载一个数据包时发现当它想要加载下一个数据包时在 USB\_INEPxCSR 寄存器中的 InPktRdy 位被设置了,这表明数据包尚未发送（也许因为一个从主机来的 IN 令牌包损坏了）。应用程序可以选择通过在 USB\_INEPxCSR 寄存器中设置 FlushFIFO 位来刷新未发送的数据包，或者可以选择跳过当前的数据包。

#### 34.4.8.2. 同步写传输

同步 OUT 端点用于从功能控制器向主机传输周期性数据。三个可选特性可用于同步 OUT 端点：

##### 1、双包缓存功能

当写入 USB\_OUTEPxCSR.OutMaxP 寄存器的值小于或等于分配给端点的 FIFO 大小的一半时，自动启用双包缓冲。当启用时，最多可以存储两个数据包在 FIFO 中。

##### 2、DMA功能

如果端点启用了 DMA，那么只要端点的 FIFO 中有一个包，就会生成一个 DMA 请求。这个特性允许在不需处理器干预的情况下从 FIFO 读取数据。然而，这个特性对于同步端点并不是特别有用，因为传输的数据包通常不是最大包大小，并且需要在每个数据包之后访问 USB\_OUTEPxCSR 寄存器，以检查溢出或 CRC 错误。

##### 3、AutoClear功能

当 AutoClear 功能启用时，USB\_OUTEPxCSR.OutMaxP 字节的数据从 FIFO 读取时，USB\_OUTEPxCSR.OutPktRdy 位将被自动清除。然而，这个特性对于同步端点并不是特别有用，因为传输的数据包通常不是最大包大小，并且需要在每个数据包之后访问 USB\_OUTEPxCSR 寄存器，以检查溢出或 CRC 错误。

在使用同步 OUT 端点之前，必须用端点的最大包大小（以字节为单位）来写入 USB\_OUTEPxCSR.OutMaxP 寄存器。这个值应该与端点的标准端点描述符的字段相同。此外，USB\_INTRE 寄存器中相关的中断启用位应该设置为 1，USB\_OUTEPxCSR 寄存器中部分位应该设置为如下所示。

表 34-2 USB\_OUTEPxCSR 寄存器

AutoClear	ISO	DMAEnab
0/1	1	0/1

一个同步端点不支持数据重传，因此，如果要避免数据溢出，必须在 FIFO 中有空间时来接收数据包。主机将每帧发送一个数据包，但是帧内的时间时可以变化的。如果一个数据包在一帧结束时收到，而另一个数据包在下一帧开始时到达，那么就没有多少时间来读取 FIFO 中的数据。因此，对于同步 OUT 端点，通常需要双包缓存功能。

AutoClear 特性可以与同步 OUT 端点一起使用。然而，除非数据接收器以绝对一致的速率接收数据，并与主机的帧时钟同步，否则主机发送的数据包大小将不得不逐帧增加或减少，以匹配所需的数据速

率。这意味着实际的数据包大小并不总是 USB\_OUTEPxCSR.OutMaxP 大小，这使得 AutoClear 功能无用。

当从主机接收到一个数据包时，就会产生一个中断，应用程序可以使用这个中断从 FIFO 读走数据，并清除 USB\_OUTEPxCSR 寄存器中的 OutPktRdy 位。由于中断几乎可能在一个帧内的任何时间发生，这取决于主机何时安排了事务，FIFO 读取数据请求的时间可能是不规则的。如果端点的数据接收器要连接到某个外部硬件，那么最好在读取 FIFO 之前等待每一帧结束，从而减少对额外缓冲的需求。这可以通过使用 SOF 中断或来自 USB 设备的 SOF\_PULSE 信号来触发数据包的读取来完成。当收到一个 SOF 包时，USB 设备生成一次 SOF\_PULSE 信号（USB 设备还保留了一个外部帧计数器，因此当 SOF 包丢失时，它仍然可以生成 SOF\_PULSE）并且生成一个 SOF 中断。该中断/信号可以用来清除 USB\_OUTEPxCSR 中的 OutPktRdy 位，并检查数据溢出/缺失。

如果 FIFO 中没有空间来存储从主机接收到的数据包，USB\_OUTEPxCSR 寄存器中的 OverRun 位将被设置。这表明应用程序读取数据的速度不够快。由应用程序决定如何处理这个错误条件。

如果 USB 设备发现一个接收到的包有 CRC 错误，它仍然将该包存储在 FIFO 中，并设置 USB\_OUTEPxCSR.OutPktRdy 位和 USB\_OUTEPxCSR.DataError 位。如何处理这个错误条件由应用程序决定。

### 34.4.9. 批量传输

#### 34.4.9.1. 批量读传输

批量 IN 端点用于将不定期数据从微控制器传输到主机。有三个可选特性可用于批量 IN 端点：

- 双包缓存功能

如果写入 USB\_INEPxCSR.InMaxP 寄存器的值小于或等于分配给端点的 FIFO 大小的一半，则将自动启用双包缓存功能。当启用时，最多可以存储两个数据包存储在 FIFO 中待传输给主机。

- DMA功能

如果端点启用了 DMA，那么只要端点能够在其 FIFO 中接受另一个包，就会生成一个 DMA 请求。这个特性可以用来在没有处理器干预的情况下将数据包加载到 FIFO 中。

启动 DMA 流程如下：

- 1、在DMA控制寄存器上将存储器地址配置成DMA传输的源地址，将USB的FIFO地址配置为外设地址，传输方向为从存储器读；
- 2、在DMA控制寄存器中配置要传输的总的字节数；
- 3、在DMA寄存器上配置通道优先级；
- 4、根据应用程序的要求，配置在传输完成一半还是全部完成时产生DMA中断；
- 5、在DMA寄存器上激活该通道；
- 6、使能USB\_INEPxCSR.DMAEnab位；
- 7、在DMA往FIFO中搬完数据后应用程序置位USB\_INEPxCSR.InPktRdy位，随即USB设备开始向USB主机传输数据。

注意：推荐 DMA 功能与下文描述的 AutoSet 功能一起使用，这样除了最后一个数据包外均不需要应用程序来置位 USB\_INEPxCSR.InPktRdy 位。

## ■ AutoSet功能

当 AutoSet 功能被启用时，USB\_INEPxCSR.InPktRdy 位将在一个 USB\_INEPxCSR.InMaxP 字节的数据包加载到 FIFO 时被自动设置。这在使用 DMA 加载 FIFO 时特别有用，因为它避免了在大容量传输期间加载单个数据包时需要任何处理器干预。

在使用批量 IN 端点之前，必须将该端点在 wMaxPacketSize（参考《USB 总线规范》）中规定的最大包大小（以字节为单位）的数值写入 USB\_INEPxCSR.InMaxP 寄存器中。另外，USB\_INTRE 寄存器中相关的中断使能位应该设置为 1，并且 USB\_INEPxCSR 寄存器应该设置为如下所示。

表 34-3 USB\_INEPxCSR 寄存器

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	0	1	0/1	0

当第一次配置批量 IN 端点时，在端点 0 上执行 SET\_CONFIGURATION 或 SET\_INTERFACE 命令后，应该写入 USB\_INEPxCSR 寄存器来设置 ClrDataTog 位。这将确保数据切换以正确的状态启动。此外，如果 FIFO 中有任何数据包（由设置的 USB\_INEPxCSR.FIFONotEmpty 位表示），它们应该通过设置 USB\_INEPxCSR.FlushFIFO 位来刷新。

当数据要通过批量 IN 端点传输时，需要将数据包加载到 FIFO 中并置位 USB\_INEPxCSR.InPktRdy 位。当数据发送完毕后，USB\_INEPxCSR.InPktRdy 位被清除，并产生一个中断，以便下一个数据包可以加载到 FIFO 中。如果双包缓存被启用，那么当第一个数据包被加载并且 USB\_INEPxCSR.InPktRdy 位被设置后，USB\_INEPxCSR.InPktRdy 位将立即被清除，并产生一个中断，以便第二个数据包可以被加载到 FIFO 中。应用程序应该以同样的方式运行，当它接收到一个中断时加载一个数据包，不管是否启用双包缓存。

数据包的大小不能超过 USB\_INEPxCSR.InMaxP 寄存器中指定的大小。当要传输一个大于 USB\_INEPxCSR.InMaxP 的数据块时，必须将其作为多个数据包发送。这些数据包的大小除了最后一个数据包外应该都是 USB\_INEPxCSR.InMaxP 寄存器中指定的大小。主机可以通过知道预期的数据总量来确定传输的所有数据已经发送。或者，当它收到一个小于 USB\_INEPxCSR.InMaxP 大小的包时，它可以推断出所有的数据已经发送。在后一种情况下，如果数据块的总大小是 USB\_INEPxCSR.InMaxP 寄存器中定义的整倍数，那么就需要在所有数据发送完毕后发送一个空数据包。这是通过设置 USB\_INEPxCSR.InPktRdy，当接收到下一个中断，不加载任何数据到 FIFO 来完成的。

如果正在传输大批量数据，那么可以通过使用 DMA 来避免调用中断服务程序来加载每个数据包。

如果应用程序想要关闭批量 IN 端点，它应该设置 USB\_INEPxCSR.SendStall 位。当 USB 设备接收到下一个 IN 令牌时，它将发送一个 STALL 握手包给主机，USB\_INEPxCSR.SentStall 位被置位并产生一个中断。

当应用程序接收到一个设置了 USB\_INEPxCSR.SentStall 位的中断时，应该清除 USB\_INEPxCSR.SentStall 位。当然，它也应该保留 USB\_INEPxCSR.SentStall 位的设置，直到准备好重新启用批量 IN 传输。

**注意：**如果主机由于某种原因无法接收到 STALL 握手包，它将发送另一个 IN 令牌包，所以建议保持 USB\_INEPxCSR.SentStall 位的设置，直到应用程序准备好重新启用批量 IN 传输。当批量 IN 传输被重新启用时，应该通过设置 USB\_INEPxCSR 寄存器中 ClrDataTog 位来重启数据切换序列。

### 34.4.9.2. 批量写传输

批量 OUT 端点用于将不定期数据从主机传输到 USB 模块。有三个可选特性可用于批量 OUT 端点：

#### 1) 双包缓存功能

如果写入 USB\_OUTEPxCSR.OutMaxP 寄存器的值小于或等于分配给端点的 FIFO 大小的一半，则将自动启用双包缓冲。启用时，FIFO 中最多可以存储两个数据包。

#### 2) DMA功能

如果端点启用了 DMA，那么只要端点的 FIFO 中有一个数据包，就会生成一个 DMA 请求。这个特性可以允许在不需要处理器干预的情况下从 FIFO 读取数据包。

启动 DMA 流程如下：

1、在DMA控制寄存器上将存储器地址配置成DMA传输的源地址，将USB的FIFO地址配置为外设地址，传输方向为从外设读；

2、在DMA控制寄存器中配置要传输的总的字节数；

3、在DMA寄存器上配置通道优先级；

4、根据应用程序的要求，配置在传输完成一半还是全部完成时产生DMA中断；

5、在DMA寄存器上激活该通道；

6、使能USB\_OUTEPxCSR.DMAEnab位；

7、在DMA从FIFO中搬完数据后，应用程序清除USB\_OUTEPxCSR.OutPktRdy位。

注意：推荐 DMA 功能与下文描述的 AutoClear 功能一起使用，这样除了最后一个数据包外均不需要应用程序来清除 USB\_OUTEPxCSR.OutPktRdy 位。

#### 3) AutoClear功能

当 AutoClear 功能启用时，USB\_OUTEPxCSR.OutMaxP 字节的数据包从 FIFO 卸载时，USB\_OUTEPxCSR.OutPktRdy 位将被自动清除。当使用 DMA 读取 FIFO 中的数据时，这特别有用。因为它避免了在大型批量传输期间读取单个数据包时需要的处理器的干预。

在使用批量 OUT 端点之前，必须将该端点在 wMaxPacketSize（参考《USB 总线规范》）中规定的最大包大小（以字节为单位）的数值写入 USB\_OUTEPxCSR.OutMaxP 寄存器中。此外，USB\_INTRE 寄存器中相关的中断启用位应该设置为 1（如果这个端点需要中断），USB\_OUTEPxCSR 寄存器中的相关位应该设置为如下所示。

表 34-4 USB\_OUTEPxCSR 寄存器

AutoClear	ISO	DMAEnab
0/1	0	0/1

当第一次配置批量 OUT 端点时，在端点 0 上的 SET\_CONFIGURATION 或 SET\_INTERFACE 命令之后，应该置位 USB\_OUTEPxCSR.ClrDataTog 位。这将确保数据切换以正确的状态启动。同样，如果 FIFO 中有任何数据包（OutPktRdy 位被设置），它们应该通过设置 USB\_OUTEPxCSR.FlushFIFO 位被刷新。

当一个数据包被批量 OUT 端点接收时，USB\_OUTEPxCSR.OutPktRdy 位被设置并产生一个中断。应用程序应该读取端点的 USB\_OUTCOUNT.OUTCOUNT 寄存器，以确定数据包的大小。然后从 FIFO 读取数据包，清除 USB\_OUTEPxCSR.OutPktRdy 位。

数据包大小不应该超过 USB\_OUTEPxCSR.OutMaxP 寄存器中指定的大小。当一个大于 USB\_OUTEPxCSR.OutMaxP 的数据块要发送给 USB 设备时，它将作为多个数据包发送。除了最后一个数据包，其他的所有数据包的大小都是 USB\_OUTEPxCSR.OutMaxP 寄存器中指定的，最后一个数据包将包含剩余部分。应用程序可以使用特定的方法来确定传输的总大小，从而确定何时接收到最后一个包。或者，当它收到一个小于 USB\_OUTEPxCSR.OutMaxP 寄存器指定大小的数据包时，就可能推断整个数据已经收到（如果数据块的总大小是 USB\_OUTEPxCSR.OutMaxP 的整数倍数，则在数据结束后将发送一个空数据包，表示传输完成）。

如果正在传输大批量数据，通过使用 DMA 可以避免调用中断服务程序来读取每个数据包的数据。

如果应用程序想要关闭使用批量 OUT 端点，它应该设置 USB\_OUTEPxCSR.SendStall 位。当 USB 设备接收到下一个数据包时，USB 设备将发送一个 STALL 握手包给主机，设置 USB\_OUTEPxCSR.SentStall 位并产生一个中断。

当应用程序接收到一个设置了 USB\_OUTEPxCSR.SentStall 位的中断时，应该清除 USB\_OUTEPxCSR.SentStall 位。应用程序也应该保留 USB\_OUTEPxCSR.SendStall 位的设置，直到它准备好重新启用批量 OUT 端点。

注意：如果主机由于某种原因未能接收到 STALL 握手包，它将发送另一个包，因此建议保留 USB\_OUTEPxCSR.SendStall 位，直到应用程序准备好重新启用批量 OUT 传输。当重新启用批量 OUT 传输时，应该设置 USB\_OUTEPxCSR.ClrDataTog 位来重启数据切换序列。

## 34.4.10. 中断传输

### 34.4.10.1. 中断读传输

中断 IN 端点用于从功能控制器向主机传输周期性数据。

中断 IN 端点使用与批量 IN 端点相同的协议，并且可以以相同的方式使用。尽管可以使用 DMA，但它提供的好处很少，因为中断端点通常期望在一个数据包中传输所有的数据。

中断 IN 端点还支持批量 IN 端点不支持的一个特性，即支持数据切换位的连续切换。这个特性是通过在 USB\_INEPxCSR 寄存器中设置 FrcDataTog 位来启用的。当这个位设置为 1 时，USB 设备将认为数据包已经成功发送，并且为端点切换数据位，无论是否从主机收到了 ACK。

### 34.4.10.2. 中断写传输

中断 OUT 端点用于从主机向功能控制器传输周期性数据。

中断 OUT 端点使用与批量 OUT 端点几乎相同的协议，并且可以以相同的方式使用。尽管 DMA 可以与一个中断 OUT 端点一起使用，但它通常提供的好处很少，因为中断端点通常期望在一个数据包中传输所有的数据。

## 34.5. USB 寄存器

注意：在对 USB 寄存器进行写操作时只能按照字节 (byte) 或者字 (word) 操作，在对 USB 寄存器进行读操作时只能按照字节 (byte) 操作。

### 34.5.1. USB 控制寄存器 (USB\_CR)



Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	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
ISO_ Update	Res			Re- set	Re- sume	Sus- pend_ Mode	Enable_ Sus- pend	Up- date	ADD						
RW	-			R	RW	R	RW	R	RW						

Bit	Name	R/W	Reset Value	Function
31: 16	保留	-	0	保留
15	ISO_Update	RW	0	ISO_Update: 该位置1后, USB 控制器在发送数据包前, 需要在 InPktRdy 置1后, 等待一个 SOF; 如果在收到 SOF 前接收到 IN Token, USB 控制器会发送一个零包。(这个寄存器只有在 ISO 传输时使用)
14: 12	保留	-	-	保留
11	Reset	R	0	Reset: USB 总线有复位信号时, 该 Bit 置1
10	Resume	RW	0	Resume: 当 USB 设备处于 Suspend 模式下, 软件置1, 产生 Resume 唤醒信号; 软件在10mS 后清除该位。(最大15mS)
9	Suspend_Mode	R	0	Suspend_Mode: 当进入 Suspend 模式后, 由 USB 设备硬件置1; 软件读取中断寄存器, 或者软件写 Resume 寄存器时, 该位清零
8	Enable_Suspend	RW	0	Enable_Suspend: Suspend 功能使能
7	Update	R	0	Update: 当写入 ADD 位时置1; 地址生效 (在传输结束) 后清零;
6: 0	ADD	RW	0	ADD: Function 地址

### 34.5.2. USB 中断状态寄存器 (USB\_INTR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0
-										R	R	R	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		EP5 OUT	EP4 OUT	EP3 OUT	EP2 OUT	EP1 OUT	Res					SOF	Reset	Re- sume	Sus- pend
-		R	R	R	R	R	-					R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 22	保留	-	-	保留
21	EP1IN	R	0	IN 端点1 中断
20	EP4IN	R	0	IN 端点4 中断
19	EP3IN	R	0	IN 端点3 中断
18	EP2IN	R	0	IN 端点2 中断
17	EP5IN	R	0	IN 端点 5中断
16	EP0	R	0	端点0 中断
15: 14	保留	-	-	保留
13	EP1OUT	R	0	OUT 端点1 中断
12	EP4OUT	R	0	OUT 端点4 中断
11	EP3OUT	R	0	OUT 端点3 中断
10	EP2OUT	R	0	OUT 端点2 中断
9	EP5OUT	R	0	OUT 端点5 中断
8: 4	保留	-	-	保留
3	SOF	R	0	SOF 中断, 每帧开始置1
2	Reset	R	0	Reset 中断, 在 USB 总线上检测到复位信号时置1
1	Resume	R	0	Resume 中断, USB 设备在 Suspend 模式时, 在 USB 总线上检测到 Resume 信号时置1
0	Suspend	R	0	Suspend 中断, 在 USB 总线上检测到 Suspend 信号时置1。

### 34.5.3. USB 中断使能寄存器 (USB\_INTRE)

Address offset: 0x08

Reset value: 0x003F 3E06

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0E

										RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		EP5 OUTE	EP4 OUTE	EP3 OUTE	EP2 OUTE	EP1 OUTE	Res					SOFE	ResetE	ResumeE	Sus- pendE
-		RW	RW	RW	RW	RW	-					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 22	保留	-	-	保留
21	EP1INE	RW	1	IN 端点1 中断使能
20	EP4INE	RW	1	IN 端点4 中断使能
19	EP3INE	RW	1	IN 端点3 中断使能
18	EP2INE	RW	1	IN 端点2 中断使能
17	EP5INE	RW	1	IN 端点5 中断使能
16	EP0E	RW	1	端点0 中断使能
15: 14	保留	-	-	保留
13	EP1OUTE	RW	1	OUT 端点1 中断使能
12	EP4OUTE	RW	1	OUT 端点4 中断使能
11	EP3OUTE	RW	1	OUT 端点3 中断使能
10	EP2OUTE	RW	1	OUT 端点2 中断使能
9	EP5OUTE	RW	1	OUT 端点5 中断使能
8: 4	保留	-	-	保留
3	SOFE	RW	0	SOF 中断使能
2	ResetE	RW	1	Reset 中断使能
1	ResumeE	RW	1	Resume 中断使能
0	SuspendE	RW	0	Suspend 中断使能

#### 34.5.4. USB 帧寄存器 (USB\_FRAME)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 20	保留	-	-	保留
19: 16	INDEX	RW	0	端点选择

15: 11	保留	-	0	保留
10: 0	FRAMENUM	R	0	最后接收到的帧号

### 34.5.5. USB 端点 0 控制寄存器 (USB\_EP0CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	COUNT0							Serviced-SetupEnd	ServicedOut-PktRdy	Send-Stall	Set-upEnd	DataEnd	Sent-Stall	InPktRdy	Out-PktRdy
-	R							W	W	W	R	W	RW0	RS	R

Bit	Name	R/W	Reset Value	Function
31: 15	保留	-	-	保留
14: 8	COUNT0	R	0	EP0接收到的数据长度, OutPktRdy 置1时读取有效
7	ServicedSetupEnd	W	0	软件写1清零 SetupEnd, 该位自动清零
6	ServicedOutPktRdy	W	0	软件写1清零 OutPktRdy, 该位自动清零
5	SendStall	W	0	软件写1终止当前传输; STALL 握手会被发送, 之后该位自动清零;
4	SetupEnd	R	0	该位在控制传输结束, DataEnd 置1前置1, 会产生中断并清除 FIFO;
3	DataEnd	W	0	软件在以下情况写1, 该位自动清零; 1, 发送最后一个数据包, 置位 InPktRdy 后; 2, 最有一个数据包被读出, 且 OutPktRdy 被软件清零后; 3, 发送零包, 置位 InPktRdy 后;
2	SentStall	RC_W0	0	发送 STALL 握手后置1, 由软件清零
1	InPktRdy	RS	0	软件写1当把一个数据包写入 FIFO 后。数据包传输完成后清零, 清零后产生中断。
0	OutPktRdy	R	0	该位在接收到一个数据包后置1, 并产生中断

### 34.5.6. USB IN 端点控制寄存器 (USB\_INEPxCSR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Res								InMaxP							
-								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	ClrData Tog	Sent Stall	Send Stall	Flush FIFO	Un- der- Run	FIFONotE mpty	InPkt- Rdy	Au- toSet	ISO	Mod e	DMAE- nab	FrcD ata- Tog	Res		
-	W	RC_ W0	RW	W	RC_W 0	RC_W0	RS	RW	RW	RW	RW	RW	-		

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23: 16	InMaxP	RW	0	IN 端点最大的包长，以8 Bytes 为单位；每个 IN 端点都有一个，EP0除外
15	保留	-	-	保留
14	ClrDataTog	W	0	软件写1，复位 IN EP data toggle 为0
13	SentStall	RC_W0	0	发送 STALL 握手包后置1；此时 FIFO 要清除，InPktRdy 也要清零，该位由软件清零
12	SendStall	RW	0	软件写1，收到 IN Token 后，发送一个 STALL 握手； 软件清零以终止 Stall 发送，该位对 ISO 无效。
11	FlushFIFO	W	0	软件写1清除 IN FIFO；只能清除下一个待传输的包，如果 FIFO 中有两包数据，需要写两次，该位自动清零。
10	UnderRun	RC_W0	0	在 ISO 模式，当收到 IN Token 后，发送零包并且 InPktRdy 没有置1时，该位置1；在 Bulk 和 Int 模式，当 USB 设备收到 IN Token 后回复了一个 NAK 时该位置1,该位由软件清零。
9	FIFONotEmpty	RC_W0	0	IN FIFO 非空标志
8	InPktRdy	RS	0	当把一个数据包写入 FIFO 后，软件写1；数据包传输完成后清零，清零后产生中断
7	AutoSet	RW	0	置1后，当 IN FIFO 中写入了数据达到最大包 (InMaxP)，InPktRdy 自动置1
6	ISO	RW	0	置1使能 ISO 传输，清零是 Bulk 或者 Interrup 传输
5	Mode	RW	0	1: IN 端点； 0: OUT 端点；
4	DMAEnab	RW	0	In 端点 DMA 请求使能
3	FrcDataTog	RW	0	置1后，无论是否收到 ACK，强制翻转 data toggle 信号并清除 FIFO 中的数据包

2: 0	保留	-	-	保留
------	----	---	---	----

### 34.5.7. USB OUT 端点控制寄存器 (USB\_OUTEPxCSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								OutMaxP							
-								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ClrData- Tog	Sent- Stall	Send- Stall	Flush- FIFO	Da- taError	Over- Run	FIFOFull	Out- PktRdy	Auto- Clear	ISO	DMA- Mode	Res				
W	RC_W0	RW	W	R	RC_W0	R	RC_W0	RW	RW	RW	RW	-			

Bit	Name	R/W	Reset Value	Function
31: 24	保留	-	-	保留
23: 16	OutMaxP	RW	0	OUT 端点最大的包，每个 OUT 端点都有一个，EP0除外；
15	ClrDataTog	W	0	软件写1复位 EP 的 data toggle 至0；
14	SentStall	RC_W0	0	发送 STALL 握手结束后置1；该位软件清零；
13	SendStall	RW	0	软件写1发送 STALL 握手，软件清零结束 STALL； 在 ISO 模式下无效；
12	FlushFIFO	W	0	清除 OUT FIFO，写一次清除一包的数据；
11	DataError	R	0	OutPktRdy 置1后，数据包有 CRC 错误或者 bit-stuff 错误；OutPktRdy 清零后自动清零；只有在 ISO 模式下有效；
10	OverRun	RC_W0	0	OUT 包不能再写入 OUT FIFO，软件清零； 只有在 ISO 模式下有效；
9	FIFOFull	R	0	OUT FIFO 满标志
8	OutPktRdy	RC_W0	0	接收到数据包后置1；数据从 FIFO 读出后，软件清零，会产生中断；
7	AutoClear	RW	0	置1后，当从 OUT FIFO 读出的数据到达 OutMaxP 的值后，OutPktRdy 自动清零
6	ISO	RW	0	1: ISO 模式； 0: Bulk or Interrupt 模式；

Bit	Name	R/W	Reset Value	Function
5	DMAEnab	RW	0	DMA 使能
4	DMAMode	RW	0	0: 所有收到的包都产生 DMA 请求, 并产生中断; 1: 接收到 OutMaxP 的数据后产生 DMA 请求, 没有中断; 其他包大小的数据, 产生中断, 但不产生 DMA 请求;
3: 0	保留	-	-	保留

### 34.5.8. USB OUT 端点计数寄存器 (USB\_OUTCOUNT)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 11	保留	-		保留
10: 0	OUTCOUNT	R	0	收到的数据长度, OutPktRdy 置1时读取有效

### 34.5.9. USB FIFO 寄存器 (USB\_FIFO)

Address offset: 0x20-0x33

Reset value: 0x0000 0000

注意: FIFO 只能按 word 或者 byte 操作

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

Bit	Name	R/W	Reset Value	Function
31: 0	FIFODATA	-	-	FIFODATA (0~5,每个 EP 占用4个 Byte 地址)

## 35. 调试支持

### 35.1. 概况

本芯片基于 Cortex-M0+ CPU，该 CPU Core 包含高级 debug 硬件扩展功能。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

调试功能在由调试主机在连接和调试 MCU 时使用，调试的接口是 serial wire。在 M0+ CPU Core 中的调试功能是一套 ARM CoreSight Design kit。

M0+提供了集成的片上调试支持，由以下部分组成：

- SW-DP: serial wire
- BPU: Break point unit
- DWT: Data watchpoint trigger

调试支持也包括了本芯片的调试集成功能：

- 灵活的调试引脚分配，SWDIO@PA13、SWCLK@PA14
- MCU 调试盒（支持低功耗模式，控制外设时钟等

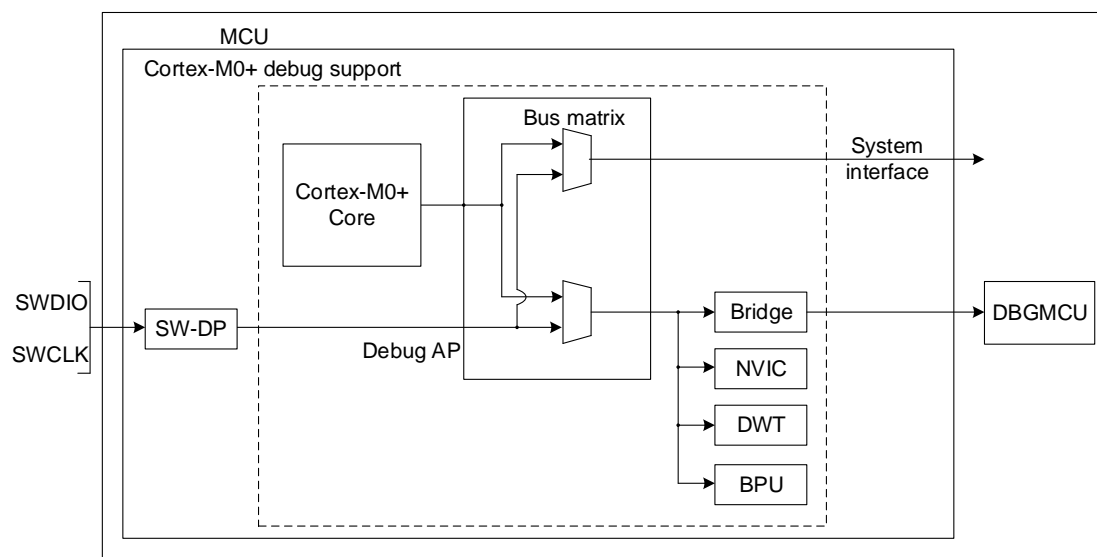


图 35-1 DBG 框图

### 35.2. 引脚分布和调试端口脚

#### 35.2.1. SWD 调试端口

调试功能相关的端口有两个，在所有封装形式都可见。



表 35-1 DBG 框图

SW-DP 端口引脚名称	SW 调试接口		引脚分配
	类型	调试功能	
SWDIO	输入/输出	串行数据输入/输出	PA13
SWCLK	输入	串行时钟	PA14

### 35.2.2. 灵活的 SW-DP 脚分配

在芯片复位后（系统复位或者上电复位），用作 SW-DP 的端口被分配作为被调试主机立刻使用的专属 pin。

然而，芯片提供了关闭 SWD 端口的方法，并释放该端口作为 GPIO 用。

### 35.2.3. SWD 脚上的内部上拉和下拉

一旦 SWD 端口被软件释放，则 GPIO 控制器控制了这两个端口。GPIO 控制寄存器的复位状态把 IO 置为同等的状态：

- SWDIO: input pull-up
- SWCLK: input pull-down

片内的上拉和下拉电阻为外围节省了增加电阻的需求。

## 35.3. ID 代码和锁定机制

芯片内存放 ID code。推荐 Keil、IAR 等工具使用该 ID Code（位于 0x4001 5800 地址）锁住调试。芯片上电后，硬件读取 Flash 的保留区的 0x1FFF 33F0 地址，装载到 DBG\_IDCODE 寄存器中。

## 35.4. SWD 调试端口

### 35.4.1. SWD 协议介绍

这是个同步的串行通讯协议，使用以下两个端口：

- SWCLK: 来自主机给芯片的 clock 信号
- SWDIO: 双向数据信号

该协议允许两个 bank 的寄存器（DPACC 寄存器和 APACC 寄存器）被读和写入。数据位是按照在线上的 LSB-first 传输。对于 SWDIO 的双向管理，线上必须在板级上拉（推荐 100k 欧的电阻）。

在协议中每次 SWDIO 方向的改变，转向时间被插入在线上既没有被主机，也没有被芯片驱动的情况。缺省状态下，这个转向时间是 1 个位的时间，然而整个可以通过配置 SWCLK 频率来调整。

### 35.4.2. SWD 协议序列

每个序列由以下阶段组成：

- 主机发送的包请求（8 bits）
- 芯片发送的应答响应（3 bits）

- 主机或者芯片的数据发送阶段 (33 bits)

表 35-2 请求包 (8 bits)

比特位	名称	描述
0	Start	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 或者 AP 寄存器的地址区域
5	Parity	以前位的校验位
6	Stop	0
7	Park	没有被主机驱动。由于上拉属性, 会被芯片读出1。

通常转向时间 (缺省为 1 bit) 跟着包请求, 此时主机和芯片都没有驱动信号线。

表 35-3 ACK 响应 (3 bits)

比特位	名称	描述
[2:0]	ACK	001: FAULT 010: WAIT 100: OK

如果一个读操作或者如果 1 个 wait 或者 FAULT 应答被接收到, 则转向时间必须跟随 ACK 响应。

表 35-4 DATA 传输 (33 bits)

比特位	名称	描述
[31:0]	WDATA 或者 RDATA	写或者读数据
32	校验位	对[31:0]的奇偶校验位

如果是读操作时, 转向时间必须跟着数据传输。

### 35.4.3. SW-DP 状态机 (reset, idle states, ID code)

SW-DP 的状态机有个定义了 SW-DP 的内部 ID 代码。它遵循 JEP-106 标准。这个 ID 代码是缺省的 ARM 代码, 并被置位 0x0BC11477 (对应 Cortex-M0+)。

### 35.4.4. DP and AP 读/写访问

- 读 DP 的操作不会被 posted: 芯片响应可以被立即 (ACK=OK), 或者可以被延迟 (ACK=WAIT)
  - 读 AP 的操作被 posted: 这意味着访问的结果被返回到下一次传输。如果下次要进行的访问不是 AP 访问, 则 DP-RDBUFF 寄存器必须被地呼出获得该结果。
- DP-CTRL/STAT 寄存器的 READOK 标志在每个 AP 读访问或者 RDBUFF 读请求 (知道是否 AP 读访问是成功的) 时被更新。

- SW-DP 实现了写 buffer (对于 DP 和 AP 写), 这甚至当其他操作仍未完成时, 接收一个写操作。如果写 buffer 满了, 芯片应答响应是“WAIT”。IDCODE 读、CTRL/STAT 读或者 ABORT 写, 是例外 (甚至当如果写 buffer 是满的)
  - 由于 SWCLK 和 HCLK 是异步时钟, 在写操作后 (校验位之后) 需要两个额外的 SWCLK 周期, 用来确保写的内部有效性。当驱动信号线为低时, 这几个周期应该被应用。
- 当为上电请求写 CTRL/STAT 时, 以上尤其重要。如果下个操作 (需要上电) 立即出现, 则会失败。

### 35.4.5. SW-DP 寄存器

当 ApnDP=0 时, 可以访问这些寄存器。

A[3: 2]	R/W	CTRLSEL 位或者 SELECT 寄存器	寄存器
00	Read		IDCODE
00	Write		ABORT
01	Read/Write	0	DP-CTRL/STAT
01	Read/Write	1	WIRE CONTROL
10	Read		READ RESEND
10	Write		SELECT
11	Read/Write		READ BUFFER

### 35.4.6. SW-AP 寄存器

Address	A[3: 2]	描述
0x0	00	保留
0x4	01	DP CTRL/STAT 寄存器, 用作 <ul style="list-style-type: none"> <li>■ 请求一个系统或者调试的 power-up</li> <li>■ 为 AP 访问配置传输操作</li> <li>■ 控制被 pushed 比较和被 pushed 验证操作</li> <li>■ 读一些状态标志 (溢出、power-up 应答)</li> </ul>
0x8	10	DP SELECTION 寄存器: 用作选择当前访问端口和 active 4个 word 的寄存器在窗口。 <ul style="list-style-type: none"> <li>■ Bit 31: 24: APSEL: 选择当前 AP</li> <li>■ Bit 23: 8: 保留</li> <li>■ Bit 7: 4: APBANKSEL: 在当前 AP,选择 active 4个 word 寄存器窗口</li> <li>■ Bit 3: 0: 保留</li> </ul>
0xC	11	DP RDBUFF 寄存器: 用于提供调试者在一个操作序列后, 得到最终的结果 (不用请求新的 JTAG-DP 操作)

## 35.5. 内核调试

通过 core debug 寄存器，可以访问 Core debug。Debug 访问这些寄存器是通过 debug 访问端口。他由下面四个寄存器组成

表 35-5 内核调试寄存器

寄存器	描述
DHCSR	32位 调试停机控制与状态寄存器
DCRSR	17位 调试内核寄存器选择器寄存器
DHCSR	32位 调试内核寄存器数据寄存器
DEMCR	32位 调试异常与监控控制寄存器

这些寄存器不会被系统复位。它们只能通过上电复位来复位，要在复位时停止 (Hart)，必须：

- 启用调试和异常监视控制寄存器的第 0 位 (VC\_CORRESET) 寄存器。
- 启用调试停止控制和状态寄存器的第 0 位 (C\_DEBUGEN) 寄存器

## 35.6. BPU 断点单元 (Break Point Unit)

Cortex-M0+ BPU 实现提供了 4 个断点寄存器。BPU 是一套 ARMv7-M 的 Flash 补丁和断点 (FPB) Block (Cortex-M3 & Cortex-M4) 。

### 35.6.1. BPU 功能

处理器断点实现基于 PC 的断点功能。

参考 ARMv6-M ARM 和 ARM Coresight Components Technical Reference Manual，以获得更多关于 BPU Coresight 的身份寄存器和他们的地址和访问种类。

## 35.7. 数据观察点 DWT (Data Watchpoint)

Cortex-M0 DWT 实现提供了 2 个 watchpoint 寄存器。

### 35.7.1. DWT 功能

处理器的断点实现基于 PC 的断点功能。

### 35.7.2. DWT 程序计数器样本寄存器

实现数据 watchpoint 单元的处理器，也实现了 ARMv6-M 可选的 DWT Program Counter Sample register (DWT\_PCSR)。该寄存器允许调试者周期性的采样 PC，而不用停止处理器。这个机制提供了粗粒度分析。

CORTEX-M0+ DWT\_PCSR 记录了通过了条件代码的指令和未通过的指令。

## 35.8. MCU 调试模块 (DBGMCU)

MCU debug component 帮助调试者提供以下支持：

- 低功耗模式
- 对 timer、watchdog、I2C、CAN、RTC 在 breakpoint 期间的时钟控制
- 对跟踪脚分配的控制

MCUDBG 寄存器还提供芯片 ID 编码。使用 JTAG 或者 SW 调试接口，或者用户程序都可以访问此 ID 编码。

### 35.8.1. 低功耗模式的调试支持

为进入低功耗模式，要执行 WFI 或者 WFE 指令。MCU 进入低功耗模式，或者是将 CPU Clock 停止掉，或者是减少 CPU 的功耗。

CPU 不允许在 debug 期间，停掉 FCLK 或者 HCLK。由于这些是调试者连接的需要，在一个调试期间，他们必须保持开启。MCU 集成了特殊的方法，允许用户在低功耗模式下调试软件。

因此，调试者主机必须先置某些调试配置寄存器的内容，以改变低功耗行为：

- 在 Sleep 模式：FCLK 和 HCLK 仍然有效。相应的，该模式不能引起任何对于标准调试功能的限制。
- 在 Stop 模式：DBG\_STOP 位必须被调试者提前置位。

### 35.8.2. 支持定时器、看门狗、CAN 和 I<sup>2</sup>C 的调试

在一个 breakpoint 期间，是有必要选择 timer 的计数器和 watchdog 要怎样的行为：

- 他们可以继续在 breakpoint 里计数。例如，这是当一个 PWM 正在控制电机时通常被需要的。
- 他们可以停下来在 breakpoint 内部计数。这是 watchdog 的特性决定的。

对于 CAN，用户可以选择在断点期间阻止接收寄存器的更新。

对于 I<sup>2</sup>C，用户可以选择在断点期间阻止 SMBUS 超时

## 35.9. DBG 寄存器

### 35.9.1. DBG 设备 ID 代码寄存器 (DBG\_IDCODE)

Address offset: 0x00

仅支持32-bit 地址访问，只读。

该寄存器可以通过软件 debug 端口 (2 pin) 或者用户软件访问。

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

R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Bit	Name	R/W	Reset Value	Function
31: 0	DBG_IDCODE[31: 0]	R	0x0618 8061	Revision ID

### 35.9.2. 调试 MCU 配置寄存器 (DBGMCU\_CR)

该寄存器配置在 debug 状态下的 MCU 低功耗模式。

该寄存器会被上电复位进行异步复位（不是系统复位）。它可以在系统复位下被调试者进行写操作。

如果调试者主机不支持该功能，对于软件使用者来说，写这些寄存器仍然是可能的。

**Address offset:** 0x04

**Reset value:** 0x0000 0000（不会被系统复位进行复位）

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	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_STOP	DBG_SLEEP
-														RW	RW

Bit	Name	R/W	Reset Value	Function
31: 2	保留	-		保留
1	DBG_STOP	RW	0	Debug stop（调试停止）模式。 0:（FCLK=off, HCLK=off）。在 Debug stop 模式，HCLK 和 FCLK 都会关闭。当从 Debug stop 模式退出时，时钟配置与上电复位后相同（系统时钟为 HSI）。随后，软件需要重新配置时钟控制器。 1:（FCLK=on, HCLK=on）。当进入 Debug stop 模式，HSI 不会关闭，FCLK 和 HCLK 由 HSI 产生。当退出 Debug stop 模式，如果需要改变时钟控制，软件需要重新配置。
0	DBG_SLEEP	RW	0	Debug sleep（调试睡眠）模式。 0:（FCLK 开, HCLK 关）。在 Debug sleep 模式，FCLK 由原先配置好的系统时钟提供，HCLK 关闭。由于 Debug sleep 模式不会复位已配置好的时钟系统，因此从 Debug sleep 模式退出后，软件不需要重新配置时钟。 1:（FCLK 开, HCLK 开）。在 SLEEP 模式，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

### 35.9.3. DBG APB 冻结寄存器 1 (DBG\_APB\_FZ1)

该寄存器用来配置 timer、RTC、IWDG、WWDG 在 debug 下的时钟。该寄存器被上电复位进行异步复位（不是系统复位）。它可以被调试者在系统复位下进行写。

**Address offset:** 0x08

**Power on Reset value:** 0x0000 0000

31	3	2	28	27	26	2	2	2	22	21	20	19	1	17	16	
0	9					5	4	3					8			
DBG_	Res								DBG_I2C2_SMBUS_	DBG_I2C1_SMBUS_	Res	DBG_CAN_	Res			
LPTIM_S									TIMEOUT	TIMEOUT		STOP				
TOP																
RW	-								R	RW	-	RW	-			
15	1	1	12	11	10	9	8	7	6	5	4	3	2	1	0	
	4	3														
Res	DBG_	DBG_	DBG_	Res			Res	DBG_TIM7_STOP	DBG_TIM6_	Res		DBG_	DBG_TIM2_			
	IWDG_S	WWDG_S	RTC_S						STOP			TIM3_S	STOP			
	TOP	TOP	TOP									TOP	STOP			
-	RW	RW	RW	-			-		RW	RW	-		RW	RW		

Bit	Name	R/W	Reset Value	Function
31	DBG_LPTIM_STOP	RW	0	当 CPU 核处于 halt 状态时，LPTIM 的计数器时钟控制位 0: 使能 1: 不使能
30: 23	保留	-	-	保留
22	DBG_I2C2_SMBUS_TIMEOUT	R	0	固定为0
21	DBG_I2C1_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C1 SMBUS 超时是否 frozen。 0: 与 normal 模式同样处理； 1: SMBUS 超时计数冻结。
20	保留	-	-	保留
19	DBG_CAN_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 CAN 停止。 0: CPU halt 时，CAN 与正常模式相同； 1: CPU halt 时，CAN 接收寄存器禁止；
18: 13	保留	-	-	保留
12	DBG_IWDG_STOP	RW	0	当 CPU 核处于 halt 状态时，IWDG 计数器的时钟控制位 0: 使能 1: 不使能

Bit	Name	R/W	Reset Value	Function
11	DBG_WWDG_STOP	RW	0	当 CPU 核处于 halt 状态时, WWDG 计数器的时钟控制位 0: 使能 1: 不使能
10	DBG_RTC_STOP	RW	0	当 CPU 核处于 halt 状态时, RTC 计数器的时钟控制位 0: 使能 1: 不使能
9: 6	保留	-		保留
5	DBG_TIM7_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM7的计数时钟。 0: 时钟使能 1: 时钟关闭
4	DBG_TIM6_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM6的计数时钟。 0: 时钟使能 1: 时钟关闭
3: 2	保留	-		保留
1	DBG_TIM3_STOP	RW	0	当 CPU 核处于 halt 状态时, TIM3计数器的时钟控制位 0: 使能 1: 不使能
0	DBG_TIM2_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM2的计数时钟。 0: 时钟使能 1: 时钟关闭

### 35.9.4. DBG APB 冻结寄存器 2 (DBG\_APB\_FZ2)

该寄存器用来配置 timer 在 debug 下的时钟控制。该寄存器被上电复位进行异步复位（不是系统复位）。它可以被调试者在系统复位下进行写。

**Address offset:** 0x0C

**Power on Reset value:** 0x0000 0000

仅支持32-bit 地址访问, 只读。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													DBG_	DBG_	DBG_
													TIM17_	TIM16_STO	TIM15_STO
													STOP	P	P
													RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0



DBG_ TIM14_STO P	Res	DBG_ TIM1_S TOP	Res
RW	-	RW	-

Bit	Name	R/W	Reset Value	Function
31: 19	保留	-		保留
18	DBG_TIM17_STOP	RW	0	当 CPU 核处于 halt 状态时, TIM17计数器的时钟控制位 0: 使能 1: 不使能
17	DBG_TIM16_STOP	RW	0	当 CPU 核处于 halt 状态时, TIM16计数器的时钟控制位 0: 使能 1: 不使能
16	DBG_TIM15_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM15的计数时钟。 0: 时钟使能 1: 时钟关闭
15	DBG_TIM14_STOP	RW	0	当 CPU 核处于 halt 状态时, TIM14计数器的时钟控制位 0: 使能 1: 不使能
14: 12	保留	-		保留
11	DBG_TIM1_STOP	RW	0	当 CPU 核处于 halt 状态时, TIM1计数器的时钟控制位 0: 使能 1: 不使能
10: 0	保留	-		保留

## 36. 版本历史

版本	日期	更新记录
V0.2	2025.5.30	初版



Puya Semiconductor Co., Ltd.

### 声 明

普冉半导体(上海)股份有限公司 (以下简称: “Puya” ) 保留更改、纠正、增强、修改 Puya 产品和/或本文档的权利, 恕不另行通知。用户可在下单前获取产品的最新相关信息。

Puya 产品是依据订单时的销售条款和条件进行销售的。

用户对 Puya 产品的选择和使用承担全责, 同时若用于其自己或指定第三方产品上的, Puya 不提供服务支持且不对此类产品承担任何责任。

Puya 在此不授予任何知识产权的明示或暗示方式许可。

Puya 产品的转售, 若其条款与此处规定不一致, Puya 对此类产品的任何保修承诺无效。

任何带有 Puya 或 Puya 标识的图形或字样是普冉的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代并替换先前版本中的信息。

普冉半导体(上海)股份有限公司 - 保留所有权利