



PY32F403 系列

32 位 ARM® Cortex®-M4F 微控制器

参考手册

# PY32F403 系列

## 32 位 ARM® Cortex® M4F 微控制器

### 参考手册



Puya Semiconductor (Shanghai) Co., Ltd

# 目录

<b>1. 简介 .....</b>	<b>15</b>
<b>2. 系统及存储器架构 .....</b>	<b>17</b>
2.1. Arm® Cortex®-M4 处理器简介 .....	17
2.2. 系统架构 .....	17
2.2.1. I_bus .....	18
2.2.2. D_bus .....	18
2.2.3. S_bus .....	18
2.2.4. DMA1/DMA2 BUS .....	18
2.2.5. 总线矩阵 .....	18
2.2.6. AHB/APB 总线桥 .....	18
2.3. 存储器组织架构 .....	18
2.3.1. 简介 .....	18
2.3.2. 存储器映射 .....	19
2.4. 嵌入式 SRAM .....	21
2.5. 嵌入式 FLASH .....	21
2.6. 位段 .....	21
2.7. 启动配置 .....	22
2.7.1. 嵌入式启动程序 .....	23
2.7.2. 物理重映射 .....	23
<b>3. 嵌入式闪存 .....</b>	<b>24</b>
3.1. 简介 .....	24
3.1.1. 主要特征 .....	24
3.1.2. 模块框图 .....	24
3.2. 功能描述 .....	24
3.2.1. 闪存结构 .....	24
3.2.2. 闪存读操作和访问延迟 .....	25
3.2.3. 自适应实时存储器加速器(ART Accelerator™) .....	26
3.2.4. 擦除和编程操作 .....	27
3.2.5. 闪存擦除操作 .....	27
3.2.6. 闪存写操作 .....	28
3.2.7. Flash 选项字节 .....	29
3.2.8. Flash 配置字节 .....	31
3.2.9. 闪存中断 .....	33
3.3. 寄存器描述 (基址 0x4002_2000) .....	33
3.3.1. FLASH access control register(FLASH_ACR) .....	33
3.3.2. FLASH key register(FLASH_KEYR) .....	34
3.3.3. FLASH option key register(FLASH_OPTKEYR) .....	34
3.3.4. FLASH status register(FLASH_SR) .....	34
3.3.5. FLASH control register(FLASH_CR) .....	35
3.3.6. FLASH option register(FLASH_OPTR) .....	36
3.3.7. FLASH WRP register(FLASH_WRPR) .....	37
3.3.8. FLASH sleep time config register(FLASH_STCR) .....	37
3.3.9. FLASH TS0 register(FLASH_TS0) .....	38
3.3.10. FLASH TS1 register(FLASH_TS1) .....	38
3.3.11. FLASH TS2P register(FLASH_TS2P) .....	38
3.3.12. FLASH TPS3 register(FLASH_TPS3) .....	38
3.3.13. FLASH TS3 register(FLASH_TS3) .....	39
3.3.14. FLASH ERASE TPE register(FLASH_ERSTPE) .....	39
3.3.15. FLASH PROGRAM TPE register(FLASH_PRGTPE) .....	39
3.3.16. FLASH PRE-PROGRAM TPE register (FLASH_PRETPE) .....	39
3.3.17. FLASH 寄存器映像 .....	40
<b>4. 电源控制 (PWR) .....</b>	<b>42</b>
4.1. 功能描述 .....	42
4.1.1. 电源结构 .....	42
4.1.2. 系统低功耗模式 .....	45
4.2. 寄存器描述 (基址 0x4000_7000) .....	50

4.2.1.	电源控制寄存器(PWR_CR)(0x00) .....	50
4.2.2.	电源控制/状态寄存器(PWR_CSR)(0x04).....	52
4.2.3.	PWR 寄存器映像.....	53
<b>5.</b>	<b>复位与时钟控制 (RCC) .....</b>	<b>55</b>
5.1.	复位功能描述 .....	55
5.1.1.	系统复位.....	55
5.1.2.	电源复位.....	55
5.1.3.	备份域复位.....	55
5.1.4.	备份域复位之外的复位的统一处理.....	55
5.2.	时钟功能描述 .....	56
5.2.1.	时钟结构.....	56
5.2.2.	时钟源 (时钟信号和参数以模拟模块需求为准) .....	57
5.3.	寄存器 (base: 0x40021000) .....	59
5.3.1.	时钟控制寄存器 (RCC_CR) .....	59
5.3.2.	时钟配置寄存器 (RCC_CFGR) .....	60
5.3.3.	时钟中断寄存器 (RCC_CIR) .....	62
5.3.4.	APB2 外设复位寄存器 (RCC_APB2RSTR) .....	63
5.3.5.	APB1 外设复位寄存器 (RCC_APB1RSTR) .....	64
5.3.6.	AHB1 外设时钟使能寄存器 (RCC_AHB1ENR) .....	66
5.3.7.	APB2 外设时钟使能寄存器 (RCC_APB2ENR) .....	67
5.3.8.	APB1 外设时钟使能寄存器 (RCC_APB1ENR) .....	67
5.3.9.	RTC domain 控制寄存器 (RCC_BDCR) .....	69
5.3.10.	控制/状态寄存器 (RCC_CSR) .....	70
5.3.11.	时钟复位配置寄存器 1 (RCC_CFGR1) .....	71
5.3.12.	AHB1 外设复位寄存器 (RCC_AHB1RSTR) .....	72
5.3.13.	AHB2 外设复位寄存器 (RCC_AHB2RSTR) .....	72
5.3.14.	AHB2 外设时钟使能寄存器 (RCC_AHB2ENR) .....	73
5.3.15.	时钟复位配置寄存器 2 (RCC_CFGR2) .....	73
5.3.16.	RCC 寄存器映像 .....	74
<b>6.</b>	<b>备份寄存器 (BKP) .....</b>	<b>76</b>
6.1.	简介 .....	76
6.1.1.	主要特性.....	76
6.2.	功能描述 .....	76
6.2.1.	侵入检测.....	76
6.2.2.	RTC 校准.....	76
6.3.	BKP 寄存器 (地址=0x40006C00) .....	79
6.3.1.	6.3.1 备份数据寄存器 (BKP_DRx) (x=1..42) .....	80
6.3.2.	RTC 时钟校准寄存器 (BKP_RTCCR) .....	80
6.3.3.	备份控制寄存器 (BKP_CR) .....	81
6.3.4.	备份控制/状态寄存器 (BKP_CSR) .....	81
6.3.5.	BKP 寄存器映射 .....	82
<b>7.</b>	<b>CRC 计算单元 (CRC) .....</b>	<b>90</b>
7.1.	简介 .....	90
7.2.	CRC 主要特征.....	90
7.3.	CRC 功能描述 .....	90
7.3.1.	CRC 框图 .....	90
7.4.	CRC 寄存器 .....	90
7.4.1.	数据寄存器 (CRC_DR) .....	91
7.4.2.	独立数据寄存器 (CRC_IDR) .....	91
7.4.3.	控制寄存器 (CRC_CR) .....	91
7.4.4.	CRC 寄存器映像 .....	92
<b>8.</b>	<b>通用 I/O (GPIO) .....</b>	<b>93</b>
8.1.	简介 .....	93
8.2.	主要特征 .....	93
8.3.	功能描述 .....	93
8.3.1.	通用 I/O (GPIO) .....	95

8.3.2.	I/O 引脚复用器和映射.....	95
8.3.3.	I/O 端口控制寄存器 .....	96
8.3.4.	I/O 端口数据寄存器 .....	96
8.3.5.	I/O 数据位操作 .....	96
8.3.6.	GPIO 锁定机制.....	96
8.3.7.	I/O 复用功能输入/输出模式配置 .....	97
8.3.8.	外部中断线/唤醒线 .....	97
8.3.9.	I/O 输入配置 .....	97
8.3.10.	输出配置.....	97
8.3.11.	复用功能配置 .....	97
8.3.12.	模拟配置.....	98
8.3.13.	HSE 或者 LSE 引脚配置为 GPIO .....	98
8.3.14.	BKP 区域 GPIO 使用 .....	98
8.4.	寄存器描述.....	98
8.4.1.	GPIO 端口模式寄存器 (GPIOx_MODER) (x= A..E) .....	98
8.4.2.	GPIO 端口输出类型寄存器 (GPIOx_OTYPER) (x= A..E) .....	98
8.4.3.	GPIO 端口输出速度寄存器 (GPIOx_OSPEEDR) (x= A..E).....	99
8.4.4.	GPIO 端口上拉/下拉寄存器 (GPIOx_PUPDR) (x= A..E).....	99
8.4.5.	GPIO 端口输入数据寄存器 (GPIOx_IDR) (x= A..E) .....	100
8.4.6.	GPIO 端口输出数据寄存器 (GPIOx_ODR) (x= A..E) .....	100
8.4.7.	GPIO 端口置位/复位寄存器 (GPIOx_BSRR) (x= A..E) .....	100
8.4.8.	GPIO 端口配置锁定寄存器 (GPIOx_LCKR) (x= A..E).....	101
8.4.9.	GPIO 复用功能低寄存器 (GPIOx_AFRL) (x= A..E).....	101
8.4.10.	GPIO 复用功能高寄存器 (GPIOx_AFRH) (x= A..E) .....	102
8.4.11.	GPIO 端口位复位寄存器 (GPIOx_BRR) (x= A..E) .....	102
8.4.12.	GPIO 寄存器地址映射 .....	102
<b>9.</b>	<b>系统配置控制器 (SYSCFG) .....</b>	<b>105</b>
9.1.	概述 .....	105
9.2.	SYSCFG 寄存器(baseaddr=0x40010000).....	105
9.2.1.	SYSCFG 配置寄存器 1 (SYSCFG_CFGR1) .....	105
9.2.2.	SYSCFG 配置寄存器 2 (SYSCFG_CFGR2).....	105
9.2.3.	SYSCFG 配置寄存器 3 (SYSCFG_CFGR3).....	106
9.2.4.	SYSCFG 配置寄存器 4 (SYSCFG_CFGR4).....	108
9.2.5.	SYSCFG 配置寄存器 5 (SYSCFG_CFGR5).....	108
9.2.6.	外部中断配置寄存器 1 (SYS_EXTICR1) .....	108
9.2.7.	外部中断配置寄存器 2 (SYS_EXTICR2) .....	109
9.2.8.	外部中断配置寄存器 3 (SYS_EXTICR3) .....	109
9.2.9.	外部中断配置寄存器 4 (SYS_EXTICR4) .....	110
9.2.10.	GPIOA 滤波使能 (PA_ENS) .....	110
9.2.11.	GPIOB 滤波使能 (PB_ENS) .....	110
9.2.12.	GPIOC 滤波使能 (PC_ENS) .....	111
9.2.13.	GPIOD 滤波使能 (PD_ENS) .....	111
9.2.14.	GPIOE 滤波使能 (PE_ENS) .....	111
9.2.15.	GPIO 模拟通道使能 (GPIO_ENA) .....	111
9.2.16.	Timer 时钟扩展控制 (TIM_CLK_EXT) .....	112
9.2.17.	SYSCFG 寄存器映像 .....	112
<b>10.</b>	<b>DMA 控制器 (DMA) .....</b>	<b>116</b>
10.1.	概述 .....	116
10.1.1.	主要特性.....	116
10.2.	功能描述 .....	116
10.2.1.	DMA 传输 .....	116
10.2.2.	仲裁器 .....	117
10.2.3.	DMA 通道 .....	117
10.2.4.	数据传输宽度/对齐方式/大小端 .....	120
10.2.5.	错误处理.....	121
10.2.6.	中断 .....	121

10.2.7. DMA 外设请求映射.....	121
10.3. 寄存器描述 (0x40020000) .....	122
10.3.1. DMA 中断状态寄存器 (DMA_ISR) .....	122
10.3.2. DMA 中断标志清零寄存器 (DMA_IFCR) .....	124
10.3.3. DMA 通道 1 配置寄存器 (DMA_CCR1) .....	126
10.3.4. DMA 通道 1 数据传输个数寄存器 (DMA_CNDTR1) .....	127
10.3.5. DMA 通道 1 外设地址寄存器 (DMA_CPAR1) .....	127
10.3.6. DMA 通道 1 存储器地址寄存器 (DMA_CMAR1) .....	127
10.3.7. DMA 通道 2 配置寄存器 (DMA_CCR2) .....	128
10.3.8. DMA 通道 2 数据传输个数寄存器 (DMA_CNDTR2) .....	129
10.3.9. DMA 通道 2 外设地址寄存器 (DMA_CPAR2) .....	129
10.3.10. DMA 通道 2 存储器地址寄存器 (DMA_CMAR2) .....	129
10.3.11. DMA 通道 3 配置寄存器 (DMA_CCR3) .....	130
10.3.12. DMA 通道 3 数据传输个数寄存器 (DMA_CNDTR3) .....	131
10.3.13. DMA 通道 3 外设地址寄存器 (DMA_CPAR3) .....	131
10.3.14. DMA 通道 3 存储器地址寄存器 (DMA_CMAR3) .....	131
10.3.15. DMA 通道 4 配置寄存器 (DMA_CCR4) .....	132
10.3.16. DMA 通道 4 数据传输个数寄存器 (DMA_CNDTR4) .....	133
10.3.17. DMA 通道 4 外设地址寄存器 (DMA_CPAR4) .....	133
10.3.18. DMA 通道 4 存储器地址寄存器 (DMA_CMAR4) .....	133
10.3.19. DMA 通道 5 配置寄存器 (DMA_CCR5) .....	134
10.3.20. DMA 通道 5 数据传输个数寄存器 (DMA_CNDTR5) .....	135
10.3.21. DMA 通道 5 外设地址寄存器 (DMA_CPAR5) .....	135
10.3.22. DMA 通道 5 存储器地址寄存器 (DMA_CMAR5) .....	135
10.3.23. DMA 通道 6 配置寄存器 (DMA_CCR6) .....	135
10.3.24. DMA 通道 6 数据传输个数寄存器 (DMA_CNDTR6) .....	136
10.3.25. DMA 通道 6 外设地址寄存器 (DMA_CPAR6) .....	137
10.3.26. DMA 通道 6 存储器地址寄存器 (DMA_CMAR6) .....	137
10.3.27. DMA 通道 7 配置寄存器 (DMA_CCR7) .....	137
10.3.28. DMA 通道 7 数据传输个数寄存器 (DMA_CNDTR7) .....	138
10.3.29. DMA 通道 7 外设地址寄存器 (DMA_CPAR7) .....	139
10.3.30. DMA 通道 7 存储器地址寄存器 (DMA_CMAR7) .....	139
10.3.31. DMA 寄存器映像 .....	139
<b>11. 中断和事件 .....</b>	<b>142</b>
11.1. 简介 .....	142
11.1.1. 主要特征 .....	142
11.1.2. 模块框图 .....	142
11.2. 功能说明 .....	142
11.2.1. 中断和异常向量表 .....	142
11.2.2. 外部中断/事件控制器 (EXIT) .....	144
11.3. 寄存器描述 .....	146
11.3.1. 中断屏蔽寄存器 (EXTI_IMR) .....	146
11.3.2. 事件屏蔽寄存器 (EXTI_EMR) .....	147
11.3.3. 上升沿触发选择寄存器 (EXTI_RTST) .....	147
11.3.4. 下降沿触发选择寄存器 (EXTI_FTSR) .....	147
11.3.5. 软件中断事件寄存器 (EXTI_SWIER) .....	148
11.3.6. 挂起寄存器 (EXTI_PR) .....	148
11.3.7. EXTI 寄存器地址映射 .....	148
<b>12. 模拟/数字转换 (ADC) .....</b>	<b>150</b>
12.1. 简介 .....	150
12.1.1. 主要特征 .....	150
12.1.2. 模块框图 .....	151
12.2. ADC 引脚定义 .....	151
12.3. 功能说明 .....	151
12.3.1. ADC 校准 .....	152
12.3.2. ADC 开-关控制 .....	152

12.3.3.	ADC 时钟 .....	152
12.3.4.	ADC 通道选择 .....	152
12.3.5.	强制停止 ADC .....	153
12.3.6.	转换模式.....	153
12.3.7.	注入通道管理 .....	155
12.3.8.	模拟看门狗 .....	156
12.3.9.	数据对齐.....	157
12.3.10.	可编程采样时间 .....	157
12.3.11.	外部触发转换.....	158
12.3.12.	可配置分辨率.....	159
12.3.13.	DMA 请求 .....	159
12.3.14.	双 ADC 模式 .....	159
12.3.15.	温度传感器和内部参考电压 .....	165
12.3.16.	电池监视 .....	166
12.3.17.	ADC 中断.....	166
12.4.	ADC 寄存器.....	166
12.4.1.	状态寄存器 (0x00: ADC_SR) .....	166
12.4.2.	ADC 控制寄存器 1 (0x04: ADC_CR1) .....	167
12.4.3.	ADC 控制寄存器 2 (0x08: ADC_CR2) .....	169
12.4.4.	ADC 采样时间寄存器 1 (0x0C: ADC_SMPR1) .....	171
12.4.5.	ADC 采样时间寄存器 2 (0x10: ADC_SMPR2) .....	171
12.4.6.	ADC 注入通道数据偏移寄存器 x (0x14-0x20: ADC_JOFR <sub>x</sub> ) x=1~4 .....	172
12.4.7.	ADC 看门狗高阈值寄存器 (0x24: ADC_HTR) .....	172
12.4.8.	ADC 看门狗低阈值寄存器 (0x28: ADC_LTR) .....	172
12.4.9.	ADC 规则序列寄存器 1 (0x2C: ADC_SQR1) .....	173
12.4.10.	ADC 规则序列寄存器 2 (0x30: ADC_SQR2) .....	173
12.4.11.	ADC 规则序列寄存器 3 (0x34: ADC_SQR3) .....	174
12.4.12.	ADC 注入序列寄存器 (0x38: ADC_JSQR) .....	174
12.4.13.	ADC 注入数据寄存器 x (0x3C-0x48: ADC_JDR <sub>x</sub> ) x=1~4 .....	175
12.4.14.	ADC 规则数据寄存器 (0x4C: ADC_DR) .....	175
12.4.15.	ADC 校准配置和状态寄存器 (0x50: ADC_CCSR) .....	175
12.4.16.	ADC 寄存器映像.....	176
13.	高级定时器 (TIM1 和 TIM8) .....	181
13.1.	简介 .....	181
13.1.1.	TIM1 和 TIM8 主要特征 .....	181
13.1.2.	模块框图 .....	181
13.2.	TIM1 和 TIM8 功能描述 .....	182
13.2.1.	时基单元 .....	182
13.2.2.	计数器模式 .....	183
13.2.3.	重复计数器 .....	191
13.2.4.	时钟选择 .....	192
13.2.5.	捕获/比较通道 .....	194
13.2.6.	输入捕获模式 .....	196
13.2.7.	PWM 输入模式 .....	197
13.2.8.	强置输出模式 .....	197
13.2.9.	输出比较模式 .....	198
13.2.10.	PWM 模式 .....	198
13.2.11.	互补输出和死区插入 .....	200
13.2.12.	使用刹车功能 .....	202
13.2.13.	在外部事件时清除 OC <sub>x</sub> REF 信号 .....	203
13.2.14.	产生六步 PWM 输出 .....	204
13.2.15.	单脉冲模式 .....	204
13.2.16.	编码器接口模式 .....	206
13.2.17.	定时器输入异或功能 .....	207
13.2.18.	与霍尔传感器的接口 .....	207
13.2.19.	TIMx 定时器和外部触发的同步 .....	209

13.2.20. 定时器同步 .....	212
13.2.21. 调试模式 .....	216
13.3. 寄存器描述 .....	216
13.3.1. TIM1 和 TIM8 控制寄存器 1 (TIMx_CR1) .....	216
13.3.2. TIM1 和 TIM8 控制寄存器 2 (TIMx_CR2) .....	217
13.3.3. TIM1 和 TIM8 从模式控制寄存器 (TIMx_SMCR) .....	218
13.3.4. TIM1 和 TIM8 DMA/中断使能寄存器 (TIMx_DIER) .....	220
13.3.5. TIM1 和 TIM8 状态寄存器 (TIMx_SR) .....	221
13.3.6. TIM1 和 TIM8 事件产生寄存器 (TIMx_EGR) .....	222
13.3.7. TIM1 和 TIM8 捕获/比较模式控制寄存器 1 (TIMx_CCMR1) .....	223
13.3.8. TIM1 和 TIM8 捕获/比较模式制寄存器 2 (TIMx_CCMR2) .....	225
13.3.9. TIM1 和 TIM8 捕获/比较使能寄存器 (TIMx_CCER) .....	226
13.3.10. TIM1 和 TIM8 计数器 (TIMx_CNT) .....	228
13.3.11. TIM1 和 TIM8 预分频器 (TIMx_PSC) .....	228
13.3.12. TIM1 和 TIM8 自动重装载寄存器 (TIMx_ARR) .....	228
13.3.13. TIM1 和 TIM8 重复计数寄存器 (TIMx_RCR) .....	229
13.3.14. TIM1 和 TIM8 捕获/比较寄存器 1 (TIMx_CCR1) .....	229
13.3.15. TIM1 和 TIM8 捕获/比较寄存器 2 (TIMx_CCR2) .....	229
13.3.16. TIM1 和 TIM8 捕获/比较寄存器 3 (TIMx_CCR3) .....	230
13.3.17. TIM1 和 TIM8 捕获/比较寄存器 4 (TIMx_CCR4) .....	230
13.3.18. TIM1 和 TIM8 刹车和死区寄存器 (TIMx_BDTR) .....	230
13.3.19. TIM1 和 TIM8 DMA 控制寄存器 (TIMx_DCR) .....	232
13.3.20. TIM1 和 TIM8 连续模式的 DMA 地址 (TIMx_DMAR) .....	232
13.3.21. TIM1 和 TIM8 寄存器映射 .....	233
<b>14. 通用定时器 (TIM2 到 TIM5) .....</b>	<b>237</b>
14.1. 简介 .....	237
14.1.1. TIM2/3/4/5 主要特征 .....	237
14.1.2. 模块框图 .....	237
14.2. TIM2/3/4/5 功能描述 .....	238
14.2.1. 时基单元 .....	238
14.2.2. 计数器模式 .....	240
14.2.3. 时钟选择 .....	247
14.2.4. 捕获/比较通道 .....	249
14.2.5. 输入捕获模式 .....	250
14.2.6. PWM 输入模式 .....	251
14.2.7. 强置输出模式 .....	252
14.2.8. 输出比较模式 .....	252
14.2.9. PWM 模式 .....	253
14.2.10. 在外部事件时清除 OCxREF 信号 .....	255
14.2.11. 单脉冲模式 .....	256
14.2.12. 编码器接口模式 .....	257
14.2.13. 定时器输入异或功能 .....	259
14.2.14. TIMx 定时器和外部触发的同步 .....	259
14.2.15. 定时器同步 .....	261
14.2.16. 调试模式 .....	265
14.3. 寄存器描述 .....	266
14.3.1. TIM2/3/4/5 控制寄存器 1 (TIMx_CR1) .....	266
14.3.2. TIM2/3/4/5 控制寄存器 2 (TIMx_CR2) .....	267
14.3.3. TIM2/3/4/5 从模式控制寄存器 (TIMx_SMCR) .....	268
14.3.4. TIM2/3/4/5 DMA/中断使能寄存器 (TIMx_DIER) .....	269
14.3.5. TIM2/3/4/5 状态寄存器 (TIMx_SR) .....	270
14.3.6. TIM2/3/4/5 事件产生寄存器 (TIMx_EGR) .....	271
14.3.7. TIM2/3/4/5 捕获/比较模式控制寄存器 1 (TIMx_CCMR1) .....	272
14.3.8. TIM2/3/4/5 捕获/比较模式制寄存器 2 (TIMx_CCMR2) .....	274
14.3.9. TIM2/3/4/5 捕获/比较使能寄存器 (TIMx_CCER) .....	275
14.3.10. TIM2/3/4/5 计数器 (TIMx_CNT) .....	276

14.3.11.	TIM2/3/4/5 预分频器 (TIMx_PSC) .....	277
14.3.12.	TIM2/3/4/5 自动重装载寄存器 (TIMx_ARR) .....	277
14.3.13.	TIM2/3/4/5 捕获/比较寄存器 1 (TIMx_CCR1) .....	277
14.3.14.	TIM2/3/4/5 捕获/比较寄存器 2 (TIMx_CCR2) .....	278
14.3.15.	TIM2/3/4/5 捕获/比较寄存器 3 (TIMx_CCR3) .....	278
14.3.16.	TIM2/3/4/5 捕获/比较寄存器 4 (TIMx_CCR4) .....	278
14.3.17.	TIM2/3/4/5 DMA 控制寄存器 (TIMx_DCR) .....	279
14.3.18.	TIM2/3/4/5 连续模式的 DMA 地址 (TIMx_DMAR) .....	280
14.3.19.	TIM2/3/4/5 寄存器映射 .....	280
<b>15.</b>	<b>通用定时器 (TIM9 和 TIM12) .....</b>	<b>284</b>
15.1.	简介 .....	284
15.1.1.	TIM9 和 TIM12 主要特征 .....	284
15.1.2.	模块框图 .....	284
15.2.	TIM9 和 TIM12 功能描述 .....	285
15.2.1.	时基单元 .....	285
15.2.2.	计数器模式 .....	286
15.2.3.	时钟选择 .....	289
15.2.4.	捕获/比较通道 .....	290
15.2.5.	输入捕获模式 .....	291
15.2.6.	PWM 输入模式 .....	292
15.2.7.	强置输出模式 .....	293
15.2.8.	输出比较模式 .....	293
15.2.9.	PWM 模式 .....	294
15.2.10.	单脉冲模式 .....	295
15.2.11.	TIMx 定时器和外部触发的同步 .....	296
15.2.12.	定时器同步 .....	298
15.2.13.	调试模式 .....	301
15.3.	寄存器描述 .....	301
15.3.1.	TIM9 和 TIM12 控制寄存器 1 (TIMx_CR1) .....	301
15.3.2.	TIM9 和 TIM12 从模式控制寄存器 (TIMx_SMCR) .....	302
15.3.3.	TIM9 和 TIM12 中断使能寄存器 (TIMx_DIER) .....	303
15.3.4.	TIM9 和 TIM12 状态寄存器 (TIMx_SR) .....	304
15.3.5.	TIM9 和 TIM12 事件产生寄存器 (TIMx_EGR) .....	304
15.3.6.	TIM9 和 TIM12 捕获/比较模式控制寄存器 1 (TIMx_CCMR1) .....	305
15.3.7.	TIM9 和 TIM12 捕获/比较使能寄存器 (TIMx_CCER) .....	307
15.3.8.	TIM9 和 TIM12 计数器 (TIMx_CNT) .....	308
15.3.9.	TIM9 和 TIM12 预分频器 (TIMx_PSC) .....	308
15.3.10.	TIM9 和 TIM12 自动重装载寄存器 (TIMx_ARR) .....	309
15.3.11.	TIM9 和 TIM12 捕获/比较寄存器 1 (TIMx_CCR1) .....	309
15.3.12.	TIM9 和 TIM12 捕获/比较寄存器 2 (TIMx_CCR2) .....	309
15.3.13.	TIM9 和 TIM12 寄存器映射 .....	310
<b>16.</b>	<b>通用定时器 (TIM10/TIM11/TIM13/TIM14) .....</b>	<b>312</b>
16.1.	简介 .....	312
16.1.1.	TIMx 主要特征 .....	312
16.2.	TIMx 功能描述 .....	313
16.2.1.	时基单元 .....	313
16.2.2.	计数器模式 .....	314
16.2.3.	时钟选择 .....	317
16.2.4.	捕获/比较通道 .....	317
16.2.5.	输入捕获模式 .....	318
16.2.6.	强置输出模式 .....	319
16.2.7.	输出比较模式 .....	319
16.2.8.	PWM 模式 .....	320
16.2.9.	单脉冲模式 .....	321
16.2.10.	定时器同步 .....	321
16.2.11.	调试模式 .....	321

16.3.	寄存器描述 .....	322
16.3.1.	TIMx 控制寄存器 1 (TIMx_CR1) .....	322
16.3.2.	TIMx 中断使能寄存器 (TIMx_DIER) .....	322
16.3.3.	TIMx 状态寄存器 (TIMx_SR) .....	323
16.3.4.	TIMx 事件产生寄存器 (TIMx_EGR) .....	323
16.3.5.	TIMx 捕获/比较模式控制寄存器 1 (TIMx_CCMR1) .....	324
16.3.6.	TIMx 捕获/比较使能寄存器 (TIMx_CCER) .....	325
16.3.7.	TIMx 计数器 (TIMx_CNT) .....	326
16.3.8.	TIMx 预分频器 (TIMx_PSC) .....	327
16.3.9.	TIMx 自动重装载寄存器 (TIMx_ARR) .....	327
16.3.10.	TIMx 捕获/比较寄存器 1 (TIMx_CCR1) .....	327
16.3.11.	TIMx 选项寄存器 (TIMx_OR) .....	327
16.3.12.	TIMx 寄存器映射 .....	328
<b>17.</b>	<b>基本定时器 (TIM6 和 TIM7) .....</b>	<b>330</b>
17.1.	简介 .....	330
17.1.1.	TIM6 和 TIM7 主要特征 .....	330
17.1.2.	模块框图 .....	330
17.2.	TIM6 和 TIM7 功能描述 .....	330
17.2.1.	时基单元 .....	330
17.2.2.	计数器模式 .....	331
17.2.3.	调试模式 .....	334
17.3.	寄存器描述 .....	334
17.3.1.	TIM6 和 TIM7 控制寄存器 1 (TIMx_CR1) .....	334
17.3.2.	TIM6 和 TIM7 控制寄存器 2 (TIMx_CR2) .....	335
17.3.3.	TIM6 和 TIM7 DMA/中断使能寄存器 (TIMx_DIER) .....	335
17.3.4.	TIM6 和 TIM7 状态寄存器 (TIMx_SR) .....	336
17.3.5.	TIM6 和 TIM7 事件产生寄存器 (TIMx_EGR) .....	336
17.3.6.	TIM6 和 TIM7 计数器 (TIMx_CNT) .....	337
17.3.7.	TIM6 和 TIM7 预分频器 (TIMx_PSC) .....	337
17.3.8.	TIM6 和 TIM7 自动重装载寄存器 (TIMx_ARR) .....	337
17.3.9.	TIM6 和 TIM7 寄存器映射 .....	337
<b>18.</b>	<b>实时时钟 (RTC) .....</b>	<b>340</b>
18.1.	概述 .....	340
18.1.1.	主要特性 .....	340
18.1.2.	结构框图 .....	340
18.2.	功能描述 .....	341
18.2.1.	寄存器复位 .....	341
18.2.2.	读 RTC 寄存器 .....	341
18.2.3.	配置 RTC 寄存器 .....	342
18.2.4.	RTC 标志设置 .....	342
18.2.5.	RTC 时序 .....	343
18.3.	寄存器描述(基址 0x4000_2800) .....	343
18.3.1.	RTC 控制寄存器高位(RTC_CRH)(0x00) .....	343
18.3.2.	RTC 控制寄存器低位(RTC_CRL)(0x04) .....	344
18.3.3.	RTC 预分频装载寄存器高位(RTC_PRLH)(0x08) .....	345
18.3.4.	RTC 预分频装载寄存器低位(RTC_PRLL)(0x0C) .....	346
18.3.5.	RTC 预分频余数寄存器高位(RTC_DIVH)(0x10) .....	346
18.3.6.	RTC 预分频余数寄存器低位(RTC_DIVL)(0x14) .....	346
18.3.7.	RTC 计数寄存器高位(RTC_CNTH)(0x18) .....	346
18.3.8.	RTC 计数寄存器低位(RTC_CNTL)(0x1C) .....	347
18.3.9.	RTC 闹钟寄存器高位(RTC_ALRH)(0x20) .....	347
18.3.10.	RTC 闹钟寄存器低位(RTC_ALRL)(0x24) .....	348
18.3.11.	RTC 寄存器映射 .....	348
<b>19.</b>	<b>独立看门狗 (IWDG) .....</b>	<b>350</b>
19.1.	简介 .....	350
19.2.	主要特征 .....	350

19.3.	功能说明 .....	350
19.3.1.	模块框图.....	350
19.3.2.	硬件看门狗 .....	351
19.3.3.	寄存器保护 .....	351
19.3.4.	调试模式.....	351
19.3.5.	IWDG 寄存器描述 .....	351
<b>20.</b>	<b>窗口看门狗 (WWDG) .....</b>	<b>355</b>
20.1.	简介 .....	355
20.1.1.	WWDG 主要特征.....	355
20.2.	WWDG 功能描述 .....	355
20.3.	如何编写看门狗超时程序 .....	356
20.4.	调试模式 .....	356
20.5.	寄存器描述 .....	356
20.5.1.	控制寄存器 (WWDG_CR) .....	357
20.5.2.	配置寄存器 (WWDG_CFR) .....	357
20.5.3.	状态寄存器 (WWDG_SR) .....	357
20.5.4.	WWDG 寄存器映射 .....	358
<b>21.</b>	<b>外部串行存储控制器 (ESMC) .....</b>	<b>359</b>
21.1.	简介 .....	359
21.2.	主要特征 .....	359
21.3.	功能说明 .....	359
21.3.1.	模块框图.....	359
21.3.2.	ESMC 功能描述 .....	360
21.3.3.	ESMC 寄存器描述(ONE BYTE 访问).....	366
21.3.4.	ESMC 寄存器描述(FOUR BYTE 访问).....	375
21.3.5.	ESMC 寄存器映像 .....	376
<b>22.</b>	<b>SDIO 接口 (SDIO) .....</b>	<b>379</b>
22.1.	简介 .....	379
22.1.1.	主要特征.....	379
22.1.2.	模块框图 .....	379
22.2.	功能说明 .....	379
22.2.1.	SDIO 适配器.....	380
22.2.2.	SDIO AHB 接口 .....	384
22.2.3.	卡功能描述 .....	385
22.2.4.	命令 .....	391
22.2.5.	响应 .....	393
22.3.	软件应用说明 .....	397
22.3.1.	电源控制 .....	397
22.3.2.	时钟程序 .....	397
22.3.3.	初始化 .....	397
22.3.4.	无数据命令和没有响应的序列 .....	399
22.3.5.	数据传输命令 .....	400
22.3.6.	单块或者多块写 .....	401
22.3.7.	数据流读 .....	402
22.3.8.	数据流写 .....	402
22.3.9.	在传输过程中发送停止和中断 .....	402
22.3.10.	暂停或恢复序列 .....	402
22.3.11.	读等待序列 .....	403
22.3.12.	控制器/DMA/FIFO 重置使用 .....	404
22.3.13.	专用中断 .....	404
22.3.14.	异步中断 .....	404
22.3.15.	错误处理 .....	404
22.3.16.	CT_ATA 操作 .....	405
22.4.	寄存器配置 .....	406
22.4.1.	SDIO 电源控制寄存器 (SDIO_POWER) .....	406
22.4.2.	SDIO 时钟控制寄存器 (SDIO_CLKCR) .....	407

22.4.3.	SDIO 参数寄存器 (SDIO_ARG) .....	408
22.4.4.	SDIO 命令寄存器 (SDIO_CMD) .....	408
22.4.5.	SDIO 命令响应寄存器 (SDIO_RESPCMD) .....	411
22.4.6.	SDIO 命令响应 1..4 寄存器 (SDIO_RESPX) .....	411
22.4.7.	SDIO 数据定时器寄存器 (SDIO_TMOUT) .....	412
22.4.8.	SDIO 数据块长度寄存器 (SDIO_BLKSIZE) .....	412
22.4.9.	SDIO 数据长度寄存器 (SDIO_DLEN) .....	413
22.4.10.	SDIO 控制寄存器 (SDIO_CTRL) .....	413
22.4.11.	SDIO 状态寄存器 (SDIO_STA) .....	415
22.4.12.	SDIO 中断状态寄存器 (SDIO_INTSTS) .....	416
22.4.13.	SDIO 中断掩码寄存器 (SDIO_INTMASK) .....	417
22.4.14.	SDIO FIFO 阈值寄存器 (SDIO_FIFOTH) .....	418
22.4.15.	SDIO 发送到卡计数器 (SDIO_TCBCNT) .....	419
22.4.16.	SDIO 发送到 FIFO 计数器 (SDIO_TBBCNT) .....	419
22.4.17.	SDIO 数据 FIFO 寄存器 (SDIO_FIFODATA) .....	419
22.4.18.	SDIO 寄存器映像.....	420
<b>23.</b>	<b>USB 全速设备接口 (USB) .....</b>	<b>423</b>
23.1.	简介 .....	423
23.2.	主要特征 .....	423
23.3.	功能说明 .....	423
23.3.1.	模块框图.....	423
23.3.2.	功能描述.....	423
23.3.3.	功能使用 .....	424
23.3.4.	USB 寄存器描述 .....	433
<b>24.</b>	<b>CAN 总线控制器.....</b>	<b>440</b>
24.1.	简介 .....	440
24.2.	主要特征 .....	440
24.3.	功能说明 .....	440
24.3.1.	模块框图.....	441
24.3.2.	动作模式.....	441
24.3.3.	波特率设定 .....	441
24.3.4.	发送缓冲器 .....	444
24.3.5.	接收缓冲器 .....	444
24.3.6.	接收筛选寄存器组 .....	444
24.3.7.	数据发送 .....	447
24.3.8.	单次数据发送 .....	447
24.3.9.	取消数据发送 .....	448
24.3.10.	数据接收 .....	448
24.3.11.	错误处理 .....	448
24.3.12.	节点关闭 .....	449
24.3.13.	仲裁失败位置捕捉 .....	449
24.3.14.	回环模式 .....	449
24.3.15.	静默模式 .....	450
24.3.16.	软件复位功能 .....	450
24.3.17.	兼容 CAN-FD 功能 .....	452
24.3.18.	时间触发 TTCAN.....	452
24.3.19.	TDC 和 RDC.....	454
24.3.20.	中断 .....	455
24.4.	寄存器说明 .....	455
24.4.1.	节点配置寄存器 (CANFD_TSNCR) .....	455
24.4.2.	位时序配置寄存器 (CANFD_ACBTR) .....	456
24.4.3.	CANFD_FDBTR.....	456
24.4.4.	限制与预分频配置寄存器 (CANFD_RLSSP) .....	457
24.4.5.	状态寄存器 (CANFD_IFR) .....	458
24.4.6.	中断使能寄存器 (CANFD_IER) .....	460
24.4.7.	传输状态寄存器 (CANFD_TSР) .....	461

24.4.8. 全局配置寄存器 (CANFD_MCR) .....	461
24.4.9. 错误警告寄存器 (CANFD_WECR) .....	466
24.4.10. 参考 ID 寄存器 (CANFD_REFMSG) .....	467
24.4.11. TTCAN 配置寄存器 (CANFD_TTCR) .....	467
24.4.12. TTCAN 触发寄存器 (CANFD_TTTR) .....	469
24.4.13. CANFD_SCMS .....	469
24.4.14. 筛选器组控制寄存器 (CANFD_ACFCR) .....	471
24.4.15. 筛选器组 code 寄存器 (CANFD_ACFC) .....	471
24.4.16. 筛选器组 mask 寄存器 (CANFD_ACFM) .....	471
24.4.17. CAN 接收 BUF 寄存器 (CANFD_RBUF) .....	472
24.4.18. CAN 发送 BUF 寄存器 (CANFD_TBUF) .....	472
24.4.19. CAN 寄存器映像.....	472
<b>25. 串行外设接口 (SPI) .....</b>	<b>473</b>
25.1. 简介 .....	473
25.1.1. 主要特征.....	473
25.1.2. 模块框图.....	474
25.2. SPI 功能描述.....	475
25.2.1. 概述 .....	475
25.2.2. 单主机和单从机通信.....	476
25.2.3. 多从机通信 .....	478
25.2.4. 多主机通信 .....	479
25.2.5. 从选择(NSS)脚管理.....	480
25.2.6. 通讯格式.....	481
25.2.7. SPI 配置 .....	482
25.2.8. SPI 使能流程 .....	482
25.2.9. 数据传输和接收流程.....	483
25.2.10. 状态标志.....	486
25.2.11. 错误标志.....	487
25.2.12. SPI 中断 .....	487
25.2.13. CRC 计算 .....	488
25.3. I2S 功能描述 .....	488
25.3.1. 简介 .....	488
25.3.2. 音频协议 .....	489
25.3.3. 时钟 .....	500
25.3.4. 发送与接收 .....	501
25.3.5. 标志位 .....	503
25.3.6. 中断 .....	504
25.4. 寄存器描述 .....	504
25.4.1. SPI_CR1 寄存器 .....	504
25.4.2. SPI_CR2 寄存器 .....	506
25.4.3. SPI_SR 寄存器.....	507
25.4.4. SPI_DR 寄存器 .....	509
25.4.5. SPI_CRCPR 寄存器.....	509
25.4.6. SPI_RXCRCR 寄存器.....	509
25.4.7. SPI_TXCRCR 寄存器.....	510
25.4.8. SPI_I2S_CFGR 寄存器.....	510
25.4.9. SPI_I2SPR 寄存器 .....	512
25.4.10. SPI 寄存器映射 .....	512
<b>26. I2C 接口 .....</b>	<b>514</b>
26.1. 简介 .....	514
26.1.1. 主要特征.....	514
26.2. I2C 功能描述 .....	515
26.2.1. 简介 .....	515
26.2.2. 包错误校验 (PEC) .....	516
26.2.3. 从模式 .....	517
26.2.4. 主模式 .....	519

26.2.5. 错误状态 .....	525
26.2.6. DMA 功能 .....	526
26.2.7. 系统管理总线 SMBus .....	528
26.3. 寄存器描述 .....	529
26.3.1. I2C Control register 1 (I2C_CR1) .....	529
26.3.2. I2C Control register 2 (I2C_CR2) .....	532
26.3.3. I2C Own address register1 (I2C_OAR1) .....	533
26.3.4. I2C own address register2 (I2C_OAR2) .....	534
26.3.5. I2C Data register(I2C_DR) .....	534
26.3.6. I2C Staus register (I2C_SR1) .....	535
26.3.7. I2C Status register2 (I2C_SR2) .....	539
26.3.8. I2C Clock control register(I2C_CCR) .....	540
26.3.9. I2C TRISE register (I2C_TRISE) .....	541
26.3.10. I2C 寄存器映射 .....	541
27. 通用同步异步收发器 (USART) .....	543
27.1. 简介 .....	543
27.1.1. 主要特征 .....	543
27.2. USART 功能描述 .....	545
27.2.1. 传输 pin 脚 .....	545
27.2.2. USART 发送 .....	546
27.2.3. 数据接收 .....	547
27.2.4. 分数波特率的产生 .....	550
27.2.5. USART 接收器容忍时钟的变化 .....	551
27.2.6. USART 自动波特率检测 .....	551
27.2.7. 多处理器通信 .....	552
27.2.8. 校验控制 .....	553
27.2.9. LIN(局域互联网)模式 .....	553
27.2.10. 同步模式 .....	555
27.2.11. 单线半双工通信 .....	556
27.2.12. 智能卡 .....	556
27.2.13. IrDA SIR ENDEC 功能模块 .....	558
27.2.14. 利用 DMA 连续通信 .....	559
27.2.15. 硬件流控制 .....	561
27.2.16. 中断请求 .....	563
27.3. 寄存器描述 .....	563
27.3.1. 控制寄存器 (SR) .....	563
27.3.2. 数据寄存器 (USART_DR) .....	566
27.3.3. 波特比率寄存器 (USART_BRR) .....	566
27.3.4. 控制寄存器 (USART_CR) .....	567
27.3.5. 保护时间和预分频寄存器 (USART_GTPR) .....	572
27.3.6. USART 寄存器映射 .....	573
28. 时钟校准控制器 (CTC) .....	574
28.1. 简介 .....	574
28.1.1. 主要特征 .....	574
28.1.2. 模块框图 .....	574
28.2. 功能描述 .....	575
28.2.1. REF 同步脉冲发生器 .....	575
28.2.2. CTC 校准计数器 .....	575
28.2.3. 频率评估和自动校准过程 .....	575
28.2.4. 软件编程指南 .....	576
28.3. 寄存器描述 (基址 0x4000_C800) .....	577
28.3.1. 控制寄存器 0(CTC_CTL0) .....	577
28.3.2. 控制寄存器 1(CTC_CTL1) .....	578
28.3.3. 状态寄存器 (CTC_SR) .....	579
28.3.4. 中断清除寄存器 (CTC_INTC) .....	581
28.3.5. CTC 寄存器映射 .....	582
29. 调试支持 (DBG) .....	583

---

29.1.	简介 .....	583
29.1.1.	主要特性 .....	583
29.2.	功能描述 .....	583
29.2.1.	低功耗模式的调试中支持 .....	583
29.3.	寄存器 baseaddr=0xE0042000 .....	583
29.3.1.	ID 编码 (DBGMCU_IDCODE) (0xE004_2000) .....	583
29.3.2.	调试配置寄存器 (DBGMCU_CR) (0xE004_2004) .....	584
29.3.3.	DBG 寄存器映像 .....	587
30.	版本历史 .....	589

# 1. 简介

系统包括 Cortex-M4 处理器内核，此处理器内核包括 SIMD 的 DSP 增强和 Floating Process Unit(FPU)单元，其中 FPU 单元可以根据配置进行打开或者关闭，但是 SIMD 的指令则默认支持。程序可以直接运行于 SRAM、芯片上的内嵌的 Flash 以及外置的 Quad SPI 的 Flash。其中片上的 Flash 有系统运行期间仅用作软件指令和数据的只读空间，为了提高 CM4 的运行效率，在系统中增加了指令和数据缓存，同时为了减少对于片上 Flash 的访问时间，片上 Flash 提供了 128-bit 的数据读取位宽。其中 Quad/Octal SPI 的 Flash 可以扩展系统非易失性的可运行存储空间，同时又仅占用很少的 GPIO 端口。系统包含的 DMA1 和 2 最多可以支持 12 个通道进行存储器、外设之间的直接数据传送，支持的外设包括 ADC、SDIO、I2S、I2C 和 USART。

系统包含有 APB1、APB2 两个外设系统总线，其速度同 AHB 总线。

外设包括 17 个定时器单元及 13 个通讯接口单元。其中前者包括有 10 个 16 位的定时器、2 个 16 位的支持 PWM 控制的定时器、2 个看门狗定时器、2 个基本定时器及 1 个 systick 定时器。后者包括 2 个 I2C 接口、5 个 USART 接口、3 个 SPI 接口、1 个 CAN 总线控制器、支持全速运行的 USB2.0 接口和 SDIO 接口。

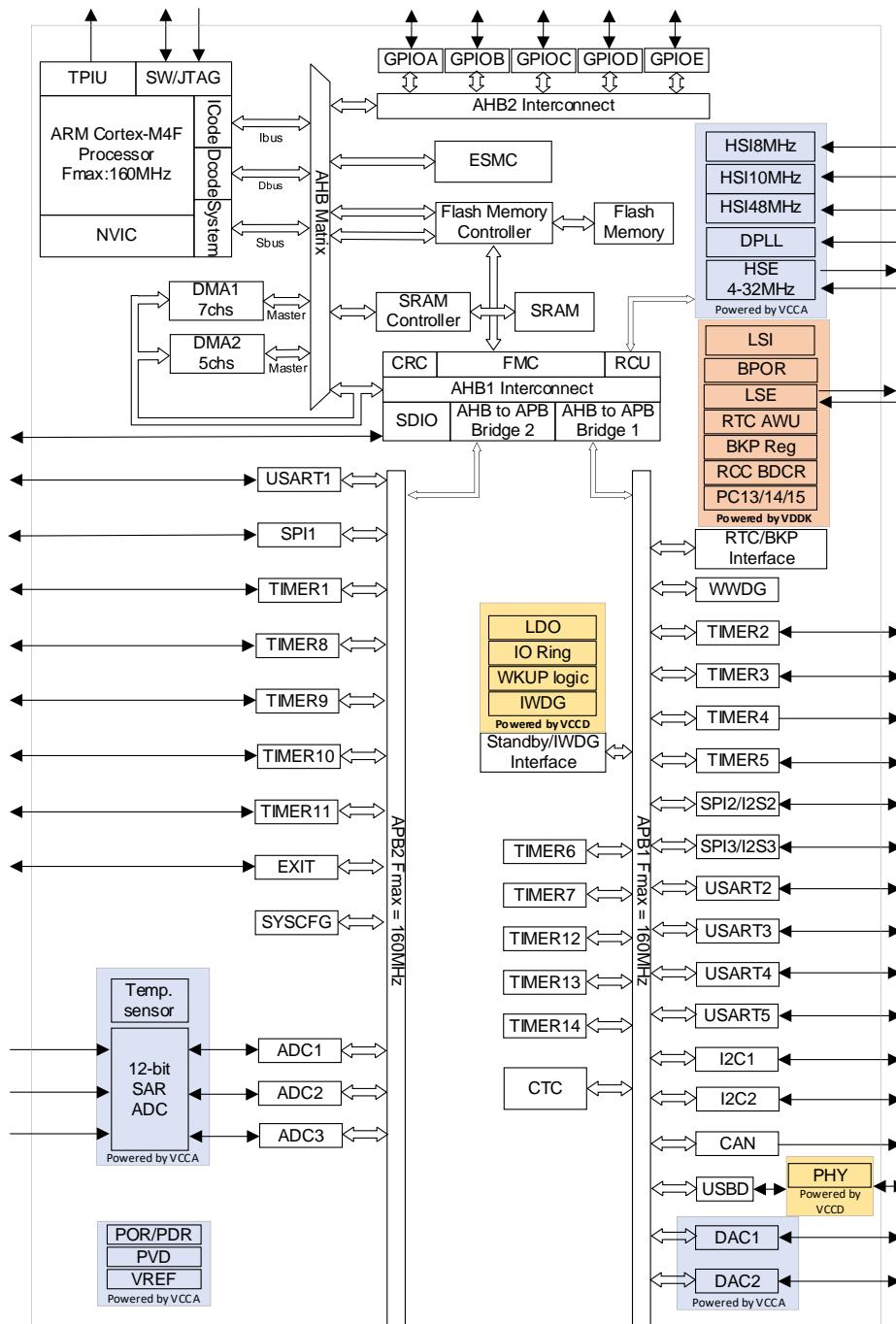


图 1-1 模块框图

## 2. 系统及存储器架构

PY32F403XX 系列器件是基于 Arm® Cortex®-M4 处理器的 32 位通用微控制器。Arm® Cortex®-M4 处理器包括三条 AHB 总线分别称为 I-CODE 总线、D-Code 总线和系统总线。Cortex®-M4 处理器的所有存储访问，根据不同的目的和目标存储空间，都会在这三条总线上执行。存储器的组织采用了哈佛结构，预先定义的存储器映射和高达 4 GB 的存储空间，充分保证了系统的灵活性和可扩展性。

### 2.1. Arm® Cortex®-M4 处理器简介

Cortex®-M4 处理器是一个具有低中断延迟时间和低成本调试特性的 32 位处理器。高集成度和增强的特性使 Cortex™-M4 处理器适合于那些需要高性能和低功耗微控制器的市场领域。Cortex®-M4 处理器基于 Armv7 架构，并且支持一个强大且可扩展的指令集，包括通用数据处理 I/O 控制任务、增强的数据处理位域操作、DSP(数字信号处理)和浮点运算指令。下面列出由 Cortex®-M4 提供的一些系统外设：

- 内部总线矩阵，用于实现 I-Code 总线、D-Code 总线、系统总线、专用总线(PPB)以及调试专用总线(AHB-AP)的互联；
- 嵌套式向量型中断控制器 (NVIC)；
- 闪存地址重载及断点单元 (FPB)；
- 数据观测点及跟踪单元 (DWT)；
- 指令跟踪宏单元 (ITM)；
- 嵌入式跟踪宏单元 (ETM)；
- 串行线和 JTAG 调试接口 (SWJ-DP)；
- 跟踪端口接口单元 (TPIU)；
- 内存保护单元 (MPU)；
- 浮点运算单元 (FPU)。

### 2.2. 系统架构

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

- 五条主控总线：
  - Cortex™-M4F 内核 I 总线、D 总线和 S 总线
  - DMA1 存储器总线
  - DMA2 存储器总线
- 五条被控总线：
  - 内部 Flash ICode 总线
  - 内部 Flash DCode 总线
  - 内部 SRAM 总线
  - AHB1 外设总线 (包括 AHB 到 APB 的总线桥和 APB 外设)
  - AHB2 外设总线
  - ESMC

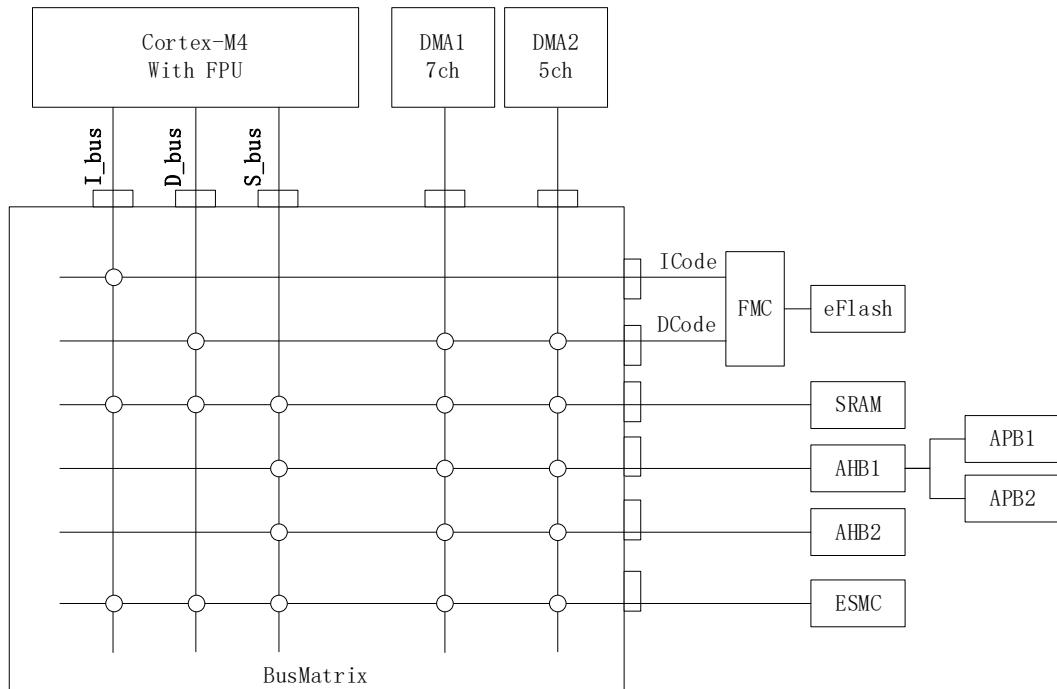


图 2-1 总线架构

### 2.2.1. I\_bus

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

### 2.2.2. D\_bus

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

### 2.2.3. S\_bus

此总线用于将 Cortex™-M4F 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据。此总线访问的对象是 64 KB 的内部 SRAM、包括 AHB 外设在内的 APB1 外设、APB2 外设以及通过 ESMC 的外部存储器。

### 2.2.4. DMA1/DMA2 BUS

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线访问的对象是 64KB 的内部 SRAM、包括 AHB 外设在内的 APB1 外设、APB2 外设以及通过 ESMC 的外部存储器。

### 2.2.5. 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。系统包含 5 个主设备 Master，分别为 ARM® Cortex™-M4 的 I\_BUS、D\_BUS、S\_BUS、DMA1 及 DMA2；及 4 个从设备 Slave，分别为内部的 Flash 存储器、内部的 SRAM、AHB 外设（AHB2APB 桥 1 对应的外设和 AHB2APB 桥 2 对应的外设）、ESMC。

### 2.2.6. AHB/APB 总线桥

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

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

## 2.3. 存储器组织架构

### 2.3.1. 简介

Arm® Cortex®-M4 处理器采用哈佛结构，可以使用相互独立的总线来读取指令和加载/存储数据。指令代码和数据都位于相同的存储器地址空间，但在不同的地址范围。程序存储器，数据存储器，寄存器和 I/O 端口都在同一个线性的 4 GB 的地址空间之内。

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

有关外设寄存器映射的详细信息，请参见相关章节。

可寻址的存储空间分为 8 个主要块，每个块为 512 MB。

### 2.3.2. 存储器映射

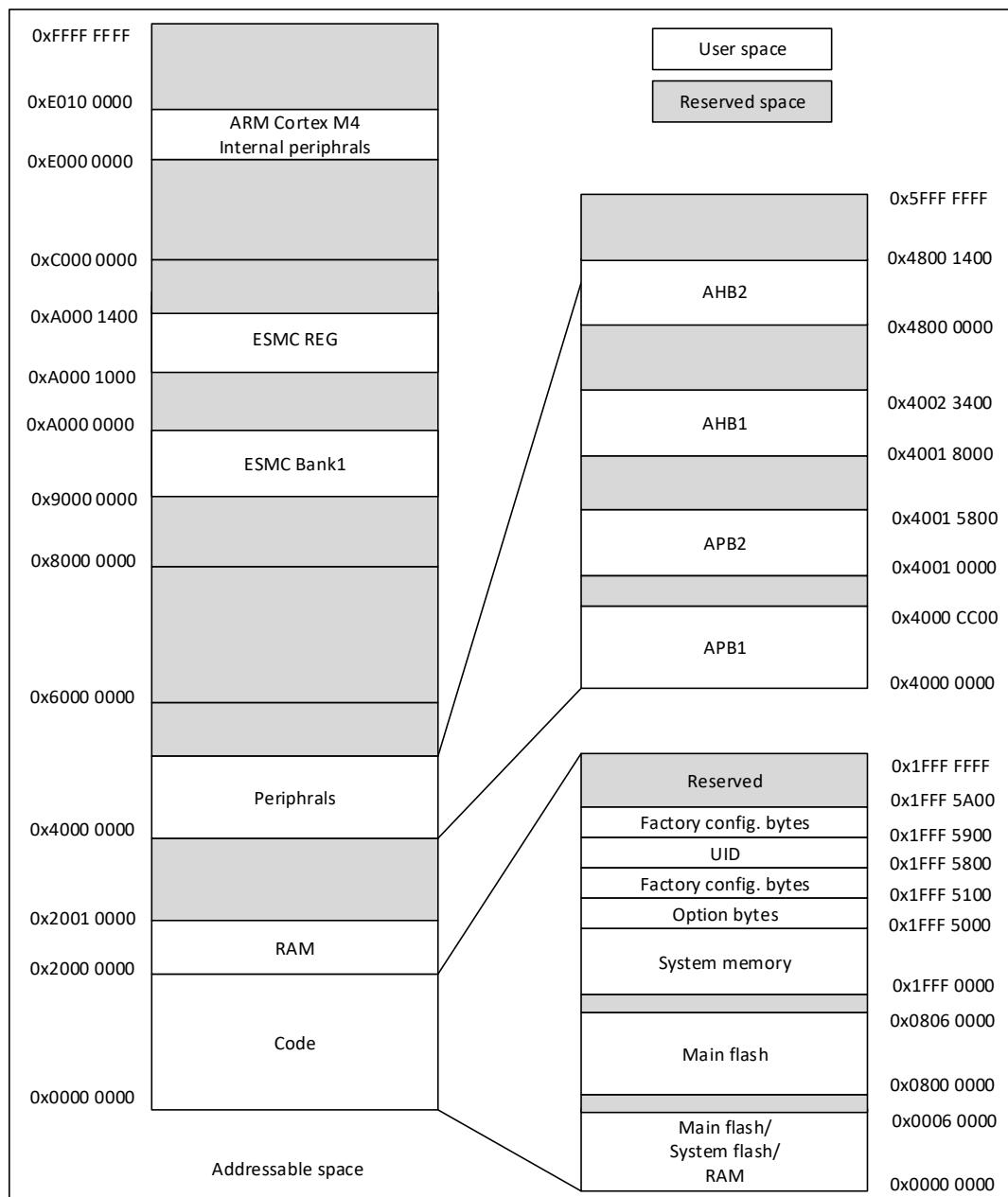


图 2-2 存储器映射

其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间。详细的存储和寄存器空间映射参考下面的表格。

下表中给出了产品中可使用的外设及对应的地址边界。

表 2-1 外设寄存器地址

存储器起止地址	外设	总线	寄存器映射
0xA000 1000 - 0xA000 13FF	ESMC	AHB	
0x4800 1400 - 0x5FFF FFFF	保留	AHB2	
0x4800 1000 - 0x4800 13FF	GPIOE		
0x4800 0C00 - 0x4800 0FFF	GPIOD		
0x4800 0800 - 0x4800 0BFF	GPIOC		
0x4800 0400 - 0x4800 07FF	GPIOB		
0x4800 0000 - 0x4800 03FF	GPIOA		
0x4002 3400 - 0x47FF FFFF	保留	AHB1	
0x4002 3000 - 0x4002 33FF	CRC		
0x4002 2400 - 0x4002 2FFF	保留		
0x4002 2000 - 0x4002 23FF	FMC		
0x4002 1400 - 0x4002 1FFF	保留		
0x4002 1000 - 0x4002 13FF	RCC		
0x4002 0800 - 0x4002 0FFF	保留		
0x4002 0400 - 0x4002 07FF	DMA2	APB2	
0x4002 0000 - 0x4002 03FF	DMA1		
0x4001 8400 - 0x4001 FFFF	保留		
0x4001 8000 - 0x4001 83FF	SDIO		
0x4001 5800 - 0x4001 7FFF	保留		
0x4001 5400 - 0x4001 57FF	TIMER11		
0x4001 5000 - 0x4001 53FF	TIMER10		
0x4001 4C00 - 0x4001 4FFF	TIMER9	APB2	
0x4001 4000 - 0x4001 4BFF	保留		
0x4001 3C00 - 0x4001 3FFF	ADC3		
0x4001 3800 - 0x4001 3BFF	USART1		
0x4001 3400 - 0x4001 37FF	TIMER8		
0x4001 3000 - 0x4001 33FF	SPI1		
0x4001 2C00 - 0x4001 2FFF	TIMER1		
0x4001 2800 - 0x4001 2BFF	ADC2	APB1	
0x4001 2400 - 0x4001 27FF	ADC1		
0x4001 0800 - 0x4001 23FF	保留		
0x4001 0400 - 0x4001 07FF	EXTI		
0x4001 0000 - 0x4001 03FF	SYSCFG		
0x4000 CC00 - 0x4000 FFFF	保留		
0x4000 C800 - 0x4000 CBFF	CTC		
0x4000 7800 - 0x4000 C7FF	保留	APB1	
0x4000 7400 - 0x4000 77FF	保留		
0x4000 7000 - 0x4000 73FF	PWR		
0x4000 6C00 - 0x4000 6FFF	BKP		
0x4000 6800 - 0x4000 6BFF	保留		
0x4000 6400 - 0x4000 67FF	CAN		
0x4000 6000 - 0x4000 63FF	USBD/CAN 共享 512 字节 SRAM		
0x4000 5C00 - 0x4000 5FFF	USBD	APB1	
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UASRT5		

存储器起止地址	外设	总线	寄存器映射
0x4000 4C00 - 0x4000 4FFF	UASRT4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	保留		
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S		
0x4000 3800 - 0x4000 3BFF	SPI2/I2S		
0x4000 3400 - 0x4000 37FF	保留		
0x4000 3000 - 0x4000 33FF	IWDG		
0x4000 2C00 - 0x4000 2FFF	WWDG		
0x4000 2800 - 0x4000 2BFF	RTC		
0x4000 2400 - 0x4000 27FF	保留		
0x4000 2000 - 0x4000 23FF	TIMER14		
0x4000 1C00 - 0x4000 1FFF	TIMER13		
0x4000 1800 - 0x4000 1BFF	TIMER12		
0x4000 1400 - 0x4000 17FF	TIMER7		
0x4000 1000 - 0x4000 13FF	TIMER6		
0x4000 0C00 - 0x4000 0FFF	TIMER5		
0x4000 0800 - 0x4000 0BFF	TIMER4		
0x4000 0400 - 0x4000 07FF	TIMER3		
0x4000 0000 - 0x4000 03FF	TIMER2		

## 2.4. 嵌入式 SRAM

PY32F403XX 系列器件内置最大 64K 字节的静态 SRAM，根据不同产品定义，SRAM 容量会有差异，具体请参考 DataSheet。它可以以字节、半字(16 位)或全字(32 位)访问。SRAM 的起始地址是 0x2000 0000。

## 2.5. 嵌入式 FLASH

Flash 存储器有两个不同的物理区域组成：

- Main flash 区域， 384KBytes， 它包含应用程序和用户数据。可根据不同产品定义，Flash 容量会有差异，具体请参考 DataSheet。用于存储用户程序和用户数据。软件对设定范围外空间的访问会产生 hard fault。
- Information 区域， 24Kbytes， 它包括以下部分：
  - Factory config. bytes: 3328Bytes，用于存放 trimming 数据（含 HSI trimming 数据）、flash size 配置信息、上电读校验码。
  - Factory config. bytes:
    - 芯片上电读校验码
    - 芯片硬件 Trimming 配置值
    - Device ID code
  - UID: 512Bytes，用于存放芯片的 UID
  - Option bytes: 256Bytes，用于存放芯片硬件和存储保护的配置值
  - System memory (系统存储器) : 20KBytes，用于存放 Boot loader

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

## 2.6. 位段

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

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

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

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$

其中：

- bit\_word\_addr 代表别名区域中将映射到目标位的字的地址
- bit\_band\_base 代表别名区域的起始地址
- byte\_offset 代表目标位所在位段区域中的字节编号
- bit\_number 代表目标位的位位置 (0-7)

#### 示例

下例说明如何将 SRAM 地址 0x20000300 处字节的位 2 映射到别名区域：

$$0x22006008 = 0x22000000 + (0x300*32) + (2^4)$$

对地址 0x22006008 执行写操作相当于在 SRAM 地址 0x20000300 处字节的位 2 执行读-修改-写操作。

对地址 0x22006008 执行读操作将返回 SRAM 地址 0x20000300 处字节的位 2 的值（0x01 表示位置位， 0x00 表示位复位）。

有关位段的详细信息，请参见 Cortex™-M4F 编程手册。

## 2.7. 启动配置

在 PY32F403xx 里，可以通过 BOOT[1:0]引脚选择三种不同引导模式，如下表所示。

表 2-2 BOOT 引脚配置

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

在系统复位后，BOOT 引脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，来选择在复位后的启动模式。从待机模式退出时，BOOT 引脚的值将被重新锁存；因此，在待机模式下 BOOT 引脚应保持为需要的启动配置。

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

- 从主闪存存储器启动：主闪存存储器被映射到启动空间（0x0000\_0000），但仍然能够在它原有的地址（0x0800\_0000）访问它，即闪存存储器的内容可以在两个地址区域访问，0x0000\_0000 或 0x0800\_0000。
- 从系统存储器启动：系统存储器被映射到启动空间（0x0000\_0000），但仍然能够在它原有的地址（0x1FFF\_B000）访问它。
- 从内置 SRAM 启动：SRAM 被映射到启动空间（0x0000\_0000），但仍然能够在它原有的地址（0x2000\_0000）访问它。

在启动延迟之后，CPU 从地址 0x0000\_0000 获取堆栈顶的地址，并从启动存储器的 0x0000\_0004 指示的地址开始执行代码。因为固定的存储器映像，代码区始终从地址 0x0000\_0000 开始（通过 ICode 和 DCode 总线访问），而数据区（SRAM）始终从地址 0x2000\_0000 开始（通过系统总线访问）。Cortex M4F 的 CPU 始终从 ICode 总线获取复位向量，即启动仅适合于从代码区开始（典型地从 Flash 启动）。PY32F403xx 系列微控制器实现了一个特殊的机

制，可以从片上 SRAM 启动。当从片上 SRAM 启动，在应用程序的初始化代码中，必须使用 NVIC 的异常表和偏移寄存器，重新映射向量表之 SRAM 中。

### 2.7.1. 嵌入式启动程序

系统存储器中包含内嵌的引导加载程序，可用于对闪存存储器编程。引导加载程序是通过 USART1、USART2 或者 USB 接口对闪存存储器进行重新编程。

### 2.7.2. 物理重映射

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

- 主 Flash
- 系统存储器
- 嵌入式 SRAM
- ESMC

表 2-3 端口位配置表

地址	主 Flash 中的启动/ 重映射	嵌入式 SRAM 中的 启动/重映射	系统存储器中的启动/重映射	ESMC 中的重映 射
0x2000 0000 - 0x2000 FFFF	SRAM	SRAM	SRAM	SRAM
0x1FFF 0000 - 0x1FFF 4FFF	系统存储器	系统存储器	系统存储器	系统存储器
0x0808 0000 - 0x0FFF FFFF	保留	保留	保留	保留
0x0800 0000 - 0x0807 FFFF	Flash	Flash	Flash	Flash
0x0006 0000 - 0x07FF FFFF	保留	保留	保留	ESMC
0x0000 0000 - 0x0005 FFFF	Flash 使用别名	SRAM 使用别名	系统存储器 (20KB) 使用别名	ESMC

## 3. 嵌入式闪存

### 3.1. 简介

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

Flash 接口通过指令预取和缓存机制加速代码执行。

#### 3.1.1. 主要特征

- 存储器组织结构
  - Main memory: 最大 384 Kbytes
  - Information memory: 24 Kbytes
  - Page size: 256 bytes
  - Sector size: 2 Kbytes
  - Block size: 32 Kbytes
  - 128-bit wide data read
  - 32-bit wide data write
- Flash 读操作
- Flash 编程操作 (页编程)
- Flash 擦除操作 (页擦/扇区擦/块擦/全擦)
- 读/写保护
- I-Code 上的预取操作
- I-Code 上的分支缓存
- D-Code 上的数据缓存
- 选项字节加载

#### 3.1.2. 模块框图

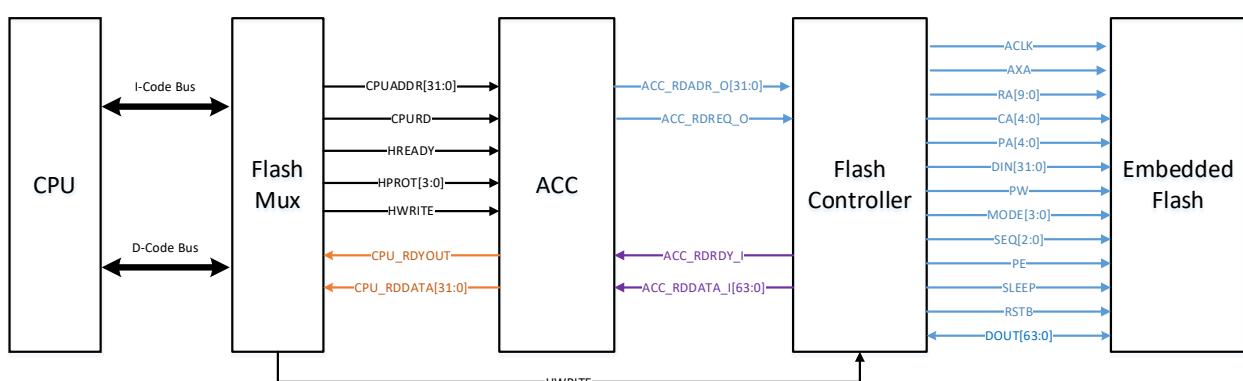


图 3-1 模块框图

### 3.2. 功能描述

#### 3.2.1. 闪存结构

Flash 存储器由 128 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 256Bytes，Sector 大小为 2Kbytes，Block 大小为 32KBytes

从功能上，Flash 存储器分为 Main Flash 和 information flash，前者容量最大是 384Kbytes，后者容量为 24Kbyte，如下：

表 3-1 闪存结构及边界地址

Area	Space Name		Address	Size (bytes)
Main memory	Block 0	Sector 0	Page 0-7	0x0800 0000 - 0x0800 07FF
		Sector 1	Page 8-15	0x0800 0800 - 0x0800 0FFF
		Sector 2	Page16-23	0x0800 1000 - 0x0800 17FF
		Sector 3	Page 24-31	0x0800 1800 - 0x0800 1FFF
		Sector 4	Page 32-39	0x0800 2000 - 0x0800 27FF
		Sector 5	Page 40-47	0x0800 2800 - 0x0800 2FFF
		Sector 6	Page 48-55	0x0800 3000 - 0x0800 37FF
		Sector 7	Page 56-63	0x0800 3800 - 0x0800 3FFF
		Sector 8	Page 64-71	0x0800 4000 - 0x0800 47FF
		Sector 9	Page 72-79	0x0800 4800 - 0x0800 4FFF
		Sector 10	Page 80-87	0x0800 5000 - 0x0800 57FF
		Sector 11	Page 88-95	0x0800 5800 - 0x0800 5FFF
		Sector 12	Page 96-103	0x0800 6000 - 0x0800 67FF
		Sector 13	Page 104-111	0x0800 6800 - 0x0800 6FFF
		Sector 14	Page 112-119	0x0800 7000 - 0x0800 77FF
		Sector 15	Page 120-127	0x0800 7800 - 0x0800 7FFF
	...	...	...	...
	Block 11	Sector 176	Page 1408-1415	0x0805 8000 - 0x0805 87FF
		Sector 177	Page 1416-1423	0x0805 8800 - 0x0805 8FFF
		Sector 178	Page 1424-1431	0x0805 9000 - 0x0805 97FF
		Sector 179	Page 1432-1439	0x0805 9800 - 0x0805 9FFF
		Sector 180	Page 1440-1447	0x0805 A000 - 0x0805 A7FF
		Sector 181	Page 1448-1455	0x0805 A800 - 0x0805 AFFF
		Sector 182	Page 1456-1463	0x0805 B000 - 0x0805 B7FF
		Sector 183	Page 1464-1471	0x0805 B800 - 0x0805 BFFF
		Sector 184	Page 1472-1479	0x0805 C000 - 0x0805 C7FF
		Sector 185	Page 1480-1487	0x0805 C800 - 0x0805 CFFF
		Sector 186	Page 1488-1495	0x0805 D000 - 0x0805 D7FF
		Sector 187	Page 1496-1503	0x0805 D800 - 0x0805 DFFF
		Sector 188	Page 1504-1511	0x0805 E000 - 0x0805 E7FF
		Sector 189	Page 1512-1519	0x0805 E800 - 0x0805 EFFF
		Sector 190	Page 1520-1527	0x0805 F000 - 0x0805 F7FF
		Sector 191	Page 1528-1535	0x0805 F800 - 0x0805 FFFF
Information block	System memory		0x1FFF 0000 - 0x1FFF 4FFF	20 K
	Option bytes		0x1FFF 5000 - 0x1FFF 50FF	256
	Factory config. bytes		0x1FFF 5100 - 0x1FFF 51FF	256
	Factory config. bytes		0x1FFF 5200 - 0x1FFF 52FF	256
	Factory config. bytes		0x1FFF 5300 - 0x1FFF 53FF	256
	Factory config. bytes		0x1FFF 5400 - 0x1FFF 54FF	256
	Factory config. bytes		0x1FFF 5500 - 0x1FFF 55FF	256
	HSI8M Trim		0x1FFF 5600 - 0x1FFF 56FF	256
	Factory config. bytes		0x1FFF 5700 - 0x1FFF 57FF	256
	UID		0x1FFF 5800 - 0x1FFF 58FF	256
	Factory config. bytes		0x1FFF 5900 - 0x1FFF 59FF	256
	Reserve		0x1FFF 5A00 - 0x1FFF 5FFF	1.5 K

### 3.2.2. 闪存读操作和访问延迟

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

表 3-2 根据 CPU 时钟 (HCLK) 频率的等待状态数

HCLK(MHz)	Wait states(Latency)	Aclk high lv width	Aclk low lv width
Fhclk <= 28MHz	0 (1 HCLK cycles)	HCLK high level	HCLK low level
28MHz < Fhclk <= 60MHz	1 (2 HCLK cycles)	1*HCLK cycle	1*HCLK cycle
60MHz < Fhclk <= 90MHz	3 (4 HCLK cycles)	2*HCLK cycle	2*HCLK cycle
90MHz < Fhclk <= 120MHz	4 (5 HCLK cycles)	3*HCLK cycle	2*HCLK cycle

HCLK(MHz)	Wait states(Latency)	Aclk high lv width	Aclk low lv width
120MHz < Fhclk <= 140MHz	5 (6 HCLK cycles)	3*HCLK cycle	3*HCLK cycle
140MHz < Fhclk	6 (7 HCLK cycles)	4*HCLK cycle	3*HCLK cycle

注：对于 flash 读操作，aclk 有高低电平宽度均大于 15ns 的时序要求，hpre 采用门控的方式，分频后的时钟高电平宽度与分频前一致，因此即使分频后时钟频率与 latency 配置相符，但是可能会不满足 aclk 高低电平宽度要求，比如系统时钟源采用 PLL 输出 112M，HPRE4 分频后 HCLK 频率为 28M，满足频率要求，但是此时 aclk 的高电平宽度为 112M 时钟的高电平宽度（约 4.5ns），不满足要求。因此建议在 latency 为 0 的情况下，若使用 HPRE 分频，分频前时钟不能高于 60MHz。

在芯片复位后（包括上电复位，系统复位，退出 standby 复位），HCLK 时钟频率为 8MHz，Flash 读周期数为 0。

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

#### ■ 提高 CPU 频率时的操作步骤

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

#### ■ 降低 CPU 频率时的操作步骤

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

### 3.2.3. 自适应实时存储器加速器(ART Accelerator™)

专有的自适应实时(ART)存储器加速器面向 ARM® Cortex™-M4 处理器进行了优化。该加速器很好地体现了 ARM Cortex M4F 的固有性能优势，克服了通常条件下，高速的处理器在运行中需要经常等待 Flash 读取的情况。

为了发挥处理器的全部性能，该加速器将实施指令预取队列和分支缓存，从而提高了 128 位 Flash 的程序执行速度。根据 CoreMark 基准测试，凭借 ART 加速器所获得的性能相当于 Flash 在 CPU 频率高达 160 MHz 时以 0 个等待周期执行程序。

#### ■ 指令预取

每个 Flash 读操作可读取 128 位，可以是 4 行 32 位指令，也可以是 8 行 16 位指令，具体取决于烧写在 Flash 中的程序。因此对于顺序执行的代码，至少需要 4 个 CPU 周期来执行前一次读取的 128 位指令行。在 CPU 请求当前指令行时，可使用 I-Code 总线的预取操作读取 Flash 中的下一个连续存放的 128 位指令行。可将 FLASH\_ACR 寄存器中的 PRFTEN 位置 1，来使能预取功能。当访问 Flash 至少需要一个等待周期时，此功能非常有用。

#### ■ 指令缓存存储器

为了减少因指令跳转而产生的时间损耗，可将 64 行 128 位的指令保存到指令缓存存储器中。可将 FLASH\_ACR 寄存器中的指令缓存使能 (ICEN) 位置 1，来使能这一特性。每当出现指令缺失（即请求的指令未存在于当前使用的指令行、预取指令行或指令缓存存储器中）时，系统会将新读取的行复制到指令缓存存储器中。如果 CPU 请求的指令已

存在于指令缓存区中，则无需任何延时即可立即获取。指令缓存存储器存满后，可采用 LRU（最近最少使用）策略确定指令缓存存储器中待替换的指令行。此特性非常适用于包含循环的代码。

### ■ 数据管理

在 CPU 流水线执行阶段，将通过 D-Code 总线访问 Flash 中的数据缓冲池。因此，直到提供了请求的数据后，CPU 流水线才会继续执行。为了减少因此而产生的时间损耗，通过 AHB 数据总线 D-Code 进行的访问优先于通过 AHB 指令总线 I-Code 进行的访问。如果频繁使用某些数据，可将 FLASH\_ACR 寄存器中的数据缓存使能 (DCEN) 位置 1，来使能数据缓存存储器。此特性的工作原理与指令缓存存储器类似，但保留的数据大小限制在 8 行 128 位/行以内。

## 3.2.4. 擦除和编程操作

通过 ICP 或者 IAP 可以对 flash 进行编程操作。

ICP：用来更新整个 Flash 存储器的内容，可以使用 SWD 协议或者 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，并消除了不必要的包处理或者 socketing。

IAP：可以使用芯片支持的通讯接口，下载要编程的数据到 flash 中。 IAP 允许用户在应用运行时，再次编程 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 flash 编程或擦除操作时，发生了复位，则 Flash 存储器的内容是不被保护的。在编程或者擦除操作期间，任何读 flash 的操作都会拖延总线。编程或擦除操作一结束，读操作就可以正确的进行。这也就意味着，当正在进行编程或擦除操作时，不能进行代码和数据的读取。

对于编程和擦除操作，必须打开 HSI。且建议用户在默认时钟下进行编程和擦除操作，防止出现问题。

### ■ 闪存解锁

在复位后，Flash 控制寄存器 (FLASH\_CR) 不允许执行写操作，以防止因电气干扰等原因出现对 Flash 的意外操作。每次对 Flash 的擦除和编程操作，都必须通过写 FLASH\_KEYR 来 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 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 Hard Fault 中断。这样的错误包括第一个写周期的 KEY1 不匹配，或者 KEY1 匹配，但第二个写周期的 KEY2 不匹配。

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

注意：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 总线的拖延，直到 BSY 位被清零。

## 3.2.5. 闪存擦除操作

Flash 擦除操作支持 page,sector,mass erase，执行 mass erase 时不会对 information memory 起作用。

### ■ 闪存页擦除

Page 擦除用来对 256bytes 的 main flash 进行擦除操作，但对 information 区不起作用。当某个 page 被 WRP 保护，它是不会被擦除的，此时 WRPER 位被置位。

page erase 操作的具体步骤如下：

- 1) 检查 FLASH\_SR 寄存器 BSY 位，确认没有正在进行的 flash 操作
- 2) 向 FLASH\_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH\_CR 寄存器的保护
- 3) 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

#### ■ 闪存扇区擦除

Sector erase 用来对 2Kbytes 的 main flash 进行擦除操作，但对 information 区不起作用。当某个 sector 被 WRP 保护，它是不会被擦除的，此时 WRPERR 位被置位。

sector erase 操作的具体步骤如下：

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

#### ■ 闪存块擦除

Block erase 用来对 32Kbytes 的 main flash 进行 erase 操作，但对 information 区不起作用。当某个 Block 被 WRP 保护，它是不会被擦除的，此时 WRPERR 位被置位。

block erase 操作的具体步骤如下：

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

#### ■ 闪存片擦除

Mass erase 用来对整片 main flash 进行 erase 操作，但对 information 区不起作用。当 WRP 被使能，mass erase 功能无效，不会产生 mass erase 操作，并且 WRPERR 位被置位。

mass erase 操作的具体步骤如下：

- 1) 检查 BSY 位，确认是否没有正在进行的 Flash 操作
- 2) 向 FLASH\_KEYR 寄存器依次写 KEY1,KEY2，解除 FLASH\_CR 寄存器保护
- 3) 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（32bit 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

### 3.2.6. 闪存写操作

Flash 存储器每次以 32-bit(word)为单位（进行 half word 或者 byte 操作会产生 hardfault）进行整个 page 的编程操作。当 FLASH\_CR 寄存器的 PG 位被置位，CPU 向 FLASH 存储器地址空间写 32-bit 数据时，编程操作开始启动。

如果要编程的 flash 地址空间，是被 FLASH\_WRPTR 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。编程操作的结束，FLASH\_SR 寄存器的 EOP 位会被置位。

具体 flash 编程的操作步骤如下所示：

- 1) 检查 FLASH\_SR 寄存器的 BSY 位，判断是否当前没有正在继续的 flash 操作
- 2) 如果没有正在进行的 flash erase 或者 编程操作，则软件读出该 Page 的 64 个 word (如果需要保留该 page 已有的数据，则进行该步骤，否则跳过该步骤)
- 3) 向 FLASH\_KEYR 寄存器依次写 KEY1 和 KEY2，解除 FLASH\_CR 寄存器的保护
- 4) 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 编程操作 (只接受 32-bit 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 编程操作已经成功，该位被置位)，然后软件清零该位
- 10) 如果不再有 编程操作，则软件清除 PG 位当上述步骤 7) 成功执行，则 编程操作自动启动，同时 BSY 位被硬件置位。

### 3.2.7. Flash 选项字节

#### 3.2.7.1. Flash 选项字节描述

芯片内的 flash 的 information 区域的部分区间作为 Option byte 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， watchdog 可以选择为硬件或者软件模式。

为了数据的安全性， option byte 以正文及反码形式分别存储。

表 3-3 Option byte format

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

Option byte 的内容可以从表 Option byte organization 所述的存储器地址读到，也可以从以下 Option byte 的相关寄存器读到：

- FLASH user option 寄存器 (FLASH\_OPTR)
- FLASH WRP address 寄存器 (FLASH\_WPR)

表 3-4 Option byte organization

Word Address	Description
0xFFFF 5000	Option byte for Flash User option and its complemented
0xFFFF 5004	Reserved
0xFFFF 5008	Reserved
0xFFFF 500C	Option byte for Flash WRP address and its complemented
0xFFFF 5010	Reserved
0xFFFF 5014	Reserved
...	Reserved
0xFFFF 50FF	Reserved

#### Option byte for Flash User option

Flash memory address: 0xFFFF 5000

Production value: 0x8F55 70AA

在上电复位 (POR/OBL\_LAUNCH) 释放后，从 flash information memory 的 option bytes 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	~nRST_STDBY	~nRST_STOP	~IWDG_SW	Res				~RDP[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
Res	nRST_STDBY	nRST_STOP	IWDG_SW	Res				RDP[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	Reserved	RES	
30	~nRST_STDBY	R	NRST_STDBY 的反码
29	~nRST_STOP	R	NRST_STOP 的反码
28	~IWDG_SW	R	IWDG_SW 的反码
27: 25	Reserved	RES	
24	Reserved	RES	
23: 16	~RDP	R	RDP 的反码
15	Reserved	RES	
14	nRST_STDBY	R	0: Reset generated when entering Standby mode 1: No reset generated
13	nRST_STOP	R	0: Reset generated when entering Stop mode 1: No reset generated
12	IWDG_SW	R	0: 硬件 watchdog 1: 软件 watchdog
11: 9	Reserved	RES	
8	Reserved	RES	
7: 0	RDP	R	0xAA: level 0, read protection inactive 非 0xAA: level 1, read protection active

#### Option byte for Flash WRP address

Flash memory address: 0x1FFF 500C

Production value: 0xF000 0FFF

在上电复位 (POR/OBL\_LAUNCH) 释放后, 从 flash information memory 的 option bytes 区域读出相应的值, 写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Function
27: 16	Complemented WRP	R	WRP 的反码
11: 0	WRP	R	0: block[y]被保护 1: block[y]无保护 y=0~11

#### 3.2.7.2. Flash 选项字节写

复位后, FLASH\_CR 寄存器中与 Option byte 相关的位是被写保护的。当对 option byte 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 unlock 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 Hard Fault 中断。

User option (information flash 的 option bytes) 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 erase/program 操作。

如果软件置位 Lock 位，则 OPTLOCK 位也被自动置位。

#### **Modifying user option bytes**

Option bytes 的 program 操作，跟对 Main flash 的操作不一样。为修改 option bytes，需要进行如下步骤：

- 1) 用之前描述的步骤，清零 OPTLOCK 位
- 2) 检查 BSY 位，确认没有正在进行的 Flash 操作
- 3) 向 option bytes 寄存器 FLASH\_OPTR/ FLASH\_WRPR 写期望的值（1~2 个 word）
- 4) 置位 OPTSTRT 位
- 5) 向 main flash 任意地址写任意 32-bit 数据（触发正式的写操作）
- 6) 等待 BSY 位被清零
- 7) 等待 EOP 拉高，软件清零

任何对 Option bytes 的改动，硬件都会先把 option byte 对应的整个 page erase 掉，然后用 FLASH\_OPTR 或者 FLASH\_WRPR 寄存器的值，写到 option bytes 中。并且，硬件自动计算相应的补码，并把计算值写到 option bytes 的相应区域。

#### **Option byte loading**

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

Option bytes 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)

“装载 option bytes” 进行的操作是：对 information memory 区域的 option bytes 进行读操作，再把读出的数据存储在内部 option 寄存器中（FLASH\_OPTR 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 option bytes 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 half word）有相应的补码。在 option bytes 装载期间，会对 option bit 和其补码的验证，这能确保装载被正确的进行了。

如果正补码匹配，则 option bytes 被 copy 到 option 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。不匹配的值被写入 option 寄存器：

- 对于 user option
  - 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，option bytes 的内容被 copy 到下面的 option 寄存器（软件可读可写）：

### **3.2.8. Flash 配置字节**

芯片内的 flash 的 information 区域的部分区间作为 Factory byte 使用，用来存放芯片的 TRIM,CP,FT 等信息。

#### **3.2.8.1. 闪存保护**

对 Flash main memory 的保护包括以下几种机制：

- read protection(RDP)，防止来自外部的访问。
- write protection (WRP) 控制，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 32Kbytes。
- Option byte 写保护，专门的解锁设计。

### 3.2.8.2. 闪存读保护(RDP)

通过设置 RDP option byte，并进行 system reset(POR 或者 OBL 复位)装载新的 RDP option byte，可以激活读保护功能。RDP 保护 flash main memory、backup registers。

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

当 RDP option byte 和补码成对正确存在于 option byte 时，Flash memory 会被保护。

表 3-5 flash memory read protection status

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
除[0xAA,0x55]组合的任何值		Level 1

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

#### Level 0: no protection

对 main flash 的读、编程和擦除操作是可以的，对 option byte 也是可以进行任何操作。

#### Level 1: Read protection

当 option byte 里的 RDP 及其补码包含任何除了[0xAA,0x55]之外的组合，则 level 1 read protection 生效，Level 1 是缺省的保护级别。

- User mode: 在用户模式下执行的程序 (boot from main flash)，可以对 main flash、option byte 进行所有操作。
- Debug, boot from SRAM, and boot from system memory modes: 在 debug 模式，或者当从 SRAM 或 system memory 启动，main flash 是不能被访问的。在这些模式下，对 main flash 读或者写访问产生一个 bus error，以及产生一个 hard fault 中断。

#### Changing the Read protection level

读保护级别可以改变：

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

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

#### RDP protection summary

To be Accessed Area	READ Protection level	Boot or executed From Main Flash			Debug/ Boot or executed From SRAM/ Boot or executed From System memory			DMA		
		Read	Write	Erase	Read	Write	Erase	Read	Write	Erase
Main Flash	0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	1	Yes	Yes	Yes	No	No	No	No	N/A	N/A
Information Block	x	Yes	No	No	Yes	No	No	Yes	N/A	N/A
		Yes	No	No	Yes	No	No	Yes	N/A	N/A
Option bytes (registers)	x	Yes	Yes	-	Yes	Yes	-	Yes	Yes	-
		Yes	Yes	-	Yes	Yes	-	Yes	Yes	-

图 3-2 访问状态与保护级别和执行模式

注意:

- 任何区域发起的 mass erase 指令都会 erase 掉 Main Flash。
- Information Block 是只读可访问的，无论保护级别和执行模式。
- RDP 从 Level 1 修改为 Level 0，会触发硬件对 main flash 的 mass erase。

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

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

### 3.2.8.3. 闪存写保护 (WRP)

Flash 可以被设置成写保护，以应对不需要的写操作。定义 WRP 寄存器每 bit 的控制粒度为 32Kbytes 的写保护 (WRP) 区域，即 1 个 block 大小。具体参见 WRP 寄存器的描述。

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

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

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

### 3.2.9. 闪存中断

表 3-6 FLASH 中断请求

Interrupt event	Event flag	Event flag/Interrupt clearing method	Enable control bit
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE

以下事件没有单独的中断标识，但会产生 Hard fault：

- Unlock flash memory 的 FLASH\_CR 寄存器的序列错误
- Unlock flash option bytes 的写操作序列错误
- FLASH program,erase 操作未进行 32 位数据的对齐
- 对 option byte 寄存器的写操作未进行 32 位数据的对齐

## 3.3. 寄存器描述 (基址 0x4002\_2000)

### 3.3.1. FLASH access control register(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.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
												RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	RES	-	Reserved
3:0	LATECY[3:0]	RW	0	<p>Flash 读操作对应的等待状态： SYSCLK &lt;= 28MHz 0: flash 读操作没有等待状态，即每次读 flash 需要 1 个系统时钟周期 28MHz &lt; SYSCLK &lt;= 60MHz 1: flash 读操作有 1 个等待状态，即每次读 flash 需要 2 个系统时钟周期 60MHz &lt; SYSCLK &lt;= 90MHz 3: flash 读操作有 3 个等待状态，即每次读 flash 需要 4 个系统时钟周期 90MHz &lt; SYSCLK &lt;= 120MHz 4: flash 读操作有 4 个等待状态，即每次读 flash 需要 5 个系统时钟周期 120MHz &lt; SYSCLK &lt;= 140MHz 5: flash 读操作有 5 个等待状态，即每次读 flash 需要 6 个系统时钟周期 140MHz &lt; SYSCLK 6: flash 读操作有 6 个等待状态，即每次读 flash 需要 7 个系统时钟周期 注：在配置 latency 位时，该寄存器其他位不能写入任何值</p>

### 3.3.2. FLASH key register(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	0x0000	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/erase 操作 KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

### 3.3.3. FLASH option key register(FLASH\_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

所有寄存器位是 write-only，读出返回 0。

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

Bit	Name	R/W	Reset Value	Function
31:0	OPTKEY[31:0]	W	0x0000	下面的值必须被连续的写入，才能 unlock flash 的 option 寄存器，并使能了 option byte 的 program/erase 操作 KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

### 3.3.4. FLASH status register(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
OPTVERR	Res.	WRPERR				EOP									
RC_W1											RC_W1				RC_W1

Bit	Name	R/W	Reset Value	Function
31:17	Reserved	RES	-	Reserved
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 and trimming bits loading validity error 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 option bytes，被强制成安全值，参考 section FLASH option byte programming。 软件写 1，清零。
14:5	Reserved	RES	-	Reserved
4	WRPERR	RC_W1	0	Write protection error

Bit	Name	R/W	Reset Value	Function
				当要被 program/erase 的地址处于被写保护的 flash 区域时 (WRP) , 硬件置位该位。 写 1, 清零该位。
3:1	Reserved	RES	-	Reserved
0	EOP	RC_W1	0	当 flash 的 program/erase 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。 写 1, 清零该位。

### 3.3.5. FLASH control register(FLASH\_CR)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	Res.	OBL_LAUN CH	Res.	ERR IE	EOP IE	Res.	Res.	Res.	Res.	PG STRT	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.	BER	SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	PER	PG	
			RW	RW								RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	Lock	RS	0	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/erase 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	0	Option bytes Lock 位。 软件对该位只能置位。当置位后, FLASH_CR 寄存器中与 option bytes 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/erase 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	Reserved		-	
27	OBL_LAUNCH	RC_W1	0	Force the option bytes loading。 当置位时, 该位强制系统进行 option bytes 的重装载。该位仅当 option byte 装载被完成后被硬件清零。如果 OPTLOCK 位被置位, 该位不能被写。 0: Option byte loading 完成 1: 产生 Option byte loading 请求, 系统产生复位, 进行 option byte 的重装载。
26	Reserved		-	
25	ERRIE	RW	0	Error interrupt enable 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 1: 有中断产生
24	EOPIE	RW	0	End of operation interrupt enable 当 FLASH_SR 寄存器的 EOP 位被置位, 该位使能中断的产生。 0: EOP 中断关闭 1: EOP 中断使能
23: 18	Reserved	RW	0	

Bit	Name	R/W	Reset Value	Function
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	Reserved		-	
17	OPTSTRT	RW	0	Flash option bytes 修改的启动位 该位启动了对 option bytes 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 option bytes 进行修改时，硬件自动把整个 128Bytes 的 page 进行 erase 操作，再进行 program 操作，其中也包括自动进行补码的写入。
16:13	Reserved		-	
12	BER	RW	0	Block erase 操作 0: 未选择 flash 的 block erase 操作 1: 选择 flash 的 block erases 操作
11	SER	RW	0	Sector erase 操作 0: 未选择 flash 的 sector erase 操作 1: 选择 flash 的 sector erases 操作
10: 3	Reserved		-	
2	MER	RW	0	Mass erase 操作 0: 未选择 flash 的 mass erase 操作 1: 选择 flash 的 mass erases 操作
1	PER	RW	0	Page erase 操作 0: 未选择 flash 的 page erase 操作 1: 选择 flash 的 page erase 操作
0	PG	RW	0	Program 操作 0: 未选择 flash 的 program 操作 1: 选择 flash 的 program 操作

### 3.3.6. FLASH option register(FLASH\_OPTR)

Address offset: 0x20

Reset value: 0x0000 XXXX。

在上电复位 (POR/OBL\_LAUNCH) 释放后，从 flash information memory 的 option bytes 区域读出相应的值，写入到该寄存器相应的 option bit。

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved			
15	Reserved			
14	nRST_STDBY	RW		0: Reset generated when entering Standby mode 1: No reset generated
13	nRST_STOP	RW		0: Reset generated when entering Stop mode 1: No reset generated
12	IWDG_SW	RW		0: 硬件 watchdog

Bit	Name	R/W	Reset Value	Function
				1: 软件 watchdog
11:8	Reserved	RW		
7:0	RDP	RW		0xAA: level 0, read protection 无效 非 0xAA: level 1, read protection 有效

### 3.3.7. FLASH WRP register(FLASH\_WRP)

Address offset:0x2C

Reset value: 0x0000 XXXX。

在上电复位 (POR/OBL\_LAUNCH) 释放后,从 flash information memory 的 option bytes 区域读出相应的值, 写入到该寄存器相应的

31	30	29	28	27	26	25	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.	WRP[11:0]											
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 12	Reserved			
11: 0	WRP[11:0]	RW		Write protection, 每一位对应一个 block 0: block N, 写保护有效 1: block N, 写保护无效 N=0-11

### 3.3.8. FLASH sleep time config register(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.	SLEEP_EN						
RW	RW	RW	RW	RW	RW	RW	RW								RW

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved			
15: 8	SLEEP_TIME[7:0]	RW	0x64	FLASH sleep time configuration(counter based on HSI_10M) 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: $t_{HSI\_10M} * SLEEP\_TIME$ Note: $t_{HSI\_10M}$ 为 HSI_10M 的周期; 为确保 Flash 功能的正确, 本寄存器最大设定值推荐设定为 0x28。 此外, 为提升易用性, 该寄存器的设定值出厂后存放在 0xFFFF XXX 中。
7: 1	Reserved			
0	SLEEP_EN	RW	0x0	FLASH Sleep enable 1: enable flash sleep

Bit	Name		R/W	Reset Value	Function											
					0: disable flash sleep											

### 3.3.9. FLASH TS0 register(FLASH\_TS0)

Address offset: 0x100

Reset value: 0x0000 003C

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.	TS0														
								RW							

Bit	Name		R/W	Reset Value	Function											
31: 8	Reserved															
7: 0	TS0		R	0x3C	TS0 = 7.5*Freq_HSI8M											

### 3.3.10. FLASH TS1 register(FLASH\_TS1)

Address offset: 0x104

Reset value: 0x0000 0090

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.	TS1														
								RW							

Bit	Name		R/W	Reset Value	Function											
31: 8	Reserved															
7: 0	TS1		R	0x90	TS1 = 18*Freq_HSI8M											

### 3.3.11. FLASH TS2P register(FLASH\_TS2P)

Address offset: 0x108

Reset value: 0x0000 003C

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.	TS2P														
								RW							

Bit	Name		R/W	Reset Value	Function											
31: 8	Reserved															
7: 0	TS2P		R	0x3C	TS2P = 7.5*Freq_HSI8M											

### 3.3.12. FLASH TPS3 register(FLASH\_TPS3)

Address offset: 0x10C

Reset value: 0x0000 0240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TPS3									
						RW									

Bit	Name	R/W	Reset Value	Function
31: 11	Reserved			
10: 0	TPS3	R	0x240	TPS3 = 72*Freq_HSI8M

### 3.3.13. FLASH TS3 register(FLASH\_TS3)

Address offset: 0x110

Reset value: 0x0000 003C

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.															
								RW							

Bit	Name	R/W	Reset Value	Function
31: 8	Reserved			
7: 0	TS1	R	0x3C	TS1 = 7.5*Freq_HSI8M

### 3.3.14. FLASH ERASE TPE register(FLASH\_ERSTPE)

Address offset: 0x114

Reset value: 0x0000 4E20

31	30	29	28	27	26	25	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
ERSTPE															
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	Reserved			
15: 0	ERSTPE	R	0x5DC0	ERSTPE = 3000*Freq_HSI8M

### 3.3.15. FLASH PROGRAM TPE register(FLASH\_PRGTPE)

Address offset: 0x118

Reset value: 0x0000 2EE0

31	30	29	28	27	26	25	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	Reserved			
15: 0	PRGTPE	R	0x1F40	PRGTPE = 1000*Freq_HSI8M

### 3.3.16. FLASH PRE-PROGRAM TPE register (FLASH\_PRETPE)

Address offset: 0x11C

Reset value: 0x0000 0640

31	30	29	28	27	26	25	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.													
			RW												
PRETPE															

Bit	Name	R/W	Reset Value	Function
31: 13	Reserved			
12: 0	PRETPE	R	0x640	PRETPE = 200*Freq_HSI8M

### 3.3.17. FLASH 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLAS_H_AC_R	Reserved																									LATENCY						
	Read /Writ e																										RW						
	Reset Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x008	FLAS_H_KE_Y_R	KEY																															
	Read /Writ e	W																															
	Reset Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C0	FLAS_H_OPTKEY_R	OPTKEY																															
	Read /Writ e	W																															
	Reset Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x110	FLAS_H_SR	Reserved												B	S	Y	O	P	T	V	E	R	R	W	RR	W	RR	EOP					
	Read /Writ e													R	C	W	R	C	W	1	1	W	1	W	1	W	1	RC					
	Reset Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x114	FLAS_H_CCR	L_O_C_K	O_P_T_L_O_C_K	Re-serv ed	O_B_L_A_U_N	R_e_ser_v	E_R_R_I_E	E_O_P_I_E	Reserved				P_G_S_T_R_T	R_e_ser_v	O_P_T_S_T_R_T	R_e_ser_v	Reserved				B_E_R	S_E_R	Reserved				M_E_R	P_E_R	P_G				

Offset	Register	31	30	29	28	27	C H	e d	26	25	24	23	22	21	20	19	e d	18	17	e d	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Re ad /W rit e	R S	R S			R C _W 1	R W	R W				R W	R W	R W	R W	R W	R W				R W	R W							R W	R W	R W						
	Re set Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0 x 2 0	FL AS H_ O PT R	Reserved																N b o ot 1	N R S T _M O D E	W W D G _S W	I W D G _S W	Reserved				RDP											
	Re ad /W rit e	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W							
	Re set Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
0 x 2 C	FL AS H_ W R P R	Reserved																Reserved				WRP															
	Re ad /W rit e	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W						
	Re set Va lue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

## 4. 电源控制 (PWR)

### 4.1. 功能描述

#### 4.1.1. 电源结构

电源结构如下图所示：

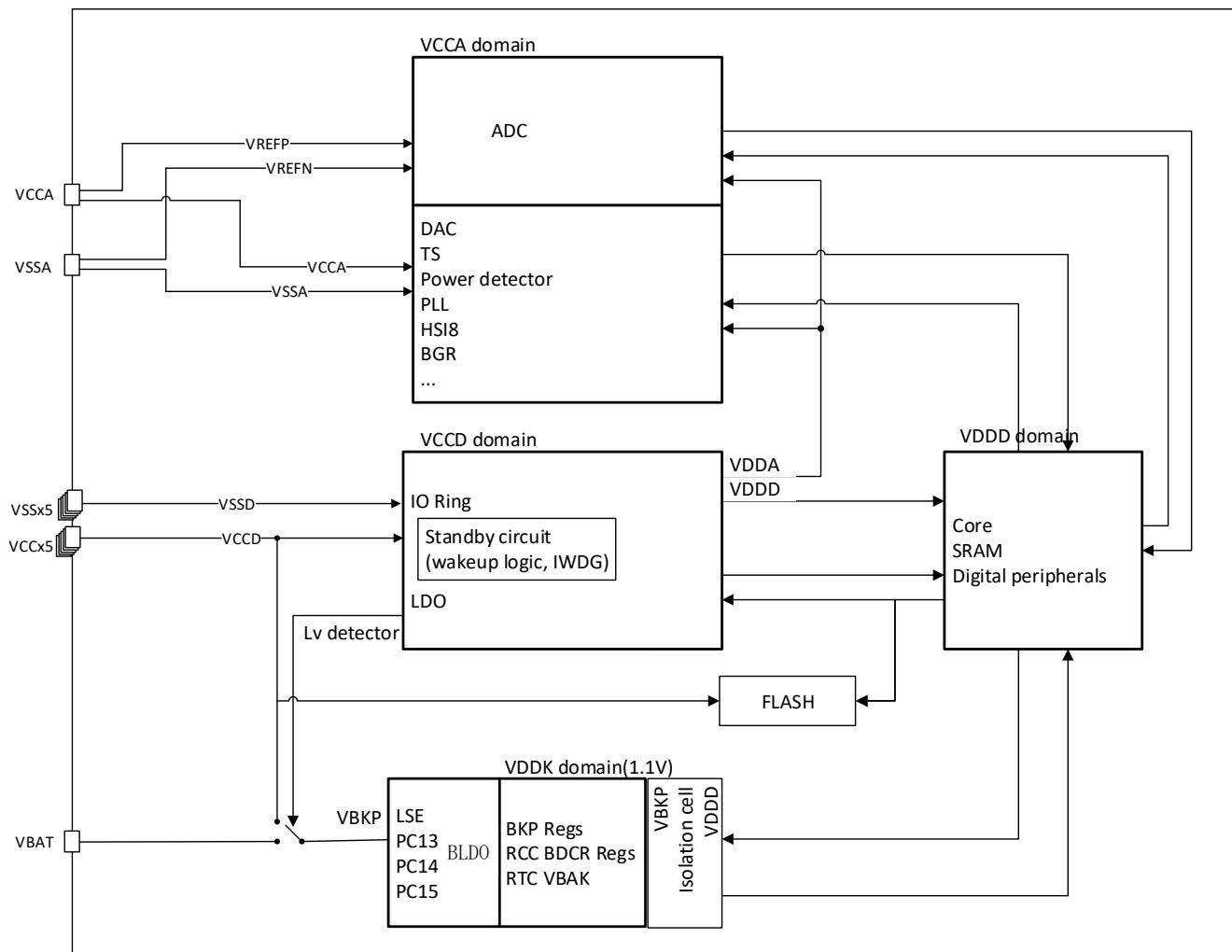


图 4-1 电源结构

表 4-1 电源结构

编号	电源	电源值	描述
1	VCC	1.7v~3.6v	通过电源管脚为芯片提供电源，其供电模块为：部分模拟 IP 和 IO 电路，待机模式控制逻辑。
2	VCCA	1.7v~3.6v	给大部分模拟模块及大部分 IO 供电，来自于 VCCA PAD。
4	VDDx (VDDD/VDDA)	1.1v/1.0v/0.9v/0.8v±10%	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 1.1v。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 1.1V/1.0V/0.9V/0.8V。

#### 4.1.1.1. 系统主要电源

##### 独立 A/D 转换器电源和参考电压

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

- ADC 电压源从单独的 VCCA 引脚输入。
- VSSA 引脚提供了独立的电源接地连接。

为了提高低压输入的精度，用户可以在 VREF 上连接单独的 ADC 外部参考电压输入。VREF 上的电压范围为 1.7 V 到 VDDA。

### 备份电池域

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

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

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

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

注意：

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

如果连接到 VBAT 引脚的电源/电池不能支持此电流注入，强烈建议在此电源和 VBAT 引脚之间连接一个外部低降二极管。

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

通过 VCCD 对备份域供电时（模拟开关连接到 VCCD），可实现以下引脚：

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

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

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

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

### 备份域访问

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

- 将 RCC\_APB1ENR 寄存器中的 PWREN 位置 1，使能电源接口时钟
- 将 PWR\_CR 寄存器中的 DBP 位置 1，使能对备份域的访问
- 配置 RCC 备份域控制寄存器 (RCC\_BDCR) 中的 RTCSEL 寄存器选择 RTC 时钟源
- 配置 RCC 备份域控制寄存器 (RCC\_BDCR) 中的 RTCEN，使能 RTC 时钟

### VDDD 调压器

嵌入式线性 VR 为备份域和待机（唤醒）电路以外的所有数字电路供电。调压器输出电压默认为 1.1 V，并可配置输出 1.0V，0.9V，0.8V。

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

- 在 run 模式中，调压器为 1.1 V 域（内核、存储器和数字外设）提供全功率。在该模式下，调压器输出电压可通过软件调节为不同的电压值（可通过 PWR\_CR 寄存器的 VOS[1:0]位配置）。
- 在停止模式下，主调压器或低功耗调压器为 VDDD 域提供低功率电压，从而保存寄存器和内部 SRAM 的内容。还可以将调压器置于主调压器模式 (MR) 或低功耗模式 (LPR)。
- 在待机模式中，调压器掉电。除待机（唤醒）电路和备份域外，寄存器和 SRAM 的内容都将丢失。

#### VBKP 调压器

嵌入式线性 VR 为备份域的所有数字电路供电。调压器输出电压默认为 1.1 V，并可配置输出 0.9V, 0.85V, 0.8V。

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

- 在 run 模式和低功耗模式下，调压器为 VDDK 域提供全功率；
- 在各种模式下，始终工作；

#### 4.1.1.2. 电源检测

##### POR/PDR

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

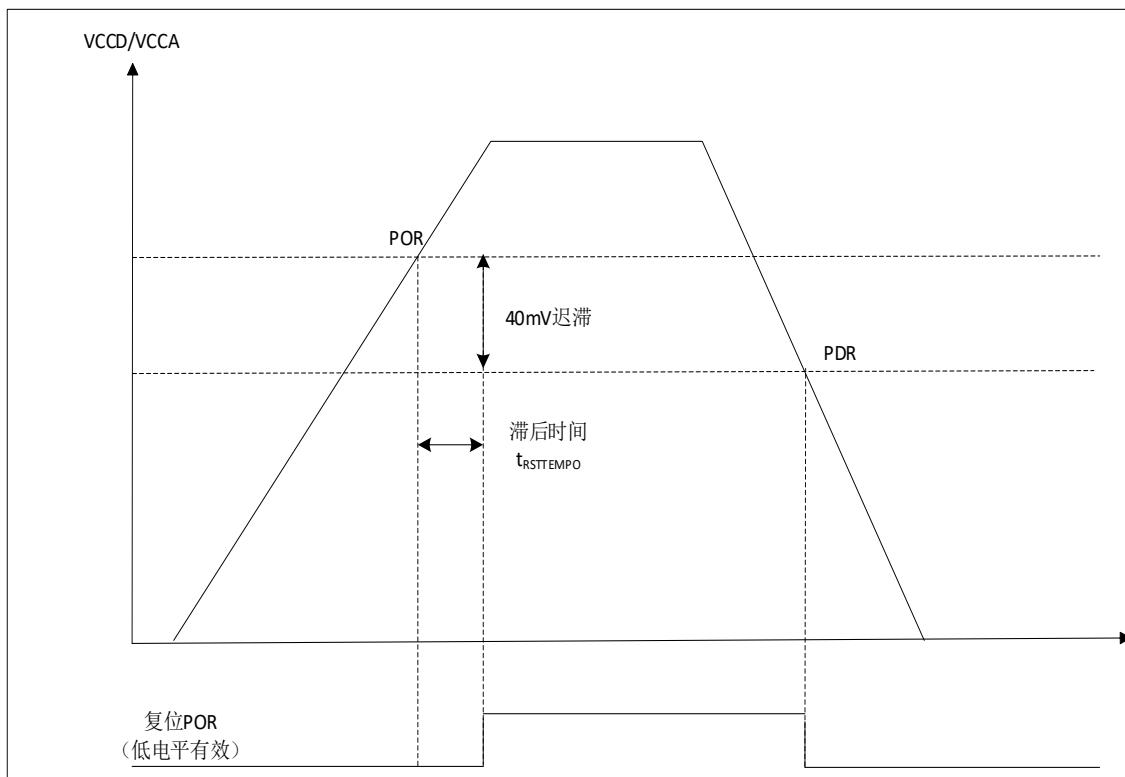


图 4-2 POR/PDR 时序

##### PVD 模块

PVD 模块是检测 VCC 是否低于 PWR\_CR.PLS 设置的阈值。

通过配置 PVDE 寄存器使能 PVD 功能。

电源控制/状态寄存器(PWR\_CSR)中的 PVDO 标志用来表明 VCC 是高于还是低于 PVD 的电压阈值。该事件在内部连接到 EXTI 的 Line16，如果该中断在外部中断寄存器中是使能的，该事件就会产生中断。当 VCC 下降到 PVD 阈值以下和(或)当 VCC 上升到 PVD 阈值之上时，根据 EXTI Line16 的上升/下降沿触发设置，就会产生 PVD 中断。实际应用中，这一特性可用于用于执行紧急关闭任务。

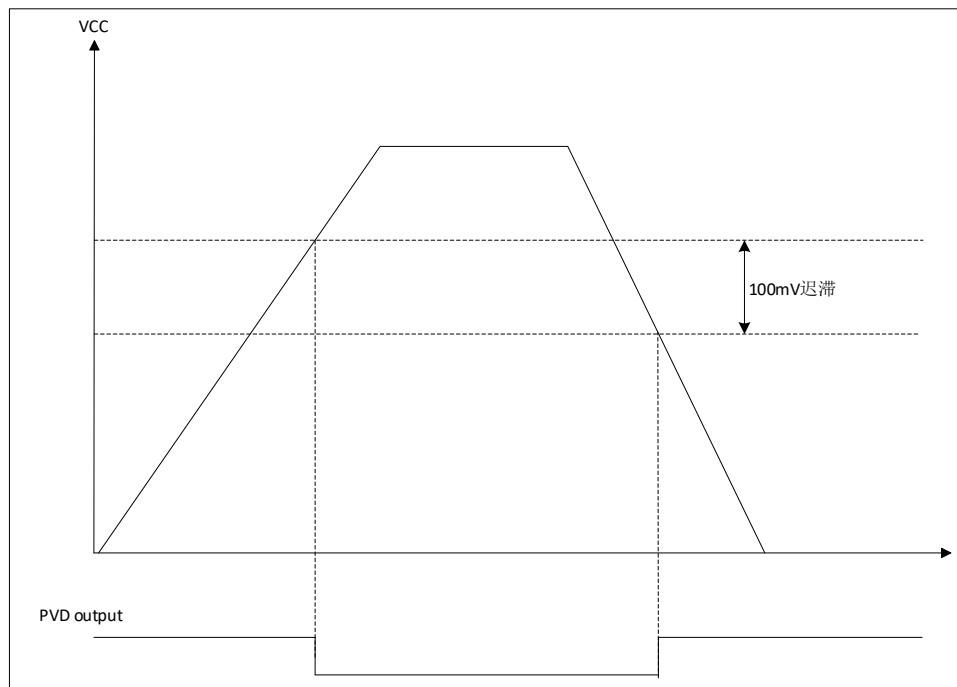


图 4-3 PVD 时序

#### 4.1.2. 系统低功耗模式

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

器件有三种低功耗模式：

- 睡眠模式（带 FPU 的 Cortex®-M4F 内核停止工作，外设包括 Cortex®-M4 的外设（NVIC, Sys Tick）等可以配置为运行）
- 停止模式（HSI48M, HSI8M, HSE 和 PLL 时钟都停止）
- 待机模式（VDDD 域断电）

芯片存在 VBAT 电源，所以在 VCC 掉电时，芯片仅 VBKP 域工作。

此外，可通过下列方法之一降低运行模式的功耗：

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

##### 4.1.2.1. 进入低功耗模式

MCU 通过执行 WFI（等待中断）或 WFE（等待事件）指令进入低功耗模式，也可通过在从 ISR 返回时将带 FPU 的 Cortex®-M4F 系统控制寄存器中的 SLEEPONEXIT 位置 1 进入低功耗模式。

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

在进入低功耗模式的进程中，出现唤醒信号，则退出低功耗进入流程。

##### 4.1.2.2. 退出低功耗模式

MCU 根据进入睡眠和停止模式的方式退出这两种低功耗模式：

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

➤ NVIC IRQ 中断：

在外设控制寄存器中使能中断，而不是在 NVIC 中使能，并且使能 Cortex®-M4F 系统控制寄存器中 SEVONPEND=1。当 MCU 从 WFE 恢复时，必须清零外设中断挂起位和外设的 NVIC 中断通道挂起位（在 NVIC 中断清零挂起寄存器中）。

➤ 事件：

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

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

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

#### 4.1.2.3. 低功耗模式总结

表 4-2 低功耗模式总结

模式	进入条件	退出条件	唤醒延时	VDDD 域状态	VCCD 域状态	VDDK 域状态	IO 状态	其它
睡眠模式 1、SLEEPONEXIT=0 时执行 WFI 或者 WFE 命令时，立即进入睡眠模式； 2、SLEEPONEXIT=1 时，系统从最低优先级的中断处理程序中退出时，CPU 就立即进入睡眠模式。	执行 WFI 或者从 ISR 返回且 SLEEPING=1，SLEEPDEEP=0	任一中断	无延时	处理器核时钟关闭；其他时钟根据配置开关；	待机模式唤醒模块不工作；IWDG 是否工作由 CPU 配置；TrimReg 模块工作；	正常工作；LSE/RTC/BKP 是否工作由 CPU 配置；TrimReg 模块工作；	保持原运行模式状态	NA
	执行 WFE 且 SLEEPING=1，SLEEPDEEP=0	唤醒事件或者 SEVONPEND 为 1 时的中断	无延时					NA
停止模式	满足以下条件时，执行 WFE 或者 WFI 或者从 ISR 返回： 1、SLEEPDEEP=1； 2、PDDS=0	中断或唤醒事件产生 WFI 进入：任何配置为停止模式中断唤醒的 EXTI line； WFE 进入：任何配置为停止模式事件唤醒的 EXTI line； IWDG 复位； NRST 复位；	HSI8M 唤醒时间 + (LDO 从低功耗模式退出时间) + FLASH 唤醒的时间 (根据配置时间可有重合)	关闭 HSI8M, HSI48M, PLL, HSE； SRAM 进入 retention； Flash 进入 sleep； LPDS=0：正常打开； LPDS=1：低功耗模式打开	待机模式唤醒模块不工作；IWDG 是否工作由 CPU 配置；TrimReg 模块工作；	正常工作；LSE/RTC/BKP 是否工作由 CPU 配置；TrimReg 模块工作；	保持原运行模式状态	如有需要 IWDG、RTC、LSI、LSE32K 需要在进入停止模式之前分别使能或者禁止

模式	进入条件	退出条件	唤醒延时	VDDD 域状态	VCCD 域状态	VDDK 域状态	IO 状态	其它
待机模式	满足以下条件时，执行 WFE 或者 WFI 或者从 ISR 返回： 1.SLEEPDEEP=1； 2.PDDS=1； 3.清除电源控制/状态寄存器(PWR_CSR)中的 WUF 位	WKUP 引脚的上升沿(使能时)； RTC 闹钟事件的上升沿； NRST 引脚上外部复位； IWDG 复位；	包括 LDO/HSI8M01 启动的复位过程	掉电； SRAM 内容丢失；	待机唤醒模块正常工作； IWDG 是否工作由 CPU 配置；	正常工作； LSE/RTC 是否工作由 CPU 配置；	除以下 IO 引脚外，均处于高阻态： 1.NRST 2.侵入检测或 RTC 校准输出 PIN 3.WKUP 功能对应的 IO 引脚 4. LSE 相关 IO	IWDG、RTC、LSE32K 可根据需要独立设置

进入睡眠模式之前，必须清零所有的中断挂起位。

#### 4.1.2.4. 睡眠模式

表 4-3 睡眠模式

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

#### 4.1.2.5. 停止模式

停止模式基于带 FPU 的 Cortex®-M4F SleepDeep 模式与外设时钟门控。VDDD 域 LDO 既可以配置为正常模式，也可以配置为低功耗模式。在停止模式下，VDDD 域的所有高频时钟都会停止，停止的时钟包括 PLL、HSI8M、HSI48M 和 HSE，停止顺序为 PLL->HSI48->HSE->HSI8。内部 SRAM 和寄存器内容将保留。

PWR\_CR 寄存器中的某些设置可进一步降低功耗。Flash 处于停止模式时，将器件从停止模式唤醒需要额外的启动延时。

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

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

在停止模式下，可以通过对各控制位进行编程来选择以下功能：

- 独立的看门狗 (IWDG)： IWDG 通过写入其密钥寄存器或使用硬件选项来启动。而且一旦启动便无法停止，除非复位。

- 实时时钟 (RTC): 通过 RCC 备份域控制寄存器 (RCC\_BDCR) 中的 RTCEN 位进行配置。
- 内部 RC 振荡器 (LSI RC): 通过 RCC 时钟控制和状态寄存器 (RCC\_CSR) 中的 LSION 位进行配置。
- 外部 32.768 kHz 振荡器 (LSE OSC): 通过 RCC 备份域控制寄存器 (RCC\_BDCR) 中的 LSEON 位进行配置。
- PVD: 通过电源控制寄存器 PWR\_CR 中的 PVDE 位进行配置。

退出停止模式，将选择 HSI8M 作为系统时钟。

表 4-4 停止模式

停止模式	说明
	WFI 或 WFE，并且： --没有中断（针对 WFI）或者事件（针对 WFE）挂起； --SLEEPDEEP=1； --将 PWR_CR 寄存器中的 PDDS 位清零； --配置 PWR_CR 寄存器中的 LPDS 选择 VR 工作模式；
进入模式	从最低优先级的 ISR 返回，并且： --SLEEPDEEP=1； --SLEEPONEXIT=1； --没有中断挂起； --将 PWR_CR 寄存器中的 PDDS 位清零； --配置 PWR_CR 寄存器中的 LPDS 选择 VR 工作模式；
	注：要进入停止模式，所有 EXTI line 挂起位 (EXTI_PR)、所有外设中断挂起位、RTC 闹钟标志位必须复位。否则将忽略进入停止模式，继续执行程序 (CPU 实现)。
退出模式	使用 WFI 或从 ISR 返回进入：任何配置为中断模式的 EXTI line (同时使能 NVIC 中对应的 EXTI 中断)。中断源为外部中断或者具有唤醒功能的外设产生的中断。 使用 WFE 进入且 SEVONPEND=0：任何配置为事件模式的 EXTI line。 使用 WFE 进入且 SEVONPEND=1： --任何配置为中断模式的 EXTI line (即使在 NVIC 中断对应的 EXTI 中断禁止)。中断源为外部中断或者具有唤醒功能的外设产生的中断。 --唤醒事件 NRST 复位 IWDG 复位
唤醒延迟	HSI8M 唤醒时间+ (LDO 从低功耗模式退出时间) +FLASH 唤醒的时间 (根据配置时间可有重合)

#### 4.1.2.6. 待机模式

待机模式下，VR 停止工作，VDDD 域断电。PLL, HSI8M, HSI48M, HSE 都关闭，关闭顺序为 PLL->HSI48->HSE->HSI8。除 VBAK 域 (RTC 寄存器和备份寄存器) 和待机电路的寄存器外，SRAM 和寄存器内容都会丢失。

在待机模式下，可以通过对各控制位进行编程来选择以下功能：

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

表 4-5 待机模式

待机模式	说明
	WFI 或 WFE，并且： --没有中断（针对 WFI）或者事件（针对 WFE）挂起； --SLEEPDEEP=1； --将 PWR_CR 寄存器中的 PDDS 位置位； --配置 PWR_CSR 寄存器中的 WUF 位清零（如果不清零，则会立即唤醒）； --将与选择唤醒源对应的标志清零；
进入模式	从 ISR 返回，并且： --SLEEPDEEP=1； --SLEEPONEXIT=1； --没有中断挂起； --将 PWR_CR 寄存器中的 PDDS 位置位； --将 PWR_CSR 寄存器中的 WUF 位清零（如果不清零，则会立即唤醒）； --将与选择唤醒源对应的标志清零；
退出模式	WKUPx 引脚 ( $x=1, 2, 3, 4, 5$ ) RTC 唤醒事件产生 NRST 引脚复位 IWDG 复位
唤醒延迟	HSI8M 唤醒时间+LDO 上电时间+LDO 等待时间（可配置 5us/10us/20us/30us）+FLASH 等待时间 3us（根据配置时间可有重合）

在待机模式下，除以下各部分外，其余 IO 都处于高阻态：

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

#### 4.1.2.7. 停止模式和待机模式的调试

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

#### 4.1.2.8. 低功耗模式下的自动唤醒 (AWU)

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的 MCU(自动唤醒模式)。 RTC 提供一个可编程的时间基数，用于周期性从停止或待机模式下唤醒。通过对备份区域控制寄存器(RCC\_BDCR)的 RTCSEL[1:0]位的编程，三个 RTC 时钟源中的二个时钟源可以选作实现此功能：

- 低功耗32.768kHz 外部晶振(LSE)  
该时钟源提供了一个低功耗且精确的时间基准。(在典型情形下消耗小于 1μA)
- 低功耗内部 RC 振荡器(LSI)

使用该时钟源，节省了一个 32.768kHz 晶振的成本。但是 RC 振荡器将少许增加电源消耗。为了用 RTC 闹钟事件将系统从停止模式下唤醒，必须进行如下操作：

- 配置 EXTI17为上升沿触发。
- 配置 RTC 使其可产生唤醒事件。

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

该外设的寄存器可以通过 half-word 或者 word 访问。

#### 4.1.2.9. 不同模式下功能

下表为不同模式下，各个功能模块是否存在的情况：

表 4-6 不同模式下功能

Peripheral	Run	Sleep	Stop		待机		VBKP/VDDK
			VR@LPR or VR@MR	Wakeup ability	-	Wakeup ability	
CPU Core	Y	-	-	-	-	-	-
Flash memory	Y	O (2)	- (3)	-	-	-	-
SRAM	Y	O (4)	- (5)	- (5)	-	-	-
PVD	O	O	O	O	-	-	-
ESMC	O	O	-	-	-	-	-
DMA	O	O	-	-	-	-	-
HSI	O	O	-	-	-	-	-
HSI48	O	O	-	-	-	-	-
HSE	O	O	-	-	-	-	-
LSI	O	O	O	-	O	-	-
LSE	O	O	O	-	O	-	O
PLL	O	O	-	-	-	-	-
HSE Clock Security System (CSS)	O	O	-	-	-	-	-
RTC/Auto Wakeup	O	O	O	O	O	O	O
Number of RTC TAMP pin	1	1	1	1	1	-	1
USB	O	O	-	-	-	-	-
USART1	O	O	-	-	-	-	-
USART2	O	O	-	-	-	-	-
USART3	O	O	-	-	-	-	-
USART4	O	O	-	-	-	-	-
USART5	O	O	-	-	-	-	-
I2C1	O	O	-	-	-	-	-
I2C2	O	O	-	-	-	-	-
SPI1	O	O	-	-	-	-	-
SPI2	O	O	-	-	-	-	-
SPI3	O	O	-	-	-	-	-
CAN	O	O	-	-	-	-	-
ADC1	O	O	-	-	-	-	-
ADC2	O	O	-	-	-	-	-
ADC3	O	O	-	-	-	-	-
Temperature sensor	O	O	-	-	-	-	-
Timers(TIM1/TIM3/TIM6/TIM14/TIM16/TIM17)	O	O	-	-	-	-	-
IWDG	O	O	O	O	O	O	-
WWDG	O	O	-	-	-	-	-
SysTick timer	O	O	-	-	-	-	-
CRC	O	O	-	-	-	-	-
GPIOs	O	O	O	O	O	5pins	3pins
ACC	O	O	-	-	-	-	-
BKP	O (5)	O (5)	O (5)	-	O (5)	-	O (6)

注：

- 1) Y = Yes (使能); O = Optional (默认关闭, 可以软件使能); - = Not available
- 2) 在睡眠模式, FLASH 控制器时钟可以被关闭。
- 3) 在停止模式, FLASH 不掉电, 但进入 Sleep 低功耗模式, 唤醒需要3us 等待。
- 4) 在睡眠模式, SRAM 的时钟可以被开或者关。
- 5) SRAM 不下电, 但无时钟提供, 进入最低功耗状态。
- 6) BKP 模块时钟有使能信号, 定义为 O

## 4.2. 寄存器描述 (基址 0x4000\_7000)

### 4.2.1. 电源控制寄存器(PWR\_CR)(0x00)

Address offset: 0x00

Reset value: 0x0011 0000(从待机模式唤醒时清零)

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	STDBY_MRRDY_WAIT		FLS_WUPT	HSION_CTRL.		
									RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS[1:0]	BKPVR_VOS[1:0]	Res	Res	DBP		PLS[2:0]		PVDE	CSBF	CWUF	PDDS	LPDS		
	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31:22	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
21:20	STDBY_MRRDY_WAIT[1:0]	RW	2'h1	待机模式唤醒, MR ready 后, 需要等待的时间。 00: 5us 01: 10us 10: 20us 11: 30us
19	Reserved	-	-	Reserved
18:17	FLS_WUPT[1:0]	RW	2'b00	停止模式唤醒时序中, 在 HSI8M 稳定后, 在 FLASH 操作前需要等待时间。 00: 3us; 01: 5us; 10: 2us; 11: 0us;
16	HSION_CTRL	RW	0	从停止模式唤醒时, HSI8M 打开时间控制。 0: 等待 MR 稳定后, 使能 HSI; 1: 与 VR 同时打开, 即唤醒时立刻使能 HSI。
15	Reserved	-	-	Reserved
14:13	VOS[1:0]	RW	2'b00	选择 VR 输出电压级别。 此位用来控制内部 VR 的输出电压, 以便实现器件与功耗的平衡。 当 VR 为 LP 模式时: 00: 1.1V 01: 1.0V 10: 0.9V 11: 0.8V 当 VR 为 MR 模式时: 0x: 1.1V 1x: 0.9V Note: 该信号对外 datasheet 不显示。
12:11	BKPVR_VOS[1:0]	RW	0	PMU BLDO 输出电压控制。 00: VDDK=1.1V 01: VDDK=0.9V 10: VDDK=0.85V 11: VDDK=0.8V 该信号在 VDDD 上电复位有效时为 0
10:9	Reserved	-	-	Reserved
8	DBP	RW	0	backup 域寄存器写保护。 0: 不可访问 RTC 和 RTC 备份寄存器; 1: 可以访问 RTC 和 RTC 备份寄存器;
7:5	PLS[2:0]	RW	0	PVD 电平选择。 由软件写入, 用于选择 PVD 的电压阈值。 000: VPVD0 (around 1.8V) 001: VPVD1 (around 2.0V) 010: VPVD2 (around 2.2V) 011: VPVD3 (around 2.4V) 100: VPVD4 (around 2.6V) 101: VPVD5 (around 2.8V) 110: VPVD6 (around 3.0V) 111: VPVD7 (around 3.2V)
4	PVDE	RW	0	PVD 使能。 0: 禁止 PVD; 1: 使能 PVD;
3	CSBF	W	0	待机标志清零。 0: No effect; 1: 将 SBF 标志位清零;
2	CWUF	W	0	将唤醒标志清零。 0: No effect; 1: 2 个系统时钟周期后将 WUF 标志清零;
1	PDDS	RW	0	掉电深度睡眠。

Bit	Name	R/W	Reset Value	Function
				<p>此位由软件置 1 和清零。与 LPDS 结合配置。</p> <p>0: CPU 进入 SLEEPDEEP 时系统进入停止模式。VR 状态取决于 LPDS 的配置；</p> <p>1: CPU 进入 SLEEPDEP 时系统进入待机模式。VR 关闭；</p>
0	LPDS	RW	0	<p>VR 低功耗深度睡眠配置。</p> <p>此位由软件置 1 和清零。与 PDDS 结合配置。</p> <p>0: 停止模式下 VR (MVR) 开启；</p> <p>1: 停止模式下 LPVR 开启；</p>

#### 4.2.2. 电源控制/状态寄存器(PWR\_CSR)(0x04)

Address offset: 0x04

Reset value: 0x0000 0000(从待机模式唤醒不能复位该寄存器的 WUF/SBF 和 EWUPx 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	FLT_CTRL[2:0]			FLTEN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1	Res	Res	Res	Res	PVDO	SBF		WUF
			RW	RW	RW	RW	RW					R	R		R

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19:17	FLT_CTRL[2:0]	RW	3'b000	<p>PVD 滤波时间配置。</p> <p>111: 保留；</p> <p>110: 滤波时间大约为 1024 个滤波时钟 (LSI 或者 LSE 时, 大约为 30.7ms)；</p> <p>101: 滤波时间大约为 128 个滤波时钟 (LSI 或者 LSE 时, 大约为 3.8ms)；</p> <p>100: 滤波时间大约为 64 个滤波时钟 (LSI 或者 LSE 时, 大约为 1.92ms)；</p> <p>011: 滤波时间大约为 16 个滤波时钟 (LSI 或者 LSE 时, 大约为 480us)；</p> <p>010: 滤波时间大约为 4 个滤波时钟 (LSI 或者 LSE 时, 大约为 120us)；</p> <p>001: 滤波时间大约为 2 个滤波时钟 (LSI 或者 LSE 时, 大约为 60us)；</p> <p>000: 滤波时间大约为 1 个滤波时钟 (LSI 或者 LSE 时, 大约为 30us)；</p>
16	FLTEN	RW	0	<p>PVD 数字滤波使能。</p> <p>0: PVD 数字滤波禁止；</p> <p>1: PVD 数字滤波使能；</p>
15:13	Reserved	-	-	Reserved
12	EWUP5	RW	0	<p>使能 WKUP5 引脚。</p> <p>此位由软件置 1 和清零。</p> <p>0: WKUP5 引脚用于通用 I/O。WKUP5 引脚上的事件不会从待机模式唤醒器件。</p> <p>1: WKUP5 引脚用于从待机模式唤醒器件并被强制配置为输入下拉 (WKUP5 引脚出现上升沿时从待机模式唤醒系统)。</p> <p>注: 此位通过系统复位进行复位。但从待机模式唤醒时该位不会被复位。</p>
11	EWUP4	RW	0	<p>使能 WKUP4 引脚。</p> <p>此位由软件置 1 和清零。</p> <p>0: WKUP4 引脚用于通用 I/O。WKUP4 引脚上的事件不会从待机模式唤醒器件。</p> <p>1: WKUP4 引脚用于从待机模式唤醒器件并被强制配置为输入下拉 (WKUP4 引脚出现上升沿时从待机模式唤醒系统)。</p> <p>注: 此位通过系统复位进行复位。但从待机模式唤醒时该位不会被复位。</p>
10	EWUP3	RW	0	<p>使能 WKUP3 引脚。</p> <p>此位由软件置 1 和清零。</p> <p>0: WKUP3 引脚用于通用 I/O。WKUP3 引脚上的事件不会从待机模式唤醒器件。</p> <p>1: WKUP3 引脚用于从待机模式唤醒器件并被强制配置为输入下拉 (WKUP3 引脚出现上升沿时从待机模式唤醒系统)。</p> <p>注: 此位通过系统复位进行复位。但从待机模式唤醒时该位不会被复位。</p>
9	EWUP2	RW	0	使能 WKUP2 引脚。

Bit	Name	R/W	Reset Value	Function
				<p>此位由软件置 1 和清零。</p> <p>0: WKUP2 引脚用于通用 I/O。WKUP2 引脚上的事件不会从待机模式唤醒器件。</p> <p>1: WKUP2 引脚用于从待机模式唤醒器件并被强制配置为输入下拉 (WKUP2 引脚出现上升沿时从待机模式唤醒系统)。</p> <p>注: 此位通过系统复位进行复位。但从待机模式唤醒时该位不会被复位。</p>
8	EWUP1	RW	0	<p>使能 WKUP1 引脚。</p> <p>此位由软件置 1 和清零。</p> <p>0: WKUP 引脚用于通用 I/O。WKUP 引脚上的事件不会从待机模式唤醒器件。</p> <p>1: WKUP 引脚用于从待机模式唤醒器件并被强制配置为输入下拉 (WKUP 引脚出现上升沿时从待机模式唤醒系统)。</p> <p>注: 此位通过系统复位进行复位。但从待机模式唤醒时该位不会被复位。</p>
7:3	Reserved	-	-	Reserved
2	PVDO	R	0	<p>PVD 输出标志。</p> <p>此位通过硬件置 1 和清零。仅当通过 PVDE 位使能 PVD 时此位才有效。</p> <p>0: VDD 高于 PLS[2:0] 位选择的 PVD 阈值。</p> <p>1: VDD 低于 PLS[2:0] 位选择的 PVD 阈值。</p> <p>注: PVD 在进入待机模式时停止。因此，待机模式后或复位后，此位等于 0，直到 PVDE 位置 1 然后根据检测电压阈值输出对应值。</p>
1	SBF	R	0	<p>待机模式标志。</p> <p>此位由硬件置 1，清零则只能通过 POR/PDR (VCC 上电复位/掉电复位) 或将 PWR_CR 寄存器中的 CSBF 位置 1 来实现。</p> <p>0: 器件未进入待机模式</p> <p>1: 器件已进入待机模式</p>
0	WUF	R	0	<p>待机模式唤醒标志。</p> <p>该位通过硬件置 1，清零则只能通过 POR/PDR (VCC 上电复位/掉电复位) 或将 PWR_CR 寄存器中的 CWUF 位置 1 清零。</p> <p>0: 未发生唤醒事件</p> <p>1: 收到唤醒事件，可能来自 WKUP 引脚、RTC 唤醒事件。</p>

#### 4.2.3. PWR 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x40000000	PWR_CCR																																		
0x40000000	Read/Write	Reserved										STDBY_MRRDY_WAIT[1:1]		FLS_WUP_T[1:0]		HS_ION_CTRL		VO_S[1:0]		BKPV_R_VOS[1:0]		D_B_P		PLS[2:0]		PVD_E		CSB_F		CWF		PDD_S		LPDS	
0x40000000	Reset Value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04000000	PWR_CSR	Reserved										FLT_CTR_L[2:0]		FLTE_N		Reserved		EWP5		EWP4		EWP3		EWP2		EWP1		Reserved		PVD_O		SBUF		WUF	



# 5. 复位与时钟控制 (RCC)

## 5.1. 复位功能描述

### 5.1.1. 系统复位

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

只要发生以下事件之一，就会产生系统复位：

- NRST 引脚低电平（外部复位）
- 窗口看门狗计数结束（WWDG 复位）
- 独立看门狗计数结束（IWDG 复位）
- 软件复位（SYSRESETREQ 复位）
- 低功耗管理复位（NRST\_STDBY/NRST\_STOP）
- Option byte 加载复位

#### 5.1.1.1. 软件复位

可通过查看 RCC 时钟控制和状态寄存器 (RCC\_CSR) 中的复位标志确定。要对器件进行软件复位，必须将带 FPU 的 Cortex®-M4 应用中断和复位控制寄存器中的 SYSRESETREQ 位置 1。

#### 5.1.1.2. 低功耗管理复位

引发低功耗管理复位的方式有两种：

- 进入待机模式时产生复位：

此复位的使能方式是清零用户选项字节中的 nRST\_STDBY 位。使能后，当检测到 SLEEPDEEP 信号，且 PWR\_CR\_PDDS 寄存器配置为待机模式，器件将复位，而非进入待机模式。

- 进入停止模式时产生复位：

此复位的使能方式是清零用户选项字节中的 nRST\_STOP 位。使能后，当检测到 SLEEPDEEP 信号，且 PWR\_CR\_PDDS 寄存器配置为停止模式，器件将复位，而非进入停止模式。

### 5.1.2. 电源复位

只要发生以下事件之一，就会产生电源复位：

- 上电/掉电复位（POR/PDR 复位）
- 在退出待机模式时

除备份域外，电源复位会将所有寄存器设为其复位值。

这些源均作用于 NRST 引脚，该引脚在复位过程中始终保持低电平。RESET 复位入口向量在存储器映射中固定在地址 0x0000 0004。

芯片内部的复位信号会向 NRST 引脚上输出一个低电平脉冲。脉冲发生器可确保每个内部复位源的复位脉冲都至少持续 20 μs。对于外部复位，在 NRST 引脚处于低电平时产生复位脉冲。

### 5.1.3. 备份域复位

备份域复位会将所有备份域寄存器复位为各自的复位值，备份域寄存器包括 RCC\_BDCR, 备份寄存器以及 RTC 部分寄存器。

只要发生以下事件之一，就会产生备份域复位：

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

### 5.1.4. 备份域复位之外的复位的统一处理

除备份域复位外，其他源复位都作用于 NRST 引脚，该引脚在复位过程中时钟保持低电平。

电路如下图所示：

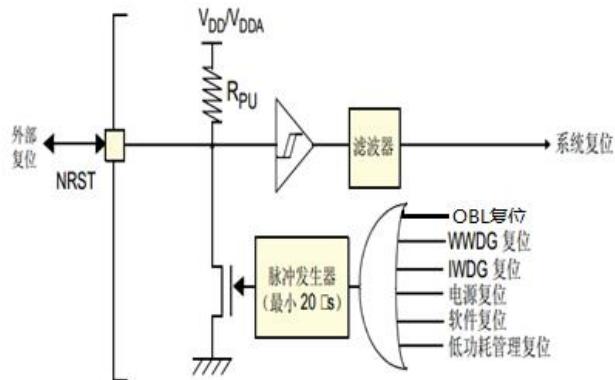


图 5-1 复位电路图

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

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

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

## 5.2. 时钟功能描述

### 5.2.1. 时钟结构

时钟结构如下图所示：

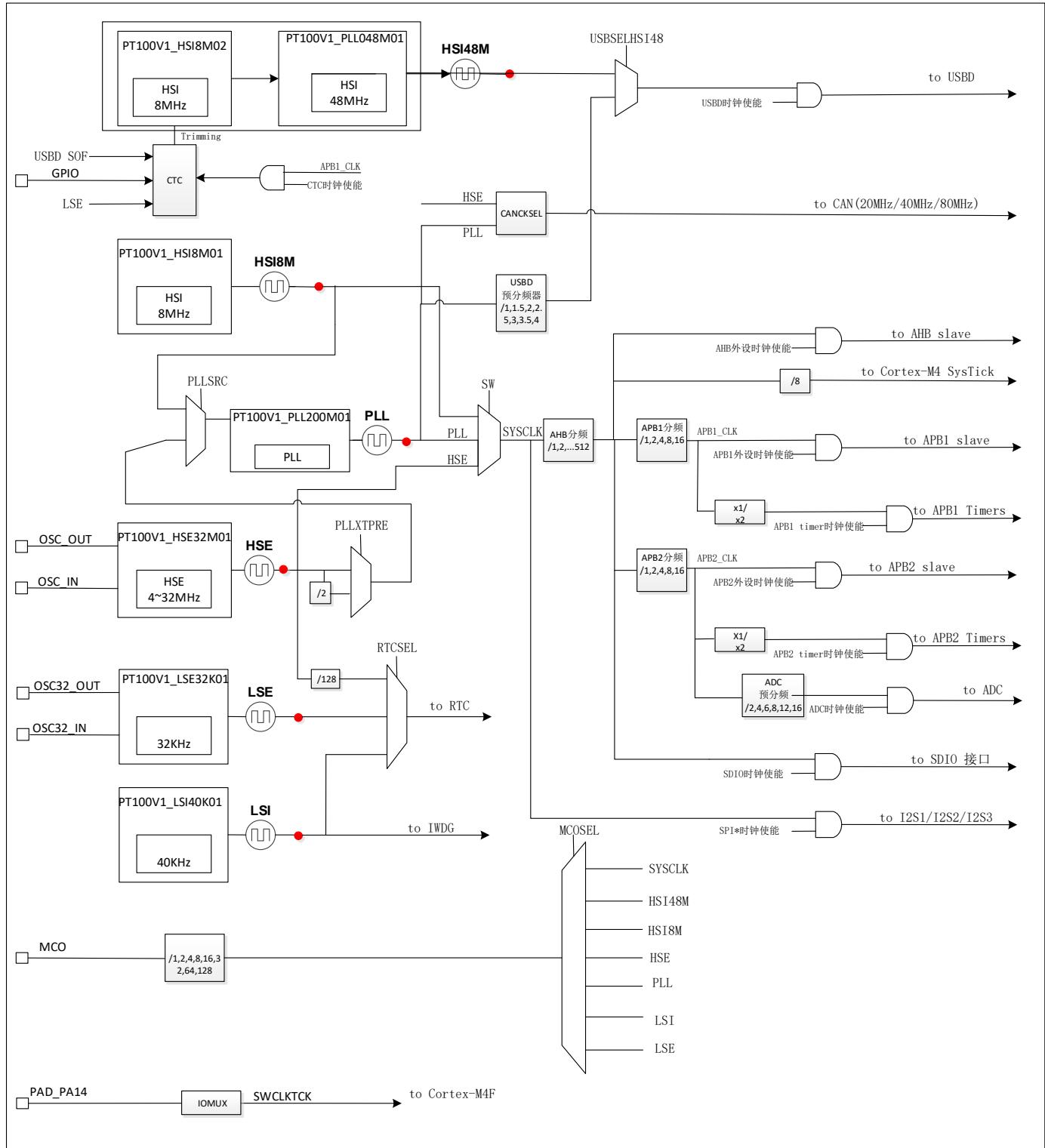


图 5-2 时钟结构

## 5.2.2. 时钟源（时钟信号和参数以模拟模块需求为准）

系统存在如下时钟源：

### 5.2.2.1. HSE 时钟

外部高频 OSC. 频率范围 4~32MHz.

HSE 时钟的稳定时间由模拟 HSE 模块计数，稳定时间由 FLASH\_TRIM9.HSE\_RDYSEL 配置。模拟 HSE 模块在 HSE 启动并且稳定后产生信号 HSE\_RDY。

HSE 时钟可以有两个来源：

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

为保证 CLOCK 输出第一个时钟就为稳定时钟，HSE 内部包含一个计数器，计数完成后输出第一个时钟上升沿。计数周期可由 HSE\_RDYSEL 设置选择四种计数挡位，四种计数挡位根据 HSE 设计选取合适计数周期。其中 00~10 计数挡位根据 HSE 设计选取合适计数周期，11 计数挡位为测试模式，无计数，CLOCK 直接输出。Normal mode 和 bypass mode 都有时钟计数，但是计数周期不同，bypass 模式三种计数挡位的计数周期是 normal mode 的一半。

#### **Bypass**

HSE\_BYP=1：

Bypass 情况下，模拟 HSE 基于 HSE\_EXT\_CLK，计数 FLASH\_TRIM9.HSE\_RDYSEL 配置的周期数的一半时间后输出 CLK\_HSE 即 HSE\_RDY 信号。

#### **5.2.2.2. HSI8M 时钟**

内部 8MHz RC 振荡器。相较于 XTAL OSC，RC OSC 功耗低，稳定时间短，但精度低。HSI 模拟模块计数稳定时间，即输出给数字的 HSI8M 时钟为稳定时钟。

上电复位后，在加载阶段，HSI 的校准值加载到 RCC\_CR.HSICAL 寄存器。

从停止模式唤醒后，HSI 将作为系统时钟源。

#### **5.2.2.3. HSI48M 时钟**

HSI48M 时钟由 HSI\_USB(8MHz)和 PLL48M 组成。HSI48MHz 时钟支持通过 CTC 模块自校准，CTC 校准 HSI\_USB，PLL48M 的参考时钟为 HSI\_USB 的输出，固定为 6 倍频。HSI48M 用做 USBD 模块的时钟。

#### **5.2.2.4. PLL 时钟**

PLL 模块是倍频 HSE 时钟，PLL 输入时钟频率范围为 8MHz~25MHz。输出频率为 24MHz~168MHz。

#### **5.2.2.5. LSE 时钟**

外部 32.76KHz OSC，用作低功耗时钟。

可以通过配置 LSEDRV 在稳定时间和功耗之间做平衡。LSE 稳定时间为 0.5s（典型值）。LSE 稳定信号由 LSE 模拟模块产生。

与 HSE 来源类似，LSE 也有两个来源：

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

为保证 CLOCK 输出第一个时钟就为稳定时钟，LSE 内部包含一个计数器，计数完成后输出第一个时钟上升沿。计数周期可由 LSE\_RDYSEL\_VBKP 设置选择四种计数挡位，其中 00~10 计数挡位根据 LSE 设计选取合适计数周期，11 计数挡位为测试模式，无计数，CLOCK 直接输出。Normal mode 和 bypass mode 都有时钟计数，但是计数周期不同，bypass 模式三种计数挡位的计数周期是 normal mode 的一半。

#### **Bypass**

在 LSE bypass 时，模拟 LSE 给出的 HSERDY 信号值为 0。

在 LSE bypass 时，硬件 VBAK\_RCC 模块通过计数产生 LSE 稳定等待时间，等待时间长短由 RCC\_BDCR.LSERDYSEL\_BP 寄存器来控制。

在非 bypass 情况下，通过配置 FLASH information 区寄存器，可以控制模拟输出给逻辑的时钟 CLK\_LSE 为非稳定时钟或者为稳定时钟，即通过 MCO 输出的 LSE 时钟包含可以包含非稳定阶段，或者只包含稳定后时钟。其 ready 信号也会在时钟输出时同步给出。

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

#### 5.2.2.6. LSI 时钟

内部低频 40KHz 时钟。用作 IWDG 时钟。

### 5.3. 寄存器 (base: 0x40021000)

#### 5.3.1. 时钟控制寄存器 (RCC\_CR)

Address:0x00

Reset value:0x0080 2083

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

Bit	Name	R/W	Reset Value	Function
31:26	Reserved	-	-	保留
25	PLLRDY	R	0	PLL 时钟 ready 标志。 硬件置位，表明 PLL 时钟 locked。 0: PLL unlocked; 1: PLL locked;
24	PLLON	RW	0	PLL 使能。 当系统进入停止、待机模式时，硬件会清零该位。当 PLL 时钟被用作 (SWS 为 PLL) 或将要被用作系统时钟 (SW 选择 PLL) 时，该位不能被清零。 0: PLL OFF; 1: PLL ON;
23:21	HSICAL[10:8]	R	0x4	HSI8M 时钟校准值高 2 bits。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSICAL+HSITRIM，当 HSITRIM 值改变时，该寄存器值也会更新。
20	Reserved	-	-	保留
19	CSSON	RW	0x0	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)； 1: 时钟安全系统开启 (如果 HSE 时钟稳定则时钟检测开启，否则关闭时钟检测)；
18	HSEBYP	RW	0	HSE 屏蔽晶振，选择管脚输入时钟。 该位只有当 HSEON=0 时才能写。 0: HSE 晶振不屏蔽，外部高速时钟选择晶振； 1: HSE 晶振屏蔽，外部高速时钟选择外部管脚输入时钟源；
17	HSERDY	R	0	HSE 晶振时钟 ready 标志。 该位由硬件置位表明 HSE 晶振稳定。 0: HSE 晶振没有 ready； 1: HSE 晶振 ready；
16	HSEON	RW	0	HSE 晶振使能。 当系统进入停止、待机模式时，硬件会清零该位，关闭 HSE 晶振。当 HSE 直接或者间接作为系统时钟源时，该位不能被置 0。

Bit	Name	R/W	Reset Value	Function
				0: HSE 晶振 OFF; 1: HSE 晶振 ON;
15:8	HSICAL[7:0]	R	0x20	HSI8M 时钟校准值。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSICAL+HSITRIM，当 HSITRIM 值改变时，该寄存器值也会更新。
7:3	HSITRIM[4:0]	RW	0x10	HSI8M 时钟 Trimming 值。 软件写入该寄存器调整 HSI8M 时钟，叠加到 HSICAL 后输出给模拟 HSI8M。 系统复位会复位该寄存器。
2	Reserved	-	-	保留
1	HSIRDY	R	1	HSI8M 时钟 ready 标志。 硬件置位表明 HSI OSC 稳定。该位只有当 HSION=1 时才有效。 0: HSI8M OSC not ready; 1: HSI8M OSC ready; 当 HSION 清零后，HSIRDY 将在 6 个 HSI8M 时钟之后拉低。
0	HSION	RW	1	HSI8M 时钟使能。 硬件在进入停止、待机模式时，会根据需要清零该寄存器停止 HSI8M。 当 HSI8M 直接或者间接作为系统时钟时，该寄存器不能为 0。 0: HSI8M OSC OFF; 1: HSI8M OSC ON; 硬件在如下情况会使能 HSI8M: 1) 硬件从停止或者待机模式唤醒后； 2) HSE 直接/间接作为系统时钟但出现 HSE CSS fail;

### 5.3.2. 时钟配置寄存器 (RCC\_CFGR)

Address:0x04

Reset value:0x0000 00000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USBPB[2]	PLLMULL[5:4]			ADCPRE[2]	Res	MCO [2:0]		USBPB[1:0]		PLLMULL[3:0]			PLLXTPRE	PLLSRC	
RW	RW			RW		RW			RW			RW	RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]			PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]			SWS[1:0]	SW[1:0]		
RW			RW			RW			RW			R	RW		

Bit	Name	R/W	Reset Value	Function
31	USBPB[2]	RW	0	与 bit 23:22 结合，配置 USB 时钟预分频值。
30:29	PLLMULL[5:4]	RW	0	与 bit 21:18 组合配置 PLL 倍频因子。
28	ADCPRE[2]	RW	0	与 bit15:14 组合配置 ADC 时钟的分频因子。
27	Reserved			保留
26:24	MCO [2:0]	RW	0	MCO 输出时钟选择。 000: no clock, MCO output disabled 001: LSE32K 010: LSI40K 011: HSI48M 100: SYSCLK 101: HSI8M01 110: HSE 111: PLL others: reserved 注 1：若 HSE 和 LSE 配置 HSE_RDYSEL 和 LSE_RDYSEL 寄存器，则可以输出 startup 期间的时钟； 注 2：MCO 输出选择系统时钟时，需要保证输出时频率不超过 IO 的最大允许频率。
23:22	USBPB[1:0]	RW	0	USB 时钟预分频。与 bit31 组合配置如下： 000: PLL 时钟 1.5 分频 001: PLL 时钟 010: PLL 时钟 2.5 分频 011: PLL 时钟 2 分频

Bit	Name	R/W	Reset Value	Function
				100: PLL 时钟 3 分频 101: PLL 时钟 3.5 分频 11x: PLL 时钟 4 分频
21:18	PLLMULL[3:0]	RW	0	与 bit31:29 组合配置 PLL 倍频因子。 00000: x2 00001: x3 00010: x4 ..... 01001: x19 010010: x20 010011: x21 其他: 保留
17	PLLXTPRE	RW	0	HSE 分频作为 PLL 源。 只能在关闭 PLL 时才能写入此位。 0: HSE 时钟作为 PLL 源 1: HSE 时钟 2 分频作为 PLL 源
16	PLLSRC	RW	0	PLL 时钟源选择。 只能在关闭 PLL 时才能写入此位。 0: res 1: PLLXTPRE 配置后的 HSE 时钟作为 PLL 输入时钟
15:14	ADCPRE[1:0]	RW	0	与 bit27 组合配置 ADC 时钟的分频因子。 000: APB2 CLK 2 分频 001: APB2 CLK 4 分频 010: APB2 CLK 6 分频 011: APB2 CLK 8 分频 100: APB2 CLK 2 分频 101: APB2 CLK 12 分频 110: APB2 CLK 8 分频 111: APB2 CLK 16 分频
13:11	PPRE2[2:0]	RW	0	高速 APB (APB2) 时钟分频系数。 PCLK 基于 HCLK 的分频系数。 0xx: 不分频 100: 2 分频 101: 4 分频 110: 8 分频 111: 16 分频
10:8	PPRE1[2:0]	RW	0	低速 APB (APB1) 时钟分频系数。 PCLK 基于 HCLK 的分频系数。 0xx: 不分频 100: 2 分频 101: 4 分频 110: 8 分频 111: 16 分频
7:4	HPRE[3:0]	RW	0	AHB 时钟 HCLK 基于 SYSCLK 的分频系数。 0xxx: 不分频 1000: 2 分频 1001: 4 分频 1010: 8 分频 1011: 16 分频 1100: 64 分频 1101: 128 分频 1110: 256 分频 1111: 512 分频 为了保证系统正常工作，需要根据 VR 电源情况配置合适频率。

Bit	Name	R/W	Reset Value	Function
				注：当 AHB 时钟的预分频系数大于 1 时，必须开启预取缓冲器。 注：建议逐级切换分频系数。
3:2	SWS[1:0]	R	0	系统时钟源选择。 该位由硬件控制，表明系统时钟源的选择。 00: HSI8M 01: HSE 10: PLL CLK Others: Reserved
1:0	SW[1:0]	RW	0	系统时钟源选择。 该位由硬件和软件控制，表明系统时钟源的选择。 00: HSI8M 01: HSE 10: PLL CLK Others: Reserved

### 5.3.3. 时钟中断寄存器 (RCC\_CIR)

Address:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								CSSC	Res	HSI48RDYC	PLLRDYC	HSER-DYC	HSERDYC	LSER-DYC	LSIRDYC
								W		W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	HSI48RDYIE	PLLRDYIE	HSER-DYIE	HSIRDYIE	LSER-DYIE	LSIRDYIE	CSSF	Res	HSI48RDYF	PLLRDYF	HSER-DYF	HSIRDYF	LSER-DYF	LSIRDYF	
	RW	RW	RW	RW	RW	RW	R		R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:24	保留	-	-	保留
23	CSSC	W	0	HSE 时钟安全系统 (CSS) 中断标志清零。 0: No effect; 1: 清零 CSSF 标志
22	保留	-	-	保留
21	HSI48RDYC	W	0	HSI48M ready 中断标志清零。 0: No effect; 1: 清零 HSI48RDYF 标志;
20	PLLRDYC	W	0	PLL ready 中断标志清零。 0: No effect; 1: 清零 PLLRDYF 标志;
19	HSERDYC	W	0	HSE ready 中断标志清零。 0: No effect; 1: 清零 HSERDYF 标志;
18	HSIRDYC	W	0	HSI ready 中断标志清零。 0: No effect; 1: 清零 HSIRDYF 标志;
17	LSERDYC	W	0	LSE ready 中断标志清零。 0: No effect; 1: 清零 LSERDYF 标志;
16	LSIRDYC	W	0	LSI ready 中断标志清零。 0: No effect; 1: 清零 LSIRDYF 标志;
15:14	Reserved	-	-	保留
13	HSI48RDYIE	RW	0	HSI48M 时钟 ready 中断使能。 0: 禁止; 1: 使能;
12	PLLRDYIE	RW	0	PLL ready 中断使能。 0: 禁止; 1: 使能;

Bit	Name	R/W	Reset Value	Function
11	HSERDYIE	RW	0	HSE 时钟 ready 中断使能。 0: 禁止; 1: 使能;
10	HSIRDYIE	RW	0	HSI8M 时钟 ready 中断使能。 0: 禁止; 1: 使能;
9	LSERDYIE	RW	0	LSE 时钟 ready 中断使能。 0: 禁止; 1: 使能;
8	LSIRDYIE	RW	0	LSI 时钟 ready 中断使能。 0: 禁止; 1: 使能;
7	CSSF	R	0	时钟安全系统中断标志。 当 HSE 时钟出现故障时, 由硬件置 1。软件通过写 CSSC 位 1 清零该标志位。 0: 无 HSE 时钟失效产生的安全系统中断产生; 1: HSE 时钟失效产生的安全系统中断产生;
6	Reserved		-	保留
5	HSI48RDYF	R	0	HSI48M 时钟 ready 中断标志。 HSI48M 时钟稳定并且 HSI48RDYIE=1 时, 硬件置位该寄存器。 0: HSI48M 时钟 ready 中断未产生; 1: HSI48M 时钟 ready 中断产生; 写 HSI48RDYC 寄存器 1 清零该位。
4	PLLRDYF	R	0	PLL ready 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 PLLRDYC1 清零该位。
3	HSERDYF	R	0	HSE 时钟 ready 中断标志。 当 HSE 时钟稳定并且 HSERDYIE=1 时, 硬件置位该寄存器。 0: HSE 时钟 ready 中断未产生; 1: HSE 时钟 ready 中断产生; 写 HSERDYC 寄存器 1 清零该位。
2	HSIRDYF	R	0	HSI8M 时钟 ready 中断标志。 HSI8M 时钟稳定并且 HSIRDYIE=1 时, 硬件置位该寄存器。 0: HSI8M 时钟 ready 中断未产生; 1: HSI8M 时钟 ready 中断产生; 写 HSIRDYC 寄存器 1 清零该位。
1	LSERDYF	R	0	LSERDY 时钟 ready 中断标志。 LSE 时钟稳定并且 LSERDYDIE=1 时, 硬件置位该寄存器。 0: LSERDY 时钟 ready 中断未产生; 1: LSERDY 时钟 ready 中断产生; 写 LSERDYC 寄存器 1 清零该位。
0	LSIRDYF	R	0	LSIRDY 时钟 ready 中断标志。 LSI 时钟稳定并且 LSIRDYIE=1 时, 硬件置位该寄存器。 0: LSIRDY 时钟 ready 中断未产生; 1: LSIRDY 时钟 ready 中断产生; 写 LSIRDYC 寄存器 1 清零该位。

### 5.3.4. APB2 外设复位寄存器 (RCC\_APB2RSTR)

Address:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										TIM11RST	TIM10RST	TIM9RST	Res		

										RW	RW	RW				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Res						SYS-CFGRST			
RW	RW	RW	RW	RW	RW	RW							RW			

Bit	Name	R/W	Reset Value	Function
31:22	Reserved		-	保留
21	TIM11RST	RW	0	TIM11 模块复位。 0: no effect; 1: 该模块复位;
20	TIM10RST	RW	0	TIM10 模块复位。 0: no effect; 1: 该模块复位;
19	TIM9RST	RW	0	TIM9 模块复位。 0: no effect; 1: 该模块复位;
18:16	保留	RES	-	保留
15	ADC3RST	RW	0	ADC3 模块复位。 0: no effect; 1: 该模块复位;
14	USART1RST	RW	0	USART1 模块复位。 0: no effect; 1: 该模块复位;
13	TIM8RST	RW	0	TIM8 模块复位。 0: no effect; 1: 该模块复位;
12	SPI1RST	RW	0	SPI1 模块复位。 0: no effect; 1: 该模块复位;
11	TIM1RST	RW	0	TIM1 模块复位。 0: no effect; 1: 该模块复位;
10	ADC2RST	RW	0	ADC2 模块复位。 0: no effect; 1: 该模块复位;
9	ADC1RST	RW	0	ADC1 模块复位。 0: no effect; 1: 该模块复位;
8:1	保留	-	-	保留
0	SYSCFGRST	RW	0	SYSCFG 模块复位。 0: no effect; 1: 该模块复位;

### 5.3.5. APB1 外设复位寄存器 (RCC\_APB1RSTR)

Address:0x10

Reset value:0x0000 0000

31	30	29	28	27	2 6	25	24	23	22	21	20	19	18	17	16
CTCRS T	Res	Re s	PWRRS T	Res	CANRS T	Res	USBRST	I2C2RST	I2C1RS T	USART5RS T	USART4RS T	USART3RS T	USART2RS T	Res	
RW			RW		RW		RW	RW	RW	RW	RW	RW	RW	RW	
15	14	13	12	11	1 0	9	8	7	6	5	4	3	2	1	0
SPI3RS T	SPI2RS T	Res	WWDRS T	Res	TIM14RS T	TIM13RS T	TIM12RS T	TIM7RS T	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RS T		
RW	RW		RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	CTCRST	RW	0	CTC 模块复位。 0: no effect; 1: 该模块复位;
30:29	保留	RES	-	保留
28	PWRRST	RW	0	Power 接口模块复位。 0: no effect; 1: 该模块复位;
27:26	保留	RES	-	保留
25	CANRST	RW	0	CAN 模块复位。 0: no effect; 1: 该模块复位;
24	保留	RES	-	保留
23	USBRST	RW	0	USBD 模块复位。 0: no effect; 1: 该模块复位;
22	I2C2RST	RW	0	I2C2 模块复位。 0: no effect; 1: 该模块复位;
21	I2C1RST	RW	0	I2C1 模块复位。 0: no effect; 1: 该模块复位;
20	USART5RST	RW	0	UART5 模块复位。 0: no effect; 1: 该模块复位;
19	USART4RST	RW	0	USART4 模块复位。 0: no effect; 1: 该模块复位;
18	USART3RST	RW	0	USART3 模块复位。 0: no effect; 1: 该模块复位;
17	USART2RST	RW	0	USART2 模块复位。 0: no effect; 1: 该模块复位;
16	保留	RES	-	保留
15	SPI3RST	RW	0	SPI3 模块复位。 0: no effect; 1: 该模块复位;
14	SPI2RST	RW	0	SPI2 模块复位。 0: no effect; 1: 该模块复位;
13:12	保留	RES	-	保留
11	WWDGRST	RW	0	WWDG 模块复位。 0: no effect; 1: 该模块复位;
10:9	保留	RES	-	保留
8	TIM14RST	RW	0	TIM14 模块复位。 0: no effect; 1: 该模块复位;
7	TIM13RST	RW	0	TIM13 模块复位。 0: no effect; 1: 该模块复位;
6	TIM12RST	RW	0	TIM12 模块复位。 0: no effect; 1: 该模块复位;
5	TIM7RST	RW	0	TIM7 模块复位。

Bit	Name	R/W	Reset Value	Function
				0: no effect; 1: 该模块复位;
4	TIM6RST	RW	0	TIM6 模块复位。 0: no effect; 1: 该模块复位;
3	TIM5RST	RW	0	TIM5 模块复位。 0: no effect; 1: 该模块复位;
2	TIM4RST	RW	0	TIM4 模块复位。 0: no effect; 1: 该模块复位;
1	TIM3RST	RW	0	TIM3 模块复位。 0: no effect; 1: 该模块复位;
0	TIM2RST	RW	0	TIM2 模块复位。 0: no effect; 1: 该模块复位;

### 5.3.6. AHB1 外设时钟使能寄存器 (RCC\_AHB1ENR)

Address:0x014

Reset value:0x0000 0014

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

Bit	Name	R/W	Reset Value	Function
31:13	保留	RES	-	保留
12	ESMCEN	RW	0	ESMC(QSPI)模块时钟使能。 0: 禁止; 1: 使能;
11	保留	RES	-	保留
10	SDIOEN	RW	0	SDIO 模块时钟使能。 0: 禁止; 1: 使能;
9:7	保留	RES	-	保留
6	CRCEN	RW	0	CRC 模块时钟使能。 0: 禁止; 1: 使能;
5	保留	RES	-	保留
4	FMCEN	RW	1	FLASH 接口模块时钟使能, 针对 sleep 模式。 0: 禁止; 1: 使能;
3	保留	RES	-	保留
2	SRAMEN	RW	1	SRAM 存储区时钟使能, 针对 sleep 模式。 0: 禁止; 1: 使能;
1	DMA2EN	RW	0	DMA2 模块时钟使能。 0: 禁止; 1: 使能;
0	DMA1EN	RW	0	DMA1 模块时钟使能。 0: 禁止;

Bit	Name	R/W	Reset Value	Function
				1: 使能;

### 5.3.7. APB2 外设时钟使能寄存器 (RCC\_APB2ENR)

Address:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res										TIM11EN	TIM10EN	TIM9EN	Res			
										RW	RW	RW				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Res									
RW	RW	RW	RW	RW	RW	RW										

Bit	Name	R/W	Reset Value	Function
31:22	保留	RES	-	保留
21	TIM11EN	RW	0	TIM11 模块时钟使能。 0: 禁止; 1: 使能;
20	TIM10EN	RW	0	TIM10 模块时钟使能。 0: 禁止; 1: 使能;
19	TIM9EN	RW	0	TIM9 模块时钟使能。 0: 禁止; 1: 使能;
18:16	保留	RES	-	保留
15	ADC3EN	RW	0	ADC3 模块时钟使能。 0: 禁止; 1: 使能;
14	USART1EN	RW	0	USART1 模块时钟使能。 0: 禁止; 1: 使能;
13	TIM8EN	RW	0	TIM8 模块时钟使能。 0: 禁止; 1: 使能;
12	SPI1EN	RW	0	SPI1 模块时钟使能。 0: 禁止; 1: 使能;
11	TIM1EN	RW	0	TIM1 模块时钟使能。 0: 禁止; 1: 使能;
10	ADC2EN	RW	0	ADC2 模块时钟使能。 0: 禁止; 1: 使能;
9	ADC1EN	RW	0	ADC1 模块时钟使能。 0: 禁止; 1: 使能;
8:1	保留	RES	-	保留
0	SYSCFGEN	RW	0	SYSCFG 模块时钟使能。 0: 禁止; 1: 使能;

### 5.3.8. APB1 外设时钟使能寄存器 (RCC\_APB1ENR)

Address:0x1C

Reset value:0x0000 0000

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

CTCEN	Res	Res	PWREN	BKPEN	Res	CANEN	Res	USBDEN	I2C2EN	I2C1EN	USART5EN	USART4EN	USART3EN	USART2EN	Res
RW			RW	RW		RW		RW							
<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>
SPI3EN	SPI2EN	Res	WWWDGEN	Res	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN		
RW	RW		RW			RW									

Bit	Name	R/W	Reset Value	Function
31	CTCEN	RW	0	CTC 模块时钟使能。 0: 禁止; 1: 使能;
30:29	保留	RES	-	保留
28	PWREN	RW	0	PWR 模块时钟使能。 0: 禁止; 1: 使能;
27	BKPEN	RW	0	BACKUP 模块时钟使能。 0: 禁止; 1: 使能;
26	保留	RES	-	保留
25	CANEN	RW	0	CAN 模块时钟使能。 0: 禁止; 1: 使能;
24	保留	RES	-	保留
23	USBDEN	RW	0	USBD 模块时钟使能。 0: 禁止; 1: 使能;
22	I2C2EN	RW	0	I2C2 模块时钟使能。 0: 禁止; 1: 使能;
21	I2C1EN	RW	0	I2C1 模块时钟使能。 0: 禁止; 1: 使能;
20	USART5EN	RW	0	UART5 模块时钟使能。 0: 禁止; 1: 使能;
19	USART4EN	RW	0	UART4 模块时钟使能。 0: 禁止; 1: 使能;
18	USART3EN	RW	0	USART3 模块时钟使能。 0: 禁止; 1: 使能;
17	USART2EN	RW	0	USART2 模块时钟使能。 0: 禁止; 1: 使能;
16	Reserved		-	保留
15	SPI3EN	RW	0	SPI3 模块时钟使能。 0: 禁止; 1: 使能;
14	SPI2EN	RW	0	SPI2 模块时钟使能。 0: 禁止; 1: 使能;
13:12	保留	RES	-	保留
11	WWWDGEN	RW	0	WWDG 模块时钟使能。 0: 禁止; 1: 使能;
10:9	Reserved		-	保留

Bit	Name	R/W	Reset Value	Function
8	TIM14EN	RW	0	TIM14 模块时钟使能。 0: 禁止; 1: 使能;
7	TIM13EN	RW	0	TIM13 模块时钟使能。 0: 禁止; 1: 使能;
6	TIM12EN	RW	0	TIM12 模块时钟使能。 0: 禁止; 1: 使能;
5	TIM7EN	RW	0	TIM7 模块时钟使能。 0: 禁止; 1: 使能;
4	TIM6EN	RW	0	TIM6 模块时钟使能。 0: 禁止; 1: 使能;
3	TIM5EN	RW	0	TIM5 模块时钟使能。 0: 禁止; 1: 使能;
2	TIM4EN	RW	0	TIM4 模块时钟使能。 0: 禁止; 1: 使能;
1	TIM3EN	RW	0	TIM3 模块时钟使能。 0: 禁止; 1: 使能;
0	TIM2EN	RW	0	TIM2 模块时钟使能。 0: 禁止; 1: 使能;

### 5.3.9. RTC domain 控制寄存器 (RCC\_BDCR)

Address:0x20

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:17	Reserved		-	保留
16	BDRST	RW	0	备份域软复位。 0: no effect; 1: 复位;
15	RTCEN	RW	0	RTC 和 TAMP 时钟使能。 0: 禁止; 1: 使能;
14:11	Reserved		-	保留
10	RSTOUT_DIS	RW	0	复位输出禁止寄存器。 0: 内部复位经过 30us 脉冲发生器后输出; 1: 内部复位禁止输出。 在待机模式进入时, 为了保证唤醒时间, 需要配置该位为 1.
9:8	RTCSEL[1:0]	RW	0	RTC 时钟源选择。 00: No clock 01: LSE

Bit	Name	R/W	Reset Value	Function
				10: LSI 11: HSE divided by 128。 一旦 RTC 时钟源选择后不能再改变，除非备份域复位被清零。
7	Reserved		-	保留
6:5	LSERDYSEL_BP[1:0]	RW	0x0	LSE Bypass 时计数时间选择。 00: 8 个 LSE 时钟 01: 16 个 LSE 时钟 10: 24 个 LSE 时钟 11: 32 个 LSE 时钟
4:3	LSEDRV[1:0]	RW	0	LSE 驱动能力设置, default 00 00: gm 2.5uA/V 01: gm 3.75uA/V 10: gm 8.5uA/V 11: gm 13.5uA/V
2	LSEBYP	RW	0	LSE OSC bypass 0: Not bypassed, 低速外部时钟选择晶振； 1: Bypassed, 低速外部时钟选择外部接口输入时钟； 注：只有当外部 32KHz 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。 在 LSEON 清零后，该位需要 6 个 LSE 时钟后再清零。
0	LSEON	RW	0	LSE OSC 使能。 0: 禁止； 1: 使能；

### 5.3.10. 控制/状态寄存器 (RCC\_CSR)

Address:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	RMVF	Res							
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						LSIRDY	LSION
														R	RW

Bit	Name	R/W	Reset Value	Function
31	LPWRRSTF	R	0	低功耗复位标志。 当进入非法的停止/待机低功耗模式时，硬件置位该寄存器。 只有当 nRST_停止, nRST_STDBY option bits 为清零（有效） 状态时才能操作该寄存器。 RMVF 置 1 会清零该位。
30	WWDGRSTF	R	0	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	POR/PDR 复位标志。 RMVF 置 1 会清零该位。
26	PINRSTF	R	0	外部 NRST 管脚复位标志。 RMVF 置 1 会清零该位。
25	OBLRSTF	R	0	Option byte loader 复位标志。 RMVF 置 1 会清零该位。
24	RMVF	RW	0	软件置位清零复位标志。

Bit	Name	R/W	Reset Value	Function
				写该位为 1 的操作会清零 bit25 开始的复位标志，软件写 1 后该位会维持为 1，硬件不会清零。
23:2	Reserved		-	保留
1	LSIRDY	R	0	LSI OSC 稳定标志。 LSIRDY 由模拟 LSI 产生。
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；

### 5.3.11. 时钟复位配置寄存器 1 (RCC\_CFGR1)

Address:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
USBSELHSI48	Res		HSI48CAL[12:0]													
RW			RW													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
HSI48TRIM[6:0]								Res			HSI48RDY	HSI48ON	Res	MCOPRE[2:0]		
RW											R	RW		RW		

Bit	Name	R/W	Reset Value	Function
31	USBSELHSI48	RW	0	USBD 模块时钟（非系统时钟）选择。 0: USBD 时钟选择 PLL（分频）； 1: USBD 时钟选择 HSI48M
30:29	Reserved			
28:16	HSI48CAL[12:0]	R	0x1080	HSI48M 时钟校准值。 上电流程的加载阶段，该寄存器加载 Flash information 区值。 软件读该寄存器返回值为 HSI48CAL+HSI48TRIM，当 HSI48TRIM 值改变时，该寄存器值也会更新。
15:9	HSI48TRIM[6:0]	RW	0x40	HSI48M 时钟 Trimming 值。 软件写入该寄存器调整 HSI48M 时钟，叠加到 HSI48CAL 后输出给模拟 HSI48M。 系统复位时该寄存器会被复位。
8:6	Reserved		-	保留
5	HSI48RDY	R	0	HSI48M 时钟 ready 标志。 硬件置位表明 HSI48M 稳定。该位只有当 HSI48MON=1 时才有效。 0: HSI48M not ready; 1: HSI48M ready; 当 HSI48MON 清零后，HSI48MRDY 将在 6 个 HSI48M 时钟之后拉低。
4	HSI48ON	RW	0	HSI48M 时钟使能。 硬件在进入停止、待机模式时，会根据需要清零该寄存器停止 HSI48M。 0: HSI48M OFF; 1: HSI48M ON;
3	Reserved		-	保留
2:0	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

Bit	Name	R/W	Reset Value	Function
				101: 32 110: 64 111: 128 推荐在 MCO 输出使能前，设置这些位。

### 5.3.12. AHB1 外设复位寄存器 (RCC\_AHB1RSTR)

Address: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		ESMCRST	Res	SDIORST	Res			CRCRST	Res					DMA2RST	DMA1RST
		RW		RW				RW						RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	Reserved		-	保留
12	ESMCRST	RW	0	ESMC(QSPI)模块复位使能。 0: 禁止; 1: 使能;
11	Reserved		-	保留
10	SDIORST	RW	0	SDIO 模块复位使能。 0: 禁止; 1: 使能;
9:7	Reserved		-	保留
6	CRCRST	RW	0	CRC 模块复位使能。 0: 禁止; 1: 使能;
5:2	Reserved	RES	-	保留
1	DMA2RST	RW	0	DMA2 模块复位使能。 0: 禁止; 1: 使能;
0	DMA1RST	RW	0	DMA1 模块复位使能。 0: 禁止; 1: 使能;

### 5.3.13. AHB2 外设复位寄存器 (RCC\_AHB2RSTR)

Address: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									IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Res	
									RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:7	Reserved		-	保留
6	IOPERST	RW	0	模块复位。 0: no effect; 1: 该模块复位;
5	IOPDRST	RW	0	模块复位。 0: no effect; 1: 该模块复位;

Bit	Name	R/W	Reset Value	Function
4	IOPCRST	RW	0	模块复位。 0: no effect; 1: 该模块复位;
3	IOPBRST	RW	0	模块复位。 0: no effect; 1: 该模块复位;
2	IOPARST	RW	0	模块复位。 0: no effect; 1: 该模块复位;
1:0	Reserved		-	保留

### 5.3.14. AHB2 外设时钟使能寄存器 (RCC\_AHB2ENR)

Address: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										IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res
										RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:7	Reserved		-	保留
6	IOPEEN	RW	0	IOPE 模块时钟使能。 0: 禁止; 1: 使能;
5	IOPDEN	RW	0	IOPD 模块时钟使能。 0: 禁止; 1: 使能;
4	IOPCEN	RW	0	IOPC 模块时钟使能。 0: 禁止; 1: 使能;
3	IOPBEN	RW	0	IOPB 模块时钟使能。 0: 禁止; 1: 使能;
2	IOPAEN	RW	0	IOPA 模块时钟使能。 0: 禁止; 1: 使能;
1:0	Reserved	RES	-	保留

### 5.3.15. 时钟复位配置寄存器 2 (RCC\_CFGR2)

Address:0x38

Reset value:0x0000 0008

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						HSE_DRV				Res				CANCKSEL[3:0]											
RW																									

Bit	Name	R/W	Reset Value	Function
31:10	Reserved		-	保留
9:8	HSE_DRV[1:0]	RW	0x0	HSE 驱动能力配置 00: gm 3.5mA/V 01: gm 5.0mA/V

Bit	Name	R/W	Reset Value	Function
				10: gm 7.5mA/V 11: gm 10mA/V
7:4	Reserved		-	保留
3:0	CANCKSEL[3:0]	RW	0x8	CAN 通讯时钟选择。 0000: PLL 时钟; 0001: PLL 时钟 2 分频; 0010: PLL 时钟 3 分频; 0011: PLL 时钟 4 分频; 0100: PLL 时钟 5 分频; 0101: PLL 时钟 6 分频; 0110: PLL 时钟 7 分频; 0111: PLL 时钟 8 分频; 1000: HSE 时钟; others: 无时钟;

### 5.3.16. RCC 寄存器映像

		Offset	Register	
0x4002 1000		RCC_CR	Reset Value	31
0x04	RCC_CFGR	0	USBPRE[2]	30
0x04	Reset Value	0	PLLMULL[5:4] 0 0 0 0 0 0	29
0x08	RCC_CIR	0	ADCPRE[2]	28
0x08	Reset Value	0	MCO[3:0] 0 0 0 0 0 0	27
0x0C	RCC_APB2RSTR	0	PLLRDY	26
0x0C	Reset Value	0	PLLON	25
0x10	RCC_APB1RSTR	0	HSICAL[7:0]	24
0x10	Reset Value	0	CSSC	23
0x14	RCC_AHB1ENR	0	HSICAL[7:0]	22
0x14	Reset Value	0	Reserved	21
0x18	RCC_APB2ENR	0	HSICAL[7:0]	20
0x18	Reset Value	0	HSICAL[7:0]	19

		Offset	Register	
			Reset Value	
0x1C	RCC_APB1ENR		CTCEN	31
		Reserved	Reserved	30
0x20	RCC_BDCR		PWREN	29
		Reserved	BKPN	28
0x24	RCC_CSR		PIRNSTF	27
		Reserved	CANEN	26
0x28	RCC_CFGR1		OBLRSTF	25
		Reserved	RMVF	24
0x2C	RCC_AHB1RSTR		SFTRSTF	23
		Reserved	I2C2EN	22
0x30	RCC_AHB2RSTR		PORRSTF	21
		Reserved	I2C1EN	20
0x34	RCC_AHB2ENR		LPWRRSTF	19
		Reserved	USART5EN	18
0x38	RCC_CFGR2		WWDRSTF	17
		Reserved	USART2EN	16
0x3C	RCC_CKTRIM		IWDGRSTF	15
		Reserved	SPI3EN	14
0x40	RCC_DCDR		RTSEN	13
		Reserved	SPI2EN	12
0x44	RCC_HSE		RTCSEL[1:0]	11
		Reserved	WWDGGEN	10
0x48	RCC_I2C		RTCSEL[0]	9
		Reserved	TIM14EN	8
0x4C	RCC_LSE		RTCSEL[1:0]	7
		Reserved	TIM13EN	6
0x50	RCC_MCO		RTCSEL[0]	5
		Reserved	TIM12EN	4
0x54	RCC_TIM1		RTCSEL[1:0]	3
		Reserved	TIM5EN	2
0x58	RCC_TIM2		RTCSEL[0]	1
		Reserved	TIM4EN	0
0x5C	RCC_TIM3		RTCSEL[1:0]	1
		Reserved	LSERDY	0
0x60	RCC_TIM4		RTCSEL[0]	0
		Reserved	LSEON	0

## 6. 备份寄存器 (BKP)

### 6.1. 简介

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

备份寄存器由备份域上电复位复位 (BPOR+BDRST)，且在检测到侵入事件时，清零备份寄存器内容。

此外，BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

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

- 通过设置寄存器 RCC\_APP1ENR 的 PWREN 和 BKOPEN 位来打开电源和备份接口的时钟
- 电源控制寄存器(PWR\_CR)的 DBP 位来使能对备份寄存器和 RTC 的访问。

#### 6.1.1. 主要特性

- 支持 84 字节数据后备寄存器
- 用来管理防侵入检测并具有中断功能的状态/控制寄存器
- 用来存储 RTC 校验值的校验寄存器。
- 在 PC13 引脚(当该引脚不用于侵入检测时)上输出 RTC 校准时钟，RTC 闹钟脉冲或者秒脉冲

### 6.2. 功能描述

#### 6.2.1. 侵入检测

当 TAMPER 引脚上的信号从 0 变成 1 或者从 1 变成 0(取决于备份控制寄存器 BKP\_CR 的 TPAL 位)，会产生一个侵入检测事件。侵入检测事件将所有数据备份寄存器内容清除。然而为了避免丢失侵入事件，侵入检测信号是边沿检测的信号与侵入检测允许位的逻辑‘与’，从而在侵入检测引脚被允许前发生的侵入事件也可以被检测到。

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

设置 BKP\_CSR 寄存器的 TPIE 位为‘1’，当检测到侵入事件时就会产生一个中断。在一个侵入事件被检测到并被清除后，侵入检测引脚 TAMPER 应该被禁止。然后，在再次写入备份数据寄存器前重新用 TPE 位启动侵入检测功能。这样，可以阻止软件在侵入检测引脚上仍然有侵入事件时对备份数据寄存器进行写操作。这相当于对侵入引脚 TAMPER 进行电平检测。

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

#### 6.2.2. RTC 校准

为方便测量，RTC 时钟可以经 64 分频输出到侵入检测引脚 TAMPER 上。通过设置 RTC 校验寄存器 (BKP\_RTCCR) 的 CCO 位来开启这一功能。

通过配置 CAL[6:0]位，此时钟可以最多减慢 121ppm。

### 6.2.2.1. 概述

实时时钟(RTC)精度是大多数嵌入式应用程序的要求，但由于外部环境-温度变化，时钟的晶体频率变化- RTC 精度可能不像预期的那样准确。

RTC 模块包含适合制造环境的数字时钟校准电路，允许应用程序补偿晶体和温度变化。

### 6.2.2.2. RTC 校准方式

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

- 温度变化
- 晶体变化

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

结构框图如下所示：

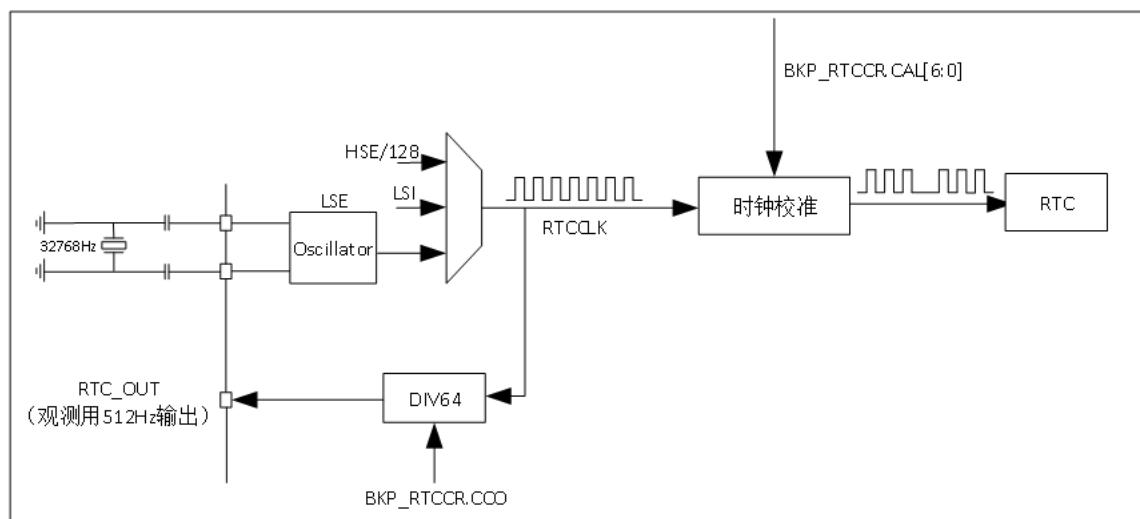


图 6-1 RTC 结构框图

注：引脚上输出的时钟是校准前的 RTC 时钟，因此校准不会改变其值。

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

下表显示了每个位在每个月(30 天)中实时表示的 ppm 和秒数。

表 6-1 实时 ppm 和秒数

校准值	四舍五入到最接近的 ppm	值以秒为单位，每月(30 天)四舍五入到最近的秒	校准值	四舍五入到最接近的 ppm	值以秒为单位，每月(30 天)四舍五入到最近的秒
0	0	0	64	61	158
1	1	2	65	62	161
2	2	5	66	63	163
3	3	7	67	64	166
4	4	10	68	65	168
5	5	12	69	66	171

校准值	四舍五入到最接近的 ppm	值以秒为单位, 每月 (30 天) 四舍五入到最近的秒	校准值	四舍五入到最接近的 ppm	值以秒为单位, 每月 (30 天) 四舍五入到最近的秒
6	6	15	70	67	173
7	7	17	71	68	176
8	8	20	72	69	178
9	9	22	73	70	180
10	10	25	74	71	183
11	10	27	75	72	185
12	11	30	76	72	188
13	12	32	77	73	190
14	13	35	78	74	193
15	14	37	79	75	195
16	15	40	80	76	198
17	16	42	81	77	200
18	17	44	82	78	203
19	18	47	83	79	205
20	19	49	84	80	208
21	20	52	85	81	210
22	21	54	86	82	213
23	22	57	87	83	215
24	23	59	88	84	218
25	24	62	89	85	220
26	25	64	90	86	222
27	26	67	91	87	225
28	27	69	92	88	227
29	28	72	93	89	230
30	29	74	94	90	232
31	30	77	95	91	235
32	31	79	96	92	237
33	31	82	97	93	240
34	32	84	98	93	242
35	33	87	99	94	245
36	34	89	100	95	247
37	35	91	101	96	250
38	36	94	102	97	252
39	37	96	103	98	255
40	38	99	104	99	257
41	39	101	105	100	260
42	40	104	106	101	262
43	41	106	107	102	264
44	42	109	108	103	267
45	43	111	109	104	269
46	44	114	110	105	272
47	45	116	111	106	274
48	46	119	112	107	277
49	47	121	113	108	279
50	48	124	114	109	282

校准值	四舍五入到最接近的 ppm	值以秒为单位, 每月(30 天)四舍五入到最近的秒	校准值	四舍五入到最接近的 ppm	值以秒为单位, 每月(30 天)四舍五入到最近的秒
51	49	126	115	110	284
52	50	129	116	111	287
53	51	131	117	112	289
54	51	133	118	113	292
55	52	136	119	113	294
56	53	138	120	114	297
57	54	141	121	115	299
58	55	143	122	116	302
59	56	146	123	117	304
60	57	148	124	118	307
61	58	151	125	119	309
62	59	153	126	120	311
63	60	156	127	121	314

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

由于晶体频率可能变化约 32.768 kHz, 一个解决方案可以考虑, 包括设置 RTC 预分频为 32766(而不是 32768)。因此, 晶体频率由 32768 改为 32766。这样, 可以补偿[32770,32766]范围内的晶体频率。

在本文档的其余部分中, 考虑的 RTC 预分频器值将为 32766。

### 6.2.2.3. 计算所需的校准量

为了确定在给定的应用程序中需要多少校准, 需要保留一种特别适合制造环境的方法。它涉及到 RTC 时钟输出模式的使用, 它从时钟分频链获得一个 512Hz 的信号。这个信号可以用来测量晶体振荡器的精度。

该方法可分为以下步骤:

- 启用低速外部振荡器(LSE), 选择 LSE 作为 RTC 时钟源, 然后启用 RTC 时钟。
- 在 RTC\_OUT 引脚上使 RTC 时钟输出的频率除以 64, 用于晶体频率测量。这是通过设置 BKP\_RTCCR 中的 CCO 位来实现的。
- 计算晶体频率偏差(ppm)。通过将 511.968 Hz 的测量偏差除以 511.968, 并将结果乘以 100 万, 就可以快速计算出 ppm 中的偏差。使用表 1 找到最近的校准值。此表是基于 ppm 表示的变化值的校准值的直接查找表。
- 在 RTC 校准寄存器中加载校准值以补偿晶体偏差。

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

例如, 如果在测试模式中测量的频率为 511.982 Hz, 则 $\delta$ 为 0.014。除以 511.968, 再乘以 100 万, 结果是 27.35 ppm。在这种情况下, 最接近的补偿值是 28。误差将从 27.35 ppm(每月~71 秒)减少到 0.65 ppm(每月~1.7 秒)。

注意:由于 RTC 校准是基于移除时钟周期, 它不能改善在短时间内的计数, 它只改善在长时间内的计数。例如, 在没有校准的情况下, 使用 RTC 计数 1/100 秒会比有校准的情况下更准确。由于校准周期在考虑的时间范围内可能发生或不发生, 因此结果值可能会发生显著变化。因此, 根据应用情况, 最好不要使用校准。

## 6.3. BKP 寄存器 (地址=0x40006C00)

注：BKP 寄存器可以半字（16 位）或者字（32 位）的方式访问。

### 6.3.1. 6.3.1 备份数据寄存器 (BKP\_DRx) (x=1...42)

Address offset: 0x04~0x28,0x40~0xBC

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

### 6.3.2. RTC 时钟校准寄存器 (BKP\_RTCCR)

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

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RW	-	Reserved
9	ASOS	RW	0	闹钟或者秒输出选择。 当设置了 ASOE 位，ASOS 位可以用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲还是闹钟脉冲信号。 0：输出 RTC 闹钟脉冲； 1：输出秒脉冲； 注：该位只能被备份区复位清零。
8	ASOE	RW	0	允许输出闹钟或者秒脉冲。 根据 ASOS 位配置，该位允许 RTC 闹钟或秒脉冲输出到 TAMPER 引脚上。 输出脉冲的宽度为一个 RTC 时钟的周期。设置了 ASOE 位时不能开启 TAMPER 的功能。 注：该位只能被备份区复位清零。
7	CCO	RW	0	校准时钟输出。 0：无影响； 1：在侵入检测引脚输出 RTC 64 分频时钟。 当 CCO 位置 1 后，必须关闭侵入检测功能以避免检测到无用的侵入信号。 注：当 VCC 供电断开时，该位清零。
6:0	CAL[6:0]	RW	0	校准值。

Bit	Name	R/W	Reset Value	Function
				校准值表示在每 $2^{20}$ 个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对 RTC 进行校准，以 $1000000/2^{20}$ ppm 的比例减慢时钟。 RTC 时钟可以被减慢 0~121ppm。

注：当 CCO 和 ASOE 同时配置时，ASOS 优先级高。

### 6.3.3. 备份控制寄存器 (BKP\_CR)

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.	TPAL	TPE													
														RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RW	-	Reserved
1	TPAL	RW	0	侵入检测 TAMPER 引脚有效电平。 0: 侵入检测 TAMPER 引脚上的高电平会清除所有数据备份寄存器 (TPE=1 时) ; 1: 侵入检测 TAMPER 引脚上的低电平会清除所有数据备份寄存器 (TPE=1 时) ; 注: 建议在 TPE 为 0 时改变 TPAL 的值, 否则会产生假的侵入事件。
0	TPE	RW	0	启动侵入检测 TAMPER 引脚。 0: 侵入检测 TAMPER 引脚作为通用 IO 口使用; 1: 侵入检测 TAMPER 引脚作为侵入检测使用;

### 6.3.4. 备份控制/状态寄存器 (BKP\_CSR)

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.	Res.	Res.	Res.	Res.	Res.	TIF	TEF	Res.	Res.	Res.	Res.	Res.	TPIE	CTI	CTE
						R	R						RW	W	W

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RW	-	Reserved
9	TIF	R	0	侵入中断标志。 当检测到侵入事件且 TPIE 位为 1 时, 此位由硬件置 1.通过向 CTI 位写 1 来清除此标志位 (同时也清除中断) 。如果 TPIE 位被清除, 此位也会被清除。 0: 无侵入中断; 1: 产生侵入中断;
8	TEF	R	0	侵入事件标志。 当检测到侵入事件时此位由硬件置 1.通过向 CTE 位写 1 可清除此标志位。 0: 无侵入事件; 1: 检测到侵入事件;

Bit	Name	R/W	Reset Value	Function
				注：侵入事件会复位所有的 BKP_DRx 寄存器。只要 TEF 为 1，所有的 BKP_DRx 寄存器就一直保持复位状态。当此位被置 1 时，若对 BKP_DRx 进行写操作，写入的值不会被保存。
7:3	Reserved	RW	-	Reserved
2	TPIE	RW	0	允许侵入 TAMPER 引脚中断。 0：禁止侵入检测中断； 1：允许侵入监测终端（BKP_CR 寄存器的 TPE 位也必须被置 1） 注 1：侵入中断无法将系统内核从低功耗模式唤醒； 注 2：仅当系统复位或者待机模式唤醒后才复位该位。
1	CTI	W	0	清除侵入检测中断。 此位只能写入，读出值为 0. 0：无效； 1：清除侵入检测中断和 TIF 侵入检测中断标志。
0	CTE	W	0	清除侵入检测事件。 此位只能写入，读出值为 0. 0：无效； 1：清除 TEF 侵入检测事件标志（并复位侵入检测器）。

### 6.3.5. BKP 寄存器映射

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4000C000	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.													
0x0400C000	Re set value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0400C004	BKP_D_R1	Res.										D[15:0]																					
0x0400C008	Re set value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0400C00C	BKP_D_R2	Res.										D[15:0]																					
0x0400C010	Re set value	0										RW																					
0x0400C014	BKP_D_R3	Res.										D[15:0]																					
0x0400C018	Re set value	0										RW																					



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Re set value																																
0x30	BKPCR																																
	Re set value																																
0x34	BKPSR																																
	Re set value																																
0x38	Res.																																
	Re set value																																
0x3C	Res.																																
	Re set value																																
0x40	BKDR11																																
	Re set value																																
0x44	BKDR12																																
	Re set value																																
0x48	BKDR13																																
	Re set value																																
0x4C	BKPD																																
	Re set value																																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	R1_4																																
	Reset value	0															RW														0		
0x54	B_K_P_D_R1_5	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x58	B_K_P_D_R1_6	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x5c	B_K_P_D_R1_7	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x60	B_K_P_D_R1_8	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x64	B_K_P_D_R1_9	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x68	B_K_P_D_R2_0	Res.															D[15:0]														RW		
	Reset value	0															0														0		
0x68	B_K_P_D	Res.															D[15:0]														0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	R2_1																																
	Reset value	0																RW															
0x6c	B_K_P_D_R2_2	Res.																D[15:0]															
	Reset value	0																RW															
0x70	B_K_P_D_R2_3	Res.																D[15:0]															
	Reset value	0																RW															
0x74	B_K_P_D_R2_4	Res.																D[15:0]															
	Reset value	0																RW															
0x78	B_K_P_D_R2_5	Res.																D[15:0]															
	Reset value	0																RW															
0x7C	B_K_P_D_R2_6	Res.																D[15:0]															
	Reset value	0																RW															
0x80	B_K_P_D_R2_7	Res.																D[15:0]															
	Reset value	0																RW															
0x84	B_K_P_D	Res.																D[15:0]															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R2 8	Re se t val ue	0															RW																
		0															0																
0x 88	B K P _D R2 9	Res.															D[15:0]																
	Re se t val ue	0															RW																
0x 8C	B K P _D R3 0	Res.															D[15:0]																
	Re se t val ue	0															RW																
0x 90	B K P _D R3 1	Res.															D[15:0]																
	Re se t val ue	0															RW																
0x 94	B K P _D R3 2	Res.															D[15:0]																
	Re se t val ue	0															RW																
0x 98	B K P _D R3 3	Res.															D[15:0]																
	Re se t val ue	0															RW																
0x 9C	B K P _D R3 4	Res.															D[15:0]																
	Re se t val ue	0															RW																
0X A0	B K P _D	Res.															D[15:0]																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R3 5	B K P _D R3 6	0														RW														0			
	Re se t val ue	0														0														0			
0x A4	B K P _D R3 6	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0X A8	B K P _D R3 7	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0x AC	B K P _D R3 8	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0X B0	B K P _D R3 9	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0x B4	B K P _D R4 0	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0X B8	B K P _D R4 1	Res.														D[15:0]														RW			
	Re se t val ue	0														0														0			
0x BC	B K	Res.														D[15:0]														0			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
P D I R4 2	Re se t val ue																																		
0																RW																0			

## 7. CRC 计算单元 (CRC)

### 7.1. 简介

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

在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。

### 7.2. CRC 主要特征

- 使用 CRC-32(以太网)多项式： 0x4C11DB7  
—  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1$
- 一个32位数据寄存器用于输入 / 输出
- CRC 计算时间： 4个 AHB 时钟周期(HCLK)
- 通用8位寄存器(可用于存放临时数据)

### 7.3. CRC 功能描述

#### 7.3.1. CRC 框图

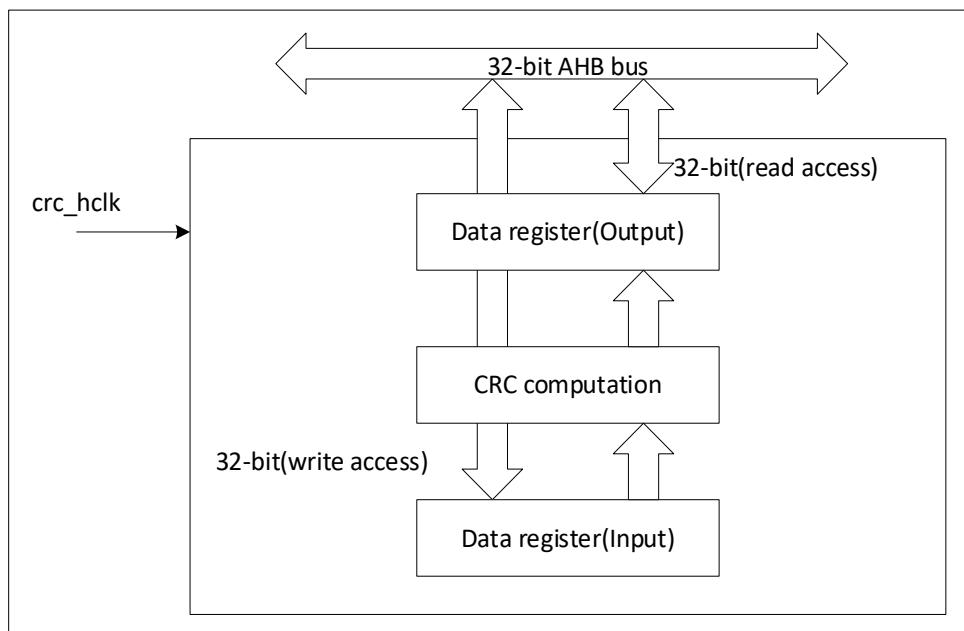


图 7-1 CRC 计算单元框图

CRC 计算单元含有 1 个 32 位数据寄存器：

对该寄存器进行写操作时，作为输入寄存器，可以输入要进行 CRC 计算的新数据。

对该寄存器进行读操作时，返回上一次 CRC 计算的结果。

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

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。

### 7.4. CRC 寄存器

必须以 32 位的方式对这些寄存器进行访问。

### 7.4.1. 数据寄存器 (CRC\_DR)

Address offset:0x00

Reset value:0xFFFF FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 0	DR	RW	32'hFFFFFF	数据寄存器位。 写入 CRC 计数器的新数据时，作为输入寄存器。 读取时返回 CRC 计算的结果。

### 7.4.2. 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
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[7:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31: 8	保留	RES	-	保留
7: 0	IDR	RW	8'h0	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。

### 7.4.3. 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
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET															
w															

Bit	Name	R/W	Reset Value	Function
31: 1	保留	RES	-	保留
0	RESET	W	0	软件置位后将会复位 CRC 模块，数据寄存器会加载 CRC_INIT 寄存器的值。软件只能写 1，由硬件清零。

#### 7.4.4. CRC 寄存器映像

		Offset		Register
		CRC_DR	Reset value	
8	0x0	0x0	0x0	31
	Res.	1	1	30
	Res.		1	29
	Res.		1	28
	Res.		1	27
	Res.		1	26
	Res.		1	25
	Res.		1	24
	Res.		1	23
	Res.		1	22
	Res.		1	21
	Res.		1	20
	Res.		1	19
	Res.		1	18
	Res.		1	17
	Res.		1	16
	Res.		1	15
	Res.		1	14
	Res.		1	13
	Res.		1	12
	Res.		1	11
	Res.		1	10
	Res.		1	9
	Res.		1	8
	Res.	0	1	7
	Res.	0	1	6
	Res.	0	1	5
	Res.	0	1	4
	Res.	0	1	3
	Res.	0	1	2
	Res.	0	1	1
	Res.	0	0	0

DR[31:0]

IDR[7:0]

## 8. 通用 I/O (GPIO)

### 8.1. 简介

每个通用 I/O 端口包括 4 个 32 位配置寄存器 (GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR 和 GPIOx\_PUPDR)、2 个 32 位数据寄存器 (GPIOx\_IDR 和 GPIOx\_ODR)、1 个 32 位置位/复位寄存器 (GPIOx\_BSRR)、1 个 32 位锁定寄存器 (GPIOx\_LCKR) 和 2 个 32 位复用功能选择寄存器 (GPIOx\_AFRH 和 GPIOx\_AFRL)。

### 8.2. 主要特征

- 输出状态：推挽或开漏 + 上拉/下拉
- 从输出数据寄存器 (GPIOx\_ODR) 或外设（复用功能输出）输出数据
- 可为每个 I/O 选择不同的速度
- 输入状态：浮空、上拉/下拉、模拟
- 将数据输入到输入数据寄存器 (GPIOx\_IDR) 或外设（复用功能输入）
- 置位和复位寄存器 (GPIOx\_BSRR)，对 GPIOx\_ODR 具有按位写权限
- 锁定机制 (GPIOx\_LCKR)，可冻结 I/O 配置
- 模拟功能
- 复用功能输入/输出选择寄存器（一个 I/O 最多可具有 16 个复用功能）
- 快速翻转，每次翻转最快只需要两个时钟周期
- 引脚复用非常灵活，允许将 I/O 引脚用作 GPIO 或多种外设功能中的一种

### 8.3. 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
- 模拟功能
- 具有上拉或下拉功能的开漏输出
- 具有上拉或下拉功能的推挽输出
- 具有上拉或下拉功能的复用功能推挽
- 具有上拉或下拉功能的复用功能开漏

每个 I/O 端口位均可自由编程，但 I/O 端口寄存器必须按 32 位字、半字或字节进行访问。GPIOx\_BSRR 寄存器旨在实现对 GPIO ODR 寄存器进行原子读取/修改访问。这样便可确保在读取和修改访问之间发生中断请求也不会有问题。

下图显示了 5 V 容忍和基本 I/O 端口位的基本结构以及可能的端口位配置方案。

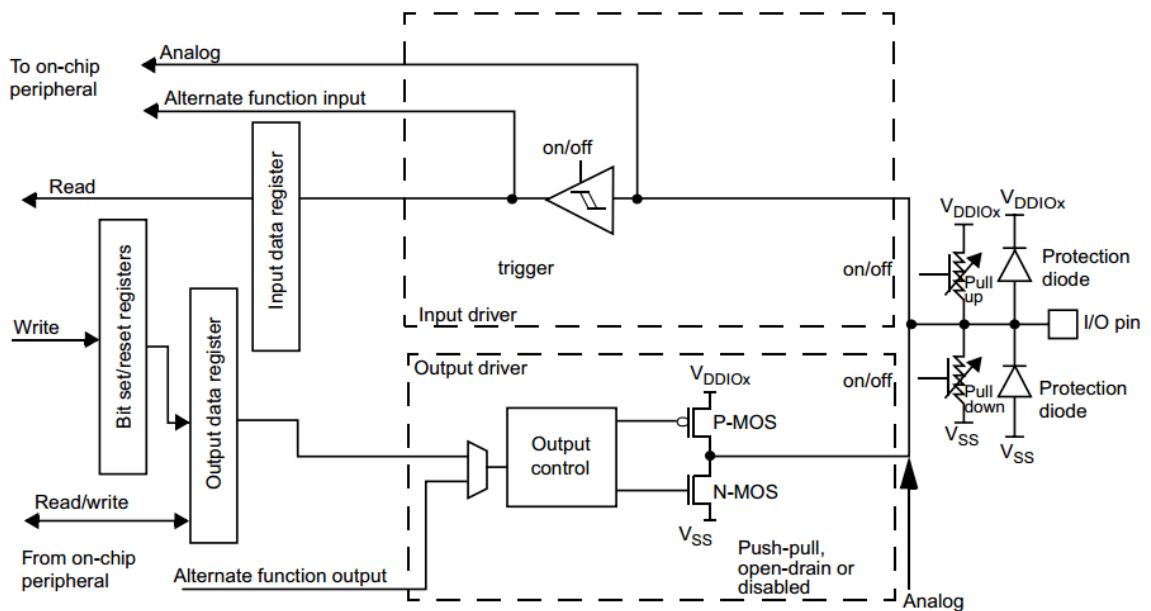


图 8-1 普通 I/O 端口位的基本结构

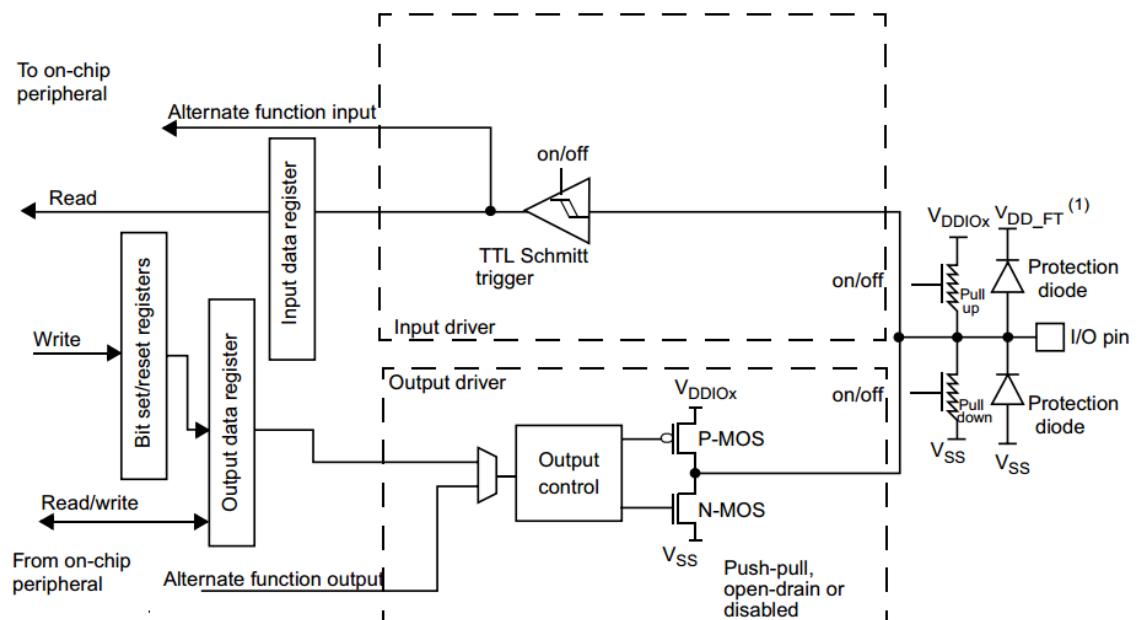


图 8-2 5V 容忍 I/O 端口位基本结构

表 8-1 端口位配置表

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP+PU
	0		1	0	GP output	PP+PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD+PU
	1		1	0	GP output	OD+PD
	1		1	1	Reserved(GP output OD)	
	10		0	0	AF	PP
10	0	SPEED[1:0]	0	1	AF	PP+PU
	0		1	0	AF	PP+PD
	0		1	1	Reserved	
	1		0	0	AF	OD
	1					

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
	1			0	1	AF	OD+PU
	1			1	0	AF	OD+PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved(input floating)	
11	x	x	x	0	0	Input/Output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/Output	Analog,PD
	x	x	x	1	1	Reserved	

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

### 8.3.1. 通用 I/O (GPIO)

在复位期间及复位刚刚完成后，复用功能尚未激活，I/O 端口被配置为输入浮空模式模拟模式。复位后，调试引脚处于复用功能上拉/下拉状态：

- PA15：JTDI 处于上拉状态
- PA14：JTCK/SWCLK 处于下拉状态
- PA13：JTMS/SWDAT 处于下拉状态 (F4是下拉, G4是上拉, F1是上拉)
- PB4：NJTRST 处于上拉状态
- PB3：JTDO 处于浮空状态

当引脚配置为输出后，写入到输出数据寄存器 (GPIOx\_ODR) 的值将在 I/O 引脚上输出。可以在推挽模式下或开漏模式下使用输出驱动器（输出 0 时仅激活 N-MOS）。

输入数据寄存器 (GPIOx\_IDR) 每隔 1 个 AHB1 时钟周期捕获一次 I/O 引脚的数据。

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

### 8.3.2. I/O 引脚复用器和映射

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

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

- 完成复位后，所有 I/O 都会连接到系统的复用功能0 (AF0)。所有的 I/O 通过 GPIOx\_MODER 寄存器配置为复用功能。
- 每个引脚的复用功能
- 除了这种灵活的 I/O 复用架构之外，各外设还可以将复用功能映射到不同 I/O 引脚，这可以优化小型封装中可用外设的数量。

要将 I/O 配制成所需功能，请按照以下步骤操作：

- 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
  - 在 GPIOx\_AFRL 或 GPIOx\_AFRH 寄存器中，将 I/O 连接到所需 AFx

- 通过 GPIOx\_OTYPER、GPIOx\_PUPDR 和 GPIOx\_OSPEEDER 寄存器，分别选择类型、上拉/下拉以及输出速度
- 在 GPIOx\_MODER 寄存器中将所需 I/O 配置为复用功能
- 额外功能
  - 对于 ADC，需要通过寄存器 GPIOx\_MODER 配置相应的 I/O 为模拟模式，并在 ADC 中配置相应功能
  - 对于额外功能 RTC，WKUPx 和外部晶体接口，需要在 RTC 和 PWR 模块配置相应功能。这些功能，优先于 GPIO 中的 AF 选择寄存器。

### 8.3.3. I/O 端口控制寄存器

每个 GPIO 有 4 个 32 位存储器映射的控制寄存器 (GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR、GPIOx\_PUPDR)，可配置多达 16 个 I/O。GPIOx\_MODER 寄存器用于选择 I/O 方向 (输入、输出、AF、模拟)。GPIOx\_OTYPER 和 GPIOx\_OSPEEDR 寄存器分别用于选择输出类型 (推挽或开漏) 和速度 (无论采用哪种 I/O 方向，都会直接将 I/O 速度引脚连接到相应的 GPIOx\_OSPEEDR 寄存器位)。无论采用哪种 I/O 方向，GPIOx\_PUPDR 寄存器都用于选择上拉/下拉。

### 8.3.4. I/O 端口数据寄存器

每个 GPIO 都具有 2 个 16 位数据寄存器：输入和输出数据寄存器 (GPIOx\_IDR 和 GPIOx\_ODR)。GPIOx\_ODR 用于存储待输出数据，可对其进行读/写访问。通过 I/O 输入的数据存储到输入数据寄存器 (GPIOx\_IDR) 中，它是一个只读寄存器。

### 8.3.5. I/O 数据位操作

置位复位寄存器 (GPIOx\_BSRR) 是一个 32 位寄存器，它允许应用程序在输出数据寄存器(GPIOx\_ODR) 中对各个单独的数据位执行置位和复位操作。置位复位寄存器的大小是 GPIOx\_ODR 的二倍。

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

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

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

在对 GPIOx\_ODR 进行位操作时，软件无需禁止中断：在一次原子 AHB1 写访问中，可以修改一个或多个位。

### 8.3.6. GPIO 锁定机制

通过将特定的写序列应用到 GPIOx\_LCKR 寄存器，可以冻结 GPIO 控制寄存器。冻结的寄存器包括 GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR、GPIOx\_PUPDR、GPIOx\_AFRL 和 GPIOx\_AFRH。

要对 GPIOx\_LCKR 寄存器执行写操作，必须应用特定的写/读序列。当正确的 LOCK 序列应用到此寄存器的第 16 位后，会使用 LCKR[15:0] 的值来锁定 I/O 的配置 (在写序列期间，LCKR[15:0] 的值必须相同)。将 LOCK 序列应用到某个端口位后，在执行下一次复位之前，将无法对该端口位的值进行修改。每个 GPIOx\_LCKR 位都会冻结控制寄存器 (GPIOx\_MODER、GPIOx\_OTYPER、GPIOx\_OSPEEDR、GPIOx\_PUPDR、GPIOx\_AFRL 和 GPIOx\_AFRH) 中的对应位。

LOCK 序列只能通过对 GPIOx\_LCKR 寄存器进行字 (32 位长) 访问的方式来执行，因为 GPIOx\_LCKR 的第 16 位必须与 [15:0] 位同时置位。

具体操作参考 GPIOx\_LCKR 寄存器描述。

### 8.3.7. I/O 复用功能输入/输出模式配置

有两个寄存器可用来从每个 I/O 可用的 16 个复用功能输入/输出中进行选择。借助这些寄存器，可根据应用程序的要求将某个复用功能连接到其它某个引脚。

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

注意：对于每个 I/O 而言，应用程序一次只能为其选择一个可用的外设功能。

### 8.3.8. 外部中断线/唤醒线

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

### 8.3.9. I/O 输入配置

对 I/O 端口进行编程作为输入时：

- 输出缓冲器被关闭
- 施密特触发器输入被打开
- 根据 GPIOx\_PUPDR 寄存器中的值决定是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 AHB1 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

### 8.3.10. 输出配置

对 I/O 端口进行编程作为输出时：

- 输出缓冲器被打开：
  - 开漏模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”会使端口保持高阻态 (Hi-Z) (P-MOS 始终不激活)。
  - 推挽模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”可激活 P-MOS。
- 施密特触发器输入被打开
- 根据 GPIOx\_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态
- 对输出数据寄存器的读访问可获取最后的写入值

### 8.3.11. 复用功能配置

对 I/O 端口进行编程作为复用功能时：

- 可将输出缓冲器配置为开漏或推挽
- 输出缓冲器由来自外设的信号驱动（发送器使能和数据）
- 施密特触发器输入被打开
- 根据 GPIOx\_PUPDR 寄存器中的值决定是否打开弱上拉电阻和下拉电阻
- 输入数据寄存器每隔 1 个 AHB 时钟周期对 I/O 引脚上的数据进行一次采样
- 对输入数据寄存器的读访问可获取 I/O 状态

### 8.3.12. 模拟配置

对 I/O 端口进行编程作为模拟配置时：

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

注意：在模拟配置中，I/O 引脚不能为 5 V 容忍。

### 8.3.13. HSE 或者 LSE 引脚配置为 GPIO

系统复位后，默认模式下，HSE 和 LSE 是关闭的，此时 HSE 和 LSE 对应的引脚可以作为普通 GPIO 使用。

通过 RCC\_CSR 寄存器使能 HSE 或者 LSE 后，对应引脚只能配置为晶振功能。优先级高于 GPIO。

当晶振被配置为外部输入时钟模式时，OSC\_IN 或 OSC32\_IN 被作为时钟输入，OSC\_OUT 或 OSC32\_OUT 可以被作为普通 I/O 使用。

### 8.3.14. BKP 区域 GPIO 使用

当 VCCD 断电后，PC13/PC14/PC15 将不在拥有 GPIO 功能。如果 RTC 没有使用这几个引脚，这些引脚被设置为模拟模式。

## 8.4. 寄存器描述

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

### 8.4.1. GPIO 端口模式寄存器 (GPIOx\_MODER) (x=A..E)

Address offset: 0x00

Reset value: 0xABFF\_FFFF (端口 A)

Reset value: 0xFFFF\_FEBF (端口 B)

Reset value: 0xFFFF\_FFFF (其他)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODERO[1:0]	
rw	rw	rw	rw	rw	rw										

Bit	Name	R/W	Reset Value	Function
31: 0	MODEy[1:0]	RW		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)

### 8.4.2. GPIO 端口输出类型寄存器 (GPIOx\_OTYPER) (x=A..E)

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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw															

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	OT	RW		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

#### 8.4.3. GPIO 端口输出速度寄存器 (GPIOx\_OSPEEDR) (x= A..E)

Address offset:0x08

Reset value:0x0C00 0000 (端口 A)

Reset value:0x0000 0000 (其他)

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										
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										

Bit	Name	R/W	Reset Value	Function
31:0	OSPEEDy[1:0]	RW		Y = 15..0 软件配置 I/O 口的输出速度 00: 非常低 01: 低速 10: 高速 11: 非常高

#### 8.4.4. GPIO 端口上拉/下拉寄存器 (GPIOx\_PUPDR) (x= A..E)

Address offset:0x0C

Reset value:0x6400 0000 (端口 A)

Reset value:0x0000 0100 (端口 B)

Reset value:0x0000 0000 (其他)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw										
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										

Bit	Name	R/W	Reset Value	Function
31:0	PUPDy [1:0]	RW		Y = 15..0 软件配置 I/O 口上拉或者下拉 00: 无上下拉

Bit	Name	R/W	Reset Value	Function
				01: 上拉 10: 下拉 11: 保留

#### 8.4.5. GPIO 端口输入数据寄存器 (GPIOx\_IDR) (x= A..E)

Address offset:0x10

Reset value:0x0000 XXXX

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

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

#### 8.4.6. GPIO 端口输出数据寄存器 (GPIOx\_ODR) (x= A..E)

Address offset:0x14

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved			
15: 0	ODy[1:0]	RW		y = 15..0 软件可读可写。 说明：对 GPIOx_BSRR or GPIOx_BRR registers. (x=A,B,C,D,E), 可以分别对各个 ODR 位进行独立的设置/清除。

#### 8.4.7. GPIO 端口置位/复位寄存器 (GPIOx\_BSRR) (x= A..E)

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		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		y = 15..0

Bit	Name	R/W	Reset Value	Function
				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

#### 8.4.8. GPIO 端口配置锁定寄存器 (GPIOx\_LCKR) (x= A..E)

当执行正确的写序列设置了 bit16 (LCKR) 时, 该寄存器用来锁定端口位的配置。bit[15:0]用于锁定 GPIO 端口的配置。在规定的写入操作期间, 不能改变 LCKR[15:0]。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.															LCK16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	LCKK	RW		该位可随时读出, 它只能通过锁键写入序列修改 0: 端口配置锁键位未激活 1: 端口配置锁键位被激活, 下次系统复位前 GPIOx_LCKR 寄存器被锁定 LOCK key write sequence: 锁键的写入时序: 写 1->写 0->写 1->读 0->读 1, 最后一个读可省略, 但 可以用来确认锁键已被激活。 注: 在操作锁键的写入时序是, 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。
15: 0	LCKy	RW		y = 15..0 这些位可读可写但只能在 LCKK 位为 0 是写入。 0: 不锁定端口的配置 1: 锁定端口配置

#### 8.4.9. GPIO 复用功能低寄存器 (GPIOx\_AFRL) (x= A..E)

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												
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												

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] ((y= 7 to 0))	RW		软件可写这些位配置复用功能 I/O AFSELy 选择: 0000:AF0 0001:AF1 1000: AF8 1001: AF9

Bit	Name	R/W	Reset Value	Function
				0010:AF2 0011:AF3 0100:AF4 0101:AF5 0110:AF6 0111:AF7 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15

#### 8.4.10. GPIO 复用功能高寄存器 (GPIOx\_AFRH) (x= A..E)

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												
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												

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] ((y= 8 to 15))	RW		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

#### 8.4.11. GPIO 端口位复位寄存器 (GPIOx\_BRR) (x= A..E)

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	Reserved			
15:0	Bry	RW		y = 15..0 这些位软件可写, 读出来返回值是 0 0: 对对应的 Ody 位不产生影响 1: 清除对应的 Ody 位

#### 8.4.12. GPIO 寄存器地址映射

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



		Offset		Register	
8	1	0	1	0	1
0	x	GPIOx_BSRR(x=A,B,F)	Res.	0	BR15
1	C	Reset value	Res.	0	BR14
2	x	GPIOx_LCKR(x=A,B,F)	Res.	0	BR13
3	0	Reset value	Res.	0	BR12
4	x	GPIOx_AFRL(x=A,B,F)	Res.	0	BR11
5	2	AFSEL7 [3:0]	AFSEL6 [3:0]	AFSEL5 [3:0]	AFSEL4 [3:0]
6	0	0	0	0	0
7	x	Reset value	0	0	0
8	2	GPIOx_AFRH(x=A,B,F)	AFSEL7 [3:0]	AFSEL6 [3:0]	AFSEL5 [3:0]
9	0	0	0	0	0
10	x	Reset value	0	0	0
11	2	GPIOx_BRR(x=A,B,F)	AFSEL7 [3:0]	AFSEL6 [3:0]	AFSEL5 [3:0]
12	0	0	0	0	0
13	x	Reset value	0	0	0
14	0	0	0	0	0
15	2	GPIOx_BSRR(x=A,B,F)	BR15	0	BS15
16	0	0	0	0	BS14
17	x	Reset value	0	0	BS13
18	2	GPIOx_BRR(x=A,B,F)	BR14	0	BS12
19	0	0	0	0	BS11
20	x	Reset value	0	0	BS10
21	0	0	0	0	BS9
22	x	GPIOx_BSRR(x=A,B,F)	BR13	0	BS8
23	0	0	0	0	BS7
24	x	Reset value	0	0	BS6
25	0	0	0	0	BS5
26	x	GPIOx_BSRR(x=A,B,F)	BR12	0	BS4
27	0	0	0	0	BS3
28	x	Reset value	0	0	BS2
29	0	0	0	0	BS1
30	x	GPIOx_BSRR(x=A,B,F)	BR11	0	BS0

## 9. 系统配置控制器 (SYSCFG)

### 9.1. 概述

SYSCFG 模块主要完成如下功能：

- I2C 类型 IO noise filter 控制使能
- 所有 IO noise filter 控制使能
- EXTI IO select 控制
- 根据不同 boot 模式，映射初始程序区
- DMA 外设通带选择控制
- TIMERS breakin 和 ETR 控制

### 9.2. SYSCFG 寄存器(baseaddr=0x40010000)

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

该寄存器用作存储器和 DMA 请求 remap 和控制特殊 IO 功能的具体配置。

有两位用作配置存储器地址 0x0000 0000 访问的种类。这两位用来选择软件的物理 remap，并 bypass 掉硬件 BOOT 选择。在复位后，这些位使用被实际 boot 模式配置的值。

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.	Res.				Res.	Res.	I2C_PB12	I2C_PB11	I2C_PB10	I2C_PB9	I2C_PB8	I2C_PB7	I2C_PB6	I2C_PB5	
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.															MEM_MODE [1:0]
															RW

Bit	Name	R/W	Reset Value	Function
31:24	Reserved	RW	-	可读可写
23: 16	I2C_IOx_EIIC	RW	0	I2C 相关 IO 的模拟滤波使能控制 PB12~PB5 0: 模拟滤波关闭 1: 模拟滤波使能
15: 2	Reserved	RW		可读可写
1:0	MEM_MODE [1:0]	RW	0	Memory mapping 选择位 软件置位，软件清零。他们控制存储器的 0x0000 0000 地址的 mapping。 在复位后，这些位采用实际启动模式配置值。 00: Main flash, mapped 在 0x0000 0000 01: System flash, mapped 在 0x0000 0000 10: ESMC, mapped 在 0x0000 0000 11: SRAM, mapped 在 0x0000 0000

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

Address offset: 0x04

Reset value: 0x0000

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

Res.																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R es ·	R es ·	R es ·	R es ·	ADC2_ETRGRE G_REMAP	ADC2_ETRGIN J_REMAP	ADC1_ETRGRE G_REMAP	ADC1_ETRGIN J_REMAP	R es ·	R es ·	R es ·	R es ·	R es ·	PV D_ LO CK	R es ·	LOC KUP _LO CK	
														RW		RW

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	-	-	Reserved
11	ADC2_ETRGREG_REMAP	RW	0	ADC2 规则转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与 ADC2 规则转换外部触发相连的触发输入。当该位置'0'时，ADC2 规则转换外部触发与 EXTI11 相连；当该位置'1'时，ADC2 规则转换外部触发与 TIM8_TRGO 相连。
10	ADC2_ETRGINJ_REMAP	RW	0	ADC2 注入转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与 ADC2 注入转换外部触发相连的触发输入。当该位置'0'时，ADC2 注入转换外部触发与 EXTI15 相连；当该位置'1'时，ADC2 注入转换外部触发与 TIM8 通道 4 相连。
9	ADC1_ETRGREG_REMAP	RW	0	ADC1 规则转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与 ADC1 规则转换外部触发相连的触发输入。当该位置'0'时，ADC1 规则转换外部触发与 EXTI11 相连；当该位置'1'时，ADC1 规则转换外部触发与 TIM8_TRGO 相连。
8	ADC1_ETRGINJ_REMAP	RW	0	ADC1 注入转换外部触发重映射 该位可由软件置'1'或置'0'。它控制与 ADC1 注入转换外部触发相连的触发输入。当该位置'0'时，ADC1 注入转换外部触发与 EXTI15 相连；当该位置'1'时，ADC1 注入转换外部触发与 TIM8 通道 4 相连。
7:3	Reserved	-	-	Reserved
2	PVD_LOCK	RW	0	PVD Lock 使能位 软件置位，系统复位清零。它可以被用作使能和锁定 PVD 连接给 TIM1 的刹车输入，也锁定 PWR_CR 寄存器的 PVDE。 0: PVD 中断不与 TIM1 的刹车输入连接。PVDE 位可以被应用写入。 1: PVD 中断与 TIM1 的刹车输入连接。PVDE 位只读。
1	Reserved	-	-	Reserved
0	LOCKUP_LOCK	RW		Cortex-M4F LOCKUP 位的使能位 软件置位，系统复位清零。它可以使能和锁定 Cortex-M4F 的 LOCKUP(hardfault) 输出给 TIM1 和 TIM8 的刹车输入。 0: Cortex-M4F 的 LOCKUP 输出不与 TIM1 和 TIM8 的刹车输入连接； 1: Cortex-M4F 的 LOCKUP 输出与 TIM1 和 TIM8 的刹车输入连接；

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

Address offset: 0x08

Reset value: 0x7F7F\_7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	DMA4_MAP							Res.	DMA3_MAP							
	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.	DMA2_MAP							Res.	DMA1_MAP							
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA4_MAP	RW	7'h7f	DMA1 通道 4 映射。

Bit	Name	R/W	Reset Value	Function
				见 DMA1_MAP 描述。
23	Res	-	-	Reserved
22:16	DMA3_MAP	RW	7'h7f	DMA1 通道 3 映射。 见 DMA1_MAP 描述。
15	Res	-	-	Reserved
14:8	DMA2_MAP	RW	7'h7f	DMA1 通道 2 映射。 见 DMA1_MAP 描述。
7	Res	-	-	Reserved
6:0	DMA1_MAP	RW	7'h7f	DMA1 通道 1 映射。 0000000: ADC1; 0000001: ADC2; 0000010: ADC3; 0000101: SPI1_RD; 0000110: SPI1_WR; 0000111: SPI2_RD; 0001000: SPI2_WR; 0001001: SPI3_RD; 0001010: SPI3_WR; 0001011: USART1_RD; 0001100: USART1_WR; 0001101: USART2_RD; 0001110: USART2_WR; 0001111: USART3_RD; 0010000: USART3_WR; 0010001: USART4_RD; 0010010: USART4_WR; 0010011: USART5_RD; 0010100: USART5_WR; 0010101: I2C1_RD; 0010110: I2C1_WR; 0010111: I2C2_RD; 0011000: I2C2_WR; 0011001: TIM1_CH1; 0011010: TIM1_CH2; 0011011: TIM1_CH3; 0011100: TIM1_CH4; 0011101: TIM1_COM; 0011110: TIM1_TRG; 0011111: TIM1_UP; 0100000: TIM2_CH1; 0100001: TIM2_CH2; 0100010: TIM2_CH3; 0100011: TIM2_CH4; 0100100: TIM2_UP; 0101001: TIM3_TRIG; 0101010: TIM4_CH1; 0101011: TIM4_CH2; 0101100: TIM4_CH3; 0101101: TIM4_UP; 0101110: TIM5_CH1; 0101111: TIM5_CH2; 0110000: TIM5_CH3; 0110001: TIM5_CH4; 0110010: TIM5_UP; 0110011: TIM5_TRIG; 0110100: TIM6; 0110101: TIM7; 0110110: TIM8_CH1; 0110111: TIM8_CH2; 0111000: TIM8_CH3; 0111001: TIM8_CH4; 0111010: TIM8_COM; 0111011: TIM8_TRG; 0111100: TIM8_UP; 0111101: TIM2_TRIG; 0111110: TIM3_CH2; 0111111: TIM4_CH4; 1000000: TIM4_TRIG; 1000001: ESMC_RX; 1000010: ESMC_RX; 1000011: SDIO; 1000100: USB; Others: 保留

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

Address offset: 0x0C

Reset value: 0x7F7F\_7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	DMA8_MAP								Res.	DMA7_MAP							
	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.	DMA6_MAP								Res.	DMA5_MAP							
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA4_MAP	RW	7'h7f	DMA2 通道 1 映射。 见 DMA1_MAP 描述。
23	Res	-	-	Reserved
22:16	DMA7_MAP	RW	7'h7f	DMA1 通道 7 映射。 见 DMA1_MAP 描述。
15	Res	-	-	Reserved
14:8	DMA6_MAP	RW	7'h7f	DMA1 通道 6 映射。 见 DMA1_MAP 描述。
7	Res	-	-	Reserved
6:0	DMA5_MAP	RW	7'h7f	DMA1 通道 5 映射。 见 DMA1_MAP 描述。

### 9.2.5. SYSCFG 配置寄存器 5 (SYSCFG\_CFGR5)

Address offset: 0x10

Reset value: 0x7F7F\_7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	DMA12_MAP								Res.	DMA11_MAP							
	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.	DMA10_MAP								Res.	DMA9_MAP							
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA12_MAP	RW	7'h7f	DMA2 通道 5 映射。 见 DMA1_MAP 描述。
23	Res	-	-	Reserved
22:16	DMA11_MAP	RW	7'h7f	DMA2 通道 4 映射。 见 DMA1_MAP 描述。
15	Res	-	-	Reserved
14:8	DMA10_MAP	RW	7'h7f	DMA2 通道 3 映射。 见 DMA1_MAP 描述。
7	Res	-	-	Reserved
6:0	DMA9_MAP	RW	7'h7f	DMA2 通道 2 映射。 见 DMA1_MAP 描述。

### 9.2.6. 外部中断配置寄存器 1 (SYS\_EXTICR1)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx 配置(x = 0 ... 3) (EXTIx configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参考 EXTI 外部中断事件映射 0000: PA[x]引脚 0001: PB[x]引脚 0010: PC[x]引脚 0011: PD[x]引脚 0100: PE[x]引脚

## 9.2.7. 外部中断配置寄存器 2 (SYS\_EXTICR2)

Address offset:0x18

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx 配置(x = 4 ... 7) (EXTIx configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参考 EXTI 外部中断事件映射。 0000: PA[x]引脚 0001: PB[x]引脚 0010: PC[x]引脚 0011: PD[x]引脚 0100: PE[x]引脚

## 9.2.8. 外部中断配置寄存器 3 (SYS\_EXTICR3)

Address offset:0x1C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx 配置(x = 8 ... 11) (EXTIx configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参考 EXTI 外部中断事件映射。 0000: PA[x]引脚

Bit	Name	R/W	Reset Value	Function
				0001: PB[x]引脚 0010: PC[x]引脚 0011: PD[x]引脚 0100: PE[x]引脚

### 9.2.9. 外部中断配置寄存器 4 (SYS\_EXTICR4)

Address offset:0x20

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx 配置(x = 12 ... 15) (EXTIx x configuration) 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参考 EXTI 外部中断事件映射。 0000: PA[x]引脚 0001: PB[x]引脚 0010: PC[x]引脚 0011: PD[x]引脚 0100: PE[x]引脚

### 9.2.10. GPIOA 滤波使能 (PA\_ENS)

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PA_ENS[x]	RW	0x0000	噪声滤波器使能，高电平有效 0: 噪声滤波器旁路 1: 启用噪声滤波器

### 9.2.11. GPIOB 滤波使能 (PB\_ENS)

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			

Bit	Name	R/W	Reset Value	Function
15:0	PA_ENS[x]	RW	0x0000	噪声滤波器使能, 高电平有效 0: 噪声滤波器旁路 1: 启用噪声滤波器

### 9.2.12. GPIOC 滤波使能 (PC\_ENS)

Address offset:0x2C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PC_ENS[x]	RW	0x0000	噪声滤波器使能, 高电平有效 0: 噪声滤波器旁路 1: 启用噪声滤波器

### 9.2.13. GPIOD 滤波使能 (PD\_ENS)

Address offset:0x30

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PD_ENS[x]	RW	0x0000	噪声滤波器使能, 高电平有效 0: 噪声滤波器旁路 1: 启用噪声滤波器

### 9.2.14. GPIOE 滤波使能 (PE\_ENS)

Address offset:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	-
15:0	PE_ENS[x]	RW	0x0000	噪声滤波器使能, 高电平有效 0: 噪声滤波器旁路 1: 启用噪声滤波器

### 9.2.15. GPIO 模拟通道使能 (GPIO\_ENA)

Address offset:0x38

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	-	-	-
21:16	PC_ENA	RW	0x00	PC[5: 0]模拟输入使能, 高电平有效
15:10	Reserved	-	-	-
9:8	PB_ENA	RW	0x00	PB[1:0] 模拟输入使能, 高电平有效
7:0	PA_ENA	RW	0x00	PA[7:0] 模拟输入使能, 高电平有效

## 9.2.16. Timer 时钟扩展控制 (TIM\_CLK\_EXT)

Address offset:0x17C

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.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Tim_pclk2_sel	Tim_pclk1_sel	Res.	Res.	Res.	Res.	Res.	Res.							
								rw	rw						

Bit	Name	R/W	Reset Value	Function
31:8	Reserved	-	-	-
7	Tim_pclk2_sel	RW	0	选择 APB2 总线下的 Timer 时钟 0: APB2 总线 Timer 使用 2 倍频 APB2 时钟 (APB2 不分频时, 等于 APB2 时钟) 1: APB2 总线 Timer 始终使用 APB2 时钟。
6	Tim_pclk1_sel	RW	0	选择 APB1 总线下的 Timer 时钟 0: APB1 总线 Timer 使用 2 倍频 APB1 时钟 (APB1 不分频时, 等于 APB1 时钟) 1: APB1 总线 Timer 始终使用 APB1 时钟。
7:0	Reserved	-	-	-

## 9.2.17. SYSCFG 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	MEM_MODE	MEM_R	MEM_W
0x00	SYS-CFG_CF_GR1	Res.								I2C_PB12	I2C_PB11	I2C_PB10	I2C_PB9	I2C_PB8	I2C_PB7	I2C_PB6	I2C_PB5	Res.																		
	Read /Write	r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w											r	w	r	w					
	Reset Value	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	SYS-CFG_CF_GR2	Res.																									PVD_LOCK		LOCKUP_LOCK							

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	Read /Write																																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0				
0x08	SYS-CFG_CF_GR3	Reserved																																	
	Read /Write	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w				
	Reset Value	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1				
0x0C	SYS-CFG_CF_GR4	Reserved																																	
	Read /Write	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w				
	Reset Value	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1				
0x10	SYS-CFG_CF_GR5	Reserved																																	
	Read /Write	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w				
	Reset Value	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1				
0x14	SYS-CFG_EX-TICR1																																		
	Read /Write																																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	SYS-CFG_EX-TICR2																																		
	Read /Write																																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	SYS-CFG_EX-TICR3																																		
	Read /Write																																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	Value																																		
0x20	SYS-CFG_EX-TICR4	Res.												EXTI5[3:0]			EXTI4[3:0]			EXTI3[3:0]			EXTI2[3:0]												
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x24	PA_ENS	Res.												PA_ENS																					
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x28	PA_ENS	Res.												PB_ENS																					
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x2C	PA_ENS	Res.												PC_ENS																					
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x30	PA_ENS	Res.												PD_ENS																					
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x34	PA_ENS	Res.												PE_ENS																					
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	GPIO_ENA	Res.												PC_ENA			Res.			PB_ENA			PA_ENA												
	Read /Write													r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x17C	TIM_CLK_EXT	Res.												Res.			Res.			Tim_pclk2sel			Res.												

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																							
	Read /Write																									r	w																													
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														

# 10. DMA 控制器 (DMA)

## 10.1. 概述

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。搬移数据无需 CPU 干预，数据可以通过 DMA 快速地移动，这就节省了 CPU 的资源来做其他操作。两个 DMA 控制器有 12 个通道 (DMA1 有 7 个通道， DMA2 有 5 个通道)，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

### 10.1.1. 主要特性

主要功能如下：

- 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。
  - 请求之间的优先级由软件可编程(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。

## 10.2. 功能描述

### 10.2.1. DMA 传输

DMA 块传输可以从外设请求，或者在存储器到存储器传输的情况下由软件触发。

事件发生后，单个 DMA 传输步骤：

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

当外设是传输的源或目标地时，使用请求/确认协议。例如，在存储器到外设的传输中，外设通过将其单个请求信号驱动到 DMA 控制器来启动传输。DMA 控制器然后读取存储器中的单个数据，并将该数据写入外设。

对于给定的通道 x, DMA 块传输由以下重复序列组成:

- 一个单一的 DMA 传输，封装两个 AHB 传输的单一数据，通过 DMA AHB master 总线:

- 从外设数据寄存器或存储器中的某个位置读取的单个数据(字节、半字或字)，通过内部当前外设/存储器地址寄存器寻址。  
用于第一次单次传输的起始地址是外设或存储器的基址，在 DMA\_CPARx 或 DMA\_CMARx 寄存器中配置。
- 单个数据写入(字节、半字或字)到外设数据寄存器或存储器中的某个位置，通过内部当前外设/存储器地址寄存器进行寻址。  
用于第一次传输的起始地址是外设或存储器的基址，并在 DMA\_CPARx 或 DMA\_CMARx 寄存器中配置。
- DMA\_CNDTRx 寄存器包含剩余要传输的数据项的数量(AHB '读后写' 传输的数量)。

重复以上序列，直到 DMA\_CNDTRx 为空。

注意：AHB 主总线源/目标地址必须与传输到源/目标的单个数据的程序大小对齐。

### 10.2.2. 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。

当一个请求通道 x 被仲裁器授予(硬件请求或软件触发)访问权限时，就会发出单个 DMA 传输(例如单个数据的 AHB “读后写” 传输)。然后，仲裁器再次仲裁所有请求通道，并选择优先级最高的通道。

优先权管理分 2 个阶段：

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

### 10.2.3. 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进行 DMA 传输。DMA 传输数据量是可编程的，最大到 65535 字节，该寄存器值在每次数据传输后递减。

#### 10.2.3.1. 传输数据量可编程

外设和存储器一笔传输数据宽度可以通过 DMA\_CCRx 寄存器中的 PSIZE 和 MSIZE 位编程。

#### 10.2.3.2. 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址。

### 10.2.3.3. 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 寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
- 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 ENABLE 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。

### 10.2.3.4. 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：

- 暂停和恢复通道

这对应以下两个动作：

- 激活的通道被软件禁用(写入 DMA\_CCRx.EN = 0)。
- 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。

- 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。

- 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：

- 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。

- 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。

#### 10.2.3.5. 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 的扫描模式)。在 DMA\_CCRx 寄存器中的 CIRC 位用于开启这一功能。当启动了循环模式,数据传输的数目变为 0 时,将会自动地被恢复成配置通道时设置的初值,DMA 操作将会继续进行。

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。

#### 10.2.3.6. 存储器到存储器模式

DMA 通道的操作可以在没有外设请求的情况下进行,这种操作就是存储器到存储器模式。当设置了 DMA\_CCRx 寄存器中的 MEM2MEM 位之后,在软件设置了 DMA\_CCRx 寄存器中的 EN 位启动 DMA 通道时,DMA 传输将马上开始。当 DMA\_CNDTRx 寄存器变为 0 时,DMA 传输结束。

存储器到存储器模式不能与循环模式同时使用。

#### 10.2.3.7. 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。

#### 10.2.3.8. 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是什么,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。

### 10.2.4. 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 10-1 数据对齐方式

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 0x1/B1 0x2/B2 0x3/B3	1:在 0x0 读 B0[7:0], 在 0x0 写 B0[7:0] 2:在 0x1 读 B1[7:0], 在 0x1 写 B1[7:0] 3:在 0x2 读 B2[7:0], 在 0x2 写 B2[7:0] 4:在 0x3 读 B3[7:0], 在 0x3 写 B3[7:0]	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1:在 0x0 读 B0[7:0], 在 0x0 写 00B0[7:0] 2:在 0x1 读 B1[7:0], 在 0x2 写 00B1[7:0] 3:在 0x2 读 B2[7:0], 在 0x4 写 00B2[7:0] 4:在 0x3 读 B3[7:0], 在 0x6 写 00B3[7:0]	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1:在 0x0 读 B0[7:0], 在 0x0 写 000000B0[31:0] 2:在 0x1 读 B1[7:0], 在 0x4 写 000000B1[31:0] 3:在 0x2 读 B2[7:0], 在 0x8 写 000000B2[31:0] 4:在 0x3 读 B3[7:0], 在 0xC 写 000000B3[31:0]	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:在 0x0 读 B1B0[15:0], 在 0x0 写 B0[7:0] 2:在 0x2 读 B3B2[15:0], 在 0x1 写 B2[7:0] 3:在 0x4 读 B5B4[15:0], 在 0x2 写 B4[7:0] 4:在 0x6 读 B7B6[15:0], 在 0x3 写 B6[7:0]	0x0/B0 0x1/B2 0x2/B4 0x3/B6
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:在 0x0 读 B1B0[15:0], 在 0x0 写 B1B0[15:0] 2:在 0x2 读 B3B2[15:0], 在 0x2 写 B3B2[15:0] 3:在 0x4 读 B5B4[15:0], 在 0x4 写 B5B4[15:0] 4:在 0x6 读 B7B6[15:0], 在 0x6 写 B7B6[15:0]	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:在 0x0 读 B1B0[7:0], 在 0x0 写 0000B1B0[31:0] 2:在 0x2 读 B3B2[7:0], 在 0x4 写 0000B3B2[31:0] 3:在 0x4 读 B5B4[7:0], 在 0x8 写 0000B5B4[31:0] 4:在 0x6 读 B7B6[7:0], 在 0xC 写 0000B7B6[31:0]	0x0/0000B1B0 0x4/000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:在 0x0 读 B3B2B1B0 [31:0], 在 0x0 写 B0[7:0] 2:在 0x4 读 B7B6B5B4 [31:0], 在 0x1 写 B4[7:0] 3:在 0x8 读 BBBAB9B8 [31:0], 在 0x2 写 B8[7:0] 4:在 0xc 读 BFBEBDBC [31:0], 在 0x3 写 BC[7:0]	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:在 0x0 读 B3B2B1B0 [31:0], 在 0x0 写 B1B0[7:0] 2:在 0x4 读 B7B6B5B4 [31:0], 在 0x2 写 B5B4[7:0] 3:在 0x8 读 BBBAB9B8 [31:0], 在 0x4 写 B9B8[7:0] 4:在 0xc 读 BFBEBDBC [31:0], 在 0x6 写 BDDBC[7:0]	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:在 0x0 读 B3B2B1B0 [31:0], 在 0x0 写 B3B2B1B0[7:0] 2:在 0x4 读 B7B6B5B4 [31:0], 在 0x2 写 B7B6B5B4[7:0] 3:在 0x8 读 BBBAB9B8 [31:0], 在 0x4 写 BBBAB9B8[7:0] 4:在 0xc 读 BFBEBDBC [31:0], 在 0x6 写 BFBEBDBC[7:0]	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

#### 10.2.4.1. 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 不适于该模块), 不会发生错误, DMA 将按照下面两个例子写入 32 位 HWDATA 数据:

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

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

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

### 10.2.5. 错误处理

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

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)。

当软件通过一个涉及外设的通道收到一个传输错误的通知时，软件首先在 DMA 模式下停止这个外设，以便禁用任何等待或将来的 DMA 请求。然后，软件通常会在 DMA 模式下重新配置 DMA 和外设，以进行新的传输。

### 10.2.6. 中断

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

表 10-2 DMA 中断

中断事件	事件标志位	使能控制位
传输过半	HTIFx	HTIE <sub>x</sub>
传输完成	TCIFx	TCIE <sub>x</sub>
传输错误	TEIFx	TEIE <sub>x</sub>
传输过半/传输完成/传输错误	GIFx	-

注：

- 1) 当传输长度 NDT 为 1 时，不会产生传输过半标志位 HTIF，传输完成会产生 TCIF 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。

### 10.2.7. DMA 外设请求映射

DMA 外设请求映射到 DMA1 (7 通道) 或者 DMA2 (5 通道) 的各个通道，由 SYSCFG 的 DMA<sub>x</sub>\_MAP 寄存器控制，每个外设请求通过配置可以映射到 12 个通道中的任意一个。

序号与外设请求的关系如下表所示：

表 10-3 DMA 外设请求映射

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	23	I2C2_RD	46	TIM5_CH1
1	ADC2	24	I2C2_WR	47	TIM5_CH2
2	ADC3	25	TIM1_CH1	48	TIM5_CH3
3	保留	26	TIM1_CH2	49	TIM5_CH4
4	保留	27	TIM1_CH3	50	TIM5_UP
5	SPI1_RD	28	TIM1_CH4	51	TIM5_TRIG
6	SPI1_WR	29	TIM1_COM	52	TIM6
7	SPI2_RD	30	TIM1_TRIG	53	TIM7

请求 MUX 输入序号	源	请求 MUX 输入序号	源	请求 MUX 输入序号	源
8	SPI2_WR	31	TIM1_UP	54	TIM8_CH1
9	SPI3_RD	32	TIM2_CH1	55	TIM8_CH2
10	SPI3_WR	33	TIM2_CH2	56	TIM8_CH3
11	USART1_RD	34	TIM2_CH3	57	TIM8_CH4
12	USART1_WR	35	TIM2_CH4	58	TIM8_COM
13	USART2_RD	36	TIM2_UP	59	TIM8_TRIG
14	USART2_WR	37	TIM3_CH1	60	TIM8_UP
15	USART3_RD	38	TIM3_CH3	61	TIM2_TRIG
16	USART3_WR	39	TIM3_CH4	62	TIM3_CH2
17	USART4_RD	40	TIM3_UP	63	TIM4_CH4
18	USART4_WR	41	TIM3_TRIG	64	TIM4_TRIG
19	USART5_RD	42	TIM4_CH1	65	ESMC_TX
20	USART5_WR	43	TIM4_CH2	66	ESMC_RX
21	I2C1_RD	44	TIM4_CH3	67	SDIO
22	I2C1_WR	45	TIM4_UP	68	USB

## 10.3. 寄存器描述 (0x40020000)

### 10.3.1. DMA 中断状态寄存器 (DMA\_ISR)

Address:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	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	Reserved	-	-	Reserved
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； 1: 通道 7 发生半传输事件；
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； 1: 通道 6 发生半传输事件；
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；

Bit	Name	R/W	Reset Value	Function
				1: 通道 6 发生 TE/HT/TC 事件; 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) ; 1: 通道 5 传输错误 (TE) ;
19	TEIF5	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; 1: 通道 5 发生半传输事件;
18	HTIF5	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) ;
17	TCIF5	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;
16	GIF5	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) ;
15	TEIF4	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; 1: 通道 4 发生半传输事件;
14	HTIF4	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) ;
13	TCIF4	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;
12	GIF4	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) ;
11	TEIF3	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; 1: 通道 3 发生半传输事件;
10	HTIF3	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) ;
9	TCIF3	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;
8	GIF3	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) ;
7	TEIF2	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; 1: 通道 2 发生半传输事件;
6	HTIF2	R	0	通道 2 传输完成标志。
5	TCIF2	R	0	通道 2 全局中断标志。

Bit	Name	R/W	Reset Value	Function
				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; 1: 通道 1 发生半传输事件;
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;

### 10.3.2. DMA 中断标志清零寄存器 (DMA\_IFCR)

Address:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.				CTEIF 7	CHTIF 7	CTCIF 7	CGIF 7	CTEIF 6	CHTIF 6	CTCIF 6	CGIF 6	CTEIF 5	CHTIF 5	CTCIF 5	CGIF 5
				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
CTEIF 4	CHTIF 4	CTCIF 4	CGIF 4	CTEIF 3	CHTIF 3	CTCIF 3	CGIF 3	CTEIF 2	CHTIF 2	CTCIF 2	CGIF 2	CTEIF 1	CHTIF 1	CTCIF 1	CGIF 1
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	Reserved	-	-	Reserved
27	CTEIF7	W	0	通道 7 传输错误标志清零。 0: No effect; 1: 清零 TEIF7;
26	CHTIF7	W	0	通道 7 半传输标志清零。 0: No effect; 1: 清零 HTIF7;
25	CTCIF7	W	0	通道 7 传输完成标志清零。 0: No effect; 1: 清零 TCIF7;
24	CGIF7	W	0	通道 7 全局中断标志清零。 0: No effect; 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: No effect; 1: 清零 TEIF6;
22	CHTIF6	W	0	通道 6 半传输标志清零。 0: No effect; 1: 清零 HTIF6;
21	CTCIF6	W	0	通道 6 传输完成标志清零。 0: No effect;

Bit	Name	R/W	Reset Value	Function
				1: 清零 TCIF6; 通道 6 全局中断标志清零。
20	CGIF6	W	0	0: No effect; 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: No effect; 1: 清零 TEIF5;
18	CHTIF5	W	0	通道 5 半传输标志清零。 0: No effect; 1: 清零 HTIF5;
17	CTCIF5	W	0	通道 5 传输完成标志清零。 0: No effect; 1: 清零 TCIF5;
16	CGIF5	W	0	通道 5 全局中断标志清零。 0: No effect; 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: No effect; 1: 清零 TEIF4;
14	CHTIF4	W	0	通道 4 半传输标志清零。 0: No effect; 1: 清零 HTIF4;
13	CTCIF4	W	0	通道 4 传输完成标志清零。 0: No effect; 1: 清零 TCIF4;
12	CGIF4	W	0	通道 4 全局中断标志清零。 0: No effect; 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: No effect; 1: 清零 TEIF3;
10	CHTIF3	W	0	通道 3 半传输标志清零。 0: No effect; 1: 清零 HTIF3;
9	CTCIF3	W	0	通道 3 传输完成标志清零。 0: No effect; 1: 清零 TCIF3;
8	CGIF3	W	0	通道 3 全局中断标志清零。 0: No effect; 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: No effect; 1: 清零 TEIF2;
6	CHTIF2	W	0	通道 2 半传输标志清零。 0: No effect; 1: 清零 HTIF2;
5	CTCIF2	W	0	通道 2 传输完成标志清零。 0: No effect; 1: 清零 TCIF2;
4	CGIF2	W	0	通道 2 全局中断标志清零。 0: No effect; 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: No effect; 1: 清零 TEIF1;

Bit	Name	R/W	Reset Value	Function
2	CHTIF1	W	0	通道1半传输标志清零。 0: No effect; 1: 清零 HTIF1;
1	CTCIF1	W	0	通道1传输完成标志清零。 0: No effect; 1: 清零 TCIF1;
0	CGIF1	W	0	通道1全局中断标志清零。 0: No effect; 1: 清零通道1的GIF/TEIF/HTIF/TCIF;

### 10.3.3. DMA 通道1配置寄存器 (DMA\_CCR1)

Address:0x08

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
				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;

#### 10.3.4. DMA 通道 1 数据传输个数寄存器 (DMA\_CNDTR1)

Address: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	Reserved	-	-	Reserved
15: 0	NDT[15:0]	RW	0	通道 1 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_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。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。

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

Address: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	通道 1 外设地址。 通道 1 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。

#### 10.3.6. DMA 通道 1 存储器地址寄存器 (DMA\_CMAR1)

Address: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。

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

Address: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	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	通道 2 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 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。

Bit	Name	R/W	Reset Value	Function
				0: 禁止; 1: TC 中断使能;
0	EN	RW	0	通道 2 使能。 0: 禁止; 1: 通道 1 使能;

### 10.3.8. DMA 通道 2 数据传输个数寄存器 (DMA\_CNDTR2)

Address: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
NDT[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	NDT[15:0]	RW	0	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。

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

Address:0x24

Reset value:0x0000 0000

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

### 10.3.10. DMA 通道 2 存储器地址寄存器 (DMA\_CMAR2)

Address:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
				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。

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

Address: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	Reserved	-	-	Reserved
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。 0: 禁止; 1: TC 中断使能;

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

### 10.3.12. DMA 通道 3 数据传输个数寄存器 (DMA\_CNDTR3)

Address: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															
NDT[15:0]															
RW															

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

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

Address: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。

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

Address: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。

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

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

Address: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	Reserved	-	-	Reserved
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; 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;

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

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

Address:0x48

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
NDT[15:0]															
															RW

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
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_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。

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

Address:0x4C

Reset value:0x0000 0000

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

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

Address: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。

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

Address:0x58

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.	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	Reserved	-	-	Reserved
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; 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;

### 10.3.20. DMA 通道 5 数据传输个数寄存器 (DMA\_CNDTR5)

Address:0x5C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
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_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。

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

Address:0x60

Reset value:0x0000 0000

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

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

Address:0x64

Reset value:0x0000 0000

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

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

Address: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	Reserved	-	-	Reserved
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; 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;

#### 10.3.24. DMA 通道 6 数据传输个数寄存器 (DMA\_CNDTR6)

Address:0x70

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
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_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。

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

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

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

Address: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。

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

Address:0x80

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.	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	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
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; 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;

### 10.3.28. DMA 通道 7 数据传输个数寄存器 (DMA\_CNDTR7)

Address:0x84

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
NDT[15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	NDT[15:0]	RW	0	通道数据传输数量。

Bit	Name	R/W	Reset Value	Function
				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。

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

Address: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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[15:0]															
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。

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

Address: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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
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。

### 10.3.31. DMA 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		TEIF	HTIF	TCIF	GIIF	TEIF	HTIF	TCIF	GIIF	TEIF	HTIF	TCIF	GIIF	TEIF	HTIF	TCIF	GIIF
0x4002 0000	DMA_ISR	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Reserved															
			MEM2ME	CHTIF	CTCIF	CGIF5	CTEIF	HTIF	TEIF								
			PL[1:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			MSIZE[1:0]	CHTIF	CTCIF	CGIF3	CTEIF	HTIF	TEIF								
			PSIZE[1:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				MINC	0	0	0	0	0	0	0	0	0	0	0	0	0
				PINC	0	0	0	0	0	0	0	0	0	0	0	0	0
				CIRC	0	0	0	0	0	0	0	0	0	0	0	0	0
				DIR	0	0	0	0	0	0	0	0	0	0	0	0	0
				TEIE	0	0	0	0	0	0	0	0	0	0	0	0	0
				HTIE	0	0	0	0	0	0	0	0	0	0	0	0	0
				TCIE	0	0	0	0	0	0	0	0	0	0	0	0	0
				EN	0	0	0	0	0	0	0	0	0	0	0	0	0



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x064	DMA_CMAR5																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x068																																					
0x06C	DMA_CCR6																																				
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6																																				
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x078	DMA_CMAR6																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x07C																																					
0x080	DMA_CCR7																																				
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_CNDTR7																																				
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x08C	DMA_CMAR7																																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

# 11. 中断和事件

## 11.1. 简介

嵌套向量中断控制器(NVIC)和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理晚到的中断。嵌套向量中断控制器管理着包括内核异常等中断。更多关于异常和 NVIC 编程的说明请参考 Cortex-M4 编程手册。

### 11.1.1. 主要特征

- 60 个可屏蔽中断通道(不包含 16 个 Cortex™-M4 的中断线)
- 16 个可编程的优先等级(使用了 4 位中断优先级)
- 低延迟的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

### 11.1.2. 模块框图

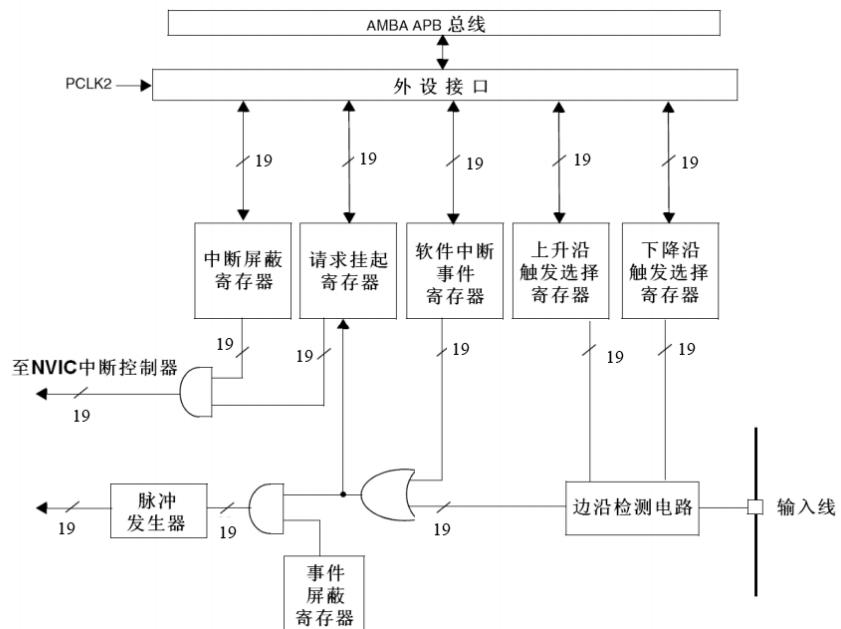


图 11-1 EXTI 模块框图

## 11.2. 功能说明

### 11.2.1. 中断和异常向量表

表 11-1 中断和异常向量表

向量编号	优先级	优先级类型	名称	地址	说明
-	-	-	-	0x0000_0000	保留
-	-3	固定	Reset	0x0000_0004	复位
-	-2	NMI	RCC HSE 时钟安全 (HSE CSS)	0x0000_0008	不可屏蔽中断
-	-1	固定	硬件失效(HardFault)	0x0000_000C	所有类型的失效
-	0	可编程	存储管理(MemManage)	0x0000_0010	存储器管理
	1	可编程	总线错误(BusFault)	0x0000_0014	预取指失败, 存储器访问失败

向量编号	优先级	优先级类型	名称	地址	说明
-	2	可编程	错误应用(UsageFault)	0x0000_0018	未定义的指令或非法状态
-	-	-	-	0x0000_001C	保留
-	-	-	-	0x0000_0020	保留
-	-	-	-	0x0000_0024	保留
-	-	-	-	0x0000_0028	保留
-	3	可编程	SVCall	0x0000_002C	通过 SWI 指令的系统服务调用
-	4	可编程	调试监控(DebugMonitor)	0x0000_0030	调试监控器
-	-	-	-	0x0000_0034	保留
-	5	可编程	PendSV	0x0000_0038	可挂起的系统服务
-	6	可编程	SysTick	0x0000_003C	系统嘀嗒定时器
0	7	可编程	WWDG	0x0000_0040	窗口看门狗定时器中断
1	8	可编程	PVD	0x0000_0044	连到 EXTI 的电源电压检测(PVD)中断
2	9	可编程	TAMPER	0x0000_0048	侵入检测中断
3	10	可编程	RTC	0x0000_004C	实时时钟(RTC)全局中断
4	11	可编程	FMC	0x0000_0050	闪存 FMC 全局中断
5	12	可编程	RCC 和 CTC	0x0000_0054	复位和时钟控制(RCC)和 CTC 中断
6	13	可编程	EXTI0	0x0000_0058	EXTI 线 0 中断
7	14	可编程	EXTI1	0x0000_005C	EXTI 线 1 中断
8	15	可编程	EXTI2	0x0000_0060	EXTI 线 2 中断
9	16	可编程	EXTI3	0x0000_0064	EXTI 线 3 中断
10	17	可编程	EXTI4	0x0000_0068	EXTI 线 4 中断
11	18	可编程	DMA1 通道 1	0x0000_006C	DMA1 通道 1 全局中断
12	19	可编程	DMA1 通道 2	0x0000_0070	DMA1 通道 2 全局中断
13	20	可编程	DMA1 通道 3	0x0000_0074	DMA1 通道 3 全局中断
14	21	可编程	DMA1 通道 4	0x0000_0078	DMA1 通道 4 全局中断
15	22	可编程	DMA1 通道 5	0x0000_007C	DMA1 通道 5 全局中断
16	23	可编程	DMA1 通道 6	0x0000_0080	DMA1 通道 6 全局中断
17	24	可编程	DMA1 通道 7	0x0000_0084	DMA1 通道 7 全局中断
18	25	可编程	ADC1_2	0x0000_0088	ADC1 和 ADC2 全局中断
19	26	可编程	USB	0x0000_008C	USB 中断
20	27	可编程	CAN	0x0000_0090	CAN 中断
21	28	可编程	-	0x0000_0094	保留
22	29	可编程	-	0x0000_0098	保留
23	30	可编程	EXTI9_5	0x0000_009C	EXTI 线[9:5]中断
24	31	可编程	TIM1_BRK_TIM9	0x0000_00A0	TIMER1 中止中断和 TIMER9 全局中断
25	32	可编程	TIM1_UP_TIM10	0x0000_00A4	TIMER1 更新中断和 TIMER10 全局中断
26	33	可编程	TIM1_TRG_COM_TIM11	0x0000_00A8	TIMER1 触发和通信中断及 TIMER11 全局中断
27	34	可编程	TIM1_CC	0x0000_00AC	TIMER1 捕获比较中断
28	35	可编程	TIM2	0x0000_00B0	TIMER2 全局中断
29	36	可编程	TIM3	0x0000_00B4	TIMER3 全局中断
30	37	可编程	TIM4	0x0000_00B8	TIMER4 全局中断
31	38	可编程	I2C1_EV	0x0000_00BC	I2C1 事件中断
32	39	可编程	I2C1_ER	0x0000_00C0	I2C1 错误中断
33	40	可编程	I2C2_EV	0x0000_00C4	I2C2 事件中断
34	41	可编程	I2C2_ER	0x0000_00C8	I2C2 错误中断
35	42	可编程	SPI1	0x0000_00CC	SPI1 全局中断
36	43	可编程	SPI2	0x0000_00D0	SPI2 全局中断
37	44	可编程	USART1	0x0000_00D4	USART1 全局中断
38	45	可编程	USART2	0x0000_00D8	USART2 全局中断

向量编号	优先级	优先级类型	名称	地址	说明
39	46	可编程	USART3	0x0000_00DC	USART3 全局中断
40	47	可编程	EXTI15_10	0x0000_00E0	EXTI 线[15:10]中断
41	48	可编程	RTCAlarm	0x0000_00E4	连到 EXTI 的 RTC 闹钟中断
42	49	可编程	-	0x0000_00E8	保留
43	50	可编程	TIM8_BRK_TIM12	0x0000_00EC	TIMER8 中止中断和 TIMER12 全局中断
44	51	可编程	TIM8_UP_TIM13	0x0000_00F0	TIMER8 更新中断和 TIMER13 全局中断
45	52	可编程	TIM8_TRG_COM_TIM14	0x0000_00F4	TIMER8 触发和通信中断 TIMER14 全局中断
46	53	可编程	TIM8_CC	0x0000_00F8	TIMER8 捕获比较中断
47	54	可编程	ADC3	0x0000_00FC	ADC3 全局中断
48	55	可编程	ESMC	0x0000_0100	ESMC 全局中断
49	56	可编程	SDIO	0x0000_0104	SDIO 全局中断
50	57	可编程	TIM5	0x0000_0108	TIM5 全局中断
51	58	可编程	SPI3	0x0000_010C	SPI3 全局中断
52	59	可编程	USART4	0x0000_0110	USART4 全局中断
53	60	可编程	USART5	0x0000_0114	USART5 全局中断
54	61	可编程	TIM6	0x0000_0118	TIM6 全局中断
55	62	可编程	TIM7	0x0000_011C	TIM7 全局中断
56	63	可编程	DMA2 通道 1	0x0000_0120	DMA2 通道 1 全局中断
57	64	可编程	DMA2 通道 2	0x0000_0124	DMA2 通道 2 全局中断
58	65	可编程	DMA2 通道 3	0x0000_0128	DMA2 通道 3 全局中断
59	66	可编程	DMA2 通道 4_5	0x0000_012C	DMA2 通道 4 和通道 5 全局中断

### 11.2.2. 外部中断/事件控制器 (EXTI)

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

#### 11.2.2.1. 主要特性

EXTI 控制器的主要特性如下：

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持多达 19 个软件的中断/事件请求
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。参见数据手册中电气特性部分的相关参数。

#### 11.2.2.2. 唤醒事件管理

P32F403xx 可以处理外部或内部事件来唤醒内核(WFE)。唤醒事件可以通过下述配置产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时在 Cortex-M4 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位(在 NVIC 中断清除挂起寄存器中)。
- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

使用外部 I/O 端口作为唤醒事件，请参见下述章节。

#### 11.2.2.3. 功能说明

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

置'1'。在挂起寄存器的对应位写'1'，将清除该中断请求。如果需要产生事件，必须先配置好并使能事件线。根据需要的边沿检测通过设置 2 个触发寄存器，同时在事件屏蔽寄存器的相应位写'1'允许事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求脉冲，对应的挂起位不被置'1'。通过在软件中断/事件寄存器写'1'，也可以通过软件产生中断/事件请求。

### 硬件中断选择

通过下面的过程来配置 19 个线路做为中断源：

- 配置 19 个中断线的屏蔽位(EXTI\_IMR)
- 配置所选中断线的触发选择位(EXTI\_RTSR 和 EXTI\_FTSR)；
- 配置对应到外部中断控制器(EXTI)的 NVIC 中断通道的使能和屏蔽位，使得 19 个中断线中的请求可以被正确地响应。

### 硬件事件选择

通过下面的过程，可以配置 19 个线路为事件源

- 配置 19 个事件线的屏蔽位(EXTI\_EMR)
- 配置事件线的触发选择位(EXTI\_RTSR 和 EXTI\_FTSR)

### 软件中断/事件的选择

19 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置 19 个中断/事件线屏蔽位(EXTI\_IMR, EXTI\_EMR)
- 设置软件中断寄存器的请求位(EXTI\_SWIER)

#### 11.2.2.4. 外部中断/事件线路映像

80 个通用 I/O 端口以下图的方式连接到 16 个外部中断/事件线上：

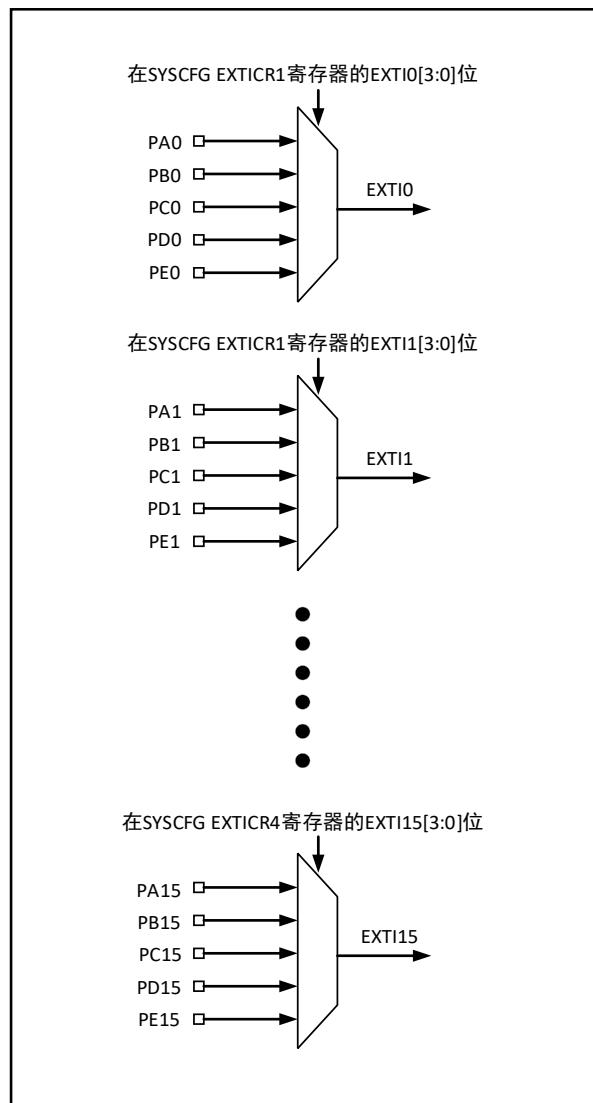


图 11-2 外部中断/事件线路映像

通过 SYSCFG\_EXTICRx 配置 GPIO 线上的外部中断/事件，必须先使能 SYSCFG 时钟。

另外 2 个 EXTI 线的连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件

### 11.3. 寄存器描述

必须以 32 位的方式对这些寄存器进行访问。

#### 11.3.1. 中断屏蔽寄存器 (EXTI\_IMR)

Address offset:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留														IMR18	IMR17	IMR16
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	
IMR15	IMR14	IMR13	IMR12	IMR11	IMR10	IMR9	IMR8	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	IMRx	RW	0	IMRx: 线 x 上的中断屏蔽 (Interrupt Mask on line x) 0: 屏蔽来自线 x 上的中断请求; 1: 开放来自线 x 上的中断请求

### 11.3.2. 事件屏蔽寄存器 (EXTI\_EMR)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
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
EMR1 5	EMR1 4	EMR1 3	EMR1 2	EMR1 1	EMR1 0	EMR 9	EMR 8	EMR 7	EMR 6	EMR 5	EMR 4	EMR 3	EMR2	EMR1	EMR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	EMRx	RW	0	EMRx: 线 x 上的事件屏蔽 (Event Mask on line x) 0: 屏蔽来自线 x 上的事件请求; 1: 开放来自线 x 上的事件请求。

### 11.3.3. 上升沿触发选择寄存器 (EXTI\_RTST)

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
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
RTSR 15	RTSR 14	RTSR 13	RTSR 12	RTSR 11	RTSR 10	RTS R9	RTS R8	RTS R7	RTS R6	RTS R5	RTS R4	RTS R3	RTSR 2	RTSR 1	RTSR 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	RTSRx	RW	0	RTSRx: 线 x 上的上升沿触发事件配置位 (Rising trigger event configuration bit of line x) 0: 禁止输入线 x 上的上升沿触发(中断和事件) 1: 允许输入线 x 上的上升沿触发(中断和事件)

### 11.3.4. 下降沿触发选择寄存器 (EXTI\_FTSR)

Address offset:0x0C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	FTSRx	RW	0	RTSRx: 线 x 上的下降沿触发事件配置位 (Falling trigger event configuration bit of line x) 0: 禁止输入线 x 上的下降沿触发(中断和事件) 1: 允许输入线 x 上的下降沿触发(中断和事件)

### 11.3.5. 软件中断事件寄存器 (EXTI\_SWIER)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留														SWIE R18	SWIE R17	SWIE R16
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	
SWIE R15	SWIE R14	SWIE R13	SWIE R12	SWIE R11	SWIE R10	SWIE R9	SWIE R8	SWIE R7	SWIE R6	SWIE R5	SWIE R4	SWIE R3	SWIE R2	SWIE R1	SWIE R0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	SWIERx	RW	0	SWIERx: 线 x 上的软件中断 (Software interrupt on line x) 当该位为'0'时, 写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 和 EXTI_EMR 中允许产生该中断, 则此时将产生一个中断。 注: 通过清除 EXTI_PR 的对应位(写入'1'), 可以清除该位为'0'。

### 11.3.6. 挂起寄存器 (EXTI\_PR)

Address offset:0x14

Reset value:0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留														PR18	PR17	PR16
														rc_w1	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: 19	Reserved			
18: 0	PRx	RC_W1	x	PRx: 挂起位 (Pending bit) 0: 没有发生触发请求 1: 发生了选择的触发请求 当在外部中断线上发生了选择的边沿事件, 该位被置'1'。在该位中写入'1'可以清除它, 也可以通过改变边沿检测的极性清除。

### 11.3.7. EXTI 寄存器地址映射

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



## 12. 模拟/数字转换 (ADC)

### 12.1. 简介

12 位 ADC 是一种采用逐次逼近方式的模拟数字转换器。它有 20 个多路复用通道，可以转换来自 16 个外部通道和 3 个内部通道的模拟信号。模拟看门狗允许应用程序来检测输入电压是否超出用户设定的高低阈值。各种通道的 A/D 转换可以配置成单次、连续、扫描或间断转换模式。ADC 转换的结果可以按照左对齐或右对齐的方式存储在 16 位数据寄存器中。

#### 12.1.1. 主要特征

- 高性能:
  - 可配置的12 bits、10 bits、8 bits 和6 bits 分辨率;
  - 最大 ADC 采样率： 1MSPs;
  - 自校准;
  - 可编程采样时间;
  - 数据寄存器可配置数据对齐方式;
  - 支持规则通道数据转换的 DMA 请求。
  - 双 ADC 模式 (带2个或以上的 ADC 器件)
- 模拟输入通道:
  - 16个外部模拟输入通道;
  - 1个内部温度传感通道(VSENSE);
  - 1个内部参考电压输入通道(VREFINT)。
  - 1个电池检测输入通道 (VBAT)
- 启动转换方式:
  - 软件启动
  - 硬件触发
- 转换模式:
  - 单次模式，每次触发转换一次选择的输入通道;
  - 扫描模式，可以扫描一系列通道;
    - 连续模式，连续转换所选择的输入通道;
    - 间断模式，每次触发连续转换子序列通道，多次触发直到转换所有通道
  - 同步模式 (适用于具有两个或多个 ADC 的设备)。
- 模拟看门狗
- 中断的产生:
  - 规则组或注入组转换结束
  - 模拟看门狗事件
- ADC 供电要求: 1.7V 到3.6V，一般电源电压为3.3V。
- ADC 输入范围:  $V_{REF-} \leq V_{IN} \leq V_{REF+}$

## 12.1.2. 模块框图

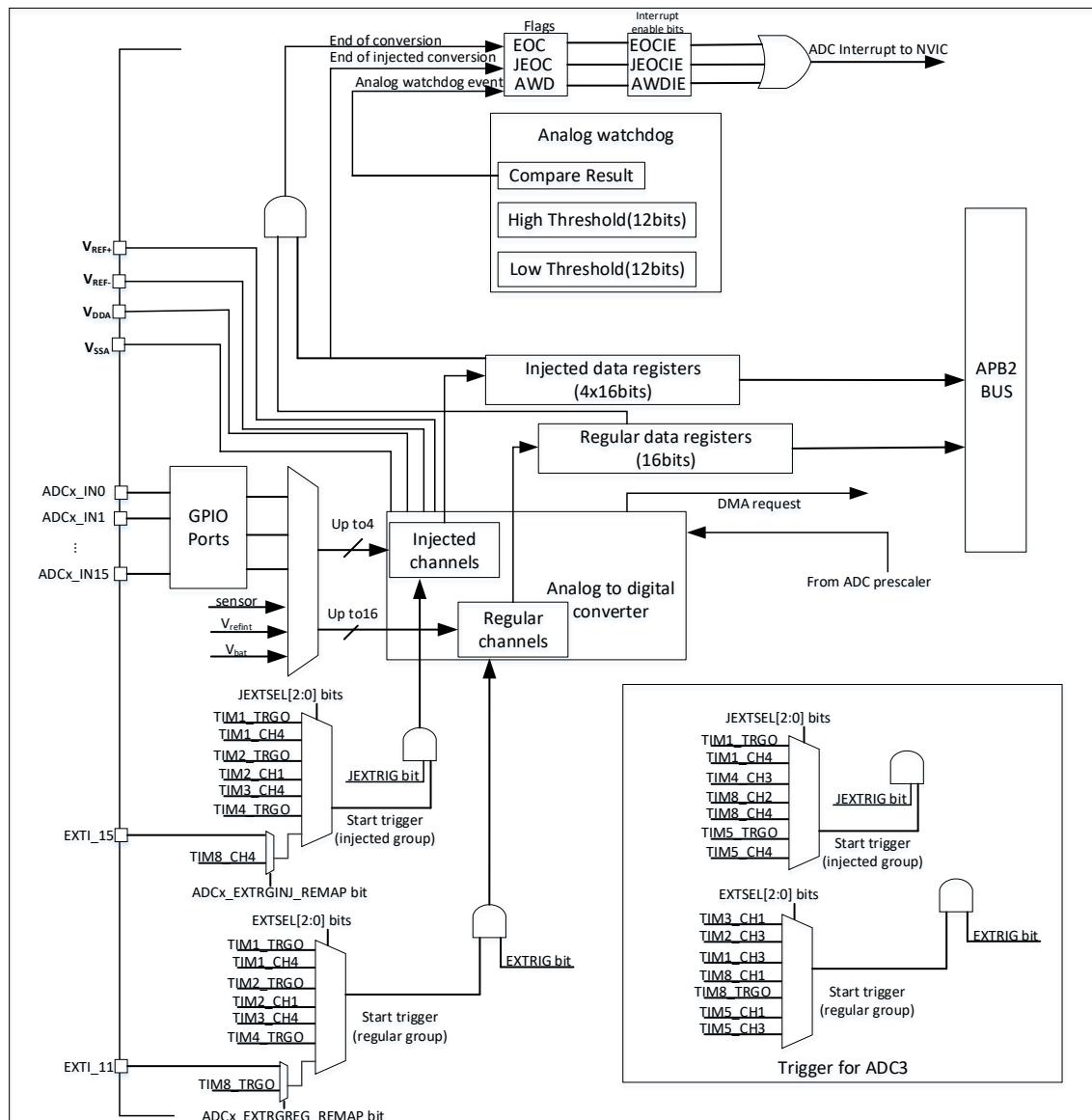


图 12-1 ADC 模块框图

## 12.2. ADC 引脚定义

表 3-2 ADC 引脚定义

名称	信号类型	注释
VDDA	输入, 模拟电源	等效于 VDD 的模拟电源, 全速运行时, $2.0V \leq VDDA \leq VDD$ 低速运行时, $1.7V \leq VDDA \leq VDD$
VSSA	输入, 模拟电源地	等效于 VSS 的模拟电源地
V <sub>REF+</sub>	输入, 模拟参考正极	ADC 使用的正极参考电压, $1.7V \leq V_{REF+} \leq VDDA$
V <sub>REF-</sub>	输入, 模拟参考负极	ADC 使用的负极参考电压, $V_{REF-} = VSSA$
ADCx_IN[15:0]	输入, 模拟信号	模拟输入通道

## 12.3. 功能说明

### 12.3.1. ADC 校准

该 ADC 具有校准功能，用户可通过软件使能进行校准。在校准期间，ADC 计算一个用于 ADC 内部的校准因子（ADC 断电后丢失）。在 ADC 校准期间、未完成校准前，应用不能使用 ADC 模块。在使用 ADC 转换前，建议用户进行校准操作。校准用于消除芯片和芯片之间的，由于工艺变化引起的失调误差和失配误差。校准操作为软件校准。

#### 12.3.1.1. ADC 软件校准

软件设置 CAL=1 可启动校准，校准只能在 ADC 未打开时 (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。校准因子会一直保持，直到产生系统复位。当 ADC 的工作条件发生改变时 (VCC 改变是失调误差和失配误差的主要因素，温度改变次之)，推荐进行再次校准操作。校准的软件操作过程：

- 确认 ADON=0
- 设置 CAL=1
- 等待到 CAL=0

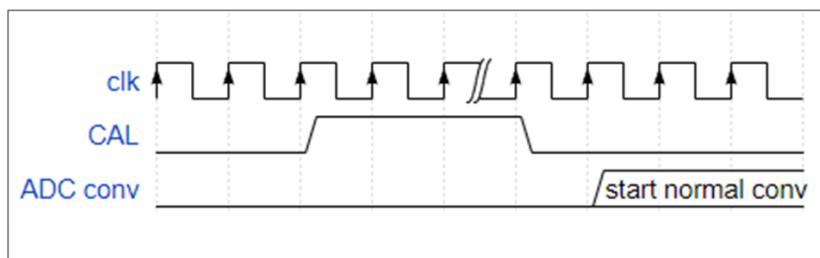


图 12-2 ADC 校准时序图

### 12.3.2. ADC 开-关控制

ADC\_CR1 寄存器中的 ADON 位是 ADC 模块的使能开关。为了省电，当 ADON 为 0 时，ADC 模拟子模块将会进入掉电模式。

注意：

- 1) ADON 位设置为 1 后，需要等待不少于 20us 的延时 (tSTAB) 后进行转换。
- 2) 通过清除 ADON 位可以停止转换，并将 ADC 模拟子模块置于断电模式。ADC 在开始精确转换前需要一个稳定时间 tSTAB，在 ADC 通道组完成转换后，EOC 标志被设置，16 位 ADC 数据寄存器包含转换的结果。

### 12.3.3. ADC 时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2(APB2 时钟)同步。RCC 控制器(CLK 控制器)为 ADC 时钟提供一个专用的可编程预分频器

### 12.3.4. ADC 通道选择

ADC 有 16 个外部通道和 3 个内部通道，其中内部通道为：温度传感器 VSENSE, VREFINT 和 VBAT。16 个外部通道分别连接到 ADC123\_IN0-ADC123\_IN15, 4 个内部通道中，温度传感器 VSENSE 内部连接到通道 ADC1\_IN16, 内部参考电压 VREFINT 连接到 ADC1\_IN17, BAT 电压 VBAT 连接到 ADC123\_IN18。

ADC 通道选择可通过 ADC\_SQRx 或 ADC\_JSQR 配置，配置值和通道对应关系如下表所示：

表 12-1 通道关系表

配置值	通道
00000	ADC 模拟输入通道 0
00001	ADC 模拟输入通道 1
00010	ADC 模拟输入通道 2
...	...

配置值	通道
01111	ADC 模拟输入通道 15
10000	ADC 模拟输入通道 16
10001	ADC 模拟输入通道 17
10010	ADC 模拟输入通道 18
10011	ADC 模拟输入通道 19

ADC 将转换分为两组：规则转换和注入转换。每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：ADC\_IN3、ADC\_IN8、ADC\_IN2、ADC\_IN2、ADC\_IN0、ADC\_IN2、ADC\_IN2、ADC\_IN15。所有通道都可以按注入或规则通道组进行转换。

注意：温度传感器、VREFINT 通道只在 ADC1 外设上可用。

- 一个规则转换组最多由 16 个转换构成。转换序列的规则通道及其转换顺序在 ADC\_SQRx 寄存器中选择。规则转换组中的序列长度必须写入 ADC\_SQR1 寄存器中的 L[3:0] 位。
- 一个注入转换组最多由 4 个转换构成。注入通道及其转换顺序在 ADC\_JSQR 寄存器中选择。注入转换组中的序列长度必须写入 ADC\_JSQR 寄存器中的 L[1:0] 位。

如果在各个通道组转换期间修改 ADC\_SQRx 或 ADC\_JSQR 寄存器，当前转换被清除，并向 ADC 发送一个新的启动脉冲以转换新选择的组。

### 12.3.5. 强制停止 ADC

用软件设置 ADC\_CR1 寄存器中的 ADSTP=1 可以停止当前正在进行的转换，让 ADC 进入空闲状态，为下次转换作好准备。

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)。

一旦结束转换过程，ADSTP 被硬件清 0。

### 12.3.6. 转换模式

(备注：ADCON=1 时，ADC 在执行转换过程中可以去更改 ADC 的转换模式，但是 ADC 转换结果会出现异常，因此不建议在 ADC 转换过程中修改转换模式)

#### 12.3.6.1. 单次转换模式

单次转换模式时，ADC 对单个通道执行一次转换，该模式能够运行在规则组和注入组。ADC\_SQR3 寄存器的 SQ1[4:0]位规定了 ADC 规则组单次转换通道；注入通道单次转换时，应用应保持 ADC\_JSQR[21:20]: JL 为 0，ADC\_JSQR 寄存器的 JSQR4 [4:0]位配置了注入组单次转换通道。

当 ADON 位为 1，CONT、SCAN 和 DISCEN/JDISCEN 皆 0 时，一旦相应外部触发发生，ADC 就会采样和转换一个通道。(注意：单次模式和连续模式不能同时使能，即单次模式时 CONT 为 0。)

- 如果一个规则通道被转换：
  - 转换数据被储存在16位 ADC\_DR 寄存器中
  - 转换结束后，EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入通道被转换：
  - 转换数据被储存在16位的 ADC\_JDRx 寄存器中
  - 转换结束后，JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

### 12.3.6.2. 连续转换模式

连续转换模式时，ADC 对选择的通道进行连续转换，该模式能够运行在规则组。

- 规则通道组：连续转换的通道由 ADC\_SQR3 寄存器的 SQ1[4:0] 位设置。
  - 注入通道组：只执行单次转换，转换的通道由 ADC\_SQR3 寄存器的 SQ1[4:0] 位设置。
- 当 ADON 位为 1，CONT 为 1 时，一旦相应外部触发发生，ADC 采样和转换选择的通道。
- 如果一个规则通道被转换：
    - 转换数据被储存在 16 位的 ADC\_DR 寄存器中
    - 转换结束后，EOC(转换结束)标志被设置
    - 如果设置了 EOCIE，则产生中断。
  - 如果一个注入通道被转换：
    - 转换数据被储存在 16 位的 ADC\_JDR1 寄存器中
    - 转换结束后，JEOC(注入转换结束)标志被设置
    - 如果设置了 JEOCIE 位，则产生中断。

### 12.3.6.3. 扫描模式

扫描转换模式时，ADC 对选择的一个序列所有通道连续转换，该模式能够运行在规则组和注入组。ADC\_SQRx 规定了一个规则序列长度以及所有转换通道，其中 ADC\_SQR1 寄存器的 L[3:0] 位规定了该序列长度。ADC\_JSQR 寄存器规定了一个注入组序列长度以及所有转换通道，其中 ADC\_JSQR 寄存器的 JL[1:0] 位规定了该序列长度。

当 ADON 位为 1，SCAN 为 1 时，一旦相应外部触发发生，ADC 采样和转换一个序列。

- 如果一个规则序列被转换：
  - 转换数据被储存在 16 位的 ADC\_DR 寄存器中
  - 转换结束后，EOC(转换结束)标志被设置
  - 如果设置了 EOCIE，则产生中断。
- 如果一个注入序列被转换：
  - 转换数据被储存在 16 位的 ADC\_JDRx 寄存器中
  - 转换结束后，JEOC(注入转换结束)标志被设置
  - 如果设置了 JEOCIE 位，则产生中断。

### 12.3.6.4. 间断转换模式

间断转换模式时，ADC 将选择的一个序列分成子序列(序列长度 1-8)，通过多次外部触发子序列，完成整个序列转换，该模式能够运行在规则组和注入组。ADC\_CR1 的 DISCNUM 规定了规则通道子序列长度。ADC\_SQRx 规定了一个规则序列所有转换通道，其中 ADC\_SQR1 寄存器的 L[3:0] 位规定了该序列长度。ADC\_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 ADC\_JSQR 寄存器的 JL[1:0] 位规定了该序列长度。

注意：注入通道组的子序列为 1. 该模式下禁止 discen 与 jdiscen 同时使能

- 规则组

一个外部触发信号启动 ADC\_SQRx 描述通道  $n(n \leq 8)$  次转换，直到此序列所有转换完成为止。总的序列长度由 ADC\_SQR1 寄存器的 L[3:0] 定义。

例如：

$n=3$ , 被转换的通道 = 0, 1, 2, 3, 6, 7, 9, 10

第一次触发：转换的序列为 0, 1, 2

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：当一规则组以间断模式转换时，转换序列结束后不自动从头开始。当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0, 1 和 2。

#### ■ 注入组

一个外部触发信号启动 ADC\_JSQR 描述通道 1 次转换，直到序列中所有转换完成为止。总的序列长度由 ADC\_JSQR 寄存器的 JL[1:0] 位定义。

例如：

1)  $n=1$ , 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：

- 1) 当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。
- 2) 不能同时使用自动注入和间断模式。
- 3) 必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

### 12.3.7. 注入通道管理

注入通道外部触发优先级高于规则通道外部触发，即注入通道外部触发可以中断正在进行中的规则通道转换。注入通道有两种中断方式：触发注入和自动注入。

#### 12.3.7.1. 触发注入

ADC 在执行一个规则序列过程中，检测到注入通道外部触发，则中断当前规则序列正在转换的通道，开始执行注入序列转换，当注入序列全部转换结束后，ADC 恢复被中断的规则通道，完成规则序列转换。

ADC\_SQRx 规定了一个规则序列所有转换通道，其中 ADC\_SQR1 寄存器的 L[3:0] 位规定了该序列长度。

ADC\_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 ADC\_JSQR 寄存器的 JL[1:0] 位规定了该序列长度。

注意：触发注入时，JAUTO 必须为 0。若扫描模式时，注入通道转换期间发生规则触发事件，该事件会在注入通道完成后进行转换，原被中断的规则通道丢弃。

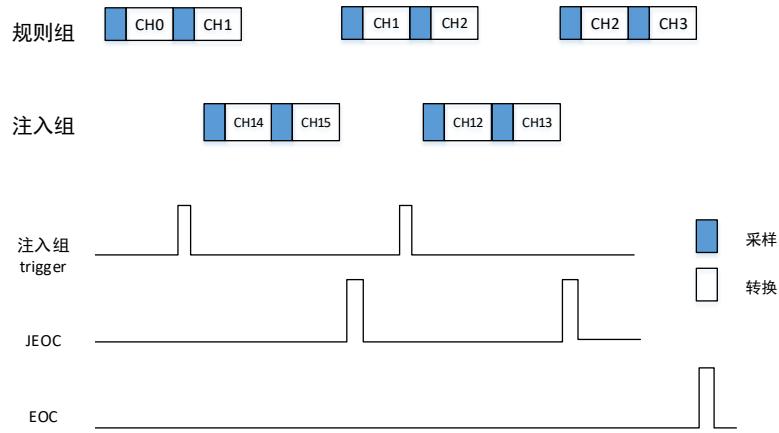


图 12-3 ADC 触发注入(扫描模式)

### 12.3.7.2. 自动注入

自动注入模式时，ADC 完成一个规则序列后，自动执行注入序列转换。ADC\_CR1 的 JAUTO 为自动注入模式使能位。ADC\_SQR1、ADC\_SQR2 和 ADC\_SQR3 规定了一个规则序列所有转换通道，其中 ADC\_SQR1 寄存器的 L[3:0]位规定了该序列长度。ADC\_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 ADC\_JSQR 寄存器的 JL[1:0]位规定了该序列长度。

(注意此模式必须禁止注入通道的外部触发；不能同时使用自动注入和中断模式)

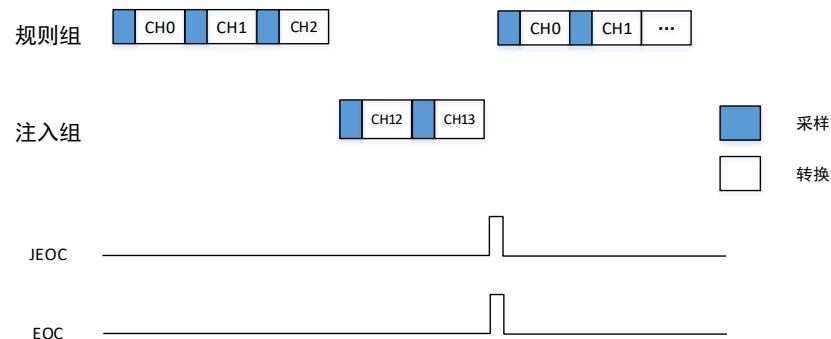


图 12-4 ADC 自动注入(cont+scan+jauto)

### 12.3.8. 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：

表 12-2 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

模拟看门狗保护通道	ADC_CR1 寄存器控制位		
	AWDSGL	AWDEN	JAWDEN
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

注：单一通道由 AWDCH[4:0]位选择

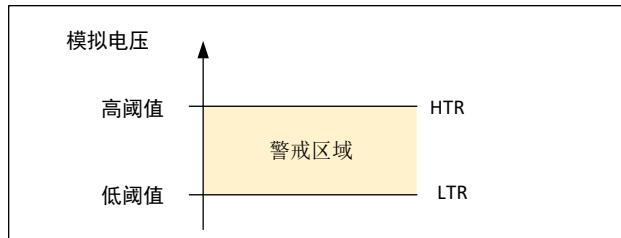


图 12-5 模拟看门狗保护区域

### 12.3.9. 数据对齐

ADC\_CR2 寄存器中的 ALIGN 位用于选择转换后存储的数据的对齐方式，可选择左对齐和右对齐两种方式。注入通道组转换的数据值已经减去了 ADC\_JOFRx 寄存器中自定义的偏移量，因此结果可以是一个负值。SEXT 位表示扩展的符号值。

对于 12bit 规则组中的通道，不会减去任何偏移量，因此只有 12 个位有效。

注入组

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
------	------	------	------	-----	-----	----	----	----	----	----	----	----	----	----	----

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	-----	-----	----	----	----	----	----	----	----	----	----	----

图 12-6 12 位数据右对齐

注入组

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
------	-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---

图 12-7 12 位数据左对齐

### 12.3.10. 可编程采样时间

ADC 会在若干个 ADC CLK 周期内对输入电压进行采样，采样周期数通过 ADC\_SMPR1 和 ADC\_SMPR2 寄存器中的 SMP[2:0] 位配置。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

$$T_{conv} = \text{采样时间} + 12.5 \text{ 个周期}$$

示例：

ADCCLK = 16MHz 且采样时间 =3.5 个周期时：

$T_{conv} = 3.5 + 12.5 = 16$  个周期 = 1  $\mu$ s

### 12.3.11. 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。

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。

表 12-3 ADC1,ADC2 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 CH2 输出		011
定时器 3 的 TRGO 输出		100
定时器 4 的 CH4 输出		101
EXTI 线 11 或定时器 8 的 TRGO 输出(1)	外部管脚或定时器 8 的内部信号	110
SWSTART	软件控制位	111

(1). 通过设置 SYSCFG 模块的 SYSCFG\_CFGR2 寄存器中的 ADC1\_ETRGREG\_REMAP, ADC2\_ETRGREG\_REMAP 选择

表 12-4 ADC1,ADC2 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 4 的 TRGO 输出		101
EXTI 线 15/定时器 8 的 CH4 输出(2)	外部管脚或定时器 8 的内部信号	110
JSWSTART	软件控制位	111

(2). 通过设置 SYSCFG 模块的 SYSCFG\_CFGR2 寄存器中的 ADC1\_ETRGINJ\_REMAP, ADC2\_ETRGINJ\_REMAP 选择

表 12-5 ADC3 规则通道的外部触发

触发源	类型	EXTSEL[2:0]
定时器 3 的 CH1 输出	片上定时器的内部信号	000
定时器 2 的 CH3 输出		001
定时器 1 的 CH3 输出		010
定时器 8 的 CH1 输出		011
定时器 8 的 TRGO 输出		100
定时器 5 的 CH1 输出		101
定时器 5 的 CH3 输出	软件控制位	110
SWSTART	软件控制位	111

表 12-6 ADC3 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

触发源	类型	JEXTSEL[2:0]
定时器 1 的 CH4 输出		001
定时器 4 的 CH3 输出		010
定时器 8 的 CH2 输出		011
定时器 8 的 CH4 输出		100
定时器 5 的 TRGO 输出		101
定时器 5 的 CH4 输出		110
JSWSTART	软件控制位	111

### 12.3.12. 可配置分辨率

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

### 12.3.13. DMA 请求

因为规则通道转换的值储存在一个唯一的数据寄存器中，所以当转换多个规则通道时需要使用 DMA，这可以避免丢失已经存储在 ADC\_DR 寄存器中的数据。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC\_DR 寄存器传输到用户指定的目的地址。双 DMA 模式时，ADC2 转换数据，利用 ADC1 的 DMA 功能传输。

### 12.3.14. 双 ADC 模式

在有 2 个 ADC 的器件中，可以使用双 ADC 模式。在双 ADC 模式里，根据 ADC1\_CR1 寄存器中 DUALMOD[2:0]位所选的模式，转换的启动可以是 ADC1 主和 ADC2 从的交替触发或同时触发。

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

共有 6 种基本转换模式：

- 同时注入模式
- 同时规则模式
- 快速交叉模式
- 慢速交叉模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同时注入模式+同时规则模式
- 同时规则模式+交替触发模式
- 同时注入模式+交叉模式

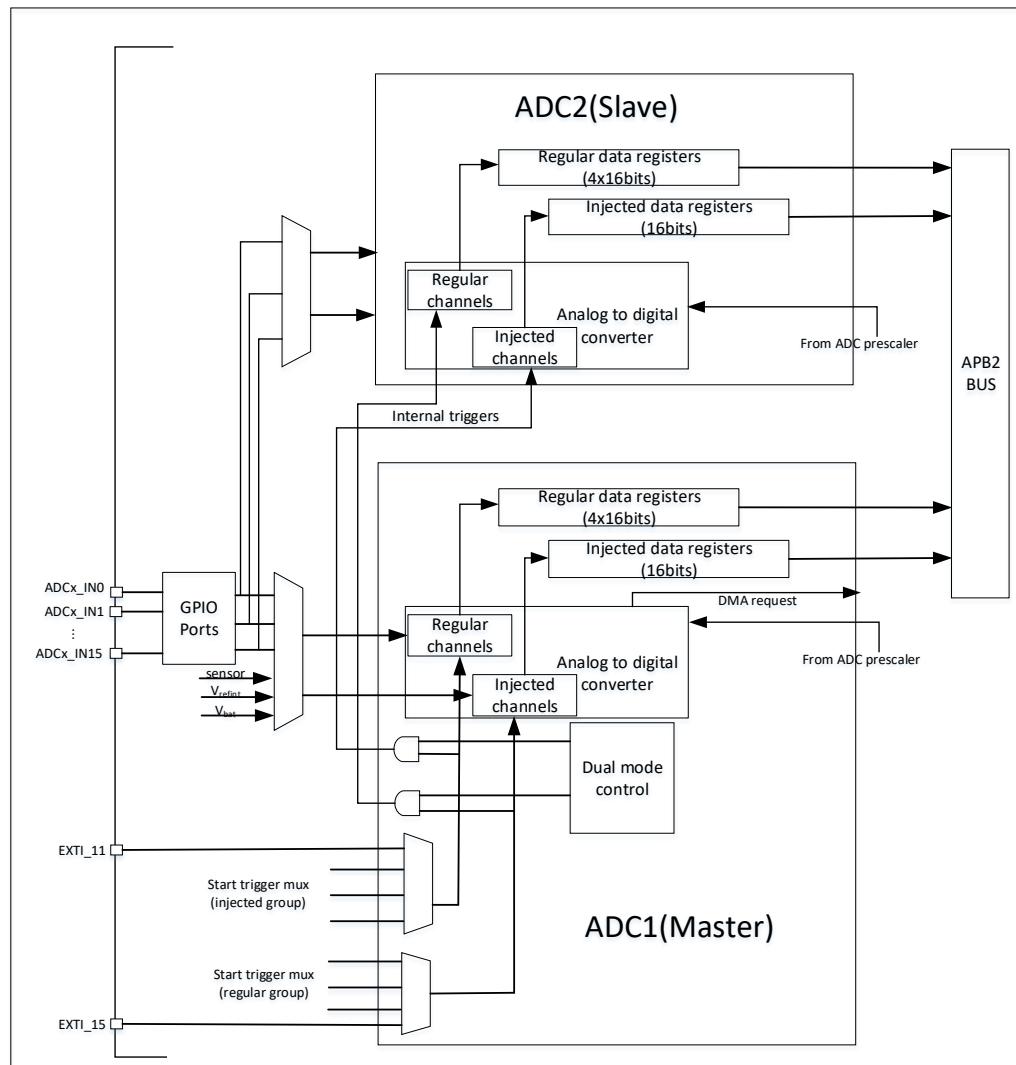


图 12-8 双 ADC 模块图

- 1) 外部触发信号作用于 ADC2，但在本图中没有显示。
- 2) 某些双 ADC 模式中，在完整的 ADC1 数据寄存器(ADC1\_DR)中包含了 ADC1 和 ADC2 的规则转换数据。

#### 12.3.14.1. 同步注入模式

同步注入模式时，ADC1 和 ADC2 同步转换注入通道组。ADC 1 和 ADC 2 的 ADC\_JSQR 寄存器规定了一个注入组序列所有转换通道，其中 ADC\_JSQR 寄存器的 JL[1:0]位规定了该序列序列长度

当 ADON 位为 1，一旦 ADC1 外部触发(由 ADC1\_CR2 寄存器的 JEXTSEL[2:0]选择)发生，ADC1 和 ADC2 同步转换一个注入序列。

注意：

- 1) 不要在 2 个 ADC 上转换相同的通道即两个 ADC 在同一个通道上的采样时间不能重叠
- 2) ADC1 和 ADC2 同步转换的通道采样时间要保持一致
- 3) 在 ADC1 或 ADC2 的转换结束时：
  - i. 转换的数据存储在每个 ADC 的 ADC\_JDRx 寄存器中。
  - ii. 当 ADC1/ADC2 注入通道都被转换时，产生 JEOC 中断(若任一 ADC 使能了中断)。

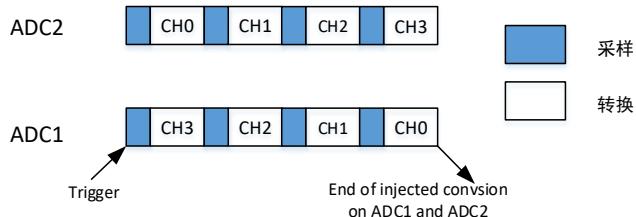


图 12-9 4 个通道上同步注入模式

#### 12.3.14.2. 同步规则模式

同步规则模式时，ADC1 和 ADC2 同步转换规则通道组。ADC 1 和 ADC 2 的 ADC\_SQRx 寄存器规定了一个规则序列所有转换通道，其中 ADC\_SQR1 寄存器的 L[1:0]位规定了该序列长度。

当 ADON 位为 1，一旦 ADC1 外部触发(由 ADC1\_CR2 寄存器的 EXTSEL[2:0]选择)发生，ADC1 和 ADC2 同步就会转换一个规则序列。(注意:用户不要在 2 个 ADC 上转换相同的通道即两个 ADC 在同一个通道上的采样时间不能重叠)

在 ADC1 或 ADC2 的转换结束时：

- 产生一个 32 位 DMA 传输请求(如果设置了 DMA 位)，传输到 SRAM 的 32 位 ADC1\_DR 寄存器的高半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。
- 当 ADC1/ADC2 规则通道都被转换完时，产生 EOC 中断(如果任一 ADC 使能了中断)。

注：

- 1) 在同步规则模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启
- 2) ADC1 和 ADC2 同步转换的通道采样时间要保持一致

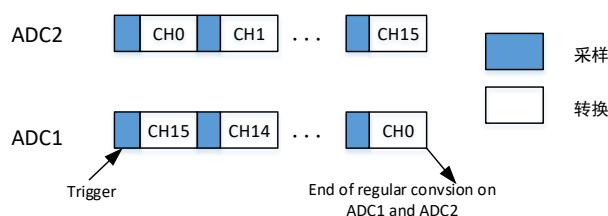


图 12-10 在 16 个通道上的同步规则模式

#### 12.3.14.3. 快速交叉模式

快速交叉模式，ADC1 和 ADC2 交替转换一个通道。此模式只应用于规则通道组（通常一个通道），ADC 1 和 ADC 2 的 ADC\_SQR3 的 SQ1 定义了所转换的通道。

当 ADON 位为 1 时，一旦 ADC1 外部触发(由 ADC1\_CR2 寄存器的 EXTSEL[2:0]选择)发生，ADC1 和 ADC2 就会交替转换一个规则通道。

外部触发源来自 ADC1 的规则通道多路器。外部触发产生后：

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

如果同时设置了 ADC1 和 ADC2 的 CONT 位, 所选的两个 ADC 规则通道将被连续地转换。ADC1 产生一 EOC 中断后(由 EOCIE 使能), 产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位), ADC1\_DR 寄存器的 32 位数据被传输到 SRAM, ADC1\_DR 的高半个字包含 ADC2 的转换数据, 低半个字包含 ADC1 的转换数据。

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

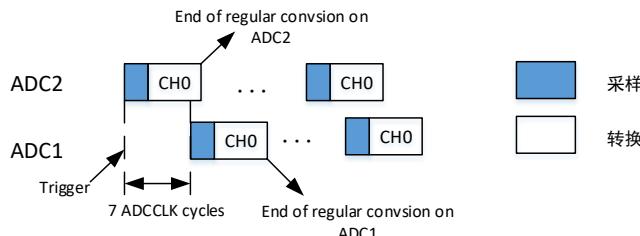


图 12-11 在一个通道上连续转换模式下的快速交叉模式

#### 12.3.14.4. 慢速交叉模式

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

- ADC2 立即启动
- ADC1 在延迟 14 个 ADC 时钟周期后启动
- 在延迟第二个 14 个 ADC 周期后 ADC2 再次启动, 如此循环。

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

ADC1 产生 EOC 中断后(由 EOCIE 使能), 产生一个 32 位的 DMA 传输请求(如果设置了 DMA 位), ADC1\_DR 寄存器的 32 位数据被传输到 SRAM, ADC1\_DR 的高半个字包含 ADC2 的转换数据, 低半个字包含 ADC1 的转换数据。

在这个模式里不能设置 CONT 位, 因为它将连续转换所选择的规则通道。

注意: 应用程序必须确保当使用交叉模式时, 不能有注入通道的外部触发产生。

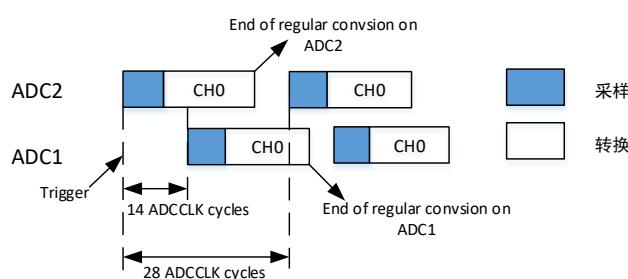


图 12-12 在一个通道上的慢速交叉模式

#### 12.3.14.5. 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路器(SCAN 模式)。

- 当第一个触发产生时, ADC1 上的所有注入组通道被转换。
- 当第二个触发到达时, ADC2 上的所有注入组通道被转换。
- 如此循环.....

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后如果产生另一个外部触发，交替触发处理从转换 ADC1 注入组通道重新开始。

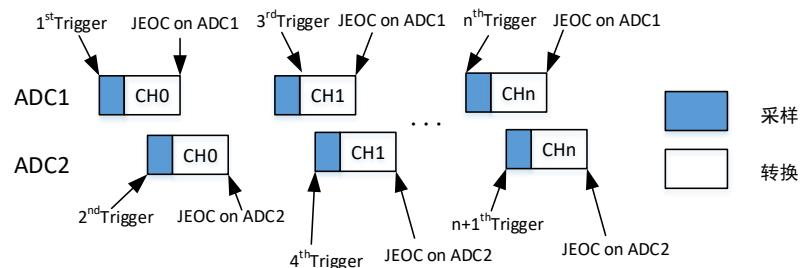


图 12-13 交替触发-每个 ADC 的注入通道组

如果 ADC1 和 ADC2 上同时使用注入间断模式：

- 当第一个触发产生时，ADC1 上的第一个注入通道被转换。
- 当第二个触发到达时，ADC2 上的第一个注入通道被转换。
- 如此循环.....

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。

如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。

当所有注入组通道都转换完后，如果产生另一个外部触发，重新开始交替触发过程。

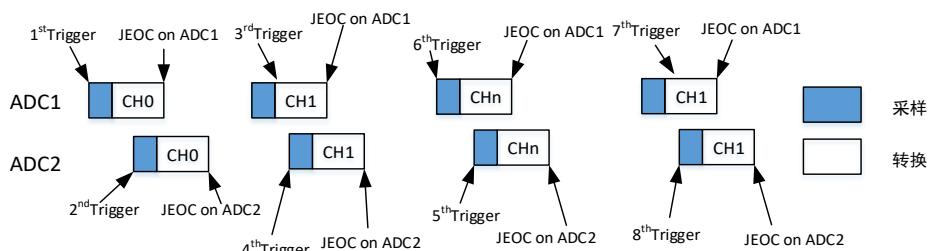


图 12-14 交替触发-在中断模式下每个 ADC 上的 4 个注入通道

#### 12.3.14.6. 独立模式

此模式里，双 ADC 非同步工作，每个 ADC 独立工作。

#### 12.3.14.7. 混合的规则/注入同步模式

规则组同步转换可以被中断，以启动注入组的同步转换。

注：

- 1) 在混合的规则/注入同步模式中，必须转换具有相同时间长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换可能会被重启。
- 2) ADC1 和 ADC2 同步转换的通道采样时间保持一致

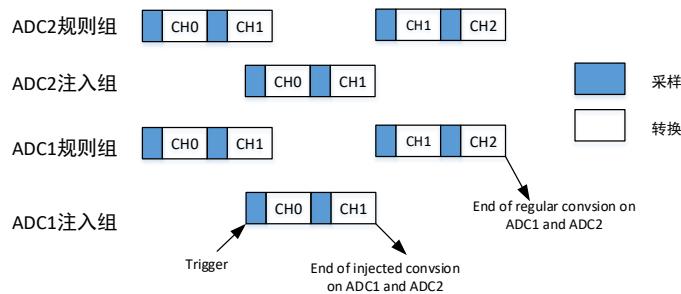


图 12-15 混合的规则/注入同步模式-scan 模式

#### 12.3.14.8. 混合的同步规则加交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。图 4-17 显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换已经在运行，为了在注入转换后确保同步，所有的 ADC(主和从)的规则转换被停止，并在注入转换结束时同步恢复。

注：

- 1) 在混合的同步规则+交替触发模式中，必须转换具有相同长度的序列，或保证触发的间隔比 2 个序列中较长的序列长，否则当较长序列的转换还未完成时，具有较短序列的 ADC 转换会被重启。
- 2) ADC1 和 ADC2 同步转换的规则通道采样时间保持一致

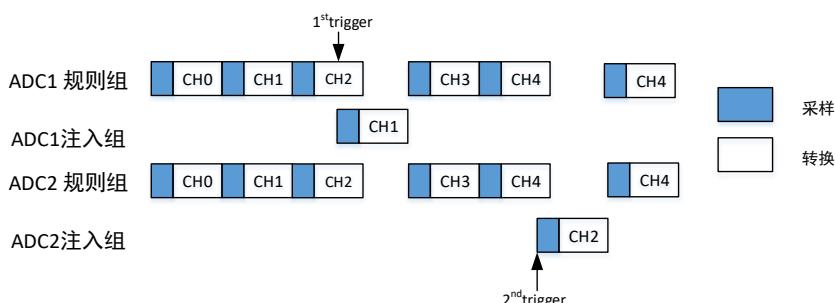


图 12-16 交替加规则同步 (以交替触发+jdiscen 为例)

如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。下图示出了这种情况的操作(第 2 个触发被忽略)

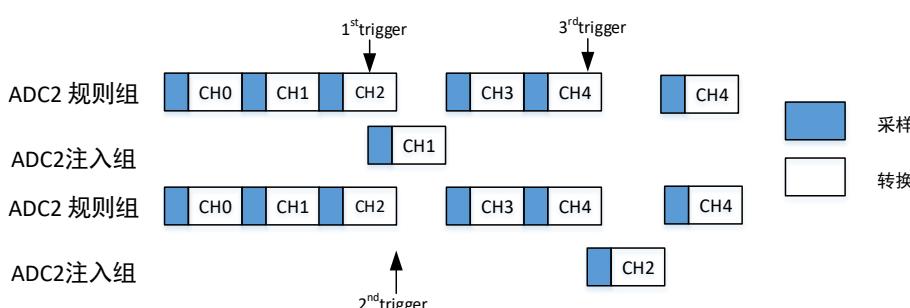


图 12-17 触发事件发生在注入转换期间

### 12.3.14.9. 混合同步注入加交叉模式

一个注入事件可以中断一个交叉转换。这种情况下，交叉转换被中断，注入转换被启动，在注入序列转换结束时，交叉转换被恢复。下图是这种情况的一个例子。

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

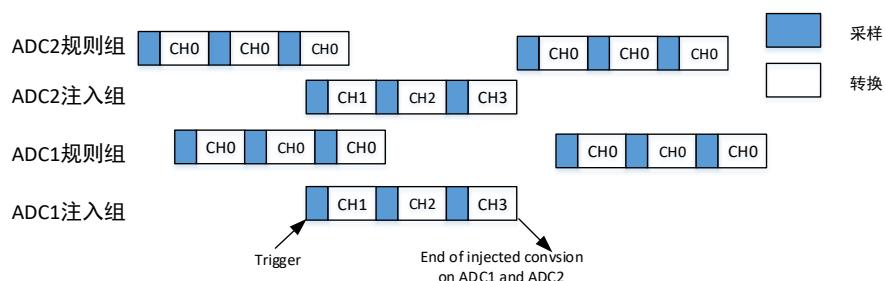


图 12-18 注入序列中断交叉的单通道转换(以同步注入中断 cont+快速交叉)

### 12.3.15. 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 (TA)。温度传感器内部连接到 ADC1\_IN16 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：ADC1\_IN16(温度传感器)和 ADC1\_IN17(VREFINT)的转换。

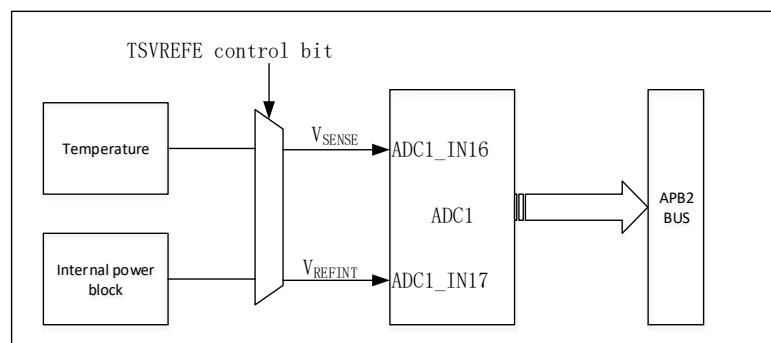


图 12-19 温度传感器通道框图

#### 主要特性

- 支持的温度范围： -40 °C 到 125 °C

- 精度： ±1.5 °C

- 1) 读温度要使用传感器，请执行以下操作：
- 2) 选择 ADC1\_IN16
- 3) 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。
- 4) 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。

- 5) 通过将 ADON 位置1，并使能外部触发，开始 ADC 转换
- 6) 读取 ADC 数据寄存器中生成的 VSENSE 数据
- 7) 使用以下公式计算温度：

Temperature(in °C)=(85°C-25°C)/(TS\_CAL2-TS\_CAL1 )×(TS\_DATA-TS\_CAL1 )+25°C

TS\_CAL2 代表 85°C 温度传感器的校准值，校准值存放地址：0x1FFF 5400

TS\_CAL1 代表 25°C 温度传感器的校准值，校准值存放地址：0x1FFF 5310

TS\_DATA 是 ADC 转换的实际输出值

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，应同时将 ADON 和 TSVREFE 位置 1。

### 12.3.16. 电池监视

由于 VBAT 电压可能高于 VDDA，因此 VBAT 引脚需要内部连接到桥接分配器，以确保 ADC 正确运行。

桥接器自动使能将 VBAT/3 连接到 ADC1/2/3\_IN18 输入通道。

### 12.3.17. ADC 中断

以下任一个事件发生都可以产生中断：它们都有独立的中断使能位。

规则组和注入组转换结束；

■ 模拟看门狗事件；

ADC\_SR 寄存器中存在 2 个其他标志，但是它们没有相关联的中断：

- JSTART (开始转换注入组的通道)
- STRT (开始转换规则组的通道)

注：ADC1 和 ADC2 的中断映射在同一个中断向量上，而 ADC3 的中断有自己的中断向量。

表 12-7 ADC 中断

中断事件	事件标志	使能控制位
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置了模拟看门狗状态位	AWD	AWDIE

## 12.4. ADC 寄存器

ADC1 寄存器基址：0x4001\_2400

ADC2 寄存器基址：0x4001\_2800

ADC3 寄存器基址：0x4001\_3C00

### 12.4.1. 状态寄存器 (0x00: 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
Reserved															
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
Reserved															
										STRT	JSTART	JEOC	EOC	AWD	

RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0
Bit	Name	R/W	Reset Value	Function											
31:5	Reserved	RES	27'h0	Reserved											
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 1，由软件写 0 置 0。 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 该位由硬件在注入通道组转换开始时设置 1，由软件写 0 置 0。 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 1，由软件写 0 置 0。 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 1，由软件写 0 置 0 或当读 ADC_DR 时置 0。 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，由软件写 0 置 0。 0: 没有发生模拟看门狗事件 1: 发生模拟看门狗事件											

### 12.4.2. ADC 控制寄存器 1 (0x04: 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
Reserved			Re-served	ADSTP	Re-served	RESSEL[1:0]		AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
RES.			RW	R_W1	RW	RW		RW	RW	RES.	RES.	RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDIS-CEN	DIS-CEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EO-CIE	AWDCH[4:0]				
RW			RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function											
31:29	Reserved	RES	4'h0	Reserved											
28	Reserved	RES	1'h0	Reserved											
27	ADSTP	R_W1	1'h0	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	Reserved	RES	1'h0	Reserved											
25:24	RESSEL[1:0]	RW	2'b00	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 (10ADCCLK 周期)											
23	AWDEN	RW	1'b0	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	1'b0	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											

Bit	Name	R/W	Reset Value	Function
				1: 在注入通道上使用模拟看门狗
21:20	Reserved	RES	2'b0	Reserved
19:16	DUALMOD[3:0]	RW	4'h0	<p>双模式选择控制位。 ADC2 和 ADC3 中这些位为保留位；（在双 ADC 模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。）</p> <p>0000: 独立模式。 0001: 混合的同步规则+注入同步模式 0010: 混合的同步规则+交替触发模式 0011: 混合同步注入+快速交替模式 0100: 混合同步注入+慢速交替模式 0101: 注入同步模式 0110: 规则同步模式 0111: 快速交替模式 1000: 慢速交替模式 1001: 交替触发模式</p>
15:13	DISCNUM[2:0]	RW	3'h0	<p>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。</p> <p>000: 1 个通道 001: 2 个通道 ..... 111: 8 个通道</p>
12	JDISCEN	RW	1'b0	<p>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。</p> <p>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</p>
11	DISCEN	RW	1'b0	<p>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</p> <p>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</p>
10	JAUTO	RW	1'b0	<p>自动注入使能 该位由软件写 1 置 1 和写 0 置 0，用于开启或关闭规则通道组转换结束后自动进行注入通道组转换</p> <p>0: 关闭自动的注入通道组转换 1: 开启自动的注入通道组转换</p>
9	AWDSGL	RW	1'b0	<p>单一通道看门狗使能。 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 1: 在单一通道上使用模拟看门狗</p>
8	SCAN	RW	1'b0	<p>扫描模式使能 该位由软件写 1 置 1 和写 0 置 0，用于开启或关闭扫描模式。在扫描模式中，由 ADC_SQRx 或 ADC_JSQRx 寄存器选中的通道被转换。</p> <p>0: 关闭扫描模式 1: 使用扫描模式</p>
7	JEOCIE	RW	1'b0	<p>注入通道转换结束中断使能 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 1: 允许 JEOC 中断。</p>
6	AWDIE	RW	1'b0	<p>模拟看门狗中断使能 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 1: 允许模拟看门狗中断</p>
5	EOCIE	RW	1'b0	<p>规则通道转换结束中断使能 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 1: 允许 EOC 中断。</p>

Bit	Name	R/W	Reset Value	Function
4:0	AWDCH[4:0]	RW	5'h0	<p>模拟看门狗通道选择位 该位由软件写设置，用于选择模拟看门狗的输入通道。</p> <p>00000: ADC 模拟输入通道 0 00001: ADC 模拟输入通道 1 ..... 01111: ADC 模拟输入通道 15 10000: ADC 模拟输入通道 16 10001: ADC 模拟输入通道 17 10010: ADC 模拟输入通道 18 10011: ADC 模拟输入通道 19</p> <p>保留所有其他数值。 注： - ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 VREFINT, ADC123 通道 18 连到了 Vbat。 - ADC23 的模拟输入通道 16 和通道 17 在芯片内部连到了 VSS。</p>

### 12.4.3. ADC 控制寄存器 2 (0x08: ADC\_CR2)

Address offset:0x08

Reset value:0x0000\_0000

31	30	29	28	27	2 6	2 5	24	23	22	21	20	19	18	17	16
Reserved						Re-serve d	TSVREF E	SWSTAR T	JSWSTAR T	EX-TTRIG	EXTSEL[2:0]			Re-serve d	
RES.						RW	RW	R_W1	R_W1	RW	RW	RW	RW	RES.	
15	14	13	12	11	1 0	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]		ALIG N	Re-served	DMA	Reserved				RSTCAL	CAL	CON T	ADO N		
RW	R W	R W	R W	RW	RES.	RW	RES.				R_W1	R W 1	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	RES	7'h0	Reserved
24	Reserved	RES	1'b0	Reserved
23	TSVREFE	RW	1'b0	<p>温度传感器和 VREFINT 使能 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 VREFINT 通道。 ADC1 中可使能，ADC2 和 ADC3 中为保留位。 0: 禁止温度传感器和 VREFINT 1: 启用温度传感器和 VREFINT</p>
22	SWSTART	R_W1	1'b0	<p>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</p>
21	JSWSTART	RW	1'b0	<p>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</p>
20	EXTTRIG	RW	1'b0	<p>规则通道外部触发使能 该位由软件写 1 置 1 和写 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</p>
19:17	EXTSEL[2:0]	RW	3'b000	<p>规则通道外部触发事件选择位。选择启动规则通道组转换的外部事件 ADC1,ADC2 外部触发事件如下： 000: 定时器 1 的 CH1 事件</p>

Bit	Name	R/W	Reset Value	Function
				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: 定时器 4 的 CH4 事件 110: 定时器 8 的 TRGO 事件或 EXTI11 (通过 SYSCFG 模块的 ADC1_ETRGREG_REMAP, ADC2_ETRGREG_REMAP 选择) 111: SWSTART ADC3 外部触发配置如下: 000: 定时器 3 的 CH1 事件 001: 定时器 2 的 CH3 事件 010: 定时器 1 的 CH3 事件 011: 定时器 8 的 CH1 事件 100: 定时器 8 的 TRGO 事件 101: 定时器 5 的 CH1 事件 110: 定时器 5 的 CH3 事件 111: SWSTART
16	Reserved	RES	1'b0	Reserved
15	JEXTTRIG	RW	1'b0	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	3'h0	注入通道组外部触发事件选择位 ADC1 和 ADC2 的外部触发事件如下: 000: 定时器 1 的 TRGO 事件 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 CH4 事件 101: 定时器 4 的 TRGO 事件 110: 定时器 8 的 CH4 事件或 EXTI15 (通过 SYSCFG 模块的 ADC1_ETRGINJ_REMAP, ADC2_ETRGINJ_REMAP 选择) 111: JSWSTART ADC3 的触发设置如下: 000: 定时器 1 的 TRGO 事件 001: 定时器 1 的 CH4 事件 010: 定时器 4 的 CH3 事件 011: 定时器 8 的 CH2 事件 100: 定时器 8 的 CH4 事件 101: 定时器 5 的 TRGO 事件 110: 定时器 5 的 CH4 事件 111: JSWSTART
11	ALIGN	RW	1'b0	数据对齐控制位 该位由软件写 1 置 1 和写 0 清除。 0: 右对齐 1: 左对齐
10:9	Reserved	RES	2'h0	Reserved
8	DMA	RW	1'b0	DMA 使能位 该位由软件写 1 置 1 和写 0 清除。 0: 禁止 DMA 模式 1: 使能 DMA 模式 注: 双 adc 模式中, 只有 ADC1 能产生 DMA 请求。
7:4	Reserved	RES	4'h0	Reserved
3	RSTCAL	R_W1	1'b0	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

Bit	Name	R/W	Reset Value	Function
				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	1'b0	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	1'b0	连续转换使能 该位由软件写 1 置 1 和写 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
0	ADON	RW	1'b0	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。

#### 12.4.4. ADC 采样时间寄存器 1 (0x0C: 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
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
RES.								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
31:24	Reserved	RES.	8'h0-	Reserved
23:0	SMPx[2:0]	RW	24'h0	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

#### 12.4.5. ADC 采样时间寄存器 2 (0x10: 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
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
RES.		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														

Bit	Name	R/W	Reset Value	Function
31:30	Reserved	RES	2'h0-	Reserved
29:0	SMPx[2:0]	RW	30'h0	选择通道 x 的采样时间

Bit	Name	R/W	Reset Value	Function
				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

#### 12.4.6. ADC 注入通道数据偏移寄存器 x (0x14~0x20: ADC\_JOFRx) x=1~4

Address offset:0x14~0x20

Reset value:0x0000\_0000

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0	Reserved
11:0	JOFFSETx[11:0]	RW	12'h0	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。

#### 12.4.7. ADC 看门狗高阈值寄存器 (0x24: ADC\_HTR)

Address offset:0x24

Reset value:0x0000\_0FFF

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0	Reserved
11:0	HT[11:0]	RW	12'hFFF	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阀值高限。

#### 12.4.8. ADC 看门狗低阈值寄存器 (0x28: ADC\_LTR)

Address offset:0x28

Reset value:0x0000\_0000

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0-	Reserved

Bit	Name	R/W	Reset Value	Function
11:0	LT[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低限。

#### 12.4.9. ADC 规则序列寄存器 1 (0x2C: ADC\_SQR1)

Address offset:0x2C

Reset value:0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
RES.								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	Reserved	RES	8'h0-	Reserved
23:20	L[3:0]	RW	4'b0000	规则通道序列长度 软件配置这些位的值。这些位定义了在规则通道转换序列中通道数目。 0000: 1个转换 0001: 2个转换 ..... 1111: 16个转换
19:15	SQ16[4:0]	RW	5'b00000	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(0~19)。
14:10	SQ15[4:0]	RW	5'b00000	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(0~19)。
9:5	SQ14[4:0]	RW	5'b00000	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(0~19)。
4:0	SQ13[4:0]	RW	5'b00000	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(0~19)。

#### 12.4.10. ADC 规则序列寄存器 2 (0x30: ADC\_SQR2)

Address offset:0x030

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]			
RES.		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	Reserved	RES	2'h0	Reserved
29:25	SQ12[4:0]	RW	5'b00000	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(0~19)。
24:20	SQ11[4:0]	RW	5'b00000	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(0~19)。
19:15	SQ10[4:0]	RW	5'b00000	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(0~19)。
14:10	SQ9[4:0]	RW	5'b00000	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(0~19)。
9:5	SQ8[4:0]	RW	5'b00000	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(0~19)。

Bit	Name	R/W	Reset Value	Function
4:0	SQ7[4:0]	RW	5'b00000	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(0~19)。

### 12.4.11. ADC 规则序列寄存器 3 (0x34: ADC\_SQR3)

Address offset:0x34

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]		
RES.	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	Reserved	RES	2'h0	Reserved
29:25	SQ6[4:0]	RW	5'b00000	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(0~19)。
24:20	SQ5[4:0]	RW	5'b00000	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(0~19)。
19:15	SQ4[4:0]	RW	5'b00000	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(0~19)。
14:10	SQ3[4:0]	RW	5'b00000	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(0~19)。
9:5	SQ2[4:0]	RW	5'b00000	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(0~19)。
4:0	SQ1[4:0]	RW	5'b00000	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(0~19)。

### 12.4.12. ADC 注入序列寄存器 (0x38: ADC\_JSQR)

Address offset:0x38

Reset value:0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												JL[1:0]	JSQ4[4:1]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.												RW	RW		
JSQ4[0]	JSQ3[4:0]						JSQ2[4:0]						JSQ1[4:0]		
RW	RW						RW						RW		

Bit	Name	R/W	Reset Value	Function
31:22	Reserved	RES	10'h0	Reserved
21:20	JL[1:0]	RW	2'b00	注入通道序列长度 软件配置这些位的值。这些位定义了在注入通道转换序列中 转换数。 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	5'b00000	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(0~19)。 注：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从(4-JL)开始。 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	5'b00000	软件配置这些位的值。注入序列中的第 3 个转换

Bit	Name	R/W	Reset Value	Function
9:5	JSQ2[4:0]	RW	5'b00000	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	5'b00000	软件配置这些位的值。注入序列中的第 1 个转换

#### 12.4.13. ADC 注入数据寄存器 x (0x3C~0x48: ADC\_JDRx) x=1~4

Address offset:0x3C~0x48

Reset value:0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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	Reserved	RES	16'h0	Reserved
15:0	JDATA[15:0]	R	16'h0	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

#### 12.4.14. ADC 规则数据寄存器 (0x4C: ADC\_DR)

Address offset:0x4C

Reset value:0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
R															
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	ADC2DATA[15:0]	R	16'h0	ADC2 转换的数据 软件可读这些位的值。 - 在 ADC1 中：双模式下，这些位包含了 ADC2 转换的规则通道数据。 - 在 ADC2 中：保留位。
15:0	DATA[15:0]	R	16'h0	规则通道转换结果 软件可读这些位的值。这些位为只读，包含了规则通道的转换结果。数据是左或右对齐

#### 12.4.15. ADC 校准配置和状态寄存器 (0x50: ADC\_CCSR)

Address offset:0x50

Reset value:0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON	CAPSUC	OFFSUC	Reserved												
R	RC_W1	RES.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Reserved	CALSMP	CALSEL	Reserved											
RW	RW	RW	RW	RW	RES.										

Bit	Name	R/W	Reset Value	Function
31	CALON	R	1'b0	校准状态位。 1: ADC 正在进行校准； 0: ADC 校准已结束或未启动。
30	CAPSUC	RC_W1	1'b0	电容校准状态位。

Bit	Name	R/W	Reset Value	Function
				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	1'b0	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	Reserved	RES.	13'h0	
15	Reserved	RES	1'h0	Reserved
14	Reserved	RES	1'h0	Reserved
13:12	CALSMMP	RW	2'h0	校准采样时间选择。当 CAL 为 0 时，软件置位选择。 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	1'h0	校准内容选择。当 CAL 为 0 时，软件置位选择需要校准的内容。 1: 校准 OFFSET 以及电容 0: 只校准 OFFSET
10:0	Reserved	RES.	11'h0	

#### 12.4.16. ADC 寄存器映像

Of fs et	Reg- ister	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x 00	ADC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.									
	Re-set value																																				
0x 04	ADC_CR	Res.	Res.	Res.	Res.	Res.	Res.	RES- SEL[1: 0]	DUALMODE[ 3:0]	DISCNU M[2:0]	JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]	STRT	JSTRRT	JEOC	EOC	AWD	0	0	0	0	0	0	0	0	0	0	0	0	
	Re-set value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Control		Control		Control		Control		Control		Control	
	Register	Reset value	Register	Reset value	Register	Reset value	Register	Reset value	Register	Reset value	Register	Reset value
0x1c	ADC_R3 <sub>JOF</sub>	Reset value	ADC_R2 <sub>JOF</sub>	Reset value	ADC_R1 <sub>JOF</sub>	Reset value	ADC_PR2	Reset value	ADC_PR1	Reset value	ADC_CR	0x08
0x1d	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	31
0x1e	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	30
0x1f	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	29
0x20	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP9[2:0]	Res.	Res.	Res.	28
0x21	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP8[2:0]	Res.	Res.	Res.	27
0x22	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP7[2:0]	Res.	Res.	Res.	26
0x23	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP6[2:0]	Res.	Res.	Res.	25
0x24	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP5[2:0]	Res.	Res.	Res.	24
0x25	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP4[2:0]	Res.	Res.	Res.	23
0x26	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP3[2:0]	Res.	Res.	Res.	22
0x27	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP2[2:0]	Res.	Res.	Res.	21
0x28	Res.	Res.	Res.	Res.	Res.	Res.	0	SMP1[2:0]	Res.	Res.	Res.	20
0x29	Res.	Res.	Res.	Res.	Res.	Res.	0	EXTSEL[2:0]	Res.	Res.	Res.	19
0x2a	Res.	Res.	Res.	Res.	Res.	Res.	0	EXTSEL[2:0]	Res.	Res.	Res.	18
0x2b	Res.	Res.	Res.	Res.	Res.	Res.	0	EXTSEL[2:0]	Res.	Res.	Res.	17
0x2c	Res.	Res.	Res.	Res.	Res.	Res.	0	EXTSEL[2:0]	Res.	Res.	Res.	16
0x2d	Res.	Res.	Res.	Res.	Res.	Res.	0	JEXTTRIG	Res.	Res.	Res.	15
0x2e	Res.	Res.	Res.	Res.	Res.	Res.	0	JEXTSEL[2:0]	Res.	Res.	Res.	14
0x2f	Res.	Res.	Res.	Res.	Res.	Res.	0	JEXTSEL[2:0]	Res.	Res.	Res.	13
0x30	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	12
0x31	0	0	0	0	0	0	0	SMP13[2:0]	0	0	0	11
0x32	0	0	0	0	0	0	0	SMP12[2:0]	0	0	0	10
0x33	0	0	0	0	0	0	0	SMP11[2:0]	0	0	0	9
0x34	0	0	0	0	0	0	0	SMP10[2:0]	0	0	0	8
0x35	0	0	0	0	0	0	0	SMP9[2:0]	0	0	0	7
0x36	0	0	0	0	0	0	0	SMP8[2:0]	0	0	0	6
0x37	0	0	0	0	0	0	0	SMP7[2:0]	0	0	0	5
0x38	0	0	0	0	0	0	0	SMP6[2:0]	0	0	0	4
0x39	0	0	0	0	0	0	0	SMP5[2:0]	0	0	0	3
0x3a	0	0	0	0	0	0	0	SMP4[2:0]	0	0	0	2
0x3b	0	0	0	0	0	0	0	SMP3[2:0]	0	0	0	1
0x3c	0	0	0	0	0	0	0	SMP2[2:0]	0	0	0	0
0x3d	0	0	0	0	0	0	0	SMP1[2:0]	0	0	0	0

0x34	ADC R3	ADC R <sub>SQ</sub>	ADC R <sub>SQ</sub>	ADC R <sub>1</sub>	ADC R <sub>1</sub> <sub>SQ</sub>	ADC R <sub>2</sub>	ADC R <sub>2</sub> <sub>SQ</sub>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
0x30	Res.	Res.	Res.	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	0	0	0	0	0	0	0	SQ5[4:0]	SQ11[4:0]	SQ12[4:0]	SQ10[4:0]	SQ9[4:0]	SQ8[4:0]	SQ7[4:0]	SQ6[4:0]	SQ5[4:0]	SQ4[4:0]	SQ3[4:0]	SQ2[4:0]	SQ1[4:0]	SQ0[4:0]	HT[11:0]	JOFFSET4[11:0]	6	5		
0x24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	



Offset	Register	Reset Value
0x50	ADC_SCCR	0
	CALON	31
	CAPSUC	30
	OFFSUC	29
	Res.	28
	Res.	27
	Res.	26
	Res.	25
	Res.	24
	Res.	23
	Res.	22
	Res.	21
	Res.	20
	Res.	19
	Res.	18
	Res.	17
	Res.	16
	Res.	15
	Res.	14
	CALSMPP	13
	0	12
	0	11
	CALSEL	10
	Res.	9
	Res.	8
	Res.	7
	Res.	6
	Res.	5
	Res.	4
	Res.	3
	Res.	2
	Res.	1
	Res.	0

# 13. 高级定时器 (TIM1 和 TIM8)

## 13.1. 简介

高级控制定时器(TIM1 和 TIM8)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。

它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM、嵌入死区时间的互补 PWM)。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

高级控制定时器(TIM1 和 TIM8)和通用定时器(TIMx)是完全独立的，它们不共享任何资源。

### 13.1.1. TIM1 和 TIM8 主要特征

TIM1 和 TIM8 定时器的功能包括：

- 16 位向上、向下、向上/下的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 多达 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 带有可编程死区时间的互补输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
    - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
    - 输入捕获
    - 输出比较
    - 刹车信号输入
  - 支持针对定位的增量(正交)编码器和霍尔传感器电路
  - 触发输入作为外部时钟或者按周期的电流管理

### 13.1.2. 模块框图

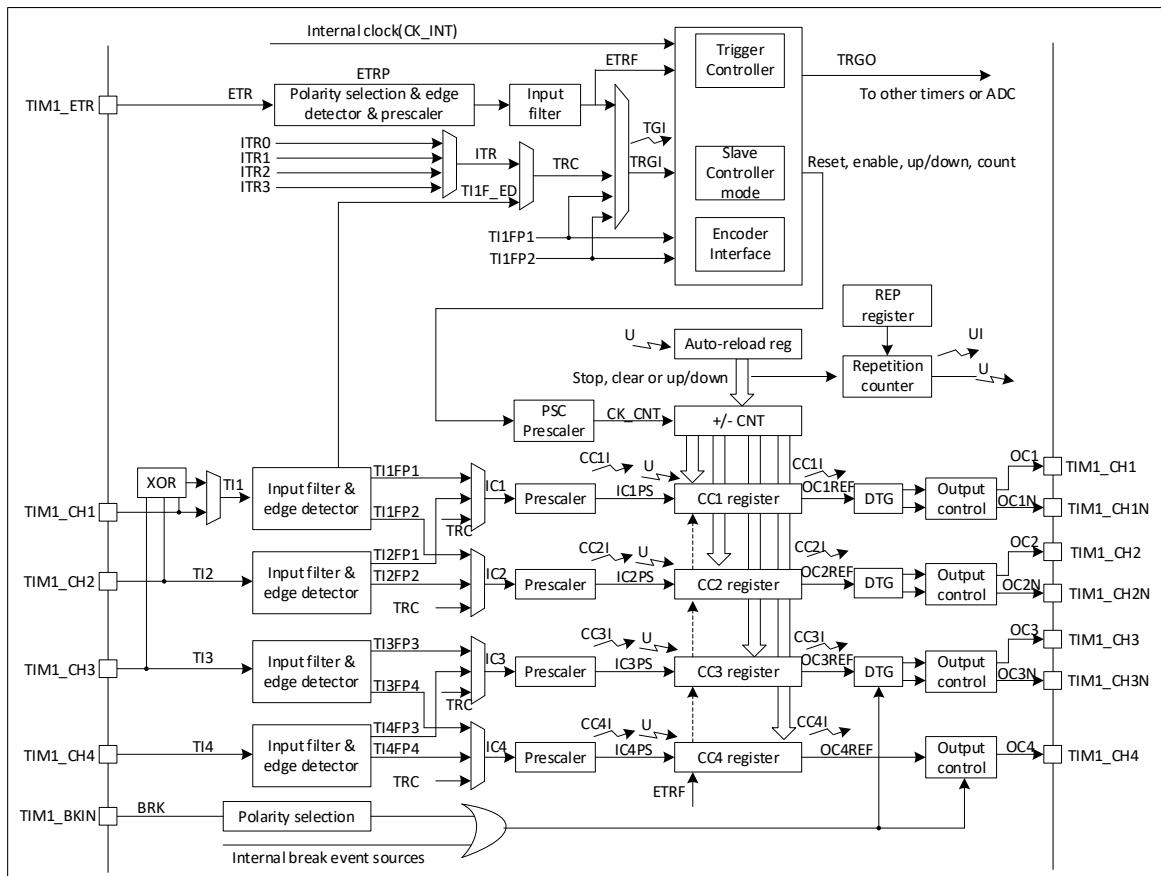


图 13-1 TIM1 和 TIM8 模块

## 13.2. TIM1 和 TIM8 功能描述

### 13.2.1. 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。计数器的时钟可以被预分频器分频。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)
- 自动装载寄存器 (TIMx\_ARR)
- 重复次数寄存器 (TIMx\_RCR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问它的预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 (UEV) 时传送到影子寄存器。当计数器达到溢出条件(上溢或者下溢)并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK\_CNT 驱动，仅当设置了 TIMx\_CR1 寄存器中的计数器使能位(CEN)，CK\_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

### 13.2.1.1. 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx\_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频参数将在下一次更新事件到来时被采用。

图 4-1 和图 4-2 给出了在预分频器运行时，更改计数器参数的例子。

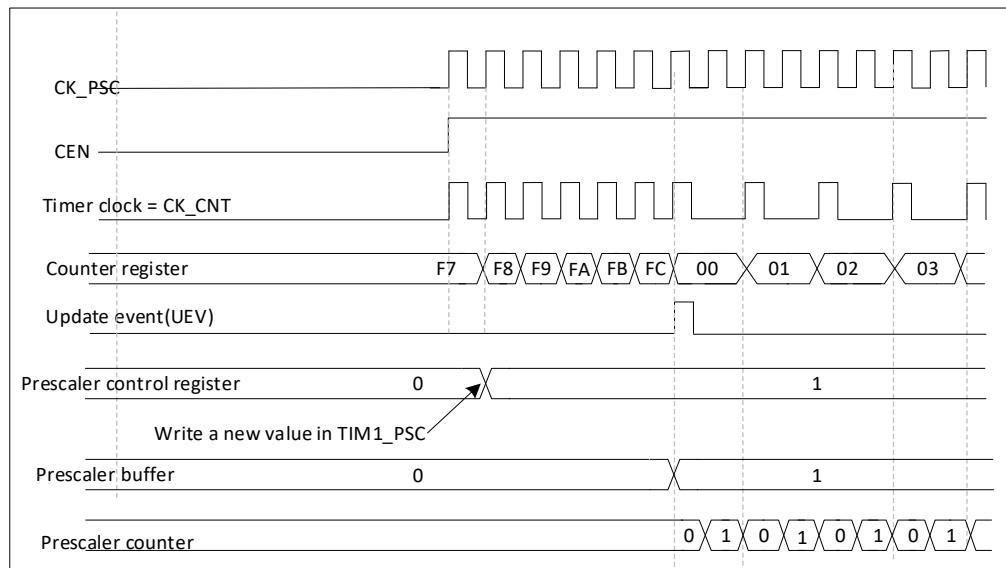


图 13-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

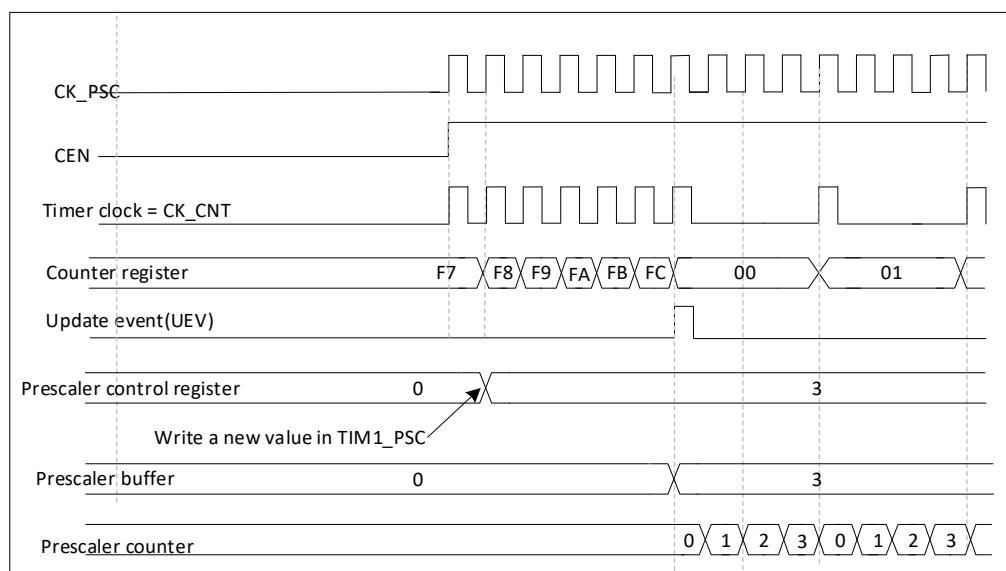


图 13-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

## 13.2.2. 计数器模式

### 13.2.2.1. 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx\_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

如果使用了重复计数器，那么更新事件(UEV)需要在上溢次数达到所配置的重复计数寄存器的值加一 (即 TIMx\_RCR+1) 时才会产生；如果没有使用重复计数器 (即 TIMx\_RCR=0) ，那么每次计数上溢都会产生更新事件。

而在 `TIMx_EGR` 寄存器中(通过软件方式或者使用从模式控制器)设置 `UG` 位也同样可以产生一个更新事件。

通过设置 `TIMx_CR1` 寄存器中的 `UDIS` 位, 可以禁止更新事件; 这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 `UDIS` 位被清‘0’之前, 将不会产生更新事件。即使这样, 在应该产生更新事件时, 计数器仍会被清‘0’, 同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外, 如果设置了 `TIMx_CR1` 寄存器中的 `URS` 位(选择更新请求源), 通过设置 `UG` 位可以产生一个更新事件 `UEV`, 但不会置起 `UIF` 标志位(即不会产生中断或 DMA 请求)。这是为了避免在捕获事件时清除计数器, 同时产生更新和捕获中断。

当发生一个更新事件时, 所有以下的寄存器都被更新, 硬件同时(依据 `URS` 位)设置更新标志位(`TIMx_SR` 寄存器中的 `UIF` 位):

- 重复计数器被重新加载为 `TIMx_RCR` 寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值(`TIMx_ARR`)。
- 预分频器的缓冲区被置入预装载寄存器的值(`TIMx_PSC` 寄存器的内容)。

下图给出一些例子, 当 `TIMx_ARR=0x36` 时计数器在不同时钟频率下的动作。

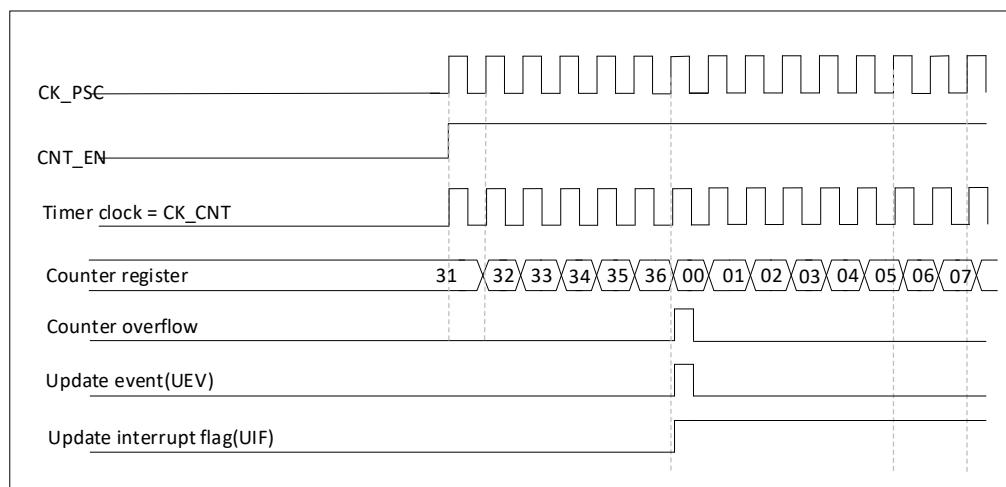


图 13-4 计数器时序图, 内部时钟分频因子为 1

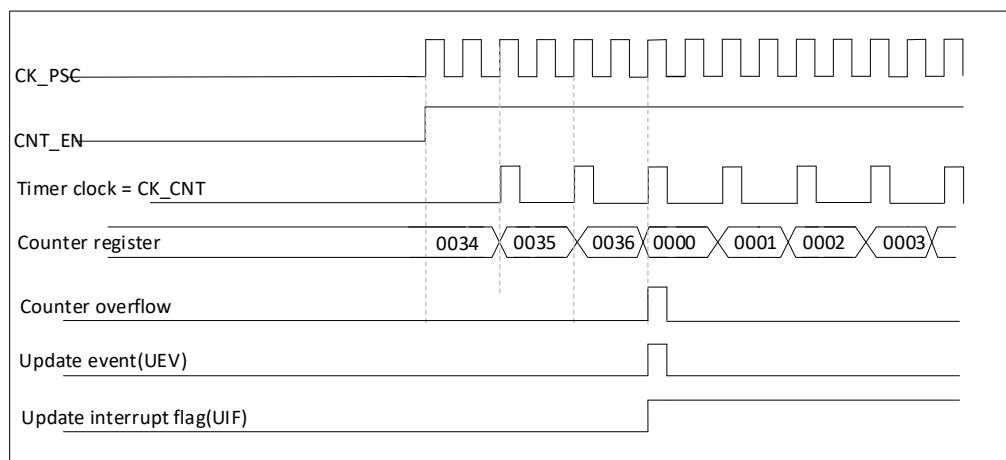


图 13-5 计数器时序图, 内部时钟分频因子为 2

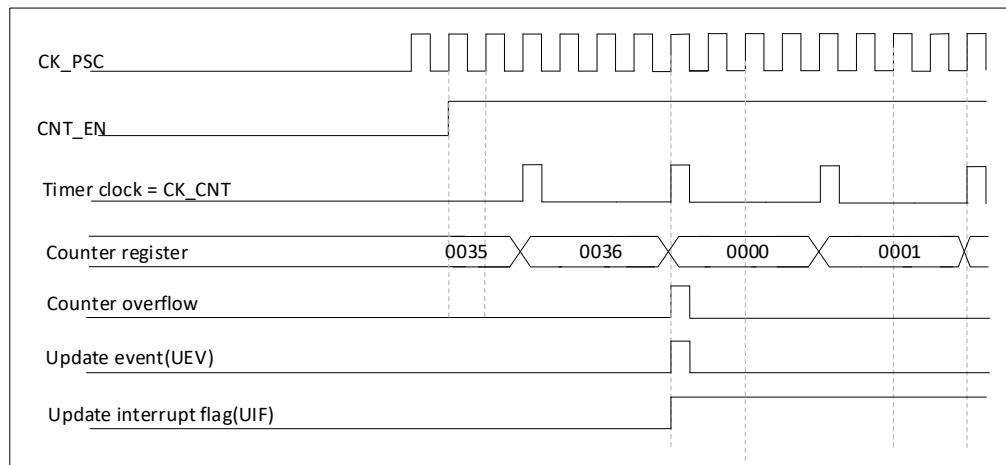


图 13-6 计数器时序图，内部时钟分频因子为 4

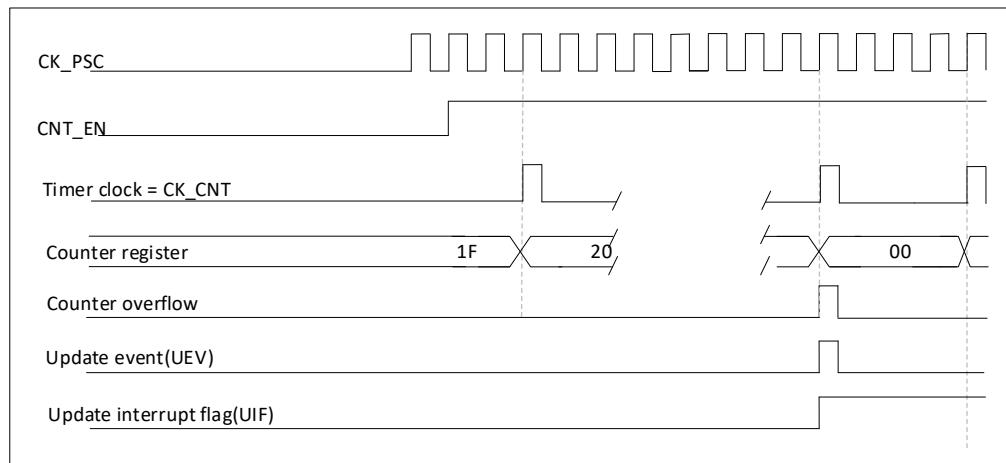


图 13-7 计数器时序图，内部时钟分频因子为 N

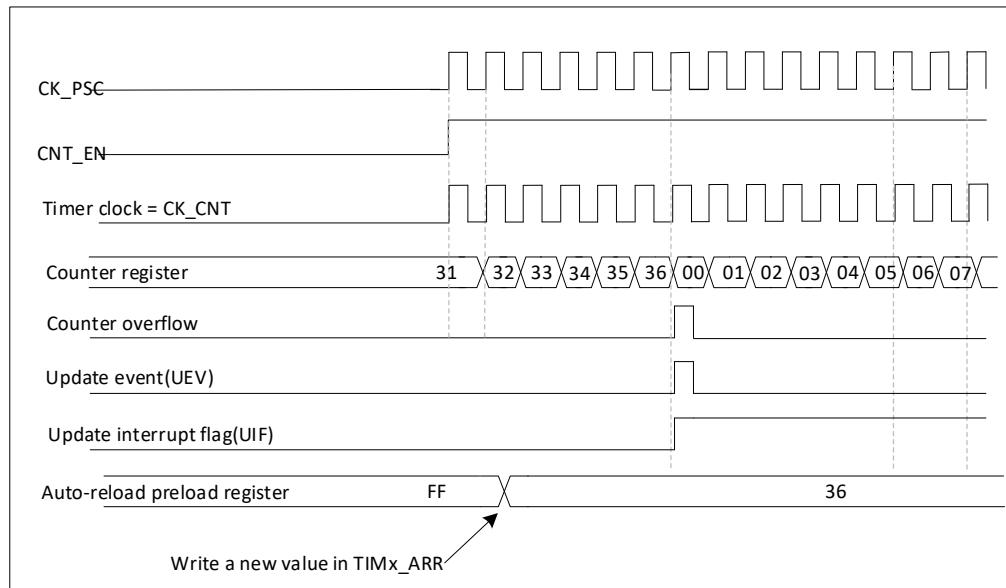


图 13-8 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx\_ARR)

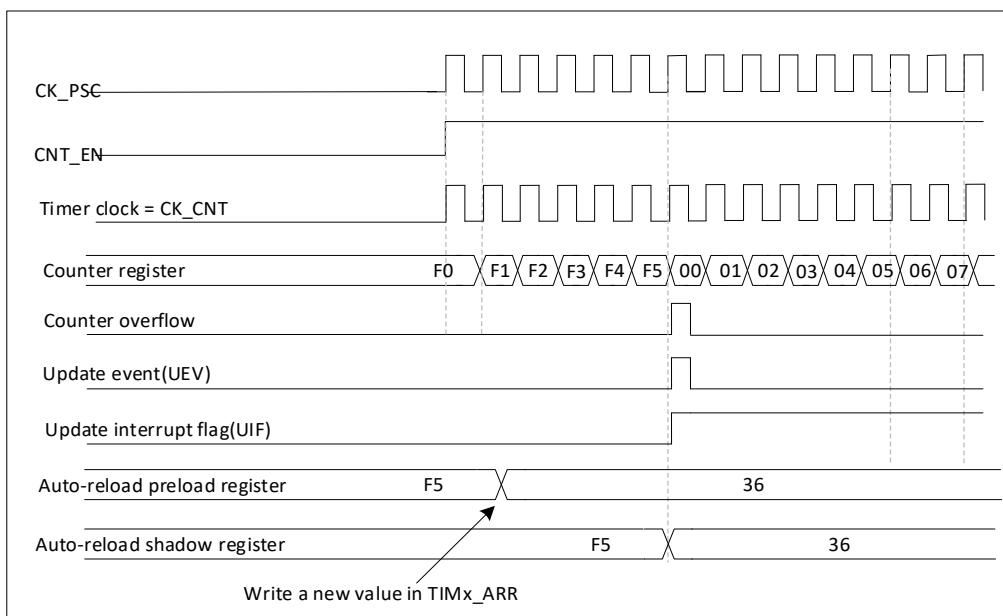


图 13-9 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx\_ARR)

### 13.2.2.2. 向下计数模式

在向下计数模式中，计数器从自动加载值(TIMx\_ARR 的内容)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数下溢事件。

如果使用了重复计数器，那么更新事件(UEV)需要在下溢次数达到所配置的重复计数寄存器的值加一（即 TIMx\_RCR+1）时才会产生；如果没有使用重复计数器（即 TIMx\_RCR=0），那么每次计数下溢都会产生更新事件。

而在 TIMx\_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清‘0’(但预分频系数不变)。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不置起 UIF 标志位(即不会产生中断或 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx\_SR 寄存器中的 UIF 位)：

- 重复计数器被重新加载为 TIMx\_RCR 寄存器中的内容。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx\_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx\_ARR 寄存器中的内容)。

注：自动加载值在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx\_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

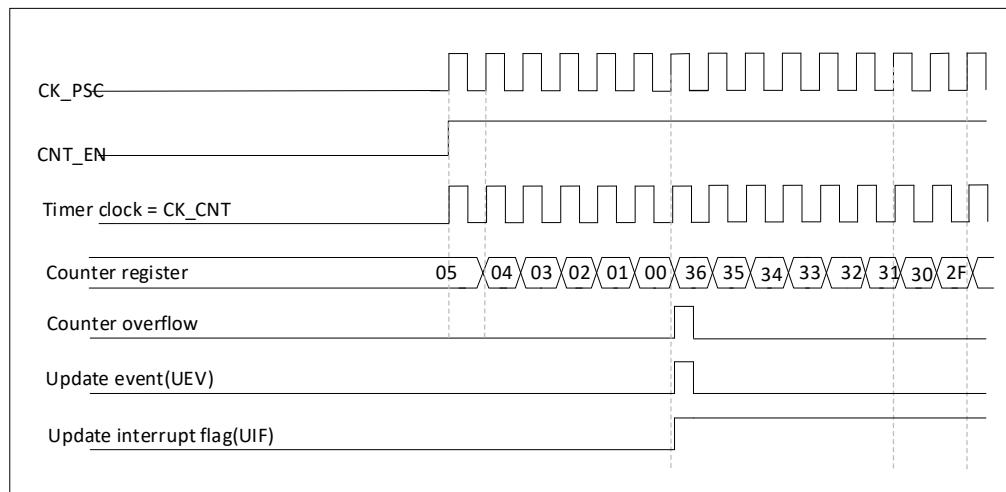


图 13-10 计数器时序图，内部时钟分频因子为 1

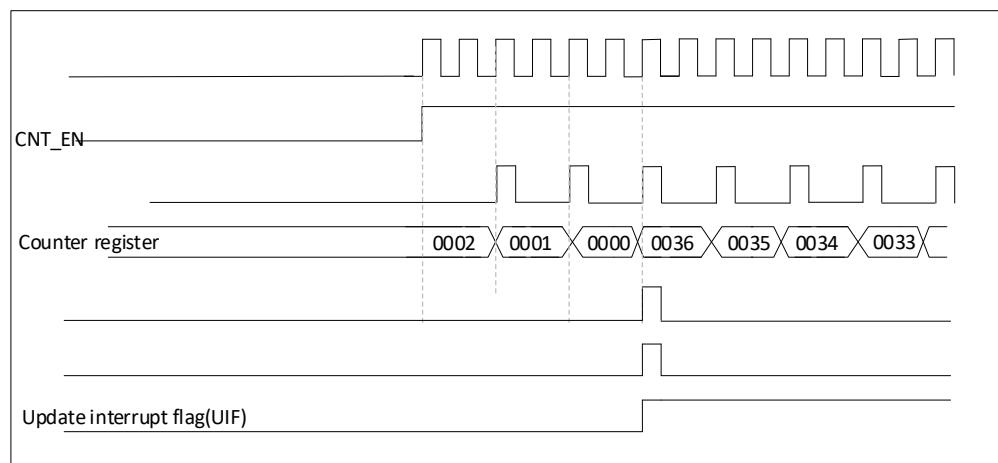


图 13-11 计数器时序图，内部时钟分频因子为 2

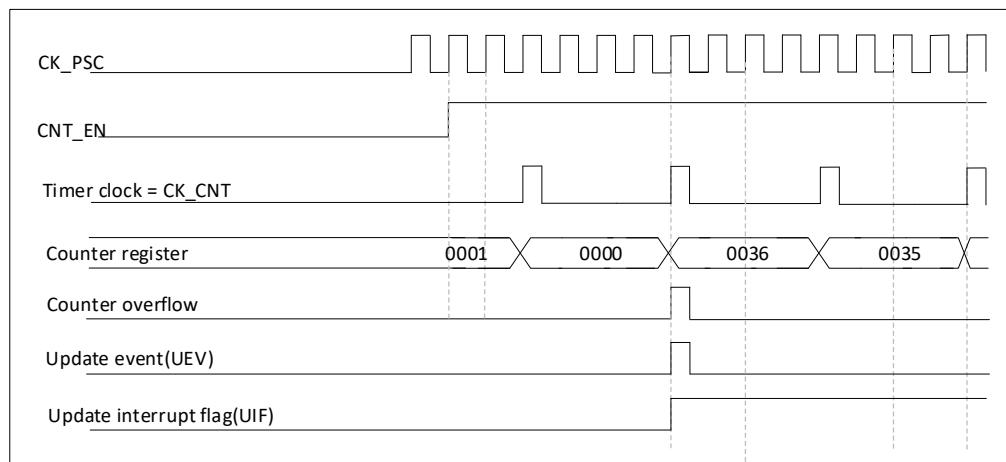


图 13-12 计数器时序图，内部时钟分频因子为 4

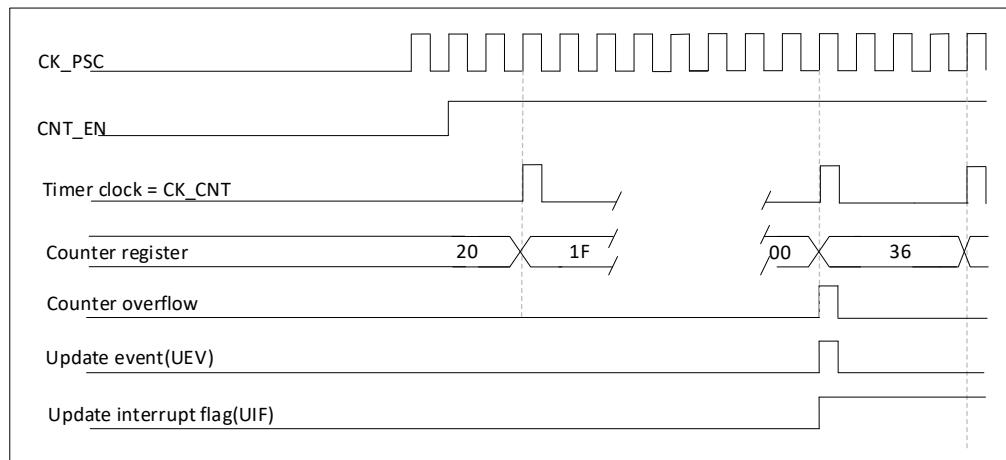


图 13-13 计数器时序图，内部时钟分频因子为 N

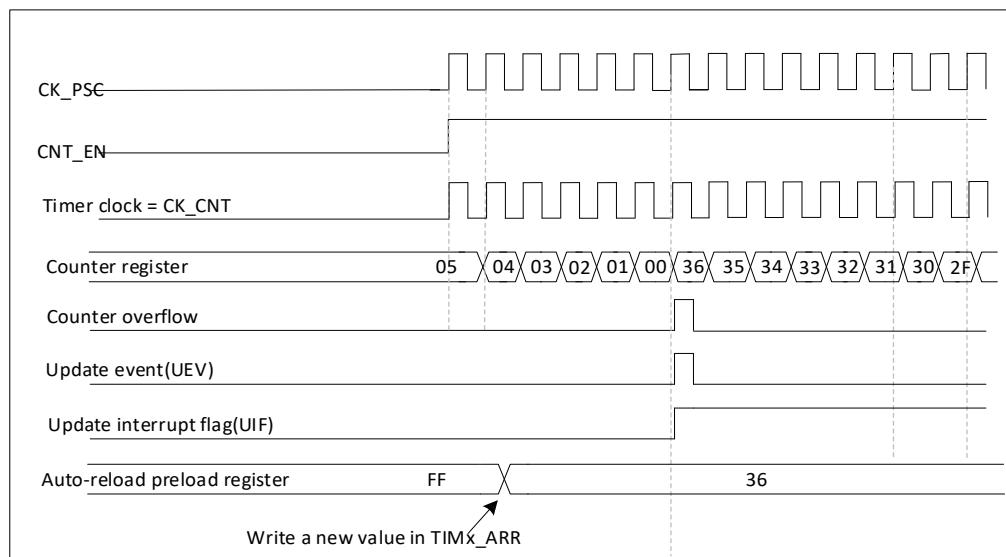


图 13-14 计数器时序图，当没有使用重复计数器时的更新事件

### 13.2.2.3. 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。

通过配置 TIMx\_CR1 寄存器中的 CMS 位不为‘00’可以得到中央对齐模式。在通道配置为输出模式下的输出比较标志位在以下几种计数过程中会被置起：向下计数时（中央对齐模式 1，CMS=‘01’）、向上计数时（中央对齐模式 2，CMS=‘10’）、向上和向下计数时（中央对齐模式 3，CMS=‘11’）。

在此模式下，不能写入 TIMx\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx\_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器内部计数器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止更新事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。虽然 UDIS 位被清为 0 之前不会产生更新事件。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(选择更新请求源)，通过设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx\_SR 寄存器中的 UIF 位)也被设置：

- 重复计数器被重置为 TIMx\_RCR 寄存器中的内容
- 预分频器的缓存器被加载为预装载(TIMx\_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx\_ARR 寄存器中的内容)。注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

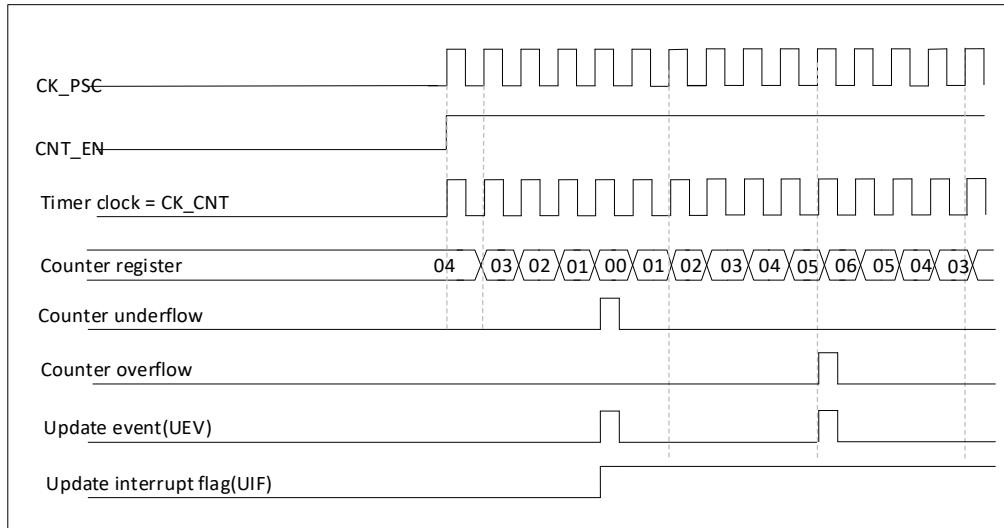


图 13-15 计数器时序图，内部时钟分频因子为 1， TIMx\_ARR=0x6

这里使用了中心对齐模式 1。

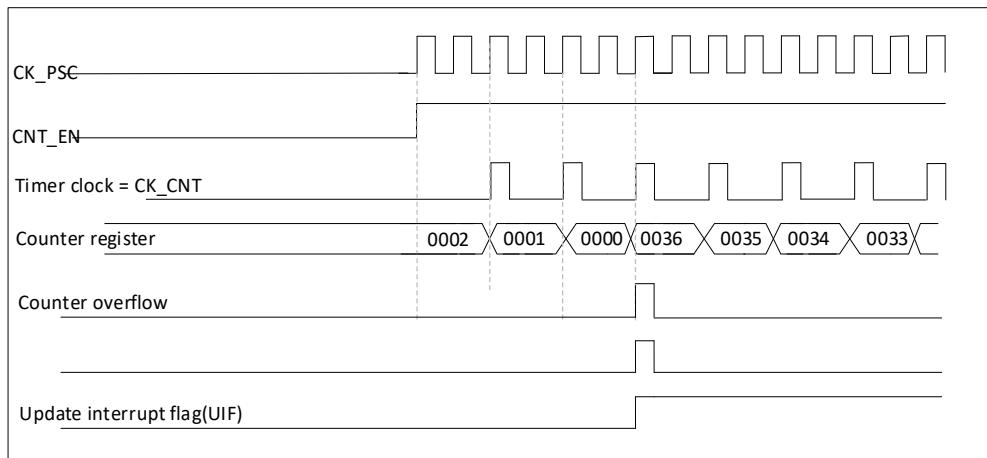
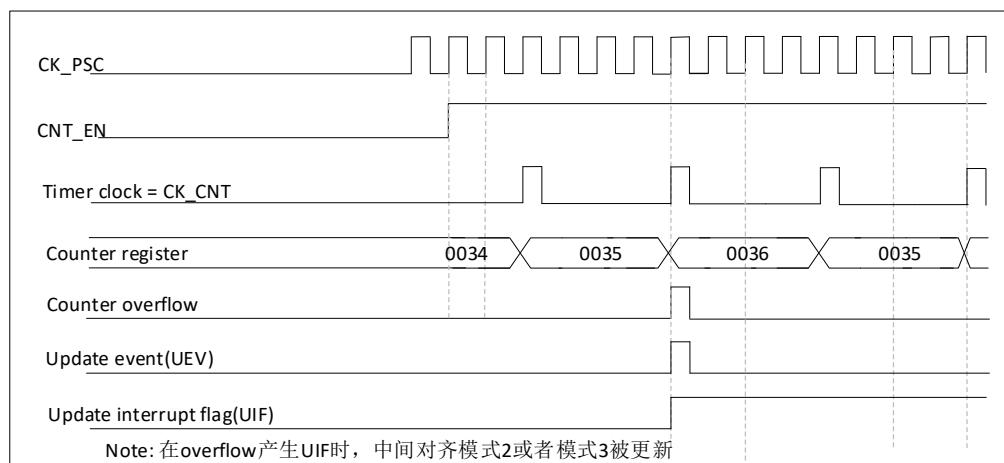
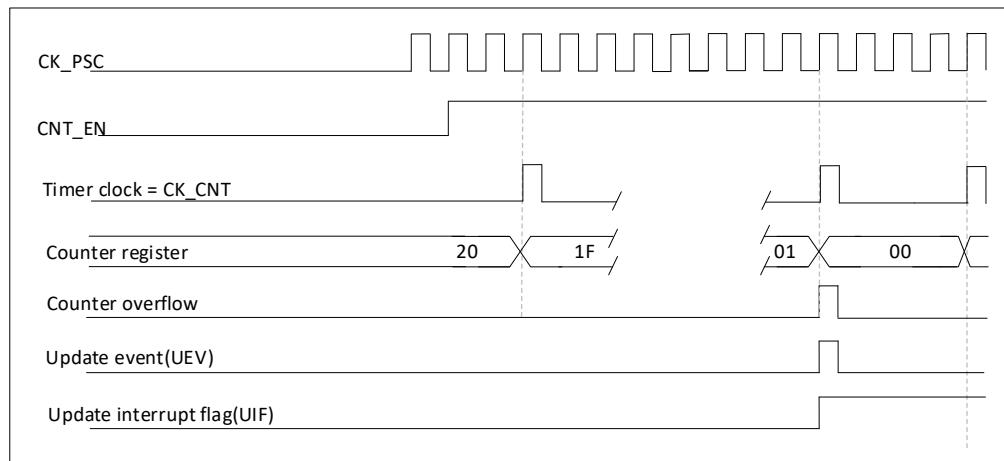
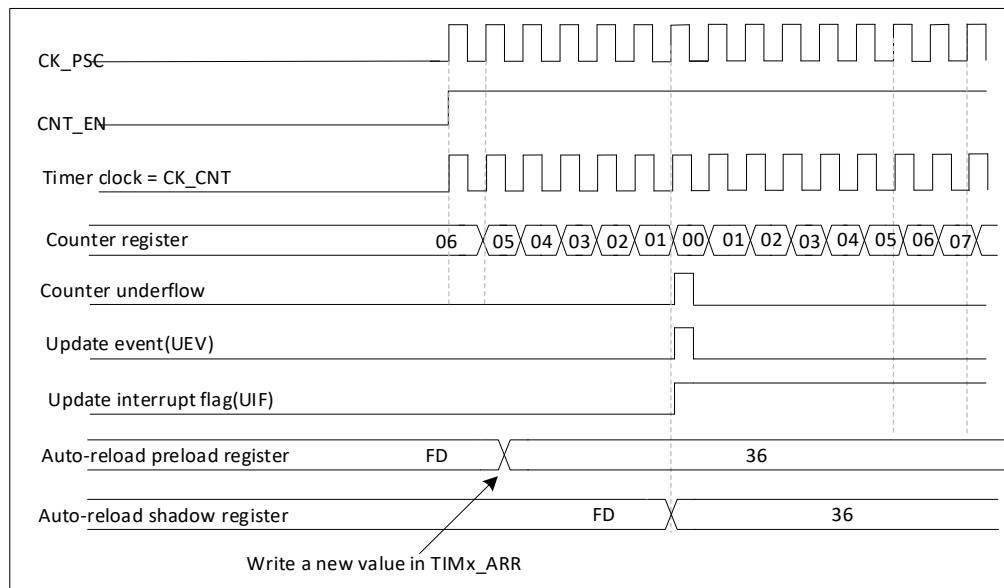


图 13-16 计数器时序图，内部时钟分频因子为 2

图 13-17 计数器时序图，内部时钟分频因子为 4， $\text{TIMx\_ARR}=0x36$ 图 13-18 计数器时序图，内部时钟分频因子为  $N$ 图 13-19 计数器时序图， $\text{ARPE}=1$  时的更新事件(计数器下溢)

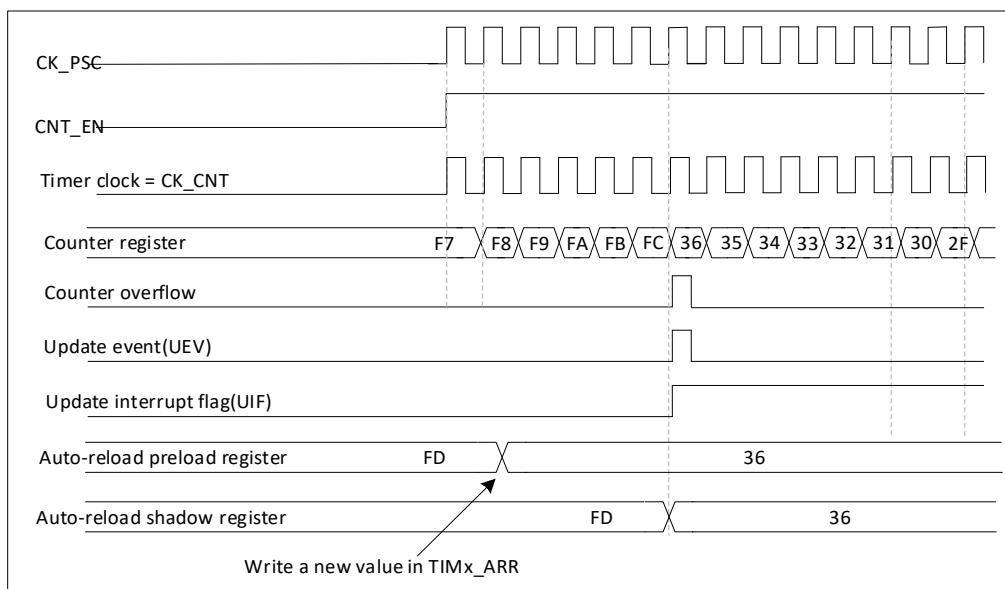


图 13-20 计数器时序图，ARPE=1 时的更新事件(计数器溢出)

### 13.2.3. 重复计数器

“时基单元”解释了计数器上溢/下溢时更新事件(UEV)是如何产生的，事实上它只能在重复计数器计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

这意味着在每 N+1 个计数上溢或下溢时，数据才会从预装载寄存器传输到影子寄存器(TIMx\_ARR 自动重载寄存器，TIMx\_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx\_CCRx)，N 是 TIMx\_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器上溢时，
- 向下计数模式下每次计数器下溢时，
- 中央对齐模式下每次上溢和每次下溢时。

虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为  $2 \times T_{ck}$ 。

重复计数器是自动重载的，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入到重复计数器。

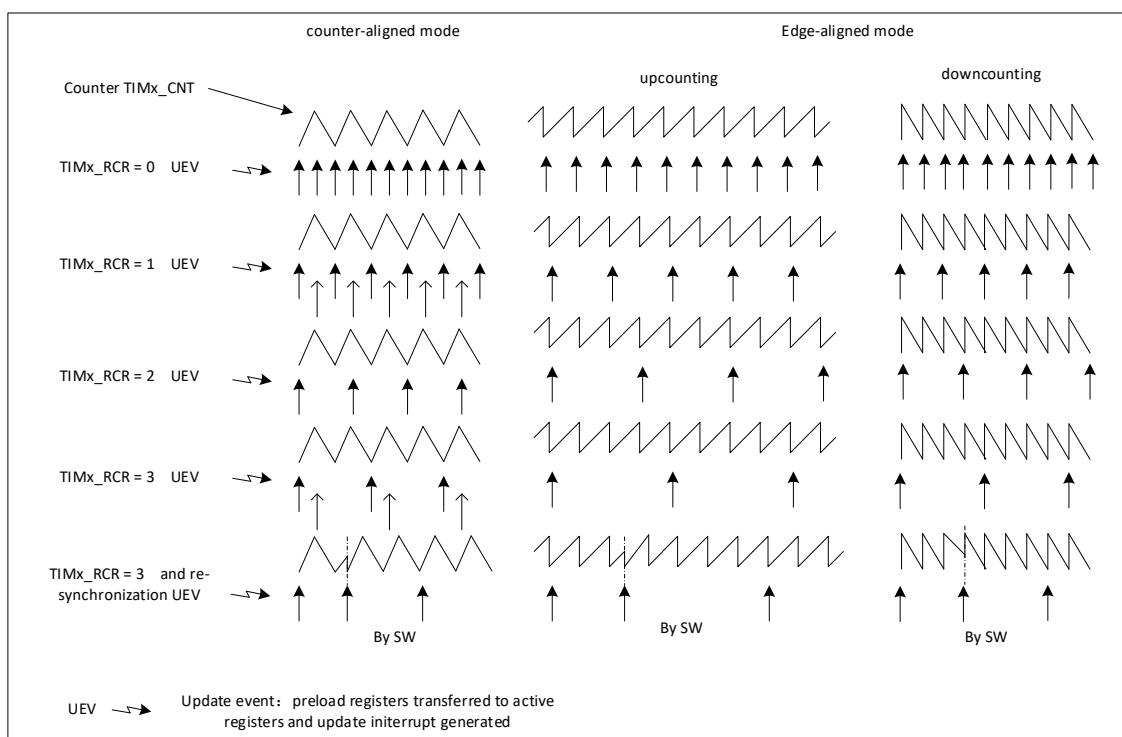


图 13-21 不同模式下更新速率的例子，及 TIMx\_RCR 的寄存器设置

### 13.2.4. 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK\_INT)
- 外部时钟模式 1：外部输入引脚
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

#### 13.2.4.1. 内部时钟源(CK\_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN、DIR(TIMx\_CR1 寄存器)和 UG 位(TIMx\_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成‘1’，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

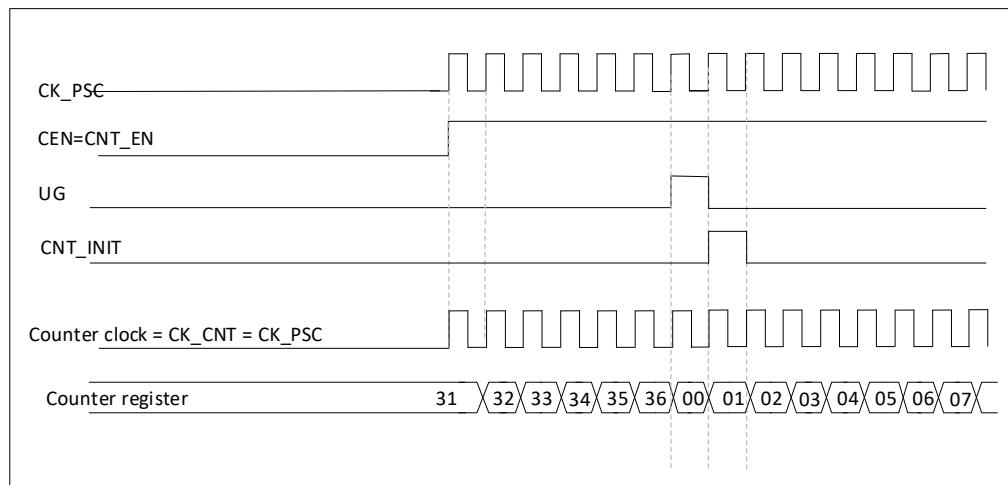


图 13-22 一般模式下的控制电路，内部时钟分频因子为 1

### 13.2.4.2. 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。

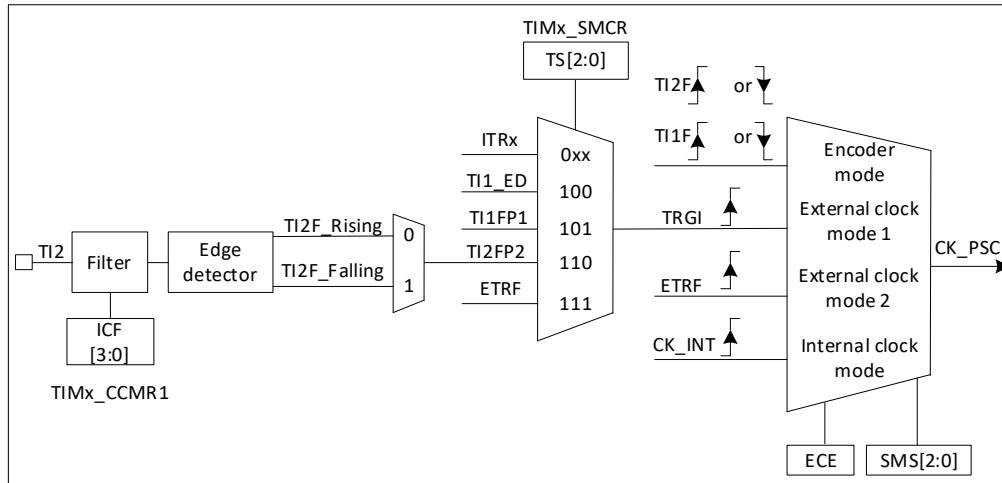


图 13-23 TIM2 外部时钟连接例子

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：

- 配置 TIMx\_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；
- 配置 TIMx\_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2F=0000）；
- 配置 TIMx\_CCER 寄存器的 CC2P=0，选定上升沿极性；
- 配置 TIMx\_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；
- 配置 TIMx\_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源；
- 设置 TIMx\_CR1 寄存器的 CEN=1，启动计数器。

注：捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

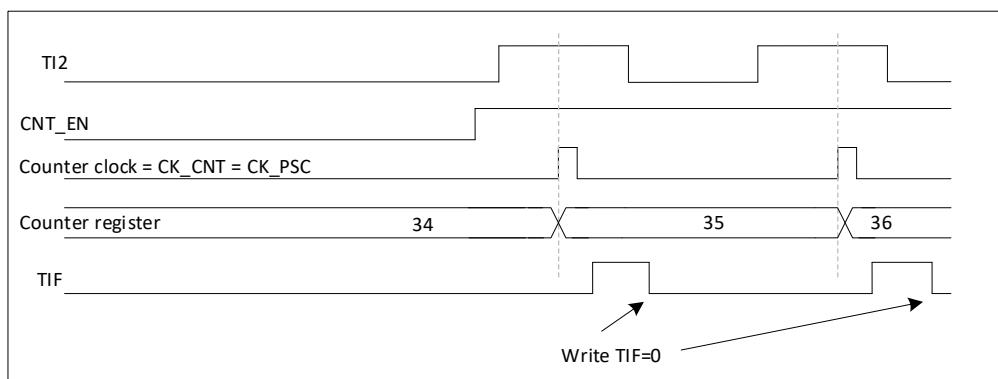


图 13-24 外部时钟模式 1 下的控制电路

### 13.2.4.3. 外部时钟源模式 2

选定此模式的方法为：令 TIMx\_SMCR 寄存器中的 ECE=1，计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图：

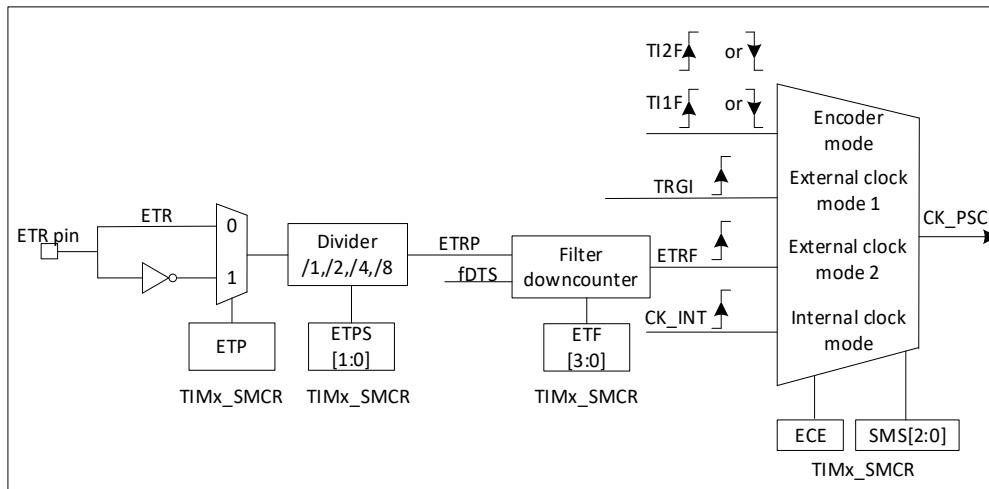


图 13-25 外部触发输入框图

例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

- 本例中不需要滤波器，置 TIMx\_SMCR 寄存器中的 ETF[3:0]=0000；
- 设置预分频器，置 TIMx\_SMCR 寄存器中的 ETPS[1:0]=01；
- 选择 ETR 输入端的上升沿，置 TIMx\_SMCR 寄存器中的 ETP=0；
- 开启外部时钟模式 2，写 TIMx\_SMCR 寄存器中的 ECE=1；
- 启动计数器，写 TIMx\_CR1 寄存器中的 CEN=1；

计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于 ETRP 信号的重新同步电路。

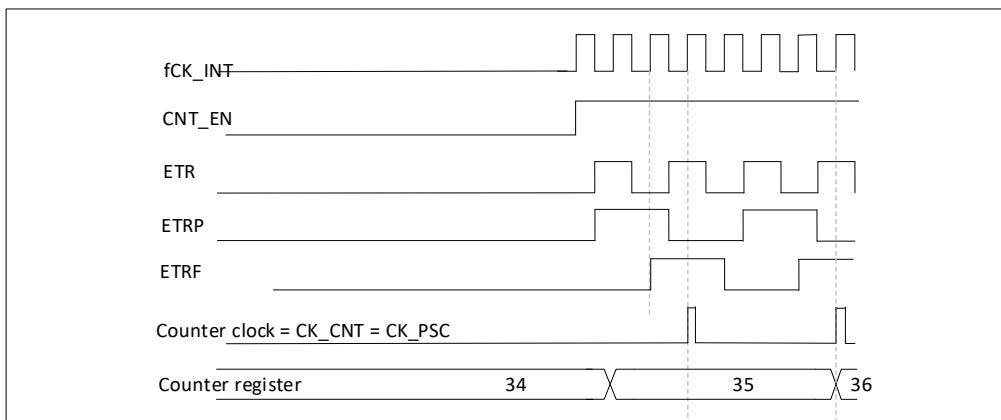


图 13-26 外部时钟模式 2 下的控制电路

### 13.2.5. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。

输入部分对相应的 TI<sub>x</sub> 输入信号采样，并产生一个滤波后的信号 TI<sub>x</sub>F。然后，一个带极性选择的边缘监测器产生一个信号(TIxFP<sub>x</sub>)，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)。在捕获寄存器前分频得到 ICxPS。

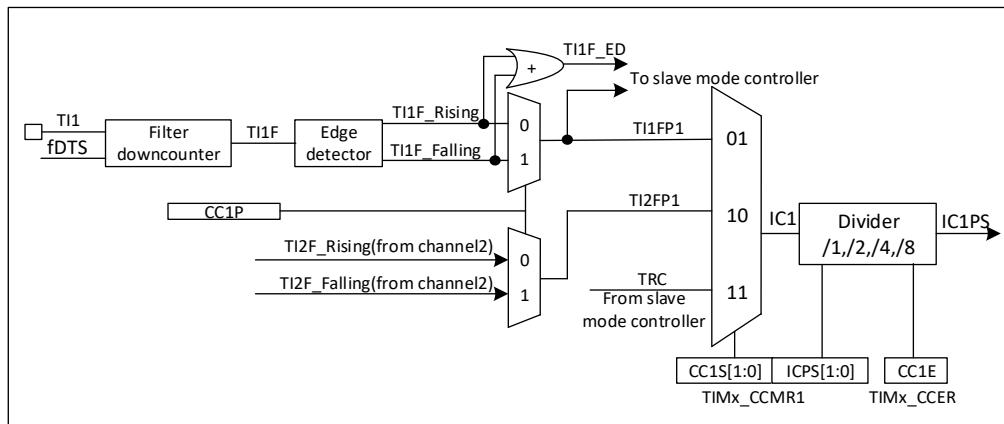


图 13-27 捕获/比较通道(如：通道 1 输入部分)

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。

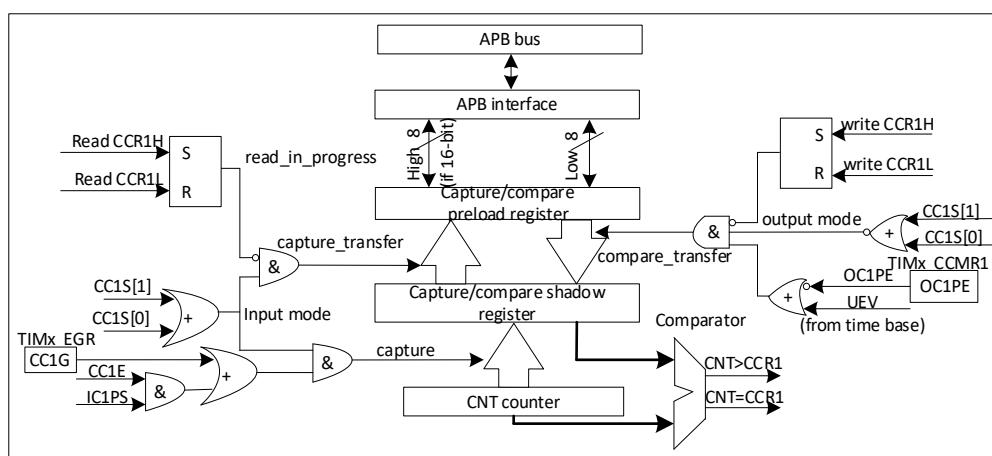


图 13-28 捕获/比较通道 1 的主电路

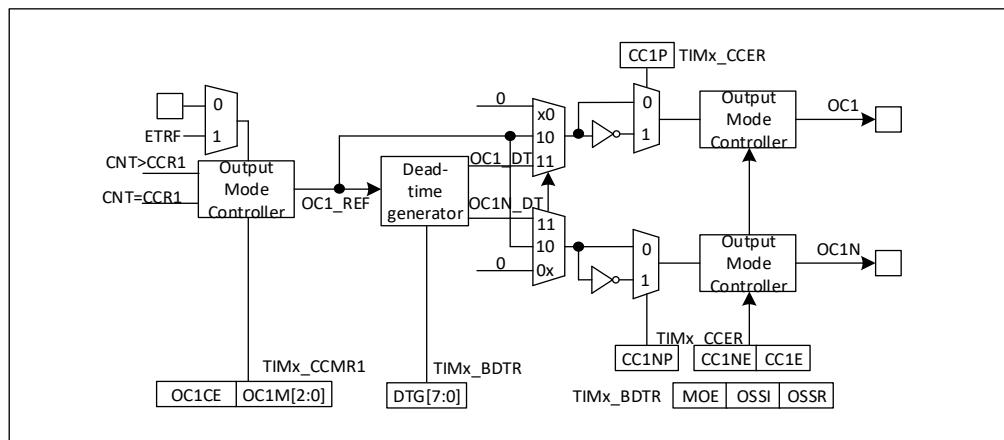


图 13-29 捕获/比较通道的输出部分(通道 1 至 3)

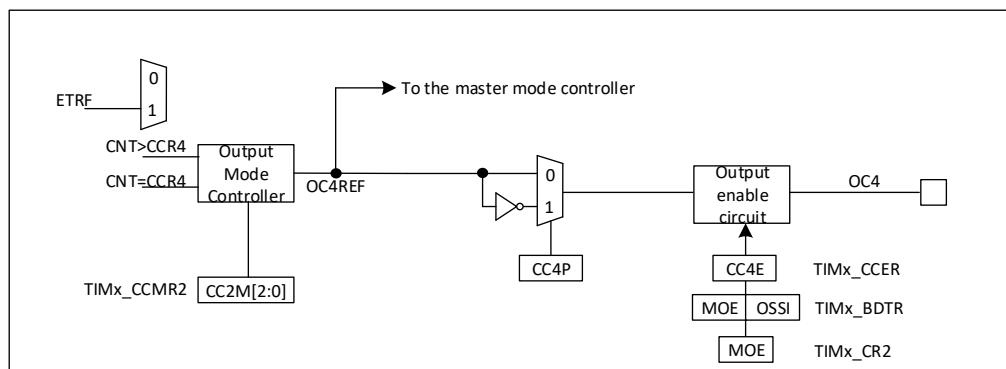


图 13-30 捕获/比较通道的输出部分(通道 4)

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。

### 13.2.6. 输入捕获模式

在输入捕获模式下，当检测到 IC<sub>x</sub> 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx\_CCR<sub>x</sub>)中。当发生捕获事件时，相应的 CC<sub>x</sub>IF 标志(TIMx\_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CC<sub>x</sub>IF 标志已经为高，那么过捕获标志 CC<sub>x</sub>OF(TIMx\_SR 寄存器)被置 1。通过写 CC<sub>x</sub>IF=0 可清除 CC<sub>x</sub>IF，或读取存储在 TIMx\_CCR<sub>x</sub> 寄存器中的捕获数据也可清除 CC<sub>x</sub>IF。写 CC<sub>x</sub>OF=0 可清除 CC<sub>x</sub>OF。

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\_CCMR1 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\_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道就被配置为输入，并且 TIMx\_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TI<sub>x</sub> 时，输入滤波器控制位是 TIMx\_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx\_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx\_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIMx\_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx\_DIER 寄存器中的 CC1IE 位允许相关中断请求，也可通过设置 TIMx\_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置  $\text{TIMx\_EGR}$  寄存器中相应的  $\text{CCxG}$  位，可以通过软件产生输入捕获中断和/或 DMA 请求。

### 13.2.7. PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，其余操作与输入捕获模式相同：

- 两个  $\text{ICx}$  信号被映射至同一个  $\text{TIx}$  输入。
- 这 2 个  $\text{ICx}$  信号为边沿有效，但是极性相反。
- 其中一个  $\text{TIxFP}$  信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，用户可以测量输入到  $\text{TI1}$  上的 PWM 信号的周期( $\text{TIMx\_CCR1}$  寄存器)和占空比( $\text{TIMx\_CCR2}$  寄存器)，具体步骤如下(取决于  $\text{CK\_INT}$  的频率和预分频器的值)。

- 选择  $\text{TIMx\_CCR1}$  的有效输入：置  $\text{TIMx\_CCMR1}$  寄存器的  $\text{CC1S}=01$ (选中  $\text{TI1}$ )。
- 选择  $\text{TI1FP1}$  的有效极性(用来捕获数据到  $\text{TIMx\_CCR1}$  中和清除计数器)：置  $\text{CC1P}=0$ (上升沿有效)。
- 选择  $\text{TIMx\_CCR2}$  的有效输入：置  $\text{TIMx\_CCMR1}$  寄存器的  $\text{CC2S}=10$ (选中  $\text{TI1}$ )。
- 选择有效的触发输入信号：置  $\text{TIMx\_SMCR}$  寄存器中的  $\text{TS}=101$ (选择  $\text{TI1FP1}$ )。
- 配置从模式控制器为复位模式：置  $\text{TIMx\_SMCR}$  中的  $\text{SMS}=100$ 。
- 使能捕获：置  $\text{TIMx\_CCER}$  寄存器中  $\text{CC1E}=1$  且  $\text{CC2E}=1$ 。

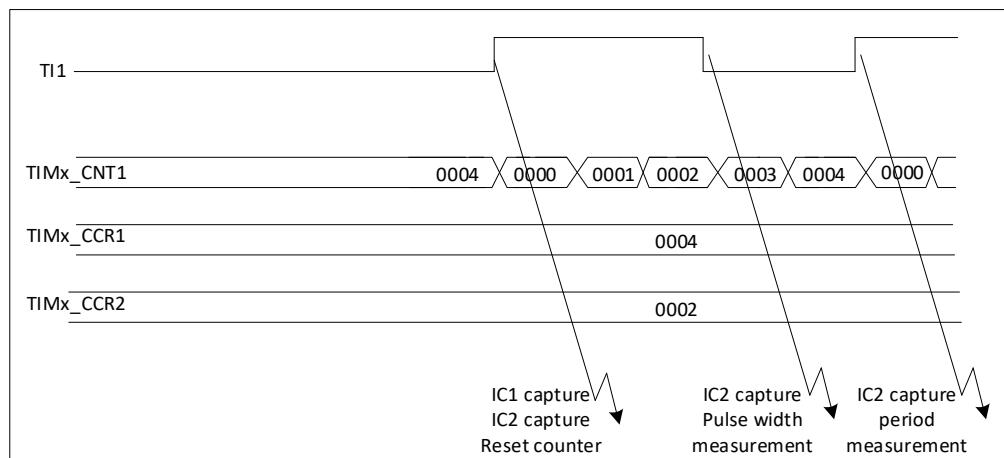


图 13-31 PWM 输入模式时序

因为只有  $\text{TI1FP1}$  和  $\text{TI2FP2}$  连到了从模式控制器，所以 PWM 输入模式只能使用  $\text{TIMx\_CH1/TIMx\_CH2}$  信号。

### 13.2.8. 强置输出模式

在输出模式( $\text{TIMx\_CCMRx}$  寄存器中  $\text{CCxS}=00$ )下，输出比较信号( $\text{OCxREF}$  和相应的  $\text{OCx/OCxN}$ )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  $\text{TIMx\_CCMRx}$  寄存器中相应的  $\text{OCxM}=101$ ，即可强置输出比较信号( $\text{OCxREF/OCx}$ )为有效状态。这样  $\text{OCxREF}$  被强置为高电平( $\text{OCxREF}$  始终为高电平有效)，同时  $\text{OCx}$  得到  $\text{CCxP}$  极性相反的信号。

例如： $\text{CCxP}=0$ ( $\text{OCx}$  高电平有效)，则  $\text{OCx}$  被强置为高电平。

置  $\text{TIMx\_CCMRx}$  寄存器中的  $\text{OCxM}=100$ ，可强置  $\text{OCxREF}$  信号为低。

该模式下，在  $\text{TIMx\_CCRx}$  影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

### 13.2.9. 输出比较模式

此项功能是用来控制一个输出波形，或者表明一段给定的时间已经到时。

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。

可以通过配置 TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

- 选择计数器时钟(内部，外部，预分频器)。
- 将相应数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
- 如果要产生一个中断请求，设置 CCxIE 位。
- 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
- 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。

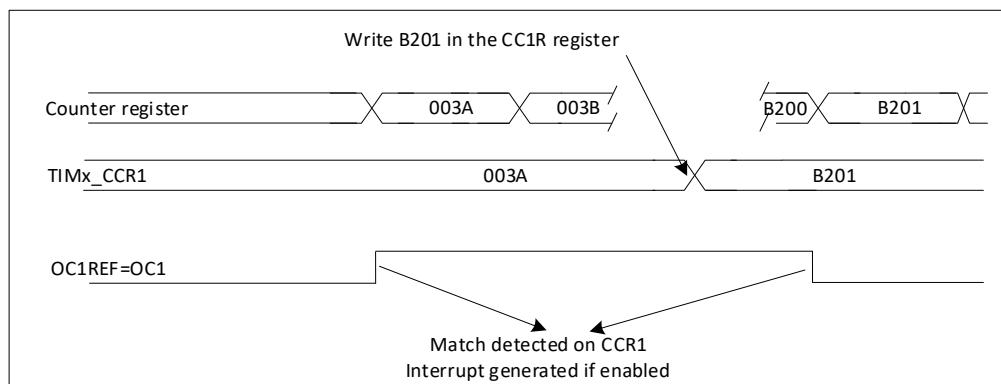


图 13-32 输出比较模式，翻转 OC1

### 13.2.10. PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入‘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，用户必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过(TIMx\_CCER 和 TIMx\_BDTR 寄存器中)CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。

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  $\text{TIMx\_CCR}x \leq \text{TIMx\_CNT}$  或者  $\text{TIMx\_CNT} \leq \text{TIMx\_CCR}x$ 。

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。

### 13.2.10.1. PWM 边沿对齐模式

#### ■ 向上计数配置

当 TIMx\_CR1 寄存器中的 DIR 位为低的时候执行向上计数。

下面是一个 PWM 模式 1 的例子。当  $\text{TIMx\_CNT} < \text{TIMx\_CCR}x$  时，PWM 参考信号 OCxREF 为高，否则为低。如果  $\text{TIMx\_CCR}x$  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。

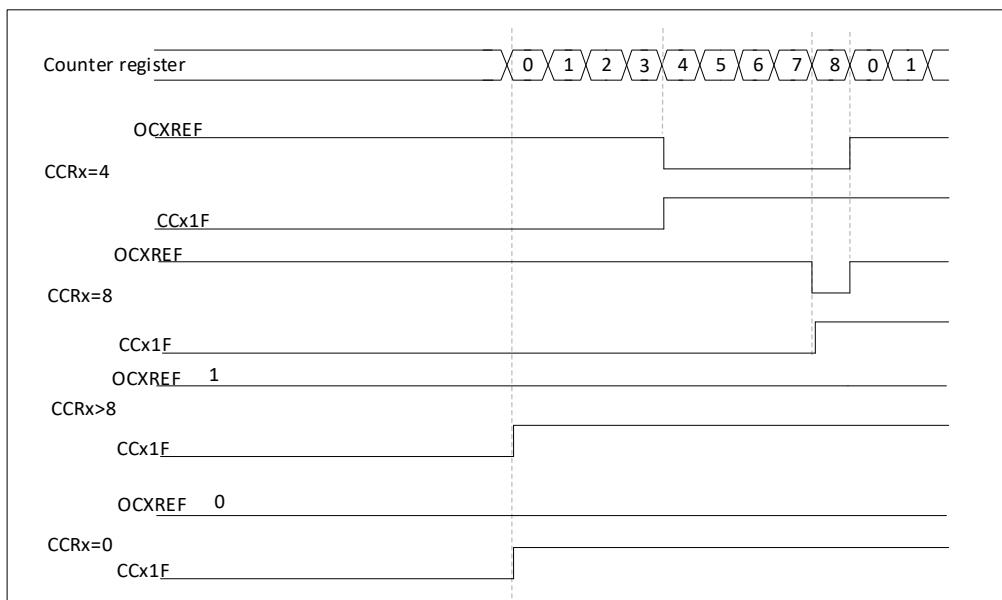


图 13-33 边沿对齐的 PWM 波形(ARR=8)

#### ■ 向下计数的配置

当 TIMx\_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当  $\text{TIMx\_CNT} > \text{TIMx\_CCR}x$  时参考信号 OCxREF 为低，否则为高。如果  $\text{TIMx\_CCR}x$  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。

### 13.2.10.2. PWM 中央对齐模式

当 TIMx\_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式(CMS 位的所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx\_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx\_ARR=8
- PWM 模式 1
- TIMx\_CR1 寄存器的 CMS=01，在中央对齐模式 1 下，当计数器向下计数时设置比较标志。

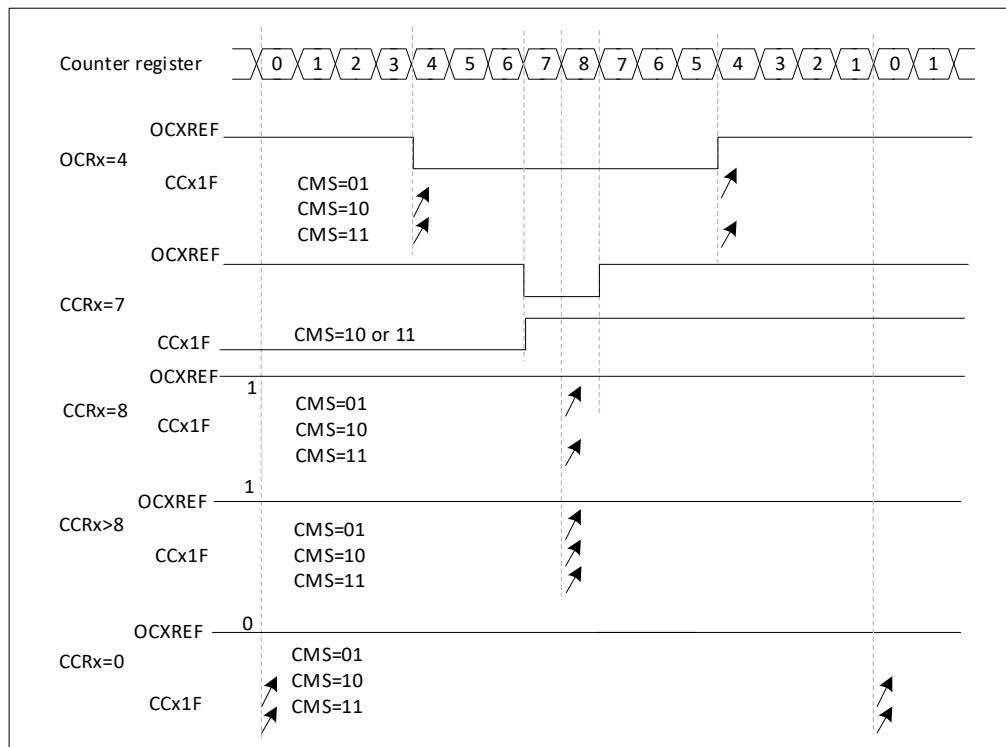


图 13-34 央对齐的 PWM 波形(APR=8)

使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx\_CR1 寄存器中 DIR 位的当前值。此外，不要通过软件同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值(TIMx\_CNT>TIMx\_ARR)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx\_ARR 的值写入计数器，方向被更新，但不会产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx\_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

### 13.2.11. 互补输出和死区插入

高级控制定时器(TIM1 和 TIM8)能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。

这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性(电平转换的延时、电源开关的延时等)来调整死区时间。

配置 TIMx\_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性(主输出 OCx 或互补输出 OCxN)。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TIMx\_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx\_BDTR 和 TIMx\_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLE 状态时(MOE 下降到 0)死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。通过配置 TIMx\_BDTR 寄存器中的 DTG[7:0]位，可以控制所有通道的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度(OCx 或者 OCxN)，则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。(假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1)

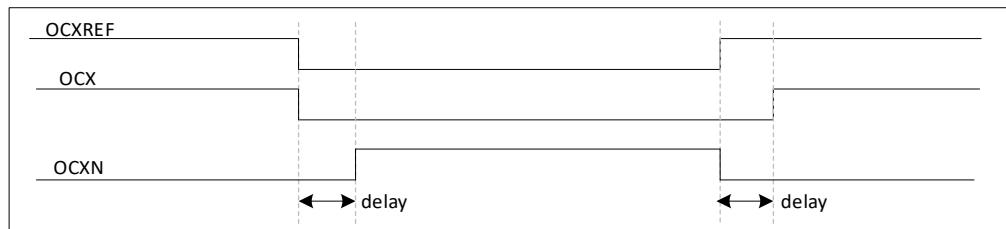


图 13-35 带死区插入的互补输出

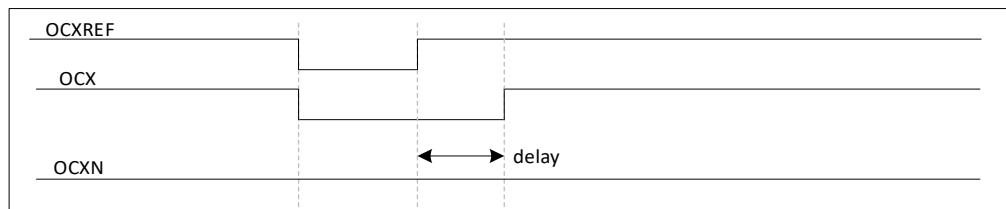


图 13-36 死区波形延迟大于负脉冲

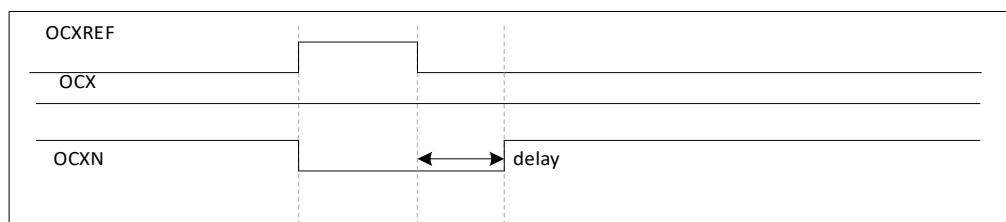


图 13-37 死区波形延迟大于正脉冲

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。详见 TIM1 和 TIM8 刹车和死区寄存器(TIMx\_BDTR)中具体的延时计算。

### 13.2.11.1. 重定向 OCxREF 到 OCx 或 OCxN

在输出模式下(强置、输出比较或 PWM)，通过配置 TIMx\_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形(例如 PWM 或者静态有效电平)。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

注：当只使能 OCxN(CCxE=0, CCxNE=1)时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时(CCxE=CCxNE=1)，当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

### 13.2.12. 使用刹车功能

当使用刹车功能时，依据相应的控制位(TIMx\_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx\_CR2 寄存器中的 OISx 和 OISxN 位)，输出使能信号和无效电平都会被修改。但无论何时，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见刹车功能的互补输出通道 OCx 和 OCxN 的控制位。

刹车源既可以是刹车输入引脚又可以是一个时钟失败事件。时钟失败事件由复位时钟控制器中的时钟安全系统产生，详见时钟安全系统(CSS)。

系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx\_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号(作用在输出端)和同步控制位(在 TIMx\_BDTR 寄存器中)之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时(空指令)才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时(在刹车输入端出现选定的电平)，有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态(由 OSSR 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 OSSR=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 OSSR=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\_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置‘1’；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

注：刹车输入为电平有效。所以，当刹车输入有效时，不能同时(自动地或者通过软件)设置 MOE。同时，状态标志 BIF 不能被清除。

刹车由 BRK 输入产生时，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(死区持续时间，OCx/OCxN 极性和被禁止的状态，OCxM 配置，刹车使能和极性)。用户可以通过设置 TIMx\_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，参看 TIM1 和 TIM8 刹车和死区寄存器(TIMx\_BDTR)。在 MCU 复位后 LOCK 位只能被修改一次。

下图显示响应刹车的输出实例。

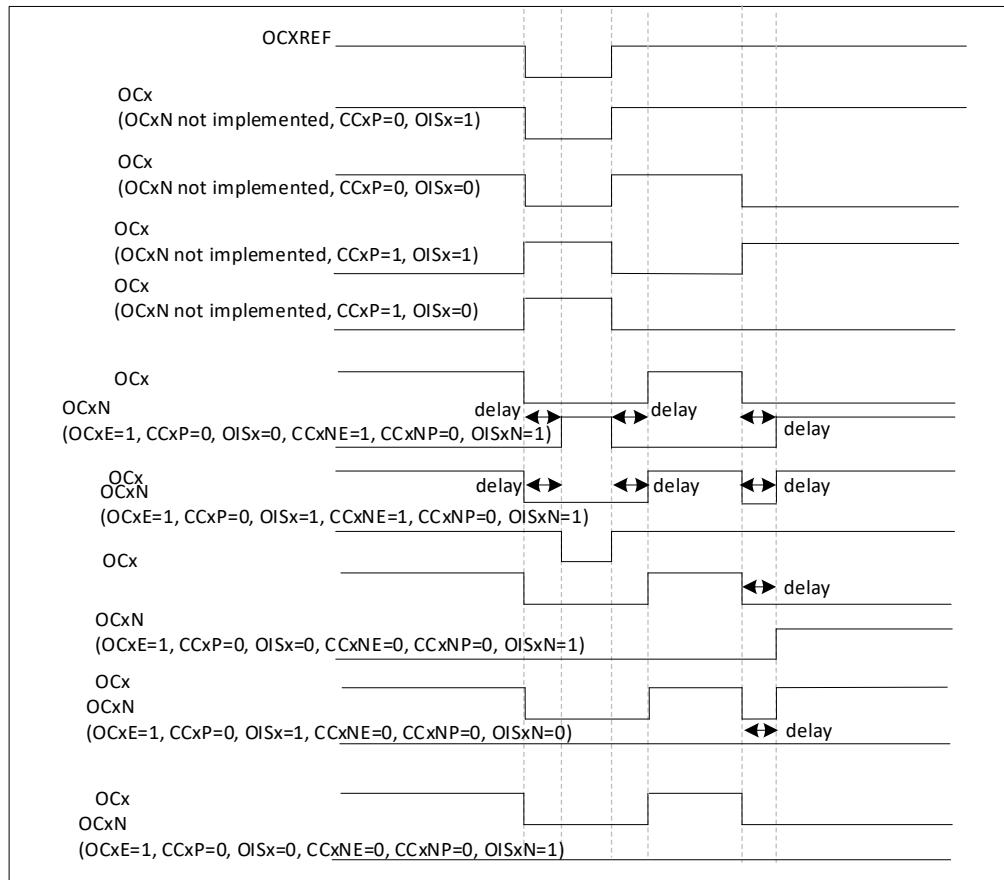


图 13-38 响应刹车的输出

### 13.2.13. 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx\_CCMRx 寄存器中对应的 OCxCE 位为‘1’，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

- 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。
- 必须禁止外部时钟模式 2：TIMx\_SMCR 寄存器中的 ECE=0。
- 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。

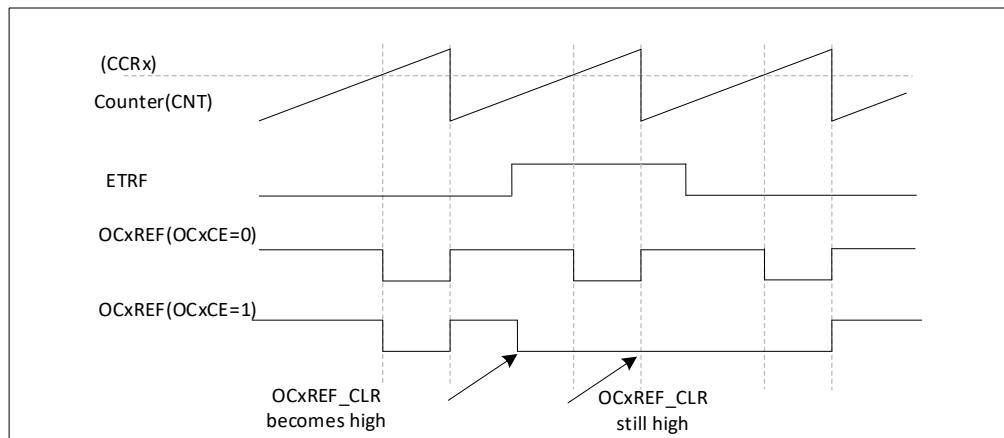


图 13-39 清除 TIMx 的 OCxREF

### 13.2.14. 产生六步 PWM 输出

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修改所有通道的配置。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。

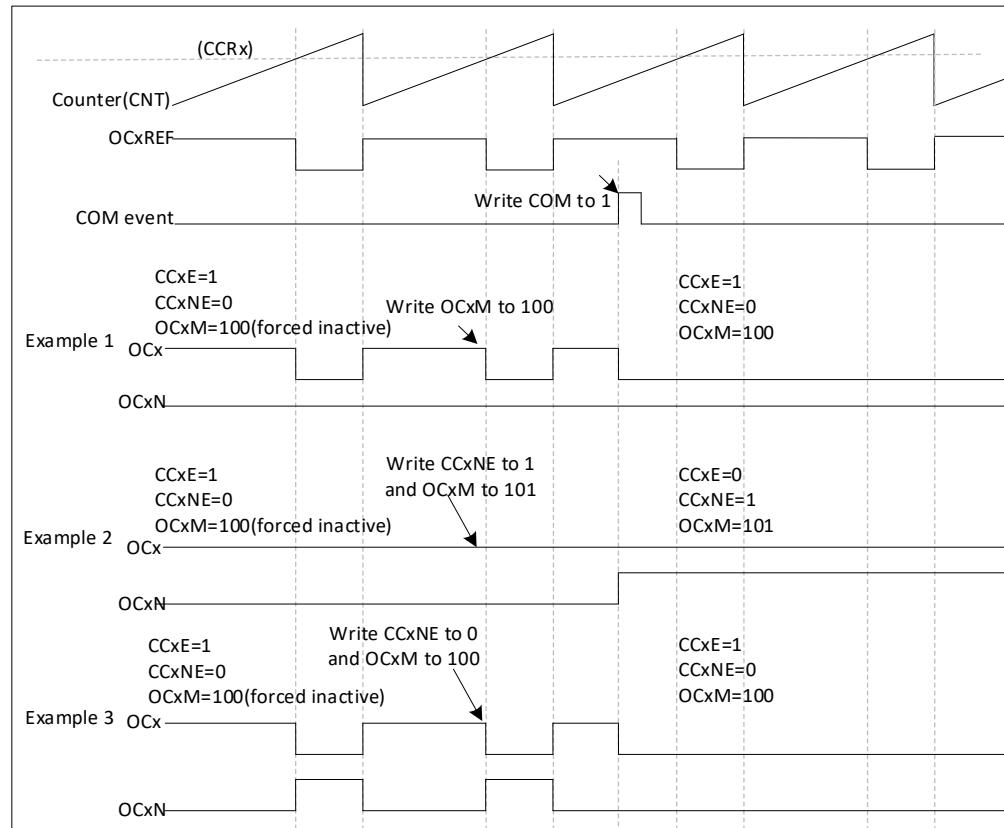


图 13-40 产生六步 PWM，使用 COM 的例子(OSSR=1)

### 13.2.15. 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$  (特别地,  $0 < CCRx$ )，
- 向下计数方式：计数器  $CNT > CCRx$ 。

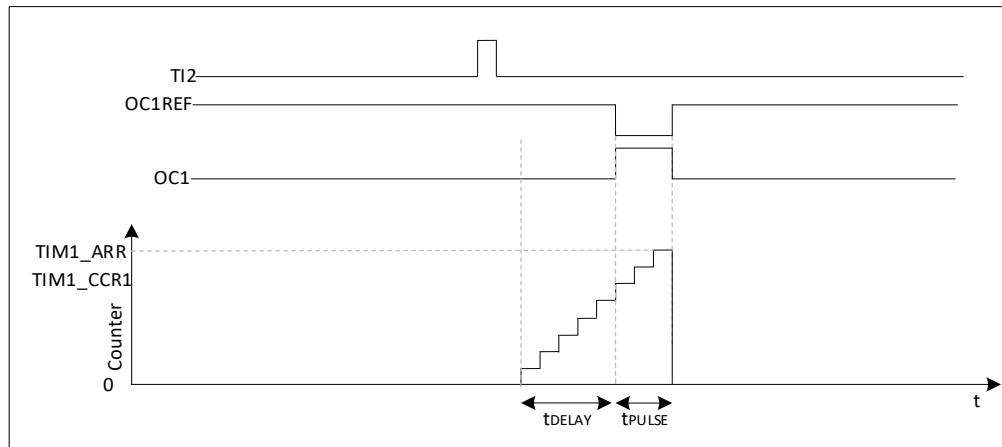


图 13-41 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 tDELAY 之后，在 OC1 上产生一个长度为 tPULSE 的正脉冲。

假定 TI2FP2 作为触发：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx\_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发(TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110(触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定(要考虑时钟频率和计数器预分频器)

- tDELAY 由 TIMx\_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义( $TIMx_ARR - TIMx_CCR1$ )。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE=1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。
- 在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM=1，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。当 OPM=0 时，重复模式被选中。

### 13.2.15.1. 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TI<sub>x</sub> 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 TIM<sub>x</sub>\_CCMR<sub>x</sub> 寄存器中的 OC<sub>x</sub>FE 位；此时 OC<sub>x</sub>REF(和 OC<sub>x</sub>)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。OC<sub>x</sub>FE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

### 13.2.16. 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 TI<sub>2</sub> 的边沿计数，则置 TIM<sub>x</sub>\_SMCR 寄存器中的 SMS=001；如果只在 TI<sub>1</sub> 边沿计数，则置 SMS=010；如果计数器同时在 TI<sub>1</sub> 和 TI<sub>2</sub> 边沿计数，则置 SMS=011。

通过设置 TIM<sub>x</sub>\_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI<sub>1</sub> 和 TI<sub>2</sub> 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI<sub>1</sub> 和 TI<sub>2</sub> 被用来作为增量编码器的接口。假定计数器已经启动(TIM<sub>x</sub>\_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI<sub>1</sub>FP1 或 TI<sub>2</sub>FP2 上的有效跳变驱动。TI<sub>1</sub>FP1 和 TI<sub>2</sub>FP2 是 TI<sub>1</sub> 和 TI<sub>2</sub> 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 TI<sub>1</sub>FP1=TI<sub>1</sub>，TI<sub>2</sub>FP2=TI<sub>2</sub>。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 TIM<sub>x</sub>\_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI<sub>1</sub> 计数、依靠 TI<sub>2</sub> 计数或者同时依靠 TI<sub>1</sub> 和 TI<sub>2</sub> 计数，在任一输入端(TI<sub>1</sub> 或者 TI<sub>2</sub>)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 TIM<sub>x</sub>\_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIM<sub>x</sub>\_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 TI<sub>1</sub> 和 TI<sub>2</sub> 不同时变换。

表 13-1 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI <sub>1</sub> FP1 的相对信号为 TI <sub>2</sub> ，TI <sub>2</sub> FP2 的相对信号为 TI <sub>1</sub> )	TI <sub>1</sub> FP1 信号		TI <sub>2</sub> FP2 信号	
		上升	下降	上升	下降
仅在 TI <sub>1</sub> 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 TI <sub>2</sub> 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 TI <sub>1</sub> 和 TI <sub>2</sub> 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S='01'(TIM<sub>x</sub>\_CCMR1 寄存器，TI<sub>1</sub>FP1 映射到 IC1)
- CC2S='01'(TIM<sub>x</sub>\_CCMR2 寄存器，TI<sub>2</sub>FP2 映射到 IC2)
- CC1P='0' (TIM<sub>x</sub>\_CCER 寄存器，TI<sub>1</sub>FP1 不反相，TI<sub>1</sub>FP1=TI<sub>1</sub>)
- CC2P='0' (TIM<sub>x</sub>\_CCER 寄存器，TI<sub>2</sub>FP2 不反相，TI<sub>2</sub>FP2=TI<sub>2</sub>)

- 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)

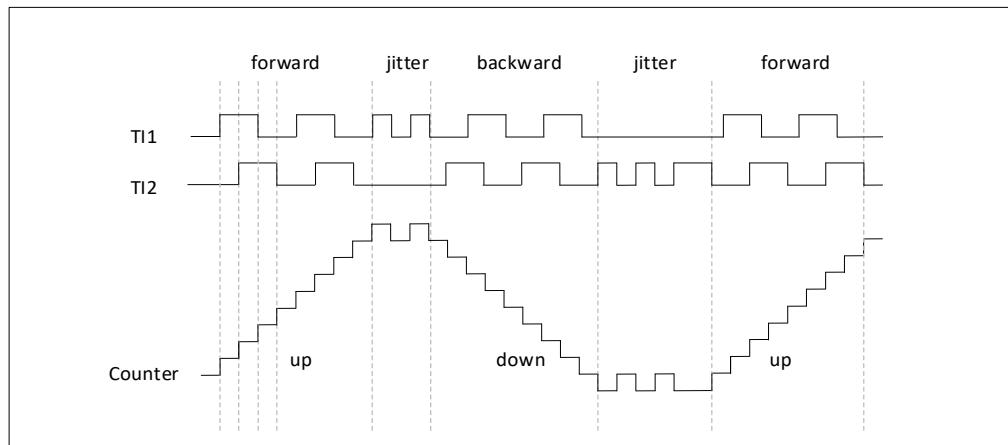


图 13-42 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1', 其他配置与上例相同)

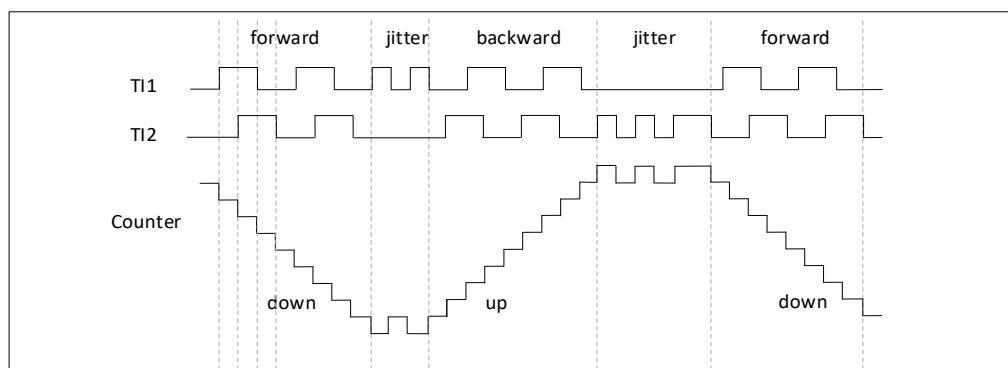


图 13-43 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时, 可以提供传感器当前位置的信息。通过将第二个定时器配置在捕获模式, 可以测量两个编码器事件的间隔, 获得动态的信息(速度, 加速度, 减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔, 可以按照固定的时间读出计数器。如果可能的话, 你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生); 也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 13.2.17. 定时器输入异或功能

TIMx\_CR2 寄存器中的 TI1S 位, 允许通道 1 的输入滤波器连接到一个异或门的输出端, 异或门的 3 个输入端为 TIMx\_CH1、TIMx\_CH2 和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能, 如触发或输入捕获。下节给出了此特性用于连接霍尔传感器的例子。

### 13.2.18. 与霍尔传感器的接口

使用高级控制定时器(TIM1 或 TIM8)产生 PWM 信号驱动马达时, 可以用另一个通用 TIMx(TIM2、TIM3、TIM4 或 TIM5)定时器作为“接口定时器”来连接霍尔传感器, 见图 4-43, 3 个定时器输入引脚(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), 此时“接口定时器”被用来捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F\_ED。每当 3 个输入之一变化时，计数器从新从 0 开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC(见图 4-26)。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以用来在输出模式产生一个脉冲，这个脉冲可以(通过触发一个 COM 事件)用于改变高级定时器 TIM1 或 TIM8 各个通道的属性，而高级控制定时器产生 PWM 信号驱动马达。

因此“接口定时器”通道必须编程为在一个指定的延时(输出比较或 PWM 模式)之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TIM1 或 TIM8。

举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIMx 的 PWM 配置。

- 置 TIMx\_CR2 寄存器的 TI1S 位为‘1’，配置三个定时器输入逻辑或到 TI1 输入；
- 时基编程：置 TIMx\_ARR 为其最大值(计数器必须通过 TI1 的变化清零)。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔；
- 设置通道 1 为捕获模式(选中 TRC)：置 TIMx\_CCMR1 寄存器中 CC1S=01，如果需要，还可以设置数字滤波器；
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIMx\_CCMR1 寄存器中的 OC2M=111 和 CC2S=00；
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx\_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的(TIMx\_CR2 寄存器中 CCPc=1)，同时触发输入控制 COM 事件(TIMx\_CR2 寄存器中 CCUS=1)。在一次 COM 事件后，写入下一步的 PWM 控制位(CCxE、OCxM)，这可以在处理 OC2REF 上升沿的中断子程序里实现。

下图显示了这个实例

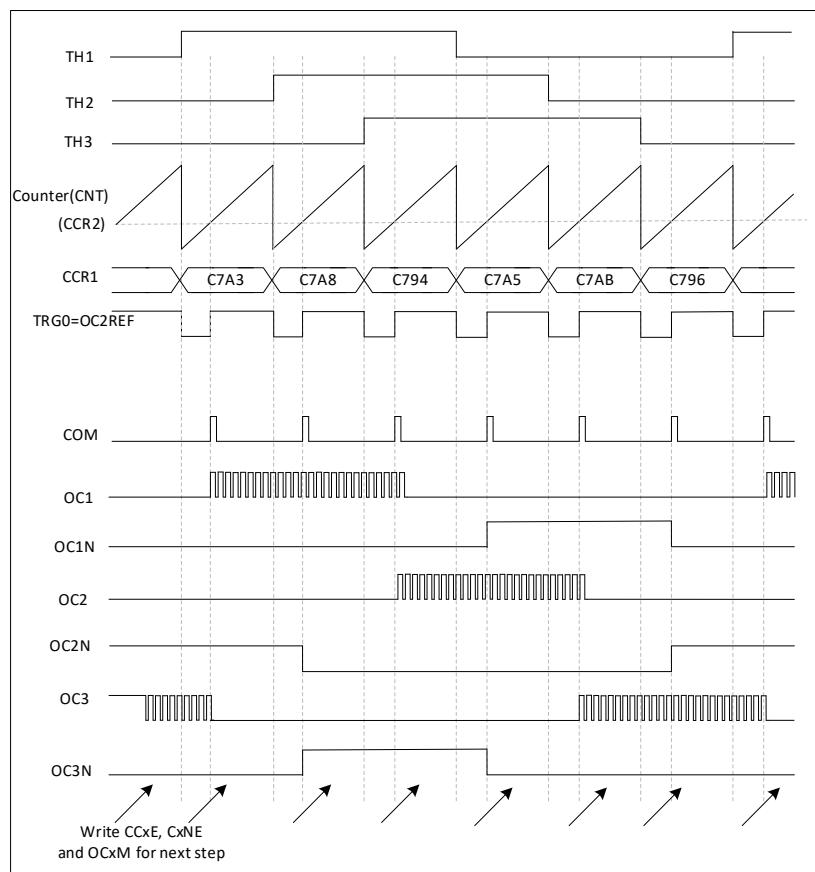


图 13-44 霍尔传感器接口的实例

### 13.2.19. TIMx 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 13.2.19.1. 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx\_ARR, TIMx\_CCRx)都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx\_SR 寄存器中的 TIF 位)被设置，根据 TIMx\_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx\_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

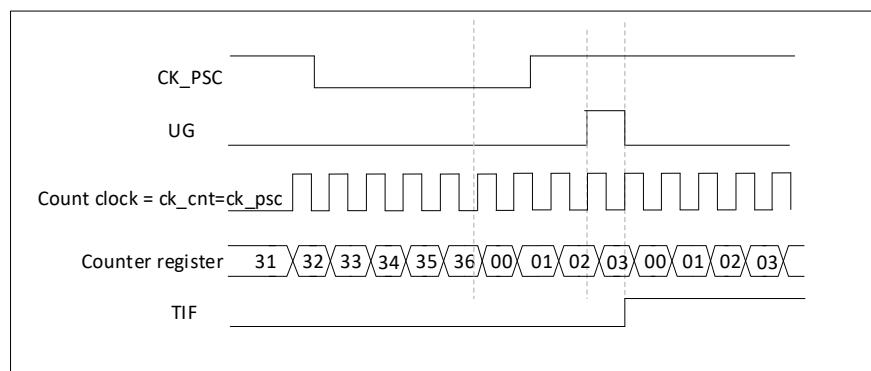


图 13-45 复位模式下的控制电路

### 13.2.19.2. 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

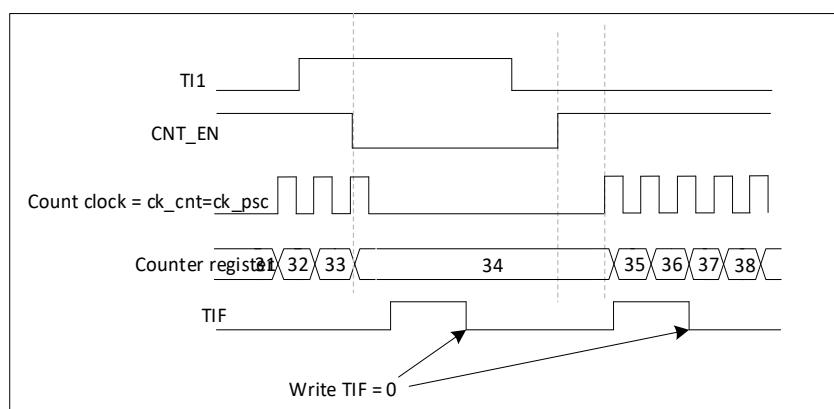


图 13-46 门控模式下的控制电路

### 13.2.19.3. 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  $\text{TIMx\_SMCR}$  寄存器中  $\text{SMS}=110$ , 配置定时器为触发模式; 置  $\text{TIMx\_SMCR}$  寄存器中  $\text{TS}=110$ , 选择  $\text{TI2}$  作为输入源。

当  $\text{TI2}$  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  $\text{TIF}$  标志。

$\text{TI2}$  上升沿和计数器启动计数之间的延时, 取决于  $\text{TI2}$  输入端的重同步电路。

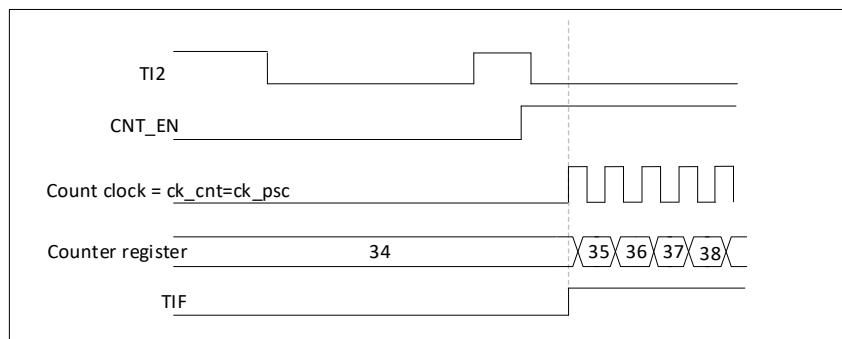


图 13-47 触发器模式下的控制电路

#### 13.2.19.4. 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,  $\text{ETR}$  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  $\text{TIMx\_SMCR}$  寄存器的  $\text{TS}$  位选择  $\text{ETR}$  作为  $\text{TRGI}$ 。

在下面的例子中, 一旦在  $\text{TI1}$  上出现一个上升沿, 计数器即在  $\text{ETR}$  的每一个上升沿向上计数一次:

- 通过  $\text{TIMx\_SMCR}$  寄存器配置外部触发输入电路:
  - $\text{ETF}=0000$ : 没有滤波
  - $\text{ETPS}=00$ : 不用预分频器
  - $\text{ETP}=0$ : 检测  $\text{ETR}$  的上升沿, 置  $\text{ECE}=1$  使能外部时钟模式 2。
- 按如下配置通道 1, 检测  $\text{TI}$  的上升沿:
  - $\text{IC1F}=0000$ : 没有滤波
  - 触发操作中不使用捕获预分频器, 不需要配置
  - 置  $\text{TIMx\_CCMR1}$  寄存器中  $\text{CC1S}=01$ , 选择输入捕获源
  - 置  $\text{TIMx\_CCER}$  寄存器中  $\text{CC1P}=0$  以确定极性(只检测上升沿)
- 置  $\text{TIMx\_SMCR}$  寄存器中  $\text{SMS}=110$ , 配置定时器为触发模式。置  $\text{TIMx\_SMCR}$  寄存器中  $\text{TS}=101$ , 选择  $\text{TI1}$  作为输入源。

当  $\text{TI1}$  上出现一个上升沿时,  $\text{TIF}$  标志被设置, 计数器开始在  $\text{ETR}$  的上升沿计数。

$\text{ETR}$  信号的上升沿和计数器实际复位间的延时, 取决于  $\text{ETRP}$  输入端的重同步电路。

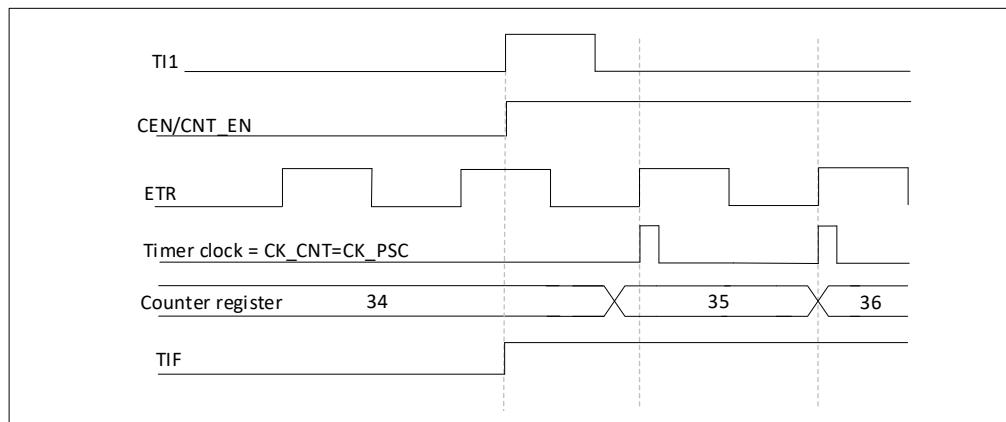


图 13-48 外部时钟模式 2 + 触发模式下的控制电路

### 13.2.20. 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

#### 13.2.20.1. 使用一个定时器作为另一个定时器的预分频器

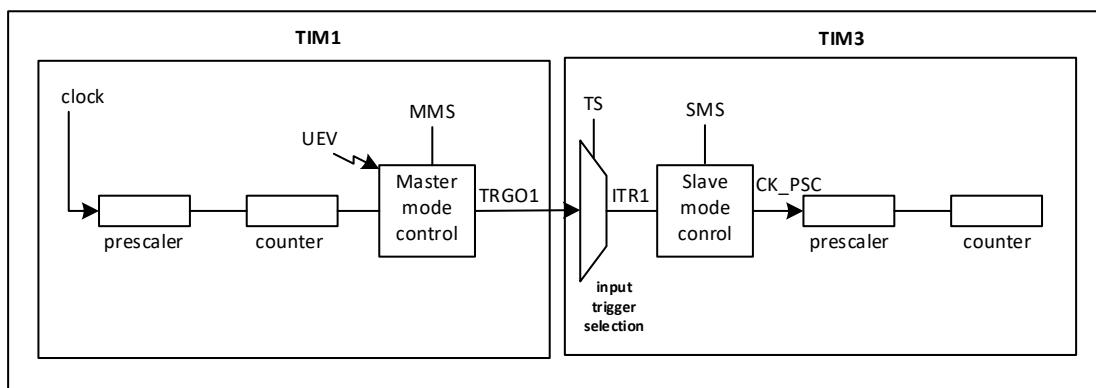


图 13-49 主/从定时器的例子

如：可以配置定时器 1 作为定时器 2 的预分频器。进行下述操作：

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1\_CR2 寄存器的 MMS='010' 时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。
- 连接定时器 1 的 TRGO1 输出至定时器 2，设置 TIM2\_SMCR 寄存器的 TS='000'，配置定时器 2 为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式 1(TIM2\_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期性的上升沿(即定时器 1 的计数器溢出)信号驱动。
- 最后，必须设置相应(TIMx\_CR1 寄存器)的 CEN 位分别启动两个定时器。

注：如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

#### 13.2.20.2. 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。只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以 3(fCK\_CNT=fCK\_INT/3)得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1\_CR2 寄存器的 MMS=100)

- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM2\_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1

注：定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

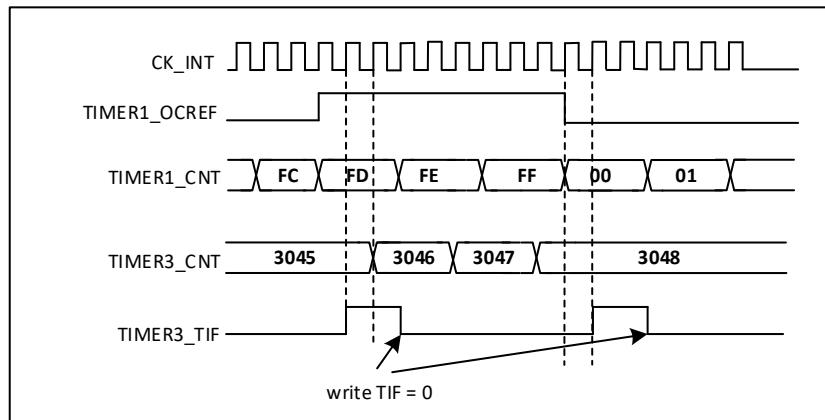


图 13-50 定时器 1 的 OC1REF 控制定时器 2

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 TIM1\_CR1 的 CEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1\_CR2 寄存器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM1\_EGR 寄存器的 UG='1'，复位定时器 1。
- 置 TIM2\_EGR 寄存器的 UG='1'，复位定时器 2。
- 写‘0xE7’至定时器 2 的计数器(TIM2\_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。

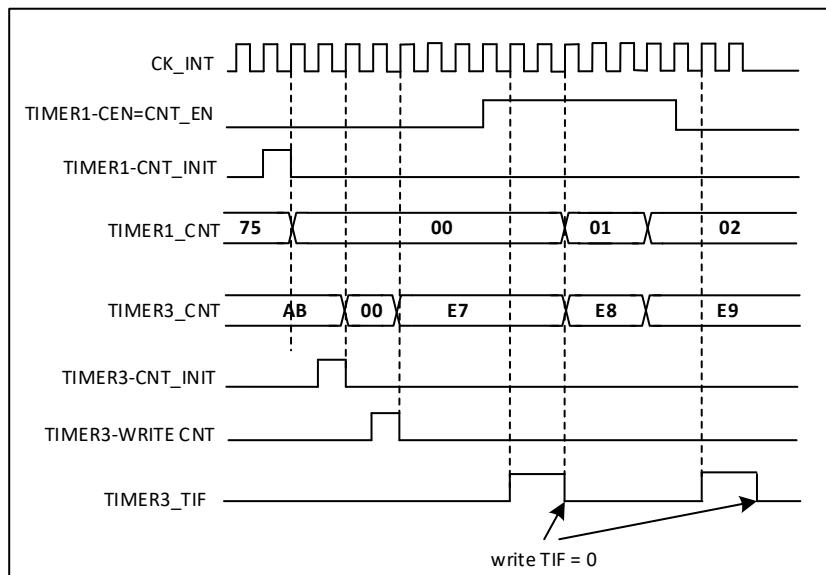


图 13-51 通过使能定时器 1 可以控制定时器 2

### 13.2.20.3. 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。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(TIM1\_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1\_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1。

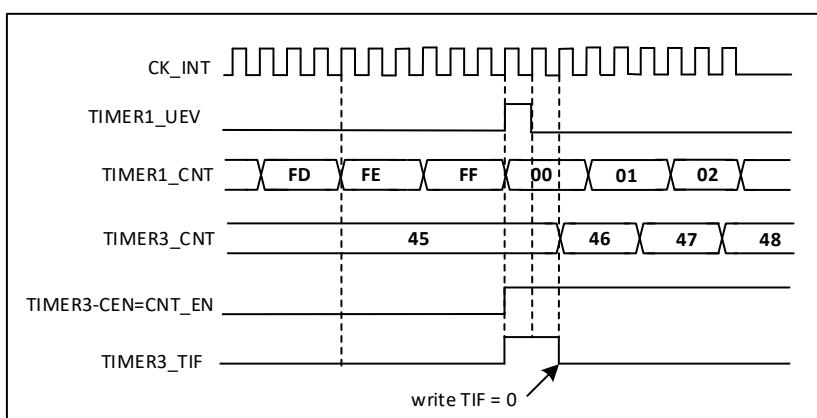


图 13-52 使用定时器 1 的更新触发定时器 2

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。

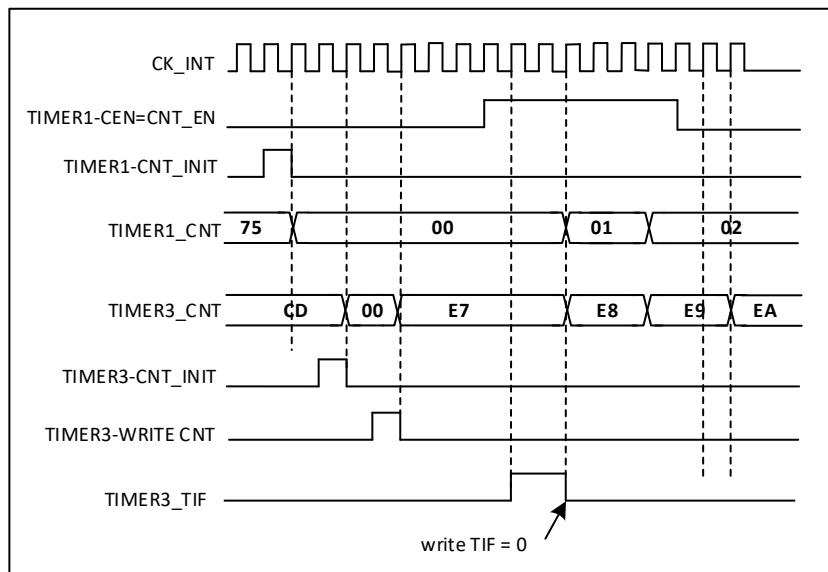


图 13-53 利用定时器 1 的使能触发定时器 2

#### 13.2.20.4. 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 4-47。保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1\_CR2 寄存器的 MMS=001)。
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1\_SMCR 寄存器的 TS=100)。
- 配置定时器 1 为触发模式(TIM1\_SMCR 寄存器的 SMS=110)。
- 配置定时器 1 为主/从模式， TIM1\_SMCR 寄存器的 MSM=1。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx\_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT\_EN 和 CK\_PSC 之间有个延迟。

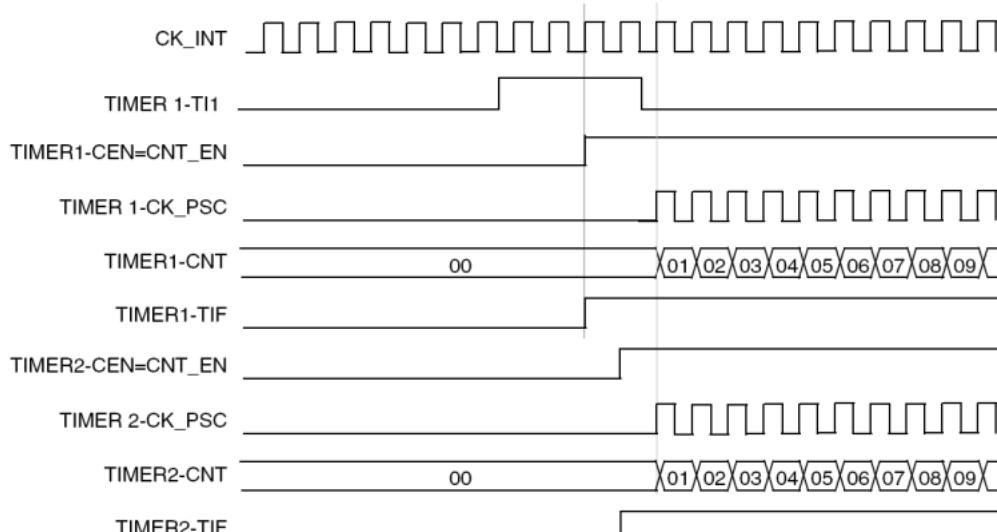


图 13-54 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

### 13.2.21. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止), 根据 DBG 模块中 DBG\_TIMx\_STOP 的设置, TIMx 计数器可以或者继续正常操作, 或者停止。详

## 13.3. 寄存器描述

TIM1 寄存器地址: 0x4001 2C00

TIM8 寄存器地址: 0x4001 3400

### 13.3.1. TIM1 和 TIM8 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
保留						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	保留, 始终为 0			
9:8	CKD	RW	0	<p>时钟分频因子 (Clock division)            这 2 位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟(tDTS)之间的分频比例。</p> <p>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</p>
7	ARPE	RW	0	<p>自动重装载预装载允许位 (Auto-reload preload enable)            0: TIMx_ARR 寄存器没有缓冲;            1: TIMx_ARR 寄存器被装入缓冲器。</p>
6:5	CMS	RW	0	<p>选择中央对齐模式 (Center-aligned mode selection)            00: 边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。            01: 中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 只在计数器向下计数时被设置。            10: 中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 只在计数器向上计数时被设置。            11: 中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 在计数器向上和向下计数时均被设置。            注: 在计数器开启时(CEN=1), 不允许从边沿对齐模式转换到中央对齐模式。</p>
4	DIR	RW	0	<p>方向 (Direction)            0: 计数器向上计数;            1: 计数器向下计数。            注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读。</p>
3	OPM	RW	0	<p>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。</p>
2	URS	RW	0	<p>更新请求源 (Update request source)            软件通过该位选择 UEV 事件的源            0: 如果使能了更新中断或 DMA 请求, 则下述任一事件产生更新中断或 DMA 请求:</p>

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> <p>1: 如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。</p>
1	UDIS	RW	0	<p>禁止更新 (Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新(UEV)事件由下述任一事件产生:</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。(译注: 更新影子寄存器) 1: 禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。</p>
0	CEN	RW	0	<p>使能计数器 (Counter enable) 0: 禁止计数器; 1: 使能计数器。 注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。</p>

### 13.3.2. TIM1 和 TIM8 控制寄存器 2 (TIMx\_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]	CCDS	CCUS	保留	CCPC		
保留	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	保留	保留	RW	RW

Bit	Name	R/W	Reset Value	Function
15	保留, 始终为 0			
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	<p>输出空闲状态 1(OC1N 输出) (Output Idle state 1) 0: 当 MOE=0 时, 死区后 OC1N=0; 1: 当 MOE=0 时, 死区后 OC1N=1。 注: 已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后, 该位不能被修改。</p>
8	OIS1	RW	0	<p>输出空闲状态 1(OC1 输出) (Output Idle state 1) 0: 当 MOE=0 时, 如果实现了 OC1N, 则死区后 OC1=0; 1: 当 MOE=0 时, 如果实现了 OC1N, 则死区后 OC1=1。 注: 已经设置了 LOCK(TIMx_BKR 寄存器)级别 1、2 或 3 后, 该位不能被修改。</p>
7	TI1S	RW	0	<p>TI1 选择 (TI1 selection) 0: TIMx_CH1 引脚连到 TI1 输入; 1: TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。</p>
6:4	MMS	RW	0	<p>主模式选择 (Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下: 000: 复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的复位(从模式控制器处于复位模式), 则 TRGO 上的信号相对实际的复位会有一个延迟。 001: 使能 – 计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时, TRGO 上会有一个延迟, 除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。</p>

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。 011: 比较脉冲 – 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时(即使它已经为高)，触发输出送出一个正脉冲(TRGO)。 100: 比较 – OC1REF 信号被用于作为触发输出(TRGO)。 101: 比较 – OC2REF 信号被用于作为触发输出(TRGO)。 110: 比较 – OC3REF 信号被用于作为触发输出(TRGO)。 111: 比较 – OC4REF 信号被用于作为触发输出(TRGO)。 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	CCDS	RW	0	捕获/比较的 DMA 选择 (Capture/compare DMA selection) 0: 当发生 CCx 事件时，送出 CCx 的 DMA 请求； 1: 当发生更新事件时，送出 CCx 的 DMA 请求。
2	CCUS	RW	0	捕获/比较控制更新选择 (Capture/compare control update selection) 0: 如果捕获/比较控制位是预装载的(CCPC=1)，只能通过设置 COM 位更新它们； 1: 如果捕获/比较控制位是预装载的(CCPC=1)，可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 注：该位只对具有互补输出的通道起作用。
1	保留，始终读为 0。			
0	CCPC	RW	0	捕获/比较预装载控制位 (Capture/compare preloaded control) 0: CCxE, CCxNE 和 OCxM 位不是预装载的； 1: CCxE, CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新。 注：该位只对具有互补输出的通道起作用。

### 13.3.3. TIM1 和 TIM8 从模式控制寄存器 (TIMx\_SMCR)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS [1:0]		ETF [3:0]		MSM		TS [2:0]		保留		SMS [2:0]			
RW	RW	RW		RW		RW		RW		保留		RW			

Bit	Name	R/W	Reset Value	Function
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	RW	0	外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 TIMxCLK 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 00: 关闭预分频； 01: ETRP 频率除以 2； 10: ETRP 频率除以 4； 11: ETRP 频率除以 8。
11:8	ETF	RW	0	外部触发滤波 (External trigger filter)

Bit	Name	R/W	Reset Value	Function								
				<p>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。</p> <p>0000: 无滤波器，以 fDTS 采样      0001: 采样频率 fSAMPLING=fCK_INT, N=2      0010: 采样频率 fSAMPLING=fCK_INT, N=4      0011: 采样频率 fSAMPLING=fCK_INT, N=8      0100: 采样频率 fSAMPLING=fDTS/2, N=6      0101: 采样频率 fSAMPLING=fDTS/2, N=8      0110: 采样频率 fSAMPLING=fDTS/4, N=6      0111: 采样频率 fSAMPLING=fDTS/4, N=8      1000: 采样频率 fSAMPLING=fDTS/8, N=6      1001: 采样频率 fSAMPLING=fDTS/8, N=8      1010: 采样频率 fSAMPLING=fDTS/16, N=5      1011: 采样频率 fSAMPLING=fDTS/16, N=6      1100: 采样频率 fSAMPLING=fDTS/16, N=8      1101: 采样频率 fSAMPLING=fDTS/32, N=5      1110: 采样频率 fSAMPLING=fDTS/32, N=6      1111: 采样频率 fSAMPLING=fDTS/32, N=8   </p>								
7	MSM	RW	0	<p>主/从模式 (Master/slave mode)</p> <p>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。</p>								
6:4	TS	RW	0	<p>触发选择 (Trigger selection)</p> <p>这 3 位选择用于同步计数器的触发输入。</p> <table> <tr> <td>000: 内部触发 0(ITR0)</td> <td>100: TI1 的边沿检测器(TI1F_ED)</td> </tr> <tr> <td>001: 内部触发 1(ITR1)</td> <td>101: 滤波后的定时器输入 1(TI1FP1)</td> </tr> <tr> <td>010: 内部触发 2(ITR2)</td> <td>110: 滤波后的定时器输入 2(TI2FP2)</td> </tr> <tr> <td>011: 内部触发 3(ITR3)</td> <td>111: 外部触发输入(ETRF)</td> </tr> </table> <p>更多有关 ITRx 的细节，参见表 5-1。      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。</p>	000: 内部触发 0(ITR0)	100: TI1 的边沿检测器(TI1F_ED)	001: 内部触发 1(ITR1)	101: 滤波后的定时器输入 1(TI1FP1)	010: 内部触发 2(ITR2)	110: 滤波后的定时器输入 2(TI2FP2)	011: 内部触发 3(ITR3)	111: 外部触发输入(ETRF)
000: 内部触发 0(ITR0)	100: TI1 的边沿检测器(TI1F_ED)											
001: 内部触发 1(ITR1)	101: 滤波后的定时器输入 1(TI1FP1)											
010: 内部触发 2(ITR2)	110: 滤波后的定时器输入 2(TI2FP2)											
011: 内部触发 3(ITR3)	111: 外部触发输入(ETRF)											
3	保留，始终读为 0。											
2:0	SMS	RW	0	<p>从模式选择 (Slave mode selection)</p> <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。      001: 编码器模式 1 – 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。      010: 编码器模式 2 – 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。      011: 编码器模式 3 – 根据另一个信号的输入电平，计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。      100: 复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。      101: 门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。      110: 触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位)，只有计数器的启动是受控的。      111: 外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。  <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。      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> </p>								

表 13-2 TIMx 内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM1	TIM5_TRGO	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO
TIM8	TIM1_TRGO	TIM2_TRGO	TIM4_TRGO	TIM5_TRGO

### 13.3.4. TIM1 和 TIM8 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
保留	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	保留, 始终为 0			
14	TDE	RW	0	允许触发 DMA 请求 (Trigger DMA request enable) 0: 禁止触发 DMA 请求; 1: 允许触发 DMA 请求。
13	COMDE	RW	0	允许 COM 的 DMA 请求 (COM DMA request enable) 0: 禁止 COM 的 DMA 请求; 1: 允许 COM 的 DMA 请求。
12	CC4DE	RW	0	允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0: 禁止捕获/比较 4 的 DMA 请求; 1: 允许捕获/比较 4 的 DMA 请求。
11	CC3DE	RW	0	允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0: 禁止捕获/比较 3 的 DMA 请求; 1: 允许捕获/比较 3 的 DMA 请求。。
10	CC2DE	RW	0	允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0: 禁止捕获/比较 2 的 DMA 请求; 1: 允许捕获/比较 2 的 DMA 请求。
9	CC1DE	RW	0	允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0: 禁止捕获/比较 1 的 DMA 请求; 1: 允许捕获/比较 1 的 DMA 请求。。
8	UDE	RW	0	允许更新的 DMA 请求 (Update DMA request enable) 0: 禁止更新的 DMA 请求; 1: 允许更新的 DMA 请求。
7	BIE	RW	0	允许刹车中断 (Break interrupt enable) 0: 禁止刹车中断; 1: 允许刹车中断。
6	TIE	RW	0	触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断; 1: 使能触发中断。
5	COMIE	RW	0	允许 COM 中断 (COM interrupt enable) 0: 禁止 COM 中断; 1: 允许 COM 中断。
4	CC4IE	RW	0	允许捕获/比较 4 中断 (Capture/Compare 4 interrupt enable) 0: 禁止捕获/比较 4 中断; 1: 允许捕获/比较 4 中断。
3	CC3IE	RW	0	允许捕获/比较 3 中断 (Capture/Compare 3 interrupt enable) 0: 禁止捕获/比较 3 中断; 1: 允许捕获/比较 3 中断。
2	CC2IE	RW	0	允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0: 禁止捕获/比较 2 中断; 1: 允许捕获/比较 2 中断。
1	CC1IE	RW	0	允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0: 禁止捕获/比较 1 中断; 1: 允许捕获/比较 1 中断
0	UIE	RW	0	允许更新中断 (Update interrupt enable)

Bit	Name	R/W	Reset Value	Function											
				0: 禁止更新中断; 1: 允许更新中断。											

### 13.3.5. TIM1 和 TIM8 状态寄存器 (TIMx\_SR)

Address offset:0x10

Reset value:0x0000

1 5	1 4	1 3	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4OF	CC3OF	CC2OF	CC1OF	保留	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	UIF	
保留	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function												
15:13	保留, 始终为 0															
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	保留, 始终读为 0.															
7	BIF	RC_W0	0	刹车中断标记 (Break interrupt flag) 一旦刹车输入有效, 由硬件对该位置'1'。如果刹车输入无效, 则该位可由软件清'0'。 0: 无刹车事件产生; 1: 刹车输入上检测到有效电平。												
6	TIF	RC_W0	0	触发器中断标记 (Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时, 在 TRGI 输入端检测到有效边沿, 或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0: 无触发器事件产生; 1: 触发中断等待响应。												
5	COMIF	RC_W0	0	COM 中断标记 (COM interrupt flag) 一旦产生 COM 事件(当捕获/比较控制位: CCxE、CCxNE、OCxM 已被更新)该位由硬件置'1'。它由软件清'0'。 0: 无 COM 事件产生; 1: COM 中断等待响应。												
4	CC4IF	RC_W0	0	捕获/比较 4 中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。												
3	CC3IF	RC_W0	0	捕获/比较 3 中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。												
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。												
1	CC1IF	RC_W0	0	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式: 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
				<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溢出, 或向下计数模式时的计数器下溢条件下, CC1IF 位变高</p> <p>如果通道 CC1 配置为输入模式:</p> <p>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;</p> <p>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)</p> <p>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 TIM1 和 TIM8 从模式控制寄存器(TIMx_SMCR))。</li> </ul>

### 13.3.6. TIM1 和 TIM8 事件产生寄存器 (TIMx\_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
保留								W	W	W	W	W	W	W	W
保留															

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
7	BG	W	0	<p>产生刹车事件 (Break generation)</p> <p>该位由软件置'1', 用于产生一个刹车事件, 由硬件自动清'0'。</p> <p>0: 无动作;</p> <p>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。</p>
6	TG	W	0	<p>产生触发事件 (Trigger generation)</p> <p>该位由软件置'1', 用于产生一个触发事件, 由硬件自动清'0'。</p> <p>0: 无动作;</p> <p>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。</p>
5	COMG	W	0	<p>捕获/比较事件, 产生控制更新 (Capture/Compare control update generation)</p> <p>该位由软件置'1', 由硬件自动清'0'。</p> <p>0: 无动作;</p> <p>1: 当 CCPC=1, 允许更新 CCxE、CCxNE、OCxM 位。</p> <p>注: 该位只对拥有互补输出的通道有效。</p>
4	CC4G	W	0	<p>产生捕获/比较 4 事件 (Capture/Compare 4 generation)</p> <p>参考 CC1G 描述。</p>
3	CC3G	W	0	<p>产生捕获/比较 3 事件 (Capture/Compare 3 generation)</p> <p>参考 CC1G 描述。</p>
2	CC2G	W	0	<p>产生捕获/比较 2 事件 (Capture/Compare 2 generation)</p> <p>参考 CC1G 描述。</p>
1	CC1G	W	0	<p>产生捕获/比较 1 事件 (Capture/Compare 1 generation)</p> <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>

Bit	Name	R/W	Reset Value	Function
				当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
0	UG	W	0	<p>产生更新事件 (Update generation) 该位由软件置'1'，由硬件自动清'0'。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'；若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。</p>

### 13.3.7. TIM1 和 TIM8 捕获/比较模式控制寄存器 1 (TIMx\_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	IC2F [3:0]			IC2PSC [1:0]				IC1F [3:0]	IC1PSC [1:0]						
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

#### 13.3.7.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15	OC2CE	RW	0	输出比较 2 清 0 使能 (Output Compare 2 clear enable)
14:12	OC2M	RW	0	输出比较 2 模式 (Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能 (Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能 (Output Compare 2 fast enable)
9:8	CC2S	RW	0	<p>捕获/比较 2 选择。 (Capture/Compare 2 selection) 该位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC2 通道被配置为输出； 01：CC2 通道被配置为输入， IC2 映射在 TI2 上； 10：CC2 通道被配置为输入， IC2 映射在 TI1 上； 11：CC2 通道被配置为输入， IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。</p>
7	OC1CE	RW	0	<p>输出比较 1 清 0 使能 (Output Compare 1 clear enable) 0：OC1REF 不受 ETRF 输入的影响； 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。</p>
6:4	OC1M	RW	0	<p>输出比较 1 模式 (Output Compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、 OC1N 的值。 OC1REF 是高电平有效，而 OC1、 OC1N 的有效电平取决于 CC1P、 CC1NP 位。 000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用； 001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。 010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。 011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 100：强制为无效电平。强制 OC1REF 为低。 101：强制为有效电平。强制 OC1REF 为高。 110：PWM 模式 1 - 在向上计数时，一旦 TIMx_CNT&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>

Bit	Name	R/W	Reset Value	Function
				<p>111: PWM 模式 2 - 在向上计数时, 一旦 <math>\text{TIMx\_CNT} &lt; \text{TIMx\_CCR1}</math> 时通道 1 为无效电平, 否则为有效电平; 在向下计数时, 一旦 <math>\text{TIMx\_CNT} &gt; \text{TIMx\_CCR1}</math> 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 (Output Compare 1 preload enable)</p> <p>0: 禁止 <math>\text{TIMx\_CCR1}</math> 寄存器的预装载功能, 可随时写入 <math>\text{TIMx\_CCR1}</math> 寄存器, 并且新写入的数值立即起作用。</p> <p>1: 开启 <math>\text{TIMx\_CCR1}</math> 寄存器的预装载功能, 读写操作仅对预装载寄存器操作, <math>\text{TIMx\_CCR1}</math> 的预装载值在更新事件到来时被加载至当前寄存器中。</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(<math>\text{TIMx\_CR1}</math> 寄存器的 OPM=1), 可以在未确认预装载寄存器情况下使用 PWM 模式, 否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较 1 快速使能 (Output Compare 1 fast enable)</p> <p>该位用于加快 CC 输出对触发输入事件的响应。</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 只在通道被配置成 PWM1 或 PWM2 模式时起作用。</p>
1:0	CC1S	RW	0	<p>捕获/比较 1 选择。 (Capture/Compare 1 selection)</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 <math>\text{TIMx\_SMCR}</math> 寄存器的 TS 位选择)。</p> <p>注: CC1S 仅在通道关闭时(<math>\text{TIMx\_CCER}</math> 寄存器的 CC1E=0)才是可写的。</p>

### 13.3.7.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	输入捕获 2 滤波器 (Input capture 2 filter)
11:10	IC2PSC	RW	0	输入捕获 2 预分频器 (Input capture 2 prescaler)
9:8	CC2S	RW	0	<p>捕获/比较 2 选择 (Capture/Compare 2 selection)</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 <math>\text{TIMx\_SMCR}</math> 寄存器的 TS 位选择)。</p> <p>注: CC2S 仅在通道关闭时(<math>\text{TIMx\_CCER}</math> 寄存器的 CC2E=0)才是可写的。</p>
7:4	IC1F	RW	0	<p>输入捕获 1 滤波器 (Input capture 1 filter)</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 <math>f_{DTS}</math> 采样 1000: 采样频率 <math>f_{SAMPLING}=f_{DTS}/8</math>, <math>N=6</math></p> <p>0001: 采样频率 <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N=2</math> 1001: 采样频率 <math>f_{SAMPLING}=f_{DTS}/8</math>, <math>N=8</math></p> <p>0010: 采样频率 <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N=4</math> 1010: 采样频率 <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N=5</math></p> <p>0011: 采样频率 <math>f_{SAMPLING}=f_{CK\_INT}</math>, <math>N=8</math> 1011: 采样频率 <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N=6</math></p> <p>0100: 采样频率 <math>f_{SAMPLING}=f_{DTS}/2</math>, <math>N=6</math> 1100: 采样频率 <math>f_{SAMPLING}=f_{DTS}/16</math>, <math>N=8</math></p> <p>0101: 采样频率 <math>f_{SAMPLING}=f_{DTS}/2</math>, <math>N=8</math> 1101: 采样频率 <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N=5</math></p> <p>0110: 采样频率 <math>f_{SAMPLING}=f_{DTS}/4</math>, <math>N=6</math> 1110: 采样频率 <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N=6</math></p> <p>0111: 采样频率 <math>f_{SAMPLING}=f_{DTS}/4</math>, <math>N=8</math> 1111: 采样频率 <math>f_{SAMPLING}=f_{DTS}/32</math>, <math>N=8</math></p>

Bit	Name	R/W	Reset Value	Function
3:2	IC1PSC	RW	0	<p>输入/捕获 1 预分频器 (Input capture 1 prescaler)            这 2 位定义了 CC1 输入(IC1)的预分频系数。            一旦 CC1E=0(TIMx_CCER 寄存器中), 则预分频器复位。</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;            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。</p>
1:0	CC1S	RW	0	<p>捕获/比较 1 选择 (Capture/Compare 1 Selection)            这 2 位定义通道的方向(输入/输出), 及输入脚的选择:            00: CC1 通道被配置为输出;            01: CC1 通道被配置为输入, IC1 映射在 TI1 上;            10: CC1 通道被配置为输入, IC1 映射在 TI2 上;            11: CC1 通道被配置为输入, IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。</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>

### 13.3.8. TIM1 和 TIM8 捕获/比较模式制寄存器 2 (TIMx\_CCMR2)

Address offset:0x1C

Reset value:0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M [2:0]			OC4PE	OC4FE [1:0]		CC4S [1:0]	OC3CE	OC3M [2:0]			OC3PE	OC3FE [1:0]		CC3S [1:0]
	C4F [3:0]			C4PSC [1:0]				OC3F [3:0]	C3PSC [1:0]			OC3F [3:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

#### 13.3.8.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15	OC4CE	RW	0	输出比较 4 清 0 使能 (Output Compare 4 clear enable)
14:12	OC4M	RW	0	输出比较 4 模式 (Output Compare 4 mode)
11	OC4PE	RW	0	输出比较 4 预装载使能 (Output Compare 4 preload enable)
10	OC4FE	RW	0	输出比较 4 快速使能 (Output Compare 4 fast enable)
9:8	CC4S	RW	0	<p>捕获/比较 4 选择。 (Capture/Compare 4 selection)            该 2 位定义通道的方向(输入/输出), 及输入脚的选择:            00: CC4 通道被配置为输出;            01: CC4 通道被配置为输入, IC4 映射在 TI4 上;            10: CC4 通道被配置为输入, IC4 映射在 TI3 上;            11: CC4 通道被配置为输入, IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。</p> <p>注: CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。</p>
7	OC3CE	RW	0	输出比较 3 清 0 使能 (Output Compare 1clear enable)
6:4	OC3M	RW	0	输出比较 3 模式 (Output Compare 3 mode)
3	OC3PE	RW	0	输出比较 3 预装载使能 (Output Compare 3 preload enable)
2	OC3FE	RW	0	输出比较 3 快速使能 (Output Compare 3 fast enable)
1:0	CC3S	RW	0	<p>捕获/比较 3 选择。 (Capture/Compare 3 selection)            该 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;</p>

Bit	Name	R/W	Reset Value	Function
				11: CC3 通道被配置为输入, IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

### 13.3.8.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:12	IC4F	RW	0	输入捕获 4 滤波器 (Input capture 4 filter)
11:10	IC4PSC	RW	0	输入/捕获 4 预分频器 (Input capture 4 prescaler)
9:8	CC4S	RW	0	捕获/比较 4 选择 (Capture/Compare 4 selection) 这 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC4 通道被配置为输出; 01: CC4 通道被配置为输入, IC4 映射在 TI4 上; 10: CC4 通道被配置为输入, IC4 映射在 TI3 上; 11: CC4 通道被配置为输入, IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7:4	IC3F	RW	0	输入捕获 3 滤波器 (Input capture 3 filter)
3:2	IC3PSC	RW	0	输入/捕获 3 预分频器 (Input capture 3 prescaler)
1:0	CC3S	RW	0	捕获/比较 3 选择 (Capture/Compare 3 Selection) 这 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC3 通道被配置为输出; 01: CC3 通道被配置为输入, IC3 映射在 TI3 上; 10: CC3 通道被配置为输入, IC3 映射在 TI4 上; 11: CC3 通道被配置为输入, IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

### 13.3.9. TIM1 和 TIM8 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E	
保留	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:14	保留, 始终为 0			
13	CC4P	RW	0	输入/捕获 4 输出极性 (Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	RW	0	输入/捕获 4 输出使能 (Capture/Compare 4 output enable) 参考 CC1E 的描述。
11	CC3NP	RW	0	输入/捕获 3 互补输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。
10	CC3NE	RW	0	输入/捕获 3 互补输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
9	CC3P	RW	0	输入/捕获 3 输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	CC3E	RW	0	输入/捕获 3 输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
7	CC2NP	RW	0	输入/捕获 2 互补输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。

Bit	Name	R/W	Reset Value	Function
6	CC2NE	RW	0	输入/捕获 2 互补输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
5	CC2P	RW	0	输入/捕获 2 输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
3	CC1NP	RW	0	输入/捕获 1 互补输出极性 (Capture/Compare 1 complementary output polarity) 0: OC1N 高电平有效; 1: OC1N 低电平有效。 注: 一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2 且 CC1S=00(通道配置为输出)则该位不能被修改。
2	CC1NE	RW	0	输入/捕获 1 互补输出使能 (Capture/Compare 1 complementary output enable) 0: 关闭 - OC1N 禁止输出, 因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 1: 开启 - OC1N 信号输出到对应的输出引脚, 其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。
1	CC1P	RW	0	输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) CC1 通道配置为输出: 0: OC1 高电平有效; 1: OC1 低电平有效。 CC1 通道配置为输入: 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0: 不反相: 捕获发生在 IC1 的上升沿; 当用作外部触发器时, IC1 不反相。 1: 反相: 捕获发生在 IC1 的下降沿; 当用作外部触发器时, IC1 反相。 注: 一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 3 或 2, 则该位不能被修改。
0	CC1E	RW	0	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出: 0: 关闭 - OC1 禁止输出, 因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 1: 开启 - OC1 信号输出到对应的输出引脚, 其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 CC1 通道配置为输入: 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0: 捕获禁止; 1: 捕获使能。

表 13-3 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

控制位					输出状态	
MOE 位	OSSI 位	OSSR 位	CCxE 位	CCxNE 位	OCx 输出状态	OCxN 输出状态
1	x	0	0	0	输出禁止 (与定时器断开) OCx=0, OCx_EN=0	输出禁止 (与定时器断开) OCxN=0, OCxN_EN=0
		0	0	1	输出禁止 (与定时器断开) OCx=0, OCx_EN=0	OCxREF+极性, OCxN=OCREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF+极性, OCx=OCREF xor CCxP, OCx_EN=1	输出禁止 (与定时器断开) OCxN=0, OCxN_EN=0
		0	1	1	OCxREF+极性+死区,, OCx_EN=1	OCxREF+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

控制位					输出状态											
		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	0	x	0	0	输出禁止 (与定时器断开) 异步的: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCx_EN=0; 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											
	0		0	1												
	0		1	0												
	0		1	1												
	1		0	0												
	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。

### 13.3.10. TIM1 和 TIM8 计数器 (TIMx\_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	计数器的值 (Counter value)

### 13.3.11. TIM1 和 TIM8 预分频器 (TIMx\_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	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)。 PSC 包含了每次当更新事件产生时, 装入当前预分频器寄存器的值; 更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

### 13.3.12. TIM1 和 TIM8 自动重装载寄存器 (TIMx\_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

ARR
RW

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 246/754 详细参考 13.3.1 节：有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

### 13.3.13. TIM1 和 TIM8 重复计数寄存器 (TIMx\_RCR)

Address offset:0x30

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终读为 0。			
7:0	REP	RW	0	重复计数器的值 (Repetition counter value) 开启了预装载功能后，这些位允许用户设置比较寄存器的更新速率(即周期性地从预装载寄存器传输到当前寄存器)；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器 REP_CNT 达到 0，会产生一个更新事件并且计数器 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，(REP+1)对应着： - 在边沿对齐模式下，PWM 周期的数目； - 在中心对称模式下，PWM 半周期的数目；

### 13.3.14. TIM1 和 TIM8 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW/RO	0	捕获/比较通道 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。 否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

### 13.3.15. TIM1 和 TIM8 捕获/比较寄存器 2 (TIMx\_CCR2)

Address offset:0x38

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW/RO	0	<p>捕获/比较通道 1 的值 (Capture/Compare 1 value)            若 CC2 通道配置为输出：            CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。            如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。            否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。            当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。            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>

### 13.3.16. TIM1 和 TIM8 捕获/比较寄存器 3 (TIMx\_CCR3)

Address offset:0x3C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR3	RW/RO	0	<p>捕获/比较通道 3 的值 (Capture/Compare 3 value)            若 CC3 通道配置为输出：            CCR3 包含了装入当前捕获/比较 3 寄存器的值(预装载值)。            如果在 TIMx_CCMR3 寄存器(OC3PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。            否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。            当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。            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>

### 13.3.17. TIM1 和 TIM8 捕获/比较寄存器 4 (TIMx\_CCR4)

Address offset:0x40

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR4	RW/RO	0	<p>捕获/比较通道 4 的值 (Capture/Compare 4 value)            若 CC4 通道配置为输出：            CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。            如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。            否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。            当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。            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>

### 13.3.18. TIM1 和 TIM8 刹车和死区寄存器 (TIMx\_BDTR)

Address offset:0x44

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK	[1:0]							DTG	[7:0]
RW	RW	RW	RW	RW	RW	RW	RW							RW	

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 (Main output enable) 一旦刹车输入有效，该位被硬件异步清'0'。根据 AOE 位的设置值，该位可以由软件清'0'或被自动置 1。它仅对配置为输出的通道有效。 0: 禁止 OC 和 OCN 输出或强制为空闲状态； 1: 如果设置了相应的使能位(TIMx_CCER 寄存器的 CCxE、CCxNE 位)，则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER)。
14	AOE	RW	0	自动输出使能 (Automatic output enable) 0: MOE 只能被软件置'1'； 1: MOE 能被软件置'1'或在下一个更新事件被自动置'1'(如果刹车输入无效)。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。
13	BKP	RW	0	刹车输入极性 (Break polarity) 0: 刹车输入低电平有效； 1: 刹车输入高电平有效。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为'1'，则该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
12	BKE	RW	0	刹车功能使能 (Break enable) 0: 禁止刹车输入(BRK 及 CCS 时钟失效事件)； 1: 开启刹车输入(BRK 及 CCS 时钟失效事件)。 注：当设置了 LOCK 级别 1 时(TIMx_BDTR 寄存器中的 LOCK 位)，该位不能被修改。 注：任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
11	OSSR	RW	0	运行模式下“关闭状态”选择 (Off-state selection for Run mode) 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 参考 OC/OCN 使能的详细说明(TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER))。 0: 当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1: 当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，首先开启 OC/OCN 并输出无效电平，然后置 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
10	OSSI	RW	0	空闲模式下“关闭状态”选择 (Off-state selection for Idle mode) 该位用于当 MOE=0 且通道设为输出时。 参考 OC/OCN 使能的详细说明(TIM1 和 TIM8 捕获/比较使能寄存器(TIMx_CCER))。 0: 当定时器不工作时，禁止 OC/OCN 输出(OC/OCN 使能输出信号=0)； 1: 当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号=1。 注：一旦 LOCK 级别(TIMx_BDTR 寄存器中的 LOCK 位)设为 2，则该位不能被修改。
9:8	LOCK	RW	0	锁定设置 (Lock configuration) 该位为防止软件错误而提供写保护。 00: 锁定关闭，寄存器无写保护； 01: 锁定级别 1，不能写入 TIMx_BDTR 寄存器的 DTG、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 位； 10: 锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位(一旦相关通道通过 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位)以及 OSSR/OSSI 位； 11: 锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位(一旦相关通道通过 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位)； 注：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。
7:0	DTG	RW	0	死区发生器设置 (Dead-time generator setup)

Bit	Name	R/W	Reset Value	Function
				<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> <p>0 到 15875ns，若步长时间为 125ns；</p> <p>16us 到 31750ns，若步长时间为 250ns；</p> <p>32us 到 63us，若步长时间为 1us；</p> <p>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，则不能修改这些位。</p>

### 13.3.19. TIM1 和 TIM8 DMA 控制寄存器 (TIMx\_DCR)

Address offset:0x48

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:13	保留，始终为 0			
12:8	DBL	RW	0	<p>DMA 连续传送长度 (DMA burst length)</p> <p>这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节：</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：  <math>(\text{TIMx\_CR1 的地址}) + \text{DBA} + (\text{DMA 索引})</math>, 其中 DMA 索引 = DBL</p> <p>其中(<math>\text{TIMx\_CR1 的地址}</math>) + 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(<math>\text{TIMx\_CR1 的地址}</math>) + 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> <ul style="list-style-type: none"> <li>- 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。</li> <li>- 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。</li> </ul>
7:5	保留，始终读为 0。			
4:0	DBA	RW	0	<p>DMA 基址 (DMA base address)</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>

### 13.3.20. TIM1 和 TIM8 连续模式的 DMA 地址 (TIMx\_DMAR)

Address offset:0x4C

Reset value:0x0000

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

DMAB  
RW

Bit	Name	R/W	Reset Value	Function
15:0	DMAB	RW	0	DMA 连续传送寄存器 (DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： TIMx_CR1 地址 + DBA + DMA 索引，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基地址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

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。

### 13.3.21. TIM1 和 TIM8 寄存器映射

		Offset	Register																	
		Re-set Value		0																
0x10	TIMx_SR			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
	Read/Write		Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	TIMx_EGR			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Read/Write		Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TIMx_CC MR1			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Read/Write		Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CC MR2			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Read/Write		Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CC ER			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Read/Write		Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Value																																	
0x24	TIMx_CNT	Reserved																												CNT				
	Read/Write																													rw				
0x28	Reset Value	0																															0	
	TIMx_PSC	Reserved																														PSC		
	Read/Write																															rw		
0x2C	Reset Value	0																														0		
	TIMx_ARR	Reserved																														ARR		
	Read/Write																															rw		
0x30	Reset Value	0																															0xFFFF	
	TIMx_RCR	Reserved																															REP	
	Read/Write	rw																																
0x34	Reset Value	0																														0		
	TIMx_CC_R1	Reserved																														CCR1		
	Read/Write																															rw/ro		
0x38	Reset Value	0																														0		
	TIMx_CC_R2	Reserved																														CCR2		
	Read/Write																															rw/ro		
0x38	Reset	0																														0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	Value																																			
0x3C	TIMx_CC_R3	Reserved														CCR3																				
0x3C	Read/Write															rw/ro																				
0x40	Reset Value	Reserved																														0				
0x40	TIMx_CC_R4															CCR4																				
0x40	Read/Write															rw/ro																				
0x44	Reset Value	Reserved																															0			
0x44	TIMx_BDTR															DTG																				
0x44	Read/Write															rw																				
0x48	Reset Value	Reserved																																0		
0x48	TIMx_DC_R															DBL																				
0x48	Read/Write															rw																				
0x4C	Reset Value	Reserved																																	0	
0x4C	TIMx_DMAR															DMAB																				
0x4C	Read/Write															rw																				
0x4C	Reset Value	Reserved																																		0

# 14. 通用定时器 (TIM2到TIM5)

## 14.1. 简介

通用定时器(TIM2/3/4/5)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较和 PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

每个通用控制定时器(TIM2/3/4/5)是完全独立的，它们不共享任何资源。它们可以同步操作。

### 14.1.1. TIM2/3/4/5 主要特征

TIM2/3/4/5 定时器的功能包括：

- 16 位向上、向下、向上/下的自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 多达 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 单脉冲模式输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

### 14.1.2. 模块框图

Figure 100. General-purpose timer block diagram

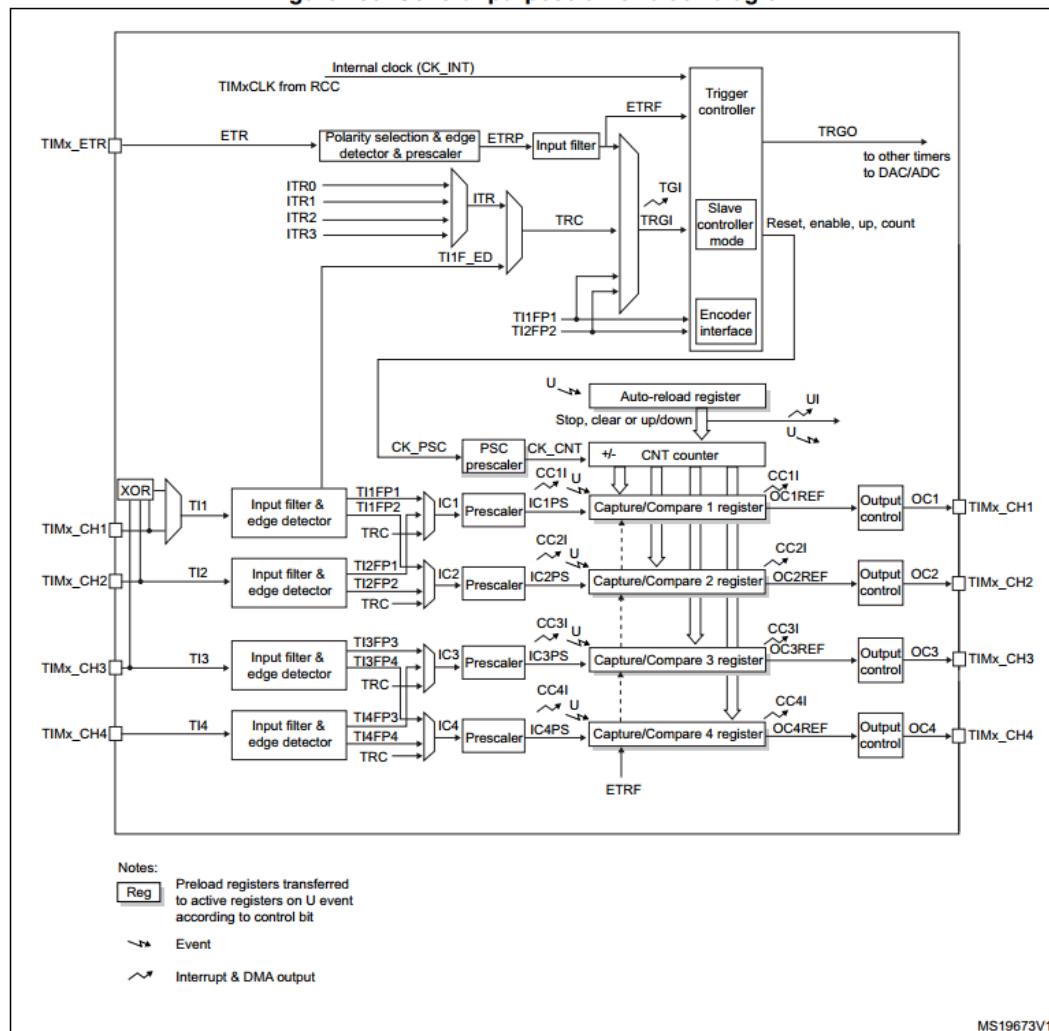


图 14-1 TIM2/3/4/5 模块

## 14.2. TIM2/3/4/5 功能描述

### 14.2.1. 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装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。计数器的时钟可以被预分频器分频。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)
- 自动装载寄存器 (TIMx\_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问它的预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 (UEV) 时传送到影子寄存器。当计数器达到溢出条件(上溢或者下溢)并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK\_CNT 驱动，仅当设置了 TIMx\_CR1 寄存器中的计数器使能位(CEN)，CK\_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx\_CR1 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

#### 14.2.1.1. 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx\_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频参数将在下一次更新事件到来时被采用。

下图给出了在预分频器运行时，更改计数器参数的例子。

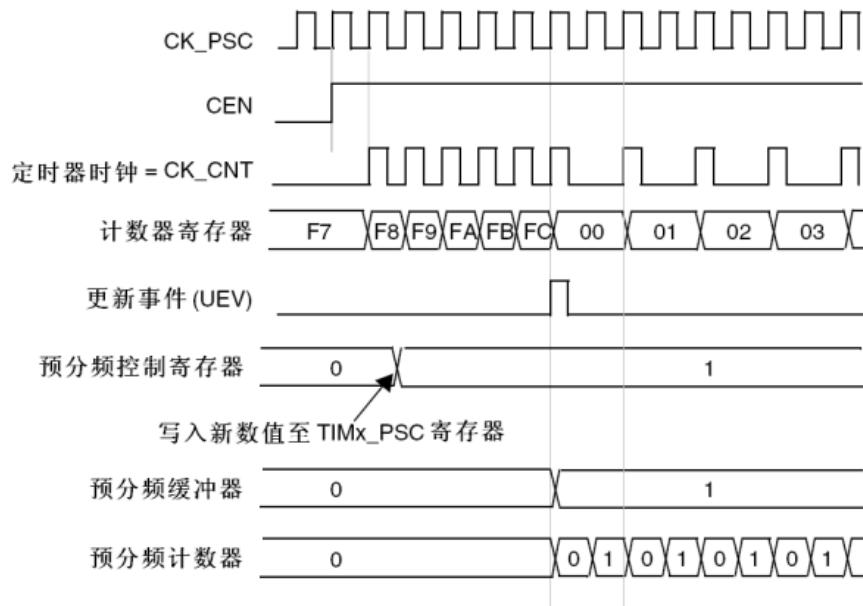


图 14-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

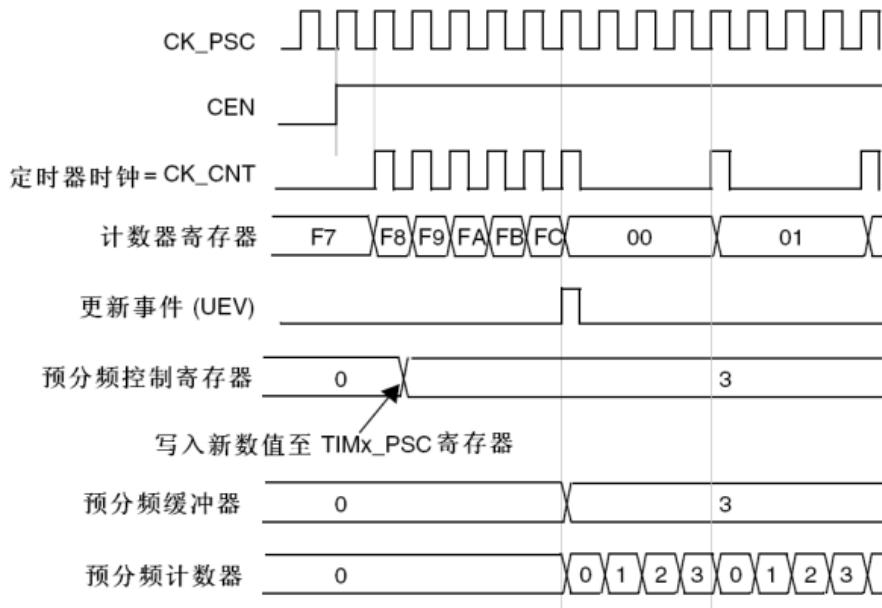


图 14-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

## 14.2.2. 计数器模式

### 14.2.2.1. 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx\_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

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；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断或 DMA 请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx\_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx\_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx\_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx\_ARR=0x36 时计数器在不同时钟频率下的动作。

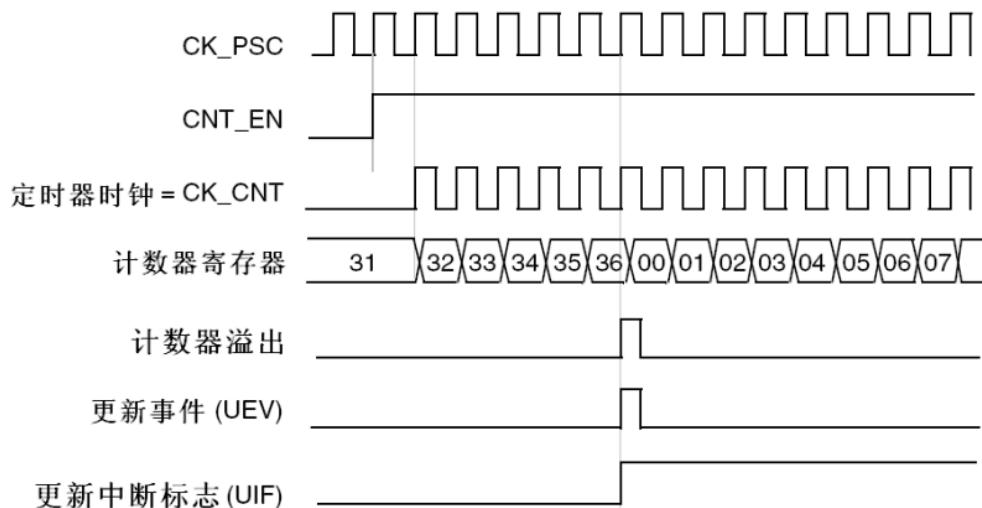


图 14-4 计数器时序图，内部时钟分频因子为 1

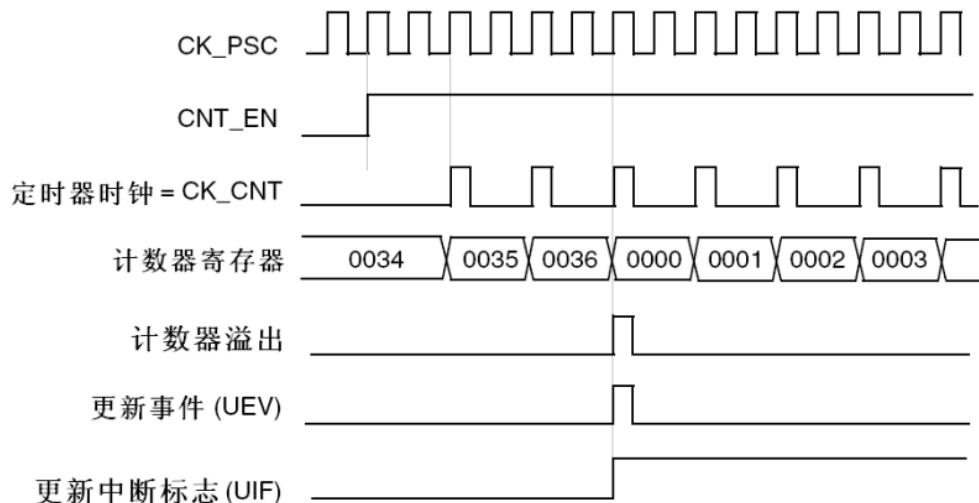


图 14-5 计数器时序图，内部时钟分频因子为 2

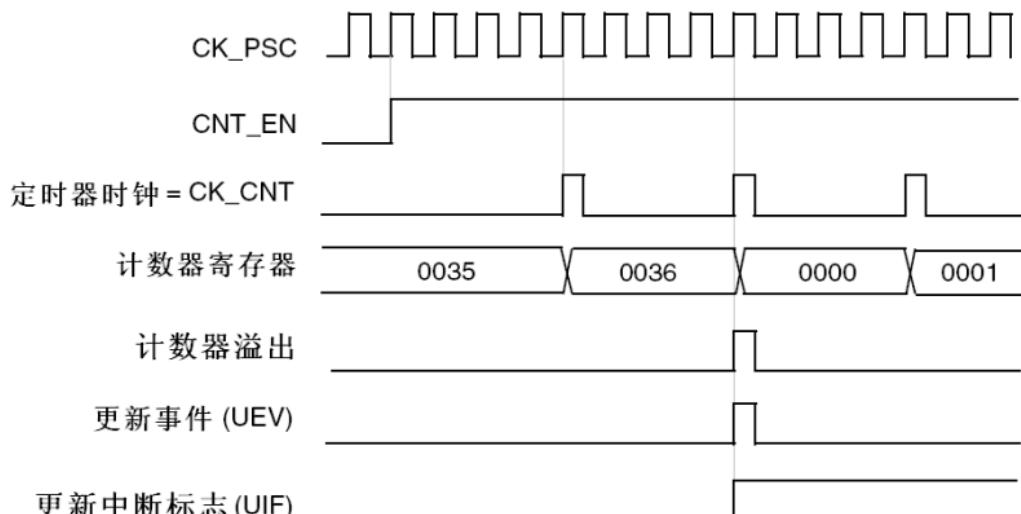


图 14-6 计数器时序图，内部时钟分频因子为 4

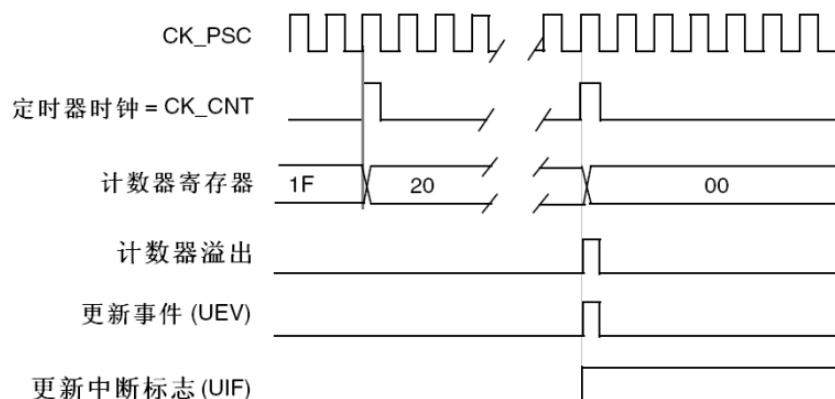


图 14-7 计数器时序图，内部时钟分频因子为 N

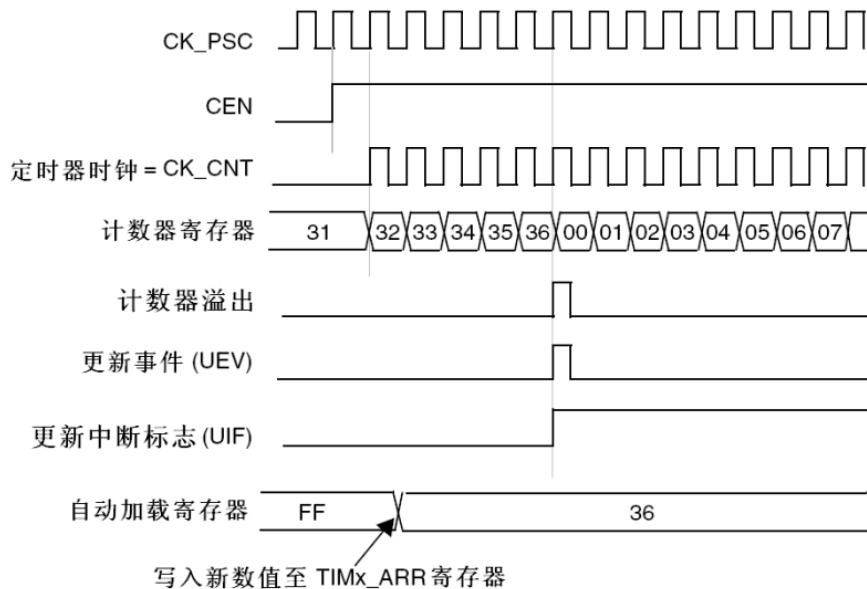


图 14-8 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx\_ARR)

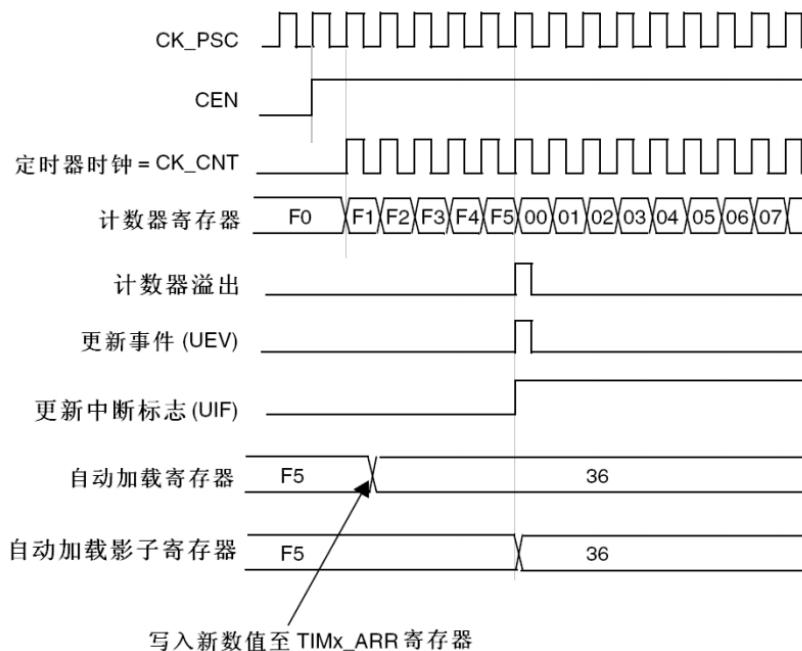


图 14-9 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx\_ARR)

#### 14.2.2.2 向下计数模式

在向下计数模式中，计数器从自动加载值(TIMx\_ARR 的内容)开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数下溢事件。

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。

通过设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会从当前自动加载值重新开始计数，同时预分频器内部的计数器被清‘0’(但预分频系数不变)。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不置起 UIF 标志位(即不会产生中断或 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx\_SR 寄存器中的 UIF 位)：

- 预分频器的缓冲区被置入预装载寄存器的值(TIMx\_PSC 寄存器的内容)。
- 当前的自动加载寄存器被更新为预装载值(TIMx\_ARR 寄存器中的内容)。

注：自动加载值在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx\_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

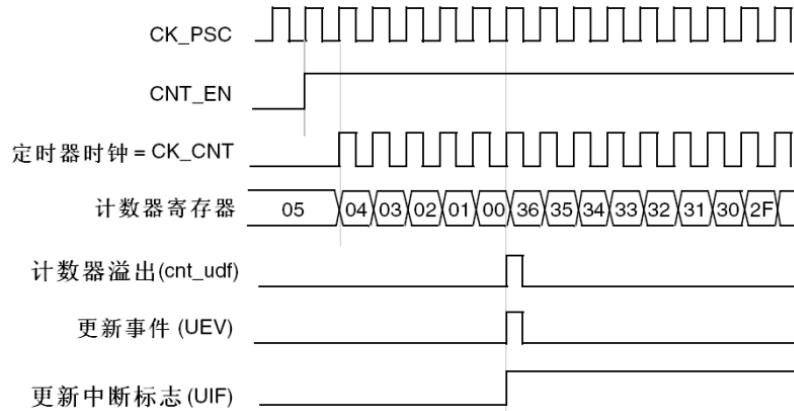


图 14-10 计数器时序图，内部时钟分频因子为 1

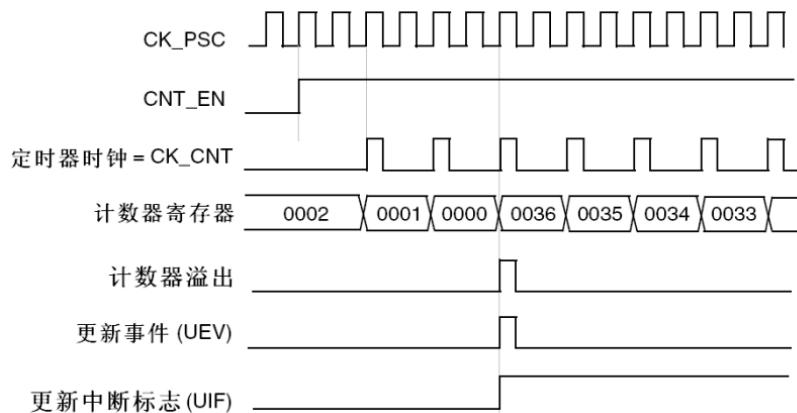


图 14-11 计数器时序图，内部时钟分频因子为 2

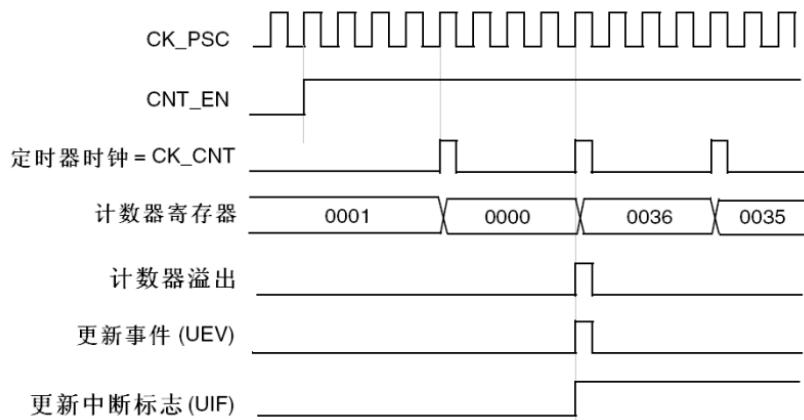


图 14-12 计数器时序图，内部时钟分频因子为 4

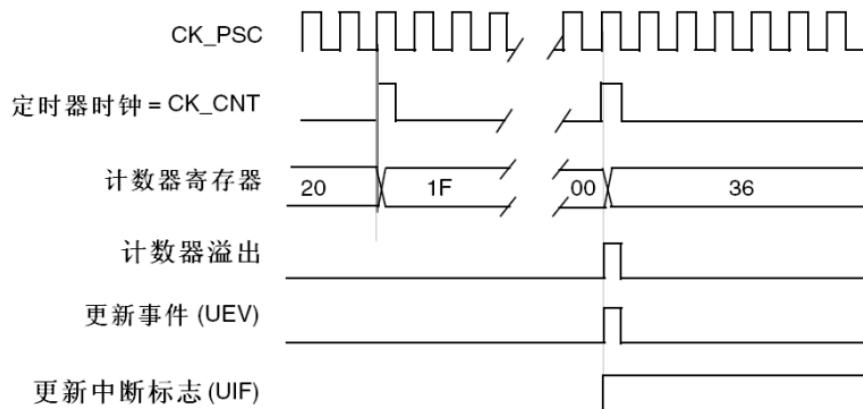


图 14-13 计数器时序图，内部时钟分频因子为 N

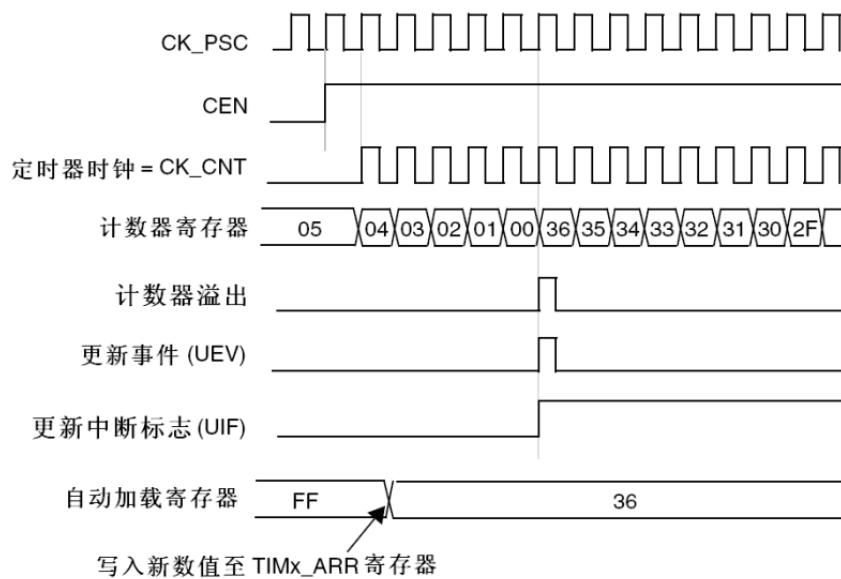


图 14-14 计数器时序图，当没有使用重复计数器时的更新事件

### 14.2.2.3. 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。

通过配置 TIMx\_CR1 寄存器中的 CMS 位不为‘00’可以得到中央对齐模式。在通道配置为输出模式下的输出比较标志位在以下几种计数过程中会被置起：向下计数时（中央对齐模式 1，CMS=‘01’）、向上计数时（中央对齐模式 2，CMS=‘10’）、向上和向下计数时（中央对齐模式 3，CMS=‘11’）。

在此模式下，不能写入 TIMx\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过(软件或者使用从模式控制器)设置 TIMx\_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器内部计数器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止更新事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。虽然 UDIS 位被清为 0 之前不会产生更新事件。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(选择更新请求源)，通过设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志(因此不产生中断和 DMA 请求)，这是为了避免在发生捕获事件时清除计数器，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且(根据 URS 位的设置)更新标志位(TIMx\_SR 寄存器中的 UIF 位)也被设置：

- 预分频器的缓存器被加载为预装载(TIMx\_PSC 寄存器)的值。
- 当前的自动加载寄存器被更新为预装载值(TIMx\_ARR 寄存器中的内容)。注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值(计数器被装载为新的值)。

以下是一些计数器在不同时钟频率下的操作的例子：

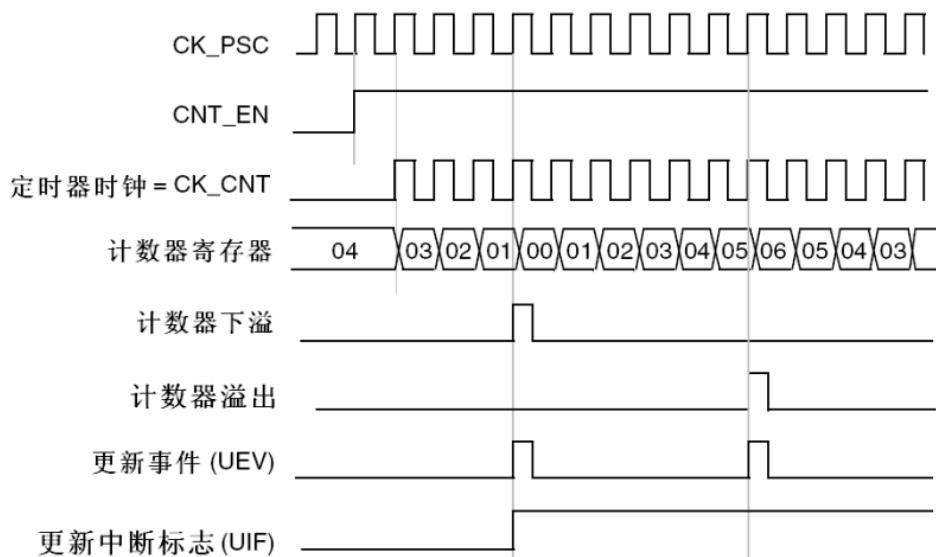


图 14-15 计数器时序图，内部时钟分频因子为 1， TIMx\_ARR=0x6

这里使用了中心对齐模式 1。

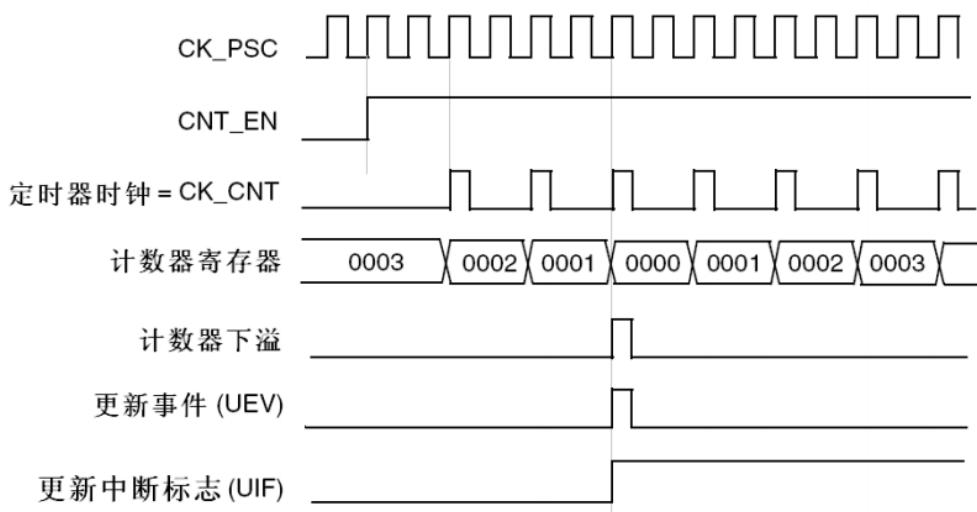
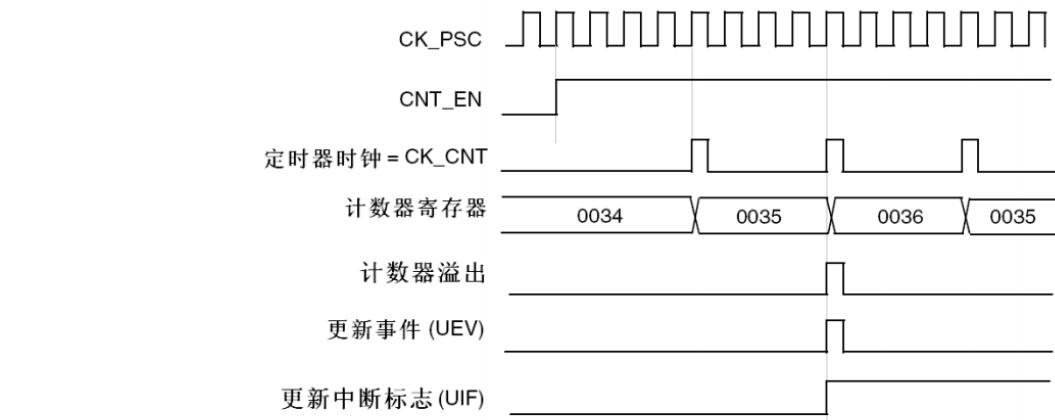


图 14-16 计数器时序图，内部时钟分频因子为 2



注：中心对齐模式2或3是在溢出时与 UIF 标志一起使用

图 14-17 计数器时序图，内部时钟分频因子为 4， $\text{TIMx\_ARR}=0x36$

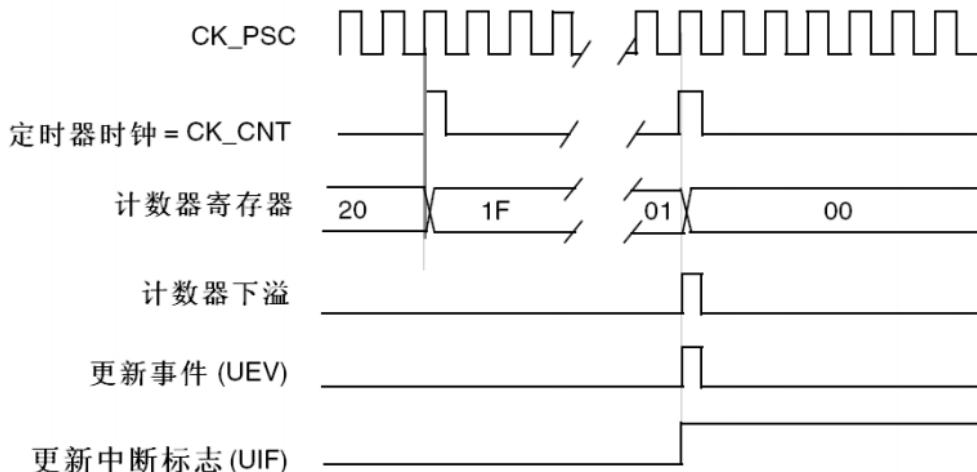


图 14-18 计数器时序图，内部时钟分频因子为  $N$

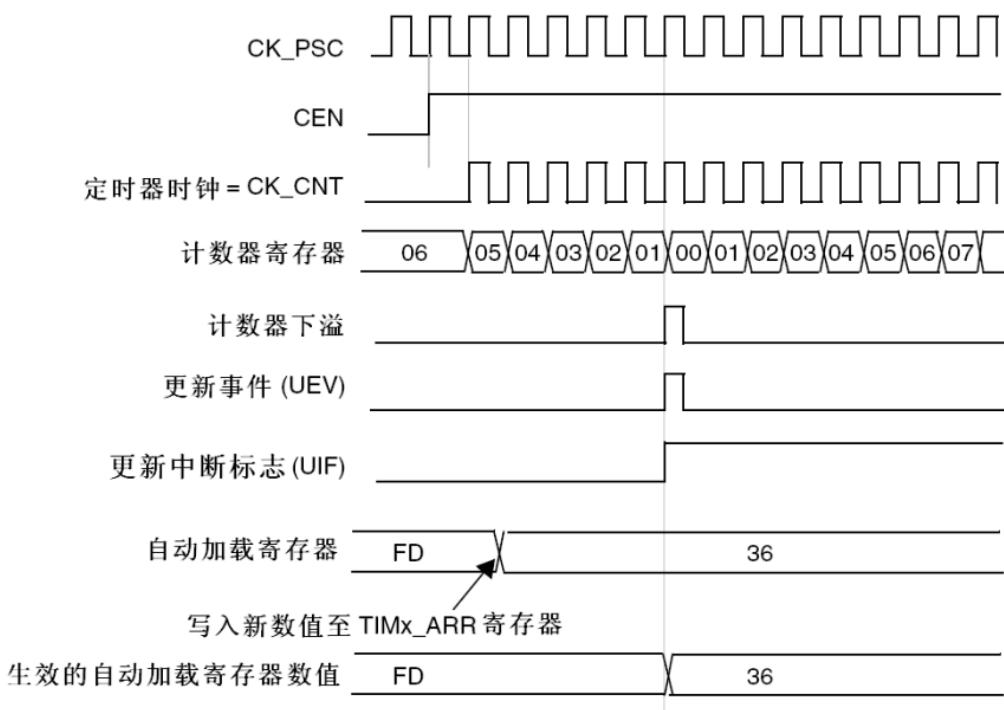


图 14-19 计数器时序图， $\text{ARPE}=1$  时的更新事件(计数器下溢)

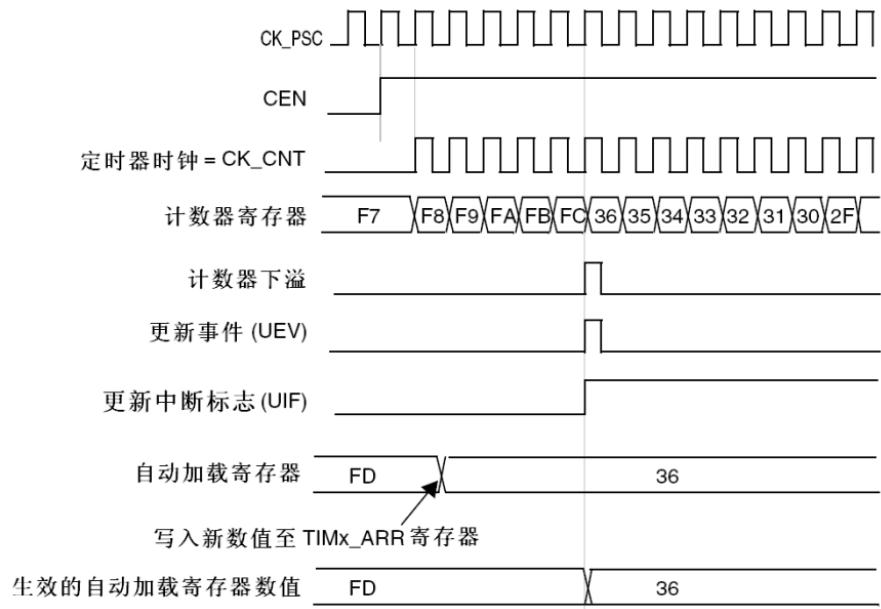


图 14-20 计数器时序图, ARPE=1 时的更新事件(计数器溢出)

### 14.2.3. 时钟选择

计数器时钟可由下列时钟源提供：

内部时钟(CK\_INT)

- 外部时钟模式 1：外部输入引脚 TI<sub>x</sub>
- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

#### 14.2.3.1. 内部时钟源(CK\_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN、 DIR(TIMx\_CR1 寄存器)和 UG 位(TIMx\_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成‘1’，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

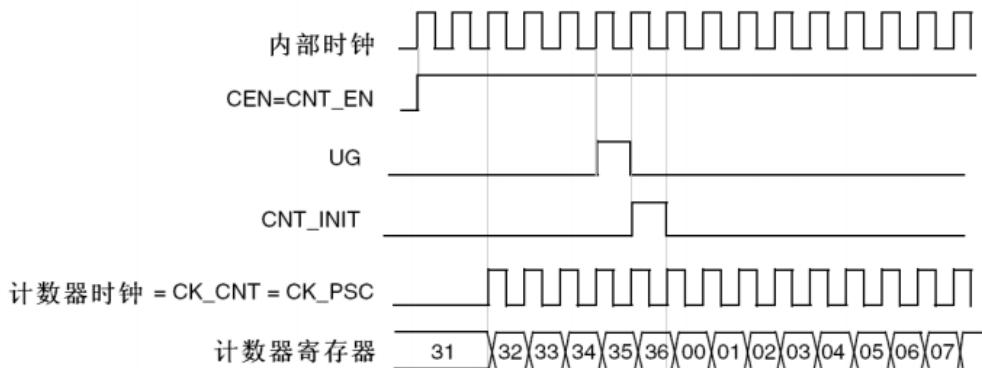


图 14-21 一般模式下的控制电路，内部时钟分频因子为 1

#### 14.2.3.2. 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。

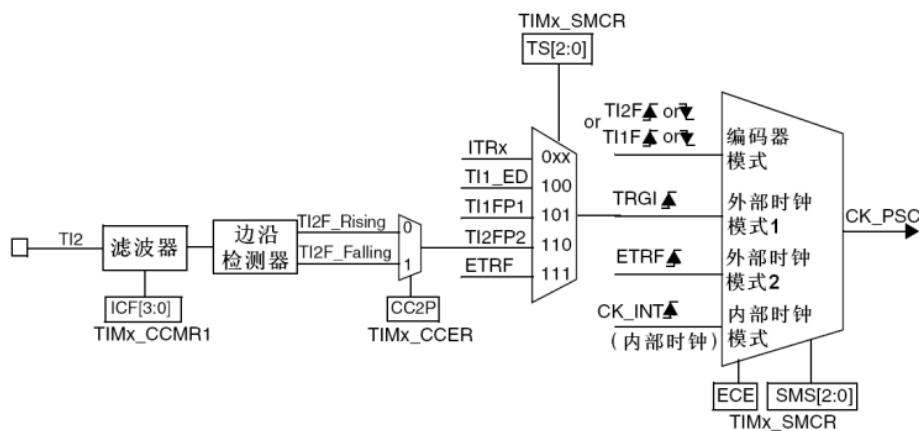


图 14-22 TI2 外部时钟连接例子

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：

- 配置 TIMx\_CCMR1 寄存器 CC2S=01，使得通道 2 检测 TI2 输入端的上升沿；
- 配置 TIMx\_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽 (如果不需要滤波器，保持 IC2F=0000)；
- 配置 TIMx\_CCER 寄存器的 CC2P=0，选定上升沿极性；
- 配置 TIMx\_SMCR 寄存器的 SMS=111，选择定时器为外部时钟模式 1；
- 配置 TIMx\_SMCR 寄存器中的 TS=110，选定 TI2 作为触发输入源；
- 设置 TIMx\_CR1 寄存器的 CEN=1，启动计数器。

注：捕获预分频器不用作触发，所以不需要对它进行配置

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

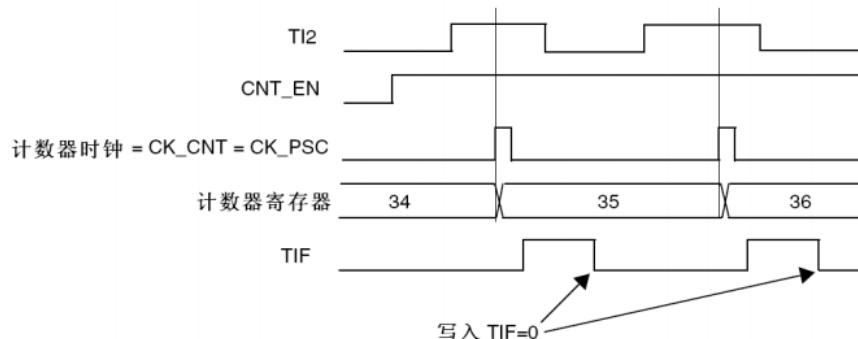


图 14-23 外部时钟模式 1 下的控制电路

#### 14.2.3.3. 外部时钟源模式 2

选定此模式的方法为：令 TIMx\_SMCR 寄存器中的 ECE=1，计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图：

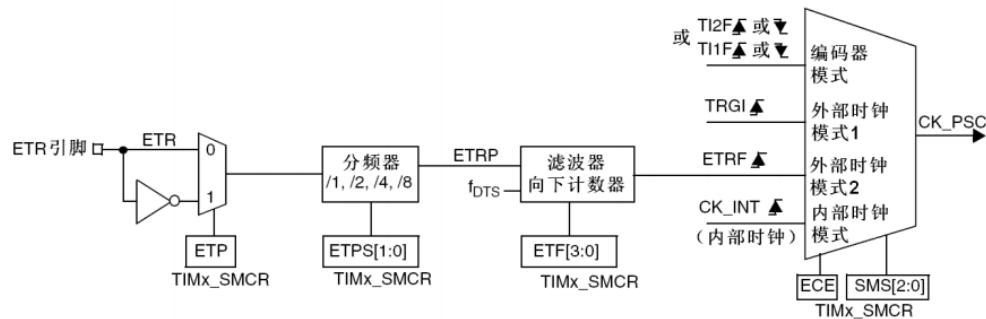


图 14-24 外部触发输入框图

例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

- 本例中不需要滤波器，置 TIMx\_SMCR 寄存器中的 ETF[3:0]=0000；
- 设置预分频器，置 TIMx\_SMCR 寄存器中的 ETPS[1:0]=01；
- 选择 ETR 输入端的上升沿，置 TIMx\_SMCR 寄存器中的 ETP=0；
- 开启外部时钟模式 2，写 TIMx\_SMCR 寄存器中的 ECE=1；
- 启动计数器，写 TIMx\_CR1 寄存器中的 CEN=1；
- 计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于 ETRP 信号的重新同步电路。

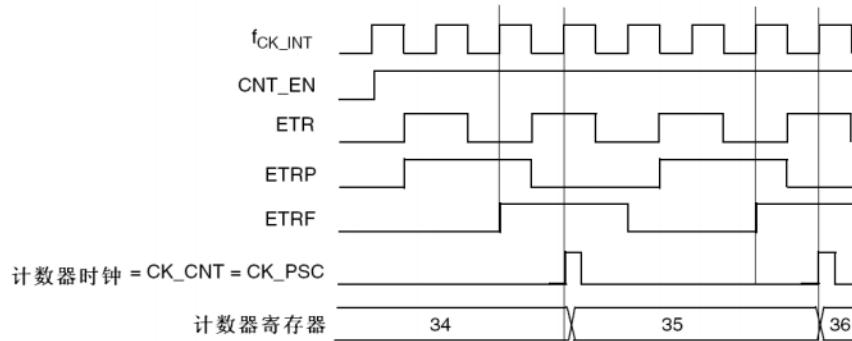


图 14-25 外部时钟模式 2 下的控制电路

#### 14.2.4. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。

输入部分对相应的 TI<sub>x</sub> 输入信号采样，并产生一个滤波后的信号 TI<sub>x</sub>F。然后，一个带极性选择的边缘监测器产生一个信号(TIxFP<sub>x</sub>)，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)。在捕获寄存器前分频得到 ICxPS。

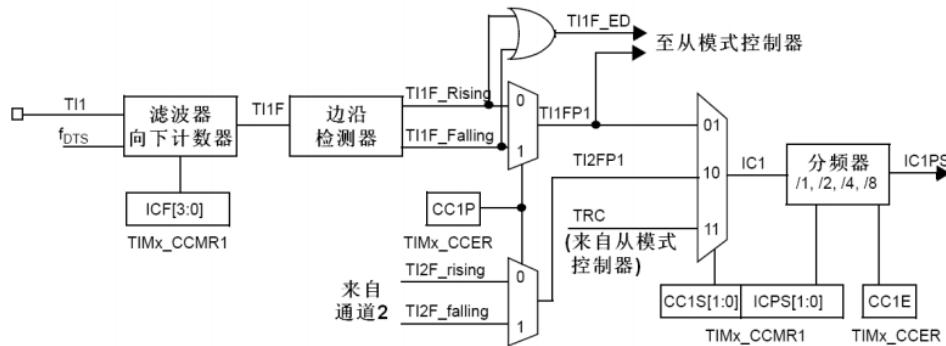


图 14-26 捕获/比较通道(如：通道 1 输入部分)

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

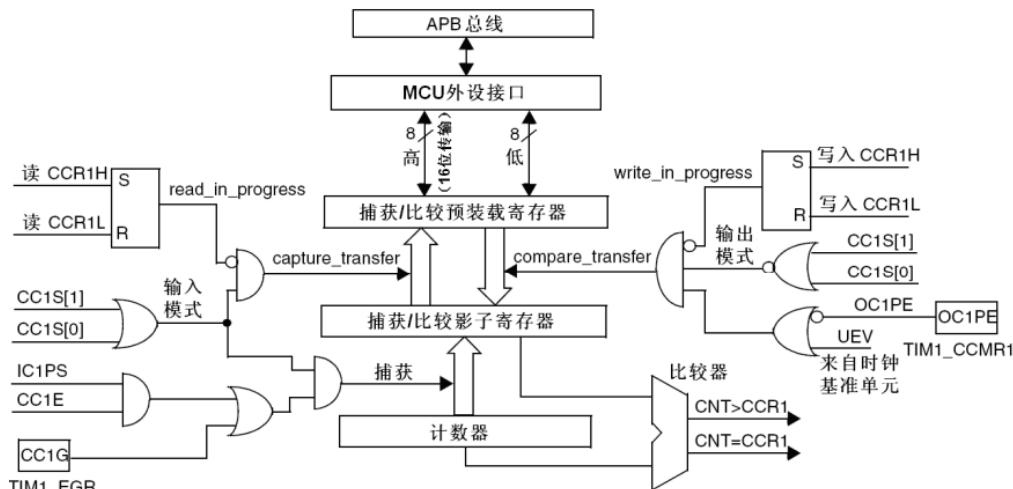


图 14-27 捕获/比较通道 1 的主电路

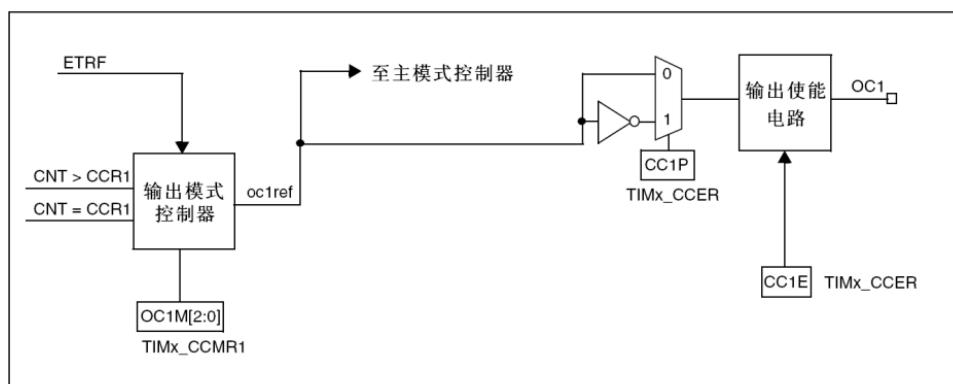


图 14-28 捕获/比较通道的输出部分(通道 1)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

#### 14.2.5. 输入捕获模式

在输入捕获模式下，当检测到 IC<sub>x</sub> 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(TIM<sub>x</sub>\_CCR<sub>x</sub>)中。当发生捕获事件时，相应的 CC<sub>x</sub>IF 标志(TIM<sub>x</sub>\_SR 寄存器)被置 1，如果开放了中断或者 DMA 操作，则将产生中断。

或者 DMA 请求。如果发生捕获事件时 CC<sub>x</sub>IF 标志已经为高，那么过捕获标志 CC<sub>x</sub>OF(TIM<sub>x</sub>\_SR 寄存器)被置 1。通过写 CC<sub>x</sub>IF=0 可清除 CC<sub>x</sub>IF，或读取存储在 TIM<sub>x</sub>\_CCR<sub>x</sub> 寄存器中的捕获数据也可清除 CC<sub>x</sub>IF。写 CC<sub>x</sub>OF=0 可清除 CC<sub>x</sub>OF。

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 TIM<sub>x</sub>\_CCR1 寄存器中，步骤如下：

- 选择有效输入端： TIM<sub>x</sub>\_CCR1 必须连接到 TI1 输入，所以写入 TIM<sub>x</sub>\_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道就被配置为输入，并且 TIM<sub>x</sub>\_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 TI<sub>x</sub> 时，输入滤波器控制位是 TIM<sub>x</sub>\_CCMR<sub>x</sub> 寄存器中的 IC<sub>x</sub>F 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 TIM<sub>x</sub>\_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIM<sub>x</sub>\_CCER 寄存器中写入 CC1P=0(设置为上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIM<sub>x</sub>\_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIM<sub>x</sub>\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIM<sub>x</sub>\_DIER 寄存器中的 CC1IE 位允许相关中断请求，也可通过设置 TIM<sub>x</sub>\_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 TIM<sub>x</sub>\_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。
- 为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIM<sub>x</sub>\_EGR 寄存器中相应的 CC<sub>x</sub>G 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

#### 14.2.6. PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，其余操作与输入捕获模式相同：

- 两个 IC<sub>x</sub> 信号被映射至同一个 TI<sub>x</sub> 输入。
- 这 2 个 IC<sub>x</sub> 信号为边沿有效，但是极性相反。
- 其中一个 TI<sub>x</sub>FP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIM<sub>x</sub>\_CCR1 寄存器)和占空比(TIM<sub>x</sub>\_CCR2 寄存器)，具体步骤如下(取决于 CK\_INT 的频率和预分频器的值)。

- 选择 TIM<sub>x</sub>\_CCR1 的有效输入：置 TIM<sub>x</sub>\_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 TIM<sub>x</sub>\_CCR1 中和清除计数器)：置 CC1P=0(上升沿有效)。
- 选择 TIM<sub>x</sub>\_CCR2 的有效输入：置 TIM<sub>x</sub>\_CCMR1 寄存器的 CC2S=10(选中 TI1)。
- 选择 TI1FP2 的有效极性(捕获数据到 TIM<sub>x</sub>\_CCR2)：置 CC2P=1(下降沿有效)。
- 选择有效的触发输入信号：置 TIM<sub>x</sub>\_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 TIM<sub>x</sub>\_SMCR 中的 SMS=100。

- 使能捕获：置  $\text{TIMx\_CCER}$  寄存器中  $\text{CC1E}=1$  且  $\text{CC2E}=1$ 。

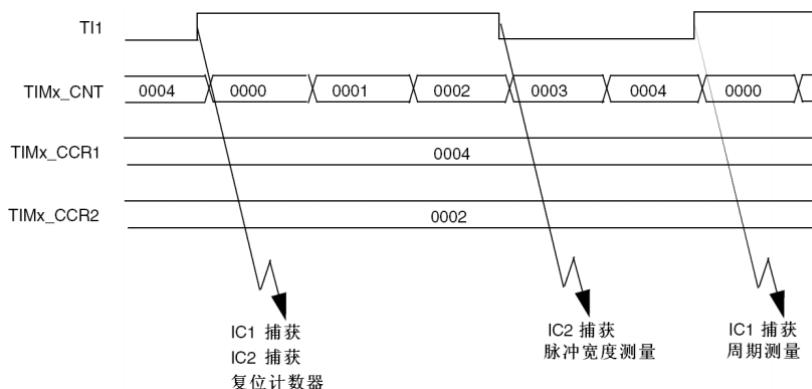


图 14-29 PWM 输入模式时序

因为只有  $\text{TI1FP1}$  和  $\text{TI2FP2}$  连到了从模式控制器，所以 PWM 输入模式只能使用  $\text{TIMx\_CH1}/\text{TIMx\_CH2}$  信号。

#### 14.2.7. 强置输出模式

在输出模式( $\text{TIMx\_CCMRx}$  寄存器中  $\text{CCxS}=00$ )下，输出比较信号( $\text{OCxREF}$  和相应的  $\text{OCx}$ )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  $\text{TIMx\_CCMRx}$  寄存器中相应的  $\text{OCxM}=101$ ，即可强置输出比较信号( $\text{OCxREF}/\text{OCx}$ )为有效状态。这样  $\text{OCxREF}$  被强置为高电平( $\text{OCxREF}$  始终为高电平有效)，同时  $\text{OCx}$  得到  $\text{CCxP}$  极性相反的信号。

例如： $\text{CCxP}=0$ ( $\text{OCx}$  高电平有效)，则  $\text{OCx}$  被强置为高电平。

置  $\text{TIMx\_CCMRx}$  寄存器中的  $\text{OCxM}=100$ ，可强置  $\text{OCxREF}$  信号为低。

该模式下，在  $\text{TIMx\_CCRx}$  影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

#### 14.2.8. 输出比较模式

此项功能是用来控制一个输出波形，或者表明一段给定的时间已经到时。

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( $\text{TIMx\_CCMRx}$  寄存器中的  $\text{OCxM}$  位)和输出极性( $\text{TIMx\_CCER}$  寄存器中的  $\text{CCxP}$  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( $\text{OCxM}=000$ )、被设置成有效电平( $\text{OCxM}=001$ )、被设置成无效电平( $\text{OCxM}=010$ )或进行翻转( $\text{OCxM}=011$ )。
- 设置中断状态寄存器中的标志位( $\text{TIMx\_SR}$  寄存器中的  $\text{CCxIF}$  位)。
- 若设置了相应的中断屏蔽( $\text{TIMx\_DIER}$  寄存器中的  $\text{CCxIE}$  位)，则产生一个中断。
- 若设置了相应的使能位( $\text{TIMx\_DIER}$  寄存器中的  $\text{CCxDE}$  位， $\text{TIMx\_CR2}$  寄存器中的  $\text{CCDS}$  位选择 DMA 请求功能)，则产生一个 DMA 请求。

可以通过配置  $\text{TIMx\_CCMRx}$  中的  $\text{OCxPE}$  位选择  $\text{TIMx\_CCRx}$  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  $\text{OCxREF}$  和  $\text{OCx}$  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：

- 选择计数器时钟(内部，外部，预分频器)。
- 将相应的数据写入  $\text{TIMx\_ARR}$  和  $\text{TIMx\_CCRx}$  寄存器中。

- 如果要产生一个中断请求，设置 CCxE 位。
- 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
- 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。

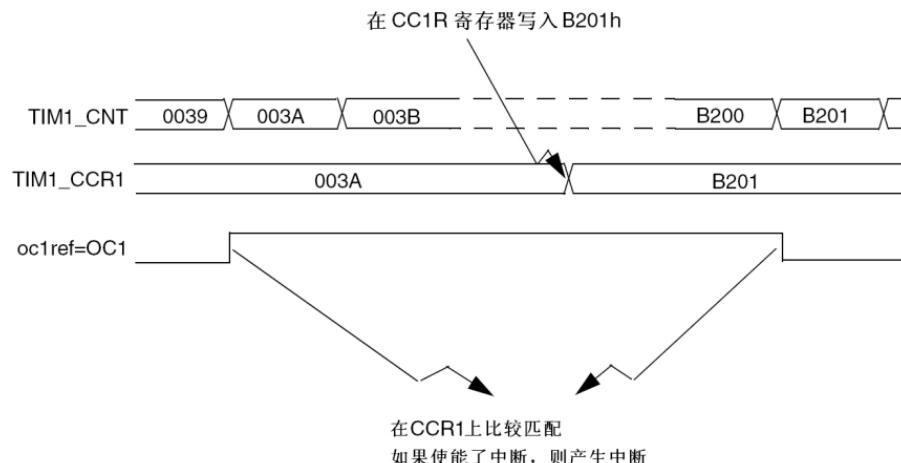


图 14-30 输出比较模式，翻转 OC1

#### 14.2.9. PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入‘110’(PWM 模式 1)或‘111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx\_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，用户必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。TIMx\_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx\_CCERx 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx\_CCRx≤TIMx\_CNT 或者 TIMx\_CNT≤TIMx\_CCRx。

然而为了与 OCREF\_CLR 的功能(在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF)一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变，或
- 当输出比较模式(TIMx\_CCMRx 寄存器中的 OCxM 位)从“冻结”(无比较，OCxM='000')切换到某个 PWM 模式 (OCxM='110'或'111')。

这样在运行中可以通过软件强置 PWM 输出。

根据 TIMx\_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

#### 14.2.9.1. PWM 边沿对齐模式

##### ■ 向上计数配置

当 TIMx\_CR1 寄存器中的 DIR 位为低的时候执行向上计数。

下面是一个 PWM 模式 1 的例子。当 TIMx\_CNT<TIMx\_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自动重装载值(TIMx\_ARR)，则 OCxREF 保持为'1'。如果比较值为 0，则 OCxREF 保持为'0'。下图为 TIMx\_ARR=8 时边沿对齐的 PWM 波形实例。

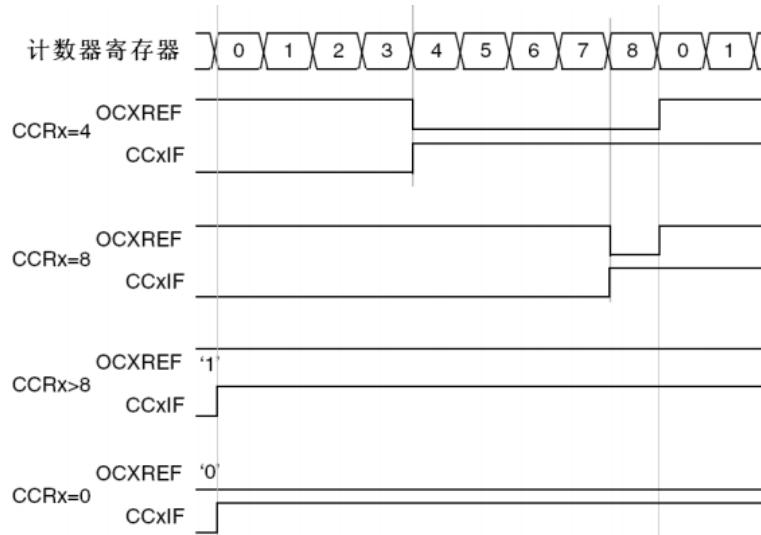


图 14-31 边沿对齐的 PWM 波形(ARR=8)

##### ■ 向下计数的配置

当 TIMx\_CR1 寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当 TIMx\_CNT>TIMx\_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx\_CCRx 中的比较值大于 TIMx\_ARR 中的自动重装载值，则 OCxREF 保持为'1'。该模式下不能产生 0% 的 PWM 波形。

#### 14.2.9.2. PWM 中央对齐模式

当 TIMx\_CR1 寄存器中的 CMS 位不为'00'时为中央对齐模式(CMS 位的所有其他的配置对 OCxREF/OCx 信号都有相同的作用)。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置 1、在计数器向下计数时被置 1、或在计数器向上和向下计数时被置 1。TIMx\_CR1 寄存器中的计数方向位(DIR)由硬件更新，不要用软件修改它。

下图给出了一些中央对齐的 PWM 波形的例子

- TIMx\_ARR=8
- PWM 模式 1
- TIMx\_CR1 寄存器的 CMS=01，在中央对齐模式 1 下，当计数器向下计数时设置比较标志。

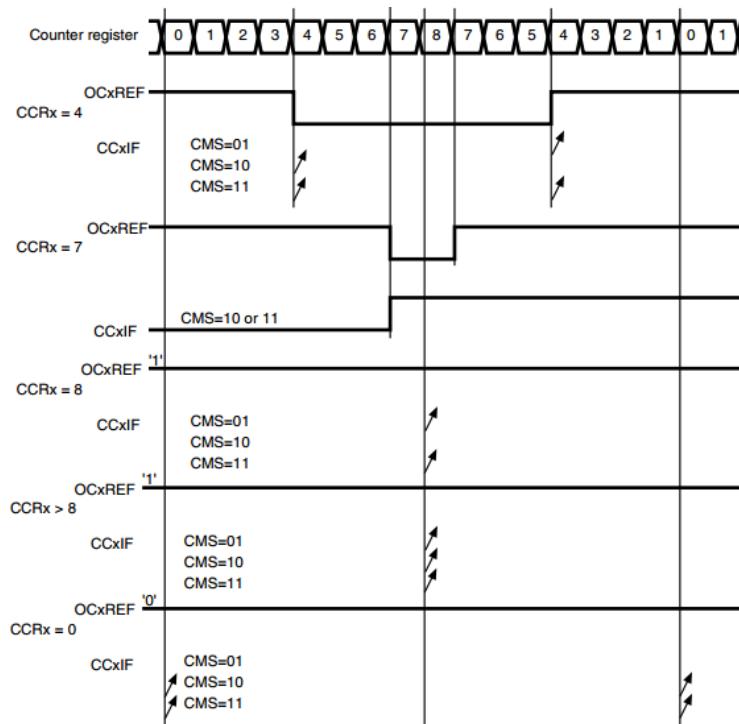


图 14-32 央对齐的 PWM 波形(APR=8)

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；这意味着计数器向上还是向下计数取决于 TIMx\_CR1 寄存器中 DIR 位的当前值。此外，不要通过软件同时修改 DIR 和 CMS 位。
  - 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
    - 如果写入计数器的值大于自动重加载的值(TIMx\_CNT>TIMx\_ARR)，则方向不会被更新。
- 例如，如果计数器正在向上计数，它就会继续向上计数。如果将 0 或者 TIMx\_ARR 的值写入计数器，方向被更新，但不会产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新(设置 TIMx\_EGR 位中的 UG 位)，并且不要在计数进行过程中修改计数器的值。

#### 14.2.10. 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx\_CCMRx 寄存器中对应的 OCxCE 位为‘1’，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以联到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

- 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。
- 必须禁止外部时钟模式 2：TIMx\_SMCR 寄存器中的 ECE=0。
- 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。

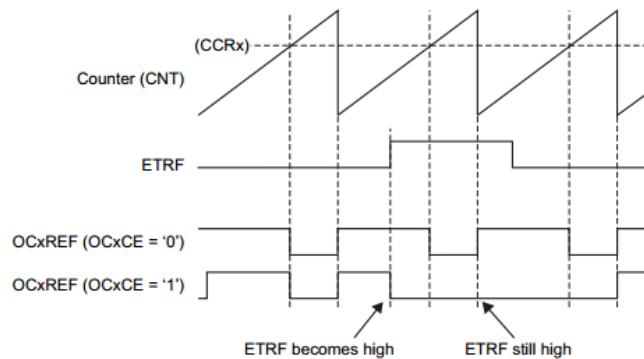


图 14-33 清除 TIMx 的 OCxREF

#### 14.2.11. 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$  (特别地,  $0 < CCRx$ )，
- 向下计数方式：计数器  $CNT > CCRx$ 。

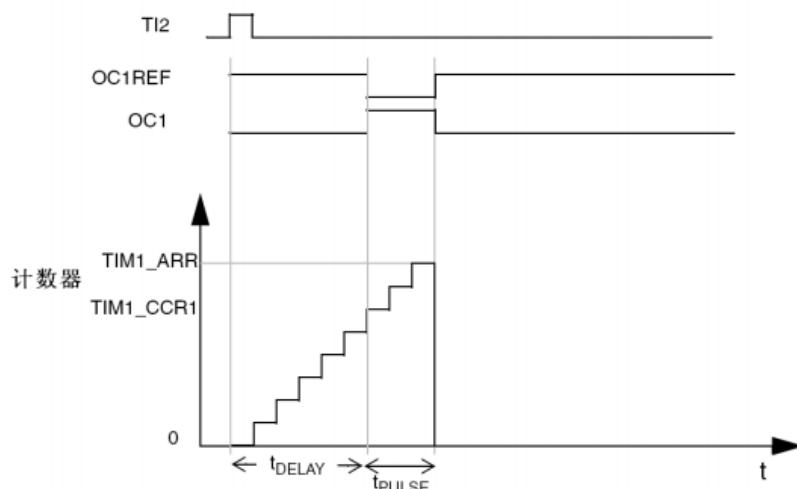


图 14-34 单脉冲模式的例子

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)

- tDELAY 由 TIMx\_CCR1 寄存器中的值定义。
- tPULSE 由自动装载值和比较值之间的差值定义(TIMx\_ARR - TIMx\_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE=1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM=1，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。当 OPM=0 时，重复模式被选中。

#### 14.2.11.1. 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 tDELAY。

如果要以最小延时输出波形，可以设置 TIMx\_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF(和 OCx)直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

#### 14.2.12. 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx\_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx\_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。假定计数器已经启动(TIMx\_CR1 寄存器中的 CEN=1)，则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx\_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端(TI1 或者 TI2)的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx\_ARR 寄存器的自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIMx\_ARR；同样，捕获器、比较器、预分频器、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 14-1 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

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TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S='01'(TIMx\_CCMR1 寄存器, 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)

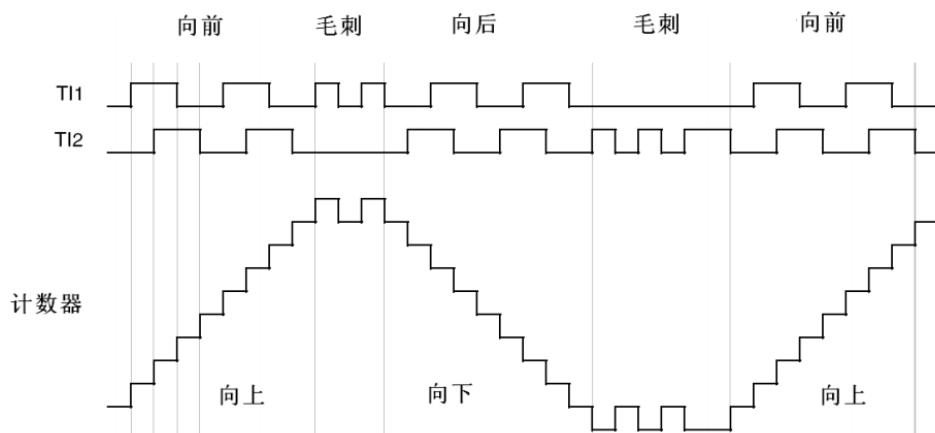


图 14-35 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1', 其他配置与上例相同)

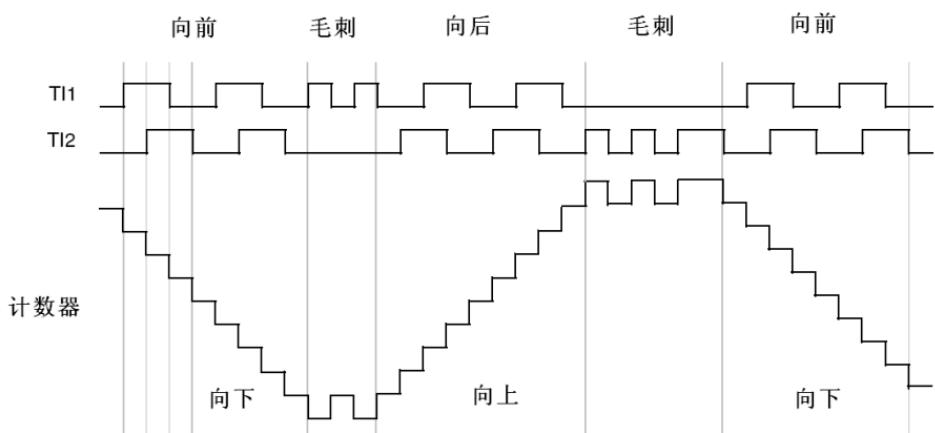


图 14-36 编码器模式下 IC1FP1 极性反相时计数器操作实例

当定时器配置成编码器接口模式时，可以提供传感器当前位置的信息。通过将第二个定时器配置在捕获模式，可以测量两个编码器事件的间隔，获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 14.2.13. 定时器输入异或功能

TIMx\_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx\_CH1、TIMx\_CH2 和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。TIM1\_8 SPEC 给出了此特性用于连接霍尔传感器的例子。

### 14.2.14. TIMx 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 14.2.14.1. 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时如果 TIMx\_CR1 寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器(TIMx\_ARR, TIMx\_CCRx)都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽(在本例中，不需要任何滤波器，因此保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=0 以确定极性(只检测上升沿)。
- 置 TIMx\_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志(TIMx\_SR 寄存器中的 TIF 位)被设置，根据 TIMx\_DIER 寄存器中 TIE(中断使能)位和 TDE(DMA 使能)位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx\_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

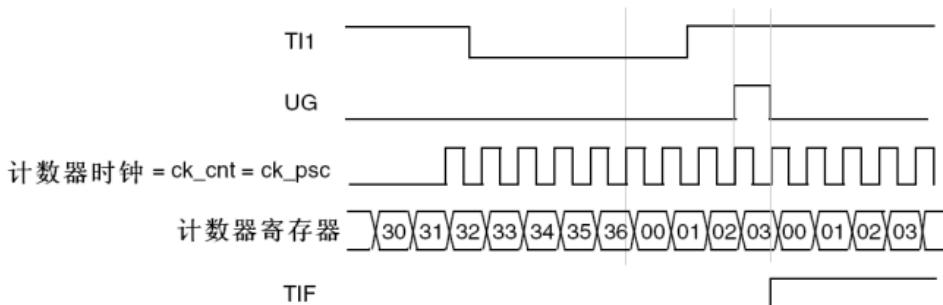


图 14-37 复位模式下的控制电路

#### 14.2.14.2. 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P=1 以确定极性(只检测低电平)。
- 置 TIMx\_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx\_SR 中的 TIF 标置。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

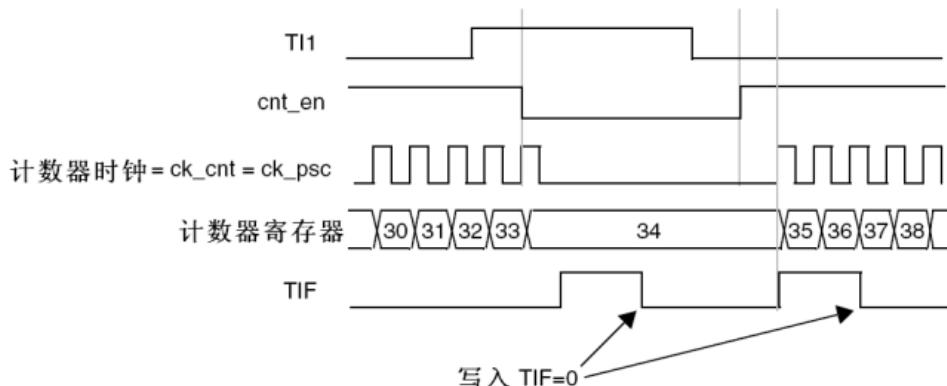


图 14-38 门控模式下的控制电路

#### 14.2.14.3. 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。

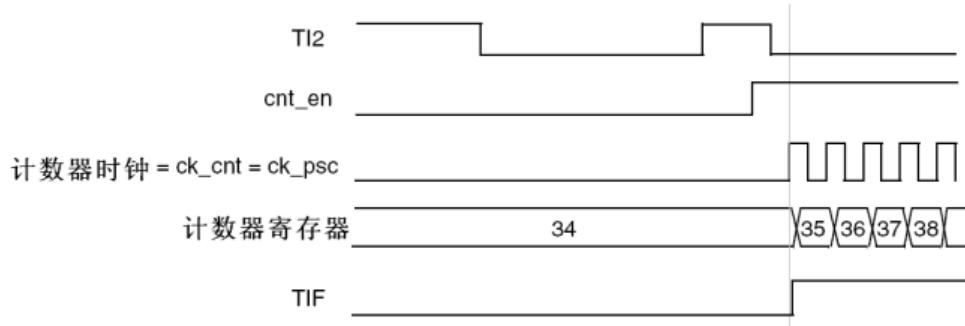


图 14-39 触发器模式下的控制电路

#### 14.2.14.4. 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：

- 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。

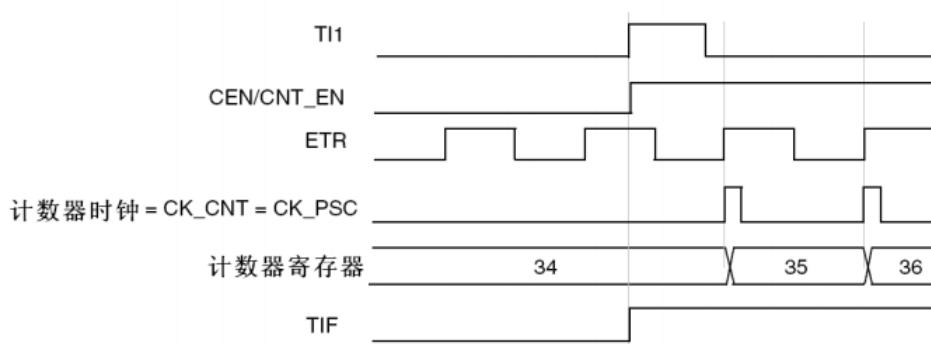


图 14-40 外部时钟模式 2 + 触发模式下的控制电路

#### 14.2.15. 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

#### 14.2.15.1. 使用一个定时器作为另一个定时器的预分频器

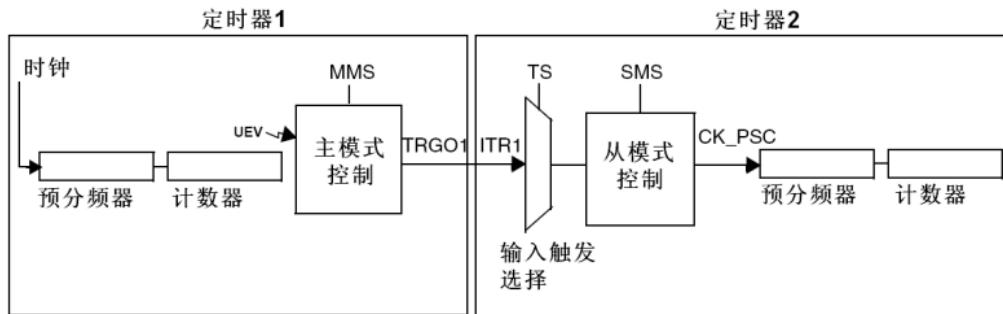


图 14-41 主/从定时器的例子

如：可以配置定时器 1 作为定时器 2 的预分频器。进行下述操作：

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1\_CR2 寄存器的 MMS=‘010’时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。
- 连接定时器 1 的 TRGO1 输出至定时器 2，设置 TIM2\_SMCR 寄存器的 TS=‘000’，配置定时器 2 为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式 1(TIM2\_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期性的上升沿(即定时器 1 的计数器溢出)信号驱动。
- 最后，必须设置相应(TIMx\_CR1 寄存器)的 CEN 位分别启动两个定时器。

注：如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

#### 14.2.15.2. 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。

只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以 3(fCK\_CNT=fCK\_INT/3)得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1\_CR2 寄存器的 MMS=100)
- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM2\_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1

注：定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

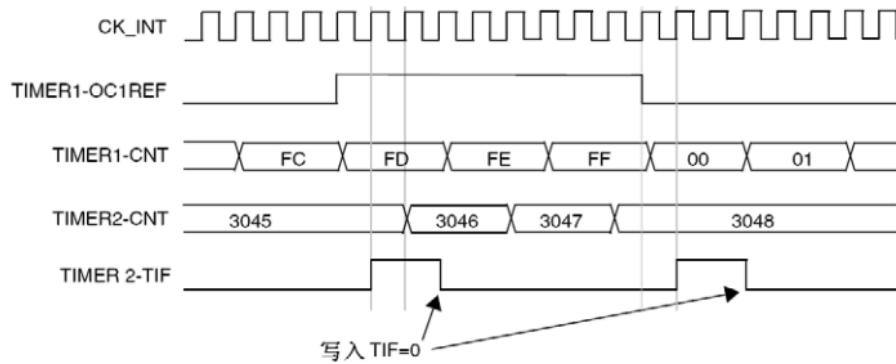


图 14-42 定时器 1 的 OC1REF 控制定时器 2

在图 4-49 的例子中，在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx\_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写‘0’到 TIM1\_CR1 的 CEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1\_CR2 寄存器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)。
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)。
- 置 TIM1\_EGR 寄存器的 UG='1'，复位定时器 1。
- 置 TIM2\_EGR 寄存器的 UG='1'，复位定时器 2。
- 写‘0XE7’至定时器 2 的计数器(TIM2\_CNTL)，初始化它为 0xE7。
- 置 TIM2\_CR1 寄存器的 CEN='1'以使能定时器 2。
- 置 TIM1\_CR1 寄存器的 CEN='1'以启动定时器 1。
- 置 TIM1\_CR1 寄存器的 CEN='0'以停止定时器 1。

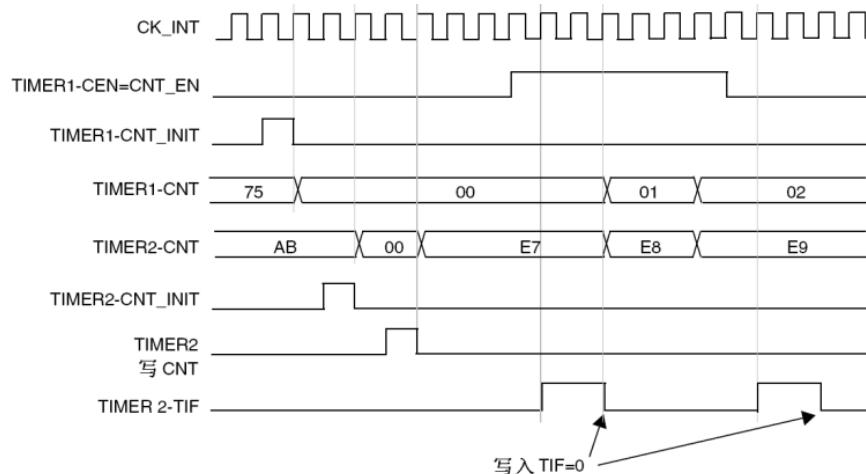


图 14-43 通过使能定时器 1 可以控制定时器 2

#### 14.2.15.3. 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。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(TIM1\_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1\_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)。
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)。
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1。

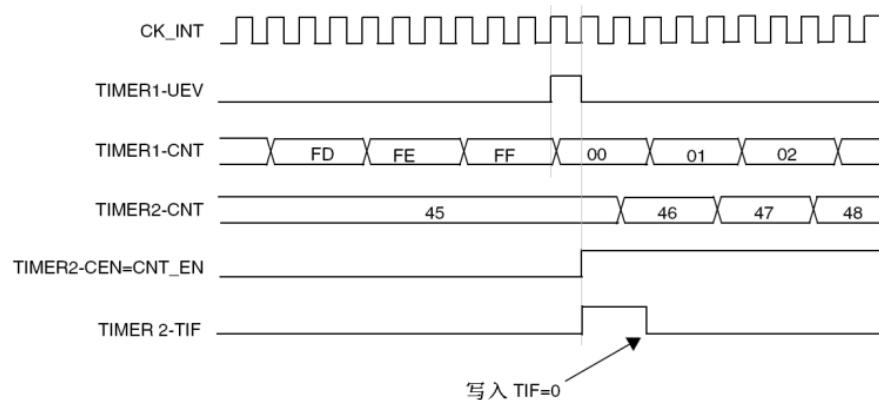


图 14-44 使用定时器 1 的更新触发定时器 2

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。

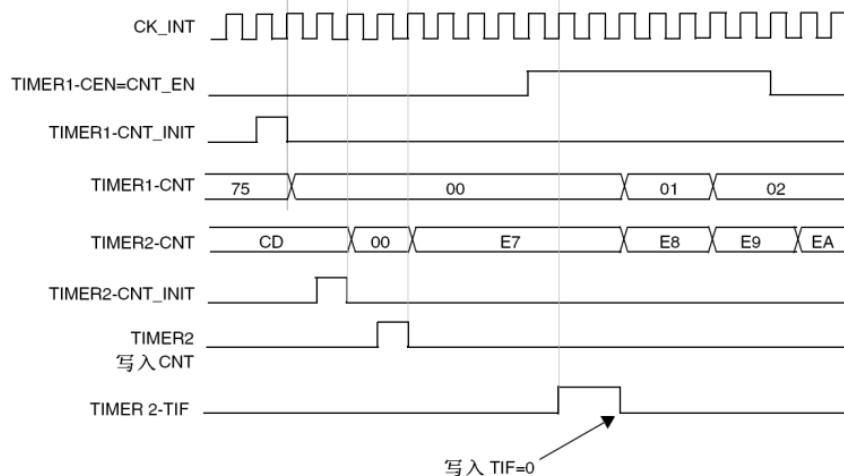


图 14-45 利用定时器 1 的使能触发定时器 2

#### 14.2.15.4. 使用一个定时器作为另一个的预分频器

这个例子使用定时器 1 作为定时器 2 的预分频器。配置如下：

● 配置定时器 1 为主模式，送出它的更新事件 UEV 做为触发输出(TIM1\_CR2 寄存器的 MMS='010')。然后每次计数器溢出时输出一个周期信号；

- 配置定时器 1 的周期(TIM1\_ARR 寄存器)；
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)；
- 配置定时器 2 使用外部时钟模式(TIM2\_SMCR 寄存器的 SMS=111)；
- 置 TIM1\_CR2 寄存器的 CEN=1 以启动定时器 2；
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1。

#### 14.2.15.5. 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 4-48。保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1\_CR2 寄存器的 MMS=001)。
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1\_SMCR 寄存器的 TS=100)。
- 配置定时器 1 为触发模式(TIM1\_SMCR 寄存器的 SMS=110)。
- 配置定时器 1 为主/从模式， TIM1\_SMCR 寄存器的 MSM=1。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx\_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT\_EN 和 CK\_PSC 之间有个延迟。

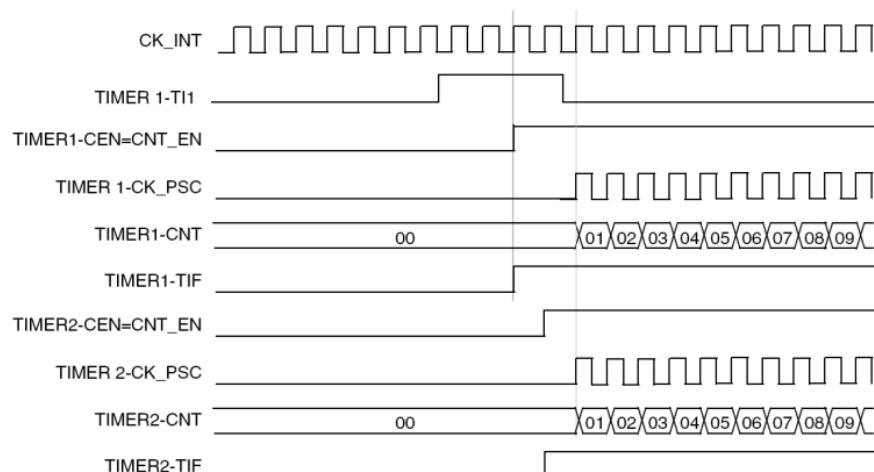


图 14-46 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

#### 14.2.16. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止)，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。

## 14.3. 寄存器描述

TIM2 寄存器地址: 0x4000 0000

TIM3 寄存器地址: 0x4000 0400

TIM4 寄存器地址: 0x4000 0800

TIM5 寄存器地址: 0x4000 0C00

### 14.3.1. TIM2/3/4/5 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
保留						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	保留, 始终为 0			
9:8	CKD	RW	0	<p>时钟分频因子 (Clock division)            这 2 位定义在定时器时钟(CK_INT)频率与数字滤波器(ETR,TIx)所用的采样时钟 (tDTS) 之间的分频比例。</p> <p>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</p>
7	ARPE	RW	0	<p>自动重装载预装载允许位 (Auto-reload preload enable)            0: TIMx_ARR 寄存器没有缓冲;            1: TIMx_ARR 寄存器被装入缓冲器。</p>
6:5	CMS	RW	0	<p>选择中央对齐模式 (Center-aligned mode selection)            00: 边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。            01: 中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 只在计数器向下计数时被设置。            10: 中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 只在计数器向上计数时被设置。            11: 中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIMx_CCMRx 寄存器中 CCxS=00)的输出比较中断标志位, 在计数器向上和向下计数时均被设置。            注: 在计数器开启时(CEN=1), 不允许从边沿对齐模式转换到中央对齐模式。</p>
4	DIR	RW	0	<p>方向 (Direction)            0: 计数器向上计数;            1: 计数器向下计数。            注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读。</p>
3	OPM	RW	0	<p>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。</p>
2	URS	RW	0	<p>更新请求源 (Update request source)            软件通过该位选择 UEV 事件的源            0: 如果使能了更新中断或 DMA 请求, 则下述任一事件产生更新中断或 DMA 请求:            - 计数器溢出/下溢            - 设置 UG 位            - 从模式控制器产生的更新</p>

Bit	Name	R/W	Reset Value	Function
				1: 如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
1	UDIS	RW	0	<p>禁止更新 (Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新(UEV)事件由下述任一事件产生: - 计数器溢出/下溢 - 设置 UG 位 - 从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。(译注: 更新影子寄存器) 1: 禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。</p>
0	CEN	RW	0	<p>使能计数器 (Counter enable) 0: 禁止计数器; 1: 使能计数器。 注: 在软件设置了 CEN 位后, 外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。</p>

### 14.3.2. TIM2/3/4/5 控制寄存器 2 (TIMx\_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								T1S	MMS [2:0]			CCDS	保留		
保留								RW	RW			RW	保留		

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
7	TI1S	RW	0	<p>TI1 选择 (TI1 selection) 0: TIMx_CH1 引脚连到 TI1 输入; 1: TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。</p>
6:4	MMS	RW	0	<p>主模式选择 (Master mode selection) 这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下: 000: 复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的复位(从模式控制器处于复位模式), 则 TRGO 上的信号相对实际的复位会有一个延迟。 001: 使能 – 计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时, TRGO 上会有一个延迟, 除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。 010: 更新 – 更新事件被选为触发输入(TRGO)。例如, 一个主定时器的时钟可以被用作一个从定时器的预分频器。 011: 比较脉冲 – 在发生一次捕获或一次比较成功时, 当要设置 CC1IF 标志时(即使它已经为高), 触发输出送出一个正脉冲(TRGO)。 100: 比较 – OC1REF 信号被用于作为触发输出(TRGO)。 101: 比较 – OC2REF 信号被用于作为触发输出(TRGO)。 110: 比较 – OC3REF 信号被用于作为触发输出(TRGO)。 111: 比较 – OC4REF 信号被用于作为触发输出(TRGO)。 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。</p>
3	CCDS	RW	0	<p>捕获/比较的 DMA 选择 (Capture/compare DMA selection) 0: 当发生 CCx 事件时, 送出 CCx 的 DMA 请求; 1: 当发生更新事件时, 送出 CCx 的 DMA 请求。</p>

Bit	Name	R/W	Reset Value	Function
2:0	保留, 始终读为 0。			

### 14.3.3. TIM2/3/4/5 从模式控制寄存器 (TIMx\_SMCR)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS [1:0]		ETF [3:0]		MSM		TS [2:0]		保留		SMS [2:0]			
RW	RW	RW		RW		RW		RW		保留		RW			

Bit	Name	R/W	Reset Value	Function
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。 注 4: tim2 的 ch1 和 etr 公用同一个 io 口, 应避免同时使用
13:12	ETPS	RW	0	外部触发预分频 (External trigger prescaler) 外部触发信号 ETRP 的频率必须最多是 TIMxCLK 频率的 1/4。当输入较快的外部时钟时, 可以使用预分频降低 ETRP 的频率。 00: 关闭预分频; 01: ETRP 频率除以 2; 10: ETRP 频率除以 4; 11: ETRP 频率除以 8。
11:8	ETF	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 0001: 采样频率 fSAMPLING=fCK_INT, N=2 0010: 采样频率 fSAMPLING=fCK_INT, N=4 0011: 采样频率 fSAMPLING=fCK_INT, N=8 0100: 采样频率 fSAMPLING=fDTS/2, N=6 0101: 采样频率 fSAMPLING=fDTS/2, N=8 0110: 采样频率 fSAMPLING=fDTS/4, N=6 0111: 采样频率 fSAMPLING=fDTS/4, N=8 1000: 采样频率 fSAMPLING=fDTS/8, N=6 1001: 采样频率 fSAMPLING=fDTS/8, N=8 1010: 采样频率 fSAMPLING=fDTS/16, N=5 1011: 采样频率 fSAMPLING=fDTS/16, N=6 1100: 采样频率 fSAMPLING=fDTS/16, N=8 1101: 采样频率 fSAMPLING=fDTS/32, N=5 1110: 采样频率 fSAMPLING=fDTS/32, N=6 1111: 采样频率 fSAMPLING=fDTS/32, N=8

Bit	Name	R/W	Reset Value	Function
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) 100: TI1 的边沿检测器(TI1F_ED) 001: 内部触发 1(ITR1) 101: 滤波后的定时器输入 1(TI1FP1) 010: 内部触发 2(ITR2) 110: 滤波后的定时器输入 2(TI2FP2) 011: 内部触发 3(ITR3) 111: 外部触发输入(ETRF) 更多有关 ITRx 的细节, 参见表 5-1。 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	保留, 始终读为 0。			
2:0	SMS	RW	0	从模式选择 (Slave mode selection) 当选择了外部信号, 触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000: 关闭从模式 – 如果 CEN=1, 则预分频器直接由内部时钟驱动。 001: 编码器模式 1 – 根据 TI1FP1 的电平, 计数器在 TI2FP2 的边沿向上/下计数。 010: 编码器模式 2 – 根据 TI2FP2 的电平, 计数器在 TI1FP1 的边沿向上/下计数。 011: 编码器模式 3 – 根据另一个信号的输入电平, 计数器在 TI1FP1 和 TI2FP2 的边沿向上/下计数。 100: 复位模式 – 选中的触发输入(TRGI)的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。 101: 门控模式 – 当触发输入(TRGI)为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110: 触发模式 – 计数器在触发输入 TRGI 的上升沿启动(但不复位), 只有计数器的启动是受控的。 111: 外部时钟模式 1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注: 如果 TI1F_EN 被选为触发输入(TS=100)时, 不要使用门控模式。这是因为, TI1F_ED 在每次 TI1F 变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。 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)

表 14-2 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	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

#### 14.3.4. TIM2/3/4/5 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
保留	RW	保留	RW	RW	RW	RW	RW	保留	RW	保留	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	保留, 始终为 0			
14	TDE	RW	0	允许触发 DMA 请求 (Trigger DMA request enable) 0: 禁止触发 DMA 请求;

Bit	Name	R/W	Reset Value	Function
				1: 允许触发 DMA 请求。
13	保留, 始终为 0			
12	CC4DE	RW	0	允许捕获/比较 4 的 DMA 请求 (Capture/Compare 4 DMA request enable) 0: 禁止捕获/比较 4 的 DMA 请求; 1: 允许捕获/比较 4 的 DMA 请求。
11	CC3DE	RW	0	允许捕获/比较 3 的 DMA 请求 (Capture/Compare 3 DMA request enable) 0: 禁止捕获/比较 3 的 DMA 请求; 1: 允许捕获/比较 3 的 DMA 请求。。
10	CC2DE	RW	0	允许捕获/比较 2 的 DMA 请求 (Capture/Compare 2 DMA request enable) 0: 禁止捕获/比较 2 的 DMA 请求; 1: 允许捕获/比较 2 的 DMA 请求。
9	CC1DE	RW	0	允许捕获/比较 1 的 DMA 请求 (Capture/Compare 1 DMA request enable) 0: 禁止捕获/比较 1 的 DMA 请求; 1: 允许捕获/比较 1 的 DMA 请求。。
8	UDE	RW	0	允许更新的 DMA 请求 (Update DMA request enable) 0: 禁止更新的 DMA 请求; 1: 允许更新的 DMA 请求。
7	保留, 始终为 0			
6	TIE	RW	0	触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断; 1: 使能触发中断。
5	保留, 始终为 0			
4	CC4IE	RW	0	允许捕获/比较 4 中断 (Capture/Compare 4 interrupt enable) 0: 禁止捕获/比较 4 中断; 1: 允许捕获/比较 4 中断。
3	CC3IE	RW	0	允许捕获/比较 3 中断 (Capture/Compare 3 interrupt enable) 0: 禁止捕获/比较 3 中断; 1: 允许捕获/比较 3 中断。
2	CC2IE	RW	0	允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0: 禁止捕获/比较 2 中断; 1: 允许捕获/比较 2 中断。
1	CC1IE	RW	0	允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0: 禁止捕获/比较 1 中断; 1: 允许捕获/比较 1 中断
0	UIE	RW	0	允许更新中断 (Update interrupt enable) 0: 禁止更新中断; 1: 允许更新中断。

#### 14.3.5. TIM2/3/4/5 状态寄存器 (TIMx\_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4OF	CC3OF	CC2OF	CC1OF	保留	TIF	保留	CC4IF	CC3IF	CC2IF	CC1IF	UIF			
保留	RC_W0	RC_W0	RC_W0	RC_W0	保留	RC_W0	保留	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0			

Bit	Name	R/W	Reset Value	Function
15:13	保留, 始终为 0			

Bit	Name	R/W	Reset Value	Function
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, 始终读为 0			
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, 始终读为 0			
4	CC4IF	RC_W0	0	捕获/比较 4 中断标记 (Capture/Compare 4 interrupt flag) 参考 CC1IF 描述。
3	CC3IF	RC_W0	0	捕获/比较 3 中断标记 (Capture/Compare 3 interrupt flag) 参考 CC1IF 描述。
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置 1，但在中心对称模式下除外(参考 TIMx_CR1 寄存器的 CMS 位)。它由软件清‘0’。 0: 无匹配发生； 1: TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APB 的内容时，在向上或向上/下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，CC1IF 位变高 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置‘1’，它由软件清‘0’或通过读 TIMx_CCR1 清‘0’。 0: 无输入捕获产生； 1: 计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	更新中断标记 (Update interrupt flag) 当产生更新事件时该位由硬件置‘1’。它由软件清‘0’。 0: 无更新事件产生； 1: 更新中断等待响应。当寄存器被更新时该位由硬件置‘1’： - 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时(重复计数器=0 时产生更新事件)。 - 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 - 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。(参考 TIM2/3/4/5 从模式控制寄存器(TIMx_SMCR))。

#### 14.3.6. TIM2/3/4/5 事件产生寄存器 (TIMx\_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										TG	保留	CC4G	CC3G	CC2G	CC1G

保留	W	保留	W	W	W	W	W
----	---	----	---	---	---	---	---

Bit	Name	R/W	Reset Value	Function
15:7	保留, 始终为 0			
6	TG	W	0	产生触发事件 (Trigger generation) 该位由软件置'1', 用于产生一个触发事件, 由硬件自动清'0'。 0: 无动作; 1: TIMx_SR 寄存器的 TIF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。
5	保留, 始终为 0			
4	CC4G	W	0	产生捕获/比较 4 事件 (Capture/Compare 4 generation) 参考 CC1G 描述。
3	CC3G	W	0	产生捕获/比较 3 事件 (Capture/Compare 3 generation) 参考 CC1G 描述。
2	CC2G	W	0	产生捕获/比较 2 事件 (Capture/Compare 2 generation) 参考 CC1G 描述。
1	CC1G	W	0	产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置'1', 用于产生一个捕获/比较事件, 由硬件自动清'0'。 0: 无动作; 1: 在通道 CC1 上产生一个捕获/比较事件: 若通道 CC1 配置为输出: 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。 若通道 CC1 配置为输入: 当前的计数器值被捕获至 TIMx_CCR1 寄存器; 设置 CC1IF=1, 若开启对应的中断和 DMA, 则产生相应的中断和 DMA。若 CC1IF 已经为 1, 则设置 CC1OF=1。
0	UG	W	0	产生更新事件 (Update generation) 该位由软件置'1', 由硬件自动清'0'。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清'0'; 若 DIR=1(向下计数)则计数器取 TIMx_ARR 的值。

#### 14.3.7. TIM2/3/4/5 捕获/比较模式控制寄存器 1 (TIMx\_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	OC2F [3:0]			OC2PSC [1:0]				OC1F [3:0]			OC1PSC [1:0]				
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

##### 14.3.7.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15	OC2CE	RW	0	输出比较 2 清 0 使能 (Output Compare 2 clear enable)
14:12	OC2M	RW	0	输出比较 2 模式 (Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能 (Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能 (Output Compare 2 fast enable)
9:8	CC2S	RW	0	捕获/比较 2 选择。(Capture/Compare 2 selection)

Bit	Name	R/W	Reset Value	Function
				<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(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	OC1CE	RW	0	<p>输出比较 1 清'0'使能 (Output Compare 1 clear enable)</p> <p>0: OC1REF 不受 ETRF 输入的影响;</p> <p>1: 一旦检测到 ETRF 输入高电平, 清除 OC1REF=0。</p>
6:4	OC1M	RW	0	<p>输出比较 1 模式 (Output Compare 1 mode)</p> <p>该 3 位定义了输出参考信号 OC1REF 的动作, 而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效, 而 OC1 的有效电平取决于 CC1P 位。</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; 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 (Output Compare 1 preload enable)</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, 并且新写入的数值立即起作用。</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加载至当前寄存器中。</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(TIMx_CR1 寄存器的 OPM=1), 可以在未确认预装载寄存器情况下使用 PWM 模式, 否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较 1 快速使能 (Output Compare 1 fast enable)</p> <p>该位用于加快 CC 输出对触发输入事件的响应。</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 只在通道被配置成 PWM1 或 PWM2 模式时起作用。</p>
1:0	CC1S	RW	0	<p>捕获/比较 1 选择。 (Capture/Compare 1 selection)</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>

### 14.3.7.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	输入捕获 2 滤波器 (Input capture 2 filter)
11:10	IC2PSC	RW	0	输入/捕获 2 预分频器 (Input capture 2 prescaler)
9:8	CC2S	RW	0	<p>捕获/比较 2 选择 (Capture/Compare 2 selection)            这 2 位定义通道的方向(输入/输出), 及输入脚的选择:            00: CC2 通道被配置为输出;            01: CC2 通道被配置为输入, IC2 映射在 TI2 上;            10: CC2 通道被配置为输入, IC2 映射在 TI1 上;            11: CC2 通道被配置为输入, IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。            注: CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。</p>
7:4	IC1F	RW	0	<p>输入捕获 1 滤波器 (Input capture 1 filter)            这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成, 它记录到 N 个事件后会产生一个输出的跳变:            0000: 无滤波器, 以 fDTS 采样 1000: 采样频率 fSAMPLING=fDTS/8, N=6            0001: 采样频率 fSAMPLING=fCK_INT, N=2 1001: 采样频率 fSAMPLING=fDTS/8, N=8            0010: 采样频率 fSAMPLING=fCK_INT, N=4 1010: 采样频率 fSAMPLING=fDTS/16, N=5            0011: 采样频率 fSAMPLING=fCK_INT, N=8 1011: 采样频率 fSAMPLING=fDTS/16, N=6            0100: 采样频率 fSAMPLING=fDTS/2, N=6 1100: 采样频率 fSAMPLING=fDTS/16, N=8            0101: 采样频率 fSAMPLING=fDTS/2, N=8 1101: 采样频率 fSAMPLING=fDTS/32, N=5            0110: 采样频率 fSAMPLING=fDTS/4, N=6 1110: 采样频率 fSAMPLING=fDTS/32, N=6            0111: 采样频率 fSAMPLING=fDTS/4, N=8 1111: 采样频率 fSAMPLING=fDTS/32, N=8</p>
3:2	IC1PSC	RW	0	<p>输入/捕获 1 预分频器 (Input capture 1 prescaler)            这 2 位定义了 CC1 输入(IC1)的预分频系数。            一旦 CC1E=0(TIMx_CCER 寄存器中), 则预分频器复位。            00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获;            01: 每 2 个事件触发一次捕获;            10: 每 4 个事件触发一次捕获;            11: 每 8 个事件触发一次捕获。</p>
1:0	CC1S	RW	0	<p>捕获/比较 1 选择 (Capture/Compare 1 Selection)            这 2 位定义通道的方向(输入/输出), 及输入脚的选择:            00: CC1 通道被配置为输出;            01: CC1 通道被配置为输入, IC1 映射在 TI1 上;            10: CC1 通道被配置为输入, IC1 映射在 TI2 上;            11: CC1 通道被配置为输入, IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。            注: CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。</p>

### 14.3.8. TIM2/3/4/5 捕获/比较模式制寄存器 2 (TIMx\_CCMR2)

Address offset:0x1C

Reset value:0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M [2:0]				OC4PE	CC4S [1:0]		OC3CE	OC3M [2:0]				OC3PE	OC3FE	CC3S [1:0]
IC4F [3:0]				IC4PSC [1:0]					IC3F [3:0]				IC3PSC [1:0]		
RW	RW	RW	RW	RW	RW		RW		RW		RW	RW	RW	RW	

#### 14.3.8.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15	OC4CE	RW	0	输出比较 4 清 0 使能 (Output Compare 4 clear enable)
14:12	OC4M	RW	0	输出比较 4 模式 (Output Compare 4 mode)
11	OC4PE	RW	0	输出比较 4 预装载使能 (Output Compare 4 preload enable)
10	OC4FE	RW	0	输出比较 4 快速使能 (Output Compare 4 fast enable)
9:8	CC4S	RW	0	捕获/比较 4 选择。 (Capture/Compare 4 selection) 该 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC4 通道被配置为输出; 01: CC4 通道被配置为输入, IC4 映射在 TI4 上; 10: CC4 通道被配置为输入, IC4 映射在 TI3 上; 11: CC4 通道被配置为输入, IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7	OC3CE	RW	0	输出比较 3 清 0 使能 (Output Compare 1 clear enable)
6:4	OC3M	RW	0	输出比较 3 模式 (Output Compare 3 mode)
3	OC3PE	RW	0	输出比较 3 预装载使能 (Output Compare 3 preload enable)
2	OC3FE	RW	0	输出比较 3 快速使能 (Output Compare 3 fast enable)
1:0	CC3S	RW	0	捕获/比较 3 选择。 (Capture/Compare 3 selection) 该 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC3 通道被配置为输出; 01: CC3 通道被配置为输入, IC3 映射在 TI3 上; 10: CC3 通道被配置为输入, IC3 映射在 TI4 上; 11: CC3 通道被配置为输入, IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

#### 14.3.8.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:12	IC4F	RW	0	输入捕获 4 滤波器 (Input capture 4 filter)
11:10	IC4PSC	RW	0	输入/捕获 4 预分频器 (Input capture 4 prescaler)
9:8	CC4S	RW	0	捕获/比较 4 选择 (Capture/Compare 4 selection) 该 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC4 通道被配置为输出; 01: CC4 通道被配置为输入, IC4 映射在 TI4 上; 10: CC4 通道被配置为输入, IC4 映射在 TI3 上; 11: CC4 通道被配置为输入, IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC4S 仅在通道关闭时(TIMx_CCER 寄存器的 CC4E=0)才是可写的。
7:4	IC3F	RW	0	输入捕获 3 滤波器 (Input capture 3 filter)
3:2	IC3PSC	RW	0	输入/捕获 3 预分频器 (Input capture 3 prescaler)
1:0	CC3S	RW	0	捕获/比较 3 选择 (Capture/Compare 3 Selection) 该 2 位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC3 通道被配置为输出; 01: CC3 通道被配置为输入, IC3 映射在 TI3 上; 10: CC3 通道被配置为输入, IC3 映射在 TI4 上; 11: CC3 通道被配置为输入, IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。 注: CC3S 仅在通道关闭时(TIMx_CCER 寄存器的 CC3E=0)才是可写的。

#### 14.3.9. TIM2/3/4/5 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4P	CC4E	保留	CC3P	CC3E	保留	CC2P	CC2E	保留	CC1P	CC1E				
保留	RW	RW													

Bit	Name	R/W	Reset Value	Function
15:14	保留, 始终为 0			
13	CC4P	RW	0	输入/捕获 4 输出极性 (Capture/Compare 4 output polarity) 参考 CC1P 的描述。
12	CC4E	RW	0	输入/捕获 4 输出使能 (Capture/Compare 4 output enable) 参考 CC1E 的描述。
11:10	保留, 始终为 0			
9	CC3P	RW	0	输入/捕获 3 输出极性 (Capture/Compare 3 output polarity) 参考 CC1P 的描述。
8	CC3E	RW	0	输入/捕获 3 输出使能 (Capture/Compare 3 output enable) 参考 CC1E 的描述。
7:6	保留, 始终为 0			
5	CC2P	RW	0	输入/捕获 2 输出极性 (Capture/Compare 2 output polarity) 参考 CC1P 的描述。
4	CC2E	RW	0	输入/捕获 2 输出使能 (Capture/Compare 2 output enable) 参考 CC1E 的描述。
3:2	保留, 始终为 0			
1	CC1P	RW	0	输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) CC1 通道配置为输出： 0: OC1 高电平有效； 1: OC1 低电平有效。 CC1 通道配置为输入： 该位选择是 IC1 还是 IC1 的反相信号作为触发或捕获信号。 0: 不反相：捕获发生在 IC1 的上升沿；当用作外部触发器时，IC1 不反相。 1: 反相：捕获发生在 IC1 的下降沿；当用作外部触发器时，IC1 反相。
0	CC1E	RW	0	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出： 0: 关闭 - OC1 禁止输出。 1: 开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0: 捕获禁止； 1: 捕获使能。

表 14-3 标准OCx通道的输出控制

CCXE 位	OCx 输出状态
0	禁止输出(OCx=0, OCx_EN=0)
1	OCx = OCxREF + 极性, OCx_EN=1

注：连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

#### 14.3.10. TIM2/3/4/5 计数器 (TIMx\_CNT)

Address offset:0x24

Reset value:0x0000

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

CNT
RW

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	计数器的值 (Counter value)

#### 14.3.11. TIM2/3/4/5 预分频器 (TIMx\_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	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)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

#### 14.3.12. TIM2/3/4/5 自动重装载寄存器 (TIMx\_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 详细参考有关 ARR 的更新和动作。 当自动重装载的值为空时，计数器不工作。

#### 14.3.13. TIM2/3/4/5 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	<p>捕获/比较通道 1 的值 (Capture/Compare 1 value)</p> <p>若 CC1 通道配置为输出:</p> <p>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。</p> <p>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:</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>

#### 14.3.14. TIM2/3/4/5 捕获/比较寄存器 2 (TIMx\_CCR2)

Address offset:0x38

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW	0	<p>捕获/比较通道 1 的值 (Capture/Compare 1 value)</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_CCMR2 寄存器(OC2PE 位)中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。</p> <p>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:</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>

#### 14.3.15. TIM2/3/4/5 捕获/比较寄存器 3 (TIMx\_CCR3)

Address offset:0x3C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR3	RW	0	<p>捕获/比较通道 3 的值 (Capture/Compare 3 value)</p> <p>若 CC3 通道配置为输出:</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_CCMR3 寄存器(OC3PE 位)中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。</p> <p>否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较 3 寄存器中。</p> <p>当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较, 并在 OC3 端口上产生输出信号。</p> <p>若 CC3 通道配置为输入:</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>

#### 14.3.16. TIM2/3/4/5 捕获/比较寄存器 4 (TIMx\_CCR4)

Address offset:0x40

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR4	RW	0	<p>捕获/比较通道 4 的值 (Capture/Compare 4 value)            若 CC4 通道配置为输出：            CCR4 包含了装入当前捕获/比较 4 寄存器的值(预装载值)。            如果在 TIMx_CCMR4 寄存器(OC4PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。            否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。            当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。            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>

#### 14.3.17. TIM2/3/4/5 DMA 控制寄存器 (TIMx\_DCR)

Address offset:0x48

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:13	保留, 始终为 0			
12:8	DBL	RW	0	<p>DMA 连续传送长度 (DMA burst length)            这些位定义了 DMA 在连续模式下的传送长度(当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送)，即：定义传输的次数，传输可以是半字(双字节)或字节：            00000: 1 次传输            00001: 2 次传输            00010: 3 次传输            .....            10001: 18 次传输            例：我们考虑这样的传输：DBL=7, DBA=TIM2_CR1            - 如果 DBL=7, DBA=TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出：            (TIMx_CR1 的地址) + DBA + (DMA 索引)，其中 DMA 索引 = DBL            其中(TIMx_CR1 的地址) + DBA 再加上 7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址(TIMx_CR1 的地址) + DBA 开始的 7 个寄存器。            根据 DMA 数据长度的设置，可能发生以下情况：            - 如果设置数据为半字(16 位)，那么数据就会传输给全部 7 个寄存器。            - 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。</p>
7:5	保留, 始终读为 0。			
4:0	DBA	RW	0	<p>DMA 基址 (DMA base address)            这些位定义了 DMA 在连续模式下的基址(当对 TIMx_DMAR 寄存器进行读或写时)，DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量：            00000: TIMx_CR1,            00001: TIMx_CR2,</p>

Bit	Name	R/W	Reset Value	Function
				00010: TIMx_SMCR, .....

#### 14.3.18. TIM2/3/4/5 连续模式的 DMA 地址 (TIMx\_DMAR)

Address offset:0x4C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DMAB	RW	0	DMA 连续传送寄存器 (DMA register for burst accesses) 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： TIMx_CR1 地址 + DBA + DMA 索引，其中： “TIMx_CR1 地址”是控制寄存器 1(TIMx_CR1)所在的地址； “DBA”是 TIMx_DCR 寄存器中定义的基址址； “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。

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。

#### 14.3.19. TIM2/3/4/5 寄存器映射

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x0000	TIMx_C_R1																																			
	Read/Write																																			
	Reset Value																																			
0x0004	TIMx_C_R2																																			
	Read/Write																																			
	Reset Value																																			
0x0008	TIMx_S_MR																																			
	Read/Write																																			
	Reset Value																																			
					</td																															

		Offset		Register	
		TIMx_DIER		TIMx_DIER	
0x10		Read/Write		Read/Write	
0x14		Reset Value		Reset Value	
0x18		TIMx_SR		Reserved	
0x1C		Read/Write		Read/Write	
0x20		Reset Value		Reset Value	
0x24		TIMx_EGR		Reserved	
		Read/Write		Read/Write	
		Reset Value		Reset Value	
		TIMx_CM_R1		Reserved	
		Read/Write		Read/Write	
		Reset Value		Reset Value	
		TIMx_CM_R2		Reserved	
		Read/Write		Read/Write	
		Reset Value		Reset Value	
		TIMx_CCE_R		Reserved	
		Read/Write		Read/Write	
		Reset Value		Reset Value	
		TIMx_CNT		Reserved	
		Read/Write		Read/Write	
		Reset Value		Reset Value	
		OC2C		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC4CE		OC4OF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		IC4F		CC3OF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC4M		CC2OF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC4PSC		CC1OF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC4PE		UDE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC4FE		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC4S		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC3CE		TIE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		IC1F		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		OC1M		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC4G		CC4IF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC3G		CC3IF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC2G		CC2IF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC1G		CC1IF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC1S		UIF	
		Read/W		Read/W	
		Write		Write	
		0		0	
		UG		UG	
		Read/W		Read/W	
		Write		Write	
		0		0	
		UIE		UIE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		TDE		TDE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC4DE		CC4DE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC3DE		CC3DE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC2DE		CC2DE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		CC1DE		CC1DE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		UDE		UDE	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	
		Read/W		Read/W	
		Write		Write	
		0		0	
		Reserved		Reserved	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Re-set Value																																
0x28	TIMx_PSC Rea d/Write Re-set Value	Reserved																															
0x2C	TIMx_ARR Rea d/Write Re-set Value																																
0x34	TIMx_CR1 Rea d/Write Re-set Value																																
0x38	TIMx_CR2 Rea d/Write Re-set Value	Reserved																															
0x3C	TIMx_CR3 Rea d/Write Re-set Value																																
0x40	TIMx_CR4 Rea d/Write Re-set Value																																
0x48	TIMx_DCR Rea d/Write Re-set Value	Reserved																DBL	Reserved														

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0									
0	TIMx_DR	Reserved																DMAB																								
4	Read/Write	rw																																								
C	Reset Value																																									

# 15. 通用定时器 (TIM9和 TIM12)

## 15.1. 简介

通用定时器(TIM9 和 TIM12)由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。

它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM)。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

通用定时器 TIM9 和 TIM12 是完全独立的，它们不共享任何资源，可以同步操作。

### 15.1.1. TIM9 和 TIM12 主要特征

TIM9 和 TIM12 定时器的功能包括：

- 16 位向上自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 多达 2 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘对齐模式)
  - 单脉冲模式输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出，计数器初始化(通过软件或者内部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部触发计数)
  - 输入捕获
  - 输出比较

### 15.1.2. 模块框图

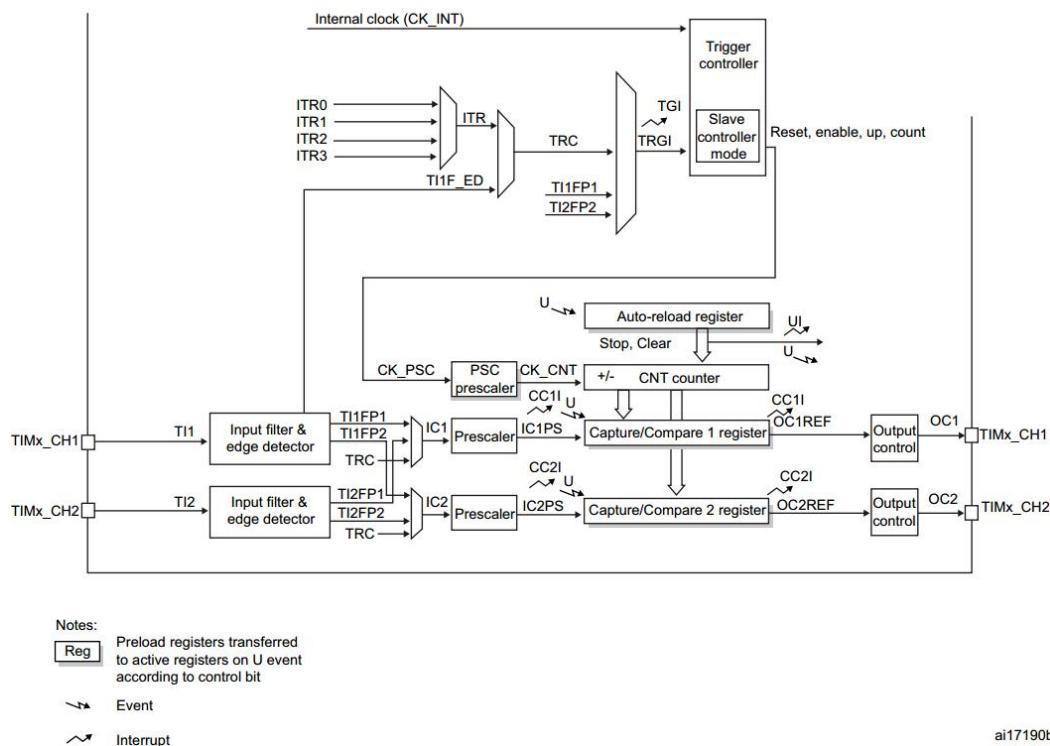


图 15-1 TIM9 和 TIM12 模块

## 15.2. TIM9 和 TIM12 功能描述

### 15.2.1. 时基单元

通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数。

计数器的时钟可以被预分频器分频。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)
- 自动装载寄存器 (TIMx\_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问它的预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 (UEV) 时传送到影子寄存器。当计数器达到上溢条件并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK\_CNT 驱动，仅当设置了 TIMx\_CR1 寄存器中的计数器使能位(CEN)，CK\_CNT 才对计数器有效。(更多有关使能计数器的细节，请参见从模式控制器的描述)。

注意，在设置了 TIMx\_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

#### 15.2.1.1. 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx\_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频参数将在下一次更新事件到来时被采用。

在预分频器运行时，更改计数器参数的例子如下所示。

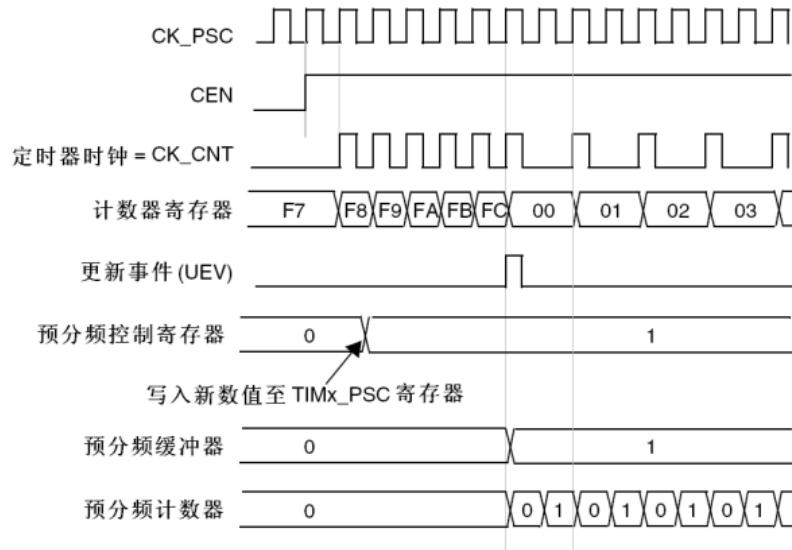


图 15-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

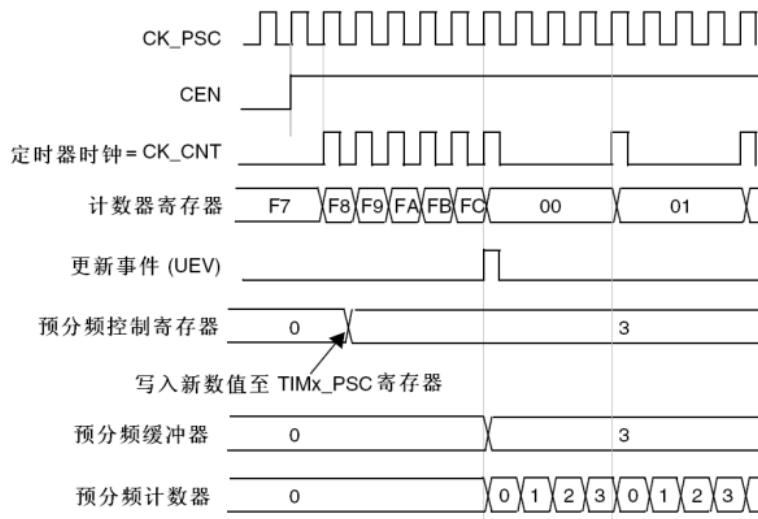


图 15-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

## 15.2.2. 计数器模式

### 15.2.2.1. 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx\_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

每次计数上溢都会产生更新事件。

而在 TIMx\_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx\_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx\_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx\_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx\_ARR=0x36 时计数器在不同时钟频率下的动作。

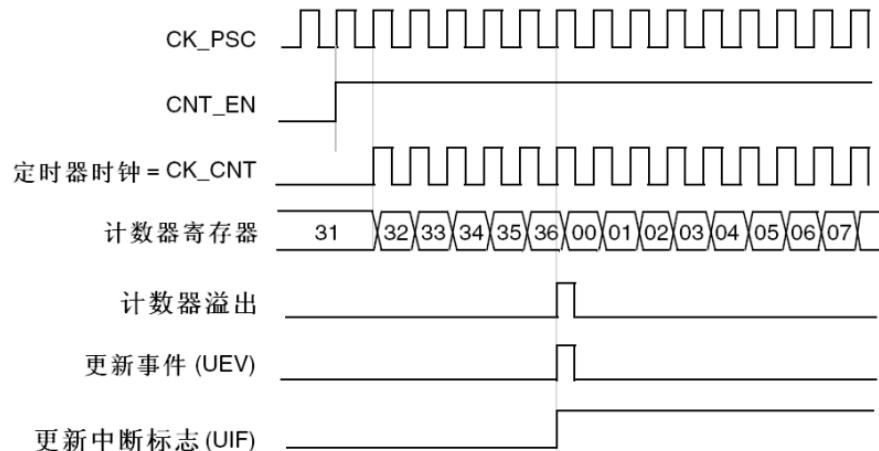


图 15-4 计数器时序图，内部时钟分频因子为 1

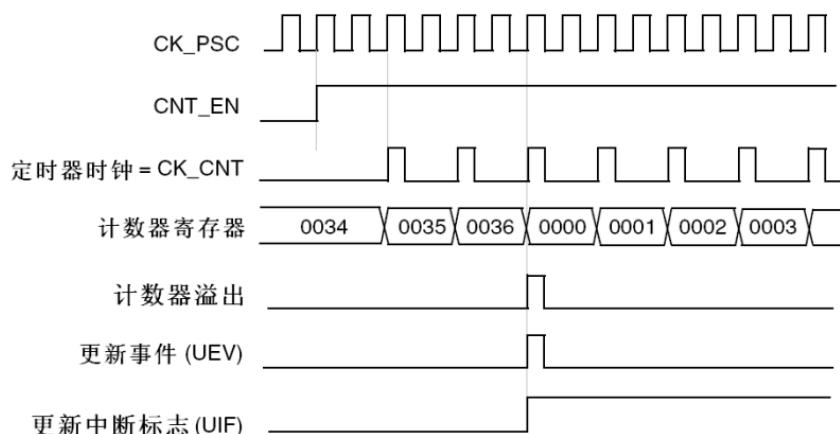


图 15-5 计数器时序图，内部时钟分频因子为 2

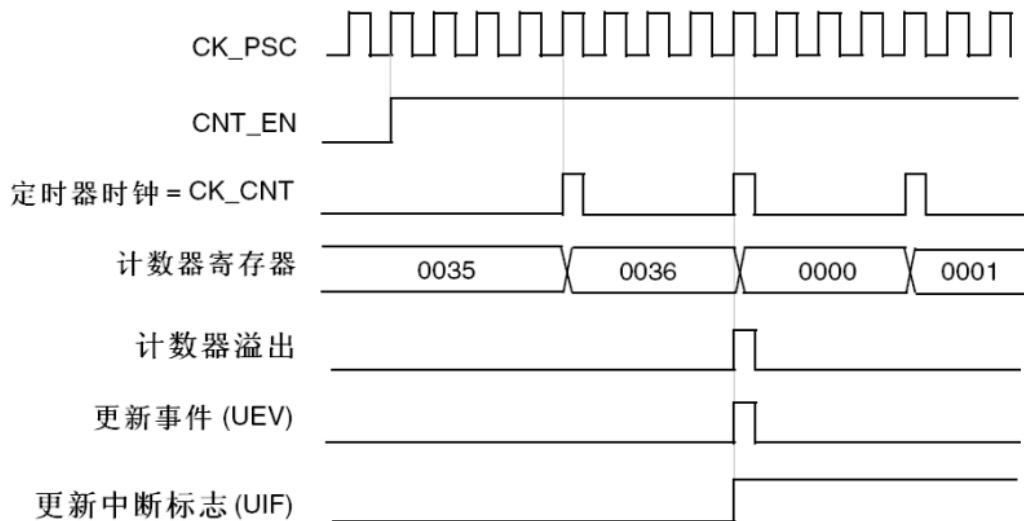


图 15-6 计数器时序图，内部时钟分频因子为 4

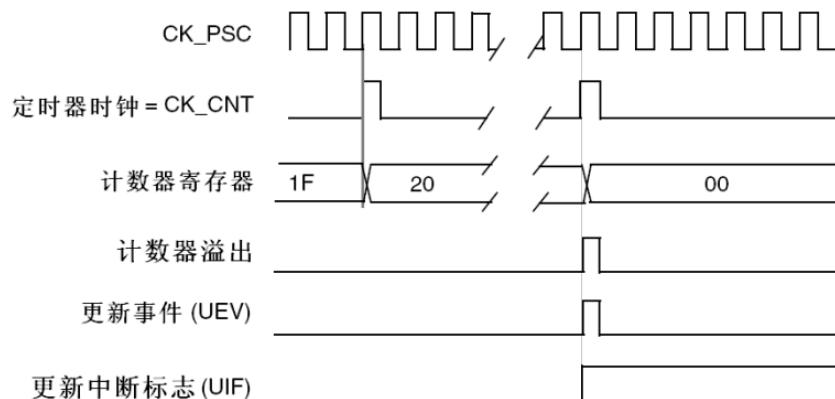


图 15-7 计数器时序图，内部时钟分频因子为 N

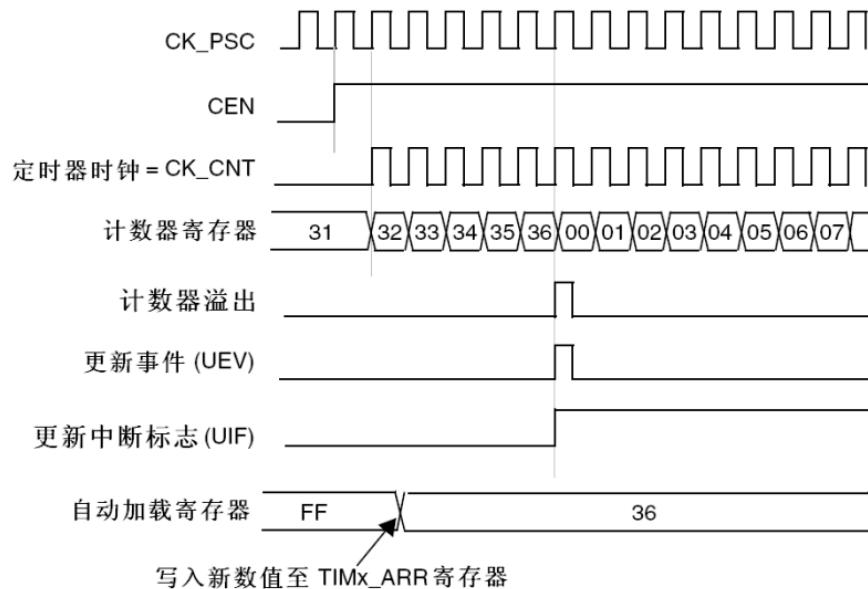


图 15-8 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx\_ARR)

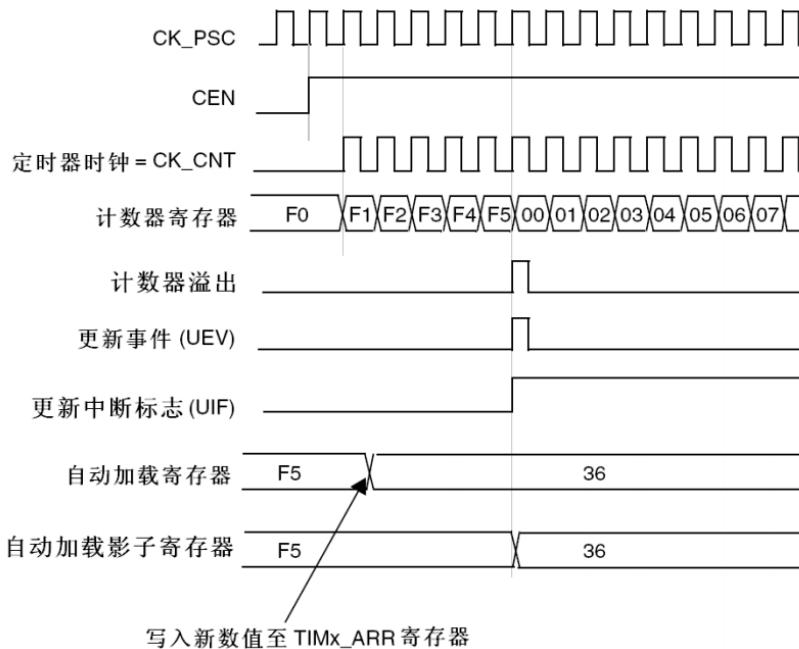


图 15-9 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx\_ARR)

### 15.2.3. 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK\_INT)
- 外部时钟模式 1：外部输入引脚 (TIx)
- 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

#### 15.2.3.1. 内部时钟源(CK\_INT)

如果禁止了从模式控制器(SMS=000)，则 CEN 和 UG 位(TIMx\_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成‘1’，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

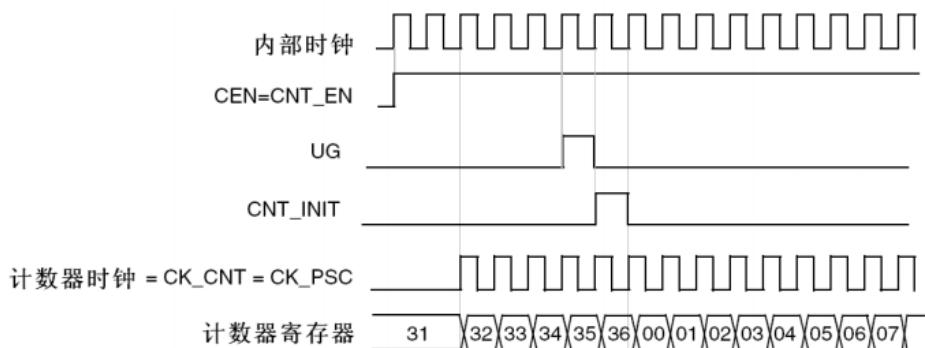


图 15-10 一般模式下的控制电路，内部时钟分频因子为 1

#### 15.2.3.2. 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。

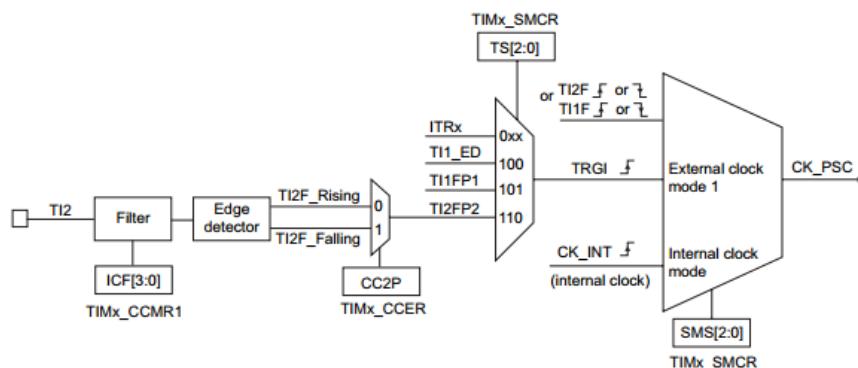


图 15-11 TI2 外部时钟连接例子

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。

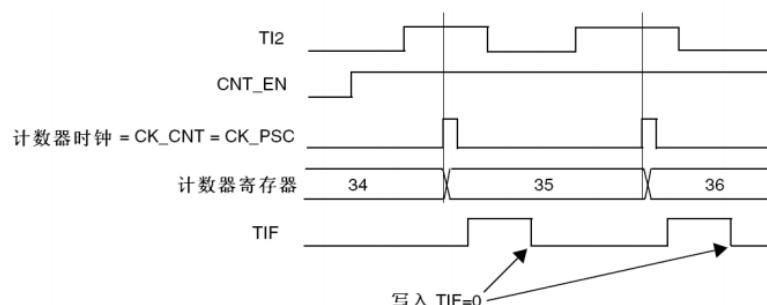


图 15-12 外部时钟模式 1 下的控制电路

#### 15.2.4. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。

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)。在捕获寄存器前分频得到 ICxPS。

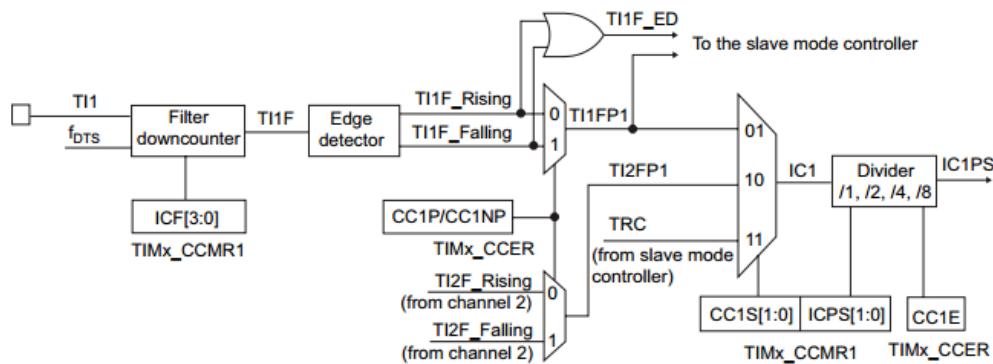


图 15-13 捕获/比较通道(如：通道 1 输入部分)

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。

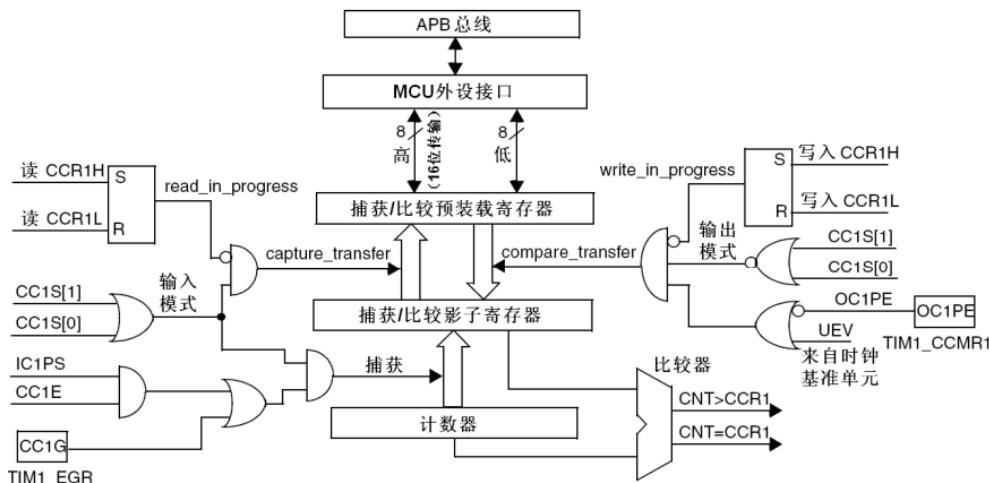


图 15-14 捕获/比较通道 1 的主电路

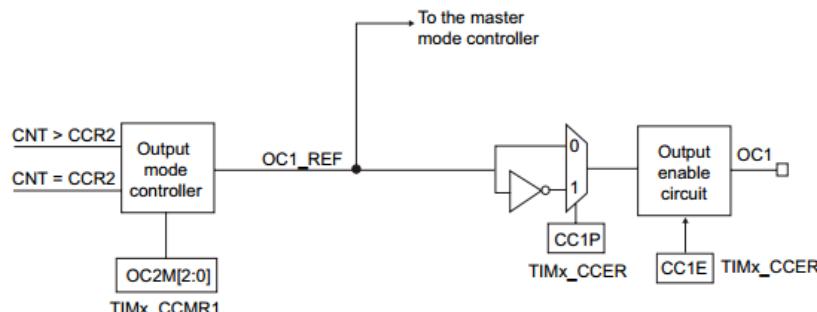


图 15-15 捕获/比较通道的输出部分(通道 1)

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

### 15.2.5. 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器(TIMx\_CCRx)中。当发生捕获事件时，相应的 CCxIF 标志(TIMx\_SR 寄存器)被置 1，如果开放了中断操作，则将产生中断请求。如果发生捕获事件时 CCxIF 标志已经为高，那么过捕获标志 CCxOF(TIMx\_SR 寄存器)被置 1。通过写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx\_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx\_CCR1 寄存器中，步骤如下：

- 选择有效输入端：TIMx\_CCR1 必须连接到 TI1 输入，所以写入 TIMx\_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为‘00’，通道就被配置为输入，并且 TIMx\_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽(即输入为 TIx 时，输入滤波器控制位是 TIMx\_CCMRx 寄存器中的 ICxF 位)。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以(以 fDTS 频率)连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx\_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIMx\_CCER 寄存器中 CC1P 和 CC1NP 位写入 00(设置为上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIMx\_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIMx\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIMx\_DIER 寄存器中的 CC1IE 位允许相关中断请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断请求。

### 15.2.6. PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，其余操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，用户可以测量输入到 TI1 上的 PWM 信号的周期(TIMx\_CCR1 寄存器)和占空比(TIMx\_CCR2 寄存器)，具体步骤如下(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 和 CC1NP 为 00(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 和 CC2NP 为 10(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。

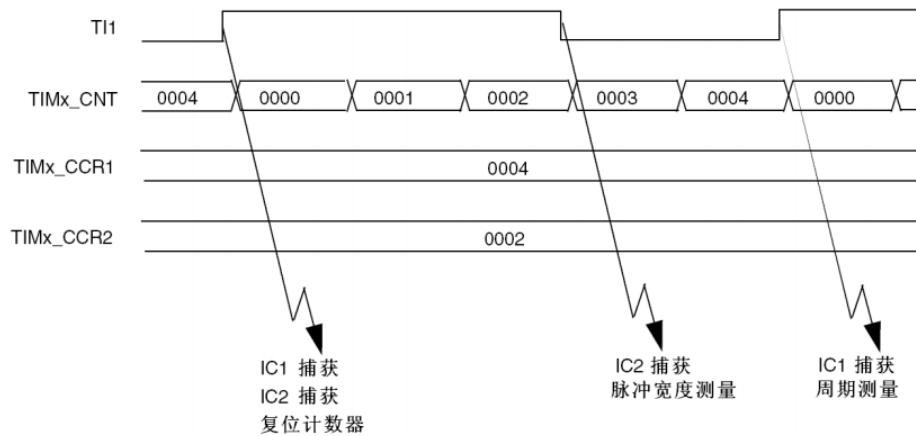


图 15-16 PWM 输入模式时序

因为只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx\_CH1/TIMx\_CH2 信号。

### 15.2.7. 强置输出模式

在输出模式(TIMx\_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx\_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断请求。这将会在下面的输出比较模式一节中介绍。

### 15.2.8. 输出比较模式

此项功能是用来控制一个输出波形，或者表明一段给定的时间已经到时。

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 CCxF 位)。
- 若设置了相应的中断屏蔽(TIMx\_DIER 寄存器中的 CCxIE 位)，则产生一个中断。

可以通过配置 TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

- 选择计数器时钟(内部，外部，预分频器)。
- 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
- 如果要产生一个中断请求，设置 CCxIE 位。
- 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。

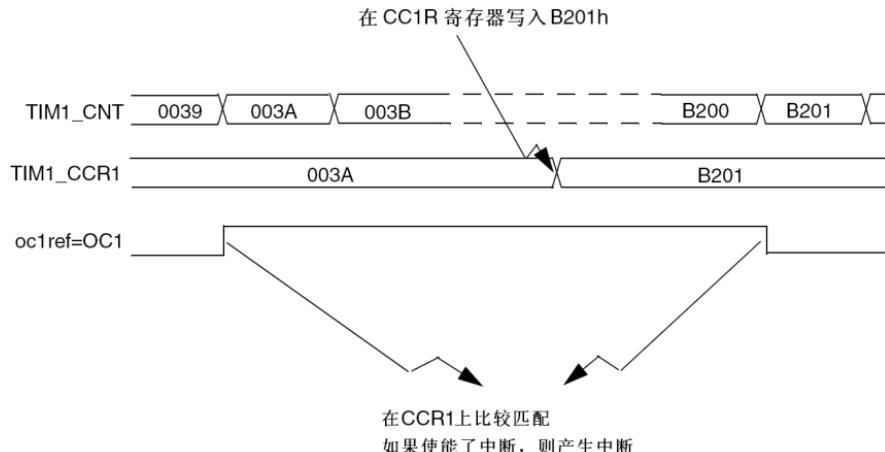


图 15-17 输出比较模式，翻转 OC1

### 15.2.9. PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入‘110’(PWM 模式 1)或‘111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx\_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，用户必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过(TIMx\_CCER 中)CCxE 位控制。详见 TIMx\_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx\_CCRx≤TIMx\_CNT 或者 TIMx\_CNT≤TIMx\_CCRx。

定时器能够产生边沿对齐的 PWM 信号。

#### 15.2.9.1. PWM 边沿对齐模式

##### • 向上计数配置

下面是一个 PWM 模式 1 的例子。当 TIMx\_CNT<TIMx\_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自动重装载值(TIMx\_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx\_ARR=8 时边沿对齐的 PWM 波形实例。

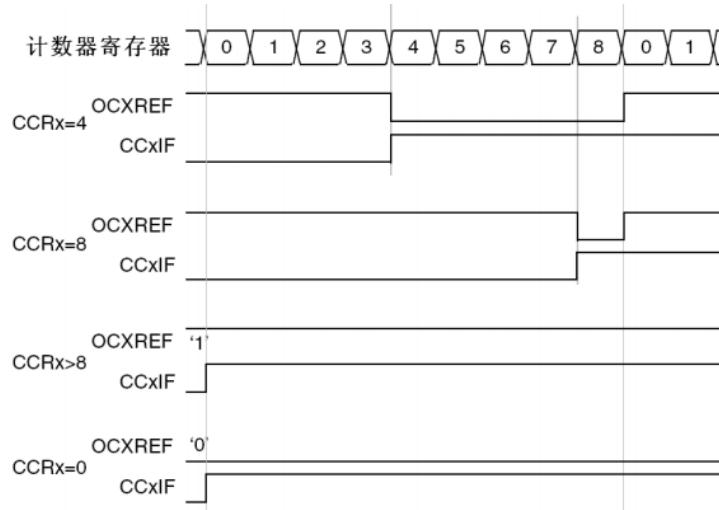


图 15-18 边沿对齐的 PWM 波形(ARR=8)

### 15.2.10. 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$  (特别地,  $0 < CCRx$ )。

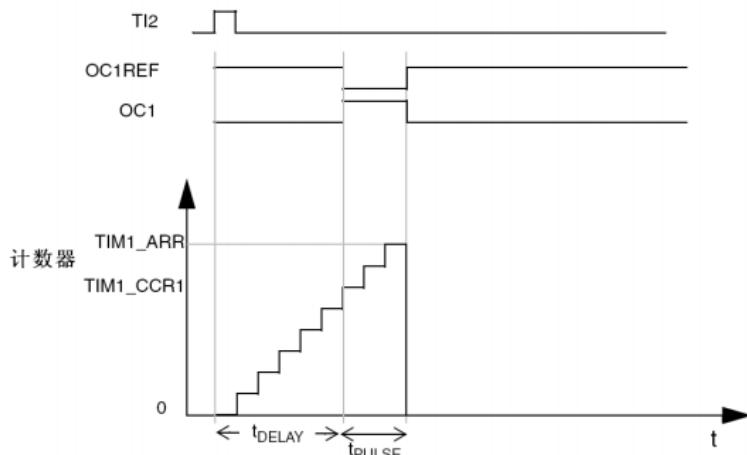


图 15-19 单脉冲模式的例子

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 和 CC2NP=00，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。

- tPULSE 由自动装载值和比较值之间的差值定义( $\text{TIMx\_ARR} - \text{TIMx\_CCR1}$ )。

• 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  $\text{TIMx\_CCMR1}$  寄存器的  $\text{OC1M}=111$ ，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置  $\text{TIMx\_CCMR1}$  中的  $\text{OC1PE}=1$  和  $\text{TIMx\_CR1}$  寄存器中的 ARPE；然后在  $\text{TIMx\_CCR1}$  寄存器中填写比较值，在  $\text{TIMx\_ARR}$  寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中， $\text{CC1P}=0$ 。

因为只需要一个脉冲，所以必须设置  $\text{TIMx\_CR1}$  寄存器中的 OPM=1，在下一个更新事件(当计数器从自动装载值翻转到 0)时停止计数。当 OPM=0 时，重复模式被选中。

#### 15.2.10.1. 特殊情况：OCx 快速使能：

在单脉冲模式下，在  $\text{TIx}$  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  $\text{TIMx\_CCMRx}$  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。

#### 15.2.11. TIMx 定时器和外部触发的同步

$\text{TIMx}$  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。

##### 15.2.11.1. 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果  $\text{TIMx\_CR1}$  寄存器的 UDIS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器( $\text{TIMx\_ARR}$ ,  $\text{TIMx\_CCRx}$ )都被更新了。

在以下的例子中， $\text{TI1}$  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  $\text{TI1}$  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  $\text{IC1F}=0000$ )。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  $\text{TIMx\_CCMR1}$  寄存器中  $\text{CC1S}=01$ 。置  $\text{TIMx\_CCER}$  寄存器中 CC1P 和 CC1NP 为 00 以确定极性(只检测上升沿)。

- 置  $\text{TIMx\_SMCR}$  寄存器中 SMS=100，配置定时器为复位模式；置  $\text{TIMx\_SMCR}$  寄存器中 TS=101，选择  $\text{TI1}$  作为输入源。

- 置  $\text{TIMx\_CR1}$  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  $\text{TI1}$  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( $\text{TIMx\_SR}$  寄存器中的 TIF 位)被设置，根据  $\text{TIMx\_DIER}$  寄存器中 TIE(中断使能)位、的设置，产生一个中断请求。

下图显示当自动重装载寄存器  $\text{TIMx\_ARR}=0x36$  时的动作。在  $\text{TI1}$  上升沿和计数器的实际复位之间的延时取决于  $\text{TI1}$  输入端的重同步电路。

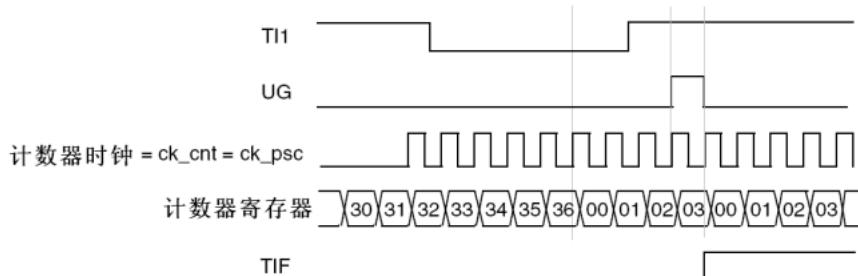


图 15-20 复位模式下的控制电路

### 15.2.11.2. 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽(本例中，不需要滤波，所以保持 IC1F=0000)。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P 和 CC1NP 为 10 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。

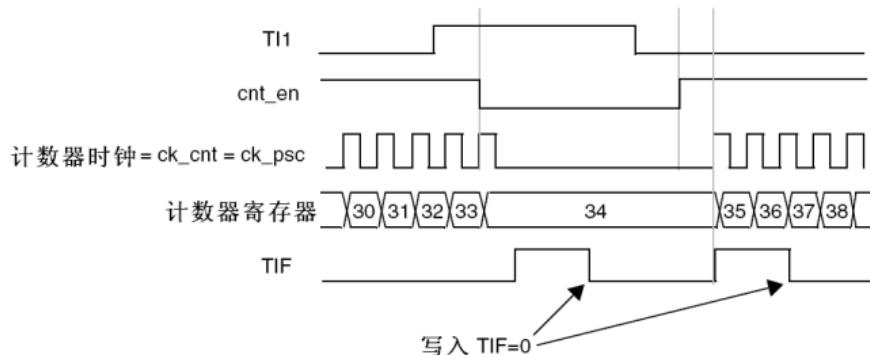


图 15-21 门控模式下的控制电路

### 15.2.11.3. 从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽(本例中，不需要任何滤波器，保持 IC2F=0000)。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC2S=01。置 TIMx\_CCER 寄存器中 CC2P 和 CC2NP 为 10 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。

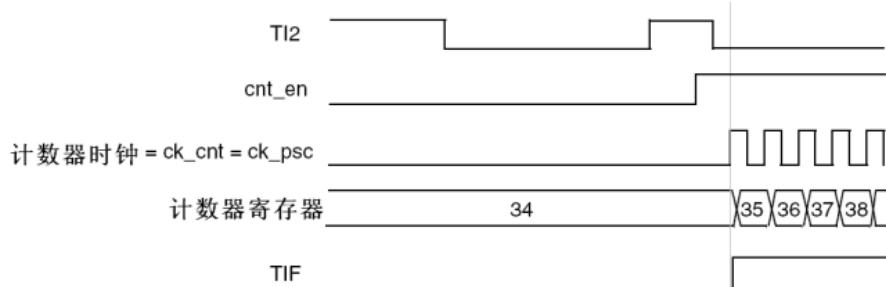


图 15-22 触发器模式下的控制电路

### 15.2.12. 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。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。

TIM9 和 TIM12 只能作为从定时器。

下图显示了触发选择和主模式选择模块的概况。

#### 15.2.12.1. 使用一个定时器作为另一个定时器的预分频器

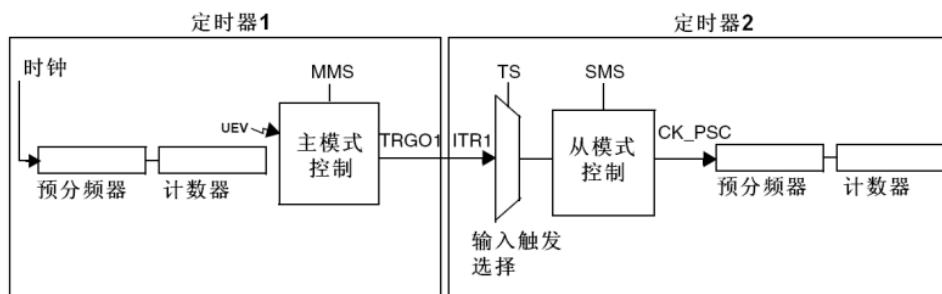


图 15-23 主/从定时器的例子

- 配置定时器 1 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIM1\_CR2 寄存器的 MMS='010' 时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。

- 连接定时器 1 的 TRGO1 输出至定时器 2，设置 TIM2\_SMCR 寄存器的 TS='000'，配置定时器 2 为使用 ITR1 作为内部触发的从模式。

- 然后把从模式控制器置于外部时钟模式 1(TIM2\_SMCR 寄存器的 SMS=111)；这样定时器 2 即可由定时器 1 周期性的上升沿(即定时器 1 的计数器溢出)信号驱动。

- 最后，必须设置相应(TIMx\_CR1 寄存器)的 CEN 位分别启动两个定时器。

注：如果 OCx 已被选中为定时器 1 的触发输出(MMS=1xx)，它的上升沿用于驱动定时器 2 的计数器。

#### 15.2.12.2. 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。只当定时器 1 的 OC1REF 为高时，定时器 2 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以 3(fCK\_CNT=fCK\_INT/3)得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号(OC1REF)为触发输出(TIM1\_CR2 寄存器的 MMS=100)
- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM2\_CR1 寄存器的 CEN=1 以使能定时器 2
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1

注：定时器 2 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 2 计数器的使能信号。

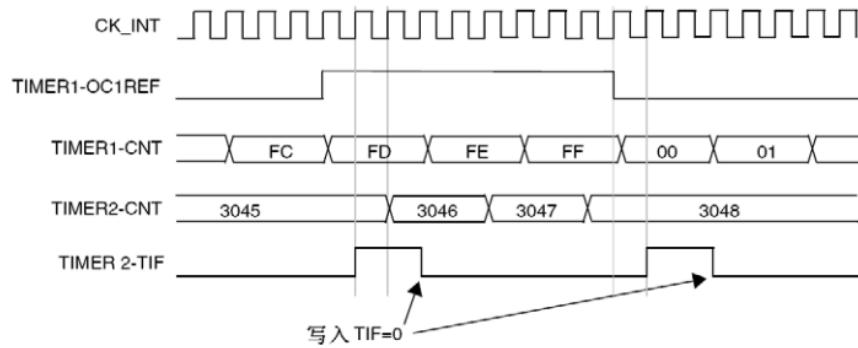


图 15-24 定时器 1 的 OC1REF 控制定时器 2

在定时器 2 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx\_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 2。定时器 1 是主模式并从 0 开始，定时器 2 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写‘0’到 TIM1\_CR1 的 CEN 位将禁止定时器 1，定时器 2 随即停止。

- 配置定时器 1 为主模式，送出输出比较 1 参考信号(OC1REF)做为触发输出(TIM1\_CR2 寄存器的 MMS=100)。
- 配置定时器 1 的 OC1REF 波形(TIM1\_CCMR1 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为门控模式(TIM2\_SMCR 寄存器的 SMS=101)
- 置 TIM1\_EGR 寄存器的 UG='1'，复位定时器 1。
- 置 TIM2\_EGR 寄存器的 UG='1'，复位定时器 2。
- 写‘0XE7’至定时器 2 的计数器(TIM2\_CNTL)，初始化它为 0xE7。
- 置 TIM2\_CR1 寄存器的 CEN='1'以使能定时器 2。
- 置 TIM1\_CR1 寄存器的 CEN='1'以启动定时器 1。
- 置 TIM1\_CR1 寄存器的 CEN='0'以停止定时器 1。

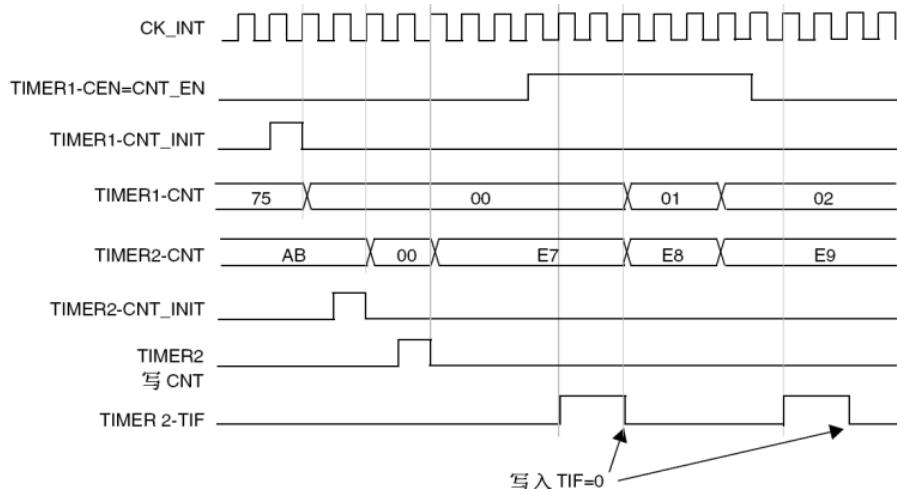


图 15-25 通过使能定时器 1 可以控制定时器 2

### 15.2.12.3. 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。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(TIM1\_CR2 寄存器的 MMS=010)。
- 配置定时器 1 的周期(TIM1\_ARR 寄存器)。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)
- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1。

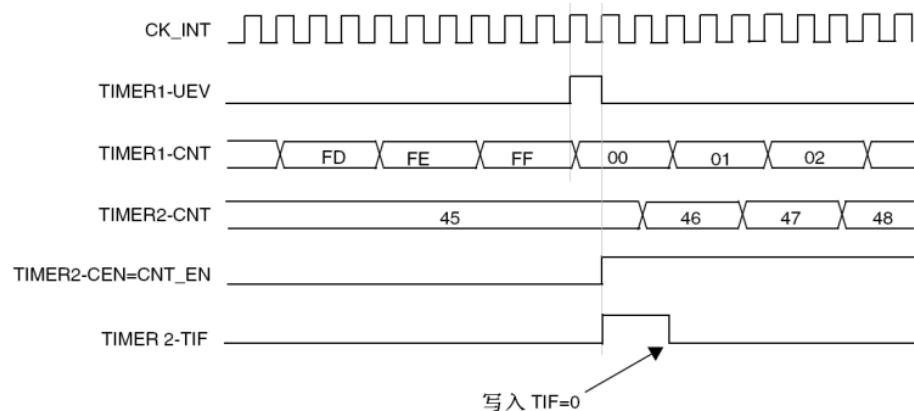


图 15-26 使用定时器 1 的更新触发定时器 2

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。

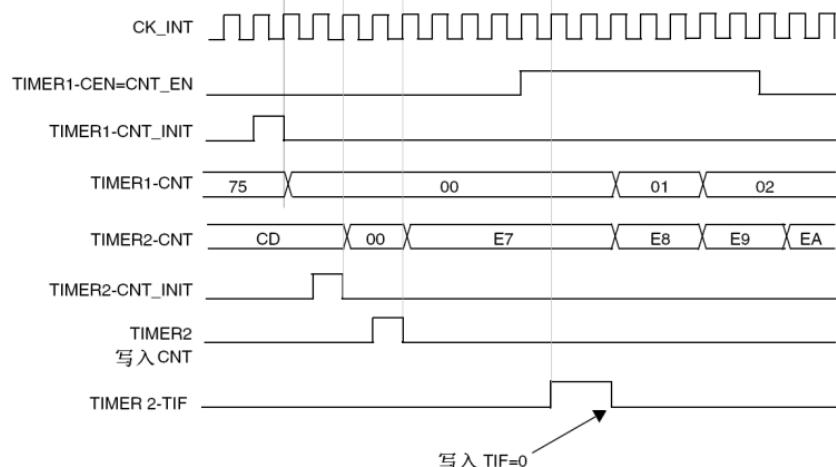


图 15-27 利用定时器 1 的使能触发定时器 2

#### 15.2.12.4. 使用一个定时器作为另一个的预分频器

这个例子使用定时器 1 作为定时器 2 的预分频器。配置如下：

- 配置定时器 1 为主模式，送出它的更新事件 UEV 做为触发输出(TIM1\_CR2 寄存器的 MMS='010')。然后每次计数器溢出时输出一个周期信号；
- 配置定时器 1 的周期(TIM1\_ARR 寄存器)；
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)；
- 配置定时器 2 使用外部时钟模式(TIM2\_SMCR 寄存器的 SMS=111)；
- 置 TIM1\_CR2 寄存器的 CEN=1 以启动定时器 2；

- 置 TIM1\_CR1 寄存器的 CEN=1 以启动定时器 1。

### 15.2.12.5. 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。保证计数器的对齐，定时器 1 必须配置为主/从模式(对应 TI1 为从，对应定时器 2 为主)：

- 配置定时器 1 为主模式，送出它的使能做为触发输出(TIM1\_CR2 寄存器的 MMS=001)。
- 配置定时器 1 为从模式，从 TI1 获得输入触发(TIM1\_SMCR 寄存器的 TS=100)。
- 配置定时器 1 为触发模式(TIM1\_SMCR 寄存器的 SMS=110)。
- 配置定时器 1 为主/从模式， TIM1\_SMCR 寄存器的 MSM=1。
- 配置定时器 2 从定时器 1 获得输入触发(TIM2\_SMCR 寄存器的 TS=000)
- 配置定时器 2 为触发模式(TIM2\_SMCR 寄存器的 SMS=110)。

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注：在这个例子中，在启动之前两个定时器都被初始化(设置相应的 UG 位)，两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器(TIMx\_CNT)在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT\_EN 和 CK\_PSC 之间有个延迟。

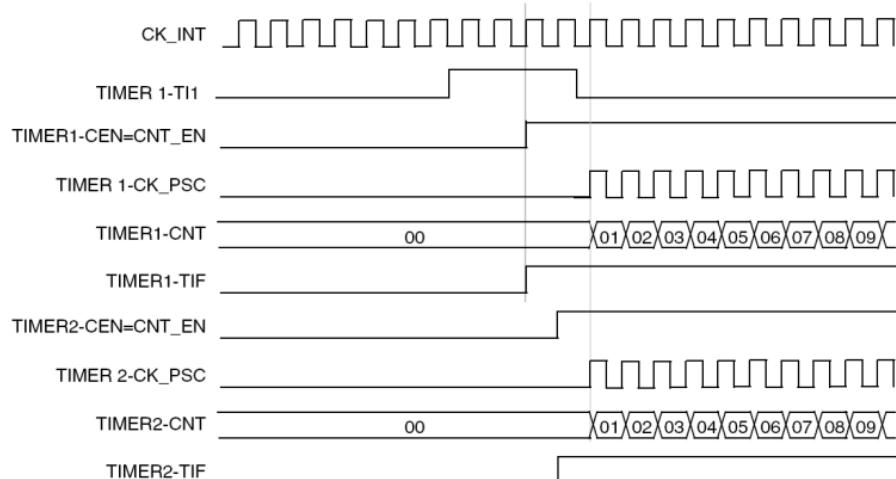


图 15-28 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

### 15.2.13. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止)，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。

## 15.3. 寄存器描述

TIM9 寄存器地址： 0x4001 4C00 - 0x4001 4FFF

TIM12 寄存器地址： 0x4000 1800 - 0x4000 1BFF

### 15.3.1. TIM9 和 TIM12 控制寄存器 1 (TIMx\_CR1)

Address offset:0x00

Reset value:0x0000

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

保留	CKD [1:0]		ARPE	保留	OPM	URS	UDIS	CEN
保留	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	保留, 始终为 0			
9:8	CKD	RW	0	<p>时钟分频因子 (Clock division) 这 2 位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟 (tDTS) 之间的分频比例。</p> <p>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</p>
7	ARPE	RW	0	<p>自动重装载预装载允许位 (Auto-reload preload enable)</p> <p>0: TIMx_ARR 寄存器没有缓冲; 1: TIMx_ARR 寄存器被装入缓冲器。</p>
6:4	保留, 始终为 0			
3	OPM	RW	0	<p>单脉冲模式 (One pulse mode)</p> <p>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。</p>
2	URS	RW	0	<p>更新请求源 (Update request source) 软件通过该位选择 UEV 事件的源</p> <p>0: 如果使能了更新中断请求, 则下述任一事件产生更新中断请求:</p> <ul style="list-style-type: none"> <li>- 计数器溢出</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果使能了更新中断请求, 则只有计数器溢出才产生更新中断请求。</p>
1	UDIS	RW	0	<p>禁止更新 (Update disable) 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:  <ul style="list-style-type: none"> <li>- 计数器溢出</li> <li>- 设置 UG 位</li> <li>- 从模式控制器产生的更新</li> </ul> 具有缓存的寄存器被装入它们的预装载值。 (译注: 更新影子寄存器)  1: 禁止 UEV。不产生更新事件, 影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。</p>
0	CEN	RW	0	<p>使能计数器 (Counter enable)</p> <p>0: 禁止计数器; 1: 使能计数器。</p> <p>注: 在软件设置了 CEN 位后, 外部时钟、门控模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。</p>

### 15.3.2. TIM9 和 TIM12 从模式控制寄存器 (TIMx\_SMCR)

Address offset:0x08

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:7	保留, 始终读为 0。			
6:4	TS	RW	0	<p>触发选择 (Trigger selection) 这 3 位选择用于同步计数器的触发输入。</p> <p>000: 内部触发 0(ITR0)      100: TI1 的边沿检测器(TI1F_ED)      001: 内部触发 1(ITR1)      101: 滤波后的定时器输入 1(TI1FP1)      010: 内部触发 2(ITR2)      110: 滤波后的定时器输入 2(TI2FP2)      011: 内部触发 3(ITR3)      111: 保留      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。</p>
3	保留, 始终读为 0。			
2:0	SMS	RW	0	<p>从模式选择 (Slave mode selection) 当选择了外部信号, 触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明)</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。      001: Reserved      010: Reserved      011: Reserved      100: 复位模式 - 选中的触发输入(TRGI)的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。      101: 门控模式 - 当触发输入(TRGI)为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止(但不复位)。计数器的启动和停止都是受控的。      110: 触发模式 - 计数器在触发输入 TRGI 的上升沿启动(但不复位), 只有计数器的启动是受控的。      111: 外部时钟模式 1 - 选中的触发输入(TRGI)的上升沿驱动计数器。      注: 如果 TI1F_EN 被选为触发输入(TS=100)时, 不要使用门控模式。这是因为, TI1F_ED 在每次 TI1F 变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。</p>

表 15-1 TIMx 内部触发连接

从定时器	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM9	TIM2_TRGO	TIM3_TRGO	TIM10_OC	TIM11_OC
TIM12	TIM4_TRGO	TIM5_TRGO	TIM13_OC	TIM14_OC

### 15.3.3. TIM9 和 TIM12 中断使能寄存器 (TIMx\_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TIE	保留				CC2IE	CC1IE	UIE
保留								RW	保留				RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:7	保留, 始终为 0			
6	TIE	RW	0	<p>触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断; 1: 使能触发中断。</p>
5:3	保留, 始终为 0			
2	CC2IE	RW	0	<p>允许捕获/比较 2 中断 (Capture/Compare 2 interrupt enable) 0: 禁止捕获/比较 2 中断; 1: 允许捕获/比较 2 中断。</p>
1	CC1IE	RW	0	<p>允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0: 禁止捕获/比较 1 中断; 1: 允许捕获/比较 1 中断</p>
0	UIE	RW	0	<p>允许更新中断 (Update interrupt enable) 0: 禁止更新中断;</p>

Bit	Name	R/W	Reset Value	Function
				1: 允许更新中断。

### 15.3.4. TIM9 和 TIM12 状态寄存器 (TIMx\_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CC2OF	CC1OF	保留		TIF	保留			CC2IF	CC1IF	UIF
保留					RC_W0	RC_W0	保留		RC_W0	保留			RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15:11	保留, 始终为 0			
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, 始终读为 0.			
6	TIF	RC_W0	0	触发器中断标记 (Trigger interrupt flag) 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时, 在 TRGI 输入端检测到有效边沿, 或门控模式下的任一边沿)时由硬件对该位置'1'。它由软件清'0'。 0: 无触发器事件产生; 1: 触发中断等待响应。
5:3	保留, 始终为 0			
2	CC2IF	RC_W0	0	捕获/比较 2 中断标记 (Capture/Compare 2 interrupt flag) 参考 CC1IF 描述。
1	CC1IF	RC_W0	0	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式: 当计数器值与比较值匹配时该位由硬件置 1。它由软件清'0'。 0: 无匹配发生; 1: TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APB 的内容时, 在计数器溢出条件下, CC1IF 位变高 如果通道 CC1 配置为输入模式: 当捕获事件发生时该位由硬件置'1', 它由软件清'0'或通过读 TIMx_CCR1 清'0'。 0: 无输入捕获产生; 1: 计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。
0	UIF	RC_W0	0	更新中断标记 (Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0: 无更新事件产生; 1: 更新中断等待响应。当寄存器被更新时该位由硬件置'1': - 若 TIMx_CR1 寄存器的 UDIS=0, 当重复计数器数值上溢时置起。 - 若 TIMx_CR1 寄存器的 URS=0、UDIS=0, 当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件, 通过软件对计数器 CNT 重新初始化时。 - 若 TIMx_CR1 寄存器的 URS=0、UDIS=0, 当计数器 CNT 被触发事件 (参考 SMCR 寄存器描述) 重新初始化时。

### 15.3.5. TIM9 和 TIM12 事件产生寄存器 (TIMx\_EGR)

Address offset:0x14

Reset value:0x0000

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

保留	TG	保留	CC2G	CC1G	UG
保留	W	保留	W	W	W

Bit	Name	R/W	Reset Value	Function
15:7	保留, 始终为 0			
6	TG	W	0	<p>产生触发事件 (Trigger generation) 该位由软件置'1', 用于产生一个触发事件, 由硬件自动清'0'。 0: 无动作; 1: TIMx_SR 寄存器的 TIF=1, 若开启对应的中断, 则产生相应的中断。</p>
5:3	保留, 始终为 0			
2	CC2G	W	0	<p>产生捕获/比较 2 事件 (Capture/Compare 2 generation) 参考 CC1G 描述。</p>
1	CC1G	W	0	<p>产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置'1', 用于产生一个捕获/比较事件, 由硬件自动清'0'。 0: 无动作; 1: 在通道 CC1 上产生一个捕获/比较事件: 若通道 CC1 配置为输出: 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。 若通道 CC1 配置为输入: 当前的计数器值被捕获至 TIMx_CCR1 寄存器; 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。 若 CC1IF 已经为 1, 则设置 CC1OF=1。</p>
0	UG	W	0	<p>产生更新事件 (Update generation) 该位由软件置'1', 由硬件自动清'0'。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。</p>

### 15.3.6. TIM9 和 TIM12 捕获/比较模式控制寄存器 1 (TIMx\_CCMR1)

Address offset:0x018

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M [2:0]			OC2PE	OC2FE			Res.	OC1M [2:0]			OC1PE	OC1FE		
	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

#### 15.3.6.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15	Res.			
14:12	OC2M	RW	0	输出比较 2 模式 (Output Compare 2 mode)
11	OC2PE	RW	0	输出比较 2 预装载使能 (Output Compare 2 preload enable)
10	OC2FE	RW	0	输出比较 2 快速使能 (Output Compare 2 fast enable)
9:8	CC2S	RW	0	<p>捕获/比较 2 选择。 (Capture/Compare 2 selection) 该位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC2 通道被配置为输出; 01: CC2 通道被配置为输入, IC2 映射在 TI2 上; 10: CC2 通道被配置为输入, IC2 映射在 TI1 上; 11: CC2 通道被配置为输入, IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。</p>

Bit	Name	R/W	Reset Value	Function
				注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。
7	Res.			
6:4	OC1M	RW	0	<p>输出比较 1 模式 (Output Compare 1 mode)      该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。      000：冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用；      001：匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为高。      010：匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时，强制 OC1REF 为低。      011：翻转。当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。      100：强制为无效电平。强制 OC1REF 为低。      101：强制为有效电平。强制 OC1REF 为高。      110：PWM 模式 1 - TIMx_CNT&lt;TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。      111：PWM 模式 2 - TIMx_CNT&lt;TIMx_CCR1 时通道 1 为无效电平，否则为有效电平。      注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。</p>
3	OC1PE	RW	0	<p>输出比较 1 预装载使能 (Output Compare 1 preload enable)      0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。      1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。      注：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM=1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。</p>
2	OC1FE	RW	0	<p>输出比较 1 快速使能 (Output Compare 1 fast enable)      该位用于加快 CC 输出对触发输入事件的响应。      0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。      1：输入到触发器的有效沿的作用就象发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。      OC1FE 只在通道被配置成 PWM1 或 PWM2 模式时起作用。</p>
1:0	CC1S	RW	0	<p>捕获/比较 1 选择。 (Capture/Compare 1 selection)      这 2 位定义通道的方向(输入/输出)，及输入脚的选择：      00：CC1 通道被配置为输出；      01：CC1 通道被配置为输入，IC1 映射在 TI1 上；      10：CC1 通道被配置为输入，IC1 映射在 TI2 上；      11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。      注：CC1S 仅在通道关闭时(TIMx_CCER 寄存器的 CC1E=0)才是可写的。</p>

### 15.3.6.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	输入捕获 2 滤波器 (Input capture 2 filter)
11:10	IC2PSC	RW	0	输入/捕获 2 预分频器 (Input capture 2 prescaler)
9:8	CC2S	RW	0	<p>捕获/比较 2 选择 (Capture/Compare 2 selection)      这 2 位定义通道的方向(输入/输出)，及输入脚的选择：      00：CC2 通道被配置为输出；      01：CC2 通道被配置为输入，IC2 映射在 TI2 上；      10：CC2 通道被配置为输入，IC2 映射在 TI1 上；      11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时(由 TIMx_SMCR 寄存器的 TS 位选择)。      注：CC2S 仅在通道关闭时(TIMx_CCER 寄存器的 CC2E=0)才是可写的。</p>
7:4	IC1F	RW	0	<p>输入捕获 1 滤波器 (Input capture 1 filter)      这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记</p>

Bit	Name	R/W	Reset Value	Function
				<p>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                    1000: 采样频率 fSAMPLING=fDTS/8, N=6      0001: 采样频率 fSAMPLING=fCK_INT, N=2    1001: 采样频率 fSAMPLING=fDTS/8, N=8      0010: 采样频率 fSAMPLING=fCK_INT, N=4    1010: 采样频率 fSAMPLING=fDTS/16, N=5      0011: 采样频率 fSAMPLING=fCK_INT, N=8    1011: 采样频率 fSAMPLING=fDTS/16, N=6      0100: 采样频率 fSAMPLING=fDTS/2, N=6    1100: 采样频率 fSAMPLING=fDTS/16, N=8      0101: 采样频率 fSAMPLING=fDTS/2, N=8    1101: 采样频率 fSAMPLING=fDTS/32, N=5      0110: 采样频率 fSAMPLING=fDTS/4, N=6    1110: 采样频率 fSAMPLING=fDTS/32, N=6      0111: 采样频率 fSAMPLING=fDTS/4, N=8    1111: 采样频率 fSAMPLING=fDTS/32, N=8</p>
3:2	IC1PSC	RW	0	<p>输入/捕获 1 预分频器 (Input capture 1 prescaler)</p> <p>这 2 位定义了 CC1 输入(IC1)的预分频系数。</p> <p>一旦 CC1E=0(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	RW	0	<p>捕获/比较 1 选择 (Capture/Compare 1 Selection)</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>

### 15.3.7. TIM9 和 TIM12 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CC2NP	保留	CC2P	CC2E	CC1NP	保留	CC1P	CC1E
保留								RW	保留	RW	RW	RW	保留	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
7	CC2NP	RW	0	<p>输入/捕获 2 互补输出极性 (Capture/Compare 2 output polarity)</p> <p>参考 CC1P 的描述。</p>
6	保留, 始终为 0			
5	CC2P	RW	0	<p>输入/捕获 2 输出极性 (Capture/Compare 2 output polarity)</p> <p>参考 CC1P 的描述。</p>
4	CC2E	RW	0	<p>输入/捕获 2 输出使能 (Capture/Compare 2 output enable)</p> <p>参考 CC1E 的描述。</p>
3	CC1NP	RW	0	<p>输入/捕获 1 互补输出极性 (Capture/Compare 1 complementary output polarity)</p> <p>CC1 通道配置为输出时: CC1NP 必须保持为 0;</p> <p>CC1 通道配置为输入时: CC1NP 被用来与 CC1P 共同作用 TIxFP1 的极性 (参考 CC1P 描述)</p>
2	保留, 始终为 0			
1	CC1P	RW	0	<p>输入/捕获 1 输出极性 (Capture/Compare 1 output polarity)</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>

Bit	Name	R/W	Reset Value	Function
				00: 不反相/上升沿: TlxFP1 上升沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ; TlxFP1 不反相 (门控模式)。 01: 反相/下降沿: TlxFP1 下降沿有效 (捕获、复位模式下触发、外部时钟或触发模式下) ; TlxFP1 反相 (门控模式)。 10: 保留, 不要使用这个配置。 11: 不反相/双沿 TlxFP1 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) ; TlxFP1 不反相 (门控模式)。
0	CC1E	RW	0	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出: 0: 关闭 - OC1 禁止输出。 1: 开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入: 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0: 捕获禁止; 1: 捕获使能。

表 15-2 输出通道 OCx 和的控制位

CCxE 位	OCx 输出状态
0	输出禁止 (与定时器断开) OCx=0, OCx_EN=0
1	OCx=OCxREF+Polarity, OCx_EN=1

注：引脚连接到互补的 OCx 通道的外部 I/O 引脚的状态，取决于 OCx 通道状态和 GPIO 寄存器。

### 15.3.8. TIM9 和 TIM12 计数器 (TIMx\_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	计数器的值 (Counter value)

### 15.3.9. TIM9 和 TIM12 预分频器 (TIMx\_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	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)。

Bit	Name	R/W	Reset Value	Function
				PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

### 15.3.10. TIM9 和 TIM12 自动重装载寄存器 (TIMx\_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。

### 15.3.11. TIM9 和 TIM12 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	捕获/比较通道 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。 否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

### 15.3.12. TIM9 和 TIM12 捕获/比较寄存器 2 (TIMx\_CCR2)

Address offset:0x38

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
RW															

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW	0	捕获/比较通道 2 的值 (Capture/Compare 2 value) 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值(预装载值)。 如果在 TIMx_CCMR2 寄存器(OC2PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。 否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件(IC2)传输的计数器值。

### 15.3.13. TIM9 和 TIM12 寄存器映射

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	0	rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0
0x000	TIMx_C_R1	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x008	TIMx_S_MCR	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x00C	TIMx_DI_ER	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x100	TIMx_SR	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x110	TIMx_EGR	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x114	TIMx_CM_R1	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
0x118	TIMx_CM_R1	Reserved																					rw	CKD	rw	ARPE	0	rw	Reserved	rw	TS	rw	0	rw	Reserved	rw	6	5	4	3	2	1	0		
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G		CC1G		CC1IF		CC2IE		CC1IE		UIF		URS		OPM		UDIS		CEN			
		Reserved		OC2M		OC1M		OC1S		OC1PE		OC1FE		OC1PSC		IC2F		IC1F		IC2S		IC1S		CC2G																					



# 16. 通用定时器 (TIM10/TIM11/TIM13/TIM14)

## 16.1. 简介

通用定时器(TIM10/TIM11/TIM13/TIM14, 以下简称 TIMx)是由一个 16 位的自动装载计数器组成，计数器由一个可编程的预分频器驱动。

它适合多种用途，包含测量输入信号的脉冲宽度(输入捕获)，或者产生输出波形(输出比较、PWM)。使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

通用定时器(TIMx)是完全独立的，它们不共享任何资源。它们可以同步操作。

### 16.1.1. TIMx 主要特征

TIMx 定时器的功能包括：

- 16 位向上自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 2 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘模式)
  - 单脉冲模式输出
- 通过外部信号来控制定时器与定时器之间互联的同步电路
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出，计数器初始化(通过软件或者内部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部触发计数)
  - 输入捕获
  - 输出比较

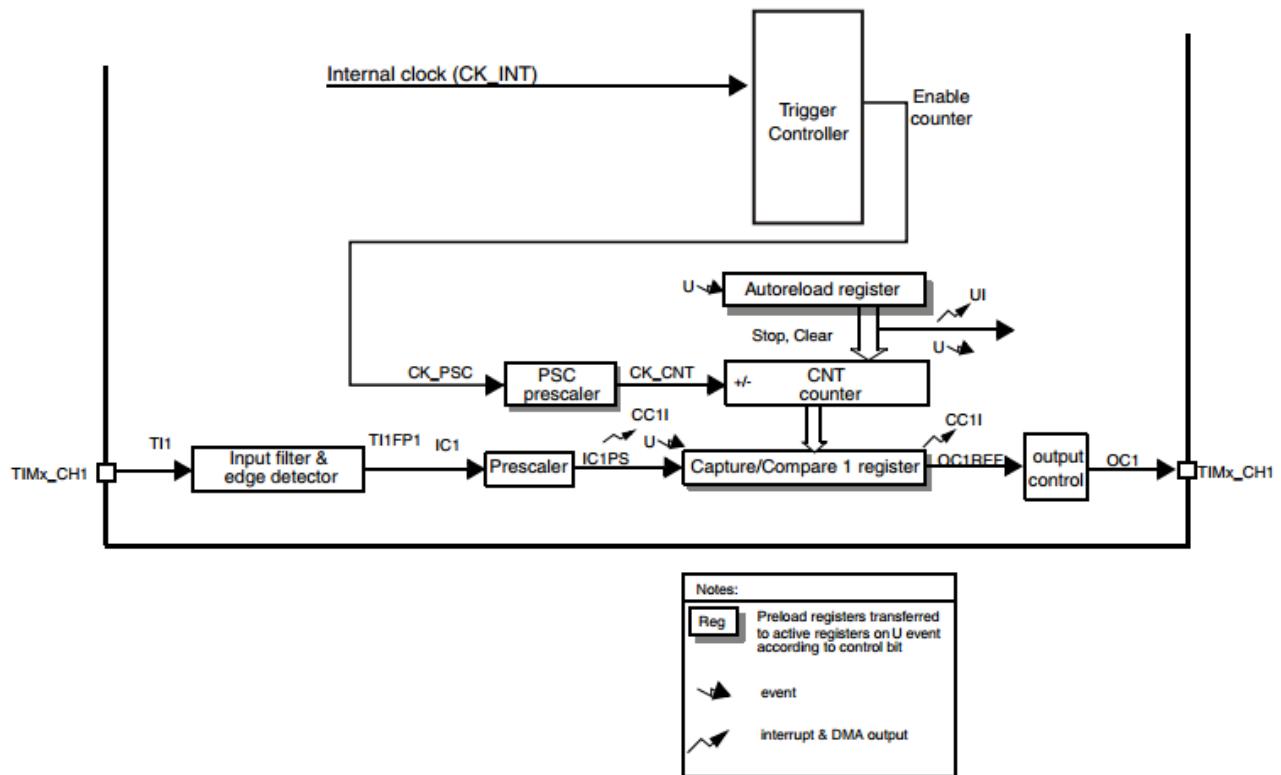


图 16-1 TIMx(TIM10/TIM11/TIM13/TIM14)模块

## 16.2. TIMx 功能描述

### 16.2.1. 时基单元

通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器向上计数。

计数器的时钟可以被预分频器分频。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)
- 自动装载寄存器 (TIMx\_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问它的预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 (UEV) 时传送到影子寄存器。当计数器达到上溢条件并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件及其他条件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK\_CNT 驱动，仅当设置了 TIMx\_CR1 寄存器中的计数器使能位(CEN)，CK\_CNT 才对计数器有效。

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。

### 16.2.1.1. 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx\_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频参数将在下一次更新事件到来时被采用。

在预分频器运行时，更改计数器参数的例子如下所示。

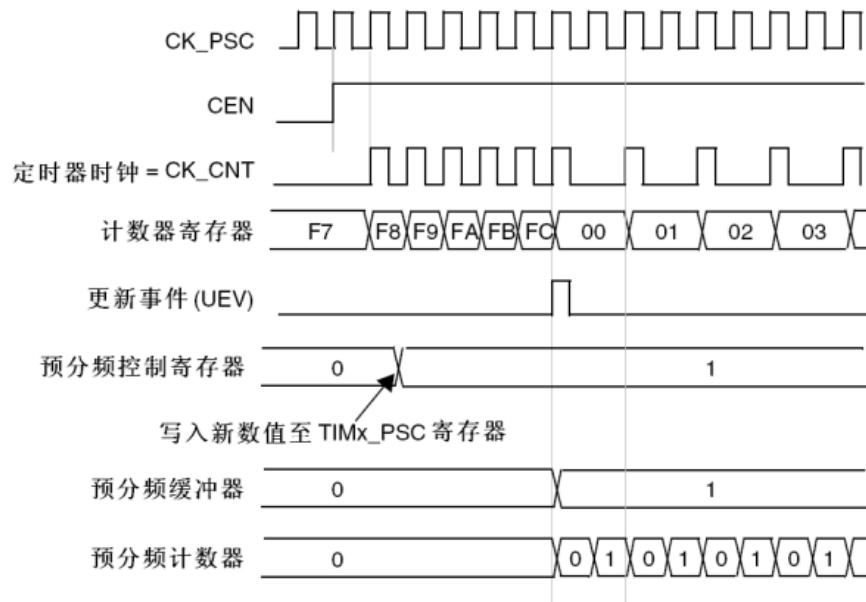


图 16-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

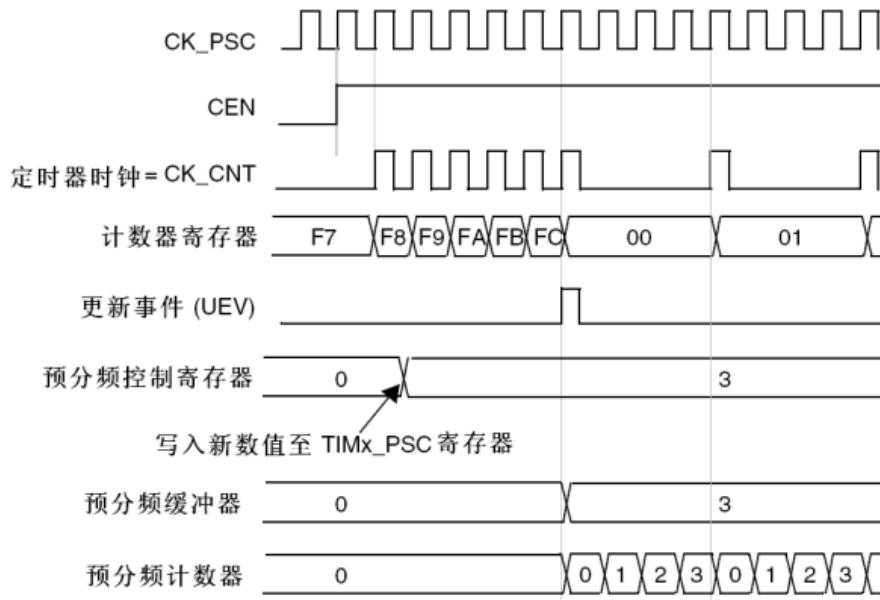


图 16-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 16.2.2. 计数器模式

#### 16.2.2.1. 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值(TIMx\_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

而在 TIMx\_EGR 寄存器中(通过软件方式)设置 UG 位也同样可以产生一个更新事件。

通过设置  $\text{TIMx\_CR1}$  寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了  $\text{TIMx\_CR1}$  寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断请求)。这是为了避免在捕获事件时清除计数器，同时产生更新和捕获中断。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位( $\text{TIMx\_SR}$  寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值( $\text{TIMx\_ARR}$ )。
- 预分频器的缓冲区被置入预装载寄存器的值( $\text{TIMx\_PSC}$  寄存器的内容)。

下图给出一些例子，当  $\text{TIMx\_ARR}=0x36$  时计数器在不同时钟频率下的动作。

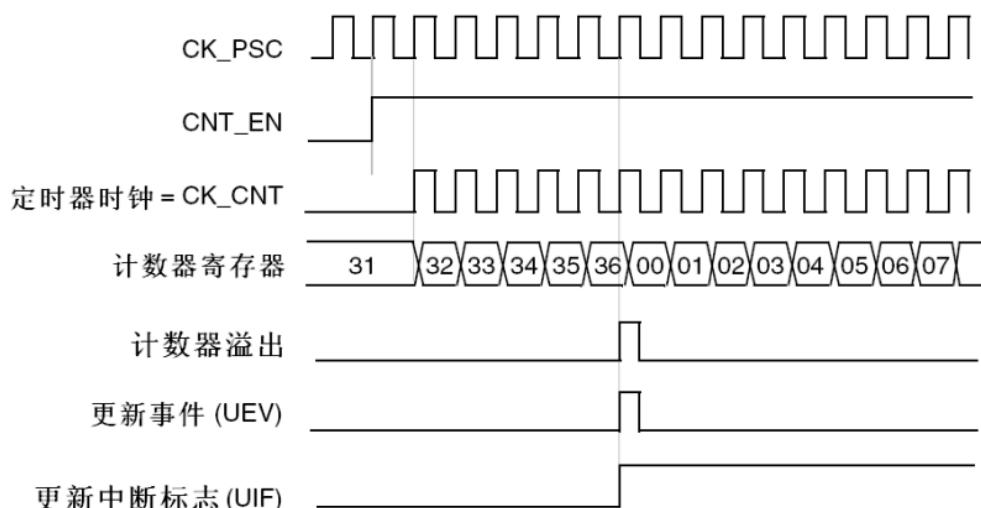


图 16-4 计数器时序图，内部时钟分频因子为 1

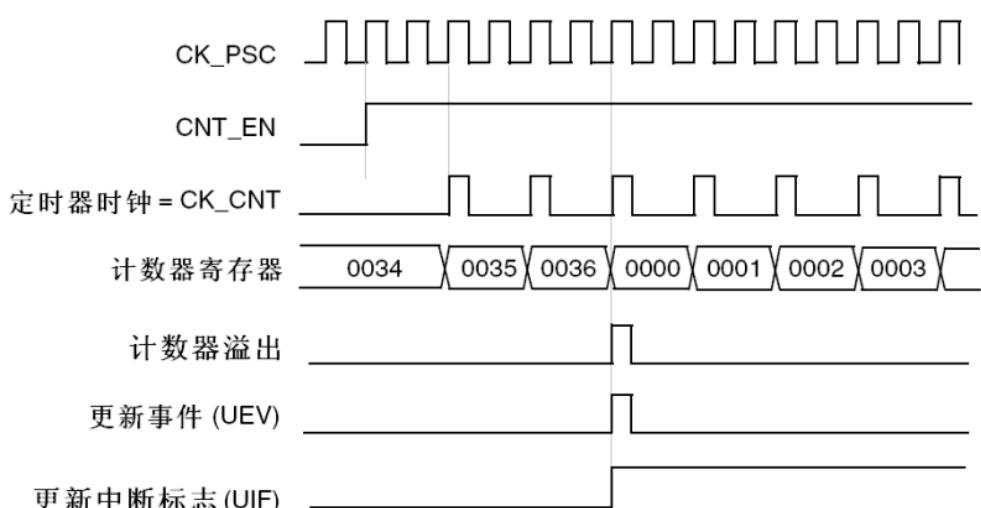


图 16-5 计数器时序图，内部时钟分频因子为 2

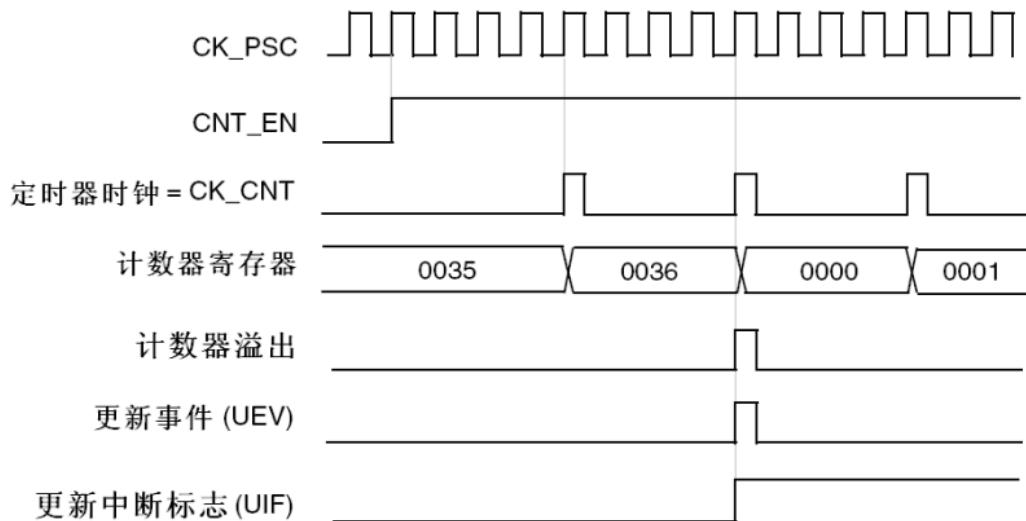


图 16-6 计数器时序图，内部时钟分频因子为 4

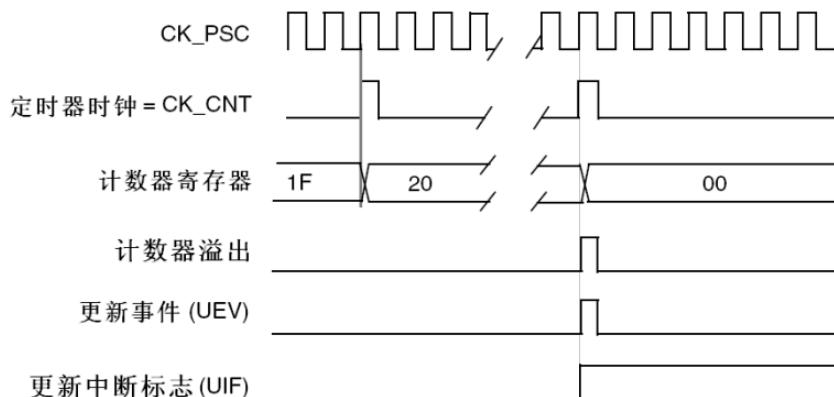


图 16-7 计数器时序图，内部时钟分频因子为 N

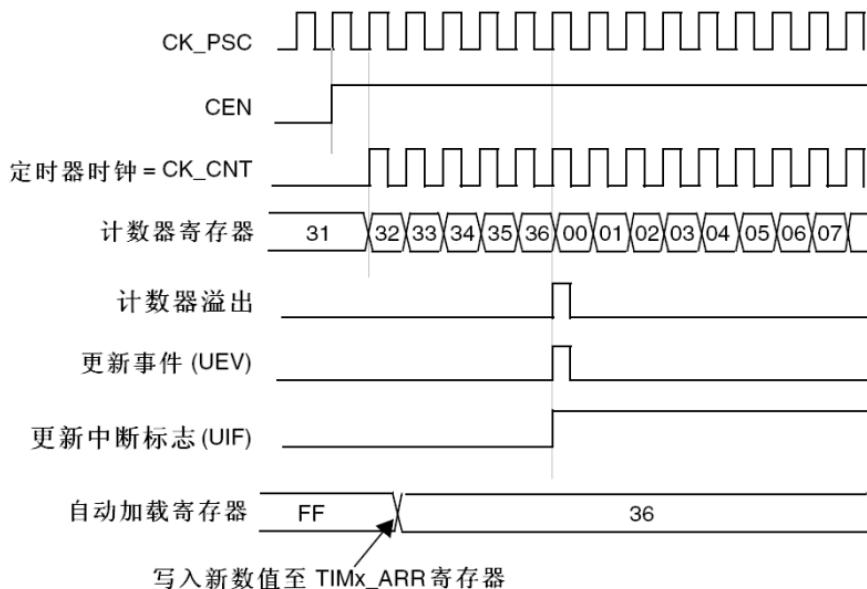


图 16-8 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx\_ARR)

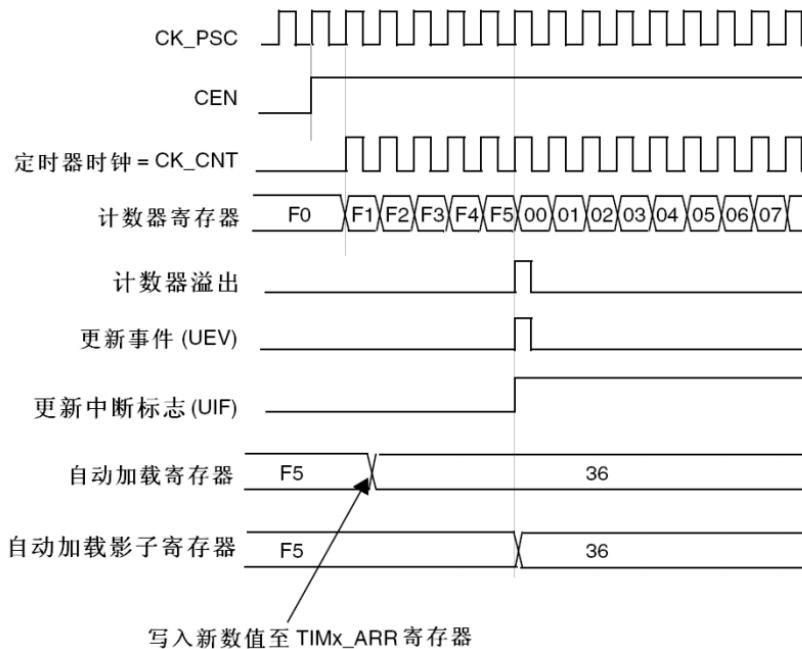


图 16-9 计数器时序图，当 ARPE=1 时的更新事件(预装了 TIMx\_ARR)

### 16.2.3. 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟(CK\_INT)

#### 16.2.3.1. 内部时钟源(CK\_INT)

对于 TIM10/TIM11/TIM13/TIM14，内部时钟源时默认时钟，则 CEN 和 UG 位(TIMx\_EGR 寄存器)是事实上的控制位，并且只能被软件修改(UG 位仍被自动清除)。只要 CEN 位被写成‘1’，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

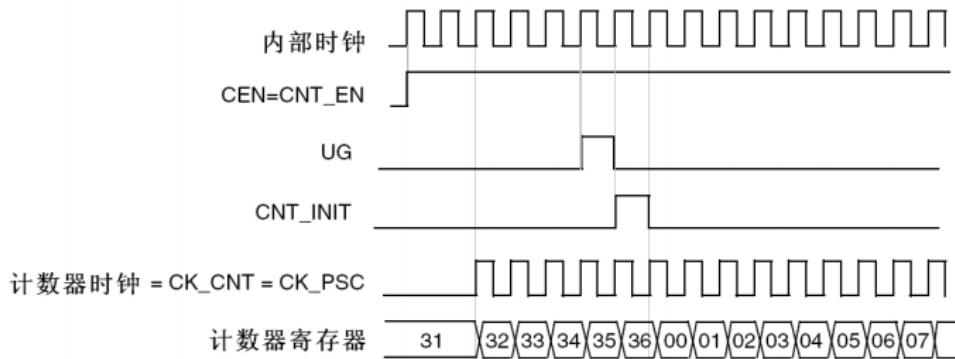


图 16-10 一般模式下的控制电路，内部时钟分频因子为 1

### 16.2.4. 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器(包含影子寄存器)，包括捕获的输入部分(数字滤波、多路复用和预分频器)，和输出部分(比较器和输出控制)。

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)。在捕获寄存器前分频得到 ICxPS。

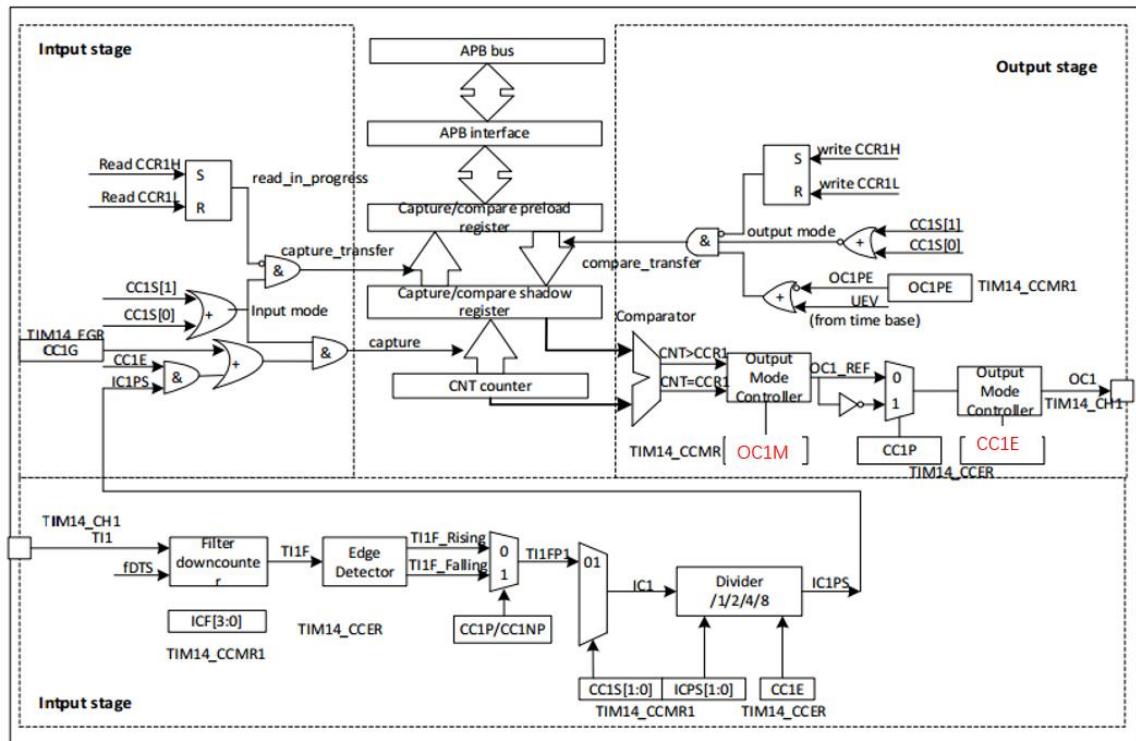


图 16-11 捕获/比较通道

输出部分产生一个中间波形 OCxRef(高有效)作为基准，链的末端决定最终输出信号的极性。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。

### 16.2.5. 输入捕获模式

在输入捕获模式下，当检测到 IC<sub>x</sub> 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(TIM<sub>x</sub>\_CCR<sub>x</sub>)中。当发生捕获事件时，相应的 CC<sub>x</sub>IF 标志(TIM<sub>x</sub>\_SR 寄存器)被置 1，如果开放了中断操作，则将产生中断请求。如果发生捕获事件时 CC<sub>x</sub>IF 标志已经为高，那么过捕获标志 CC<sub>x</sub>OF(TIM<sub>x</sub>\_SR 寄存器)被置 1。通过写 CC<sub>x</sub>IF=0 可清除 CC<sub>x</sub>IF，或读取存储在 TIM<sub>x</sub>\_CCR<sub>x</sub> 寄存器中的捕获数据也可清除 CC<sub>x</sub>IF。写 CC<sub>x</sub>OF=0 可清除 CC<sub>x</sub>OF。

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 TIM<sub>x</sub>\_CCMR1 寄存器中，步骤如下：

- 选择有效输入端： TIM<sub>x</sub>\_CCR1 必须连接到 TI1 输入，所以写入 TIM<sub>x</sub>\_CCR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道就被配置为输入，并且 TIM<sub>x</sub>\_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 TI<sub>x</sub> 时，输入滤波器控制位是 TIM<sub>x</sub>\_CCMR<sub>x</sub> 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 TIM<sub>x</sub>\_CCMR1 寄存器中写入 IC1F=0011。
- 选择 TI1 通道的有效转换边沿，在 TIM<sub>x</sub>\_CCER 寄存器中写入 CC1P 和 CC1NP 位写入 00 (设置为上升沿)。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写 TIM<sub>x</sub>\_CCMR1 寄存器的 IC1PS=00)。
- 设置 TIM<sub>x</sub>\_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
- 如果需要，可以通过设置 TIM<sub>x</sub>\_DIER 寄存器中的 CC1IE 位允许相关中断请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx\_CCR1 寄存器。
- CC1IF 标志位被设置(中断标志)。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置 1。
- 如设置了 CC1IE 位，则会产生一个中断。

为了处理过捕获，建议在过捕获标志被置起之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断请求。

### 16.2.6. 强置输出模式

在输出模式(TIMx\_CCMRx 寄存器中 CCxS=00)下，输出比较信号(OCxREF 和相应的 OCx/OCxN)能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx\_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号(OCxREF/OCx)为有效状态。这样 OCxREF 被强置为高电平(OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0(OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断请求。这将会在下面的输出比较模式一节中介绍。

### 16.2.7. 输出比较模式

此项功能是用来控制一个输出波形，或者表明一段给定的时间已经到时。

当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

• 将输出比较模式(TIMx\_CCMRx 寄存器中的 OCxM 位)和输出极性(TIMx\_CCER 寄存器中的 CCxP 位)定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无效电平(OCxM=010)或进行翻转(OCxM=011)。

- 设置中断状态寄存器中的标志位(TIMx\_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽(TIMx\_DIER 寄存器中的 CCxIE 位)，则产生一个中断。

可以通过配置 TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。同步的精度可以达到计数器的一个计数周期。输出比较模式(在单脉冲模式下)也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟(内部，外部，预分频器)。
2. 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
  - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011
  - 置 OCxPE = 0 禁用预装载寄存器
  - 置 CCxP = 0 选择极性为高电平有效
  - 置 CCxE = 1 使能输出
5. 设置 TIMx\_CR1 寄存器的 CEN 位启动计数器

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器 (OCxPE='0'，否则 TIMx\_CCRx 的影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

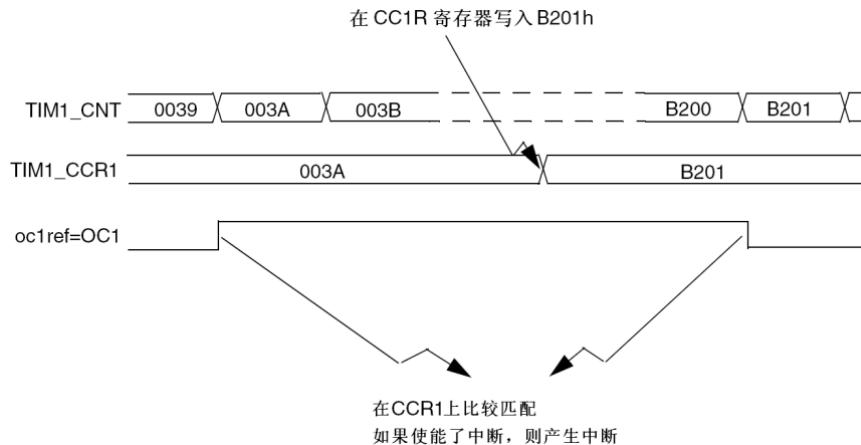


图 16-12 输出比较模式，翻转 OC1

### 16.2.8. PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx\_ARR 寄存器确定频率、由 TIMx\_CCRx 寄存器确定占空比的信号。

在 TIMx\_CCMRx 寄存器中的 OCxM 位写入‘110’(PWM 模式 1)或‘111’(PWM 模式 2)，能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx\_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx\_CR1 寄存器的 ARPE 位，(在向上计数或中心对称模式中)使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，用户必须通过设置 TIMx\_EGR 寄存器中的 UG 位来初始化所有的寄存器。

OCx 的极性可以通过软件在 TIMx\_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过(TIMx\_CCER 中)CCxE 位控制。详见 TIMx\_CCER 寄存器的描述。

在 PWM 模式(模式 1 或模式 2)下，TIMx\_CNT 和 TIMx\_CCRx 始终在进行比较，(依据计数器的计数方向)以确定是否符合 TIMx\_CCRx≤TIMx\_CNT 或者 TIMx\_CNT≤TIMx\_CCRx。

定时器能够产生边沿对齐的 PWM 信号。

#### 16.2.8.1. PWM 边沿对齐模式

- 向上计数配置

当 TIMx\_CR1 寄存器中的 DIR 位为低的时候执行向上计数。。

下面是一个 PWM 模式 1 的例子。当 TIMx\_CNT<TIMx\_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx\_CCRx 中的比较值大于自动重装载值(TIMx\_ARR)，则 OCxREF 保持为‘1’。如果比较值为 0，则 OCxREF 保持为‘0’。下图为 TIMx\_ARR=8 时边沿对齐的 PWM 波形实例。

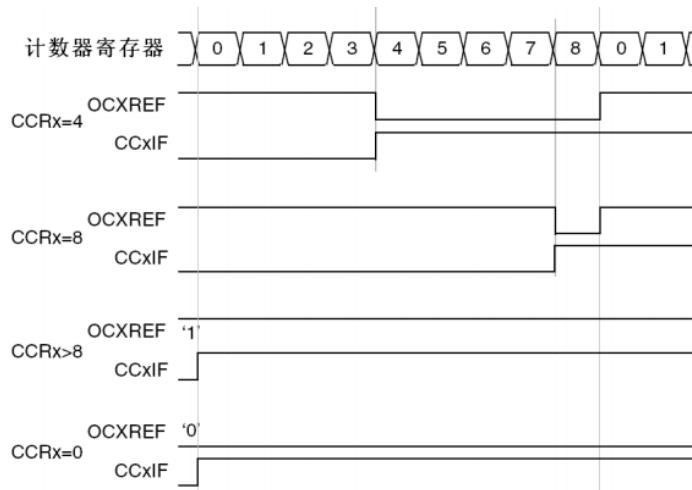


图 16-13 边沿对齐的 PWM 波形(ARR=8)

### 16.2.9. 单脉冲模式

单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx\_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前(当定时器正在等待触发)，必须如下配置：

- 向上计数方式：计数器  $CNT < CCRx \leq ARR$  (特别地,  $0 < CCRx$ )。

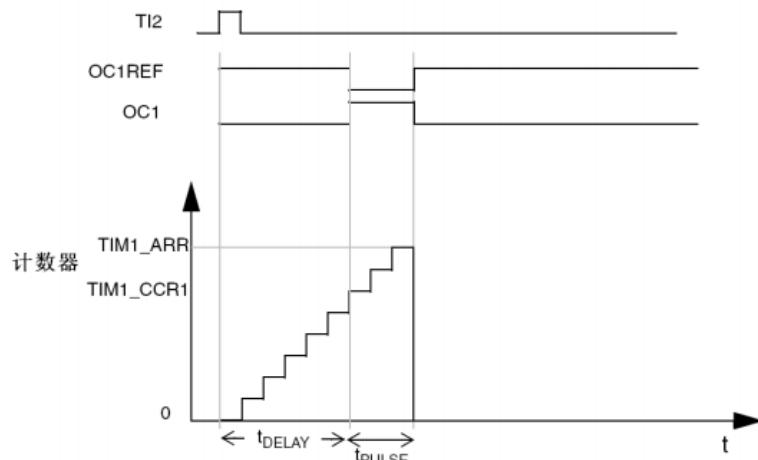


图 16-14 单脉冲模式的例子

### 16.2.10. 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。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。TIM10/11/13/14 作为主定时器输出 oc 给相应配置下的从定时器 TIM9/12。

### 16.2.11. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止)，根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。

## 16.3. 寄存器描述

TIM10 寄存器基址: 0x4001 5000 - 0x4001 53FF

TIM11 寄存器基址: 0x4001 5400 - 0x4001 57FF

TIM13 寄存器基址: 0x4000 1C00 - 0x4000 1FFF

TIM14 寄存器基址: 0x4000 2000 - 0x4000 23FF

### 16.3.1. TIMx 控制寄存器 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				CKD [1:0]		ARPE		保留				OPM		URS	
保留				RW		RW		保留				RW		RW	

Bit	Name	R/W	Reset Value	Function
15:10	保留, 始终为 0			
9:8	CKD	RW	0	时钟分频因子 (Clock division) 这 2 位定义在定时器时钟(CK_INT)频率、死区时间和由死区发生器与数字滤波器(ETR,TIx)所用的采样时钟 (tDTS) 之间的分频比例。 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: 保留, 不要使用这个配置
7	ARPE	RW	0	自动重装载预装载允许位 (Auto-reload preload enable) 0: TIMx_ARR 寄存器没有缓冲; 1: TIMx_ARR 寄存器被装入缓冲器。
6:4	保留, 始终为 0			
3	OPM	RW	0	单脉冲模式 (One pulse mode) 0: 在发生更新事件时, 计数器不停止; 1: 在发生下一次更新事件(清除 CEN 位)时, 计数器停止。
2	URS	RW	0	更新请求源 (Update request source) 软件通过该位选择 UEV 事件的源 0: 如果使能了更新中断请求, 则下述任一事件产生更新中断请求: - 计数器溢出 - 设置 UG 位 1: 如果使能了更新中断请求, 则只有计数器溢出/下溢才产生更新中断请求。
1	UDIS	RW	0	禁止更新 (Update disable) 软件通过该位允许/禁止 UEV 事件的产生 0: 允许 UEV。更新(UEV)事件由下述任一事件产生: - 计数器溢出 - 设置 UG 位 具有缓存的寄存器被装入它们的预装载值。(译注: 更新影子寄存器) 1: 禁止 UEV。不产生更新事件, 影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位, 则计数器和预分频器被重新初始化。
0	CEN	RW	0	使能计数器 (Counter enable) 0: 禁止计数器; 1: 使能计数器。

### 16.3.2. TIMx 中断使能寄存器 (TIMx\_DIER)

Address offset: 0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														CC1IE	UIF
保留														RW	RW

Bit	Name	R/W	Reset Value	Function
15:2	保留, 始终为 0			
1	CC1IE	RW	0	允许捕获/比较 1 中断 (Capture/Compare 1 interrupt enable) 0: 禁止捕获/比较 1 中断; 1: 允许捕获/比较 1 中断
0	UIF	RW	0	允许更新中断 (Update interrupt enable) 0: 禁止更新中断; 1: 允许更新中断。

### 16.3.3. TIMx 状态寄存器 (TIMx\_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CC1OF								CC1IF	UIF
保留														RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function										
15:10	保留, 始终为 0													
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'。										
保留, 始终读为 0.														
1	CC1IF	RC_W0	0	捕获/比较 1 中断标记 (Capture/Compare 1 interrupt flag) 如果通道 CC1 配置为输出模式: 当计数器值与比较值匹配时该位由硬件置 1。它由软件清'0'。 0: 无匹配发生; 1: TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 的内容时, 在计数器溢出时 CC1IF 位变高 如果通道 CC1 配置为输入模式: 当捕获事件发生时该位由硬件置'1', 它由软件清'0'或通过读 TIMx_CCR1 清'0'。 0: 无输入捕获产生; 1: 计数器值已被捕获(拷贝)至 TIMx_CCR1(在 IC1 上检测到与所选极性相同的边沿)。										
0	UIF	RC_W0	0	更新中断标记 (Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0: 无更新事件产生; 1: 更新中断等待响应。当寄存器被更新时该位由硬件置'1': - 若 TIMx_CR1 寄存器的 UDIS=0, 当重复计数器数值上溢时。 - 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。										

### 16.3.4. TIMx 事件产生寄存器 (TIMx\_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														CC1G	UG
保留														W	W

Bit	Name	R/W	Reset Value	Function
15:2	保留, 始终为 0			
1	CC1G	W	0	<p>产生捕获/比较 1 事件 (Capture/Compare 1 generation) 该位由软件置'1', 用于产生一个捕获/比较事件, 由硬件自动清'0'。 0: 无动作; 1: 在通道 CC1 上产生一个捕获/比较事件: 若通道 CC1 配置为输出: 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。 若通道 CC1 配置为输入: 当前的计数器值被捕获至 TIMx_CCR1 寄存器; 设置 CC1IF=1, 若开启对应的中断, 则产生相应的中断。 若 CC1IF 已经为 1, 则设置 CC1OF=1。</p>
0	UG	W	0	<p>产生更新事件 (Update generation) 该位由软件置'1', 由硬件自动清'0'。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意预分频器的计数器也被清'0'(但是预分频系数不变)。</p>

### 16.3.5. TIMx 捕获/比较模式控制寄存器 1 (TIMx\_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留									OC1M [2:0]		OC1PE	保留	CC1S [1:0]		
保留									IC1F [3:0]		IC1PSC [1:0]				
保留									RW	RW	RW	RW	RW	RW	RW

#### 16.3.5.1. 输出比较模式

Bit	Name	R/W	Reset Value	Function
15: 7	保留, 始终为 0			
6:4	OC1M	RW	0	<p>输出比较 1 模式 (Output Compare 1 mode) 该 3 位定义了输出参考信号 OC1REF 的动作, 而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效, 而 OC1 的有效电平取决于 CC1P 位。 000: 冻结。输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用; 001: 匹配时设置通道 1 为有效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时, 强制 OC1REF 为高。 010: 匹配时设置通道 1 为无效电平。当计数器 TIMx_CNT 的值与捕获/比较寄存器 1(TIMx_CCR1)相同时, 强制 OC1REF 为低。 011: 翻转。当 TIMx_CCR1=TIMx_CNT 时, 翻转 OC1REF 的电平。 100: 强制为无效电平。强制 OC1REF 为低。 101: 强制为有效电平。强制 OC1REF 为高。 110: PWM 模式 1 - TIMx_CNT&lt;TIMx_CCR1 时通道 1 为有效电平, 否则为无效电平。 111: PWM 模式 2 - TIMx_CNT&lt;TIMx_CCR1 时通道 1 为无效电平, 否则为有效电平。</p>

Bit	Name	R/W	Reset Value	Function
				注：在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
3	OC1PE	RW	0	<p>输出比较 1 预装载使能 (Output Compare 1 preload enable)</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，并且新写入的数值立即起作用。</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加载至当前寄存器中。</p> <p>注：仅在单脉冲模式下(TIMx_CR1 寄存器的 OPM=1)，可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。</p>
2	保留，始终为 0			
1:0	CC1S	RW	0	<p>捕获/比较 1 选择。 (Capture/Compare 1 selection)</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: Reserved；</p> <p>11: Reserved；</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>

### 16.3.5.2. 输入捕获模式

Bit	Name	R/W	Reset Value	Function
15:8	保留，始终为 0			
7:4	IC1F	RW	0	<p>输入捕获 1 滤波器 (Input capture 1 filter)</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 1000: 采样频率 fSAMPLING=fDTS/8, N=6</p> <p>0001: 采样频率 fSAMPLING=fCK_INT, N=2 1001: 采样频率 fSAMPLING=fDTS/8, N=8</p> <p>0010: 采样频率 fSAMPLING=fCK_INT, N=4 1010: 采样频率 fSAMPLING=fDTS/16, N=5</p> <p>0011: 采样频率 fSAMPLING=fCK_INT, N=8 1011: 采样频率 fSAMPLING=fDTS/16, N=6</p> <p>0100: 采样频率 fSAMPLING=fDTS/2, N=6 1100: 采样频率 fSAMPLING=fDTS/16, N=8</p> <p>0101: 采样频率 fSAMPLING=fDTS/2, N=8 1101: 采样频率 fSAMPLING=fDTS/32, N=5</p> <p>0110: 采样频率 fSAMPLING=fDTS/4, N=6 1110: 采样频率 fSAMPLING=fDTS/32, N=6</p> <p>0111: 采样频率 fSAMPLING=fDTS/4, N=8 1111: 采样频率 fSAMPLING=fDTS/32, N=8</p>
3:2	IC1PSC	RW	0	<p>输入/捕获 1 预分频器 (Input capture 1 prescaler)</p> <p>这 2 位定义了 CC1 输入(IC1)的预分频系数。</p> <p>一旦 CC1E=0(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	RW	0	<p>捕获/比较 1 选择 (Capture/Compare 1 Selection)</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: Reserved；</p> <p>11: Reserved；</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>

### 16.3.6. TIMx 捕获/比较使能寄存器 (TIMx\_CCER)

Address offset:0x20

Reset value:0x0000

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

保留	CC1NP	保留	CC1P	CC1E
保留	RW	保留	RW	RW

Bit	Name	R/W	Reset Value	Function
15:4	保留, 始终为 0			
3	CC1NP	RW	0	输入/捕获 1 互补输出极性 (Capture/Compare 1 complementary output polarity) CC1 通道配置为输出时：CC1NP 必须保持为 0； CC1 通道配置为输入时：CC1NP 被用来与 CC1P 共同作用 TIxFP1 的极性（参考 CC1P 描述）
2	保留, 始终为 0			
1	CC1P	RW	0	输入/捕获 1 输出极性 (Capture/Compare 1 output polarity) CC1 通道配置为输出： 0：OC1 高电平有效； 1：OC1 低电平有效。 CC1 通道配置为输入： CC1NP/CC1P 位选择作为触发或捕获信号的 TI1FP1 和 TI2FP1 的极性。 00：不反相/上升沿： TIxFP1 上升沿有效（捕获模式下触发）； 01：反相/下降沿： TIxFP1 下降沿有效（捕获模式下触发）； 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（捕获模式下触发）；
0	CC1E	RW	0	输入/捕获 1 输出使能 (Capture/Compare 1 output enable) CC1 通道配置为输出： 0：关闭 - OC1 禁止输出。 1：开启 - OC1 信号输出到对应的输出引脚。 CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 0：捕获禁止； 1：捕获使能。

表 16-1 输出通道 OCx 的控制位

CCxE 位	OCx 输出状态
0	输出禁止（与定时器断开） OCx=0, OCx_EN=0
1	OCx=OCxREF+Polarity, OCx_EN=1

注：引脚连接到 OCx 通道的外部 I/O 引脚的状态，取决于 OCx 通道状态和 GPIO 寄存器。

### 16.3.7. TIMx 计数器 (TIMx\_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	计数器的值 (Counter value)

### 16.3.8. TIMx 预分频器 (TIMx\_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	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)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

### 16.3.9. TIMx 自动重装载寄存器 (TIMx\_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。

### 16.3.10. TIMx 捕获/比较寄存器 1 (TIMx\_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	捕获/比较通道 1 的值 (Capture/Compare 1 value) 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值(预装载值)。 如果在 TIMx_CCMR1 寄存器(OC1PE 位)中未选择预装载功能，写入的数值会立即传输至当前寄存器中。 否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件(IC1)传输的计数器值。

### 16.3.11. TIMx 选项寄存器 (TIMx\_OR)

Address offset:0x50

Reset value:0x0000

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

	保留	T11_RMP
	保留	RW

Bit	Name	R/W	Reset Value	Function
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: TIM 通道 1 连接到 GPIO, 具体参考数据手册的复用功能。 01: 保留 10: TIM 通道 1 连接到 HSE/128 时钟 11: TIM 通道 1 连接到 MCU 时钟输出 (MCO) .这个配置是通过 RCC_CFRG 寄存器的 MCO[2:0]的设置来决定源, RCC_CFRG1 寄存器的 MCOPRE[2:0]决定分频的。

### 16.3.12. TIMx 寄存器映射

表 16-2 TIMx 寄存器映射



# 17. 基本定时器 (TIM6和 TIM7)

## 17.1. 简介

基本定时器 TIM6 和 TIM7 各包含一个 16 位自动装载计数器，由各自的可编程预分频器驱动。

TIM6 和 TIM7 是完全独立的，它们不共享任何资源。

### 17.1.1. TIM6 和 TIM7 主要特征

TIM6 和 TIM7 定时器的主要功能包括：

- 16 位自动装载计数器
- 16 位可编程(可以实时修改)的预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 在更新事件 (计数器溢出) 发生时产生中断/DMA

### 17.1.2. 模块框图

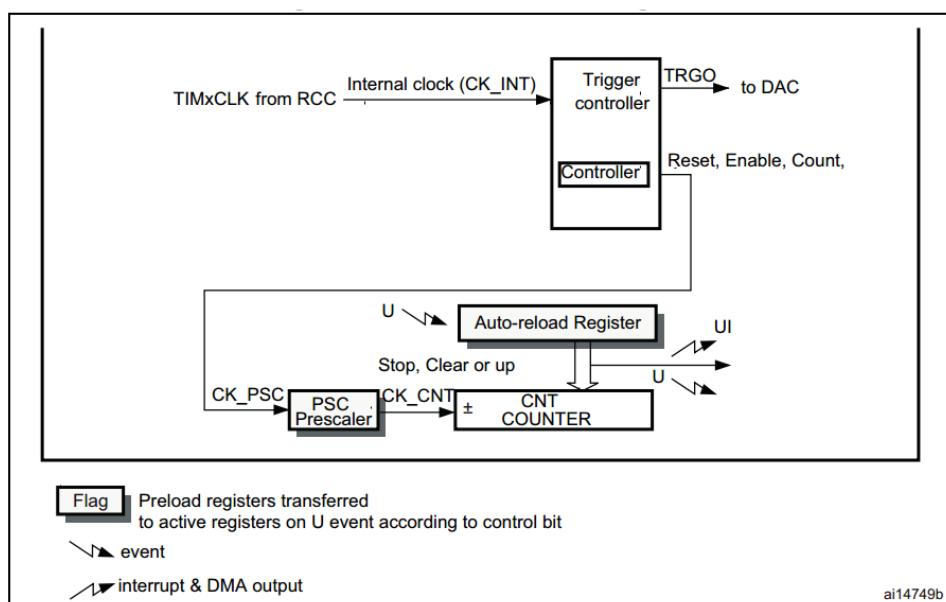


图 17-1 TIM6 和 TIM7 模块

## 17.2. TIM6 和 TIM7 功能描述

### 17.2.1. 时基单元

这个可编程定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。计数器的时钟可以被预分频器分频。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，即使计数器还在运行读写仍然有效。

时基单元包含：

- 计数器寄存器(TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)
- 自动装载寄存器 (TIMx\_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问它的预装载寄存器。根据在 TIMx\_CR1 寄存器中的自动装载预装载使能位(ARPE)的设置，预装载寄存器的内容被立即或在每次的更新事件 (UEV) 时传送到影子寄存器。当计数器达到溢出条件，并当 TIMx\_CR1 寄存器中的 UDIS 位等于 0 时，会产生更新事件。更新事件也可以由软件产生。后续会详细描述每一种配置下更新事件的产生。

计数器由预分频器分频后的时钟输出 CK\_CNT 驱动，仅当设置了 TIMx\_CR1 寄存器中的计数器使能位(CEN)，CK\_CNT 才对计数器有效。

注意，在设置了 TIMx\_CR1 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

### 17.2.1.1. 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个(在 TIMx\_PSC 寄存器中的)16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频参数将在下一次更新事件到来时被采用。

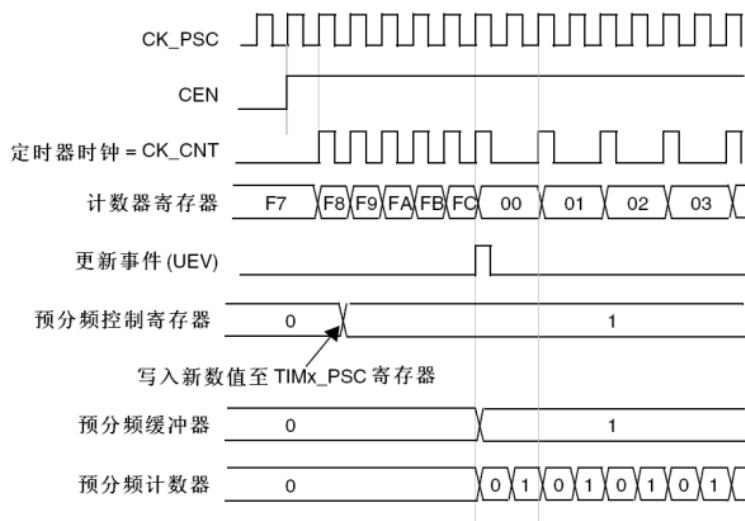


图 17-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

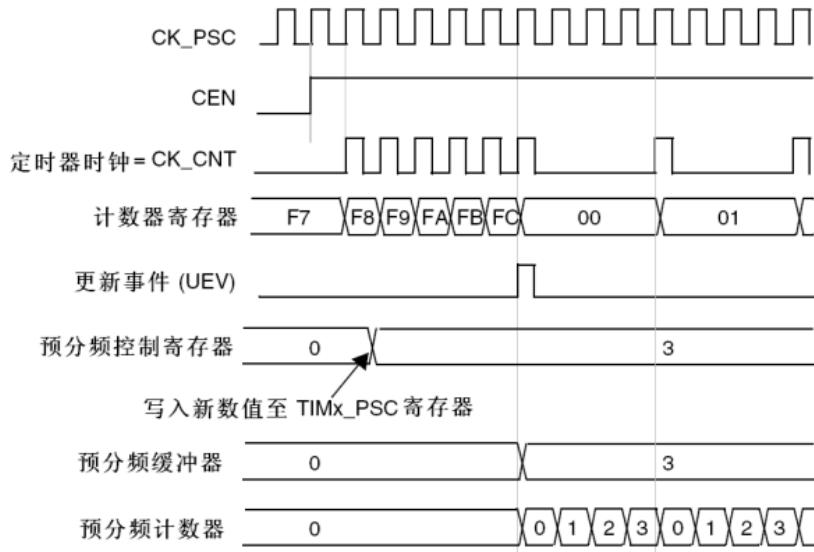


图 17-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

### 17.2.2. 计数器模式

计数器从 0 计数到自动加载值(TIMx\_ARR 的内容)，然后重新从 0 开始计数并且产生一个计数上溢事件。

每次计数上溢都会产生更新事件。而在 TIMx\_EGR 寄存器中(通过软件方式或者使用从模式控制器)设置 UG 位也同样可以产生一个更新事件。

通过设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清‘0’之前，将不会产生更新事件。即使这样，在应该产生更新事件时，计数器仍会被清‘0’，同时预分频器内部的计数器也被清‘0’(但预分频器的数值不变)。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位(选择更新请求源)，通过设置 UG 位可以产生一个更新事件 UEV，但不会置起 UIF 标志位(即不会产生中断或 DMA 请求)。

当发生一个更新事件时，所有以下的寄存器都被更新，硬件同时(依据 URS 位)设置更新标志位(TIMx\_SR 寄存器中的 UIF 位)：

- 自动装载影子寄存器被重新置入预装载寄存器的值(TIMx\_ARR)。
- 预分频器的缓冲区被置入预装载寄存器的值(TIMx\_PSC 寄存器的内容)。

下图给出一些例子，当 TIMx\_ARR=0x36 时计数器在不同时钟频率下的动作。

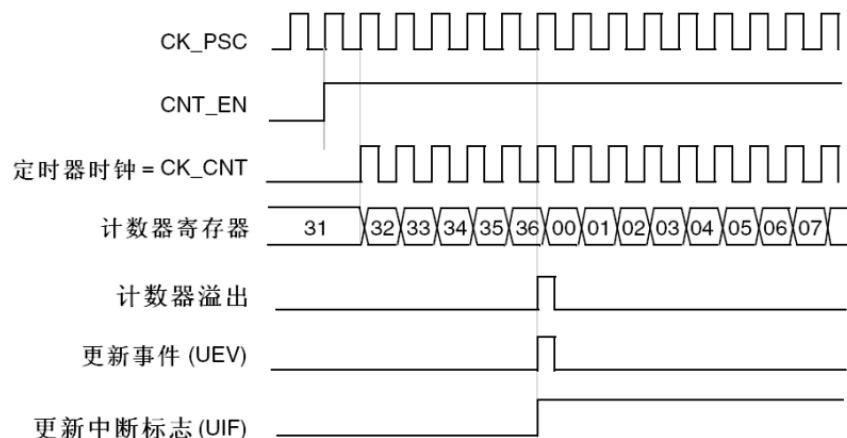


图 17-4 计数器时序图，内部时钟分频因子为 1

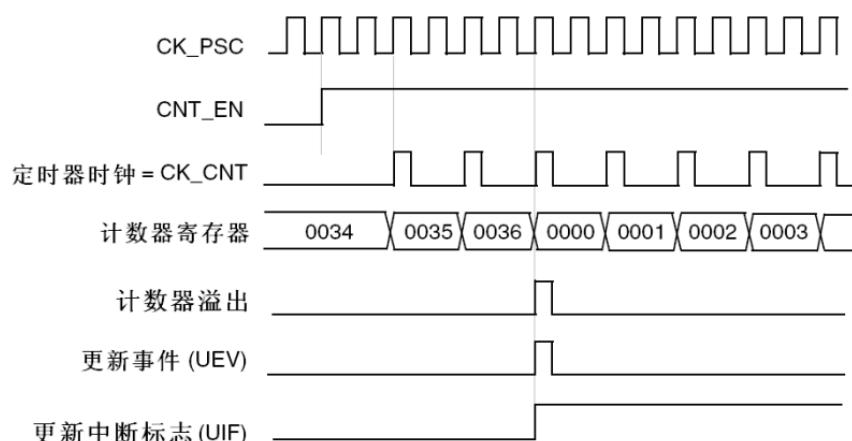


图 17-5 计数器时序图，内部时钟分频因子为 2

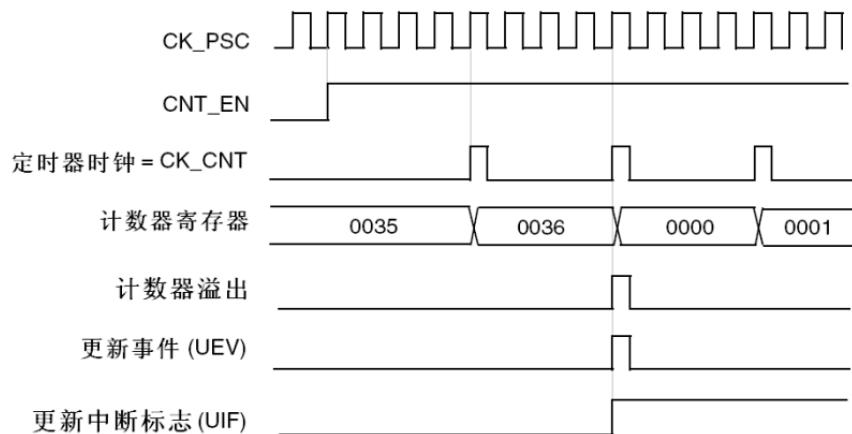


图 17-6 计数器时序图，内部时钟分频因子为 4

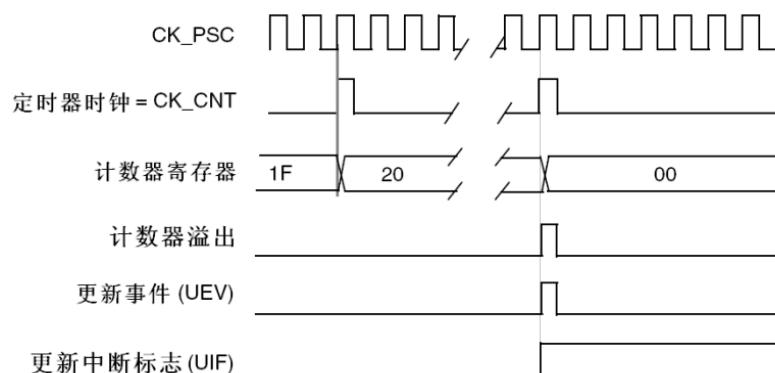


图 17-7 计数器时序图，内部时钟分频因子为 N

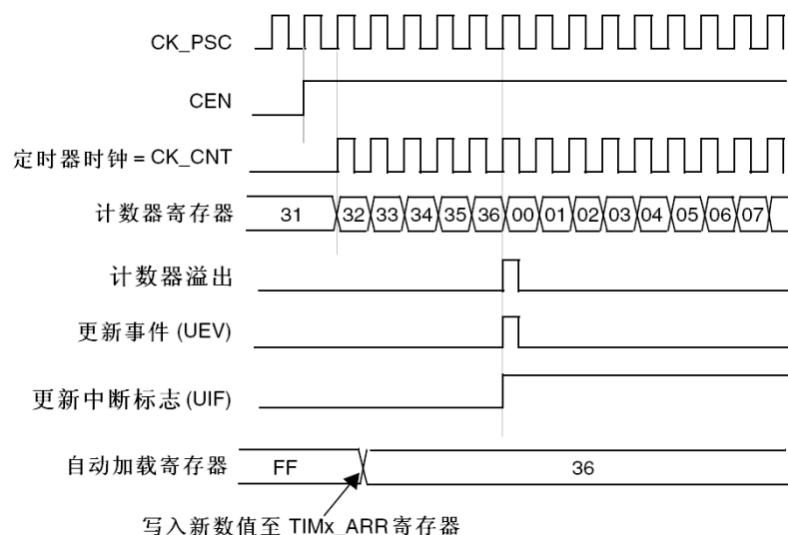


图 17-8 计数器时序图，当 ARPE=0 时的更新事件(没有预装 TIMx\_ARR)

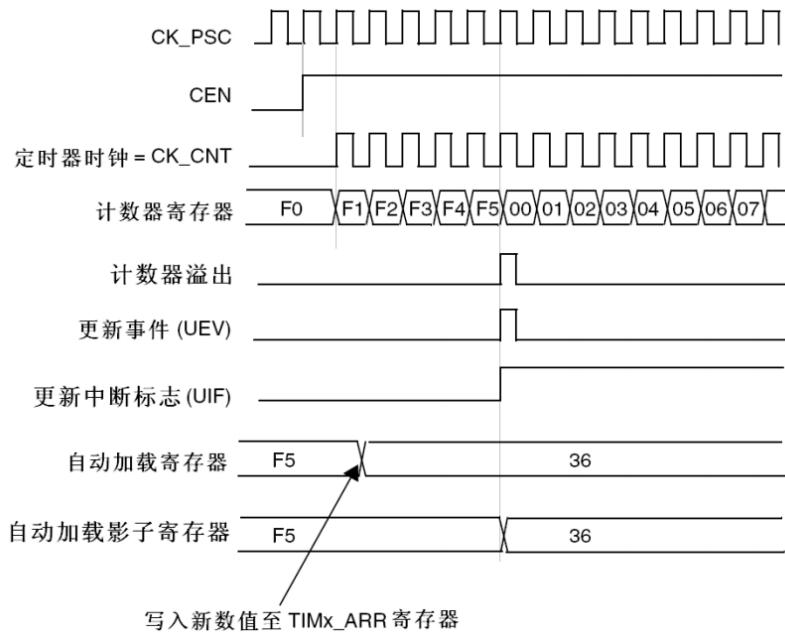


图 17-9 计数器时序图, 当 ARPE=1 时的更新事件(预装了 TIMx\_ARR)

### 17.2.3. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止), 根据 DBG 模块中 DBG\_TIMx\_STOP 的设置, TIMx 计数器可以或者继续正常操作, 或者停止。

## 17.3. 寄存器描述

TIM6 寄存器地址: 0x4000 1000

TIM7 寄存器地址: 0x4000 1400

### 17.3.1. TIM6 和 TIM7 控制寄存器 1 (TIMx\_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ARPE	保留			OPM	URS	UDIS	CEN
保留								RW	保留			RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
7	ARPE	RW	0	自动重装载预装载允许位 (Auto-reload preload enable) 0: TIMx_ARR 寄存器没有缓冲; 1: TIMx_ARR 寄存器被装入缓冲器。
6:4	保留, 始终为 0			
3	OPM	RW	0	单脉冲模式 (One pulse mode) 0: 在发生更新事件时, 计数器不停止; 1: 在发生下一次更新事件(清除 CEN 位)时, 计数器停止。
2	URS	RW	0	更新请求源 (Update request source)

Bit	Name	R/W	Reset Value	Function
				<p>软件通过该位选择 UEV 事件的源            0: 如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求：            - 计数器溢出/下溢            - 设置 UG 位            - 从模式控制器产生的更新            1: 如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。</p>
1	UDIS	RW	0	<p>禁止更新 (Update disable)            软件通过该位允许/禁止 UEV 事件的产生            0: 允许 UEV。更新(UEV)事件由下述任一事件产生：            - 计数器溢出/下溢            - 设置 UG 位            - 从模式控制器产生的更新            具有缓存的寄存器被装入它们的预装载值。 (译注：更新影子寄存器)            1: 禁止 UEV。不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。</p>
0	CEN	RW	0	<p>使能计数器 (Counter enable)            0: 禁止计数器；            1: 使能计数器。            注：在软件设置了 CEN 位后，门控模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。            在单脉冲模式下，当产生更新事件时 CEN 被自动清除。</p>

### 17.3.2. TIM6 和 TIM7 控制寄存器 2 (TIMx\_CR2)

Address offset:0x04

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:7	保留, 始终为 0			
6:4	MMS	RW	0	<p>主模式选择 (Master mode selection)            这 3 位用于选择在主模式下送到从定时器的同步信息(TRGO)。可能的组合如下：            000: 复位 – TIMx_EGR 寄存器的 UG 位被用于作为触发输出(TRGO)。如果是触发输入产生的复位(从模式控制器处于复位模式)，则 TRGO 上的信号相对实际的复位会有一个延迟。            001: 使能 – 计数器使能信号 CNT_EN 被用于作为触发输出(TRGO)。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式(见 TIMx_SMCR 寄存器中 MSM 位的描述)。            010: 更新 – 更新事件被选为触发输入(TRGO)。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。            注意：从定时器和 ADC 的时钟必须先被使能以接收主定时器的信号，并在接收时不要改变。</p>
3:0	保留, 始终读为 0			

### 17.3.3. TIM6 和 TIM7 DMA/中断使能寄存器 (TIMx\_DIER)

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:9	保留, 始终为 0			
8	UDE	RW	0	允许更新的 DMA 请求 (Update DMA request enable) 0: 禁止更新的 DMA 请求; 1: 允许更新的 DMA 请求。
7:1	保留, 始终为 0			
0	UIE	RW	0	允许更新中断 (Update interrupt enable) 0: 禁止更新中断; 1: 允许更新中断。

#### 17.3.4. TIM6 和 TIM7 状态寄存器 (TIMx\_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留								UIF	保留							
保留								RC_W0	保留							

Bit	Name	R/W	Reset Value	Function
15:1	保留, 始终为 0			
0	UIF	RC_W0	0	更新中断标记 (Update interrupt flag) 当产生更新事件时该位由硬件置'1'。它由软件清'0'。 0: 无更新事件产生; 1: 更新中断等待响应。当寄存器被更新时该位由硬件置'1': - 若 TIMx_CR1 寄存器的 UDIS=0, 当重复计数器数值上溢或下溢时。 - 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。

#### 17.3.5. TIM6 和 TIM7 事件产生寄存器 (TIMx\_EGR)

Address offset:0x14

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:1	保留, 始终为 0			
0	UG	W	0	产生更新事件 (Update generation) 该位由软件置'1', 由硬件自动清'0'。 0: 无动作;

Bit	Name	R/W	Reset Value	Function
				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)。

### 17.3.6. TIM6 和 TIM7 计数器 (TIMx\_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	计数器的值 (Counter value)

### 17.3.7. TIM6 和 TIM7 预分频器 (TIMx\_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	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)。 PSC 包含了每次当更新事件产生时，装入当前预分频器寄存器的值；更新事件包括计数器被 TIM_EGR 的 UG 位清'0'或被工作在复位模式的从控制器清'0'。

### 17.3.8. TIM6 和 TIM7 自动重装载寄存器 (TIMx\_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	自动重装载的值 (Prescaler value) ARR 包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。

### 17.3.9. TIM6 和 TIM7 寄存器映射

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_C_R1 Rea d/Wr ite Re-set Valu e	Reserved																								Reserved							
0x004	TIMx_C_R2 Rea d/Wr ite Re-set Valu e	Reserved																								Reserved							
0x00C	TIMx_DI_ER Rea d/Wr ite Re-set Valu e	Reserved																								Reserved							
0x110	TIMx_SR Rea d/Wr ite Re-set Valu e	Reserved																								Reserved							
0x114	TIMx_E_GR Rea d/Wr ite Re-set Valu e	Reserved																								Reserved							
0x224	TIMx_CNT Rea d/Wr ite Re-set Valu e	Reserved												CNT								rw				0							
0x228	TIMx_PSC Rea d/Wr ite Re-set Valu e	Reserved												PSC								rw				0							

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0									
0	TIMx_A_RR	Reserved																ARR																								
x	Read/Write	rw																																								
C	Reset Value	0xFFFF																																								

# 18. 实时时钟 (RTC)

## 18.1. 概述

实时时钟是一个独立的定时器。 RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统(RCC\_BDCR 寄存器)处于备份区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对备份寄存器和 RTC 的访问被禁止，这是为了防止对备份区域(BKP)的意外写操作。执行以下操作将使能对备份寄存器和 RTC 的访问：

- 设置寄存器 RCC\_APB1ENR 的 PWREN 和 BKREN 位，使能电源和备份接口时钟
- 设置寄存器 PWR\_CR 的 DBP 位，使能对备份寄存器和 RTC 的访问。

### 18.1.1. 主要特性

主要功能如下：

- 可编程的预分频系数：分频系数最高为  $2^{20}$ 。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟
- 可以选择以下三种 RTC 的时钟源：
  - ✓ HSE 时钟除以 128；
  - ✓ LSE 振荡器时钟；
  - ✓ LSI 振荡器时钟
- 2 个独立的复位类型：
  - ✓ APB1 接口由系统复位；
  - ✓ RTC 核心(预分频器、闹钟、计数器和分频器)只能由备份域复位。
- 3 个专门的可屏蔽中断：
  - ✓ 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - ✓ 秒中断，用来产生一个可编程的周期性中断信号(最长可达 1 秒)。
  - ✓ 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态

### 18.1.2. 结构框图

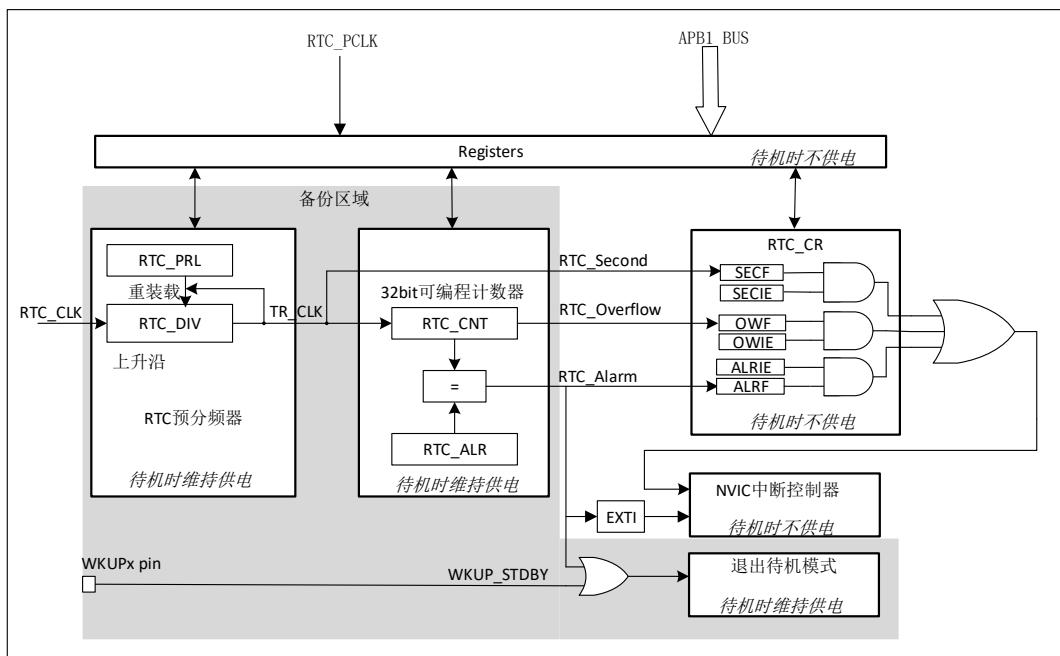


图 18-1 RTC 模块框图

RTC 由两个主要部分组成(参见上图)。第一部分(APB1 接口)用来和 APB1 总线相连。此单元还包含一组 16 位寄存器 (RTC\_CR, 实际分散在两个地址的寄存器) , 可通过 APB1 总线对其进行读写操作。APB1 接口由 APB1 总线时钟驱动, 用来连接 APB1 总线。

另一部分(RTC 核)由一组可编程计数器组成, 分成两个主要模块。

第一个模块是 RTC 的预分频器模块, 它可编程产生最长为 1 秒的 RTC 时间基准 TR\_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器(RTC 预分频器)。如果在 RTC\_CR 寄存器中设置了相应的允许位, 则在每个 TR\_CLK 周期中 RTC 产生一个中断(秒中断) (second) 。

第二个模块是一个 32 位的可编程计数器, 可被初始化为当前的系统时间。系统时间按 TR\_CLK 周期累加并与存储在 RTC\_ALR 寄存器中的可编程时间相比较, 如果 RTC\_CR 控制寄存器中设置了相应允许位, 比较匹配时将产生一个闹钟中断 (alarm) 。

闹钟中断作为待机模式和停止模式的唤醒信号。上图为待机模式的情况, 从待机模式唤醒后。在停止模式, RTC 闹钟作为唤醒需要配置 EXTI line17 对应寄存器。

## 18.2. 功能描述

### 18.2.1. 寄存器复位

RTC\_PRL, RTC\_ALR, RTC\_CNT 和 RTC\_DIV 寄存器仅能通过备份域复位信号复位。其他系统寄存器 (RTC\_CR) 由系统复位或电源复位进行异步复位。

### 18.2.2. 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。

软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是, 相关的可读寄存器只在与 RTC 时钟的上升沿同步到 RTC APB1 时钟后的信号有效时更新。RTC 标志也是如此的。

这意味着, 如果 APB1 接口曾经被关闭, 而读操作又是在刚刚重新开启 APB1 之后, 则在第一次的内部寄存器更新之前, 从 APB1 上读出的 RTC 寄存器数值可能被破坏了(通常读到 0)。下述几种情况下能够发生这种情形:

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒
- 系统刚从停机模式唤醒

所有以上情况下，APB1 接口被禁止时(复位、无时钟或断电)RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC\_CRL 寄存器中的 RSF 位(寄存器同步标志)被硬件置'1'。

注：RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。

说明：

- 1) CPU 可读的寄存器包括 RTC\_CR, RTC\_CNT 和 RTC\_DIV;
- 2) RTC\_CR 寄存器为 RTC\_PCLK 域，CPU 任何时候读能读到稳定值；
- 3) RTC\_CNT 和 RTC\_DIV 来源于 RTC\_CLK 域。在 RTC 工作后，RTC\_DIV 寄存器在每个 RTC\_CLK 的上升沿更新；RTC\_CNT 和来源于 RTC\_CLK 时钟域的标志位同样采用跟 RTC\_DIV 寄存器同样的更新信号，虽然这样不是每次更新时 RTC\_CNT 的值都改变；
- 4) RSF 实现在 RTC\_PCLK 域，在 RTC\_CLK 同步到 RTC\_PCLK 后的脉冲信号有效时置位；
- 5) RSF 仅控制 RTC\_CNT 和 RTC\_DIV 的读取时机（硬件不会控制）

### 18.2.3. 配置 RTC 寄存器

必须设置 RTC\_CRL 寄存器中的 CNF 位，使 RTC 进入配置模式后，才能写入 RTC\_PRL、RTC\_CNT、RTC\_ALR 寄存器。

另外，对 RTC 任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询 RTC\_CR 寄存器中的 RTOFF 状态位，判断 RTC 寄存器是否处于更新中。仅当 RTOFF 状态位是'1'时，才可以写入 RTC 寄存器。

配置过程：

- 查询 RTOFF 位，直到 RTOFF 的值变为'1'；
- 置 CNF 值为 1，进入配置模式；
- 对一个或多个 RTC 寄存器进行写操作；
- 清除 CNF 标志位，退出配置模式；
- 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) )

### 18.2.4. RTC 标志设置

在每一个 RTC 核心的时钟周期中，更改 RTC 计数器之前设置 RTC 秒标志(SECF)。

在计数器到达 0x0000 之前的最后一个 RTC 时钟周期中，设置 RTC 溢出标志(OWF)。

在计数器的值到达闹钟寄存器的值加 1(RTC\_ALR+1)之前的 RTC 时钟周期中，设置 RTC\_Alarm 和 RTC 闹钟标志(ALRF)。

对 RTC 闹钟寄存器 (RTC\_ALR) 的写操作必须使用下述过程之一与 RTC 秒标志同步：

- 使用 RTC 闹钟中断，并在中断处理程序中修改 RTC 闹钟寄存器 (RTC\_ALR) 和/或 RTC 计数器寄存器 (RTC\_CNT) 。
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟寄存器 (RTC\_ALR) 和/或 RTC 计数器寄存器 (RTC\_CNT) 。

### 18.2.5. RTC 时序

RTC 秒和闹钟时序如下图所示：

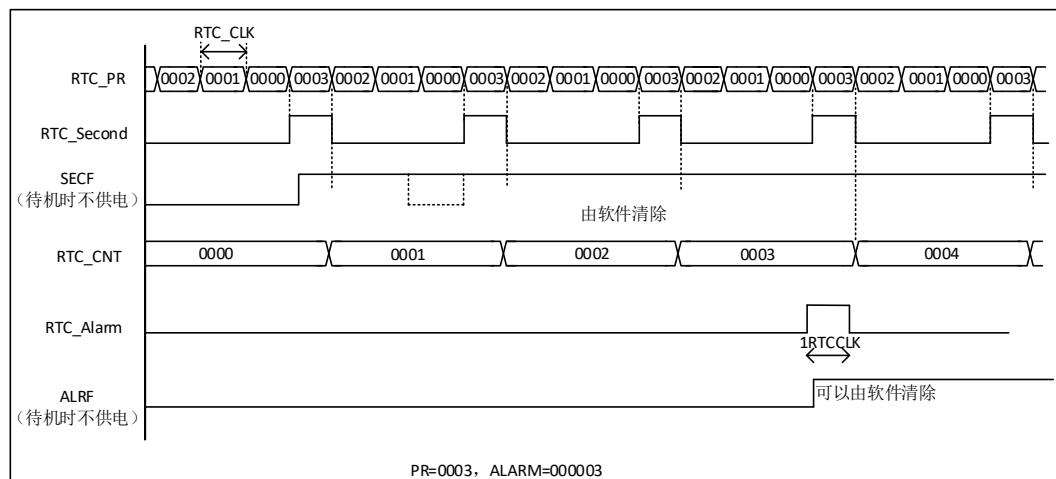


图 18-2 RTC 秒和闹钟时序

SECF 和 ALRF 为 RTC\_PCLK 域信号，分别采样 RTC\_CLK 域 RTC\_Second 和 RTC\_Alarm 信号产生。

RTC 溢出时序如下图所示：

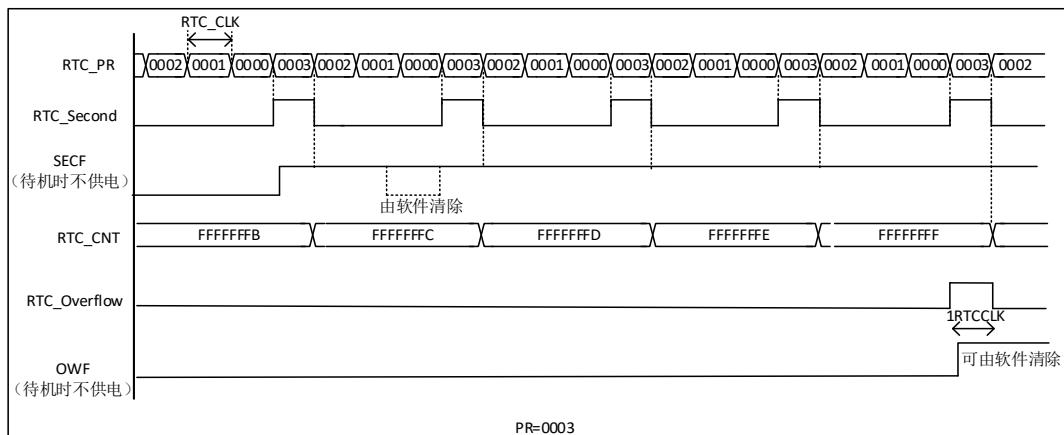


图 18-3 RTC 溢出时序

SECF 和 OWF 为 RTC\_PCLK 域信号，分别采样 RTC\_CLK 域 RTC\_Second 和 RTC\_Overflow 信号产生。

## 18.3. 寄存器描述(基址 0x4000\_2800)

### 18.3.1. RTC 控制寄存器高位(RTC\_CRH)(0x00)

Address offset:0x00

Reset value:0x0000

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

该寄存器由系统复位复位。

Bit	Name	R/W	Reset Value	Function
31:3	Reserved			Reserved

Bit	Name	R/W	Reset Value	Function
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。

### 18.3.2. RTC 控制寄存器低位(RTC\_CRL)(0x04)

Address offset:0x04

Reset value:0x0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	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	Reserved	-	-	Reserved
5	RTOFF	R	1	RTC 操作关闭 (RTC operation OFF)，该位只读。 RTC 模块利用该位来指示对其寄存器进行的最后一次操作的状态（指示操作是否完成）。 若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。 0: 上一次对 RTC 寄存器的写操作仍在进行 1: 上一次对 RTC 寄存器的写操作已经完成
4	CNF	RW	0	配置标志 (Configuration flag)。 此位必须由软件置'1'以进入配置模式，从而允许向 RTC_CNT、RTC_ALR 或 RTC_PRL 寄存器写入新值。 只有当此位在被置'1'，并重新由软件清'0'后，才会执行写操作。 0: 退出配置模式(开始更新 RTC 寄存器) 1: 进入配置模式
3	RSF	RC_W0	0	寄存器同步标志 (Registers synchronized flag).该寄存器由硬件置位，软件清零。当 RTC_CNT 寄存器和 RTC_DIV 寄存器更新后，硬件置'1'该位。 在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清'0'。 要进行任何的读操作之前，用户程序必须等待该位被硬件置'1'，以确保 RTC_CNT、RTC_ALR 或 RTC_PRL 已经被同步。 0: 寄存器尚未被同步 1: 寄存器已经被同步
2	OWF	RC_W0	0	溢出标志 (Overflow flag)。 当 32 位可编程计数器溢出时，此位由硬件置'1'。如果 RTC_CRH 寄存器中 OWIE=1，则产生中断。 此位只能由软件清'0'，写'1'无效。

Bit	Name	R/W	Reset Value	Function
				0: 无溢出; 1: 32 位可编程计数器溢出
1	ALRF	RC_W0	0	闹钟标志 (Alarm flag)。 当 32 位可编程计数器达到 RTC_ALR 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清'0'，写'1'无效。 0: 无闹钟； 1: 有闹钟。
0	SECF	RC_W0	0	秒标志 (Second flag)。 当 32 位可编程预分频器溢出时，此位由硬件置'1'，同时 RTC 计数器加 1。因此，此标志为分辨率可编程的 RTC 计数器提供一个周期性的信号(通常为 1 秒)。如果 RTC_CRH 寄存器中 SECIE=1，则产生中断。此位只能由软件清除，写'1'无效。 0: 秒标志条件不成立 1: 秒标志条件成立

RTC 的功能是被该控制寄存器控制的。当外设正在继续上一次写操作时 (RTOFF=0) , 是不能写 RTC\_CRL 寄存器的。

注：

- 任何标志位都将保持挂起状态，直到适当的 RTC\_CR 请求位被软件复位，表示所请求的中断已经被接受。
  - 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（）。
  - 当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新（无法同步）。
  - OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零。
  - 若 ALRF=1 且 ALRIE=1，则允许产生 RTC 全局中断。如果在 EXTI 控制器中允许产生 EXTI 线 17 中断，则允许产生 RTC 全局中断和 RTC 闹钟中断。
  - 若 ALRF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的事件模式，则这条线上会产生一个脉冲(不会产生 RTC 闹钟中断)。

### 18.3.3. RTC 预分频装载寄存器高位(RTC\_PRLH)(0x08)

PRL 寄存器用来保存 RTC 预分频器周期性的计数值。该寄存器是被 RTC\_CR 寄存器的 RTOFF 位写保护的，只有 RTOFF=1，才允许 CPU 进行写操作。

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.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PRL[19:16]														
												W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	-	-	Reserved
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： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位

### 18.3.4. RTC 预分频装载寄存器低位(RTC\_PRL)(0x0C)

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	Reserved	-	-	Reserved
15:0	PRL	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： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ 注：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。

### 18.3.5. RTC 预分频余数寄存器高位(RTC\_DIVH)(0x10)

在每个 TR\_CLK 周期，RTC\_PRL 寄存器的值被重装载到 RTC 预分频计数器里。用户可以通过读取 RTC\_DIV 寄存器，以获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。

该寄存器只读属性，当 RTC\_PRL 或者 RTC\_CNT 寄存器的值发生任何变化，该寄存器值将由硬件重新装载。

Address offset:0x10

Reset value:0x0000

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.															
														R	R

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	-	-	Reserved
3:0	RTC_DIV[19:16]	R	0	RTC 时钟分频器余数高位。

### 18.3.6. RTC 预分频余数寄存器低位(RTC\_DIVL)(0x14)

Address offset:0x14

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
RTC_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	Reserved	-	-	Reserved
15:0	RTC_DIV[15:0]	R	0x8000	RTC 时钟分频器

### 18.3.7. RTC 计数寄存器高位(RTC\_CNTH)(0x18)

RTC 模块有个 32bit 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\_CNT 寄存器用来存放计数器的计数值。寄存器是被写保护的，仅当 RTOFF=1 时 CPU 才能进行写操作。对高 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	Reserved	-	-	Reserved
15:0	RTC_CNT[31:16]	RW	0x0000	RTC core counter 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。

### 18.3.8. RTC 计数寄存器低位( RTC\_CNTL )(0x1C)

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	Reserved	-	-	Reserved
15:0	RTC_CNT[15:0]	RW	0x0000	RTC core counter 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。

### 18.3.9. RTC 闹钟寄存器高位( RTC\_ALRH )(0x20)

当可编程计数器（计数）达到存储在 RTC\_ALR 寄存器的 32bit 值时，并产生 alarm 中断请求。该寄存器是被 RTOFF 位写保护的，只有 RTOFF=1，才允许写访问。

Address offset:0x20

Reset value: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]															
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	Reserved	-	-	
15:0	ALR[31:16]	RW	0xFFFF	RTC 闹钟值高 16bit。 该寄存器用来保存由软件写入的闹钟时间的高 16bit。写该寄存器必须进入配置模式。

### 18.3.10. RTC 闹钟寄存器低位(RTC\_ALRL)(0x24)

Address offset:0x24

Reset value:0xFFFF

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	ALR[15:0]	RW	0xFFFF	RTC 闹钟值低 16bit。 该寄存器用来保存由软件写入的闹钟时间的高 16bit。写该寄存器必须进入配置模式。

### 18.3.11. RTC 寄存器映射

Offset	0 0 x 0	4 1 x 0	0 1 x 0	C 0 x 0	0 0 x 0	4 0 x 0	0 0 x 0	Offset
$\overline{\text{RTC\_C2}}$	$\overline{\text{RTC\_DIV}}$	$\overline{\text{RTC\_e}}$						
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Register
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	31
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	30
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	29
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	28
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	27
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	26
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	25
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	24
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	23
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	22
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	21
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	20
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	19
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	18
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	17
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	16
0	1			1			Res.	15
0	0			0			Res.	14
0	0			0			Res.	13
0	0			0			Res.	12
0	0			0			Res.	11
0	0			0			Res.	10
0	0			0			Res.	9
0	0			0			Res.	8
0	0			0			Res.	7
0	0			0			Res.	6
0	0			0			Res.	5
0	0			0			Res.	4
0	0			0			Res.	3
0	0			0			Res.	2
0	0			0			ALRI	1
0	0			0			SECIE	0
0	0			0			SECF	0

				Offset
4	2 x 0	0	0	C 1 x 0
RTC_AL_RL	Reset value	RTC_AL_RH	RTC_AL_CZ	Register
Res.	Res.	Res.	Res.	31
Res.	Res.	Res.	Res.	30
Res.	Res.	Res.	Res.	29
Res.	Res.	Res.	Res.	28
Res.	Res.	Res.	Res.	27
Res.	Res.	Res.	Res.	26
Res.	Res.	Res.	Res.	25
Res.	Res.	Res.	Res.	24
Res.	Res.	Res.	Res.	23
Res.	Res.	Res.	Res.	22
Res.	Res.	Res.	Res.	21
Res.	Res.	Res.	Res.	20
Res.	Res.	Res.	Res.	19
Res.	Res.	Res.	Res.	18
Res.	Res.	Res.	Res.	17
Res.	Res.	Res.	Res.	16
1	1	1	1	15
1	1	1	1	14
1	1	1	1	13
1	1	1	1	12
1	1	1	1	11
1	1	1	1	10
1	1	1	1	9
1	1	1	1	8
1	1	1	1	7
1	1	1	1	6
1	1	1	1	5
1	1	1	1	4
1	1	1	1	3
1	1	1	1	2
1	1	1	1	1
0	0	0	0	0

RTC\_CNT[15:0]

RTC\_ALR[31:16]

RTC\_ALR[15:0]

# 19. 独立看门狗 (IWDG)

## 19.1. 简介

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 可用来检测和解决由软件错误引起的故障，并在计数器达到指定的超时值时 (TIMEOUT) 触发系统复位。

独立看门狗(IWDG)由专用的低速时钟(LSI)驱动，即使主时钟发生故障它也仍然有效。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。

## 19.2. 主要特征

- Free-running 向下计数器
- 时钟由独立的 RC 振荡器提供(可在 STOP 和 STANDBY 模式下工作)
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位

## 19.3. 功能说明

### 19.3.1. 模块框图

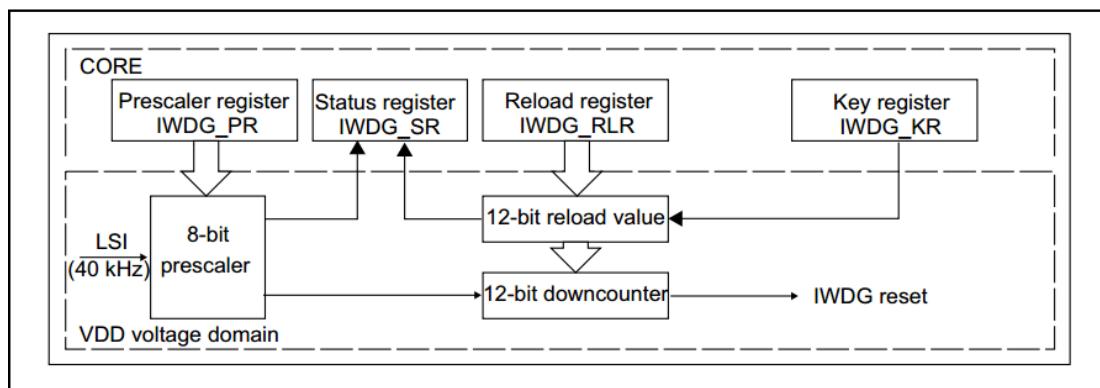


图 19-1 IWDG 模块框图

注：看门狗功能处于 VDD 供电区，即在停机和待机模式时仍能正常工作。

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 19-1

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	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6或7)	6.4	26214.4

注：这些时间是按照 40kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30kHz 到 60kHz 之间变化。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。通过对 LSI 进行校准可获得相对精确的看门狗超时时间。

### 19.3.2. 硬件看门狗

如果上电装载的 option bytes 设置了打开硬件 watchdog，则 IWDG 上电被自动使能，并且如果在计数器计数到终值之前，IWDG key 寄存器没有被软件改写，则产生复位信号。

### 19.3.3. 寄存器保护

对寄存器 IWDG Prescaler 和 IWDG Reload 的写访问是被保护的。为了修改他们，用户必须先向 IWDG 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 Prescaler 寄存器、Reload 寄存器的值正在更新，状态寄存器是会体现出来的。

### 19.3.4. 调试模式

本功能为系统支持 DBG MCU 时才存在。如果 CPU 进入调试模式，IWDG 继续正常计数还是进入 stop 模式，取决于 DBG 模块中 DBG\_IWDG\_STOP 的配置。

### 19.3.5. IWDG 寄存器描述

可以用半字(16 位)或字(32 位)的方式操作这些外设寄存器。

#### 19.3.5.1. 键值寄存器 (IWDG\_KR)

Address offset:0x00

Reset value:0x0000 0000 (由 Standby 模式复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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	Reserved	RES	-	Reserved
15:0	KEY[15:0]	W	32'h0	Key 值。 软件必须以一定的时间间隔向该寄存器写入 0xAAAA，否则，当计数器计数到 0 时，看门狗会产生复位。 0x5555：表示允许访问 IWDG_PR 和 IWDG_RLR； 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）。

### 19.3.5.2. 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.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]														
														RW	

Bit	Name	R/W	Reset Value	Function
31:3	Reserved	RES	-	Reserved
2:0	PR[2:0]	RW	0	预分频值。 通过配置该寄存器选择计数器时钟的预分频值。 要改变该寄存器，IWDG_SR 寄存器的 PVU 必须为 0。 000: 4 分频； 001: 8 分频； 010: 16 分频； 011: 32 分频； 100: 64 分频； 101: 128 分频； 110: 256 分频； 111: 256 分频；

### 19.3.5.3. 重装载寄存器 (IWDG\_RLR)

Address offset:0x08

Reset value:0x0000 0FFF (由 Standby 模式复位)

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	-	Reserved
11:0	RL[11:0]	RW	0	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。

### 19.3.5.4. 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.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RVU PVU														
															R R

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RES	-	Reserved
1	RVU	R	0	看门狗计数器重装值更新。 此位由硬件置 1，表明重装载值正在更新。当重装载值更新结束后，此位由硬件清零。
0	PVU	R	0	看门狗预分频值更新。 此位由硬件置 1，表明预分频值正在更新。当预分频值更新结束后，此位由硬件清零。

注：在更新 IWDG\_PR 和 IWDG\_SR.RLR 前，要分别等待 IWDG\_PVU 和 IWDG\_SR.RVU 为 0.但在更新 IWDG\_PR、IWDG\_RLR 后，不必再等待 IWDG\_SR.PVU、IWDG\_SR.RVU 为 0，可继续执行下面的代码。

### 19.3.5.5. IWDG 寄存器映射

# 20. 窗口看门狗 (WWDG)

## 20.1. 简介

窗口看门狗通常被用来监测，由外部干扰或不可预见的逻辑条件造成应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 T6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减计数器数值(在控制寄存器中)被刷新，那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

### 20.1.1. WWDG 主要特征

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于 0x40，(若看门狗被启动)则产生复位。
  - 当递减计数器在窗口外被重新装载，(若看门狗被启动)则产生复位。
- 如果启动了看门狗并且允许早期唤醒中断(EWI)功能开启，当递减计数器等于 0x40 时产生，它可以被用于重装载计数器以避免 WWDG 复位。

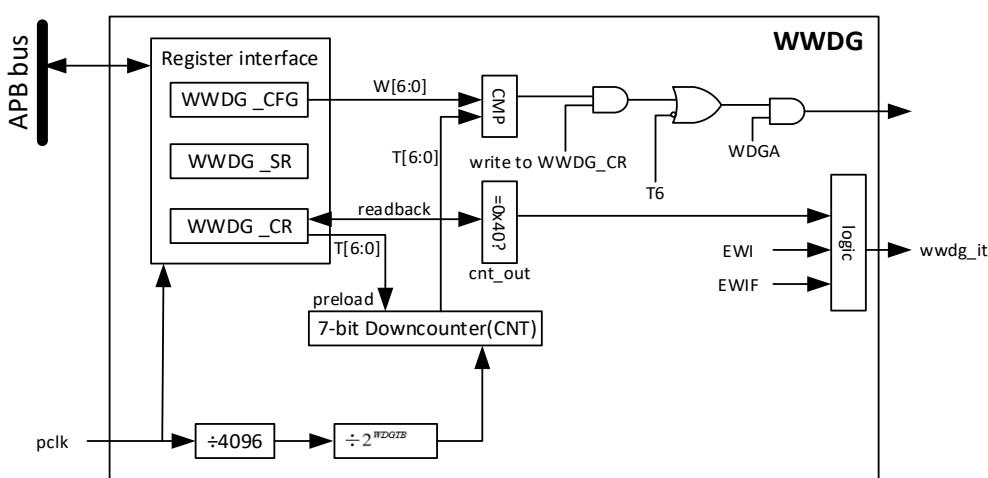


图 20-1 WWDG 模块框图

## 20.2. WWDG 功能描述

如果看门狗被启动(WWDG\_CR 寄存器中的 WDGA 位被置‘1’)，并且当 7 位(T[6:0])递减计数器从 0x40 翻转到 0x3F(T6 位清零)时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

应用程序在正常运行过程中必须定期地写入 WWDG\_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 WWDG\_CR 寄存器中的数值必须在 0xFF 和 0xC0 之间：

- 启动看门狗

在系统复位后，看门狗总是处于关闭状态，设置 WWDG\_CR 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

- 控制递减计数器

递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG\_CR 寄存器时，预分频值是未知的。配置寄存器(WWDG\_CFR)中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于 0x3F 时被重新装载，图 5-1 描述了窗口寄存器的工作过程。

另一个重装载计数器的方法是利用早期唤醒中断(EWI)。设置 WWDG\_CFR 寄存器中的 WEI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序(ISR)可以用来加载计数器以防止 WWDG 复位。在 WWDG\_SR 寄存器中写‘0’可以清除该中断。

注：可以用 T6 位产生一个软件复位(设置 WDGA 位为‘1’，T6 位为‘0’)。

## 20.3. 如何编写看门狗超时程序

可以使用图 5-1 中提供的公式计算窗口看门狗的超时时间。

警告：当写入 WWDG\_CR 寄存器时，始终置 T6 位为‘1’以避免立即产生一个复位。

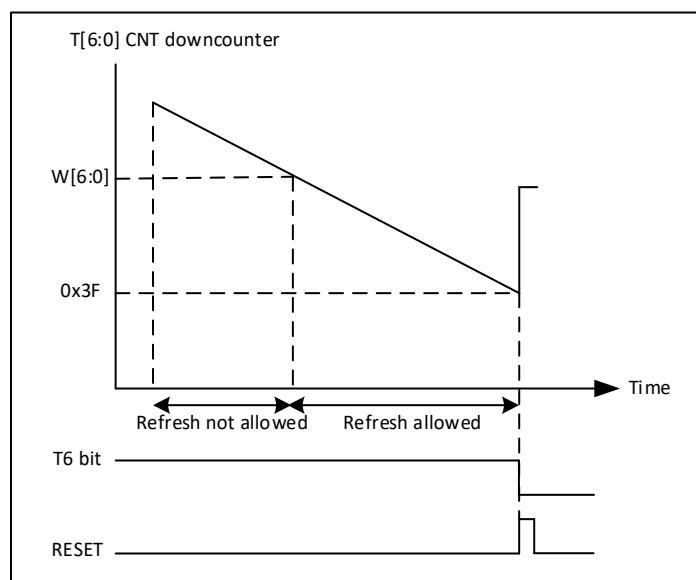


图 20-2 窗口看门狗时序图

计算 WWDG timeout 值的公式如下：

$$t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WWDG\_TB}[1:0]} \times (\text{T}[5:0] + 1) \quad (\text{ms})$$

## 20.4. 调试模式

当微控制器进入调试模式时(Cortex-M4 核心停止)，根据调试模块中的 DBG\_WWDG\_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。

## 20.5. 寄存器描述

WWDG 寄存器基地址：0x4000 2C00 - 0x4000 2FFF

### 20.5.1. 控制寄存器 (WWDG\_CR)

Address offset:0x00

Reset value:0x7F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								WDGA	T6	T5	T4	T3	T2	T1	T0
保留								RS	RW						

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
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: 禁止看门狗 1: 启用看门狗
6:0	T	RW	7F	T[6:0]: 7 位计数器(MSB 至 LSB) (7-bit counter) 这些位用来存储看门狗的计数器值。每( $4096 \times 2^{\text{WDGTB}}$ )个 PCLK1 周期减 1。当计数器值从 40h 变为 3Fh 时(T6 变成 0)，产生看门狗复位。

### 20.5.2. 配置寄存器 (WWDG\_CFR)

Address offset:0x04

Reset value:0x7F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EWI	WDGTB1	WDGTB0	W6	W5	W4	W3	W2
保留								RS	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	保留, 始终为 0			
9	EWI	RS	0	EWI: 提前唤醒中断 (Early wakeup interrupt) 此位若置‘1’，则当计数器值达到 40h，即产生中断。 此中断只能由硬件在复位后清除。
8:7	WDGTB	RW	0	WDGTB[1:0]: 时基 (Timer base) 预分频器的时基可以设置如下： 00: CK 计时器时钟(PCLK1 除以 4096)除以 1 01: CK 计时器时钟(PCLK1 除以 4096)除以 2 10: CK 计时器时钟(PCLK1 除以 4096)除以 4 11: CK 计时器时钟(PCLK1 除以 4096)除以 8
6:0	W	RW	7F	W[6:0]: 7 位窗口值 (7-bit window value) 这些位包含了用来与递减计数器进行比较用的窗口值。

### 20.5.3. 状态寄存器 (WWDG\_SR)

Address offset:0x08

Reset value:0x0000

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

保留	EWIF
保留	RC_W0

Bit	Name	R/W	Reset Value	Function
15:1	保留, 始终为 0			
0	EWIF	RC_W0	0	<p>EWIF: 提前唤醒中断标志 (Early wakeup interrupt flag)</p> <p>当计数器值达到 40h 时, 此位由硬件置' 1'。它必须通过软件写' 0' 来清除。对此位写' 1' 无效。若中断未被使能, 此位也会被置' 1'。</p>

#### 20.5.4. WWDG 寄存器映射

# 21. 外部串行存储控制器 (ESMC)

## 21.1. 简介

ESMC (External Serial Memory Controller) 是一种专用通信接口，用于单 (Single SPI)、双 (Dual SPI)、四 (Quad SPI) 和八 (Octal SPI) 通道 SPI 接口存储器 (NOR Flash, PSRAM 等)。它可以在以下两种模式中的任何一种模式下运行：

- 间接模式：所有操作均使用 ESMC 寄存器执行 (indirect mode)
- 内存映射模式：外部 Flash 映射到设备地址空间，系统将其视为内部存储器 (memory mapped mode)

使用双存储器模式，即同时访问两个 Quad SPI 存储器，可以实现类似 Octal SPI 存储器一样提高两倍的吞吐量和存储容量。

## 21.2. 主要特征

- 两种功能模式：间接和内存映射
- 可同时发送/接收 8 位
  - 双 Flash 模式，通过并行访问两个 Flash，可同时发送/接收 8 位
  - Octal SPI
- SDR 和 DDR 支持
- 用于间接和内存映射模式的完全可编程操作码
- 间接和内存映射模式的完全可编程帧格式
- 用于接收和传输的集成 FIFO
- 允许 8 位、16 位和 32 位数据访问
- 用于间接模式操作的 DMA 信道
- FIFO、操作完成上的中断生成

## 21.3. 功能说明

### 21.3.1. 模块框图

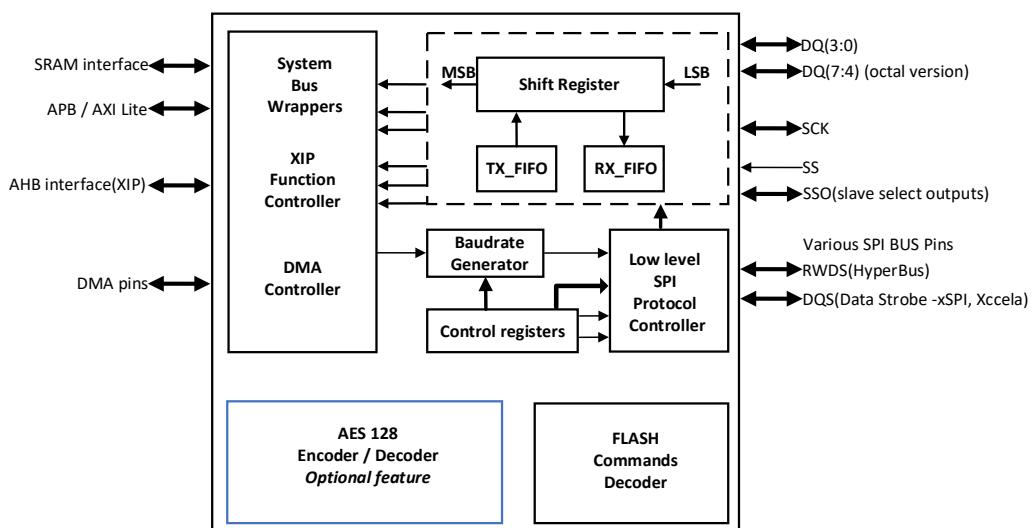


图 21-1 ESMC 模块框图

### 21.3.2. ESMC 功能描述

#### 21.3.2.1. ESMC 外部存储器连接图

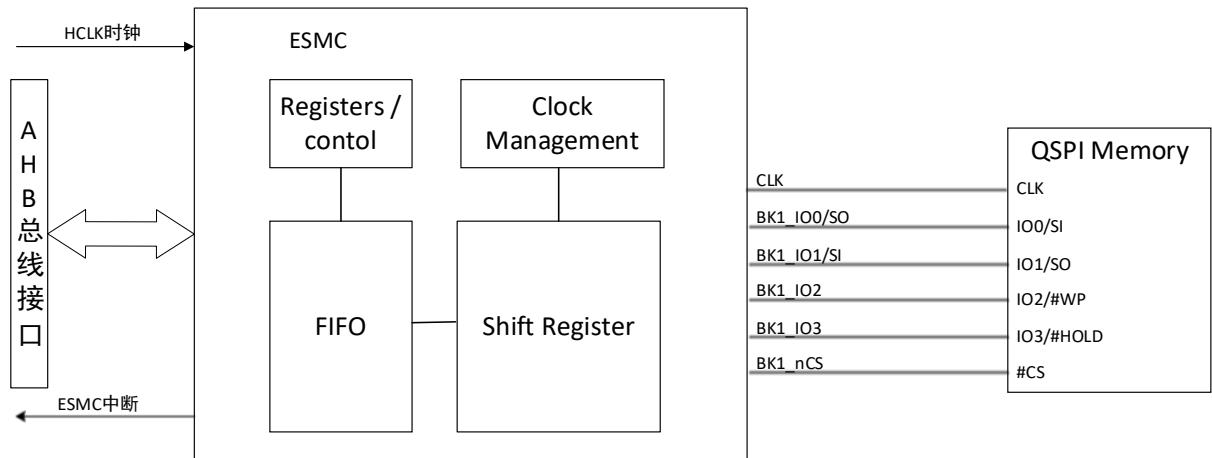


图 21-2 ESMC 连接图 (Dual QSPI Memory 模式禁止时)

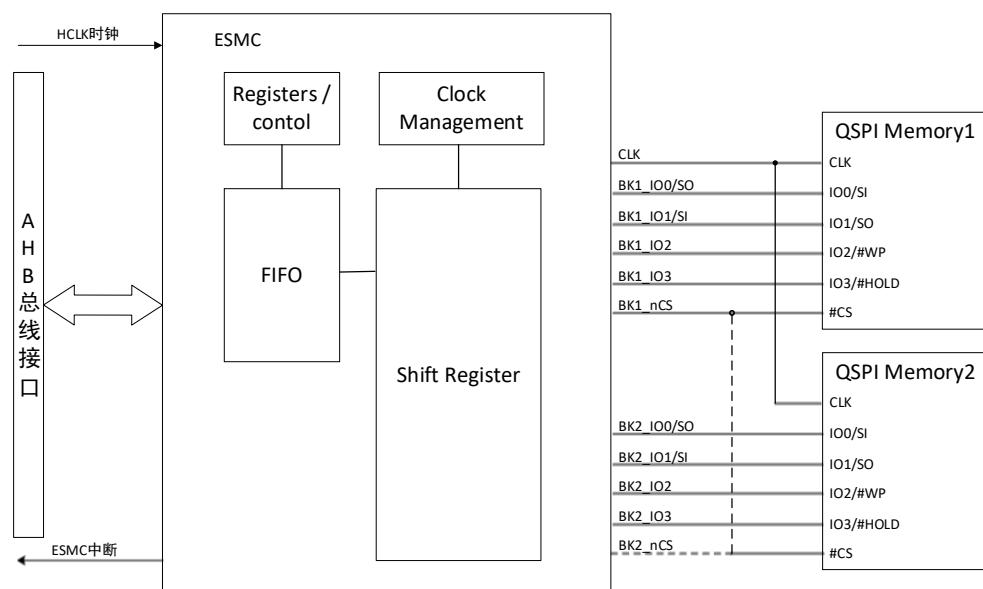


图 21-3 ESMC 连接图 (Dual QSPI Memory 模式设置时)

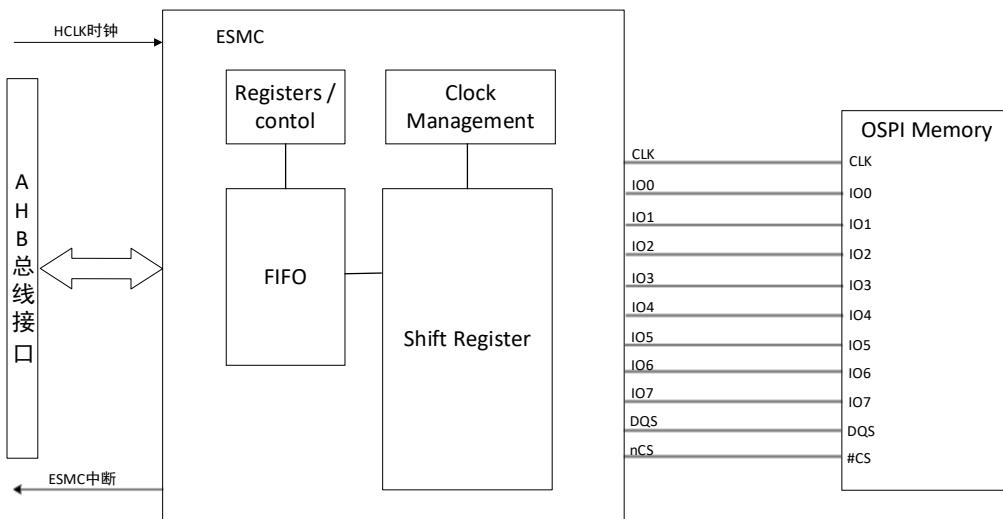


图 21-4 ESMC 连接图 (OSPI Memory 模式设置时)

### 21.3.2.2. ESMC 命令序列

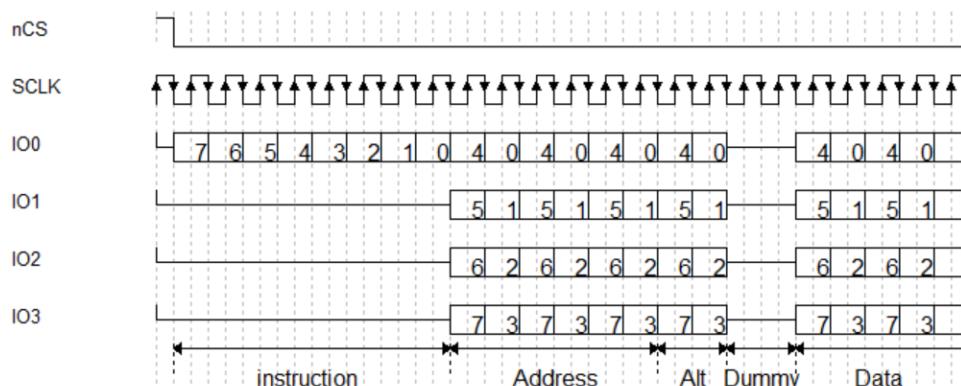


图 21-5 QSPI 指令示例

ESMC 使用命令与外部存储器通信。每个命令可以包括 5 个阶段：指令、地址、备用字节、Dummy、数据。可以将这些阶段中的任何一个配置为跳过，但必须至少存在指令、地址、备用字节或数据阶段中的一个。

nCS 在每个命令开始前下降，在每个命令完成后再次上升

#### 指令阶段

在此阶段，在 ESMC\_SFCR[7:0]或 ESMC\_ADDR24[7:0](上述两个 INS 寄存器在电路中为同一个寄存器)寄存器的指令字段中配置的 8 位指令被发送到 Flash，指定要执行的操作类型。

尽管大多数 Flash 每次只能从 IO0 信号（单 SPI 模式）接收一位指令，但指令阶段可以选择一次发送 2 位（双 SPI 模式下 IO0/IO1）、一次发送 4 位（四 SPI 模式下 IO0/IO1/IO2/IO3）或一次发送 8 位（八 SPI 模式下 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7）。

#### 地址阶段

在地址阶段，1 至 4 个字节被发送到 Flash，以指示操作的地址。要发送的地址其字节数在 ESMC\_CR3 寄存器的 32ADDRBIT、16ADDRBIT、8ADDRBIT 位配置，上述三 bit 都为 0 时，则地址的位宽为 24bit。在间接模式下，要发送的地址字节在 ESMC\_ADDR24 中的 ADDR0~2 和 ESMC\_ADDR32 中的 ADDR3 寄存器指定，而在内存映射模式下，地址直接通过 AHB（从 Cortex®或 DMA）给出。

地址可以一次发送 1 位（在单 SPI 模式下通过 SO），一次发送 2 位（在双 SPI 模式下通过 IO0/IO1），一次发送 4 位（在四 SPI 模式下通过 IO0/IO1/IO2/IO3）或一次发送 8 位（八 SPI 模式下 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7）。

当 ESMC\_DCR 寄存器 NO\_ADDR 位设置时，将跳过地址阶段，命令序列将直接进入下一阶段。

### **Alternate 字节阶段**

在备用字节阶段，1 个字节被发送到 Flash，通常用于控制操作模式。要发送的备用字节数据在 ESMC\_ADDR32[15:8]寄存器的 MREG[7:0]字段中配置。

备用字节阶段可以一次发送 2 位（在双 SPI 模式下通过 IO0/IO1），或一次发送 4 位（在四 SPI 模式下通过 IO0/IO1/IO2/IO3）。

当 ESMC\_SOCSR 的 SENM 或 ESMC\_XSOCSR 的 SENM=0 时，将跳过备用字节阶段，命令序列将直接进入下一阶段（如果有）。

### **Dummy 周期阶段**

在 DUMMY 周期阶段，在不发送或接收任何数据的情况下给出 1-31 个周期，以便 Flash 在使用更高时钟频率时有时间准备数据。该阶段给出的周期数在 ESMC\_DCR 寄存器的 DUMMY[4:0]字段中指定。在 SDR 和 DDR 模式中，持续时间都指定为完整 CLK 周期数。

当 DUMMY 为零时，跳过 DUMMY 周期阶段，命令序列直接进入数据阶段（如果存在）。

### **数据阶段**

在数据阶段，可以向 Flash 发送或从 Flash 接收任意数量的字节。

在间接模式下，发送/接收数据的字节数在 ESMC\_TCR 寄存器中指定。

在间接写入模式下，发送至 Flash 的数据必须写入 ESMC\_DATA 寄存器，而在间接读取模式下，从 Flash 接收的数据通过读取 ESMC\_DATA 寄存器获得。

在内存映射模式下，从 Flash 接收的数据通过读取 ESMC\_DATA 寄存器获得。

在 DMA 模式下，读取的数据直接通过 AHB 发送回 DMA。

数据阶段一次可以发送/接收 1 位（在单 SPI 模式下通过 SO/SI）、2 位（在双 SPI 模式下通过 IO0/IO1）、4 位（在四 SPI 模式下通过 IO0/IO1/IO2/IO3）或八位（八 SPI 模式下 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7）。

## **21.3.2.3. ESMC 接口协议模式**

### **Single SPI 模式**

传统 SPI 模式只允许串行发送/接收单个位。在此模式下，数据通过 SO/IO0 发送到 Flash。从 Flash 接收的数据通过 SI/IO1 到达。

通过将 ESMC\_SOCSR 或 ESMC\_XSOCSR 的 SPIMODE 设置为 000，以使用单 SPI 模式。

### **Dual SPI 模式**

在双 SPI 模式下，通过 IO0/IO1 信号同时发送/接收两位数据。

通过将 ESMC\_SOCSR 或 ESMC\_XSOCSR 的 SPIMODE 设置为 001，以使用双 SPI 模式。

### **Quad SPI 模式**

在四 SPI 模式下，通过 IO0/IO1/IO2/IO3 信号同时发送/接收四位数据。

通过将 ESMC\_SOCSR 或 ESMC\_XSOCSR 的 SPIMODE 设置为 010，以使用四 SPI 模式。

### **Dual Flash 模式**

当 2XQSPI 位 (ESMC\_CR[2]) 为 1 时，ESMC 处于双 Flash 模式，其中使用两个外部 QUAD SPI Flash (Flash1 和 Flash2)，以便每个周期发送/接收 8 位数据（或 DDR 模式下的 16 位数据），有效地将吞吐量和容量加倍。

每个 Flash 使用相同的 CLK 和可选的相同 nCS 信号，但每个都有单独的 IO0、IO1、IO2 和 IO3 信号。

双 Flash 模式可与单位、双位和四位模式以及 SDR 或 DDR 模式结合使用。

每个 byte 7:4 位数据存放在 Flash1，每个 byte 3:0 位数据存放在 Flash2。

在双 Flash 模式下读取 Flash 状态寄存器时，与在单 Flash 模式下读取相同字节相比，应读取两倍的字节。这意味着，如果每个 Flash 在获取状态寄存器的指令后给出 8 个有效位，则必须为 QUAD SPI 配置 2 字节（16 位）的数据长度，并且 QUADSPI 从每个 Flash 接收一个字节。如果每个 Flash 的状态为 16 位，则必须将 QUAD SPI 配置为读取 4 个字节，以在双 Flash 模式下获取两个 Flash 的所有状态位。

结果的最低有效字节（在数据寄存器中）是 Flash1 状态寄存器的最低有效字节，而下一个字节是 Flash2 状态寄存器的最低有效字节。然后，数据寄存器的第三个字节是 Flash1 第二字节，而第四个字节是 Flash2 第二字节（在 Flash 具有 16 位状态寄存器的情况下）。

在双 Flash 模式下，必须始终访问偶数个字节。

### **Octal SPI 模式**

在八 SPI 模式下，通过 IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7 信号同时发送/接收八位数据。

通过将 ESMC\_SOCSR 或 ESMC\_XSOCSR 的 SPI MODE 设置为 100，以使用八 SPI 模式。

### **SDR 模式**

默认情况下，DDR 位 (ESMC\_SOCSR 或 ESMC\_XSOCSR 的 DDRCMD、DDRADDR、DDRRDATA) 为 0，ESMC 以单数据速率 (SDR) 模式运行。

在 SDR 模式下，当 ESMC 驱动 IO0/SO、IO1、IO2、IO3 信号时，这些信号仅随 CLK 下降沿转换。

在 SDR 模式下接收数据时，Flash 使用 CLK 的下降沿发送数据，ESMC 使用 CLK 的上升沿对信号进行采样。

### **DTR 模式**

当 DDR 位 (ESMC\_SOCSR 或 ESMC\_XSOCSR 的 DDRCMD、DDRADDR、DDRRDATA) 设置为 1 时，ESMC 以双数据速率 (DDR) 模式运行。

在 DDR 模式下，当 ESMC 在地址/备用字节/数据阶段驱动 IO0、IO1、IO2、IO3、IO4、IO5、IO6、IO7 信号时，在 CLK 的每个下降沿和上升沿上发送一位数据。

在 DDR 模式下接收数据时，Flash 使用上升和下降 CLK 边缘发送数据。因此，在 CLK 周期的一半之后（在下一个相反的边缘）ESMC 对信号进行采样

#### **21.3.2.4. 使用 ESMC**

ESMC 工作模式通过 XIPEN 位 (ESMC\_CR[6]) 设置，XIPEN=0, 此时为间接模式，否则为 XIP 内存映射模式。

#### **ESMC 间接模式**

在此模式下，通过 ESMC 控制寄存器读取和写入 SPI Flash 设备。从 ESMC 数据缓冲区读取或写入到 SPI Flash 的数据。因此，读取 Flash 时，CPU 会从 ESMC 连续读取相同的 RX FIFO 位置。下表显示了简化的 AMBA 总线系统，SPI 控制器可通过 AHB 内存空间访问。来自 SPI Flash 的数据可以通过 AHB 总线空间访问，而所有 SPI 控制器内部寄存器都可以通过 AHB 总线空间访问。

表 21-1 ESMC 间接模式

CPU Addr READ/WRITE	ADDR Bus	SPI Data Bus	Flash Memory
SPI TX/RX Buffer	0x0xx10	[0x0000]	0x0000
	0x0xx10	[0x0001]	0x0001
	0x0xx10	[0x0002]	0x0002
	0x0xx10		
	0x0xx10	[0xFFFF]	0xFFFF

### ESMC 内存映射模式

在内存映射模式下， SPI Flash 与任何其他并行内存一样映射到系统总线。读取选定的存储器位置时， CPU 从 SPI Flash 读取相应的存储器单元。 SPI 控制器使用 SPI Flash 命令，其地址为 AHB 访问的映射地址。使用此模式需要主机配置 SPI 控制器的操作，此设置也可以在 reset 期间预设。

表 21-2 ESMC 内存映射模式

CPU Addr READ/WRITE	ADDR Bus	SPI Data Bus	Flash Memory
SPI Flash location	Addr0	[0x0000]	0x0000
	Addr1	[0x0001]	0x0001
	Addr2	[0x0002]	0x0002
	...		
	Addr N	[0xN]	0xN

在多 SPI 模式下，传输方向由读写命令确定。从外部 SPI 设备写入和读取数据需要 ESMC 生成 SCK 时钟脉冲。发送数据时，主机 CPU 需要向 FIFO 中写入需要发送的数据。接收数据时，将接收到的数据存储在 FIFO 中。接收数据需要主机软件在 ESMC\_DCR 寄存器中设置位 7 (REC:Reception) 。在任何方向的传输开始之前，主机需要通过所选的 SSO/nCS 驱动为低来寻址 SPI 从设备。要结束当前传输，SSO/nCS 线路需要被驱动为高电平。

#### 间接模式写数据到外部串行存储器

- 用适当的值加载 ESMC 控制寄存器(ESMC\_CR、ESMC\_CR2、ESMC\_TCR、ESMC\_BAUD、ESMC\_SFCR、ESMC\_SOCR、ESMC\_DCR、ESMC\_CR3)
- 用 Flash 单元地址加载地址寄存器 (ESMC\_ADDR24, ESMC\_ADDR32)
- 用要发送的 Flash 命令加载到指令寄存器 (ESMC\_SFCR[7:0], ESMC\_ADDR24[7:0])
- 使用要发送的字节数据加载到数据缓冲区/FIFO 寄存器 (ESMC\_DATA)
- 继续加载数据缓冲区 FIFO，直到最后一个字节发送完成
- 通过写入 ESMC 控制寄存器位 SS\_CLEAR\_REQUEST (ESMC\_CR3[6]) ，清除 SSO 来解除对存储器 CS 信号的选中

#### 间接模式从外部串行存储器读数据

- 用适当的值加载 ESMC 控制寄存器(ESMC\_CR、ESMC\_CR2、ESMC\_TCR、ESMC\_BAUD、ESMC\_SFCR、ESMC\_SOCR、ESMC\_DCR、ESMC\_CR3)
- 用 Flash 单元地址加载地址寄存器 (ESMC\_ADDR24, ESMC\_ADDR32)
- 用要发送的 Flash 命令加载到指令寄存器 (ESMC\_SFCR[7:0], ESMC\_ADDR24[7:0])
- 等待数据缓冲区写入数据。
- 从数据缓冲区 (ESMC\_DATA) 读取数据，直到要读取的最后一个字节。
- 通过写入 ESMC 控制寄存器位 SS\_CLEAR\_REQUEST (ESMC\_CR3[6]) ，清除 SSO 来解除对存储器 CS 信号的选中。

## 内存映射模式从外部串行存储器读数据

- 用适当的值加载 ESMC 控制寄存器(ESMC\_CR、ESMC\_CR2、ESMC\_TCR、ESMC\_BAUD、ESMC\_XSFCR、ESMC\_XSOCR、ESMC\_DCR、ESMC\_CR3)
- 用 Flash 单元地址加载地址寄存器 (ESMC\_ADDR24, ESMC\_ADDR32)
- 用要发送的 Flash 命令加载到指令寄存器 (ESMC\_SFCR[7:0], ESMC\_ADDR24[7:0])
- 使用 AHB 接口直接访问内存映射地址
- 等待数据缓冲区写入数据。
- 从数据缓冲区 (ESMC\_DATA) 读取数据，直到要读取的最后一个字节。
- 通过写入 ESMC 控制寄存器位 SS\_CLEAR\_REQUEST (ESMC\_CR3[6])，清除 SSO 来解除对存储器 CS 信号的选中。

在读取方向上，数据传输足以满足内存需求。一些存储器要求在开始读取之前将 Dummy 数据传输到存储器。在这种情况下，用户应使用要发送的正确 Dummy 周期数对 ESMC\_DCR (DUMMY 控制寄存器) 进行编程。在 Dummy 传输期间，IO 线保持在高阻抗状态。

### 21.3.2.5. ESMC 中断

可在以下事件中产生中断：

#### DATA\_WAIT

当前执行 Flash 读写操作，且 ESMC 状态机处于等待状态

- 在写操作期间，发送了命令字节和地址，ESMC 正在等待发送下一部分数据字节。FIFO/数据缓冲区为空。
- 在读取操作中，ESMC 填充整个数据缓冲区，并等待主机清空数据缓冲区

#### SPI IDLE

ESMC IDLE 状态

#### FIFO Overflow

FIFO 溢出，FIFO 满状态下发生写 FIFO 动作

#### FIFO Full, FIFO Half, FIFO Empty, FIFO Not Empty

FIFO 处于不同状态

独立的中断使能位保证了灵活性。

表 21-3 中断使能位

Interrupt Event	Event Flag	Enable Control Bit
SPI Busy	DATA_WAIT_FLAG	DATA_WAIT_IE
SPI Idle	IDLE_FLAG	IDLE_IE
FIFO Overflow	FIFO_OVF	FIFO_OVIE
FIFO Full	FIFO_FF	FIFO_FIE
FIFO Half	FIFO_HF	FIFO_HIE
FIFO Empty	FIFO_EF	FIFO_EIE
FIFO Not Empty	FIFO_NEF	FIFO_NEIE

### 21.3.3. ESMC 寄存器描述(ONE BYTE 访问)

ESMC 寄存器基地址: 0xA000 1000

#### 21.3.3.1. 配置寄存器 (ESMC\_CR)

Address offset: 0x00

Reset value: 0xF8

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIEN	XIP EN	Res	GIE	DMAEN	2xQSPI	Res	SOFTRST
RW	RW		RW	RW	RW		RW

Bit	Name	R/W	Reset Value	Function
7	SPIEN	RW	1	SPI 系统使能 该位由软件置位或清除 0: SPI 系统禁止, 所有内部寄存器和计数器停止, 降低功耗 1: SPI 系统使能
6	XIPEN	RW	1	启用/禁用 XIP 该位由软件置位或清除。此位值的任何更改, 终止任何 SPI 操作, 清除 TX 和 RX FIFO, 并将 ESMC 驱动到空闲状态。清除 XIP EN 位后, 只能通过寄存器空间编程的命令和操作 0: 禁止 XIP 功能, 使能间接模式 1: 使能 XIP 功能, 禁止间接模式 注: XIP 为内存映射模式
5	Reserved			
4	GIE	RW	1	全局中断使能(GLOBLE INTERRUPT ENABLE) 0: 由各中断使能位控制 1: 所有中断使能
3	DMAEN	RW	1	DMA 使能 该位由软件置位或清除, 在 TXDMAREQ 和 RXDMAREQ 上启用 DMA 请求。清除时, TXDMAREQ 和 RXDMAREQ 保持零未激活。功能仅在特殊要求时可用, 否则此位将被忽略。 0: DMA 禁止 1: DMA 使能
2	2XQSPI	RW	0	双 Flash 模式 (Dual Flash Mode) 该位由软件置位或清除, 此位的工作方式与 OSPI 类似, 不同之处在于此模式下允许使用 2 个 QSPI flash 操作。命令和地址行通过 IO0~IO4 行发送。 0: 单 Flash 模式 1: 双 Flash 模式
1	Reserved			
0	SOFTRST	RW	0	软件复位 (Software Reset) 0: 1: 注意: 本复位寄存器, 在内部逻辑中, 当总线发出写本寄存器为 1 的命令后, 间隔一个 ESMC 的模块时钟后, 在第二个时钟开始时 soft_rst 拉高为 1, 当在第二个时钟结束时 soft_rst 自动清除为 0。总结: 间隔一个时钟拉高, 拉高保持一时钟后自动清除为 0。

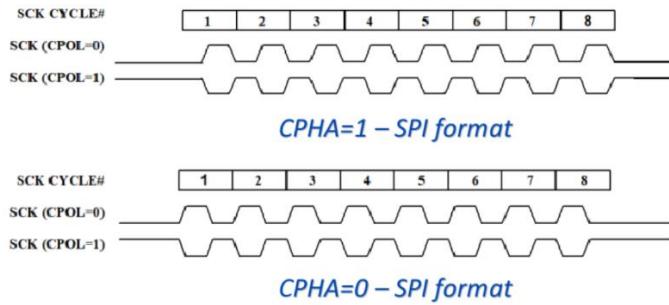


图 21-6 串行接口模式

### 21.3.3.2. 配置寄存器 2 (ESMC\_CR2)

Address offset: 0x01

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
						CPOL	CPHA
						RW	RW

Bit	Name	R/W	Reset Value	Function
7:2	Reserved			
1	CPOL	RW	0	时钟极性 (clock polarity), 该位由软件置位或清除 0: 空闲状态时, SCK 保持低电平 1: 空闲状态时, SCK 保持高电平
0	CPHA	RW	0	时钟相位 (clock phase), 该位由软件置位或清除 0: 从第二个时钟边沿开始采样数据 1: 从第一个时钟边沿开始采样数据

### 21.3.3.3. 传输计数寄存器(ESMC\_TCR)

Address offset: 0x02

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCR[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	TCR[7:0]	R/W	0x00	<p>读取数据个数 此寄存器仅在启用缓冲内存时使用。 在 Flash 读取模式下, ESMC 可在两种主要读取模式下工作: -从 Flash 读取未定义数量的数据字节-模式 0(连续读) -从 Flash 读取指定数量的数据字节-模式 1。 默认情况下, 只要接收器数据缓冲区至少有一个可用空间, ESMC 就会读取数据。一旦缓冲区已满, ESMC 进入等待状态, 等待主机从缓冲区读取数据。只要数据缓冲区中至少有一个位置为空, ESMC 就会恢复读取操作。 在读取指定字节数模式下, ESMC 读取 TCR 寄存器中指定的字节数。 此模式仅在非 XIP 操作模式下可用。TCR=0x00 的读取操作, 表示如上所述的连续读取模式。 在模式 1 中, 读取指定数量的字节后, SSO 行被停用, ESMC 进入空闲状态, 等待下一个命令执行。 该寄存器允许用户命令接收 1-255 字节的数据。对于较大字节数的接收, 用户应使用连续接收模式 0。</p>

### 21.3.3.4. 波特率寄存器(ESMC\_BAUD)

Address offset: 0x03

Reset value: 0x04

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BAUD[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	BAUD[7:0]	R/W	0x04	波特率寄存器用于定义 SCK 时钟线相对于系统时钟的速度。加载此寄存器时，用户定义用于生成 SCK 信号的 CLK 预分频器值。可接受值的范围为：2~255 的偶数。禁止使用值 0x00 和 0x01，将 0x00 或 0x01 加载到波特率寄存器会导致生成 SCK=CLK/2。 串行 SPI 传输产生的 SCK 时钟频率根据以下公式计算：SCK = CLK/QBAND

### 21.3.3.5. SPI Flash 命令寄存器(ESMC\_SFCR)

Address offset: 0x04

Reset value: 0x0B

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INS0[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	INS0[7:0]	R/W	0x0B	SPI 指令 (Command Only)  指令格式只包含命令字节。地址，备用，Dummy，数据字节都不发送。  默认情况下，它以扩展 SPI 格式发送，但是对于选定的 SPI 操作模式，它也可以分别以双模式、四模式或 QCTAL 模式发送。无论 DDR 控制位值如何，该命令永远不会在 DDR 模式下发送。

### 21.3.3.6. SPI 输出控制寄存器(ESMC\_SOCR)

Address offset: 0x05

Reset value: 0x02

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE1	SPI_MODE0
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	DDDRDATA	RW	0	设置后，在数据接收/传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。
6	DDRADDR	RW	0	设置后，在 Flash 地址字节传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。
5	DDRCMD	RW	0	设置后，在 Flash 命令字节传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。
4	MULTIADDR	RW	1	设置后，启用快速 I/O 读取，其中地址位分别以双、四或八进制格式发送，如果 MULTICMD 清除，则表示地址以单 SPI 模式发送。  在非扩展 SPI 传输 (MULTI_COMM=1) 中，地址字节以与数据字节相同的格式发送。
3	MULTICMD	RW	1	未设置时，启用扩展 SPI 操作。这意味着 Flash 命令和地址通过 DQ0 线以单 SPI 模式发送，而不管八进制、四进制和双 SPI 位值如何。
2	SENM	RW	0	间接模式下，在 ADDR(8 位、16 位、24 位或 32 位)之后发送 MREG[7:0]。
1:0	SPI MODE[1:0]	RW	0x2	设置 SPI 传输格式： 2'b00 SINGLE SPI 2'b01 DUAL SPI

Bit	Name	R/W	Reset Value	Function
				2'b10 QUAD SPI 2'b11 OCTAL SPI

MUL\_COMM 位启用扩展 SPI 协议，其中命令和地址仅在 DQ0 行上发送，数据在编程行数上发送/接收（选择双 SPI 时为 IO[1:0]，设置四位时为 IO[3:0]，设置八位时为 IO[7:0]）。具有多通信位集的双/四/八进制 SPI 位集意味着分别选择 SPI 的双/四/八进制模式。下图显示了特定模式的数据传输差异。

下图显示了几种受支持的 Flash 读/写操作及其格式。程序员需要手动控制 SPI 命令、地址、虚拟和数据线，如下所述。提供的控制寄存器允许设置 Flash 命令的任意组合，包括多 SPI 线命令、所有命令、地址和数据的 DDR 选项以及任意数量的虚拟周期。

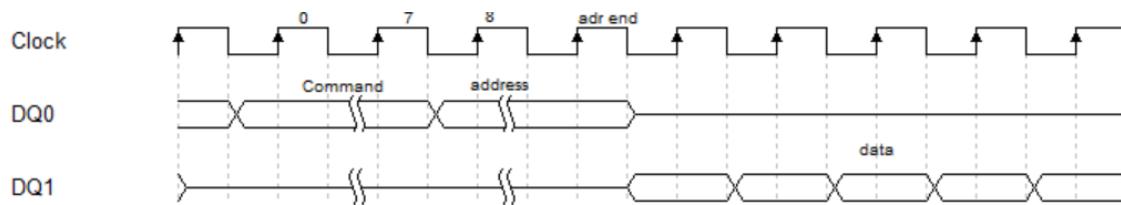


图 21-7 快速读指令 I/O Fast Read =0; MULTI COMM =0; SPI\_MODE=2'b00



图 21-8 快速读指令 DUAL I/O Fast Read =0; MULTI COMM =1; SPI\_MODE=2'b01

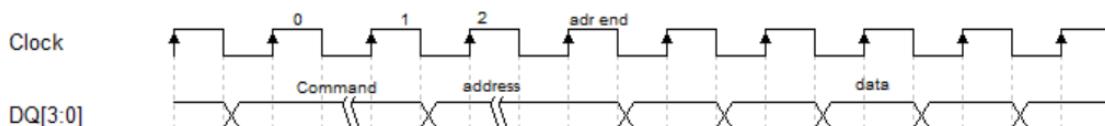


图 21-9 快速读指令 QUAD I/O Fast Read =0; MULTI COMM =1; SPI\_MODE=2'b10

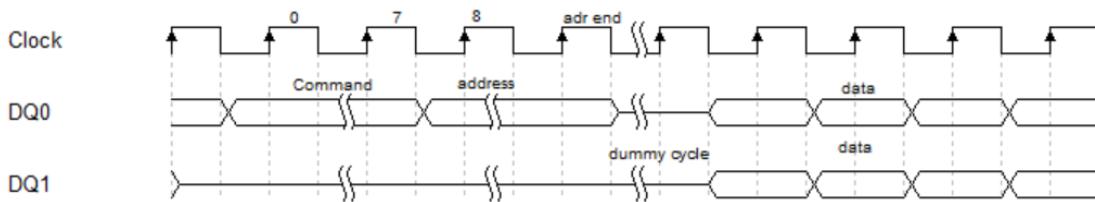


图 21-10 双路输出 快速读指令 I/O Fast Read =0; MULTI COMM =0; SPI\_MODE=2'b01; DUMMY !=0

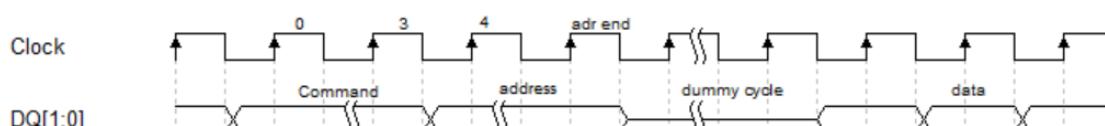
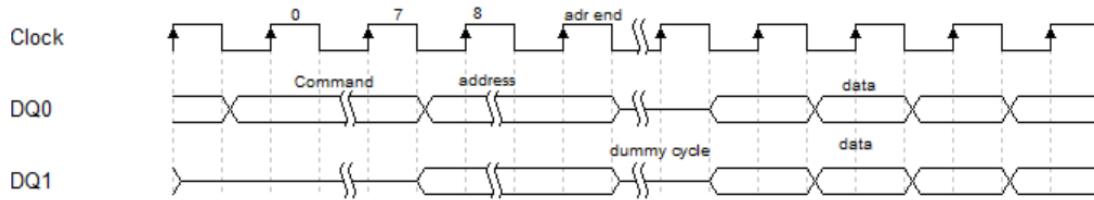
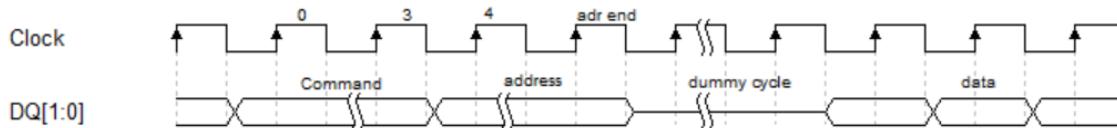


图 21-11 双路输出 快速读指令 I/O Fast Read = 0; MULTI COMM = 1; SPI\_MODE = 2'b01; DUMMY != 0

图 21-12 双路 I/O 快速读指令 I/O Fast Read = 1; MULTI COMM = 0; SPI\_MODE = 2'b01  
DUMMY != 0图 21-13 双路 I/O 快速读指令 I/O Fast Read = 1; MULTI COMM = 1; SPI\_MODE = 2'b01  
DUMMY != 0

四通道和八通道读取操作与双通道类似，只是数据在多个 I/O 线路上传输。

#### 21.3.3.7. Dummy 控制寄存器(ESMC\_DCR)

Address offset: 0x06

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REC	NO_CMD	NO ADDR		DUMMY[4:0]			
RW	RW	RW		RW			

Bit	Name	R/W	Reset Value	Function
7	REC	RW	0	接收数据 (Reception) 设置后，ESMC 应执行 Flash 读取命令，在发送命令、地址和（如果需要）DUMMY 字节后，ESMC 开始接收数据字节。 如果设置了这个位，ESMC 接收数据(读模式)，否则 ESMC 发送数据(写模式)。 注意：TCR[7]=0 会开启连续读取模式。ESMC 执行读操作，直到 FIFO 被完全填满，并在软件清空 FIFO 后继续执行 read 命令。然后它恢复数据接收。
6	NO_CMD	RW	0	该位用于 SPI NOR Flash 连续读取的手动操作模式(没有地址解码器)。将会省略命令，而以地址开始传输。
5	NO_ADDR	RW	0	设置 NO_ADDR 位后，ESMC 不发送地址字节，命令字节后面跟着 DUMMY CYCLES(如果是 eny) 和 DATA Bytes
4:0	DUMMY[4:0]	RW	5'b00000	定义任何地址传输后生成的 DUMMY 周期数。 11111: 31 个 dummy 周期 11110: 30 个 dummy 周期 ... 00001: 1 个 dummy 周期 00000: 无 dummy 周期

在某些配置中，串行 Flash 要求在地址字节之后发送 DUMMY 周期。ESMC\_DCR[4:0] 定义要发送的 DUMMY 周期数，允许设置为 31 个 DUMMY 周期。当命令需要 DUMMY 时，DUMMY 将自动发送，但如果特定操作需要，则必须在

位[4:0]上定义要发送的 DUMMY 周期数。默认情况下，DUMMY 周期数设置为 8。但时钟频率有变化时，或者 Flash 的数据准备时间过长时，程序员也应该设置要发送的 DUMMY 周期数。当 DUMMY[4:0]=0 时，不会发送任何 DUMMY 周期。与 0 不同的 ESMC\_DCR[4:0]会导致 ESMC 在地址字段后发送编程的 DUMMY 周期数。在发送 DUMMY 周期期间，DQ 线路被设置为高阻抗状态，并且没有数据存储在接收器缓冲器（RX FIFO）中，也没有数据被发送出去。

### 21.3.3.8. 配置寄存器 3(ESMC\_CR3)

Address offset: 0x07

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SS SET	SS CLEAR	FIFO CLR		Res	32BIT_ADDR	16BIT_ADDR	8BIT_ADDR
RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	SS_SET_RQ	RW	0	从机选择输出设置请求，将逻辑 1 写入该位会导致由 SS[3:0]选择的从机选择输出(SSxO)被驱动为低电平。一旦 SSxO 输出激活，该位将自动清除。
6	SS_CLR_RQ	RW	0	从机选择输出清除请求，将逻辑 1 写入该位会导致从机选择输出(SSxO)被驱动为高电平。该位自动清除。
5	FIFO CLR	RW	0	清除所有 FIFO 内存指针
4:3	Reserved	RW	0	
2	ADDR32BIT	RW	-	启用 32 位地址，当 ESMC_CR3[2:0]=000 时，地址位宽为 24bit
1	ADDR16BIT	RW	0	启用 16 位地址
0	ADDR8BIT	RW	0	启用 8 位地址

### 21.3.3.9. 状态寄存器(ESMC\_SR)

Address offset: 0x14

Reset value: 0x10

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIF	FIFO OV	Res	IDLE STATE	Res	TX in pro-gress	RX in pro-gress	SS_ACTIVE
R	R		R		R	R	R

Bit	Name	R/W	Reset Value	Function
7	SPIF	R	0	SPI 中断请求标志(SPI INTERRUPT FLAG) 0: 没有中断请求挂起 1: 至少有一个中断请求挂起。 该标志将一直置高，直到清除所有中断请求。
6	FIFO overflow	R	0	FIFO overflow 标志位
5	Reserved			
4	IDLE	R	1	ESMC 处于 IDLE 状态标志位
3	Reserved			
2	TXBUSY	R	0	控制器忙于内存写操作的标志位
1	RXBUSY	R	0	ESMC 忙于接收来自 Flash 的数据的标志位 1: ESMC 从外部 Flash 读取数据。 0: RX FIFO Full 没有完成任何读操作，或者完成指定字节数的读操作，或者停止连续读操作
0	SSACTIVE	R	0	如果任何 SS 输出是有效的话，此位为 0

### 21.3.3.10. 中断标志寄存器(ESMC\_IFR)

Address offset: 0x15

Reset value: 0x02

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR DONE	FIFO OV	DATA WAIT	IDLE	FIFO Full	FIFO HALF Full	FIFO Empty	COMMAND DONE
R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIF	R	0	地址发送完成标志(ADDRESS DONE FLAG) 当地址字段刚刚发出时设置位。通过将新命令加载到 SFCR 寄存器或将 1 写入 ADDR Done 标志位置，位自动清除。
6	FIFOIF	R	0	FIFO 溢出标志 (FIFO OVERFLOW FLAG) -FIFO/数据缓冲区溢出。缓冲区已满时发生数据写入，缓冲区中加载的字节太多。
5	DATAWAITIF	R	0	此位为 1 表示 ESMC 当前执行 Flash 读写操作，且 ESMC 状态机处于等待状态，这意味着： 在写操作期间，发送了命令字节和地址，ESMC 正在等待发送下一部分 od 数据字节。 FIFO/数据缓冲区为空。 在读取操作中，ESMC 填充整个数据缓冲区，并等待主机清空数据缓冲区。 数据缓冲区清空或加载新数据部分后，位被清除，以便 ESMC 恢复当前 Flash 读/写操作。 0: 未进入 data wait 阶段，当前 transfer 未结束 1: 进入 data wait 阶段，当前 transfer 已经结束
4	IDLEIF	R	0	空闲状态标志。 0: 操作中，正在进行命令、地址、Dummy 或数据传输。 1: ESMC 处于空闲状态。
3	FIFOIF	R	0	FIFO 满 (FIFO FULL FLAG) -当 set FIFO 完全满时，FIFO 内存中没有可用空间。 清除 0 时-FIFO 中至少有 1 个空空间。
2	FIFOHIF	R	0	FIFO 半满(FIFO HALF FLAG) -FIFO 内存已满一半。
1	FIFOEIF	R	1	FIFO 空(FIFO EMPTY FLAG) 0: 1: (Data Buffer 和 FIFO 为空时设置)。
0	CMDIF	R	0	指令发送完成标志(COMMAND DONE FLAG) 0: 在 INS 载入新的指令或对该位写 1 时清除 1: 指令字节发送完成时置位。在 INS 载入新的指令或对该位

### 21.3.3.11. 中断使能寄存器(ESMC\_IER)

Address offset: 0x16

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR DONE IE	FIFO OV IE	DATA WAIT IE	IDLE IE	FIFO Full IE	FIFO HALF Full IE	FIFO EmptyIE	COMMAND DONE IE
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIE	RW	0	地址发送完成中断使能(ADDRESS DONE INTERRUPT ENABLE) 0: 地址发送完成中断禁止 1: 地址发送完成中断使能
6	FIFOIFIE	RW	0	FIFO 溢出中断使能(FIFO OVERFLOW INTERRUPT ENALBE)

Bit	Name	R/W	Reset Value	Function
				0: FIFO 溢出中断禁止 1: FIFO 溢出中断使能
5	DATAWAITIE	RW	0	Busy 中断使能 0: BUSY 中断禁止 1: BUSY 中断使能
4	IDLEIE	RW	0	空闲中断使能 0: 空闲中断禁止 1: 空闲中断使能
3	FIFOFIE	RW	0	FIFO 满中断使能(FIFO FULL INTERRUPT ENABLE) 0: FIFO 满中断禁止 1: FIFO 满中断使能
2	FIFOHIE	RW	0	FIFO 半满中断使能(FIFO HALF INTERRUPT ENABLE) 0: FIFO 半满中断禁止 1: FIFO 半满中断使能
1	FIFOEIE	RW	0	FIFO 空中断使能(FIFO EMPTY INTERRUPT ENABLE) 0: FIFO 空中断禁止 1: FIFO 空中断使能
0	CMDIE	RW	0	指令发送完成中断使能(COMMAND DONE INTERRUPT ENABLE) 0: 指令发送完成中断禁止 1: 指令发送完成中断使能

### 21.3.3.12. XIP SPI Flash 命令寄存器(ESMC\_XSFCR)

Address offset: 0x1C

Reset value: 0x0B

本寄存器不支持 soft\_rst 复位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
XINS[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	XINS[7:0]	RW	0x0B	XIP 模式下指令

ESMC XIP 指令寄存器保存在 XIP 访问期间发送的 Flash 指令代码。该寄存器的功能与 SFCR 寄存器类似，但仅在 XIP 模式下使用，写入该寄存器不会启动指令代码传输。需要使用 AHB 访问内存映射地址后，命令才会自动发出。

### 21.3.3.13. XIP SPI 输出控制寄存器(ESMC\_XSOCR)

Address offset: 0x1D

Reset value: 0x02

本寄存器不支持 soft\_rst 复位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE1	SPI_MODE0
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	DDRDAT	RW	0	设置后，在数据接收/传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。

Bit	Name	R/W	Reset Value	Function
6	DDRADDR	RW	0	设置后，在 Flash 地址字节传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。
5	DDRCMD	RW	0	设置后，在 Flash 命令字节传输期间启用双数据速率操作。设置此位后，数据在两个 SCK 时钟边缘上移入/移出。
4	MULTIADDR	RW	0	设置后，启用快速 I/O 读取，其中地址位分别以双、四或八进制格式发送，如果 MUL_COMM 清除，则表示地址以单 SPI 模式发送。 在非扩展 SPI 传输 (MUL_COMM=1) 中，地址字节以与数据字节相同的格式发送。
3	MULTICMD	RW	0	未设置时，启用扩展 SPI 操作。这意味着 Flash 命令通过 DQ0 线以单 SPI 模式发送，而不管八进制、四进制和双 SPI 位值如何。
2	SEND_M	RW	0	内存映射模式下，在 ADDR(8 位、16 位、24 位或 32 位)之后发送 MREG[7:0]。
1:0	SPI_MODE[1:0]	RW	0x2	设置 SPI 传输格式 2'b00 SINGLE SPI 2'b01 DUAL SPI 2'b10 QUAD SPI 2'b11 OCTAL SPI

XSOCR 寄存器用于控制 ESMC 在 XIP 模式下的操作。该寄存器具有与 SOCR 寄存器相同的功能，但其设置仅在 XIP 模式 Flash 读取操作期间使用。

XDCR[6]用于在 XIP 模式下启用/禁用向 SPI Flash 发送指令字节。在 XIP 访问期间设置位 6 时，ESMC 仅发送地址和编程的 DUMMY 周期，后跟数据字节。

#### 21.3.3.14. XIP Dummy 控制寄存器(ESMC\_XDCR)

Address offset: 0x1E

Reset value: 0x00

本寄存器不支持 soft\_rst 复位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REC	NO_CMD	NO_ADDR			XDUMMY[4:0]		
RW	RW	RW			RW		

Bit	Name	R/W	Reset Value	Function
7	REC	RW	0	接收模式 ( Reception) 需要为 xSPI XIP 操作设置，以实现数据的接收。
6	NOCMD	RES	-	该位用于 SPI NOR Flash 连续读取的手动操作模式(没有地址解码器)。将会省略命令，而以地址开始传输。
5	NOADDR	RW	0	设置 NO_ADDR 位后，ESMC 不发送地址字节，命令字节后面跟着 DUMMY CYCLES(如果是 eny)和 DATA Bytes
4:0	XDUMMY[4:0]	RW	5'b0000	XIP 模式下 DUMMY 周期 定义 XIP 模式下，任何地址传输后生成的 DUMMY 周期数。 11111: 31 个 dummy 周期 11110: 30 个 dummy 周期 ... 00001: 1 个 dummy 周期 00000: 无 dummy 周期

#### 21.3.3.15. XIP 配置寄存器 3(ESNC\_XCR3)

Address offset: 0x1F

Reset value: 0x00

本寄存器不支持 soft\_rst 复位

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Res	SSCLRRQ	Res			32BIT_ADDR	16BIT_ADDR	8BIT_ADDR

Bit	Name	R/W	Reset Value	Function
7	Reserved			Reserved
6	SSCLRRQ	RW	0	从机选择输出清除请求，将逻辑 1 写入该位会导致从机选择输出被驱动为高电平。该位自动清除。
5	Reserved			
4	Reserved			Reserved
3	Reserved			
2	32BITADDR	RW	0	启用 32 位地址
1	16BITADDR	RW	0	启用 16 位地址
0	8BITADDR	RW	0	启用 8 位地址

### 21.3.4. ESMC 寄存器描述(FOUR BYTE 访问)

ESMC 寄存器基址: 0xA000 1000

#### 21.3.4.1. 24 位地址寄存器 (ESMC\_ADDR24)

Address offset:0x08

Reset value:0x0000000B

该寄存器只能按字 (32 位) 访问

ESMC 包含 4 个 8 位地址寄存器 ADDR3、ADDR2、ADDR1、ADDR0，允许寻址支持 24 位和 32 位地址字段的 Flash。默认情况下，ESMC 配置为使用 24 位地址运行。

地址寄存器可在 2 个不同的地址集下访问。在 32 位系统中，地址寄存器 2-0 可在单个 32 位寄存器内访问，并与 Flash 命令一起加载以发送。这通过预定义的解码器命令简化了 Flash 数据读取操作。新命令可以通过单次写入启动。

32 位宽的 ADDR 寄存器保存在访问外部 SPI Flash 期间从 ESMC 发送的地址。默认情况下，只发送地址寄存器的 24 个 LSB 位。设置 32 位 EN 位 (CR3 (2) ) 后，32 位地址将从 ESMC 传输出去。地址寄存器值以当前选择的 SPI 模式扩展 SPI、单 SPI、双 SPI、四 SPI 或八 SPI 通道发送。地址寄存器应在新 Flash 命令加载到 SFCR 寄存器之前或在相同的写入顺序期间加载。在 XIP 模式下，地址寄存器的值受从地址总线读取的地址的影响。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR2[7:0]								ADDR1[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR0[7:0]								INS[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
31:24	ADDR2[7:0]	RW	0x00	24 位地址
23:16	ADDR1[7:0]	RW	0x00	CR(14) = 0 (默认 24 位地址模式) 时，使用 24 位地址
15:8	ADDR0[7:0]	RW	0x00	地址寄存器值以当前选择的 SPI 模式扩展 SPI、单 SPI、双 SPI、四 SPI 或八通道发送。 地址寄存器应在新 Flash 命令加载到 SFCR 寄存器之前或在相同的写入顺序期间加载。
7:0	INS1[7:0]	RW	0x0B	SPI 指令  地址寄存器 2-0 可在单个 32 位寄存器内访问，并与 Flash 命令一起加载以发送。新命令可以通过单次写入启动。  默认情况下，命令在每个序列的开头发送到 Flash。默认情况下，它以扩展 SPI 格式发送，但是对于选定的 SPI 操作模式，它也可以分别以双模式、四模式或 QCTAL 模式发

Bit	Name	R/W	Reset Value	Function
				送。无论 DDR 控制位值如何，该命令永远不会在 DDR 模式下发送。ESMC 可以生成并发送任何选定格式的 FLASH 命令。

#### 21.3.4.2. 32 位地址寄存器 (ESMC\_ADDR32)

Address offset:0x0C

Reset value:0x0101 0000

该寄存器只能按字 (32 位) 访问

SS0~1 用于配置 SPI 主传输期间应驱动哪个从选择输出。寄存器的内容自动分配给 SS1O-SS0O。该寄存器允许使用多达 2 个不同的 SPI Flash 轻松进行 SPI 操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RW				SS3 X	SS2 X	SS1 X	SS0 X	RW				SS3 X	SS2 X	SS1 X	SS0 X
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
MREG[7:0]								ADDR3[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
31:28	Reserved	RW	0x00	Reserved
27:24	XSSxO[3:0]	RW	0x01	在 XIP 模式下选择主机传输时的 SSxO 引脚，SSxO 引脚也是从机的 CS(chip sel)输入
23:20	Reserved	RW	0x00	Reserved
19:16	SSxO[3:0]	RW	0x01	间接模式选择主机传输时的 SSxO 引脚，SSxO 引脚也是从机的 CS(chip sel)输入
15:8	MREG	RW	0x00	只有在 CR3 寄存器中设置了 SEND_M 位时，该寄存器的内容才会在地址字节之后发送。
7:0	ADDR3[7:0]	RW	0x00	32 位地址 CR3(2) = 1 (32 位地址使能) 时，使用 32 位地址 地址寄存器值以当前选择的 SPI 模式扩展 SPI、单 SPI、双 SPI、四 SPI 或八通道发送。 地址寄存器应在新 Flash 命令加载到 SFCR 寄存器之前或在相同的写入顺序期间加载。

#### 21.3.4.3. 数据寄存器 (ESMC\_DATA)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3 [7:0]								DATA2 [7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1 [7:0]								DATA0 [7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	DATA3 [7:0]	RW	0x00	向外部 SPI 设备发送/接收的数据
23:16	DATA2[7:0]	RW	0x00	Flash 写入操作期间用于数据传输的数据缓冲区，读取命令期间用于数据字节存储的数据缓冲区。这些地址可用于两个方向。当数据缓冲存储器启用时，这些寄存器在读写方向上都指向 FIFO 顶部。
15:8	DATA1[7:0]	RW	0x00	
7:0	DATA0[7:0]	RW	0x00	

#### 21.3.5. ESMC 寄存器映像

ESMC 基地址: 0xA000 1000

通信模式设置为 HyperBus™ 时, 0x04, 0x08 地址寄存器功能切换为 CA0, CA1, CA2 功能。

**FOUR BYTE 访问寄存器**

Offset	Register																
0xA0001000	ESMC_CR	BAUD[7:0]															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		RW	SSSETRQ	SSCLRQQ	FIFOCLR	Reserved	DUMMY[4:0]	RW	NOADDR	0	0	0	0	0	0	0	0
0x04	ESMC_CTL	RW															
		RW	RW	RW	RW	REC	RW	NOCMD	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	ESMC_ADDR24	RW															
		RW	SSS3	SSS2	SSS1	SSS0	RW	SS3	SS2	SS1	SS0	0	0	0	0	0	0
		0	0	0	0	0	RW	0	0	0	0	0	0	0	0	0	0
0x0C	ESMC_ADDR32	RW															
		RW	DDRRDATA	DDRADDR	DDRCMD	0	RW	MULTIADDR	0	RW	MULTICMD	0	RW	XSP[1:0]	INS[7:0]		
		0	0	0	0	0	RW	0	0	0	0	0	0	0	0	0	0
0x10	ESMC_DATA	RW															
		RW	DATA3 [7:0]	DATA2 [7:0]	DATA1 [7:0]	DATA0 [7:0]	RW	MREG[7:0]	RW	RW	RW	RW	RW	RW	RW	RW	
		0	0	0	0	0	RW	0	0	0	0	0	0	0	0	0	0
0x14	ESMC_SR	RW															
		RW	IDLEIE	FIFOIF	IDLEIF	FIFOHIF	FIFOEIE	CMDIE	DATAWAITIE	0	0	0	0	0	0	0	0
		0	0	0	0	0	RW	0	0	0	0	0	0	0	0	0	0
0x1C	ESMC_XIP	RW															
		RW	DDDRDATA	DDRADDR	DDRCMD	0	RW	MULTIADDR	0	RW	MULTICMD	0	RW	SENDM	0	SPI-MODE[1:0]	XINS[7:0]
		0	0	0	0	0	RW	0	0	0	0	0	0	0	0	0	0

**ONE BYTE 访存寄存器**

偏移	寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
0x00	CR	SPIEN	XIP EN		GIE	DMA EN	2xQUAD	-	SOFT RST	0xF8
	R/W	RW	RW		RW	RW	RW		RW	

偏移	寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Re-set		
	Reset Value	1	1		1	1	0		0			
0x01	<b>CR2</b>	-	-					CPOL	CPHA	0x00		
	R/W							RW	RW			
	Reset Value							0	0			
0x02	<b>TCR</b>	TCR[7:0]								0x00		
	R/W	RW										
	Reset Value	0										
0x03	<b>BAUD</b>	BAUD[7:0]								0x04		
	R/W	RW										
	Reset Value	0x04										
0x04	<b>SFCR0 COMMAND ONLY</b>	INS[7:0]								0x0B		
	R/W	RW										
	Reset Value	0x0B										
0x05	<b>SOCR</b>	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02		
	R/W	RW	RW	RW	RW	RW	RW	RW				
	Reset Value	0	0	0	0	0	0	0x02				
0x06	<b>DCR</b>	REC	NO_CMD	NO ADDR	DUMMY[4:0]					0x00		
	R/W	RW	RW	RW	RW							
	Reset Value	0	0	0	0							
0x07	<b>CR3</b>	SS SET	SS CLEAR	FIFO CLR			32BIT_ADDR	16BIT_ADDR	8BIT_ADDR	0x00		
	R/W	RW	RW	RW			RW	RW	RW			
	Reset Value	0	0	0			0	0	0			
0x14	<b>SR</b>	SPIF	FIFO OV		IDLE STATE		TX in progress	RX in progress	SS_ACTIVE	0X10		
	R/W	RW	RW		RW		RW	RW	RW			
	Reset Value	0	0		1		0	0	0			
0x15	<b>IFR</b>	ADDR DONE	FIFO OV	DATA WAIT	IDLE	FIFO Full	FIFO HALF Full	FIFO Empty	COMMAND DONE	0x02		
	R/W	R	R	R	R	R	R	R	R			
	Reset Value	0	0	0	0	0	0	1	0			
0x16	<b>IER</b>	ADDR DONE IE	FIFO OV IE	DATA WAIT IE	IDLE IE	FIFO Full IE	FIFO HALF Full IE	FIFO Empty IE	COMMAND DONE IE	0x00		
	R/W	RW	RW	RW	RW	RW	RW	RW	RW			
	Reset Value	0	0	0	0	0	0	0	0			
0x1C	<b>XSCR</b>	XINS[7:0]								0x0B		
	R/W	RW										
	Reset Value	0x0B										
0x1D	<b>XSOCR</b>	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02		
	R/W	RW	RW	RW	RW	RW	RW	RW				
	Reset Value	0	0	0	0	0	0	0x02				
0x1E	<b>XDCR</b>	REC	NO_CMD	NO_ADDR	DUMMY[4:0]					0x00		
	R/W	RW	RW	RW	RW							
	Reset Value	0	0	0	0							
0x1F	<b>XCR3</b>	-	SSCLRRQ	-	-	-	32BIT_ADDR	16BIT_ADDR	8BIT_ADDR	0x00		
	R/W		RW				RW	RW	RW			
	Reset Value		0				0	0	0			

## 22. SDIO 接口 (SDIO)

### 22.1. 简介

#### 22.1.1. 主要特征

SD/SDIO MMC 卡主机模块(SDIO)在 AHB 外设总线和多媒体卡(MMC)、 SD 存储卡、 SDIO 卡和 CE-ATA 设备间提供了操作接口。

多媒体卡系统规格书由 MMCA 技术委员会发布，可以在多媒体卡协会的网站上获得。

CE-ATA 系统规格书可以在 CE-ATA 工作组的网站上获得。

支持以下功能：

- 支持 SD 卡 2.0 版本
- 支持 SD I/O 卡 2.0 版本
- 支持 MMC4.2 版本
- 支持 CE-ATA 1.1 版本
- 支持命令完成信号和向主机处理器中断
- 命令完成信号关闭功能

注：

- 1) SDIO 不支持 SPI 模式的通信模式
- 2) 只支持 I/O 模式的 SD 卡或复合卡中的 I/O 部分不能支持 SD 存储设备中很多需要的命令，这里有些命令在 SDI/O 设备中不起作用，如擦除命令，因此 SDIO 不支持这些命令。另外， SD 存储卡和 SD I/O 卡中有些命令是不同的， SDIO 也不支持这些命令。
- 3) MMC4.1 不支持 DDR 启动

#### 22.1.2. 模块框图

### 22.2. 功能说明

SDIO 包含 2 个部分：

- SDIO 适配器模块：实现所有 MMC/SD/SD I/O 卡的相关功能，如时钟的产生、命令和数据的传送。
- AHB 总线接口：操作 SDIO 适配器模块中的寄存器，并产生中断和 DMA 请求信号。

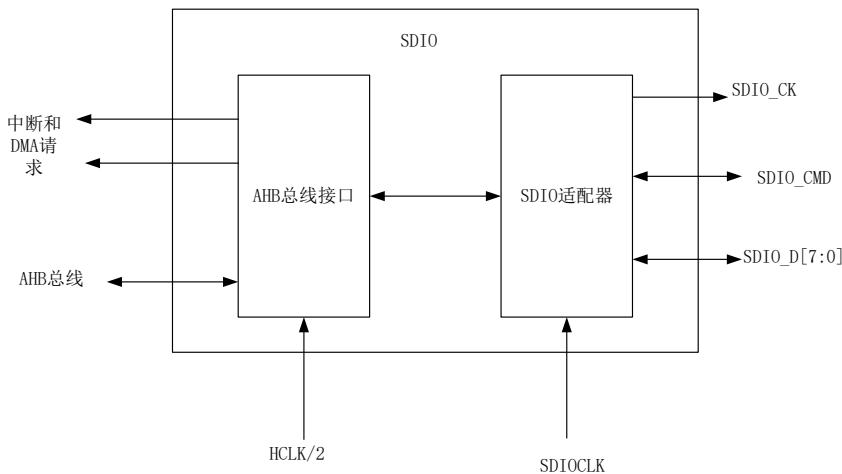


图 22-1 SDIO 框图

复位后默认情况下 SDIO\_D0 用于数据传输。初始化后主机可以改变数据总线的宽度。如果一个多媒体卡接到了总线上，则 SDIO\_D0、SDIO\_D[3:0]或 SDIO\_D[7:0]可以用于数据传输。MMC 版本 V3.31 和之前版本的协议只支持 1 位数据线，所以只能用 SDIO\_D0。

如果一个 SD 或 SD I/O 卡接到总线，可通过主机配置数据传输使用 SDIO\_D0 或 SDIO\_D[3:0]。所有的数据线都工作在推挽模式。

SDIO\_CMD 有两种操作模式：

- 用于初始化时的开路模式(仅用于 MMC 版本 V3.31 或之前版本)
- 用于命令传输的推挽模式(SD/SD I/O 卡和 MMC V4.2 在初始化时也使用推挽驱动)。

SDIO 使用两个时钟信号：

- SDIO 适配器时钟(SDIOCLK=HCLK)
- AHB 总线时钟(HCLK/2)

下表适用于多媒体卡/SD/SD I/O 卡总线：

表 22-1 SDIO 引脚定义

引脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO 卡时钟。这是从主机至卡的时钟线
SDIO_CMD	双向	多媒体卡/SD/SDIO 卡命令。这是双向的命令/响应信号线
SDIO_D[7:0]	双向	多媒体卡/SD/SDIO 卡数据。这些是双向的数据总线。

## 22.2.1. SDIO 适配器

SDIO 适配器包括控制单元、命令单元和数据单元，并且可以向卡生成信号。这些信号的具体描述如下：

SDIO\_CLK：SDIO 控制器提供给卡的时钟。每个时钟周期在命令线(SDIO\_CMD)和所有的数据线(SDIO\_DAT)上直接发送一位命令或数据。对于 MMC 卡 V3.31 版本，SDIO\_CLK 频率可以在 0 MHz 到 20 MHz 之间，对于 MMC 卡 V4.2 版本可以在 0 MHz 到 48MHz 之间，对于 SD 或 SD I/O 卡可以在 0 MHz 到 25 MHz。

SDIO\_CMD：该信号是双向命令通道，用于卡的初始化和命令的传输。命令从 SDIO 控制器发送到卡，响应从卡发送到主机。CMD 信号有两种操作模式：用于初始化的开漏模式（仅用于 MMC 卡 V3.31 及之前版本）和用于命令传送的推挽模式（SD 卡/SD I/O 卡和 MMC 卡 4.2 版本初始化时也是用推挽模式）。

SDIO\_DAT[7:0]：这些信号线都是双向数据通道。数据信号线操作在推挽模式。每次只有卡或者主机会驱动这些信号。默认情况下，上电或者复位后仅 DAT0 用于数据传输。SDIO 适配器可以配置更宽的数据总线用于数据传输，使用

DAT0-DAT3 或者 DAT0-DAT7(仅适用于 MMCV4.2)。SDIO 对数据信号线 DAT1-DAT7 有内部上拉。在进入 4 位模式后，卡断开 DAT1 和 DAT2 的内部上拉 (DAT3 内部上拉保持不变是由于 SPI 模式下 CS 片选的使用)。相应地，在进入 8 位模式后，断开 DAT1, DAT2 和 DAT4-DAT7 的内部上拉。

SDIO 适配器是 SD/SD I/O /MMC/CE-ATA 的接口，它由 3 个子单元组成：

### 22.2.1.1. 控制单元

控制单元包含电源管理功能和时钟管理功能用于存储卡时钟。电源管理是由 SDIO\_PWRCTL 寄存器控制的，实现电源的掉电和上电。通过设置 SDIO\_CLKCR 的 PWRSAV 位来配置省电模式，实现当总线空闲时，关闭 SDIO\_CLK。时钟管理向卡生成 SDIO\_CLK 时钟信号。

### 22.2.1.2. 命令单元

命令单元实现向卡发送和接收命令。数据传输流由命令状态机 (CSM) 控制。对 SDIO\_CMD 寄存器进行一次写操作并设置该寄存器的 start\_cmd 位为 1 后，命令传输开始。首先向卡发送一个命令，这个命令包含 48 位，通过 SDIO\_CMD 线发出，每个 SDIO\_CLK 发送一个比特数据。这 48 位命令包含 1 位起始位、1 位传输位、6 位命令索引 (由 SDIO\_CMD 寄存器的 CMDINDEX 位定义)、32 位参数 (由 SDIO\_CMDARG 定义)、7 位 CRC 和 1 位停止位。然后接收来自卡的响应 (在 SDIO\_CMD 寄存器的 CMDINDEX 位不为 0b00 或 0b10 的情况下)，响应分为 48 位的短响应和 136 位的长响应，响应都存在 SDIO\_RESP0 - SDIO\_RESP3 寄存器中。

### 命令通道状态机(CPSM)

当写入命令寄存器并设置了使能位，开始发送命令。命令发送完成时，命令通道状态机(CPSM)设置状态标志并在不需要响应时进入空闲状态(见下图)。当收到响应后，接收到的 CRC 码将会与内部产生的 CRC 码比较，然后设置相应的状态标志。

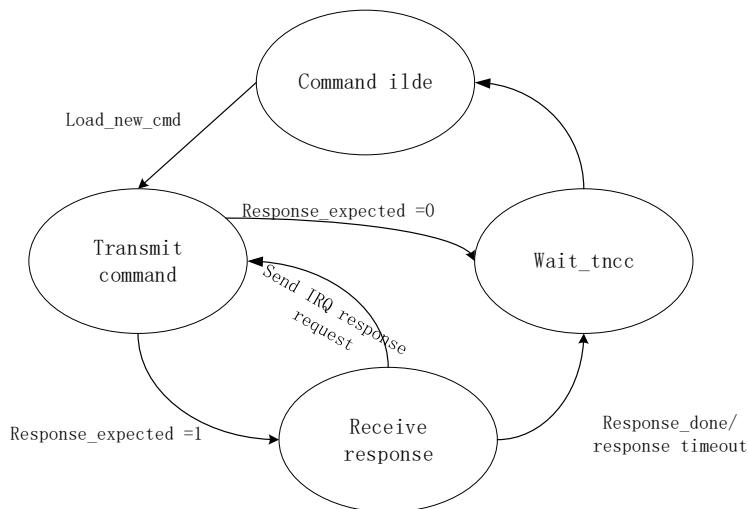


图 22-2 SD\_MMC 命令通道状态机 (CPSM)

命令路径状态机根据命令寄存器位值执行以下功能：

1. send\_initialization – 在发送命令之前发送 80 个时钟的初始化序列。
2. response\_expected – 对命令的预期响应。命令发出后，命令路径状态机接收 48 位或 136 位响应，发送给 SDIO 适配器。如果在超时寄存器中编程的时钟数内未收到卡响应的起始位，则在中断状态寄存器中设置响应超时和命令完成位。如果未设置响应预期位，则命令发出后，在中断状态寄存器中设置命令完成位。
3. response\_length – 如果设置了该位，则接收到 136 位的响应；如果未设置，则接收 48 位响应。
4. check\_response\_crc – 如果设置了该位，命令路径将在响应中接收到的 CRC7 与内部生成的 CRC7 进行比较。如果两者不匹配，则响应 CRC 错误信号发送给 BIU；也就是说，在原始中断状态寄存器中设置响应 CRC 错误位。

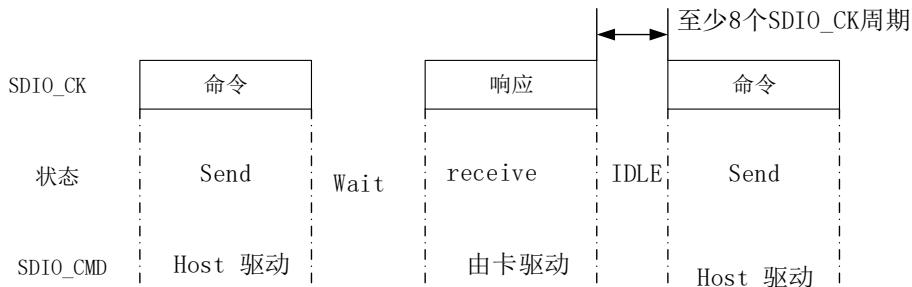


图 22-3 SDIO 命令传输

### • 命令格式

**命令：**命令是用于开始一项操作。主机向一个指定的卡或所有的卡发出带地址的命令或广播命令(广播命令只适合于 MMC V3.31 或之前的版本)。命令在 CMD 线上串行传送。所有命令的长度固定为 48 位，下表给出了多媒体卡、 SD 存储卡和 SDIO 卡上一般的命令格式。

CE-ATA 命令是 MMC V4.2 命令的扩充，所以具有相同的格式。

命令通道为半双工模式，这样命令和响应可以分别发送和接收。如果 CPSM 不处在发送状态， SDIO\_CMD 输出处于高阻状态。 SDIO\_CMD 上的数据与 SDIO\_CK 的上升沿同步。

表 22-2 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:0]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

**响应：**响应是由一个被指定地址的卡发送到主机，对于 MMC V3.31 或以前版本所有的卡

同时发送响应；响应是对先前接收到命令的一个应答。响应在 CMD 线上串行传送。

SDIO 支持 2 种响应类型， 2 种类型都有 CRC 错误检测：

- 48 位短响应
- 136 位长响应

如果响应不包含 CRC(如 CMD1 的响应)，设备驱动应该忽略 CRC 失败状态。

表 22-3 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:0]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

表 22-4 长响应格式

位	宽度	数值	说明
135	1	0	开始位

位	宽度	数值	说明
134	1	0	传输位
[133:128]	6	111111	保留
[127: 1]	127	-	CID 或 CSD(包含内部 CRC7)
0	1	1	结束位

命令寄存器包含命令索引(发至卡的 6 位)和命令类型；命令本身决定了是否需要响应和响应的类型、48 位还是 136 位。命令通道中的状态标志示于下表：

表 22-5 命令通道状态标志

标志	说明
CMDREND	响应的 CRC 正确
CCRCFAIL	响应的 CRC 错误
CMDSENT	命令(不需要响应的命令)已经送出
CTIMEOUT	响应超时
CMDACT	正在发送命令

CRC 发生器计算 CRC 码之前所有位的 CRC 校验和，包括开始位、发送位、命令索引和命令参数(或卡状态)。对于长响应格式，CRC 校验和计算的是 CID 或 CSD 的前 120 位；注意，长响应格式中的开始位、传输位和 6 个保留位不参与 CRC 计算。

CRC 校验和是一个 7 位的数值：

### 22.2.1.3. 数据单元

数据单元实现主机与卡之间的数据传输。当数据宽度为 8 位时，数据传输使用 SDIO\_DAT[7:0]信号线；当数据宽度为 4 位时，数据传输使用 SDIO\_DAT[3:0]信号线；当数据宽度为 1 位时，数据传输使用 SDIO\_DAT[0]信号线。数据传输流由数据状态机(DSM)控制。

#### • 数据发送状态机(DTSM)

DTSM 工作在 SDIO\_CK 频率，卡总线信号与 SDIO\_CK 的上升沿同步。

如图所示，数据发送状态机在收到数据写入命令的响应后两个时钟开始数据发送；即使命令路径状态机检测到错误或响应 CRC 错误，也会发生这种情况。如果由于响应超时而未从卡接收到响应，则不传输数据。根据命令寄存器中 transfer\_mode 位的值，数据发送状态机将数据以流或块的形式放在卡数据总线上。

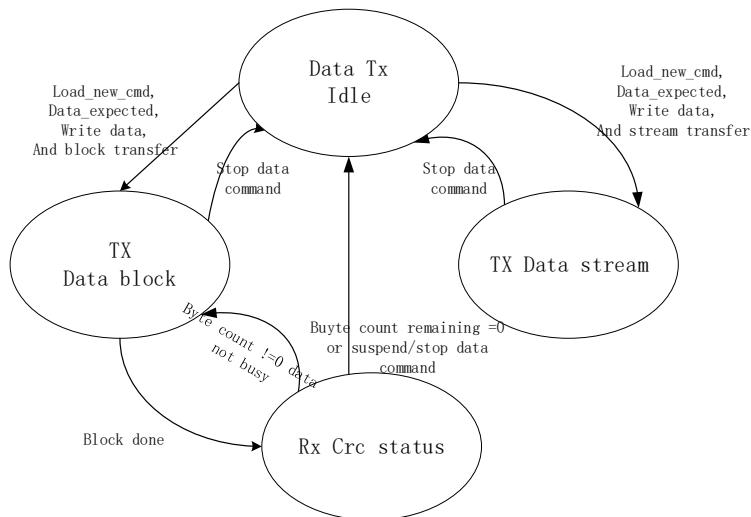


图 22-4 数据发送状态机 (DTSM)

#### • 数据接收状态机(DRSM)

如图所示，数据接收状态机在数据读取命令的结束位后两个时钟周期接收数据，即使命令路径检测到响应错误或响应 CRC 错误也是如此。如果因为响应超时而没有收到卡的响应，则 SDIO 适配器不会收到数据传输完成的信号；如果发送的命令是对卡的非法操作，则会发生这种情况，这会阻止卡开始读取数据传输。如果在数据超时之前未收到数据，则数据接收状态机向 SDIO 适配器发出数据超时信号并结束数据传输。根据命令寄存器中 transfer\_mode 位的值，数据接收状态机以流或块的形式从卡数据总线获取数据。

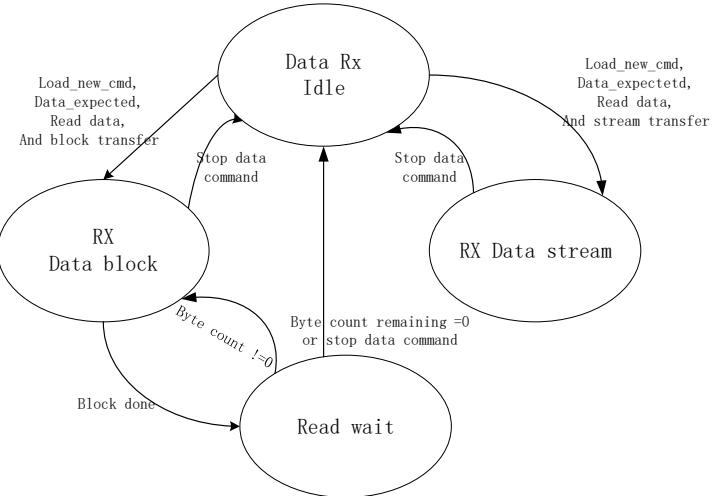


图 22-5 数据接收状态机 (DRSM)

#### 22.2.1.4. SDIO 时钟控制

有三个寄存器位可以用来进行时钟处理，即 clk 的 CKSEL, SMPEN, SMPCLKSEL 位：

1. CKSEL：用来控制输出命令，数据选择
2. SMPEN: 预采样使能， SMPCLKSEL: 时钟相位选择，可以选择时钟的具体相位

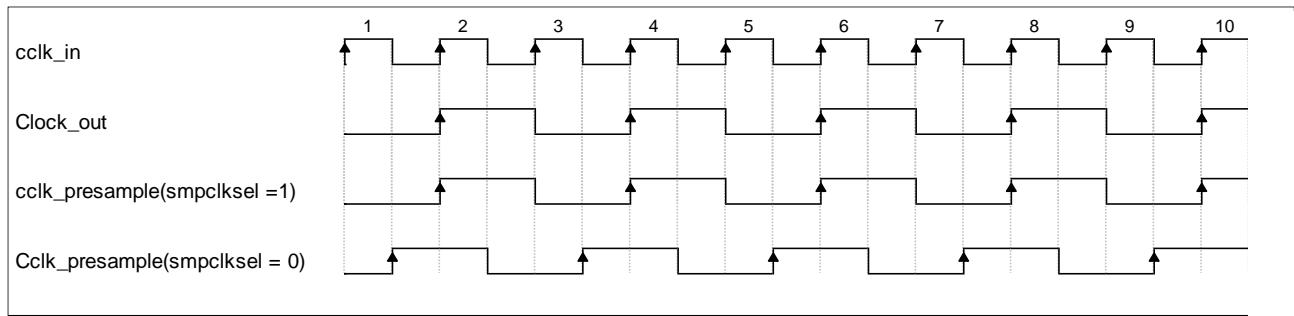


图 22-6 cclk\_presample 的示意图

#### 22.2.2. SDIO AHB 接口

AHB 接口产生中断和 DMA 请求，并访问 SDIO 接口寄存器和数据 FIFO。它包含一个数据通道、寄存器译码器和中断/DMA 控制逻辑。

##### 22.2.2.1. SDIO 中断

当至少有一个选中的状态标志为高时，中断控制逻辑产生中断请求。有一个屏蔽寄存器用于选择可以产生中断的条件，如果设置了相应的屏蔽标志，则对应的状态标志可以产生中断。

##### 22.2.2.2. SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程

SDIO/DMA：在 SDIO 和存储器之间数据传输的过程。

在下面的例子中，主机控制器使用 CMD24(WRITE\_BLOCK)从主机传送 512 字节到 MMC 卡，DMA 控制器用于从存储器向 SDIO 的 FIFO 填充数据。

1. 执行卡识别过程
2. 提高 SDIO\_CK 频率
3. 发送 CMD7 命令选择卡
4. 按下述步骤配置 DMA2:
  - a) 使能 DMA2 控制器并清除所有的中断标志位
  - b) 设置 DMA2 通道 4 的源地址寄存器为存储器缓冲区的基地址， DMA2 通道 4 的目标地址寄存器为 SDIO\_FIFO 寄存器的地址
  - c) 设置 DMA2 通道 4 控制寄存器(存储器递增， 非外设递增， 外设和源的数据宽度为字宽度)
  - d) 使能 DMA2 通道 4
5. 发送 CMD24(WRITE\_BLOCK)，操作如下：
  - a) 设置 SDIO 数据长度寄存器(SDIO 数据时钟寄存器应该在执行卡识别过程之前设置好)
  - b) 设置 SDIO 参数寄存器为卡中需要传送数据的地址
  - c) 设置 SDIO 命令寄存器： CmdIndex 置为 24(WRITE\_BLOCK)； WaitRest 置为 1(SDIO 卡主机等待响应)； start\_cmd 置为 1(使能 SDIO 卡主机发送命令)，保持其它域为他们的复位值。
  - d) 等待 CD 中断，然后设置 SDIO 数据寄存器： DTEN 置为 1(使能 SDIO 卡主机发送数据)； DTDIIR 置为 0(控制器至卡方向)； DTMODE 置为 0(块数据传送)； DMAEN 置为 1(使能 DMA)； DBLOCKSIZE 置为 9(512 字节)； 其它域不用设置。
  - e) 等待 SDIO\_STA[10]=DBCKEND。
6. 查询 DMA 通道的使能状态寄存器，确认没有通道仍处于使能状态。

### 22.2.3. 卡功能描述

#### 22.2.3.1. 卡寄存器

卡内部定义了接口寄存器： OCR， CID， CSD， EXT\_CSD， RCA， DSR 和 SCR。这些寄存器只能通过相应的命令来访问。OCR， CID， CSD 和 SCR 寄存器包含卡的一些特定信息，而 RCA 和 DSR 寄存器是配置寄存器，存储实际的配置参数。ECSD 寄存器同时包含卡的特定信息和实际的结构参数。有关具体信息，请参考相关的规范。

OCR 寄存器： 32 位操作条件寄存器 (OCR) 储存卡的 VDD 电压描述和存取模式指示 (MMC)。另外，该寄存器包括一个状态信息位。如果卡上电过程已经完成该状态位被置位。该寄存器在 MMC 和 SD 卡之间有一点不同。主机可以使用 CMD1 (MMC) , ACMD41 (SD 存储卡) , CMD5 (SD I/O) 来获取该寄存器的内容。

CID 寄存器： 卡识别寄存器 (CID) 是 128 位宽。它包含在卡识别阶段使用的卡识别信息。每个读/写 (RW) 卡应具有唯一的标识号。主机可以使用 CMD2 和 CMD10 得到这个寄存器的内容。

CSD 寄存器： 卡特定数据寄存器提供访问卡中的内容信息。CSD 定义了数据格式、错误校正类型、最大数据访问时间、数据传输速度、 DSR 寄存器是否可以使用等。寄存器的可编程部分可通过 CMD27 来修改。主机可以使用 CMD9 得到这个寄存器的内容。

RCA 寄存器： 可写的 16 位相对卡地址寄存器存放卡地址，该地址在卡的初始化期间由卡向外发布。这个地址用于卡识别过程之后，所寻址的主机和卡通信。主机可以使用 CMD3 要求卡发布一个新的相对地址 (RCA)。

注意：RCA 的寄存器的缺省值是 0x0001 (MMC) 或 0x0000 (SD/SD I/O)。这个数值是保留值，用于通过 CMD7 设置所有卡到待机 (Stand-by) 状态。

**DSR 寄存器 (可选)**：16 位驱动阶段寄存器是可选的，可用于在扩展操作条件中提高总线性能（取决于类似于总线长度，传输速率和卡数目这些参数）。CSD 寄存器中有 DSR 寄存器使用情况的信息。DSR 寄存器的默认值是 0x404。主机可以使用 CMD4 得到这个寄存器的内容。

**SCR 寄存器**：仅 SD/ SD I/O (如果有存储模块) 有这个寄存器。除了 CSD 寄存器，除了 CSD 寄存器，还有另一种配置寄存器名为 SD 卡配置寄存器 (SCR)，它仅用于 SD 卡。SCR 提供了被配置到特定 SD 存储卡的特殊功能的信息。SCR 寄存器的大小是 64 位。该寄存器应在出厂前通过 SD 存储卡制造商进行设置。主机可以使用 ACMD51 得到这个寄存器的内容。

响应格式 R1 包含了一个 32 位的卡状态域，这个域是用于向卡主机发送卡的状态信息(这些信息有可能存在本地的状态寄存器中)。除非特别说明，卡返回的状态始终是与之前的命令相关的。

表 4-2 定义了不同的状态信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E: 错误位
- S: 状态位
- R: 检测位，并依据实际的命令响应而设置
- X: 检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A: 依据卡的当前状态
- B: 始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C: 读即可清除

表 22-6 卡状态

位	名称	类型	数值	说明	清除条件
31	AD-DRESS_OUT_OF_RANGE	E R X	'0'= 无错误 '1'= 错误	一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写超出卡的容量的部分。	C
30	ADDRESS_MISALIGN		'0'= 无错误 '1'= 错误	命令中的地址参数(与当前的数据块长度对照)定义的第一个数据块未与卡的物理块对齐。 一个多数据块或数据流读/写操作(即使从一个合法的地址开始)试图读或写未与物理块对齐的数据块。	C
29	BLOCK_LEN_ERROR		'0'= 无错误 '1'= 错误	SET_BLOCKLEN 命令的参数超出了卡的最大允许范围，或先前定义的数据块长度对于当前命令来说是非法的(例如：主机发出一个写命令，当前的块长度小于卡所允许的最小长度，同时又不允许写入部分数据块)。	C
28	ERASE_SEQ_ERROR		'0'= 无错误 '1'= 错误	发送擦除命令的顺序错误。	C
27	ERASE_PARAM	EX	'0'= 无错误 '1'= 错误	擦除时选择了非法的擦除组。	C
26	WP_VIOLATION	E X	'0'= 无错误 '1'= 错误	试图对一个写保护的数据块编程。	C
25	CARD_IS_LOCKED	S R	'0'= 卡未锁	当设置了该位，表示卡已经被锁住。	A

位	名称	类型	数值	说明	清除条件
			'1'= 卡已锁		
24	LOCK_UNLOCK_FAILED	E X	'0'= 无错误 '1'= 错误	在上锁/解锁中有命令的顺序错误或检测到密码错误	C
23	COM_CRC_ERROR	E R	'0'= 无错误 '1'= 错误	之前的命令中 CRC 校验错误。	B
22	ILLEGAL_COMMAND	E R	'0'= 无错误 '1'= 错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	E X	'0'= 成功 '1'= 失败	卡的内部实施了 ECC 校验，但在更正数据时失败。	C
20	CC_ERROR	E R	'0'= 无错误 '1'= 错误	(标准中未定义)卡内部发生错误，与主机的命令无关。	C
19	ERROR	E X	'0'= 无错误 '1'= 错误	产生了与执行上一个主机命令相关的(标准中未定义)卡内部的错误	C
18:17	保留				
16	CID/CSD_OVERWRITE	E X	'0'= 无错误 '1'= 错误	可以是任何一个下述的错误： 已经写入了 CID 寄存器，不能覆盖 CSD 的只读部分与卡的内容不匹配 试图进行拷贝或永久写保护的反向操作，即恢复原状或解除写保护。	C
15	WP_ERASE_SKIP	E X	'0'= 未保护 '1'= 已保护	遇到已经存在的写保护数据块，仅有部分地址空间被擦除	C
14	CARD_ECC_DISABLED	S X	'0'= 不允许 '1'= 允许	执行命令时没有使用内部的 ECC。	A
13	ERASE_RESET		'0'= 未保护 '1'= 已保护	因为收到一个擦除顺序之外的命令(非 CMD35、CMD36、CMD38 或 CMD13 命令)，进入擦除过程的序列被中止。	C
12:9	CURRENT_STATE	S R	0 = 空闲 1 = 就绪 2 = 识别 3 = 待机 4 = 发送 5 = 数据 6 = 接收 7 = 编程 8 = 断开 9 = 忙测试 10~15 = 保留	当收到命令时卡内状态机的状态。如果命令的执行导致状态的变化，这个变化将会在下个命令的响应中反映出来。这四个位按十进制数 0 至 15 解释。	B
8	READY_FOR_DATA	S R	'0'= 未就绪 '1'= 就绪	与总线上的缓冲器空的信号相对应。	
7	SWITCH_ERROR	E X	'0'= 无错误 '1'= 错误	卡没有按照 SWITCH 命令的要求转换到希望的模式。	B
6	保留				
5	APP_CMD	S R	'0'= 不允许 '1'= 允许	卡期望 ACMD，或指示命令已经被解释为 ACMD 命令	C
4	保留给 SD I/O 卡				

位	名称	类型	数值	说明	清除条件
3	AKE_SEQ_ERROR	E R	'0'= 无错误 '1'= 错误	验证的顺序有错误	C
2	保留给与应用相关的命令。				
1:0	保留给生产厂家的测试模式				

### 22.2.3.2. SD 状态寄存器

SD 状态包含与 SD 存储器卡特定功能相关状态位和一些与未来应用相关状态位，SD 状态的长度是一个 512 位的数据块。收到 ACMD13 命令(CMD55，然后是 CMD13)后，这个寄存器的内容被传送到 SDIO 卡主机。只有卡处于传输状态时(卡已被选择)才能发送 ACMD13 命令。

表 4-3 定义了不同的 SD 状态寄存器信息。表中有关类型和清除条件域的缩写定义如下类型：

- E: 错误位
- S: 状态位
- R: 检测位，并依据实际的命令响应而设置
- X: 检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A: 依据卡的当前状态
- B: 始终与之前的命令相关。接收到正确的命令即可清除(具有一个命令的延迟)。
- C: 读即可清除

表 22-7 SD 状态

位	名称	类型	数值	说明	清除条件
511:510	DAT_BUS_WIDTH	S R	'00'= 1(默认) '01'= 保留 '10'= 4 位宽 '11'= 保留	由 SET_BUS_WIDTH 命令定义的当前数据总线宽度。	A
509	SECURED_MODE	S R	'0'= 未处于保密模式 '1'= 处于保密模式	卡处于保密操作模式 '0'= 未处于保密模式 '1'= 处于保密模式	A
508:496	保留				
495:480	SD_CARD_TYPE	S R	00xxh= 在物理规范版本 1.01 A ~2.00 的 SD 存储器卡('x'表示任意值)。已定义的卡有： '0000'= 通用 SD 读写卡 '0001'= SD ROM 卡	这个域的低 8 位可以在未来定义 SD 存储卡的不同变种(每个位可以用于定义不同的 SD 类型)。高 8 位可以用于定义那些不遵守当前的 SD 物理层规范的 SD 卡	A
479:448	SIZE_OF_PROTECTED_AREA	S R	受保护的区域大小(见以下说明)	(见以下说明)	A
447:440	SPEED_CLASS	S R	卡的速度类型(见以下说明)	(见以下说明)	A
439:432	PERFORMANCE_MOVE	S R	以 1MB/秒为单位的传输性能(见以下说明)	(见以下说明)	A
431:428	AU_SIZE	S R	AU 的大小(见以下说明)	(见以下说明)	A
427:424	保留				
423:408	ERASE_SIZE	S R	一次可以擦除的 AU 数目	(见以下说明)	A
407:402	ERASE_TIMEOUT	S R	擦除 UNIT_OF_ERASE_A 指定的范围的超时数值	(见以下说明)	A

位	名称	类型	数值	说明	清除条件
401:400	ERASE_OFFSET	S R	在擦除时增加的固定偏移数值	(见以下说明)	A
399:312	保留				
311:0	保留给生产厂商				

**SIZE\_OF\_PROTECTED\_AREA**

标准容量卡和高容量卡设置该位的方式不同。对于标准容量卡，受保护区域的容量由下式计算：

$$\text{受保护区域} = \text{SIZE\_OF\_PROTECTED\_AREA} * \text{MULT} * \text{BLOCK\_LEN}$$

SIZE\_OF\_PROTECTED\_AREA 的单位是 MULT \* BLOCK\_LEN。

对于高容量卡，受保护区域的容量由下式计算：

$$\text{受保护区域} = \text{SIZE\_OF\_PROTECTED\_AREA}$$

SIZE\_OF\_PROTECTED\_AREA 的单位是字节。

**SPEED\_CLASS**

这 8 位指示速度的类型和可以通过计算 PW/2 的数值(PW 是写的性能)。

表 22-8 速度类型代码

SPEED_CLASS	数值定义
00h	类型 0
01h	类型 2
02h	类型 4
03h	类型 6
04h-FFH	保留

**PERFORMANCE\_MOVE**

这 8 位以 1MB/秒为单位指示移动性能(Pm)。如果卡不用 RU(纪录单位)移动数据，应该认为 Pm 是无穷大。设置这个域为 FFh 表示无穷大。

表 22-9 速度类型代码

SPEED_CLASS	数值定义
00h	未定义
01h	1MB/S
02h	2MB/S
.....	.....
Feh	254MB/秒
FFh	无穷大

**AU\_SIZE**

这 4 位指示 AU 的长度，数值是 16K 字节为单位 2 的幂次的倍数。

表 22-10 AU\_SIZE 代码

SPEED_CLASS	数值定义
00h	未定义
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB
AH-Fh	保留

依据卡的容量，最大的 AU 长度由下表定义。卡可以在 RU 长度和最大的 AU 长度之间设置任意的 AU 长度。

表 22-11 最大的 AU 长度

容量	16MB-64MB	128MB-256MB	512MB	1GB-32GB
最大的 AU 长度	512KB	1MB	2MB	4MB

#### ERASE\_SIZE

这个 16 位域给出了 NERASE，当 NERASE 个 AU 被擦除时，ERASE\_TIMEOUT 定义了超时时间。主机应该确定适当的一次操作中擦除的 AU 数目，这样主机可以显示擦除操作的进度。如果该域为 0，则不支持擦除的超时计算。

表 22-12 ERASE\_SIZE 代码

ERASE_CLASS	数值定义
0000h	不支持擦除的超时运算
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
.....	.....
FFFFh	65535 个 AU

#### ERASE\_TIMEOUT

这 6 位给出了 TERASE，ERASE\_SIZE 指示的多个 AU 被擦除，这个数值给出从偏移量算起擦除超时。ERASE\_TIMEOUT 的范围可以定义到最多 63 秒，卡的生产商可以根据具体实现选择合适的 ERASE\_SIZE 与 ERASE\_TIMEOUT 的组合，先确定 ERASE\_TIMEOUT 再确定 ERASE\_SIZE。

表 22-13 擦除超时代码

ERASE_TIMEOUT	数值定义
00	不支持擦除的超时运算
01	1 秒
02	2 秒
03	3 秒
.....	.....
63	63 秒

#### ERASE\_OFFSET

这 2 位给出了 TOFFSET，当 ERASE\_SIZE 和 ERASE\_TIMEOUT 同为 0 时这个数值没有意义。

表 22-14 擦除偏移代码

ERASE_OFFSET	数值定义
0	0 秒
1	1 秒
2	2 秒
3	3 秒

### 22.2.3.3. SD 的 I/O 模式

#### SD 的 I/O 中断

为了让 SD I/O 卡能够中断多媒体卡/SD 模块，在 SD 接口上有一个具有中断功能的引脚——第 8 脚，在 4 位 SD 模式下这个脚是 SDIO\_D1，卡用它向多媒体卡/SD 模块提出中断申请。对于每一个卡或卡内的功能，中断功能是可选的。SD I/O 的中断是电平有效，即在被识别并得到多媒体卡/SD 模块的响应之前，中断信号线必须保持有效电平(低)，在中断过程结束后保持无效电平(高)。在多媒体卡/SD 模块服务了中断请求后，通过一个 I/O 写操作，写入适当的位到 SD I/O 卡的内部寄存器，即可清除中断状态位。所有 SD I/O 卡的中断输出是低电平有效，多媒体卡/SD 模块所有数据

线(SDIO/D[3:0])上提供上拉电阻。多媒体卡/SD 模块在中断阶段对第 8 脚(SDIO\_D/IRQ)采样并进行中断检测，其它时间该信号线上的数值将被忽略。

存储器操作和 I/O 操作都具有中断阶段，单个数据块操作的中断阶段定义与多个数据块传输操作的中断阶段定义不同。

## SD 的 I/O 暂停和恢复

在一个多功能的 SD I/O 卡或同时具有 I/O 和存储器功能的卡中，多个设备(I/O 和存储器)共用 MMC/SD 总线。为了使 MMC/SD 模块中的多个设备能够共用总线，SD I/O 卡和复合卡可以有选择地实现暂停/恢复的概念；如果一个卡支持暂停/恢复，MMC/SD 模块能够暂时地停止一个功能或存储器的数据传输操作(暂停)，借此让出总线给具有更高优先级的其它功能或存储器，在这个具有更高优先级的传输完成后，再恢复原先暂停的传输。支持暂停/恢复的操作是可选的。在 MMC/SD 总线上执行暂停/恢复操作有下述步骤：

1. 确定 SDIO\_D[3:0]信号线的当前功能
2. 请求低优先级或慢的操作暂停
3. 等待暂停操作完成，确认设备已暂停
4. 开始高优先级的传输
5. 等待高优先级的传输结束
6. 恢复暂停的操作

## SD I/O 读等待(ReadWait)

可选的读等待(RW)操作只适用于 SD 卡的 1 位或 4 位模式。读等待操作允许 MMC/SD 模块在一个卡正在读多个寄存器(IO\_RW\_EXTENDED, CMD53)时，要求它暂时停止数据传输，同时允许 MMC/SD 模块发送命令到 SD I/O 设备中的其他功能。判断一个卡是否支持读等待协议，MMC/SD 模块应该检测卡的内部寄存器。读等待的时间与中断阶段相关。

## 22.2.4. 命令

### 22.2.4.1. 应用相关命令和通用命令

SD 卡主机模块系统是用于提供一个适用于多种应用类型的标准接口，但同时又要兼顾特定用户和应用的功能，因此标准中定义了两类通用命令：应用相关命令(ACMD)和通用命令(GEN\_CMD)。

当卡收到 APP\_CMD(CMD55)命令时，卡期待下一个命令是应用相关命令。应用相关命令(ACMD)具有普通多媒体卡相同的格式结构，并可以使用相同的 CMD 号码，因为它是出现在 APP\_CMD(CMD55)后面，所以卡把它识别为 ACMD 命令。如果跟随 APP\_CMD(CMD55)之后不是一个已经定义的应用相关命令，则认为它是一个标准命令；例：有一个 SD\_STATUS(ACMD13)应用相关命令，如果在紧随 APP\_CMD(CMD55)后收到 CMD13，它被解释为 SD\_STATUS(ACMD13)；但是如果卡在紧随 APP\_CMD(CMD55)之后收到 CMD7，而这个卡没有定义 ACMD7，则它将被解释为一个标准的 CMD7(SELECT/DESELECT\_CARD)命令。

如果要使用生产厂家自定义的 ACMD，SD 卡主机需要做以下操作：

1. 发送 APP\_CMD(CMD55)命令

卡送回响应给多媒体/SD 卡模块，指示设置了 APP\_CMD 位并等待 ACMD 命令。

2. 发送指定的 ACMD

卡送回响应给多媒体/SD 卡模块，指示设置了 APP\_CMD 位，收到的命令已经正确地按照 ACMD 命令解析；如果发送了一个非 ACMD 命令，卡将按照普通的多媒体卡命令处理同时清除卡中状态寄存器的 APP\_CMD 位。

如果发送了一个非法的命令(不管是 ACMD 还是 CMD)，将被按照标准的非法多媒体卡命令进行错误处理。

GEN\_CMD 命令的总线操作过程，与单数据块读写命令(WRITE\_BLOCK, CMD24 或 READ\_SINGLE\_BLOCK, CMD17)相同；这时命令的参数表示数据传输的方向而不是地址，数据块具有用户自定义的格式和意义。

发送 GEN\_CMD(CMD56)命令之前，卡必须被选中(状态机处于传输状态)，数据块的长度由 SET\_BLOCKLEN(CMD16)定义。 GEN\_CMD(CMD56)命令的响应是 R1b 格式。

#### 22.2.4.2. 命令类型

应用相关命令和通用命令有四种不同的类型：

1. 广播命令(BC)：发送到所有卡，没有响应返回。
2. 带响应的广播命令(BCR)：发送到所有卡，同时收到从所有卡返回的响应。
3. 带寻址(点对点)的命令(AC)：发送到选中的卡，在 SDIO\_D 信号线上不包括数据传输。
4. 带寻址(点对点)的数据传输命令(AC)：发送到选中的卡，在 SDIO\_D 信号线上包含数据传输。

#### 22.2.4.3. 命令格式

### 多媒体卡/SD 卡模块的命令

表 22-15 基于块传输的写命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD23	ac	[31:16] = 0 [15:0] = 数据块数目	R1	SET_BLOCK_COUNT	定义在随后的多块读或写命令中需要传输块的数目
CMD24	adtc	[31:0] = 数据地址	R1	WRITE_BLOCK	按照 SET_BLOCKLEN 命令选择的长度写一个块
CMD25	adtc	[31:0] = 数据地址	R1	WRITE_MULTIPLE_BLOCK	收到一个 STOP_TRANSMISSION 命令或达到了指定的块数目之前，连续地写数据块
CMD26	adtc	[31:0] = 填充位	R1	PROGRAM_CID	对卡的识别寄存器编程。对于每个卡只能发送一次这个命令。卡中有硬件机制防止多次的编程操作。通常该命令保留给生产厂商
CMD27	adtc	[31:0] = 填充位	R1	PROGRAM_CSD	对卡的 CSD 中可编程的位编程。
CMD28	ac	[31:0] = 数据地址	R1b	SET_WRITE_PROT	如果卡有写保护功能，该命令设置指定组的写保护位。写保护特性设置在卡的特殊数据区(WP_GRP_SIZE)。
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PROT	如果卡有写保护功能，该命令清除指定组的写保护位
CMD30	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态
CMD31	保留				

表 22-16 基于块传输的写保护命令

CMD 納引	类型	参数	响应格式	编写	说明
CMD28	ac	[31:0] = 数据块地址	R1b	SET_WRITE_PORT	定义在随后的多块读或写命令中需要传输块的数目
CMD29	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PORT	按照 SET_BLOCKLEN 命令选择的长度写一个块
CMD30	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态
CMD31	保留				

表 22-17 擦除命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD32 ... CMD34		保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这些命令代码。			
CMD35	ac	[31:0] = 数据地址	R1b	CLR_WRITE_PORT	按照 SET_BLOCKLEN 命令选择的长度写一个块
CMD36	adtc	[31:0] = 写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态
CMD37		保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这个命令代码			
CMD38	ac	[31:0]=填充物	R1	ERASE	擦除之前选择的数据块

表 22-18 I/O 模式命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD39	ac	[31:16] = RCA [15] = 寄存器写标志 [14:8] = 寄存器地址 [7:0] = 寄存器数据	R4	FAST_IO	用于写和读 8 位(寄存器)数据域。该命令指定一个卡和寄存器，如果设置了写标志还提供写入的数据。R4 响应包含从指定寄存器读出的数据。该命令访问未在多媒体卡标准中定义的与应用相关的寄存器。
CMD40	bcr	[31:0] = 写保护数据地址	R5	GO_IRO_STATE	置系统于中断模式。
CMD41		保留			

表 22-19 上锁命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD42	adtc	[31:0] = 填充位	R1b	LOCK_UNLOCK	设置/清除密码或对卡上锁/解锁。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD43 ..... CMD54		保留			

表 22-20 用相关命令

CMD 索引	类型	参数	响应格式	编写	说明
CMD55	ac	[31:16] = RCA [15: 0] = 填充位	R1	APP_CMD	指示卡下一个命令是应用相关命令而不是标准命令
CMD56	adtc	[31:1] = 填充位 [0] = RD/WR			在通用或应用相关命令中，或者用于向卡中传输一个数据块，或者用于从卡中读取一个数据块。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD57 ... CMD59		保留			
CMD60 ..... CMD63		保留给生产厂商			

## 22.2.5. 响应

所有的响应是通过 MCCMD 命令在 SDIO\_CMD 信号线上传输。响应的传输总是从对应响应字的位串的最左面开始，响应字的长度与响应的类型相关。

一个响应总是有一个起始位(始终为 0)，跟随着传输的方向位(卡=0)。下表中标示为 x 的数值表示一个可变的部分。除了 R3 响应类型，所有的响应都有 CRC 保护。每一个命令码字都有一个结束位(始终为 1)。

共有 7 种响应类型，它们的格式定义如下：

#### 22.2.5.1. R1(普通响应命令)

代码长度=48 位。位 45:40 指示要响应的命令索引，它的数值介于 0 至 63 之间。卡的状态由 32 位进行编码。

表 22-21 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	x	命令索引
[39:8]	32	x	卡状态
[7:1]	7	x	CRC7
0	1	1	结束位

#### 22.2.5.2. R1b

与 R1 格式相同，但可以选择在数据线上发送一个繁忙信号。收到这些命令后，依据收到命令之前的状态，卡可能变为繁忙。

#### 22.2.5.3. R2(CSD、CSD 寄存器)

代码长度=136 位。CID 寄存器的内容将作为 CMD2 和 CMD10 的响应发出。CSD 寄存器的内容将作为 CMD9 的响应发出。卡只送出 CID 和 CSD 的位[127...1]，在接收端这些寄存器的位 0 被响应的结束位所取代。卡通过拉低 MCDAT 指示它正在进行擦除操作；实际擦除操作的时间可能非常长，主机可以发送 CMD7 命令不选中这个卡。

表 22-22 响应

位	域宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	'111111'	命令索引
[127:1]	127	x	卡状态
0	1	1	结束位

#### 22.2.5.4. R3(OCR 寄存器)

代码长度=48 位。OCR 寄存器的内容将作为 CMD1 的响应发出。电平代码的定义是：限制的电压窗口=低，卡繁忙=低。

表 22-23 R3 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'111111'	保留
[39:8]	32	x	OCR 寄存器

位	域宽度	数值	说明
[7:1]	7	'111111'	保留
0	1	1	结束位

### 22.2.5.5. R4 (快速 I/O)

代码长度=48 位。参数域包含指定卡的 RCA、需要读出或写入寄存器的地址、和它的内容。

表 22-24 R4 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'111111'	保留
[39:8]参数域	[31:16]	16	X
	[15:8]	8	X
	[7:0]	8	X
[7:1]	7	'111111'	CRC7
0	1	1	结束位

### 22.2.5.6. R4b

仅适合 SD I/O 卡：一个 SDIO 卡收到 CMD5 后将返回一个唯一的 SDIO 响应 R4。

表 22-25 R4 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	X	保留
[39:8]参数域	[39]	1	X
	[38:36]	3	X
	[35]	1	X
	[34:32]	3	X
	[31:8]	24	X
[7:1]	7	X	保留
0	1	1	结束位

当一个 SD I/O 卡收到命令 CMD5，卡的 I/O 部分被使能并能够正常地响应所有后续的命令。I/O 卡的使能状态将保持到下一次复位、断电或收到 I/O 复位的 CMD52 命令。注意，一个只包含存储器功能的 SD 卡可以响应 CMD5 命令，它的正确响应可以是：当前存储器=1，I/O 功能数目=0。按照 SD 存储器卡规范版本 1.0 设计的只包含存储器功能的

SD 卡，可以检测到 CMD5 命令为一个非法命令并不响应它。可以处理 I/O 卡的主机将发送 CMD5 命令，如果卡返回响应 R4，则主机会依据 R4 响应中的数据确定卡的配置。

#### 22.2.5.7. R5 (中断请求)

仅适用于多媒体卡。代码长度=48 位。如果这个响应由主机产生，则参数中的 RCA 域为 0x0。

表 22-26 R5 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'111111'	CMD40
[39:8]参数域	[31:16]	16	X 成功的卡或主机的 RCA[31:16]
	[15:0]	16	X 未定义。可以作为中断数据。
[7:1]	7	X	CRC7
0	1	1	结束位

#### 22.2.5.8. R6 (中断请求)

仅适用于 SD I/O 卡。这是一个存储器设备对 CMD3 命令的正常响应。

表 22-27 R6 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'101000'	CMD40
[39:8]参数域	[31:16]	16	X 成功的卡或主机的 RCA[31:16]
	[15:0]	16	X 未定义。可以作为中断数据。
[7:1]	7	X	CRC7
0	1	1	结束位

当发送 CMD3 命令到只有 I/O 功能的卡时，卡的状态位[23:8]会改变；此时，响应中的 16 位将是只有 I/O 功能的 SD 卡中的数值：

- 位 15 = COM\_CRC\_ERROR
- 位 14 = ILLEGAL\_COMMAND
- 位 13 = ERROR
- 位[12:0] = 保留

#### 22.2.5.9. R7

长度为 48 位。由发送 CMD8 响应提供的卡的支持电压信息。第 16-19 位表示该卡支持的电压范围。卡使用 R7 反馈可支持的电压。在响应中，卡将回复参数中设置的电压范围和检查模式

表 22-28 R7 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	'001000'	CMD40
[39:20]	20	'00000h'	保留位
[19:16]	4	X	可接受的电压
[15:8]	8	X	检查模式的返回
[7:1]	7	X	CRC7
0	1	1	结束位

## 22.3. 软件应用说明

### 22.3.1. 电源控制

可以使用以下寄存器和外部电路实现功率控制:

电源使能寄存器 0x00 -这些位的状态反映在 IO 引脚上。时钟寄存器：CLKCR 的 PWRSAV 位，可以选择在总线空闲时，是否关闭时钟。当打开或关闭时开启电源后，请确认“电源设置”正确。通常在切断电源时，对所有卡都可以应关闭电源。

### 22.3.2. 时钟程序

软件可以为卡选择时钟源，选择是否给与时钟以及分频的时钟项目。支持此功能的寄存器 SDIO\_CLKCR:

- (a) CLKEN:时钟的使能位
- (b) CLKDIV:时钟的分频系数，定义了输入时钟(SDIOCLK)和输出时钟(SDIOCK)的分频系数
- (c) SMPCLKSEL:CMD 和 DAT 的预采样时钟沿

SMPEN:预采样使能

CKSEL:CMD&DAT 输出时钟选择

只在设置了 CMD 寄存器的 STARTCMD 位和 REGSYNC 位时才加载这些寄存器。当一个命令被成功加载时，DWC\_mobile\_storage 会清除这个位，除非在队列中已经有另一个命令，这时它会给出一个 HLE (Hardware Locked Error)。软件应该设置 REGSYNC 位(该位为只更新时钟，而不发送命令)，还应该设置 WAITPEND 位，以确保时钟参数在数据传输期间不更改。注意，即使 start\_cmd 被设置为更新时钟寄存器,host 直到命令完成时才会发送 command\_done 信号。

### 22.3.3. 初始化

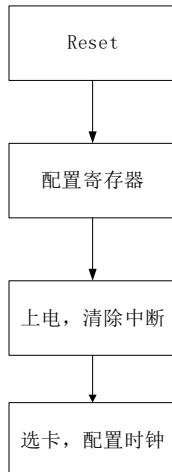


图 22-7 初始话序列

复位将初始化设计中的寄存器、端口、fifo 指针、DMA 接口控件和状态机。上电复位后，软件应执行以下操作：

#### 22.3.3.1. 配置控制寄存器

配置控制寄存器-对于 MMC-Ver3.3-only 模式，通过设置启用 CTRL 寄存器中的 ODPUEN (第 24 位),目前已默认设置为 1。

#### 22.3.3.2. 卡上电

卡上电-在使能电源之前,请确定电压调整期的电压设置正确。通过启用寄存器 0x00 将所连接的卡通电源相应的位设置为 1。

#### 22.3.3.3. 设置中断

通过清除中断掩码寄存器中的适当位来设置中断掩码。设置控制寄存器的 INTEN 位。建议将 0xffff\_ffff 写入原始中断寄存器，以便在设置 INTEN 位之前清除任何挂起的中断。

#### 22.3.3.4. 卡识别过程

主机复位后进入卡识别模式，寻找总线上的新卡。在卡识别模式下，主机复位所有的卡，验证工作电压范围，识别卡并询问每个卡的相对卡地址 (RCA)。这个操作是在每个卡自己的命令信号线 CMD 上分别完成的。在卡识别模式中的所有数据通信只使用命令信号线 (CMD)。在卡识别过程中，卡应该工作在时钟频率为时钟速率 FOD (400 kHz)的情况下。

##### a.卡复位：

命令 GO\_IDLE\_STATE (CMD0) 是软件复位命令，并设置 MMC 和 SD 存储卡进入空闲状态 (Idle State)，不管当前卡的状态是什么。复位命令 (CMD0) 仅用于存储器或组合卡的存储器部分。为了重置只有 I/O 卡或组合卡的 I/O 部分，使用 CMD52 写 1 到 CCCR 的 RES 位。在非激活状态 (Inactive State) 的卡不受此命令的影响。

主机上电后，所有的卡都处于空闲状态 (Idle State)，包括之前已在非激活状态 (Inactive State) 的卡。上电或 CMD0 后，所有卡的 CMD 线处于输入模式，等待下一个命令的起始位。这些卡都是用卡相对地址 (RCA) 初始化，并用默认 400 kHz 的时钟频率驱动器。初始化序列为 80 个周期—这个序列是必须的，在所有卡上电后，可容纳增加的时间。为此，设置 CMD 寄存器的 AUTOINT(第 15 位)。

##### b.工作电压范围确认

在 host 和卡之间开始通信时，主机可能不知道卡支持的电压，并且卡也不知道主机提供的电压是否支持。因此就需要工作电压的确定。

在协议规范中定义的命令包括：SEND\_OP\_COND (CMD1 用于 MMC), SD\_SEND\_OP\_COND (ACMD41 用于 SD 存储卡)，IO\_SEND\_OP\_COND (CMD5 用于 SDI/O 卡)，这些命令提供给主机一种机制去识别和拒绝那些不匹配主机所需的 VDD 范围的卡。这是由主机发送所需的 VDD 电压窗口作为此命令参数来实现的。如果卡不能在指定的范围内进行数据传输，必须从总线断开并进入非激活状态 (Inactive State)。否则，该卡将响应返回它的 VDD 范围。

如果该卡可以工作在所提供的电压下，响应将返回供电电压和在命令参数中设置的检查模式。如果该卡不能在提供的电压下工作，它不返回响应，并保持在空闲状态。初始化 SDHC 卡时强制性的在 ACMD41 命令之前发送 CMD8。收到 CMD8 是让该卡知道主机支持物理层 2.00 协议及卡支持高版本的功能。

#### c. 卡识别过程的过程

对于不同的卡，卡的识别过程不同。这些卡包括 MMC、CE-ATA、SD，或 SD I/O 卡。支持所有类型的 SD I/O 卡，即 SDIO\_IO\_ONLY 卡、SDIO\_MEM\_ONLY 卡和 SDIO COMBO 卡。卡识别过程步骤如下：

1. 检测卡是否连接。
2. 识别卡的类型：SD 卡、MMC(CE-ATA)或 SD I/O 卡。

发送 CMD5 命令。如果主机接收到响应，则是 SD I/O 卡；如果没有响应，发送 ACMD41。如果主机接收到响应，则是 SD 卡；否则，是 MMC 或者 CE-ATA 设备。

3. 根据卡的类型初始化卡。

使用 FOD (400 KHz) 为时钟源，并按照下列命令顺序发送命令：

- SD 卡 - 发送 CMD0, ACMD41, CMD2, CMD3;
- SDHC 卡 - 发送 CMD0, CMD8, ACMD41, CMD2, CMD3;
- SD I/O 卡 - 如果卡没有存储器端口，发送 CMD52, CMD0, CMD5, CMD3；否则，发送 CMD52, CMD0, CMD5, ACMD41, CMD11 (可选), CMD2, CMD3;

4. 识别 MMC/CE-ATA 设备

CPU 应该通过发送 CMD8 查询 EXT\_CSD 寄存器的 504 字节。如果第 4 位被设置为 1，则该设备支持 ATA 模式；如果支持 ATA 模式，CPU 应通过设置 EXT\_CSD 寄存器的 191 字节 (CMD\_SET) 的 (第 4 位) ATA 位选择 ATA 模式，以激活使用 ATA 命令集。CPU 使用 SWITCH (CMD6) 命令选择命令集；

如果 CE-ATA 设备存在，FAST\_IO (CMD39) 和 RW\_MULTIPLE\_REGISTER (CMD60) 命令将会成功，并且返回的数据将会是 CE-ATA 复位信号。

#### d. 设置其他参数

根据规范，可以设置响应时间超时

ResponseTimeOut = 0x64;

DateTimeOut 则选择以下最大值：1.  $(10 * ((TAAC * Fop) + (100 * NSAC)))$ ；2. FIFO 空/满的读写延迟。去抖动的值为 25ms；3. FIFO 的阈值默认为 15。

#### 22.3.4. 无数据命令和没有响应的序列

要发送任何非数据命令，软件需要编写 CMD 寄存器和 CMDARG 寄存器和适当的参数。使用两个寄存器，Host 生成命令并将发送到命令总线。Host 通过 RINTSTS 寄存器的错误位中的来判断正确传输。收到错误或有效的响应时，

Host 将在中断寄存器中设置 command\_done 位。在 response Register0 中放置短响应，而将长响应放置到所有四个响应寄存器。其中 Response3 寄存器位 31 表示 MSB, Response0 寄存器位 0 表示长响应的 LSB。

对于基本命令和非数据命令，请按照以下步骤执行。

1. 用适当的命令参数编程命令寄存器。
2. 按照表 22-29 中的设置编写命令寄存器
3. 等待主机接受命令。将该命令加载到的接受执行命令并清除 start\_cmd 位 CMD 寄存器，除非有一个命令在进程中，此时 DWC\_mobile\_storage 可以加载并保存第二个命令在 buffer 中。

如果 DWC\_mobile\_storage 无法加载命令——也就是说，命令已经在进程中，第二个命令在缓冲区中，然后想进行第三个命令——然后生成一个 HLE(硬件锁定错误)。

表 22-29 无数据命令寄存器配置

parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	无时钟参数更新命令
AUTOINIT	0	可以是 1，但仅用于卡重置命令，如 CMD0
ABORTCMD	0	是否为 1 表示停止数据传输的命令，如 CMD12
AUTOSTOP	0 或 1	0 - 在数据传输结束时没有发送停止命令 1 - 在数据传输结束时发送停止命令
RESPECT	0	命令
RESPLEN	0	R2(长)响应可以为 1
CMD_INDEX		
WAITPEND	1	0 立刻发送命令 1 在之前的数据传输结束后再发送命令
CHECKRESPCRC	1	0 不应检查响应 CRC 1 应检查响应 CRC

4. 检查是否有 HLE。
5. 等待命令执行完成。在收到卡片响应或超时，DWC\_mobile\_storage 设置中断寄存器中的 CD 位。软件可以轮询这个位，也可以响应一个生成的中断。
6. 检查是否设置了 response\_timeout error、response\_CRC error 或 response error。这是可以通过轮询来自 RINTSTS 的第 1,6,8 位，或者这些 error 导致的对应中断。如果没有收到响应错误，则响应是有效的。如果响应错误的话，软件可以从响应寄存器复制响应。当命令正在执行时，软件不应该修改时钟参数。

### 22.3.5. 数据传输命令

数据传输命令在存储卡和 host 之间传输数据。要发送数据命令，host 需要一个命令参数、总数据大小和块大小。软件可以通过 FIFO 接收或发送数据。

在执行数据传输命令之前，软件应该确认卡不繁忙，处于传输状态，这可以分别使用 CMD13 (选卡) 和 CMD7 (发送状态) 命令来完成。对于数据传输命令，重要的是应该在 CLKCR 中设置卡中编程的相同总线宽度。host 在数据传输过程中为不同的条件生成一个中断，具体内容可见中断寄存器。

1. 单块或多块读取涉及的步骤如下：将数据大小(以字节为单位)写入 DLEN 寄存器。
2. 将块大小以字节为单位写入 BLKSIZE 寄存器。host 期望从卡中获得大小为 BLKSIZE 的数据块。

3. 对于有超过 0.5 cclk\_in 周期的往返延迟的卡，编程 Card Read Threshold 以确保卡时钟不会在从卡传输到主机的数据块中间停止;如果读卡阈值功能未启用这样的卡片,那么主机系统应该确保 Rx FIFO 不会成为完整的读数据 传输期间通过确保 Rx FIFO 流出速度快于推入 FIFO 的数据;另外，可以提供一个足够大的 FIFO。

4. 用数据读取开始的数据地址编程 SDIO\_ARG 寄存器。使用如表 6-2 所示的参数编程命令寄存器。对于 SD 和 MMC 卡，使用 CMD17 进行单块读取，使用 CMD18 进行多块读取。对于 SDIO 卡，对于单块和多块传输都使用 CMD53。写入到 CMD 寄存器后，host 开始执行命令;当命令被发送到总线，CD 中断被生成。

表 22-30 单块和多块读的命令寄存器设置

parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	无时钟参数更新命令
AUTOINIT	0	可以是 1，但仅用于卡重置命令，如 CMD0
ABORTCMD	0	是否为 1 表示停止数据传输的命令，如 CMD12
AUTOSTOP	0 或 1	0 - 在数据传输结束时没有发送停止命令 1 - 在数据传输结束时发送停止命令
DTMODE	0	块传输
DIR	0	读卡
DEXPECT	0	数据命令
RESPLEN	0	R2(长)响应可以为 1
WAITRESP	1	没有响应的命令可以为 0，举个例子：CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 立刻发送命令 1 在之前的数据传输结束后再发送命令
CHECKRESPCRC	1	0 不应检查响应 CRC 1 应检查响应 CRC

5. 软件应该查找数据错误中断。如果出现，软件可以通过发送停止命令来终止数据传输。

6. 软件应该在主机超时寻找 Receive\_FIFO\_Data\_request 和/或数据 starvation 条件。在这两种情况下，软件应该从 FIFO 读取数据，并在 FIFO 中留出空间接收更多的数据。

7. 当接收到 DTO 时，软件应该读取 FIFO 中剩余的数据。

### 22.3.6. 单块或者多块写

1. 将数据大小(以字节为单位)写入 DLEN 寄存器。
2. 将块大小以字节为单位写入 BLKSIZE 寄存器,host 发送数据大小为 BLKSIZE 的块。 (是否需要发送 CMD16)
3. 程序 CMDARG 寄存器与数据应该写入的数据地址。
4. FIFO 写入数据;通常最好是从 FIFO 的最深处开始填充数据。使用如表 6-3 所示的参数对命令寄存器进行编程。对于 SD 和 MMC 卡，使用 CMD24 进行单块写，使用 CMD25 进行多块写。对于 SDIO 卡，对于单块和多块传输都使用 CMD53。

表 22-31 单块或多块写的命令寄存器设置

parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	无时钟参数更新命令
AUTOINIT	0	可以是 1，但仅用于卡重置命令，如 CMD0
ABORTCMD	0	是否为 1 表示停止数据传输的命令，如 CMD12
AUTOSTOP	0 或 1	0 - 在数据传输结束时没有发送停止命令

parameter	Value	Comment
		1 -在数据传输结束时发送停止命令
DTMODE	0	块传输
DIR	1	读卡
DEXPECT	1	数据命令
RESPLEN	0	R2(长)响应可以为 1
WAITRESP	1	没有响应的命令可以为 0, 举个例子: CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 立刻发送命令 1 在之前的数据传输结束后再发送命令
CHECKRESPCRC	1	0 不应检查响应 CRC 1 应检查响应 CRC

6. 软件应该查找数据错误中断。如果所需时，软件可以通过发送 STOP 命令终止数据传输。

7. 软件应该从数据中查找 Transmit\_FIFO\_Data\_request 和/或超时条件 starvation。在这两种情况下，软件应该将数据写入 FIFO。

8. 当接收到 Data\_Transfer\_Over 中断时，数据命令结束。对于一个开放式块传输，如果字节计数为 0，软件必须发送 STOP 命令。如果字节数为而不是 0，则在完成给定字节数的传输后，host 应该发送停止命令。AUTO-STOP 命令的完成通过 RINTSTS 寄存器的 ACD 中断。

### 22.3.7. 数据流读

数据流读取类似单块读取或多块读取中提到的块读取，除了命令寄存器中的以下位：

Transfer\_mode=1 cmd\_index = CMD20;流传输只允许单位总线宽度。

### 22.3.8. 数据流写

数据流写就像“单块或多块写”中提到的块写一样，除了命令寄存器中的以下位：

transfer\_mode = 1; cmd\_index = CMD11;

在流传输中，如果字节计数为 0，那么软件必须发送 STOP 命令。如果字节计数不为 0，那么当给定的字节数完成一个传输时，DWC\_mobile\_storage 发送 STOP 命令。这个 AUTO\_STOP 命令的完成被 Auto\_command\_done 中断反映出来。

流传输只允许单位总线宽度。

### 22.3.9. 在传输过程中发送停止和中断

STOP 命令可以终止存储卡和 DWC\_mobile\_storage 之间的数据传输，而 ABORT 命令只能终止 SDIO\_IOONLY 和 SDIO\_COMBO 卡之间的 I/O 数据传输。

发送停止命令-当数据传输正在进行时，可以在命令行上发送;这在数据传输过程中,可以在任何时间发送命令。有关发送此命令的信息，参考“无响应序列的无数据命令”。

您还可以使用此命令的附加设置来设置命令寄存器 bits(5-0)为 CMD12，并将第 14 位 ABORTCMD 设置为 1。如果 ABORTCMD 未设置为 1，则 DWC\_mobile\_storage 不知道用户停止了数据传输。复位命令寄存器(WAITPEND)为 0 以便 DWC\_mobile\_storage 立即发送命令（即使有数据传输在进行中）。

发送中止命令只能在 SDIO\_IOONLY or SDIO\_COMBO 卡上使用。要中止正在传输数据的函数，可以使用 CMD52 在 ASx 位(卡的 CCCR 寄存器，地址 0x06)中编写函数号。这是非数据命令。关于发送此命令的信息，请参见无数据命令没有响应序列。

### 22.3.10. 暂停或恢复序列

在 SDIO 卡中, I/O 功能和 host 之间的数据传输可以使用 SUSPEND 命令暂时停止;这可能是为了执行高优先级, 具有另一功能的数据传输。如果需要, 可以使用 RESUME 命令恢复数据传输。

以下函数可以通过在 SDIO 卡的 CCCR 寄存器(卡片通用控制寄器)中编写适当的位来实现。要读取或写入 CCCR 寄存器, 使用 CMD52 命令。

#### 1. SUSPEND data transfer—非数据传输命令

- a. 检查 SDIO 卡是否支持 SUSPEND/RESUME 协议;可以通过卡的 CCCR 寄存器 @0x08 中的 SBS 位来完成。
- b. 检查所需功能的数据传输是否正在进行;当前活动的函数号反映在 CCCR 寄存器的 0-3 位上。
- c. 为了暂停传输, 设置 CCCR 寄存器的第 2 位。
- d. 轮询 CCCR 的 BR(第 1 位)和 BS(第 0 位)的清除状态。当前选择的功能正在使用数据总线时, BS (Bus Status) 位为 1;BR(总线发布)位保持 1, 直到总线发布完成。当 BR 和 BS 位均为 0 时, 所选功能的数据传输暂停。
- e. 在读取数据传输过程中, DWC\_mobile\_storage 可以等待卡上的数据。如果数据传输是从卡读取的, 那么在 SUSPEND 命令成功完成后必须通知 DWC\_mobile\_storage。DWC\_mobile\_storage 然后重置数据状态机并退出等待状态。要实现这一点, 在 Control 寄存器中设置。
- f. 等待数据完成。通过读取 TCBCNT 寄存器获得要传输的挂起字节。

#### 2. RESUME data transfer—这是一个数据命令。

- a. 检查卡是否处于传输状态, 确认总线没有数据传输。
- b. 如果卡处于断开状态, 使用 CMD7 选择它。卡状态可以在响应 CMD52/CMD53 命令。
- c. 检查要恢复的函数是否准备好进行数据传输;这一点可以通过 CCCR 的 RFx 标志。如果 RF = 1, 则函数可以进行数据传输。
- d. 若要恢复传输 CMD52, 将函数号写入 CCCR 寄存器 @0x0D 的 FSx 位(0-3)。CMD52 的命令参数, 并将其写入 CMDARG。
- e. 将块大小写入 BLKSIZ 寄存器;数据将以该块为单位进行传输大小。
- f. 将字节数写入 BYTCNT 寄存器。这是数据的总大小;也就是说, 剩余要传输的字节数。处理数据是软件的责任。
- g. 程序命令寄存器;类似于块传输。具体操作请参见“多块读取”和“单块或多块写入”。
- h. 当命令寄存器被编程后, 命令被发送, 功能恢复
- 数据传输。读取 DF 标志(恢复数据标志)。如果为 1, 则函数有传输, 一旦功能或内存恢复, 将开始数据传输。如果是 0, 那么函数没有数据进行传输。
- i. 如果 DF 标志为 0, 那么在读取的情况下, host 等待数据。数据超时周期后, 它给出一个数据超时错误。

表 22-32 CMDARG bit 值

CMDARG bits	Content	Value
31	R/W flag	1
30-28	Function Number	0,for CCCR access
27	RAW flag	1,read after write
26		
25-9	Register address	0x0d
8		
7-0	Write data	Function number that is to be resumed

#### 22.3.11. 读等待序列

Read\_wait 仅用于 SDIO 卡, 可以暂时停止数据传输——来自函数或内存, 并允许主机发送命令到 SDIO 设备中的任何功能。主机可以需要多久就拖延多久。

操作步骤:

1. 检查卡是否支持 read\_wait 工具;读取 CCCR 寄存器@0x08 的 SRW(第 2 位)。如果这位为 1, 那么卡中的所有函数都支持 read\_wait 工具。使用 CMD52 来读取这个位。
2. 如果卡支持 read\_wait 信号, 那么通过在 CTRL 中设置 read\_wait(第 6 位)来断言它登记。
3. 清除 CTRL 寄存器中的 read\_wait 位。

### 22.3.12. 控制器/DMA/FIFO 重置使用

与卡的交互包括以下内容:

1. 控制 host 的所有功能。
2. FIFO -用来保存要发送或接收的数据。
3. DMA -如果启用 DMA 传输模式, 在系统内存和 FIFO 之间传输数据。通过在 CTRL 寄存器中设置 controller\_reset 位(第 0 位)来重置控制器;这会重置 CIU 和状态机, 也会重置 biu -CIU 接口。因为这个重置位是自清除, 在发出复位后, 等待直到该位被清除。

4. FIFO reset -通过设置 FIFO 复位(CTRL 寄存器第 1 位)来复位 FIFO;这个重置 FIFO 指针和 FIFO 计数器。由于这个复位是自清除的, 在发出复位后, 等待这部分被清除。以下是发出 reset 命令的推荐方法:

1. 非 dma 传输模式-同时设置 controller\_reset 和 fifo\_reset;使用另一个写操作清除中断寄存器, 以清除任何产生的中断。
2. 通用的 DMA 方式: 同时设置 controller\_reset, fifo\_reset 和 dma\_reset;通过使用另一个写操作来清除中断寄存器, 以清除任何产生的中断。如果需要完成 DMA, 在重置 DMA 接口控制和发出额外的 FIFO 重置之前, 轮寻状态寄存器以查看 DMA 请求是否为 0。

### 22.3.13. 专用中断

专用中断端口不需要特殊的编程。在这种情况下, 应用程序必须读取所有连接到中断的函数, 以便检测是哪个函数生成了中断。

### 22.3.14. 异步中断

异步中断机制不需要特殊的编程。应用程序必须读取所有连接到标记中断的函数, 以检测是哪个函数生成了中断。在此模式下, 卡可以使用 DAT[1]线路或专用中断端口发送中断

### 22.3.15. 错误处理

DWC\_mobile\_storage 实现错误检查;ERROR 反映在 RAWINTS 寄存器中并且可以通过中断与软件通信, 或者软件可以轮询这些位。在上电后, 中断被禁用(CTRL 寄存器中的 int\_enable 为 0), 并且所有中断都被屏蔽(INTMASK 寄存器的 0-31 位;默认是 0)

错误处理:

a 响应和数据超时错误

对于响应超时, 软件可重发命令。DWC\_mobile\_storage 还没有收到数据开始位在数据超时时, 无论是第一个块还是中间块-在超时时间内, 因此软件可以重试再次整个数据传输或从指定的块开始传输。之后通过阅读 TCBCNT 的内容, 可以了解软件可以还有多少字节需要复制。

b 响应错误

当响应接收过程中收到错误时设置。例如, 响应在响应寄存器中出现的是无效的。软件可以重新发送该命令。

**c 数据错误**

当观测到数据接收错误时设置;例如, data CRC, 起始位不已找到, 未找到结束位, 等等。这些错误会发生在任何数据块中—第一块, 中间块, 或最后一个块。在收到错误时, 软件可以发出 STOP 或 ABORT 命令, 并针对于整个数据或部分数据重新发送该命令。

**d 硬件锁定错误**

当 DWC\_mobile\_storage 不能设置软件来加载命令时。当软件在 CMD 寄存器中设置 STARTCMD 位时, DWC\_mobile\_storage 尝试加载命令。如果命令缓冲区已经被命令填充, 则会引发此错误。软件必须重新加载命令。

**f FIFO underrun/overrun 错误**

如果 FIFO 是满的, 软件试图写入数据在 FIFO, 然后设置溢出错误。相反, 如果 FIFO 是空的, 软件试图从 FIFO 读, 一个 underrun 错误被设置。在 FIFO 读取或写入数据之前, 软件应该先读取状态寄存器中的 fifo\_empty 或 fifo\_full 位。

**g 主机超时导致的数据匮乏**

当 DWC\_mobile\_storage 正在等待软件干预来将数据传输到 FIFO 或从 FIFO 传输出去, 但软件没有在规定的超时时间内进行传输时导致数据匮乏。在这种情况下, 当读传输过程中, 软件从 FIFO 读取数据, 并为进一步的数据接收提供空间。当一个传输操作正在进行时, 软件应该在 FIFO 中填充数据, 以便开始向卡传输数据。

**h 命令中的 CRC 错误**

如果检测到命令 CRC 错误, CE-ATA 设备不会发送响应且预期 DWC\_mobile\_storage 响应超时。ATA 层标识 MMC 传输层发生错误。

**I 写操作**

任何已知对于设备的任何 MMC 传输层错误都会导致终止一个未执行的 ATA 命令。ERR 位在 ATA 状态寄存器, 相应的错误码被发送到 ATA 错误寄存器。如果 nIEN=0, 则向主机发送 CCS。如果设备中断未启用(nIEN=1), 则如果主机控制器不终止正在进行的传输, 则设备完成整个 Data Unit Count。

**j 读操作**

主控制器可以发出一个命令完成信号禁用信号(CCSD), 在一个停止传输命令发送后(CMD12), 以中止读传输。

如果主机控制器检测到 MMC 传输层错误, 则主机以错误状态完成 ATA 命令。主控制器可以发出一个命令完成信号禁用(CCSD), 然后是一个停止传输(CMD12), 以中止读传输。主机还可以在不中断数据传输的情况下传输整个 Data Unit Count 字节。

### **22.3.16. CT\_ATA 操作**

介绍 CE-ATA 数据传输命令。关于不同条件下产生的基本设置和中断的信息可以看设计传输命令

#### **22.3.16.1. 复位和设备 recovery**

在开始 CE-ATA 操作之前, 主机应该执行 MMC 复位和初始化过程。在完成正常的 MMC 重置和初始化过程后, 主机应该使用 RW\_REG/CMD39 查询初始的 ATA 值。默认情况下, MMC 块大小为 512 字节, 由 CE-ATA 设备内部的 srcControl 寄存器的 1:0 位表示。主机可以协商使用 1KB 或 4KB 的 MMC 块大小。设备通过 srcCapabilities 寄存器指示它可以支持的 MMC 块大小;主机读取这个寄存器是为了协商 MMC 块的大小。当主控制器将 MMC 块大小写入设备的 srcControl 寄存器位为 1:0 时则完成。

#### **22.3.16.2. ATA Task File 传输**

在设置 CMDARG 和 CMD 寄存器之前, 主机软件堆栈将任务文件映像写入 FIFO。然后, 主机处理器在设置 CMD(偏移 0x2C)寄存器位之前, 在 CMDARG 中设置地址和字节计数。对于 RW\_REG, CE-ATA 设备没有命令完成信号。

### 22.3.16.3. 使用 RW\_MULTIPLE\_REGISTER 进行 ATA 任务文件传输

该命令涉及 CE-ATA 设备与 host 之间的数据传输。要发送数据命令，host 需要一个命令参数、总数据大小和块大小。软件可以通过 FIFO 接收或发送数据 ATA 文件传输(读或写)涉及的步骤如下：

1. 将数据大小(以字节为单位)写入 BYTCNT 寄存器
2. 将块大小以字节为单位写入 BLKSIZ 寄存器
3. DWC\_mobile\_storage 需要一个单独的块传输。
4. 用起始寄存器地址编写 CMDARG 寄存器。

表 22-33 CMDARG 编程寄存器

CMDARG bits	Content	Value
31	R/W flag	1or0
30-24	0	Reserved ,bits cleared to 0 by host processor
23:18	0	Starting register address for read/write; Dword aligned
17:16	0	Register address; Dword aligned
15:8	0	Reserved; bits cleared to 0 by host processor
7:2	16	Number of bytes to read/write; integral number of Dwords
1:0	0	Byte count in integral number of Dword

表 22-34 CMD 寄存器

parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	无时钟参数更新命令
DIR	1 或 0	读卡, 或者写卡
IEN	0	Command Completion signals Done signals
ATACMD	1 或 0	CE_ATA CMD
DEXPECT	1	数据命令预期
RESPLEN	0	R2(长)响应可以为 1
WAITRESP	1	没有响应的命令可以为 0, 举个例子: CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 立刻发送命令 1 在之前的数据传输结束后再发送命令
CHECKRESPCRC	1	0 不应检查响应 CRC 1 应检查响应 CRC

### 22.3.16.4. 使用 RW\_MULTIPLE\_REGISTER 进行 ATA 传输

在等待命令完成信号(CCS)的 RW\_BLK 时, 主机可以发送一个命令完成信号禁用(CCSD)。发送 CCSD DWC\_mobile\_storage 发送 CCSD 到 CE-ATA 设备, 如果 send\_ccsd 位设置在 CTRL 寄存器;这个位应该只在收到 RW\_BLK 的响应后设置。在发送 CCSD 模式后, 发送内部生成的 Stop (CMD12)命令。如果 send\_auto\_stop\_ccsd 位也设置在控制器编程发送 CCSD 模式时, DWC\_mobile\_storage 在 CMD 行上发送内部生成的 STOP 命令。发送 STOP 命令后, DWC\_mobile\_storage 在 RINTSTS 寄存器中设置 Auto command Done 位。

## 22.4. 寄存器配置

### 22.4.1. SDIO 电源控制寄存器 (SDIO\_POWER)

地址偏移: 0x00

复位值: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	Reserved	RW	0	保留, 始终读为 0。
0	PWRCTRL	RW	0	PWRCTRL:电源控制位 (Power supply control bits) 这些位用于定义卡时钟的当前功能状态： 0: 电源关闭, 卡的时钟停止。 1: 保留的上电状态。

注: 写数据后的 7 个 HCLK 时钟周期内, 不能写入这个寄存器。

## 22.4.2. SDIO 时钟控制寄存器 (SDIO\_CLKCR)

地址偏移: 0x04

复位值: 0x0000 7000

SDIO\_CLKCR 寄存器控制 SDIO\_CK 输出时钟。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CKSEL	SMPEN	SMP-CLKSEL	WIDBUS		PWRSAV	CLKEN	CLKDIV[0: 7]							
	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	RW	0	保留, 始终读为 0。
14	CKSEL	RW	1	CMD & DAT 输出时钟选择: 0: SD_CLK 时钟偏移 90 度输出; 1: SD_CLK 时钟偏移 180 度输出;
13	SMPEN	RW	1	预采样使能: 0: 不使用预采样时钟; 1: 使用预采样时钟;
12	SMP-CLKSEL	RW	1	CMD & DAT 预采样时钟选择 (都是上升沿) : 1: SD_CLK 时钟; 0: SD_CLK 时钟偏移 270 度;
11: 10	WIDBUS	RW	0	WIDBUS: 宽总线模式使能位 (Wide bus mode enable bit) 00: 默认总线模式, 使用 SDIO_D0。 01: 4 位总线模式, 使用 SDIO_D[3:0]。 10/11: 8 位总线模式, 使用 SDIO_D[7:0]。

Bit	Name	R/W	Reset Value	Function
9	PWRSAV	RW	0	PWRSAV: 省电配置位 (Power saving configuration bit) 为了省电, 当总线为空闲时, 设置 PWRSAV 位可以关闭 SDIO_CK 时钟输出。 0: 始终输出 SDIO_CK。 1: 仅在有总线活动时才输出 SDIO_CK。
8	CLKEN	RW	0	CLKEN: 时钟使能位 (Clock enable bit) 0: SDIO_CK 关闭。 1: SDIO_CK 使能。
7:0	CLKDIV	RW	0	CLKDIV: 时钟分频系数 (Clock divide factor) 这个域定义了输入时钟 (SDIOCLK) 与输出时钟(SDIO_CK)间的分频系数: SDIO_CK 频率 = SDIOCLK/[CLKDIV*2]

注意:

- 当 SD/SDIO 卡或多媒体卡在识别模式, SDIO\_CK 的频率必须低于 400kHz。
- 当所有卡都被赋予了相应的地址后, 时钟频率可以改变到卡总线允许的最大频率。
- 写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。对于 SD I/O 卡, 在读等待期间可以停止 SDIO\_CK, 此时 SDIO\_CLKCR 寄存器不控制 SDIO\_CK。

#### 22.4.3. SDIO 参数寄存器 (SDIO\_ARG)

地址偏移: 0x08

复位值: 0x0000 0000

SDIO\_ARG 寄存器包含 32 位命令参数, 它将作为命令的一部分发送到卡中。SDIO\_CLKCR 寄存器控制 SDIO\_CK 输出时钟。

31:0	CMDARG[31:0]	RW
------	--------------	----

Bit	Name	R/W	Reset Value	Function
31: 0	CMDARG	RW	0	CMDARG: 命令参数 (Command argument) 命令参数是发送到卡中命令的一部分, 如果一个命令包含一个参数, 必须在写命令到命令寄存器之前加载这个寄存器。

#### 22.4.4. SDIO 命令寄存器 (SDIO\_CMD)

地址偏移: 0x0C

复位值: 0x2000 0000

SDIO\_CMD 寄存器包含命令索引和命令类型位。命令索引是作为命令的一部分发送到卡中。命令类型位控制命令通道状态机(CPSM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTCM D	Res	USE- HOLDREG	Res	BOOT- MODE	BOOTDIS	BOOTACK	BOOTEN	IEN	ATACMD	REGSYNC	Res				
rw		rw		rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

AUTOINIT	ABORTCMD	WAIT-PEND	AU-TOSTOP	DTMODE	DIR	DEXPECT	CHECK-RESPCRC	RESPLEN	WAIT-RESP	rwCMDINDEX[0:5]
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	Name	R/W	Re-set Value	Function
31	STARTCMD	RW	0	Start command 开始命令。一旦命令被 CIU 接收，位就会被清除。 当该位被设置后，主机不能够写入命令寄存器。如果试图写，硬件锁错误将在 interrupt register 中设置。 一旦发送命令，并从 SD_MMC_CEATA 卡收到响应，命令完成位在 interrupt register 中设置。
30	RESERVED	RW	0	RESERVED
29	USEHOLDREG		1	强制为 1，软件读为 0
28	RESERVED	RW	0	RESERVED
27	BOOTMODE	RW	0	Boot Mode 0 - 强制启动操作 1 - 备用启动操作
26	BOOTDIS	RW	0	Disable_boot 禁用启动。当软件这个位和 start_cmd 一起设置时，CIU 将终止引导操作。“disable_boot”和“enable_boot”不能同时设置。
25	BOOTACK	RW	0	expect_boot_ack 预期启动响应。当 Software 将此位与 enable_boot 一起设置时。所选卡上模式为 0-1-0，CIU expect 确认启动时。
24	BOOTEN	RW	0	enable_boot 启用启动—该位仅应设置为强制启动模式。 当 Software 将此位与 start_cmd 一起设置时，CIU 通过断言 CMD 启动对应卡的 boot 线拉低。“disable_boot”和“enable_boot”不能同时设置。
23	IEN	RW	0	ccs_expected 0 - CE-ATA 设备中没有启用中断(在 ATA 控制寄存器中 nIEN = 1)，或者命令收到来自设备 CCS 的 expect。 1 - CE-ATA 设备上启用中断(nIEN = 0)，RW_BLK 命令收到 CE-ATA 设备发出命令完成信号。
22	ATACMD	RW	0	ATACMD 0 主机对 CE-ATA 设备没有执行读访问(RW_REG 或 RW_BL)。 1 主机对 CE-ATA 设备执行读访问(RW_REG 或 RW_BL)。

Bit	Name	R/W	Re-set Value	Function
21	REGSYNC	RW	0	<p>update_clock_registers_only 0 -正常的命令序列 1 -不发送命令，只是更新时钟寄存器值到卡时钟域 把以下寄存器值转移到卡时钟域: CLKDIV, CLKENA. 改变卡片时钟(改变频率, 切断或打开, 设置低频模式);提供了改变时钟频率或停止时钟, 而无需发送命令到卡 在正常命令序列中, update_clock_registers_only = 0 时, 以下控制寄存器从 BIU 转移到 CIU: CMD, CMDARG, TMOUT, BLKSIZ, bytct。CIU 为卡的新命令序列使用新的寄存器值。 当设置位时, 没有命令完成中断, 因为没有命令发送到 SD_MMC_CEATA 卡。</p>
20:16	Reserved	RW	0	
15	AUTOINIT	RW	0	<p>send_initialization 0 -在发送此命令之前, 不发送初始化序列(80 个时钟) 1 -在发送此命令之前发送初始化序列</p>
14	ABORTCMD	RW	0	<p>stop_abort_cmd 0 -既不停止也不中止命令停止当前正在进行的数据传输。 如果 abort 被发送到当前选择的函数 number 或不在数据传输模式中, 那么 bit 应该设置为 0。 1 -停止或中止命令为了停止当前正在进行的数据传输。</p>
13	WAITPEND	RW	0	<p>wait_prvdata_complete 0 -立即发送命令, 即使之前的数据传输还没有完成 1 -等待之前的数据传输完成之前发送命令。 wait_prvdata_complete = 0 通常用于在数据传输时查询卡状态或停止当前数据传输;Card_number 应该与前面的命令相同。</p>
12	AUTOSTOP	RW	0	<p>send_auto_stop 0 -在数据传输结束时没有发送停止命令 1 -在数据传输结束时发送停止命令 注: 在命令寄存器中设置 send_auto_stop 位时在命令路径中加载。auto-stop 命令帮助发送精确的数据字节数, 使用 MMC 的流读或写, 以及 SD 卡的多块读或写。</p>
11	DTMODE	RW	0	<p>transfer_mode 0 -块数据传输命令 1 -流数据传输命令 不关心是否是预期数据。</p>
10	DIR	RW	0	<p>read/write 0-从卡片中读取 1 -写在卡片上 不关心预期卡中是否有数据。</p>

Bit	Name	R/W	Re-set Value	Function
9	DEXPECT	RW	0	data_expected 0 -预期没有数据传输(读/写) 1 -预期的数据传输(读/写)
8	CHECKRESPCRC	RW	0	check_response_crc 0 -不检查响应 CRC 1 -检查响应 CRC 一些命令响应不返回有效的 CRC 位。 软件应该禁用这些命令的 CRC 检查，以禁用控制器的 CRC 检查。
7	RESPLEN	RW	0	response_length 0 -预期从卡片中得到短回应 1 -预期从卡片中得到长回应
6	WAITRESP	RW	0	response_expect 0 -没有预期从卡收到响应 1 -期望从卡片上收到响应
5:0	CMDINDEX	RW	0	CMDINDEX 命令索引

注：1：写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

2：多媒体卡可以发送 2 种响应：48 位长的短响应，或 136 位长的长响应。SD 卡和 SD I/O 卡只能发送短响应，参数可以根据响应的类型而变化，软件将根据发送的命令区分响应的类型。CE-ATA 设备只发送短响应。

#### 22.4.5. SDIO 命令响应寄存器 (SDIO\_RESPCMD)

地址偏移：0x10

复位值：0x0000 0000

SDIO\_RESPCMD 寄存器包含最后收到的命令响应中的命令索引。如果传输的命令响应不包含命令索引(长响应或 OCR 响应)，尽管它应该包含 111111b(响应中的保留域值)，但 RESPCMD 域的内容未知。

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

Bit	Name	R/W	Reset Value	Function
31: 6	保留	RW	0	保留，始终读为 0。
5:0	RESPCMD	R	0	RESPCMD：响应的命令索引 (Response command index) 只读位，包含最后收到的命令响应中的命令索引。

#### 22.4.6. SDIO 命令响应 1..4 寄存器 (SDIO\_RESPX)

地址偏移：0x14+4\*(x-1), 其中 x = 1..4

复位值: 0x0000 0000

SDIO\_RESP1/2/3/4 寄存器包含卡的状态，即收到响应的部分信息。

31:0
RESPX[31:0]
r

Bit	Name	R/W	Reset Value	Function
31: 0	CARDSTATUSX	R	0	CARDSTATUSx: 见下表

根据响应状态，卡的状态长度是 32 位或 127 位。

表 22-35 响应类型和 SDIO\_RESPx 寄存器

寄存器	短响应	长响应
SDIO_RESP1	卡状态[31:0]	卡状态[127:96]
SDIO_RESP2	不用	卡状态[95:64]
SDIO_RESP3	不用	卡状态[63:32]
SDIO_RESP4	不用	卡状态[31:1]

总是先收到卡状态的最高位，SDIO\_RESP3 寄存器的最低位始终为 0。

#### 22.4.7. SDIO 数据定时器寄存器 (SDIO\_TMOOUT)

地址偏移: 0x24

复位值: 0xFFFFFFFF40

SDIO\_DTIMER 寄存器包含以卡总线时钟周期为单位的数据超时时间。

一个计数器从 SDIO\_DTIMER 寄存器加载数值，并在数据通道状态机(DPSM)进入 Wait\_R 或繁忙

状态时进行递减计数，当 DPSM 处在这些状态时，如果计数器减为 0，则设置超时标志。

31:8	DATATIME[31:8]	7:0
	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	DATATIME	RW	0xFFFFFFFF	DATATIME: 数据超时时间 (Data timeout period) 以卡总线时钟周期为单位的数据超时时间。
7:0	RESPTIME	RW	0X40	RESPTIME: 命令超时时间 (CMD timeout period) 以卡总线时钟周期为单位的数据超时时间。

注：在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

#### 22.4.8. SDIO 数据块长度寄存器 (SDIO\_BLKSIZE)

地址偏移: 0x28

复位值: 0x0000 0200

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

Bit	Name	R/W	Reset Value	Function
31: 16		Res	0	保留始终为 0
15:0	DBLOCKSIZE	RW	0x200	DBLOCKSIZE: 数据块长度 (Data block size) 当选择了块数据传输模式，该域定义数据块长度

#### 22.4.9. SDIO 数据长度寄存器 (SDIO\_DLEN)

地址偏移: 0x2C

复位值: 0x0000 0200

SDIO\_DLEN 寄存器包含需要传输的数据字节长度。当数据传输开始时，这个数值被加载到数据计数器中。

31:25	24:0
Res	DATALENGTH RW

Bit	Name	R/W	Reset Value	Function
31:25		Res	0	保留始终为 0
24:0	DATALENGTH	RW	0x200	DATALENGTH: 数据长度 (Data length value) 要传输的数据字节数目。

注意 对于块数据传输，数据长度寄存器中的数值必须是数据块长度(SDIO\_BLKSIZ)的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

#### 22.4.10. SDIO 控制寄存器 (SDIO\_CTRL)

地址偏移: 0x30

复位值: 0x01000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								ODPUE	Res						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				CEA-TAINEN	AU-TOS-TOPCCSD	CCSDEN	ABORTRD	AU-TOIRQRESP	READWAIT	DMAEN	INTEN	Res		FIFORST	SDIORST
				RW	RW	RW	RW	RW	RW	RW	RW			RW	RW

Bit	Name	R/W	Reset Value	Function
31: 25	Reserved			
24	ODPUE	RW	1	enable_od_pullup 外部 OD 上拉： 0: Disable 1: Enable

Bit	Name	R/W	Reset Value	Function
				CMD 线上拉，当上拉使能 SDIO 输出 0 或者高阻，不会输出 1；（MMC 初始化时使用，在识别结束之后需要修改，可以通过配置 gpio 修改）
23: 12	Reserved			
11	CEATAINTEN	RW	0	ceata_device_interrupt_status 0: 在 CE-ATA 设备中，中断没有使能； 1: 在 CE-ATA 设备中，中断使能；
10	AUTOSTOPCCSD	RW	0	send_auto_stop_ccsd 1: 在向 CE-ATA 发送 CCSD 后，自动发送 STOP； 发送结束后，该 bit 自动清零；controller_reset 可以复位该 BIT，也可以软件写 0
9	CCSDEN	RW	0	send_ccsd 1: 向 CE-ATA 发送 CCSD； 设置后，DWC_mobile_storage 将 CCSD 发送到 CE-ATA 设备。 只有当前命令期望 CCS(即 RW_BLK)并在 CE-ATA 设备中启用中断时，软件才会设置此位。一旦 CCSD 模式被发送到设备，SDIO 自动清除 send_ccsd 位。它还在 RINTSTS 寄存器中设置命令完成(CD)位，如果命令完成中断没有被屏蔽，则生成到主机的中断。
8	ABORTRD	RW	0	abort_read_data 0 -没有变化 1 -当挂起命令在读取传输期间发出后，软件轮询卡来查找挂起发生的时间。一旦挂起，软件设置位重置数据状态机，等待下一个数据块。位自动清除一旦数据状态机重置为空闲。 用于 SDIO 卡挂起队列。
7	AUTOIRQRESP	RW	0	send_irq_response 0 -没有变化 1 -发送自动 IRQ 响应 一旦响应发送，位自动清除。为了等待 MMC 卡中断，主机发出 CMD40, SDIO 等待来自 MMC 卡的中断响应。同时，如果主机希望 SDIO 退出等待中断状态，可以设置此位，此时 SDIO 命令状态机在总线上发送 CMD40 响应，返回空闲状态。
6	READWAIT	RW	0	read_wait 0 -清除读等待 1 -断言读等待 用于向 SDIO 卡发送读等待。
5	DMAEN	RW	0	DMAEN 0 -禁用 DMA 传输模式

Bit	Name	R/W	Reset Value	Function
				1 -启用 DMA 传输模式
4	INTEN	RW	0	int_enable 全局中断启用/禁用位: 0-禁用中断 1-启用中断 只有当该位为 1 且有一个或多个未屏蔽中断位设置时, int 端口才为 1
3: 2	Reserved			
1	FIFORST	RW	0	fifo_reset 0 -没有变化 1 – 复位数据 FIFO, 复位 FIFO 指针 要复位 FIFO, 固件应该将 bit 设置为 1。这 bit 在完成复位操作之后会自动清除
0	SDIORST	RW	0	controller_reset 0 -没有变化 1_重置控制器 要复位控制器, 固件应该设置位为 1。这个位是自动清除的。

#### 22.4.11. SDIO 状态寄存器 (SDIO\_STA)

地址偏移: 0x34

复位值: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res		FIFO_CNT													Res	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res						CARDBSY	CARD-PRESENT	CMDFSM					FIFOF	FIFOE	TXVMARK	RXVMARK
						R	R					R	R	R	R	

Bit	Name	R/W	Reset Value	Function
31:30	RESERVED			
29:17	FIFO_CNT	R	0	FIFO count
16:10	RESERVED			
9	CARDBSY	R	0	data_busy 选中 card_data[0]状态的相反 0—卡数据不忙 1—卡数据繁忙
8	CARDPRESENT	R	0	data_3_status 选择 card_data [3];检查卡片是否存在

				0 - 卡片不存在 1 - 卡片存在
7:4	CMDFSM	R	0	cmd_fsm_states 命令 FSM 状态: 0 - 闲置 1 - 发送 init 序列 2 - Tx cmd 起始位 3 - Tx cmd Tx 位 4 - Tx cmd 索引+参数 5 - Tx cmd crc7 6 - Tx cmd 结束位 7 - Rx 响应起始位 8 - Rx 响应 IRQ 响应 9 - Rx 响应 tx 位 10 - Rx 响应 cmd index 11 - Rx 响应数据 12 - Rx 对应 crc7 13 - Rx 响应结束位 14 - Cmd 路径等待 NCC 15 - wait; CMD-to-response turnaround
3	FIFOF	R	0	fifo_full FIFO 满
2	FIFOE	R	1	fifo_empty FIFO 空
1	TXWMARK	R	1	fifo_tx_watermark FIFO 达到传输水位线; 不符合数据传输的条件.
0	RXWMARK	R	0	fifo_rx_watermark FIFO 达到输出水位线; 不符合数据传输的条件

#### 22.4.12. SDIO 中断状态寄存器 (SDIO\_INTSTS)

地址偏移: 0x38

复位值: 0x0000 0000

在对应位置'1', SDIO\_INTSTS 的该位清 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															SDIONT
															Rc_w 1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EBE	ACD	SBE	HLE	FRUN	HTO	DRT0_BDS	RTO_BAR	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	CAD

Rc_w 1																
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	SDIOINT	Rc_w1	0	sdio_int_mask 0 -没有来自卡的 SDIO 中断 1 -从卡中断 SDIO
15	EBE	Rc_w1	0	bit 15 - End-bit error (read)/write no CRC (EBE)
14	ACD	Rc_w1	0	bit 14 -自动命令完成(ACD)
13	SBE	Rc_w1	0	bit 13 -起始位错误(SBE)
12	HLE	Rc_w1	0	bit 12 -硬件锁定写错误(HLE)
11	FRUN	Rc_w1	0	bit 11 - FIFO underrun/溢出错误(FRUN)
10	HTO	Rc_w1	0	bit 10 - host timeout (HTO)
9	DRTO_BDS	Rc_w1	0	bit 9 -数据读超时(DRTO_BDS)
8	RTO_BAR	Rc_w1	0	bit 8 -响应超时(RTO_BAR)
7	DCRC	Rc_w1	0	bit 7 -数据 CRC 错误(DCRC)
6	RCRC	Rc_w1	0	bit 6 -响应 CRC 错误(RCRC)
5	RXDR	Rc_w1	0	bit 5 -接收 FIFO 数据请求(RXDR)
4	TXDR	Rc_w1	0	bit 4 -发送 FIFO 数据请求(TXDR)
3	DTO	Rc_w1	0	bit 3 -数据传输(DTO)
2	CD	Rc_w1	0	bit 2 -命令完成(CD)
1	RE	Rc_w1	0	bit 1 -响应错误(RE)
0	CAD	Rc_w1	0	bit 0 -卡检测(CAD)

### 22.4.13. SDIO 中断掩码寄存器 (SDIO\_INTMASK)

地址偏移: 0x3C

复位值: 0x0000 0000

在对应位置‘1’， SDIO\_MASK 中断掩码寄存器决定哪一个状态位产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															SDIOINTIE
															RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EBeIE	ACDIE	SBEIE	HLE	FRUNIE	HTOIE	DRTQ_BDSIE	RTO_BARIE	DCRCIE	RCRCIE	RXDRIE	TXDRIE	DTOIE	CDIE	REIE	CADIE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			

Bit	Name	R/W	Reset Value	Function
16	SDIOINTIE	RW	0	sdio_int_mask 0 -没有来自卡的 SDIO 中断 1 -从卡中断 SDIO
15	EBEIE	RW	0	bit 15 - End-bit error (read)/write no CRC (EBE)
14	ACDIE	RW	0	bit 14 -自动命令完成(ACD)
13	SBEIE	RW	0	bit 13 -起始位错误(SBE)
12	HLEIE	RW	0	bit 12 -硬件锁定写错误(HLE)
11	FRUNIE	RW	0	bit 11 - FIFO underrun/溢出错误(FRUN)
10	HTOIE	RW	0	bit 10 - host timeout (HTO)
9	DRTO_BDSIE	RW	0	bit 9 -数据读超时(DRTO_BDS)
8	RTO_BARIE	RW	0	bit 8 -响应超时(RTO_BAR)
7	DCRCIE	RW	0	bit 7 -数据 CRC 错误(DCRC)
6	RCRCIE	RW	0	bit 6 -响应 CRC 错误(RCRC)
5	RXDRIE	RW	0	bit 5 -接收 FIFO 数据请求(RXDR)
4	TXDRIE	RW	0	bit 4 -发送 FIFO 数据请求(TXDR)
3	DTOIE	RW	0	bit 3 -数据传输(DTO)
2	CDIE	RW	0	bit 2 -命令完成(CD)
1	REIE	RW	0	bit 1 -响应错误(RE)
0	CADIE	RW	0	bit 0 -卡检测(CAD)

#### 22.4.14. SDIO FIFO 阈值寄存器 (SDIO\_FIFOTH)

地址偏移: 0x40

复位值: 0x000F 000F

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

Bit	Name	R/W	Reset Value	Function
31:28	Reserved			
27:16	RXWMARK	RW	F	<p>当卡接收数据时, FIFO 的阈值水位线。当 FIFO 数据计数超过这个数目时, DMA/FIFO 请求被提出。当包结束时, 不管阈值编程如何, 都会产生请求, 以完成剩余的数据。</p> <p>在非 DMA 模式下, 当启用接收 FIFO 阈值中断(RXDR)时, 则产生中断而不是 DMA 请求。</p> <p>在数据包结束期间, 如果阈值编程大于任何剩余数据, 则不会产生中断。它是主机的责任读剩余字节在看到数据传输中断。</p> <p>在 DMA 模式中, 在数据包结束时, 即使剩余字节小于阈值, DMA 请求在设置“数据传输完成”中断之前进行单次传输以清除剩 0 余字节。</p>

Bit	Name	R/W	Reset Value	Function
15:12	Reserved			
11:0	TXWMARK	RW	F	<p>当传输数据到卡, FIFO 阀值水位线。当 FIFO 数据计数小于或等于这个数字时, DMA/FIFO 请求被提出。如果中断被启用, 则中断发生。在包的结束, 请求或中断产生, 无论阀值编程。</p> <p>在非 DMA 模式下, 当启用发送 FIFO 阀值(TXDR)中断时, 产生中断而不是 DMA 请求。在数据包结束时, 在上次中断时, 主机负责用只需要的剩余字节填充 FIFO(不是在 FIFO 满之前, 也不是在 CIU 完成数据传输之后, 因为 FIFO 可能不是空的)。</p> <p>在 DMA 模式中, 在数据包的末端, 如果最后的传输小于突发大小, DMA 控制器做单个周期, 直到所需的字节被传输。</p>

#### 22.4.15. SDIO 发送到卡计数器 (SDIO\_TCBCNT)

地址偏移: 0x44

复位值: 0x0000 0000

31:0
TRANSCARDBYTECOUNT
R

Bit	Name	R/W	Reset Value	Function
31:0	TRANSCARDBYTECOUNT	R	0	CIU 单元传输到卡上的字节数。

#### 22.4.16. SDIO 发送到 FIFO 计数器(SDIO\_TBBCNT)

地址偏移: 0x48

复位值: 0x0000 0000

31:0
trans_fifo_byte_count
R

Bit	Name	R/W	Reset Value	Function
31:0	trans_fifo_byte_count	R	0	HOST/DMA 内存和 BIU FIFO 之间传输的字节数。

#### 22.4.17. SDIO 数据 FIFO 寄存器 (SDIO\_FIFODATA)

地址偏移: 0x200

复位值: 0x0000 0000

接收和发送 FIFO 是 32 位的宽度读或写一组寄存器, 它在连续的 32 个地址上包含 32 个寄存器。注意数据 fifo 寄存器包括发送和接收 fifo, 因此要将这 32 个地址分为 16 个一组, 发送接收各占一半。而且在我们每次读写的时候, 最多的是读取接收 FIFO 或者写入发送 FIFO 的一般大小的数据。

CPU 可以使用 FIFO 读写多个操作数。

31:0
FIFODATA
RW

Bit	Name	R/W	Reset Value	Function
31: 0	FIFODATA	RW	0	FIFODATA: 接收或发送 FIFO 数据 (Receive and transmit FIFO data) FIFO host 不支持从一个端口进行同时读写。为了调试的目的，软件可能会尝试写入 FIFO 并读回数据，结果是不确定的，因为设计不支持从相同的端口读/写访问。

## 22.4.18. SDIO 寄存器映像

0x00	SDIO_POWER	Reserved												PWRCTRL						
	Read/Write													r w						
	Reset Value													0						
0x04	SDIO_CLKCR	Reserved												CLKDIV						
	Read/Write													rw						
	Reset Value													0						
0x08	SDIO_ARG	CMDARG																		
	Read/Write	rw																		
	Reset Value	0												0						
0x0C	SDIO_CMD	STARTCMD	USEHOLDREG RESERVED	BOOTMODE	BOOTDIS	BOOTACK	BOOTEN	IEN	ATACMD	REGSYNC	AUTINIT	ABORTCMD	WAITPEND	DTMOODE	DIR	DEXPECT	CHECKRESPCRC	RESPLN	WAITRESP	CMDINDEX
	Read/Write	r w		r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	r w	rw
	Reset Value	0 0 1 0		0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0x10	SDIO_RESPCMD	Reserved													RESPCMD					
	Read/Write														r					
	Reset Value														0					
0x14	SDIO_RESP0	SDIO_RESP1																		
	Read/Write	r																		
	Reset Value	0																		
0x18	SDIO_RESP1	SDIO_RESP2																		
	Read/Write	r																		
	Reset Value	0																		

	SDIO RESP2	SDIO_RESP3																																			
0x1C	Read/Write	r																																			
	Reset Value	0																																			
	SDIO RESP3	SDIO_RESP4																																			
0x20	Read/Write	r																																			
	Reset Value	0																																			
	SDIO_DTIME_R	DATATIME												RESPTIME																							
0x24	Read/Write	rw												rw																							
	Reset Value	0xFFFFFFF												0x40																							
	SDIO_BLK-SIZ	Reserved						DBLOCKSIZE																													
0x28	Read/Write							rw																													
	Reset Value	0						0x200																													
	SDIO_DLEN	DATALENGTH																																			
0x2C	Read/Write	rw																																			
	Reset Value	0x200																																			
	SDIO_CTRL	Reserved				ODPUEEN	Reserved				CEATAINTEN	Reserved				AUTOSTOPCSD	Reserved																				
0x30	Read/Write	rw					rw				r w	rw				CCSDEN	rw																				
	Reset Value	0				1	0				0	0				ABORTRD	rw																				
	Reset Value	0					0				0	0				AUTOIRQRESP	rw																				
0x34	SDIO_STATUS	Reserved		FIFOCNT				Reserved				CARDBSY	CMDFSM				READWAIT	rw																			
	Read/Write			r				r					r					r																			
	Reset Value	0	0	0				0				0	0				0	0																			
0x38	SDIO_INSTS	Reserved								SDIOINT	EBE	ACD	SBE	HLE	FRUN	HTO	DRTO_BDS	RTO_BAR	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	CAD											
	Read/Write									rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1	rc w 1											
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

0x 3 C		SDIO_I NTMA SK	Reserved												SDIOINTIE											
			Read/ Write			0			EBEIE	ACDIE	SBEIE	HLEIE	FRUNIE	HTOIE	DRTOIE	RTOIE	DCRCIE	RCRCIE	RXDRIE	TXDRIE	DTOIE	CDIE	REIE	CADIE		
			Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x 40	SDIO_ FIFOT H	Reserved	RXWMARK			rw	Reserved			TXWMARK												rw				
			rw							rw												rw				
		Reset Value	0	f	f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x 44	SDIO_ TCBC NT	TCBCNT																								
		r																								
		0																								
0x 48	SDIO_ TBBC NT	TBBCNT																								
		r																								
		0																								
0x 20 0	SDIO_ FIFO	FIFODATA																								
		rw																								
		0x00																								

## 23. USB 全速设备接口 (USB)

### 23.1. 简介

USBD 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。

USBD 外设支持 USB 挂起/唤醒操作，可以停止设备时钟实现低功耗。

### 23.2. 主要特征

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点(ENDPOINT0 ~ENDPOINT7)
- 使用一个专用的 512 字节数据存储器
- 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/唤醒操作
- 帧锁定时钟脉冲生成

### 23.3. 功能说明

#### 23.3.1. 模块框图

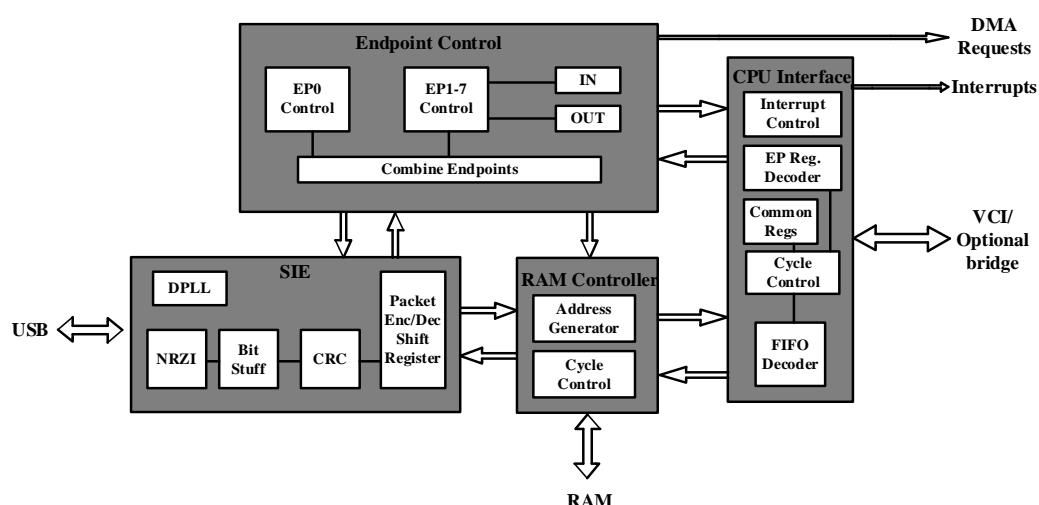


图 23-1 USB 模块框图

#### 23.3.2. 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合《USB 规范》的通信连接。PC 主机和微控制器之间的数据传输是通过共享一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小是固定的，每个端点最大可使用 64 字节缓冲区，端点 0 使用 64 字节缓冲区，其余的 IN 端点使用 64 字节缓冲区，OUT 端点使用 64 字节缓冲区，同一个端点的 IN 和 OUT 缓冲区的入口地址是一样的。USB 模块同 PC 主机通信，根据《USB 规范》实现令牌包(packets)的检测，数据发送/接收的处理，和握手包的处理。

当一有效的令牌包被 USB 模块识别时，相关的数据传输随之发生(如果需要传输数据，并且端点已配置)。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包。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务
- 产生如位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类型的传输。

在任何不需要使用 USB 模块的时候，可以通过写控制寄存器使 USB 模块置于低功耗模式(SUSPEND 模式)。在这种模式下，USB 内部的数据时钟会停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。

USB 模块使用固定的时钟，此时钟被 USB 标准定义为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

### 23.3.2.1. USB 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

**串行接口引擎(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 48MHz 时钟里产生一个 12MHz 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。它会为传输的包产生头文件(headers)并在接收到数据包时进行解码。

**端点控制器(ENDPOINT CONTROLLERS):**使用两个状态控制器，一个用于端点 0 上的控制传输，另一个用于端点 1—7 上的批量、中断或同步传输事务。

**CPU 接口(CPU INTERFACE):**CPU 接口允许访问每个端点的控制/状态寄存器和用作数据缓冲区的 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 模式唤醒时，它会产生一个中断给 CPU。

**RAM 控制器(RAM CONTROLLER):**RAM 控制器提供了一个接口，这个接口接到了同步单端口 RAM 的单个 block，这个 block 用于缓存 CPU 和 USB 之间的数据包。它从端点控制器获取 FIFO 指针，并将它们转换为 RAM block 内的地址指针并生成 RAM 访问控制信号。

### 23.3.3. 功能使用

#### 23.3.3.1. 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

- 首先，由应用程序激活寄存器单元的时钟；
- 其次，再配置设备时钟管理逻辑单元的相关控制位；
- 最后，清除复位信号。

当系统复位时，应用程序应该初始化所有需要的寄存器，使 USB 模块能够产生正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化(比如中断使能的选择等)。接下来，就进入 USB 复位状态。

#### 23.3.3.2. USB 复位(RESET 中断)

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。

当软件接收到一个复位中断时，它应该关闭所有打开的管道，并等待总线枚举开始

### 23.3.3.3. USB 挂起/唤醒模式

当 USB 设备在 3ms 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 suspend 中断被使能了(USB\_INTRE 寄存器中的 EN\_Suspend 位)，那么在此时会产生一个 suspend 中断。

当 USB 进入 suspend 模式后，内部 12MHz 的数据时钟将会被停止以此来降低功耗。名为 USB\_SUSPEND 的输出信号将会置 1，这可以被用来关闭 USB 的驱动程序。同时，输入的 48MHz 系统时钟需要一直保持，以便 USB 模块可以检测总线上的信号。当然输入的 48M 系统时钟也是可以停止从而进一步降低功耗，但是如果 48M 系统时钟停止了的话，USB 模块将无法检测到总线上的信号，用户将不得不重启系统时钟。

当 USB 模块处于 suspend 模式时，任何总线上的活动(非空闲信号)都可以把设备唤醒，从而退出 suspend 模式并产生一个 resume 中断。当然也可以通过 CPU 设置 USB 控制寄存器中的 Resume 位来强制设备离开 Suspend 模式，在这种情况下，CPU 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。

### 23.3.3.4. IN 分组(用于数据发送)

IN 端点 1-7 的 FIFO 大小都是固定的 64 字节，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 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\_INEPCSR 寄存器中的 InPktRdy 位。如果设置了 USB\_INEPCSR 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 CPU 设置)。

当 InPktRdy 位被设置时，USB\_INEPCSR 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，InPktRdy 位和 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。

### 双包缓存功能(DOUBLE PACKET BUFFERING)

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 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，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 InPktRdy 将再次设置(手动或自动)，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，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 FIFONotEmpty 位的状态表示可能加载了多少数据包。如果 FIFONotEmpty 是置位状态，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 FIFONotEmpty 位是清除状态，则表明 FIFO 中没有数据包，可以加载两个以上的数据包。

### 23.3.3.5. OUT 分组(用于数据接收)

OUT 端点 1 到 7 的 FIFO 大小都是固定 64 字节，但 OUT 传输的最大包大小是可编程的，由写入每个端点的 OutMaxP 寄存器决定。

当一个数据包被接收到并放置在 OUT FIFO 中时，USB\_OUTEPCSR 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 OutPktRdy 位，以便接收更多的数据包。如果设置了 USB\_OUTEPCSR 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，OutPktRdy 总是必须手动清除(即由 CPU 清除)。

### **双包缓存(DOUBLE PACKET BUFFERING)**

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 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\_OUTEPCSR 中的 FIFOFull 位在这一时间点上没有设置，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 OutPktRdy 位，以便接收更多的包。如果设置了 USB\_OUTEPCSR 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。对于小于最大包大小的包，OutPktRdy 总是必须手动清除(即由 CPU 清除)。

如果当 OutPktRdy 被清除时 FIFOFull 位被置为 1，USB 模块首先会清除 FIFOFull 位。然后再次设置 OutPktRdy 位，以表明在 FIFO 中有另一个数据包等待被读走。

#### **23.3.6. 控制传输**

端点 0 是 USB 的主控制端点。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。

软件可以通过端点 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。这些设备请求的协议涉及到每次传输的不同数量和类型的事务。为了适应这种情况，CPU 需要采用一种状态机方法对命令进行解码和处理。标准设备请求可以分为三类：零数据请求、写请求和读请求。本节将介绍软件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 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 USB 主机自动拒绝。

### **零数据请求(ZERO DATA REQUESTS)**

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。

与所有请求一样，当软件接收到一个端点 0 中断时，事件序列将开始。OutPktRdy 位(在 USB\_EP0CSR 寄存器中)也将被设置。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，则命令中包含的 7 位地址值应该写入 FAddr 寄存器。

然后应该写入 USB\_EP0CSR 寄存器来设置 ServicedOutPktRdy 位(表明该命令已经从 FIFO 读取)和设置 DataEnd 位(表明该请求不需要进一步的数据)。

当主机移动到请求的状态阶段时，将生成第二个端点 0 中断，以表明请求已经完成。软件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，应该编写 USB\_EP0CSR 寄存器来设置 ServicedOutPktRdy 位和设置 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。这时将生成第二个端点 0 中断，并设置 SentStall 位。

如果主机在设置 DataEnd 位之后发送更多的数据，那么设备将发送一个 STALL。这时将生成一个端点 0 的中断，并且 SentStall 位被置位。

### **写请求(WRITE REQUESTS)**

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。

与所有请求一样，当软件接收到一个端点 0 中断时，事件序列将开始。OutPktRdy 位(在 USB\_EP0CSR 寄存器中)也将被设置。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，USB\_EP0CSR 寄存器中的 ServicedOutPktRdy 位随后应该被设置(表明该命令已经从 FIFO 读取)，但在这种情况下不应该设置 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。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 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(由命令中的 wLength 字段指示)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，应该编写 USB\_EP0CSR 来设置 ServicedOutPktRdy 位，但不应该设置 DataEnd 位。

当接收到所有预期的数据包时，应该将 USB\_EP0CSR 寄存器中的 ServicedOutPktRdy 位和 DataEnd 位设置起来(表示不需要更多的数据)。

当主机移动到请求的状态阶段时，将生成另一个端点 0 中断，以表明请求已经完成。软件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 寄存器中的 ServicedOutPktRdy 位和 SendStall 位。当主机发送更多的数据时，USB 设备将发送一个 STALL 来告诉主机这个请求没有被执行。这时将生成一个端点 0 中断，并且 SentStall 位将会被设置。

如果主机在设置 DataEnd 后发送了更多的数据，那么 USB 设备将发送一个 STALL。一个端点 0 中断将会生成，并将 SentStall 位设置起来。

### **读请求(READ REQUESTS)**

在 8 字节的命令之后，读请求有一个(或多个)数据包从功能(function)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。

与所有请求一样，当软件接收到一个端点 0 中断时，事件序列将开始。OutPktRdy 位(在 USB\_EP0CSR 寄存器中)也将被设置。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 位设置(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 位被设置(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 位应该被设置(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，将生成另一个端点 0 中断，以表明请求已经完成。软件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，应该将 USB\_EP0CSR 寄存器中的 ServicedOutPktRdy 位和 SendStall 位设置。当主机请求数据时，USB 设备将发送一个 STALL 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 SentStall 位将被设置。

如果主机在设置 DataEnd 之后请求更多的数据，那么 USB 设备将发送一个 STALL。一个端点 0 中断将会产生，并且 SentStall 位将会被设置。

### **错误处理(ERROR HANDLING)**

在 USB 模块上的控制传输可能由于协议错误而中止，主机会提前终止传输，或者如果软件功能控制器(function controller software)希望终止传输(例如，因为它不能处理该命令)。USB 在以下情况下会自动检测协议错误，并向主机发送 STALL。

- 1) 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 DataEnd 位后主机发送 OUT 令牌时，将检测到此条件。
- 2) 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。当主机在设置了 USB\_EP0CSR 寄存器中的 DataEnd 位后发送 IN 令牌时，将检测到此条件。
- 3) 主机发送的 OUT 数据包中大于 MaxP 数据字节。
- 4) 主机在读请求的 STATUS 阶段发送一个非零长度的 DATA1 数据包。

当 USB 设备发送了 STALL 包后，它将 SentStall 位置位并产生一个中断。当软件接收到 SentStall 置位的的端点 0 中断时，它应该中止当前的传输，清除 SentStall 位，并返回到空闲状态。

如果主机在传输请求的所有数据传输完成之前进入状态阶段，或者在完成当前传输之前发送一个新的 SETUP 包，从而提前结束传输，那么 SetupEnd 位将被设置，并生成一个端点 0 中断。当软件接收到一个 SetupEnd 位被置位的端点 0 中断时，它应该中止当前传输，设置 ServicedSetupEnd 位，并返回到空闲状态。如果 OutPktRdy 位被设置，这表明主机已经发送了另一个 SEPUP 包，然后软件应该处理这个命令。

如果软件不能处理该命令或其他内部错误而希望终止当前传输时，它应该设置 SendStall 位。然后 USB 设备将发送一个 STALL 包给主机，SentStall 位置位并生成一个端点 0 中断。

### **同步传输**

USB 协议定义了一种全速的需要保持固定和精确的数据传输速率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 包（由端点传输方向确定包的类型）。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 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。

#### **23.3.3.7. 同步读传输(ISOCHRONOUS IN ENDPOINT)**

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 InMaxP 寄存器的值小于或等于分配给端点的 FIFO 大小的一半时，自动启用双包缓冲。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 中接收另一个包，就会生成一个 DMA 请求。这个特性可以用来允许外部 DMA 控制器在没有处理器干预的情况下将数据包加载到 FIFO 中。

#### ■ AutoSet 功能

当 AutoSet 功能被启用时，InPktRdy 位将在一个 InMaxP 字节的数据包加载到 FIFO 时被自动设置。

在使用一个同步 IN 端点之前，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 wMaxPacketSize 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\_INEPCSR 寄存器中部分位应该设置为如下所示。

表 23-1 USB\_INEPCSR 寄存器中部分位设置

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，将有很少的时间来重新加载 FIFO。由于这个原因，在同步 IN 端点中通常是需要双缓存的。

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 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，软件可以使用这个中断来将下一个数据包加载到 FIFO 中，并在 USB\_INEPCSR 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，这可能会导致 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 设备的外部 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，每帧就会生成一个 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)。这个 SOF\_PULSE 中断仍然可以用来设置 USB\_INEPCSR 中的 InPktRdy 位，并检查数据溢出/缺失。

启用双缓存的同步 IN 管道可能是一个问题的来源。双缓存要求数据包只有在载入帧后才能被传送。如果功能在主机建立管道之前至少一帧加载第一个数据包(然后开始发送 IN 令牌)，则没有问题。但是，如果主机在加载第一个包时已经开始发送 IN 令牌，则包可能在加载时的同一帧中传输，这取决于它是在接收到 IN 令牌之前还是之后加载的。这个潜在的问题可以通过在 USB\_CR 寄存器中设置 ISO Update 位来避免。当这个位设置为 1 时，任何加载到同步 IN 端点 FIFO 的数据包将不会被发送，直到收到下一个 SOF 包，从而确保数据包不会过早发送。

如果端点在收到 IN 令牌时 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。这表明该软件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。

如果软件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)。这取决于应用程序如何处理这个情况：它可以選擇通过在 USB\_INEPxCSR 寄存器中设置 FlushFIFO 位来刷新未发送的数据包，或者它可以選擇跳过当前的数据包。

### 同步写传输(ISOCHRONOUS OUT ENDPOINT)

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：

#### ■ 双包缓存功能

当写入 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。

- 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。这个特性可以用来允许外部 DMA 控制器在不需要处理器干预的情况下从 FIFO 读取数据包。

- AutoClear 功能

当 AutoClear 功能启用时，OutMaxP 字节的数据包从 FIFO 读取时，OutPktRdy 位将被自动清除。

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 OutMaxP 寄存器。这个值应该与端点的标准端点描述符的 wMaxPacketSize 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 23-2 USB\_OUTEPxCSR 寄存器中部分位设置

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 中有空间来接收包。主机将每帧发送一个包，但是帧内的时间可以变化。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 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，软件可以使用这个中断从 FIFO 读走数据包，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传输到一些外部硬件，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 设备的外部 SOF\_PULSE 信号来触发数据包的读取来完成。当收到一个 SOF 包时，每帧生成一次 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)。该中断仍然可以用来清除 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 寄存器中的溢出位将被设置。这表明该软件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 OutPktRdy 位和 DataError 位。如何处理这个错误条件由应用程序决定。

### 23.3.8. 批量传输

#### 批量读传输(BULK IN ENDPOINT)

Bulk IN 端点用于将不定期数据从功能控制器传输到主机。有三个可选特性可用于 Bulk IN 端点：

- 双包缓存功能

如果写入 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储存在 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。这个特性可以用来允许外部 DMA 控制器在没有处理器干预的情况下将数据包加载到 FIFO 中。

- AutoSet 功能

当 AutoSet 功能被启用时，InPktRdy 位将在一个 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，因为它避免了在批量传输期间加载单个数据包时需要处理器干预。

在使用 Bulk IN 端点之前，必须使用该端点的最大包大小(以字节为单位)来写入 InMaxP 寄存器。这个值应该与端点的标准端点描述符的 wMaxPacketSize 字段相同。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\_INEPCSR 寄存器应该设置为如下所示。

表 23-3 USB\_INEPCSR 寄存器中部分位设置

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	0	1	0/1	0

当第一次配置 Bulk IN 端点时，在端点 0 上执行 SET\_CONFIGURATION 或 SET\_INTERFACE 命令后，应该写入 USB\_INEPCSR 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 FIFONotEmpty 位表示)，它们应该通过设置 FlushFIFO 位来刷新。

注意:如果启用了双包缓存功能，可能需要连续设置该位两次。

当数据要通过 Bulk IN 管道传输时，需要将一个数据包加载到 FIFO 和置位 USB\_INEPxCSR 寄存器中的 InPktRdy 位。当数据包发送完毕后，InPktRdy 位将被清除，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，那么当第一个包被加载并且 InPktRdy 位被设置后，InPktRdy 位将立即被清除，并产生一个中断，以便第二个包可以被加载到 FIFO 中。软件应该以同样的方式运行，当它接收到一个中断时加载一个包，不管是否启用双包缓存。

数据包的大小不能超过 InMaxP 寄存器中指定的大小。当要传输一个大于 InMaxP 的数据块时，必须将其作为多个数据包发送。这些包的大小应该是 InMaxP 设置的值，除了最后一个包。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 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 InMaxP 的倍数，那么功能(function)就需要在所有数据发送完毕后发送一个空包。这是通过在接收到下一个中断时设置 InPktRdy 来完成的，而不需要将任何数据加载到 FIFO 中。

如果正在传输大批量数据，那么通过使用 DMA 可以避免调用中断服务程序来加载每个包。

如果软件想要关闭 Bulk IN 管道，它应该设置 SendStall 位。当 USB 设备接收到下一个 IN 令牌时，它将发送一个 STALL 给主机，设置 SentStall 位并产生一个中断。

当软件接收到一个设置了 SentStall 位的中断时，它应该清除这个 SentStall 位。当然，它也应该保留 SendStall 位的设置，直到准备好重新启用 Bulk IN 管道。

注意:如果主机由于某种原因无法接收到 STALL 包，它将发送另一个 IN 令牌，所以建议保持 SendStall 位的设置，直到软件准备好重新启用 Bulk IN 管道。当管道被重新启用时，应该通过在 USB\_INEPCSR 寄存器中设置 ClrDataTog 位来重启数据切换序列。

## 批量写传输(BULK OUT ENDPOINT)

Bulk OUT endpoint 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 Bulk OUT 端点：

- 双包缓存功能

如果写入 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存。启用时，FIFO 中最多可以存储两个数据包。

#### ■ 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。这个特性可以用来允许外部 DMA 控制器在不需要处理器干预的情况下从 FIFO 读取数据包。

#### ■ AutoSet 功能

当 AutoClear 功能启用时，OutMaxP 字节的数据包从 FIFO 卸载时，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 Bulk 传输期间读取单个数据包时需要的处理器的干预。

在使用 Bulk OUT 端点之前，必须使用端点的最大包大小(以字节为单位)来写入 OutMaxP 寄存器。这个值应该与端点的标准端点描述符的 wMaxPacketSize 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(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 23-4 USB\_OUTEPxCSR 寄存器中部分位设置

AutoClear	ISO	DMAEnab
0/1	0	0/1

当第一次配置 Bulk OUT 端点时，在端点 0 上的 SET\_CONFIGURATION 或 SET\_INTERFACE 命令之后，应该编写 USB\_OUTEPCSR 来设置 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 FlushFIFO 位被刷新。

注意:如果启用了双包缓存，可能需要连续设置该位两次。

当一个数据包被 Bulk OUT 端点接收时，OutPktRdy 位被设置并产生一个中断。软件应该读取端点的 OutCount 寄存器，以确定数据包的大小。应该先从 FIFO 读取数据包，然后清除 OutPktRdy 位。

数据包大小不应该超过 OutMaxP 寄存器中指定的大小(因为这应该是发送到主机的端点描述符的 wMaxPacketSize 字段中设置的值)。当一个大于 OutMaxP 的数据块要发送给功能时，它将作为多个数据包发送。所有包的大小都是 OutMaxP 设置的大小，除了最后一个包，它将包含剩余部分。软件可以使用应用程序特定的方法来确定块的总大小，从而确定何时接收到最后一个包。或者，当它收到一个小于 OutMaxP 大小的包时，它可能推断整个数据已经收到(如果数据块的总大小是 OutMaxP 的倍数，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包的数据。

如果软件想要关闭 Bulk OUT 管道，它应该设置 SendStall 位。当 USB 设备接收到下一个数据包时，它将发送一个 STALL 给主机，设置 SentStall 位并产生一个中断。

当软件接收到一个设置了 SentStall 位的中断时，它应该清除这个 SentStall 位。它也应该保留 SendStall 位的设置，直到它准备好重新启用 Bulk OUT 管道。

注意:如果主机由于某种原因未能接收到 STALL 包，它将发送另一个包，因此建议保留 SendStall 位，直到软件准备好重新启用 Bulk OUT 管道。当重新启用 Bulk OUT 管道时，应该通过在 USB\_OUTEPxCSR 寄存器中设置 ClrDataTog 位来重启数据切换序列。

### 23.3.9. 中断传输

#### 中断读传输(INTERRUPT IN ENDPOINT)

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 Bulk 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包中传输所有的数据。

中断 IN 端点还支持 Bulk IN 端点不支持的一个特性，即它们支持数据切换位的连续切换。这个特性是通过在 USB\_INEPCSR 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。

#### **中断写传输(INTERRUPT OUT ENDPOINT)**

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 Bulk 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包中传输所有的数据。

### **23.3.4. USB 寄存器描述**

注意：在对 USB 寄存器进行写操作时只能按照 byte 或者 word 操作，在对 USB 寄存器进行读操作时只能按照 byte 操作。

#### **23.3.4.1. USB 控制寄存器(USB\_CR)**

Address offset: 0x00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISO Update	Res		Reset	Resume	Suspend Mode	Enable Suspend	Update	ADD							
RW			R	RW	R	RW	R	RW							

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15	ISO Update	RW	0	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	Reserved			
11	Reset	R	0	USB 总线有复位信号时，该 Bit 置 1
10	Resume	RW	0	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	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	Suspend 功能使能
7	Update	R	0	当 Func Addr 写入时置 1；地址生效（在传输结束）后清零
6:0	ADD	RW	0	Function 地址

#### **23.3.4.2. USB 中断状态寄存器(USB\_INTR)**

Address offset: 0x04

Reset value: 0x00000000

31	30	29	28	27	26	25	2 4	23	22	21	20	19	18	17	16
Reserved								EP7I N	EP6I N	EP5I N	EP4I N	EP3I N	EP2I N	EP1I N	EP0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT						SOF	Re-set	Re-sume	Sus pend
R	R	R	R	R	R	R						R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23	EP7IN	R	0	IN Endpoint 7 中断
22	EP6IN	R	0	IN Endpoint 6 中断
21	EP5IN	R	0	IN Endpoint 5 中断
20	EP4IN	R	0	IN Endpoint 4 中断
19	EP3IN	R	0	IN Endpoint 3 中断
18	EP2IN	R	0	IN Endpoint 2 中断
17	EP1IN	R	0	IN Endpoint 1 中断
16	EP0	R	0	IN Endpoint 0 中断
15	EP7OUT	R	0	OUT Endpoint 7 中断
14	EP6OUT	R	0	OUT Endpoint 6 中断
13	EP5OUT	R	0	OUT Endpoint 5 中断
12	EP4OUT	R	0	OUT Endpoint 4 中断
11	EP3OUT	R	0	OUT Endpoint 3 中断
10	EP2OUT	R	0	OUT Endpoint 2 中断
9	EP1OUT	R	0	OUT Endpoint 1 中断
8:4	Reserved			
3	SOF	R	0	每帧开始时置 1
2	Reset	R	0	在 USB 总线上侦测到复位信号时置 1
1	Resume	R	0	USB 设备在 Suspend 模式时，在 USB 总线上侦测到 Resume 信号时置 1
0	Suspend	R	0	在 USB 总线上侦测到 Suspend 信号时置 1

### 23.3.4.3. USB 中断使能寄存器(USB\_INTRE)

Address offset: 0x08

Reset value: 0x00FFFE06

31	30	29	28	27	26	25	2 4	23	22	21	20	19	18	17	16
Reserved								EP7I NE	EP6I NE	EP5I NE	EP4I NE	EP3IN E	EP2I NE	EP1I NE	EP0E
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT	Reserved					EN_S OF	EN_ Re-set	EN_ Re-sume	EN_S uspend
RW	RW	RW	RW	RW	RW	RW						RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23	EP7INE	RW	1	IN Endpoint 7 中断使能
22	EP6INE	RW	1	IN Endpoint 6 中断使能
21	EP5INE	RW	1	IN Endpoint 5 中断使能

Bit	Name	R/W	Reset Value	Function
20	EP4INE	RW	1	IN Endpoint 4 中断使能
19	EP3INE	RW	1	IN Endpoint 3 中断使能
18	EP2INE	RW	1	IN Endpoint 2 中断使能
17	EP1INE	RW	1	IN Endpoint 1 中断使能
16	EP0E	RW	1	IN Endpoint 0 中断使能
15	EP7OUTE	RW	1	OUT Endpoint 7 中断使能
14	EP6OUTE	RW	1	OUT Endpoint 6 中断使能
13	EP5OUTE	RW	1	OUT Endpoint 5 中断使能
12	EP4OUTE	RW	1	OUT Endpoint 4 中断使能
11	EP3OUTE	RW	1	OUT Endpoint 3 中断使能
10	EP2OUTE	RW	1	OUT Endpoint 2 中断使能
9	EP1OUTE	RW	1	OUT Endpoint 1 中断使能
8:4	Reserved			
3	EN SOF	RW	0	SOF 中断使能
2	EN Reset	RW	1	Reset 中断使能
1	EN Resume	RW	1	Resume 中断使能
0	EN Suspend	RW	0	Suspend 中断使能

#### 23.3.4.4. USB\_FRAME

Address offset: 0x0C

Reset value: 0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:20	Reserved			
19:16	INDEX	RW	0	Endpoint 选择寄存器
15:11	Reserved			
10:0	FramNUM	RW	0	最后接收的 Fram Number

#### 23.3.4.5. USB 端点 0 状态控制寄存器(USB\_EP0CSR)

Address offset: 0x10

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															
COUNT0								Serviced Set-upEnd		Serviced OutPktRdy		SendStall		Set-upEnd	
R								W1		W1		R		W1	
								W1		R		W1		R	

Bit	Name	R/W	Reset Value	Function
31:15	Reserved			
14:8	COUNT0	R	0	EP0 接收到的数据长度, OutPktRdy 置 1 时读取有效
7	Serviced-SetupEnd	W1	0	软件写 1 清零 SetupEnd, 该位自动清零
6	ServicedOutPktRdy	W1	0	软件写 1 清零 OutPktRdy, 该位自动清零
5	SendStall	W1	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	W1	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	RW0	0	发送 STALL 握手后置 1, 由软件清零;
1	InPktRdy	RW1	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

### 23.3.4.6. USB\_INEPxCSR

Address offset: 0x14

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								INMAXP							
								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re-served	ClrData-Tog	Sent-Stall	Send-Stall	Flush-FIFO	UnderRun	FIFONotEmpty	InPktRdy	AU-TOSet	ISO	Mode	DMAE-enable	FrcData-Tog	Reserved		
	W1	RW0	RW	W1	R	RW0	RW	RW	RW	RW	RW	RW			

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23:16	INMAXP	RW	0	IN 端点最大的包; 最大包大小为该 8 位的值*8, 每个 IN Endpoint 都有一个, EP0 除外
15	Reserved			
14	ClrDataTog	W1	0	软件写 1, 复位 IN EP data toggle
13	SentStall	RW0	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; 当 stall 环境消失后, 软清零, 该位对 ISO 无效
11	Flush FIFO	W1	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	R	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	FIFONot Empty	RW0	0	IN FIFO 非空标志
8	InPktRdy	RW	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 Interrupt 传输
5	Mode	RW	0	1: IN Endpoint; 0: OUT Endpoint;
4	DMAEnable	RW	0	In Endpoint DMA 请求使能
3	FrcDataTog	RW	0	置 1 后, 强制翻转 tog 信号并清除 FIFO 中的数据包
2:0	Reserved			

### 23.3.4.7. USB\_OUTEPxCSR

Address offset: 0x18

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								OUTMAXP							
RW								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	DataError	Over-Run	FIFOFull	Out-PktRdy	AU-TO-Clear	ISO	DMAE-nable	DMA-Mode	Reserved			
W1	RW0	RW	W1	R	RW0	R	RW0	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23:16	OUTMAXP	RW	0	OUT 端点最大的包，最大包大小为该 8 位的值*8，每个 OUT Endpoint 都有一个，EP0 除外
15	ClrDataTog	W1	0	软件写 1 复位 EP 的 data toggle 至 0
14	SentStall	RW0	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	Flush FIFO	W1	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	RW0	0	OUT 包不能再写入 OUT FIFO，软件清零； 只有在 ISO 模式下有效
9	FIFO Full	R	0	OUT FIFO 满标志
8	OutPktRdy	RW	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；
5	DMAEnalbe	RW	0	DMA 使能信号
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	Reserved			

### 23.3.4.8. USB\_OUTCOUNT

Address offset: 0x1C

Reset value: 0x00000000

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

Bit	Name	R/W	Reset Value	Function
31:11	Reserved			
10:0	OUTCOUNT	R	0	接收到的数据长度，OutPktRdy 置 1 时读取有效

### 23.3.4.9. USB\_FIFODATA

Address offset: 0x20 ~ 0x3F

Reset value: 0x00000000

注意: FIFO 只能按 word 或者 byte 操作

FIFODATA(0~7,每个 EP 占用 4 个 Byte 地址)															
RW															
0															

### 23.3.4.10. USB 寄存器映像

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	ISO Update	Reserved	Reset	ADD				
0x000	USB_CR	Reserved																rw	rw	rw					
		r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	0								
		w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	0								
0x004	USB_INTR	Reserved				0	EP7IN	EP6IN	EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0IN	EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT				
		r	r	r	r	0	0	0	0	0	0	0	0	0	r	r	r	r	r	r	r				
		w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w				
0x008	USB_INTR	Reserved				0	EP7INE	EP6INE	EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0INE	EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT				
		r	r	r	r	w	w	w	w	w	w	w	w	w	r	r	r	r	r	r	r				
		w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w				
0x01C	USB_FRAM_E	Reserved				0	INDEX				Reserved				FramNUM				r						
		rw				0	rw				rw				0				0						
		rw				0	rw				rw				0				0						
0x010	USB_EP0CSR	Reserved										COUNT0				r				r					
		rw										0				0				0					
		rw										0				0				0					
0x014	USB_INEPXCSR	Reserved				INMAXP				rw				rw				Res.							
		rw				rw				rw				rw				Res.							
		rw				rw				rw				rw				Res.							

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
0x18	USB_OUTEPxCSR	Reserved								OUTMAXP								Res.								OUTCOUNT																											
Read/Write	rw									w	r	w	0	CnData-	SentStall	w	r	w	1	w	r	w	0	r	w	0	r	w	0	0	0	0	0	0																			
Reset Value	0	0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																				
0x1C	USB_OUTCOUNT	Reserved																OUTCOUNT																																			
Read/Write	r																	r																																			
Reset Value	0																	0																																			
0x20~0x3F	USB_FIFODATA	FIFODATA(0~7, 每个 EP 占用 4 个 Byte 地址)																																																			
Read/Write	rw																																																				
Reset Value	0																																																				

## 24. CAN 总线控制器

### 24.1. 简介

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 FD 控制器遵循 CAN 总线 CAN2.0(CAN2.0A、CAN2.0B) 和 CAN FD 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 FD 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 FD 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 可以理解为两个 3 级 FIFO，FIFO 完全由硬件控制。

CAN FD 总线控制器同时也可以支持时间触发 CAN 通信（Time-trigger communication）。

### 24.2. 主要特征

- 完全支持 CAN2.0A/CAN2.0B/CAN FD 协议。
- CAN2.0 支持最高通信波特率 1Mbit/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

### 24.3. 功能说明

### 24.3.1. 模块框图

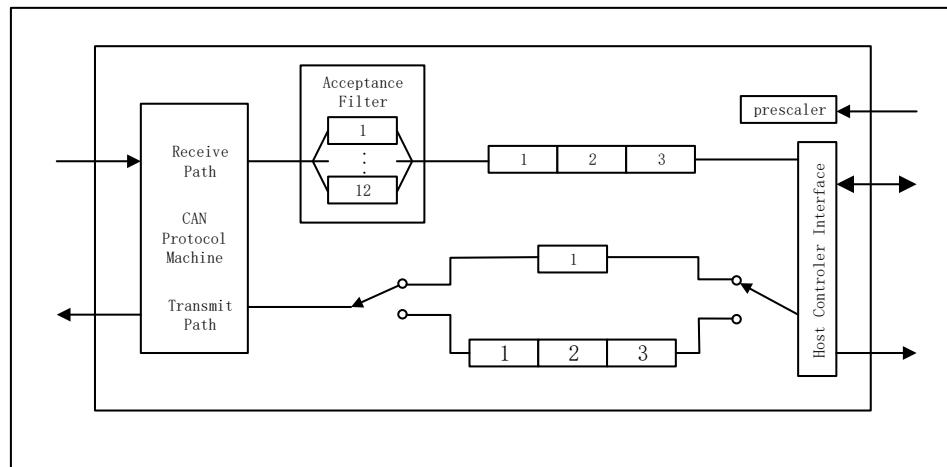


图 24-1 CAN FD 模块框图

### 24.3.2. 动作模式

CAN FD 控制器存在两个操作模式，复位模式 (CANFD\_MCR.RESET=1) 和动作模式 (CANFD\_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，然后退出复位模式，在动作模式中操作其余寄存器。

### 24.3.3. 波特率設定

CAN 通信使用时钟 can\_clk 的时钟源为外部高速振荡器，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 CANFD\_ACBTR 和 CANFD\_FDBTR 设定。寄存器 CANFD\_ACBTR 和 CANFD\_FDBTR 只能在 CANFD\_MCR.RESET=1 即 CAN 软件复位时设定。CANFD\_ACBTR 寄存器用于 CAN2.0 和 CAN FD 的仲裁段，CANFD\_FDBTR 寄存器用于 CAN FD 数据段。

FD 通信时，建议通信时钟选取 20MHz/40MHz/80MHz。

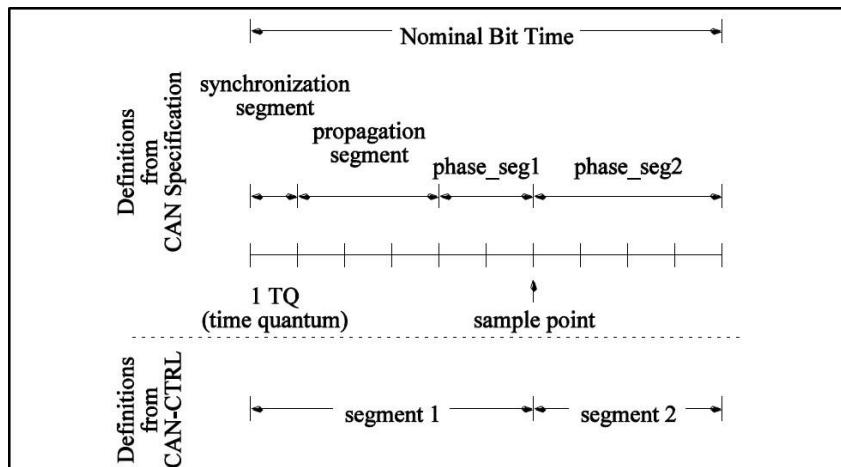


图 24-2 CAN 位时间定义

TQ 计算方法请参考以下公式，其中 PRESC 通过 CANFD\_RLSSP 的 PRESC 位设定。 $f_{can\_clk}$  为 CAN 通信时钟频率。

$$S\_TQ = \frac{S\_PRESC + 1}{f_{can\_clk}}$$

$$F\_TQ = \frac{F\_PRESC + 1}{f_{can\_clk}}$$

位时间计算方法请参考以下公式，其中 AC\_segment1 和 AC\_segment2 通过 CANFD\_ACBTR 寄存器的 AC\_SEG\_1 位和 AC\_SEG\_2 位设定, FD\_segment1 和 FD\_segment2 通过 CANFD\_FDBTR 寄存器的 FD\_SEG\_1 位和 FD\_SEG\_2 位设定。

$$\text{Slow_BT} = t_{S\_segment1} + t_{S\_segment2} = ((AC\_SEG\_1 + 2) + (AC + SEG\_2 + 1)) \times AC\_TQ$$

$$\text{Fast_BT} = t_{F\_segment1} + t_{F\_segment2} = ((FD\_SEG\_1 + 2) + (FD + SEG\_2 + 1)) \times FD\_TQ$$

表 24-1 CAN 时间设定规则

位	设定范围			规则
CANFD_ACBTR寄存器的AC_SEG_1位	[0..63]	CAN2.0 bits	(slow)	SEG_1 ≥ SEG_2 + 1
	[0..63]	CAN FD nominal bits	(slow)	SEG_2 ≥ SJW
CANFD_ACBTR寄存器的AC_SEG_2位	[0..7]	CAN2.0 bits	(slow)	
	[0..31]	CAN FD nominal bits	(slow)	
CANFD_ACBTR寄存器的AC_SJW位	[0..15]	CAN2.0 bits	(slow)	
	[0..15]	CAN FD nominal bits	(slow)	
CANFD_FDBTR寄存器的FD_SEG_1位	[0..15]	CAN FD data bits	(fast)	SEG_1 ≥ SEG_2
CANFD_FDBTR寄存器的FD_SEG_2位	[0..7]	CAN FD data bits	(fast)	SEG_2 ≥ SJW
CANFD_FDBTR寄存器的FD_SJW位	[0..7]	CAN FD data bits	(fast)	

以下给出 CANFD 的波特率设定推荐，仅供参考。

PSP：一级采样点

SSP：二级采样点

Seg 1：段 1

Seg 2：段 2

TDC：发送延时补偿

表 24-2 20MHz 通信时钟时波特率设定建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	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	禁止	1	40	32	8	8	-
1	80	80	1	20	16	4	4	16
2	80	80	1	10	8	2	2	8
4	80	80	1	5	4	1	1	4
5	75	75	1	4	3	1	1	3

表 24-3 40MHz 通信时钟时波特率设定建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	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	禁止	2	40	32	8	8	-
1	80	80	1	40	32	8	8	32
2	80	80	1	20	16	4	4	16
4	80	80	1	10	8	2	2	8
5	75	75	1	8	6	2	2	6
8	80	80	1	5	4	1	1	4

表 24-4 80MHz 通信时钟时波特率设定建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN 通信时钟]
0.25 (仲裁)	80	-	4	80	64	16	16	-
0.5 (仲裁)	80	-	2	80	64	16	16	-
0.5	80	禁止	4	40	32	8	8	-
1	80	80	2	40	32	8	8	64
2	80	80	2	20	16	4	4	32
4	80	80	1	20	16	4	4	16
5	75	75	1	16	12	4	4	12
8	80	80	1	10	8	2	2	8

#### 24.3.4. 发送缓冲器

CAN\_CTRL 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 CANFD\_MCR 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 CANFD\_MCR 寄存器的 TBSEL 位选择 PTB 或者 STB，TBSEL=0，选择 PTB，TBSEL=1，选择 STB。通过 CANFD\_MCR 寄存器的 TSNEXT 位选择 STB 中的下一个 SLOT。对应关系如下图所示：

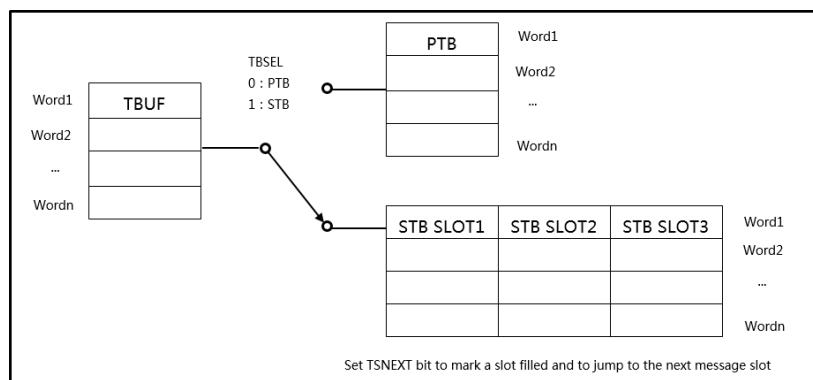


图 24-3 CAN FD TBUF 寄存器写发送缓冲器和示意图

#### 24.3.5. 接收缓冲器

CAN\_CTRL 提供 3 个 SLOT 的接收缓冲器用于存储接收到的数据，该 3 个 SLOT 的接收缓冲器工作在 FIFO 模式。RB SLOT 通过 CANFD\_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 CANFD\_MCR 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。

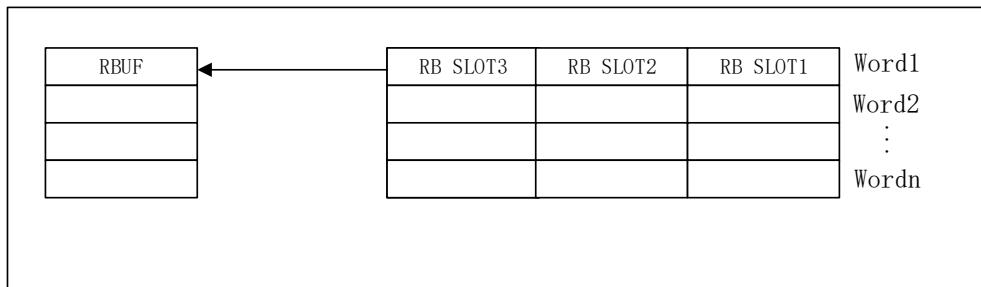


图 24-4 CAN RBUF 寄存器读接收缓冲器示意图

#### 24.3.6. 接收筛选寄存器组

CAN\_CTRL 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。每组筛选器有一个 32 位 ID CODE 寄存器和一个 32 位 ID MASK 寄存器，ID CODE 寄存

器用于比较接收到 CAN ID，而 ID MASK 寄存器用于选择比较的 CAN ID 位。对应的 ID MASK 位为 1 时，不比较该位的 ID CODE。

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 CANFD\_ACFCR 寄存器 AE\_n 寄存器使能或者禁止。ID CODE 和 ID MASK 通过 CANFD\_ACFCR 寄存器设定，筛选器地址通过 CANFD\_ACFCR 寄存器的 ACFADR 位选择。ID CODE 和 ID MASK 通过 CANFD\_ACFC 寄存器和 CANFD\_ACFM 寄存器访问且只能在 CANFD\_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。

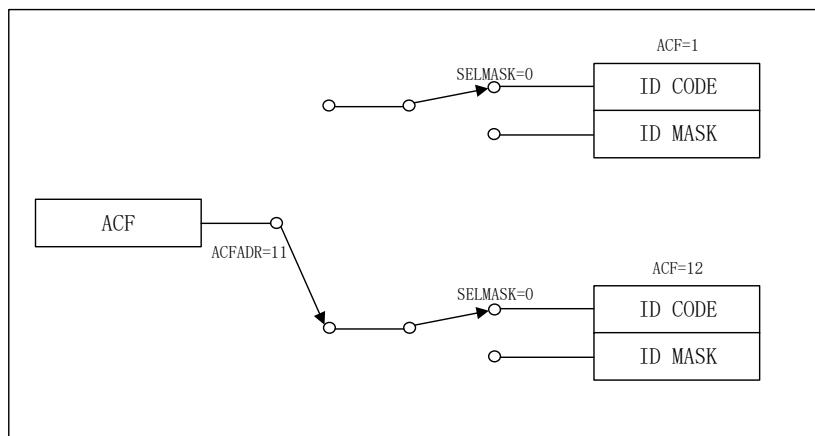


图 24-5 CAN ACF 寄存器访问筛选器组示意图

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 CANFD\_TBUF 中）、接收帧（从 CANFD\_RBUF 读取）和配置接收过滤器（CANFD\_ACFC 和 CANFD\_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
		Res.	Res.	Res.	LBF	ESI	KOER[2:0]			Res.	Res.	SEC	RMF	XLF	BRS	FDF	IDE
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Res.	Res.	Res.	Res.	Res.	DLC[10:0]										
0x08	CAN_TYPE	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HANDLE[7:0]								Res.							
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0c	CAN_AF	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		AF[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		AF[15:0]															
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.

偏移地址	寄存器																
0x1c	CAN_DATA1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CYCLE_TIME[15:0]															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Data[31:16]															
0x20	CAN_DATA16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Data[15:0]															
		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（表 3-42）。</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>Identifier Extension</p> <p>0 – Standard Format: ID(28:18)</p> <p>1 – Extended Format: ID(28:0)</p>
FDF	该位应当设置为 0 从而支持 CAN 2.0 frame
BRS	<p>CAN FD 比特率开关使能</p> <p>0 – 完整帧的标称/慢比特率</p> <p>1 – 为数据有效负载和 CAN FD 帧的 CRC 切换到数据/快速比特率</p> <p>BRS 对经典 CAN 没有意义。</p>
RMF	<p>远程帧</p> <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（第 4.10.4 章），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>
ESI	<p>错误状态指示器</p> <p>协议机器自动将 ESI 的正确值嵌入到传输的帧中。</p> <p>0 – CAN 节点处于错误状态</p> <p>1 – CAN 节点错误被动</p> <p>ESI 仅包含在 CAN FD 帧中，因此对于接收的经典 CAN 2.0B 始终为 0。</p> <p>传输的错误状态由寄存器 INTF 中的位 EPASS 显示。</p>
LBF	<p>环回帧</p> <p>如果激活了环回模式并且 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>

位	描述
	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。
CYCLE_TIME	周期时间 (TTCAN 的时间戳) 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。
AF	接受域 建议主要用于帧接受筛选，保留位全写 0xFFFFFFFF。
Payload Data	帧的有效载荷数据。 经典 CAN 2.0B 最多 8 个字节，CANFD 最多 64 个字节。

#### 24.3.7. 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 CANFD\_MCR 寄存器的 TSSTAT 位确认。发送数据设定步骤如下：

- 设定TBSEL 从PTB 和STB 中选择发送BUF
- 通过CANFD\_TBUF寄存器写需要发送的数据。
- 如果选择的是STB，设置TSNEXT=1 以完成STB SLOT 的装载。
- 发送使能
  - PTB 发送使用TPE
  - STB 发送使用TSALL 或者 TSONE
- 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

#### 24.3.8. 单次数据发送

不需要使用自动重新发送功能时，可以通过寄存器设定为单次发送模式，CANFD\_MCR 寄存器的 TPE 位用于 PTB 的单次发送模式设定，TSONE 位用于 STB 的单次发送模式。数据成功发送时单次发送和正常发送模式时动作相同。但是数据没有成功发送时会出现以下结果：

- TPIF 置位 (TPIE=1) ，对应的 BUF SLOT 数据会被清除。
- 有错误发送时，KOER 更新，BEIF 置位 (BEIE=1) 。
- 仲裁失败，ALIF 置位 (ALIE=1)。
- 单次发送模式，不能单独依靠 TPIF 来判断发送完成，需要同 BEIF 和 ALIF 一起判断发送是否完成。

### 24.3.9. 取消数据发送

可以通过 CANFD\_MCR 寄存器的 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：

- 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。
- 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。
  - TPA 释放 PTB，且使 TPE=0。
  - 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。

### 24.3.10. 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。

### 24.3.11. 错误处理

CAN\_CTRL 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\_CTRL 提供 CANFD\_WECR 寄存器的 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 CANFD\_WECR 寄存器的 EWL 位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

#### 24.3.12. 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从而不参与 CAN 通信，直到返回到错误主动状态。可以通过 CANFD\_MCR 寄存器的 BUSOFF 位确认 CAN 节点关闭状态。BUSOFF 被置位的同时 CANFD\_IFR 寄存器的 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 位的隐形位序列（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。

#### 24.3.13. 仲裁失败位置捕捉

CAN\_CTRL 能够精确捕捉到仲裁失败位的位置并反映到 CANFD\_WECR 寄存器的 ALC 寄存器中。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 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。

#### 24.3.14. 回环模式

CAN\_CTRL 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 CANFD\_MCR 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\_CTRL。

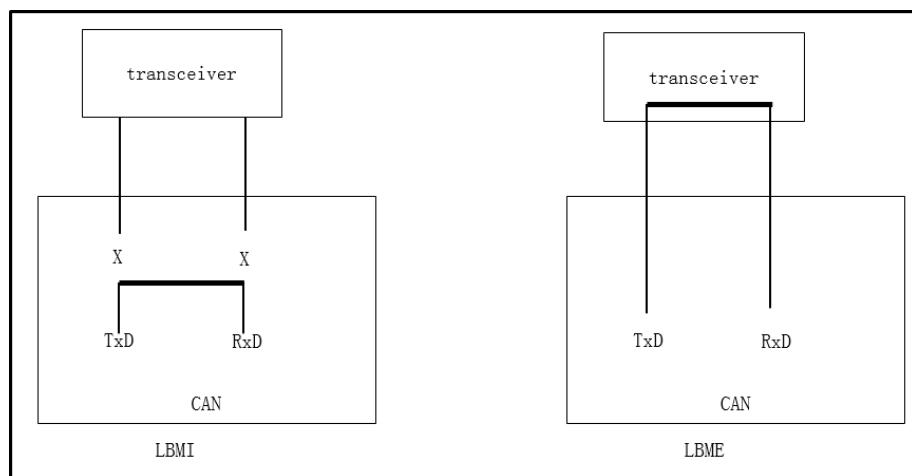


图 24-6 CANFD 内部回环 LBMI 和外部回环 LBME 示意图

### 24.3.15. 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 CANFD\_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。

外部回环模式可以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。

### 24.3.16. 软件复位功能

通过设定 CANFD\_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 24-5 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	-	FD_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	FD_SEG_2	是	只能在软件复位时可写
AE_x	否	-	FD_SJW	是	只能在软件复位时可写

寄存器位名	软件复位	备注	寄存器位名	软件复位	备注
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	-
BEIE	否	-	RBALL	是	-
BEIF	是	-	CANFD_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	-
FD_ISO	否	只能在软件复位时可写	ROIF	是	-
			ROM	否	
ROV	是	-	TSMODE	否	
RREL	是	-	TSNEXT	是	-
RSTAT	是		TSONE	是	-
SACK	是	-	TPIE	否	-
SELMASK	否	-	TPIF	是	-
PRESC	否	只能在软件复位时可以写	TPSS	是	-
AC_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
AC_SEG_2	否	只能在软件复位时可以写	TSIE	否	-
AC_SJW	否	只能在软件复位时可以写	TSIF	是	-

寄存器位名	软件复位	备注	寄存器位名	软件复位	备注
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	-	TTTYPE	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	-			

### 24.3.17. 兼容 CAN-FD 功能

CAN-CTRL 在 CAN FD 功能禁止时即使在包含 CAN FD 网络中收到 CAN FD 的帧，接收器会自动忽略这些帧，不返回 ACK，等到总线空闲时，再发送或者接收下一个 CAN2.0B 的帧。

### 24.3.18. 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(lever 1)硬件支持。本章节从以下 5 个部分介绍 TTCAN 功能。

#### 24.3.18.1. 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 CANFD\_TTCR 寄存器 TBPTR 位指定发送 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 CANFD\_TTCR 寄存器的 TBE 和 TBF 位来标记发送 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 CANFD\_IFR 寄存器的 TSIF 位。

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。

#### **24.3.18.2. 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 CANFD\_TTCR 寄存器的 TTPTTR 位指定的 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。

#### **24.3.18.3. TTCAN 时序**

CAN\_CTRL 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。该时间用作为接收消息的时间戳或者发送消息的触发时间基准。

#### **24.3.18.4. TTCAN 触发方式**

通过 CANFD\_TTCR 寄存器的 TTTYPE 位定义 TTCAN 的触发方式，TTPTTR 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，TTPTTR 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\_TRIGGER 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。此时，忽略 CANFD\_RLSSP 寄存器的 RELIM 位。

通过 CANFD\_TTCR 寄存器的 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\_TRIGGER 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。RELIM 用于决定是否自动重发或者单次发送模式。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\_TRIGGER 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 TPF 就可以再次发送。

如果 TT\_TRIGGER 值小于实际的 cycle time 则 TEIF 置位且执行停止。

### 24.3.18.5. 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。

### 24.3.19. TDC 和 RDC

CANFD 通信时，数据的延迟可能超过一位时间，因此需要补偿。TDC (Transmitter Delay Compensation) 和 RDC(Receiver Delay Compensation)用于数据延迟的补偿。其中 TDC 需要软件控制开关，而 RDC 不需要，自动有效。TDC 中采用辅助采样点 SSP (Secondary Sample Point) 的方式去补偿数据延迟。如下图：

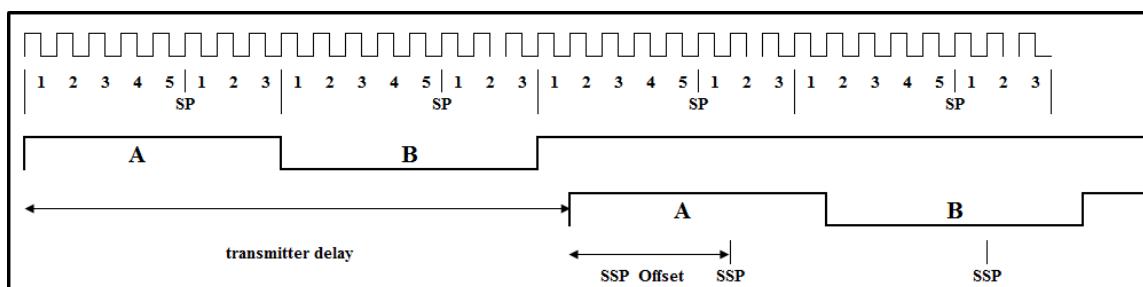


图 24-7 TDC 功能示意图

软件使能 TDC 功能时，本控制器可以自动确定 transmitter delay，通过设定 CANFD\_RLSSP 寄存器的 SSPOFF 来设定 SSP Offset。建议 SSP Offset 设定值和  $tF\_segment1$  设定值相同。

### 24.3.20. 中断

符号	中断标志	描述
CAN_HOST	RIF	接收中断
	ROIF	接收上溢中断
	ROIF	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

## 24.4. 寄存器说明

### 24.4.1. 节点配置寄存器 (CANFD\_TSNCR)

Address offset: 0x00

Reset value: 0x02010801

31	30	29	28	27	26	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	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
25	Res	-	-	-
24	Res	-	-	-
18	Res	-	-	-
17	ROP	rw	0	限制操作 0 - 限制操作不使能

Bit	Name	R/W	Reset Value	Function
				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，则无法开始传输。
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 和 CAN FD）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		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。

#### 24.4.2. 位时序配置寄存器 (CANFD\_ACBTR)

Address offset: 0x04

Reset value: 0x05050008

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.	Res.	AC_SEG_1[8:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
30:24	AC_SJW	rw	0x5	同步跳转宽度 同步跳转宽度 $tSJW = (x_{SJW} + 1) \cdot TQ$ 是缩短或延长重新同步位时间的最大时间。
22:16	AC_SEG_2	rw	0x5	位时序段 2 时间 $tSEG_2 = (x_{SEG\_2} + 1) \cdot TQ$ 从采样点到位结束
8:0	AC_SEG_1	rw	0x8	位时序段 1 采样点将在位时间开始后设置为 $tSEG_1 = (x_{SEG\_1} + 2) \cdot TQ$ 。

#### 24.4.3. CANFD\_FDBTR

Address offset: 0x08

Reset value: 0x02020003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD_SJW[6:0]							Res.	FD_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.	Res.	FD_SEG_1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
30:24	FD_SJW	rw	0x2	Synchronization Jump Width The Synchronization Jump Width $tSJW = (x\_SJW + 1) \cdot TQ$ is the maximum time for shortening or lengthening the Bit Time for resynchronization.
22:16	FD_SEG_2	rw	0x2	Bit Timing Segment 2 Time $tSEG_2 = (x\_SEG_2 + 1) \cdot TQ$ after the sample point to the end of the bit.
7:0	FD_SEG_1	rw	0x3	Bit Timing Segment 1 The sample point will be set to $tSEG_1 = (x\_SEG_1 + 2) \cdot TQ$ after start of bit time.

#### 24.4.4. 限制与预分频配置寄存器 (CANFD\_RLSSP)

Address offset: 0x10

Reset value: 0x77000000

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

Bit	Name	R/W	Reset Value	Function
30:28	RETLIM	rw	0x111	111 – 无限制 (仅受发送错误计数器 TECNT 限制) 110 – 7 次尝试 ... 000 – 1 次尝试 (不重传) 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。 RETLIM 可以随时更新, 但在更新后, 寄存器的内容需要使用时钟域交叉传输到 CAN 协议机器。在这很短的时间内 (几个时钟), RETLIM 被写锁定。
26:24	REALIM	rw	0x111	0x111 重新仲裁限制 111 – 无限制 110 – 7 次尝试 ... 000 – 1 次尝试 (无重新仲裁) 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。 REALIM 可以随时更新, 但在更新后, 寄存器的内容需要使用时钟域交叉传输到 CAN 协议机器。在这很短的时间内 (几个时钟), REALIM 被写锁定。

Bit	Name	R/W	Reset Value	Function
23:16	Res	-	-	-
15:8	FD_SSPOFF	rw	0x0	<p>二次采样点偏移 如果 SSPOFF<math>\neq</math>0 且 CES=1，则启用发送器延迟补偿 (TDC)。如果 BRS 处于活动状态，则 TDC 将在 CAN FD 帧的数据阶段或 CAN XL 帧的数据阶段被激活。有关 TDC 的更多详细信息，请参阅功能描述。 发射器延迟加上 SSPOFF 定义了 TDC 的二次采样点的时间。SSPOFF 以 TQ 的数量给出。 对于 CAN FD 帧，FD_SSPOFF 定义 SSPOFF。</p>
4:0	PRESC	rw	0x0	<p>预分频器 预分频器对系统时钟进行分频以合成时间量子 TQ。<math>nprescaler = PRESC + 1</math> and <math>tTQ = nprescaler / fcan\_clk</math></p>

#### 24.4.5. 状态寄存器 (CANFD\_IFR)

Address offset: 0x14

Reset value: 0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EWARN	EPASS	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	RAIF	TPIF	TSIF	EIF	AlF
		rw													

Bit	Name	R/W	Reset Value	Function
31	EWARN	r	0x0	<p>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</p>
30	EPASS	r	0x0	<p>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</p>
18	Res	-	-	-
17	Res	-	-	-
16	Res	-	-	-
15	Res	-	-	-
14	Res	-	-	-
13	WTIF	rw	0x0	<p>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。</p>
12	TEIF	rw	0x0	<p>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</p>
11	TTIF	rw	0x0	TTCAN: 时间触发中断标志

Bit	Name	R/W	Reset Value	Function
				如果设置了 TTIE 并且循环时间等于触发时间 TT_TRIG，则将设置 TTIF。向 TTIF 写入 1 会重置它。写一个零没有影响。TTIF 将只设置一次。如果 TT_TRIG 没有更新，则在下一个基本周期中不会再次设置 TTIF。
10	EPIF	rw	0x0	错误被动中断标志。 如果 EPASS 改变并且 EPIE=1，EPIF 将被激活。
9	ALIF	rw	0x0	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 0。
8	BEIF	rw	0x0	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 0。
7	RIF	rw	0x0	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 0。
6	ROIF	rw	0x0	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 上溢时 ROIF 和 RFIF 同时置 1。通过应用程序写 1 清 0。
5	RFIF	rw	0x0	接收 BUF 满中断标志 (RB Full Interrupt Flag) 0: RB FIFO 未满 1: RB FIFO 满 通过应用程序写 1 清 0。
4	RAFIF	rw	0x0	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 0。
3	TPIF	rw	0x0	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 0。 注意：TTCAN 模式时，TPIF 无效，仅适用 TSIF 标志
2	TSIF	rw	0x0	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 0。 注意：TTCAN 模式时，TPIF 无效，仅使用 TSIF 标志
1	EIF	rw	0x0	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。

Bit	Name	R/W	Reset Value	Function
				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 0。
0	AIF	rw	0x0	取消发送中断标志 (Abort Interrupt Flag) 0: 未取消发送数据 1: 通过 TPA 和 TSA 请求的发送消息被成功取消。 通过应用程序写 1 清 0。

#### 24.4.6. 中断使能寄存器 (CANFD\_IER)

Address offset: 0x18

Reset value: 0x0000468fe

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WTIE	Res.	TTIE	EPIE	ALIE	BEIE	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	Res.
		rw		rw	rw	rw	rw								

Bit	Name	R/W	Reset Value	Function
18	Res	-	-	-
14	Res	-	-	-
13	WTIE	rw	0x1	触发看门中断使能 (Watch Trigger Interrupt Enable) 0: 禁止 1: 使能
11	TTIE	rw	0x1	时间触发中断使能 (Time Trigger Interrupt Enable) 0: 禁止 1: 使能
10	EPIE	rw	0x1	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	0x1	仲裁失败中断使能 (Arbitration Lost Interrupt Enable) 0: 禁止 1: 使能
8	BEIE	rw	0x1	总线错误中断使能 (Bus Error Interrupt Enable) 0: 禁止 1: 使能

Bit	Name	R/W	Reset Value	Function
7	RIE	rw	0x1	接收中断使能 (Receive Interrupt Enable) 0: 禁止 1: 使能
6	ROIE	rw	0x1	接收上溢中断使能 (Receive Overrun Interrupt Enable) 0: 禁止 1: 使能
5	RFIE	rw	0x1	接收 BUF 满中断使能 (RB Full Interrupt Enable) 0: 禁止 1: 使能
4	RAFIE	rw	0x1	接收 BUF 将满中断使能 (RB Almost Full Interrupt Enable) 0: 禁止 1: 使能
3	TPIE	rw	0x1	PTB 发送中断使能 (Transmission Primary Interrupt Enable) 0: 禁止 1: 使能
2	TSIE	rw	0x1	STB 发送中断使能 (Transmission Secondary Interrupt Enable) 0: 禁止 1: 使能
1	EIE	rw	0x1	错误中断使能 (Error Interrupt Enable) 0: 禁止 1: 使能

#### 24.4.7. 传输状态寄存器 (CANFD\_TSR)

Address offset: 0x1c

Reset value: 0x0

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
26:24	TSTAT_H	rw	0	传输状态码
23:16	HANDLE_H	rw	0	帧识别 handle
10:8	TSTAT_L	rw	0	传输状态码
7:0	HANDLE_L	rw	0	帧识别 handle

#### 24.4.8. 全局配置寄存器 (CANFD\_MCR)

Address offset: 0x28

Reset value: 0x00900080

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]		FD_ISO	TSNEXT	TSMODE	TTBFM	Res.	TSFF	TSSTAT[1:0]	
rw	rw	r	rw	rw		r	r	rw	rw	rw	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	RESET	LBME	LBMI	Res.	Res.	Res.	Res.	BUSOFF
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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 - 几乎满 (AFWL 的可编程阈值) 但未满且无溢出 11 - 满 (溢出时保持设置 - 溢出信号参见 ROV)
23	FD_ISO	rw	1	CAN FD ISO 模式 0 - Bosch CAN FD (非 ISO) 模式 1 - ISO CAN FD 模式 (ISO 11898-1:2015) ISO CAN FD 模式具有不同的 CRC 初始化值和额外的填充位计数。 两种模式不兼容, 不得在一个 CAN 网络中混合使用。 该位对 CAN 2.0B 没有影响。 该位仅在 RESET=1 时可写。

Bit	Name	R/W	Reset Value	Function
22	TSNEXT	rw	0	<p>下一个 STB (Transmit buffer Secondary NEXT)</p> <p>0: 无动作</p> <p>1: 当前 STB SLOT 已填充, 指向下一个 SLOT</p> <p>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。</p> <p>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。</p> <p>所有的 STB SLOT 被填满后, TSNEXT 保持为 1 直到有 STB SLOT 被释放。</p> <p>注意: TTCAN 模式时此位固定为 0。</p>
21	TSMODE	rw	0	<p>STB 发送模式 (Transmit buffer Secondary operation MODE)</p> <p>0: FIFO 模式</p> <p>1: 优先级模式</p> <p>FIFO 模式根据数据帧写入的先后顺序发送。</p> <p>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。</p> <p>注意: TSMODE 位只能在 STB 空时设定。</p>
20	TTTBM	rw	1	<p>TTCAN BUF 模式 (TTCAN Transmit Buffer Mode) TTEN=0 时, TTTBM 被忽略。</p> <p>0: TSMODE 决定, PTB 和 STB</p> <p>1: 通过 TBPTR 和 TTPTR 设定</p> <p>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。</p> <p>注意: TSMODE 位只能在 STB 空时设定。</p>
18	TSFF	r	0	<p>TTEN=0 or TTTBM=0: STB 满标志 (Transmit Secondary buffer Full Flag)</p> <p>0: STB SLOT 没有被全部填充</p> <p>1: STB SLOT 被全部填充</p> <p>TTEN=1 and TTTBM=1: TB 满标志 (Transmit buffer Full Flag)</p> <p>0: TBPTR 选择的发送 BUF 没有被全部填充</p> <p>1: TBPTR 选择的发送 BUF 被全部填充</p>
17:16	TSSTAT	r	0	<p>STB 状态 (Transmission Secondary Status bits)</p> <p>TTEN=0 或 TTEN=1 &amp; TTTBM=0</p> <p>00: STB 空</p> <p>01: STB 小于等于半满</p> <p>10: STB 大于半满</p> <p>11: STB 满</p> <p>TTEN=1 且 TTTBM=1</p> <p>00: PTB 和 STB 空</p> <p>01: PTB 和 STB 非满</p>

Bit	Name	R/W	Reset Value	Function
				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 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 ROP=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 该位必须由主机控制器设置并由 CAN-CTRL 复位。设置 TPA 自动取消置位 TPE。主机控制器可以将 TPA 设置为 1, 但不能将其重置为 0。在 CAN-CTRL 重置该位的短时间内, 主机不能设置它。该位将重置为 如果 RESET=1 或 (TTEN=1 和 TTTBM=1), 则为硬件复位值。TPA 不应与 TPE 同时设置。
10	TSONE	rw	0	发送一帧 STB 数据设定位 (Transmit Secondary ONE frame)

Bit	Name	R/W	Reset Value	Function
				<p>0: 不发送 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 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 BUOFF=1 LOM=1&amp;LBME=0 TTEN=1&amp;TTTBM=1 ROP=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 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 BUSOFF=1 LOM=1&amp;LBME=0 TTEN=1&amp;TTTBM=1</p> <p>如果在传输期间 STB 加载了新帧, 则新帧也将被传输。</p>
8	TSA	rw	0	<p>STB 发送取消位 (Transmit Secondary Abort)</p> <p>0: 不取消 1: 取消已经通过 TSONE 或者 TSALL 置 1 请求但还未开始的 STB 发送 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 BUSOFF=1</p> <p>TSA 不应与 TSONE 或 TSALL 同时设置。</p>
7	RESET	rw	1	<p>复位请求位</p> <p>0: 不请求局部复位 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 RESET 置位 1。请注意, 当 RESET=0 后需要 11 个 CAN bit times 该节点才能参与通信。</p>
6	LBME	rw	0	<p>外部回环模式使能位</p> <p>0: 禁止外部回环模式 1: 使能外部回环模式</p> <p>注意: 通信中禁止设定该位。</p>
5	LBMI	rw	0	<p>内部回环模式使能位</p> <p>0: 禁止内部回环模式 1: 使能内部回环模式</p>

Bit	Name	R/W	Reset Value	Function
				注意：通信中禁止设定该位。
0	BUSOFF	rw	0	总线关闭 1 - 控制器状态为“总线关闭”。 0 - 控制器状态为“总线开启”。 向 BUSOFF 写入 1 将复位 TECNT 和 RECNT。这应该只用于调试。

#### 24.4.9. 错误警告寄存器 (CANFD\_WECR)

Address offset: 0x2c

Reset value: 0x00000001b

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 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。

Bit	Name	R/W	Reset Value	Function											
3:0	EWL	rw	0xb	Error Waring 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。											

#### 24.4.10. 参考 ID 寄存器 (CANFD\_REFMSG)

Address offset: 0x30

Reset value: 0x0

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

#### 24.4.11. TTCAN 配置寄存器 (CANFD\_TTCR)

Address offset: 0x34

Reset value: 0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	T_PRES_C[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]				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
26:25	T_PRESC	rw	0	<p>TTCAN 定时器 PREScaler 00b - 1 01b - 2 10b - 4 11b - 8</p> <p>TTCAN 时基是由 PRESC、AC_SEG_1 和 AC_SEG_2 定义的 CAN 位时间。</p> <p>使用 T_PRESC 定义了额外的预缩放因子 1、2、4 或 8。</p> <p>T_PRESC 只能在 TTEN=0 时修改，但可以修改 T_PRESC 并同时设置 TTEN 与一次写访问。</p>
24	TTEN	rw	0	<p>TTCAN 使能 (Time Trigger Enable) 0: 禁止 1: 使能 TTCAN, 计数器开始计数。</p>
23	TBE	rw	0	<p>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。</p>
22	TBF	rw	0	<p>设置 TB 已填充 (set TB slot to "Filled") 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。</p>
21:16	TB PTR	rw	0x00	<p>TB SLOT 指针 (Pointer to a TB message slot) 0x00 - 指向 PTB 的指针 其他 - 指向 STB 中的插槽的指针 TB PTR 指向的帧槽可以使用 TBUF 寄存器读取/写入。 只有在 TSFF=0 时才能进行写访问。 将 TBF 设置为 1 将所选插槽标记为已填充, 将 TBE 设置为 1 将所选插槽标记为空。 TBSEL 和 TSNEXT 在 TTCAN 模式下没有使用, 没有任何意义。 TB PTR 只能指向存在于硬件中的缓冲槽。 TB PTR 的不可用位固定为 0。 TB PTR 仅限于 PTB 和 63 个 STB 插槽。 TTCAN 模式下不能使用更多插槽。 如果 TB PTR 太大并指向一个不可用的插槽, 则 TBF 和 TBE 将自动重置并且不执行任何操作。</p>
15:12	TEW	rw	0	<p>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。</p>
10:8	TTYPE	rw	0	<p>触发类型 (Trigger Type) 000: 立即触发 (Immediate Trigger for immediate transmission)</p>

Bit	Name	R/W	Reset Value	Function
				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。
5:0	TTPTR	rw	00	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 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。

#### 24.4.12. TTCAN 触发寄存器 (CANFD\_TTTR)

Address offset: 0xfffff0000

Reset value: 0x02020003

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	观察触发时间 TT_WTRIG(15:0) 定义手表触发的周期时间。 初始监视触发是最大周期时间 0xffff。
15:0	TT_TRIG	rw	0	触发时间 TT_TRIG(15:0) 定义触发的周期时间。 对于传输触发, 相应帧的 SOF 的最早传输点将是 TT_TRIG+1。

#### 24.4.13. CANFD\_SCMS

Address offset: 0x3c

Reset value: 0x0

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

Res.	Res.	Res.	HELOC[2:0]				TXB	TXS	ACFA	Res.	Res.							
			r	r	r	r	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTIM[2:0]	Res.

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	-
28:27	HELOC	r	0	<p>主机端内存错误位置            00 – 从主机端访问时没有错误            01 – 在 TBUF 中从主机端访问时出错            10 – 在 RBUF 中从主机端访问时出错            11 – 在 ACF 中从主机端访问时出错            在主机端的读取访问期间，HELOC 将使用每个新错误进行更新。            这就足够了，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。            HELOC 只会在出现错误时更新，但在由更正的单比特错误引起的警告时不会更新。</p>
26	TXB	r	0x0	<p>传输块            0 – 正常运行            1 – 传输受阻            如果在 CAN 协议机器读取数据进行传输时由于错误而设置了 MEIF 或 MAEIF，则传输将立即被阻止。            如果设置了 SEIF，则传输也会立即被阻止。            如果 TXB=1，则冻结错误计数器。            如果重置=1，则重置 TXB。</p>
25	TXS	r	0	<p>传输停止            0 – 正常运行            1 – 传输停止            如果在优先级重新排序计算机访问内存时由于错误而设置 MEIF 或 MAEIF，则任何新的传输都将停止。如果有活动传输，这将在停止之前完成，但如果在此传输过程中发生错误，则不会开始重新传输。            如果重置=1，则重置 TXS。</p>
24	ACFA	rw	0	<p>接受过滤器接收            0 – ACF 的正常运行            1 – ACF 禁用：接受所有接收的帧            如果由于 ACF 地址范围中的错误而设置 MEIF 或 MAEIF，则设置 ACFA。然后禁用接受过滤，所有帧都将被接受。</p>

Bit	Name	R/W	Reset Value	Function
				ACFA 可以通过写入 1 来重置类似于中断标志。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并设置 RIF 后重置它。 如果重置=1，ACFA 也会重置。
3:1	FSTIM	rw	0x000	故障刺激 故障刺激只有在 XMREN=1 时才有可能。
0	Res	-	-	-

#### 24.4.14. 筛选器组控制寄存器 (CANFD\_ACFCR)

Address offset: 0x44

Reset value: 0x00010000

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.	Res.	Res.	Res.	Res.
ACFADR															

Bit	Name	R/W	Reset Value	Function
31:16	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。禁用的过滤器拒绝一帧。如果适当的 ACFC / ACFM 配置匹配，则只有启用的过滤器才能接受帧。
3:0	ACFADR	rw	0	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

#### 24.4.15. 筛选器组 code 寄存器 (CANFD\_ACFC)

Address offset: 0x48

Reset value: 0xFFFFFFFF

0x00	CANFD_ID
0x04	CANFD_FORMAT
0x08	CANFD_TYPE

注：上述寄存器的控制含义请参考 LLC 帧格式定义

#### 24.4.16. 筛选器组 mask 寄存器 (CANFD\_ACFM)

Address offset: 0x58

Reset value: 0xFFFFFFFF

0x00	CANFD_ID
0x04	CANFD_FORMAT
0x08	CANFD_TYPE

注：上述寄存器的控制含义请参考 LLC 帧格式定义

#### 24.4.17. CAN 接收 BUF 寄存器 (CANFD\_RBUF)

Address offset: 0x70

Reset value: 0xFFFFFFFF

注：上述寄存器的控制含义请参考 LLC 帧格式定义

#### 24.4.18. CAN 发送 BUF 寄存器 (CANFD\_TBUF)

Address offset: 0xcc

Reset value: 0xFFFFFFFF

注：上述寄存器的控制含义请参考 LLC 帧格式定义

#### 24.4.19. CAN 寄存器映像

Offset	Register	Data															
0x00	CANFD_ID	Res.	Res.	31	Res.	Res.	30	Res.	Res.	29	Res.	Res.	28	Res.	Res.	27	Res.
0x04	CANFD_FORMAT	Res.	Res.	ESI	LBF	KOER[2:0]	HANDLE[7:0]	Res.	Res.	26	Res.	Res.	25	Res.	Res.	24	Res.
0x08	CANFD_TYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	23	Res.	Res.	22	Res.	Res.	21	Res.
0x0C	CANFD_AF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	20	Res.	Res.	19	Res.	Res.	18	Res.
0x10	CANFD_TTCAN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	17	Res.	Res.	16	Res.	Res.	15	Res.
0x14	CANFD_DA TA1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	14	Res.	Res.	13	Res.	Res.	12	Res.
0x18	...	...	...	...	...	...	...	...	...	11	Res.	Res.	10	Res.	Res.	9	Res.
0x1C	...	...	...	...	...	...	...	...	...	8	Res.	Res.	7	Res.	Res.	6	Res.
0x20	...	...	...	...	...	...	...	...	...	5	Res.	Res.	4	Res.	Res.	3	Res.
0x24	...	...	...	...	...	...	...	...	...	2	Res.	Res.	1	Res.	Res.	0	Res.
0x28	CANFD_DDATA16	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]	data[31:0]

# 25. 串行外设接口 (SPI)

## 25.1. 简介

SPI 接口可以配置为支持 SPI 协议或者支持 I2S 音频协议。 SPI 接口默认工作在 SPI 方式，可以通过软件把功能从 SPI 模式切换到 I2S 模式。

串行外设接口(SPI)允许芯片与外部设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。它可用于多种用途，包括使用一条双向数据线的双线单工同步传输，还可使用 CRC 校验的可靠通信。

I2S 也是一种 3 引脚的同步串行接口通讯协议。它支持四种音频标准，包括飞利浦 I2S 标准， MSB 和 LSB 对齐标准，以及 PCM 标准。它在半双工通讯中，可以工作在主和从 2 种模式下。当它作为主设备时，通过接口向外部的从设备提供时钟信号。

### 25.1.1. 主要特征

#### 25.1.1.1. SPI 特征

- 3 线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8 或 16 位传输帧格式选择
- 主或从操作
- 支持多主模式
- 8 个主模式波特率预分频系数(最大为 fPCLK/4)
- 从模式频率 (最大为 fPCLK/4)
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 可编程的时钟极性和相位
- 可编程的数据顺序， MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
  - SPI 总线忙状态标志
- 支持可靠通信的硬件 CRC
  - 在发送模式下， CRC 值可以被作为最后一个字节发送
  - 在全双工模式中对接收到的最后一个字节自动进行 CRC 校验
- 可触发中断的主模式故障、过载以及 CRC 错误标志
- 两个具有 DMA 功能的 64 (16x4) 位嵌入式 Rx 和 Tx FIFO
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接受请求

#### 25.1.1.2. I2S 功能

- 单工通信(仅发送或接收)
- 主或者从操作

- 8 位线性可编程预分频器，获得精确的音频采样频率(8KHz 到 96kHz)
- 数据格式可以是 16 位， 24 位或者 32 位
- 音频信道固定数据包帧为 16 位(16 位数据帧)或 32 位(16、 24 或 32 位数据帧)
- 可编程的时钟极性(稳定态)
- 从发送模式下的下溢标志位和主/从接收模式下的溢出标志位
- 16 位数据寄存器用来发送和接收，在通道两端各有一个寄存器
- 支持的 I2S 协议：
  - I2S 飞利浦标准
  - MSB 对齐标准(左对齐)
  - LSB 对齐标准(右对齐)
  - PCM 标准(16 位通道帧上带长或短帧同步或者 16 位数据帧扩展为 32 位通道帧)
- 数据方向总是 MSB 在先
- 发送和接收都具有 DMA 能力
- 主时钟可以输出到外部音频设备，比率固定为 256xFs(Fs 为音频采样频率)

### 25.1.2. 模块框图

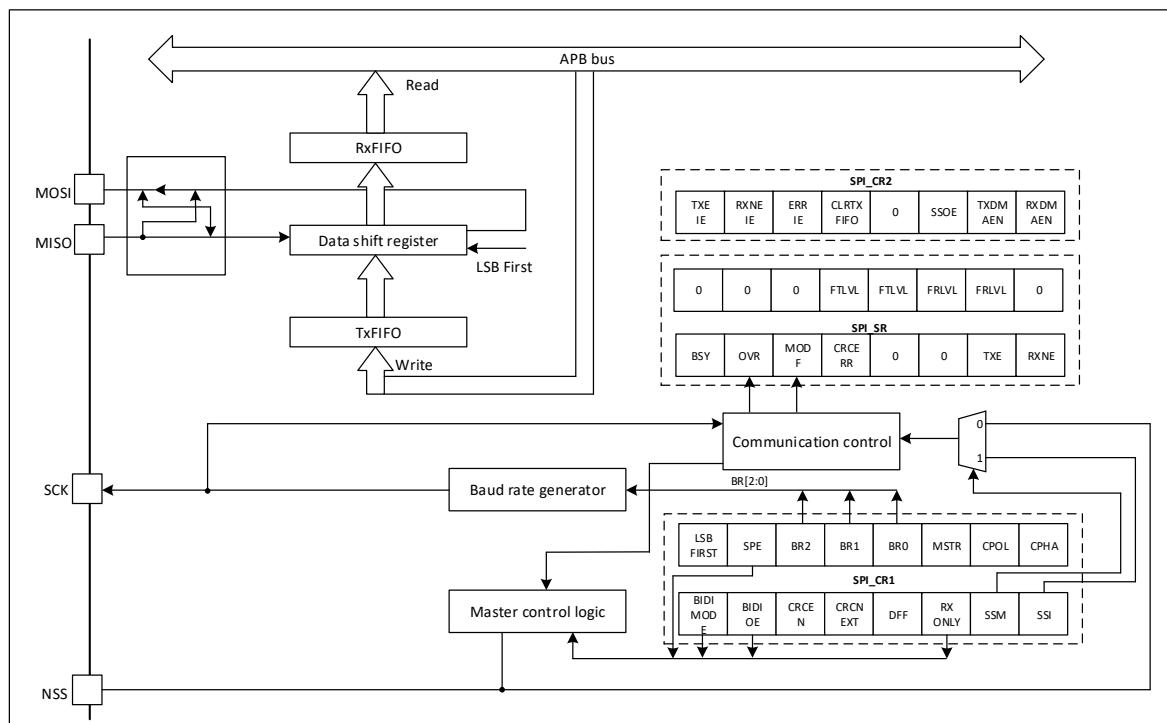


图 25-1 SPI 模块

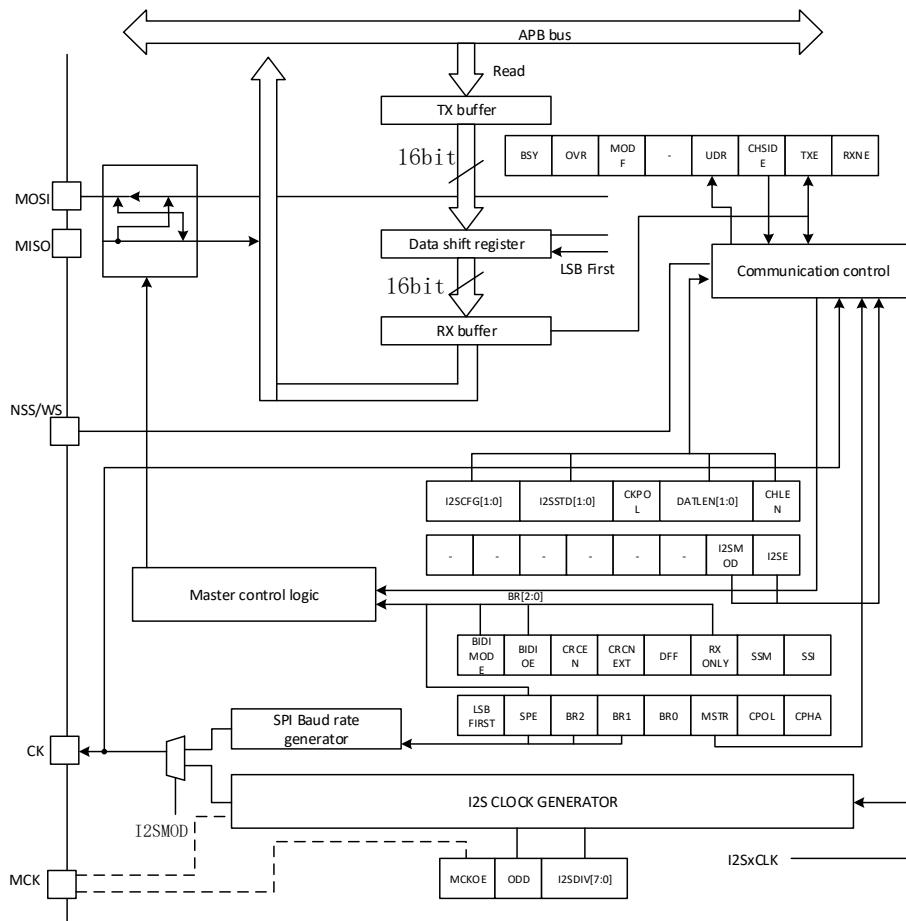


图 25-2 I2S 模块

## 25.2. SPI 功能描述

### 25.2.1. 概述

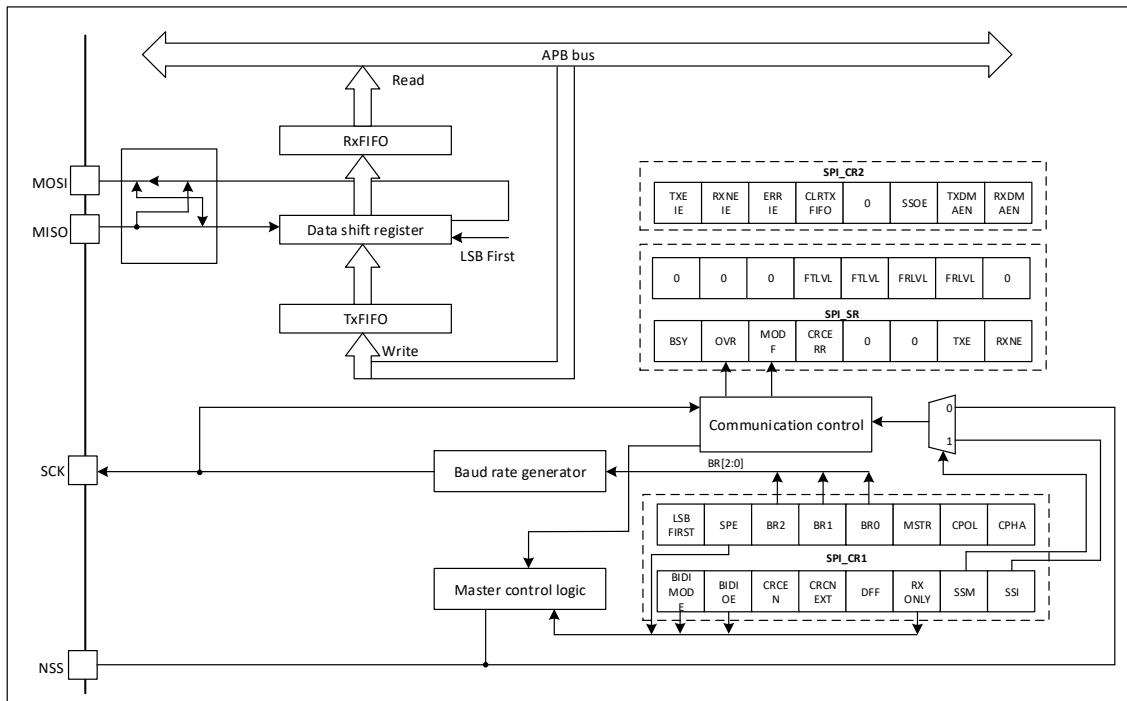


图 25-3 SPI 框图

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 的设定，该 pin 可以用作：

- 选择要通讯的 slave
- 同步数据帧
- 发现多 master 间的冲突

SPI 总线允许在一个 master 和 1 个或者多个 slave 之间的通讯。总线由至少两根线组成：一个是时钟，另一个是被同步传输的数据。根据应用场景，可以选择增加另外一根数据线和 slave select 信号。

## 25.2.2. 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 management），或者 3 线、4 线（硬件 NSS management）。通讯通常都被主机启动。

### 25.2.2.1. 全双工通信

缺省情况，SPI 被配置成全双工通讯。在这种配置下，主机和从机的 shift 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的被移位。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 bit 被转换完成），在主机和从机之间的信息就被交互过了。

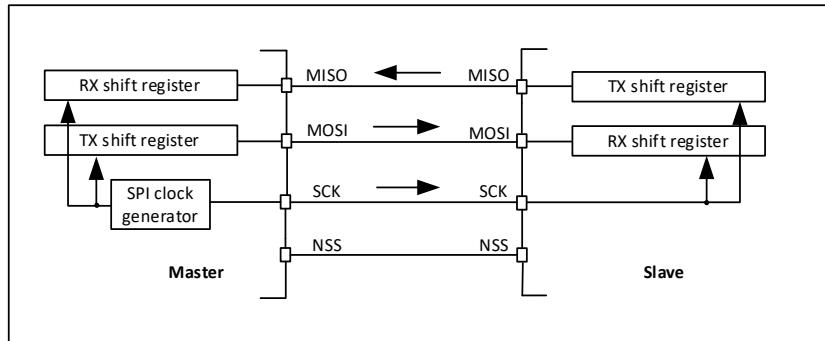


图 25-4 全双工单主机/单从机应用

### 25.2.2.2. 半双工通信

通过设定 BIDIMODE bit (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 shift 寄存器的连接。在通讯过程中, 在 SCK 的时钟沿, 数据在两个 shift 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。

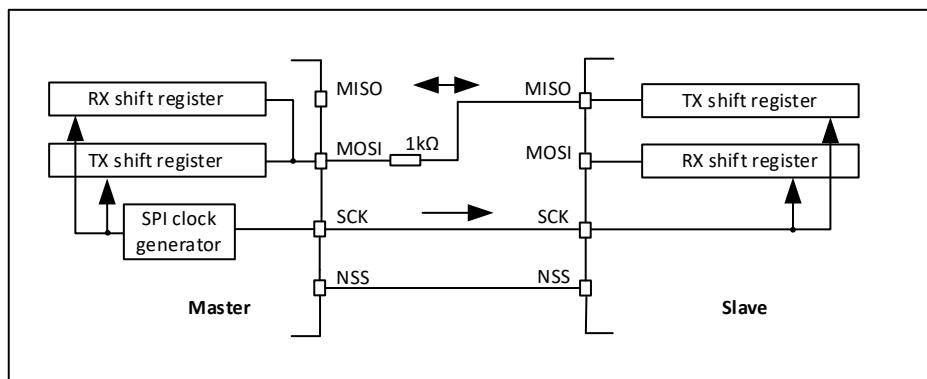


图 25-5 半双工单主机/单从机应用

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 和从机的 MOSI 可以用作通用端口给其他应用使用。

### 25.2.2.3. 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 工作在单线模式下。在这个配置下, 在主机和从机的 shift 寄存器之间只使用 1 根线。另一对 MISO 和 MOSI 不被使用, 可以被释放成通用端口。

- 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, 应用可以禁能 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 buffer 的配置。在主机配置下, MOSI 输出被禁能, 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。

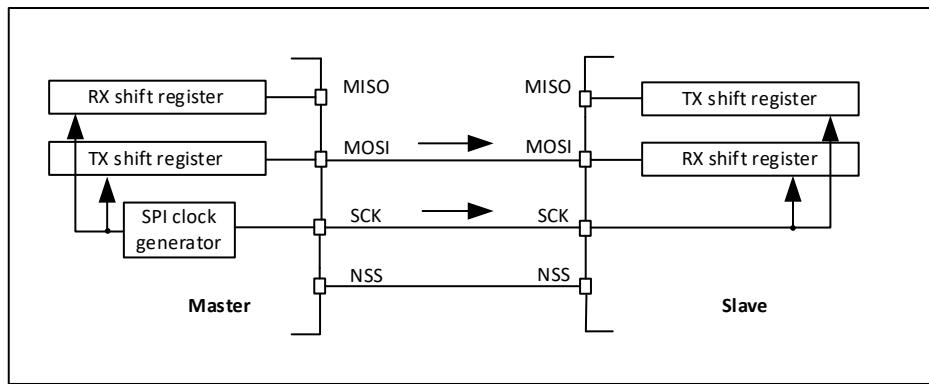


图 25-6 单工单从机/单主机应用

(主设备处于仅发送模式/从模式处于仅接收模式)

- 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 shift 寄存器的输入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 pin 都被用作 GPIO。

通过用传输方向的设定（在 BIDIO bit 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。

### 25.2.3. 多从机通信

在一个有两个或者更多独立从机的配置里，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 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。

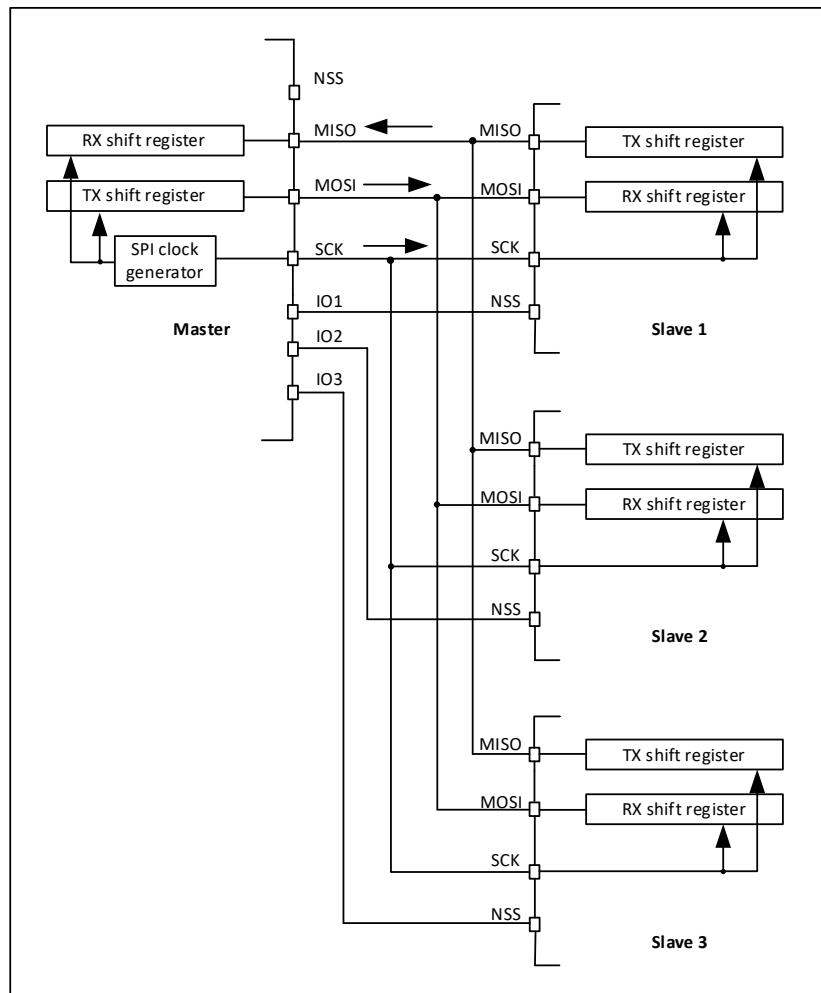


图 25-7 主机与三个独立的从机通信

NSS 在这种配置下在主机端未被使用。必须通过  $SSM=1$ ,  $SSI=1$  来防止任何 MODF 错误。

由于从机的 MISO 连接到一起，所有从机必须把他们 MISO 的 GPIO 配置作为 AF open-drain。

#### 25.2.4. 多主机通信

除非 SPI 总线不是被设计成多主机能力，否则用户可以使用其内嵌 feature，该 feature 可以发现两个试图同时控制总线的节点存在潜在冲突。当需要用到这种检测时，要使用配置为硬件输入模式的 NSS pin。

在该模式下超过两个 SPI 节点工作的连接是不可能的，原因是同一时间仅有一个节点能够把输出放到共用数据线上。

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，它自己切换成主机模式，并把有效的电平经过专门的 GPIO 给予其余节点的从机 select input。在该进程完成后，有效的 select 信号被释放，控制总线的节点暂时返回 passive mode，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：通过提前定义的给两个节点的不同的超时推迟下一次尝试）

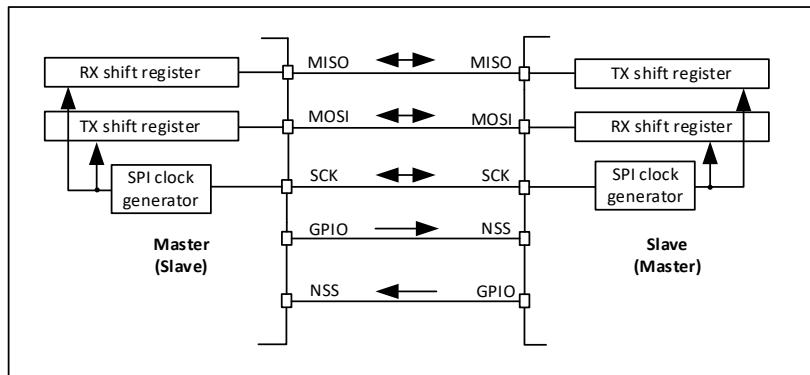


图 25-8 多主机应用

NSS pin 在两个节点都被配置成硬件输入模式。他的有效电平使能了 MISO 输出控制，而 passive node 被配置成从机。

### 25.2.5. 从选择(NSS)脚管理

在 slave mode, 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 bit，可以选择硬件或者软件从机管理：

- 软件 NSS 管理 (SSM=1)：在这个配置下，从机 select 信号被内部的 SSI bit (SPI\_CR1寄存器) 值驱动。外部 NSS pin 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 pin。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 disable (SPE=0) 。在多主机应用中，SPI 不能进行这种 NSS 配置。
  - 2) NSS 输出 disable (SSM=0, SSOE=0)：如果 MCU 在总线上作为主机，这个配置允许进行多主机能力。如果 NSS pin 此时被拉低，SPI 进入主机 mode fault 状态，芯片自动被重配置为从机模式。在从机模式，NSS pin 作为标准的片选输入，当 NSS 为低时，从机被选中。

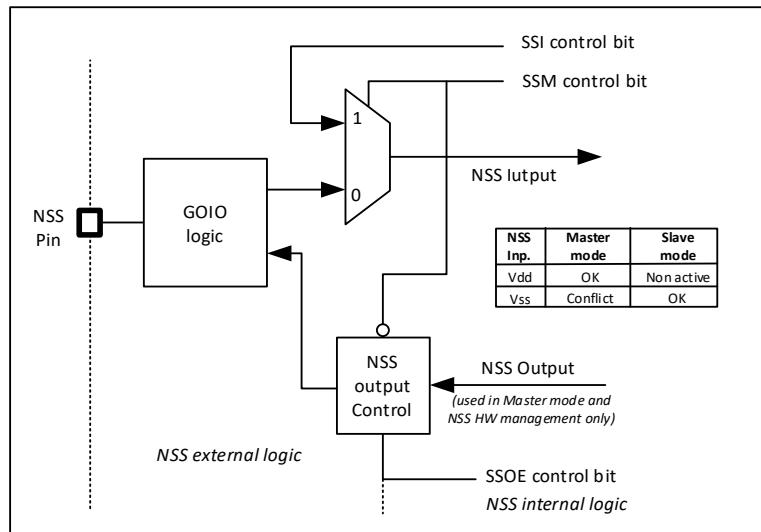


图 25-9 硬件/软件从机选择管理

## 25.2.6. 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。

### 25.2.6.1. 时钟相位和时钟极性控制

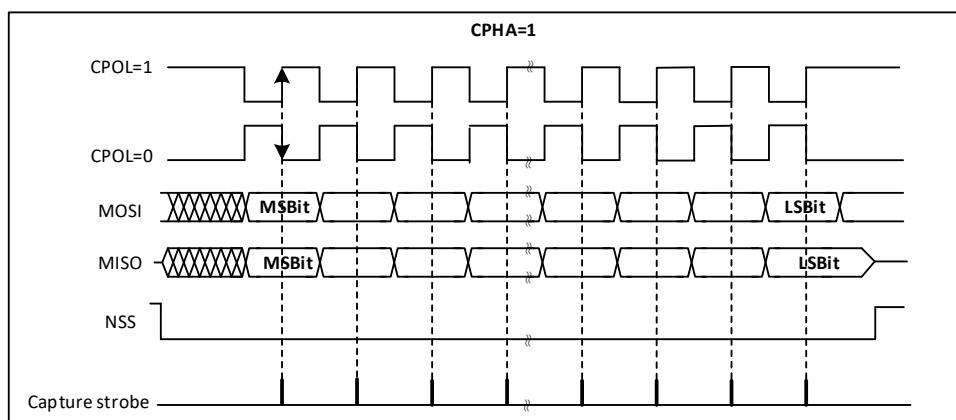
通过 CPOL 和 CPHA bit (SPI\_CR1 寄存器)，软件可以配置 4 种可能的时序。CPOL (clock polarity) 控制当没有数据传输时的 clock 的 IDLE 状态。该位对主机和从机都有影响。如果 CPOL 被复位，SCK pin 有低电平的状态。如果 CPOL 被置位，SCK pin 有高电平的 IDLE 状态。

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）。在时钟变化类型的出现，数据被锁存。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）。在该时钟变化类型出现时，数据被锁存。

CPOL 和 CPHA 的组合选择了数据捕获时钟边沿。

在 CPOL/CPHA 改变之前，SPI 必须被 disable (SPE=0)。

SCK 的 IDLE 状态必须对应被 SPI\_CR1 寄存器选择的极性。



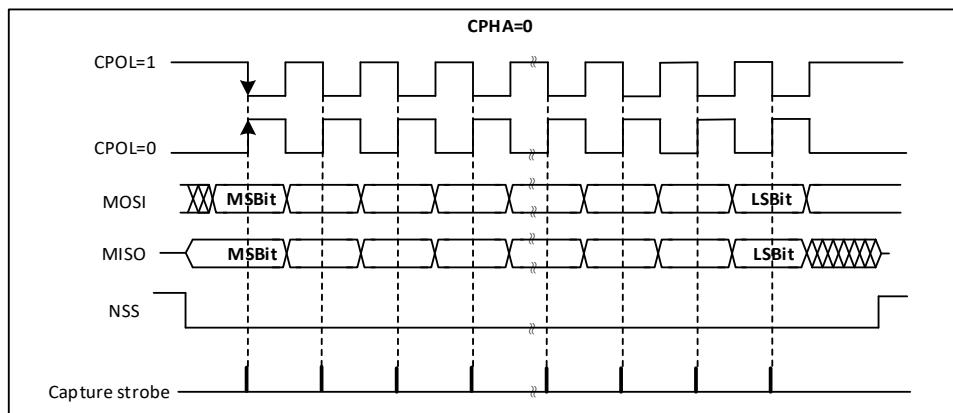


图 25-10 数据时钟时序图

数据 bit 的顺序取决于 LSBFIRST bit 的设定。

#### 25.2.6.2. 数据帧格式

通过 LSBFIRST bit(SPI\_CR1 寄存器)，SPI shift 寄存器可以设定为 MSB-FIRST 或者 LSB-FIRST。通过使用 DFF bit(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。

#### 25.2.7. 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：

- 写相关的 GPIO 寄存器：配置 MOSI、MISO 和 SCK pin
- 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
  - 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
  - 配置 SSM 和 SSI
  - 配置 MSTR bit (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 bit，选择数据帧位数
- 写 SPI\_CR2 寄存器
  - 配置 SSOE (从机模式不需要)
- 写相应的 DMA 寄存器：配置 DMA 的 SPI Tx 和 Rx 通道

#### 25.2.8. SPI 使能流程

推荐在主机发送时钟之前使能 SPI 从机。如果不这样处理，不期望的数据传输可能会发生。从机的数据寄存器必须在开始与主机通讯之前，已经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）。SCK 信号必须在 SPI slave 被使能之前，固定在 IDLE 状态 level（相应的被选择极性）。

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，主机开始通讯。

在任何主机只收模式（RXONLY=1，或者 BIDIMODE=1 且 BIDIOE=0），在 SPI 被使能后，master 开始通讯，时钟立即被提供。

对于 DMA 处理，遵从具体的章节内容。

## 25.2.9. 数据传输和接收流程

### 25.2.9.1. 接收和发送缓冲区

SPI 所有数据通讯都通过 FIFO，该特性使 SPI 能够以连续数据流进行工作，并防止由于 CPU 来不及处理数据导致的通讯问题。发送和接收有独立的 FIFO，叫做 TXFIFO 和 RXFIFO。这些 FIFO 被用在所有的 SPI 模式。

FIFO 的处理取决于多种参数，包括：数据交换模式（全双工、半双工）、数据帧格式、访问 FIFO 数据寄存器的大小（8 位还是 16 位）。

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。读访问必须通常与 RXFIFO 阈值对齐，FTLVL[1:0]和 FRLVL[1:0]位显示了两个 FIFO 当前的占用级别。

对 SPI\_DR 寄存器的都访问必须通过 RXNE 事件管理。当数据存储器在 RXFIFO 非空，该事件被触发。当 RXNE 被清零，RXFIFO 就被认为为空的。

相似地，写要发送的数据帧，通过 TXE 事件管理。当 TXFIFO Level 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，而 TXFIFO 仅可以存 3 个数据帧（当数据帧格式不大于 8bit）。这样的差别，可以防止以下混乱情况的发生：3 个 8-bit 数据帧已经存在 TXFIFO，而此时软件要向 TXFIFO 以 16-bit（CPU 数据宽度）模式再写入数据。

TXE 和 RXNE 事件都都可以通过查询、中断和 DMA 方式处理。

当 RXFIFO 满时，如果下一个数据被接收，则 overrun 事件产生。Overrun 事件可以通过查询和中断的方式处理。

被置位的 BSY 位显示了 1 个当前数据帧的通讯正在进行。当时钟信号连续的提供，在 master 端的两个数据帧之间，BSY 标志保持置位。但在 slave 端的每个数据帧传输之间，BSY 会保持最小 1 个 SPI Clock 宽度的低电平。

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。

### 25.2.9.2. 顺序处理

一些数据帧可以通过单一序列传递来完成一条信息。当发送被使能，当主机的 TXFIFO 里有任何数据，sequence 开始并继续进行。时钟信号被主机连续的提供，直到 TXFIFO 空，然后停止等待额外的数据。

在只接收模式，即半双工（BIDIMODE=1, BIDIOE=0）或者单工模式（BIDIMODE=0, RXONLY=1），在 SPI 被使能和只接收模式被激活，主机就立即开始发送。主机一直会提供时钟，直到主机停止了 SPI 或者只接收 模式。主机连续的接收数据。

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，并提供或者更慢的时钟，或者分开的帧，或者重组 delay 的数据包。要注意的是，对于主机或者从机来说，没有 underflow 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 sequence 都必须被 NSS 脉冲包住，同时在多从机系统钟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，但是再次提供脉冲

当 BSY 被置位，它显示了正在进行的数据帧交互。当专门的帧交互被完成时，RXNE 标志置位。最后一个 bit 被采样，并且整个数据帧被存在 RXFIFO 中。

#### 25.2.9.3. 禁用 SPI 流程

当 SPI 被 disable 掉，必须按照特定的 disable 流程。对于系统 disable SPI 的流程是很重要的，因为此后应用上，外设时钟会被停掉，系统进入低功耗模式。这种情况下（disable），正在进行的交互会被破坏。在一些模式下，disable 流程是唯一停止连续通讯的办法。

全双工或者只发送模式下，主机可以当停止提供要发送的数据时完成交互。在这种情况下，在最后的数据交互后，时钟被停止。要额外注意 packing mode（当交互奇数个数的数据帧，以放置一些 dummy 字节交互）。在这些模式下，SPI 被 disable 之前，用户必须使用标准的 disable 流程。当 SPI 被 disable 在主机发送时，如果此时一个帧交互正在进行，或者下一个数据帧存在 TXFIFO 中，SPI 的功能是不能被保证的。

当主机处在任何只接收模式，停止连续时钟的唯一方法是停止外设（SPE=0）。该模式下，要进行专门的 SPI disable 流程。

当 SPI 被 disable，接收到未读走的数据存放在 RXFIFO 中，这些数据必须在下一次 SPI 使能要开始新的序列之前被处理掉。为防止有未读的数据，要确保当 SPI 被 disable 时，RXFIFO 是空的（使用正确的 disable 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 disable 流程是基于 BSY 状态，并查看 FTLVL[1:0]，以确保传输彻底完成。也可以通过特定别的检查来鉴别正在进行交互的结束，例如：

- 当 NSS 信号被软件管理，主机要向从机提供正确的 NSS 脉冲。或者
- 当完成来自 DMA 或者 FIFO 的交互数据流时，此时最后的数据帧或者 CRC 帧仍在传输过程中。

正确的 disable 流程是（receive-only 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）
- Disable 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 receive-only 模式，正确的 disable 流程是：

- 在最后一个数据帧传输过程中，通过 disable 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）

#### 25.2.9.4. 使用 DMA 通信

为以最大的速度工作，并加快数据的读/写过程，以避免 overrun，SPI 具备简单请求/应答协议的 DMA 能力。

当 TXE 或者 RXNE 被置位，会产生 DMA 请求。Tx 和 Rx buffer 有独立的请求。

- 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。
- 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），可以通过监测 BSY 标志来确保 SPI 通讯已完成。这是用来避免当 disable SPI 或者进入 stop 模式时，最后的传输被破坏。软件必须等待 FTLVL[1:0]=00，然后再等待 BSY=0。

当开始使用 DMA 通讯时，违反之 DMA 通道管理的错误事件发生，必须遵从以下步骤：

- 使能 DMA Rx buffer (SPI\_CR2 的 RXDMAEN bit) (如果 Rx DMA 被使用)
- 使能 Tx Rx DMA streams (在 DMA 寄存器里) (如果 streams 被用到)
- 使能 DMA Tx buffer (在 SPI\_CR2 寄存器的 TXDMAEN bit) (如果 Tx DMA 被使用)
- 通过 SPE bit 使能 SPI

强制用以下步骤关闭通讯：

- Disable DMA Tx Rx streams (在 DMA 寄存器里) (如果 stream 被使用)
- 通过 SPI disable 流程 disable SPI
- 通过清除 TXDMAEN 和 RXDMAEN (SPI\_CR2 寄存器)， disable DMA Tx 和 Rx buffer (如果 DMA Tx and/or Rx 被使用)

#### 25.2.9.5. 通信时序

本节介绍一些典型的时序，这些时序对于查询、中断或者 DMA 都是有效的。为了简化，假定 LSBFIRST=0，CPOL=0，CPHA=1。也不提供完整的 DMA 操作配置。

- 当 NSS 有效，SPI 被使能，从机开始控制 MISO，当 NSS 被释放或者 SPI 关闭从机失去对 MISO 的控制。对于从机，在传输开始前必须提供充足的时间给主机，以便提前准备数据。
  - 在主机端，仅在 SPI 被使能时，SPI 外设会控制 MOSI 和 SCK 信号（也包括 NSS 信号）。如果 SPI 被 disable，SPI 外设就从 GPIO 断开，因此在这些线上的电平值取决于 GPIO 的设定。
- 在主机端，如果通讯是连续的，则 BSY 在帧之间保持有效。在从机端，BSY 信号在数据帧之间通常变低至少一个时钟周期。
- 只有当 TXFIFO 是满的，TXE 信号才被清零。
- 在 TXDMAEN bit 一被置位，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装进了 TxFIFO，DMA Tx TCIF 标志变高发生在甚至 SPI 通讯之前。这个标志在 SPI 交互完成之前，一直都为高。

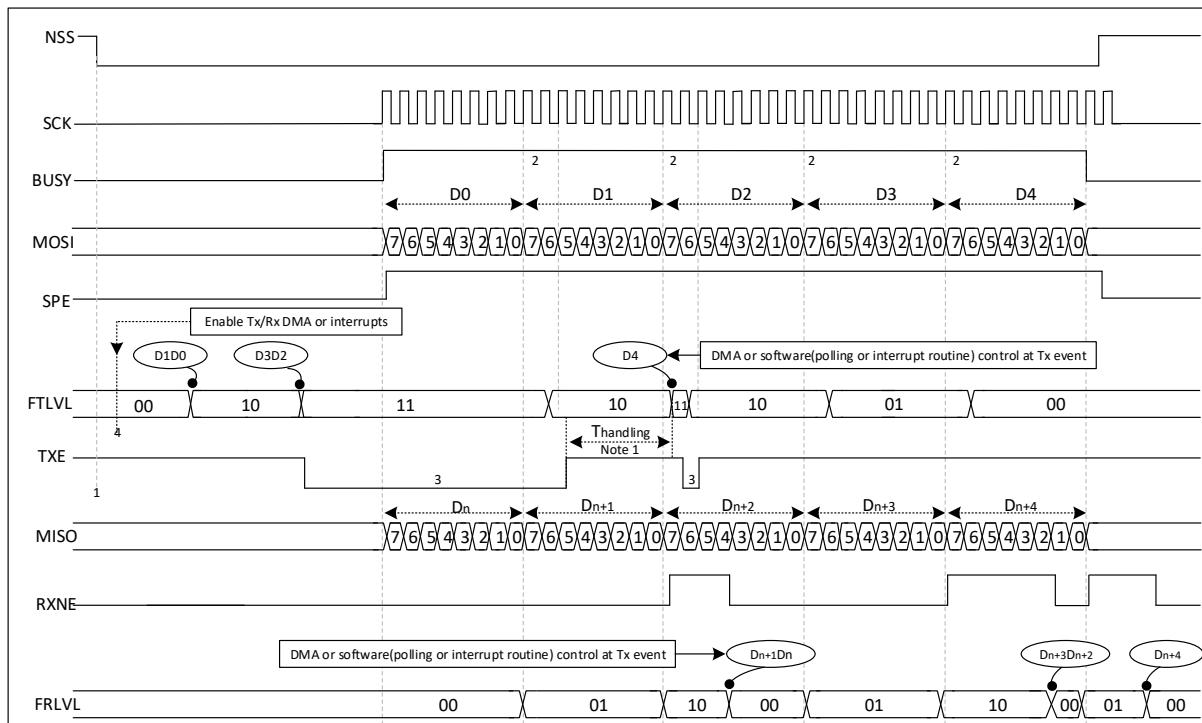


图 25-11 主机全双工通信时序图(bit frame=8)

### 25.2.10. 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。

#### 25.2.10.1. 发送缓冲区为空标志(TXE)

当 TXFIFO 有足够的空间存放要发送的数据时，TXE 标志位被置位。TXE 标志位与 TXFIFO level 有关。该标志位变高并保持高电平，直到 TXFIFO level 小于等于 1/2 FIFO 深度。如果 TXEIE (SPI\_CR2) 被置位，则会产生中断请求。当 TXFIFO level 大于 1/2，该位被自动清零。

#### 25.2.10.2. 接收缓冲区非空标志(RXNE)

RXNE 标志位被置位：

- RXNE 变高并保持高电平，直到 RXFIFO level 大于或者等于 1/4(8-bit)。
- 如果 RXNEIE 位 (SPI\_CR2) 被置位，则产生中断。
- 当上述条件不再成立，则 RXNE 被硬件自动清零。

#### 25.2.10.3. 忙标志(BSY)

BSY 标志由硬件设置与清除(写入此位无效果)，此标志表明 SPI 通信层的状态。

当它被设置为'1'时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下(MSTR=1、BDM=1 并且 BDOE=0)，在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式(或关闭设备时钟)之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多 master 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':

- 当 SPI 被正确的 disable 掉
- 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

Note: 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。

### 25.2.11. 错误标志

#### 25.2.11.1. 主模式失效(MODF)

主模式失效 (MODF) 仅发生在: 当 NSS 作为输入信号 (SSOE=0), NSS 引脚硬件模式管理下, 主设备的 NSS 脚被拉低; 或者在 NSS 引脚软件模式管理下, SSI 位被置为'0'时。此时, MODF 位被自动置位。主模式失效对 SPI 设备有以下影响:

- MODF 位被置为'1', 如果设置了 ERRIE 位, 则产生 SPI 中断;
- SPE 位被清为'0'。这将停止一切输出, 并且关闭 SPI 接口;
- MSTR 位被清为'0', 因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位:

- 当 MODF 位被置为'1'时, 执行一次对 SPI\_SR 寄存器的读或写操作;
- 然后写 SPI\_CR1 寄存器。

在有多个 MCU 的系统中, 为了避免出现多个从设备的冲突, 必须先拉高该主设备的 NSS 脚, 再对 MODF 位进行清零。在完成清零之后, SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑, 当 MODF 位为'1'时, 硬件不允许设置 SPE 和 MSTR 位。

通常配置下, 从设备的 MODF 位不能被置为'1'。然而, 在多主配置里, 一个设备可以在设置了 MODF 位的情况下, 处于从设备模式; 此时, MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

#### 25.2.11.2. 过载模式

当数据被主机或者从机接收, 并且 RXFIFO 没有足够的空间存接收到的数据时, 产生 overrun 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 overrun 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, 并且所有接下来发送的数据丢失。

依次读出 SPI\_DR 寄存器和 SPI\_SR 寄存器可将 OVR 清除。

### 25.2.12. SPI 中断

表 25-1 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
主模式失效事件	MODF	ERRIE
溢出错误	OVR	ERRIE

### 25.2.13. CRC 计算

为了检查发送和接收数据的可靠性，利用两个独立的 CRC 计算器。 SPI 提供独立于帧数据长度的 CRC8 或 CRC16 计算，可固定为 8 位或 16 位。对于所有其他数据帧长度，没有可用的 CRC。

#### 25.2.13.1. CRC 准则

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，设置  $CRCERR$  标志以指示数据错误。处理 CRC 计算的正确程序取决于 SPI 配置和选择的传输管理方式。

#### 25.2.13.2. CPU 管理的 CRC 传输

开始并连续通信直到发送或接收  $SPIx\_DR$  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设置。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需要多出一个数据帧。但是，当设置一个通过 16 位 CRC 校验的 8 位数据帧时，需要多两帧才能发送完整的 CRC 值。

当接收到最后一个 CRC 数据时，将执行自动校验，比较接收到的值和  $SPIx\_RXCRC$  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。CRC 接收后，CRC 值存储在 RXFIFO 中，必须读  $SPIx\_DR$  寄存器以清除 RXNE 标志。

#### 25.2.13.3. DMA 管理的 CRC 传输

当使用带 CRC 的 DMA 模式启用 SPI 通信时，通信结束时 CRC 的发送和接收是自动完成的（在仅接收模式下读取 CRC 数据除外）， $CRCNEXT$  位不必由软件处理。SPI 传输 DMA 通道的计数器必须设置为要传输的数据帧数，其中不包括 CRC 帧。在接收端，接收到的 CRC 值在传输结束时由 DMA 自动处理，但用户必须注意从 RXFIFO 中清除接收到的 CRC 信息，因为它总是存放到 RXFIFO 中。在全双工模式下，接收 DMA 通道的计数器可以设置为要接收的包括 CRC 在内的数据帧的数量。

在只接收模式下，DMA 接收通道计数器应该只包含传输的数据量，而不包括 CRC 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。

## 25.3. I2S 功能描述

### 25.3.1. 简介

通过将寄存器  $SPI\_I2SCFGR$  的  $I2SMOD$  位置为 ‘1’，即可使能 I2S 功能。此时，可以把 SPI 模块用作 I2S 音频接口。I2S 接口与 SPI 接口使用大致相同的引脚、标志和中断。

I2S 与 SPI 共用 3 个引脚：

SD：串行数据(映射至 MOSI 引脚)，用来发送和接收 2 路时分复用通道的数据；

WS：字选(映射至 NSS 引脚)，主模式下作为数据控制信号输出，从模式下作为输入；

CK：串行时钟(映射至 SCK 引脚)，主模式下作为时钟信号输出，从模式下作为输入。

在某些外部音频设备需要主时钟时，可以另有一个附加引脚输出时钟：

MCK：主时钟(独立映射)，在 I2S 配置为主模式，寄存器 SPI\_I2SPR 的 MCKOE 位为 ‘为 K 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。

### 25.3.2. 音频协议

三线总线支持 2 个声道上音频数据的时分复用：左声道和右声道，但是只有一个 16 位寄存器用作发送或接收。因此，软件必须在对数据寄存器写入数据时，根据当前传输中的声道写入相应数据；同样，在读取寄存器数据时，通过检查寄存器 SPI\_SR 的 CHSIDE 位来判明接收到的数据属于哪个声道。左声道总是先于右声道发送数据(CHSIDE 位在 PCM 协议下无意义)。有四种可用的数据和包帧组合。可以通过以下四种数据格式发送数据：

- 有 16 位数据打包进 16 位帧
- 帧 16 位数据打包进 32 位帧
- 帧 24 位数据打包进 32 位帧
- 帧 32 位数据打包进 32 位帧

在使用 16 位数据扩展到 32 位帧时，前 16 位(MSB)是有意义的数据，后 16 位(LSB)被强制为 0，该操作不需要软件干预，也不需要有 DMA 请求(仅需要一次读/写操作)。24 位和 32 位数据帧需要 CPU 对寄存器 SPI\_DR 进行 2 次读或写操作，在使用 DMA 时，需要 2 次 DMA 传输。对于 24 位数据，扩展到 32 位后，最低 8 位由硬件置 0。对于所有的数据格式和通讯标准，总是先发送最高位(MSB)。

I2S 接口支持四种音频标准，可以通过设置寄存器 SPI\_I2SCFGR 的 I2SSSTD[1:0]位和 PCMSYNC 位来选择。

#### 25.3.2.1. I2S 飞利浦协议

在此标准下，引脚 WS 用来指示正在发送的数据属于哪个声道。在发送第一位数据(MSB)前 1 个时钟周期，该引脚即为有效。

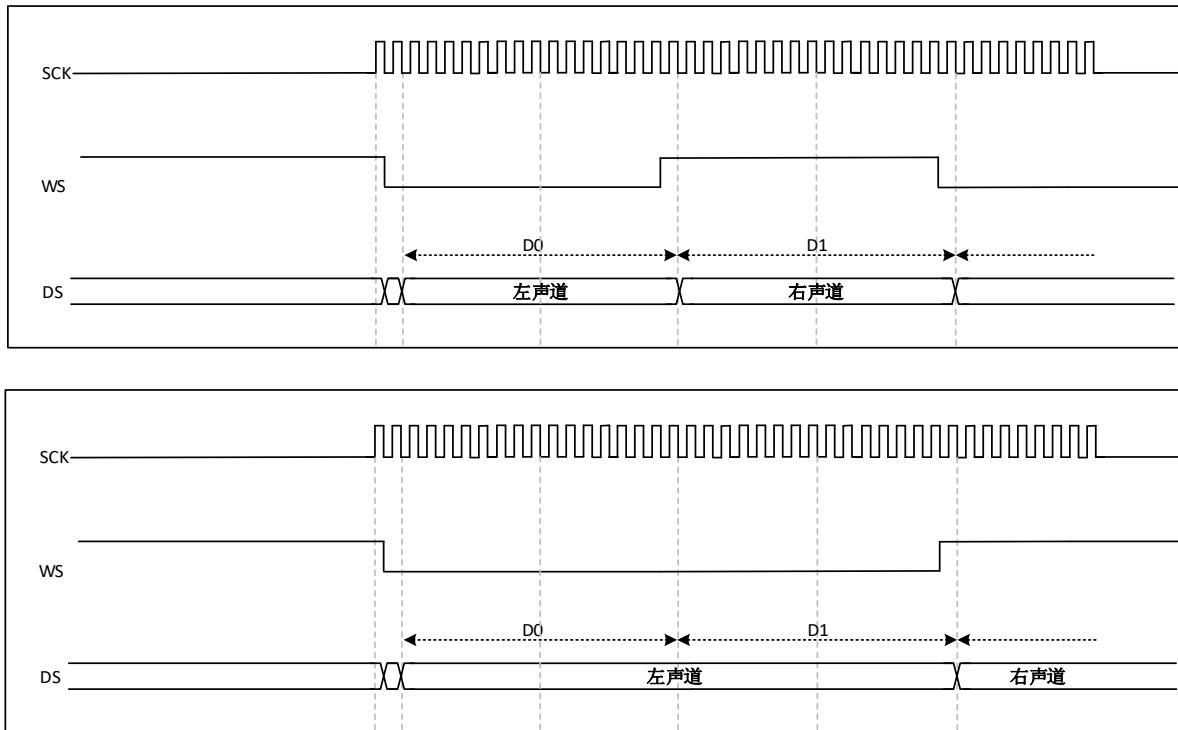


图 25-12 I2S 飞利浦协议波形(16/32 位全精度, CKPOL = 0)

当 CKPOL=1 时,

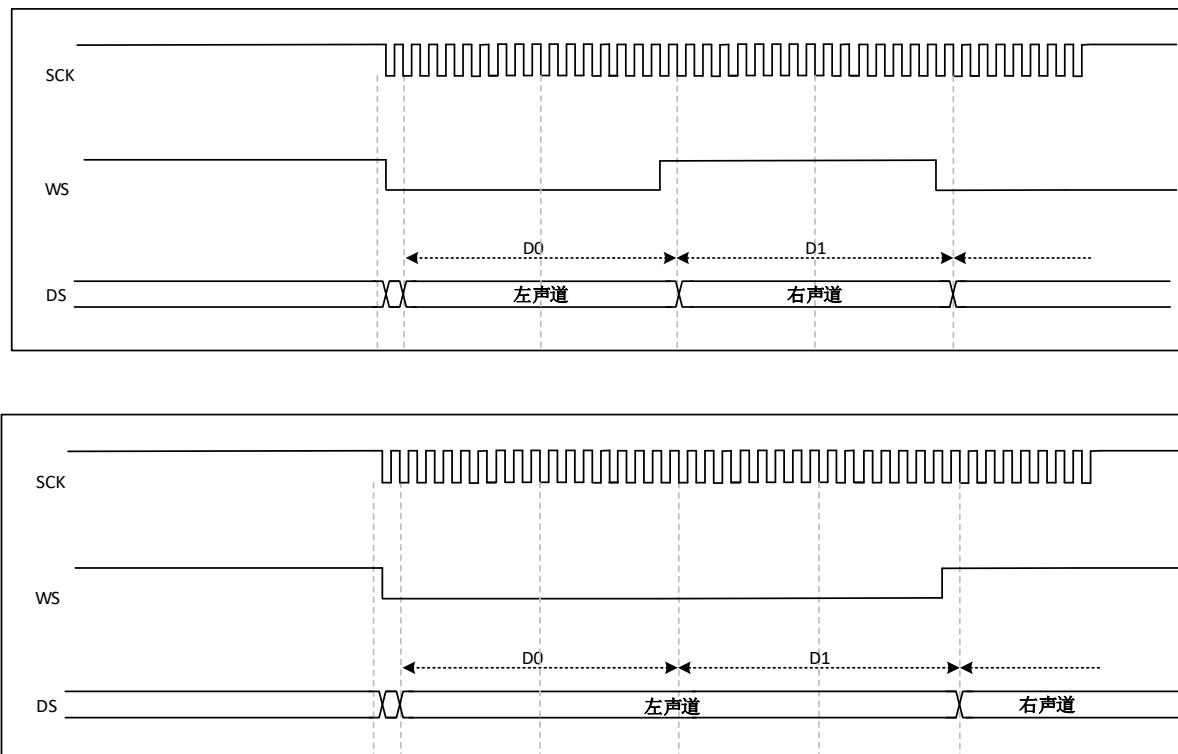


图 25-13 I2S 飞利浦协议波形(16/32 位全精度, CKPOL = 1)

发送方在时钟信号(CK)的下降沿改变数据, 接收方在上升沿读取数据。WS 信号也在时钟信号的下降沿变化。

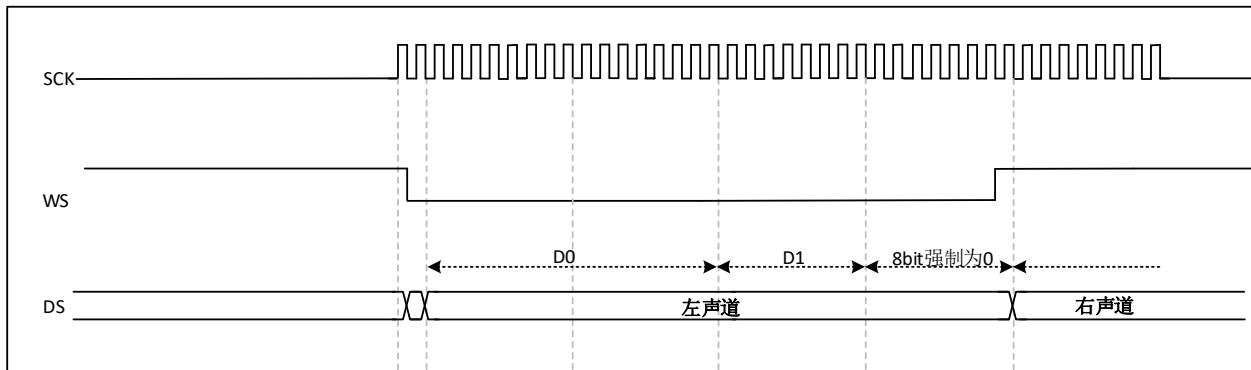


图 25-14 I2S 飞利浦协议标准波形(24 位帧, CKPOL = 0)当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

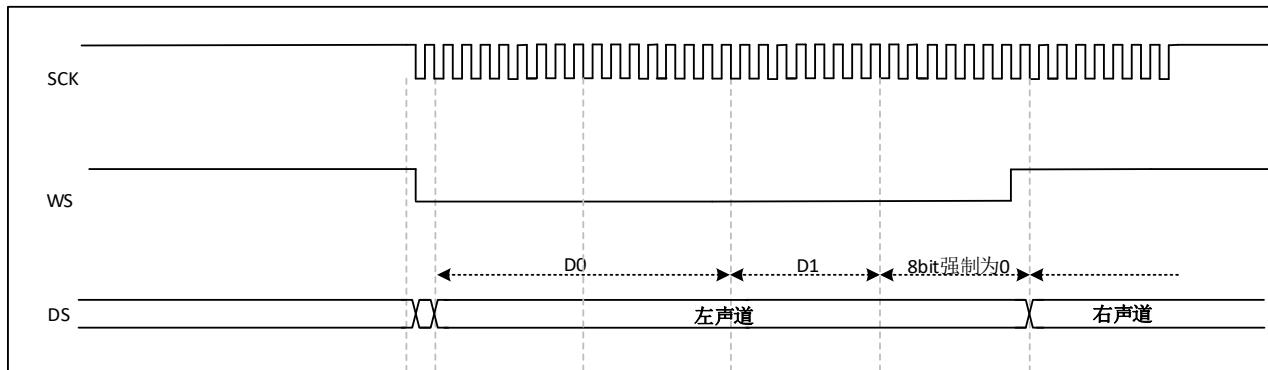


图 25-15 I2S 飞利浦协议标准波形(24 位帧, CKPOL = 1)

此模式需要对寄存器 SPI\_DR 进行 2 次读或写操作。

- 读在发送模式下：如果需要发送 0x8EAA33(24 位)：

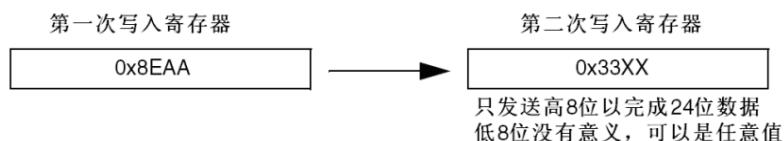


图 25-16 读在发送模式下

发送 0x8EAA33

- 在接收模式下：如果接收 0x8EAA33：

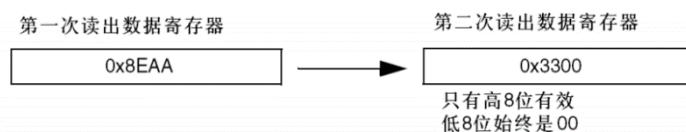


图 25-17 读在接收模式下

接收 0x8EAA33

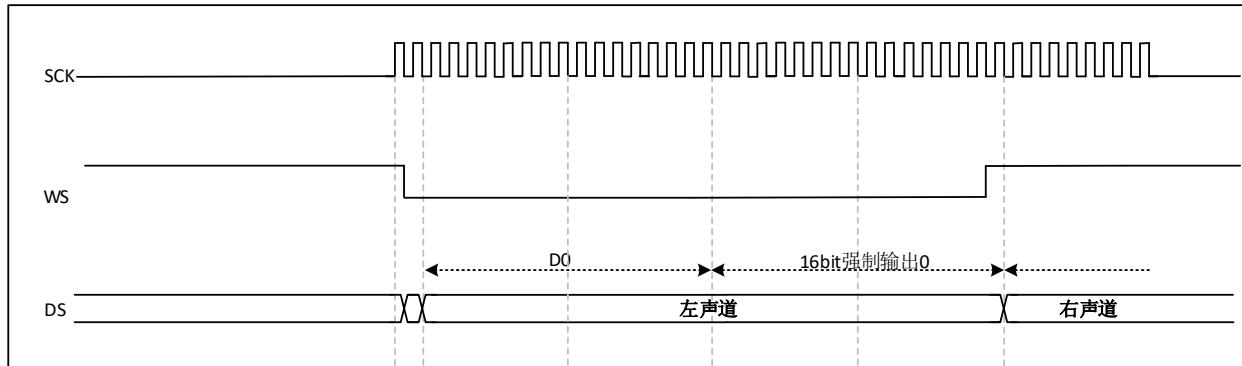


图 25-18 I2S 飞利浦协议标准波形(16 位扩展至 32 位包帧, CKPOL = 0)

当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

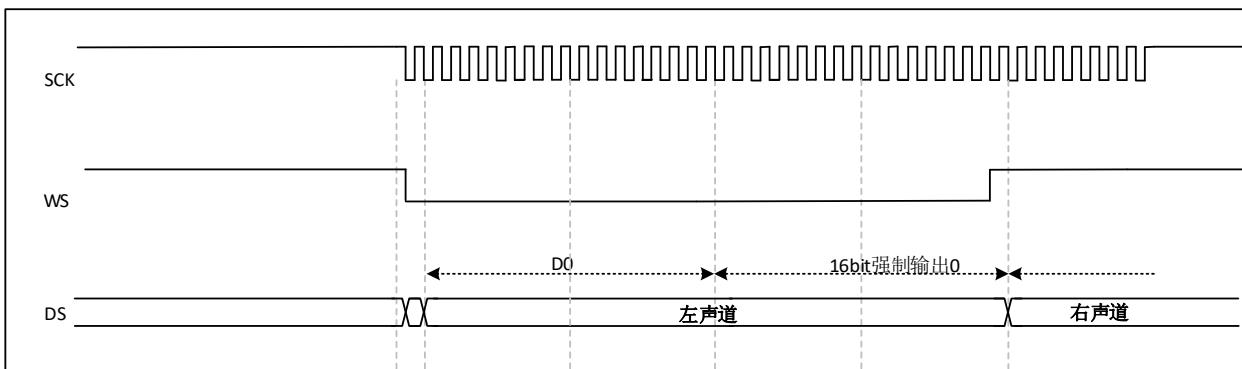


图 25-19 I2S 飞利浦协议标准波形(16 位扩展至 32 位包帧, CKPOL = 1)

在 I2S 配置阶段, 如果选择将 16 位数据扩展到 32 位声道帧, 只需要访问一次寄存器 SPI\_DR。用来扩展到 32 位的低 16 位被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3(扩展到 32 位是 0x76A30000), 需要的操作如下图所示。

只需要操作一次 SPI\_DR

0X76A3

在发送时需要将 MSB 写入寄存器 SPI\_DR; 标志位 TXE 为 ‘‘E 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 ‘‘N, 如果允许了相应的中断, 则可以产生中断。

这样, 在 2 次读和写之间有更多的时间, 可以防止下溢或者上溢的情况发生。

### 25.3.2.2. MSB 对齐标准•

在此标准下, WS 信号和第一个数据位, 即最高位(MSB)同时产生。

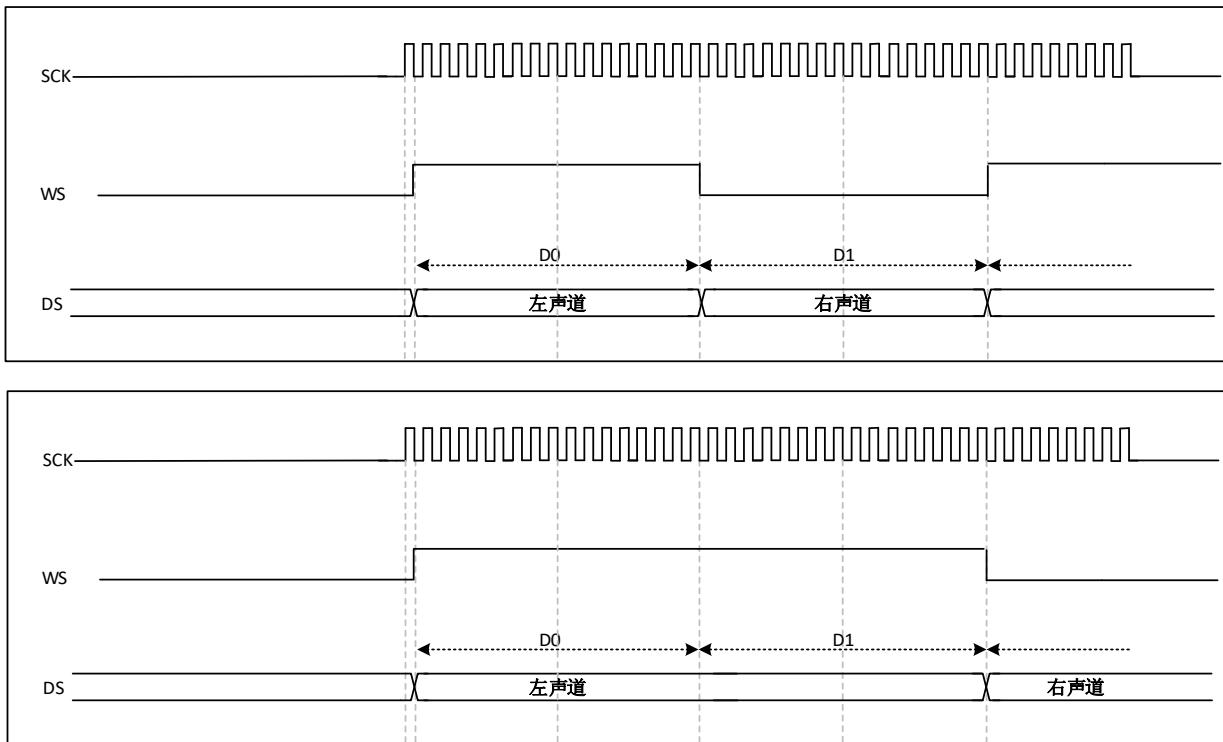


图 25-20 MSB 对齐 16 位或 32 位全精度, CKPOL = 0

当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

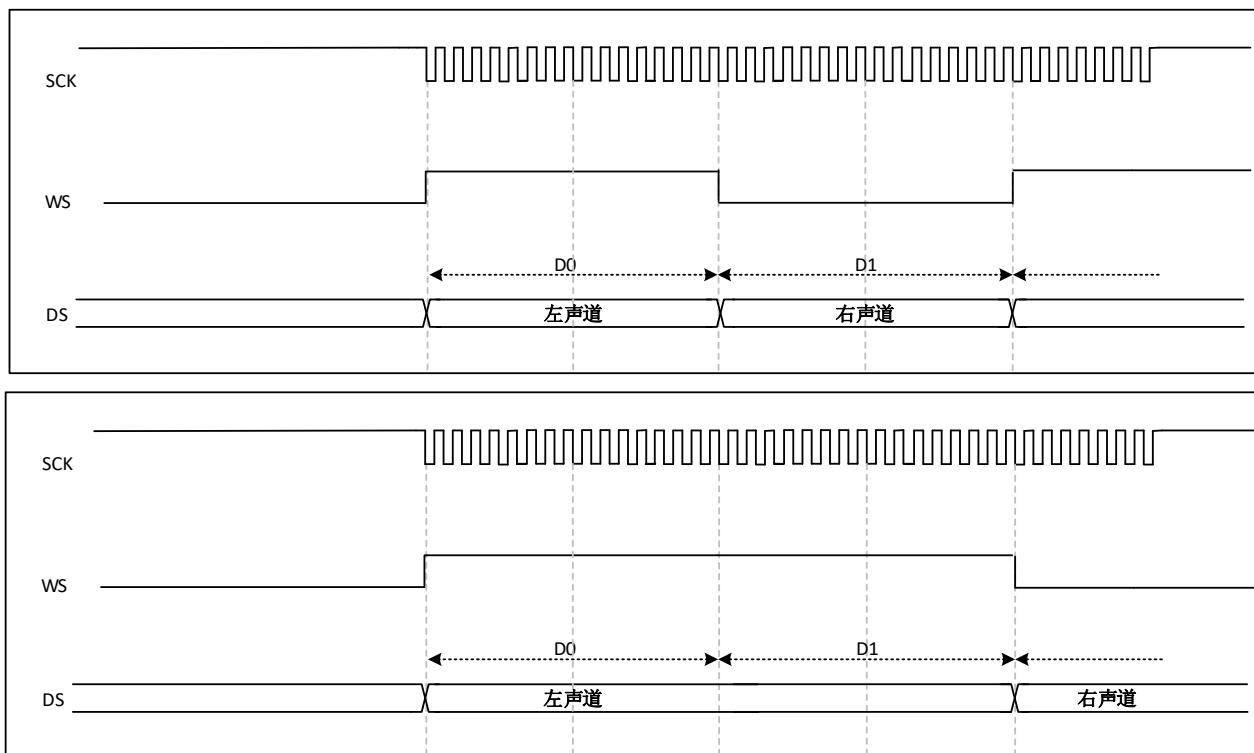


图 25-21 MSB 对齐 16 位或 32 位全精度, CKPOL = 1

发送方在时钟信号的下降沿改变数据; 接收方是在上升沿读取数据。

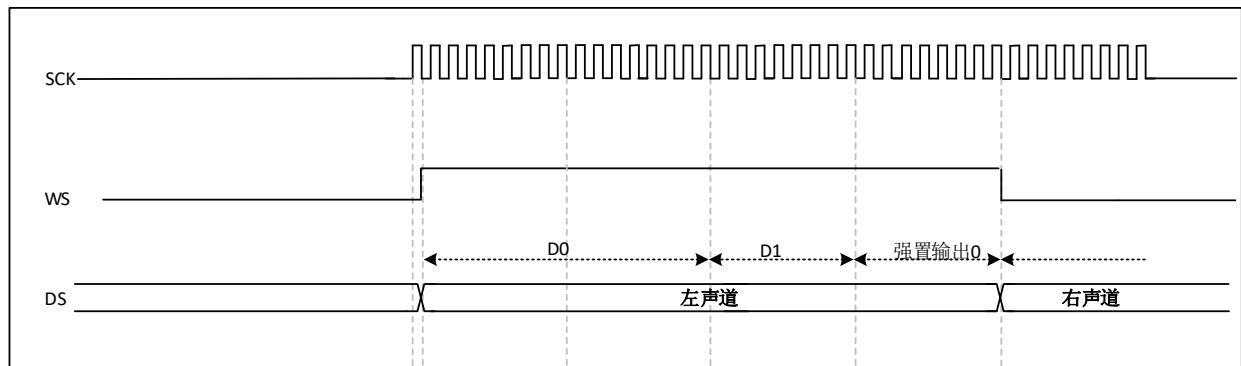


图 25-22 MSB 对齐 24 位数据, CKPOL = 0

当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

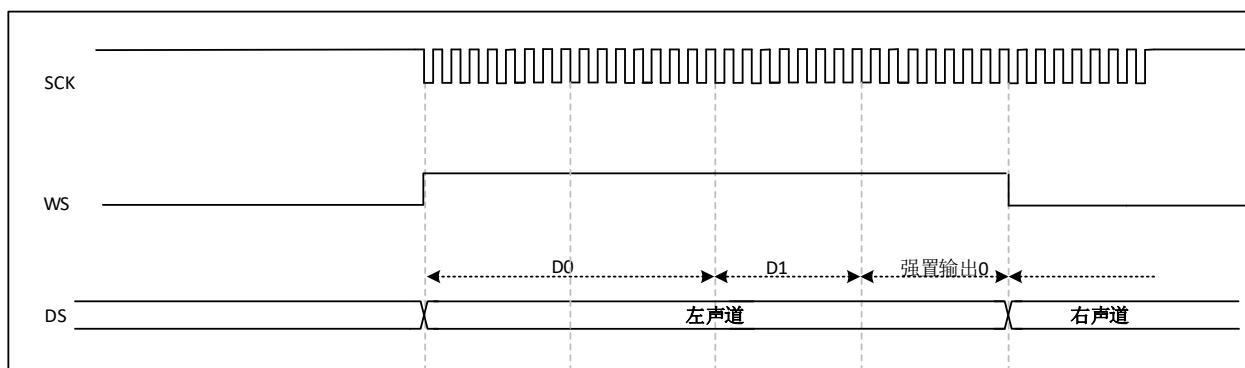


图 25-23 MSB 对齐 24 位数据, CKPOL = 1

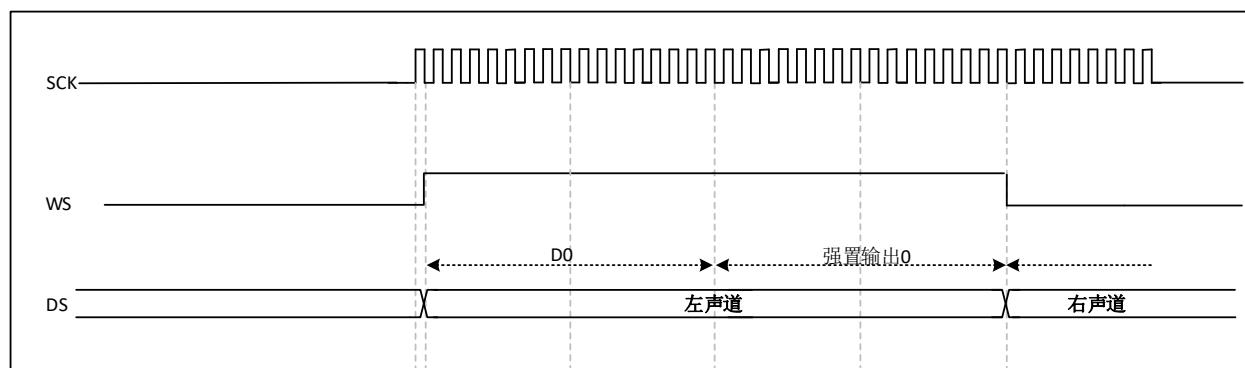


图 25-24 MSB 对齐 16 位数据扩展到 32 位包帧, CKPOL = 0

当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

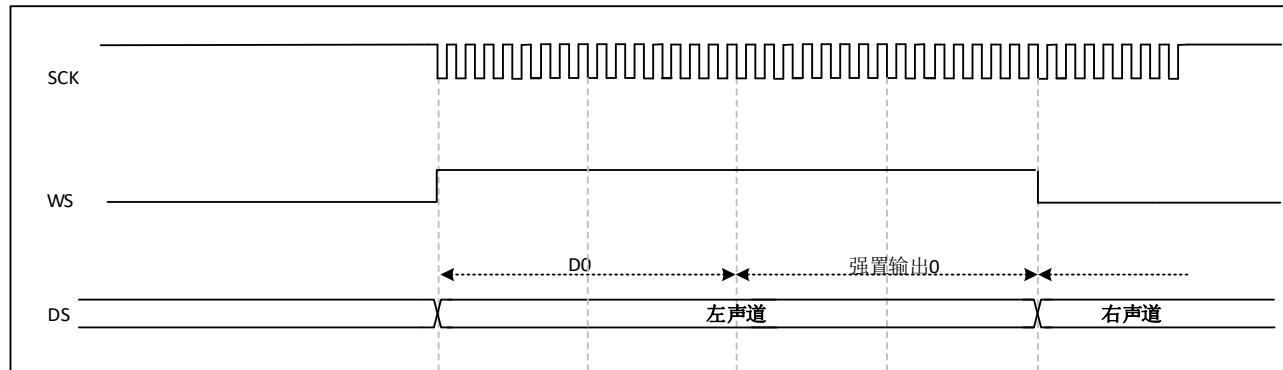


图 25-25 MSB 对齐 16 位数据扩展到 32 位包帧, CKPOL = 1

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。

### 25.3.2.3. LSB 对齐标准

此标准与 MSB 对齐标准类似(在 16 位或 32 位全精度帧格式下无区别)。

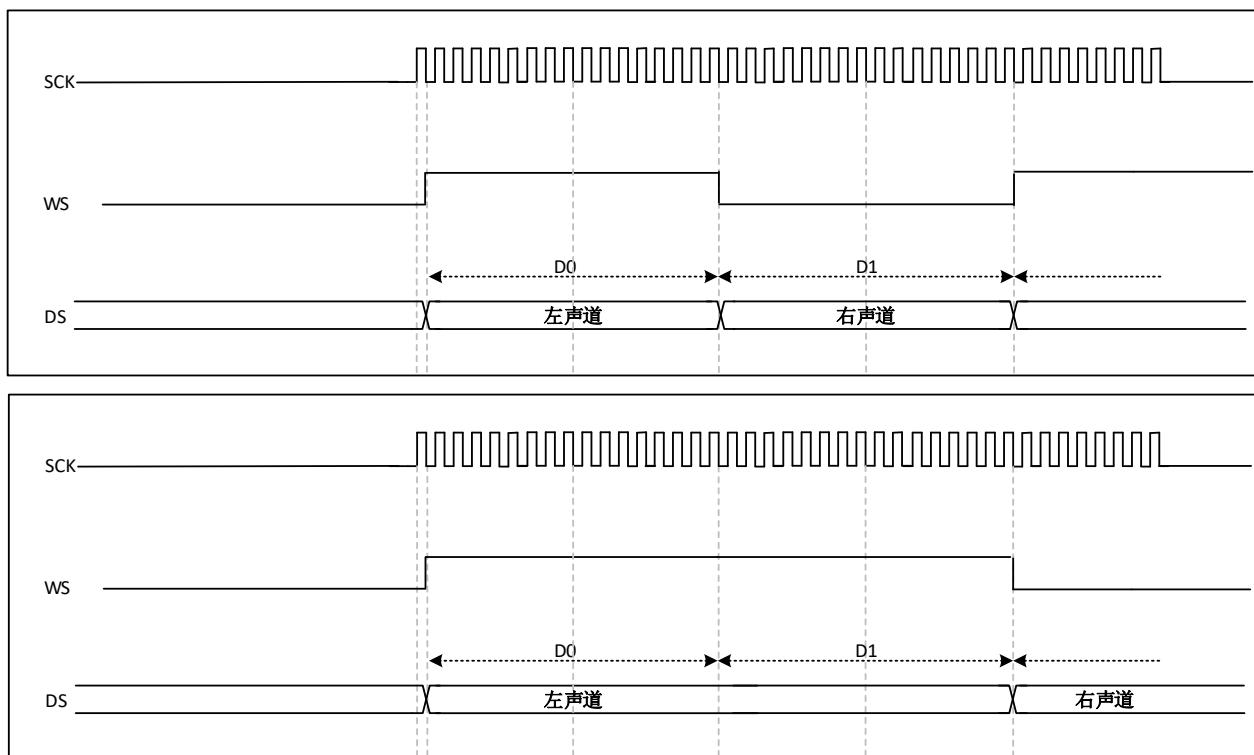


图 25-26 LSB 对齐 16 位或 32 位全精度, CKPOL = 0

当 CKPOL=1 时，发送方也在时钟信号(CK)的下降沿改变数据

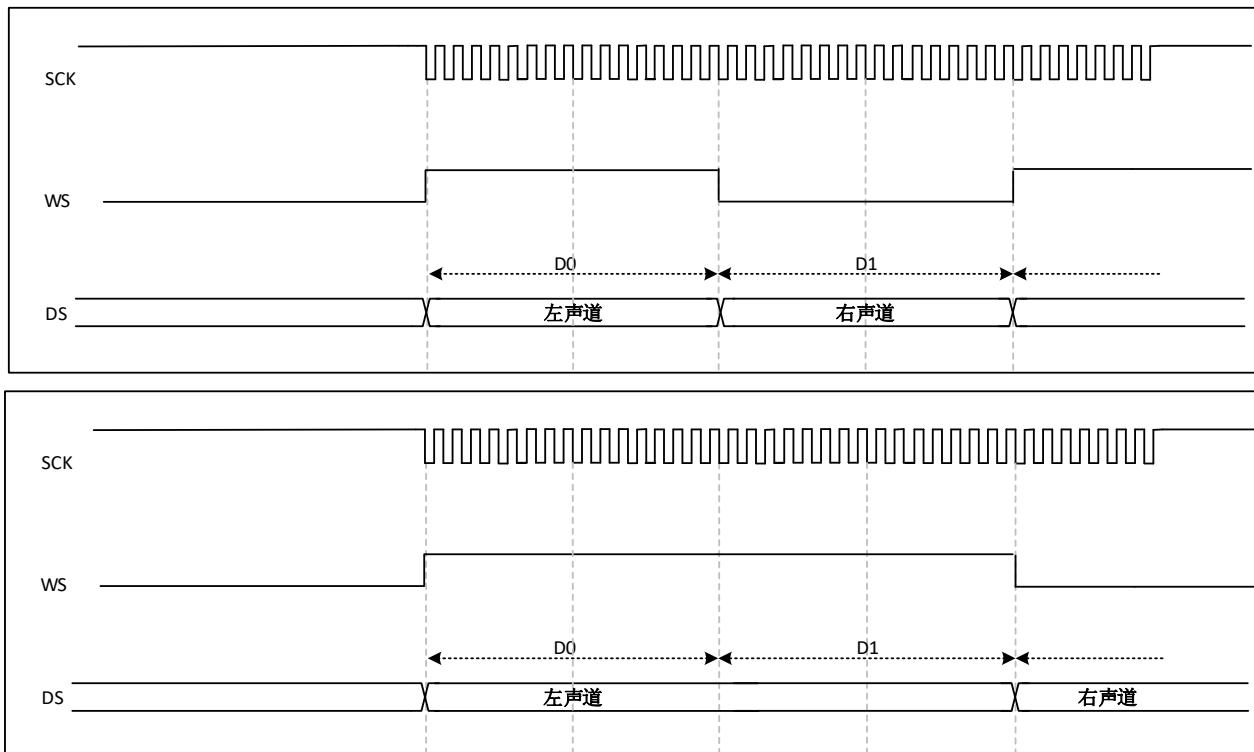


图 25-27 LSB 对齐 16 位或 32 位全精度, CKPOL = 1

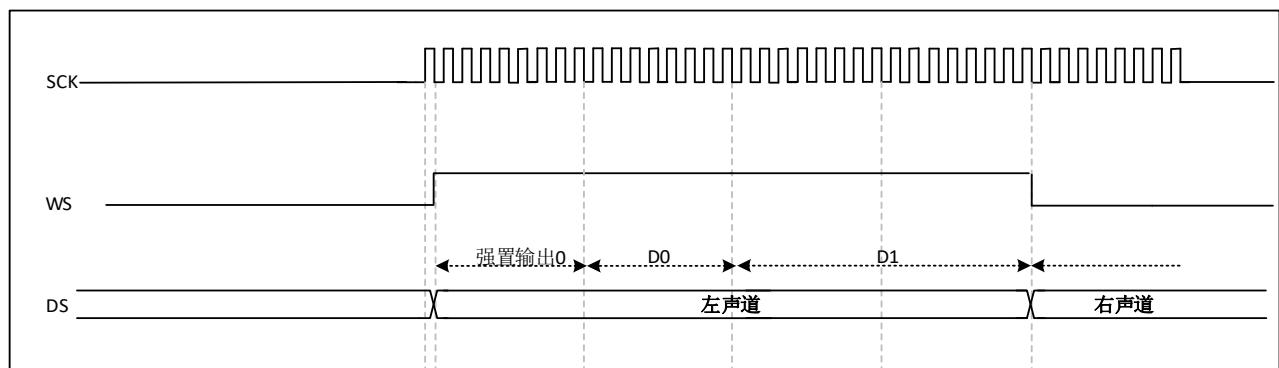


图 25-28 LSB 对齐 24 位数据, CKPOL = 0

当 CKPOL=1 时, 发送方也在时钟信号(CK)的下降沿改变数据

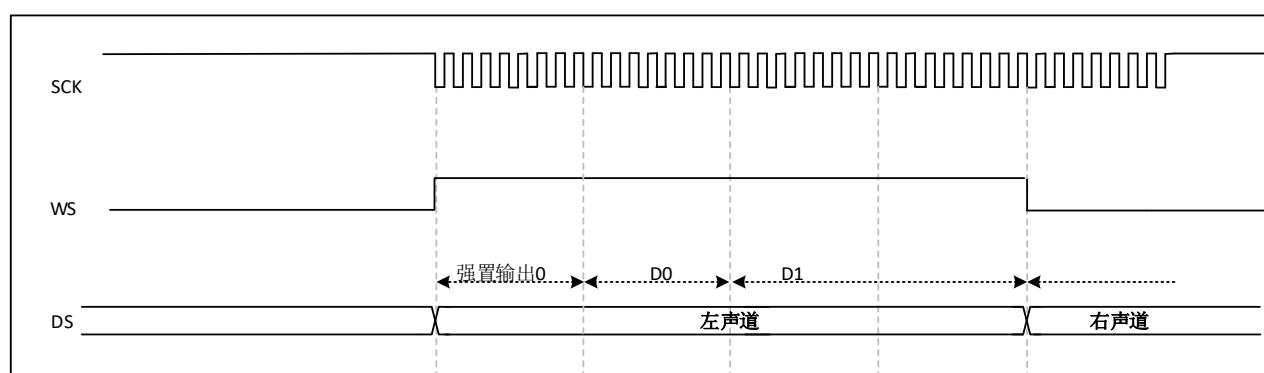


图 25-29 LSB 对齐 24 位数据, CKPOL = 1

■ 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。操作流程如下图所示。

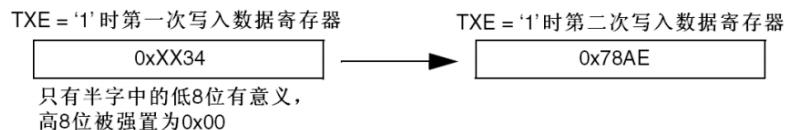


图 25-30 在发送模式下

要求发送 0x3478AE 的操作

■ 在接收模式下

如果要接收数据 0x3478AE，需要在 2 个连续的 RXNE 事件发生时，分别对寄存器 SPI\_DR 进行 1 次读操作。

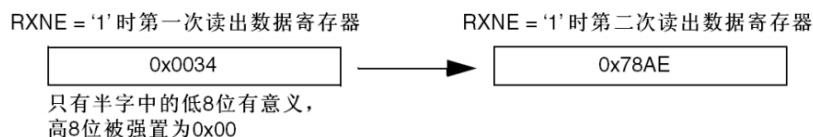


图 25-31 在接收模式下

要求接收 0x3478AE 的操作

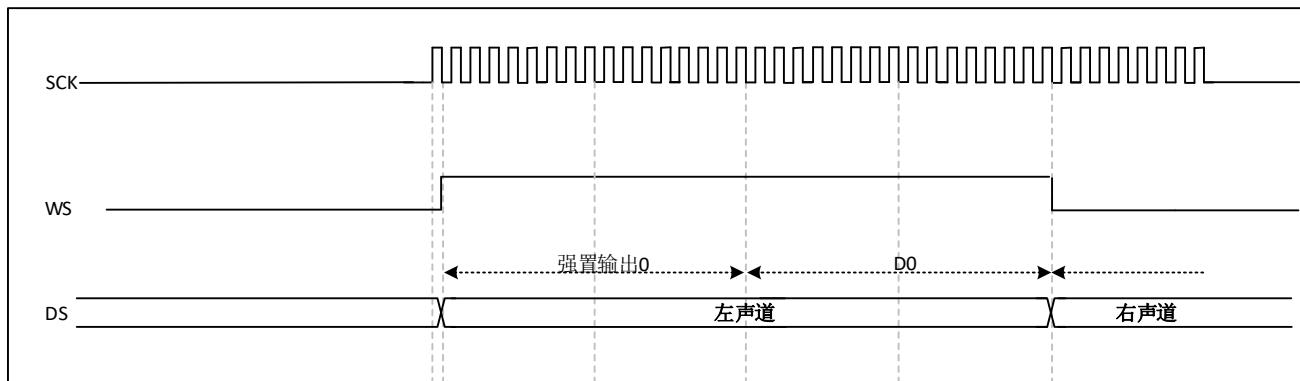


图 25-32 LSB 对齐 16 位数据扩展到 32 位包帧, CKPOL = 0

当 CKPOL=1，发送方也在时钟信号(CK)的下降沿改变数据

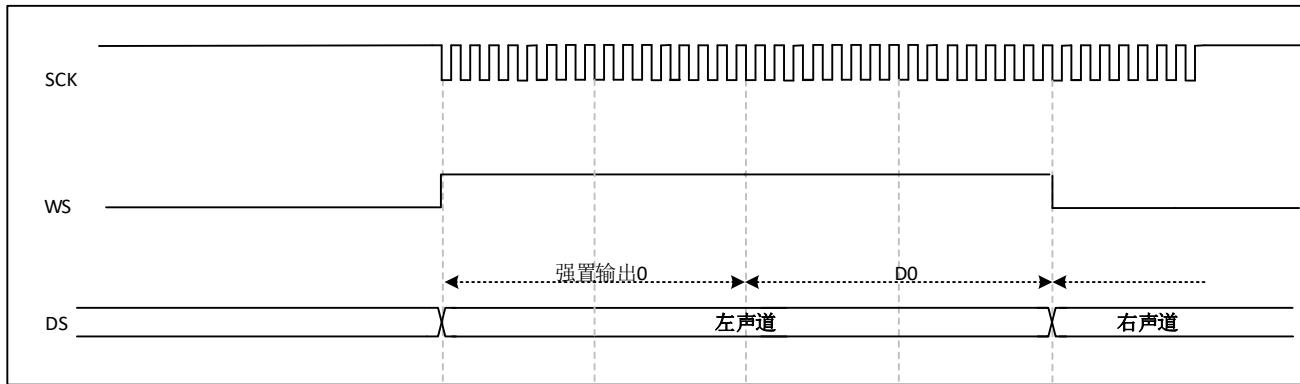


图 25-33 LSB 对齐 16 位数据扩展到 32 位包帧，CKPOL = 1

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)，需要的操作如下图所示。

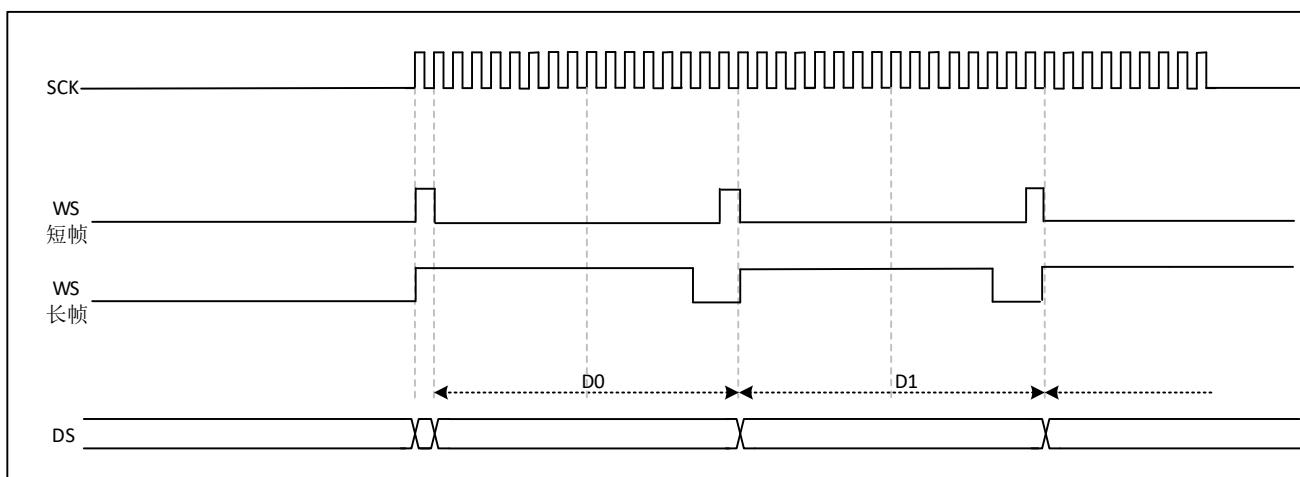
只需操作一次 SPI\_DR

0X76A3

在发送时，如果 TXE 为 ‘E，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。

#### 25.3.2.4. PCM 标准

在 PCM 标准下，不存在声道选择的信息。PCM 标准有 2 种可用的帧结构，短帧或者长帧，可以通过设置寄存器 SPI\_I2SCFGR 的 PCMSYNC 位来选择。



当 CKPOL=1 时，发送方也在时钟信号(CK)的上升沿改变数据

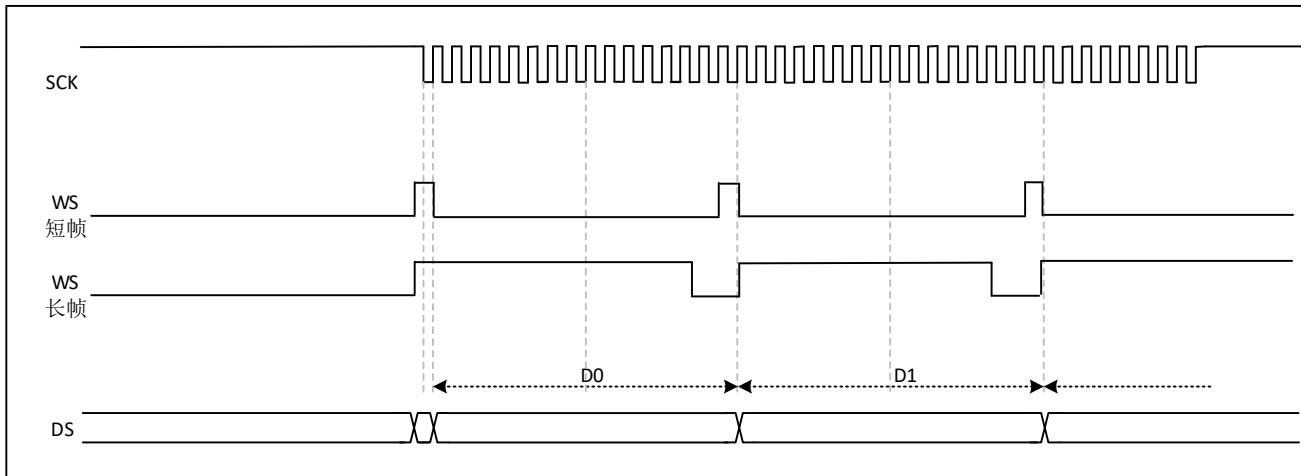
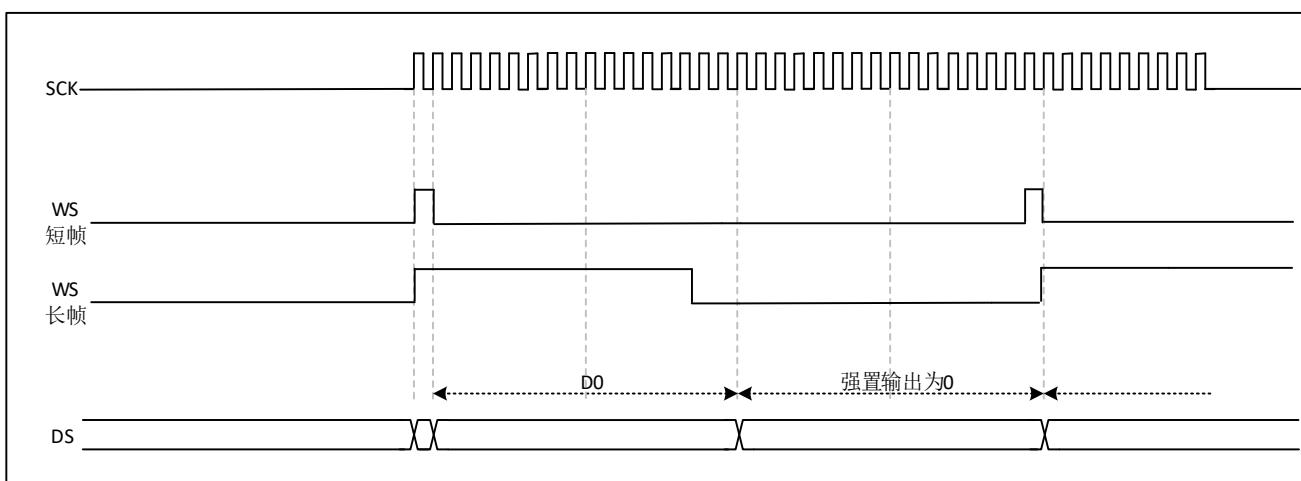


图 25-34 PCM 标准波形(16 位)

对于长帧，主模式下，用来同步的 WS 信号有效的时间固定为 13 位。

对于短帧，用来同步的 WS 信号长度只有 1 位。



当 CKPOL=1 时，发送方也在时钟信号(CK)的上升沿改变数据

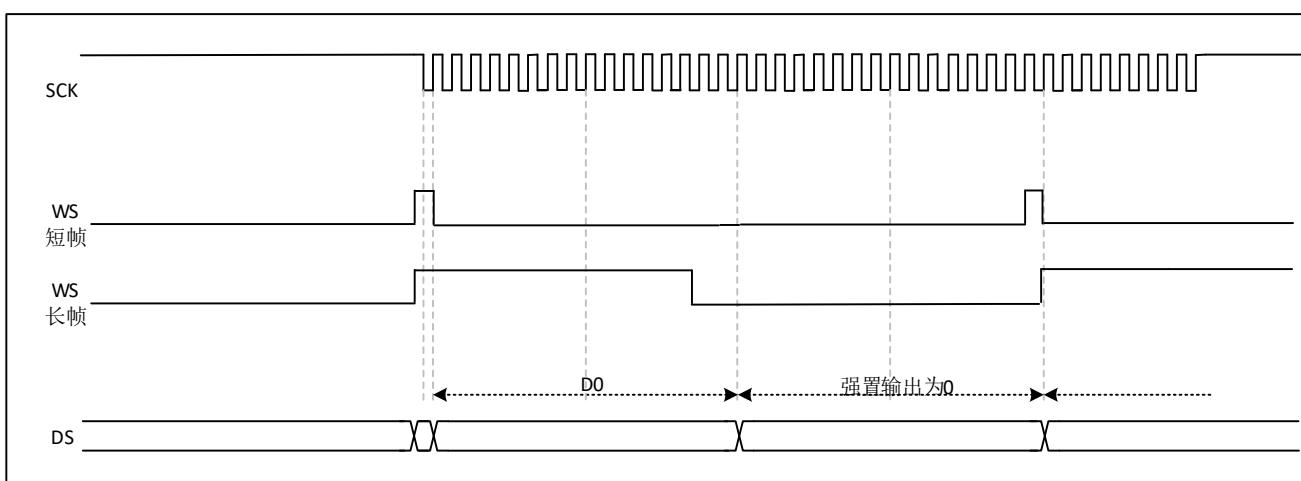


图 25-35 PCM 标准波形(16 位扩展到 32 位包帧)

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。

### 25.3.3. 时钟

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。

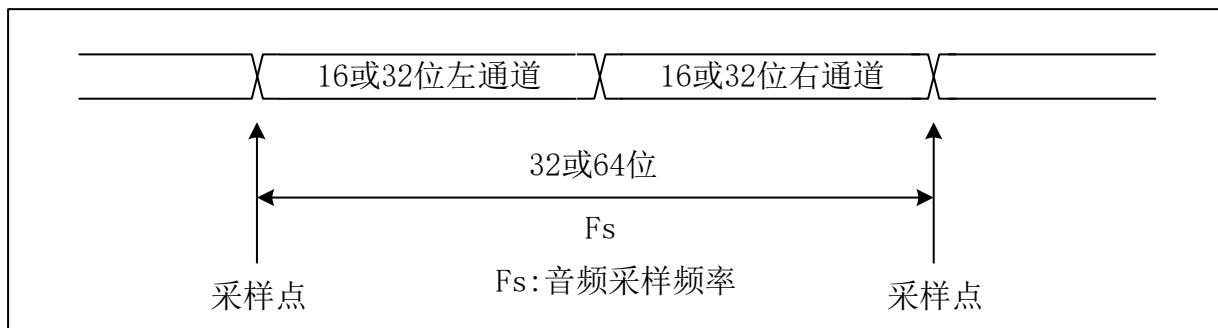
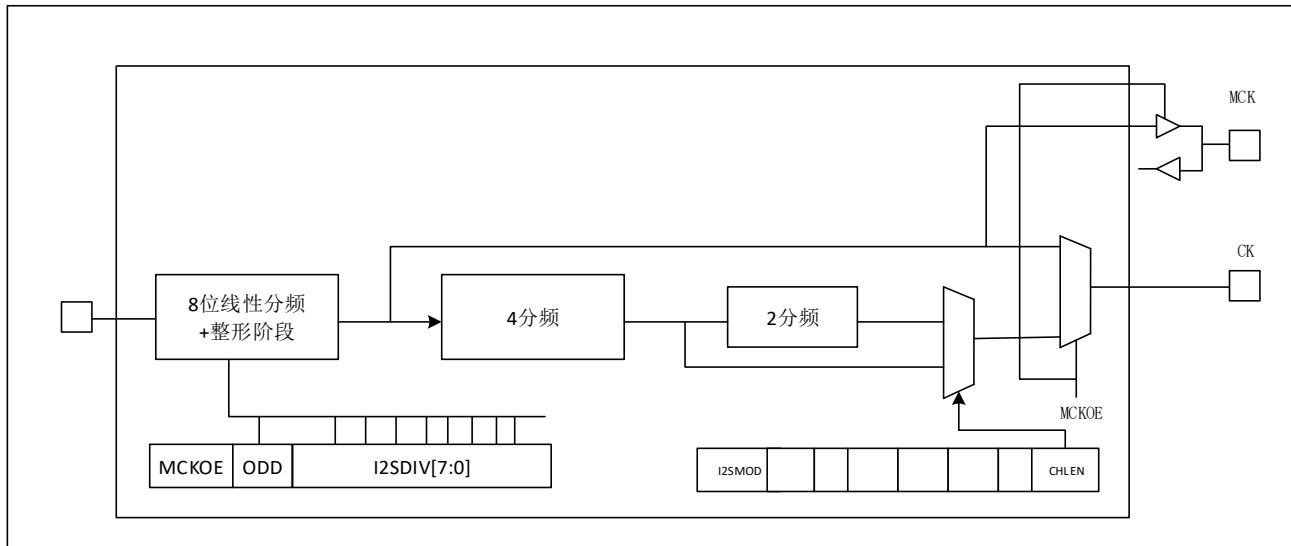


图 25-36 音频采样定义

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。

#### I2S 时钟发生器结构



音频的采样频率可以是 96kHz、48kHz、44.1kHz、32kHz、22.05kHz、16kHz、11.025kHz 或者 8kHz(或任何此范围内的数值)。为了获得需要的频率，需按照以下公式设置线性分频器而获得，当需要生成主时钟时(寄存器 SPI\_I2SPR 的 MCKOE 位为 '1')：

声道的帧长为 16 位时， $F_s = I2S \times CLK / [(16 \times 2) \times ((2 \times I2SDIV) + ODD) \times 8]$

声道的帧长为 32 位时， $F_s = I2S \times CLK / [(32 \times 2) \times ((2 \times I2SDIV) + ODD) \times 4]$

当关闭主时钟时(MCKOE 位为 '0' 为 'K':

声道的帧长为 16 位时,  $F_s = I2SxCLK / [(16 * 2) * ((2 * I2SDIV) + ODD)]$

声道的帧长为 32 位时,  $F_s = I2SxCLK / [(32 * 2) * ((2 * I2SDIV) + ODD)]$

## 25.3.4. 发送与接收

### 25.3.4.1. 主模式

设置 I2S 工作在主模式, 串行时钟由引脚 CK 输出, 字选信号由引脚 WS 产生。可以通过设置寄存器 SPI\_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟(MCK)。

#### ■ 发送流程

当写入 1 个半字(16 位)的数据至发送缓存, 发送流程开始。

假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时, 标志位 TXE 置 '1', 这时, 要把对应右声道的数据写入发送缓存。标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。标志位 CHSIDE 的值在 TXE 为 '1' 时更新, 因此它在 TXE 为 '1' 时有意义。在先左声道后右声道的数据都传输完成后, 才能被认为是一个完整的数据帧。不可以只传输部分数据帧, 如仅有左声道的数据。

当发出第一位数据的同时, 半字数据被并行地传送至 16 位移位寄存器, 然后后面的位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓存移至移位寄存器时, 标志位 TXE 置为 '1', 如果寄存器 SPI\_CR2 的 TXIE 位为 '1', 则产生中断。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。

#### ■ 接收流程

无论何种数据和声道长度, 音频数据总是以 16 位包的形式接收。即每次填满接收缓存后, 标志位 RXNE 置 '1', 如

果寄存器 SPI\_CR2 的 RXIE 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。

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; 在传输期间 BSY 标志始终为低。

### 25.3.4.2. 从模式

在从模式下，I2S 可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要 I2S 接口提供时钟。时钟信号和 WS 信号都由外部主 I2S 设备提供，连接到相应的引脚上。因此用户无需配置时钟。

#### ■ 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。设置 I2SE 位为 ‘1’ 的时间，应当比 CK 引脚上的主 I2S 时钟信号早至少 2 个 PCLK 时钟周期。

当发出第一位数据的时候，半字数据并行地通过 I2S 内部总线传输至 16 位移位寄存器，然后其它位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓冲器传送至移位寄存器时，标志位 TXE 置 ‘1’，如果寄存器 SPI\_CR2 的 TXEIE 位为 ‘1’，则产生中断。

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。

#### ■ 接收流程

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收，即每次填满接收缓存，标志位 RXNE 置 ‘1’，如果寄存器 SPI\_CR2 的 RXNEIE 位为 ‘1’，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要 1 次或者 2 次传输数据至接收缓冲器的过程。每次接收到数据(将要从 SPI\_DR 读出)以后即更新 CHSIDE，它对应 I2S 单元产生的 WS 信号。读取 SPI\_DR 寄存器，将清除 RXNE 位。

在还没有读出前一个接收到的数据，又接收到新数据时，即产生上溢，并设置标志位 OVR 为 ‘1’；如果寄存器 SPI\_CR2 的 ERRIE 位为 ‘1’，则产生中断，指示发生了错误。

### 25.3.5. 标志位

#### 25.3.5.1. 状态标志

有 3 个状态标志位供用户监控 I2S 总线的状态。

- 忙标志位(BSY)

BSY 标志由硬件设置与清除(写入此位无效果)，该标志位指示 I2S 通信层的状态。该位为 ‘1’ 时表明 I2S 通讯正在进行中，但有一个例外：主接收模式(I2SCFG=11)下，在接收期间 BSY 标志始终为低。在软件要关闭 SPI 模块之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。当传输开始时，BSY 标志被置为 ‘志被，除非 I2S 模块处于主接收模式。

下述情况时，该标志位被清除：

- 当传输结束时(除了主发送模式，这种模式下通信是连续的)；
- 当关闭 I2S 模块时。
- 当通信是连续的时候：
  - 在主发送模式时，整个传输期间，BSY 标志始终为高；
  - 在从模式时，每个数据项传输之间，BSY 标志在 1 个 I2S 时钟周期内变低。

- 发送缓存空标志位(TXE)

该标志位为 ‘1’ 表示发送缓冲器为空，可以对发送缓冲器写入新的待发送数据。在发送缓冲器中已有数据时，标志位清 ‘0’。在 I2S 被关闭时(I2SE 位为 ‘0’ )，该标志位也为 ‘0’。

- 接收缓存非空标志位(RXNE)

该标志位置 ‘1’ 表示在接收缓存里有接收到的有效数据。在读取寄存器 SPI\_DR 时，该位清 ‘0’。

- 声道标志位(CHSIDE)

在发送模式下，该标志位在 TXE 为高时刷新，指示从 SD 引脚上发送的数据所在的声音。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音。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。

### 25.3.5.2. 错误标志

I2S 单元有 2 个错误标志位。

- 下溢标志位(UDR)

在从发送模式下，如果数据传输的第一个时钟边沿到达时，新的数据仍然没有写入 SPI\_DR 寄存器，该标志位会被置 '1'。在寄存器 SPI\_I2SCFGR 的 I2SMOD 位置 '1' 后，该标志位才有效。如果寄存器 SPI\_CR2 的 ERRIE 位为 '1'，就会产生中断。通过对寄存器 SPI\_SR 进行读操作来清除该标志位。

- 上溢标志位(OVR)

如果还没有读出前一个接收到的数据时，又接收到新的数据，即产生上溢，该标志位置 '1'，如果寄存器 SPI\_CR2 的 ERRIE 位为 '1'，则产生中断指示发生了错误。这时，接收缓存的内容，不会刷新为从发送设备送来的 newData。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。

### 25.3.6. 中断

表 25-2 I2S 中断请求

中断事件	事件标志位	使能标志位
发送缓冲器空标志位	TXE	TXEIE
接收缓冲器非空标志位	RXNE	RXNEIE
下溢标志位	OVR	ERRIE
上溢标志位	UDR	

## 25.4. 寄存器描述

SPI1 寄存器基地址: 0x4001 3000

SPI2 寄存器基地址: 0x4000 3800

SPI3 寄存器基地址: 0x4000 3C00

### 25.4.1. SPI\_CR1 寄存器

Address offset:0x00

Reset value:0x0000

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

BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	双向数据模式使能 (Bidirectional data mode enable) 0: 选择“双线双向”模式; 1: 选择“单线双向”模式。 注: I2S 模式下不使用。
14	BIDIOE	RW	0	双向模式下的输出使能 (Output enable in bidirectional mode) 和 BIDIMODE 位一起决定在“单线双向”模式下数据的输出方向 0: 输出禁止(只收模式); 1: 输出使能(只发模式)。 这个“单线”数据线在主设备端为 MOSI 引脚, 在从设备端为 MISO 引脚。 注: I2S 模式下不使用。
13	CRCEN	RW	0	硬件 CRC 校验使能 (Hardware CRC calculation enable) 0: 禁止 CRC 计算; 1: 启动 CRC 计算。 注: 只有在禁止 SPI 时(SPE=0), 才能写该位, 否则出错。 该位只能在全双工模式下使用。 注: I2S 模式下不使用。
12	CRCNEXT	RW	0	下一个发送 CRC (Transmit CRC next) 0: 下一个发送的值来自发送缓冲区。 1: 下一个发送的值来自发送 CRC 寄存器。 注: 在 SPI_DR 寄存器写入最后一个数据后应马上设置该位。 注: I2S 模式下不使用。
11	DFF	RW	0	数据帧格式 (Data frame format) 0: 使用 8 位数据帧格式进行发送/接收; 1: 使用 16 位数据帧格式进行发送/接收。 注: 只有当 SPI 禁止(SPE=0)时, 才能写该位, 否则出错。 注: I2S 模式下不使用。
10	RXONLY	RW	0	只接收 (Receive only) 该位和 BIDIMODE 位一起决定在“双线双向”模式下的传输方向。在多个从设备的配置中, 在未被访问的从设备上该位被置 1, 使得只有被访问的从设备有输出, 从而不会造成数据线上数据冲突。 0: 全双工(发送和接收); 1: 禁止输出(只接收模式)。 注: I2S 模式下不使用。
9	SSM	RW	0	软件从设备管理 (Software slave management) 当 SSM 被置位时, NSS 引脚上的电平由 SSI 位的值决定。 0: 禁止软件从设备管理; 1: 启用软件从设备管理。 注: I2S 模式下不使用。
8	SSI	RW	0	内部从设备选择 (Internal slave select) 该位只在 SSM 位为 ‘1’ 时有意义。它决定了 NSS 上的电平, 在 NSS 引脚上的 I/O 操作无效。 注: I2S 模式下不使用。
7	LSBFIRST	RW	0	帧格式 (Frame format) 0: 先发送 MSB; 1: 先发送 LSB。 注: 当通信正在进行的时候, 不能修改该位。 注: 当通信在进行时不能改变该位的值, 若软件在通信进行时修改通信将发生错误。 注: I2S 模式下不使用。
6	SPE	RW	0	SPI 使能 (SPI enable) 0: 禁止 SPI 设备;

Bit	Name	R/W	Reset Value	Function
				1: 开启 SPI 设备。 注: I2S 模式下不使用。
5:3	BR[2:0]	RW	0	波特率控制 (Baud rate control) 000: fPCLK/2 001: fPCLK/4 010: fPCLK/8 011: fPCLK/16 100: fPCLK/32 101: fPCLK/64 110: fPCLK/128 111: fPCLK/256 注: 当通信正在进行的时候, 不能修改该位。 注: 当通信在进行时不能改变该位的值, 若软件在通信进行时修改通信将发生错误。 注意: I2S 模式下不使用。
2	MSTR	RW	0	主设备选择 (Master selection) 0: 配置为从设备; 1: 配置为主设备。 注: 当通信正在进行的时候, 不能修改该位。 注: I2S 模式下不使用。
1	CPOL	RW	0	时钟极性 (Clock polarity) 0: 空闲状态时, SCK 保持低电平; 1: 空闲状态时, SCK 保持高电平。 注: 当通信正在进行的时候, 不能修改该位。 注: 当通信在进行时不能改变该位的值, 若软件在通信进行时修改通信将发生错误。 注: I2S 模式下不使用。
0	CPHA	RW	0	时钟相位 (Clock phase) 0: 数据采样从第一个时钟边沿开始; 1: 数据采样从第二个时钟边沿开始。 注: 当通信正在进行的时候, 不能修改该位。 注: 当通信在进行时不能改变该位的值, 若软件在通信进行时修改通信将发生错误。 注: I2S 模式下不使用。

#### 25.4.2. SPI\_CR2 寄存器

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLVFM	LDMA_TX	LDMA_RX	FRXTH	RES				TXEIE	RXNEIE	ERRIE	CLRTXFF_O	RES	SSOE	TXDMAE_N	RXDMAE_N
RW	RW	RW	RW					RW	RW	RW	W		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	SLVFM	RW	0	Slave fast mode enable 0: slave normal mode, slave 模式支持最快 SPI clock 的速度小于 pclk/4 1: slave fast mode, 可支持 slave 模式下 SPI clock 速度可到 pclk/4 注: 当 SPI clock 的速度小于 pclk/4 时, 一定不能设定该寄存器位。 注: I2S 模式下不使用。
14	LDMA_TX	RW	0	最后一个 DMA 传输 (发送) 该位用来定义 DMA 发送的总数量是偶数或者奇数。该功能仅在 data length=8, TXDMAEN 位被置位, 且 16 位写访问时才有效。该位需要在 SPE=0 时才能被写。 0: 要发送的数据总量是偶数 1: 要发送的数据总量是奇数

Bit	Name	R/W	Reset Value	Function
				注： I2S 模式下不使用。
13	LDMA_RX	RW	0	<p>最后一个 DMA 传输（接收）</p> <p>该位用来定义 DMA 接收的总数量是偶数或者奇数。该功能仅在 data length=8, RXDMAEN 位被置位，且 16 位写访问时才有效。该位需要在 SPE=0 时才能被写。</p> <p>0: 要接收的数据总量是偶数 1: 要接收的数据总量是奇数 注： I2S 模式下不使用。</p>
12	FRXTH	RW	0	<p>FIFO 接收阈值</p> <p>该位用来设定触发 RXNE 事件的 RXFIFO 阈值。</p> <p>0: 如果 FIFO level 大于或者等于 1/2 (16-bit) , 产生 RXNE 1: 如果 FIFO level 大于或者等于 1/4 (8-bit) , 产生 RXNE 注： I2S 模式下不使用。</p>
11:8	Reserved	RES	-	Reserved
7	TXEIE	RW	0	<p>发送缓冲区空中断使能 (Tx buffer empty interrupt enable)</p> <p>0: 禁止 TXE 中断； 1: 允许 TXE 中断，当 TXE 标志置位为 ‘1’ 时产生中断请求。</p>
6	RXNEIE	RW	0	<p>接收缓冲区非空中断使能 (RX buffer not empty interrupt enable)</p> <p>0: 禁止 RXNE 中断； 1: 允许 RXNE 中断，当 RXNE 标志置位时产生中断请求。</p>
5	ERRIE	RW	0	<p>错误中断使能 (Error interrupt enable)</p> <p>当错误(CRCERR、OVR、MODF)产生时，该位控制是否产生中断</p> <p>0: 禁止错误中断； 1: 允许错误中断。</p>
4	CLRTXFIFO	W		<p>清空 FIFO</p> <p>0: 不对 FIFO 产生影响 1: 清空 FIFO 软件写 1, 硬件自动清 0 spi 使能后写该位无效</p>
3	Reserved	RES	-	Reserved
2	SSOE	RW	0	<p>SS 输出使能 (SS output enable)</p> <p>0: 禁止在主模式下 SS 输出，该设备可以工作在多主设备模式； 1: 设备开启时，开启主模式下 SS 输出，该设备不能工作在多主设备模式。 注： I2S 模式下不使用。</p>
1	TXDMAEN	RW	0	<p>发送缓冲区 DMA 使能 (Tx buffer DMA enable)</p> <p>当该位被设置时， TXE 标志一旦被置位就发出 DMA 请求</p> <p>0: 禁止发送缓冲区 DMA； 1: 启动发送缓冲区 DMA。</p>
0	RXDMAEN	RW	0	<p>接收缓冲区 DMA 使能 (Rx buffer DMA enable)</p> <p>当该位被设置时， RXNE 标志一旦被置位就发出 DMA 请求</p> <p>0: 禁止接收缓冲区 DMA； 1: 启动接收缓冲区 DMA。</p>

### 25.4.3. SPI\_SR 寄存器

Address offset:0x08

Reset value:0x0002

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

RES.	FTLVL	FRLVL	RES	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	R	R		R	R	R	RCW0	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15:13	Reserved	RES	-	Reserved
12:11	FTLVL	R	0	<p>FIFO transmission level 这些位由硬件设置和清除 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO 已满 (当 FIFO 阀值大于 1/2 时, 视为已满) 注意: 此位不能用于 I2S 模式</p>
10:9	FRLVL	R	0	<p>FIFO reception level 这些位由硬件设置和清除 00: FIFO 空 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO 已满 注意: 此位不能用于 I2S 模式</p>
8	Reserved	RES	-	Reserved
7	BSY	R	0	<p>忙标志 (Busy flag) 0: SPI 不忙; 1: SPI 正忙于通信, 或者发送缓冲非空。 该位由硬件置位或者复位。</p>
6	OVR	R	0	<p>溢出标志 (Overrun flag) 0: 没有出现溢出错误; 1: 出现溢出错误。 该位由硬件置位, 由软件序列复位。</p>
5	MODF	RW	0	<p>模式错误 (Mode fault) 0: 没有出现模式错误; 1: 出现模式错误。 该位由硬件置位, 由软件序列复位。 注: I2S 模式下不使用。</p>
4	CRCERR	RCW0	0	<p>CRC 错误标志 (CRC error flag) 0: 收到的 CRC 值和 SPI_RXCRCR 寄存器中的值匹配; 1: 收到的 CRC 值和 SPI_RXCRCR 寄存器中的值不匹配。 该位由硬件置位, 由软件写 '0' 而复位。 注: I2S 模式下不使用。</p>
3	UDR	R	0	<p>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。</p>
2	CHSIDE	R	0	<p>声道 (Channel side) 0: 需要传输或者接收左声道; 1: 需要传输或者接收右声道。 注: 在 SPI 模式下不使用。在 PCM 模式下无意义。</p>
1	TXE	R	1	<p>发送缓冲为空 (Transmit buffer empty) 0: 发送缓冲非空; 1: 发送缓冲为空。</p>

Bit	Name	R/W	Reset Value	Function
0	RXNE	R	0	接收缓冲非空 (Receive buffer not empty) 0: 接收缓冲为空; 1: 接收缓冲非空。

#### 25.4.4. SPI\_DR 寄存器

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DR[15:0]	RW	0	数据寄存器 (Data register) 待发送或者已经收到的数据 数据寄存器对应两个缓冲区：一个用于写(发送缓冲)；另外一个用于读(接收缓冲)。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。 对 SPI 模式的注释：根据 SPI_CR1 的 DFF 位对数据帧格式的选择，数据的发送和接收可以是 8 位或者 16 位的。为保证正确的操作，需要在启用 SPI 之前就确定好数据帧格式。 对于 8 位的数据，缓冲器是 8 位的，发送和接收时只会用到 SPI_DR[7:0]。在接收时，SPI_DR[15:8]被强制为 0。 对于 16 位的数据，缓冲器是 16 位的，发送和接收时会用到整个数据寄存器，即 SPI_DR[15:0]。

#### 25.4.5. SPI\_CRCPR 寄存器

Address offset:0x10

Reset value:0x0007

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

Bit	Name	R/W	Reset Value	Function
15:0	CRCPOLY[15:0]	RW	0	CRC 多项式寄存器 (CRC polynomial register) 该寄存器包含了 CRC 计算时用到的多项式。 其复位值为 0x0007，根据应用可以设置其他数值。 注：在 I2S 模式下不使用。

#### 25.4.6. SPI\_RXCRCR 寄存器

Address offset:0x14

Reset value:0x0000

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

RxCRC[15:0]																
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15:0	RxCRC[15:0]	R	0	<p>接收 CRC 寄存器 在启用 CRC 计算时，RxCRC[15:0]中包含了依据收到的字节计算的 CRC 数值。当在 SPI_CR1 的 CRCEN 位写入 ‘1’ 时，该寄存器被复位。CRC 计算使用 SPI_CRCPR 中的多项式。</p> <p>当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行；当数据帧格式为 16 位时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的标准。</p> <p>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。</p>

#### 25.4.7. SPI\_TXCRCR 寄存器

Address offset:0x18

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	TxCRC[15:0]	R	0	<p>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。</p> <p>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。</p> <p>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。</p>

#### 25.4.8. SPI\_I2S\_CFGR 寄存器

Address offset:0x1c

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.					I2SMOD	I2SE	I2SCFG	PCM_SYNC		RES.	I2SSSTD	CKPO_L	DATL_EN	CHLN	
					RW	RW	RW	RW		RW	RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
15:12	Reserved	RES	-	Reserved

Bit	Name	R/W	Reset Value	Function
11	I2SMOD	RW	0	I2S 模式选择 (I2S mode selection) 0: 选择 SPI 模式; 1: 选择 I2S 模式。 注: 该位只有在关闭了 SPI 或者 I2S 时才能设置。
10	I2SE	RW	0	I2S 使能 (I2S enable) 0: 关闭 I2S; 1: I2S 使能。 注: 在 SPI 模式下不使用。
9:8	I2SCFG	RW	0	I2S 模式设置 (I2S configuration mode) 00: 从设备发送; 01: 从设备接收; 10: 主设备发送; 11: 主设备接收。 注: 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
7	PCMSYNC	RW	0	PCM 帧同步 (PCM frame synchronization) 0: 短帧同步; 1: 长帧同步。 注: 该位只在 I2SSTD = 11 (使用 PCM 标准)时有意义。 在 SPI 模式下不使用。
6	Reserved	RES	-	Reserved
5:4	I2SSTD	RW	0	I2S 标准选择 (I2S standard selection) 00: I2S 飞利浦标准; 01: 高字节对齐标准 (左对齐); 10: 低字节对齐标准(右对齐); 11: PCM 标准。 注: 为了正确操作, 只有在关闭了 I2S 时才能设置该位。 在 SPI 模式下不使用。
3	CKPOL	RW	0	静止态时钟极性 (Steady state clock polarity) 0: I2S 时钟静止态为低电平; 1: I2S 时钟静止态为高电平。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
2:1	DATLEN	RW	0	待传输数据长度 (Data length to be transferred) 00: 16 位数据长度; 01: 24 位数据长度; 10: 32 位数据长度; 11: 不允许。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。
0	CHLEN	RW	0	声道长度 (每个音频声道的数据位数) (Channel length (number of bits per audio channel)) 0: 16 位宽; 1: 32 位宽。 只有在 DATLEN = 00 时该位的写操作才有意义, 否则声道长度都由硬件固定为 32 位。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。 在 SPI 模式下不使用。

### 25.4.9. SPI\_I2SPR 寄存器

Address offset:0x20

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.						MCK OE	ODD	I2SDIV							
						RW	RW	RW							

Bit	Name	R/W	Reset Value	Function
15:10	Reserved	RES	0	Reserved
9	MCKOE	RW	0	主设备时钟输出使能 (Master clock output enable) 0: 关闭主设备时钟输出; 1: 主设备时钟输出使能。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。 在 SPI 模式下不使用。
8	ODD	RW	0	奇系数预分频 (Odd factor for the prescaler) 0: 实际分频系数 = I2SDIV *2; 1: 实际分频系数 = (I2SDIV * 2)+1。 注: 为了正确操作, 该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。 在 SPI 模式下不使用。
7:0	I2SDIV	RW	0	I2S 线性预分频 (I2S linear prescaler) 禁止设置 I2SDIV [7:0] = 0 或者 I2SDIV [7:0] = 1 注: 为了正确操作, 该位只有在关闭了 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。

### 25.4.10. SPI 寄存器映射

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	ERRIE	CLRTXFIFO	MSTR	CPOL	CPHA	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	SPI_CR2	SLVFM	LDMA_TX	LDMA_RX	FRXTH	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	Res.	SSOE	TXDMAEN	CPOL	CPHA	
	Reset value	0	0	0	0	0			0	0	0	0	0	0	0	0	

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	SPI_SR	Res.	Res.	Res.	FTLV[1:0]				Res.	BSY	OVR	MODEF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value				0	0	0	0	0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SPI_RXCRCR	RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2S_CFGR	Res.	Res.	Res.	I2SMOD	I2SCFG[1:0]	PCMSYNC	0	Res.	I2SSSTD	0	0	CKPOL	DATLEN[1:0]			
	Reset value							0			0	0	0	0	0	0	0
0x20	SPI_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	0	Res.	Res.	0	0	0	0	0	0	0
	Reset value							0			0	0	0	0	0	0	0

# 26. I2C 接口

## 26.1. 简介

I2C(inter-integrated circuit)总线接口连接微控制器和串行 I2C 总线。它提供多主机功能，控制所有 I2C 总线特定的顺序、协议、仲裁和时序。支持标准 (Sm) 、快速 (Fm) 。根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

### 26.1.1. 主要特征

- 并行总线/I2C 总线协议转换器
- 多主机功能：同一接口既可做主设备也可做从设备
- I2C 主设备功能
  - 产生时钟
  - 产生起始和停止信号
- I2C 从设备功能
  - 可编程的 I2C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测 7 位/10 位地址和广播呼叫
- 支持不同的通讯速度
  - 标准速度(高至 100 kHz)
  - 快速(高至 400 kHz)
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I2C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答(ACK)错误
  - 检测到起始和停止错位
  - 过载和欠载 (需先禁止拉长时钟功能)
- 2 个中断向量
  - 1 个事件中断，用于地址/数据通讯成功
  - 1 个错误中断
- 可选的拉长时钟功能

- 具单字节缓冲器的 DMA
- 可配置的 PEC(信息包错误检测)的产生或校验:
  - 发送模式中 PEC 值可以作为最后一个字节传输
  - 用于最后一个接收字节的 PEC 错误校验
- 支持 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 产生/校验
  - 支持地址分辨协议 (ARP)

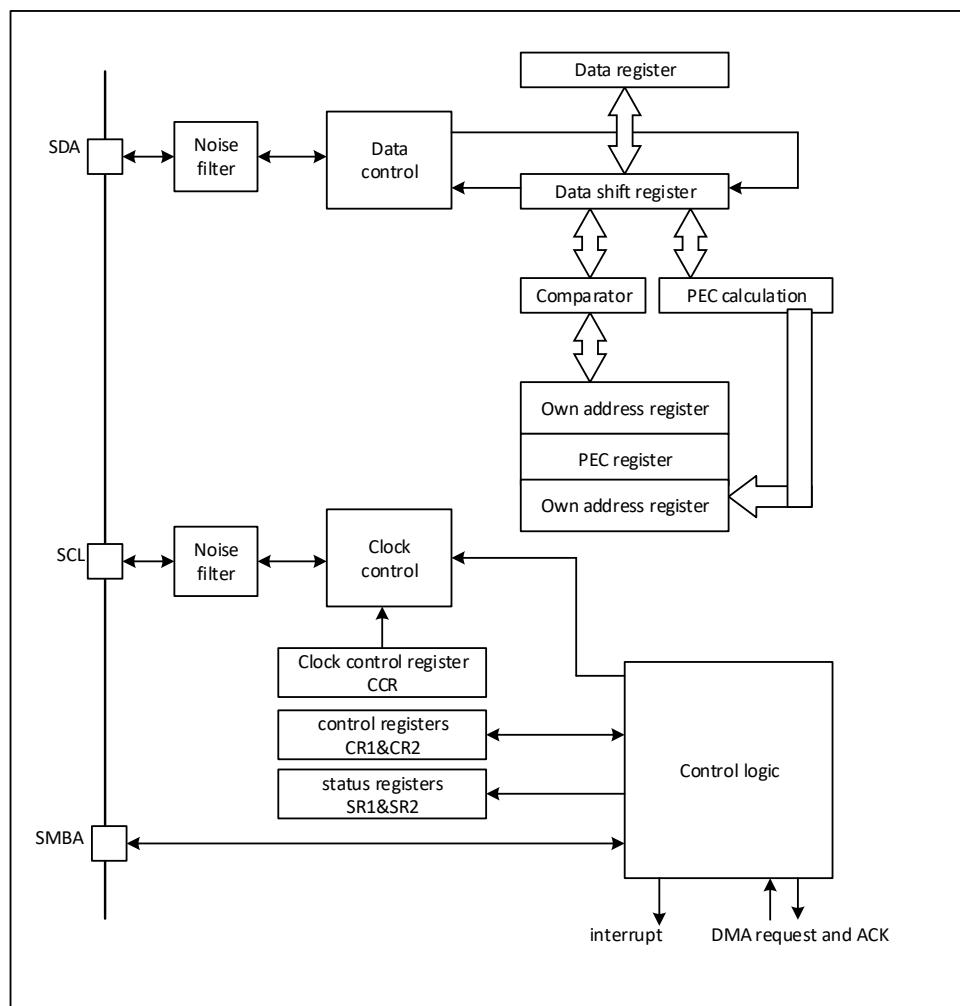


图 26-1 I2C 模块框图

## 26.2. I2C 功能描述

(按内部功能模块, 进行详细描述。包括但不仅限于总线时序, 内部模块及其连接关系, 数据通路, 中断, DMA, 关键状态机, 时钟复位方案等)

### 26.2.1. 简介

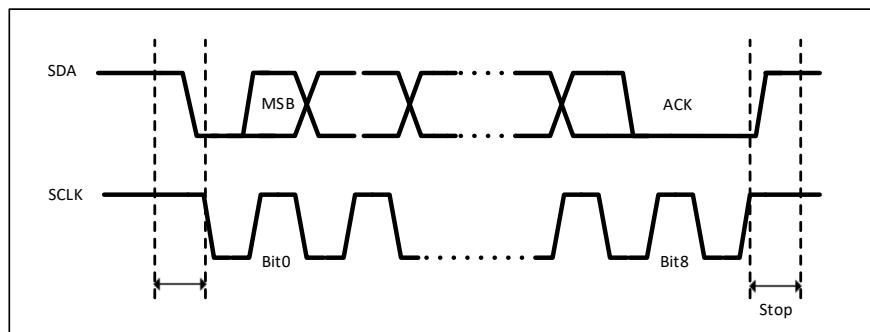


图 26-2 I2C 总线协议

I2C 支持以下四种模式：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

默认工作在从模式。接口在生成起始条件后，自动地从从模式切换到主模式；在允许多主机功能的情况下，当仲裁丢失或产生停止信号时，则从主模式切换到从模式。

作为主模式时，I2C 接口启动数据传输，并产生时钟信号。串行数据的传输总是以起始条件开始，并以停止条件结束。起始条件和停止条件都是在主模式模式下由软件控制产生。

作为从模式，I2C 接口能识别自己的地址(7 位或 10 位)和广播呼叫地址。软件能够控制开启或禁止对广播呼叫地址的识别。

数据和地址按 8 位（字节）进行传输，高位在前。跟在起始条件后的 1 或 2 个字节是地址(7 位模式为 1 个字节，10 位模式为 2 个字节)。地址只在主模式模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收方必须回送一个应答位(ACK)给发送方。

### 26.2.2. 包错误校验 (PEC)

包错误校验(PEC)计算器是用于提高通信的可靠性，这个计算器使用一个可编程的多项式对每一位串行数据进行计算。

- PEC 计算由 I2C\_CR1 寄存器的 ENPEC 位激活。PEC 使用 CRC-8 算法对所有信息字节进行计算，包括地址和读/写位在内。
  - 在发送时：在最后一个 TxE 事件时设置 I2C\_CR1 寄存器中的 PEC 传输位，PEC 将在当前字节后被发送。
  - 在接收时：在最后一个 RxNE 事件时设置 I2C\_CR1 寄存器中的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器发送一个 NACK。如果是主接收器，不管校对的结果如何，PEC 后都将发送 NACK。
- 在 I2C\_SR1 寄存器中可获得 PECERR 错误标记/中断。
- 如果 DMA 和 PEC 计算器都被激活：

- 在发送时：当 I2C 接口从 DMA 控制器处接收到 EOT 信号时，它在最后一个字节后自动发送 PEC。
- 在接收时：当 I2C 接口从 DMA 处接收到一个 EOT\_1 信号时，它将自动把下一个字节作为 PEC，并且将检查它。在接收到 PEC 后产生一个 DMA 请求。
- 为了允许中间 PEC 传输，在 I2C\_CR2 寄存器中有一个控制位(LAST 位)用于判别是否真是最后一个 DMA 传输。如果确实是最后一个主接收器的 DMA 请求，在接收到最后一个字节后自动发送 NACK。
- 仲裁丢失时 PEC 计算失效。

### 26.2.3. 从模式

默认情况下，I2C 接口总是工作在 slave 模式。从 slave 模式切换到 master 模式，需要产生一个起始条件。为了产生正确的时序，必须在 I2C\_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址，被送到 shift 寄存器，并与芯片的地址 OAR1 或者 general call 地址(如果 ENGC=1)相比较。

#### 头段或地址不匹配：

I2C 接口将其忽略并等待另一个起始条件。

#### 地址匹配：

I2C 接口产生以下时序：

- 如果 ACK 被软件置‘1’，则产生一个应答脉冲
- 硬件置位 ADDR 位，如果设置了 ITEVTEN 位，则产生中断

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

#### 26.2.3.1. 从机发送模式

在接收到地址并清除 ADDR 位后，(如果地址字节的最低位是 1) Slave 将数据(字节)从 DR 寄存器，经由内部 shift 寄存器发送到 SDA 上。

Slave 拉低 SCL，直到 ADDR 位被清除，并且待发送数据已写入 DR 寄存器(参考 EV1、EV3)。

当收到应答脉冲时：TxE 位被硬件置位，如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 位被置位，但在下一个数据发送结束之前，没有新数据写入到 I2C\_DR 寄存器，则 BTF 位被置位。Slave 拉低 SCL，直到 BTF 位被软件清零(读 I2C\_SR1 之后，再写入 I2C\_DR 寄存器)。

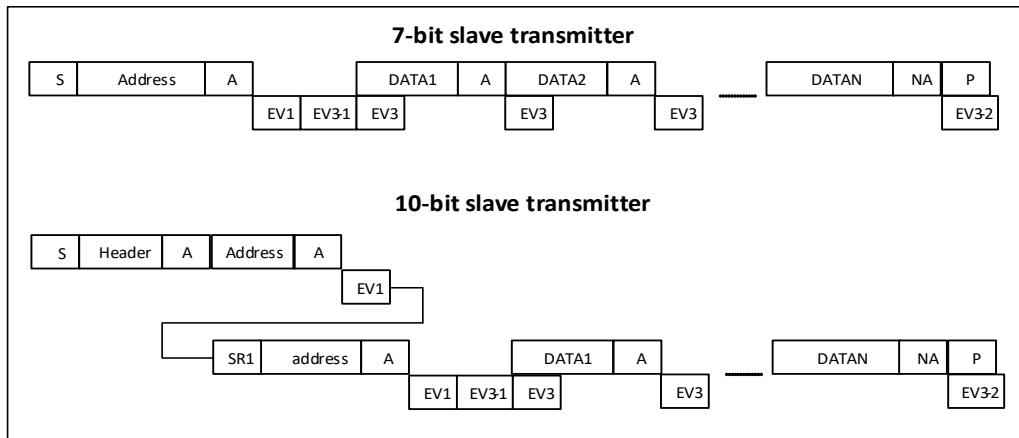


图 26-3 从发送器的传送序列图

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=1 时产生中断)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

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, shift 寄存器 empty, 数据寄存器 empty, 向 DR 寄存器写 Data1

EV3: TxE=1, shift 寄存器不 empty, 数据寄存器 empty, 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

### 26.2.3.2. 从机接收模式

在接收到地址并清除 ADDR 后, (如果地址字节的最低位是 0) slave 将通过内部移位寄存器把从 SDA 线接收到的字节存进 DR 寄存器。I2C 接口在接收到每个字节后都执行下列操作:

- 如果设置了 ACK 位, 则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVTEN 和 ITBUFEN 位, 则产生一个中断。

如果 RxNE 被置位, 并且在接收新的数据结束之前, DR 寄存器未被读出, 则 BTF 位被置位, 在清除 BTF (读出 I2C\_SR1 之后再读 I2C\_DR 寄存器) 之前, slave 一直拉低 SCL。(见下图)。

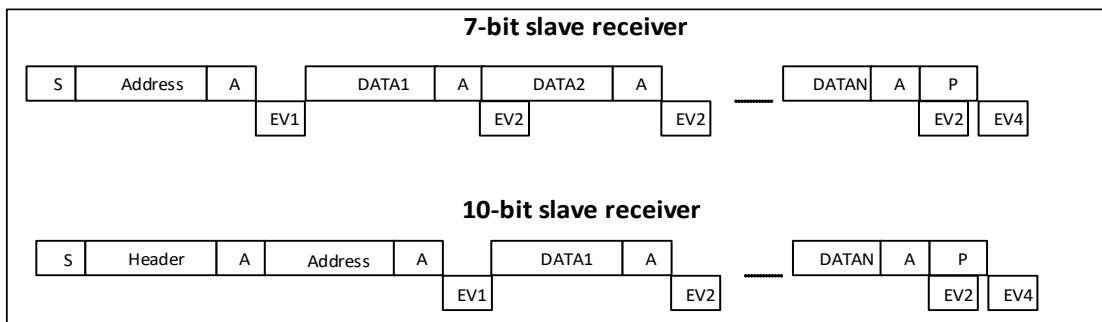


图 26-4 从接收器的传送序列图

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=1 时产生中断)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledged, EVx= Event(with interrupt if ITEVFEN=1)

**EV1**: ADDR=1, 通过先读 SR1, 后读 SR2, 实现 ADDR 的清零

**EV2**: RxNE=1, 读 DR 寄存器清零该位

**EV4**: STOPF=1, 通过先读 SR1 寄存器, 后写 CR1 寄存器实现对该位的清零。

Note:

1) EV1 event 拉低 SCL, 直到相应软件 sequence 的结束。

2) EV2 软件 sequence 必须在当前 byte 传输完成前即完成。

3) 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 sequence。比如 ADDR 和 STOPF 标志位, 需要用以下 sequence:

如果 ADDR=1, 先读 SR, 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。

### 26.2.3.3. 关闭通信

在传输完最后一个数据字节后, master 产生一个停止条件, slave 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。

### 26.2.4. 主模式

在 Master 模式时, I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始, 并以停止条件结束。当通过 START 位在总线上产生了起始条件, 设备就进入了 master 模式。

以下是 master 模式所要求的操作顺序:

- 在 I2C\_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程 I2C\_CR1 寄存器启动外设
- 置 I2C\_CR1 寄存器中的 START 位为 1, 产生起始条件

I2C 模块的输入时钟频率必须至少是:

- 标准模式下为: 2MHz
- 快速模式下为: 4MHz

#### 26.2.4.1. 主机产生 clock

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 slave 可能拉长 SCL 信号, 在 SCL 上升沿产生后, master 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 slave 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 slave 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。

#### 26.2.4.2. 开始条件

当 BUSY=0 时，设置 START=1，I2C 接口将产生一个 Start 条件，并切换至 master 模式(MSL 被置位)。

注：

- 1) 在 master 模式下设置 START 位，将在当前字节传输完后，由硬件产生一个 ReStart 条件。
- 2) 一旦发出 Start 条件：SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。master 读 SR1 寄存器，再把 slave 地址写入 DR 寄存器。

#### 26.2.4.3. 从机地址发送

slave 地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  - ADD10 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  - 然后主设备等待一次读 SR1 寄存器，跟着将第二个地址字节写入 DR 寄存器。
  - ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  - 随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器。
- 在 7 位地址模式时，将送出一个地址字节。
  - 一旦该地址字节被送出，ADDR 位被硬件置位，如果设置了 ITEVFEN 位，则产生一个中断。
  - 随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器。
  - 根据送出从地址的 LSB 位，主设备决定进入是发送器模式还是接收器模式。
- 在 7 位地址模式时，
  - 要进入发送器模式，主设备发送从地址时让 LSB 等于 0。
  - 要进入接收器模式，主设备发送从地址时让 LSB 等于 1。
- 在 10 位地址模式时
  - 要进入发送器模式，主设备先送头字节(11110xx0)，然后送 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

#### 26.2.4.4. 主发送器

在发送了地址和清除了 ADDR 位后，主设备 master 通过内部移位寄存器将字节从 DR 寄存器发送到 SDA 线上。Master 等待，直到第一个数据字节被写入 DR 寄存器。

当收到 ACK 脉冲时，TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

如果 TxE 被置位，且在上一次数据发送结束之前，没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位。在清除 BTF (读 I2C\_SR1 之后，再写 I2C\_DR 寄存器) 之前，I2C 接口将保持 SCL 为低电平。

#### 26.2.4.5. 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，然后 I2C 接口将自动回到从模式 (MSL 位清除)。

注：

- 当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8\_2 事件时。

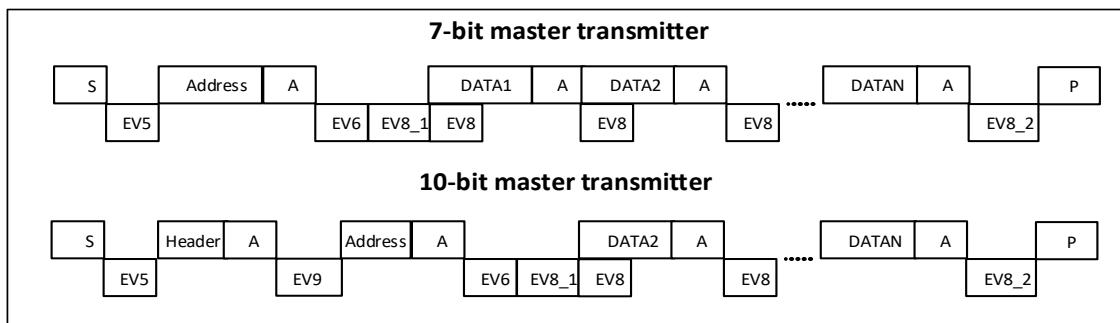


图 26-5 主发送器的传送序列图

说明：S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=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

Note:

- EV5, EV6, EV8\_1 和 EV8\_2 事件，拉长 SCL 的低电平，直到相应的软件 sequence 执行结束
- EV8 软件 sequence 必须在当前字节发送完成前执行完毕。在 EV8 的软件 sequence 不能在当前传输的字节结束前被完成，推荐使用 BTF 代替 TxE，这产生的不利是减慢了通讯。

#### 26.2.4.6. 主接收器

在发送地址和清除 ADDR 之后，I2C 接口进入主接收器模式。在此模式下，I2C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DR 寄存器。在每个字节后，I2C 接口依次执行以下操作：

- 如果 ACK 位被置位，发出一个应答脉冲。
- 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断。

如果 RxNE 位被置位，并且在接收新数据结束前，DR 寄存器中的数据没有被读走，硬件将设置 BTF=1，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2C\_SR1 之后再读出 I2C\_DR 寄存器将清除 BTF 位。

#### 26.2.4.7. 关闭通信

**Method 1:** 该方法的应用场景是：当 I2C 被设成应用程序中最高优先级的中断

Master 在从 Slave 接收到最后一个字节后，发送一个 NACK。接收到 NACK 后，Slave 释放对 SCL 和 SDA 线的控制。Master 就可以发送一个 Stop/Restart 条件。

- 1) 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)必须清除 ACK 位。
- 2) 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后(在倒数第二个 RxNE 事件之后)设置 STOP/START 位。
- 3) 只接收一个字节时，刚好在 EV6 之后(EV6\_1 时，清除 ADDR 之后)要关闭应答和停止条件的产生位。在产生了停止条件后，I2C 接口自动回到从模式(MSL 位被清除)。

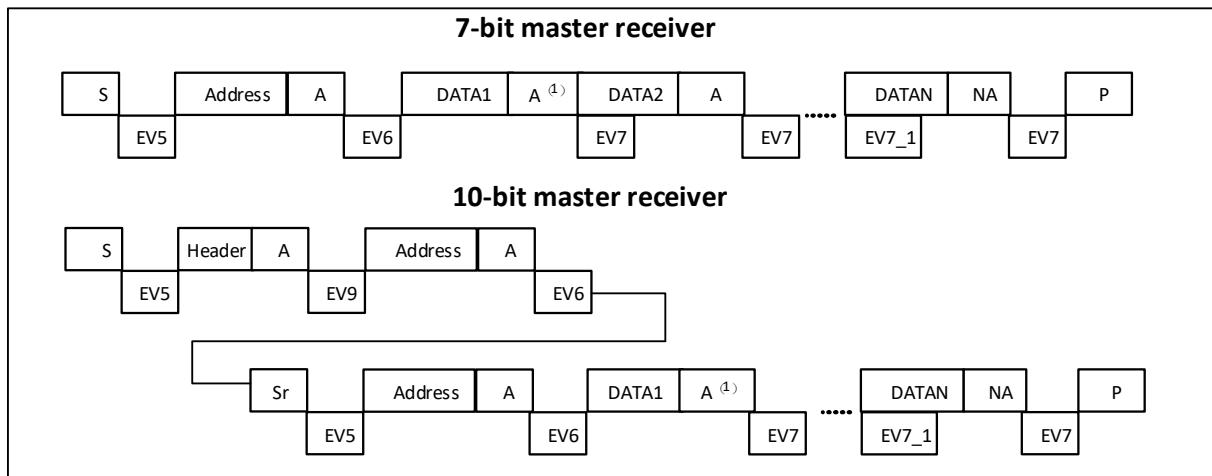


图 26-6 方法 1: 主模式接收时的时序

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=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

NOTE:

- 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 sequence 执行结束
- 3) EV7 软件 sequence 必须在当前字节发送完成前执行完毕。在 EV7, 软件 sequence 不能在当前传输的字节传输完成前, 被管理。推荐使用 BTF 代替 RXNE, 这产生的不利是减慢了通讯。
- 4) EV6\_1 或者 EV7\_1 的软件 sequence 必须在当前字节传输的 ACK 之前完成。

**Method 2:** 这个方法的应用场景是: I2C 的中断在应用中不是最高优先级, 或者使用查询方式

用这个方法, 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。

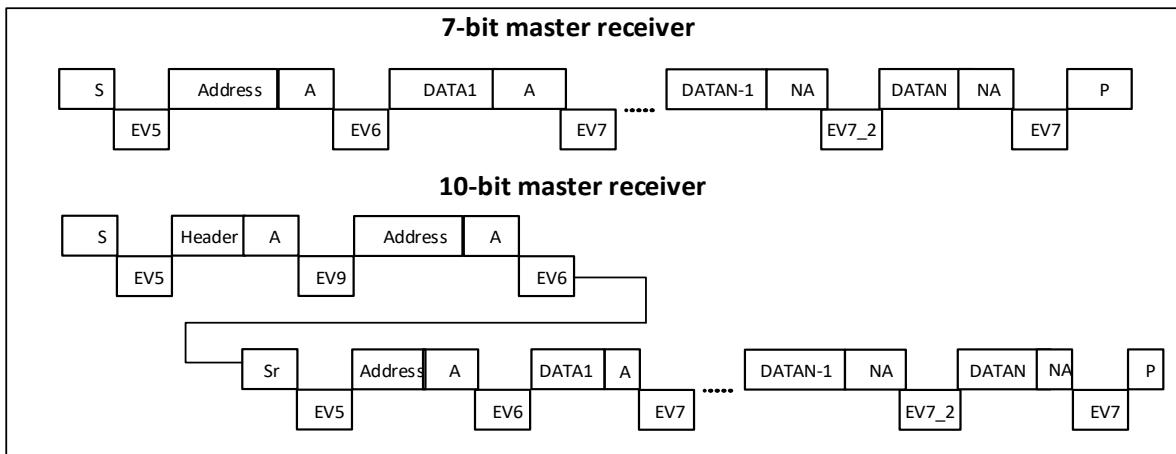


图 26-7 方法 2:N&gt;2 时主模式接收时的时序

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=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 shift 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

Note:

- 1) EV5, EV6 事件, 拉长 SCL 的低电平, 直到相应的软件 sequence 执行结束
- 2) EV7 软件 sequence 必须在当前字节发送完成前执行完毕。在 EV7, 软件 sequence 不能在当前传输的字节传输完成前, 被管理。推荐使用 BTF 代替 RXNE, 这产生的不利是减慢了通讯。

●当 3 个字节要被读走:

— RxNE=1 => Nothing(DataN-2 not read)。

— DataN-1 received

— BTF=1, shift 和 data 寄存器都是 full: 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 shift 寄存器对 DataN 的接收

— DataN 接收完成 (with a NACK)

— 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

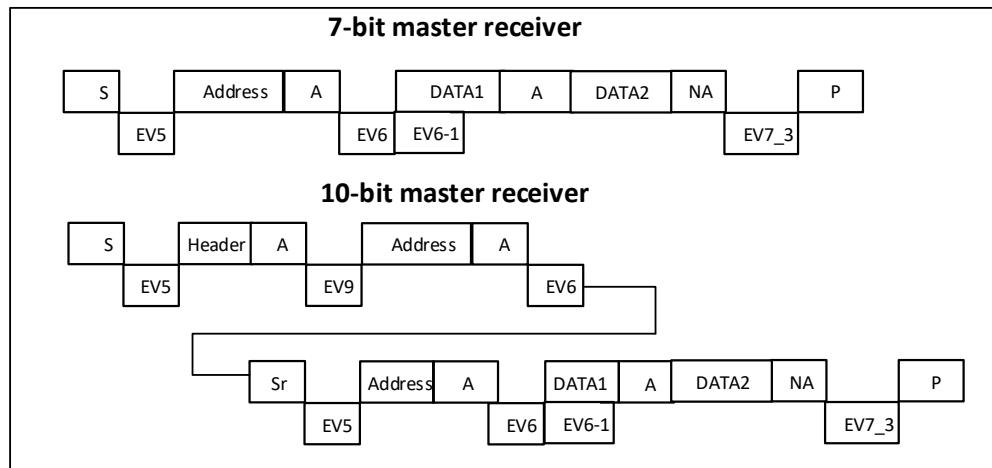


图 26-8 N=2 时主模式接收时的时序

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=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

Note:

1) EV5, EV6 事件, 拉长 SCL 的低电平, 直到相应的软件 sequence 执行结束

2) EV6\_1 的软件 sequence 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

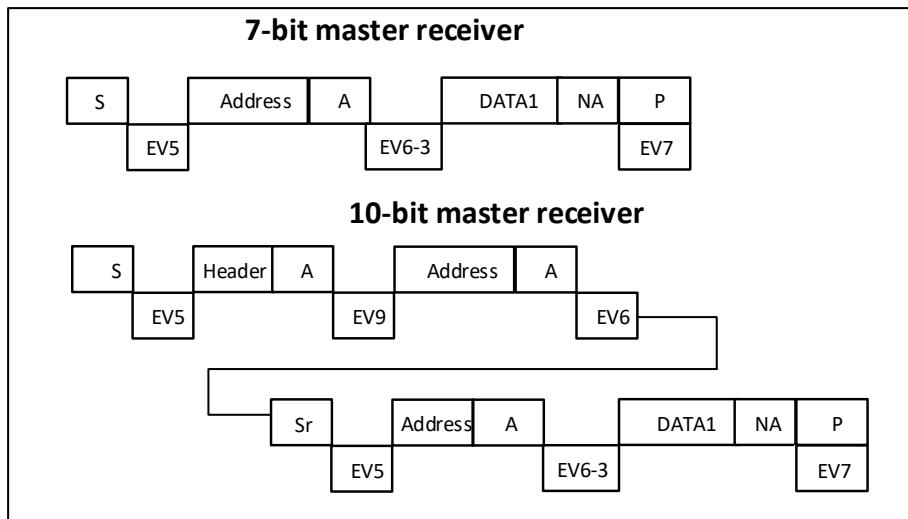


图 26-9 N=1 时主模式接收时的时序

说明: S=start, Sr=restart, P=stop, A=ack, NA=nack, EVx=event (当 ITEVTEN=1 时产生中断)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

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

Note:

EV5, EV6, EV8\_1 和 EV8\_2 事件, 拉长 SCL 的低电平, 直到相应的软件 sequence 执行结束。

## 26.2.5. 错误状态

### 26.2.5.1. 总线错误

在一个地址或数据字节传输期间, 当 I2C 接口检测到一个外部的停止或起始条件则产生总线错误。此时:

- BERR 位被置位为' 1' ; 如果设置了 ITERREN 位, 则产生一个中断;
- 在 slave 模式: 数据被丢弃, 硬件释放总线:
  - 如果是错误的 Start 条件, slave 认为是一个 Restart, 并等待地址或停止条件
  - 如果是错误的 Stop 条件, slave 按正常的停止条件操作, 同时硬件释放总线
- 在 master 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。

### 26.2.5.2. 应答错误(AF)

当接口检测到一个无应答位时, 产生应答错误。此时:

- AF 位被置位, 如果设置了 ITERREN 位, 则产生一个中断
- 当发送器接收到一个 NACK 时, 必须复位通讯:
  - 如果是处于 slave 模式, 硬件释放总线。

— 如果是处于 master 模式，软件必须生成一个停止条件或者 repeated start。

### 26.2.5.3. 仲裁丢失(ARLO)

当 I2C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

- ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断
- I2C 接口自动回到从模式(MSL 位被清除)。当 I2C 接口丢失了仲裁，则它无法在同一个传输中响应它的从地址，但它可以在赢得总线的 master 发送 repeated start 条件之后响应
- 硬件释放总线

### 26.2.5.4. 过载/欠载错误(OVR)

在 slave 模式下，如果禁止时钟延长 (nostretch = 1)，I2C 接口正在接收数据时，当它已经接收到一个字节 (RxNE=1)，但在 DR 寄存器中前一个字节数据还没有被读出，则发生 overrun 错误。

此时：

- 最后接收的数据被丢弃
- 在 overrun 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 slave 模式下，如果禁止时钟延长，I2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DR 寄存器(TxE=1)，则发生欠载错误。此时：

- 在 DR 寄存器中的前一个字节将被重复发出
- 用户应该确定在发生 underrun 错时，接收端应丢弃重复接收到的数据。发送端应按 I2C 总线标准在规定的时间更新 DR 寄存器

在发送第一个字节时，必须在清除 ADDR 之后且在第一个 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。

## 26.2.6. DMA 功能

DMA 请求(当被使能时)仅用于数据传输。发送时数据寄存器变空，或接收时数据寄存器变满，则产生 DMA 请求。DMA 必须在当前字节传输结束之前被初始化和使能。DMAEN 位 (I2C\_CR2 寄存器中) 必须在 ADDR 事件发生前使能。

在 master 或者 slave 模式，当时钟延展功能使能，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 controller 向 I2C 发送 EOT (End of transfer)，并产生 transfer complete 中断 (如果中断使能位有效)：

- Master transmitter：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后，置位 stop 条件。
- Master receiver：当要接收的数据数目大于或等于 2 时，DMA controller 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。

### 26.2.6.1. 利用 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 I2C 分配一个 DMA 通道，须执行以下步骤(x 是通道号)：

1. 在 DMA\_CPARx 寄存器中设置 I2C\_DR 寄存器地址。数据将在每个 TxE 事件后，从存储器传送至这个地址。
2. 在 DMA\_CMARx 寄存器中设置存储器地址。数据在每个 TxE 事件后从这个存储区传送至 I2C\_DR。

3. 在 DMA\_CNDTRx 寄存器中设置所需的传输字节数。在每个 TxE 事件后，此值将被递减。
4. 利用 DMA\_CCRx 寄存器中的 PL[0:1]位配置通道优先级。
5. 设置 DMA\_CCRx 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全部完成时发出中断请求。
6. 通过设置 DMA\_CCTx 寄存器上的 EN 位激活通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/EOT\_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注：如果使用 DMA 进行发送时，不要设置 I2C\_CR2 寄存器的 ITBUFEN 位。

#### 26.2.6.2. 利用 DMA 接收

通过设置 I2C\_CR2 寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2C\_DR 寄存器的数据传送到设置的存储区(参考 DMA 说明)。设置 DMA 通道进行 I2C 接收，须执行以下步骤(x 是通道号)：

1. 在 DMA\_CPARx 寄存器中设置 I2C\_DR 寄存器的地址。数据将在每次 RxNE 事件后从此地址传送到存储区。
2. 在 DMA\_CMARx 寄存器中设置存储区地址。数据将在每次 RxNE 事件后从 I2C\_DR 寄存器传送到此存储区。
3. 在 DMA\_CNDTRx 寄存器中设置所需的传输字节数。在每个 RxNE 事件后，此值将被递减。
4. 用 DMA\_CCRx 寄存器中的 PL[0:1]配置通道优先级。
5. 清除 DMA\_CCRx 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部完成时发出中断请求。
6. 设置 DMA\_CCRx 寄存器中的 EN 位激活该通道。

当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的 EOT/EOT\_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注：

- 1) 如果使用 DMA 进行接收时，不要设置 I2C\_CR2 寄存器的 ITBUFEN 位。

表 26-10 I2C 中断请求

中断事件	事件标志	开启控制位
起始位已发送(Master)	SB	ITEVTEN
地址已发送(Master) 或 地址匹配(Slave)	ADDR	
ADD10	10 位头段已发送(主)	
已收到停止(Slave)	STOPF	
数据字节传输完成	BTB	
接收缓冲区非空	RxNE	ITEVTEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	
仲裁丢失(Master)	ARLO	
响应失败	AF	
过载/欠载	OVR	ITERREN
PEC 错误	PECERR	
超时/Tlow 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

## 26.2.7. 系统管理总线 SMBus

### 26.2.7.1. 介绍

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 I2C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

系统管理总线(SMBus)标准涉及三类设备。从设备，接收或响应命令的设备。主设备，用来发布命令，产生时钟和终止发送的设备。主机，是一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能，并且必须支持 SMBus 通报协议。在一个系统里只允许有一个主机。

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

本项目的 I2C1 模块支持 SMBus/PMbus 功能。SMBus 与 I2C 比较

### 26.2.7.2. SMBus 和 I2C 之间的不同点

SMBus	I2C
最大传输速度 100kHz	最大传输速度 400kHz
最小传输速度 10kHz	无最小传输速度
35ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 VDD 决定
不同的地址类型(保留、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

### 26.2.7.3. SMBus 和 I2C 之间相似点

- 2 条线的总线协议(1 个时钟，1 个数据) + 可选的 SMBus 提醒线
- 主-从通信，主设备提供时钟
- 多主机功能
- SMBus 数据格式类似于 I2C 的 7 位地址格式

### 26.2.7.4. 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 有以下的属性：

- 地址分配利用标准 SMBus 物理层仲裁机制
- 当设备维持供电期间，分配的地址仍保持不变，允许设备在断电时保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销(也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的)。
- 任何一个 SMBus 主设备可以遍历总线。

### 26.2.7.5. 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。与 SMBus 有关的消息为 2 字节。

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。

### 26.2.7.6. 超时错误 (TIMEOUT)

在定时规范上 I2C 和 SMBus 之间有很多差别。SMBus 定义一个时钟低超时，35ms 的超时。SMBus 规定 TLOW:SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 TLOW:MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范(<http://smbus.org/specs/>)。I2C\_SR1 中的状态标志 Timeout 或 Tlow 错误表明了这个特征的状态。

### 26.2.7.7. 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)。

## 26.3. 寄存器描述

寄存器可以 half-word 或者 word 访问。

### 26.3.1. I2C Control register 1 (I2C\_CR1)

Address offset:0x00

Reset value:0x0000

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	PE
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	R W

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	软件复位。 当被置位时，I2C 处于复位状态。在复位释放前，要确保 I2C 的引脚被释放，总线是空闲状态。 0：I2C 模块不处于复位状态 1：I2C 模块处于复位状态

Bit	Name	R/W	Reset Value	Function
				注：该位可以用于 error 或 locked 状态时重新初始化 I2C。如 BUSY 位为 1，在总线上又没有检测到停止条件时。
14	Reserved	RES	-	Reserved
13	ALERT	RW	0	SMBus 提醒。 0：释放 SMBAlert 引脚使其变高。提醒响应地址头紧跟在 NACK 信号后面； 1：驱动 SMBAlert 引脚使其变低。提醒响应地址头紧跟在 ACK 信号后面； PE=0 时，由硬件清除。
12	PEC	RW	0	数据包出错检查 软件可置位和清零该位，硬件可以在以下情况下清零该位：当传送 PEC 后，起始条件、或者停止条件、或者当 PE=0 时； 0：无 PEC 传输 1：PEC 传输（在发送或接收模式） 注：仲裁丢失时，PEC 的计算失效。
11	POS	RW	0	ACK/PEC 位置（用于数据接收），软件可置位/清零该寄存器，或 PE=0 时由硬件清零。 0：ACK 位控制当前移位寄存器内正在接收的字节的 (N)ACK。PEC 位表明当前移位寄存器内的字节是 PEC 1：ACK 位控制在移位寄存器里接收的下一个字节的 (N)ACK。PEC 位表明在移位寄存器里接收的下一个字节是 PEC 注：POS 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。 为了 NACK 第 2 个字节，必须在清除 ADDR 之后清除 ACK 位。 为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，ADDR stretch 事件时设置 PEC 位。
10	ACK	RW	0	应答使能。软件可置位/清零该寄存器，或 PE=0 时由硬件清零。 0：无应答返回 1：在接收到一个字节后返回一个应答。（匹配的地 址或数据）
9	STOP	RW	0	停止条件产生，软件可以置位/清零该寄存器，或者当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件置位。 在主模式下： 0：无停止条件产生

Bit	Name	R/W	Reset Value	Function
				1: 在当前字节传输或在当前起始条件发出后产生停止条件 在从模式下: 0: 无停止条件产生 1: 在当前字节传输后释放 SCL 和 SDA 线
8	START	RW	0	起始条件产生。 软件可置位/清零该寄存器，或当起始条件发出后或 PE=0 时由硬件清零。 主模式： 0: 无起始条件产生 1: 重复产生起始条件 从模式： 0: 无起始条件产生 1: 当总线空闲时，产生起始条件（并由硬件自动切换到 master mode）
7	NOSTRETCH	RW	0	禁止时钟延长（Slave）。 当 ADDR 或 BTF 标志被置位时，该位用于 slave 禁止时钟延长，直到被软件复位。 0: 允许时钟延长 1: 禁止时钟延长
6	ENGC	RW	0	广播呼叫使能。 0: 禁止广播呼叫。以 NACK 响应地址 00h 1: 允许广播呼叫。以 ACK 响应地址 00h
5	ENPEC	RW	0	PEC 使能。 0: 禁止 PEC 计算 1: 开启 PEC 计算
4	ENARP	RW	0	ARP 使能。 0: 禁止 ARP; 1: 使能 ARP; 如果 SMBTYPE=0，使用 SMBus 设备的默认地址; 如果 SMBTYPE=1，使用 SMBus 的主地址。
3	SMBTYPE	RW	0	SMBus 类型。 0: SMBus 设备; 1: SMBus 主机;
2	Reserved	RES	-	Reserved
1	SMBUS	RW	0	SMBus 模式。 0: I2C 模式; 1: SMBus 模式;
0	PE	RW	0	I2C 模块使能。 0: 禁止

Bit	Name	R/W	Reset Value	Function
				<p>1: I2C 使能根据 SMBus 位的配置，相应的 I/O 口需配置为复用功能。</p> <p>注：如果清除该位时通讯正在进行，在当前通讯结束后，I2C 模块被禁用并返回空闲状态。</p> <p>由于在通讯结束后 PE=0，所有的位被清除。</p> <p>在主模式下，通讯结束之前，绝不能清除该位。</p>

### 26.3.2. I2C Control register 2 (I2C\_CR2)

Address offset:0x04

Reset value:0x0000

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	Reserved	RES	-	Reserved
12	LAST	RW	0	<p>DMA 最后一次传输。</p> <p>0: 下一次 DMA 的 EOT 不是最后的传输</p> <p>1: 下一次 DMA 的 EOT 是最后的传输</p> <p>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。</p>
11	DMAEN	RW	0	<p>DMA 请求使能。</p> <p>0: 禁止 DMA 请求；</p> <p>1: 当 TxE=1 或 RxNE=1 时，允许 DMA 请求。</p>
10	ITBUFEN	RW	0	<p>缓冲器中断使能。</p> <p>0: 当 TxE=1 或 RxNE=1 时，不产生中断</p> <p>1: 当 TxE=1 或 RxNE=1 时，产生事件中断（不管 DMAEN 是何值）</p>
9	ITEVTEN	RW	0	<p>事件中断使能。</p> <p>0: 禁止</p> <p>1: 允许事件中断</p> <p>在下列条件下，将产生该中断：</p> <ul style="list-style-type: none"> <li>● SB=1 (主模式)；</li> <li>● ADDR=1 (主/从模式)</li> <li>● ADD10=1 (主模式)；</li> <li>● STOPF=1 (从模式)</li> <li>● BTF=1，但没有 TxE 或 RxNE 事件</li> <li>● 如果 ITBUFFEN=1，TxE 事件为 1</li> <li>● 如果 ITBUFEN=1，RxNE 事件为 1</li> </ul>
8	ITERREN	RW	0	出错中断使能。

Bit	Name	R/W	Reset Value	Function
				<p>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：</p> <ul style="list-style-type: none"> <li>● BERR=1</li> <li>● ARLO=1</li> <li>● AF=1</li> <li>● OVR=1</li> <li>● PECERR=1</li> <li>● TIMEOUT=1;</li> <li>● SMBAlert=1.</li> </ul>
7:6	Reserved	RES	-	Reserved
5:0	FREQ	RW	0	<p>I2C 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。运行时钟为偶数时钟。      最小允许可设定的频率分别是 100k 模式下 4MHz 及以上、400k 模式下大于 8MHz。</p> <p>000000: 禁止      000001: 禁止      .....      000011: 禁止      000100: 4MHz      .....      110010: 50MHz      大于 110010: 禁止。</p>

### 26.3.3. I2C Own address register1 (I2C\_OAR1)

Address offset:0x08

Reset value:0x4000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD-MODE	Res.	Res.	Res.	Res.	Res.	ADD[9:8]		ADD[7:1]							ADD0
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	ADDMODE	RW	0	<p>寻址模式（从模式）。</p> <p>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）；</p>
14:10	Reserved	RES	0	Reserved
9:8	ADD[9:8]	RW	0	<p>接口地址。</p> <p>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。</p>
7:1	ADD[7:1]	RW	0	接口地址的 7~1 位。

Bit	Name	R/W	Reset Value	Function
0	ADD0	RW	0	接口地址。 7 位地址模式该寄存器无效。 10 位地址模式位地址的 0 位。

### 26.3.4. I2C own address register2 (I2C\_OAR2)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADD2[7:1]														
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
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。

### 26.3.5. I2C Data register(I2C\_DR)

Address offset:0x10

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
7:0	DR[7:0]	RW	0	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)。 <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, I2C 模块将保持连续的数据流。 <b>接收器模式:</b>

Bit	Name	R/W	Reset Value	Function
				<p>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 slave 模式下，地址不会被 copy 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 copy 到数据寄存器里，因此不能读到</li> </ol>

### 26.3.6. I2C Status register (I2C\_SR1)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBA LERT	TIMEO UT	Re s.	PECE RR	OVR	AF	ARLO	BERR	Tx E	RxN E	Re s.	STOP F	ADD 10	BT F	ADD R	S B
RC_W 0	RC_W0		RC_W 0	RC_ W0	RC_ W0	RC_ W0	RC_ W0	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。  SMBus 主机模式：  0: 无 SMBus 提醒；  1: 在引脚产生 SMBAlert 提醒事件；  SMBus 从机模式：  0: 没有 SMBAlert 响应地址头序列；  1: 收到 SMBAlert 响应地址头序列直到 SMBAlert 变低。  该位由软件写 0 或 PE=0 时由硬件清除。</p>
14	TIMEOUT	RC_W0	0	<p>超时或 Tlow 错误。  0: 无超时错误；  1: SCL 为低的持续时间已达到 25ms (超时)；或者主机低电平累计时钟扩展时间超过 10ms (Tlow:next)；或从设备低电平累积时钟扩展时间超过 25ms (Tlow:sext)。  当在从模式下设置该位：从设备复位通讯，硬件释放总线；  当在主模式下设置该位：硬件发出停止条件；  该位由软件写 0 清除，或当 PE=0 时由硬件清除。  Note: 该功能仅在 SMBUS 模式有作用。</p>
13	Reserved	RES	-	Reserved

Bit	Name	R/W	Reset Value	Function
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> <p>该位由软件写 0 清除, 或当 PE=0 时由硬件清除。</p> <p>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。</p>
10	AF	RC_W0	0	<p>应答失败标志。</p> <p>0: 没有应答失败;</p> <p>1: 应答失败。</p> <p>当没有返回应答时, 硬件将置位该寄存器。</p> <p>该位由软件写 0 清除, 或当 PE=0 时由硬件清除。</p>
9	ARLO	RC_W0	0	<p>仲裁丢失 (主模式)。</p> <p>0: 没有检测到仲裁丢失;</p> <p>1: 检测到仲裁丢失。</p> <p>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。</p> <p>该位由软件写 0 清除, 或在 PE=0 时由硬件清除。</p> <p>在 ARLO 事件之后, I2C 接口自动切换回从模式 (M/SL=0)。</p> <p>注: 在 SMBUS 模式下, 在从模式下对数据的仲裁仅发生在数据解读, 或应答传输区间 (不包括地址的应答)。</p>
8	BERR	RC_W0	0	<p>总线出错标志。</p> <p>0: 无起始或者停止条件出错;</p> <p>1: 起始或者停止条件出错。</p> <p>当接口检测到错误的起始或者停止条件, 硬件将该位置 1。</p>

Bit	Name	R/W	Reset Value	Function
				该位由软件写 0 清除，或者在 PE=0 时由硬件清除。
7	TxE	R	0	<p>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。</p>
6	RxNE	R	0	<p>数据寄存器非空（接收时）标志。            0：数据寄存器为空；            1：数据寄存器非空。            在接收时，当数据寄存器不为空，置位该寄存器。在接收地址阶段，该寄存器不置位。            软件对数据寄存器的读写操作会清除该寄存器，或当 PE=0 时由硬件清除。            注：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为此时数据寄存器仍为满。</p>
5	Reserved	RES	-	Reserved
4	STOPF	R	0	<p>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。            注：在收到 NACK 后，STOPF 位不会置位。</p>
3	ADD10	R	0	<p>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，当主设备将已将第一个字节发送出去时，硬件将该位置 1。            软件读取 I2C_SR1 寄存器后，对 I2C_DR 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。            主：收到 NACK 后，该寄存器不被置位。</p>
2	BTF	R	0	字节传输发送结束标志位。

Bit	Name	R/W	Reset Value	Function
				<p>0: 字节传输发送未完成      1: 字节传输发送成功结束      当 NOSTRETCH=0 时，在下列情况下硬件将置位该寄存器（当 slave 模式，NOSTRETCH=0 时；master 模式，与 NOSTRETCH 无关）：      — 接收时，当收到一个新字节（包括 ACK 脉冲）且数据寄存器还未被读取（RxNE=1）。      — 发送时，当一个新数据应该被发送，且数据寄存器还未被写入新的数据（TxE=1）。      软件读取 I2C_SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。      注：      在收到一个 NACK 后，BTF 位不会被置位。      如果下一个要传输的字节是 PEC      (I2C_SR2.TRA=1,I2C_CR1.PEC=1), BTF 位不会被置位。</p>
1	ADDR	R	0	<p>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 寄存器对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除。</p> <p><b>地址匹配 (Slave) :</b>      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 general call 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。</p> <p><b>Note:</b> 在 slave 模式下，推荐进行完整的清零 sequence，即在 ADDR 被置位后，先读 SR1 寄存器，再读 SR2 寄存器。</p> <p><b>地址已发送 (Master) :</b>      0: 地址发送没有结束；      1: 地址发送结束。      ● 10 位地址时，当收到地址的第二个字节的 ACK 后置位；      — 7 位地址时，当收到 ACK byte 后置位。  <b>Note:</b> 在收到 NACK 后，该寄存器不会被置位。</p>
0	SB	R	0	<p>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。</p>

Bit	Name	R/W	Reset Value	Function
				—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。

### 26.3.7. I2C Status register2 (I2C\_SR2)

Address offset:0x18

Reset value:0x0000

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。 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	Reserved	RES	-	Reserved
2	TRA	R	0	发送/接收标志。 0：接收到数据 1：数据已发送

Bit	Name	R/W	Reset Value	Function
				<p>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。</p> <p>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。</p>
1	BUSY	R	0	<p>总线忙标志。</p> <p>0：在总线上无数据通讯</p> <p>1：在总线上正在极性数据通讯</p> <p>当检测到 SDA 或 SCL 为低电平时，硬件置位。</p> <p>当检测到一个停止条件时，硬件清零。</p> <p>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。</p>
0	MSL	R	0	<p>主从模式。</p> <p>0：slave</p> <p>1：master</p> <p>—当接口处于主模式 (SB=1) 时，硬件置位；</p> <p>—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。</p>

### 26.3.8. I2C Clock control register(I2C\_CCR)

Address offset:0x1c

Reset value:0x0000

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											

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	<p>I2C 主模式选择。</p> <p>0：标准模式</p> <p>1：快速模式</p>
14	DUTY	RW	0	<p>快速模式时的占空比。</p> <p>0：快速模式下：<math>T_{low}/T_{high}=2</math></p> <p>1：快速模式下：<math>T_{low}/T_{high}=16/9</math></p>
13:12	Reserved	RES	-	Reserved
11:0	CCR[11:0]	RW	0	<p>快速/标准模式下的时钟控制分频系数（主模式）。</p> <p>该分频系数用于设置主模式下的 SCL 时钟。</p> <ul style="list-style-type: none"> <li>● 标准模式或者 SMBus 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> </li> <li>● 快速模式：           <ul style="list-style-type: none"> <li>✓ DUTY=0:</li> </ul> </li> </ul>

Bit	Name	R/W	Reset Value	Function
				<p>1. <math>T_{high}=CCR \times Tpclk</math>      2. <math>T_{low} = 2 \times CCR \times Tpclk</math>      ✓ DUTY=1(为达到 400KHz):      3. <math>T_{high}=9 \times CCR \times Tpclk</math>      4. <math>T_{low} = 16 \times CCR \times Tpclk</math></p> <p>注:</p> <ol 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> <li>—</li> </ol>

### 26.3.9. I2C TRISE register (I2C\_TRISE)

Address offset:0x20

Reset value:0x0002

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

Bit	Name	R/W	Reset Value	Function
15:6	Reserved	RES	-	Reserved
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 master mode 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 I2C 总线规范里给出的最大的 SCL 上升时间, 增长步幅为 1。      例如: 标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2C_CR2 寄存器中 FREQ[5:0]中的值等于 0x08, Tpclk=125ns, 则 TRISE 中配置为 0x09 (1000ns/125ns = 8 + 1 = 9)。      滤波器的值也可以加到 TRISE 内。      如果结果不为整数, 则将整数部分写入 TRISE, 以确保 tHIGH 参数。</p> <p>注: 当 PE=0 时才能设置该寄存器。</p>

### 26.3.10. I2C 寄存器映射



# 27. 通用同步异步收发器 (USART)

## 27.1. 简介

通用同步异步收发器 (USART) 提供了一种灵活的方法和使用工业标准NRZ异步串行数据格式的外部设备之间进行全双工数据交换。USART利用分数波特率发生器提供宽范围的波特率选择。

他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

### 27.1.1. 主要特征

#### 27.1.1.1. USART 特征

- 全双工异步通信
- 发送和接收共用的可编程波特率，最高达4.5Mbit/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
- 支持8位和16位过采样模式
  - 全双工异步、半双工异步、多处理器通信（属于全双工异步）：支持8位和16位采样的配置；
  - 全双工同步、LIN、智能卡、红外：默认为16位采样，无需配置USART\_CR3的OVER8位。同步不考虑几位采样的问题。

### 27.1.1.2. USART 特性描述

接口通过三个引脚与其他设备连接在一起(见图248)。任何USART双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

RX：接收数据串行输入。通过过采样技术来区别数据和噪音，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的I/O端口配置。当发送器被激活，并且不发送数据时，TX引脚处于高电平。在单线和智能卡模式里，此I/O口被同时用于数据的发送和接收。

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(8或9位)，最低有效位在前
- 0.5, 1.5, 2个的停止位，由此表明数据帧的结束
- 使用分数波特率发生器——12位整数和4位小数的表示方法。
- 一个状态寄存器(USART\_SR)
- 数据寄存器(USART\_DR)
- 一个波特率寄存器(USART\_BRR)，12位的整数和4位小数
- 一个智能卡模式下的保护时间寄存器(USART\_GTPR)

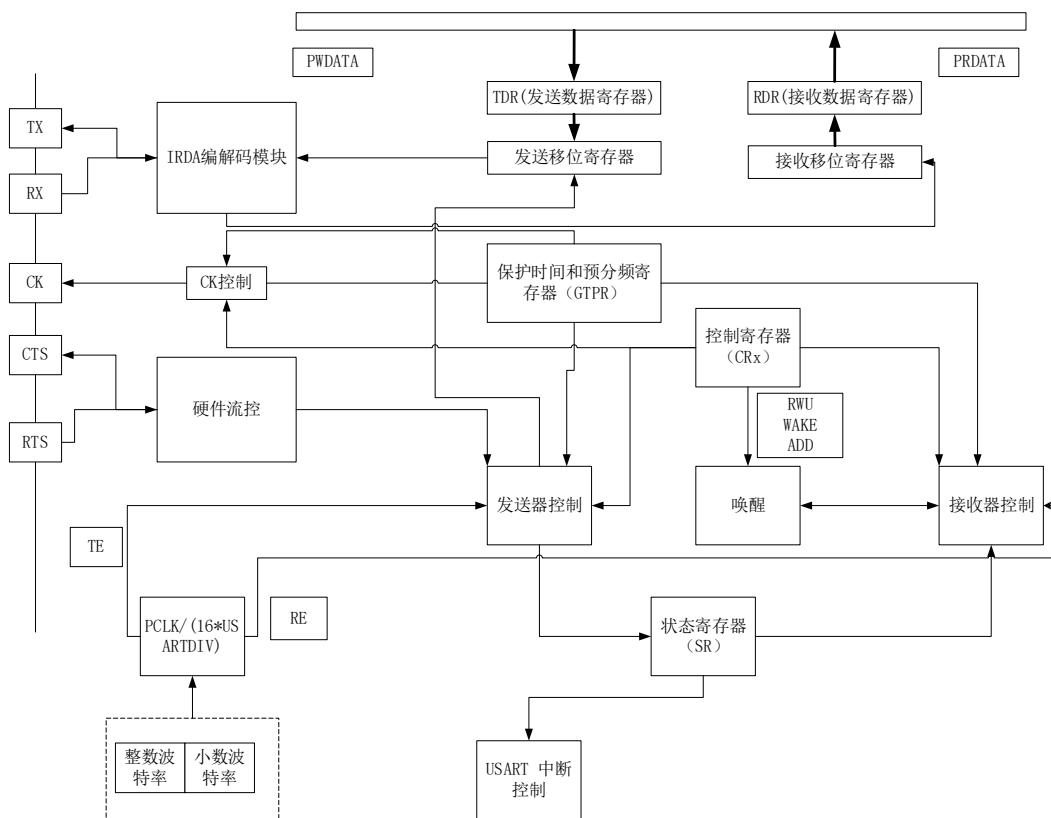


图 27-1 USART 框图

字长可以通过编程USART\_CR1寄存器中的M位，选择成8或9位（见图1.1）。在起始位期间，TX脚处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由‘1’组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位（‘1’的位数也包括了停止位的位数）。

断开符号 被视为在一个帧周期内全部收到‘0’（包括停止位期间，也是‘0’）。在断开帧结束时，发送器再插入1或2个停止位（‘1’）来应答起始位。

发送和接收由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。

## 27.2. USART 功能描述

### 27.2.1. 传输 pin 脚

#### 27.2.1.1. 基本传输 pin

任何USART双向通信至少需要两个脚：接收数据输入（RX）和发送数据输出（TX）。

RX：接收数据串行输入。通过采样技术来区别数据和噪声，从而恢复数据。

TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的I/O端口配置。当发送器使能，但不发送数据时，TX引脚处于高电平。在单线模式里，此I/O口被同时用于数据的发送和接收。

#### 27.2.1.2. 模式 pin

同步模式下增加CK pin。

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。时钟相位和极性都是软件可编程的。

### 27.2.1.3. 硬件流控制模式 pin

CTS：低电平开始发送，发送结束变为高电平。

RTS：低电平表明USART准备好接收数据。

## 27.2.2. USART 发送

发送器根据M位的状态发送8位或9位的数据字。当发送使能位(TE)被设置时，发送移位寄存器中的数据在TX脚上输出，相应的时钟脉冲在CK脚上输出。

字符发送

在USART发送期间，在TX引脚上首先移出数据的最低有效位。在此模式里，USART\_DR寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器(见图1-1)。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

USART支持多种停止位的配置：0.5、1、1.5和2个停止位。

注：

- 1) 在数据传输期间不能复位TE位，否则将破坏TX脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。
- 2) TE位被激活后将发送一个空闲帧。

### 27.2.2.1. 传输步骤

- 配置 USART\_CR1.M，决定是 8bit 还是 9bit 传输；
- 配置 USART\_BRR 寄存器，选择波特率；
- 配置 USART\_CR2.STOP 寄存器，定义 stop bit 个数
- 配置 USART\_CR1.UE=1,使能 USART
- 当多处理器传输时，配置 USART\_CR3.DMAT=1，使能 DMA；
- 配置 TE=1 (在写入数据到 USART\_TDR 寄存器前需要配置 TE=1，且整个传输阶段，TE 维持高) 。
- 硬件插入 Idle 时序
- 写传输数据到 USART\_TDR 寄存器 (该操作清零 TXE 位)。该项操作可能重复多次。
- 硬件置位 TXE，表明 USART\_TDR 寄存器数据已经移动至移位寄存器，USART\_TDR 寄存器空，可以写入新数据。
- 硬件产生发送时序：
  - Start bit，开始传送
  - 传送移位寄存器输出数据 (默认 LSB) 到 TX pin，时钟信号输出的 CK pin (同步模式) 。
  - Stop bit (根据配置发送个数) 。
  - 传送完最后 1 帧数据后，产生 TC=1 (TXE=1 时) .当 TCIE=1，产生 TC 中断。

### 27.2.2.2. TC/TXE 时序

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

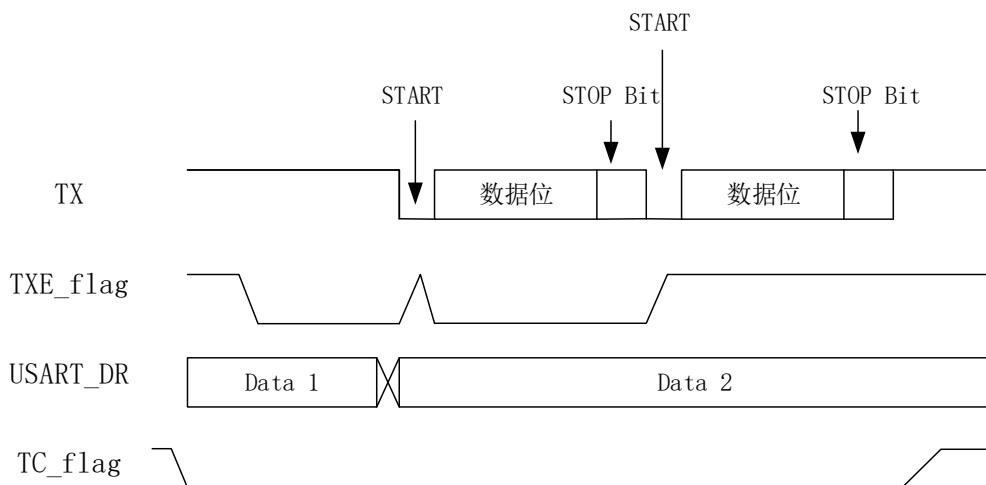


图 27-2 USART 发送时 TC/TXE 的变化

### 断开符号

设置SBK可发送一个断开符号。断开帧长度取决M位。如果设置SBK=1，在完成当前数据发送后，将在TX线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK被硬件复位。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。

### 27.2.3. 数据接收

#### 27.2.3.1. 起始位侦测

在USART中，如果辨认出一个特殊的采样序列，就认为侦测到一个起始位。该序列为：1110X0X0X0000。注：如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态(不设置标志位)等待下降沿。如果3个采样点都为0(在第3、5、7位的第一次采样，和在第8、9、10的第二次采样都为0)，则确认收到起始位，这时设置RXNE标志位，如果RXNEIE=1，则产生中断。如果两次3个采样点上仅有2个是0(第3、5、7位的采样点和第8、9、10位的采样点)，那么起始位仍然是有效的，但是会设置NE噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态(不设置标志位)。

如果有一次3个采样点上仅有2个是0(第3、5、7位的采样点或第8、9、10位的采样点)，那么起始位仍然是有效的，但是会设置NE噪声标志位。

#### 27.2.3.2. 字符接收

字符接收配置步骤：

- 配置 USART\_CR1.M，选择接收长度；

- 配置 USART\_BRR 寄存器，选择波特率；
- 配置 USART\_CR2.STOP，1bit 或者 2bit stop 位；
- 配置 USART\_CR1.UE=1，使能 USART；
- 多处理器传输时，配置 USART\_CR3.DMAR=1，使能 DMA；
- 配置 USART\_CR1.RE，使能接收，开始检测 start 位；

当接收到 1 字符后：

- 置位 RXNE。表明移位寄存器的内容已被转移到 RDR，即数据可以被读出；
- 当 RXNEIE=1 时，产生中断；
- 在接收时，如果检测到帧错误、噪声或溢出错误，则设置相应错误标志位；
- 多处理器通讯时，RXNE 在每个字节接收后被置位，并由 DMA 对接收数据寄存器的读操作而清零；
- 在单缓冲器模式，软件读 USART\_RDR 寄存器完成对 RXNE 位清零。RXNE 标志也可以通过写 USART\_RQR.RXFRQ=1 清零。RXNE 位必须在下一字符接收结束前被清零，以免溢出错误。

注：在接收数据的时候，RE 位不应该被复位。如果 RE 位在接收时被清零，当时字节的接收被丢失。

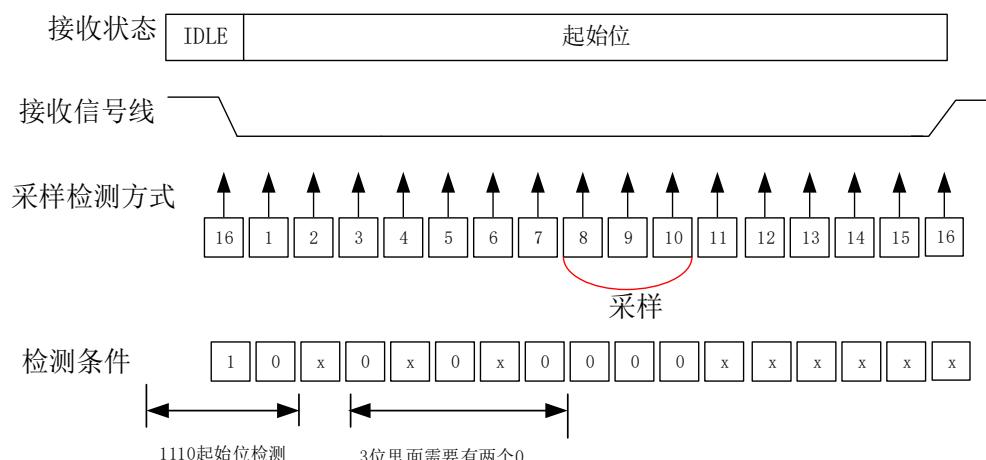


图 27-3 USART 的起始位检测

#### 特殊帧及错误处理：

- Break 帧：当接收到一个break帧时，USART按错误帧处理。
- Idle 帧：当检测到Idle帧时，按normal帧处理，如果IDLEIE=1，则产生中断。
- Overrun 错误：如果RXNE没有被复位，又接收到一个字符，则发生溢出错误。数据只有当RXNE位被清零后才能从移位寄存器转移到RDR寄存器。RXNE标志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请求还没执行时，RXNE标志仍为置起状态，溢出错误产生。

溢出错误产生时，执行如下错误处理：

- 置位 ORE；
- RDR 内容不会丢失。读 USART\_DR 寄存器仍可正常读出先前数据；
- 移位寄存器内容被覆盖，随后接收到的数据丢失；

- 如果 RXNEIE=1，或者 EIE=1，则产生中断；
- 写 ORECF=1 清零 ORE 位；

ORE 置位，表明至少 1 个数据已经丢失。有如下两种情况：

- 如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 中，可以被读出；
- 如果 RXNE=0，上一个有效数据已经被读走，RDR 无有效数据可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（被丢失）的数据时，此种情况可能发生。

### 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。

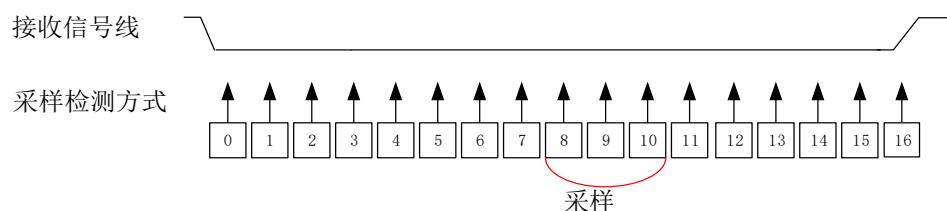


图 27-4 检测噪声的数据采样

表 27-1 检测噪声的数据采样

采样值	NE 状态	接收数据	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- 在RXNE位的上升沿设置NE标志。
- 无效数据从移位寄存器传送到USART\_DR寄存器。
- 在单个字节通信情况下，没有中断产生。然而，因为NE标志位和RXNE标志位是同时被设置，RXNE将产生中断。在多缓冲器通信情况下，如果已经设置了USART\_CR3寄存器中EIE位，将产生一个中断。

先读出USART\_SR，再读出USART\_DR寄存器，将清除NE标志位。

### 帧错误：

当以下情况发生时检测到帧错误：由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接和收识别出来。

当帧错误被检测到时：

- FE位被硬件置起
- 无效数据从移位寄存器传送到USART\_DR寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和RXNE位同时置起，后者将产生中断。在多缓冲器通信情况下，如果USART\_CR3寄存器中EIE位被置位的话，将产生中断。

顺序执行对USART\_SR和USART\_DR寄存器的读操作，可复位FE位。

#### 接收期间的可配置的停止位：

被接收的停止位的个数可以通过控制寄存器2的控制位来配置，在正常模式时，可以是1或2个，在智能卡模式里可能是0.5或1.5个。

- 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内时，拉低数据线，以此表示出现了帧错误。FE在1.5个停止位结束时和RXNE一起被置起。对1.5个停止位的采样是在第16，第17和第18采样点进行的。1.5个的停止位可以被分成2部分：一个是0.5个时钟周期，期间不做任何事情。随后是1个时钟周期的停止位，在这段时间的中点处采样。
- 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。

#### 27.2.4. 分数波特率的产生

接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx/Rx 波特率} = \frac{f_{ck}}{(16 * \text{USARTDIV})}$$

这里的  $f_{ck}$  是给外设的时钟(PCLK1 用于 USART2、3、4、5， PCLK2 用于 USART1)USARTDIV 是一个无符号的定点数。这 12 位的值设置在 USART\_BRR 寄存器。

注：在写入 USART\_BRR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

表 27-2 设置波特率时的误差计算

波特率		Fpclk = 8MHz			Fpclk = 36MHz			Fpclk = 100MHz		
序号	kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
1	2.4	2.400	208.375	0.02%	2.400	937.5	0.00%	2.39998	2604.1875	0.00%
2	9.6	9.604	52.0625	0.04%	9.600	234.375	0.00%	9.59417	651.4375	0.06%
3	19.2	19.198	26.04375	0.01%	19.200	117.1875	0.00%	19.2012	325.5000	0.01%
4	57.6	57.554	8.6875	0.08%	57.600	39.0625	0.00%	57.6037	108.5000	0.01%
5	115.2	115.942	4.3125	0.64%	115.385	19.5	0.16%	115.207	54.2500	0.01%

波特率		Fpclk = 8MHZ			Fpclk = 36MHZ			Fpclk = 100MHZ		
序号	kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%	实际	置于波特率寄存器中的值	误差%
6	230.4	228.571	2.1875	0.80%	230.769	9.75	0.16%	230.415	27.1250	0.01%
7	460.8	470.588	1.0625	2.08%	461.538	4.875	0.16%	460.829	13.5625	0.01%
8	921.6		不可能		923.077	2.4375	0.16%	925.926	6.7500	0.47%
9	2250		不可能		2250.000	1	0.00%	2272.73	2.7500	1.00%
10	4500		不可能	不可能	不可能	不可能	不可能	4545.45	1.3750	1.00%

注:

- 1) CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。
- 2) 只有 USART1 使用 PCLK2。其它 USART 使用 PCLK1。
- 3) 在配置时钟的时候，由于波特率时钟是在系统时钟上进行分频的，但得到的时钟频率与实际时钟频率会有误差。配置的时钟误差需要在 2%以内，才可以正常工作。在 fpclk 在 100MHZ 时，不可以设置 baud 时钟大于 4.5MHZ。

### 27.2.5. USART 接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围， USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化(包括发送器端振荡器的变化)
- DQUANT：接收器端波特率取整所产生的误差
- DREC：接收器端振荡器的变化
- DTCL：由于传输线路产生的变化(通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成)。

需要满足：  $DTRA + DQUANT + DREC + DTCL < \text{USART 接收器的容忍度}$

### 27.2.6. USART 自动波特率检测

#### ■ 自动波特率检测

USART能够基于接收到的字符自动检测波特率，并配置波特率寄存器USART\_BRR.

自动波特率检测适用于如下情况：

系统通讯速度预先未知；

系统使用精度较低的时钟源，这种机制允许在不测量时钟偏差的情况下获得正确的波特率；

时钟源频率必须与期望的通讯速度兼容（过采样16的波特率必须在 $f_{CK}/65535$ 和 $f_{CK}/16$ 之间）。

通过USART\_CR2.ABRMOD配置自动波特率检测模式。波特率需要检测几次，每次的测量值与前一次做比较。

#### ■ 自动波特率检测模式

Mode0：此模式针对接收字节以1为起始的情况。这种情况下，是测量start bit到bit1（下降沿到上升沿）。

Mode1：此模式针对以10xx位模式开头的任何字符。在这种情况下，USART测量开始和第一个数据位的持续时间。测量是沿下降沿到下降沿进行的，确保在慢信号倾斜的情况下有更好的精度。

并行数据模式：对RX线的每个中间转换执行的检查。如果RX上的转换没有与接收器充分同步(接收器基于0位计算

的波特率), 就会产生一个错误。

### 27.2.7. 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。
- USART\_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

根据 USART\_CR1 寄存器中的 WAKE 位状态, USART 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位: 进行空闲总线检测。
- 如果 WAKE 位被设置: 进行地址标记检测。

#### 27.2.7.1. 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

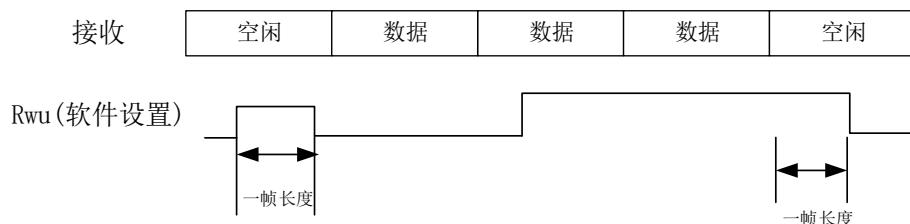


图 27-5 利用空闲总线检测的静默模式

#### 27.2.7.2. 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 27-6 利用地址标记检测的静默模式

### 27.2.8. 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 27-3 帧格式

M 位	PCE 位	USART 帧
0	0	起始位 8 位数据 停止位
0	1	起始位 7 位数据 奇偶检验位 停止位
1	0	起始位 9 位数据 停止位
1	1	起始位 8 位数据 奇偶检验位 停止位

偶校验：校验位使得一帧中的7或8个LSB数据以及校验位中1的个数为偶数。例如：数据=00110101，有4个1，如果选择偶校验(在USART\_CR1中的PS = 0)，校验位将是0。

奇校验：此校验位使得一帧中的7或8个LSB数据以及校验位中1的个数为奇数。

例如：数据=00110101，有4个1，如果选择奇校验(在USART\_CR1中的PS = 1)，校验位将是1。

传输模式：如果USART\_CR1的PCE位被置位，写进数据寄存器的数据的MSB位被校验位替换后发送出去(如果选择偶校验偶数个1，如果选择奇校验奇数个1。如果奇偶校验失败，USART\_SR寄存器中的PE标志被置1，并且如果USART\_CR1寄存器的PEIE在被预先设置的话，中断产生。

### 27.2.9. LIN(局域互联网)模式

配置 USART\_CR2.LINEN=1 选择 LIN 模式。在 LIN 模式下，下列位必须保持位 0：

- USART\_CR2 寄存器的 CLKEN;
- USART\_CR3 寄存器的 STOP/SCEN/HDSEL/IREN.

#### 27.2.9.1. 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。

#### 27.2.9.2. 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

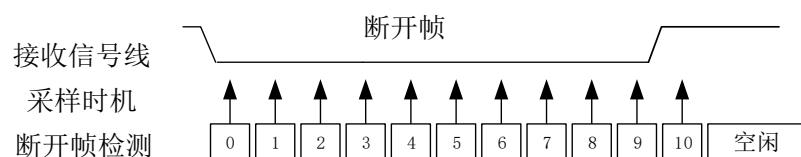
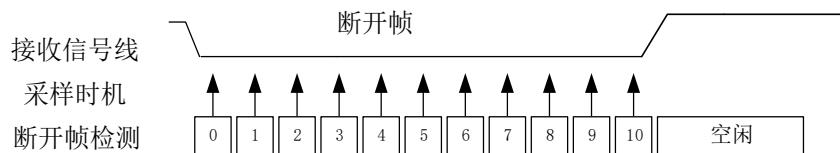


图 27-7 LIN 模式下的断开帧长度不够的情况

本情况：断开帧长度刚好，设置LBD



本情况：断开帧长度很长，设置LBD

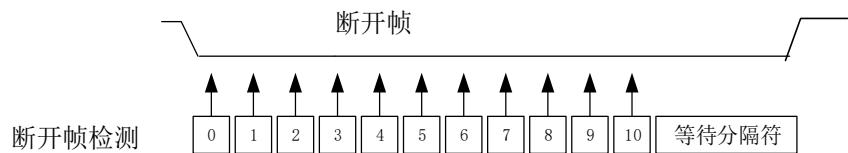
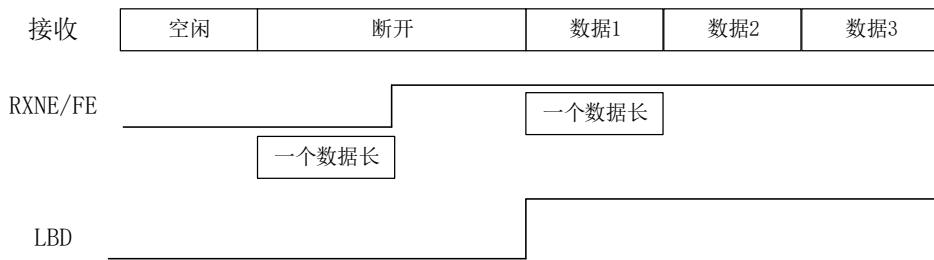


图 27-8 LIN 模式下的断开帧长度够的情况

断开发生在空闲后



断开发生在正在接收数据时

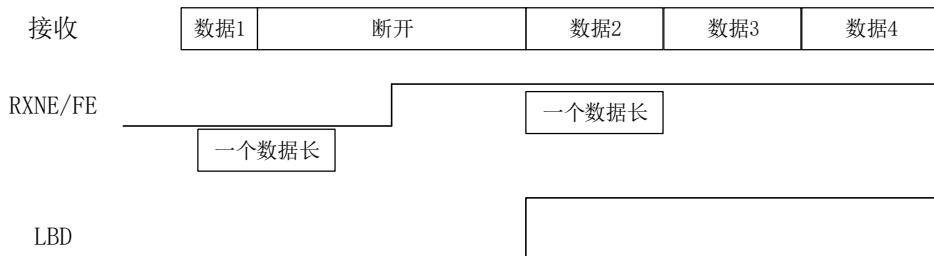


图 27-9 LIN 模式下的断开检测与帧错误的检测

## 27.2.10. 同步模式

通过在 USART\_CR2 寄存器上写 CLKEN 位选择同步模式

在同步模式里，下列位必须保持清零状态：

- USART\_CR2 寄存器中的 LINEN 位
- USART\_CR3 寄存器中的 SCEN,HDSEL 和 IREN 位

USART允许用户以主模式方式控制双向同步串行通信。CK脚是USART发送器时钟的输出。在起始位和停止位期间，CK脚上没有时钟脉冲。根据USART\_CR2寄存器中LBCL位的状态，决定在最后一个有效数据位期间产生或不产生时钟脉冲。USART\_CR2寄存器的CPOL位允许用户选择时钟极性，USART\_CR2寄存器上的CPHA位允许用户选择外部时钟的相位。

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：

- 1) CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器(TE = 1)，并且发送数据时(写入数据至 USART\_DR 寄存器)才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据的。
- 2) LBCL,CPOL 和 CPHA 位的正确配置，应该在发送器和接收器都被禁止时；当使能了发送器或接收器时，这些位不能被改变

- 3) 建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。
- 4) USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据(CK 永远是输出)。

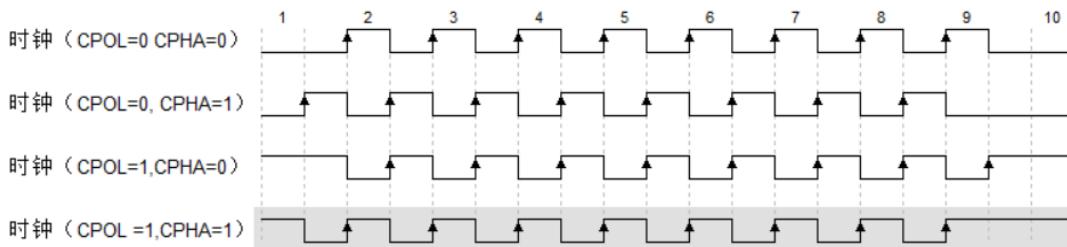


图 27- USART 数据时钟时序示例 (M=0)

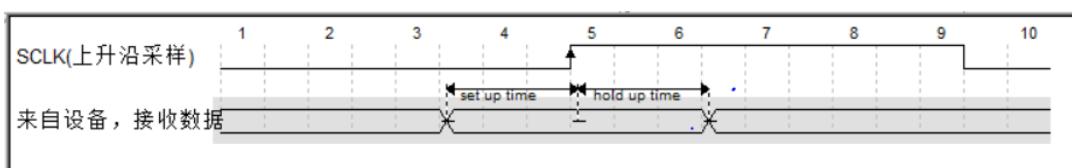


图 27-10 RX 数据采样/保持时间 (M=1)

### 27.2.11. 单线半双工通信

单线半双方模式通过设置 USART\_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USART\_CR2 寄存器的 LINEN 和 CLKEN 位
- USART\_CR3 寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位“HALF DUPLEX SEL”(USART\_CR3 中的 HDSEL 位)选择半双工和全双工通信。

#### 当 HDSEL 为“1”时：

- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味着该 I/O 在不被 USART 驱动时，必须配置成悬空输入(或开漏的输出高)。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

### 27.2.12. 智能卡

设置 USART\_CR3 寄存器的 SCEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USART\_CR2 寄存器的 LINEN 位
- USART\_CR3 寄存器的 HDSEL 位和 IREN 位

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

该接口符合 ISO7816-3 标准，支持智能卡异步协议。 USART 应该被设置为：

- 8 位数据位加校验位：此时 USART\_CR1 寄存器中 M=1、PCE=1
- 发送和接收时为 1.5 个停止位：即 USART\_CR2 寄存器的 STOP=11

注：也可以在接收时选择 0.5 个停止位，但为了避免在 2 种配置间转换，建议在发送和接收时使用 1.5 个停止位。

下图给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

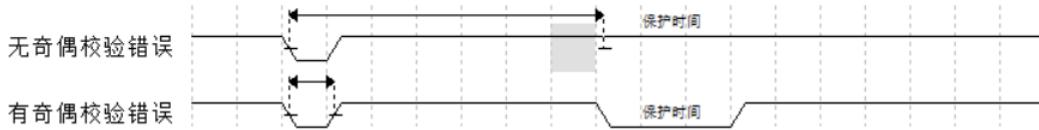


图 27-11 ISO7816-3 异步协议

当与智能卡相连接时，USART 的 TX 驱动一根智能卡也驱动的双向线。为了做到这点，SW\_RX 必须和 TX 连接到相同的 I/O 口。在发送开始位和数据字节期间，发送器的输出使能位 TX\_EN 被置起，在发送停止位期间被释放(弱上拉)，因此在发现校验错误的情况下接收器可以将数据线拉低。如果 TX\_EN 不被使用，在停止位期间 TX 被拉到高电平：这样的话，只要 TX 配置成开漏，接收器也可以驱动这根线。

#### 智能卡是一个单线半双工通信协议

- 从发送移位寄存器把数据发送出去，要被延时最小1/2波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟1/2波特时钟。
- 如果在接收一个设置为0.5或1.5个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后(即停止位结束时)，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到USART的数据没有被正确地接收到。此NACK信号(拉低发送线一个波特时钟周期)在发送端将产生一个帧错误(发送端被配置成1.5个停止位)。应用程序可以根据协议处理重新发送数据。如果设置了NACK控制位，发生校验错误时接收器会给出一个NACK信号；否则就不会发送NACK。
- TC标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC被置起。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC被置高。
- TC标志的撤销不受智能卡模式的影响。
- 如果发送器检测到一个帧错误(收到接收器的NACK信号)，发送器的接收功能模块不会把NACK当作起始位检测。根据ISO协议，接收到的NACK的持续时间可以是1或2波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且NACK被发送，接收器不会把NACK检测成起始位。

#### 注意：

- 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。
- 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。

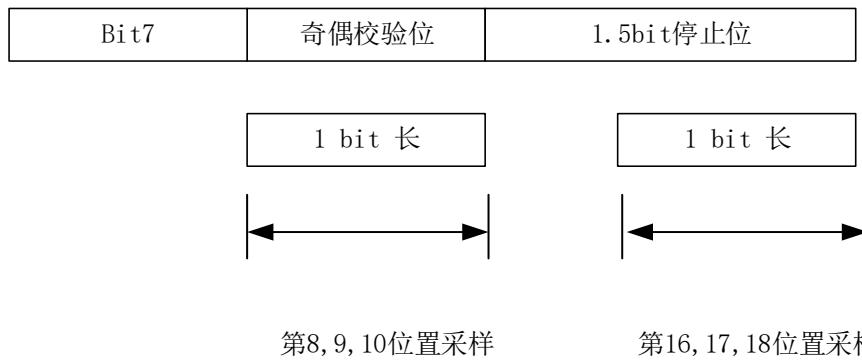


图 27-12 使用 1.5 停止位检测奇偶检验错

USART可以通过CK输出为智能卡提供时钟。在智能卡模式里，CK不和通信直接关联，而是先通过一个5位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器USART\_GTPR中配置。CK频率可以从fCK/2到fCK/62，这里的fCK是外设输入时钟。

### 27.2.13. 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。

#### 27.2.13.1. IrDA 正常模式

IrDA SIR物理层规定使用反相归零调制方案(RZI)，该方案用一个红外光脉冲代表逻辑'0'(见图4-12)。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(见图4-13)。
- SIR接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR输出处于低状态。
- SIR解码器把IrDA兼容的接收信号转变成给USART的比特流。
- IrDA规范要求脉冲要宽于1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于2个PSC周期的脉冲(PSC是在IrDA低功耗波特率寄存器USART\_GTPR中编程的预分频值)。宽度小于1个PSC周期的脉冲一定被滤除掉，但是那些宽度大于1个而小于2个PSC周期的脉冲可能被接收或滤除，那些宽度大于2个周期的将被视为一个有效的脉冲。当PSC=0时，IrDA编码器/解码器不工作。

- 接收器可以与一低功耗发送器通信。
- 在IrDA模式里， USART\_CR2寄存器上的STOP位必须配置成1个停止位。

### 27.2.13.2. 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：

- 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)。

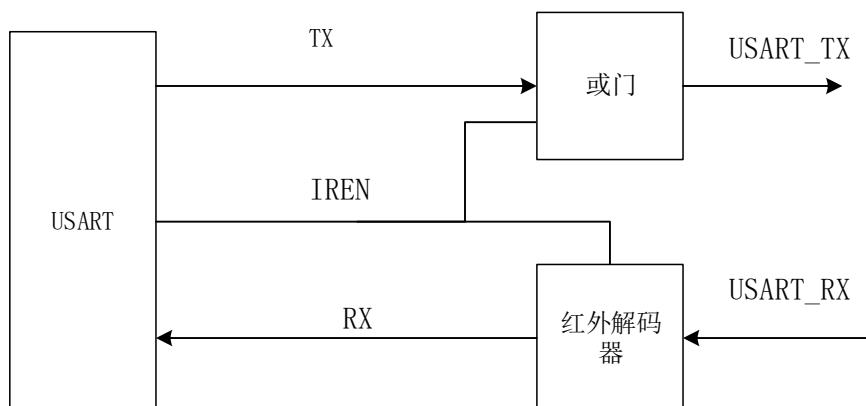


图 27-13 IrDA SIR ENDEC 框图

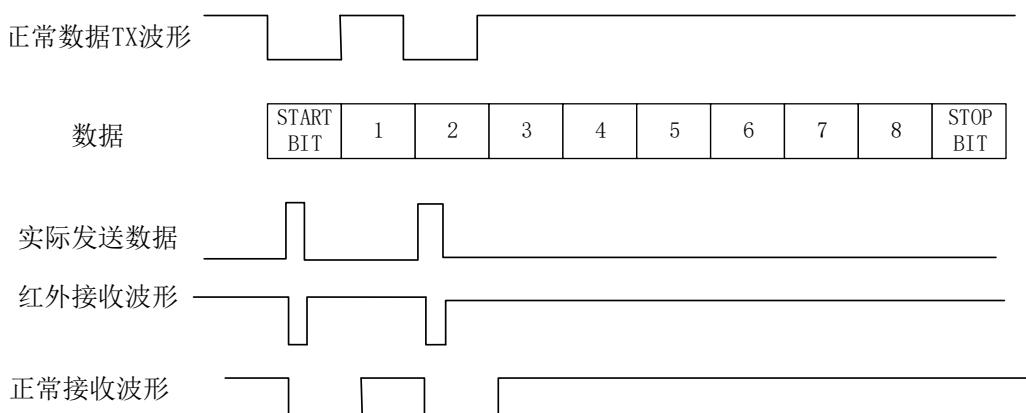


图 27-14 IrDA 数据调制 (3/16) —普通模式

### 27.2.14. 利用 DMA 连续通信

USART可以利用DMA连续通信。 Rx缓冲器和Tx缓冲器的DMA请求是分别产生的。

注意：参考产品技术说明以确定是否可用DMA控制器。在*USART2\_SR*寄存器里，可以清零*TXE/RXNE*标志来实现连续通信。

#### **利用 DMA 发送：**

使用 DMA 进行发送，可以通过设置 USART\_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为'1'时， DMA 就从指定的 SRAM 区传送数据到 USART\_DR 寄存器

- 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。
- 在DMA控制寄存器中配置要传输的总的字节数。
- 在DMA寄存器上配置通道优先级。
- 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。
- 在DMA寄存器上激活该通道。

当传输完成DMA控制器指定的数据量时， DMA控制器在该DMA通道的中断向量上产生一中断。在发送模式下，当 DMA 传输完所有要发送的数据时， DMA 控制器设置 DMA\_ISR 寄存器的 TCIF 标志；监视 USART\_SR 寄存器的 TC 标志可以确认 USART 通信是否结束，这样可以在关闭 USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等待 TC=1。

#### **利用 DMA 接收：**

可以通过设置 USART\_CR3 寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节， DMA 控制器就就把数据从 USART\_DR 寄存器传送到指定的 SRAM 区(参考 DMA 相关说明)。为 USART 的接收分配一个 DMA 通道的步骤如下(x 表示通道号)：

- 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。
- 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。
- 在 DMA 控制寄存器中配置要传输的总的字节数。
- 在 DMA 寄存器上配置通道优先级。
- 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。
- 在 DMA 控制寄存器上激活该通道。

当接收完成 DMA 控制器指定的传输量时， DMA 控制器在该 DMA 通道的中断矢量上产生一中断。

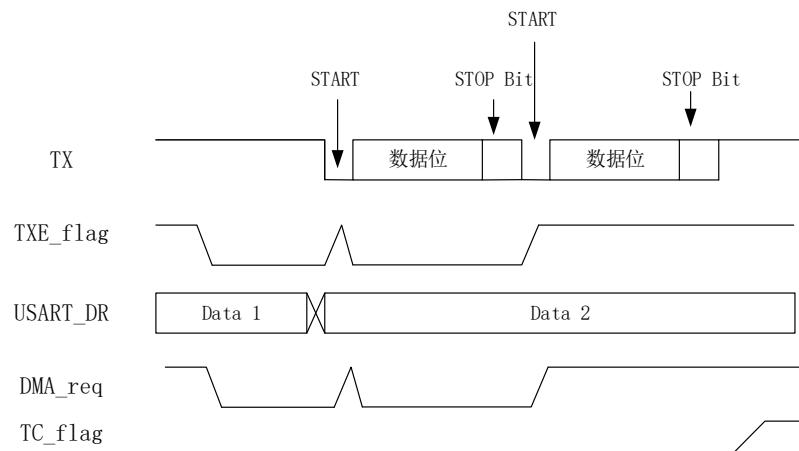


图 27-15 利用 DMA 发送

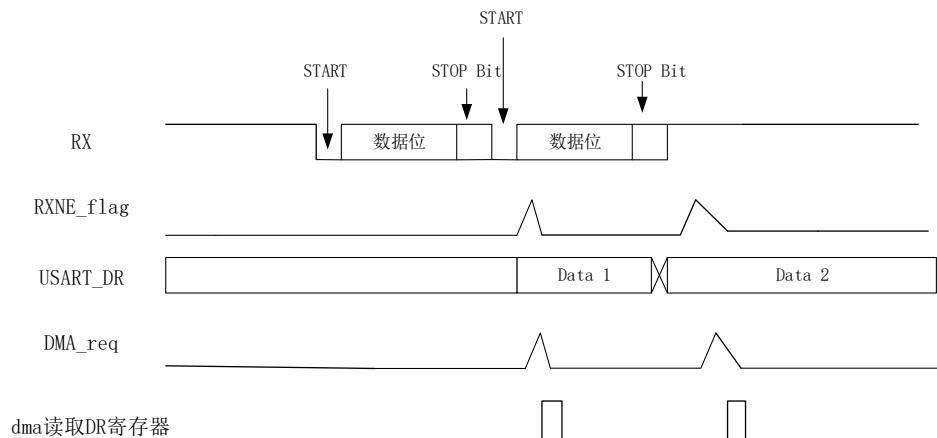


图 27-16 利用 DMA 接收

### 多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。

如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和RXNE一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了，会在当前字节传输结束后，产生中断。

#### 27.2.15. 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。

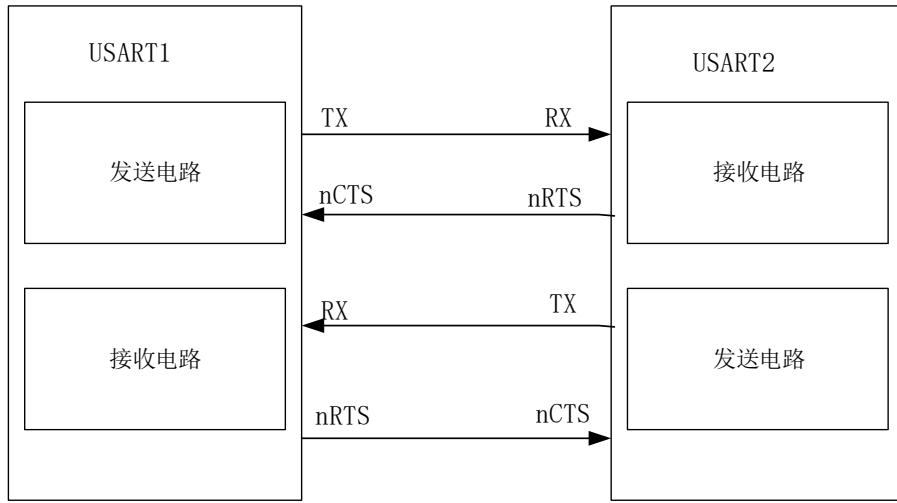


图 27-17 两个 USART 间的硬件流控制

通过将 UASRT\_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

#### RTS 流控制：

如果 RTS 流控制被使能(RTSE=1)，只要 USART 接收器准备好接收新的数据，nRTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

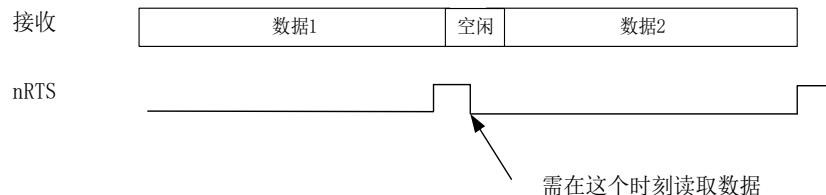


图 27-18 RTS 流控制

#### CTS 流控制

如果 CTS 流控制被使能(CTSE=1)，发送器在发送下一帧前检查 nCTS 输入。如果 nCTS 有效(被拉成低电平)，则下一个数据被发送(假设那个数据是准备发送的，也就是 TXE=0)，否则下一帧数据不被发出去。若 nCTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 nCTS 输入一变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USART\_CT3 寄存器的 CTSIE 位，则产生中断。下图是一个启用 CTS 流控制通信的例子。



图 27-19 CTS 流控制

## 27.2.16. 中断请求

表 3-5 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	传送完成	TC	TCIE	发送
4	接收寄存器未空 (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	NR/ORE/FE	EIE	接收

仅当使用 DMA 接收数据时, 才使用这个标志位。

USART 的各种中断事件被连接到同一个中断向量(见下图), 有以下各种中断事件:

- 发送期间: 发送完成、清除发送、发送数据寄存器空。
- 接收期间: 空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN 断开符号检测、噪音标志(仅在多缓冲器通信)和帧错误(仅在多缓冲器通信)。

如果设置了对应的使能控制位, 这些事件就可以产生各自的中断。

电路实现为:

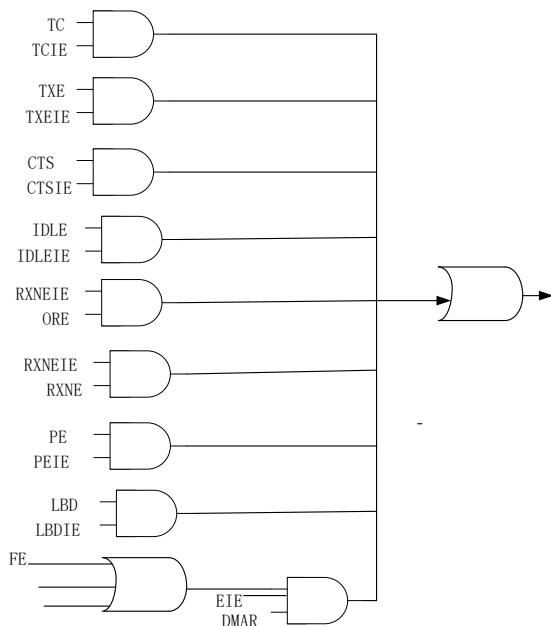


表 27-4 USART 中断映像图

## 27.3. 寄存器描述

### 27.3.1. 控制寄存器 (SR)

地址偏移: 0x00

复位值: 0x00c0

12	11	10	9	8	7	6	5	4	3	2	1	0
ABRRQ	ABRE	ABRF	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
W	R	R	Rc_W0	Rc_W0	R	RC_W0	RC_W0	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RES	\	\
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: CTS 标志 (CTS flag) 如果设置了 CTSE 位，当 nCTS 输入变化状态时，该位被硬件置高。由软件将其清零。如果 USART_CR3 中的 CTSIE 为'1'，则产生中断。 0: nCTS 状态线上没有变化； 1: nCTS 状态线上发生变化。
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	TXE:发送数据寄存器空 (Transmit data register empty) 当 TDR 寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。 如果 USART_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 USART_DR 的写操作，将该位清零。 0: 数据还没有被转移到移位寄存器； 1: 数据已经被转移到移位寄存器。 注意：单缓冲器传输中使用该位。
6	TC	RC_W0	1	发送完成 (Transmission complete)

Bit	Name	R/W	Reset Value	Function
				<p>当包含有数据的一帧发送完成后，并且 TXE=1，由硬件将该位置为 1。如果 USART_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 USART_DR 的写操作，将该位清零。</p> <p>0：发送还未完成； 1：发送完成。</p>
5	RXNE	RC_W0	0	<p>读数据寄存器非空 (Read data register not empty)</p> <p>当 RDR 移位寄存器中的数据被转移到 USART_DR 寄存器中，该位被硬件置位。如果 USART_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 USART_DR 的读操作可以将该位清零。RXNE 位也可以通过写入 0 来清除，只有在多缓存通讯中才推荐这种清除程序。</p> <p>0：数据也没有收到 1：收到数据，可以读出</p>
4	IDLE	R	0	<p>IDLE：监测到总线空闲</p> <p>当监测到总线空闲时，该位被硬件置位。如果 USART_CR1 中 IDLEIE 为 1，则产生中断。由于软件序列清除该位（先读 USART_SR;然后读 USART_DR）。</p> <p>0：没有检测到空闲总线； 1：检测到空闲总线</p> <p>注意：IDLE 位不会再次被置高直到 RXNE 位被置起(即又检测到一次空闲总线)</p>
3	ORE	R	0	<p>ORE (overrun error)</p> <p>当 RXNE=1 时，当前被接收在移位寄存器中的数据，需要传送到 RDR 寄存器时，硬件将该位置位。如果 USART_CR1 中的 RXNEIE 则产生中断。由软件序列将其清零(先读 USART_SR，然后读 USART_CR)。</p> <p>0：没有过载错误； 1：检测到过载错误。</p> <p>注意：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 EIE 位，在多缓冲器通信模式下，ORE 标志置位会产生中断的。</p>
2	NE	R	0	<p>NE: 噪声错误标志 (Noise error flag)</p> <p>在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清零(先 USART_SR，再读 USART_DR)。</p> <p>0：没有检测到噪声； 1：检测到噪声。</p> <p>注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。</p> <p>在多缓冲区通信模式下，如果设置了 EIE 位，则设置 NE 标志时会产生中断</p>
1	FE	R	0	<p>FE:帧错误 (Framing error)</p> <p>当检测到同步错位，过多的噪声或者检测到断开符，该位被硬件置位。由软件序列将其清零(先读 USART_SR,在读 USART_DR)。</p> <p>0:没有检测到帧错误；</p>

Bit	Name	R/W	Reset Value	Function
				<p>1: 检测到帧错误或者 break 符。</p> <p>注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。</p> <p>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。</p> <p>在多缓冲区通信模式下，如果设置了 EIE 位，则设置 FE 标志时会产生中断。</p>
0	PE	R	0	<p>PE:校验错误 (Parity error)</p> <p>在接收模式下，如果出现奇偶校验错误，硬件对该位置位。由软件序列对其清零（依次读 USART_SR 和 USART_DR）。在清除 PE 位前，软件必须等待 RXNE 标志位被指 1，如果 USART_CR1 中的 PEIE 为 1，则产生中断。</p> <p>0: 没有奇偶校验错误； 1: 奇偶校验错误。</p>

### 27.3.2. 数据寄存器 (USART\_DR)

地址偏移: 0x04

复位值: 0xxxxx

31:9	8	7	6	5	4	3	2	1	0
DR[8:0]									
	RW								

Bit	Name	R/W	Reset Value	Function
31:9	Reserve	RES	\	\
8:0	DR[8:0]	RW	Undefined	<p>接收/发送数据寄存器。一共是由两个寄存器组成（一个是发送的 TDR, 一个是接收的 RDR），所以 DR 寄存器实现了读和写的两个功能。</p> <p>TDR 寄存器提供了内部总线和输出移位寄存器之间的并行接口（参见图 248）。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。</p> <p>当使能校验位 (USART_CR1 中 PCE 位置位) 进行发送时，写到 MSB 的值（根据数据的不同长度，MSB 是第 7 位或者第 8 位）会被后来的校验位所取代。</p>

### 27.3.3. 波特比率寄存器 (USART\_BRR)

注意：如果 TE 或 RE 被分别禁止，波特计数器停止计数

地址偏移: 0x08

复位值: 0x0000

31:16	15:4	3:0
保留位	DIV_Mantissa[11:0]	DIV_Fraction[3:0]
RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	\	Reserved
15:4	DIV_Mantissa[15:4]	RW	0	12 bits 整数
3:0	DIV_Fraction[3:0]	RW	0	4 bits 小数

### 27.3.4. 控制寄存器 (USART\_CR)

#### 27.3.4.1. 控制寄存器 1 (USART\_CR1)

地址偏移: 0x0C

Reset Value: 0x0000\_0000

31:14															
Res															
13	12	11	10	9	8	7	6	5	4	3	2	1	0		
UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:14	Reserved	RES	\	Reserved
13	UE	RW	0	UE:USART 使能 (USART enable) 当该位被清零，在当前字节传输完成后 USART 的分频器和输出停止工作，以此减少功耗。该位由软件置位和清零。 0: USART 分频器和输出被禁止； 1: USART 模块使能。
12	M	RW	0	M: 字长 该位定义了数据字的长度，由软件对其进行设置和清零。 0: 一个起始位，8个数据位，n个停止位； 1: 一个起始位，9个数据位，n个停止位。 在数据传输过程中，不能修改这个位。
11	WAKE	RW	0	Wake:唤醒的方法 (Wakeup method) 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 0: 被空闲总线唤醒； 1: 被地址标记唤醒。
10	PCE	RW	0	PCE:校验控制使能 (Parity control enable) 用该位选择是否进行硬件校验控制（对于发送来说就是校验位的产生；对于接收来说就是校验位的检测）。当使能了该位，在发送数据的最高位（如果 M=1,最高位就是第 9 位；如果 M=0, 最高位就是第 8 位）插入校验位；对接收到的数据检查其校验位。软件对它置“1”或清“0”。一旦设置了该位，当前字节传输完成后，校验控制才生效。 0: 禁止校验控制； 1: 使能校验控制。
9	PS	RW	0	PS:校验选择 (Parity selection)

Bit	Name	R/W	Reset Value	Function
				当校验控制使能后，该位用来选择是采用偶校验还是奇校验。软件对它置“1”或者清“0”。当前字节传输完成后，该选择生效。 0：偶校验； 1：奇校验。
8	PEIE	RW	0	PEIE:PE 中断使能 (PE interrupt enable) 由软件置位和清零。 0：禁止产生中断 1：PE 中断使能
7	TXEIE	RW	0	TXE: TXE 中断使能 软件置位和清零。 0：禁止； 1：TXE 中断使能
6	TCIE	RW	0	TCIE:发送完成中断使能 该位由软件设置和清零。 0:禁止产生中断。 1:当 USART_SR 中的 ORE 或者 RXNE 为 1 时，产生 USART 中断。
5	RXNEIE	RW	0	RXNEIE:接收缓冲区非空中断使能 (RXNE interrupt enable) 该位由软件设置或清除。 0：禁止产生中断 1：当 USART_SR 中的 ORE 或者 RXNE 为“1”时，产生 USART 中断。
4	IDLEIE	RW	0	IDLE:IDLE 中断使能 (IDLE interrupt enable) 该位由软件设置或清除 0：禁止产生中断； 1：当 USART_SR 中的 IDLE 为“1”时，产生 USART 中断
3	TE	RW	0	TE:发送使能 (Transmitter enable) 该位使能发送器。该位由软件设置或清除 0：禁止发送； 1：使能发送。 在数据传输过程中，除了在智能卡模式下，如果 TE 位上有个 0 脉冲(即设置为‘0’之后，再设置为‘1’)，会在当前数据字传输完成后，发送一个“前导符”(空闲总线)。当 TE 被设置后，在真正发送开始之前，有一个比特时间的延迟。
2	RE	RW	0	RE:接收使能 (Receiver enable) 0：禁止接收； 1：使能接收，并开始搜寻 RX 引脚的起始位。
1	RWU	RW	0	RWU:接收唤醒 (Receiver wakeup) 该位用来决定是否把 USART 置于静默模式。该位由软件设置或清除。当唤醒序列到来时，硬件也会将其清零。

Bit	Name	R/W	Reset Value	Function
				0: 接收器处于正常工作模式; 1: 接收器处于静默模式。 在把 USART 置于静默模式(设置 RWU 位)之前, USART 已经先接收了一个数据字节。否则在静默模式下, 不能被空闲总线检测唤醒。当配置成地址标记检测唤醒(WAKE 位=1), 在 RXNE 位被置位时, 不能用软件修改 RWU 位
0	SBK	RW	0	SBK:发送断开帧 (Send break) 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。

#### 27.3.4.2. 控制寄存器 2 (USART\_CR2)

地址偏移: 0x10

Reset Value: 0x0000\_0000

14	13	12	11	10	9	8	7	6	5	4	[3: 0]
LINEN	STO P	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	RES	-	保留
14	LINEN	RW	0	LIN 模式使能。 软件置位和清零。 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	2'b0	Stop 位配置。 00: 1 stop bit; 01: 0.5stop; 10: 2 stop 位; 11: 1.5stop;
11	CLKEN	RW	0	时钟使能, 该位用来使能 CK 引脚。 0: CK 引脚禁止; 1: CK 引脚使能; 不支持同步模式时, 该位保留。
10	CPOL	RW	0	时钟极性。 同步模式时, 用该位选择 SLCK 引脚上时钟输出的极性。和 CPHA 位一起配合来产生需要的时钟/数据的采样关系 0: 总线空闲时 CK 引脚上保持低电平;

Bit	Name	R/W	Reset Value	Function
				1: 总线空闲时 CK 引脚上保持高电平。
9	CPHA	RW	0	时钟相位 在同步模式下，可以用该位选择 SLCK 引脚上时钟输出的相位。和 CPOL 位一起配合来产生需要的时钟/数据的采样关系 0: 在时钟的第一个边沿进行数据捕获； 1: 在时钟的第二个边沿进行数据捕获。
8	LBCL	RW	0	最后一位时钟脉冲 在同步模式下，使用该位来控制是否在 CK 引脚上输出最后发送的那个字节 (MSB) 对应的时钟脉冲。 0: 最后一位数据的时钟脉冲不在 CK pin 输出； 1: 最后一位数据的时钟脉冲在 CK pin 输出。 注：最后一位数据位就是第八个或者第九个发送的位（由 USART_CR1 中的 M 位决定）
7	保留	RES	-	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	RES	-	保留
3: 0	ADD[3:0]	RW	4'b0	USART 地址。 该寄存器用于多处理器静默模式，用作 4bit 地址唤醒时的地址。

#### 27.3.4.3. 控制寄存器 3 (USART\_CR3)

地址偏移: 0x14

Reset Value: 0x0000\_0000

14:13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABR-MOD	ABREN	OVER8	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	保留	RES		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；

Bit	Name	R/W	Reset Value	Function
				当 ABREN=0 或者 UE=0 时，该寄存器只写。 注：本芯片只支持 00 和 01 两种模式。
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	Oversampling 模式。 0：Oversampling by 16； 1：Oversampling by 8； 注：该位只有当 UE=0 时才能写。
10	CTSIE	RW	0	CTS 中断使能。 0：禁止； 1：CTS 为 1 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，只有当接收缓冲区未满时才会请求下一个数据。当前数据发送完成后，发送操作暂停。如果可以接收数据了，将 RTS 置为有效 (0)。
7	DMAT	RW	0	DMA 使能发送。 0：禁止； 1：传送时使能 DMA；
6	DMAR	RW	0	DMA 使能接收。 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；

Bit	Name	R/W	Reset Value	Function
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 NF 中断使能;

### 27.3.5. 保护时间和预分频寄存器 (USART\_GTPR)

地址偏移: 0x18

Reset Value: 0x0000\_0000

[31:16]	[15:8]	[7:0]
Reserve	GT	PSC
RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	保留	RES	-	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	预分频器值 (Prescaler value)。 1. 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; ..... 2. 在红外(IrDA)的正常模式下： PSC 只能设置为 00000001。 3. 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;

Bit	Name	R/W	Reset Value	Function
				00011: 对源时钟进行 6 分频; ..... 注意: 位[7:5]在智能卡模式下没有意义。

### 27.3.6. USART 寄存器映射

# 28. 时钟校准控制器 (CTC)

## 28.1. 简介

时钟校准控制器 (CTC) 采用硬件的方式，自动校准内部 48MHz RC 晶振 (HSI48M)。当 USBD 模块使用 HSI48M 时钟作为时钟源时，必须要求 HSI48M 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 HSI48M 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 HSI48M 时钟。

### 28.1.1. 主要特征

CTC 模块主要完成如下功能：

- 三个外部参考信号源：GPIO，LSE 时钟，USBD\_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) 和错误状态 (ERRIF)。

### 28.1.2. 模块框图

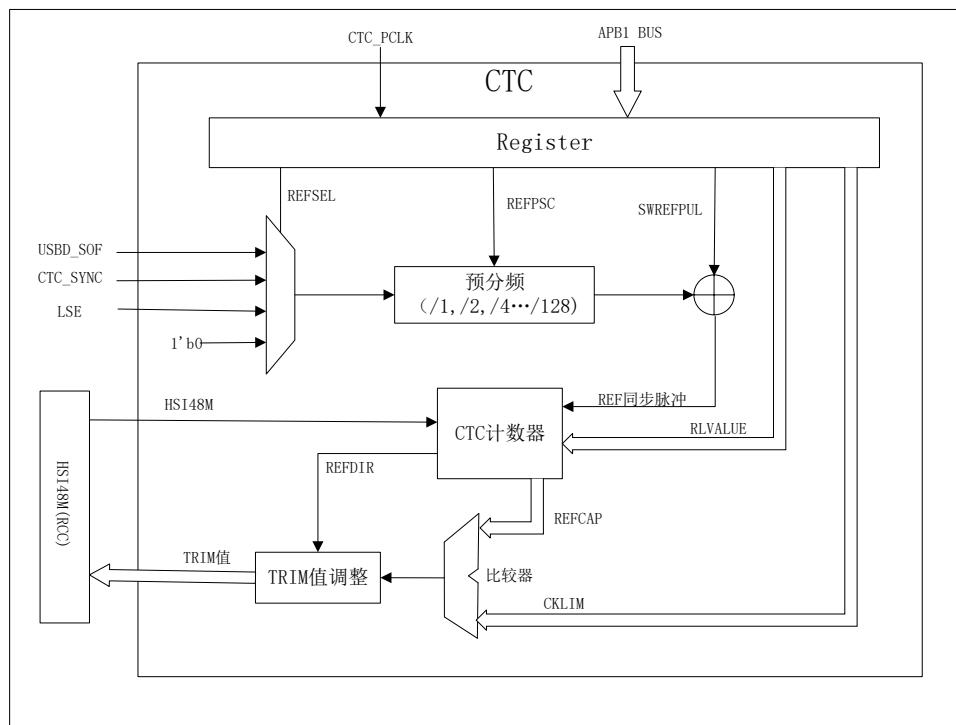


图 28-1 CTC 模块框图

## 28.2. 功能描述

### 28.2.1. REF 同步脉冲发生器

首先，通过设置 CTC\_CTL1 寄存器（CTC 控制寄存器 1）中的 REFSEL 位来选择参考信号源：GPIO，LSE 时钟输出或者 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。

如果需要使用软件参考脉冲信号，则需要设置 CTC\_CTL0 寄存器（CTC 控制寄存器 0）中的 SWREFPUL 位为 1。软件参考脉冲信号与外部参考脉冲信号最后进行逻辑‘或’操作。

### 28.2.2. CTC 校准计数器

CTC 时钟校准计数器由 HSI48M 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  $128 \times CKLIM$ （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\_STAT CTC 状态寄存器）中的 REFCAP 位，同时，当前计数器的计数方向被存入 CTC\_STAT 中的 REFDIR 位。详细内容如下图所示。

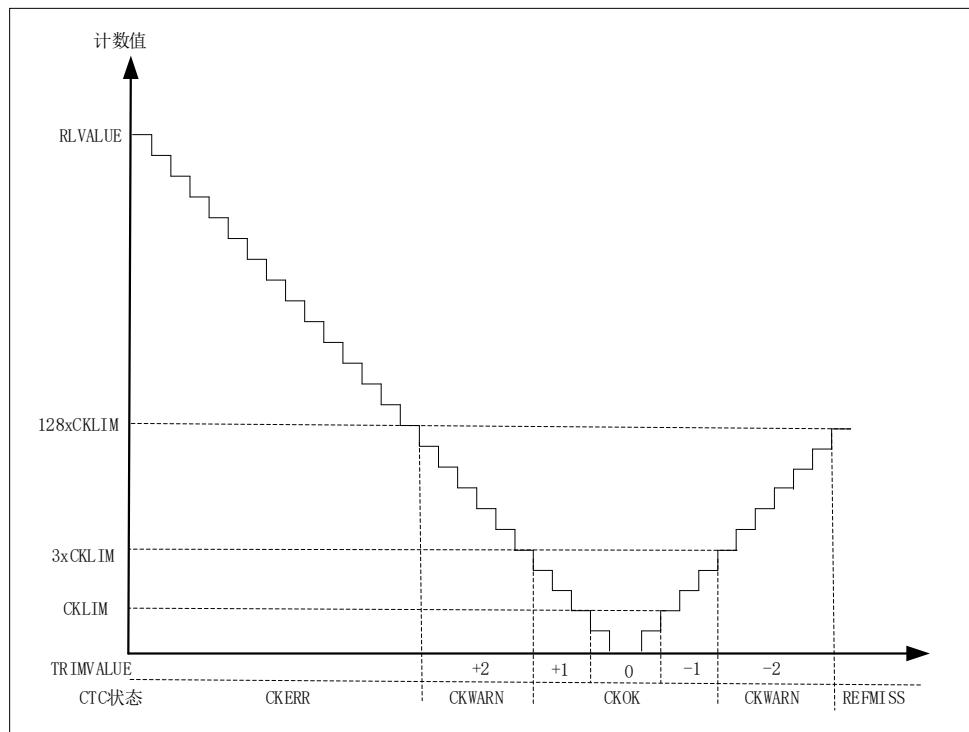


图 28-2 CTC 校准计数器

### 28.2.3. 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 48M）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\_STAT 中的 CKOKIF 位，CKWARNIF 位，CKERR 位和 REFMISST 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\_STAT 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 × CKLIM 时，检测到 REF 同步脉冲信号：
  - CTC\_STAT 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 × CKLIM ≤ Counter < 128 × CKLIM 时，检测到 REF 同步脉冲信号：
  - CTC\_STAT 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 × CKLIM，计数器在向下计数过程中，检测到 REF 同步脉冲信号：
  - CTC\_STAT 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 × CKLIM，计数器在向上计数过程中：
  - CTC\_STAT 中的 REFMISST 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\_STAT 中的 TRIMERR 位（校准值错误位）将被置位，如果 CTC\_CTL0 中的 ERRIE 位置 1，将会产生一个中断。

#### 28.2.4. 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（HSI48M：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：

$$\text{RLVALUE} = (\text{F}_{\text{clock}} \div \text{F}_{\text{REF}}) - 1$$

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

$$\text{CKLIM} = (\text{F}_{\text{clock}} \div \text{F}_{\text{REF}}) \times 0.12\% \div 2$$

典型的步长值是 0.12%， $\text{F}_{\text{clock}}$  是期望时钟的频率 (48MHz)， $\text{F}_{\text{REF}}$  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-48M 时钟的频率，因此，不应该随意通过软件修改 TRIMVALUE。建议用户在两次参考信号中间，根据标志位判断修改；或者若已经存在可靠的值，用户可以直接修改 RCC\_CFGR1 中 HSI48TRIM 的值。

## 28.3. 寄存器描述 (基址 0x4000\_C800)

### 28.3.1. 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]								SWREFPUL	AUTOTRIM	CNTEN	Res.	EREFIE	ERRIE	CKWARNIE	CKOKIE
	RW	RW	RW	RW	RW	RW	RW	W	RW	RW		RW	RW	RW	RW	

Bit	Name	R/W	Reset Value	Function
31:15	Reserved	RES	-	Reserved
14:8	TRIMVALUE[6:0]	RW	7'b1000000	HSI48M 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，HSI48M 时钟频率增加大约 57KHz。当 TRIMVALUE 值减 1 时，HSI48M 时钟频率的减少大约 57KHz。
7	SWREFPUL	W	0	软件生成同步参考信号脉冲。 该位由软件置位，并为 CTC 计数器提供一个同步参考脉冲信号。该位由硬件自动清除，读操作时返回 0。 0：没有影响； 1：软件产生一个同步参考脉冲信号；
6	AUTOTRIM	RW	0	硬件自动校准模式。

Bit	Name	R/W	Reset Value	Function
				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 HSI48M 的时钟频率达到 48MHz。 0：禁止硬件自动校准模式 1：使能硬件自动校准模式
5	CNTEN	RW	0	CTC 计数器使能。 该位由软件置位或清除，用于使能或禁止 CTC 计数器。当该位置 1 时，不能修改 CTC_CTL1 的值。 0：禁止 CTC 计数器 1：使能 CTC 计数器
4	Reserved	RES	-	Reserved
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

### 28.3.2. 控制寄存器 1(CTC\_CTL1)

**Address offset:** 0x04

**Reset value:** 0x2022 BB7F

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

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

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

Bit	Name	R/W	Reset Value	Function											
				1: 选择下降沿											
30	Reserved	RES	-	Reserved											
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	Reserved	RES	-	Reserved											
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。											

### 28.3.3. 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
REFDR	Res.				TRIMERR	REFMISS	CKERR	Res.				EREFIF	ERRIF	CKWARNIF	CKOKIF
R					R	R	R					R	R	R	R

Bit	Name	R/W	Reset Value	Function							
31:16	REFCAP[15:0]	R	0x0000	CTC 计数器捕获值。							

Bit	Name	R/W	Reset Value	Function
				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	<p>CTC 校准时钟计数方向。</p> <p>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。</p> <p>0：向上计数 1：向下计数</p>
14:11	Reserved	RES	-	Reserved
10	TRIMERR	R	0	<p>校准值错误位。</p> <p>当 CTC_CTL0 中的 TRIMVALUE 值发生上溢或下溢时，该位由硬件置位。若 CTC_CTL0 中的 ERRIE 位置 1，则会产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 TRIMERR 位清零。</p> <p>0：无校准值错误发生 1：发生校准值错误</p>
9	REFMISS	R	0	<p>同步参考脉冲信号丢失。</p> <p>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 <math>128 \times CKLIM</math> 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，或者其他错误产生。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 REFMISS 位清零。</p> <p>0：无同步参考脉冲信号丢失 1：同步参考脉冲信号丢失</p>
8	CKERR	R	0	<p>时钟校准错误位。</p> <p>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 <math>128 \times CKLIM</math>，并检测到同步参考脉冲信号时，CKERR 置位，说明当前时钟太慢，无法校准到期望频率值。当 CTC_CTL0 中的 ERRIE 置 1 时，产生一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 CKERR 位清零。</p> <p>0：无时钟校准错误发生 1：发生时钟校准错误</p>
7:4	Reserved	RES	-	Reserved
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</p>

Bit	Name	R/W	Reset Value	Function
				<p>一个中断。通过写 1 到 CTC_INTC 中的 ERRIC 位，可以将 EERRIF 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>

#### 28.3.4. 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.												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.												
												EREFIC	ERRIC	CKWARNIC	CKOKIE
												W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	RES	-	Reserved
3	EREFIC	W	0	<p>EREFIF 中断清除位。</p> <p>该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 EREFIF 位，写 0 没影响。</p>
2	ERRIC	W	0	ERRIF 中断清除位。

Bit	Name	R/W	Reset Value	Function
				该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 ERRIF 位，TRIMERR 位，REFMISS 位和 CKERR 位，写 0 没影响。
1	CKWARNIC	W	0	CKWARNIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 CKWARNIF 位，写 0 没影响。
0	CKOKIC	W	0	CKOKIF 中断清除位。 该位只能由软件写，读操作返回 0。写 1 可以清除 CTC_STAT 中的 CKOKIF 位，写 0 没影响

### 28.3.5. CTC 寄存器映射

0x0C		0x08		0x04		0x00		偏移	
复位值	CTC_INTC	复位值	CTC_SR	复位值	CTC_CTL1	复位值	CTC_CTL0	寄存器	
Res		Res		0	REFPOL		Res	31	
Res		Res			Res		Res	30	
Res		Res		1	REFSEL[1:0]		Res	29	
Res		Res		0			Res	28	
Res		Res			Res		Res	27	
Res		Res		0	REFPSC[2:0]		Res	26	
Res		Res		0			Res	25	
Res	0	REFCAP[15:0]		0			Res	24	
Res							Res	23	
Res							Res	22	
Res							Res	21	
Res							Res	20	
Res							Res	19	
Res							Res	18	
Res							Res	17	
Res							Res	16	
Res	1	REFDIR					Res	15	
Res		Res					1	14	
Res		Res					0	13	
Res		Res					0	12	
Res		Res					0	11	
Res	0	TRIMERR					0	10	
Res	0	REFMISS					0	9	
Res	0	CKERR					0	8	
Res								SWREFPL	7
Res								AUTOTRIM	6
Res								CNTEN	5
Res								Res	4
0	EREFIC	0	EREFIF					EREFIE	3
0	ERRIC	0	ERRIF					ERRIE	2
0	CKWARNIC	0	CKWARNIF					CKWARNIE	1
0	CKOKIC	0	CKOKIF					CKOKIE	0

# 29. 调试支持 (DBG)

## 29.1. 简介

MCUDBG 模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器、看门狗、I2C 和 CAN 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。

### 29.1.1. 主要特性

- 支持睡眠模式，停止模式和待机模式
- CPU 进入 HALT 时，控制定时器、看门狗停止计数或者继续计数
- CPU 进入 HALT 时，阻止 I2C1 和 I2C2 SMBUS 超时
- CPU 进入 HALT 时，阻止 CAN 的接收寄存器更新
- 分配跟踪引脚

## 29.2. 功能描述

### 29.2.1. 低功耗模式的调试中支持

使用 WFI 和 WFE 可以进入低功耗模式。

MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。

内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU\_CR 寄存器的 DBG\_SLEEP 位。这将为 HCLK 提供与 FCLK(由代码配置的系统时钟)相同的时钟。
- 在停止模式下，调试器必须先置位 DBG\_STOP 位。这将激活 HSI8M 时钟，在停止模式下为 FCLK 和 HCLK 提供时钟。

## 29.3. 寄存器 baseaddr=0xE0042000

### 29.3.1. ID 编码 (DBGMCU\_IDCODE) (0xE004\_2000)

Address offset: 0x00

Reset value: 0xXXXX XXXX

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

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

Bit	Name	R/W	Reset Value	Function
31:0	REV_ID	R	x	MCU 产品仿真器 ID。该值保存在 FLASH information 区。

### 29.3.2. 调试配置寄存器 (DBGMCU\_CR) (0xE004\_2004)

该寄存器有 VDDD 域上电复位复位，系统复位不会复位该寄存器。当内核处于复位状态下时，调试器可以些该寄存器。

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	DBG_T IM11_S TOP	DBG_T IM10_S TOP	DBG_T IM9_S STOP	DBG_T IM14_S TOP	DBG_T IM13_S TOP	DBG_TI M12_S TOP	Res				DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TI MEOUT
	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS_TI MEOUT	DBG_C AN_ST OP	DBG_T IM4_ST OP	DBG_T IM3_ST OP	DBG_T IM2_ST OP	DBG_T IM1_ST OP	DBG_W WDG_S TOP	DBG_I WDG_STOP	TRACE [1:0]	MODE CE_I OEN	TRA CE_I OEN	Res		DBG_STDB_Y	DBG_STOP	DBG_SLEEP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW					RW

Bit	Name	R/W	Reset Value	Function
31	保留	RES	-	保留
30	DBG_TIM11_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM11 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
29	DBG_TIM10_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM10 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
28	DBG_TIM9_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM9 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；

Bit	Name	R/W	Reset Value	Function
27	DBG_TIM14_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM14 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
26	DBG_TIM13_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM13 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
25	DBG_TIM12_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM12 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
24:21	保留	RES	-	保留
20	DBG_TIM7_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM7 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
19	DBG_TIM6_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM6 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
18	DBG_TIM5_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM5 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
17	DBG_TIM8_STOP	RW	0	当 CPU 核处于 halt 状态时，控制 TIM8 计数停止。 0: CPU halt 时，计数器时钟使能，正常计数； 1: CPU halt 时，计数器时钟禁止，停止计数，输出禁止；
16	DBG_I2C2_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C2 SMBUS 超时模式停止。 0: CPU halt 时，I2C2 与正常模式相同； 1: CPU halt 时，I2C2 SMBUS 超时功能禁止；
15	DBG_I2C1_SMBUS_TIMEOUT	RW	0	当 CPU 核处于 halt 状态时，控制 I2C1 SMBUS 超时模式停止。

Bit	Name	R/W	Reset Value	Function
				0: CPU halt 时, I2C1 与正常模式相同; 1: CPU halt 时, I2C21 SMBUS 超时功能禁止;
14	DBG_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;
13	DBG_TIM4_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM4 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
12	DBG_TIM3_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM3 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
11	DBG_TIM2_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM2 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
10	DBG_TIM1_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 TIM1 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
9	DBG_WWDG_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 WWDG 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
8	DBG_IWDG_STOP	RW	0	当 CPU 核处于 halt 状态时, 控制 IWDG 计数停止。 0: CPU halt 时, 计数器时钟使能, 正常计数; 1: CPU halt 时, 计数器时钟禁止, 停止计数;
7:6	TRACE_MODE[1:0]	RW	0	跟踪引脚分配。 00: 跟踪引脚使用异步模式; 01: 跟踪引脚使用同步模式, 且 TRACEDATA 长度为 1;

Bit	Name	R/W	Reset Value	Function
				10：跟踪引脚使用同步模式，且 TRACEDATA 长度为 2； 11：跟踪引脚使用同步模式，且 TRACEDATA 长度为 4；
5	TRACE_IOEN	RW	0	跟踪引脚分配使能。 0：不分配跟踪引脚（默认状态）； 1：按照 TRACE_MODE 定义，分配跟踪引脚；
4:3	保留	RES	-	保留
2	DBG_STDBY	RW	0	调试待机模式。 0：(FCLK 关, HCLK 关)，正常待机模式处理； 1：(FCLK 开, HCLK 开)，且系统时钟由 HSI8M 提供。MCU 产生系统复位退出待机模式后与上电复位一样。
1	DBG_STOP	RW	0	调试停止模式。 0：(FCLK 关, HCLK 关)。在停止模式，HCLK 和 FCLK 都会关闭。当从停止模式退出时，时钟配置与复位后相同（系统时钟为 HSI8M）。随后，软件需要重新配置时钟控制器以使能 PLL 和 HSE 等。 1：(FCLK 开, HCLK 开)。当进入停止模式，系统内部时钟源不会关闭，FCLK 和 HCLK 存在。当退出停止模式，如果需要改变时钟控制，软件需要重新配置。
0	DBG_SLEEP	RW	0	调试睡眠模式。 0：(FCLK 开, HCLK 关)。在睡眠模式，FCLK 由原先配置好的系统时钟提供，HCLK 关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出后，软件不需要重新配置时钟。 1：(FCLK 开, HCLK 开)。在睡眠模式，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

### 29.3.3. DBG 寄存器映像

Offset	Register	REV_ID[31:0]																											
0xE004_2000	DBG_IDCODE	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DBG_CR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## 30. 版本历史

版本	日期	更新记录
V1.0	2023.08.30	1. 初版

### 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