

用户手册

User Manual

TKM32F499

---

版本：0.85

## 目录

<b>1. CRC计算单元 .....</b>	<b>22</b>
1.1 CRC简介 .....	22
1.2 CRC主要特征特征 .....	22
1.3 CRC功能介绍 .....	22
1.4 CRC寄存器 .....	23
1.4.1 CRC数据寄存器(CRC_DR) .....	23
1.4.2 CRC独立数据寄存器(CRC_IDR) .....	23
1.4.3 CRC控制寄存(CRC_CTRL) .....	24
<b>2. 电源控制 (PWR) .....</b>	<b>25</b>
2.1 电源 .....	25
2.1.1 独立的A/D转换器供电和参考电压 .....	25
2.1.2 电池备份区域 .....	25
2.1.3 电压调节器 .....	26
2.1.4 上电复位 (POR) 和掉电复位 (PDR) .....	26
2.2 低功耗模式 .....	27
2.2.1 降低系统时钟 .....	27
2.2.2 外部时钟的控制 .....	27
2.2.3 睡眠模式 .....	27
2.2.4 停止模式 .....	28
2.2.5 低功耗模式下的自动唤醒 (AWU) .....	29
2.3 电源控制寄存器 .....	29
2.3.1 电源控制寄存器 (PWR_CR) .....	29
2.3.2 电源控制/状态寄存器 (PWR_CSR) .....	30
<b>3. 备份寄存器 (BKP) .....</b>	<b>31</b>
3.1 BKP简介 .....	31
3.2 BKP特征 .....	31
3.3 BKP功能描述 .....	31
3.3.1 侵入检测 .....	31
3.3.2 RTC校准 .....	31
3.4 BKP寄存器描述 .....	32

3.4.1 RTC时钟校准寄存器 (BKP_RTCCR) .....	32
3.4.2 备份控制寄存器 (BKP_CR) .....	32
3.4.3 备份控制/状态寄存器 (BKP_CSR) .....	33
3.4.4 备份数据寄存器x (BKP_DRx) (x = 1 ... 20) .....	33
<b>4. 复位和时钟控制 (RCC) .....</b>	<b>35</b>
4.1 复位.....	35
4.1.1 系统复位 .....	35
4.1.2 电源复位 .....	35
4.1.3 备份域复位.....	35
4.2 时钟.....	35
4.2.1 HSE时钟 .....	37
4.2.2 HSI时钟 .....	37
4.2.3 PLL .....	38
4.2.4 LSE时钟.....	38
4.2.5 LSI时钟 .....	38
4.2.6 系统时钟 (SYSCLK) 选择.....	38
4.2.7 时钟安全系统 (CSS) .....	38
4.2.8 RTC时钟 .....	39
4.2.9 看门狗时钟 .....	39
4.2.10 时钟输出 .....	39
4.3 RCC寄存器描述 .....	39
4.3.1 时钟控制寄存器 (RCC_CR) .....	39
4.3.2 PLL配置寄存器 (RCC_PLLCFGR) .....	41
4.3.3 时钟配置寄存器 (RCC_CFGR) .....	42
4.3.4 时钟中断寄存器 (RCC_CIR) .....	43
4.3.5 AHB1外设复位寄存器 (RCC_AHB1RSTR) .....	45
4.3.6 AHB2外设复位寄存器 (RCC_AHB2RSTR) .....	46
4.3.7 APB1外设复位寄存器 (RCC_APB1RSTR) .....	47
4.3.8 APB2外设复位寄存器 (RCC_APB2RSTR) .....	49
4.3.9 AHB1外设时钟使能寄存器 (RCC_AHB1ENR) .....	50
4.3.10 AHB2外设时钟使能寄存器 (RCC_AHB2ENR) .....	52
4.3.11 APB1外设时钟使能寄存器 (RCC_APB1ENR) .....	52

4.3.12 APB2外设时钟使能寄存器 (RCC_APB2ENR) .....	54
4.3.13 备份域控制寄存器 (RCC_BDCR) .....	56
4.3.14 控制状态寄存器 (RCC_CSR) .....	57
4.3.15 LCDPLL配置寄存器 (RCC_LCD_PLLCFGR) .....	58
4.3.16 RCC专用时钟配置寄存器 (RCC_DCKCFGR) .....	59
<b>5. 通用功能I/O (GPIO) .....</b>	<b>60</b>
<b>5.1 GPIO功能描述.....</b>	<b>60</b>
5.1.1 通用I/O (GPIO) .....	61
5.1.2 单独的位设置或位清除 .....	61
5.1.3 外部中断/唤醒线.....	61
5.1.4 复用功能 (AF) .....	61
5.1.5 软件重新映射I/O复用功能 .....	61
5.1.6 GPIO锁定机制 .....	61
5.1.7 输入配置 .....	62
5.1.8 输出配置 .....	62
5.1.9 复用功能配置 .....	62
5.1.10 模拟输入配置 .....	62
5.1.11 外设的GPIO配置.....	63
<b>5.2 复用功能I/O和调试配置 .....</b>	<b>64</b>
5.2.1 SWD复用功能 .....	64
<b>5.3 GPIO寄存器描述 .....</b>	<b>65</b>
5.3.1 端口配置低寄存器 (GPIOx_CRL) (x=A..E) .....	65
5.3.2 端口配置高寄存器 (GPIOx_CRH) (x=A..E) .....	65
5.3.3 端口输入数据寄存器 (GPIOx_IDR) (x=A..E) .....	66
5.3.4 端口输出数据寄存器 (GPIOx_ODR) (x=A..E) .....	66
5.3.5 端口设置/清除寄存器 (GPIOx_BSRR) (x=A..D) .....	67
5.3.6 端口位清除寄存器 (GPIOx_BRR) (x=A..E) .....	67
5.3.7 端口配置锁定寄存器 (GPIOx_LCKR) (x=A..E) .....	68
5.3.8 端口复用功能低位寄存器 (GPIOx_AFRL) (x=A..E) .....	69
5.3.9 端口复用功能高位寄存器 (GPIOx_AFRH) (x=A..E) .....	70
5.3.10 端口配置高寄存器 (GPIOE_CRH_EXT) .....	70
5.3.11 端口设置/清除寄存器 (GPIOE_BSRR_EXT) .....	71

5.3.12 端口复用功能高位寄存器 (GPIOE_AFRH_EXT) .....	72
<b>6. 系统配置控制器 .....</b>	<b>73</b>
6.1 SYSCFG寄存器描述 .....	73
6.1.1 SYSCFG配置寄存器 (SYSCFG_CFGR) .....	73
6.1.2 外部中断配置寄存器1 (SYSCFG_EXTICR1) .....	76
6.1.3 外部中断配置寄存器2 (SYSCFG_EXTICR2) .....	76
6.1.4 外部中断配置寄存器3 (SYSCFG_EXTICR3) .....	77
6.1.5 外部中断配置寄存器4 (SYSCFG_EXTICR4) .....	77
<b>7. DMA控制器 (DMA) .....</b>	<b>78</b>
7.1 DMA简介 .....	78
7.2 DMA主要特征 .....	78
7.3 功能描述 .....	78
7.3.1 DMA处理 .....	78
7.3.2 仲裁器 .....	78
7.3.3 DMA通道 .....	79
7.3.4 可编程的数据传输宽度, 对齐方式和数据大小端 .....	79
7.3.5 错误管理 .....	80
7.3.6 中断 .....	81
7.3.7 DMA请求映像 .....	81
7.4 DMA寄存器描述 .....	82
7.4.1 DMA中断状态寄存器 (DMA_ISR) .....	82
7.4.2 DMA中断标志清除寄存器 (DMA_IFCR) .....	83
7.4.3 DMA通道x配置寄存器 (DMA_CCRx) ( $x = 1 \dots 8$ ) .....	83
7.4.4 DMA通道x传输数量寄存器 (DMA_CNDTRx) ( $x = 1 \dots 8$ ) .....	84
7.4.5 DMA通道x外设地址寄存器 (DMA_CPARx) ( $x = 1 \dots 8$ ) .....	85
7.4.6 DMA通道x存储器地址寄存器 (DMA_CMARx) ( $x = 1 \dots 8$ ) .....	85
<b>8. 中断和事件 .....</b>	<b>86</b>
8.1 嵌套向量中断控制器 .....	86
8.1.1 系统嘀嗒 (SysTick) 校准值寄存器 .....	86
8.1.2 中断和异常向量 .....	86
8.2 外部中断/事件控制器 (EXTI) .....	89
8.2.1 主要特征 .....	89

8.2.2 框图 .....	90
8.2.3 唤醒事件管理 .....	90
8.2.4 功能说明 .....	90
8.2.5 外部中断/事件线路映像 .....	91
8.3 EXTI寄存器描述 .....	93
8.3.1 中断屏蔽寄存器 (EXTI_IMR) .....	93
8.3.2 事件屏蔽寄存器 (EXTI_EMR) .....	93
8.3.3 上升沿触发选择寄存器 (EXTI_RTSR) .....	94
8.3.4 下降沿触发选择寄存 (EXTI_FTSR) .....	94
8.3.5 软件中断事件寄存器 (EXTI_SWIER) .....	95
8.3.6 软件中断事件寄存 (EXTI_PR) .....	95
<b>9. 模拟/数字转换 (ADC) .....</b>	<b>96</b>
9.1 ADC介绍 .....	96
9.2 ADC主要特征 .....	96
9.3 ADC功能描述 .....	96
9.3.1 ADC开关控制 .....	97
9.3.2 ADC时钟 .....	97
9.3.3 通道选择 .....	97
9.4 ADC工作模式 .....	98
9.4.1 单次转换模式 .....	98
9.4.2 单周期扫描模式 .....	98
9.4.3 连续扫描模式 .....	99
9.4.4 DMA请求 .....	99
9.4.5 采样频率设置 .....	100
9.5 数据对齐 .....	100
9.6 外部触发转换 .....	100
9.7 窗口比较器模式下AD转换结果监控 .....	100
9.8 触摸屏模式下AD转换结果监控 .....	100
9.9 ADC寄存器描述 .....	101
9.9.1 A/D数据寄存器 (ADC_ADDATA) .....	101
9.9.2 A/D 配置寄存器 (ADC_ADCFG ) .....	101
9.9.3 AD控制寄存器 (ADC_ADCR) .....	102

9.9.4	AD通道选择寄存器(ADC_ADCHS) .....	103
9.9.5	A/D窗口比较寄存器(ADC_ADCMPR) .....	104
9.9.6	A/D状态寄存器(ADC_ADSTA ) .....	105
9.9.7	A/D数据寄存器(ADC_ADDR0~9) .....	105
9.9.8	A/D触摸屏X+数据寄存器(ADC_TPXDR) .....	107
9.9.9	A/D触摸屏Y+数据寄存器(ADC_YPDR) .....	107
9.9.10	A/D触摸屏控制寄存器(ADC_TPCR) .....	108
9.9.11	A/D触摸屏滤波寄存器(ADC_TPFR) .....	108
9.9.12	A/D触摸屏通道选择寄存器(ADC_TPCSR) .....	108
9.10	IF.....	109
9.10.1	Hardware IF.....	109
9.10.2	Software information.....	109
<b>10.</b>	<b>LCD-TFT控制器.....</b>	<b>109</b>
10.1	简介 .....	109
10.2	主要特性 .....	109
10.3	寄存器说明 .....	110
10.4	接口定义 .....	113
10.4.1	引脚列表 .....	113
10.4.2	接口信号说明 .....	113
10.5	应用说明 .....	114
10.5.1	控制器初始化 .....	114
10.5.2	显示层的交替使用 .....	114
10.5.3	从显示缓存读取数据 .....	116
10.5.4	显示缓存Endian mode .....	116
10.5.5	VESA时序 .....	119
<b>11.</b>	<b>高级控制定时器 (TIM1/2) .....</b>	<b>119</b>
11.1	TIM1和TIM2简介 .....	119
11.2	主要特征 .....	119
11.3	功能描述 .....	121
11.3.1	时基单元 .....	121
11.3.2	计数模式 .....	122
11.3.3	重复计数器.....	130

11.3.4	时钟选择 .....	131
11.3.5	捕捉/比较通道 .....	134
11.3.6	输入捕捉模式 .....	136
11.3.7	PWM输入模式 .....	136
11.3.8	强制输出模式 .....	137
11.3.9	输出比较模式 .....	137
11.3.10	PWM模式 .....	139
11.3.11	互补输出和死区插入 .....	141
11.3.12	使用刹车功能 .....	142
11.3.13	在外部事件时清除OCxREF信号 .....	144
11.3.14	产生六步PWM输出 .....	145
11.3.15	单脉冲模式 .....	146
11.3.16	编码器接口模式 .....	148
11.3.17	定时器输入异或功能 .....	149
11.3.18	与霍尔传感器的接口 .....	150
11.3.19	TIMx定时器和外部触发的同步 .....	151
11.3.20	定时器同步 .....	154
11.3.21	调试模式 .....	154
11.4	寄存器描述 .....	154
11.4.1	控制寄存器1 (TIMx_CR1) .....	154
11.4.2	控制寄存器2 (TIMx_CR2) .....	156
11.4.3	从模式控制寄存器 (TIMx_SMCR) .....	157
11.4.4	DMA/中断使能寄存器 (TIMX_DIER) .....	160
11.4.5	状态寄存器 (TIMx_SR) .....	161
11.4.6	事件产生寄存器 (TIMx_EGR) .....	163
11.4.7	捕捉/比较模式寄存器1 (TIMx_CCMR1) .....	164
11.4.8	捕捉/比较模式寄存器2 (TIMx_CCMR2) .....	167
11.4.9	捕捉/比较使能寄存器 (TIMx_CCER) .....	169
11.4.10	计数器 (TIMx_CNT) .....	171
11.4.11	预分频器 (TIMx_PSC) .....	171
11.4.12	自动装载寄存器 (TIMx_ARR) .....	172
11.4.13	重复计数寄存器 (TIMx_RCR) .....	172

11.4.14	捕获/比较寄存器1 (TIMx_CCR1) .....	173
11.4.15	捕获/比较寄存器2 (TIMx_CCR2) .....	173
11.4.16	捕获/比较寄存器3 (TIMx_CCR3) .....	174
11.4.17	捕获/比较寄存器4 (TIMx_CCR4) .....	174
11.4.18	刹车和死区寄存器 (TIMx_BDTR) .....	175
11.4.19	DMA控制寄存器 (TIMx_DCR) .....	176
11.4.20	连续模式的DMA地址 (TIMx_DMAR) .....	177
<b>12.</b>	<b>通用定时器 (TIM3/4) .....</b>	<b>178</b>
12.1	TIMx简介 .....	178
12.2	TIMx主要功能 .....	178
12.3	TIMX功能描述 .....	179
12.3.1	时基单元 .....	179
12.3.2	计数模式 .....	181
12.3.3	时钟选择 .....	188
12.3.4	捕捉/比较通道 .....	191
12.3.5	输入捕捉模式 .....	192
12.3.6	PWM输入模式 .....	193
12.3.7	强制输出模式 .....	194
12.3.8	输出比较模式 .....	194
12.3.9	PWM模式 .....	195
12.3.10	单脉冲模式 .....	197
12.3.11	在外部事件时清除OCxREF信号 .....	199
12.3.12	编码器接口模式 .....	199
12.3.13	定时器输入异或功能 .....	201
12.3.14	定时器和外部触发的同步 .....	202
12.3.15	定时器同步 .....	204
12.3.16	调试模式 .....	209
12.4	TIMx寄存器描述 .....	209
12.4.1	控制寄存器1 (TIMx_CR1) .....	209
12.4.2	控制寄存器2 (TIMx_CR2) .....	211
12.4.3	从模式控制寄存器 (TIMx_SMCR) .....	211
12.4.4	DMA/中断使能寄存器 (TIMx_DIER) .....	213

12.4.5 状态寄存器 (TIMx_SR) .....	215
12.4.6 事件产生寄存器 (TIMx_EGR) .....	216
12.4.7 捕捉/比较模式寄存器1 (TIMx_CCMR1) .....	217
12.4.8 捕捉/比较模式寄存器2 (TIMx_CCMR2) .....	221
12.4.9 捕捉/比较使能寄存器 (TIMx_CCER) .....	222
12.4.10 计数器 (TIMx_CNT) .....	224
12.4.11 预分频器 (TIMx_PSC) .....	224
12.4.12 自动装载寄存器 (TIMx_ARR) .....	224
12.4.13 捕获/比较寄存器1 (TIMx_CCR1) .....	225
12.4.14 捕获/比较寄存器2 (TIMx_CCR2) .....	225
12.4.15 捕获/比较寄存器3 (TIMx_CCR3) .....	226
12.4.16 捕获/比较寄存器4 (TIMx_CCR4) .....	226
12.4.17 DMA控制寄存器 (TIMx_DCR) .....	227
12.4.18 连续模式的DMA地址 (TIMx_DMAR) .....	228
<b>13. 通用定时器 (TIM5/6/7) .....</b>	<b>229</b>
13.1 TIMx简介 .....	229
13.2 TIM5/TIM6/TIM7主要功能 .....	229
13.3 TIM5/TIM6/TIM7功能描述 .....	230
13.3.1 时基单元 .....	230
13.3.2 计数模式 .....	231
13.3.3 时钟选择 .....	234
13.3.4 捕捉/比较通道 .....	236
13.3.5 输入捕捉模式 .....	237
13.3.6 PWM输入模式 .....	238
13.3.7 强制输出模式 .....	238
13.3.8 输出比较模式 .....	239
13.3.9 PWM模式 .....	240
13.3.10 单脉冲模式 .....	241
13.3.11 定时器外部触发的同步 .....	243
13.3.12 定时器同步 .....	244
13.3.13 调试模式 .....	245
13.4 TIMx寄存器描述 .....	245

13.4.1	控制寄存器1 (TIMx_CR1) .....	245
13.4.2	控制寄存器2 (TIMx_CR2) .....	246
13.4.3	从模式控制寄存器 (TIMx_SMCR) .....	247
13.4.4	DMA/中断使能寄存器 (TIMX_DIER) .....	248
13.4.5	状态寄存器 (TIMx_SR) .....	248
13.4.6	事件产生寄存器 (TIMx_EGR) .....	250
13.4.7	捕捉/比较模式寄存器1 (TIMx_CCMR1) .....	251
13.4.8	捕捉/比较使能寄存器 (TIMx_CCER) .....	254
13.4.9	计数器 (TIMx_CNT) .....	255
13.4.10	预分频器 (TIMx_PSC) .....	255
13.4.11	自动装载寄存器 (TIMx_ARR) .....	256
13.4.12	捕获/比较寄存器1 (TIMx_CCR1) .....	256
13.4.13	捕获/比较寄存器2 (TIMx_CCR2) .....	257
<b>14.</b>	<b>基础定时器 (TIM8/9/10) .....</b>	<b>258</b>
14.1	TIM8/9/10简介 .....	258
14.2	TIM8/9/10主要功能 .....	258
14.3	TIM8/9/10功能描述 .....	258
14.3.1	时基单元 .....	258
14.3.2	计数模式 .....	260
14.3.3	时钟源 .....	263
14.3.4	调试模式 .....	263
14.4	TIM8/9/10寄存器描述 .....	264
14.4.1	控制寄存器1 (TIMx_CR1) .....	264
14.4.2	控制寄存器2 (TIMx_CR2) .....	265
14.4.3	DMA/中断使能寄存器 (TIMX_DIER) .....	265
14.4.4	状态寄存器 (TIMx_SR) .....	266
14.4.5	事件产生寄存器 (TIMx_EGR) .....	266
14.4.6	计数器 (TIMx_CNT) .....	267
14.4.7	预分频器 (TIMx_PSC) .....	267
14.4.8	自动装载寄存器 (TIMx_ARR) .....	267
<b>15.</b>	<b>独立看门狗 (IWDG) .....</b>	<b>269</b>
15.1	IWDG简介 .....	269

15.2 IWDG主要性能.....	269
15.3 IWDG功能描述.....	269
15.3.1 硬件看门狗.....	270
15.3.2 寄存器访问保护 .....	270
15.3.3 调试模式 .....	270
15.4 IWDG寄存器描述.....	271
15.4.1 键寄存器 (IWDG_KR) .....	271
15.4.2 预分频寄存器 (IWDG_PR) .....	271
15.4.3 重装载寄存器 (IWDG_RLR) .....	272
15.4.4 状态寄存器(IWDG_SR) .....	272
<b>16. 窗口看门狗 (WWDG) .....</b>	<b>274</b>
16.1 WWDG简介 .....	274
16.2 WWDG主要特征 .....	274
16.3 WWDG功能描述 .....	274
16.4 如何编写看门狗超时程序 .....	275
16.5 调试模式 .....	276
16.6 WWDG寄存器描述.....	276
16.6.1 控制寄存器 (WWDG_CR) .....	276
16.6.2 配置寄存器 (WWDG_CFR) .....	276
16.6.3 状态寄存器 (WWDG_SR) .....	277
<b>17. 实时时钟 (RTC) .....</b>	<b>278</b>
17.1 RTC简介 .....	278
17.2 主要特征 .....	278
17.3 功能描述 .....	278
17.3.1 概述 .....	278
17.3.2 复位过程 .....	279
17.3.3 读RTC寄存器.....	279
17.3.4 配置RTC寄存器 .....	280
17.3.5 RTC标志的设置 .....	280
17.4 RTC寄存器描述 .....	281
17.4.1 RTC控制寄存器高位 (RTC_CRH) .....	281
17.4.2 RTC 控制寄存器低位(RTC_CRL) .....	282

17.4.3	RTC 预分频装载寄存器 (RTC_PRLH/RTC_PRLL) .....	283
17.4.4	RTC 预分频器分频因子寄存器 (RTC_DIVH/RTC_DIVL) .....	284
17.4.5	RTC 计数器寄存器 (RTC_CNTH/RTC_CNTL) .....	285
17.4.6	RTC 闹钟寄存器 (RTC_ALRH/RTC_ALRL) .....	286
<b>18.</b>	<b>I2C接口.....</b>	<b>287</b>
18.1	I2C简介 .....	287
18.2	I2C主要特征 .....	287
18.3	I2C协议 .....	287
18.3.1	起始和停止条件 .....	287
18.3.2	从机寻址协议 .....	287
18.3.3	发送和接收协议 .....	289
18.3.4	起始字节传输协议 .....	290
18.3.5	发送缓冲管理以及起始、停止和重复起始条件产生 .....	291
18.3.6	多个主机仲裁 .....	291
18.3.7	时钟同步 .....	292
18.4	I2C工作模式 .....	293
18.4.1	从模式 .....	294
18.4.2	主模式 .....	295
18.4.3	I2C中止传输 .....	296
18.5	利用DMA通信 (将在未来的版本支持, 本版本不建议使用) .....	297
18.6	I2C中断 .....	297
18.7	I2C寄存器描述 .....	298
18.7.1	I2C控制寄存器 (IC_CON) .....	298
18.7.2	I2C目标地址寄存器 (IC_TAR) .....	300
18.7.3	I2C从机地址寄存器 (IC_SAR) .....	301
18.7.4	I2C高速模式主机码地址寄存器 (IC_HS_MADDR) .....	301
18.7.5	I2C数据命令寄存器 (IC_DATA_CMD) .....	301
18.7.6	标准模式I2C时钟高电平计数寄存器 (IC_SS_SCL_HCNT) .....	302
18.7.7	标准模式I2C时钟低电平计数寄存器 (IC_SS_SCL_LCNT) .....	302
18.7.8	快速模式I2C时钟高电平计数寄存器 (IC_FS_SCL_HCNT) .....	303
18.7.9	快速模式I2C时钟低电平计数寄存器 (IC_FS_SCL_LCNT) .....	303
18.7.10	高速模式I2C时钟高电平计数寄存器 (IC_HS_SCL_HCNT) .....	303

18.7.11	高速模式I2C时钟低电平计数寄存器(IC_HS_SCL_LCNT) .....	304
18.7.12	I2C中断状态寄存器(IC_INTR_STAT) .....	304
18.7.13	I2C中断屏蔽寄存器(IC_INTR_MASK) .....	305
18.7.14	I2C RAW中断寄存器(IC_RAW_INTR_STAT) .....	305
18.7.15	I2C接收阈值(IC_RX_TL) .....	307
18.7.16	I2C发送阈值(IC_TX_TL) .....	307
18.7.17	I2C组合和独立中断清除寄存器(IC_CLR_INTR) .....	307
18.7.18	I2C清除RX_UNDER中断寄存器(IC_CLR_RX_UNDER) .....	308
18.7.19	I2C清除RX_OVER中断寄存器(IC_CLR_RX_OVER) .....	308
18.7.20	I2C清除TX_OVER中断寄存器(IC_CLR_TX_OVER) .....	308
18.7.21	I2C清除RD_REQ中断寄存器(IC_CLR_RD_REQ) .....	309
18.7.22	I2C清除TX_ABRT中断寄存器(IC_CLR_TX_ABRT) .....	309
18.7.23	I2C清除RX_DONE中断寄存器(IC_CLR_RX_DONE) .....	310
18.7.24	I2C清除ACTIVITY中断寄存器(IC_CLR_ACTIVITY) .....	310
18.7.25	I2C清除STOP_DET中断寄存器(IC_CLR_STOP_DET) .....	311
18.7.26	I2C清除START_DET中断寄存器(IC_CLR_START_DET) .....	311
18.7.27	I2C清除GEN_CALL中断寄存器(IC_CLR_GEN_CALL) .....	311
18.7.28	I2C使能寄存器(IC_ENABLE) .....	312
18.7.29	I2C状态寄存器(IC_STATUS) .....	312
18.7.30	I2C 发送缓冲水平寄存器(IC_TXFLR) .....	313
18.7.31	I2C 接收缓冲水平寄存器(IC_RXFLR) .....	313
18.7.32	I2C SDA保持时间寄存器(IC_SDA_HOLD) .....	314
18.7.33	I2C DMA控制寄存器(IC_DMA_CR) .....	314
18.7.34	I2C SDA建立时间寄存器(IC_SDA_SETUP) .....	315
18.7.35	I2C广播呼叫ACK寄存器(IC_ACK_GENERAL_CALL) .....	315
<b>19. I2S .....</b>	<b>316</b>	
19.1	I2S 特性 .....	316
19.2	I2S功能 .....	316
19.2.1	I2S clock generator.....	316
19.2.2	I2S data format .....	316
19.2.3	I2S timing information .....	317
19.3	I2S寄存器描述: .....	318

19.3.1	I2S写数据寄存器 (I2S_WR) .....	318
19.3.2	I2S接收使能寄存器 (I2S_RD) .....	319
19.3.3	I2S状态寄存器 (I2S_CSR) .....	319
19.3.4	I2S全局控制寄存器 (I2S_GCR) .....	320
19.3.5	I2S数据格式寄存器 (I2S_DFR) .....	321
19.3.6	I2S中断状态寄存器 (I2S_ISR) .....	322
19.3.7	I2S中断使能寄存器 (I2S_IER) .....	323
19.3.8	I2S中断清除寄存器 (I2S_ICR) .....	324
19.3.9	I2S分频寄存器 (I2S_PRE) .....	324
<b>20.</b>	<b>串行外设接口 (SPI) .....</b>	<b>326</b>
20.1	SPI简述 .....	326
20.2	主要特征 .....	326
20.3	SPI功能描述 .....	326
20.3.1	概述 .....	326
20.3.2	SPI从模式 .....	330
20.3.3	SPI主模式 .....	331
20.3.4	状态标志 .....	332
20.3.5	波特率设置 .....	332
20.3.6	利用DMA的SPI通信 .....	333
20.4	寄存器堆和存储器映射描述 .....	333
20.4.1	发送数据寄存器 (SPI_TXREGA) .....	333
20.4.2	发送数据寄存器 (SPI_TXREGBL) .....	334
20.4.3	发送数据寄存器 (SPI_TXREGBH) .....	334
20.4.4	接收数据寄存器 (SPI_RXREG) .....	334
20.4.5	当前状态寄存器 (SPI_CSTAT) .....	335
20.4.6	中断状态寄存器 (SPI_INTSTAT) .....	336
20.4.7	中断使能寄存器 (SPI_INTEN) .....	337
20.4.8	中断清除寄存器 (SPI_INTCLR) .....	338
20.4.9	全局控制寄存器 (SPI_GCTL) .....	339
20.4.10	通用控制寄存器 (SPI_CCTL) .....	340
20.4.11	波特率发生器 (SPI_SPBRG) .....	341
20.4.12	接收数据个数寄存器 (SPI_RXDNR) .....	342

20.4.13 从机片选寄存器 (SPI_SCSR) .....	342
20.4.14 数据控制寄存器(SPI_EXTCTL) .....	342
20.4.15 重复发送数据数量控制寄存器(SPI_TX_NUM) .....	343
20.4.16 传输模式寄存器(SPI_TRANSF_MODE) .....	343
<b>21. 串行外设接口 (QSPI) .....</b>	<b>345</b>
21.1 QSPI简述 .....	345
21.2 主要特征 .....	345
21.3 SPI功能描述 .....	345
21.3.1 概述 .....	345
21.3.2 SPI主模式 .....	348
21.3.3 状态标志 .....	349
21.3.4 波特率设置 .....	350
21.3.5 利用DMA的SPI通信 .....	350
21.4 寄存器堆和存储器映射描述 .....	351
21.4.1 发送数据寄存器(QSPI_TXREG) .....	351
21.4.2 接收数据寄存器(QSPI_RXREG) .....	351
21.4.3 当前状态寄存器 (QSPI_CSTAT) .....	352
21.4.4 中断状态寄存器 (QSPI_INTSTAT) .....	352
21.4.5 中断使能寄存器 (QSPI_INTEN) .....	354
21.4.6 中断清除寄存器 (QSPI_INTCLR) .....	355
21.4.7 全局控制寄存器(QSPI_GCTL) .....	356
21.4.8 通用控制寄存器(QSPI_CCTL) .....	357
21.4.9 波特率发生器 (QSPI_SPBRG) .....	358
21.4.10 接收数据个数寄存器 (QSPI_RXDNR) .....	358
21.4.11 从机片选寄存器 (QSPI_SCSR) .....	359
21.4.12 从机片选寄存器 (QSPI_MODE) .....	359
<b>22. 通用异步收发器 (UART) .....</b>	<b>360</b>
22.1 UART简介 .....	360
22.2 UART主要特征 .....	360
22.3 UART功能概述 .....	360
22.3.1 UART 特性描述 .....	361
22.3.2 发送器 .....	362

22.3.3	接收器 .....	363
22.3.4	波特率发生器（BRG） .....	364
22.3.5	采样 .....	365
22.3.6	校验控制 .....	365
22.3.7	硬件流控制 .....	365
22.3.8	利用DMA通信 .....	367
22.4	UART中断请求 .....	367
22.5	UART寄存器描述 .....	368
22.5.1	UART发送数据寄存器 (UART_TDR) .....	368
22.5.2	UART接收数据寄存器 (UART_RDR) .....	368
22.5.3	UART当前状态寄存器 (UART_CSR) .....	369
22.5.4	UART中断状态寄存器 (UART_ISR) .....	369
22.5.5	UART中断使能寄存器 (UART_IER) .....	370
22.5.6	UART中断清除寄存器 (UART_ICR) .....	371
22.5.7	UART全局控制寄存器 (UART_GCR) .....	372
22.5.8	UART通用控制寄存器 (UART_CCR) .....	373
22.5.9	UART波特率寄存器 (UART_BRR) .....	373
<b>23.</b>	<b>安全数字输入/输出接口（SDIO） .....</b>	<b>375</b>
23.1	SDIO主要特性 .....	375
23.2	SDIO 寄存器 .....	375
23.2.1	SDIO mmc_ctrl .....	375
23.2.2	SDIO mmc_io .....	376
23.2.3	SDIO mmc_bytectl .....	377
23.2.4	SDIO mmc_tr_blockcnt .....	378
23.2.5	SDIO mmc_crcctl .....	378
23.2.6	SDIO cmd_crc .....	379
23.2.7	SDIO dat_crcl .....	379
23.2.8	SDIO dat_crch .....	380
23.2.9	SDIO mmc_port .....	380
23.2.10	SDIO mmc_int_mask .....	380
23.2.11	SDIO clr_mmc_int .....	381
23.2.12	SDIO mmc_cardsel .....	382
23.2.13	SDIO mmc_sig .....	383

23.2.14 SDIO mmc_io_mbctl.....	383
23.2.15 SDIO mmc_blockcnt.....	384
23.2.16 SDIO mmc_timeoutcnt.....	385
23.2.17 SDIO cmd_bufx(x = 0..15) .....	385
23.2.18 SDIO buf_ctl.....	386
23.2.19 SDIO data_buf.....	387
<b>24. 控制器局域网（CAN） .....</b>	<b>388</b>
24.1 CAN简介 .....	388
24.2 CAN主要特点 .....	388
24.3 CAN控制器的总体描述.....	388
24.3.1 CAN 2.0B主动内核 .....	389
24.3.2 CAN 方框图 .....	389
24.3.3 接口管理逻辑（IML） .....	389
24.3.4 发送缓冲器（TXB） .....	389
24.3.5 接收缓冲器（RXB, RXFIFO） .....	389
24.3.6 验收滤波器（ACF） .....	390
24.3.7 位流处理器（BSP） .....	390
24.3.8 位时序逻辑（BTL） .....	390
24.3.9 错误管理逻辑（EML） .....	390
24.4 CAN工作模式 .....	390
24.4.1 Basic CAN和PeliCAN模式的区别 .....	390
24.5 CAN功能描述 .....	391
24.5.1 Basic CAN模式 .....	391
24.5.2 Peli CAN模式.....	392
24.5.3 发送处理 .....	394
24.5.4 接收管理 .....	395
24.5.5 标识符过滤.....	395
24.5.6 报文存储 .....	402
24.5.7 出错管理 .....	405
24.5.8 位时间特性.....	409
24.5.9 仲裁丢失 .....	409
24.5.10 CAN中断.....	410
24.6 CAN寄存器描述 .....	411

24.6.1	CAN模式寄存器 (CAN_MOD) .....	411
24.6.2	CAN控制寄存器 (CAN_CR) .....	412
24.6.3	CAN命令寄存器 (CAN_CMR) .....	413
24.6.4	CAN状态寄存器 (CAN_SR) .....	414
24.6.5	CAN中断寄存器 (CAN_IR) .....	415
24.6.6	CAN中断使能寄存器 (CAN_IER) .....	417
24.6.7	CAN验收代码寄存器 .....	418
24.6.8	CAN验收屏蔽寄存器 .....	419
24.6.9	CAN总线定时0 (CAN_BTR0) .....	420
24.6.10	CAN总线定时1 (CAN_BTR1) .....	420
24.6.11	CAN发送识别码寄存器0 (CAN_TXID0) .....	421
24.6.12	CAN发送识别码寄存器1 (CAN_TXID1) .....	421
24.6.13	CAN仲裁丢失捕捉寄存器 (CAN_ALC) .....	422
24.6.14	CAN错误代码捕捉 (CAN_ECC) .....	424
24.6.15	CAN错误报警限制寄存器 (CAN_EWLR) .....	425
24.6.16	CAN RX错误计数寄存器 (CAN_RXERR) .....	426
24.6.17	CAN TX错误计数寄存器 (CAN_TXERR) .....	427
24.6.18	CAN发送帧信息寄存器 (CAN_SFF) .....	428
24.6.19	CAN发送识别码寄存器0 (CAN_TXID0) .....	428
24.6.20	CAN发送识别码寄存器1 (CAN_TXID1) .....	429
24.6.21	CAN发送数据寄存器0 (CAN_TXDATA0) .....	429
24.6.22	CAN发送数据寄存器1 (CAN_TXDATA1) .....	430
24.6.23	CAN时钟分频寄存器 (CAN_CDR) .....	430
<b>25.</b>	<b>USB全速设备接口.....</b>	<b>432</b>
25.1	USB介绍 .....	432
25.2	USB主要特征 .....	432
25.3	USB功能描述 .....	433
25.3.1	USB功能模块描述 .....	433
25.4	编程中需要考虑的问题 .....	433
25.4.1	USB传输概述 .....	434
25.4.2	USB枚举 .....	435
25.4.3	USB传输处理 .....	436

25.4.4	IN令牌包 .....	437
25.4.5	OUT令牌包 .....	438
25.5	USB寄存器描述 .....	438
25.5.1	USB TOP寄存器(USB_TOP) .....	438
25.5.2	USB 中断状态寄存器(USB_INT_STATE) .....	439
25.5.3	USB端点中断状态寄存器(EP_INT_STATE) .....	439
25.5.4	USB端点0中断状态寄存器(EP0_INT_STATE) .....	440
25.5.5	USB 中断使能寄存器(USB_INT_EN) .....	441
25.5.6	USB端点中断使能寄存器(EP_INT_EN) .....	441
25.5.7	USB端点0中断使能寄存器(EP0_INT_EN) .....	442
25.5.8	USB端点X中断状态寄存器(EPX_INT_STATE) (X=1~4) .....	442
25.5.9	USB端点X中断使能寄存器(EPX_INT_EN) (X=1~4) .....	443
25.5.10	USB地址寄存器(USB_ADDR) .....	444
25.5.11	USB端点使能寄存器(EP_EN) .....	444
25.5.12	USB数据翻转控制寄存器(TOG_CTRL1_4) .....	444
25.5.13	USB设置包数据寄存器(SETUPX) (0~7) .....	445
25.5.14	USB传输包大小寄存器(PACKET_SIZE) .....	446
25.5.15	USB 端点X有效数据寄存器(EPX_AVAIL) .....	446
25.5.16	USB 端点X控制寄存器(EPX_CTRL) .....	446
25.5.17	USB 端点X FIFO寄存器(EPX_FIFO) .....	447
25.5.18	USB 端点DMA使能寄存器(EP_DMA) .....	447
25.5.19	USB端点暂停寄存器(EP_HALT) .....	447
25.5.20	USB功耗控制存器(USB_POWER) .....	448
<b>26.</b>	<b>TK80.....</b>	<b>449</b>
26.1	Features.....	449
26.2	接口信号 .....	449
26.3	接口时序 .....	450
26.3.1	写命令 .....	450
26.3.2	写数据 .....	450
26.3.3	读命令 .....	451
26.3.4	读数据 .....	451
26.3.5	I80_busy .....	451

26.4	时序参数 .....	452
26.5	寄存器描述 .....	452
26.5.1	控制寄存器CR .....	452
26.5.2	配置寄存器CFGREG1 .....	453
26.5.3	配置寄存器CFGREG2 .....	453
26.5.4	状态寄存器SR.....	454
26.5.5	命令输入寄存器CMDIR.....	454
26.5.6	数据输入寄存器DINR.....	454
26.5.7	命令输出寄存器CMDOR.....	455
26.5.8	数据输出寄存器DOUTR.....	455
26.5.9	盲读数据输出寄存器BRDR .....	455
26.5.10	配置寄存器CFGREG3 .....	455

## 1. CRC 计算单元

### 1.1 CRC 简介

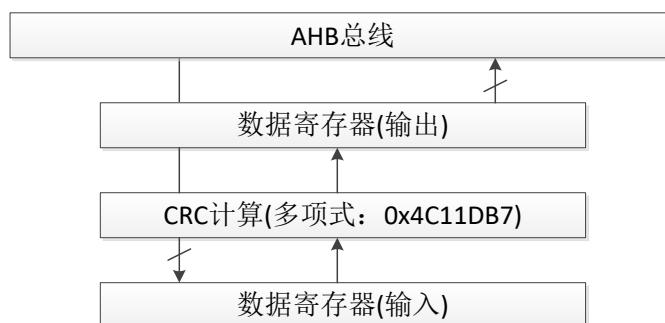
循环冗余校验(CRC)计算单元是根据固定的生成多项式得到任一 32 位全字的 CRC 计算结果。在其他的应用中，CRC 技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准 EN/IEC 60335-1 即提供了一种核实闪存存储器完整性的方法。CRC 计算单元可以在程序运行时计算出软件的标识，之后与在连接时生成的参考标识比较，然后存放在指定的存储器空间。

### 1.2 CRC 主要特征特征

- 使用 CRC-32(以太网)多项式: 0x4C11DB7  
 $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- 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)

下图为 CRC 计算单元框图

图 1. CRC 计算单元框图



### 1.3 CRC 功能介绍

CRC 计算单元含有 1 个 32 位数据寄存器:

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

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

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

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

## 1.4 CRC 寄存器

CRC 计算单元包括了 2 个数据寄存器和一个控制寄存器。

### 1.4.1 CRC 数据寄存器(CRC\_DR)

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

位31: 0	DR: 数据寄存器位 (Data register bits) 写入CRC计算器的新数据时，作为输入寄存器 读取时返回CRC计算结果
--------	--

### 1.4.2 CRC 独立数据寄存器(CRC\_IDR)

地址偏移: 0x04

复位值: 0x0000 0000

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

位31: 8	保留
位7: 0	IDR: 通用8位数据寄存器位 (General-purpose 8-bit data register bits) 可用于临时存放1字节的数据。 寄存器CRC_CTRL的RESET位产生的CRC复位对本寄存器没有影响

注: 此寄存器不参与 CRC 计算, 可以存放任何数据。

### 1.4.3 CRC 控制寄存(CRC\_CTRL)

地址偏移: 0x08

复位值: 0x0000 0000



位31: 1	保留
位0	<b>RESET:</b> 复位CRC计算单元 (CRC reset) 设置数据寄存器为0xFFFF FFFF。 只能对该位写 ‘1’，它由硬件自动清 ‘0’。

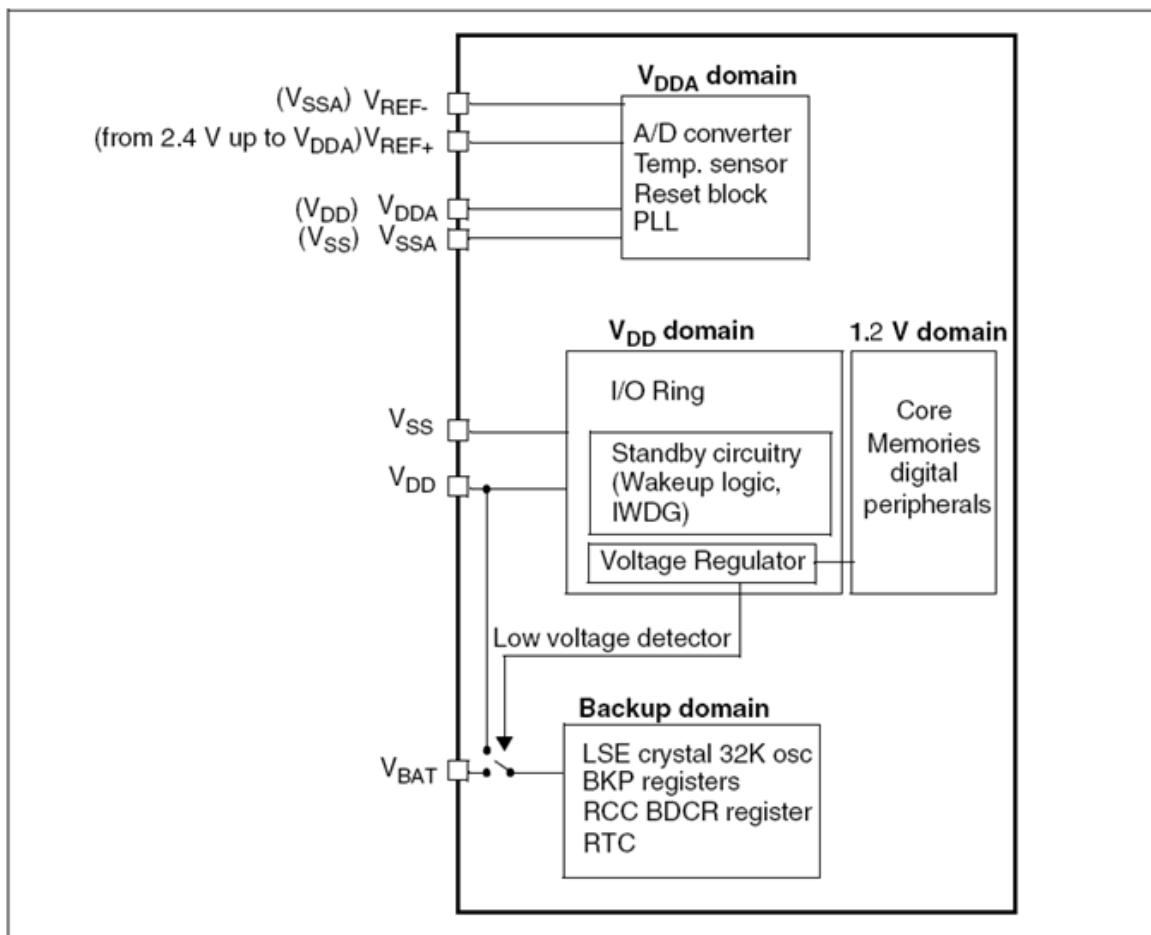
## 2. 电源控制 (PWR)

### 2.1 电源

芯片的工作电压 ( $V_{DD}$ ) 为  $2.5V \sim 3.6V$ 。通过内置的电压调节器提供所需的  $1.2V$  电源。

当主电源  $V_{DD}$  掉电后，通过  $V_{BAT}$  脚为实时时钟 (RTC) 和备份寄存器提供电源。

图 2. 电源框图



注：  $V_{DDA}$  和  $V_{SSA}$  必须分别连到  $V_{DD}$  和  $V_{SS}$ 。

#### 2.1.1 独立的 A/D 转换器供电和参考电压

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

- ADC 的电源引脚为  $V_{DDA}$
- 独立的电源地  $V_{SSA}$

如果有  $V_{REF-}$  引脚（根据封装而定），它必须连接到  $V_{SSA}$ 。

#### 2.1.2 电池备份区域

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

$V_{BAT}$  脚为 RTC、LSE 振荡器和 PB13 至 PB15 端口供电，可以保证当主电源被切断时 RTC 能继续工作。切换到  $V_{BAT}$  供电的开关，由复位模块中的掉电复位功能控制。

**警告:**

在  $V_{DD}$  上升阶段 ( $t_{RSTTEMPO}$ ) 或者探测到 PDR (掉电复位) 之后,  $V_{BAT}$  和  $V_{DD}$  之间的电源开关仍会保持连接在  $V_{BAT}$ 。

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

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

如果在应用中没有外部电池, 建议  $V_{BAT}$  在外部连接到  $V_{DD}$  并连接一个  $100nF$  的陶瓷滤波电容。

当备份区域由  $V_{DD}$  内部模拟开关连到  $V_{DD}$  供电时, 下述功能可用:

- PB14 和 PB15 可以用于 GPIO 或 LSE 引脚
- PB13 可以作为通用 I/O 口、TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出 (参见备份寄存器 (BKP))

*注: 因为模拟开关只能通过少量的电流 (3mA), 在输出模式下使用 PB13 至 PB15 的 I/O 口功能是有限制的: 速度必须限制在 2MHz 以下, 最大负载为 30pF, 而且这些 I/O 口绝对不能当作电流源(如驱动 LED)。*

当后备区域由  $V_{BAT}$  供电时 ( $V_{DD}$  消失后模拟开关连到  $V_{BAT}$ ), 可以使用下述功能:

- PB14 和 PB15 能用于 LSE 引脚, 及 Timer7 的功能复用
- PB13 可以作为 TAMPER 引脚、RTC 闹钟或秒输出 (参见: RTC 时钟校准寄存器 (BKP\_RTCRR))

### 2.1.3 电压调节器

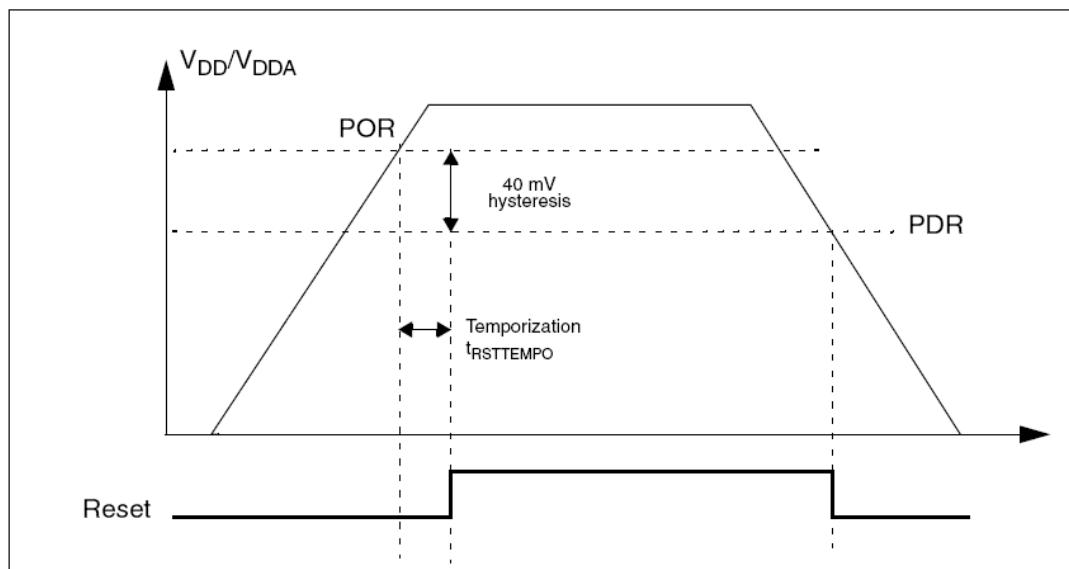
复位后调节器总是使能的, 并提供 1.2V 电压。

### 2.1.4 上电复位 (POR) 和掉电复位 (PDR)

TK499 内部有一个完整的上电复位 (POR) 和掉电复位 (PDR) 电路, 当供电电压达到 1.45V 时系统既能正常工作。

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

图 3. 上电复位和掉电复位的波形图



## 2.2 低功耗模式

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

TK499 有两种低功耗模式：

- 睡眠模式（CPU 停止，所有外设包括 CPU 的外设，如 NVIC、系统时钟（SysTick）等仍在运行）
- 停止模式（所有的时钟都已停止）

此外，在运行模式下，可以通过以下方式中的一种降低功耗：

- 降低系统时钟
- 关闭 APB 和 AHB 总线上未被使用的外设时钟。

表 1. 低功耗模式一览

模式	进入	唤醒	对1.2V区域时钟的影响	对VDD区域时钟的影响	电压调节器
睡眠 (SLEEP-NOW 或 SLEEP-ON-EXIT)	WFI (Wait for Interrupt)	任一中断	CPU时钟关，对其他时钟和ADC时钟无影响	无	开
	WFE (Wait for Event)	唤醒事件			
停机	PDDS位 +SLEEPDEEP位 +WFI或WFE	任一外部中断 或外部事件	所有使用1.2V的区域的时钟都已关闭	PLL、HSI和HSE的振荡器关闭	开

### 2.2.1 降低系统时钟

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

详见：时钟配置寄存器（RCC\_CFRG）

### 2.2.2 外部时钟的控制

在运行模式下，任何时候都可以通过停止为外设和内存提供时钟（HCLK 和 PCLKx）来减少功耗。

为了在睡眠模式下更多地减少功耗，可在执行 WFI 或 WFE 指令前关闭所有外设的时钟。

通过设置 AHB1 外设时钟使能寄存器（RCC\_AHB1ENR）、AHB2 外设时钟使能寄存器（RCC\_AHB2ENR）、APB1 外设时钟使能寄存器（RCC\_APB1ENR）和 APB2 外设时钟使能寄存器（RCC\_APB2ENR）来开关各个外设模块的时钟。

### 2.2.3 睡眠模式

#### 进入睡眠模式

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

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

在睡眠模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

关于如何进入睡眠模式，更多的细节参考表 2 表 3。

## 退出睡眠模式

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

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

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

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。

关于如何退出睡眠模式，更多的细节参考下表。

表 2. SLEEP - NOW 模式

SLEEP-NOW模式	说明
进入	在以下条件下执行WFI（Wait for Interrupt）或WFE（Wait for Event）指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 0 参考CPU系统控制寄存器。
退出	如果执行WFI进入睡眠模式： 中断唤醒，参考中断向量表 如果执行WFE进入睡眠模式： 事件唤醒，参考唤醒事件管理
唤醒延时	无

表 3. SLEEP - ON - EXIT 模式

SLEEP-ON_EXIT模式	说明
进入	在以下条件下执行WFI（Wait for Interrupt）或WFE（Wait for Event）指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 1 参考CPU系统控制寄存器
退出	如果执行WFI进入睡眠模式： 中断唤醒或清除CPU控制寄存器位1 如果执行WFE进入睡眠模式： 唤醒事件，参考唤醒事件管理
唤醒延时	无

## 2.2.4 停止模式

停止模式是在 CPU 的深睡眠模式基础上结合了外设的时钟控制机制，在停止模式下电压调节器仍在工作。此时在 1.2V 供电区域的所有时钟都被停止，PLL、HSI 和 HSE 振荡器的功能被禁止，SRAM 和寄存器内容被保留下来。

在停止模式下，所有的 I/O 引脚都保持它们在运行模式时的状态。

### 进入停止模式

关于如何进入停止模式，详见表 4。

通过对独立的控制位进行编程，可选择以下功能：

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

在停止模式下，如果在进入该模式前 ADC 和 DAC 没有被关闭，那么这些外设仍然消耗电流。通过设置寄存器 ADC\_CR2 的 ADON 位和寄存器 DAC\_CR 的 ENx 位为 0 可关闭这 2 个外设。

### 退出停止模式

关于如何退出停止模式，详见下表。

当一个中断或唤醒事件导致退出停止模式时，HSE 振荡器被选为系统时钟。

表 4. 停止模式

停止模式	说明
进入	<p>在以下条件下执行WFI（Wait for Interrupt）或WFE（Wait for Event）指令：</p> <ul style="list-style-type: none"> <li>- 设置CPU系统控制寄存器中的SLEEPDEEP位</li> <li>- 清除电源控制寄存器（PWR_CTRL）中的PDDS位</li> </ul> <p>注：为了进入停止模式，所有的外部中断的请求位（挂起寄存器（EXTI_PEND））和RTC的闹钟标志都必须被清除，否则停止模式的进入流程将会被跳过，程序继续运行。</p>
退出	<p>在以下条件下执行WFI（Wait for Interrupt）指令：</p> <p>任一外部中断引线被设置为中断模式（相应的外部中断向量在NVIC中必须使能），参见中断向量表。</p> <p>在以下条件下执行WFE（Wait for Event）指令：</p> <p>任一外部中断引线被设置为事件模式，参见唤醒事件管理。</p> <p>NRST引脚上外部复位（复位整个系统）。</p>
唤醒延时	HSE唤醒时间。

### 2.2.5 低功耗模式下的自动唤醒（AWU）

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的微控制器（自动唤醒模式）。RTC 提供一个可编程的时间基数，用于周期性从停止模式下唤醒。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）
- 低功耗内部振荡器（LSI 振荡器）
  - 使用该时钟源，节省了一个 32.768KHz 晶振的成本。但是振荡器将少许增加电源消耗。

为了用 RTC 闹钟事件将系统从停止模式下唤醒，必须进行如下操作：

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

## 2.3 电源控制寄存器

### 2.3.1 电源控制寄存器（PWR\_CR）

地址偏移：0x0

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				PLS[2: 0]		DBP		保留		PVDE	CSBF	CWUF	PDDS	LPDS	
	rW	rW	rW	rW	rW				rW	rc_w1	rW	rW	rW	rW	
位31: 9	保留，始终读为0														

位8	<b>DBP:</b> 取消后备区域的写保护 (domain write protection) 在复位后, RTC和后备寄存器处于被保护状态以防意外写入。设置这位允许写入这些寄存器。 1 = 允许写入RTC和后备寄存器 0 = 禁止写入RTC和后备寄存器 注: 如果RTC的时钟是HSE/128, 该位必须保持为'1'。
位7: 3	保留, 始终读为0
位2	<b>CWUF:</b> 清除唤醒位 (Clear wakeup flag) 始终读出为0 1 = 2个系统时钟周期后清除WUF唤醒位 (写) 0 = 无功效
位1: 0	保留

### 2.3.2 电源控制/状态寄存器 (PWR\_CSR)

地址偏移: 0x04

复位值: 0x0000 0000

与标准的 APB 读相比, 读次寄存器需要额外的 APB 周期

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

rw

位31: 1	保留
位0	<b>WUF:</b> 唤醒标志 (Wakeup flag) 该位由硬件设置, 并只能由POR/PDR (上电/掉电复位) 或设置电源控制寄存器 (PWR_CR) 的 CWUF位清除。 1 = 在WKUP引脚上发生唤醒事件或出现RTC闹钟事件 0 = 没有发生唤醒事件 注: 当WKUP引脚已经是高电平时, 在 (通过设置EWUP位) 使能WKUP引脚时, 会检测到一个额外的事件。

### 3. 备份寄存器 (BKP)

#### 3.1 BKP 简介

备份寄存器是 20 个 32 位的寄存器, 可用来存储 80 个字节的用户应用程序数据, 4K 字节的备份 SRAM 也可以用来存储用户数据, 他们处在备份域里, 当  $V_{DD}$  电源被切断, 他们仍然由  $V_{BAT}$  维持供电。当系统复位或电源复位时, 他们也不会被复位。

此外, BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

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

- 通过设置寄存器 RCC\_APB1ENR 的 PWREN 和 BKOPEN 位来打开电源和后备接口的时钟
- 电源控制寄存器 (PWR\_CR) 的 DBP 位来使能对后备寄存器和 RTC 的访问。

#### 3.2 BKP 特征

- 80 字节数据备份寄存器
- 4K 字节的备份 SRAM
- 用来管理防侵入检测并具有中功能的状态/控制寄存器
- 用来存储 RTC 校验值的校验寄存器。
- 在 PB13 管脚 (当该管脚不用于侵入检测时) 上输出 RTC 校准时钟, RTC 阔钟脉冲或者秒脉冲

#### 3.3 BKP 功能描述

##### 3.3.1 侵入检测

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

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

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

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

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

**注:** 当  $V_{DD}$  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。

##### 3.3.2 RTC 校准

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

通过配置 CAL[6: 0]位, 此时钟可以最多减慢 121ppm。

### 3.4 BKP 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

#### 3.4.1 RTC 时钟校准寄存器（BKP\_RTCCR）

地址偏移: 0x2C

复位值: 0x0000 0000

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

位15: 8	保留，始终读为0。
位9	<b>ASOS:</b> 阵列输出选择 (Alarm or second output selection) 当设置了ASOE位，ASOS位可用于选择在TAMPER引脚上输出的是RTC秒脉冲还是闹钟脉冲信号。 0: 输出RTC闹钟脉冲 1: 输出秒脉冲 注: 该位只能被后备区的复位所清除。
位8	<b>ASOE:</b> 允许输出闹钟或秒脉冲 (Alarm or second output enable) 根据ASOS位的设置，该位允许RTC闹钟或秒脉冲输出到TAMPER引脚上。 输出脉冲的宽度为一个RTC时钟的周期。设置了ASOE位时不能开启TAMPER的功能。 注: 该位只能被后备区的复位所清除。
位7	<b>CCO:</b> 校准时钟输出 (Calibration clock output) 0: 无影响 1: 此位置1可以在侵入检测引脚输出经64分频后的RTC时钟。当CCO位置1时，必须关闭侵入检测功能以避免检测到无用的侵入信号。 注: 当VDD供电断开时，该位被清除。
位6: 0	<b>CAL[6: 0]:</b> 校准值 (Calibration value) 校准值表示在每220个时钟脉冲内将有多少个时钟脉冲被跳过。这可以用来对RTC进行校准，以1000000/(220) ppm的比例减慢时钟。 RTC时钟可以被减慢0 ~ 121ppm。

#### 3.4.2 备份控制寄存器（BKP\_CR）

地址偏移: 0x30

复位值: 0x0000 0000

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

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

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

### 3.4.3 备份控制/状态寄存器 (BKP\_CSR)

地址偏移: 0x34

复位值: 0x0000 0000

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

位15: 10	保留, 始终读为0。
位9	<p><b>TIF:</b> 侵入中断标志 (Tamper interrupt flag)  当检测到有侵入事件且TPIE位为1时, 此位由硬件置‘1’。通过向CTI位写‘1’来清除此标志位 (同时也清除了中断)。如果TPIE位被清除, 则此位也会被清除。  0: 无侵入中断  1: 产生侵入中断  注意: 仅当系统复位才复位该位。</p>
位8	<p><b>TEF:</b> 侵入事件标志 (Tamper event flag)  当检测到侵入事件时此位由硬件置‘1’。通过向CTE位写‘1’可清除此标志位。  0: 无侵入事件  1: 检测到侵入事件  注: 侵入事件会复位所有的BKP_DRx寄存器。只要TEF为1, 所有的BKP_DRx寄存器就一直保持复位状态。当此位被置‘1’时, 若对BKP_DRx进行写操作, 写入的值不会被保存。</p>
位7: 3	保留, 始终读为0。
位2	<p><b>TPIE:</b> 允许侵入TAMPER引脚中断 (TAMPER pin interrupt enable)  0: 禁止侵入检测中断  1: 允许侵入检测中断 (BKP_CR寄存器的TPE位也必须被置‘1’)  注1: 侵入中断无法将系统内核从低功耗模式唤醒。  注2: 仅当系统复位才复位该位。</p>
位1	<p><b>CTI:</b> 清除侵入检测中断 (Clear tamper interrupt)  此位只能写入, 读出值为0。  0: 无效  1: 清除侵入检测中断和TIF侵入检测中断标志</p>
位0	<p><b>CTE:</b> 清除侵入检测事件 (Clear tamper event)  此位只能写入, 读出值为0。  0: 无效  1: 清除TEF侵入检测事件标志 (并复位侵入检测器)</p>

### 3.4.4 备份数据寄存器 x (BKP\_DRx) (x = 1 ... 20)

地址偏移: 0x38 ~ 0x84

复位值: 0x0000 0000

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

位31: 0	<p>BKP[31: 0]: 备份数据 这些位可以被用来写入用户数据。 <b>注意:</b> BKP_DRx寄存器不会被系统复位、电源复位所复位。它们可以由备份域复位来复位或 (如果侵入检测引脚TAMPER功能被开启时)由侵入引脚事件复位。</p>
--------	--

## 4. 复位和时钟控制 (RCC)

### 4.1 复位

支持三种复位形式，分别为系统复位、上电复位和备份区域复位。

#### 4.1.1 系统复位

系统复位将复位除时钟控制寄存器 CSR 中的复位标志和备份区域中的寄存器以外的所有寄存器。

当以下事件中的一件发生时，产生一个系统复位：

1. NRST 管脚上的低电平（外部复位）
2. 窗口看门狗计数终止（WWDG 复位）
3. 独立看门狗计数终止（IWDG 复位）
4. 软件复位（SW 复位）

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

#### 软件复位

通过将 CPU 中断应用和复位控制寄存器中的 SYSRESETREQ 位置 ‘1’，可实现软件复位。

#### 4.1.2 电源复位

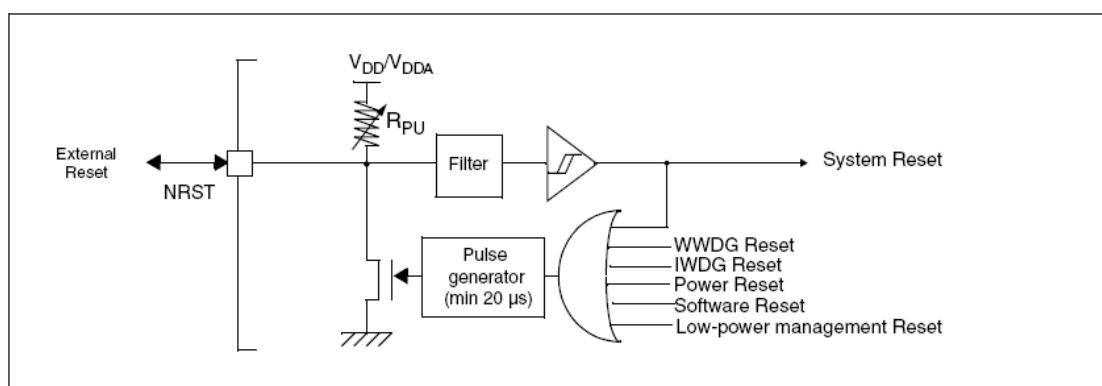
当以下事件中发生时，产生电源复位：

##### 1. 上电/掉电复位（POR/PDR 复位）

电源复位将复位除了备份区域外的所有寄存器。

图中复位源将最终作用于 RESET 管脚，并在复位过程中保持低电平。复位入口矢量被固定在地址 0x0000\_0004。

图 9. 复位电路



#### 4.1.3 备份域复位

备份区域复位可由设置备份区域控制寄存器 RCC\_BDCR 中的 BDRST 位产生，它会将所有 RTC 寄存器和 RCC\_BDCR 寄存器复位为各自的复位值。

在电源 VDD 和 VBAT 都掉电后，其中任何一个再上电，也会复位备份域。

注：在 VBAT 上电后且 BDRST 复位前，备份区域是未复位的状态，需要设置 BDRST 位复位备份区域，同样 BKPSRAM 也需要设置 BKPSRAMRST 复位。

## 4.2 时钟

三种不同的时钟源可被用来驱动系统时钟 (SYSCLK)：

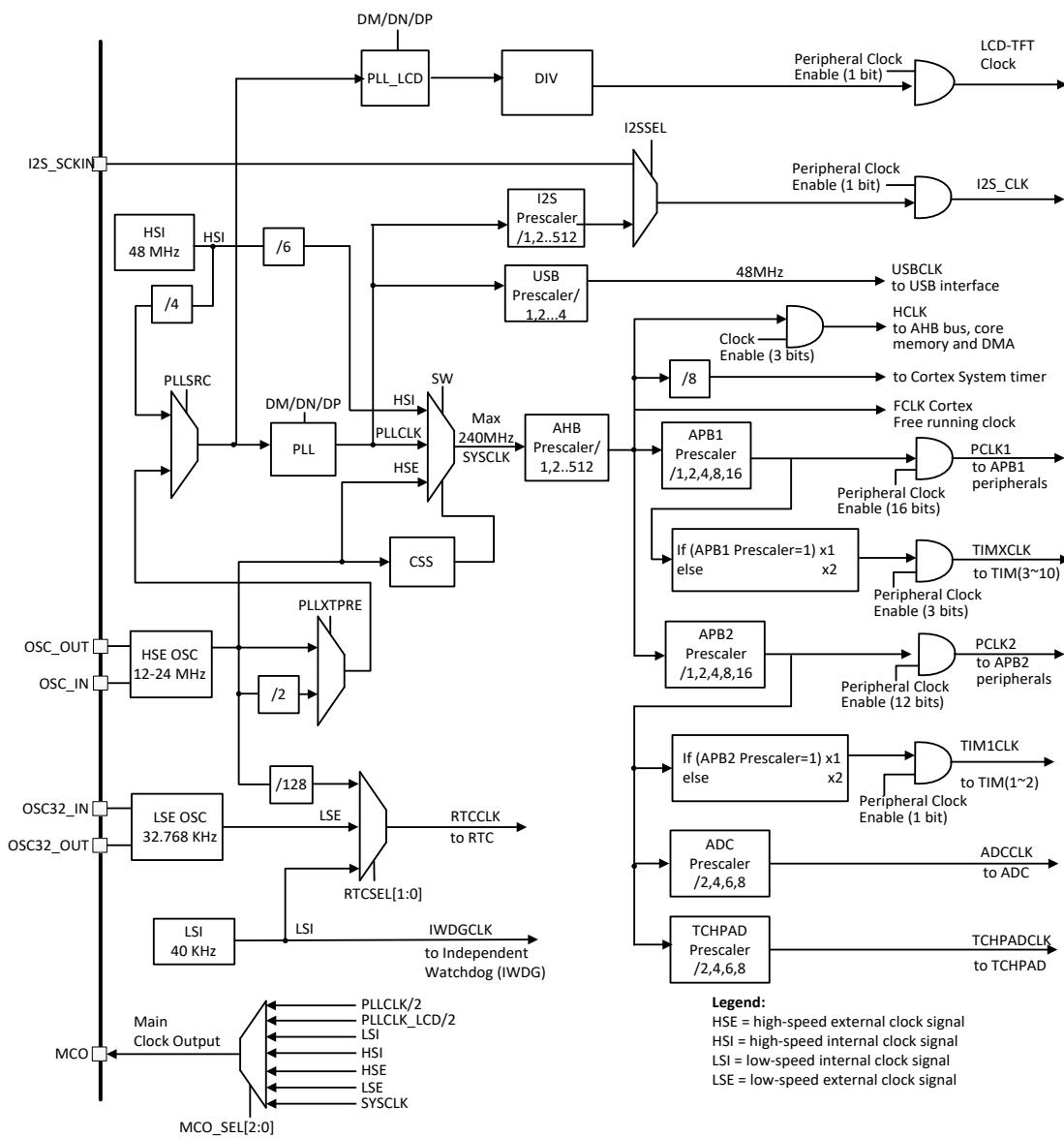
- HSI 振荡器时钟
- HSE 振荡器时钟
- PLL 时钟

这些设备有以下 2 种二级时钟源：

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

当不被使用时，任一个时钟源都可被独立地启动或关闭，由此优化系统功耗。

图 10. 时钟树



用户可通过多个预分频器配置 AHB、高速 APB (APB2) 和低速 APB (APB1) 域的频率。AHB 域的最大频率是 240MHz，APB1 和 APB2 域的最大频率是 120MHz。

RCC 通过 AHB 时钟 8 分频后供给 CPU 系统定时器的 (SysTick) 外部时钟。通过对 SysTick 控制与状态寄存器的设置，可选择上述时钟或 AHB 时钟作为 SysTick 时钟。ADC 时钟由高速 APB2 时钟经 2、4、6 或 8 分频后获得。

定时器时钟频率分配由硬件按以下 2 种情况自动设置：

1. 如果相应的 APB 预分频系数是 1，定时器的时钟频率与所在 APB 总线频率一致。

2. 否则，定时器的时钟频率被设为与其相连的 APB 总线频率的 2 倍。

FCLK 是 CPU 的自由运行时钟。

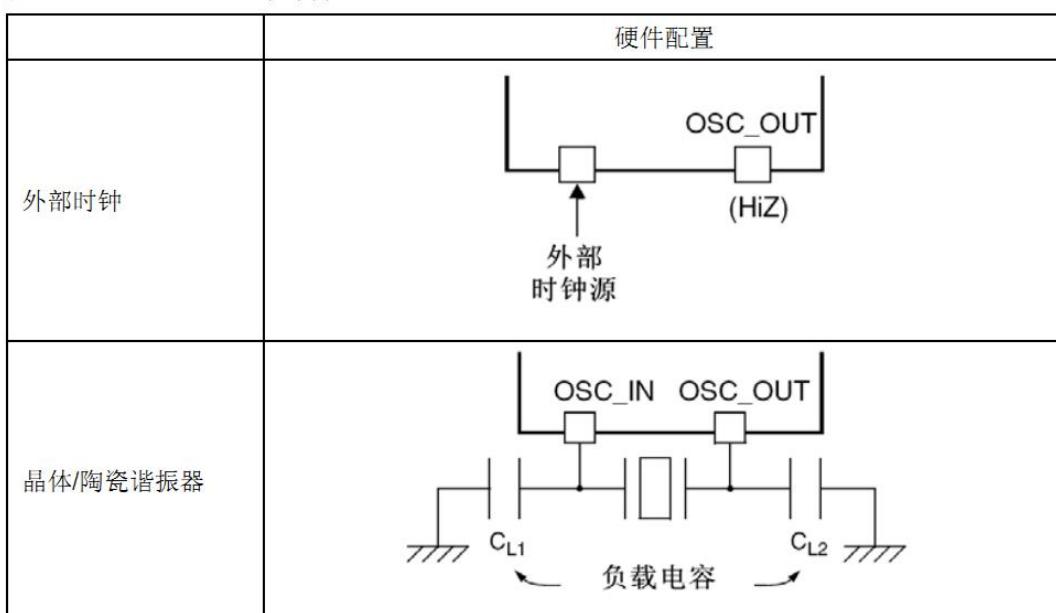
#### 4.2.1 HSE 时钟

高速外部时钟信号（HSE）由以下两种时钟源产生：

- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

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

图 11. HSE/LSE 时钟源



#### 外部时钟源（HSE 旁路）

在这个模式里，必须提供外部时钟。它的频率最高可达 24MHz。用户可通过设置在时钟控制寄存器中的 HSEBYP 和 HSEON 位来选择这一模式。外部时钟信号（50% 占空比的方波、正弦波或三角波）必须连到 OSC\_IN 管脚，同时保证 OSC\_OUT 管脚悬空。

#### 外部晶体/陶瓷谐振器（HSE 晶体）

12 ~ 24MHz 外部振荡器可为系统提供更为精确的主时钟。相关的硬件配置可参考图 11，进一步信息可参考数据手册的电气特性部分。

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

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

#### 4.2.2 HSI 时钟

HSI 时钟信号由内部 48MHz 的振荡器产生，可经过 6 分频后作为系统时钟或 4 分频后作为 PLL 输入。HSI 振荡器能够在不需要任何外部器件的条件下提供系统时钟。它的启动时间比 HSE 晶体振荡器短。

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

如果 HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源。参考 7.2.7 时钟安全系统（CSS）。

#### 4.2.3 PLL

内部 PLL 可以用来倍频 HSI 振荡器的输出时钟或 HSE 晶体输出时钟。参考图 10 和时钟控制寄存器 PLL 的设置（选择 HSI 振荡器或 HSE 振荡器为 PLL 的输入时钟，和选择倍频因子）必须在其被激活前完成。一旦 PLL 被激活，这些参数就不能被改动。

如果 PLL 中断在时钟中断寄存器里被允许，当 PLL 准备就绪时，可产生中断申请。如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48 或 96MHz 时钟，用于提供 48MHz 的 USBCLK 时钟。

#### 4.2.4 LSE 时钟

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

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

##### 外部时钟源（LSE 旁路）

在这个模式里必须提供一个 32.768KHz 频率的外部时钟源。你可以通过设置在备份域控制寄存器（RCC\_BDCR）里的 LSEBYP 和 LSEON 位来选择这个模式。具有 50% 占空比的外部时钟信号（方波、正弦波或三角波）必须连到 OSC32\_IN 管脚，同时保证 OSC32\_OUT 管脚悬空。

#### 4.2.5 LSI 时钟

LSI 振荡器担当一个低功耗时钟源的角色，它可以在停机模式下保持运行，为独立看门狗和自动唤醒单元提供时钟。LSI 时钟频率大约 40KHz（在 26KHz 和 52KHz 之间）。

LSI 振荡器可以通过控制/状态寄存器（RCC\_CSR）里的 LSION 位来启动或关闭。在控制/状态寄存器（RCC\_CSR）里的 LSIRDY 位指示低速内部振荡器是否稳定。在启动阶段，直到这个位被硬件设置为‘1’后，此时钟才被释放。如果在时钟中断寄存器（RCC\_CIR）里被允许，将产生 LSI 中断申请。

#### 4.2.6 系统时钟（SYSCLK）选择

系统复位后，HSI 振荡器被选为系统时钟。当时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。

只有当目标时钟源准备就绪了（经过启动稳定阶段的延迟或 PLL 稳定），从一个时钟源到另一个时钟源的切换才会发生。在被选择时钟源没有就绪时，系统时钟的切换不会发生。直至目标时钟源就绪，才发生切换。

在时钟控制寄存器（RCC\_CR）里的状态位指示哪个时钟已经准备好了，哪个时钟目前被用作系统时钟。

#### 4.2.7 时钟安全系统（CSS）

时钟安全系统可以通过软件被激活。一旦其被激活，时钟监测器将在 HSE 振荡器启动延迟后被使能，并在 HSE 时钟关闭后关闭。

如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟失效事件将被送到高级定时器 TIM1 的刹车输入端，并产生时钟安全中断 CSSI，允许软件完成营救操作。此 CSSI 中断连接到 CPU 的 NMI 中断。

注：一旦 CSS 被激活，并且 HSE 时钟出现故障，CSS 中断就产生，并且 NMI 也自动产生。NMI 将被不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器（RCC\_CIR）里的 CSSC 位来清除 CSS 中断。

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

#### 4.2.8 RTC 时钟

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

LSE 时钟在备份域里，但 HSE 和 LSI 时钟不是。因此：

- 如果 LSE 被选为 RTC 时钟：
  - 只要 VBAT 维持供电，尽管 VDD 供电被切断，RTC 仍继续工作。
  - 如果 LSI 被选为自动唤醒单元 (AWU) 时钟：详见 7.2.5 LSI 时钟。
  - 如果 VDD 供电被切断，AWU 状态不能被保证
  - 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 VDD 供电被切断或 1.2V 域的供电被切断，则 RTC 状态不确定。
  - 必须设置电源控制寄存器的 DBP 位（取消后备区域的写保护）为 ‘1’。

#### 4.2.9 看门狗时钟

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

#### 4.2.10 时钟输出

微控制器允许输出时钟信号到外部 MCO 管脚。

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

- SYSCLK
- HSI
- HSE
- LSI
- LSE
- PLL 的 2 分频时钟
- PLL\_LCDCLK 的 2 分频时钟
- 时钟的选择由时钟配置寄存器 (RCC\_CFGR) 中的 MCO\_SEL[2: 0]位控制。

### 4.3 RCC 寄存器描述

#### 4.3.1 时钟控制寄存器 (RCC\_CR)

地址偏移：0x00

复位值：0x0003 XX03，X 代表未定义

访问：无等待状态，字，半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留	PLL_LCD_RDY	PLL_LCD_ON	保留	PLL_RDY	PLL_ON	保留	保留	保留	保留	保留	保留	CSS_ON	HSE_BYP	HSE_RDY	HSE_ON
	r	rw		r	rw							rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												HSI_RDY	HSI_ON		
												r	rw		

位31: 30	保留
位29	<b>PLL_LCDRDY:</b> PLL_LCD时钟就绪标志(PLL_LCD clock ready flag) PLL锁定后由硬件置‘1’。 0: PLL未锁定 1: PLL锁定
位28	<b>PLL_LCDON:</b> PLL_LCD使能 (PLL_LCD enable) 由软件置‘1’或清零。 当进入停止模式时，该位由硬件清零。 0: PLL_LCD关闭 1: PLL_LCD使能
位27: 26	保留
位25	<b>PLL RDY:</b> PLL时钟就绪标志 (PLL clock ready flag) PLL锁定后由硬件置‘1’。 0: PLL未锁定 1: PLL锁定
位24	<b>PLLON:</b> PLL使能 (PLL enable) 由软件置‘1’或清零。 当进入停止模式时，该位由硬件清零。当PLL时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0: PLL关闭 1: PLL使能
位23: 20	保留，始终读为0
位19	<b>CCSON:</b> 时钟安全系统使能 (Clock security system enable) 由软件置‘1’或清零以使能时钟监测器 0: 时钟监测器关闭 1: 如果HSE振荡器就绪，时钟监测器开启
位18	<b>HSEBYP:</b> 外部高速时钟旁路 (External high-speed clock bypass) 在调试模式下由软件置‘1’或清零来旁路外部晶体振荡器。只有外部振荡器关闭的情况下，才能写入该位 0: 外部振荡器没有旁路 1: 外部外部晶体振荡器被旁路
位17	<b>HSERDY:</b> 外部高速时钟就绪标志 (External high-speed clock ready flag) 由硬件置‘1’来指示HSE时钟已经稳定。 0: 外部时钟没有就绪 1: 外部时钟就绪
位16	<b>HSEON:</b> 外部高速时钟使能 (External high-speed clock enable) 由软件置‘1’或清零。 当进入停止模式时，该位由硬件清零，关闭外部时钟。当HSE时钟被用作或被选择将要作为系统时钟时，该位不能被清零。 0: HSE振荡器关闭 1: HSE振荡器开启
位15: 2	保留
位1	<b>HSIRDY:</b> 内部高速时钟就绪标志 (Internal high-speed clock ready flag) 由硬件置‘1’来指示HSI时钟已经稳定。 0: HSI时钟没有就绪 1: HSI时钟就绪
位0	<b>HSION:</b> 内部高速时钟使能 (Internal high-speed clock enable) 由软件置‘1’或清零。 当从停止模式返回或用作系统时钟的外部时钟发生故障时，该位由硬件置‘1’来启动HSI振荡器。当HSI时钟被直接或间接地用作或被选择将要作为系统时钟时，该位不能被清零。 0: HSI振荡器关闭 1: HSI振荡器开启

### 4.3.2 PLL 配置寄存器 (RCC\_PLLCFGR)

偏移地址: 0x04

复位值: 0x0000 0000

访问: 无等待周期, 按字、半字和字节访问。

此寄存器用于根据公式配置PLL 时钟输出:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留										PLL SRC	保留				
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL BY PASS	PLLIS	PLLDN				PLLDP	PLLDM				rw	rw	rw	rw	rw
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 23	保留
位22	<b>PLLSRC:</b> PLL输入时钟源 (PLL entry clock source) 由软件置‘1’或清‘0’来选择PLL输入时钟源。该位只有在PLL关闭时才可以被写入。 0: HSI时钟4分频作为PLL输入时钟 1: HSE时钟或2分频时钟作为PLL输入时钟
位21: 13	保留
位15	<b>PLLBYPASS:</b> PLL旁路控制(PLL bypass control) 0: 不旁路PLL 1: 旁路PLL
位14: 13	<b>PLLIS:</b> PLL电流控制(PLL current control) 00: DN为1-20时, 配置为00 01: DN为21-40时, 配置为01 10: DN为41-60时, 配置为10 11: DN为61以上时, 配置为11
位12: 6	<b>PLLDN:</b> PLL时钟配置系数(PLL clock configure factor) 由软件置 1和清零, 用于控制 PLL的系数。
位5: 4	<b>PLLDP:</b> PLL时钟配置系数(PLL clock configure factor) 由软件置 1和清零, 用于控制 PLL的系数。 00: P=1 01: P=2 10: P=4 11: P=8
位3: 0	<b>PLLDM:</b> PLL时钟配置系数 (PLL clock configure factor) 由软件置 1 和清零, 用于控制 PLL的系数。 PLL配置公式: $F_{CLKO} = F_{REFIN} * N / (M * P)$ $F_{CLKO}$ 是PLL输出频率, $F_{REFIN}$ 是PLL输入参考时钟频率 $N = PLLDN[6: 0] + 1$ $M = PLLDM[3: 0] + 1$

### 4.3.3 时钟配置寄存器 (RCC\_CFRG)

地址偏移: 0x08

复位值: 0x0000 0005

访问: 无等待状态, 字, 半字和字节访问

只有当访问发生在时钟切换时, 才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2S_PRE					I2S_SEL	MCO_SEL			USBPRE			PLLX_TPREG			
						rw		rw	rw	rw		rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2		PPRE1			保留		HPRE				SWS		SW		
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	r	r	rw	rw	

位31: 24	<b>I2SPRE:</b> I2S预分频 (I2S prescaler) 由软件置'1'或清'0'来控制I2S时钟的预分频系数, 分频系数为 (I2SPRE+1)。 00000000: 1分频 00000001: 2分频 ... 11111110: 255分频 11111111: 256分频
	<b>I2SSEL:</b> I2S时钟源选择 (I2S clock selection) 由软件置'1'或清零。这一位是选择PLLCLK或来自外部的I2S_SCKIN作为其时钟源。 0: PLLCLK作为I2S时钟源 1: 外部的I2S_SCKIN作为I2S时钟源
位23	<b>MCO_SEL:</b> 微控制器时钟输出选择 (Microcontroller clock output select) 由软件置'1'或清零。 001: LSI 时钟输出; 010: LSE 时钟输出; 011: 系统时钟 (SYSCLK) 输出; 100: HSI 时钟输出; 101: HSE 时钟输出; 110: PLLCLK时钟2分频后输出。 111: PLL_LCDCCLK时钟2分频后输出。 注意: - 该时钟输出在启动和切换MCO时钟源时可能会被截断。 - 在系统时钟作为输出至MCO管脚时, 请保证输出时钟频率不超过50MHz (IO口最高频率)
	<b>USBPRE:</b> USB预分频 (USB prescaler) 由软件置'1'或清'0'来产生48MHz的USB时钟。在RCC_APB1ENR寄存器中使能USB时钟之前, 必须保证该位已经有效。 000: PLL时钟直接作为USB时钟 001: PLL时钟2分频作为USB时钟 ... 110: PLL时钟7分频作为USB时钟 111: PLL时钟8分频作为USB时钟
位19: 17	<b>PLLXTPREG:</b> HSE分频器作为PLL输入 (HSE divider for PLL entry) 由软件置'1'或清'0'来分频HSE后作为PLL输入时钟。该位只有在PLL关闭时才可以被写入。 0: HSE不分频 1: HSE2分频

位15: 13	<b>PPRE2:</b> 高速APB预分频 (APB2) (APB high-speed prescaler (APB2)) 由软件置'1'或清'0'来控制高速APB2时钟 (PCLK2) 的预分频系数。 注意：软件必须保证APB1时钟频率不超过120MHz。 0xx: HCLK 不分频 100: HCLK 2分频 101: HCLK 4分频 110: HCLK 8分频 111: HCLK 16分频
位12: 10	<b>PPRE1:</b> 低速APB预分频 (APB1) (APB low-speed prescaler (APB1)) 由软件置'1'或清'0'来控制低速APB1时钟 (PCLK1) 的预分频系数。 注意：软件必须保证APB1时钟频率不超过60MHz。 0xx: HCLK 不分频 100: HCLK 2分频 101: HCLK 4分频 110: HCLK 8分频 111: HCLK 16分频
位9: 8	保留
位7: 4	<b>HPRE:</b> AHB预分频 (AHB Prescaler) 由软件置'1'或清'0'来控制AHB时钟的预分频系数。 0xxx: SYSCLK不分频 1000: SYSCLK 2分频                  1100: SYSCLK 64分频 1001: SYSCLK 4分频                  1101: SYSCLK 128分频 1010: SYSCLK 8分频                  1110: SYSCLK 256分频 1011: SYSCLK 16分频                  1111: SYSCLK 512分频
位3: 2	<b>SWS:</b> 系统时钟切换状态 (System clock switch status) 由硬件置'1'或清'0'来指示哪一个时钟源被作为系统时钟。 00: HSI 6分频作为系统时钟; 01: HSE作为系统时钟; 10: PLL输出作为系统时钟; 11: 不可用。
位1: 0	<b>SW:</b> 系统时钟切换 (System clock switch) 由软件置'1'或清'0'来选择系统时钟源。 在从停止模式中返回时或直接或间接作为系统时钟的HSE出现故障时，由硬件强制选择HSI作为系统时钟（如果时钟安全系统已经启动） 00: HSI 6分频作为系统时钟; 01: HSE作为系统时钟; 10: PLL输出作为系统时钟; 11: 不可用。

#### 4.3.4 时钟中断寄存器 (RCC\_CIR)

偏移地址: 0x0C

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CSSC	PLL LCD RDYC	保留	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
									w	w	w	w	w	w	w
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留	PLL LCD RDYIE	保留	PLL RDY IE	HSE RDY IE	HSI RDY IE	LSE RDY IE	LSI RDY IE	CSSF	PLL LCD RDYF	保留	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw		rw	rw	rw	rw	rw	r	r		r	r	r	r	r

位31：24	保留，始终读为0
位23	<b>CSSC:</b> 清除时钟安全系统中断（Clock security system interrupt clear） 由软件置‘1’来清除CSSF安全系统中断标志位CSSF。 0: 无作用 1: 清除CSSF安全系统中断标志位
位22	<b>PLL_LCDRDYC:</b> 0: 无作用 1: 清除PLL_LCD就绪中断标志位PLL_LCDRDY
位21	保留
位20	<b>PLLRDYC:</b> 清除PLL就绪中断（PLL ready interrupt clear） 由软件置‘1’来清除PLL就绪中断标志位PLLRDYF。 0: 无作用 1: 清除PLL就绪中断标志位PLLRDYF
位19	<b>HSERDYC:</b> 清除HSE就绪中断（HSE ready interrupt clear） 由软件置‘1’来清除HSE就绪中断标志位HSERDYF。 0: 无作用 1: 清除HSE就绪中断标志位HSERDYF
位18	<b>HSIRDYC:</b> 清除HSI就绪中断（HSI ready interrupt clear） 由软件置‘1’来清除HSI就绪中断标志位HSIRDYF。 0: 无作用 1: 清除HSI就绪中断标志位HSIRDYF
位17	<b>LSERDYC:</b> 清除LSE就绪中断（LSE ready interrupt clear） 由软件置‘1’来清除LSE就绪中断标志位LSERDYF。 0: 无作用 1: 清除LSE就绪中断标志位LSERDYF
位16	<b>LSIRDYC:</b> 清除LSI就绪中断（LSI ready interrupt clear） 由软件置‘1’来清除LSI就绪中断标志位LSIRDYF。 0: 无作用 1: 清除LSI就绪中断标志位LSIRDYF
位15	保留，始终读为0
位14	<b>PLL_LCDRDYIE:</b> PLL_LCD就绪中断使能（PLL_LCD ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭PLL_LCD就绪中断。 0: PLL_LCD就绪中断关闭 1: PLL_LCD就绪中断使能
位13	保留
位12	<b>PLLRDYIE:</b> PLL就绪中断使能（PLL ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭PLL就绪中断。 0: PLL就绪中断关闭 1: PLL就绪中断使能
位11	<b>HSERDYIE:</b> HSE就绪中断使能（HSE ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭外部8 ~ 24MHz振荡器就绪中断。 0: HSE就绪中断关闭 1: HSE就绪中断使能
位10	<b>HSIRDYIE:</b> HSI就绪中断使能（HSI ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭内部8MHz振荡器就绪中断。 0: HSI就绪中断关闭 1: HSI就绪中断使能
位9	<b>LSERDYIE:</b> LSE就绪中断使能（LSE ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭外部32.768KHz振荡器就绪中断。 0: LSE就绪中断关闭 1: LSE就绪中断使能
位8	<b>LSIRDYIE:</b> LSI就绪中断使能（LSI ready interrupt enable） 由软件置‘1’或清‘0’来使能或关闭内部40KHz振荡器就绪中断。 0: LSI就绪中断关闭 1: LSI就绪中断使能

位7	<b>CSSF:</b> 时钟安全系统中断标志 (Clock security system interrupt flag) 在外部8 ~ 24MHz振荡器时钟出现故障时，由硬件置‘1’。 由软件通过置‘1’CSSC位来清除。 0: 无HSE时钟失效产生的安全系统中断 1: HSE时钟失效导致了时钟安全系统中断
位6	<b>PLL_LCDRDYF:</b> PLL_LCD就绪中断标志 (PLL_LCD ready interrupt flag) 在PLL_LCD就绪且PLL_LCDRDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’PLL_LCDRDYC位来清除。 0: 无PLL_LCD上锁产生的时钟就绪中断 1: PLL_LCD上锁导致时钟就绪中断
位5	保留
位4	<b>PLLRDYF:</b> PLL就绪中断标志 (PLL ready interrupt flag) 在PLL就绪且PLLRDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’PLLRDYC位来清除。 0: 无PLL上锁产生的时钟就绪中断 1: PLL上锁导致时钟就绪中断
位3	<b>HSERDYF:</b> HSE就绪中断标志 (HSE ready interrupt flag) 在外部低速时钟就绪且HSERDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’HSERDYC位来清除。 0: 无外部8 ~ 24MHz振荡器产生的时钟就绪中断 1: 外部8 ~ 24MHz振荡器导致时钟就绪中断
位2	<b>HSIRDYF:</b> HSI就绪中断标志 (HSI ready interrupt flag) 在内部高速时钟就绪且HSIRDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’HSIRDYC位来清除。 0: 无内部8MHz振荡器产生的时钟就绪中断 1: 内部8MHz振荡器导致时钟就绪中断
位1	<b>LSERDYF:</b> LSE就绪中断标志 (LSE ready interrupt flag) 在外部低速时钟就绪且LSERDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’LSERDYC位来清除。 0: 无外部32.768KHz振荡器产生的时钟就绪中断； 1: 外部32.768KHz振荡器导致时钟就绪中断。
位0	<b>LSIRDYF:</b> LSI就绪中断标志 (LSI ready interrupt flag) 在内部低速时钟就绪且LSIRDYIE位被置‘1’时，由硬件置‘1’。 由软件通过置‘1’LSIRDYC位来清除。 0: 无内部40KHz振荡器产生的时钟就绪中断； 1: 内部40KHz振荡器导致时钟就绪中断。

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

偏移地址: 0x10

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
LCDR ST	保留				DMA2 RST	DMA1 RST	保留									
rw														rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留		CRCR ST	保留				GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST	rw				
rw																

位31	<b>LCDRST:</b> LCD-TFT复位 (LCD-TFT reset) 由软件置‘1’或清‘0’。 0: 无作用 1: LCD-TFT复位
位30:23	保留
位22	<b>DMA2RST:</b> DMA2复位 (DMA2 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: DMA2复位
位21	<b>DMA1RST:</b> DMA1复位 (DMA1 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: DMA1复位
位20: 13	保留
位12	<b>CRCRST:</b> CRC复位 (CRC reset) 由软件置‘1’或清‘0’。 0: 无作用 1: CRC复位
位11:5	保留
位4	<b>GPIOERST:</b> 端口E复位 (IO port E reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 端口E复位
位3	<b>GPIODRST:</b> 端口D复位 (IO port D reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 端口D复位
位2	<b>GPIOCRST:</b> 端口C复位 (IO port C reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 端口C复位
位1	<b>GPIOBRST:</b> 端口B复位 (IO port B reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 端口B复位
位0	<b>GPIOARST:</b> 端口A复位 (IO port A reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 端口A复位

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

偏移地址: 0x14

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TK80	保留														
RST															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															

位31	<b>TK80RST:</b> TK80接口复位 (TK80 interface reset) 由软件置‘1’或清‘0’。 0: 无作用 1: TK80复位												
位30:0	保留												

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

偏移地址: 0x18

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留	I2S1 RST	PWR RST	USB RST	CAN2 RST	CAN1 RST	保留	保留	I2C3 RST	I2C2 RST	I2C1 RST	保留	保留	保留	保留	保留
	rw	rw	rw	rw	rw			rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	保留	WWD GRST	保留	BKPR ST	保留	TIM10 RST	TIM9 RST	TIM8 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST		
		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	

位31	保留
位30	<b>I2S1RST:</b> I2S1接口复位 (I2S1 interface reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位I2S1接口
位29	<b>PWRRST:</b> 电源接口复位 (Power interface reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位电源接口
位28	<b>USBRST:</b> USB复位 (USB reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位USB
位27	<b>CAN2RST:</b> CAN2复位 (CAN2 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位CAN2
位26	<b>CAN1RST:</b> CAN1复位 (CAN1 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位CAN1
位25:24	保留
位23	<b>I2C3RST:</b> I2C3复位 (I2C3 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位I2C3
位22	<b>I2C2RST:</b> I2C2复位 (I2C2 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位I2C2

位21	<b>I2C1RST:</b> I2C1复位 (I2C1 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位I2C1
位20: 12	保留
位11	<b>WWDGRST:</b> 窗口看门狗复位 (Window watchdog reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位窗口看门狗
位10	保留, 始终读为0。
位9	<b>BKPRST:</b> 电源域下备份接口复位 (Digital Domain Backup interface reset) 由软件置‘1’或清‘0’。 0: 无作用; 1: 复位备份接口。
位8	保留
位7	<b>TIM10RST:</b> 定时器10复位 (Timer10 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM10定时器
位6	<b>TIM9RST:</b> 定时器9复位 (Timer9 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM9定时器
位5	<b>TIM8RST:</b> 定时器8复位 (Timer8 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM8定时器
位4	<b>TIM7RST:</b> 定时器7复位 (Timer7 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM7定时器
位3	<b>TIM6RST:</b> 定时器6复位 (Timer6 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM6定时器
位2	<b>TIM5RST:</b> 定时器5复位 (Timer5 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM5定时器
位1	<b>TIM4RST:</b> 定时器4复位 (Timer4 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM4定时器
位0	<b>TIM3RST:</b> 定时器3复位 (Timer3 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TIM3定时器

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

偏移地址: 0x1C

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留				TCH PAD RST	QSPI1 RST	SPI4 RST	SPI3 RST	SPI2 RST	SPI1 RST	保留					
				rw	rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	SYSC FG RST	保留	SDIO2	SDIO1	保留	保留	ADC1 RST	保留	UART 5RST	UART 4RST	UART 3RST	UART 2RST	UART 1RST	TIM2 RST	TIM1 RST
	rw		rw	rw			rw		rw	rw	rw	rw	rw	rw	rw

位31: 26	保留
位25	<b>TCHPADRST:</b> TCHPAD复位 (TCHPAD reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位TCHPAD
位24	<b>QSPI1RST:</b> QSPI复位 (QSPI reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位QSPI
位23	<b>SPI4RST:</b> SPI4复位 (SPI4 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位SPI4
位22	<b>SPI3RST:</b> SPI3复位 (SPI3 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位SPI3
位21	<b>SPI2RST:</b> SPI2复位 (SPI2 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位SPI2
位20	<b>SPI1RST:</b> SPI1复位 (SPI1 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位SPI1
位19: 15	保留
位14	<b>SYSCFGRST:</b> 系统配置控制器复位 (System Configuration Controller reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位System Configuration Controller
位13	保留
位12	<b>SDIO2:</b> SDIO2复位 (SDIO2 reset) 由软件置‘1’或清‘0’。 0: 无作用 1: 复位SDIO2

位11	<b>SDIO1:</b> SDIO1复位 (SDIO1 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位SDIO1
位10: 9	保留
位8	<b>ADC1RST:</b> ADC1复位 (ADC1 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位ADC1
位7	保留
位6	<b>UART5RST:</b> UART5复位 (UART5 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位UART5
位5	<b>UART4RST:</b> UART4复位 (UART4 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位UART4
位4	<b>UART3RST:</b> UART3复位 (UART3 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位UART3
位3	<b>UART2RST:</b> UART2复位 (UART2 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位UART2
位2	<b>UART1RST:</b> UART1复位 (UART1 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位UART1
位1	<b>TIM2RST:</b> 定时器2复位 (Timer2 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位TIM2定时器
位0	<b>TIM1RST:</b> 定时器1复位 (Timer1 reset) 由软件置'1'或清'0'。 0: 无作用 1: 复位TIM1定时器

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

偏移地址: 0x20

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCD EN	保留				DMA2 EN	DMA1 EN	保留								
RW					RW	RW									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	BKP SRAM EN	CRC EN	保留				GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN				
RW	RW						RW	RW	RW	RW	RW				

位31	<b>LCDEN:</b> LCD-TFT时钟使能 (LCD-TFT clock enable) 由软件置‘1’或清‘0’。 0: LCD-TFT时钟关闭 1: LCD-TFT时钟使能
位30:23	保留
位22	<b>DMA2EN:</b> DMA2时钟使能 (DMA2 clock enable) 由软件置‘1’或清‘0’。 0: DMA2时钟关闭 1: DMA2时钟使能
位21	<b>DMA1EN:</b> DMA1时钟使能 (DMA1 clock enable) 由软件置‘1’或清‘0’。 0: DMA1时钟关闭 1: DMA1时钟使能
位20: 14	保留
位13	<b>BKPSRAMEN:</b> BKPSRAM时钟使能 (BKPSRAM clock enable) 由软件置‘1’或清‘0’。 0: BKPSRAM时钟关闭 1: BKPSRAM时钟开启
位12	<b>CRCEN:</b> CRC时钟使能 (CRC clock enable) 由软件置‘1’或清‘0’。 0: CRC时钟关闭 1: CRC时钟开启
位11:5	保留
位4	<b>GPIOEEN:</b> 端口E时钟使能 (IO port E clock enable) 由软件置‘1’或清‘0’。 0: 端口E时钟关闭 1: 端口E时钟开启
位3	<b>GPIODEN:</b> 端口D时钟使能 (IO port D clock enable) 由软件置‘1’或清‘0’。 0: 端口D时钟关闭 1: 端口D时钟开启
位2	<b>GPIOCEN:</b> 端口C时钟使能 (IO port C clock enable) 由软件置‘1’或清‘0’。 0: 端口C时钟关闭 1: 端口C时钟开启
位1	<b>GIOPBEN:</b> 端口B时钟使能 (IO port B clock enable) 由软件置‘1’或清‘0’。 0: 端口B时钟关闭 1: 端口B时钟关闭
位0	<b>GPIOAEN:</b> 端口A时钟使能 (IO port A clock enable) 由软件置‘1’或清‘0’。 0: 端口A时钟关闭 1: 端口A时钟开启

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

偏移地址: 0x24

复位值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TK80 EN															
保留															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															

位31	<b>TK80EN:</b> TK80时钟使能 (TK80 clock enable) 由软件置‘1’或清‘0’。 0: TK80时钟关闭 1: TK80时钟使能														
位30:0	保留														

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

偏移地址: 0x28

时钟使能值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留	I2S1 EN	PWR EN	USB EN	CAN2 EN	CAN1 EN	保留	保留	I2C3 EN	I2C2 EN	I2C1 EN					保留
rw	rw	rw	rw	rw				rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				WWD GEN	保留	BKP EN	保留	TIM10 EN	TIM9 EN	TIM8 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN
rw															

位31	保留														
位30	<b>I2S1EN:</b> I2S1接口时钟使能 (I2S1 interface clock enable) 由软件置‘1’或清‘0’。 0: I2S1时钟关闭 1: I2S1时钟使能														
位29	<b>PWREN:</b> 电源接口时钟使能 (Power interface clock enable) 由软件置‘1’或清‘0’。 0: PWR时钟关闭 1: PWR时钟使能														
位28	<b>USBEN:</b> USB时钟使能 (USB clock enable) 由软件置‘1’或清‘0’。 0: USB时钟关闭 1: USB时钟使能														

位27	<b>CAN2EN:</b> CAN2时钟使能 (CAN2 clock enable) 由软件置‘1’或清‘0’。 0: CAN2时钟关闭 1: CAN2时钟使能
位26	<b>CAN1EN:</b> CAN1时钟使能 (CAN1 clock enable) 由软件置‘1’或清‘0’。 0: CAN1时钟关闭 1: CAN1时钟使能
位25: 24	保留
位23	<b>I2C3EN:</b> I2C3时钟使能 (I2C3 clock enable) 由软件置‘1’或清‘0’。 0: I2C3时钟关闭 1: I2C3时钟使能
位22	<b>I2C2EN:</b> I2C2时钟使能 (I2C2 clock enable) 由软件置‘1’或清‘0’。 0: I2C2时钟关闭 1: I2C2时钟使能
位21	<b>I2C1EN:</b> I2C1时钟使能 (I2C1 clock enable) 由软件置‘1’或清‘0’。 0: I2C1时钟关闭 1: I2C1时钟使能
位20: 12	保留
位11	<b>WWDGEN:</b> 窗口看门狗时钟使能 (Window watchdog clock enable) 由软件置‘1’或清‘0’。 0: 窗口看门狗时钟关闭 1: 窗口看门狗时钟使能
位10	保留, 始终读为0。
位9	<b>BKOPEN:</b> 备份接口时钟使能 (Backup interface clock enable) 由软件置‘1’或清‘0’。 0: 备份接口时钟关闭; 1: 备份接口时钟使能备份接口。
位8	保留
位7	<b>TIM10EN:</b> 定时器10时钟使能 (Timer10 clock enable) 由软件置‘1’或清‘0’。 0: 定时器10时钟关闭 1: 定时器10时钟使能
位6	<b>TIM9EN:</b> 定时器9时钟使能 (Timer9 clock enable) 由软件置‘1’或清‘0’。 0: 定时器9时钟关闭 1: 定时器9时钟使能
位5	<b>TIM8EN:</b> 定时器8时钟使能 (Timer8 clock enable) 由软件置‘1’或清‘0’。 0: 定时器8时钟关闭 1: 定时器8时钟使能
位4	<b>TIM7EN:</b> 定时器7时钟使能 (Timer7 clock enable) 由软件置‘1’或清‘0’。 0: 定时器7时钟关闭 1: 定时器7时钟使能
位3	<b>TIM6EN:</b> 定时器6时钟使能 (Timer6 clock enable) 由软件置‘1’或清‘0’。 0: 定时器6时钟关闭 1: 定时器6时钟使能
位2	<b>TIM5EN:</b> 定时器5时钟使能 (Timer5 clock enable) 由软件置‘1’或清‘0’。 0: 定时器5时钟关闭 1: 定时器5时钟使能

位1	<b>TIM4EN:</b> 定时器4时钟使能 (Timer4 clock enable) 由软件置‘1’或清‘0’。 0: 定时器4时钟关闭 1: 定时器4时钟使能
位0	<b>TIM3EN:</b> 定时器3时钟使能 (Timer3 clock enable) 由软件置‘1’或清‘0’。 0: 定时器3时钟关闭 1: 定时器3时钟使能

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

偏移地址: 0x2C

时钟使能值: 0x0000 0000

访问: 无等待周期, 字, 半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留								TCHP ADEN	QSPI1 EN	SPI4 EN	SPI3 EN	SPI2 EN	SPI1 EN	保留		
								rw	rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留	SYSC FGEN	保留	SDIO2	SDIO1	保留	ADC1 EN	保留	UART 5EN	UART 4EN	UART 3EN	UART 2EN	UART 1EN	TIM2 EN	TIM1 EN		
	rw		rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	

位31: 26	保留
位25	<b>TCHPADEN:</b> TCHPAD时钟使能 (TCHPAD clock enable) 由软件置‘1’或清‘0’。 0: TCHPADI时钟关闭 1: TCHPAD时钟使能
位24	<b>QSPI1EN:</b> QSPI时钟使能 (QSPI clock enable) 由软件置‘1’或清‘0’。 0: QSPI时钟关闭 1: QSPI时钟使能
位23	<b>SPI4EN:</b> SPI4时钟使能 (SPI4 clock enable) 由软件置‘1’或清‘0’。 0: SPI4时钟关闭 1: SPI4时钟使能
位22	<b>SPI3EN:</b> SPI3时钟使能 (SPI3 clock enable) 由软件置‘1’或清‘0’。 0: SPI3时钟关闭 1: SPI3时钟使能
位21	<b>SPI2EN:</b> SPI2时钟使能 (SPI2 clock enable) 由软件置‘1’或清‘0’。 0: SPI2时钟关闭 1: SPI2时钟使能
位20	<b>SPI1EN:</b> SPI1时钟使能 (SPI1 clock enable) 由软件置‘1’或清‘0’。 0: SPI1时钟关闭 1: SPI1时钟使能
位19: 15	保留
位14	<b>SYSCFGEN:</b> 系统配置控制器时钟使能 (System Configuration Controller clock enable) 由软件置‘1’或清‘0’。 0: 系统配置控制器时钟关闭 1: 系统配置控制器时钟使能

位13	保留
位12	<b>SDIO2:</b> SDIO2时钟使能 (SDIO2 clock enable) 由软件置'1'或清'0'。 0: SDIO2时钟关闭 1: SDIO2时钟使能
位11	<b>SDIO1:</b> SDIO1时钟使能 (SDIO1 clock enable) 由软件置'1'或清'0'。 0: SDIO1时钟关闭 1: SDIO1时钟使能
位10: 9	保留
位8	<b>ADC1EN:</b> ADC1时钟使能 (ADC1 clock enable) 由软件置'1'或清'0'。 0: ADC1时钟关闭 1: ADC1时钟使能
位7	保留
位6	<b>UART5EN:</b> UART5时钟使能 (UART5 clock enable) 由软件置'1'或清'0'。 0: UART5时钟关闭 1: UART5时钟使能
位5	<b>UART4EN:</b> UART4时钟使能 (UART4 clock enable) 由软件置'1'或清'0'。 0: UART4时钟关闭 1: UART4时钟使能
位4	<b>UART3EN:</b> UART3时钟使能 (UART3 clock enable) 由软件置'1'或清'0'。 0: UART3时钟关闭 1: UART3时钟使能
位3	<b>UART2EN:</b> UART2时钟使能 (UART2 clock enable) 由软件置'1'或清'0'。 0: UART2时钟关闭 1: UART2时钟使能
位2	<b>UART1EN:</b> UART1时钟使能 (UART1 clock enable) 由软件置'1'或清'0'。 0: UART1时钟关闭 1: UART1时钟使能
位1	<b>TIM2EN:</b> 定时器2时钟使能 (Timer2 clock enable) 由软件置'1'或清'0'。 0: TIM2时钟关闭 1: TIM2时钟使能
位0	<b>TIM1EN:</b> 定时器1时钟使能 (Timer1 clock enable) 由软件置'1'或清'0'。 0: TIM1时钟关闭 1: TIM1时钟使能

### 4.3.13 备份域控制寄存器 (RCC\_BDCR)

偏移地址: 0x30

复位值: 0x0000 0000, 只能由备份域复位有效复位

访问: 0 ~ 3 等待周期, 字, 半字和字节访问

当连续对该寄存器进行访问时, 将插入等待状态。

**注:** 备份域控制寄存器中 (RCC\_BDCR) 的 LSEON、LSEBYP、RTCSEL 和 RTCEN 位处于备份域。因此, 这些位在复位后处于写保护状态, 只有在电源控制寄存器 (PWR\_CR) 中的 DBP 位置“1”后才能对这些位进行改动。这些位只能由备份域复位清除。任何内部或外部复位都不会影响这些位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														BKP SRAM RST	BD RST
rw rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	保留				RTC SEL[1: 0]		保留				LSE BYP	LSE RDY	LSE ON		
	rw				rw	rw					rw	rw	rw		

位31: 18	保留, 始终读为0。
位17	<b>BKPSRAMRST:</b> 备份域SRAM软件复位 (Backup SRAM software reset) 由软件置‘1’或清‘0’。 0: 复位未激活 1: 复位备份SRAM
位16	<b>BDRST:</b> 备份域软件复位 (Backup domain software reset) 由软件置‘1’或清‘0’。 0: 复位未激活 1: 复位整个备份域
位15	<b>RTCEN:</b> RTC时钟使能 (RTC clock enable) 由软件置‘1’或清‘0’。 0: RTC时钟关闭 1: RTC时钟开启
位14: 10	保留, 始终读为0。
位9: 8	<b>RTCSEL[1: 0]:</b> RTC时钟源选择 (RTC clock source selection) 由软件设置来选择RTC时钟源。一旦RTC时钟源被选定, 直到下次后备域被复位, 它不能在被改变。可通过设置BDRST位来清除。 00: 无时钟 01: LSE振荡器作为RTC时钟 10: LSI振荡器作为RTC时钟 11: HSE振荡器在128分频后作为RTC时钟
位7: 3	保留, 始终读为0。
位2	<b>LSEBYP:</b> 外部低速时钟振荡器旁路 (External low-speed oscillator bypass) 在调试模式下由软件置‘1’或清‘0’来旁路LSE。只有在外部32.768KHz振荡器关闭时, 才能写入该位。 0: LSE时钟未被旁路 1: LSE时钟被旁路
位1	<b>LSERDY:</b> 外部低速LSE就绪 (External low-speed oscillator ready) 由硬件置‘1’或清‘0’来指示是否外部32.768KHz振荡器就绪。在LSEON被清零后, 该位需要6个外部低速振荡器的周期才被清零。 0: 外部32.768KHz振荡器未就绪 1: 外部32.768KHz振荡器就绪

位0	<b>LSEON:</b> 外部低速振荡器使能 (External low-speed oscillator enable) 由软件置‘1’或清‘0’ 0: 外部32.768KHz振荡器关闭 1: 外部32.768KHz振荡器开启
----	--

#### 4.3.14 控制状态寄存器 (RCC\_CSR)

偏移地址: 0x34

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

访问: 0 ~ 3 等待周期, 字, 半字和字节访问

当连续对该寄存器进行访问时, 将插入等待状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR	WWDG	IWDG	SFT	POR	PIN	保留	RMVF								保留
RSTF	RSTF	RSTF	RSTF	RSTF	RSTF										
rw	rw	rw	rw	rw	rw		w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						保留								LSI RDY	LSI ON
														r	rw

位31	<b>LPWRRSTF:</b> 低功耗管理复位标志 (Low power reset flag) 在低功耗管理复位发生时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无低功耗管理复位发生 1: 发生低功耗管理复位
位30	<b>WWDGRSTF:</b> 窗口看门狗复位标志 (Window watchdog reset flag) 在窗口看门狗复位发生时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无窗口看门狗复位发生 1: 发生窗口看门狗复位
位29	<b>IWDGRSTF:</b> 独立看门狗复位标志 (Independent watchdog reset flag) 在独立看门狗复位发生在V <sub>DD</sub> 区域时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无独立看门狗复位发生 1: 发生独立看门狗复位
位28	<b>SFRSTF:</b> 软件复位标志 (Software reset flag) 在软件复位发生时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无软件复位发生 1: 发生软件复位
位27	<b>PORRSTF:</b> 上电/掉电复位标志 (POR/PDR reset flag) 在上电/掉电复位发生时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无上电/掉电复位发生 1: 发生上电/掉电复位
位26	<b>PINRSTF:</b> NRST管脚复位标志 (PIN reset flag) 在NRST管脚复位发生时由硬件置‘1’。 由软件通过写RMVF位清除。 0: 无NRST管脚复位发生 1: 发生NRST管脚复位
位25	保留, 读操作返回0。

位24	<b>RMVF:</b> 清除复位标志 (Remove reset flag) 由软件置'1'来清除复位标志。 0: 无作用 1: 清除复位标志
位23: 2	保留, 读操作返回0.
位1	<b>LSIRDY:</b> 内部低速时钟就绪 (Internal low-speed oscillator ready) 由硬件置'1'或清'0'来指示内部40KHz振荡器是否就绪。 在LSION清零后, 3个内部40KHz振荡器的周期后LSIRDY被清零。 0: 内部40KHz振荡器时钟未就绪 1: 内部40KHz振荡器时钟就绪
位0	<b>LSION:</b> 内部低速振荡器使能 (Internal low-speed oscillator enable) 由软件置'1'或清'0'。 0: 内部40KHz振荡器关闭 1: 内部40KHz振荡器开启

#### 4.3.15 LCDPLL 配置寄存器 (RCC\_LCD\_PLLCFGR)

偏移地址: 0x38

复位值: 0x0000 0000

访问: 无等待周期, 按字、半字和字节访问。

此寄存器用于根据公式配置 PLL 时钟输出:

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
PLL LCD BY PASS	PLLLCDIS	LCDPLLDN	LCDPLLDP	LCDPLLDM											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 16	保留
位15	<b>PLLLCDBYPASS:</b> PLLLCD旁路控制(PLLLCD bypass control) 0: 不旁路PLLLCD 1: 旁路PLLLCD
位14: 13	<b>PLLLCDIS:</b> PLL电流控制(PLLLCD current control) 00: DN为1-20时, 配置为00 01: DN为21-40时, 配置为01 10: DN为41-60时, 配置为10 11: DN为61以上时, 配置为11
位12: 6	<b>LCDPLLDN:</b> PLL时钟配置系数(PLL clock configure factor) 由软件置 1 和清零, 用于控制 PLL 的系数。
位5: 4	<b>LCDPLLDP:</b> PLL时钟配置系数(PLL clock configure factor) 由软件置 1 和清零, 用于控制 PLL 的系数。 00: P=1 01: P=2 10: P=4 11: P=8

位3: 0	<b>LCDPLLDM:</b> PLL时钟配置系数 (PLL clock configure factor) 由软件置1和清零，用于控制PLL的系数。 PLL配置公式: $F_{CLKO} = F_{REFIN} * N / (M * P)$ $F_{CLKO}$ 是PLL输出频率, $F_{REFIN}$ 是PLL输入参考时钟频率 $N = LCDPLLDN[6: 0] + 1$ $M = LCDPLLDM[3: 0] + 1$
-------	---

#### 4.3.16 RCC 专用时钟配置寄存器 (RCC\_DCKCFGR)

偏移地址: 0x3C

复位值: 0x0000 0000

访问: 无等待周期, 按字、半字和字节访问。

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

位31: 18	保留
位17: 16	<b>LCD_PLLDIV:</b> PLL_LCDCCLK的分频系数 (division factor for PLL_LCDCCLK) 由软件置1和清零，用于控制PLL_LCDCCLK的分频。该位只有在PLL_LCD禁止时允许写入。 00:PLL_LCDCCLK 2分频 01:PLL_LCDCCLK 4分频 10:PLL_LCDCCLK 6分频 11:PLL_LCDCCLK 8分频
位15: 0	保留

## 5. 通用功能 I/O (GPIO)

### 5.1 GPIO 功能描述

GPIOA-D 端口有两个 32 位配置寄存器 (GPIOx\_CRL, GPIOx\_CRH)，两个 32 位数据寄存器 (GPIOx\_IDR 和 GPIOx\_ODR)，一个 32 位置位/复位寄存器 (GPIOx\_BSRR)，一个 16 位复位寄存器 (GPIOx\_BRR)、一个 32 位锁定寄存器 (GPIOx\_LCKR) 和两个复用功能选择寄存器 (GPIOx\_AFRH 和 GPIOx\_AFRL)。

GPIOE 端口有三个 32 位配置寄存器 (GPIOE\_CRL, GPIOE\_CRH, GPIOE\_CRH\_EXT)，两个 32 位数据寄存器 (GPIOE\_IDR 和 GPIOE\_ODR)，二个 32 位置位/复位寄存器 (GPIOE\_BSRR 和 GPIOE\_BSRR\_EXT)，一个 24 位复位寄存器 (GPIOE\_BRR)、一个 32 位锁定寄存器 (GPIOE\_LCKR) 和三个复用功能选择寄存器 (GPIOE\_AFRH, GPIOE\_AFRL 和 GPIOE\_AFRH\_EXT)。

表格：

	32位配置 寄存器	32位数据 寄存器	32位置位/复位 寄存器	16位复位 寄存器	32位锁定 寄存器	复用功能选择 寄存器
GPIOA-D	2个	2个	1个	1个	1个	2个
GPIOE	3个	2个	2个	1个	1个	3个

GPIO 端口的每个位可以由软件分别配置成多种模式。

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能（仅 I2C 复用引脚有此配置）

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

表 5. 端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR寄存器
通用输出		0	0	01		0或1
复用功能输出	推挽 (Push-Pull)	1	0	11		不使用
	开漏 (Open-Drain)		1	见表6		不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表 6. 输出驱动能力

MODE[1: 0]	驱动能力
01	8mA
10	12mA
11	16mA

### 5.1.1 通用 I/O (GPIO)

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

复位后，SWD 引脚被置于输入上拉或下拉模式：

- PA14: SWC 置于下拉模式
- PA15: SWD 置于上拉模式

PB14、PB15 原功能是 RTC 的晶振引脚(模拟功能)，当弃用 RTC 时，只能用数字功能，即只有上拉，下拉输入及推挽输出三种模式

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

输入数据寄存器（GPIOx\_IDR）在每个 AHB1 时钟周期捕捉 I/O 引脚上的数据。

所有 GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，他们可以被激活也可以被断开。

### 5.1.2 单独的位设置或位清除

当 GPIOx\_ODR 的个别位编程时，软件不需要禁止中断：在单次 AHB1 写操作里，可以只更改一个或多个位。

这是通过对“置位/复位寄存器”（GPIOx\_BSRR，复位是 GPIOx\_BRR）中想要更改的位写“1”来实现的，没被选择的位将不被更改。

### 5.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，参考 7.2 节：外部中断/事件控制器（EXTI）

### 5.1.4 复用功能(AF)

使用默认复用功能前必须对端口位配置寄存器编程。

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

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

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

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

### 5.1.5 软件重新映射 I/O 复用功能

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

### 5.1.6 GPIO 锁定机制

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

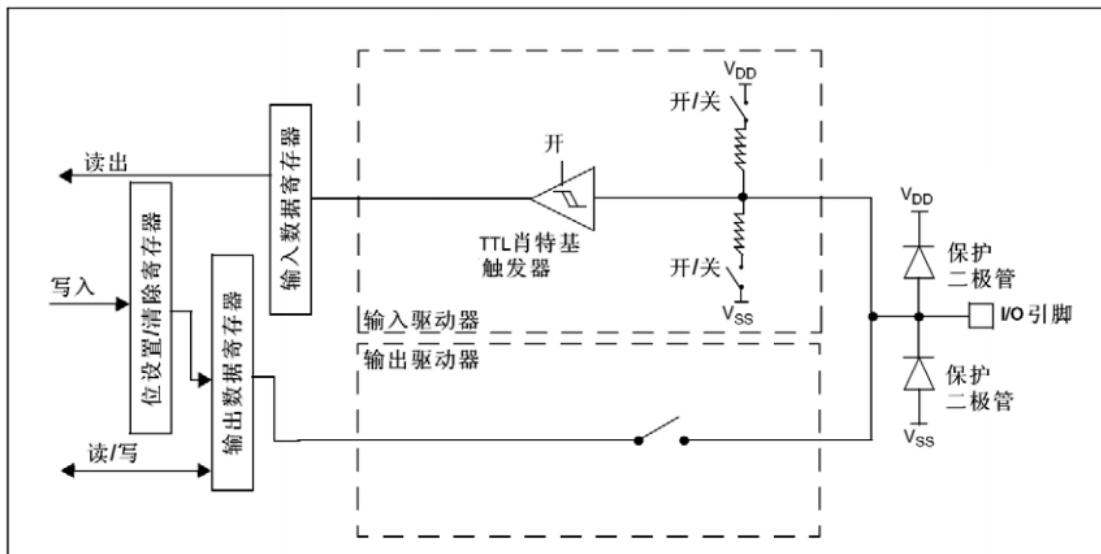
### 5.1.7 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入被激活
- 根据输入配置（上拉，下拉或浮动）的不同，弱上拉和下拉的电阻被连接
- 出现在 I/O 脚上的数据在每个 AHB1 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

下图给出了 I/O 端口位的输入配置

图 4. 输入浮空/上拉/下拉配置



### 5.1.8 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被激活
- 推挽模式：输出寄存器上的“0”激活 N-MOS，而输出寄存器上的“1”将激活 P-MOS。
- 施密特输入被激活
- 弱上拉和下拉电阻被禁止
- 出现在 I/O 脚上的数据在每个 AHB1 时钟被采样到输入数据寄存器
- 在推挽模式时，可对输出数据寄存器的读访问得到最后一次写的值。

### 5.1.9 复用功能配置

当 I/O 端口被配置为复用功能时：

- 在推挽式配置中，输出缓冲器被打开
- 内置外设的信号驱动输出缓冲器（复用功能输出）
- 施密特触发输入被激活
- 弱上拉和下拉电阻被禁止
- 在每个 AHB1 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值

### 5.1.10 模拟输入配置

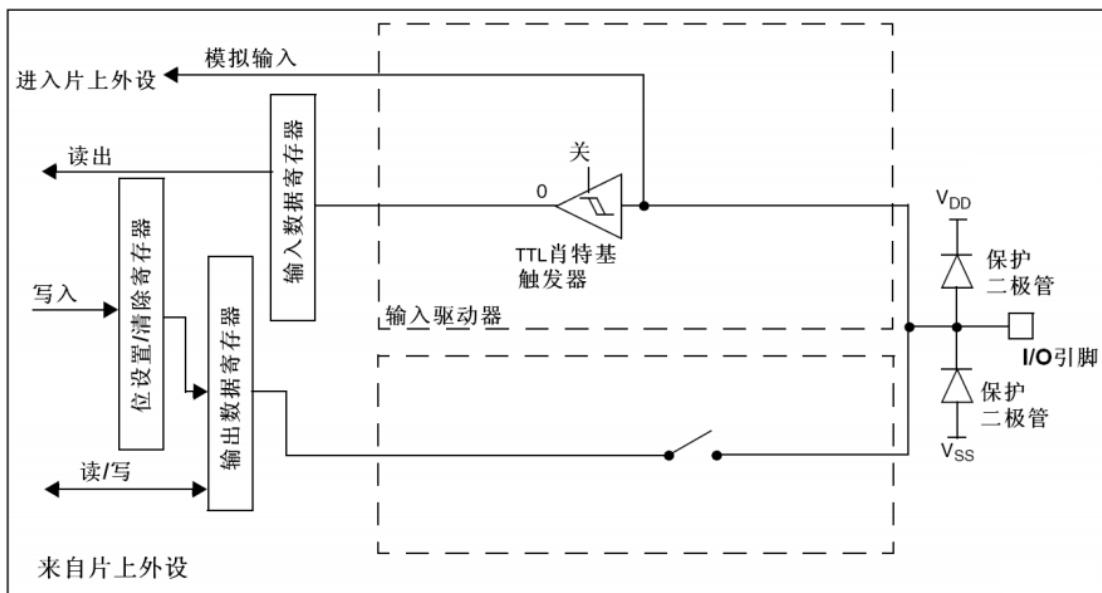
当 I/O 端口被配置成模拟输入配置时：

- 输出缓冲器禁止
- 禁止施密特触发输入，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强制为“0”
- 弱上拉和下拉电阻被禁止

- 读取输入数据寄存器时数值为“0”

下图给出了 I/O 端口位的高阻抗输入配置：

图 5. 高阻抗的模拟输入配置



### 5.1.11 外设的 GPIO 配置

下列表格列出了各个外设的引脚配置。

表 7. 高级定时器 TIMx

TIM1引脚	配置	GPIO配置
TIMx_CHx	输入捕获通道x	浮空输入
	输出比较通道x	推挽复用输出
TIMx_CHxN	互补输出通道x	推挽复用输出
TIMx_BKIN	刹车输入	浮空输入
TIMx_ETR	外部触发时钟输入	浮空输入

表 8. 通用定时器 TIMx

TIMx引脚	配置	GPIO配置
TIMx_CHx	输入捕获通道x	浮空输入
	输出比较通道x	推挽复用输出
TIMx_ETR	外部触发时钟输入	浮空输入

表 9. UART

UART 引脚	配置	GPIO配置
UART_TX	全双工模式	推挽复用输出
	半双工同步模块	推挽复用输出
UART_RX	全双工模式	浮空输入或带上拉输入
	半双工同步模式	未用，可作为通用I/O
UART_RTS	硬件流量控制	推挽复用输出

UART_CTS	硬件流量控制	浮空输入或带上拉输入
----------	--------	------------

表 10. SPI

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

表 11. I2C

I2C引脚	配置	GPIO配置
I2C_SCL	I2C时钟	开漏复用输出
I2C_SDA	I2C数据	开漏复用输出

表 12. ADC

ADC引脚	GPIO配置
ADC	模拟输入

表 13. 其他 I/O 引脚

引脚	配置	GPIO配置
MCO	时钟输出	推挽复用输出
EXTI输入线	外部中断输入	浮空输入或带上拉输入或下拉输入

## 5.2 复用功能 I/O 和调试配置

为了优化外设数目，可以把一些复用功能重新映射到其他引脚上。设置复用功能寄存器（AFR）实现引脚的重新映射。这时，复用功能不再映射到它们的原始分配上。

### 5.2.1 SWD 复用功能

调试接口信号被映射到 GPIO 端口上，如下表所示

表 14. 调试接口信号

复用功能	GPIO端口
------	--------

SWD	PA15
SWC	PA14

为了在调试期间可以使用更多 GPIOs，通过设置复用功能寄存器，可以改变上述重映射配置。

### 5.3 GPIO 寄存器描述

#### 5.3.1 端口配置低寄存器（GPIOx\_CRL）（x=A..E）

偏移地址: 0x00

复位值: 0x4444 4444

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

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

#### 5.3.2 端口配置高寄存器（GPIOx\_CRH）（x=A..E）

偏移地址: 0x04

复位值: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1: 0]	MODE15 [1: 0]	CNF14[1: 0]	MODE14 [1: 0]	CNF13[1: 0]	MODE13 [1: 0]	CNF12[1: 0]	MODE12 [1: 0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1: 0]	MODE11 [1: 0]	CNF10[1: 0]	MODE10 [1: 0]	CNF9[1: 0]	MODE9[1: 0]	CNF8[1: 0]	MODE8[1: 0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

偏移地址: 0x08

复位值: 0x0000 xxxx

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

位31: 24	保留, 始终读为0
位23: 16	<b>IDRy[23: 16]:</b> 端口E输入数据 (y=16..23) (Port input data)
位15: 0	<b>IDRy[15: 0]:</b> 端口输入数据 (y=0..15) (Port input data)

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

偏移地址: 0x0C

复位值: 0x0000\_0000

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

位31: 16	保留, 始终读为0
位23: 16	<b>ODRy[23: 16]:</b> 端口E输出数据 (y=16..23) (Port output data) 注: 对GPIOx_BSRR(x=A...D), 可以分别地对各个ODR位进行独立的设置/清除; 对GPIOx_BSRR(x=E)及GPIOE_BSRR_EXT, 可以分别地对GPIOE的各个ODR位进行独立的设置/清除。
位15: 0	<b>ODRy[15: 0]:</b> 端口输出数据 (y=0..15) (Port output data) 注: 对GPIOx_BSRR (x=A...D), 可以分别地对各个ODR位进行独立的设置/清除。

### 5.3.5 端口设置/清除寄存器 (GPIOx\_BSRR) (x=A..D)

偏移地址: 0x10

复位值: 0x0000\_0000

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

位31: 16	<b>BRy:</b> 清除端口x的位y (y=0...15) (Port x Reset bit y) 这些位只能写入并只能以字 (16位) 的形式操作 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0
位15: 0	<b>BSy:</b> 设置端口x的位y (y=0..15) (Port x Set bit y) 这些位只能写入并只能以字 (16位) 的形式操作。 0: 对对应的ODRy位不产生影响 1: 设置对应的ODRy位为1

### 5.3.6 端口位清除寄存器 (GPIOx\_BRR) (x=A..D)

偏移地址: 0x14

复位值: 0x0000\_0000

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

位31: 24	保留
位23: 16	<b>BRy:</b> 清除端口E的位y (y=16..23) (Port x Reset bit y) 0: 对对应的ODREy位不产生影响 1: 清除对应的ODREy位为0
位15: 0	<b>BRy:</b> 清除端口x的位y (y=0..15) (Port x Reset bit y) 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0

### 5.3.7 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位 24 (LCKK) 时, 该寄存器用来锁定端口位的配置。位[23: 0]用于锁定 GPIO 端口的配置。在规定的写入操作期间, 不能改变 LCKP[15: 0]。当对响应的端口位执行了 LOCK 序列后, 在下次系统复位之前将不能再更改端口位的配置。

每个锁定位锁定控制寄存器 (CRL, CRH) 中相应的 4 个位。

**注:** 因为 **GPIOE** 与 **GPIOA-D** 的引脚数不同, 扩展了 8 位, 在锁定寄存器上 **LCKK** 位有区别, 所以分开描述。

**GPIOA-D** 寄存器描述:

地址偏移: 0x18

复位值: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															LCKK
rw	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
LCK[15: 0]															rw
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

**GPIOE** 寄存器描述:

**注:** **GPIOE** 是 24 个端口。

地址偏移: 0x1C

复位值: 0x0000\_0000

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

位31: 17	保留
位24	<p><b>LCKK:</b> 锁键 (Lock key) 给位可随时读出, 它只可通过锁键写入序列修改。 0: 端口配置锁键位激活 1: 端口配置锁键位被激活, 下次系统复位前GPIOx_LCKR寄存器被锁住 锁键的写入序列: 写1-&gt;写0-&gt;写1-&gt;读0-&gt;读1 最后一个读可省略, 但可以用来确认锁键已被激活。 注: 在操作锁键的写入序列时, 不能改变LCK[15: 0]的值。操作锁键写入序列中的任何错误将不能激活锁键</p>
位23: 16	<p><b>LCKy:</b> 端口E的锁位y (y=16...23) (Port E Lock bit y) 这些位可读可写但只能在LCKK位为0时写入。 0: 不锁定端口的配置 1: 锁定端口的配置</p>
位15: 0	<p><b>LCKy:</b> 端口x的锁位y (y=0...15) (Port x Lock bit y) 这些位可读可写但只能在LCKK位为0时写入。 0: 不锁定端口的配置 1: 锁定端口的配置</p>

### 5.3.8 端口复用功能低位寄存器 (GPIOx\_AFRL) (x=A..E)

偏移地址: 0x20

复位值: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw												

位31: 0	<p><b>AFRy:</b> 端口x的位y (y=0...7) 的复用功能选择 这些位能软件写入配置IO复用功能 0000: GPIO_AF_MCO_SW 0001: GPIO_AF_TIM_1_2 0010: GPIO_AF_TIM_34567 0011: GPIO_AF_I2S 0100: GPIO_AF_I2C 0101: GPIO_AF_SPI 0110: GPIO_AF_QSPI 0111: GPIO_AF_UART_2345 1000: GPIO_AF_UART_1 1001: GPIO_AF_CAN 1010: GPIO_AF_USB 1011: AF11 1100: GPIO_AF_TK80_SDIO 1101: GPIO_AF_Touchpad 1110: GPIO_AF_LTDC 1111: AF15</p>
--------	--

### 5.3.9 端口复用功能高位寄存器 (GPIOx\_AFRH) (x=A..E)

偏移地址: 0x24

复位值: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw												

位31: 0	<b>AFRy:</b> 端口x的位y (y=8...15) 的复用功能选择 这些位能软件写入配置IO复用功能														
	0000: GPIO_AF_MCO_SW														
	0001: GPIO_AF_TIM_1_2														
	0010: GPIO_AF_TIM_34567														
	0011: GPIO_AF_I2S														
	0100: GPIO_AF_I2C														
	0101: GPIO_AF_SPI														
	0110: GPIO_AF_QSPI														
	0111: GPIO_AF_UART_2345														
	1000: GPIO_AF_UART_1														
	1001: GPIO_AF_CAN														
	1010: GPIO_AF_USB														
	1011: AF11														
	1100: GPIO_AF_TK80_SDIO														
	1101: GPIO_AF_Touchpad														
	1110: GPIO_AF_LTDC														
	1111: AF15														

### 5.3.10 端口配置高寄存器 (GPIOE\_CRH\_EXT)

偏移地址: 0x28

复位值: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF23[1: 0]		MODE23 [1: 0]		CNF22[1: 0]		MODE22 [1: 0]		CNF21[1: 0]		MODE21 [1: 0]		CNF20[1: 0]		MODE20 [1: 0]	
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF19[1: 0]		MODE19 [1: 0]		CNF18[1: 0]		MODE18 [1: 0]		CNF17[1: 0]		MODE17 [1: 0]		CNF16[1: 0]		MODE16 [1: 0]	
rw	rw	rw	rw												

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

### 5.3.11 端口设置/清除寄存器 (GPIOE\_BSRR\_EXT)

偏移地址: 0x2C

复位值: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR23	BR22	BR21	BR20	BR19	BR18	BR17	BR16	BS23	BS22	BS21	BS20	BS19	BS18	BS17	BS16
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

位15: 8  位7: 0	<b>BRy:</b> 清除端口E的位y (y=16...23) (Port x Reset bit y) 这些位只能写入并只能以字 (16位) 的形式操作 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0													
	<b>BSy:</b> 设置端口E的位y (y=16..23) (Port x Set bit y)	这些位只能写入并只能以字 (16位) 的形式操作。 0: 对对应的ODRy位不产生影响 1: 设置对应的ODRy位为1												

### 5.3.12 端口复用功能高位寄存器 (GPIOE\_AFRH\_EXT)

偏移地址: 0x30

复位值: 0x0000\_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR23[3:0]				AFR22[3:0]				AFR21[3:0]				AFR20[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR19[3:0]				AFR18[3:0]				AFR17[3:0]				AFR16[3:0]			
rw	rw	rw	rw												

位31: 0	<b>AFRy:</b> 端口E的位y (y=16...23) 的复用功能选择 这些位能软件写入配置IO复用功能
	0000: GPIO_AF_MCO_SW
	0001: GPIO_AF_TIM_1_2
	0010: GPIO_AF_TIM_34567
	0011: GPIO_AF_I2S
	0100: GPIO_AF_I2C
	0101: GPIO_AF_SPI
	0110: GPIO_AF_QSPI
	0111: GPIO_AF_UART_2345
	1000: GPIO_AF_UART_1
	1001: GPIO_AF_CAN
	1010: GPIO_AF_USB
	1011: AF11
	1100: GPIO_AF_TK80_SDIO
	1101: GPIO_AF_Touchpad
	1110: GPIO_AF_LTDC
	1111: AF15

## 6. 系统配置控制器

TK499 具有一组系统配置寄存器。这些寄存器的主要功能如下：

- 重映射部分外设的 DMA 触发源到其它不同的 DMA 通道上。
- 管理连接到 GPIO 口的外部中断。
- 重映射存储器到代码起始区域。
- 外部中断引脚配置

### 6.1 SYSCFG 寄存器描述

#### 6.1.1 SYSCFG 配置寄存器 (SYSCFG\_CFGR)

该寄存器专门用于配置内存起始区域映射和 DMA 请求重映射。具有两个可配置内存起始 0x0000 0000 地址存储区类型的控制位，这两个控制位可软件配置来屏蔽 BOOT 的选择。复位后，这两个控制位为实际的 BOOT 模式配置。

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI3_RX_D	SPI3_TX_D	UART3_TX_D	I2C1_RX_D	I2C1_RX_D	I2C2_RX_D	TIM3_CH2	TIM3_CH4	TIM3_UP_D	TIM3_CH3	TIM4_CH4	TIM4_UP_D	TIM4_UP_D	TIM8_UP_D	TIM9_UP_D	ADC1_DMA
MA1_RMP	MA1_RMP	DMA1_RMP	MA1_RMP	MA1_RMP	MA1_RMP	DMA1_RMP	DMA1_RMP	MA1_RMP	DMA1_RMP	DMA1_RMP	MA1_RMP	MA1_RMP	MA1_RMP	MA1_RMP	2_RM_P
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
SPI1_RX_D	SPI1_TX_D	SPI4_RX_D	SPI4_TX_D	UART1_RX_D	TIM1_CH1	TIM1_CH2	TIM1_TRIGGER	TIM2_CH2	TIM2_CH3	QSPI_RX_D	QSPI_TX_D	SDIO1_DMA	SDIO2_DMA	MEM_MODE	
MA2_RMP	MA2_RMP	MA2_RMP	MA2_RMP	DMA2_RMP	DMA2_RMP	DMA2_RMP	DMA2_RMP	DMA2_RMP	DMA2_RMP	MA2_RMP	MA2_RMP	2_RM_P	2_RM_P		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

位31	<b>SPI3_RX_DMA1_RMP:</b> SPI3 DMA1请求重映射位 (SPI3 DMA1 request remapping bit) 由软件设置和清除该位。它控制着SPI3 DMA1通道请求的重映射。 0: 无重映射 (SPI3_RX DMA1请求映射在DMA1通道1上) 1: 重映射 (SPI3_RX DMA1请求映射在DMA1通道3上)
位30	<b>SPI3_TX_DMA1_RMP:</b> SPI3 DMA1请求重映射位 (SPI3 DMA1 request remapping bit) 由软件设置和清除该位。它控制着SPI3 DMA1通道请求的重映射。 0: 无重映射 (SPI3_TX DMA1请求映射在DMA1通道6上) 1: 重映射 (SPI3_TX DMA1请求映射在DMA1通道8上)
位29	<b>UART3_TX_DMA1_RMP:</b> UART3_TX DMA1请求重映射位 (UART3_TX DMA1 request remapping bit) 由软件设置和清除该位。它控制着UART3_TX DMA1通道请求的重映射。 0: 无重映射 (UART3_TX DMA1请求映射在DMA1通道4上) 1: 重映射 (UART3_TX DMA1请求映射在DMA1通道5上)
位28	<b>I2C1_RX_DMA1_RMP:</b> I2C2_RX DMA1 请求重映射位 (I2C2_RX DMA1 request remapping bit) 由软件设置和清除该位。它控制着I2C2_RX DMA1通道请求的重映射。 0: 无重映射 (I2C2_RX DMA1请求映射在DMA1通道1上) 1: 重映射 (I2C2_RX DMA1请求映射在DMA1通道6上)
位27	<b>I2C1_TX_DMA1_RMP:</b> I2C2_TX DMA1 请求重映射位 (I2C2_TX DMA1 request remapping bit) 由软件设置和清除该位。它控制着I2C2_TX DMA1通道请求的重映射。 0: 无重映射 (I2C2_TX DMA1请求映射在DMA1通道7上) 1: 重映射 (I2C2_TX DMA1请求映射在DMA1通道8上)

位26	<b>I2C2_RX_DMA1_RMP:</b> I2C2_RX DMA1 请求重映射位 (I2C2_RX DMA1 request remapping bit) 由软件设置和清除该位。它控制着I2C2_RX DMA1通道请求的重映射。 0: 无重映射 (I2C2_RX DMA1请求映射在DMA1通道3上) 1: 重映射 (I2C2_RX DMA1请求映射在DMA1通道4上)
位25	<b>TIM3_CH2_DMA1_RMP:</b> TIM3 DMA1请求重映射位 (TIM3 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM3 DMA1通道请求的重映射。 0: 无重映射 (TIM3_CH2 DMA1请求分别映射在DMA1通道6上) 1: 重映射 (TIM3_CH2 DMA1请求分别映射在DMA1通道7上)
位24	<b>TIM3_CH4_DMA1_RMP:</b> TIM3 DMA1请求重映射位 (TIM3 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM3 DMA1通道请求的重映射。 0: 无重映射 (TIM3_CH4 DMA1请求分别映射在DMA1通道3上) 1: 重映射 (TIM3_CH4 DMA1请求分别映射在DMA1通道7上)
位23	<b>TIM3_UP_DMA1_RMP:</b> TIM3 DMA1请求重映射位 (TIM3 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM3 DMA1通道请求的重映射。 0: 无重映射 (TIM3_UP DMA1请求分别映射在DMA1通道3上) 1: 重映射 (TIM3_UP DMA1请求分别映射在DMA1通道8上)
位22	<b>TIM4_CH3_DMA1_RMP:</b> TIM4 DMA1请求重映射位 (TIM4 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM4 DMA1通道请求的重映射。 0: 无重映射 (TIM4_CH3 DMA1请求分别映射在DMA1通道1上) 1: 重映射 (TIM4_CH3 DMA1请求分别映射在DMA1通道8上)
位21	<b>TIM4_CH4_DMA1_RMP:</b> TIM4 DMA1请求重映射位 (TIM4 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM4 DMA1通道请求的重映射。 0: 无重映射 (TIM4_CH4 DMA1请求分别映射在DMA1通道2上) 1: 重映射 (TIM4_CH4 DMA1请求分别映射在DMA1通道4上)
位20	<b>TIM4_UP_DMA1_RMP:</b> TIM4 DMA1请求重映射位 (TIM4 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM4 DMA1通道请求的重映射。 0: 无重映射 (TIM4_UP DMA1请求分别映射在DMA1通道1上) 1: 重映射 (TIM4_UP DMA1请求分别映射在DMA1通道7上)
位19	<b>TIM8_UP_DMA1_RMP:</b> TIM8 DMA1请求重映射位 (TIM8 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM8 DMA1通道请求的重映射。 0: 无重映射 (TIM8_UP DMA1请求映射在DMA1通道2上) 1: 重映射 (TIM8_UP DMA1请求映射在DMA1通道6上)
位18	<b>TIM9_UP_DMA1_RMP:</b> TIM9 DMA1请求重映射位 (TIM9 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM9 DMA1通道请求的重映射。 0: 无重映射 (TIM9_UP DMA1请求映射在DMA1通道3上) 1: 重映射 (TIM9_UP DMA1请求映射在DMA1通道5上)
位17	<b>TIM10_UP_DMA1_RMP:</b> TIM10 DMA1请求重映射位 (TIM10 DMA1 request remapping bit) 由软件设置和清除该位。它控制着TIM10 DMA1通道请求的重映射。 0: 无重映射 (TIM10_UP DMA1请求映射在DMA1通道1上) 1: 重映射 (TIM10_UP DMA1请求映射在DMA1通道4上)
位16	<b>ADC1_DMA2_RMP:</b> ADC1 DMA2请求重映射位 (ADC1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着ADC1 DMA2通道请求的重映射 0: 无重映射 (ADC1 DMA2请求映射在DMA2通道1上) 1: 重映射 (ADC1 DMA2请求映射在DMA2通道5上)
位15	<b>SPI1_RX_DMA2_RMP:</b> SPI1 DMA2 请求重映射位 (SPI1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SPI1 DMA2通道请求的重映射。 0: 无重映射 (SPI1_RX DMA2请求分别映射在DMA2通道1上) 1: 重映射 (SPI1_RX DMA2请求分别映射在DMA2通道3上)
位14	<b>SPI1_TX_DMA2_RMP:</b> SPI1 DMA2 请求重映射位 (SPI1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SPI1 DMA2通道请求的重映射。 0: 无重映射 (SPI1_TX DMA2请求分别映射在DMA2通道4上) 1: 重映射 (SPI1_TX DMA2请求分别映射在DMA2通道6上)
位13	<b>SPI4_RX_DMA2_RMP:</b> SPI4 DMA2 请求重映射位 (SPI4 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SPI4 DMA2通道请求的重映射。 0: 无重映射 (SPI4_RX DMA2请求分别映射在DMA2通道1上) 1: 重映射 (SPI4_RX DMA2请求分别映射在DMA2通道4上)

位12	<b>SPI4_TX_DMA2_RMP:</b> SPI4 DMA2 请求重映射位 (SPI4 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SPI4 DMA2通道请求的重映射。 0: 无重映射 (SPI4_TX DMA2请求分别映射在DMA2通道2上) 1: 重映射 (SPI4_TX DMA2请求分别映射在DMA2通道5上)
位11	<b>UART1_RX_DMA2_RMP:</b> UART1_RX DMA2 请求重映射位 (UART1_RX DMA2 request remapping bit) 由软件设置和清除该位。它控制着UART1_RX DMA2通道请求的重映射。 0: 无重映射 (UART1_RX DMA2请求映射在DMA2通道3上) 1: 重映射 (UART1_RX DMA2请求映射在DMA2通道6上)
位10	<b>TIM1_CH1_DMA2_RMP:</b> TIM1 DMA2请求重映射位 (TIM1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着TIM1 DMA2通道请求的重映射。 0: 无重映射 (TIM1_CH1 DMA2请求分别映射在DMA2通道4上) 1: 重映射 (TIM1_CH1 DMA2请求分别映射在DMA2通道7上)
位9	<b>TIM1_CH2_DMA2_RMP:</b> TIM1 DMA2请求重映射位 (TIM1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着TIM1 DMA2通道请求的重映射。 0: 无重映射 (TIM1_CH2 DMA2请求分别映射在DMA2通道3上) 1: 重映射 (TIM1_CH2 DMA2请求分别映射在DMA2通道7上)
位8	<b>TIM1_TRIG_DMA2_RMP:</b> TIM1 DMA2请求重映射位 (TIM1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着TIM1 DMA2通道请求的重映射。 0: 无重映射 (TIM1_TRIG DMA2请求分别映射在DMA2通道1上) 1: 重映射 (TIM1_TRIG DMA2请求分别映射在DMA2通道5上)
位7	<b>TIM2_CH2_DMA2_RMP:</b> TIM2 DMA2请求重映射位 (TIM2 DMA2 request remapping bit) 由软件设置和清除该位。它控制着TIM2 DMA2通道请求的重映射。 0: 无重映射 (TIM2_CH2 DMA2请求映射在DMA2通道3上) 1: 重映射 (TIM2_CH2 DMA2请求分别映射在DMA2通道4上)
位6	<b>TIM2_CH3_DMA2_RMP:</b> TIM2 DMA2请求重映射位 (TIM2 DMA2 request remapping bit) 由软件设置和清除该位。它控制着TIM2 DMA2通道请求的重映射。 0: 无重映射 (TIM2_CH3 DMA2请求映射在DMA2通道3上) 1: 重映射 (TIM2_CH3 DMA2请求分别映射在DMA2通道5上)
位5	<b>QSPI_RX_DMA2_RMP:</b> QSPI DMA2请求重映射位 (QSPI DMA2 request remapping bit) 由软件设置和清除该位。它控制着QSPI_RX DMA2通道请求的重映射 0: 无重映射 (QSPI_RX DMA2请求分别映射在DMA2通道4上) 1: 重映射 (QSPI_RX DMA2请求分别映射在DMA2通道6上)
位4	<b>QSPI_TX_DMA2_RMP:</b> QSPI DMA2请求重映射位 (QSPI DMA2 request remapping bit) 由软件设置和清除该位。它控制着QSPI_TX DMA2通道请求的重映射 0: 无重映射 (QSPI_TX DMA2请求分别映射在DMA2通道5上) 1: 重映射 (QSPI_TX DMA2请求分别映射在DMA2通道7上)
位3	<b>SDIO1_DMA2_RMP:</b> SDIO1 DMA2请求重映射位 (SDIO1 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SDIO1 DMA2通道请求的重映射 0: 无重映射 (SDIO1 DMA2请求映射在DMA2通道4上) 1: 重映射 (SDIO1 DMA2请求映射在DMA2通道7上)
位2	<b>SDIO2_DMA2_RMP:</b> SDIO2 DMA2请求重映射位 (SDIO2 DMA2 request remapping bit) 由软件设置和清除该位。它控制着SDIO2 DMA2通道请求的重映射 0: 无重映射 (SDIO2 DMA2请求映射在DMA2通道2上) 1: 重映射 (SDIO2 DMA2请求映射在DMA2通道8上)
位1: 0	<b>MEM_MODE:</b> 存储映射选择位 (Memory selection bit) 由软件设置和清除这些位。它控制存储器内部映射到地址0x0000 0000。 00: ROM映射到0x0000 0000 其他: 保留

### 6.1.2 外部中断配置寄存器 1 (SYSCFG\_EXTICR1)

偏移地址: 0x08

复位值: 0x0000\_0000

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

位31: 16	保留。
位15: 0	EXTI3[3: 0], EXTI2[3: 0], EXTI1[3: 0], EXTI0[3: 0] EXTIx配置 ( $x = 0 \dots 3$ ) (EXTIx configuration) 这些位可用于软件读写。用于选择EXTIx外部中断的输入源。 0000: PA[x]管脚 0001: PB[x]管脚 0010: PC[x]管脚 0011: PD[x]管脚 0100: PE[x]管脚

### 6.1.3 外部中断配置寄存器 2 (SYSCFG\_EXTICR2)

偏移地址: 0x0C

复位值: 0x0000\_0000

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

位31: 16	保留。
位15: 0	EXTI7[3: 0], EXTI6[3: 0], EXTI5[3: 0], EXTI4[3: 0] EXTIx配置 ( $x = 7 \dots 4$ ) (EXTIx configuration) 这些位可用于软件读写。用于选择EXTIx外部中断的输入源。 0000: PA[x]管脚 0001: PB[x]管脚 0010: PC[x]管脚 0011: PD[x]管脚 0100: PE[x]管脚

### 6.1.4 外部中断配置寄存器 3 (SYSCFG\_EXTICR3)

偏移地址: 0x10

复位值: 0x0000\_0000

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

位31: 16	保留。
位15: 0	EXTI11[3: 0], EXTI10[3: 0], EXTI9[3: 0], EXTI8[3: 0] EXTIx配置 ( $x = 11\dots8$ ) (EXTIx configuration) 这些位可用于软件读写。用于选择EXTIx外部中断的输入源。 0000: PA[x]管脚 0001: PB[x]管脚 0010: PC[x]管脚 0011: PD[x]管脚 0100: PE[x]管脚

### 6.1.5 外部中断配置寄存器 4 (SYSCFG\_EXTICR4)

偏移地址: 0x14

复位值: 0x0000\_0000

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

位31: 16	保留。
位15: 0	EXTI15[3: 0], EXTI14[3: 0], EXTI13[3: 0], EXTI12[3: 0] EXTIx配置 ( $x=15\dots12$ ) (EXTIx configuration) 这些位可用于软件读写。用于选择EXTIx外部中断的输入源。 0000: PA[x]管脚 0001: PB[x]管脚 0010: PC[x]管脚 0011: PD[x]管脚 0100: PE[x]管脚 (PE[23:16]不支持外部中断功能)

## 7. DMA 控制器 (DMA)

### 7.1 DMA 简介

直接存储器存取用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须 CPU 任何干预，通过 DMA 数据可以快速地移动。这就节省了 CPU 的资源来做其他操作。

包含 DMA1 和 DMA2，每个 DMA 控制器有 8 个通道。

### 7.2 DMA 主要特征

- 8 个独立的可配置的通道。
- 每个通道都直接连接专用的硬件 DMA 请求，每个通道都同样支持软件触发。这些功能通过软件来配置。
- 在七个请求间的优先权可以通过软件编程设置(共有四级：很高、高、中等和低)，假如在相等优先权时由硬件决定(请求 0 优先于请求 1，依此类推)。
- 独立的源和目标数据区的传输宽度(字节、半字、全字)，模拟打包和拆包的过程。源和目标地址必须按数据传输宽度对齐。
- 支持循环的缓冲器管理。
- 每个通道都有 3 个事件标志(DMA 半传输，DMA 传输完成和 DMA 传输出错)，这 3 个事件标志逻辑或成为一个单独的中断请求。
- 存储器和存储器间的传输。
- 外设和存储器，存储器和外设的传输。
- SRAM、外设的 SRAM、APB1、APB2 和 AHB 外设均可作为访问的源和目标。
- 可编程的数据传输数目：最大传输个数高达  $2^{32}=4,294,967,296$ ，对于高分辨率液晶屏可单次完成数据填充。

### 7.3 功能描述

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

#### 7.3.1 DMA 处理

在发生一个事件后，外设发送一个请求信号到 DMA 控制器。DMA 控制器根据通道的优先权处理请求。当 DMA 控制器开始访问外设的时候，DMA 控制器立即发送给外设一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器同时撤销应答信号。如果发生更多的请求时，外设可以启动下次处理。

总之，每个 DMA 传送由 3 个操作组成：

- 从外设数据寄存器或者从 DMA\_CMARx 寄存器指定地址的存储器单元执行加载操作。
- 存数据到外设数据寄存器或者存数据到 DMA\_CMARx 寄存器指定地址的存储器单元。
- 执行一次 DMA\_CNDTRx 寄存器的递减操作。该寄存器包含未完成的操作数目。

#### 7.3.2 仲裁器

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

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

### 7.3.3 DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。DMA 传输的数据量是可编程的，最大达到 4,294,967,296。包含要传输的数据项数量的寄存器，在每次传输后递减。

#### 可编程的数据量

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

#### 指针增量

通过设置 DMA\_CCRx 寄存器中 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以有选择地完成自动增量。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决于与所选的数据宽度为 1、2 或 4。第一个传输的地址存放在 DMA\_CPARx/DMA\_CMARx 寄存器中。

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。

#### 通道配置

下面是配置 DMA 通道 x 的过程(x 代表通道号)：

1. 在 DMA\_CPARx 寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在 DMA\_CMARx 寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
3. 在 DMA\_CNDTRx 寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减。
4. 在 DMA\_CCRx 寄存器的 PL[1: 0]位中设置通道的优先级。
5. 在 DMA\_CCRx 寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置 DMA\_CCRx 寄存器的 ENABLE 位，启动该通道。一旦启动了 DMA 通道，它既可响应联到该通道上的外设的 DMA 请求。

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

#### 循环模式

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

#### 存储器到存储器模式

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

### 7.3.4 可编程的数据传输宽度，对齐方式和数据大小端

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

表 15. 可编程的数据传输宽度和大小端操作 (当 PINC=MINC=1)

源端宽度	目标宽度	传输数目	源： 地址/数据	传输操作	目标： 地址/数据
8	8	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7: 0], 在0x0写B0[7: 0] 2: 在0x1读B1[7: 0], 在0x1写B1[7: 0] 3: 在0x2读B2[7: 0], 在0x2写B2[7: 0] 4: 在0x3读B3[7: 0], 在0x3写B3[7: 0]	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3

源端宽度	目标宽度	传输数目	源: 地址/数据	传输操作	目标: 地址/数据
8	16	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7: 0], 在0x0写00B0[15: 0] 2: 在0x1读B1[7: 0], 在0x2写00B1[15: 0] 3: 在0x2读B2[7: 0], 在0x4写00B2[15: 0] 4: 在0x3读B3[7: 0], 在0x6写00B3[15: 0]	0x0 / 00B0 0x2 / 00B1 0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7: 0], 在0x0写000000B0[31: 0] 2: 在0x1读B1[7: 0], 在0x4写000000B1[31: 0] 3: 在0x2读B2[7: 0], 在0x8写000000B2[31: 0] 4: 在0x3读B3[7: 0], 在0xC写000000B3[31: 0]	0x0 / 000000B0 0x4 / 000000B1 0x8 / 000000B2 0xC / 000000B3
16	8	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15: 0], 在0x0写B0[7: 0] 2: 在0x2读B3B2[15: 0], 在0x1写B2[7: 0] 3: 在0x4读B5B4[15: 0], 在0x2写B4[7: 0] 4: 在0x6读B7B6[15: 0], 在0x3写B6[7: 0]	0x0 / B0 0x1 / B2 0x2 / B4 0x3 / B6
16	16	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15: 0], 在0x0写B1B0[15: 0] 2: 在0x2读B3B2[15: 0], 在0x2写B3B2[15: 0] 3: 在0x4读B5B4[15: 0], 在0x4写B5B4[15: 0] 4: 在0x6读B7B6[15: 0], 在0x6写B7B6[15: 0]	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15: 0], 在0x0写0000B1B0[31: 0] 2: 在0x2读B3B2[15: 0], 在0x4写0000B3B2[31: 0] 3: 在0x4读B5B4[15: 0], 在0x8写0000B5B4[31: 0] 4: 在0x6读B7B6[15: 0], 在0xC写0000B7B6[31: 0]	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31: 0], 在0x0写B0[7: 0] 2: 在0x4读B7B6B5B4[31: 0], 在0x1写B4[7: 0] 3: 在0x8读BBBAB9B8[31: 0], 在0x2写B8[7: 0] 4: 在0xC读BFBEBDBC[31: 0], 在0x3写BC[7: 0]	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31: 0], 在0x0写B1B0[15: 0] 2: 在0x4读B7B6B5B4[31: 0], 在0x2写B5B4[15: 0] 3: 在0x8读BBBAB9B8[31: 0], 在0x4写B9B8[15: 0] 4: 在0xC读BFBEBDBC[31: 0], 在0x6写BDBC[15: 0]	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	32	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31: 0], 在0x0写B3B2B1B0[31: 0] 2: 在0x4读B7B6B5B4[31: 0], 在0x4写B7B6B5B4[31: 0] 3: 在0x8读BBBAB9B8[31: 0], 在0x8写BBBAB9B8[7: 0] 4: 在0xC读BFBEBDBC[31: 0], 在0xC写BFBEBDBC[31: 0]	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC

### 操作一个不支持字节或半字写的 AHB 设备

当 DMA 模块开始一个 AHB 的字节或半字写操作时, 数据将在 HWDATA[31: 0]总线中未使用的部分重复。因此, 如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时(即 HSIZEx 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 HSIZEx=半字时, 写入半字 ‘0xABCD’ , DMA 将设置 HWDATA 总线为 ‘0xABCDABCD’ 。
- 当 HSIZEx=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 HSIZEx 参数, 它将按照下述方式把任何 AHB 上的字节或半字按 32 位传送到 APB 上:

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

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

### 7.3.5 错误管理

读写一个保留的地址区域, 将会产生 DMA 传输错误。当在 DMA 读写操作时发生 DMA 传输错误时, 硬件会自动地清除发生错误的通道所对应的通道配置寄存器 (DMA\_CCRx) 的 EN 位, 该通道 操作被停止。

此时，在 DMA\_IFT 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。

### 7.3.6 中断

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

表 16. DMA 中断请求

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

### 7.3.7 DMA 请求映像

#### DMA 控制器

从外设产生的 DMA 请求，通过逻辑或输入到 DMA 控制器，这意味着同时只能有一个请求有效。外设的 DMA 请求，可以通过设置相应外设寄存器中的控制位，被独立地开启或关闭。参见下图的 DMA1,DMA2 映射一览表。

表 17. DMA1 通道映射一览表

Peripherals	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8
SPI2				SPI2_RX	SPI2_TX			
SPI3	SPI3_RX <sup>(1)</sup>		SPI3_RX <sup>(2)</sup>			SPI3_TX <sup>(1)</sup>		SPI3_TX <sup>(2)</sup>
UART2						UART2_RX	UART2_TX	
UART3		UART3_RX		UART3_TX <sup>(1)</sup>	UART3_TX <sup>(2)</sup>			
UART4			UART4_RX		UART4_TX			
UART5	UART5_RX							UART5_TX
I2C1	I2C1_RX <sup>(1)</sup>					I2C1_RX <sup>(2)</sup>	I2C1_TX <sup>(1)</sup>	I2C1_TX <sup>(2)</sup>
I2C2			I2C2_RX <sup>(1)</sup>	I2C2_RX <sup>(2)</sup>				I2C2_TX
I2C3			I2C3_RX		I2C3_TX			
I2C4		I2C4_TX <sup>(1)</sup>				I2C4_RX	I2C4_TX <sup>(2)</sup>	
TIM3			TIM3_CH4 <sup>(1)</sup> TIM3_UP <sup>(1)</sup>		TIM3_CH1 TIM3_TRIG	TIM3_CH2 <sup>(1)</sup>	TIM3_CH2 <sup>(2)</sup> TIM3_CH4 <sup>(2)</sup>	TIM3_CH3 TIM3_UP <sup>(2)</sup>
TIM4	TIM4_CH3 <sup>(1)</sup> TIM4_UP <sup>(1)</sup>	TIM4_CH4 <sup>(1)</sup> TIM4_TRIG	TIM4_CH1	TIM4_CH4 <sup>(2)</sup> TIM4_TRIG	TIM4_CH2		TIM4_UP <sup>(2)</sup>	TIM4_CH3 <sup>(2)</sup>
TIM8		TIM8_UP <sup>(1)</sup>				TIM8_UP <sup>(2)</sup>		
TIM9			TIM9_UP <sup>(1)</sup>		TIM9_UP <sup>(2)</sup>			
TIM10	TIM10_UP <sup>(1)</sup>			TIM10_UP <sup>(2)</sup>				

表 18. DMA2 通道映射一览表

Peripherals	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8
ADC	ADC <sup>(1)</sup>				ADC <sup>(2)</sup>			
SPI1	SPI1_RX <sup>(1)</sup>		SPI1_RX <sup>(2)</sup>	SPI1_TX <sup>(1)</sup>		SPI1_TX <sup>(2)</sup>		
SPI4	SPI4_RX <sup>(1)</sup>	SPI4_TX <sup>(1)</sup>		SPI4_RX <sup>(2)</sup>	SPI4_TX <sup>(2)</sup>			
UART1			UART1_RX <sup>(1)</sup>			UART1_RX <sup>(2)</sup>		UART1_TX
TIM1	TIM1_TRIG <sup>(1)</sup>		TIM1_CH2 <sup>(1)</sup>	TIM1_CH1 <sup>(1)</sup>	TIM1_CH4 TIM1_TRIG <sup>(2)</sup> TIM1_COM	TIM1_UP	TIM1_CH1 <sup>(2)</sup> TIM1_CH2 <sup>(2)</sup> TIM1_CH3	
TIM2		TIM2_UP	TIM2_CH1 TIM2_CH2 <sup>(1)</sup> TIM2_CH3 <sup>(1)</sup>	TIM2_CH2 <sup>(2)</sup>	TIM2_CH3 <sup>(2)</sup>			TIM2_CH4 TIM2_TRIG TIM2_COM
QSPI				QSPI_RX <sup>(1)</sup>	QSPI_TX <sup>(1)</sup>	QSPI_RX <sup>(2)</sup>	QSPI_TX <sup>(2)</sup>	
SDIO1				SDIO1 <sup>(1)</sup>			SDIO1 <sup>(2)</sup>	
SDIO2		SDIO2 <sup>(1)</sup>						SDIO2 <sup>(2)</sup>
USB				USB[0]	USB[1]			

1. 如果 SYSCFG\_CFGR 寄存器的映射位被清除, 带 (1) 上标的请求被映射在这个 DMA 通道
2. 如果 SYSCFG\_CFGR 寄存器的映射位被置位, 带 (2) 上标的请求被映射在这个 DMA 通道

## 7.4 DMA 寄存器描述

### 7.4.1 DMA 中断状态寄存器(DMA\_ISR)

偏移地址: 0x00

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEIF8	HTIF8	TCIF8	GIF8	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31 , 27, 23, 19 , 15, 11, 7 , 3	<b>TEIFx:</b> 通道x的传输错误标志(x = 1 ... 8) (Channel x transfer error flag) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入'1'可以清除这里对应的标志位。 0: 在通道x没有传输错误(TE); 1: 在通道x发生了传输错误(TE)。
位 30 , 26, 22, 18 , 14, 10, 6 , 2	<b>HTIFx:</b> 通道x的半传输标志(x = 1 ... 8) (Channel x half transfer flag) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入'1'可以清除这里对应的标志位。 0: 在通道x没有半传输事件(HT); 1: 在通道x产生了半传输事件(HT)。
位 29 , 25, 21, 17 , 13, 9, 5 , 1	<b>TCIFx:</b> 通道x的传输完成标志(x = 1 ... 8) (Channel x transfer complete flag) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入'1'可以清除这里对应的标志位。 0: 在通道x没有传输完成事件(TC); 1: 在通道x产生了传输完成事件(TC)。
位 28 , 24, 20, 16 , 12, 8, 4 , 0	<b>GIFx:</b> 通道x的全局中断标志(x = 1 ... 8) (Channel x global interrupt flag) 硬件设置这些位。在DMA_IFCR寄存器的相应位写入'1'可以清除这里对应的标志位。 0: 在通道x没有TE、HT或TC事件; 1: 在通道x产生了TE、HT或TC事件。

### 7.4.2 DMA 中断标志清除寄存器(DMA\_IFCR)

偏移地址: 0x04

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTE IF8	CHT IF8	CTC IF8	CG IF8	CTE IF7	CHT IF7	CTC IF7	CG IF7	CTE IF6	CHT IF6	CTC IF6	CG IF6	CTE IF5	CHT IF5	CTC IF5	CG IF5
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31 , 27, 23, 19 , 15, 11, 7 , 3	<b>CTEIFx:</b> 清除通道x的传输错误标志(x = 1 ... 8) (Channel x transfer error clear) 这些位由软件设置和清除。 0: 不起作用 1: 清除DMA_ISR寄存器中的对应TEIF标志。
位 30 , 26, 22, 18 , 14, 10, 6 , 2	<b>CHTIFx:</b> 清除通道x的半传输标志(x = 1 ... 8) (Channel x half transfer clear) 这些位由软件设置和清除。 0: 不起作用 1: 清除DMA_ISR寄存器中的对应HTIF标志。
位 29 , 25, 21, 17 , 13, 9, 5 , 1	<b>TCEIFx:</b> 清除通道x的传输完成标志(x = 1 ... 8) (Channel x transfer complete clear) 这些位由软件设置和清除。 0: 不起作用 1: 清除DMA_ISR寄存器中的对应TCIF标志。
位 28 , 24, 20, 16 , 12, 8, 4 , 0	<b>CGIFx:</b> 清除通道x的全局中断标志(x = 1 ... 8) (Channel x global interrupt clear) 这些位由软件设置和清除。 0: 不起作用 1: 清除DMA_ISR寄存器中的对应的GIF、TEIF、HTIF和TCIF标志。

### 7.4.3 DMA 通道 x 配置寄存器(DMA\_CCRx) (x = 1…8)

偏移地址: 0x08 + 20 x (通道编号-1)

复位值: 0x0000 0000

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

位31: 15	保留, 始终读为0。
位14	<b>MEM2MEM:</b> 存储器到存储器模式 (Memory to memory mode) 该位由软件设置和清除。 0: 非存储器到存储器模式 1: 启动存储器到存储器模式
位13: 12	<b>PL[1: 0]:</b> 通道优先级 (Channel priority level) 这些位由软件设置和清除。 00: 低 01: 中 10: 高 11: 最高

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

#### 7.4.4 DMA 通道 x 传输数量寄存器(DMA\_CNDTRx) (x = 1…8)

偏移地址: 0x0C + 20 x (通道编号-1)

复位值: 0x0000 0000

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

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

#### 7.4.5 DMA 通道 x 外设地址寄存器(DMA\_CPARx) (x = 1…8)

偏移地址: 0x10 + 20 x (通道编号-1)

复位值: 0x0000 0000

当开启通道(DMA\_CCRx 的 EN=1)时不能写该寄存器。

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

位31: 0	<b>PA[31: 0]:</b> 外设地址 (Peripheral address) 外设数据寄存器的基址, 作为数据传输的源或目标。 当PSIZE= ‘01’ (16位), 不使用PA[0]位。操作自动地与半字地址对齐。 当PSIZE= ‘10’ (32位), 不使用PA[1: 0]位。操作自动地与字地址对齐。
--------	---

#### 7.4.6 DMA 通道 x 存储器地址寄存器(DMA\_CMARx) (x = 1…8)

偏移地址: 0x14 + 20 x (通道编号-1)

复位值: 0x0000 0000

当开启通道(DMA\_CCRx 的 EN=1)时不能写该寄存器。

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

位31: 0	<b>MA[31: 0]:</b> 存储器地址 (Memory address) 存储器地址作为数据传输的源或目标。 当MSIZE= ‘01’ (16位), 不使用MA[0]位。操作自动地与半字地址对齐。 当MSIZE= ‘10’ (32位), 不使用MA[1: 0]位。操作自动地与字地址对齐。
--------	---

## 8. 中断和事件

### 8.1 嵌套向量中断控制器

特征

- 中断都可屏蔽（除了 NMI）
- 16 个可编程的优先等级（使用了 4 位中断优先级）
- 低延迟的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

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

嵌套向量中断控制器管理着包括核异常等中断。关于更多的异常和 NVIC 编程的说明请参考 CPU 技术参考手册。

#### 8.1.1 系统嘀嗒（SysTick）校准值寄存器

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

#### 8.1.2 中断和异常向量

下表列出了产品向量表。

表 33. 产品向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	Reset	复位	0x0000_0004
	-2	固定	NMI	不可屏蔽中断	
RCC时钟安全系统（CSS）联接到NMI向量	0x0000_0008				
	-1	固定	硬件失效（HardFault）	所有类型的失效	0x0000_000C
	0	可设置	存储管理（MemManage）	存储器管理	0x0000_0010
	1	可设置	总线错误（BusFault）	预取指失败，存储器访问失败	0x0000_0014
	2	可设置	错误应用（UsageFault）	未定义的指令或非法状态	0x0000_0018
	-	-	-	保留	0x0000_001C
~0x0000_002B					
	3	可设置	SVCall	通过SWI指令的系统服务调用	0x0000_002C
	4	可设置	调试监控（DebugMonitor）	调试监控器	0x0000_0030
	-	-	-	保留	0x0000_0034
	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	-	-	-	保留	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048
3	10	可设置	RTC	实时时钟（RTC）全局中断	0x0000_004C

位置	优先级	优先级类型	名称	说明	地址
4	-	-	-	保留	0x0000_0050
5	12	可设置	RCC	复位和时钟控制 (RCC) 中断	0x0000_0054
6	13	可设置	EXTI0	EXTI线0中断	0x0000_0058
7	14	可设置	EXTI1	EXTI线1中断	0x0000_005C
8	15	可设置	EXTI2	EXTI线2中断	0x0000_0060
9	16	可设置	EXTI3	EXTI线3中断	0x0000_0064
10	17	可设置	EXTI4	EXTI线4中断	0x0000_0068
11	18	可设置	DMA1通道1	DMA1通道1全局中断	0x0000_006C
12	19	可设置	DMA1通道2	DMA1通道2全局中断	0x0000_0070
13	20	可设置	DMA1通道3	DMA1通道3全局中断	0x0000_0074
14	21	可设置	DMA1通道4	DMA1通道4全局中断	0x0000_0078
15	22	可设置	DMA1通道5	DMA1通道5全局中断	0x0000_007C
16	23	可设置	DMA1通道6	DMA1通道6全局中断	0x0000_0080
17	24	可设置	DMA1通道7	DMA1通道7全局中断	0x0000_0084
18	25	可设置	ADC1	ADC1中断	0x0000_0088
19	26	可设置	CAN1	CAN1中断	0x0000_008C
20	-	-	-	保留	0x0000_0090
21	-	-	-	保留	0x0000_0094
22	-	-	-	保留	0x0000_0098
23	30	可设置	EXTI9_5	EXTI线[9: 5]中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1断开中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1捕获比较中断	0x0000_00AC
28	35	可设置	TIM3	TIM3全局中断	0x0000_00B0
29	36	可设置	TIM4	TIM4全局中断	0x0000_00B4
30	37	可设置	TIM5	TIM5全局中断	0x0000_00B8
31	38	可设置	TIM6	TIM6全局中断	0x0000_00BC
32	39	可设置	TIM7	TIM7全局中断	0x0000_00C0
33	40	可设置	I2C1	I2C1全局中断	0x0000_00C4
34	41	可设置	I2C2	I2C2全局中断	0x0000_00C8
35	42	可设置	SPI1	SPI1全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2全局中断	0x0000_00D0
37	44	可设置	UART1	UART1全局中断	0x0000_00D4
38	45	可设置	UART2	UART2全局中断	0x0000_00D8
39	46	可设置	UART3	UART3全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI线[15: 10]中断	0x0000_00E0
41	48	可设置	RTC_ALARM	连到EXTI17的RTC闹钟中断	0x0000_00E4

位置	优先级	优先级类型	名称	说明	地址
42	49	可设置	USB	连到EXTI18的从USB待机唤醒中断	0x0000_00E8
43	50	可设置	TIM2_BRK	TIM2断开中断	0x0000_00EC
44	51	可设置	TIM2_UP	TIM2更新中断	0x0000_00F0
45	52	可设置	TIM2_TRG_COM	TIM2触发和通信中断	0x0000_00F4
46	53	可设置	TIM2_CC	TIM2捕捉,比较中断	0x0000_00F8
47	54	可设置	DMA1通道8	DMA1通道8全局中断	0x0000_00FC
48	55	可设置	TK80	TK80全局中断	0x0000_0100
49	56	可设置	SDIO1	SDIO1全局中断	0x0000_0104
50	57	可设置	SDIO2	SDIO2全局中断	0x0000_0108
51	58	可设置	SPI3	SPI3全局中断	0x0000_010C
52	59	可设置	UART4	UART4全局中断	0x0000_0110
53	60	可设置	UART5	UART5全局中断	0x0000_0114
54	-	-	-	保留	0x0000_0118
55	62	可设置	TIM8	TIM8全局中断	0x0000_011C
56	63	可设置	DMA2通道1	DMA2通道1全局中断	0x0000_0120
57	64	可设置	DMA2通道2	DMA2通道2全局中断	0x0000_0124
58	65	可设置	DMA2通道3	DMA2通道3全局中断	0x0000_0128
59	66	可设置	DMA2通道4	DMA2通道4全局中断	0x0000_012C
60	67	可设置	DMA2通道5	DMA2通道5全局中断	0x0000_0130
61	68	可设置	TIM9	TIM9全局中断	0x0000_0134
62	69	可设置	TIM10	TIM10全局中断	0x0000_0138
63	70	可设置	CAN2	CAN2全局中断	0x0000_013C
64	-	-	-	保留	0x0000_0140
65	-	-	-	保留	0x0000_0144
66	-	-	-	保留	0x0000_0148
67	74	可设置	USB	USB全局中断	0x0000_014C
68	75	可设置	DMA2通道6	DMA2通道6全局中断	0x0000_0150
69	76	可设置	DMA2通道7	DMA2通道7全局中断	0x0000_0154
70	77	可设置	DMA2通道8	DMA2通道8全局中断	0x0000_0158
71	-	-	-	保留	0x0000_015C
72	79	可设置	I2C3	I2C3全局中断	0x0000_0160
73	80	可设置	I2C4	I2C4全局中断	0x0000_0164
74	-	-	-	保留	0x0000_0168
75	-	-	-	保留	0x0000_016C
76	-	-	-	保留	0x0000_0170
77	-	-	-	保留	0x0000_0174
78	-	-	-	保留	0x0000_0178
79	-	-	-	保留	0x0000_017C

位置	优先级	优先级类型	名称	说明	地址
80	-	-	-	保留	0x0000_0180
81	88	可设置	FPU	FPU全局中断	0x0000_0184
82	-	-	-	保留	0x0000_0188
83	-	-	-	保留	0x0000_018C
84	91	可设置	SPI4	SPI4全局中断	0x0000_0190
85	-	-	-	保留	0x0000_0194
86	93	可设置	TCHPAD	TCHPAD全局中断	0x0000_0198
87	94	可设置	QSPI	QPI4全局中断	0x0000_019C
88	95	可设置	LCD-TFT	LCD全局中断	0x0000_01A0
89	-	-	-	保留	0x0000_01A4
90	97	可设置	I2S1	I2S1全局中断	0x0000_01A8

## 8.2 外部中断/事件控制器 (EXTI)

外部中断和时间控制器 (EXTI) 管理外部和内部异步事件/中断，并生成相应的事件请求到 CPU/中断控制器和到电源管理的唤醒请求。

有 21 个能产生事件/中断请求的边沿检测器。每个输入线可以独立地配置输入类型（脉冲或挂起）和对应的触发事件（上升沿或下降沿或者双边沿都触发）。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

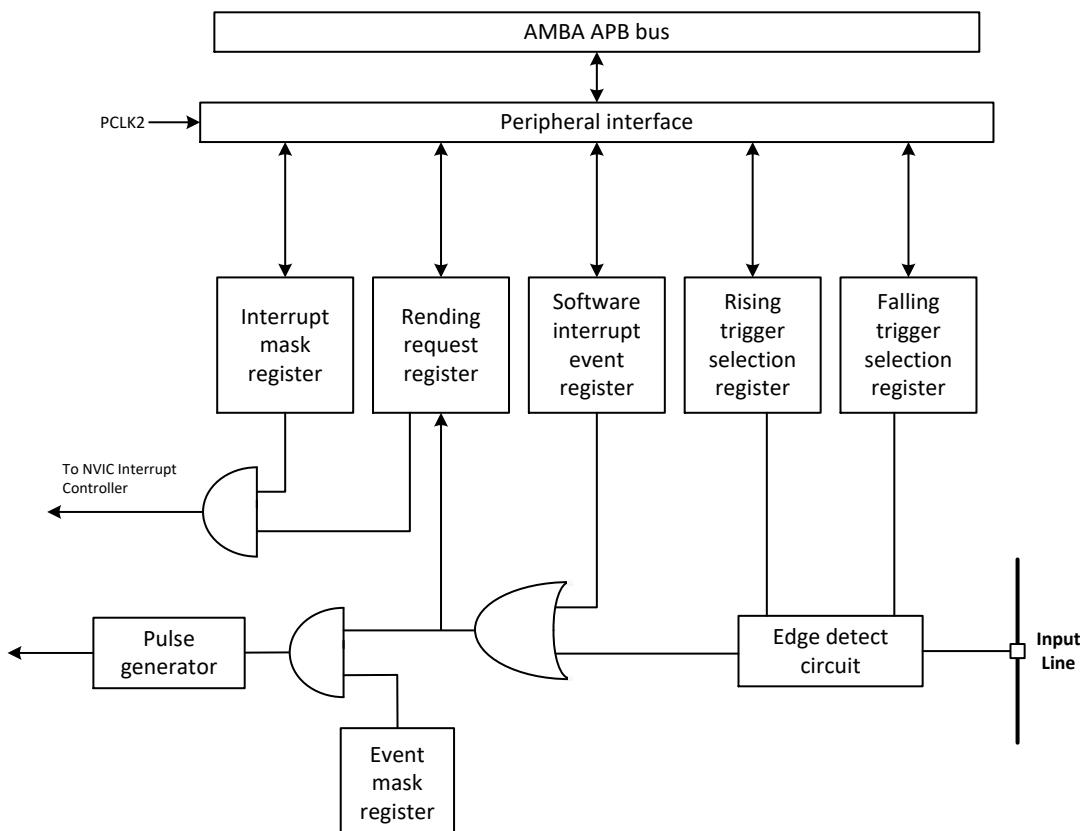
### 8.2.1 主要特征

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 21 个软件的中断/事件请求
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。参见数据手册中电气特性部分的相关参数。

### 8.2.2 框图

图 17. 外部中断/事件控制器框图



### 8.2.3 唤醒事件管理

TK499 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 CPU 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，请参见下节的功能说明

### 8.2.4 功能说明

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

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

通过在软件中断/事件寄存器写 ‘1’，也可以通过软件产生中断/事件请求。

硬件中断选择

通过下面的过程来配置 21 个线路做为中断源：

- 配置 21 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 16 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 21 个线路为事件源：

- 配置 2 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

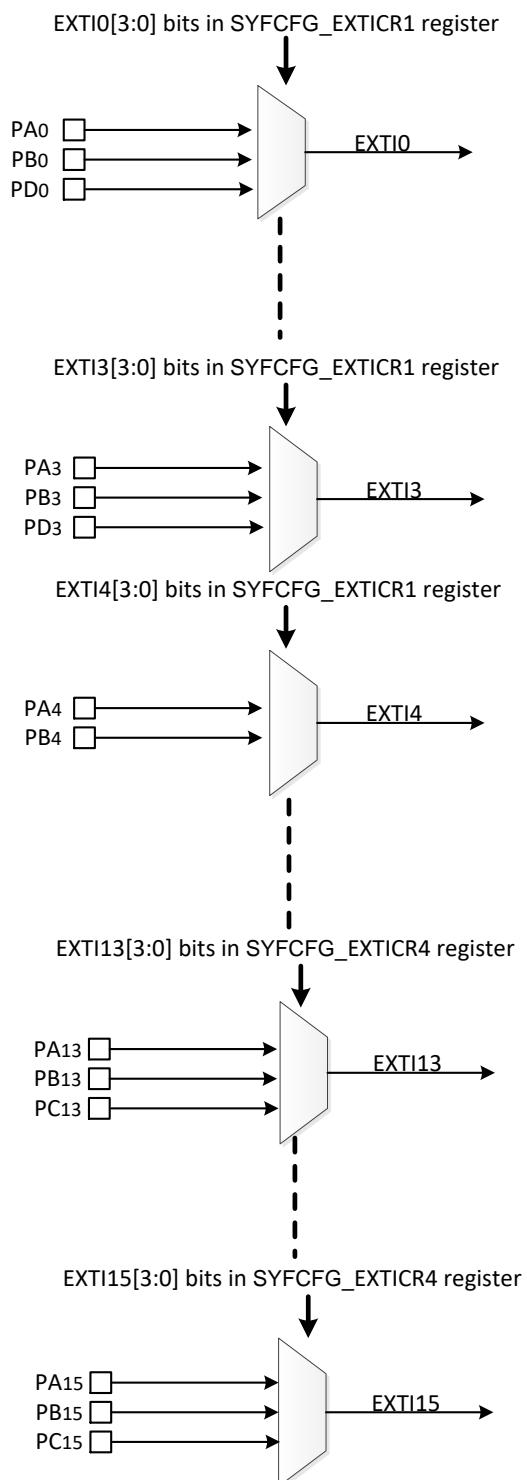
21 个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置 21 个中断/事件线屏蔽位（**EXTI\_IMR**, **EXTI\_EMR**）
- 设置软件中断寄存器的请求位（**EXTI\_SWIER**）

### 8.2.5 外部中断/事件线路映像

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 18. 外部中断通用 I/O 映像



另外三种其他的外部中断/事件控制器的连接如下：

- EXTI 线 16, 19, 20 保留
- EXTI 线 17 连接到 RTC 阔钟事件
- EXTI 线 18 连接到 USB 唤醒事件

## 8.3 EXTI 寄存器描述

### 8.3.1 中断屏蔽寄存器 (EXTI\_IMR)

偏移地址: 0x00

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														MR20	MR19
														MR18	MR17
														MR16	
														rw	rw
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 21	保留。
位20: 0	<b>IMRx:</b> 线x上的中断屏蔽 (Interrupt Mask on line x) 1 = 开放来自线x上的中断请求 0 = 屏蔽来自线x上的中断请求

### 8.3.2 事件屏蔽寄存器 (EXTI\_EMR)

偏移地址: 0x04

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														MR20	MR19
														MR18	MR17
														MR16	
														rw	rw
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 21	保留。
位20: 0	<b>EMRx:</b> 线x上的事件屏蔽 (Event Mask on line x) 1 = 开放来自线x上的事件请求 0 = 屏蔽来自线x上的事件请求

### 8.3.3 上升沿触发选择寄存器 (EXTI\_RTSR)

偏移地址: 0x08

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留												TR20	TR19	TR18	TR17	TR16
												rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

位31: 21	保留。
位20: 0	<b>TR<sub>x</sub>:</b> 线x上的上升沿触发事件配置位 (Rising trigger event configuration bit of line x) 1 = 允许输入线x上的上升沿触发 (中断和事件) 0 = 禁止输入线x上的上升沿触发 (中断和事件)

注: 外部唤醒线是边沿触发的, 这些线上不能出现毛刺信号。

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

### 8.3.4 下降沿触发选择寄存 (EXTI\_FTSR)

偏移地址: 0x0C

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
保留												TR20	TR19	TR18	TR17	TR16
												rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

位31: 21	保留。
位20: 0	<b>TR<sub>x</sub>:</b> 线x上的下降沿触发事件配置位 (Falling trigger event configuration bit of line x) 1 = 允许输入线x上的下降沿触发 (中断和事件) 0 = 禁止输入线x上的下降沿触发 (中断和事件)

注: 外部唤醒线是边沿触发的, 这些线上不能出现毛刺信号。

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

### 8.3.5 软件中断事件寄存器 (EXTI\_SWIER)

偏移地址: 0x10

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
保留														SW IER20	SW IER19	SW IER18	SW IER17	SW IER16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
SW IER15	SW IER14	SW IER13	SW IER12	SW IER11	SW IER10	SW IER9	SW IER8	SW IER7	SW IER6	SW IER5	SW IER4	SW IER3	SW IER2	SW IER1	SW IER0			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

位31: 21	保留。
位20: 0	<b>SWIERx:</b> 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_INTMASK和EXTI_EVNTMASK中允许产生该中断，则此时将产生一个中断。 注：通过清除EXTI_PEND的对应位（写入'1'），可以清除该位为'0'。

### 8.3.6 软件中断事件寄存 (EXTI\_PR)

偏移地址: 0x14

复位值: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
保留														PR20	PR19	PR18	PR17	PR16
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																		

位31: 21	保留。
位20: 0	<b>PRx:</b> 挂起位 (Pending bit) 1 = 发生了选择的触发请求 0 = 没有发生触发请求 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。

## 9. 模拟/数字转换 (ADC)

### 9.1 ADC 介绍

12 位 ADC 是逐次逼近式的模拟—数字转换器 (SAR A/D 转换器)。控制器设计 10 个通道，本芯片只引出 6 个通道。

A/D 转换器支持多种工作模式：单次转换和连续转换模式，并且可以选择通道自动扫描。A/D 转换的启动方式有软件设定、外部引脚触发以及各个定时器启动。

窗口比较器（模拟看门狗）允许应用程序检测输入电压是否超出了用户设定的高/低阈值值。

ADC 的输入时钟不得超过 15MHz，它是由 PCLK2 经分频产生。

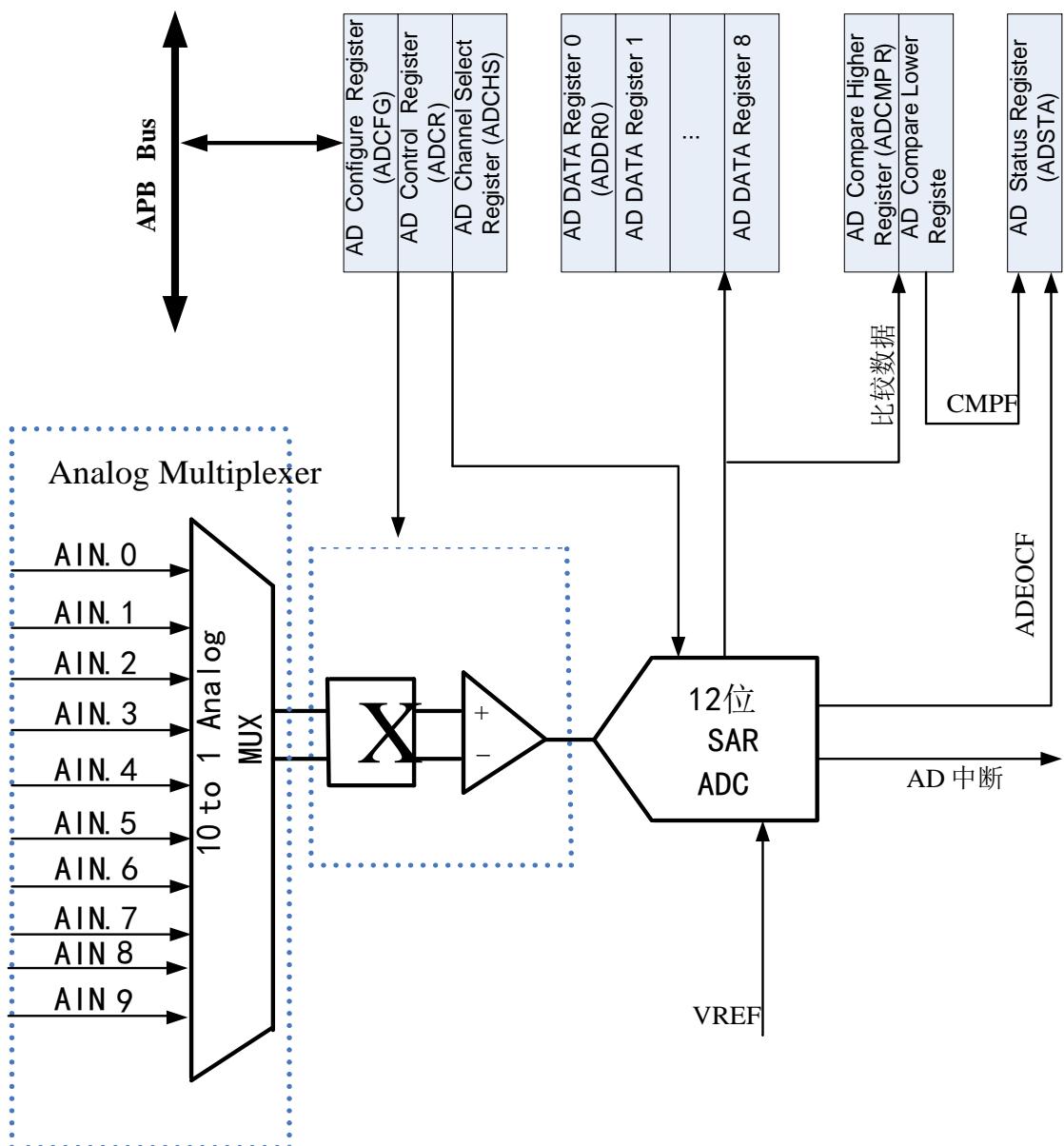
### 9.2 ADC 主要特征

- 12 位 SAR ADC，多达 6 路外部输入通道。
- 高达 1Msps 转换速率；
- 支持多种工作模式
  - 单次转换模式：A/D 转换在指定通道完成一次转换
  - 单周期扫描模式：A/D 转换在所有指定通道完成一个周期（从低序号通道到高序号通道）转换
  - 连续扫描模式：A/D 转换连续执行单周期扫描模式直到软件停止 A/D 转换
- 支持 DMA 传输
- A/D 转换开始条件
  - 软件启动
  - 外部触发启动
  - Timer1/2/3/4 匹配或者 TRGO 信号
- 模拟看门狗，转换结果可和指定的值相比较，当转换值和设定值相匹配时，用户可设定是否产生中断请求；
- 模拟 TouchPad，外接电阻触摸屏器件，可做为电阻触摸屏的控制电路。

### 9.3 ADC 功能描述

下图显示了 AD 框图

图 6. AD 框图



### 9.3.1 ADC 开关控制

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

ADC 上电延迟一段时间后(tSTAB)，设置 ADST 位开始进行转换。

通过清除 ADST 位可以停止转换，设置 ADEN 位可置于断电模式。

### 9.3.2 ADC 时钟

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

### 9.3.3 通道选择

有 6 路外部输入通道。

每个外部输入通道都有独立的使能位，可通过设置 ADCHS 寄存器的对应位来设置。

## 9.4 ADC 工作模式

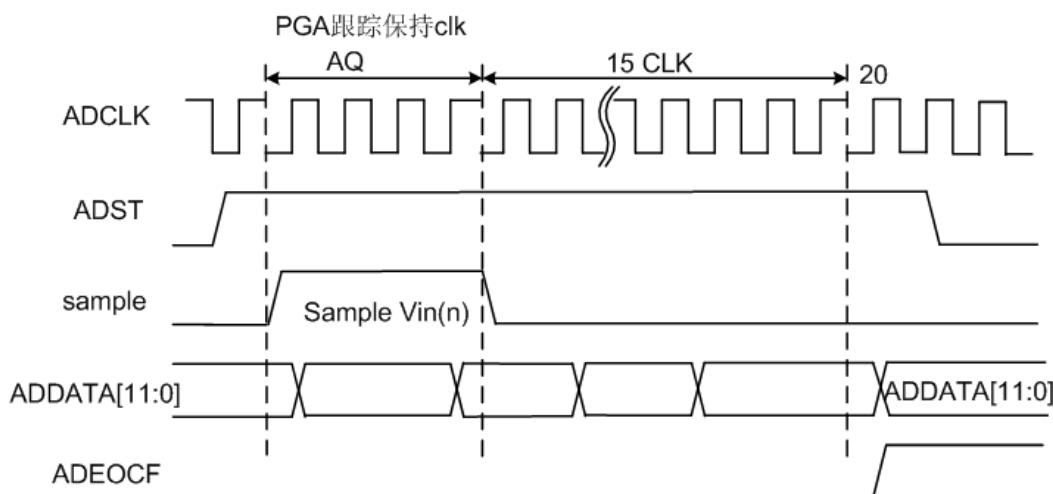
### 9.4.1 单次转换模式

在单次转换模式下，A/D 转换相应通道上只执行一次，具体流程如下：

- 通过软件、外部触发输入及定时器溢出置位 ADST，CONT=0，开始 A/D 转换。
- 当 A/D 转换完成，A/D 转换的数据值将存储于 A/D 的数据寄存器 ADDATA 和 ADDRn 中。
- A/D 转换完成，状态寄存器 ADSTA 的 ADIF 位置 1。若此时控制寄存器 ADCRL 的 ADIE 位置 1，将产生 AD 转换结束中断请求。
- A/D 转换期间，ADST 位保持为 1。A/D 转换结束，ADST 位自动清 0，A/D 转换器进入空闲模式。

注：在单次转换模式下，如果软件使能多个通道，序号最小的通道被转换，其他通道被忽略。

图 7. 单次转换模式时序图



### 9.4.2 单周期扫描模式

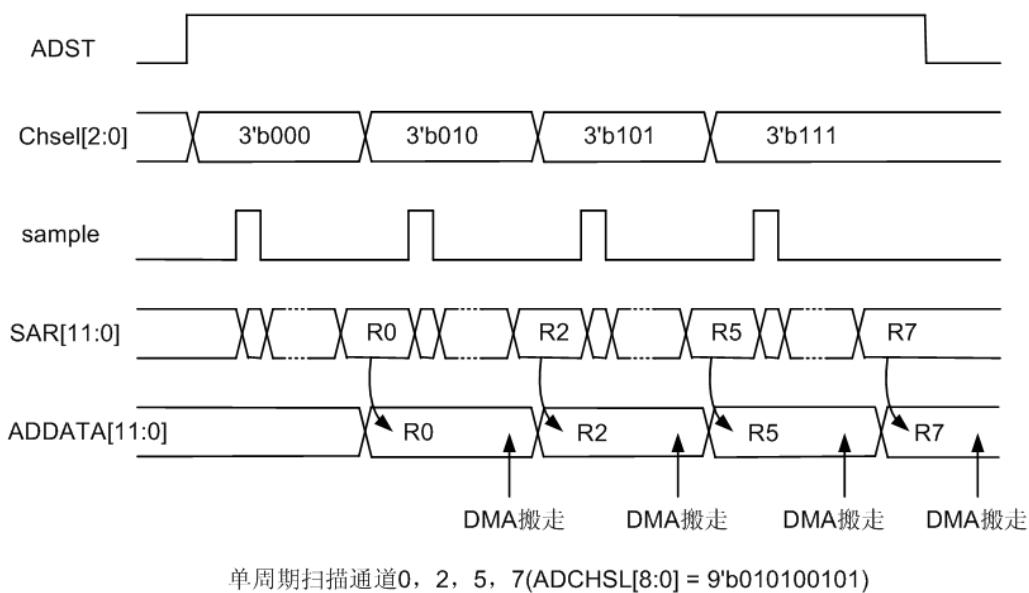
在单周期扫描模式下，将进行一次从被使能的最小序号通道向最大序号通道的 A/D 转换，操作步骤如下：

软件或外部触发置位 ADST 开始，从最小序号通道到最大序号通道的 A/D 转换。

每路 A/D 转换完成后，A/D 转换数值将有序装载到相应通道的数据寄存器中，ADIF 转换结束标志被设置，如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。

转换结束后，ADST 位自动清 0，A/D 转换器进入空闲状态。

图 8. 单周期扫描下使能通道转换时序图



#### 9.4.3 连续扫描模式

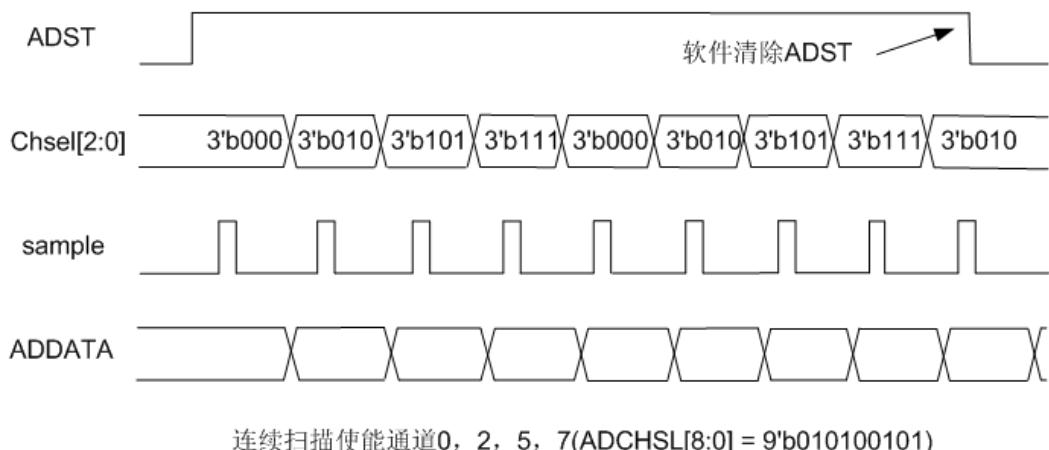
在连续扫描模式下，A/D 转换在 ADCHS 寄存器中的 CHENn 位被使能的通道上顺序进行，操作步骤如下：

软件或外部触发置位 ADST 开始，从最小序号通道到最大序号通道的 A/D 转换。

当所有通道的 A/D 转换完成一遍后，A/D 转换数值将有序装载到相应的数据寄存器中，ADIF 转换结束标志被设置，如果设置了转换结束中断，则在所有通道转换都完成后产生中断请求。

只要 ADST 位保持为 1，就重复步骤 2 到 3。当 ADST 位被清 0，A/D 转换停止，A/D 转换器进入空闲状态。当 ADST 清 0，A/D 转换将完成当前转换。

图 9. 连续扫描模式使能通道转换时序图



#### 9.4.4 DMA 请求

单周期扫描和连续扫描时通道转换的值存储在各自通道的数据寄存器（ADDRn）中，最近一次转换的结果也会保存在 ADDATA 寄存器中。DMA 传输时可以选择传输某个特定通道的数据，或者传输所有扫描通道的结果。

#### 9.4.5 采样频率设置

ADC 的时钟 ADCLK 由 PCLK2 分频得到, 分频系数可通过设置 ADCFG 寄存器的 ADCPRE 位来确定, 即 PCLK2 (n+1) 分频后作为 ADC 时钟。ADC 工作时, 每 15 ADCLK 周期采样一次, 即采样频率为  $F_{\text{sample}}=F_{\text{ADCLK}}/15$ 。

#### 9.5 数据对齐

ADCR 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐, 如图 10 所示。

图 10. 数据对齐方式

数据右对齐

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

数据左对齐

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

#### 9.6 外部触发转换

ADC 转换可以由外部事件触发(例如定时器捕获, EXTI 线)。如果设置了 ADCFG 寄存器的 TRGEN 位, 就可以使用外部事件触发转换。通过设置 TRGSEL 位可以选择外部触发源。

具体的外部触发源选择情况, 可以参考 AD 控制寄存器(ADCR) bit[6: 4] TRGSEL bits 的描述。

#### 9.7 窗口比较器模式下 AD 转换结果监控

比较模式下提供了上限和下限两个比较寄存器。可通过软件设定 CMPCH 位选择监控通道。

当 CPMHDATA $\geq$ CPMLDATA 时, 比较结果大于或等于 ADCMPR 寄存器的 CMPHDATA 指定值 或者 小于 CMPLDATA 指定值, 状态寄存器 ADSTA 的 ADWIF 位置 1。

当 CPMHDATA<CPMLDATA 时, 比较结果大于或等于 CMPHDATA 指定值且小于 CMPLDATA 指定值, 状态寄存器 ADSTA 的 ADWIF 位置 1。

如果控制寄存器 ADCR 的 ADWIE 置位, 将产生 ADINT 中断请求。

#### 9.8 触摸屏模式下 AD 转换结果监控

触摸屏模式下提供了 X 轴和 Y 轴两个数据寄存器。可通过软件设定 TPEN 位选择开启触摸屏模式。当 AD 转换结果小于 ADCTPCR 中的 TPCMP 指定值并且次数达到 TPCNT 指定值 N+1 后, 状态寄存器 ADSTA 的 ADTPIF 位置 1。

如果控制寄存器 ADCR 的 ADTPIE 置位, 将产生 ADINT 中断请求。

ADC 启用触摸屏模式时将启动连续扫描模式。

## 9.9 ADC 寄存器描述

### 9.9.1 A/D 数据寄存器(ADC\_ADDATA)

地址偏移: 0x0

复位值: 0x0000 0000

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

位31: 22	保留。必须保持为0。
位21	<b>VALID:</b> 有效标志位(只读) (Valid flag) 1 = DATA[11: 0]位数据有效。 0 = DATA[11: 0]位数据无效。 相应模拟通道转换完成后, 将该位置位, 读ADDATA寄存器后, 该位由硬件清除.
位20	<b>OVERRUN:</b> 数据覆盖标志位(只读) (Overrun flag) 1 = DATA [11: 0]数据被覆盖。 0 = DATA [11: 0]数据 最近一次转换结果. 新的转换结果装载至寄存器之前, 若DATA[11: 0] 的数据没有被读取, OVERRUN将置1。读ADDATA寄存器后, 该位由硬件清除.
位19: 16	<b>CHANNELSEL:</b> 该4位显示当前数据所对应的通道 (Channel selection) 0000 = 通道0的转换数据 0001 = 通道1的转换数据 0010 = 通道2的转换数据 0011 = 通道3的转换数据 0100 = 通道4的转换数据 0101 = 通道5的转换数据 0110 = 通道6的转换数据 0111 = 通道7的转换数据 1000 = 通道8的转换数据 1001 = 通道9的转换数据 其他: 无效
位15: 0	<b>DATA:</b> 12位A/D转换结果 (Transfer data) 根据设置左对齐或者右对齐。

### 9.9.2 A/D 配置寄存器(ADC\_ADCFG )

地址偏移: 0x04

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留	ADC_ANA_PD[3:0]				ACD_ANA_PU[3:0]				ADC_PRE[7:3]				ADC_MODSEL		
	rw				rw				rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	ADC_SAMCTL [2:0]				ADC_RSLTCTL [2:0]				ADCPRE				保留	ADWEN	ADEN

rw rw rw rw

位31: 30	保留。必须保持为0。
位29: 26	<b>ADC_ANA_PD:</b> 触摸模式下, xp, yp, xn, yn为0时, ADC_ANA_PD[3:0]控制端口portB[3:0]下拉
位25: 22	<b>ACD_ANA_PU:</b> 触摸模式下, xp, yp, xn, yn为0时, ADC_ANA_PU[3:0]控制端口portB[3:0]上拉
位21: 17	<b>ADCPRE:</b> ADC预分频高5位 (ADC prescaler) 由软件置“1”或清“0”来确定ADC时钟频率 n: PCLK2 2 <sup>n+1</sup> 分频后作为ADC时钟
位16	<b>ADC_MODSEL:</b> 1 = 差分输入 0 = 单端输入
位15: 13	保留, 必须保持为0
位12: 10	<b>ADC_SAMCTL[2:0]:</b> 可编程采样时间 000 = 1.5周期 100 = 41.5周期 001 = 7.5周期 101 = 55.5周期 010 = 13.5周期 110 = 71.5周期 011 = 28.5周期 111 = 239.5周期
位9: 7	<b>ADC_RSLTCTL[2:0]:</b> 可编程分辨率 000 = 8位有效 001 = 9位有效 010 = 10位有效 011 = 11位有效 100 = 12位有效 其他无效
位6: 4	<b>ADCPRE:</b> ADC预分频 (ADC prescaler) 由软件置“1”或清“0”来确定ADC时钟频率 n: PCLK2 2 <sup>n+1</sup> 分频后作为ADC时钟
位3: 2	保留
位1	<b>ADWEN:</b> A/D窗口比较器使能 (ADC window comparison enable) 1 = A/D窗口比较器使能 0 = A/D窗口比较器禁用
位0	<b>ADEN:</b> A/D转换使能 (ADC enable) 1 = 使能 0 = 禁用

### 9.9.3 AD 控制寄存器(ADC\_ADCR)

地址偏移: 0x08

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPCH	ALIGN	ADM	ADS T	保留	TRGSEL	DMA EN	TRG EN	ADWI E	ADIE						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						

位31: 16	保留。必须保持为0。
---------	------------

位15: 12	<b>CMPCH:</b> 窗口比较通道选择 (Window comparison channel selection) 0000 = 选择比较通道0转换结果 0001 = 选择比较通道1转换结果 0010 = 选择比较通道2转换结果 0011 = 选择比较通道3转换结果 0100 = 选择比较通道4转换结果 0101 = 选择比较通道5转换结果 0110 = 选择比较通道6转换结果 0111 = 选择比较通道7转换结果 1000 = 选择比较通道8转换结果 1001 = 选择比较通道9转换结果 1111 = 所有扫描通道 其他: 无效
位11	<b>ALIGN:</b> 数据对齐 (Data alignment) 0: 右对齐 1: 左对齐
位10: 9	<b>ADMD:</b> A/D转换模式 (ADC mode) 00: 单次转换 01: 单周期扫描 10: 连续扫描 当改变转换模式时, 软件要先禁用ADST位。
位8	<b>ADST:</b> A/D转换开始 (ADC start) 1 = 转换开始. 0 = 转换结束或进入空闲状态. ADST置位有下列两种方式: 在单次模式或者单周期模式下, 转换完成后, ADST将被硬件自动清除, 在连续扫描模式下, A/D转换将一直进行, 直到软件写0到该位或系统复位.
位7	保留
位6: 4	<b>TRGSEL:</b> 外部触发源选择 (External trigger selection) 000: TIM1_CC1 001: TIM1_CC2 010: TIM1_CC3 011: TIM2_CC2 100: TIM3_TRGO 101: TIM4_CC4 110: TIM3_CC1 111: EXTI线11
位3	<b>DMAEN:</b> DMA使能 (Direct memory access enable) 1 = DMA请求使能 0 = DMA禁止
位2	<b>TRGEN:</b> 外部硬件触发源 (External trigger enable) 1 = 使用外部触发信号启动A/D转换 0 = 不用外部触发信号启动A/D转换
位1	<b>ADWIE:</b> A/D窗口比较器中断使能 (ADC window comparator interrupt enable) 1 = 使能A/D窗口比较器中断 0 = 禁用A/D窗口比较器中断
位0	<b>ADIE:</b> A/D中断使能 (ADC interrupt enable) 1 = 使能A/D中断 0 = 禁用A/D中断 如果ADINT置位, A/D转换结束后产生中断请求.

#### 9.9.4 AD 通道选择寄存器(ADC\_ADCHS)

地址偏移: 0x0C

复位值: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CHE N9	CHE N8	CHE N7	CHE N6	CHE N5	CHE N4	CHE N3	CHE N2	CHE N1	CHE N0					
	rw	rw	rw	rw	rw	rw									

位31: 9	保留。必须保持为0。
位9	<b>CHEN9:</b> 模拟输入通道9使能 (Analog input channel 9 enable) 1 = 使能 0 = 禁用
位8	<b>CHEN8:</b> 模拟输入通道8使能 (Analog input channel 8 enable) 1 = 使能 0 = 禁用
位7	<b>CHEN7:</b> 模拟输入通道7使能 (Analog input channel 7 enable) 1 = 使能 0 = 禁用
位6	<b>CHEN6:</b> 模拟输入通道6使能 (Analog input channel 6 enable) 1 = 使能 0 = 禁用
位5	<b>CHEN5:</b> 模拟输入通道5使能 (Analog input channel 5 enable) 1 = 使能 0 = 禁用
位4	<b>CHEN4:</b> 模拟输入通道4使能 (Analog input channel 4 enable) 1 = 使能 0 = 禁用
位3	<b>CHEN3:</b> 模拟输入通道3使能 (Analog input channel 3 enable) 1 = 使能 0 = 禁用
位2	<b>CHEN2:</b> 模拟输入通道2使能 (Analog input channel 2 enable) 1 = 使能 0 = 禁用
位1	<b>CHEN1:</b> 模拟输入通道1使能 (Analog input channel 1 enable) 1 = 使能 0 = 禁用
位0	<b>CHENO:</b> 模拟输入通道0使能 (Analog input channel 0 enable) 1 = 使能 0 = 禁用

注：如果通道使能都为0，则通道0使能。

### 9.9.5 A/D 窗口比较寄存器(ADC\_ADCMPR)

地址偏移: 0x10

复位值: 0x0000 0000

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

位31: 9	保留
--------	----

位27: 16	<b>CMPHDATA:</b> 比较数值上限 (Compare data high limit) 该12位数值将和指定通道的转换结果相比较。
位15: 12	保留
位11: 0	<b>CMPLDATA:</b> 比较数值下限 (Compare data low limit) 该12位数值将和指定通道的转换结果相比较。

### 9.9.6 A/D 状态寄存器(ADC\_ADSTA)

地址偏移: 0x10

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALID[8: 0]															
CHANNEL[3: 0]															
保留															
BUSY															
位2															
1 = A/D转换器忙碌 0 = A/D转换器空闲															
位1															
<b>ADWIF:</b> 比较标志位 (ADC window comparator interrupt flag) 选择的A/D转换通道 结果大于等于ADCMR或小于ADCMPLR, 该位置1。写1清该位。															
位0															
<b>ADIF:</b> A/D转换结束标志位 (ADC interrupt flag) 该位由硬件在通道组转换结束时设置, 由软件清除 1 = A/D转换完成 0 = A/D转换未完成 该标志位写1清零.															

### 9.9.7 A/D 数据寄存器(ADC\_ADDR0~9)

地址偏移: 0x18~0x40

复位值: 0x0000 0000

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

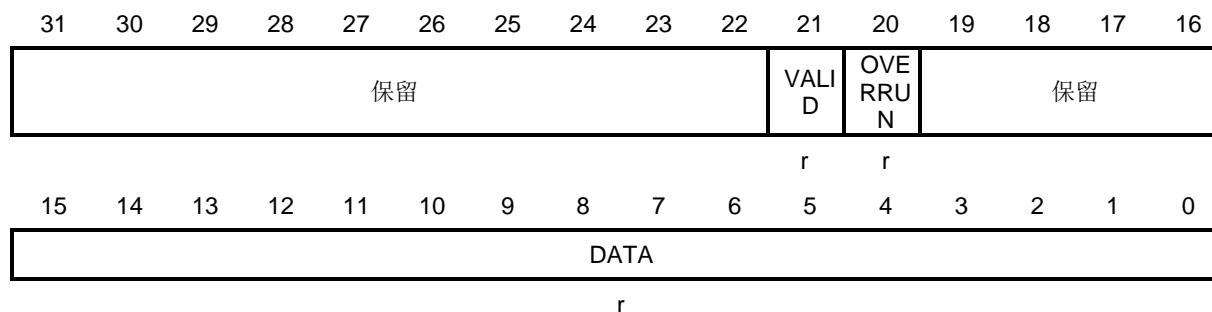
---

位31: 22	保留
位21	<b>VALID:</b> 有效标志位(只读) (Valid flag) 1 = DATA[11: 0]位数据有效. 0 = DATA[11: 0]位数据无效. 相应模拟通道转换完成后, 将该位置位, 读ADDATA寄存器后, 该位由硬件清除.
位20	<b>OVERRUN:</b> 数据覆盖标志位(只读) (Overrun flag) 1 = DATA [11: 0]数据被覆盖. 0 = DATA [11: 0]数据最近一次转换结果. 新的转换结果装载至寄存器之前, 若DATA[11: 0] 的数据没有被读取, OVERRUN将置1。读ADDATA寄存器后, 该位由硬件清除.
位19: 16	保留
位15: 0	<b>DATA:</b> 通道0~9的12位A/D转换结果 (Transfer data) 根据设置左对齐或者右对齐。

### 9.9.8 A/D 触摸屏 X+数据寄存器(ADC\_TPXDR)

地址偏移: 0x48

复位值: 0x0000 0000

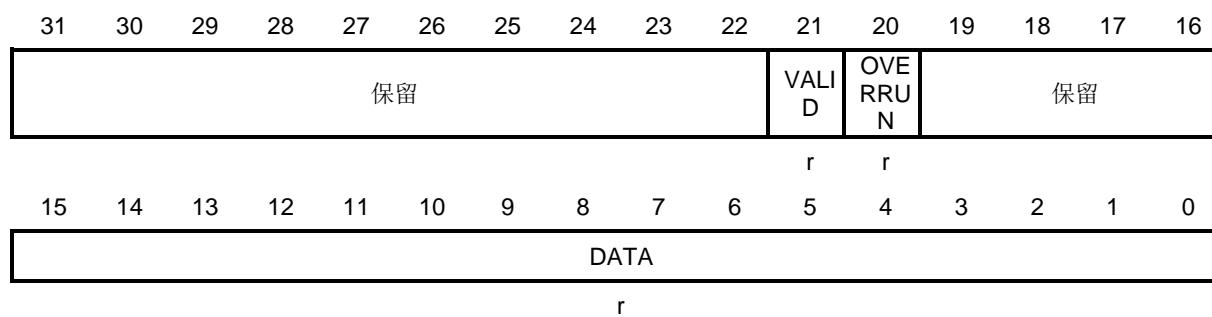


位31: 22	保留
位21	<b>valid_x:</b> 有效标志位(只读) (Valid flag) 1 = DATA[11: 0]位数据有效. 0 = DATA[11: 0]位数据无效. 相应模拟通道转换完成后, 将该位置位, 读寄存器后, 该位由硬件清除.
位20	<b>overrun_x:</b> 数据覆盖标志位(只读) (Overrun flag) 1 = DATA [11: 0]数据被覆盖. 0 = DATA [11: 0]数据最近一次转换结果. 新的转换结果装载至寄存器之前, 若DATA[11: 0] 的数据没有被读取, OVERRUN将置1。读寄存器后, 该位由硬件清除.
位19: 16	保留
位15: 0	<b>data_x:</b> 通道0~9的12位A/D转换结果 (Transfer data) 根据设置左对齐或者右对齐。

### 9.9.9 A/D 触摸屏 Y+数据寄存器(ADC\_YPDR)

地址偏移: 0x4c

复位值: 0x0000 0000



位31: 22	保留
位21	<b>valid_y:</b> 有效标志位(只读) (Valid flag) 1 = DATA[11: 0]位数据有效. 0 = DATA[11: 0]位数据无效. 相应模拟通道转换完成后, 将该位置位, 读寄存器后, 该位由硬件清除.

位20	<b>overrun_y:</b> 数据覆盖标志位(只读) (Overrun flag) 1 = DATA [11: 0]数据被覆盖。 0 = DATA [11: 0]数据最近一次转换结果。 新的转换结果装载至寄存器之前, 若DATA[11: 0] 的数据没有被读取, OVERRUN将置1。读寄存器后, 该位由硬件清除。
位19: 16	保留
位15: 0	<b>data_y:</b> 通道0~9的12位A/D转换结果 (Transfer data) 根据设置左对齐或者右对齐。

### 9.9.10 A/D 触摸屏控制寄存器(ADC\_TPCR)

地址偏移: 0x50

复位值: 0x0000 0000

位16	<b>adtp_if:</b> 比较标志位 (ADC TouchPad interrupt flag) 选择的A/D TouchPad转换通道 结果小于ADTPCMP且连续有效, 该位置1。写1清该位。
位1	<b>adtp_ie:</b> A/D触摸屏中断使能 (ADC touchpad interrupt enable) 1 = 使能A/D触摸屏中断 0 = 禁用A/D触摸屏中断
位0	<b>adtp_en:</b> 触摸屏使能控制位 (TouchPad mode enable) 1 = 触摸屏模式使能 0 = 触摸屏模式禁止

### 9.9.11 A/D 触摸屏滤波寄存器(ADC\_TPFR)

地址偏移: 0x54

复位值: 0x00ff ffff

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
TPCNT[3:0]				TPCMP[11:0]											

r

位31: 28	保留
位27: 24	<b>adtp_cnt:</b> 触摸屏模式下连续采用次数n+1次。
位23: 12	<b>adtp_cmy:</b> 触摸屏模式下Y方向有效阀值。当AD转换结果小于该阀值时为有效采样值。
位11: 0	<b>adtp_cmplx:</b> 触摸屏模式下X方向有效阀值。当AD转换结果小于该阀值时为有效采样值。

### 9.9.12 A/D 触摸屏通道选择寄存器(ADC\_TPCSR)

地址偏移: 0x58

复位值: 0x0000 0010

位31: 8	保留
位7: 4	<b>adtp_chy:</b> 触摸屏模式下Y方向采样使用的AD通道
位3: 0	<b>adtp_chx:</b> 触摸屏模式下X方向采样使用的AD通道

## 9.10 IF

### 9.10.1 Hardware IF

signal	
AHB bus	.....
ADC signals	.....
xp_oe	控制屏的X+端口 1: 向X+对应的IO口输出1 0: 将X+对应的IO口设置为模拟输入模式
xn_oe	控制屏的X-端口 1: 向X-对应的IO口输出0 0: 将X-对应的IO口设置为floating模式
yp_oe	控制屏的Y+端口 1: 向Y+对应的IO口输出1 0: 将Y+对应的IO口设置为模拟输入模式
yn_oe	控制屏的Y-端口 1: 向Y-对应的IO口输出0 0: 将Y-对应的IO口设置为floating模式
touch_on	IO mux时控制IO与gpx_p、gpx_n、gpy_p、gpy_n的连接 1: AD正在采集屏对应的AD通道数据 0: AD未采集屏对应的AD通道数据

### 9.10.2 Software information

基本与 AD 的连续扫描模式一致，区别在于：

- 1、模式的配置不通过配置寄存器 ADC\_ADCR 的 admd 位，而是配置寄存器 ADC\_TPCR 的 adtp\_en 位。
- 2、需要提前配置好滤波寄存器 ADC\_TPFR（也可以保持默认值）。
- 3、屏对应通道通过 ADC\_TPCSR 配置，同时需要在 ADC\_ADCHS 中打开相应通道（默认 Y 方向的采样使用 channel1,X 方向采样使用 channel0）。
- 4、采样结果通过寄存器 ADC\_TPCR 确认，并通过 ADC\_TPXDR 和 ADC\_TPYDR 获取。

## 10. LCD-TFT 控制器

### 10.1 简介

LTD<sub>C</sub> 提供了 24 位的并行数字 RGB（红、绿、蓝），传送的所有信号可直接与最高 1024x600 分辨率的 LCD 和 TFT 面板接口。

### 10.2 主要特性

- 支持标准水平/垂直同步数字视频格式
- 输出数字视频时序可调
- 输出支持 RGB888 格式，向下兼容 RGB666、RGB565 等
- 在支持 RGB666、RGB565 时，不用于显示的 IO 可以不复用到 LCD 上，可节省 IO 资源
- VGA 输出，支持刷新率不低于 20Hz: 640x480, 800x600

### 10.3 寄存器说明

表 19. 控制器寄存器说明

Offset	Name	Access	Default	Description
00h	DP_ADDR0	RW	00000000h	Bits[31:8]: The 24MSB of the start address of the display image buffer0 in the system memory. The 8LSB will always treated as all zero. Bits[7:0]: Reserved. Always zero.
04h	DP_ADDR1	RW	00000000h	Bits[31:8]: The 24MSB of the start address of the display image buffer1 in the system memory. The 8LSB will always treated as all zero. Bits[7:0]: Reserved. Always zero.
08h	P_HOR	RW	00000697h	Bit[31:16]: Reserved. Bit[15:0]: This value plus 1 define the total width counted in unit of one pixel. For example, 679h mean the output image width is 1688 pixels.
0ch	HSYNC	RW	0030009fh	Bit[31:16]: Horizontal sync start This value define the hsync signal started by horizontal counter Bit[15:0]: Horizontal sync end This value define the hsync signal ended by horizontal counter
10h	A_HOR	RW	01980697h	Bit[31:16]: A_HOR_START This value define the active area ended by horizontal counter Bit[15:0]: A_HOR_END This value define the active area started by horizontal counter. And A_HOR_END must equal to A_HOR_START+A_HOR_LEN
14h	A_HOR_LEN	RW	000004FFh	Bit[31:16]: Reserved Bit[15:0]: This value plus 1 define the total number of horizontal active area in unit of pixel. The LST 2bit are always zero This value plus 1 must multiple of 8pixel
18h	BLK_HOR	RW	00000197h	Bit[15:0] BLK_HOR_START This value define the blanking area started by horizontal counter Bit[31:16]: BLK_HOR_END This value define the blanking area ended by horizontal counter.
1ch	P_VER	RW	00000429h	Bit[15:0]: This value plus 1 define the total height counted in unit of one line. For example, 429h mean the output image height is 1066 lines. Bit[31:16]: Reserved.

Offset	Name	Access	Default	Description
20h	VSYNC	RW	00010003h	Bit[31:16]: Vertical sync start This value define the vsync signal started by vertical counter Bit[15:0]: Vertical sync end This value define the vsync signal ended by vertical counter
24h	A_VER	RW	002a0429h	Bit[31: 16]: A_VER_START This value define the vertical active area ended by vertical counter Bit[15:0]: A_VER_END This value define the vertical active area started by vertical counter
28h	A_VER_LEN	RW	000003ffh	Bit[31:16]: Reserved Bit[15:0]: This value plus 1 define the total number of vertical active area in unit of line
2ch	BLK_VER	RW	00000029h	Bit[31:16]: BLK_VER_START This value define the vertical blanking area ended by vertical counter Bit[15:0]: BLK_VER_END This value define the vertical blanking area started by vertical counter
30h	BLK_DATA	RW	00000000h	Bit[31:24]: Reserved Bit[23:0]: The blank data that will be inserted into the data stream when blank period. This register is also be used to fill the color of the image background.
34h	POL_CTL	RW	00000000h	Bit[31:4]: Reserved Bit[3]: Polarity of the pixel output clock p_out_clk. 1: negative edge sampling. 0: post edge sampling. Bit[2]: data_en polarity 1: low enable 0: high enable Bit[1]:vsync polarity 1: low active 0: high active Bit[0]: hsync polarity 1: low active 0: high active

Offset	Name	Access	Default	Description
38h	OUT_EN	RW	00000000h	<p>Bit[31:9] Reserved</p> <p>Bit[8]: Global enable.</p> <p>1: Enable the controller. When enabled, the controller will regenerate the output frame from the start of the image</p> <p>0: Disable the whole controller, VO can be configured</p> <p>Bit[7:3]: Reserved</p> <p>Bit[2]: data_en output enable.</p> <p>Bit[1]: v_sync output enable.</p> <p>Bit[0]: h_sync output enable.</p> <p>0: disable.</p> <p>1: enable.</p>
3ch	INTR_STA	RO	00000000h	<p>Interrupt status bits</p> <p>Bit[31:6] Reserved</p> <p>Bit[5]: Active_cmd_finish interrupt, assert when axi finish all active cmd. This interrupt only generate when global enable bit is disabled</p> <p>Bit[4:2]: Reserved</p> <p>Bit[1]: frame over int</p> <p>Bit[0]: line buffer Error int</p>
40h	INTR_EN	RW	00000000h	<p>Bit[31:6] Reserved</p> <p>Bit[5]: Acitve_cmd_finish interrupt enable</p> <p>Bit[4:2]: Reserved</p> <p>Bit[1]: frame over int enable</p> <p>Bit[0]: line buffer error int enable , high active</p>
44h	INTR_CLR	WO	--	<p>Interrupt clear register</p> <p>Write 1 to each bit will clear correspond interrupt in INTR_STA</p> <p>Bit[31:6] Reserved</p> <p>Bit[5]: Active_cmd_finish interrupt clear</p> <p>Bit[4:2] : Reserved</p> <p>Bit[1]: frame buffer over int clear</p> <p>Bit[0]: line buffer error int clear</p>
48h	DP_SWT	RW	00000000h	<p>Bit[31:1]: Reserved</p> <p>Bit[0]: Next display frame buffer</p> <p>0: Next frame is buffer0</p> <p>1: Next frame is buffer1</p> <p>When the display buffer finished display the frame in current buffer, it will check this register's next displaying frame buffer, and get the corresponding data.</p>

Offset	Name	Access	Default	Description
4ch	VI_FORMAT	RW	00000000h	video input format 0: RGB888 If you need to use RGB666 or RGB565, you can put the unused bits NC. e.g. RGB565, D18, D17, D16, D9, D8, D2, D1, D0 for NC.

## 10.4 接口定义

### 10.4.1 引脚列表

表 20. 控制器引脚列表

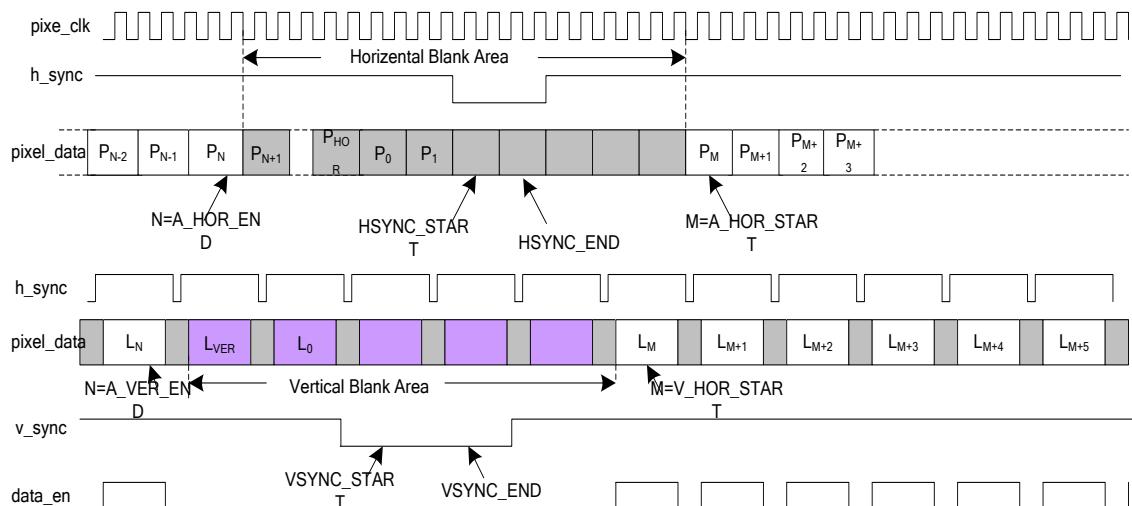
Pin Name	IO	Description
Video Output Interface		
data_r[7:0]	O	Pixel "r" data output
data_g[7:0]	O	Pixel "g" data output
data_b[7:0]	O	Pixel "b" data output
pix_clk_o	O	output pixel clock
hsync	O	Output video horizontal synchronize signal.
vsync	O	Output video vertical synchronize
data_en	O	Output video data_en signal.

### 10.4.2 接口信号说明

#### 视频输出接口

视频输出接口具有独立同步控制信号。下图是波形示例：

图 11. 视频输出波形



所有信号包括 h\_syn, v\_syn, pixel\_data and data\_en 都在 clk\_p 的有效边沿采样。

第一个有效像素的行位置有 A\_HOR\_START 设定。一行中有效像素的数量由寄存器 A\_HOR\_LEN 设定。A\_HOR\_END 必须等于 A\_HOR\_START+A\_HOR\_LEN-1。

一帧中第一个有效行位置由 A\_VER\_START 设定。有效行的总数由寄存器 A\_VER\_LEN 设定。

数字像素值只支持 RGB888 格式。可以把不用的位丢弃以实现向下兼容 RGB666 或者 RGB565 等。

## 10.5 应用说明

### 10.5.1 控制器初始化

为了初始化控制器，寄存器 OUT\_EN[8]先写 0。

下面是初始化步骤的示例：

表 21. 控制器初始化步骤

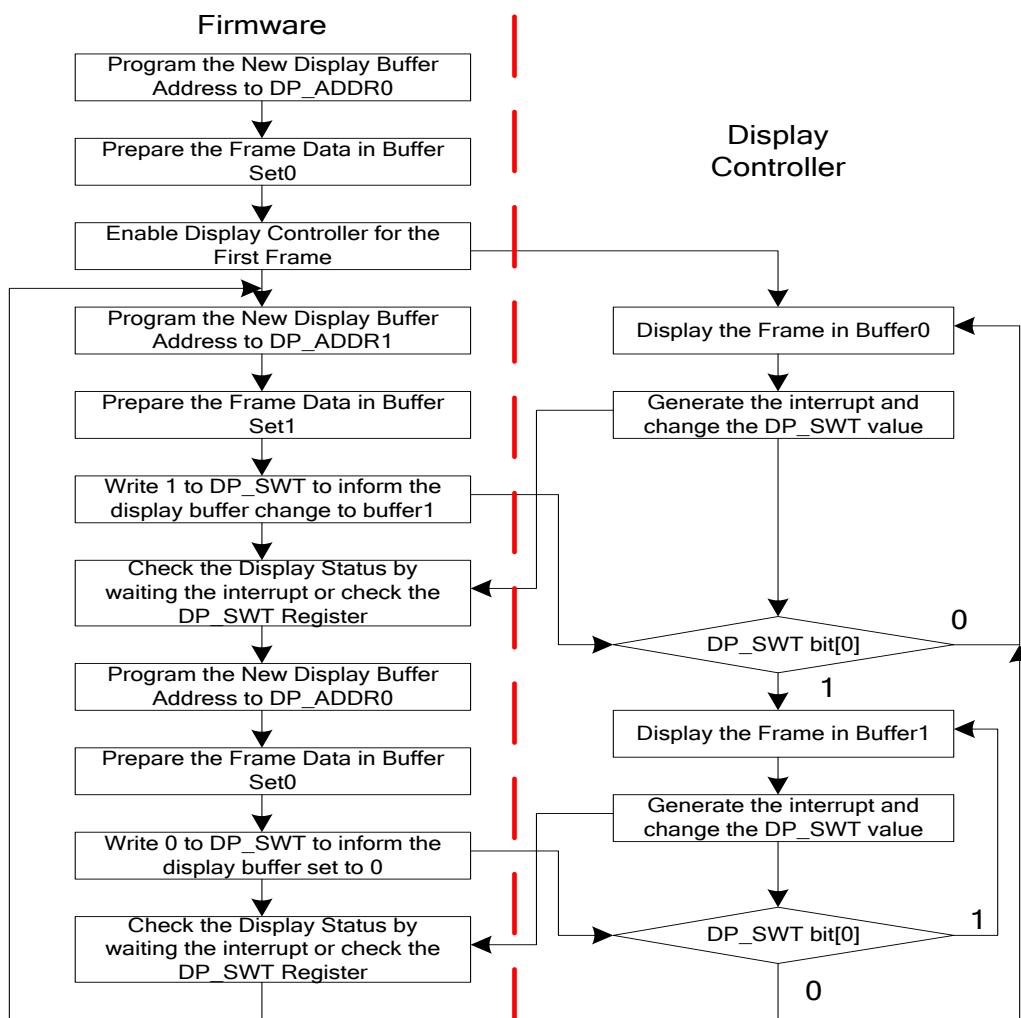
Steps	Description
1	Write OUT_EN with zero.
2	Configure PLL_config register, enable PLL
3	Configure DAC_control register
4	Configure DP_ADDR0, DP_ADDR1
5	Configure register addressed from 08h to 34h in any order
6	Configure INTR_EN register
7	After PLL output stable pixel clock. Video out controller start to work
8	Enable the output by write the corresponding bit with 1 to OUT_EN register.
9	After one frame been displayed, VO will check DP_SWT bit[0] and get the correspond DP_ADDR to display. Before VO get the DP_ADDR , CPU or GPU can change DP_ADDR any time.

### 10.5.2 显示层的交替使用

控制器支持交替使用 2 个显示层来增强刷新率低时视频输出的稳定性。同时有助于交替使用视频帧序列的基地址。

交替使用显示层需要在控制器和软件之间互相交互，下图说明交互流程：

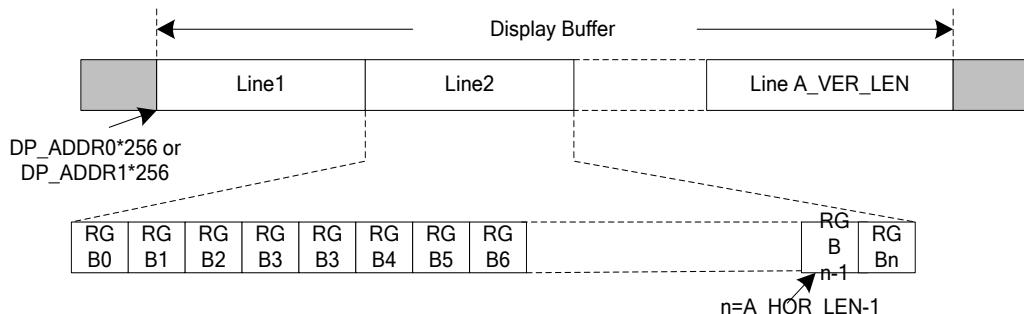
图 12. 显示层的切换控制



### 10.5.3 从显示缓存读取数据

显示缓存的地址由 DP\_ADD0/1 设定。显示缓存存储方式如下图所示：

图 13. 显示缓存存储方式



### 10.5.4 显示缓存 Endian mode

LTD<sub>C</sub> 显示层支持 little-endian 和 big-endian(3type)：

**图 14. RGB 565 Display Buffer Storage**

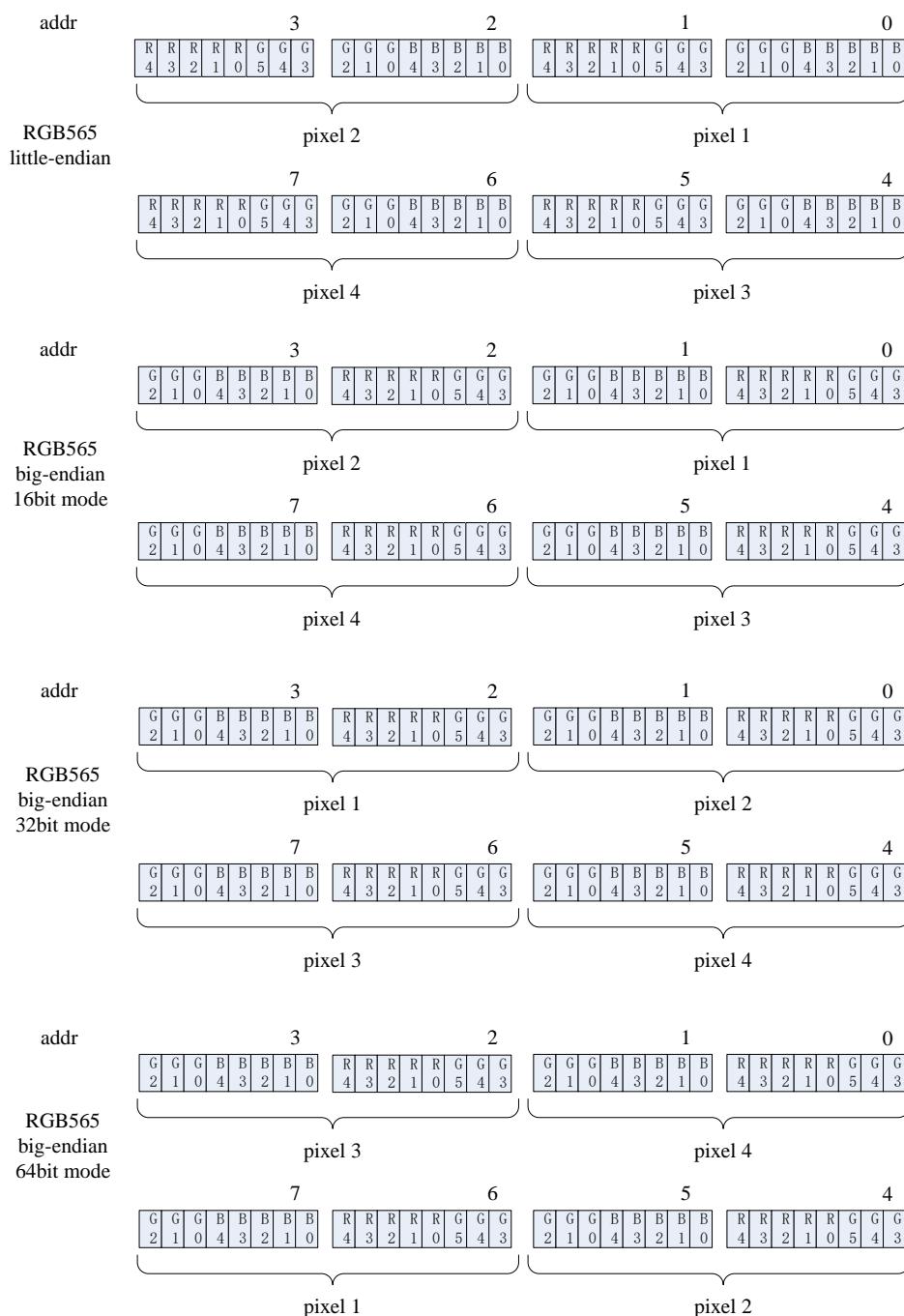
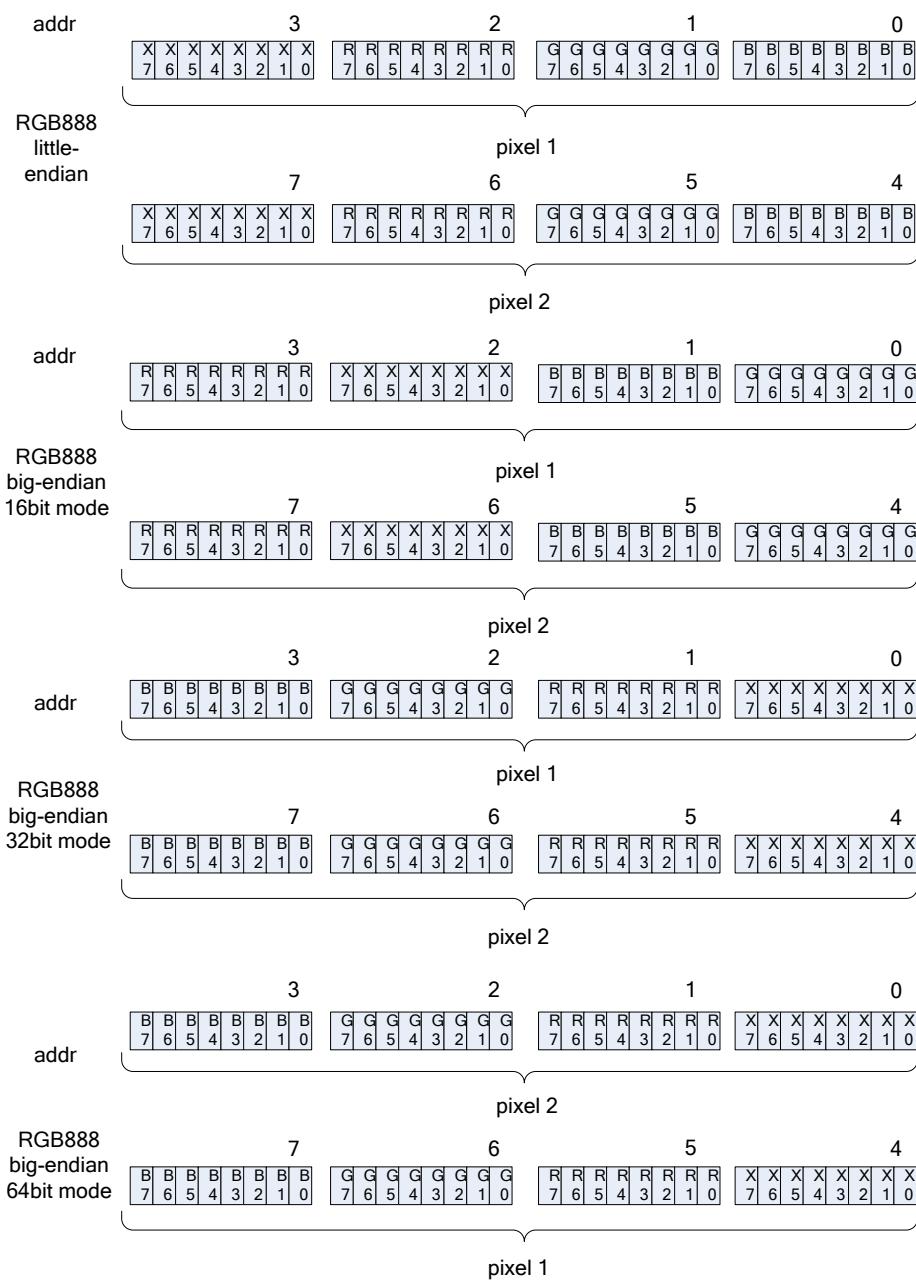


图 15. RGB 888 Display Buffer Storage



### 10.5.5 VESA 时序

VESA 标准分辨率频率:

表 22. Resolution horizontal value

Resolution	p_hor	sync start	sync end	active start	active end	active len	blk start	blk end
800x600	41f h	28 h	a7 h	100 h	41f h	31f h	0 h	ff h
640x480	31f h	8 h	67 h	98 h	317 h	27f h	0 h	8f h

表 23. Resolution vertical configure value

Resolution	pver	sync start	sync end	active start	active end	active len	blk start	blk end
800x600	273 h	1 h	4 h	1c h	273 h	257 h	0 h	1b h
640x480	20c h	2 h	3 h	25 h	204 h	1df h	0 h	1c h

Notice: 1. 800X480 is not the standard VESA STD, so its values are estimated.

## 11. 高级控制定时器 (TIM1/2)

### 11.1 TIM1 和 TIM2 简介

高级控制定时器 (TIM1 和 TIM2) 由一个 32 位的自动装载计数器组成，它由一个可编程的预分频器驱动。

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

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

高级控制定时器 (TIM1 和 TIM2) 和通用定时器 (TIMx) 是完全独立的，它们不共享任何资源。它们可以同步操作，具体描述参看通用定时器同步的章节。

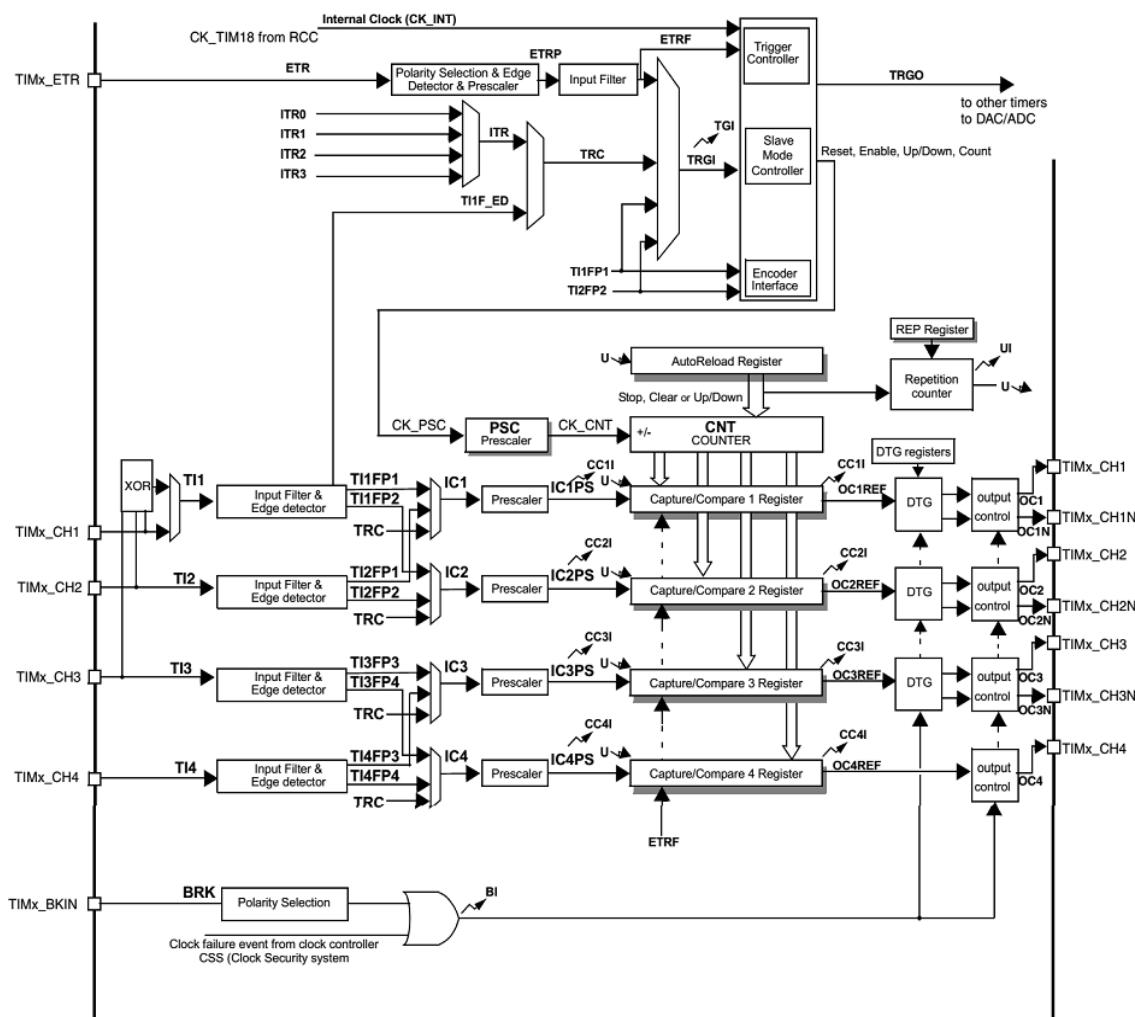
### 11.2 主要特征

TIM1 和 TIM2 定时器的功能包括:

- 32 位向上、向下、向上/下自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 多达 4 个独立通道:
  - 输入捕获
  - 输出比较
  - PWM 生成（边缘或中间对齐模式）
  - 单脉冲模式输出
- 死区时间可编程的互补输出

- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA:
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 16. 高级控制定时器框图



注： 根据控制位的设定，在 U 事件时传送预加载寄存器的内容至工作寄存器

事件

中断和 DMA 输出

## 11.3 功能描述

### 11.3.1 时基单元

可编程高级控制定时器的主要部分是一个 32 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

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

时基单元包含：

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

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

计数器由预分频器的时钟输出 **CK\_CNT** 驱动，仅当设置了计数器 **TIMx\_CR1** 寄存器中的计数器使能位 (**CEN**) 时，**CK\_CNT** 才有效。（更多有关使能计数器的细节，请参见控制器的从模式描述）。

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

### 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个 (**TIMx\_PSC** 寄存器中的) 16 位寄存器控制的 32 位计数器。因为这个控制寄存器带有缓冲器，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下面两个图分别给出了在预分频器运行时，更改计数器参数的例子。

图 17. 当预分频器的参数从 1 变到 2 时，计数器的时序图

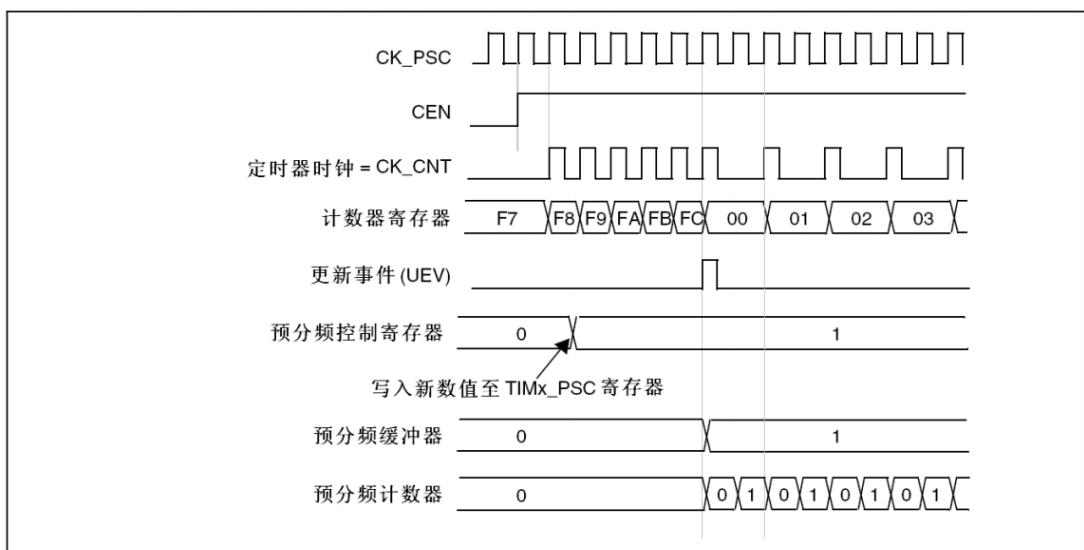
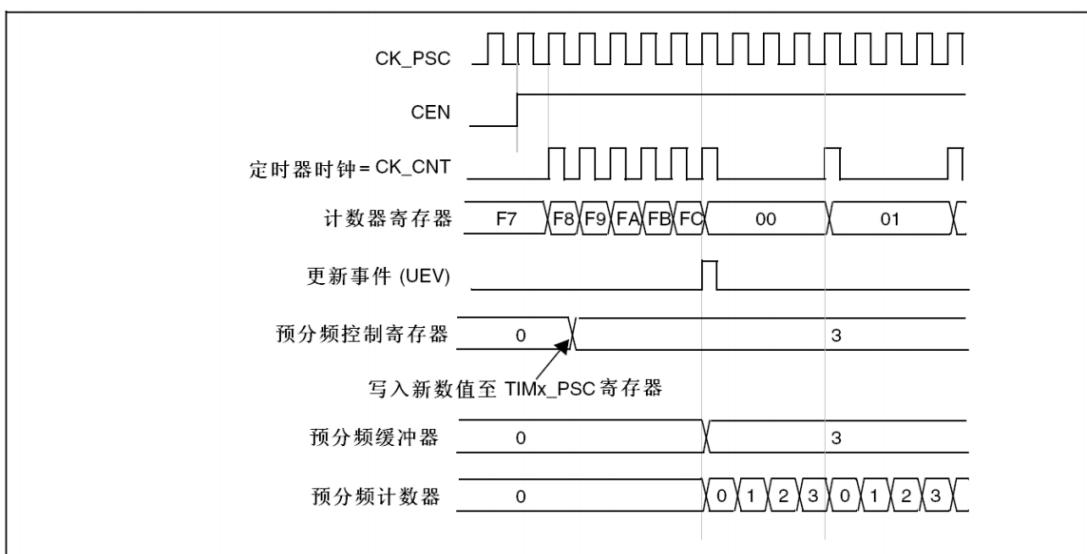


图 18. 当预分频器的参数从 1 变到 4 时，计数器的时序图



### 11.3.2 计数模式

#### 向上计数模式

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

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数（**TIMx\_RCR**）时，产生更新事件（**UEV**）；否则每次计数器溢出时才产生更新事件。

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

设置 **TIMx\_CR1** 寄存器中的 **UDIS** 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 **UDIS** 位被清 0 之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 **TIMx\_CR1** 寄存器中的 **URS** 位（选择更新请求），设置 **UG** 位将产生一个更新事件 **UEV**，但硬件不设置 **UIF** 标志（即不产生中断或 **DMA** 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

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

- 重复计数器被重新加载为 **TIMx\_RCR** 寄存器的内容。
- 自动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 19. 计数器时序图, 内部时钟分频因子为 1

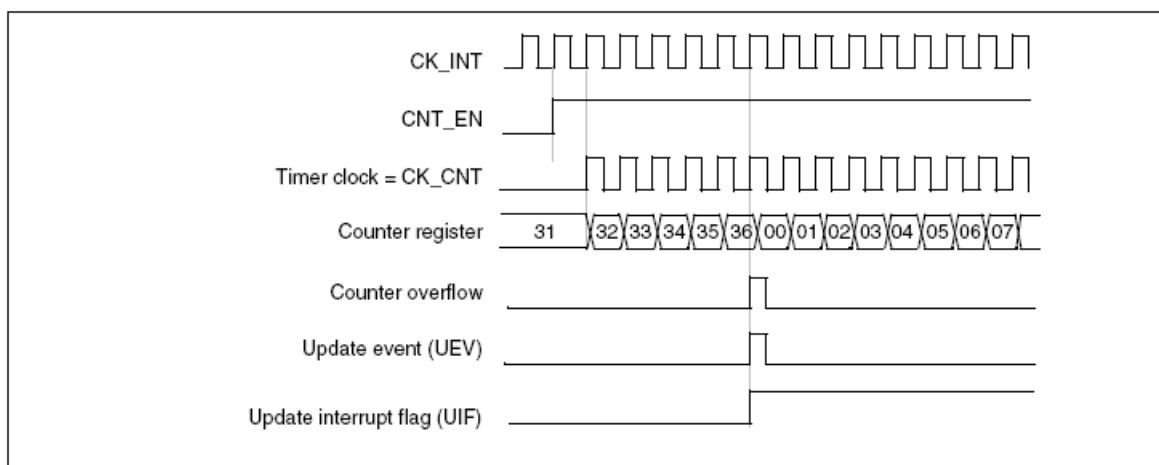


图 20. 计数器时序图, 内部时钟分频因子为 2

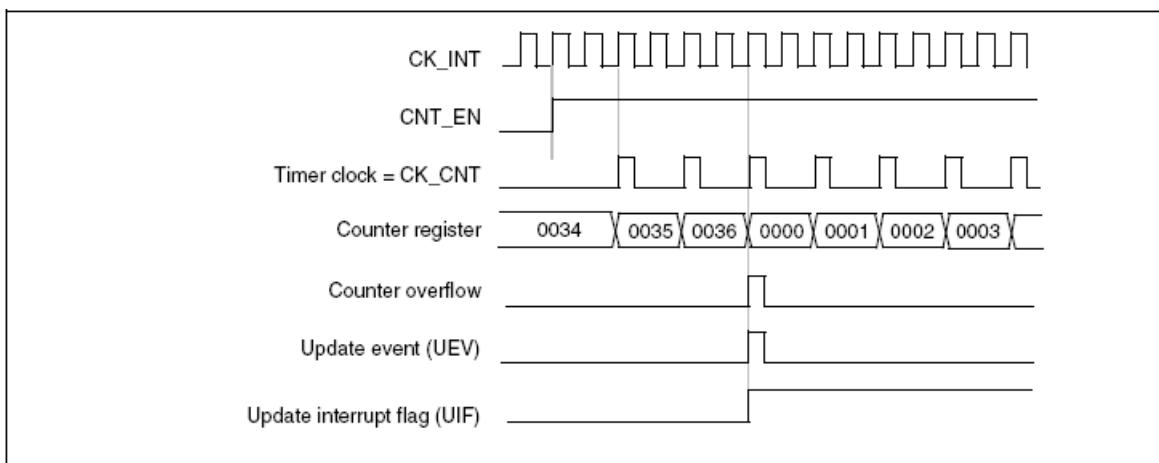


图 21. 计数器时序图, 内部时钟分频因子为 4

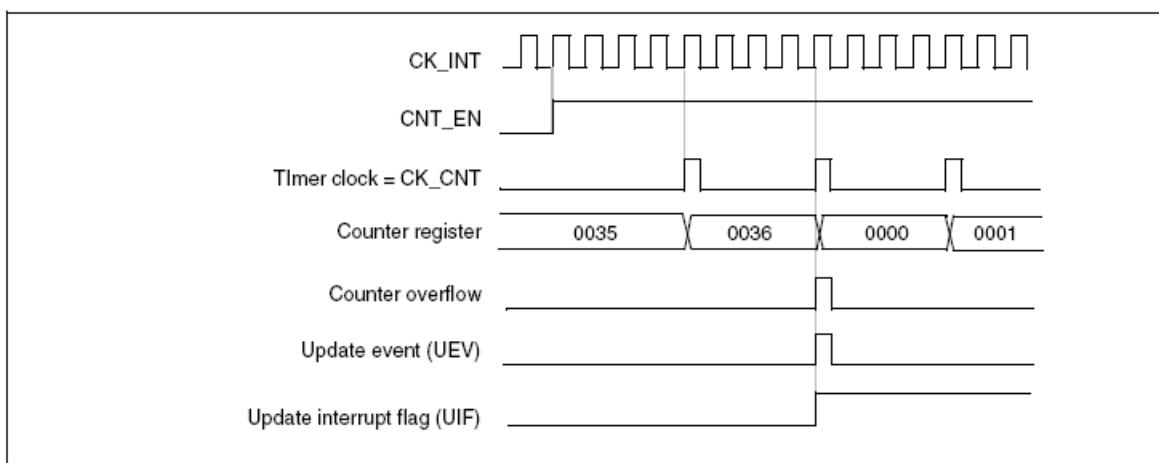


图 22. 计数器时序图，内部时钟分频因子为 N

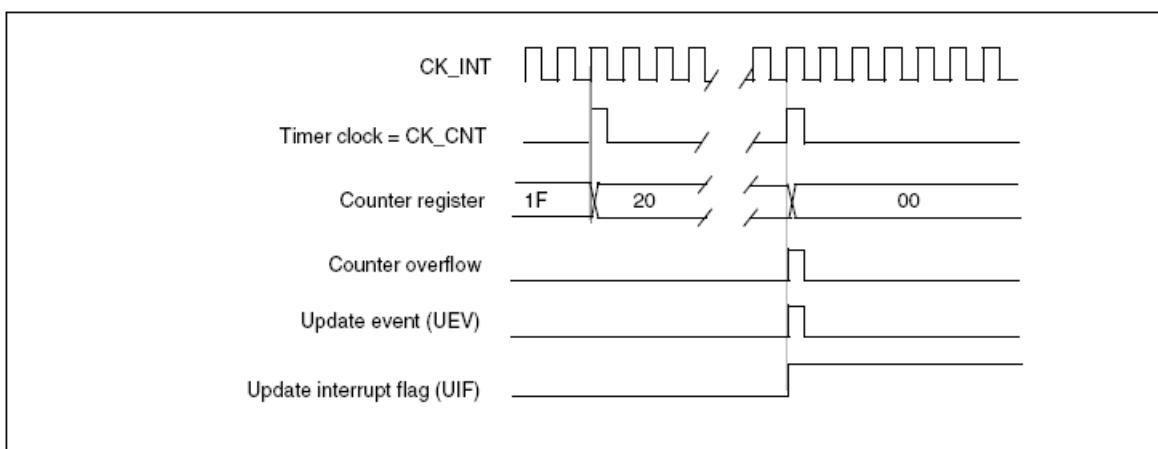


图 23. 计数器时序图，当 ARPE = 0 时的更新事件（TIMx\_ARR 没有预装入）

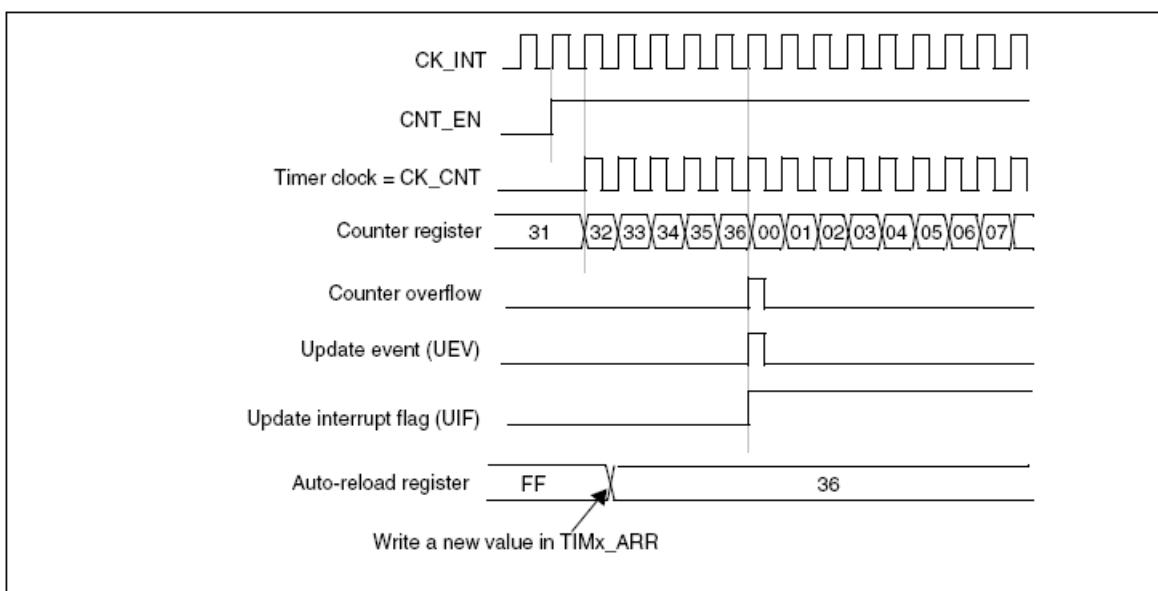
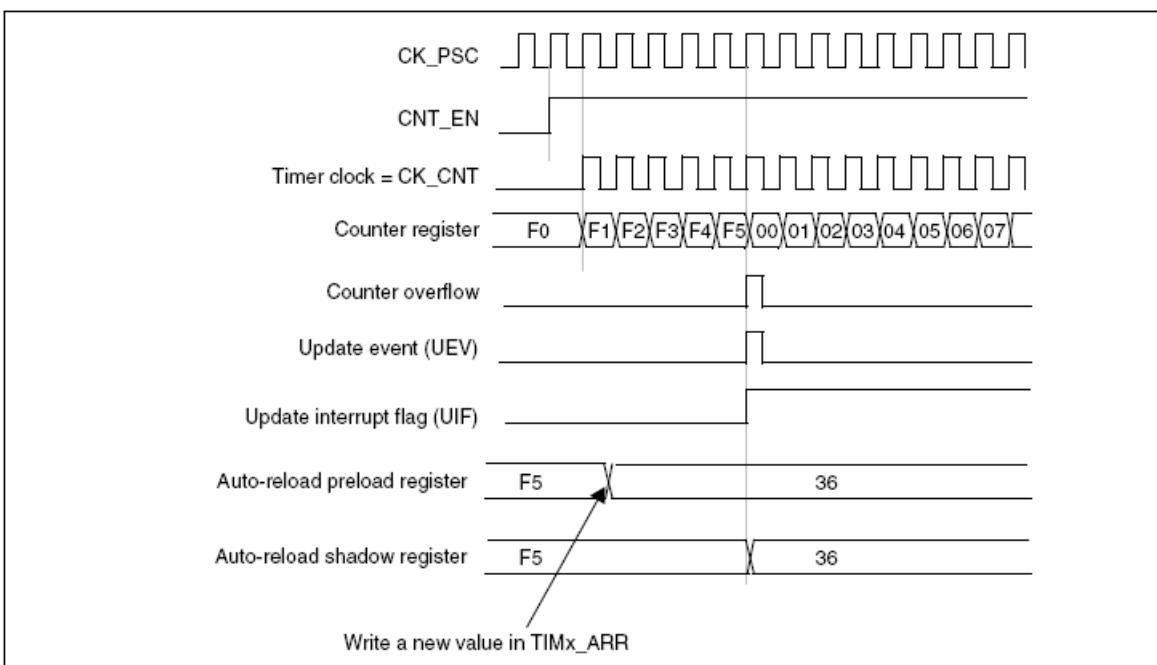


图 24. 计数器时序图, 当 ARPE = 1 时的更新事件 (预装入了 TIMx\_ARR)



### 向下计数模式

在向下模式中, 计数器从自动装入的值 (TIMx\_ARR 寄存器的值) 开始向下计数到 0, 然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

如果使用了重复计数器, 当向下计数重复了重复计数寄存器 (TIMx\_RCR) 中设定的次数后, 将产生更新事件 (UEV), 否则每次计数器下溢时才产生更新事件。

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

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而, 计数器仍会从当前自动加载值重新开始计数, 并且预分频器的计数器重新从 0 开始 (但预分频器的速率不能被修改)。

此外, 如果设置了 TIMx\_CR1 寄存器中的 URS 位 (选择更新请求), 设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志 (因此不产生中断和 DMA 请求), 这是为了避免在发生捕获事件并清除计数器时, 同时产生更新和捕获中断。

当发生更新事件时, 所有的寄存器都被更新, 并且 (根据 URS 位的设置) 更新标志位 (TIMx\_SR 寄存器中的 UIF 位) 也被设置。

- 重复计数器被重置为 TIMx\_RCR 寄存器中的内容。
- 预分频器的缓存器被加载为预装载的值 (TIMx\_PSC 寄存器的值)。
- 当前的自动加载寄存器被更新为预装载值 (TIMx\_ARR 寄存器中的内容)。

*注: 自动装载在计数器重载入之前被更新, 因此下一个周期将是预期的值。*

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

图 25. 计数器时序图, 内部时钟分频因子为 1

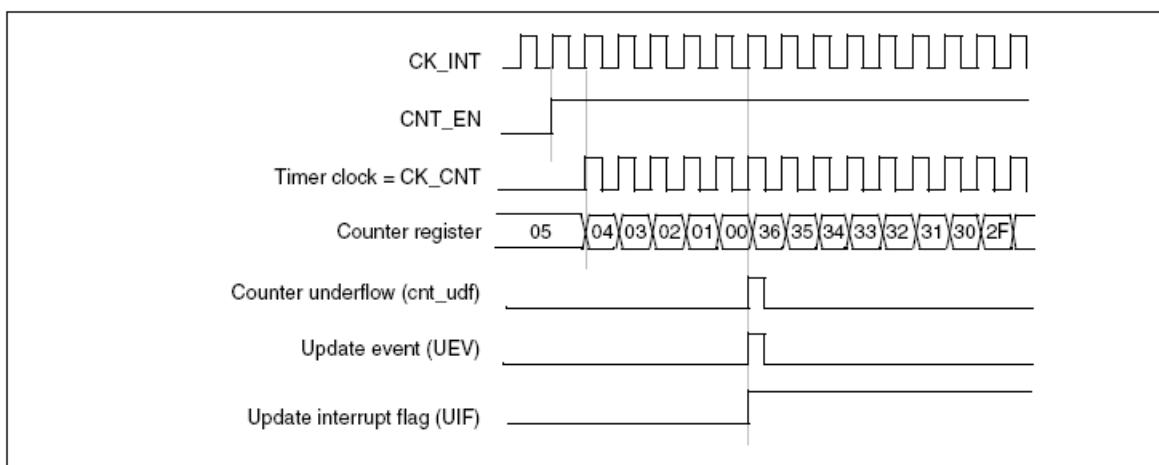


图 26. 计数器时序图, 内部时钟分频因子为 2

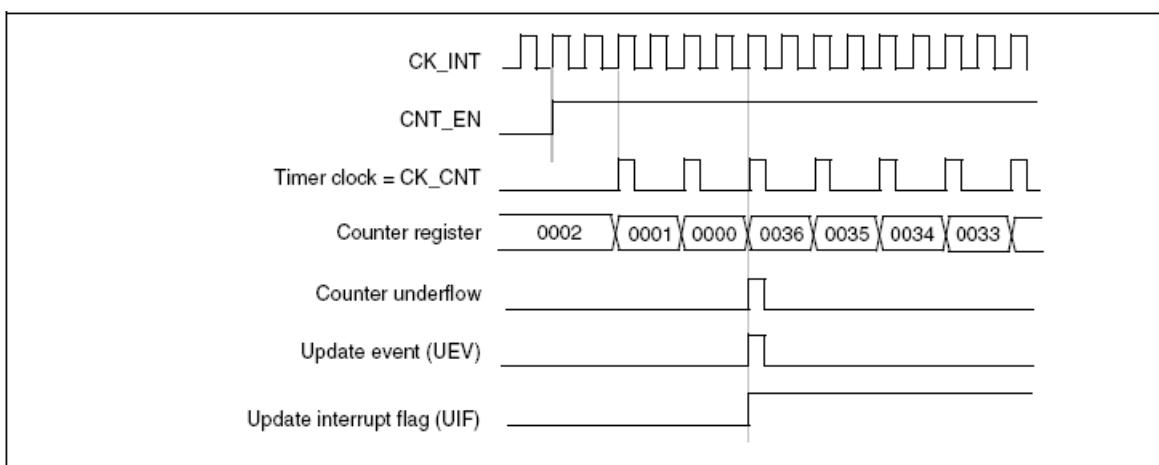


图 27. 计数器时序图, 内部时钟分频因子为 4

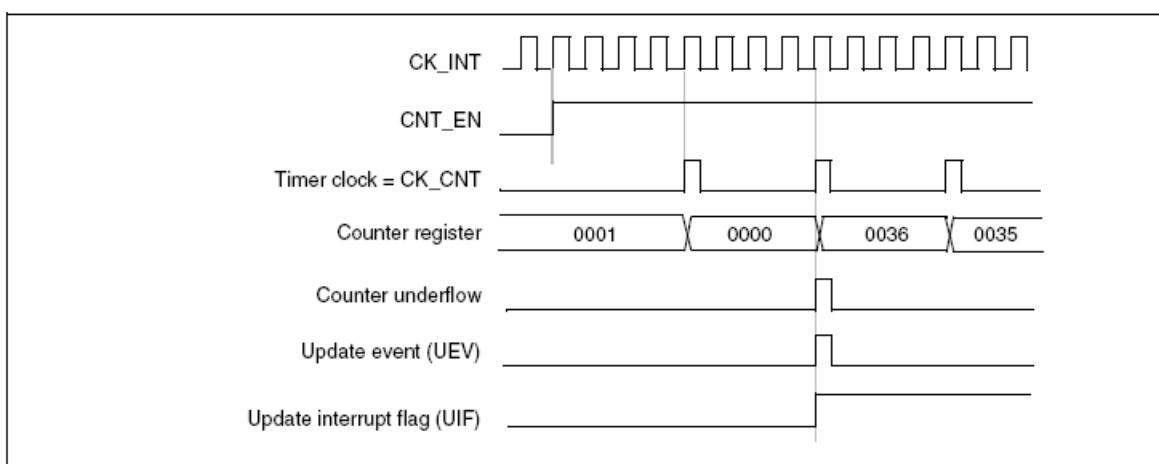


图 28. 计数器时序图，内部时钟分频因子为 N

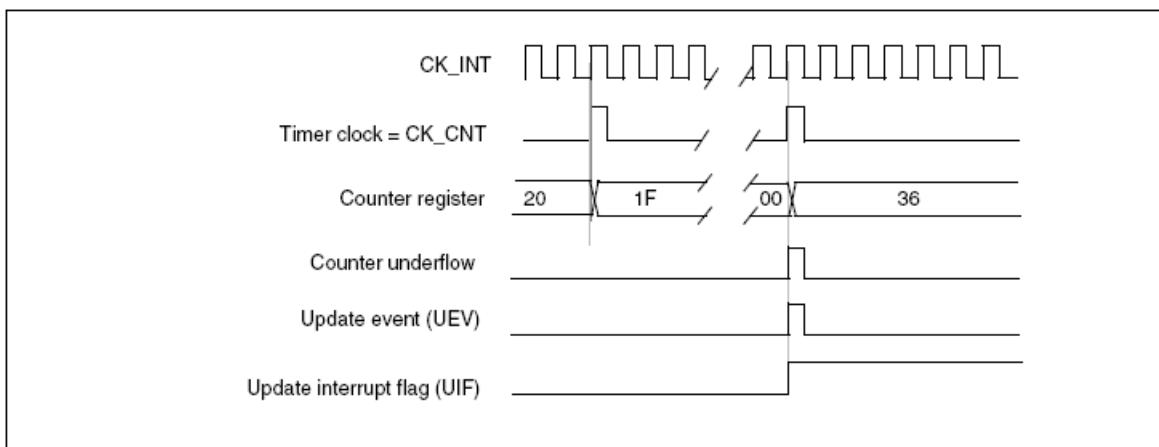
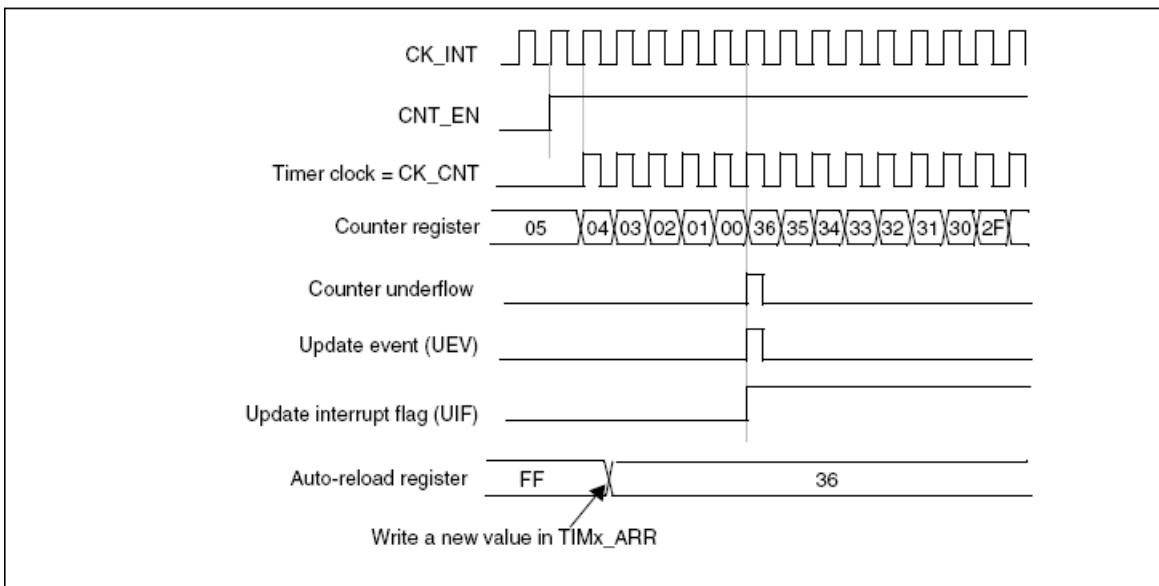


图 29. 计数器时序图，当没有使用重复计数器时的更新事件



### 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值（TIMx\_ARR 寄存器）-1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

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

更新事件可以产生在每次计数上溢和每次计数下溢；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。此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（**TIMx\_SR** 寄存器中的 **UIF** 位）也被设置。

- 重复计数器被重置为 **TIMx\_RCR** 寄存器中的内容。
- 预分频器的缓存器被加载为预装载（**TIMx\_PSC** 寄存器）的值。
- 当前的自动加载寄存器被更新为预装载值（**TIMx\_ARR** 寄存器中的内容）。

**注：**如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。

以下是一些计数器在不同时钟频率下的操作的例子：

图 30. 计数器时序图，内部时钟分频因子为 1，**TIMx\_ARR = 0x6**

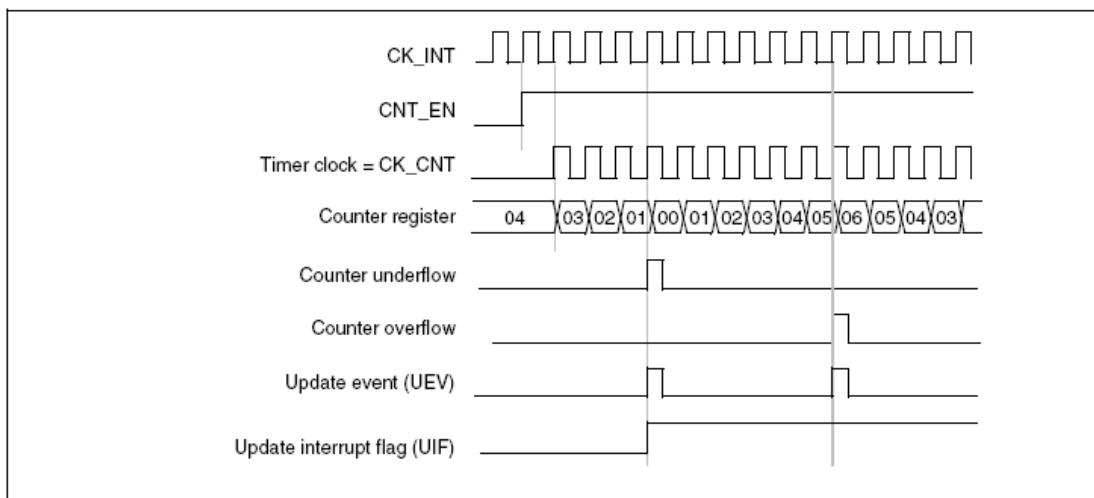


图 31. 计数器时序图，内部时钟分频因子为 2

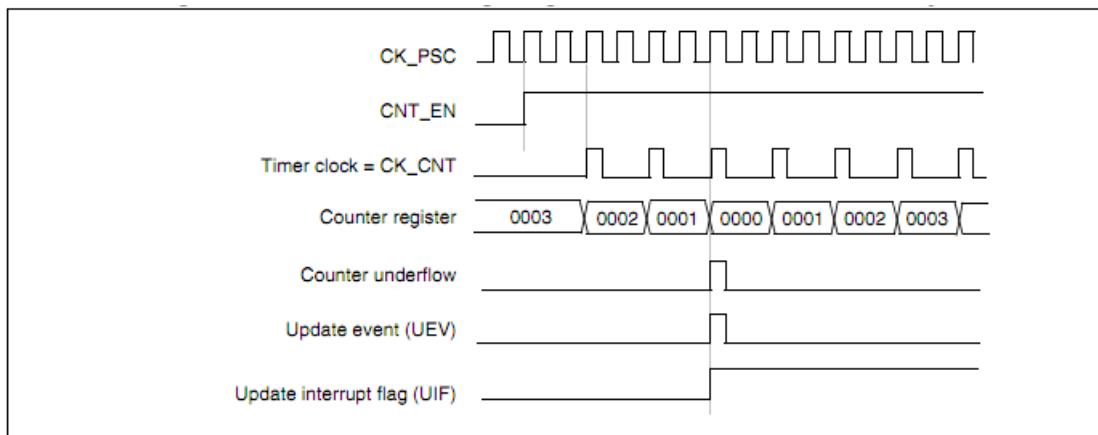


图 32. 计数器时序图, 内部时钟分频因子为 4, TIMx\_ARR = 0x36

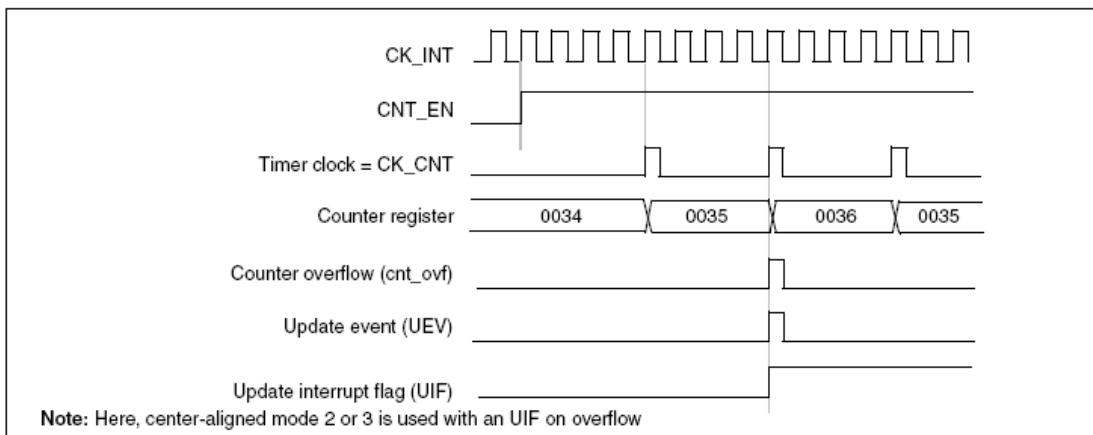


图 33. 计数器时序图, 内部时钟分频因子为 N

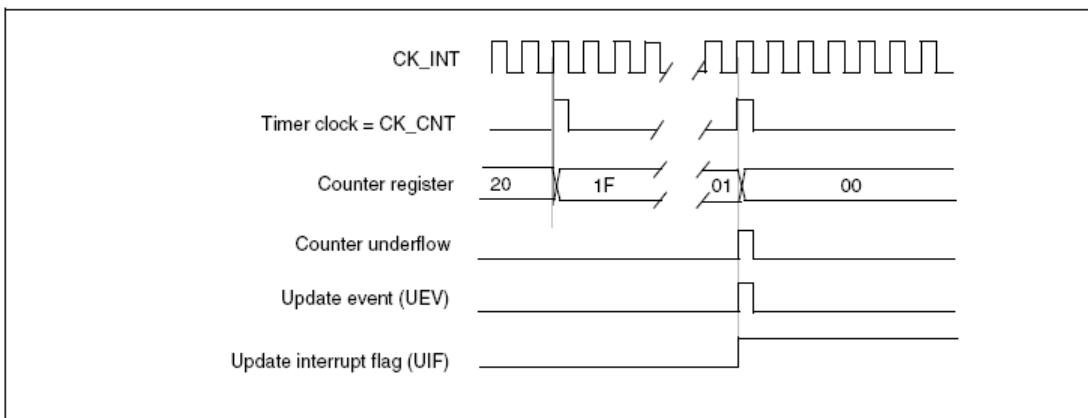


图 34. 计数器时序图, ARPE = 1 时的更新事件（计数器下溢）

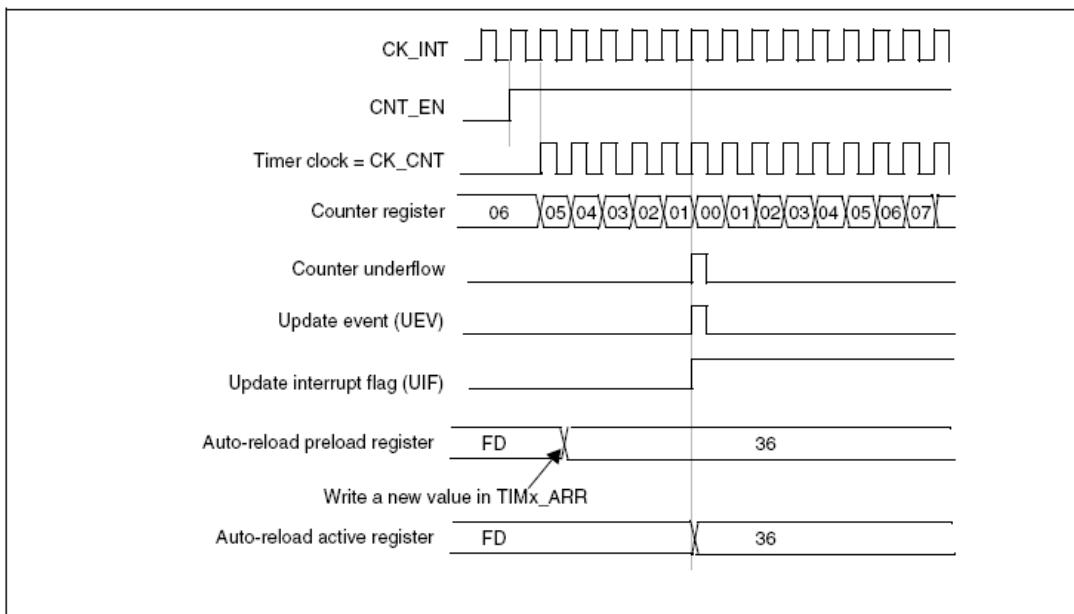
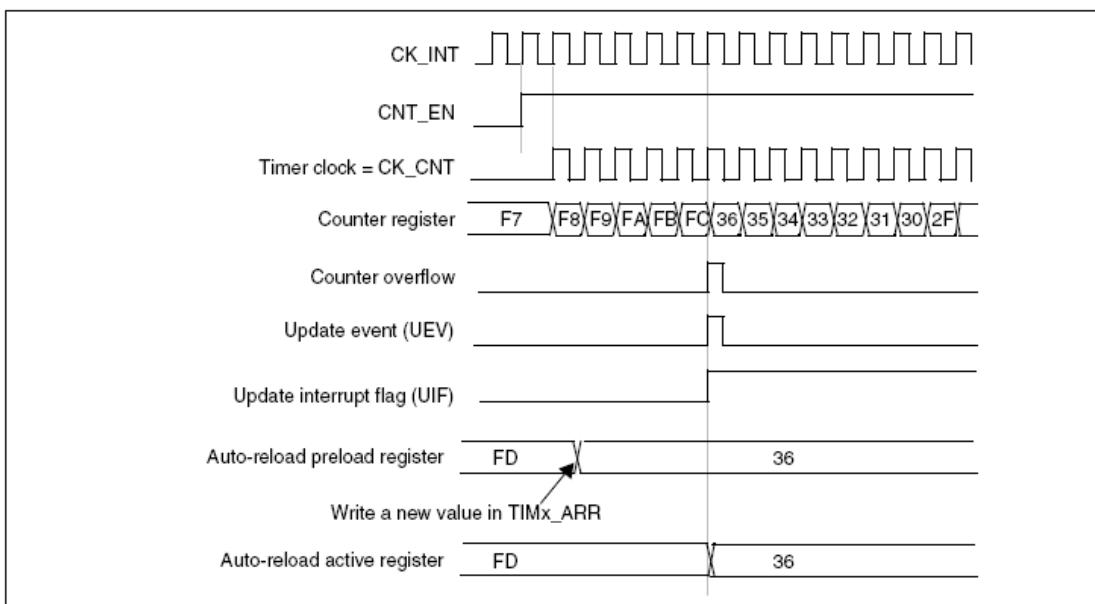


图 35. 计数器时序图, ARPE = 1 时的更新事件 (计数器溢出)



### 11.3.3 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，然而事实上它只能在重复计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

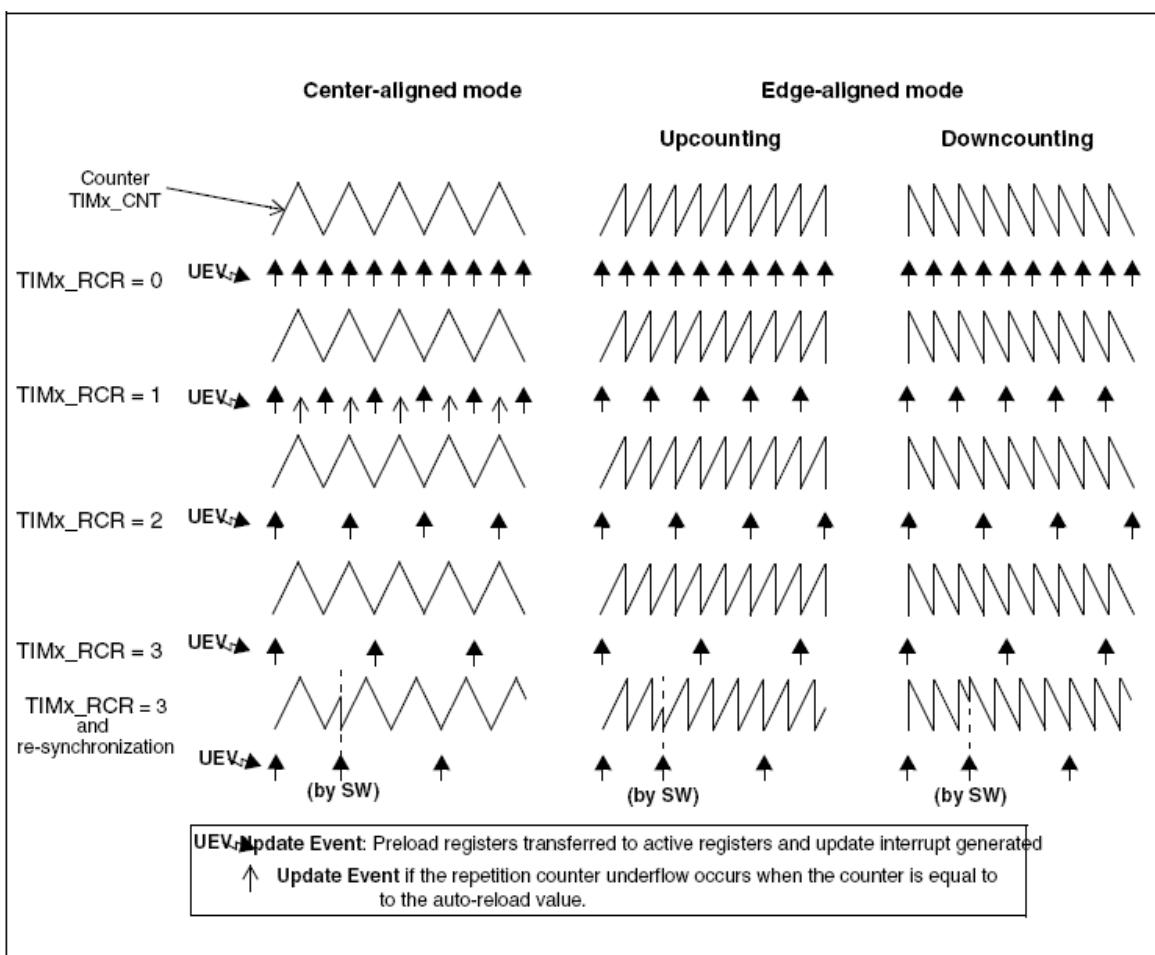
这意味着在每 N 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器 (TIMx\_ARR 自动重载入寄存器，TIMx\_PSC 预装载寄存器，还有在比较模式下的捕获 / 比较寄存器 TIMx\_CCRx)，N 是 TIMx\_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器溢出时，
- 向下计数模式下每次计数器下溢时，
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为  $2 \times T_{ck}$ 。

重复计数器是自动加载的，重复速率是由 TIMx\_RCR 寄存器的值定义（参看图 66）。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 36. 不同模式下更新速率的例子，及 TIMx\_RCR 的寄存器设置



#### 11.3.4 时钟选择

计数器时钟可由下列时钟源提供：

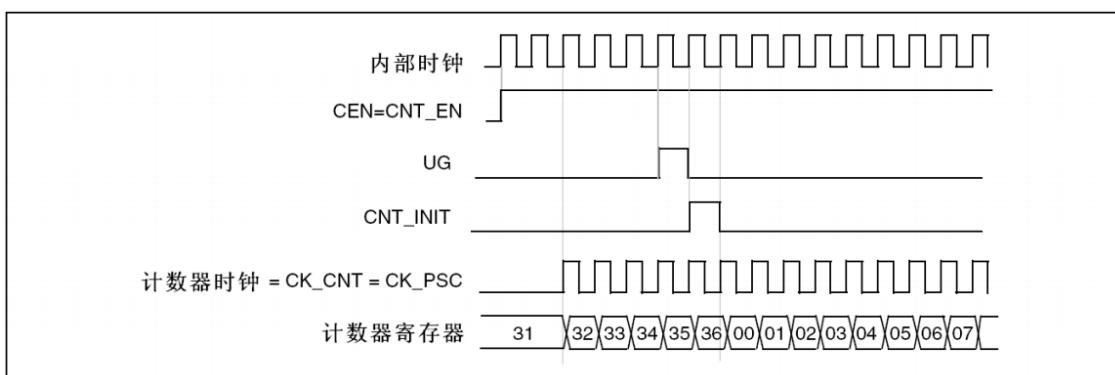
- 内部时钟（CK\_INT）。
- 外部时钟模式 1：外部输入脚（TI<sub>x</sub>）。
- 外部时钟模式 2：外部触发输入（ETR）。
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

##### 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）。一旦 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

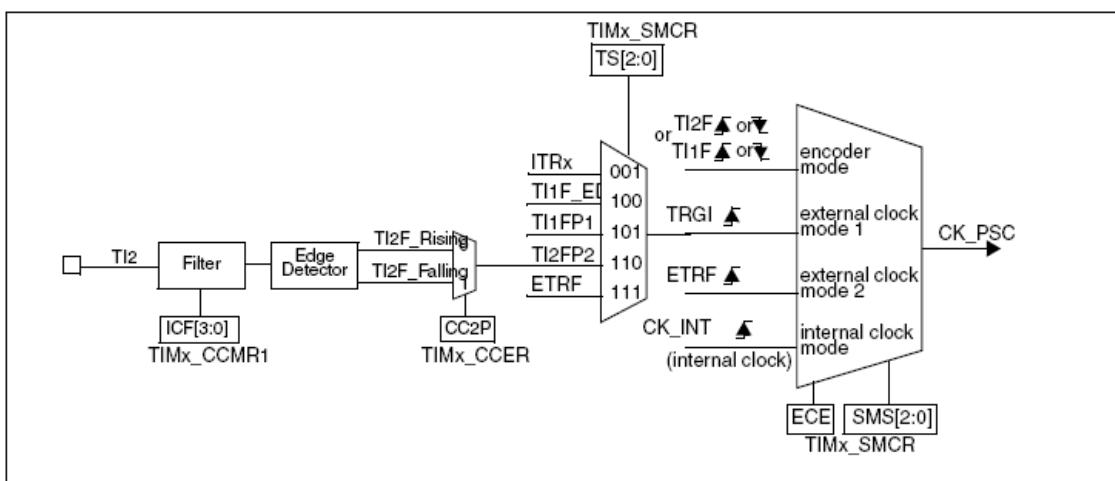
图 37. 一般模式下的控制电路，内部时钟分频因子为 1



### 外部时钟源模式 1

当 **TIMx\_SMCR** 寄存器的 **SMS=111** 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 38. TI2 外部时钟连接例子



例如，要配置向上计数器在 **TI2** 输入端的上升沿计数，使用下列步骤：

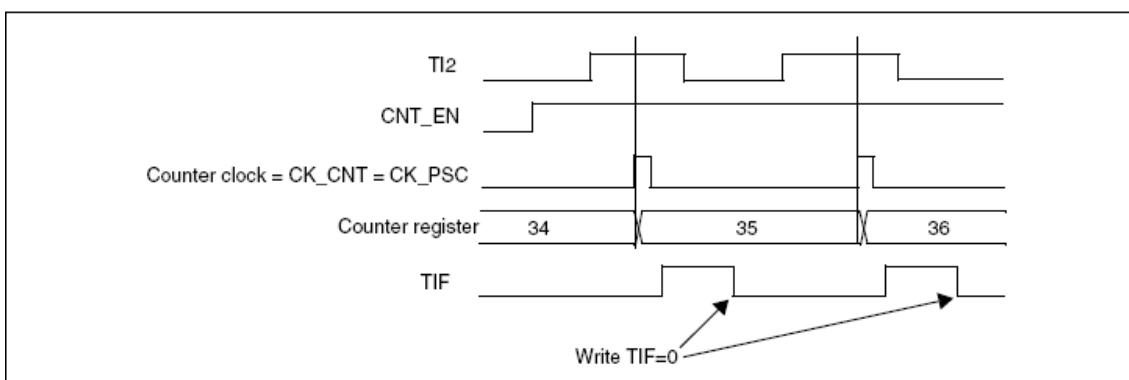
1. 配置 **TIMx\_CCMR1** 寄存器 **CC2S=01**，配置通道 2 检测 **TI2** 输入的上升沿。
2. 配置 **TIMx\_CCMR1** 寄存器的 **IC2F[3: 0]**，选择输入滤波器带宽（如果不需滤波器，保持 **IC2F=0000**）。
3. 配置 **TIMx\_CCER** 寄存器的 **CC2P=0**，选定上升沿极性。
4. 配置 **TIMx\_SMCR** 寄存器的 **SMS=111**，选择定时器外部时钟模式 1。
5. 配置 **TIMx\_SMCR** 寄存器中的 **TS=110**，选定 **TI2** 作为触发输入源。
6. 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 39. 外部时钟模式 1 下的控制电路



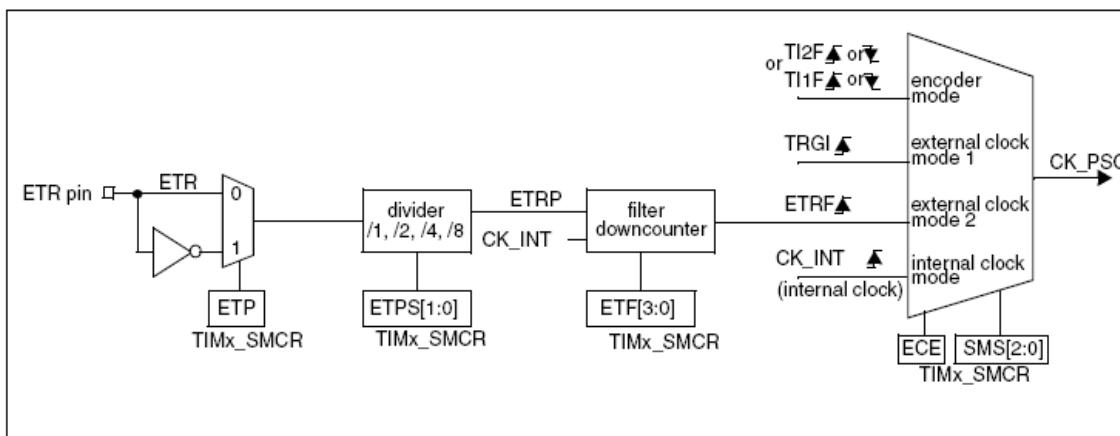
## 外部时钟源模式 2

选定此模式的方法为：令 **TIMx\_SMCR** 寄存器中的 **ECE=1**。

计数器能够在外部触发 **ETR** 的每一个上升沿或下降沿计数。

下图是外部触发输入的总体框图：

图 40. 外部触发输入框图



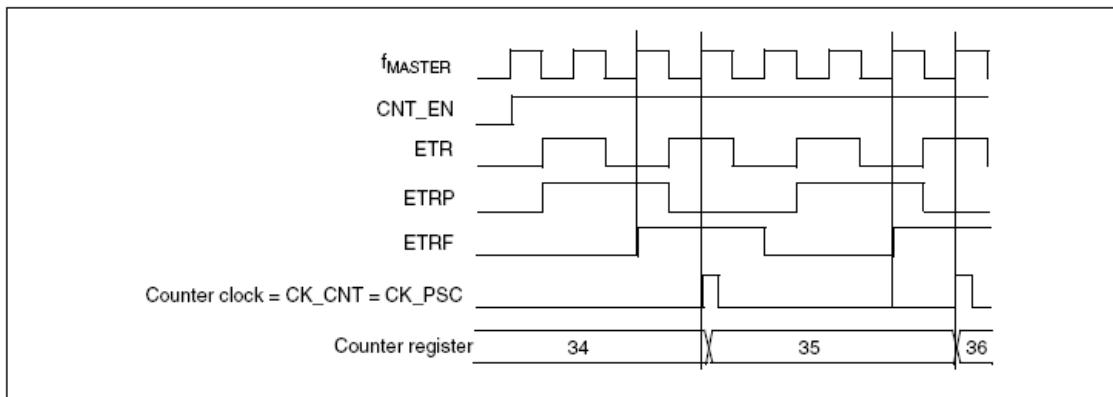
例如，要配置在 **ETR** 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 **TIMx\_SMCR** 寄存器中的 **ETF[3: 0]=0000**
2. 设置预分频器，置 **TIMx\_SMCR** 寄存器中的 **ETPS[1: 0]=01**
3. 选择 **ETR** 的上升沿检测，置 **TIMx\_SMCR** 寄存器中的 **ETP=0**
4. 开启外部时钟模式 2，写 **TIMx\_SMCR** 寄存器中的 **ECE=1**
5. 启动计数器，写 **TIMx\_CR1** 寄存器中的 **CEN=1**

计数器在每 2 个 **ETR** 上升沿计数一次。

在 **ETR** 的上升沿和计数器实际时钟之间的延时取决于在 **ETRP** 信号端的重新同步电路。

图 41. 外部时钟模式 2 下的控制电路



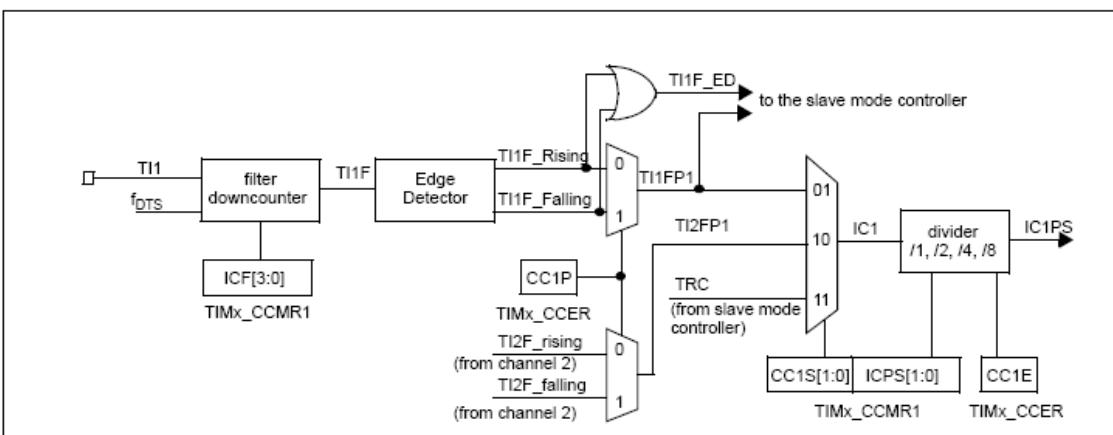
### 11.3.5 捕捉/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

图 42 至图 45 是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

图 42. 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 43. 捕获/比较通道 1 的主电路

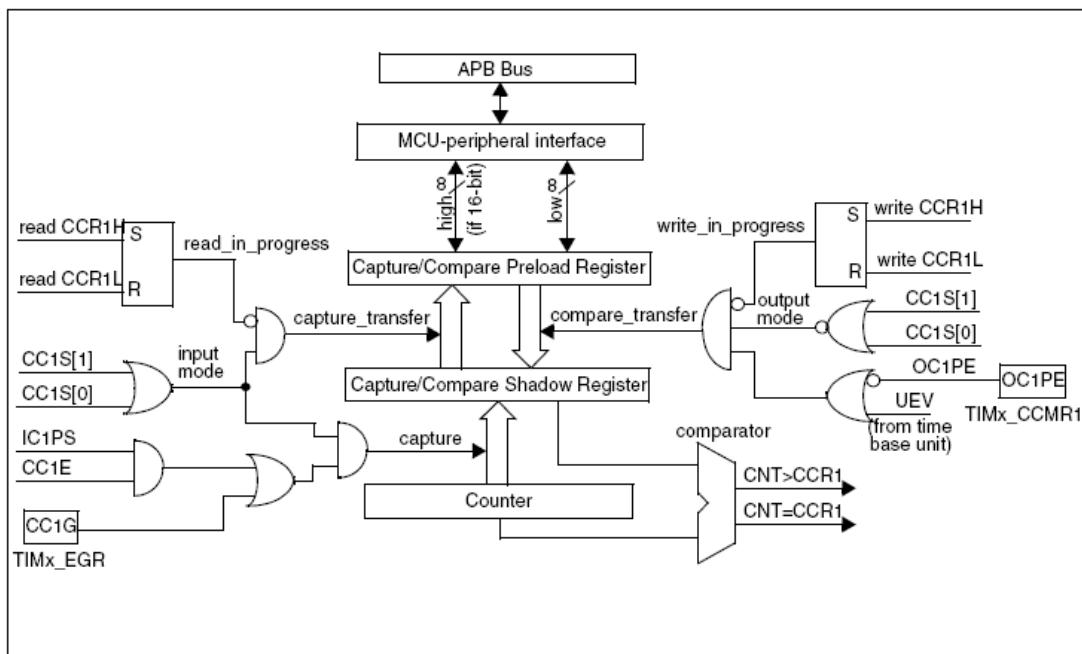


图 44. 捕获/比较通道的输出部分（通道 1 至 3）

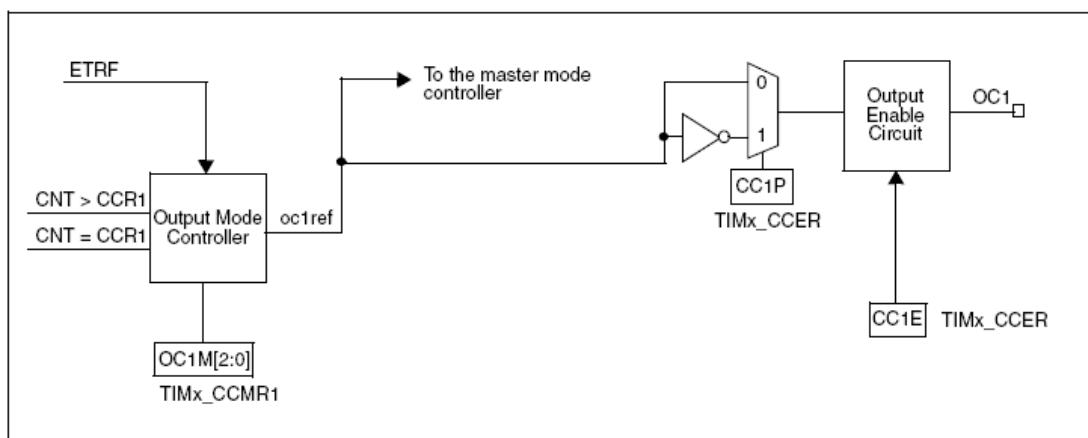
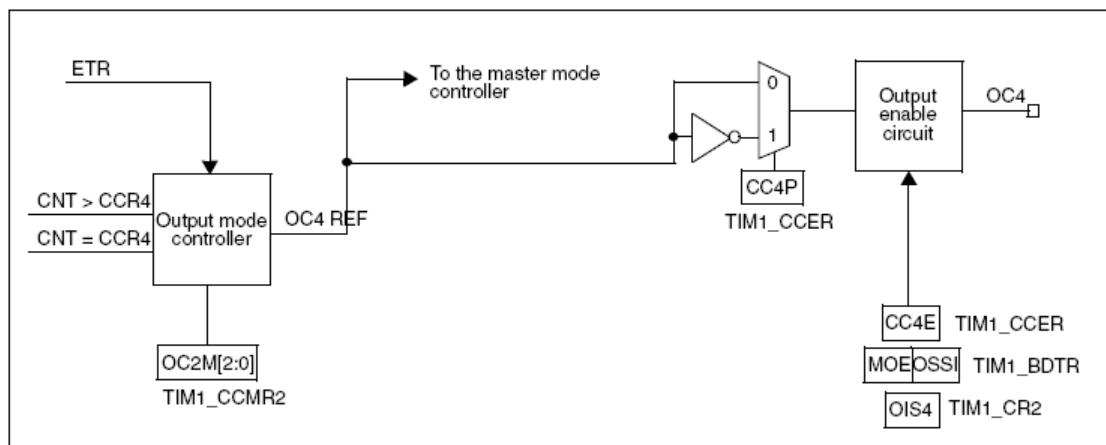


图 45. 捕获/比较通道的输出部分（通道 4）



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。

### 11.3.6 输入捕捉模式

在输入捕获模式下，当检测到  $ICx$  信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器 ( $TIMx\_CCRx$ ) 中。当发生捕获事件时，相应的  $CCxIF$  标志 ( $TIMx\_SR$  寄存器) 被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时  $CCxIF$  标志已经为高，那么重复捕获标志  $CCxOF$  ( $TIMx\_SR$  寄存器) 被置 1。写  $CCxIF=0$  可清除  $CCxIF$ ，或读取存储在  $TIMx\_CCRx$  寄存器中的捕获数据也可清除  $CCxOF$ 。写  $CCxOF=0$  可清除  $CCxOF$ 。

以下例子说明如何在  $TI1$  输入的上升沿时捕获计数器的值到  $TIMx\_CCR1$  寄存器中，步骤如下：

- 选择有效输入端： $TIMx\_CCR1$  必须连接到  $TI1$  输入，所以写入  $TIMx\_CCR1$  寄存器中的  $CC1S=01$ ，一旦  $CC1S$  不为 00 时，通道被配置为输入，并且  $TIMx\_CCR1$  寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为  $TIx$  时，输入滤波器控制位是  $TIMx\_CCMRx$  寄存器中的  $ICxF$  位）。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期；因此我们可以（以  $fDTS$  频率）连续采样 8 次，以确认在  $TI1$  上一次真实的边沿变换，即在  $TIMx\_CCMR1$  寄存器中写入  $IC1F=0011$ 。
- 选择  $TI1$  通道的有效转换边沿，在  $TIMx\_CCER$  寄存器中写入  $CC1P=0$ （上升沿）。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写  $TIMx\_CCMR1$  寄存器的  $IC1PS=00$ ）。
- 设置  $TIMx\_CCER$  寄存器的  $CC1E=1$ ，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置  $TIMx\_DIER$  寄存器中的  $CC1IE$  位允许相关中断请求，通过设置  $TIMx\_DIER$  寄存器中的  $CC1DE$  位允许 DMA 请求。

当发生一个输入捕获时：

- 当产生有效的电平转换时，计数器的值被传送到  $TIMx\_CCR1$  寄存器。
- $CC1IF$  标志被设置（中断标志）。当发生至少 2 个连续的捕获时，而  $CC1IF$  未曾被清除， $CC1OF$  也被置 1。
- 如设置了  $CC1IE$  位，则会产生一个中断。
- 如设置了  $CC1DE$  位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

**注：**设置  $TIMx\_EGR$  寄存器中相应的  $CCxG$  位，可以通过软件产生输入捕获中断和/or DMA 请求。

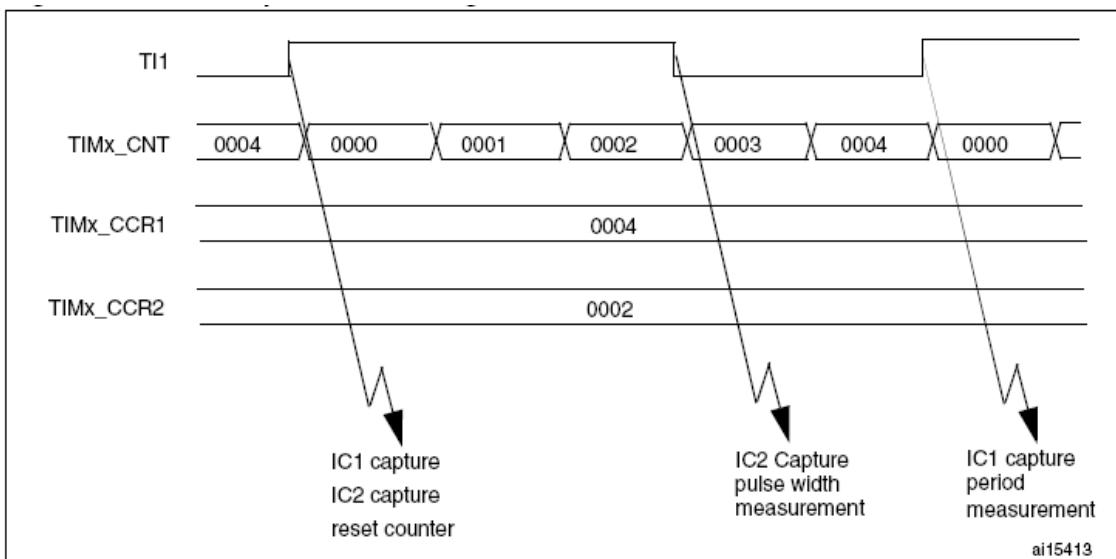
### 11.3.7 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个  $ICx$  信号被映射至同一个  $TIx$  输入。
- 这 2 个  $ICx$  信号为边沿有效，但是极性相反。
- 其中一个  $TIxFP$  信号被作为触发输入信号，而从模式控制器被配置成复位模式。例如，你需要测量输入到  $TI1$  上的 PWM 信号的长度 ( $TIMx\_CCR1$  寄存器) 和占空比 ( $TIMx\_CCR2$  寄存器)，具体步骤如下（取决于  $CK\_INT$  的频率和预分频器的值）

- 选择 TIMx\_CCR1 的有效输入：置 TIMx\_CCMR1 寄存器的 CC1S=01（选中 TI1）。
- 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx\_CCR1 中和清除计数器）：置 CC1P=0（上升沿有效）。
- 选择 TIMx\_CCR2 的有效输入：置 TIMx\_CCMR1 寄存器的 CC2S=10（选中 TI1）。
- 选择 TI1FP2 的有效极性（捕获数据到 TIMx\_CCR2）：置 CC2P=1（下降沿有效）。
- 选择有效的触发输入信号：置 TIMx\_SMCR 寄存器中的 TS=101（选择 TI1FP1）。
- 配置从模式控制器为复位模式：置 TIMx\_SMCR 中的 SMS=100。
- 使能捕获：置 TIMx\_CCER 寄存器中 CC1E=1 且 CC2E=1。

图 46. 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。

### 11.3.8 强制输出模式

在输出模式 (TIMx\_CCMRx 寄存器中 CCxS=00) 下，输出比较信号 (OCxREF 和相应的 OCx/OCxN) 能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx\_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号 (OCxREF/OCx) 为有效状态。这样 OCxREF 被强置为高电平 (OCxREF 始终为高电平有效)，同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0 (OCx 高电平有效)，则 OCx 被强置为高电平。

置 TIMx\_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

### 11.3.9 输出比较模式

此项功能是用来控制一个输出波形或者指示何时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管脚上。在比较匹配时, 输出管脚可以保持它的电平(OCxM=000)、被设置成有效电平(OCxM=001)、被设置成无有效电平(OCxM=010)或进行 翻转(OCxM=011)。
- 设置中断状态寄存器中的标志位 (TIMx\_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx\_DIER 寄存器中的 CCxIE 位), 则产生一个中断。
- 若设置了相应的使能位 (TIMx\_DIER 寄存器中的 CCxDE 位, TIMx\_CR2 寄存器中的 CCDS 位选择 DMA 请求功能), 则产生一个 DMA 请求。

TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下, 更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

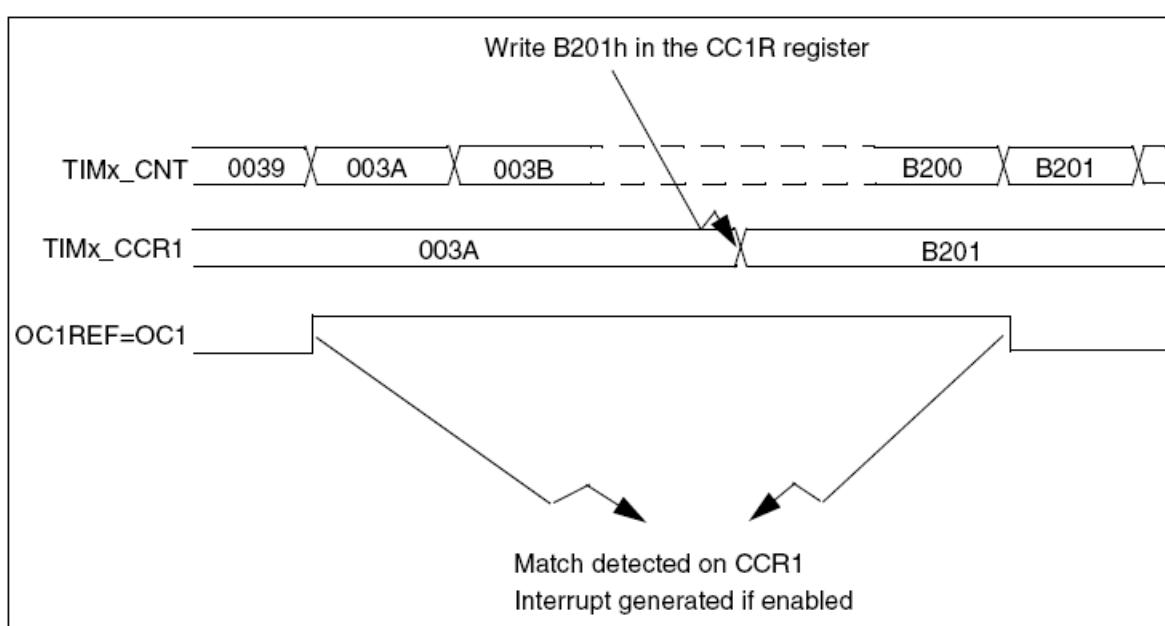
同步的精度可以达到计数器的一个计数周期。输出比较模式 (在单脉冲模式下) 也能用来输出一个单脉冲。

输出比较模式的配置步骤:

- 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 的输出管脚, 设置 OCxM=011
  - 置 OCxPE = 0 禁用预装载寄存器
  - 置 CCxE = 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 47. 输出比较模式, 翻转 OC1



### 11.3.10 PWM 模式

脉冲宽度调制模式可以产生一个由 **TIMx\_ARR** 寄存器确定频率、由 **TIMx\_CCRx** 寄存器确定占空比的信号。

在 **TIMx\_CCMRx** 寄存器中的 **OCxM** 位写入‘110’（PWM 模式 1）或‘111’（PWM 模式 2），能够独立地设置每个 **OCx** 输出通道产生一路 PWM。必须通过设置 **TIMx\_CCMRx** 寄存器的 **OCxPE** 位使能相应的预装载寄存器，最后还要设置 **TIMx\_CR1** 寄存器的 **ARPE** 位使能自动重装载的预装载寄存器（在向上计数或中心对称模式中）。

因为仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 **TIMx\_EGR** 寄存器中的 **UG** 位来初始化所有的寄存器。

**OCx** 的极性可以通过软件在 **TIMx\_CCER** 寄存器中的 **CCxP** 位设置，它可以设置为高电平有效或低电平有效。**OCx** 的输出使能通过（**TIMx\_CCER** 和 **TIMx\_BDTR** 寄存器中）**CCxE**、**CCxNE**、**MOE**、**OSSI** 和 **OSSR** 位的组合控制。详见 **TIMx\_CCER** 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，**TIMx\_CNT** 和 **TIMx\_CCRx** 始终在进行比较，（依据计数器的计数方向）以确定是否符合 **TIMx\_CCRx**  $\leq$  **TIMx\_CNT** 或者 **TIMx\_CNT**  $\leq$  **TIMx\_CCRx**。

根据 **TIMx\_CR1** 寄存器中 **CMS** 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

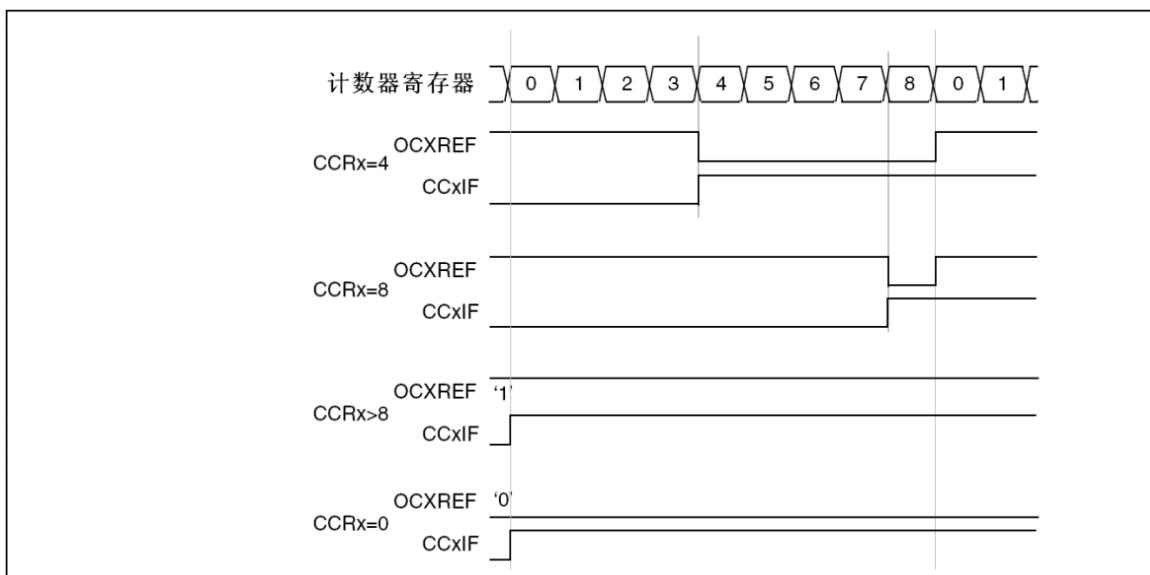
#### PWM 边沿对齐模式

##### 向上计数配置

当 **TIMx\_CR1** 寄存器中的 **DIR** 位为低的时候执行向上计数。参看 11.3.2 节。

下面是一个 PWM 模式 1 的例子。当 **TIMx\_CNT** < **TIMx\_CCRx** 时，PWM 参考信号 **OCxREF** 为高，否则低。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’。图 48 为 **TIMx\_ARR = 8** 时边沿对齐的 PWM 波形实例。

图 48. 边沿对齐的 PWM 波形（**ARR = 8**）



## 向下计数的配置

当  $\text{TIMx\_CR1}$  寄存器的 DIR 位为高时执行向下计数。参看 11.3.2 节。

在 PWM 模式 1, 当  $\text{TIMx\_CNT} > \text{TIMx\_CCR}_x$  时参考信号  $\text{OCxREF}$  为低, 否则为高。如果  $\text{TIMx\_CCR}_x$  中的比较值大于  $\text{TIMx\_ARR}$  中的自动重装载值, 则  $\text{OCxREF}$  保持为'1'。该模式下不能产生 0% 的 PWM 波形。

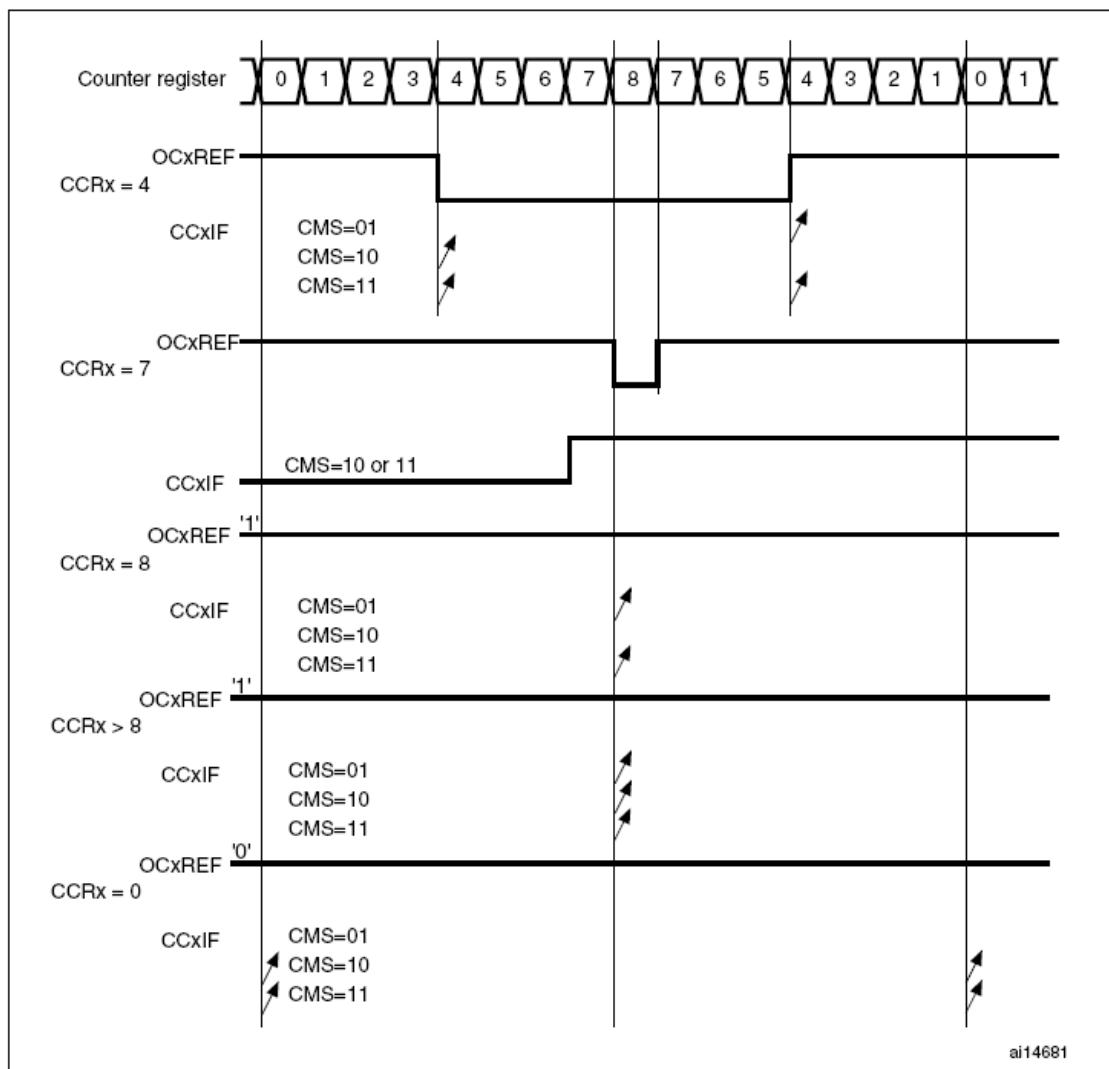
## PWM 中央对齐模式

当  $\text{TIMx\_CR1}$  寄存器中的 CMS 位不为'00'时为中央对齐模式（所有其他的配置对  $\text{OCxREF}/\text{O Cx}$  信号都有相同的作用）。根据不同的 CMS 位的设置, 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'。 $\text{TIMx\_CR1}$  寄存器中的计数方向位 (DIR) 由硬件更新, 不要用软件修改它。参看 11.3.2 节的中央对齐模式。

图 49 给出了一些中央对齐的 PWM 波形的例子

- $\text{TIMx\_ARR}=8$
- PWM 模式 1
- $\text{TIMx\_CR1}$  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 49. 中央对齐的 PWM 波形 (APR = 8)



### 使用中央对齐模式的提示:

- 进入中央对齐模式时，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数取决于 **TIMx\_CR1** 寄存器中 **DIR** 位的当前值。此外，软件不能同时修改 **DIR** 和 **CMS** 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值 (**TIMx\_CNT > TIMx\_ARR**)，则方向不会被更新
  - 例如，如果计数器正在向上计数，它就会继续向上计数
  - 如果将 0 或者 **TIMx\_ARR** 的值写入计数器，方向被更新，但不产生更新事件 **UEV**
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置 **TIMx\_EGR** 位中的 **UG** 位），不要在计数进行过程中修改计数器的值。

### 11.3.11 互补输出和死区插入

高级控制定时器（**TIM1** 和 **TIM2**）能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性（电平转换的延时、电源开关的延时等）来调整死区时间。

配置 **TIMx\_CCER** 寄存器中的 **CCxP** 和 **CCxNP** 位，可以为每一个输出独立地选择极性（主输出 **OCx** 或互补输出 **OCxN**）。

互补信号 **OCx** 和 **OCxN** 通过下列控制位的组合进行控制：**TIMx\_CCER** 寄存器的 **CCxE** 和 **CCxNE** 位，**TIMx\_BDTR** 和 **TIMx\_CR2** 寄存器中的 **MOE**、**OISx**、**OISxN**、**OSSI** 和 **OSSR** 位，详见表 56 带刹车功能的互补输出通道 **OCx** 和 **OCxN** 的控制位。特别的是，在转换到 **IDLE** 状态时（**MOE** 下降 到 0）死区被激活。

同时设置 **CCxE** 和 **CCxNE** 位将插入死区，如果存在刹车电路，则还要设置 **MOE** 位。每一个通道都有一个 10 位的死区发生器。参考信号 **OCxREF** 可以产生 2 路输出 **OCx** 和 **OCxN**。如果 **OCx** 和 **OCxN** 为高有效：

- **OCx** 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- **OCxN** 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。如果延时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 50. 带死区插入的互补输出

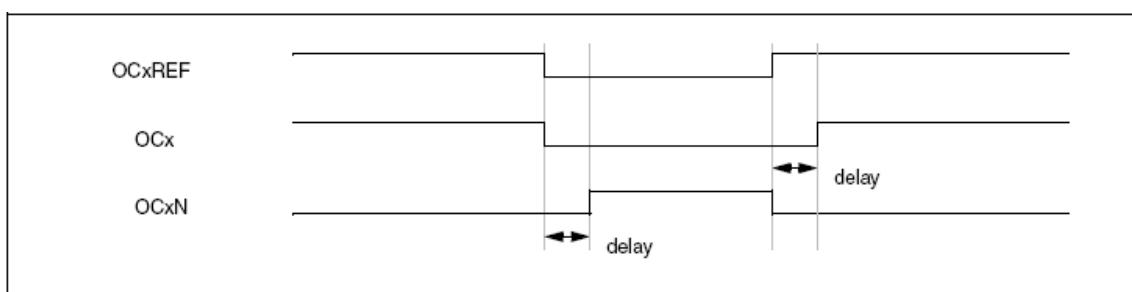


图 51. 死区波形延迟大于负脉冲

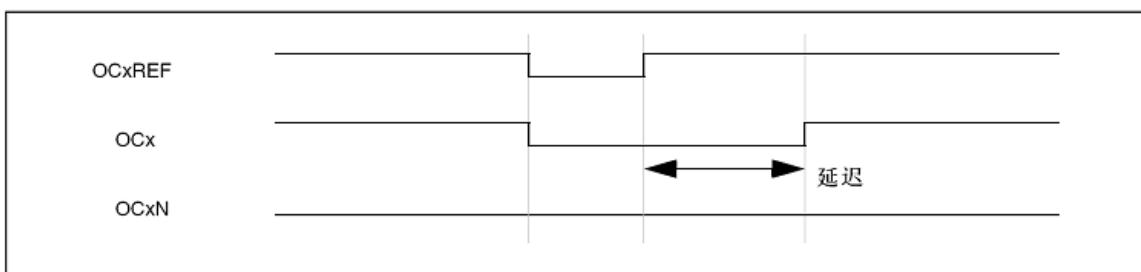
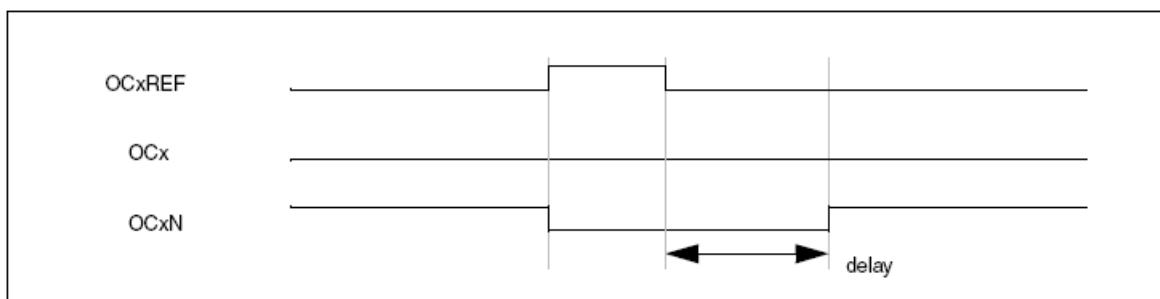


图 52. 死区波形延迟大于正脉冲



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 11.4.18 节中的延时计算。

#### 重定向 OCxREF 到 OCx 或 OCxN

在输出模式下（强置、输出比较或 PWM），通过配置 **TIMx\_CCER** 寄存器的 **CCxE** 和 **CCxNE** 位，**OCxREF** 可以被重定向到 **OCx** 或者 **OCxN** 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形（例如 PWM 或者静态有效电平）。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

**注：**当只使能 **OCxN** (**CCxE** = 0, **CCxNE** = 1) 时，它不会反相，当 **OCxREF** 有效时立即变高。例如，如果 **CCxNP** = 0，则 **OCxN** = **OCxREF**。另一方面，当 **OCx** 和 **OCxN** 都被使能时 (**CCxE** = **CCxNE** = 1)，当 **OCxREF** 为高时 **OCx** 有效；而 **OCxN** 相反，当 **OCxREF** 低时 **OCxN** 变为有效。

#### 11.3.12 使用刹车功能

当使用刹车功能时，依据相应的控制位 (**TIMx\_BDTR** 寄存器中的 **MOE**、**OSSI** 和 **OSSR** 位，**TIMx\_CR2** 寄存器中的 **OISx** 和 **OISxN** 位)，输出使能信号和无效电平都会被修改。但无论何时，**OCx** 和 **OCxN** 输出不能在同一时间同时处于有效电平上。详见寄存器表中带刹车功能的互补输出通道 **OCx** 和 **OCxN** 的控制位。

刹车源既可以是刹车输入管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。

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** 可以被同时修改。

因为 **MOE** 下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（在 **TIMx\_BDTR** 寄存器中）之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果

当它为低时写 **MOE=1**，则读出它之前必须先插入一个延时（空指令）才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时（在刹车输入端出现选定的电平），有下述动作：

- **MOE** 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态（由 **OSSI** 位选择）。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 **MOE = 0**，每一个输出通道输出由 **TIMx\_CR2** 寄存器中的 **OISx** 位设定的电平。如果 **OSSI = 0**，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 **OISx** 和 **OISxN** 位指示的电平驱动输出端口。即使在这种情况下，**OCx** 和 **OCxN** 也不能被同时驱动到有效的电平。注，因为重新同步 **MOE**，死区时间比通常情况下长一些（大约 2 个 **CK\_TIM** 的时钟周期）。
  - 如果 **OSSI = 0**，定时器释放使能输出，否则保持使能输出；或一旦 **CCxE** 与 **CCxNE** 之一变高时，使能输出变为高。
- 如果设置了 **TIMx\_DIER** 寄存器中的 **BIE** 位，当刹车状态标志（**TIMx\_SR** 寄存器中的 **BIF** 位）为'1'时，则产生一个中断。如果设置了 **TIMx\_DIER** 寄存器中的 **BDE** 位，则产生一个 DMA 请求。
- 如果设置了 **TIMx\_BDTR** 寄存器中的 **AOE** 位，在下一个更新事件 **UEV** 时 **MOE** 位被自动置位；例如，这可以用来进行整形。否则，**MOE** 始终保持低直到被再次置'1'；此时，这个特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

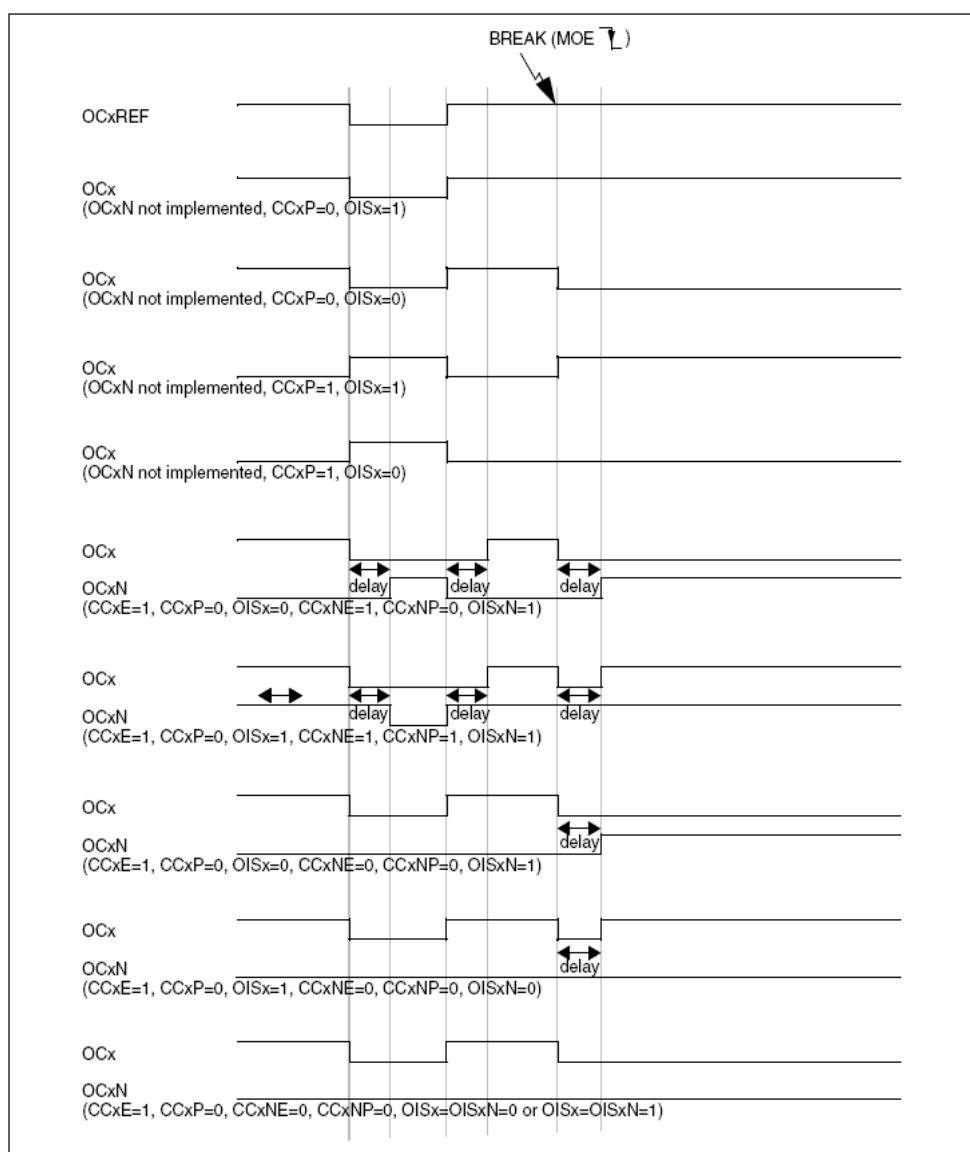
**注：** 刹车输入为电平有效。所以，当刹车输入有效时，不能同时（自动地或者通过软件）设置 **MOE**。同时，状态标志 **BIF** 不能被清除。

刹车由 **BRK** 输入产生，它的有效极性是可编程的，且由 **TIMx\_BDTR** 寄存器中的 **BKE** 位开启。

除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数（死区长度，**OCx/OCxN** 极性和被禁止的状态，**OCxM** 配置，刹车使能和极性）。用户可以通过 **TIMx\_BDTR** 寄存器中的 **LOCK** 位，从三级保护中选择一种，参看 11.4.18 节。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 53. 响应刹车的输出



#### 11.3.13 在外部事件时清除 OCxREF 信号

对于一个给定的通道，在 ETRF 输入端（设置 TIMx\_CCMRx 寄存器中对应的 OCxCE 位为‘1’）的高电平能够把 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。

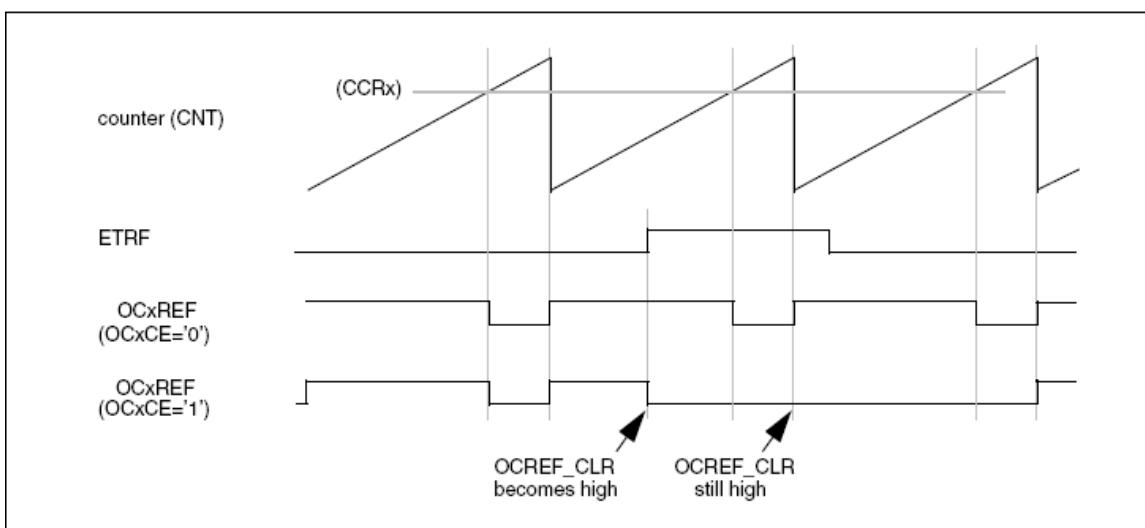
该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以连到一个外部输入。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 54. 清除 TIMx 的 OCxREF



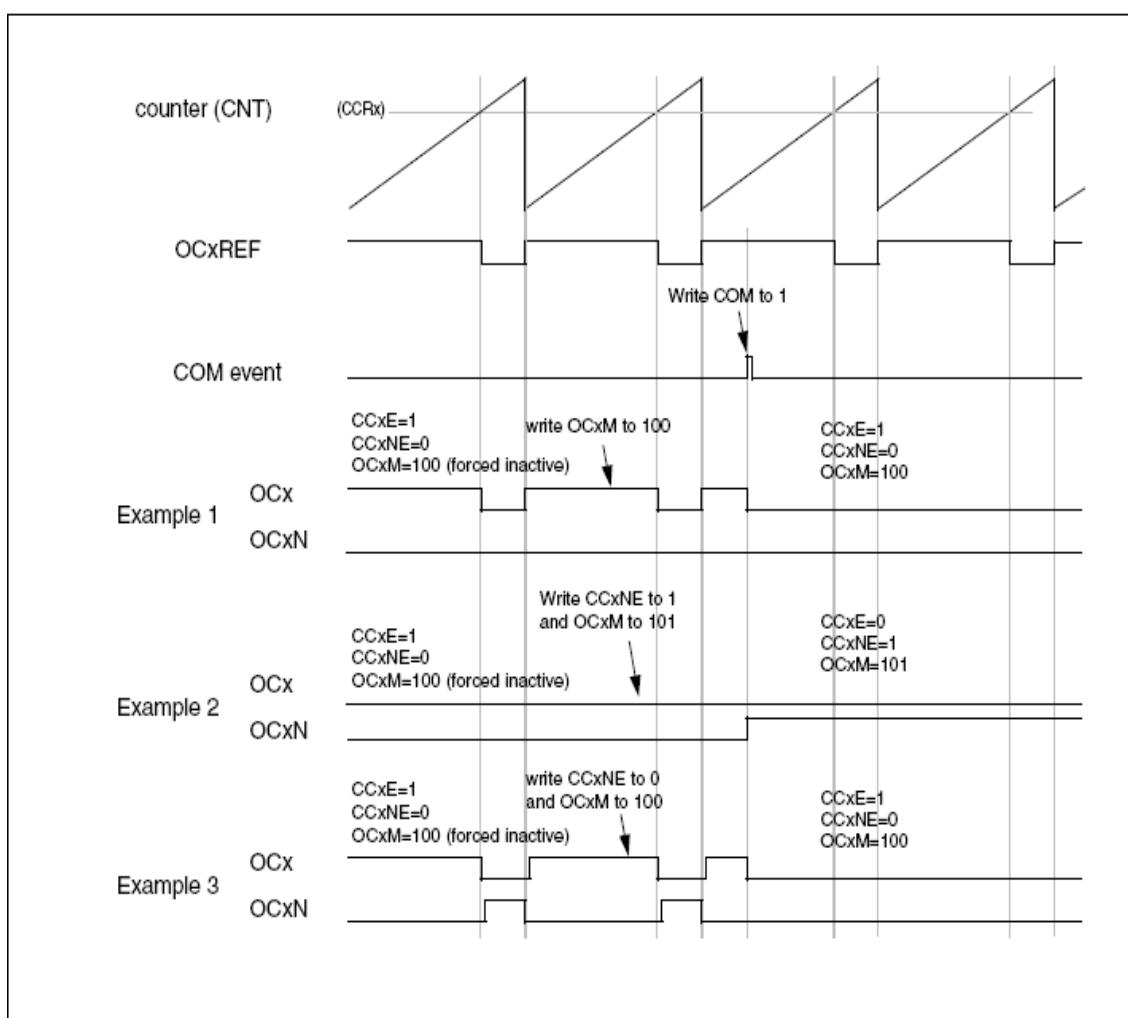
#### 11.3.14 产生六步 PWM 输出

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修改所有通道的配置。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 55. 产生六步 PWM，使用 COM 的例子 (OSSR = 1)



### 11.3.15 单脉冲模式

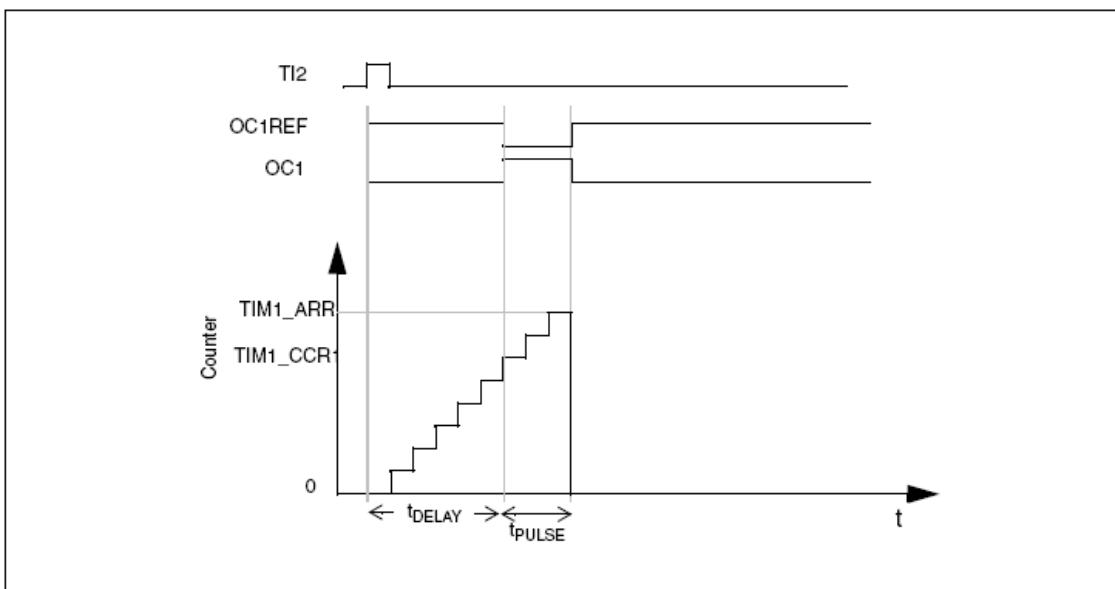
单脉冲模式 (OPM) 是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIM<sub>x</sub>\_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 < CCR<sub>x</sub> ≤ ARR（特别地，0 < CCR<sub>x</sub>）
- 向下计数方式：计数器 CNT > CCR<sub>x</sub>

图 56. 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 1：

- 置 TIMx\_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映像到 TI2。
- 置 TIMx\_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx\_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发 (TRGI)。
- 置 TIMx\_SMCR 寄存器中的 SMS=110 (触发模式)，TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)

- $t_{DELAY}$  由 TIMx\_CCR1 寄存器中的值定义。
- $t_{PULSE}$  由自动装载值和比较值之间的差值定义 (TIMx\_ARR - TIMx\_CCR1)。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx\_CCMR1 寄存器的 OC1M = 111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx\_CCMR1 中的 OC1PE = 1 和 TIMx\_CR1 寄存器中的 ARPE；然后在 TIMx\_CCR1 寄存器中填写比较值，在 TIMx\_ARR 寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P = 0。

在这个例子中，TIMx\_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx\_CR1 寄存器中的 OPM = 1，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

#### 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时  $t_{DELAY}$ 。

如果要以最小延时输出波形，可以设置 `TIMx_CCMRx` 寄存器中的 `OCxFE` 位；此时强制 `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。

### 11.3.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 `TI2` 的边沿计数，则置 `TIMx_SMCR` 寄存器中的 `SMS = 001`；如果只在 `TI1` 边沿计数，则置 `SMS = 010`；如果计数器同时在 `TI1` 和 `TI2` 边沿计数，则置 `SMS = 011`。

通过设置 `TIMx_CCER` 寄存器中的 `CC1P` 和 `CC2P` 位，可以选择 `TI1` 和 `TI2` 极性；如果需要，还可以对输入滤波器编程。两个输入 `TI1` 和 `TI2` 被用来作为增量编码器的接口。参看表 54，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`；如果没有滤波和变相，则 `TI2FP2 = TI2`。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 `TIMx_CR1` 寄存器的 `DIR` 位进行相应的设置。不管计数器是依靠 `TI1` 计数、依靠 `TI2` 计数或者同时依靠 `TI1` 和 `TI2` 计数，在任一输入端（`TI1` 或者 `TI2`）的跳变都会重新计算 `DIR` 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 `TIMx_ARR` 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 `ARR` 计数，或是 `ARR` 到 0 计数）。所以在开始计数之前必须配置 `TIMx_ARR`；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 `TI1` 和 `TI2` 不同时变换。

表 24. 计数方向与编码器信号的关系

有效边沿	相对信号的电平 ( <code>TI1FP1</code> 对应 <code>TI2</code> , <code>TI2FP2</code> 对应 <code>TI1</code> )	<code>TI1FP1</code> 信号		<code>TI2FP2</code> 信号	
		上升	下降	上升	下降
仅在 <code>TI1</code> 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 <code>TI2</code> 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 <code>TI1</code> 和 <code>TI2</code> 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

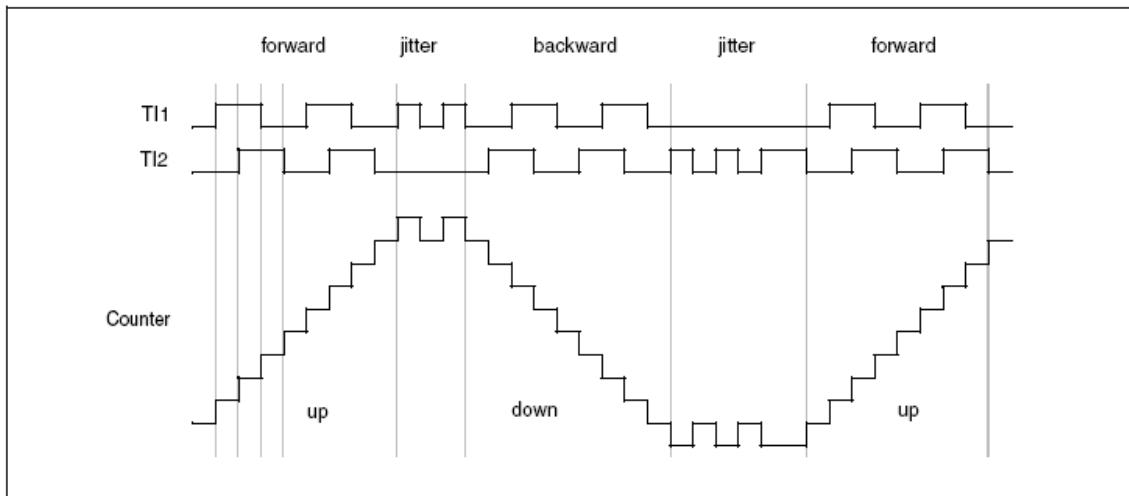
一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- `CC1S = '01'` (`TIMx_CCMR1` 寄存器, `IC1FP1` 映射到 `TI1`)
- `CC2S = '01'` (`TIMx_CCMR2` 寄存器, `IC2FP2` 映射到 `TI2`)
- `CC1P = '0'` (`TIMx_CCER` 寄存器, `IC1FP1` 不反相, `IC1FP1=TI1`)

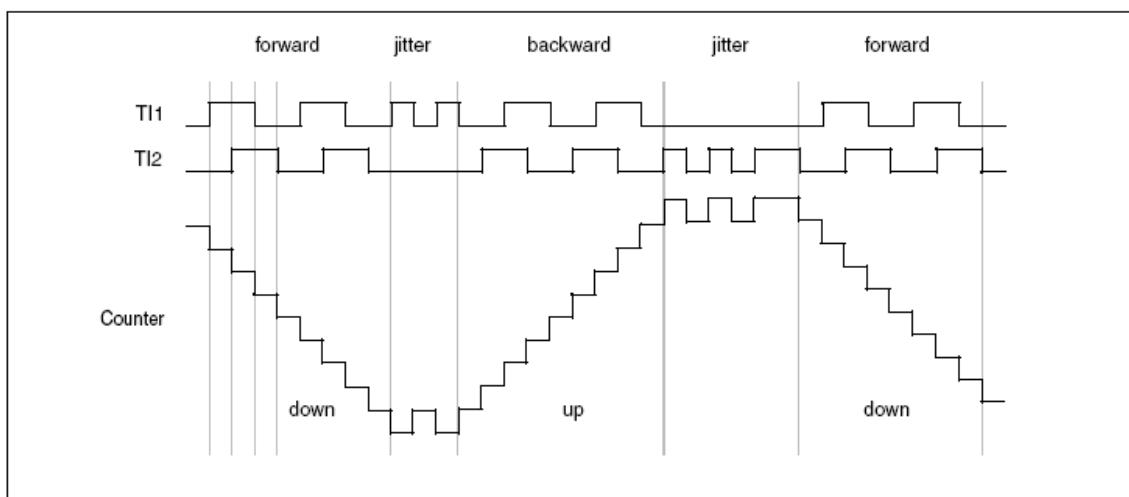
- CC2P = '0' (TIMx\_CCER 寄存器, IC2FP2 不反相, IC2FP2=TI2)
- SMS = '011' (TIMx\_SMCR 寄存器, 所有的输入均在上升沿和下降沿有效) .
- CEN = '1' (TIMx\_CR1 寄存器, 计数器使能)

图 57. 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 58. IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器测量两个编码器事件的间隔，可以获得动态的信息（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）。它也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 11.3.17 定时器输入异或功能

TIMx\_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx\_CH1、TIMx\_CH2 和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。下节 11.3.18 给出了此特性用于连接霍尔传感器的例子。

### 11.3.18 与霍尔传感器的接口

使用高级控制定时器（TIM1 或 TIM2）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 59，3 个定时器输入脚（CC1、CC2、CC3）通过一个异或门连接到 TI1 输入通道（通过设置 TIMx\_CR2 寄存器中的 TI1S 位 来选择），‘接口定时器’捕获这个信号。

从模式控制器被配置于复位模式，从输入是 TI1F\_ED。每当 3 个输入之一变化时，计数器从新从 0 开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

‘接口定时器’上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC（见图 42）。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 TIM2）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 TIM2）。

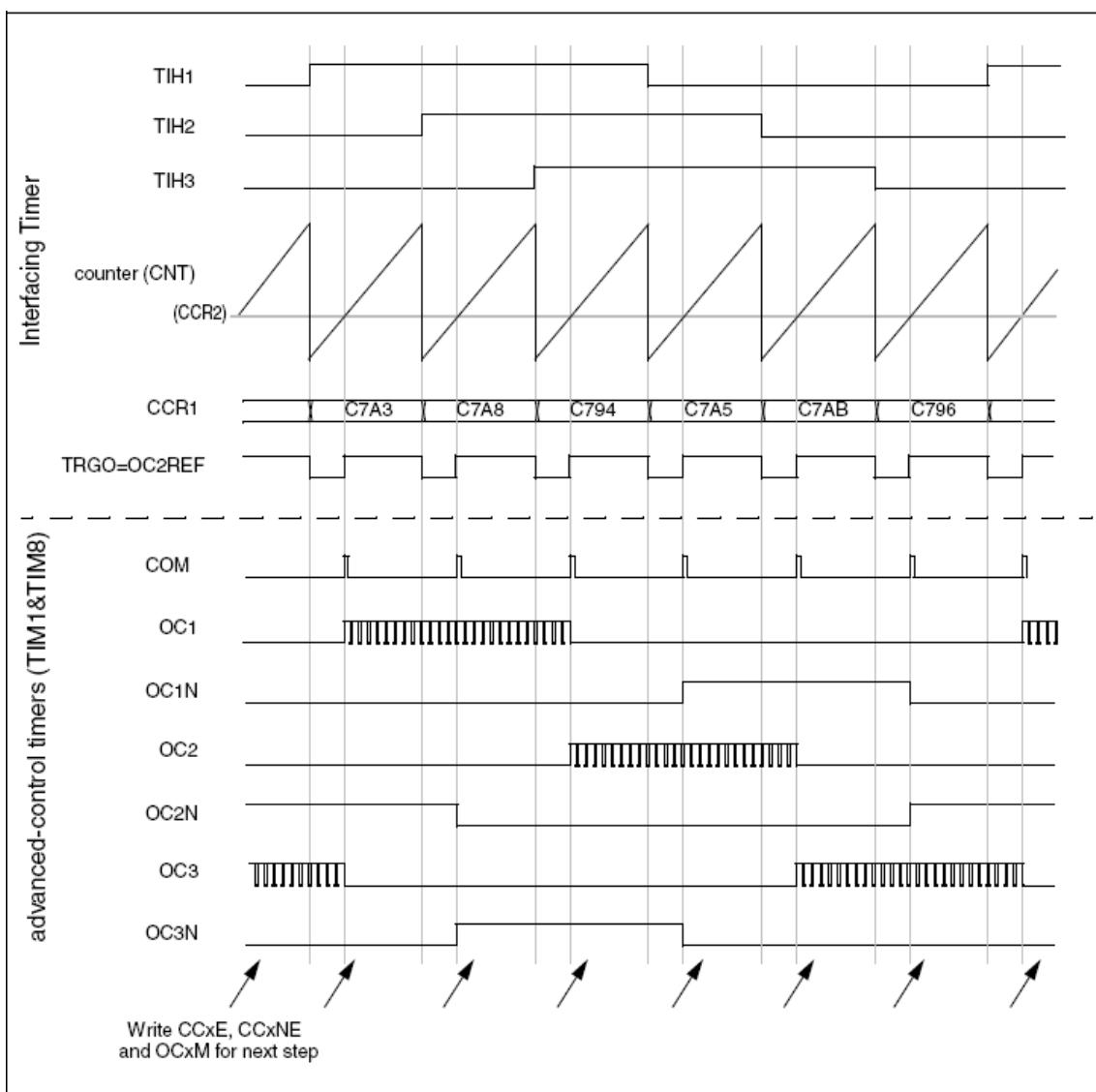
**举例：**霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIMx 的 PWM 配置。

- 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 59. 霍尔传感器接口的实例



### 11.3.19 TIMx 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 IMx\_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

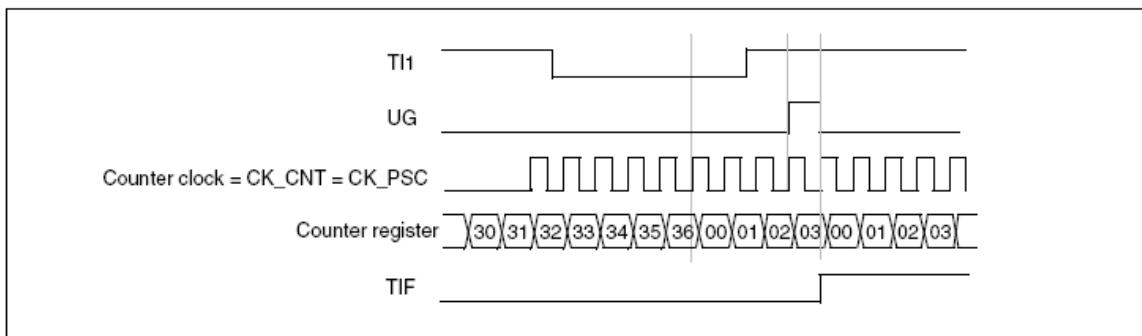
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F = 0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1 寄存器中 CC1S=01。置 TIMx\_CCER 寄存器中 CC1P = 0 以确定极性（只检测上升沿）。

- 置 **TIMx\_SMCR** 寄存器中 **SMS = 100**, 配置定时器为复位模式; 置 **TIMx\_SMCR** 寄存器中 **TS = 101**, 选择 **TI1** 作为输入源。
- 置 **TIMx\_CR1** 寄存器中 **CEN = 1**, 启动计数器。

计数器开始依据内部时钟计数, 然后正常运转直到 **TI1** 出现一个上升沿; 此时, 计数器被清零然后从 0 重新开始计数。同时, 触发标志 (**TIMx\_SR** 寄存器中的 **TIF** 位) 被设置, 根据 **TIMx\_DIER** 寄存器中 **TIE** (中断使能) 位和 **TDE** (DMA 使能) 位的设置, 产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 **TIMx\_ARR = 0x36** 时的动作。在 **TI1** 上升沿和计数器的实际复位之间的延时取决于 **TI1** 输入端的重同步电路。

图 60. 复位模式下的控制电路



### 从模式: 门控模式

计数器的使能依赖于选中的输入端的电平。

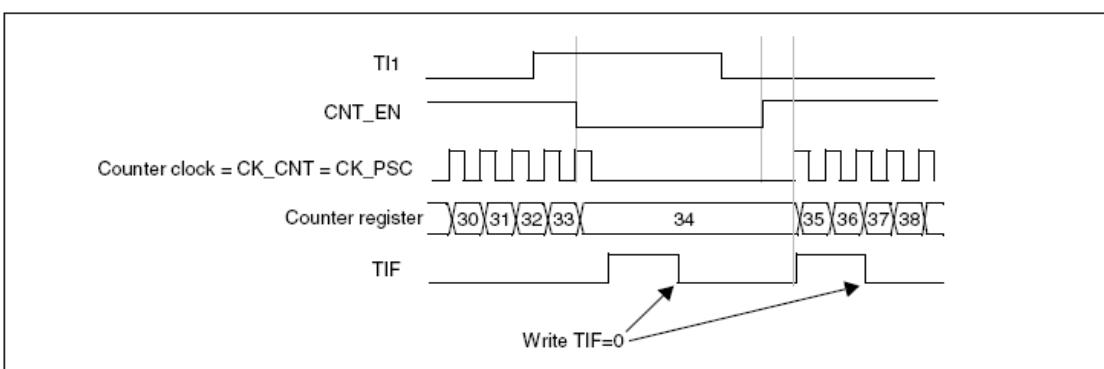
在如下的例子中, 计数器只在 **TI1** 为低时向上计数:

- 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 61. 门控模式下的控制电路



### 从模式：触发模式

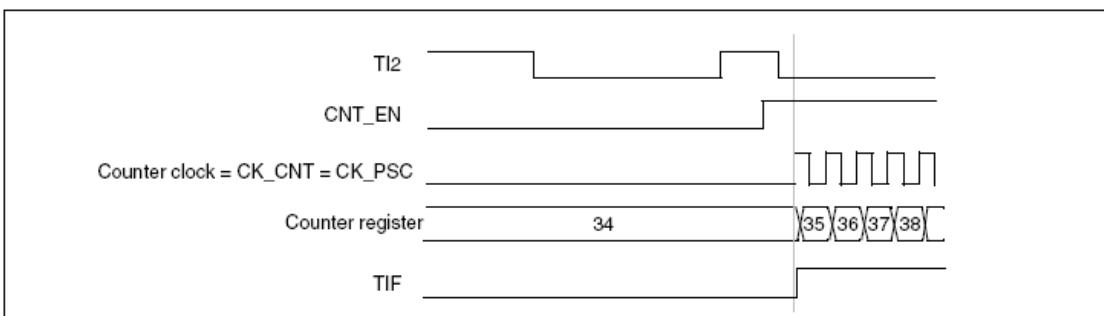
计数器的使能依赖于选中的输入端上的事件。

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 62. 触发器模式下的控制电路



### 从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，**ETR** 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 **TIMx\_SMCR** 寄存器的 **TS** 位选择 **ETR** 作为 **TRGI**。

在下面的例子中，一旦在 **TI1** 上出现一个上升沿，计数器即在 **ETR** 的每一个上升沿向上计数一次：

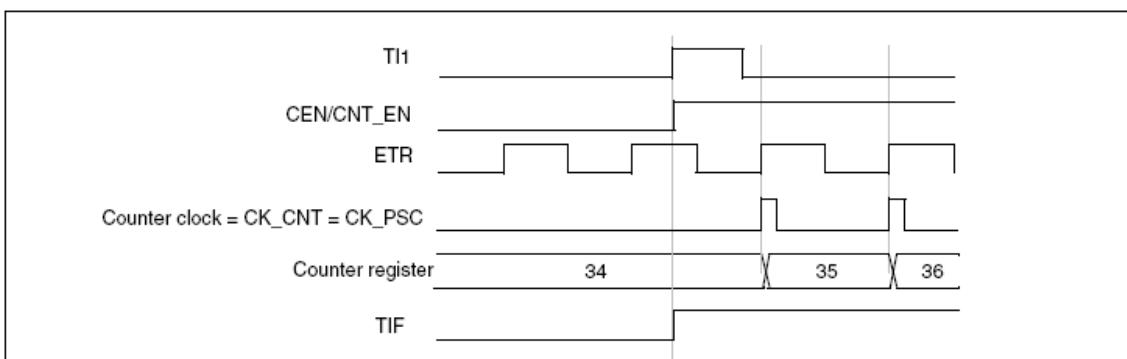
- 通过 **TIMx\_SMCR** 寄存器配置外部触发输入电路：
  - **ETF = 0000**: 没有滤波
  - **ETPS = 00**: 不用预分频器
  - **ETP = 0**: 检测 **ETR** 的上升沿，置 **ECE = 1** 使能外部时钟模式 2。
- 按如下配置通道 1，检测 **TI** 的上升沿：

- IC1F = 0000: 没有滤波
- 触发操作中不使用捕获预分频器, 不需要配置
- 置 TIMx\_CCMR1 寄存器中 CC1S = 01, 选择输入捕获源
- 置 TIMx\_CCER 寄存器中 CC1P = 0 以确定极性 (只检测上升沿)
- 置 TIMx\_SMCR 寄存器中 SMS = 110, 配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS = 101, 选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时, TIF 标志被设置, 计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时取决于 ETRP 输入端的重同步电路。

图 63. 外部时钟模式 2+触发模式下的控制电路



### 11.3.20 定时器同步

所有 TIM 定时器在内部相连, 用于定时器同步或链接。详见下一章 TIM2/3/4。

### 11.3.21 调试模式

当微控制器进入调试模式时 (CPU 核心停止), 根据 DBG 模块中 DBG\_TIMx\_STOP 的设置, TIMx 计数器可以或者继续正常操作, 或者停止。详见随后的调试章节。

## 11.4 寄存器描述

### 11.4.1 控制寄存器 1 (TIMx\_CR1)

偏移地址: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 10 保留

位9: 8	<p><b>CKD[1: 0]:</b> 时钟分频因子 (Clock division)          这2位定义在定时器时钟 (CK_INT) 频率、死区时间和由死区发生器与数字滤波器 (ETR, TIx) 所用的采样时钟之间的分频比例。</p> <p>00: <math>t_{DTs} = t_{CK\_INT}</math>          01: <math>t_{DTs} = 2 \times t_{CK\_INT}</math>          10: <math>t_{DTs} = 4 \times t_{CK\_INT}</math>          11: 保留, 不要使用这个配置</p>
位7	<p><b>ARPE:</b> 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>
位6: 5	<p><b>CMS[1: 0]:</b> 选择中央对齐模式 (Center-aligned mode selection)</p> <p>00: 边沿对齐模式。计数器依据方向位 (DIR) 向上或向下计数          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。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。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>
位4	<p><b>DIR:</b> 方向 (Direction)</p> <p>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>
位3	<p><b>OPM:</b> 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>
位2	<p><b>URS:</b> 更新请求源 (Update request source)</p> <p>软件通过该位选择UEV事件的源。</p> <p>0: 如果允许产生更新中断或DMA请求, 则下述任一事件产生一个更新中断或DMA请求:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置UG位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果允许产生更新中断或DMA请求, 则只有计数器溢出/下溢才产生一个更新中断或DMA请求</p>
位1	<p><b>UDIS:</b> 禁止更新 (Update disable)</p> <p>软件通过该位允许/禁止UEV事件的产生</p> <p>0: 允许UEV。更新 (UEV) 事件由下述任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置UG位</li> <li>- 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。</li> </ul> <p>1: 禁止UEV。不产生更新事件, 影子寄存器 (ARR、PSC、CCRx) 保持它们的值。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>

位0	<p><b>CEN:</b> 允许计数器 (Counter enable)</p> <p>0: 禁止计数器</p> <p>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>
----	--

#### 11.4.2 控制寄存器 2 (TIMx\_CR2)

偏移地址: 0x04

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OIS4	OIS3 N	OIS3	OIS2 N	OIS2	OIS1 N	OIS1	TI1S		MMS	CCDS	CCUS	保留	CCPC	

rw rw

位31: 15	保留
位14	<b>OIS4:</b> 输出空闲状态4 (OC4输出)。参见OIS1位。
位13	<b>OIS3N:</b> 输出空闲状态3 (OC3N输出)。参见OIS1N位。
位12	<b>OIS3:</b> 输出空闲状态3 (OC3输出)。参见OIS1位。
位11	<b>OIS2N:</b> 输出空闲状态2 (OC2N输出)。参见OIS1N位。
位10	<b>OIS2:</b> 输出空闲状态2 (OC2输出)。参见OIS1位。
位9	<p><b>OIS1N:</b> 输出空闲状态1 (OC1N输出) (Output Idle state 1)</p> <p>0: 当MOE = 0时, 死区后OC1N = 0</p> <p>1: 当MOE = 0时, 死区后OC1N = 1</p> <p>注: 已经设置了LOCK (TIMx_BKR寄存器) 级别1、2或3后, 该位不能被修改。</p>
位8	<p><b>OIS1:</b> 输出空闲状态1 (OC1输出) (Output Idle state 1)</p> <p>0: 当MOE = 0时, 如果实现了OC1N, 则死区后OC1 = 0</p> <p>1: 当MOE = 0时, 如果实现了OC1N, 则死区后OC1 = 1。</p> <p>注: 已经设置了LOCK (TIMx_BKR寄存器) 级别1、2或3后, 该位不能被修改。</p>
位7	<p><b>TI1S:</b> TI1选择 (TI1 selection)</p> <p>0: TIMx_CH1管脚连到TI1输入</p> <p>1: TIMx_CH1、TIMx_CH2和TIMx_CH3管脚经异或后连到TI1输入</p>

位6: 4	<p><b>MMS[1: 0]:</b> 主模式选择 (Master mode selection)</p> <p>这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下:</p> <ul style="list-style-type: none"> <li>000: 复位 – TIMx_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位, 则TRGO上的信号相对实际的复位会有一个延迟。</li> <li>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)。</li> <li>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。</li> <li>011: 比较脉冲 – 一旦发生一次捕获或一次比较成功时, 当要设置CC1IF标志时 (即使它已经为高), 触发输出送出一个正脉冲 (TRGO)。</li> <li>100: 比较 – OC1REF信号被用于作为触发输出 (TRGO)</li> <li>101: 比较 – OC2REF信号被用于作为触发输出 (TRGO)</li> <li>110: 比较 – OC3REF信号被用于作为触发输出 (TRGO)</li> <li>111: 比较 – OC4REF信号被用于作为触发输出 (TRGO)</li> </ul>
位3	<p><b>CCDS:</b> 捕获/比较的DMA选择 (Capture/compare DMA selection)</p> <p>0: 当发生CCx事件时, 送出CCx的DMA请求 1: 当发生更新事件时, 送出CCx的DMA请求</p>
位2	<p><b>CCUS:</b> 捕获/比较控制更新选择 (Capture/compare control update selection)</p> <p>0: 如果捕获/比较控制位是预装载的 (CCPC = 1), 只能通过设置COM位更新它们 1: 如果捕获/比较控制位是预装载的 (CCPC = 1), 可以通过设置COM位或TRGI上的一个上升沿更新它们 注: 该位只对具有互补输出的通道起作用。</p>
位1	保留, 始终读为0。
位0	<p><b>CCPC:</b> 捕获/比较预装载控制位 (Capture/compare preloaded control)</p> <p>0: CCxE, CCxNE和OCxM位不是预装载的 1: CCxE, CCxNE和OCxM位是预装载的; 设置该位后, 它们只在设置了COM位后被更新 注: 该位只对具有互补输出的通道起作用。</p>

#### 11.4.3 从模式控制寄存器 (TIMx\_SMCR)

偏移地址: 0x08

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF		MSM		TS		保留	SMS				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 16	保留。
---------	-----

位15	<p><b>ETP:</b> 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</p>
位14	<p><b>ECE:</b> 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信号上的任意有效上升沿驱动 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。</p>
位13: 12	<p><b>ETPS[1: 0]:</b> 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</p>
位11: 8	<p><b>ETF[3: 0]:</b> 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f<sub>DTS</sub>采样 0001: 采样频率f<sub>SAMPLING</sub>=f<sub>CLOCK</sub>, N = 2 0010: 采样频率f<sub>SAMPLING</sub>=f<sub>CLOCK</sub>, N = 4 0011: 采样频率f<sub>SAMPLING</sub>=f<sub>CLOCK</sub>, N = 8 0100: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/2, N = 6 0101: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/2, N = 8 0110: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/4, N = 6 0111: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/4, N = 8 1000: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/8, N = 6 1001: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/8, N = 8 1010: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 5 1011: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 6 1100: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 8 1101: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 5 1110: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 6 1111: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 8</p>
位7	<p><b>MSM:</b> 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</p>

位6: 4	<p><b>TS[2: 0]: 触发选择 (Trigger selection)</b> 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) 001: 内部触发1 (ITR1) 010: 内部触发2 (ITR2) 011: 内部触发3 (ITR3) 100: TI1的边沿检测器 (TI1F_ED) 101: 滤波后的定时器输入1 (TI1FP1) 110: 滤波后的定时器输入2 (TI2FP2) 111: 外部触发输入 (ETRF)</p> <p>更多有关ITRx的细节, 参见下表。</p> <p>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>
位3	保留, 始终读为0。
位2: 0	<p><b>SMS: 从模式选择 (Slave mode selection)</b> 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 – 根据另一个输入的电平, 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> <p>注: 如果TI1F_EN被选为触发输入 (TS = 100) 时, 不要使用门控模式。这是因为, TI1F_ED在每次TI1F变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。</p>

表 25. TIMx 内部触发连接

从定时器	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM4	TIM5	TIM2	TIM3
TIM2	TIM4	TIM1	TIM6	TIM3

#### 11.4.4 DMA/中断使能寄存器 (TIMX\_DIER)

偏移地址: 0x0C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	COM DE	CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE	BIE	TIE	COM IE	CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 15	保留
位14	<b>TDE:</b> 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
位13	<b>COMDE:</b> 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
位12	<b>CC4DE:</b> 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
位11	<b>CC3DE:</b> 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
位10	<b>CC2DE:</b> 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
位9	<b>CC1DE:</b> 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
位8	<b>UDE:</b> 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
位7	<b>BIE:</b> 允许刹车中断 (Break interrupt enable) 0: 禁止刹车中断 1: 允许刹车中断
位6	<b>TIE:</b> 触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断 1: 使能触发中断
位5	<b>COMIE:</b> 允许COM中断 (COM interrupt enable) 0: 禁止COM中断 1: 允许COM中断

位4	<b>CC4IE:</b> 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
位3	<b>CC3IE:</b> 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
位2	<b>CC2IE:</b> 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
位1	<b>CC1IE:</b> 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
位0	<b>UIE:</b> 允许更新中断 (Update interrupt enable) 0: 禁止更新中断 1: 允许更新中断

#### 11.4.5 状态寄存器 (TIMx\_SR)

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4 OF	CC3 OF	CC2 OF	CC1 OF	保留	BIF	TIF	COM IF	CC4 IF	CC3 IF	CC2 IF	CC1 IF	UIF		
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 13	保留
位12	<b>CC4OF:</b> 捕获/比较4重复捕获标记 (Capture/Compare 4 overcapture flag) 参见CC1OF描述。
位11	<b>CC3OF:</b> 捕获/比较3重复捕获标记 (Capture/Compare 3 overcapture flag) 参见CC1OF描述。
位10	<b>CC2OF:</b> 捕获/比较2重复捕获标记 (Capture/Compare 2 overcapture flag) 参见CC1OF描述。
位9	<b>CC1OF:</b> 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。
位8	保留，始终读为0。
位7	<b>BIF:</b> 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

位6	<b>TIF:</b> 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，或或门控模式下的任一边沿）时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生 1: 触发器中断等待响应
位5	<b>COMIF:</b> 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
位4	<b>CC4IF:</b> 捕获/比较4中断标记 (Capture/Compare 4 interrupt flag) 参考CC1IF描述。
位3	<b>CC3IF:</b> 捕获/比较3中断标记 (Capture/Compare 3 interrupt flag) 参考CC1IF描述。
位2	<b>CC2IF:</b> 捕获/比较2中断标记 (Capture/Compare 2 interrupt flag) 参考CC1IF描述。
位1	<b>CC1IF:</b> 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 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）
位0	<b>UIF:</b> 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: 更新事件等待响应。当寄存器被更新时该位由硬件置‘1’： <ul style="list-style-type: none"><li>- 若TIMx_CR1寄存器的UDIS = 0，当REP_CNT = 0时产生更新事件（重复向下计数器上溢或下溢时）</li><li>- 若TIMx_CR1寄存器的UDIS = 0、URS = 0，当TIMx_EGR寄存器的UG = 1时产生更新事件（软件对计数器CNT重新初始化）</li><li>- 若TIMx_CR1寄存器的UDIS = 0、URS = 0，当计数器CNT被触发事件重初始化时产生更新事件。（参考同步控制寄存器的说明）</li></ul>

### 11.4.6 事件产生寄存器 (TIMx\_EGR)

偏移地址: 0x14

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							保留	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								W	W	W	W	W	W	W	W

位31: 8	保留
位7	<b>BG:</b> 产生刹车事件 (Break generation) 该位由软件置‘1’，用于产生一个刹车事件，由硬件自动清‘0’。 0: 无动作 1: 产生一个刹车事件。此时MOE = 0、BIF = 1，若开启对应的中断和DMA，则产生相应的中断和DMA
位6	<b>TG:</b> 产生触发事件 (Trigger generation) 该位由软件置‘1’，用于产生一个刹车事件，由硬件自动清‘0’。 0: 无动作 1: 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
位5	<b>COMG:</b> 捕获/比较事件，产生控制更新 (Capture/Compare control update generation) 该位由软件置‘1’，由硬件自动清‘0’。 0: 无动作 1: 当CCPC = 1，允许更新CCxE、CCxNE、OCxM位 注：该位只对拥有互补输出的通道有效。
位4	<b>CC4G:</b> 产生捕获/比较4事件 (Capture/Compare 4 generation) 参考CC1G描述。
位3	<b>CC3G:</b> 产生捕获/比较3事件 (Capture/Compare 3 generation) 参考CC1G描述。
位2	<b>CC2G:</b> 产生捕获/比较2事件 (Capture/Compare 2 generation) 参考CC1G描述。
位1	<b>CC1G:</b> 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。

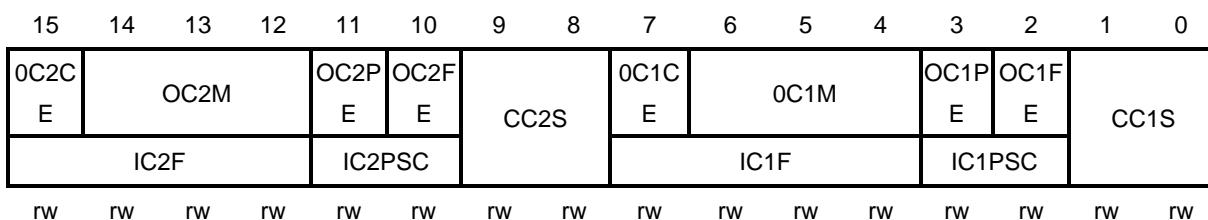
位0	<p><b>UG:</b> 产生更新事件 (Update generation) 该位由软件置'1'，由硬件自动清'0'。</p> <p>0: 无动作 1: 重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清'0'（但是预分频系数不变）。若在中心对称模式下或DIR = 0（向上计数）则计数器被清'0'；若DIR = 1（向下计数）则计数器取TIMx_ARR的值。</p>
----	---

#### 11.4.7 捕捉/比较模式寄存器 1 (TIMx\_CCMR1)

偏移地址: 0x18

复位值: 0x0000

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的CCxS定义。该寄存器其他位的作用和输出模式下不同。OCxx描述了通道在输出模式下的功能，ICxx描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。



输出比较模式:

位15	<b>OC2CE:</b> 输出比较2清0使能 (Output compare 2 clear enable)
位14: 12	<b>OC2M[2: 0]:</b> 输出比较2模式 (Output compare 2 mode)
位11	<b>OC2PE:</b> 输出比较2预装载使能 (Output compare 2 preload enable)
位10	<b>OC2FE:</b> 输出比较2快速使能 (Output compare 4 fast enable)
位9: 8	<p><b>CC2S[1: 0]:</b> 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：</p> <p>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>
位7	<p><b>OC1CE:</b> 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 1: 一旦检测到ETRF输入高电平，清除OC1REF = 0</p>

位6: 4	<p><b>OC1M[2: 0]:</b> 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、OC1N的值。</p> <p>OC1REF是高电平有效，而OC1、OC1N的有效电平取决于CC1P、CC1NP位。</p> <p>000: 冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用</p> <p>001 : 匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时，强制OC1REF为高</p> <p>010 : 匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时，强制OC1REF为低</p> <p>011: 翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平</p> <p>100: 强制为无效电平。强制OC1REF为低</p> <p>101: 强制为有效电平。强制OC1REF为高</p> <p>110: PWM模式1 - 在向上计数时，一旦TIMx_CNT &lt; TIMx_CCR1时通道1为有效电平，否则为 无效电平；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>
位3	<p><b>OC1PE:</b> 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>
位2	<p><b>OC1FE:</b> 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。OCFE只在通道被配置成PWM1或PWM2模式时起作用</p>

位1: 0	<p><b>CC1S[1: 0]:</b> 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：</p> <ul style="list-style-type: none"> <li>00: CC1通道被配置为输出</li> <li>01: CC1通道被配置为输入，IC1映射在TI1上</li> <li>10: CC1通道被配置为输入，IC1映射在TI2上</li> <li>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)</li> </ul> <p>注: CC1S仅在通道关闭时 (TIMx_CCER寄存器的CC1E = 0) 才是可写的。</p>
-------	--

### 输入捕获模式:

位15: 12	<b>IC2F[3: 0]:</b> 输入捕获2滤波器 (Input capture 2 filter)
位11: 10	<b>IC2PSC[1: 0]:</b> 输入/捕获2预分频器 (Input capture 2 prescaler)
位9: 8	<p><b>CC2S[1: 0]:</b> 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：</p> <ul style="list-style-type: none"> <li>00: CC2通道被配置为输出</li> <li>01: CC2通道被配置为输入，IC2映射在TI2上</li> <li>10: CC2通道被配置为输入，IC2映射在TI1上</li> <li>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)</li> </ul> <p>注: CC2S仅在通道关闭时 (TIMx_CCER寄存器的CC2E = 0) 才是可写的。</p>
位7: 4	<p><b>IC1F[3: 0]:</b> 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：</p> <ul style="list-style-type: none"> <li>0000: 无滤波器，以f<sub>DTS</sub>采样</li> <li>1000: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/8, N = 6</li> <li>0001: 采样频率f<sub>SAMPLING</sub>=f<sub>CCK_INT</sub>, N = 2</li> <li>1001: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/8, N = 8</li> <li>0010: 采样频率f<sub>SAMPLING</sub>=f<sub>CCK_INT</sub>, N = 4</li> <li>1010: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 5</li> <li>0011: 采样频率f<sub>SAMPLING</sub>=f<sub>CCK_INT</sub>, N = 8</li> <li>1011: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 6</li> <li>0100: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/2, N = 6</li> <li>1100: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/16, N = 8</li> <li>0101: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/2, N = 8</li> <li>1101: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 5</li> <li>0110: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/4, N = 6</li> <li>1110: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 6</li> <li>0111: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/4, N = 8</li> <li>1111: 采样频率f<sub>SAMPLING</sub>=f<sub>DTS</sub>/32, N = 8</li> </ul>

位3: 2	<b>IC1PSC[1: 0]:</b> 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
位1: 0	<b>CC1S[1: 0]:</b> 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。

#### 11.4.8 捕捉/比较模式寄存器 2 (TIMx\_CCMR2)

偏移地址: 0x1C

复位值: 0x0000

参看以上 CCMR1 寄存器的描述。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C4CE	OC4M	OC4PE	OC4FE	CC4S	0C3CE	OC3M	OC3PE	OC3FE	CC3S						
	IC4F	IC4PSC			IC3F	IC3PSC									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### 输出比较模式

位15	<b>OC4CE:</b> 输出比较4清0使能 (Output compare 4 clear enable)
位14: 12	<b>OC4M[2: 0]:</b> 输出比较4模式 (Output compare 4 mode)
位11	<b>OC4PE:</b> 输出比较4预装载使能 (Output compare 4 preload enable)
位10	<b>OC4FE:</b> 输出比较4快速使能 (Output compare 4 fast enable)
位9: 8	<b>CC4S[1: 0]:</b> 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。
位7	<b>OC3CE:</b> 输出比较3清0使能 (Output compare 3 clear enable)
位6: 4	<b>OC3M[2: 0]:</b> 输出比较3模式 (Output compare 3 mode)

位3	<b>OC3PE:</b> 输出比较3预装载使能 (Output compare 3 preload enable)
位2	<b>OC3FE:</b> 输出比较3快速使能 (Output compare 3 fast enable)
位1: 0	<p><b>CC3S[1: 0]:</b> 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)</p> <p>注: CC3S仅在通道关闭时 (TIMx_CCER寄存器的CC3E = 0) 才是可写的。</p>

### 输入比较模式

位15: 12	<b>IC4F[3: 0]:</b> 输入捕获4滤波器 (Input capture 4 filter)
位11: 10	<b>IC4PSC[1: 0]:</b> 输入/捕获4预分频器 (Input capture 4 prescaler)
位9: 8	<p><b>CC4S[1: 0]:</b> 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)</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>
位7: 4	<b>IC3F[3: 0]:</b> 输入捕获3滤波器 (Input capture 3 filter)
位3: 2	<b>IC3PSC[1: 0]:</b> 输入/捕获3预分频器 (Input capture 3 prescaler)
位1: 0	<p><b>CC3S[1: 0]:</b> 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)</p> <p>注: CC3S仅在通道关闭时 (TIMx_CCER寄存器的CC3E = 0) 才是可写的。</p>

### 11.4.9 捕捉/比较使能寄存器 (TIMx\_CCER)

偏移地址: 0x20

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4P	CC4E	CC3NP P	CC3NE E	CC3P	CC3E	CC2NP P	CC2NE E	CC2P	CC1E	CC1NP P	CC1NE E	CC1P	CC1E	

W      W      W      W      W      W      W      W      W      W      W      W      W      W      W      W

位15: 14	保留, 始终读为0。
位13	<b>CC4P:</b> 输入/捕获4输出极性 (Capture/Compare 4 output polarity) 参考CC1P的描述。
位12	<b>CC4E:</b> 输入/捕获4输出使能 (Capture/Compare 4 output enable) 参考CC1E 的描述。
位11	<b>CC3NP:</b> 输入/捕获3互补输出极性 (Capture/Compare 3 complementary output polarity) 参考CC1NP的描述。
位10	<b>CC3NE:</b> 输入/捕获3互补输出使能 (Capture/Compare 3 complementary output enable) 参考CC1NE的描述。
位9	<b>CC3P:</b> 输入/捕获3输出极性 (Capture/Compare 3 output polarity) 参考CC1P的描述。
位8	<b>CC3E:</b> 输入/捕获3输出使能 (Capture/Compare 3 output enable) 参考CC1E 的描述。
位7	<b>CC2NP:</b> 输入/捕获2互补输出极性 (Capture/Compare 2 complementary output polarity) 参考CC1NP的描述。
位6	<b>CC2NE:</b> 输入/捕获2互补输出使能 (Capture/Compare 2 complementary output enable) 参考CC1NE的描述。
位5	<b>CC2P:</b> 输入/捕获2输出极性 (Capture/Compare 2 output polarity) 参考CC1P的描述。
位4	<b>CC2E:</b> 输入/捕获2输出使能 (Capture/Compare 2 output enable) 参考CC1E的描述。
位3	<b>CC1NP:</b> 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LCCK位) 设为3或2且CC1S = 00 (通道配置为输出) 则该位不能被修改。
位2	<b>CC1NE:</b> 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的输出电平依赖于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

位1	<p><b>CC1P:</b> 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 1: OC1低电平有效</p> <p>CC1通道配置为输入:</p> <p>该位选择是IC1还是IC1的反相信号作为触发或捕获信号。</p> <p>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</p> <p>注: 一旦LOCK级别 (TIMx_BDTR寄存器中的LCCK位) 设为3或2, 则该位不能被修改。</p>
位0	<p><b>CC1E:</b> 输入/捕获1输出使能 (Capture/Compare 1 output enable)</p> <p>CC1通道配置为输出:</p> <p>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</p> <p>CC1通道配置为输入:</p> <p>该位决定了计数器的值是否能捕获入TIMx_CCR1寄存器。</p> <p>0: 捕获禁止 1: 捕获使能</p>

表 26. 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 <sup>(1)</sup>	
MOE位	OSSI位	OSSR位	CCxE位	CCxNE位	OCx 输出状态	OCxN 输出状态
1	X	0	0	0	输出禁止 (与定时器断开) OCx = 0, OCx_EN = 0	输出禁止 (与定时器断开) OCxN = 0, OCxN_EN = 0
		0	0	1	输出禁止 (与定时器断开) OCx = 0, OCx_EN = 0	OCxREF + 极性, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		0	1	0	OCxREF + 极性, OCx = OCxREF xor CCxP, OCx_EN = 1	输出禁止 (与定时器断开) OCxN = 0, OCxN_EN = 0
		0	1	1	OCxREF + 极性 + 死区, OCx_EN=1	OCxREF反相 + 极性 + 死区, OCxN_EN = 1
		1	0	0	输出禁止 (与定时器断开) OCx = CCxP, OCx_EN = 0	输出禁止 (与定时器断开) OCxN = CCxNP, OCxN_EN = 0
		1	0	1	关闭状态 (输出使能且为无效电平) OCx = CCxP, OCx_EN = 1	OCxREF + 极性, OCxN = OCxREF xor CCxNP, OCxN_EN = 1
		1	1	0	OCxREF + 极性, OCx = OCxREF xor CCxP, OCx_EN = 1	关闭状态 (输出使能且为无效电平) OCxN = CCxNP, OCxN_EN = 1
		1	1	1	OCxREF + 极性 + 死区, OCx_EN = 1	OCxREF反相 + 极性 + 死区, OCxN_EN = 1
0	0	X	0	0	输出禁止 (与定时器断开)	

控制位					输出状态 <sup>(1)</sup>														
MOE位	OSSI位	OSSR位	CCxE位	CCxNE位	OCx 输出状态					OCxN 输出状态									
0	0		0	1	异步地: OCx = CCxP, OCx_EN = 0, OCxN = CCxNP, OCxN_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		1	0															
	0		1	1															
	1		0	0	关闭状态 (输出使能且为无效电平) 异步地: OCx = CCxP, OCx_EN = 1, OCxN = CCxNP, OCxN_EN = 1; 若时钟存在: 经过一个死区时间后 OCx = OISx, OCxN = OISxN, 假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。														
	1		0	1															
	1		1	0															
	1		1	1															

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: 管脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 管脚的状态, 取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

#### 11.4.10 计数器 (TIMx\_CNT)

偏移地址: 0x24

复位值: 0x0000

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

位31: 0	CNT[31: 0]: 计数器的值 (Counter value)
--------	-----------------------------------

#### 11.4.11 预分频器 (TIMx\_PSC)

偏移地址: 0x28

复位值: 0x0000

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

位15: 0	<b>PSC[15: 0]:</b> 预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 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'。
--------	---

#### 11.4.12 自动装载寄存器 (TIMx\_ARR)

偏移地址: 0x2C

复位值: 0x0000

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

位31: 0	ARR[31: 0]: 自动重装载的值 (Prescaler value) ARR包含了将要装载入实际的自动重装载寄存器的数值。 详细参考13.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。
--------	--

#### 11.4.13 重复计数寄存器 (TIMx\_RCR)

偏移地址: 0x30

复位值: 0x0000

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

位15: 8	保留，始终读为0。
位7: 0	<b>REP[7: 0]:</b> 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

#### 11.4.14 捕获/比较寄存器 1 (TIMx\_CCR1)

偏移地址: 0x34

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0

**CCR1[31: 0]:** 捕获/比较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。

#### 11.4.15 捕获/比较寄存器 2 (TIMx\_CCR2)

偏移地址: 0x38

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0

**CCR2[31: 0]:** 捕获/比较2的值 (Capture/Compare 2 value)

若CC2通道配置为输出:

CCR2包含了装入当前捕获/比较2寄存器的值（预装载值）。

如果在TIMx\_CCMR2 寄存器 (OC2PE位) 中未选择预装载特性, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较2寄存器中。当前捕获/比较寄存器参与同计数器 TIMx\_CNT 的比较, 并在OC2端口上产生输出信号。

若CC2通道配置为输入:

CCR2包含了由上一次输入捕获2事件 (IC2) 传输的计数器值。

### 11.4.16 捕获/比较寄存器 3 (TIMx\_CCR3)

偏移地址: 0x3C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0

**CCR3[31: 0]:** 捕获/比较3的值 (Capture/Compare 3 value)

若CC3通道配置为输出:

CCR3包含了装入当前捕获/比较3寄存器的值（预装载值）。

如果在TIMx\_CCMR3寄存器 (OC3PE位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较3寄存器中。当前捕获/比较寄存器参与同计数器TIMx\_CNT的比较，并在OC3端口上产生输出信号。

若CC3通道配置为输入:

CCR3包含了由上一次输入捕获3事件 (IC3) 传输的计数器值。

### 11.4.17 捕获/比较寄存器 4 (TIMx\_CCR4)

偏移地址: 0x40

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0

**CCR4[31: 0]:** 捕获/比较4的值 (Capture/Compare 4 value)

若CC4通道配置为输出:

CCR4包含了装入当前捕获/比较4寄存器的值（预装载值）。

如果在TIMx\_CCMR4 寄存器 (OC4PE位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较4寄存器中。当前捕获/比较寄存器参与同计数器TIMx\_CNT的比较，并在OC4端口上产生输出信号。

若CC4通道配置为输入:

CCR4包含了由上一次输入捕获4事件 (IC4) 传输的计数器值。

### 11.4.18 刹车和死区寄存器 (TIMx\_BDTR)

偏移地址: 0x44

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK									DTG
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

注: 根据锁定设置, AOE、BKP、BKE、OSSI、OSSR 和 DTG[7: 0]位均可被写保护, 有必要在第一次写入 TIMx\_BDTR 寄存器时对它们进行配置。

位15	<b>MOE:</b> 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15.4.9节, 捕获/比较使能寄存器 (TIMx_CCER)。
位14	<b>AOE:</b> 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。
位13	<b>BKP:</b> 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。
位12	<b>BKE:</b> 刹车功能使能 (Break enable) 0: 禁止刹车输入 (BRK及BRK_ACTH) 1: 开启刹车输入 (BRK及BRK_ACTH) 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。
位11	<b>OSSR:</b> 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 (12.4.9节, 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。
位10	<b>OSSI:</b> 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 (15.4.9节, 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。

位9: 8	<p><b>LOCK[1: 0]:</b> 锁定设置 (Lock configuration)      该位为防止软件错误而提供写保护。</p> <p>00: 锁定关闭, 寄存器无写保护      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。</p>
位7: 0	<p><b>UTG[7: 0]:</b> 死区发生器设置 (Dead-time generator setup)      这些位定义了插入互补输出之间的死区持续时间。假设DT表示其持续时间:</p> <p>DTG[7: 5] = 0xx =&gt; DT = DTG[7: 0] × Tdtg, Tdtg = TDTS;      DTG[7: 5] = 10x =&gt; DT = (64+DTG[5: 0]) × Tdtg, Tdtg = 2 × TDTS;      DTG[7: 5] = 110 =&gt; DT = (32+DTG[4: 0]) × Tdtg, Tdtg = 8 × TDTS; DTG[7: 5] = 111 =&gt; DT = (32+DTG[4: 0]) × Tdtg, Tdtg = 16 × TDTS; 例: 若TDTS = 125ns (8MHz), 可能的死区时间为:      0到15875nS, 若步长时间为125nS      16uS到31750nS, 若步长时间为250nS      32uS到63uS, 若步长时间为1uS      64uS到126uS, 若步长时间为2uS      注: 一旦LOCK级别 (TIMx_BDTR寄存器中的LOCK位) 设为1、2或3, 则不能修改这些位。</p>

#### 11.4.19 DMA 控制寄存器 (TIMx\_DCR)

偏移地址: 0x48

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	DBL				保留	DBA				rw	rw	rw	rw	rw	rw
位15: 13	保留, 始终读为0。														

	<p><b>DBL[4: 0]: DMA连续传送长度 (DMA burst length)</b>      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  <b>位12: 8</b>      - 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>(TIMx\_CR1\text{的地址}) + DBA + (\text{DMA索引})</math>, 其中 DMA索引 = DBL      其中 <math>(TIMx\_CR1\text{的地址}) + DBA</math> 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>(TIMx\_CR1\text{的地址}) + DBA</math> 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>
<b>位7: 5</b>	保留, 始终读为0。
<b>位4: 0</b>	<p><b>DBA[4: 0]: DMA基地址 (DMA base address)</b>      这些位定义了DMA在连续模式下的基地址 (当对TIMx_DMAR寄存器进行读或写时), DBA定义为从TIMx_CR1寄存器所在地址开始的偏移量:      00000: TIMx_CR1      00001: TIMx_CR2      00010: TIMx_SMCR      .....</p>

#### 11.4.20 连续模式的 DMA 地址 (TIMx\_DMAR)

偏移地址: 0x4C

复位值: 0x0000

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

<b>位15: 0</b>	<p><b>DMAB[15: 0]: DMA连续传送寄存器 (DMA register for burst accesses)</b>      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:  <math>\text{TIMx\_CR1地址} + DBA + \text{DMA索引}</math>, 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。</p>
---------------	--

## 12. 通用定时器 (TIM3/4)

### 12.1 TIMx 简介

通用定时器是一个通过可编程预分频器驱动的 32 位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

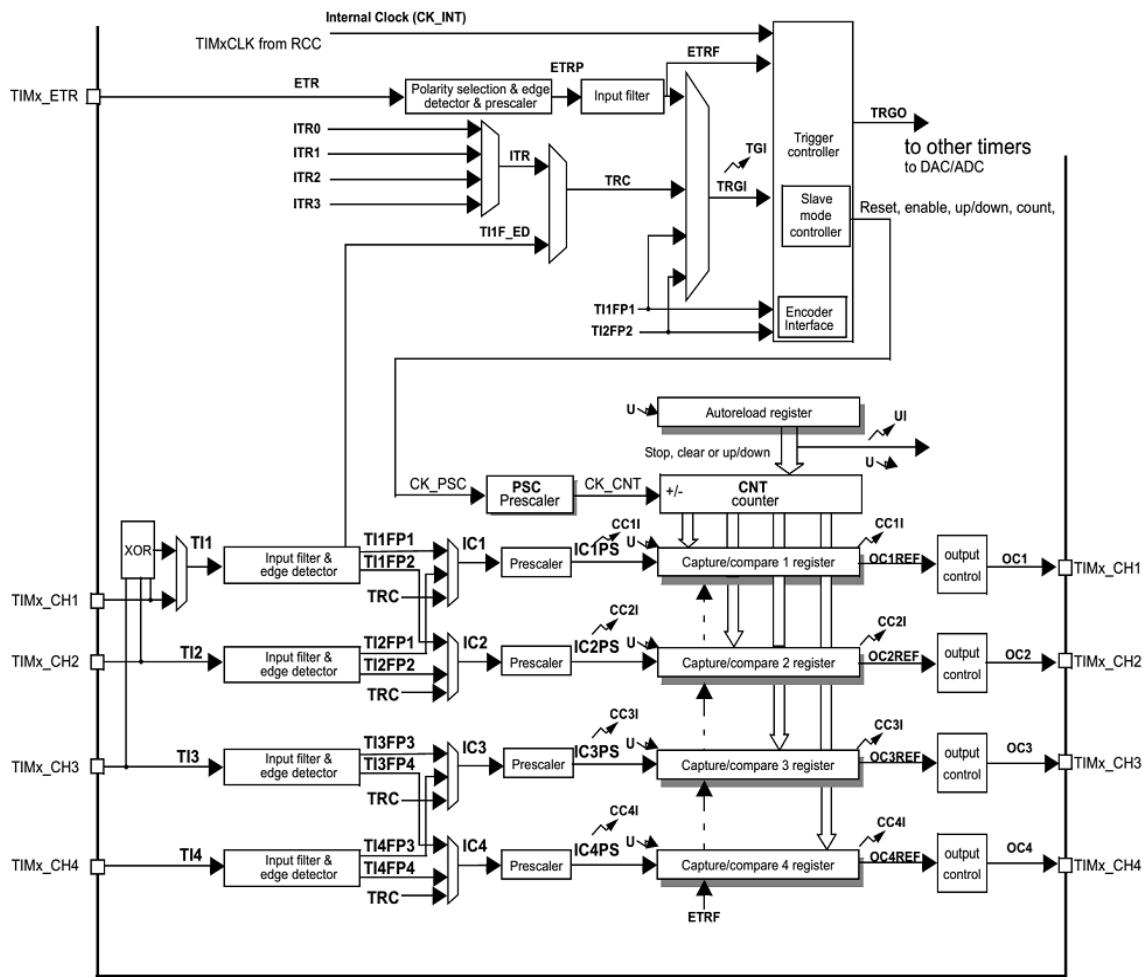
TIMx 定时器是完全独立的，而且没有互相共享任何资源。它们可以一起同步操作。

### 12.2 TIMx 主要功能

通用 TIMx (TIM3、TIM4) 定时器功能包括：

- 32 位向上、向下、向上/向下自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 4 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 生成（边缘或中间对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

图 64. 通用定时器框图



注: 根据控制位的设定, 在  $U$  事件时传送预加载寄存器的内容至工作寄存器

$U \searrow$  事件

$\nearrow$  中断和 DMA 输出

## 12.3 TIMX 功能描述

### 12.3.1 时基单元

可编程通用定时器的主要部分是一个 32 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写，时基单元包含：

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

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TIMX\_CR1 寄存器中的自动装载预装载使能位 (ARPE) 的设置，预装载寄存器的内容被立即或在 每次的更新事件 UEV

时传送到影子寄存器。当计数器达到溢出条件(向下计数时的下溢条件)并当 TIMX\_CR1 寄存器中的 UDIS 位等于 0 时,产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动,仅当设置了计数器 TIMX\_CR1 寄存器中的计数器使能位(CEN)时,CK\_CNT 才有效。(有关计数器使能的细节,请参见控制器的从模式描述)。

### 预分频器描述

预分频器可以将计数器的时钟频率按 1 ~ 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 32 位计数器。因为这个控制寄存器带有缓冲器,它能够在工作时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下面两个图分别给出了在预分频器运行时,更改计数器参数的例子。

图 65. 当预分频器的参数从 1 变到 2 时,计数器的时序图

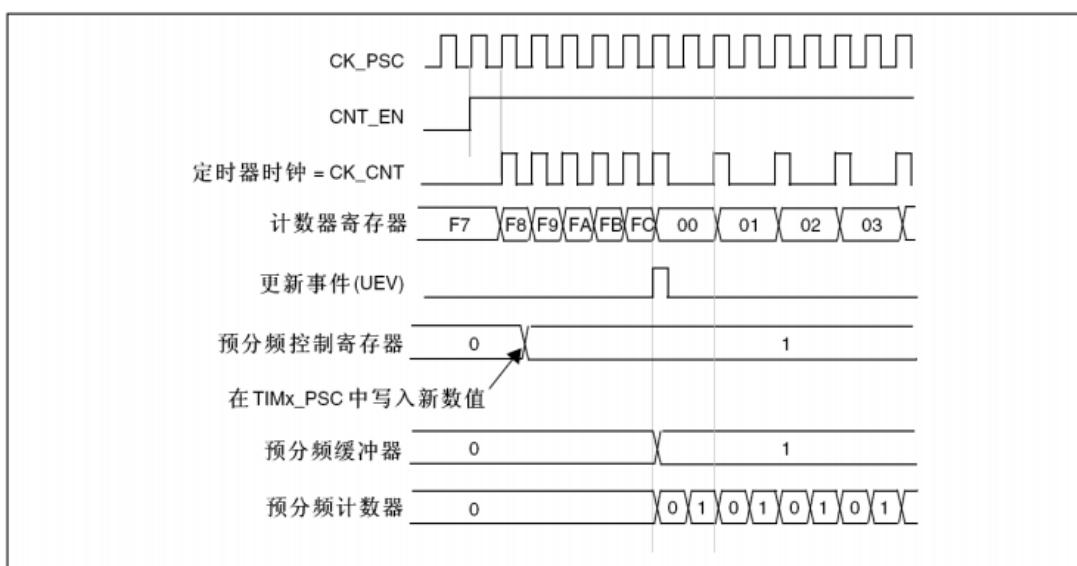
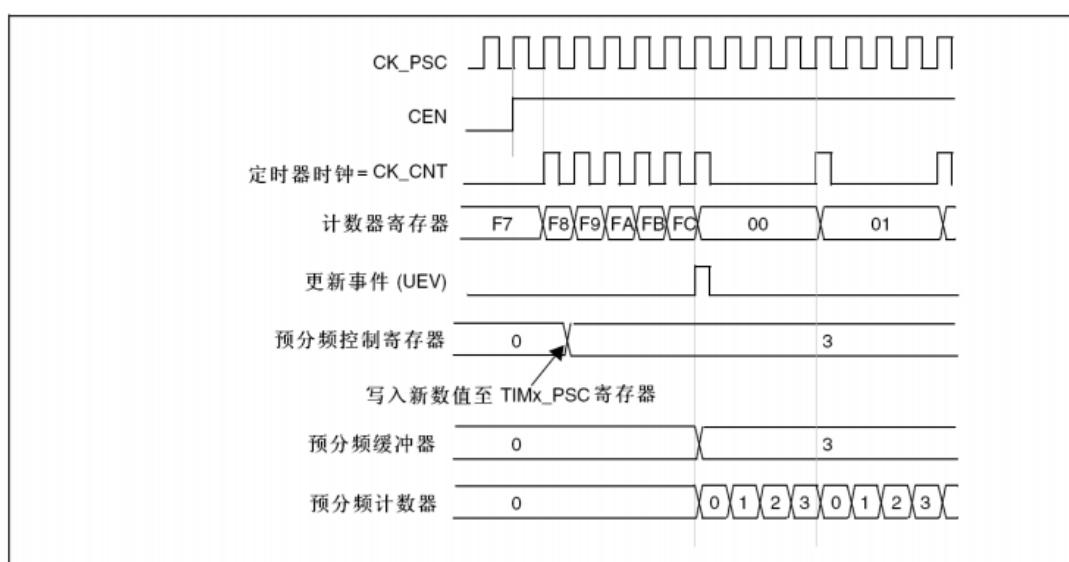


图 66. 当预分频器的参数从 1 变到 4 时,计数器的时序图



### 12.3.2 计数模式

#### 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（**TIMx\_ARR** 寄存器的内容），然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 **TIMx\_EGR** 寄存器中设置 **UG** 位（通过软件方式或者使用从模式控制器）也同样可以产生一个更新事件。

设置 **TIMx\_CR1** 寄存器中的 **UDIS** 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 **UDIS** 位被清 0 之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 **TIMx\_CR1** 寄存器中的 **URS** 位（选择更新请求），设置 **UG** 位将产生一个更新事件 **UEV**，但硬件不设置 **UIF** 标志（即不产生中断或 **DMA** 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

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

- 预分频器的缓冲区被置入预装载寄存器的值（**TIMx\_PSC** 寄存器的内容）
- 自动装载影子寄存器被重新置入预装载寄存器的值（**TIMx\_ARR**）

下图给出一些例子，当 **TIMx\_ARR = 0x36** 时计数器在不同时钟频率下的动作：

图 67. 计数器时序图，内部时钟分频因子为 1

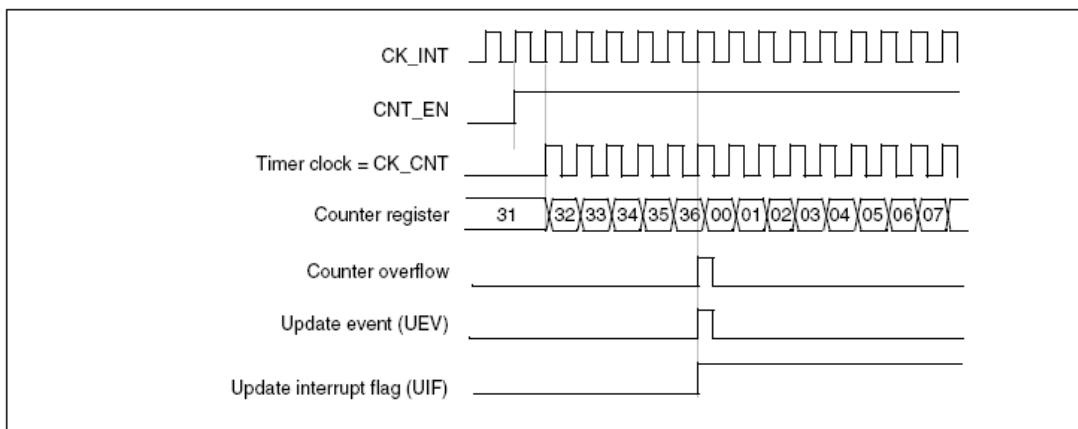


图 68. 计数器时序图，内部时钟分频因子为 2

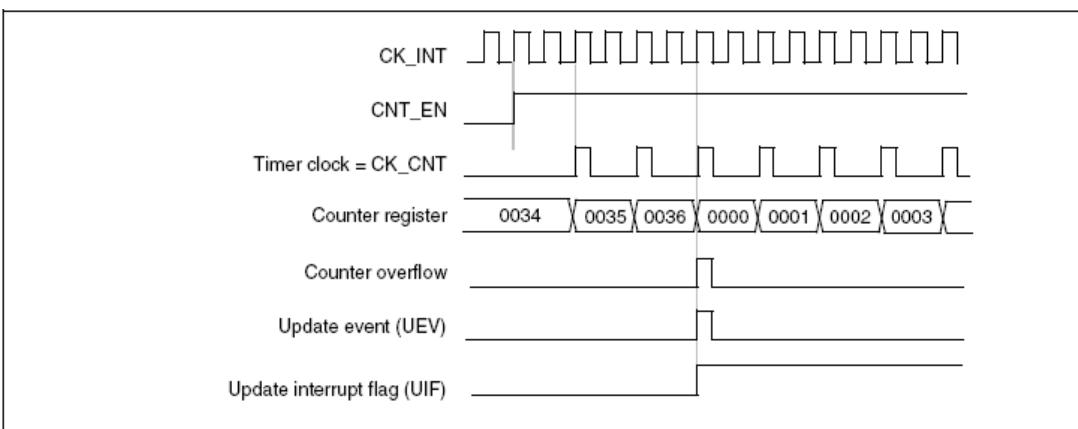


图 69. 计数器时序图, 内部时钟分频因子为 4

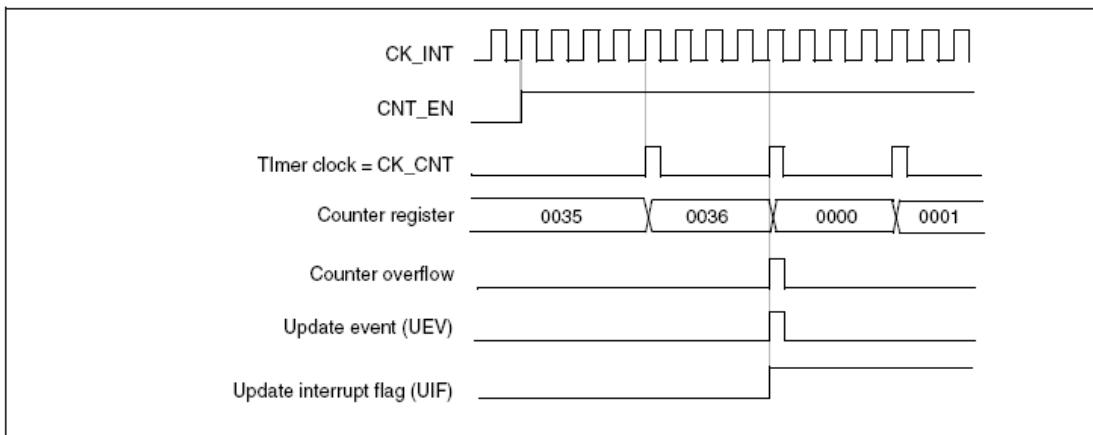


图 70. 计数器时序图, 内部时钟分频因子为 N

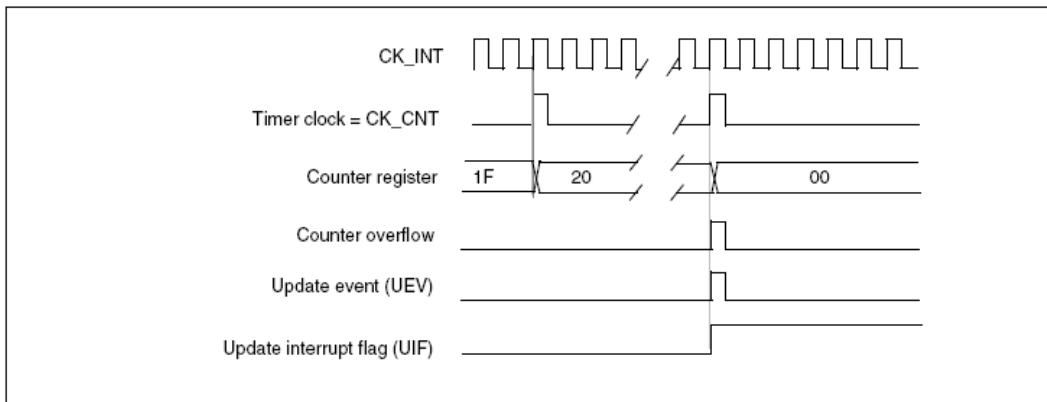


图 71. 计数器时序图, 当 ARPE = 0 时的更新事件 (TIMx\_ARR 没有预装入)

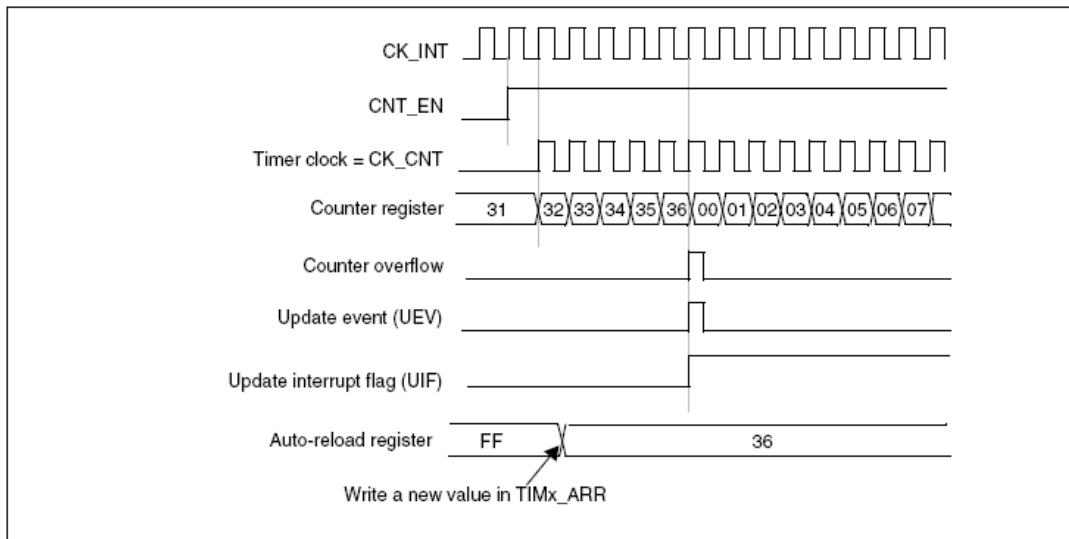
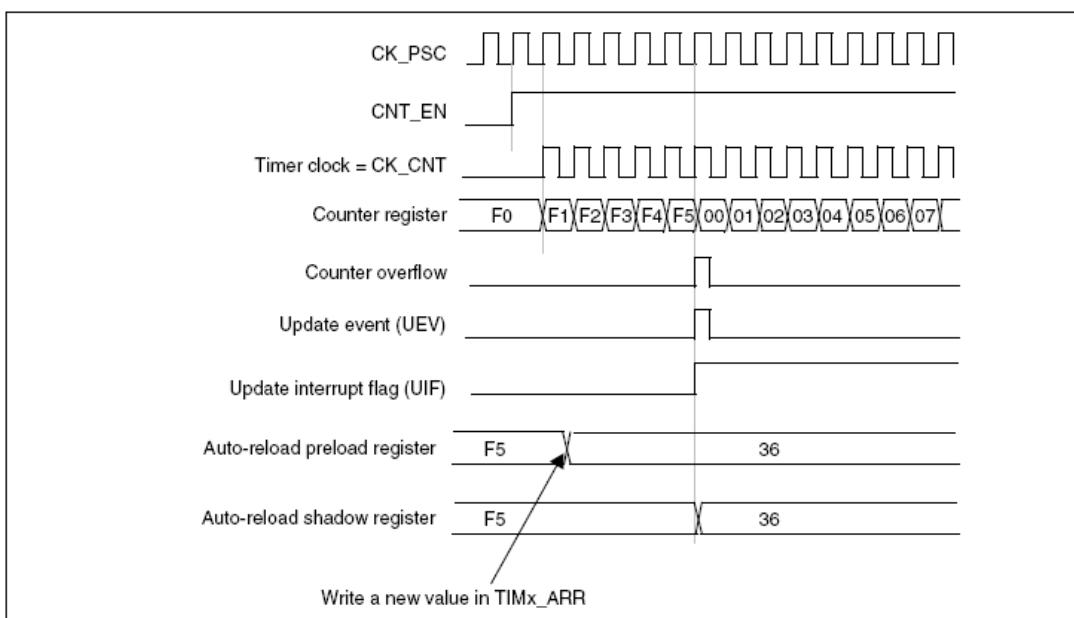


图 72. 计数器时序图, 当 ARPE = 1 时的更新事件 (预装入了 TIMx\_ARR)



### 向下计数模式

在向下模式中，计数器从自动装入的值（**TIMx\_ARR** 寄存器的值）开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 **TIMx\_EGR** 寄存器中设置 **UG** 位（通过软件方式或者使用从模式控制器）也同样可以产生一个更新事件。

设置 **TIMx\_CR1** 寄存器的 **UDIS** 位可以禁止 **UEV** 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 **UDIS** 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，同时预分频器的计数器重新从 0 开始（但预分频器的速率不能被修改）。

此外，如果设置了 **TIMx\_CR1** 寄存器中的 **URS** 位（选择更新请求），设置 **UG** 位将产生一个更新事件 **UEV** 但不设置 **UIF** 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 **URS** 位的设置）更新标志位（**TIMx\_SR** 寄存器中的 **UIF** 位）也被设置。

- 预分频器的缓存器被置入预装载寄存器的值（**TIMx\_PSC** 寄存器的值）。
- 当前的自动加载寄存器被更新为预装载值（**TIMx\_ARR** 寄存器中的内容）。注：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 **TIMx\_ARR** = 0x36 时，计数器在不同时钟频率下的操作实例：

图 73. 计数器时序图, 内部时钟分频因子为 1

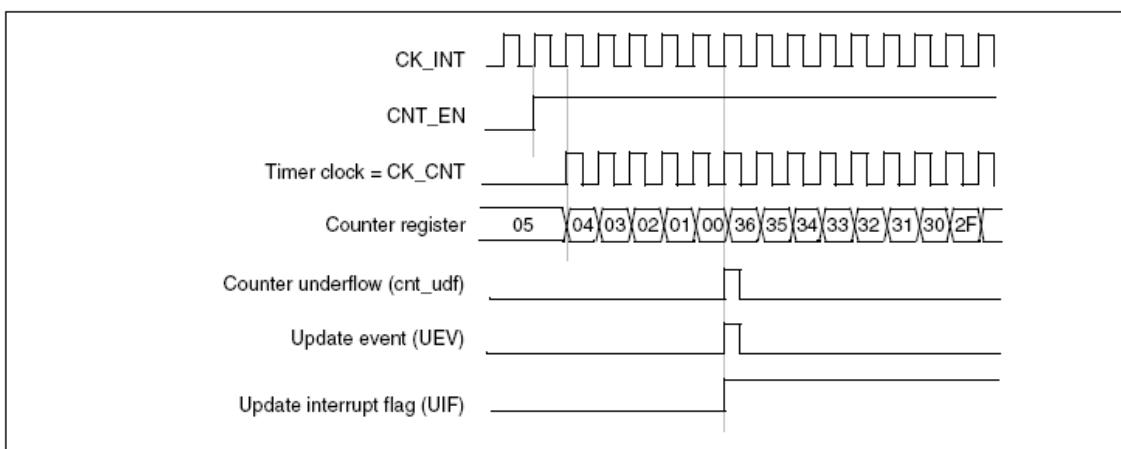


图 74. 计数器时序图, 内部时钟分频因子为 2

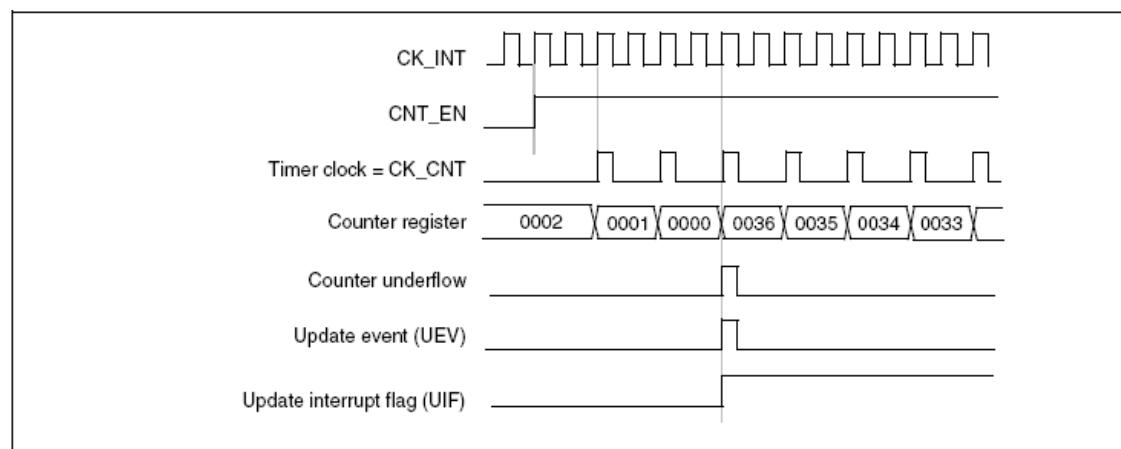


图 75. 计数器时序图, 内部时钟分频因子为 4

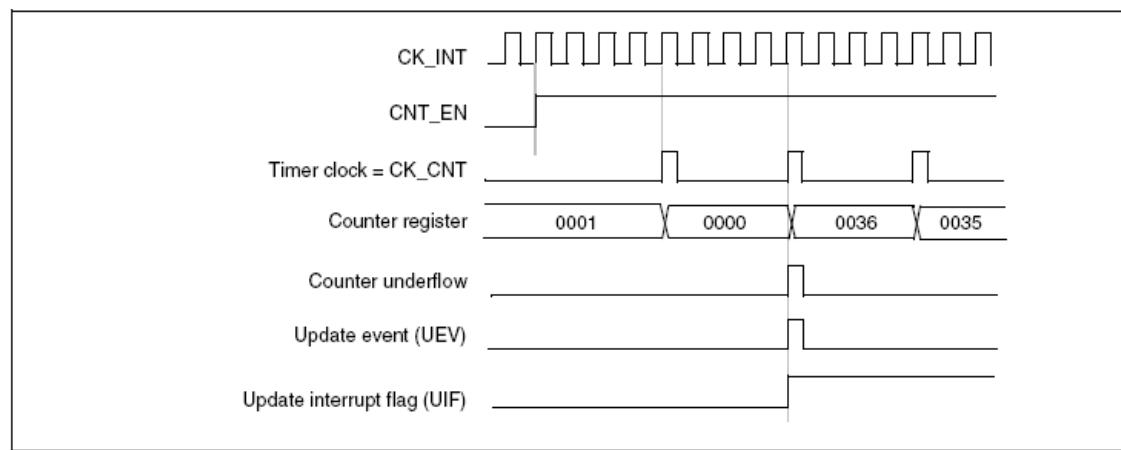


图 76. 计数器时序图，内部时钟分频因子为 N

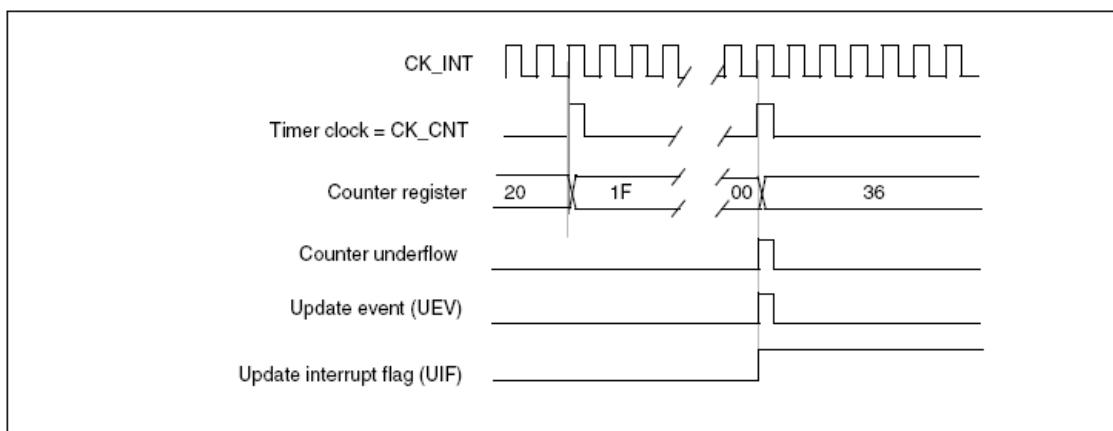
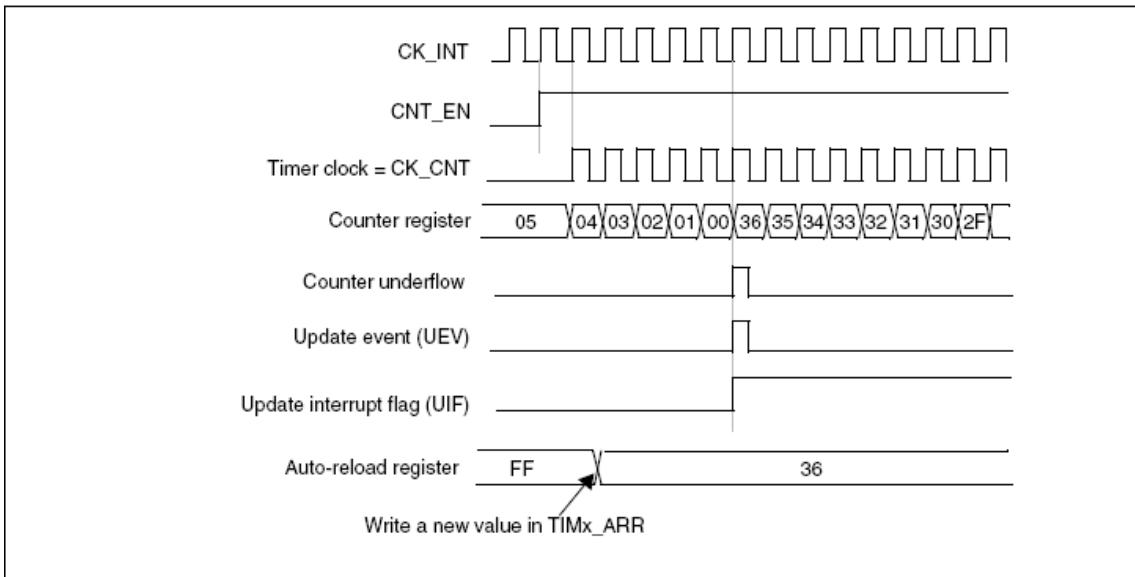


图 77. 计数器时序图，当没有使用重复计数器时的更新事件



### 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值 (TIMx\_ARR 寄存器) -1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式，不能写入 TIMx\_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。更新事件可以产生在每次计数溢出和每次计数下溢；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，此时，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx\_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位 (TIMx\_SR 寄存器中的 UIF 位) 也被设置。

- 预分频器的缓存器被加载为预装载（**TIMx\_PSC** 寄存器）的值。
- 当前的自动加载寄存器被更新为预装载值（**TIMx\_ARR** 寄存器中的内容）。注：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。

以下是一些计数器在不同时钟频率下的操作的例子：

图 78. 计数器时序图，内部时钟分频因子为 1，**TIMx\_ARR = 0x6**

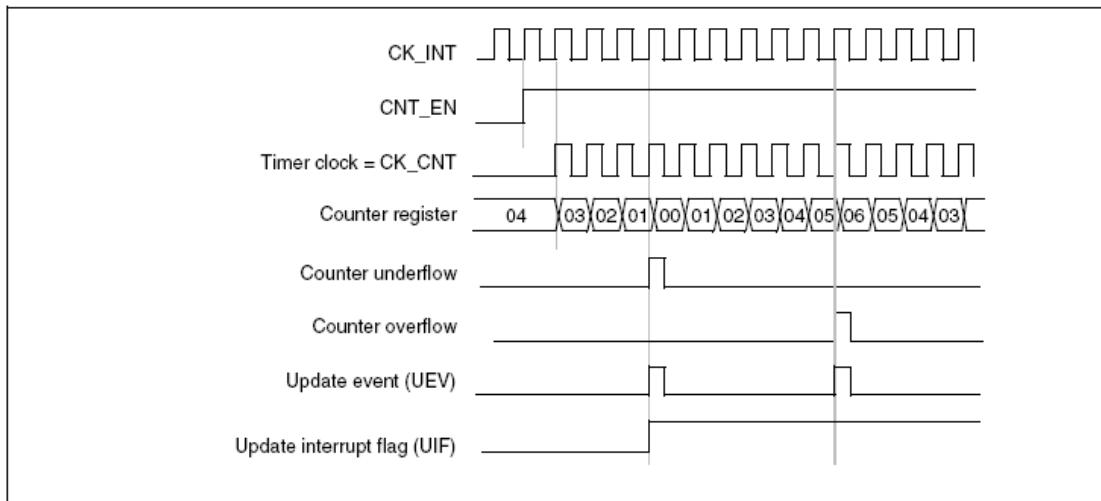


图 79. 计数器时序图，内部时钟分频因子为 2

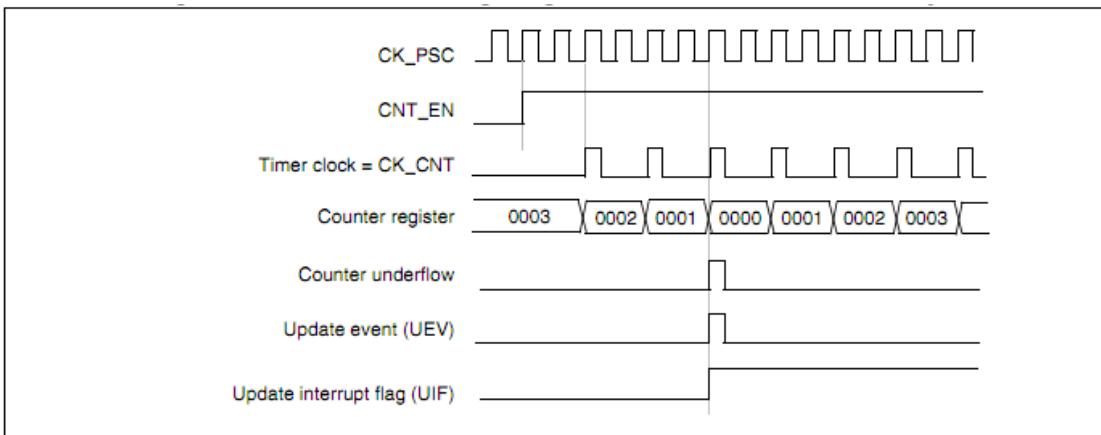


图 80. 计数器时序图, 内部时钟分频因子为 4, TIMx\_ARR = 0x36

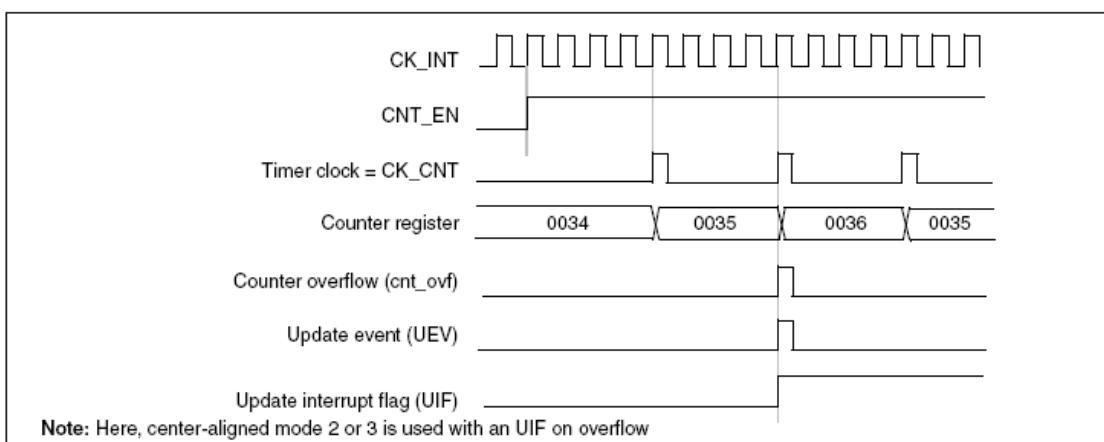


图 81. 计数器时序图, 内部时钟分频因子为 N

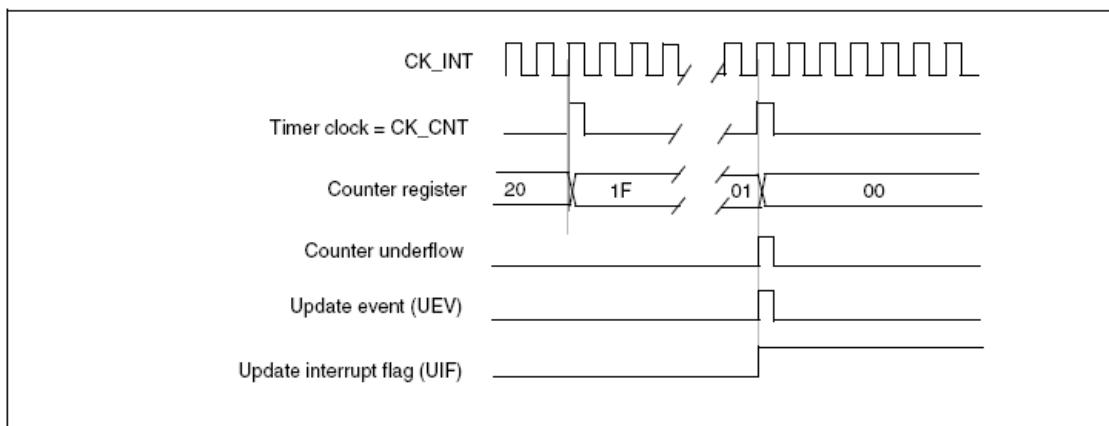


图 82. 计数器时序图, ARPE = 1 时的更新事件（计数器下溢）

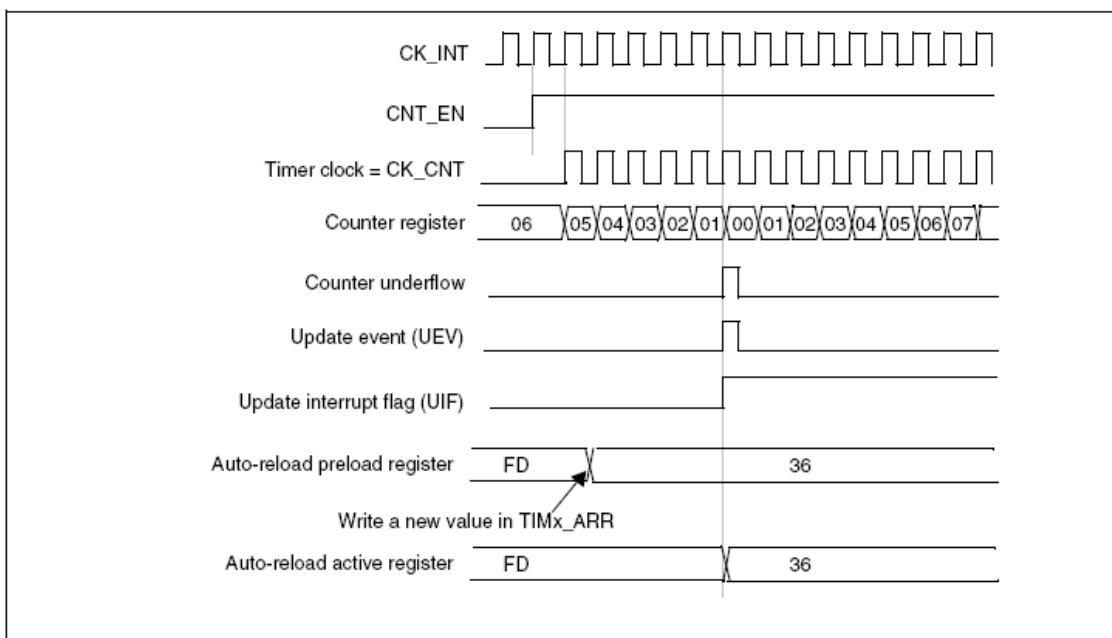
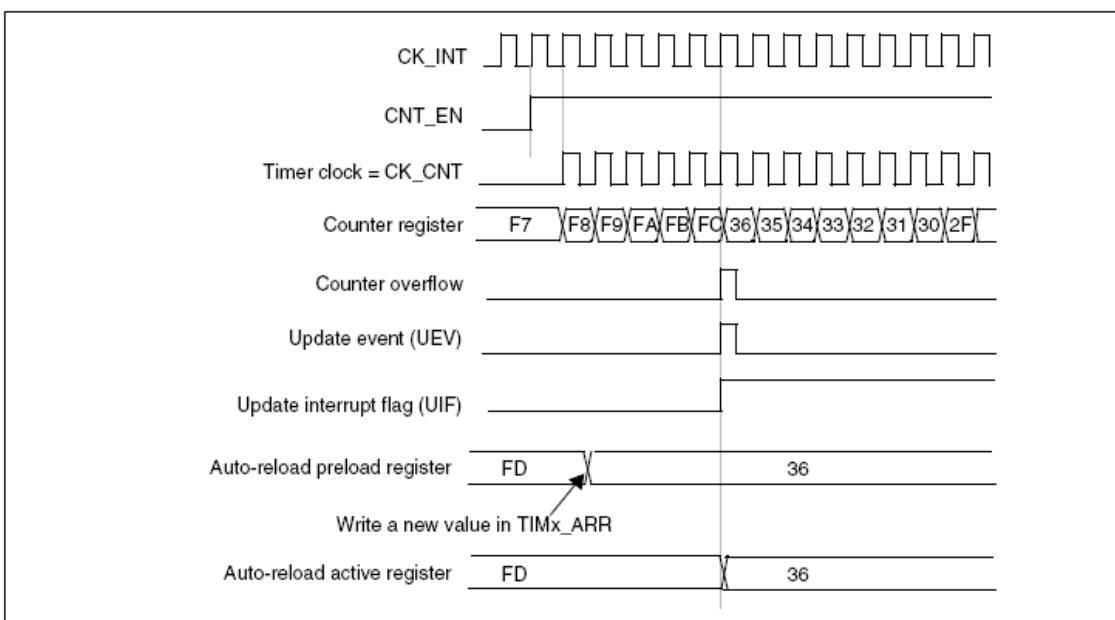


图 83. 计数器时序图, ARPE = 1 时的更新事件 (计数器溢出)



### 12.3.3 时钟选择

计数器时钟可由下列时钟源提供：

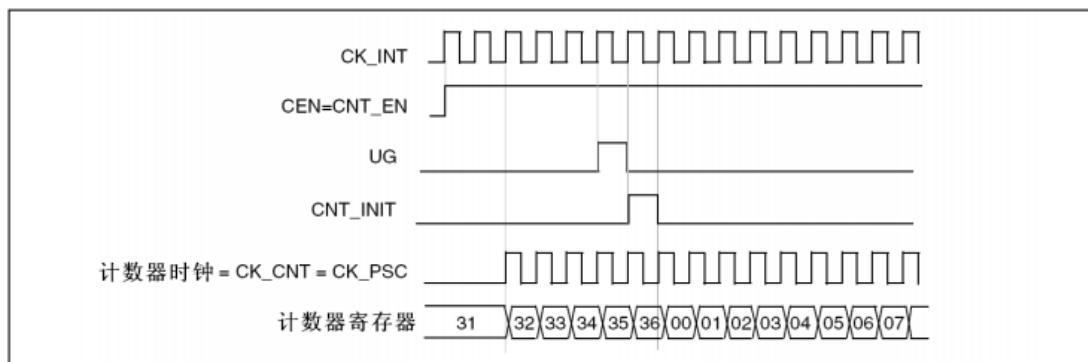
- 内部时钟 (**CK\_INT**)
- 外部时钟模式 1：外部输入脚 (**TIx**)
- 外部时钟模式 2：外部触发输入 (**ETR**)
- 内部触发输入 (**ITRx**)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

#### 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)。一旦 **CEN** 位被写成 1，预分频器的时钟就由内部时钟 **CK\_INT** 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

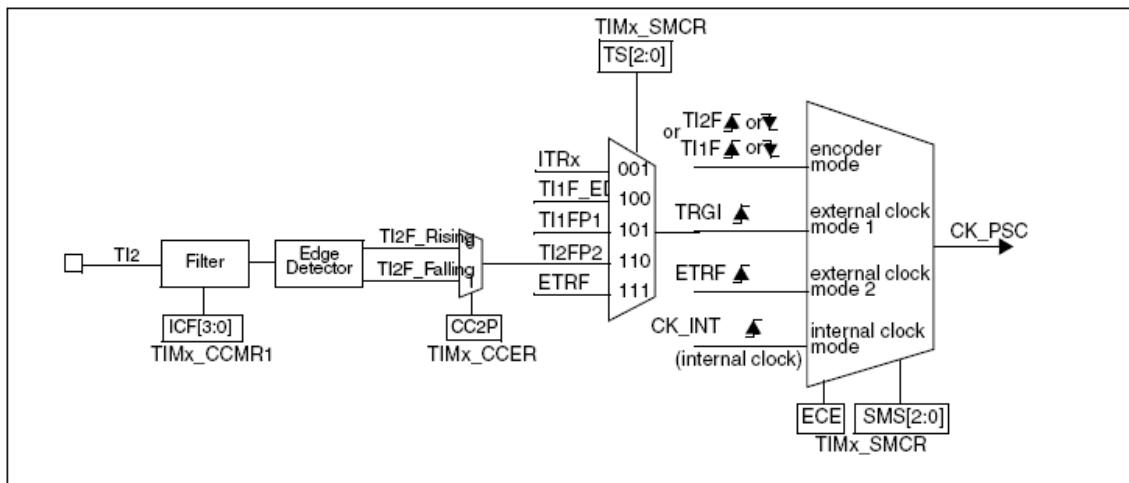
图 84. 一般模式下的控制电路，内部时钟分频因子为 1



## 外部时钟源模式 1

当 TIMx\_SMCR 寄存器的 SMS = 111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 85. TI2 外部时钟连接例子



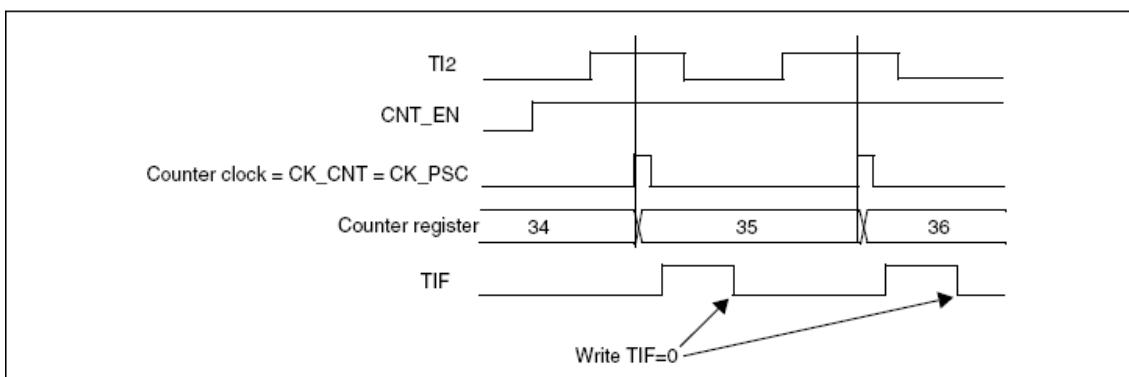
例如，要配置向上计数器在 TI2 输入端的上升沿计数，使用下列步骤：

1. 配置 TIMx\_CCMR1 寄存器 CC2S = 01，配置通道 2 检测 TI2 输入的上升沿
2. 配置 TIMx\_CCMR1 寄存器的 IC2F[3: 0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2F = 0000）
- 注：捕获预分频器不用作触发，所以不需要对它进行配置
3. 配置 TIMx\_CCER 寄存器的 CC2P = 0，选定上升沿极性
4. 配置 TIMx\_SMCR 寄存器的 SMS = 111，选择定时器外部时钟模式 1
5. 配置 TIMx\_SMCR 寄存器中的 TS = 110，选定 TI2 作为触发输入源
6. 设置 TIMx\_CR1 寄存器的 CEN = 1，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时取决于在 TI2 输入端的重新同步电路。

图 86. 外部时钟模式 1 下的控制电路

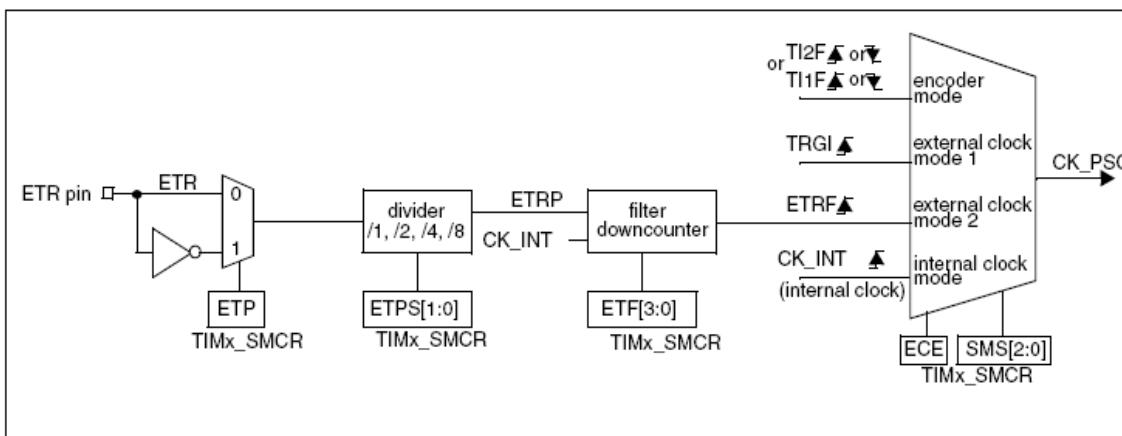


## 外部时钟源模式 2

选定此模式的方法为：令 **TIMx\_SMCR** 寄存器中的 **ECE = 1** 计数器能够在外部触发 **ETR** 的每一个上升沿或下降沿计数。

下图是外部触发输入的总体框图：

图 87. 外部触发输入框图



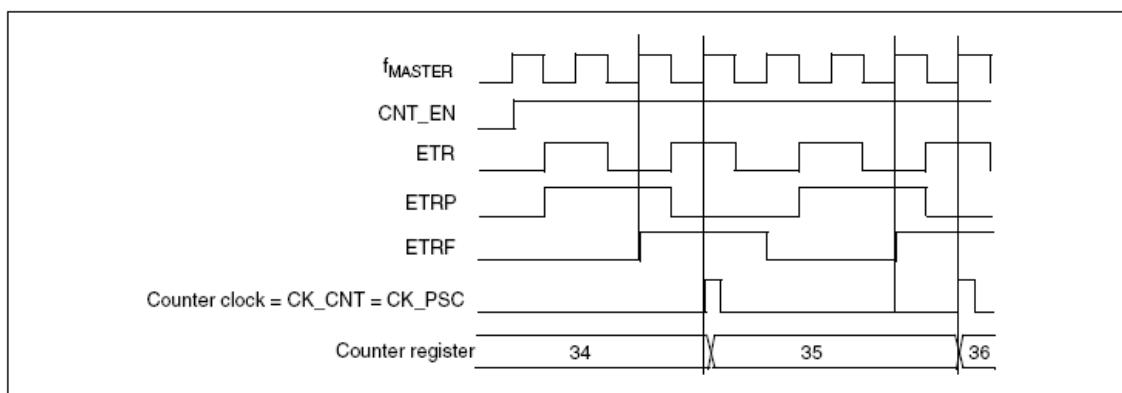
例如，要配置在 **ETR** 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 **TIMx\_SMCR** 寄存器中的 **ETF[3: 0] = 0000**
2. 设置预分频器，置 **TIMx\_SMCR** 寄存器中的 **ETPS[1: 0] = 01**
3. 设置在 **ETR** 的上升沿检测，置 **TIMx\_SMCR** 寄存器中的 **ETP = 0**
4. 开启外部时钟模式 2，置 **TIMx\_SMCR** 寄存器中的 **ECE = 1**
5. 启动计数器，置 **TIMx\_CR1** 寄存器中的 **CEN = 1**

计数器在每 2 个 **ETR** 上升沿计数一次。

在 **ETR** 的上升沿和计数器实际时钟之间的延时取决于在 **ETRP** 信号端的重新同步电路。

图 88. 外部时钟模式 2 下的控制电路

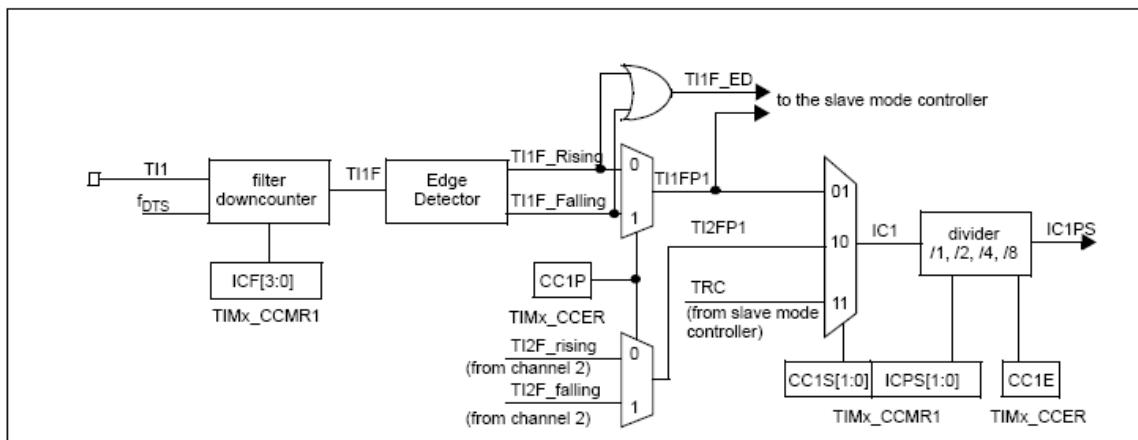


### 12.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

下面几张图是一个捕获/比较通道概览。输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

图 89. 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 90. 捕获/比较通道 1 的主电路

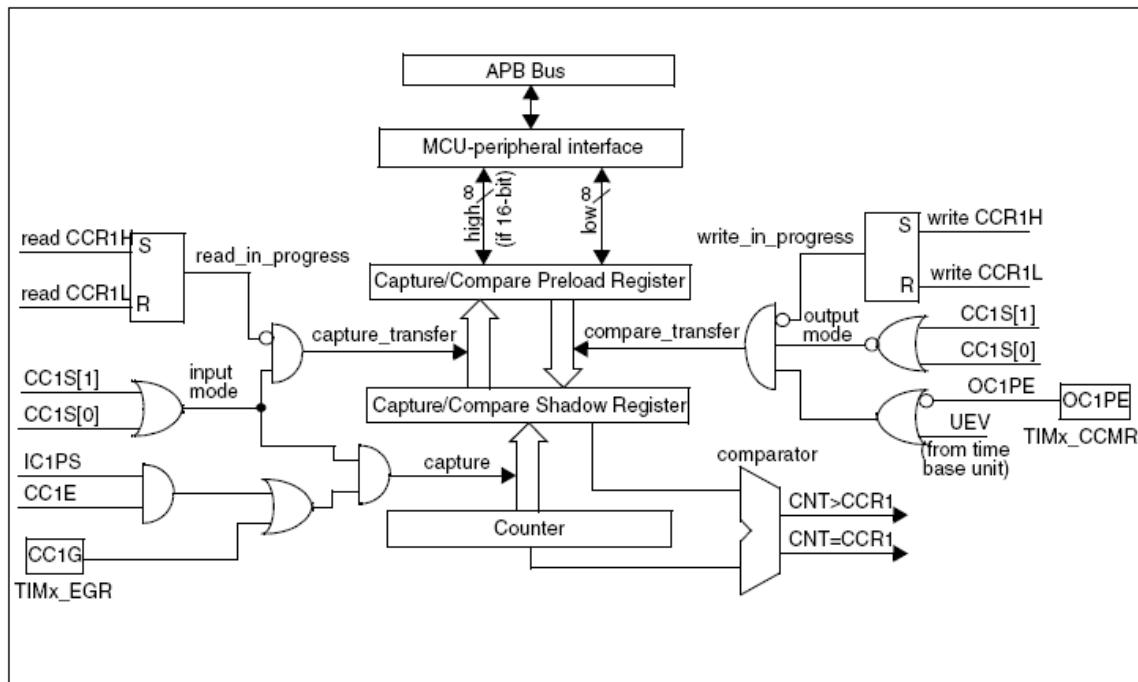
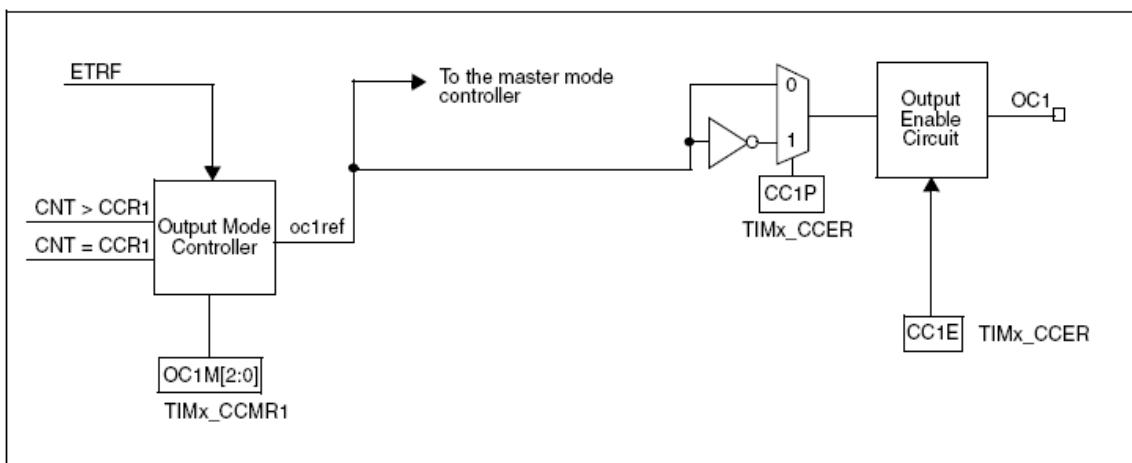


图 91. 捕获/比较通道的输出部分（通道 1）



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。

### 12.3.5 输入捕捉模式

在输入捕获模式下，当检测到  $ICx$  信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器 ( $TIMx\_CCRx$ ) 中。当捕获事件发生时，相应的  $CCxIF$  标志 ( $TIMx\_SR$  寄存器) 被置 1，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时  $CCxIF$  标志已经为高，那么重复捕获标志  $CCxOF$  ( $TIMx\_SR$  寄存器) 被置 1。写  $CCxIF = 0$  可清除  $CCxIF$ ，或读取存储在  $TIMx\_CCRx$  寄存器中的捕获数据也可清除  $CCxIF$ 。写  $CCxOF = 0$  可清除  $CCxOF$ 。

以下例子说明如何在  $TI1$  输入的上升沿时捕获计数器的值到  $TIMx\_CCR1$  寄存器中，步骤如下：

- 选择有效输入端： $TIMx\_CCR1$  必须连接到  $TI1$  输入，所以写入  $TIMx\_CCR1$  寄存器中的  $CC1S = 01$ ，一旦  $CC1S$  不为 00 时，通道被配置为输入，并且  $TIMx\_CCR1$  寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为  $TIx$  时，输入滤波器控制位是  $TIMx\_CCMRx$  寄存器中的  $ICxF$  位）。假设输入信号在最多 5 个时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此我们可以（以  $fDTS$  频率）连续采样 8 次，以确认在  $TI1$  上一次真实的边沿变换，即在  $TIMx\_CCMR1$  寄存器中写入  $IC1F = 0011$ 。
- 选择  $TI1$  通道的有效转换边沿，在  $TIMx\_CCER$  寄存器中写入  $CC1P = 0$ （上升沿）。
- 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写  $TIMx\_CCMR1$  寄存器的  $IC1PS = 00$ ）。
- 设置  $TIMx\_CCER$  寄存器的  $CC1E = 1$ ，允许捕获计数器的值到捕获寄存器中。
- 如果需要，通过设置  $TIMx\_DIER$  寄存器中的  $CC1IE$  位允许相关中断请求，通过设置  $TIMx\_DIER$  寄存器中的  $CC1DE$  位允许 DMA 请求。

发生当一个输入捕获时：

- 当产生有效的电平转换时，计数器的值被传送到  $TIMx\_CCR1$  寄存器。
- $CC1IF$  标志被设置（中断标志）。当发生至少 2 个连续的捕获时，而  $CC1IF$  未曾被清除。
- $CC1OF$  也被置 1。

- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

*注：设置 TIMx\_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。*

### 12.3.6 PWM 输入模式

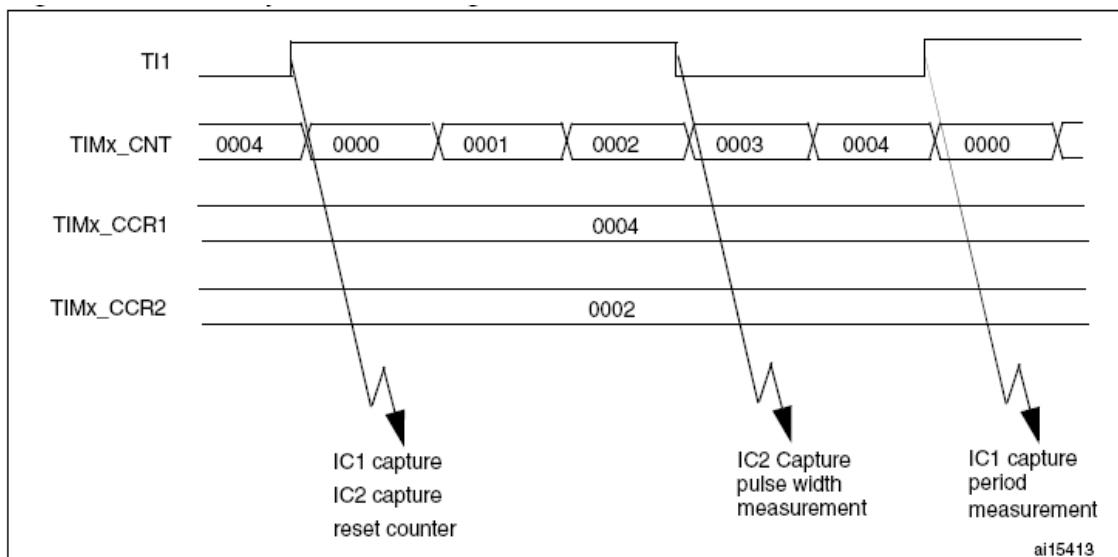
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 IC<sub>x</sub> 信号被映射同一个 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，你需要测量输入到 TI1 上的 PWM 信号的长度 (TIMx\_CCR1 寄存器) 和占空比 (TIMx\_CCR2 寄存器)，具体步骤如下（取决于 CK\_INT 的频率和预分频器的值）

- 选择 TIMx\_CCR1 的有效输入：置 TIMx\_CCMR1 寄存器的 CC1S = 01（选择 TI1）。
- 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx\_CCR1 中和清除计数器）：置 CC1P = 0（上升沿有效）。
- 选择 TIMx\_CCR2 的有效输入：置 TIMx\_CCMR1 寄存器的 CC2S = 10（选择 TI1）。
- 选择 TI1FP2 的有效极性（捕获数据到 TIMx\_CCR2）：置 CC2P = 1（下降沿有效）。
- 选择有效的触发输入信号：置 TIMx\_SMCR 寄存器中的 TS = 101（选择 TI1FP1）。
- 配置从模式控制器为复位模式：置 TIMx\_SMCR 中的 SMS = 100。
- 使能捕获：置 TIMx\_CCER 寄存器中 CC1E = 1 且 CC2E = 1。

图 92. PWM 输入模式时序



由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器。所以 PWM 输入模式只能使用 TIMx\_CH1 / TIMx\_CH2 信号。

### 12.3.7 强制输出模式

在输出模式 (TIMx\_CCMRx 寄存器中 CCxS = 00) 下, 输出比较信号 (OCxREF 和相应的 OCx) 能够直接由软件强置为有效或无效状态, 而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx\_CCMRx 寄存器中相应的 OCxM = 101, 即可强置输出比较信号 (OCxREF/OCx) 为有效状态。这样 OCxREF 被强置为高电平 (OCxREF 始终为高电平有效), 同时 OCx 得到 CCxP 极性位相反的值。

例如: CCxP = 0 (OCx 高电平有效), 则 OCx 被强置为高电平。

置 TIMx\_CCMRx 寄存器中的 OCxM = 100, 可强置 OCxREF 信号为低。

该模式下, 在 TIMx\_CCRx 影子寄存器和计数器之间的比较仍然在进行, 相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

### 12.3.8 输出比较模式

此项功能是用来控制一个输出波形或者指示何时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管脚上。在比较匹配时, 输出管脚可以保持它的电平 (OCxM = 000)、被设置成有效电平 (OCxM = 001)、被设置成无有效电平 (OCxM = 010) 或进行翻转 (OCxM = 011)。
- 设置中断状态寄存器中的标志位 (TIMx\_SR 寄存器中的 CCxIF 位)。
- 若设置了相应的中断屏蔽 (TIMx\_DIER 寄存器中的 CCxIE 位), 则产生一个中断。
- 若设置了相应的使能位 (TIMx\_DIER 寄存器中的 CCxDE 位, TIMx\_CR2 寄存器中的 CCDS 位选择 DMA 请求功能), 则产生一个 DMA 请求。

TIMx\_CCMRx 中的 OCxPE 位选择 TIMx\_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下, 更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

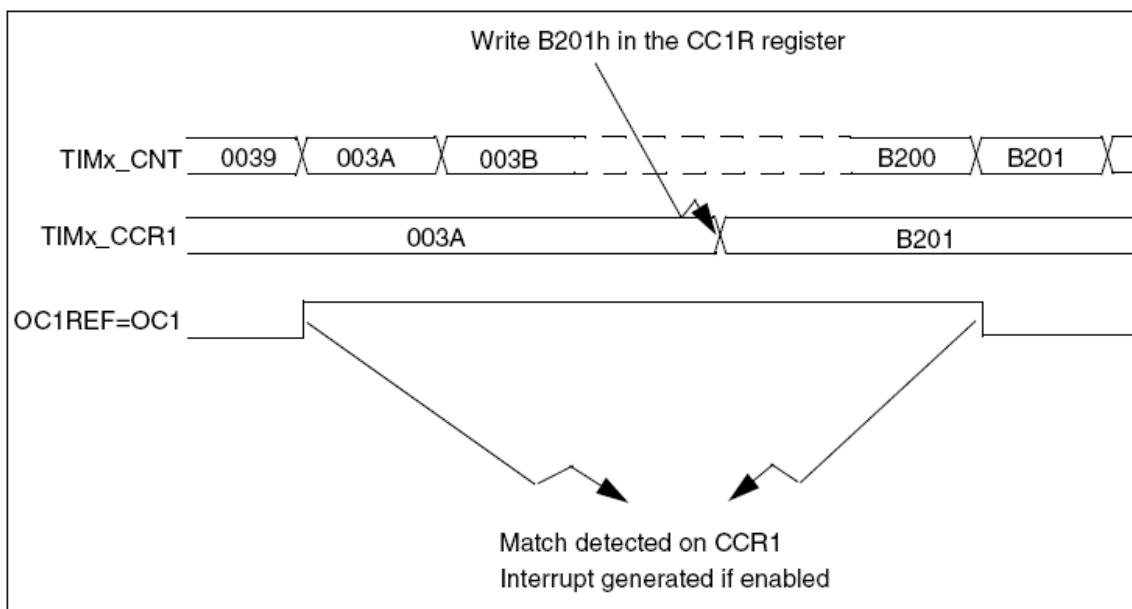
同步的精度可以达到计数器的一个计数周期。输出比较模式 (在单脉冲模式下) 也能用来输出一个单脉冲。

输出比较模式的配置步骤:

1. 选择计数器时钟 (内部, 外部, 预分频器)
2. 将相应的数据写入 TIMx\_ARR 和 TIMx\_CCRx 寄存器中
3. 如果要产生一个中断请求和/或一个 DMA 请求, 设置 CCxIE 位和/或 CCxDE 位。
4. 选择输出模式, 例如: 必须设置 OCxM = '011'、OCxPE = '0'、CCxP = '0' 和 CCxE = '1', 当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出管脚, CCRx 预装载未用, 开启 OCx 输出且高电平有效。
5. 设置 TIMx\_CR1 寄存器的 CEN 位启动计数器

TIMx\_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形, 条件是未使用预装载寄存器 (OCxPE = '0', 否则 TIMx\_CCRx 影子寄存器只能在发生下一次更新事件时被更新)。下图给出了一个例子。

图 93. 输出比较模式，翻转 OC1



### 12.3.9 PWM 模式

脉冲宽度调制模式可以产生一个由 **TIMx\_ARR** 寄存器确定频率、由 **TIMx\_CCRx** 寄存器确定占空比的信号。

在 **TIMx\_CCMRx** 寄存器中的 **OCxM** 位写入‘110’（PWM 模式 1）或‘111’（PWM 模式 2），能够独立地设置每个 **OCx** 输出通道产生一路 PWM。必须设置 **TIMx\_CCMRx** 寄存器 **OCxPE** 位以使能相应的预装载寄存器，最后还要设置 **TIMx\_CR1** 寄存器的 **ARPE** 位使能自动重装载的预装载寄存器（在向上计数或中心对称模式中）。

因为仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 **TIMx\_EGR** 寄存器中的 **UG** 位来初始化所有的寄存器。

**OCx** 的极性可以通过软件在 **TIMx\_CCER** 寄存器中的 **CCxP** 位设置，它可以设置为高电平有效或低电平有效。**TIMx\_CCER** 寄存器中的 **CCxE** 位控制 **OCx** 输出使能。详见 **TIMx\_CCERx** 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，**TIMx\_CNT** 和 **TIM1\_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 **TIM1\_CCRx ≤ TIM1\_CNT** 或者 **TIM1\_CNT ≤ TIM1\_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。

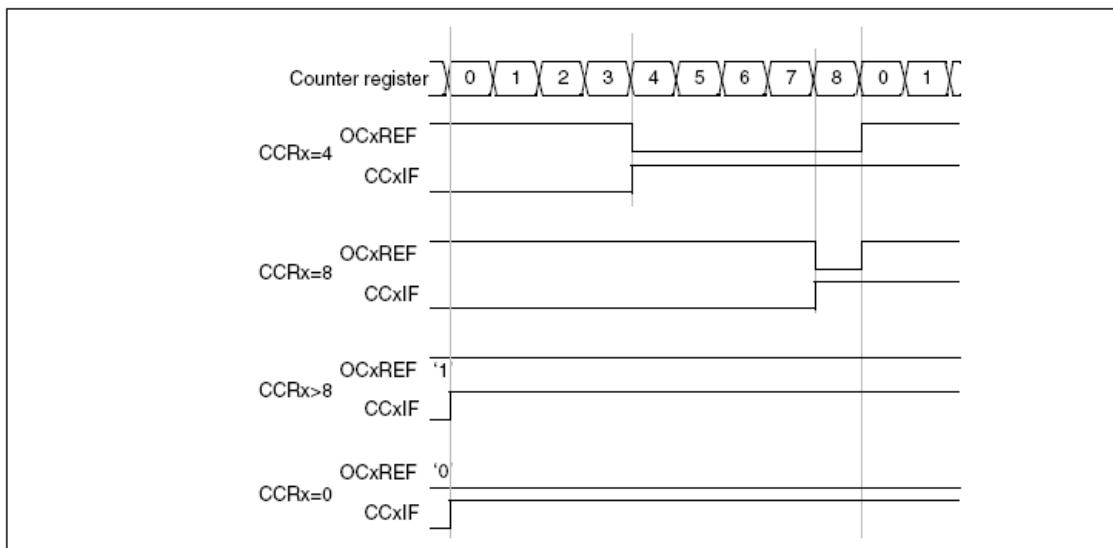
#### PWM 边沿对齐模式

##### 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  $\text{TIMx_CNT} < \text{TIMx_CCRx}$  时 PWM 信号参考  $\text{OCxREF}$  为高，否则为低。如果  $\text{TIMx_CCRx}$  中的比较值大于自动重装载值 ( $\text{TIMx_ARR}$ )，则  $\text{OCxREF}$  保持为‘1’。如果比较值为 0，则  $\text{OCxREF}$  保持为‘0’。下图为  $\text{TIMx_ARR} = 8$  时边沿对齐的 PWM 波形实例。

图 94. 边沿对齐的 PWM 波形 ( $\text{ARR} = 8$ )



### 向下计数的配置

当  $\text{TIMx_CR1}$  寄存器的 DIR 位为高时执行向下计数。

在 PWM 模式 1，当  $\text{TIMx_CNT} > \text{TIMx_CCRx}$  时参考信号  $\text{OCxREF}$  为低，否则为高。如果  $\text{TIMx_CCRx}$  中的比较值大于  $\text{TIMx_ARR}$  中的自动重装载值，则  $\text{OCxREF}$  保持为‘1’。该模式下不能产生 0% 的 PWM 波形。

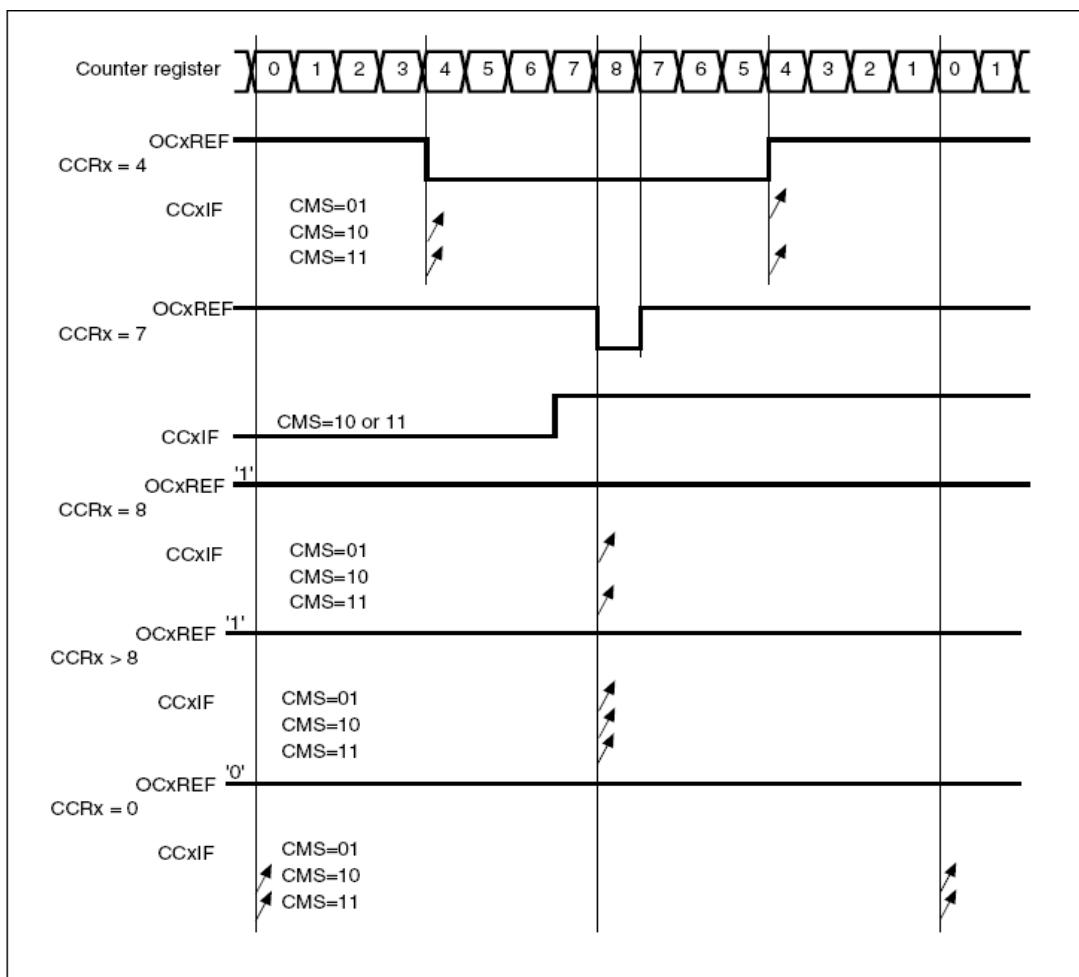
### PWM 中央对齐模式

当  $\text{TIMx_CR1}$  寄存器中的 CMS 位不为‘00’时为中央对齐模式（所有其他的配置对  $\text{OCxREF}/\text{OCx}$  信号都有相同的作用）。根据不同的 CMS 位的设置，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。 $\text{TIMx_CR1}$  寄存器中的计数方向位 (DIR) 由硬件更新，不要用软件修改它。参看中央对齐模式章节。

下图给出了一些中央对齐的 PWM 波形的例子

- $\text{TIMx_ARR} = 8$
- PWM 模式 1
- $\text{TIMx_CR1}$  寄存器中的 CMS = 01，在中央对齐模式 1 时，当计数器向下计数时设置比较标志。

图 95. 中央对齐的 PWM 波形 (APR = 8)



#### 使用中央对齐模式的提示:

- 进入中央对齐模式时，使用当前的上/下计数配置；这就意味着计数器向上还是向下计数取决于 **TIMx\_CR1** 寄存器中 **DIR** 位的当前值。此外，软件不能同时修改 **DIR** 和 **CMS** 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值 (**TIMx\_CNT > TIMx\_ARR**)，则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 **TIMx\_ARR** 的值写入计数器，方向被更新，但不产生更新事件 **UEV**。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置 **TIMx\_EGR** 位中的 **UG** 位），不要在计数进行过程中修改计数器的值。

#### 12.3.10 单脉冲模式

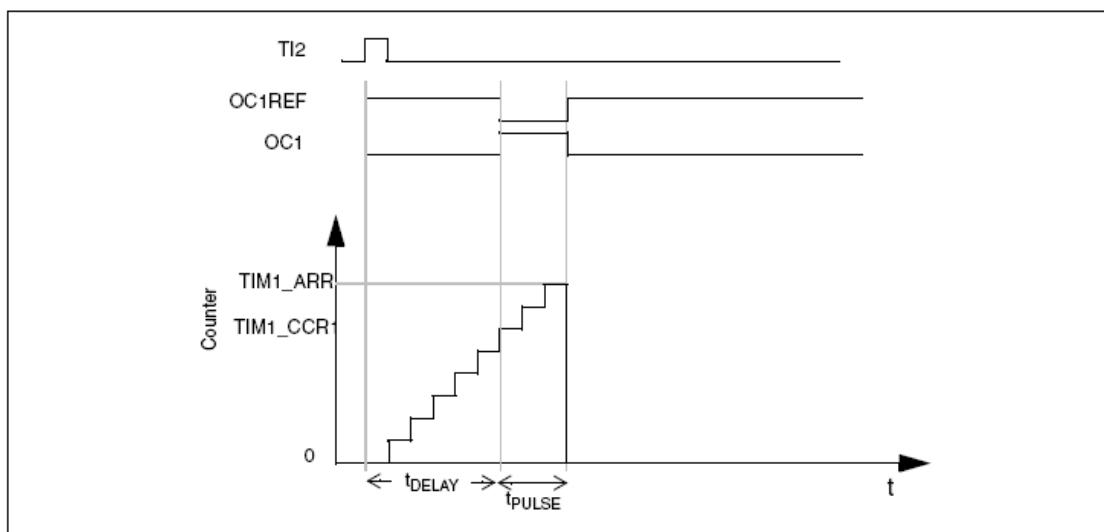
单脉冲模式 (OPM) 是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 **TIMx\_CR1** 寄存器中的 **OPM** 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 **UEV** 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式:  $CNT < CCRx \leq ARR$  (特别地,  $0 < CCRx$ )
- 向下计数方式:  $CNT > CCRx$

图 96. 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 1:

- 置  $TIMx\_CCMR1$  寄存器中的  $CC2S = 01$ , 把  $TI2FP2$  映像到  $TI2$ 。
- 置  $TIMx\_CCER$  寄存器中的  $CC2P = 0$ , 使  $TI2FP2$  能够检测上升沿。
- 置  $TIMx\_SMCR$  寄存器中的  $TS = 110$ ,  $TI2FP2$  作为从模式控制器的触发 (TRGI)。
- 置  $TIMx\_SMCR$  寄存器中的  $SMS = 110$  (触发模式),  $TI2FP2$  被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定 (要考虑时钟频率和计数器预分频器)。

- $t_{DELAY}$  由写入  $TIMx\_CCR1$  寄存器中的值定义。
- $t_{PULSE}$  由自动装载值和比较值之间的差值定义 ( $TIMx\_ARR - TIMx\_CCR1$ )。
- 假定当发生比较匹配时要产生从 0 到 1 的波形, 当计数器到达预装载值是要产生一个从 1 到 0 的波形; 首先要置  $TIMx\_CCMR1$  寄存器的  $OC1M = 111$ , 进入 PWM 模式 2; 根据需要有选择地使能预装载寄存器: 置  $TIMx\_CCMR1$  中的  $OC1PE = 1$  和  $TIMx\_CR1$  寄存器中的  $ARPE$ ; 然后在  $TIMx\_CCR1$  寄存器中填写比较值, 在  $TIMx\_ARR$  寄存器中填写自动装载值, 修改  $UG$  位来产生一个更新事件, 然后等待在  $TI2$  上的一个外部触发事件。本例中,  $CC1P = 0$ 。

在这个例子中,  $TIMx\_CR1$  寄存器中的  $DIR$  和  $CMS$  位应该置低。

因为只需一个脉冲, 所以必须设置  $TIMx\_CR1$  寄存器中的  $OPM = 1$ , 在下一个更新事件 (当计数器从自动装载值翻转到 0) 时停止计数。

**特殊情况: OCx 快速使能:**

在单脉冲模式下, 在  $TIx$  输入脚的边沿检测逻辑设置  $CEN$  位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期, 因此它限制了可得到的最小延时  $t_{DELAY}$ 。

如果要以最小延时输出波形，可以设置 **TIMx\_CCMRx** 寄存器中的 **OCxFE** 位；此时强制 **OCxREF**（和 **OCx**）被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。**OCxFE** 只在通道配置为 **PWM1** 和 **PWM2** 模式时起作用。

### 12.3.11 在外部事件时清除 OCxREF 信号

对于一个给定的通道，在 **ETRF** 输入端设置 **TIMx\_CCMRx** 寄存器中对应的 **OCxCE** 位为‘1’的高电平能够把 **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**。

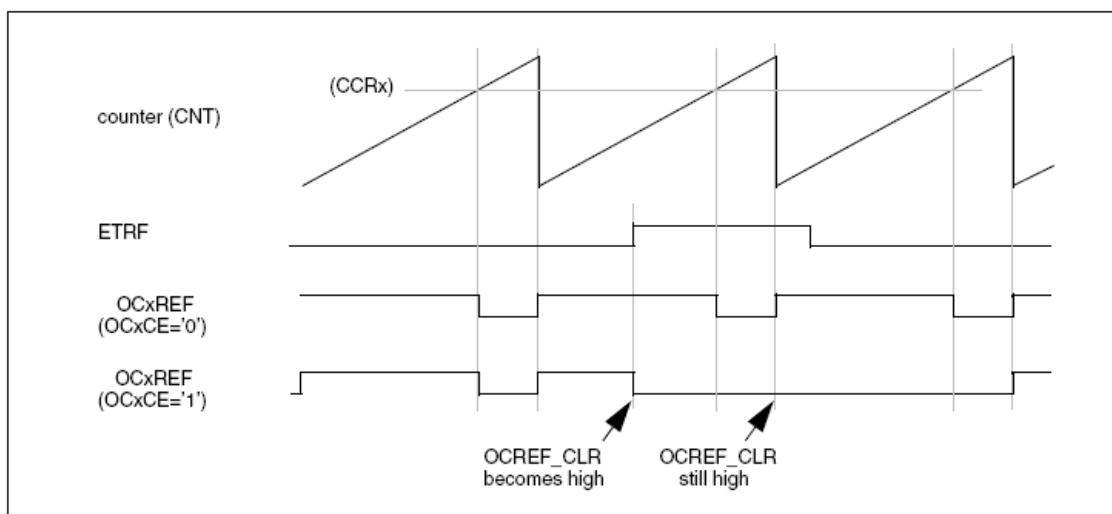
该功能只能用于输出比较和 **PWM** 模式，而不能用于强置模式。

例如，**OCxREF** 信号可以连到一个外部输入。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 97. 清除 TIMx 的 OCxREF



### 12.3.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 **TI2** 的边沿计数，则置 **TIMx\_SMCR** 寄存器中的 **SMS = 001**；如果只在 **TI1** 边沿计数，则置 **SMS = 010**；如果计数器同时在 **TI1** 和 **TI2** 边沿计数，则置 **SMS = 011**。

通过设置 **TIMx\_CCER** 寄存器中的 **CC1P** 和 **CC2P** 位，可以选择 **TI1** 和 **TI2** 极性；如果需要，还可以对输入滤波器编程。

两个输入 **TI1** 和 **TI2** 被用来作为增量编码器的接口。下表，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**；如果没有滤波和变相，则 **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正常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 **TI1** 和 **TI2** 不同时变换。

表 27. 计数方向与编码器信号的关系

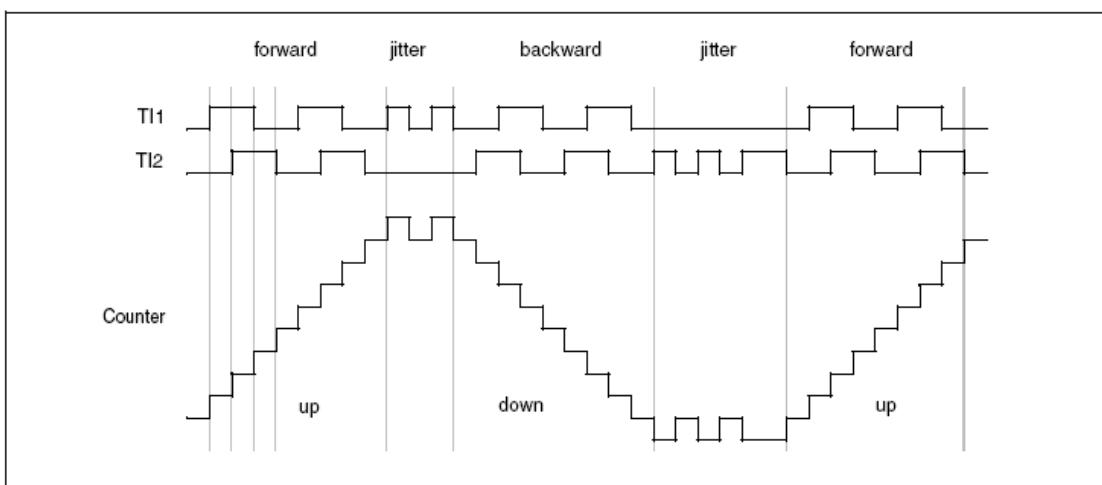
有效边沿	相对信号的电平 ( <b>TI1FP1</b> 对应 <b>TI2</b> , <b>TI2FP2</b> 对应 <b>TI1</b> )	<b>TI1FP1</b> 信号		<b>TI2FP2</b> 信号	
		上升	下降	上升	下降
仅在 <b>TI1</b> 计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在 <b>TI2</b> 计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在 <b>TI1</b> 和 <b>TI2</b> 上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	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，一般使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

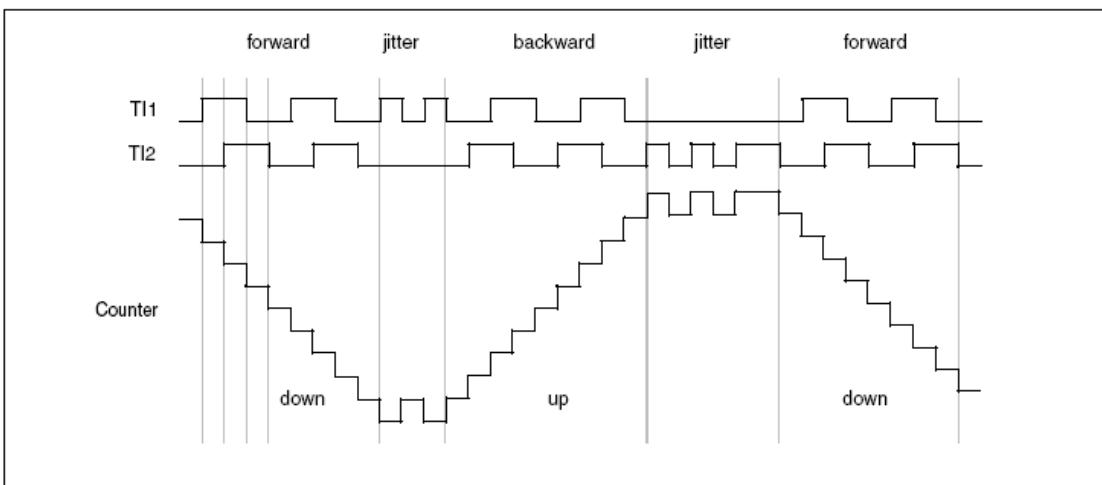
- **CC1S = '01'** (**TIMx\_CCMR1** 寄存器, **IC1FP1** 映射到 **TI1**)
- **CC2S = '01'** (**TIMx\_CCMR2** 寄存器, **IC2FP2** 映射到 **TI2**)
- **CC1P = '0'** (**TIMx\_CCER** 寄存器, **IC1FP1** 不反相, **IC1FP1 = TI1**)
- **CC2P = '0'** (**TIMx\_CCER** 寄存器, **IC2FP2** 不反相, **IC2FP2 = TI2**)
- **SMS = '011'** (**TIMx\_SMCR** 寄存器, 所有的输入均在上升沿和下降沿有效) .
- **CEN = '1'** (**TIMx\_CR1** 寄存器, 计数器使能)

图 98. 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 99. IC1FP1 反相的编码器接口模式实例



当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式定时器测量两个编码器事件的间隔，可以获得动态的信息（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）。它也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

### 12.3.13 定时器输入异或功能

TIMx\_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx\_CH1、TIMx\_CH2 和 TIMx\_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。上一章 15.3.18 节给出了此特性用于连接霍尔传感器的例子。

### 12.3.14 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

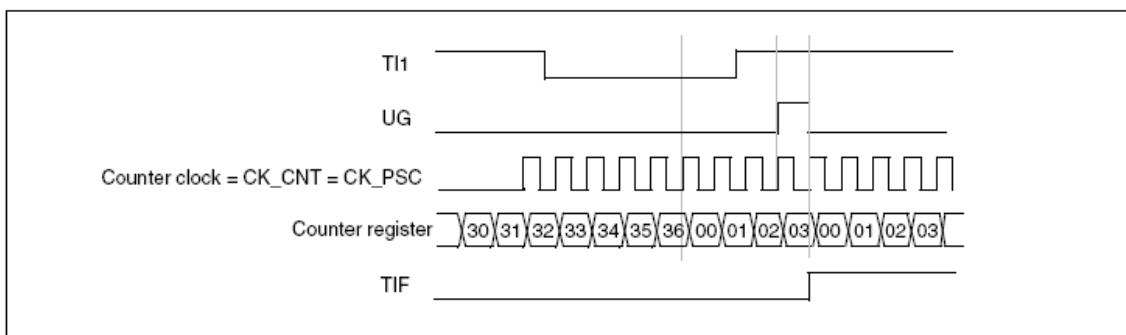
在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

- 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 100. 复位模式下的控制电路



#### 从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

在如下的例子中，计数器只在 TI1 为低时向上计数：

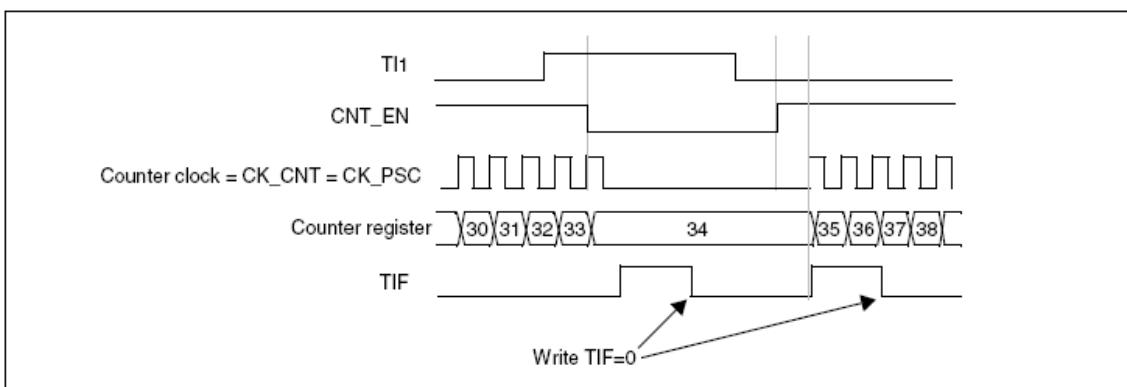
- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F = 0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S = 01。置 TIMx\_CCER 寄存器中 CC1P = 1 以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS = 101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。

- 置  $\text{TIMx\_CR1}$  寄存器中  $\text{CEN} = 1$ , 启动计数器。在门控模式下, 如果  $\text{CEN} = 0$ , 则计数器不能启动, 不论触发输入电平如何。

只要  $\text{TI1}$  为低, 计数器开始依据内部时钟计数, 在  $\text{TI1}$  变高时停止计数。当计数器开始或停止时都设置  $\text{TIMx\_SR}$  中的  $\text{TIF}$  标置。

$\text{TI1}$  上升沿和计数器实际停止之间的延时取决于  $\text{TI1}$  输入端的重同步电路。

图 101. 门控模式下的控制电路



### 从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

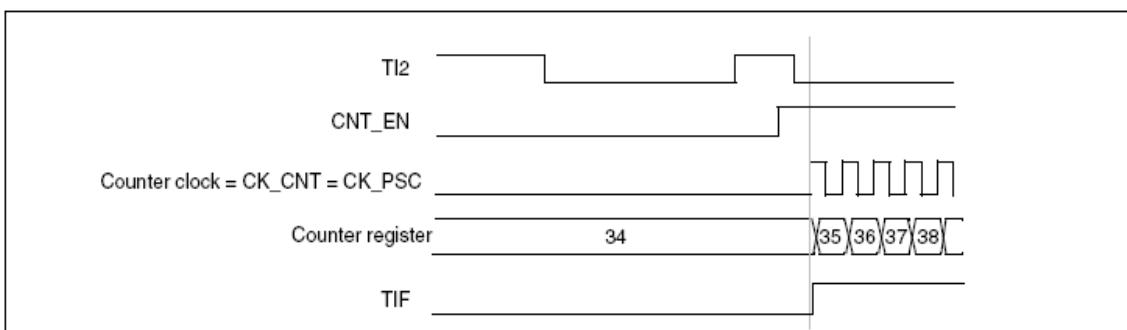
在下面的例子中, 计数器在  $\text{TI2}$  输入的上升沿开始向上计数:

- 配置通道 2 检测  $\text{TI2}$  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  $\text{IC2F} = 0000$ )。触发操作中不使用捕获预分频器, 不需要配置。 $\text{CC2S}$  位只用于选择输入捕获源, 置  $\text{TIMx_CCMR1}$  寄存器中  $\text{CC2S} = 01$ 。置  $\text{TIMx_CCER}$  寄存器中  $\text{CC1P} = 1$  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 102. 触发器模式下的控制电路



### 从模式：外部时钟模式 2 + 触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用 TIMx\_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

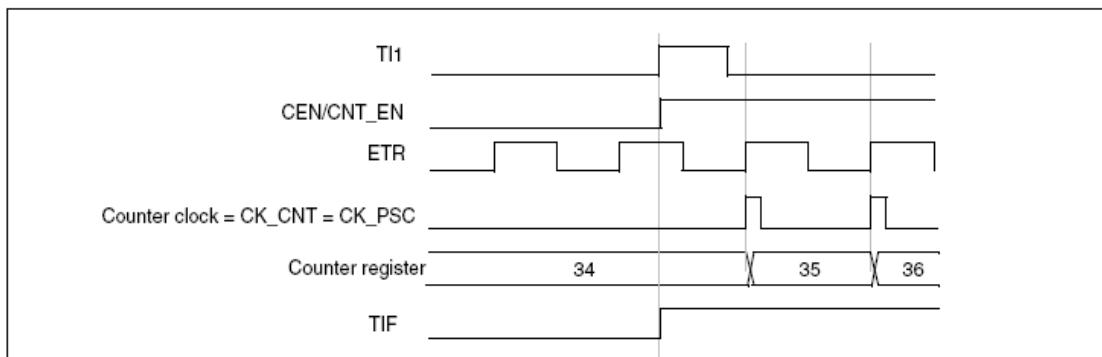
在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

- 通过 TIMx\_SMCR 寄存器配置外部触发输入电路：
  - ETF = 0000: 没有滤波
  - ETPS = 00: 不用预分频器
  - ETP = 0: 检测 ETR 的上升沿，置 ECE = 1 使能外部时钟模式 2
- 按如下配置通道 1，检测 TI 的上升沿：
  - IC1F=0000: 没有滤波
  - 触发操作中不使用捕获预分频器，不需要配置
  - 置 TIMx\_CCMR1 寄存器中 CC1S = 01，选择输入捕获源
  - 置 TIMx\_CCER 寄存器中 CC1P = 0 以确定极性（只检测上升沿）
- 置 TIMx\_SMCR 寄存器中 SMS = 110，配置定时器为触发模式。置 TIMx\_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位间的延时取决于 ETRP 输入端的重同步电路。

图 103. 外部时钟模式 2 + 触发模式下的控制电路



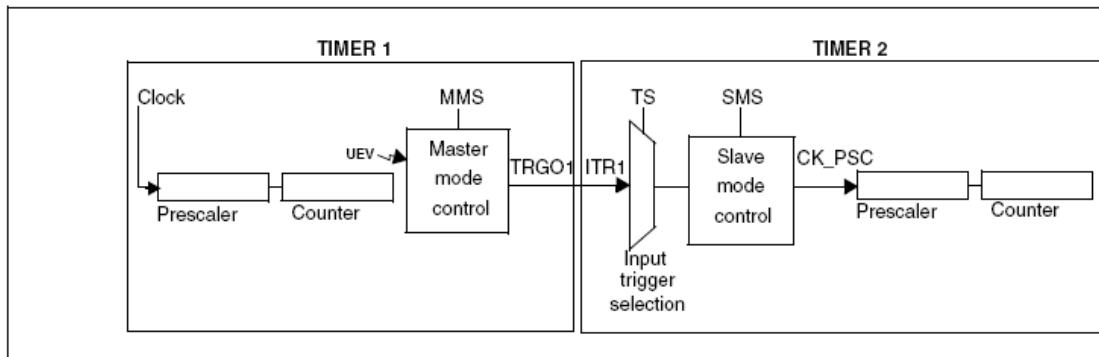
### 12.3.15 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

使用一个定时器作为另一个定时器的预分频器

图 104. 主/从定时器的例子



如：可以配置定时器 1 作为定时器 3 的预分频器。参考上图，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 3，设置 TIM3\_SMCR 寄存器的 TS = '000'，配置定时器 3 为使用 ITR1 作为内部触发的从模式。
- 然后把从模式控制器置于外部时钟模式 1 (TIM3\_SMCR 寄存器的 SMS = 111)；这样定时器 3 即可由定时器 1 周期性的上升沿（即定时器 1 的计数器溢出）信号驱动。
- 最后，必须设置相应 (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 3 的计数器。

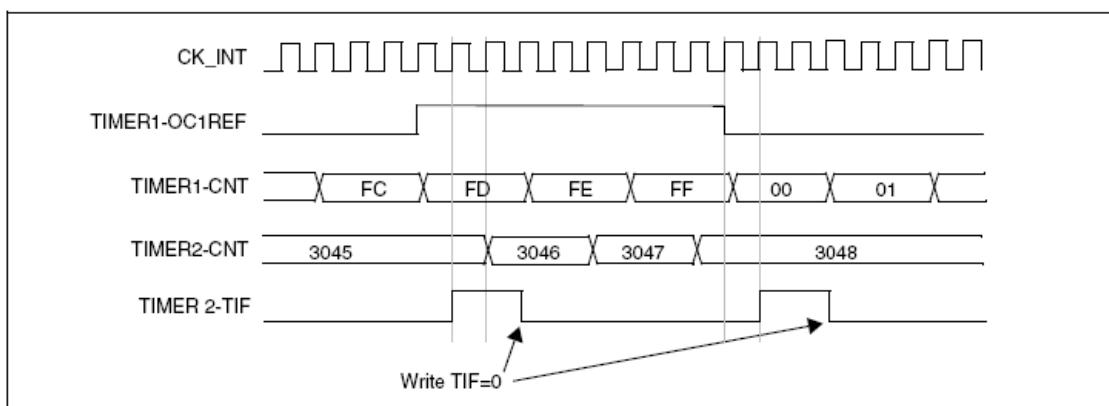
### 使用一个定时器使能另一个定时器

在这个例子中，定时器 3 的运行受由定时器 1 的输出比较控制。参考图 42 的连接。只当定时器 1 的 OC1REF 为高时定时器 3 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器 对 CK\_INT 除以 3 ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ) 得到。

- 配置定时器 1 为主模式，送出它的输出比较参考信号 (OC1REF) 为触发输出 (TIM1\_CR2 寄存器的 MMS = 100)
- 配置定时器 1 的 OC1REF 波形 (TIM1\_CCMR1 寄存器)
- 配置定时器 3 从定时器 1 获得输入触发 (TIM3\_SMCR 寄存器的 TS = 001)
- 配置定时器 3 为门控模式 (TIM3\_SMCR 寄存器的 SMS = 101)
- 置 TIM3\_CR1 寄存器的 CEN = 1 以使能定时器 3
- 置 TIM1\_CR1 寄存器的 CEN = 1 以启动定时器 1

**注：**定时器 3 的时钟不与定时器 1 的时钟同步，这个模式只影响定时器 3 计数器的使能信号。

图 105. 定时器 1 的 OC1REF 控制定时器 3

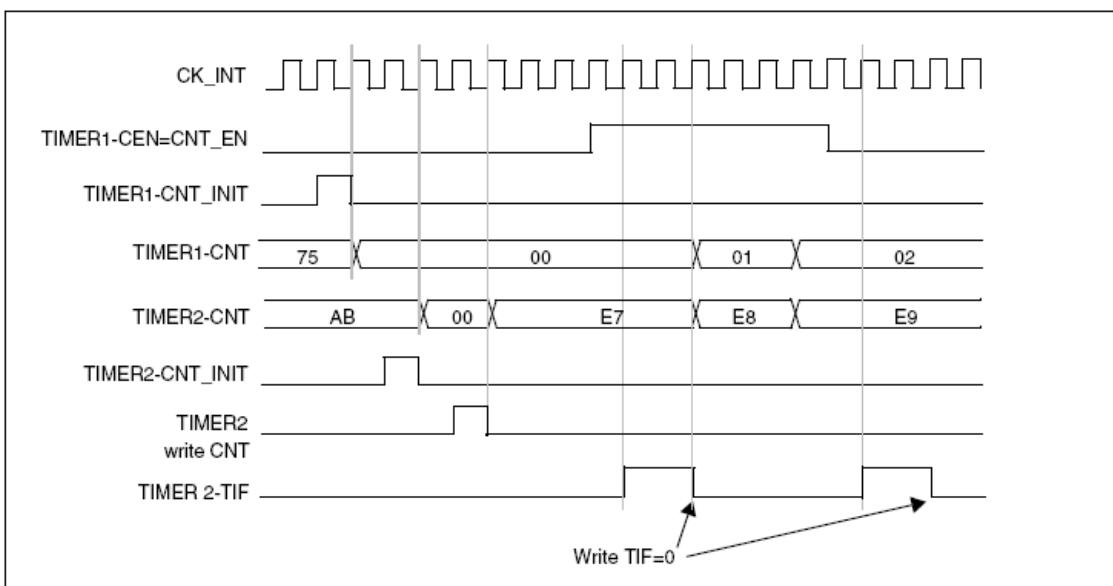


在上图的例子中，在定时器 3 启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1 之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMx\_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1 和定时器 3。定时器 1 是主模式并从 0 开始，定时器 3 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 3 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 3 从定时器 1 获得输入触发（TIM3\_SMCR 寄存器的 TS = 000）
- 配置定时器 3 为门控模式（TIM3\_SMCR 寄存器的 SMS = 101）
- 置 TIM1\_EGR 寄存器的 UG = 1，复位定时器 1。
- 置 TIM3\_EGR 寄存器的 UG = 1，复位定时器 3。
- 写 0xE7 至定时器 3 的计数器（TIM3\_CNTL），初始化它为 0xE7。
- 置 TIM3\_CR1 寄存器的 CEN = 1 以使能定时器 3。
- 置 TIM1\_CR1 寄存器的 CEN = 1 以启动定时器 1。
- 置 TIM1\_CR1 寄存器的 CEN = 0 以停止定时器 1。

图 106. 通过使能定时器 1 可以控制定时器 3

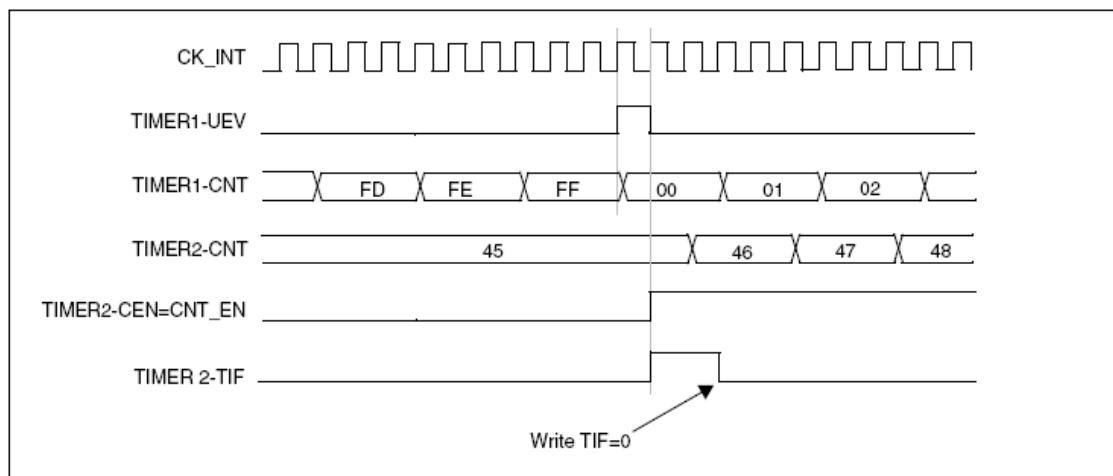


### 使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1 的更新事件使能定时器 3。参考图 43 的连接。一旦定时器 1 产生更新事件，**定时器 3** 即从它当前的数值（可以是非 0）按照分频的内部时钟开始计数。在收到触发信号时，**定时器 3** 的 CEN 位被自动地置 1，同时计数器开始计数直到写 0 到 TIM3\_CR1 寄存器的 CEN 位。两个定时器的时钟频率都是由预分频器对 CK\_INT 除以 3 ( $f_{CK\_CNT}=f_{CK\_INT}/3$ )。

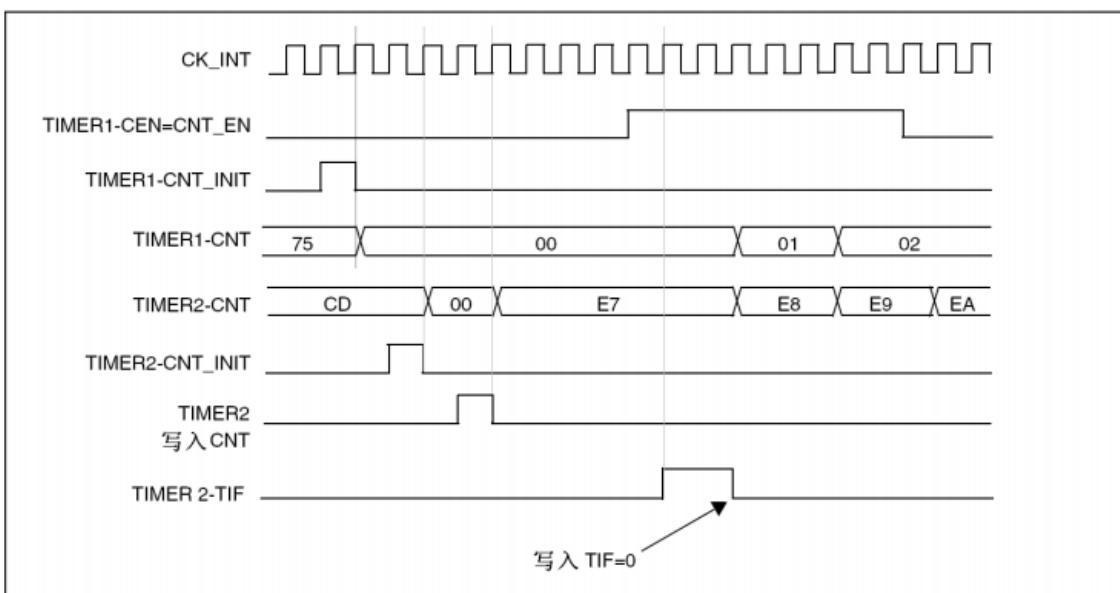
- 配置定时器 1 为主模式，送出它的更新事件（UEV）做为触发输出（TIM1\_CR2 寄存器的 MMS = 010）。
- 配置定时器 1 的周期（TIM1\_ARR 寄存器）。
- 配置**定时器 3** 从定时器 1 获得输入触发（TIM3\_SMCR 寄存器的 TS = 000）
- 配置**定时器 3** 为触发模式（TIM3\_SMCR 寄存器的 SMS = 110）
- 置 TIM1\_CR1 寄存器的 CEN = 1 以启动定时器 1。

图 107. 使用定时器 1 的更新触发定时器 3



在上一个例子中，可以在启动计数之前初始化两个计数器。下图显示在与 0 相同配置情况下，使用触发模式而不是门控模式（TIM3\_SMCR 寄存器的 SMS = 110）的动作。

图 108. 利用定时器 1 的使能触发定时器 3



### 使用一个额定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 3 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 作为触发输出 (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 3 从定时器 1 获得输入触发 (TIM3\_SMCR 寄存器的 TS = 000)
- 配置定时器 3 使用外部时钟模式 (TIM3\_SMCR 寄存器的 SMS = 111)
- 置 TIM1\_CR2 寄存器的 CEN = 1 以启动定时器 3
- 置 TIM1\_CR1 寄存器的 CEN = 1 以启动定时器 1

### 使用一个外部触发同步地启动 2 个定时器

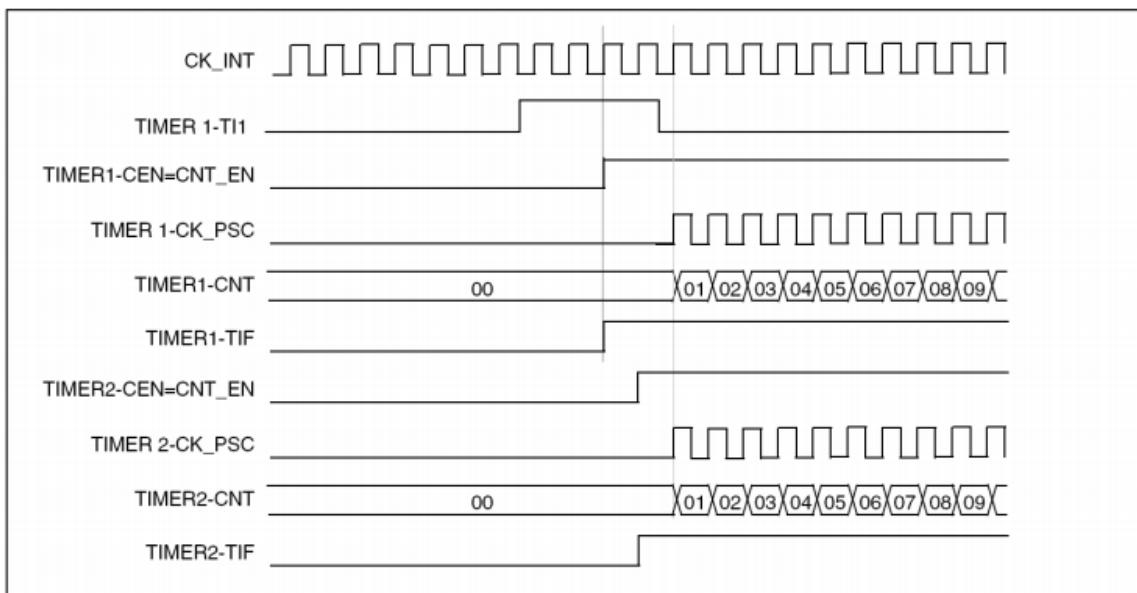
这个例子中当定时器 1 的 TI1 输入上升时使能定时器 1，使能定时器 1 的使能定时器 3。为保证计数器的对齐，定时器 1 必须配置为主/从模式（对应 TI1 为从，对应定时器 3 为主）：

- 配置定时器 1 为主模式，送出它的使能作为触发输出 (TIM1\_CR2 寄存器的 MMS='001')
- 配置定时器 1 为从模式，从 TI1 获得输入触发 (TIM1\_SMCR 寄存器的 TS = '100')
- 配置定时器 1 为触发模式 (TIM1\_SMCR 寄存器的 SMS='110')
- 配置定时器 3 从定时器 1 获得输入触发 (TIM3\_SMCR 寄存器的 TS = 000)
- 配置定时器 3 为触发模式 (TIM3\_SMCR 寄存器的 SMS = 110)

当定时器 1 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

**注：**在这个例子中，在启动之前两个定时器都被初始化（设置相应的 UG 位），两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器 (TIMx\_CNT) 在定时器间插入一个偏移。下图中能看到主/从模式下在定时器 1 的 CNT\_EN 和 CK\_PSC 之间有个延迟。

图 109. 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 3



### 12.3.16 调试模式

当微控制器进入调试模式（CPU 核心停止），根据 DBG 模块中 `DBG_TIMx_STOP` 的设置，`TIMx` 计数器或者继续正常操作，或者停止。详见调试模块章节。

## 12.4 TIMx 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。

### 12.4.1 控制寄存器 1 (`TIMx_CR1`)

偏移地址: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CKD	ARPE	CMS	DIR	OPM	URS	UDIS				CEN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 10	保留
位9: 8	<b>CKD[1: 0]:</b> 时钟分频因子 (Clock division) 这2位定义在定时器时钟 (CK_INT) 频率、死区时间和由死区发生器与数字滤波器 (ETR, TIx) 所用的采样时钟之间的分频比例。 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: 保留，不要使用这个配置
位7	<b>ARPE:</b> 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

位6: 5	<p><b>CMS[1: 0]:</b> 选择中央对齐模式 (Center-aligned mode selection)</p> <p>00: 边沿对齐模式。计数器依据方向位 (<b>DIR</b>) 向上或向下计数。</p> <p>01: 中央对齐模式1。计数器交替地向上和向下计数。配置为输出的通道 (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。</p> <p>10: 中央对齐模式2。计数器交替地向上和向下计数。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。</p> <p>11: 中央对齐模式3。计数器交替地向上和向下计数。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。</p> <p>注: 在计数器开启时 (<b>CEN</b> = 1), 不允许从边沿对齐模式转换到中央对齐模式。</p>
位4	<p><b>DIR:</b> 方向 (Direction)</p> <p>0: 计数器向上计数</p> <p>1: 计数器向下计数</p> <p>注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读。</p>
位3	<p><b>OPM:</b> 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</p> <p>1: 在发生下一次更新事件 (清除CEN位) 时, 计数器停止</p>
位2	<p><b>URS:</b> 更新请求源 (Update request source)</p> <p>软件通过该位选择UEV事件的源</p> <p>0: 如果允许产生更新中断或DMA请求, 则下述任一事件产生一个更新中断或DMA请求:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置UG位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果允许产生更新中断或DMA请求, 则只有计数器溢出/下溢才产生一个更新中断或DMA请求</p>
位1	<p><b>UDIS:</b> 禁止更新 (Update disable)</p> <p>软件通过该位允许/禁止UEV事件的产生</p> <p>0: 允许UEV。更新 (UEV) 事件由下述任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器溢出/下溢</li> <li>- 设置UG位</li> <li>- 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。</li> </ul> <p>1: 禁止UEV。不产生更新事件, 影子寄存器 (ARR、PSC、CCRx) 保持它们的值。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>
位0	<p><b>CEN:</b> 允许计数器 (Counter enable)</p> <p>0: 禁止计数器</p> <p>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> <p>在单脉冲模式下, 当发生更新事件时, CEN被自动清除。</p>

### 12.4.2 控制寄存器 2 (TIMx\_CR2)

偏移地址: 0x04

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留				TI1S	MMS	CCDS	保留								
							rw								

位31: 8	保留
位7	<b>TI1S:</b> TI1选择 (TI1 selection) 0: TIMx_CH1管脚连到TI1输入; 1: TIMx_CH1、TIMx_CH2和TIMx_CH3管脚经异或后连到TI1输入。
位6: 4	<b>MMS[1: 0]:</b> 主模式选择 (Master mode selection) 这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下: 000: 复位 – TIMx_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位，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 – 一旦发生一次捕获或一次比较成功时，当要设置CC1IF标志时 (即使它已经为高)，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)。
位3	<b>CCDS:</b> 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
位2: 0	保留，始终读为0。

### 12.4.3 从模式控制寄存器 (TIMx\_SMCR)

偏移地址: 0x08

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS		ETF		MSM	TS		保留	SMS					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

位31: 16	保留
位15	<p><b>ETP:</b> 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</p>
位14	<p><b>ECE:</b> 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信号上的任意有效上升沿驱动。      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。</p>
位13: 12	<p><b>ETPS[1: 0]:</b> 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</p>
位11: 8	<p><b>ETF[3: 0]:</b> 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f<sub>DTS</sub>采样      0001: 采样频率f<sub>SAMPLING</sub> = f<sub>CK_INT</sub>, N = 2      0010: 采样频率f<sub>SAMPLING</sub> = f<sub>CK_INT</sub>, N = 4      0011: 采样频率f<sub>SAMPLING</sub> = f<sub>CK_INT</sub>, N = 8      0100: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 6      0101: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 8      0110: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 6      0111: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 8      1000: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 6      1001: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 8      1010: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 5      1011: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 6      1100: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 8      1101: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 5      1110: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 6      1111: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 8</p>
位7	<p><b>MSM:</b> 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</p>

位6: 4	<p><b>TS[2: 0]: 触发选择 (Trigger selection)</b> 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) 001: 内部触发1 (ITR1) 010: 内部触发2 (ITR2) 011: 内部触发3 (ITR3) 100: TI1的边沿检测器 (TI1F_ED) 101: 滤波后的定时器输入1 (TI1FP1) 110: 滤波后的定时器输入2 (TI2FP2) 111: 外部触发输入 (ETRF)</p> <p>更多有关ITRx的细节, 参见下表。</p> <p>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>
位3	保留, 始终读为0。
位2: 0	<p><b>SMS: 从模式选择 (Slave mode selection)</b> 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 – 根据另一个输入的电平, 计数器在TI1FP1和TI2FP2的边沿向上/向下计数。 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> <p>注: 如果TI1F_EN被选为触发输入 (TS = 100) 时, 不要使用门控模式。这是因为, TI1F_ED在每次TI1F变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。</p>

表 28. TIMx 内部触发连接

从定时器	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM3	TIM4	TIM1	TIM2	TIM7
TIM4	TIM5	TIM1	TIM2	TIM3

#### 12.4.4 DMA/中断使能寄存器 (TIMX\_DIER)

偏移地址: 0x0C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	保留	CC4 DE	CC3 DE	CC2 DE	CC1 DE	UDE	保留	TIE	保留	CC4 IE	CC3 IE	CC2 IE	CC1 IE	UIE

位31: 15	保留
位14	<b>TDE:</b> 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
位13	保留, 始终读为0。
位12	<b>CC4DE:</b> 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
位11	<b>CC3DE:</b> 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
位10	<b>CC2DE:</b> 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
位9	<b>CC1DE:</b> 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
位8	<b>UDE:</b> 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
位7	保留, 始终读为0。
位6	<b>TIE:</b> 触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断 1: 使能触发中断
位5	保留, 始终读为0
位4	<b>CC4IE:</b> 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
位3	<b>CC3IE:</b> 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
位2	<b>CC2IE:</b> 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
位1	<b>CC1IE:</b> 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
位0	<b>UIE:</b> 允许更新中断 (Update interrupt enable) 0: 禁止更新中断 1: 允许更新中断

### 12.4.5 状态寄存器 (TIMx\_SR)

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4 OF	CC3 OF	CC2 OF	CC1 OF	保留	TIF	保留	CC4 IF	CC3 IF	CC2 IF	CC1 IF	UIF			
	rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	

位31: 13	保留
位12	<b>CC4OF:</b> 捕获/比较4重复捕获标记 (Capture/Compare 4 overcapture flag) 参见CC1OF描述。
位11	<b>CC3OF:</b> 捕获/比较3重复捕获标记 (Capture/Compare 3 overcapture flag) 参见CC1OF描述。
位10	<b>CC2OF:</b> 捕获/比较2重复捕获标记 (Capture/Compare 2 overcapture flag) 参见CC1OF描述。
位9	<b>CC1OF:</b> 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
位8: 7	保留，始终读为0。
位6	<b>TIF:</b> 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，或或门控模式下的任一边沿）时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生 1: 触发器中断等待响应
位5	保留，始终读为0。
位4	<b>CC4IF:</b> 捕获/比较4 中断标记 (Capture/Compare 4 interrupt flag) 参考CC1IF描述。
位3	<b>CC3IF:</b> 捕获/比较3 中断标记 (Capture/Compare 3 interrupt flag) 参考CC1IF描述。
位2	<b>CC2IF:</b> 捕获/比较2 中断标记 (Capture/Compare 2 interrupt flag) 参考CC1IF描述。

位1	<p><b>CC1IF:</b> 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 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）</p>
位0	<p><b>UIF:</b> 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: 更新事件等待响应。当寄存器被更新时该位由硬件置1 - 若TIMx_CR1寄存器的UDIS = 0，当REP_CNT = 0时产生更新事件（重复向下计数器上溢或下溢时）； - 若TIMx_CR1寄存器的UDIS = 0、URS = 0，当TIMx_EGR寄存器的UG = 1时产生更新事件 (软件对计数器CNT重新初始化)； - 若TIMx_CR1寄存器的UDIS = 0、URS = 0，当计数器CNT被触发事件重初始化时产生更新事件。（参考同步控制寄存器的说明）</p>

#### 12.4.6 事件产生寄存器 (TIMx\_EGR)

偏移地址: 0x14

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					保留			TG	保留	CC4G	CC3G	CC2G	CC1G	UG	
									W	W	W	W	W	W	W

位31: 7	保留，始终读为0。
位6	<p><b>TG:</b> 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</p>
位5	保留，始终读为0。
位4	<b>CC4G:</b> 产生捕获/比较4事件 (Capture/compare 4 generation) 参考CC1G描述。
位3	<b>CC3G:</b> 产生捕获/比较3事件 (Capture/compare 3 generation) 参考CC1G描述。
位2	<b>CC2G:</b> 产生捕获/比较2事件 (Capture/compare 2 generation) 参考CC1G描述。

位1	<p><b>CC1G:</b> 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。</p>
位0	<p><b>UG:</b> 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>

#### 12.4.7 捕捉/比较模式寄存器1 (TIMx\_CCMR1)

偏移地址: 0x18

复位值: 0x0000

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CCxS 定义。该寄存器其他位的作用和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C2C E	OC2M		OC2P E	OC2F E	CC2S	0C1C E	0C1M	OC1P E	OC1F E	CC1S					
	IC2F		IC2PSC			IC1F		IC1PSC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

输出比较模式：

位15	<b>OC2CE:</b> 输出比较2清0使能 (Output compare 2 clear enable)
位14: 12	<b>OC2M[2: 0]:</b> 输出比较2模式 (Output compare 2 mode)
位11	<b>OC2PE:</b> 输出比较2预装载使能 (Output compare 2 preload enable)
位10	<b>OC2FE:</b> 输出比较2快速使能 (Output compare 2 fast enable)

位9: 8	<p><b>CC2S[1: 0]:</b> 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>
位7	<p><b>OC1CE:</b> 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>
位6: 4	<p><b>OC1M[2: 0]:</b> 输出比较1模式 (Output compare 1 enable)      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)。      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。      注1: 一旦LOCK级别设为3 (TIMx_BDTR寄存器中的LOCK位) 并且CC1S = 00 (该通道配置成输出) 则该位不能被修改。      注2: 在PWM模式1或PWM模式2中, 只有当比较结果改变了或在输出比较模式中从冻结模式切换到PWM模式时, OC1REF电平才改变。</p>
位3	<p><b>OC1PE:</b> 输出比较1预装载使能 (Output compare 1 preload enable)      0: 禁止TIMx_CCR1寄存器的预装载功能, 可随时写入TIMx_CCR1寄存器, 并且新写入的数值立即起作用。      1: 开启TIMx_CCR1寄存器的预装载功能, 读写操作仅对预装载寄存器操作, TIMx_CCR1的预装载值在更新事件到来时被加载至当前寄存器中。注1: 一旦LOCK级别设为3 (TIMx_BDTR寄存器中的LOCK位) 并且CC1S = 00 (该通道配置成输出) 则该位不能被修改。      注2: 仅在单脉冲模式下 (TIMx_CR1寄存器的OPM = 1), 可以在未确认预装载寄存器情况下使用PWM模式, 否则其动作不确定。</p>

位2	<b>OC1FE:</b> 输出比较1快速使能 (Output compare 1 fast enable) 该位用于加快CC输出对触发输入事件的响应。 0: 根据计数器与CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活CC1输出的最小延时为5个时钟周期。 1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC被设置为比较电平而与 比较结果无关。采样触发器的有效沿和CC1输出间的延时被缩短为3个时钟周期。OCFE只在通道被配置成PWM1或PWM2模式时起作用。
位1: 0	<b>CC1S[1: 0]:</b> 捕获/比较1选择 (Capture/Compare 1 selection) 这2位定义通道的方向 (输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC1映射在TI1上; 10: CC1通道被配置为输入, IC1映射在TI2上; 11: CC1通道被配置为输入, IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时 (由 TIMx_SMCR 寄存器的TS位选择)。 注: CC1S仅在通道关闭时 (TIMx_CCER寄存器的CC1E = 0) 才是可写的。

**输入捕获模式:**

位15: 12	<b>IC2F[3: 0]:</b> 输入捕获2滤波器 (Input capture 2 filter)
位11: 10	<b>IC2PSC[1: 0]:</b> 输入/捕获2预分频器 (input capture 2 prescaler)
位9: 8	<b>CC2S[1: 0]:</b> 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。

位7: 4	<p><b>IC1F[3: 0]:</b> 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> <ul style="list-style-type: none"> <li>0000: 无滤波器，以f<sub>DTS</sub>采样</li> <li>1000: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 6</li> <li>0001: 采样频率f<sub>SAMPLING</sub> = f<sub>CLOCK_INT</sub>, N = 2</li> <li>1001: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 8</li> <li>0010: 采样频率f<sub>SAMPLING</sub> = f<sub>CLOCK_INT</sub>, N = 4</li> <li>1010: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 5</li> <li>0011: 采样频率f<sub>SAMPLING</sub> = f<sub>CLOCK_INT</sub>, N = 8</li> <li>1011: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 6</li> <li>0100: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 6</li> <li>1100: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 8</li> <li>0101: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 8</li> <li>1101: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 5</li> <li>0110: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 6</li> <li>1110: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 6</li> <li>0111: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 8</li> <li>1111: 采样频率f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 8</li> </ul>
位3: 2	<p><b>IC1PSC[1: 0]:</b> 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> <ul style="list-style-type: none"> <li>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；</li> <li>01: 每2个事件触发一次捕获；</li> <li>10: 每4个事件触发一次捕获；</li> <li>11: 每8个事件触发一次捕获。</li> </ul>
位1: 0	<p><b>CC1S[1: 0]:</b> 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> <ul style="list-style-type: none"> <li>00: CC1通道被配置为输出；</li> <li>01: CC1通道被配置为输入，IC1映射在TI1上；</li> <li>10: CC1通道被配置为输入，IC1映射在TI2上；</li> <li>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）。</li> </ul> <p>注：CC1S仅在通道关闭时 (TIMx_CCER寄存器的CC1E = 0) 才是可写的。</p>

### 12.4.8 捕捉/比较模式寄存器 2 (TIMx\_CCMR2)

偏移地址: 0x1C

复位值: 0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C4C E	OC4M		OC4P E	OC4F E		CC4S	0C3C E	0C3M		OC3P E	OC3F E		CC3S		
	IC4F			IC4PSC				IC3F		IC3PSC					
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

### 输出比较模式

位15	<b>OC4CE:</b> 输出比较4清0使能 (Output compare 4 clear enable)
位14: 12	<b>OC4M[2: 0]:</b> 输出比较4模式 (Output compare 4 mode)
位11	<b>OC4PE:</b> 输出比较4预装载使能 (Output compare 4 preload enable)
位10	<b>OC4FE:</b> 输出比较4快速使能 (Output compare 4 fast enable)
位9: 8	<b>CC4S[1: 0]:</b> 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。
位7	<b>OC3CE:</b> 输出比较3清0使能 (Output compare 3 clear enable)
位6: 4	<b>OC3M[2: 0]:</b> 输出比较3模式 (Output compare 3 mode)
位3	<b>OC3PE:</b> 输出比较3预装载使能 (Output compare 3 preload enable)
位2	<b>OC3FE:</b> 输出比较3快速使能 (Output compare 3 fast enable)
位1: 0	<b>CC3S[1: 0]:</b> 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。

### 输入比较模式

位15: 12	<b>IC4F[3: 0]:</b> 输入捕获4滤波器 (Input capture 4 filter)
位11: 10	<b>IC4PSC[1: 0]:</b> 输入/捕获4预分频器 (input capture 4 prescaler)
位9: 8	<b>CC4S[1: 0]:</b> 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。
位7: 4	<b>IC3F[3: 0]:</b> 输入捕获3滤波器 (Input capture 3 filter)
位3: 2	<b>IC3PSC[1: 0]:</b> 输入/捕获3预分频器 (Input capture 3 prescaler)
位1: 0	<b>CC3S[1: 0]:</b> 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。

#### 12.4.9 捕捉/比较使能寄存器 (TIMx\_CCER)

偏移地址: 0x20

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	CC4P	CC4E	保留	CC3P	CC3E	保留	CC2P	CC1E	保留	CC1P	CC1E				
	W	W		W	W		W	W		W	W		W	W	

位15: 14	保留, 始终读为0。
位13	<b>CC4P:</b> 输入/捕获4输出极性 (Capture/Compare 4 output polarity) 参考CC1P的描述。
位12	<b>CC4E:</b> 输入/捕获4输出使能 (Capture/Compare 4 output enable) 参考CC1E的描述。
位11: 10	保留, 始终读为0。
位9	<b>CC3P:</b> 输入/捕获3输出极性 (Capture/Compare 3 output polarity) 参考CC1P的描述。
位8	<b>CC3E:</b> 输入/捕获3输出使能 (Capture/Compare 3 output enable) 参考CC1E的描述。
位7: 6	保留, 始终读为0。

位5	<b>CC2P:</b> 输入/捕获2输出极性 (Capture/Compare 2 output polarity) 参考CC1P的描述。
位4	<b>CC2E:</b> 输入/捕获2输出使能 (Capture/Compare 2 output enable) 参考CC1E的描述。
位3: 2	保留, 始终读为0。
位1	<b>CC1P:</b> 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LCCK位) 设为3或2, 则该位不能被修改。
位0	<b>CC1E:</b> 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 29. 标准 OCx 通道的输出控制位

CCxE位	OCx输出状态
0	禁止输出 (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + 极性, OCx_EN = 1

注: 管脚连接到标准的 OCx 通道的外部 I/O 管脚的状态, 取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

#### 12.4.10 计数器 (TIMx\_CNT)

偏移地址: 0x24

复位值: 0x0000

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

位31: 0	CNT[31: 0]: 计数器的值 (Counter value)
--------	-----------------------------------

#### 12.4.11 预分频器 (TIMx\_PSC)

偏移地址: 0x28

复位值: 0x0000

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

位15: 0	PSC[15: 0]: 预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 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'。
--------	--

#### 12.4.12 自动装载寄存器 (TIMx\_ARR)

偏移地址: 0x2C

复位值: 0x0000

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

位31: 0	<b>ARR[31: 0]:</b> 自动重装载的值 (Auto reload value) ARR包含了将要装载入实际的自动重装载寄存器的数值。 详细参考13.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。
--------	---

#### 12.4.13 捕获/比较寄存器 1 (TIMx\_CCR1)

偏移地址: 0x34

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0	<b>CCR1[31: 0]:</b> 捕获/比较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。
--------	--

#### 12.4.14 捕获/比较寄存器 2 (TIMx\_CCR2)

偏移地址: 0x38

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0	<p><b>CCR2[31: 0]:</b> 捕获/比较2的值 (Capture/Compare 2 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) 中未选择预装载特性, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较2寄存器中。当前捕获/比较寄存器参与同计数器TIMx_CNT的比较, 并在OC2端口上产生输出信号。</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>
--------	---

#### 12.4.15 捕获/比较寄存器 3 (TIMx\_CCR3)

偏移地址: 0x3C

复位值: 0x0000

CCR3[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0	<p><b>CCR3[31: 0]:</b> 捕获/比较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) 中未选择预装载特性, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较3寄存器中。当前捕获/比较寄存器参与同计数器TIMx_CNT的比较, 并在OC3端口上产生输出信号。</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>
--------	---

#### 12.4.16 捕获/比较寄存器 4 (TIMx\_CCR4)

偏移地址: 0x40

复位值: 0x0000

CCR4[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

<p>位31: 0</p>	<p><b>CCR4[31: 0]:</b> 捕获/比较4的值 (Capture/Compare 4 value) 若CC4通道配置为输出: CCR4包含了装入当前捕获/比较4寄存器的值（预装载值）。 如果在TIMx_CCMR4寄存器 (OC4PE位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较4寄存器中。当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC4端口上产生输出信号。 若CC4通道配置为输入: CCR4包含了由上一次输入捕获4事件 (IC4) 传输的计数器值。</p>
---------------	--

#### 12.4.17 DMA 控制寄存器 (TIMx\_DCR)

偏移地址: 0x48

复位值: 0x0000

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

位15: 13	保留, 始终读为0。
位12: 8	<p><b>DBL[4: 0]:</b> 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</p>
位7: 5	保留, 始终读为0。
位4: 0	<p><b>DBA[4: 0]:</b> DMA基地址 (DMA base address) 这些位定义了DMA在连续模式下的基地址 (当对TIMx_DMAR寄存器进行读或写时), DBA定义为从TIMx_CR1寄存器所在地址开始的偏移量: 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR .....</p>

#### 12.4.18 连续模式的 DMA 地址 (TIMx\_DMAR)

偏移地址: 0x48

复位值: 0x0000

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

位15: 0	<b>DMAB[15: 0]:</b> 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。

## 13. 通用定时器 (TIM5/6/7)

### 13.1 TIMx 简介

通用定时器是一个通过可编程预分频器驱动的 32 位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

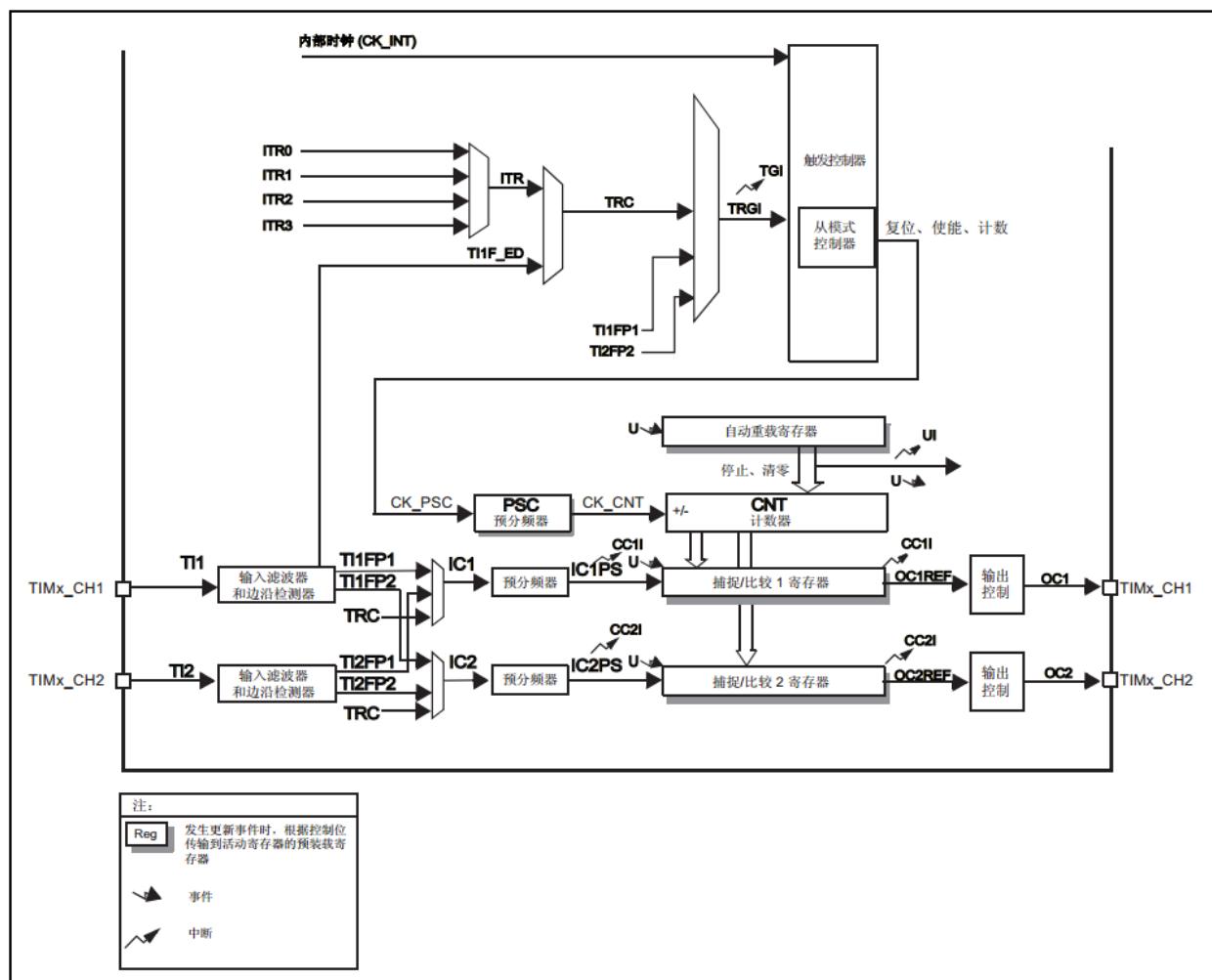
TIM5/TIM6/TIM7 定时器是完全独立的，而且没有互相共享任何资源。它们可以一起同步操作。

### 13.2 TIM5/TIM6/TIM7 主要功能

通用 TIM5/TIM6/TIM7 定时器功能包括：

- 32 位向上自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 2 个独立的通道
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路
- 如下事件发生时产生中断：
  - 更新：计数器向上溢出，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

图 110. 通用定时器框图



### 13.3 TIM5/TIM6/TIM7 功能描述

#### 13.3.1 时基单元

可编程通用定时器的主要部分是一个 32 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写，时基单元包含：

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

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 **TIMX\_CR1** 寄存器中的自动装载预装载使能位 (ARPE) 的设置，预装载寄存器的内容被立即或在 每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件 (向下计数时的下溢条件) 并且 **TIMX\_CR1** 寄存器中的 UDIS 等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 **TIMX\_CR1** 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。（有关计数器使能的细节，请参见控制器的从模式描述）。

## 预分频器描述

预分频器可以将计数器的时钟频率按 1 ~ 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 32 位计数器。因为这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下面两个图分别给出了在预分频器运行时，更改计数器参数的例子。

图 111. 当预分频器的参数从 1 变到 2 时，计数器的时序图

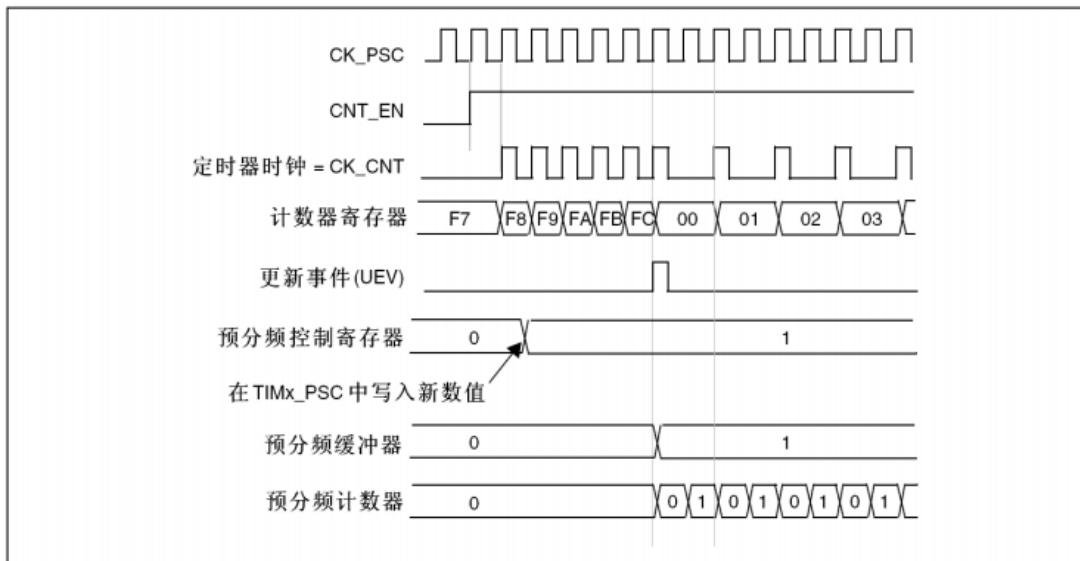
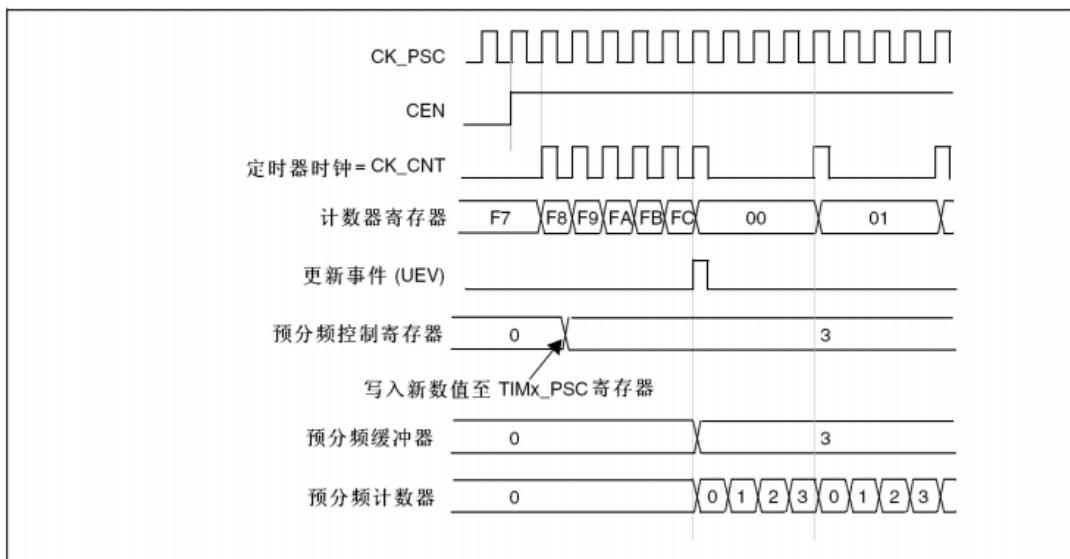


图 112. 当预分频器的参数从 1 变到 4 时，计数器的时序图



### 13.3.2 计数模式

#### 递增计数模式

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

每次计数器溢出时可以产生更新事件，在 **TIMx\_EGR** 寄存器中设置 **UG** 位（通过软件方式或者使用从模式控制器）也同样可以产生一个更新事件。

设置 **TIMx\_CR1** 寄存器中的 **UDIS** 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 **UDIS** 位被清 0 之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 **TIMx\_CR1** 寄存器中的 **URS** 位（选择更新请求），设置 **UG** 位将产生一个更新事件 **UEV**，但硬件不设置 **UIF** 标志（即不产生中断）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

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

- 预分频器的缓冲区被置入预装载寄存器的值（**TIMx\_PSC** 寄存器的内容）
- 自动装载影子寄存器被重新置入预装载寄存器的值（**TIMx\_ARR**）

下图给出一些例子，当 **TIMx\_ARR = 0x36** 时计数器在不同时钟频率下的动作：

图 113. 计数器时序图，内部时钟分频因子为 1

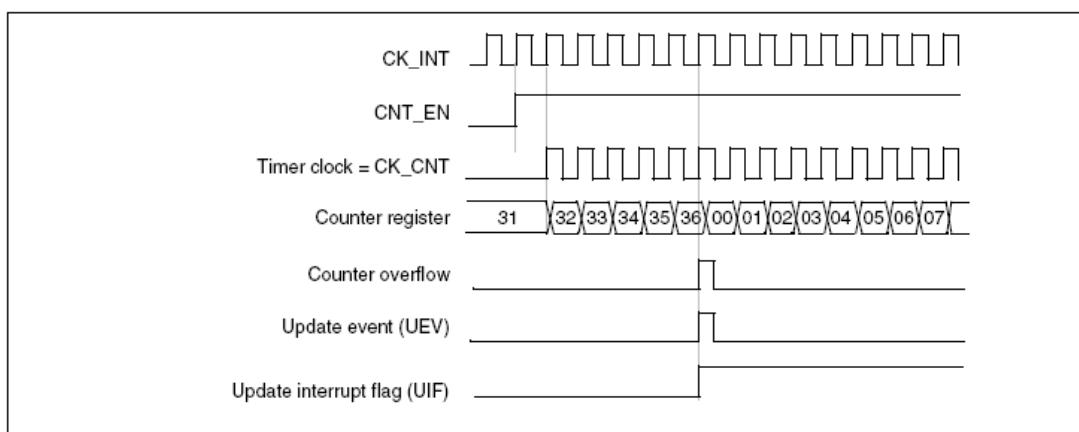


图 114. 计数器时序图，内部时钟分频因子为 2

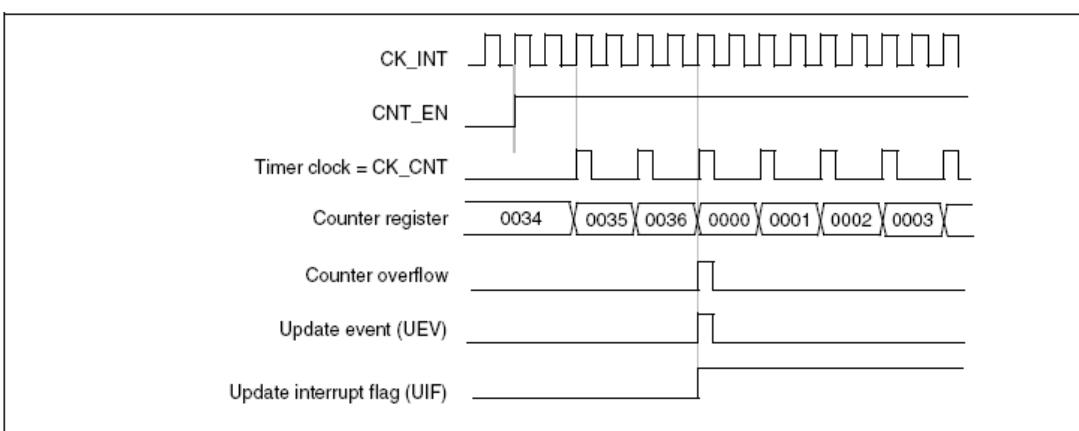


图 115. 计数器时序图，内部时钟分频因子为 4

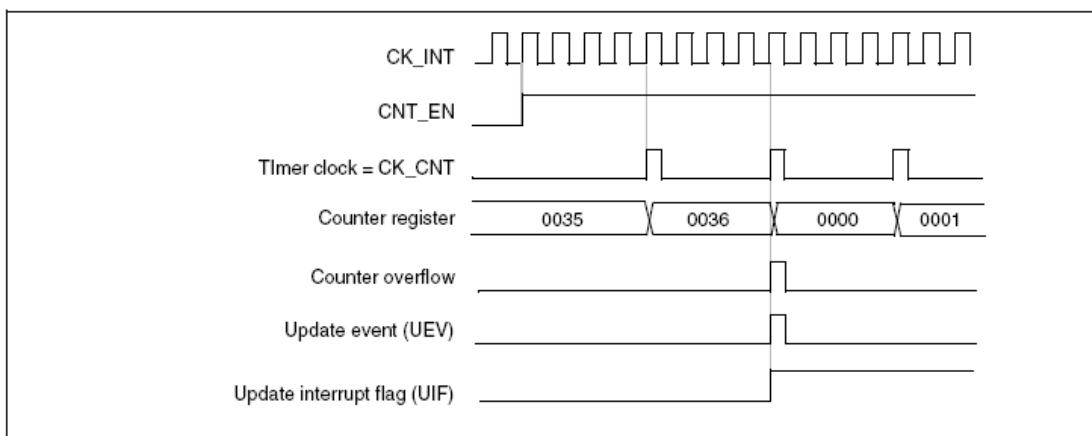


图 116. 计数器时序图，内部时钟分频因子为 N

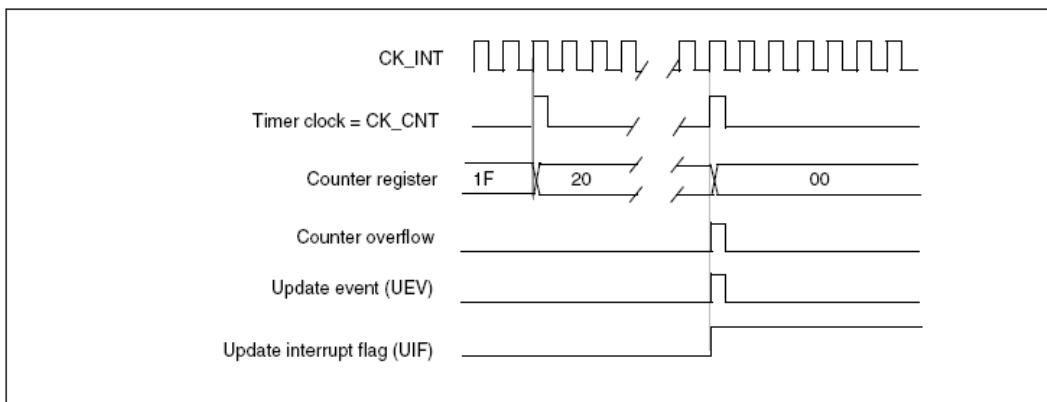


图 117. 计数器时序图，当 ARPE = 0 时的更新事件（TIMx\_ARR 没有预装入）

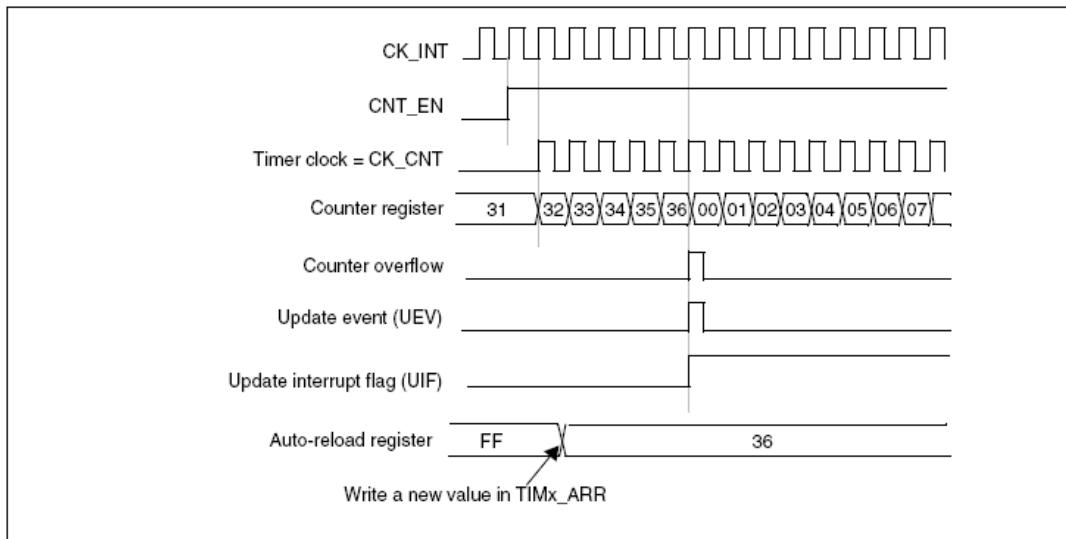
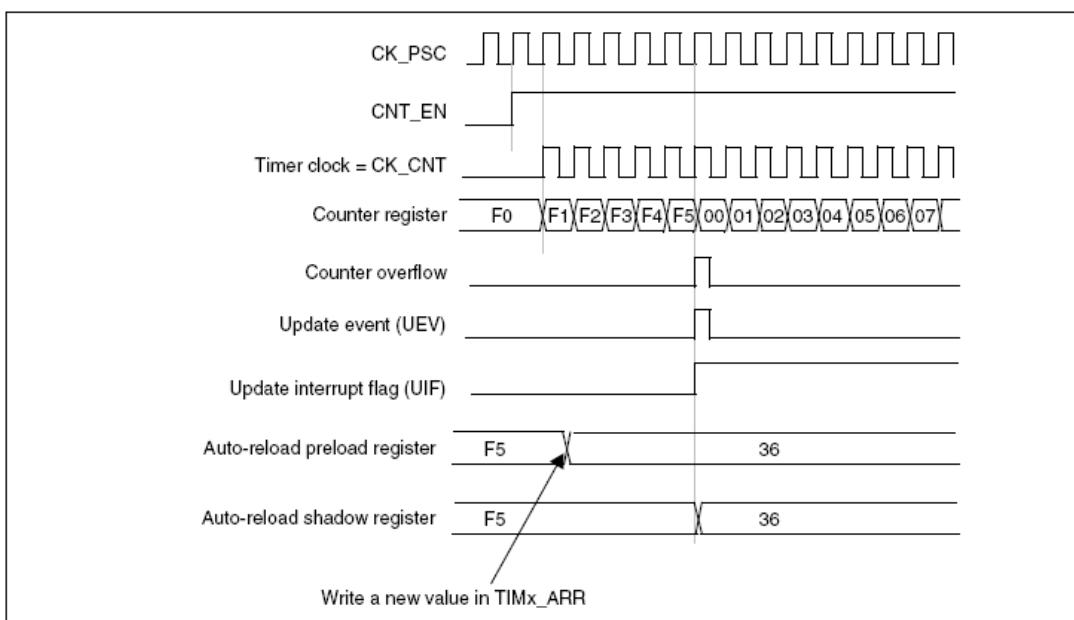


图 118. 计数器时序图, 当 ARPE = 1 时的更新事件 (预装入了 TIMx\_ARR)



### 13.3.3 时钟选择

计数器时钟可由下列时钟源提供：

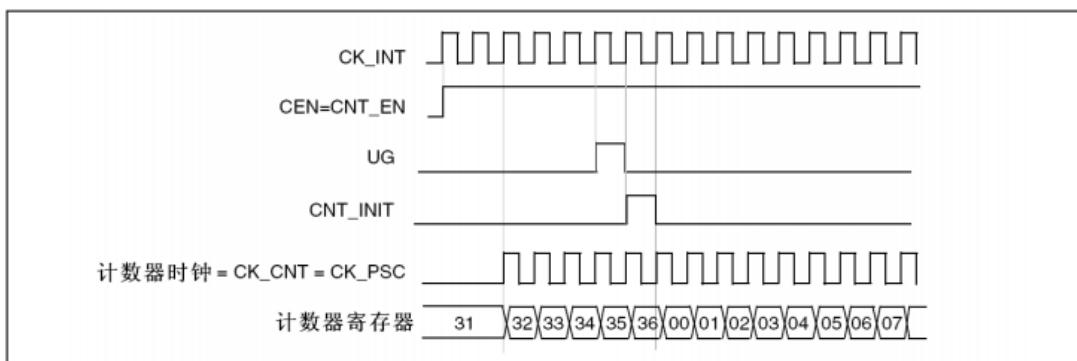
- 内部时钟 (**CK\_INT**)
- 外部时钟模式 1：外部输入脚 (**TIx**)
- 内部触发输入 (**ITRx**)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。

#### 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)。一旦 **CEN** 位被写成 1，预分频器的时钟就由内部时钟 **CK\_INT** 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

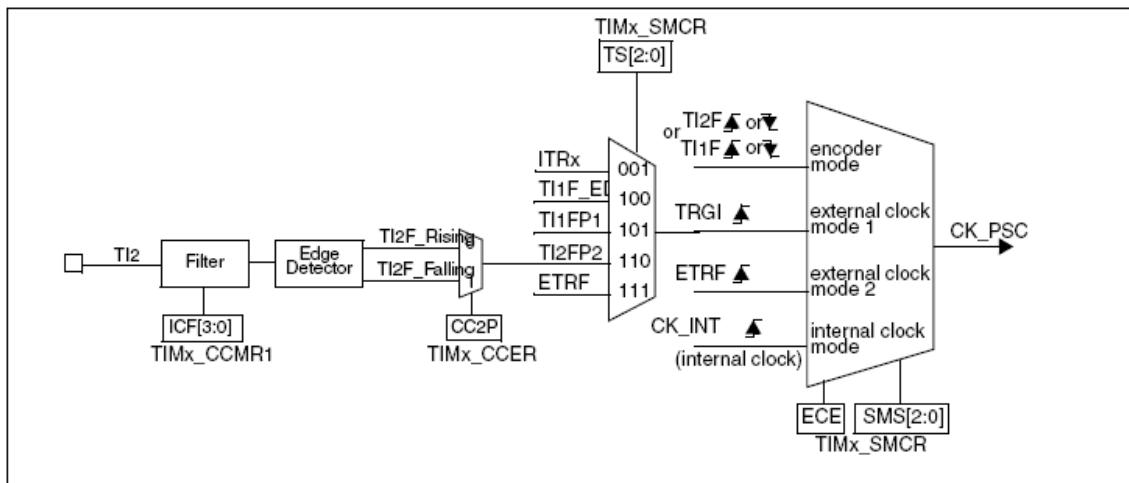
图 119. 一般模式下的控制电路，内部时钟分频因子为 1



## 外部时钟源模式 1

当 TIMx\_SMCR 寄存器的 SMS = 111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

图 120. TI2 外部时钟连接例子



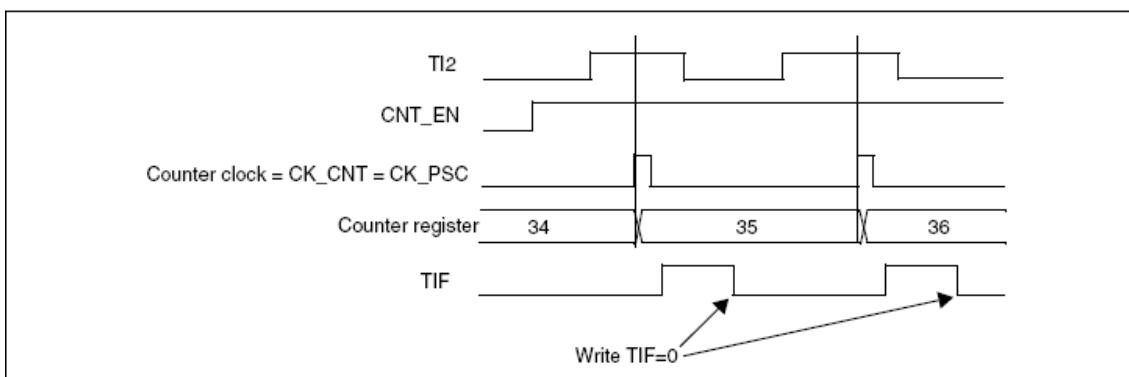
例如，要配置向上计数器在 TI2 输入端的上升沿计数，使用下列步骤：

7. 配置 TIMx\_CCMR1 寄存器 CC2S = 01，配置通道 2 检测 TI2 输入的上升沿
8. 配置 TIMx\_CCMR1 寄存器的 IC2F[3: 0]，选择输入滤波器带宽（如果不需要滤波器，保持 IC2F = 0000）
- 注：捕获预分频器不用作触发，所以不需要对它进行配置
9. 配置 TIMx\_CCER 寄存器的 CC2P = 0，选定上升沿极性
10. 配置 TIMx\_SMCR 寄存器的 SMS = 111，选择定时器外部时钟模式 1
11. 配置 TIMx\_SMCR 寄存器中的 TS = 110，选定 TI2 作为触发输入源
12. 设置 TIMx\_CR1 寄存器的 CEN = 1，启动计数器

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时取决于在 TI2 输入端的重新同步电路。

图 121. 外部时钟模式 1 下的控制电路

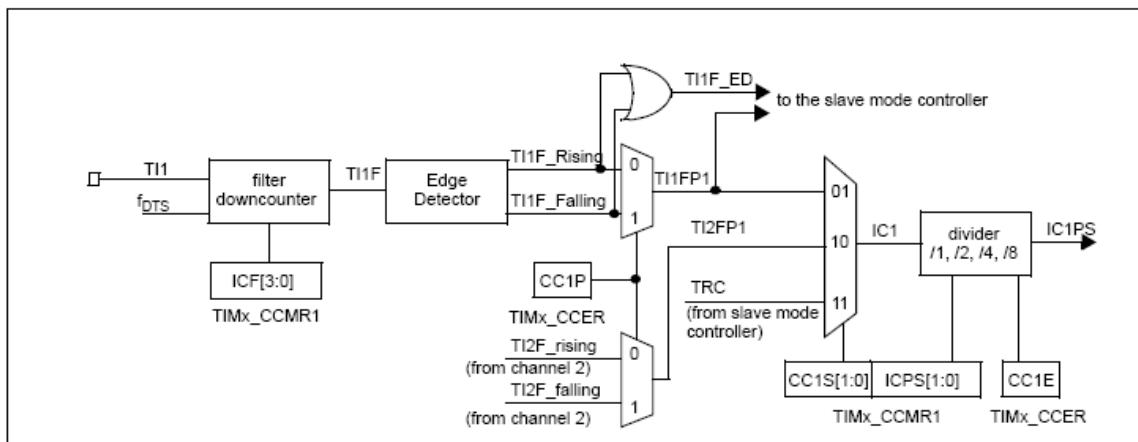


### 13.3.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

下面几张图是一个捕获/比较通道概览。输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边缘监测器产生一个信号 (TIxFPx)，它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器 (ICxPS)。

图 122. 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 123. 捕获/比较通道 1 的主电路

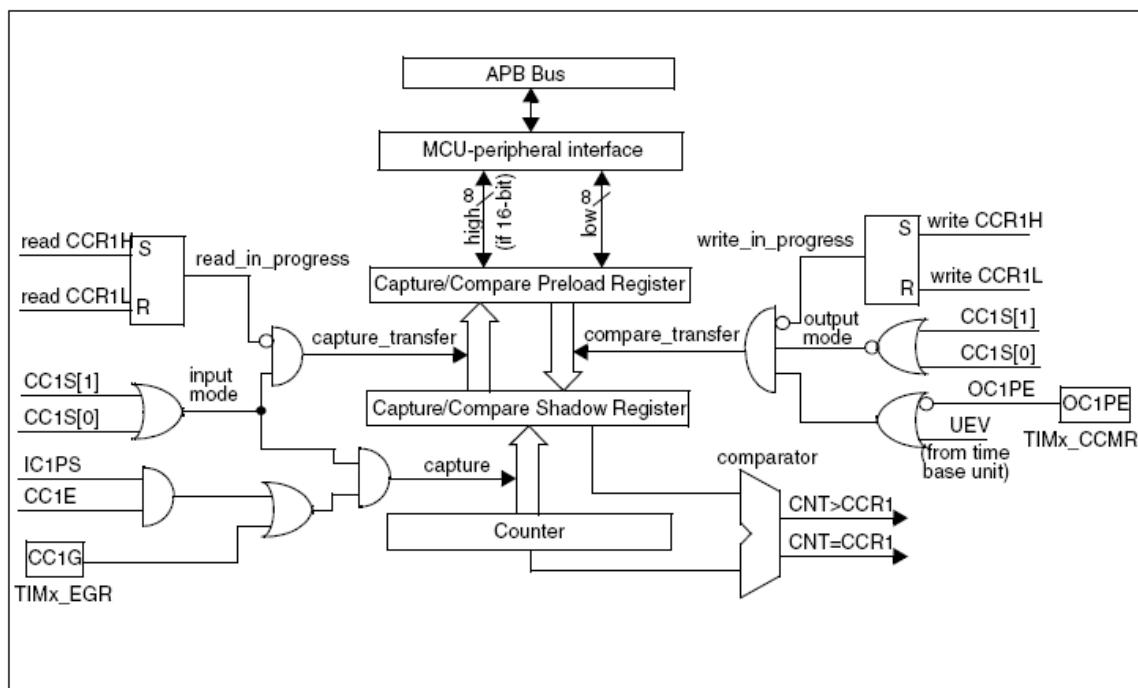
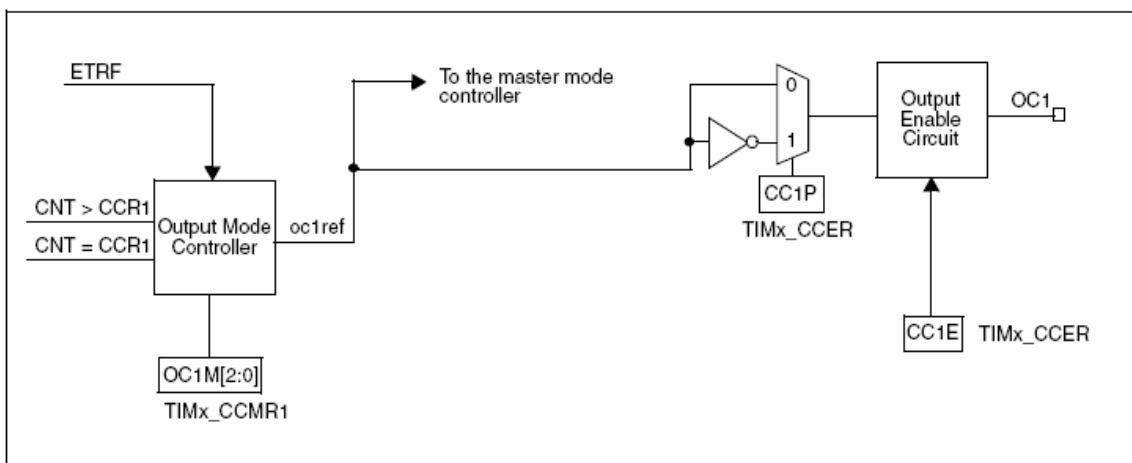


图 124. 捕获/比较通道的输出部分（通道 1）



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。

### 13.3.5 输入捕捉模式

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。当捕获事件发生时，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，则将产生中断操作。如果捕获事件发生时 CC<sub>x</sub>IF 标志已经为高，那么重复捕获标志 CC<sub>x</sub>OF (TIM<sub>x</sub>\_SR 寄存器) 被置 1。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 = 01，一旦 CC1S 不为 00 时，通道被配置为输入，并且 TM1\_CCR1 寄存器变为只读。
- 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 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（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，通过设置 TIM<sub>x</sub>\_DIER 寄存器中的 CC1IE 位允许相关中断请求。

发生当一个输入捕获时：

- 当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。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注：设置  $TIMx\_EGR$  寄存器中相应的  $CCxG$  位，可以通过软件产生输入捕获中断请求。

### 13.3.6 PWM 输入模式

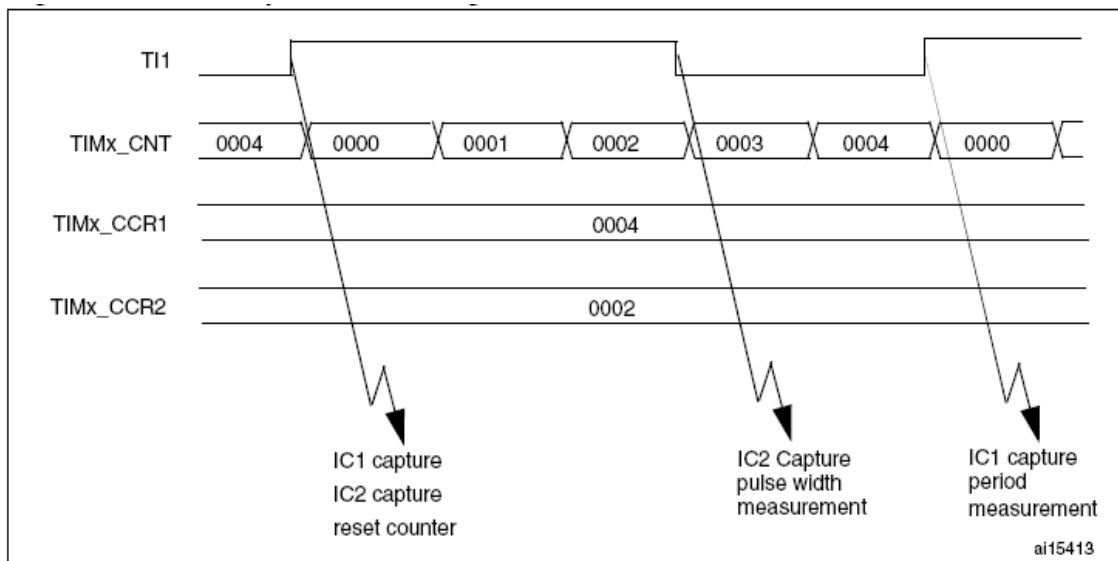
该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个  $ICx$  信号被映射同一个  $TIx$  输入。
- 这 2 个  $ICx$  信号为边沿有效，但是极性相反。
- 其中一个  $TIxFP$  信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到  $TI1$  上的 PWM 信号的长度 ( $TIMx\_CCR1$  寄存器) 和占空比 ( $TIMx\_CCR2$  寄存器)，具体步骤如下（取决于  $CK\_INT$  的频率和预分频器的值）

- 选择  $TIMx\_CCR1$  的有效输入：置  $TIMx\_CCMR1$  寄存器的  $CC1S = 01$ （选择  $TI1$ ）。
- 选择  $TI1FP1$  的有效极性（用来捕获数据到  $TIMx\_CCR1$  中和清除计数器）：置  $CC1P = 0$ （上升沿有效）。
- 选择  $TIMx\_CCR2$  的有效输入：置  $TIMx\_CCMR1$  寄存器的  $CC2S = 10$ （选择  $TI1$ ）。
- 选择  $TI1FP2$  的有效极性（捕获数据到  $TIMx\_CCR2$ ）：置  $CC2P = 1$ （下降沿有效）。
- 选择有效的触发输入信号：置  $TIMx\_SMCR$  寄存器中的  $TS = 101$ （选择  $TI1FP1$ ）。
- 配置从模式控制器为复位模式：置  $TIMx\_SMCR$  中的  $SMS = 100$ 。
- 使能捕获：置  $TIMx\_CCER$  寄存器中  $CC1E = 1$  且  $CC2E = 1$ 。

图 125. PWM 输入模式时序



由于只有  $TI1FP1$  和  $TI2FP2$  连到了从模式控制器。所以 PWM 输入模式只能使用  $TIMx\_CH1 / TIMx\_CH2$  信号。

### 13.3.7 强制输出模式

在输出模式 ( $TIMx\_CCMRx$  寄存器中  $CCxS = 00$ ) 下，输出比较信号 ( $OCxREF$  和相应的  $OCx$ ) 能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 **TIMx\_CCMRx** 寄存器中相应的 **OCxM = 101**，即可强置输出比较信号（**OCxREF/OCx**）为有效状态。这样 **OCxREF** 被强置为高电平（**OCxREF** 始终为高电平有效），同时 **OCx** 得到 **CCxP** 极性位相反的值。

例如：**CCxP = 0**（**OCx** 高电平有效），则 **OCx** 被强置为高电平。

置 **TIMx\_CCMRx** 寄存器中的 **OCxM = 100**，可强置 **OCxREF** 信号为低。

该模式下，在 **TIMx\_CCRx** 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断。这将会在下面的输出比较模式一节中介绍。

### 13.3.8 输出比较模式

此项功能是用来控制一个输出波形或者指示何时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管脚上。在比较匹配时，输出管脚可以保持它的电平（**OCxM = 000**）、被设置成有效电平（**OCxM = 001**）、被设置成无有效电平（**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。

**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。

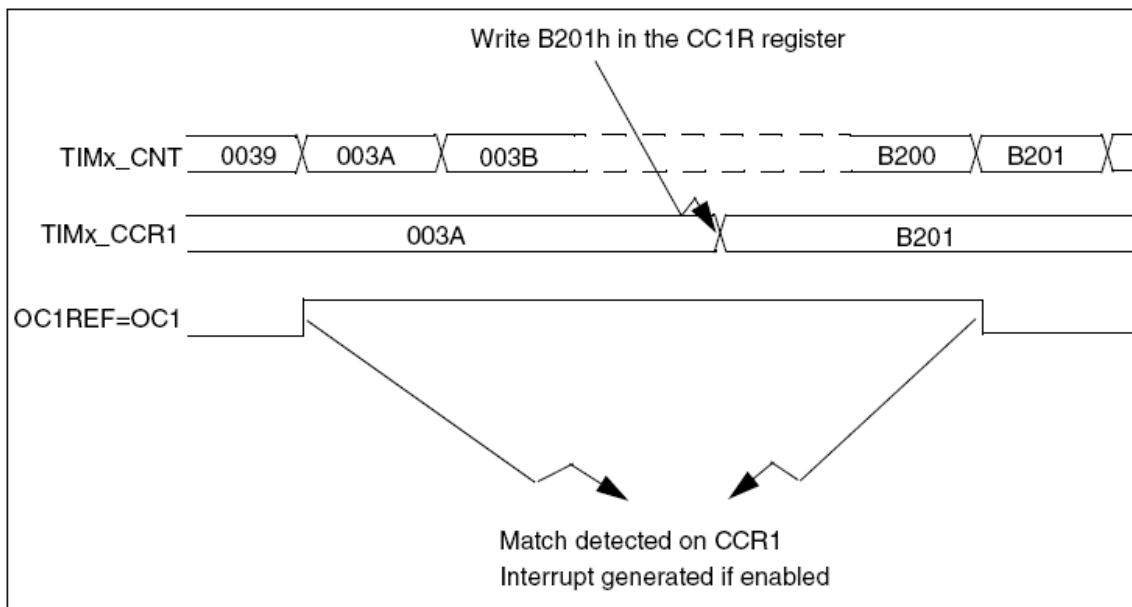
同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

6. 选择计数器时钟（内部，外部，预分频器）
7. 将相应的数据写入 **TIMx\_ARR** 和 **TIMx\_CCRx** 寄存器中
8. 选择输出模式，例如：必须设置 **OCxM = '011'**、**OCxPE = '0'**、**CCxP = '0'** 和 **CCxE = '1'**，当计数器 **CNT** 与 **CCRx** 匹配时翻转 **OCx** 的输出管脚，**CCRx** 预装载未用，开启 **OCx** 输出且高电平有效。
9. 设置 **TIMx\_CR1** 寄存器的 **CEN** 位启动计数器

**TIMx\_CCRx** 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（**OCxPE = '0'**，否则 **TIMx\_CCRx** 影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

图 126. 输出比较模式，翻转 OC1



### 13.3.9 PWM 模式

脉冲宽度调制模式可以产生一个由 **TIMx\_ARR** 寄存器确定频率、由 **TIMx\_CCRx** 寄存器确定占空比的信号。

在 **TIMx\_CCMRx** 寄存器中的 **OCxM** 位写入‘110’（PWM 模式 1）或‘111’（PWM 模式 2），能够独立地设置每个 **OCx** 输出通道产生一路 PWM。必须设置 **TIMx\_CCMRx** 寄存器 **OCxPE** 位以使能相应的预装载寄存器，最后还要设置 **TIMx\_CR1** 寄存器的 **ARPE** 位使能自动重装载的预装载寄存器（在向上计数或中心对称模式中）。

因为仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 **TIMx\_EGR** 寄存器中的 **UG** 位来初始化所有的寄存器。

**OCx** 的极性可以通过软件在 **TIMx\_CCER** 寄存器中的 **CCxP** 位设置，它可以设置为高电平有效或低电平有效。**TIMx\_CCER** 寄存器中的 **CCxE** 位控制 **OCx** 输出使能。详见 **TIMx\_CCERx** 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，**TIMx\_CNT** 和 **TIM1\_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 **TIM1\_CCRx ≤ TIM1\_CNT** 或者 **TIM1\_CNT ≤ TIM1\_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：

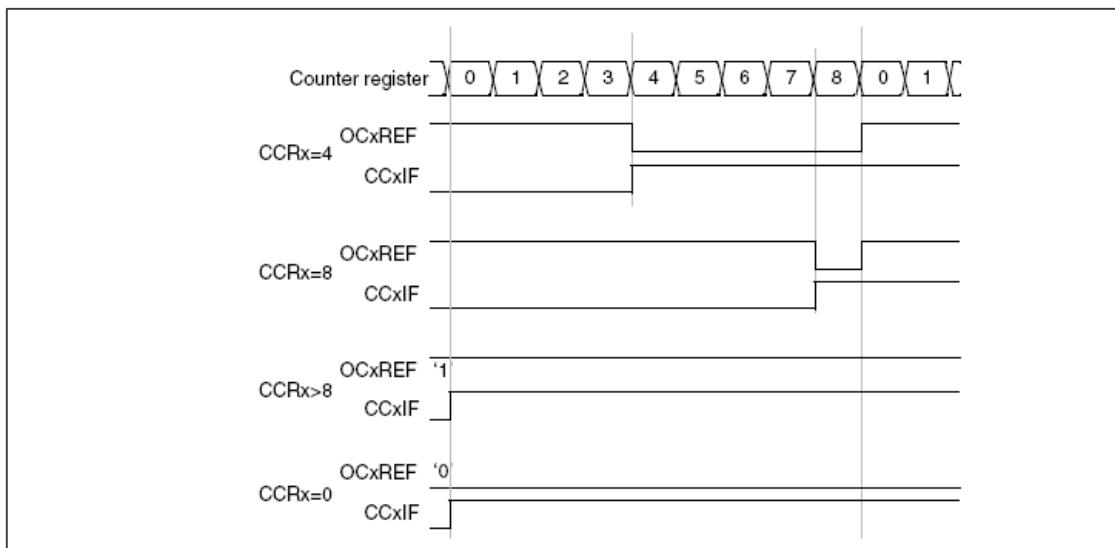
- 当比较的结果改变，或
- 当输出比较模式（**TIMx\_CCMRx** 寄存器中的 **OCxM** 位）从‘冻结’（无比较，**OCxM = ‘000’**）切换到某个 PWM 模式（**OCxM = ‘110’或‘111’**）。

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。

## PWM 边沿对齐模式

下面是一个 PWM 模式 1 的例子。当  $\text{TIMx_CNT} < \text{TIMx_CCRx}$  时 PWM 信号参考  $\text{OCxREF}$  为高，否则为低。如果  $\text{TIMx_CCRx}$  中的比较值大于自动重装载值 ( $\text{TIMx_ARR}$ )，则  $\text{OCxREF}$  保持为‘1’。如果比较值为 0，则  $\text{OCxREF}$  保持为‘0’。下图为  $\text{TIMx_ARR} = 8$  时边沿对齐的 PWM 波形实例。

图 127. 边沿对齐的 PWM 波形 ( $\text{ARR} = 8$ )



### 13.3.10 单脉冲模式

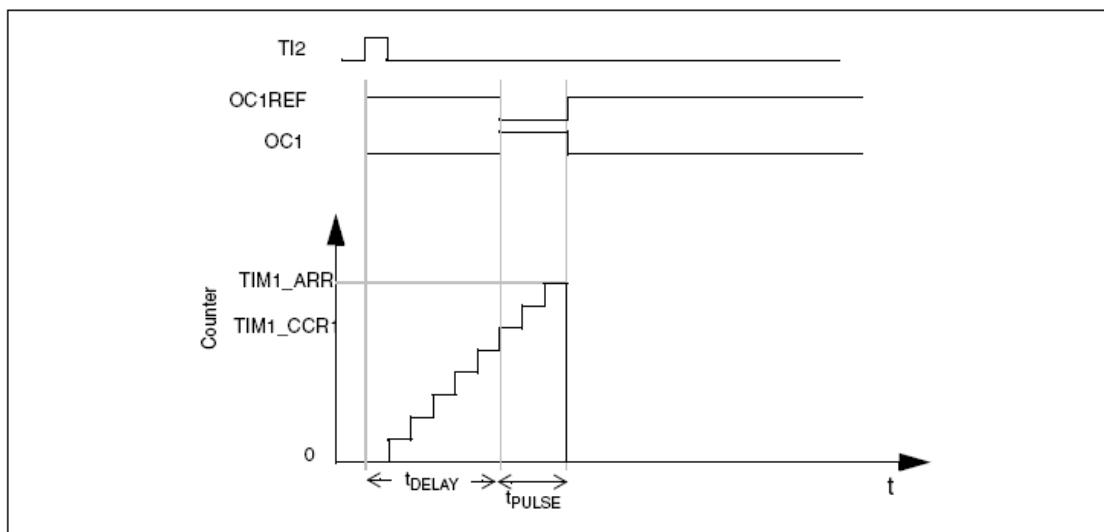
单脉冲模式 (OPM) 是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置  $\text{TIMx_CR1}$  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：

- $\text{CNT} < \text{CCRx} \leq \text{ARR}$  (特别地,  $0 < \text{CCRx}$ )

图 128. 单脉冲模式的例子



例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟  $t_{DELAY}$  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 1：

- 置  $\text{TIMx\_CCMR1}$  寄存器中的  $\text{CC2S} = 01$ ，把  $\text{TI2FP2}$  映像到  $\text{TI2}$ 。
- 置  $\text{TIMx\_CCER}$  寄存器中的  $\text{CC2P} = 0$ ，使  $\text{TI2FP2}$  能够检测上升沿。
- 置  $\text{TIMx\_SMCR}$  寄存器中的  $\text{TS} = 110$ ， $\text{TI2FP2}$  作为从模式控制器的触发 ( $\text{TRGI}$ )。
- 置  $\text{TIMx\_SMCR}$  寄存器中的  $\text{SMS} = 110$  (触发模式)， $\text{TI2FP2}$  被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）。

- $t_{DELAY}$  由写入  $\text{TIMx\_CCR1}$  寄存器中的值定义。
- $t_{PULSE}$  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，当计数器到达预装载值是要产生一个从 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  $\text{ARPE}$ ；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，修改  $\text{UG}$  位来产生一个更新事件，然后等待在  $\text{TI2}$  上的一个外部触发事件。本例中， $\text{CC1P} = 0$ 。

在这个例子中， $\text{TIMx\_CR1}$  寄存器中的  $\text{DIR}$  和  $\text{CMS}$  位应该置低。

因为只需一个脉冲，所以必须设置  $\text{TIMx\_CR1}$  寄存器中的  $\text{OPM} = 1$ ，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

#### 特殊情况：OCx 快速使能：

在单脉冲模式下，在  $\text{TIx}$  输入脚的边沿检测逻辑设置  $\text{CEN}$  位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时  $t_{DELAY}$ 。

如果要以最小延时输出波形，可以设置  $\text{TIMx\_CCMRx}$  寄存器中的  $\text{OCxFE}$  位；此时强制  $\text{OCxREF}$  (和  $\text{OCx}$ ) 被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。 $\text{OCxFE}$  只在通道配置为 PWM1 和 PWM2 模式时起作用。

### 13.3.11 定时器外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

#### 从模式：复位模式

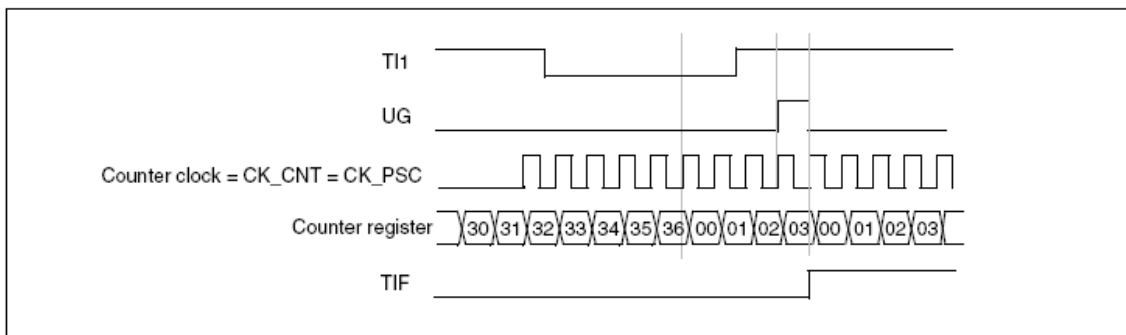
在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx\_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx\_ARR，TIMx\_CCRx）都被更新了。

- 在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：
- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F = 0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx\_CCMR1 寄存器中 CC1S = 01。置 TIMx\_CCER 寄存器中 CC1P = 0 以确定极性（只检测上升沿）。
- 置 TIMx\_SMCR 寄存器中 SMS = 100，配置定时器为复位模式；置 TIMx\_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。
- 置 TIMx\_CR1 寄存器中 CEN = 1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TIMx\_SR 寄存器中的 TIF 位）被设置，根据 TIMx\_DIER 寄存器中 TIE（中断使能）位的设置，产生一个中断请求。

下图显示当自动重装载寄存器 TIMx\_ARR = 0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

图 129. 复位模式下的控制电路



#### 从模式：门控模式

计数器的使能依赖于选中的输入端的电平。

在如下的例子中，计数器只在 TI1 为低时向上计数：

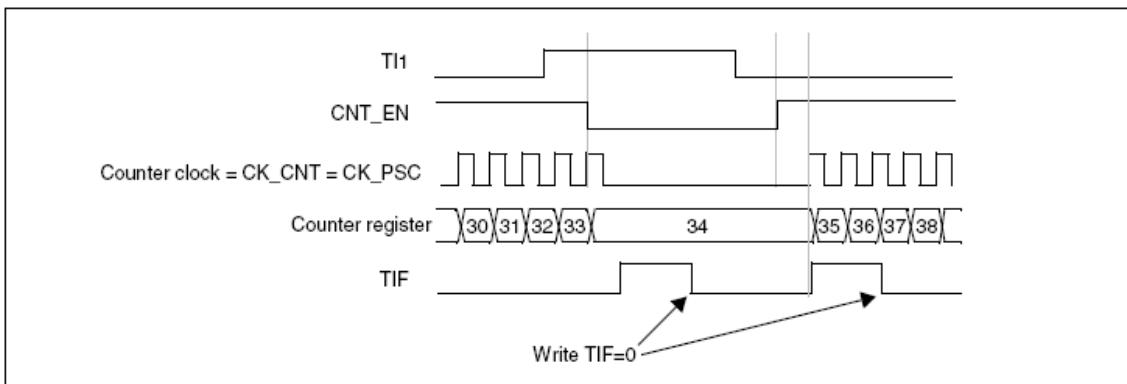
- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F = 0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx\_CCMR1 寄存器中 CC1S = 01。置 TIMx\_CCER 寄存器中 CC1P = 1 以确定极性（只检测低电平）。
- 置 TIMx\_SMCR 寄存器中 SMS = 101，配置定时器为门控模式；置 TIMx\_SMCR 寄存器中 TS = 101，选择 TI1 作为输入源。

- 置  $\text{TIMx\_CR1}$  寄存器中  $\text{CEN} = 1$ , 启动计数器。在门控模式下, 如果  $\text{CEN} = 0$ , 则计数器不能启动, 不论触发输入电平如何。

只要  $\text{TI1}$  为低, 计数器开始依据内部时钟计数, 在  $\text{TI1}$  变高时停止计数。当计数器开始或停止时都设置  $\text{TIMx\_SR}$  中的  $\text{TIF}$  标置。

$\text{TI1}$  上升沿和计数器实际停止之间的延时取决于  $\text{TI1}$  输入端的重同步电路。

图 130. 门控模式下的控制电路



### 从模式：触发模式

计数器的使能依赖于选中的输入端上的事件。

在下面的例子中, 计数器在  $\text{TI2}$  输入的上升沿开始向上计数:

- 配置通道 2 检测  $\text{TI2}$  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  $\text{IC2F} = 0000$ )。触发操作中不使用捕获预分频器, 不需要配置。 $\text{CC2S}$  位只用于选择输入捕获源, 置  $\text{TIMx_CCMR1}$  寄存器中  $\text{CC2S} = 01$ 。置  $\text{TIMx_CCER}$  寄存器中  $\text{CC1P} = 1$  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 131. 触发器模式下的控制电路



### 13.3.12 定时器同步

TIM 定时器从内部链接在一起, 以实现定时器同步或级联。有关详细信息, 请参见-----。

### 13.3.13 调试模式

当微控制器进入调试模式（CPU 核心停止），根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见调试模块章节。

## 13.4 TIMx 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。

### 13.4.1 控制寄存器 1 (TIMx\_CR1)

偏移地址: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留				CKD		ARPE		保留		OPM		URS		UDIS		CEN
								rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 10	保留
位9: 8	<b>CKD[1: 0]:</b> 时钟分频因子（Clock division） 这2位定义在定时器时钟 (CK_INT) 频率、死区时间和由死区发生器与数字滤波器 (ETR, TIx) 所用的采样时钟之间的分频比例。 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: 保留，不要使用这个配置
位7	<b>ARPE:</b> 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
位6: 4	保留
位3	<b>OPM:</b> 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
位2	<b>URS:</b> 更新请求源（Update request source） 软件通过该位选择UEV事件的源 0: 如果允许产生更新中断，则下述任一事件产生一个更新中断： - 计数器溢出/下溢 - 设置UG位 - 从模式控制器产生的更新 1: 如果允许产生更新中断，则只有计数器溢出/下溢才产生一个更新中断

位1	<p><b>UDIS:</b> 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:</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 被缓存的寄存器被装入它们的预装载值。</li> </ul> <p>1: 禁止UEV。不产生更新事件, 影子寄存器 (ARR、PSC、CCRx) 保持它们的值。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>
位0	<p><b>CEN:</b> 允许计数器 (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> <p>在单脉冲模式下, 当发生更新事件时, CEN被自动清除。</p>

### 13.4.2 控制寄存器 2 (TIMx\_CR2)

偏移地址: 0x04

复位值: 0x0000

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

位31: 7	保留, 必须保持复位值。
位6: 4	<p><b>MMS[1: 0]:</b> 主模式选择 (Master mode selection)</p> <p>这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下:</p> <p>000: 复位 – TIMx_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位, 则TRGO上的信号相对实际的复位会有一个延迟。</p> <p>001: 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> <p>010: 更新 – 更新事件被选为触发输入 (TRGO)。例如, 一个主定时器的时钟可以被用作一个从定时器的预分频器。</p> <p>011: 比较脉冲 – 一旦发生一次捕获或一次比较成功时, 当要设置CC1IF标志时 (即使它已经为高), 触发输出送出一个正脉冲 (TRGO)。</p> <p>100: 比较 – OC1REF信号被用于作为触发输出 (TRGO)。</p> <p>101: 比较 – OC2REF信号被用于作为触发输出 (TRGO)。</p> <p>110: 保留。</p> <p>111: 保留。</p>
位3: 0	保留, 必须保持复位值。

### 13.4.3 从模式控制寄存器 (TIMx\_SMCR)

偏移地址: 0x08

复位值: 0x0000

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

位31: 8	保留, 必须保持复位值。
位7	<b>MSM:</b> 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
位6: 4	<b>TS[2: 0]:</b> 触发选择 (Trigger selection) 这3位选择用于同步计数器的触发输入。 000: 内部触发0 (ITR0) 001: 内部触发1 (ITR1) 010: 内部触发2 (ITR2) 011: 内部触发3 (ITR3) 100: TI1的边沿检测器 (TI1F_ED) 101: 滤波后的定时器输入1 (TI1FP1) 110: 滤波后的定时器输入2 (TI2FP2) 111: 保留 更多有关ITRx的细节, 参见下表。 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。
位3	保留, 必须保持复位值。
位2: 0	<b>SMS:</b> 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: 禁止从模式 – 如果CEN = 1, 则预分频器直接由内部时钟驱动。 001: 保留。 010: 保留 011: 保留 100: 复位模式 – 选中的触发输入 (TRGI) 的上升沿重新初始化计数器, 并且产生一个更新寄存器的信号。 101: 门控模式 – 当触发输入 (TRGI) 为高时, 计数器的时钟开启。一旦触发输入变为低, 则计数器停止 (但不复位)。计数器的启动和停止都是受控的。 110: 触发模式 – 计数器在触发输入TRGI的上升沿启动 (但不复位), 只有计数器的启动是受控的。 111: 外部时钟模式1 – 选中的触发输入 (TRGI) 的上升沿驱动计数器。 注: 如果TI1F_EN被选为触发输入 (TS = 100) 时, 不要使用门控模式。这是因为, TI1F_ED在每次TI1F变化时输出一个脉冲, 然而门控模式是要检查触发输入的电平。

表 30. TIMx 内部触发连接

从定时器	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM3	TIM4	TIM1	TIM2	TIM7
TIM4	TIM5	TIM1	TIM2	TIM3
TIM5	TIM3	TIM4	TIM1	TIM6
TIM6	TIM3	TIM4	TIM7	TIM2
TIM7	TIM3	TIM4	TIM1	TIM5

#### 13.4.4 DMA/中断使能寄存器 (TIMx\_DIER)

偏移地址: 0x0C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									TIE				CC2 IE	CC1 IE	UIE

rw                    rw

位31: 7	保留, 必须保持复位值。
位6	<b>TIE:</b> 触发中断使能 (Trigger interrupt enable) 0: 禁止触发中断 1: 使能触发中断
位5: 3	保留, 必须保持复位值。
位2	<b>CC2IE:</b> 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
位1	<b>CC1IE:</b> 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
位0	<b>UIE:</b> 允许更新中断 (Update interrupt enable) 0: 禁止更新中断 1: 允许更新中断

#### 13.4.5 状态寄存器 (TIMx\_SR)

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CC2 OF	CC1 OF	保留	TIF	保留			CC2 IF	CC1 IF	UIF	

rc\_w0 rc\_w0                    rc\_w0                    rc\_w0 rc\_w0 rc\_w0

位31: 11	保留
位10	<b>CC2OF:</b> 捕获/比较2重复捕获标记 (Capture/Compare 2 overcapture flag) 参见CC1OF描述。
位9	<b>CC1OF:</b> 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
位8: 7	保留，必须保持复位值。
位6	<b>TIF:</b> 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，或或门控模式下的任一边沿）时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生 1: 触发器中断等待响应
位5: 3	保留，必须保持复位值。
位2	<b>CC2IF:</b> 捕获/比较2 中断标记 (Capture/Compare 2 interrupt flag) 参考CC1IF描述。
位1	<b>CC1IF:</b> 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 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）
位0	<b>UIF:</b> 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: 更新事件等待响应。当寄存器被更新时该位由硬件置1 -- 上溢且当 TIMx_CR1 寄存器中 UDIS = “0” 时。 -- 当由于 TIMx_CR1 寄存器中 URS = “0” 且 UDIS = “0” 而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。 -- 当由于 TIMx_CR1 寄存器中 URS = “0” 且 UDIS = “0” 而通过触发事件（请参见同步控制寄存器说明）重新初始化 CNT 时。

### 13.4.6 事件产生寄存器 (TIMx\_EGR)

偏移地址: 0x14

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								TG	保留				CC2G	CC1G	UG
									W	W	W	W	W	W	W

位31: 7	保留, 必须保持复位值。
位6	<b>TG:</b> 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。
位5:3	保留, 必须保持复位值。
位2	<b>CC2G:</b> 产生捕获/比较2事件 (Capture/compare 2 generation) 参考CC1G描述。
位1	<b>CC1G:</b> 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。
位0	<b>UG:</b> 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)。

### 13.4.7 捕捉/比较模式寄存器 1 (TIMx\_CCMR1)

偏移地址: 0x18

复位值: 0x0000

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CC<sub>x</sub>S 定义。该寄存器其他位的作用和输出模式下不同。OC<sub>xx</sub> 描述了通道在输出模式下的功能，IC<sub>xx</sub> 描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	OC2M	OC2P E	OC2F E	CC2S	保留	0C1M	OC1P E	OC1F E	CC1S						
IC2F	IC2PSC				IC1F	IC1PSC									
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

输出比较模式:

位15	保留，必须保持复位值。
位14: 12	<b>OC2M[2: 0]:</b> 输出比较2模式 (Output compare 2 mode)
位11	<b>OC2PE:</b> 输出比较2预装载使能 (Output compare 2 preload enable)
位10	<b>OC2FE:</b> 输出比较2快速使能 (Output compare 2 fast enable)
位9: 8	<b>CC2S[1: 0]:</b> 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）。 <b>注：</b> CC2S仅在通道关闭时 (TIMx_CCER寄存器的CC2E = 0) 才是可写的。
位7	保留，必须保持复位值。

位6: 4	<p><b>OC1M[2: 0]:</b> 输出比较1模式 (Output compare 1 enable)</p> <p>该3位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1、OC1N的值。</p> <p>OC1REF 是高电平有效，而OC1、OC1N的有效电平取决于CC1P、CC1NP位。</p> <p>000: 冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用；</p> <p>001: 匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时，强制OC1REF为高。</p> <p>010: 匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1) 相同时，强制OC1REF为低。</p> <p>011: 翻转。当TIMx_CCR1 = TIMx_CNT时，翻转OC1REF的电平。</p> <p>100: 强制为无效电平。强制OC1REF为低。</p> <p>101: 强制为有效电平。强制OC1REF为高。</p> <p>110: PWM模式1 — 在向上计数时，一旦TIMx_CNT &lt; TIMx_CCR1时通道1为有效电平，否则为无效电平；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：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>
位3	<p><b>OC1PE:</b> 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。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>
位2	<p><b>OC1FE:</b> 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。OCFE只在通道被配置成PWM1或PWM2模式时起作用。</p>
位1: 0	<p><b>CC1S[1: 0]:</b> 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>

## 输入捕获模式:

位15: 12	<b>IC2F[3: 0]:</b> 输入捕获2滤波器 (Input capture 2 filter)
位11: 10	<b>IC2PSC[1: 0]:</b> 输入/捕获2预分频器 (input capture 2 prescaler)
位9: 8	<p><b>CC2S[1: 0]:</b> 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>
位7: 4	<p><b>IC1F[3: 0]:</b> 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<math>f_{DTS}</math>采样            1000: 采样频率<math>f_{SAMPLING} = f_{DTS}/8, N = 6</math>            0001: 采样频率<math>f_{SAMPLING} = f_{CK\_INT}, N = 2</math>            1001: 采样频率<math>f_{SAMPLING} = f_{DTS}/8, N = 8</math>            0010: 采样频率<math>f_{SAMPLING} = f_{CK\_INT}, N = 4</math>            1010: 采样频率<math>f_{SAMPLING} = f_{DTS}/16, N = 5</math>            0011: 采样频率<math>f_{SAMPLING} = f_{CK\_INT}, N = 8</math>            1011: 采样频率<math>f_{SAMPLING} = f_{DTS}/16, N = 6</math>            0100: 采样频率<math>f_{SAMPLING} = f_{DTS}/2, N = 6</math>            1100: 采样频率<math>f_{SAMPLING} = f_{DTS}/16, N = 8</math>            0101: 采样频率<math>f_{SAMPLING} = f_{DTS}/2, N = 8</math>            1101: 采样频率<math>f_{SAMPLING} = f_{DTS}/32, N = 5</math>            0110: 采样频率<math>f_{SAMPLING} = f_{DTS}/4, N = 6</math>            1110: 采样频率<math>f_{SAMPLING} = f_{DTS}/32, N = 6</math>            0111: 采样频率<math>f_{SAMPLING} = f_{DTS}/4, N = 8</math>            1111: 采样频率<math>f_{SAMPLING} = f_{DTS}/32, N = 8</math></p>
位3: 2	<p><b>IC1PSC[1: 0]:</b> 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>

位1: 0	<p><b>CC1S[1: 0]:</b> 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:</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>
-------	---

### 13.4.8 捕捉/比较使能寄存器 (TIMx\_CCER)

偏移地址: 0x20

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留							CC2P	CC1E	保留			CC1P	CC1E		
w	w			w	w			w	w			w	w		

位15: 6	保留, 始终读为0。
位5	<p><b>CC2P:</b> 输入/捕获2输出极性 (Capture/Compare 2 output polarity) 参考CC1P的描述。</p>
位4	<p><b>CC2E:</b> 输入/捕获2输出使能 (Capture/Compare 2 output enable) 参考CC1E的描述。</p>
位3: 2	保留, 始终读为0。
位1	<p><b>CC1P:</b> 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LCCK位) 设为3或2, 则该位不能被修改。</p>

位0	<b>CC1E:</b> 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 31. 标准 Ocx 通道的输出控制位

CCxE位	OCx输出状态
0	禁止输出 (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + 极性, OCx_EN = 1

注: 管脚连接到标准的 Ocx 通道的外部 I/O 管脚的状态, 取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

#### 13.4.9 计数器 (TIMx\_CNT)

偏移地址: 0x24

复位值: 0x0000

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

位31: 0	CNT[31: 0]: 计数器的值 (Counter value)
--------	-----------------------------------

#### 13.4.10 预分频器 (TIMx\_PSC)

偏移地址: 0x28

复位值: 0x0000

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

位15: 0	<b>PSC[15: 0]:</b> 预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 PSC包含了每次当更新事件产生时，装入当前预分频器寄存器的值。更新事件包括计数器被TIM_EGR的UG位清'0'或被工作在复位模式的从控制器清'0'。
--------	---

#### 13.4.11 自动装载寄存器 (TIMx\_ARR)

偏移地址: 0x2C

复位值: 0x0000

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

位31: 0	<b>ARR[31: 0]:</b> 自动重装载的值 (Auto reload value) ARR包含了将要装载入实际的自动重装载寄存器的数值。 当自动重装载的值为空时，计数器不工作。
--------	---

#### 13.4.12 捕获/比较寄存器 1 (TIMx\_CCR1)

偏移地址: 0x34

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31: 16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0	<b>CCR1[31: 0]:</b> 捕获/比较1的值 (Capture/Compare 1 value) 若CC1通道配置为输出： CCR1包含了装入当前捕获/比较1寄存器的值（预装载值）。 如果在TIMx_CCMR1寄存器 (OC1PE位) 中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较1寄存器中。当前捕获/比较寄存器参与同计数器TIMx_CNT的比较，并在OC1端口上产生输出信号。 若CC1通道配置为输入： CCR1包含了由上一次输入捕获1事件 (IC1) 传输的计数器值。
--------	--

### 13.4.13 捕获/比较寄存器 2 (TIMx\_CCR2)

偏移地址: 0x38

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31: 16]															
rw	rw	rw	rw	rw	rw	rw	Rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
位31: 0		<b>CCR2[31: 0]:</b> 捕获/比较2的值 (Capture/Compare 2 value) 若CC2通道配置为输出: CCR2包含了装入当前捕获/比较2寄存器的值（预装载值）。 如果在TIMx_CCMR2寄存器 (OC2PE位) 中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较2寄存器中。当前捕获/比较寄存器参与计数器TIMx_CNT的比较，并在OC2端口上产生输出信号。 若CC2通道配置为输入: CCR2包含了由上一次输入捕获2事件 (IC2) 传输的计数器值。													

## 14. 基础定时器 (TIM8/9/10)

### 14.1 TIM8/9/10 简介

通用定时器是一个通过可编程预分频器驱动的 32 位自动装载计数器构成。

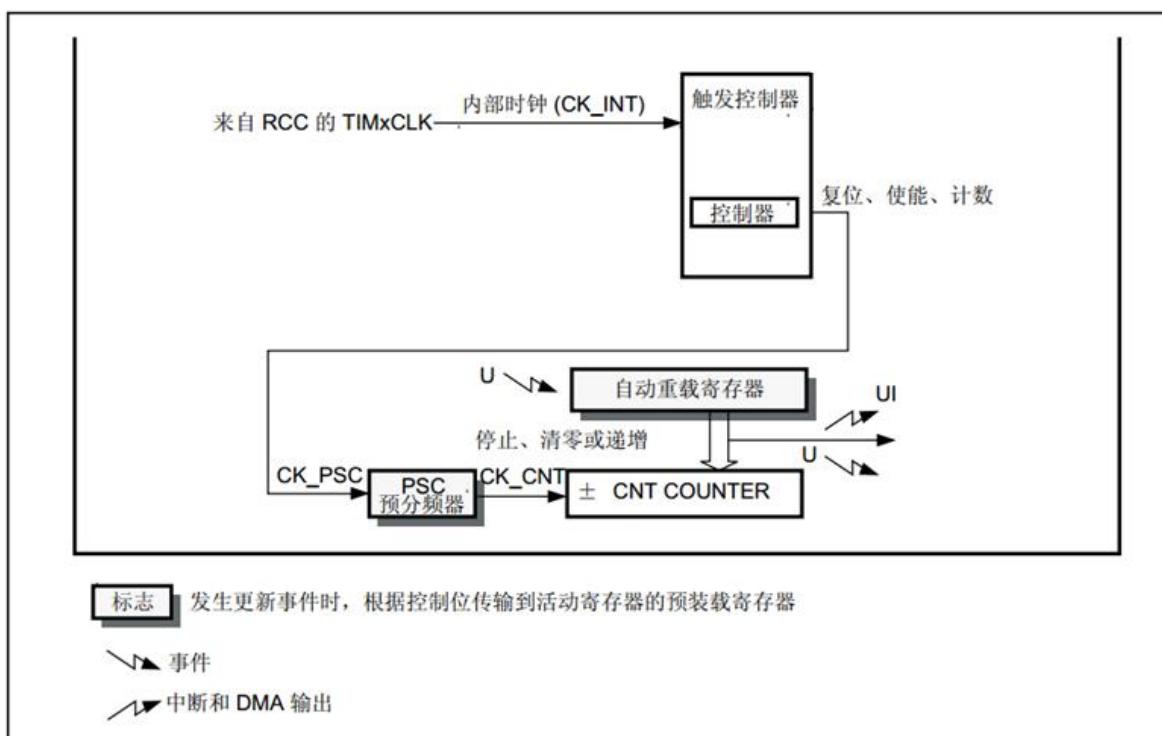
TIM8/9/10 定时器是完全独立的，而且没有互相共享任何资源。它们可以一起同步操作。

### 14.2 TIM8/9/10 主要功能

通用 TIM8/9/10 定时器功能包括：

- 32 位自动装载递增计数器。
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1 ~ 65536 之间的任意数值
- 如下事件发生时产生中断/DMA：计数器向上溢出

图 132. 基本定时器框图



### 14.3 TIM8/9/10 功能描述

#### 14.3.1 时基单元

可编程通用定时器的主要部分是一个 32 位递增计数器和与其相关的自动装载寄存器。计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写，时基单元包含：

- 计数器寄存器 (TIMx\_CNT)
- 预分频器寄存器 (TIMx\_PSC)

- 自动装载寄存器 (TIMx\_ARR)

自动装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TIMX\_CR1 寄存器中的自动装载预装载使能位 (ARPE) 的设置，预装载寄存器的内容被立即或在 每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件 (向下计数时的下溢条件) 并且 TIMX\_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK\_CNT 驱动，仅当设置了计数器 TIMX\_CR1 寄存器中的计数器使能位 (CEN) 时，CK\_CNT 才有效。

### 预分频器描述

预分频器可以将计数器的时钟频率按 1 ~ 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 32 位计数器。因为这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下面两个图分别给出了在预分频器运行时，更改计数器参数的例子。

图 133. 当预分频器的参数从 1 变到 2 时，计数器的时序图

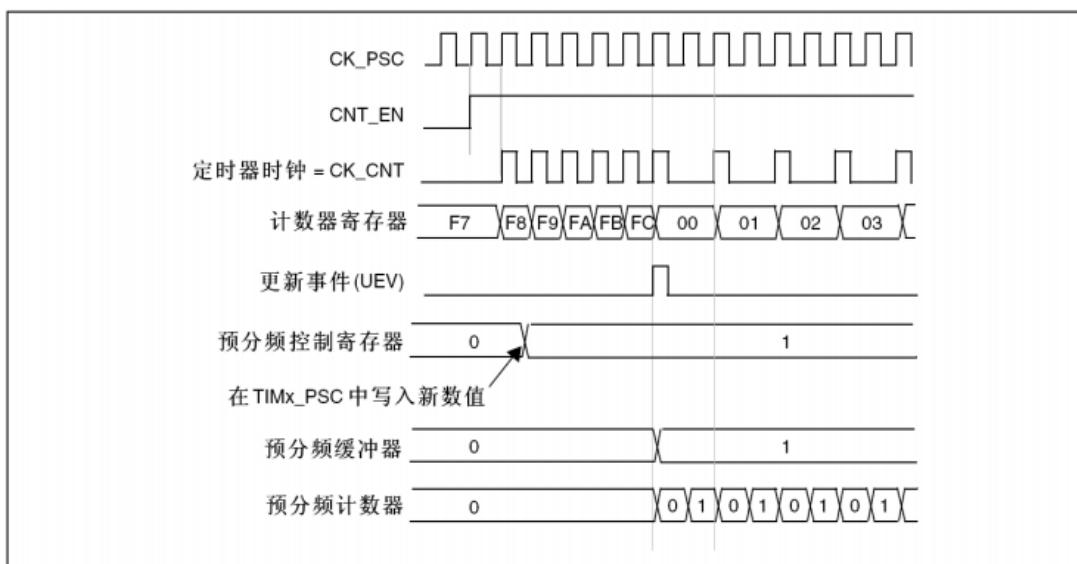
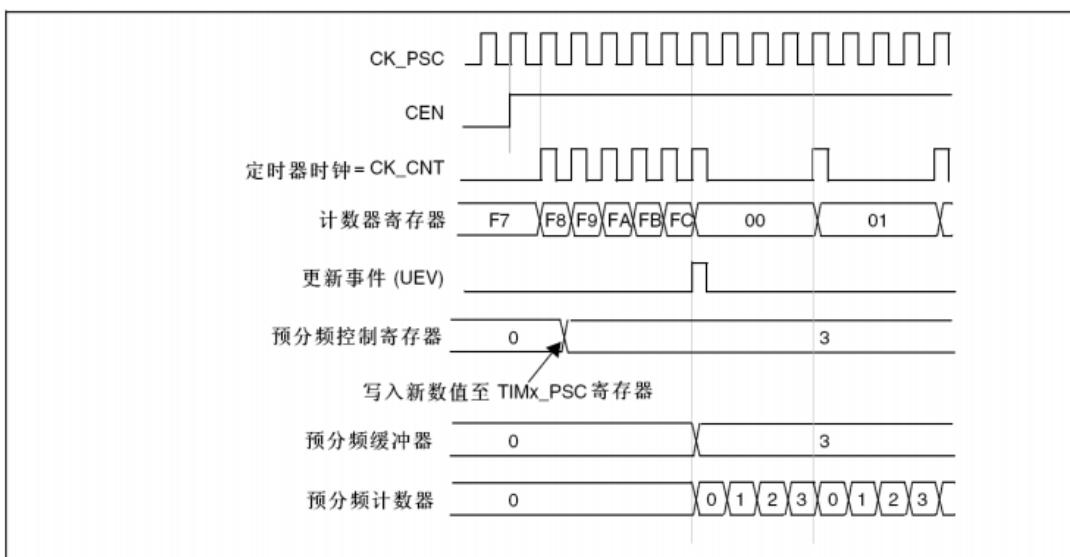


图 134. 当预分频器的参数从 1 变到 4 时，计数器的时序图



### 14.3.2 计数模式

计数器从 0 计数到自动加载值 (TIMx\_ARR 寄存器的内容)，然后重新从 0 开始计数并且产生一个计数器向上溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx\_EGR 寄存器中设置 UG 位（通过软件方式或者使用从模式控制器）也同样可以产生一个更新事件。

设置 TIMx\_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清 0 之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清 0，同时预分频器的计数也被清 0（但预分频器的数值不变）。此外，如果设置了 TIMx\_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

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

- 预分频器的缓冲区被置入预装载寄存器的值 (TIMx\_PSC 寄存器的内容)
- 自动装载影子寄存器被重新置入预装载寄存器的值 (TIMx\_ARR)

下图给出一些例子，当 TIMx\_ARR = 0x36 时计数器在不同时钟频率下的动作：

图 135. 计数器时序图，内部时钟分频因子为 1

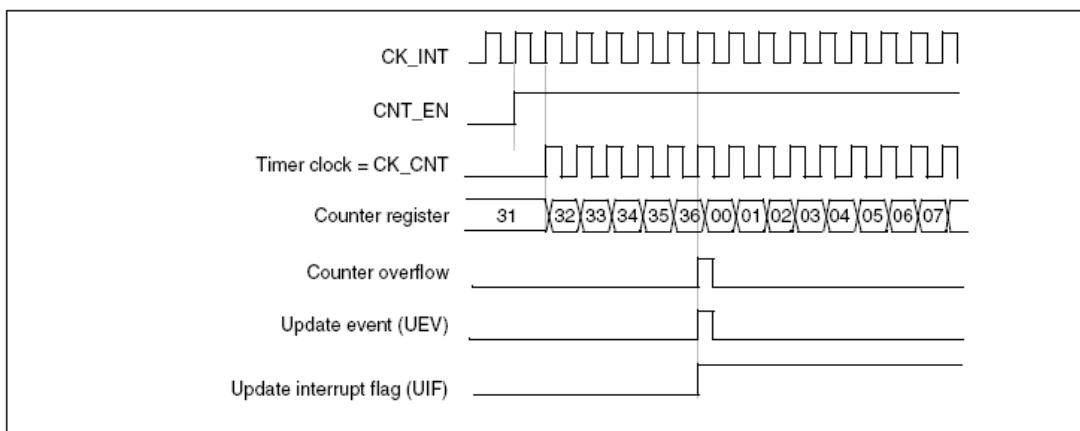


图 136. 计数器时序图，内部时钟分频因子为 2

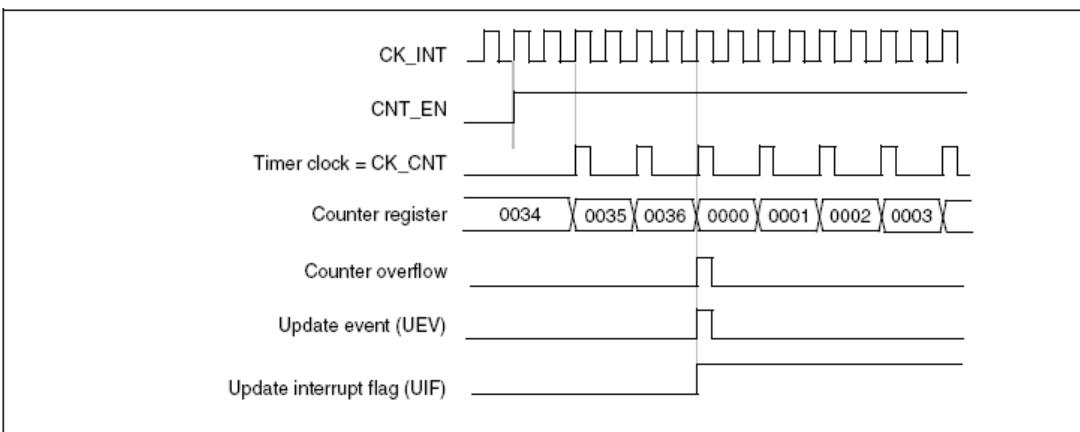


图 137. 计数器时序图，内部时钟分频因子为 4

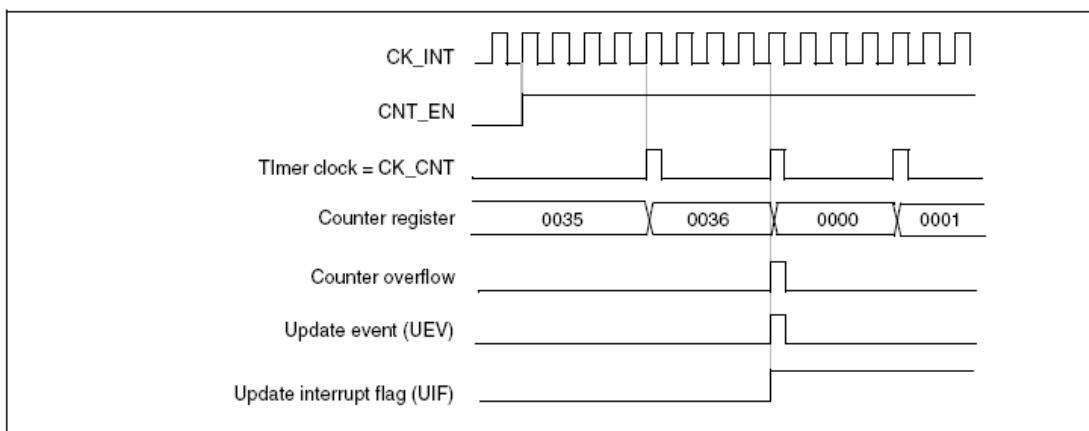


图 138. 计数器时序图，内部时钟分频因子为 N

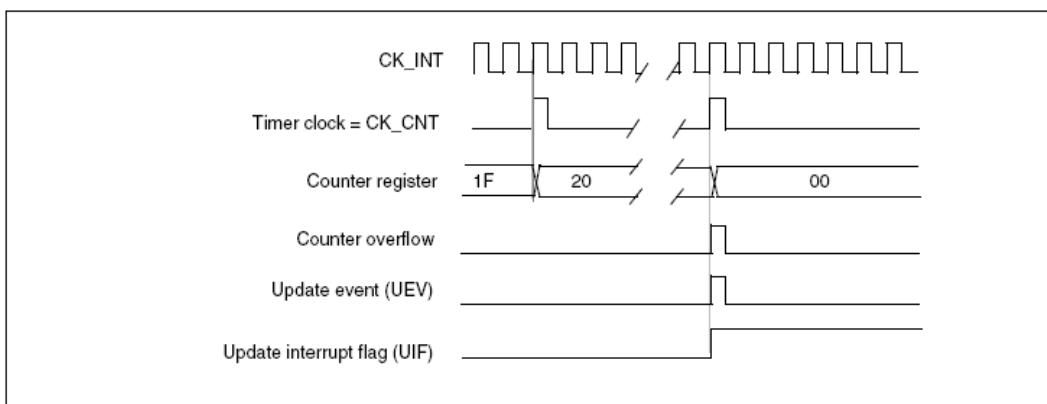


图 139. 计数器时序图，当 ARPE = 0 时的更新事件 (TIMx\_ARR 没有预装入)

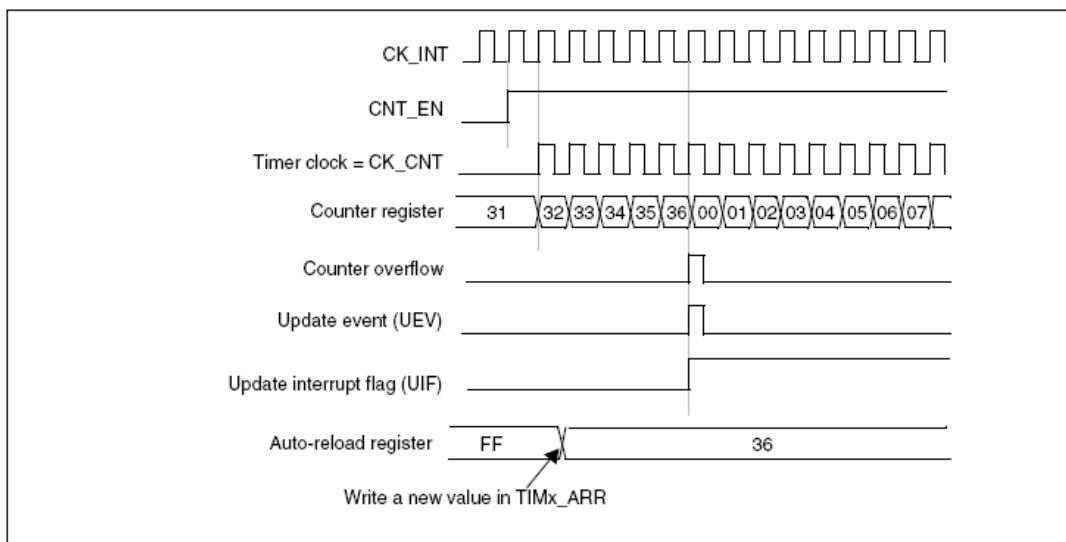
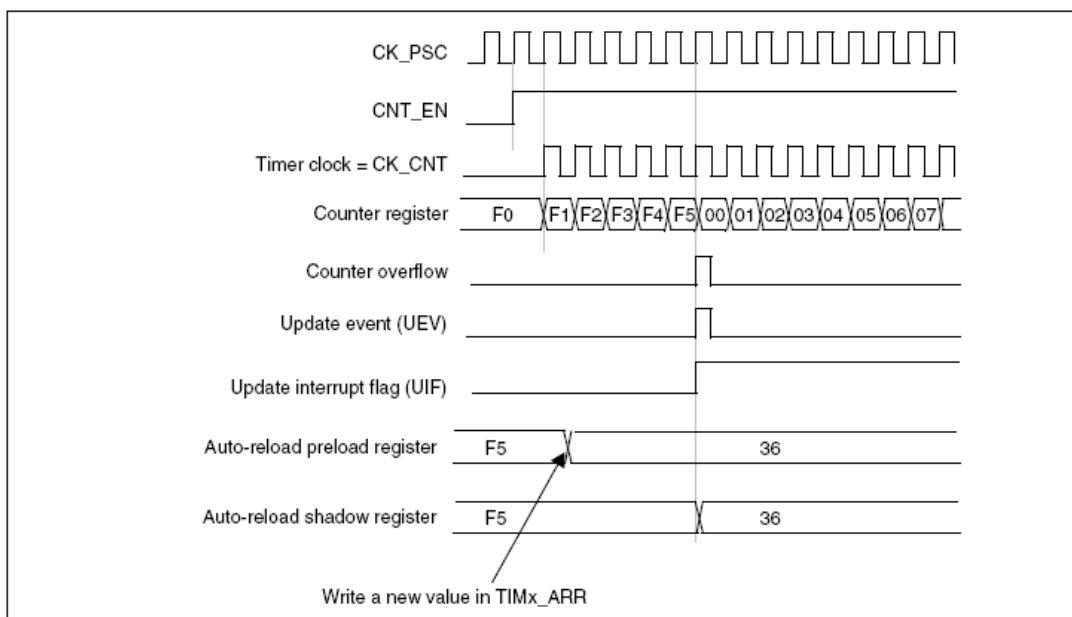


图 140. 计数器时序图，当 ARPE = 1 时的更新事件 (预装入了 TIMx\_ARR)



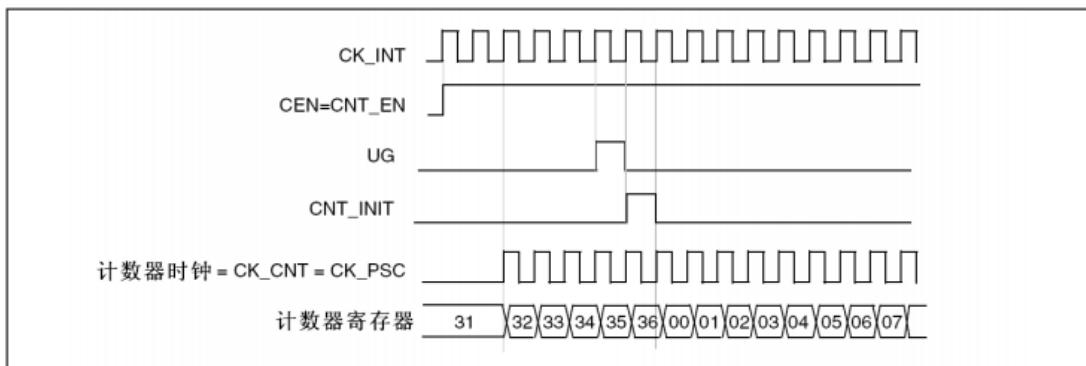
### 14.3.3 时钟源

计数器时钟可由内部时钟（CK\_INT）时钟源提供。

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）。一旦 CEN 位被写成 1，预分频器的时钟就由内部时钟 CK\_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

图 141. 一般模式下的控制电路，内部时钟分频因子为 1



### 14.3.4 调试模式

当微控制器进入调试模式（CPU 核心停止），根据 DBG 模块中 DBG\_TIMx\_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见调试模块章节。

## 14.4 TIM8/9/10 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。

### 14.4.1 控制寄存器 1 (TIMx\_CR1)

偏移地址: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留		ARPE		保留		OPM		URS		UDIS		CEN			

位31: 8	保留, 必须保持复位值。
位7	<b>ARPE:</b> 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
位6: 4	保留, 必须保持复位值。
位3	<b>OPM:</b> 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
位2	<b>URS:</b> 更新请求源 (Update request source) 软件通过该位选择UEV事件的源 0: 如果允许产生更新中断或DMA请求, 则下述任一事件产生一个更新中断或DMA请求: <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> 1: 如果允许产生更新中断或DMA请求, 则只有计数器上溢才产生一个更新中断或DMA请求
位1	<b>UDIS:</b> 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: <ul style="list-style-type: none"> <li>- 计数器上溢</li> <li>- 设置UG位</li> <li>- 从模式控制器产生的更新 被缓存的寄存器被装入它们的预装载值。</li> </ul> 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。
位0	<b>CEN:</b> 允许计数器 (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。 在单脉冲模式下, 当发生更新事件时, CEN被自动清除。

#### 14.4.2 控制寄存器 2 (TIMx\_CR2)

偏移地址: 0x04

复位值: 0x0000

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

位31: 7	保留, 必须保持复位值。
位6: 4	<b>MMS[1: 0]:</b> 主模式选择 (Master mode selection) 这两位用于选择在主模式下送到从定时器的同步信息 (TRGO)。可能的组合如下: 000: 复位 – TIMx_EGR寄存器的UG位被用于作为触发输出 (TRGO)。如果触发输入 (从模式控制器处于复位模式) 产生复位, 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。
位3: 0	保留, 必须保持复位值。

#### 14.4.3 DMA/中断使能寄存器 (TIMX\_DIER)

偏移地址: 0x0C

复位值: 0x0000

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

位31: 9	保留, 必须保持复位值。
位8	<b>位 8 UDE:</b> 更新 DMA 请求使能 (Update DMA request enable) 0: 禁止更新 DMA 请求。 1: 使能更新 DMA 请求。
位7: 1	保留, 必须保持复位值。
位0	<b>UIE:</b> 允许更新中断 (Update interrupt enable) 0: 禁止更新中断 1: 允许更新中断

#### 14.4.4 状态寄存器 (TIMx\_SR)

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														UIF	
rc_w0															

位31: 1	保留, 必须保持复位值。
位0	<p><b>UIF:</b> 更新中断标志 (Update interrupt flag) 该位在发生更新事件时通过硬件置 1。但需要通过软件清零。 0: 未发生更新。 1: 更新中断挂起。该位在以下情况下更新寄存器时由硬件置 1: — 上溢或下溢并且当 TIMx_CR1 寄存器中 UDIS = 0 时。 — 当由于 TIMx_CR1 寄存器中 URS = 0 且 UDIS = 0 而通过软件使用 TIMx_EGR 寄存器中的 UG 位重新初始化 CNT 时。</p>

#### 14.4.5 事件产生寄存器 (TIMx\_EGR)

偏移地址: 0x14

复位值: 0x0000

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

位31: 7	保留, 必须保持复位值。
位0	<p><b>UG:</b> 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>

#### 14.4.6 计数器 (TIMx\_CNT)

偏移地址: 0x24

复位值: 0x0000

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

位31: 0	CNT[31: 0]: 计数器的值 (Counter value)
--------	-----------------------------------

#### 14.4.7 预分频器 (TIMx\_PSC)

偏移地址: 0x28

复位值: 0x0000

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

位15: 0	PSC[15: 0]: 预分频器的值 (Prescaler value) 计数器的时钟频率 (CK_CNT) 等于 $f_{CK\_PSC} / (PSC[15: 0] + 1)$ 。 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'。
--------	--

#### 14.4.8 自动装载寄存器 (TIMx\_ARR)

偏移地址: 0x2C

复位值: 0x0000

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

位31: 0	<p><b>ARR[31: 0]:</b> 自动重装载的值 (Auto reload value) ARR包含了将要装载入实际的自动重装载寄存器的数值。 详细参考13.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。</p>
--------	---

## 15. 独立看门狗 (IWDG)

### 15.1 IWDG 简介

内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备（独立看门狗和窗口看门狗）可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断（仅适用于窗口型看门狗）或产生系统复位。

独立看门狗 (IWDG) 由专门的低速时钟 (LSI) 驱动，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

IWDG 最适合应用于那些需要看门狗作为一个正在主程序外，能够完全独立工作，并且对时间精度要求低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

### 15.2 IWDG 主要性能

- 自由运行的递减计数器
- 时钟由独立的 RC 振荡器提供（可在停止模式下工作）
- 看门狗被激活后，则在计数器计数至 0x0000 时产生复位。

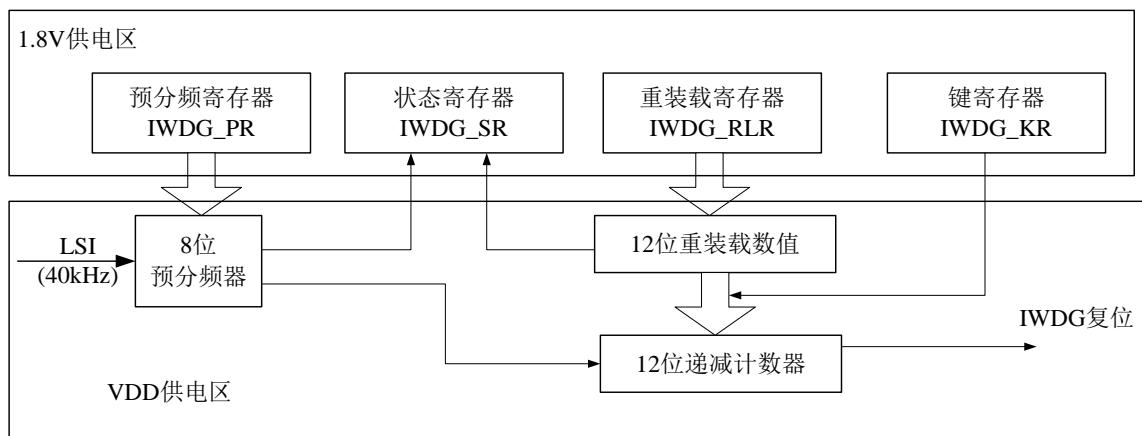
### 15.3 IWDG 功能描述

下图为独立看门狗模块的功能框图。

在键寄存器 (IWDG\_KR) 中写入 0xCCCC。开始启动独立看门狗；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 0x0000 时，会产生一个复位信号 (IWDG\_RESET)。

无论何时，只要在键寄存器 IWDG\_KR 中写入 0xAAAA，IWDG\_RLR 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

图 142. 独立看门狗框图



注：看门狗功能处于 VDD 供电区，即在停机模式时仍能正常工作。

表 32. 看门狗超时时间 (40kHz 的输入时钟 (LSI) )

预分频系数	PR[2: 0]位	最短时间 RL[11: 0]=0x000	最长时间 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 频率会再 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。

### 15.3.1 硬件看门狗

如果用户在选择字节中启动了“硬件看门狗”功能, 在系统上电复位后, 看门狗会自动开始运行; 如果在计数器计数结束前, 若软件没有向键寄存器写入相应的值, 则系统会产生复位。

### 15.3.2 寄存器访问保护

IWDG\_PR 和 IWDG\_RLR 寄存器具有写保护功能。要修改这两个寄存器的值, 必须先向 IWDG\_KR 寄存器中写入 0x5555。以不同的值的写入这个寄存器将会打乱操作顺序, 寄存器将重新被保护。重装载操作(即写入 0xAAAA)也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

### 15.3.3 调试模式

当微控制器进入调试模式时(CPU 核心停止), 根据调试模块中的 DBG\_IWDG\_STOP 配置位的状态, IWDG 的计数器能够继续工作或停止。详见调试模块的章节。

## 15.4 IWDG 寄存器描述

### 15.4.1 键寄存器 (IWDG\_KR)

偏移地址: 0x00

复位值: 0x0000 0000

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

位31: 15	保留, 始终读为0.
位15: 0	<p>KEY[15: 0]: 键值(只写寄存器, 读出值为0x0000) (Key value)            软件必须以一定的间隔写入0xAAAA, 否则, 当计数器为0时, 看门狗会产生复位。            写入0x5555表示允许访问IWDG_PR和IWDG_RLR寄存器。            写入0xCCCC, 启动看门狗工作</p>

### 15.4.2 预分频寄存器 (IWDG\_PR)

偏移地址: 0x04

复位值: 0x0000 0000

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

rw rw rw

位31: 3	保留, 始终读为0.								
位2: 0	<p>PR[2: 0]: 预分频因子 (Prescaler divider)            这些位具有写保护设置。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子, IWDG_SR寄存器的PVU位必须为0。</p> <table> <tbody> <tr> <td>000: 预分频因子=4</td> <td>100: 预分频因子=64</td> </tr> <tr> <td>001: 预分频因子=8</td> <td>100: 预分频因子=128</td> </tr> <tr> <td>010: 预分频因子=16</td> <td>100: 预分频因子=256</td> </tr> <tr> <td>011: 预分频因子=32</td> <td>100: 预分频因子=256</td> </tr> </tbody> </table> <p>注意: 对此寄存器进行读操作, 将从VDD电压域返回预分频值。如果写操作正在进行, 则读回的值可能是无效的。因此, 只有对那个IWDG_SR寄存器的PUV位为0时, 读出的值才有效</p>	000: 预分频因子=4	100: 预分频因子=64	001: 预分频因子=8	100: 预分频因子=128	010: 预分频因子=16	100: 预分频因子=256	011: 预分频因子=32	100: 预分频因子=256
000: 预分频因子=4	100: 预分频因子=64								
001: 预分频因子=8	100: 预分频因子=128								
010: 预分频因子=16	100: 预分频因子=256								
011: 预分频因子=32	100: 预分频因子=256								

### 15.4.3 重装载寄存器 (IWDG\_RLR)

偏移地址: 0x08

复位值: 0x0000 0FFF

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

位31: 12	保留, 始终读为0.
位11: 0	<p>RL[11: 0]: 看门狗计数器重装载值 (Watchdog counter reload value)          这些位具有写保护。用于定义看门狗计数器的重装载值, 每当向IWDG_KR寄存器写入0xAAAA时, 重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。          看门狗超时周期可通过次重装载值和时钟预分频值来计算。          只有当IWDG_SR寄存器中的RVU位为0时, 才能对此寄存器进行修改。          注: 对此寄存器进行读操作, 将从VDD电压域返回预分频值。如果写操作正在进行, 则读回的值可能是无效的。因此, 只有当IWDG_SR寄存器的RUV位为0时, 读出的值才有效。</p>

### 15.4.4 状态寄存器(IWDG\_SR)

地址偏移: 0x0C

复位值: 0x0000 0000

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

位31: 2	保留, 始终读为0.
位1	<p>RVU: 看门狗计数器重装载值更新 (Watchdog counter reload value update)          此位由硬件置“1”用来指示重装载值的更新正在进行中。当在VDD域中的重装载更新结束后, 此位由硬件清“0” (最多需要5个40kHz的RC周期) 重装载值只有在RVU位被清“0”后才可更新。</p>
位0	<p>PVU: 看门狗预分频更新 (Watchdog prescaler value update)          此位由硬件置“1”用来指示预分频值的更新正在进行中。当在VDD域中的预分频值更新结束后, 此位由硬件清“0” (最多需要5个40kHz的RC周期) 预分频值只有在RVU位被清“0”后才可更新。</p>

**注：**如果在应用程序中使用多个重装载值或预分频值，则必须在 RVU 位被清除后才能重新改变预装载值，在 PVU 位被清除后才能重新改变预分频值。然而，在预分频和/或重装值更新后，不必等待 RVU 或 PVU 复位，可以继续执行下面的代码。（即使在低功耗模式下，次写操作仍会被继续执行完成）

## 16. 窗口看门狗 (WWDG)

### 16.1 WWDG 简介

窗口看门狗通常被用来监测由外部干扰或不可预见的逻辑条件造成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。

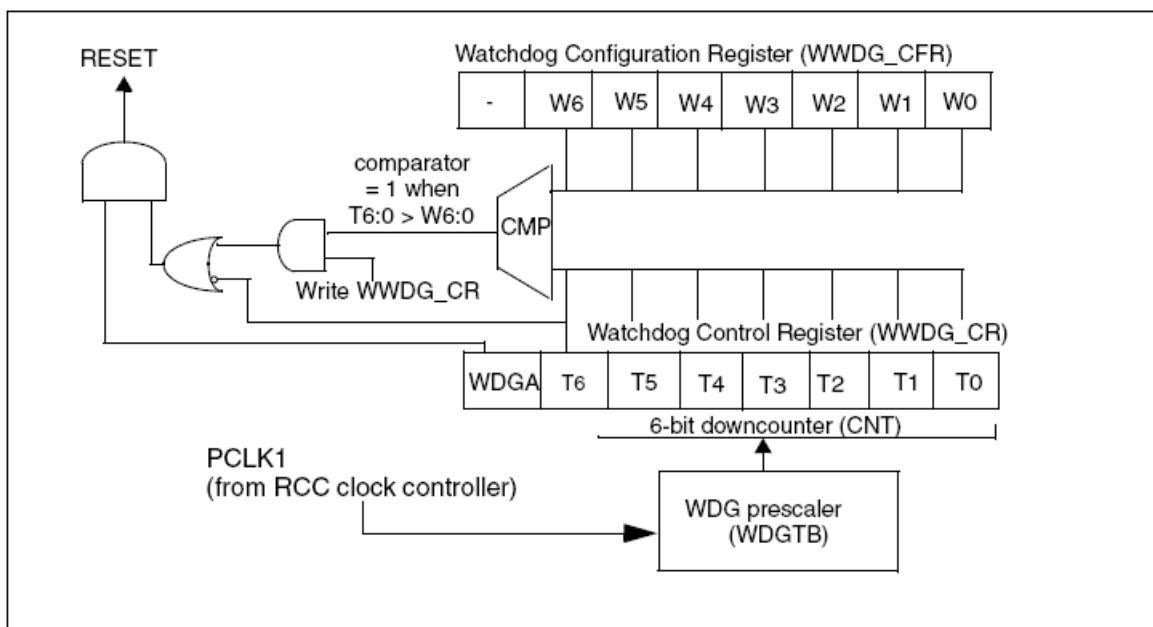
### 16.2 WWDG 主要特征

- 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中断，当递减计数器等于 0x40 时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免 WWDG 复位。

### 16.3 WWDG 功能描述

如果看门狗被启动(WWDG\_CR 寄存器中的 WDGA 位被置'1')，并且当 7 位(T[6: 0])递减计数器从 0x40 翻转到 0x3F(T6 位清零)时，则产生一个复位。如果软件在计数器值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。

图 143. 看门狗框图



应用程序在正常运行过程中必须定期地写入 WWDG\_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于窗口寄存器的值时，才能进行写操作。储存在 WWDG\_CR 寄存器中的数值必须在 0xFF 和 0xC0 之间：

- 启动看门狗

在系统复位后，看门狗总是处于关闭状态，设置 WWDG\_CR 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

- 控制递减计数器

递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。

T[5: 0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG\_CFR 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，下图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。

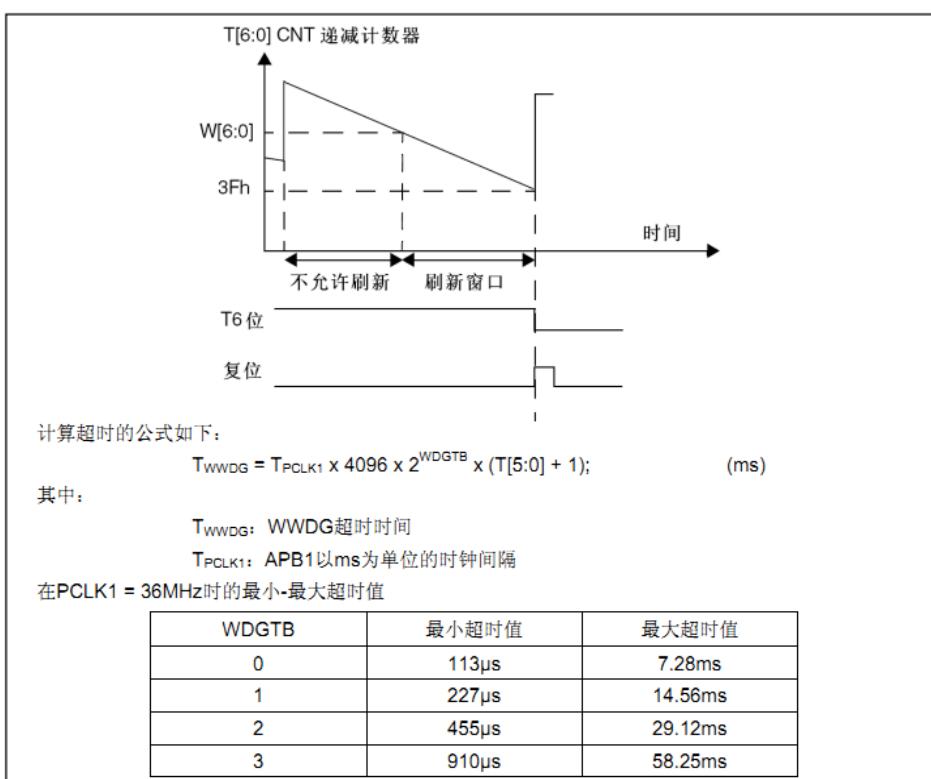
注：可以用 T6 位可以被用来产生一个软件复位 (WDGA 位被置位，T6 位清零)

#### 16.4 如何编写看门狗超时程序

下图显示了装载到看门狗计数器(CNT)中的 6 位计数值和看门狗的延迟时间之间的线性关系(以 ms 为单位)。此图可用来做为快速计算的参考，而未将时间的偏差考虑在内。如果需要更高的精度，可以使用下图提供的计算公式。

**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 144. 窗口看门狗时序图



## 16.5 调试模式

当微控制器进入调试模式时(CPU核心停止),根据调试模块中的DBG\_WWDG\_STOP配置位的状态, WWDG的计数器能够继续工作或停止。详见有关调试模块的章节。

## 16.6 WWDG 寄存器描述

### 16.6.1 控制寄存器 (WWDG\_CR)

偏移地址: 0x00

复位值: 0x7F

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

位31: 8	保留, 始终读为0.
位7	<p>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: 启动看门狗</p>
位6: 0	<p>T[6: 0]: 7位计数器(MSB至LSB) (7-bit counter) 这些位用来存储看门狗的计数器值。每(4096x2WDGTB)个PCLK1周期减1。当计数器值从40h变为3Fh时(T6变成0), 产生看门狗复位。</p>

### 16.6.2 配置寄存器 (WWDG\_CFR)

偏移地址: 0x04

复位值: 0x7F

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

位31: 8	保留, 始终读为0.
位9	<p>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。</p>

位8: 7	WDGTB[1: 0]: 时基 (Timer base) 预分频器的时基可根据如下修改: 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
位6: 0	W[6: 0]: 7位窗口值 (7-bit window value) 这些位包含了用来与递减计数器进行比较用的窗口值。

### 16.6.3 状态寄存器 (WWDG\_SR)

偏移地址: 0x08

复位值: 0x00

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

rc w0

位31: 1	保留, 始终读为0。
位0	EWIF: 提前唤醒中断标志 (Early wakeup interrupt flag) 当计数器值达到40h时, 此位由硬件置1。它必须通过软件写'0'来清除。对此位写'1'无效。 若中断未被使能, 此位也会被置'1'。

## 17. 实时时钟 (RTC)

### 17.1 RTC 简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统（RCC\_BDCR 寄存器）处于后备区域，即在系统复位后，RTC 的设置和时间维持不变。

系统复位后，对后备寄存器和 RTC 的访问被禁止，这是为了防止对后备区（BKP）的意外写操作。执行以下操作将使能对后备寄存器和 RTC 的访问。

- 设置寄存器 RCC\_APB1ENR 的 PWREN 和 BKREN 位，使能电源和后备接口时钟
- 设置寄存器 PWR\_CR 的 DBP 位，使能对后备寄存器和 RTC 的访问。

### 17.2 主要特征

- 可编程的预分频系数：分频系数最高为 220
- 32 位的可编程计数器，用于较长时间段的测量
- 2 个分离的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟（RTC 时钟的频率必须小于 PCLK1 时钟频率的四分之一以上）。
- 可以选择一下三种 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）只能由后备域复位。
- 3 个专门的屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断
  - 秒中断，用来产生一个可编程的周期性中断信号（最长可达 1 秒）
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态

### 17.3 功能描述

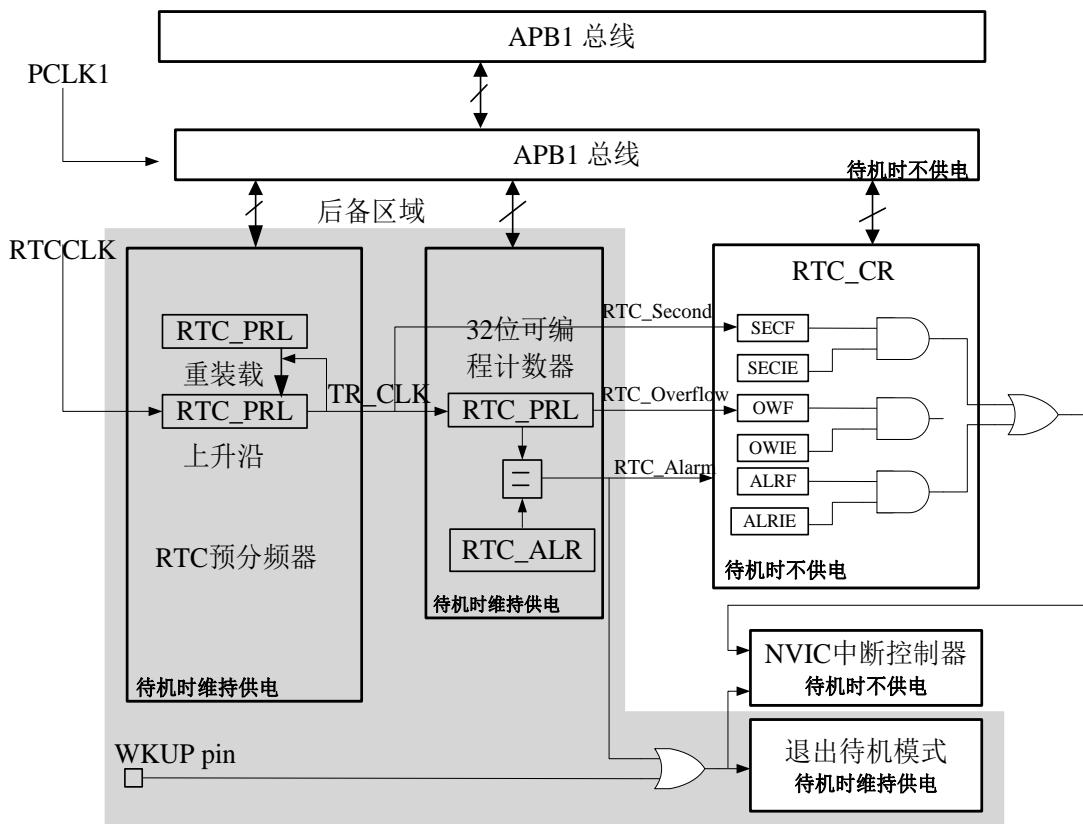
#### 17.3.1 概述

RTC 由两个主要部分组成。参见上图。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，可以通过 APB 总线对其进行读写操作。APB1 接口由 APB1 总线时钟驱动，用来与 APB1 总线接口。

另一部分（RTC 核心）由一组可编程计数器组成，分成两个主要模块。第一个模块是 RTC 的预分频模块，它可编程产生最长为 1 秒的 RTC 时间基准 TR\_CLK.RTC 的预分频模块包含了一个 20 位的可编程预分频器（RTC 预分频器）。如果在 RTC\_CR 寄存器中设置了相应的允许位，则在每个 TR\_CLK 周期中 RTC 产生一个中断（秒中断）。第二个模块是一个 32 位的可编程计数器，可被初始化为当前的系统时间。系统时间接 TR\_CLK 周期累加与存储在 RTC\_ALR 寄存器中的可编程时间比较，如果 RTC\_CR 控制寄存器中设置了相应的允许位，比较匹配时将产生一个闹钟中断。

下图简化的 RTC 框图

图 145. 实时时钟方框图



### 17.3.2 复位过程

除了 RTC\_PRL、RTC\_ALR、RTC\_CNT 和 RTC\_DIV 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

RTC\_PRL、RTC\_ALR、RTC\_CNT 和 RTC\_DIV 寄存器仅能通过备份域复位信号复位。

### 17.3.3 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。

软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在与 RTC APB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 之后，则在第一次内部寄存器更新之前，从 APB1 上读出 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
- 系统刚从停机模式唤醒

所有以上情况中，APB1 接口被禁止时（复位、无时钟或断点）RTC 核仍保持运行状态。

因此，若在读取 RTC 寄存器时，RTC 的 APB1 接口曾经处于禁止状态，则软件首先必须等待 RTC\_CRL 寄存器中的 RSF 位（寄存器同步标志）被硬件置“1”。

**注：**RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。

### 17.3.4 配置 RTC 寄存器

必须设置 RTC\_CRL 寄存器中的 CNF 位，使 RTC 进入配置模式后，才能写入 RTC\_PRL、RTC\_CNT、RTC\_ALR 寄存器。

另外，对 RTC 任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询 RTC\_CR 寄存器中的 RTOFF 状态位，判断 RTC 寄存器是否处于更新中。仅当 RTOFF 状态位是“1”时，才可以写入 RTC 寄存器。

**配置过程：**

- 查询 RTOFF 位，知道 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。
- 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 RTCCLK 周期。

### 17.3.5 RTC 标志的设置

在每一个 RTC 核心的时钟周期中，更改 RTC 计数器之前设置 RTC 秒标志（SECF）。

在计数器到达 0x0000 之前的最后一个 RTC 时钟周期中，设置 RTC 溢出标志（OWF）。

在计数器的值到达闹钟寄存器的值加 1 (RTC\_ALR+1) 之前的 RTC 时钟周期中，设置 RTC\_Alarm 和 RTC 闹钟标志（ALRF）。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志同步：

- 时钟 RTC 闹钟中断，并在中断处理程序中修改 RTC 闹钟和/或 RTC 计数器
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟和/或 RTC 计数器。

图 146. RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004

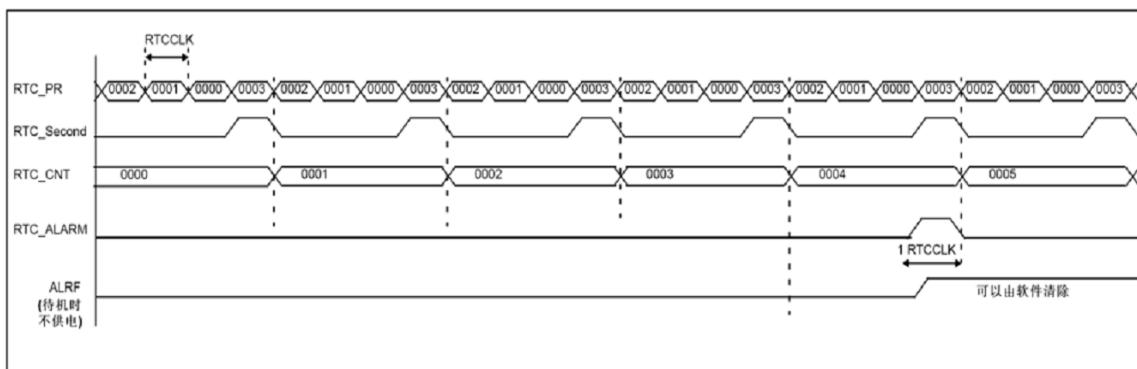
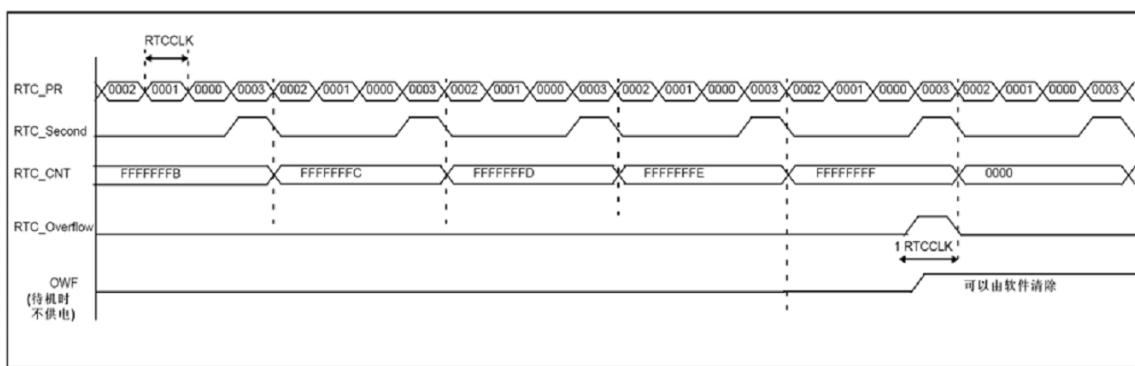


图 147. RTC 溢出波形图示例, PR=0003



## 17.4 RTC 寄存器描述

### 17.4.1 RTC 控制寄存器高位 (RTC\_CRH)

地址偏移量: 0x00

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留														OWIE	ALRIE	SECI E
														rw	rw	rw

位31: 3	保留, 被硬件强制为0
位2	OWIE: 允许溢出中断位 (Overflow interrupt enable) 0: 屏蔽(不允许)溢出中断 1: 允许溢出中断
位1	ALRIE: 允许闹钟中断 (Alarm interrupt enable) 0: 屏蔽(不允许)溢出中断 1: 允许溢出中断
位0	SECIE: 允许秒中断 (Second interrupt enable) 0: 屏蔽(不允许)溢出中断 1: 允许溢出中断

这些位用来屏蔽中断请求。

注: 系统复位后所有的中断被屏蔽, 因此可通过写 RTC 寄存器来确保在初始化后没有被挂起的中断请求。当外设正在完成前一次写操作时 (标志位 RTOFF=0), 不能对 RTC\_CRH 寄存器进行写操作。

RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成。(见 17.3.4 节)

### 17.4.2 RTC 控制寄存器低位(RTC\_CRL)

偏移地址: 0x04

复位值: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								RTOFF	CNF	RSF	OWF	ALRF	SECF		
								r	rw	rc w0	rc w0	rc w0	rc w0		

位0	SECIE: 允许秒中断 (Second interrupt enable)) 0: 屏蔽(不允许)溢出中断 1: 允许溢出中断
位15: 6	保留, 被硬件强制为0
位5	RTOFF: RTC操作关闭 (RTC operation OFF) RTC模块利用这位来指示对其寄存器进行的最后一次操作的状态, 指示操作是否完成。若此位为“0”, 则表示无法对任何的RTC寄存器进行读写操作。此位为只读位。 0: 上一次对RTC寄存器的写操作仍在进行; 1: 上一次对RTC寄存器的写操作已经完成
位4	CNF: 配置标志 (Configuration flag) 此位必须由软件置“1”以静茹配置模式, 从而允许RTC_CNTL/H、RTC_ALRL/H或RTC_PRLL/H寄存器写入数据。只有当此位在被置“1”并重新由软件清“0”后, 才会执行写操作。 0: 退出配置模式(开始更新RTC寄存器) 1: 进入配置模式
位3	RSF: 寄存器同步标志 (Registers synchronized flag) 每当RTC_CNT寄存器和RTC_DIV寄存器由软件更新或清“0”时, 此位由硬件置“1”。在APB1复位后, 或APB1时钟停止后, 此位必须由软件清“0”。要进行任何的读操作之前, 用户程序必须等待这位被硬件置“1”, 以确保RTC_CNT、RTCALR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
位2	OWF: 溢出标志 (Overflow flag) 当32位可编程计数器溢出时, 此位由硬件置“1”。如果RTC_CRH寄存器中的OWIE=1, 则产生中断。此位只能由软件清“0”。对此写“1”无效 0: 无溢出 1: 32位可编程计数器溢出
位1	ALRF: 闹钟标志 (Alarm flag) 当32位可编程计数器到达了RTC_ALR寄存器所设置的预定值, 此位由硬件置“1”。如果RTC_CRH寄存器中的ALRIE=1, 则产生中断。此位只能由软件清“0”。对此位写“1”是无效的。 0: 无闹钟 1: 有闹钟

位0	<b>SECF:</b> 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计数器提供了一个周期性信号（通常1秒）。如果RTC_CRH寄存器SECIE=1，则产生中断。此位只能由软件清除。对此位写“1”是无效的。 0: 秒标志条件不成立 1: 秒标志条件成立
----	---

RTC 的功能由这个控制寄存器控制。当前一个写操作还未完成时（RTOFF=0 时），不能写 RTC\_CR 寄存器。

- 注： 1. 任何标志位都将保持挂起状态，直到适当的 RTC\_CR 请求位被软件复位，表示所请求的中断已经被接受
- 2. 在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作。
  - 3. 当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新
  - 4. OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零
  - 5. 若 ALRF=1 且 ALRIE=1，则允许产生 RTC 全局中断。如果在 EXTI 控制寄存器中语序产生 EXTI 线 17 断，则允许产生 RTC 全局中断和 RTC 闹钟中断。
  - 6. 若 ALRF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的时间模式，则这条线上会产生一个脉冲（不会产生 RTC 闹钟中断）。

#### 17.4.3 RTC 预分频装载寄存器 (RTC\_PRLH/RTC\_PRLL)

预分频装载寄存器用来保护 RTC 预分频器的周期计数值。它们受 RTC\_CR 寄存器的 RTOFF 位保护，仅当 RTOFF 值为“1”时允许进行写操作

##### RTC 预分频装载寄存器高位 (RTC\_PRLH)

偏移地址：0x08

只写（见 17.3.4 节）

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
保留														PRL[19: 16]			
														W	W	W	W

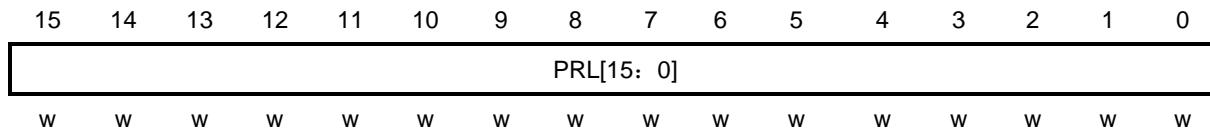
位15: 4	保留，被硬件强制为0
位3: 0	<b>PRL[19: 16]:</b> RTC预分频器装载值高位 (RTC prescaler reload value high) 根据以下公式，这些位用来定义计数器的时钟频率： $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19: 0]+1)$ 注：不推荐使用0值，否则无法正确产生RTC中断和标志位

**RTC 预分频装载寄存器低位 (RTC\_PRL)**

偏移地址: 0x0C

只写 (见 17.3.4 节)

复位值: 0x8000



位15: 0	PRL[15: 0]: RTC预分频器装载值低位 (RTC prescaler reload value low) 根据以下公式, 这些位用来定义计数器的时钟频率: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19: 0]+1)$
--------	--

注: 如果输入时钟频率是 32.768kHz( $f_{RTCCLK}$ ), 这个寄存器中写入 7FFFFh 可获得周期为 1 秒钟的信号

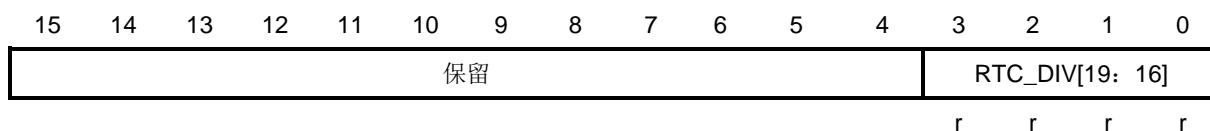
**17.4.4 RTC 预分频器分频因子寄存器 (RTC\_DIVH/RTC\_DIVL)**

在 TR\_CLK 的每个周期里, RTC 预分频器中计数器的值都会被重新设置为 RTC\_PRL 寄存器的值。用户可通过读取 RTC\_DIV 寄存器, 以获得预分频计数器的当前值, 而不停止分频计数器的工作, 从而获得精确的时间测量。此寄存器是只读寄存器, 其值再 RTC\_PRL 或 RTC\_CNT 寄存器中的值发生改变后, 由硬件重新装载。

**RTC 预分频器分频因子寄存器高位 (RTC\_DIVH)**

偏移地址: 0x10

复位值: 0x0000

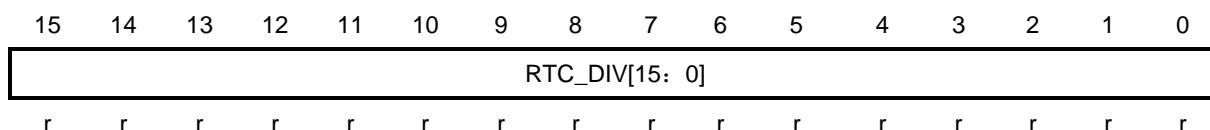


位15: 4	保留, 被硬件强制为0
位3: 0	RTC_DIV[19: 16]: RTC时钟分频器分频因子高位 (RTC clock divider high)

**RTC 预分频器分频因子寄存器低位 (RTC\_DIVL)**

偏移地址: 0x14

复位值: 0x8000



位15: 0	RTC_DIV[15: 0]: RTC时钟分频器分频因子低位 (RTC clock divider low)
--------	--

#### 17.4.5 RTC 计数器寄存器 (RTC\_CNTH/RTC\_CNTL)

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 TR\_CLK 时间基准为参考进行计数。RTC\_CNT 寄存器用于存放计数器的计数值。他们受 RTC\_CR 的位 RTOFF 写保护，仅当 RTOFF 值为“1”时，允许写操作。在高或低寄存器 (RTC\_CNTH 或 RTC\_CNTL) 上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。在进行读操作时，直接返回计数器内的计数值（系统时间）。

##### RTC 计数器寄存器高位 (RTC\_CNTH)

偏移地址: 0x18

复位值: 0x0000

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

位15: 0	RTC_CNT[31: 16]: RTC计数器高位 (RTC counter high) 可通过读RTC_CNTH寄存器来获得RTC计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式。
--------	--

##### RTC 计数器寄存器低位 (RTC\_CNTL)

偏移地址: 0x1C

复位值: 0x0000

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

位15: 0	RTC_CNT[15: 0]: RTC计数器低位 (RTC counter low) 可通过读RTC_CNTL寄存器来获得RTC计数器当前值的低位部分。要对此寄存器进行写操作前，必须先进入配置模式。
--------	--

#### 17.4.6 RTC 闹钟寄存器 (RTC\_ALRH/RTC\_ALRL)

当可编程计数器的值与 RTC\_ALR 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器受 RTC\_CR 寄存器里的 RTOFF 位写保护，仅当 RTOFF=1 时，允许写操作。

##### RTC 闹钟寄存器高位 (RTC\_ALRH)

偏移地址: 0x20

只写 (见 17.3.4 节)

复位值: 0xFFFF

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

位15: 0	RTC_ALR[31: 16]: RTC闹钟值高位 (RTC alarm high) 此寄存器用来保护由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先进入配置模式
--------	---

##### RTC 计数器寄存器低位 (RTC\_ALRL)

偏移地址: 0x24

只写 (见 17.3.4 节)

复位值: 0xFFFF

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

位15: 0	RTC_ALR[15: 0]: RTC闹钟值低位 (RTC alarm low) 此寄存器用来保护由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入配置模式
--------	---

## 18. I<sup>2</sup>C 接口

### 18.1 I<sup>2</sup>C 简介

I<sup>2</sup>C(芯片间)总线接口连接微控制器和串行 I<sup>2</sup>C 总线。它提供多主机功能，控制所有 I<sup>2</sup>C 总线特定的时序、协议、仲裁和定时。

I<sup>2</sup>C 总线是一个两线串行接口，其中两线位串行数据 (SDA) 和串行时钟 (SCL) 线在连接到总线器件间传递信息。每个器件都有一个唯一的地址识别，而且都可以作为一个发送或接收器。除了发送器和接收器外，器件在执行数据传输时也可以被看做是主机或者从机。主机是初始化总线的数据传输并产生允许传输的时钟信号的器件。此时，任何被寻址的器件都被认为是从机。

I<sup>2</sup>C 可以工作在标准模式（数据传输速率为 0~100Kb/s），快速模式（数据传输速率最大为 400Kb/s）以及高速模式（数据传输速率最大为 3.4Mb/s）。

### 18.2 I<sup>2</sup>C 主要特征

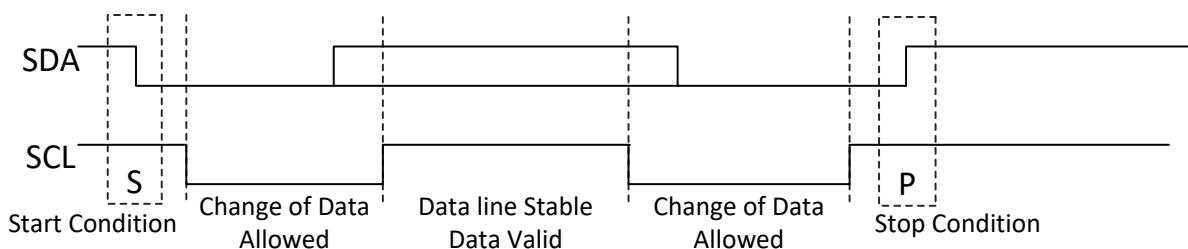
- 并行总线 I<sup>2</sup>C 总线协议转换器
- 半双工同步操作
- 支持主从模式
- 支持 7 位地址和 10 位地址
- 支持标准模式 100Kbps，快速模式 400Kbps 和高速模式 3.4Mbps
- 产生 Start、stop、重新发 start、应答 Acknowledge 信号检测
- 在主模式下只支持一个主机
- 分别有 2 字节的发送和接收缓冲
- 在 scli 和 sdai 上增加了无毛刺电路
- DMA 操作（将在未来的版本支持，本版本不建议使用）
- 支持中断和查询操作

### 18.3 I<sup>2</sup>C 协议

#### 18.3.1 起始和停止条件

当总线处于空闲状态时，SCL 和 SDA 同时被外部上拉电阻拉为高电平。当主机启动数据传输时，必须先产生一个起始条件。在 SCL 线是高电平时，SDA 线从高电平向低电平切换表示起始条件。当主机结束传输时要发送停止条件。在 SCL 线是高电平，SDA 线由低电平向高电平切换表示停止条件。下图显示了起始和停止条件的时序图。数据传输过程中，当 SCL 为 1 时，SDA 必须保持稳定。

图 148. 起始和停止条件



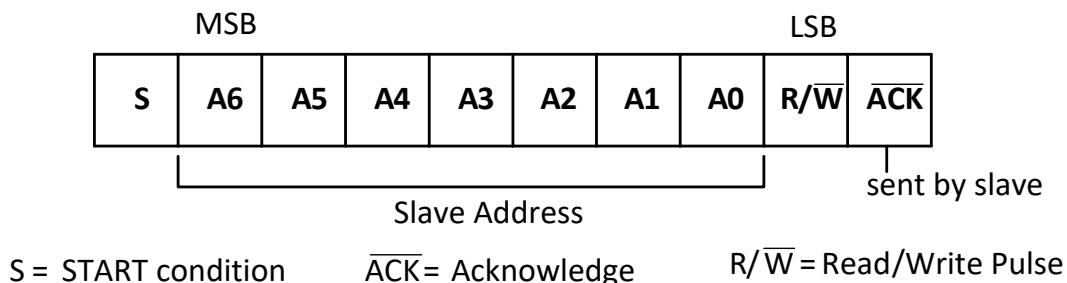
#### 18.3.2 从机寻址协议

I<sup>2</sup>C 有两种地址格式：7 位的地址格式和 10 位的地址格式

## 7 位的地址格式

下图中显示在起始条件(S)后发送的一个字节的前 7 位(bit 7: 1)为从机地址，最低位(bit 0)是数据方向位，当 bit 0 为 0，表示主机写数据到从机，1 表示主机从从机读数据。

图 149. 7 位的地址格式

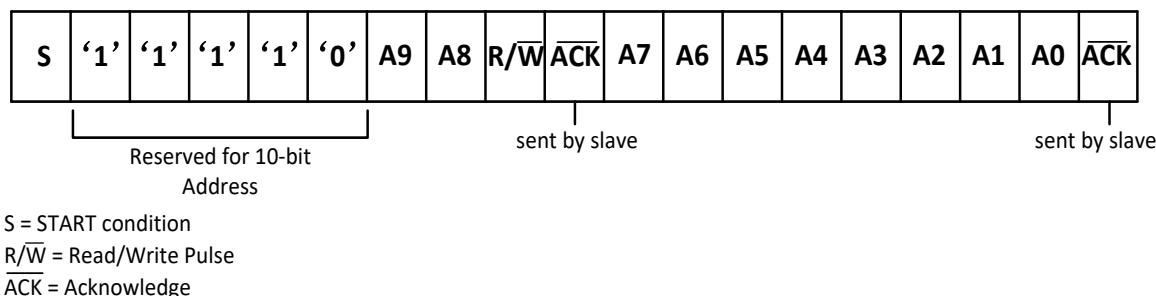


## 10 位的地址格式

在 10 位的地址格式中，发送 2 个字节来传输 10 位地址。发送的第一个字节的位的描述如下：第一个 5 位(bit 7: 3)用于告示从机接下来是 10 位的传输。第一个字节的后两个字节 (bit 2: 1) 位从机地址的 bit 9: 8，最低位 (bit 0) 是数据方向位(R/W)。传输的第二个字节为 10 位地址的低八位。

具体如下图所示：

图 150. 10 位的地址格式



下表定义了 I2C 首字节的特殊用途和保留地址：

表 33. I2C 首字节

从机地址	R/W位	描述
0000 000	0	广播呼叫地址。I2C将数据放入接收缓冲，并产生一个广播呼叫中断
0000 000	1	起始字节
0000 001	X	CBUS地址。I2C接口忽略该访问
0000 010	X	保留
0000 011	X	保留
0000 1xx	X	高速模式主机码
1111 1xx	X	保留
1111 0xx	X	10位从机寻址

### 18.3.3 发送和接收协议

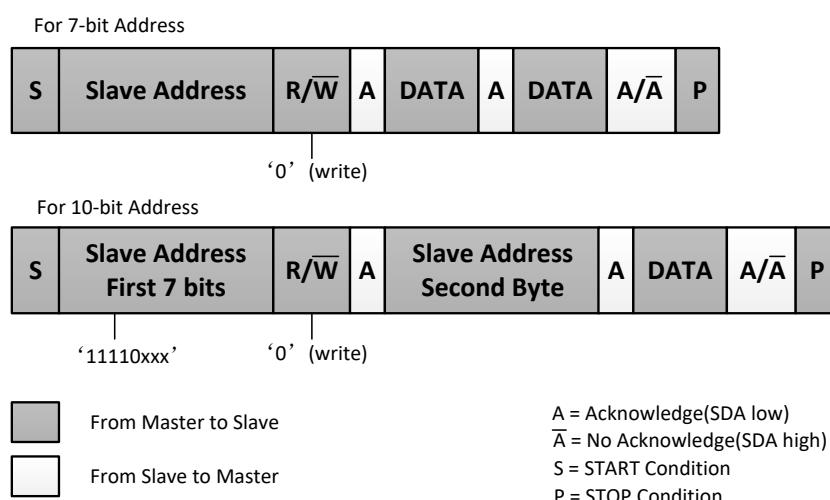
主机初始化数据传输并且从总线上发送或接收数据，作为主发送或者主接收。从机响应主机的请求来发送或接收数据，作为从发送或从接收器。

#### 主发送和从接收

所有数据都是以字节格式传输，且不限制每次传输的字节数。当主机发送完地址和 R/W 位或者主机发送一个字节的数据到从机上，从接收器必须产生一个响应信号 (ACK)。当从接收器不能产生 ACK 响应信号，主机将会产生一个停止条件中止传输。从机不能响应时，必须释放 SDA 为高电平才能使得主机产生停止条件。

当主发送器传输数据如下图所示，从接收器在接收到的每个字节后产生一个 ACK 来响应主发送器。

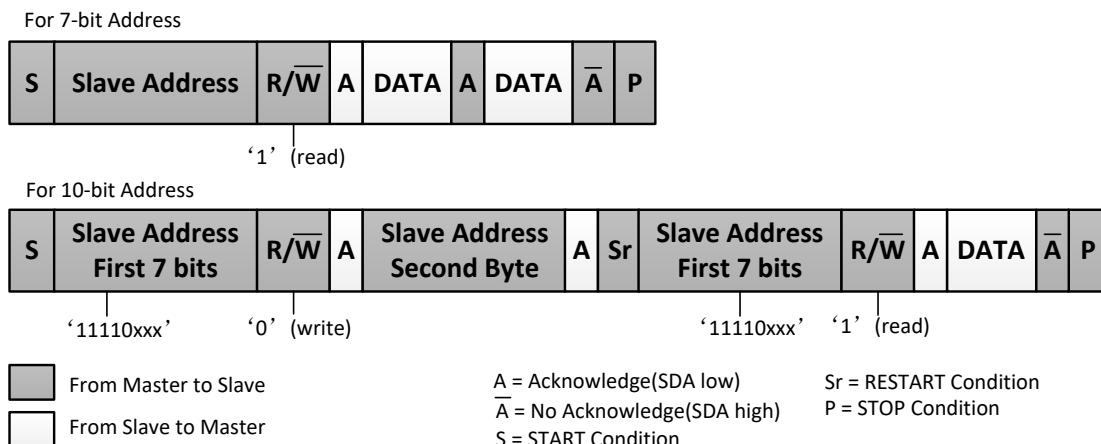
图 151. 主发送协议



## 主接收和从发送

当主机接收数据如下图所示，主机必须在每次接收到一个字节数据后响应从发送器，除了最后一个字节。通过这种方式，主接收器能通知从发送器是否是最后一个字节。从发送器在检测到 NACK 时必须释放 SDA，这样主机可以产生停止条件。

图 152. 主接收协议



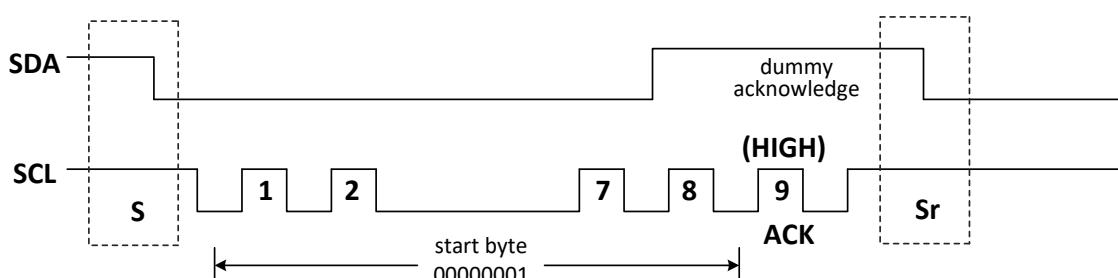
当主机不想产生停止条件而释放总线，可以产生一个重复起始条件。重复起始条件与起始条件相同，只是它实在 ACK 后产生。工作在主机模式下，I2C 接口可以使用不同的传输方向与相同的从机通信。

### 18.3.4 起始字节传输协议

起始字节传输协议是用来给没有专用的 I2C 硬件模块的系统使用。当 I2C 模块作为主机时，在每次传输开始可以给需要的从机产生起始字节输出。

该协议由 7 个 0 以及一个 1 组成，如下图所示。处理器可以在地址阶段用低速采样 0 来查询总线。一旦检测到 0，处理器可以从低速采样切换到主机的正常速率。

图 153. 起始字节传输



起始字节程序流程如下：

1. 主机产生一个起始条件
2. 主机发送起始字节 (0000 0001)
3. 主机发送 ACK 时钟脉冲(ACK)
4. 没有从机响应 ACK 信号
5. 主机产生重复起始条件 (RESTART)

硬件 I2C 接收器不需要响应开始字节，因为这是一个保留地址，而且地址会在 RESTART 后复位。

### 18.3.5 发送缓冲管理以及起始、停止和重复起始条件产生

当工作在主机模式，每当发送为空时 I2C 模块就在总线上产生一个停止条件。如果重复起始产生功能使能 (IC\_RESTART\_EN=1)，则传输方向从读变为写或者写变为读时产生重复起始条件。如果没有使能重复起始条件，则会在停止条件后产生一个起始条件。

下图显示了 IC\_DATA\_CMD 寄存器的位。

图 154. IC\_DATA\_CMD 寄存器

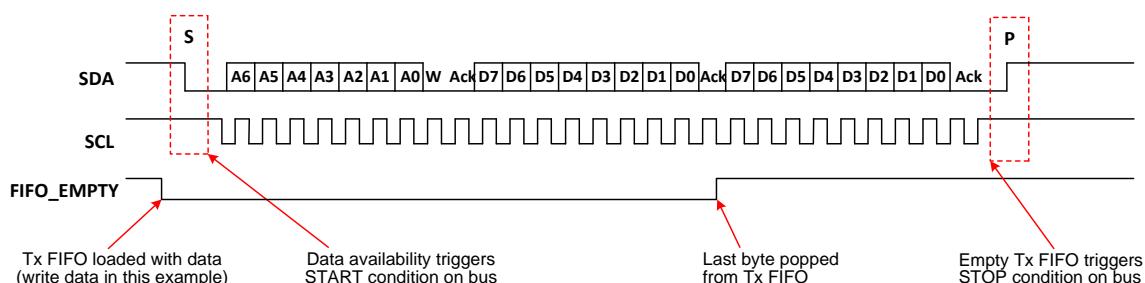
IC_DATA_CMD	CMD	DATA
	8	7
		0

DATA —Read/Write field; data retrieved from slave is read from this field ; data to be sent to slave is written to this field.

CMD —Write-only field; this bit determines whether transfer to be carried out is read (CMD=1) or Write (CMD=0)

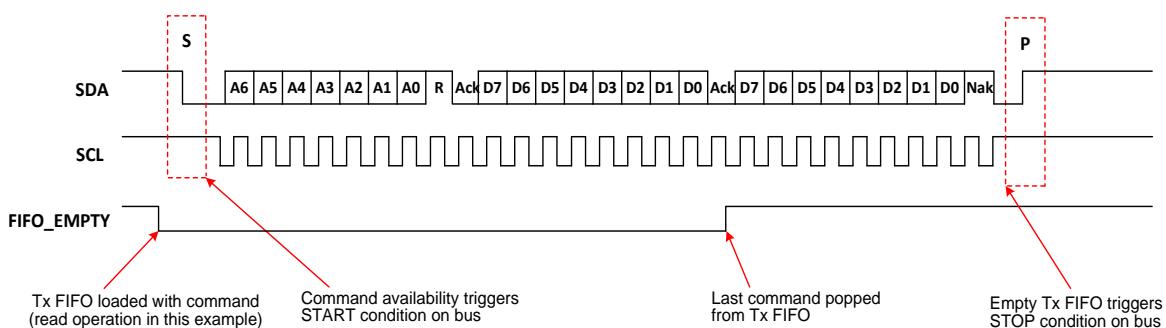
下面的时序图描述了 I2C 模块工作在主发送模式下 Tx FIFO 变为空时行为。

图 155. 主发送 — Tx FIFO 为空



下面的时序图描述了 I2C 模块工作在主接收模式下当 Tx FIFO 变为空时行为。

图 156. 主接收 — Tx FIFO 为空



### 18.3.6 多个主机仲裁

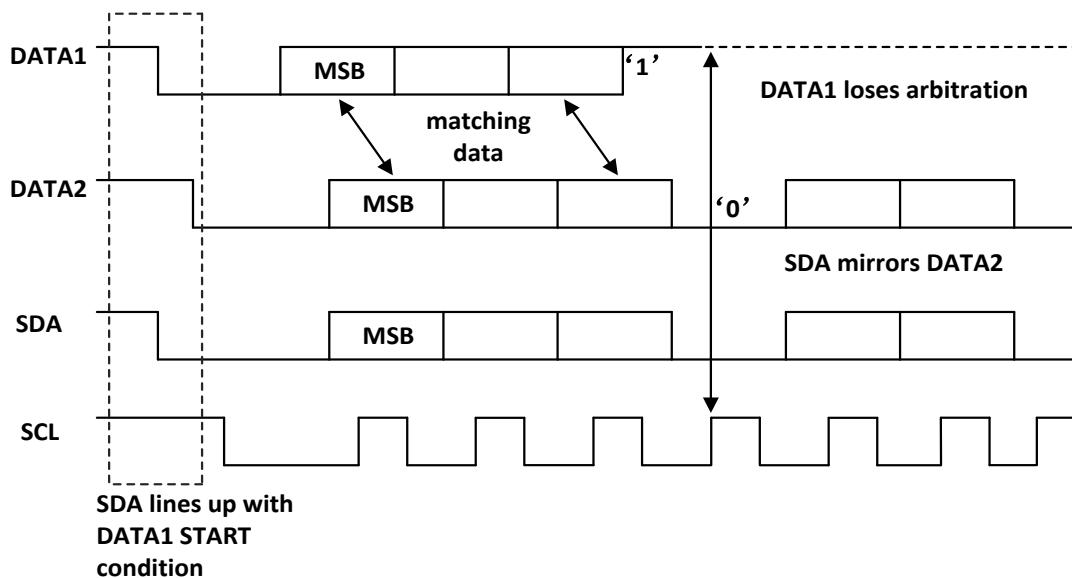
I2C 总线是一个多主机的总线。仲裁是一个在有多个主机同时尝试控制总线，但只允许其中一个控制总线并使报文不被破坏的过程。一旦其中一个主机已经控制了总线，那么直到该主机发送一个停止条件并且将总线释放为空闲状态时，其他主机才能控制总线。

当 SCL 线是高电平时，仲裁在 SDA 线发生。如果两个或多个主机尝试发送信息到总线，在其他主机都产生“0”的情况下，首先产生一个“1”的主机将丢失仲裁。丢失仲裁的主机可以继续产生时钟脉冲直到字节传输结束。如果每个主机都尝试寻址相同的器件，仲裁会继续在数据阶段进行。

检测到丢失仲裁后，I2C 接口会停止产生 SCL 信号。

下图显示了两个主机的仲裁的总线时序

图 157. 多个主机仲裁



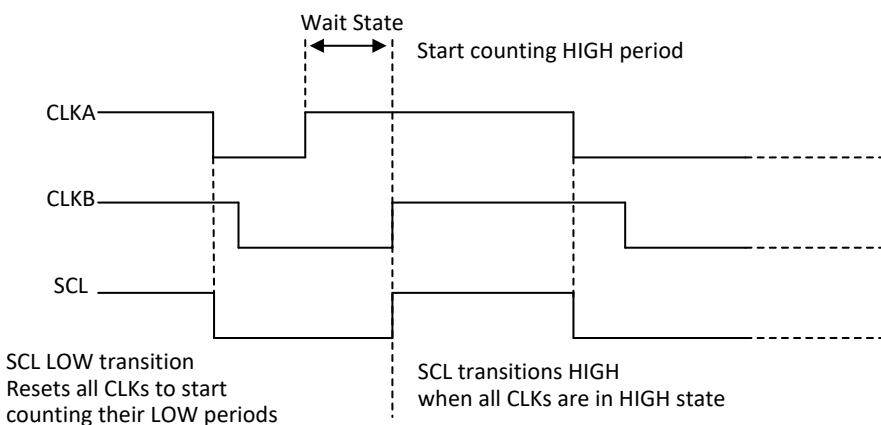
对于高速模式，每个主机设置了独一无二的主机码，所以仲裁不会进入数据阶段。这 8 位的主机码由软件写入到高速主机模式码寄存器（IC\_HS\_MADDR）。由于每个主机码都不相同，因此当传输高速主机码之后只有一个主机可以赢得仲裁。

### 18.3.7 时钟同步

当两个或多个主机试图同时在总线传输信息时，他们必须仲裁和同步 SCL 时钟。所有的主机产生自己的时钟来传输消息。数据只在时钟的高电平有效。时钟同步是通过 SCL 信号的线“与”连接进行的。当主机把 SCL 时钟变成 0，主机会计算 SCL 低电平的时间，在下一个时钟周期开始把 SCL 时钟变成 1。但是，假如另一个主机把 SCL 保持为 0，那么这个主机会进入等待状态直到 SCL 时钟变为 1。

所有的主机会计算它们的高电平时间，最短高电平时间的主机会把 SCL 变为 0。接下来主机会计算低电平时间，最长低电平时间的主机会强制其他主机进入等待状态。这样就产生一个同步后的 SCL 时钟，如下图所示。

图 158. 多个主机时钟同步



## 18.4 I2C 工作模式

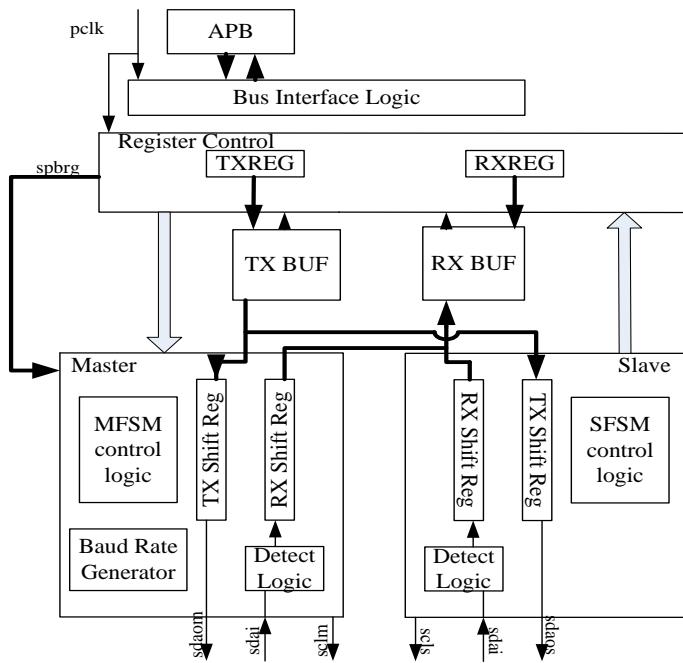
I2C 接口可以以下述 4 种方式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

**注：**I2C 接口模块只能工作在主机模式或者从机模式，但不能同时工作在两种模式下。因此要确保寄存器 IC\_CON 中位 6 (IC\_SLAVE\_DISABLE) 和位 0 (IC\_MASTER\_MODE) 不能分别设置为 0 和 1 (或者分别为 1 和 0)。

I2C 功能框图如下：

图 159. I2C 功能框图



### 18.4.1 从模式

下面介绍从模式的程序流程图

#### 初始化配置

1. 写 0 到 IC\_ENABLE 寄存器位 0 禁止 I2C.
2. 通过初始化 IC\_SAR 寄存器来配置从机地址。该地址为 I2C 接口所响应的地址。
3. 配置 IC\_CON 寄存器指定地址格式（设置 bit 3 来选择 7 位或 10 位地址格式）。写 0 到寄存器 IC\_CON 寄存器的位 6 (IC\_SLAVE\_DISABLE) 和写 0 到位 0 (MASTER\_MODE) 。
4. 写 1 到 IC\_ENABLE 寄存器中位 0 来使能 I2C 接口模块。

#### 从发送的单字节操作

当 I2C 接口被其他 I2C 主机寻址并请求数据的时候，I2C 接口工作在从发送模式，步骤如下：

1. 其他 I2C 主器件初始化 I2C 传输，发送地址与 IC\_SAR 寄存器中的从机地址匹配。
2. I2C 接口响应发送的地址，识别传输的方向是工作在从发送模式。
3. I2C 接口产生 RD\_REQ 中断（寄存器 IC\_RAW\_INTR\_STAT 位 5），并且将 SCL 线拉低。总线一直处于等待状态直到软件响应。

如果 RD\_REQ 中断被屏蔽(寄存器 IC\_INTR\_MASK[5]=0)，建议 CPU 定期查询 IC\_RAW\_INTA\_STAT 寄存器。

1. IC\_RAW\_INTR\_STAT 位 5 置位等效于产生了一个 RD\_REQ 中断。
2. 软件必须满足 I2C 传输的要求。
3. 时间间隔通常在 10 个 SCL 时钟周期左右。例如，对于 400kb/s，时间间隔是 25us。
4. 如果在接收读请求之前 Tx FIFO 仍然有数据，I2C 接口就会产生一个 TX\_ABRT 中断 (IC\_RAW\_INTR[6])，清空 Tx FIFO 中的数据。
5. 软件写数据到 IC\_DATA\_CMD 寄存器（其中位 8 设置为 0）。
6. 软件必须先清除 IC\_RAW\_INTA\_STAT 寄存器 RD\_REQ 和 TX\_ABRT 中断（分别为 bit5, 6）
7. I2C 接口释放 SCL，并发送数据字节。
8. 主器件发送重复起始条件控制总线或者发送停止条件释放总线。

#### 从接收的单字节操作

当其他主器件寻址 I2C 接口并且发送数据，I2C 接口工作在从接收模式，步骤如下：

1. 其他 I2C 主器件初始化 I2C 传输，发送地址与 IC\_SAR 寄存器中的从机地址匹配。
2. I2C 接口响应发送的地址，识别传输的方向是工作在从接收模式。
3. I2C 接口收到主机发送的数据并将数据存储在接收缓冲中。
4. I2C 接口产生 RX\_FULL 中断 (IC\_RAW\_INTR\_STAT[2]) .

如果 RX\_FULL 中断被屏蔽 (IC\_INTR\_MASK[2]=0)，建议软件定期查询 IC\_STATUS 寄存器中。读到 IC\_STATUS 寄存器位 3 (RFNE) 为 1 时等效于 RX\_FULL 中断产生。

5. 软件通过读 IC\_DATA\_CMD 寄存器中的 bit 7: 0 来获得接收到的数据。
6. 主器件发送重复起始条件控制总线或者发送停止条件释放总线。

#### 从机的块传输操作

标准的 I2C 协议中，所有的数据处理都是单个字节的处理，程序通过写一个字节到从机的 Tx FIFO 响应主机的读请求。当一个从机（从发送）接收到主机（主接收）的读请求 (RD\_REQ) 时，最少有一个数

据放到从发送的 Tx FIFO。这个 I2C 接口模块可以处理 Tx FIFO 中有多个数据，所以接下来的读请求不需要再产生中断来取数据。最终，这极大的减少了因为每次数据中断导致等待时间。

该模式仅存在当 I2C 接口作为从发送模式。如果主机发送响应从发送传输的数据，从机的 TX FIFO 中没有数据，I2C 接口将拉低 I2C 总线的 SCL 线直到读请求中断 (RD\_REQ) 产生并且 TX FIFO 的数据准备好后才释放 SCL 线。

如果 RX\_REQ 中断被屏蔽 (IC\_INTR\_STAT[5]=0)，软件可以定期查询读 IC\_RAW\_INTR\_STAT 寄存器。当读到 IC\_RAW\_INTR\_STAT[5] 返回为 1 等效于产生了 RX\_REQ 中断。

RD\_REQ 中断由于读请求产生，像中断一样必须退出中断服务程序 (ISR) 时清除。在中断服务程序中 (ISR) 可以写一个或多个字节的数据到 TX FIFO。在这些字节传输给主机的过程中，如果主机响应了最后一个字节，从机将必须再次产生 RD\_REQ 中断请求。这是因为主机要求更多的数据。

如果主机接收了来自 I2C 接口的 n 字节，但是程序写到 Tx FIFO 中的数据个数大于 n，从机在完成要求的 n 字节的数据发送后，将会清空 Tx FIFO 并且忽略额外的字节。

#### 18.4.2 主模式

##### 初始化配置

1. 通过设置 IC\_ENABLE [0]=0 来禁止 I2C 接口
2. 配置 IC\_CON 寄存器的 bit 2: 1 设置 I2C 工作的速率模式（标准模式、快速模式以及高速模式）。同时确保 bit 6 (IC\_SLAVE\_DISABLE) 为 1，且 bit 0 (MASTER\_MODE) 为 1.
3. 往 IC\_TAR 寄存器写入 I2C 器件地址。设置该寄存器可配置为广播地址或起始字节命令。
4. 仅供高速模式传输：写入数据到 IC\_HS\_MADDR 配置 I2C 接口主机码。该主机码是由软件自己定义。
5. 置位 IC\_ENABLE[0]使能 I2C 接口。
6. 将传输的数据以及传输方向写入到 IC\_DATA\_CMD 寄存器中。如果在使能 I2C 接口之前配置了 IC\_DATA\_CMD 寄存器，数据和命令都会丢失，这是因为在 I2C 接口禁止的情况下缓冲是清空的。

以上的步骤将会使得 I2C 接口产生一个起始条件并发送地址字节数据到 I2C 总线上。

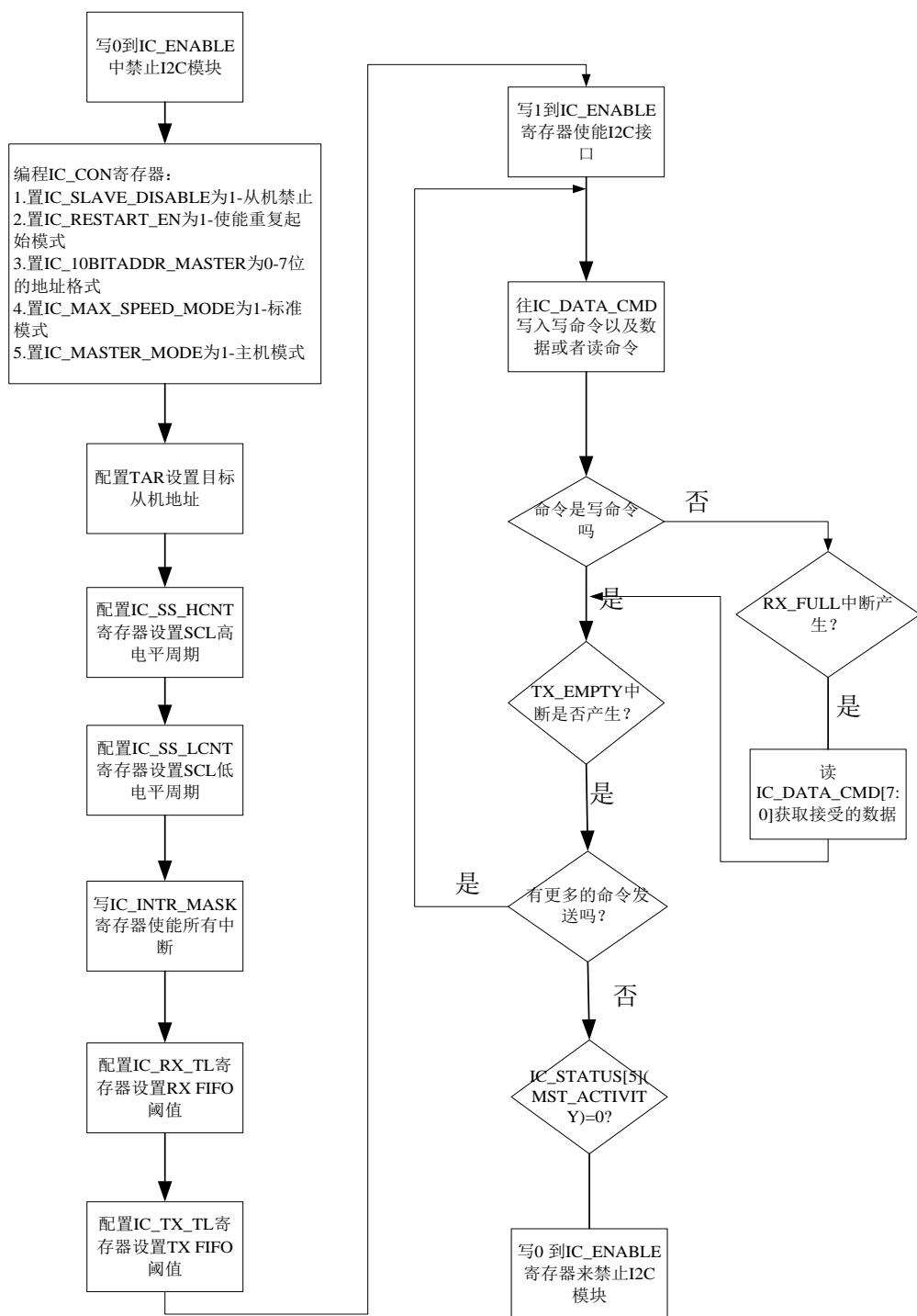
##### 主发送和主接收

I2C 接口支持读写的动态切换。当发送数据时，写数据到 I2C RX/TX 数据缓冲和命令寄存器的低字节中 (IC\_DATA\_CMD)，配置 CMD 位为 0 产生写操作。接下来的读命令，不需要设置 IC\_DATA\_CMD 寄存器的低字节，只需要确保 CMD 位为 1。如果发送 FIFO 为空，I2C 模块拉低 SCL 直到下个命令写入到发送 FIFO 中。

##### 程序流程图

下面的流程图为 I2C 接口作为主机的程序示例：

图 160. I2C 接口主机流程图



### 18.4.3 I2C 中止传输

IC\_ENABLE 寄存器中的 ABRT 控制位允许软件在完成 TX FIFO 中传输命令之前放弃 I2C 总线。作为 ABORT 请求的响应, I2C 模块发出一个停止条件到 I2C 总线, 同时清空 TX FIFO。中止传输操作值允许在主模式下。

#### 程序流程

1. 停止往 Tx FIFO (IC\_DATA\_CMD) 中写新的命令

2. 如果工作在 DMA 模式中，置 TDMAE=0 禁止发送 DMA。
3. 置 IC\_ENABLE 寄存器 ABRT 位为 1
4. 等待 TX\_ABRT 中断

## 18.5 利用 DMA 通信（将在未来的版本支持，本版本不建议使用）

I2C 接口支持用 DMA 来发送和接收数据。通过设置 IC\_DMA\_CR 寄存器中的对应位可以单独开启 DMA 发送或者 DMA 接收。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。

### 利用 DMA 发送

通过设置 IC\_DMA\_CR 寄存器的 TDMAE 位可以激活 DMA 发送模式。为 I2C 分配好 DMA 通道后，当发送数据时，DMA 控制器会将数据从预置的存储区装载进 IC\_DATA\_CMD 寄存器。

### 利用 DMA 接收

通过设置 IC\_DMA\_CR 寄存器的 RDMAE 位可以激活 DMA 接收模式。为 I2C 分配好 DMA 通道后，当每次接收到数据字节时，DMA 控制器会将数据从 IC\_DATA\_CMD 寄存器中传送到预置的存储区。

## 18.6 I2C 中断

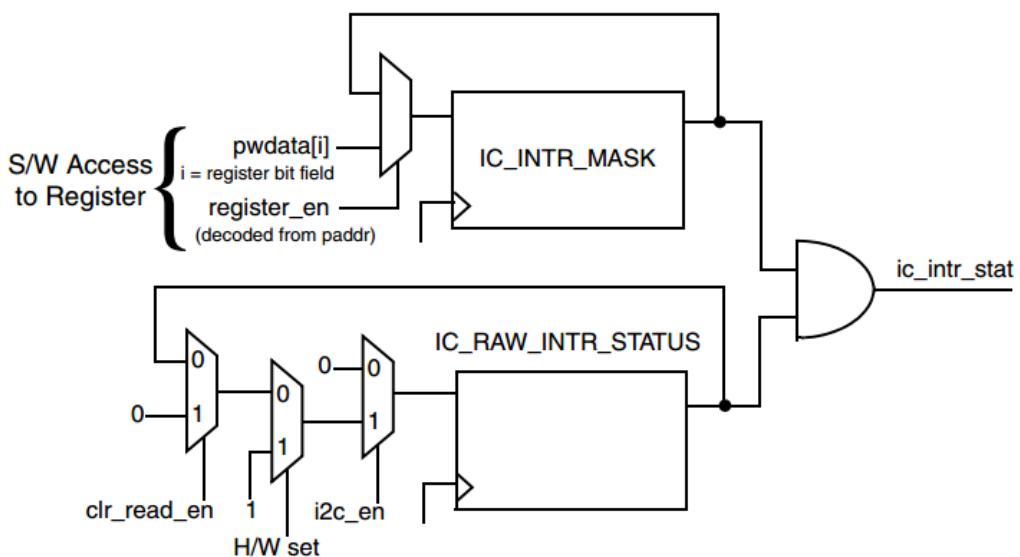
下表列出了 I2C 的中断位以及它们的设置和清除方式。部分位由硬件置位并由软件清除；另一部分位由硬件置位和清除。

表 34. 中断位的置位和清除

中断位	硬件置位/软件清除	硬件置位和清除
GEN_CALL	√	✗
START_DET	√	✗
STOP_DET	√	✗
ACTIVITY	√	✗
RX_DONE	√	✗
TX_ABRT	√	✗
RD_REQ	√	✗
TX_EMPTY	✗	√
TX_OVER	√	✗
RX_FULL	✗	√
RX_OVER	√	✗
RX_UNDER	√	✗

下图描述了中断寄存器中，中断位被硬件置位和软件清除的操作

图 161. 中断机制

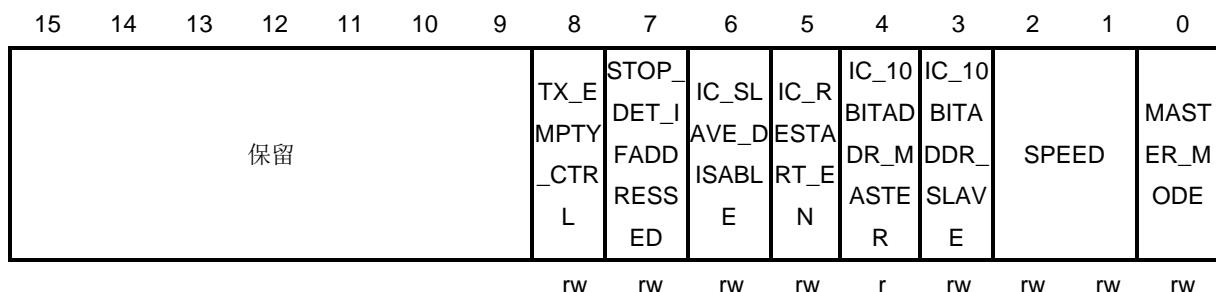


## 18.7 I2C 寄存器描述

### 18.7.1 I2C 控制寄存器 (IC\_CON)

偏移地址: 0x00

复位值: 0x0011



位31: 9	保留, 始终读为0
位8	<b>TX_EMPTY_CTRL:</b> 该位控制TX_EMPTY中断产生, 细节参考IC_RAW_INTR_STAT寄存器。 (This bit controls the generation of the TX_EMPTY interrupt, as described in the IC_RAW_INTR_STAT register.)
位7	<b>STOP_DET_IFADDRESSED:</b> 在从机模式下, 是否产生STOP_DET中断。 1-当地址匹配时才产生STOP_DET中断 (In the slave mode ,1'b1 – issues the STOP_DET interrupt only when it is addressed) 0-无论地址是否匹配, 都产生STOP_DET中断 (In the slave mode ,1'b0 – issues the STOP_DET irrespective of whether it's addressed or not) 该位仅适用于从机模式 注: 广播地址寻址时, 如果该位置位, 从机不产生STOP_DET中断。STOP_DET中断仅当发送的地址与从机地址匹配时产生。

位6	<b>IC_SLAVE_DISABLE:</b> 该位控制I2C接口从机禁止 (This bit controls whether I2C has its slave disabled) 0: 从机使能 1: 从机禁止
位5	<b>IC_RESTART_EN:</b> 当作为主机时该位控制是否发送RESTART条件 (Determines whether RESTART conditions may be sent when acting as a master) 0: 禁止 1: 使能 当RESTART禁止, I2C接口作为主机时不能执行以下功能: 发送起始字节 执行工作在高速模式下 组合格式模式下改变传输方向 10位的地址格式的读操作 替换RESTART条件为先发送停止条件再发送起始条件。如果上述操作执行会置位 IC_RAW_INTR_STAT 寄存器的位6(TX_ABRT)
位4	<b>IC_10BITADDR_MASTER:</b> I2C作为主机时的地址格式 (Address mode when acting as a master) 0: 7位的地址格式 1: 10位的地址格式
位3	<b>IC_10BITADDR_SLAVE:</b> 当作为从机时, 该位控制响应10位或者7位地址 (When acting as a slave, this bit controls whether the DW_apb_i2c responds to 7- or 10-bit addresses) 0: 7位的寻址地址。I2C接口忽略处理10位的寻址。对于7位寻址, 仅比较 IC_SAR 寄存器的低7位。 1: 10位的寻址地址。I2C仅响应10位的寻址, 接收地址与IC_SAR的10位比较
位2: 1	<b>SPEED:</b> 该两位控制I2C接口工作的速率模式 (These bits control at which speed the DW_apb_i2c operates) 该设置仅当I2C接口工作在主机模式下有效。 1: 标准模式(0~100Kb/s) 2: 快速模式( $\leq$ 400Kb/s) 3: 高速模式( $\leq$ 3.4Mb/s)
位0	<b>MASTER_MODE:</b> 该位控制主机模式 (This bit controls whether the DW_apb_i2c master is enabled) 0: 主机禁止 1: 主机使能

IC\_SLAVE\_DISABLE (bit6) 和 MASTER\_MODE (bit0) 配置如下表所列

表 35. IC\_SLAVE\_DISABLE(bit6)和MASTER\_MODE(bit 0)配置

IC_SLAVE_DISABLE (IC_CON[6])	MASTER_MODE IC_CON[0]	状态
0	0	从机器件
0	1	配置错误
1	0	配置错误
1	1	主机器件

### 18.7.2 I2C 目标地址寄存器 (IC\_TAR)

偏移地址: 0x04

复位值: 0x0055

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	SPEC IAL	GC_O R_ST ART													IC_TAR[9: 0]
		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

位31: 12	保留, 始终读为0
位11	<b>SPECIAL:</b> 该位指示软件执行的是否是特殊命令(广播呼叫或者起始字节命令) (This bit indicates whether software performs a General Call or START BYTE command) 0: 忽略位10 GC_OR_START, 正常使用 IC_TAR 位 1: 执行特殊 I2C 命令如 GC_OR_START 位描述
位10	<b>GC_OR_START:</b> 如果位11置位, 该位显示I2C执行的是广播呼叫还是起始字节 (If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the DW_apb_i2c) 0: 广播呼叫地址。发送广播呼叫地址时只能执行写操作。I2C接口一直工作在广播地址模式下直到SPECIAL(bit11)的值被清零。 1: 起始字节命令
位9: 0	<b>IC_TAR:</b> 主操作的目标地址 (This is the target address for any master transaction) 当发送一个广播地址, 这些位就可以忽略。 要产生开始字节的命令, CPU只需要对这些位写一次。

### 18.7.3 I2C 从机地址寄存器 (IC\_SAR)

偏移地址: 0x08

复位值: 0x55

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

位31: 12	保留, 始终读为0
位9: 0	<b>IC_SAR:</b> 当I2C接口工作在从机模式下, 这些存储从机地址 (The IC_SAR holds the slave address when the I2C is operating as a slave) 对于7位的地址格式, 只有 IC_SAR[6: 0]有效。

### 18.7.4 I2C 高速模式主机码地址寄存器 (IC\_HS\_MADDR)

偏移地址: 0x0C

复位值: 0x1

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

位31: 3	保留, 始终读为0
位2: 0	<b>IC_HS_MAR:</b> I2C 高速模式下主机码 (This bit field holds the value of the I2C HS mode master code) HS模式的主机码保留为8位 (00001xxx), 该主机码不是供从机寻址或者其他作用。每个主机都有自己独立的主机码。多达8个I2C高速模式下的主机可以挂接同一个I2C总线上。有效值为0~7。 当I2C接口工作在标准 (1) 或快速 (2) 模式下, 读该位返回为0。

### 18.7.5 I2C 数据命令寄存器 (IC\_DATA\_CMD)

偏移地址: 0x10

复位值: 0x1

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

位31: 9	保留, 始终读为0
--------	-----------

位8	<b>CMD:</b> 控制在主模式下执行读或写操作 (This bit controls whether a read or a write is performed) 1: 读 0: 写 当一个命令进入TX FIFO, 该位用于区分读写命令。在从接收模式下, 该位写值操作被忽略。 在从发送模式下, 写0表示IC_DATA_CMD寄存器的数据准备发送。
位7: 0	<b>DAT:</b> I2C总线待发送或者接收到的数据 (This register contains the data to be transmitted or received on the I2C bus)

#### 18.7.6 标准模式 I2C 时钟高电平计数寄存器(IC\_SS\_SCL\_HCNT)

偏移地址: 0x14

复位值: 0x190

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

位31: 16	保留, 始终读为0
位15: 0	<b>IC_SS_SCL_HCNT:</b> I2C接口标准模式下SCL时钟高电平周期 (This register sets the SCL clock high-period count for standard speed) 注意: 该寄存器可配置值在 6 和 65525 之间, 这是由于I2C接口使用了一个16位的计时器, 该计数器值等于IC_SS_SCL_HCNT+10时标志I2C总线处于空闲状态。

#### 18.7.7 标准模式 I2C 时钟低电平计数寄存器(IC\_SS\_SCL\_LCNT)

偏移地址: 0x18

复位值: 0x1D6

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

位31: 16	保留, 始终读为0
位15: 0	<b>IC_SS_SCL_LCNT:</b> I2C接口标准模式下SCL时钟低电平周期 (This register sets the SCL clock low period count for standard speed) 最小值为8.

### 18.7.8 快速模式 I2C 时钟高电平计数寄存器 (IC\_FS\_SCL\_HCNT)

偏移地址: 0x1C

复位值: 0x036

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

位31: 16	保留, 始终读为0
位15: 0	<b>IC_FS_SCL_HCNT:</b> I2C接口快速模式下SCL时钟高电平周期 (This register sets the SCL clock high-period count for fast mode or fast mode plus) 用于高速模式下发送主机码和起始字节或者广播地址。 当I2C工作在标准模式下该寄存器为只读且返回值为0。 最小值为6。

### 18.7.9 快速模式 I2C 时钟低电平计数寄存器 (IC\_FS\_SCL\_LCNT)

偏移地址: 0x20

复位值: 0x082

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

位31: 16	保留, 始终读为0
位15: 0	<b>IC_FS_SCL_LCNT:</b> I2C接口快速模式下SCL时钟低电平周期 (This register sets the SCL clock low period count for fast mode or fast mode plus) 同时用于高速模式下发送主机码或起始字节或广播地址。 当I2C工作在标准模式下该寄存器为只读且返回值为0。 最小值为8。

### 18.7.10 高速模式 I2C 时钟高电平计数寄存器 (IC\_HS\_SCL\_HCNT)

偏移地址: 0x24

复位值: 0x006

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

位31: 16	保留, 始终读为0
---------	-----------

位15: 0	<p><b>IC_HS_SCL_HCNT:</b> I2C接口高速模式下SCL时钟高电平周期 (This register sets the SCL clock high period count for high speed)</p> <p>SCL高电平时序与总线的负荷有关。例如100pF负荷，SCL高电平持续时间为60ns。对于400pF负荷，SCL高电平持续时间为120ns。</p> <p>当I2C工作在高速模式以外该寄存器为只读且返回值为0。</p> <p>最小值为6。</p>
--------	---

### 18.7.11 高速模式 I2C 时钟低电平计数寄存器(IC\_HS\_SCL\_LCNT)

偏移地址: 0x28

复位值: 0x010

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

位31: 16	保留, 始终读为0
位15: 0	<p><b>IC_HS_SCL_LCNT:</b> I2C接口高速模式下SCL时钟低电平周期 (This register sets the SCL clock low period count for high speed)</p> <p>SCL低电平时序与总线的负荷有关。例如100pF负荷，SCL低电平持续时间为160ns.对于400pF负荷，SCL低电平持续时间为320ns。</p> <p>当I2C工作在高速模式以外该寄存器为只读且返回值为0。</p> <p>最小值为8。</p>

### 18.7.12 I2C 中断状态寄存器(IC\_INTR\_STAT)

偏移地址: 0x2C

复位值: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留	D	R_RE STAR T_DE T	R_GE N_CA LL	R_ST ART_ DET	R_ST OP_D	R_AC TIVIT Y	R_RX _DON E	R_TX _ABRT Q	R_R D_RE Y	R_TX _EMPT Y	R_TX _OVE R	R_RX _OVE L	R_RX _FUL R	R_RX _UND ER	r	r

位31: 16	保留, 始终读为0
位15: 0	具体每位描述可以参考IC_RAW_INTR_STAT寄存器 (See "IC_RAW_INTR_STAT" for a detailed description of these bits)

### 18.7.13 I2C 中断屏蔽寄存器(IC\_INTR\_MASK)

偏移地址: 0x30

复位值: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	M_GE N_CA LL	M_ST ART_D ET	M_ST OP_D ET	M_AC TIVIT Y	M_RX _DON E	M_TX _ABR T	M_R D_RE Q	M_TX _EMP TY	M_TX _OVE R	M_RX _FUL L	M_RX _OVE R	M_RX _UND ER			
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 12	保留, 始终读为0
位11: 0	每一位屏蔽IC_INTR_STAT对应位。(These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register)

### 18.7.14 I2C RAW 中断寄存器(IC\_RAW\_INTR\_STAT)

偏移地址: 0x34

复位值: 0x000

IC\_RAW\_INTR\_STAT 与 IC\_INTR\_STAT 寄存器的区别在于前者不会被屏蔽。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	GEN_CALL	STAR_T_DET	STOP_DET	ACTIVITY	RX DONE	TX_ABRT	RD_EQ	TX_EMPTY	TX_OVERFLOW	RX_FULL	RX_OVERFLOW	RX_UNDER			
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

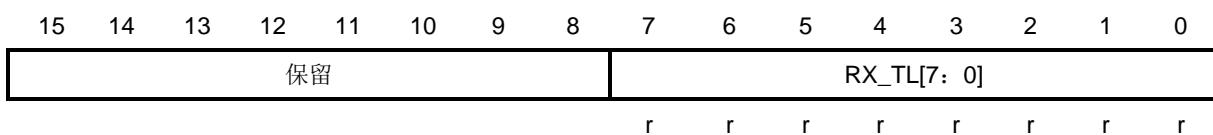
位31: 16	保留, 始终读为0
位11	<b>GEN_CALL:</b> 广播呼叫 (General call) 接收到广播呼叫地址时置位。 禁止I2C接口或者当CPU读IC_CLR_GEN_CALL寄存器时清零。I2C将接收的数据存储在接收缓冲中。
位10	<b>START_DET:</b> 起始条件检测 (Start condition detection) 无论I2C接口工作在主机或者从机, 一旦检测到I2C接口上起始或者重复起始条件即置位该位
位9	<b>STOP_DET:</b> 停止条件检测 (Stop condition detection) 该位状态依据IC_CON寄存器的STOP_DET_IFADDRESSED的状态 当STOP_DET_IFADDRESSED=0 无论I2C接口工作在主机或者从机, 一旦检测到I2C接口上停止条件时即置位该位。从机模式下, 无论寻址是否匹配都会产生一个STOP_DET中断 当STOP_DET_IFADDRESSED=1 在主机模式下(MASTER_MODE=1), 该位显示I2C接口是否发生停止条件 在从机模式下(MASTER_MODE=0), 仅当从机地址匹配成功时产生一个STOP_DET中断。

位8	<p><b>ACTIVITY:</b> I2C接口激活，该位用于捕捉I2C模块的活动状态 (This bit captures DW_apb_i2c activity and stays set until it is cleared)</p> <p>置位后只能由以下四种方式清零：</p> <ul style="list-style-type: none"> <li>禁止I2C接口</li> <li>读IC_CLR_ACTIVITY寄存器</li> <li>读IC_CLR_INTR寄存器</li> <li>系统复位</li> </ul> <p>一旦置位后，只能由上述方式清零，即使I2C处于空闲状态，该位也仍然保持为高直到被清零。</p>
位7	<p><b>RX_DONE:</b> 从发送结束 (Transmit done)</p> <p>当I2C作为从发送时，如果发送一个字节的数据后主机没有响应，将会置位该位。 该情况发生在传输的最后一个字节，表示传输结束。</p>
位6	<p><b>TX_ABRT:</b> 发送中止 (Transmit abort)</p> <p>当I2C接口作为发送机时，不能发送完缓冲中的数据时置位。</p> <p>注意：发送中止会将I2C接口中的接收和发送缓冲清空。发送缓冲会处于刷新状态直到读IC_CLR_TX_ABRT寄存器。一旦该读操作执行后，发送就可以接收APB总线上的新的数据。</p>
位5	<p><b>RD_REQ:</b> 读请求 (Read request)</p> <p>当I2C作为从机，其他主机试图从I2C接口读取数据时置位。</p> <p>I2C接口会使总线保持等待状态 (SCL=0) 直到中断被处理。这就意味着I2C接口作为从机时被其他主机寻址成功且要求发送数据。处理器必须响应该中断然后写入数据到IC_DATA_CMD寄存器中。当处理器读IC_CLR_RD_REQ寄存器该位清零。</p>
位4	<p><b>TX_EMPTY:</b> 发送缓冲空 (Transmit buffer empty)</p> <p>该位状态取决于IC_CON寄存器中的TX_EMPTY_CTRL状态：</p> <ul style="list-style-type: none"> <li>当TX_EMPTY_CTRL=0，发送缓冲为空时置位</li> <li>当TX_EMPTY_CTRL=1，发送缓冲为空且内部移位寄存器结束时置位</li> <li>当发送缓冲非空时由硬件自动清零。</li> </ul>
位3	<p><b>TX_OVER:</b> 发送缓冲过载 (Transmit buffer over)</p> <p>发送缓冲满时处理器写入新的数据导致溢出时置位。</p>
位2	<p><b>RX_FULL:</b> 接收缓冲非空 (Receive buffer not empty)</p> <p>当接收缓冲非空时置位。</p> <p>当接收缓冲为空时由硬件清零。</p>
位1	<p><b>RX_OVER:</b> 接收缓冲过载 (Receive buffer over)</p> <p>接收缓冲满且有收到新的数据时置位。此时I2C接口会响应，但新的数据会丢失。</p>
位0	<p><b>RX_UNDER:</b> 接收缓冲欠载 (Receive buffer under)</p> <p>当RX FIFO 为空时处理器读IC_DATA_CMD寄存器时置位。</p>

### 18.7.15 I2C 接收阈值(IC\_RX\_TL)

偏移地址: 0x38

复位值: 0x000

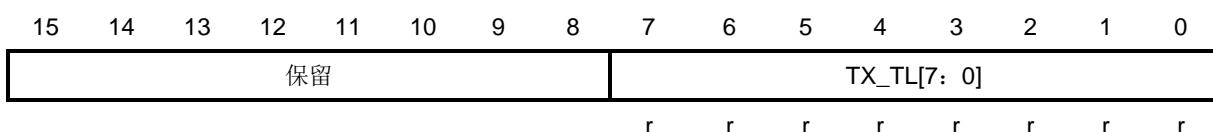


位31: 8	保留, 始终读为0
位7: 0	<b>RX_TL[7: 0]:</b> 接收FIFO阈值 (Receive FIFO threshold level) 控制RX_FULL中断触发。

### 18.7.16 I2C 发送阈值(IC\_TX\_TL)

偏移地址: 0x3C

复位值: 0x000



位31: 8	保留, 始终读为0
位7: 0	<b>TX_TL[7: 0]:</b> 发送FIFO阈值 (Transmit FIFO threshold level) 控制TX_EMPTY中断触发。

### 18.7.17 I2C 组合和独立中断清除寄存器(IC\_CLR\_INTR)

偏移地址: 0x40

复位值: 0x000



位31: 1	保留, 始终读为0
位0	<b>CLR_INTR:</b> 读该寄存器将会清除所有组合中断、独立中断 (Read this register to clear the combined interrupt, all individual interrupts) 该位不清除硬件可自动清除的中断, 仅清除软件可清除中断。

**18.7.18 I2C 清除 RX\_UNDER 中断寄存器(IC\_CLR\_RX\_UNDER)**

偏移地址: 0x44

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_RX_UNDER:</b> 读该寄存器清零RX_UNDER中断 (IC_RAW_INTR_STAT[0]) (Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register)

**18.7.19 I2C 清除 RX\_OVER 中断寄存器(IC\_CLR\_RX\_OVER)**

偏移地址: 0x48

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_RX_OVER:</b> 读该寄存器清零RX_OVER中断 (IC_RAW_INTR_STAT[1]) (Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register)

**18.7.20 I2C 清除 TX\_OVER 中断寄存器(IC\_CLR\_TX\_OVER)**

偏移地址: 0x4C

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
--------	-----------

位0	<b>CLR_TX_OVER:</b> 读该寄存器清零TX_OVER中断(IC_RAW_INTR_STAT[3]) (Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register)
----	--

### 18.7.21 I2C 清除 RD\_REQ 中断寄存器(IC\_CLR\_RD\_REQ)

偏移地址: 0x50

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_RD_REQ:</b> 读该寄存器清零RD_REQ中断 (IC_RAW_INTR_STAT[5]) (Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register)

### 18.7.22 I2C 清除 TX\_ABRT 中断寄存器(IC\_CLR\_TX\_ABRT)

偏移地址: 0x54

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_TX_ABRT:</b> 读该寄存器清零TX_ABRT中断(IC_RAW_INTR_STAT[6]) (Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register) 同时也将TX FIFO从刷新/复位状态中释放, 以便接收写入的数据。

### 18.7.23 I2C 清除 RX\_DONE 中断寄存器(IC\_CLR\_RX\_DONE)

偏移地址: 0x58

复位值: 0x000

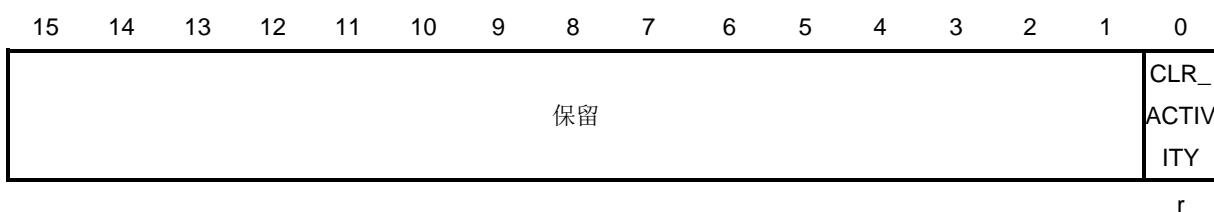


位31: 1	保留, 始终读为0
位0	<b>CLR_RX_DONE:</b> 读该寄存器清零RX_DONE中断(IC_RAW_INTR_STAT[7]) (Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register)

### 18.7.24 I2C 清除 ACTIVITY 中断寄存器(IC\_CLR\_ACTIVITY)

偏移地址: 0x5C

复位值: 0x000



位31: 1	保留, 始终读为0
位0	<b>CLR_ACTIVITY:</b> 如果I2C总线不活动则读该寄存器清零 ACTIVITY 中断 (IC_RAW_INTR_STAT[8]) (Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore) 如果I2C仍然活动, 那么ACTIVITY中断将继续置位。当I2C模块禁止或者在I2C总线不再活动时该位由硬件清零。可以通过读该寄存器得到 IC_RAW_INTR_STAT 中的 ACTIVITY (bit 8) 的状态。

**18.7.25 I2C 清除 STOP\_DET 中断寄存器 (IC\_CLR\_STOP\_DET)**

偏移地址: 0x60

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_STOP_DET:</b> 读该寄存器清零STOP_DET中断(IC_RAW_INTR_STAT[9]) (Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register)

**18.7.26 I2C 清除 START\_DET 中断寄存器 (IC\_CLR\_START\_DET)**

偏移地址: 0x64

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
位0	<b>CLR_START_DET:</b> 读该寄存器清零START_DET中断 (IC_RAW_INTR_STAT[10]) (Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register)

**18.7.27 I2C 清除 GEN\_CALL 中断寄存器 (IC\_CLR\_GEN\_CALL)**

偏移地址: 0x68

复位值: 0x000

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

r

位31: 1	保留, 始终读为0
--------	-----------

位0	<b>CLR_GEN_CALL:</b> 读该寄存器清零GEN_CALL中断(IC_RAW_INTR_STAT[11]) (Read this register to clear the GEN_CALL interrupt (bit 11) of IC_RAW_INTR_STAT register)
----	---

### 18.7.28 I2C 使能寄存器 (IC\_ENABLE)

偏移地址: 0x6C

复位值: 0x000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														ABORT	ENABLE
														T	LE
														rw	rw

位31: 1	保留, 始终读为0
位1	<b>ABORT:</b> I2C传输中止(I2C transfer abort) 0: 中止没有发生或者已经结束 1: 中止操作正在进行 I2C模块作为主机时置位时可以软件中止I2C的传输。一旦置位不能立即清除。置位后I2C模块控制逻辑会在完成当前传输之后产生一个STOP条件和清空发送缓冲, 中止操作之后产生TX_ABRT中断。 该ABRT位会在中止操作结束后自动清零。
位0	<b>ENABLE:</b> I2C模块使能 (I2C mode enable) 0: 禁止I2C模块 (发送和接收缓冲保持擦除状态) 1: 使能I2C模块

### 18.7.29 I2C 状态寄存器 (IC\_STATUS)

偏移地址: 0x70

复位值: 0x006

该寄存器只读, 指示当前传输和缓冲状态, 状态位不产生中断。

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

位31: 7	保留, 始终读为0
位6	<b>SLV_ACTIVITY:</b> 从机状态机活动状态位 (Slave FSM activity status) 0: 从机状态机处于IDLE状态, 所以I2C从机部分不活动 1: 从机状态机不处于IDLE状态, 所以I2C从机部分活动

位5	<b>MST_ACTIVITY:</b> 主机状态机活动状态位 (Master FSM activity status) 0: 主机状态机处于IDLE状态，所以I2C主机部分不活动 1: 主机状态机不处于IDLE状态，所以I2C主机部分活动
位4	<b>RFF:</b> 接收缓冲满 (Receive FIFO completely full) 0: 接收缓冲未满 1: 接收缓冲满
位3	<b>RFNE:</b> 接收缓冲非空 (Receive FIFO not empty) 0: 接收缓冲空 1: 接收缓冲非空
位2	<b>TFE:</b> 发送缓冲空 (Transmit FIFO completely empty) 0: 发送缓冲非空 1: 发送缓冲空
位1	<b>TFNF:</b> 发送缓冲未满 (Transmit FIFO not full) 0: 发送缓冲满 1: 发送缓冲未满
位0	<b>ACTIVITY:</b> I2C位活动状态 (I2C activity status) MST_ACTIVITY位与SLV_ACTIVITY位相或的结果。

### 18.7.30 I2C 发送缓冲水平寄存器 (IC\_TXFLR)

偏移地址: 0x74

复位值: 0x006

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

位31: 2	保留, 始终读为0
位1: 0	<b>TXFLR:</b> 发送缓冲中有效数据个数(0~2) (Transmit FIFO level.Contains the number of valid data entries in the transmit FIFO)

### 18.7.31 I2C 接收缓冲水平寄存器 (IC\_RXFLR)

偏移地址: 0x78

复位值: 0x006

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

位31: 2	保留, 始终读为0
--------	-----------

位1: 0	<b>RXFLR:</b> 接收缓冲中有效数据个数(0~2) (Receive FIFO level. Contains the number of valid data entries in the receive FIFO)
-------	--

### 18.7.32 I2C SDA 保持时间寄存器(IC\_SDA\_HOLD)

偏移地址: 0x7C

复位值: 0x0001 0001

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

位31: 24	保留, 始终读为0
位23: 16	<b>IC_SDA_RX_HOLD:</b> 当I2C器件作为接收时, SDA保持时间, 单位为APB1系统时钟周期 (Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a receiver)
位15: 0	<b>IC_SDA_TX_HOLD:</b> 当I2C器件作为发送时, SDA保持时间, 单位为APB1系统时钟周期 (Sets the required SDA hold time in units of ic_clk period, when DW_apb_i2c acts as a transmitter)

### 18.7.33 I2C DMA 控制寄存器(IC\_DMA\_CR)

偏移地址: 0x88

复位值: 0x0000

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

位31: 2	保留, 始终读为0
位1	<b>TDMAE:</b> 发送DMA使能 (Transmit DMA enable) 0: 发送DMA禁止 1: 发送DMA使能
位0	<b>RDMAE:</b> 接收DMA使能 (Receive DMA enable) 0: 接收DMA禁止 1: 接收DMA使能

**18.7.34 I2C SDA 建立时间寄存器(IC\_SDA\_SETUP)**

偏移地址: 0x94

复位值: 0x0064

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

位31: 8	保留, 始终读为0
位7: 0	<b>SDA_SETUP:</b> SDA建立时间 (SDA setup) 如果有要求建议延迟时间为1000ns, APB1时钟频率为10MHZ时, 建议该寄存器设为11。该寄存器最小值为2

**18.7.35 I2C 广播呼叫 ACK 寄存器(IC\_ACK\_GENERAL\_CALL)**

偏移地址: 0x98

复位值: 0x0001

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

位31: 1	保留, 始终读为0
位0	<b>ACK_GEN_CALL:</b> 广播呼叫ACK (ACK general call) 1: 接收到广播呼叫后响应ACK。 0: 接收到广播呼叫后不响应, 也不产生中断。

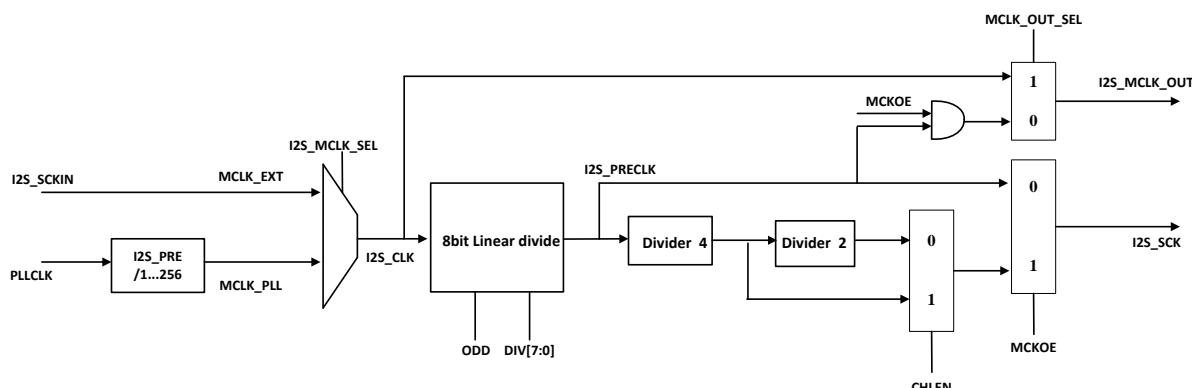
## 19. I2S

### 19.1 I2S 特性

- AMBA 2.0 APB 总线接口
- 只支持主模式操作，提供 SCK 和 WS，发送或接收 SD
- 每个通道的采样率可配置为 16bit 或 24bit，单个通道长度只支持 32bit
- 支持音频 stereo（立体声）或 mono（单声道）的发送和接收
- 支持连续或非连续的发送和接收
- 支持左,右对齐
- 单声道模式下只需选择并配置一个通道
- SD,WS 的锁存可配置为在 SCK 的上沿或下沿
- 数据格式可在延时（Philips mode）和非延时（non-Philips mode）之间切换
- FIFO 数据空或满可触发中断
- 内嵌的 TX\_FIFO 和 RX\_FIFO 是 32x16bit

### 19.2 I2S 功能

#### 19.2.1 I2S clock generator



#### 19.2.2 I2S data format

当 data format 为 0，采样率为 16bit 时的数据格式如下：

(1) Mono data, for one channel

31	16	15	0
Second 16bit mono channel data		First 16bit mono channel data	
Fourth 16bit mono channel data		Third 16bit mono channel data	

(2) Stereo data, for two channel

31	16	15	0
First 16bit channel2 data		First 16bit channel1 data	
Second 16bit channel2 data		Second 16bit channel1 data	

当 data format 为 1, 采样率为 16bit 时的数据格式如下:

(1) Mono data, for one channel

31                    16    15                    0

16'h0	First 16bit mono channel data	
16'h0	Second 16bit mono channel data	

(2) Stereo data, for two channel

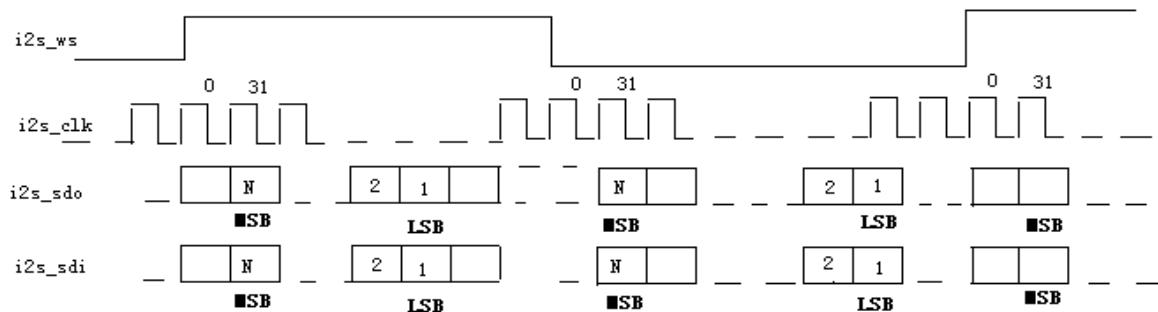
31                    16    15                    0

16'h0	First 16bit channel1 data	
16'h0	First 16bit channel2 data	
16'h0	Second 16bit channel1 data	
16'h0	Second 16bit channel2 data	

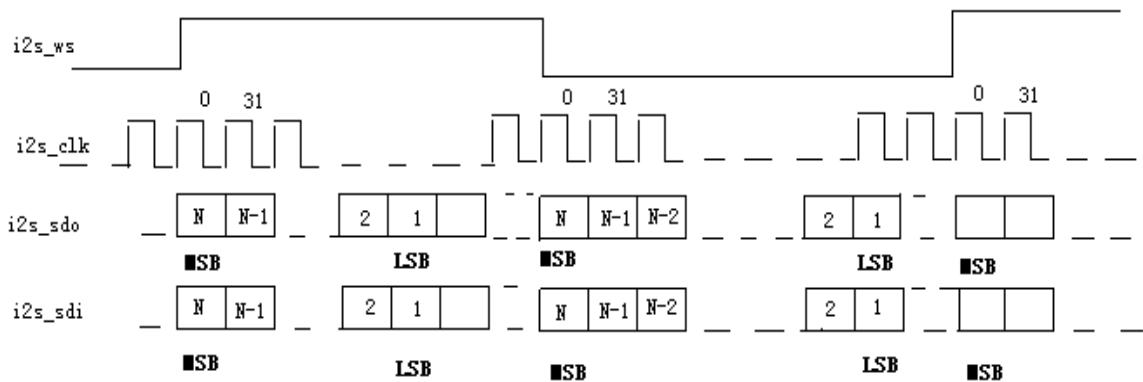
Note: 采样率为 24bit 时, 数据格式与 16bit 类似

### 19.2.3 I2S timing information

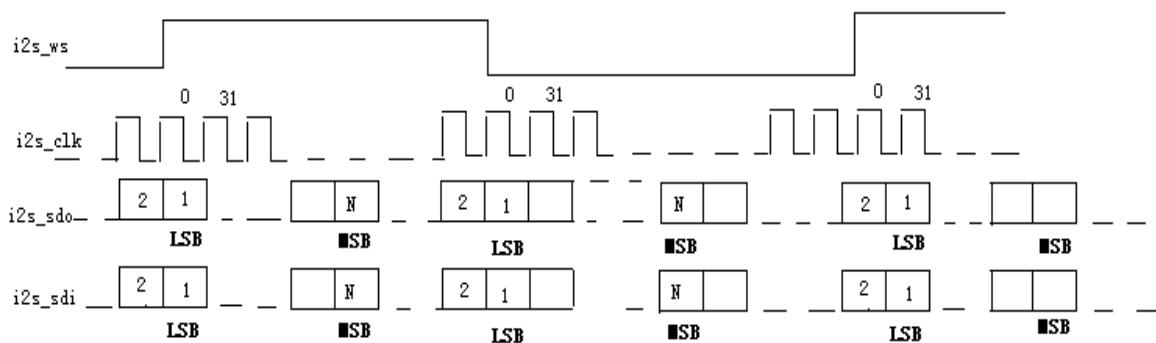
(1) I2S bus with delay mode, left alignment (lr\_align=0)      Note: N is 16 or 24.



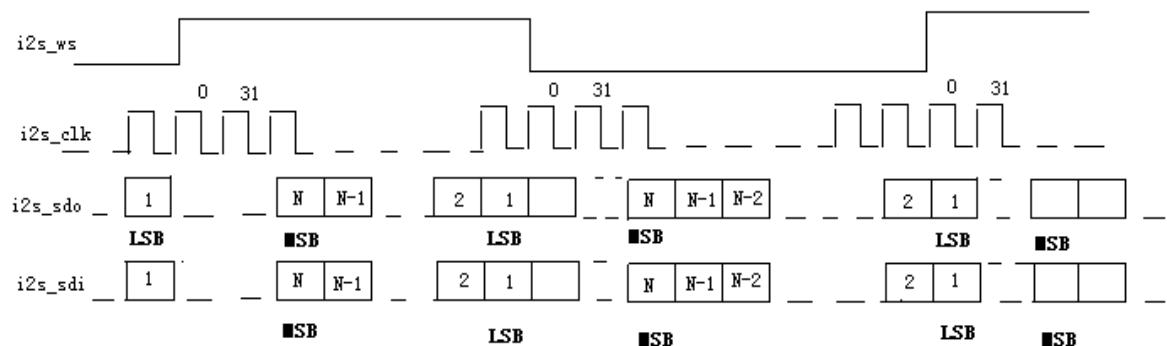
(2) I2S bus without delay mode, left alignment (lr\_align=0) Note: N is 16 or 24.



(3) I2S bus with delay mode, right alignment (lr\_align=1) Note: N is 16 or 24.



(4) I2S bus without delay mode, right alignment (lr\_align=1) Note: N is 16 or 24.



## 19.3 I2S 寄存器描述:

### 19.3.1 I2S 写数据寄存器 (I2S\_WR)

寄存器描述: I2S Write Data Register

地址偏移: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2S_WR															
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
I2S_WR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0	I2S_WR: 写数据寄存器 (Write Data to I2S)
--------	------------------------------------

### 19.3.2 I2S 接收使能寄存器 (I2S\_RD)

寄存器描述: I2S Read Data Register

地址偏移: 0x4

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2S_RD															
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
I2S_RD															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位31: 0	I2S_RD: 读数据寄存器 (Read Data from I2S)
--------	-------------------------------------

### 19.3.3 I2S 状态寄存器 (I2S\_CSR)

寄存器描述: I2S Current Status Register

地址偏移: 0x8

复位值: 0x0000 0001

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															

位31: 3	保留
位2	TXFULL: 传输FIFO满状态位 1: 传输FIFO满 0: 传输FIFO未满
位1	RXAVL: 接收到有效数据状态位 1: 接收FIFO中存在有效数据 0: 接收FIFO空
位0	TXEPT: 传输空状态位 1: 传输FIFO空 0: 传输FIFO非空

### 19.3.4 I2S 全局控制寄存器 (I2S\_GCR)

寄存器描述: I2S Global Control Register

地址偏移: 0xC

复位值: 0x0001 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留														nFs_sel	
rw rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
txfifo_flush	rxfifo_flush	i2sen	dma mode	int_en	fill datas	txen	rxen	保留	txtlf	保留	rxtlf				
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw		

位31: 18	保留
位17: 16	<p>nFs_sel: Fs与MCLK配置            00: MCLK=128Fs            01: MCLK=256Fs            10: MCLK=512Fs            11: MCLK=1024Fs</p> <p>Note: SCK=64Fs, 只有当clk_bypass=1时, 此配置才有用</p>
位15	<p>txfifo_flush: txfifo刷新            1: 刷新txfifo            0: txfifo正常工作</p>
位14	<p>rxfifo_flush: rx FIFO刷新            1: 刷新rx FIFO            0: rx FIFO正常工作</p>
位13	<p>i2sen: I2S使能            1: 使能I2S            0: 禁止I2S</p>
位12	<p>dmamode: DMA模式选择            1: 使能DMA模式            0: 禁止DMA模式</p>
位11	<p>int_en: 中断使能            1: 使能I2S中断            0: 禁止I2S中断</p>
位10	<p>filldatas: 当传输欠载时填充数据选择            1: 传输先前的数据            0: 传输全0</p>
位9	<p>txen: TXFIFO使能            1: 使能TXFIFO            0: 禁止TXFIFO</p>

位8	rxen: RXFIFO使能 1: 使能RXFIFO 0: 禁止RXFIFO
位7: 6	保留
位5: 4	txtlf: TXFIFO触发深度 01: TXFIFO中小于等于8个words有效数据 other: TXFIF未满 当使能DMA时, txtlf设为01
位3: 2	保留
位1: 0	rxtlf: RXFIFO触发深度 01: RXFIFO中大于等于8个words有效数据 other: RXFIF非空 当使能DMA时, rxtlf设为01

### 19.3.5 I2S 数据格式寄存器 (I2S\_DFR)

寄存器描述: I2S Data Format Register

地址偏移: 0x10

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								data_format	lr_align	width	ch_switch	delay_m	edgem	wssel	stereo
								rw	rw	rw	rw	rw	rw	rw	rw

位31: 9	保留
位8	data_format: I2S传输或接收的数据格式 1: 存储器中是非连续的有效音频数据。当width=0时, 一个word的低16位数据是有效的, 当width=1时, 一个word的低24位数据是有效的, 其他位补0. 0: 存储器中是连续的有效音频数据 Note: 具体数据合适见1.22章节
位7	lr_align: 对齐方式 1: 右对齐 0: 左对齐
位6	width: 采样率选择 1: 24bit 0: 16bit

位5: 4	ch_switch: 传输或接收通道选择 00: 无通道有效 01: 第一个通道有效 10: 第二个通道有效 11: 双通道有效 Note: 无效的通道传输或接收全0
位3	delaym: 延时模式选择 1: SD与WS同时变化 (Non-Philips mode) 0: SD相较于WS延时一个SCK (Philips mode)
位2	edgem: 边沿模式选择 1: SD,WS在SCK上升沿变化 0: SD,WS在SCK下降沿变化
位1	wssel: WS极性选择 1: 当WS为高时传输第一个数据 0: 当WS为低时传输第一个数据
位0	stereo: 立体声,单声道数据传输或接收 1: 选择立体声 0: 选择单声道

### 19.3.6 I2S 中断状态寄存器 (I2S\_ISR)

寄存器描述 I2S Interrupt Status Register

地址偏移: 0x14

复位值: 0x0000 0000

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

位31: 4	保留
位3	underrun_intf: 传输欠载错误中断标志 1: 欠载错误 0: 无欠载错误
位2	rxoerr_intf: 接收过载错误中断标志 1: 过载错误 0: 无过载错误

位1	rx_intf: 接收有效数据中断标志 1: 当rxtlf=01, RXFIFO已接收8个数据; 当rxtlf=other, RXFIFO非空 0: 当rxtlf=01, RXFIFO未接收8个数据; 当rxtlf=other, RXFIFO空
位0	tx_intf: TXFOFO空中断标志 1: 当txtlf=01, TXFIFO中的数据少于8个; 当txtlf=other, TXFIFO未满 0: 当txtlf=01, TXFIFO中的数据大于8个; 当txtlf=other, TXFIFO满

### 19.3.7 I2S 中断使能寄存器 (I2S\_IER)

寄存器描述: I2S Interrupt Enable Register

地址偏移: 0x18

复位值: 0x0000 0000

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

位31: 4	保留
位3	underrun_inten: 传输欠载错误中断使能 1: 欠载错误中断使能 0: 欠载错误中断禁止
位2	rxoerr_inten: 接收过载错误中断使能 1: 过载错误中断使能 0: 过载错误中断禁止
位1	rx_inten: 接收有效数据中断使能 1: 接收有效数据中断使能 0: 接收有效数据中断禁止
位0	tx_inten: TXFOFO空中断使能 1: TXFIFO空中断使能 0: TXFIFO空中断禁止

### 19.3.8 I2S 中断清除寄存器 (I2S\_ICR)

寄存器描述: I2S Interrupt Clear Register

地址偏移: 0x1C

复位值: 0x0000 0000

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

位31: 4	保留
位3	underrun_intclr: 传输欠载错误中断清除 1: 欠载错误中断清除 0: 欠载错误中断不清除
位2	rxoerr_intclr: 接收过载错误中断清除 1: 过载错误中断清除 0: 过载错误中断不清除
位1	rx_intclr: 接收有效数据中断清除 1: 接收有效数据中断清除 0: 接收有效数据中断不清除
位0	tx_intclr: TXFOFO空中断清除 1: TXFIFO空中断清除 0: TXFIFO空中断不清除

### 19.3.9 I2S 分频寄存器 (I2S\_PRE)

寄存器描述: I2S Prescale Register

地址偏移: 0x24

复位值: 0x0000 0C02

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

位12	mclk_out_sel: master clock输出选择 1: master clock输出为I2S_CLK 0: master clock输出为I2S_PRECLK
位11	SCK bypass 1: SCK来自Linear Prescaler分频 0: SCK=64Fs, 通过设置nFs_sel配置为I2S_CLK的2/4/8/16分频
位10	chlen: 通道长度选择 1: 32bit 0: 16bit 只支持32bit
位9	mckoe: master clock输出使能 1: master clock输出使能 0: master clock输出禁止
位8	odd: I2S odd分频系数 1: 分频系数为 <div><math>div \times 2 + 1</math></div> 0: 分频系数为 <div><math>div \times 2</math></div> 此位需在I2S禁止时配置
位7: 0	div: I2S div分频系数 实际分频系数为 <div><math>div \times 2 + odd</math></div> div[7:0]不能配置为0或1, 否则无时钟输出 此位需在I2S禁止时配置

## 20. 串行外设接口（SPI）

### 20.1 SPI 简述

SPI 接口广泛用于不同设备之间的板级通讯，如微处理器，ADC 等。事实上目前 SPI 已经成为整个行业可接收的指导方针。许多 IC 制造商生产的器件都兼容 SPI。

SPI 允许 MCU 与外部设备以全双工、同步、串行方式通信。应用软件可以通过查询状态或 SPI 中断来通信。

### 20.2 主要特征

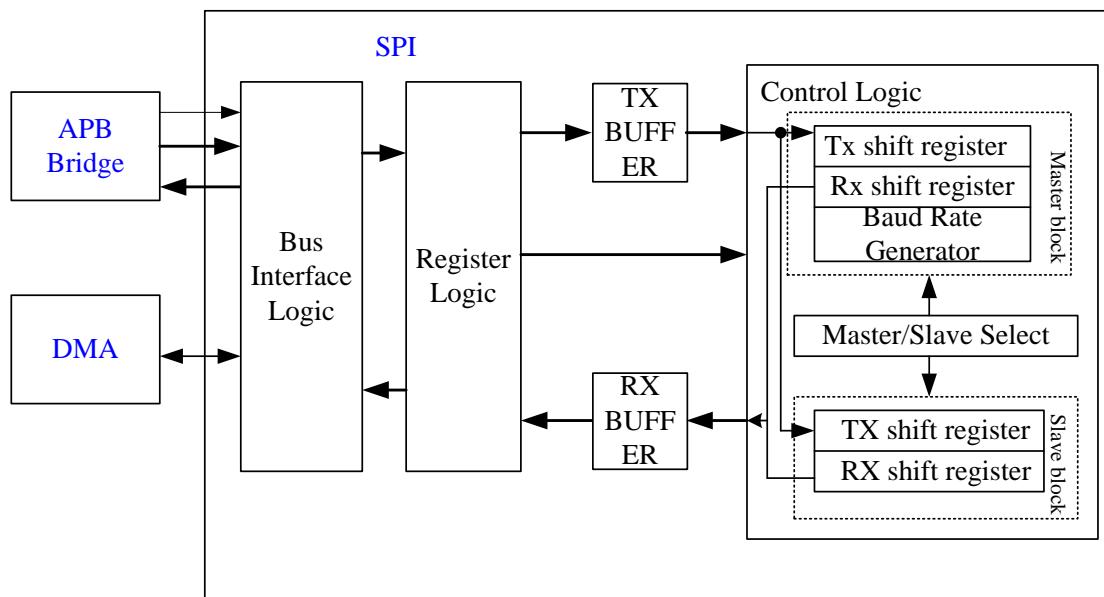
- 完全兼容 Motorola 的 SPI 规格
- 支持 DMA 请求
- 全双工同步传输
- 16 位的可编程波特率生成器
- 支持主机模式和从机模式
- SPI 作为主机模式下 SPI 的时钟最快可高达  $\text{pclk}/2$ （ $\text{pclk}$  为 APB 时钟），作为从机模式下 SPI 的时钟最快可高达  $\text{pclk}/4$ .
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或者 LSB 在前
- 支持一个主机多个从机操作
- 支持发送和接收 7 位或者 8 位的数据同时进行
- 支持 9 位及 8 位可配置长度的数据传输
- 支持主模式下硬件自动重复发送同一个数据
- 具有各 8 字节的发送缓冲器和接收缓冲器
- 中断驱动操作
  - 发送端空，发送端溢出
  - 接收的数据有效，接收端的数据溢出
  - 在 SPI 主模式完整接收，发送端为空
  - 硬件自动重复发送结束。

### 20.3 SPI 功能描述

#### 20.3.1 概述

SPI 的方框图见下图

图 162. SPI 框图



SPI 支持接收和发送 7 或者 8 位数据同时进行，并且支持可配置 4-16bit 数据长度的发送与接收。SPI 可以被配置为从模式或者在一个主机环境下配置为主模式。可以通过配置时钟极性 CPOL 和相位 CPHA 选择四种可能的时序关系。可编程的数据顺序，MSB 在前或者 LSB 在前。

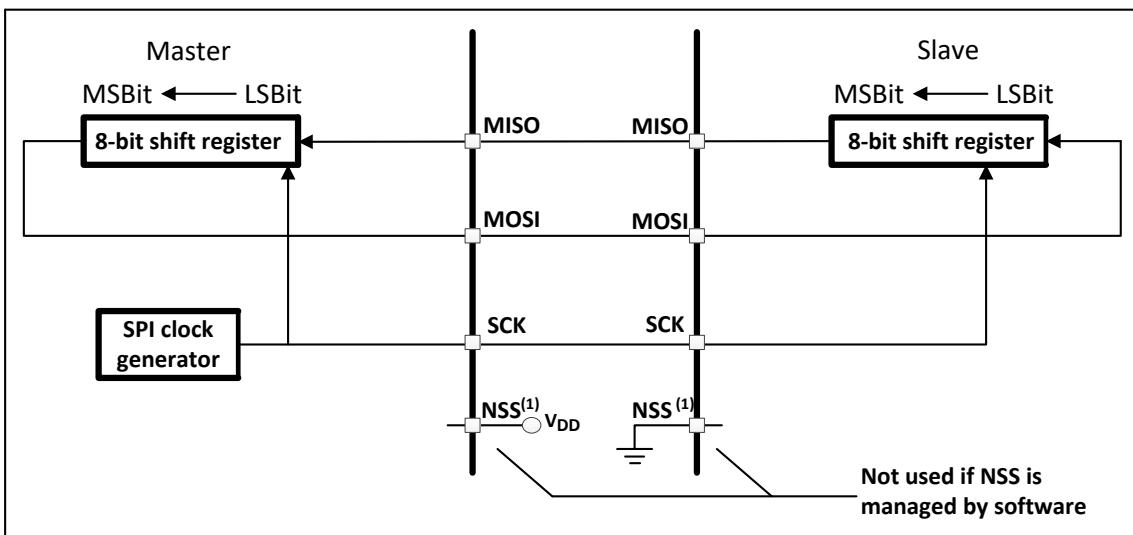
发送和接收部分使用相同的时钟。数据在时钟的上升沿或者下降沿输出，在 SCLK 相反的有效沿锁存数据。因为 SPI 是用于交换数据，因此数据必须在转移结束后读取，即使数据不是有效数据。在 SPI 模式下，主机和与其通信的从机的时钟相位和极性必须相同。

通常 SPI 通过 4 个管脚与外部器件相连：

- **MISO:** 主设备输入/从设备输出管脚。该管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/从设备输入管脚。该管脚在主模式下发送数据，在从模式下接收数据。
- **SCK:** 串口时钟，作为主设备的输出，从设备的输入
- **NSS:** 从设备选择。这是一个可选的管脚，用来选择主/从设备。它的功能是用来作为‘片选管脚’，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 管脚可以由主设备当作一个标准的 IO 来驱动。一旦被使能，NSS 管脚也可以作为输出管脚，并在 SPI 设置为主模式时拉低；此时，所有 NSS 管脚连接到主设备 NSS 管脚的 SPI 设备，会检测到低电平。

下图是一个单主和单从设备互连的例子。

图 163. 单主和单从应用



MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输（MSB 位在前）。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味着全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

### 数据发送寄存器

SPI 定义了两个数据发送寄存器：32bit 的 SPI\_TXREGA 和 16bit 的 SPI\_TXREGB（SPI\_TXREGB 分为两个寄存器定义：8bit 的 SPI\_TXREGBL 和 8bit 的 SPI\_TXREGH）。

当数据写入 SPI\_TXREGA 时，数据直接发送；当数据单独写入 SPI\_TXREGBH 或者 SPI\_TXREGBL 数据都不发送，只有先写入 SPI\_TXREGH 寄存器再写入 SPI\_TXREGBL 寄存器之后，一起发送 16 位的数据。

注：在 SPI\_GCTL 中定义了 TXREG\_SEL 位，用来选择此时使用的是 SPI\_TXREGA 寄存器还是 SPI\_TXREGB 寄存器。

### 硬件自动重复发送数据

SPI 在 SPI\_CCTL 寄存器中定义了硬件自动传输数据控制位 SER\_TRANF\_EN，开启使能后，在此写入发送寄存器的数据将会被硬件自动传输 n 次，传输次数是由 SPI\_TX\_NUM 寄存器定义的。

注：SPI\_TX\_NUM 中写入值为 i，则传输 i-1 次。

### 时钟信号的相位和极性

SPI\_CCTL 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL（时钟极性）位控制在没有数据传输时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清‘0’，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置‘1’，SCK 引脚在空闲状态保持高电平。

如果 CPHA（时钟相位）位被置‘1’，SCK 时钟的第二个边沿（CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿）进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清‘0’，SCK 时钟的第一边沿（CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿）进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。图 164 显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

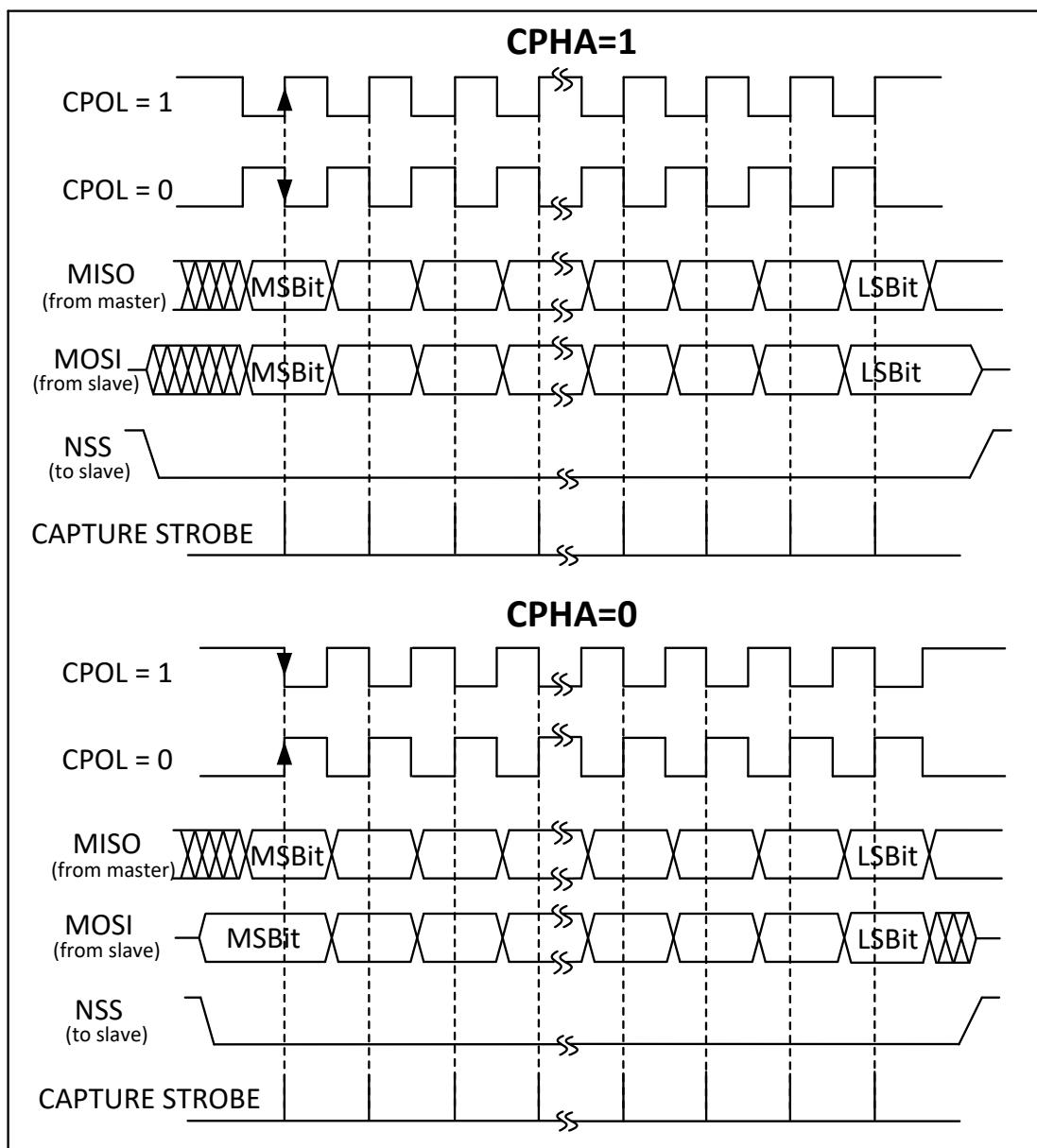
### 高速传输

针对高速传输模式下对板级延时的敏感，在 SPI\_CCTL 寄存器中由 TXEDGE 和 RXEDGE 控制位对发送相位和接收采样进行时间调整。

- 在从模式下，TXEDGE 为 1 时，发送数据立即发送到数据总线，用于高速模式时（SPBRG = 4）；为 0 时，发送数据在一个有效时钟边沿后发送到数据总线，用于低速模式时（SPBRG > 4）。
- 在主模式下，RXEDGE 为 1 时，在传输数据位的中间采样数据；为 0 时，在传输数据位的尾时钟沿采样数据（用于高速模式）。

注：1. 在改变 CPOL/CPHA 位之前，必须清除 SPIEN 位将 SPI 禁止。  
2. 主和从必须配置成相同的时序模式。  
3. SCK 的空闲状态必须和 SPI\_CCTL 寄存器指定的极性一致（CPOL 为 1 时，空闲时应上拉 SCK 为高电平；CPOL 为 0 时，空闲时应下拉 SCK 为低电平）。

图 164. 数据时钟时序图



### 数据帧格式

根据 SPI\_CCTL 寄存器中的 LSBFE 位，输出数据位时可以 MSB 在先也可以 LSB 在先。根据 SPI\_GCTL 寄存器的 DATA\_SEL、SPI\_CCTL 寄存器的 SPILEN 位以及 SPI\_EXTLEN 寄存器，每个数据帧可以是 4 位到 16 位之间任意一位。所选择的数据帧格式对发送或接收都有效。

#### 20.3.2 SPI 从模式

在从配置里，SCK 引脚用于接收到从主设备来的串行时钟。SPI\_SPBRG 寄存器中的设置不影响数据传输速率。

##### 配置步骤

1. 设置 SPILEN 位、DATA\_DEL 位以及 SPI\_EXTLEN 位以定义数据帧格式。

2. 选择 CPOL 和 CPHA 位来定义数据传输和串行时钟之间的相位关系。为保证正确的数据传输，从设备和主设备的 CPOL 和 CPHA 位必须配置成相同的方式。
3. 帧格式（MSB 在前还是 LSB 在前取决于 SPI\_CCTL 寄存器中的 LSBFE 位）必须和主设备相同。
4. 清除 MM 位，设置 SPIEN 位，使相应引脚工作于 SPI 模式下。在这个配置里，MOSI 引脚是数据输入，MISO 引脚是数据输出。

#### 数据发送过程

在写操作中，数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始，第一个位被发送出去。余下的位被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPI\_INTSTAT 寄存器里的 TX\_INTF 标志被设置。如果设置了 SPI\_INTEN 寄存器上的 TXIEN 位，将会产生中断。

#### 数据接收过程

对于接收方，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI\_INTSTAT 寄存器中的 RX\_INTF 标志被设置。
- 如果设置了 SPI\_INTEN 寄存器中的 RXIEN 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置‘1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI\_RXREG 寄存器时，SPI 设备返回这个值。

### 20.3.3 SPI 主模式

在主配置时，串行时钟在 SCK 脚产生。

#### 配置步骤

1. 通过 SPI\_SPBRG 寄存器定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位，定义数据传输和串行时钟间的相位关系。
3. 设置 SPILEN 位、DATA\_SEL 位以及 SPI\_EXTLEN 位以定义数据帧格式。
4. 配置 SPI\_CCTL 寄存器的 LSBFE 位定义帧格式。
5. 如果只接收而不发送数据，配置 SPI\_RNDNR 寄存器，定义需要接收的字节数。
6. 必须设置 MM 和 SPIEN 位。

在这个配置中，MOSI 脚是数据输出，而 MISO 脚是数据输入，NSS 是从设备选择信号输出。

#### 数据发送过程

当一字节写进发送缓冲器时，发送过程开始。在发送第一个数据位时，数据字被并行地（通过内部总线）传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI\_CCTL 寄存器中的 LSBFE 位。数据从发送缓冲器传输到移位寄存器时 TX\_INTF 标志将被置位，如果设置 SPI\_INTEN 寄存器中的 TXIEN 位，将产生中断。

#### 数据接收过程

对于接收器来说，当数据传输完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI\_INTSTAT 寄存器中的 RX\_INTF 标志被设置。
- 如果设置了 SPI\_INTEN 寄存器中的 RXIEN 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置‘1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI\_RXREG 寄存器时，SPI 设备返回这个值。

如果只接收而不发送数据，在接收完 RXDNR 定义的字节数，RXMATCH\_INTF 位被置‘1’，表示所有的数据接收完毕，主模式下不再发送时钟信号。

#### 20.3.4 状态标志

为了软件操作的方便，应用程序可以通过 4 个当前状态标志和 7 个中断状态标志来监控 SPI 总线的状态。当前状态标志是只读，由硬件自动置位和清除。中断状态标志位在事件发生时置位，并在中断使能时产生 CPU 中断，由软件清除。

SPI 内部分别有一个 8 字节的发送缓冲和接收缓冲，根据 SPI\_GCTL 的 DATA\_SEL 位的设置，CPU 每次可以读写 1 或者 4 个字节。根据 DATA\_SEL 的设置，发送和接收缓冲分别有一个字节或者一个有效数据的状态标志。

表 36. SPI 状态

分类	状态标志	缓冲器和信号状态
中断状态	TX_INTF	根据 DATA_SEL 设置，至少有一个有效数据的空间，能完成一次发送数据寄存器的写操作
	RX_INTF	根据 DATA_SEL 设置，至少有一个有效数据的数据，能完成一次接收数据寄存器的读操作
	UNDERRUN_INTF	发送缓冲器空且重复发送
	RXOERR_INTF	接收缓冲器非空且被覆盖
	RXMATCH_INTF	非空，最后一个数据传送到接收缓冲中
	RXFULL_INTF	接收缓冲器满，不能再接收新的数据
	TXEPT_INTF	发送缓冲器空，不能再发送
当前状态	SER_TRANF_INTF	重复发送结束，不再发送
	SER_TRANF_END	没有重复发送动作
	RXAVL_4BYTE	接收缓冲器有超过 4 字节有效数据
	TXFULL	发送缓冲器满
	TXEPT	发送缓冲器空
	RXAVL	接收缓冲器非空，至少还能接收一个字节

注：当 SPI\_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时 TX\_INTF 置位；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时 TX\_INTF 置位。

当 SPI\_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时，RX\_INTF 置位；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时 RX\_INTF 置位。

#### 20.3.5 波特率设置

波特率是生成的 SCLK 的频率，一般是 PCLK 的分频。BRG 是一个 16 位的波特率发生器。SPBREG 寄存器控制 16 位计数器的计数周期。

提供期望的波特率和  $f_{\text{pclk}}$  (APB 模块的频率)，使用下表所示的公式计算出的值近似数赋值给 SPBRG 寄存器。其中下表中的 X 等于 SPBRG 寄存器的值 (2 ~ 65535)。

表 37. 波特率公式

模式	公式
SPI 模式	波特率 = $f_{\text{pclk}} / X$

### 20.3.6 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI\_GCTL 寄存器上的 DMAEN 位被设置时，SPI 模块可以发出 DMA 传输数据的请求。发送缓冲器和接收缓冲器的 DMA 请求都由 DMAEN 使能。

- 发送时，当 SPI\_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时即进行 DMA 传输请求；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及发送缓冲器每个数据大小由 DATA\_SEL 为决定。
- 接收时，当 SPI\_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时即进行 DMA 传输请求；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及接收缓冲器每个数据大小由 DATA\_SEL 为决定。

## 20.4 寄存器堆和存储器映射描述

### 20.4.1 发送数据寄存器 (SPI\_TXREGA)

偏移地址：0x00

复位值：0x0000 0000

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

位 31: 0	<b>TXREG:</b> 发送数据寄存器 (Transmit data register)
	有效数据位由 data_sel 控制。 0: 只有低 8 位有效 1: TXREG[31: 0]都有效

#### 20.4.2 发送数据寄存器(SPI\_TXREGBL)

偏移地址: 0x04

复位值: 0x0000 0000

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

位31: 8	保留
位7: 0	<b>TXREG:</b> 发送数据寄存器B低8位 (Transmit data register)

#### 20.4.3 发送数据寄存器(SPI\_TXREGBH)

偏移地址: 0x08

复位值: 0x0000 0000

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

位31: 8	保留
位7: 0	<b>TXREG:</b> 发送数据寄存器B高8位 (Transmit data register)

#### 20.4.4 接收数据寄存器 (SPI\_RXREG)

偏移地址: 0x0C

复位值: 0x0000 0000

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

位 31: 0	<b>RXREG:</b> 接收数据寄存器 (Receive data register) 有效数据位由 <b>data_sel</b> 控制。 0: 只有低 8 位有效 1: RXREG[31: 0]都有效 该寄存器可读不可写。
---------	---

#### 20.4.5 当前状态寄存器 (SPI\_CSTAT)

偏移地址: 0x10

复位值: 0x0000 0001

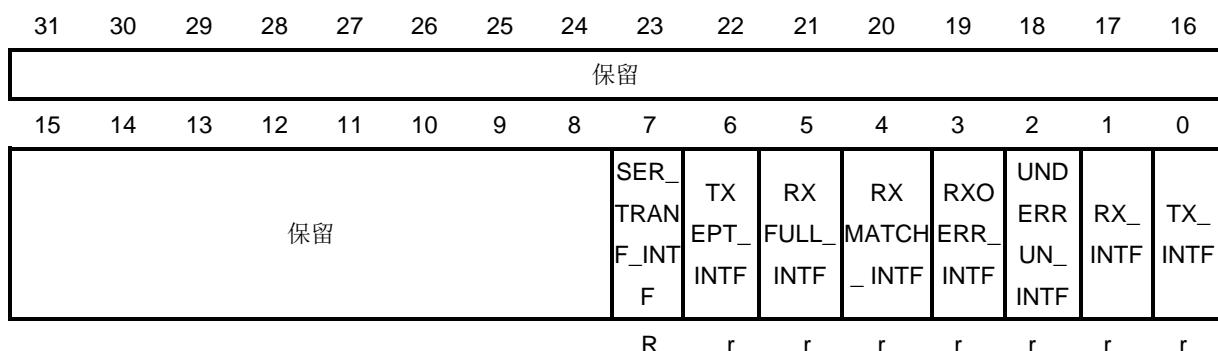
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留												RX AVL_ 4BYTE	TX FULL	RX AVL	TX EPT
r r r r															

位 31: 4	保留。
位 3	<b>RXAVL_4BYTE:</b> 接收缓冲器中有效数据达到 4 个字节标志位 (Receive available 4 byte data message) 1 = 接收缓冲器中有超过 4 个字节 0 = 接收缓冲器中数据小于 4 个字节
位 2	<b>TXFULL:</b> 发送缓冲器满标志位 (Transmitter FIFO full status bit) 1 = 发送缓冲器满 0 = 发送缓冲器未满
位 1	<b>RXAVL:</b> 接收有效字节数据信息位 (Receive available byte data message) 当接收端缓冲器接收了一个完整字节的数据时置位该位。 1 = 接收端缓冲器已经接收了一个有效字节数据 0 = 接收端缓冲器空 该位只读, 由硬件自动置位和清除。
位 0	<b>TXEPT:</b> 发送端空位 (Transmitter empty bit) 1 = 发送端缓冲器和发送移位寄存器为空 0 = 发送端不为空 该位只读, 由硬件自动置位和清除。

#### 20.4.6 中断状态寄存器(SPI\_INTSTAT)

偏移地址: 0x14

复位值: 0x0000 0000



位31: 8	保留
位7	<b>SER_TRANF_INTF:</b> 连续传输数据结束中断标志位 (Serial transform over interrupt flag bit) 1: 连续传输数据结束 0: 连续传输进行中
位6	<b>TXEPT_INTF:</b> 发送端空中断标志位 (Transmitter empty interrupt flag bit) 硬件自动置位, 写INTCLR寄存器TXEPT_ICLR位清除 1=发送端缓冲器和TX 移位寄存器为空 0=发送端不为空; 注意: 该位是中断状态信号, TXEPT是状态信号
位5	<b>RXFULL_INTF:</b> 接收端缓冲器满中断标志位 (RX FIFO full interrupt flag bit) 硬件自动置位, 写INTCLR寄存器RXFULL_ICLR位清除 1=RX 缓冲器满 0=RX 缓冲器未满
位4	<b>RXMATCH_INTF:</b> 接收指定字节数中断标志位 (Receive data match the RXDNR number, the receive process will be completed and generate the interrupt) 硬件自动置位, 写INTCLR寄存器RXMATCH_ICLR位清除 1=接收了RXDNR寄存器指定的字节数 0=未完成RXDNR寄存器指定的字节数
位3	<b>RXOERR_INTF:</b> 接收端溢出错误中断标志位 (Receive overrun error interrupt flag bit) 硬件自动置位, 写INTCLR寄存器RXOERR_ICLR位清除 1=溢出错误 0=没有溢出错误
位2	<b>UNDERRUN_INTF :</b> SPI从机模式下溢标志位 (SPI underrun interrupt flag bit) 硬件自动置位, 写INTCLR寄存器UNDERRUN_ICLR位清除 1=下溢错误 0=没有下溢错误

位1	<b>RX_INTF:</b> 接收端数据有效中断标志位 (Receive data available interrupt flag bit) 硬件自动置位，写INTCLR寄存器RX_ICLR位清除 当接收端缓冲器接收了一个完整字节数据 1=接收端缓冲器有有效字节数据 0=接收端缓冲器空
位0	<b>TX_INTF :</b> 发送缓冲器有效中断标志位 (发送了一个字节的数据) (Transmit FIFO available interrupt flag bit) 硬件自动置位，写INTCLR寄存器TX_ICLR位清除 1=发送端缓冲器有效 0=发送端缓冲器无效

#### 20.4.7 中断使能寄存器(SPI\_INTEN)

偏移地址: 0x18

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								SER_TRANF_IEN	TX_EPT_IEN	RX_FULL_IEN	RX_MATCH_IEN	RXO_ERR_IEN	UNDERRUN_IEN	RX_IEN	TX_IEN
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留
位7	<b>SER_TRANF_IEN:</b> 连续传输数据结束中断标志位 (Serial transform over interrupt enable bit) 1: 中断使能 0: 禁止中断
位6	<b>TXEPT_IEN:</b> 发送端空中断使能位 (Transmit empty interrupt enable bit) 1=中断使能 0=禁止中断
位5	<b>RXFULL_IEN:</b> 接收端缓冲器满中断使能位 (Receive FIFO full interrupt enable bit) 1=中断使能 0=禁止中断
位4	<b>RXMATCH_IEN:</b> 接收指定字节数中断使能位 (Receive data complete interrupt enable bit) 1=中断使能 0=禁止中断
位3	<b>RXOERR_IEN:</b> 接收端溢出错误中断使能位 (Overrun error interrupt enable bit) 1=中断使能 0=禁止中断

位2	<b>UNDERRUN_IEN</b> : SPI从机模式下溢中断使能位(SPI 从机模式) (Transmitter underrun interrupt enable bit(SPI slave mode only)) 1=中断使能 0=禁止中断
位1	<b>RX_IEN</b> : 接收端数据中断使能位 (Receive FIFO interrupt enable bit) 1=中断使能 0=禁止中断
位0	<b>TX_IEN</b> : 发送缓冲器空中断使能位 (Transmit FIFO empty interrupt enable bit) 1=中断使能 0=禁止中断

#### 20.4.8 中断清除寄存器(SPI\_INTCLR)

偏移地址: 0x1C

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								SER_TRAN_F_ICL_R	TX_EPT_ICL_R	RX_FULL_ICL_R	RX_MATCH_ICL_R	RXO_ERR_ICL_R	UND_UN_ICL_R	RX_ICL_R	TX_ICL_R
W W W W W W W W															

位31: 8	保留
位7	<b>SER_TRANF_ICL_R</b> : 连续传输数据结束中断清除位 (Serial transform over interrupt clear bit) 1: 中断清除 0: 中断没有清除
位6	<b>TXEPT_ICL_R</b> : 发送端空中断清除位 (Transmitter empty interrupt clear bit) 1=中断清除 0=中断没有清除
位5	<b>RXFULL_ICL_R</b> : 接收端缓冲器满中断清除位 (Receiver buffer full interrupt clear bit) 1=中断清除 0=中断没有清除
位4	<b>RXMATCH_ICL_R</b> : 接收指定字节数中断清除位 (Receive completed interrupt clear bit) 1=中断清除 0=中断没有清除
位3	<b>RXOERR_ICL_R</b> : 接收端溢出错误中断清除位 (Overrun error interrupt clear bit) 1=中断清除 0=中断没有清除

位2	<b>UNDERRUN_ICLR</b> : SPI从机模式下溢中断清除位(SPI 从机模式) (Transmitter underrun interrupt clear bit(SPI slave mode only)) 1=中断清除 0=中断没有清除
位1	<b>RX_ICLR</b> : 接收端数据中断清除位 (Receive interrupt clear bit) 1=中断清除 0=中断没有清除
位0	<b>TX_ICLR</b> : 发送缓冲器空中断清除位 (Transmitter FIFO empty interrupt clear bit) 1=中断清除 0=中断没有清除

#### 20.4.9 全局控制寄存器(SPI\_GCTL)

偏移地址: 0x20

复位值: 0x0000 0004

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			TXRE G_SE L	DATA _SEL	NSS_ SEL	DMA EN	TXTLF		RXTLF	RXEN	TXEN	MM	INT_ EN	SPI EN	
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 13	保留
位12	<b>TXREG_SEL</b> : 发送数据寄存器选择 (TXREG select) 0: TXREGA 1: TXRGB
位11	<b>DATA_SEL</b> : 发送和接收数据寄存器有效数据选择 (Valid byte or double-word data select signal) 0: 只有低8位有效 1: 32位数据都有效 注: 不管是通过CPU还是DMA都必须用指定数据格式访问。
位10	<b>NSS_SEL</b> : 硬件或软件控制主模式下的NSS输出 (NSS select signal that from software or hardware) 0: 由NSSR寄存器值控制 1: 进行数据传输时硬件自动控制
位9	<b>DMAEN</b> : 接收和发送的DMA模式使能 (DMA access mode enable) 0: DMA模式禁止 1: DMA模式使能

位8: 7	<b>TXTLF:</b> 发送缓冲器触发DMA请求的边沿选择 (TX FIFO trigger level bit) 00: 发送缓冲器有大于等于1个空闲数据空间时即进行DMA请求或发送中断请求 01: 发送缓冲器有超过一半的空闲空间时即进行DMA请求或发送中断请求 1x: 保留 注: 当DATA_SEL为0时, 一个数据空间代表1个字节; 为1时, 一个数据空间代表4字节。
位6: 5	<b>RXTLF:</b> 接收缓冲器触发DMA请求的边沿选择 (RX FIFO trigger level bit) 00: 接收缓冲器有大于等于1个有效数据时即进行DMA请求或接收中断请求 01: 接收缓冲器有超过一半的有效数据时即进行DMA请求或接收中断请求 1x: 保留 注: 当DATA_SEL为0时, 一个有效数据代表1个字节; 为1时, 一个有效数据代表4字节。
位4	<b>RXEN:</b> 接收使能位 (Receive enable bit) 1=接收使能 0=接收禁止。同时可以清空RX 缓冲器 注意: 当SPI只工作在主机接收模式时, txen必须设置为0
位3	<b>TXEN:</b> 发送使能位 (Transmit enable bit) 1=发送使能 0=发送禁止。同时可以清空TX 缓冲器 注意: 当在主机模式下发送和接收同时发生
位2	<b>MM:</b> 主机模式位 (Master mode bit) 1=主机模式 (由内部BRG产生串行时钟) 0=从机模式 (串行时钟来自外部主机)
位1	<b>INT_EN :</b> SPI中断使能位 (SPI interrupt enable bit) 1=使能SPI中断 0=禁止SPI中断
位0	<b>SPIEN :</b> SPI选择位 (SPI select bit) 0=SPI禁止 (复位状态) 1=SPI使能

#### 20.4.10 通用控制寄存器(SPI\_CCTL)

偏移地址: 0x24

复位值: 0x0000 0008

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										TX EDGE	RX EDGE	SPI LEN	LSB FE	CPOL	CPHA	
RW      RW      RW      RW      RW      RW																

位31: 6	保留
--------	----

位5	<b>TXEDGE:</b> 发送数据相位调整位(从模式) (Transmit data edge select) 1=发送数据立即发送到数据总线，可用于高速模式时(SPBRG=4) 0=发送数据在一个有效时钟边沿后发送到数据总线，可用于低速模式时(SPBRG>4)
位4	<b>RXEDGE:</b> 接收数据采样时钟沿选择位(主模式) (Receive data edge select) 1=在传输数据位的尾时钟沿采样数据 (用于高速模式) 0=在传输数据位的中间采样数据
位3	<b>SPILEN :</b> SPI数据宽度位 (SPI character length bit) 1=8位数据(缺省) 0=7位数据
位2	<b>LSBFE :</b> LSB在前使能位 (LSI first enable bit) 1=数据传输或接收最低位在前 0=数据传输或接收最高位在前
位1	<b>CPOL:</b> 时钟极性标志位 (Clock polarity select bit) 1=时钟在空闲状态为高电平 0=时钟在空闲状态为低电平
位0	<b>CPHA:</b> 时钟相位选择位 (Clock phase select bit) 1=数据采样从第一个时钟边沿开始 0=数据采样从第二个时钟边沿开始

#### 20.4.11 波特率发生器 (SPI\_SPBRG)

偏移地址: 0x28

复位值: 0x0000 0002

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
SPBRG[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31: 16	保留
位 15: 0	<b>SPBRG :</b> SPI 波特率控制寄存器用于产生波特率 (SPI baud rate control register for baud rate) 波特率公式: 波特率 = $f_{\text{pclk}}/\text{SPBRG}$ ( $f_{\text{pclk}}$ 是 APB 时钟频率) 注意: 不要往该寄存器写 0 和 1。

#### 20.4.12 接收数据个数寄存器 (SPI\_RXDNR)

偏移地址: 0x2C

复位值: 0x0000 0001

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
RXDNR [15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31: 0

**RXDNR:** 该寄存器用于存储下次接收过程需要接收字节的个数 (The register is used to hold a count of to be received bytes in next receive process)

该寄存器的值在SPI为主机接收模式下有效。缺省值是1。

该寄存器值通过 MCU 写值改变。

注意: 不要往该寄存器写'0'值。

#### 20.4.13 从机片选寄存器 (SPI\_SCSR)

偏移地址: 0x30

复位值: 0x0000 00FF

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

位 15: 8

保留

位 7: 0

**CSN:** 主模式下片选输出信号。低有效, 从模式下该位无效 (Chip select output signal in Master mode)

0: 从器件被选中

1: 从器件未选中

#### 20.4.14 数据控制寄存器 (SPI\_EXTCTL)

偏移地址: 0x34

复位值: 0x0000 0008

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								EXTLEN[4:0]							
rw	rw	rw	rw	rw	rw	rw	rw								

位31: 5	保留
位4: 0	<p><b>EXTLEN:</b> 控制SPI数据长度</p> <p>0 0100: 4-bit      0 0101: 5-bit      0 0110: 6-bit      0 0111: 7-bit      0 1000: 8-bit      0 1001: 9-bit      0 1010: 10-bit      0 1011: 11-bit      0 1100: 12-bit      0 1101: 13-bit      0 1110: 14-bit      0 1111: 15-bit      1 0000: 16-bit</p>

#### 20.4.15 重复发送数据数量控制寄存器(SPI\_TX\_NUM)

偏移地址: 0x38

复位值: 0x0000 0001

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
TX_NUM[15:0]															

位31: 16	保留
位15: 0	<b>TX_NUM:</b> 重复发送数据数量

#### 20.4.16 传输模式寄存器(SPI\_TRANSF\_MODE)

偏移地址: 0x3C

复位值: 0x0000 0000

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

位31: 2	保留
位1: 0	<b>MODE_SEL:</b> 模式选择 00: Standard mode (标准模式) 01: Serial mode (连续模式)

## 21. 串行外设接口 (QSPI)

### 21.1 QSPI 简述

SPI 接口广泛用于不同设备之间的板级通讯，如微处理器，DAC，ADC 等。事实上目前 SPI 已经成为整个行业可接收的指导方针。许多 IC 制造商生产的器件都兼容 SPI。

SPI 允许 MCU 与外部设备以全双工、同步、串行方式通信。应用软件可以通过查询状态或 SPI 中断来通信。

### 21.2 主要特征

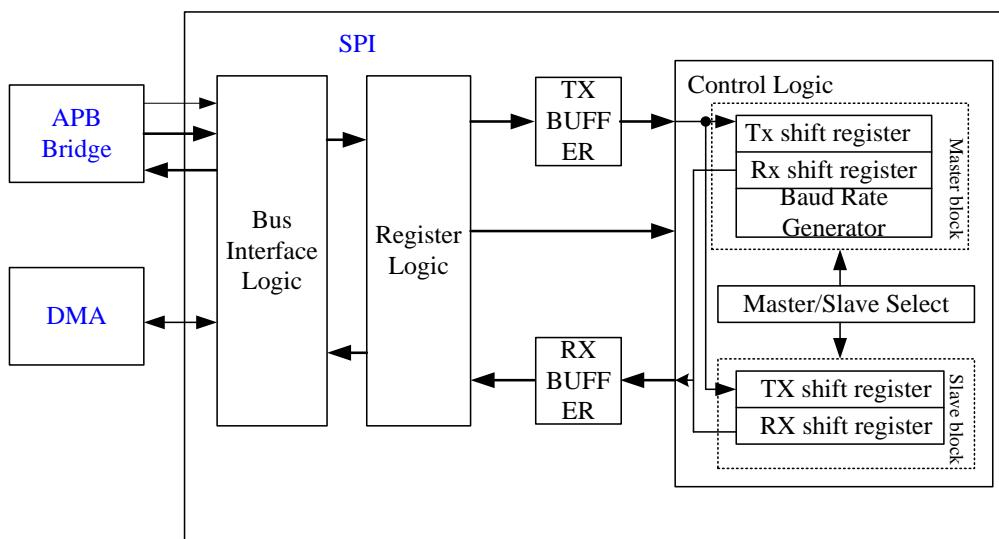
- 完全兼容 Motorola 的 SPI 规格
- 支持 DMA 请求
- 全双工同步传输
- 16 位的可编程波特率生成器
- 支持主机模式和从机模式
- SPI 作为主机模式下 SPI 的时钟最快可高达  $\text{pclk}/2$  ( $\text{pclk}$  为 APB 时钟)，作为从机模式下 SPI 的时钟最快可高达  $\text{pclk}/4$ .
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB 在前或者 LSB 在前
- 支持一个主机多个从机操作
- 具有各 8 字节的发送缓冲器和接收缓冲器
- 中断驱动操作
  - 发送端空，发送端溢出
  - 接收的数据有效，接收端的数据溢出
  - 在 SPI 主模式完整接收，发送端为空。

### 21.3 SPI 功能描述

#### 21.3.1 概述

SPI 的方框图见下图

图 165. SPI 框图



SPI 支持接收和发送 7 或者 8 位数据同时进行。SPI 可以被配置为从模式或者在一个主机环境下配置为主模式。可以通过配置时钟极性 CPOL 和相位 CPHA 选择四种可能的时序关系。可编程的数据顺序，MSB 在前或者 LSB 在前。

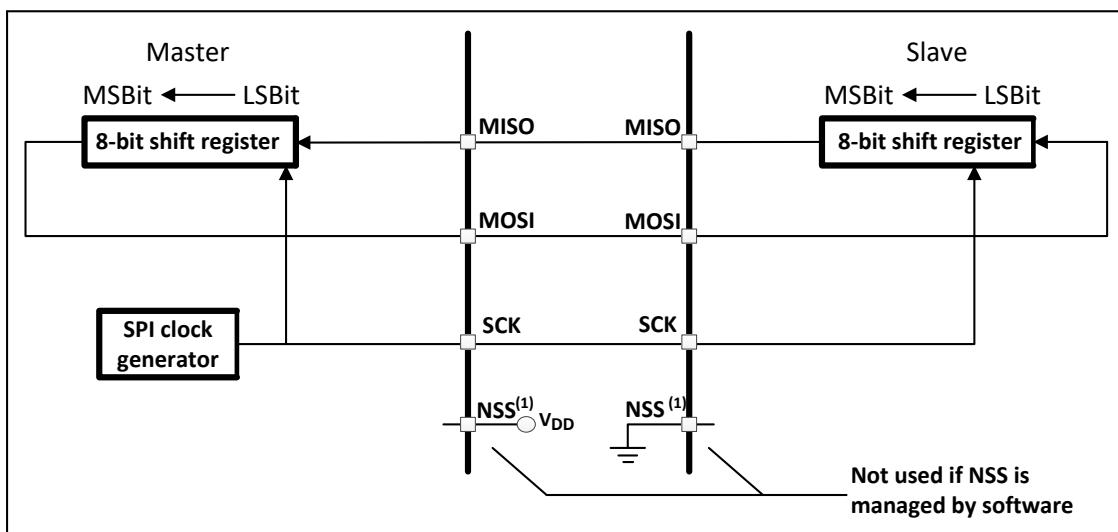
发送和接收部分使用相同的时钟。数据在时钟的上升沿或者下降沿输出，在 SCLK 相反的有效沿锁存数据。因为 SPI 是用于交换数据，因此数据必须在转移结束后读取，即使数据不是有效数据。在 SPI 模式下，主机和与其通信的从机的时钟相位和极性必须相同。

通常 SPI 通过 4 个管脚与外部器件相连：

- **MISO:** 主设备输入/从设备输出管脚。该管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/从设备输入管脚。该管脚在主模式下发送数据，在从模式下接收数据。
- **SCK:** 串口时钟，作为主设备的输出，从设备的输入
- **NSS:** 从设备选择。这是一个可选的管脚，用来选择主/从设备。它的功能是用来作为“片选管脚”，让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 管脚可以由主设备当作一个标准的 IO 来驱动。一旦被使能，NSS 管脚也可以作为输出管脚，并在 SPI 设置为主模式时拉低；此时，所有 NSS 管脚连接到主设备 NSS 管脚的 SPI 设备，会检测到低电平。

下图是一个单主和单从设备互连的例子。

图 166. 单主和单从应用



MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输 (MSB 位在前)。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传 数据。这意味着全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

#### 时钟信号的相位和极性

SPI\_CCTL 寄存器的 CPOL 和 CPHA 位，能够组合成四种可能的时序关系。CPOL (时钟极性) 位控制 在没有数据传输时时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清 ‘0’，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置 ‘1’，SCK 引脚在空闲状态保持高电平。

如果 CPHA (时钟相位) 位被置 ‘1’，SCK 时钟的第二个边沿 (CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿) 进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清 ‘0’，SCK 时钟的第一边沿 (CPOL 位为 0 时就是下降沿，CPOL 位为 1 时就是上升沿) 进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。图 164 显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

#### 高速传输

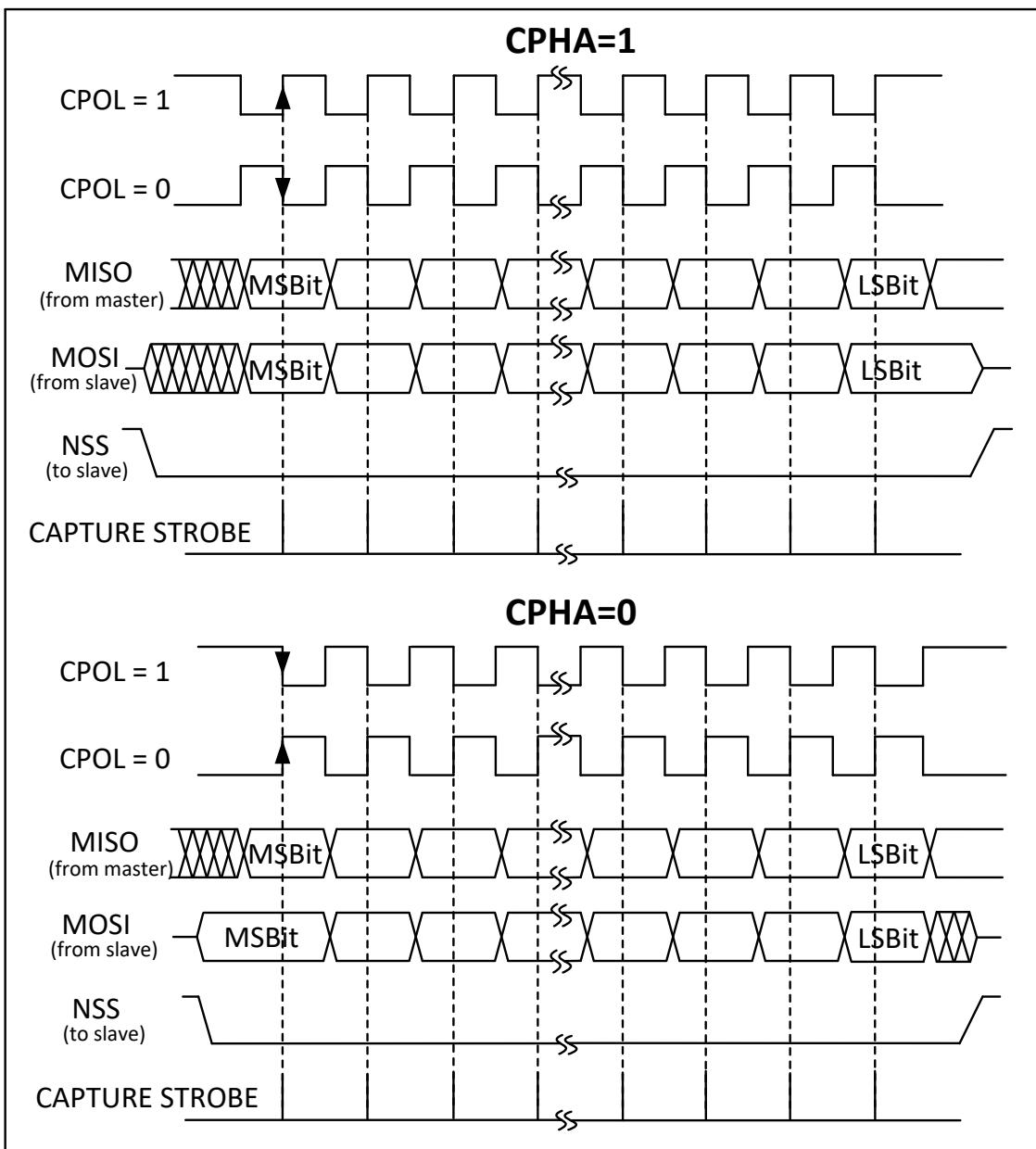
针对高速传输模式下对板级延时的敏感，在 SPI\_CCTL 寄存器中由 TXEDGE 和 RXEDGE 控制位对发送相位和接收采样进行时间调整。

- 在从模式下，TXEDGE 为 1 时，发送数据立即发送到数据总线，用于高速模式时 (SPBRG=4)；为 0 时，发送数据在一个有效时钟边沿后发送到数据总线，用于低速模式时 (SPBRG>4)。
- 在主模式下，RXEDGE 为 1 时，在传输数据位的中间采样数据；为 0 时，在传输数据位的尾时钟沿采样数据（用于高速模式）。

**注：**1. 在改变 CPOL/CPHA 位之前，必须清除 SPIEN 位将 SPI 禁止。  
2. 主和从必须配置成相同的时序模式。

3. SCK 的空闲状态必须和 SPI\_CCTL 寄存器指定的极性一致 (CPOL 为 1 时, 空闲时应上拉 SCK 为高电平; CPOL 为 0 时, 空闲时应下拉 SCK 为低电平)。

图 167. 数据时钟时序图



### 数据帧格式

根据 SPI\_CCTL 寄存器中的 LSBFE 位, 输出数据位时可以 MSB 在先也可以 LSB 在先。根据 SPI\_CCTL 寄存器的 SPILEN 位, 每个数据帧可以是 8 位或是 7 位。所选择的数据帧格式对发送和/或接收都有效。

#### 21.3.2 SPI 主模式

在主配置时, 串行时钟在 SCK 脚产生。

#### 配置步骤

1. 通过 SPI\_SPBRG 寄存器定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位, 定义数据传输和串行时钟间的相位关系。

3. 设置 SPILEN 位来定义 8 或 7 位数据帧格式。
4. 配置 SPI\_CCTL 寄存器的 LSBFE 位定义帧格式。
5. 如果只接收而不发送数据，配置 SPI\_RNDNR 寄存器，定义需要接收的字节数。
6. 必须设置 MM 和 SPIEN 位。

在这个配置中，MOSI 脚是数据输出，而 MISO 脚是数据输入，NSS 是从设备选择信号输出。

### 数据发送过程

当一字节写进发送缓冲器时，发送过程开始。在发送第一个数据位时，数据字被并行地（通过内部总线）传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPI\_CCTL 寄存器中的 LSBFE 位。数据从发送缓冲器传输到移位寄存器时 TX\_INTF 标志将被置位，如果设置 SPI\_INTEN 寄存器中的 TXIEN 位，将产生中断。

### 数据接收过程

对于接收器来说，当数据传输完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPI\_INTSTAT 寄存器中的 RX\_INTF 标志被设置。
- 如果设置了 SPI\_INTEN 寄存器中的 RXIEN 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置 ‘1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPI\_RXREG 寄存器时，SPI 设备返回这个值。

如果只接收而不发送数据，在接收完 RXDNR 定义的字节数，RXMATCH\_INTF 位被置 ‘1’，表示所有的数据接收完毕，主模式下不再发送时钟信号。

### 21.3.3 状态标志

为了软件操作的方便，应用程序可以通过 4 个当前状态标志和 7 个中断状态标志来监控 SPI 总线的状态。当前状态标志是只读，由硬件自动置位和清除。中断状态标志位在事件发生时置位，并在中断使能时产生 CPU 中断，由软件清除。

SPI 内部分别有一个 8 字节的发送缓冲和接收缓冲，根据 SPI\_GCTL 的 DATA\_SEL 位的设置，CPU 每次可以读写 1 或者 4 个字节。根据 DATA\_SEL 的设置，发送和接收缓冲分别有一个字节或者一个有效数据的状态标志。

表 38. SPI 状态

分类	状态标志	缓冲器和信号状态
中断状态	TX_INTF	根据DATA_SEL设置，至少有一个有效数据的空间，能完成一次发送数据寄存器的写操作
	RX_INTF	根据DATA_SEL设置，至少有一个有效数据的数据，能完成一次接收数据寄存器的读操作
	UNDERRUN_INTF	发送缓冲器空且重复发送
	RXOERR_INTF	接收缓冲器非空且被覆盖
	RXMATCH_INTF	非空，最后一个数据传送到接收缓冲中
	RXFULL_INTF	接收缓冲器满，不能再接收新的数据
	TXEPT_INTF	发送缓冲器空，不能再发送
当前状态	RXAVL_4BYTE	接收缓冲器有超过4字节有效数据
	TXFULL	发送缓冲器满
	TXEPT	发送缓冲器空
	RXAVL	接收缓冲器非空，至少还能接收一个字节

注：当 SPI\_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时 TX\_INTF 置位；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时 TX\_INTF 置位。

当 SPI\_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时，RX\_INTF 置位；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时 RX\_INTF 置位。

#### 21.3.4 波特率设置

波特率是生成的 SCLK 的频率，一般是 PCLK 的分频。BRG 是一个 16 位的波特率发生器。SPBREG 寄存器控制 16 位计数器的计数周期。

提供期望的波特率和  $f_{\text{pclk}}$  (APB 模块的频率)，使用下表所示的公式计算出的值近似数赋值给 SPBREG 寄存器。其中下表中的 X 等于 SPBREG 寄存器的值 (2~65535)。

表 39. 波特率公式

模式	公式
SPI模式	波特率 = $f_{\text{pclk}}/X$

#### 21.3.5 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPI\_GCTL 寄存器上的 DMAEN 位被设置时，SPI 模块可以发出 DMA 传输数据的请求。发送缓冲器和接收缓冲器的 DMA 请求都由 DMAEN 使能。

- 发送时，当 SPI\_GCTL 寄存器的 TXTLF 为 00 时，发送缓冲器有大于等于 1 个空闲数据空间时即进行 DMA 传输请求；TXTLF 为 01 时，发送缓冲器有超过一半的空闲空间时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及发送缓冲器每个数据大小由 DATA\_SEL 为决定。

- 接收时，当 SPI\_GCTL 寄存器的 RXTLF 为 00 时，接收缓冲器有大于等于 1 个有效数据时即进行 DMA 传输请求；RXTLF 为 01 时，接收缓冲器有超过一半的有效数据时即进行 DMA 请求。每次请求只进行一次 DMA 传输。每次 DMA 传输数据大小以及接收缓冲器每个数据大小由 DATA\_SEL 为决定。

## 21.4 寄存器堆和存储器映射描述

### 21.4.1 发送数据寄存器 (QSPI\_TXREG)

偏移地址: 0x00

复位值: 0x0000 0000

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

位31: 0	<b>TXREG:</b> 发送数据寄存器 (Transmit data register)
	有效数据位由data_sel控制 0: 只有低8位有效 1: TXREG[31: 0]都有效

### 21.4.2 接收数据寄存器 (QSPI\_RXREG)

偏移地址: 0x04

复位值: 0x0000 0000

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

位31: 0	<b>RXREG:</b> 接收数据寄存器 (Receive data register)
	有效数据位由data_sel控制 0: 只有低8位有效 1: RXREG[31: 0]都有效 该寄存器可读不可写。

### 21.4.3 当前状态寄存器 (QSPI\_CSTAT)

偏移地址: 0x08

复位值: 0x0000 0001



位31:	4	保留
位3		<b>RXAVL_4BYTE:</b> 接收缓冲器中有效数据达到4个字节标志位 (Receive available 4 byte data message) 1=接收缓冲器中有超过4个字节 0=接收缓冲器中数据小于4个字节
位2		<b>TXFULL:</b> 发送缓冲器满标志位 (Transmitter FIFO full status bit) 1=发送缓冲器满 0=发送缓冲器未满
位1		<b>RXAVL:</b> 接收有效字节数据信息位 (Receive available byte data message) 当接收端缓冲器接收了一个完整字节的数据时置位该位 1=接收端缓冲器已经接收了一个有效字节数据 0=接收端缓冲器空 该位只读，由硬件自动置位和清除
位0		<b>TXEPT:</b> 发送端空位 (Transmitter empty bit) 1=发送端缓冲器和发送移位寄存器为空。 0=发送端不为空 该位只读，由硬件自动置位和清除

### 21.4.4 中断状态寄存器 (QSPI\_INTSTAT)

偏移地址: 0x0C

复位值: 0x0000 0000



位31: 7	保留
位6	<p><b>TXEPT_INTF:</b> 发送端空中断标志位 (Transmitter empty interrupt flag bit) 硬件自动置位, 写INTCLR寄存器TXEPT_ICLR位清除 1=发送端缓冲器和TX 移位寄存器为空 0=发送端不为空; 注意: 该位是中断状态信号, TXEPT是状态信号</p>
位5	<p><b>RXFULL_INTF:</b> 接收端缓冲器满中断标志位 (RX FIFO full interrupt flag bit) 硬件自动置位, 写INTCLR寄存器RxFull_ICLR位清除 1=RX 缓冲器满 0=RX 缓冲器未满</p>
位4	<p><b>RXMATCH_INTF:</b> 接收指定字节数中断标志位 (Receive data match the RXDNR number, the receive process will be completed and generate the interrupt) 硬件自动置位, 写INTCLR寄存器RXMATCH_ICLR位清除 1=接收了RXDNR寄存器指定的字节数 0=未完成RXDNR寄存器指定的字节数</p>
位3	<p><b>RXOERR_INTF:</b> 接收端溢出错误中断标志位 (Receive overrun error interrupt flag bit) 硬件自动置位, 写INTCLR寄存器RXOERR_ICLR位清除 1=溢出错误 0=没有溢出错误</p>
位2	<p><b>UNDERRUN_INTF :</b> SPI从机模式下溢标志位 (SPI underrun interrupt flag bit) 硬件自动置位, 写INTCLR寄存器UNDERRUN_ICLR位清除 1=下溢错误 0=没有下溢错误</p>
位1	<p><b>RX_INTF:</b> 接收端数据有效中断标志位 (Receive data available interrupt flag bit) 硬件自动置位, 写INTCLR寄存器RX_ICLR位清除 当接收端缓冲器接收了一个完整字节数据 1=接收端缓冲器有有效字节数据 0=接收端缓冲器空</p>
位0	<p><b>TX_INTF :</b> 发送缓冲器有效中断标志位 (发送了一个字节的数据) (Transmit FIFO available interrupt flag bit) 硬件自动置位, 写INTCLR寄存器TX_ICLR位清除 1=发送端缓冲器有效 0=发送端缓冲器无效</p>

### 21.4.5 中断使能寄存器 (QSPI\_INTEN)

偏移地址: 0x10

复位值: 0x0000 0000

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

位31: 7	保留
位6	<b>TXEPT_IEN:</b> 发送端空中断使能位 (Transmit empty interrupt enable bit) 1=中断使能 0=禁止中断
位5	<b>RXFULL_IEN:</b> 接收端缓冲器满中断使能位 (Receive FIFO full interrupt enable bit) 1=中断使能 0=禁止中断
位4	<b>RXMATCH_IEN:</b> 接收指定字节数中断使能位 (Receive data complete interrupt enable bit) 1=中断使能 0=禁止中断
位3	<b>RXOERR_IEN:</b> 接收端溢出错误中断使能位 (Overrun error interrupt enable bit) 1=中断使能 0=禁止中断
位2	<b>UNDERRUN_IEN :</b> SPI从机模式下溢中断使能位(SPI 从机模式) (Transmitter underrun interrupt enable bit(SPI slave mode only)) 1=中断使能 0=禁止中断
位1	<b>RX_IEN:</b> 接收端数据中断使能位 (Receive FIFO interrupt enable bit) 1=中断使能 0=禁止中断
位0	<b>TX_IEN:</b> 发送缓冲器空中断使能位 (Transmit FIFO empty interrupt enable bit) 1=中断使能 0=禁止中断

### 21.4.6 中断清除寄存器 (QSPI\_INTCLR)

偏移地址: 0x14

复位值: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
									TXEPT _ICLR	RXFU LL_IC LR	RXMA TCH_I CLR	RXOE RR_I CLR	UNDE RRUN _ICLR	RX_IC LR	TX_IC LR
									w	w	w	w	w	w	w

位31: 7	保留
位6	<b>TXEPT_ICLR:</b> 发送端空中断清除位 (Transmitter empty interrupt clear bit) 1=中断清除 0=中断没有清除
位5	<b>RXFULL_ICLR:</b> 接收端缓冲器满中断清除位 (Receiver buffer full interrupt clear bit) 1=中断清除 0=中断没有清除
位4	<b>RXMATCH_ICLR:</b> 接收指定字节数中断清除位 (Receive completed interrupt clear bit) 1=中断清除 0=中断没有清除
位3	<b>RXOERR_ICLR:</b> 接收端溢出错误中断清除位 (Overrun error interrupt clear bit) 1=中断清除 0=中断没有清除
位2	<b>UNDERRUN_ICLR :</b> SPI从机模式下溢中断清除位(SPI 从机模式) (Transmitter underrun interrupt clear bit(SPI slave mode only)) 1=中断清除 0=中断没有清除
位1	<b>RX_ICLR:</b> 接收端数据中断清除位 (Receive interrupt clear bit) 1=中断清除 0=中断没有清除
位0	<b>TX_ICLR:</b> 发送缓冲器空中断清除位 (Transmitter FIFO empty interrupt clear bit) 1=中断清除 0=中断没有清除

### 21.4.7 全局控制寄存器(QSPI\_GCTL)

偏移地址: 0x18

复位值: 0x0000 0004

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				DATA_SEL	NSS_SEL	DMAE_N	TXTLF	RXTLF	RXEN	TXEN	MM	INT_E_N	SPIE_N		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 12	保留
位11	<b>DATA_SEL:</b> 发送和接收数据寄存器有效数据选择 (Valid byte or double-word data select signal) 0: 只有低8位有效 1: 32位数据都有效 注: 不管是通过CPU还是DMA都必须用指定数据格式访问。
位10	<b>NSS_SEL:</b> 硬件或软件控制主模式下的NSS输出 (NSS select signal that from software or hardware) 0: 由NSSR寄存器值控制 1: 进行数据传输时硬件自动控制
位9	<b>DMAEN:</b> 接收和发送的DMA模式使能 (DMA access mode enable) 0: DMA模式禁止 1: DMA模式使能
位8: 7	<b>TXTLF:</b> 发送缓冲器触发DMA请求的边沿选择 (TX FIFO trigger level bit) 00: 发送缓冲器有大于等于1个空闲数据空间时即进行DMA请求或发送中断请求 01: 发送缓冲器有超过一半的空闲空间时即进行DMA请求或发送中断请求 1x: 保留 注: 当DATA_SEL为0时, 一个数据空间代表1个字节; 为1时, 一个数据空间代表4字节。
位6: 5	<b>RXTLF:</b> 接收缓冲器触发DMA请求的边沿选择 (RX FIFO trigger level bit) 00: 接收缓冲器有大于等于1个有效数据时即进行DMA请求或接收中断请求 01: 接收缓冲器有超过一半的有效数据时即进行DMA请求或接收中断请求 1x: 保留 注: 当DATA_SEL为0时, 一个有效数据代表1个字节; 为1时, 一个有效数据代表4字节。
位4	<b>RXEN:</b> 接收使能位 (Receive enable bit) 1=接收使能 0=接收禁止。同时可以清空RX 缓冲器 注意: 当SPI只工作在主机接收模式时, rxen必须设置为0
位3	<b>TXEN:</b> 发送使能位 (Transmit enable bit) 1=发送使能 0=发送禁止。同时可以清空TX 缓冲器 注意: 当在主机模式下发送和接收同时发生

位2	<b>MM:</b> 主机模式位 (Master mode bit) 1=主机模式 (由内部BRG产生串行时钟) 0=从机模式 (串行时钟来自外部主机)
位1	<b>INT_EN :</b> SPI中断使能位 (SPI interrupt enable bit) 1=使能SPI中断 0=禁止SPI中断
位0	<b>SPIEN :</b> SPI选择位 (SPI select bit) 0=SPI禁止 (复位状态) 1=SPI使能

#### 21.4.8 通用控制寄存器(QSPI\_CCTL)

偏移地址: 0x1C

复位值: 0x0000 0008

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										<b>TXED</b> GE	<b>RXED</b> GE	<b>SPILE</b> N	<b>LSBF</b> E	<b>CPOL</b>	<b>CPHA</b>
rw rw rw rw rw rw rw															

位31: 6	保留
位5	<b>TXEDGE:</b> 发送数据相位调整位(从模式) (Transmit data edge select) 1=发送数据立即发送到数据总线, 可用于高速模式时(SPBRG=4) 0=发送数据在一个有效时钟边沿后发送到数据总线, 可用于低速模式时(SPBRG>4)
位4	<b>RXEDGE:</b> 接收数据采样时钟沿选择位(主模式) (Receive data edge select) 1=在传输数据位的尾时钟沿采样数据 (用于高速模式) 0=在传输数据位的中间采样数据
位3	<b>SPILEN :</b> SPI数据宽度位 (SPI character length bit) 1=8位数据(缺省) 0=7位数据
位2	<b>LSBFE :</b> LSB在前使能位 (LSI first enable bit) 1=数据传输或接收最低位在前 0=数据传输或接收最高位在前
位1	<b>CPOL:</b> 时钟极性标志位 (Clock polarity select bit) 1=时钟在空闲状态为高电平 0=时钟在空闲状态为低电平
位0	<b>CPHA:</b> 时钟相位选择位 (Clock phase select bit) 1=数据采样从第一个时钟边沿开始 0=数据采样从第二个时钟边沿开始

### 21.4.9 波特率发生器(QSPI\_SPBRG)

偏移地址: 0x20

复位值: 0x0000 0002

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

位31: 0

**SPBRG** : SPI波特率控制寄存器用于产生波特率 (SPI baud rate control register for baud rate)

波特率公式:

$$\text{波特率} = \text{fpclk}/\text{SPBRG}$$

(fpclk/是APB时钟频率)

注意: 不要往该寄存器写0和1

### 21.4.10 接收数据个数寄存器(QSPI\_RXDNR)

偏移地址: 0x24

复位值: 0x0000 0001

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
RXDNR [15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 0

**RXDNR**: 该寄存器用于存储下次接收过程需要接收字节的个数 (The register is used to hold a count of to be received bytes in next receive process)

该寄存器的值在SPI为主机接收模式下有效。缺省值是1。该寄存器值通过MCU写值改变。

注意: 不要往该寄存器写“0”值。

### 21.4.11 从机片选寄存器(QSPI\_SCSR)

偏移地址: 0x28

复位值: 0x0000 00FF

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

位15: 8	保留
位7: 0	<b>CSN:</b> 主模式下片选输出信号。低有效，从模式下该位无效 (Chip select output signal in Master mode)。 0: 从器件被选中 1: 从器件未选中

### 21.4.12 从机片选寄存器(QSPI\_MODE)

偏移地址: 0x2C

复位值: 0x0000 000C

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												IO3_P	IO2_P	TRANF_MO	
												OL_S	OL_S	DE_SEL	
												EL	EL		
rw      rw															

位31: 2	保留
位3	<b>IO3_POL_SEL:</b> IO3输出极性选择位 0: 输出状态为0 1: 输出状态为1 注: 仅在STANDARD、DUAL模式下需要选择
位2	<b>IO2_POL_SEL:</b> IO2输出极性选择位 0: 输出状态为0 1: 输出状态为1 注: 仅在STANDARD、DUAL模式下需要选择
位1: 0	<b>TRANF_MODE_SEL:</b> SPI 模式选择 (mode select)。 00: 标准SPI模式 (只有master) 01: Dual SPI模式 10: Quad SPI 模式 11: 保留

## 22.通用异步收发器（UART）

### 22.1 UART 简介

通用异步收发器(UART)提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用分数波特率发生器提供宽范围的波特率选择。它支持同步单向通信和半双工单线通信，以及调制解调器(CTS/RTS)操作。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

### 22.2 UART 主要特征

- 支持异步方式下 RS-232S 协议，符合工业标准 16550。
- 支持 DMA 请求
- 全双工异步操作
- 内置 16 位的可编程波特率发生器。
- 单独分开的发送和接收缓冲寄存器
- 内置一个字节发送和接收缓冲
- 发送和接收数据低位在前
- 一个起始位开始，后面接数据位，输出的数据长度可为 5 位、6 位、7 位、8 位，最后为停止位。  
另外可选择是否有加奇偶校验位，奇偶校验位在数据位之后停止位之前。
- 支持硬件奇数或者偶数校验产生和侦测
- 线断开产生和侦测
- 支持硬件自动流控制
- 支持下面中断源：
  - 发送端 BUFFER 空
  - 接收端数据有效
  - 接收缓冲缓存溢出
  - 帧错误
  - 奇偶校验错误
  - 断开错误

### 22.3 UART 功能概述

任何 UART 双向通信至少需要两个脚：接收数据输入(RX)和发送数据输出(TX)。

**RX:** 接收数据串行输入。通过过采样技术来区别数据和噪音，从而恢复数据。

**TX:** 发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，并且不发送数据时，TX 引脚处于高电平。

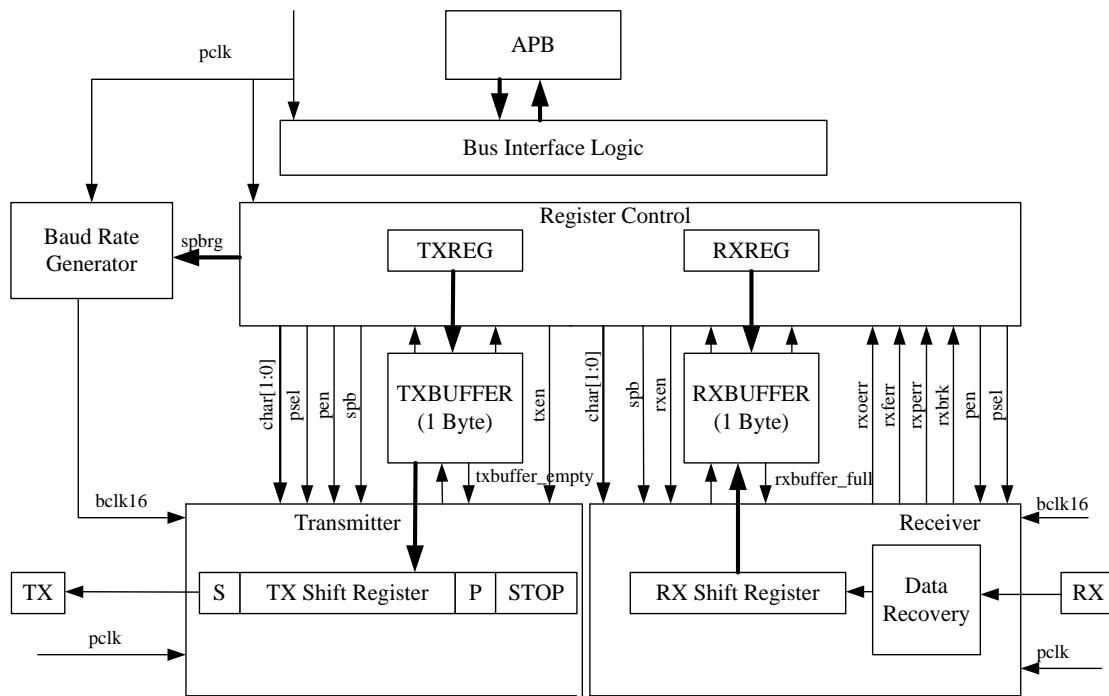
- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(5, 6, 7 或 8 位)，最低有效位在前
- 1.5, 2 个的停止位，由此表明数据帧的结束
- 使用 16 位波特率发生器

下列引脚在硬件流控模式中需要：

- nCTS: 清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。

- nRTS: 发送请求, 若是低电平, 表明 UART 准备好接收数据。

图 168. UART 方框图



### 22.3.1 UART 特性描述

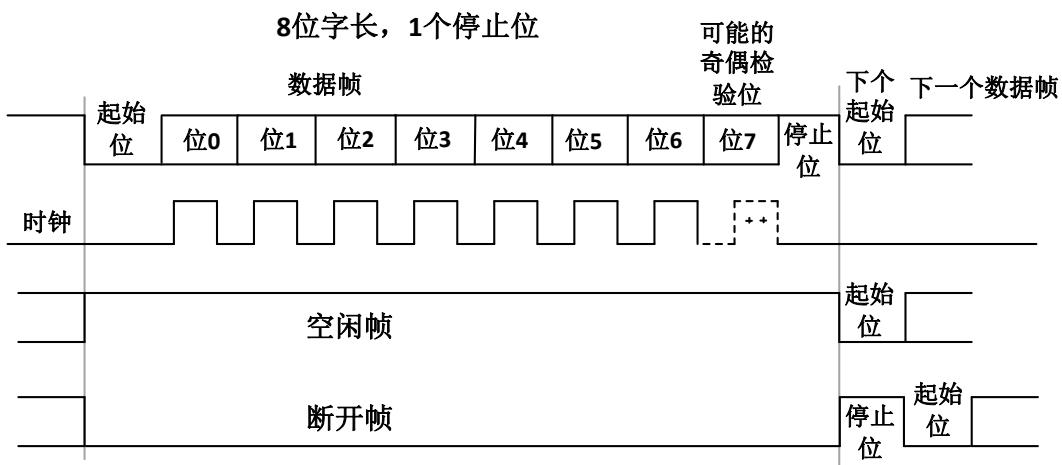
字长可以通过编程 **UART\_CCR** 寄存器中的 **CHAR** 位, 选择 5~8 位。在起始位期间, **TX** 脚处于低电平, 在停止位期间处于高电平。

空闲符号被视为完全由 ‘1’ 组成的一个完整的数据帧, 后面跟着包含了数据的下一帧的开始位(‘1’的位数也包括了停止位的位数)。

断开符号被视为在一个帧周期内全部收到 ‘0’ (包括停止位期间, 也是 ‘0’)。在断开帧结束时, 发送器再插入 1 或 2 个停止位(‘1’)来应答起始位。

发送和接收由一个共用的波特率发生器驱动, 当发送器和接收器的使能位分别置位时, 分别为其产生时钟。

图 169. UART 时序



### 22.3.2 发送器

发送器根据 CHAR 位的状态发送 5~8 位的数据字。当发送使能位 (TXEN) 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 SCLK 脚上输出。

#### 字符发送

在 UART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，UART\_TDR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

**注：**在数据传输期间不能复位 TE 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

#### 可配置的停止位

随每个字符发送的停止位的位数可以通过 SPB 位进行编程。

断开帧是 10 位低电平，后跟停止位；或者 11 位低电平，后跟停止位。不可能传输更长的断开帧(长度大于 10 或者 11 位)，否则会置位中断状态寄存器的 RXBRK\_INTF 位。

#### 配置步骤

1. 通过在 UART\_GCR 寄存器上置位 UARTEN 位来激活 UART。
2. 编程 UART\_CCR 的 CHAR 位来定义字长。
3. 在 UART\_CCR 中 SPB 编程停止位的位数。
4. 设置 UART\_GCR 中的 TXEN 位。
5. 利用 UART\_BRR 寄存器选择要求的波特率。
6. 把要发送的数据写进 UART\_TDR 寄存器(此动作清除 TX\_INTF 位)。在只有一个缓冲器的情况下，对每个待发送的数据重复步骤 6。

#### 单字节通信

清零 TX\_INTF 位总是通过对数据寄存器的写操作来完成的。TX\_INTF 位由硬件来设置，它表明：

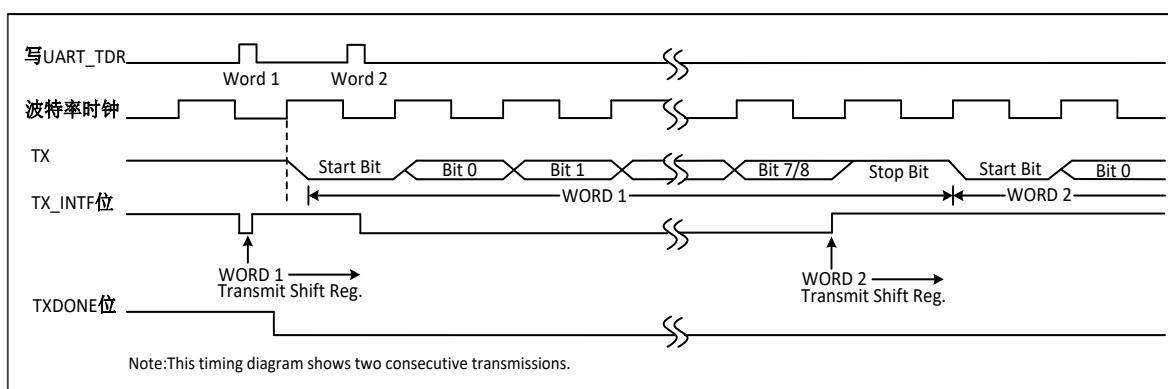
- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始
- TDR 寄存器被清空

- 下一个数据可以被写进 **UART\_TDR** 寄存器而不会覆盖先前的数据。

如果 **TXIEN** 位被设置，此标志将产生一个中断。如果此时 **UART** 正在发送数据，对 **UART\_TDR** 寄存器的写操作把数据存进 **TDR** 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 **UART** 没有在发送数据，处于空闲状态，对 **UART\_TDR** 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，**TX\_INTF** 位立即被置起。同时 **UART\_CSR** 的 **TXBUF\_EMPTY** 也会置起。当一帧发送完成时(停止位发送后)，同时没有往 **UART\_TDR** 写入新的数据(**TDR** 寄存器为空)，**TXC** 会置位，表示所有的传输都已经完成。

图 170. 发送时状态位变化



### 断开符号

设置 **BRK** 可发送一个断开符号。如果设置 **BRK=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(在断开字符的停止位时)软件必须设置 **BRK=0**。**UART** 在最后一个断开帧的结束处插入一逻辑‘1’，以保证能识别下一帧的起始位。

### 22.3.3 接收器

#### 字符接收

在 **USART** 接收期间，数据的最低有效位首先从 **RX** 脚移进。在此模式里，**UART\_RDR** 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

- 将 **USART\_GCR** 寄存器的 **UARTEN** 置 1 来激活 **UART**。
- 编程 **USART\_CCR** 的 **CHAR** 位定义字长。
- 在 **USART\_CCR** 中 **SPB** 编程停止位的位数。
- 利用 **USART\_BRR** 寄存器选择要求的波特率。
- 设置 **USART\_GCR** 的 **RXEN** 位。激活接收器，使它开始寻找起始位。

当一字符被接收到时，

- RX\_INTF** 位被置位。它表明移位寄存器的内容被转移到 **RDR**。换句话说，数据已经被接收并且可以被读出(包括与之有关的错误标志)。
- 如果 **RXIEN** 位被设置，产生中断。
- 在接收期间如果检测到帧错误，或溢出错误，错误标志将被置起，
- 软件读 **UART\_RDR** 寄存器。**RX\_INTF** 位必须在下一字符接收结束前被清零。

**注:** 在接收数据时, RXEN 位不应该被复位。如果 RXEN 位在接收时被清零, 当前字节的接收被丢失。

### 断开符号

当接收到一个断开帧时, UART 会置位 RXBRK\_INTF 中断。

### 溢出错误

如果在 UART\_RDR 没有读出前又接收到一个字符, 则发生溢出错误。

当溢出错误产生时:

- RXOERR\_INTF 位被置位。
- RDR 内容将不会丢失。读 UART\_RDR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXOERREN 位被设置, 中断产生。

### 帧错误

当停止位没有在预期的时间上接收和识别出来时检测到帧错误。当帧错误被检测到时:

- RXFERR\_INTF 位被硬件置起。
- 无效数据不会从移位寄存器传送到 UART\_RDR 寄存器。
- 如果 RXFERREN 位被设置, 中断产生。

### 22.3.4 波特率发生器 (BRG)

BRG 是一个专用的 16 位波特率发生器。UART\_BRR 寄存器控制 16 位自由运转的计数器的计数周期。

提供期望的波特率和 Fosc (APB 时钟频率), 使用下表所示的公式计算出的值计算近似整数赋值给 UART\_BRR(SPBRG) 寄存器。其中下表中的 X 等于 UART\_BRR(SPBRG) 寄存器的值 (1~65535)。同时还可以计算出波特率的误差。

表 40. 波特率公式

模式	Formula
UART模式	波特率=Fosc/16X

X = SPBRG 寄存器值 (1 to 65535)

**例 4-1** 下面是计算波特率误差的例子:

Fosc = 100 MHz

期望波特 = 9600

期望波特率 = Fosc / 16X

9600 = 100000000 / 16X

X = 651.04 = 651

波特率计算值 = 100000000 / 16 \* 651 = 9600.6

误差= (波特率计算值-波特率期望值)/波特率期望值

= (9600.6 - 9600) / 9600

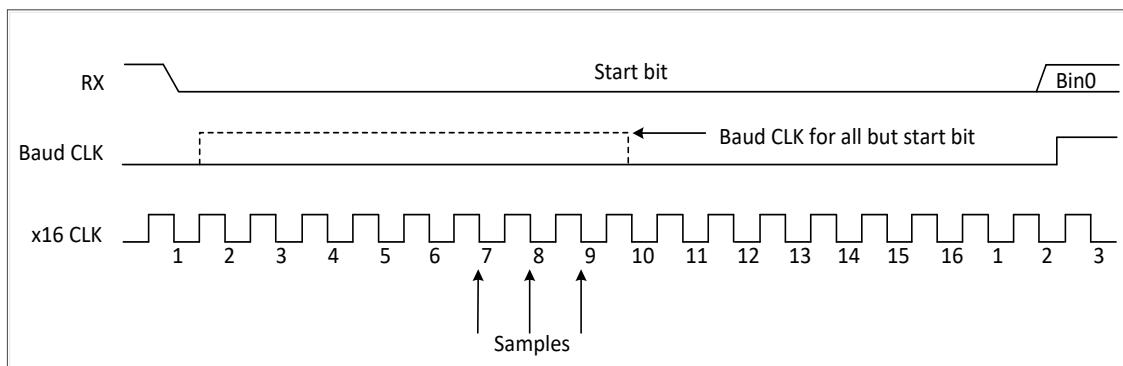
= 0.006%

往 SPBRG 寄存器写入新值会将 BRG 计数器复位（或者清零）。该功能确保了 BRG 不要等到下一个计数溢出后才产生新的波特率。

### 22.3.5 采样

由于异步操作没有单独的时钟，接收器需要一个同步于接收器方法。为了能够在接受引脚“RX”获得正确的字符数据，UART 有一个检测电路。UART 采用 16 倍数据波特率“bclk16”的时钟进行采样 RX 引脚的数据，每个数据有 16 个时钟采样，取中间第 7, 8, 9 的下降沿的采样值。

图 171. RX 引脚采样方案



### 22.3.6 校验控制

奇偶控制(发送时生成一个奇偶位，接收时进行奇偶校验)可以通过设置 UART\_CCR 寄存器上的 PEN 位而激活。如果奇偶校验出错，无效数据不会从移位寄存器传送到 UART\_RDR 寄存器。

**偶校验：**校验位使得一帧中的数据以及校验位中‘1’的个数为偶数。

例如：数据=00110101，有 4 个‘1’，如果选择偶校验(在 UART\_CCR 中的 PSEL=0)，校验位将是‘0’。

**奇校验：**此校验位使得一帧中的数据以及校验位中‘1’的个数为奇数。

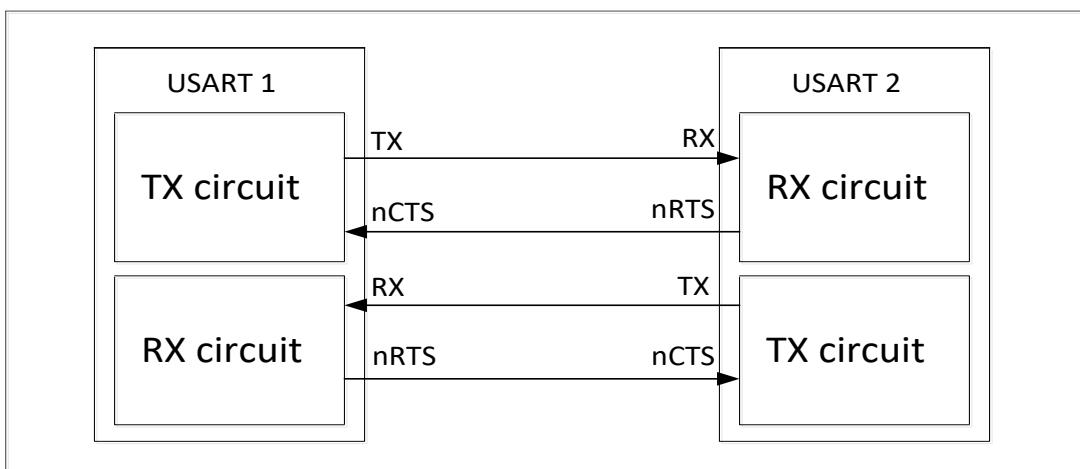
例如：数据=00110101，有 4 个‘1’，如果选择奇校验(在 UART\_CCR 中的 PSEL=1)，校验位将是‘1’。

**传输模式：**如果 UART\_CCR 的 PEN 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去(如果选择偶校验偶数个‘1’，如果选择奇校验奇数个‘1’)。如果奇偶校验失败，UART\_ISR 寄存器中的 RXPERR\_INTF 标志被置‘1’，并且如果 RXPERREN 在被预先设置的话，中断产生。

### 22.3.7 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 172. 两个 USART 间的硬件流控制

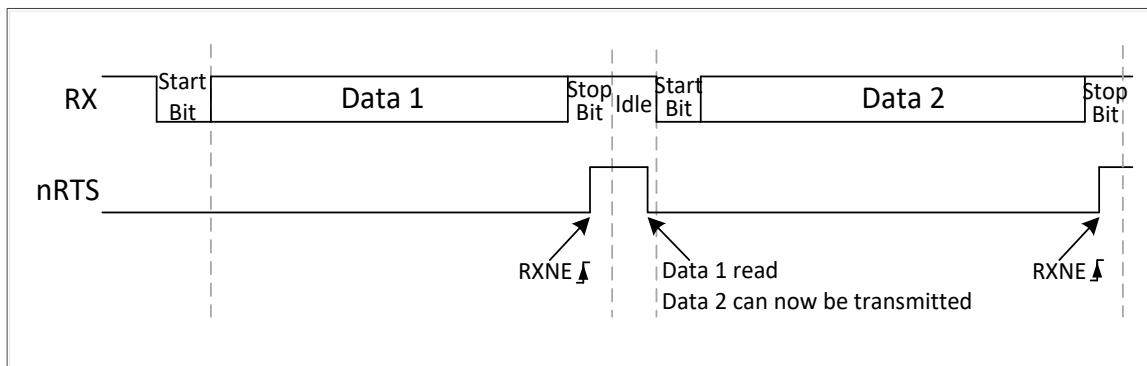


通过将 UASRT\_GCR 中的 AUTOFLLOWEN 置位，可以使能 RTS 和 CTS 流控制。

### RTS 流控制

如果 RTS 流控制被使能，只要 UART 接收器准备好接收新的数据，nRTS 就变成有效(接低电平)。当接收寄存器内有数据到达时，nRTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

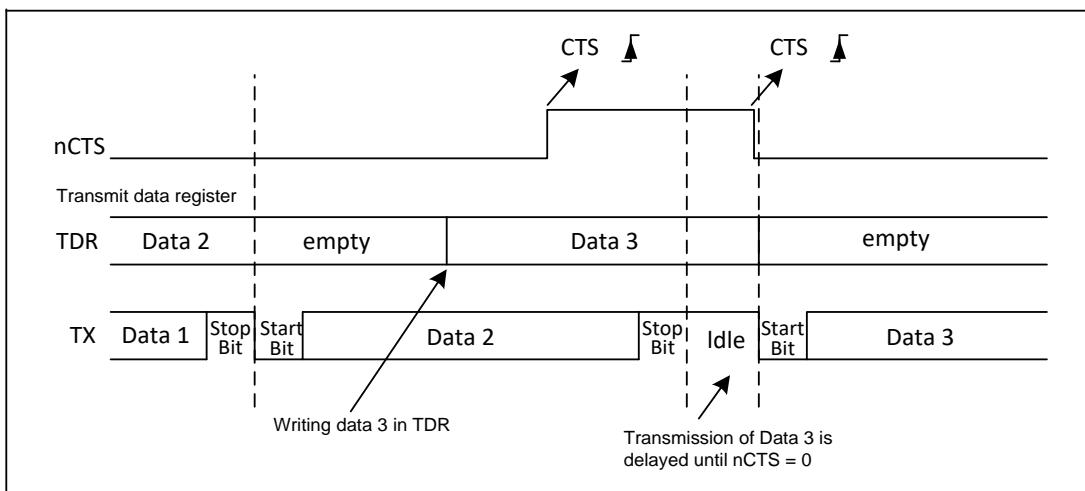
图 173. RTS 流控制



### CTS 流控制

如果 CTS 流控制被使能，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)，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。下图是一个 CTS 流 控制被启用的通信的例子。

图 174. CTS 流控制



### 22.3.8 利用 DMA 通信

UART 可以利用 DMA 进行通信。

#### 利用 DMA 发送

使用 DMA 进行发送时，首先在 DMA 控制寄存器上将 UART\_TDR 寄存器的地址配置成 DMA 传输的目的地址，将存储器地址配置成 DMA 传输的源地址，并配置传输的数据量。通过设置 UART\_GCR 寄存器的 DMAMODE 位来激活 DMA 模式。当 TXEN 位被置 ‘1’ 时，DMA 就从指定的 SRAM 区传送数据到 UART\_TDR 寄存器。

#### 利用 DMA 接收

使用 DMA 进行接收时，首先在 DMA 控制寄存器上将 UART\_RDR 寄存器的地址配置成 DMA 传输的源地址，将存储器地址配置成 DMA 传输的目的地址，并配置传输的数据量。通过设置 UART\_GCR 寄存器的 DMAMODE 位来激活 DMA 模式。当 RXEN 位使能时，每接收到一个字节，DMA 就把数据从 UART\_RDR 寄存器传送到指定的 SRAM 区。

## 22.4 UART 中断请求

表 41. UART 中断请求

中断事件	中断状态	使能位
发送缓冲空	TX_INTF	TXIEN
接收到有效数据	RX_INTF	RXIEN
接收溢出错误	RXOERR_INTF	RXOERREN
奇偶校验错误	RXPERR_INTF	RXPERRREN
帧错误	RXFERR_INTF	RXFERREN
UART接收断开帧	RXBKRK_INTF	RXBKRKEN

如果设置了对应的中断使能控制位，这些设置就可以产生各自对应的中断。

## 22.5 UART 寄存器描述

### 22.5.1 UART 发送数据寄存器(UART\_TDR)

偏移地址: 0x00

复位值: 0x0000 0000



位31: 8	保留, 读始终为0.
位7: 0	<b>TXREG:</b> 发送数据寄存器 (Transmit data register)

### 22.5.2 UART 接收数据寄存器 (UART\_RDR)

偏移地址: 0x04

复位值: 0x0000



位31: 8	保留, 读始终为0
位7: 0	<b>RXREG:</b> UART接收数据寄存器 (Receive data register) 该寄存器只读。

### 22.5.3 UART 当前状态寄存器(UART\_CSR)

偏移地址: 0x08

复位值: 0x0009

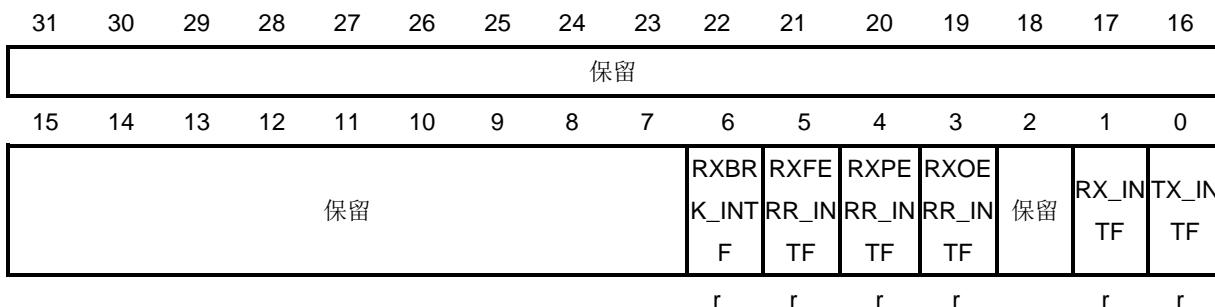


位31: 4	保留, 读始终为0.
位3	<b>TXBUF_EMPTY:</b> 发送缓冲空标识位 (Transmit buffer empty flag bit) 1 = 发送缓冲为空 0 = 发送缓冲不为空
位2	<b>TXFULL:</b> 发送缓冲满满标志位 (Transmit buffer full flag bit) 1 = 发送缓冲为满 0 = 发送缓冲不满
位1	<b>RXAVL:</b> 接收有效字节数据标识位 (Receive valid data flag bit) 当接收缓冲接收了一个完整字节的数据时置位该位. 1 = 接收缓冲接收了一个完整有效的字节数据 0 = 接收缓冲为空
位0	<b>TXC:</b> 发送结束标识位 (Transmit complete flag bit) 1 = 发送缓冲和发送移位寄存器都为空 0 = 发送不为空

### 22.5.4 UART 中断状态寄存器(UART\_ISR)

偏移地址: 0x0C

复位值: 0x0000



位31: 7	保留, 读始终为0.
--------	------------

位6	<b>RXBRK_INTF:</b> UART接收断开帧中断标志位 (Receive frame break interrupt flag bit) 在异常停止位后RX引脚在一段时间内接收到10个或大于10位的低电平 1=检测断开帧 0=没有断开帧
位5	<b>RXFERR_INTF:</b> 帧错误中断标志位 (Frame error interrupt flag bit) 帧错误发生在当检测到异常停止位。 1 = 检测一个帧错误 0 = 没有帧错误
位4	<b>RXPERR_INTF:</b> 奇偶校验错误中断标志位 (Parity error interrupt flag bit) 1 = 检测到奇偶校验错误 0 = 没有奇偶校验错误
位3	<b>RXOERR_INTF:</b> 接收溢出错误中断标志位 (Receive overflow error interrupt flag bit) 仅当autoflowen=0 时置位 1 = 接收溢出错误 0 = 没有溢出错误
位2	保留
位1	<b>RX_INTF:</b> 接收有效数据中断标志位 (Receive valid data interrupt flag bit) 当接收缓冲接收了一个完整字节的数据时置位该位。 1 = 接收缓冲有效字节数据 0 = 接收缓冲为空
位0	<b>TX_INTF:</b> 发送缓冲空中断标志位 (Transmit buffer empty interrupt flag bit) 1 = 发送缓冲空 0 = 发送缓冲不为空

### 22.5.5 UART 中断使能寄存器 (UART\_IER)

偏移地址: 0x10

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
RXBR KEN	RXFE RREN	RXPE RRREN	RXOE RRREN	TIME OUTEN	RXIE N	TXIE N									

位31: 7	保留
位6	<b>RXBRKEN:</b> UART接收断开帧中断使能位 (Receive frame break interrupt enable bit) 1=中断使能 0=中断禁止

位5	<b>RXFERREN:</b> 帧错误中断使能位 (Frame error interrupt enable bit) 1=中断使能 0=中断禁止
位4	<b>RXPERRREN:</b> 奇偶校验错误中断使能位 (Parity error interrupt enable bit) 1=中断使能 0=中断禁止
位3	<b>RXOERREN:</b> 接收溢出错误中断使能位 (Receive overflow error interrupt enable bit) 1=中断使能 0=中断禁止
位2	<b>TIMEOUTEN:</b> 接收数据超时中断使能标志 (Receive timeout interrupt enable bit) 1=中断使能 0=中断禁止
位1	<b>RXIEN:</b> 接收缓冲中断使能位 (Receive buffer interrupt enable bit) 1=中断使能 0=中断禁止
位0	<b>TXIEN:</b> 发送缓冲空中断使能位 (Transmit buffer empty interrupt enable bit) 1=中断使能 0=中断禁止

## 22.5.6 UART 中断清除寄存器 (UART\_ICR)

偏移地址: 0x14

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
									RXBR KCLR	RXFE RRCL R	RXPE RRCL R	RXOE RRCL R	TIME OUTC LR	RXICL R	TXICL R
									W	W	W	W	W	W	W

位31: 7	保留
位6	<b>RXBKRCLR:</b> UART接收断开帧中断清除位 (Receive frame break interrupt clear bit) 1=中断清除 0=中断没有清除
位5	<b>RXFERRCLR:</b> 帧错误中断清除位 (Frame error interrupt clear bit) 1=中断清除 0=中断没有清除
位4	<b>RXPERRCLR:</b> 奇偶校验错误中断清除位 (Parity error interrupt clear bit) 1=中断清除 0=中断没有清除

位3	<b>RXOERRCLR:</b> 接收溢出错误中断清除位 (Receive overflow error interrupt clear bit) 1=中断清除 0=中断没有清除
位2	<b>TIMEOUTCLR:</b> 接收数据超时中断清除标志 (Receive timeout interrupt clear bit) CPU必须先读RXREG然后才能清除该中断 1=中断清除 0=中断没有清除
位1	<b>RXICLR:</b> 接收中断清除位 (Receive interrupt clear bit) 1=中断清除 0=中断没有清除
位0	<b>TXICLR:</b> 发送缓冲空中断清除位 (Transmit buffer empty interrupt clear bit) 1 =中断清除 0=中断没有清除

### 22.5.7 UART 全局控制寄存器 (UART\_GCR)

偏移地址: 0x18

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留															
											TXEN	RXEN	AUTO FLOW EN	DMA MOD E	UART EN

位31: 5	保留
位4	<b>TXEN:</b> 发送使能位 (Enable transmit) 1=发送使能 0=发送禁止。可以清除TX BUFFER
位3	<b>RXEN:</b> 接收使能位 (Enable receive) 1 = 接收使能 0 = 接收禁止。可以清除RX BUFFER.
位2	<b>AUTOFLOWEN:</b> 自动流控制使能位 (Automatic flow control enable bit) 1 = 自动流控制使能 0 = 自动流控制禁止
位1	<b>DMAMODE:</b> DMA方式选择位 (DMA mode selection bit) 1 = 选择DMA方式 0 = 选择正常方式
位0	<b>UARTEN:</b> UART模块选择位 (UART mode selection bit) 1 = UART 模块使能 0 = UART 模块禁止

### 22.5.8 UART 通用控制寄存器 (UART\_CCR)

偏移地址: 0x1C

复位值: 0x0030

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																				
														CHAR	BRK	SPB	PSEL	PEN		
															rw	rw	rw	rw	rw	rw

位31: 6	保留
位5: 4	<b>CHAR:</b> UART 数据位宽度位 (UART width bit) 00 = 5-位数据 01 = 6-位数据 10 = 7-位数据 11 = 8-位数据 (缺省)
位3	<b>BRK:</b> UART发送断开帧 (UART transmit frame break) 1 = 串行强制输出逻辑 “0” (断开帧) 0 = 禁止断开
位2	<b>SPB:</b> 停止位选择 (Stop bit selection) 设置发送停止位位数。接收器通常测查一个停止位。 1 = 2个停止位 (当数据位为5位时停止位为1, 其他情况为2个停止位) 0 = 1个停止位
位1	<b>PSEL:</b> 校验选择位 (Parity selection bit) 当校验使能后, 该位用于选择是采用偶校验还是奇校验。 1 = 偶校验 0 = 奇校验
位0	<b>PEN:</b> 校验使能位 (Parity enable bit) 1 = 发送接收使能校验 0 = 禁止校验

### 22.5.9 UART 波特率寄存器 (UART\_BRR)

偏移地址: 0x20

复位值: 0x0001

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
SPBRG[15: 0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 15	保留
位15: 0	<b>SPBRG:</b> UART波特率控制寄存器 (UART baud rate control register) 用于产生波特率Baud。 波特率 = $F_{osc} / 16SPBRG$ ( $F_{osc}$ 为APB时钟频率) 特别注意： 当SPBRG为0x1， 波特率 = $F_{osc}/16\times 2$ 。

## 23. 安全数字输入/输出接口（SDIO）

### 23.1 SDIO 主要特性

SD/SDIO MMC 卡主机接口（SDIO）提供 APB2 外设总线与多媒体卡（MMC）、SD 卡以及 SDIO 卡设备之间的接口。

SDIO 具有以下特性：

- 完全兼容多媒体卡系统规范版本 2.0-4.2。卡支持两种不同数据总线模式：1 位（默认）、4 位
- 完全兼容先前版本的多媒体卡（向前兼容性）
- 完全兼容 SD 存储卡规范版本 1.0
- 完全兼容 SD 存储卡规范版本 1.1（高速）
- 完全兼容 SD 存储卡规范版本 2.0（SDHC）
- 完全兼容 SD I/O 卡规范版本 1.1.0：卡支持两种不同数据总线模式：1 位（默认）和 4 位
- 支持标准 MMC 模型接口
- 可编程的时钟频率
- 自动命令/回应 CRC 生成/检测
- 自动数据 CRC 生成/检测

### 23.2 SDIO 寄存器

器件通过 32 位控制寄存器（可通过 APB2 访问）与系统进行通信。

外设寄存器必须按字（32 位）进行访问。

#### 23.2.1 SDIO mmc\_ctrl

偏移地址：0x00

复位值：0x0045

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					RDWT EN	INTEN	MDEN	DATW T	SPM		CLKSP		OUTM	SelSM	OPMS el
					rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

位31: 11	保留，读始终为0。
位10	RDWTEN: SDIO读等待使能位（SDIO read wait enable signal） 1: SDIO读等待使能 0: SDIO读等待禁止
位9	INTEN: SDIO中断使能位（SDIO interrupt enable signal） 1: SDIO中断使能 0: SDIO中断禁止
位8	MDEN: SDIO模式选择位（SDIO mode enable） 1: SDIO模式 0: SD/MMC模式

位7	DATWT: 定义SD/MMC/SDIO端口数据传输宽度位 (Define the bus width of SD/MMC/SDIO port DAT line) 1: 4bit 0: 1bit
位6	SelPTSM: SD/MMC/SDIO端口传输速率选择位 (Select SD/MMC/SDIO port transfer speed mode) 1: SD/MMC/SDIO端口高速传输模式 0: SD/MMC/SDIO端口低速传输模式
位5: 3	CLKSP: SD/MMC/SDIO端口时钟CLK速率选择位 (SD/MMC/SDIO port CLK line speed selection) 000: 1/2 base clock 001: 1/4 base clock 010: 1/6 base clock 011: 1/8 base clock 100: 1/10 base clock 101: 1/12 base clock 110: 1/14 base clock 111: 1/16 base clock
位2	OUTM: SD/MMC/SDIO端口CMD输出驱动选择位 (SD/MMC/SDIO port CMD line output driver mode selection) 1: 开漏输出 0: 推挽输出
位1	SelSM: 选择信号模式位 (Select Signal mode) 1: SD/MMC/SDIO端口自动传输模式 0: SD/MMC/SDIO端口使用mmc_port寄存器
位0	OPMSel: SD/MMC/SDIO端口操作模式选择位 (SD/MMC/SDIO port operation mode select) 1: SD/MMC/SDIO模式 0: SPI模式

### 23.2.2 SDIO mmc\_io

偏移地址: 0x04

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留					CMDA F	CMDC H	AUTO CLKG	ENRR ESP	PCLK G	CID/C SDRD	RESP CMDS EL	AUTO TR	TRAN SFDIR	AUTO DATT R	

位31: 10	保留, 读始终为0。
---------	------------

位9	CMDAF: SDIO cmd12/IO中止标志位 (SDIO cmd12/IO Abort flag) 1: 标志当前是cmd/IO中止命令 0: 标志当前不是cmd/IO中止命令
位8	CMDCH: SDIO命令特性位 (SDIO command character) 1: 标记当前命令后跟数据块 0: 标记当前命令后面既没有数据块也没有响应
位7	AUTOCLKG: 使能在回应/命令/单个数据块后自动生成8个空时钟位 (Enable auto generate 8 null clock after response/command or single block data) 1: 使能 0: 禁止
位6	ENRRESP: 使能命令发送之后自动接收回应 (Enable auto receive response after command) 1: 使能 0: 禁止
位5	PCLKG: SD/MMC/SDIO端口CLK上8个空时钟生成位 (SD/MMC/SDIO port CLK line 8 null clocks generation) 1: 8个空时钟生成 0: 通过位3接回应/传输命令选择
位4	CID/CSDRD: CID/CSD读控制位 (CID and CSD read) 1: 读出的CID/CSD存入buffer[135:8] 0: 无效
位3	RESPCMDSEL: 当位5为0时, 回应/命令选择位 (Response/Command selection when bit[5] is '0') 1: 接回应 0: 传输命令
位2	AUTOTR: 设置8位空时钟/命令/回应自动传输位 (Set auto 8null/command/response transfer) 1: 使能8位空时钟/命令/回应自动传输 0: 禁止空时钟/命令/回应自动传输
位1	TRANSFDIR: 设置数据传输方向位 (Set data transfer direction) 1: 读数据 0: 写数据
位0	AUTODATTR: 设置自动数据传输位 (Set auto data transfer) 1: 使能数据自动传输 0: 禁止数据自动传输 当数据传输完成, 该位会自动被清除。

### 23.2.3 SDIO mmc\_bytectl

偏移地址: 0x08

复位值: 0x0200

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

rw

位31: 16	保留, 读始终为0。
位15: 0	数据传输计数器 (Data transfer byte count register)

### 23.2.4 SDIO mmc\_tr\_blockcnt

偏移地址: 0xC

复位值: 0x0000

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

位31: 16	保留, 读始终为0。
位15: 0	当多个数据块传输, 传输完成时的计数器值 (When multiple block transfer, transfer block count that are finished)

### 23.2.5 SDIO mmc\_crcctl

偏移地址: 0x10

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								CMD_ CRCE N	DAT_ CRCE N	ENCH K	ENRD MB	DAT_CRC S	CMD_ CRCE	DAT_ CRCE	
rw								rw	rw	rw	rw	rw	rw	r	r

位31: 8	保留, 读始终为0。
位7	<p>CMD_CRCEN: SD/MMC/SDIO端口CMD的CRC计算控制位 (SD/MMC/SDIO port CMD Line CRC circuit enable)</p> <p>1: 使能 0: 禁止</p>
位6	<p>DAT_CRCEN: SD/MMC/SDIO端口DAT的CRC计算控制位 (SD/MMC/SDIO port DAT Line CRC circuit enable)</p> <p>1: 使能 0: 禁止</p>

位5	ENCHK: 自动检测crc_status[2:0]使能位 (Enable auto check crc_status[2:0]) 1: 使能。如果crc_status[2:0] != 3'b010, crc错误状态中断生成, 写数据传输将被一个停止命令终止并清除mmc_io[0]和mmc_io_mbctl[2:0] 0: 禁止 注: crc_status[2:0]位请参考mmc_sig寄存器
位4	ENRDMB: 接收回应之前读多个数据块使能位 (Enable read multiple block data before response) 1: 使能 0: 禁止
位3: 2	DAT_CRCSEL: DAT CRC选择dat_crcl 和 dat_crch 寄存器存储CRC值控制位 (DAT CRC selection) 00: SD/MMC/SDIO DAT0或MMC DAT的CRC值 01: SD/MMC/SDIO DAT1的CRC值 10: SD/MMC/SDIO DAT2的CRC值 11: SD/MMC/SDIO DAT3的CRC值
位1	CMD_CRCERR: CMD CRC校验标志位 (CMD CRC Error)。只读。
位0	DAT_CRCERR: DAT CRC校验标志位 (DAT CRC Error)。只读。

### 23.2.6 SDIO cmd\_crc

偏移地址: 0x14

复位值: 0x0000

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

位31: 7	保留, 读始终为0。
位6: 0	CMD_CRCVAL: CMD CRC值 (CMD CRC register value)

### 23.2.7 SDIO dat\_crcl

偏移地址: 0x18

复位值: 0x0000

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

位31: 8	保留, 读始终为0。
位7: 0	DAT_CRCVAL: DAT CRC值低位 (The DAT CRC low register value)

**23.2.8 SDIO dat\_crch**

偏移地址: 0x1C

复位值: 0x0000

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

位31: 8	保留, 读始终为0。
位7: 0	DAT_CRCHV: DAT CRC值高位 (The DAT CRC high register value)

**23.2.9 SDIO mmc\_port**

偏移地址: 0x20

复位值: 0x007f

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								PCLK S	PCM DS	PDAT S	AUTO NTEN	NTCR			
								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留, 读始终为0。
位7	PCLKS: SD/MMC/SDIO端口CLK信号 (SD/MMC/SDIO port CLK line signal)
位6	PCMDs: SD/MMC/SDIO端口CMD信号 (SD/MMC/SDIO port CMD line signal)
位5	PDATS: SD/MMC/SDIO端口DAT信号 (SD/MMC/SDIO port DAT line signal)
位4	AUTONTEN: 自动Ncr超时使能位 (Auto Ncr Timer out enable) 1: 使能自动检查Ncr超时 0: 禁止自动检查Ncr超时
位3: 0	NTCR: Ncr超时计数器 (SD/MMC/SDIO时钟数) (Ncr Timeout count register(SD/MMC/SDIO clock number))

**23.2.10 SDIO mmc\_int\_mask**

偏移地址: 0x24

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留								D1INT M	CRCI NTM	CRTIN TM	MBTIN TM	MBDI NTM	CMDE INTM	DATEI NTM	DATDI NT	CMDD INTM
								rw	rw	rw	rw	rw	rw	rw	rw	

位31: 9	保留, 读始终为0。
位8	D1INTM: SDIO data1线中断屏蔽位 (SDIO data1 line interrupt mask) 1: 开放请求 0: 屏蔽请求
位7	CRCINTM: CRC状态错误标志中断屏蔽位 (CRC status token err interrupt mask) 1: 开放请求 0: 屏蔽请求
位6	CRTINTM: 命令和回应Ncr超时中断屏蔽位 (Cmd and Resp Ncr Timeout interrupt mask) 1: 开放请求 0: 屏蔽请求
位5	MBTINTM: 多块传输超时中断屏蔽位 (Multi Block Timeout interrupt mask) 1: 开放请求 0: 屏蔽请求
位4	MBDINTM: 多块传输完成中断屏蔽位 (Multi Block done interrupt mask) 1: 开放请求 0: 屏蔽请求
位3	CMDEINTM: CMD CRC错误中断屏蔽位 (CMD CRC error interrupt mask) 1: 开放请求 0: 屏蔽请求
位2	DATCINTM: DAT CRC错误中毒屏蔽位 (DAT CRC error interrupt mask) 1: 开放请求 0: 屏蔽请求
位1	DATDINTM: DAT完成中断屏蔽位 (DAT done interrupt mask) 1: 开放请求 0: 屏蔽请求
位0	CMDDINTM: CMD完成中断屏蔽位 (CMD done interrupt mask) 1: 开放请求 0: 屏蔽请求

注: 在其他中断生成时, CRC 状态位, Ncr 超时, CMD CRC 错误和 DAT CRC 错误中断无效。

### 23.2.11 SDIO clr\_mmc\_int

偏移地址: 0x28

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							D1MC	CRCE	CRNT	MBTM	MBDM	CMDE	DATE	DATD	CMDD

rw      rw

位31: 9	保留, 读始终为0。
--------	------------

位8	D1MC: SDIO data1线中断标志/清除位 (SDIO data1 line interrupt mask/clear) W (写) : 清除SDIO data1线中断标志 R (读) : SDIO data1线中断标志
位7	CRCEMC: CRC状态错误标志中断屏蔽位 (CRC status token err interrupt mask/clear) W (写) : 清除CRC状态错误标志中断标志 R (读) : CRC状态错误标志中断标志
位6	CRNTMC: 命令和回应Ncr超时中断屏蔽位 (Cmd and Resp Ncr Timeout interrupt mask/clear) W (写) : 清除命令和回应Ncr超时中断标志 R (读) : 命令和回应Ncr超时中断标志
位5	MBTMC: 多块传输超时中断屏蔽位 (Multi Block Timeout interrupt mask/clear) W (写) : 清除多块传输超时中断标志 R (读) : 多块传输超时中断标志
位4	MBDMC: 多块传输完成中断屏蔽位 (Multi Block Done interrupt mask/clear) W (写) : 清除多块传输完成中断标志 R (读) : 多块传输完成中断标志
位3	CMDEMC: CMD CRC错误中断屏蔽位 (CMD CRC error interrupt mask/clear) W (写) : 清除CMD CRC错误中断标志 R (读) : CMD CRC错误中断标志
位2	DATEMC: DAT CRC错误中断屏蔽位 (DAT CRC error interrupt mask/clear) W (写) : 清除DAT CRC错误中断标志 R (读) : DAT CRC错误中断标志
位1	DATDMC: DAT完成中断屏蔽位 (DAT done interrupt mask/clear) W (写) : 清除DAT完成中断标志 R (读) : DAT完成中断标志
位0	CMDMMC: CMD完成中断屏蔽位 (CMD done interrupt mask/clear) W (写) : 清除CMD完成中断标志 R (读) : CMD完成中断标志

### 23.2.12 SDIO mmc\_cardsel

偏移地址: 0x2C

复位值: 0x0040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CTRE	ENPC	TSCALE					
								N	LK						
								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留, 读始终为0。
位7	CTREN: SD/MMC/SDIO控制器使能位 (SD/MMC/SDIO controller enable)
位6	ENPCLK: 使能SD/MMC/SDIO卡端口CLK时钟 (Enable SD/MMC/SDIO port CLK line for card)

位5: 0	TSCALE: SD/MMC/SDIO时钟分频系数(基于1Mhz) (SD/MMC/SDIO Time scale base(1Mhz) coefficient) 1MHz = Fpclk/((mmc_cardssel[5:0] + 1)*2)
-------	---

### 23.2.13 SDIO mmc\_sig

偏移地址: 0x30

复位值: 0x00ff

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

位31: 8	保留, 读始终为0。
位7	PCMDS: SD/MMC/SDIO端口CMD线信号 (SD/MMC/SDIO port CMD Line signal)
位6: 4	CRC status[2:0] 写入数据时CRC状态令牌 (CRC status[2:0] when write data CRC status token)
位3	PDAT3S: SD/MMC/SDIO端口DAT3线信号 (SD/MMC/SDIO port DAT3 Line signal)
位2	PDAT2S: SD/MMC/SDIO端口DAT2线信号 (SD/MMC/SDIO port DAT2 Line signal)
位1	PDAT1S: SD/MMC/SDIO端口DAT1线信号 (SD/MMC/SDIO port DAT1 Line signal)
位0	PDAT0S: SD/MMC/SDIO端口DAT0线信号 (SD/MMC/SDIO port DAT0 Line signal)

注: 当主设备读寄存器, SD/MMC/SDIO 控制器将在 SD/MMC/SDIO 端口的 CLK 线上生成一个周期脉冲, 并且 SD/MMC/SDIO 端口的信号状态将被锁存到寄存器作为 SD/MMC/SDIO 端口 CLK 线的上升沿。

### 23.2.14 SDIO mmc\_io\_mbctl

偏移地址: 0x34

复位值: 0x0010

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								NTSSel	BTSSel	PCLK P	PAUT OTR	SMBD TD	SPMB DTR		
rw								rw	rw	rw	rw	rw	rw	rw	

位31: 8	保留, 读始终为0。
位7: 6	NTSSel: SD/MMC/SDIO Nac 超时级别选择位 (SD/MMC/SDIO NAC timeout scale selection) 00: 1us 01: 100us 10: 10ms 11: 1s

位5: 4	BTSSel: SD/MMC/SDIO Busy超时级别选择位 (SD/MMC/SDIO Busy timeout scale selection) 00: 1us 01: 100us 10: 10ms 11: 1s
位3	PCLKP: SD/MMC/SDIO端口CLK线极性选择位 (SD/MMC/SDIO port CLK line polarity) 1: 时钟下降沿push, 上升沿pull 0: 时钟上升沿push, 下降沿pull
位2	PAUTOTR: 使能SD/MMC/SDIO端口全自动命令和多数据块传输位 (Set SD/MMC/SDIO port full auto cmd and multiple block data transferring) 1: 使能 0: 禁止 注: 如果mmc_io[7:6]==11, 此位置一将触发SD/MMC/SDIO命令, 回应, 8个空时钟, 多数据块。
位1	SMBDTD: 多数据块传输方向选择位 (Select multiple block data transfer direction) 1: 读数据 0: 写数据
位0	SPMBDTR: 使能SD/MMC/SDIO端口自动多块数据传输位 (Set SD/MMC/SDIO port auto multiple block data transfer) 1: 使能 0: 禁止 注: 此位置一将触发SD/MMC/SDIO多块数据传输, 块计数器由mmc_blockcnt寄存器定义。当数据传输完成, 此位将自动清除。

### 23.2.15 SDIO mmc\_blockcnt

偏移地址: 0x38

复位值: 0x0001

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

rw

位31: 16	保留, 读始终为0。
位15: 0	数据块计数寄存器 (Data block number register) 在多块传输模式下, 配置这些位定义将要传输的数据块数量。

**23.2.16 SDIO mmc\_timeoutcnt**

偏移地址: 0x3C

复位值: 0x0040

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

位31: 8	保留, 读始终为0。
位7: 0	<p>DTCNT: 数据传输超时计数寄存器 (Data transfer timeout count register)</p> <p>Time = Scale* bit[7:0]</p> <p>注: Scale根据mmc_io_mbctl[7:6]/[5:4]进行定义。</p>

**23.2.17 SDIO cmd\_bufx(x = 0..15)**

偏移地址: 0x40 – 0x7C

复位值: 0x0000

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

位31: 8	保留, 读始终为0。
位7: 0	<p>cmd_buf字节x, 映射到命令[(15+8x):8(x+1)]位</p> <p>cmd_buf字节0, 映射到命令[15:8]位</p> <p>cmd_buf字节1, 映射到命令[23:16]位</p> <p>cmd_buf字节2, 映射到命令[31:24]位</p> <p>cmd_buf字节3, 映射到命令[39:32]位</p> <p>cmd_buf字节4, 映射到命令[47:40]位</p> <p>cmd_buf字节5, 映射到命令[55:48]位</p> <p>cmd_buf字节6, 映射到命令[63:56]位</p> <p>cmd_buf字节7, 映射到命令[71:64]位</p> <p>cmd_buf字节8, 映射到命令[79:72]位</p> <p>cmd_buf字节9, 映射到命令[87:80]位</p> <p>cmd_buf字节10, 映射到命令[95:88]位</p> <p>cmd_buf字节11, 映射到命令[103:96]位</p> <p>cmd_buf字节12, 映射到命令[111:104]位</p> <p>cmd_buf字节13, 映射到命令[119:112]位</p> <p>cmd_buf字节14, 映射到命令[127:120]位</p> <p>cmd_buf字节15, 映射到命令[135:128]位</p>

**23.2.18 SDIO buf\_ctl**

偏移地址: 0x80

复位值: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBFE N	DRM		DFIFO SM	SBAD	DMAH EN				DBML				DBE		DBF
									rW	rW	rW	rW	rW	rW	rW

位31: 16	保留, 读始终为0。
位15	DBFEN: 数据buff清空使能位 (Data Buf flush enable) 1: 触发清空数据buff 0: 无效 注: 当此位置一, 一个时钟周期后此位自动清空。
位14	DRM: DMA请求屏蔽位 (Dma Request mask) 1: 开放请求 0: 屏蔽请求 注意: 在配置使能DMA前屏蔽此位, 使能DMA后此位置一, DMA请才开始。
位13	保留
位12	DFIFOSM: 数据FIFO状态屏蔽位 (Data FIFO status signal mask bit) 1: 激活显示数据FIFO状态 0: 默认值, 屏蔽数据FIFO状态 注: 数据FIFO状态位, 活跃为高电平。活跃含义, 读卡访问, FIFO满; 写卡访问, FIFO空。
位11	SBAD: 设置buff访问方向位 (Set buff access direction) 1: 写数据 0: 读数据
位10	DMAHEN: DMA硬件接口使能位 (DMA hardware interface enable) 1: DMA硬件接口握手 0: 正常的APB访问buff数据 注: 当使用DMA接口, 块(单块传输)或多块数据传输完成此位将自动复位。
位9: 2	DBML: 数据buff标记, 此位只有buf_ctl[10] = 1才有效。 (Data buff data water mark level)
位1	DBE: 数据buff空状态 (Data buff empty)。只读。
位0	DBF: 数据buff满状态 (Data buff full)。只读。

### 23.2.19 SDIO data\_buf

偏移地址: 0x100-0x2FF

复位值: 0x0000

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

位31: 0      DB: 数据buff (Data buffer)  
注: 该范围内的所有访问都将被视为地址AMBA读访问。

## 24. 控制器局域网（CAN）

### 24.1 CAN 简介

它的设计目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求（优先级特性可软件配置）。

对于安全紧要的应用，bxCAN 提供所有支持时间触发通信模式所需的硬件功能。

### 24.2 CAN 主要特点

- 支持 CAN 协议的 2.0A 和 2.0B
- 扩展的接收缓冲器（64 字节、先进先出 FIFO）
- 同时支持 11 位和 29 位识别码
- 位速率可达 1Mbits/s
- PeliCAN 模式扩展功能
  - 可读/写访问的错误计数器
  - 可编程的错误报警限制
  - 最近一次错误代码寄存器
  - 对每一个 CAN 总线错误的中断
  - 具体控制位控制的仲裁丢失中断
  - 单次发送（无重发）
  - 只听模式（无确认、无活动的出错标志）
  - 软件位速率检测
  - 验收滤波器扩展（4 字节代码，4 字节屏蔽）
  - 自身信息接收（自接收请求）

### 24.3 CAN 控制器的总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量（每个节点都需要处理）急剧增加。除了应用层报文外，网路管理和诊断报文也被引入。

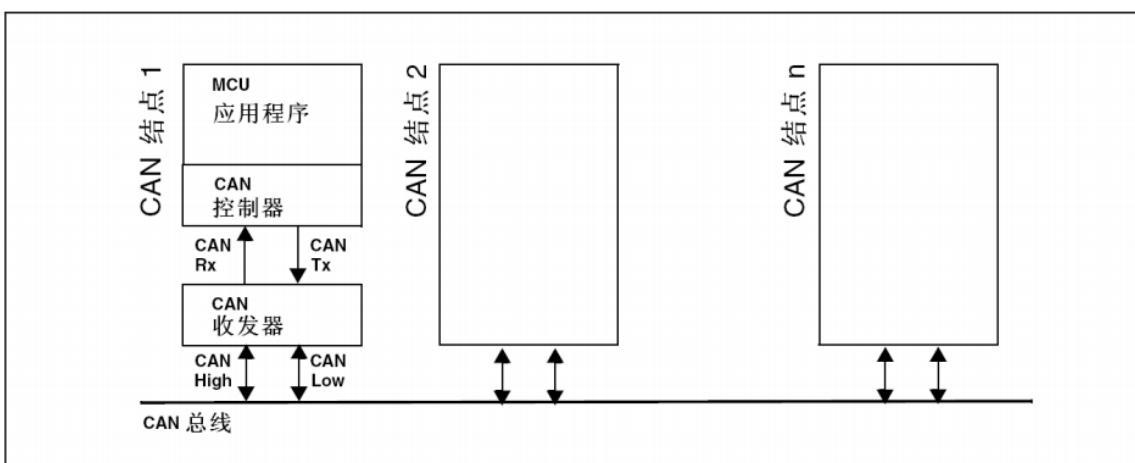
需要一个增强的过滤机制来处理各种类型的报文。

此外应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。

接收 FIFO 的方案允许，CPU 花很长时间处理应用层任务而不会丢失报文。

构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间的高效的接口。

图 175. CAN 网络拓扑结构

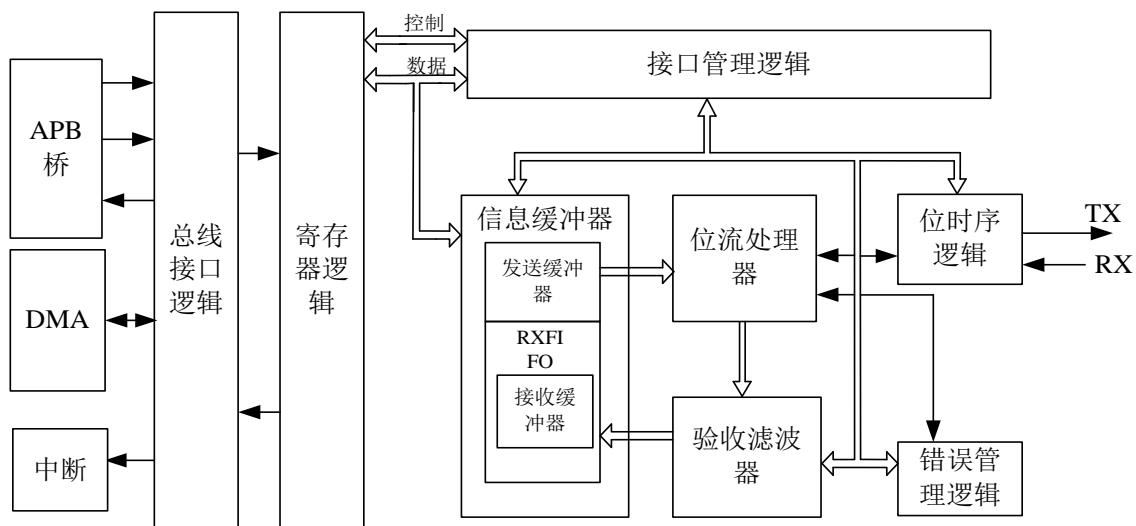


### 24.3.1 CAN 2.0B 主动内核

CAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符（11 位）和扩展标识符（29 位）。

### 24.3.2 CAN 方框图

图 176. CAN 结构方框图



### 24.3.3 接口管理逻辑 (IML)

接口管理逻辑解释来自 CPU 的命令，控制 CAN 寄存器的寻址向主控制器提供中断信息和状态信息。

### 24.3.4 发送缓冲器 (TXB)

发送缓冲器是 CPU 和 BSP（位流处理器）之间的接口，能够存储发送到 CAN 网络上的完整信息。缓冲器长 13 个字节，由 CPU 写入、BSP 读出。

### 24.3.5 接收缓冲器 (RXB, RXFIFO)

接收缓冲器是验收滤波器和 CPU 之间的接口用来储存从 CAN 总线上接收和接收的信息。接收缓冲器 (RXB, 13 个字节) 作为接收 FIFO (RXFIFO, 长 64 字节) 的一个窗口，可被 CPU 访问。

CPU 在此 FIFO 的支持下可以在处理信息的时候接收其它信息。

#### 24.3.6 验收滤波器 (ACF)

验收滤波器把它其中的数据和接收的识别码的内容相比较，以决定是否接收信息。在纯粹的接收测试中，所有的信息都保存在 RXFIFO 中。

#### 24.3.7 位流处理器 (BSP)

位流处理器是一个在发送缓冲器、RXFIFO 和 CAN 总线之间控制数据流的程序装置。它还在 CAN 总线上执行错误检测、仲裁、填充和错误处理。

#### 24.3.8 位时序逻辑 (BTL)

位时序逻辑监视串口的 CAN 总线和处理与总线有关的位时序。它在信息开头‘弱势-支配’的总线传输时同步 CAN 总线位流（硬同步），接收信息时再次同步下一次传送（软同步）。BTL 还提供了可编程的时间段来补偿传播延迟时间、相位转换和定义采样点和一位时间内的采样次数。

#### 24.3.9 错误管理逻辑 (EML)

EML 负责传送层模块的错误管制。它接收 BSP 的出错报告，通知 BSP 和 IML 进行错误统计。

### 24.4 CAN 工作模式

CAN 控制器有 2 个主要的工作模式：

- BasicCAN 模式
- PeliCAN 模式

系统复位时默认模式是 BasicCAN 模式。

PeliCAN 模式是新的操作模式，它能处理所有的 CAN 2.0B 规范的帧类型。而且它还提供一些增强功能使得应用于更宽的领域。

寄存器 CAN\_CDR.7 定义了 CAN 模式。如果 CDR.7 是 0，CAN 控制器工作于 BasicCAN 模式。否则，CAN 控制器工作于 PeliCAN 模式。

#### 24.4.1 Basic CAN 和 PeliCAN 模式的区别

在 Peli CAN 模式下，CAN 控制器有一个含很多功能的重组寄存器。Peli CAN 模式支持 CAN 2.0B 协议规定的所有功能（29 字节的识别码）。下面是 PeliCAN 模式的主要新功能：

- 标准帧和扩展帧的接收和传送
- 接收 FIFO (64 字节)
- 在标准和扩展格式中都有单/双验收滤波器（含屏蔽和代码寄存器）
- 读/写访问的错误计数器
- 可编程的错误限制报警
- 最近一次的误码寄存器
- 对每一个 CAN 总线错误的错误中断
- 仲裁丢失以及详细的位位置
- 一次性发送（当错误或仲裁丢失时不重发）
- 只听模式（CAN 总线监听，无应答，无错误标志）

## 24.5 CAN 功能描述

### 24.5.1 Basic CAN 模式

#### 复位模式

复位模式即初始化模式。在硬件启动或总线状态设置为‘1’（总线关闭）时，复位请求位（CAN\_CR.0）被置为‘1’（当前）。如果这些位被软件访问，其值将发生变化，而且会影响内部时钟的下个上升沿。复位请求位的变化时内部分频时钟同步的读复位请求位能够反映出这种同步状态。复位模式主要用于 CAN 通讯参数配置，在不同工作模式下内核对 CAN 寄存器的访问权限不同。

复位请求位被设为‘0’后 CAN 控制器将会等待：

- a) 一个总线空闲信号（11 个弱势位），如果前一次复位请求是硬件复位或 CPU 初始复位。
- b) 128 个总线空闲，如果前一次复位请求是 CAN 控制器在重新进入总线开启模式前初始化总线造成的；必须说明的是，如果复位请求位被置位，一些寄存器的值会被改变的。

#### 工作模式

在复位模式完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。复位模式下，一旦向复位位传送了‘1-0’的下降沿，CAN 控制器将返回工作模式，进行报文的发送和接收。

#### 睡眠模式

睡眠模式位设为 1（sleep），CAN 控制器将进入睡眠模式；没有总线活动和中断等待。至少破坏这两种情况之一时将会导致睡眠模式产生唤醒中断。睡眠模式位设为低（唤醒）之后总线进入活动状态或中断被激活。唤醒后，时钟启动且产生一个唤醒中断。由于总线活动唤醒的直到检测到 11 个连续的隐藏（弱势）位（总线空闲序列）后才能接收这条信息。注意在复位模式中是不能设置睡眠模式位的。清除复位模式后，再一次检测到总线空闲时，睡眠模式位的设置才开始有效。

Basic CAN 模式寄存器权限分配表：

CAN地址偏移 (HEX)	段	工作模式		复位模式	
		读	写	读	写
00	控制	控制	控制	控制	控制
04		(FFH)	命令	(FFH)	命令
08		状态	—	状态	—
0C		(FFH)	—	中断	—
10		(FFH)	—	验收代码	验收代码
14		(FFH)	—	验收屏蔽	验收屏蔽
18		(FFH)	—	总线定时0	总线定时0
1C		(FFH)	—	总线定时1	总线定时1
20		(FFH)	—	—	—
24		测试	测试	测试	测试
28	发送缓冲器	识别码（10 ~ 3）	识别码（10 ~ 3）	(FFH)	—
2C		识别码（2 ~ 0） RTR 和 DLC	识别码（2 ~ 0）RTR 和 DLC	(FFH)	—
30		DATA1	DATA1	(FFH)	—

CAN地址偏移 (HEX)	段	工作模式		复位模式	
		读	写	读	写
34		DATA2	DATA2	(FFH)	—
38		DATA3	DATA3	(FFH)	—
3C		DATA4	DATA4	(FFH)	—
40		DATA5	DATA5	(FFH)	—
44		DATA6	DATA6	(FFH)	—
48		DATA7	DATA7	(FFH)	—
4C		DATA8	DATA8	(FFH)	—
50	接收缓冲器	识别码 (10 ~ 3)	识别码 (10 ~ 3)	识别码 (10 ~ 3)	识别码 (10 ~ 3)
54		识别码 (2 ~ 0) RTR和DLC	识别码(2 ~ 0)RTR 和DLC	识别码 (2 ~ 0) RTR和DLC	识别码 (2 ~ 0) RTR和DLC
58		DATA1	DATA1	DATA1	DATA1
5C		DATA2	DATA2	DATA2	DATA2
60		DATA3	DATA3	DATA3	DATA3
64		DATA4	DATA4	DATA4	DATA4
68		DATA5	DATA5	DATA5	DATA5
6C		DATA6	DATA6	DATA6	DATA6
70		DATA7	DATA7	DATA7	DATA7
74		DATA8	DATA8	DATA8	DATA8
78		(FFH)	—	(FFH)	—
7C		时钟分频器	时钟分频器	时钟分频器	时钟分频器

注：‘(FFH)’代表读出数据全为1，‘—’代表无写操作权限，其余表示可操作。偏移地址为0x7C时钟分频器，用于选择 BasicCAN 和 PeliCAN。

## 24.5.2 Peli CAN 模式

### 复位模式

复位模式即初始化模式。在硬件复位或总线状态位为‘1’（总线关闭）时复位模式位被置为‘1’（当前）。如果通过软件访问这一位，值将发生变化且下一个内部时钟（频率为外部振荡器的 1/2）的上升沿有效。复位请求位的改变和内部分频时钟同步。读复位请求位能够反映出这种同步状态。复位模式位为‘0’后，CAN 控制器会等待：

- a) 一个总线空闲信号（11 个隐藏（弱势）位），如果上一次复位是硬件复位或 CPU 初始复位。
- b) 128 个总线空闲，如果上一次复位是 CAN 控制器在重新进入总线开启之前初始化复位。

### 工作模式

在复位模式完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。复位模式下，一旦检测到 CAN\_MOD 寄存器的 RM 位出现了‘1-0’的下降沿，CAN 控制器将返回工作模式，进行报文的发送和接收。

## 睡眠模式

睡眠模式位（CAN\_MOD.4）设为 1（sleep），CAN 控制器将进入睡眠模式；没有总线活动和中断等待。至少破坏这两种情况之一时将会导致睡眠模式产生唤醒中断。睡眠模式位设为低（唤醒）之后总线进入活动状态或中断被激活。唤醒后，时钟启动且产生一个唤醒中断。由于总线活动唤醒的直到检测到 11 个连续的隐藏（弱势）位（总线空闲序列）后才能接收这条信息。注意在复位模式中是不能设置睡眠模式位的。清除复位模式后，再一次检测到总线空闲时，睡眠模式位的设置才开始有效。

## 自检测模式

此模式主要用于测试。设置自检测模式位（CAN\_MOD.2）为 1，进入自检测模式。此模式可以检测所有节点，没有任何活动的节点使用自接收命令；即使没有应答，CAN 控制器也会成功发送。

## 只听模式

此主要用于测试。设置只听模式位（CAN\_MOD.1）为 1 进入只听模式。这种工作模式使 CAN 控制器进入错误消极状态。信息传送是不可能的。以软件驱动的位速检测可使用只听模式。所有其它功能都能象在正常工作模式中一样使用。

在该模式中，CAN 控制器不能再 CAN 总线上写显性位。激活错误标志或超载标志不能都写，成功接收后的应答信号也不会给出。

**注：在进入只听模式之前，必须进入复位模式。**

Peli CAN 模式寄存器权限分配表：

CAN地址偏移 (HEX)	工作模式			复位模式	
	读	写	读	写	
00	模式	模式	模式	模式	模式
04	(00H)	命令	(00H)	命令	
08	状态	—	状态	—	
0C	中断	—	中断	—	
10	中断使能	—	中断使能	中断使能	
14	(00H)	—	(00H)	—	
18	总线定时0	—	总线定时0	总线定时0	
1C	总线定时1	—	总线定时1	总线定时1	
20	保留	—	—	—	
24	检测	检测	检测	检测	
28	保留	—	保留	—	
2C	仲裁丢失捕捉	—	仲裁丢失捕捉	—	
30	错误代码捕捉	—	错误代码捕捉	—	
34	错误报警限制	—	错误报警限制	错误报警限制	
38	RX错误计数器	—	RX错误计数器	RX错误计数器	
3C	TX错误计数器	—	TX错误计数器	TX错误计数器	
40	RX帧信息 SFF	RX帧信息 EFF	TX帧信息 SFF	TX帧信息 EFF	验收代码0
44	RX识别码1	RX识别码1	TX识别码1	TX识别码1	验收代码1

CAN地址偏移 (HEX)	工作模式				复位模式	
	读		写		读	写
48	RX识别码2	RX识别码2	TX识别码2	TX识别码2	验收代码2	验收代码2
4C	RX数据1	RX识别码3	TX数据1	TX识别码3	验收代码3	验收代码3
50	RX数据2	RX识别码4	TX数据2	TX识别码4	验收屏蔽0	验收屏蔽0
54	RX数据3	RX数据1	TX数据3	TX数据1	验收屏蔽1	验收屏蔽1
58	RX数据4	RX数据2	TX数据4	TX数据2	验收屏蔽2	验收屏蔽2
5C	RX数据5	RX数据3	TX数据5	TX数据3	验收屏蔽3	验收屏蔽3
60	RX数据6	RX数据4	TX数据6	TX数据4	保留	—
64	RX数据7	RX数据5	TX数据7	TX数据5	保留	—
68	RX数据8	RX数据6	TX数据8	TX数据6	保留	—
6C	(FIFO RAM)	RX数据7	—	TX数据7	保留	—
70	(FIFO RAM)	RX数据8	—	TX数据8	保留	—
74	RX信息计数器		—		RX信息计数器	—
78	RX缓冲器起始地址		—		RX缓冲器起始地址	RX缓冲器起始地址
7C	时钟分频器		时钟分频器		时钟分频器	时钟分频器
80	内部RAM地址0 (FIFO)		—		内部RAM地址0	内部RAM地址0
84	内部RAM地址1 (FIFO)		—		内部RAM地址1	内部RAM地址1
...	...		...		...	...
17C	内部RAM地址63 (FIFO)		—		内部RAM地址63	内部RAM地址63
180	内部RAM地址64 (TX缓冲器)		—		内部RAM地址64	内部RAM地址64
	...		...		...	...
1B0	内部RAM地址76 (TX缓冲器)		—		内部RAM地址76	内部RAM地址76
1B4	内部RAM地址77 (空闲)		—		内部RAM地址77	内部RAM地址77
1B8	内部RAM地址78 (空闲)		—		内部RAM地址78	内部RAM地址78
1BC	内部RAM地址79 (空闲)		—		内部RAM地址79	内部RAM地址79
1C0	(00H)		—		(00H)	—
	...		...		...	...
1FC	(00H)		—		(00H)	—

### 24.5.3 发送处理

根据 CAN 协议规范，报文的传输由 CAN 控制器独立完成。微控制器设置标识符，数据长度和待发送数据；然后对命令寄存器的‘发送请求’位置‘1’，来请求发送。当 CAN 控制器正在发送报文时，发送缓冲器被写锁定。所以在防止一个新报文到发送缓冲器之前，微控制器必须检查状态寄存器的‘发送缓冲器状态’标志 (TBS)。

设置命令位 CMR.0 和 CMR.1 会立即产生一次信息发送，当发送错误或仲裁丢失时是不会重发的（单次发送）。只设置命令位 CMR.0 数据发送失败会重传。在自检测模式下，设置命令位 CMR.4 和 CMR.1 会立即产生一次自接收性质的信息发送。

### 中止

一个已经请求发送的报文，可以通过置位命令寄存器位的相应位执行‘中止发送’，通过对 CAN\_CMR 寄存器中的 AT 位置‘1’，可以中止发送请求。

当 CPU 需要当前请求发送等待时，例如：先发送一条比较紧急的信息时。但当前正在处理的传送是不停止。要想知道源信息是否成功发送，可以通过传送完毕状态位来查看。不过，这应在发送缓冲器状态位置‘1’或产生发送中断后。

要注意的是，即使因为发送缓冲器状态位变为‘释放’而使信息被中止，也会产生发送中断。

如果前一条指令中发送请求被置为‘1’，它不能通过设置发送请求位为‘0’来取消，而应通过中止发送位为‘0’取消。

### 24.5.4 接收管理

接收到的报文由 CAN 控制器独立完成。收到的报文放在接收缓冲器。可以发送给微控制器的报文，由状态寄存器的接收缓冲器状态标志‘RBS’和接收中断标志‘RI’标出。

- 查询控制接收

微控制器读 CAN 控制器的状态寄存器，检查接收缓冲状态（RBS）查看是否收到一个报文。

当读到 RBS 位为 1，表示收到一个或多个报文，微控制器从 CAN 中取得报文，然后置位命令寄存器的响应标志位‘RRB’发送一个释放接收缓冲器命令。

- 中断控制接收

中断使能标志位位于 CAN 控制器寄存器里（对于 BasicCAN 模式）或位于中断使能寄存器里（对于 PeliCAN 模式）。如果 CAN 控制器已接收一个报文，而且报文已经通过验收滤波器并放在接收 FIFO，那么会产生一个接收中断。进入中断服务程序，微控制器取走报文，然后置位命令寄存器的响应标志位‘RRB’发送一个释放接收缓冲器命令。

### 溢出

在接收 FIFO 满了但还接收其他的报文的时候就会导致溢出，同时置位状态寄存器中的数据超载状态位（如果使能）通知微控制器有数据溢出的情况，CMR.3 位置‘1’可清除溢出状态。

### 有效报文

根据 CAN 协议，当报文被正确接收（直到 EOF 域的最后一位都没有错误），且通过了标识符过滤，那么该报文被认为是有效报文。

CAN\_CMR 寄存器中的 RRB 位是用于清除由数据溢出状态位指出的数据溢出情况。

### 24.5.5 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者以广播的形式把报文发送给所有的接收者。节点在接收报文时一根据标识符的值一决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需，报文就被丢弃且无需软件的干预。

独立的 CAN 控制器装配了一个多功能的验收滤波器，该滤波器允许自动检查标识符和数据字节。使用这些有效的滤波方法，可以防止对于某个节点无效的报文或报文组存储在接收缓冲器里。因此降低了微控制器的处理负载。

滤波器由验收代码寄存器和屏蔽寄存器根据给定算法来控制。接收到的数据会和验收代码寄存器中的值进行逐位比较。接收屏蔽寄存器定义与比较相关的位置（0 = 相关，1 = 不相关）。只有收到报文的相应位于验收代码寄存器相应的位相同，报文才会被接收。

### BasicCAN 模式里的验收滤波器

该滤波器是由两个寄存器-验收码寄存器（ACR）和验收屏蔽寄存器（AMR）控制。CAN 报文标识符的高 8 位和这些寄存器里值相比较。可以定义若干个组的标识符为被任何一个节点接收。

例子：

验收码寄存器（ACR）包括：

验收码寄存器（ACR）包括：  
验收屏蔽寄存器（AMR）包括：  
带有11位标识符信息被接收  
(X=无关)

MSB	LSB						
0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	0
0	1	X	X	X	0	1	0

ID.10

在验收屏蔽寄存器里是‘1’的位置上，标识符相应的位可以是任意值。这对于三个最低位也一样。因此在这个例子里可以接收 64 个不同的标识符。标识符其他的位必须等于验收代码寄存器相应位的值。

### PeliCAN 模式里的验收滤波器

在验收滤波器的帮助下，只有当接收信息中的识别位和验收滤波器预定义的值相等时，CAN 控制器才允许将已接收信息存入 RXFIFO。

PeliCAN 模式中，验收滤波器由验收代码寄存器（ACRn）和验收屏蔽寄存器（AMRn）定义。要接收的信息的位模式在验收代码寄存器中定义。相应的验收屏蔽寄存器允许定义某些位为‘不影响’（即可为任意值）。

有两种不同的过滤模式可在模式寄存器中的位 3 选择：

- 单滤波器模式<sup>(1)</sup>
- 双滤波器模式<sup>(0)</sup>

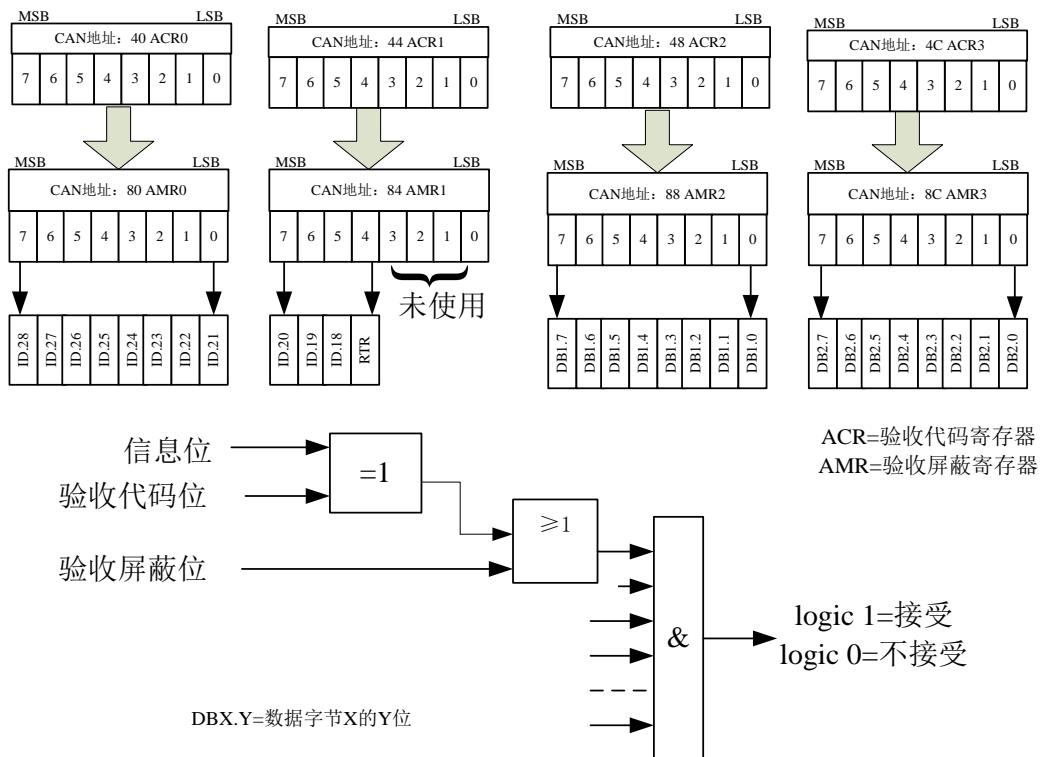
### 单滤波器配置

这种滤波器配置可以定义一个长滤波器（4 字节）。滤波器字节和信息字节之间位的对应关系取决于当前接收帧格式。

**标准帧：**如果接收的是标准帧格式的信息，在验收滤波中只使用前两个数据字节来存放包括 RTR 位的完整的识别码。如果由于置位 RTR 位而导致没有数据字节，或因为设置相应的数据长度代码而没有或只有一个数据字节，信息也会被接收的。对于一个成功接收的信息，所有单个位的比较后都必须发出接收信号。

注: ACR1 和 AMR1 的低四位是不用的, 为了和未来产品兼容, 这些位可通过设置 AMR1.3、AMR1.2、AMR1.1、AMR1.0 为 1 而定为 ‘不影响’。

图 177. 接收标准结构信息时的单个滤波器配置

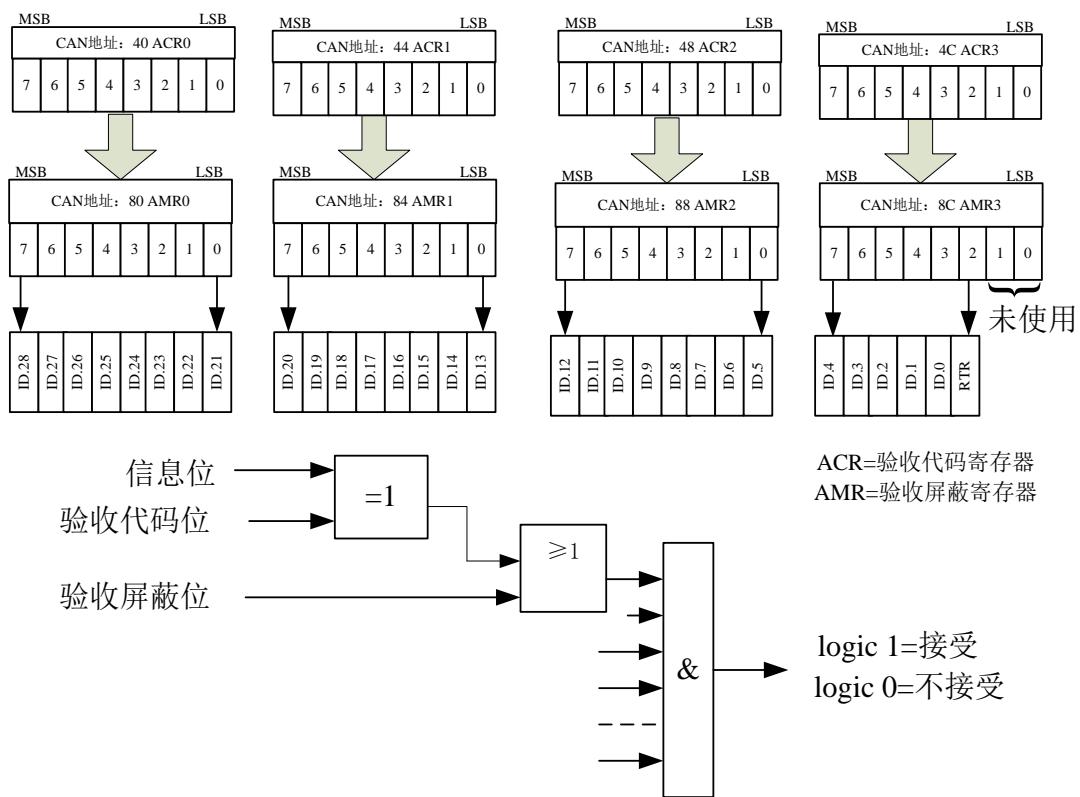


**扩展帧:** 如果接收的信息是扩展帧格式的, 包括 RTR 位的全部识别码将被接收过滤使用。

为了成功接收信息, 每个位的比较后都必须发出接收信号。

注: AMR3 的最低两位和 ACR3 是不用的。这些位应该通过置位 AMR3.1 和 AMR3.0 来定为 ‘不影响’。

图 178. 单滤波器配置，接收扩展帧信息



### 双滤波器的配置

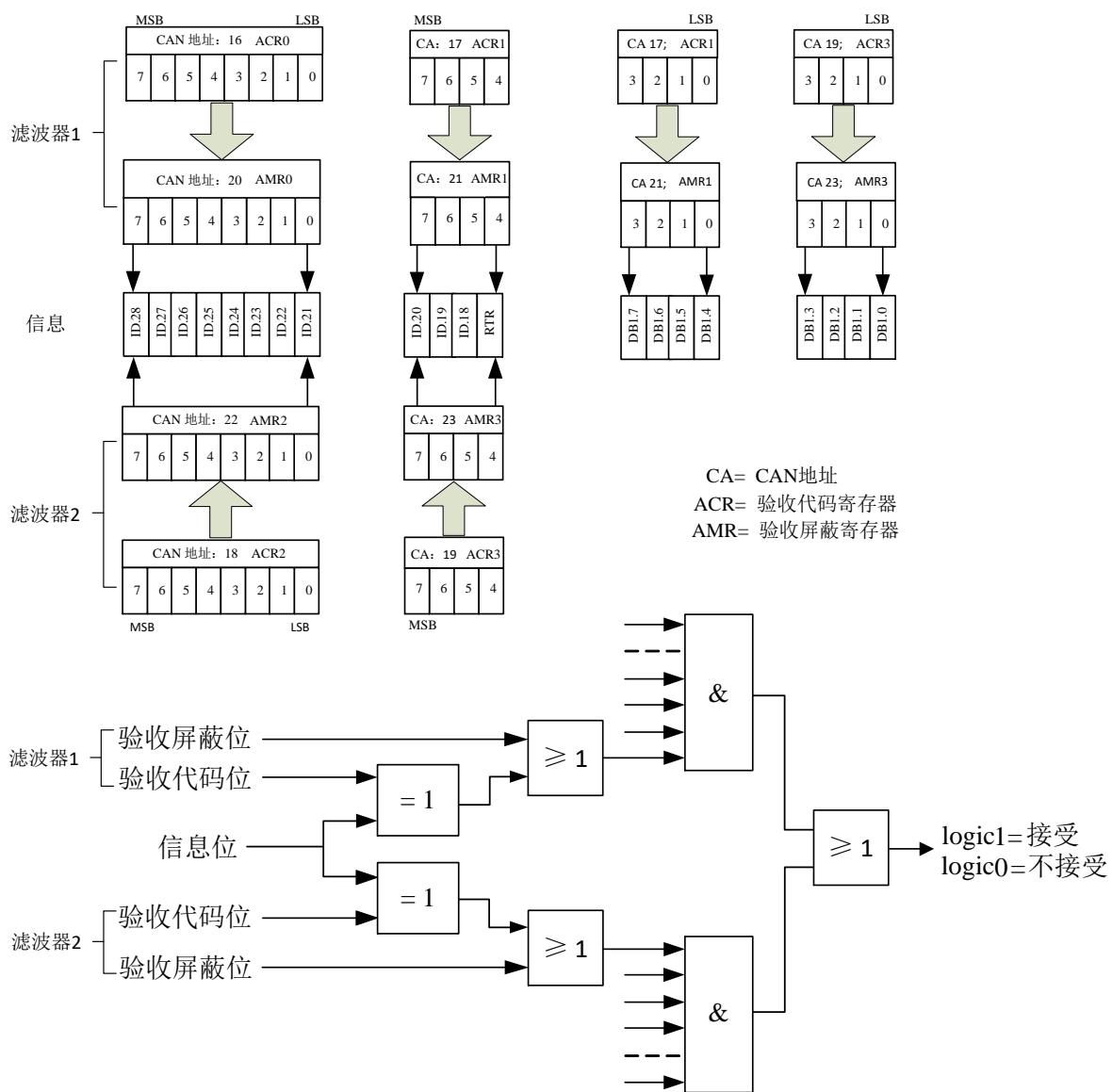
这种配置可以定义两个短滤波器。一条接收的信息要和两个滤波器比较来决定是否放入接收缓冲器中。至少有一个滤波器发出接收信号，接收的信息才有效。滤波器字节和信息字节之间位的对应关系取决于当前接收的帧格式。

**标准帧：**如果接收的是标准帧信息，被定义的两个滤波器是不一样的。第一个滤波器比较包括 RTR 位的整个标准识别码和信息的第一个数据字节。第二个滤波器只比较包括 RTR 位的整个标准识别码。

为了成功接收信息，所有单个位的比较时应至少有一个滤波器表示接收。RTR 位置位或数据长度代码是 0 时表示没有数据字节存在。无论怎样，只要从开始到 RTR 位的部分都被表示接收，信息就可以通过滤波器 1。

如果没有向滤波器请求数据字节过滤，AMR1 和 AMR3 的低四位必须被置为‘1’（不影响）。当使用包括 RTR 位的整个标准识别码时，两个滤波器都同样工作。

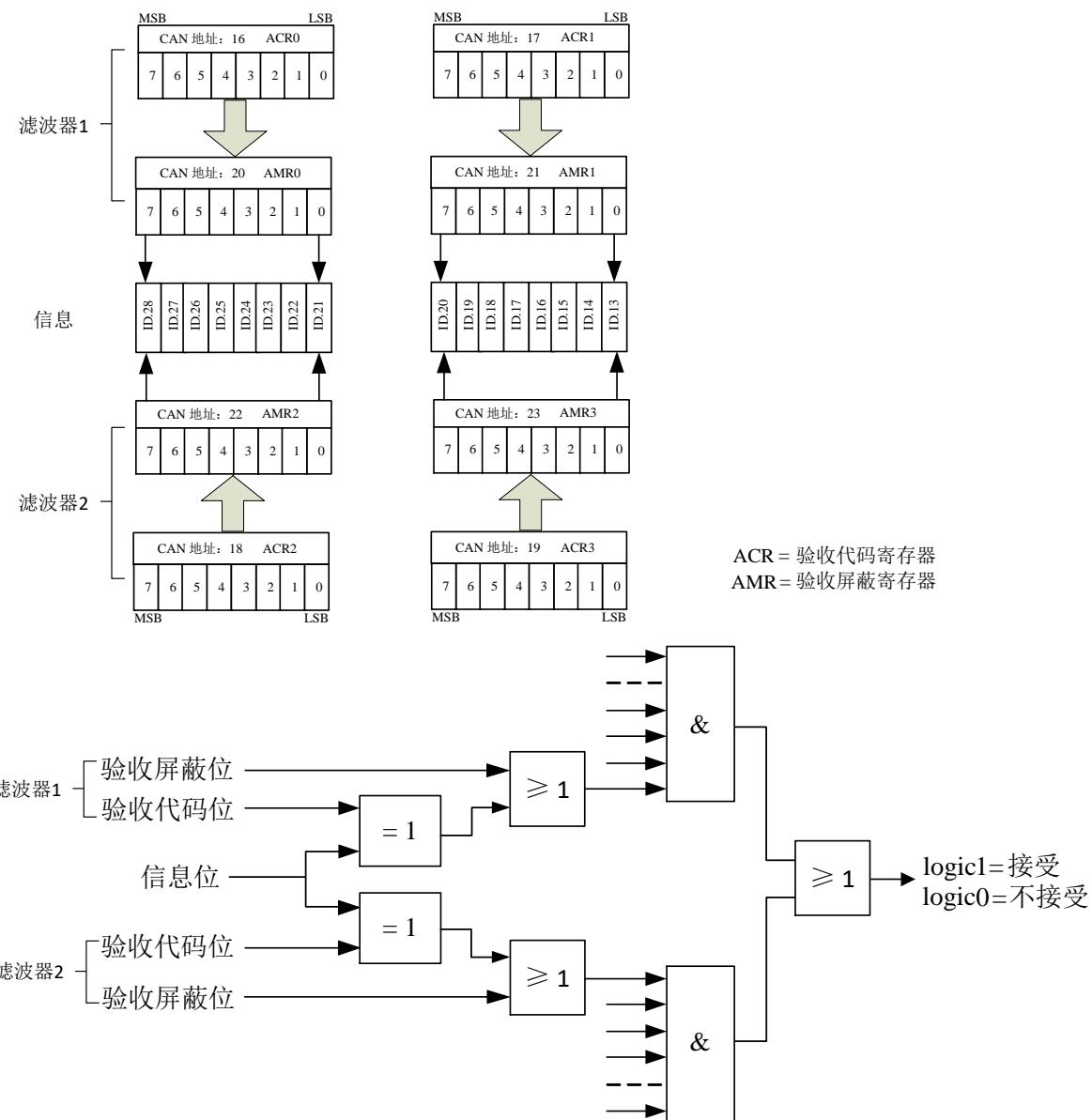
图 179. 接收标准结构信息时的双个滤波器配置



**扩展帧:** 如果接收到扩展帧信息, 定义的两个滤波器是相同的。两个滤波器都只比较扩展识别码的前两个字节。

为了能成功接收信息, 所有单个位的比较时至少有一个滤波器表示接收。

图 180. 双滤波器配置, 接收扩展帧信息



**例 1:** 假设 61 个标准帧报文要在 PeliCAN 模式里滤波, 可以通过使用一个长滤波器完成 (单滤波器模式)。

验收代码寄存器 (ACRn) 和验收屏蔽寄存器 (AMRn) 包括:

n	0	1 (高四位)	2	3
ACRn	01XX X010	XXXX	XXXX XXXX	XXXX XXXX
AMRn	0011 1000	1111	1111 1111	1111 1111
接收的报文 (ID28 ~ ID.18, RTR)	01xx x010    xxxx			

(‘X’ = 不相关, ‘x’ = 任意值, 只使用了 ACR1 和 AMR1 的高四位。)

**例 2:** 假设下面 2 个有标准帧标识符的报文在标识符不用进一步译码就被接收。数据和远程帧必须被正确接收。数据字节不要求验收滤波。

报文 1: (ID.28) 1011 1100 101 (ID.18)

报文 2: (ID.28) 1111 0100 101 (ID.18)

使用单滤波模式可以接收到四个报文而不仅是要求的两个:

n	0	1 (高四位)	2	3
ACRn	1X11 X100	101X	XXXX XXXX	XXXX XXXX
AMRn	0100 1000	0001	1111 1111	1111 1111
接收的报文 (ID.28 ~ ID.18, RTR)	1011 0100 101x 1111 0100 101x (报文2) 1011 1100 101x (报文1) 1111 1100 101x			

(‘X’ = 不相关, ‘x’ = 任意值, 只使用了 ACR1 和 AMR1 的高四位。)

这个结果不满足不进一步解码而接收两条信息的要求。

使用双滤波器可以得到正确的结果

n	滤波器1			滤波器2	
	0	1	3 低四位	2	3 高四位
ACRn	1011 1100	101X XXXX	... XXXX	1111 0100	101X ...
AMRn	0000 0000	0001 1111	... 1111	0000 0000	0001 ...
接收的信息 (ID.28 ~ ID.18, RTR)	1011 1100 101X (报文1)			0100 101X (报文2)	

(‘X’ = 不相关, ‘x’ = 任意值)

报文 1 被滤波器 1 接收, 报文 2 被滤波器 2 接收。如果报文至少被两个滤波器中的一个接收, 报文就被存放到接收 FIFO。这种方法可满足于这种要求。

**例 3:**

在这个例子里, 使用一个长的验收滤波器老板一组带有扩展帧标识符的报文

n	0	1	2	3 (高六位)
ACRn	1011 0100	1011 000X	1100 XXXX	0011 0XXX
AMRn	0000 0000	0001 0001	0000 1111	0000 0111
接收的信息 (ID.28 ~ ID.18, RTR)	1011 0100 1011 000X 1100 xxxx 0011 0X			

(‘X’ = 不相关, ‘x’ = 任意值, 只使用了 ACR1 和 AMR1 的高六位。)

#### 例 4:

有些使用标准帧系统仅用 11 位标识符和头两个数据字节识别报文。如果报文超过 8 个数据字节，头两个数据字节定义为报文头和使用分段存储协议就会使用像这样的协议。例如 DeviceNet。对于这种系统类型，CAN 控制器除了 11 位标识符和 RTR 位外，在单滤波器模式里能滤波两个数据字节，在双滤波器模式里能过滤一个数据字节（除了 11 位标识符和 RTR 位）。

下面的例子显示了用双滤波器模式，在这种系统里有效地滤波报文：

n	滤波器1			滤波器2	
	0	1	3 低四位	2	3 高四位
ACRn	1110 1011	0010 1111	... 1001	1111 0100	XXX0 ...
AMRn	0000 0000	0000 0000	... 0000	0000 0000	1110 ...
接收的信息 (ID.28 ~ ID.18, RTR)	1110 1011 0010 标识符 + RTR		1111 ... 1001 头一个数据字节	1111 0100 标识符	xxx0 RTR

('X' = 不相关, 'x' = 任意值)

- 滤波器 1 滤波的报文有：
  - 标识符‘11101011001’
  - RTR = ‘0’，也就是说是数据帧，以及
  - 数据字节‘11111001’（这是指例如 DeviceNet：一个信息的所有段都被过滤）。
- 滤波器 2 用来过滤一组 8 个报文，其中报文有：
  - 标识符‘11110100 000’到‘11110100111’，以及
  - RTR = ‘0’，也即是数据帧

#### 24.5.6 报文存储

要在 CAN 总线上发送的数据被载入 CAN 控制器的存储区，这个存储区叫‘发送缓冲器’。从 CAN 总线上收到的数据也存储在 CAN 控制器的存储区，这个存储区叫‘接收缓冲器’。这些缓冲器包括 2, 3 或 5 个字节的标识符和帧信息（取决于模式和帧类型），而最多可以包含 8 个数据字节。

#### BasicCAN 模式

缓冲器长达 10 个字节

- 2 个标识符字节
- 最多 8 个数据字节

### BasicCAN 模式里的 RX 和 TX 缓冲器

相对CAN偏移量		寄存器名		组成和注释
TX (16进制)	RX (16进制)	TX	RX	
28	50	CAN_TXIDR1	CAN_RXIDR1	8位标识符
2C	54	CAN_TXIDR2	CAN_RXIDR2	3位标识符, 1位远程传输请求位, 4位数据长度代码, 表示数据字节的数量
30	58	CAN_TXDR1	CAN_RXDR1	由数据长度代码志明, 最多8个数据字节
34	5C	CAN_TXDR2	CAN_RXDR2	
38	60	CAN_TXDR3	CAN_RXDR3	
3C	64	CAN_TXDR4	CAN_RXDR4	
40	68	CAN_TXDR5	CAN_RXDR5	
44	6C	CAN_TXDR6	CAN_RXDR6	
48	70	CAN_TXDR7	CAN_RXDR7	
4C	74	CAN_TXDR8	CAN_RXDR8	

### PeliCAN 模式

这些缓冲器是 13 个字节长

- 1 字节帧信息
- 2 个或 4 个标识符字节（标准帧或扩展帧）
- 最多 8 个数据字节

### 发送缓冲器

发送缓冲器的全部列表见下图。请务必分清标准帧格式（SFF）和扩展帧格式（EFF）配置。发送缓冲器允许定义长达 8 个数据字节发送信息。

图 181. 标准帧和扩展帧格式配置在发送缓冲器的列表

CAN地址	40	TX帧信息	CAN地址	40	TX帧信息
	44	TX识别码1		44	TX识别码1
	48	TX识别码2		48	TX识别码2
	4C	TX数据字节1		4C	TX识别码3
	50	TX数据字节2		50	TX识别码4
	54	TX数据字节3		54	TX数据字节1
	58	TX数据字节4		58	TX数据字节2
	5C	TX数据字节5		5C	TX数据字节3
	60	TX数据字节6		60	TX数据字节4
	64	TX数据字节7		64	TX数据字节5
	68	TX数据字节8		68	TX数据字节6
	6C	未使用		6C	TX数据字节7
	70	未使用		70	TX数据字节8

a.标准帧格式

b.扩展帧格式

帧信息中的 FF 位决定 CAN 控制器将要发送扩展帧格式还是标准帧格式。

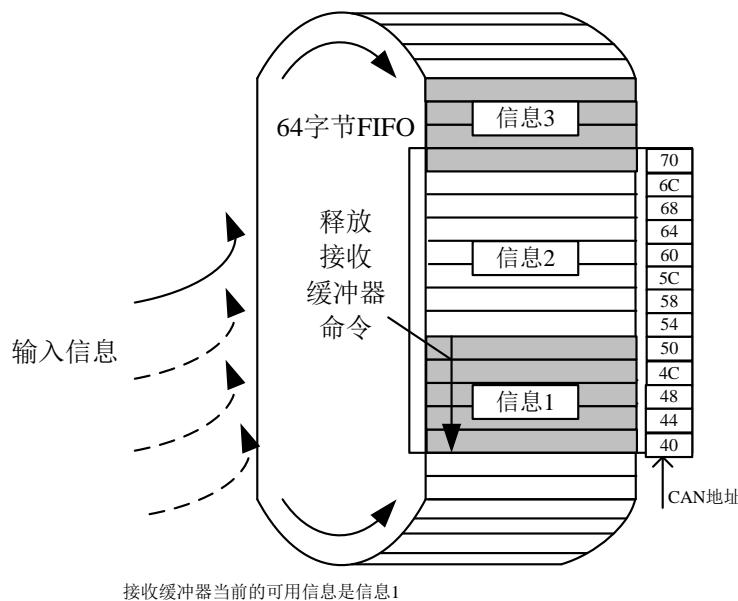
### 接收缓冲器

接收缓冲器的列表与发送缓冲器很相似。接收缓冲器是 RXFIFO 的可访问部分，位于 CAN 地址的 40 ~ 70.每条信息都非为描述区和数据区。

**注：**在帧信息字节中的接收字节长度代码代表实际发送的数据长度代码，它有可能大于 8（取决于发送器）。无论如何，最大接收数据字节数是 8。这一点在读接收缓冲器中的信息时应当考虑。

见下图，RXFIFO 共有 64 个信息字节的空间。一次可以存储多少条信息取决于数据的长度。如果 RXFIFO 中没有足够的空间来存储新的信息，CAN 控制器会产生数据溢出条件，此时信息有效且接收检测为肯定。发生数据溢出情况时，已部分写入 RXFIFO 的信息将被删除。这种情况可以通过状态寄存器和数据超限中断（中断允许）反应到 CPU。

图 182. RXFIFO 中的信息存储举例



#### 24.5.7 出错管理

基于错误计数器的值，每个 CAN 控制器能够在三种错误状态之一中工作：错误激活、错误认可或总线离线。如果错误计数器的值都在 0 ~ 127 之间，CAN 控制器是错误激活的。此时产生错误激活标志（6 个显性位）。如果一个错误计数器的值在 128 ~ 255 之间，CAN 控制器是错误认可的。此时，在检测到错误前，产生认可错误标志（6 个隐性位）。如果发送错误计数器的值高于 255，则到达总线离线状态。在这种状态下，自动置位复位请求，CAN 控制器对总线没有影响。总线离线状态只能在微控制器控制器用命令‘复位请求 = 0’退出。这将启动总线离线恢复定时器，发送错误计数器计数 128 个总线释放信号。计数结束后，两个错误计数器都是 0，器件再次处于错误激活状态。

##### 错误计数器

如上面描述，CAN 的错误状态和发送错误计数器和接收错误计数器的值直接相关。

为了仔细研究错误界定，支持 CAN 控制器的增强的错误分析功能，CAN 控制器提供可读的错误计数器。另外，在复位模式，允许对于两个错误计数器进行写访问。

##### 出错中断

有三个中断源来向微处理器发出错的状态。每个中断都能在中断使能寄存器里分别使能。

- 总线出错中断：

在 CAN 总线上检测到任何一个错误都会产生中断。

- 出错警告中断：

如果超过出错警告界限，产生出错警告中断。而且它在 CAN 控制器进入总线离线状态和再次之前再一次进入错误激活状态也会产生这个中断。CAN 控制器的出错警告界限在复位模式中可编程。复位后的默认值是 96。

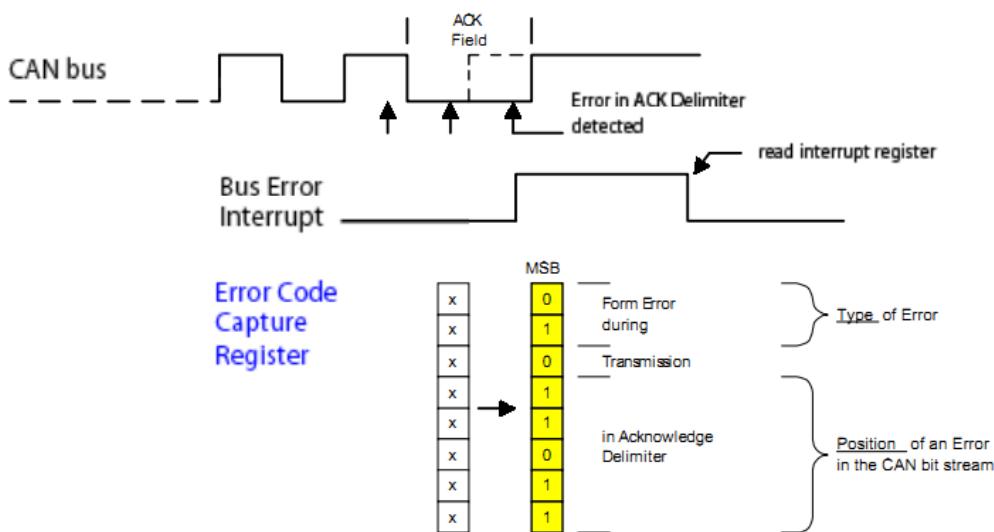
- 错误认可中断：

如果错误状态从错误激活变成错误认可或相反，将产生错误认可中断。

## 错误码捕捉

CAN 控制器可以执行在 CAN2.0B 规范定义的所有错误界定。每个 CAN 控制器处理错误的整个过程是完全自动的。但是，为了向用户提供某个错误的详细信息，CAN 控制器提供了错误代码捕捉功能。无论什么时候发生 CAN 总线错误，它都会强制产生相应的总线出错中断。同时，当前位的位置被捕获入错误代码捕捉寄存器。在主控制器将捕捉的数据读出前，它都会被保存在寄存器中。然后捕捉机制再次激活。寄存器可以内容区分四种错误类型：格式出错、填充出错、位出错和其他错误。如下图表示，寄存器还另外表明在错误是在报文的接收还是发送期间发生。这个寄存器中的五个位表示 CAN 帧内错误的位位置，更多信息参考下面的表和数据表。

图 183. 错误码捕捉功能举例



CAN 规范定义了：CAN 总线上的每个位只有特殊类型的错误。下面两张显示了 CAN 报文发送和接收期间可能出现的所有错误。左边的部分包括位置和错误的类型，这些由错误码捕捉寄存器捕捉。每张表的右边部分是将错误码转换成上层的错误描述，可以直接从寄存器内容知道其含义。通过使用这些表格，能得到有关错误计数器的变化和在器件发送和接收管脚的错误状态的更多信息。使用这些表时，例如在错误分析软件里，可以详细地分析每个错误状态。关于 CAN 错误类型和位置的信息能用于错误统计和系统维护或在系统优化器件进行纠正。

表 42. 接收时可能出现的错误

CAN位流里的错误位置	错误类型	RX错误计数	错误码捕捉	描述
标识符 SRR、IDE和RTR 位 保留位 数据长度码 数据场 CRC序列	填充	+ 1	收到5个电平相同的连续位	--
CRC定界符	格式 填充	+ 1 + 1	RX = 显性 收到超过5个电平相同的连续的位	位必须是隐性
应答位	位	+ 1	TX = 显性, 但RX = 隐性	不能写显性位
应答定界符	格式	+ 1	RX = 显性, 或 检测到CRC错误	临界的总线定时或总线长度 CRC序列不正确
帧结束	格式 其他	+ 1 ± 0	RX = 头六位是显性 RX = 最后一位的显性	-- 反应: 发出超载标志, 如果发送器重新发送, 数据可能重复
间隔	其他	± 0	RX = 显性	反应: 接收器发出超载标志
激活错误标志	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
容许的显性位	其他	+ 8	RX = 出错标志后的第一位是显性 RX = 出错或过载标志后有超过7位显性位	
错误定界符	格式 其他	+ 1 ± 0	RX = 头七位是显性位 RX = 定界符的最后一一位是显性位	-- 发送超载标志
超载标志	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位

表 43. 发送时可能出现的错误

CAN位流里的错误位置	错误类型	TX错误计数	错误码捕捉	描述
帧起始	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
标识符	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
	填充	± 0	TX = 隐性, 但RX = 显性	--
SRR位	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
	填充	± 0	TX = 隐性, 但RX = 显性	--
IDE和RTR位	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
	填充	+ 8	TX = 隐性, 但RX = 显性	--
保留位 数据长度码 数据场 CRC序列	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
CRC定界符	格式	+ 8	RX = 显性	位必须为隐性
应答隙	其他	+ 8	RX = 隐性 (错误激活)	没有应答
	其他	± 0	RX = 隐性 (错误认可)	没有应答, 节点可能单独在总线上
应答定界符	格式	+ 8	RX = 显性	临界的总线定时或总线长度
帧结束	格式	+ 8	RX = 头六位是显性位	--
	其他	+ 8	RX = 最后一位是显性位	帧已经被一些节点接收, 再次发送可能导致接收器里数据重复
间隔	其他	± 0	RX = 显性	来自于'旧'CAN控制器的超载标志
激活错误标志 过载标志	位	+ 8	TX = 显性, 但RX = 隐性	不能写显性位
允许显性位	格式	+ 8	RX = 在激活错误标志或过载标志后有超过7个显性位	----
错误定界符	格式	+ 8	RX = 头七位是显性位	----
	其他	± 0	RX = 定界符的最后一一位是显性位	----
认可错误标志	其他	+ 8	RX = 显性 (错误认可)	没有收到应答, 节点不是单独在总线上。

### 离线恢复

如果发送错误计数器的值高于 255, 则达到总线离线状态。总线状态位被置为'1'。在这种状态下, 自动置位复位请求位, CAN 控制器对总线没有影响。在错误中断允许的情况下, 会产生一个错误中断。这种状态会持续到 CPU 清除复位请求位。所有这些完成后, CAN 控制器将会等待协议规定的最长时间 (128 个总线空闲信号)。

## 24.5.8 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过跟帧起始位的边沿进行同步，及通过跟后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位的时间分为 3 段：

- 同步段 ( $t_{SYNCSEG}$ )：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元 ( $1 \times t_{CAN}$ )。
- 时间段 1 ( $t_{TSEG1}$ )：定义采样点的位置。它包含 CAN 标准里的 PROP\_SEG 和 PHASE\_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- 时间段 2 ( $t_{TSEG2}$ )：定义发送点的位置。它代表 CAN 标准里的 PHASE\_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

CAN 系统时钟  $t_{SCL}$  的周期是可编程的，而且决定了相应的位时序。

CAN 系统时钟由如下公式计算： $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ 。这里  $t_{CLK} = APB1$  的频率周期。

同步跳跃宽度 (SJW) 定义了，在每位中可以延长或缩短多少个时间单元的上限。为了补偿在不同总线控制器的时钟振荡器之间的相位偏移，任何总线控制器必须在当前传送的相关信号边沿重新同步。同步跳转宽度定义了每一位周期可以被重新同步缩短或延长的时钟周期的最大数目，

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

时间段 1 (TSEG1) 和时间段 2 (TSEG2) 决定了每一位的时钟数目和采样点的位置，这里：

$$t_{SYNCSEG} = 1 \times t_{SCL}$$

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

有效跳变被定义为，当 CAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。如果在时间段 1 ( $t_{TSEG1}$ ) 而不是在同步段 ( $t_{SYNCSEG}$ ) 检测到有效跳变，那么  $t_{TSEG1}$  的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2 ( $t_{TSEG2}$ ) 而不是在  $t_{SYNCSEG}$  检测到有效跳变，那么  $t_{TSEG2}$  的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器 (CAN\_BTR) 的设置，只能在 CAN 处于初始化状态下进行。

$$\text{CAN 波特率} = APB1 / (2^* (BRP + 1) * (TSEG1 + 1 + TSEG2 + 1 + 1)) ;$$

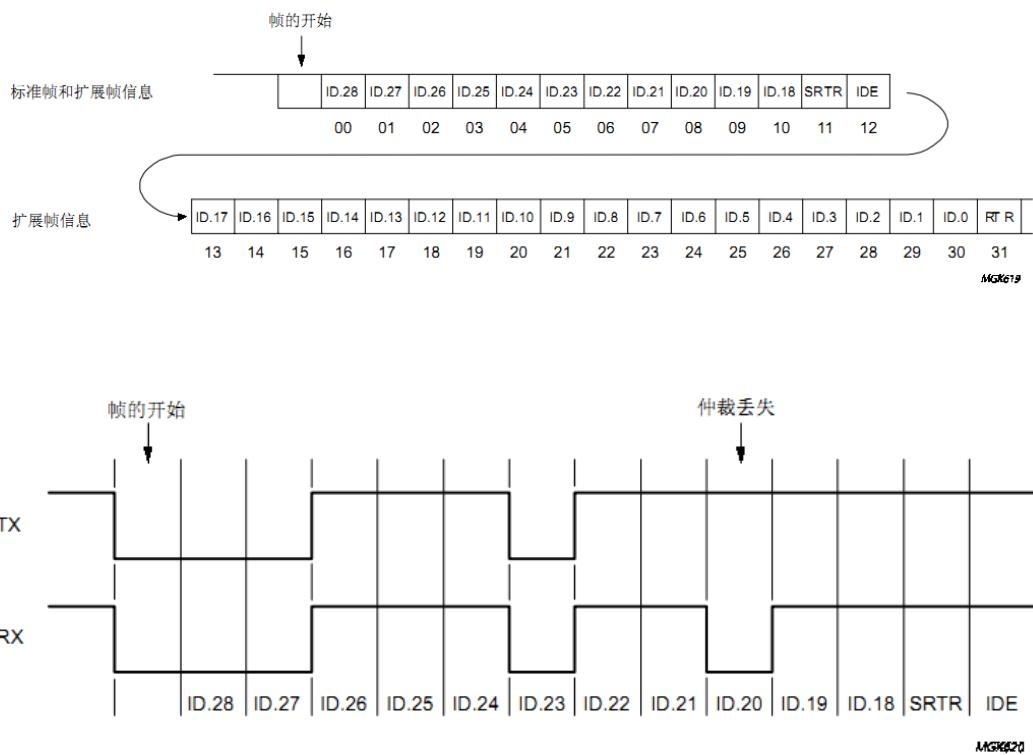
## 24.5.9 仲裁丢失

仲裁丢失时，会产生相应的仲裁丢失中断（中断允许）。同时，位流处理器的当前位置被捕获送入仲裁丢失捕捉寄存器。一直到用户通过软件读这个值，寄存器中的内容都不会变。随后，捕捉机制又被激活了。

读中断寄存器时，中断寄存器中相应的中断标志位被清除。直到仲裁丢失捕捉寄存器被读一次之后，新的仲裁丢失中断才有效。

下图为仲裁丢失位解释

图 184. 仲裁丢失解释举例



#### 24.5.10 CAN 中断

**BasicCAN 模式**里有 5 个中断:

- 接收中断

当接收 FIFO 不空和接收中断使能 (CAN\_CR 寄存器位 1) 时产生。CAN\_IR 寄存器 RI 位被置‘1’

- 发送中断

发送缓冲器状态从 0 变为 1 (释放) 和发送中断使能 (CAN\_CR 寄存器位 2) 时产生

- 错误中断

错误中断使能 (CAN\_CR 寄存器位 3) 时, 错误状态位或总线状态位的变化会置位此位

- 数据溢出中断

大概数据溢出中断使能位 (CAN\_CR 寄存器位 4) 被置‘1’时向数据溢出状态位‘0-1’跳变

- 唤醒中断

退出睡眠模式时产生该中断。

**PeliCAN 模式**里有 8 个不同的中断:

- 接收中断

接收 FIFO 不空且中断寄存器的 RIE 位被置位时产生中断

- 发送中断

发送缓冲器状态从‘0-1’ (释放) 跳变且中断寄存器的 TIE 位被置位时产生中断

- 错误报警中断

错误状态位和总线状态位的改变和中断寄存器的 **EIE** 位被置位时产生中断

- 数据溢出中断

数据溢出状态位有‘0-1’跳变且中断寄存器的 **DOIE** 位被置位时产生中断

- 唤醒中断

当 CAN 控制器在睡眠模式中检测到总线的活动且中断寄存器的 **WUIE** 位被置‘1’时产生中断

- 错误消极中断

当 CAN 控制器到达错误消极状态（至少一个错误计数器超过协议规定的值 127）或从错误消极状态又进入错误活动状态以及中断寄存器的 **EPIE** 位被置位时产生中断

- 仲裁丢失中断

当 CAN 控制器丢失仲裁，变为接收器和中断使能寄存器的 **ALIE** 为被置位时产生中断

- 总线错误中断

当 CAN 控制器检测到总线错误且中断使能寄存器中的 **BEIE** 被置位时产生中断

## 24.6 CAN 寄存器描述

### 24.6.1 CAN 模式寄存器 (CAN\_MOD)

仅 PeliCAN 模式

偏移地址: 0x000

复位值: 0x0001

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										<b>SM</b>	<b>AFM</b>	<b>STM</b>	<b>LOM</b>	<b>RM</b>	
rw      rw      rw      rw      rw															

位31: 5	保留, 读位3的值始终为‘1’。
位4	<b>SM:</b> 睡眠模式 (Sleep mode) 1: 睡眠; 没有CAN中断等待和总线活动时, CAN控制器进入睡眠模式 0: 唤醒; 从睡眠状态唤醒
位3	<b>AFM:</b> 验收滤波器模式 (Acceptance filter mode) 1: 单; 选择单个验收滤波器 (32位长度) 0: 双; 选择两个验收滤波器 (每个有16位激活)
位2	<b>STM:</b> 自检测模式 (Self test mode) 1: 自检测; 此模式可以检测所有节点, 没有任何活动的节点使用自接收命令; 即使没有应答, CAN控制器也会成功发送。 0: 正常模式; 成功发送时必需应答信号
位1	<b>LOM:</b> 只听模式 (Listen only mode) 1: 只听; 这种模式中, 即使成功接收信息, CAN控制器也不向总线发应答信号; 错误计数器停止在当前值。 0: 正常模式

位0	<b>RM:</b> 复位模式 (Reset mode) 1: 复位; 检测到复位模式位被置位, 中止目前正在接收/发送的信息, 进入复位模式。 0: 正常; 复位模式位接收到'1-0'的跳变后, CAN控制器回到工作模式。
----	--

#### 24.6.2 CAN 控制寄存器 (CAN\_CR)

仅 BasicCAN 模式:

偏移地址: 0x000

复位值: 0x0001

控制寄存器的内容是用于改变 CAN 控制器的行为的。这些位可以被微控制器设置或置位, 微控制器可以对控制寄存器进行读/写操作。

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															
rw      rw      rw      rw      rw      rw															

位31: 5	保留, 读位3的值始终为'1'。
位4	<b>OIE:</b> 溢出中断使能 (Overflow interrupt enable) 1: 使能; 如果置位数据溢出位, 微控制器接收溢出中断信号 0: 禁止; 微控制器不从CAN控制器接收溢出中断信号
位3	<b>EIE:</b> 错误中断使能 (Error interrupt enable) 1: 使能; 如果出错或总线状态改变, 微控制器接收错误中断信号 0: 禁止; 微控制器不从CAN控制器接收错误中断信号
位2	<b>TIE:</b> 发送中断使能 (Transmit interrupt enable) 1: 使能; 当信息被成功发送或发送缓冲器又被访问时 (例如, 中止发送命令后), 微控制器接收CAN控制器发出的一个发送中断信号 0: 禁止; 微控制器不从CAN控制器接收发送中断信号
位1	<b>RIE:</b> 接收中断使能 (Receive interrupt enable) 1: 使能; 信息被无错误接收时, CAN控制器发出的一个接收中断信号到微控制器 0: 禁止; 微控制器不从CAN控制器接收接收中断信号
位0	<b>RR:</b> 复位请求 (Reset request) 1: 当前; CAN控制器检测到复位请求后, 中止当前发送/接收的信息, 进入复位模式 0: 空缺; 复位请求位接收到一个下降沿后。CAN控制器回到工作模式

### 24.6.3 CAN 命令寄存器 (CAN\_CMR)

偏移地址: 0x004

复位值: 0x0000

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

位31: 5	保留。
位4	<p><b>BasicCAN模式:</b></p> <p><b>GTS:</b> 睡眠 (Go to sleep)            1: 睡眠; 如果没有 CAN 中断等待和总线活动, CAN控制器进入睡眠模式。            0: 唤醒; CAN控制器正常工作模式。</p> <p><b>PeliCAN模式:</b></p> <p><b>SRR:</b> 自接收请求 (Self reset request)            1: 当前; 信息可被同时发送和接收            0: (空缺)</p>
位3	<p><b>CDO:</b> 清除数据溢出 (Clear data overrun)            1: 清除; 清除数据溢出状态位            0: 无动作</p> <p>清除数据溢出这个命令位是用来清除由数据溢出状态位指出的数据溢出情况的。果数据溢出位被置位, 不会产生数据溢出中断了。释放接收缓冲器命令的同时是可以发出清除数据溢出命令的。</p>
位2	<p><b>RRB:</b> 释放接收缓冲器 (Release receive buffer)            1: 释放; 接收缓冲器中存放信息的内存空间将被释放            0: 无动作</p> <p>读接收缓冲器之后, 微控制器可以通过设置释放接收缓冲器位为1来释放RXFIFO中当前信息的内存空间。这可能会导致接收缓冲器中的另一条信息立即有效。这样会再产生一次接收中断 (使能条件下)。如果没有其它可用信息, 就不会再产生接收中断, 接收缓冲器状态位被清除。</p>
位1	<p><b>AT:</b> 中止传输 (Abort transmission)            1: 当前; 如果不是在处理过程中, 等待处理的发送请求将取消            0: 空缺; 无动作</p> <p>中止传送位是在CPU要求当前传送暂停时使用的, 例如, 传送一条紧急信息。正在进行的传送是不停止的。要查看原始信息是否被成功发送, 可以通过传送成功状态位来检测。不过, 这必须在发送缓冲器状态位为1 (释放) 或发送中断产生的情况下才能实现。</p>

位0	<b>TR:</b> 发送请求 (Transmission request)
	1: 当前; 信息被发送 0: 空缺; 无动作  如果发送请求在前面的命令中被置位。它就不可以通过直接设置为0来取消它了。不过, 可以通过设置中止发送位为0来取消。

#### 24.6.4 CAN 状态寄存器 (CAN\_SR)

偏移地址: 0x008

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								BS	ES	TS	RS	TCS	TBS	DOS	RBS
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
位7	<b>BS:</b> 总线状态 (Bus status) 1: 总线关闭; CAN控制器退出总线活动 0: 总线开启; CAN控制器加入总线活动  当传输错误计数器超过限制 (255) (总线状态位置'1' - 总线关闭), CAN控制器就会将复位请求位置'1' (当前), 在错误中断允许的情况下, 会产生一个错误中断。这种状态会持续直到CPU清除复位请求位。所有这些完成之后, CAN控制器将会等待协议规定的最小时 (128个总线空闲信号)。总线状态位被清除后 (总线开启), 错误状态位被置为 '0' (ok), 错误计数器复位且产生一个错误中断 (中断允许)
位6	<b>ES:</b> 出错状态 (Error status) 1: 出错; 至少出现一个错误计数器满或超过 CPU报警限制 0: ok; 两个错误计数器都在报警限制以下  根据CAN 2.0B协议说明, 在接收或发送时检测到错误会影响错误计数。当至少有一个错误计数器满或超出CPU警告限制 (96) 时, 错误状态位被置位。在允许情况下, 会产生错误中断。
位5	<b>TS:</b> 发送状态 (Transmit status) 1: 发送; CAN控制器在传送信息 0: 空闲; 没有要发送的信息  如果接收状态位和发送状态位都是0, 则CAN总线是空闲的。
位4	<b>RS:</b> 接收状态 (Receive status) 1: 接收; CAN控制器正在接收信息 0: 空闲; 没有正在接收的信息  如果接收状态位和发送状态位都是0, 则CAN总线是空闲的。

位3	<p><b>TCS:</b> 发送完毕状态 (Transmission complete status)</p> <p>1: 完毕; 最近一次发送请求被成功处理 0: 未完毕; 当前发送请求未处理完毕</p> <p>无论何时发送请求位被置为'1', 发送完毕位都会被置为'0' (未完毕)。发送完毕位的'0'会一直保持到信息被成功发送。</p>
位2	<p><b>TBS:</b> 发送缓冲器状态 (Transmit buffer status)</p> <p>1: 释放; CPU可以向发送缓冲器写信息 0: 锁定; CPU不能访问发送缓冲器; 有信息正在等待发送或正在发送</p> <p>如果CPU在发送缓冲器状态位是0 (锁定) 时试图写发送缓冲器, 则写入的字节被拒绝接收且会在无任何提示的情况下丢失。</p>
位1	<p><b>DOS:</b> 数据溢出状态 (Data overrun status)</p> <p>1: 溢出; 信息丢失, 因为RXFIFO中没有足够的空间来存储它 0: 空缺; 自从最后一次清除数据溢出命令执行无数据溢出发生</p> <p>当要被接收的信息成功的通过验收滤波器后 (例如, 仲裁之初), CAN控制器需要在RXFIFO中用一些空间来存储这条信息的描述符。因此必须有足够的空间来存储接收的每一个数据字节。如果没有足够的空间存储信息, 信息将会丢失且只向 CPU提示数据溢出情况。如果这个接收到的信息除了最后一位之外都无错误, 信息有效。</p>
位0	<p><b>RBS:</b> 接收缓冲器状态 (Receive buffer status)</p> <p>1: 满; RXFIFO中有可用信息 0: 空; 无可用信息</p> <p>在读 RXFIFO中的信息且用释放接收缓冲器命令来释放内存空间之后, 这一位被清除。如果FIFO中还有可用信息, 此位将在下一位的时限 (tSCL) 中被重新设置。</p>

#### 24.6.5 CAN 中断寄存器 (CAN\_IR)

偏移地址: 0x00C

复位值: 0x0000

中断寄存器允许中断源的识别。当寄存器的一位或多位被置位时中断就被激活了。中断寄存器对微控制器来说是只读存储器。

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								BIE	ALI	EPI	WUI	DOI	EI	TI	RI
r r r r r r r r															

位31: 8	保留, 读位7, 位6, 位5的值总是1。
--------	-----------------------

位7	<p><b>Basic模式:</b> 保留，读出值为1。</p> <p><b>PeliCAN模式:</b></p> <p><b>BEI:</b> 总线错误中断 (Bus error interrupt)</p> <p>1: 置位；当CAN控制器检测到总线错误且中断使能寄存器中的BEIE被置位时此位被置位 0: 复位</p>
位6	<p><b>Basic模式:</b> 保留，读出值为1</p> <p><b>PeliCAN模式:</b></p> <p><b>ALI:</b> 仲裁丢失中断 (Arbitration lost interrupt)</p> <p>1: 置位；当CAN控制器丢失仲裁，变为接收器和中断使能寄存器的ALIE为被置位时，此位被置位。 0: 复位</p>
位5	<p><b>Basic模式:</b> 保留，读出值为1。</p> <p><b>PeliCAN模式:</b></p> <p><b>EPI:</b> 错误消极中断 (Error passive interrupt)</p> <p>1: 置位；当CAN控制器到达错误消极状态（至少一个错误计数器超过协议规定的值127）或从错误消极状态又进入错误活动状态以及中断寄存器的EPIE位被置位时此位被置‘1’ 0: 复位</p>
位4	<p><b>WUI:</b> 唤醒中断 (Wake-up interrupt)</p> <p>1: 置位；退出睡眠模式时此位被置位 0: 复位；微控制器的任何读访问将清除此位</p> <p>如果当CAN控制器参与总线活动或CAN中断正在等待时，CPU试图进入睡眠模式，唤醒中断也会产生的。</p>
位3	<p><b>DOI:</b> 数据溢出中断 (Data overrun interrupt)</p> <p>1: 设置；当数据溢出中断使能位被置为‘1’时向数据溢出状态位‘0-1’跳变，此位被置位。 0: 复位；微控制器的任何读访问将清除此位</p> <p>溢出中断位（中断允许情况下）和溢出状态位是同时被置位的。</p>
位2	<p><b>EI:</b> 错误中断 (Error interrupt)</p> <p>1: 置位；错误中断使能时，错误状态位或总线状态位的变化会置位此位。 0: 复位；微控制器的任何读访问将清除此位</p>
位1	<p><b>TI:</b> 发送中断 (Transmit interrupt)</p> <p>1: 置位；发送缓冲器状态从0变为1（释放）和发送中断使能时，置位此位。 0: 复位；微控制器的任何读访问将清除此位</p>
位0	<p><b>RI:</b> 接收中断 (Receive interrupt)</p> <p>1: 置位；当接收FIFO不空和接收中断使能时置位此位 0: 复位；微控制器的任何读访问将清除此位</p> <p>接收中断位（中断允许时）和接收缓冲器状态位是同时置位的。 必须说明的是接收中断位在读的时候被清除，即使FIFO中还有其它可用信息。一旦释放接收缓冲器命令执行后，接收缓冲器中还有其它可用信息，接收中断（中断允许时）会在下一个tSCL被重置。</p>

### 24.6.6 CAN 中断使能寄存器 (CAN\_IER)

偏移地址: 0x010

复位值: 0x0000

仅存在 Pelican 模式

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								BEIE	ALIE	EPIE	WUIE	DOIE	EIE	TIE	RIE
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 5	保留。
位7	<b>BEIE:</b> 总线错误中断使能 (Bus error interrupt enable) 1: 使能; 如果检测到总线错误, 则CAN控制器请求相应的中断 0: 禁止
位6	<b>ALIE:</b> 仲裁丢失中断使能 (Arbitration lost interrupt enable) 1: 使能; 如果CAN控制器已丢失了仲裁, 则请求相应的中断。 0: 禁止
位5	<b>EPIE:</b> 错误消极中断使能 (Error passive interrupt enable) 1: 使能; 若CAN控制器的错误状态改变 (从消极到活动或反之), 则请求相应的中断 0: 禁止
位4	<b>WUIE:</b> 唤醒中断使能 (Wake-up interrupt enable) 1: 使能; 如果睡眠模式中的CAN控制器被唤醒, 则请求相应的中断。 0: 禁止
位3	<b>DOIE:</b> 数据溢出中断使能 (Data overrun interrupt enable) 1: 使能; 如果数据溢出状态位被置位 (见状态寄存器), CAN控制器请求相应的中断。 0: 禁止
位2	<b>EIE:</b> 错误报警中断使能 (Error interrupt enable) 1: 使能; 如果错误或总线状态改变 (见状态寄存器), CAN控制器请求相应的中断。 0: 禁止
位1	<b>TIE:</b> 发送中断使能 (Transmit interrupt enable) 1: 使能; 当信息被成功发送或发送缓冲器又可访问 (例如, 中止发送命令后) 时, CAN控制器请求相应的中断。 0: 禁止
位0	<b>RIE:</b> 接收中断使能 (Receive interrupt enable) 1: 使能; 当接收缓冲器状态是‘满’时, CAN控制器请求相应的中断。 0: 禁止

### 24.6.7 CAN 验收代码寄存器

BasicCAN 模式: CAN\_ACR

偏移地址: 0x010

复位值: 0x0000

在验收滤波器的帮助下, CAN 控制器能够允许 RXFIFO 只接收同识别码和验收滤波器中预设值相一致的信息。验收滤波器通过验收代码寄存器和验收屏蔽寄存器来定义。

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								AC							
rw								rw							

位31: 8	保留。
位7: 0	<b>AC[7: 0]:</b> (Acceptance code) 复位请求位被置高 (当前) 时, 这个寄存器是可以访问 (读/写) 的。如果一条信息通过了验收滤波器的测试而且接收缓冲器有空间, 那么描述符和数据将被分别顺次写入RXFIFO。当信息被正确的接收完毕就会: 接收状态位置高 (满) 接收中断使能位置高 (使能) 接收中断置高 (产生中断) 验收代码位 (AC.7-AC.0) 和信息识别码的高8位 (ID.10-ID.3) 相等, 且与验收屏蔽位 (AM.7-AM.0) 的相应位相或为1。即如果满足以下方程的描述, 则被接收: $[(ID.10-ID.3) \equiv (AC.7-AC.0)] \vee (AM.7-AM.0) \equiv 11111111$

PeliCAN 模式: 有四个验收代码寄存器分别是 CAN\_ACR0, CAN\_ACR1, CAN\_ACR2, CAN\_ACR3

CAN\_ACR0: 偏移地址: 0x040

复位值: 0x0000

CAN\_ACR1: 偏移地址: 0x044

复位值: 0x0000

CAN\_ACR2: 偏移地址: 0x048

复位值: 0x0000

CAN\_ACR3: 偏移地址: 0x04C

复位值: 0x0000

注: 详细说明见的标识符过滤中 *peliCAN* 模式介绍

#### 24.6.8 CAN 验收屏蔽寄存器

BasicCAN 模式:

偏移地址: 0x014

复位值: 0x0000

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

位31: 8	保留。
位7: 0	<b>AM[7: 0]:</b> (Acceptance mask) 如果复位请求位置高 (当前) 这个寄存器可以被访问 (读/写)。验收屏蔽寄存器定义验收代码寄存器的相应位对验收滤波器是‘相关的’或‘无影响的’ (即可为任意值)。

PeliCAN 模式: 有四个验收屏蔽寄存器分别是 CAN\_AMR0, CAN\_AMR1, CAN\_AMR2, CAN\_AMR3

CAN\_AMR0: 偏移地址: 0x050

复位值: 0x0000

CAN\_AMR1: 偏移地址: 0x054

复位值: 0x0000

CAN\_AMR2: 偏移地址: 0x058

复位值: 0x0000

CAN\_AMR3: 偏移地址: 0x05C

复位值: 0x0000

注: 详细说明见的标识符过滤中 *peliCAN* 模式介绍

#### 24.6.9 CAN 总线定时 0 (CAN\_BTR0)

偏移地址: 0x018

复位值: 0x0000

总线定时寄存器 0 定义了波特率预设值 (BRP) 和同步跳转宽度 (SJW) 的值。复位模式有效时这个寄存器是可以被访问 (读/写) 的。

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										SJW[1: 0]	BRP[5: 0]				
											rw	rw	rw	rw	rw

位31: 8	保留。
位7: 6	<b>SJW[1: 0]:</b> 同步跳转宽度 (Synchronization jump width) 为了补偿在不同总线控制器的时钟振荡器之间的相位偏移, 任何总线控制器必须在当前传送的相关信号边沿重新同步。同步跳转宽度定义了每一位周期可以被重新同步缩短或延长的时钟周期的最大数目。 $t_{SJW} = t_{SCL} \times (SJW + 1)$
位5: 0	<b>BRP[5: 0]:</b> 波特率预设值 (Baud rate prescaler) CAN系统时钟tSCL的周期是可编程的而且决定了相应的位时序。CAN系统时钟由如下公式计算 $t_{SCL} = 2 \times t_{CLK} \times (BRP + 1)$ 这里 $t_{CLK}$ = APB1的时钟周期。

#### 24.6.10 CAN 总线定时 1 (CAN\_BTR1)

偏移地址: 0x01C

复位值: 0x0000

总线定时寄存器 1 定义了每个位周期的长度、采样点的位置和在每个采样点的采样数目。在复位模式中, 这个寄存器可以被读/写访问。

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					SAM	TESG2[6: 4]					TESG1[3: 0]				
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
--------	-----

位7	<b>SAM:</b> 采样 (Sampling) 1: 三倍; 总线采样三次; 建议在低/中速总线 (A和B级) 上使用, 这对过滤总线上的毛刺波是有益的 0: 单倍; 总线采样一次; 建议使用在高速总线上 (SAE C级)
位6: 0	<b>TSEG2[6: 4], TSEG1[3: 0]:</b> 时间段 1 (Time segment 1) 和时间段2 (Time segment 2) TSEG1和TSEG2决定了每一位的时钟数目和采样点的位置, 这里: $t_{SYNCSEG} = 1 \times t_{SCL}$ $t_{TSEG1} = t_{SCL} \times (8 \times TSEG1.3 + 4 \times TSEG1.2 + 2 \times TSEG1.1 + TSEG1.0 + 1)$ $t_{TSEG2} = t_{SCL} \times (4 \times TSEG2.2 + 2 \times TSEG2.1 + TSEG2.0 + 1)$

#### 24.6.11 CAN 发送识别码寄存器 0 (CAN\_TXID0)

仅存在 BasicCAN 模式:

偏移地址: 0x028

复位值: 0x0000

发送识别码寄存器 0 定义发送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问。

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								ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
位7: 0	CAN 识别码10 ~ 3 (CAN identifier byte 10 ~ 3)

#### 24.6.12 CAN 发送识别码寄存器 1 (CAN\_TXID1)

仅存在 BasicCAN 模式:

偏移地址: 0x02C

复位值: 0x0000

发送识别码寄存器 1 定义发送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问。

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								ID.2	ID.1	ID.0	RTR	DLC3	DLC2	DLC1	DLC0
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
位7: 5	CAN识别码2 ~ 0 (CAN identifier byte 2 ~ 0)
位4	<b>RTR:</b> 帧格式 (Remote transmission request) 1: 远程; CAN将发送远程帧 0: 数据; CAN将发送数据帧
位3: 0	发送数据区长度0 ~ 8 (Data length code 0 ~ 8)

仅存在 **BasicCAN** 模式:

偏移地址: 0x030 ~ 0x04C

复位值: 0x0000

发送数据寄存器 CAN\_TXDRO ~ 7。仅在工作模式下这个寄存器可以被读/写访问。

接收缓冲与发送缓冲数据格式一致。

#### 24.6.13 CAN 仲裁丢失捕捉寄存器 (CAN\_ALC)

仅存在 **PeliCAN** 模式:

偏移地址: 0x02C

复位值: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留										BITNO4					
										rw	rw	rw	rw	rw	rw

位31: 5	保留, 读值为0。
位4: 0	<b>BITNO[4: 0]:</b> 值和功能参考下表 (Bit number) 仲裁丢失时, 会产生相应的仲裁丢失中断 (中断允许)。同时, 位流处理器的当前位置被捕获送入仲裁丢失捕捉寄存器。一直到用户通过软件读这个值, 寄存器中的内容都不会变。随后, 捕捉机制又被激活了。 读中断寄存器时, 中断寄存器中相应的中断标志位被清除。直到仲裁丢失捕捉寄存器被读一次之后, 新的仲裁丢失中断才有效。

表 44. 仲裁丢失捕捉寄存器的 bit 4 - bit 0 的功能

位					十进制值	功能
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	0	仲裁丢失在识别码的bit1
0	0	0	0	1	1	仲裁丢失在识别码的bit2
0	0	0	1	0	2	仲裁丢失在识别码的bit3
0	0	0	1	1	3	仲裁丢失在识别码的bit4
0	0	1	0	0	4	仲裁丢失在识别码的bit5
0	0	1	0	1	5	仲裁丢失在识别码的bit6
0	0	1	1	0	6	仲裁丢失在识别码的bit7
0	0	1	1	1	7	仲裁丢失在识别码的bit8
0	1	0	0	0	8	仲裁丢失在识别码的bit9
0	1	0	0	1	9	仲裁丢失在识别码的bit10
0	1	0	1	0	10	仲裁丢失在识别码的bit11
0	1	0	1	1	11	仲裁丢失在SRTR位; 注2
0	1	1	0	0	12	仲裁丢失在IDE位
0	1	1	0	1	13	仲裁丢失在识别码的bit12; 注3
0	1	1	1	0	14	仲裁丢失在识别码的bit13; 注3
0	1	1	1	1	15	仲裁丢失在识别码的bit14; 注3
1	0	0	0	0	16	仲裁丢失在识别码的bit15; 注3
1	0	0	0	1	17	仲裁丢失在识别码的bit16; 注3
1	0	0	1	0	18	仲裁丢失在识别码的bit17; 注3
1	0	0	1	1	19	仲裁丢失在识别码的bit18; 注3
0	1	1	0	1	13	仲裁丢失在识别码的bit12; 注3
0	1	1	1	0	14	仲裁丢失在识别码的bit13; 注3
0	1	1	1	1	15	仲裁丢失在识别码的bit14; 注3
1	0	0	0	0	16	仲裁丢失在识别码的bit15; 注3
1	0	0	0	1	17	仲裁丢失在识别码的bit16; 注3
1	0	0	1	0	18	仲裁丢失在识别码的bit17; 注3
1	0	0	1	1	19	仲裁丢失在识别码的bit18; 注3
1	0	1	0	0	20	仲裁丢失在识别码的bit19; 注3
1	0	1	0	1	21	仲裁丢失在识别码的bit20; 注3
1	0	1	1	0	22	仲裁丢失在识别码的bit21; 注3
1	0	1	1	1	23	仲裁丢失在识别码的bit22; 注3
1	1	0	0	0	24	仲裁丢失在识别码的bit23; 注3
1	1	0	0	1	25	仲裁丢失在识别码的bit24; 注3
1	1	0	1	0	26	仲裁丢失在识别码的bit25; 注3
1	1	0	1	1	27	仲裁丢失在识别码的bit26; 注3
1	1	1	0	0	28	仲裁丢失在识别码的bit27; 注3

位					十进制值	功能
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
1	1	1	0	1	29	仲裁丢失在识别码的bit28; 注3
1	1	1	1	0	30	仲裁丢失在识别码的bit29; 注3
1	1	1	1	1	31	仲裁丢失在RTR位; 注3

注：仲裁丢失的二进制编码结构位的号码。

标准帧信息的 RTR 位。

只使用于扩展帧信息。

#### 24.6.14 CAN 错误代码捕捉 (CAN\_ECC)

仅存在 PeliCAN 模式：

偏移地址：0x030

复位值：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								ERRC1	DIR	SEG					
								r	r	r	r	r	r	r	r

位31: 8	保留，读值为0。
位7: 6	<b>ERRC[1: 0]:</b> 错误代码 (Error code) 00: 位错 01: 格式错 10: 填充错 11: 其它错误
位5	<b>DIR:</b> 方向 (Direction) 1: RX; 接收时发生的错误 0: TX; 发送时发生的错误

位4: 0	<b>ECC[4: 0]: 段 (Error code capture)</b> 00010: ID.28-ID.21 00011: 帧开始 00100: SRTR位 00101: IDE位 00110: ID.20-ID.18 00111: ID.17-ID.13 01000: CRC序列 01001: 保留位0 01010: 数据区 01011: 数据长度代码 01100: RTR位 01101: 保留位1 01110: ID.4-ID.0 01111: ID.12-ID.5 10001: 活动错误标志 10010: 中止 10011: 支配 (控制) 位误差 10110: 消极错误标志 10111: 错误定义符 11000: CRC定义符 11001: 应答通道 11010: 帧结束 11011: 应答定义符 11100: 溢出标志 其他: 保留
-------	--

注: 位的设置反映了当前结构段的不同错误事件。

总线发生错误时被迫产生相应的错误中断 (中断允许时)。同时, 位流处理器的当前位置被捕捉送入错误代码捕捉寄存器。其内容直到用户通过软件读出时都是不变的。读出后, 捕捉机制又被激活了。访问中断寄存器期间, 中断寄存器中相应的中断标志位被清除。新的总线中断直到捕捉寄存器被读出一次才可能有效。

#### 24.6.15 CAN 错误报警限制寄存器 (CAN\_EWLR)

仅存在 PeliCAN 模式:

偏移地址: 0x034

复位值: 0x0096

错误报警限制在这个寄存器中被定义。默认值 (复位值) 是 96。复位模式中, 此寄存器对 CPU 来说是可读/写的。工作模式中是只读的。

注: 只有之前进入复位模式, **EWLR** 才有可能被改变。直到复位模式被再次取消后, 才有可能发生错误状态的改变 (见状态寄存器) 和由新的寄存器内容引起的错误报警中断。

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								EWL							

位31: 8	保留。
位7: 0	<b>EWL[7: 0]:</b> 可编程的错误限制报警 (Programmable error warning limit ) 当只要一个错误计数器超过错误限制编程值，错误状态位被置位。

#### 24.6.16 CAN RX 错误计数寄存器 (CAN\_RXERR)

仅存在 PeliCAN 模式:

偏移地址: 0x038

复位值: 0x0000

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

位31: 8	保留。
位7: 0	<b>RXERR[7: 0]:</b> RX错误计数寄存器 (RX error counter register) 反应了接收错误计数器的当前值。硬件复位后寄存器被初始化为0。工作模式中，对CPU来说是只读的。只有在复位模式中才可以写访问此寄存器。 如果发生总线关闭，RX错误计数器就被初始化为0。总线关闭期间，写这个寄存器是无效的。 注意：只有之前进入复位模式，才有可能由CPU迫使RX错误计数器发生改变。直到复位模式被取消后，错误状态的改变（见状态寄存器）、错误报警和由新的寄存器内容引起的错误中断才可能有效。

#### 24.6.17 CAN TX 错误计数寄存器 (CAN\_TXERR)

仅存在 Pelican 模式:

偏移地址: 0x03C

复位值: 0x0000

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

位31: 8	保留。
	<b>TXERR[7: 0]: TX 错误计数寄存器 (TX error counter register)</b> 反映了发送错误计数器的当前值。 工作模式中, 这个寄存器对CPU是只读内存。复位模式中才可以写访问这个寄存器。硬件复位后, 寄存器被初始化为0。如果总线关闭, TX错误计数器被初始化为127来计算总线定义的最长时间 (128个总线空闲信号)。这段时间里读TX错误计数器将反映出总线关闭恢复的状态信息。 如果总线关闭是激活的, 写访问TXERR的0 ~ 254单元会清除总线关闭标志, 复位模式被清除后控制器会等待一个11位的连续隐藏 (弱势) 位 (总线空闲)。 向TXERR写入255会初始化CPU驱动的总线关闭事件。只有之前进入复位模式, 才有可能发生CPU引起的TX错误计数器内容的改变。直到复位模式被再次取消, 错误或总线状态的改变 (见状态寄存器)、错误报警和由新的寄存器内容引起的错误中断才有可能有效。离开复位模式后, 就象总线错误引起的一样, 给出新的TX计数器内容且总线关闭被同样的执行。这意味着重新进入复位模式, TX错误计数器被初始化到127, RX计数器被清0, 所有的相关状态和中断寄存器位被置位。 复位模式的清除将会执行协议规定的总线关闭恢复序列 (等待128个总线空闲信号)。 如果在总线关闭恢复 (TXERR > 0) 之前又进入复位模式, 总线关闭保持有效且TXERR被锁定。
位7: 0	

#### 24.6.18 CAN 发送帧信息寄存器 (CAN\_SFF)

仅存在 Pelican 模式:

偏移地址: 0x040

复位值: 0x0000

发送帧信息寄存器设置送帧的类型与数据长度。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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								FF	RTR	X	X	DLC.3	DLC.2	DLC.1	DLC.0
								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
位7	<b>FF:</b> 帧格式 (Frame format) 1: EFF; CAN将发送/接收扩展帧格式 0: SFF; CAN将发送/接收标准帧格式
位6	<b>RTR:</b> 帧格式 (Remote transmission request) 1: 远程; CAN将发送/接收远程帧 0: 数据; CAN将发送/接收数据帧
位5: 4	保留。
位3: 0	<b>DLC:</b> 数据区长度 (Data length code bit) 发送数据区长度0 ~ 8。

#### 24.6.19 CAN 发送识别码寄存器 0 (CAN\_TXID0)

仅存在 Pelican 模式:

偏移地址: 0x044

复位值: 0x0000

发送识别码寄存器 0 设置帧的标识符。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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								ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
								rw							

位31: 8	保留。
--------	-----

位7: 0	<b>IDx:</b> CAN标识符ID28 ~ ID21 (Identifier bit 28-21) 标准帧时，与ID20 ~ ID18组成11位标识符。
-------	--

#### 24.6.20 CAN 发送识别码寄存器 1 (CAN\_TXID1)

仅存在 PeliCAN 模式:

偏移地址: 0x048

复位值: 0x0000

发送识别码寄存器 1 设置帧的标识符。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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								ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
rw								rw							

位31: 8	保留。
位7: 5	<b>IDx:</b> CAN标识符ID20 ~ ID18 (Identifier bit 20 -18)
位40:  位40:	标准帧 无意义 扩展帧  <b>IDx:</b> CAN标识符ID17 ~ ID13 (Identifier bit 17-13)

#### 24.6.21 CAN 发送数据寄存器 0 (CAN\_TXDATA0)

仅存在 PeliCAN 模式:

偏移地址: 0x04C

复位值: 0x0000

发送数据寄存器 0，用于 CAN 数据发送存储。仅在工作模式下这个寄存器可以被读/写访问，写时为发送寄存器，读时为接收寄存器，数据结构相同。

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								DATA0							
rw								rw	rw	rw	rw	rw	rw	rw	rw

位31: 8	保留。
--------	-----

位7: 0	标准帧 <b>DATA0:</b> CAN发送数据0 (CAN transmit data 0) 扩展帧 ID12 ~ 5
-------	--

#### 24.6.22 CAN 发送数据寄存器 1 (CAN\_TXDATA1)

仅存在 PeliCAN 模式:

偏移地址: 0x050

复位值: 0x0000

发送数据寄存器 1, 用于 CAN 数据发送存储。仅在工作模式下这个寄存器可以被读/写访问, 写时为发送寄存器, 读时为接收寄存器, 数据结构相同。

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								DATA1							
rw								rw							

位31: 8	保留。
位7: 0	标准帧 <b>DATA1:</b> CAN发送数据1 (CAN transmit data 1) 扩展帧 ID4 ~ 0, 低3位无意义

仅存在 PeliCAN 模式:

偏移地址: 0x054 ~ 0x070

复位值: 0x0000

以上偏移地址均为 CAN 数据寄存器, 扩展帧时 CAN 发送数据 0 储存在 DATA2 余下数据依次类推。这些寄存器写时为发送寄存器, 读时为接收寄存器, 数据结构相同。

#### 24.6.23 CAN 时钟分频寄存器 (CAN\_CDR)

偏移地址: 0x07C

复位值: 0x0000

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

位31: 8	保留。
位7	<b>MODE:</b> CAN模式 (CAN mode) 如果MODE是0, CAN控制器工作于BasicCAN模式。否则, CAN控制器工作于PeliCAN模式。只有在复位模式中是可以写的。
位6: 0	保留。

## 25. USB 全速设备接口

### 25.1 USB 介绍

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。

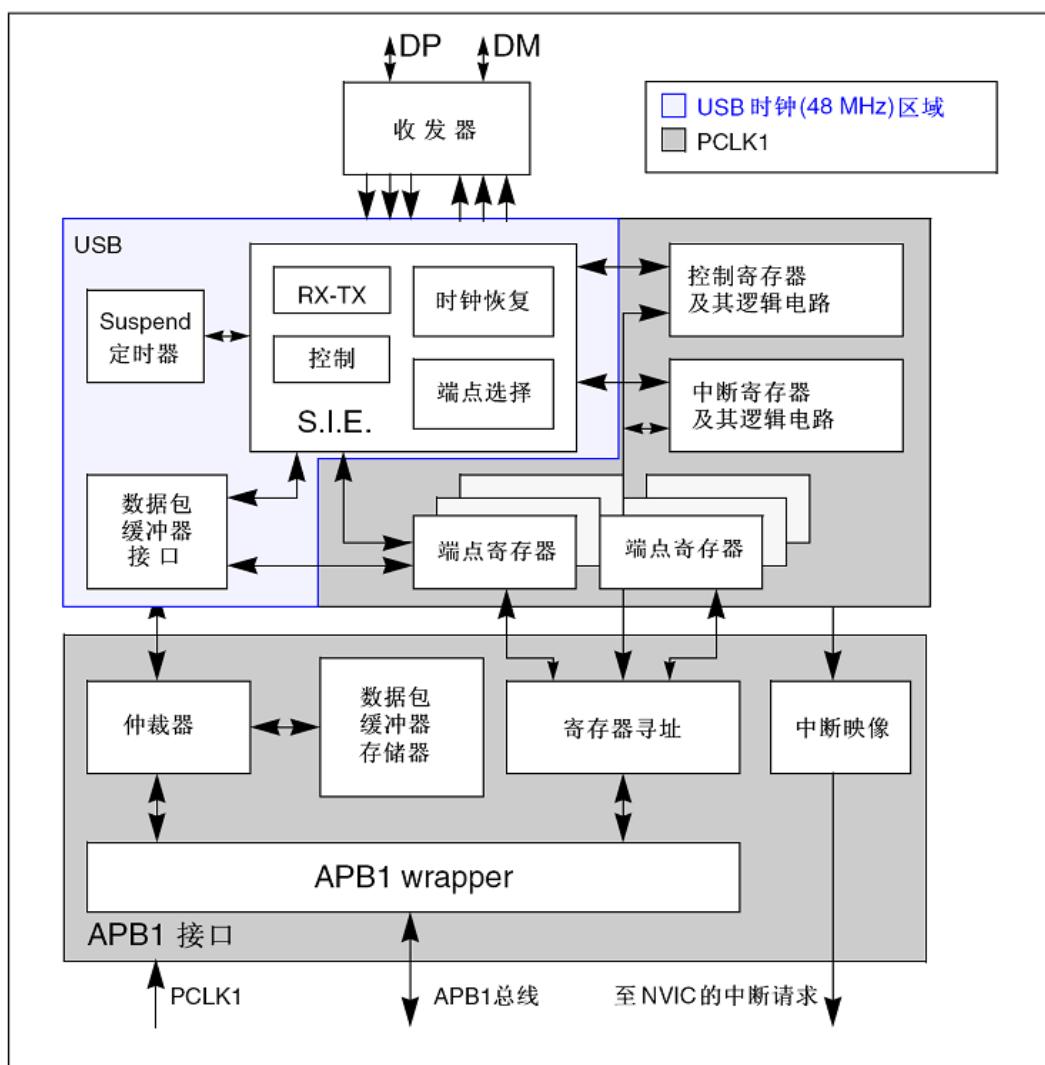
USB 外设支持 USB 挂起/恢复操作，可以停止设备时钟实现低功耗。

### 25.2 USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 支持全速 12M 模式和低速 1.5M 模式。
- 一个控制传输端点和四个独立的通用端点用于中断传输和批量传输。
- 控制、批量以及中断传输最大可传输 64 字节的包。
- 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 USB 挂起/恢复操作
- 支持 DMA 传输

下图是 USB 外设的方框图

图 185. USB 方框图



## 25.3 USB 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过数据缓冲区来完成的，USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输的格式由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个 64 字节缓冲区描述块，且该缓冲区是在 USB 模块内部，不能直接被 CPU 访问的。当 USB 模块识别出一个有效的功能/端点的令牌分组时，(如果需要传输数据并且端点已配置)随之发生相关的数据传输。USB 模块通过一个内部的寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 主机请求的传输类型
- 哪个端点需要得到服务
- 产生如正在进行的是哪种类型的传输
- 端点的应答
- 传输是否完成

在任何不需要使用 USB 模块的时候，通过写 USB\_POWER 寄存器总可以使 USB 模块置于低功耗模式 (SUSPEND 模式)。在这种模式下，不产生任何动态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块，也可以通过软件设置直接唤醒 USB 系统，还可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常时钟系统，并支持直接启动或停止时钟系统。

### 25.3.1 USB 功能模块描述

USB 模块包含 8 位宽的 320 字节大小的 FIFO，每个端点有 64 字节 FIFO。在 APB1 总线上，每个寄存器或者存储数据使用总线 32 位宽的低 8 位。

USB 模块包含一下几种寄存器：

- USB 寄存器。用于配置 USB 参数以及查询状态
- 端点状态寄存器
- 端点设置寄存器
- Setup 数据寄存器，存储 SETUP 包的数据。
- 端点数据控制寄存器，控制数据流。

APB1 总线或者 DMA 往数据端口写数据或者读数据，FIFO 数据个数就会增加或者递减。只有在端点接收和发送数据成功后，FIFO 指针才会变化。每个端点有一个 FIFO 的数据寄存器，作为写入或者读出 FIFO 的入口地址。每个端点还有专门的寄存器可以查询当前 FIFO 内有效数据个数。

只有端点 1 和端点 2 支持 DMA 传输。

## 25.4 编程中需要考虑的问题

下面的章节中，将介绍 USB 模块和应用程序之间的交互过程，有利于简化应用程序的开发。

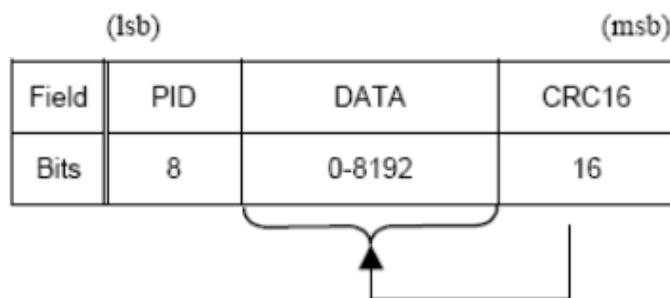
### 25.4.1 USB 传输概述

下面显示了包、事物以及传输三者之间的关系。

包（Packet）是 USB 系统中信息传输的基本单元，所有数据都是经过打包后在总线上传输的。

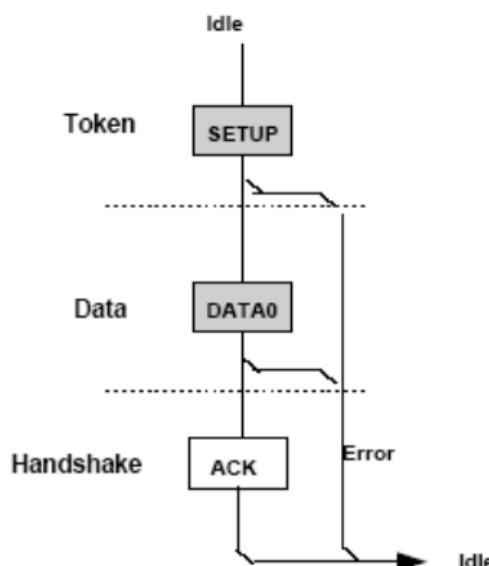
基本的 USB 包如下所示，如令牌包，数据包以及握手包。

图 186. 包的基本格式



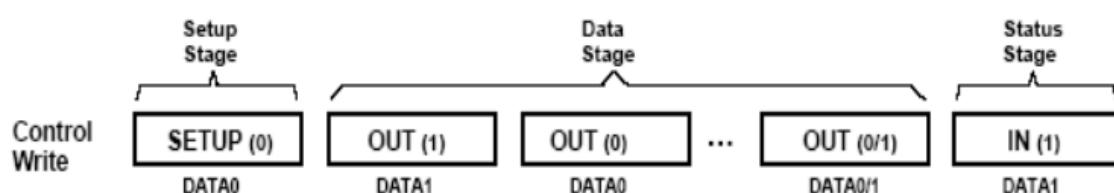
在 USB 上的数据信息的一次接收或发送的处理过程称为事务（Transaction）。事务处理的类型包括输入（IN）事务处理、输出（OUT）事务处理、设置（SETUP）事务处理等。

图 187. USB 事务



下图描述了一个端点的完整的传输过程。对于批量/控制传输，一个传输必须在上一次传输结束结束后开始。

图 188. USB 传输



当 USB 控制器开始工作，USB 处于 IDLE 状态，然后等待总线命令。当接收到一个总线命令，如复位、设置等，就会产生一个中断，CPU 查询该命令类型。例如，如果命令是复位，CPU 就会清零和复位所有可编程的状态。如果是传输请求命令，CPU 就会写\读 USB 控制器 FIFO 中的数据。对于输入的批量传输或者控制传输，当 CPU 或者 DMA 准备好数据后，CPU 会发送一个 ACK 握手信号或者 STALL 握手信号来结束传输。对于批量传输输出，当数据放入到相应的 FIFO 后，USB 会自动发送 ACK 信号。

传输中当数据大小超过最大包的大小时，会将数据分割成超过一个包传输，否则只会传输一个包。

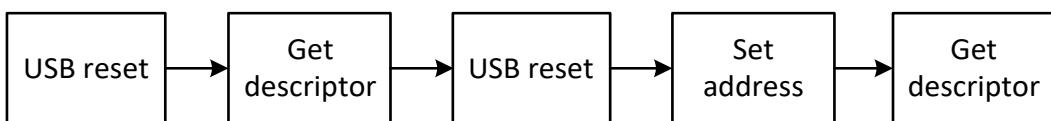
#### 25.4.2 USB 枚举

主机以控制传输的方式，通过端点 0 对设备发送各种请求，设备收到主机发来的请求后恢复相应的信息，进行枚举操作。

系统上电复位后，首先配置 USB 控制器：

1. 设置端点使能位。
2. 打开复位和端点中断
3. 打开连接位（USB\_TOP 寄存器的 CONNECT 位）

图 189. 枚举过程



当端点 0 接收到 SETUP 包，系统将会读取设置的 8 个字节数据从而决定下个步骤。对于 SetAddress 描述符，USB 控制器将会自动载入新地址。

枚举过程中，第一个来回的分析。

检测到设备，主机发总线复位。这个复位与 USB 上电复位和系统复位是不同的。这个是 SIE 根据总线状态通知用户的一种复位。设备产生复位中断，如何处理由设备固件程序决定。

主机发起第一个控制传输：

- (1) 主机 SETUP 包（发往地址 0 端点 0）、主机数据包（请求设备描述符）、设备握手包 ACK。

设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在端点 0 输入缓冲区准备好设备描述符。

- (2) 数据过程，主机先发一个 IN 令牌包、设备发一个数据包（这个数据已经准备好，SIE 收到 IN 令牌后，直接送到总线上，用户此时不干预）、主机发 ACK 包。

此时 SIE 产生端点 0 数据输入中断，表明主机已经取走了设备所准备的数据，用户也可以在该中断处理程序中作自己的处理。（SIE 指串行接口引擎，是所有 USB 控制器内部的“核心”。SIE 负责处理底层协议，如填充位，CRC 生成和校验，并可发出错误报告。SIE 的主要任务是将低级信号转换成字节，以供控制器使用）

此时，主机只接受一次数据，最少 8 个字节。如果用户数据没有发完，又在控制端点输入缓冲区，准备了数据，主机也不理会。

- (3) 状态过程：主机发 OUT 包（通知设备要输出）、主机发 0 字节状态数据包（这个是 0 字节，表明自己收到设备描述符）、设备发握手 ACK 包。

此时设备不会产生端点 0 数据输出中断，此时没有数据。

枚举过程中，第二个来回：设置地址。

第一个来回成功以后，主机再次复位总线。进入地址设置控制传输阶段。

(1) 主机 SETUP 包（发往地址 0 端点 0）、主机数据包（请求设置地址）、设备握手包 ACK。所以 SETUP 包后面都会跟一个表明主机 SETUP 目的的数据包，要么 GET，要么 SET。

设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在根据主机发来的地址写入自己的地址控制寄存器。

(2) 数据过程，本次传输没有数据。

(3) 状态过程：主机发 IN 包（通知设备要返回数据）、设备发 0 字节状态数据包（表明地址设置已经成功）、主机握手手 ACK 包（地址设置已经生效）。

此时设备不会产生端点 0 数据输入中断，此时没有数据。

枚举过程中，第三个来回：主机使用新地址获取完整的设备描述符。

主机采用新地址发起第一个控制传输：

(1) 主机 SETUP 包（发往新的地址端点 0）、主机数据包（请求设备描述符）、设备握手包 ACK。

设备产生端点 0 数据输出中断，固件程序要根据数据包中的主机要求做好准备，这里是在端点 0 输入缓冲区准备好设备描述符。

(2) 数据过程，主机先发一个 IN 令牌包、设备发一个数据包（这个数据已经准备好，SIE 收到 IN 令牌后，直接送到总线上，用户此时不干预）、主机发 ACK 包。

此时 SIE 产生端点 0 数据输入中断，表明主机已经取走了设备所准备的数据，用户可以该中断处理程序中要做如下处理：如果一次没有将描述符送完，要再次将剩下的内容填充端点 0 输入缓冲区。

第二次数据传输：主机再发一个 IN 令牌包、设备发一个数据包、主机发 ACK 包。

此时 SIE 再次产生端点 0 数据输入中断，如果数据已经发完了。这里就不处理了。进入状态过程。

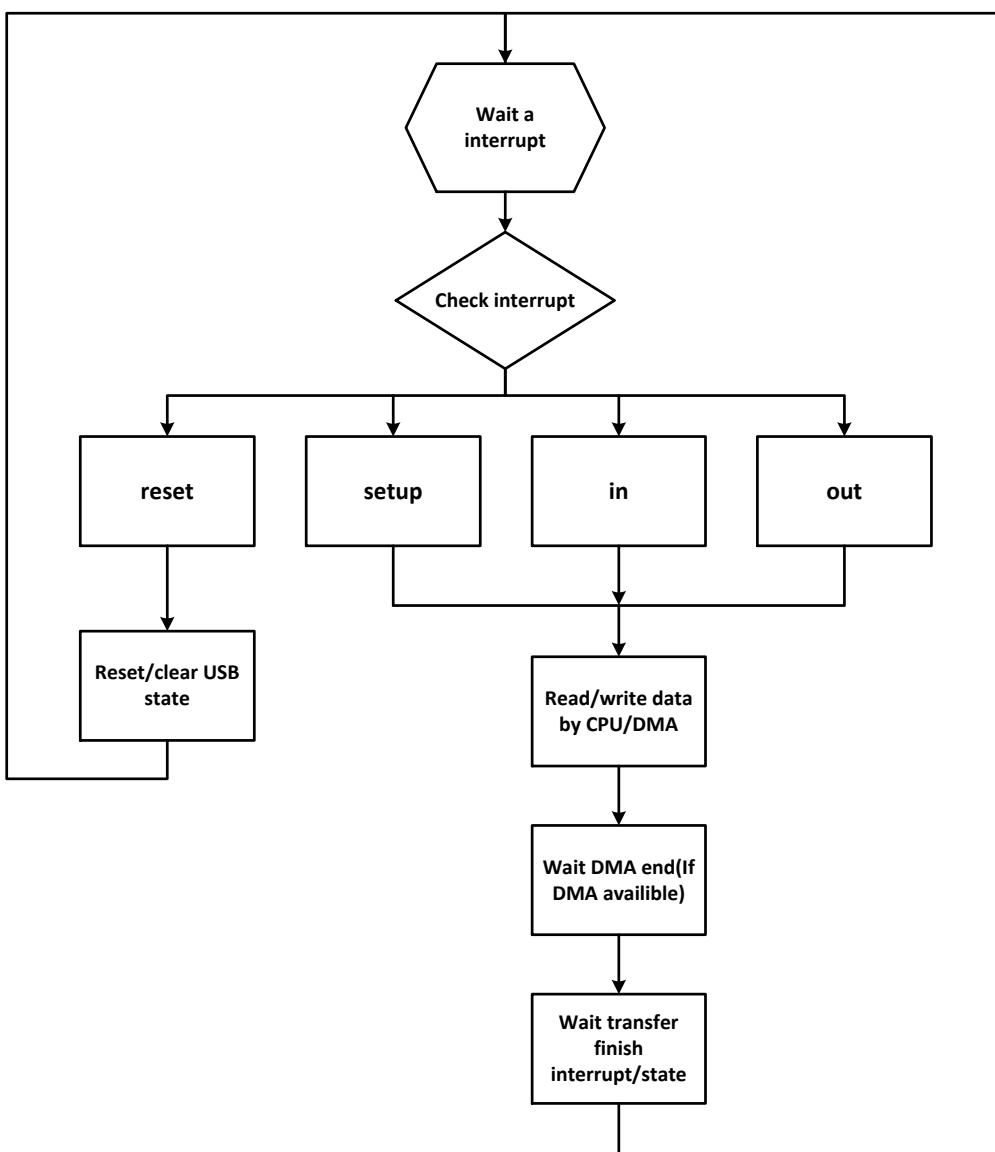
(3) 状态过程：主机发 OUT 包（通知设备要输出）、主机发 0 字节状态数据包（表明自己收到设备描述符）、设备发握手手 ACK 包。

### 25.4.3 USB 传输处理

系统需要设置一个无操作的循环来等待 USB 主机的请求。USB 主机的请求会设置一个中断位，系统通过不断查询中断位或者进入中断服务程序从而跳出循环。当系统检测到中断位时，跳到相应的子程序中处理。

下图是一个典型的 USB 传输的流程图：

图 190. USB 传输流程图



#### 25.4.4 IN 令牌包

当接收到一个主机发过来的 IN 请求时，如果收到 NACK 应答，USB 主机会重发 IN 请求直到该端点确认请求有效。如果接收到的地址和一个配置好的端点地址相符合的话，可以按照下面步骤：

1. 如果 FIFO 中的数据小于寄存器 EPX\_CTRL[6: 0]中设置的大小，或者 EPX\_CTRL[7]未设为 1，USB 模块会自动发送 NACK，直到数据准备好。
2. CPU 收到 IN\_NACK 状态。
3. CPU 把数据写入 FIFO。
4. CPU 在 EPX\_CTRL 寄存器设置待发送的数据大小和发送使能。
5. USB 模块在收到下一个 IN 请求的时候自动发送 FIFO 中的数据发送。最后一个数据字节发送完成后，自动发送计算好的 CRC。
6. CPU 会收到 IN\_ACK 状态，并且在发送结束后收到 END 状态。
7. 如果端点未使能，或者 EP\_HALT 寄存器设置暂停，USB 模块会应答 IN\_STALL 而不发送数据。

### 25.4.5 OUT 令牌包

当接收到一个主机发过来的 OUT 请求时, 如果收到 NACK 应答, USB 主机会重发 OUT 请求直到该端点确认请求有效。如果接收到的地址和一个配置好的端点地址相符合的话, 可以按照下面步骤:

1. 如果 FIFO 中的空间小于主机发过来的包的大小, USB 模块会自动发送 NACK, 直到 CPU 把数据从 FIFO 中取走。
2. CPU 会收到 OUT\_ACK 状态。
3. CPU 从 FIFO 读数据。
4. 如果端点未使能, 或者 EP\_HALT 寄存器设置暂停, USB 模块会应答 OUT\_STALL 而不发送数据。

## 25.5 USB 寄存器描述

### 25.5.1 USB TOP 寄存器(USB\_TOP)

地址偏移: 0x00

复位值: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							保留		ACTIV E	DP/DM STATE	SUSP END	RESE T	保留	CONN ECT	SPEE D

rw      r      r      r      rw      rw      rw      rw

位15: 8	保留, 始终读为0
位7	<b>ACTIVE:</b> USB总线活跃状态 (USB bus is active) 0: USB总线不活跃 1: USB总线活跃
位6: 5	<b>DP/DM STATE:</b> USB DP/DM线当前状态 (Current USB DP/DM line state)
位4	<b>SUSPEND:</b> USB SUSPEND状态 (USB suspend state) 该位监控控制器状态与APB_POWER寄存器无关 0: 控制器处于工作状态 1: 控制器处于挂起状态
位3	<b>RESET:</b> 复位USB控制器的端点和FIFO (Reset EP and FIFO in USB controller) 0: 不复位 1: 复位 注意, 该位置位后需要软件清零。
位2	保留
位1	<b>CONNECT:</b> USB连接状态 (USB connection) 0: 断开连接 1: 连接
位0	<b>SPEED:</b> 设置USB速率 0: 全速传输 1: 低速传输

### 25.5.2 USB 中断状态寄存器(USB\_INT\_STATE)

地址偏移: 0x04

复位值: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留												EPINTF	SOFF	RESUMF	SUNPENDF	RSTF
r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1											

位15: 7	保留, 始终读为0
位6: 5	保留
位4	<b>EPINTF:</b> 端点中断标志位 (EP interrupt received) 任何一个端点产生中断时, 该位被置1。具体参考EP_INT_STATE描述。该位只读。
位3	<b>SOFF:</b> SOF检测标志位 (BUS received) 当SOF被检测到时, 该位被置1。写1清除。
位2	<b>RESUMF:</b> 唤醒标志位 (BUS resume received) 当USB总线被激活, 该位被置1。写1清除。
位1	<b>SUSPENDF:</b> USB总线挂起标志位 (BUS suspend received) 当检测到总线上挂起状态时, 该位被置1。写1清除。
位0	<b>RSTF:</b> USB总线复位请求标志位 (BUS reset received) 当总线的复位信号输入被检测到, 该位被置1。写1清除。

### 25.5.3 USB 端点中断状态寄存器(EP\_INT\_STATE)

地址偏移: 0x08

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
保留												EP4F	EP3F	EP2F	EP1F	EP0F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位15: 7	保留, 始终读为0
位4	<b>EP4F:</b> 端点4中断标志位 (EP4 interrupt received)
位3	<b>EP3F:</b> 端点3中断标志位 (EP3 interrupt received)
位2	<b>EP2F:</b> 端点2中断标志位 (EP2 interrupt received)
位1	<b>EP1F:</b> 端点1中断标志位 (EP1 interrupt received)
位0	<b>EP0F:</b> 端点0中断标志位 (EP0 interrupt received)

### 25.5.4 USB 端点 0 中断状态寄存器(EP0\_INT\_STATE)

地址偏移: 0x0C

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								OUT-S TALLF	OUT-A CKF	OUT-NACK F	IN-ST ALLF	IN-AC KF	IN-NA CKF	END	SETU PF
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位15: 8	保留, 始终读为0
位7	<b>OUT-STALL:</b> OUT包应答STALL标识 (OUT-STALL received) 端点接收来自主机的OUT包后, 控制器应答STALL。在寄存器EP_HALTI[0]设置为1且EP0_CTRL[7]使能时, USB控制器会应答STALL。 写1清除。
位6	<b>OUT-ACK:</b> OUT包应答ACK标识 (OUT-ACK received) 端点0接收来自主机的OUT包后, FIFO有足够的空间, 主机完成当前传输。USB控制器自动应答ACK。 写1清除。
位5	<b>OUT-NACK:</b> OUT包应答NACK标识 (OUT-NACK received) 端点接收来自主机的OUT包后, 但此时没有足够空间存储来自主机发送的数据。USB控制器自动应答NACK。 写1清除。
位4	<b>IN-STALL:</b> IN包应答STALL标识 (IN-STALL received) 端点接收来自主机的IN包后, 控制器应答STALL。在寄存器EP_HALTI[0]设置为1且EP0_CTRL[7]使能时, USB控制器会应答STALL。 写1清除。
位3	<b>IN-ACK:</b> IN包应答ACK标识 (IN-ACK received) 端点接收来自主机的IN包后, FIFO中有足够的数据, 主机完成当前传输。USB控制器自动应答ACK。 写1清除。 端点接收来自主机的IN包后主机完成当前传输置位该位。
位2	<b>IN-NACK:</b> IN包应答NACK标识 (IN-NACK received) 端点接收来自主机的IN包后, 但此时没有足够数据完成当前传输。USB控制器自动应答NACK。 写1清除。
位1	<b>END:</b> 传输完成标识 (Status stage finished) 端点传输完成时, 该位被置1。写1清除。
位0	<b>SETUP:</b> 接收到SETUP包标识 (SETUP packet received) 置位后可以从寄存器SETUP0~7读取SETUP包的内容。 写1清除。

### 25.5.5 USB 中断使能寄存器(USB\_INT\_EN)

地址偏移: 0x10

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											EPIE	SOFIE	RESUME	SUNP	RSTIE
保留								保留							

rw      rw      rw      rw      rw      rw

位15: 8	保留, 始终读为0
位7: 5	保留
位4	<b>EPINTIE:</b> 端点中断使能位 (EP interrupt enable)
位3	<b>SOFIE:</b> SOF检测中断使能位 (SOF interrupt enable)
位2	<b>RESUMIE:</b> 唤醒中断使能位 (BUS resume interrupt enable)
位1	<b>SUSPENDIE:</b> USB总线挂起中断使能位 (BUS suspend interrupt enable)
位0	<b>RSTIE:</b> USB总线复位中断使能位 (BUS reset interrupt enable)

### 25.5.6 USB 端点中断使能寄存器(EP\_INT\_EN)

地址偏移: 0x14

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											EP4IE	EP3IE	EP2IE	EP1IE	EP0IE
保留															

rc\_w1   rc\_w1   rc\_w1   rc\_w1   rc\_w1

位15: 5	保留, 始终读为0
位4	<b>EP4IE:</b> 端点4中断使能位 (EP4 interrupt enable)
位3	<b>EP3IE:</b> 端点3中断使能位 (EP3 interrupt enable)
位2	<b>EP2IE:</b> 端点2中断使能位 (EP2 interrupt enable)
位1	<b>EP1IE:</b> 端点1中断使能位 (EP1 interrupt enable)
位0	<b>EP0IE:</b> 端点0中断使能位 (EP0 interrupt enable)

### 25.5.7 USB 端点 0 中断使能寄存器 (EP0\_INT\_EN)

地址偏移: 0x18

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OUT-STALLIE	OUT-ACKIE	OUT-NACKIE	IN-STALLIE	IN-ACKIE	IN-NACKIE	ENDIE	SETUPIE
保留								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位15: 8	保留, 始终读为0
位7	<b>OUT-STALLIE:</b> OUT包应答STALL中断使能位 (OUT-STALL interrupt enable)
位6	<b>OUT-ACKIE:</b> OUT包应答ACK中断使能位 (OUT-ACK interrupt enable)
位5	<b>OUT-NACKIE:</b> OUT包应答NACK中断使能位 (OUT-NACK interrupt enable)
位4	<b>IN-STALLIE:</b> IN包应答STALL中断使能位 (IN-STALL interrupt enable)
位3	<b>IN-ACKIE:</b> IN包应答ACK中断使能位 (IN-ACK interrupt enable)
位2	<b>IN-NACKIE:</b> IN包应答NACK中断使能位 (IN-NACK interrupt enable)
位1	<b>ENDIE:</b> 传输完成中断使能位 (Status stage finished interrupt enable)
位0	<b>SETUPIE:</b> 接收到SETUP包中断使能位 (SETUP packet interrupt enable)

### 25.5.8 USB 端点 X 中断状态寄存器 (EPX\_INT\_STATE) (X=1~4)

地址偏移: 0x20~0x2C

复位值: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OUT-STALLF	OUT-ACKF	OUT-NACKF	IN-STALLF	IN-ACKF	IN-NACKF	END	保留
保留								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

位15: 8	保留, 始终读为0
位7	<b>OUT-STALL:</b> OUT包应答STALL标识 (OUT-STALL received) 端点接收来自主机的OUT包后, 控制器应答STALL。在寄存器EP_HALT[0]设置为1且EP0_CTRL[7]使能时, USB控制器会应答STALL。 写1清除。
位6	<b>OUT-ACK:</b> OUT包应答ACK标识 (OUT-ACK received) 端点接收来自主机的OUT包后, FIFO有足够的空间, 主机完成当前传输。USB控制器自动应答ACK。 写1清除。

位5	<b>OUT-NACK:</b> OUT包应答NACK标识 (OUT-NACK received) 端点接收来自主机的OUT包后，但此时没有足够空间存储来自主机发送的数据。USB控制器自动应答NACK。 写1清除。
位4	<b>IN-STALL:</b> IN包应答STALL标识 (IN-STALL received) 端点接收来自主机的IN包后，控制器应答STALL。在寄存器EP_HALT[0]设置为1且EP0_CTRL[7]使能时，USB控制器会应答STALL。 写1清除。
位3	<b>IN-ACK:</b> IN包应答ACK标识 (IN-ACK received) 端点接收来自主机的IN包后，FIFO中有足够的数据，主机完成当前传输。USB控制器自动应答ACK。 写1清除。 端点接收来自主机的IN包后主机完成当前传输置位该位。
位2	<b>IN-NACK:</b> IN包应答NACK标识 (IN-NACK received) 端点接收来自主机的IN包后，但此时没有足够数据完成当前传输。USB控制器自动应答NACK。 写1清除。
位1	<b>END:</b> 传输完成标识 (Transfer finished) 端点传输完成时，该位被置1。写1清除。
位0	保留

### 25.5.9 USB 端点 X 中断使能寄存器(EPX\_INT\_EN) (X=1~4)

地址偏移: 0x40 到 0x4C

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OUT-STALLIE 保留	OUT-ACKIE 保留	OUT-NACKIE 保留	IN-STALLIE 保留	IN-ACKIE 保留	IN-NACKIE 保留	ENDIE 保留	

位15: 8	保留，始终读为0
位7	<b>OUT-STALLIE:</b> OUT包应答STALL中断使能位 (OUT-STALL interrupt enable)
位6	<b>OUT-ACKIE:</b> OUT包应答ACK中断使能位 (OUT-ACK interrupt enable)
位5	<b>OUT-NACKIE:</b> OUT包应答NACK中断使能位 (OUT-NACK interrupt enable)
位4	<b>IN-STALLIE:</b> IN包应答STALL中断使能位 (IN-STALL interrupt enable)
位3	<b>IN-ACKIE:</b> IN包应答ACK中断使能位 (IN-ACK interrupt enable)
位2	<b>IN-NACKIE:</b> IN包应答NACK中断使能位 (IN-NACK interrupt enable)
位1	<b>ENDIE:</b> 传输完成中断使能位 (Finished interrupt enable)
位0	保留

### 25.5.10 USB 地址寄存器(USB\_ADDR)

地址偏移: 0x60

复位值: 0x0000 0000

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

位15: 7	保留, 始终读为0
位6: 0	<b>ADDR[6: 0]: USB地址 (USB address)</b> 接收到主机发送的设置地址描述符时, 硬件自动将地址装载至该寄存器中。 当接收到总线复位时硬件自动清除该寄存器的值。

### 25.5.11 USB 端点使能寄存器(EP\_EN)

地址偏移: 0x64

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EP4E EP3E EP2E EPIE EP0E							
rw								rw rw rw rw rw rw							

位15: 7	保留, 始终读为0
位4	<b>EP4EN:</b> 使能端点4 (Enable End Point 4)
位3	<b>EP3EN:</b> 使能端点3 (Enable End Point 3)
位2	<b>EP2EN:</b> 使能端点2 (Enable End Point 2)
位1	<b>EP1EN:</b> 使能端点1 (Enable End Point 1)
位0	<b>EP0EN:</b> 使能端点0 (Enable End Point 0)

### 25.5.12 USB 数据翻转控制寄存器(TOG\_CTRL1\_4)

地址偏移: 0x78

复位值: 0x0000 0000

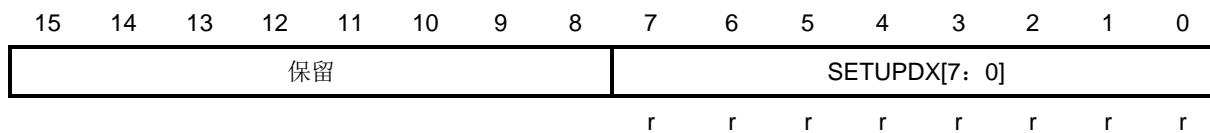
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								DTOG4EN DTOG4 DTOG3EN DTOG3 DTOG2EN DTOG2 DTOG1EN DTOG1 DTOG0 G1							
w rw w rw w rw w rw								w rw w rw w rw w rw							

位15: 8	保留, 始终读为0
位7	<b>DTOG4EN:</b> 端点4数据翻转使能位 (Set End Point 4 enable) 0: 不改变端点4的数据翻转位 1: 将位6 DTOG4设置为端点1的数据翻转位
位6	<b>DTOG4:</b> 端点4数据翻转位 (Set End Point 4 Toggle) 0: DATA0 1: DATA1
位5	<b>DTOG3EN:</b> 端点3数据翻转使能位 (Set End Point 3 enable) 0: 不改变端点3的数据翻转位 1: 将位4 DTOG3设置为端点1的数据翻转位
位4	<b>DTOG3:</b> 端点3数据翻转位 (Set End Point 3 Toggle) 0: DATA0 1: DATA1
位3	<b>DTOG2EN:</b> 端点2数据翻转使能位 (Set End Point 2 enable) 0: 不改变端点2的数据翻转位 1: 将位2 DTOG2设置为端点1的数据翻转位
位2	<b>DTOG2:</b> 端点2数据翻转位 (Set End Point 2 Toggle) 0: DATA0 1: DATA1
位1	<b>DTOG1EN:</b> 端点1数据翻转使能位 (Set End Point 1 enable) 0: 不改变端点1的数据翻转位 1: 将位0 DTOG1设置为端点1的数据翻转位
位0	<b>DTOG1:</b> 端点1数据翻转位 (Set End Point 1 Toggle) 0: DATA0 1: DATA1

### 25.5.13 USB 设置包数据寄存器(SETUPX) (0~7)

地址偏移: 0x80 到 0x9C

复位值: 0x0000 0000



位15: 8	保留, 始终读为0
位7: 0	<b>SETUPDX:</b> USB设置包数据位(x = 0,1,2,...,7) (Setup Data X) 64位SETUP数据, 由硬件根据主机发送的数据自动设置。

### 25.5.14 USB 传输包大小寄存器 (PACKET\_SIZE)

地址偏移: 0xA0

复位值: 0x40

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								SIZE0[7: 0]							
								RW	RW	RW	RW	RW	RW	RW	RW

位15: 8	保留, 始终读为0
位7: 0	<b>SIZE0:</b> USB最大传输包大小 (USB DMA Max Packet Size) 最大可设置64字节。

### 25.5.15 USB 端点 X 有效数据寄存器 (EPX\_AVAIL)

地址偏移: 0x100 到 0x110

复位值: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								EPXAVAIL7: 0]							
								R	R	R	R	R	R	R	R

位15: 8	保留, 始终读为0
位7: 0	<b>EPXAVAIL:</b> USB 端点X FIFO有效数据个数 (EPX FIFO available data number)

### 25.5.16 USB 端点 X 控制寄存器 (EPX\_CTRL)

地址偏移: 0x140 到 0x150

复位值: 0x00

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

位15: 8	保留, 始终读为0
位7	<b>TRANEN:</b> USB 端点X 传输使能位 (EPX transfer enable) 如果该位设置为1, 端点X会在IN传输的后面应答位 6: 0 中定义的数据个数, 否则端点X应答NACK。 如果端点x FIFO中没有足够的数据, 端点X同样会应答NACK。 如果端点X HALT使能位设置为1, 端点X就会自动应答STALL。 传输结束该位会自动变为0。

位6: 0	<b>TRANCOUNT:</b> 端点X传输数量 (EPX transfer counter) 端点X要传输的数据个数。数据保存在该端点的FIFO中，最大传输数量不能超过寄存器PACKAGE_SIZE定义的最大包的大小，最后一个包的传输数量可能小于最大包，甚至可以为零，意味着传输一个空包。
-------	---

### 25.5.17 USB 端点 X FIFO 寄存器(EPX\_FIFO)

地址偏移: 0x160 到 0x170

复位值: 0x00

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

位15: 8	保留，始终读为0
位7: 0	<b>EPX_FIFO :</b> 端点X FIFO数据端口 (EPX FIFO port)

### 25.5.18 USB 端点 DMA 使能寄存器(EP\_DMA)

地址偏移: 0x184

复位值: 0x00

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

位15: 2	保留，始终读为0
位1	<b>DMA2EN:</b> 端点2 DMA 使能位 (EP2 DMA enable)
位0	<b>DMA1EN:</b> 端点1 DMA 使能位 (EP1 DMA enable)

注: USB 控制器只支持端点 1 和端点 2 的 DMA 操作。

### 25.5.19 USB 端点暂停寄存器(EP\_HALT)

地址偏移: 0x188

复位值: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留								HALT4	HALT3	HALT2	HALT1	HALT0			
								rw	rw	rw	rw	rw	rw	rw	rw

位15: 5	保留, 始终读为0
位4	<b>HALT4:</b> 端点4暂停位 (EP4 halt) 当该位设为1, 设备会在IN/OUT传输后自动响应STALL。当接收到令牌包时该位会被硬件自动清零。
位3	<b>HALT3:</b> 端点3暂停位 (EP3 halt) 当该位设为1, 设备会在IN/OUT传输后自动响应STALL。当接收到令牌包时该位会被硬件自动清零。
位2	<b>HALT2:</b> 端点2暂停位 (EP2 halt) 当该位设为1, 设备会在IN/OUT传输后自动响应STALL。当接收到令牌包时该位会被硬件自动清零。
位1	<b>HALT1:</b> 端点1暂停位 (EP1 halt) 当该位设为1, 设备会在IN/OUT传输后自动响应STALL。当接收到令牌包时该位会被硬件自动清零。
位0	<b>HALT0:</b> 端点0暂停位 (EP0 halt) 当该位设为1, 设备会在IN/OUT传输后自动响应STALL。当接收到令牌包时该位会被硬件自动清零。

### 25.5.20 USB 功耗控制存器(USB\_POWER)

地址偏移: 0x1C0

复位值: 0x0000 0000

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

位15: 4	保留, 始终读为0
位3	<b>WKUP:</b> 控制器从挂起状态唤醒 (Enable controller wake up from suspend state) 1: 唤醒 0: 不唤醒
位2	保留
位1	<b>SUSP:</b> 挂起位 (suspend) 1: 正常工作模式 0: 挂起模式
位0	<b>SUSPEN:</b> 总线挂起使能位 (BUS suspend enable bit) 1: 控制器根据位1的状态直接控制USB是否挂起 0: 由控制器控制是否挂起信号

## 26. TK80

### 26.1 Features

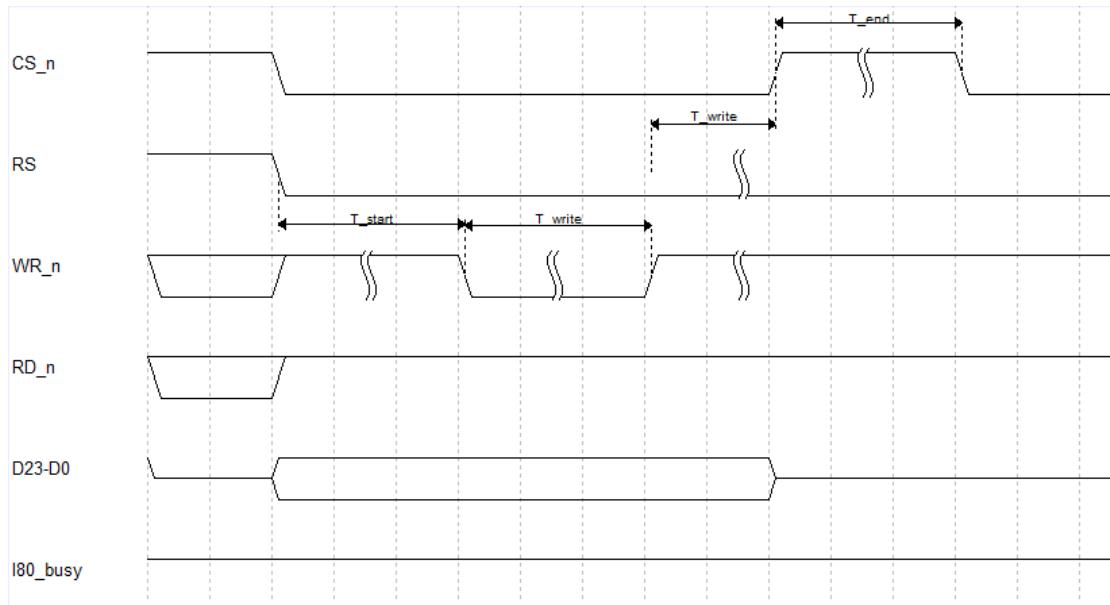
- AHB2.0 总线接口
- 支持读写命令和读写数据操作
- 读（数据或命令）支持直接读和中断/查询读两种模式
- 支持盲读
- 支持内存区域顺序读取数据操作
- 支持区域填充操作
- 支持 DMA（仅支持写）
- CS\_n 支持硬件自动生成和软件生成两种方式
- 支持读传输完成和写传输完成中断
- 双向口半双工数据传输

### 26.2 接口信号

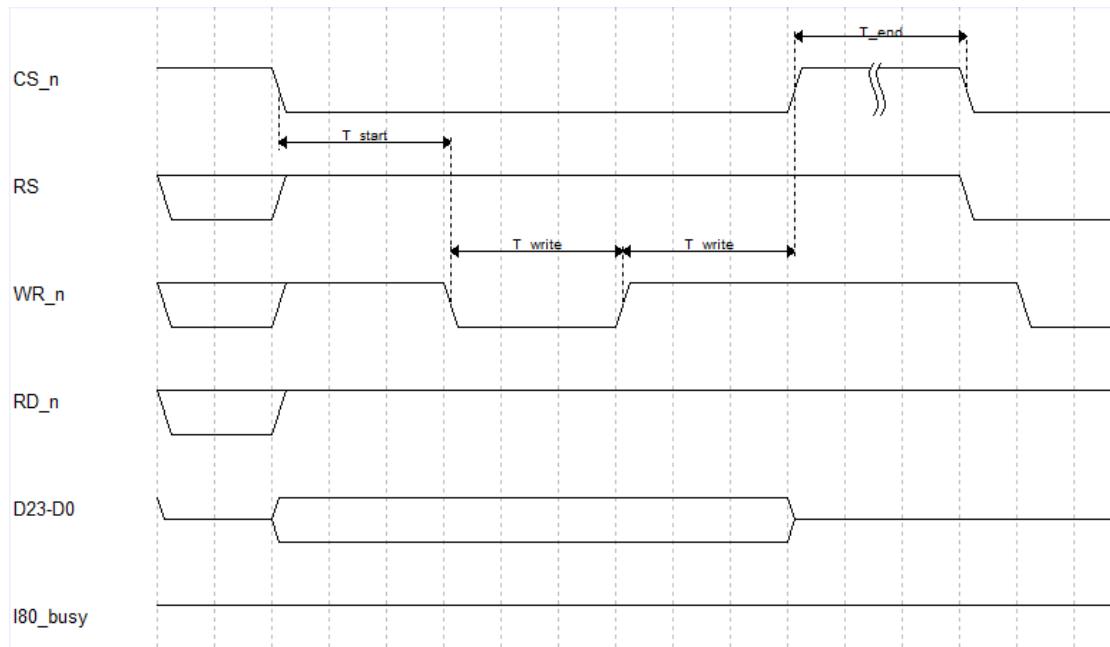
Name	IO	Function and Description
AHB总线	BUS	AHB总线，具体信号暂略
CS_n	O	<b>外设片选信号</b> 0, 表示外设被选中 1, 表示外设未被选中
RS	O	<b>命令或数据使能信号</b> 0, 表示执行的是命令操作, D23-D0上传输的是命令 1, 表示执行的是数据操作, D23-D0上传输的是数据
WR_n	O	<b>写使能</b> 用于控制向D23-D0端口上输出数据, D23-D0上的数据将在WR上升沿被锁存
RD_n	O	<b>读使能</b> 用于控制从D23-D0端口上读取数据, D23-D0上的数据将在RD上升沿之后有效
D23-D0	I/O	<b>数据输入输出信号</b> D23-D0用于向外设输出命令和数据, 或者从外设读取数据。D23为最高位数据, D0为最低位数据
l80_busy	I	外设忙信号
dma_req	O	DMA请求
dma_ack	I	DMA应答
irq	O	中断

## 26.3 接口时序

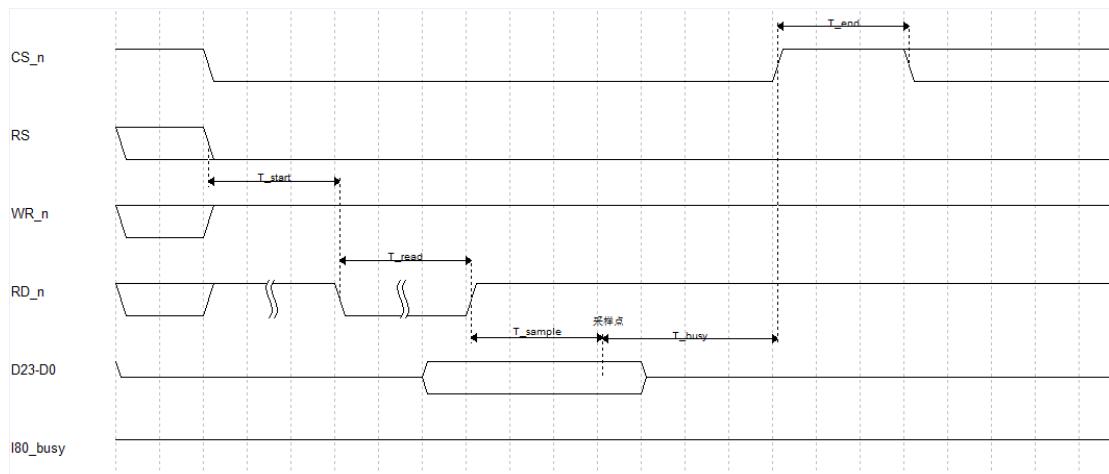
### 26.3.1 写命令



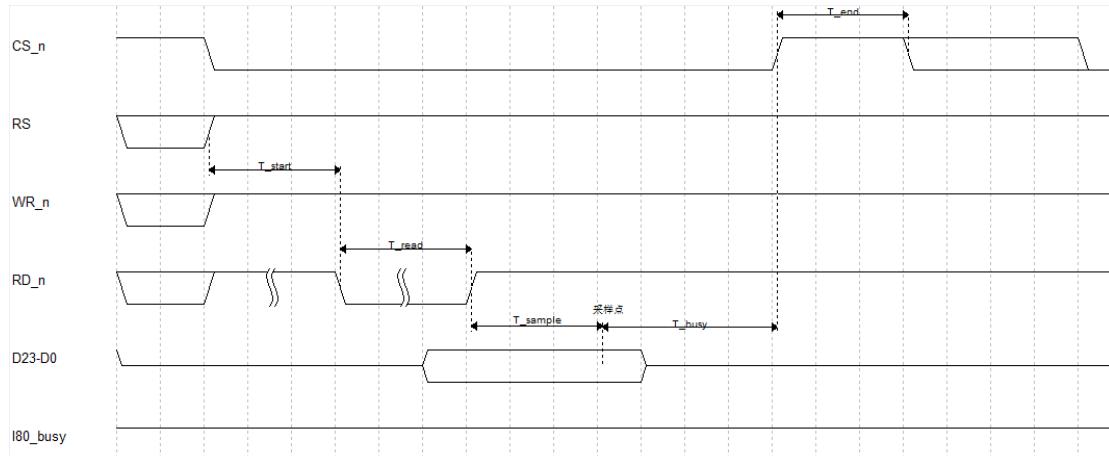
### 26.3.2 写数据



### 26.3.3 读命令

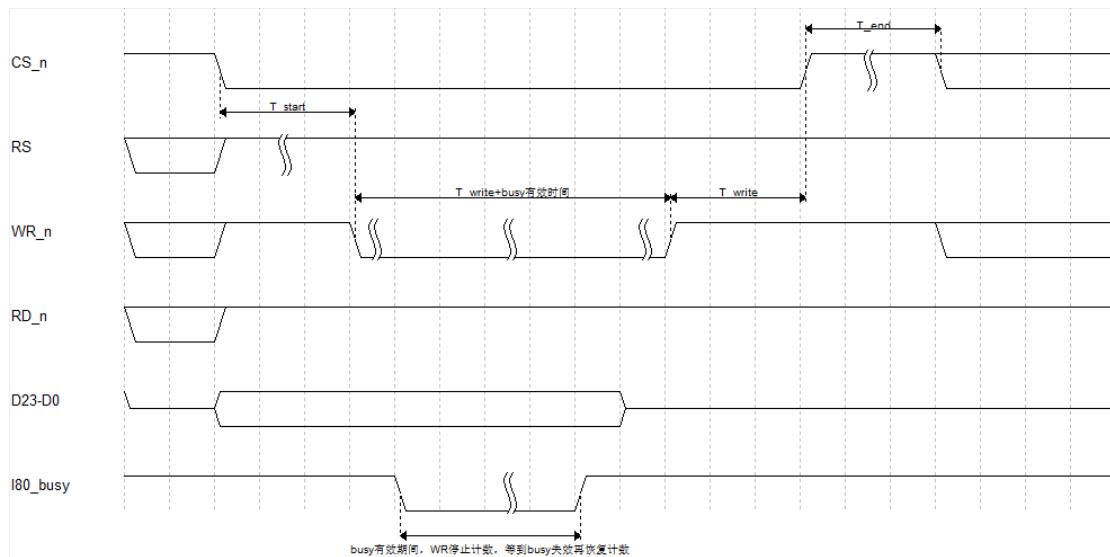


### 26.3.4 读数据



### 26.3.5 I80\_busy

任意情况下，如果碰到 I80\_busy 有效，直接停止相应操作，并等待 I80\_busy 失效，然后继续之前的操作。等待 I80\_busy 的时间不计入控制信号计数时间。以写数据举例如下。



## 26.4 时序参数

单位: clock cycles

T_start	参见配置寄存器CFGRI
T_write	参见配置寄存器CFGRI
T_read	参见配置寄存器CFGRI
T_sample	参见配置寄存器CFGRI
T_busy	参见配置寄存器CFGRII
T_end	参见配置寄存器CFGRII

## 26.5 寄存器描述

### 26.5.1 控制寄存器 CR

offset: 0x00

default: 0x000c0b02

位31:22	res
位21:18	narrow_num: 输出数据有效valid时间缩短cycle 实际cycle为narrow_num+1
位17	narrow_valid: 输出数据有效valid时间缩短 0: 不缩短 1: 缩短
位16	burst: 填充模式 1: 启动填充模式 0: 非填充模式
位15:12	res
位11	busy_val 0: 外部busy信号无效 1: 外部busy信号有效

位10	busy_resp_level 0: I80_busy为高表示device忙 1: I80_busy为低表示device忙
位9:8	read_mode: 定义两种读数据模式 00: 读命令模式2 01: 读数据模式2 10: 读命令模式1 11: 读数据模式1
位7:3	res
位2	CS_sel: 软件CS选择信号 0: 硬件产生CS 1: 软件产生CS
位1	CS_soft_n: 软件CS, CS_soft为1时有效, CS_soft为0时无效 0: 软件拉低CS_n信号 1: 软件拉高CS_n信号
位0	dma_en: DMA使能 1: DMA使能 0: DMA禁能

### 26.5.2 配置寄存器 CFGR1

offset: 0x04

default: 0x01cf01

位31:24	T_sample, 实际值为T_sample +1, T_sample>1
位23:16	T_read, 实际值为T_read +1, T_read>1
位15:8	T_write, 实际值为T_write +1, T_write>1
位7:0	T_start, 实际值为T_start +1, T_start>1

### 26.5.3 配置寄存器 CFGR2

offset: 0x08

default: 0x00010101

位31:16	res
位15:8	T-Busy
位7:0	T-end, 实际值为T_end+1, T_end>1

#### 26.5.4 状态寄存器 SR

offset: 0x0C

default: 0x00000000

位31:17	res
位16	busy: 判断TK80是否正在对device操作 1: 表示TK80接口正在对device进行操作 0: 未操作
位15:12	res
位11	write_single_ie 单次写数据完成中断使能
位10	read_single_ie 单次读取数据完成中断使能
位9	write_ie 写数据完成中断使能
位8	read_ie 读取数据完成中断使能
位7: 4	res
位3	write_end_single: 单次写数据完成, 写1清零
位2	read_end_single 单次读数据完成, 写1清零
位1	write_end 写数据完成, 写1清零
位0	read_end 读取数据完成, 写1清零

#### 26.5.5 命令输入寄存器 CMDIR

offset: 0x10

default: 0x00000000

31:24	res
23:0	cmdi:

#### 26.5.6 数据输入寄存器 DINR

offset: 0x14

default: 0x00000000

31:24	res
23:0	din:

CPU 通过写该寄存器向外设传递数据, 该寄存器的值通过 D23-D0 传递到外设

### 26.5.7 命令输出寄存器 CMDOR

offset: 0x18

default: 0x00000000

31:24	res
23:0	cmdo:

### 26.5.8 数据输出寄存器 DOUTR

offset: 0x20

default: 0x00000000

31:24	res
23:0	dout:

### 26.5.9 盲读数据输出寄存器 BRDR

offset: 0x24

default: 0x00000000

31:24	res
23:0	brdr:

### 26.5.10 配置寄存器 CFGR3

offset: 0x30

default: 0x00000001

位31:0	circle_num: 填充点数 0: 单点填充 N: 自动进行N点填充
-------	--