



导言

本参考手册为应用开发人员提供了关于如何使用STM8S微控制器的存储器和外设的完整信息。STM8S是一个拥有不同存储器大小，封装和外设的微控制器家族。

- STM8S针对通用应用而设计，关于订货信息，引脚描述，器件的机械及电气参数，请参考STM8S增强型及基本型数据手册。
- 关于内部FLASH存储器的编程，擦除和保护，请参考STM8S Flash编程手册([PM0051](#)) 和STM8 SWIM 通讯协议及调试模块用户手册([UM0470](#))
- 关于STM8内核，请参考STM8 CPU 编程手册([PM0044](#))。

本译文的英文原文下载地址为：<http://www.st.com/stonline/products/literature/rm/14587.pdf>

目录

1 中央处理器(CPU)	11
1.1 简介	11
1.2 CPU寄存器	11
1.2.1 CPU寄存器描述	11
1.2.2 STM8 CPU寄存器映射	13
1.3 全局配置寄存器(CFG_GCR)	14
1.3.1 激活级别	14
1.3.2 禁用SWIM	14
1.3.3 全局配置寄存器(CFG_GCR)描述	14
1.3.4 全局配置寄存器映射及复位值	14
2 用于系统启动的只读存储器	15
3 存储器和寄存器映射	16
3.1 寄存器描述缩写	16
4 FLASH程序存储器和数据EEPROM	17
4.1 介绍	17
4.2 词汇表	17
4.3 FLASH主要特性	17
4.4 存储器组织结构	17
4.4.1 用户启动区域(UBC)	20
4.4.2 数据EEPROM(DATA)	23
4.4.3 主程序区	23
4.4.4 选项字节(Option byte)	23
4.5 存储器保护	24
4.5.1 读保护	24
4.5.2 存储器存取安全系统(MASS)	24
4.5.3 对选项字节的写操作	25
4.6 存储器编程	25
4.7 读同时写(RWW)	25
4.7.1 字节编程	25
4.7.2 字编程	26
4.7.3 块编程	26
4.7.4 选项字节(Option byte)编程	27
4.8 ICP和IAP	27
4.9 FLASH寄存器	29
4.9.1 FLASH控制寄存器1(FLASH_CR1)	29
4.9.2 FLASH控制寄存器2(FLASH_CR2)	30
4.9.3 FLASH互补控制寄存器2(FLASH_NCR2)	31
4.9.4 FLASH保护寄存器(FLASH_FPR)	32
4.9.5 FLASH保护寄存器(FLASH_NFPR)	33
4.9.6 FLASH程序存储器解保护寄存器(FLASH_PUKR)	34
4.9.7 DATA EEPROM解保护寄存器(FLASH_DUKR)	35
4.9.8 FLASH状态寄存器(FLASH_IAPSR)	36
4.9.9 FLASH寄存器映射和复位值	37
5 单线接口模块(SWIM)和调试模块(DM)	38
5.1 介绍	38
5.2 主要特性	38
5.3 SWIM模式	38
6 供电电源	39
7 复位(RST)	40

7.1	复位电路.....	40
7.2	内部复位源.....	40
7.2.1	上电复位(POR)和掉电复位(BOR).....	40
7.2.2	看门狗复位.....	41
7.2.3	软件复位.....	41
7.2.4	SWIM复位.....	41
7.2.5	非法操作码复位.....	41
7.2.6	EMS复位.....	41
7.3	复位(RST)寄存器.....	42
7.3.1	复位状态寄存器(RST_SR).....	42
7.4	复位寄存器地址映射.....	42
8	时钟控制.....	43
8.1	主时钟源.....	44
8.1.1	HSE.....	45
8.1.2	HSI.....	46
8.1.3	LSI.....	46
8.2	主时钟切换.....	46
8.2.1	系统启动.....	46
8.2.2	主时钟切换的过程.....	46
8.3	低速时钟源的选择.....	49
8.4	CPU时钟分频器.....	49
8.5	外设时钟门控.....	49
8.6	时钟安全系统(CSS).....	50
8.7	时钟输出功能(CCO).....	50
8.8	时钟中断.....	51
8.9	时钟寄存器.....	52
8.9.1	内部时钟寄存器(CLK_ICKR).....	52
8.9.2	外部时钟寄存器(CLK_ECKR).....	53
8.9.3	主时钟状态寄存器(CLK_CMSR).....	54
8.9.4	主时钟切换寄存器(CLK_SWR).....	55
8.9.5	切换控制寄存器(CLK_SWCR).....	56
8.9.6	时钟分频寄存器(CLK_CKDIVR).....	57
8.9.7	外设时钟门控寄存器(CLK_PCKENR1).....	58
8.9.8	外设时钟门控寄存器2(CLK_PCKENR2).....	59
8.9.9	时钟安全系统寄存器(CLK_CSSR).....	60
8.9.10	可配置时钟输出寄存器.....	61
8.9.11	CAN外部时钟控制寄存器(CLK_CANCCR).....	62
8.9.12	HSI时钟修正寄存器(CLK_HSITRIMR).....	63
8.9.13	SWIM时钟控制寄存器(CLK_SWIMCCR).....	64
8.10	时钟寄存器地址映射.....	65
9	电源管理.....	66
9.1	常规考虑.....	66
9.2	低功耗的时钟管理.....	66
9.2.1	降低系统时钟.....	66
9.2.2	外设时钟门控.....	66
9.3	低功耗模式.....	67
9.3.1	等待(Wait)模式.....	67
9.3.2	停机(Halt)模式.....	67
9.3.3	活跃停机(Active Halt)模式.....	68
9.4	附加的模拟功耗控制.....	68
9.4.1	停机模式下的快速内存唤醒.....	68
9.4.2	活跃停机模式下的超低内存功耗.....	68
10	中断控制器(ITC).....	69

10.1	简介	69
10.2	中断屏蔽和处理流程	69
10.2.1	处理等待(排队)的中断	70
10.2.2	中断源	71
10.3	中断和低功耗模式	72
10.4	活动等级/低功耗模式的控制	72
10.5	同时的和嵌套的中断管理	72
10.5.1	同时发生中断管理模式	72
10.5.2	嵌套中断管理模式	73
10.6	外部中断	74
10.7	中断指令	74
10.8	中断映射	75
10.9	ITC寄存器	76
10.9.1	CPU CC寄存器中断位	76
10.9.2	软件优先级寄存器x (ITC_SPRx)	77
10.9.3	外部中断控制寄存器1 (EXTI_CR1)	78
10.9.4	外部中断控制寄存器2 (EXTI_CR2)	79
10.9.5	寄存器表和复位值	80
11	通用输入输出口(GPIO)	81
11.1	简介	81
11.2	GPIO主要功能	81
11.3	I/O的配置和使用	82
11.3.1	输入模式	83
11.3.2	输出模式	83
11.4	复位后的默认配置	83
11.5	没有使用的引脚	83
11.6	低功耗模式	83
11.7	输入模式的详述	83
11.7.1	复用功能输入	83
11.7.2	中断功能	84
11.7.3	模拟通道	84
11.7.4	施密特触发器	84
11.8	输出模式详述	84
11.8.1	复用功能的输出	84
11.8.2	摆率控制	84
11.9	GPIO寄存器	84
11.9.1	端口x输出数据寄存器 (Px_ODR)	85
11.9.2	端口x输入寄存器 (Px_IDR)	86
11.9.3	端口x数据方向 (Px_DDR)	87
11.9.4	端口x控制寄存器1 (Px_CR1)	88
11.9.5	端口x控制寄存器2 (Px_CR2)	89
11.9.6	GPIO寄存器表和复位值	89
12	自动唤醒(AWU)	90
12.1	简介	90
12.2	AWU功能描述	90
12.2.1	AWU操作	90
12.2.2	时基选择	91
12.2.3	LSI低速内部时钟频率检测	91
12.3	AWU寄存器	92
12.3.1	控制/状态寄存器 (AWU_CSR)	92
12.3.2	异步预分频寄存器 (AWU_APR)	93
12.3.3	时基选择寄存器 (AWU_TBR)	94
12.3.4	AWU寄存器表和复位值	95

13 蜂鸣器(BEEP)	96
13.1 简介	96
13.2 功能描述	96
13.2.1 蜂鸣器操作	96
13.2.2 蜂鸣器校准	96
13.3 蜂鸣器 寄存器	97
13.3.1 蜂鸣器 控制/状态 寄存器 (BEEP_CSR)	97
13.3.2 BEEP寄存器表和复位值	97
14 独立看门狗(IWDG)	98
14.1 介绍	98
14.2 独立看门狗功能说明	98
14.3 IWDG寄存器	99
14.3.1 键寄存器(IWDG_KR)	99
14.3.2 预分频寄存器(IWDG_PR)	100
14.3.3 重装载寄存器(IWDG_RLR)	101
14.3.4 IWDG寄存器映像和复位数值	101
15 窗口看门狗(WWDG)	102
15.1 介绍	102
15.2 WWDG主要功能	102
15.3 WWDG功能说明	102
15.4 在停止模式下使用WWDG	103
15.5 如何设置看门狗的超时	103
15.6 WWDG低功耗模式	104
15.7 硬件看门狗选项	104
15.8 在停止模式下使用WWDG	104
15.9 WWDG中断	105
15.10 WWDG寄存器	105
15.10.1 控制寄存器(WWDG_CR)	105
15.10.2 窗口寄存器(WWDG_WR)	106
15.11 窗口看门狗寄存器映像和复位数值	106
16 定时器概述	107
16.1 定时器功能比较	108
16.2 定时器信号术语表	108
17 16位高级控制定时器(TIM1)	110
17.1 简介	110
17.2 主要特性	110
17.3 时基单元	112
17.3.1 读写16位计数器	112
17.3.2 16位TIM1_ARR寄存器的写操作	113
17.3.3 预分频器	113
17.3.4 向上计数模式	113
17.3.5 向下计数模式	115
17.3.6 中央对齐模式(向上/向下计数)	116
17.3.7 重复计数器	117
17.4 时钟/触发控制器	118
17.4.1 预分频时钟(CK_PSC)	119
17.4.2 内部时钟源(f_{MASTER})	119
17.4.3 外部时钟源模式1	119
17.4.4 外部时钟源模式2	120
17.4.5 触发同步	121
17.4.6 与TIM5/TIM6定时器的同步	124
17.5 捕获/比较通道	129

17.5.1	16位TIM1_CCR <i>i</i> 寄存器的写流程	130
17.5.2	输入模块	130
17.5.3	输入捕获模式	131
17.5.4	输出模块	132
17.5.5	强制输出模式	133
17.5.6	输出比较模式	133
17.5.7	PWM模式	134
17.5.8	使用刹车功能	139
17.5.9	在外部事件发生时清除OCREF信号	141
17.5.10	编码器接口模式	142
17.6	中断	143
17.7	TIM1寄存器描述	145
17.7.1	控制寄存器1(TIM1_CR1)	145
17.7.2	控制寄存器2(TIM1_CR2)	146
17.7.3	从模式控制寄存器(TIM1_SMCR)	147
17.7.4	外部触发寄存器(TIM1_ETR)	148
17.7.5	中断使能寄存器(TIM1_IER)	149
17.7.6	状态寄存器1 (TIM1_SR1)	150
17.7.7	状态寄存器2(TIM1_SR2)	151
17.7.8	事件产生寄存器(TIM1_EGR)	152
17.7.9	捕获/比较模式寄存器1(TIM1_CCMR1)	153
17.7.10	捕获/比较模式寄存器2(TIM1_CCMR2)	156
17.7.11	捕获/比较模式寄存器3(TIM1_CCMR3)	157
17.7.12	捕获/比较模式寄存器4(TIM1_CCMR4)	158
17.7.13	捕获/比较使能寄存器1(TIM1_CCER1)	159
17.7.14	捕获/比较使能寄存器2(TIM1_CCER2)	161
17.7.15	计数器高8位(TIM1_CNTRH)	162
17.7.16	计数器低8位(TIM1_CNTRL)	163
17.7.17	预分频器高8位(TIM1_PSCRH)	164
17.7.18	预分频器低8位(TIM1_PSCRL)	165
17.7.19	自动重装载寄存器高8位(TIM1_ARRH)	166
17.7.20	自动重装载寄存器低8位(TIM1_ARRL)	167
17.7.21	重复计数寄存器(TIM1_RCR)	168
17.7.22	捕获/比较寄存器1高8位(TIM1_CCR1H)	169
17.7.23	捕获/比较寄存器1低8位(TIM1_CCR1L)	170
17.7.24	捕获/比较寄存器2高8位(TIM1_CCR2H)	171
17.7.25	捕获/比较寄存器2低8位(TIM1_CCR2L)	172
17.7.26	捕获/比较寄存器3高8位(TIM1_CCR3H)	173
17.7.27	捕获/比较寄存器3低8位(TIM1_CCR3L)	174
17.7.28	捕获/比较寄存器4高8位(TIM1_CCR4H)	175
17.7.29	捕获/比较寄存器4低8位(TIM1_CCR4L)	176
17.7.30	刹车寄存器(TIM1_BKR)	177
17.7.31	死区寄存器(TIM1_DTR)	178
17.7.32	输出空闲状态寄存器(TIM1_OISR)	179
17.7.33	TIM1寄存器图	180
18	16位通用定时器(TIM2, TIM3, TIM5)	182
18.1	介绍	182
18.2	TIM2/TIM3的主要功能	182
18.3	TIM5主要功能	182
18.4	TIM2/TIM3/TIM5功能概述	183
18.4.1	时基单元	183
18.4.2	时钟/触发控制器	184
18.4.3	捕获/比较通道	184
18.5	中断	187
18.6	TIM2/TIM3/TIM5寄存器	188

18.6.1	控制寄存器 1 (TIMx_CR1)	188
18.6.2	控制寄存器 2 (TIMx_CR2)	189
18.6.3	触发从模式控制寄存器 (TIMx_SMCR)	190
18.6.4	中断使能寄存器 (TIMx_IER)	191
18.6.5	状态寄存器 1 (TIMx_SR1)	192
18.6.6	状态寄存器 2 (TIMx_SR2)	193
18.6.7	事件产生寄存器 (TIMx_EGR)	194
18.6.8	捕获/比较模式寄存器 1 (TIMx_CCMR1)	195
18.6.9	捕获/比较模式寄存器 2 (TIMx_CCMR2)	197
18.6.10	捕获/比较模式寄存器 3 (TIMx_CCMR3)	198
18.6.11	捕获/比较使能寄存器 1 (TIMx_CCER1)	199
18.6.12	捕获/比较使能寄存器 2 (TIMx_CCER2)	200
18.6.13	计数器高位 (TIMx_CNTRH)	201
18.6.14	计数器低位 (TIMx_CNTRL)	202
18.6.15	预分频器 (TIMx_PSCR)	203
18.6.16	自动装载寄存器高位 (TIMx_ARRH)	204
18.6.17	自动装载寄存器低位 (TIMx_ARRL)	205
18.6.18	捕获/比较寄存器 1 高位 (TIMx_CCR1H)	206
18.6.19	捕获/比较寄存器 1 低位 (TIMx_CCR1L)	207
18.6.20	捕获/比较寄存器 2 高位 (TIMx_CCR2H)	208
18.6.21	捕获/比较寄存器 2 低位 (TIMx_CCR2L)	209
18.6.22	捕获/比较寄存器 3 高位 (TIMx_CCR3H)	210
18.6.23	捕获/比较寄存器 3 低位 (TIMx_CCR3L)	211
18.6.24	TIM2/TIM3/TIM5 寄存器图和复位值	212
19	8 位基本定时器 (TIM4, TIM6)	218
19.1	简介	218
19.2	TIMER4 的主要功能	218
19.3	TIMER6 的主要功能	218
19.4	TIM4/TIM6 中断	219
19.5	TIM4/TIM6 时钟选择	219
19.6	TIM4/TIM6 寄存器	220
19.6.1	控制寄存器 1 (TIMx_CR1)	220
19.6.2	控制寄存器 2 (TIMx_CR2)	221
19.6.3	从模式控制寄存器 (TIMx_SMCR)	222
19.6.4	中断使能寄存器 (TIMx_IER)	223
19.6.5	状态寄存器 1 (TIMx_SR1)	224
19.6.6	事件产生寄存器 (TIMx_EGR)	225
19.6.7	计数器 (TIMx_CNTR)	226
19.6.8	预分频寄存器 (TIMx_PSCR)	227
19.6.9	自动重装载寄存器 (TIMx_ARR)	228
19.6.10	TIM4/TIM6 寄存器表和复位值	229
20	串行外设接口 (SPI)	230
20.1	SPI简介	230
20.2	SPI主要特征	230
20.3	SPI功能描述	230
20.3.1	概述	230
20.3.2	SPI从模式	233
20.3.3	SPI主模式	234
20.3.4	单工通信	234
20.3.5	状态标志	235
20.3.6	CRC计算	235
20.3.7	错误标志	236
20.3.8	关闭SPI	236
20.3.9	低功耗	237

20.3.10 SPI中断	238
20.4 SPI寄存器描述	238
20.4.1 SPI控制寄存器1(SPI_CR1)	238
20.4.2 SPI控制寄存器2(SPI_CR2)	239
20.4.3 SPI中断控制寄存器(SPI_ICR)	240
20.4.4 SPI状态寄存器(SPI_SR)	241
20.4.5 SPI数据寄存器(SPI_DR)	242
20.4.6 SPI CRC多项式寄存器(SPI_CRCPR)	243
20.4.7 SPI Rx CRC寄存器(SPI_RXCRCR)	244
20.4.8 SPI Tx CRC寄存器(SPI_TXCRCR)	245
20.5 SPI寄存器地址映象以及复位值	245
21 I ² C接口	246
21.1 I ² C简介	246
21.2 I ² C主要特点	246
21.3 I ² C简介	246
21.4 I ² C功能描述	248
21.4.1 I ² C从模式	248
21.4.2 I ² C主模式	250
21.4.3 出错状态	252
21.4.4 SDA/SCL线控制	253
21.5 低功耗模式	253
21.6 I ² C中断请求	254
21.7 I ² C寄存器描述	255
21.7.1 控制寄存器1(I2C_CR1)	255
21.7.2 控制寄存器2(I2C_CR2)	256
21.7.3 频率寄存器(I2C_FREQR)	257
21.7.4 自身地址寄存器LSB(I2C_OARL)	258
21.7.5 自身地址寄存器MSB(I2C_OARH)	259
21.7.6 数据寄存器(I2C_DR)	260
21.7.7 状态寄存器1(I2C_SR1)	261
21.7.8 状态寄存器2(I2C_SR2)	263
21.7.9 状态寄存器3(I2C_SR3)	264
21.7.10 中断寄存器(I2C_ITR)	265
21.7.11 时钟控制寄存器低位部分(I2C_CCRL)	266
21.7.12 时钟控制寄存器高位部分(I2C_CCRH)	267
21.7.13 TRISE寄存器(I2C_TRISE)	268
21.7.14 I ² C寄存器地址映射和复位值	269
22 通用异步收发器(UART)	270
22.1 UART介绍	270
22.2 UART主要特性	270
22.3 UART功能概述	271
22.3.1 UART特性描述	275
22.3.2 发送器	275
22.3.3 接收器	277
22.3.4 高精度波特率发生器	279
22.3.5 奇偶校验控制	280
22.3.6 多处理器通信	280
22.3.7 LIN(局域互联网)模式	282
22.3.8 UART同步模式	282
22.3.9 单线半双工通信	284
22.3.10 智能卡	284
22.3.11 IrDA SIR ENDEC功能块	285
22.4 LIN模式功能描述	287
22.4.1 主模式	287

22.4.2	自动重同步功能禁用的从模式	290
22.4.3	自动重同步使能的从模式	293
22.4.4	LIN模式选择	296
22.5	低功耗模式	297
22.6	中断	297
22.7	UART寄存器描述	299
22.7.1	状态寄存器(UART_SR)	299
22.7.2	数据寄存器(UART_DR)	301
22.7.3	波特比率寄存器1(UART_BRR1)	302
22.7.4	波特比率寄存器2 (UART_BRR2)	303
22.7.5	控制寄存器1(UART_CR1)	304
22.7.6	控制寄存器2(UART_CR2)	305
22.7.7	控制寄存器3(UART_CR3)	307
22.7.8	控制寄存器4(UART_CR4)	308
22.7.9	控制寄存器5(UART_CR5)	309
22.7.10	控制寄存器6(UART_CR6)	310
22.7.11	保护时间寄存器(UART_GTR)	311
22.7.12	分频寄存器(UART_PSCR)	312
22.7.13	UART寄存器地址映射	313
23	控制器局域网(BECAN)	316
23.1	简介	316
23.2	主要特点	316
23.3	总体描述	316
23.3.1	CAN 2.0B (active)内核	317
23.3.2	控制、状态和配置寄存器	317
23.3.3	发送邮箱	317
23.3.4	接收过滤器	317
23.4	工作模式	318
23.4.1	初始化模式	319
23.4.2	正常模式	319
23.4.3	睡眠模式(低功耗)	319
23.4.4	时间触发通讯模式	319
23.5	测试模式	320
23.5.1	静默模式	320
23.5.2	环回模式	320
23.5.3	环回静默模式	320
23.6	功能描述	321
23.6.1	发送处理	321
23.6.2	接收处理	322
23.6.3	标识符过滤	324
23.6.4	报文存储	328
23.6.5	出错管理	329
23.6.6	位时序	330
23.7	中断	332
23.8	寄存器访问保护	333
23.9	时钟系统	333
23.10	BECAN低功耗模式	334
23.11	CAN寄存器描述	335
23.11.1	CAN主控制寄存器 (CAN_MCR)	335
23.11.2	CAN主状态寄存器 (CAN_MSR)	336
23.11.3	CAN发送状态寄存器 (CAN_TSR)	337
23.11.4	CAN发送优先级寄存器 (CAN_TPR)	338
23.11.5	CAN接收FIFO 1 寄存器(CAN_RFR)	339
23.11.6	CAN中断允许寄存器 (CAN_IER)	340
23.11.7	CAN诊断寄存器 (CAN_DGR)	341

23.11.8	CAN页面选择寄存器 (CAN_PSR)	342
23.11.9	CAN错误状态寄存器 (CAN_ESR)	343
23.11.10	CAN出错中断使能寄存器 (CAN_EIER)	344
23.11.11	CAN发送出错计数器寄存器 (CAN_TECR)	345
23.11.12	CAN接收出错计数器寄存器 (CAN_RECRR)	346
23.11.13	CAN位时间特性寄存器 (CAN_BTR1)	347
23.11.14	CAN位时间特性寄存器 (CAN_BTR2)	348
23.11.15	邮箱寄存器	349
23.11.16	CAN过滤器寄存器	352
23.12	BECAN寄存器列表	356
23.12.1	CAN的页映射	357
24	模拟 / 数字转换器 (ADC)	361
24.1	简介	361
24.2	主要功能	361
24.3	扩展 (增强) 功能	361
24.4	引脚描述	363
24.5	功能描述	363
24.5.1	ADC 开-关控制	363
24.5.2	ADC 时钟	364
24.5.3	通道选择	364
24.5.4	转换模式	364
24.5.5	溢出标志位	365
24.5.6	模拟看门狗	365
24.5.7	基于外部触发信号的转换	366
24.5.8	模拟放大	366
24.5.9	时序图	366
24.6	低功耗模式	367
24.7	中断	367
24.8	数据对齐	370
24.9	读取转换结果	371
24.10	施密特触发器禁止寄存器	371
24.11	寄存器描述	372
24.11.1	ADC高位数据缓存寄存器(ADC_DBxRH) ($x=0..7$ or $0..9$)	372
24.11.2	ADC低位数据缓存寄存器(ADC_DBxRL) ($x=0..7$ or $0..9$)	373
24.11.3	ADC控制/状态寄存器(ADC_CSR)	374
24.11.4	ADC 配置寄存器 1 (ADC_CR1)	375
24.11.5	ADC 配置寄存器 2 (ADC_CR2)	376
24.11.6	ADC配置寄存器 3 (ADC_CR3)	377
24.11.7	ADC 数据高位寄存器(ADC_DRH)	378
24.11.8	ADC 数据低位寄存器(ADC_DRL)	379
24.11.9	ADC 施密特触发器禁止寄存器高位 (ADC_TDRH)	380
24.11.10	ADC 施密特触发器禁止寄存器低位 (ADC_TDRL)	381
24.11.11	ADC 上限门槛值高位寄存器(ADC_HTRH)	382
24.11.12	ADC 上限门槛值低位寄存器(ADC_HTRL)	383
24.11.13	ADC 下限门槛值高位寄存器(ADC_LTRH)	384
24.11.14	ADC 下限门槛值低位寄存器(ADC_LTRL)	385
24.11.15	ADC看门狗状态高位寄存器(ADC_AWSRH)	386
24.11.16	ADC看门狗状态低位寄存器(ADC_AWSRL)	387
24.11.17	ADC看门狗控制高位寄存器(ADC_AWCRH)	388
24.11.18	ADC看门狗控制低位寄存器(ADC_AWCRL)	389
24.12	ADC寄存器映像表和复位值	390

1 中央处理器(CPU)

1.1 简介

STM8S是基于8位框架结构的微控制器，其CPU内核有6个内部寄存器，通过这些寄存器可高效地进行数据处理。STM8S的指令集支持80条基本语句及20种寻址模式，而且CPU的6个内部寄存器都拥有可寻址的地址。如果想了解全部STM8S指令集，请参考STM8 微控制器家族编程手册(PM0044)。

1.2 CPU寄存器

在图1所示的编程模型中可以看到6个CPU寄存器。在一个中断发生后，寄存器以图2所示顺序入栈，它们以相反的顺序出栈。如果需要的话，中断服务程序可使用POP和PUSH指令来对之进行操作。

1.2.1 CPU寄存器描述

累加器(A)

累加器是一个8位通用目的寄存器，用于保存算术运算、逻辑运算以及数据操作的操作数及结果。

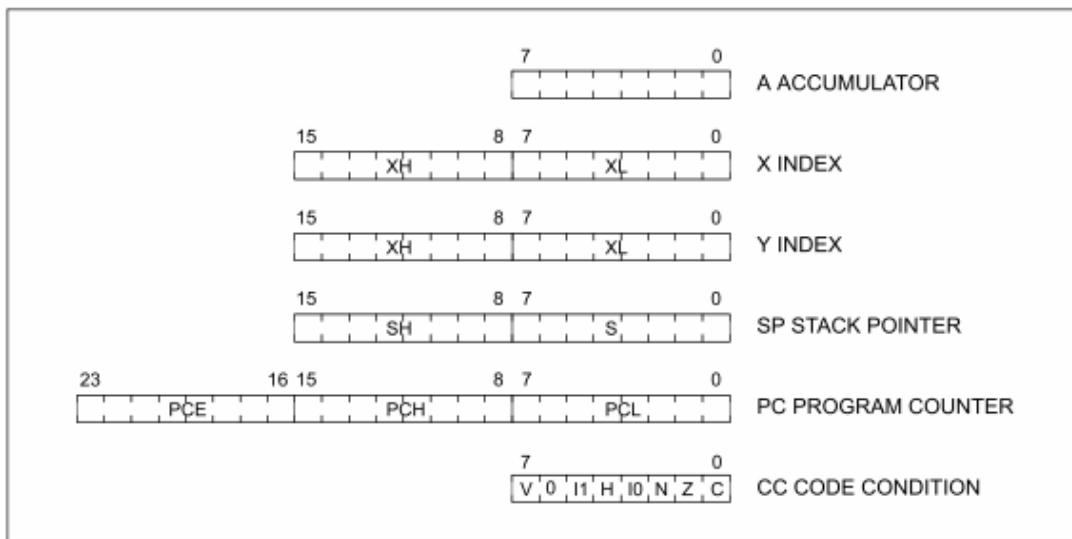
索引寄存器(X和Y)

X和Y都是16位的寄存器，可实现高效率的寻址模式。它们也可用作数据操作的暂存器以及用于像乘除法这样的操作。在大多数情况下，交叉汇编器会在使用了Y寄存器的指令代码中生成PRECODE指令，用以和使用了X寄存器的指令相区别。

程序计数器(PC)

程序计数器是一个24位的寄存器，用于存储CPU下一条要执行指令的地址。其内容在每一次指令操作后被自动刷新。由于程序指针有24位，因此STM8的最大寻址范围可达16M字节。

图1 编程模型



堆栈指针(SP)

堆栈指针是一个16位的寄存器，其内容为堆栈中下一个可自由分配的单元地址。根据不同的型号，堆栈指针的高位会有一个指定的预设值。

堆栈一般用于在中断调用或子程序调用时存储CPU的上下文(程序计数器，关键寄存器，相关函数的参数及局部变量，等等)。用户也可以通过POP和PUSH指令直接对堆栈操作。

SP可以被C编译器的启动代码初始化，C语言应用程序会根据用户所使用的包含绝对地址信息的链接文件来进行初始化。如果用户使用了自己编写的链接文件和启动代码，请确认SP被恰当地

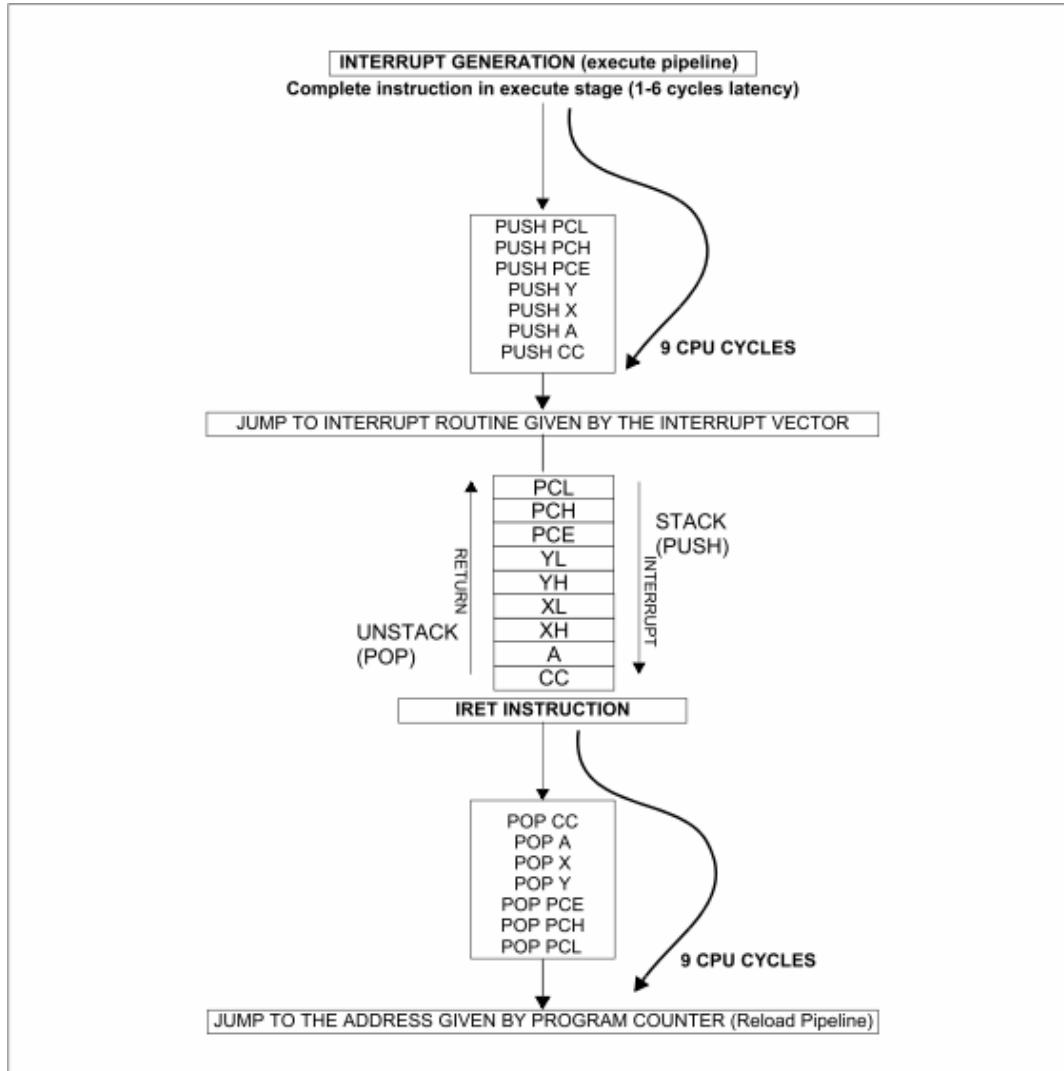
初始化(具体地址信息请参考相应的数据手册)。在MCU复位后, 或在执行了堆栈复位指令后(RSP), 堆栈指针被设为其被允许的最大值。对于使用了汇编语言的应用程序, 用户可使用ST提供的启动代码或编写自己的启动代码来对SP进行正确的初始化。

入栈操作使堆栈指针值减小, 出栈操作使堆栈指针值增加。当堆栈指针值为其被允许的最小值时, 继续入栈会使堆栈指针值回卷至其最大值, 从而会导致先前存储的数据被覆盖, 但此时没有中断或硬件标志位来指示该事件发生。应用程序需确保堆栈被正确的操作, 没有溢出。

子程序调用会占用2或3字节空间。中断调用会占用9字节空间来存储内部寄存器(除SP之外)。请参考图2。

注: WFI/HALT指令会预先保存CPU上下文。如果CPU处于WFI或HALT状态下有中断发生, 则进入中断所需的延时会相应减少。

图2 入栈出栈顺序



条件代码寄存器(CC)

条件代码寄存器是一个8位寄存器, 用于指示刚刚被执行的指令结果及处理器的状态。寄存器的第7位(MSB)是保留位, 这些位可以被用户的程序或代码单独地测试, 测试的结果可用于指示程序或代码执行后的状态。下面的段落将描述每一位的含义。

● V:溢出

在上一次有符号数的算术操作中, 如果结果的最高位有溢出发生, 则当该位被置1。请参考 INC, INCW, DEC, DECW, NEG, NEGW, ADD, ADDW, ADC, SUB, SUBW, SBC, CP, CPW等指令。

表1 中断级别

可中断性	优先级	I1	I0
可中断 主程序	最低	1	0
可中断 级别1		0	1
可中断 级别2		0	0
不可中断	最高	1	1

● I1:中断屏蔽级别1

I1和I0共同用于指示当前状态下CPU的可中断性, 请参考表1。通过执行RIM, SIM, HALT, WFI, IRET, TRAP和POP指令可对I1和I0置位或清零。I1和I0也会在CPU进入中断服务程序时被硬件自动设置为该中断对应的中断级别。

● H:半进位

在执行ADD或ADC操作的过程中, 当ALU的第3位和第4位间发生进位时, H位会被置1, 这对于BCD码算术运算很有意义。

● I0:中断屏蔽级别0

请参考表1。

● N:负数

当上一次的算术、逻辑或数据操作的结果是负的情况下, N位被置1(例如结果的最高位是逻辑1)。

● Z:零

当上一次的算术、逻辑或数据操作的结果是零时, Z位被置1。

● C:进位

在上一次的算术操作中, 如果结果的最高位发生进位或借位, 则当该位被置1。当执行位测试, 分支, 移位, 旋转和加载指令时, 该位也会受到影响。请参考ADD, ADC, SUB, SBC等指令。

在除法操作中, C位用来指示在指令执行中是否有错误发生(商溢出或0作除数)。请参考DIV指令。

在位测试操作中, 被测试的位被复制到C位; 请参考BTJF, BTJT指令。在移位和旋转操作中, C位根据结果进行相应地更新; 请参考RRC, RLC, SRL, SLL, SRA指令。

用户还可以通过SCF, RCF, CCF指令对C位进行置位, 清除和取反。

例子: 加法操作

$$\$B5 + \$94 = "C" + \$49 = \$149$$

C	7	0
0	1 0 1 1 0 1 0 1	
+	0	
	1 0 0 1 0 1 0 0	
=	0 1 0 0 1 0 0 1	

1.2.2 STM8 CPU寄存器映射

CPU寄存器在STM8的地址空间映射如表2所示。只有CPU的调试模块才可以使用这些寄存器的地址对其操作, 在CPU核内执行的指令只能通过直接使用寄存器名才可以读写这些寄存器。

表2 CPU寄存器映射

地址	寄存器名	7	6	5	4	3	2	1	0
00 7F00h	A	MSB	—	—	—	—	—	—	LSB
00 7F01h	PCE	MSB	—	—	—	—	—	—	LSB
00 7F02h	PCH	MSB	—	—	—	—	—	—	LSB
00 7F03h	PCL	MSB	—	—	—	—	—	—	LSB
00 7F04h	XH	MSB	—	—	—	—	—	—	LSB
00 7F05h	XL	MSB	—	—	—	—	—	—	LSB
00 7F06h	YH	MSB	—	—	—	—	—	—	LSB
00 7F07h	YL	MSB	—	—	—	—	—	—	LSB
00 7F08h	SPH	MSB	—	—	—	—	—	—	LSB

1.3 全局配置寄存器(CFG_GCR)

1.3.1 激活级别

用户可通过对CFG_GCR寄存器中的AL位编程来配置MCU的激活级别。如何使用该位请参考[10.4活动等级/低功耗模式的控制](#)。

1.3.2 禁用SWIM

在MCU复位后的默认情况下，SWIM引脚被配置为可以通过SWIM协议和外部工具通讯来对CPU调试或对FLASH/EEPROM编程。当CFG_GCR寄存器的SWD位被置1时，SWIM引脚被配置为普通I/O口。

1.3.3 全局配置寄存器(CFG_GCR)描述

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0					
保留区						AL	SWD					
							rw					
							rw					
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">位7:2</td> <td>保留位，必须保持为0</td> </tr> <tr> <td>位1</td> <td> AL 激活级别 该位可由软件来置位或清零。该位用于配置主激活级别或中断激活级别。 0: 主激活级别。执行IRET指令会使CPU上下文从堆栈中恢复出来；在执行WFI指令后主程序会继续执行。 1: 中断激活级别。执行IRET指令会导致CPU重新回到WFI/HALT模式，此时并不从堆栈中恢复CPU上下文。 </td> </tr> <tr> <td>位0</td> <td> SWD: 禁用SWIM 0: SWIM模式被使能。 1: SWIM模式被禁用。 当SWIM模式被使能时，SWIM引脚不能被用作普通I/O口。 </td> </tr> </table>							位7:2	保留位，必须保持为0	位1	AL 激活级别 该位可由软件来置位或清零。该位用于配置主激活级别或中断激活级别。 0: 主激活级别。执行IRET指令会使CPU上下文从堆栈中恢复出来；在执行WFI指令后主程序会继续执行。 1: 中断激活级别。执行IRET指令会导致CPU重新回到WFI/HALT模式，此时并不从堆栈中恢复CPU上下文。	位0	SWD : 禁用SWIM 0: SWIM模式被使能。 1: SWIM模式被禁用。 当SWIM模式被使能时，SWIM引脚不能被用作普通I/O口。
位7:2	保留位，必须保持为0											
位1	AL 激活级别 该位可由软件来置位或清零。该位用于配置主激活级别或中断激活级别。 0: 主激活级别。执行IRET指令会使CPU上下文从堆栈中恢复出来；在执行WFI指令后主程序会继续执行。 1: 中断激活级别。执行IRET指令会导致CPU重新回到WFI/HALT模式，此时并不从堆栈中恢复CPU上下文。											
位0	SWD : 禁用SWIM 0: SWIM模式被使能。 1: SWIM模式被禁用。 当SWIM模式被使能时，SWIM引脚不能被用作普通I/O口。											

1.3.4 全局配置寄存器映射及复位值

CFG_GCR在STM8地址空间的映射如[表2](#)。

表3 CFG_GCR寄存器映射

地址偏移值	寄存器名	7	6	5	4	3	2	1	0
0x00	CFG_GCR	—	—	—	—	—	—	AL	SWD
	复位值	0	0	0	0	0	0	0	0

2 用于系统启动的只读存储器

在某些STM8型号中有2K字节的内部BOOT ROM，其中包含有用于启动的代码。这段代码的主要作用是利用STM8的SPI, CAN或UART接口，将应用程序代码，数据，选项字节(Option byte)和中断向量表下载到内部的FLASH和EEPROM中去。

在复位以后，启动代码开始执行。更多详细内容请参考STM8 启动代码用户手册 (UM0560)。

3 存储器和寄存器映射

要了解关于存储器映射、I/O端口硬件寄存器映射以及CPU/SWIM/调试模块/中断控制寄存器的详细内容请参考产品数据手册。

3.1 寄存器描述缩写

在本参考手册每一章的寄存器描述中，使用下列缩写：

read/write (rw)	该位可以进行读写操作。
read-only (r)	该位只能进行读操作。
write only (w)	该位只能进行写操作。读操作的返回值无意义。
read/write once (rwo)	该位只能被写一次，但可以在任意时刻进行读操作。只有系统复位才可以使该位恢复为复位值。
read/clear (rc_w1)	读该位和向该位写1都可以使该位为0。写0对该位无影响。
read/clear (rc_w0)	读该位和向该位写0都可以使该位为0。写1对该位无影响。
read/set (rs)	读该位和向该位写1都可以使该位为1。写0对该位无影响。

4 FLASH程序存储器和数据EEPROM

4.1 介绍

STM8内部的FLASH程序存储器和数据EEPROM由一组通用寄存器来控制。用户可以使用这些寄存器来编程或擦除存储器的内容、设置写保护、或者配置特定的低功耗模式。用户也可以对器件的选项字节(Option byte)进行编程。

4.2 词汇表

- 块(BLOCK)

一个块是指可由一个简单编程操作编程或擦除的一组字节。块级的操作非常快，是标准的编程和擦除操作。请参考[表4](#)来了解块的大小。

- 页(PAGE)

一页由一组块组成。STM8S 器件拥有启动代码，程序代码和数据EEPROM，这些区域都由特定的结构所保护。通过对特定的选项字节进行操作，这些区域的大小能够以页为单位来进行调整。

4.3 FLASH主要特性

- STM8S EEPROM分为两个存储器阵列：

- 最多至 128K字节的FLASH程序存储器，不同的器件容量有所不同。请参考[4.4存储器组织结构](#)了解更多细节。
- 最多至 2K字节的数据EEPROM(包括option byte—选择字节)，不同的器件容量有所不同。请参考[4.4存储器组织结构](#)了解更多细节。

- 编程模式

- 字节编程和自动快速字节编程(没有擦除操作)
- 字编程
- 块编程和快速块编程(没有擦除操作)
- 在编程/擦除操作结束时和发生非法编程操作时产生中断

- 读同时写(RWW)功能。该特性并不是所有STM8S器件都拥有。请参考具体的数据手册了解更多细节。

- 在应用编程(IAP)和在线编程(ICP)能力。

- 保护特性

- 存储器读保护(ROP)
- 基于存储器存取安全系统(MASS 密钥)的程序存储器写保护
- 基于存储器存取安全系统(MASS 密钥)的数据存储器写保护
- 可编程的用户启动代码区域(UBC)写保护

- 在待机(Halt)模式和活跃待机(Active-halt)模式下，存储器可配置为运行状态和掉电状态。

4.4 存储器组织结构

STM8S的EEPROM以32位字长(每字4字节)为基础组织起来。根据不同的器件，存储器组织机构有所不同：

- 小容量STM8S器件

- 8K FLASH 程序存储器，每页 64 字节，共 128 页
- 640 字节数据 EEPROM，每页 64 字节，共 10 页。数据 EEPROM 包括一页的选项字节(64 字节)。

- 中容量STM8S器件

- 从 16K 到 32K FLASH 程序存储器，每页 512 字节，最多 64 页

- 1K字节数据 EEPROM，每页 512字节，共 2页。数据 EEPROM 包括一页的选项字节 (512字节)。

- 大容量STM8S器件

- 从 64K 到 128K FLASH 程序存储器，每页 512字节，最多 256页
- 从 1K 到 2K 字节数据 EEPROM，每页 512字节，共 4页。数据 EEPROM 包括一页的选项字节(512字节)。

页的大小定义了用户启动代码区域(UBC)大小的最小可调整值。请参考 [4.4.1 用户启动区域\(UBC\)](#)。

[图3](#)和[图4](#)展示了STM8S系列FLASH存储器和数据EEPROM的组织机构。

图3 小容量STM8S的FLASH存储器和数据EEPROM组织机构

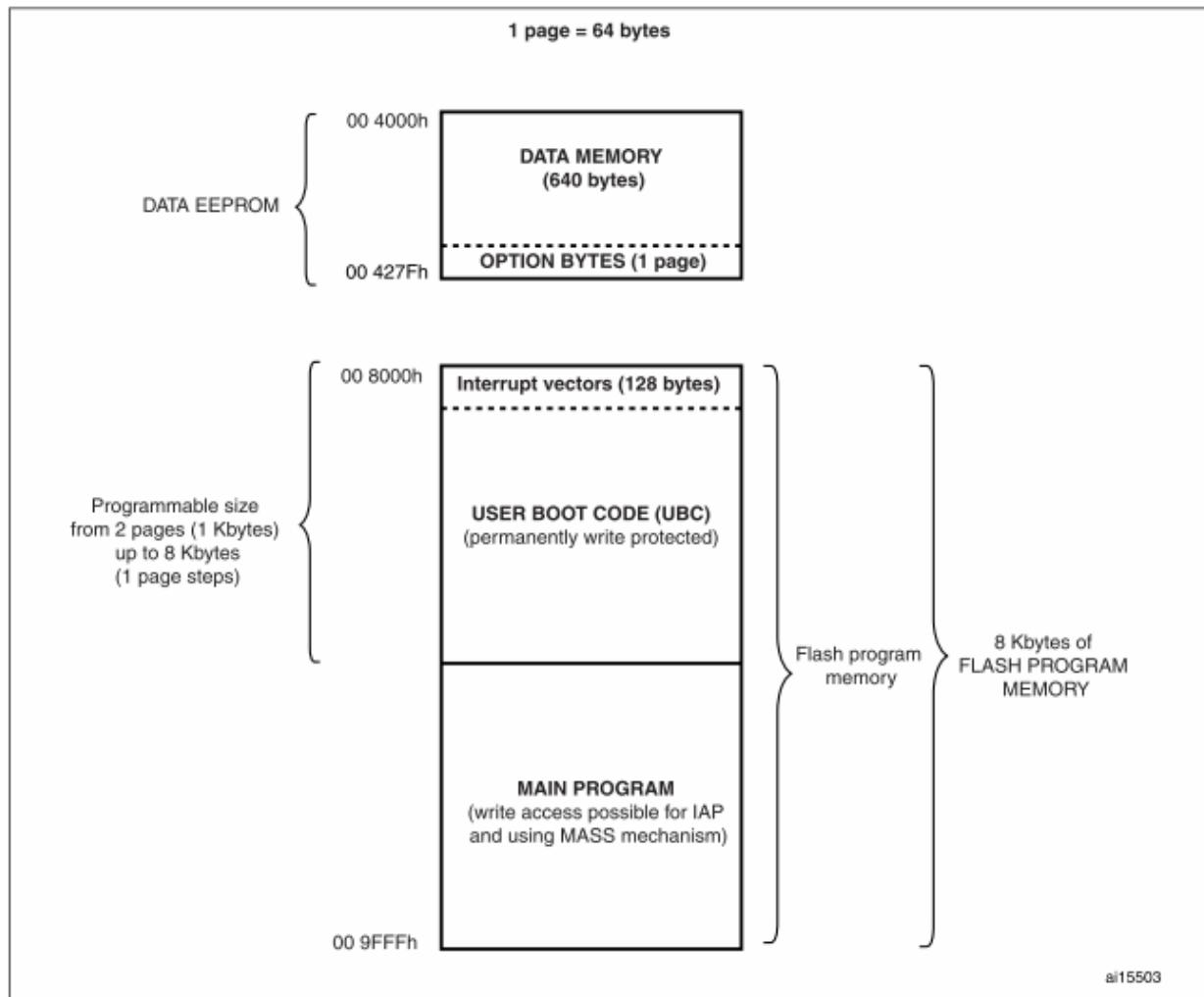


图4 中容量STM8S的FLASH存储器和数据EEPROM组织机构

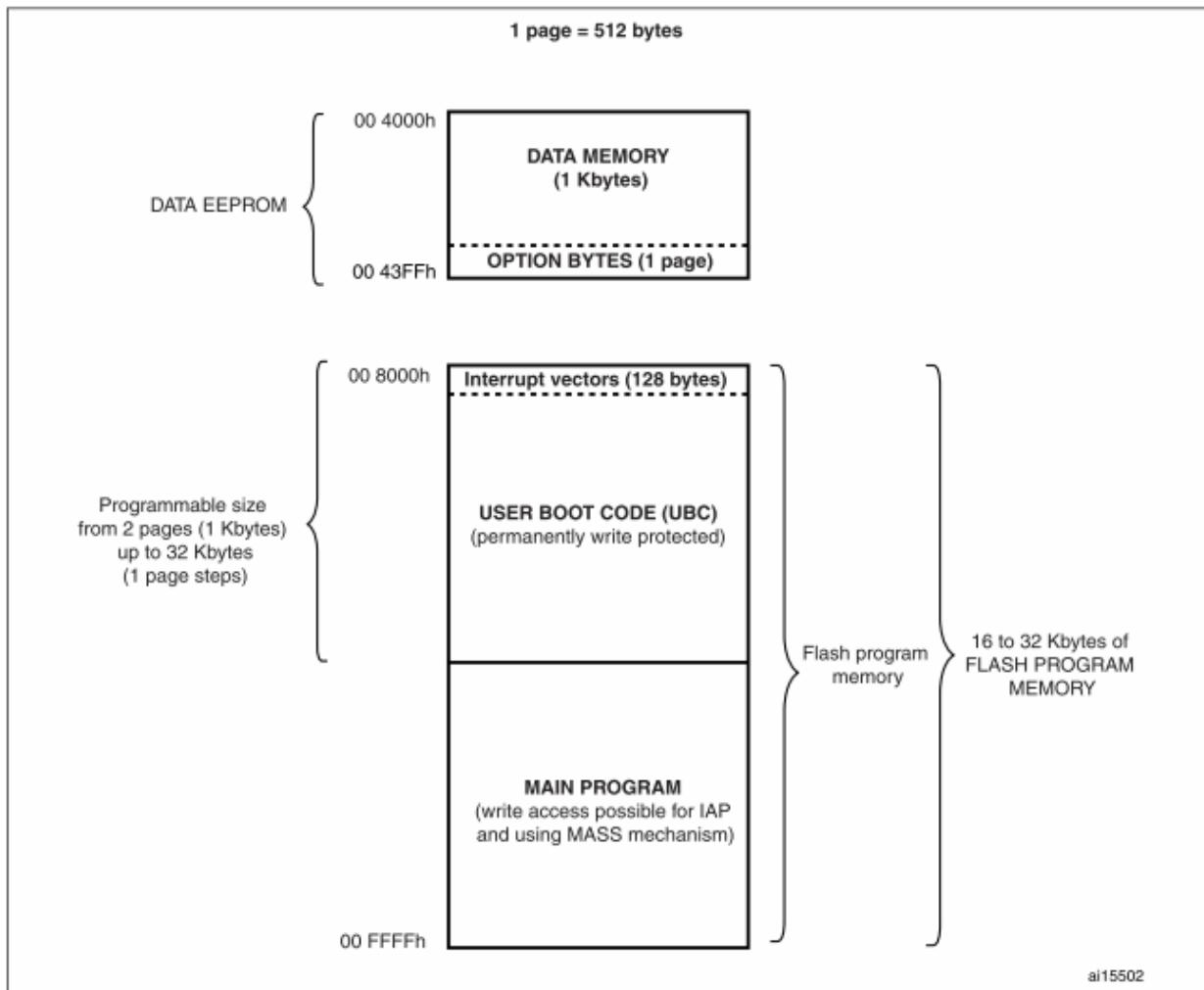
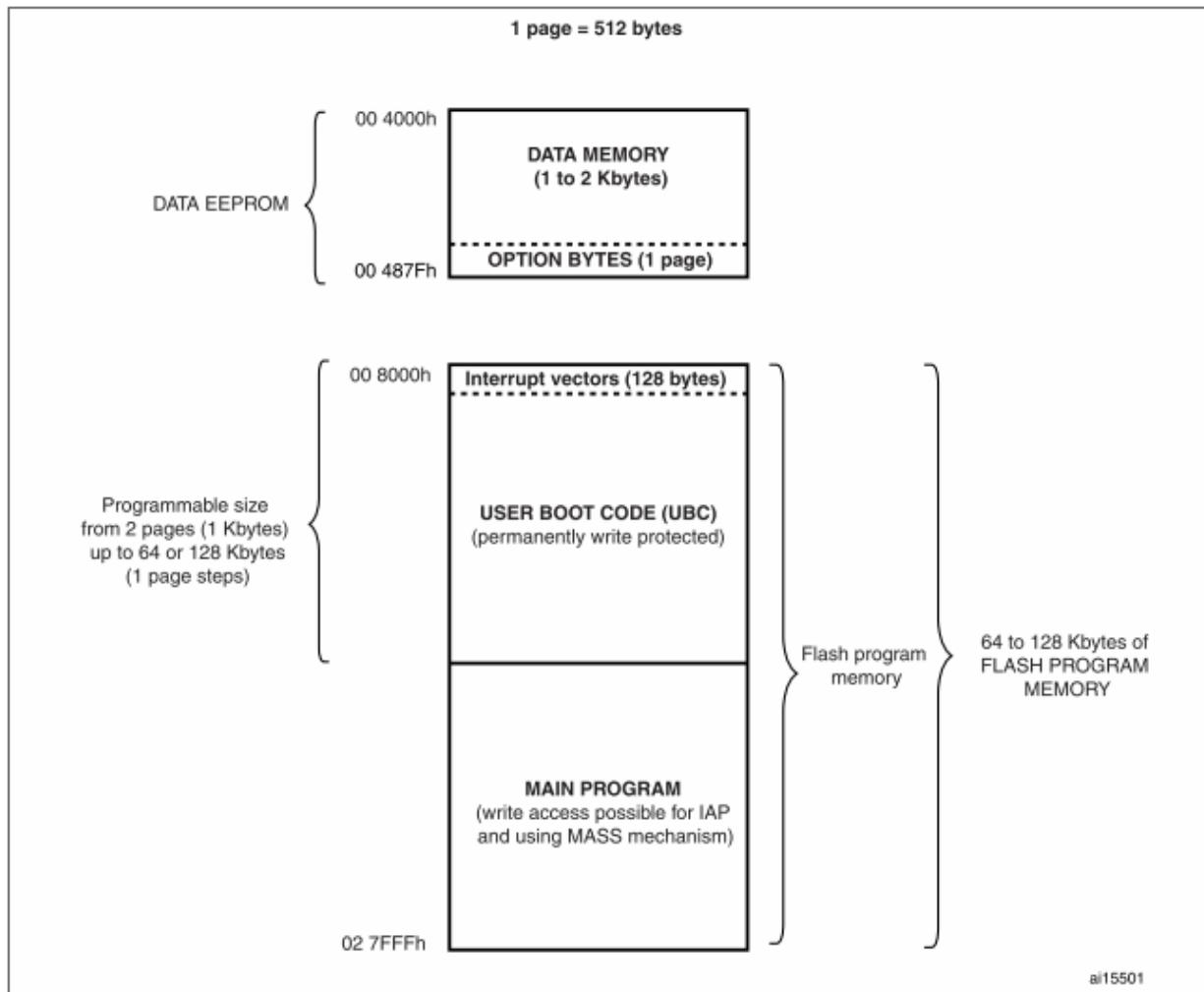


图5 大容量STM8S的FLASH存储器和数据EEPROM组织机构



4.4.1 用户启动区域(UBC)

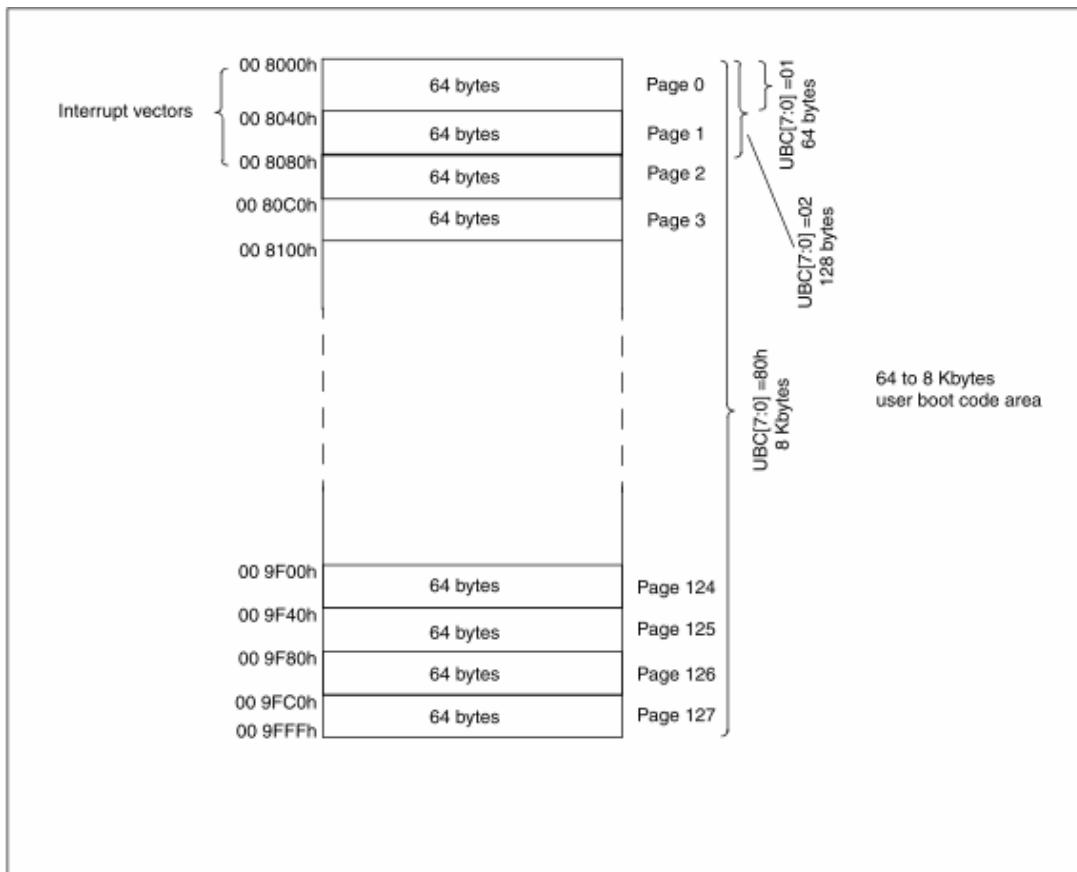
用户启动区域(UBC)包含有复位和中断向量表，它可用于存储IAP及通讯程序。UBC有一个两级保护结构可保护用户代码及数据在IAP编程中免于无意的擦除或修改。这意味着该区域总是写保护的，而且写保护不能通过使用MASS密钥来解锁。

在ICP模式下(使用SWIM接口)可以通过修改选项字节来配置UBC的大小。UBC选项字节指定了分配在UBC中的页的数量。UBC区域的起始地址是0x00 8000。

可以通过读取UBC选项字节来获得UBC区域的大小。

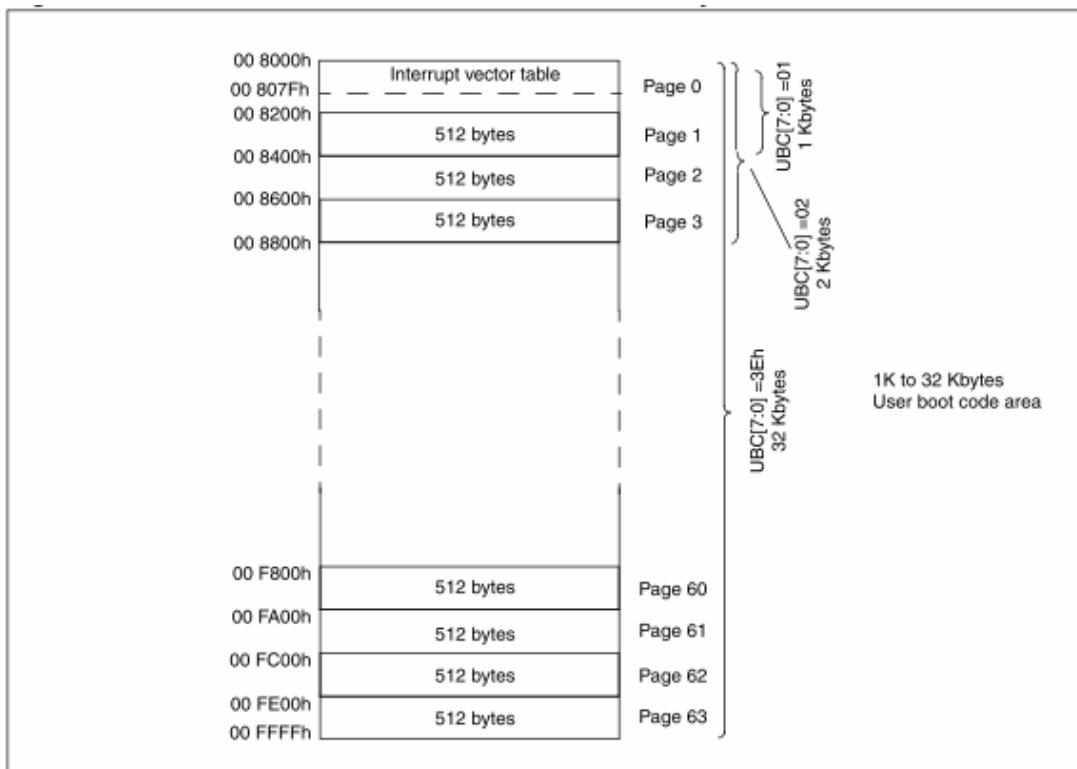
请参考[图6](#), [图7](#)和[图8](#)来了解UBC区域的存储器映射。对于选项字节部分，请参考相应的数据手册了解更多的UBC选项字节的细节。

图6 小容量STM8S的UBC区域大小



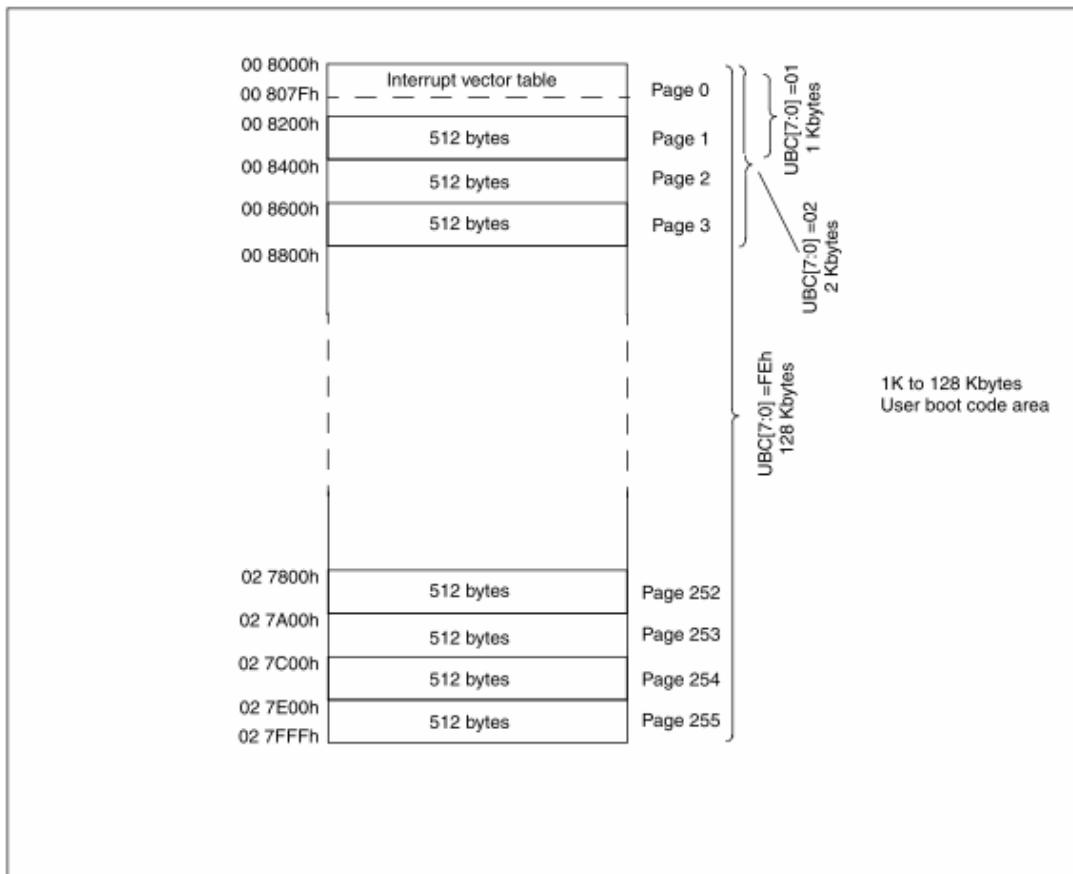
1. UBC[7:0]=0x00意味着没有定义用户启动区域。请参考相应的数据手册了解UBC选项字节的细节。
2. 头两页(128字节)包含中断向量表。

图7 中容量STM8S的UBC区域大小



1. UBC[7:0]=0x00意味着没有定义用户启动区域。请参考相应的数据手册了解UBC选项字节的细节。
2. 头两页(1K字节)包含中断向量表。中断向量表只占用128字节(32个中断向量)。

图8 大容量STM8S的UBC区域大小



1. UBC[7:0]=0x00意味着没有定义用户启动区域。请参考相应的数据手册了解UBC选项字节的细节。
2. 头两页(1K字节)包含中断向量表。中断向量表只占用128字节(32个中断向量)。

4.4.2 数据EEPROM(DATA)

数据EEPROM(DATA)区域可用于存储用户具体项目所需的数据。默认情况下，DATA区域是写保护的，这样可以在主程序工作在IAP模式时防止DATA区域被无意地修改。只有使用特定的MASS密钥才能对DATA区域的写保护解锁(请参考[对DATA区域的写操作](#))。

请参考[4.4存储器组织结构](#)来了解不同的STM8S MCU的DATA区域大小。

4.4.3 主程序区

主程序区是指在FLASH程序存储器中用于存储应用代码的区域(请参考[图3](#)、[图3](#)和[图4](#))。

4.4.4 选项字节(Option byte)

选项字节用于配置硬件特性和存储器保护状态，这些字节位于同一页的特定存储器阵列中。

选项字节可以在ICP/SWIM模式中或IAP模式中修改，注意此时要保证FLASH_CR2中的OPT位为1以及FLASH_NCR2中的NOPT位为0(请参考[4.9.2 FLASH控制寄存器2\(FLASH_CR2\)](#)和[4.9.3 FLASH互补控制寄存器2\(FLASH_NCR2\)](#))。

请参考数据手册中的选项字节部分了解更多关于选项字节的细节，并请参考STM8 SWIM 通讯协议和调试模块用户手册(UM0470)来了解如何对选项字节编程。

4.5 存储器保护

4.5.1 读保护

当选项字节中的ROP字节被编程为'0xAA'时，读保护就生效了。这种情况下，无论写保护是否生效，在ICP模式中(使用SWIM接口)读取或修改FLASH程序存储器和DATA区域都是被禁止的。即使认为没有什么保护是完全不可破解的，对于一个通用微处理器来说，STM8的读保护的特性也提供了一个非常高水平的保护级别。

可以在ICP模式中通过对选项字节中的ROP字节重新编程来解除程序存储器、UBC和DATA区域的读保护。在这种情况下，程序存储器、UBC、DATA区域以及选项字节都被自动擦除，器件也可以被重新编程了。

请参考[表5不同编程模式下的存储器存取\(1\)](#) 来了解当读保护使能或禁止时存储器存取的更多细节。

4.5.2 存储器存取安全系统(MASS)

在复位以后，主程序和DATA区域都被自动保护以防止无意的写操作。在试图修改其内容前必须对其解锁，而解锁的机制由存储器存取安全系统(MASS)来管理。

UBC区域的特性指明了在UBC中的内容一直是写保护的(请参考[4.4.1 用户启动区域\(UBC\)](#))。

一旦存储器内容被修改完毕，推荐将写保护使能以防止数据被破坏。

对主程序存储器的写操作

在器件复位后，可以通过向FLASH_PUKR寄存器连续写入两个被叫作MASS密钥的值来解除主程序存储器的写保护(请参考[4.9.6 FLASH程序存储器解保护寄存器\(FLASH_PUKR\)](#))。这两个写入FLASH_PUKR的值会和以下两个硬件密钥值相比较：

- 第一个硬件密钥: 0b0101 0110 (0x56)
- 第二个硬件密钥: 0b1010 1110 (0xAE)

需要通过如下步骤来解除主程序存储器区域的写保护：

1. 向FLASH_PUKR写入第一个8位密钥。在系统复位后，当这个寄存器被首次写入值时，数据总线上的值没有被直接锁存到这个寄存器中，而是和第一个硬件密钥值(0x56)相比较。
2. 如果密钥输入错误，FLASH_PUKR寄存器在下一次系统复位之前将一直被锁住。在下一次复位前，再向该寄存器进行的任何写操作都会被系统忽略掉。
3. 如果第一个硬件密钥正确，当这个寄存器被第二次写入值时，数据总线上的值没有被直接锁存到这个寄存器中，而是和第二个硬件密钥值(0xAE)相比较。
4. 如果密钥输入错误，FLASH_PUKR寄存器在下一次系统复位之前将一直被锁住。在下一次复位前，再向该寄存器进行的任何写操作都会被系统忽略掉。
5. 如果第二个硬件密钥正确，主程序存储器写保护被解除，同时FLASH_IAPSR中的PUL位为1(请参考[4.9.8 FLASH状态寄存器\(FLASH_IAPSR\)](#))。

在开始编程之前，应用程序可以校验PUL位是否被有效地置1。应用程序可以在任意时刻通过清PUL位来重新禁止对FLASH程序区域的写操作。

对DATA区域的写操作

在器件复位后，可以通过向FLASH_DUKR寄存器连续写入两个被叫作MASS密钥的值来解除DATA区域的写保护(请参考[4.9.7](#))。这两个写入FLASH_DUKR的值会和以下两个硬件密钥值相比较：

- 第一个硬件密钥: 0b0101 0110 (0x56)
- 第二个硬件密钥: 0b1010 1110 (0xAE)

需要通过如下步骤来解除数据区域的写保护：

1. 向FLASH_DUKR写入第一个8位密钥。在系统复位后，当这个寄存器被首次写入值时，数据总线上的值没有被直接锁存到这个寄存器中，而是和第一个硬件密钥值(0x56)相比较。
2. 如果密钥输入错误，应用程序可以尝试重新输入这两个MASS密钥来对DATA区域进行解锁。
3. 如果第一个硬件密钥正确，当这个寄存器被第二次写入值时，数据总线上的值没有被直接锁存到这个寄存器中，而是和第二个硬件密钥值(0xAE)相比较。
4. 如果密钥输入错误，DATA EEPROM区域在下一次系统复位之前将一直保持写保护状态。在下一次复位前，再向该寄存器进行的任何写操作都会被系统忽略掉。
5. 如果第二个硬件密钥正确，DATA区域的写保护被解除，同时FLASH_IAPSR中的DUL位为1(请参考[4.9.8 FLASH状态寄存器\(FLASH_IAPSR\)](#))。

在开始编程之前，应用程序可以通过校验DUL位是否被有效地置1来确认DATA区域已经将写保护解锁。应用程序可以在任意时刻通过清DUL位来重新禁止对DATA区域的写操作。

4.5.3 对选项字节的写操作

对选项字节的写操作的步骤和对DATA EEPROM的操作大致相同。但是要注意到FLASH_CR2中的OPT位要为1以及FLASH_NCR中的NOPT位要为0，这样才可以对选项字节进行写操作。

4.6 存储器编程

在尝试执行任何编程操作之前，必须对主程序存储器和DATA区域解锁。将要被编程的存储器区域的解锁机制在[4.5.2 存储器存取安全系统\(MASS\)](#)中描述。

4.7 读同时写 (RWW)

RWW特性允许用户在执行程序和读程序存储器时对DATA EEPROM区域进行写操作，因此执行的时间被优化了。相反的操作是不允许的：即你不可以在写程序存储器时对DATA EEPROM进行读操作。

RWW特性是一直有效的而且可以在任意时刻使用。

注意：并不是所有STM8都拥有RWW特性，请参考相应的数据手册来了解更多信息。

4.7.1 字节编程

可以对主程序存储器和DATA区域逐字节地编程。要对一个字节编程，应用程序可直接向目标地址写入数据。

- 在主程序存储器中

当字节编程操作执行时，应用程序停止运行。

- 在DATA区域中

- 有RWW功能的器件：在IAP模式下，应用程序不停止运行，字节编程利用RWW功能进行操作。
- 无RWW功能的器件：当字节编程操作执行时，应用程序停止运行。

要擦除一个字节，向对应的字节简单写入'0x00'即可。

应用程序可以通过读FLASH_IAPSR寄存器来校验编程或擦除操作是否已被正确执行：

- 在一次成功的编程操作后EOP位被置1。
- 当软件试图对一个被保护的页进行写操作时WP_PG_DIS位被置1。在这种情况下，写操作不会被执行。

如果FLASH_CR1中的IE位已经被预先使能，则只要这些标志位(EOP/WP_PG_DIS)中有一个被置位就会产生一个中断。

自动快速字节编程

根据目标地址的初始化内容的不同，编程持续时间可能也有所不同。如果字(4个字节)中包含不为空的字节，在编程前整个字会被自动擦除。相反，如果字为空，由于不会执行擦除操作从而编程时间变短(请参考 t_{PROG} 参数，在数据手册的“Flash program memory”表中)。

然而，可以通过对FLASH_CR1中的FIX位置1来强迫执行系统擦除操作而不管其内容是否为空，从而使编程时间固定(请参考FLASH控制寄存器)。编程总时间随之被规定为擦除时间和写操作时间的和(请参考 t_{PROG} 参数，在数据手册的“Flash program memory”表中)。

注意：为了快速写一个字节(没有擦除操作)，将要被写入数据的整个字(4个字节)必须被预先擦除。因此不可能对同一个字做连续两次快速写操作(在第二次写之前没有擦除操作)：第一次写字节操作将是快速操作但针对另外一个字节的第二次写操作将需要一个擦除操作。

4.7.2 字编程

字写入操作允许一次对整个4字节的字进行编程，从而将编程时间缩短。

主程序存储器和DATA EEPROM都可以进行字操作。在一些STM8S器件中，也拥有当DATA EEPROM在进行写操作时同时具备RWW功能。请参考数据手册了解更多信息。

为了对一个字编程，FLASH_CR2和FLASH_NCR2中的WPRG/NWPRG位必须预先置位/清零来使能字编程模式(请参考4.9.2 FLASH控制寄存器2(FLASH_CR2)和4.9.3 FLASH互补控制寄存器2(FLASH_NCR2))。然后将要被编程字的4个字节必须被从首地址开始装载。当四个字节都被写入后，编程周期自动开始。

像字节操作一样，FLASH_IAPSR中的EOP与WR_PG_DIS控制位和FLASH中断相配合，可用于检查操作是否被正确执行完毕。

4.7.3 块编程

块编程比字节编程和字编程都要快。在块编程操作中，整个块的编程或擦除在一个编程周期就可以完成。请参考表4了解具体器件的块的大小。

在主程序存储器和DATA区域都可以执行块操作。

- 在主程序存储器中

用于块编程的代码必须全部在RAM中执行。

- 在DATA区域中

- 有RWW功能的器件：DATA块操作可在主程序存储器中执行，然而数据装载阶段(下文中有述)必须在RAM中执行。
- 无RWW功能的器件：用于块编程的代码必须全部在RAM中执行。

一共有三种可能的块操作：

- 块编程(也叫标准块编程)：整个块在编程前被自动擦除。
- 快速块编程：在编程前没有预先的块擦除操作。
- 块擦除。

在块编程时，中断被硬件自动屏蔽。

标准块编程

块编程操作允许一次对整个块进行编程，整个块在编程前被自动擦除。

为了对整个块编程，FLASH_CR2和FLASH_NCR2中的PRG/NPRG位必须预先置位/清零来使能标准块编程(请参考4.9.2 FLASH控制寄存器2(FLASH_CR2)和4.9.3 FLASH互补控制寄存器2(FLASH_NCR2))。然后需要向主程序存储器或DATA区域的目标地址依次写入要编程的数据，这样数据会被锁存在内部缓存中。为编程整个块，块中的所有字节都需要被写入数据。但要注意，所有被写入缓存的数据必须位于同一个块中，这意味着这些数据必须有同样的高位地址：仅仅低6位的地址可以不一样。当目标块的最后一个字节被装载到缓存后，编程就自动开始了。编程前首先会自动执行一次擦除操作。

当对DTA区域进行块编程时，应用程序可以检查FLASH_IAPSR中的HVOFF位确认编程状态。一旦HVOFF被置0，真正的编程阶段就开始了，此时应用程序就可以返回到主程序中去了。FLASH_IAPSR中的EOP与WR_PG_DIS控制位和FLASH中断相配合，可用于检查操作是否被正确执行完毕。

快速块编程

快速块编程允许不擦除存储器内容就对块进行编程，因此快速块编程的编程速度是标准块编程的两倍。

该模式仅用于被编程部分已经被擦除过的情况，同时这种模式对向空白部分写入完整的应用代码特别有用，因为这种模式可以节省相当可观的时间。

快速块编程的步骤和标准块编程的步骤大致一样，FLASH_CR2和FLASH_NCR2中的FPRG/NFPRG位必须预先置位/清零来使能快速块编程(请参考[4.9.2 FLASH控制寄存器2\(FLASH_CR2\)](#)和[4.9.3 FLASH互补控制寄存器2\(FLASH_NCR2\)](#))。

FLASH_IAPSR中的EOP与WR_PG_DIS控制位和FLASH中断相配合，可用于检查快速块编程操作是否被正确执行完毕。

警告：在执行快速块编程之前如果这个块不是空的话，不能保证写入的数据无误。

块擦除

块擦除允许擦除整个块。

为了擦除整个块，FLASH_CR2和FLASH_NCR2中的ERASE/NERASE位必须预先置位/清零来使能块擦除(请参考[4.9.2 FLASH控制寄存器2\(FLASH_CR2\)](#)和[4.9.3 FLASH互补控制寄存器2\(FLASH_NCR2\)](#))。通过对块中所有的字写入‘0x00 00 00 00’来擦除整个块。字的起始地址必须以‘0’，‘4’，‘8’，或‘C’作为结尾。

FLASH_IAPSR中的EOP与WR_PG_DIS控制位和FLASH中断相配合，可用于检查操作是否被正确执行完毕。

表4 块的大小

STM8 微控制器家族	块的大小
小容量 STM8S	64字节
小容量 STM8S	128字节
小容量 STM8S	128字节

4.7.4 选项字节(Option byte)编程

对选项字节编程和对DATA EEPROM区域编程非常相似。

应用程序可直接向目标地址进行写操作。利用STM8的RWW功能，在对选项字节写操作的同时程序不必停下来。

请参考相应的数据手册来了解选项字节内容的细节。

4.8 ICP和IAP

在线编程(ICP)用于更新整个存储器的内容。ICP使用SWIM接口把用户的程序装载到微控制器中，同时提供迅速而有效的设计迭代并且去除了不必要的封装处理和器件插槽。SWIM接口(单线接口模块)使用SWIM引脚和编程工具相连接。

相对于ICP方式，在应用编程(IAP)可使用STM8支持的任意通讯接口(I/O、I2C、SPI、UART...)来下载要编入存储器中的数据。IAP允许在应用程序运行中对FLASH程序存储器的内容重新编程。然而要想使用IAP，必须通过ICP对FLASH程序存储器预先编程。

请参考STM8 Flash 编程手册(PM0051)和STM8 SWIM通信协议和调试模块用户手册 (UM0470)来了解关于编程步骤的更多细节。

表5 不同编程模式下的存储器存取⁽¹⁾

模式	读保护	存储器区域	存取
用户模式和 IAP 启动代码 (如果有的话)	读保护使能	用户启动代码区域 (UBC)	R/E
		主程序	R/W/E ⁽²⁾
		DATA EEPROM 区域 (DATA)	R/W/E ⁽³⁾
		选项字节 (Option byte)	R
	读保护禁止	用户启动代码区域 (UBC)	R/E ⁽⁴⁾
		主程序	R/W/E ⁽²⁾
		DATA EEPROM 区域 (DATA)	R/W/E ⁽³⁾
		选项字节 (Option byte)	R/W ⁽⁵⁾
ICP 和 SWIM	读保护使能	用户启动代码区域 (UBC)	P
		主程序	P
		DATA EEPROM 区域 (DATA)	P
		选项字节 (Option byte)	P/W _{ROP} ⁽⁶⁾
	读保护禁止	用户启动代码区域 (UBC)	R/E ⁽⁴⁾
		主程序	R/W/E ⁽²⁾
		DATA EEPROM 区域 (DATA)	R/W/E ⁽³⁾
		选项字节 (Option byte)	R/W

1. R/W/E = 读; 写和运行;

R/E = 读和运行(写操作被禁止);

R = 读(写操作和运行被禁止);

P = 该区域不可存取(读; 写和运行被禁止);

P/W_{ROP} = 被保护。除ROP选项字节外, 写操作被禁止。

- 在向FLASH_PUKR写入正确的MASS密钥之前, Flash程序存储器是写保护的(锁住)。可以通过清PUL位来重新锁住该区域, 但在两次复位之间仅可解锁一次。
- 在向FLASH_DUKR写入正确的MASS密钥之前, DATA存储器是写保护的(锁住)。可以通过清DUL位来重新锁住该区域。
- 如果想对UBC区域编程, 首先要清除UBC对应的选项字节位。
- 在向FLASH_DUKR写入正确的MASS密钥(同时OPT位要置1)之前, 选项字节是写保护的(锁住)。可以通过清DUL位来重新锁住该区域。
- 当ROP位被清除, 整个存储器(包括选项字节)被自动擦除。

4.9 FLASH寄存器

4.9.1 FLASH控制寄存器 1(FLASH_CR1)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
			Reserved	HALT	AHALT	IE	FIX
				rw	rw	rw	rw

位7:4	保留位, 必须保持为0
位3	HALT: 停机(Halt)模式下掉电 该位可由软件来置位或清零。 0: 当MCU在停机(Halt)模式时FLASH处于掉电模式 1: 当MCU在停机(Halt)模式时FLASH处于运行模式
位2	AHALT: 活跃停机(Active halt)模式下掉电 0: 当MCU在活跃停机模式时FLASH处于掉电模式 1: 当MCU在活跃停机模式时FLASH处于运行模式
位1	IE: FLASH中断使能 0: 中断禁止 1: 中断使能。当FLASH_IAPSR寄存器中的EOP或WR_PG_DIS位被置位时产生中断
位0	FIX: 固定的编程时间 0: 当存储器已经被擦除过时, 编程时间为标准编程时间的一半($1/2 t_{prog}$), 否则为标准的编程时间 t_{prog} 。 1: 编程时间固定为标准编程时间 t_{prog} 。

4.9.2 FLASH控制寄存器 2(FLASH_CR2)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
OPT	WPRG	ERASE	FPRG		Reserved		PRG
rw	rw	rw	rw				ro

位7	OPT: 对选项字节进行写操作 该位可由软件来置位或清零。 0: 对选项字节进行写操作被禁止 1: 对选项字节进行写操作被使能
位6	WPRG: 字编程 当操作完成时, 该位由硬件来置位或清零。 0: 字编程操作被禁止 1: 字编程操作被使能
位5	ERASE⁽¹⁾: 块擦除 当操作完成时, 该位由硬件来置位或清零。 0: 块擦除操作被禁止 1: 块擦除操作被使能
位4	FPRG⁽¹⁾: 快速块编程 当操作完成时, 该位由硬件来置位或清零。 0: 快速块编程操作被禁止 1: 快速块编程操作被使能
位3:1	保留位
位0	PRG: 标准块编程 当操作完成时, 该位由硬件来置位或清零。 0: 标准块编程操作被禁止 1: 标准块编程操作被使能

1.当存储器忙时, ERASE和FPRG位被锁住。

4.9.3 FLASH互补控制寄存器 2(FLASH_NCR2)

地址偏移值: 0x02

复位值: 0xFF

7	6	5	4	3	2	1	0
NOPT	NWPRG	NERASE	NFPRG		Reserved		NPRG
rw	rw	rw	rw				rw
位7	NOPT: 对选项字节进行写操作 该位可由软件来置位或清零。 0: 对选项字节进行写操作被使能 1: 对选项字节进行写操作被禁止						
位6	NWPRG: 字编程 当操作完成时, 该位由硬件来置位或清零。 0: 字编程操作被使能 1: 字编程操作被禁止						
位5	NERASE: 块擦除 当操作完成时, 该位可由软件来置位或清零。 0: 块擦除操作被使能 1: 块擦除操作被禁止						
位4	NFPRG: 快速块编程 当操作完成时, 该位可由软件来置位或清零。 0: 快速块编程操作被使能 1: 快速块编程操作被禁止						
位3:1	保留位						
位0	NPRG: 标准块编程 当操作完成时, 该位可由软件来置位或清零。 0: 标准块编程操作被使能 1: 标准块编程操作被禁止						

4.9.4 FLASH保护寄存器(FLASH_FPR)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
Reserved		WPB5	WPB4	WPB3	WPB2	WPB1	WPB0
		ro	ro	ro	ro	ro	ro

位7:6	保留位, 必须保持为 '0'
位5:0	WPB[5:0]: 用户启动代码保护位 这些位指示用户启动代码的大小, 其值在启动时从UBC选项字节装载。请参考数据手册来了解关于保护页部分的细节。

4.9.5 FLASH保护寄存器(FLASH_NFPR)

地址偏移值: 0x04

复位值: 0xFF

7	6	5	4	3	2	1	0
Reserved		NWPB5	NWPB4	NWPB3	NWPB2	NWPB1	NWPB0
		ro	ro	ro	ro	ro	ro

位7:6	保留位, 必须保持为‘1’
位5:0	NWPB[5:0]: 用户启动代码保护位 这些位指示用户启动代码的大小, 其值为NUBC选项字节的对应字节。请参考数据手册来了解关于保护页部分的细节。

4.9.6 FLASH程序存储器解保护寄存器(FLASH_PUKR)

地址偏移值: 0x08

复位值: 0x00

7	6	5	4	3	2	1	0
MASS_PRG KEYS							
RW	RW	RW	RW	RW	RW	RW	RW

位7:0

PUK[7:0]: 主程序存储器解锁密钥

该位可由软件来进行写操作(在任何模式下)。当读该寄存器时, 返回值为0x00。请参考[对主程序区域的写操作](#)来了解主程序区域解除写保护机制的更多细节。

4.9.7 DATA EEPROM解保护寄存器(FLASH_DUKR)

地址偏移值: 0x0A

复位值: 0x00

7	6	5	4	3	2	1	0
MASS_DATA KEYS							
rw	rw	rw	rw	rw	rw	rw	rw
位7:0	DUK[7:0]: DATA EEPROM解锁密钥 该位可由软件来进行写操作(在任何模式下)。当读该寄存器时, 返回值为00h。请参考 对DATA区域的写操作 来了解数据区域解除写保护机制的更多细节。						

4.9.8 FLASH状态寄存器(FLASH_IAPSR)

地址偏移值: 0x05

复位值: 0x00

7	6	5	4	3	2	1	0
		Reserved		DUL	EOP	PUL	WR_PG_DIS
				rc	rc	rc	rc

位7	保留位, 由硬件保证强迫其为0
位6	HVOFF: 高压结束标志 0: HV 开, 开始真正的编程 1: HV 关, 高压结束
位5:4	保留位, 由硬件保证强迫其为0
位3	DUL: DATA EEPROM区域解锁标志 该位由硬件置位, 可由软件向其写入0来清零。 0: DATA EEPROM区域写保护使能 1: DATA EEPROM区域写保护可通过使用MASS密钥来解除
位2	EOP: 编程结束(写或擦除操作)标志 0: 没有EOP事件发生 1: 有EOP事件发生。如果FLASH_CR1中的IE为1, 将有中断产生
位1	PUL: 快速程序存储器结束标志 该位由硬件置位, 可由软件向其写入0来清零 0: 主程序存储器区域写保护使能 1: 主程序存储器区域写保护可通过使用MASS密钥来解除
位0	WR_PG_DIS: 试图向被保护页进行写操作的标志 该位由硬件置位, 可由软件通过读该寄存器来清零 0: 没有WR_PG_DIS事件发生。 1: 试图向被保护页进行写操作事件发生。如果FLASH_CR1中的IE为1, 将有中断产生

4.9.9 FLASH寄存器映射和复位值

要了解更多寄存器边界地址, 请参考数据手册中通用硬件寄存器映射部分。

表6 FLASH寄存器映射及复位值

地址偏移值	寄存器名	7	6	5	4	3	2	1	0
0x00	FLASH_CR1 复位值	— 0	— 0	— 0	— 0	HALT 0	AHALT 0	IE 0	FIX 0
0x01	FLASH_CR2 复位值	OPT 0	WPRG 0	ERASE 0	FPRG 0	— 0	— 0	— 0	PRG 0
0x02	FLASH_NCR2 复位值	NOPT 1	NWPRG 1	NERASE 1	NFPRG 1	— 1	— 1	— 1	NPRG 1
0x03	FLASH_FPR 复位值	— 0	— 0	WPB5 0	WPB4 0	WPB3 0	WPB2 0	WPB1 0	WPB0 0
0x04	FLASH_NFPR 复位值	— 1	— 1	NWPB5 1	NWPB4 1	NWPB3 1	NWPB2 1	NWPB1 1	NWPB0 1
0x05	FLASH_IAPSR 复位值	— 0	HVOFF 1	—	—	DUL 0	EOP 0	PUL 0	WR_PG_DIS 0
0x06-0x07	保留区								
0x08	FLASH_PUKR 复位值	PUK7 0	PUK6 0	PUK5 0	PUK4 0	PUK3 0	PUK2 0	PUK1 0	PUK0 0
0x09	保留区								
00 5064h	FLASH_DUKR 复位值	DUK7 0	DUK6 0	DUK5 0	DUK4 0	DUK3 0	DUK2 0	DUK1 0	DUK0 0

5 单线接口模块(SWIM)和调试模块(DM)

5.1 介绍

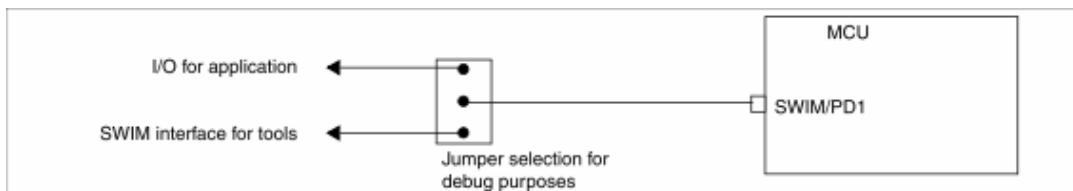
在线调试模式或在应用编程模式由一个单线硬件接口来管理，该接口拥有对存储器超高速编程的特性。该接口和在线调试模块相配合，可提供一种非侵入性(non-intrusive)的仿真模式，在这种仿真模式下，在线调试器的调试仿真功能非常强大，其性能已经接近于一个全功能仿真器。

5.2 主要特性

- 基于一个异步，高注入电流(8mA)，漏极开路的双向通讯。
- 允许读写存储器空间的任意位置。
- 可读写CPU寄存器(A, X, Y, CC, CP)。它们有用于读写的寄存器映射地址。
- 在运行中可对RAM和外设寄存器进行非侵入式读写。
- 器件复位有相应的复位状态指示位，请参考复位状态寄存器 (RST_SR)。
- 时钟速度可选，请参考SWIM 时钟控制寄存器 (CLK_SWIMCCR)。

SWIM引脚可用作普通I/O口，但如果用户还想使用该引脚做调试，则在使用上有一些限制。最安全的作法是在PCB板上提供一个跳线选择。

图9 SWIM引脚连接



5.3 SWIM模式

在上电复位后，SWIM模块复位，然后进入OFF模式。

1. OFF: 在上电复位后的默认状态。SWIM不能用作普通I/O口。
2. I/O: 将全局配置寄存器 (CFG_GCR) 中的SWD位置位后进入该模式。在这种模式下，SWIM引脚可用作普通I/O口。一旦系统复位，SWIM模块重新回到OFF模式。
3. SWIM: 当在SWIM引脚上输入特定的序列时进入该模式。在这种模式下，调试工具通过SWIM引脚使用三种命令(SRST 系统复位，ROTF运行中读，WOTF运行中写)来控制STM8。

注意：可参考STM8 SWIM通讯协议和调试模块用户手册来了解 SWIM和调试模块(DM)的更多信息。

6 供电电源

STM8 MCU有四种相对独立的供电电源:

- V_{DD}/V_{SS} : 主电源(3V到5.5V)
- V_{DDIO}/V_{SSIO} : I/O口供电电源(3V到5.5V)
- V_{DDA}/V_{SSA} : 模拟部分供电电源
- V_{REF+}/V_{REF-} : ADC参考电源

V_{DD}/V_{SS} 引脚用于给内部主电压调节器(MVR)和内部低功耗电压(LPVR)调节器供电。这两个调节器的输出连接在一起, 向MCU的核(CPU, FLASH和RAM)提供1.8V电源(V_{18})。

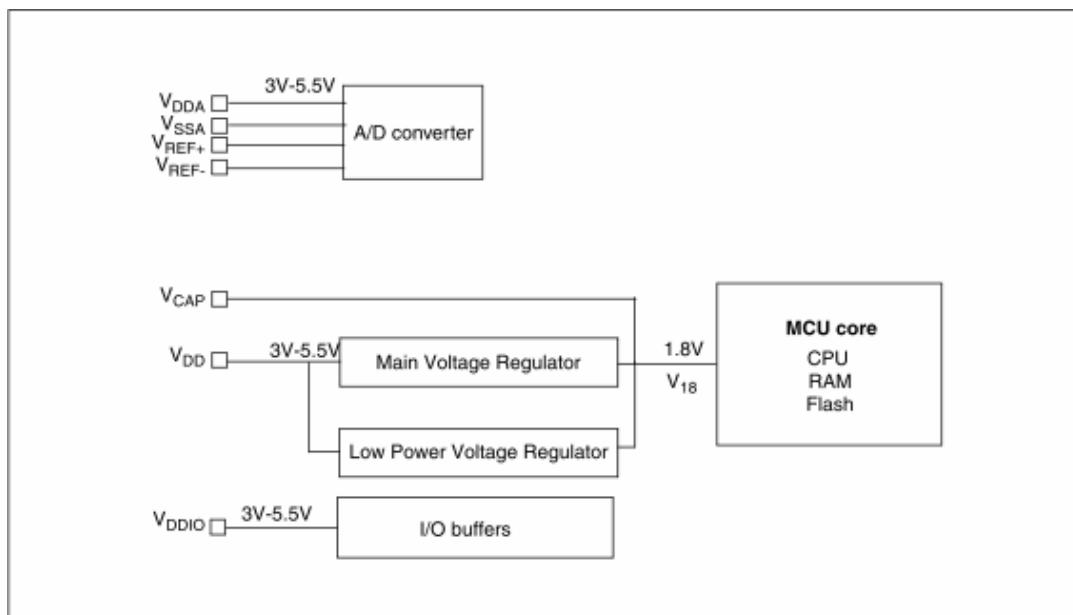
在低功耗模式下, 系统会将供电电源从MVR自动切换到LPVR以减少电流消耗。

为稳定MVR, 在 V_{CAP} 引脚必须连接一个电容。该电容应该拥有较低的等效串联电阻值(ESR), 电容最小的推荐容值为470nF。

根据封装的大小, 可能有一对或两对特定的 V_{DDIO}/V_{SSIO} 来给I/O供电。

V_{DDA}/V_{SSA} 和 V_{REF+}/V_{REF-} 都和ADC模块相连接。

图10 供电电源示意图



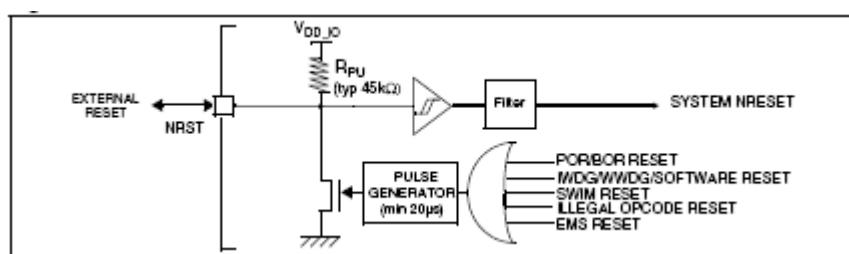
7 复位(RST)

STM8S共有9个复位源:

- NRST引脚产生的外部复位
- 上电复位(POR)
- 掉电复位(BOR)
- 独立看门狗复位
- 窗口看门狗复位
- 软件复位
- SWIM复位
- 非法操作码复位
- EMS复位: 当一些关键的寄存器被破坏或错误加载时产生的复位

所有的复位源最终都作用于NRST管脚，并在复位过程中保持低电平。复位入口向量在内存映射中位于固定的地址6000h。

图11 复位电路



7.1 复位电路

复位引脚NRST内部集成了弱上拉电阻R_{PU}，即可作为输入，也可作为开漏输出。

一个在复位引脚上宽度最小为500ns的低电平脉冲即可产生一个外部复位。对于复位的检测是异步进行的，因此即使MCU处于停机(Halt)模式，也有可能进入复位状态。

复位引脚也可以作为开漏输出用于对外部设备进行复位。

无论内部复位源是什么，一旦复位，内部复位电路都会产生一个至少脉宽为20μs的复位脉冲。当没有外部复位发生时，内部弱上拉电阻可保证复位引脚处于高电平。

请参考图11和见数据手册中的电特性参数章节来了解更多细节。

7.2 内部复位源

除了上电复位(POR)和掉电复位(BOR)，每个内部复位源在复位状态寄存器中都有一个标志位与之相对应。复位时，根据导致复位的复位源，这些标志位被分别设置。因此，这些标志位可用于指示引起最后一次复位的复位源。通过软件写‘1’可清除标志位。

7.2.1 上电复位(POR)和掉电复位(BOR)

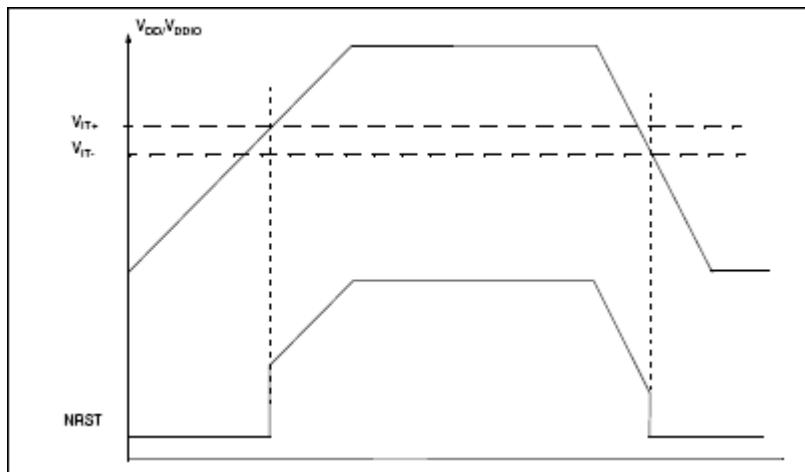
在上电期间，POR保持设备处于复位状态，直到供电电压(V_{DD}和V_{DDIO})到达BOR的启动电压。此时，BOR复位取代POR，POR自动关闭。BOR复位一直持续到供电电压到达工作电压。

详情请参见数据手册的电特性章节。

当工作电压降到门限值V_{TR}以下时，BOR也将产生一个复位，此后POR模块将重新准备好以响应下一次上电复位。

电压迟滞用以保证清楚地检测电压的上升和下降。

即使是MCU处于低功耗模式，BOR也总是保持激活状态。

图12 V_{DD}/V_{DDIO} 电压检测: POR/BOR门限

7.2.2 看门狗复位

详情请参见[14独立看门狗\(IWDG\)](#)和[15窗口看门狗\(WWDG\)](#)。

7.2.3 软件复位

应用程序可通过清除寄存器WWDG_CR中的T6位来触发一个复位，详情请参见[15窗口看门狗\(WWDG\)](#)。

7.2.4 SWIM复位

连接到SWIM接口的外部设备可通过SWIM模块产生一个MCU复位。

7.2.5 非法操作码复位

为了提高设备的可靠性，防止意外行为的发生，使用了非法操作码检测系统。如果一个被执行的代码与任意操作码或预置字节均不相符，则产生一个复位。此功能与看门狗相配合，可使设备从一个意外错误或干扰中恢复。

注意：一个有效的预置字节与一个有效的操作码组成的一个非法的组合将不会产生复位。

7.2.6 EMS复位

为了避免由电磁干扰造成的对应用程序误写操作或系统挂起，大多数关键寄存器都有一个互补寄存器与之相对应。系统将会自动检测这些关键寄存器与其互补寄存器之间是否匹配。如果不匹配，则产生一个EMS复位，从而使应用程序恢复到正常操作。

7.3 复位(RST)寄存器

7.3.1 复位状态寄存器(RST_SR)

地址偏移值: 0x00

复位值: 未定义

7	6	5	4	3	2	1	0
保留		EMCF	SWIMF	ILLOPF	IWDGF	WWDGF	
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	

位7:5	保留, 必须保持为0。
位4	EMCF: EMC复位标志 由硬件置位, 可通过软件写"1"清除 0: 无EMC复位发生; 1: 有一个EMC复位发生(可能的复位原因: 互补寄存器或选项字节不匹配)。
位3	SWIMF: SWIM复位标志位 由硬件置位, 可通过软件写"1"清除 0: 无SWIM复位发生; 1: 有一个SWIM复位发生。
位2	ILLOPF: 非法操作码复位标志位 由硬件置位, 可通过软件写"1"清除 0: 无非法操作码复位发生; 1: 有一个非法操作码复位发生。
位1	IWDGF: 独立型看门狗复位标志位 由硬件置位, 可通过软件写"1"清除 0: 无IWDG复位发生; 1: 有一个IWDG复位发生。
位0	WWDGF: 窗口型看门狗复位标志位 由硬件置位, 可通过软件写"1"清除 0: 无WWDG复位发生; 1: 有一个WWDG复位发生。

7.4 复位寄存器地址映射

请参考对应的数据手册了解基址地址信息。

表7 复位寄存器地址映射

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	RST_SR 复位值	- x	- x	- x	EMCF x	SWIMF x	ILLOPF x	IWDGF x	WWDGF x

时钟控制器功能强大而且灵活易用。其目的在于使用户在获得最好性能的同时，亦能保证消耗的功率最低。

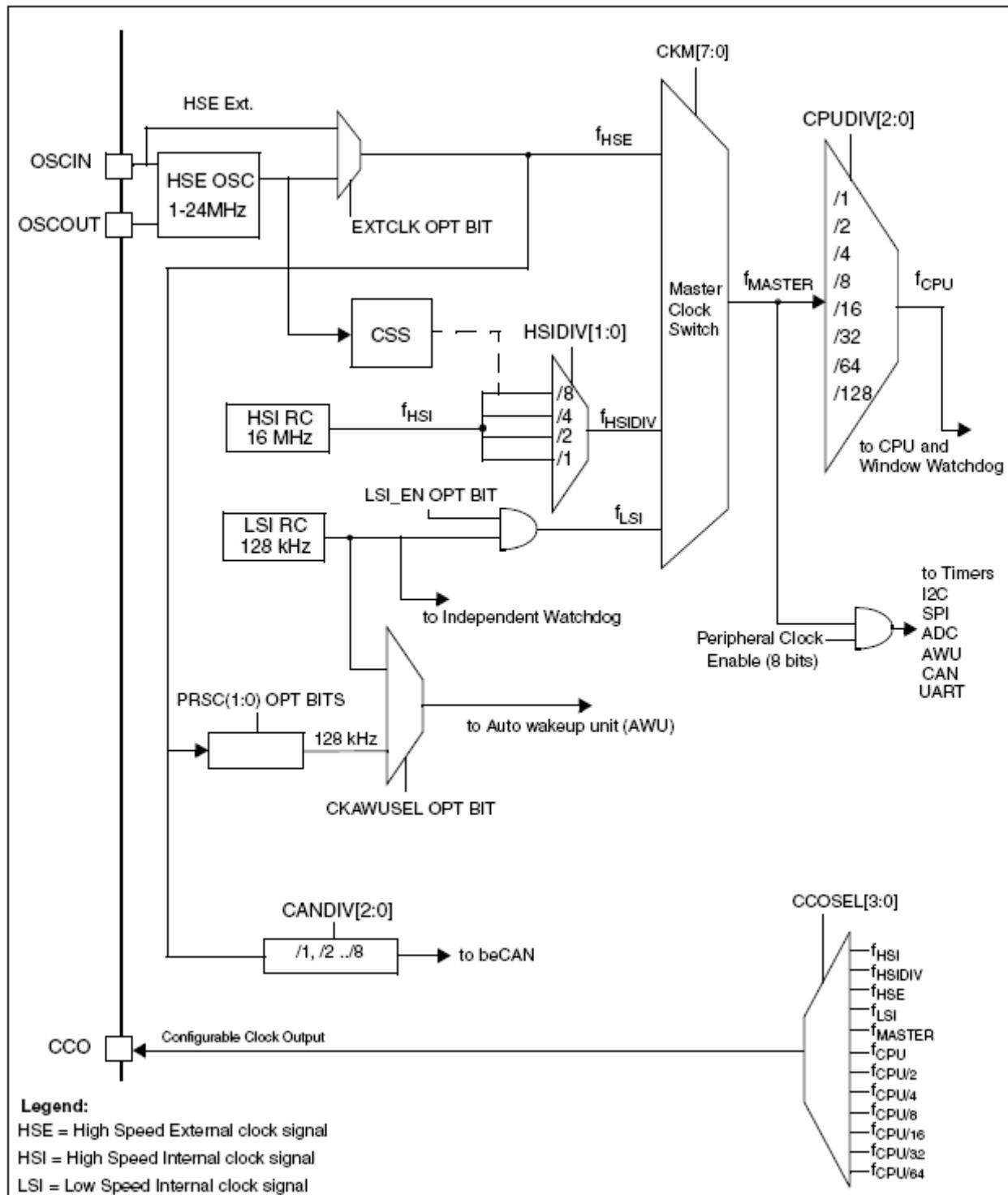
用户可独立地管理各个时钟源，并将它们分配到**CPU**或各个外设。主时钟和**CPU**时钟均带有预分频器。

具有安全可靠的无故障时钟切换机制，可在程序运行中将主时钟从一个时钟源切换到另一个时钟源。

抗电磁干扰时钟配置寄存器

为了避免由电磁干扰造成的对应用程序误写操作或系统挂起，大多数关键的时钟配置寄存器都有一个互补寄存器与之相对应。系统将会自动检测这些关键寄存器与其互补寄存器之间是否匹配。如果不匹配，则产生一个**EMS**复位，从而使应用程序恢复到正常操作。详情请参见时钟寄存器描述。

图13 时钟树



8.1 主时钟源

下面4种时钟源可用做主时钟：

- 1-24MHz高速外部晶体振荡器(HSE)
- 最大24MHz高速外部时钟信号(HSE user-ext)
- 16MHz高速内部RC振荡器(HSI)
- 128KHz低速内部RC(LSI)

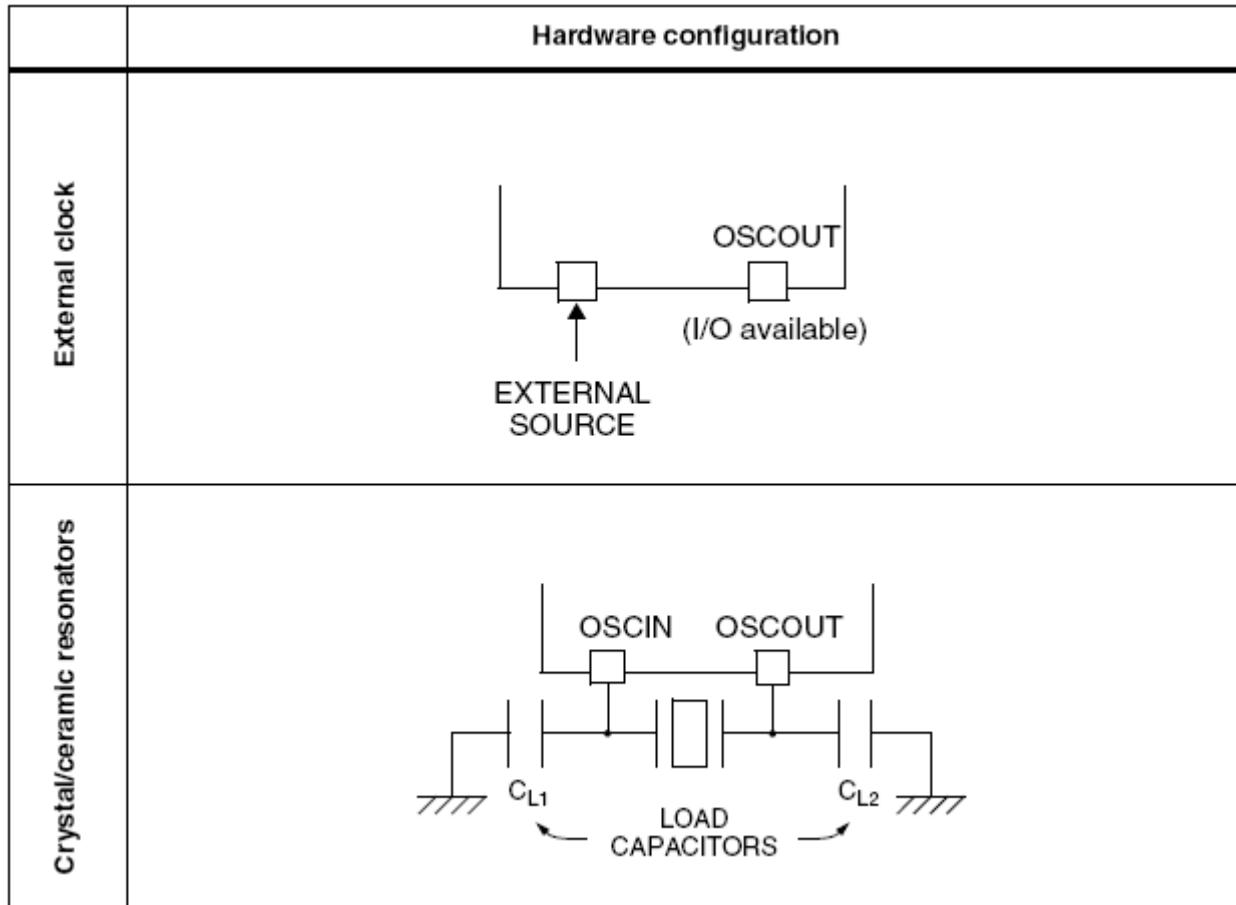
各个时钟源可单独打开或关闭，从而优化功耗。

8.1.1 HSE

高速外部时钟信号可由下面两个时钟源产生：

- HSE外部晶体/陶瓷谐振器
- HSE用户外部有源时钟

图14 HSE时钟源



为了最大限度减小输出失真和减小启动的稳定时间，谐振器和负载电容应尽可能得靠近振荡器引脚。负载电容值应根据所选的振荡器进行调整。

外部晶体/陶瓷谐振器(HSE晶体)

外部1至24MHz的振荡器其优点在于能够产生精确的占空比为50%的主时钟信号。

硬件连接如图14所示。更多详情请参见数据手册电特性章节。

振荡器在启动时的输出时钟信号是不稳定的，默认情况下，在时钟信号被使用之前会插入2048个振荡器周期的延迟。用户可通过设置选项字节HSECNT来缩短稳定时间，请参见数据手册的选项字节章节。

外部时钟寄存器CLK_ECKR中的标志位HSERDY用以指示高速外部振荡器是否稳定。启动时，HSE时钟信号将不会生效直至此标志位被硬件置位。

HSE晶体可通过设置外部时钟寄存器CLK_ECKR中的HSEEN位来打开或关闭。

外部时钟源(HSE用户外部时钟)

这种模式下，必须由用户提供一个外部时钟，此时钟的最高频率可为24MHz。用户可通过编程选项位EXTCLK选择此模式。详情请参见数据手册的选项字节章节。此时，占空比约50%的外部时钟信号(方波，正弦波，三角波)用以驱动OSCIN引脚，而OSCOUT引脚可做为通用输入/输出管脚使用。请参见图13。

8.1.2 HSI

HSI信号由内部16MHz RC振荡器与一个可编程分频器(分频因子从1至8)产生。分频因子由寄存器CLK_CKDIVR决定。

注意: 启动时, 主时钟源默认为HSI RC时钟的8分频, 即 $f_{HSI}/8$

HSI RC可以提供一个低成本的16MHz时钟源(无需外部器件), 其占空比为50%。HSI启动速度比HSE晶体振荡器快, 但是其精度即使经过校准也仍然比外部晶体振荡器或陶瓷谐振器低。

内部时钟寄存器CLK_ICKR中的标志位HSIRDY用以指示HSI RC是否稳定。启动时, HSI时钟信号将不会生效直至此标志位被硬件置位。

HSI RC可通过设置内部时钟寄存器CLK_ICKR中的HSIEN位打开或关闭。

备份时钟源

当HSE晶体振荡器失效时, HSI/8可作为备份时钟源(辅助时钟源)使用。请参见[8.6时钟安全系统\(CSS\)](#)。

快速启动特性

如果寄存器CLK_ICKR中的FHWU位被置1, 则MCU从停机(Halt)模式或活跃停机(Active Halt)模式唤醒时, HSI将自动被设为主时钟源。

校准

每个产品在出厂时均已经ST校准。

复位后, 出厂校准值将被自动加载至内部校准寄存器。

如果实际应用中电压或温度偏差较大, 将会影响RC振荡器的速度。用户可使用HSI时钟校准寄存器(CLK_HSI TRIMR)修正HSI的时钟频率。此寄存器中有3或4位用以存放一个附加的修正值, 并与内部HSI校准寄存器的值相加来对时钟进行校正。

8.1.3 LSI

128KHz的LSI RC时钟是一个低功耗, 低成本的可选主时钟源, 也可在停机(Halt)模式下作为维持独立看门狗和自动唤醒单元(AWU)运行的低功耗时钟源。

LSI可通过设置内部时钟寄存器CLK_ICKR中的LSIEN位打开或关闭。

内部时钟寄存器CLK_ICKR中的标志位LSIRDY用以指示LSI是否稳定。启动时, LSI时钟信号将不会生效直至此标志位被硬件置位。

校准

同HSI一样, LSI出厂时已经校准。但是, 不可能再执行进一步的校准。

注意: 当独立看门狗使用LSI为时钟源时, 为了保证CPU在系统出错时不与独立看门狗使用同一个时钟, 当选项字节位LSI_EN为0时, LSI不能做为主时钟。请参见数据手册中的选项字节章节。

8.2 主时钟切换

时钟切换功能为用户提供了一种易用、快速、安全的从一个时钟源切换到另一个时钟源的途径。

8.2.1 系统启动

为使系统快速启动, 复位后时钟控制器自动使用HSI的8分频(HSI/8)做为主时钟。其原因为HSI的稳定时间短, 而8分频可保证系统在较差的V_{DD}条件下安全启动。

一旦主时钟源稳定, 用户程序可将主时钟切换到另外的时钟源。

8.2.2 主时钟切换的过程

用户可选择下面两种方式切换时钟源:

- 自动切换
- 手动切换

自动切换

自动切换使用户可使用最少的指令完成时钟源的切换。应用软件可继续其它操作而不用考虑切换事件所占的确切时间。

如[图15](#)所示。

1. 设置切换控制寄存器(CLK_SWCR)中的位SWEN，使能切换机制。
2. 向主时钟切换寄存器(CLK_SWR)写入一个8位的值，用以选择目标时钟源。寄存器CLK_SWCR中的SWBSY被硬件置位，目标源振荡器启动。原时钟源依然被用于驱动内核和外设。

一旦目标时钟源稳定，寄存器CLK_SWR中的值将被复制到主时钟状态寄存器(CLK_CMSR)中去。

此时，SWBSY位被清除，新时钟源替代旧时钟源。寄存器CLK_SWCR中的标志位SWIF被置位，如果SWIEN为1，则会产生一个中断。

手动切换

手动切换与自动切换不同，不能够立即切换，但它允许用户精确地控制切换事件发生的时间，如[图16](#)所示。

1. 向主时钟切换寄存器(CLK_SWR)写入一个8位的值，用以选择目标时钟源。寄存器CLK_SWCR中的SWBSY被硬件置位，目标源振荡器启动。原时钟源依然被用于驱动内核和外设。
2. 用户软件需等待至目标时钟源稳定。寄存器CLK_SWCR中的标志位SWIF用以指示目标时钟源是否已稳定，如果SWIEN为1，则会产生一个中断。
3. 最后，由用户软件在所选的时间点，设置寄存器CLK_SWCR中的位SWEN，执行切换。

无论是手动切换还是自动切换，如果原时钟源仍然在被其他模块使用(如LSI在被独立看门狗使用)，则原时钟源将不会被自动关闭。配置内部时钟寄存器CLK_ICKR和外部时钟寄存器CLK_ECKR中的相应位，可关闭原时钟源。

如果由于某种原因时钟切换没有成功，软件可通过清除标志位SWBSY以复位当前的切换操作，使寄存器CLK_SWR恢复原值(原时钟源)。

图15 时钟切换流程图(自动切换)

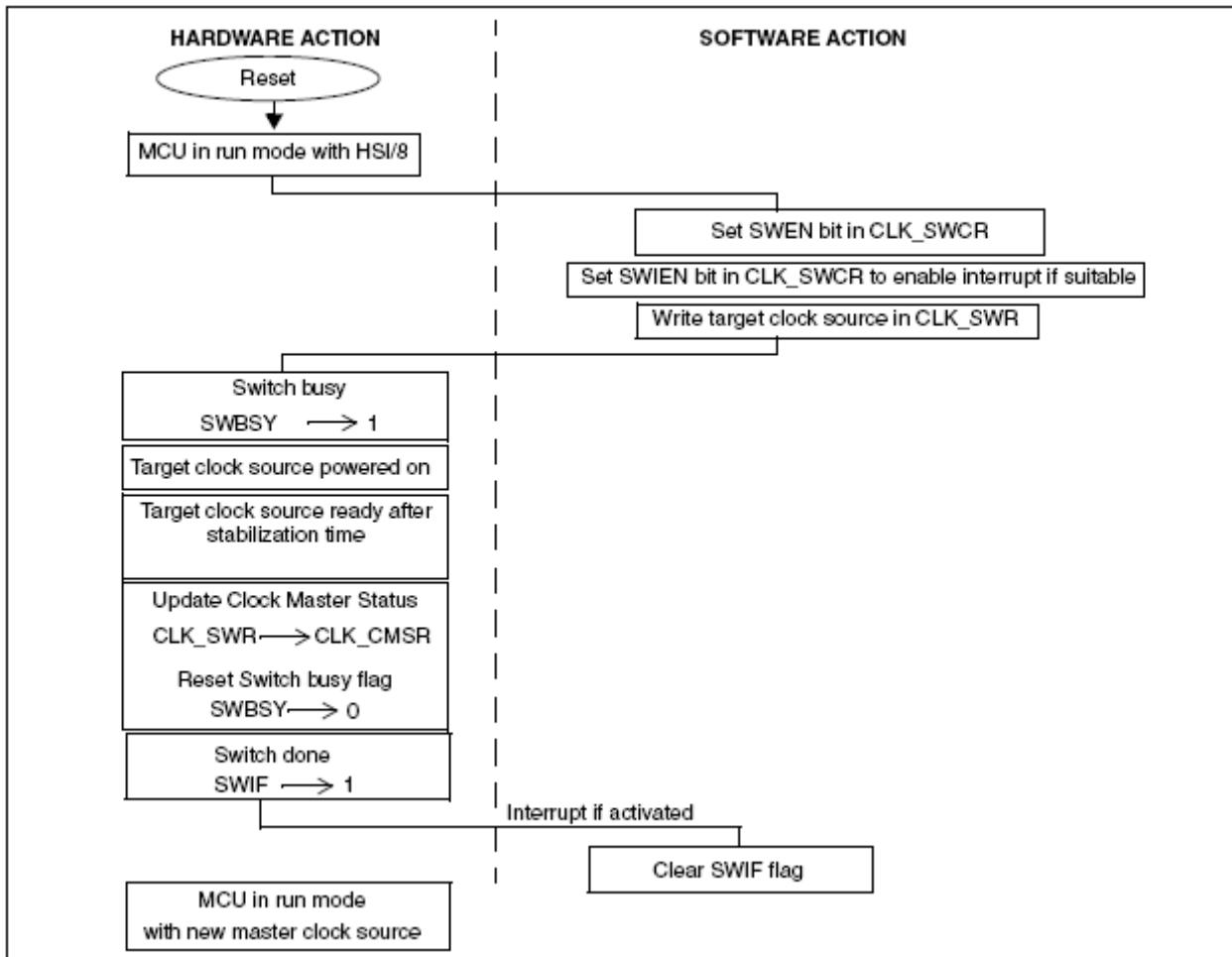
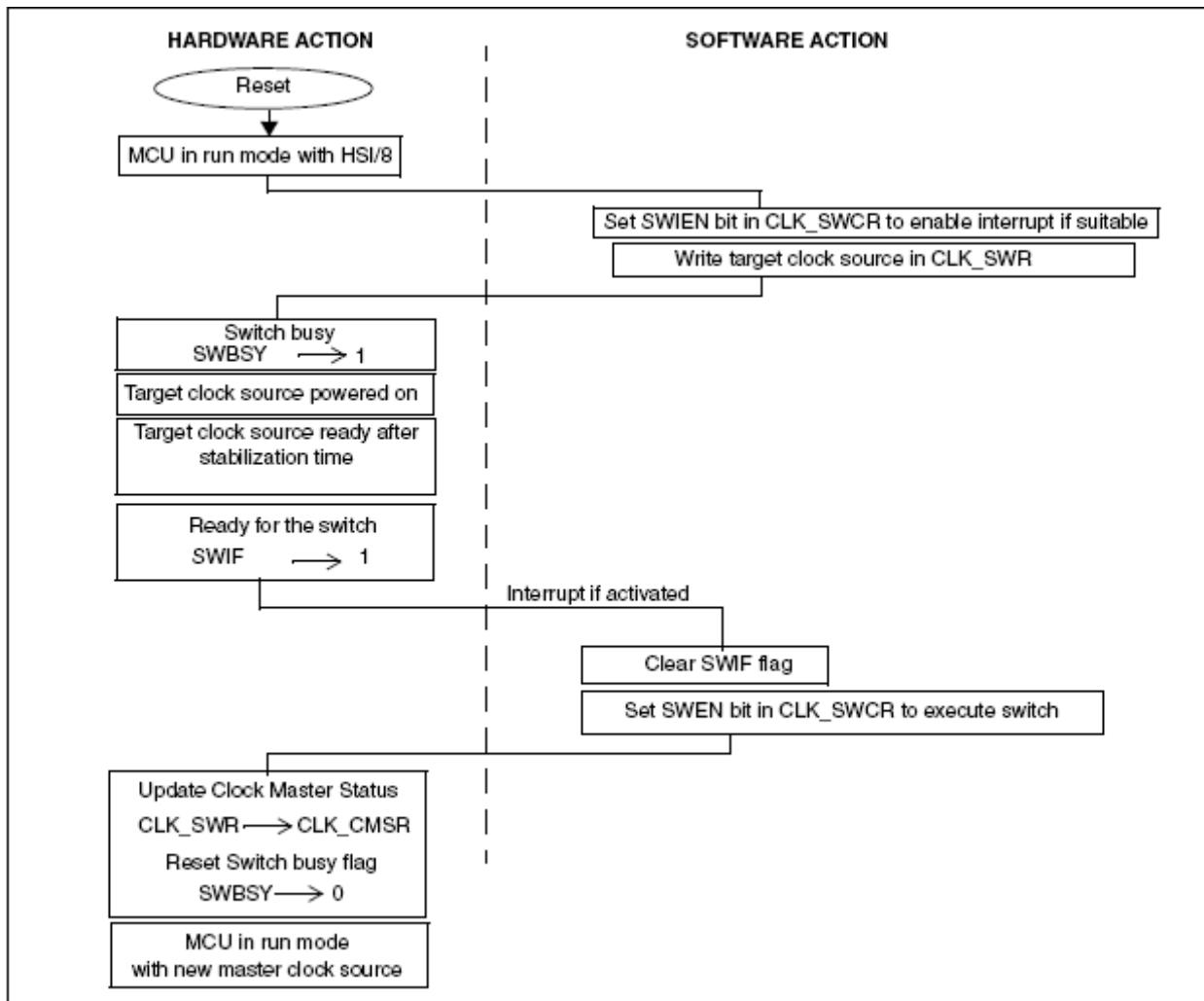


图16 时钟切换流程图(手动切换)



8.3 低速时钟源的选择

可通过选择位CKAWUSEL来选择自动唤醒单元(AWU)和独立看门狗所使用的低速时钟源，可以是LSI或HSE的分频。请参见数据手册选项字节章节。

HSE的分频值可通过选项字节位HSEPRSC[1:0]编程。请参见数据手册选项字节章节。目的是使HSE分频器输出一个128KHz的时钟信号。

8.4 CPU时钟分频器

CPU时钟(f_{CPU})由主时钟(f_{MASTER})分频而来，分频因子由时钟分频寄存器(CLK_CKDIVR)中的位CPUDIV[2:0]决定。共7个分频因子可供选择(1至128中，2的幂)。如图13所示。

f_{CPU} 为CPU和窗口看门狗提供时钟。

8.5 外设时钟门控

关闭未使用外设的时钟可降低功耗。外设的时钟门控(PCG)模式使用户可在运行模式下随时打开或关闭 f_{MASTER} 与下列外设的连接：

- ADC
- I2C
- AWU(寄存器时钟，而非计数器时钟)
- SPI
- TIM[4:1]

- UART
- CAN(寄存器时钟, 而非CAN时钟)

系统复位后, 所有外设时钟均处于开的状态。用户可通过清除 CLK_PCKENR1 或 CLK_PCKENR2 中的 PCKEN 位来关闭相应的外设时钟。但是在关闭外设的时钟前, 用户必须设置相应的位禁用该外设。

为了使能一个外设, 用户必须先设置寄存器 CLK_PCKENR 中对应的 PCKEN 位, 然后设置外设控制寄存器中的外设使能位。

AWU计数器是由独立于 f_{MASTER} 的内部或外部时钟(LSI或HSE)驱动, 因此, 即使寄存器的时钟已被关掉, 该外设依然可以继续运行。

8.6 时钟安全系统(CSS)

时钟安全系统用于监控 HSE 时钟源是否失效。当 f_{MASTER} 使用 HSE 做为时钟源时, 如果 HSE 时钟由于谐振器损坏、断开或其它原因而失效, 时钟控制器将激活安全恢复机制, 将 f_{MASTER} 自动切换到辅助时钟源 HSI/8。系统将一直使用辅助时钟源, 直至 MCU 被复位。

设置时钟安全系统寄存器 CLK_CSSR 中的 CSSEN 位, 可使能时钟安全系统。为安全起见, CSS 一旦使能就不能被关闭, 直到下一次复位。

必须满足下面的条件, CSS 方可检测 HSE 石英晶体的失效:

- HSE 晶体开: (外部时钟寄存器 CLK_ECKR 中的位 HSEEN=1)
- HSE 振荡器被置为石英晶体(选项位 EXTCLK 为 1)
- CSS 功能开: (寄存器 CLK_CSSR 中 CSSEN=1)

如果当前的主时钟源为 HSE, 当失效被检测到时, CSS 将执行以下操作:

- 寄存器 CLK_CSSR 中的 CSSD 位被置位, 如果 CSSIEN 为 1, 则同时产生一个中断。
- CLK_CMSR, CLK_SWR, 及 CLK_CKDIVR 中的 HSIDIV[1:0] 位被置为复位值 (CKM[7:0]=SWI[7:0]=E1h)。HSI/8 成为主时钟。
- 内部时钟寄存器 CLK_ICKR 中的 HSIEN 被置位(HSI开)。
- 外部时钟寄存器 CLK_ECKR 中的 HSEEN 被清除(HSE关)。
- AXU 位被置位, 用以指示辅助时钟源 HSI/8 被强制使用。

用户可通过软件清除 CSSD 位, 但 AXU 位只能由复位清除。

为了提高时钟频率, 用户在清除寄存器 CLK_CSSR 中的 CSSD 位以后, 可修改寄存器 CLK_CKDIVR 中的 HSIDIV[1:0] 位。

如果失效发生时 HSE 不是主时钟源, 主时钟将不会被切换到辅助时钟源, 以上操作也不会发生, 仅执行下面的操作:

- 外部时钟寄存器 CLK_ECKR 中的 HSEEN 被清除, HSE 关闭。
- 寄存器 CLK_CSSR 中的位 CSSD 被置位, 如果 CSSIEN 为 1, 则同时产生一个中断。

如果 HSE 不是当前主时钟源, 且主时钟正在被切换至 HSE, 则在清除 CSSD 位之前, 必须先清除寄存器 CLK_SWCR 的 SWBSY 位。

如果当失效被检测到时, HSE 被 CCOSEL 选择为时钟输出模式(参见 [时钟输出功能\(CCO\)](#)), 则 HSI(HSIDIV) 将替代 HSE, 被自动强制选择为输出时钟。

8.7 时钟输出功能(CCO)

可配置的时钟输出功能使用户可在外部管脚 CCO 上输出指定的时钟。用户可选择下面 6 种时钟信号之一做为 CCO 时钟:

- f_{HSE}
- f_{HSI}
- f_{HSIDIV}
- f_{LSI}

- f_{MASTER}
- f_{CPU} (可选择分频值)

注意: 在所有可能的分频值下, 不能保证信号的占空比全部为50%

通过配置时钟输出寄存器CLK_CCOR中域CCOSEL[3:0]可选择输出的时钟。

用户需为指定的I/O引脚(参见管脚描述章节)选择期望输出的时钟。此I/O必须通过配置寄存器Px_CR1对应的位为1来设置为上拉输入或推挽输出模式。

一旦可配置时钟输出寄存器CLK_CCOR的位CCOEN=1, 就开始输出所选定的时钟信号。

如果CCOBSY为1, 则表明可配置时钟输出系统正在工作。只要CCOBSY为1, CCOSEL位就会被写保护。

如果需要, CCO可自动激活目标振荡器。当所选时钟就绪时, CCORDY被置位。

用户可通过清除CCOEN位来禁用时钟输出功能。CCOBSY位和CCORDY位都将保持为1直到禁用操作结束。从清除CCOEN位到这两个标志位被复位之间的时间可能会很长, 例如当所选的输出时钟相对于 f_{CPU} 频率很低时。

8.8 时钟中断

当下列事件发生时, 时钟控制器可产生中断:

- 主时钟源切换事件
- CSS事件

这两个中断均可被独立屏蔽。

表8 时钟中断请求

中断事件	事件标志位	使能控制位	从Wait模式退出	从Halt模式退出
CSS事件	CSSD	CSSDIE	是	否
主时钟切换事件	SWIF	SWIEN	是	否

8.9 时钟寄存器

8.9.1 内部时钟寄存器(CLK_ICCR)

地址偏移值: 0x00

复位值: 0x01

7	6	5	4	3	2	1	0
保留	REGAH	LSIRDY	LSIEN	FHW	HSIRDY	HSIEN	

r w r r w r w r r w

位 7:6	保留。始终为0。
位 5	REGAH: 活跃停机(Active Halt)模式下电压调节器关闭 由软件置位或清除。为1时, 一旦MCU进入活跃停机(Active Halt)模式, 主电压调节器将关闭, 从而唤醒时间将比较长。 0: 活跃停机(Active Halt)模式下主电压调节器处于开 1: 活跃停机(Active Halt)模式下主电压调节器处于关
位 4	LSIRDY: 低速内部振荡器准备就绪 由硬件置位或清除 0: LSI时钟未准备就绪 1: LSI时钟准备就绪
位 3	LSIEN: 低速内部振荡器使能 由软件置位或清除。如果LSI为必需的, 则硬件将该位置1, 例如: - 当时钟源切换至LSI时(参见寄存器CLK_SWR) - 当LSI被指定为时钟输出源(CCO)时(参见寄存器CLK_CCOR) - 当BEEP被使能时(寄存器BEEP_CSR的位BEEPEN=1) - 当LSI测量被使能时(寄存器AWU_CSR的位MSR=1) 当LSI被指定为主时钟源/CCO时钟源/AWU/IWDG的时钟源时, 该位不能被清除。 0: 关闭低速内部振荡器 1: 打开低速内部振荡器
位 2	FHWU: 从停机(Halt)或活跃停机(Active Halt)模式快速唤醒 由软件置位或清除 0: 从停机(Halt)或活跃停机(Active Halt)模式快速唤醒禁用 1: 从停机(Halt)或活跃停机(Active Halt)模式快速唤醒使能
位 1	HSIRDY: 高速内部振荡器准备就绪 由硬件置位或清除 0: HSI未准备就绪 1: HSI准备就绪
位 0	HSIEN: 高速内部RC振荡器使能 由软件置位或清除。如果HSI为必需的, 则硬件将该位置1, 例如: - 当被CSS激活, 做为安全备用振荡器 - 当时钟源切换至HSI(参见寄存器CLK_SWR) - 当HSI被指定为时钟输出源(CCO)时(参见寄存器CLK_CCOR) 当HSI被指定为主时钟源, 或CCO时钟源, 或安全备份(辅助)时钟源时, 该位不能被清除。 0: 高速内部RC关。 1: 高速内部RC开。

8.9.2 外部时钟寄存器(CLK_ECKR)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
保留						HSERDY	HSEEN
r							rw

位 7:2	保留。始终为0。
位 1	HSERDY: 高速外部晶体振荡器准备就绪 由硬件置位或清除。 0: HSE未准备就绪 1: HSE准备就绪
位 0	HSEEN: 高速外部晶体振荡器使能 由软件置位或清除。用于打开或关闭外部晶体振荡器。下列情况下, 由硬件将该位置1: -当时钟源切换至HSE(参见寄存器CLK_SWR) -当HSE被指定为时钟输出源(CCO)时(参见寄存器CLK_CCOR) 当HSE被指定为主时钟源, 或CCO时钟源时, 该位不能被清除。 0: HSE关 1: HSE开

8.9.3 主时钟状态寄存器(CLK_CMSR)

地址偏移值: 0x03

复位值: 0xE1

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

CKM[7:0]

r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---

位 7:0	CKM[7:0]: 主时钟状态位
-------	-------------------------

由硬件置位或清除。用以指示当前所选的主时钟源。如果该寄存器中的值为无效值，则产生MCU复位

0xE1: HSI为主时钟源(复位值)

0xD2: LSI为主时钟源(仅当LSI_EN选项位为1时)

0xB4: HSE为主时钟源

8.9.4 主时钟切换寄存器(CLK_SWR)

地址偏移值: 0x04

复位值: 0xE1

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

SWI[7:0]

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

位 7:0

SWI[7:0]: 主时钟选择位

由软件写入。用以选择主时钟源。当时钟切换正在进行(SWBSY=1)时, 该寄存器的内容将被写保护。如果寄存器CLK_CSSR的位AUX=1, 则该寄存器将被置位复位值(HSI)。如果选择了快速Halt唤醒模式(寄存器CLK_ICKR的位FHW=1), 从停机(Halt)/ 活跃停机(Active Halt)唤醒时, 该寄存器将被硬件设置为E1h(选择HSI)

0xE1: HSI为主时钟源(复位值)

0xD2: LSI为主时钟源(仅当LSI_EN选项位为1时)

0xB4: HSE为主时钟源

8.9.5 切换控制寄存器(CLK_SWCR)

地址偏移值: 0x05

复位值: 未定义

7	6	5	4	3	2	1	0
		保留		SWIF	SWIEN	SWEN	SWBSY
				rc_w0	rw	rw	rw

位 7:4	保留。始终为0。
位 3	<p>SWIF: 时钟切换中断标志位 由硬件置位或软件写0清除。该位的含义取决于SWEN位的状态。参见图15和图16。</p> <p>手动切换模式下(SWEN=0):</p> <p>0: 目标时钟源未准备就绪 1: 目标时钟源准备就绪</p> <p>自动切换模式下(SWEN=1):</p> <p>0: 无时钟切换事件发生 1: 有时钟切换事件发生</p>
位 2	<p>SWIEN: 时钟切换中断使能 由软件置位或清除</p> <p>0: 时钟切换中断禁用 1: 时钟切换中断使能</p>
位 1	<p>SWEN: 切换启动/停止 由软件置位或清除。向该位写1将切换主时钟至寄存器CLK_SWR指定的时钟源。</p> <p>0: 禁止时钟切换的执行 1: 使能时钟切换的执行</p>
位 0	<p>SWBSY: 切换忙 由硬件置位或清除。可由软件清除以复位时钟切换过程。</p> <p>0: 无时钟切换正在进行。 1: 时钟切换正在进行。</p>

8.9.6 时钟分频寄存器(CLK_CKDIVR)

地址偏移值: 0x06

复位值: 0x18

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

保留	HSIDIV[1:0]	CPUDIV[2:0]
	rw	rw

位 7:5	保留。始终为0。
位 4:3	HSIDIV[1:0]: 高速内部时钟预分频器 由软件写入, 用于指定HSI分频因子。 00: $f_{HSI} = f_{HSI}$ RC输出 01: $f_{HSI} = f_{HSI}$ RC输出/2 10: $f_{HSI} = f_{HSI}$ RC输出/4 11: $f_{HSI} = f_{HSI}$ RC输出/8
位 2:0	CPUDIV[2:0]: CPU时钟预分频器 由软件写入, 用于指定CPU时钟预分频因子。 000: $f_{CPU} = f_{MASTER}$ 001: $f_{CPU} = f_{MASTER}/2$ 010: $f_{CPU} = f_{MASTER}/4$ 011: $f_{CPU} = f_{MASTER}/8$ 100: $f_{CPU} = f_{MASTER}/16$ 101: $f_{CPU} = f_{MASTER}/32$ 110: $f_{CPU} = f_{MASTER}/64$ 111: $f_{CPU} = f_{MASTER}/128$

8.9.7 外设时钟门控寄存器(CLK_PCKENR1)

地址偏移值: 0x07

复位值: 0xFF

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

PCKEN1[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW

位 7:0

PCKEN1[7:0]: 外设时钟使能

由软件写入。使能或禁止 f_{MASTER} 时钟与对应外设的连接。参见 [表9](#)

0: 禁止 f_{MASTER} 与外设连接

1: 使能 f_{MASTER} 与外设的连接

表9 外设时钟门控位

控制位	外设
PCKEN17	TIM1
PCKEN16	TIM3
PCKEN15	TIM2
PCKEN14	TIM4
PCKEN13	UART2/3
PCKEN12	UART1
PCKEN11	SPI
PCKEN10	I2C

8.9.8 外设时钟门控寄存器 2(CLK_PCKENR2)

地址偏移值: 0x0A

复位值: 0xFF

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

PCKEN2[7:0]							
-------------	--	--	--	--	--	--	--

rw							
----	----	----	----	----	----	----	----

位 7:0	PCKEN2[7:0]: 外设时钟使能
-------	----------------------------

由软件写入。使能或禁止 f_{MASTER} 时钟与对应外设的连接。参见[表10](#)

0: 禁止 f_{MASTER} 与外设连接

1: 使能 f_{MASTER} 与外设的连接

表10 外设时钟门控位

控制位	外设
PCKEN27	CAN
PCKEN26	Reserved
PCKEN25	Reserved
PCKEN24	Reserved
PCKEN23	ADC
PCKEN22	AWU
PCKEN21	Reserved
PCKEN20	Reserved

8.9.9 时钟安全系统寄存器(CLK_CSSR)

地址偏移: 0x08

复位值: 0x00

7 6 5 4 3 2 1 0

保留	CSSD	CSSDIE	AUX	CSSEN
	rc_w0	rw	r	rw0

位 7:4	保留。始终为0。
位 3	CSSD: 时钟安全系统监测 由硬件置位或软件写0清除。 0: CSS关或未检测到HSE失效 1: 检测到HSE失效
位 2	CSSDIE: 时钟安全系统监测中断使能 由软件置位或清除 0: 时钟安全系统监测中断禁用 1: 时钟安全系统监测中断使能
位 1	AUX: 辅助振荡器连接至主时钟 由硬件置位或清除。 0: 辅助振荡器关 1: 辅助振荡器(HSI/8)开，并做为当前的主时钟源
位 0	CSSEN: 时钟安全系统使能 可读，但只能由软件写一次。 0: 时钟安全系统关。 1: 时钟安全系统开。

8.9.10 可配置时钟输出寄存器

地址偏移: 0x09

复位值: 0x00

7	6	5	4	3	2	1	0
保留	CCOBSY	CCORDY		CCOSEL[3:0]		CCOEN	

r r rw rw rw rw rw

位 7	保留。始终为0。
位 6	CCOBSY: 可配置时钟输出忙 由硬件置位或清除。用于指示所选的CCO时钟源正处于切换状态或稳定状态。当CCOBSY为1时，CCOSEL位域将被写保护。CCOBSY保持为1直至CCO时钟被使能。 0: CCO时钟空闲 1: CCO时钟忙
位 5	CCORDY: 可配置时钟输出准备就绪 由硬件置位或清除。用于指示CCO时钟的状态 0: CCO时钟可用 1: CCO时钟不可用
位 4:1	CCOSEL[3:0]: 可配置时钟输出源选择 由软件写入。用于选择CLK_CCO管脚上的输出时钟源。当CCOBSY=1时，该位域被写保护。 0000: f_{HSIDIV} 0001: f_{LSI} 0010: f_{HSE} 0011: Reserved 0100: f_{CPU} 0101: $f_{CPU}/2$ 0110: $f_{CPU}/4$ 0111: $f_{CPU}/8$ 1000: $f_{CPU}/16$ 1001: $f_{CPU}/32$ 1010: $f_{CPU}/64$ 1011: f_{HSI} 1100: f_{MASTER} 1101: f_{CPU} 1110: f_{CPU} 1111: f_{CPU}
位 0	CCOEN: 可配置时钟输出使能 由软件置位或清除 0: 禁止CCO时钟输出 1: 使能CCO时钟输出

8.9.11 CAN外部时钟控制寄存器(CLK_CANCCR)

地址偏移值: 0x0B

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

保留	CANDIV[2:0]
	rw rw rw

位 7:3	保留。始终为0。
位 2:0	<p>CANDIV[2:0]: 外部CAN时钟分频值 由软件写入, 用于指定外部CAN时钟的分频值。详情请参见章节23.9</p> <p>000: 外部CAN时钟 = $f_{HSE}/1$(复位值) 001: 外部CAN时钟 = $f_{HSE}/2$ 010: 外部CAN时钟 = $f_{MASTER}/4$... 111: 外部CAN时钟 = $f_{MASTER}/8$</p>

8.9.12 HSI时钟修正寄存器(CLK_HSITRIMR)

地址偏移: 0x0C

复位值: 未定义

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

保留	HSITRIMR[3:0]
	rw rw rw rw

位 7:4	保留。始终为0。
位 3:0	<p>HSITRIM[3:0]: HSI修正值 由软件写入, 用于微调HSI的校准值 注意: 在大容量产品上, 只有位2:0是可用的。 在中等容量或小容量产品上, 位3:0或2:0是可用的, 取决于选项字节的配置(参见数据手册)。</p>

8.9.13 SWIM时钟控制寄存器(CLK_SWIMCCR)

地址偏移值: 0x0D

复位值: 未定义

7

6

5

4

3

2

1

0

保留

SWIMCLK

RW

位 7:1	保留。始终为0。
位 0	<p>SWIMCLK: SWIM时钟分频值 由软件置位或清除。</p> <p>0: SWIM时钟被2分频 1: SWIM时钟未2分频</p>

8.10 时钟寄存器地址映射

表11 时钟寄存器地址映射与复位值

地址偏移	寄存器	7	6	5	4	3	2	1	0	
0x00	CLK_ICCR 复位值	0	0	0	REGAH	LSIRDY	LSIEN	FHWU	HSIRDY	HSIEN
0x01	CLK_ECCR 复位值	x	x	x	x	x	x	x	HSERDY	HSEEN
0x02		保留区域(1个字节)								
0x03	CLK_CMSR 复位值	1	1	1	0	0	0	0	1	
0x04	CLK_SWR 复位值	1	1	1	0	0	0	0	1	
0x05	CLK_SWCR 复位值	x	x	x	x	SWIF	SWIEN	SWEN	SWBSYF	
0x06	CLK_CKDIVR 复位值	0	0	0	1	1	0	0	0	
0x07	CLK_PCKENR1 复位值	PCLKEN17 1	PCLKEN16 1	PCLKEN15 1	PCLKEN14 1	PCLKEN13 1	PCLKEN12 1	PCLKEN11 1	PCLKEN10 1	
0x08	CLK_CSSR 复位值	0	0	0	0	CSSD	CSSDIE	AUX	CSSEN	
0x09	CLK_CCOR 复位值	0	CCOBSY 0	CCORDY 0	CCOSEL3 0	CCOSEL2 0	CCOSEL1 0	CCOSEL0 0	CCOEN 0	
0x0A	CLK_PCKENR2 复位值	PCKEN27 1	PCKEN26 1	PCKEN25 1	PCKEN24 1	PCKEN23 1	PCKEN22 1	PCKEN21 1	PCKEN20 1	
0x0B	CLK_CANCCR 复位值	x	x	x	x	x	CANDIV2 0	CANDIV1 0	CANDIVO 0	
0x0C	CLK_HSITRIMR 复位值	x	x	x	x	x	HSITRIM2 0	HSITRIM1 0	HSITRIM0 0	
0x0D	CLK_SWIMCCR 复位值	x	x	x	x	x	0	0	SWIMCLK 0	

9 电源管理

默认情况下在系统或电源复位后，MCU处于运行模式。在这种模式下，CPU由 f_{CPU} 提供时钟并执行程序代码，系统时钟分别为各个处于激活状态的外设提供时钟，MCU功耗最大。

在运行模式下，为了保持CPU继续运行并执行代码，有下列几种途径可降低功率消耗：

- 降低系统时钟
- 关闭未使用外设的时钟
- 关闭所有未使用的模拟功能块

但是，如果CPU不需要保持运行，可使用下列三种低功耗模式：

- 等待(Wait)
- 活跃停机(Active Halt)(可配置为慢速或快速唤醒)
- 停机(Halt) (可配置为慢速或快速唤醒)

用户可选择以上三种模式中的一种，并合理配置，以在最低功耗、最快唤醒速度和可使用的唤醒源之间获得最佳平衡点。

9.1 常规考虑

一般来说，低功耗特性在要求节省能量的应用中非常重要。对于要求电池使用寿命较长的便携式应用，超低功耗显得尤为重要。而且这对于环境保护也是至关重要的。

硅片中通常存在两种功耗：

- 静态功耗：由极化电流和漏电流造成。静态功耗很小，只在停机(Halt)模式和活跃停机(Active Halt)模式(参见9.3)下有意义。
- 动态功耗：来自于芯片上正在运行的数字模块。它取决于 V_{DD} ，时钟频率和负载电容。

一个微控制器的功耗取决于：

- V_{DD} 供电电压
- 模拟性能
- MCU大小及数字逻辑门数(漏电流及负载电容)
- 时钟频率
- 处于激活状态的外设数目
- 可用的低功耗模式及级别

微控制器的处理速度也很重要，这使得用户程序只需很短时间处于运行状态，而更多时间处于低功耗模式下。

使用MCU灵活的低功耗特性，用户可在很大范围内降低系统功耗并快速恢复操作。

9.2 低功耗的时钟管理

9.2.1 降低系统时钟

在运行模式，为了即能满足系统性能又能降低功耗，选择合适的系统时钟源是很重要的。可通过写时钟控制寄存器选择时钟源。参见时钟控制章节。

通过写时钟分频寄存器CLK_CKDIVR的位CPUDIV[2:0]，可降低 f_{CPU} 的时钟频率。这会降低CPU的速度，但同时可降低CPU的功耗。其它外设(由 f_{MASTER} 提供时钟)不会受此设置影响。

在运行模式下，任何时候需要恢复全速运行，将CPUDIV[2:0]清0即可。

9.2.2 外设时钟门控

为了更进一步降低功耗，可使用时钟门控。用户可在任意时间打开或关闭 f_{MASTER} 与各个外设的连接。参见时钟控制章节。

此设置在运行模式和等待模式均有效。

9.3 低功耗模式

四种低功耗模式的主要特性如表12。

表12 低功耗模式管理

模式 (功耗级别)	主电压调节器	振荡器	CPU	外设	唤醒触发事件
等待(Wait) (-)	开	开	关	开 ⁽¹⁾	所有内部中断(包括AWU)或外部中断, 复位
快速活跃停机 (Active Halt) (--)	开	关 除LSI(或 HSE)	关	仅AWU和 IWDG(如果已被 激活)	AWU或外部 ⁽²⁾ 中 断, 复位
慢速活跃停机 (Active Halt) (---)	关 (低电压调节器 开)	关 仅LSI除外	关	仅AWU和 IWDG(如果已被 激活)	AWU或外部 ⁽²⁾ 中 断, 复位
停机(Halt) (----)	关 (低电压调节器 开)	关	关	关	外部 ⁽²⁾ 中断, 复位

1. 如果外设时钟未被关闭
2. 包括通讯外设的中断(参见中断向量表)

9.3.1 等待(Wait)模式

在运行模式下执行WFI(等待中断)指令, 可进入等待模式。此时CPU停止运行, 但外设与中断控制器仍保持运行, 因此功耗会有所降低。等待模式可与PCG(外设时钟门控), 降低CPU时钟频率, 以及选择低功耗时钟源(LSI, HSI)相结合使用, 以进一步降低系统功耗。参见时钟控制(CLK)的说明。

在等待模式下, 所有寄存器与RAM的内容保持不变, 之前所定义的时钟配置也保持不变(主时钟状态寄存器CLK_CMSR)。

当一个内部或外部中断请求产生时, CPU从等待模式唤醒并恢复工作。

9.3.2 停机(Halt)模式

在该模式下主时钟停止。即由f_{MASTER}提供时钟的CPU及所有外设均被关闭。因此, 所有外设均没有时钟, MCU的数字部分不消耗能量。

在停机模式下, 所有寄存器与RAM的内容保持不变, 默认情况下时钟配置也保持不变(主时钟状态寄存器CLK_CMSR)。

MCU可通过执行HALT指令进入停机模式。外部中断可将MCU从停机模式唤醒。外部中断指配置为中断输入的GPIO端口或具有触发外设中断能力的端口。

在这种模式下, 为了节省功耗主电压调节器关闭。仅低电压调节器(及掉电复位)处于工作状态。

快速时钟启动

HSI RC的启动速度比HSE快(参见数据手册中电特性参数)。因此, 为了减少MCU的唤醒时间, 建议在进入暂停模式前选择HSI做为f_{MASTER}的时钟源。

在进入停机模式前可通过设置内部时钟寄存器CLK_ICKR的FHWU位选择HSI做为f_{MASTER}的时钟源, 而无需时钟切换。参见时钟控制章节。

9.3.3 活跃停机(Active Halt)模式

活跃停机模式与停机模式类似，但它不需要外部中断唤醒。它使用AWU，在一定的延时后产生一个内部唤醒事件，延迟时间是用户可编程的。

在活跃暂停模式下，主振荡器、CPU及几乎所有外设都被停止。如果AWU和IWD已被使能，则只有LSI RC与HSE仍处于运行状态，以驱动AWU和IWD计数器。

为进入活跃停机模式，需首先使能AWU(如AWU章节所述)，然后执行HALT指令。

主电压调节器自动关闭

默认情况下，为了从活跃停机模式快速唤醒，主电压调节器处于激活状态。但其电流消耗是不可忽视的。

为进一步降低功耗，当MCU进入活跃停机模式时，主电压调节器可自动关闭。通过设置内部时钟寄存器CLK_ICKR的REGAH位可实现此功能。此时：

- MCU内核由低功耗电压调节器(LPVR)供电(如同停机模式)。
- 仅LSI时钟源可用，因为HSE时钟源对于LPVR来说电流消耗太大。

在唤醒时主电压调节器重新被打开，这需要一个比较长的唤醒时间(参见数据手册电特性部分唤醒时间与电流消耗的相关数据)。

快速唤醒时钟

如停机模式所述，为了缩短唤醒时间，建议使用HSI做为f_{MASTER}的时钟源。FHWU位也可用于缩短切换时间。

在活跃停机模式下，快速唤醒是很重要的。这可以提高CPU的执行效率，使MCU处于运行状态与低功耗模式之间的时间最短，从而减少整体平均功耗。

9.4 附加的模拟功耗控制

9.4.1 停机模式下的快速内存唤醒

默认情况下，微控制器进入停机模式后FLASH是处于掉电状态的。此时，漏电流可忽略不计，功耗是非常低的。但FLASH的唤醒时间较长(几微妙)。

如果用户需要从停机模式快速唤醒，可将FLASH_CR1的HALT位置1。当微控制器进入停机模式时，这将确保FLASH处于等待状态，唤醒时间降至几纳秒。但功耗将增至几微安。

详情请参见数据手册电特性章节。

9.4.2 活跃停机模式下的超低内存功耗

在活跃停机模式下，为加快唤醒时间，默认情况下FLASH处于工作状态，因此并没有降低功耗。

为降低功耗，用户可将FLASH_CR1的AHALT位置1。在进入活跃停机模式时，这将停止向FLASH供电以降低功耗，但唤醒时间将增至微秒级。

10 中断控制器(ITC)

10.1 简介

中断控制器提供如下功能：

- 硬件中断的管理
 - 所有 I/O 引脚都具有外部中断能力，每一个端口都有独立的中断向量以及独立的标志。
 - 外设中断能力
- 软件中断的管理(TRAP)
- 具有灵活的优先级和中断等级管理，支持可嵌套的或同级中断管理：
 - 多达4个软件可编程的嵌套等级
 - 最多有32个中断向量，其入口地址由硬件固定
 - 2不可屏蔽的事件：RESET, TRAP
 - 1个不可屏蔽的最高优先级的硬件中断 (TLI)

基于如下资源的中断管理：

- 位I1和I0位于CPU的条件代码寄存器(CCR)
- 软件优先级寄存器 (ITC_SPRx)
- 复位向量地址0x00 8000位于程序空间的起始部分。对于具有启动ROM的型号，ST公司把复位初始化程序固化在ROM区中。
- 固定的中断向量地址位于程序空间映像的高位地址段(0x00 8004 to 0x00 807C)，其地址顺序即为硬件的优先顺序。

10.2 中断屏蔽和处理流程

中断屏蔽是通过CC寄存器的位I1和位I0以及设置每个中断向量([表13](#))的软件优先级的ITC_SPRx来管理的。处理流程如[图17](#)所示：

当一个中断请求必须被响应时：

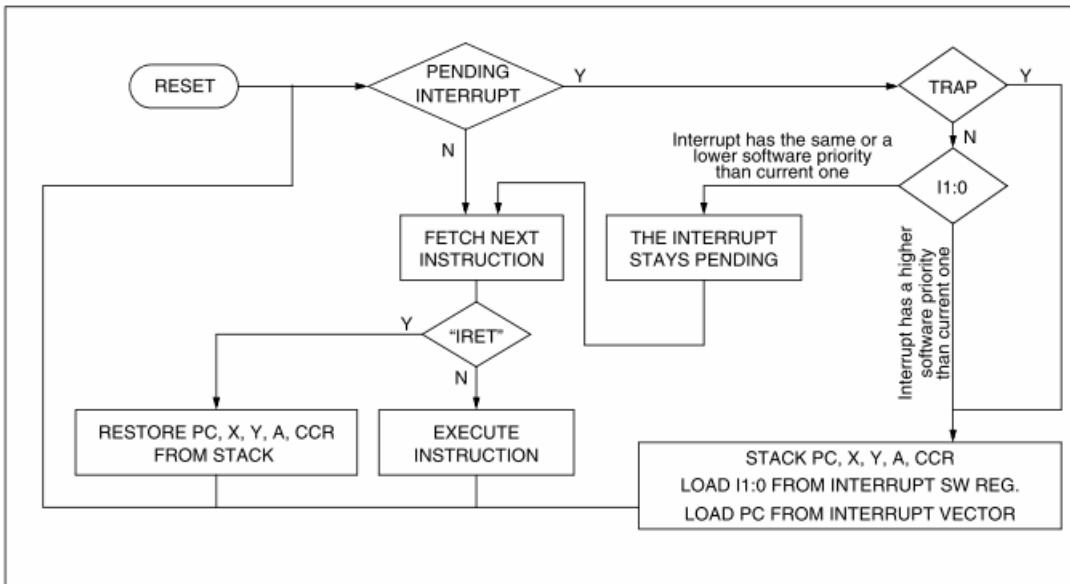
1. 在当前正在执行指令结束之后，正常的操作被悬起；
2. PC,X,Y,A和CC寄存器被自动压栈；
3. 根据ITC_SPRx寄存器中的值对应的中断服务向量，CC寄存器中的位I1和I0被相应设置；
4. 通过中断向量载入中断服务子程序的入口地址，接着对中断服务子程序的第一条指令取址(参考[表16中断映射表](#)来了解向量地址的更详细情况)。

中断服务子程序必须以IRET指令结束，该指令会把堆栈中的保存的寄存器内容出栈，同时由于运行IRET指令，位I1和位I0被重新恢复，程序也恢复运行。

表13 软件优先级

软件优先级别	优先级	I1	I2
0级 (主程序)	低 ↓ 高	1	0
1级		0	1
2级		0	0
3级 (=无软件优先级)		1	1

图17 中断处理流程图



10.2.1 处理等待（排队）的中断

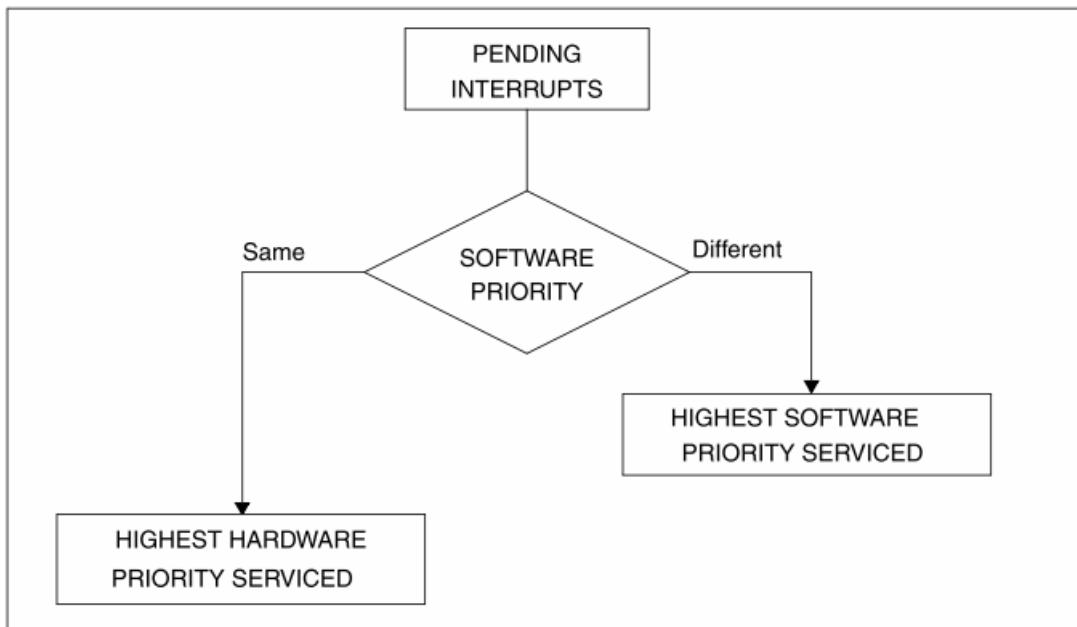
同一时间可以有几个中断排队等待处理。中断响应是根据如下两步来决定的：

1. 最高软件优先级的中断被响应；
2. 如果几个排队的中断具有相同的软件优先级，那么最高硬件优先级的中断先响应。

当中断请求没有立即得到响应时，该中断请求被锁存；当其软件优先级及硬件优先级均为最高的时候，该中断被处理。

- 注意:
- 1 与软件优先级不同，每个中断的硬件优先级是唯一且互不相同的，这样就可保证一个时刻只有一个中断被唯一确定地处理。
 - 2 RESET, TLI 和 TRAP 这几个中断被认为是拥有最高的软件优先级来处理。
 - 3 一个TLI中断可中断除TRAP及RESET之外的3级中断。
- 见 [图18](#) 了解更加详细的排队等待处理的中断服务过程。

图18 优先级处理过程



10.2.2 中断源

STM8中断控制器处理2种类型的中断源：

- 不可屏蔽的中断：RESET, TLI 和 TRAP
- 可屏蔽中断：外部中断或者内嵌的外设中断

不可屏蔽中断源

不可屏蔽中断不会考虑CC寄存器的I1和I0的状态(参见[图17](#))。仅仅当TRAP中断发生时候将PC, X, Y, A 和 CC 寄存器的内容压栈。相应的向量载入到PC寄存器中同时置位I1和I0位禁止中断(3级优先级)。

- TRAP (不可屏蔽的软件中断)

当执行TRAP指令时就响应软件中断。它响应过程如[图17](#)所示的流程图。

TRAP中断不能使处理器从停机(Halt)模式下退出。

- RESET 复位

复位中断是STM8的软件和硬件中断的最高优先级，这也就是说在复位程序的开始所有的中断被禁止。必须通过RIM指令来使能它们(见[表15](#))。

复位中断可以使处理器从停机(Halt)模式退出。

更详细的复位中断管理见复位章节。

- TLI最高等级的硬件中断

当在特定的 I/O边沿检测到在相应的TLI输入时将产生硬件中断。

注意：在[TLI中断服务子程序](#)中禁止使用TRAP指令。

可屏蔽的中断源

对于可屏蔽中断，如果相应的中断被使能，而且如果在ITC_SPRx寄存器的中断优先级比当前正在执行的中断(根据CC寄存器的I1和I0位)的优先级高的话那么就可以被响应。如果上面2个条件中的任何一个不满足那么该中断会被锁存并保持在等待状态。

- 外部中断

外部中断可以用来把MCU从停机(Halt)模式唤醒。外部中断触发方式的选择可以通过软件写控制外部中断控制寄存器(EXTI_CRx)来实现。

当多个连接到同一个中断向量的外部引脚中断被同时选定时候，那么他们是‘逻辑或’的关系。

当外部的电平触发中断被锁存后，如果该给定的电平一直保持到中断子程序结束，那么该电平信号将再次触发中断，除非在中断子程序中禁用该中断。

- 外设中断

大部分的外设中断会导致MCU从停机(Halt)模式下唤醒。见[表16](#)。

当对应外设状态寄存器的中断标志位被置位，同时相应的外设控制寄存器的使能位被置位时将产生一个外设中断。

清除一个外设中断的标准顺序是在对状态寄存器的访问后再对相关寄存器进行读或者写操作。当一个清除过程被执行之后相应的悬起中断(一个将被执行的中断)会丢失。

10.3 中断和低功耗模式

所有的中断都可以使处理器从待机模式(Wait)退出。

仅有外部中断和另外一些特定中断使才能使处理器从停机(Halt)模式退出(请参考[表16](#))。

当MCU从挂起模式唤醒时候,如果有多个排队中断存在，那么第一个被响应的中断一定具有从挂起模式退出的能力。该选择是通过如[图18](#)所示的判断过程实现的。如果最高优先级的待相应的中断不能把设备从挂起模式唤醒的话，那么它将在后续被响应。

如果在执行HALT指令时，有一个内部或外部中断(例如时钟中断)发生，HALT指令会继续执行完毕，但这个中断会立刻调用唤醒进程。

这种情况下MCU实际上是从停机(Halt)模式被唤醒到运行模式，模式切换的延时为 t_{WUH} ，详见数据手册。

10.4 活动等级/低功耗模式的控制

MCU的活动等级的配置是通过编程CFG_GCR寄存器的AL位来实现。(见[1.3全局配置寄存器\(CFG_GCR\)](#))。

该位是用来控制MCU的低功耗模式。在超低功耗的应用中，MCU大部分时间是运行在WFI/Halt模式中，仅在为执行特别任务的时候被唤醒(通过中断)。一些重复的任务可以直接在一个ISR(中断服务子程序)执行完成而不需要返回到主程序。为了处理这样情况，用户可以在进入低功耗模式(通过执行WFI/HALT指令)之前置位AL位，之后中断子程序返回之后就直接回到低功耗模式。由于相关寄存器保存只是在第一次中断会进行所以减少了中断服务程序运行的时间。

在一些非常简单的应用中所有的操作都可以只在ISR中执行。对于一些更复杂的任务，中断子程序要判断是否要启动主程序，可以通过重设AL的简单方式来实现。

例如：一个应用需要通过自动唤醒功能来每隔50ms唤醒一次来检测一些引脚/传感器/按键的状态。如果这些引脚大部分时间是不工作的，那么MCU可以直接回到低功耗模式而不需要运行主程序的。如果其中之一的引脚处于工作状态，那么ISR将要进行相应判断，通过重设AL位来启动主程序。

10.5 同时的和嵌套的中断管理

STM8S提供2种中断管理模式：

- 同时发生模式
- 嵌套模式

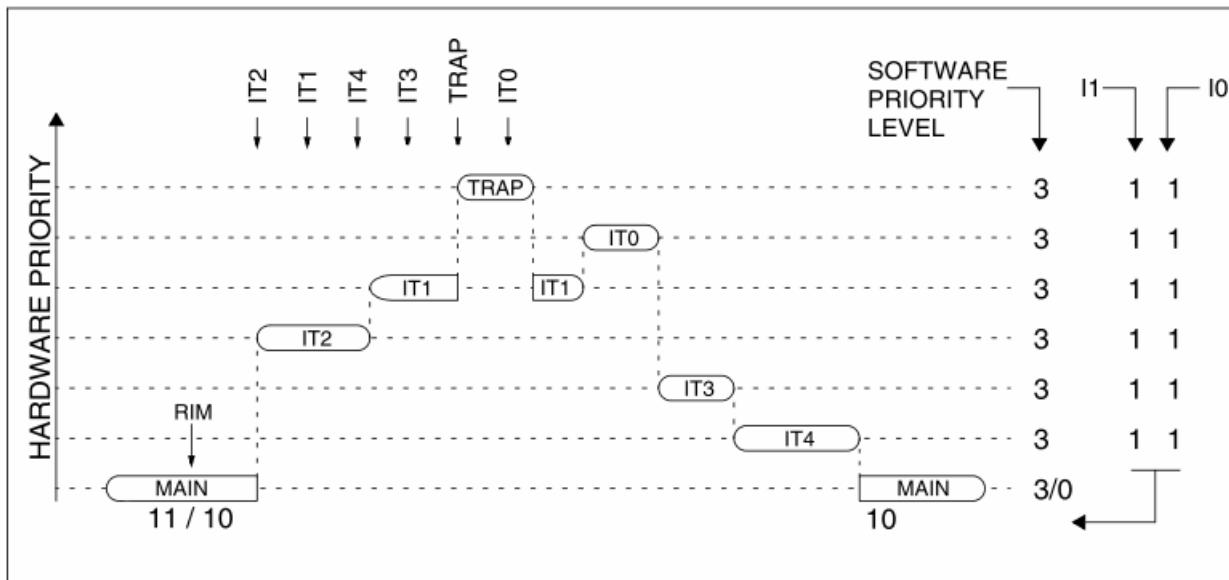
10.5.1 同时发生中断管理模式

在该模式下，所有的中断的中断优先级都是3级，因此它们都是不可以被中断的(除了被TLI, RESET或TRAP中断之外)。

硬件的中断优先级按如下顺序排列，从低到高的优先级是：MAIN, IT4, IT3, IT2, IT1, IT0, TRAP/TLI(同等优先级)以及RESET。

[图19](#)所示是一个同时发生中断管理模式的例子。

图19 同时发生的中断的管理



10.5.2 嵌套中断管理模式

在该模式下，允许在中断子程序中响应中断。一旦一个中断的优先级被设置低于3级时该模式就立即有效。

硬件优先级从低到高按如下顺序给定，即MAIN, IT4, IT3, IT2, IT1, IT0 和TRAP。

通过设定ITC_SPRx寄存器的相应的I1_x和I0_x位来配置每一个中断向量的软件优先级。I1_x和I0_x位具有和CC寄存器的I1 和I0位相同的意思(见表14)。

不可以将中断优先级设为级别0(I1_x=1, I0_x=0)，在这种情况下，该中断的优先级将保持为先前的值。例如：如果先前的值是CFh，然后编程的值是64h，那么结果是44h。

RESET 和 TRAP 向量是没有软件优先级的。当两者的任何一个被响应时，CC寄存器的位I1和I0两位都被置位。

注意：在中断被响应时如果位 I1 和 I0 被修改，那么设备将作如下处理：如果一个中断 X 仍然处在悬起状态（新的中断或者中断标志没有被清除）同时该新的优先级又比先前的优先级高的话，那么该中断 X 会被重新响应。否则该中断的软件优先级在下一个中断请求(X中断的IRET之后)来之前保持不变。

在中断子程序的执行过程中，执行 HALT, POP CC, RIM, SIM 和 WFI 指令会改变当前的软件优先级直到下一条IRET指令被执行或者先前提到的指令之一被执行。详见 10.7。

图20 所示嵌套中断管理的例子。

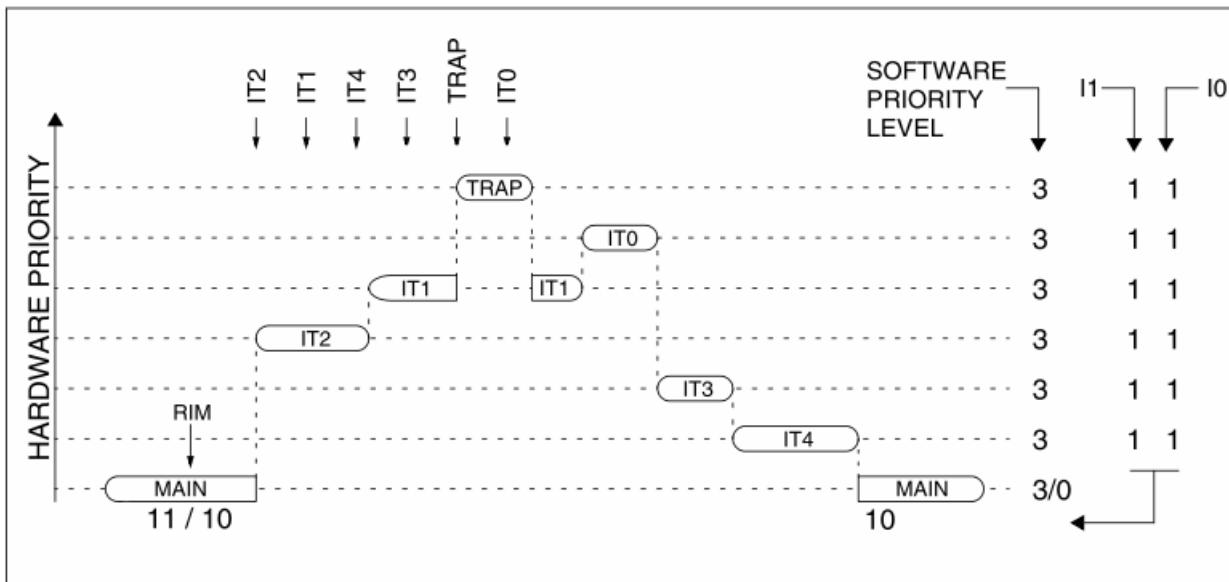
警告：没有标志位及中断来指示堆栈发生溢出

表14 向量地址映象对应软件的优先级位

向量地址	ITC_SPRx 寄存器位
8008h	I1_0 and I0_0 bits(1)
800Ch	I1_1 and I0_1 bits
...	...
80C7h	I1_29 and I0_29 bits

1. ITC_SPRx 寄存器对应于TLI的位可以被读写，但是它们对中断处理的管理是没有作用的。

图20 嵌套中断管理



10.6 外部中断

STM8S为外部中断事件专门分配了五个中断向量：

- Port A 口的5个引脚: PA[6:2]
- Port B 口的8个引脚: PB[7:0]
- Port C 口的8个引脚: PC[7:0]
- Port D 口的7个引脚: PD[6:0]
- Port E 口的8个引脚: PE[7:0]

PD7 是最高优先级的中断源 (TLI)。

为了产生中断，相应的GPIO端口必须被配置为中断使能的输入口，详细内容请参考GPIO章节的寄存器描述部分。

中断的触发方式由外部中断控制寄存器1(EXTI_CR1)和外部中断控制寄存器2(EXTI_CR2)所配置 (见 [10.9.3](#) 和 [10.9.4](#))

10.7 中断指令

表15列出了中断指令：

表15 专用中断指令集

指令	描述	功能/例子	I1	H	I0	N	Z	C
HALT	进入HALT模式		1		0			
IRET	中断程序返回	POP CC,A,X,Y,PC	I1	H	I0	N	Z	C
JRM	如果I1:0=11(3级) 则跳转	I1:0=11 ?						
JRNM	如果I1:0<>11则跳 转	I1:0<>11 ?						
POP CC	CC出栈	Mem => CC	I1	H	I0	N	Z	C
RIM	使能中断(0级设 置)	Load 10 in I1:0 of CC	1		0			
SIM	禁止中断(3级设 置)	Load 11 in I1:0 of CC	1		1			

TRAP	软件中断 TRAP	Software NMI	1		1		
WFI	等待中断		1		0		

10.8 中断映射

表16 中断映射表

中断向量号	中断源	描述	从停机(Halt)模式唤醒功能	从活跃停机(Active Halt)模式唤醒功能	向量地址
	RESET	复位	是	是	8000h
	TRAP	软件中断			8004h
0	TLI	外部最高级中断			8008h
1	AWU	自动唤醒HALT模式中断		是	800Ch
2	CLK	时钟控制器			8010h
3	EXTI0	端口A外部中断	是	是	8014h
4	EXTI1	端口B外部中断	是	是	8018h
5	EXTI2	端口C外部中断	是	是	801Ch
6	EXTI3	端口D外部中断	是	是	8020h
7	EXTI4	端口E外部中断	是	是	8024h
8	CAN	CAN RX 中断	是	是	8028h
9	CAN	CAN TX/ER/SC 中断			802Ch
10	SPI	发送完成	是	是	8030h
11	TM1	更新/上溢出/下溢出/触发/刹车			8034h
12	TM1	捕获/比较			8038h
13	TM2	更新/上溢出			803Ch
14	TM2	捕获/比较			8040h
15	TM3	更新/上溢出			8044h
16	TM3	捕获/比较			8048h
17	UART1	发送完成			804Ch
18	UART1	接收寄存器满			8050h
19	I2C	I2C中断	是	是	8054h
20	UART2/3	发送完成			8058h
21	UART2/3	接收寄存器满			805Ch
22	ADC	转换结束			8060h
23	TIM4	更新/上溢出			8064h
24	FLASH	编程结束/禁止编程			8068h
保留					806Ch到807Ch

10.9 ITC寄存器

10.9.1 CPU CC 寄存器中断位

地址: 请参考数据手册通用硬件寄存器映射表

复位值: 0x28

7	6	5	4	3	2	1	0
V	-	I1	H	I0	N	Z	C

读写权限: r, rw

位5, 3	I[1:0] :软件中断优先级位 这两位表明当前中断请求的优先级。当一个中断请求发生时, 相应的中断向量的软件优先级自动从(ITC_SPRx)载入I[1:0]。 I[1:0] 可以通过RIM, SIM, HALT, WFI, IRET或者PUSH/POP等指令来软件置位和清零。(参见图20)			
	I1	I0	优先级	级别
	1	0	0级(主程序)	低 ↓ 高
	0	1	级别1	
	0	0	级别2	
	1	1	3级(=禁用软件优先级)	

注意: TLI, TRAP 和RESET可以中断一个级别为3的程序。

10.9.2 软件优先级寄存器 x (ITC_SPRx)

地址偏移值: 0x00 到 0x07

复位值: 0xFF

7 6 5 4 3 2 1 0

ITC_SPR1	VECT3SPR[1:0]	VECT2SPR[1:0]	VECT1SPR[1:0]	VECT0SPR[1:0]			
ITC_SPR2	VECT7SPR[1:0]	VECT6SPR[1:0]	VECT5SPR[1:0]	VECT4SPR[1:0]			
ITC_SPR3	VECT11SPR[1:0]	VECT10SPR[1:0]	VECT9SPR[1:0]	VECT8SPR[1:0]			
ITC_SPR4	VECT15SPR[1:0]	VECT14SPR[1:0]	VECT13SPR[1:0]	VECT12SPR[1:0]			
ITC_SPR5	VECT19SPR[1:0]	VECT18SPR[1:0]	VECT22SPR[1:0]	VECT16SPR[1:0]			
ITC_SPR6	VECT23SPR[1:0]	VECT22SPR[1:0]	VECT21SPR[1:0]	VECT20SPR[1:0]			
ITC_SPR7	VECT27SPR[1:0]	VECT26SPR[1:0]	VECT25SPR[1:0]	VECT24SPR[1:0]			
ITC_SPR8	保留		VECT29SPR[1:0]	VECT28SPR[1:0]			

rw rw rw rw rw rw rw rw

位7:0	VECTxSPR[1:0]: 向量X的软件优先级位 通过软件对这8个读/写寄存器(ITC_SPR1到ITC_SPR8)的操作, 可以定义各个中断向量的软件优先级。 注意: 禁止写10B,(优先级0)。 详细列表在 表14 。 参考 10.9.1 。各个向量优先级编程位的值。 ITC_SPR1 位 1:0 由硬件强制设置1 (TLI) ITC_SPR8 位7:4 由硬件强制设置1

10.9.3 外部中断控制寄存器 1 (EXTI_CR1)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
PDIS[1:0]	PCIS[1:0]	PBIS[1:0]	PAIS[1:0]				

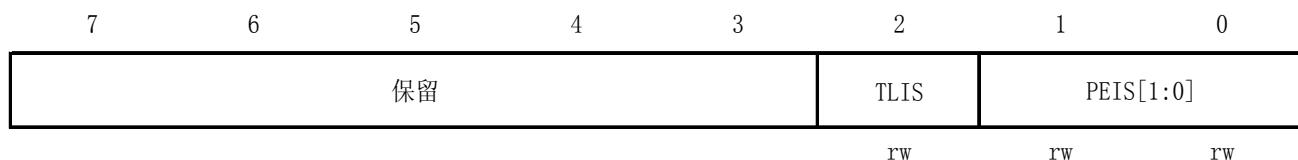
rW rW rW rW rW rW rW rW

位7:0	PDIS[1:0] : PORT D 的中断触发位 这些位仅在CC寄存器的I1和I0位都为1(级别3)时才可以写入。这些位定义端口D 的中断触发位 00: 下降沿和低电平触发 01: 仅上升沿触发 10: 仅下降沿触发 11: 上升沿和下降沿触发
位5:4	PCIS[1:0] : PORT C 的中断触发位 这些位仅在CC寄存器的I1和I0位都为1(级别3)时才可以写入。这些位定义端口C 的中断触发位 00: 下降沿和低电平触发 01: 仅上升沿触发 10: 仅下降沿触发 11: 上升沿和下降沿触发
位3:2	PBIS[1:0] : PORT B 的中断触发位 这些位仅在CC寄存器的I1和I0位都为1(级别3)时才可以写入。。这些位定义端口B 的中断触发位 00: 下降沿和低电平触发 01: 仅上升沿触发 10: 仅下降沿触发 11: 上升沿和下降沿触发
位1:0	PAIS[1:0] : PORT A 的中断触发位 这些位仅在CC寄存器的I1和I0位都为1(级别3)时才可以写入。。这些位定义端口A 的中断触发位 00: 下降沿和低电平触发 01: 仅上升沿触发 10: 仅下降沿触发 11: 上升沿和下降沿触发

10.9.4 外部中断控制寄存器 1 (EXTI_CR2)

地址偏移值: 0x01

复位值: 0x00



位7:4	保留位, 须保持清零
位2	<p>TLIS: 高级中断触发位 此位由软件设置。此位仅在外部相应的中断引脚PD7禁止中断时才能写入。</p> <p>0: 下降沿触发 1: 上升沿触发</p>
位1:0	<p>PEIS[1:0] : PORT E 的中断触发位 这些位仅仅在CC寄存器的I1和I0位都为1(级别3)时才可以写入。这些位定义端口E的中断触发位</p> <p>00: 下降沿和低电平触发 01: 仅上升沿触发 10: 仅下降沿触发 11: 上升沿和下降沿触发</p>

10.9.5 寄存器表和复位值

表17 中断寄存器表

地址偏移	寄存器	7	6	5	4	3	2	1	0
ITC-SPR 模块(1)									
0x00	ITC_SPR 复位值	VECT3SPR1 1	VECT3SPR0 1	VECT2SPR1 1	VECT2SPR0 1	VECT1SPR1 1	VECT1SPR0 1	VECT0SPR1 1	VECT0SPR0 1
0x01	ITC_SPR 复位值	VECT7SPR1 1	VECT7SPR0 1	VECT6SPR1 1	VECT6SPR0 1	VECT5SPR1 1	VECT5SPR0 1	VECT5SPR1 1	VECT5SPR0 1
0x02	ITC_SPR 复位值	VECT11SPR1 1	VECT11SPR0 1	VECT10SPR1 1	VECT10SPR0 1	VECT9SPR1 1	VECT9SPR0 1	VECT8SPR1 1	VECT8SPR0 1
0x03	ITC_SPR 复位值	VECT15SPR1 1	VECT15SPR0 1	VECT14SPR1 1	VECT14SPR0 1	VECT13SPR1 1	VECT13SPR0 1	VECT12SPR1 1	VECT12SPR0 1
0x04	ITC_SPR 复位值	VECT19SPR1 1	VECT19SPR0 1	VECT18SPR1 1	VECT18SPR0 1	VECT17SPR1 1	VECT17SPR0 1	VECT16SPR1 1	VECT16SPR0 1
0x05	ITC_SPR 复位值	VECT23SPR1 1	VECT23SPR0 1	VECT22SPR1 1	VECT22SPR0 1	VECT21SPR1 1	VECT21SPR0 1	VECT20SPR1 1	VECT20SPR0 1
0x06	ITC_SPR 复位值	VECT27SPR1 1	VECT27SPR0 1	VECT26SPR1 1	VECT26SPR0 1	VECT25SPR1 1	VECT25SPR0 1	VECT24SPR1 1	VECT24SPR0 1
0x07	CLK_PCKENR1 复位值	-	-	-	-	-	-	VECT24SPR1 1	VECT24SPR0 1
ITC-EXTI 模块(2)									
0x00	EXTI_CR1 复位值	PDIS1 0	PDIS0 0	PCIS1 0	PCIS0 0	PBIS1 0	PBIS0 0	PAIS1 0	PAIS0 0
0x01	EXTI_CR2 复位值	-	-	-	-	-	-	PEIS1 0	PEIS0 0

- 相对于ITC-SPR模块基地址的偏移值(请参考数据手册中CPU/SWIM/调试模块/中断控制器的寄存器表)。
- 相对于ITC-EXTI模块基地址的偏移值(请参考数据手册中CPU/SWIM/调试模块/中断控制器的寄存器表)。

11 通用输入输出口(GPIO)

11.1 简介

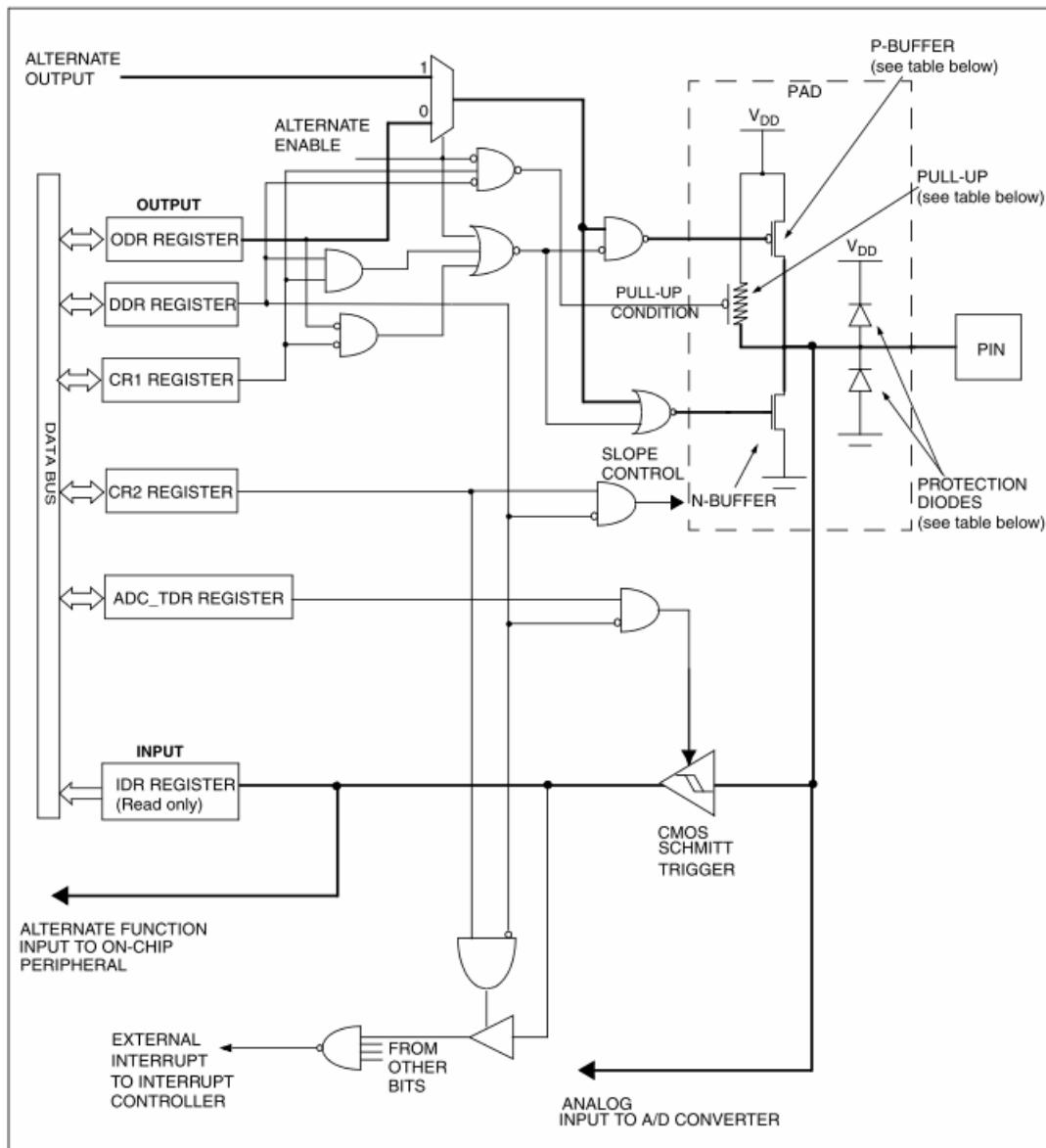
通用输入/输出口用于芯片和外部进行数据传输。一个IO端口可以包括多达8个引脚，每个引脚可以被独立编程作为数字输入或者数字输出口。另外部分口还可能具有如模拟输入，外部中断，片上外设的输入/输出等复用功能。但是在同一时刻仅有一个复用功能可以映射到引脚上。复用功能的映射是通过选项字节控制的。请参考数据手册关于选项字节的描述。

每个端口都分配有一个输出数据寄存器，一个输入引脚寄存器，一个数据方向寄存器，一个选择寄存器，和一个配置寄存器。一个I/O口工作在输入还是输出是取决于该口的数据方向寄存器的状态。

11.2 GPIO主要功能

- 端口的各个位可以被单独配置
- 可选择的输入模式：浮动输入和带上拉输入
- 可选择的输出模式：推挽式输出和开漏输出
- 数据输入和输出采用独立的寄存器
- 外部中断可以单独使能和关闭
- 输出摆率控制用以减少EMC噪声
- 片上外设的I/O功能复用
- 当作为模拟输入时可以关闭输入施密特触发器来降低功耗
- 在数据输出锁存时支持读-修改-写
- 输入兼容 5V电压
- I/O口工作电压范围为1.6 V 到 $V_{DDIOmax}$

图21 GPIO模块框图



11.3 I/O的配置和使用

每一个端口都有一个输出数据寄存器 (ODR)，一个引脚输入寄存器 (IDR)和一个数据方向寄存器 (DDR) 总是同相关的。

控制寄存器1(CR1)和控制寄存器2(CR2)用于对输入/输出进行配置。任何一个I/O引脚可以通过对DDR,ODR,CR1和CR2寄存器的相应位进行编程来配置。

寄存器中的位n对应于口的引脚 n。

各种不同配置总结如表18。

表18 I/O口配置表

配置模式	DDR 位	CR1 位	CR2 位	配置模式	上拉电 阻	P- Buffer	二极管	
							连接VDD	连接VSS
输入	0	0	0	悬浮输入	OFF	OFF	ON	ON
	0	1	0	上拉输入	ON			
	0	0	1	中断悬浮输入	OFF			

	0	1	1	中断上拉输入	ON			
输出	1	0	0	开漏输出	OFF	OFF		
	1	1	0	推挽输出		ON		
	1	X	1	输出(最快速度 10MHz)		CR1 位决定		
	X	X	X	真正的开漏输出(特定引脚)	未采用			

注意：连接VDD的二极管在实际开漏极状态引脚是无效的，在引脚和VOL之间的局部保护设备重要性是有效的。

11.3.1 输入模式

将DDRx 位清零就选择了输入模式。在该模式下读IDR寄存器的位将返回对应I/O引脚上的电平值。

请参考 [11.7](#) 来了解关于模拟输入，外部中断，和施密特触发使能/关闭的细节。

如 [表18](#) 所示，理论上可以通过软件配置得到四种不同的输入模式：悬浮不带中断输入，悬浮带中断输入，上拉不带中断输入和上拉带中断输入。但是在实际情况下不是所有的口都具有外部中断能力和上拉，用户应参考数据手册中关于每个引脚的实际硬件性能描述来了解更多细节。

11.3.2 输出模式

将DDRx 位置1就选择了输出模式。在该模式下向ODR寄存器的位写入数据将会通过锁存器输出对应数字值到I/O口。读IDR的位将会返回相应的I/O引脚电平值。通过软件配置CR1，CR2寄存器可以得到不同的输出模式：上拉输出，开漏输出。

更多信息请参考 [11.8](#)。

11.4 复位后的默认配置

复位之后，所有的引脚都是悬浮输入模式。

11.5 没有使用的引脚

没有使用的I/O引脚必须连接到一个固定的电平值。或者是上拉或者是下拉。

11.6 低功耗模式

表19 低功耗模式对GPIO口的影响

模式	描述
等待(Wait)	对I/O口无影响。外部中断可以使MCU退出等待(Wait)模式
停机(Halt)	对I/O口无影响。外部中断可以使MCU从停机(Halt)模式唤醒

注意：如果PA1/PA2被用来连接外部谐振器，为了确保在HALT模式下有最低功耗必须配置PA1和PA2为带上拉输入。

11.7 输入模式的详述

11.7.1 复用功能输入

部分I/O口可以被用作复用功能输入。例如：可以被用来作为输入到定时器的输入捕捉口。复用的输入功能是不会自动选择的，用户可以通过写相应的外设寄存器的控制位来选择复用功能。

对于复用功能的输入，用户必须通过配置DDR和CR1寄存器设置将对应的I/O口设为悬浮或是上拉输入。

11.7.2 中断功能

用户可以在I/O引脚为输入模式时通过设置Px_CR2寄存器的相应位来配置某个I/O作为外部输入中断模式。在该配置下，I/O引脚上的一个信号沿或是低电平会产生一个中断请求。

在EXTI_CR[2:1]寄存器中对于每一个中断向量都可以独立编程为上升沿或下降沿触发。

外部中断只有在对应I/O口被设置为输入模式下才有效。

中断屏蔽

通过对Px_CR2寄存器的相应位进行编程来单独使能/关闭外部中断功能。复位后外部中断是关闭的。

11.7.3 模拟通道

ADC外设可以选择某些I/O口作为模拟输入通道。如下面一节描述，当使用模拟通道的时候，ADC_TDR寄存器的输入施密特触发器必须被关闭。

表20 推荐的和不推荐的模拟输入配置

DDR	CR1	CR2	ADC_TDR	配置	说明
0	0	0	1	浮空输入。无中断。禁止施密特触发	推荐模拟输入配置方式
0	1	X	X	带上拉输入	不推荐这种模拟输入配置方式。如果引脚上有模拟电压，这样的配置将使输入脚上产生额外的电流。
1	0	X	X	输出	
1	1	X	X	输出	

11.7.4 施密特触发器

部分I/O口包括一个内嵌的输入施密特触发器。可以通过ADC_TDR施密特触发器禁止寄存器来使能/禁止施密特触发器。

11.8 输出模式详述

11.8.1 复用功能的输出

复用输出功能为外设输出到外部或者I/O引脚提供一个方便的操作方法。当复用功能使能时，复用功能模块接管了输出锁存寄存器(Px_ODR)并强制Px_ODR相应的位为1。

复用输出功能可以是上拉或者开漏输出，取决于外设本身和控制寄存器1(Px_CR1)，输出摆率取决于控制寄存器2 (Px_CR2)的值。

例如：

考虑到要达到最佳性能，SPI输出引脚必须设置为上拉，快速摆率。UART_Tx可以被配置为或者是上拉或者是开漏带外部上拉来实现多从机的配置。

11.8.2 摆率控制

输出摆率可以使用CR2的相应位通过软件控制。置位CR相应位选择为10MHz的输出频率。该功能既可以用在开漏也可以用在带上拉的输出模式I/O口，输出类型为O3 或者O4。请参考引脚描述表来了解每个口的输出类型。

11.9 GPIO 寄存器

注意：每一个端口寄存器位驱动相应的端口的引脚

11.9.1 端口 x 输出数据寄存器 (Px_ODR)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0

rw rw rw rw rw rw rw rw

位7:0

ODR[7:0]: 端口输出数据寄存器位

在输出模式下, 写入寄存器的数值通过锁存器加到相应的引脚上。读ODR寄存器, 返回之前锁存的寄存器值。

在输入模式下, 写入ODR的值将被锁存到寄存器中, 但不会改变引脚状态。ODR寄存器在复位后总是为0。位操作指令(BSET, BRST) 可以用来设置DR寄存器来驱动相应的引脚, 但不会影响到其他引脚。

11.9.2 端口 x 输入寄存器 (Px_IDR)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0

r r r r r r r r

位7:0

IDR[7:0]: 端口输入数据寄存器位

不论引脚是输入还是输出模式，都可以通过该寄存器读入引脚状态值。该寄存器为只读寄存器。

0:逻辑低电平

1:逻辑高电平

11.9.3 端口 x 数据方向 (Px_DDR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
rw							

位7:0

DDR[7:0]: 数据方向寄存器位

这些位可通过软件置1或置0, 选择引脚输入或输出

0: 输入模式

1: 输出模式

11.9.4 端口 x 控制寄存器 1 (Px_CR1)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
C17	C16	C15	C14	C13	C12	C11	C10

rw rw rw rw rw rw rw rw

位7:0

C1[7:0]控制寄存器位

这些位可通过软件置1或置0, 用来在输入或输出模式下选择不同的功能。请参考[表18](#)

在输入模式时(**DDR=0**):

0: 浮空输入

1: 带上拉电阻输入

在输出模式时(**DDR=1**):

0: 模拟开漏输出(不是真正的开漏输出)

1: 推挽输出, 由CR2相应的位做输出摆率控制

注意: 该位对于真正的开漏输出端口是无影响的(请参考数据手册中标志为‘T’的引脚描述)。

11.9.5 端口 x 控制寄存器 2 (Px_CR2)

地址偏移值: 0x04

复位值: 0x00

7	6	5	4	3	2	1	0
C27	C26	C25	C24	C23	C22	C21	C20

rW rW rW rW rW rW rW rW

位7:0	C2[7:0] 控制寄存器位
	相应的位通过软件置1或置0，用来在输入或输出模式下选择不同的功能。在输入模式下，由CR2相应的位使能中断。如果该引脚无中断功能，则对该引脚无影响。 在输出模式下，置位将提高IO速度。此功能适用O3和O4输出类型。(参见引脚描述表) 在输入模式时(DDR=0): 0: 禁止外部中断 1: 使能外部中断 在输出模式时(DDR=1): 0: 输出速度最大为2MHZ. 1: 输出速度最大为10MHZ

11.9.6 GPIO 寄存器表和复位值

每一个GPIO端口都有5个对应的寄存器，如表21所示。请参考相应的数据手册中寄存器映射部分来了解每个端口的基址。

注意： 初始复位时，所有引脚设置为浮空输入。

表21 GPIO 寄存器表

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	Px_ODR	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	复位值	0	0	0	0	0	0	0	0
0x01	Px_IDR	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	复位值	0	0	0	0	0	0	0	0
0x02	Px_DDR	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
	复位值	0	0	0	0	0	0	0	0
0x03	Px_CR1	C17	C16	C15	C14	C13	C12	C11	C10
	复位值	0	0	0	0	0	0	0	0
0x04	Px_CR2	C27	C26	C25	C24	C23	C22	C21	C20
	复位值	0	0	0	0	0	0	0	0

12 自动唤醒(AWU)

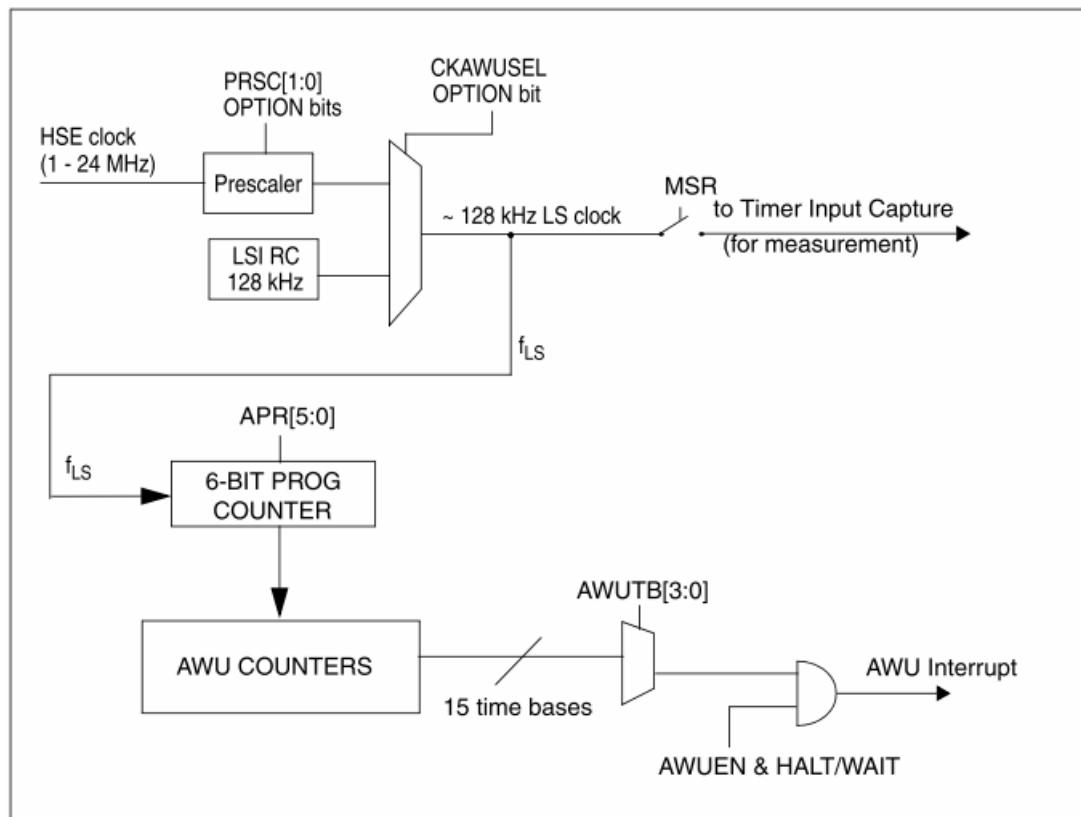
12.1 简介

AWU是用来当MCU进入低功耗的活跃停机(Active Halt)模式时提供一个内部的唤醒时间基准。该时间基准的时钟是由内部的低速RC振荡器时钟(LSI)或者通过预分频的HSE晶振时钟来提供的。

LSI 时钟测量

在使用LSI低速内部时钟时,为了确保最好的精度,它的频率可以通过TIM3的输入捕捉1来测定。

图22 AWU 时钟框图



注意: LS低速时钟源的选择是通过编程CKAWUSEL选项位来实现的。详见时钟控制器章节。

12.2 AWU功能描述

12.2.1 AWU 操作

为了使用AWU功能,按顺序执行如下步骤:

1. 使用AWU_CSR寄存器的MSR位和TIM3的输入捕捉通道1来检测LS的时钟频率;
2. 通过写AWU_APR的APR[5:0]位来定义适当的预分频值;
3. 通过写AWU_TBR的AWUTB[3:0]来选择需要的自动唤醒延时;
4. 置位AWU_CSR 的AWUEN位;
5. 执行HALT指令。

注意: 计数器仅仅在HALT指令之后MCU进入活跃停机模式时才开始计数(请参考电源管理的活跃停机模式章节), AWU中断同时被使能。

预分频计数器仅仅在APR[5:0]值不同于它的复位值0x3F值时才开始计数。

空闲模式

如果不使用AWU，必须载入‘0000’值到AWU_TBR的AWUTB[3:0]位来降低功耗。

12.2.2 时基选择

请参考AWU_PAR和AWU_TBR的说明。

AWU的时间间隔取决于 AWUTB[3:0] 位的值和 APR[5:0] 位的值 (APR_{DIV})，可以定义15种不重叠的时间间隔，如下所示：

表22 AWUTB[3:0] 选择

AWUTB[3:0]	Time interval range	APR _{DIV} range
0b0001	$2/f_{LS} - 64/f_{LS}$	2 to 64
0b0010	$2 \times 32/f_{LS} - 2 \times 64/f_{LS}$	32 to 64
0b0011	$2 \times 2 \times 32/f_{LS} - 2^2 \times 64/f_{LS}$	32 to 64
0b0100	$2^2 \times 2 \times 32/f_{LS} - 2^3 \times 64/f_{LS}$	32 to 64
...		
0b1100	$2^{10} \times 2 \times 32/f_{LS} - 2^{11} \times 64/f_{LS}$	32 to 64
0b1101	$2^{11} \times 2 \times 32/f_{LS} - 2^{12} \times 64/f_{LS}$	32 to 64
0b1110	$2^{11} \times 130/f_{LS} - 2^{11} \times 320/f_{LS}$	26 to 64
0b1111	$2^{11} \times 330/f_{LS} - 2^{12} \times 960/f_{LS}$	11 to 64

为了获得AWUTB[3:0]和APR_{DIV}的正确值，用户必须根据期望的时间间隔值来找出一个对应的间隔范围，从而找出对应的AWUTB[3:0]值。然后选择APR_{DIV}的值来得到一个尽可能接近期望的时间间隔值。这个也可以使用AWU_TBR描述中列出的公式获得。

注意：如果目标值在 $2^{12} \times 64/f_{LS}$ 和 $2^{11} \times 130/f_{LS}$ 或者 $2^{11} \times 320/f_{LS}$ 和 $2^{11} \times 330/f_{LS}$ 之间时，必须选择更靠近目标的那个值。

表23 当 $f_{LS}=128$ kHz，目标时间是 78.5 ms 时的一个例子

AWUTB[3:0]	Interval range	APR _{DIV} range
0001	0.015625 ms - 0.5 ms	2 to 64
0010	0.5 ms - 1.0 ms	32 to 64
...		
1000	32 ms - 64 ms	32 to 64
1001	64 ms - 128 ms	32 to 64
...		
1101	1.024 s - 2.048 s	32 to 64
1110	2.080 s - 5.120 s	26 to 64
1111	5.280 s - 30.720 s	11 to 64

正确的TB[3:0]值是 1001，“理想的 APR_{DIV} = $0.0785 \times f_{LS}/2^8 = 39.25$ ”。因此可以分配给 APR_{DIV} 的值就是 39，这样给定的实际的间隔时间是 78 ms。

12.2.3 LSI 低速内部时钟频率检测

在经过出厂校验后，在全温度范围内低速内部RC(LSI)振荡器的频率离散性是128 kHz +/- 12.5%。为了获得精确的AWU时间间隔或者蜂鸣器输出，必须精确测量LSI频率。

可采用如下的步骤：

参照2009年1月 RM0016 Reference Manual STM8S microcontroller family 英文第4版
本译文仅供参考，如有翻译错误，请以英文原稿为准。请读者随时注意在ST网站下载更新版本

1. 将AWU_CSR的MSR位置1来把LSI的内部时钟连接到TIM3定时器的ICAP1;
2. 通过定时器的输入捕捉中断来测量LSI的时钟频率;
3. 到向AWU_APP的 APR [5:0] 位写入一个适当的值来调整AWU定时间隔到期望的时间间隔。AWUTB[3:0]位可以被更改来选择不同的时间间隔。

LSI的时钟频率测量方法也可以被用来校准蜂鸣器的频率(见 [13.2.2](#))。

12.3 AWU 寄存器

12.3.1 控制/状态寄存器 (AWU_CSR)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
保留		AWUF	AWUEN				MSR
		rw	rw	rw	rw	rw	rw

位7:6	保留, 须保持清零
位5	AWUF: 自动唤醒标志位 此位在自动唤醒模块产生中断时被置位, 通过读AWU_CSR清零。写操作不影响此位的数值。 0: 无自动唤醒中断产生 1: 自动唤醒中断产生
位4	AWUEN: 自动唤醒使能位 此位由软件置位和清零。由此位使能自动唤醒功能。如果MCU进入Active-halt或 Wait 模式, 则自动唤醒模块按照预先编程设置延时一段时间唤醒MCU。 0: 禁止自动唤醒功能 1: 使能自动唤醒功能
位3: 1	保留, 须保持清零
位 0	MSR: 测量使能位 此位使能fLS 时钟连接到TM3的输入捕获。允许定时器测量低速时钟频率 0: 禁止测量功能 1: 使能测量功能

12.3.2 异步预分频寄存器 (AWU_APR)

地址偏移值: 0x01

复位值: 0x3F

7	6	5	4	3	2	1	0
保留	APR[5:0]						
	rW	rW	rW	rW	rW	rW	rW

位7:6	保留, 须保持清零
位5:0	<p>APR[5:0]: 异步分频</p> <p>此位由软件设置选择提供给计数器时钟的分频值</p> <p>00h: APRDIV = 2</p> <p>01h: APRDIV = 3</p> <p>...</p> <p>06h: APRDIV = 8</p> <p>...</p> <p>0Eh: APRDIV = 16</p> <p>0Fh: APRDIV = 17</p> <p>....</p> <p>3Eh: APRDIV = 64</p> <p>注意: 此寄存器不能设置成其初始复位值(3Fh)</p>

12.3.3 时基选择寄存器 (AWU_TBR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
保留				AWUTB[3:0]			
				rw	rw	rw	rw

位7:4	保留, 须保持清零		
位3:0	AWUTB[3:0]: 自动唤醒时基选择 此位由软件设置选择自动唤醒时基, 来定义AWU自动唤醒中断间隔时间。AWU自动唤醒中断由AWUEN=1 来使能。 0000: 无自动唤醒中断 0001: APR_{DIV}/f_{LS} 0010: $2 \times APR_{DIV}/f_{LS}$ 0011: $2^2 APR_{DIV}/f_{LS}$ 0100: $2^3 APR_{DIV}/f_{LS}$ 0101: $2^4 APR_{DIV}/f_{LS}$ 0110: $2^5 APR_{DIV}/f_{LS}$ 0111: $2^6 APR_{DIV}/f_{LS}$ 1000: $2^7 APR_{DIV}/f_{LS}$ 1001: $2^8 APR_{DIV}/f_{LS}$ 1010: $2^9 APR_{DIV}/f_{LS}$ 1011: $2^{10} APR_{DIV}/f_{LS}$ 1100: $2^{11} APR_{DIV}/f_{LS}$ 1101: $2^{12} APR_{DIV}/f_{LS}$ 1110: $5 \times 2^{11} APR_{DIV}/f_{LS}$ 1111: $30 \times 2^{11} APR_{DIV}/f_{LS}$		

12.3.4 AWU 寄存器表和复位值

表24 AWU寄存器表

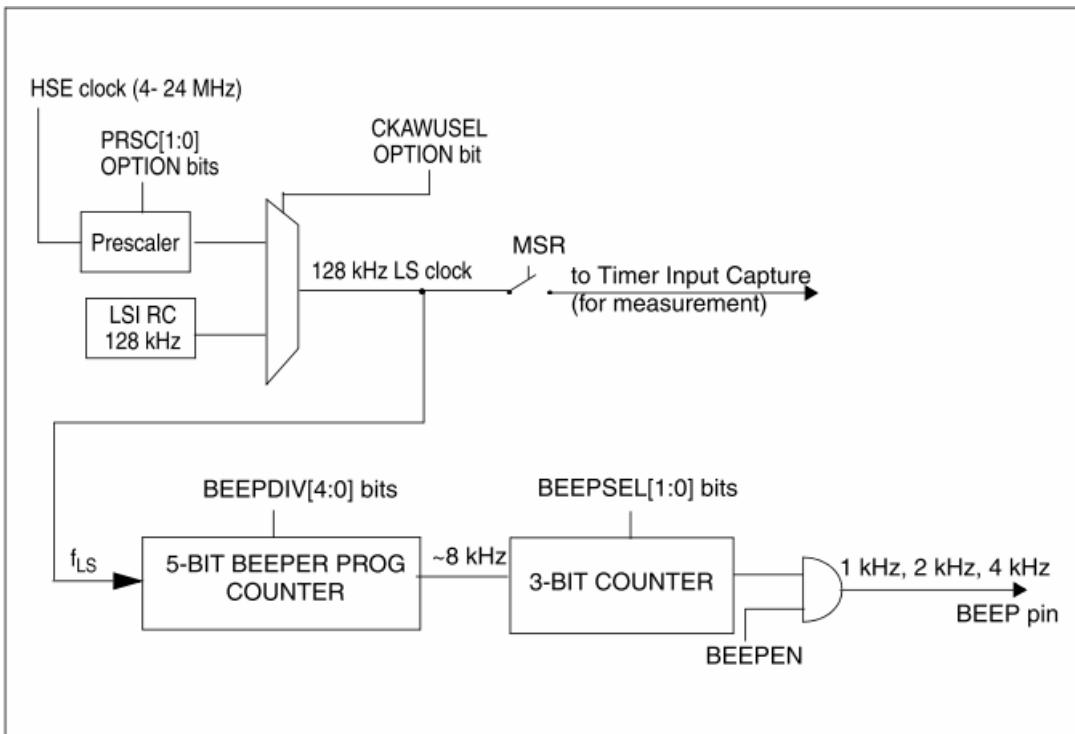
地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	AWU_CSR	—	—	AWUF	AWUEN	—	—	—	MSR
	复位值	0	0	0	0	0	0	0	0
0x01	AWU(APR	—	—	APR5	APR4	APR3	APR2	APR1	APR0
	复位值	0	0	0	0	0	0	0	0
0x02	AWU_TBR	—	—	—	—	AWUTB3	AWUTB2	AWUTB1	AWUTB0
	复位值	0	0	0	0	0	0	0	0

13 蜂鸣器(BEEP)

13.1 简介

当LS时钟工作在128kHz时可产生频率为1kHz, 2 kHz或者4 kHz的蜂鸣信号。

图23 蜂鸣器功能图



13.2 功能描述

13.2.1 蜂鸣器操作

为了使用蜂鸣功能，按顺序执行如下的步骤：

1. 根据 13.2.2 中描述的方法确定BEEPDIV[4:0]的值来校准LS时钟的频率；
2. 通过写BEEP_CSR的 BEEPSEL[1:0] 位来选择1 kHz, 2 kHz 或 4 kHz 的输出频率；
3. 置位BEEP_CSR的 BEEPEN 位来使能LS的时钟源；

注意： 预分频计算器仅仅在当BEEPDIV[4:0]的值不同于复位值0x1F时才开始运行。

13.2.2 蜂鸣器校准

该步骤可以用来校准LS 128 kHz 的时钟以便达到标准的1 kHz, 2 kHz 或 4 kHz 频率输出。

采用如下的步骤：

1. 测量LSI的时钟频率(请参考 12.2.3)
2. 采用如下方法计算BEEP_{DIV}的值，这里 A 和 x 是 f_{LS}/8 (kHz) 的整数和小数部分值：
当 x 小于或者等于 A/(1+2*A)时， BEEP_{DIV} = A-2 ；
否则 BEEP_{DIV} = A-1
3. 将BEEP_{DIV}值写入到BEEP_CSR的BEEPDIV[4:0] 位。

13.3 蜂鸣器 寄存器

13.3.1 蜂鸣器 控制/状态 寄存器 (BEEP_CSR)

地址偏移值: 0x00

复位值: 0x1F

7	6	5	4	3	2	1	0		
				BEEPDIV[4:0]					
RW	RW	RW	RW	RW	RW	RW	RW		

位7:6	BEEPSEL[1:0] :蜂鸣频率选择 00: 输出 $f_{LS}/(8 \times \text{BEEPDIV})$ kHz 01: 输出 $f_{LS}/(4 \times \text{BEEPDIV})$ kHz 1x: 输出 $f_{LS}/(2 \times \text{BEEPDIV})$ kHz
位5	BEEPEN 蜂鸣器允许 此位由软件设置和清零, 使能蜂鸣器功能 0: 禁止蜂鸣器功能 1: 使能蜂鸣器功能
位4	BEEPDIV[4:0] 蜂鸣器预分频器 此位由软件置位和清零。设置蜂鸣器分频因数BEEPDIV. 00h: BEEPDIV = 2 01h: BEEPDIV = 3 ... 0Eh: BEEPDIV = 16 0Fh: BEEPDIV = 17 ... 1Eh: BEEPDIV = 32 注意: 此寄存器不能设置成其初始复位值(0x1F)

13.3.2 BEEP寄存器表和复位值

表25 蜂鸣器寄存器表

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	BEEP_CSR	BEEPSEL2	BEEPSEL1	BEEPEN	BEENDIV4	BEEPDIV3	BEEPDIV2	BEEPDIV1	BEEPDIV0

14 独立看门狗(IWDG)

14.1 介绍

独立看门狗模块可以用于解决处理器因为硬件或软件的故障所发生的错误。它由一个内部的128kHz的LSI阻容振荡器作为时钟源驱动，因此即使是主时钟失效时它仍然照常工作。

14.2 独立看门狗功能说明

图24是独立看门狗模块的功能框图。

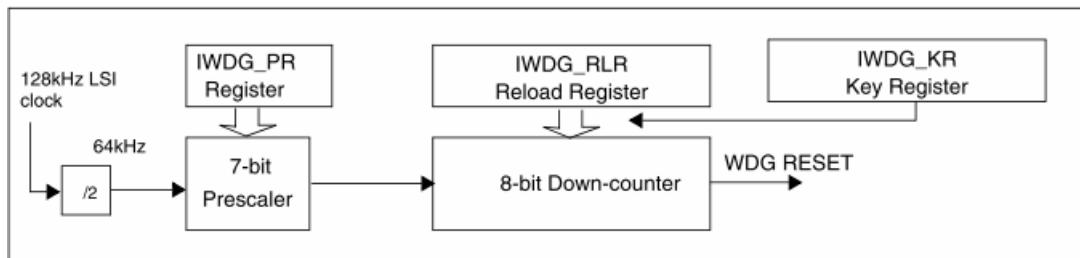
当在键寄存器(IWDG_KR)中写入数值0xCC后，独立看门狗就被启动了，计数器开始从它的复位值0xFF开始递减计数，当计数减到0x00时就会产生一个复位信号(WDG RESET)。

使用IWDG_PR和IWDG_RLR寄存器配置独立看门狗。IWDG_PR寄存器是用于选择驱动计数器时钟的预分频系数。每当KEY_REFRESH的数值(0xAA)写入到IWDG_KR寄存器时，独立看门狗将用IWDG_RLR的数值刷新计数器的内容，从而避免了产生看门狗的复位。

IWDG_PR和IWDG_RLR寄存器具有写保护功能，要修改它们前，需首先在IWDG_KR寄存器写入KEY_ACCESS代码(0x55)；在IWDG_KR写入0xAA将恢复写保护状态。

关于IWDG寄存器的详细内容，请参考 14.3。

图24 独立看门狗框图



硬件看门狗功能

如果在IWDG_HW选择字节中使能了硬件看门狗的功能，在芯片上电时看门狗的功能被自动开启，如果软件不能及时操作键寄存器，则在计数器达到0x00时产生复位。关于选择字节的内容请参考数据手册中的说明。

超时周期

超时周期由计数器数值和时钟预分频器决定，下表列出了它们的数值

表26 看门狗超时周期(假定计数器时钟为64kHz)

预分频系数	PR[2:0]	最短超时(RL[7:0]=0x00)	最长超时(RL[7:0]=0xFF)
/4	0	62.5 μs	15.90 ms
/8	1	125 μs	31.90 ms
/16	2	250 μs	63.70 ms
/32	3	500 μs	127 ms
/64	4	1.00 ms	255 ms
/128	5	2.00 ms	510 ms
/256	6	4.00 ms	1.02 s

14.3 IWDG寄存器

14.3.1 键寄存器(IWDG_KR)

地址偏移值: 0x00

复位值: 未定义

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

KEY[7:0]

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

位7:0	KEY[7:0]: 键值 软件必须在规定的时间内写入KEY_REFRESH数值, 否则当计数器数值达到0时, 看门狗会产生一个复位。 KEY_ENABLE 数值=0xCC 写入KEY_ENABLE数值将启动IWDG。 KEY_REFRESH 数值=0xAA 写入KEY_REFRESH数值将刷新IDDG。 KEY_ACCESS 数值=0x55 写入KEY_ACCESS数值将允许对受保护的IWDG_PR和IWDG_RLR寄存器的操作(见 14.2)。

14.3.2 预分频寄存器(IWDG_PR)

地址偏移值: 0x01

复位值: 0x00

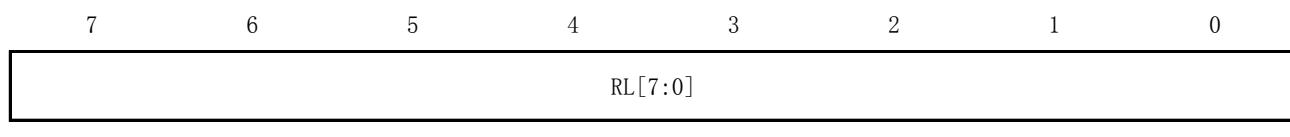
7	6	5	4	3	2	1	0
保留					PR[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw

位7:3	保留, 必须保持为0。
位2:0	<p>PR[2:0]: 预分频系数</p> <p>这些位是写保护的(见 14.2)。它们用于指定对计数器时钟分频的分频系数。</p> <p>000: 分频系数=4 001: 分频系数=8 010: 分频系数=16 011: 分频系数=32 100: 分频系数=64 101: 分频系数=128 110: 分频系数=256 111: 保留</p>

14.3.3 重装载寄存器(IWDG_RLR)

地址偏移值: 0x02

复位值: 0xFF



位7:0	RL[7:0]: 看门狗计数器重装载数值
	这些位是写保护的(见 14.2)。每次在IWDG_KR寄存器中写入0xAA时,这个寄存器中的内容会被传送到看门狗的计数器中,看门狗的计数器将重新从这个值开始计数。超时时间由这个数值和时钟的预分频系数决定,见表26。

14.3.4 IWDG寄存器映像和复位数值

表27 IWDG寄存器映像

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	IWDG_KR	KEY7	KEY6	KEY5	KEY4	KEY3	KEY2	KEY1	KEY0
	复位值	x	x	x	x	x	x	x	x
0x01	IWDG_PR	-	-	-	-	-	PR2	PR1	PRO
	复位值	0	0	0	0	0	0	0	0
0x02	IWDG_RLR	RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0
	复位值	1	1	1	1	1	1	1	1

15 窗口看门狗(WWDG)

15.1 介绍

窗口看门狗用于监测由于外部干扰或不可预知的逻辑条件所产生的软件错误，这样的软件错误通常会导致应用程序不按照预期的方式运行。除非程序在递减计数器的T6位变为0之前刷新递减计数器，看门狗电路将在一个预置的时间间隔后产生系统复位；如果在7位的递减计数器数值达到窗口寄存器数值之前刷新递减计数器，同样会产生系统复位。这就意味着只能在一个有限的时间窗口内刷新递减计数器。

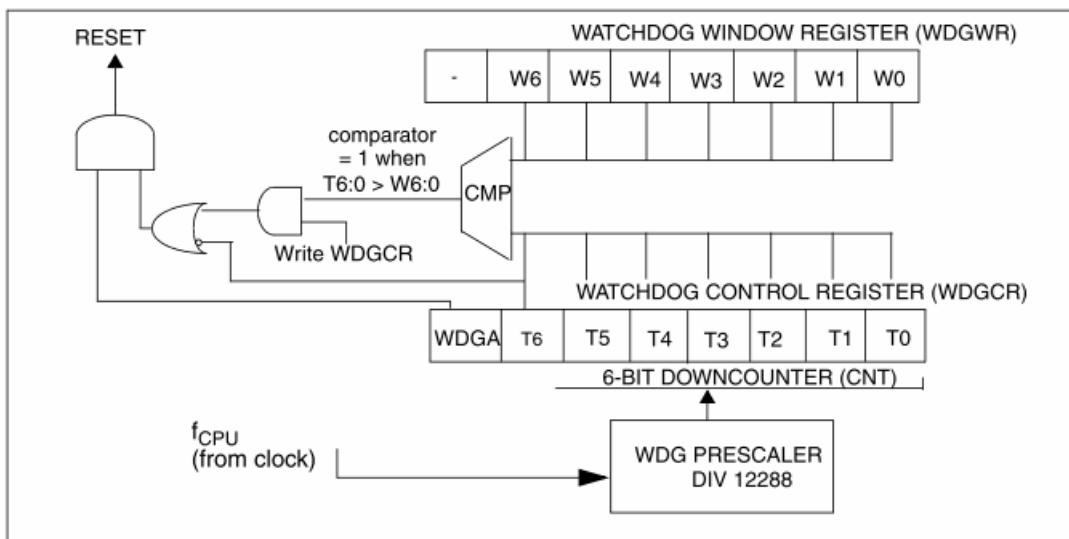
15.2 WWDG主要功能

- 可编程的自由运行递减计数器
- 有条件的复位
 - 如果开启了看门狗，当递减计数器的数值小于 0x40 时产生复位
 - 如果开启了看门狗，当在指定的时间窗口之外重加载递减计数器的数值(见 [图27](#)) 时产生复位
- 硬件或软件启动看门狗(由选择字节指定)
- 可在HALT指令时产生复位(由选择字节配置)

15.3 WWDG功能说明

如果开启了看门狗(设置了WDGA=1)，当7位的递减计数器(T[6:0]位)从0x40变为0x3F时(即T6变为0)，看门狗产生一个复位信号并把复位引脚拉低。如果软件刷新计数器时，计数器的数值大于窗口寄存器中的数值，同样会产生复位。

图25 窗口看门狗框图



在正常的操作期间，应用程序必须定期地写入WDGCR寄存器，以避免产生复位；这个写的操作必须在计数器的数值小于窗口寄存器的数值时进行。写入WDGCR寄存器的数值必须是介于0xFF和0xC0之间(见 [图26](#))：

- 开启看门狗：

如果(通过选择字节)选择了软件看门狗，在系统复位后看门狗处于关闭状态。设置WDGCR寄存器中的WDGA位将开启看门狗，随后在下次复位之前将不能关闭看门狗。

如果(通过选择字节)选择了硬件看门狗，看门狗将始终开启，而WDGA位将不起作用。

- 控制递减计数器：

递减计数器是自由运行计数器：即使未开启看门狗，它依然不断地递减计数。当开启看门狗时，必须设置T6位以避免立刻产生复位。

T[5:0]位中包含了看门狗产生复位前允许的时间延迟(见图26); 因为写入WDGCR寄存器时, 预分频器的状态是不可知的(见图27), 所以这个时间延迟介于一个最小和最大数值之间。

窗口寄存器(WDGWR)的数值是指定窗口的高限: 为防止复位, 必须在递减计数器的数值小于窗口寄存器的数值并大于0x3F时刷新递减计数器。图27描述了窗口看门狗操作过程。

注意: T6位可以用于产生一个软件复位(即设置WDGA位同时清除T6位)

- 在停止时产生看门狗复位

如果开启了看门狗, 并且选择了停止时产生看门狗复位的选项, 则执行HALT指令将产生复位。

15.4 在停止模式下使用WWDG

如果在选择字节中使能了停止模式下的看门狗(HALT指令不产生看门狗复位), 建议在执行HALT指令前先刷新看门狗计数器, 以避免在唤醒微控制器后立刻进入不希望的看门狗复位。

15.5 如何设置看门狗的超时

下图显示了看门狗计数器(CNT)中的6位数值, 与以毫秒为单位的超时时间的线性关系, 这个表可以在不考虑时序变化时作为一个快速的粗略计算参考, 如果需要更精确的计算, 请使用图27的公式。

警告: 每次写入WDGCR寄存器时, 首先要置T6位为'1', 以避免立刻产生看门狗复位。

图26 大约的超时时间

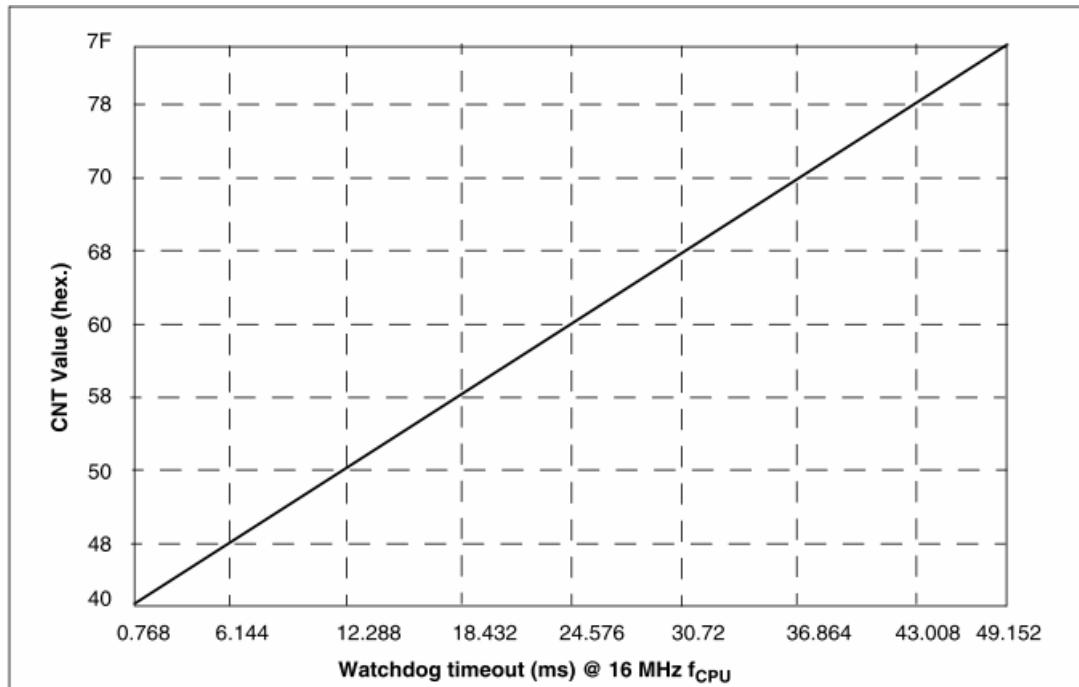
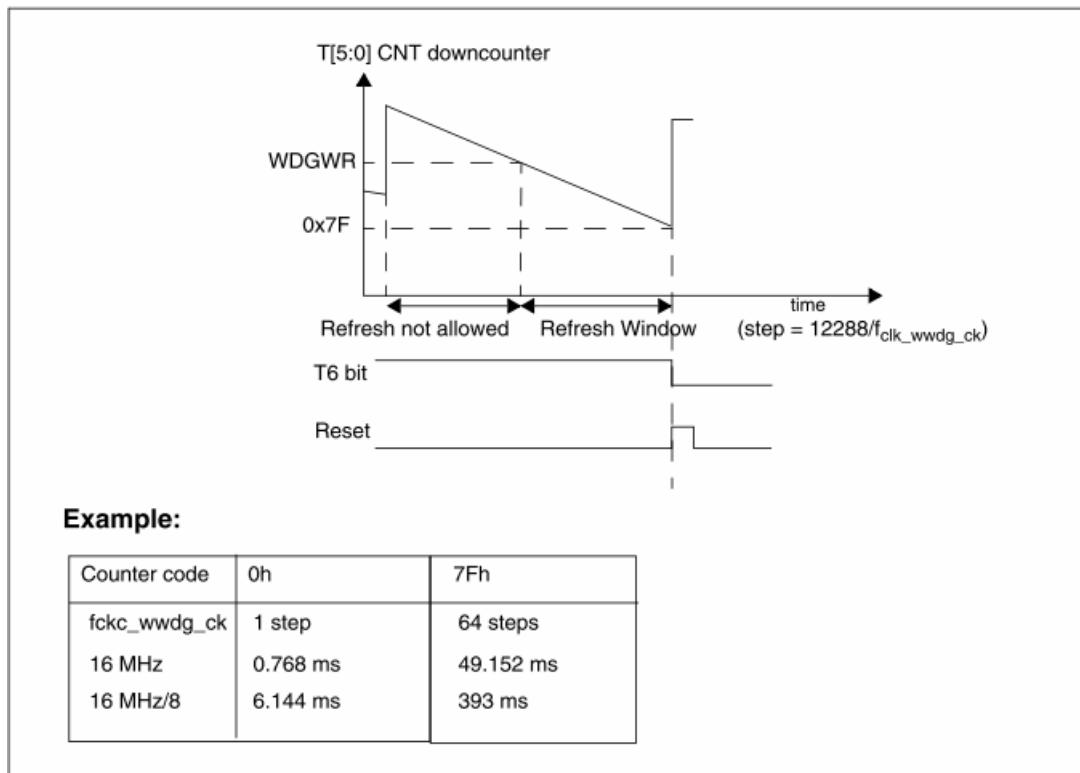


图27 窗口看门狗时序图



15.6 WWDG低功耗模式

表28 WWDG在低功耗模式下的影响

模式	说明	
等待(Wait)	看门狗不受影响：递减计数器照常工作	
停机(Halt)	选择字节中 WWDG_HALT	
	0	不产生看门狗复位。微控制器进入停止模式。递减计数器递减一次后停止计数，在微控制器收到一个外部中断或复位之前，它不会再产生看门狗复位。 如果收到了一个中断(参考中断映像表，查看停止模式下可以产生哪些中断)，在经过稳定延迟后看门狗将恢复计数。如果系统被复位，除非在选择字节中选择了硬件看门狗，否则看门狗将被关闭。请参考下面15.8的建议。
活跃停机 (Active Halt)	1	产生一个复位而不是进入停止模式。

15.7 硬件看门狗选项

如果在选择字节中选择了硬件看门狗选项，则看门狗始终开启，同时WDGCR寄存器中的WDGA将不起作用。请参考数据手册中有关选择字节的说明。

15.8 在停止模式下使用WWDG

如果开启了看门狗，则建议在停止模式下做如下操作。

在执行HALT指令前先刷新看门狗计数器，以避免在唤醒微控制器后立刻进入不希望的看门狗复位。

15.9 WWDG中断

无。

15.10 WWDG寄存器

15.10.1 控制寄存器(WWDG_CR)

地址偏移值: 0x00

复位值: 0x7F

7	6	5	4	3	2	1	0
WDGA	T6	T5	T4	T3	T2	T1	T0

位7	WDGA: 开启位 ⁽¹⁾ 该位由软件设置, 只能由硬件在复位后清除。当WDGA=1时, 看门狗可以产生复位。 0: 关闭看门狗 1: 开启看门狗
位6:0	T[6:0]: 7位计数器(MSB至LSB) 这些位包含看门狗计数器的数值, 每过(大约)12288个fckc_wwdg_ck周期递减一次。当它的内容从0x40变为0x3F(T6被清除)时, 则产生一个复位。

(1) 如果在选择字节中使能了硬件看门狗功能, 则此位不起作用。

15.10.2 窗口寄存器(WWDG_WR)

地址偏移值: 0x01

复位值: 0x7F

7	6	5	4	3	2	1	0
保留	W6	W5	W4	W3	W2	W1	W0
rw	rw	rw	rw	rw	rw	rw	rw
位7	保留						
位6:0	W[6:0]: 7位计数器(MSB至LSB) 这些位包含了窗口的数值，这是需要与递减计数器比较的数值。						

15.11 窗口看门狗寄存器映像和复位数值

表29 WWDG寄存器映像和复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	WWDG_CR	T7	T6	T5	T4	T3	T2	T1	T0
	复位值	0	1	1	1	1	1	1	1
0x01	WWDG_WR	-	W6	W5	W4	W3	W2	W1	W0
	复位值	0	1	1	1	1	1	1	1

16 定时器概述

STM8S提供三种类型的 TIM 定时器：高级控制型(TIM1)、通用型(TIM2/TIM3/TIM5)和基本型定时器(TIM4/TIM6)。它们虽有不同功能但都基于共同的架构。此共同的架构使得采用各个定时器来设计应用变得非常容易与方便(相同的寄存器映射，相同的基本功能)。

STM8S系列的定时器TIM1,TIM5和TIM6之间没有共享任何资源，但是它们可以按[TIM5/TIM6定时器的同步](#)中的描述来同步和连接。在拥有TIM1,TIM2,TIM3和TIM4定时器的STM8S系列产品中，定时器是没有连接在一起的。

本章仅给出不同定时器功能之间的比较和内部定时器信号名的词汇表。

下一章[16位高级控制定时器\(TIM1\)](#)包含所有定时工作模式的完整描述。其他的定时器章节相对简单，仅仅给出了每个定时器的框图和寄存器的描述。

表30 部分定时器参数

符号	参数	最小值	典型值	最大值	单位
$t_{w(ICAP)in}$	输入捕获脉冲时间	2			t_{MASTER}
$t_{res(TIM)}$	定时器精度	1			t_{MASTER}
Res_{TIM}	16位计数器定时器精度		16		Bit
	8位计数器定时器精度		8		Bit
$t_{COUNTER}$	当使用内部时钟时，计数器时钟周期		1		t_{MASTER}
t_{MAX_COUNT}	16位计数器最大可能计数值			65536	t_{MASTER}
	8位计数器最大可能计数值			256	t_{MASTER}

16.1 定时器功能比较

表31 定时器功能比较

定时器	计数器长度	计数方式	预分频因数	捕获/比较通道	互补输出	重复计数器	外部触发输入	外部刹车输入	定时器同步级联
TIM1 定时器1 (高级定时器)	16位	向上/向下	从1到65536任何整数	4	3	有	1	1	与定时器5/或定时器6
TIM2 定时器2 (通用定时器)			从1到32768的任何2的指数幂	3	无	无	0	0	无
TIM3 定时器2 (通用定时器)			从1到128的任何2的指数幂	2					
TIM4 定时器4 (基本定时器)	8位	向上	从1到128的任何2的指数幂	0	无	无	0	0	有
TIM5 定时器5 (通用定时器)	16位		从1到32768的任何2的指数幂	3					
TIM6 定时器2 (基本定时器)	8位		从1到128的任何2的指数幂	0					

16.2 定时器信号术语表

表32 内部定时器信号术语表

内部信号名	描述	相关框图
BI	刹车中断	图28 TIM1框图
CC1I, CC2I, CC3I, CC4I	捕获/比较中断	
CK_CNT	计数器时钟	图32 当ARPE=0(ARR不预装载), 预分频为2时的计数器更新。
CK_PSC	分频时钟	
CNT_EN	计数器使能	图42 TI2外部时钟连接例子
CNT_INIT	计数器初始化	
ETR	TIMx_ETR外部触发信号	图44 外部触发输入框图
ETRF	外触发滤波	

ETRP	外触发分频	
fMASTER	来自时钟控制器(CLK)的定时器外设时钟	图13 时钟树
ICi, IC1, IC2	输入捕获	图61 TIM1通道1的输入
ICiPS, IC1PS, IC2PS	输入捕获预分频	
MATCH1	比较匹配	图51 触发从模式/主模式的选择框图
OCi, OC1, OC2	定时器输出通道	图65 详细的带互补输出的输出模块框图(通道1)
OCiREF, OC1REF, OC2REF	输出比较参考信号	
TGI	触发中断	图40 时钟/触发控制器框图
TIi, TI1, TI2	定时器输入	图61 TIM1通道1的输入
TIiF, TI1F, TI2F	定时器输入滤波	
TI1_ED	定时器输入边缘检测	
TIiFPx, TI1FP1, TI1FP2, TI2FP1, TI2FP2	定时器输入滤波预分频	
TRC	触发捕获	
TRGI	时钟/触发器/从模式控制器的触发输入	图41 普通模式下的控制电路, fMASTER分频因子为1
UEV	更新事件	图32 当ARPE=0(ARR不预装载), 预分频为2时的计数器更新。
UIF	更新中断	

17 16位高级控制定时器(TIM1)

本章系统地描述了高级控制定时器的特性。

17.1 简介

TIM1由一个16位的自动装载计数器组成，它由一个可编程的预分频器驱动。

本章中使用i来代表1、2、3、4，分别对应于四个不同的捕获/比较通道。

高级控制定时器适用于许多不同的用途：

- 基本的定时
- 测量输入信号的脉冲宽度(输入捕获)
- 产生输出波形(输出比较，**PWM**和单脉冲模式)
- 对应与不同事件(捕获，比较，溢出，刹车，触发)的中断
- 与TIM5/TIM6或者外部信号(外部时钟，复位信号，触发和使能信号)同步

高级控制定时器广泛的适用于各种控制应用中，包括那些需要中间对齐模式**PWM**的应用，该模式支持互补输出和死区时间控制。

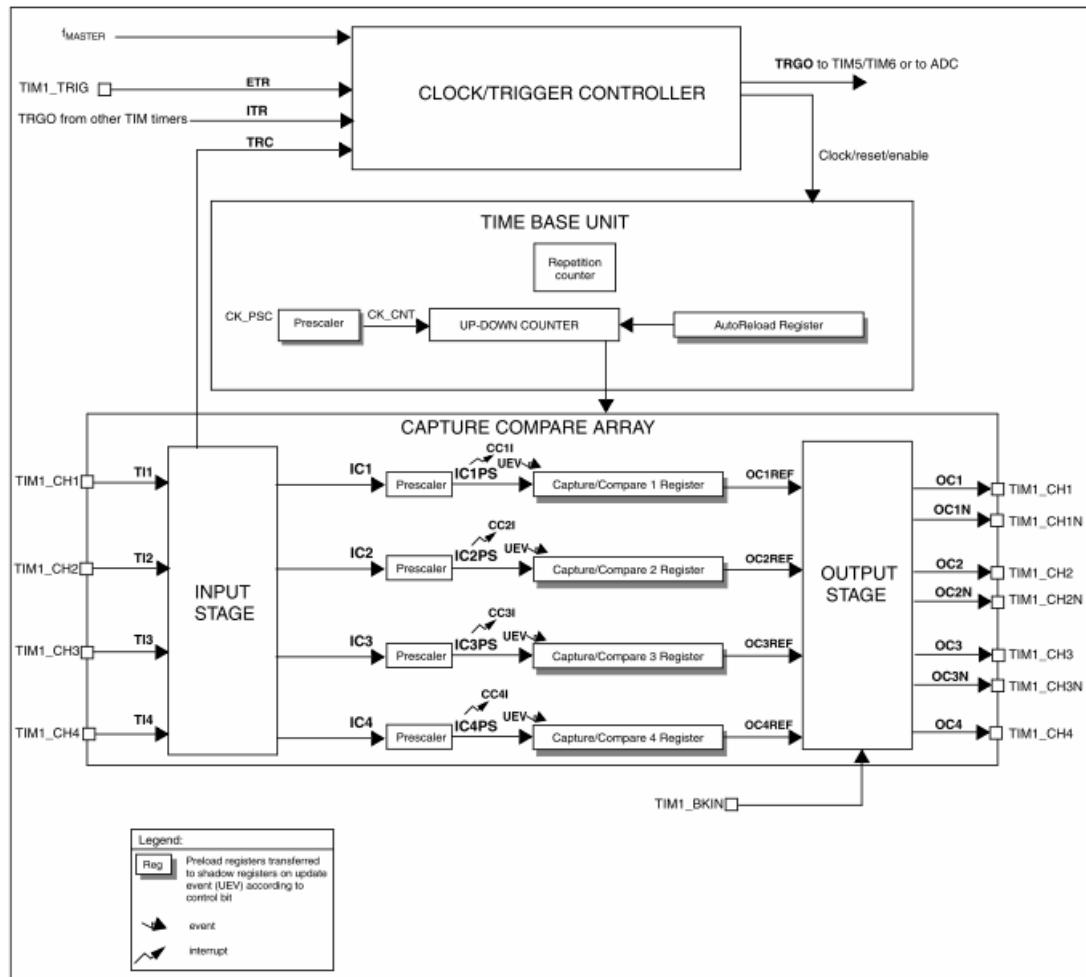
高级控制定时器的时钟源可以是内部时钟，也可以是外部的信号，可以通过配置寄存器来进行选择。

17.2 主要特性

TIM1的特性包括：

- 16位向上、向下、向上/下自动装载计数器
- 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
- 16位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为1~65535之间的任意数值
- 同步电路，用于使用外部信号控制定时器以及定时器互联(某些型号的芯片没有定时器互联功能)
- 多达4个独立通道可以配置成：
 - 输入捕获
 - 输出比较
 - **PWM**生成(边缘或中间对齐模式)
 - 六步**PWM**输出
 - 单脉冲模式输出
 - 三个支持带互补输出，并且死区时间可编程的通道
- 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
- 产生中断的事件包括：
 - 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

图28 TIM1框图



注:

Reg 根据控制位的设定，在U事件时传送预加载寄存器的内容至工作寄存器

事件

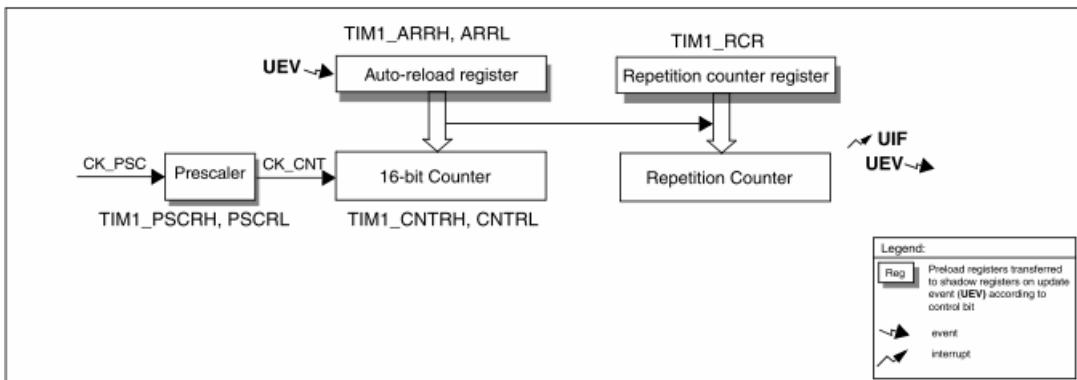
中断

17.3 时基单元

时基单元包含：

- 16位向上/向下计数器
- 16位自动重载寄存器
- 重复计数器
- 预分频器

图29 时基单元



16位计数器，预分频器，自动重载寄存器和重复计数器寄存器都可以通过软件进行读写操作。

自动重载寄存器由预装载寄存器和影子寄存器组成。

可在两种模式下写自动重载寄存器：

- **自动预装载已使能**(TIM1_CR1寄存器的ARPE位置位)。在此模式下，写入自动重载寄存器的数据将被保存在预装载寄存器中，并在下一个更新事件(UEV)时传送到影子寄存器。
- **自动预装载已禁止**(TIM1_CR1寄存器的ARPE位清除)。在此模式下，写入自动重载寄存器的数据将立即写入影子寄存器。

更新事件的产生条件：

- 计数器向上或向下溢出。
- 软件置位了TIM1_EGR寄存器的UG位。
- 时钟/触发控制器产生了触发事件。

在预装载使能时(**ARPE=1**)，如果发生了更新事件，预装载寄存器中的数值(**TIM1_ARR**)将写入影子寄存器中，并且**TIM1_PSCR**寄存器中的值将写入预分频器中。

置位**TIM1_CR1**寄存器的UDIS位将禁止更新事件(UEV)。

计数器由预分频器的输出CK_CNT驱动，而CK_CNT仅在IM1_CR1寄存器的计数器使能位(CEN)被置位时才有效。

注意：在使能了CEN位的一个时钟周期后，计数器才开始计数。

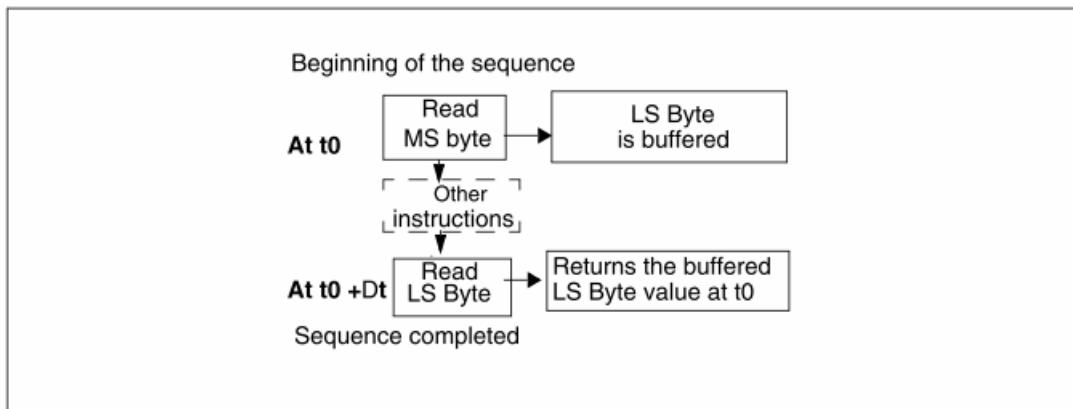
17.3.1 读写 16 位计数器

写计数器的操作没有缓存，并且可以在任何时候写**TIM1_CNTRH**和**TIM1_CNTRL**寄存器，因此我们建议不要在计数器运行时写入新的数值，以免写入了错误的数值。

读计数器的操作带有8位的缓存。在用户读了高位(MS)字节后，低位(LS)字节将被自动缓存，缓存的数据在16位的读操作完成之前不会有变化，[图30](#)解释了这一过程。

注意：不要使用LDW指令来读取16位计数器的值，因为此指令先读低位(LS)字节，这样读出的数值是错误的。

图30 读16位计数器的过程(TIM1_CNTR)



17.3.2 16位TIM1_ARR寄存器的写操作

预装载寄存器中的值将写入16位的TIM1_ARR寄存器中，此操作由两条指令完成，每条指令写入1个字节，高位(MS)字节是先写入的。

影子寄存器在高位(MS)字节写入时被锁定，并保持到低位(LS)字节写完。不要使用LDW指令，因为此指令先写低位(LS)字节，这将导致写入的数值错误。

17.3.3 预分频器

预分频器的实现：

- TIM1的预分频器基于一个由16位寄存器(TIM1_PSCR)控制的16位计数器。由于这个控制寄存器带有缓冲器，因此它能够在运行时被改变。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。

计数器的频率可以由下式计算：

$$f_{CK_CNT} = f_{CK_PSC} / (PSCR[15:0] + 1)$$

预分频器的值由预装载寄存器写入，保存了当前使用值的影子寄存器在低位(LS)写入时被载入。

需两次单独的写操作来写16位寄存器，高位(MS)先写。不要使用先写低位(LS)的LDW指令。

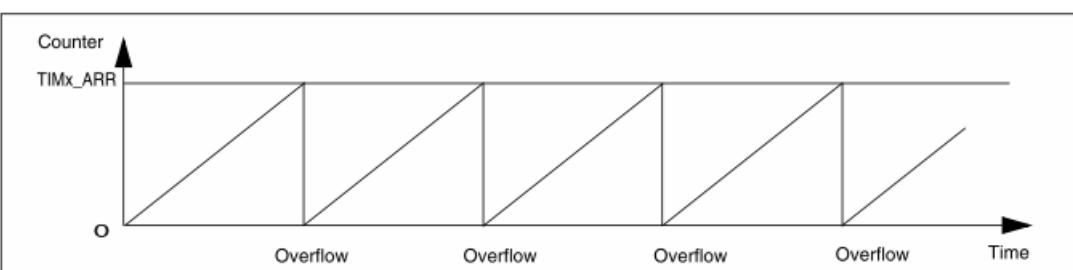
新的预分频器的值在下一次更新事件到来时被采用。

对TIM1_PSCR寄存器的读操作通过预装载寄存器完成，因此不需要特别的关注。

17.3.4 向上计数模式

在向上计数模式中，计数器从0计数到用户定义的比较值(TIMx_ARR寄存器的值)，然后重新从0开始计数并产生一个计数器溢出事件，同时，如果TIM1_CR1寄存器的UDIS位是0，将会产生一个更新事件(UEV)。[O](#)描述了向上计数模式。

图31 向上计数模式的计数器



置位TIMx_EGR寄存器的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。

使用软件置位TIMx_CR1寄存器的UDIS位，可以禁止更新事件，这样可以避免在更新预装载寄存器时更新影子寄存器。在UDIS位被清除之前，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寄存器的UIF位)：

自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

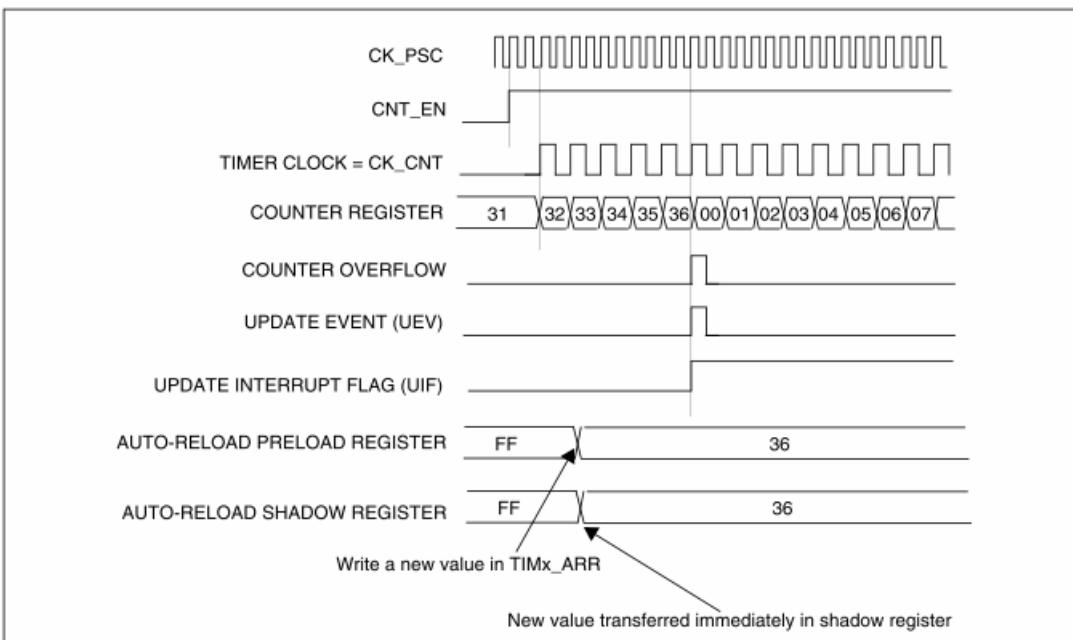
预分频器的缓存器被置入预装载寄存器的值(TIMx_PSC寄存器的内容)。

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

[图32](#)的预分频为2，因此计数器的时钟(CK_CNT)频率是预分频时钟(CK_PSC)频率的一半。

[图32](#)禁止了自动装载功能(ARPE=0)，所以在计数器达到0x36时，计数器溢出，影子寄存器立刻被更新，同时产生一个更新事件。

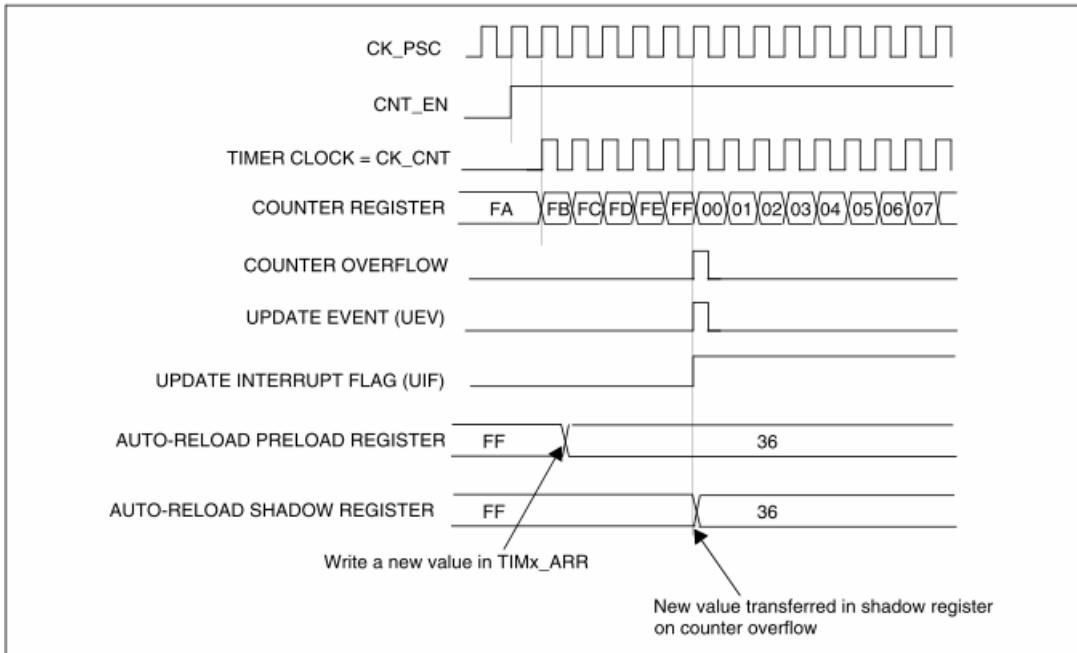
图32 当ARPE=0(ARR不预装载)，预分频为2时的计数器更新。



[图33](#)的预分频为1，因此CK_CNT的频率与CK_PSC一致。

[图33](#)使能了自动重载(ARPE=1)，所以在计数器达到0xFF产生溢出。0x36将在溢出时被写入，同时产生一个更新事件。

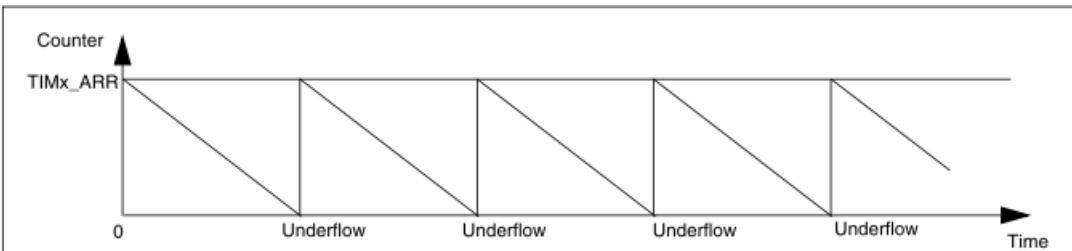
图33 ARPE=1(TIM1_ARR预装载)时的计数器更新



17.3.5 向下计数模式

在向下模式中，计数器从自动装载的值(TIMx_ARR寄存器的值)开始向下计数到0，然后再从自动装载的值重新开始计数，并产生一个计数器向下溢出事件。如果TIM1_CR1寄存器的UDIS位被清除，还会产生一个更新事件(UEV)。图34描述了向下计数模式的计数器。

图34 向下计数模式的计数器



置位TIMx_EGR寄存器的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。

置位TIMx_CR1寄存器的UDIS位可以禁止UEV事件。这样可以避免在更新预装载寄存器时更新影子寄存器。因此UDIS位清除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)，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存入预装载的值(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)。要注意自动装载寄存器在计数器重载入之前被更新，因此下一个周期才是预期的值。

以下是一些当TIMx_ARR=0x36时，计数器在不同时钟频率下的图表。

下图描述了在向下计数模式下，预装载不使能时新的数值在下个周期时被写入。

图35 ARPE=0(ARR不预装载)，预分频为2时的计数器更新

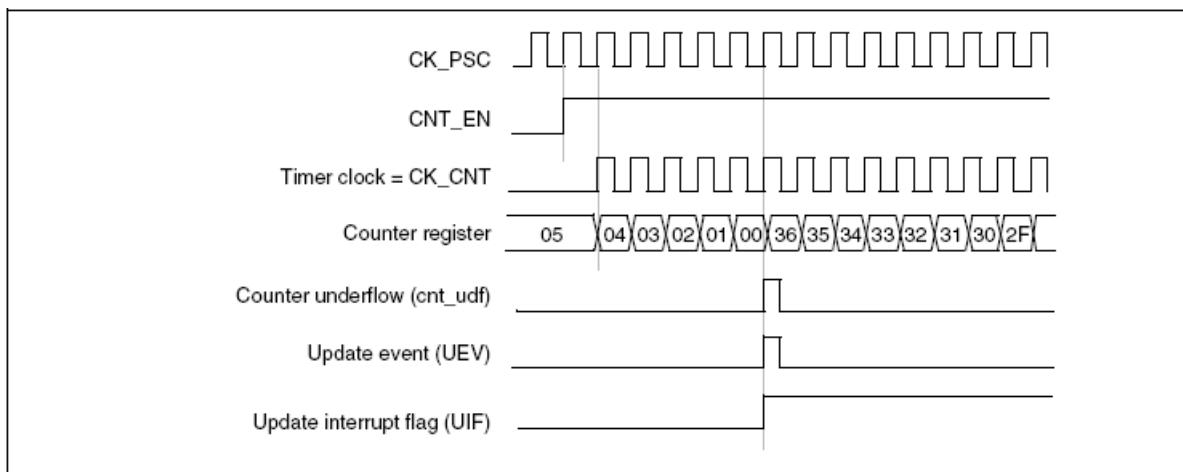
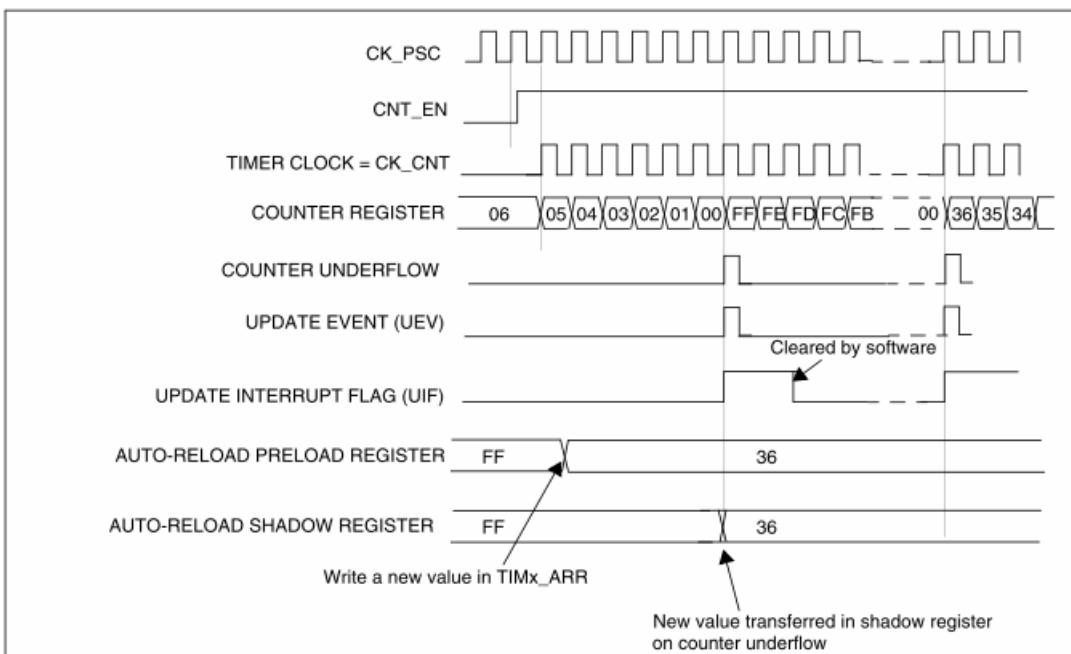


图36 ARPE=1(ARR预装载)，预分频为1时的计数器更新



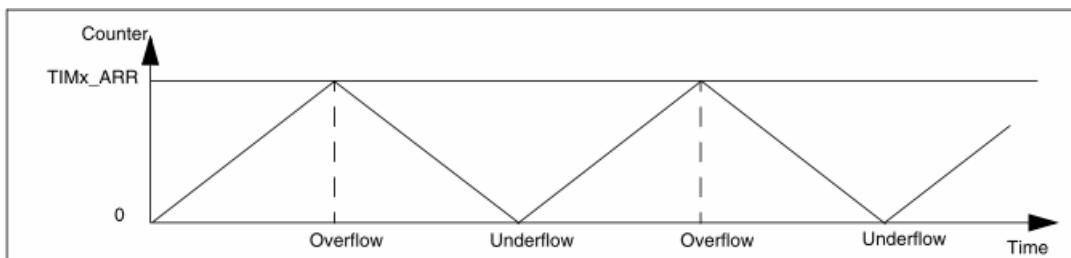
17.3.6 中央对齐模式(向上/向下计数)

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0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。

下图给出一个中央对齐模式的例子。

图37 中央对齐模式的计数器



如果定时器带有重复计数器(如TIM1)，在重复了指定次数(TIM1_RCR的值)的向上和向下溢出之后会产生更新事件(UEV)。否则每一次的向上向下溢出都会产生更新事件。

置位TIMx_EGR寄存器的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。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。这样可以避免在更新预装载寄存器时更新影子寄存器。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。如果定时器带有重复计数器，由于重复寄存器没有双重的缓冲，新的重复数值将立刻生效，因此在修改时需要小心。

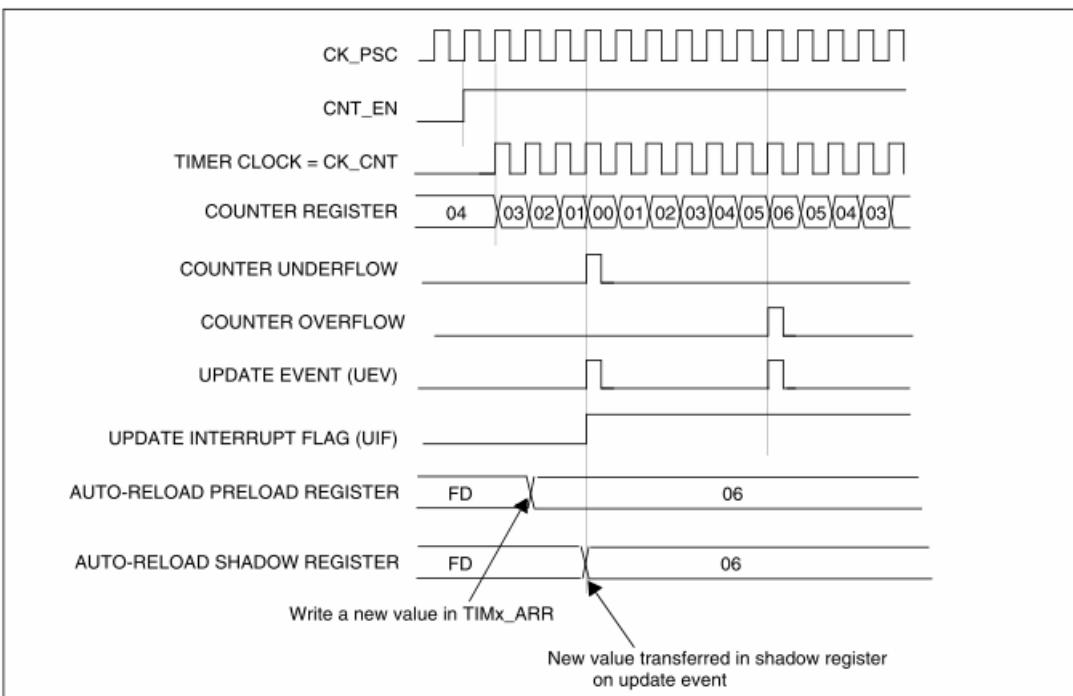
此外，如果设置了TIMx_CR1寄存器中的URS位(选择更新请求)，设置UG位将产生一个更新事件UEV但不设置UIF标志(因此不产生中断)，这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

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)。要注意到如果因为计数器溢出而产生更新，自动重装载寄存器将在计数器重载入之前被更新，因此下一个周期才是预期的值(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38 计数器时序图，内部时钟分频因子为1，TIMx_ARR=0x6，ARPE=1



使用中央对齐模式的提示：

- 启动中央对齐模式时，计数器将按照原有的向上/向下的配置计数。也就是说TIM1_CR1寄存器中的DIR位将决定计数器是向上还是向下计数。此外，软件不能同时修改DIR位和CMS位的值。
- 不推荐在中央对齐模式下，计数器正在计数时写计数器的值，这将导致不能预料的后果。具体的说：
 - 向计数器写入了比自动装载值更大的数值时(TIM1_CNT>TIM1_ARR)，但计数器的计数方向不发生改变。例如计数器已经向上溢出，但计数器仍然向上计数。
 - 向计数器写入了0或者TIM1_ARR的值，但更新事件不发生。
- 安全使用中央对齐模式的计数器的方法是在启动计数器之前先用软件(置位TIM1_EGR寄存器的UG位)产生一个更新事件，并且不在计数器计数时修改计数器的值。

17.3.7 重复计数器

17.3时基单元解释了计数器向上/向下溢出时更新事件(UEV)是如何产生的，然而事实上它只能在重复计数器的值达到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：

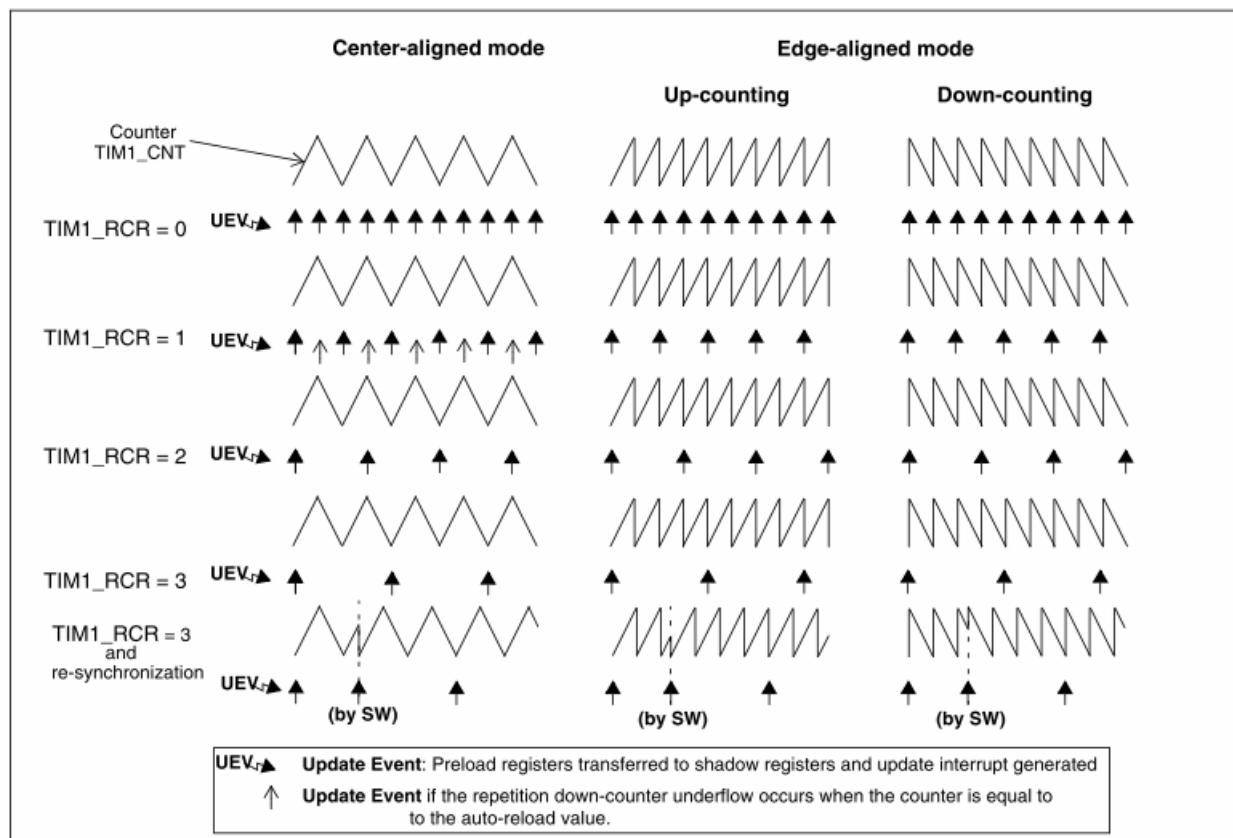
- 向上计数模式下每次计数器向上溢出时
- 向下计数模式下每次计数器向下溢出时
- 中央对齐模式下每次上溢和每次下溢时。

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 $2t_{CK_PSC}$ 。

重复计数器是自动加载的，重复速率由TIMx_RCR寄存器的值定义(参考图39)。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。

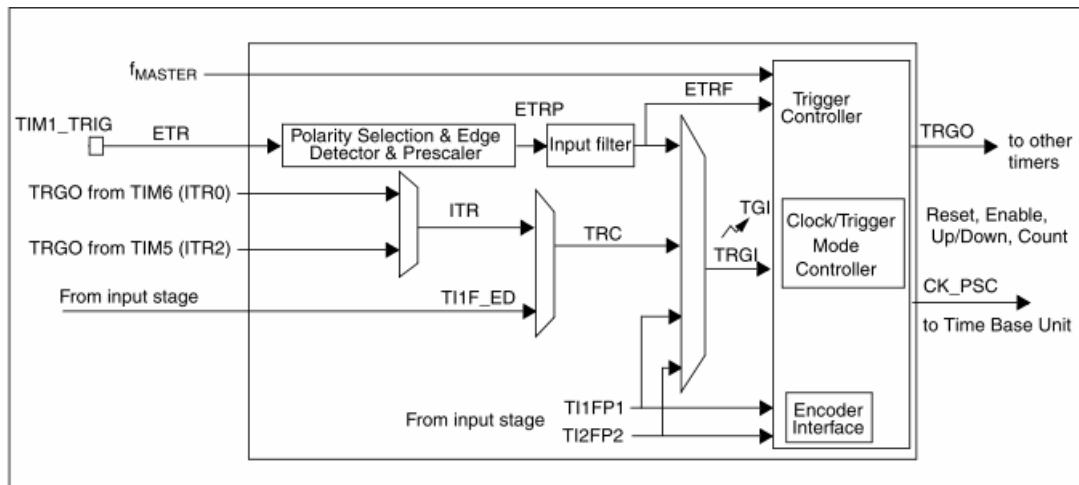
图39 不同模式下更新速率的例子，及TIMx_RCR的寄存器设置



17.4 时钟/触发控制器

时钟/触发控制器允许用户选择计数器的时钟源，输入触发信号和输出信号，框图如图40所示。

图40 时钟/触发控制器框图



17.4.1 预分频时钟(CK_PSC)

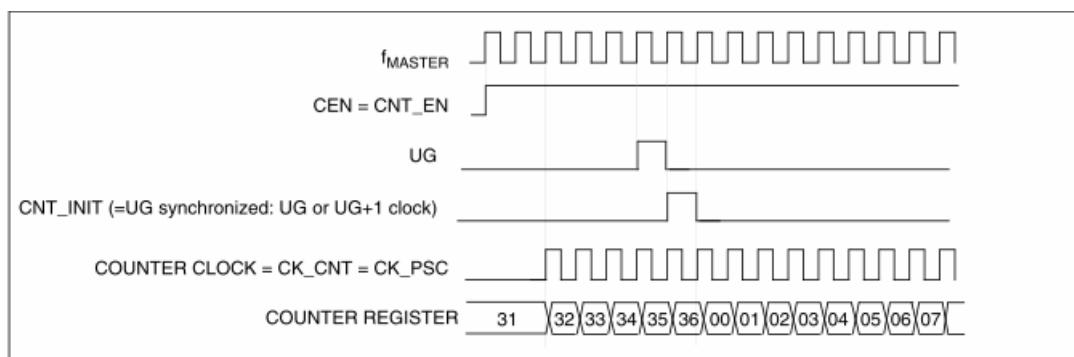
时基单元的预分频时钟(CK_PSC)可以由以下源提供:

- 内部时钟(f_{MASTER})
- 外部时钟模式1: 外部时钟输入(TI_x)
- 外部时钟模式2: 外部触发输入 ETR
- 内部触发输入($ITRx$): 使用一个定时器做为另一个定时器的预分频时钟。更多信息请参考图52的例子。

17.4.2 内部时钟源(f_{MASTER})

如果同时禁止了触发模式控制器和外部触发输入(TIM1_SMCR寄存器的SMS=000, TIM1_ETR寄存器的ECE=0), 则CEN、DIR和UG位是实际上的控制位, 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提供。

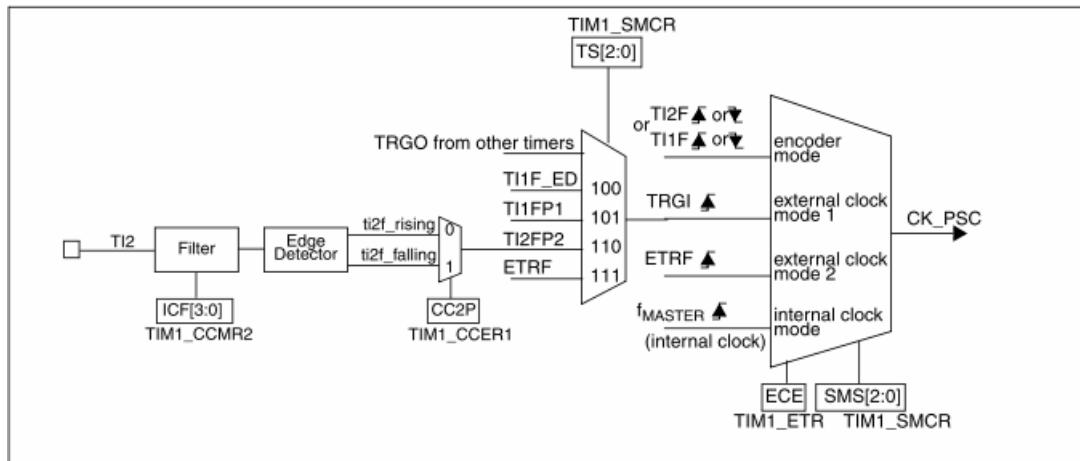
下图描述了控制电路和向上计数器在普通模式下, 不带预分频器时的操作。

图41 普通模式下的控制电路, f_{MASTER} 分频因子为1

17.4.3 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42 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：

4. 配置TIM1_CCMR2寄存器的CC2S=01，使用通道2检测TI2输入的上升沿
5. 配置TIM1_CCMR2寄存器的IC2F[3:0]位，选择输入滤波器带宽(如果不需滤波器，保持IC2F=0000)

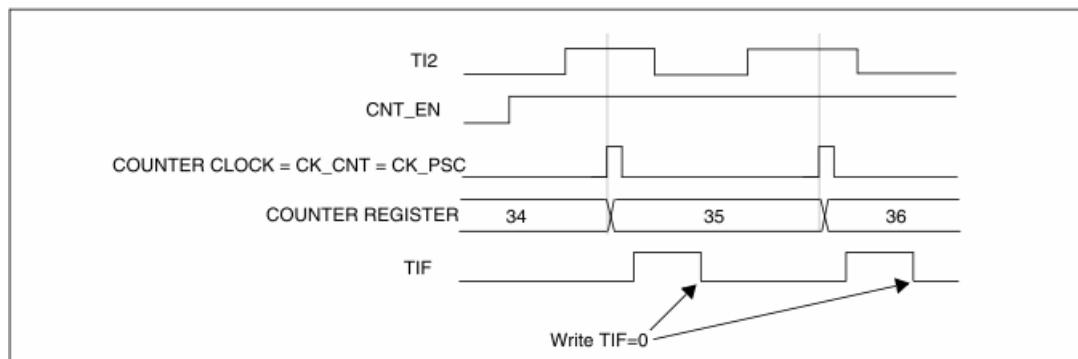
注：捕获预分频器不用作触发，所以不需要对它进行配置，同样也不需要配置TI2S位，他们仅用来选择输入捕获源。

6. 配置TIM1_CCER1寄存器的CC2P=0，选定上升沿极性
7. 配置TIM1_SMCR寄存器的SMS=111，配置计数器使用外部时钟模式1
8. 配置TIM1_SMCR寄存器的TS=110，选定TI2作为输入源
9. 设置TIM1_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，且触发标识位(TIM1_SR1寄存器的TIF位)被置1，如果使能了中断(在TIM1_IER寄存器中配置)则会产生中断请求。

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43 外部时钟模式1下的控制电路

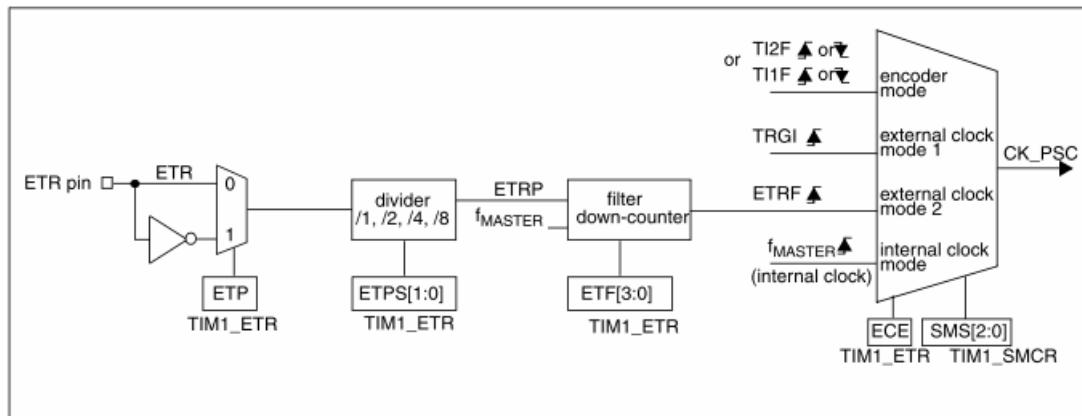


17.4.4 外部时钟源模式 2

计数器能够在外部触发输入ETR信号的每一个上升沿或下降沿计数。将TIM1_ETR寄存器的ECE位写1，即可选定此模式。

[图44](#)描述了外部触发输入的总体框图。

图44 外部触发输入框图



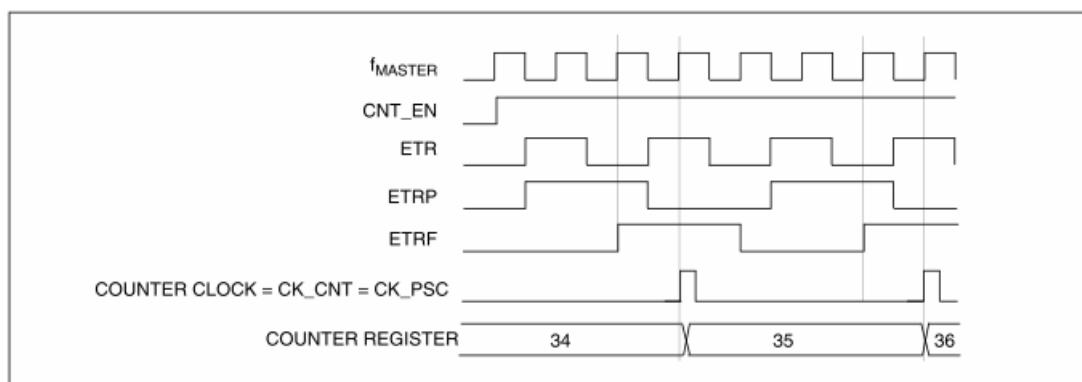
例如, 要配置计数器在ETR信号的每2个上升沿时向上计数一次, 需使用下列步骤:

1. 本例中不需要滤波器, 配置TIM1_ETR寄存器的ETF[3:0]=0000
2. 设置预分频器, 配置TIM1_ETR寄存器的ETPS[1:0]=01
3. 选择ETR的上升沿检测, 配置TIM1_ETR寄存器的ETP=0
4. 开启外部时钟模式2, 配置TIM1_ETR寄存器中的ECE=1
5. 启动计数器, 写TIM1_CR1寄存器的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45 外部时钟模式2下的控制电路



17.4.5 触发同步

计数器允许四种触发输入(请参考表32内部定时器信号术语表)

- ETR
- TI1
- TI2
- 来自TIM5/TIM6的TRGO

TIM1的计数器使用三种模式与外部的触发信号同步: 标准触发模式, 复位触发模式和门控触发模式。

标准触发模式

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:

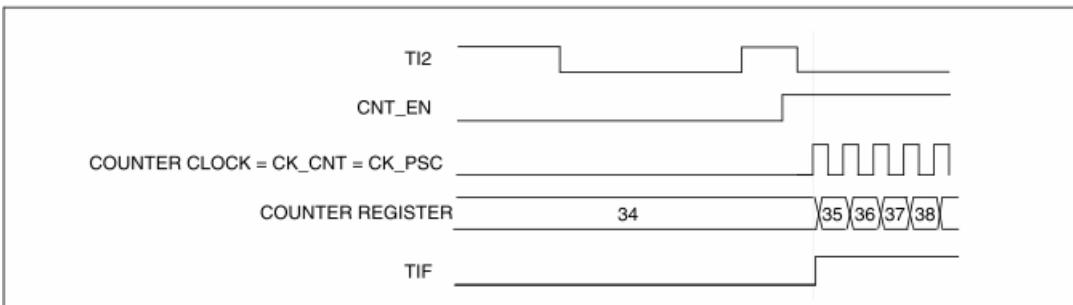
- 配置通道2检测TI2的上升沿。配置输入滤波器带宽(本例中, 不需要任何滤波器, 保持IC2F=0000)。触发操作中不使用捕获预分频器, 不需要配置。TI2S位仅用于选择输入捕获源, 也不需要配置。配置TIM1_CCER1寄存器的CC2P=0, 选择上升沿做为触发条件。

- 配置TIM1_SMCR寄存器的SMS=110, 选择计数器为触发模式; 配置TIM1_SMCR寄存器的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, 同时置位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46 标准触发模式的控制电路



复位触发模式

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TIM1_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(TIM1_ARR, TIM1_CCRx)都被更新了。

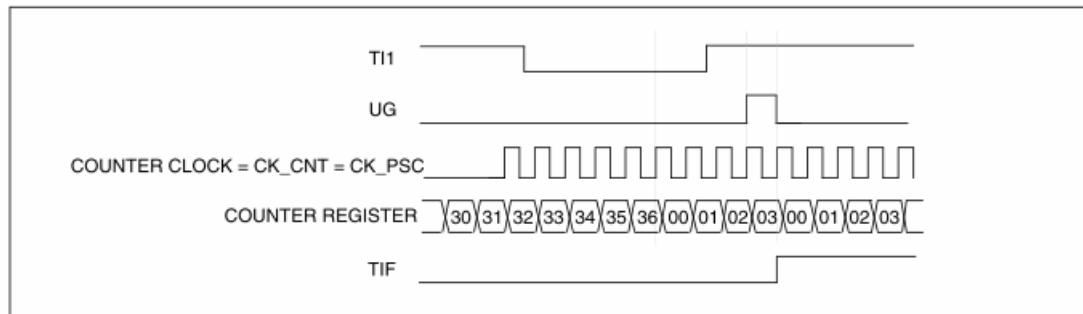
在以下的例子中, TI1输入端的上升沿导致向上计数器被清零:

- 配置通道1用于检测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位仅用于选择输入捕获源, 也不需要配置。配置TIM1_CCER1寄存器的CC1P=0来选择极性(只检测上升沿)。
- 配置TIM1_SMCR寄存器的SMS=100, 选择定时器为复位触发模式; 配置TIM1_SMCR寄存器的TS=101, 选择TI1作为输入源。
- 配置TIM1_CR1寄存器的CEN=1, 启动计数器。

计数器开始依据内部时钟计数, 然后正常计数直到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(TIM1_SR1寄存器的TIF位)被置位, 如果使能了中断(TIM1_IER寄存器的TIE位), 则产生一个中断请求。

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47 复位触发模式下的控制电路



门控触发模式

计数器由选中的输入端信号的电平使能。

在如下的例子中, 计数器只在TI1为低时向上计数:

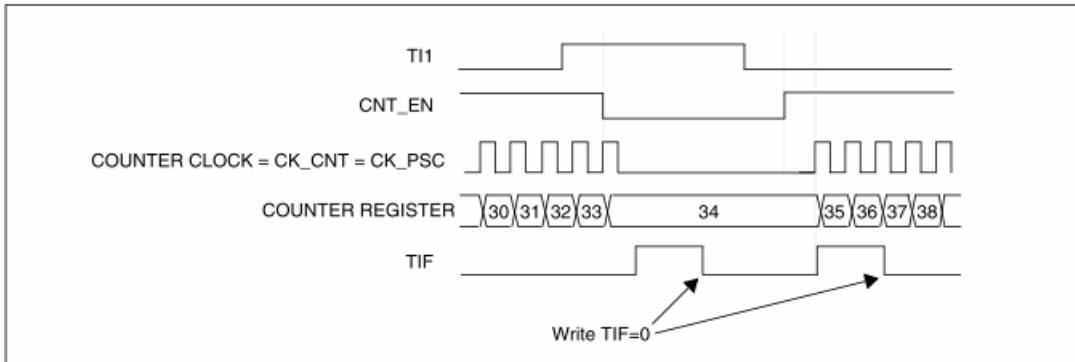
1. 配置通道1用于检测TI1上的低电平。配置输入滤波器带宽(本例中, 不需要滤波, 所以保持IC1F=0000)。触发操作中不使用捕获预分频器, 所以不需要配置。CC1S位用于选择输入捕获源, 也不需要配置。配置TIM1_CCER1寄存器的CC1P=1来确定极性(只检测低电平)。

2. 配置TIM1_SMCR寄存器的SMS=101，选择定时器为门控触发模式；配置TIM1_SMCR寄存器中TS=101，选择TI1作为输入源。
3. 配置TIM1_CR1寄存器的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TIF标志位都会被置位。

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48 门控触发模式下的控制电路



外部时钟模式2 + 触发模式

外部时钟模式2可以与另一个输入信号的触发模式一起使用。这时，ETR信号被用作外部时钟的输入，另一个输入信号可用作触发模式(支持标准触发模式，复位触发模式和门控触发模式)。请注意不能把ETR配置成TRGI(通过TIM1_SMCR寄存器的TS位)。

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：

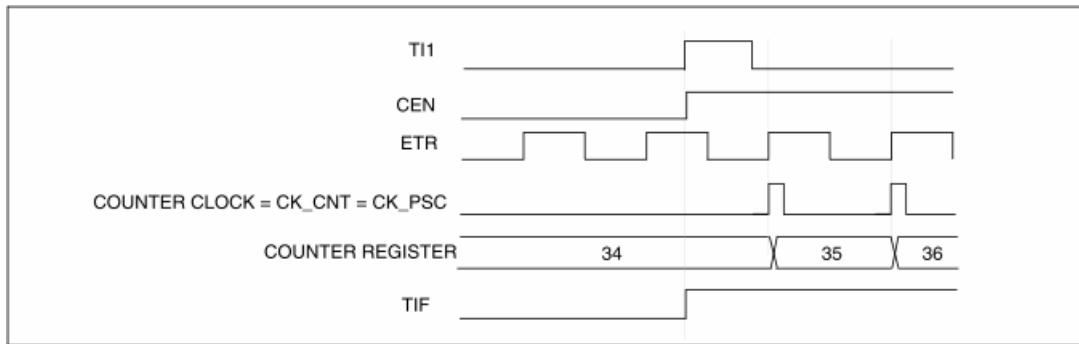
- 通过TIM1_ETR寄存器配置外部触发输入电路。在这个例子中，由于不使用滤波，因此ETF = 0000。配置ETPS=00禁止预分频，配置ETP=0监测ETR信号的上升沿，配置ECE=1使能外部时钟模式2。
- 使用通道1监测TI1的上升沿。配置输入滤波(由于本例不使用滤波，因此配置IC1F=0000)。由于触发操作不使用预分频，所以不配置预分频器，CC1S位仅用于选择输入捕获源，因此也不需要配置。配置TIM1_CCER1寄存器的CC1P=0来选择上升沿触发。
- 配置TIM1_SMCR寄存器的SMS=110来选择定时器为触发模式。配置TIM1_SMCR寄存器的TS=101来选择TI1作为输入源。

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。

TI1信号的上升沿和计数器实际时钟之间的延时取决于TI1输入端的重同步电路。

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

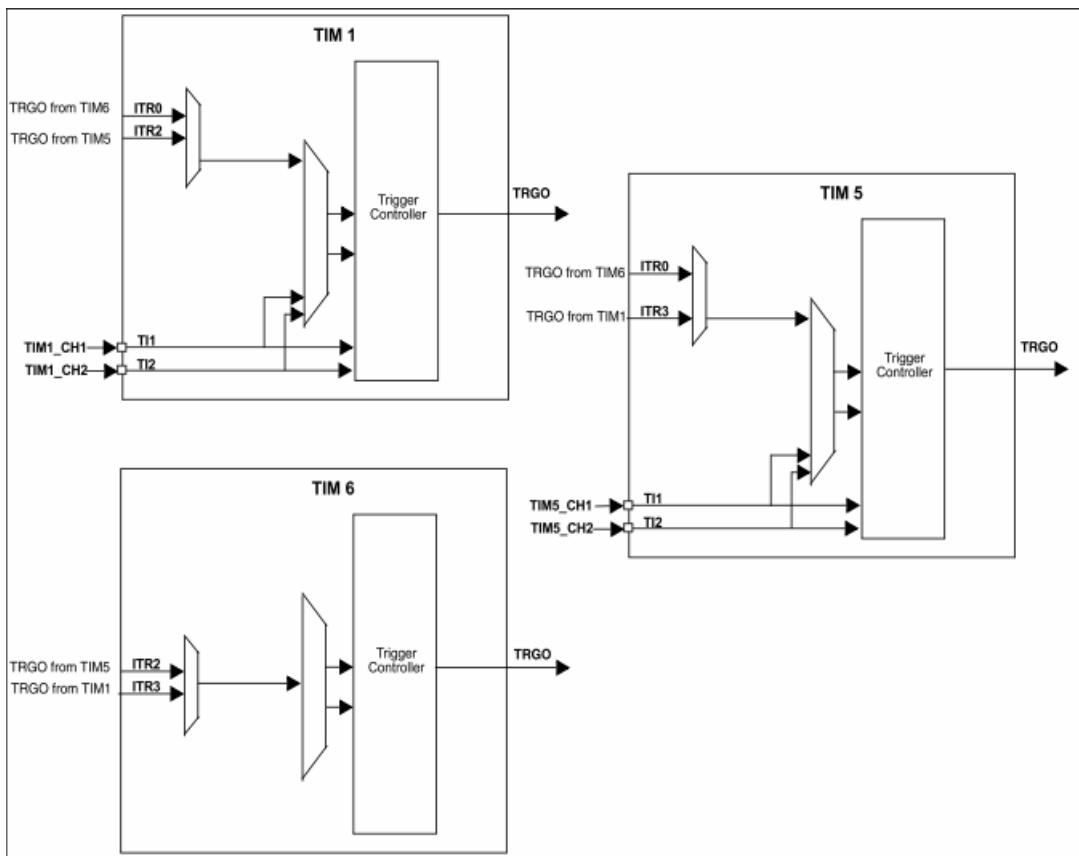
图49 外部时钟模式2+触发模式下的控制电路



17.4.6 与TIM5/TIM6 定时器的同步

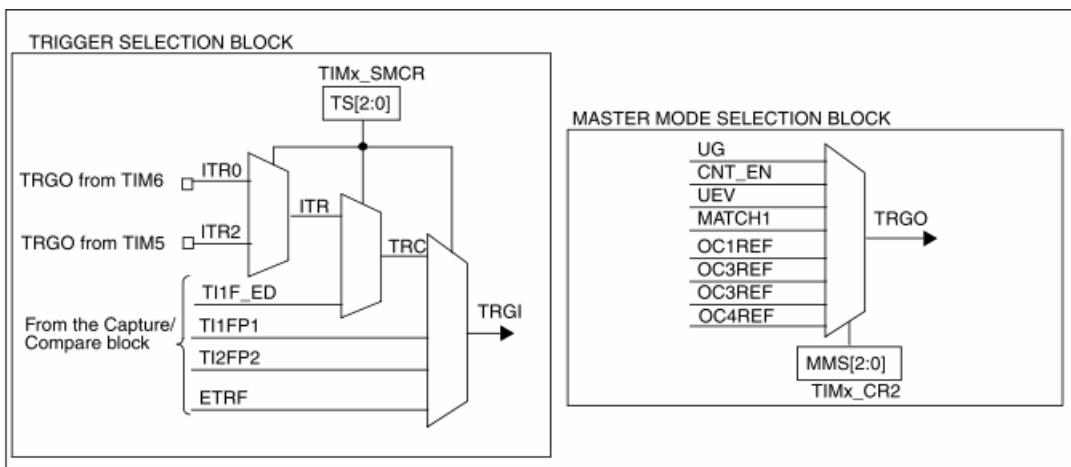
在某些型号的芯片中，定时器在内部互相联结，用于定时器的同步或链接。当某个定时器配置成主模式时，可以输出触发信号(TRGO)到那些配置为从模式的定时器来完成复位，启动，停止的操作，或者作为那些定时器的驱动时钟。

图50 定时器链接系统的实现图例



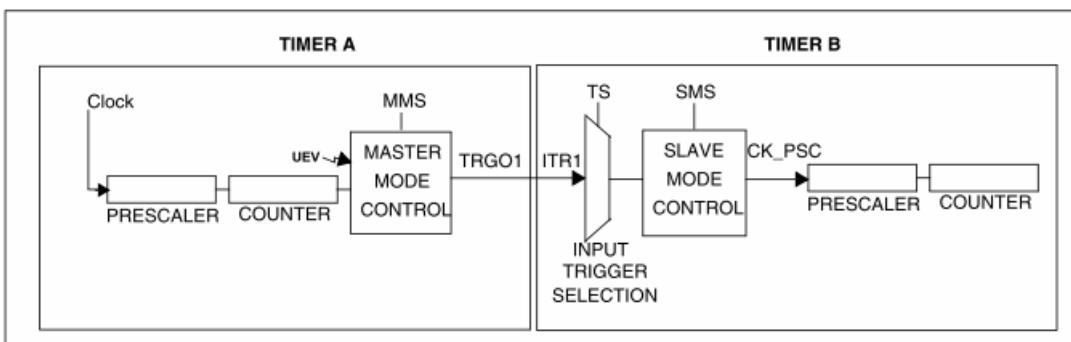
下图描述了触发从模式选择和主模式选择的框图

图51 触发从模式/主模式的选择框图



使用一个定时器作为另一个定时器的预分频时钟

图52 主/触发从模式的定时器例子



例如，用户可以配置定时器A作为定时器B的预分频时钟，需进行如下配置：

1. 配置定时器A为主模式，使得在每个更新事件(UEV)时输出周期性的触发信号。配置TIM1_CR2寄存器的MMS=010，使每个更新事件时TRGO1能输出一个上升沿。
2. 定时器A输出的TRGO1信号链接到定时器B。定时器B需要配置成触发从模式，使用ITR1作为输入触发信号。以上操作可以通过配置TIM1_SMCR寄存器的TS=001实现。
3. 配置TIM1_SMCR寄存器的SMS=111将时钟/触发控制器设置为外部时钟模式1。此操作将使定时器A输出的周期性触发信号(由定时器A的溢出产生)的上升沿驱动定时器B的时钟。
4. 最后，置位两个定时器的CEN位(TIM1_CR1寄存器中)，使能两个定时器。

注意：如果选择OCi作为定时器A输出的触发信号(MMS=1xx)，它的上升沿将驱动定时器B的时钟。

使用一个定时器使能另一个定时器

在本例中，我们用定时器A的比较输出使能定时器B，定时器的连接请参考图52。定时器B仅在定时器A的OC1REF信号为高时按照自己的驱动时钟计数。两个定时器都使用4分频的f_MASTER为时钟($f_{CK_CNT} = f_{MASTER}/4$)。

1. 配置定时器A为主模式，将比较输出信号(OC1REF)作为触发信号输出。(配置TIM1_CR2寄存器的MMS=100)。
2. 配置定时器A的OC1REF信号的波形(TIM1_CCMR1寄存器)。
3. 配置定时器B把定时器A的输出作为自己的触发输入信号(配置TIM1_SMCR寄存器的TS=001)。
4. 配置定时器B为门控触发模式(配置TIM1_SMCR寄存器的SMS=101)。
5. 置位CEN位(TIM1_CR1寄存器)，使能定时器B。

6. 置位CEN位(TIM1_CR1寄存器), 使能定时器A。

注意: 两个计数器的时钟并不同步, 但仅影响定时器B的使能信号。

图53 定时器A的输出门控触发定时器B

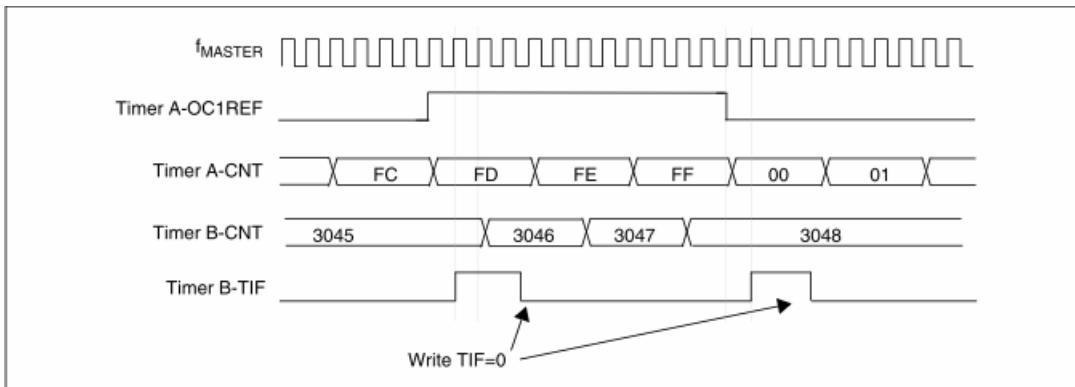
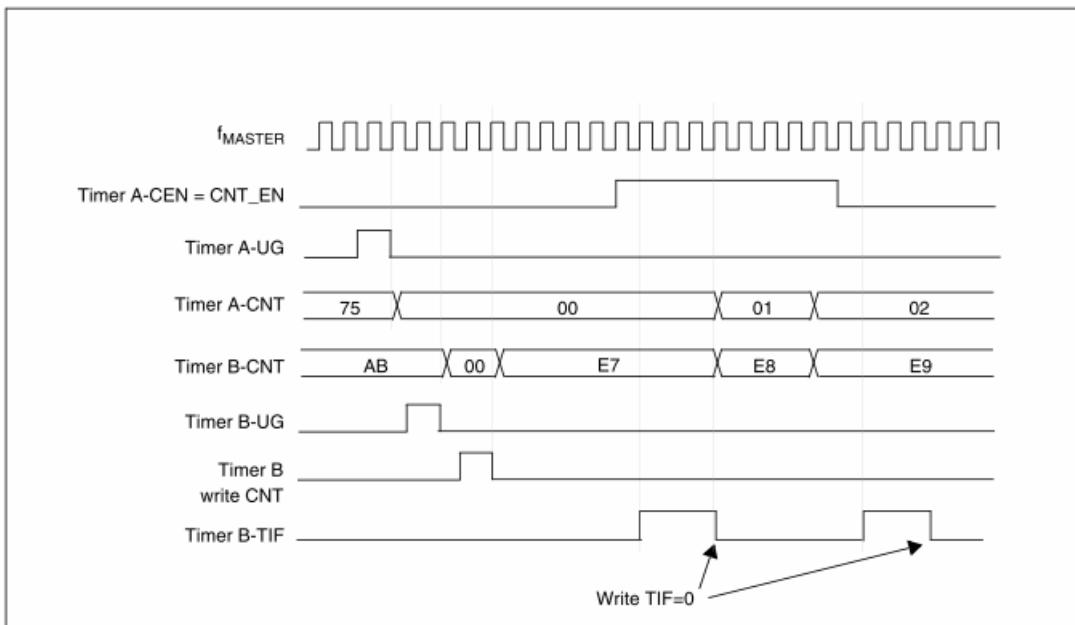


图53的例中, 定时器B的计数器和预分频器都没有在启动前初始化, 所以都是从现有值开始计数的。如果在启动定时器A之前复位两个定时器, 用户就可以写入期望的数值到定时器的计数器, 使之从指定值开始计数。对定时器的复位操作可以通过软件写TIM1_EGR寄存器的UG位实现。

在下一个例子中, 我们使定时器A和定时器B同步。定时器A为主模式并从0启动计数。定时器B为触发从模式, 并从0xE7启动计数。两个定时器采用相同的分频系数。当清除TIM1_CR1寄存器的CEN位时, 定时器A被禁止, 同时定时器B停止计数。

1. 配置定时器A为主模式, 将比较输出信号(OC1REF)作为触发信号输出。(配置TIM1_CR2寄存器的MMS=100)。
2. 配置定时器A的OC1REF信号的波形(TIM1_CCMR1寄存器)。
3. 配置定时器B把定时器A的输出作为自己的触发输入信号(配置TIM1_SMCR寄存器的TS=001)。
4. 配置定时器B为门控触发模式(配置TIM1_SMCR寄存器的SMS=101)。
5. 通过对UG位(TIM1_EGR寄存器)写1, 复位定时器A。
6. 通过对UG位(TIM1_EGR寄存器)写1, 复位定时器B。
7. 将0xE7写入定时器B的计数器中(TIM1_CNTRL), 初始化定时器B。
8. 通过对CEN位(TIM1_CR1寄存器)写1, 使能定时器B。
9. 通过对CEN位(TIM1_CR1寄存器)写1, 启动定时器A。
10. 通过对CEN位(TIM1_CR1寄存器)写0, 停止定时器A。

图54 定时器A的计数器使能信号(CNT_EN)门控触发定时器B

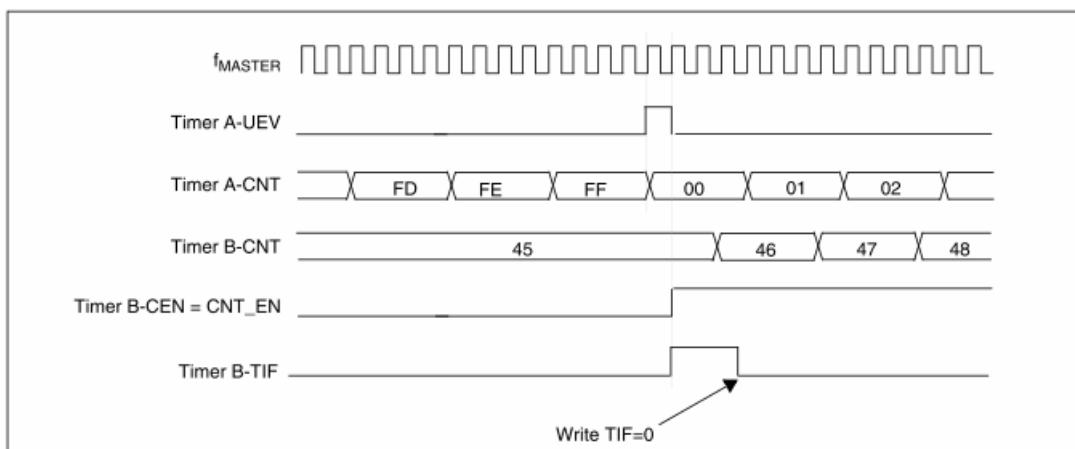


使用一个定时器启动另一个定时器

在本例中，我们用定时器A的更新事件来使能定时器B。定时器B在定时器A发生更新事件时按照定时器B自己的驱动时钟从它的现有值开始计数(可以是非0值)。定时器B在收到触发信号后自动使能CEN位，并开始计数，一直持续到用户向TIM1_CR1寄存器的CEN位写0。两个定时器都使用4分频的 f_{MASTER} 作为驱动时钟($f_{CK_CNT} = f_{MASTER}/4$)。

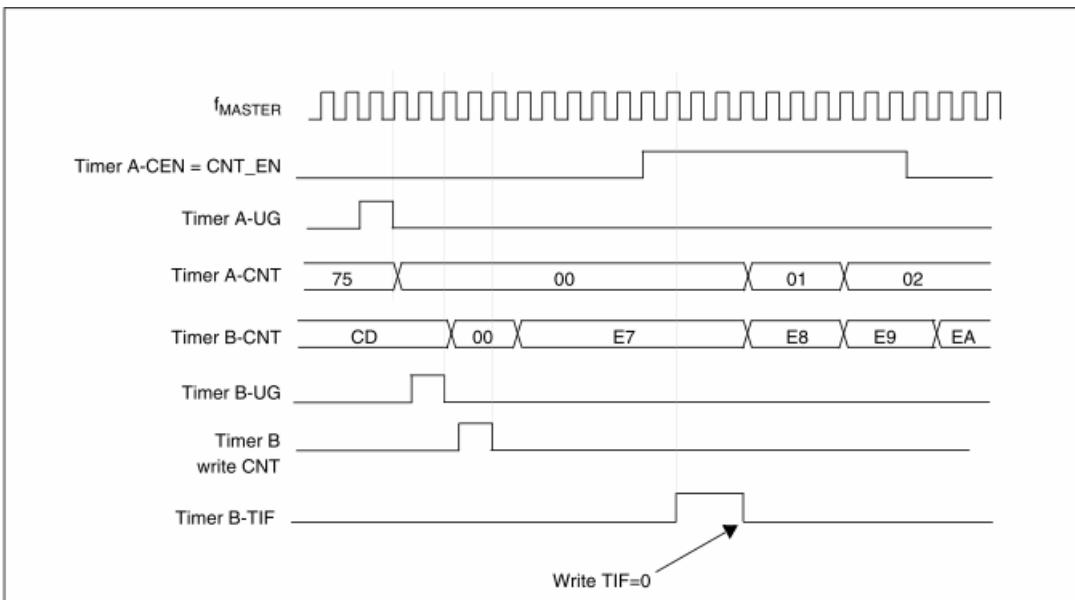
1. 配置定时器A为主模式，输出更新信号(UEV)。(配置TIM1_CR2寄存器的MMS=010)。
2. 配置定时器A的周期(TIM1_ARR寄存器)。
3. 配置定时器B用定时器A的输出作为输入的触发信号(配置TIM1_SMCR寄存器的TS=001)。
4. 配置定时器B为触发模式(配置TIM1_SMCR寄存器的SMS=110)。
5. 置位CEN位(TIM1_CR1寄存器)启动定时器A。

图55 定时器A的更新事件(TIMERA-UEV)触发定时器B



如同前面的例子，用户也可以在启动计数器前对它们初始化。[图54](#)的例子采用和[图54](#)一样的配置，但用普通触发模式(配置TIM1_SMCR寄存器的SMS=110)取代了门控触发模式。

图56 定时器A的计数器使能信号CNT_EN触发定时器B



用外部信号同步的触发两个定时器

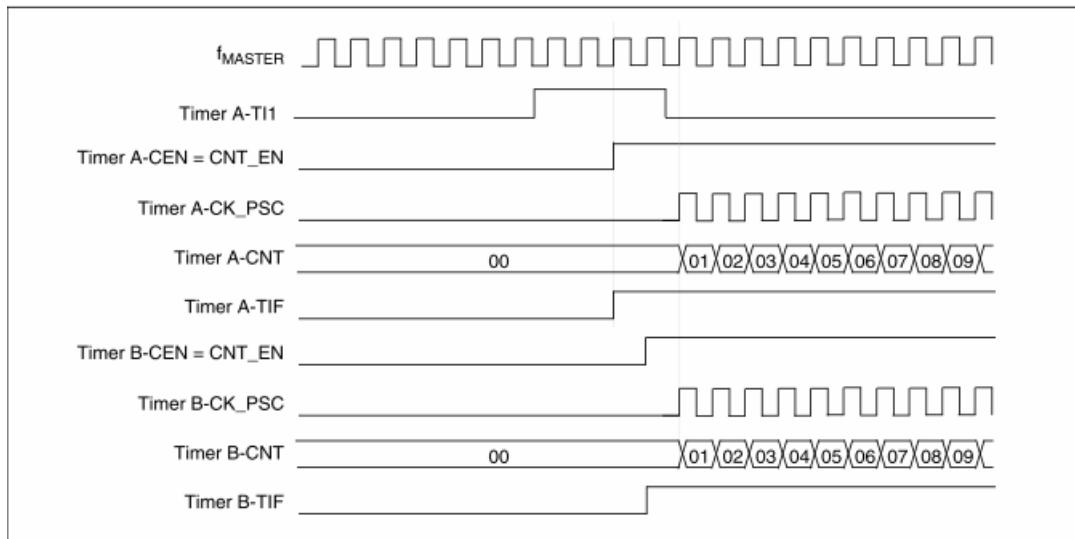
在本例中，使用TI1的上升沿使能定时器A，并同时使能定时器B。定时器如图52连接。为了保持定时器的链接，定时器A需要配置成主/从模式(对于TI1信号为从模式，对于定时器B为主模式)。

1. 配置定时器A为主模式，以输出使能信号作为定时器B的触发(配置TIM1_CR2寄存器的MMS=001)。
2. 配置定时器A为从模式，把TI1信号作为输入的触发信号(配置TIM1_SMCR寄存器的TS=100)。
3. 配置定时器A的触发模式(配置TIM1_SMCR寄存器的SMS=110)。
4. 配置定时器A为主/从模式(配置TIM1_SMCR寄存器的MSM=1)。
5. 配置定时器B以定时器A的输出为输入触发信号(配置TIM1_SMCR寄存器的TS=001)。
6. 配置定时器B的触发模式(配置TIM1_SMCR寄存器的SMS=110)。

当TI1(定时器A的)上出现上升沿时，两个定时器同步的开始计数，并且TIF位都被置起。

注意：在本例中，两个定时器在启动前都进行了初始化(设置UG位)，所以它们都从0开始计数，但是用户也可以通过修改计数器寄存器(TIM1_CNT)来插入一个偏移量，这样的话，在定时器A的CK_PSC信号和CNT_EN信号间会插入延时。

图57 定时器A的TI1信号触发定时器A和定时器B

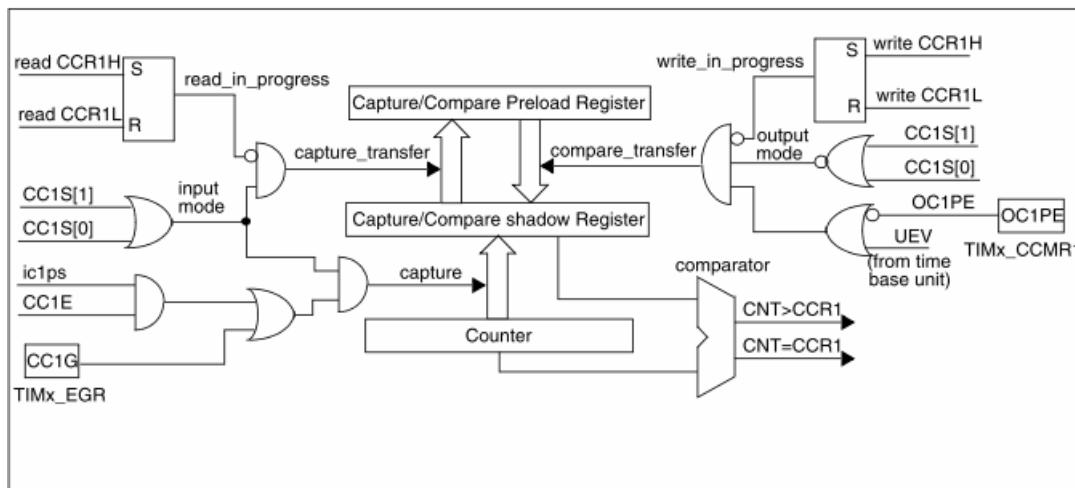


17.5 捕获/比较通道

定时器的I/O引脚(TIM1_CC_i)可以用作输入捕获或者输出比较，这个功能可以通过配置捕获/比较通道模式寄存器(TIM1_CCMR_i)的CC_iS通道选择位来实现，此处的_i代表通道数。

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)来构建的，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58 捕获/比较通道1的主要电路



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。

当通道被配置成输出模式时(TIM1_CCMR_i寄存器的CC_iS=0)，可以随时访问TIM1_CCR_i寄存器。(此处的_i指通道数)

当通道被配置成输入模式时，对TIM1_CCR_i寄存器的读操作类似于计数器的读操作，请参考图59。当捕获发生时，计数器的内容被捕获到TIM1_CCR_i影子寄存器，随后再复制到预装载寄存器中。在读操作进行中，预装载寄存器是被冻结的。

图59 捕获模式下的16位TIM1_CCRi寄存器的读操作

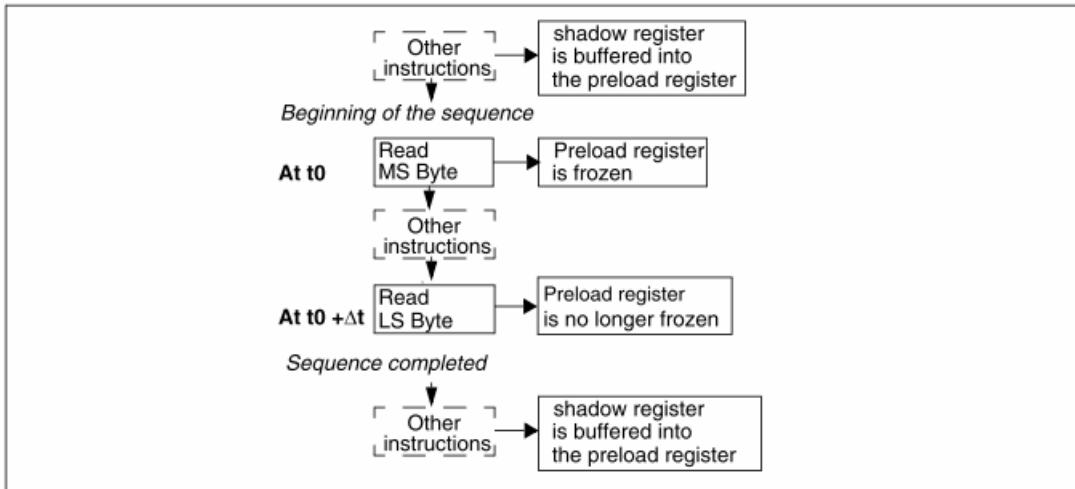


图59描述了16位的CCRi寄存器的读操作流程，被缓存的数据将保持不变直到读流程结束。

在整个读流程结束后，如果仅仅读了TIM1_CCRiL寄存器，返回计数器数值的低位(LS)。

如果在读了低位(LS)数据以后再读高位(MS)数据，将不再返回同样的低位数据。

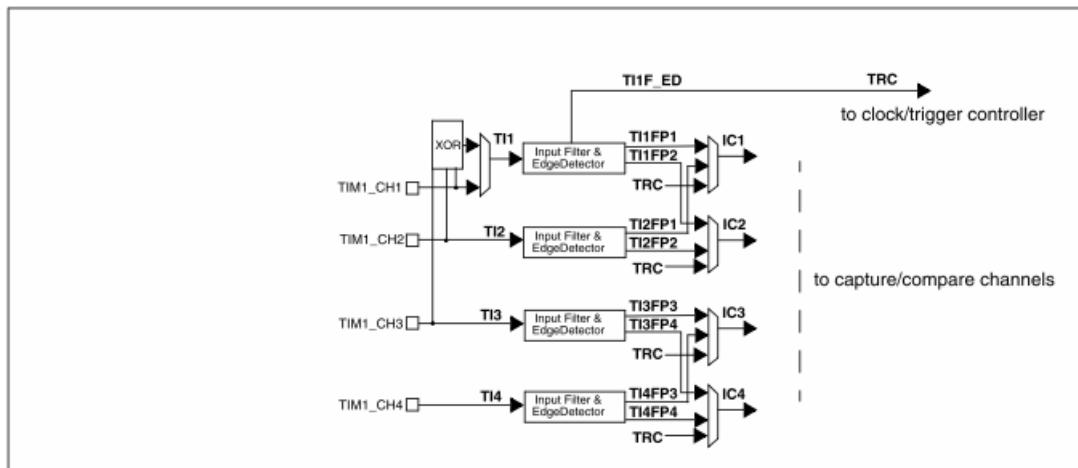
17.5.1 16位TIM1_CCRi寄存器的写流程

16位TIM1_CCRi寄存器的写操作通过预装载寄存器完成。必需使用两条指令来完成整个流程，一条指令对应一个字节。必需先写高位字节(MS)。

在写高位字节(MS)时，影子寄存器的更新被禁止直到低位字节(LS)的写操作完成。不要使用LDW指令，因为该指令先写低位字节，会导致错误的写入。

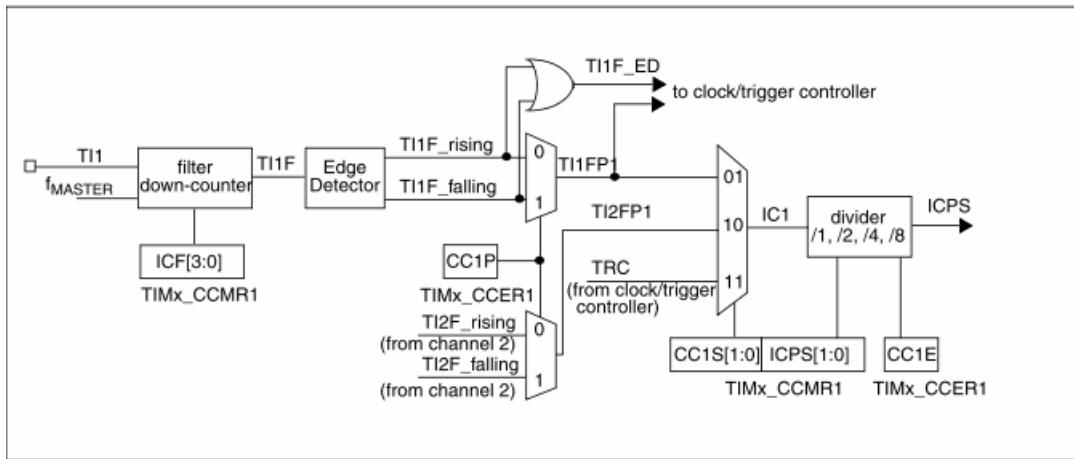
17.5.2 输入模块

图60 输入模块的框图



如图61，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)，它可以作为触发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61 TIM1通道1的输入



17.5.3 输入捕获模式

在输入捕获模式下，当检测到 IC_i 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(TIM1_CCRx)中。当发生捕获事件时，相应的 CC_iIF 标志(TIM1_SR寄存器)被置1。

如果TIM1_IER寄存器的 CC_iIE 位被置位，也就是使能了中断，则将产生中断请求。如果发生捕获事件时 CC_iIF 标志已经为高，那么重复捕获标志 CC_iOF (TIM1_SR2寄存器)被置1。写 $CC_iIF=0$ 或读取存储在TIM1_CCRiL寄存器中的捕获数据都可清除 CC_iIF 。写 $CC_iOF=0$ 可清除 CC_iOF 。

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TIM1_CCR1寄存器中，步骤如下：

1. 选择有效输入端：例如TIM1_CCR1连接到TI1输入，所以写入TIM1_CCR1寄存器中的 $CC1S=01$ ，此时通道被配置为输入，并且TIM1_CCR1寄存器变为只读。
2. 根据输入信号 TI_i 的特点，可通过配置TIM1_CCMRi寄存器中的 IC_iF 位来设置相应的输入滤波器的滤波时间。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；因此我们可以连续采样8次，以确认在 TI_i 上一次真实的边沿变换，即在TIMi_CCMR1寄存器中写入 $IC1F=0011$ ，此时，只有连续采样到8个相同的 TI_i 信号，信号才为有效(采样频率为 f_{MASTER})。
3. 选择TI1通道的有效转换边沿，在TIM1_CCER1寄存器中写入 $CC1P=0$ (上升沿)。
4. 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写TIM1_CCMR1寄存器的 $IC1PS=00$)。
5. 设置TIM1_CCER1寄存器的 $CC1E=1$ ，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置TIM1_IER寄存器中的 $CC1IE$ 位允许相关中断请求。

当发生一个输入捕获时：

- 当产生有效的电平转换时，计数器的值被传送到TIM1_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 $CC1IF$ 未曾被清除时， $CC1OF$ 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($CC1OF$ 位)，建议在读出重复捕获标志之前读取数据，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重复捕获信息。

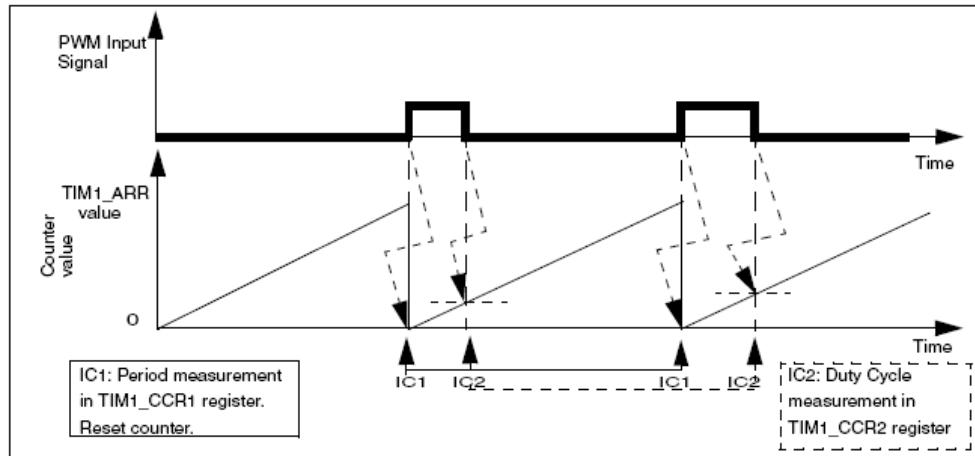
注意：设置TIM1_EGR寄存器中相应的 $CCiG$ 位，可以通过软件产生输入捕获中断。

PWM输入信号测量

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_i 信号被映射至同一个 TI_i 输入。
- 这两个 IC_i 信号的有效边沿的极性相反。
- 其中一个 TI_iFP 信号被作为触发输入信号，而触发模式控制器被配置成复位触发模式。

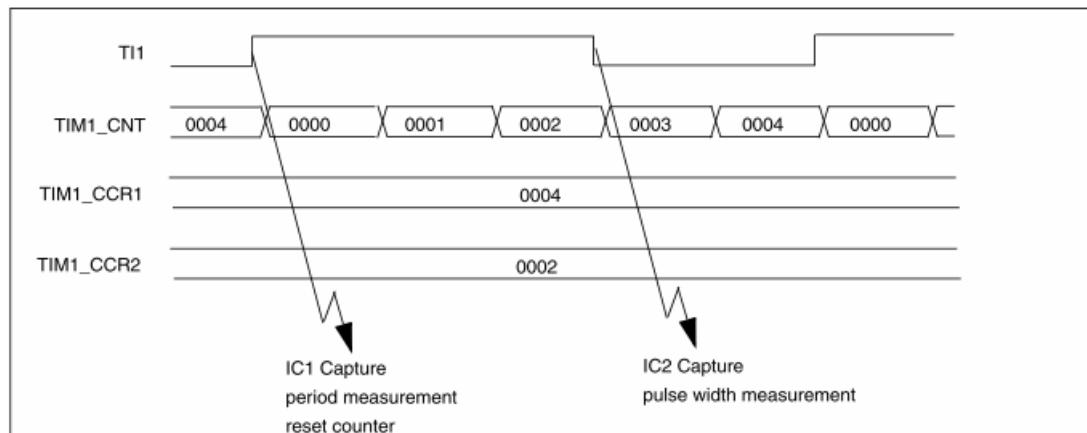
图62 PWM输入信号测量



例如，你可以用以下方式测量TI1上输入的PWM信号的周期(TIM1_CCR1寄存器)和占空比(TIM1_CCR2寄存器)。(具体取决于 f_{MASTER} 的频率和预分频器的值)

1. 选择TIM1_CCR1的有效输入：置TIM1_CCMR1寄存器的CC1S=01(选中TI1)。
2. 选择TI1FP1的有效极性(用来捕获数据到TIM1_CCR1中和清除计数器)：置CC1P=0(上升沿有效)。
3. 选择TIM1_CCR2的有效输入：置TIM1_CCMR2寄存器的CC2S=10(选中TI1FP2)。
4. 选择TI1FP2的有效极性(捕获数据到TIM1_CCR2)：置CC2P=1(下降沿有效)。
5. 选择有效的触发输入信号：置TIM1_SMCR寄存器中的TS=101(选择TI1FP1)。
6. 配置触发模式控制器为复位触发模式：置TIM1_SMCR中的SMS=100。
7. 使能捕获：置TIM1_CCER1寄存器中CC1E=1, CC2E=1。

图63 PWM输入信号测量实例



17.5.4 输出模块

输出模块会产生一个用来做参考的中间波形，称为OCiREF(高有效)。刹车功能和极性的处理都在模块的最后处理。

图64 输出模块框图

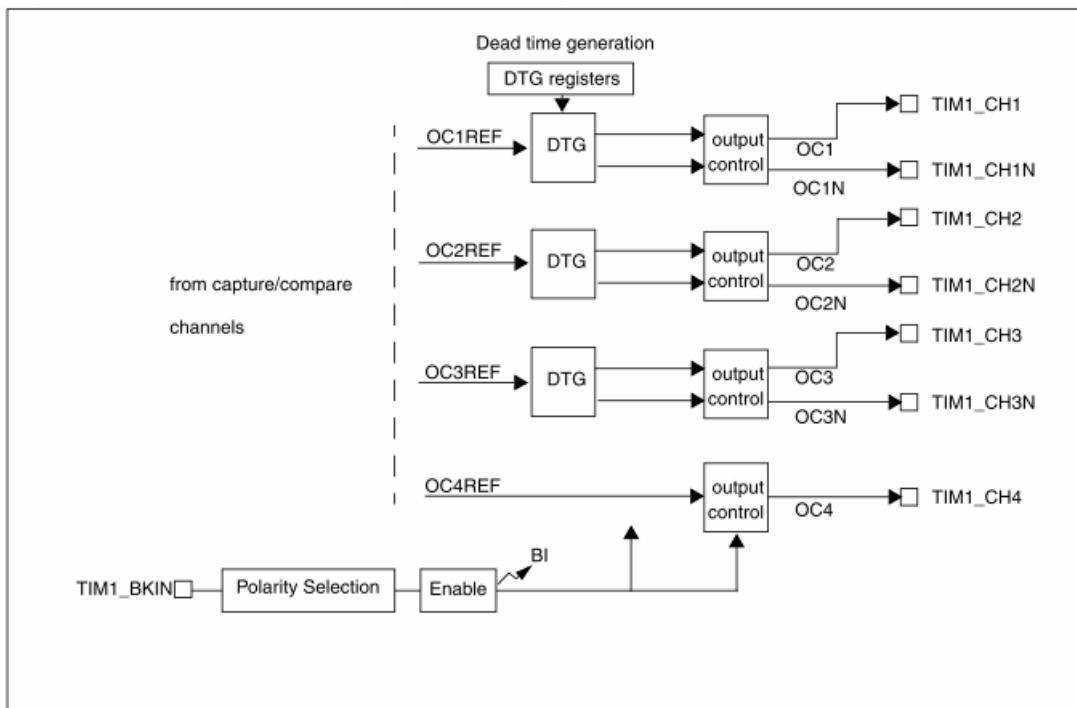
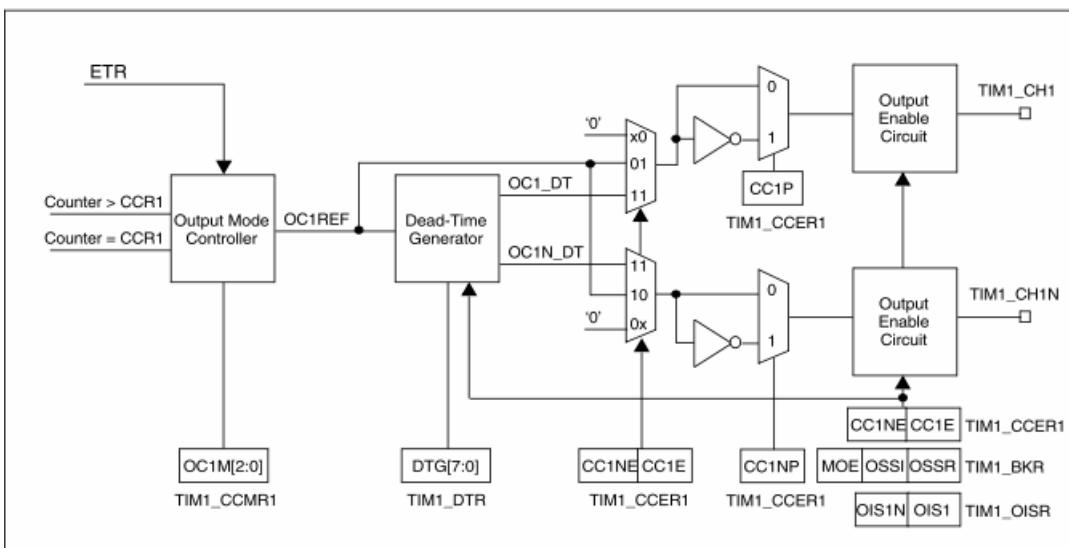


图65 详细的带互补输出的输出模块框图(通道1)



17.5.5 强制输出模式

在输出模式(TIM1_CCMR i 寄存器中CC i S=00)下，输出比较信号能够直接由软件强置为高或低状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置TIM1_CCMR i 寄存器中相应的OC i M=101，即可强置输出比较信号为有效状态。这样OC i REF被强置为高电平(OC i REF始终为高电平有效)，而OC i 的输出是高还是低则取决于CC i P极性标志位。

例如：CC i P=0(OC i 高电平有效)，则OC i 被强置为高电平。

置TIM1_CCMR i 寄存器的OC i M=100，可强置OC i REF信号为低。

该模式下，在TIM1_CCR i 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，也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。

17.5.6 输出比较模式

此模式用来控制一个输出波形或者指示一段给定的时间已经达到。

当计数器与捕获/比较寄存器的内容相同时，有如下操作：

- 根据不同的输出比较模式，相应的OCi输出信号：
 - 保持不变(OCiM=000)
 - 设置为有效电平(OCiM=001)
 - 设置为无效电平(OCiM=010)
 - 翻转(OCiM=011)
- 设置中断状态寄存器中的标志位(TIM1_SR1寄存器中的CCiIF位)。
- 若设置了相应的中断使能位(TIM1_IER寄存器中的CCiIE位)，则产生一个中断。

TIM1_CCMR_i寄存器的OCiM位用于选择输出比较模式，而TIM1_CCMR_i寄存器的CCiP位用于选择有效和无效的电平极性。

TIM1_CCMR_i寄存器的OCiPE位用于选择TIM1_CCR_i寄存器是否需要使用预装载寄存器。

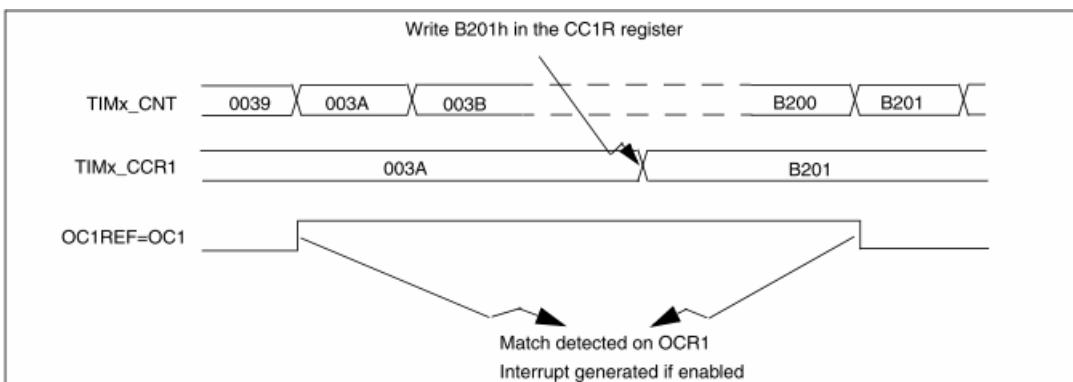
在输出比较模式下，更新事件UEV对OCiREF和OCi输出没有影响。时间精度为计数器的一个计数周期。输出比较模式也能用来输出一个单脉冲。

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TIM1_ARR和TIM1_CCR_i寄存器中。
3. 如果要产生一个中断请求，设置CCiIE位。
4. 选择输出模式步骤：
 - 要求计数器与 CCR_i 匹配时翻转 OCiM 的输出管脚，设置 OCiM=011
 - 置 OCiPE = 0 禁用预装载寄存器
 - 置 CCiP = 0 选择高电平为有效电平
 - 置 CCiE = 1 使能输出
5. 设置TIM1_CR1寄存器的CEN位来启动计数器

TIM1_CCR_i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(OCiPE='0'，否则TIM1_CCR_i的影子寄存器只能在发生下一次更新事件时被更新)。图66给出了一个例子。

图66 输出比较模式：OC1的翻转



17.5.7 PWM模式

脉冲宽度调制(PWM)模式可以产生一个由TIM1_ARR寄存器确定频率、由TIM1_CCR_i寄存器确定占空比的信号。

在TIM1_CCMR_i寄存器中的OCiM位写入'110'(PWM模式1)或'111'(PWM模式2)，能够独立地设置每个OCi输出通道产生一路PWM。必须设置TIM1_CCMR_i寄存器的OCiPE位使能相应的预装载寄存器，也可以设置TIM1_CR1寄存器的ARPE位使能自动重装载的预装载寄存器(在向上计数模式或中央对称模式中)。

由于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TIM1_EGR寄存器的UG位来初始化所有的寄存器。

OCi的极性可以通过软件在TIM1_CCERi寄存器中的CCiP位设置, 它可以设置为高电平有效或低电平有效。OCi的输出使能通过(TIM1_CCERi和TIM1_BKR寄存器中)CCiE、MOE、OISi和OSSI位的组合来控制。详见TIM1_CCERi寄存器的描述。

在PWM模式(模式1或模式2)下, TIM1_CNT和TIM1_CCRI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_CCRI≤TIM1_CNT或者TIM1_CNT≤TIM1_CCRI。

根据TIM1_CR1寄存器中CMS位域的状态, 定时器能够产生边沿对齐的PWM信号或中央对齐的PWM信号。

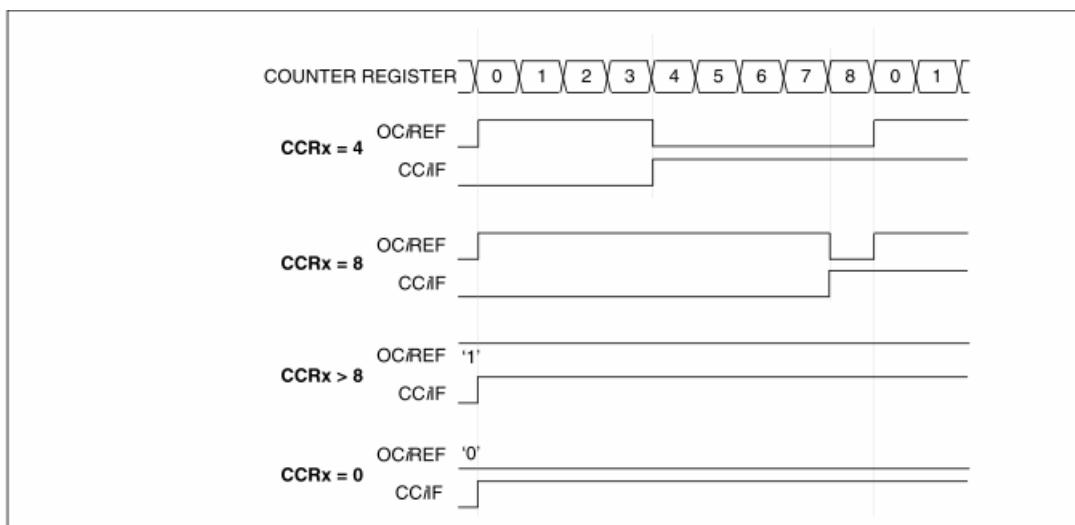
PWM 边沿对齐模式

● 向上计数配置

当TIM1_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TIM1_CNT<TIM1_CCRI时, PWM参考信号OCiREF为高, 否则为低。如果TIM1_CCRI中的比较值大于自动重装载值(TIM1_ARR), 则OCiREF保持为'1'。如果比较值为0, 则OCiREF保持为'0'。下图为TIM1_ARR=8时边沿对齐的PWM波形实例。

图67 边沿对齐, PWM模式1的波形(ARR=8)



● 向下计数的配置

当TIM1_CR1寄存器的DIR位为高时执行向下计数。请参看 [17.3.5](#)。

在PWM模式1时, 当TIM1_CNT>TIM1_CCRI时参考信号OCiREF为低, 否则为高。如果TIM1_CCRI中的比较值大于TIM1_ARR中的自动重装载值, 则OCiREF保持为'1'。该模式下不能产生0%的PWM波形。

PWM 中央对齐模式

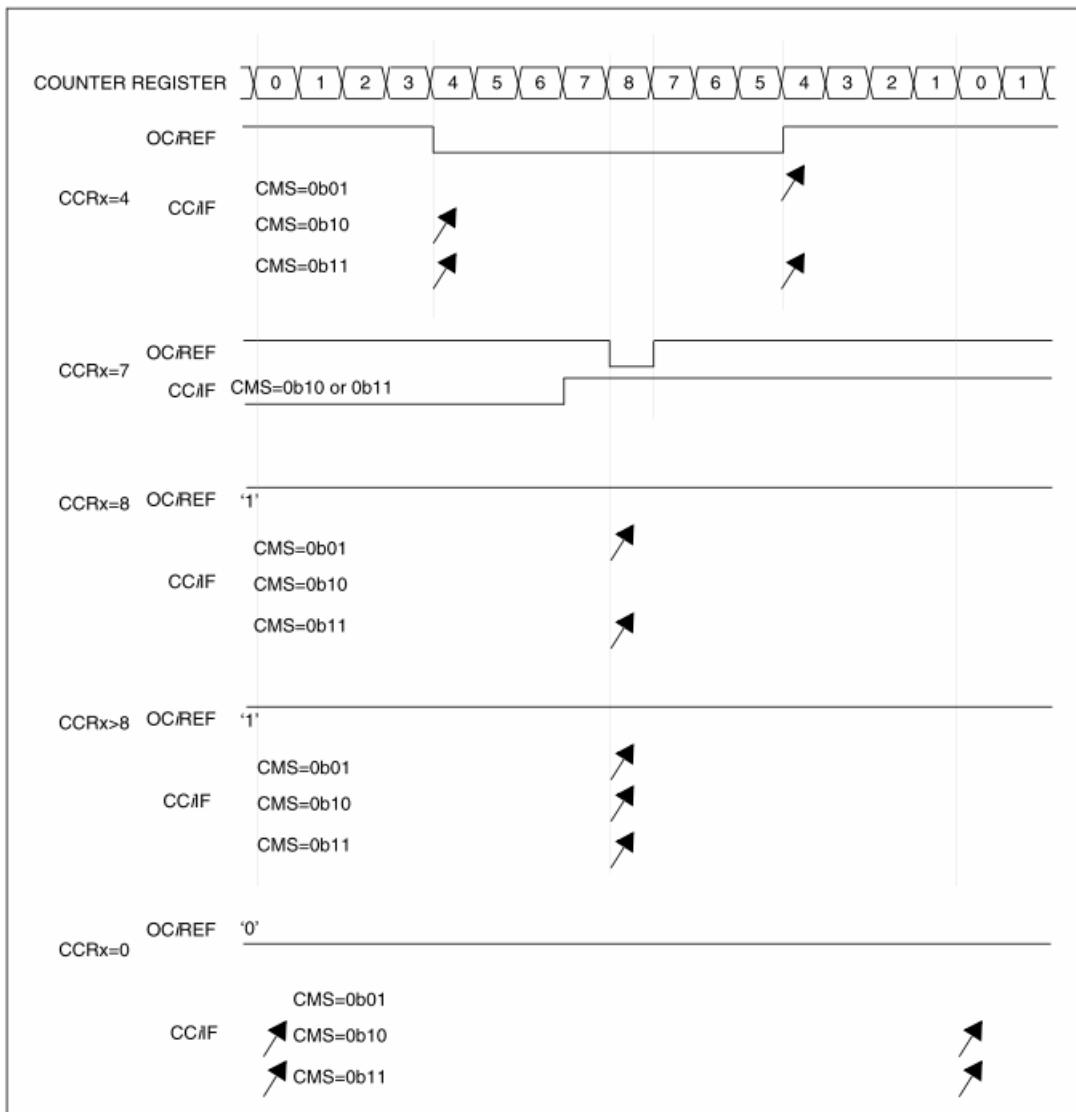
当TIM1_CR1寄存器中的CMS位不为'00'时为中央对齐模式(所有其他的配置对OCiREF/OCi信号都有相同的作用)。

根据不同的CMS位的设置, 比较标志可以在计数器向上计数, 向下计数, 或向上和向下计数时被置1。TIM1_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 [17.3.6](#) 的中央对齐模式。

图68给出了一些中央对齐的PWM波形的例子

- TIMx_ARR=8
- PWM模式1
- 标志位在以下三种情况下被置位(以箭头形式在一中指出)
 - 只有在计数器向下计数时(CMS=01)
 - 只有在计数器向上计数时(CMS=10)
 - 在计数器向上和向下计数时(CMS=11)

图68 中央对齐的PWM波形(APR=8)



单脉冲模式

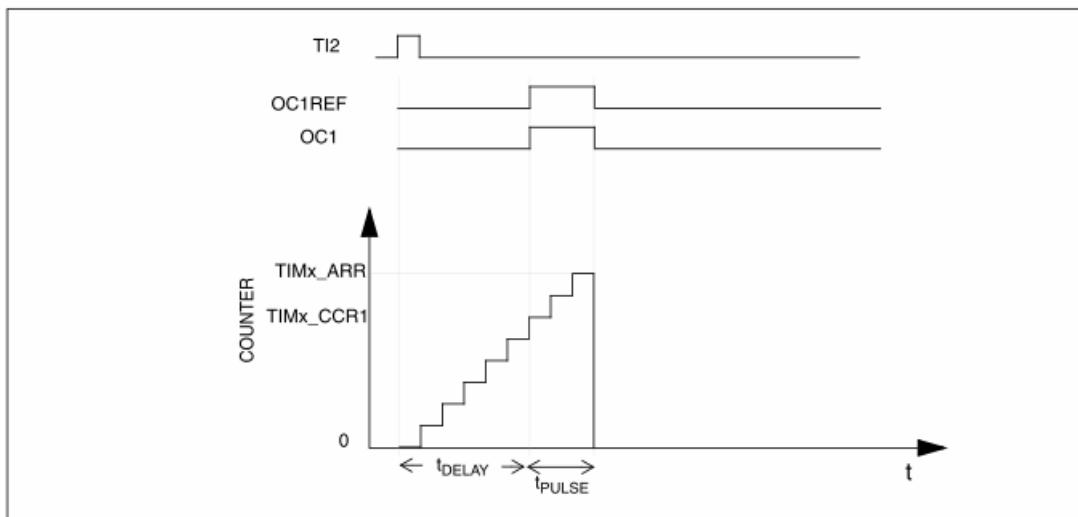
单脉冲模式(OPM)是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可控的脉冲。

可以通过时钟/触发控制器启动计数器，在输出比较模式或者PWM模式下产生波形。设置TIM1_CR1寄存器的OPM位将选择单脉冲模式，此时计数器自动地在下一个更新事件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 $CNT < CCRi \leq ARR$ ，
- 向下计数方式：计数器 $CNT > CCRi$ 。

图69 单脉冲模式图例



例如，你需要在从TI2输入脚上检测到一个上升沿之后，延迟 t_{DELAY} ，在OC1上产生一个 t_{PULSE} 宽度的正脉冲：

假定IC2作为触发1通道的触发源：

- 置TIM1_CCMR2寄存器的CC2S=01，把IC2映射到TI2。
- 置TIM1_CCER1寄存器的CC2P=0，使IC2能够检测上升沿。
- 置TIM1_SMCR寄存器的TS=110，使IC2作为时钟/触发控制器的触发源(TRGI)。
- 置TIM1_SMCR寄存器的SMS=110(触发模式)，IC2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TIM1_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($TIM1_ARR - TIM1_CCR1$)。
- 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TIM1_CCMR1寄存器的OCiM=111，进入PWM模式2；根据需要有选择地设置TIM1_CCMR1寄存器的OC1PE=1，置位TIM1_CR1寄存器中的ARPE，使能预装载寄存器；然后在TIM1_CCR1寄存器中填写比较值，在TIM1_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。

在这个例子中，TIM1_CR1寄存器中的DIR和CMS位应该置低。

因为只需要一个脉冲，所以设置TIM1_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，对T_{li}输入脚的边沿检测会设置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单脉冲的输出。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TIM1_CCMR_i寄存器中的OC_iFE位；此时强制OC_iREF(和OC_x)直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OC_iFE只在通道配置为PWM1和PWM2模式时起作用。

互补输出和死区插入

TIM1能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。请参考图28。

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TIM1_CCER_i寄存器中的CC_iP和CC_iNP位，可以为每一个输出独立地选择极性(主输出OC_i或互补输出OC_iN)。

互补信号OCi和OCiN通过下列控制位的组合进行控制：TIM1_CCERi寄存器的CCiE和CCiNE位，TIM1_BKR寄存器中的MOE、OISi、OISiN、OSSI和OSSR位，详见表34。特别的是，在转换到IDLE状态时(MOE下降到0)死区控制被激活。

同时设置CCiE和CCiNE位将插入死区，如果存在刹车电路，则还要设置MOE位。每一个通道都有一个8位的死区发生器。参考信号OCiREF可以产生2路输出OCi和OCiN。如果OCi和OCiN为高有效：

- OCi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。
- OCiN输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。

如果延迟大于当前有效的输出宽度(OCi或者OCiN)，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OCiREF之间的关系。(假设CCiP=0、CCiNP=0、MOE=1、CCiE=1并且CCiNE=1)

图70 带死区插入的互补输出

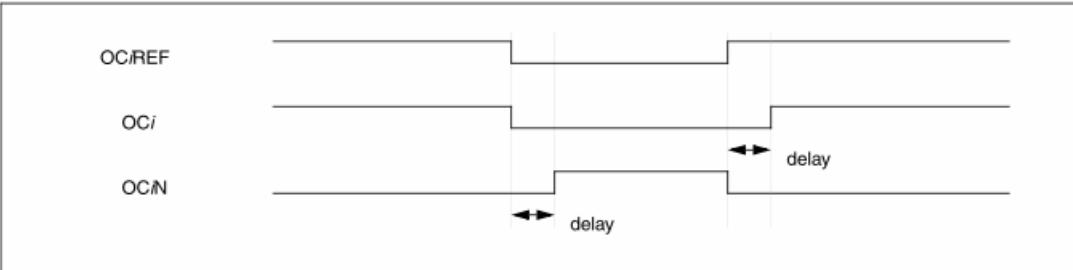


图71 死区波形延迟大于负脉冲

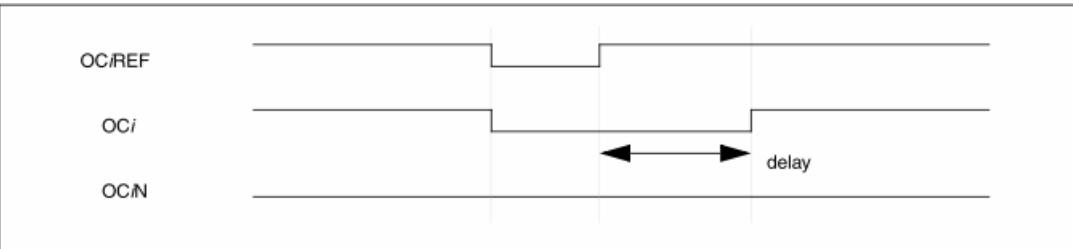
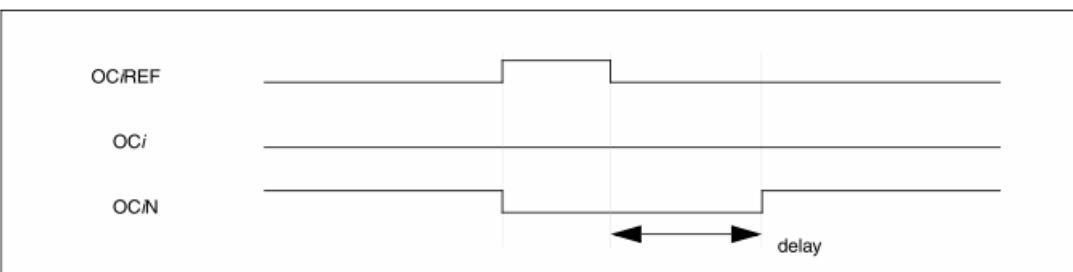


图72 死区波形延迟大于正脉冲



每一个通道的死区延时都是相同的，是由TIM1_DTR寄存器中的DTG位编程配置。延时的计算请参考17.7.31死区寄存器(TIM1_DTR)。

重定向OCiREF到OCi或OCiN

在输出模式下(强置、输出比较或PWM)，通过配置TIM1_CCERi寄存器的CCiE和CCiNE位，OCiREF可以被重定向到OCi或者OCiN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，或同时处于有效电平(此时仍然是带死区的互补输出)。

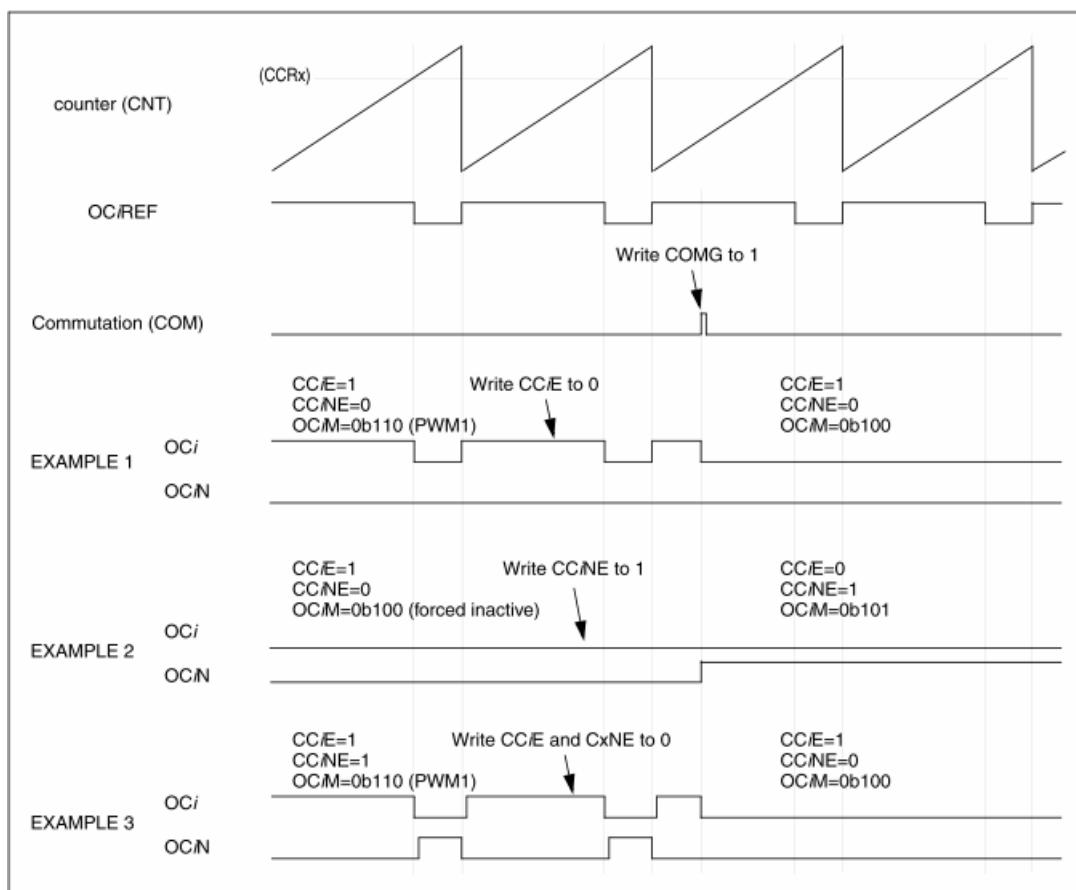
注：当只使能OCiN(CCiE=0, CCiNE=1)时，它不会反相，而当OCiREF变高时立即有效。例如，如果CCiNP=0，则OCiN=OCiREF。另一方面，当OCi和OCiN都被使能时(CCiE=CCiNE=1)，当OCiREF为高时OCi有效；而OCiN相反，当OCiREF低时OCiN变为有效。

针对马达控制的六步PWM输出

当在一个通道上需要互补输出时，预装载位有OCiM、CCiE和CCiNE。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TIM1_EGR寄存器的COMG位由软件产生，或在TRGI上升沿由硬件产生。

图73显示当发生COM事件时，三种不同配置下OCx和OCxN输出。

图73 产生六步PWM，使用COM的例子(OSSR=1)



17.5.8 使用刹车功能

刹车功能常用于马达控制中。当使用刹车功能时，依据相应的控制位(TIM1_BKR寄存器中的MOE、OSSI和OSSR位)，输出使能信号和无效电平都会被修改。

系统复位后，刹车电路被禁止，MOE位为低。设置TIM1_BKR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。

MOE下降沿相对于时钟模块可以是异步的，因此在实际信号(作用在输出端)和同步控制位(在TIM1_BKR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TIM1_OISR寄存器的OISi位设定的电平。如果OSSI=0，则定时器不再控制输出使能信号，否则输出使能信号始终为高。
- 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 OISi 和 OISiN 位指示的电平驱动输出端口。即使在这种情况下，OCi 和 OCiN 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 个时钟周期)。
- 如果设置了TIM1_IER寄存器的BIE位，当刹车状态标志(TIM1_SR1寄存器中的BIF位)为'1'时，则产生一个中断。
- 如果设置了TIM1_BKR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，这可以用来进行波形控制。否则，MOE始终保持低直到被再次置'1'。种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。

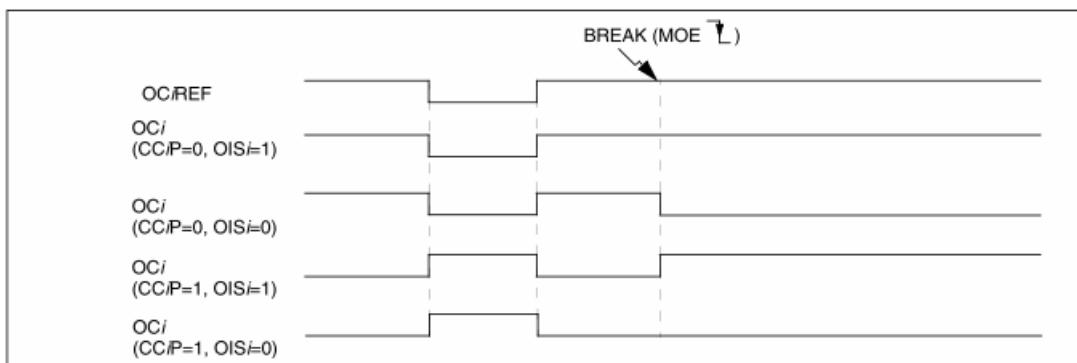
注：刹车输入为电平有效。所以，当刹车输入有效时，不能同时(自动地或者通过软件)设置MOE。同时，状态标志BIF不能被清除。

刹车由BRK输入(BKIN)产生，它的有效极性是可编程的，且由TIM1_BKR寄存器的BKE位开启或禁止。

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(OCi极性和被禁止时的状态，OCiM配置，刹车使能和极性)。用户可以通过TIM1_BKR寄存器的LOCK位，从三种级别的保护中选择一种。在MCU复位后LOCK位域只能被修改一次。

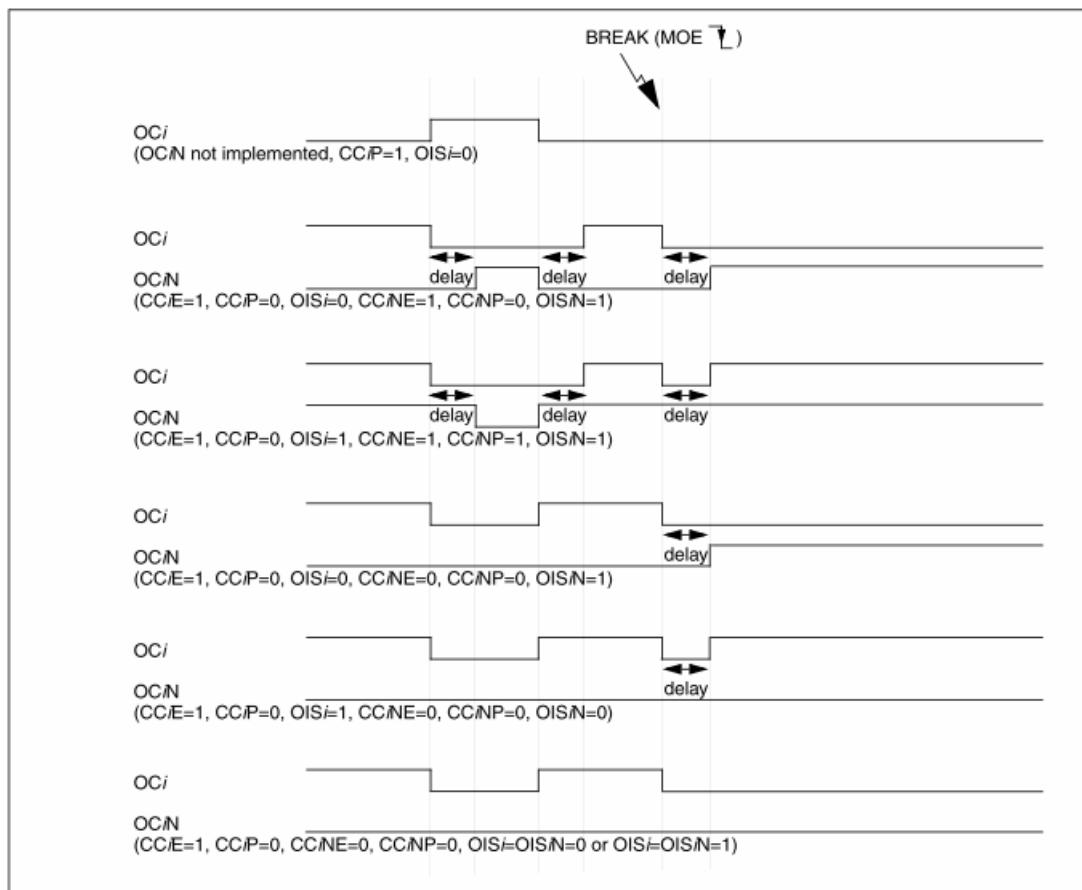
[图74](#)显示刹车响应的输出实例。

图74 刹车响应的输出(不带互补输出的通道)



[图75](#)显示了带互补输出的刹车响应实例

图75 刹车响应的输出(TIM1互补输出)



17.5.9 在外部事件发生时清除OCREF信号

对于一个给定的通道，在ETRF输入端(设置TIM1_CCMR i 寄存器中对应的OCiCE位为'1')的高电平能够把OCREF信号拉低，OCREF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。

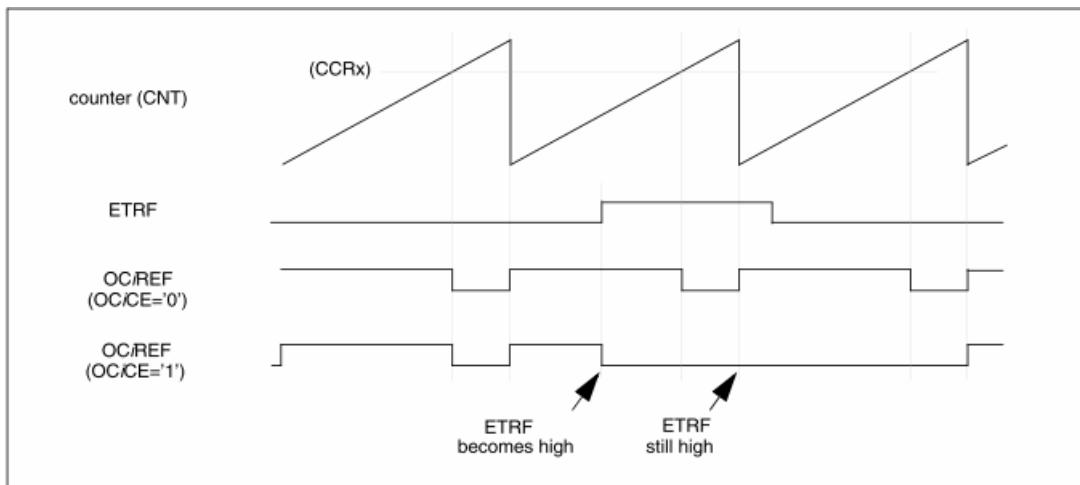
例如，OCREF信号可以联到一个比较器的输出，用于控制电流。这时，ETR必须配置如下：

1. 外部触发预分频器必须处于关闭：TIM1_ETR寄存器中的ETPS[1:0]=00。
2. 必须禁止外部时钟模式2：TIM1_ETR寄存器中的ECE=0。
3. 外部触发极性(ETP)和外部触发滤波器(ETF)可以根据需要配置。

请参考图44，外部触发输入框图。

下图显示了当ETRF输入变为高时，对应不同OCiCE的值，OCREF信号的动作。在这个例子中，定时器TIMx被置于PWM模式。

图76 ETR清除TIM1的OC/REF



17.5.10 编码器接口模式

该模式一般用于马达控制。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TIM1_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TIM1_CCER1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33，假定计数器已经启动(TIM1_CR1寄存器中的CEN=1)，则计数器在每次TI1FP1或TI2FP2上产生有效跳变时计数。TI1FP1和TI2FP2是TI1和TI2在通过输入滤波器和极性控制后的信号；如果没有滤波和极性变换，则TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对TIM1_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TIM1_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TIM1_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33 计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与MCU连接而不需要外部接口逻辑。但是，一般使用比较器将编码器的差分输出转换成数字信号，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。

图77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' (TIM1_CCMR1寄存器, IC1FP1映射到TI1)
- CC2S='01' (TIM1_CCMR2寄存器, IC2FP2映射到TI2)
- CC1P='0' (TIM1_CCER1寄存器, IC1不反相, IC1=TI1)
- CC2P='0' (TIM1_CCER1寄存器, IC2不反相, IC2=TI2)
- SMS='011' (TIM1_SMCR寄存器, 所有的输入均在上升沿和下降沿有效).
- CEN='1' (TIM1_CR1寄存器, 计数器使能)

图77 编码器模式下的计数器操作实例

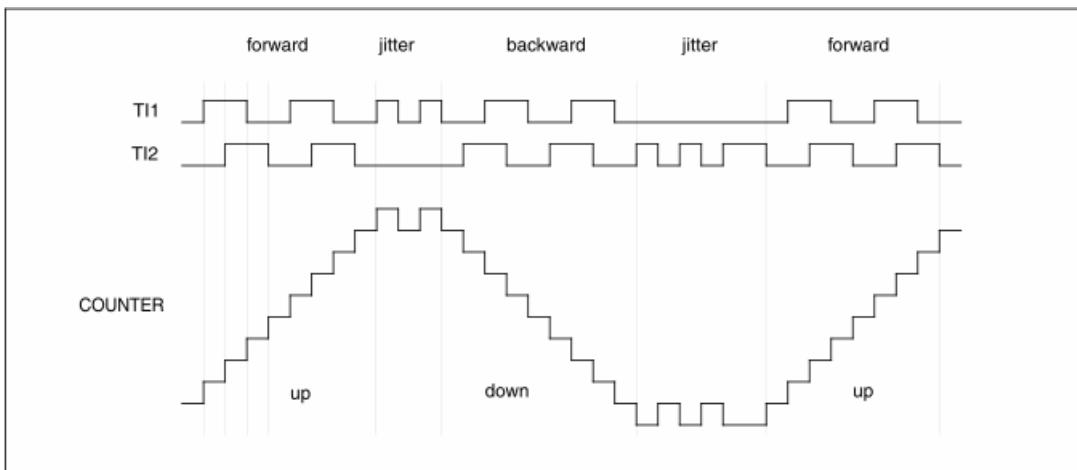
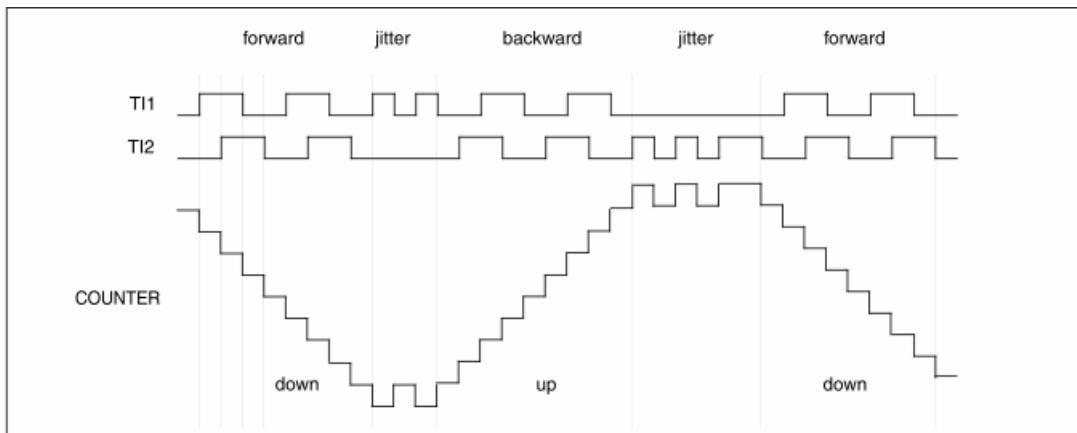


图78为当IC1极性反相时计数器的操作实例(CC1P='1'，其他配置与上例相同)

图78 IC1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。使用另外一个配置在捕获模式下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，可以按照一定的时间间隔读出计数器。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)。

17.6 中断

TIM1有8个中断请求源，分别映射到2个中断矢量上：

- 刹车中断
- 触发中断
- COM事件中断
- 输入捕捉/输出比较4中断

- 输入捕捉/输出比较3中断
- 输入捕捉/输出比较2中断
- 输入捕捉/输出比较1中断
- 更新事件中断(如：计数器上溢，下溢及初始化)

为了使用中断特性，对每个被使用的中断通道，设置**TIM1_IER**寄存器中相应的中断使能位：即**BIE**, **TIE**, **COMIE**, **CCIE**, **UIE**位。

通过设置**TIM1_EGR**寄存器中的相应位，也可以用软件产生上述各个中断源。

17.7 TIM1寄存器描述

17.7.1 控制寄存器 1(TIM1_CR1)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN

r w r w r w r w r w r w r w r w

位7	ARPE: 自动预装载允许位 0: TIM1_ARR寄存器没有缓冲, 它可以被直接写入; 1: TIM1_ARR寄存器由预装载缓冲器缓冲。
位6:5	CMS: 选择中央对齐模式 00: 边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01: 中央对齐模式1。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCIS=00)的输出比较中断标志位, 只在计数器向下计数时被置1。 10: 中央对齐模式2。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCIS=00)的输出比较中断标志位, 只在计数器向上计数时被置1。 11: 中央对齐模式3。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中CCIS=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置1。 注1: 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。 注2: 在中央对齐模式下, 编码器模式 (GPT_SMCR寄存器中的SMS=001, 010, 011) 必须被禁止。
位4	DIR: 方向 0: 计数器向上计数; 1: 计数器向下计数。 注: 当计数器配置为中央对齐模式或编码器模式时, 该位为只读。
位3	OPM: 单脉冲模式 0: 在发生更新事件时, 计数器不停止; 1: 在发生下一次更新事件(清除CEN位)时, 计数器停止。
位2	URS: 更新请求源 0: 如果UDIS允许产生更新事件, 则下述任一事件产生一个更新中断: <ul style="list-style-type: none">- 寄存器被更新(计数器上溢/下溢)- 软件设置UG位- 时钟/触发控制器产生的更新 1: 如果UDIS允许产生更新事件, 则只有当下列事件发生时才产生更新中断, 并UIF置1: <ul style="list-style-type: none">- 寄存器被更新(计数器上溢/下溢)
位1	UDIS: 禁止更新 0: 一旦下列事件发生, 产生更新(UEV)事件: <ul style="list-style-type: none">- 计数器溢出/下溢- 产生软件更新事件- 时钟/触发模式控制器产生的硬件复位 被缓存的寄存器被装入它们的预装载值。 1: 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位或时钟/触发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	CEN: 允许计数器 0: 禁止计数器; 1: 使能计数器。 注: 在软件设置了CEN位后, 外部时钟、门控模式和编码器模式才能工作。然而触发模式可以自动地通过硬件设置CEN位。

17.7.2 控制寄存器 2(TIM1_CR2)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
TI1S		MMS [2:0]		保留	COMS	保留	CCPC

rw rw rw rw rw rw rw

位7	TI1S: TI1选择 0: CC1输入管脚连到TI1(数字滤波器的输入); 1: CC1、CC2和CC3管脚经异或后连到TI1。
位6:4	MMS: 主模式选择 该位用于选择在主模式下送到ADC或其它从定时器的同步信息(TRGO)。可能的组合如下: 000: 复位 – TIM1_EGR寄存器的UG位被用于作为触发输出(TRGO)。如果触发输入(时钟/触发控制器配置为复位模式)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 – 计数器使能信号被用于作为触发输出(TRGO)。其用于启动多个定时器或ADC，以便控制在一段时间内使能从定时器或ADC。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。除非选择了主/从模式(见TIM1_SMCR寄存器中MSM位的描述)，当计数器使能信号受控于触发输入时，TRGO上会有一个延迟。 010: 更新 – 更新事件被选为触发输入(TRGO)。 011: 比较脉冲(MATCH1) – 一旦发生一次捕获或一次比较成功，当CC1IF标志被置1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	保留，必须保持为0。
位2	COMS: 捕获/比较控制位的更新控制选择 0: 当捕获/比较的控制位为预装载时(CCPC=1)，只有在COMG位置1的时候这些控制位才被更新； 1: 当捕获/比较的控制位为预装载时(CCPC=1)，只有在COMG位置1或TRGI发生上升沿的时候这些控制位才被更新； 注: 该位只对拥有互补输出的通道有效。
位1	保留，被硬件设为0。
位0	CCPC: 捕获/比较预装载控制位 0: CCIE, CCNE, CCP, CCNP位(TIM1_CCERx寄存器)和OCIM位(TIM1_CCMRx寄存器)不是预装载的； 1: CCIE, CCNE, CCP, CCNP和OCIM位是预装载的；设置该位后，它们只在设置了COMG位(TIM1_EGR寄存器)后被更新。 注: 该位只对具有互补输出的通道起作用。

17.7.3 从模式控制寄存器(TIM1_SMCR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
MSM	TS[2:0]			-	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw

位7	MSM: 主/从模式 0: 无作用; 1: 触发输入(TRGI)上的事件被延迟了, 以允许定时器1与它的从定时器间的完美同步(通过TRGO)。
位6:4	TS: 触发选择 这3位选择用于选择同步计数器的触发输入。 000: 内部触发ITR0连接到TIM6 TRGO 100: TI1的边沿检测器(TI1F_ED) 001: 保留 101: 滤波后的定时器输入1(TI1FP1) 010: 内部触发ITR2连接到TIM5 TRGO 110: 滤波后的定时器输入2(TI2FP2) 011: 保留 注: 这些位只能在未用到(如SMS=000)时被改变, 以避免在改变时产生错误的边沿检测。
位3	保留, 始终读为0。
位2:0	SMS: 时钟/触发/从模式选择 当选择了外部信号, 触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000: 时钟/触发控制器禁止 – 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 – 在选中的触发输入(TRGI)的上升沿时重新初始化计数器, 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_ED被选为触发输入(TS=100)时, 不要使用门控模式。这是因为TI1F_ED在每次TI1F变化时只是输出一个脉冲, 然而门控模式是要检查触发输入的电平。

17.7.4 外部触发寄存器(TIM1_ETR)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]			
RW	RW	RW	RW	RW	RW	RW	RW

位7	ETP: 外部触发极性 该位决定是ETR还是 \overline{ETR} 用于触发操作。 0: ETR不反相, 即高电平或上升沿有效; 1: ETR反相, 即低电平或下降沿有效。																	
位6	ECE: 外部时钟使能 该位用于使能外部时钟模式2。 0: 禁止外部时钟模式2; 1: 使能外部时钟模式2, 计数器的时钟为ETRF的有效沿。 注1: ECE位置1的效果与选择把TRGI连接到ETRF的外部时钟模式1相同(TIM1_SMCR寄存器中, SMS=111, TS=111)。 注2: 外部时钟模式2可与下列模式同时使用: 触发标准模式; 触复发位模式; 触发门控模式。但是, 此时TRGI决不能与ETRF相连(TIM1_SMCR寄存器中, TS不能为111)。 注3: 外部时钟模式1与外部时钟模式2同时使能, 外部时钟输入为ETRF。																	
位5:4	ETPS: 外部触发预分频器 外部触发信号EPRP的频率最大不能超过 $f_{MASTER}/4$ 。可用预分频器来降低ETRP的频率, 当EPRP的频率很高时, 它非常有用: 00: 预分频器关闭; 01: EPRP的频率/2; 02: EPRP的频率/4; 03: EPRP的频率/8。																	
位3:0	ETF: 外部触发滤波器选择 该位域定义了ETRP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: <table> <tbody> <tr> <td>0000: 无滤波器, 以f_{MASTER}采样</td> <td>1000: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=6</td> </tr> <tr> <td>0001: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=2</td> <td>1001: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=8</td> </tr> <tr> <td>0010: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=4</td> <td>1010: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=5</td> </tr> <tr> <td>0011: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=8</td> <td>1011: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=6</td> </tr> <tr> <td>0100: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=6</td> <td>1100: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=8</td> </tr> <tr> <td>0101: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=8</td> <td>1101: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=5</td> </tr> <tr> <td>0110: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=6</td> <td>1110: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=6</td> </tr> <tr> <td>0111: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=8</td> <td>1111: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=8</td> </tr> </tbody> </table>		0000: 无滤波器, 以 f_{MASTER} 采样	1000: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=6	0001: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=2	1001: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=8	0010: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=4	1010: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=5	0011: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=8	1011: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=6	0100: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=6	1100: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=8	0101: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=8	1101: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=5	0110: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=6	1110: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=6	0111: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=8	1111: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=8
0000: 无滤波器, 以 f_{MASTER} 采样	1000: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=6																	
0001: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=2	1001: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=8																	
0010: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=4	1010: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=5																	
0011: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=8	1011: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=6																	
0100: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=6	1100: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=8																	
0101: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=8	1101: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=5																	
0110: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=6	1110: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=6																	
0111: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=8	1111: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=8																	

17.7.5 中断使能寄存器(TIM1_IER)

地址偏移值: 0x04

复位值: 0x00

7	6	5	4	3	2	1	0
BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
rw	rw	rw	rw	rw	rw	rw	rw

位7	BIE: 允许刹车中断 0: 禁止刹车中断; 1: 允许刹车中断。
位6	TIE: 触发中断使能 0: 禁止触发中断; 1: 使能触发中断。
位5	COMIE: 允许COM中断 0: 禁止COM中断; 1: 允许COM中断。
位4	CC4IE: 允许捕获/比较4中断 0: 禁止捕获/比较4中断; 1: 允许捕获/比较4中断。
位3	CC3IE: 允许捕获/比较3中断 0: 禁止捕获/比较3中断; 1: 允许捕获/比较3中断。
位2	CC2IE: 允许捕获/比较2中断 0: 禁止捕获/比较2中断; 1: 允许捕获/比较2中断。
位1	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断; 1: 允许捕获/比较1中断。
位0	UIE: 允许更新中断 0: 禁止更新中断; 1: 允许更新中断。

17.7.6 状态寄存器 1 (TIM1_SR1)

地址偏移值: 0x05

复位值: 0x00

7	6	5	4	3	2	1	0
BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
rw	rw	rw	rw	rw	rw	rw	rw

位7	BIF: 刹车中断标记 一旦刹车输入有效, 由硬件对该位置1。如果刹车输入无效, 则该位可由软件清0。 0: 无刹车事件产生; 1: 刹车输入上检测到有效电平。
位6	TIF: 触发器中断标记 当发生触发事件(当从模式控制器处于除门控模式外的其它模式时, 在TRGI输入端检测到有效边沿, 或门控模式下的任一边沿)时由硬件对该位置1。它由软件清0。 0: 无触发器事件产生; 1: 触发中断等待响应。
位5	COMIF: COM中断标记 一旦产生COM事件(当捕获/比较控制位: CC1E、CC1NE、OCM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	CC4IF: 捕获/比较4中断标记 参考CC1IF描述。
位3	CC3IF: 捕获/比较3中断标记 参考CC1IF描述。
位2	CC2IF: 捕获/比较2中断标记 参考CC1IF描述。
位1	CC1IF: 捕获/比较1中断标记 如果通道CC1配置为输出模式: 当计数器值与比较值匹配时该位由硬件置1, 但在中心对称模式下除外(参考TIM1_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。 注: 在中心对称模式下, 当计数器值为0时, 向上计数, 当计数器值为ARR时, 向下计数(它从0向上计数到ARR-1, 再由ARR向下计数到1)。因此, 对所有的SMS位值, 这两个值都不置标记。但是, 如果CCR1>ARR, 则当CNT达到ARR值时, CC1IF置1。 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TIM1_CCR1L清0。 0: 无输入捕获产生; 1: 计数器值已被捕获(拷贝)至TIM1_CCR1(在IC1上检测到与所选极性相同的边沿)。
位0	UIF: 更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0: 无更新事件产生; 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1: <ul style="list-style-type: none">- 若TIM1_CR1寄存器的UDIS=0, 当计数器上溢或下溢时;- 若TIM1_CR1寄存器的UDIS=0、URS=0, 当设置TIM1_EGR寄存器的UG位软件对计数器CNT重新初始化时;- 若TIM1_CR1寄存器的UDIS=0、URS=0, 当计数器CNT被触发事件重新初始化时 (参考0从模式控制寄存器TIM1_SMCR)。

17.7.7 状态寄存器 2(TIM1_SR2)

地址偏移值: 0x06

复位值: 0x00

7	6	5	4	3	2	1	0
	res		CC40F	CC30F	CC20F	CC10F	res
rw	rw	rw	rw	rw	rw	rw	rw

位7:5	保留, 始终读为0。
位4	CC4OF: 捕获/比较4重复捕获标记 参见CC1OF描述。
位3	CC3OF: 捕获/比较3重复捕获标记 参见CC1OF描述。
位2	CC2OF: 捕获/比较2重复捕获标记 参见CC1OF描述。
位1	CC1OF: 捕获/比较1重复捕获标记 仅当相应的通道被配置为输入捕获时, 该标记可由硬件置1。写0可清除该位。 0: 无重复捕获产生; 1: 计数器的值被捕获到TIM1_CCR1寄存器时, CC1IF的状态已经为1。
位0	保留, 始终读为0。

17.7.8 事件产生寄存器(TIM1_EGR)

地址偏移值: 0x07

复位值: 0x00

7	6	5	4	3	2	1	0
BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
rw	rw	rw	rw	rw	rw	rw	rw

位7	BG: 产生刹车事件 该位由软件置1, 用于产生一个刹车事件, 由硬件自动清0。 0: 无动作; 1: 产生一个刹车事件。此时MOE=0、BIF=1, 若开启对应的中断(BIE=1), 则产生相应的中断。
位6	TG: 产生触发事件 该位由软件置1, 用于产生一个触发事件, 由硬件自动清0。 0: 无动作; 1: TIM1_SR寄存器的TIF=1, 若开启对应的中断 (TIE=1), 则产生相应的中断。
位5	COMG: 捕获/比较事件, 产生控制更新 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 当CCPC=1, 允许更新CCIE、CCNE、CCP, CCNP, OCIM位。 注: 该位只对拥有互补输出的通道有效。
位4	CC4G: 产生捕获/比较4事件 参考CC1G描述。
位3	CC3G: 产生捕获/比较3事件 参考CC1G描述。
位2	CC2G: 产生捕获/比较2事件 参考CC1G描述。
位1	CC1G: 产生捕获/比较1事件 该位由软件置1, 用于产生一个捕获/比较事件, 由硬件自动清0。 0: 无动作; 1: 在通道CC1上产生一个捕获/比较事件: 若通道CC1配置为输出: 设置CC1IF=1, 若开启对应的中断, 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TIM1_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	UG: 产生更新事件 该位由软件置1, 由硬件自动清0。 0: 无动作; 1: 重新初始化计数器, 并产生一个更新事件。注意预分频器的计数器也被清0(但是预分频系数不变)。若在中心对称模式下或DIR=0(向上计数)则计数器被清0; 若DIR=1(向下计数)则计数器取TIM1_ARR的值。

17.7.9 捕获/比较模式寄存器 1(TIM1_CCMR1)

地址偏移值: 0x08

复位值: 0x00

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CC1S位定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx描述了通道在输出模式下的功能，ICxx描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

- **输出模式:**

7	6	5	4	3	2	1	0
OC1CE	OC1M[2:0]		OC1PE	OC1FE		CC1S[1:0]	
rW	rW	rW	rW	rW	rW	rW	rW

位7	OC1CE: 输出比较1清零使能 该位用于使能使用TIM1_TRIG引脚上的外部事件来清通道1的输出信号(OC1REF)，参考 17.5.9在外部事件发生时清除OCREF信号 0: OC1REF 不受ETRF输入(来自TIM1_TRIG引脚)的影响； 1: 一旦检测到ETRF输入高电平，OC1REF=0。
位6:4	OC1M[2:0]: 输出比较1模式 该3位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1的值。OC1REF是高电平有效，而OC1的有效电平取决于CC1P位。 000: 冻结。输出比较寄存器TIM1_CCR1与计数器TIM1_CNT间的比较对OC1REF不起作用； 001: 匹配时设置通道1的输出为有效电平。当计数器TIM1_CNT的值与捕获/比较寄存器1(TIM1_CCR1)相同时，强制OC1REF为高。 010: 匹配时设置通道1的输出为无效电平。当计数器TIM1_CNT的值与捕获/比较寄存器1(TIM1_CCR1)相同时，强制OC1REF为低。 011: 翻转。当TIM1_CCR1=TIM1_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TIM1_CNT<TIM1_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TIM1_CNT>TIM1_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TIM1_CNT<TIM1_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TIM1_CNT>TIM1_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(TIM1_BKR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。(参考17.5.7PWM模式) 注3: 在有互补输出的通道上，这些位是预装载的。如果TIM1_CR2寄存器的CCPC=1，OCM位只有在COM事件发生时，才从预装载位取新值。
位3	OC1PE: 输出比较1预装载使能 0: 禁止TIM1_CCR1寄存器的预装载功能，可随时写入TIM1_CCR1寄存器，并且新写入的数值立即起作用。 1: 开启TIM1_CCR1寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIM1_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(TIM1_BKR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: 为了操作正确，在PWM模式下必须使能预装载功能。但在单脉冲模式下(TIM1_CR1寄存器的OPM=1)，它不是必须的。

位2	<p>OC1FE: 输出比较1 快速使能 该位用于加快CC输出对触发输入事件的响应。</p> <p>0: 根据计数器与CCR1的值, CC1正常操作, 即使触发器是打开的。当触发器的输入有一个有效沿时, 激活CC1输出的最小延时为5个时钟周期。</p> <p>1: 输入到触发器的有效沿的作用就象发生了一次比较匹配。因此, OC被设置为比较电平而与比较结果无关。采样触发器的有效沿和CC1输出间的延时被缩短为3个时钟周期。</p> <p>OCFE只在通道被配置成PWM1或PWM2模式时起作用。</p>
位1:0	<p>CC1S[1:0]: 捕获/比较1 选择。 这2位定义通道的方向(输入/输出), 及输入脚的选择:</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TI1FP1上;</p> <p>10: CC1通道被配置为输入, IC1映射在TI2FP1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TIM1_SMCR寄存器的TS位选择)。</p> <p>注: CC1S仅在通道关闭时(TIM1_CCER1寄存器的CC1E=0)才是可写的。</p>

● 输入捕获模式:

7	6	5	4	3	2	1	0
IC1F[3:0]			IC1PSC[1:0]			CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

位7:4	<p>IC1F[3:0]: 输入捕获1滤波器 这几位定义了TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成, 只有发生了N个事件后输出的跳变才被认为有效。</p> <table> <tr> <td>0000: 无滤波器, $f_{SAMPLING}=f_{MASTER}$</td><td>1000: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=6</td></tr> <tr> <td>0001: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=2</td><td>1001: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=8</td></tr> <tr> <td>0010: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=4</td><td>1010: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=5</td></tr> <tr> <td>0011: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=8</td><td>1011: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=6</td></tr> <tr> <td>0100: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=6</td><td>1100: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=8</td></tr> <tr> <td>0101: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=8</td><td>1101: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=5</td></tr> <tr> <td>0110: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=6</td><td>1110: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=6</td></tr> <tr> <td>0111: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=8</td><td>1111: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=8</td></tr> </table> <p>注: 即使对于带互补输出的通道, 该位域也是非预装载的, 并且不会考虑CCPC (TIM1_CR2寄存器) 的值。</p>	0000: 无滤波器, $f_{SAMPLING}=f_{MASTER}$	1000: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=6	0001: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=2	1001: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=8	0010: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=4	1010: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=5	0011: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=8	1011: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=6	0100: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=6	1100: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=8	0101: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=8	1101: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=5	0110: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=6	1110: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=6	0111: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=8	1111: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=8
0000: 无滤波器, $f_{SAMPLING}=f_{MASTER}$	1000: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=6																
0001: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=2	1001: 采样频率 $f_{SAMPLING}=f_{MASTER}/8$, N=8																
0010: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=4	1010: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=5																
0011: 采样频率 $f_{SAMPLING}=f_{MASTER}$, N=8	1011: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=6																
0100: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=6	1100: 采样频率 $f_{SAMPLING}=f_{MASTER}/16$, N=8																
0101: 采样频率 $f_{SAMPLING}=f_{MASTER}/2$, N=8	1101: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=5																
0110: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=6	1110: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=6																
0111: 采样频率 $f_{SAMPLING}=f_{MASTER}/4$, N=8	1111: 采样频率 $f_{SAMPLING}=f_{MASTER}/32$, N=8																
位3:2	<p>IC1PSC[1:0]: 输入/捕获1预分频器 这2位定义了CC1输入(IC1)的预分频系数。</p> <p>一旦CC1E=0(TIM1_CCER寄存器中), 则预分频器复位。</p> <p>00: 无预分频器, 捕获输入口上检测到的每一个边沿都触发一次捕获;</p> <p>01: 每2个事件触发一次捕获;</p> <p>10: 每4个事件触发一次捕获;</p> <p>11: 每8个事件触发一次捕获。</p>																

位1:0	<p>CC1S[1:0]: 捕获/比较1 选择。</p> <p>这2位定义通道的方向(输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC1映射在TI1FP1上;</p> <p>10: CC1通道被配置为输入, IC1映射在TI2FP1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TIM1_SMCR寄存器的TS位选择)。</p> <p>注: CC1S仅在通道关闭时(TIM1_CCER1寄存器的CC1E=0)才是可写的。</p>
------	--

17.7.10 捕获/比较模式寄存器 2(TIM1_CCMR2)

地址偏移值: 0x09

复位值: 0x00

● 输出比较模式:

7	6	5	4	3	2	1	0
0C2CE	OC2M[2:0]		0C2PE	0C2FE	CC2S[1:0]		

rW rW rW rW rW rW rW rW

位7	OC2CE: 输出比较2清零使能 该位用于使能使用TIM1_TRIG引脚上的外部事件来清通道2的输出信号(OC2REF)，参考 17.5.9 在外部事件发生时清除OCREF信号 0: OC2REF 不受ETRF输入(来自TIM1_TRIG引脚)的影响； 1: 一旦检测到ETRF输入高电平，OC2REF=0。
位6:4	OC2M[2:0]: 输出比较2模式
位3	OC2PE: 输出比较2预装载使能
位2	OC2FE: 输出比较2快速使能
位1:0	CC2S[1:0]: 捕获/比较2选择。 该位定义通道的方向(输入/输出)，及输入脚的选择： 00: CC2通道被配置为输出； 01: CC2通道被配置为输入，IC2映射在TI2FP2上； 10: CC2通道被配置为输入，IC2映射在TI1FP2上； 11: 预留 注: CC2S仅在通道关闭时(TIM1_CCER1寄存器的CC2E=0, CC2NE=0且已被更新)才是可写的。

● 输入捕获模式:

7	6	5	4	3	2	1	0
	IC2F[3:0]		IC2PSC[1:0]	CC2S[1:0]			

rW rW rW rW rW rW rW rW

位7:4	IC2F[3:0]: 输入捕获2滤波器
位3:2	IC2PSC[1:0]: 输入/捕获2预分频器
位1:0	CC2S[1:0]: 捕获/比较2选择。 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TI2FP2上； 10: CC2通道被配置为输入，IC2映射在TI1FP2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TIM1_SMCR寄存器的TS位选择)。 注: CC2S仅在通道关闭时(TIM1_CCER1寄存器的CC2E=0, CC2NE=0且已被更新)才是可写的。

17.7.11 捕获/比较模式寄存器 3(TIM1_CCMR3)

地址偏移值: 0x0A

复位值: 0x00

请参考上述CCMR1寄存器描述。

- **输出比较模式:**

7	6	5	4	3	2	1	0
OC3CE		OC3M[2:0]		OC3PE	OC3FE		CC3S[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

位7	OC3CE: 输出比较3清零使能 该位用于使能使用TIM1_TRIG引脚上的外部事件来清通道3的输出信号(OC3REF), 参考 17.5.9 在外部事件发生时清除OCREF信号 0: OC3REF 不受ETRF输入 (来自TIM1_TRIG引脚) 的影响; 1: 一旦检测到ETRF输入高电平, OC3REF=0。
位6:4	OC3M[2:0]: 输出比较3模式
位3	OC3PE: 输出比较3预装载使能
位2	OC3FE: 输出比较3快速使能
位1:0	CC3S[1:0]: 捕获/比较3选择。 该位定义通道的方向(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TI3FP3上; 10: CC3通道被配置为输入, IC3映射在TI4FP3上; 11: 预留 注: CC3S仅在通道关闭时(TIM1_CCER2寄存器的CC3E=0, CC3NE=0且已被更新)才是可写的。

- **输入捕获模式:**

7	6	5	4	3	2	1	0
		IC3F[3:0]		IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

位7:4	IC3F[3:0]: 输入捕获3滤波器
位3:2	IC3PSC[1:0]: 输入/捕获3预分频器
位1:0	CC3S[1:0]: 捕获/比较3选择。 这2位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC3通道被配置为输出; 01: CC3通道被配置为输入, IC3映射在TI3FP3上; 10: CC3通道被配置为输入, IC3映射在TI4FP3上; 11: 预留 注: CC3S仅在通道关闭时(TIM1_CCER2寄存器的CC3E=0, CC3NE=0且已被更新)才是可写的。

17.7.12 捕获/比较模式寄存器 4(TIM1_CCMR4)

地址偏移值: 0x0B

复位值: 0x00

请参考上述CCMR1寄存器描述。

● 输出比较模式:

7	6	5	4	3	2	1	0
OC4CE		OC4M[2:0]		OC4PE	OC4FE		CC4S[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

位7	OC4CE: 输出比较4清零使能 该位用于使能使用TIM1_TRIG引脚上的外部事件来清通道4的输出信号(OC4REF)，参考 17.5.9 在外部事件发生时清除OCREF信号 0: OC4REF 不受ETRF输入 (来自TIM1_TRIG引脚) 的影响； 1: 一旦检测到ETRF输入高电平，OC4REF=0。
位6:4	OC4M[2:0]: 输出比较4模式
位3	OC4PE: 输出比较4预装载使能
位2	OC4FE: 输出比较4快速使能
位1:0	CC4S[1:0]: 捕获/比较4选择。 该位定义通道的方向(输入/输出)，及输入脚的选择： 00: CC4通道被配置为输出； 01: CC4通道被配置为输入，IC4映射在TI3FP4上； 10: CC4通道被配置为输入，IC4映射在TI4FP4上； 11: 预留 注: CC4S仅在通道关闭时(TIM1_CCER2寄存器的CC4E=0, CC4NE=0且已被更新)才是可写的。

● 输入捕获模式:

7	6	5	4	3	2	1	0
	IC4F[3:0]			IC4PSC[1:0]		CC4S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

位7:4	IC4F[3:0]: 输入捕获4滤波器
位3:2	IC4PSC[1:0]: 输入/捕获4预分频器
位1:0	CC4S[1:0]: 捕获/比较4选择。 该位定义通道的方向(输入/输出)，及输入脚的选择： 00: CC4通道被配置为输出； 01: CC4通道被配置为输入，IC4映射在TI3FP4上； 10: CC4通道被配置为输入，IC4映射在TI4FP4上； 11: 预留 注: CC4S仅在通道关闭时(TIM1_CCER2寄存器的CC4E=0, CC4NE=0且已被更新)才是可写的。

17.7.13 捕获/比较使能寄存器 1(TIM1_CCER1)

地址偏移值: 0x0C

复位值: 0x00

7	6	5	4	3	2	1	0
CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw

位7	CC2NP: 输入捕获/比较2互补输出极性。参考CC1NP的描述。
位6	CC2NE: 输入捕获/比较2互补输出使能。参考CC1NE的描述。
位5	CC2P: 输入捕获/比较2输出极性。参考CC1P的描述。
位4	CC2E: 输入捕获/比较2输出使能。参考CC1E的描述。
位3	<p>CC1NP: 输入捕获/比较1互补输出极性</p> <p>0: OC1N高电平有效; 1: OC1N低电平有效。</p> <p>注1: 一旦LOCK级别(TIM1_BKR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。</p> <p>注2: 对于有互补输出的通道, 该位是预装载的。如果CCPC=1 (TIM1_CR2寄存器), 只有在COM事件发生时, CC1NP位才从预装载位中取新值。</p>
位2	<p>CC1NE: 输入捕获/比较1互补输出使能</p> <p>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。</p> <p>注: 对于有互补输出的通道, 该位是预装载的。如果CCPC=1(TIM1_CR2寄存器), 只有在COM事件发生时, CC1NE位才从预装载位中取新值。</p>
位1	<p>CC1P: 输入捕获/比较1输出极性</p> <p>CC1通道配置为输出:</p> <p>0: OC1高电平有效; 1: OC1低电平有效。</p> <p>CC1通道配置为触发(参考图61):</p> <p>0: 触发发生在T11F的高电平或上升沿; 1: 触发发生在T11F的低电平或下降沿。</p> <p>CC1通道配置为输入(参考图61):</p> <p>0: 捕捉发生在T11F的高电平或上升沿; 1: 捕捉发生在T11F的低电平或下降沿。</p> <p>注1: 一旦LOCK级别(TIM1_BKR寄存器中的LCCK位)设为3或2, 则该位不能被修改。</p> <p>注2: 对于有互补输出的通道, 该位是预装载的。如果CCPC=1 (TIM1_CR2寄存器), 只有在COM事件发生时, CC1P位才从预装载位中取新值。</p>
位0	<p>CC1E: 输入捕获/比较1输出使能</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TIM1_CCR1寄存器。</p> <p>0: 捕获禁止; 0: 捕获使能。</p> <p>注: 对于有互补输出的通道, 该位是预装载的。如果CCPC=1(TIM1_CR2寄存器), 只有在COM事件发生时, CC1E位才从预装载位中取新值。</p>

表34 带刹车功能的互补输出通道OC*i*和OC*iN*的控制位

控制位					输出状态	
MOE位	OSSI位	OSSR位	CC <i>iE</i> 位	CC <i>iNE</i> 位	OC <i>i</i> 输出状态	OC <i>iN</i> 输出状态
1	X	0	0	0	输出禁止(与定时器断开)	输出禁止(与定时器断开)
		0	0	1	输出禁止(与定时器断开)	OC <i>iREF</i> + 极性, OC <i>iN</i> = OC <i>iREF xor CCiP</i>
		0	1	0	OC <i>iREF</i> + 极性, OC <i>i= OCiREF xor CCiP</i>	输出禁止(与定时器断开)
		0	1	1	OC <i>iREF</i> + 极性 + 死区,	OC <i>iREF</i> 反相 + 极性 + 死区,
		1	0	0	输出禁止(与定时器断开)	输出禁止(与定时器断开)
		1	0	1	关闭状态(输出使能且为无效电平)OC <i>i=CCiP</i>	OC <i>iREF</i> + 极性, OC <i>iN</i> = OC <i>iREF xor CCiP</i>
		1	1	0	OC <i>iREF</i> + 极性, OC <i>i= OCiREF xor CCiP</i>	关闭状态(输出使能且为无效电平)OC <i>iN=CCiP</i>
		1	1	1	OC <i>iREF</i> + 极性 + 死区	OC <i>iREF</i> 反相 + 极性 + 死区
0	X	0	X	X	输出禁止(与定时器断开)	
		0			关闭状态(输出使能且为无效电平) 异步地: OC <i>i=CCiP</i> , OC <i>iN=CCiP</i> ;	
		0			然后, 若时钟存在: 经过一个死区时间后OC <i>i=OISi</i> , OC <i>iN=OISiN</i> , 假设OIS <i>i</i> 与OIS <i>iN</i> 并不都对应OC <i>i</i> 和OC <i>iN</i> 的有效电平。	
		0				
		1				
		1				
		1				
		1				

注: 管脚连接到互补的OC*i*和OC*iN*通道的外部I/O管脚的状态, 取决于OC*i*和OC*iN*通道状态和GPIO寄存器。

17.7.14 捕获/比较使能寄存器 2(TIM1_CCER2)

地址偏移值: 0x0D

复位值: 0x00

7	6	5	4	3	2	1	0
预留		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E

rw rw rw rw rw rw rw

位7:6	保留。
位5	CC4P: 输入捕获/比较4输出极性。参考CC1P的描述。
位4	CC4E: 输入捕获/比较4输出使能。参考CC1E 的描述。
位3	CC3NP: 输入捕获/比较3互补输出极性。参考CC1NP的描述。
位2	CC3NE: 输入捕获/比较3互补输出使能。参考CC1NE的描述。
位1	CC3P: 输入捕获/比较3输出极性。参考CC1P的描述。
位0	CC3E: 输入捕获/比较3输出使能。参考CC1E 的描述。

17.7.15 计数器高 8 位(TIM1_CNTRH)

地址偏移值: 0x0E

复位值: 0x00

7 6 5 4 3 2 1 0

CNT[15:8]

RW

RW

RW

RW

RW

RW

RW

RW

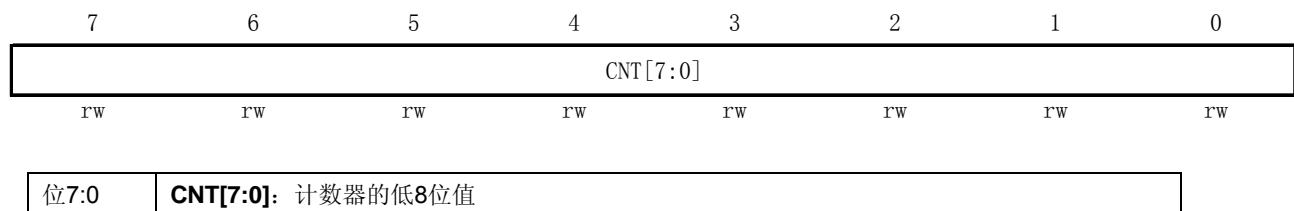
位7:0

CNT[15:8]: 计数器的高8位值

17.7.16 计数器低 8 位(TIM1_CNTRL)

地址偏移值: 0x0F

复位值: 0x00



17.7.17 预分频器高 8 位(TIM1_PSCRH)

地址偏移值: 0x10

复位值: 0x00

7	6	5	4	3	2	1	0
PSC[15:8]							
RW	RW	RW	RW	RW	RW	RW	RW

位7:0	<p>PSC[15:7]: 预分频器的高8位值 预分频器用于对CK_PSC进行分频。 计数器的时钟频率(f_{CK_CNT})等于$f_{CK_PSC}/(PSCR[15:0]+1)$。 PSCR包含了当更新事件产生时装入当前预分频器寄存器的值(更新事件包括计数器被TIM_EGR的UG位清0或被工作在复位模式的从控制器清0)。这意味着为了使新的值起作用，必须产生一个更新事件。</p>
------	---

17.7.18 预分频器低 8 位(TIM1_PSCRL)

地址偏移值: 0x11

复位值: 0x00

7	6	5	4	3	2	1	0
PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

位7:0

PSC[7:0]: 预分频器的低8位值

预分频器用于对CK_PSC进行分频。

计数器的时钟频率(f_{CK_CNT})等于 $f_{CK_PSC}/(PSCR[15:0]+1)$ 。

PSCR包含了当更新事件产生时装入当前预分频器寄存器的值(更新事件包括计数器被TIM_EGR的UG位清0或被工作在复位模式的从控制器清0)。即: 为了使新的值起作用, 必须产生一个更新事件。

17.7.19 自动重装载寄存器高 8 位(TIM1_ARRH)

地址偏移值: 0x12

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

ARR[15:8]

rw							
----	----	----	----	----	----	----	----

位7:0	ARR[15:8]: 自动重装载的高8位值
------	------------------------------

ARR包含了将要装载入实际的自动重装载寄存器的值。

详细请参考 [17.3](#)。

当自动重装载的值为空时，计数器不工作。

17.7.20 自动重装载寄存器低 8 位(TIM1_ARRL)

地址偏移值: 0x13

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

ARR[7:0]

rw							
----	----	----	----	----	----	----	----

位7:0

ARR[7:0]: 自动重装载的低8位值

17.7.21 重复计数寄存器(TIM1_RCR)

地址偏移值: 0x14

复位值: 0xFF

7	6	5	4	3	2	1	0
REP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

位7:0

REP[7:0]: 重复计数器的值

开启了预装载功能后, 这些位允许用户设置比较寄存器的更新速率(即周期性地从预装载寄存器传输到当前寄存器); 如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TIM1_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;

17.7.22 捕获/比较寄存器 1 高 8 位(TIM1_CCR1H)

地址偏移值: 0x15

复位值: 0x00

7	6	5	4	3	2	1	0
CCR1[15:8]							
	rw						

位7:0	CCR1[15:8]: 捕获/比较1的高8位值
	若CC1通道配置为输出(TIM1_CCMR1的CC1S位): 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TIM1_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/比较寄存器的值同计数器TIM1_CNT的值相比较, 并在OC1端口上产生输出信号。 若CC1通道配置为输入: CCR1包含了上一次输入捕获1事件(IC1)发生时的计数器值 (此时该寄存器为只读)。

17.7.23 捕获/比较寄存器 1 低 8 位(TIM1_CCR1L)

地址偏移值: 0x16

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

CCR1[7:0]

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

位7:0	CCR1[7:0]: 捕获/比较1的低8位值
------	-------------------------------

17.7.24 捕获/比较寄存器 2 高 8 位(TIM1_CCR2H)

地址偏移值: 0x17

复位值: 0x00

7	6	5	4	3	2	1	0
CCR2[15:8]							
	rw						

位7:0

CCR2[15:8]: 捕获/比较2的高8位值

若CC2通道配置为输出(TIM1_CCMR2的CC2S位):

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TIM1_CCMR2寄存器(OC2PE位)中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较1寄存器中。

当前捕获/比较寄存器的值同计数器TIM1_CNT的值相比较, 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(此时该寄存器为只读)。

17.7.25 捕获/比较寄存器 2 低 8 位(TIM1_CCR2L)

地址偏移值: 0x18

复位值: 0x00

7	6	5	4	3	2	1	0
CCR2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw
位7:0	CCR2[7:0]: 捕获/比较2的低8位值						

17.7.26 捕获/比较寄存器 3 高 8 位(TIM1_CCR3H)

地址偏移值: 0x19

复位值: 0x00

7	6	5	4	3	2	1	0
CCR3[15:8]							
	rw						

位7:0	CCR3[15:8]: 捕获/比较3的高8位值
	<p>若CC3通道配置为输出(TIM1_CCMR3的CC3S位): 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TIM1_CCMR3寄存器(OC3PE位)中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较1寄存器中。 当前捕获/比较寄存器的值同计数器TIM1_CNT的值相比较, 并在OC3端口上产生输出信号。</p> <p>若CC3通道配置为输入: CCR3包含了由上一次输入捕获3事件(IC3)传输的计数器值(此时该寄存器为只读)。</p>

17.7.27 捕获/比较寄存器 3 低 8 位(TIM1_CCR3L)

地址偏移值: 0x1A

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

CCR3[7:0]

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

位7:0	CCR3[7:0]: 捕获/比较3的低8位值
------	------------------------

17.7.28 捕获/比较寄存器 4 高 8 位(TIM1_CCR4H)

地址偏移值: 0x1B

复位值: 0x00

7	6	5	4	3	2	1	0
CCR4[15:8]							
	rw						

位7:0	<p>CCR4[15:8]: 捕获/比较4的高8位值</p> <p>若CC4通道配置为输出(TIM1_CCMR4的CC4S位):</p> <p>CCR4包含了装入当前捕获/比较4寄存器的值(预装载值)。</p> <p>如果在TIM1_CCMR4寄存器(OC4PE位)中未选择预装载功能, 写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时, 此预装载值才传输至当前捕获/比较1寄存器中。</p> <p>当前捕获/比较寄存器的值同计数器TIM1_CNT的值相比较, 并在OC4端口上产生输出信号。</p> <p>若CC4通道配置为输入:</p> <p>CCR4包含了由上一次输入捕获4事件(IC4)传输的计数器值(此时该寄存器为只读)。</p>
------	---

17.7.29 捕获/比较寄存器 4 低 8 位(TIM1_CCR4L)

地址偏移值: 0x1C

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

CCR4[7:0]

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

位7:0	CCR4[7:0]: 捕获/比较4的低8位值
------	-------------------------------

17.7.30 刹车寄存器(TIM1_BKR)

地址偏移值: 0x1D

复位值: 0x00

7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI		LOCK
rw	rw	rw	rw	rw	rw	rw	rw

位7	MOE: 主输出使能 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, 该位可以由软件置1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(TIM1_CCERX寄存器的CCIE位), 则使能OC和OCN输出。 有关OC/OCN使能的细节, 参见 17.7.13 。
位6	AOE: 自动输出使能 0: MOE只能被软件置1; 1: MOE能被软件置1或在下一个更新事件被自动置1(如果刹车输入无效)。 注: 一旦LOCK级别(TIM1_BKR寄存器中的LOCK位)设为1, 则该位不能被修改。
位5	BKP: 刹车输入极性 0: 刹车输入低电平有效; 1: 刹车输入高电平有效。 注: 一旦LOCK级别(TIM1_BKR寄存器中的LOCK位)设为1, 则该位不能被修改。
位4	BKE: 刹车功能使能 0: 禁止刹车输入(BRK); 1: 开启刹车输入(BRK)。 注: 一旦LOCK级别(TIM1_BKR寄存器中的LOCK位)设为1, 则该位不能被修改。
位3	OSSR: 运行模式下“关闭状态”选择 该位用于当MOE=1且通道为互补输出时。 参考OC/OCN使能的详细说明(参见 17.7.13)。 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CCIE=1或CCNE=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(TIM1_BKR寄存器中的LOCK位)设为2, 则该位不能被修改。
位2	OSSI: 空闲模式下“关闭状态”选择 该位用于当MOE=0且通道设为输出时。 参考OC/OCN使能的详细说明(参见 17.7.13)。 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CCIE=1或CCNE=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(TIM1_BKR寄存器中的LOCK位)设为2, 则该位不能被修改。
位1:0	LOCK[1:0]: 锁定设置 该位为防止软件错误而提供写保护。 00: 锁定关闭, 寄存器无写保护; 01: 锁定级别1, 不能写入TIM1_BKR寄存器的BKE、BKP、AOE位和TIM1_OISR寄存器的OIS1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CCIS位设为输出, CC极性位是TIM1_CCERX寄存器的CCP1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CCIS位设为输出, CC控制位是TIM1_CCMRx寄存器的OCM1/OCPE位); 注: 在系统复位后, 只能写一次LOCK位, 一旦写入TIM1_BDR寄存器, 则其内容保持不变直至复位。

注：由于BKE、BKP、AOE、OSSR、OSSI位可被锁定（依赖于LOCK位），因此在第一次写TIM1_BKR寄存器时必须对它们进行设置。

17.7.31 死区寄存器(TIM1_DTR)

地址偏移值：0x1E

复位值：0x00

7	6	5	4	3	2	1	0
DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

位7:0	<p>UTG[7:0]: 死区发生器设置</p> <p>这些位定义了插入互补输出之间的死区持续时间。假设DT表示其持续时间，t_{CK_PSC}为TIM1的时钟脉冲：</p> <p>DTG[7:5]=0xx => DT=DTG[7:0]x tdtg，其中：$tdtg=t_{CK_PSC}$. (f1)</p> <p>DTG[7:5]=10x => DT=(64+DTG[5:0])x tdtg，其中：$tdtg=t_{CK_PSC}$. (f2)</p> <p>DTG[7:5]=110 => DT=(32+DTG[4:0])x tdtg，其中：$tdtg=8x t_{CK_PSC}$. (f3)</p> <p>DTG[7:5]=111 => DT=(32+DTG[4:0])x tdtg，其中：$tdtg=16x t_{CK_PSC}$. (f4)</p> <p>举例：</p> <p>如果$t_{CK_PSC} = 125$ ns (8 MHz)，可能的死区时间为：</p> <p>DTG[7:0] = 0 到 7Fh，0 到 15875 ns，步长时间为125 ns (参考f1),</p> <p>DTG[7:0] = 80h 到 BFh，16 μs 到 31750 ns，步长时间为250 ns (参考f2),</p> <p>DTG[7:0] = C0h 到 DFh，32 μs 到 63 μs，步长时间为1μs (参考f3),</p> <p>DTG[7:0] = E0h 到 FFh，64 μs 到 126 μs，步长时间为2 μs (参考f4),</p> <p>注：一旦LOCK级别(TIM1_BKR寄存器中的LOCK位)设为1、2或3，则不能修改这些位。</p>
------	---

17.7.32 输出空闲状态寄存器(TIM1_OISR)

地址偏移值: 0x1F

复位值: 0x00

7	6	5	4	3	2	1	0
保留	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1

rw rw rw rw rw rw rw

位7	保留, 始终读为0。
位6	OIS4: 输出空闲状态4(OC4输出)。参见OIS1位。
位5	OIS3N: 输出空闲状态3(OC3N输出)。参见OIS1N位。
位4	OIS3: 输出空闲状态3(OC3输出)。参见OIS1位。
位3	OIS2N: 输出空闲状态2(OC2N输出)。参见OIS1N位。
位2	OIS2: 输出空闲状态2(OC2输出)。参见OIS1位。
位1	OIS1N: 输出空闲状态1(OC1N输出)。 0: 当MOE=0时, 则在一个死区时间后, OC1N=0; 1: 当MOE=0时, 则在一个死区时间后, OC1N=1。 注: 已经设置了LOCK(TIM1_BKR寄存器)级别1、2或3后, 该位不能被修改。
位0	OIS1: 输出空闲状态1(OC1输出)。 0: 当MOE=0时, 如果OC1N使能, 则在一个死区后, OC1=0; 1: 当MOE=0时, 如果OC1N使能, 则在一个死区后, OC1=1。 注: 已经设置了LOCK(TIM1_BKR寄存器)级别1、2或3后, 该位不能被修改。

17.7.33 TIM1 寄存器图

表35 TIM1寄存器图

地址	寄存器	7	6	5	4	3	2	1	0
00 5250h	TIM1_CR1	ARPE 0	CMS1 0	CMS0 0	DIR 0	OPM 0	URS 0	UDIS 0	CEN 0
	复位值								
00 5251h	TIM1_CR2	TI1S 0	MMS2 0	MMS1 0	MMS0 0	- 0	COMS 0	- 0	CCPC 0
	复位值								
00 5252h	TIM1_SMCR	MSM 0	TS2 0	TS1 0	TS0 0	- 0	SMS2 0	SMS1 0	SMS0 0
	复位值								
00 5253h	TIM1_ETR	ETP 0	ECE 0	ETPS1 0	ETPS0 0	EFT3 0	EFT2 0	EFT1 0	EFT0 0
	复位值								
00 5254h	TIM1_IER	BIE 0	TIE 0	COMIE 0	CC4IE 0	CC3IE 0	CC2IE 0	CC1IE 0	UIE 0
	复位值								
00 5255h	TIM1_SR1	BIF 0	TIF 0	COMIF 0	CC4IF 0	CC3IF 0	CC2IF 0	CC1IF 0	UIF 0
	复位值								
00 5256h	TIM1_SR2	- 0	- 0	- 0	CC4OF 0	CC3OF 0	CC2OF 0	CC1OF 0	- 0
	复位值								
00 5257h	TIM1_EGR	BG 0	TG 0	COMG 0	CC4G 0	CC3G 0	CC2G 0	CC1G 0	UG 0
	复位值								
00 5258h	TIM1_CCMR1	OC1CE 0	OC1M2 0	OC1M1 0	OC1M0 0	OC1PE 0	OC1FE 0	CC1S1 0	CC1S0 0
	复位值								
00 5259h	TIM1_CCMR1	IC1F3 0	IC1F2 0	IC1F1 0	IC1F0 0	IC1PSC1 0	IC1PSC0 0	CC1S1 0	CC1S0 0
	复位值								
00 525Ah	TIM1_CCMR2	OC2CE 0	OC2M2 0	OC2M1 0	OC2M0 0	OC2PE 0	OC2FE 0	CC2S1 0	CC2S0 0
	复位值								
00 525Bh	TIM1_CCMR2	IC2F3 0	IC2F2 0	IC2F1 0	IC2F0 0	IC2PSC1 0	IC2PSC0 0	CC2S1 0	CC2S0 0
	复位值								
00 525Ch	TIM1_CCMR3	OC3CE 0	OC3M2 0	OC3M1 0	OC3M0 0	OC3PE 0	OC3FE 0	CC3S1 0	CC3S0 0
	复位值								
00 525Dh	TIM1_CCER2	- 0	- 0	- 0	CC4P 0	CC4E 0	CC3NP 0	CC3NE 0	CC3P 0
	复位值								
00 525Eh	TIM1_CNTRH	CNT15 0	CNT14 0	CNT13 0	CNT12 0	CNT11 0	CNT10 0	CNT9 0	CNT8 0
	复位值								
00 525Fh	TIM1_CNTRL	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT3 0	CNT1 0	CNT0 0
	复位值								
00 5260h	TIM1_PSCRH	PSC15 0	PSC14 0	PSC13 0	PSC12 0	PSC11 0	PSC10 0	PSC9 0	PSC8 0
	复位值								
00 5261h	TIM1_PSCRL	PSC7 0	PSC6 0	PSC5 0	PSC4 0	PSC3 0	PSC2 0	PSC1 0	PSC0 0
	复位值								

地址	寄存器							
00 5262h	TIM1_ARRH	ARR15 1	ARR14 1	ARR13 1	ARR12 1	ARR11 1	ARR10 1	ARR9 1
	复位值							
00 5263h	TIM1_ARRL	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1
	复位值							
00 5264h	TIM1_RCR	RCR7 0	RCR6 0	RCR5 0	RCR4 0	RCR3 0	RCR2 0	RCR1 0
	复位值							
00 5265h	TIM1_CCR1H	CCR115 0	CCR114 0	CCR113 0	CCR112 0	CCR111 0	CCR110 0	CCR19 0
	复位值							
00 5266h	TIM1_CCR1L	CCR17 0	CCR16 0	CCR15 0	CCR14 0	CCR13 0	CCR12 0	CCR11 0
	复位值							
00 5267h	TIM1_CCR2H	CCR215 0	CCR214 0	CCR213 0	CCR212 0	CCR211 0	CCR210 0	CCR29 0
	复位值							
00 5268h	TIM1_CCR2L	CCR27 0	CCR26 0	CCR25 0	CCR24 0	CCR23 0	CCR22 0	CCR21 0
	复位值							
00 5269h	TIM1_CCR3H	CCR315 0	CCR314 0	CCR313 0	CCR312 0	CCR311 0	CCR310 0	CCR39 0
	复位值							
00 526Ah	TIM1_CCR3L	CCR37 0	CCR36 0	CCR35 0	CCR34 0	CCR33 0	CCR32 0	CCR31 0
	复位值							
00 526Bh	TIM1_CCR4H	CCR415 0	CCR414 0	CCR413 0	CCR412 0	CCR411 0	CCR410 0	CCR49 0
	复位值							
00 526Ch	TIM1_CCR4L	CCR47 0	CCR46 0	CCR45 0	CCR44 0	CCR43 0	CCR42 0	CCR41 0
	复位值							
00 526Dh	TIM1_BKR	MOE 0	AOE 0	BKP 0	BKE	OSSR 0	OSSI 0	LOCK 0
	复位值							
00 526Eh	TIM1_DTR	DTG7 0	DTG6 0	DTG7 0	DTG4 0	DTG3 0	DTG2 0	DTG1 0
	复位值							
00 526Fh	TIM1_OISR	- 0	OIS4 0	OIS3N 0	OIS3 0	OIS2N 0	OIS2 0	OIS1N 0
	复位值							

18 16位通用定时器(TIM2, TIM3, TIM5)

18.1 介绍

本章介绍了通用定时器TIM2, TIM3和TIM5, 其中TIM2有3个通道, TIM3有2个通道, TIM5与TIM2类似但带有两个额外的寄存器, 用于定时器的同步和级联。

通用定时器由带有可编程预分频器的16位自动装载计数器构成。

它适用于多种场合, 包括:

- 基本的定时
- 测量输入信号的脉冲长度(输入捕获)
- 产生输出波形(输出比较, PWM和单脉冲)
- 与其他定时器或外部信号同步(外部时钟, 复位, 触发和使能信号)(仅针对带有TIM5的芯片)
定时器可由内部时钟驱动。

本章仅介绍了通用定时器的主要功能, 更多的功能信息请参考 [16位高级控制定时器\(TIM1\)](#)。

18.2 TIM2/TIM3的主要功能

TIM2/TIM3的功能包括:

- 16位向上计数和自动装载计数器
- 4位可编程(可以实时修改的)预分频器, 计数器时钟频率的分频系数为1~32768之间的2的幂
- 3个独立通道:
 - 输入捕获
 - 输出比较
 - PWM生成(边缘对齐模式)
 - 单脉冲模式输出
- 如下事件发生时产生中断:
 - 更新: 计数器向上溢出, 计数器初始化(通过软件)
 - 输入捕获
 - 输出比较

18.3 TIM5主要功能

TIM5的功能包括:

- 16位向上计数和自动装载计数器
- 4位可编程(可以实时修改的)预分频器, 计数器时钟频率的分频系数为值为1~32768之间的2的幂
- 3个独立通道:
 - 输入捕获
 - 输出比较
 - PWM生成(边缘对齐模式)
 - 单脉冲模式输出
- 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)
 - 输入捕获
 - 输出比较

18.4 TIM2/TIM3/TIM5功能概述

图79 TIM2/TIM3框图

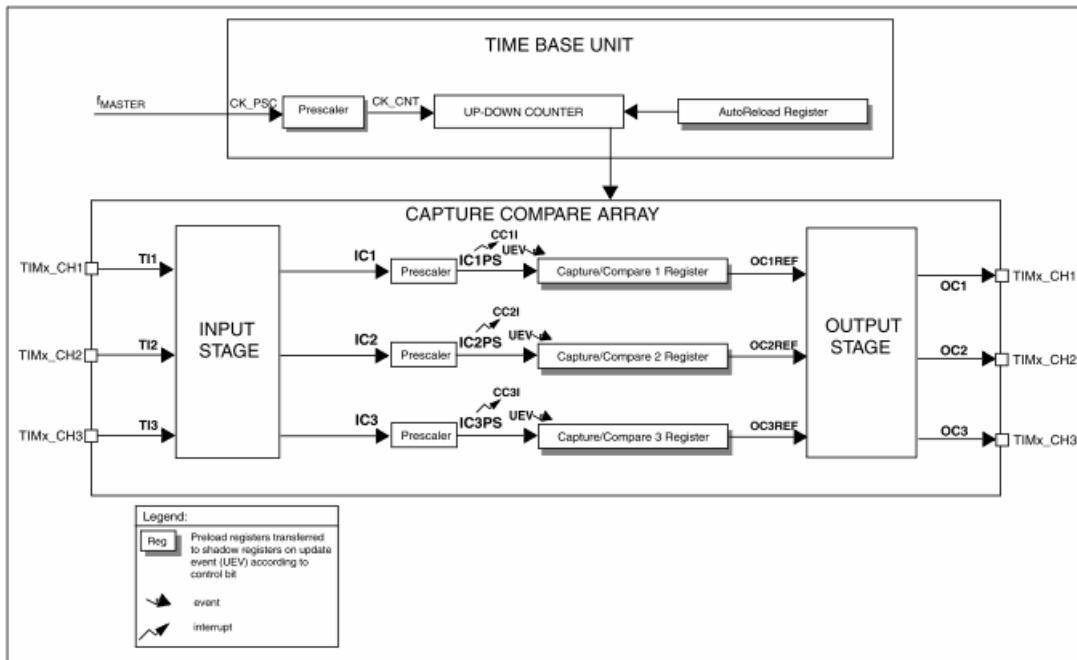
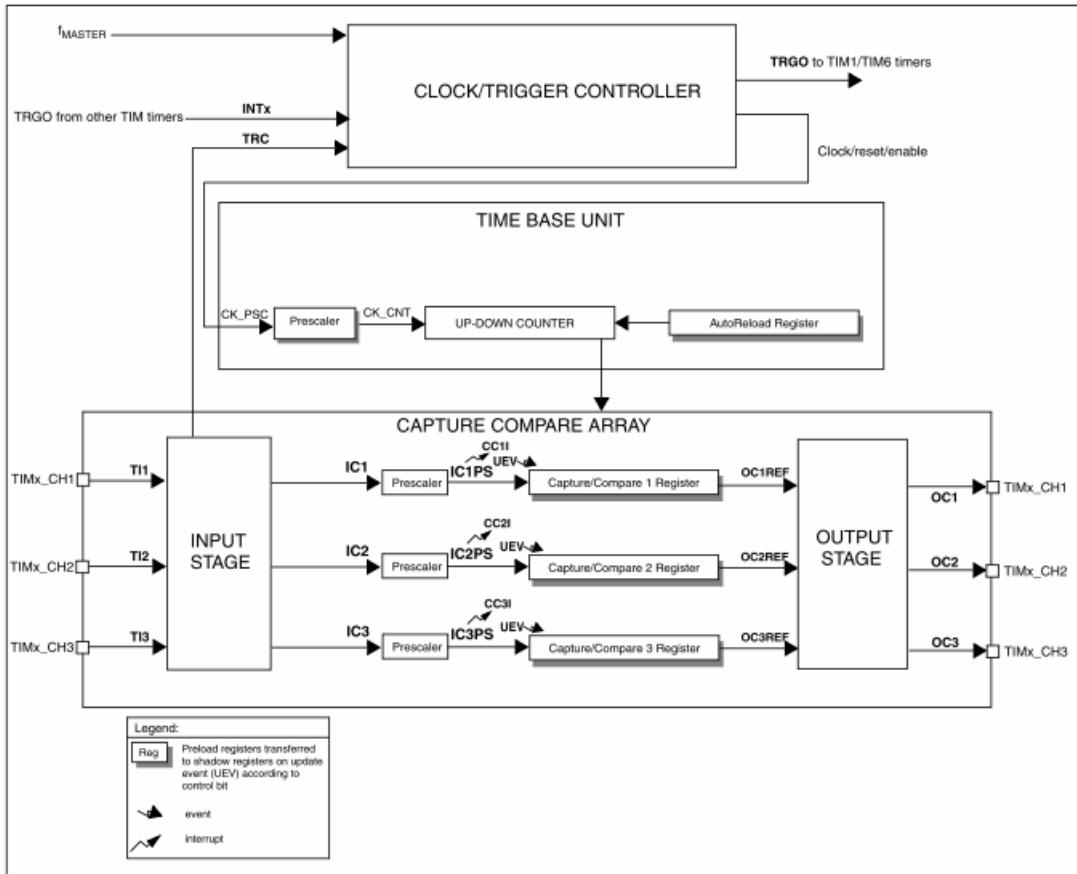


图80 TIM5框图



18.4.1 时基单元

时基单元包含：

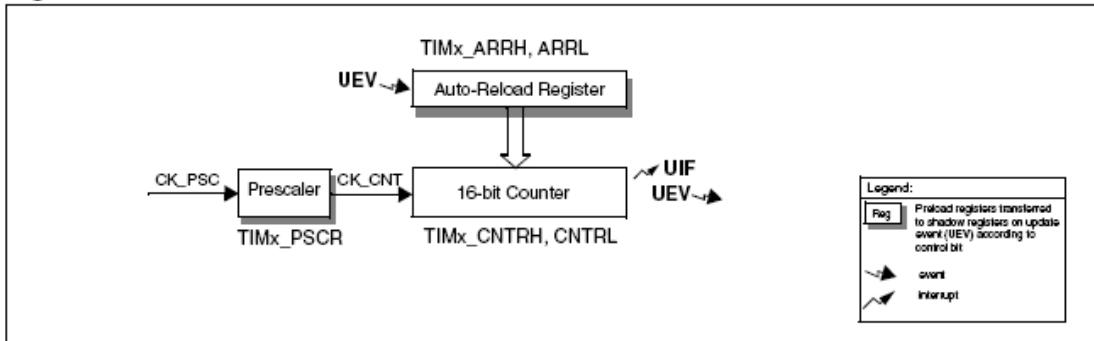
- 16位向上计数器

- 预分频器
- 16位自动装载寄存器

没有重复寄存器。

计数器使用内部时钟(f_{MASTER})，它由CK_PSC提供，并经过预分频器分频产生计数器时钟CK_CNT。

图81 时基单元



更多信息请参考 [17.3](#)。

预分频器

预分频器的实现：

- 预分频器基于4位寄存器控制的16位计数器，由于寄存器带有缓冲器因此可以随时修改预分频的数值。计数器可以取值为1到32768之间的2的幂进行分频。

计数器时钟频率的计算公式：

$$f_{CK_CNT} = f_{CK_PSC} / 2^{(PSCR[3:0])}$$

预分频器的值由预装载寄存器写入。一旦写入预装载寄存器的LS字节时，带有当前使用值的影子寄存器就被写入了新的值。

新的预分频值在下一个周期时生效(在下一个更新事件之后)。

对TIMx_PSCR寄存器的读操作通过预装载寄存器实现，因此可以随时读取不受限制。

计数器的操作

请参考 [17.3.4](#)。

18.4.2 时钟/触发控制器

时钟/触发控制器以及相应的TIMx_CR2和TIMx_SMCR寄存器仅存在于TIM5中，更多信息请请参考 [17.4](#)。

18.4.3 捕获/比较通道

输入部分

请参考 [17.5](#)。

如 [图82](#)输入部分框图所示，定时器带有两个输入通道，通道1在内部链接到比较器。

图82 输入部分框图

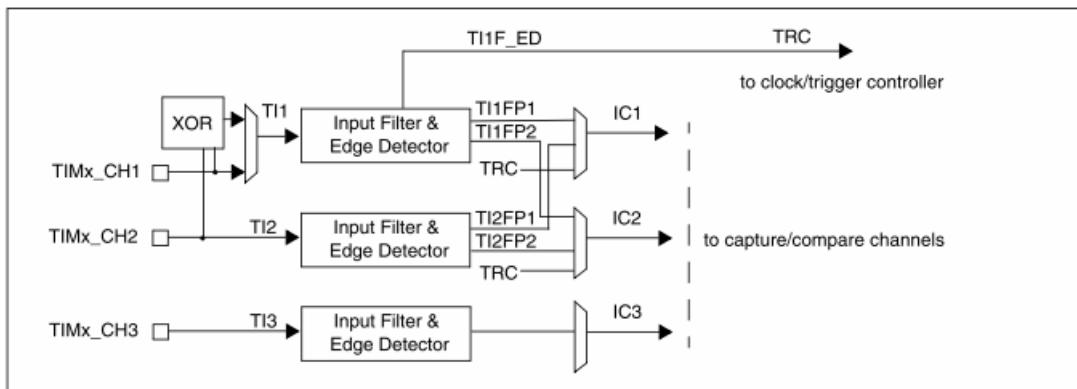
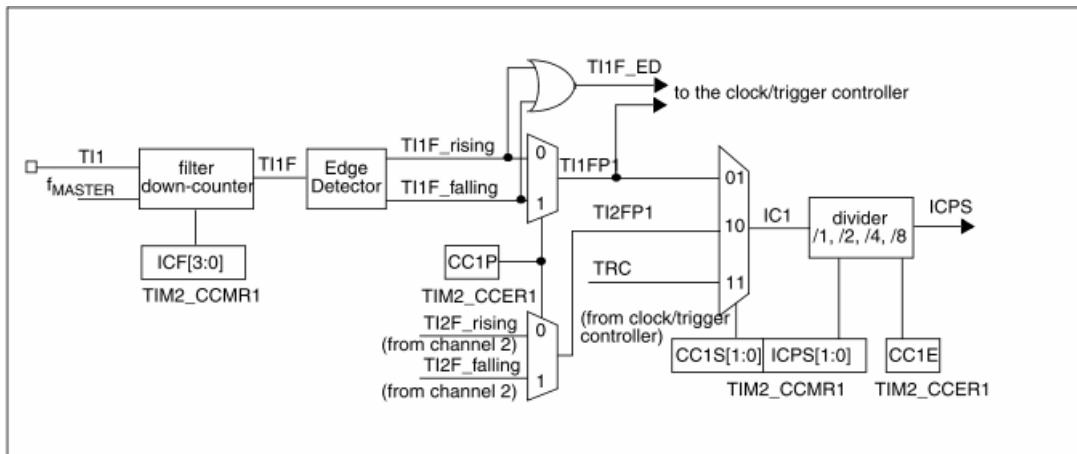


图83 TIM2通道1的输入部分框图

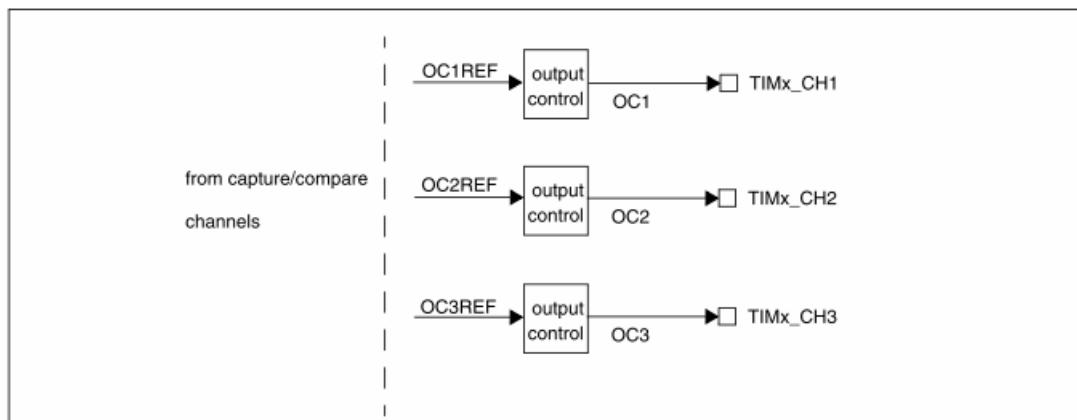


输出部分

详细信息请参考 [17.5.4输出模块](#), [17.5.5强制输出模式](#), 和 [17.5.7PWM模式](#)。

如图所示通用定时器不带死区控制和互补输出。

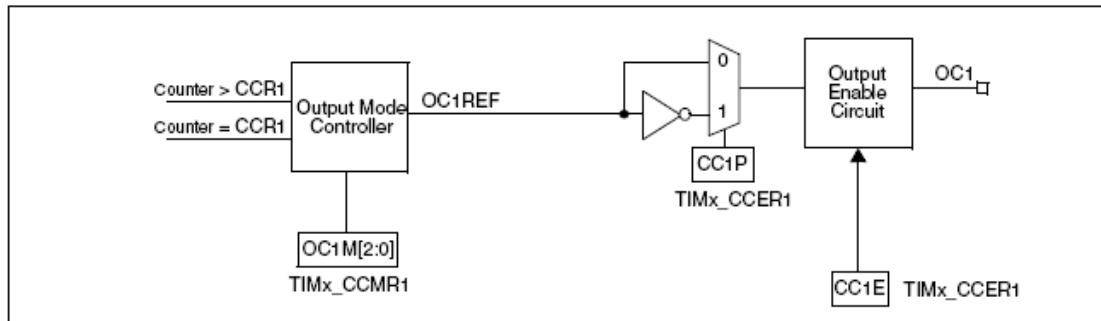
图84 输出部分框图



输出部分产生内部的波形到参考信号OCxREF(高有效), 极性的判断在最后进行。(参考 [图85](#))

通道1的输出部分框图)

图85 通道1的输出部分框图



18.5 中断

通用定时器包括4个中断源：

- 捕获/比较3中断
- 捕获/比较2中断
- 捕获/比较1中断
- 更新中断

在使用中断功能时，需要先设置TIMx_IER寄存器的CC3IE位或CC2IE位或CC1IE位使能中断请求。

通过软件设置TIMx_EGR寄存器的相应位也能产生不同的中断源。

18.6 TIM2/TIM3/TIM5寄存器

18.6.1 控制寄存器 1(TIMx_CR1)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
ARPE	保留		OPM	URS	UDIS	CEN	

rw rw rw rw rw

位7	ARPE: 自动重装载预装载允许位 0: TIMx_ARR寄存器没有预装载寄存器可以缓冲；可以直接对其进行写操作。 1: TIMx_ARR寄存器通过预装载寄存器可以缓冲。
位6:4	保留
位3	OPM: 单脉冲模式 0: 在发生更新事件时，计数器不停止； 1: 在发生下一次更新事件(清除CEN位)时，计数器停止。
位2	URS: 更新请求源 0: 当更新请求使能时，只要寄存器被更新了就产生更新中断。 1: 当更新请求使能时，只有计数器溢出才产生更新中断。
位1	UDIS: 禁止更新 软件通过该位允许/禁止UEV事件的产生 0: 只要计数器溢出，或者产生了软件更新，或者通过时钟/触发模式控制器产生了硬件复位，就产生更新事件。 1: 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则计数器和预分频器被重新初始化。
位0	CEN: 使能计数器 0: 禁止计数器； 1: 使能计数器。

18.6.2 控制寄存器 2(TIM5_CR2)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
保留	MMS[2:0]			保留			
rw	rw	rw	rw				

注意: 该寄存器只在TIM5才有, 参考表38 TIM5 – 寄存器图。

位7	保留, 读为0。
位6:4	<p>MMS: 主模式选择</p> <p>这两位用于选择在主模式下送到TIM1和TIM2的同步信息(TRGO)。可能的组合如下:</p> <p>000: 复位 – TIM3_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)。用来同时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TIM3_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出(TRGO)。</p> <p>011: 保留</p> <p>100: 保留</p> <p>101: 保留</p> <p>111: 保留</p>
位3:0	保留, 始终读为0。

18.6.3 触发从模式控制寄存器(TIM5_SMCR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
MSM	TS[2:0]			保留	SMS[2:0]		
rw	rw	rw	rw		rw	rw	rw

注意: 该寄存器只在TIM5有, 参考表38TIM – 寄存器图。

位7	MSM: 主/从模式 0: 无作用; 1: 触发输入(TRGI)上的事件被延迟了, 以允许在当前定时器(通过TRGO)与它的从定时器间的完美同步。
位6:4	TS[2:0]: 触发选择 这些位域用于选择同步计数器的触发输入。 000: 内部触发ITR0连至TIM5的TRGO 001: 保留 010: 保留 011: 内部触发ITR3连至TIM1的TRGO 100: 保留 101: 保留 110: 保留 111: 保留 注: 这些位只能在未用到(如SMS=000)时被改变, 以避免在改变时产生错误的边沿检测。
位3	保留, 始终读为0。
位2:0	SMS: 时钟/触发/从模式选择 当选择了外部信号, 触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000: 关闭时钟/触发控制器 – 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: 触发复位模式 – 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。

18.6.4 中断使能寄存器(TIMx_IER)

地址偏移值: 0x01(TIM2/TIM3), 0x03(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TIE	保留		CC3IE	CC2IE	CC1IE	UIE

位7	保留
位6	TIE: 触发中断使能 0: 触发中断禁用 1: 触发中断使能
位5:4	保留
位3	CC3IE: 允许捕获/比较3中断 0: 禁止捕获/比较3中断; 1: 允许捕获/比较3中断。
位2	CC2IE: 允许捕获/比较2中断 0: 禁止捕获/比较2中断; 1: 允许捕获/比较2中断。
位1	CC1IE: 允许捕获/比较1中断 0: 禁止捕获/比较1中断; 1: 允许捕获/比较1中断。
位0	UIE: 允许更新中断 0: 禁止更新中断; 1: 允许更新中断。

18.6.5 状态寄存器 1(TIMx_SR1)

地址偏移值: 0x02(TIM2/TIM3), 0x04(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TIF	保留		CC3IF	CC2IF	CC1IF	UIF
rc_w0				rc_w0	rc_w0	rc_w0	rc_w0

位7	保留
位6	<p>TIF: 触发中断标志 当产生触发事件时该位由硬件置1(在TRGI信号上检测到有效的触发沿, 当选择门控模式时, 上升及下降沿都有效)。它由软件清0。</p> <p>0: 没有触发事件发生。 1: 触发中断悬挂。</p> <p>注意: 在TIM2/TIM3中该位保留。</p>
位5:4	保留
位3	<p>CC3IF: 捕获/比较3 中断标志 参考CC1IF描述。</p>
位2	<p>CC2IF: 捕获/比较2 中断标志 参考CC1IF描述。</p>
位1	<p>CC1IF: 捕获/比较1 中断标志 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, 由软件清0。</p> <p>0: 无匹配发生; 1: TIMx_CNT的值与TIMx_CCR1的值匹配。</p> <p>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_CCR1L清0。</p> <p>0: 无输入捕获产生; 1: 计数器值已被捕获(拷贝)至TIMx_CCR1(在IC1上检测到与所选极性相同的边沿)。</p>
位0	<p>UIF: 更新中断标志 当产生更新事件时该位由硬件置1。它由软件清0。</p> <p>0: 无更新事件产生; 1: 更新事件等待响应。当寄存器被更新时该位由硬件置1:</p> <ul style="list-style-type: none"> - 若TIMx_CR1寄存器的UDIS=0, 计数器溢出; - 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);

18.6.6 状态寄存器 2(TIMx_SR2)

地址偏移值: 0x03(TIM2/TIM3), 0x05(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

保留	CC3OF rc_w0	CC2OF rc_w0	CC1OF rc_w0	保留
----	----------------	----------------	----------------	----

位7:4	保留
位3	CC3OF: 捕获/比较3 过捕获标志 参考CC1OF描述。
位2	CC2OF: 捕获/比较2 过捕获标志 参考CC1OF描述。
位1	CC1OF: 捕获/比较1 过捕获标志 只有当对应通道配置为输入捕获模式下, 才会被硬件置位; 软件写0清除该位。 0: 没有检测到过捕获; 1: CC1IF标志已经置位的情况下, 计数器的值又被捕获到TIMx_CCR1寄存器中。
位0	保留, 硬件强制为0

18.6.7 事件产生寄存器(TIMx_EGR)

地址偏移值: 0x04(TIM2/TIM3), 0x06(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TG	保留		CC3G	CC2G	CC1G	UG

位7	保留
位6	TG: 产生触发事件 该位由软件置1，用于产生一个触发事件，由硬件自动清0。 0: 无动作。 1: TIMx_SR1的TIF标志被置1。如果TIE为1则产生一个中断。 注意：在TIM2/TIM3中，该位保留。
位5:4	保留
位3	CC3G: 产生捕获/比较3事件 参考CC1G描述。
位2	CC2G: 产生捕获/比较2事件 参考CC1G描述。
位1	CC1G: 产生捕获/比较1事件 该位由软件置1，用于产生一个捕获/比较事件，由硬件自动清0。 0: 无动作； 1: 在通道CC1上产生一个捕获/比较事件： 若通道CC1配置为输出： 设置CC1IF=1，若开启对应的中断，则产生相应的中断。 若通道CC1配置为输入： 当前的计数器值捕获至TIMx_CCR1寄存器，设置CC1IF=1，若开启对应的中断，则产生相应的中断。若CC1IF已经为1，则设置CC1OF=1。
位0	UG: 产生更新事件 该位由软件置1，由硬件自动清0。 0: 无动作； 1: 重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清0。

18.6.8 捕获/比较模式寄存器 1(TIMx_CCMR1)

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CC1S定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx描述了通道在输出模式下的功能，ICxx描述了通道在输出模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

地址偏移值: 0x05(TIM2/TIM3), 0x07(TIM5)

复位值: 0x00

- ### ● 通道配置为输出模式:

7	6	5	4	3	2	1	0
保留		OC1M[2:0]		OC1PE	保留		CC1S[1:0]

位7	保留
位6:4	<p>0C1M: 输出比较1模式</p> <p>这些位定义了输出参考信号OC1REF的动作，而OC1REF决定了OC1的值。OC1REF是高电平有效，而OC1的有效电平取决于CC1P位。</p> <p>000: 冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用；</p> <p>001: 匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为高。</p> <p>010: 匹配时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1(TIMx_CCR1)相同时，强制OC1REF为低。</p> <p>011: 翻转。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。</p> <p>100: 强制为无效电平。强制OC1REF为低。</p> <p>101: 强制为有效电平。强制OC1REF为高。</p> <p>110: PWM模式1— 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。</p> <p>111: PWM模式2— 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。</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。细节请参考17.5.7PWM模式。</p>
位3	<p>OC1PE: 输出比较1预装载使能</p> <p>0: 禁止TIMx_CCR1寄存器的预装载功能，可随时写入TIMx_CCR1寄存器，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载入影子寄存器中。</p> <p>注：为了保证正确的操作，当工作于PWM模式时，预装载寄存器必须使能；在单脉冲模式下(TIMx_CR1寄存器的OPM=1)，这点不用强求。</p>
位2	保留
位1:0	<p>CC1S: 捕获/比较1选择。</p> <p>这2位定义通道的方向(输入/输出)，及输入脚的选择：</p> <p>00: CC1通道被配置为输出；</p> <p>01: CC1通道被配置为输入，IC1映射在TI1FP1上；</p> <p>10: CC1通道被配置为输入，IC1映射在TI2FP1上；</p> <p>11: 保留。</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>

● 通道配置为输入模式:

7	6	5	4	3	2	1	0
				IC1PSC[1:0]		CC1S[1:0]	
RW	RW	RW	RW	RW	RW	RW	RW

位7:4	<p>IC1F: 输入捕获1滤波器</p> <p>这几位定义了TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，只有发生了N个事件后输出的跳变才被认为有效：</p> <p>0000: 无滤波器，以f_{MASTER}采样</p> <p>0001: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=2</p> <p>0010: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=4</p> <p>0011: 采样频率$f_{SAMPLING}=f_{MASTER}$, N=8</p> <p>0100: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=6</p> <p>0101: 采样频率$f_{SAMPLING}=f_{MASTER}/2$, N=8</p> <p>0110: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=6</p> <p>0111: 采样频率$f_{SAMPLING}=f_{MASTER}/4$, N=8</p> <p>1000: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=6</p> <p>1001: 采样频率$f_{SAMPLING}=f_{MASTER}/8$, N=8</p> <p>1010: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=5</p> <p>1011: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=6</p> <p>1100: 采样频率$f_{SAMPLING}=f_{MASTER}/16$, N=8</p> <p>1101: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=5</p> <p>1110: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=6</p> <p>1111: 采样频率$f_{SAMPLING}=f_{MASTER}/32$, N=8</p>
位3:2	<p>IC1PSC: 输入/捕获1预分频器</p> <p>这2位定义了CC1输入(IC1)的预分频系数。</p> <p>一旦CC1E=0(TIMx_CCER寄存器中)，则预分频器复位。</p> <p>00: 无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获；</p> <p>01: 每2个事件触发一次捕获；</p> <p>10: 每4个事件触发一次捕获；</p> <p>11: 每8个事件触发一次捕获。</p> <p>注：IC1PSC动态改变时不会复位内部的事件计数器。这样的话，在下次捕获之前都用旧的值。如果要立即采用新的值，可以清零CC1E位，并且再将它置位。</p>
位1:0	<p>CC1S: 捕获/比较1选择。</p> <p>这2位定义通道的方向(输入/输出)，及输入脚的选择：</p> <p>00: CC1通道被配置为输出；</p> <p>01: CC1通道被配置为输入，IC1映射在TI1FP1上；</p> <p>10: CC1通道被配置为输入，IC1映射在TI2FP1上；</p> <p>11: 保留。</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>

18.6.9 捕获/比较模式寄存器 2(TIMx_CCMR2)

注意：具体如何使用这些位，请参考[捕获/比较模式寄存器1\(TIMx_CCMR1\)](#)。

地址偏移值：0x06(TIM2/TIM3), 0x08(TIM5)

复位值：0x00

● 通道配置为输出模式：

7	6	5	4	3	2	1	0
保留		OC2M[2:0]		OC2PE	保留		CC2S[1:0]
		rw		rw			rw

位7	保留
位6:4	OC2M: 输出比较2模式
位3	OC2PE: 输出比较2预装载使能
位2	保留
位1:0	<p>CC2S: 捕获/比较1 选择。 这2位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC2通道被配置为输出; 01: CC2通道被配置为输入, IC1映射在TI2FP2上; 10: CC2通道被配置为输入, IC1映射在TI1FP2上; 11: 保留。 注: CC2S仅在通道关闭时(TIMx_CCER1寄存器的CC2E=0)才是可写的。</p>

● 通道配置为输入模式：

7	6	5	4	3	2	1	0
		IC2F[3:0]		IC2PSC[1:0]		CC2S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

位7:4	IC2F: 输入捕获2滤波器
位3:2	IC2PSC: 输入/捕获2预分频器
位1:0	<p>CC2S: 捕获/比较2选择。 这2位定义通道的方向(输入/输出), 及输入脚的选择: 00: CC1通道被配置为输出; 01: CC1通道被配置为输入, IC2映射在TI2FP2上; 10: CC1通道被配置为输入, IC2映射在TI1FP2上; 11: 保留。 注: CC2S仅在通道关闭时(TIMx_CCER1寄存器的CC2E=0)才是可写的。</p>

18.6.10 捕获/比较模式寄存器 3(TIMx_CCMR3)

地址偏移值: 0x07(TIM2/TIM3), 0x09(TIM5)

复位值: 0x00

注意: 具体如何使用这些位, 参考[18.6.8](#)。

● 通道配置为输出模式:

7	6	5	4	3	2	1	0
保留		OC3M[2:0]		OC3PE	保留		CC3S[1:0]

rw

rw

rw

注意: TIM3没有该寄存器。

位7	保留
位6:4	OC3M: 输出比较3模式
位3	OC3PE: 输出比较3预装载使能
位2	保留
位1:0	<p>CC3S: 捕获/比较3选择。</p> <p>这2位定义通道的方向(输入/输出), 及输入脚的选择:</p> <p>00: CC3通道被配置为输出;</p> <p>01: CC3通道被配置为输入, IC3映射在TI3FP3上;</p> <p>10: 保留;</p> <p>11: 保留。</p> <p>注: CC3S仅在通道关闭时(TIMx_CCER2寄存器的CC3E=0)才是可写的。</p>

● 通道配置为输入模式:

7	6	5	4	3	2	1	0
		IC3F[3:0]		IC3PSC[1:0]		CC3S[1:0]	

rw

rw

rw

rw

rw

rw

rw

rw

位7:4	IC3F[3:0]: 输入捕获3滤波器
位3:2	IC3PSC: 输入/捕获3预分频器
位1:0	<p>CC3S[1:0]: 捕获/比较3选择。</p> <p>这2位定义通道的方向(输入/输出), 及输入脚的选择:</p> <p>00: CC1通道被配置为输出;</p> <p>01: CC1通道被配置为输入, IC3映射在TI3FP3上;</p> <p>10: 保留;</p> <p>11: 保留。</p> <p>注: CC3S仅在通道关闭时(TIMx_CCER2寄存器的CC3E=0)才是可写的。</p>

18.6.11 捕获/比较使能寄存器 1(TIMx_CCER1)

地址偏移值: 0x08(TIM2), 0x07(TIM3), 0x0A(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	CC2P	CC2E	保留	CC1P	CC1E		

r w r w r w r w

位7:6	保留
位5	CC2P: 输入/捕获2输出极性。 参考CC1P的描述。
位4	CC2E: 输入/捕获2输出使能。 参考CC1E的描述。
位3:2	保留
位1	CC1P: 输入/捕获1输出极性 CC1通道配置为输出: 0: OC1高电平有效 1: OC1低电平有效 CC1通道配置为输入或者捕获(参考图61): 0: 捕获发生在TI1F或TI2F的上升沿; 1: 捕获发生在TI1F或TI2F的下降沿.
位0	CC1E: 输入/捕获1输出使能 CC1通道配置为输出: 0: 关闭— OC1禁止输出。 1: 开启— OC1信号输出到对应的输出引脚。 CC1 通道配置为输入: 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。

18.6.12 捕获/比较使能寄存器 2(TIMx_CCER2)

地址偏移值: 0x09(TIM2), 0x0B(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0	
保留	保留	保留	保留	保留	保留	CC3P	CC3E	rw

注意: TIM3没有该寄存器

位7:2	保留
位1	CC3P: 输入/捕获3输出极性。 参考CC1P的描述。
位0	CC3E: 输入/捕获3输出使能。 参考CC1E的描述。

18.6.13 计数器高位(TIMx_CNTRH)

地址偏移值: 0x0A(TIM2), 0x08(TIM3), 0x0C(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

CNT[15:8]

rw

rw

rw

rw

rw

rw

rw

rw

位7:0

CNT[15:8]: 计数器的值(MSB)

18.6.14 计数器低位(TIMx_CNTRL)

地址偏移值: 0x0B(TIM2), 0x09(TIM3), 0x0D(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

CNT[7:0]

rw

rw

rw

rw

rw

rw

rw

rw

位7:0

CNT[7:0]: 计数器的值(LSB)

18.6.15 预分频器(TIMx_PSCR)

地址偏移值: 0x0C(TIM2), 0x0A(TIM3), 0x0E(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
保留				PSC[3:0]			
				rw	rw	rw	rw

位7:4	保留
位3:0	<p>PSC[3:0]: 预分频器的值 预分频器对输入的CK_PSC时钟进行分频。 计数器的时钟频率f_{CK_CNT}等于$f_{CK_PSC}/2^{(PSC[3:0])}$。PSC[7:4]由硬件清0。 PSCR包含了当更新事件产生时装入当前预分频器寄存器的值(包括由于清除TIMx_EGR寄存器的UG位产生的计数器清除事件)。这意味着如要新的预分频值生效, 必须产生更新事件。</p>

18.6.16 自动装载寄存器高位(TIMx_ARRH)

地址偏移值: 0x0D(TIM2), 0x0B(TIM3), 0x0F(TIM5)

复位值: 0xFF

7 6 5 4 3 2 1 0

ARR[15:8]

RW RW RW RW RW RW RW RW

位7:0

ARR[15:8]自动装载值(MSB)

ARR是自动装载的值, 请参考 [17.3时基单元](#)。

当自动装载的值为0时, 计数器不工作。

18.6.17 自动装载寄存器低位(TIMx_ARRL)

地址偏移值: 0x0E(TIM2), 0x0C(TIM3), 0x10(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

ARR[7:0]

RW RW RW RW RW RW RW RW

位7:0

ARR[7:0]自动装载值 LSB

18.6.18 捕获/比较寄存器 1 高位(TIMx_CCR1H)

地址偏移值: 0x0F(TIM2), 0x0D(TIM3), 0x11(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
CCR1[15:8]							
RW	RW	RW	RW	RW	RW	RW	RW

位7:0

CCR1[15:8]: 捕获/比较1值(MSB)

当CC1通道为输出时(TIMx_CCMR1寄存器的CC1S位):

CCR1包含了装入当前捕获/比较1寄存器的值(预装载值)。

如果在TIMx_CCMR1寄存器(OC1PE位)中未选择预装载特性, 写入的数值会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。

当CC1通道为输入时(TIMx_CCMR1寄存器的CC1S位):

CCR1包含了由上一次输入捕获1事件(IC1)传输的计数器值, 此时寄存器只读。

18.6.19 捕获/比较寄存器 1 低位(TIMx_CCR1L)

地址偏移值: 0x10(TIM2), 0x0E(TIM3), 0x12(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

CCR1[7:0]

rw

rw

rw

rw

rw

rw

rw

rw

位7:0

CCR1[7:0]: 捕获/比较1值(LSB)

18.6.20 捕获/比较寄存器 2 高位(TIMx_CCR2H)

地址偏移值: 0x11(TIM2), 0x0F(TIM3), 0x13(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
CCR2[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

位7:0	<p>CCR2[15:8]: 捕获/比较2值(MSB)</p> <p>当CC2通道为输出时(TIMx_CCMR2寄存器的CC2S位):</p> <p>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。</p> <p>当前捕获/比较寄存器参与同计数器TIMx_CNT的比较, 并在OC2端口上产生输出信号。</p> <p>当CC21通道为输入时(TIMx_CCMR2寄存器的CC2S位):</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, 此时寄存器只读。</p>
------	--

18.6.21 捕获/比较寄存器 2 低位(TIMx_CCR2L)

地址偏移值: 0x12(TIM2), 0x10(TIM3), 0x14(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

CCR2[7:0]

RW RW RW RW RW RW RW RW

位7:0

CCR2[7:0]: 捕获/比较2值(LSB)

18.6.22 捕获/比较寄存器 3 高位(**TIMx_CCR3H**)

地址偏移值: 0x13(TIM2), 0x15(TIM5)

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

CCR3[15:8]

rw							
----	----	----	----	----	----	----	----

注意: *TIM3*不含此寄存器

位7:0	CCR3[15:8]: 捕获/比较3值(MSB)
------	--------------------------

当CC3通道为输出时(**TIMx_CCMR3**寄存器的**CC3S**位):

CCR3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_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。

当CC3通道为输入时(**TIMx_CCMR3**寄存器的**CC3S**位):

CCR3包含了由上一次输入捕获3事件(IC3)传输的计数器值, 此时寄存器只读。

18.6.23 捕获/比较寄存器 3 低位(TIMx_CCR3L)

地址偏移值: 0x14(TIM2), 0x16(TIM5)

复位值: 0x00

7 6 5 4 3 2 1 0

CCR3[7:0]

rw rw rw rw rw rw rw rw

注意: TIM3不含此寄存器

位7:0

CCR3[7:0]: 捕获/比较3值 LSB)

18.6.24 TIM2/TIM3/TIM5 寄存器图和复位值

表36 TIM2 – 寄存器图

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	TIM2_CR1	ARPE				OPM	URS	UDIS	CEN
	复位值	0	0	0	0	0	0	0	0
0x01	TIM2_IER					CC3IE	CCWIE	CC1IE	UIE
	复位值	0	0	0	0	0	0	0	0
0x02	TIM2_SR1					CC3IF	CC2IF	CC1IF	UIF
	复位值	0	0	0	0	0	0	0	0
0x03	TIM2_SR2					CC3OF	CC2OF	CC1OF	
	复位值	0	0	0	0	0	0	0	0
0x04	TIM2_EGR					CC3G	CC2G	CC1G	UG
	复位值	0	0	0	0	0	0	0	0
0x05	TIM2_CCMR1 输出比较模式		OC1M2	OC1M1	OC1MO	OC1PE		CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
	TIM2_CCMR1 输入捕获模式	IC1F3	IC1F2	IC1F1	IC1F0	IC1PSC1	IC1PSC0	CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
0x06	TIM2_CCMR2 输出比较模式		OC2M2	OC2M1	OC2M0	OC2PE		CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
	TIM2_CCMR2 输入捕获模式	IC2F3	IC2F2	IC2F1	IC2F0	IC2PSC1	IC2PSC0	CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
0x07	TIM2_CCMR3 输出比较模式		OC3M2	OC3M1	OC3M0	OC3PE		CC3S1	CC3S0
	复位值	0	0	0	0	0	0	0	0
	TIM2_CCMR3 输入捕获模式	IC3F3	IC3F2	IC3F1	IC3F0	IC3PSC1	IC3PSC0	CC3S1	CC3S0
	复位值	0	0	0	0	0	0	0	0
0x08	TIM2_CCER1			CC2P	CC2E			CC1P	CC1E
	复位值	0	0	0	0	0	0	0	0
0x09	TIM2_CCER2							CC3P	CC3E
	复位值	0	0	0	0	0	0	0	0
0x0A	TIM2_CNTRH	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8
	复位值	0	0	0	0	0	0	0	0
0x0B	TIM2_CNTRL	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	复位值	0	0	0	0	0	0	0	0
0x0C	TIM2_PSCR					PSC3	PSC2	PSC1	PSC0
	复位值	0	0	0	0	0	0	0	0

偏移	寄存器	7	6	5	4	3	2	1	0
0x0D	TIM2_ARRH	ARR15	ARR14	ARR13	ARR12	ARR11	ARR10	ARR9	ARR8
	复位值	1	1	1	1	1	1	1	1
0x0E	TIM2_ARRL	ARR7	ARR6	ARR5	ARR4	ARR3	ARR2	ARR1	ARR0
	复位值	1	1	1	1	1	1	1	1
0x0F	TIM2_CCR1H	CCR1_15	CCR1_14	CCR1_13	CCR1_12	CCR1_11	CCR1_10	CCR1_9	CCR1_8
	复位值	0	0	0	0	0	0	0	0
0x10	TIM2_CCR1L	CCR1_7	CCR1_6	CCR1_5	CCR1_4	CCR1_3	CCR1_2	CCR1_1	CCR1_0
	复位值	0	0	0	0	0	0	0	0
0x11	TIM2_CCR2H	CCR2_15	CCR2_14	CCR2_13	CCR2_12	CCR2_11	CCR2_10	CCR2_9	CCR2_8
	复位值	0	0	0	0	0	0	0	0
0x12	TIM2_CCR2L	CCR2_7	CCR2_6	CCR2_5	CCR2_4	CCR2_3	CCR2_2	CCR2_1	CCR2_0
	复位值	0	0	0	0	0	0	0	0
0x13	TIM2_CCR3H	CCR3_15	CCR3_14	CCR3_13	CCR3_12	CCR3_11	CCR3_10	CCR3_9	CCR3_8
	复位值	0	0	0	0	0	0	0	0
0x14	TIM2_CCR3L	CCR3_7	CCR3_6	CCR3_5	CCR3_4	CCR3_3	CCR3_2	CCR3_1	CCR3_0
	复位值	0	0	0	0	0	0	0	0

表37 TIM3 – 寄存器图

偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	TIM3_CR1	ARPE				OPM	URS	UDIS	CEN
	复位值	0	0	0	0	0	0	0	0
0x01	TIM3_IER						CC2IE	CC1IE	UIE
	复位值	0	0	0	0	0	0	0	0
0x02	TIM3_SR1						CC2IF	CC1IF	UIF
	复位值	0	0	0	0	0	0	0	0
0x03	TIM3_SR2						CC2OF	CC1OF	
	复位值	0	0	0	0	0	0	0	0
0x04	TIM3_EGR						CC2G	CC1G	UG
	复位值	0	0	0	0	0	0	0	0
0x05	TIM3_CCMR1 输出比较模式		OC1M2	OC1M1	OC1MO	OC1PE		CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
	TIM3_CCMR1 输入捕获模式	IC1F3	IC1F2	IC1F1	IC1F0	IC1PSC1	IC1PSC0	CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
0x06	TIM3_CCMR2 输出比较模式		OC2M2	OC2M1	OC2M0	OC2PE		CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
	TIM3_CCMR2 输入捕获模式	IC2F3	IC2F2	IC2F1	IC2F0	IC2PSC1	IC2PSC0	CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
0x07	TIM3_CCER1			CC2P	CC2E			CC1P	CC1E
	复位值	0	0	0	0	0	0	0	0
0x08	TIM3_CNTRH	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8
	复位值	0	0	0	0	0	0	0	0
0x09	TIM3_CNTRL	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	复位值	0	0	0	0	0	0	0	0
0x0A	TIM3_PSCR					PSC3	PSC2	PSC1	PSC0
	复位值	0	0	0	0	0	0	0	0

偏移值	寄存器	7	6	5	4	3	2	1	0
0x0B	TIM3_ARRH	ARR15	ARR14	ARR13	ARR12	ARR11	ARR10	ARR9	ARR8
	复位值	1	1	1	1	1	1	1	1
0x0C	TIM3_ARRL	ARR7	ARR6	ARR5	ARR4	ARR3	ARR2	ARR1	ARR0
	复位值	1	1	1	1	1	1	1	1
0x0D	TIM3_CCR1H	CCR115	CCR114	CCR113	CCR112	CCR111	CCR110	CCR19	CCR18
	复位值	0	0	0	0	0	0	0	0
0x0E	TIM3_CCR1L	CCR17	CCR16	CCR15	CCR14	CCR13	CCR12	CCR11	CCR10
	复位值	0	0	0	0	0	0	0	0
0x0F	TIM3_CCR2H	CCR215	CCR214	CCR213	CCR212	CCR211	CCR210	CCR29	CCR28
	复位值	0	0	0	0	0	0	0	0
0x10	TIM3_CCR2L	CCR27	CCR26	CCR25	CCR24	CCR23	CCR22	CCR21	CCR20
	复位值	0	0	0	0	0	0	0	0

表38 TIM5 – 寄存器图

偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	TIM5_CR1	ARPE				OPM	URS	UDIS	CEN
	复位值	0	0	0	0	0	0	0	0
0x01	TIM5_CR2	TI1S	MMS2	MMS1	MMS0		COMS		CCPC
	复位值	0	0	0	0	0	0	0	0
0x02	TIM5_SMCR	MSM	TS2	TS1	TS0		SMS2	SMS1	SMS0
	复位值	0	0	0	0	0	0	0	0
0x03	TIM5_IER					CC3IE	CC2IE	CC1IE	UIE
	复位值	0	0	0	0	0	0	0	0
0x04	TIM5_SR1					CC3IF	CC2IF	CC1IF	UIF
	复位值	0	0	0	0	0	0	0	0
0x05	TIM5_SR2					CC3OF	CC2OF	CC1OF	
	复位值	0	0	0	0	0	0	0	0
0x06	TIM5_EGR					CC3G	CC2G	CC1G	UG
	复位值	0	0	0	0	0	0	0	0
0x07	TIM5_CCMR1 输出比较模式		OC1M2	OC1M1	OC1MO	OC1PE		CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
	TIM5_CCMR1 输入捕获模式	IC1F3	IC1F2	IC1F1	IC1F0	IC1PSC1	IC1PSC0	CC1S1	CC1S0
	复位值	0	0	0	0	0	0	0	0
0x08	TIM5_CCMR2 输出比较模式		OC2M2	OC2M1	OC2M0	OC2PE		CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
	TIM5_CCMR2 输入捕获模式	IC2F3	IC2F2	IC2F1	IC2F0	IC2PSC1	IC2PSC0	CC2S1	CC2S0
	复位值	0	0	0	0	0	0	0	0
0x09	TIM5_CCMR3 输出比较模式		OC3M2	OC3M1	OC3M0	OC3PE		CC23S1	CC3S0
	复位值	0	0	0	0	0	0	0	0
	TIM5_CCMR3 输入捕获模式	IC3F3	IC3F2	IC3F1	IC3F0	IC3PSC1	IC3PSC0	CC3S1	CC3S0
	复位值	0	0	0	0	0	0	0	0

偏移值	寄存器	7	6	5	4	3	2	1	0
0x0A	TIM5_CCER1			CC2P	CC2E			CC1P	CC1E
	复位值	0	0	0	0	0	0	0	0
0x0B	TIM5_CCER2							CC3P	CC3E
	复位值	0	0	0	0	0	0	0	0
0x0C	TIM5_CNTRH	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8
	复位值	0	0	0	0	0	0	0	0
0x0D	TIM5_CNTRL	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	复位值	0	0	0	0	0	0	0	0
0x0E	TIM5_PSCR					PSC3	PSC2	PSC1	PSC0
	复位值	0	0	0	0	0	0	0	0
0x0F	TIM5_ARRH	ARR15	ARR14	ARR13	ARR12	ARR11	ARR10	ARR9	ARR8
	复位值	1	1	1	1	1	1	1	1
0x10	TIM5_ARRL	ARR7	ARR6	ARR5	ARR4	ARR3	ARR2	ARR1	ARR0
	复位值	1	1	1	1	1	1	1	1
0x11	TIM5_CCR1H	CCR115	CCR114	CCR113	CCR112	CCR111	CCR110	CCR19	CCR18
	复位值	0	0	0	0	0	0	0	0
0x12	TIM5_CCR1L	CCR17	CCR16	CCR15	CCR14	CCR13	CCR12	CCR11	CCR10
	复位值	0	0	0	0	0	0	0	0
0x13	TIM5_CCR2H	CCR215	CCR214	CCR213	CCR212	CCR211	CCR210	CCR29	CCR28
	复位值	0	0	0	0	0	0	0	0
0x14	TIM5_CCR2L	CCR27	CCR26	CCR25	CCR24	CCR23	CCR22	CCR21	CCR20
	复位值	0	0	0	0	0	0	0	0
0x15	TIM5_CCR3H	CCR315	CCR314	CCR313	CCR312	CCR311	CCR310	CCR39	CCR38
	复位值	0	0	0	0	0	0	0	0
0x16	TIM5_CCR3L	CCR37	CCR36	CCR35	CCR34	CCR33	CCR32	CCR31	CCR30
	复位值	0	0	0	0	0	0	0	0

19 8位基本定时器 (TIM4, TIM6)

19.1 简介

该定时器由一个带可编程预分频器的8位自动重载的向上计数器所组成，它可以用来作为时基发生器，具有溢出中断功能。

TIM6 同时钟/触发信号控制器一起用于定时器同步和级联。关于定时器基本功能的描述参考 [17.3时基单元](#) 内容。

图86 TIM4功能图

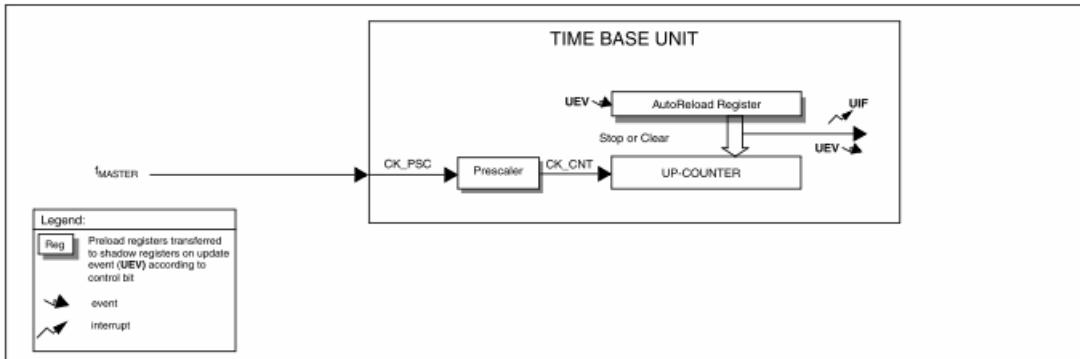
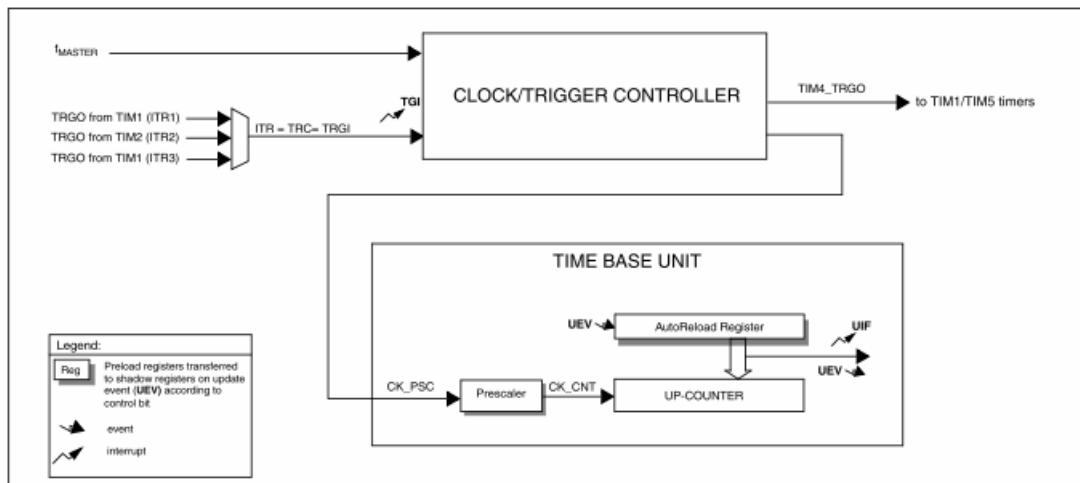


图87 TIM6功能图



19.2 TIMER4的主要功能

主要功能包括如下：

- 8位向上计数的自动重载计数器；
- 3位可编程的预分配器(可在运行中修改)，提供1, 2, 4, 8, 16, 32, 64 和 128 这8种分频比例。
- 中断产生
 - 在计数器更新时：计数器溢出

19.3 TIMER6的主要功能

主要功能包括如下：

- 8位向上计数的自动重载计数器；
- 3位可编程的预分配器(可在运行中修改)，提供1, 2, 4, 8, 16, 32, 64 和 128 这8种分频比例。

- 用于和外部信号相连和定时器级联的同步电路
- 中断产生
 - 在计数器更新时: 计数器溢出
 - 在触发信号输入时

19.4 TIM4/TIM6中断

该定时器具有2个中断请求源:

- 更新中断(溢出, 计数器初始化);
- 触发信号输入(仅TIM6可用)

19.5 TIM4/TIM6时钟选择

该定时器的时钟源是内部时钟(f_{MASTER})。该时钟源是直接连接到 CK_PSC 时钟的, CK_PSC时钟通过预分频器分频后给定时器提供CK_CNT时钟。

预分配器

预分频器的功能如下:

- 预分频器是基于由一个3位寄存器 (在TIMX_PSCR寄存器中) 来控制的一个7位的计数器。由于该控制寄存器是带缓冲的所以它可以在系统运行中被改变。可以分频计数器的时钟频率为1到128之间的2的任意次幂。 $f_{CK_CNT} = f_{CK_PSC}/2^{(PSCR[2:0])}$

预分频器的值是通过一个预装载寄存器来载入的。一旦LS字节被写入时, 保存当前要被使用值的影子寄存器的值就被立即载入。对TIMX_PSCR寄存器的读操作是访问预装载寄存器, 因此在读的过程中没有什么特别要注意的地方。

19.6 TIM4/TIM6 寄存器

19.6.1 控制寄存器 1 (TIMx_CR1)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
ARPE		保留		OPM	URS	UDIS	CEN

rW

rW

rW

rW

rW

位7	APPRE: AUTO_RELOAD 预装载使能 0: TIM4_ARR寄存器通过预装载寄存器无缓冲。可直接写。 1: TIM4_ARR寄存器通过缓冲预装载
位6:4	保留, 须保持清零
位3	OPM: 单脉冲模式 0: 计数器在更新事件时不停止 1: 在下一个更新事件时计数器停止计数。(清零CEN位)
位2	URS: 更新请求 0: 当使能时, 寄存器更新(计数器溢出)时立即发送一个中断请求。 1: 当使能时, 仅当计数器达到向上溢出/向下溢出时才发送一个中断请求。
位1	UDIS: 禁止更新 0: 当计数器溢出或者软件更新时, 立即产生一次更新事件。预装载寄存器的值立即加载到缓冲寄存器。 1: 禁止产生更新事件, 影子寄存器保持当前的值(ARR,PSC)。如果UG位置位, 则计数器和预分频器被重新初始化。
位0	CEN: 计数器使能位 0: 禁止计数器 1: 使能计数器

19.6.2 控制寄存器 2 (TIMx_CR2)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
保留		TS[2:0]		保留			

rw rw rw

注意: TIM4没有该寄存器。

位7:4	保留, 须保持清零
位6:4	<p>MMS: 主模式选择 在主模式下, 这些位用来选择用于给其他模块作同步的输出信号(TRGO)。组合方式如下所示:</p> <p>000: 复位 – TIM3_EGR的UG位用来做触发输出(TRGO)。如果由触发输入(时钟/触发控制器配置成触发复位模式)产生复位, 那么与实际的复位比较, TRGO上的信号被延迟。</p> <p>001: 使能 – 计数器使能信号被用来作为信号输出。用来同时启动几个定时器或者控制一个窗口, 使能一个从模式定时器。当配置成GATED MODE 时, 计数器使能信号由CEN 控制位和触发输入信号进行逻辑或产生。当计数器使能信号由触发输入控制时, TRGO产生一个延迟, 除非选择了主/从模式(参考TIM_SMCR寄存器的MSM位的描述)。</p> <p>010: 更新 – 由更新事件作为触发输出(TRGO)。</p> <p>011: 保留</p> <p>100: 保留</p> <p>101: 保留</p> <p>111: 保留</p>
位3:0	保留, 须保持清零

19.6.3 从模式控制寄存器 (TIMx_SMCR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
MSM	TS[2:0]		保留		SMS[2:0]		

rw rw rw rw rw rw rw

注意: TIM4没有该寄存器。

位7	MSM: 主/从模式 0: 无影响 1: 触发输入(TRGI)上事件的动作被延迟来使定时器之间完成同步(通过TRGO),
位6:4	TS: 触发选择 这些位用来选择相应的触发输入模式来同步计数器。 000: 保留 001: 保留 010: 内部触发ITR2 连接到 TIM5 TRGO 011: 内部触发ITR3 连接到 TIM1 TRGO 100: 保留 101: 保留 110: 保留 111: 保留 注意: 只有在他们没有使用时才能改变这些位的设置(例如, 当SMS=00时), 避免传送时检测到错误的边沿。
位3	保留, 须保持清零
位2:0	SMS: 时钟/触发/从模式选择 当选择了外部信号, 触发信号(TRGI)的边沿检测被连接到外部输入的极性选择上。(参考输入控制寄存器和控制器描述)。 000: 时钟/触发控制器禁止 – 如果 CEN=1, 则由内部时钟提供分频器的时钟。 001: 保留 010: 保留 011: 保留 100: 触发复位模式 – 触发信号(TRGI)的上升沿使计数器重新初始化, 并且产生一次寄存器更新事件。 101: 门控模式 – 当触发信号(TRGI)为高时, 计数器时钟使能。当触发信号变低时, 计数器停止(但是不复位)。所有的计数器启动和停止都受控。 110: 触发模式 – 当触发信号(TRGI)上升沿时, 计数器启动(但是没有复位)。只有计数器的启动受控。 111: 外部时钟模式1 – 触发信号(TRGI)上升沿作为计数器的时钟输入。

19.6.4 中断使能寄存器 (TIMx_IER)

地址偏移值: 0x01(TIM4), 0x03(TIM6)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TIE			保留			UIE

位7	保留, 须保持清零
位6	TIE: 触发中断使能 0: 触发中断禁止 1: 触发中断使能
位5:1	保留, 须保持清零
位0	UIE: 更新中断使能 0: 更新中断禁止 1: 更新中断使能

19.6.5 状态寄存器 1 (TIMx_SR1)

地址偏移值: 0x02(TIM4), 0x04(TIM6)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TIF			保留			UIF

位7	保留, 须保持清零
位6	<p>TIF: 触发中断标志 此位在触发事件发生时(检测到TRGI信号的有效沿, 在选择门控模式时上升沿和下降沿都有效)由硬件置位。可以由软件清零。</p> <p>0: 无触发事件产生 1: 触发事件发生。此位当寄存器更新时由硬件置位。</p> <p>注意: 在TIM4中该位保留</p>
位5:1	保留, 须保持清零
位0	<p>UIF: 更新中断标志 此位在更新事件发生时由硬件置位。可以由软件清零。</p> <p>0: 无更新事件产生 1: 跟新事件发生。此位当寄存器更新时由硬件置位。</p> <p>-如果TIM4_CR1中的UDIS=0,则发生在计数器溢出时 -如果TIM4_CR1中的UDIS=0和URS=0,则发生在通过设置TIM4_EGR的UG位产生软件重新初始化计数器时</p>

19.6.6 事件产生寄存器(TIMx_EGR)

地址偏移值: 0x03(TIM4), 0x05(TIM6)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TG	保留					UG
W							W

位7	保留, 须保持清零
位0	<p>TG 触发事件产生 可用软件对该位置位以产生一个触发事件。该位由硬件自动清0。 0: 无触发产生 1: TIM4_SR1中TIF标志被置1。如果TIE位为1则产生中断。 注意: 在TIM4中该位保留</p>
位5:1	保留, 须保持清零
位0	<p>UG 更新事件产生 0: 无更新事件产生 1: 计数器重新初始化并产生寄存器更新。 注意: 分频计数器也同时清零</p>

19.6.7 计数器 (TIMx_CNTR)

地址偏移值: 0x04(TIM4), 0x06(TIM6)

复位值: 0x00

7 6 5 4 3 2 1 0

CNT[7:0]

RW RW RW RW RW RW RW RW

位7:0	CNT[7:0] :计数器值
------	----------------

19.6.8 预分频寄存器 (TIMx_PSCR)

地址偏移值: 0x05(TIM4), 0x07(TIM6)

复位值: 0x00

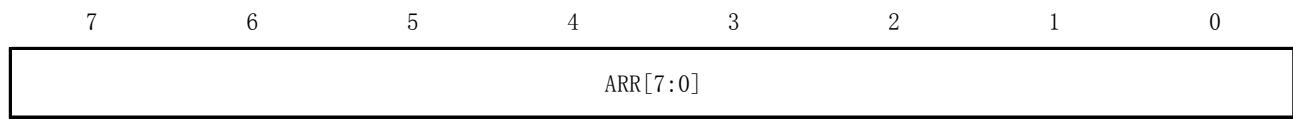
7	6	5	4	3	2	1	0
保留				PSC[2:0]			
				rW	rW	rW	

位7:3	保留, 须保持清零
位2:0	<p>PSC [2:0]: 分频器的值 分频器的值除分频时钟 CK_PSC 计数器时钟频率 $f_{CK_CNT} = f_{CK_PSC} / 2^{(PSC[2:0])}$ PSC中包含了每次更新事件(包含当通过TIM4_EGR寄存器的UG产生的更新事件)需要加载到实际分频寄存器的值 这就意味着为了使新的分频器值启用, 必须产生一次更新事件。</p>

19.6.9 自动重装载寄存器(TIMx_ARR)

地址偏移值: 0x06(TIM4), 0x07(TIM6)

复位值: 0xFF



19.6.10 TIM4/TIM6 寄存器表和复位值

表39 TIM4寄存器表

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	TIM4_CR1 复位值	ARPE 0	— 0	— 0	— 0	OPM 0	URS 0	UDIS 0	CEN 0
0x01	TIM4_IER 复位值	— 0	UIE 0						
0x02	TIM4_SR1 复位值	— 0	UIF 0						
0x03	TIM4_EGR 复位值	— 0	UG 0						
0x04	TIM4_CNTR 复位值	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0x05	TIM4_PSCR 复位值	— 0	— 0	— 0	— 0	— 0	PSC2 0	PSC1 0	PSC0 0
0x06	TIM4_ARR 复位值	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1	ARRO 1

表40 TIM6寄存器表

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	TIM6_CR1 复位值	ARPE 0	— 0	— 0	— 0	OPM 0	URS 0	UDIS 0	CEN 0
0x01	TIM6_CR2 复位值	— 0	MMS2 0	MMS1 0	MMS0 0	— 0	— 0	— 0	— 0
0x02	TIM6_SMCR 复位值	MSM 0	TS2 0	TS1 0	TS0 0	— 0	SMS2 0	SMS1 0	SMS0 0
0x03	TIM6_IER 复位值	— 0	UIE 0						
0x04	TIM6_SR1 复位值	— 0	UIF 0						
0x05	TIM6_EGR 复位值	— 0	UG 0						
0x06	TIM6_CNTR 复位值	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0x07	TIM6_PSCR 复位值	— 0	— 0	— 0	— 0	— 0	PSC2 0	PSC1 0	PSC0 0
0x08	TIM6_ARR 复位值	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1	ARRO 1

20 串行外设接口(SPI)

20.1 SPI简介

串行外设接口(SPI)允许芯片与其他设备以半/全双工、同步、串行方式通信。此接口可以被配置成主模式，并为从设备提供通信时钟(SCK)。接口还能以多主配置方式工作。

它可用于多种用途，包括带或不带第三根双向数据线的双线单工同步传输，还可使用CRC校验来进行可靠通信。

20.2 SPI主要特征

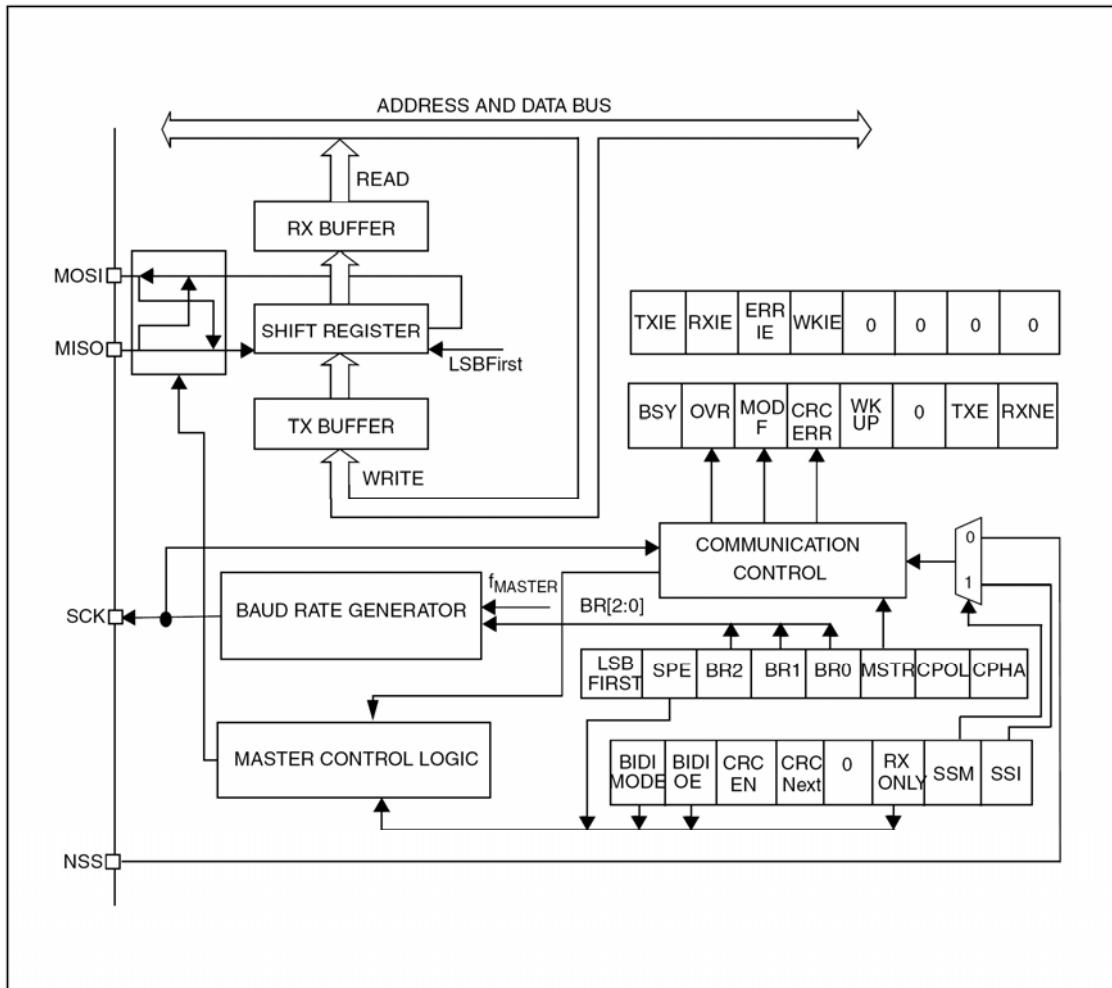
- 3线全双工同步传输
- 带或不带第三根双向数据线的双线单工同步传输
- 8或16位传输帧格式选择
- 主或从操作
- 8个主模式频率(最大为 $f_{MASTER}/2$)
- 从模式频率(最大为 $f_{PCLK}/2$)
- 快速通信：最大SPI速度达到10MHz
- 主模式和从模式下均可以由软件或硬件进行NSS管理
- 可编程的时钟极性和相位
- 可编程的数据顺序，MSB在前或LSB在前
- 可触发中断的专用发送和接收标志
- SPI总线忙状态标志
- 可触发中断的主模式出错和溢出标志
- 支持可靠通信的硬件CRC
 - 在发送模式下，CRC值可以被作为最后一个字节发送
 - 在接收到最后一个字节时自动进行CRC出错检查
- 唤醒功能
 - 在全或半双工只发送模式下MCU可以从低功耗模式唤醒

20.3 SPI功能描述

20.3.1 概述

SPI的方框图见[图88](#)。

图88 SPI框图



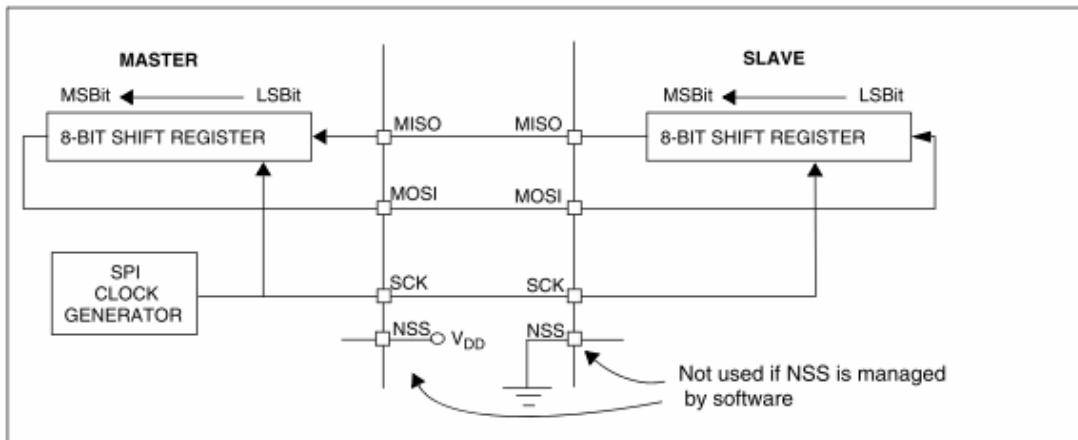
通常SPI通过4个管脚与外部器件相连：

- **MISO**: 主设备输入/从设备输出管脚(端口C7)。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(端口C6)。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(端口C5)，作为主设备的输出，从设备的输入
 - **NSS**: 从设备选择(端口E5)。配置主/从模式时，这是一个可选的管脚。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的从设备通讯，避免数据线上的冲突。从设备的**NSS**管脚可以被主设备的标准IO来驱动。

图89是一个单主和单从设备互连的例子。

注意：当使用SPI的高速模式时，SPI输出端口对应的I/O必须配置为快速摆率输出，以满足要求的总线速度。

图89 单主和单从应用



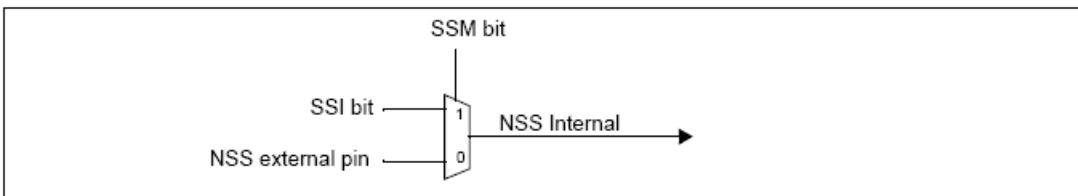
MOSI脚相互连接, MISO脚相互连接。这样, 数据在主和从之间串行地传输(MSB位在前)。

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。

从选择(NSS)脚管理

应用程序可以使用软件的方式代替使用NSS管脚(NSS引脚, 端口E5)来控制从设备选择信号, 参见图90。这样的话, 外部的NSS引脚可以空出来给其他应用使用; 而内部NSS信号的电平由SPI_CSR寄存器中的SSI位控制了。

图90 硬件/软件的从选择管理



时钟信号的相位和极性

使用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。

注意: 确保SPI引脚配置成SPI空闲状态时的电平, 以免在使能或者禁止SPI模块的时候在SPI时钟引脚上产生一个边沿。

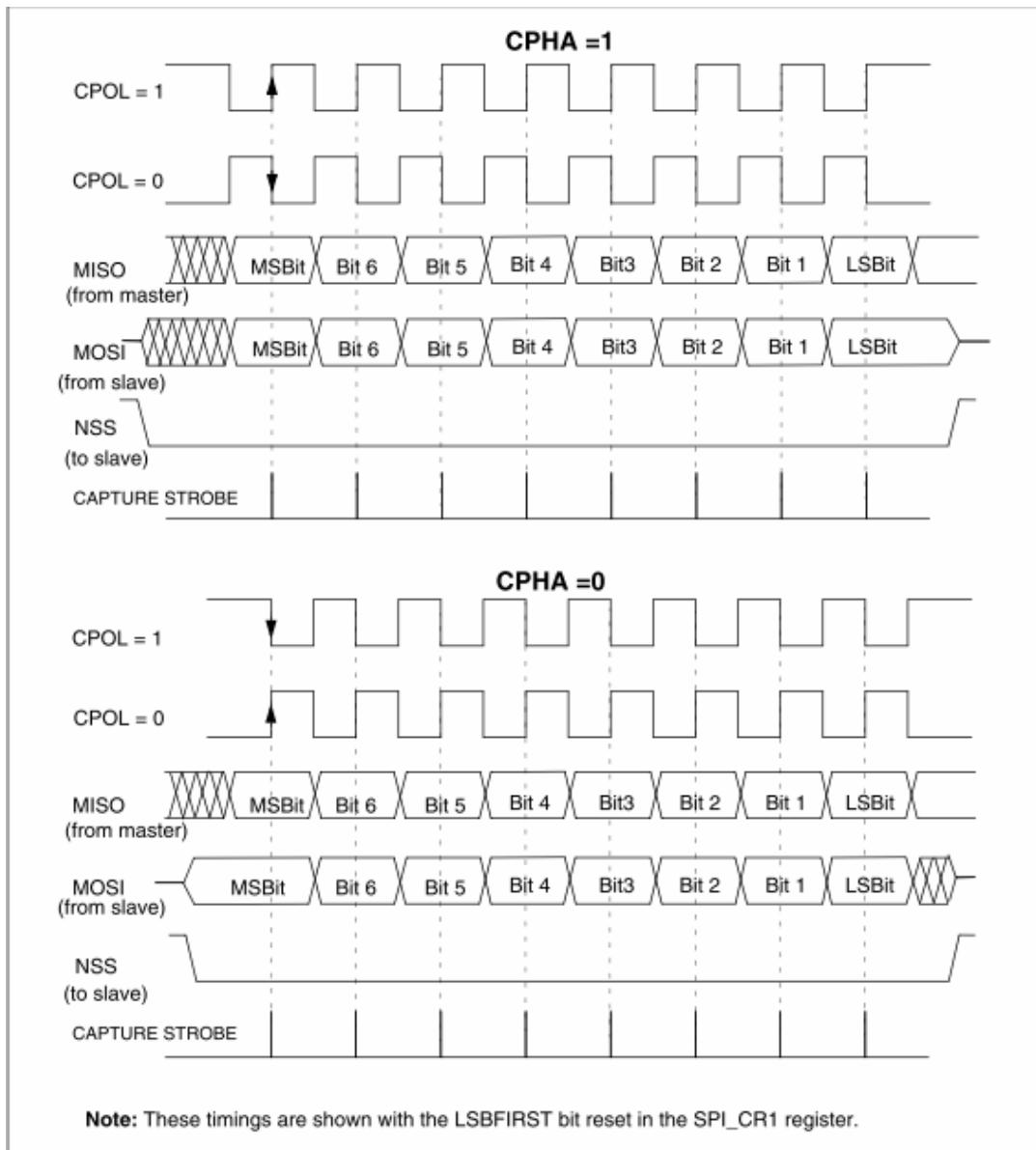
如果CPHA(时钟相位)位被置'1', SCK时钟的第二个边沿(CPOL位为0时就是下降沿, CPOL位为1时就是上升沿)进行最高数据位的采样, 数据在第一个时钟传输周期被锁存。如果CPHA位被清'0', SCK时钟的第一边沿(CPOL位为0时就是下降沿, CPOL位为1时就是上升沿)进行数据位采样, 数据在第二个时钟传输周期被锁存。

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91显示了SPI传输的4种CPHA和CPOL位选择不同的组合主设备与从设备的SCK, MISO, MOSI引脚直接连接时, 这些管脚上的时序。

- 注意:**
1. 在改变CPOL/CPHA位之前, 必须清除SPE位将SPI禁止。
 2. 主设备和从设备必须配置成相同的时序模式。
 3. SCK的空闲状态必须和SPI_CR1寄存器指定的极性一致(CPOL为1时, 应上拉SCK为高电平; CPOL为0时, 应下拉SCK为低电平)。

图91 数据时钟时序图



数据帧格式

根据SPI_CR1寄存器中的LSBFIRST位，输出数据位时可以MSB在先也可以LSB在先。

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_CR1寄存器中BR[2:0]的设置不影响数据传输速率。

配置步骤

1. 选择CPOL和CPHA位来定义数据传输和串行时钟之间的相位关系(见图91)。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。
2. 帧格式(MSB在前还是LSB在前取决于SPI_CR1寄存器中的LSBFIRST位)必须和主设备相同。
3. 在使用硬件模式 (参考[从选择\(NSS\)脚管理部分](#)) 时，NSS引脚在字节传输的全部过程中都必须为低电平。在使用件模式时，设置SPI_CR2寄存器中的SSM位并清除SSI位。
4. 清除MSTR位，设置SPE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。

数据传输过程

数据字节被并行地写入发送缓冲器。

当从设备收到时钟信号时，发送过程开始。第一位数据发送到MOSI引脚上。余下的7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，TXE标志被置位。如果设置了SPI_ICR寄存器的TXEIE位，将会产生中断。

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，RXNE标志被置位。
- 如果设置了RXIE位，则产生中断。

在最后一个采样时钟边沿，RXNE位被置‘1’，移位寄存器中接收到的数据字节拷贝到接收缓冲器。读取SPI_DR寄存器得到这个缓冲值。读SPI_DR寄存器时，RXNE位被清除。

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_CR1寄存器的BR[2:0]位定义串行时钟波特率。
2. 选择CPOL和CPHA位，定义数据传输和串行时钟间的相位关系(见[图91](#))。
3. 配置SPI_CR1寄存器的LSBFIRST位定义帧格式。
4. 硬件模式下，在数据帧的全部传输过程中应把NSS脚连接到高电平；在软件模式下，需设置SPI_CR2寄存器的SSM和SSI位为‘1’。
5. 必须设置MSTR和SPE位(只当NSS脚被连到高电平，这些位才能保持为‘1’)。

在这个配置中，MOSI脚是数据输出，而MISO脚是数据输入。

数据传输过程

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_CR1寄存器中的LSBFIRST位。数据从发送缓冲器传输到移位寄存器时TXE标志将被置位，如果设置SPI_CR1寄存器中的TXEIE位，将产生中断。

当数据传输完成时：

- 移位寄存器里的数据传送到接收缓冲器，并且RXNE标志被置位。
- 如果SPI_ICR寄存器中的RXIE位被设置，则产生中断。

在最后一个采样时钟沿，RXNE位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读取SPI_DR寄存器得到这个缓冲值。读SPI_DR寄存器将清除RXNE位。

一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。注意在试图写发送缓冲器之前，需确认TXE标志应该是1。

20.3.4 单工通信

SPI能够以两种配置工作于单工方式：

- 1条时钟线和1条双向数据线
- 1条时钟线和1条数据线(双工或接收方式)

1条时钟线和1条双向数据线

设置SPI_CR2寄存器中的BDM位启用此模式。在这个模式中，SCK用作时钟，主模式中的MOSI或从模式中的MISO用作数据通信。传输的方向(输入或输出)由SPI_CR2寄存器里的BDOE控制，当这个位是1的时候，数据线是输出，否则是输入。

1条时钟和1条数据线(双工或只接收方式)

为了释放一根I/O脚作为它用，可以通过设置SPI_CR2寄存器中的RXONLY位来禁止SPI输出功能。这样的话，SPI将运行于只接收模式。当RXONLY位置0时，SPI又会恢复到全双工模式。

只接收模式

在只接收模式下，必须首先配置并使能SPI。

- 在主模式下，一旦SPE被置1，通信立即启动，当SPE位被置0时通信即停止。在这个模式下，不需要读取BUSY标志位。因为通信在进行并且总线被占用，这个标志位一直为1，直到SPE位被置0。
- 在从模式下，只要NSS被拉低(或SSI位为0)并且SCK持续送到从设备，SPI就一直在接收。

注意：当SPI_CR2寄存器中的RXONLY位为'0'时，SPI可以工作于只发送模式，接收引脚(主设备的MISO，或者从设备的MOSI)可以当作通用IO口使用。因此读数据寄存器时，读不到接收的值。

20.3.5 状态标志

应用程序通过3个状态标志可以完全监控SPI总线的状态。

总线忙(Busy)标志

此标志表明SPI通信层的状态。当它被置1时，表明SPI正忙于通信，并且/或者在发送缓冲器里有一个有效的数据正在等待被发送。此标志的目的是说明在SPI总线上是否有正在进行的通信。以下情况时此标志将被置1：

1. 数据被写进主设备的SPI_DR寄存器上。
2. SCK时钟出现在从设备的时钟引脚上。

发送/接收一个字(字节)完成后，BUSY标志立即清除；此标志由硬件设置和清除。监视此标志可以避免写冲突错误。写此标志无效。仅当SPE位被置1时此标志才有意义。

发送缓冲器空标志(TXE)

此标志被置1时表明发送缓冲器为空，因此下一个待发送的数据可以写进缓冲器里。当发送缓冲器有一个待发送的数据时，TXE标志被清除。当SPI被禁止时(SPE位置0)，此标志被清除。

接收缓冲器非空(RXNE)

此标志为'1'时表明在接收缓冲器中包含有效的接收数据。读SPI数据寄存器可以清除此标志。

20.3.6 CRC计算

CRC校验仅用于保证通信的可靠性。数据发送和数据接收分别使用单独的CRC计算器。通过对每一个接收位进行可编程的多项式运算来计算CRC。CRC的计算是在由SPI_CR1寄存器中CPHA和CPOL位定义的采样时钟边沿进行的。

CRC计算是通过设置SPI_CR1寄存器中的CRCEN位启用的。设置CRCEN位时同时复位CRC寄存器(SPI_RXCRCR和SPI_TXCRCR)。当设置了SPI_CR2的CRCNEXT位，SPI_TXCRCR的内容将在当前字节发送之后发出。

如果Tx缓冲区中已经有一个字节，该字节发送完成后再发送CRC值。在发送CRC值的过程中，CRC计算器被关闭，CRC寄存器的值保持不变。

如果在发送SPI_TXCRCR值的过程中，接收到移位寄存器中的值和SPI_RXCRCR的值不匹配，SPI_SR寄存器中的CRCEERR标志被置位。

SPI通信可以通过以下步骤使用CRC：

- 设置CPOL、CPHA、LSBFirst、BR、SSM、SSI和MSTR的值；
- 在SPI_CRCPR寄存器输入多项式；
- 通过设置SPI_CR1寄存器CRCEN位使能CRC计算，该操作也会清除寄存器SPI_RXCRCR和SPI_TXCRCR；

- 设置SPI_CR1寄存器的SPE位启动SPI功能；
- 启动通信并且维持通信，直到只剩最后一个字节未被发送或者接收；
- 当把最后一个字节写进发送缓冲器，设置SPI_CR2的CRCNext位，指示硬件在最后一个数据字节发送完成后，发送CRC。在发送CRC期间，CRC计算停止；
- 当最后一个字节被发送后，SPI发送CRC，CRCNext位被复位。同样，接收到的CRC和SPI_RXCRCR值进行比较，如果比较不相配，SPI_SR上的CRCERR标志被置位，当设置了SPI_ICR寄存器的ERRIE时，则产生中断。

注意：当SPI时钟频率较高时，用户在采用CRC校验传输时必须小心。在采用CRC数据校验的全部传输期间内，使用CPU的时间应尽可能少。为了避免在接收最后的数据和CRC时出错，在发送带有CRC校验值的数据传输过程中应禁止函数调用。

20.3.7 错误标志

主模式错误(MODF)

主模式故障仅发生在：在片选引脚硬件模式管理下，主设备的NSS脚被拉低；或者在片选引脚软件模式管理下，SSI位被复位时。MODF位被自动置位。主模式故障对SPI设备有以下影响：

- MODF位被置位，如果设置了ERRIE位，则产生SPI中断。
- SPE位被复位。这将停止一切输出，并且关闭SPI接口。
- MSTR位被复位，因此强迫此设备进入从模式。

下面的步骤用于清除MODF位：

1. 当MODF位被置位时，执行一次对SPI_SR寄存器的读或写操作。
2. 然后写SPI_CR1寄存器。

在有多个MCU的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的NSS脚，再对MODF位进行清零。在清零的过程中或者清零完成之后，SPE和MSTR位可以恢复到它们的原始状态。

出于安全的考虑，当MODF位被置位的情况下，硬件不允许设置SPE和MSTR位。

从设备的MODF位不能被置位。然而，在多主配置里，一个设备可以在MODF位被置1的情况下，处于从设备模式；此时，MODF位指示可能出现了多主冲突。用户可以在中断服务程序中执行一个复位或者返回到一个默认的状态，使设备从错误的状态中恢复。

溢出错误

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的RXNE时，产生溢出错误。当产生溢出错误时：

- OVR位被设置；当设置了ERRIE位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读SPI_DR寄存器返回的是之前未读的字节，所有随后传送的字节都被丢弃。

对SPI_SR寄存器的读操作可以清除OVR。

CRC 错误

当设置了SPI_CR2寄存器上的CRCEN位时，CRC错误标志用来核对接收数据的正确性。如果在发送SPI_RXCRCR值的过程中，移位寄存器中接收到的值和SPI_RXCRCR寄存器中的值不匹配，SPI_SR寄存器上的CRCERR标志被置位。参见[20.3.6](#)。

20.3.8 关闭SPI

当传输结束，可以通过关闭SPI外设来终止通讯。清除SPE位即可关闭SPI。只要设备不处于主发送模式下，在最后一个字节的传输未完成时关闭SPI并不会影响通讯的可靠性。

注意：在主传送模式下(全双工或单工发送)，必须在关闭SPI前，通过查询SPI_SR寄存器的BSY标志位，保证没有任何正在进行的通讯。

20.3.9 低功耗

表41 低功耗模式下的SPI

模式	描述
等待 Wait	对SPI没有影响。 SPI中断将设备从Wait模式唤醒
停机 Halt	SPI寄存器被冻结 在Halt模式，SPI处于非激活状态。如果SPI处于主模式，“从Halt模式唤醒”中断可以唤醒设备，使得通信继续进行。 如果SPI处于从模式，当检测到第一个数据的采样边沿，就会从Halt模式唤醒。

使用SPI将设备从停机(Halt)模式唤醒

--全双工和只发模式的半双工

当MCU处于停机(Halt)模式的情况下，SPI作为从设备，只要在进入停机(Halt)模式之前，NSS引脚还接着低电平或者SSI位仍然复位，还是可以响应通信的。

当检测到数据的第一个采样沿(由CPHA位定义)

- SPI_SR寄存器中的WKUP位被置位
- 如果SPI_ICR寄存器中的WKIE位被置位的话，还会产生中断
- 该中断可以把设备从Halt模式唤醒
- 由于恢复系统时钟需要时间，SPI会发送/接收一些数据后，才能正确通信。因此需要遵循以下的步骤：
 - 在进入停机(Halt)模式之前，把一个特殊值写入 SPI_DR。该值告诉外部的主设备这个SPI进入停机(Halt)模式。
 - 外部主设备就一直连续的发送这个特殊值，直到它收到新的数据值，表明SPI从设备已经被唤醒，并且可以正确通信了。

--只接收模式的半双工

由于恢复系统时钟的时间可能比接收数据的时间还长，所以只接收的半双工模式下，唤醒功能不能保证。也就是说会导致丢失数据。

20.3.10 SPI中断

表42 SPI中断请求

中断事件	事件标志	使能控制位	是否退出等待 (wait)模式	是否退出停机 (halt)模式
发送缓冲器空标志	TXE	TXEIE	是	否
接收缓冲器非空标志	RXNE	RXNEIE	是	否
唤醒事件标志	WKUP	WKIE	是	是
主模式错误事件	MODF	ERRIE	是	否
溢出错误	OVR		是	否
CRC错误标志	CRCERR		是	否

20.4 SPI寄存器描述

20.4.1 SPI控制寄存器 1(SPI_CR1)

地址偏移: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
LSBFIRST	SPE	BR[2:0]		MSTR	CPOL	CPHA	
rw	rw	rw	rw	rw	rw	rw	rw
位7	LSBFIRST: 帧格式 ⁽¹⁾ 0: 先发送MSB; 1: 先发送LSB。						
位6	SPE: SPI使能 ⁽¹⁾ 0: 禁止SPI设备; 1: 开启SPI设备。						
位5:3	BR[2:0]: 波特率控制 000: $f_{MASTER}/2$ 001: $f_{MASTER}/4$ 010: $f_{MASTER}/8$ 011: $f_{MASTER}/16$ 100: $f_{MASTER}/32$ 101: $f_{MASTER}/64$ 110: $f_{MASTER}/128$ 111: $f_{MASTER}/256$ 注意: 当通信正在进行的时候, 不能修改这些位。						
位2	MSTR: 主设备选择 ⁽¹⁾ 0: 配置为从设备; 1: 配置为主设备。						
位1	CPOL: 时钟极性 ⁽¹⁾ 0: 空闲状态时, SCK保持低电平; 1: 空闲状态时, SCK保持高电平。						
位0	CPHA: 时钟相位 ⁽¹⁾ 0: 数据采样从第一个时钟边沿开始; 1: 数据采样从第二个时钟边沿开始。						

- 当通信正在进行的时候, 不能修改该位。

20.4.2 SPI控制寄存器 2(SPI_CR2)

地址偏移: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
BDM	BDOE	CRCEN	CRCNEXT	保留	RXOnly	SSM	SSI
RW	RW	RW	RW		RW	RW	RW

位7	BDM: 双向数据模式使能 0: 选择双线单向数据模式; 1: 选择单线双向数据模式。
位6	BDOE: 双向模式下输出使能 该位和BDM位结合起来控制双向模式下传输的方向 0: 输入使能(只接收模式); 1: 输出使能(只发送模式)。 主模式下使用MOSI引脚, 从模式下使用MISO引脚
位5	CRCEN: 硬件CRC计算使能 0: CRC计算禁止; 1: CRC计算使能。 注意: 正确的操作是先关闭SPI (SPE位置0) 再写该位
位4	CRCNEXT: 接着发送CRC 0: 下一个发送的数据来自Tx缓冲区 1: 下一个发送的数据来自Tx CRC计数器
位3	保留, 保持清零
位2	RXONLY: 只接收 0: 全双工(同时发送和接收); 1: 输出禁止(只接收)。 该位和BDM一起选择双线单向模式下传输的方向 此位还可用于多从系统中, 该从设备不被访问时, 被访问的其他从设备的输出不会被破坏
位1	SSM: 软件从设备管理 0: 禁止软件从设备管理; 1: 使能软件从设备管理 当该位被置位时, SSI位的值代替NSS引脚的输入控制从设备的选择
位0	SSI: 内部从设备选择 只有SSM被置位的情况下, 该位才有效。NSS引脚上电平决定于该位的值, 而不是NSS引脚上I/O端口的值。 0: 从模式; 1: 主模式。

20.4.3 SPI 中断控制寄存器(SPI_ICR)

地址偏移: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
TXIE	RXIE	ERRIE	WKIE			保留	
rw	rw	rw	rw				
位7	TXIE: Tx缓冲空中断使能 0: TXE中断禁止; 1: TXE中断使能。当TXE标志置位时, 允许产生中断请求。 注意: 为了正确地运行, TXIE为不要同时被置为1。						
位6	RXIE: Rx缓冲非空中断使能 0: RxNE中断禁止; 1: RxNE中断使能。当RXNE标志置位时, 允许产生中断请求。 注意: 为了正确地运行, RXIE为不要同时被置为1。						
位5	ERRIE: 错误中断使能 0: 错误中断禁止; 1: 错误中断使能。当出现错误情况时(CRCERR,OVR,MODF), 允许产生中断请求。						
位4	WKIE: 唤醒中断使能 0: 唤醒中断禁止; 1: 唤醒中断使能。当WKUP标志置位时, 允许产生中断请求。						
位3: 0	保留, 强制为0						

20.4.4 SPI 状态寄存器(SPI_SR)

地址偏移: 0x03

复位值: 0x02

7	6	5	4	3	2	1	0
BSY	OVR	MODF	CRCERR	WKUP	保留	TXE	RxNE
r	rc_w0	rc_w0	rc_w0	rc_w0	r	r	r
位7	BSY: 总线忙标志 0: SPI空闲; 1: SPI正忙于通信; 或者Tx缓冲区非空。 该位由硬件置位或清零。 注意: 单线双向主接收模式下, 禁止查询BSY标志。						
位6	OVR: 溢出标志 0: 没有发生溢出错误; 1: 发生溢出错误。 该位由硬件置位, 由软件序列清零。						
位5	MODF: 模式错误 0: 没有发生模式错误; 1: 发生模式错误。 该位由硬件置位, 由软件序列清零。						
位4	CRCERR: CRC错误标志 0: 收到的CRC值和SPI_RXCRCR值匹配; 1: 收到的CRC值和SPI_RXCRCR值不匹配。 该位由硬件置位, 由软件序列清零。						
位3	WKUP: 唤醒标志 0: 没有发生唤醒事件 1: 发生唤醒事件 当STM8处于Halt模式并且配置为从模式, 在SCK的第一个采样沿, 该位置位。 软件写零清除该位。						
位2	保留, 强制为0						
位1	TXE: 发送缓冲区空 0: 发送缓冲区非空 1: 发送缓冲区空						
位0	RXNE: 接收缓冲区非空 0: 接收缓冲区空 1: 接收缓冲区非空						

20.4.5 SPI 数据寄存器(SPI_DR)

地址偏移: 0x04

复位值: 0x00

7 6 5 4 3 2 1 0

DR[7:0]

RW RW RW RW RW RW RW RW

位7:0

DR[7:0]: 数据寄存器

待发送或者已经收到的数据。

数据寄存器对应两个缓冲区：一个用于写（发送缓冲）；另外一个用于读（接收缓冲）。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。

20.4.6 SPI CRC多项式寄存器(SPI_CRCPR)

地址偏移: 0x05

复位值: 0x07

7	6	5	4	3	2	1	0
CRCPOLY[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW
位7:0	CRCPOLY[7:0]: CRC多项式寄存器 该寄存器包含了CRC计算时用到的多项式。其复位值为07h, 根据应用可以设置其他数值。						

20.4.7 SPI Rx CRC寄存器(SPI_RXCRCR)

地址偏移: 0x06

复位值: 0x00

7	6	5	4	3	2	1	0
RxCRC[7:0]							
r	r	r	r	r	r	r	r
位7:0	RXCRC[7:0] : 接收CRC寄存器 在启用CRC计算时, RXCRC[7:0]中包含了依据收到的字节计算的CRC数值。当在SPI_CR2的CRCEN位写入'1'时, 该寄存器被复位。CRC计算使用SPI_CRCPR中的多项式。 注意: 当BSY标志为'1'时读该寄存器, 将可能读到不正确的数值。						

20.4.8 SPI Tx CRC寄存器(SPI_TXCRCR)

地址偏移: 0x07

复位值: 0x00

7	6	5	4	3	2	1	0
TxCRC[7:0]							
r	r	r	r	r	r	r	r
位7:0	TxCRC[7:0]: 发送CRC寄存器 在启用CRC计算时, TxCRC[7:0]中包含了依据将要发送的字节计算的CRC数值。当在SPI_CR2中的CRCEN位写入'1'时, 该寄存器被复位。CRC计算使用SPI_CRCPR中的多项式。 注意: 当BSY标志为'1'时读该寄存器, 将可能读到不正确的数值。						

20.5 SPI 寄存器地址映象以及复位值

表43 SPI寄存器列表及其复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	SPI_CR1	LSBFIRST	SPE	BR2	BR1	BR0	MSTR	CPOL	CPHA
	复位值	0	0	0	0	0	0	0	0
0x01	SPI_CR2	BDM	BDOE	CRCEN	CRCNEXT	保留	RXONLY	SSM	SSI
	复位值	0	0	0	0	0	0	0	0
0x02	SPI_ICR	TXIE	RXIE	ERRIE	WKIE	保留	保留	保留	保留
	复位值	0	0	0	0	0	0	0	0
0x03	SPI_SR	BSY	OVR	MODF	CRCERR	WKUP	保留	TXE	RXNE
	复位值	0	0	0	0	0	0	1	0
0x04	SPI_DR	MSB	—	—	—	—	—	—	LSB
	复位值	0	0	0	0	0	0	0	0
0x05	SPI_CRCPR	MSB	—	—	—	—	—	—	LSB
	复位值	0	0	0	0	0	1	1	1
0x06	SPI_RXCRCR	MSB	—	—	—	—	—	—	LSB
	复位值	0	0	0	0	0	0	0	0
0x07	SPI_TXCRCR	MSB	—	—	—	—	—	—	LSB
	复位值	0	0	0	0	0	0	0	0

21 I²C接口

21.1 I²C简介

I²C(芯片间)总线接口连接微控制器和串行I²C总线。它提供多主功能，控制所有I²C总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式。

21.2 I²C主要特点

- 并行总线/I²C总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I²C主设备功能
 - 产生时钟
 - 产生起始和停止信号
- I²C从设备功能
 - 可编程的 I²C 地址检测
 - 停止位检测
- 产生和检测7位/10位地址和广播呼叫
- 支持不同的通讯速度
 - 标准速度(最高 100 kHz)
 - 快速(最高 400 kHz)
- 状态标志：
 - 发送器/接收器模式标志
 - 字节发送结束标志
 - I²C 总线忙标志
- 错误标志
 - 主模式时的仲裁失败
 - 地址/数据传输后的应答(ACK)错误
 - 检测到错误的起始或停止条件
 - 禁止时钟展宽功能时数据过载或欠载
- 3种中断
 - 1 个通讯中断
 - 1 个出错中断
 - 1 个唤醒中断
- 唤醒功能
 - 从模式下如果检测到地址匹配可以将 MCU 从低功耗模式中唤醒
- 可选的时钟展宽功能

21.3 I²C简介

I²C模块不仅可以接收和发送数据，还可以在接收时将数据从串行转换成并行数据，在发送时将数据从并行转换成串行数据。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到I²C总线。允许连接到标准(最高100kHz)或快速(最高400kHz)的I²C总线。

模式选择

接口可以下述4种模式中的一种运行：

- 从设备发送模式
- 从设备接收模式

- 主设备发送模式
- 主设备接收模式

默认条件下，该模块工作于从模式。接口在产生起始条件后自动地从从模式切换到主模式；当仲裁失败或发送STOP信号时，则从主模式切换到从模式。允许多主机功能。

通信过程

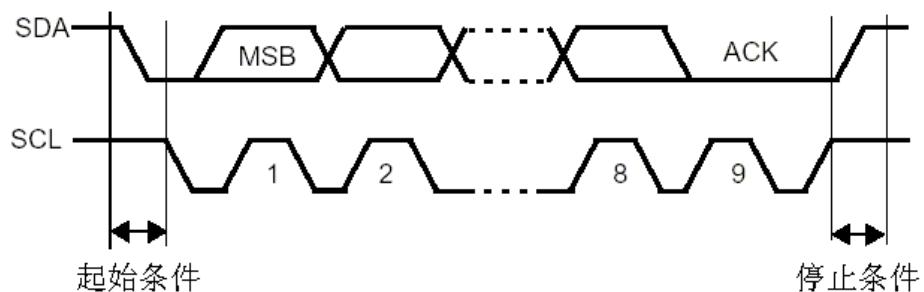
主模式时，I²C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I²C接口能识别它自己的地址(7位或10位)和广播呼叫地址。软件能够控制开启或禁止广播呼叫地址的识别。

数据和地址按8位/字节进行传输，高位在前。跟在起始条件后的1或2个字节是地址(7位模式为1个字节，10位模式为2个字节)。地址只在主模式发送。

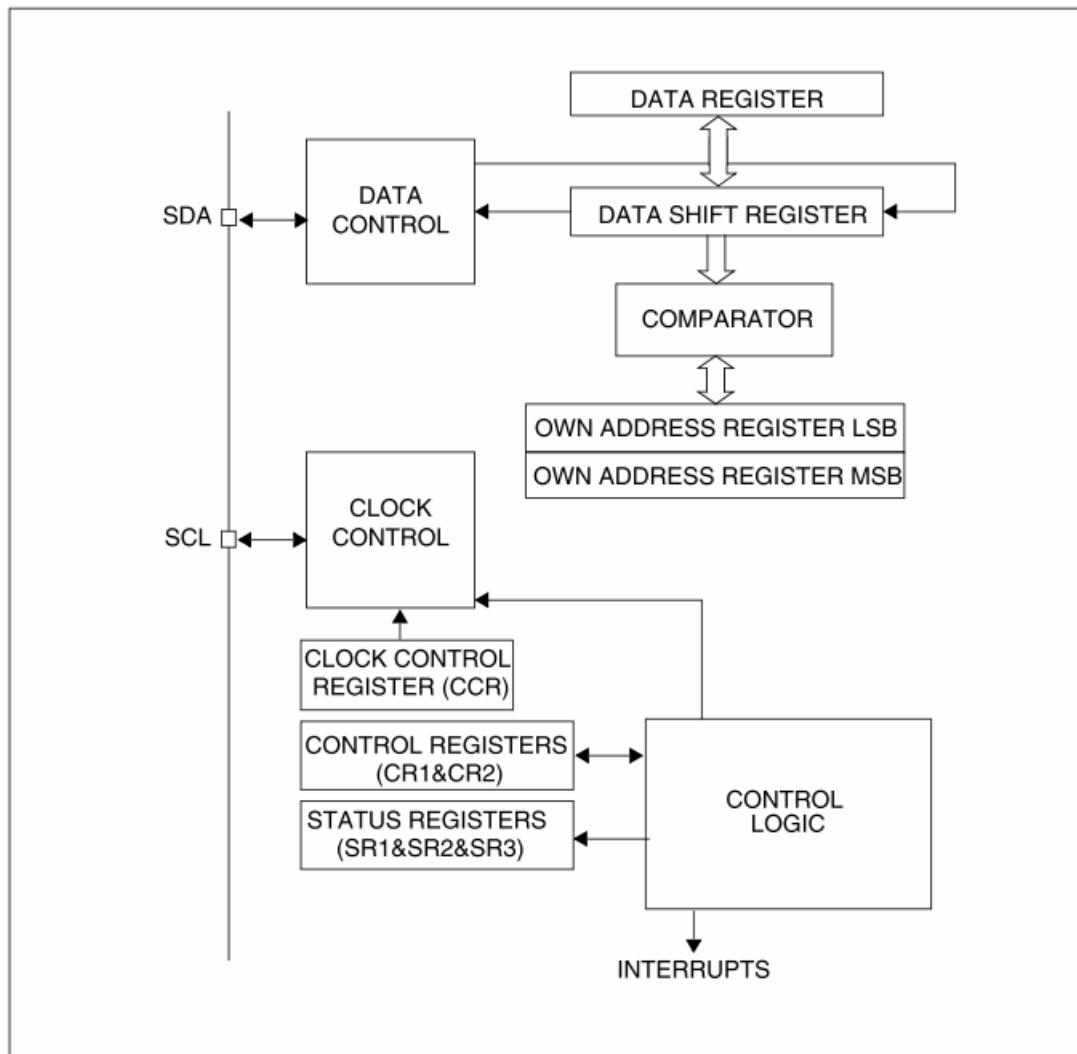
在一个字节传输的8个时钟后的第9个时钟期间，接收器必须回送一个应答位(ACK)给发送器。参考下图。

图92 I²C总线协议



软件可以开启或禁止应答(ACK)，并可以设置I²C接口的地址(7位、10位地址或广播呼叫地址)。

I²C接口的功能框图示于图93。

图93 I²C的功能框图

21.4 I²C功能描述

默认情况下, I²C接口总是工作在从模式。从默认的从模式切换到主模式, 需要产生一个起始条件。

21.4.1 I²C从模式

为了产生正确的时序, 必须在I₂C_FREQR寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是:

- 标准模式下为: 1MHz
- 快速模式下为: 4MHz

一旦检测到起始条件, 在SDA线上接收到的地址被送到移位寄存器。然后与芯片自己的地址OARLSB和OAR2或者广播呼叫地址(如果ENG_C=1)相比较。

注: 在10位地址模式时, 比较包括头段序列(11110xx0), 其中的xx是地址的两个最高有效位。

头段或地址不匹配: I²C接口将其忽略并等待另一个起始条件。

头段匹配(仅10位模式): 如果ACK位被置'1', I²C接口产生一个应答脉冲并等待8位从地址。

地址匹配: I²C接口产生以下时序:

- 如果ACK被置'1', 则产生一个应答脉冲
- 硬件将ADDR位置为1; 如果设置了ITEVFEN位, 则产生一个中断

在10位模式，接收到地址序列后，从设备总是处于接收模式。当接收到重复的起始条件，接着后面跟随与地址匹配的头序列并且最低位为'1'(即11110xx1)后，设备进入发送模式。

在从模式下TRA位指示当前是处于接收模式还是发送模式。

从发送模式

在接收到地址和清除ADDR位后，从设备将字节从DR寄存器经由内部移位寄存器发送到SDA线上。

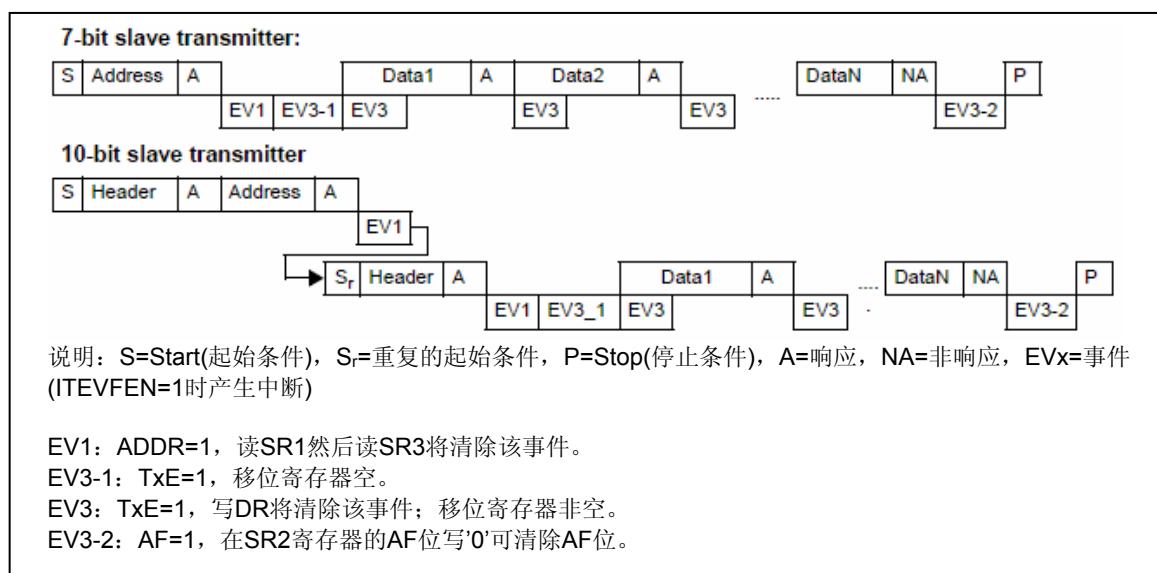
从设备保持SCL为低电平，直到ADDR位被清除并且待发送数据已写入DR寄存器。(见图94中的EV1和EV3)。

当收到应答脉冲时：

- TxE位被硬件置为1，如果设置了ITEVFEN和ITBUFEN位，则产生一个中断。

如果TxE位为1，但在上一次数据发送结束之前没有新数据写入到DR寄存器，则BTF位被置为1，I²C接口将保持SCL为低电平，以等待DR寄存器写操作。

图94 从发送器的传送序列图



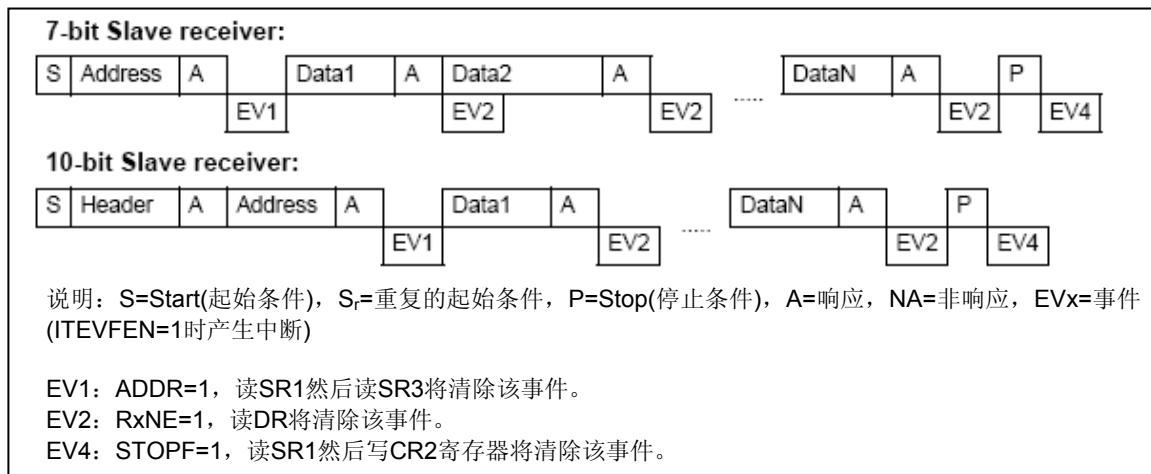
从设备接收模式

在接收到地址并清除ADDR后，从接收器将通过内部移位寄存器从SDA线接收到的字节存进DR寄存器。I²C接口在接收到每个字节后都执行下列操作：

- 如果设置了ACK位，则产生一个应答脉冲
- 硬件设置RxNE=1。如果设置了ITEVFEN和ITBUFEN位，则产生一个中断。

如果RxNE为1，并且在接收新的数据结束之前DR寄存器未被读出，BTF位被置位，I²C接口保持SCL为低电平，等待DR寄存器读操作(见下图)。

图95 从接收器的传送序列图



关闭从模式通信

在传输完最后一个数据字节后, 主设备产生一个停止条件, I²C接口检测到这一条件时:

- 设置STOPF=1, 如果设置了ITEVFEN位, 则产生一个中断。
然后I²C接口等待读SR1寄存器, 再写CR1寄存器。(见图95的EV4)。

21.4.2 I²C主模式

在主模式时, I²C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过START位在总线上产生了起始条件, 设备就进入了主模式。

以下是主模式所要求的操作顺序:

- 在I2C_FREQR寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程I2C_CR1寄存器启动外设
- 置I2C_CR1寄存器中的START位为1, 产生起始条件
- I²C模块的输入时钟频率必须至少是:
 - 标准模式下为: 1MHz
 - 快速模式下为: 4MHz

起始条件

当BUSY=0时, 设置START=1, I²C接口将产生一个开始条件并切换至主模式(M/SL位置为1)。

注: 在主模式下, 设置START位将在当前字节传输完后由硬件产生一个重开始条件。

一旦发出开始条件:

- SB位被硬件置为1, 如果设置了ITEVFEN位, 则会产生一个中断。
然后主设备等待读SR1寄存器, 紧跟着将从地址写入DR寄存器(见图96和图97的EV5)。

从地址的发送

从地址通过内部移位寄存器被送到SDA线上。

- 在10位地址模式时, 发送一个头段序列产生以下事件:
 - ADD10位被硬件置为1, 如果设置了ITEVFEN位, 则产生一个中断。
然后主设备等待程序读取SR1寄存器, 并将第二个地址字节写入DR寄存器(见图96和图97
传送序列EV9)。

- ADDR位被硬件置为1, 如果设置了ITEVFEN位, 则产生一个中断。
- 在7位地址模式时, 只需送出一个地址字节。
一旦该地址字节被送出,
- ADDR位被硬件置为1, 如果设置了ITEVFEN位, 则产生一个中断。
随后主设备等待程序一次读SR1寄存器, 跟着读SR3寄存器(见[图96](#)和[图97](#)传送序列EV6)。

根据送出从设备地址的最低位, 主设备决定进入发送模式还是进入接收模式。

- 在7位地址模式时,
 - 要进入发送模式, 主设备发送从地址时置最低位为'0'。
 - 要进入接收模式, 主设备发送从地址时置最低位为'1'。
 - 在10位地址模式时
 - 要进入发送模式, 主设备先送头字节, 然后送最低位为'0'的从地址。
 - 要进入接收模式, 主设备先送头字节, 然后送最低位为'0'的从地址。接着再发送一个重复开始条件, 后面跟着一个和地址匹配的头字节, 并且最低位为'1'(11110xx1)。
- TRA位指示主设备是在接收模式还是发送模式。

主设备发送模式

在发送了地址和清除了ADDR位后, 主设备通过内部移位寄存器将字节从DR寄存器发送到SDA线上。

主设备等待, 直到TxE被清除, (见[图96](#)传送序列的EV8)。

当收到应答脉冲时:

- TxE位被硬件置为1, 如果设置了INEVFEN和ITBUFEN位, 则产生一个中断。

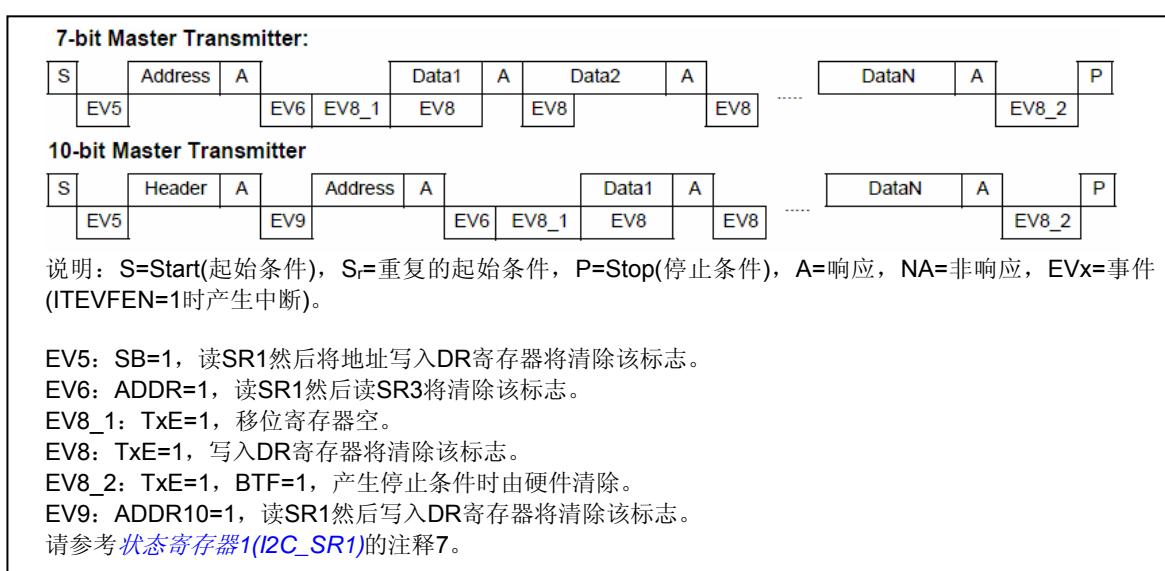
如果TxE为1并且在上一次数据发送结束之前没有写新的数据字节到DR寄存器, 则BTF被置为1, I²C接口等待BTF被清除。

关闭通信

在DR寄存器中写入最后一个字节后, 通过设置STOP位产生一个停止条件(见[图96](#)传送序列的EV8_2), 然后I²C接口将自动回到从模式(M/S位清除)。

注: 当TxE或BTF位置位时, 应该在EV8_2事件时设置停止条件。

图96 主设备发送模式发送序列图



主设备接收

在发送地址和清除ADDR之后, I²C接口进入主设备接收模式。在此模式下, I²C接口从SDA线接收数据字节, 并通过内部移位寄存器送至DR寄存器。在每个字节后, I²C接口依次执行以下操作:

- 如果ACK位被置为1, 发出一个应答脉冲。
- 硬件设置RxNE=1, 如果设置了INEVFEN和ITBUFEN位, 则会产生一个中断(见图97传送序列的EV7)。

如果RxNE位被置为1, 并且在接收新数据结束前, DR寄存器中的数据没有被读走, 硬件将设置BTF=1, I²C接口等待读DR寄存器。

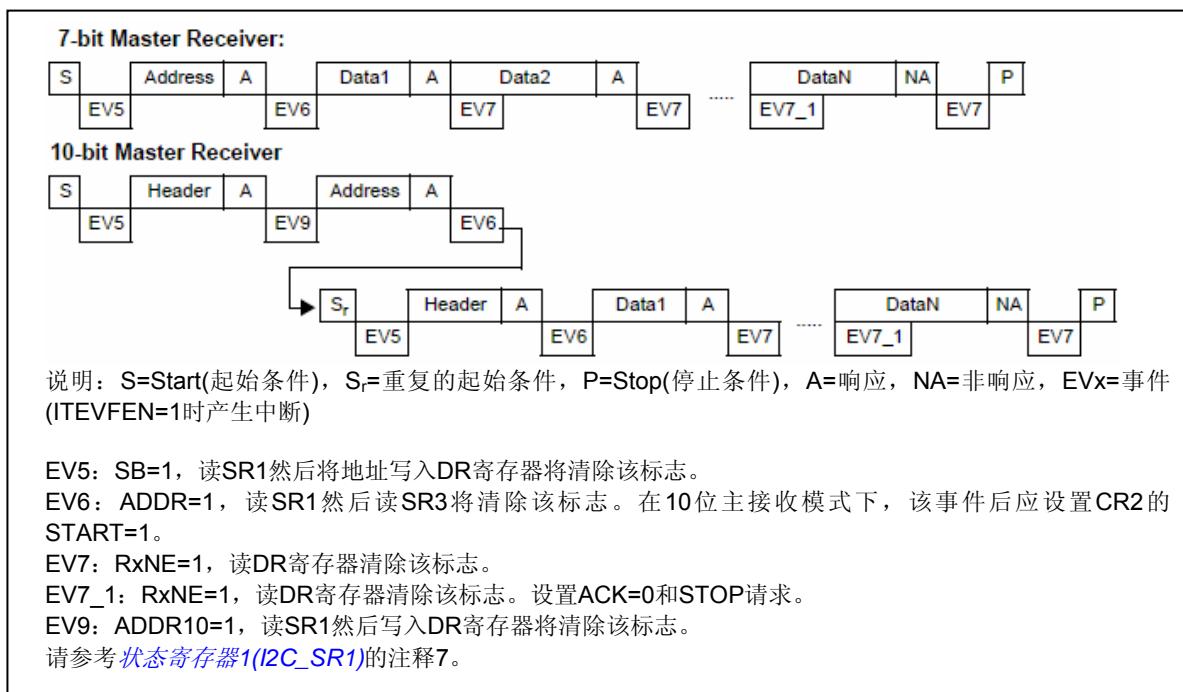
关闭通信

主设备在接收到从设备发送的最后一个字节后发送一个NACK。从设备接收到NACK后, 释放对SCL和SDA线的控制; 主设备就可以发送一个停止/重起始条件。

- 为了在收到最后一个字节后产生一个NACK脉冲, 在读倒数第二个数据字节之后(在倒数第二个RxNE事件之后)必须清除ACK位。
- 为了产生一个停止/重起始条件, 软件必须在读倒数第二个数据字节之后(在倒数第二个RxNE事件之后)设置STOP/START位。

在产生了停止条件后, I²C接口自动回到从模式(M/SL位被清除)。

图97 主设备接收模式接收序列图



21.4.3 出错状态

以下出错状态可能造成通讯失败。

总线错误(BERR)

在一个字节传输期间, 当I²C接口检测到一个停止或起始条件则产生总线错误。此时:

- BERR位被置为1, 如果设置了ITERREN位, 则产生一个中断;
- 在从模式情况下, 数据被丢弃, 硬件释放总线:
 - 如果是错误的开始条件, 从设备认为是一个重启动, 并等待地址或停止条件。
 - 如果是错误的停止条件, 从设备按正常的停止条件操作, 同时硬件释放总线。

- 在主模式下，停止条件必须由程序产生。

应答错误(AF)

当接口检测到一个无应答位时，产生应答错误。此时：

- AF位被置为1，如果设置了ITERREN位，则产生一个中断；
- 当发送设备接收到一个NACK时，必须复位通讯：
 - 如果是处于从模式，硬件释放总线。
 - 如果是处于主模式，软件必须生成一个停止条件。

仲裁失败(ARLO)

当I²C接口检测到仲裁失败时产生仲裁失败错误，此时：

- ARLO位被硬件置为1，如果设置了ITERREN位，则产生一个中断；
- I²C接口自动回到从模式(M/SL位被清除)；
- 硬件释放总线。

过载/欠载错误(OVR)

在从模式下，如果禁止时钟展宽并且I²C接口正在接收数据可能会产生过载错误。当接口接收到一个字节之前已经收到了一个字节(RxNE=1)并且DR中的数据仍没有被读取时：

- 最后接收的数据被丢弃；
- 在过载错误时，软件应清除RxNE位，发送方应该重新发送最后一次发送的字节。

在从模式下，如果禁止时钟展宽并且I²C接口正在发送数据可能会产生欠载错误。接口在应该发送下一个字节的时钟边沿来到时仍然没有更新DR(TxE=1)，此时：

- 在DR寄存器中的前一个字节将被重复发出；
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按I²C总线标准在规定的时钟为低的时间内更新DR寄存器。

21.4.4 SDA/SCL线控制

- 如果允许时钟展宽：
 - 发送模式：如果 TxE=1 且 BTF=1：I²C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器(缓冲器和移位寄存器都是空的)。
 - 接收模式：如果 RxNE=1 且 BTF=1：I²C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR(缓冲器和移位寄存器都是满的)。
- 如果在从模式中禁止时钟延长：
 - 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载错。接收到的最后一个字节丢失。
 - 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载错。相同的字节将被重复发出。
 - 不控制写冲突。

21.5 低功耗模式

表44 低功耗模式下的I²C接口

模式	描述
等待 (WAIT)	对I ² C接口没有影响 I ² C中断可以把设备从等待(Wait)模式退出
停机 (HALT)	从设备模式：除了控制寄存器，通信被复位。设备处于从模式。 如果ITEVTEN=1并且地址匹配（包括地址头序列，如果有的话），则产生从停机(Halt)唤醒中断。

	<p>在停机(Halt)模式下，匹配的地址不会被应答，所以主设备需要在CPU被唤醒后重新发送地址，以收到应答。</p> <p>如果NOSTRETCH=0，在停机(Halt)模式下收到应答脉冲后会延长时钟，直到软件把WUFH清零。</p> <p>唤醒CPU的地址字节不会置位任何一个标志</p> <p>主设备模式：通信被冻结，直到CPU被唤醒。从停机(Halt)唤醒，有相应标志置位，如果ITEVTEN=1还会产生中断并且有一个HALT指令</p> <p>注意：当通信正在进行时，禁止进入停机(Halt)模式</p>
--	---

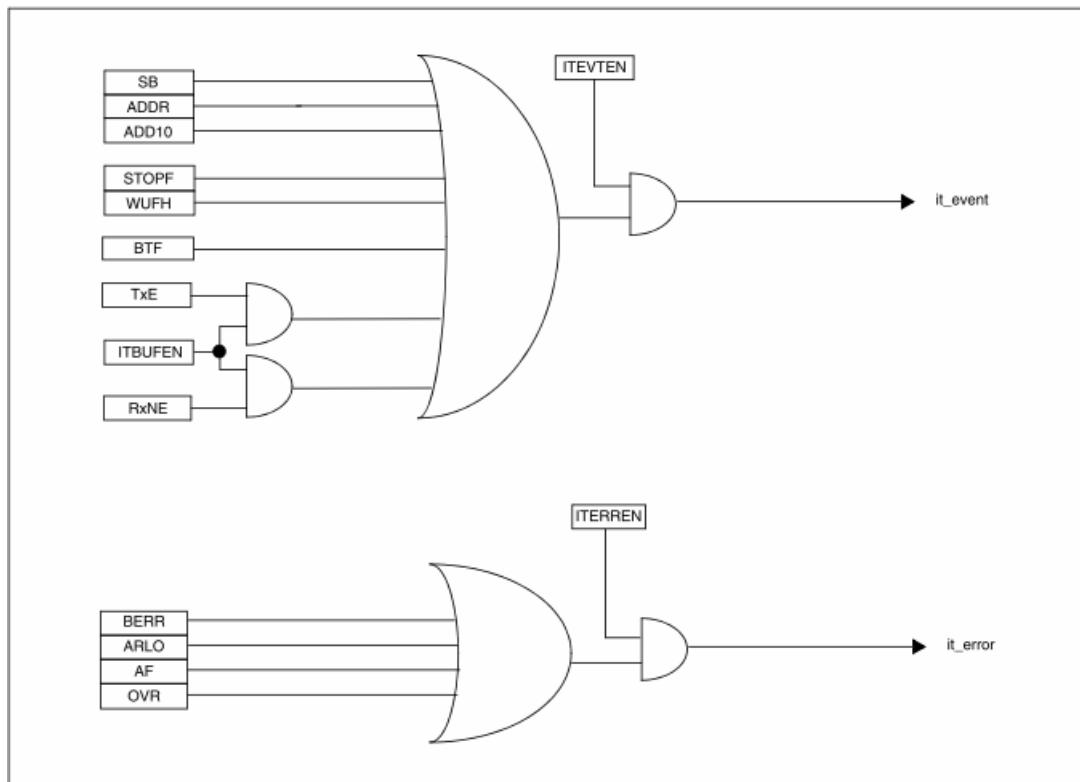
21.6 I²C中断请求

表45 I²C中断请求表：

中断事件	事件标志	开启控制位	退出wait	退出halt
起始位已发送(主)	SB	ITEVFEN	是	否
地址已发送(主) 或 地址匹配(从)	ADDR		是	否
10位头段已发送(主)	ADD10		是	否
已收到停止(从)	STOPF		是	否
数据字节传输完成	BTF		是	否
从halt唤醒	WUFH	ITEVFEN	是	是
接收缓冲区非空	RxNE	ITEVFEN 和 ITBUFEN	是	否
发送缓冲区空	TxE		是	否
总线错误	BERR	ITERREN	是	否
仲裁丢失(主)	ARLO		是	否
响应失败	AF		是	否
过载/欠载	OVR		是	否

注：

1. SB、ADDR、ADD10、STOPF、BTF、RxNE和TxE通过逻辑或汇到同一个中断通道中。
2. BERR、ARLO、AF和OVR通过逻辑或汇到同一个中断通道中。
3. WUFH使用另外一个中断通道。
4. 之前的3个通道可以或到同一个。

图98 I²C中断映射图

21.7 I²C寄存器描述

21.7.1 控制寄存器1(I2C_CR1)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
NOSTRETCH	ENGC		保留			PE	
RW	RW						RW
位7	NOSTRETCH: 时钟延展禁止(从模式) 该位用在从模式下当ADDR或者BTF标志置位时, 是否禁止时钟延展, 直到软件将该位复位。 0: 时钟延展使能; 1: 时钟延展禁止。						
位6	ENGC: 广播呼叫使能 0: 广播呼叫禁止, 对地址00h不响应; 1: 广播呼叫使能, 对地址00h响应。						
位5: 1	保留, 读出0						
位0	PE: I ² C模块使能 0: 禁用I ² C模块; 1: 启用I ² C模块: 相应的I/O口需配置为复用功能。 注: 如果清除该位时通讯正在进行, 在当前通讯结束后, I ² C模块被禁用并返回空闲状态。 由于PE=0, 通讯结束后所有的位被置为0。						

21.7.2 控制寄存器 2(I2C_CR2)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
SWRST	保留		POS	ACK	STOP	START	

rw

rw

rw

rw

rw

位7	<p>SWRST: 软件复位</p> <p>当该位置为1, I2C模块处于复位状态。确保I2C总线被释放, 并且总线空闲, 然后再复位该位</p> <p>0: I2C模块不在复位状态; 1: I2C模块处于复位状态。</p> <p>注意: 可以用于当STOP没有被检测到, 而使得BUSY位一直置为1的情况</p>
位6:4	保留位, 读为0。
位3	<p>POS: 应答的位置(接收数据时)</p> <p>该位可以被软件置位或者清零; 也可以当PE=0时被硬件清零。</p> <p>0: ACK位控制被移位寄存器正在接收的这个当前字节的应答或者不应答; 1: ACK位控制下一个将被移位寄存器接收的字节的应答或者不应答</p> <p>注意: 该位必须在数据接收开始前配置</p>
位2	<p>ACK: 应答使能</p> <p>该位可以被软件置位或者清零; 也可以当PE=0时被硬件清零。</p> <p>0: 不返回应答; 1: 收到一个字节后(匹配的地址字节或者数据字节)后返回应答。</p>
位1	<p>STOP: 停止位产生</p> <p>该位可以被软件置位或者清零; 硬件也可以测到停止位后将该位清零; 当超时错位被检测到时, 硬件也会将该位置为1。</p> <p>主模式: 0: 不产生停止位 1: 当前字节传输完成后, 或者当前起始位发送完后, 产生停止位。</p> <p>从模式: 0: 没有停止位 1: 当前字节传输完后, 释放SCL和SDA线。</p>
位0	<p>START: 起始位产生</p> <p>该位可以被软件置位或者清零; 也可以当PE=0时被硬件清零, 或者起始位发送完后由硬件清零。</p> <p>主模式: 0: 不产生起始位 1: 产生重复起始位</p> <p>从模式: 0: 不产生起始位 1: 当总线空闲时产生起始位</p>

21.7.3 频率寄存器(I2C_FREQR)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
保留	FREQ[5:0]						
r	r	rw	rw	rw	rw	rw	rw
位7:6	保留位, 读为0。						
位5:0	FREQ[5:0]: 外设时钟频率 ⁽¹⁾ 为了产生正确的时序, 必须配置合适的输入时钟频率: 允许的时钟范围在1MHz和50MHz之间 000000: 不允许 000001: 1MHz 000010: 2MHz ... 110010: 50MHz 不允许更高的值						

1. I2C总线时序要求最小的外设时钟频率为: 标准模式1MHz; 快速模式4MHz



21.7.4 自身地址寄存器LSB(I2C_OARL)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
ADD[7:1]							ADD0
RW	RW	RW	RW	RW	RW	RW	RW
位7:1	ADD [7:1]: 接口地址 地址的7:1位						
位0	ADD0: 接口地址 7位地址模式: 此位不用关心; 10位地址模式: 地址的位0						

21.7.5 自身地址寄存器MSB(I2C_OARH)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
ADDMODE	ADDCONF		保留		ADD[9:8]		保留
rw	rw	r	r	r	rw		r
位7	ADDMODE: 寻址模式(从模式) 0: 7位从地址(对10位地址不响应) 1: 10位从地址(对7位地址不响应)						
位6	ADDCONF: 地址模式配置 软件必须配置该位(只能配置成1)						
位5: 3	保留, 读出0						
位2: 1	ADD[9:8] 接口地址 10位寻址模式: 地址9: 8位						
位0	保留, 读出0						

21.7.6 数据寄存器(I2C_DR)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
DR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw
位7:0	DR[7:0]: 数据寄存器 ⁽¹⁾⁽²⁾⁽³⁾ 用于存放接收到的数据或放置用于发送到总线的数据 发送模式: 当写一个字节至DR寄存器时, 自动启动数据传输。一旦传输开始(TxE=1), 如果能及时把下一个需传输的数据写入DR寄存器, I ² C模块将保持连续的数据流。 接收模式: 接收到的字节被拷贝到DR寄存器(RxNE=1)。						

1. 在从模式下, 地址不会被拷贝进数据寄存器;
2. 硬件不管理写冲突(如果 TxE=0, 仍能写入数据寄存器);
3. 如果在处理 ACK 脉冲时发生 ARLO 事件, 接收到的字节不会被拷贝到数据寄存器里, 因此不能读到它。



21.7.7 状态寄存器 1(I2C_SR1)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
TxE	RxNE	保留	STOPF	ADD10	BTF	ADDR	SB
r	r	r	r	r	r	r	r
位7	TxE : 数据寄存器为空(发送时) ⁽¹⁾ 0: 数据寄存器非空; 1: 数据寄存器空。 – 在发送数据时, 数据寄存器为空时该位被置'1', 在发送地址阶段不设置该位。 – 软件写数据到DR寄存器可清除该位; 或在发生一个起始或停止条件后, 或当PE=0时由硬件自动清除。						
位6	RxNE : 数据寄存器非空(接收时) ⁽²⁾⁽³⁾ 0: 数据寄存器为空; 1: 数据寄存器非空。 – 在接收时, 当数据寄存器不为空, 该位被置'1'. 在接收地址阶段, 该位不被置位。 – 软件对数据寄存器的读写操作清除该位, 或当PE=0时由硬件清除。						
位5	保留位, 硬件强制为0						
位4	STOPF : 停止条件检测位(从模式) ⁽⁴⁾ 0: 没有检测到停止条件; 1: 检测到停止条件。 – 在一个应答之后(如果ACK=1), 当从设备在总线上检测到停止条件时, 硬件将该位置'1'。 – 软件读取SR1寄存器后, 对CR2寄存器的写操作将清除该位, 或当PE=0时, 硬件清除该位。						
位3	ADD10 : 10位头序列已发送(主模式) ⁽⁵⁾ 0: 没有ADD10事件发生; 1: 主设备已经将第一个地址字节发送出去。 – 在10位地址模式下, 当主设备已经将第一个字节发送出去时, 硬件将该位置'1'。 – 软件读取SR1寄存器, 接着将第二个地址字节写入DR, 可以清除该位; 或当PE=0时, 硬件清除该位。						
位2	BTF : 字节发送结束 ⁽⁶⁾⁽⁷⁾ 0: 数据字节发送未完成; 1: 数据字节发送结束。 – 当NOSTRETCH=0时, 在下列情况下硬件将该位置'1': – 在接收时, 当收到一个新字节(包括ACK脉冲)且数据寄存器还未被读取(RxNE=1)。 – 在发送时, 当一个新数据将被发送且数据寄存器还未被写入新的数据(TxE=1)。 – 在软件读取SR1寄存器后, 对数据寄存器的读或写操作将清除该位; 或在传输中发送一个起始或停止条件后, 或当PE=0时, 由硬件清除该位。						

位1	<p>ADDR: 地址已被发送(主模式)/地址匹配(从模式)⁽⁷⁾</p> <p>在软件读取SR1寄存器后, 对SR3寄存器的读操作将清除该位, 或当PE=0时, 由硬件清除该位。</p> <p>地址匹配(从模式)</p> <p>0: 地址不匹配或没有收到地址; 1: 收到的地址匹配。</p> <ul style="list-style-type: none"> 当收到的从地址与OAR寄存器中的内容相匹配、或发生广播呼叫时(当对应的设置被使能时), 硬件就将该位置'1'。 <p>地址已被发送(主模式)</p> <p>0: 地址发送没有结束; 1: 地址发送结束。</p> <ul style="list-style-type: none"> 10位地址模式时, 当收到地址的第二个字节的ACK后该位被置'1'。 7位地址模式时, 当收到地址的ACK后该位被置'1'。 <p>注: 在收到NACK后, ADDR位不会被置位。</p>
位0	<p>SB: 起始位(主模式)⁽⁷⁾</p> <p>0: 未发送起始条件; 1: 起始条件已发送。</p> <ul style="list-style-type: none"> 当发送出起始条件时该位被置'1'。 <p>软件读取SR1寄存器后, 写数据寄存器的操作将清除该位, 或当PE=0时, 硬件清除该位。</p>

- 接收到 ACK 脉冲后, 当 DR 中的内容被拷贝到移位寄存器时, 会产生中断; 如果收到的是 NACK, 不会做拷贝的操作, TxE 也不会被置位。
- 收到 ACK 脉冲后, 当移位寄存器中的内容拷贝到 DR 中时, 也会产生中断。
- ARLO 发生的话, 就不会置位 RxNE 了。
- 收到 NACK, 不会置位 STOPF。
- 收到 NACK, 不会置位 ADD10。
- 收到 NACK 或者发生 ARLO, 不会置位 BTF。
- 如果 f_{MASTER} 小于 2MHz, 强烈建议使用中断来管理 I²C 通信。否则如果如果使用查询的方式来管理 SB, ADDR 或 BTF 标志, 必须在检测到标志位之后插入 5 个 CPU 周期, 然后再执行清除标志位的操作 (写 I2C_DR 清除 SB 位, 写或读 I2C_DR 清除 BTF 位, 读 I2C_SR3 清除 ADDR 位), 可以执行 5 个 NOP 指令来插入 5 个 CPU 周期。

21.7.8 状态寄存器 2 (I2C_SR2)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
保留	WUFH	保留	OVR	AF	ARLO	BEER	
	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位7: 6	保留位, 读出0
位5	WUFH: 从停机(Halt)模式唤醒 0: 没有从Halt模式唤醒; 1: Halt模式下, 7位地址或者地址头序列匹配; 或者是主模式下进入Halt模式。 注意: 该位在从模式下(Halt模式)的被置位是异步的。只有ITEVTEN=1时才会被置位。 -- 软件写0, 清除此位; 或者PE=0时由硬件清零。
位4	保留位, 读出0
位3	OVR: 上溢/下溢 0: 没有发生上溢/下溢; 1: 发生了上溢/下溢。 -- 当NOSTRETCH=1, 从模式, 并且满足以下条件, 由硬件置位: -- 接收时: 当DR寄存器中的内容还没有读出, 又收到新的字节(包括ACK脉冲)。新收到的字节因此被丢失。 -- 发送时: 该发新的数据了, DR寄存器还没有被写入数据。同样的字节将会被发送两次。 软件写0可以清除该位; 或者当PE=0时由硬件清零。 注意: 如果写DR的时刻非常接近于SCL的上升沿, 发送的数据是不能确定的, 并且会产生保持时间错误。
位2	AF: 应答失败 0: 没有应答失败; 1: 应答失败。 当没有返回应答时, 由硬件置为1。 软件写0清除该位; 或者当PE=0时由硬件清零。
位1	ARLO: 仲裁失败(主模式) 0: 没有检测到仲裁失败; 1: 检测到仲裁失败。 当该模块丢失了对总线的仲裁控制并转交给其他主设备, 硬件自动置位。 软件写0清除该位; 或者当PE=0时由硬件清零。 仲裁失败发生后, 模块自动切换回从模式(M/SL=0)
位0	BERR: 总线错误 0: 正常的起始或者结束条件; 1: 错误的起始或者结束条件。 当硬件检测到错误的起始或者结束条件后, 自动置为1; 软件写0清除该位; 或者当PE=0时由硬件清零。

21.7.9 状态寄存器 3 (I2C_SR3)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
	保留		GENCALL	保留	TRA	BUSY	MSL
r	r	r	r	r	r	r	r
位7: 5 保留位, 读出0							
位4 GENCALL: 广播呼叫头序列(从模式) 0: 没有广播呼叫; 1: 当ENGC=1时收到了广播呼叫地址头序列。 -- 总线上出现结束或者重复起始条件后, 或者在PE=0时, 由硬件清零。							
位3 保留位, 读出0							
位2 TRA: 发送器/接收器 0: 接收数据; 1: 发送数据。 该位在整个寻址阶段结束时, 根据地址字节的R/W位来决定。 当检测到结束条件(STOPF=1), 重复起始条件, 总线仲裁失败(ARLO=1), 或者PE=0时, 由硬件清零。							
位1 BUSY: 总线忙 0: 总线上没有通信; 1: 总线上有通信。 -- 硬件检测到SDA或者SCL变成低电平, 该位置位; -- 检测到结束条件时, 硬件清零该位。 该位表明总线上时候正有通信在进行。即使模块没有使能的情况下(PE=0), 该位也有效。							
位0 MSL: 主/从模式 0: 从模式; 1: 主模式。 -- 当模块处于主模式(SB=1), 硬件置位; -- 检测到总线上出现结束条件, 或仲裁失败, 或者当PE=0时, 由硬件清零。							

21.7.10 中断寄存器 (I2C_ITR)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
					ITBUFEN	ITEVTEN	ITERREN
r	r	r	rw	rw	rw	rw	rw
位7: 3	保留						
位2	ITBUFEN: 缓冲中断使能 0: TxE=1或者RxNE=1不产生任何中断; 1: TxE=1或者RxNE=1产生事件中断。						
位1	ITEVTEN: 事件中断使能 0: 事件中断禁止; 1: 事件中断使能。 发生以下事件时, 产生中断: -- SB=1(主模式) -- ADDR=1(主/从模式) -- ADD10=1(主模式) -- STOPF=1(从模式) -- BTF=1, 而没有TxE或者RxNE中断 -- TxE事件, 如果ITBUFEN=1 -- RxNE事件, 如果ITBUFEN=1 -- WUFH=1(从halt模式唤醒的异步中断)						
位0	ITERREN: 错误中断使能 0: 错误中断禁止; 1: 错误中断使能。 当发生以下错误时, 产生中断: -- BERR=1; -- ARLO=1; -- AF=1; -- OVR=1。						

21.7.11 时钟控制寄存器低位部分(I2C_CCRL)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
CCR[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW
位7:0	<p>CCR[7:0]: 时钟控制寄存器(主模式) 控制主模式下的SCLH时钟。</p> <p><u>在I²C标准模式下:</u></p> $t_{high} = CCR \times t_{CK}$ $t_{low} = CCR \times t_{CK}$ <p><u>在I²C快速模式下:</u></p> <p>如果DUTY = 0:</p> $t_{high} = CCR \times t_{CK}$ $t_{low} = 2 \times CCR \times t_{CK}$ <p>如果DUTY = 1: (速度达到400kHz)</p> $T_{high} = 9 \times CCR \times t_{CK}$ $T_{low} = 16 \times CCR \times t_{CK}$ <p>注意:</p> <ol style="list-style-type: none"> 1. $t_{CK} = 1/f_{CK}$。 f_{CK}是由时钟控制寄存器配置的外设输入时钟。 2. 允许设定的最小值为0x04, 在快速DUTY模式下允许的最小值为0x01; 3. t_{high}包含SCL的上升边沿; 4. t_{low}包含SCL的下降边沿; 5. I²C通信速度, $f_{SCL} = 1/(T_{high} + T_{low})$; 6. 这些时序均没有过滤器。 						

21.7.12 时钟控制寄存器高位部分(I2C_CCRH)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
F/S	DUTY	保留		CCR[11:8]			
rw	rw	r				rw	
位7	F/S: I2C主模式选择 0: 标准模式I2C 1: 快速模式I2C						
位6	DUTY: 快速模式下的占空比 0: 快速模式 $t_{low}/t_{high} = 2$; 1: 快速模式 $t_{low}/t_{high} = 16/9$ (参见CCR)						
位5: 4	保留, 必须为0						
位3: 0	CCR[11:8]: 时钟控制寄存器(主模式) 控制主模式下的SCLH时钟。 <u>在I²C标准模式下:</u> $t_{high} = CCR \times t_{CK}$ $t_{low} = CCR \times t_{CK}$ <u>在I²C快速模式下:</u> 如果DUTY = 0: $t_{high} = CCR \times t_{CK}$ $t_{low} = 2 \times CCR \times t_{CK}$ 如果DUTY = 1: (速度达到400kHz) $T_{high} = 9 \times CCR \times t_{CK}$ $T_{low} = 16 \times CCR \times t_{CK}$ 比如: 标准模式下, 为了产生100kHz的SCL频率: 假设FREQR = 08, $t_{CK} = 125\text{ns}$, 那么CCR就必须为28h (28h \leftrightarrow 40d *125 ns = 5000ns) 注意: 1. t_{high} 包含SCL的上升边沿; 2. t_{low} 包含SCL的下降边沿; 3. 这些时序均没有过滤器。						

- 注意: 1 只有I2C禁止时(PE=0)才能配置CCR寄存器
2 要产生快速时钟400kHz, f_{CK} 要是10MHz的整数倍
3 要产生100kHz的标准时钟, 要求 $f_{CK} \geq 1\text{MHz}$

21.7.13 TRISE寄存器(I2C_TRISE)

地址偏移: 0x0D

复位值: 0x02

7	6	5	4	3	2	1	0
保留	TRISE[5:0]						
r	r	rw	rw	rw	rw	rw	rw
位7:6	保留位, 读为0						
位5:0	<p>TRISE[5:0]: 在快速/标准模式下的最大上升时间(主模式) 这些位必须设置为^{I2C}总线规范里给出的最大的SCL上升时间, 增长步幅为1。 例如: 标准模式中最大允许SCL上升时间为1000ns。如果在I2C_CR2寄存器中FREQ[5:0]中的值等于0x08且$T_{PCLK1}=125\text{ns}$, 故TRISE[5:0]中必须写入09h($1000\text{ns}/125\text{ ns} = 8+1$)。 滤波器的值也可以加到TRISE[5:0]内。 如果结果不是一个整数, 则将整数部分写入TRISE[5:0]以确保t_{HIGH}参数。 注: 只有当^{I2C}被禁用($PE=0$)时, 才能设置TRISE[5:0]。</p>						

21.7.14 I²C寄存器地址映射和复位值

表46 I²C寄存器地址映射

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	I2C_CR1	NO STRETCH	ENGC	—	—	—	—	—	PE
	复位值	0		0	0	0	0	0	0
0x01	I2C_CR2	SWRST	—	—	—	POS	ACK	STOP	START
	复位值	0	0	0	0	0	0	0	0
0x02	I2C_FREQR	—	—	FRWQ5	FREQ4	FREQ3	FREQ2	FREQ1	FREQ0
	复位值	0	0	0	0	0	0	0	0
0x03	I2C_OARL	ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
	复位值	0	0	0	0	0	0	0	0
0x04	I2C_OARH	ADDMODE	ADDCONF	—	—	—	ADD9	ADD8	—
	复位值	0	0	0	0	0	0	0	0
0x05		保留							
0x06	I2C_DR	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	复位值	0	0	0	0	0	0	0	0
0x07	I2C_SR1	TxE	RxNE	—	STOPF	ADD10	BTF	ADDR	SB
	复位值	0	0	0	0	0	0	0	0
0x08	I2C_SR2	—	—	WUFH	—	OVR	AF	ARLO	BERR
	复位值	0	0	0	0	0	0	0	0
0x09	I2C_SR3	—	—	—	GENCALL	—	TRA	BUSY	MSL
	复位值	0	0	0	0	0	0	0	0
0x0A	I2C_ITR	—	—	—	—	—	ITBUFEN	ITEVTEN	ITERREN
	复位值	0	0	0	0	0	0	0	0
0x0B	I2C_CCRL	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
	复位值	0	0	0	0	0	0	0	0
0x0C	I2C_CCRH	FS	DUTY	—	—	CCR11	CCR10	CCR9	CCR8
	复位值	0	0	0	0	0	0	0	0
0x0D	I2C_TRISER	—	—	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0
	复位值	0	0	0	0	0	0	1	0

22 通用异步收发器(UART)

22.1 UART介绍

STM8S微控制器家族的通用同步异步收发器(UART1, UART2或UART3)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。STM8的UART提供宽范围的波特率选择，并且支持多处理器通讯。UART也支持LIN(局部互连网)协议版本1.3, 2.0和2.1以及在主模式下的J2602。

UART1和UART2具有以下扩展特征(见表47)

- UART2和UART3支持LIN从模式。
- UART1和UART2支持同步单向通信，也支持智能卡协议和IrDA(红外数据组织)SIR ENDEC规范。
- UART1支持半双工单线通讯。

关于每个STM8微控制器型号中的可用UART配置信息，请查阅数据手册。

表47 UART配置⁽¹⁾

UART模式	UART1	UART2	UART3
异步模式	X	X	X
多处理器通讯	X	X	X
同步通讯	X	X	NA
智能卡模式	X	X	NA
IrDA	X	X	NA
半双工(单线模式)	X	NA	NA
LIN主模式	X	X	X
LIN从模式	NA	X	X

(1) X = 支持, NA = 不支持该应用

22.2 UART主要特性

- 全双工的，异步通信
- NRZ标准格式
- 高精度波特率发生器系统
 - 发送和接收共用的可编程波特率，最高达 2.5Mbits/s
- 可编程数据字长度(8位或9位)
- 可配置的停止位-支持1或2个停止位
- LIN主模式
 - LIN 断开和分隔符生成
 - 通过不同标志位和不同中断源检测 LIN 断开和分隔符，用于回读检测。
- 发送方为同步传输提供时钟(UART1, UART2)
- IRDA SIR 编码器解码器(UART1, UART2)
 - 在正常模式下支持 3/16 位的持续时间
- 智能卡模拟功能(UART1, UART2)
 - 智能卡接口支持 ISO7816-3 标准里定义的异步智能卡协议
 - 智能卡用到的 1.5 个停止位
- 单线半双工通信(UART1)

- 单独的发送器和接收器使能位
- 检测标志
 - 接收缓冲器满
 - 发送缓冲器空
 - 传输结束标志
- 奇偶校验控制
 - 发送奇偶校验位
 - 对接收数据进行校验
- 四个错误检测标志
 - 溢出错误
 - 噪音错误
 - 帧错误
 - 奇偶校验错误
- 6个带标志的中断源
 - 发送数据寄存器空
 - 发送完成
 - 接收数据寄存器满
 - 检测到总线为空闲
 - 校验错误
 - LIN 断开和分隔符检测(UART2, UART3)
- 2个中断向量
 - 发送中断
 - 接收中断
- 低功耗模式
- 多处理器通信 – 如果地址不匹配, 则进入静默模式
- 从静默模式中唤醒 (通过空闲总线检测或地址标志检测)
- 2种唤醒接收器的方式:
 - 地址位(MSB)
 - 总线空闲

22.3 UART功能概述

接口通过两个或三个引脚与其他设备连接在一起(见图99, 图100, 图101)。任何UART双向通信至少需要两个脚: 接收数据输入(UART_RX)和发送数据输出(UART_TX)。

UART_RX: 串行数据输入。使用过采样技术来区别数据和噪音, 从而恢复数据。

UART_TX: 串行数据输出。当发送器被禁止时, 输出引脚状态由其GPIO端口配置决定。当发送器被激活, 并且不发送数据时, TX引脚处于高电平。

通过这些管脚, 在普通UART模式下串行数据的发送接收帧结构组成如下:

- 总线在发送或接收前应处于空闲状态
- 一个起始位
- 一个数据字(8或9位), 最低有效位在前
- 1, 1.5, 2个的停止位, 由此表明数据帧的结束
- 一个状态寄存器(UART_SR)
- 数据寄存器(UART_DR)
- 一个16位波特率寄存器(UART_BRR)
- 一个智能卡模式下的保护时间寄存器(UART_GTR)

关于以上寄存器中每个位的具体定义, 请参考寄存器描述章节。

在同步模式中需要下列引脚:

- **UART_SK**: 发送器时钟输出。此引脚输出用于同步传输的时钟, (在Start位和Stop位上没有时钟脉冲, 可以通过软件选择在最后一个数据位送出一个时钟脉冲)。这可以用来控制带有移位寄存器的外部设备(例如LCD驱动器)。时钟相位和极性都是软件可编程的。

在IrDA模式里, 管脚**UART_RX**和**UART_TX**作用如下:

- **UART_RX** = IrDA_RDI: IrDA模式下的数据输入。
- **UART_TX** = IrDA_TDO: IrDA模式下的数据输出。

图99 UART1框图

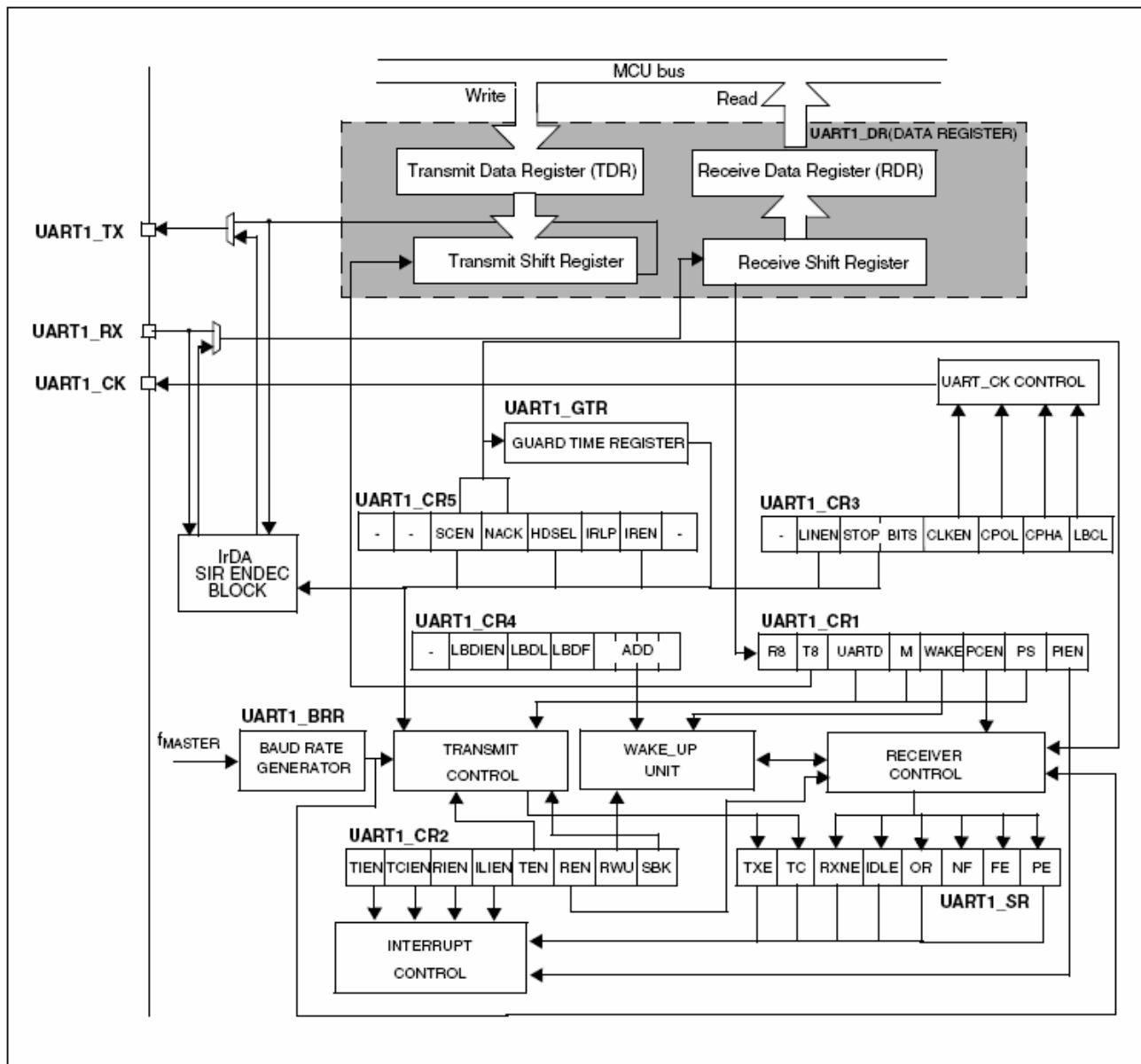


图100 UART2框图

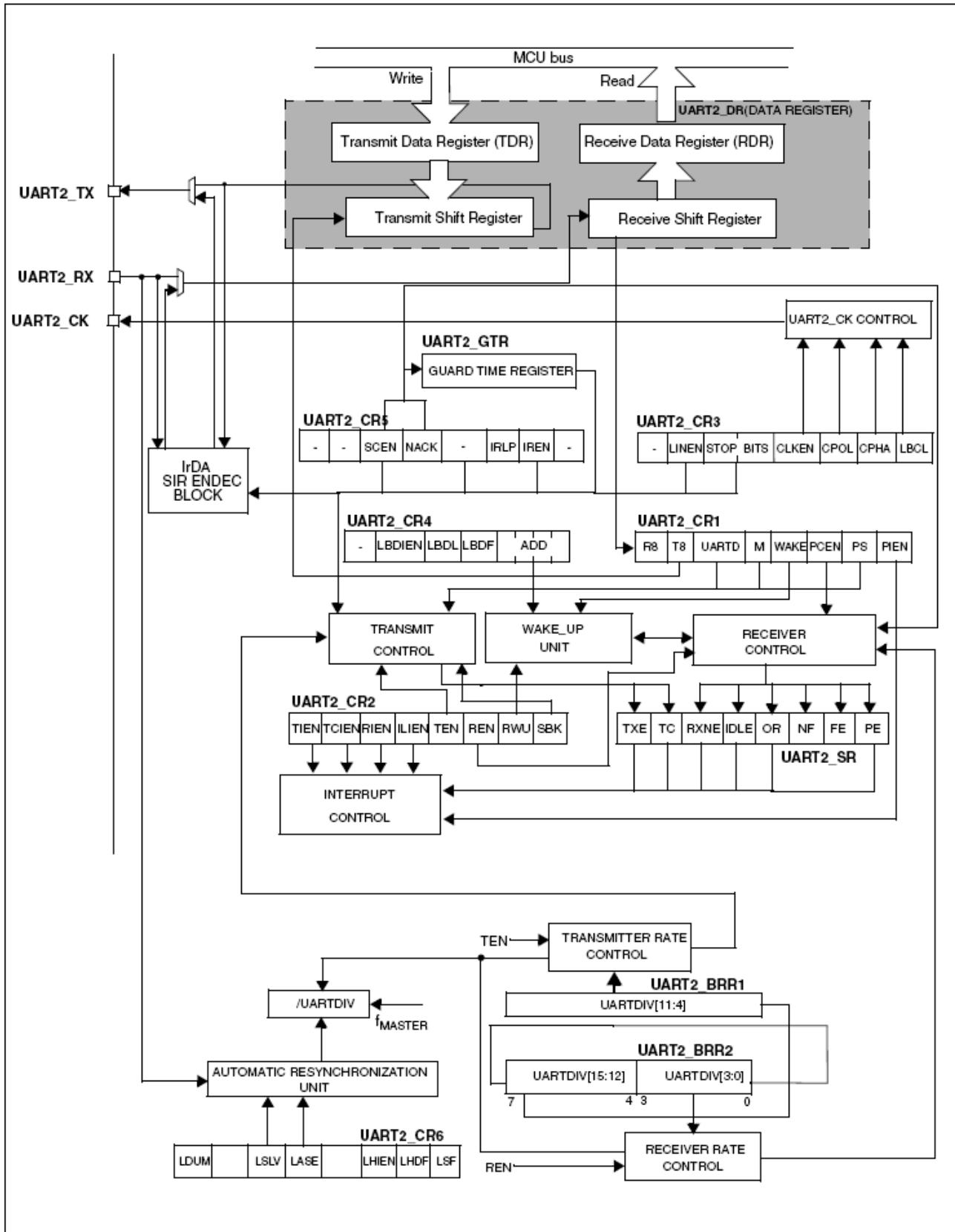
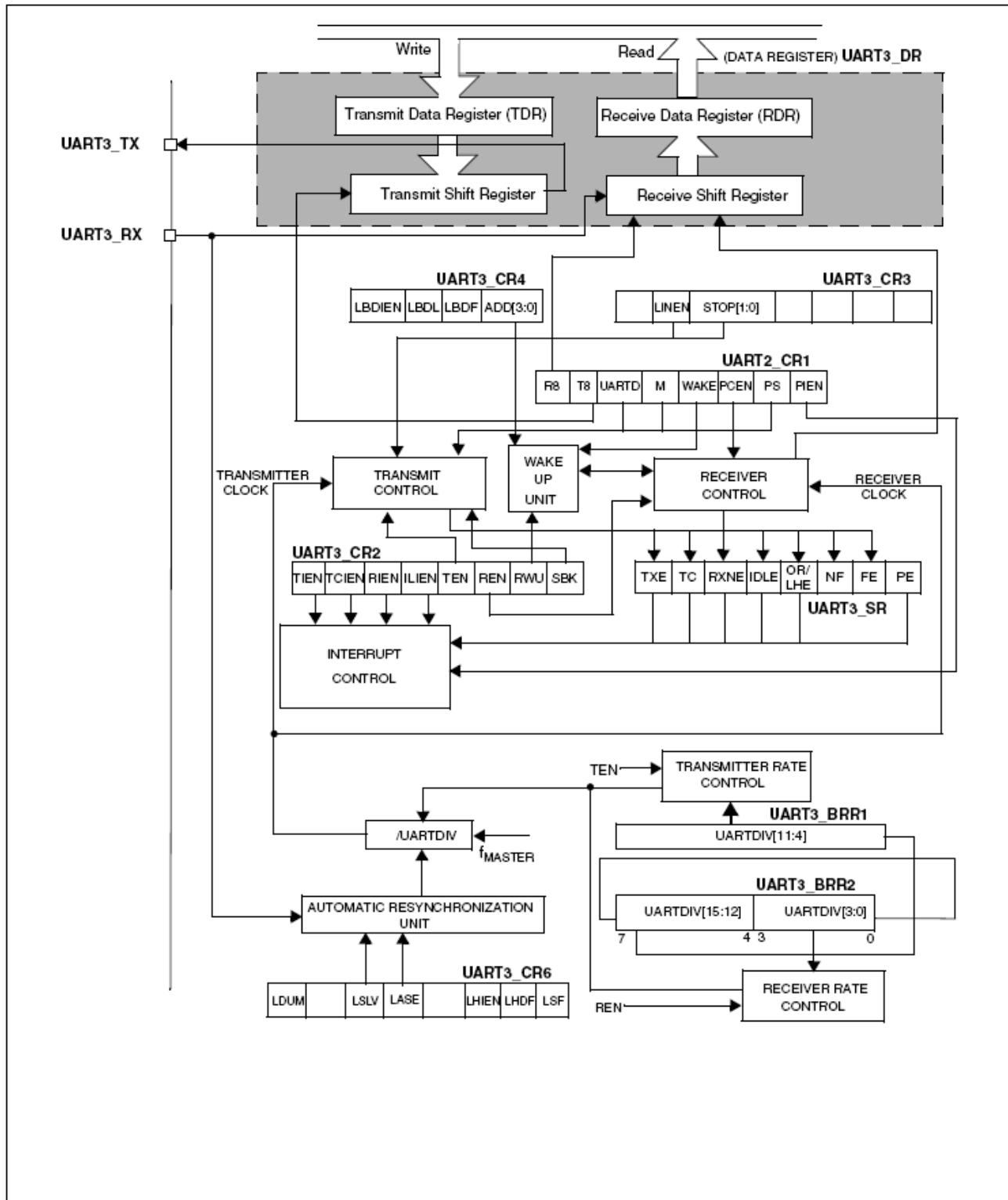


图101 UART3框图



22.3.1 UART 特性描述

字长可以通过编程UART_CR1寄存器中的M位，选择成8或9位(见[图102](#))。

在起始位期间，TX脚处于低电平，在停止位期间处于高电平。

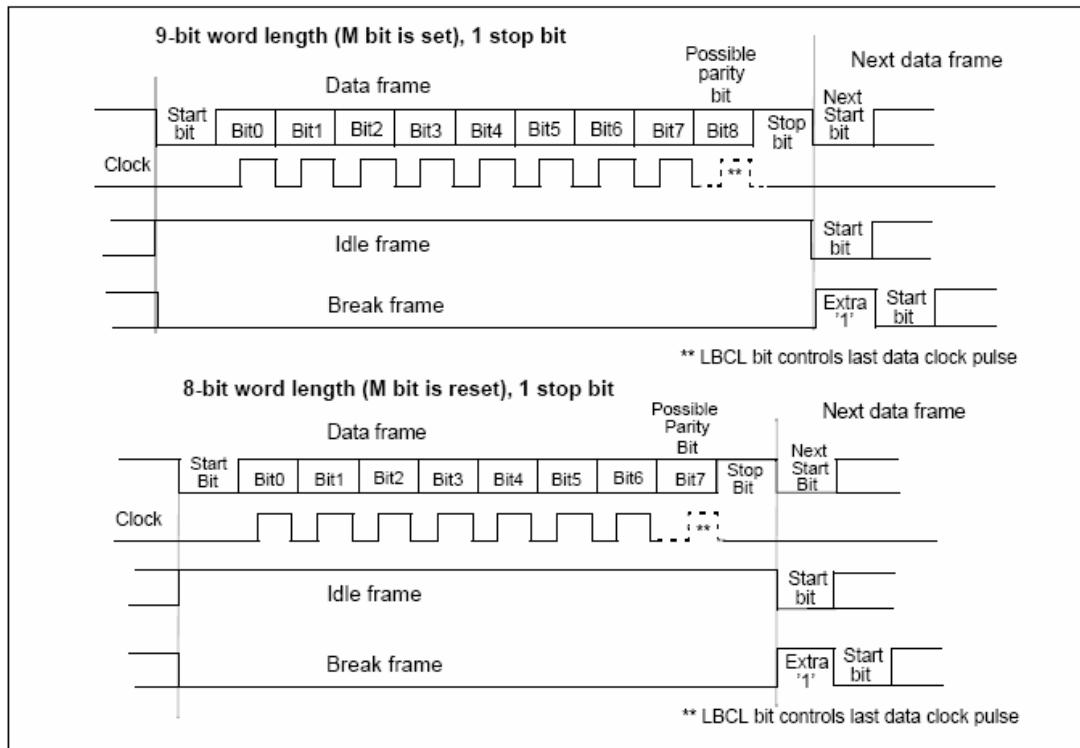
空闲符号被视为完全由'1'组成的一个完整的数据帧 ('1'的位数包括了起始位，数据位和停止位的位数)。

断开符号被视为在一个帧周期内全部收到'0'(包括停止位期间，也是'0')。在断开帧结束时，发送器再插入1或2个停止位('1')来应答起始位。

发送和接收由共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，波特率发生器为其提供相应的时钟。

随后将有每个功能块的详细说明。

图102 字长设置



22.3.2 发送器

发送器根据M位的状态发送8位或9位的数据字。当M位置1，字长为9位，并且第九位(MSB)应该写入寄存器UART_CR1的T8位。

当发送使能位(TE)被设置时，发送移位寄存器中的数据在TX脚上输出，相应的时钟脉冲在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_DR寄存器有一个缓冲器(TDR)，位于内部总线和发送移位寄存器之间(见[图99](#))。

每个字符之前都有一个低电平的起始位；之后跟着数目可配置的停止位。

UART支持以下停止位。

注意: 1. 在数据传输期间不能复位TE位，否则将破坏TX脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

2. *TE*位被激活后将发送一个空闲帧。

可配置的停止位

随每个字符发送的停止位的位数可以通过控制寄存器3的位5、4进行编程。

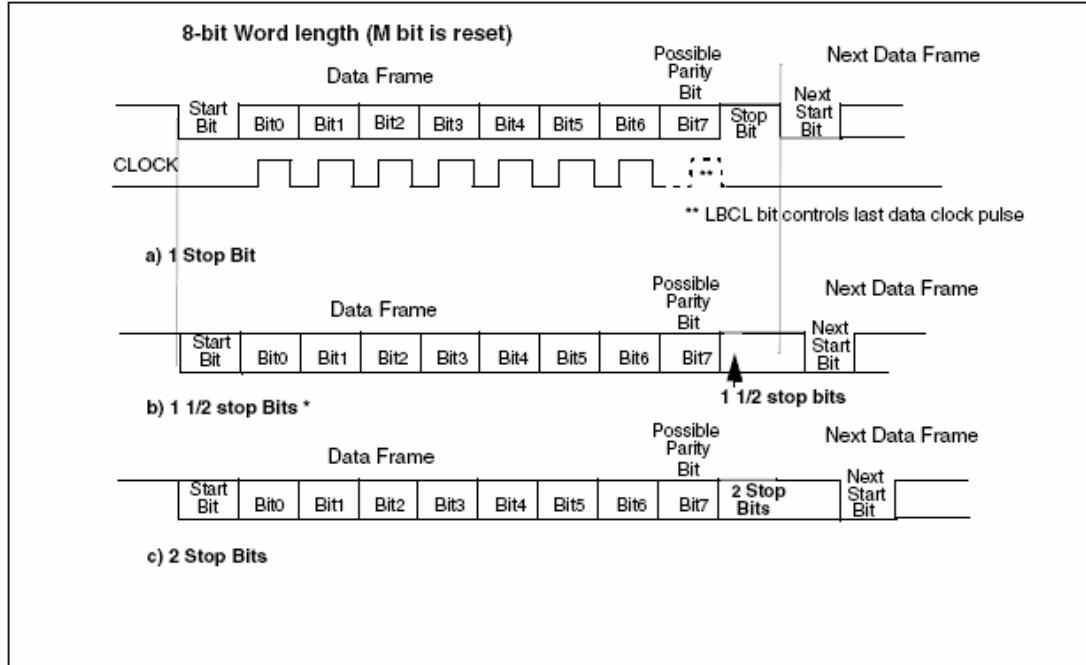
- 1个停止位: 停止位位数的默认值。
- 2个停止位: 可用于常规UART模式。
- 1.5个停止位: 仅在智能卡模式下使用。

空闲帧包括了停止位。

断开帧是10位低电平, 后跟停止位(当m=0时); 或者11位低电平, 后跟停止位(m=1时)。不可能传输更长的断开帧(长度大于10或者11位)。

注意: 在LIN模式(见22.3.7)下, 总是生成标准的13位断开符。

图103 配置停止位



配置步骤:

1. 编程UART_CR1的M位来定义字长。
2. 在UART_CR3中编程停止位的位数。
3. 按下列顺序编写波特率寄存器选择要求的波特率。

- a) UART_BRR2
- b) UART_BRR1

4. 设置UART_CR2中的TE位来使能发送模式。

5. 把要发送的数据写进UART_DR寄存器(此动作清除TXE位)。在只有一个缓冲器的情况下, 对每个待发送的数据重复此步骤。

单字节通信

清零TXE位总是通过对数据寄存器的写操作来完成的。

TXE位由硬件来置1, 它表明:

- 数据已经从TDR移到移位寄存器, 数据发送已经开始
- TDR寄存器为空
- 下一个数据可以被写进UART_DR寄存器而不会覆盖先前的数据

如果TIEN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_DR寄存器的写操作把数据存进TDR寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时UART没有在发送数据，处于空闲状态，对UART_DR寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE位立即被置起。

当一帧发送完成时(停止位发送后)，TC位被置起，并且如果UART_CR1寄存器中的TCIE位被置起时，中断产生。按以下步骤对TC位的清零。

1. 读UART_SR寄存器
2. 写UART_DR寄存器

断开符号

设置SBK可发送一个断开符号。断开帧长度取决M位(见图102)。

如果设置SBK=1，在完成当前数据发送后，将在UART_TX线上发送一个断开符号。断开字符发送完成时(在断开符号的停止位时)SBK被硬件复位。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。

注意：发送的断开帧是不计入停止位位数的。如果设置UART为2个停止位，Tx线路会被拉低直到第一个停止位发送结束。之后在第二个字符前插入2个逻辑1。

注意：如果在开始发送断开帧之前，软件又复位了SBK位，断开字符将不被发送。如果要发送两个连续的断开帧，SBK位应该在前一个断开字符的停止位之后置1。

空闲符号

TEN置位将使得UART在第一个数据帧前发送一空闲帧。

22.3.3 接收器

UART可以接收8位或9位的数据字。如果M位置1，字长为9位，其中MSB存放在寄存器UART_CR1的R8位。

字符接收

在UART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_DR寄存器有一个缓冲器(TDR)，位于内部总线和接收移位寄存器之间。

配置步骤：

1. 编程UART_CR1的M位定义字长
 2. 在UART_CR3中编写停止位的个数
 3. 按下列顺序编写波特率寄存器选择要求的波特率。
 - a) UART_BRR2
 - b) UART_BRR1
 4. 将UART_CR2的REN位置1。这将激活接收器，使它开始寻找起始位。
- 当一字符被接收到时，
- RXNE位被置位。它表明移位寄存器的内容被转移到RDR。
 - 如果RIEN位被设置，则产生中断。
 - 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起，
 - 由软件读UART_DR寄存器完成对RXNE位清除。RXNE标志也可以通过对它写0来清除。RXNE位必须在下一字符接收结束前被清零，以避免溢出错误。

注意：在接收数据时，RE位不应该被复位。如果RE位在接收时被清零，当前接收的字节会丢失。

断开符号

当接收到一个断开帧时，UART像处理帧错误一样处理它。

空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果RIEN位被置1将产生一个中断。

过载错误

如果RXNE还没有被复位，又接收到一个字符，则发生溢出错误。数据只有当RXNE位被清零后才能从移位寄存器转移到RDR寄存器。

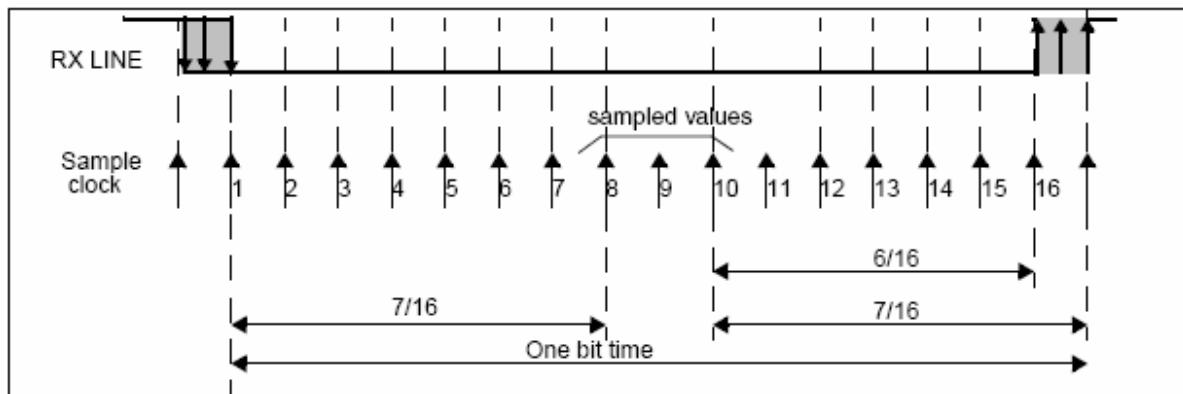
当溢出错误产生时：

- OR位被置位。
- RDR内容将不会丢失。读UART_DR寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果RIEN位被置1，则产生中断。
- 顺序执行对UART_SR和UART_DR寄存器的读操作，可复位OR位

噪音错误

使用过采样技术(同步模式除外)，通过区别有效输入数据和噪音来进行数据恢复。

图104 检测噪声的数据采样



注意：采样频率是波特率的16倍。

表48 检测噪声的数据采样

采样值	NE状态	接收的位值	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- NF在RXNE位的上升沿被置1。
- 无效数据从移位寄存器移动到UART_DR寄存器。

NF这个位和RXNE位同时置1，后者会引发中断。顺序执行对UART_SR和UART_DR寄存器的读操作，可复位NF位。

帧错误

当以下情况发生时检测到帧错误：

由于没有同步上或大量噪音的原因，停止位没有在预期的时间上接收和识别出来。

当帧错误被检测到时：

- FE位被硬件置1
- 无效数据从移位寄存器传送到UART_DR寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和RXNE位同时置1，后者将引发中断。

顺序执行对UART_SR和UART_DR寄存器的读操作，可复位FE位。

接收期间的可配置的停止位

被接收的停止位的个数可以通过控制寄存器3的控制位来配置。在正常模式时，可以是1或2个，IrDA模式里是1个，在智能卡模式里是1.5个。

1. 1个停止位：对1个停止位的采样在第8，第9和第10采样点上进行。
2. 1.5个停止位(仅智能卡模式)：对1.5个停止位的采样是在第16，第17和第18采样点进行的。接收到NACK信号的智能卡会在采样时拉低数据线，以此表示出现了帧错误。FE在1.5个停止位结束时和RXNE一起被置1。
3. 2个停止位：对2个停止位的采样是在第一停止位的第8，第9和第10个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被置1。第二个停止位不再检查帧错误。在第一个停止位结束时RXNE标志将被置1。

22.3.4 高精度波特率发生器

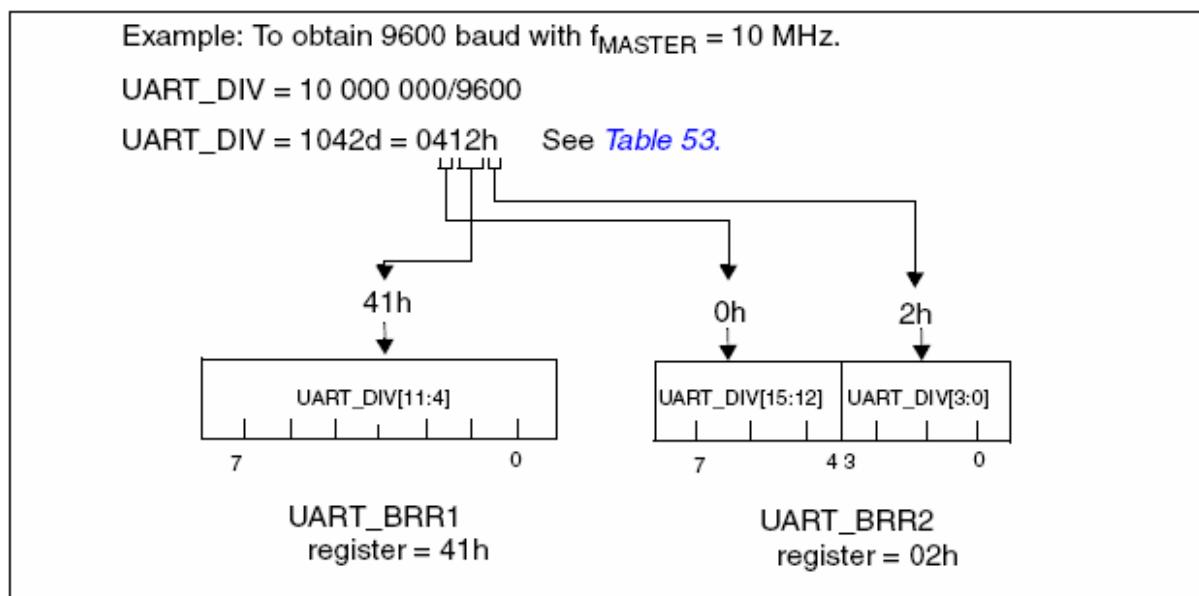
接收器和发送器的波特率可按照下面的公式通过配置16位除法器UART_DIV来设置：

$$\text{Tx/Rx baud rate} = \frac{f_{\text{MASTER}}}{\text{UART_DIV}}$$

UART_DIV是一个无符号的整数。存储在寄存器BRR1和BRR2中。如图105。

请参考[表49](#)来了解典型的波特率设置。

图105 在BRR寄存器里如何编写UART_DIV



注意：波特计数器会在对寄存器BRR1写入新值时更新为新的波特率寄存器值。考虑到波特率寄存器值在传输进行时不该被修改，应当在写寄存器BRR1前，先写寄存器BRR2。

注意：UART_DIV应当大于等于16。

表49 设置波特率时的误差计算

波特率	$f_{MASTER} = 10MHz$					$f_{MASTER} = 24MHz$				
	Kbps	实际	误差% ⁽¹⁾	UART_DIV	BRR1	BRR2	实际	误差% ⁽¹⁾	UART_DIV	BRR1
2.4	2.399	-0.04%	1047h	04h	17h	2.4	0.0%	2710h	71h	20h
9.6	9.596	-0.04%	0412h	41h	02h	9.6	0.0%	09C4h	9Ch	04h
19.2	19.193	-0.03%	0209h	20h	09h	19.2	0.0%	04E2h	4Eh	02h
57.6	57.471	-0.22%	00AEh	0Ah	0Eh	57.554	-0.08%	01A1h	1Ah	01h
115.2	114.942	-0.22%	0057h	05h	07h	115.385	0.16%	00D0h	0Dh	00h
230.4	232.558	-0.94%	002Bh	02h	0Bh	230.769	0.16%	0068h	06h	08h
460.8	454.545	-1.36%	0016h	01h	06h	461.538	0.16%	0034h	03h	04h
921.6	NA	NA	NA			923.077	0.16%	001Ah	01h	0Ah

1. 误差% = (计算波特率 - 期望波特率) / 期望波特率

注意: f_{MASTER} 的频率越低, 对于给定波特率所得到的精确度越低。能获得的波特率上限可以由这个数据给定。

22.3.5 奇偶校验控制

奇偶校验控制(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_CR1寄存器上的PCEN位而激活。根据M位定义的帧长度, 在表50中给出可能的UART帧格式。

表50 帧格式

M位	PCE位	UART帧
0	0	起始位 8位数据 停止位
0	1	起始位 7位数据 奇偶检验位 停止位
1	0	起始位 9位数据 停止位
1	1	起始位 8位数据 奇偶检验位 停止位

注意: 在用地址标记唤醒设备时, 地址的匹配只考虑到数据的MSB位, 而不用关心校验位。

偶校验: 校验位计算一帧中7或8个LSB数据(根据M位是0或1)中'1'的个数是否为偶数, 偶数个则校验位为0, 反之则为1。

例如: 数据=00110101, 有4个'1', 如果选择偶校验(在UART_CR1中的PS=0), 校验位将是'0'。

奇校验: 校验位计算一帧中7或8个LSB数据(根据M位是0或1)中'1'的个数是否为奇数, 奇数个则校验位为0, 反之则为1。

例如: 数据=00110101, 有4个'1', 如果选择奇校验(在UART_CR1中的PS=1), 校验位将是'1'。

传输模式: 如果UART_CR1的PCE位被置位, 写进数据寄存器的数据的MSB位被校验位替换后发送出去(根据PS值来选择偶校验还是奇校验)。

接收模式: 如果奇偶校验失败, UART_SR寄存器中的PE标志被置'1', 并且如果UART_CR1寄存器的PIEN在被预先置1的话, 会产生中断。

22.3.6 多处理器通信

通过UART可以实现多处理器通信(将几个UART连在一个网络里)。例如某个UART设备可以是主设备, 它的TX输出和其他UART从设备的RX输入相连接; UART从设备的各自TX输出作逻辑与运算后和主设备的RX输入相连接。

在多处理器配置中, 我们通常希望只有被寻址的接收者才被激活, 来接收随后的数据, 这样就可以减少由未被寻址的接收器的参与带来的多余的UART服务开销。

未被寻址的设备可启用其静默功能置于静默模式。在静默模式里:

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- **UART_CR1**寄存器中的**RWU**位被置1。**RWU**可以被硬件自动控制或在某个特定条件下由软件写入。

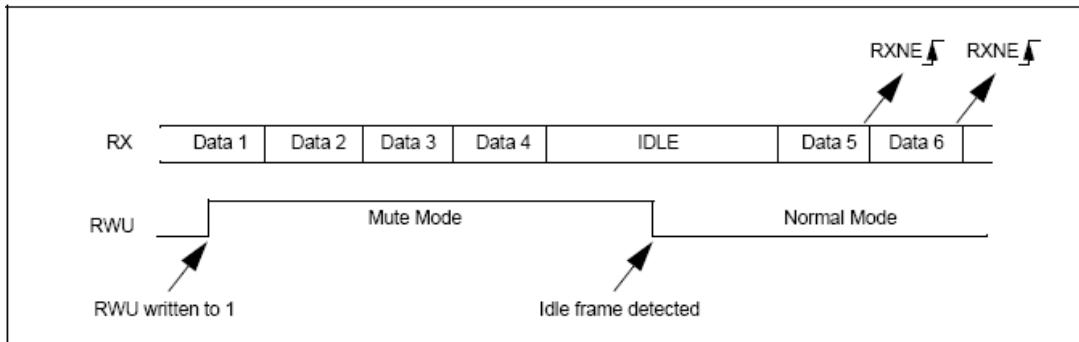
根据**UART_CR1**寄存器中的**WAKE**位状态, **UART**可以用二种方法进入或退出静默模式。

- 如果**WAKE**位被复位, 进行空闲总线检测。
- 如果**WAKE**位被置位, 进行地址标记检测。

空闲总线检测(**WAKE=0**)

当**RWU**位被写1时, **UART**进入静默模式。当检测到一空闲帧时, 它被唤醒。然后**RWU**位被硬件清零, 但是**UART_SR**寄存器中的**IDLE**位并不置1。**RWU**还可以通过软件清0。图106给出利用空闲总线检测来唤醒和进入静默模式的一个例子。

图106 利用空闲总线检测的静默模式



地址标记(address mark)检测(**WAKE=1**)

在这个模式里, 如果**MSB**是1, 该字节被认为是地址, 否则被认为是数据。在一个地址字节中, 目标接收器的地址被放在4个**LSB**中。这个4位地址被接收器同它自己地址做比较, 接收器的地址被编程在**UART_CR2**寄存器的**ADD**位域中。

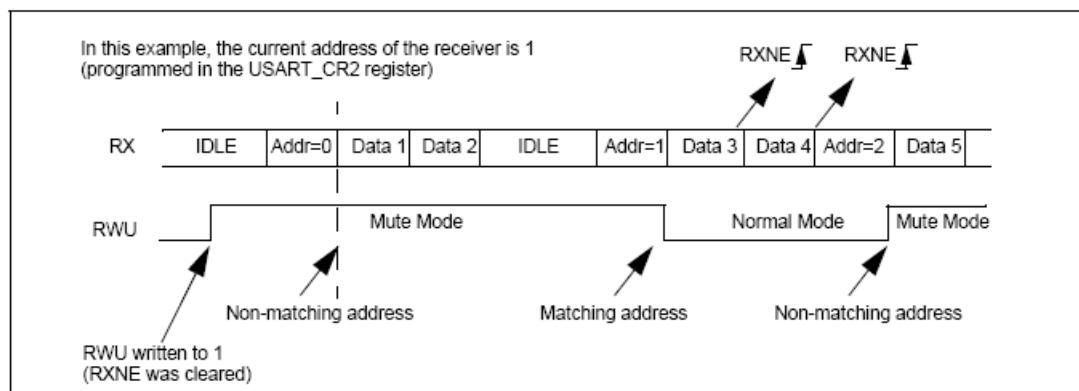
如果接收到的字节与它的编程地址不匹配时, **UART**进入静默模式。该字节的接收既不会置1, **RXNE**标志也不会产生中断, 因为**UART**已经在静默模式。

当接收到的字节与接收器内编程地址匹配时, **UART**退出静默模式。然后**RWU**位被清零, 随后的字节被正常接收。匹配的地址字节将把**RXNE**位置1, 因为**RWU**位已被清零。

当接收缓冲器不包含数据时(**UART_SR**的**RXNE**=0), **RWU**位可以被写0或1。否则, 该次写操作被忽略。

图107给出利用地址标记检测来唤醒和进入静默模式的例子。

图107 利用地址标记检测的静默模式



注意: 如果使能校验控制, 校验位在**MSB**, 而地址位在“**MSB-1**”位。

例如，对于7位数据，地址模式和校验控制位如下所示：

起始位 | 7位数据 | 地址位 | 校验位 | 停止位

22.3.7 LIN(局域互联网)模式

UART在LIN主模式下支持LIN断开和分隔符生成。

参见[22.4.1](#)获取更多细节。UART2, UART3支持LIN从模式，UART1不支持。

LIN模式是通过设置UART_CR3寄存器的LINEN位来实现。在LIN模式下，下列位必须保持为0：

- UART_CR3的CLKEN位，STOP[1:0]位
- UART_CR5的SCEN, HDSEL和IREN

22.3.8 UART 同步模式

UART发送器允许用户在主模式下控制双向同步串行通讯。

在同步模式里，下列位必须保持清零状态：

- UART_CR3寄存器中的LINEN位
- UART_CR5寄存器中的SCEN,HDSEL和IREN位

注意：该功能仅UART1和UART2可用。

UART_CK脚是UART发送器时钟的输出。在起始位和停止位期间，UART_CK脚上没有时钟脉冲。根据UART_CR3寄存器中LBCL位的状态，发送器决定在最后一个有效数据位期间产生或不产生时钟脉冲。UART_CR3寄存器的CPOL位允许用户选择时钟极性，UART_CR3寄存器上的CPHA位允许用户选择外部时钟的相位(见[图108](#)、[图109](#)和[图110](#))。

在总线空闲帧和断开帧中，外部CK时钟处于非激活状态。

同步模式的UART接收器工作方式与异步模式不同。如果RE=1，数据在SCLK边沿采样（根据CPOL和CPHA决定在上升沿还是下降沿），不需要任何的过采样。但建立时间和保持时间(即使SPI协议没有和保持时间相关的内容)产生的影响必须被考虑进来(取决于波特率，一个整数波特率1/16位时间)。

- 注意：
1. **UART_CK脚同UART_TX脚一起联合工作。**当UART发送端被禁用时(TEN和REN=0)，UART_CK和UART_TX管脚为高阻态。
 2. 在UART发送端和接收端都被禁用(TEN=REN=0)时，LBCL,CPOL和CPHA位必须被正确配置以保证时钟脉冲正确工作；当发送器或接收器被激活时，这些位不能被改变。
 3. 建议在同一条指令中设置TE和RE，以减少接收器的建立时间和保持时间。
 4. **UART只支持主模式：**它不能使用来自其他设备的输入时钟以接收或发送数据(SCLK只能配置为输出状态)。
 5. 本节给出的数据只有在寄存器UART_BRR2的UART_DIV[3:0]为0时才有效。否则建立时间和持续时间不再是1/16位时间，而是4/16位时间。

该功能选项可以串行地控制那些由移位寄存器组成的外设，而不会失去异步通讯的功能，即仍然可以与其他异步发送接收器通讯。

图108 UART同步传输的例子

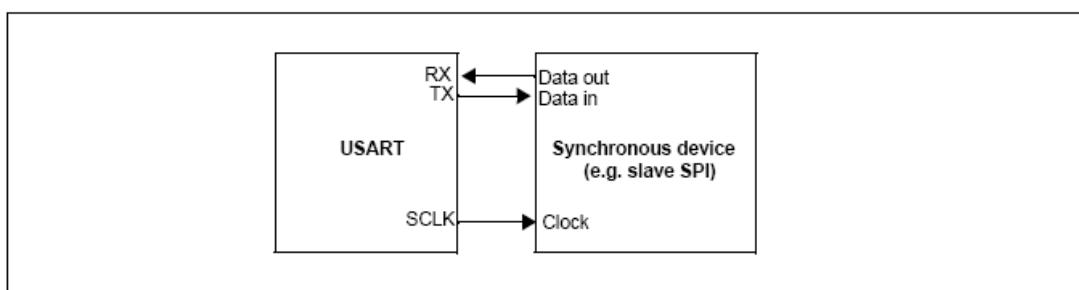


图109 UART数据时钟时序示例(M=0)

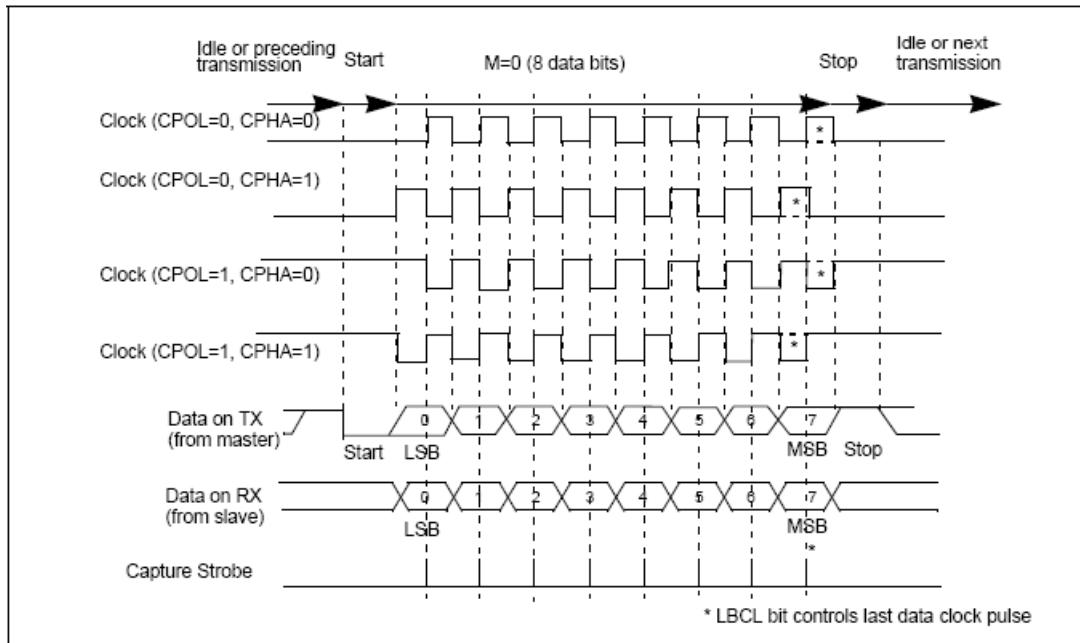


图110 UART数据时钟时序示例(M=1)

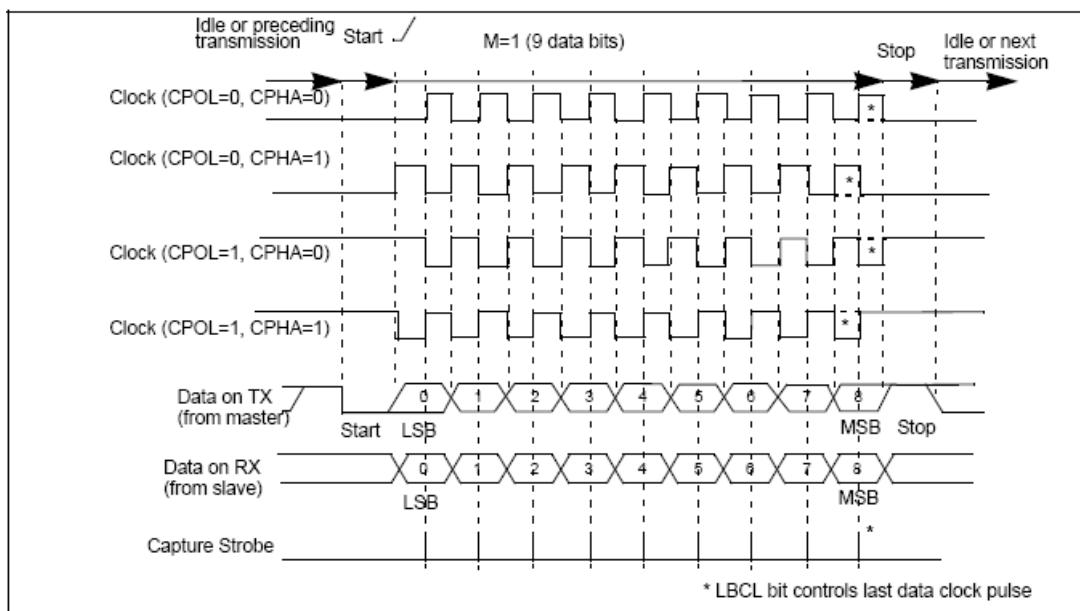
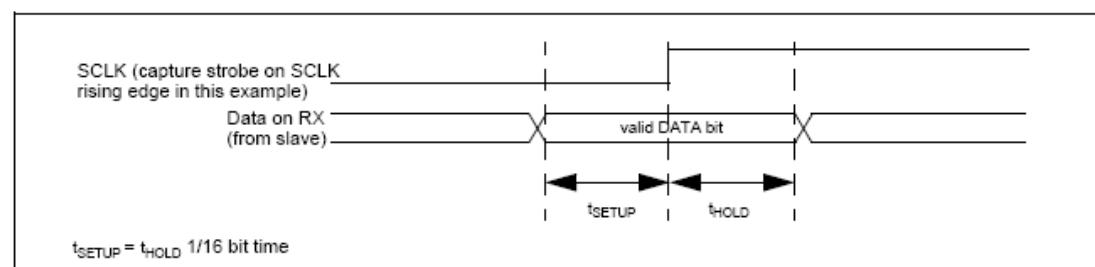


图111 RX数据采样/保持时间



注：在智能卡模式下SCLK的功能不同，有关细节请参考智能卡模式部分。

22.3.9 单线半双工通信

UART可以配置成遵循单线半双工协议。单线半双工模式通过设置UART_CR5寄存器的HDSEL位实现。在该模式下，下面的位必须保持清零状态：

- UART_CR3寄存器的LINEN和CLKEN位
- UART_CR5寄存器的SCEN和IREN位

注意：该功能只适用于UART1。

当HDSEL写'1'时

- UART_RX不再被使用
- 当没有数据传输时，UART_TX处于释放状态。因此，它在空闲状态的或接收状态时表现为一个标准I/O口。这就意味该I/O在不被UART驱动时，必须配置成悬空输入(或开漏的输出高)。

除此以外，通信与正常UART模式类似。要注意的是必须由软件来管理线上的冲突(例如通过使用一个中央仲裁器)。特别的是，发送从不会被硬件所阻碍。当TE位被设置时，只要数据一写到数据寄存器上，发送就继续。

22.3.10 智能卡

设置UART_CR5寄存器的SCEN位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

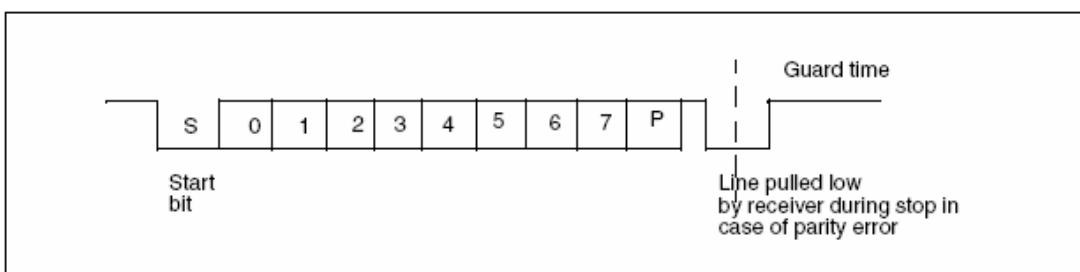
- UART_CR3寄存器的LINEN位
- UART_CR5寄存器的HDSEL位和IREN位

此外，CLKEN位可以被设置，以提供时钟给智能卡。

注意：该功能只适用于UART1和UART2。

智能卡接口设计成支持ISO7816-3标准所定义的异步协议智能卡。UART应该被设置为8位数据位加校验位和1.5位停止位。当智能卡模式使能时(寄存器UART_CR5的SCEN位置1)，UART可以与异步智能卡通讯。

图112 ISO7816-3异步协议



当与智能卡相连接时，UART的TX和智能卡共同驱动一根双向通讯线。

智能卡是一个单线半双工通信协议

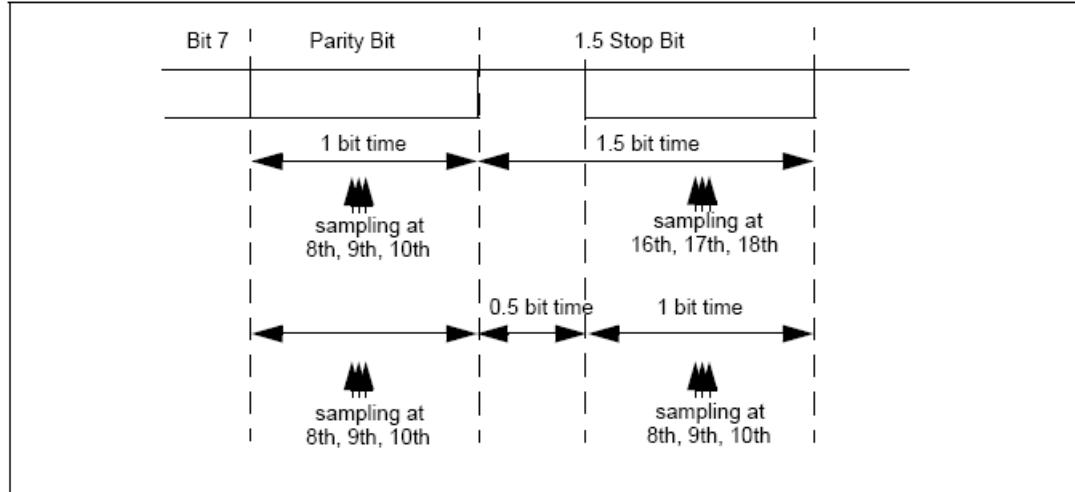
- 从发送移位寄存器把数据发送出去，要被延时最小1/2波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被再延迟1/2波特时钟。
- 如果在接收一个设置为1.5停止位的数据帧期间，检测到一个校验错误，在1/2波特时钟周期后，发送线被拉低一个波特时钟周期。这是告诉智能卡发送到UART的数据没有被正确接收到。此NACK信号(拉低发送线一个波特时钟周期)在发送端会识别为一个帧错误(发送端被配置成1.5个停止位)。应用程序可以根据协议处理重新发送的数据。如果NACK控制位被置1，发生校验错误时接收器就会给出一个NACK信号；否则就不会发送NACK。
- TE位必须通过置1来使能
 - 数据发送

- 在校验错误发生时，发送应答
当向数据寄存器写入新数据时，必须由软件来管理数据发送时间以避免数据线上的冲突。
- RE位必须置1来使能
 - 数据接收(由智能卡或者 **UART** 发送)
 - 侦测在校验错误发生时发送的应答
- 对保护时间寄存器编程会使得TC标志置1的时间点推后。在正常操作时，当发送移位寄存器变空并且没有新的发送请求出现时，TC被置1。在智能卡模式里，空的发送移位寄存器将触发保护时间计数器开始向上计数，直到计到等于保护时间寄存器中的值。TC在这段时间被强制拉低。当保护时间计数器达到保护时间寄存器中的值时，TC被置高。
- TC标志的撤销不受智能卡模式的影响。
- 如果发送器检测到一个帧错误(收到接收器的NACK信号)，发送器的接收功能模块不会把NACK当作起始位检测。根据ISO协议，接收到的NACK的持续时间可以是1或2波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且NACK被发送，接收器不会把NACK检测成起始位。
- 智能卡I/O的输出使能信号可以使能对双向信号线的驱动，这条线路同时也由智能卡驱动。在发送起始位，数据位和NACK信号的时候，该信号有效。在发送停止位的时候，该信号被禁用，这样UART在双向信号线上的状态为弱上拉信号“1”。

注意： 1. 断开符号在智能卡模式里没有意义。一个带帧错误的00h数据将被当成数据而不是断开符号。
2. 当来回切换TE位时，没有IDLE帧被发送。ISO协议没有定义IDLE帧。

图113详述了UART是如何采样NACK信号的。在这个例子里，UART正在发送数据，并且被配置成1.5个停止位。为了检查数据的完整性和NACK信号，UART的接收功能块被激活。

图113 使用1.5停止位检测奇偶检验错



UART可以通过UART_CK输出为智能卡提供时钟。在智能卡模式里，UART_CK不和通信直接关联，而是先通过一个5位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频系数在预分频寄存器UART_PSCR中配置。UART_CK频率可调整的范围是从 $f_{MASTER}/2$ 到 $f_{MASTER}/62$ ，这里的 f_{MASTER} 是外设输入时钟。

22.3.11 IrDA SIR ENDEC 功能块

通过设置UART_CR5寄存器的IREN位选择IrDA模式。UART_CR3寄存器的STOP位必须设置成“1个停止位”。在IRDA模式里，下列位必须保持清零：

- UART_CR3寄存器的LINEN,STOP和CLKEN位
- UART_CR5寄存器的SCEN和HDSEL位。

注意： 该功能只适用于UART1和UART2。

IrDA SIR物理层规定使用反相归零调制方案(RZI)，该方案用一个红外光脉冲代表逻辑'0'(见图114)。

SIR发送编码器对从UART输出的NRZ(非归零)比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外LED。对于SIR ENDEC应用，UART最高只支持到115.2Kbps速率。在正常模式里，脉冲宽度规定为一个位周期的3/16。

SIR接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的NRZ串行比特流输出到UART。在空闲状态里，解码器输入通常是高(标记状态marking state)。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA是一个半双工通信协议。如果发送器忙(也就是UART正在送数据给IrDA编码器)，IrDA接收线上的任何数据都将被IrDA解码器所忽略。如果接收器忙(也就是UART正在接收从IrDA解码器来的解码数据)，从UART的TX上到IrDA的数据将不会被IrDA编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。
- SIR发送逻辑把'0'作为高脉冲发送，把'1'作为低电平发送。脉冲的宽度规定为正常模式时位周期的3/16(见图115)。
- SIR解码器把接收到的IrDA信号转变成比特流后发送给UART。
- SIR接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR输出处于低状态。
- IrDA规范要求脉冲要宽于1.41us。脉冲宽度是可编程的。接收器端的尖峰脉冲检测电路会通过对宽度小于2个PSC周期的脉冲进行过滤操作(PSC是在UART_GTPR中编程的预分频值)。宽度小于1个PSC周期的脉冲一定会被过滤掉，但是那些宽度大于1个而小于2个PSC周期的脉冲可能被接收或滤除，那些宽度大于2个周期的将被视为一个有效的脉冲。当PSC=0时，IrDA编码器/解码器不工作。
- 接收器可以与一低功耗发送器通信。
- 在IrDA模式里，UART_CR2寄存器上的STOP位必须配置成1个停止位。

IrDA低功耗模式

IrDA可以工作在正常模式，也可以工作在低功耗模式。选择低功耗模式需要把UART_CR5寄存器的IRLP位置1。

发送器

在低功耗模式，脉冲宽度不再持续3/16个位周期。取而代之，脉冲的宽度是低功耗波特率时钟周期的3倍，该波特率的频率最小可以是1.42MHz。通常这个值是1.8432MHz(1.42 MHz < PSC < 2.12 MHz)。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

接收器

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，UART应该滤除宽度短于1个周期的脉冲。只有持续时间大于2个周期的IrDA低功耗波特率时钟(UART_GTPR中的PSC)的低电平信号才被接受为有效的信号。

- 注意：
1. 宽度小于2个大于1个PSC周期的脉冲可能会也可能不会被滤除。
 2. 接收器的建立时间应该由软件管理。IrDA物理层技术规范规定了在发送和接收之间最小要有10ms的延时(IrDA是一个半双工协议)。

图114 IrDA SIR ENDEC – 框图

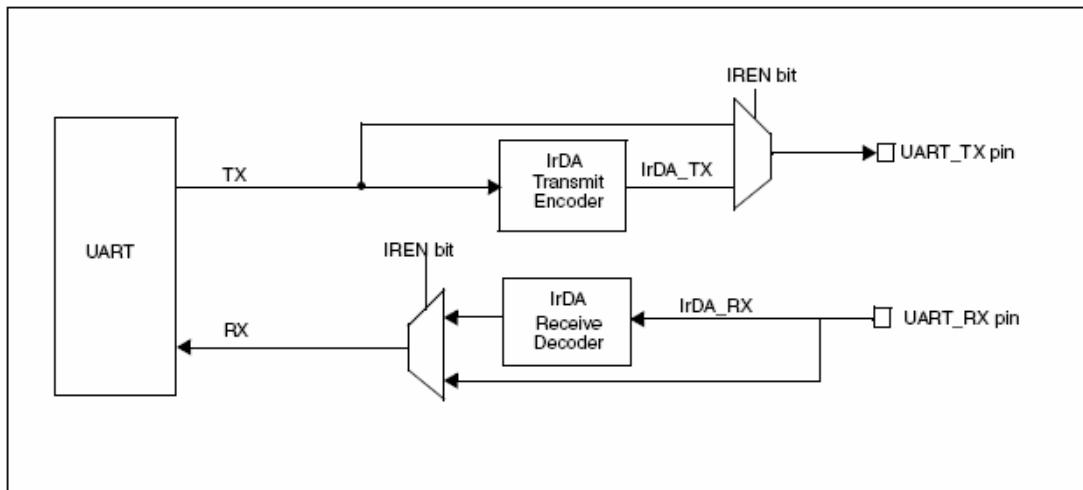
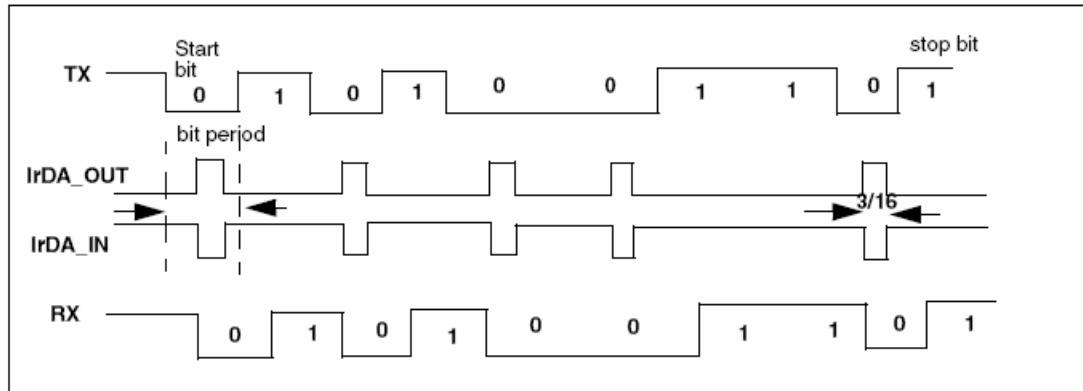


图115 IrDA数据调制(3/16) – 普通模式



22.4 LIN模式功能描述

在LIN模式下，LIN标准要求的数据格式是8位数据位加1位停止位。

完成这样的设置需要将UART_CR1的M位清0，同时把UART_CR3的STOP[1:0]位也清0。

22.4.1 主模式

UART初始化

步骤：

1. 设置UART_BRR2和UART_BRR1来选择期望的波特率。
2. 将UART_CR3的LINEN位置1来使能LIN模式。
3. 将UART_CR2的TEN位和REN位置1来使能发送器和接收器。

LIN报文头(header)的发送

按照LIN协议，所有在LIN总线上的通讯都由主设备通过发送报文头(header)发起，报文头之后是响应。报文头由主任务(主节点)发送，而数据由节点(主节点或者从节点)的从任务发送。

不带错误侦测的步骤：

1. 将UART_CR2的SBK位置1来请求发送断开符+分界符。
2. 对UART_DR写入0x55来请求发送同步域。
3. 等待UART_SR的标志位TC为1。
4. 对UART_DR写入被保护识别符值来请求发送识别符域。

5. 等待UART_SR的标志位TC为1。

带错误侦测的步骤：

1. 将UART_CR2的SBK位置1来请求发送断开符+分界符。
2. 等待UART_CR4的标志位LBDF为1。
3. 对UART_DR写入0x55来请求发送同步域。
4. 等待UART_SR的标志位RXNE为1，并读回UART_DR。
5. 对UART_DR写入被保护识别符值来请求发送识别符域。
6. 等待UART_SR的标志位RXNE为1，并读回UART_DR。

只有在UART_RX管脚上接收回有效的断开符和分界符以后，标志位LBDF才会置1。

LIN断开符和分界符侦测

当LIN模式被使能时，断开符号检测电路被激活。该检测完全独立于USART接收器。断开符只要一出现就能检测到，不管是在总线空闲时还是在发送某数据帧期间。

当接收器被激活时(USART_CR1的REN=1)，电路监测RX上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来位的第8, 9, 10个过采样时钟点上进行采样。如果10个(当USART_CR4的LBDL = 0)或11个(当USART_CR4的LBDL = 1)连续位都是'0'，并且又跟着一个分界符，USART_SR的LBD标志被置1。如果LBDIEN位=1，会有中断产生。

如果在第10或11个采样点之前采样到了'1'，检测电路取消当前检测并重新寻找起始位。如果LIN模式被禁止(LINEN=0)，接收器继续如正常USART那样工作，不需要考虑检测断开符。

如果LIN模式被激活(LINEN=1)，只要一发生帧错误(例如在发送断开帧时，停止位检测到'0')，接收器就停止，直到断开符号检测电路接收到一个'1'(这种情况发生于断开符没有完整的发出来)，或一个定界符(这种情况发生于已经检测到一个完整的断开符号)。

[图116](#)说明了断开符检测器状态机的行为和断开符号标志的关系。

LBDF标志用于主模式，LHDF标志用于从模式。

[图117](#)给了一个断开帧的例子。

图116 LIN模式下的断开符检测(11位断开长度 – 设置了LBDL位)

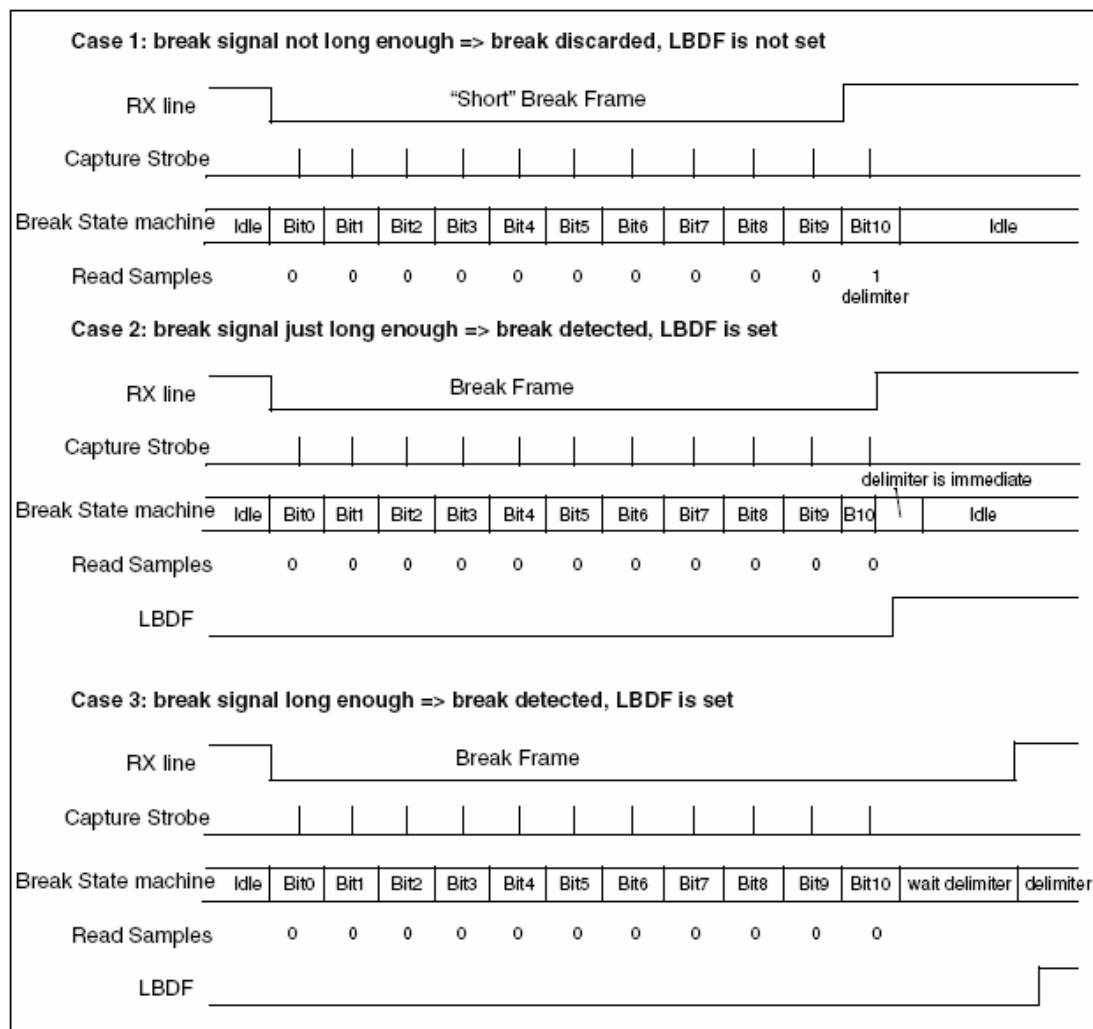
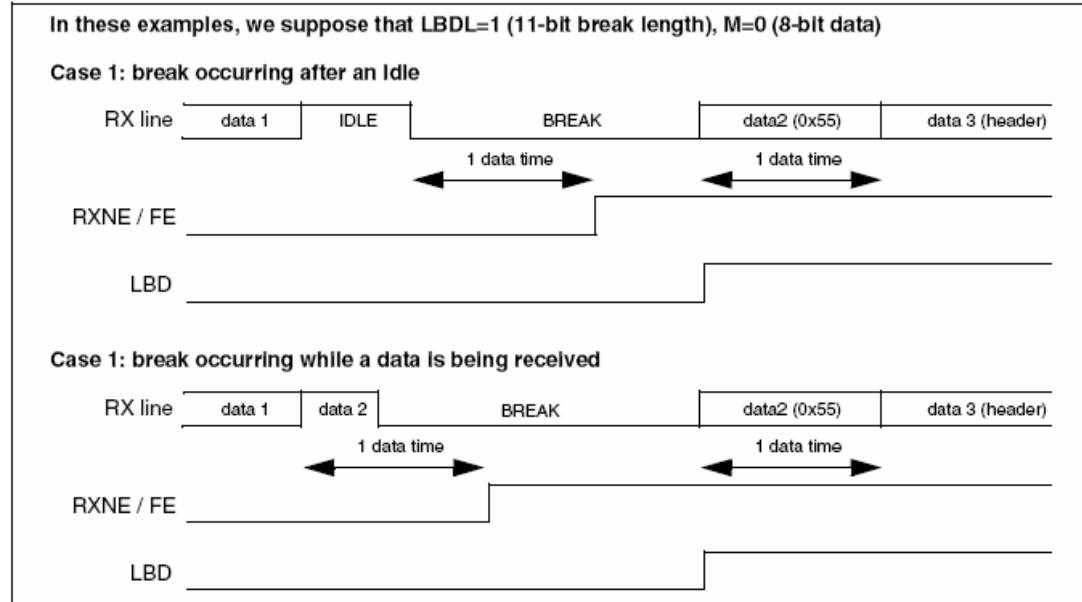


图117 LIN模式下的断开符检测与帧错误的检测



响应发送(主设备是响应的发布方)

响应由符合UART规范的字节组成：8位数据位，1位停止位，无校验位。

要发送n字节的数据，应当按顺序重复n次以下步骤：

1. 对UART_DR寄存器写入数据
2. 等待UART_SR寄存器的标志位RXNE为1
3. 读UART_DR寄存器，检查读回值

响应接收(主设备是响应的签署方)

要接收n字节的数据，应当按顺序重复n次以下步骤：

1. 等待UART_SR寄存器的标志位RXNE为1
2. 读UART_DR寄存器

响应忽略(从设备对从设备通讯)

在从设备与从设备的通讯，并且主设备不需要检查响应中错误的情况下，应用程序可以直到下一帧传输前都不必检查标志位RXNE。在下一次断开符发送之前，应当清除标志位RXNE和OR。

注意：接收回一个断开符也会在设置标志位LBDF前设置标志位RXNE和FE。因此，如果使用RX中断，最好在发送断开符之前关闭中断，来避免一次额外的中断。在从设备对从设备通讯的情况下，一旦报文头被发出，就可以清0标志位RIEN。

22.4.2 自动重同步功能禁用的从模式

注意：该特征只应用于UART2和UART3。

UART初始化

步骤：

1. 设置UART_BRR2和UART_BRR1来选择期望的波特率。
2. 将UART_CR2的TEN位和REN位置1来使能发送器和接收器。
3. 使能UART_CR6的LSLV位
4. 将UART_CR3的LINEN位置1来使能LIN模式。

LIN报文头(header)的接收

按照LIN协议，从节点必须等待由主节点发送的合法报文头(header)。按照不同的报文头识别符值，应用程序应进行以下操作：

- 接收响应
- 发送响应
- 忽略响应并等待下一个报文头

当接收到报文头时：

- UART_CR6寄存器的LHDF标志位提示探测到LIN报文头
- 如果UART_CR6寄存器的LHDIEN位置1则产生中断
- 在UART_DR中的LIN识别符已准备好

注意：建议把RWU位置1来将UART置为静默模式。该模式下只允许探测报文头，不接收其他任何字符。

在LIN作为从设备时，把UART_CR2的PCEN位置1可使能识别符的校验位检测。如果识别符的校验位错误，那么UART_CR6的PE标志位和LHDF标志位置1。

响应发送(从设备是响应的发布方)

要发送n字节的数据，应当按顺序重复n次以下步骤：

1. 对UART_DR写入数据
2. 等待UART_SR的标志位RXNE为1
3. 读UART_DR，检查读回值

一旦完成响应的发送，软件可以把RWU位置1。

响应接收(从设备是响应的签署方)

要接收n字节的数据，应当按顺序重复以下步骤：

1. 等待UART_SR的标志位RXNE为1
2. 读UART_DR

一旦完成响应的发送，软件可以把RWU位置1。

响应忽略

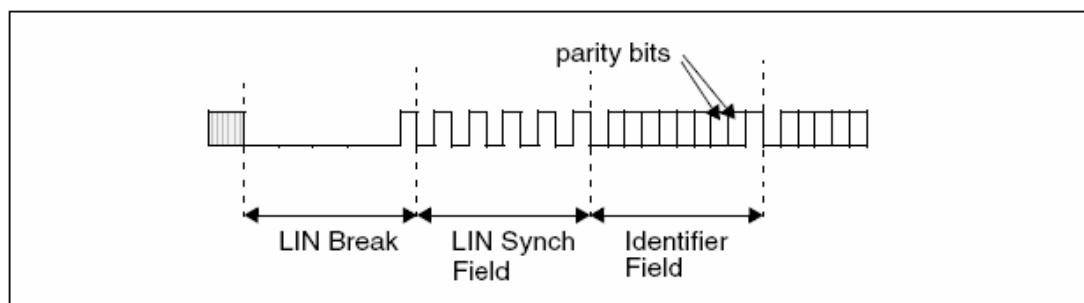
软件可以立刻把RWU位置1。

LIN从模式奇偶校验

在LIN从模式下(LINEN位和LSLV位置1)，可以设置PCEN位使能LIN奇偶校验位检测。如果标识符校验位错误发生(标志位PE置1)并且PIEN位置1，则产生中断。

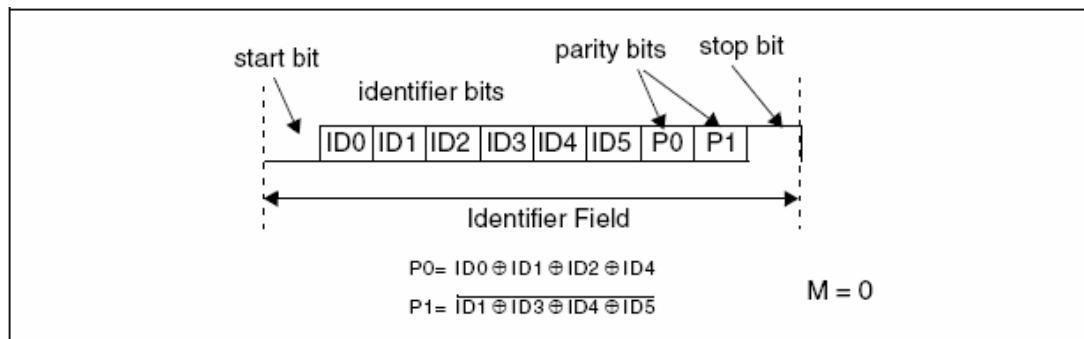
在这种情况下，LIN标识符域的校验位会被检测。作为断开符后(包括断开符)的第三个接收到的字符被认为是标识符字符。

图118 LIN标识符域校验位



这些位包含标识符字符的2个MSB位(第7位和第8位)。检测按照LIN标准指定的方式进行。

图119 LIN标识符域校验位检测



LIN报文头错误检测

LIN报文头错误标志位提示接收到非法的LIN报文头。

当LIN报文头错误发生时：

- 标志位LHE置1

- 如果UART_CR2的RIEN位置1则产生中断

在访问寄存器UART_SR后，读寄存器UART_DR可以对LHE位置0。

以下几种情况之一发生时，LHE位置1

- 断开分界符过短
- 同步域不是55h
- 在同步域或者标识符域有帧错误
- LIN报文头接收超时

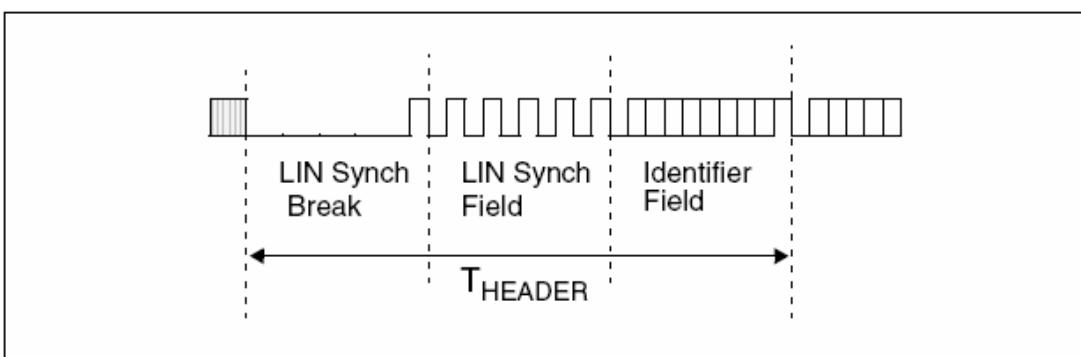
注意：如果发生LIN报文头错误，应用程序一定要把UART_CR6寄存器的LSF位清0。

LIN报文头超时错误

UART自动检测LIN协议给出的T_HEADER_MAX条件。

如果整个报文头(直到并包括标识符域的STOP位)没有在最大时限(57个位时间)内接收完毕，那么将产生LIN报文头错误，UART_SR寄存器的LHE标志位置1。

图120 LIN报文头接收超时



超时计数器在每次断开符检测时开始计数，在以下情况下停止：

- LIN标识符域被接收
- LHE错误发生(不同于超时错误)
- 在LIN同步域分析时发生LSF位的软件复位(从高电平转到低电平)

如果LHE位置1发生于LIN同步域(如果LASE位=1)，那么UART进入阻塞模式(LSF位置1)。

如果LHE位置1发生于LIN同步域以外，或者LASE位置0，那么会弃当前的报文头，UART搜索新的断开域。

LIN报文头超时注意事项

按照LIN协议，不会引起超时的LIN报文头最大长度等于：

$$1.4 * (34+1) = 49 \text{ T}_{\text{BIT_MASTER}}$$

$\text{T}_{\text{BIT_MASTER}}$ 指主设备波特率

在检测超时时，从节点为了接收LIN断开符和同步域而解除同步。因此，必须允许一定的裕量并考虑最极端情况：此时LIN标识符正好持续10个 $\text{T}_{\text{BIT_MASTER}}$ 周期。这样LIN断开符和同步域持续 $49-10 = 39 \text{ T}_{\text{BIT_MASTER}}$ 周期。

假设从设备用偏移为15.5%的时钟测量这39位，这就导致允许的最大报文头长度为：

$$\begin{aligned} 39 * (1/0.845) \text{ T}_{\text{BIT_MASTER}} + 10 \text{ T}_{\text{BIT_MASTER}} \\ = 56.15 \text{ T}_{\text{BIT_SLAVE}} \end{aligned}$$

加上一点裕量，这样在报文头长度大于 $57 \text{ T}_{\text{BIT_SLAVE}}$ 周期时就会发生超时。如果报文头小于或等于 57 个 $\text{T}_{\text{BIT_SLAVE}}$ 周期，则不发生超时。

静默模式与错误

在静默模式下, 如果LHE错误发生在LIN同步域解析期间, 或者LIN报文头超时, 那么LHE位置1, 但是UART不会被从静默模式中唤醒。在这种情况下, 当前的报文头解析被放弃。如果必要, 软件应当把LSF位置0, 然后UART开始搜索新的LIN报文头。

在静默模式下, 如果帧错误发生在数据(而不是断开符)上, 该数据被丢弃, FE位不置1。

任何符合以下情况的报文头, 能够导致从静默模式中唤醒

- 有效的LIN断开符和分界符
- 有效的LIN同步域(没有偏移错误)
- 不含帧错误的LIN标识符域。注意即使发生LIN标识符域的LIN校验错误, UART仍然会从静默模式中唤醒
- 在报文头接收过程中不发生LIN报文头超时

22.4.3 自动重同步使能的从模式

此模式与22.4.2描述的从模式类似, 只是要额外把LASE位置1使能的自动重同步功能。在此模式下, UART在每次接收到LIN同步域后, 调整波特率发生器。

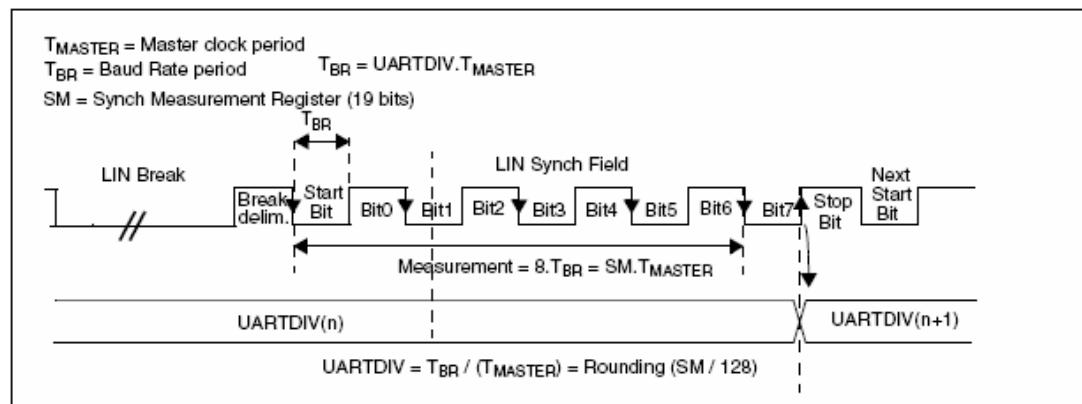
注意: 该特征只用于UART2和UART3。

自动重同步

当使能自动重同步时, 每次LIN断开符之后, 以 f_{MASTER} 为基频采样RDI上5个下降沿之间的时间, 结果存放在内部名为SM的19位寄存器(用户不能访问)内。(见图121)。

然后, 在第五个下降沿以后自动更新UARTDIV值(及与之相关的寄存器UART_BRR1和UART_BRR2)的值。在LIN同步域测量期间, UART状态机停止工作, 数据不会转移到数据寄存器。

图121 LIN同步域测量



UARTDIV是不带符号位的整数, 存放在寄存器BRR1和BRR2中, 如图105。

如果LASE位=1, 那么UARTDIV在每个LIN同步域之后自动更新。

有三个内部寄存器被用于管理LIN分频数(UARTDIV):

- UARTDIV_NOM(由软件写入UART_BRR1和UART_BRR2地址的名义值)
- UARTDIV_MEAS(同步域测量的结果)
- UARTDIV(用以生成本地波特率)

图122和图123解释了3个寄存器之间的控制和关联情况, 它们按照LDUM(LIN分频数更新方式)位的设置情况工作。

如图122和图123所示, 2种并发操作可对UARTDIV更新: 在LIN同步域结尾从UARTDIV_MEAS寄存器传出, 或者在使用软件对BRR1寄存器进行写操作后从UART_NOM寄存器传出。如果2个操作同时发生, 从UART_NOM传出的数据有更高的优先级。

图122 LDUM=0 时UARTDIV的读/写操作

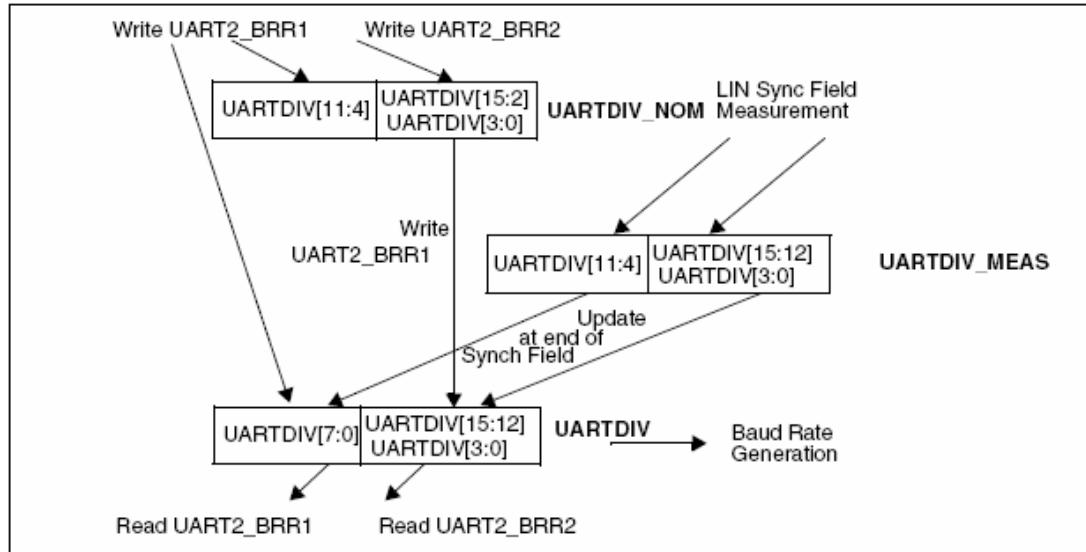
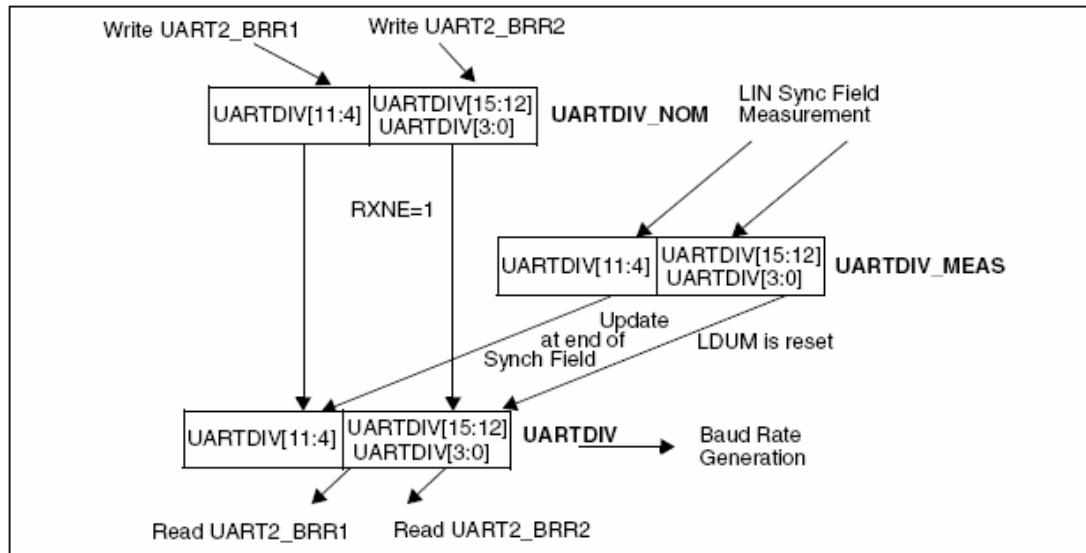


图123 LDUM=1 时UARTDIV的读/写操作



同步域的偏移错误

通过比较当前波特率(与从设备振荡器相关)和接收到的LIN同步域(与主设备振荡器相关)来侦测偏移错误。并行地进行2次检测。

第一次检测基于LIN同步域第一个下降沿和第二个下降沿之间的测量。

- 如果 $D1 > 14.84\%$, LHE置1
- 如果 $D1 < 14.06\%$, LHE不置1
- 如果 $14.06\% < D1 < 14.84\%$, LHE可能置1也可能不置1, 这取决于 f_{MASTER} 时钟和UART_RX管脚上信号的相移。

第二次检测基于LIN同步域每一个下降沿之间的测量。

- 如果 $D2 > 18.75\%$, LHE置1
- 如果 $D2 < 15.62\%$, LHE不置1
- 如果 $15.62\% < D2 < 18.75\%$, LHE可能置1也可能不置1, 这取决于 f_{MASTER} 时钟和UART_RX管脚上信号的相移。

注意UART无需检查下一个边沿是否发生得比预期慢, 因为该检测已经被对整个同步域的偏移检测覆盖。

注意：偏移检测基于当前波特率而不是名义上的波特率。因此为了保证偏移检测的正确，应当在接收到新的断开符前，对波特率发生器重载名义上的波特值。这个名义上的波特值应当在初始化时由用户编写。为此，软件必须在接收校验求和之前把LDUM位置1。

如果LDUM位置1，在下一个字符接收时会自动用名义波特值重载波特率发生器。

用户也可以通过写入BRR1和BRR2寄存器来重载名义波特值。这种方法一般在相应发送和接收过程中发生错误时使用。

如果由于任何原因，在UART接收新的断开符和同步域时把LDUM位置1，该位会被忽略并且被清0。UART会按照由同步域计算来的波特值调整波特率发生器。

LIN报文头错误侦测

如果以下情况之一发生，则LHE置1

- 断开分界符过短
- 同步域的偏移错误超出LIN规范规定：主从的振荡器之间周期偏移允许范围为 $+/-14\%$
- 同步域或者标识符域帧错误
- LIN报文头接收超时
- 同步域测量时溢出，并导致分频数寄存器溢出

LIN报文头超时错误

章节LIN报文头超时错误的描述在自动重同步使能的情况下依然适用。

同步后UART时钟的允许误差

在同步完毕并接收完LIN断开符以后，UART时钟允许的误差与在UART模式下一致，具体解释如下：

在接受过程中，每一位过采样16次，其中取第8, 9, 10次采样的平均值作为这一位的值。

时钟频率在一位时间内，不应当偏移超过 $6/16(37.5\%)$ 。

采样时钟在每个起始位重新同步，这样在接受10位数据(1起始位，8数据位，1停止位)的时间内，时钟偏移不应当超过3.75%。

未经同步UART时钟的允许误差

在LIN从设备未经同步时(即在一段较长的时间内未接收任何字符)，UART时钟允许的最大偏移是 $+/-14\%$ 。

如果偏移在此范围内，接收新数据时，LIN断开符即可被正确识别。

可能有以下情况：主设备发送13位低电平作为LIN断开符，被一个“较快”的从设备翻译成11位低电平($13\text{位} - 14\% = 11.18$)的LIN断开符。按照LIN规范，LIN断开符只有在它的长度大于 $t_{SBRKTS} = 10$ 时才有效。这意味着LIN断开符长度至少是11位低电平。

如果从设备的时钟偏移是 $+14\%$ (从设备较慢)，字符“00h”，即连续的9个低电平绝对不可以被认为是LIN断开符($9\text{位} + 14\% = 10.26$)。这样，LIN断开符长度至少是11位低电平。

时钟偏移原因

可能作为总的时钟偏移的部分有：

- **DTRA**：由发送错误造成的偏移。注意：发送方可以是主，也可以是从(此时从设备接收另一个从设备的响应)。
- **DMEAS**：由接收方对LIN同步域测量引起的错误。
- **DQUANT**：由接收方对波特率量化引起的错误。
- **DREC**：接收方本地振荡器的偏移：假设在消息的起始时已经对偏移进行了补偿的情况下，这种偏移可能发生在接受一个完整LIN消息的过程中。
- **DTCL**：传输线路引起的偏移(通常由发送方造成)

- 系统的全部偏移应当累加，最终结果和UART时钟允许偏移比较：

$$- \text{DTRA} + \text{DMEAS} + \text{DQUANT} + \text{DREC} + \text{DTCL} < 3.75\%$$

LIN同步域测量引起的错误

LIN同步域测量的时间超过8个位时间。

测量通过一个计数器进行，该计数器时钟由CPU时钟提供。边沿检测也使用CPU时钟周期来同步。

这就是以2个CPU周期为单位的时基来对持续 $8 * \text{UARTDIV}$ 个时钟周期的信号进行测量。

这样，错误(DMEAS)等于：

$$2 / (8 * \text{UARTDIVMIN})$$

UARTDIVMIN表示最小的LIN分频值，代表了考虑 $+/-14\%$ 偏移因素的最大的波特率。

波特率量化引起的错误

波特率可以以 $1 / (\text{UARTDIV})$ 的步长调整。最坏情况发生在“真实”波特率正好在2步中间。

这样，量化错误DQUANT等于 $1 / (2 * \text{UARTDIVMIN})$ 。

在最大波特率时时钟偏移的影响

名义波特率的选择(UARTDIVNOM)会对量化错误DQUANT和测量错误DMEAS同时产生影响。最坏情况发生在UARTDIVMIN时。

这样，在给定的CPU频率的情况下，确定最大可能的名义波特率值应当考虑到时钟的最大偏移，并受下面等式的约束：

$$\text{DTRA} + 1 / (2 * \text{UARTDIVMIN}) + \text{DREC} + \text{DTCL} < 3.75\%$$

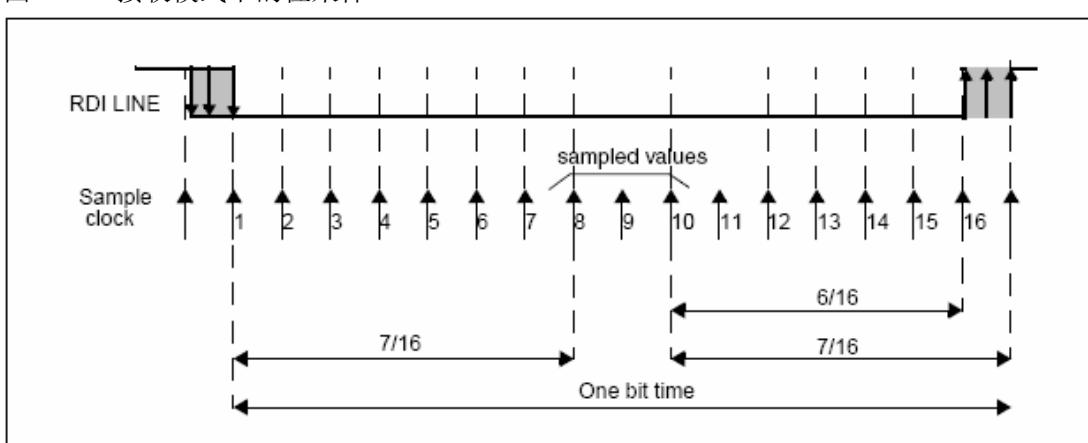
例：

名义波特率为20 Kbits/s， $\text{TCPU} = 125 \text{ ns}(8 \text{ MHz})$ ，这样 $\text{UARTDIVMIN} = 25$ 。

$$\text{UARTDIVMIN} = 25 - 0.15 * 25 = 21.25$$

$$\text{DQUANT} = 1 / (2 * \text{UARTDIVMIN}) = 0.0015\%$$

图124 接收模式下的位采样



22.4.4 LIN模式选择

表51 LIN模式选择

LINE	LSLV	LASE	含义
0	0	0	LIN模式禁用

1	1		LIN主模式
		0	LIN从模式 自动重同步禁用
	1	1	LIN从模式 自动重同步使能

22.5 低功耗模式

表52 UART接口在低功耗模式时表现

模式	描述
等待 WAIT	对UART没有影响 UART中断可以将MCU从WAIT模式唤醒
停机 HALT	UART寄存器被冻结 在HALT模式下, UART停止发送/接收直到退出HALT模式

22.6 中断

表53 UART中断请求

中断事件	事件标志	使能位	退出等待(WAIT) 模式	退出停机(HALT) 模式
发送数据寄存器空	TXE	TIEN	是	否
发送完成	TC	TCIEN	是	否
接收数据就绪可读	TXNE	RIEN	是	否
检测到过载 / LIN报文头错误	OR/LHE		是	否
检测到空闲线路	IDLE	ILEIEN	是	否
奇偶检验错	PE	PIEN	是	否
断开标志	LBDF	LBDIE	是	否
报文头标志位	LHDF	LHDIEN	是	否

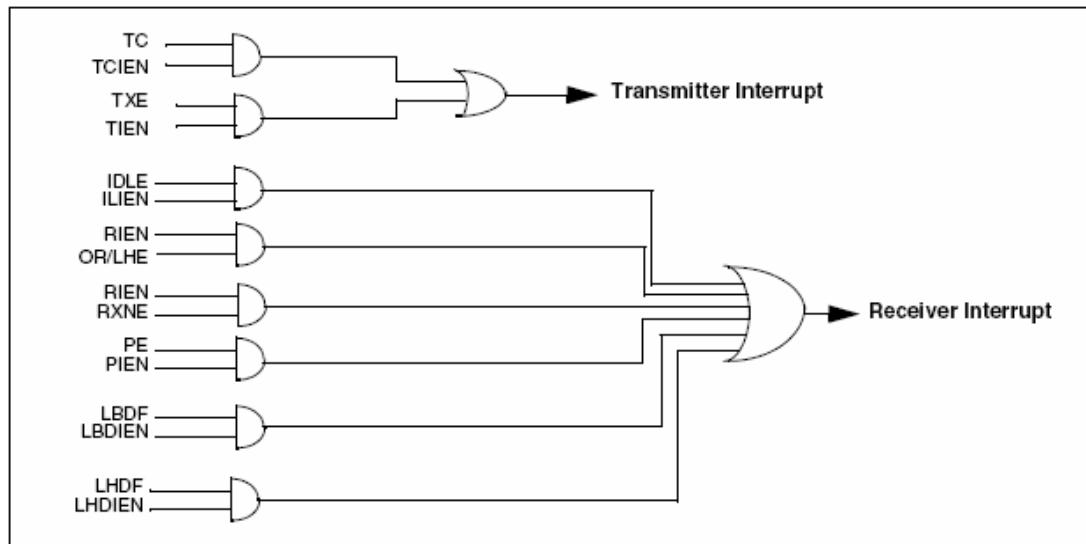
注意: 1. UART的中断事件被连接到2个中断向量(见图125)。

a) 发送完成或者发送寄存器空中断。

b) 检测到空闲线路, 过载错误, 接收寄存器满, 校验错误中断和噪声标志位。

2. 如果对应的使能控制位被置1, 并且相应的在CC寄存器的中断屏蔽被置0, 这些事件就会产生相应的中断。

图125 UART中断映射图



22.7 UART寄存器描述

22.7.1 状态寄存器(UART_SR)

地址偏移值: 0x00

复位值: 0xC0

7	6	5	4	3	2	1	0
TXE	TC	RXNE	IDLE	OR/LHE	NF	FE	PE

r rc_w0 rc_w0 r r r r r

位7	TXE: 发送数据寄存器空 当TDR寄存器中的数据被硬件转移到移位寄存器的时候, 该位被硬件置位。如果UART_CR2寄存器中的TIEN位为1, 则产生中断。对UART_DR的写操作会使该位清零。 0: 数据还没有被转移到移位寄存器; 1: 数据已经被转移到移位寄存器。
位6	TC: 发送完成 当包含有数据的一帧发送完成后, 由硬件将该位置位。如果UART_CR2中的TCIEN为1, 则产生中断。可用用户程序清除该位(先读UART_SR, 然后写入UART_DR)。对于UART2和UART3, 该位也可以通过写入0来清除。 0: 发送还未完成; 1: 发送完成。
位5	RXNE: 读数据寄存器非空 当RDR移位寄存器中的数据被转移到UART_DR寄存器中, 该位被硬件置位。如果UART_CR1寄存器中的RXNEIE为1, 则产生中断。对UART_DR的读操作可以将该位清零。RXNE位也可以通过写入0来清除, 对于UART2和UART3, 该位也可以通过写入0来清除。 0: 数据没有收到; 1: 收到数据, 可以读出。
位4	IDLE: 监测到IDLE总线 ⁽¹⁾ 当检测到空闲总线时, 该位被硬件置位。如果UART_CR1中的ILIEN为1, 则产生中断。由软件按下列操作顺序清0该位(先读UART_SR, 然后读UART_DR)。 0: 没有检测到空闲总线; 1: 检测到空闲总线。
位3	OR: 过载错误 ⁽²⁾ 当RXNE=1, 并且当前接收到的数据在移位寄存器中就绪, 准备转移到RDR寄存器时, 该位由硬件置1。如果UART_CR2寄存器中的RIEN为1, 则产生中断。由软件按下列操作顺序将该位清零(先读UART_SR, 然后读UART_DR)。 0: 没有过载错误; 1: 检测到过载错误。 LHE: LIN报文头错误(LIN从模式) 在LIN报文头接收期间, 该位表示四种错误类型: - 断开分界符过短 - 同步域错误 - 偏移错误(如果LASE =1) - 标识符帧错误 0: 没有LIN报文头错误; 1: 检测到LIN报文头错误。

位2	NF: 噪声标志位 ⁽³⁾ 在接收到的帧检测到噪音时, 由硬件对该位置位。由软件按下列操作顺序清0该位(先读UART_SR, 然后读UART_DR)。 0: 没有检测到噪音; 1: 检测到噪音。
位1	FE: 帧错误 ⁽⁴⁾ 当检测到同步错位, 过多的噪音或者检测到break符, 该位被硬件置位。由软件按下列操作顺序清0该位(先读UART_SR, 然后读UART_DR)。 0: 没有检测到帧错误; 1: 检测到帧错误或者break符。
位0	PE: 奇偶校验错误 在接收模式下, 如果出现奇偶校验错误, 硬件对该位置位。由软件按下列操作顺序清0该位(先读UART_SR, 然后读UART_DR)。在清除PE位前, 软件必须等待RXNE标志位被置1。如果UART_CR1中的PIEN为1, 则产生中断。 0: 没有校验错误; 1: 校验错误。

1. IDLE位直到RXNE位自己置1前不会在再次置1(例如一个新IDLE总线发生)。
2. 当该位置1, RDR寄存器的内容不会丢失, 但是移位寄存器会被覆盖。
3. 该位不会产生中断, 因为它和RXNE一起出现, 而RXNE标志置位时能够产生中断
4. 该位不会产生中断, 因为它和RXNE一起出现, 而RXNE标志置位时能够产生中断。如果当前传输的字符既产生帧错误又产生过载错误, 该字会被发送但仅有OR位会置1。

22.7.2 数据寄存器(UART_DR)

地址偏移值: 0x01

复位值: 未定义

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

DR[7:0]

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

位7:0	DR[7:0]: 数据值 包含了发送或接收的数据，其值取决于对该寄存器的操作是读取还是写入。 由于它是由两个寄存器组成的，一个给发送用(TDR)，一个给接收用(RDR)，该寄存器兼具读和写的功能。 TDR寄存器提供了内部总线和输出移位寄存器之间的并行接口。 RDR寄存器提供了输入移位寄存器和内部总线之间的并行接口。
------	---

22.7.3 波特比率寄存器 1(UART_BRR1)

波特率寄存器对于发送方和接收方来说是一致的。波特率通过对2个寄存器BRR1和BRR2编程来确定。写BRR2(如果需要)应当先于在写BRR1, 因为对BRR1的写操作会更新波特计数器。

请参考[图105在BRR寄存器里如何编写UART_DIV](#)和[表49设置波特率时的误差计算](#)。

注意: 如果TE或RE被分别禁止, 波特计数器停止计数

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
UART_DIV[11:4]							
rw	rw	rw	rw	-	rw	rw	rw
位7:0	UART_DIV[11:4]: UART_DIV位⁽¹⁾ 这8位定义了16位UART分频数的第二, 三半位元组						

1. BRR1 = 00h意味着UART时钟被禁用.

22.7.4 波特比率寄存器 2 (UART_BRR2)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
UART_DIV[15:12]				UART_DIV[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw
位7:4	UART_DIV[15:4]: UART_DIV位 这8位定义了16位UART分频数的MSB						
位3:0	UART_DIV[11:4]: UART_DIV位 这8位定义了16位UART分频数的LSB						

22.7.5 控制寄存器 1(UART_CR1)

地址偏移值: 0x04

复位值: 0x00

7	6	5	4	3	2	1	0
R8	T8	UARTD	M	WAKE	PCEN	PS	PIEN
rw	rw	rw	rw	rw	rw	rw	rw

位7	R8: 接收数据位8 该位用来在M=1时存放接收到字的第9位
位6	T8: 接收数据位8 该位用来在M=1时存放待发送字的第9位
位5	UARTD: UART禁用(用以实现低功耗) 当该位置1, UART预分频器和输出在当前字节传输完成后停止工作, 用来降低功耗。该位由软件置1或者清0 0: UART使能; 1: UART预分频器和输出禁用。
位4	M: 字长 该位定义了数据字的长度, 由软件对其置位和清零操作 0: 一个起始位, 8个数据位, n个停止位(n取决于UART_CR3中的STOP[1:0]位) 1: 一个起始位, 9个数据位, 一个停止位。 注意: 在数据传输过程中(发送或者接收时), 不能修改这个位。 在LIN从模式, M位和UART_CR3寄存器的STOP[1:0]应当保持为0
位3	WAKE: 唤醒的方法 这位决定了把USART唤醒的方法, 由软件对该位置位或者清零。 0: 被空闲总线唤醒; 1: 被地址标记唤醒。
位2	PCEN: 奇偶校验控制使能 UART模式: 用该位来选择是否进行硬件奇偶校验控制(对于发送来说就是校验位的产生; 对于接收来说就是校验位的检测)。当使能了该位, 在发送数据的MSB(如果M=1, MSB就是第9位; 如果M=0, MSB就是第8位)位后插入校验位; 对接收到的数据检查其校验位。软件对它置位或者清'0'。一旦该位被置位, 当前字节传输完成后, 校验控制才生效。 0: 奇偶校验控制被禁用; 1: 奇偶校验控制被使能。 LIN从模式: 在LIN从模式下, 该位使能LIN标识符奇偶校验检测 0: 标识符奇偶校验控制被禁止; 1: 标识符奇偶校验控制被使能。
位1	PS: 奇偶校验选择 该位用来选择当奇偶校验控制使能后, 是采用偶校验还是奇校验。软件对它置位或者清零。当前字节传输完成后, 该选择生效。 0: 偶校验; 1: 奇校验。
位0	PIEN: 校验中断使能 软件对该位置位或者清零 0: 中断被禁止; 1: 当USART_SR中的PE为1时, 产生USART中断。

22.7.6 控制寄存器 2(UART_CR2)

地址偏移值: 0x05

复位值: 0x00

7	6	5	4	3	2	1	0
TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	SBK
rw	rw	rw	rw	rw	rw	rw	rw
位7	TIEN: 发送中断使能 软件对该位置位或者清零 0: 中断被禁止; 1: 当USART_SR中的TXE为1时, 产生USART中断。						
位6	TCIEN: 发送完成中断使能 软件对该位置位或者清零 0: 中断被禁止; 1: 当USART_SR中的TC为1时, 产生USART中断。						
位5	RIEN: 接收中断使能 软件对该位置位或者清零 0: 中断被禁止; 1: 当USART_SR中的OR或者RXNE为1时, 产生USART中断。						
位4	ILIEN: IDLE中断使能 软件对该位置位或者清零 0: 中断被禁止; 1: 当USART_SR中的IDLE为1时, 产生USART中断。						
位3	TEN: 发送使能 ⁽¹⁾⁽²⁾ 该位使能发送器。软件对该位置位或者清零 0: 发送被禁止; 1: 发送被使能。						
位2	REN: 接收使能 软件对该位置位或者清零 0: 接收被禁止; 1: 接收被使能, 开始搜寻RX引脚上的起始位。						
位1	RWU: 接收唤醒 UART模式: 该位用来决定是否把USART置于静默模式。软件对该位置位或者清零。当一个唤醒序列被识别出来时, 硬件也会将其清零。 ⁽³⁾⁽⁴⁾ LIN模式: 在LIN从模式下, 设置RWU位允许对LIN报文头的检测而拒绝接收其他字符。参见章节静默模式与错误。在LIN从模式下, , 当RDRF位置1时, 软件不能设置或者清零RWU位。 0: 接收器处于正常工作模式; 1: 接收器处于静默模式。						
位0	SBK: 发送断开帧 使用该位来发送断开字符。软件可以对该位置位或者清零。应该由软件来置位它, 然后在断开帧的停止位时, 由硬件将该位复位。 0: 没有发送断开字符; 1: 将要发送断开字符。						

1. 在发送过程中, TEN位上的"0"脉冲("0"然后"1")会发送在当前字发送完毕后发送导言字节(空闲线)。
2. TEN置1后, 到传输开始之前有1个位时间的延时。

3. 在选中静默模式前(**RWU**位置1), **UART**必须首先接收一个数据字节, 否则在通过空闲线(**idle line**)检测从静默模式下唤醒的功能无效。
4. 在地址标记检测唤醒配置模式下(**WAKE**位 = 1), 且**RXNE**位为1时, **RWU**位不能由软件修改。

22.7.7 控制寄存器3(UART_CR3)

地址偏移值: 0x06

复位值: 0x00

7	6	5	4	3	2	1	0
保留	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL
	rw	rw	rw	rw	rw	rw	rw
位7	保留位, 必须保持清0						
位6	LINEN: LIN模式使能 软件对该位置位或者清零。 0: LIN模式被禁止; 1: LIN模式被使能。						
位5:4	STOP: 停止位 用来设置停止位的位数。 00: 1个停止位; 01: 保留 10: 2个停止位; 11: 1.5个停止位; 注意: 对于LIN从模式, 这两位应该都是0。						
位3	CLKEN: 时钟使能 该位用来使能SCLK引脚。 0: SCK引脚被禁止; 1: SCK引脚被使能。 注: UART3上不存在这一位。						
位2	CPOL: 时钟极性 ⁽¹⁾ 用户可以用该位来选择同步模式下SCLK引脚上时钟输出的极性。和CPHA位一起配合来产生用户希望的时钟/数据的采样关系 0: 总线空闲时SCLK引脚上保持低电平; 1: 总线空闲时SCLK引脚上保持高电平。 注: UART3上不存在这一位。						
位1	CPHA: 时钟相位 ⁽¹⁾ 用户可以用该位来选择同步模式下SCLK引脚上时钟输出的相位。和CPOL位一起配合来产生用户希望的时钟/数据的采样关系(参见图109和图110)。 0: 时钟第一个边沿进行数据捕获; 1: 时钟第二个边沿进行数据捕获。 注: UART3上不存在这一位。						
位0	LBCL: 最后一位时钟脉冲 ⁽¹⁾⁽²⁾ 使用该位来控制是否在同步模式下, 在SCLK引脚上输出最后发送的那个数据字节(MSB)对应的时钟脉冲 0: 最后一位数据的时钟脉冲不从SCLK输出; 1: 最后一位数据的时钟脉冲会从SCLK输出。 注: UART3上不存在这一位。						

1. 这三位(CPOL, CPHA, LBCL)在发送端被使能的时候不能对其进行写操作。
2. 根据UART_CR1寄存器的M位选择的8位或者9位数据格式, 最后一位是指被发送数据的第8位或者第9位。

22.7.8 控制寄存器 4(UART_CR4)

地址偏移值: 0x07

复位值: 0x00

7	6	5	4	3	2	1	0
保留	LBDIEN	LBDL	LBDF	ADD[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw
位7	保留位, 必须保持清0						
位6	LBDIEN: LIN断开符检测中断使能 断开符中断屏蔽(使用断开定界符来检测断开符) 0: LIN断开符检测中断被禁止; 1: LIN断开符检测中断被使能。						
位5	LBDL: LIN断开符检测长度 该位用来选择是11位还是10位的断开符检测 0: 10位的断开符检测; 1: 11位的断开符检测。						
位4	LBDF: LIN断开符检测标志位 LIN断开符检测标志位(状态标志位) 0: 没有检测到LIN断开符; 1: 检测到LIN断开符。 如果LBDIEN=1, 那么LBDF=1就会产生中断						
位3:0	ADD[3:0]: UART节点地址 该位域定义了UART节点的地址。 用于在多处理器通讯的静默状态下的地址标识唤醒检测。						

22.7.9 控制寄存器 5(UART_CR5)

地址偏移值: 0x08

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

保留	SCEN	NACK	HDSEL	IRLP	IREN	保留
	r	r	rw	rw	rw	

位7:6	保留位, 必须保持清0
位5	SCEN: 智能卡模式使能 该位用来使能智能卡模式 0: 智能卡模式使能; 1: 智能卡模式被禁止。 注: UART3上不存在这一位。
位4	NACK: 智能卡NACK使能 0: 校验错误出现时, 不发送NACK; 1: 校验错误出现时, 发送NACK。 注: UART3上不存在这一位。
位3	HDSEL: 半双工选择 选择单线半双工模式 0: 不选择半双工模式; 1: 选择半双工模式。 注: UART2和UART3上不存在这一位。
位2	IRLP: 红外低功耗 该位用来选择普通模式还是低功耗红外模式 0: 普通模式; 1: 低功耗模式。 注: UART3上不存在这一位。
位1	IREN: 红外模式使能 由软件对该位清零或者置位 0: 红外被禁止; 1: 红外使能。 注: UART3上不存在这一位。
位0	保留位, 必须保持清0

22.7.10 控制寄存器 6(UART_CR6)

地址偏移值: 0x09

复位值: 0x00

7	6	5	4	3	2	1	0
LDUM	-	LSLV	LASE	-	LHDIEN	LHDF	LSF
rw		rw	rw		rw	rc_w0	rc_w0

注意: 该寄存器不存在于UART1

位7	LDUM : LIN分频数更新方法 0: LDIV在对BRR1写入以后立即更新(如果同时没有自动重同步引起LDIV更新的话) 1: LDIV在对BRR1写入以后, 在下一个接收到字符时(RXNE = 1)更新 LDIV的编写通过写BRR1和BRR2两个寄存器实现。 一旦LDIV随着同步域结束时得到测量到的波特率时被更新, 则该位由硬件自动清0。
位6	保留位, 必须保持清0
位5	LSLV : LIN从模式使能 0: LIN主模式; 1: LIN从模式。
位4	LASE : LIN自动重同步使能 0: LIN自动重同步禁用; 1: LIN自动重同步使能。
位3	保留位, 必须保持清0
位2	LHDIEN : LIN报文头检测中断使能 报文头中断屏蔽。 0: LIN报文头检测中断禁用; 1: LIN报文头检测中断使能。
位1	LHDF : LIN报文头检测标志位 该位在LIN从模式下检测到LIN报文头时由硬件置1, 通过软件写0来清零 0: 没有检测到LIN报文头; 1: 检测到LIN报文头。 如果LHDIEN=1, 那么LHDF=1就会产生中断
位0	LSF : LIN同步域 该位提示LIN同步域已经被解析。该位仅用于LIN从模式。在自动重同步模式下(LASE=1), 当UART处于LIN同步域状态, 会等待或者对RDI线路上的下降沿计数。 一旦检测到LIN断开符, 该位即由硬件置1。当LIN同步域解析完成, 该位由硬件置0。通过软件对该位写0也可以清零该位, 并退出LIN同步域状态, 返回空闲状态。 0: 当前字符不是LIN同步域; 1: LIN同步域(LIN同步域解析进行中)。

22.7.11 保护时间寄存器(UART_GTR)

地址偏移值: 0x09(UART1), 0x0A(UART2)

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

GT[7:0]

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

位7:0	GT[7:0]: 保护时间值
------	-----------------------

该位域规定了以波特时钟为单位的保护时间的值。在智能卡模式下，需要这个功能。当保护时间过去后，发送完成标志才被置起。

注: UART3上不存在这些位。

22.7.12 分频寄存器(UART_PSCR)

地址偏移值: 0x0A(UART1), 0x0B(UART2)

复位值: 0x00

注意: 在同时使用智能卡和IrDA接口的时候, 必须对该寄存器写入正确的值

7	6	5	4	3	2	1	0
PSC[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW
<p>位7:0</p> <p>PSC[7:0]: 预分频器值</p> <p>- 在红外低功耗模式下:</p> <p>PSC[7:0]=红外低功耗波特率⁽¹⁾</p> <p>对系统时钟分频已到达低功耗的频率:</p> <p>时钟源被寄存器中的值(仅有8位有效)分频</p> <p>00000000: 保留 – 不要写入该值;</p> <p>00000001: 对源时钟1分频;</p> <p>00000010: 对源时钟2分频;</p> <p>.....</p> <p>- 在智能卡模式下:</p> <p>PSC[4:0]:预分频值⁽²⁾⁽³⁾</p> <p>对系统时钟进行分频, 给智能卡提供时钟。</p> <p>寄存器中给出的值 (5个有效位) 乘以2后, 作为对时钟源的分频因子</p> <p>00000: 保留 – 不要写入该值;</p> <p>00001: 对源时钟进行2分频;</p> <p>00010: 对源时钟进行4分频;</p> <p>00011: 对源时钟进行6分频;</p> <p>.....</p> <p>注意: 在UART3上没有这些位。</p>							

1. 如果未使能IrDA模式, 那么分频设置是无效的。
2. 如果未使能智能卡模式, 那么分频设置是无效的。
3. 即使智能卡模式使能, 位[7:5]也是无效的。

22.7.13 UART寄存器地址映射

表54 UART1寄存器列表及其复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0							
0x00	UART_SR	TXE	TC	RXNE	IDLE	0R	NF	FE	PE							
	复位值															
0x01	UART_DR	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DRO							
	复位值															
0x02	UART_BRR1	UART_DIV[11:4] 00000000														
	复位值															
0x03	UART_BRR2	UART_DIV[15:12] 0000				UART_DIV[3:0] 0000										
	复位值															
0x04	UART_CR1	R8	T8	UARTD	M	WAKE	PCEN	PS	PIEN							
	复位值															
0x05	UART_CR2	TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	SBK							
	复位值															
0x06	UART_CR3	-	LINEN	STOP 00	CKEN	CPOL	CPHA	LBCL	0							
	复位值															
0x07	UART_CR4	-	LBDIEN	LBDL	LBDF	ADD[3:0] 0000										
	复位值															
0x08	UART_CR5	-	-	SCEN	NACK	HDSEL	IRLP	IREN	-							
	复位值															
0x09	UART_GTR	GT7	GT6	GT5	GT4	GT3	GT2	GT1	GT0							
	复位值															
0x0A	UART_PSCR	PSC7	PSC6	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0							
	复位值															

表55 UART2寄存器列表及其复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	UART_SR	TXE	TC	RXNE	IDLE	0R	NF	FE	PE
	复位值	1	1	0	0	0	0	0	0
0x01	UART_DR	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	复位值	x	x	x	x	x	x	x	x
0x02	UART_BRR1	UART_DIV[11:4] 00000000							
	复位值								
0x03	UART_BRR2	UART_DIV[15:12] 0000				UART_DIV[3:0] 0000			
	复位值								
0x04	UART_CR1	R8	T8	UARTD	M	WAKE	PCEN	PS	PIEN
	复位值	0	0	0	0	0	0	0	0
0x05	UART_CR2	TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	SBK
	复位值	0	0	0	0	0	0	0	0
0x06	UART_CR3	-	LINEN	STOP 00		CKEN	CPOL	CPHA	LBCL
	复位值	0	0			0	0	0	0
0x07	UART_CR4	-	LBDIEN	LBDL	LBDF	ADD[3:0] 0000			
	复位值	0	0	0	0				
0x08	UART_CR5	-	-	SCEN	NACK	HDSEL	IRLP	IREN	-
	复位值	0	0	0	0	0	0	0	0
0x09	UART_CR6	LDUM	-	LSLV	LASE	-	LHDIEN	LHDF	LSF
	复位值	0	0	0	0	0	0	0	0
0x0A	UART_GTR	GT7	GT6	GT5	GT4	GT3	GT2	GT1	GT0
	复位值	0	0	0	0	0	0	0	0
0x0B	UART_PSCR	PSC7	PSC6	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0
	复位值	0	0	0	0	0	0	0	0

表56 UART3寄存器列表及其复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	UART_SR	TXE	TC	RXNE	IDLE	0R	NF	FE	PE
	复位值	1	1	0	0	0	0	0	0
0x01	UART_DR	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	复位值	X	X	X	X	X	X	X	X
0x02	UART_BRR1	UART_DIV[11:4] 00000000							
	复位值	0000							
0x03	UART_BRR2	UART_DIV[15:12]				UART_DIV[3:0]			
	复位值	0000				0000			
0x04	UART_CR1	R8	T8	UARTD	M	WAKE	PCEN	PS	PIEN
	复位值	0	0	0	0	0	0	0	0
0x05	UART_CR2	TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	SBK
	复位值	0	0	0	0	0	0	0	0
0x06	UART_CR3	-	LINEN	STOP 00		CKEN	CPOL	CPHA	LBCL
	复位值	0	0	00		0	0	0	0
0x07	UART_CR4	-	LBDIEN	LBDL	LBDF	ADD[3:0] 0000			
	复位值	0	0	0	0	0000			
0x08	UART_CR5	-	-	SCEN	NACK	HDSEL	IRLP	IREN	-
	复位值	0	0	0	0	0	0	0	0
0x09	UART_CR6	LDUM	-	LSLV	LASE	-	LHDIEN	LHDF	LSF
	复位值	0	0	0	0	0	0	0	0

23 控制器局域网(beCAN)

23.1 简介

beCAN是基本扩展CAN(Basic Extended CAN)的缩写，它支持CAN协议2.0A和2.0B。它的设计目标是，以最小的CPU负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求(优先级特性可软件配置)。

对于安全紧要的应用，beCAN提供所有支持时间触发通信模式所需的硬件功能。

23.2 主要特点

- 支持CAN协议2.0A和2.0B主动模式
- 波特率最高可达1兆位/秒
- 支持时间触发通信功能
- 可选择时钟源(f_{MASTER} 或 f_{CANEXT})

发送

- 3个发送邮箱
- 发送报文的优先级特性可软件配置
- 记录发送SOF时刻的时间戳

接收

- 1个3级深度的接收FIFO
- 6个位宽可变的过滤器组
- 标识符列表
- FIFO溢出处理方式可配置
- 记录接收SOF时刻的时间戳

时间触发通信模式

- 禁止自动重传模式
- 16位自由运行定时器
- 可配置定时器精度
- 可在最后2个数据字节发送时间戳

管理

- 中断可屏蔽
- 邮箱占用唯一的地址空间，便于提高软件效率

23.3 总体描述

在当今的CAN应用中，CAN网络的节点在不断增加，并且多个CAN常常通过网关连接起来，因此整个CAN网中的报文数量(每个节点都需要处理)急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

- 需要一个增强的过滤机制来处理各种类型的报文

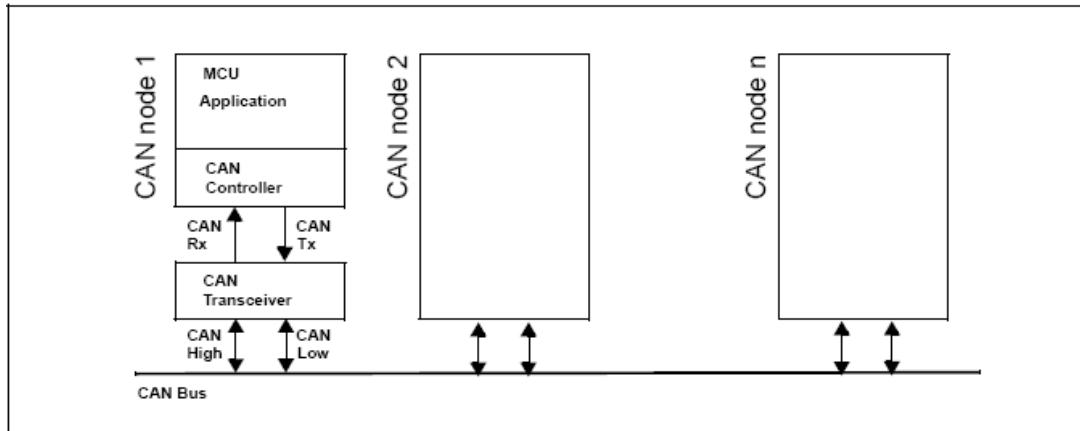
此外，应用层任务需要更多CPU时间，因此报文接收所需的实时响应程度需要减轻。

- 采用接收FIFO方式，从而允许CPU花很长时间处理应用层任务而不会丢失报文。

构筑在底层CAN驱动程序上的高层协议软件，要求跟CAN控制器之间有高效的接口。

- 所有邮箱和报文以16个字节为一页映射到同一个地址，通过页面选择寄存器选择页面。

图126 CAN网拓扑结构



23.3.1 CAN 2.0B (active)内核

beCAN模块可以完全自动地接收和发送CAN报文；且硬件完全支持标准标识符(11位)和扩展标识符(29位)。

23.3.2 控制、状态和配置寄存器

应用程序通过这些寄存器，可以：

- 配置CAN参数，如波特率
- 请求发送报文
- 处理报文接收
- 管理中断
- 获取诊断信息

23.3.3 发送邮箱

共有3个发送邮箱供软件来安排要发送报文。由发送调度器决定哪个邮箱的报文先被发送。

23.3.4 接收过滤器

共有6个位宽可变/可配置的标识符过滤器组，用来选择留下软件所需要的报文，丢弃其它报文。

接收FIFO

接收FIFO用于存储CAN控制器接收的报文，FIFO中可以存放3个完整的报文。软件可以在同一个地址访问下一个可用的报文。FIFO完全由硬件来管理。

图127 beCAN功能框图

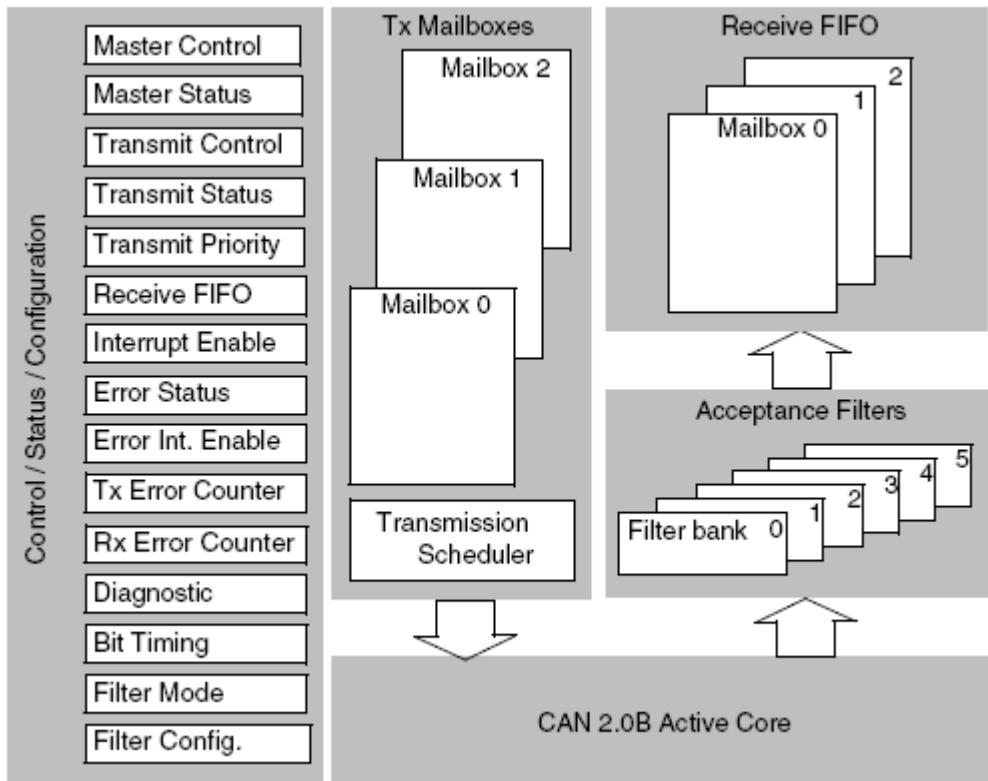
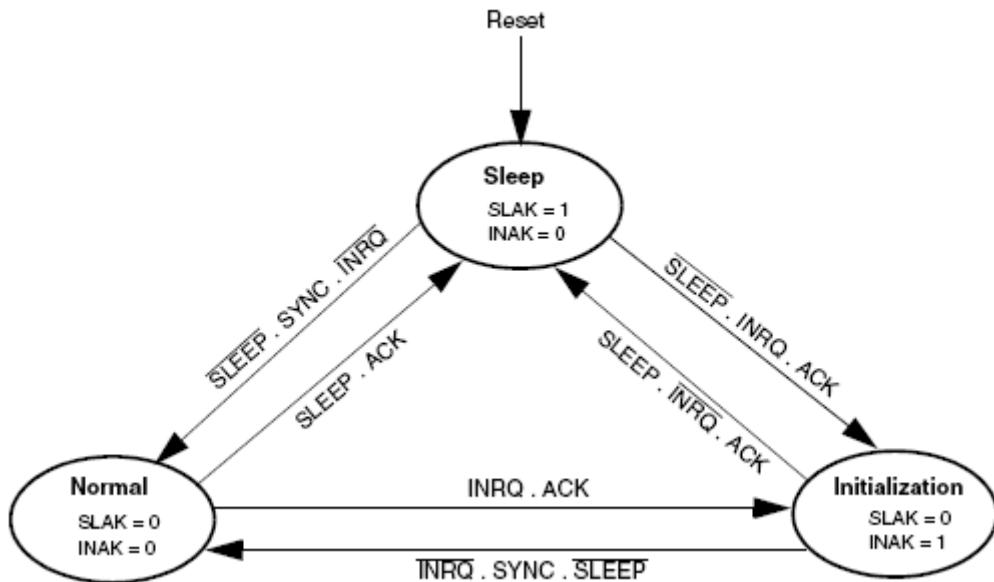


图128 beCAN工作模式



23.4 工作模式

beCAN有3个主要的工作模式：初始化模式、正常模式和睡眠模式。在硬件复位后，beCAN工作在睡眠模式以减少功耗。软件通过对CAN_MCR寄存器的INRQ或SLEEP位置‘1’，可以请求beCAN进入初始化或睡眠模式。一旦进入了初始化或睡眠模式，beCAN就对CAN_MSR寄存器的INAK或SLAK位置‘1’来进行确认。当INAK和SLAK位都为‘0’时，beCAN就处于正常模式。在进入正常模式前，beCAN必须跟CAN总线取得同步；为取得同步，beCAN要等待直到CAN总线处于空闲状态，即在CANRX引脚上监测到11个连续的隐性位。

23.4.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置CAN_MCR寄存器的INRQ位为'1'，请求beCAN进入初始化模式，然后等待硬件对CAN_MSR寄存器的INAK位置'1'来进行确认。

可以通过清除CAN_MCR寄存器的INRQ位，来请求beCAN退出初始化模式，一旦硬件对CAN_MSR寄存器的INAK位清'0'，beCAN就退出了初始化模式。不管怎样，退出初始化模式时RX引脚必须处于隐性状态。

当beCAN处于初始化模式时，禁止CANz总线上报文的接收和发送，并且CANTX引脚输出隐性位(高电平)。

进入初始化模式，不会改变寄存器的配置。

要初始化beCAN控制器，软件至少要对位时间特性寄存器和过滤器组进行设置。如果没有使用过滤器组，建议保持其处于非激活状态(使CAN_FCRx寄存器中相应的FACT位为0)

23.4.2 正常模式

在初始化完成后，软件必须请求硬件进入正常模式，以便正常接收和发送报文。软件可以通过对CAN_MCR寄存器的INRQ位清'0'，来请求从初始化模式进入正常模式，然后要等待硬件对CAN_MSR寄存器的INAK位置'1'的确认。再与CAN总线取得同步，即在CANRX引脚上监测到11个连续的隐性位(等效于总线空闲)后，beCAN才能正常接收和发送报文。

过滤器初值的设置不需要在初始化模式中完成，但必须在过滤器处于非激活状态下完成(相应的FACT位为0)。但是过滤器的位宽和模式的设置，则必须在初始化模式中进行。

23.4.3 睡眠模式(低功耗)

为了降低功耗，beCAN可工作在低功耗模式---睡眠模式。根据软件的使用需求，通过对CAN_MCR寄存器的SLEEP位置'1'，以进入睡眠模式。在睡眠模式下，beCAN的时钟停止了，但软件仍然可以访问邮箱寄存器。

注：当beCAN处于睡眠模式时，软件通过对CAN_MCR寄存器的INRQ位置'1'来请求进入初始化模式，必须同时将SLEEP位清'0'，才能进入初始化模式。

可以通过两种方式将beCAN唤醒(退出睡眠模式)：由软件清除SLEEP位，或者当硬件检测到CAN总线的活动时。

如果CAN_MCR寄存器的AWUM位为'1'，一旦检测到CAN总线的活动，硬件就自动对SLEEP位清'0'来唤醒beCAN。如果CAN_MCR寄存器的AWUM位为'0'，当唤醒中断发生时，软件必须将SLEEP位清'0'以退出睡眠状态。

注：如果唤醒中断被使能(CAN_IER寄存器的WKUIE位为'1')，那么一旦检测到CAN总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒beCAN。

当SLEEP位被清'0'后，睡眠模式的退出必须与CAN总线同步，请参考[图128：beCAN工作模式](#)。不管怎样，在退出初始化模式时RX引脚必须处于隐性状态。当硬件对SLAK位清'0'时，就确认了睡眠模式的退出。

23.4.4 时间触发通讯模式

在该模式下，CAN的内部硬件计数器被激活，用于产生(Rx和Tx邮箱)时间戳，分别存储在CAN_MTSRH寄存器和CAN_MTSRL寄存器中，内部计数器在接收和发送的帧起始位的采样点位置被捕捉，生成时间戳。

TGT位(CAN_MDLCR寄存器中发送时间戳位)能将CAN_MTSRH寄存器和CAN_MTSRL寄存器里的内容以报文的最后两个字节自动发送出去(请参考TTCAN的规范ISO 11898-4)。在这种情况下，TTCM位(CAN_MCR寄存器中时间触发通讯模式位)必须置位以使能时间触发通讯机制。

23.5 测试模式

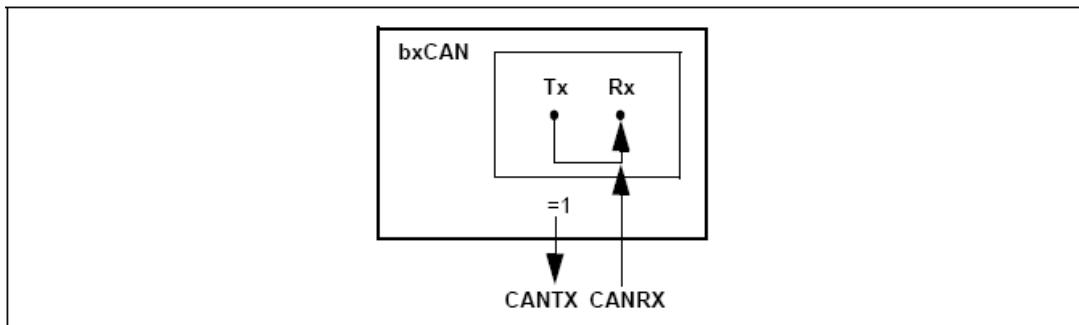
通过对CAN_DGR寄存器的SILM和/或LBKM位置'1'，来选择测试模式。这2位只能在初始化模式下修改。若已经在某一种测试模式时，可通过软件对CAN_MCR寄存器的INRQ位清'0'，以进入正常模式。

23.5.1 静默模式

通过对CAN_DGR寄存器的SILM位置'1'，来选择静默模式。

在静默模式下，beCAN可以正常地接收数据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果beCAN需要发出显性位(确认位、过载标志、主动错误标志)，那么这样的显性位在内部被接回来从而可以被CAN内核检测到，同时CAN总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析CAN总线的阻塞，而不会对总线造成影响——显性位的传输(确认位、错误帧)不会真正发送到总线上。

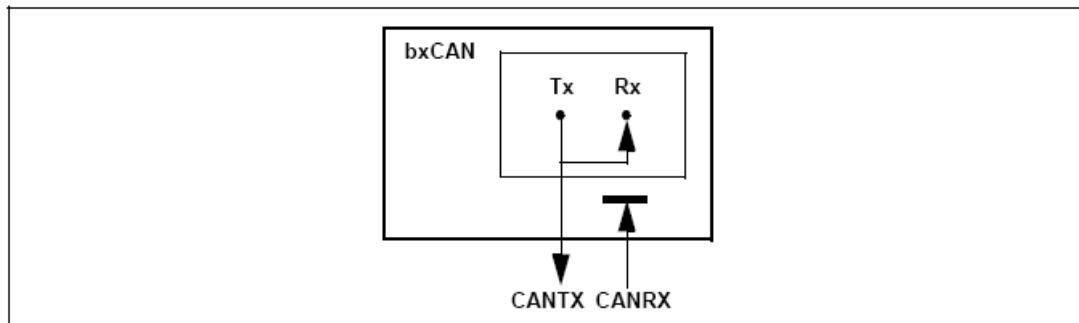
图129 beCAN工作在静默模式



23.5.2 环回模式

通过对CAN_DGR寄存器的LBKM位置'1'，来选择环回模式。在环回模式下，beCAN把发送的报文当作接收的报文并保存(如果可以通过接收过滤)在FIFO里。

图130 beCAN工作在环回模式



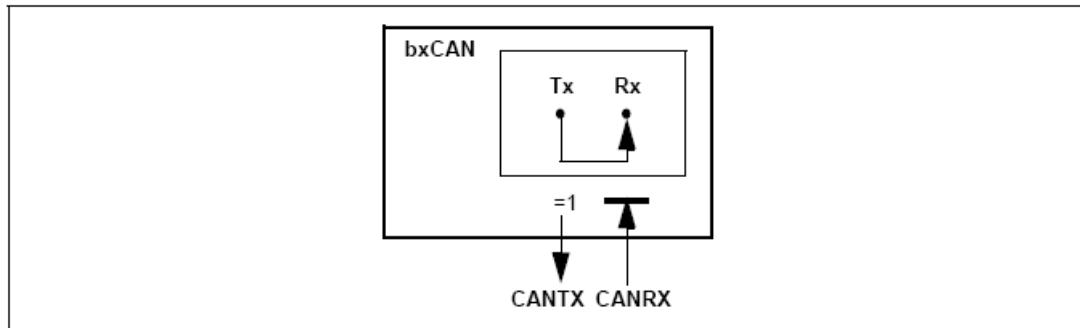
环回模式可用于自测试。为了避免外部的影响，在环回模式下CAN内核忽略确认错误(在数据/远程帧的确认位时刻，不检测是否有显性位)。在环回模式下，beCAN在内部把Tx输出回馈到Rx输入上，而完全忽略CANRX引脚的实际状态。发送的报文可以在CANTX引脚上检测到。

注：由于在环回模式下，Tx引脚仍然是激活的，需注意它能扰乱CAN总线上的通讯。

23.5.3 环回静默模式

通过对CAN_BTR寄存器的LBKM和SILM位同时置'1'，可以选择环回静默模式。该模式可用于“热自测试”，即可以象环回模式那样测试beCAN，但却不会影响CANTX和CANRX所连接的整个CAN系统。在环回静默模式下，CANRX引脚与CAN总线断开，同时CANTX引脚输出隐性位状态。

图131 beCAN工作在环回静默模式



23.6 功能描述

23.6.1 发送处理

发送报文的流程为：应用程序选择1个空发送邮箱；设置标识符、数据长度代码(DLC)和待发送数据；然后对CAN_MCSR寄存器的TXRQ位置'1'，来请求发送。一旦邮箱不再为空，软件对邮箱寄存器就不再有写的权限。TXRQ位置1后，邮箱马上进入挂号状态，等待成为最高优先级的邮箱，参见发送优先级。当邮箱成为最高优先级的邮箱，其状态就变为预定发送状态。当CAN总线进入空闲状态，预定发送邮箱中的报文就马上被发送(进入发送状态)。一旦邮箱中的报文被成功发送后，它就变为空邮箱；硬件相应地将CAN_TSR寄存器的RQCP位和TXOK位置1，来表明发送成功。

如果发送失败，失败是由于仲裁丢失的原因造成的，硬件将CAN_MCSR寄存器的ALST位置'1'；失败是由于发送出错的原因造成的，硬件将TERR位置'1'。

发送优先级

由标识符决定

当有超过1个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据CAN协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

由发送请求次序决定

通过对CAN_MCR寄存器的TXFP位置'1'，可以把发送邮箱配置为发送FIFO。在该模式下，发送的优先级由发送请求次序决定。

该模式对分段发送很有用。

中止

通过对CAN_MCSR寄存器的ABRQ位置'1'，可以中止发送请求。邮箱如果处于挂号或预定状态，发送请求马上就被中止了。如果邮箱处于发送状态，那么中止请求可能导致2种结果：如果邮箱中的报文被成功发送，那么邮箱变为空邮箱，并且CAN_MCSR寄存器和CAN_TSR寄存器的TXOK位被硬件置'1'；如果邮箱中的报文发送失败了，那么邮箱变为预定状态，然后发送请求被中止，邮箱变为空邮箱且TXOK位被硬件清'0'。因此无论上面那种情况，只要邮箱处于发送状态，那么在发送操作结束后，邮箱都会变为空邮箱。

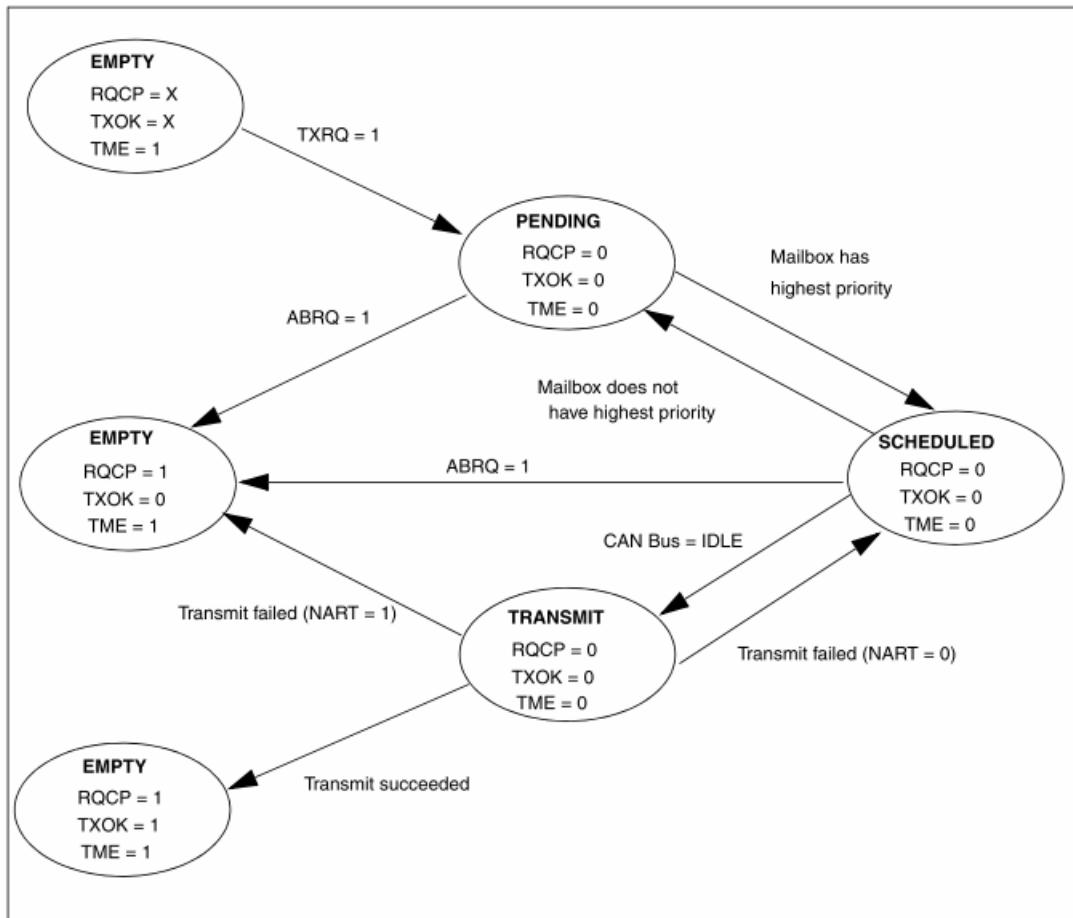
禁止自动重传模式

该模式主要用于满足CAN标准中，时间触发通信选项的需求。通过对CAN_MCR寄存器的NART位置'1'，来让硬件工作在该模式。

在该模式下，每个发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动重发该报文。

在一次发送操作结束后, 硬件认为发送请求已经完成, 从而将CAN_MCSR寄存器的RQCP位置'1', 同时发送的结果反映在CAN_MCSR寄存器的TXOK、ALST和TERR位上。

图132 发送邮箱状态



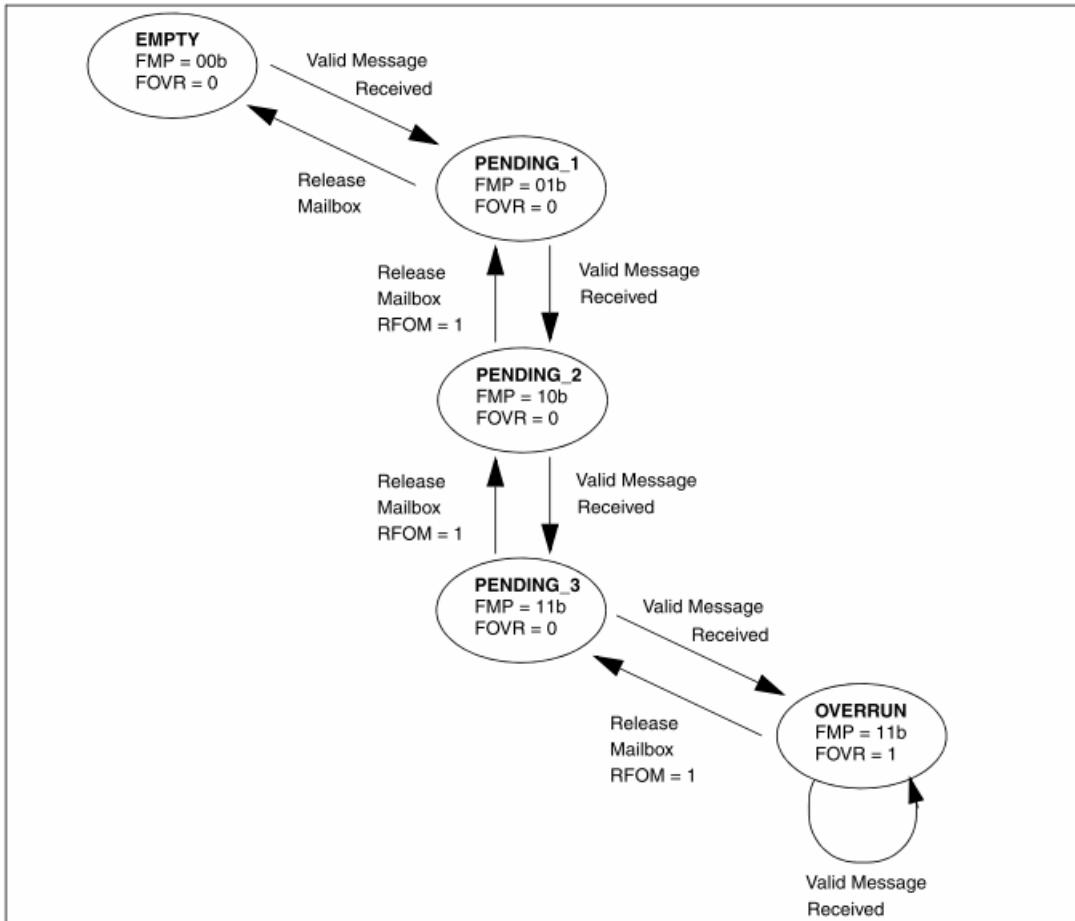
23.6.2 接收处理

接收到的报文, 被存储在3级深度的邮箱FIFO中。FIFO完全由硬件来管理, 从而节省了CPU的处理负荷, 简化了软件并保证了数据的一致性。应用程序只能通过访问FIFO的输出邮箱, 来读取FIFO中最先收到的报文。

有效报文

根据CAN协议, 当报文被正确接收(直到EOF域的最后一位都没有错误), 且通过了标识符过滤, 那么该报文被认为是有效报文。请参考[23.6.3: 标识符过滤](#)。

图133 接收FIFO状态



FIFO管理

FIFO起始状态为**空**，在接收到的第一个有效报文存入FIFO后，FIFO状态变为**挂号_1**(pending_1)，硬件相应地将CAN_RFR寄存器的FMP[1:0]位设置为'01'(二进制01b)。软件可以从FIFO输出邮箱来读出邮箱中的报文，然后通过将CAN_RFR寄存器的RFOM位设置'1'来释放邮箱，FIFO又变为空状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么FIFO仍然保留在**挂号_1**状态，软件仍然可以从FIFO输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO状态变为**挂号_2**(pending_2)，硬件相应地将FMP[1:0]设置为'10'(二进制10b)。重复上面的过程，第三个有效的报文把FIFO变为**挂号_3**状态(FMP[1:0]=11b)。此时，软件必须对RFOM位设置1来释放邮箱，以便FIFO可以有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失。参见[23.6.4: 报文存储](#)。

溢出

当FIFO处于**挂号_3**状态(即FIFO的3个邮箱都是满的)，下一个有效的报文就会导致**溢出**，并且一个报文会丢失。此时，硬件将CAN_RFR寄存器的FOVR位置'1'来表明溢出的状况。至于哪个报文会被丢弃，取决于对FIFO的设置：

- 如果禁用FIFO锁定功能(CAN_MCR寄存器的RFLM位被清'0')，那么FIFO中最后收到的报文就被新报文所覆盖。因此，最新的那个报文是不会丢失的。
注：先前收到的报文仍然保留再FIFO原有的位置中，只有最后一个报文会被覆盖。
- 如果启用FIFO锁定功能(CAN_MCR寄存器的RFLM位被置'1')，那么新近收到的报文被丢弃，软件可以读到FIFO中最早收到的3个报文。

与接收相关的中断

一旦往FIFO存入一个报文，硬件就会将FIFO的FMP[1:0]位从00b更新到01b，如果CAN_IER寄存器的FMPIE位为'1'，那么就会产生一个中断请求。

当FIFO变满时(即第3个报文被存入)，CAN_RFR寄存器的FULL位就被置'1'，如果CAN_IER寄存器的FFIE位为'1'，那么就会产生一个满中断请求。

在溢出的情况下，FOVR位被置'1'，如果CAN_IER寄存器的FOVIE位为'1'，那么就会产生一个溢出中断请求。

23.6.3 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RAM里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，beCAN为应用程序提供了6个可配置的、位宽可变的过滤器组(5~0)，用于只接收那些软件需要的报文。硬件过滤的做法节省了CPU开销，否则就必须由软件进行过滤，从而占用一定的CPU资源。每个过滤器组x包含8个8位寄存器：CAN_FxR[8:1]。

可变的位宽

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1个32位过滤器，包括：STDID[10:0]/EXTID[28:18]、IDE、EXID[17:0]和RTR位；
- 2个16位过滤器，包括：STDID[10:0]/EXTID[28:18]、IDE和RTR位；
- 4个8位过滤器，包括：STDID[10:3]/EXTID[28:21]，其他位可以不用关心；
- 1个16位过滤器和2个8位过滤器，具体的过滤器描述如上16位和8位过滤器描述。

可参见[图134](#)到[图137](#)。

此外过滤器可配置为，屏蔽位模式和标识符列表模式。

屏蔽位模式

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理。

标识符列表模式

在标识符列表模式下，屏蔽寄存器当作标识符寄存器用。因此，使用2个标识符来代替上面的标识符加屏蔽位的方式。接收报文标识符的每一位都必须跟过滤器标识符相同。

过滤器组位宽和模式的设置

过滤器组可以通过相应的CAN_FCRx寄存器配置。在配置一个过滤器组前，必须通过清除CAN_FCRx寄存器的FACT位，把它设置为禁用状态。通过设置CAN_FCRx的相应FSC[1:0]位，可以配置一个过滤器组的位宽。通过对CAN_FMRx的FMLx和FMHx位，可以配置对应的屏蔽/标识符寄存器的将其设置为标识符列表模式或屏蔽位模式，FMLx位用于定义该寄存器组的低半组(CAN_FxR1-4寄存器)的模式，FMHx位定义该寄存器组的高半组(CAN_FxR5-8寄存器)的模式，请参见[图134](#)到[图137](#)。

例如：

- 如果过滤器组1设置为2个16位的过滤器，那么FML1定义CAN_F1R3和CAN_F1R4寄存器的模式，FMH1定义CAN_F1R7和CAN_F1R8寄存器的模式。
- 如果过滤器组1设置为4个8位的过滤器，那么FML1定义CAN_F1R2和CAN_F1R4寄存器的模式，FMH1定义CAN_F1R6和CAN_F1R8寄存器的模式。

注：在32位的配置中，FMLx和FMHx位必须拥有相同的值以确保4个屏蔽位/标识符过滤器位处于相同的模式。

当接收到标准标识符(IDE位为0), 32位或16位过滤器的扩展部分不必比较。

为了过滤出一组标识符, 应该设置屏蔽位/标识符过滤器组工作为屏蔽位模式。

为了过滤出一个标识符, 应该设置屏蔽位/标识符过滤器组工作为标识符列表模式。

应用程序不用的过滤器组, 应该保持在禁用状态。

过滤器组中的每个过滤器, 都被编号为(叫做过滤器号)从0开始, 到某个最大数值——取决于6个过滤器组的模式和位宽的设置。

关于过滤器配置, 可参见图134到图137。

图134 32位过滤器组设置(CAN_FCRx寄存器的FSCx位为11b)

Filter registers							Filter mode ¹	
Mapping	STID[10:3] / EXID[28:21]	STID [2:0] / EXID[20:18]	RTR	IDE	EXID [17:15]	EXID [14:7]	EXID[6:0]	0
Identifier	CAN_FxR1	CAN_FxR2				CAN_FxR3	CAN_FxR4	
Identifier/Mask	CAN_FxR5	CAN_FxR6				CAN_FxR7	CAN_FxR8	
ID= Identifier n = Filter number M = Mask x = Filter bank number								
¹ The FMHx and FMLx bits are located in the CAN_FMR1 and CAN_FMR2 registers								

图135 16位过滤器组设置(CAN_FCRx寄存器的FSCx位为10b)

Filter registers							Filter mode ¹			
Mapping	STID[10:3] / EXID[28:21]	STID [2:0] / EXID[20:18]	RTR	IDE	EXID [17:15]		FMHx = 0 FMLx = 0	FMHx = 0 FMLx = 1	FMHx = 1 FMLx = 0	FMHx = 1 FMLx = 1
Identifier	CAN_FxR1	CAN_FxR2					ID n	ID n	ID n	ID n
Identifier/Mask	CAN_FxR3	CAN_FxR4					ID M	ID n+1	M	ID n+1
Identifier	CAN_FxR5	CAN_FxR6					ID n+1	ID n+2	ID n+1	ID n+2
Identifier/Mask	CAN_FxR7	CAN_FxR8					ID M	ID n+2	ID n+2	ID n+3
ID= Identifier n = Filter number M = Mask x = Filter bank number										
¹ The FMHx and FMLx bits are located in the CAN_FMR1 and CAN_FMR2 registers										

图136 16/8位过滤器组设置(CAN_FCRx寄存器的FSCx位为01b)

Filter registers							Filter mode ¹				
Mapping	STID[10:3] / EXID[28:21]	STID [2:0] / EXID[20:18]	RTR	IDE	EXID [17:15]		FMHx = 0 FMLx = 0	FMHx = 0 FMLx = 1	FMHx = 1 FMLx = 0	FMHx = 1 FMLx = 1	
Identifier	CAN_FxR1	CAN_FxR2					ID n	ID n	ID n	ID n	
Identifier/Mask	CAN_FxR3	CAN_FxR4					ID M	ID n+1	M	ID n+1	
Identifier	CAN_FxR5	CAN_FxR6					ID n+1	ID n+2	ID n+1	ID n+2	
Identifier/Mask	CAN_FxR7	CAN_FxR8					ID M	ID n+2	ID n+2	ID n+3	
ID= Identifier n = Filter number M = Mask x = Filter bank number											
¹ The FMHx and FMLx bits are located in the CAN_FMR1 and CAN_FMR2 registers											

图137 8位过滤器组设置(CAN_FCRx寄存器的FSCx位为00b)

		Filter registers		Filter mode ¹							
Mapping		STID[10:3] / EXID[28:21]		FMHx = 0 FMLx = 0		FMHx = 0 FMLx = 1		FMHx = 1 FMLx = 0		FMHx = 1 FMLx = 1	
Identifier	CAN_FXR1	ID M	n	ID	n	ID	n	ID	n	ID	n
	CAN_FXR2			ID	n+1	M		ID	n+1	ID	n+1
Identifier/Mask	CAN_FXR3	ID M	n+1	ID	n+2	ID	n+1	ID	n+2	ID	n+2
	CAN_FXR4			ID	n+3	M		ID	n+3	ID	n+3
Identifier	CAN_FXR5	ID M	n+2	ID	n+4	ID	n+2	ID	n+4	ID	n+4
	CAN_FXR6			M		ID	n+3	ID	n+5	ID	n+5
Identifier/Mask	CAN_FXR7	ID M	n+3	ID	n+5	ID	n+4	ID	n+6	ID	n+6
	CAN_FXR8			M		ID	n+5	ID	n+7	ID	n+7

ID = Identifier
M = Mask
n = Filter number
x = Filter bank number

¹ The FMHx and FMLx bits are located in the CAN_FMR1 and CAN_FMR2 registers

过滤器匹配序号

一旦收到的报文被存入FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到RAM中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。beCAN提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起被存入邮箱中。因此每个收到的报文都有与它相关联的过滤器匹配序号。

过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较
- 把过滤器匹配序号当作数组中的一个索引来访问目标所在地址

对于非屏蔽模式下的过滤器，软件不需要直接跟标识符进行比较。

对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位进行比较即可。

在给过滤器编号时，无需考虑过滤器组的激活状态。请参考的例子。

表57 过滤器编号的例子

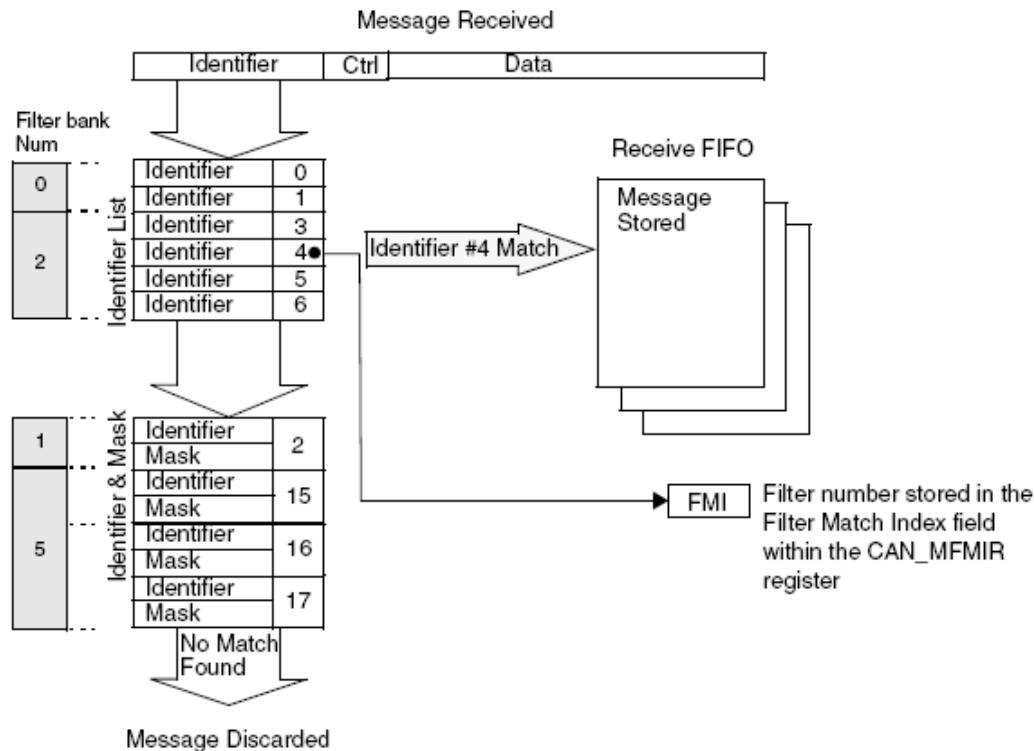
Filter bank						Filter number
Number	FCS	FMH	FML	FACT	Configuration	
0	0b11	1	1	1	Identifier list (32-bit)	0 1
1	0b11	0	0	1	Identifier mask (32-bit)	2
2	0b10	1	1	1	Identifier list (16-bit)	3 4 5 6
3	0b00	0	1	0	Deactivated Identifier List/Identifier mask (8-bit)	7 8 9 10 11 12
4	0b10	0	0	0	Deactivated Identifier Mask (16-bit)	13 14
5	0b01	0	0	1	Identifier Mask (16/8-bit)	15 16 17

过滤器优先级规则

根据过滤器的不同配置，有可能一个报文标识符能通过多个过滤器的过滤；在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据下列优先级规则来确定：

- 32位位宽的过滤器优先级高于16位位宽的过滤器，而16位位宽的过滤器优先级高于8位位宽的过滤器
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高

图138 过滤器机制的例子



上面的例子说明了beCAN的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的FIFO中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4标识符匹配，因此报文内容和FMI4被存入FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。

如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

23.6.4 报文存储

邮箱是软件和硬件之间关于报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好(然后再发出发送的请求)。发送的状态可通过查询CAN_MCSR寄存器获知。

表58 发送邮箱寄存器列表

相对发送邮箱地址的偏移量(字节)	寄存器名
0	CAN_MCSR
1	CAN_MDLCR
2	CAN_MIDR1
3	CAN_MIDR2
4	CAN_MIDR3
5	CAN_MIDR4
6	CAN_MDAR1
7	CAN_MDAR2
8	CAN_MDAR3
9	CAN_MDAR4

10	CAN_MDAR5
11	CAN_MDAR6
12	CAN_MDAR7
13	CAN_MDAR8
14	CAN_MTSRL
15	CAN_MTSRH

接收邮箱

在接收到一个报文后，软件就可以访问接收FIFO的输出邮箱来读取它。一旦软件处理了报文(如将它读出来)，软件就应该对CAN_RFR寄存器的RFOM位进行置'1'，来释放该FIFO输出邮箱，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在CAN_MFMIR寄存器中。16位的时间戳存放在CAN_MTSRH寄存器和CAN_MTSR寄存器中。

表59 接收邮箱寄存器列表

相对接收邮箱基地址的偏移量(字节)	寄存器名
0	CAN_MFMIR
1	CAN_MDLCR
2	CAN_MIDR1
3	CAN_MIDR2
4	CAN_MIDR3
5	CAN_MIDR4
6	CAN_MDAR1
7	CAN_MDAR2
8	CAN_MDAR3
9	CAN_MDAR4
10	CAN_MDAR5
11	CAN_MDAR6
12	CAN_MDAR7
13	CAN_MDAR8
14	CAN_MTSRL
15	CAN_MTSRH

23.6.5 出错管理

CAN协议描述的出错管理，完全由硬件通过发送错误计数器(CAN_TESR寄存器)和接收错误计数器(CAN_RECER寄存器)来处理，其值根据错误的情况而增加或减少。关于TEC和REC管理的详细信息，请参考CAN标准。

软件可以读出它们的值来判断CAN网络的稳定性。

此外，CAN的硬件将当前错误状态的详细信息存放在CAN_ESR寄存器中。通过设置CAN_EIER寄存器和CAN_IER寄存器中ERRIE位，软件可以灵活地控制检测到的出错中断的产生。

离线恢复

当TEC的值大于255时，beCAN就进入离线状态，CAN_ESR寄存器的BOFF位被置'1'。在离线状态下，beCAN不再接收和发送报文。

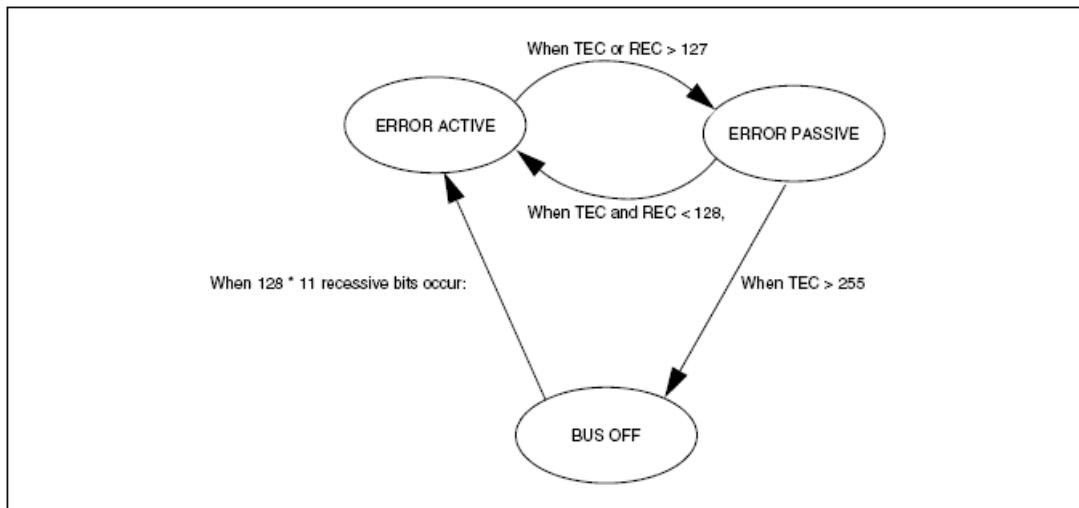
根据CAN_MCR寄存器的ABOM位的设置，beCAN可以自动或在软件的请求下，从离线状态恢复(变为错误主动状态)。在这两种情况下，beCAN都必须等待一个CAN标准所描述的恢复过程(CAN RX引脚上检测到128次11个连续的隐性位)。

如果ABOM位为'1'，beCAN进入离线状态后，就自动开启恢复过程。

如果ABOM位为'0'，软件必须通过请求beCAN进入初始化模式进行初始化恢复序列操作，然后当beCAN退出初始化模式时，beCAN启动恢复过程。

注：在初始化模式下，beCAN不会监视CAN RX引脚的状态，这样就不能完成恢复过程。为了完成恢复过程，**beCAN必须工作在正常模式**。

图139 CAN错误状态图



23.6.6 位时序

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。

可以将位时间分为如下3段简单描述位时间的操作方式：

- **同步段(SYNC_SEG)**: 通常一个预期的位变化发生在该时间段内。其时间长度固定为1个时间单元($1 \times t_{CAN}$)。
- **时间段1(BS1)**: 定义采样点的位置。它包含CAN标准里的PROP_SEG和PHASE_SEG1。其持续时间可以编程为1到16个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- **时间段2(BS2)**: 定义发送点的位置。它代表CAN标准里的PHASE_SEG2。其持续时间可以编程为1到8个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度(SJW)定义了，在每位段中可以延长或缩短多少个时间单元的上限。其持续时间可以编程为1到4个时间单元。

注：更多的关于CAN位时间特性和重新同步机制的详细信息，请参考ISO11898标准。

为了避免软件的编程错误，对位时间特性寄存器CAN_BTR1和CAN_BTR2的设置，只能在beCAN处于初始化状态下进行。

图140 位时序

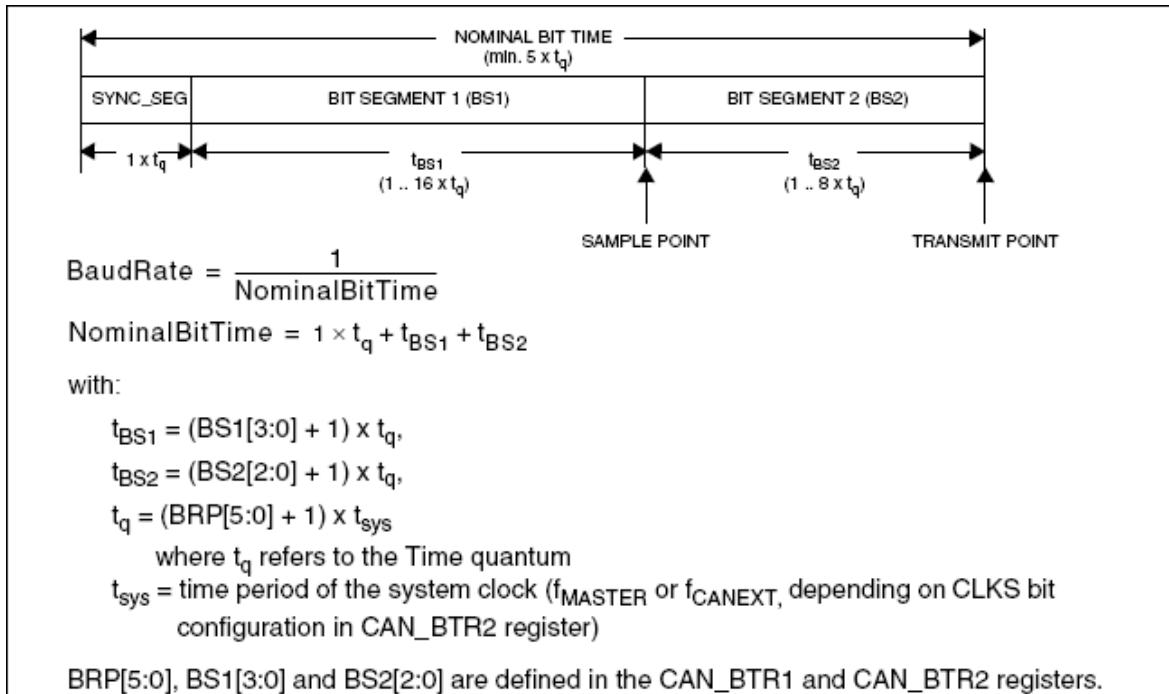
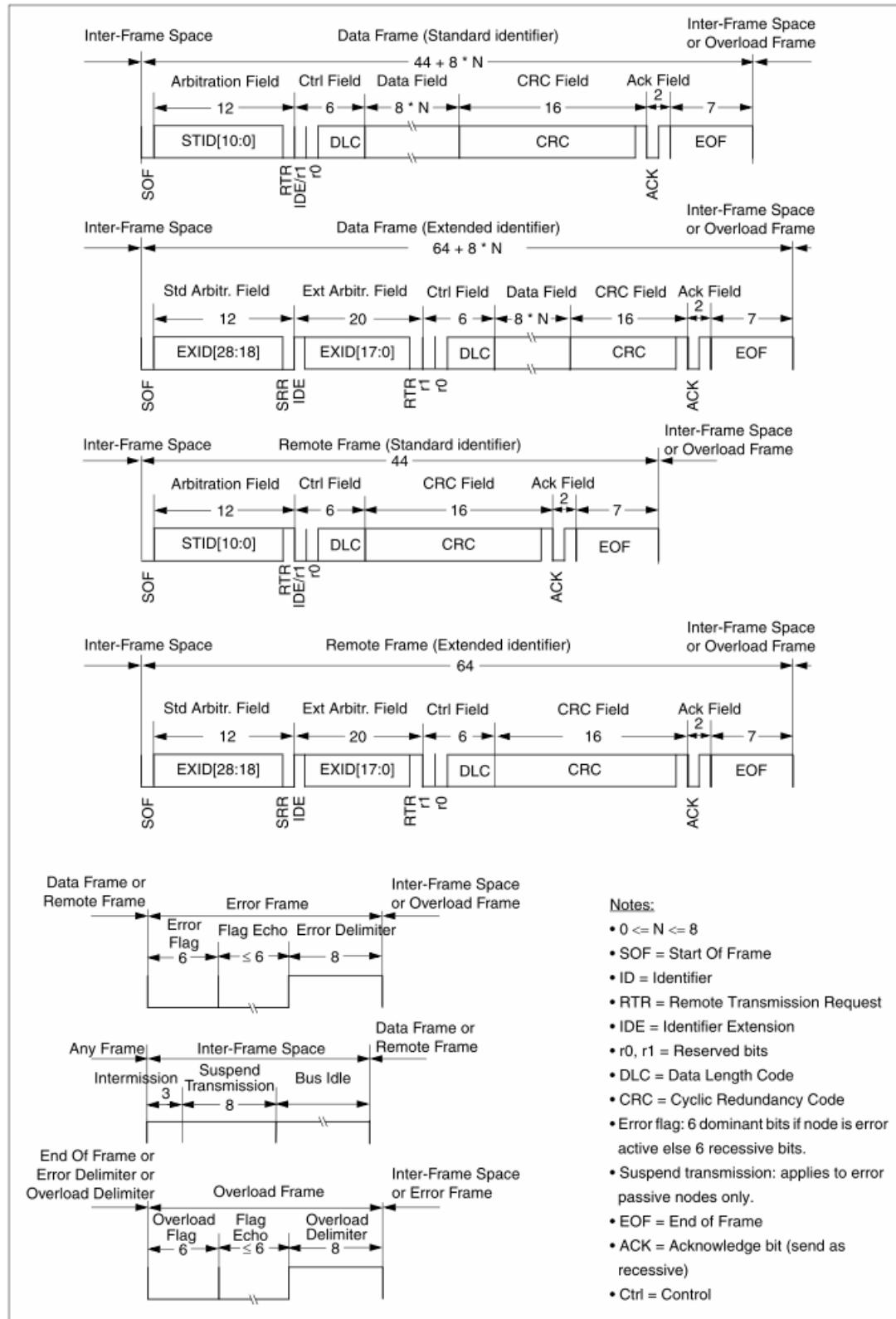


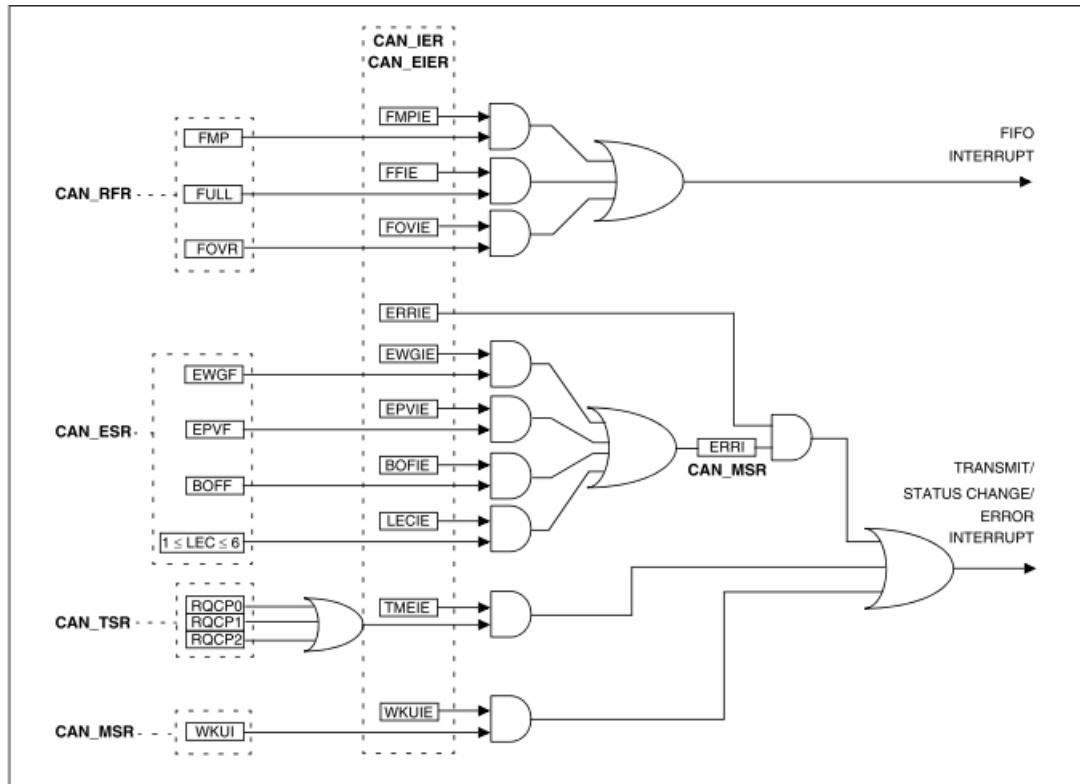
图141 CAN帧的示例



23.7 中断

beCAN有2个专用中断向量。每个中断源都可以单独通过设置CAN中断使能寄存器(CAN_IER)和CAN出错中断使能寄存器来使能和禁用。

图142 事件标志和中断产生



● FIFO中断可由下列事件产生:

- FIFO接收到一个新报文, CAN_RFR寄存器的FMP位自动加1。
- FIFO变为满的状况, CAN_RFR寄存器的FULL位被置'1'。
- FIFO发生溢出的状况, CAN_RFR寄存器的FOVR位被置'1'。

● 发送、错误和状态变化中断可由下列事件产生:

- 发送邮箱0变为空, CAN_TSR寄存器的RQCP0位被置'1'。
- 发送邮箱1变为空, CAN_TSR寄存器的RQCP1位被置'1'。
- 发送邮箱2变为空, CAN_TSR寄存器的RQCP2位被置'1'。
- 出错情况, 关于出错情况的详细信息请参考CAN错误状态寄存器(CAN_ESR)。
- 唤醒情况, 在CAN接收引脚上监视到帧起始位(SOF)。

23.8 寄存器访问保护

对某些寄存器的错误访问会导致一个CAN节点对整个CAN网络的暂时性干扰。因此,以下所列的寄存器只能在CAN处于初始化模式时修改:

CAN_BTR1、CAN_BTR2、CAN_FCR1、CAN_FCR2、CAN_FMR1、CAN_FMR2和CAN_DGR寄存器。

虽然错误数据的发送对CAN的网络层不会带来问题,但却会对应用程序造成严重影响。因此,软件只能在发送邮箱为空的状态改变它,请参见图132。

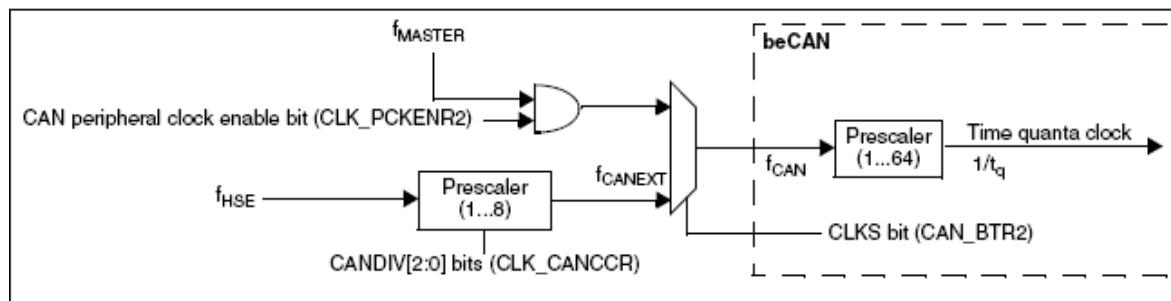
在修改过滤器的寄存器值之前必须关闭对应的过滤器组。软件对过滤器配置(位宽或模式)的修改只有在初始化模式下才可进行。

23.9 时钟系统

根据CAN协议,时钟误差极限为:当速度最大为125Kbps时,误差为1.58%。对于更高的波特率,协议建议使用晶振。为使beCAN可使用全波特率的范围,提供了一个接口用于切换beCAN使用2个不同的时钟区域: f_{MASTER} 或精确的外部时钟(HSE)。即使CAN网络通讯节点使用波特率

的时钟是由外部时钟产生的, beCAN和CPU之间的接口仍以CPU的速度运行, 详细描述请参考CAN_BTR2寄存器的CLKS位。

图143 时钟接口



外部时钟 f_{CANEXT} 的频率必须小于CPU的时钟频率(f_{CPU}).

有两种方法可以配置beCAN的时钟:

- 选择 f_{MASTER} 作为 CAN 的时钟。在这种情况下, 在低功耗模式期间时钟可以在外设级别(外设时钟门控寄存器 CLK_PCKENR2)被停止。
显然, 对于高速 CAN 应用, f_{MASTER} 必须由外部晶振提供。
- 或选择 f_{CANEXT} 的(设置 CLKS 位)作为 CAN 的时钟。在这种情况下, 通过外设时钟门控寄存器不能将时钟停止。

注: 如果在CLK控制器(参考时钟安全系统寄存器中的CSSEN位的描述)中开启了时钟安全系统, 有一种方法可以在主时钟失效时, 自动将CAN网络进入隐性状态, 以保证CAN网络的通讯不会被设备破坏。不管使用何种方法保证, 在使用beCAN前, PG0 I/O引脚必须设置为pull-up模式。使用这种方法, 当错误发生时I/O的可变功能被禁止掉, 该引脚处于内部上拉状态而不是悬浮状态。

23.10 beCAN低功耗模式

表60 beCAN在低功耗模式下的状态:

模式	描述
等待(WAIT)	除了访问发送/接收邮箱和过滤器的值被禁止, 其他对beCAN无影响 (CPU时钟停止)。 beCAN的中断会将设备从等待(WAIT)模式退出。
慢速(SLOW)	对beCAN无影响。 外部时钟的频率(如果选择了)必须小于 f_{CPU} 。参见CAN位时序寄存器2中的CLKS位
停机(HALT)/活跃停机(Active HALT)	beCAN停止。 beCAN的接收中断可将设备从停机(HALT)/活跃停机(Active HALT)模式退出。 实际上, 任意一个出现在接收引脚上的下降沿会将CPU唤醒。

注: 如果CAN帧是在WAIT,HALT 或者 Active HALT 模式时收到的, MCU会被唤醒但是这一帧数据会被丢失。

23.11 CAN 寄存器描述

23.11.1 CAN主控制寄存器 (CAN_MCR)

地址偏移值: 0x00

复位值: 0x02

7	6	5	4	3	2	1	0
TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rw	rw	rw	rw	rw	rw	rw	rw
位7	TTCM: 时间触发通信模式 0: 禁止时间触发通信模式; 1: 允许时间触发通信模式。 注: 要想了解关于时间触发通信模式的更多信息, 请参考 23.4.4时间触发通讯模式 。						
位6	ABOM: 自动离线(Bus-Off)管理 该位决定CAN硬件在什么条件下可以退出离线状态。 0: 通过软件请求使CAN退出离线状态。请参考 23.6.5出错管理 离线恢复。 1: 一旦硬件检测到128次11位连续的隐性位, 自动退出离线状态。 注: 要想了解关于离线状态的更多信息, 请参考 23.6.5出错管理 。						
位5	AWUM: 自动唤醒模式 该位决定CAN处在睡眠模式时由硬件还是软件唤醒CAN: 0: 睡眠模式通过清除CAN_MCR寄存器的SLEEP位, 由软件唤醒; 1: 睡眠模式通过检测CAN报文, 由硬件自动唤醒。唤醒的同时, 硬件自动对CAN_MSR寄存器的SLEEP和SLAK位清0。						
位4	NART: 禁止报文自动重传 0: 按照CAN标准, CAN硬件在发送报文失败时会一直自动重发, 直到发送成功; 1: 不管发送的结果如何(成功、出错或仲裁丢失), CAN报文只被发送1次。						
位3	RFLM: 接收FIFO锁定模式 0: 在接收溢出时FIFO未被锁定, 当接收FIFO处于满状态, 下一个收到的报文会覆盖先前的报文; 1: 在接收溢出时FIFO被锁定, 当接收FIFO处于满状态, 下一个收到的报文会被丢弃。						
位2	TXFP: 发送FIFO优先级 当有多个报文同时在等待发送时, 该位决定这些报文的发送顺序 0: 优先级由报文的标识符来决定; 1: 优先级由发送请求的顺序来决定(按时间发生顺序)。						
位1	SLEEP: 睡眠模式请求 软件对该位置1可以请求CAN进入睡眠模式, 一旦当前CAN的任务(发送或接收报文)结束, CAN就立即进入睡眠。 如果软件对该位清0, 将使CAN退出睡眠模式。 当AWUM位被置位且在CAN Rx信号中检测出SOF位时, 硬件对该位清0。						
位0	INRQ: 初始化请求 软件对该位清0可使CAN从初始化模式进入正常工作模式: 当CAN在接收引脚检测到连续的11个隐性位后, CAN就达到同步, 并准备好接收和发送数据。同时硬件相应地将CAN_MSR寄存器的INAK位清0。 软件对该位置1可使CAN进入初始化模式: 一旦软件将该位置位, CAN硬件等待当前任务(发送或接收)结束, 再进入初始化模式。相应地, 硬件将CAN_MSR寄存器的INAK位置1。						

23.11.2 CAN主状态寄存器 (CAN_MSR)

地址偏移值: 0x01

复位值: 0x02

7	6	5	4	3	2	1	0
保留	RX	TX	WKUI	ERRI	SLEEP	INAK	
	r	r	rc_w1	rc_w1	r	r	
位7:6	保留位, 读出值为0						
位5	RX: 接收标志 1: 表示CAN引脚当前正在接收。						
位4	TX: 发送标志 1: 表示CAN引脚当前正在发送。						
位3	WKUI: 唤醒中断标志 当CAN处于睡眠状态时, 一旦帧起始位(SOF)被检测到, 硬件就对该位置1; 并且如果CAN_IER寄存器的WKUIE位为1, 则相应的中断被触发。 软件对该位由写1将清除该位。						
位2	ERRI: 出错中断标志 当检测到错误, CAN_ESR寄存器的某个位被置1, 且CAN_EIER寄存器的相应中断位被使能时, 硬件对该位置位; 如果CAN_IER寄存器的ERRIE位被置位, 则错误中断被触发。 软件对该位由写1, 将清除该位。						
位1	SLAK: 睡眠模式确认标志 该位由硬件置1, 表示CAN当前处于睡眠模式, 供软件进行状态查询。该位是对软件请求进入睡眠模式的确认(将CAN_MCR寄存器的SLEEP位置1)。 当CAN退出睡眠模式时硬件对该位清0。 如果将CAN_MCR的SLEEP位清0, CAN退出睡眠模式。 有关清除SLEEP位的详细信息, 参见CAN_MCR寄存器的AWUM位的描述。						
位0	INAK: 初始化确认标志 该位由硬件置1, 表示CAN当前处于初始化模式, 供软件进行状态查询。该位是对软件请求进入初始化模式的确认(对CAN_MCR寄存器的INRQ位置1)。 当CAN退出初始化模式时硬件对该位清0(需要跟CAN总线同步)。这里跟CAN总线同步是指, 硬件需要在CAN的RX引脚上检测到连续的11位隐性位。						

23.11.3 CAN发送状态寄存器 (CAN_TSR)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TXOK2	TXOK1	TXCK0	保留	RQCP2	RQCP1	RQCP0
	r	r	r		r	rc_w1	rc_w1
位7	保留位, 读出值为0						
位6	TXOK2: 邮箱2发送成功 当邮箱2的发送请求被成功完成后, 硬件对该位置1。请参见 图132 。 每次在邮箱2进行发送请求, 或者软件清除RQCP2位时, 硬件清除该位。						
位5	TXOK1: 邮箱1发送成功 当邮箱1的发送请求被成功完成后, 硬件对该位置1。请参见 图132 。 每次在邮箱1进行发送请求, 或者软件清除RQCP1位时, 硬件清除该位。						
位4	TXOK0: 邮箱0发送成功 当邮箱0的发送请求被成功完成后, 硬件对该位置1。请参见 图132 。 每次在邮箱2进行发送请求, 或者软件清除RQCP0位时, 硬件清除该位。						
位3	保留位, 读出值为0						
位2	RQCP2: 邮箱2请求完成 当上次对邮箱2的请求(发送或中止)完成后, 硬件对该位置1。 软件对该位写'1'可以将其清0。						
位1	RQCP1: 邮箱1请求完成 当上次对邮箱2的请求(发送或中止)完成后, 硬件对该位置1。 软件对该位写'1'可以将其清0。						
位0	RQCP0: 邮箱0请求完成 当上次对邮箱2的请求(发送或中止)完成后, 硬件对该位置1。 软件对该位写'1'可以将其清0。						

23.11.4 CAN发送优先级寄存器 (CAN_TPR)

地址偏移值: 0x03

复位值: 0x1C

7	6	5	4	3	2	1	0
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE1	CODE0
r	r	r	r	r	r	r	r
位7	LOW2: 邮箱2最低优先级标志 当有多个邮箱在等待发送报文, 且邮箱2的优先级最低时, 硬件对该位置1。 注: 当只有一个邮箱等待发送时, 该位被置0。						
位6	LOW1: 邮箱1最低优先级标志 当有多个邮箱在等待发送报文, 且邮箱1的优先级最低时, 硬件对该位置1。 注: 当只有一个邮箱等待发送时, 该位被置0。						
位5	LOW0: 邮箱0最低优先级标志 当有多个邮箱在等待发送报文, 且邮箱0的优先级最低时, 硬件对该位置1。 注: 当只有一个邮箱等待发送时, 该位被置0。						
位4	TME2: 发送邮箱2空 当邮箱2中没有等待发送的报文时, 硬件对该位置1。 注: 当处于与ST7的beCAN兼容的模式(CAN_DGR寄存器的TXM2E位为0)时, 该位被保留并硬件将该位强制置0。						
位3	TME1: 发送邮箱1空 当邮箱1中没有等待发送的报文时, 硬件对该位置1。						
位2	TME0: 发送邮箱0空 当邮箱0中没有等待发送的报文时, 硬件对该位置1。						
位1:0	CODE[1:0]: 邮箱号 当有至少1个发送邮箱为空时, 邮箱号为下一个空的发送邮箱号。 当所有的发送邮箱都为空时, 邮箱号为优先级最低的那个发送邮箱号。 注: 当处于与ST7的beCAN兼容的模式(CAN_DGR寄存器中的TXM2E0位为0)时, CODE1始终为0。						

23.11.5 CAN接收FIFO 1 寄存器(CAN_RFR)

地址偏移值: 0x04

复位值: 0x00

7	6	5	4	3	2	1	0
保留		RFOM	FOVR	FULL	保留	FMP[1:0]	
		rs	rc_w1	rc_w1		r	r
位7:6	保留位, 读出值为0						
位5	RFOM: 释放接收FIFO输出邮箱 软件通过将该位置1来释放接收FIFO的输出邮箱。只有当FIFO中至少有一个报文等待读取时, 对该位置1才有能释放输出邮箱。 如果接收FIFO为空, 那么对该位置1没有任何效果。如果FIFO中有1个以上的报文, 软件必须释放输出邮箱, 才能访问下一个报文。 当输出邮箱被释放时, 硬件对该位清0。						
位4	FOVR1: FIFO溢出 当FIFO已满, 又收到新的报文且报文符合过滤条件, 硬件对该位置1。 软件对该位写'1'可以将其清0。						
位3	FULL1: FIFO满 当有3个报文被存入FIFO 1时, 硬件对该位置1。 软件对该位写'1'、或通过RFOM释放FIFO, 可以将其清0。						
位2	保留位, 读出值为0						
位1:0	FMP[1:0]: FIFO内接收到报文数目 这2位反映了当前接收FIFO中存放的报文数目。 每次当硬件将1个新的报文被存入接收FIFO时, FMP加1; 每次当(软件预先设定后RFOM位被清除)硬件释放输出邮箱时, FMP减1。						

23.11.6 CAN中断允许寄存器 (CAN_IER)

地址偏移值: 0x05

复位值: 0x00

7	6	5	4	3	2	1	0
WKUIE	保留		FOVIE	FFIE	FMPIE	TMEIE	
r			r	r	r	r	
位7	WKUIE: 睡眠唤醒中断允许位 0: 当WKUI位被置1时, 没有中断产生; 1: 当WKUI位被置1时, 产生中断。						
位6:4	保留位, 读出值为0。						
位3	FOVIE: FIFO溢出中断允许 0: 当FIFO的FOVR位被置1时, 没有中断产生; 1: 当FIFO的FOVR位被置1时, 产生中断。						
位2	FFIE: FIFO满中断允许 0: 当FIFO的FULL位被置1时, 没有中断产生; 1: 当FIFO的FULL位被置1时, 产生中断。						
位1	FMPIE: FIFO消息挂号中断允许 0: 当FIFO的FMP[1:0]位由00b转变为01b时, 没有中断产生; 1: 当FIFO的FMP[1:0]位由00b转变为01b时, 产生中断。						
位0	TMEIE: 发送邮箱中断允许 0: 当RQCPx位被置1时, 没有中断产生; 1: 当RQCPx位被置1时, 产生中断。						

23.11.7 CAN诊断寄存器 (CAN_DGR)

地址偏移值: 0x06

复位值: 0x0C

7	6	5	4	3	2	1	0
保留			TXM2E	RX	SAMP	SILM	LBKM
位7:5		保留位, 读出值为0。					
位4		TXM2E: 发送邮箱2使能位 0: 强制beCAN与ST7的beCAN(2个发送邮箱)兼容 – 复位状态。 1: 使能第3个发送邮箱(邮箱号为2)					
位3		RX: CAN接收电平 该位反映CAN接收引脚(CAN_RX)的实际电平。					
位2		SAMP: 上次采样值 CAN接收引脚的上次采样值。					
位1		SILM: 静默模式 0: 正常状态; 1: 静默模式。					
位0		LBKM: 环回模式 0: 禁止环回模式; 1: 允许环回模式。					

23.11.8 CAN页面选择寄存器 (CAN_PSR)

地址偏移值: 0x07

复位值: 0x00

7	6	5	4	3	2	1	0
保留						PS[2:0]	
						RW	RW
位7:3						保留位, 读出值为0。	
位2:0						PS[2:0]: 页面选择 用于选择寄存器的页面: 000: 发送邮箱0; 001: 发送邮箱1; 010: 接收过滤器0:1; 011: 接收过滤器2:3; 100: 接收过滤器4:5; 101: 发送邮箱2; 110: 设置/诊断; 111: 接收FIFO; 详细细节参见 图145	

23.11.9 CAN错误状态寄存器 (CAN_ESR)

地址偏移值: 见[表63](#)

复位值: 0x00

7	6	5	4	3	2	1	0
保留		LEC[2:0]		保留	BOFF	EPVF	EWGF
	rw	rw	rw		r	r	r
位7	保留位, 读出值为0。						
位6:4	<p>LEC[2:0]: 上次错误代码 在检测到CAN总线上发生错误时, 硬件根据出错情况设置其为1~6的值。当报文被正确发送或接收后, 硬件清除其值为'0'。硬件没有使用错误代码7, 软件可以设置该值, 从而可以检测代码的更新。</p> <p>000: 没有错误; 001: 位填充错; 010: 格式(Form)错; 011: 确认(ACK)错; 100: 隐性位错; 101: 显性位错; 110: CRC出错; 111: 由软件设置。</p>						
位3	保留位, 读出值为0。						
位2	<p>BOFF: 离线(Bus Off)标志 当进入离线状态时, 硬件将该位置1。当发送错误计数器TEC溢出, 即大于255时, CAN进入离线状态。请参考23.6.5。</p>						
位1	<p>EPVF: 被动错误标志 当出错次数达到被动错误的阈值时, 硬件将该位置1。 (接收错误计数器或发送错误计数器的值>127)。</p>						
位0	<p>EWGF: 错误警告标志 当出错次数达到警告的阈值时, 硬件将该位置1。 (接收错误计数器或发送错误计数器的值≥96)。</p>						

23.11.10 CAN出错中断使能寄存器 (CAN_EIER)

地址偏移值: 见 [表63](#)

复位值: 0x00

7	6	5	4	3	2	1	0
ERRIE	保留	保留	LECIE	保留	BOFIE	EPVIE	EWGIE
RW			RW		RW	RW	RW
位7	ERRIE: 出错中断使能 0: 当在CAN_ESR(CAN_MSR寄存器中的ERRI位置1)有一个错误中断请求时, 不产生中断; 1: 当在CAN_ESR(CAN_MSR寄存器中的ERRI位置1)有一个错误中断请求时, 产生中断。 请参考 图142						
位6:5	保留位, 读出值为0。						
位4	LECIE: 上次错误代码中断使能 0: 当检测到错误时, LEC[2:0]中的错误代码被硬件置位, EERI不会被置位; 1: 当检测到错误时, LEC[2:0]中的错误代码被硬件置位, EERI将被置位。						
位3	保留位, 读出值为0。						
位2	BOFIE: 离线中断使能 0: 当BOFF被置位, EERI不会被置位; 1: 当BOFF被置位, EERI将被置位。						
位1	EPVIE: 错误被动中断使能 0: 当EPVF被置位, EERI不会被置位; 1: 当EPVF被置位, EERI将被置位。						
位0	EWGIE: 错误警告中断使能 0: 当EWGF被置位, EERI不会被置位; 1: 当EWGF被置位, EERI将被置位。						

23.11.11 CAN发出出错计数器寄存器 (CAN_TECR)

地址偏移值: 见 [表63](#)

复位值: 0x00

7	6	5	4	3	2	1	0
TEC[7:0]							
r	r	r	r	r	r	r	r
位7:0	TEC[7:0]: 发送出错计数器 在发送期间由于出错, 根据CAN的标准中定义的出错条件, 该计数器加8。在每次发送成功后, 计数器减1, 或者如果CAN控制器退出离线状态时, 复位到0。当计数器值超过127时, CAN控制器进入出错被动状态(error passive state)。当计数器的值超过255时, CAN的控制器进入离线状态。						

23.11.12 CAN接收出错计数器寄存器 (CAN_RECRR)

地址偏值: 见表63

复位值: 0x00

7	6	5	4	3	2	1	0
REC[7:0]							
r	r	r	r	r	r	r	r
位7:0	REC[7:0]: 接收出错计数器 接收出错计数是CAN协议中失败限制机制操作的一部分。在接收期间由于出错，该计数器加1，或根据CAN的标准中定义的出错条件加8。在每次成功接收后，计数器减1，或者如果该值大于128时复位到120。当计数器的值超过127时，CAN的控制器进入出错被动状态。						

23.11.13 CAN位时间特性寄存器 (CAN_BTR1)

地址偏移: 值: 见[表63](#)

复位值: 0x40

7	6	5	4	3	2	1	0
SJW[1:0]		BRP[5:0]					
RW	RW	RW	RW	RW	RW	RW	RW

本寄存器只能在CAN硬件为初始化模式下由软件访问。

位7:6	SJW[1:0]: 重新同步跳跃宽度 为了重新同步, 该位域定义了CAN硬件在每位中可以延长或缩短多少个时间单元的上限。 跳跃宽度 = (SJW[1:0] + 1)。
位5:0	BRP[5:0]: 波特率分频器 该位域定义了时间单元(t_q)的时间长度 $t_q = (BRP[5:0]+1) / f_{CAN}$ 不管 $f_{CAN}=f_{CANEXT}$ 或 f_{MASTER} (参考CAN_BTR2寄存器中的CLK位) 更多关于位时间的信息, 请参考 23.6.6节 位

23.11.14 CAN位时间特性寄存器 (CAN_BTR2)

地址偏移值: 见[表63](#)

复位值: 0x23

7	6	5	4	3	2	1	0
CLKS	BS[2:0]						BS1[3:0]
RW	RW	RW	RW	RW	RW	RW	RW

本寄存器只能在CAN硬件为初始化模式下由软件访问。

位7	CLKS: 时钟输入选择位 0: 选择CPU时钟($f_{CAN} = f_{MASTER}$); 1: 选择外部时钟($f_{CAN} = f_{CANEXT}$).
位6:4	BS2[2:0]: 时间段2 该位域定义了时间段2占用了多少个时间单元 时间段2 = BS2[2:0] + 1
位3:0	BS1[3:0]: 时间段1 该位域定义了时间段1占用了多少个时间单元 时间段1 = BS1[3:0] + 1 关于位时间特性的详细信息, 请参考 23.6.6节 位 。

23.11.15 邮箱寄存器

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息,请参考[23.6.4报文存储](#)。

除了下述例外,发送和接收邮箱几乎一样:

- 发送邮箱中的CAN_MCSR寄存器被接收邮箱中的CAN_MFMIR寄存器代替;
- 接收邮箱是只读的;
- 发送邮箱只有在它为空时才是可写的(CAN_TPR寄存器的相应TME位为'1')。

CAN报文控制/状态寄存器 (CAN_MCSR)

地址偏移值: 见[表58](#)和[表59](#)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	TERR	ALST	TXOK	RQCP	ABRQ	TXRQ	
	r	r	r	rc_w1	rs	rs	

注意: 本寄存器只有发送邮箱有。在接收邮箱中,在这个位置是CAN_MFMIR寄存器。

位7:6	保留位, 读出值为0。
位5	TERR: 发送出错标志 每次尝试发送后, 该位由硬件更新。 0: 先前的发送成功; 1: 先前的发送由于一个错误导致失败。
位4	ALST: 仲裁丢失 每次尝试发送后, 该位由硬件更新。 0: 先前的发送成功; 1: 先前的发送由于仲裁丢失导致失败。
位3	TXOK: 发送成功 每次尝试发送后, 该位由硬件更新。 0: 先前的发送失败; 1: 先前的发送成功。 注: 该位与CAN_TSR寄存器中相应的TXOKx位的值相同。
位2	RQCP: 请求完成 当最后一次请求(发送或中止)被执行后, 该位由硬件置位。 该位由软件写1清除, 或当有发送请求时由硬件清除。 注: 该位与CAN_TSR寄存器中相应的RQCPx位的值相同。 清除该位将清除CAN_MCSR寄存器中所有的状态位(TXOKALST和TERIR)和CAN_TSR寄存器中相应的RQCPx位和TXOKx位。
位1	ABRQ: 中止对邮箱的请求 通过软件将该位置位, 将中止相应邮箱的发送请求。 当邮箱为空时, 该位由硬件清除。 当邮箱不处于等待发送状态时, 将该位置位没有任何影响。
位0	TXRQ: 发送邮箱请求 软件对该位置位可以请求相应的邮箱进行发送。 当邮箱为空时, 硬件将清除该位。

CAN邮箱过滤匹配索引寄存器 (CAN_MFMIR)

地址偏移值: 见[表58](#)和[表59](#)

复位值: 未定义位

7	6	5	4	3	2	1	0
FMI[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

注意：本寄存器只有接收邮箱有。在发送邮箱中，在这个位置是CAN_MCSR寄存器。

位7:0	FMI[7:0]: 过滤器匹配序号 这里是存在邮箱中的信息传送的过滤器序号。关于标识符过滤的细节，请参考 23.6.3标识符过滤 中有关过滤器匹配序号。
------	--

CAN邮箱标识符寄存器 1(CAN_MIDR1)

地址偏移值：见[表58](#)和[表59](#)

复位值：未定义

7	6	5	4	3	2	1	0
保留	IDE	RTR		STID[10:6] / EXID[28:24]			
rw	rw	rw	rw	rw	rw	rw	rw
位7	保留位，读出值为0。						
位6	IDE: 扩展标识符 该位决定发送邮箱中报文使用的标识符类型 0：使用标准标识符； 1：使用扩展标识符。						
位5	RTR: 远程发送请求 0：数据帧； 1：远程帧。						
位4:0	STID[10:6]: 标准标识符 标识符的标准部分的高5位。 或 EXID[28:24]: 扩展标识符 扩展标识符的“基本”部分的高5位。						

CAN邮箱标识符寄存器 2(CAN_MIDR2)

地址偏移值：见[表58](#)和[表59](#)

复位值：未定义

7	6	5	4	3	2	1	0
STID[5:0] / EXID[23:18]							EXID[17:16]
rw	rw	wr	rw	rw	rw	rw	rw
位7:2	STID[5:0]: 标准标识符 标识符的标准部分的低6字节。 或 EXID[23:18]: 扩展标识符 扩展标识符的“基本”部分的低6位。						
位1:0	EXID[17:16]: 扩展标识符 扩展标识符的“扩展”部分的高2位。						

CAN邮箱标识符寄存器 3(CAN_MIDR3)

地址偏移值：见[表58](#)和[表59](#)

复位值: 未定义

7	6	5	4	3	2	1	0
EXID[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw
位7:0	EXID[15:8]: 扩展标识符 扩展标识符的“扩展”部分的位15~位8。						

CAN邮箱标识符寄存器 4(CAN_MIDR4)地址偏移值: 见 [表58](#)和 [表59](#)

复位值: 未定义

7	6	5	4	3	2	1	0
EXID[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw
位7:0	EXID[7:0]: 扩展标识符 扩展标识符的“扩展”部分的低8位。						

CAN邮箱数据长度控制寄存器 (CAN_MDLCR)地址偏移值: 见 [表58](#)和 [表59](#)

复位值: 未定义

7	6	5	4	3	2	1	0
TGT	保留			DLC[3:0]			
rw				rw	rw	rw	rw
位7	TGT: 发送时间戳 只有在CAN处于时间触发通信模式, 即CAN_MCR寄存器的TTCM位为1时, 该位才有效。 0: 时间戳CAN_MTSRH寄存器和CAN_MTSRL寄存器不发送; 1: 时间戳CAN_MTSRH寄存器和CAN_MTSRL寄存器将作为报文的最后2个字节发送出去。						
位6:4	保留位, 读出值为0。						
位3:0	DLC[3:0]: 发送数据长度码 该域指定了一个数据帧或者一个远程帧请求的报文的数据长度。						

CAN邮箱数据寄存器 x(CAN_MDARx) (x=1..8)地址偏移值: 见 [表58](#)和 [表59](#)

复位值: 未定义

7	6	5	4	3	2	1	0
DATA[7:0]							
rw	rw	wr	rw	rw	rw	rw	rw
位7:0	DATA[7:0]: 数据 报文的一个数据字节。 一个报文包含0到8个字节。 注: 当邮箱非空时, 该寄存器内的数据是写保护的。						

CAN邮箱时间戳低位寄存器 (CAN_MTSRL)地址偏移值: 见 [表58](#)和 [表59](#)

复位值: 未定义位



7	6	5	4	3	2	1	0
TIME[7:0]							
r	r	r	r	r	r	r	r
位7:0	TIME[7:0]: 报文时间戳的低位 该域包含了在检测到SOF的时刻时，捕捉的16位定时器的低8位字节。						

CAN邮箱时间戳高位寄存器 (CAN_MTSRH)

地址偏移值: 见[表58](#)和[表59](#)

复位值: 未定义

7	6	5	4	3	2	1	0
TIME[15:8]							
r	r	r	r	r	r	r	r
位7:0	TIME[15:8]: 报文时间戳的低位 该域包含了在检测到SOF的时刻时，捕捉的16位定时器的高8位字节。						

23.11.16 CAN过滤器寄存器

CAN 过滤器主控寄存器 (CAN_FMR1)

地址偏移值: 见[表63](#)

复位值: 0x00

7	6	5	4	3	2	1	0
FMH3	FML3	FMH2	FML2	FMH1	FML1	FMH0	FML0
rw	rw	rw	rw	rw	rw	rw	rw
位7	FMH3: 过滤器3模式的高位 过滤器3的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。						
位6	FML3: 过滤器3模式的低位 过滤器3的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。						
位5	FMH2: 过滤器2模式的高位 过滤器2的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。						
位4	FML2: 过滤器2模式的低位 过滤器2的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。						
位3	FMH1: 过滤器1模式的高位 过滤器1的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。						
位2	FML1: 过滤器1模式的低位 过滤器1的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。						

位1	<p>FMH0: 过滤器0模式的高位 过滤器0的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。</p>
位0	<p>FML0: 过滤器0模式的低位 过滤器0的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。</p>

CAN 过滤器模式寄存器 (CAN_FMR2)

地址偏移值：见表63

复位值: 0x00

7	6	5	4	3	2	1	0
	保留			FMH5	ML5	FMH4	FML4

位7:4	保留位, 读出为0
位3	FMH5: 过滤器5模式的高位 过滤器5的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。
位2	FML5: 过滤器5模式的低位 过滤器5的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。
位1	FMH4: 过滤器4模式的高位 过滤器4的标识符/屏蔽位的高位寄存器的模式: 0: 高位寄存器工作在屏蔽位模式; 1: 高位寄存器工作在标识符列表模式。
位0	FML4: 过滤器4模式的低位 过滤器4的标识符/屏蔽位的低位寄存器的模式: 0: 低位寄存器工作在屏蔽位模式; 1: 低位寄存器工作在标识符列表模式。

CAN 过滤器设置寄存器 (CAN_FCR1)

地址偏移量：见表6.3

复位值: 0x00

7	6	5	4	3	2	1	0
保留	FSC11	FSC10	FACT1	保留	FSC01	FSC00	FACT0

位7	保留位, 读出值为0。
位6:5	FSC1[1:0]: 过滤器1位宽设置 这两位定义了过滤器1的位宽。
位4	FACT1: 过滤器1激活 软件将该位置位来激活过滤器1。修改过滤器1的寄存器(CAN_F1Rx)时, 必须将该位清0。 0: 过滤器1被禁用; 1: 过滤器1被激活。
位3	保留位, 读出值为0。

位2:1	FSC1[1:0]: 过滤器0位宽设置 这两位定义了过滤器0的位宽。						
位0	FACT0: 过滤器0激活 软件将该位置位来激活过滤器0。修改过滤器0的寄存器(CAN_F0Rx)时，必须将该位清0。 0: 过滤器0被禁用； 1: 过滤器0被激活。						

CAN 过滤器设置寄存器 (CAN_FCR2)

地址偏移量: 见 [表63](#)

复位值: 0000 0000(00h)

7	6	5	4	3	2	1	0
保留	FSC31	FSC30	FACT3	保留	FSC21	FSC20	FACT2

rw rw rw rw rw rw rw

位7	保留位, 读出值为0。
位6:5	FSC3[1:0]: 过滤器3位宽设置 这两位定义了过滤器3的位宽。
位4	FACT3: 过滤器3激活 软件将该位置位来激活过滤器3。修改过滤器3的寄存器(CAN_F3Rx)时，必须将该位清0。 0: 过滤器3被禁用； 1: 过滤器3被激活。
位3	保留位, 读出值为0。
位2:1	FSC2[1:0]: 过滤器2位宽设置 这两位定义了过滤器2的位宽。
位0	FACT2: 过滤器2激活 软件将该位置位来激活过滤器2。修改过滤器2的寄存器(CAN_F2Rx)时，必须将该位清0。 0: 过滤器2被禁用； 1: 过滤器2被激活。

CAN 过滤器设置寄存器 (CAN_FCR3)

地址偏移值: 见 [表63](#)

复位值: 0x00

7	6	5	4	3	2	1	0
保留	FSC51	FSC50	FACT5	保留	FSC41	FSC40	FACT4

rw rw rw rw rw rw rw

位7	保留位, 读出值为0。
位6:5	FSC5[1:0]: 过滤器5位宽设置 这两位定义了过滤器5的位宽。
位4	FACT5: 过滤器5激活 软件将该位置位来激活过滤器5。修改过滤器5的寄存器(CAN_F5Rx)时，必须将该位清0。 0: 过滤器5被禁用； 1: 过滤器5被激活。
位3	保留位, 读出值为0。
位2:1	FSC4[1:0]: 过滤器4位宽设置 这两位定义了过滤器4的位宽。
位0	FACT4: 过滤器4激活 软件将该位置位来激活过滤器4。修改过滤器4的寄存器(CAN_F4Rx)时，必须将该位清0。

	0: 过滤器4被禁用; 1: 过滤器4被激活。
--	----------------------------

CAN 过滤器组x寄存器 (CAN_FiRx) (i=0..5, x=1..8)

地址偏移值: 见 [图145](#)

复位值: 未定义位

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

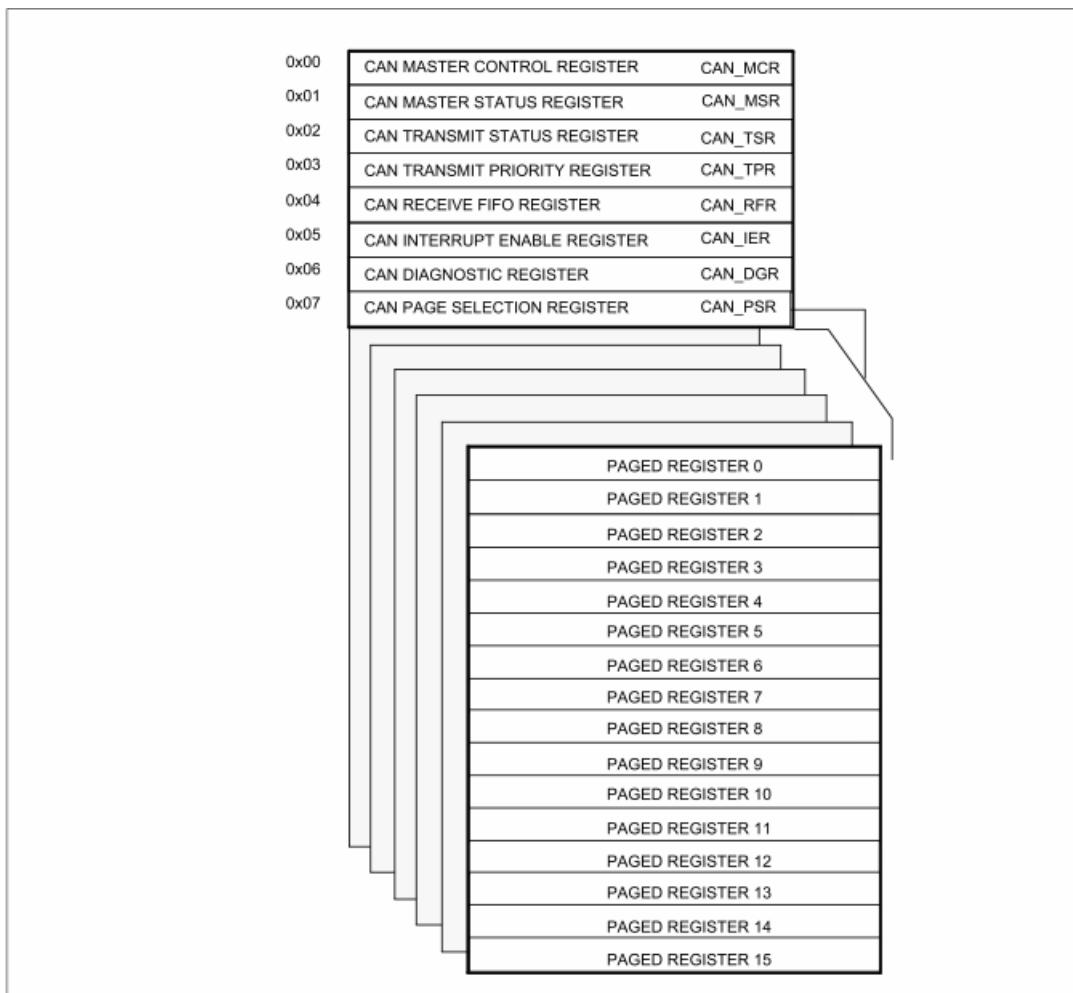
FB[7:0]

rw							
----	----	----	----	----	----	----	----

位7:0	FB[7:0] : 过滤器位 标识符模式 : 寄存器的每一位对应于所期望的标识符的相应的位的电平。 0: 期望的位为显性位; 1: 期望的位为隐性位。 屏蔽位模式 : 寄存器的每一位指示对应的标识符寄存器的位是否一定要与期望的标识符的相应的位一致。 0: 不关心, 该位不用于比较; 1: 必须匹配, 收到的标识符位必须与滤波器对应的标识符寄存器位相一致。 注: 每个过滤器i由8个寄存器组成: CAN_FiR1..8。根据位宽和模式的不同设置, 过滤器组中的每个寄存器的功能也不尽相同。关于过滤器的映射, 功能描述和屏蔽寄存器的关联, 请参见 23.6.3节 标识符过滤。 屏蔽位模式下的屏蔽/标识符寄存器, 与 标识符列表模式 下的寄存器位定义是相同。 注: 修改这些寄存器时, 相应的CAN_FCRx寄存器中FACT位必须清0。

23.12 beCAN寄存器列表

图144 CAN寄存器映射图



23.12.1 CAN的页映射

图145 CAN的页映射

	PAGE 0	PAGE 1	PAGE 2	PAGE 3	PAGE 4
0x00	CAN_MCSR	CAN_MCSR	CAN_F0R1	CAN_F2R1	CAN_F4R1
0x01	CAN_MDLCR	CAN_MDLCR	CAN_F0R2	CAN_F2R2	CAN_F4R2
0x02	CAN_MIDR1	CAN_MIDR1	CAN_F0R3	CAN_F2R3	CAN_F4R3
0x03	CAN_MIDR2	CAN_MIDR2	CAN_F0R4	CAN_F2R4	CAN_F4R4
0x04	CAN_MIDR3	CAN_MIDR3	CAN_F0R5	CAN_F2R5	CAN_F4R5
0x05	CAN_MIDR4	CAN_MIDR4	CAN_F0R6	CAN_F2R6	CAN_F4R6
0x06	CAN_MDAR1	CAN_MDAR1	CAN_F0R7	CAN_F2R7	CAN_F4R7
0x07	CAN_MDAR2	CAN_MDAR5	CAN_F0R8	CAN_F2R8	CAN_F4R8
0x08	CAN_MDAR3	CAN_MDAR6	CAN_F1R1	CAN_F3R1	CAN_F5R1
0x09	CAN_MDAR4	CAN_MDAR4	CAN_F1R2	CAN_F3R2	CAN_F5R2
0x0A	CAN_MDAR5	CAN_MDAR5	CAN_F1R3	CAN_F3R3	CAN_F5R3
0x0B	CAN_MDAR6	CAN_MDAR6	CAN_F1R4	CAN_F3R4	CAN_F5R4
0x0C	CAN_MDAR7	CAN_MDAR7	CAN_F1R5	CAN_F3R5	CAN_F5R5
0x0D	CAN_MDAR8	CAN_MDAR8	CAN_F1R6	CAN_F3R6	CAN_F5R6
0x0E	CAN_MTSRL	CAN_MTSRL	CAN_F1R7	CAN_F3R7	CAN_F5R7
0x0F	CAN_MTSRH	CAN_MTSRH	CAN_F1R8	CAN_F3R8	CAN_F5R8
	Tx Mailbox 0	Tx Mailbox 1	Acceptance Filter 0:1	Acceptance Filter 2:3	Acceptance Filter 4:5
	PAGE 5	PAGE 6	PAGE 7		
0x00	CAN_MCSR	CAN_ESR	CAN_MFMR		
0x01	CAN_MDLCR	CAN_EIER	CAN_MDLCR		
0x02	CAN_MIDR1	CAN_TECR	CAN_MIDR1		
0x03	CAN_MIDR2	CAN_RECRR	CAN_MIDR2		
0x04	CAN_MIDR3	CAN_BTR1	CAN_MIDR3		
0x05	CAN_MIDR4	CAN_BTR2	CAN_MIDR4		
0x06	CAN_MDAR1	Reserved	CAN_MDAR1		
0x07	CAN_MDAR2	Reserved	CAN_MDAR2		
0x08	CAN_MDAR3	CAN_FMR1	CAN_MDAR3		
0x09	CAN_MDAR4	CAN_FMR2	CAN_MDAR4		
0x0A	CAN_MDAR5	CAN_FCR1	CAN_MDAR5		
0x0B	CAN_MDAR6	CAN_FCR2	CAN_MDAR6		
0x0C	CAN_MDAR7	CAN_FCR3	CAN_MDAR7		
0x0D	CAN_MDAR8	Reserved	CAN_MDAR8		
0x0E	CAN_MTSRL	Reserved	CAN_MTSRL		
0x0F	CAN_MTSRH	Reserved	CAN_MTSRH		
	Tx Mailbox 2 (if TXM2E=1 in CAN_DGR register)	Configuration/Diagnostic	Receive FIFO		

表61 beCAN控制和状态页列表 - 寄存器映射及其复位值

地址偏移值	寄存器	7	6	5	4	3	2	1	0
0x00	CAN_MCR	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	复位值	0	0	0	0	0	0	1	0
0x01	CAN_MSR	-	-	RX	TX	WKUI	ERRI	SLAK	INAK
	复位值	0	0	0	0	0	0	1	0
0x02	CAN_TSR	-	TXOK2	TXOK1	TXOK0	-	RQCP2	RQCP1	RQCP0
	复位值	0	0	0	0	0	0	0	0
0x03	CAN_TPR	LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE1	CODE0
	复位值	0	0	0	1	1	1	0	0
0x04	CAN_RFR	-	-	RFOM	FOVR	FULL	-	FMP1	FMP0
	复位值	0	0	0	0	0	0	0	0
0x05	CAN_IER	WKUIE	-	-	-	FOVIE	FFIE	FMPIE	TMEIE
	复位值	0	0	0	0	0	0	0	0
0x06	CAN_DGR	-	-	-	TXM2E	RX	SAMP	SLIM	LBKM
	复位值	0	0	0	0	1	1	0	0
0x07	CAN_PSR	-	=	-	-	-	PS2	PS1	PS0
	复位值	0	0	0	0	0	0	0	0

表62 beCAN邮箱页列表 – 寄存器映射及其复位值

地址 (Hex)	寄存器	7	6	5	4	3	2	1	0
00h	CAN_MFMIR	FMI7	FMI6	FMI5	FMI4	FMI3	FMI2	FMI1	FMI0
接收	复位值	x	x	x	x	x	x	x	x
00h	CAN_MCSR	-	-	TERR	ALST	TXOK	RQCP2	ABRQ	TXRQ
发送	复位值	0	0	0	0	0	0	0	0
01h	CAN_MDLCR	TGT	-	-	-	DLC3	DLC2	DLC1	DLC0
	复位值	x	x	x	x	x	x	x	x
02h	CAN_MIDR1	-	IDE	RTR	STDI10/ EXID28	STID9/ EXID27	STID8/ EXID26	STID7/ EXID25	STID6/ EXID24
	复位值	x	x	x	x	x	x	x	x
03h	CAN_MIDR2	STID5/ EXID23	STID4/ EXID22	STID3/ EXID21	STID2/ EXID20	STID1/ EXID19	STID0/ EXID18	EXID17	EXID16
	复位值	x	x	x	x	x	x	x	x
04h	CAN_MIDR3	EXID15	EXID14	EXID13	EXID12	EXID11	EXID10	EXID9	EXID8
	复位值	x	x	x	x	x	x	x	x
05h	CAN_MIDR4	EXID7	EXID6	EXID5	EXID4	EXID3	EXID2	EXID1	EXID0
	复位值	x	x	x	x	x	x	x	x
06h:0Dh	CAN_MDAR1:8	MDAR7	MDAR6	MDAR5	MDAR4	MDAR3	MDAR2	MDAR1	MDAR0
	复位值	x	x	x	x	x	x	x	x
0Eh	CAN_MTSRL	TIME7	TIME6	TIME5	TIME4	TIME3	TIME2	TIME1	TIME0
	复位值	x	x	x	x	x	x	x	x
0Fh	CAN_MTSRH	TIME15	TIME14	TIME13	TIME12	TIME11	TIME10	TIME9	TIME8
	复位值	x	x	x	x	x	x	x	x

表63 beCAN过滤器配置页列表 – 寄存器映射及其复位值

地址 (Hex)	寄存器	7	6	5	4	3	2	1	0
00h	CAN_ESR	–	LEC2	LEC1	LEC0	–	BOFF	EPVF	EWGF
	复位值	0	0	0	0	0	0	0	0
01h	CAN_EIER	ERRIE	–	–	LECIE	–	BOFIE	EPVIE	EWGIE
	复位值	0	0	0	0	0	0	0	0
02h	CAN_TECR	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
	复位值	0	0	0	0	0	0	0	0
03h	CAN_RECRR	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
	复位值	0	0	0	1	1	1	0	0
04h	CAN_BTR1	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	复位值	0	1	0	0	0	0	0	0
05h	CAN_BTR2	CLKS	BS22	BS21	BS20	BS13	BS12	BS11	BS10
	复位值	0	0	1	0	0	0	1	1
06h	保留	– x							
07h	保留	– x							
08h	CAN_FMR1	FMH3	FML3	FMH2	FML2	FMH1	FML1	FMH0	FML0
	复位值	0	0	0	0	0	0	0	0
09h	CAN_FMR2	– 0	– 0	– 0	– 0	FMH5	FML5	FMH4	FML4
	复位值	0	0	0	0	0	0	0	0
0Ah	CAN_FCR1	– 0	FSC11	FSC10	FACT1	– 0	FSC01	FSC00	FACT0
	复位值	0	0	0	0	0	0	0	0
0Bh	CAN_FCR2	– 0	FSC31	FSC30	FACT3	– 0	FSC21	FSC20	FACT2
	复位值	0	0	0	0	0	0	0	0
0Ch	CAN_FCR3	– 0	FSC51	FSC50	FACT5	– 0	FSC41	FSC40	FACT4
	复位值	0	0	0	0	0	0	0	0

24 模拟 / 数字转换器(ADC)

24.1 简介

ADC1和ADC2是10位的逐次比较型模拟数字转换器。提供多达16个多功能的输入通道(实际准确的通道数量在数据手册的引脚描述说明)。A/D转换的各个通道可以执行单次和连续的转换模式。

相对于ADC2, ADC1具有一些扩展功能, 包括扫描模式, 带缓存的连续模式以及模拟看门狗。请参考数据手册来了解不同产品型号的ADC1和ADC2的功能信息。

24.2 主要功能

ADC1和ADC2的功能如下:

- 10位的分辨率
- 单次和连续的转换模式
- 可编程的(转换频率的)预分频: fMASTER 可以被分频 2 到 18
- 可以选择ADC专用外部中断(ADC_ETR)或者定时器触发信号(TRGO)来作为外部触发信号
- 模拟放大 (对于具有VREF引脚的型号)
- 转换结束时可产生中断
- 灵活的数据对齐方式
- ADC 输入电压范围: $V_{SSA} \leq V_{IN} \leq V_{DDA}$

24.3 扩展 (增强) 功能

ADC1具有以下扩展功能:

- 带缓冲的连续转换模式⁽¹⁾
- 单次和连续转换的扫描模式
- 具有上限和下限门槛的模拟看门狗
- 模拟看门狗事件发生可产生中断

ADC1 和 ADC2 的方块图如 [图146](#)和[图147](#)所示:

(1)数据缓存的大小依不同的型号而异(10X10位或8X10位)。详情请参考数据手册。

图146 ADC1方块图

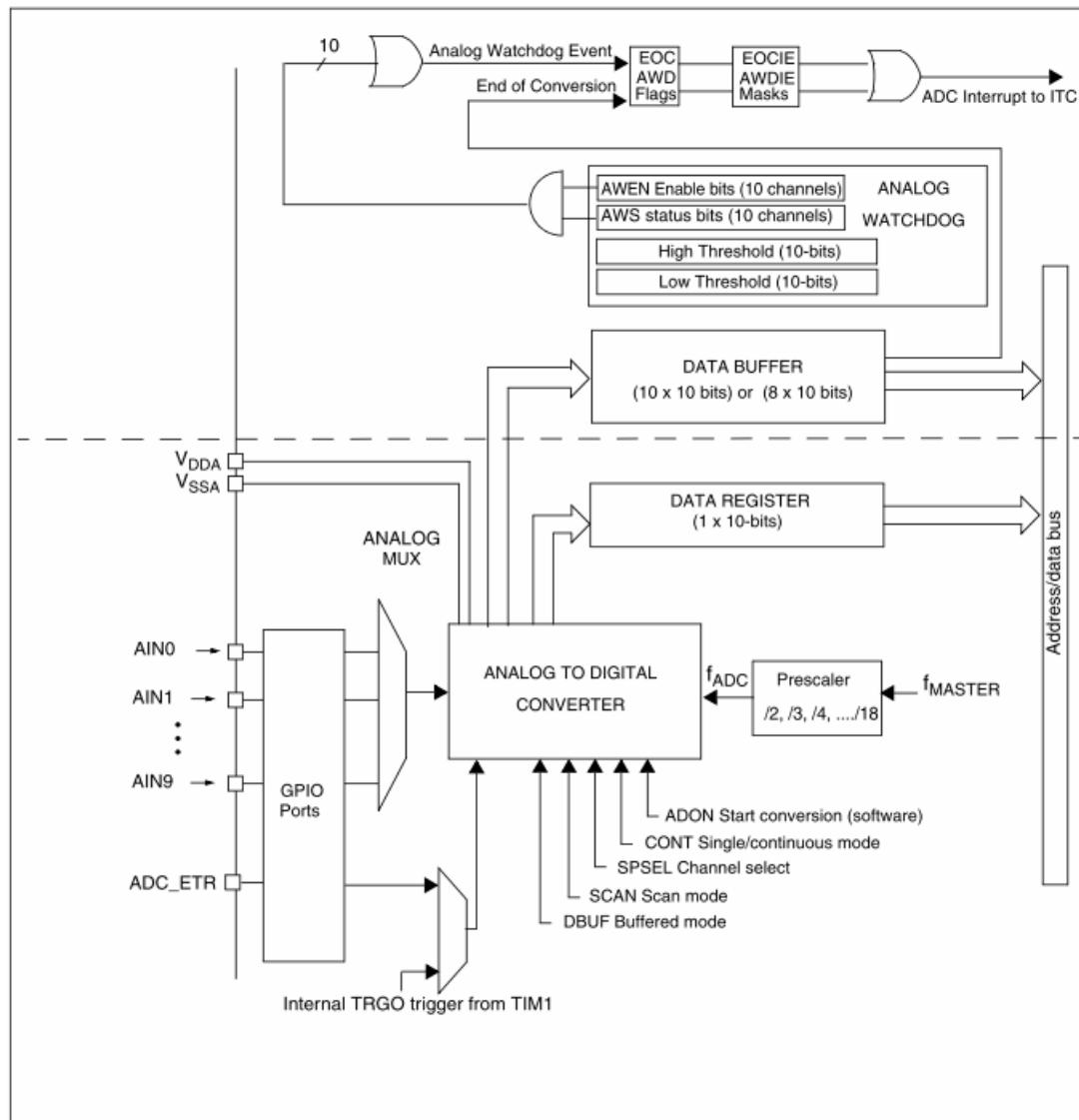
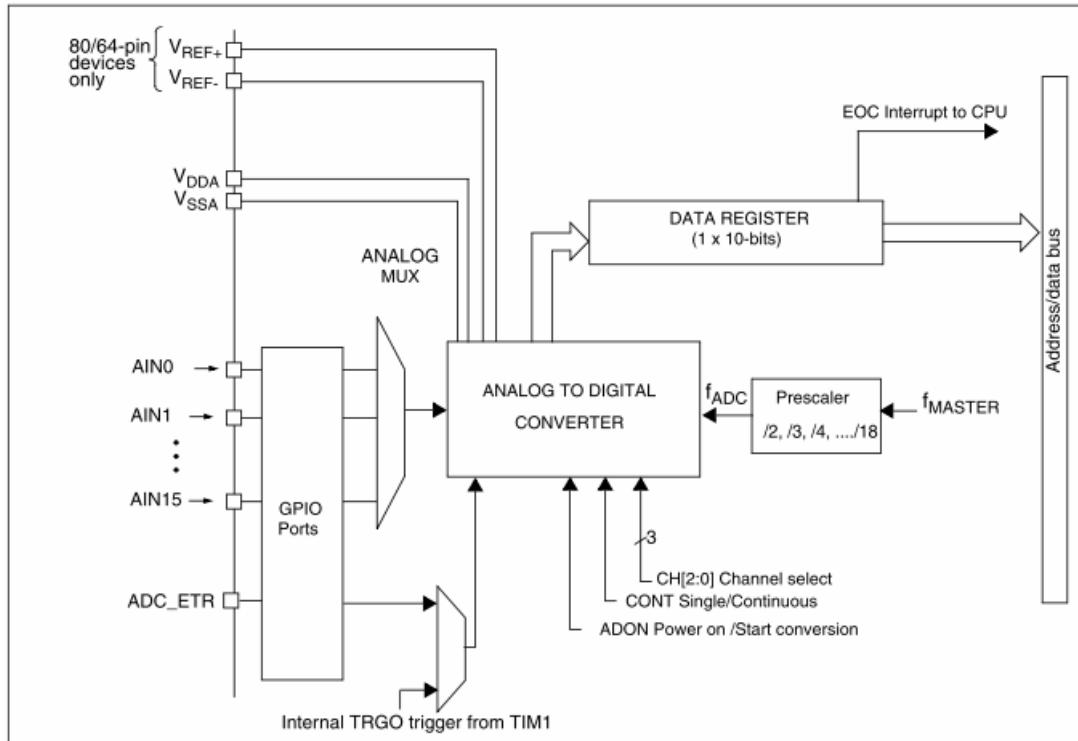


图147 ADC2方块图



24.4 引脚描述

表64 ADC 引脚

名称	信号类型	注解
V _{DDA}	输入, 模拟电源	模拟电源供电端。对于没有外部V _{DDA} 引脚的产品该输入脚是连接到V _{DD} 端
V _{SSA}	输入, 模拟电源地	模拟电源地端。对于没有外部V _{SSA} 引脚的产品该输入脚是连接到V _{SS} 端
V _{REF-}	输入, 模拟参考负极	ADC使用的低端/负极参考电压, 电压范围是V _{SSA} 到(V _{SSA} + 500 mV)。对于没有外部V _{REF-} 引脚的产品该输入脚是连接到V _{SSA} 端(48引脚封装或者更少引脚的封装)
V _{REF+}	输入, 模拟参考正极	ADC使用的高端/正极参考电压, 电压范围是2.75V到V _{DDA} 。对于没有外部V _{REF+} 引脚的产品该输入脚是连接到V _{DDA} 端(48引脚封装或者更少引脚的封装)
AIN[15:0]	模拟输入信号	多达16个模拟输入通道, 每次只一个通道被ADC转换
ADC_ETR	数字输入通道	外部触发信号

24.5 功能描述

24.5.1 ADC 开-关控制

通过置位ADC_CR1寄存器的ADON位来开启ADC。当首次置位ADON位时, ADC从低功耗模式唤醒。为了启动转换必须第二次使用写指令来置位ADC_CR1寄存器的ADON位。

在转换结束时ADC会保持在上电状态, 用户只需要置位ADON位一次来启动下一次的转换。

如果长时间没有使用ADC, 推荐将ADC模块切换到低功耗模式来降低功耗, 这可以通过清零ADON位来实现。

当ADC模块上电后, 所选通道对应的I/O口输出模块是被禁用的。因此推荐在ADC上电之前要选择合适的ADC转换通道。

24.5.2ADC 时钟

ADC 的时钟是由fMASTER时钟经过预分频后供给的。时钟的预分频因子是由 **ADC_CR1**寄存器的 **SPSEL[2:0]**决定的。

24.5.3通道选择

有多达 16 个外部输入通道。实际外部通道的数量取决于**MCU** 封装大小。

如果在一次转换过程中改变通道选择，那么当前的转换被复位同时一个新的开始指令脉冲被发送到 ADC。

24.5.4转换模式

ADC支持5种转换模式：单次模式，连续模式，带缓存的连续模式，单次扫描模式，连续扫描模式。

单次模式

在单次转换模式中，ADC仅在由**ADC_CSR**寄存器的**CH[3:0]**选定的通道上完成一次转换。该模式是在当**CONT**位为0时通过置位**ADC_CR1**寄存器的**ADON**位来启动的。

一旦转换完成，转换后的数据存储在**ADC_DR**寄存器中，**EOC**(转换结束)标志被置位，如果**EOCIE** 被置位将产生一个中断。

连续和带缓存的连续模式

在连续转换模式中，ADC在完成一次转换后就立刻开始下一次的转换。当**CONT**位被置位时即刻将ADC设为连续模式，该模式是通过置位 **ADC_CR1**寄存器的 **ADON** 位来启动的。

- 如果缓冲功能没有被使能(**ADC_CR3**寄存器的**DBUF**位=0)，那么转换结果数据保存在 **ADC_DR**寄存器中同时 **EOC** 标志被置位。如果**EOCIE** 位已被置位时将产生一次中断。然后开始下一次转换。
- 如果缓存功能被使能(**DBUF**=1)，那么某个选定通道上的8个或者10个连续的转换结果会填满数据缓存，当缓存被填满时，**EOC**(转换结束)标志被置位，如果**EOCIE**位已被置位，则会产生一个中断，然后一个新的转换自动开始。如果某个数据缓存寄存器在被读走之前被覆盖，**OVR**标志将置1。(见[24.5.5](#))

如果要停止连续转换，可以复位清零**CONT**位来停止转换或者复位清零**ADON**位来关闭ADC的电源。

单次扫描模式

该模式是用来转换从**AIN0**到**AINn**之间的一连串模拟通道，‘n’ 是在 **ADC_CSR**寄存器的 **CH[3:0]**位中指定的通道编号。在扫描转换的过程中，序号 **CH[3:0]**位的值是被硬件自动更新的，它总保存当前正在被转换的通道编号。

单次转换模式可以在在**SCAN** 位被置位且 **CONT** 位以被清零时通过置位 **ADON** 位来启动。

注意：当使用扫描模式时，不可以将**AIN0**到**AINn**之间通道对应的I/O口设为输出状态，因为ADC的多路选择器已经将这些I/O口的输出模块禁用了。

对于单次扫描模式，转换是从**AIN0**通道开始的，而且结果数据被存储在数据缓冲寄存器 **ADC_DBxR** 中，当最后一个通道(通道 ‘n’)被转换完成后，**EOC**(转换结束)标志被置位，当**EOCIE** 位已被置位时将产生一个中断。

可以从缓冲寄存器中读取各个通道的转换结果值。如果某个数据缓存寄存器在被读走之前被覆盖，**OVR**标志将置1。(见[24.5.5](#))

在转换序列正在进行过程中不要清零**SCAN**位；单次扫描模式可通过清零**ADON**位来立即停止。

为了开启一次新**SCAN**扫描转换，可以通过对**ADC_CR1**寄存器的**EOC**位清零和**ADON**位置位来实现。

连续扫描模式

该模式和单次扫描模式相近，只是每一次在最后通道转换完成时，一次新的从通道0到通道n扫描转换会自动开始。如果某个数据缓存寄存器在被读走之前被覆盖，OVR标志将置1。(见24.5.5) 连续扫描模式是在当SCAN位和CONT位已被置时，通过置位ADON位来启动的。

在转换序列正在进行过程中不要清零SCAN位。

连续扫描模式可以通过清零ADON位来立即停止。另外一种选择就是当转换过程中清除CONT位那么转换会在下一次的最后一个通道转换完成时停止。

注意：在扫描模式中，不要使用位操作指令(BRES)去清除EOC标志位，这是因为该指令是对整个ADC_CSR寄存器的一个读-修改-写操作。从CH[3:0]寄存器中读取当前的通道编号和写回该寄存器，将会改变扫描系列的最后通道编号。

在连续扫描模式中正确的清除EOC标志位的方法是从一个RAM变量中载入一个字节到ADC_CSR寄存器，这样来清除EOC标志位同时还重新载入扫描系列新的最后通道编号。

24.5.5溢出标志位

在带缓冲的连续模式，单次扫描模式或者连续扫描模式中，溢出错误标志位OVR位是由硬件置位的。它是用来指示10个缓冲寄存器中的某个值在被读走之前被一个新的转换结果值覆盖。在这种情况下，推荐重新开启一次新的转换过程(因为某个有效的结果已经被覆盖了)。

注意：置位ADON位会自动清除OVR标志位。

24.5.6模拟看门狗

在单次转换模式和不带缓存的连续模式中模拟看门狗可以通过置位ADC_CSR寄存器的AWDEN位来使能。

如图148当模拟电压通过ADC转换后的值低于下限门槛值或者高于上限门槛值时AWD模拟看门狗会被置位。可通过对ADC_HTR和ADC_LTR的10位寄存器编程来设定门槛值，并且可以通过置位ADC_CSR寄存器的AWDIE位可使能中断。

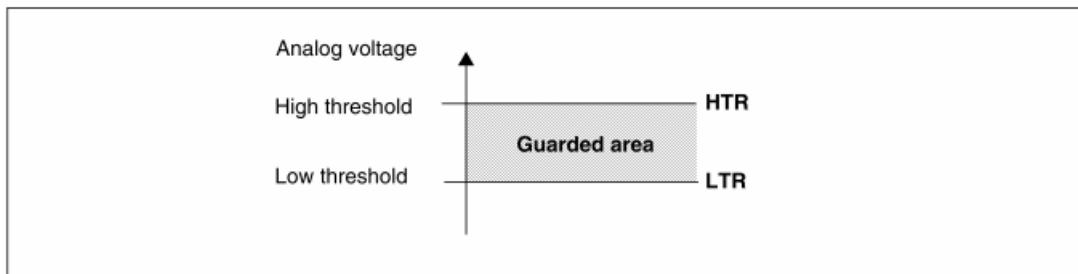
对于扫描模式，通过使用ADC_AWCRH和ADC_AWCRL寄存器的AWENx位在选定的通道上使能模拟看门狗功能。对于每个通道上的模拟看门狗的状态可以通过读ADC_AWSRH 和ADC_AWSRL寄存器的AWSx来获得。一旦任何一个AWS标志位被置位这同时也会置位AWD标志位。如果AWDIE的中断使能位为1，会在一个SCAN序列结束时产生一个中断，中断子程序必须清除ADC_CSR寄存器的AWS和 AWD标志位。

对于带缓存的连续模式，模拟看门狗可以如扫描模式一样来在选定的通道上使能并对之管理。

关于中断的更详细的情况参考24.7。

注意：为了在扫描和带缓存的连续模式中优化模拟看门狗中断等待延时时间，推荐在转换序列的最后一个通道使用模拟看门狗。

图148 模拟看门狗的安全区域



24.5.7 基于外部触发信号的转换

ADC转换可以通过ADC_ETR引脚上的上升沿事件或来自定时器的TRGO事件来触发启动。(请参考数据手册来了解关于定时器触发信号的细节)。当EXTTRIG控制位被置位时,那么任一ADC外部触发事件都可以用作转换启动的触发信号。EXTSEL[1:0]位被用来从2个信号源中选择触发信号源。

为了使用外部触发信号模式需操作如下:

1. ADC是在关闭状态 (ADON=0)以及EOC位被清零。
2. 选择触发信号源 (EXTSEL[1:0])。
3. 使用BSET指令(防止更改同一寄存器的其他位)设置外部触发信号模式 EXTTRIG=1。
4. 如果触发信号源是在高电平状态,这会导致开启ADC。因此测试ADC是否在关闭状态 (ADON=0),然后再开启ADC(ADON=1)。
5. 等待一个稳定时间 (tSTAB)。如果在一个外部触发信号在tSTAB时间段中发生,那么转换结果将是不精确的。
6. 当外部触发信号事件发生时转换开始。

注意: 1 如果定时器触发模式被选定(定时器事件作为触发源,而不是外部引脚),那么推荐在ADC完成设置之后启动定时器和在关闭ADC之前先停止定时器。

2 在执行 HALT 指令之前必须先禁止外部触发模式(EXTTRIG=0)。

24.5.8 模拟放大

带外部参考电压引脚(VREF+和VREF-)的产品支持模拟放大功能,在模拟放大中,可通过减小参考电压来提供更大的分辨率。详细的关于允许的参考电压范围参考数据手册。

24.5.9 时序图

如图149所示,在ADC上电后,在开始精确转换之前ADC需要一个稳定时间tSTAB (等于一次转换的时间 tCONV),对于之后接下来的转换就不需要稳定延时,而且ADON位只需要被置位一次。一次ADC转换需要14个时钟周期,在转换完成后EOC标志被置位,同时转换结果保存在10位ADC数据寄存器里面。

图149 单次模式的时序图 (CONT=0)

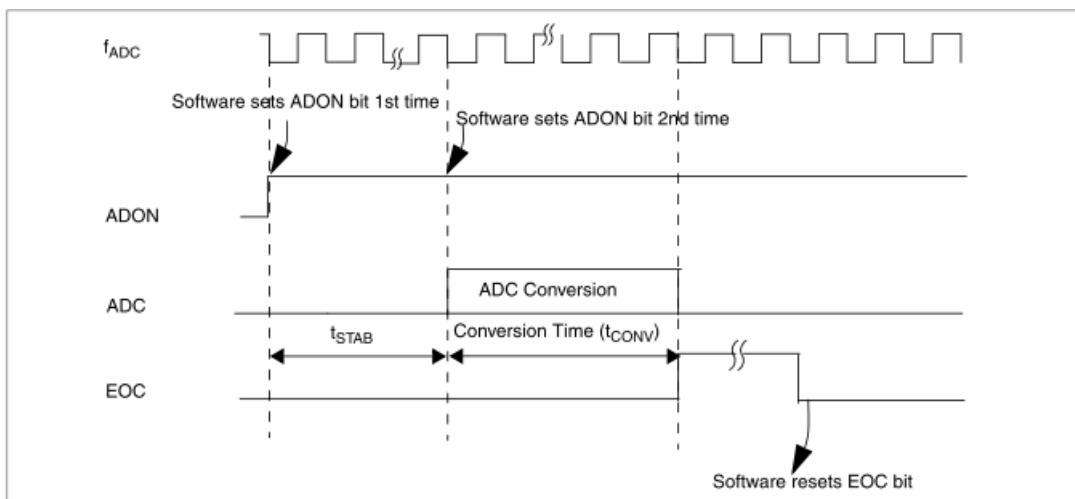
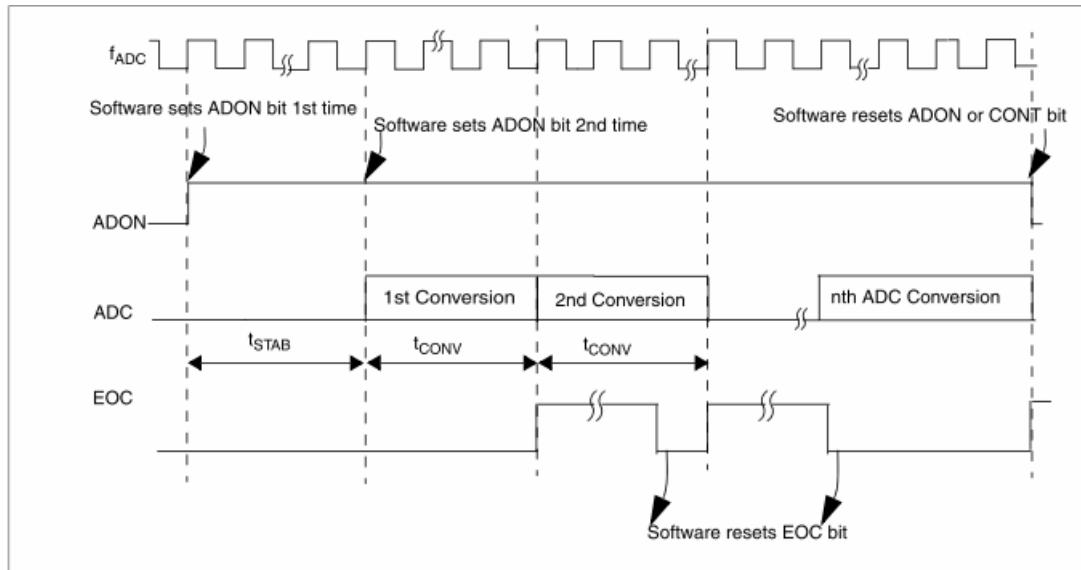


图150 连续模式的时序图 (CONT=1)



24.6 低功耗模式

表65 低功耗模式

模式	描述
WAIT	无影响
HALT/ Fast Active HALT/ Slow Active HALT	对于具有扩展功能的产品，在进入HALT/Active HALT模式之前会自动关闭ADC。当从HALT/Fast Active HALT或Slow Active HALT模式中唤醒时，必须软件置位ADON位来上电ADC，之后在开始一次新的转换之前必须有7 μ s的延时。

ADC 不能够从活跃停机(Active Halt)模式或者停机(Halt)模式中唤醒MCU。

24.7 中断

ADC 的中断控制位总结在 [表 66](#)、[表 67](#)和 [表 68](#)中。

表66 在单次模式和不带缓存的连续模式中的 ADC 中断 (ADC1 和 ADC2)

使能位			状态标志			从等待 (WAIT) 模式退出	从停机 (HALT) 模 式退出
AWENx	AWDIE	EOCIE	AWSx	AWD	EOC		
不起作用	0	0	不起作用	如果通道的电压超过编程的门槛值置位该标志位	在每次转换结束置位该标志位	No	No
	0	1		如果通道的电压超过编程的门槛值置位该标志位	在每次转换结束和中断产生时置位该标志位	Yes	No
	1	0		如果通道的电压超过编程的门槛值会置位该标志位。此时会产生一个中断但连续转换不会停止。	在每次转换结束置位该标志位	Yes	No
	1	1		如果通道的电压超过编程的门槛值会置位该标志位。此时会产生一个中断但连续转换不会停止。	在每次转换结束和中断产生时置位该标志位	Yes	No

表67 带缓存的连续模式中的 ADC 中断 (ADC1)

使能位			状态标志			从等待 (WAIT) 模式退出	从停机 (HALT) 模式退出
AWEN _x	AWDIE	EOCIE	AWS _x	AWD	EOC		
0	不起作用	0	0	0	在BSIZE转换结束时置位该标志位	No	No
0	不起作用	1	0		在BSIZE转换结束和中断产生时置位该标志位	Yes	No
1	0	0	当至少一个AWS _x 被置位时，在BSIZE转换结束置位该标志位 当至少一个AWS _x 被置位时，在BSIZE转换结束时置位该标志位，并且产生一个中断。但是连续转换不会停止。	在BSIZE转换结束置位该标志位(数据缓冲区满)	No	No	
1	0	1			Yes	No	
1	1	0	当至少一个AWS _x 被置位时，在BSIZE转换结束置位该标志位 一旦任何一个AWS _x 被置位时该标志被立即置位。当一个中断产生时连续转换停止。	在BSIZE转换结束和中断产生时置位该标志位	Yes	No	
1	1	1			Yes	No	

注意: **BSIZE** = 数据缓存大小(根据产品型号不同, 有大小为8或10字节)

表68 在扫描模式中的 ADC 中断 (ADC1)

使能位			状态标志			从等待 (WAIT) 模式退出	从停机 (HALT) 模式退出
AWEN _x	AWDIE	EOCIE	AWS _x	AWD	EOC		
0	不起作用	0	0	0	在扫描序列结束后, 置位该标志位。	No	No
0	不起作用	1	0	0	在扫描序列结束后, 置位该标志位, 同时产生一个中断。	Yes	No
1	0	0	当存储于第 _x 个缓存区的转换结果超过了在ADC_HTR和ADC_LTR中的预设值时, 置位, 该标志位。	当至少一个AWS _x 被置位时, 在扫描序列结束置位该标志位。	在扫描序列结束后, 置位该标志位。	No	No
1	1	0		当至少一个AWS _x 被置位时, 在扫描序列结束时置位该标志位, 并且产生一个中断。但是连续转换不会停止。	在扫描序列结束后, 置位该标志位。	Yes	No
1	0	1		当至少一个AWS _x 被置位时, 在扫描序列结束时, 置位该标志位。	在扫描序列结束后, 置位该标志位, 同时产生一个中断。	Yes	No
1	1	1		一旦任何一个AWS _x 被置位, 该标志被立即置位。当一个中断产生时连续转换停止。	在扫描序列结束后, 置位该标志位, 同时产生一个中断。	Yes	No

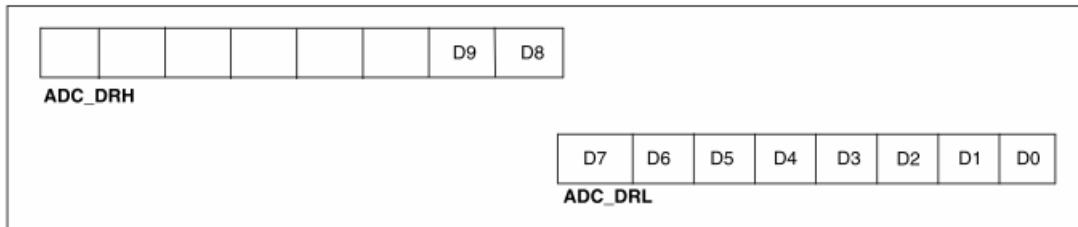
24.8 数据对齐

ADC_CR2寄存器中的ALIGN位用于选择转换后数据的对齐方式。

数据可以按如下方式对齐。

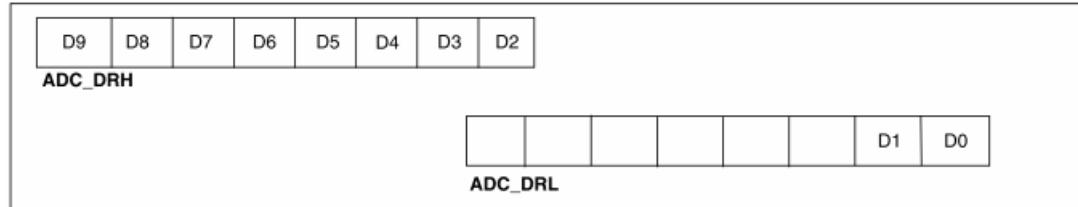
右对齐: 8个低位数据被写入ADC_DL寄存器中, 其余的高位数据被写入ADC_DH寄存器中。读取时必须先读低位再读高位。

图151 数据右对齐



左对齐：8个高位数据被写入ADC_DRH寄存器，其余的低位数据被写入ADC_DRL寄存器。读取时必须先读高位再读低位。

图152 数据左对齐



24.9 读取转换结果

当读取ADC转换结果时，须注意要依据所选择的数据对齐方式，按照指定的方式连续使用两条指令来读取数据寄存器。

为了保证数据一致性，MCU采用内部锁存机制。对应ADC_DR的高位寄存器和低位寄存器，直到读取一个指定数据寄存器之前，另一个寄存器的转换数据结果不会被修改。因此，按照错误的顺序来读取寄存器将得到错误的结果。

寄存器读取顺序由数据对齐方式决定(参考24.8)

为了得到正确的结果：

- 在左对齐模式下，先读高位字节寄存器(ADC_DRH)，再读低位字节寄存器(ADC_DRL)。
- 在右对齐模式下，先读低位字节寄存器(ADC_DRL)，再读高位字节寄存器(ADC_DRH)。此时，用户也可以选择使用LDW指令来读取整个ADC_DR，因为该指令和右对齐模式采用同样的读取顺序。

24.10 施密特触发器禁止寄存器

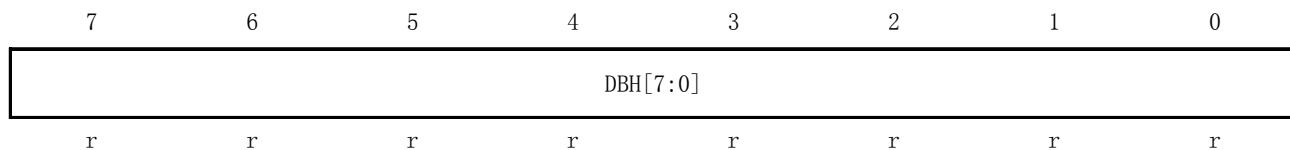
ADC_TDRH和ADC_TDRL寄存器可以用来禁止AIN模拟输入引脚中的施密特触发器工作。禁止施密特触发器工作可以降低I/O引脚的功耗。

24.11 寄存器描述

24.11.1 ADC高位数据缓存寄存器(ADC_DBxRH)(x=0..7 or 0..9)

地址偏移值: 0x00+2*通道号

复位值: 0x00



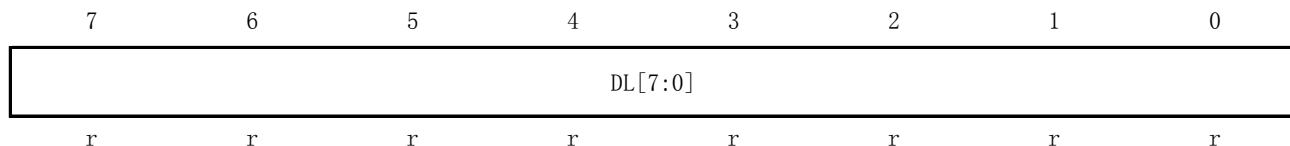
注意: ADC2无数据缓存寄存器。不同的MCU有不同大小的数据缓存, 请查询相应的数据手册。

位7:0	<p>DBH[7:0]: 数据高位</p> <p>这些位由硬件置0或置1, 并且只能读取。当ADC处于连续缓存或扫描的模式时, 这些位为转换结果的高位部分。数据的左对齐还是右对齐由ALIGN位决定。</p> <p>数据左对齐</p> <p>这些数据位包含高8位的转换数据。须在读低位数据前先读取。(参考24.9 和 图152)</p> <p>数据右对齐</p> <p>这些数据包含(ADC数据宽度减8)的高位转换结果数据。剩下的位为0。 请参考 图151。</p>
------	---

24.11.2 ADC低位数据缓存寄存器(ADC_DBxRL)(x=0..7 or 0..9)

地址偏移值: 0x01+2*通道号

复位值: 0x00



注意: ADC2无数据缓存寄存器。不同的MCU有不同大小的数据缓存, 请查询相应的数据手册。

位7:0	DL[7:0]: 数据低位 这些位由硬件置0或置1, 并且只能读取。当ADC处于连续缓存或扫描的模式时, 这些位为转换结果的低位部分。数据的左对齐还是右对齐由ALIGN位决定。 数据左对齐 这些数据位包含低位(ADC数据宽度减8)的转换数据。剩下的位为0。(参考 图152) 数据右对齐 这些数据位包含低8位转换结果数据。低位字节必须先读, 再读高位字节。(参考 24.9 和 图151)
------	--

24.11.3 ADC控制/状态寄存器(ADC_CSR)

地址偏移值: 0x00

复位值: 0x00

7	6	5	4	3	2	1	0
EOC	AWD	EOCIE	AWDIE		CH[3:0]		

rw rc_w0 rw rw rw rw rw rw

位7	EOC: 转换结束 此位在AD转换结束后由硬件置位。由软件通过写“0”来清零 0: 转换未结束 1: 转换结束
位6	AWD: 模拟看门狗标志 0: 无模拟看门狗事件 1: 产生了模拟看门狗事件。在连续缓存和扫描模式下，用户可以读ADC_AWSR寄存器来判定与此事件相关的数据缓存寄存器。如果AWDIE=1, 将产生一次中断。 注意: 此位在ADC2上无效。
位5	EOCIE: 转换结束EOC的中断使能 此位由软件置位和清零。设置此位使能转换结束中断。 0: 禁止转换结束中断 1: 使能转换结束中断
位4	AWDIE: 模拟看门狗使能位 0: 禁止AWD模拟看门狗中断 1: 使能AWD模拟看门狗中断 注意: 此位在ADC2上无效。
位3:0	CH[3:0]: 选择转换通道位 此位由软件置位和清零。此位选择要转换的通道。 0000: 模拟通道 AIN0 0001: 模拟通道 AIN1 1111: 模拟通道 AIN15

24.11.4 ADC 配置寄存器 1 (ADC_CR1)

地址偏移值: 0x01

复位值: 0x00

7	6	5	4	3	2	1	0
保留		SPSEL[2:0]		保留		CONT	ADON

r w r w r w r w r w r w

位7	保留位, 读此位为0
位6:4	<p>SPSEL [2:0]: 预分频选择位 由软件设置这些位来选择相应的预分频因数。</p> <p>000: $f_{ADC} = f_{MASTER}/2$ 001: $f_{ADC} = f_{MASTER}/3$ 010: $f_{ADC} = f_{MASTER}/4$ 011: $f_{ADC} = f_{MASTER}/6$ 100: $f_{ADC} = f_{MASTER}/8$ 101: $f_{ADC} = f_{MASTER}/10$ 110: $f_{ADC} = f_{MASTER}/12$ 111: $f_{ADC} = f_{MASTER}/18$</p> <p>参考24.5.2 注意: 建议在ADC进入低功耗模式时改变SPSEL位。这是因为在转换时, 可能对内部时钟产生一个干扰。否则, 如果在非低功耗模式下改变通道, 则用户须忽略第一次转换结果。</p>
位3,2	保留位, 读此位为0
位1	<p>CONT: 连续转换 此位由软件置位和清零。如果此位置1, 将产生连续转换, 直到此位被软件复位。</p> <p>0: 单次转换模式 1: 连续转换模式</p>
位0	<p>ADON: A/D 转换开/关 此位由软件置位和清零。须通过写此位来把ADC从低功耗模式唤醒并触发一次AD转换。如果此位是0时, 并且写1到此位, 那么将把ADC从低功耗模式下唤醒。如果在此位是1, 并且写1到此位, 那么将启动AD转换。一旦ADC上电, 所选转换通道的I/O输出功能就被禁用了。</p> <p>0: 禁止ADC 转换/校准, 并且进入低功耗模式。 1: 使能ADC并开始转换。</p> <p>注意: 如果此寄存器中除了ADON的其他位同时改变, 那么将不能触发转换。这样可避免产生一次错误的转换。</p>

24.11.5 ADC 配置寄存器 2 (ADC_CR2)

地址偏移值: 0x02

复位值: 0x00

7	6	5	4	3	2	1	0
保留	EXTTRIG	EXSEL[1:0]		ALIGN	保留	SCAN	保留

r w r w r w r w r w r w

位7	保留位, 必须为0
位6	EXTTRIG: 外触发使能位 此位由软件置位和清零。此位用来使能外部触发来触发一次转换。 0: 禁止外部触发转换 1: 使能外部触发转换 注意: 为了避免错误的触发事件, 使用BSET指令来设置EXTTRIG位, 不用改变此寄存器的其他位
位5;4	EXTSEL[1:0]: 外部事件选择位 这两位由软件写入设置, 用来选择四种类型的外部事件触发和启动AD转换。 00: 内部定时器1 TRG事件 01: ADC_ETR引脚上的外部中断 10: 保留 11: 保留
位3	ALIGN: 数据排列 此位由软件置位和清零。 0: 数据左对齐。(高8位字节在ADC_DRH寄存器, 其余的低位字节在ADC_DRL寄存器)。读顺序为先读高, 后读低。 1: 数据右对齐。(低8字节在ADC_DRL寄存器, 其余高字节在ADC_DRH寄存器)读顺序应先读低位, 再读高位字节
位2	保留位, 必须为0
位1	SCAN: 扫描模式使能 此位由软件置位和清零。 0: 禁止扫描模式 1: 使能扫描模式 注意: 此位在ADC2上无效。
位0	保留位, 必须为0

24.11.6 ADC配置寄存器3 (ADC_CR3)

地址偏移值: 0x03

复位值: 0x00

7	6	5	4	3	2	1	0
DBUF	OVR				保留		

rw rc_w0

注意: ADC2无此寄存器

位7	DBUF: 数据缓存使能 此位由软件置位和清零。此位与CONT位一起使能连续缓存模式(DBUF=1, CONT=1)。当DBUF位置1时, 转换结果存在ADC_DBxRH 和 ADC_DBxRL 寄存器中, 而不是在ADC_DRH 和 ADC_DRL寄存器中。 0: 数据缓存功能禁止。 1: 数据缓存功能使能。
位6	OVR: 溢出标志位 此位由硬件置位和软件清零。 0: 无数据溢出发生。 1: 数据缓存发生了数据溢出事件 详细请参考 24.5.5 。
位5:0	保留位, 必须为0

24.11.7 ADC 数据高位寄存器(ADC_DRH)

地址偏移值: 0x04

复位值: 未定义

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

DH[7:0]

r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---

位7	DH [7:0] 数据高位 这些位由硬件置位和清零，并且为只读。当ADC处于单次或非缓冲转换模式时，寄存器的值为ADC转换结果的高位值。至于数据左对齐还是右对齐，由ALIGN位决定。
	数据左对齐 这些位包含高8位转换数据。必须先读高8位，再读低位。(参考 24.9 和 图152)
	数据右对齐 这些数据位包含高位(ADC数据宽度减8)的转换数据。其余的位为0。见 图151

24.11.8 ADC 数据低位寄存器(ADC_DRL)

地址偏移值: 0x06

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

DL[7:0]

r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---

位7	DL [7:0] 数据低位
	这些位由硬件置位和清零，并且为只读。当ADC处于单次或非缓冲转换模式时，寄存器的值为ADC转换结果的低位值。至于数据左对齐还是右对齐，由ALIGN位决定。
	数据左对齐 这些数据位包含低位位(ADC数据宽度减8)的转换数据。其余的位为0。见 图152 数据右对齐 这些位包含低8位转换数据。必须先读低8位，再读高位。(参考 24.9 和 图151)

24.11.9 ADC 施密特触发器禁止寄存器高位 (ADC_TDRH)

地址偏移值: 0x06

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

TD[15:8]

rw							
----	----	----	----	----	----	----	----

位7:1	保留位, 必须为0
位7	<p>TD[15:8] : 施密特触发器禁止高位</p> <p>这些位由软件置位和清零。当TDx位置位时, 即使当时该通道无AD转换, 也禁止该通道相应的施密特触发器功能。此项是为了降低IO口的静态功耗。</p> <p>0: 使能施密特触发功能。</p> <p>1: 禁止施密特触发功能。</p>

24.11.10 ADC 施密特触发器禁止寄存器低位 (ADC_TDRL)

地址偏移值: 0x07

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

TD[17:0]

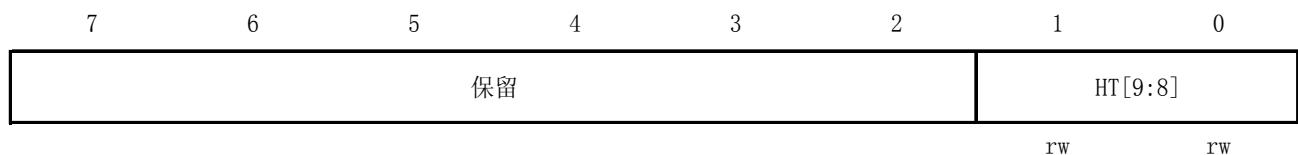
rw							
----	----	----	----	----	----	----	----

位7:1	保留位, 必须为0
位7	<p>TD[7:0] : 施密特触发器禁止高位</p> <p>这些位由软件置位和清零。当TDx位置位时, 即使D当时该通道无AD转换, 也禁止该通道相应的施密特触发器功能。此项是为了降低IO口的静态功耗。</p> <p>0: 使能施密特触发功能。</p> <p>1: 禁止施密特触发功能。</p>

24.11.11 ADC 上限门槛值高位寄存器(ADC_HTRH)

地址偏移值: 0x08

复位值: 0x03



注意: ADC2无此寄存器

位7:2	保留位, 必须为0
位1:0	HT [9:8]: 模拟看门狗上限电压高位 此位由硬件置位和软件清零。这些位定义了模拟看门狗上限电压高位值(VREFH)的MSB

24.11.12 ADC 上限门槛值低位寄存器(ADC_HTRL)

地址偏移值: 0x09

复位值: 0xFF

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

HT[7:0]

rw							
----	----	----	----	----	----	----	----

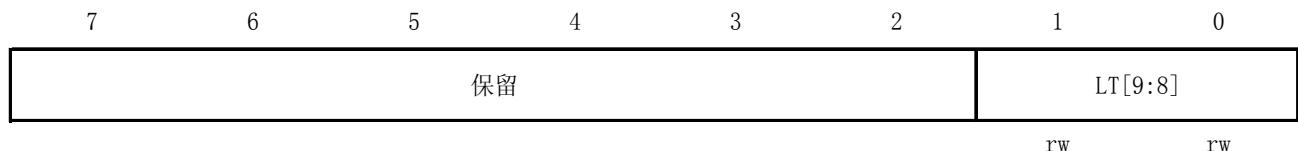
注意: ADC2无此寄存器

位7:0	HT[7:0]: 模拟看门狗上限电压低位 此位由硬件置位和软件清零。这些位定义了模拟看门狗上限电压低位值(VREFH)的LSB
------	--

24.11.13 ADC 下限门槛值高位寄存器(ADC_LTRH)

地址偏移值: 0x0A

复位值: 0x03



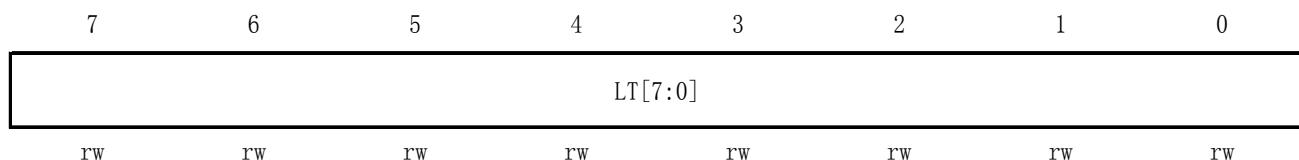
注意: ADC2无此寄存器

位7:2	保留位, 必须为0
位1:0	LT [9:8]: 模拟看门狗下限电压高位 此位由硬件置位和软件清零。这些位定义了模拟看门狗下限电压高位值(VREFH)的MSB

24.11.14 ADC 下限门槛值低位寄存器(ADC_LTRL)

地址偏移值: 0x0B

复位值: 0x00



注意: ADC2 无此寄存器

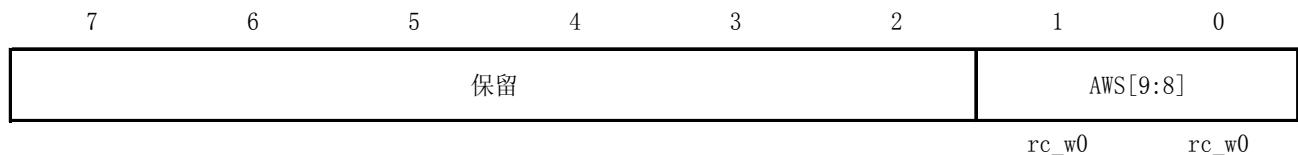
位7:0	LT[7:0]: 模拟看门狗下限电压低位 此位由硬件置位和软件清零。这些位定义了模拟看门狗下限电压低位值(VREFH)的LSB
------	---



24.11.15 ADC看门狗状态高位寄存器(ADC_AWSRH)

地址偏移值: 0x0C

复位值: 0x00



注意: ADC2无此寄存器

位7:2	保留位, 必须为0
位1:0	<p>AWS [9:8]: 模拟看门狗状态标志位 9:8 此位由硬件置位和软件清零。</p> <ul style="list-style-type: none"> - 在连续缓存模式下(DBUF=1, CONT=1), AWS 标志功能如表67所示。 - 在扫描模式下(SCAN=1), AWS标志功能如表68所示。 <p>0: 在数据寄存器X中无模拟看门狗事件。 1: 数据寄存器X中发生了模拟看门狗事件。</p>

24.11.16 ADC看门狗状态低位寄存器(ADC_AWSRL)

地址偏移值: 0x0D

复位值: 0x00

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

AWS[7:0]

rc_w0							
-------	-------	-------	-------	-------	-------	-------	-------

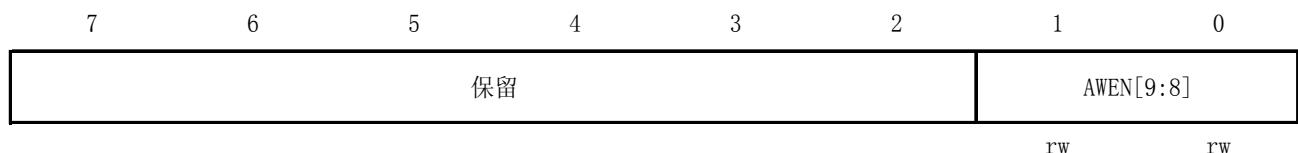
注意: ADC2无此寄存器

位7:0	AWS [7:0]: 模拟看门狗状态标志位 7:0
	此位由硬件置位和软件清零。 - 在连续缓存模式下(DBUF=1, CONT=1), AWS 标志功能如 表67 所示。 - 在扫描模式下(SCAN=1), AWS标志功能如 表68 所示。 0: 在数据寄存器X中无模拟看门狗事件 1: 数据寄存器X中发生了模拟看门狗事件。

24.11.17 ADC看门狗控制高位寄存器(ADC_AWCRH)

地址偏移值: 0x0E

复位值: 0x00



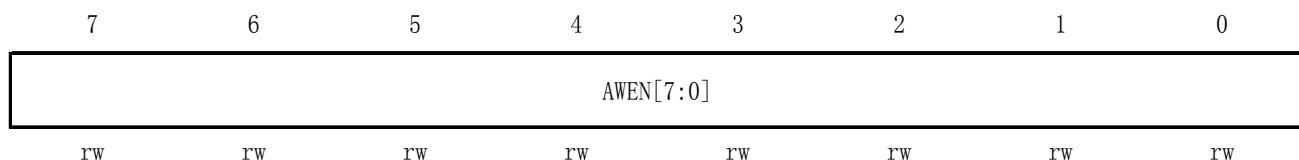
注意: ADC2无此寄存器

位7:2	保留位, 必须为0
位1:0	<p>AWEN [9:8]: 模拟看门狗使能位 9:8 此位由软件置位清零。</p> <p>在连续缓存模式下(DBUF=1, CONT=1)和扫描模式下(SCAN=1), AWENx位使能每一个10位缓存寄存器的看门狗功能。</p> <p>0: 在数据寄存器x中禁止模拟看门狗功能。 1: 数据寄存器x中使能模拟看门狗。</p>

24.11.18 ADC看门狗控制低位寄存器(ADC_AWCRL)

地址偏移值: 0x0F

复位值: 0x00



注意: ADC2无此寄存器

位7:0	AWEN [7:0]: 模拟看门狗使能位 7:0 此位由软件置位清零。 在连续缓存模式下(DBUF=1, CONT=1)和扫描模式下(SCAN=1), AWENx位使能每一个10位缓存寄存器的看门狗功能。 0: 在数据寄存器x中禁止模拟看门狗功能 1: 数据寄存器x中使能模拟看门狗。
------	---

24.12 ADC寄存器映像表和复位值

表69 ADC1寄存器映像表和复位值

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	ADC1_DB0RH 复位值	- 0	- 0	- 0	- 0	- 0	- 0	DATA9 0	DATA8 0
0x01	ADC1_DB0RL 复位值	DATA7 0	DATA6 0	DATA5 0	DATA4 0	DATA3 0	DATA2 0	DATA1 0	DATA0 0
-					-				
0x0E	ADC1_DB7RH 复位值	- 0	- 0	- 0	- 0	- 0	- 0	DATA9 0	DATA8 0
0x0F	ADC1_DB7RL 复位值	DATA7 0	DATA6 0	DATA5 0	DATA4 0	DATA3 0	DATA2 0	DATA1 0	DATA0 0
0x0E	ADC1_DB8RH ⁽¹⁾ 复位值	- 0	- 0	- 0	- 0	- 0	- 0	DATA9 0	DATA8 0
0x0F	ADC1_DB8RL ⁽¹⁾ 复位值	DATA7 0	DATA6 0	DATA5 0	DATA4 0	DATA3 0	DATA2 0	DATA1 0	DATA0 0
0x0E	ADC1_DB9RH ⁽¹⁾ 复位值	- 0	- 0	- 0	- 0	- 0	- 0	DATA9 0	DATA8 0
0x0F	ADC1_DB9RL ⁽¹⁾ 复位值	DATA7 0	DATA6 0	DATA5 0	DATA4 0	DATA3 0	DATA2 0	DATA1 0	DATA0 0
0x00	ADC1_CSR 复位值	EOC 0	AWD 0	EOCIE 0	AWDIE 0	CH3 0	CH2 0	CH1 0	CHO 0
0x01	ADC1_CR1 复位值	- 0	SPSEL2 0	SPSEL1 0	SPSEL0 0	- 0	- 0	CONT 0	ADON 0
0x02	ADC1_CR2 复位值	- 0	EXTITRIG 0	EXTSEL1 0	EXTSEL0 0	ALIGN 0	- 0	SCAN 0	- 0
0x03	ADC1_CR3 复位值	DBUF 0	OVR 0	- 0	- 0	- 0	- 0	- 0	- 0
0x04	ADC1_DRH 复位值	- X	- X	- X	- X	- X	- X	DATA9 X	DATA8 X
0x05	ADC1_DRL 复位值	DATA7 X	DATA6 X	DATA5 X	DATA4 X	DATA3 X	DATA2 X	DATA1 X	DATA0 X
0x06	ADC1_TDRH ⁽²⁾ 复位值	TD15 X	TD14 X	TD13 X	TD12 X	TD11 X	TD10 X	TD9 X	TD8 X

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x07	ADC1_TDRL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
	复位值	X	X	X	X	X	X	X	X
0x08	ADC1_HTRH	-	-	-	-	-	-	HT9	HT8
	复位值	X	X	X	X	X	X	X	X
0x09	ADC1_HTRL	HT7	HT6	HT5	HT4	HT3	HT2	HT1	HT0
	复位值	X	X	X	X	X	X	X	X
0x0A	ADC1_LTRH	-	-	-	-	-	-	LT9	LT8
	复位值	X	X	X	X	X	X	X	X
0x0B	ADC1_LTRL	LT7	LT6	LT5	LT4	LT3	LT2	LT1	LT0
	复位值	X	X	X	X	X	X	X	X
0x0C	ADC1_AWSRH ⁽²⁾	-	-	-	-	-	-	AWS9	AWS8
	复位值	X	X	X	X	X	X	X	X
0x0D	ADC1_AWSRL	AWS7	AWS6	AWS5	AWS4	AWS3	AWS2	AWS1	AWS0
	复位值	X	X	X	X	X	X	X	X
0x0E	ADC1_AWCRH ⁽²⁾	-	-	-	-	-	-	AWEN9	AWEN8
	复位值	X	X	X	X	X	X	X	X
0x0F	ADC1_AWCRL	AWEN7	AWEN6	AWEN5	AWEN4	AWEN3	AWEN2	AWEN1	AWENO
	复位值	X	X	X	X	X	X	X	X

1. 在8x10位缓存的MCU中，这些寄存器被保留。
2. 在没有ADC通道8和9的MCU中，这些寄存器被保留。

表70 ADC2寄存器映像表和复位值

地址偏移	寄存器	7	6	5	4	3	2	1	0
0x00	ADC2_CSR	EOC	AWD	EOCIE	AWDIE	CH3	CH2	CH1	CH0
	复位值	0	0	0	0	0	0	0	0
0x01	ADC2_CR1	-	SPSEL2	SPSEL1	SPSEL0	-	-	CONT	ADON
	复位值	0	0	0	0	0	0	0	0
0x02	ADC2_CR2	-	EXTTRIG	EXTSEL1	EXTSEL0	-	-	-	-
	复位值	0	0	0	0	0	0	0	0
0x03	ADC2_CR3	DBUF	OVR	-	-	-	-	-	-
	复位值	0	0	0	0	0	0	0	0
0x04	ADC2_DRH	-	-	-	-	-	-	DATA9	DATA8
	复位值	0	0	0	0	0	0	0	0
0x05	ADC2_DRL	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
	复位值	0	0	0	0	0	0	0	0
0x06	ADC2_TDRH	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8
	复位值	0	0	0	0	0	0	0	0
0x07	ADC2_TDRL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
	复位值	0	0	0	0	0	0	0	0