

**32 位 MCU**  
**ES32W3120**

# 参 考 手 册

- 产品简介
- 数据手册
- 参考手册

上海东软载波微电子有限公司

2021-5-6

## 目 录

### 内容目录

<b>第 1 章</b>	<b>文档约定</b> .....	<b>47</b>
1.1	寄存器读写权限的设定 .....	47
<b>第 2 章</b>	<b>系统概述</b> .....	<b>48</b>
2.1	概述 .....	48
2.2	系统框图 .....	48
2.3	模块功能类别.....	49
2.3.1	ARM 32 位 Cortex-M3 内核模块 .....	50
2.3.2	存储器及存储器接口 .....	50
2.3.3	系统模块.....	51
2.3.4	时钟管理.....	51
2.3.5	外部接口.....	51
2.3.6	安全管理及运算加速 .....	52
2.3.7	模拟 .....	52
2.3.8	定时器 .....	52
2.3.9	通信 .....	54
<b>第 3 章</b>	<b>芯片配置指引</b> .....	<b>56</b>
3.1	概述 .....	56
3.2	ARM Cortex-M3 内核配置.....	56
3.2.1	ARM Cortex-M3 内核.....	56
3.2.2	总线.....	56
3.2.3	系统节拍定时器.....	56
3.2.4	调试器件.....	57
3.3	嵌套向量中断控制器 .....	57
3.3.1	中断优先级 .....	57
3.3.2	中断向量分配 .....	57
3.4	异步唤醒中断和事件.....	60
3.4.1	异步中断唤醒源.....	60
3.4.2	事件唤醒.....	61
3.5	存储器及存储器接口 .....	62
3.5.1	系统总线和存储器 .....	62
3.5.2	存储器系统控制配置 .....	62
3.5.3	扩展外部存储器配置 .....	63
3.6	系统模块配置.....	63
3.6.1	系统配置控制器配置 .....	63
3.6.2	电源管理及低功耗模式配置 .....	63
3.6.3	复位管理配置 .....	63
3.6.4	DMA 控制器配置 .....	64
3.6.5	外设互联(PIS)配置.....	67
3.6.6	独立看门狗定时器配置.....	67
3.6.6.1	独立看门狗定时器的时钟.....	67
3.6.6.2	独立看门狗定时器的低功耗动作模式.....	67

3.6.7	窗口看门狗定时器配置.....	67
3.6.7.1	窗口看门狗定时器的时钟.....	68
3.6.7.2	窗口看门狗定时器的低功耗动作模式.....	68
3.6.8	时钟管理配置.....	68
3.6.8.1	LOSC 的低功耗动作模式.....	68
3.6.8.2	HOSC 的低功耗动作模式.....	69
3.6.8.3	HRC 的低功耗动作模式.....	69
3.6.8.4	LRC 的低功耗动作模式.....	69
3.6.8.5	ULRC 的低功耗动作模式.....	69
3.7	外部接口配置.....	69
3.7.1	通用 IO 及端口控制配置.....	69
3.8	安全管理及运算加速配置.....	70
3.8.1	循环冗余校验(CRC)配置.....	70
3.8.2	加密处理配置.....	70
3.8.3	真随机数发生器配置.....	70
3.8.4	运算加速器配置.....	70
3.9	模拟配置.....	70
3.9.1	ADC 控制配置.....	70
3.9.1.1	ADC 转换通道配置.....	71
3.9.1.2	ADC 电源及参考电压.....	71
3.9.1.3	ADC 的时钟.....	71
3.9.1.4	ADC 的低功耗动作模式.....	71
3.10	定时器配置.....	71
3.10.1	高级定时器配置.....	71
3.10.1.1	高级定时器例化说明.....	72
3.10.1.2	高级定时器的时钟.....	72
3.10.1.3	高级定时器的低功耗动作模式.....	72
3.10.2	通用定时器配置.....	72
3.10.2.1	通用定时器例化说明.....	72
3.10.2.2	通用定时器的时钟.....	72
3.10.2.3	通用定时器的低功耗动作模式.....	73
3.10.3	基本定时器配置.....	73
3.10.3.1	基本定时器例化说明.....	73
3.10.3.2	基本定时器的时钟.....	73
3.10.3.3	基本定时器的低功耗动作模式.....	73
3.10.4	低功耗定时器配置.....	73
3.10.4.1	低功耗定时器例化说明.....	74
3.10.4.2	低功耗定时器的时钟.....	74
3.10.4.3	低功耗定时器的低功耗动作模式.....	74
3.10.5	RTC 定时器配置.....	74
3.10.5.1	RTC 的时钟.....	75
3.10.5.2	RTC 的低功耗动作模式.....	75
3.11	通信配置.....	75
3.11.1	I2C 接口配置.....	75

3.11.1.1	I2C 接口的时钟 .....	75
3.11.1.2	I2C 接口的低功耗动作模式 .....	75
3.11.2	串行外设接口(SPI)配置 .....	76
3.11.2.1	串行外设接口(SPI)的时钟 .....	76
3.11.2.2	串行外设接口(SPI)的低功耗动作模式 .....	76
3.11.3	通用异步收发器(UART) .....	76
3.11.3.1	通用异步收发器(UART)的时钟 .....	76
3.11.3.2	通用异步收发器(UART)的低功耗动作模式 .....	76
<b>第 4 章</b>	<b>系统总线和存储器 .....</b>	<b>77</b>
4.1	概述 .....	77
4.2	系统总线 .....	77
4.2.1	S0: I 总线 .....	77
4.2.2	S1: D 总线 .....	78
4.2.3	S2: S 总线 .....	78
4.2.4	S3: DMA 总线 .....	78
4.2.5	总线矩阵 .....	78
4.2.6	AHB/APB 总线桥 .....	78
4.3	存储器的组织结构 .....	78
4.3.1	系统存储器映射 .....	78
4.3.2	FLASH 存储器映射 .....	79
4.3.3	SRAM 存储器映射 .....	80
4.3.4	外设存储映射 .....	80
4.3.5	私有外设存储器映射 .....	83
4.3.6	位带 Bitband .....	83
4.3.6.1	SRAM 位带扩展 .....	83
4.3.6.2	外设位带扩展 .....	83
4.4	启动引导 .....	84
<b>第 5 章</b>	<b>存储器系统控制 .....</b>	<b>86</b>
5.1	概述 .....	86
5.2	特性 .....	86
5.3	结构图 .....	86
5.4	功能描述 .....	86
5.4.1	Flash 写保护 .....	86
5.4.2	Flash 私有代码读保护 .....	87
5.4.3	Flash 全局读保护 .....	87
5.4.4	Data Flash 划分 .....	88
5.4.5	Flash 操作请求流程 .....	88
5.4.6	Flash 程序区全擦除模式 .....	89
5.4.7	Flash 页擦除模式 .....	90
5.4.8	Flash 双字编程模式 .....	91
5.4.9	Flash 快速编程模式 .....	92
5.4.10	自编程硬件固化函数 .....	95
5.4.10.1	IAP_PageErase .....	95
5.4.10.2	IAP_DWordProgram .....	95

5.4.10.3	IAP_FastProgram.....	96
5.4.10.4	IAP_PageErase_DF.....	96
5.4.10.5	IAP_DWordProgram_DF.....	97
5.4.10.6	IAP_FastProgram_DF.....	97
5.4.10.7	IAP_EraseAll.....	98
5.5	特殊功能寄存器.....	98
5.5.1	寄存器列表.....	98
5.5.2	寄存器描述.....	99
5.5.2.1	Flash 控制状态寄存器 (MSC_IAPC).....	99
5.5.2.2	Flash 擦除编程地址寄存器 (MSC_IAPA).....	100
5.5.2.3	Flash 控制命令寄存器 (MSC_IAPT).....	101
5.5.2.4	Flash 控制解锁寄存器 (MSC_IAPUL).....	102
5.5.2.5	Flash 状态寄存器 (MSC_IAPSR).....	103
5.5.2.6	Flash 编程数据低位寄存器 (MSC_IAPDL).....	104
5.5.2.7	Flash 编程数据高位寄存器 (MSC_IAPDH).....	105
5.5.2.8	存储器读取等待时间寄存器 (MSC_MEMWAIT).....	105
5.5.2.9	Flash 修复信息状态寄存器 (MSC_FREPSR).....	106
<b>第 6 章</b>	<b>系统配置控制器.....</b>	<b>107</b>
6.1	概述.....	107
6.2	特性.....	107
6.3	功能描述.....	107
6.3.1	系统寄存器写保护.....	107
6.3.2	存储器重映射.....	107
6.3.3	定时器刹车源配置.....	107
6.4	特殊功能寄存器.....	108
6.4.1	寄存器列表.....	108
6.4.2	寄存器描述.....	109
6.4.2.1	系统写保护寄存器 (SYSCFG_PROT).....	109
6.4.2.2	存储器重映射寄存器 (SYSCFG_MEMRMP).....	109
6.4.2.3	Flash 读保护区数据访问标志寄存器 (SYSCFG_RDPF).....	110
6.4.2.4	TIMx 刹车触发控制寄存器 (SYSCFG_TIMBRK).....	111
6.4.2.5	芯片出厂配置寄存器 (CFG_CHIP).....	111
6.4.2.6	芯片启动配置寄存器 (CFG_BOOT).....	112
6.4.2.7	芯片配置字寄存器 (CFG_WORD0).....	112
6.4.2.8	写保护区 0 配置寄存器 (CFG_WRPT0).....	114
6.4.2.9	写保护区 1 配置寄存器 (CFG_WRPT1).....	114
6.4.2.10	数据 eFlash 配置寄存器 (CFG_DAFLS).....	115
6.4.2.11	全局读保护配置寄存器 (CFG_GBRDP).....	116
6.4.2.12	晶振频率调校配置寄存器 (CFG_XTAL).....	116
6.4.2.13	私有代码读保护区 0 配置寄存器 (CFG_PENCRY0).....	117
6.4.2.14	私有代码读保护区 1 配置寄存器 (CFG_PENCRY1).....	118
<b>第 7 章</b>	<b>电源管理及低功耗模式(PMU).....</b>	<b>119</b>
7.1	概述.....	119
7.2	特性.....	119

7.3	芯片电源.....	120
7.3.1	主系统电源域.....	120
7.3.2	备份域电源.....	120
7.4	电源监视.....	121
7.4.1	上电复位 POR.....	121
7.4.2	欠压复位 BOR.....	121
7.4.3	低电压检测 LVD.....	122
7.5	低功耗模式.....	123
7.5.1	低功耗模式转换.....	123
7.5.2	系统时钟速度.....	124
7.5.3	外设时钟门控.....	124
7.5.4	运行模式.....	125
7.5.5	SLEEP 模式.....	125
7.5.6	STOP1 模式.....	125
7.5.7	STOP2 模式.....	125
7.5.8	STANDBY 模式.....	125
7.5.9	低功耗模式下各模块状态.....	125
7.5.10	低功耗模式下唤醒源.....	128
7.5.11	低功耗模式设置.....	129
7.6	电源管理.....	132
7.6.1	内部 LDO 模式.....	132
7.6.2	内部 DC/DC 转换模式.....	132
7.6.3	供电模式切换.....	133
7.7	特殊功能寄存器.....	135
7.7.1	寄存器列表.....	135
7.7.2	寄存器描述.....	136
7.7.2.1	PMU 控制寄存器 (PMU_CR).....	136
7.7.2.2	PMU 状态寄存器 (PMU_SR).....	137
7.7.2.3	LVD 控制寄存器 (PMU_LVDCR).....	138
7.7.2.4	电源控制寄存器 (PMU_PWRCCR).....	139
7.7.2.5	唤醒时间控制寄存器 (PMU_TWUR).....	140
7.7.2.6	VDET 控制寄存器 (PMU_VDETCR).....	141
7.7.2.7	DCDC 控制寄存器 (PMU_DCCR).....	141
<b>第 8 章</b>	<b>备份电源域控制 (BKPC).....</b>	<b>144</b>
8.1	概述.....	144
8.2	特性.....	144
8.3	备份域电源管理.....	144
8.4	特殊功能寄存器.....	144
8.4.1	寄存器列表.....	144
8.4.1.1	备份电源域写保护寄存器 (BKPC_PROT).....	145
8.4.1.2	STANDBY 唤醒控制寄存器 (BKPC_WKR).....	145
8.4.1.3	STANDBY 唤醒状态寄存器 (BKPC_WKSR).....	146
8.4.1.4	备份电源域时钟控制寄存器 (BKPC_PCCR).....	146
8.4.1.5	备份电源域电源控制寄存器 (BKPC_PCR).....	148

<b>第 9 章</b>	<b>复位管理(RMU)</b> .....	<b>150</b>
9.1	概述 .....	150
9.2	特性 .....	150
9.3	结构框图 .....	150
9.4	系统复位 .....	151
9.4.1	硬件复位 .....	153
9.4.1.1	上电复位 .....	153
9.4.1.2	欠压复位 .....	154
9.4.1.3	端口复位 .....	155
9.4.1.4	看门狗复位 .....	155
9.4.1.5	LOCKUP 复位 .....	155
9.4.1.6	读取配置字错误复位 .....	155
9.4.2	软件复位 .....	155
9.4.2.1	芯片复位(CHIPRST) .....	155
9.4.2.2	CPU 复位(CPURST) .....	155
9.4.2.3	MCU 复位(MCURST) .....	155
9.4.2.4	外设软件复位 .....	155
9.5	特殊功能寄存器 .....	156
9.5.1	寄存器列表 .....	156
9.5.2	寄存器描述 .....	157
9.5.2.1	RMU 控制状态寄存器 (RMU_CSR) .....	157
9.5.2.2	RMU 复位状态寄存器 (RMU_RSTSR) .....	158
9.5.2.3	RMU 清复位状态寄存器 (RMU_CRSTSR) .....	160
9.5.2.4	AHB1 外设复位寄存器 (RMU_AHB1RSTR) .....	162
9.5.2.5	AHB2 外设复位寄存器 (RMU_AHB2RSTR) .....	163
9.5.2.6	APB1 外设复位寄存器 (RMU_APB1RSTR) .....	164
9.5.2.7	APB2 外设复位寄存器 (RMU_APB2RSTR) .....	166
9.5.2.8	APB3 外设复位寄存器 (RMU_APB3RSTR) .....	167
9.5.2.9	AHB2 BLE 复位寄存器 (RMU_AHB2BLERSTR) .....	168
<b>第 10 章</b>	<b>时钟管理(CMU)</b> .....	<b>169</b>
10.1	概述 .....	169
10.2	特性 .....	169
10.3	结构框图 .....	170
10.4	系统时钟 .....	170
10.4.1	外部高速振荡器时钟 HOSC .....	170
10.4.2	内部高速 RC 振荡器时钟 HRC .....	171
10.4.3	外部低速振荡器时钟 LOSC .....	171
10.4.4	内部低速 RC 振荡器时钟 LRC .....	171
10.4.5	内部超低速 RC 振荡器时钟 ULRC .....	171
10.4.6	内部倍频时钟 (PLL) .....	171
10.4.7	系统时钟选择 .....	171
10.4.8	高速振荡器安全管理 HOSM .....	171
10.4.9	低速振荡器安全管理 LOSM .....	172
10.4.10	RTC 时钟 .....	172

10.4.11	IWDT 时钟 .....	172
10.4.12	时钟输出选择 .....	172
10.5	特殊功能寄存器 .....	173
10.5.1	寄存器列表 .....	173
10.5.2	寄存器描述 .....	175
10.5.2.1	CMU 控制状态寄存器 (CMU_CSR) .....	175
10.5.2.2	CMU 配置寄存器 (CMU_CFGR) .....	177
10.5.2.3	CMU 时钟使能寄存器 (CMU_CLKENR) .....	179
10.5.2.4	CMU 时钟状态寄存器 (CMU_CLKSR) .....	181
10.5.2.5	PLL 配置寄存器 (CMU_PLLCFG) .....	183
10.5.2.6	HOSC 安全管理控制寄存器 (CMU_HOSMCR) .....	184
10.5.2.7	LOSC 安全管理控制寄存器 (CMU_LOSMCR) .....	185
10.5.2.8	PLL 失锁管理控制寄存器 (CMU_PULMCR) .....	186
10.5.2.9	CMU 时钟输出控制寄存器 (CMU_CLKOCR) .....	187
10.5.2.10	BUZZ 控制寄存器 (CMU_BUZZCR) .....	188
10.5.2.11	AHB1 外设时钟使能寄存器 (CMU_AHB1ENR) .....	189
10.5.2.12	APB1 外设时钟使能寄存器 (CMU_APB1ENR) .....	190
10.5.2.13	APB2 外设时钟使能寄存器 (CMU_APB2ENR) .....	192
10.5.2.14	时钟模块低功耗模式控制寄存器 (CMU_AHBLPENR) .....	193
10.5.2.15	外设时钟控制寄存器 (CMU_PERICR) .....	194
10.5.2.16	时钟模块 LDO 稳定时间控制寄存器 (CMU_CKLDOT) .....	195
10.5.2.17	HOSC 稳定时间控制寄存器 (CMU_HOSCT) .....	196
10.5.2.18	PLL 稳定时间控制寄存器 (CMU_PLLT) .....	197
10.5.2.19	时钟开启模式控制寄存器 (CMU_CKSEQR) .....	198
10.5.2.20	PDMPCM 时钟配置寄存器 (CMU_PDPCMCFG) .....	199
10.5.2.21	BLE 时钟使能寄存器 (CMU_BLEENR) .....	200
10.5.2.22	BLE 时钟配置寄存器 (CMU_BLECFG) .....	201
10.5.2.23	BLE 时钟选择寄存器 (CMU_BLECKS) .....	202
10.5.2.24	ULRC 时钟用户校准模式寄存器 (CMU_ULRCUMC) .....	203
10.5.2.25	LRC 时钟用户校准模式寄存器 0 (CMU_LRCUMC0) .....	204
10.5.2.26	LRC 时钟用户校准模式寄存器 1 (CMU_LRCUMC1) .....	205
10.5.2.27	LOSC 时钟用户校准模式寄存器 0 (CMU_LOSCUMC0) .....	206
10.5.2.28	LOSC 时钟用户校准模式寄存器 1 (CMU_LOSCUMC1) .....	207
10.5.2.29	HOSC 时钟用户校准模式寄存器 (CMU_HOSCUMC) .....	208
<b>第 11 章</b>	<b>DMA 控制器(DMA).....</b>	<b>209</b>
11.1	概述 .....	209
11.2	特性 .....	209
11.3	结构框图 .....	210
11.4	功能描述 .....	210
11.4.1	通道选择配置 .....	210
11.4.2	DMA 控制 .....	210
11.4.2.1	DMA 仲裁率 .....	210
11.4.2.2	优先级 .....	212
11.4.2.3	DMA 周期类型 .....	214



11.4.3	通道控制数据结构 .....	224
11.4.3.1	源数据结束指针 .....	224
11.4.3.2	目标数据结束指针 .....	225
11.4.3.3	控制数据配置 .....	225
11.4.3.4	地址计算 .....	228
11.5	特殊功能寄存器 .....	231
11.5.1	编程器模型介绍 .....	231
11.6	特殊功能寄存器 .....	232
11.6.1	寄存器列表 .....	232
11.6.2	寄存器描述 .....	233
11.6.2.1	DMA 状态寄存器 (DMA_STATUS) .....	233
11.6.2.2	DMA 配置寄存器 (DMA_CFG) .....	234
11.6.2.3	DMA 通道控制数据基指针寄存器 (DMA_CTRLBASE) .....	235
11.6.2.4	DMA 通道交替控制数据基指针寄存器 (DMA_ALTCTRLBASE) .....	236
11.6.2.5	DMA 通道等待请求状态寄存器 (DMA_CHWAITSTATUS) .....	236
11.6.2.6	DMA 通道软件请求寄存器 (DMA_CHSWREQ) .....	237
11.6.2.7	DMA 通道使用突发设置寄存器 (DMA_CHUSEBURSTSET) .....	238
11.6.2.8	DMA 通道使用突发清除寄存器 (DMA_CHUSEBURSTCLR) .....	239
11.6.2.9	DMA 通道请求屏蔽设置寄存器 (DMA_CHREQMASKSET) .....	239
11.6.2.10	DMA 通道请求屏蔽清除寄存器 (DMA_CHREQMASKCLR) .....	240
11.6.2.11	DMA 通道使能设置寄存器 (DMA_CHENSET) .....	241
11.6.2.12	DMA 通道使能清除寄存器 (DMA_CHENCLR) .....	242
11.6.2.13	DMA 通道主要-交替设置寄存器 (DMA_CHPRIALTSET) .....	243
11.6.2.14	DMA 通道主要-交替清除寄存器 (DMA_CHPRIALTCLR) .....	244
11.6.2.15	DMA 通道优先级设置寄存器 (DMA_CHPRSET) .....	245
11.6.2.16	DMA 通道优先级清除寄存器 (DMA_CHPRCLR) .....	246
11.6.2.17	DMA 总线错误清除寄存器 (DMA_ERRCLR) .....	247
11.6.2.18	DMA 中断状态寄存器 (DMA_IFLAG) .....	248
11.6.2.19	DMA 中断使能控制寄存器 (DMA_IER) .....	250
11.6.2.20	DMA 通道 0 复用选择寄存器 (DMA_CH0_SELCON) .....	252
11.6.2.21	DMA 通道 1 复用选择寄存器 (DMA_CH1_SELCON) .....	255
11.6.2.22	DMA 通道 2 复用选择寄存器 (DMA_CH2_SELCON) .....	258
11.6.2.23	DMA 通道 3 复用选择寄存器 (DMA_CH3_SELCON) .....	261
11.6.2.24	DMA 通道 4 复用选择寄存器 (DMA_CH4_SELCON) .....	264
11.6.2.25	DMA 通道 5 复用选择寄存器 (DMA_CH5_SELCON) .....	267
11.6.2.26	DMA 通道 6 复用选择寄存器 (DMA_CH6_SELCON) .....	270
11.6.2.27	DMA 通道 7 复用选择寄存器 (DMA_CH7_SELCON) .....	273
11.6.2.28	DMA 通道 8 复用选择寄存器 (DMA_CH8_SELCON) .....	276
11.6.2.29	DMA 通道 9 复用选择寄存器 (DMA_CH9_SELCON) .....	279
11.6.2.30	DMA 通道 10 复用选择寄存器 (DMA_CH10_SELCON) .....	282
11.6.2.31	DMA 通道 11 复用选择寄存器 (DMA_CH11_SELCON) .....	285
<b>第 12 章</b>	<b>外设互联(PIS).....</b>	<b>288</b>
12.1	概述 .....	288
12.2	特性 .....	288

12.3	结构框图.....	289
12.4	功能描述.....	289
12.4.1	生产端信号.....	290
12.4.2	消费端信号.....	290
12.4.3	PIS 通道选择.....	293
12.4.3.1	同一时钟域互联.....	294
12.4.3.2	APB1 和 APB2 外设之间互联.....	294
12.4.3.3	生产端信号为异步信号.....	294
12.4.4	UART 输出调制.....	294
12.5	特殊功能寄存器.....	296
12.5.1	寄存器列表.....	296
12.5.2	寄存器描述.....	297
12.5.2.1	PIS 通道 0 控制寄存器 (PIS_CH0_CON).....	297
12.5.2.2	PIS 通道 1 控制寄存器 (PIS_CH1_CON).....	300
12.5.2.3	PIS 通道 2 控制寄存器 (PIS_CH2_CON).....	303
12.5.2.4	PIS 通道 3 控制寄存器 (PIS_CH3_CON).....	306
12.5.2.5	PIS 通道 4 控制寄存器 (PIS_CH4_CON).....	309
12.5.2.6	PIS 通道 5 控制寄存器 (PIS_CH5_CON).....	312
12.5.2.7	PIS 通道 6 控制寄存器 (PIS_CH6_CON).....	315
12.5.2.8	PIS 通道 7 控制寄存器 (PIS_CH7_CON).....	318
12.5.2.9	PIS 通道 8 控制寄存器 (PIS_CH8_CON).....	321
12.5.2.10	PIS 通道 9 控制寄存器 (PIS_CH9_CON).....	324
12.5.2.11	PIS 通道 10 控制寄存器 (PIS_CH10_CON).....	327
12.5.2.12	PIS 通道 11 控制寄存器 (PIS_CH11_CON).....	330
12.5.2.13	PIS 通道 12 控制寄存器 (PIS_CH12_CON).....	333
12.5.2.14	PIS 通道 13 控制寄存器 (PIS_CH13_CON).....	336
12.5.2.15	PIS 通道 14 控制寄存器 (PIS_CH14_CON).....	339
12.5.2.16	PIS 通道 15 控制寄存器 (PIS_CH15_CON).....	342
12.5.2.17	PIS 通道端口输出使能寄存器 (PIS_CH_OER).....	345
12.5.2.18	PIS 消费端通道控制寄存器 0 (PIS_TAR_CON0).....	346
12.5.2.19	PIS 消费端通道控制寄存器 1 (PIS_TAR_CON1).....	349
12.5.2.20	UART0 输出调制控制寄存器 (UART0_TXMCR).....	350
12.5.2.21	UART1 输出调制控制寄存器 (UART1_TXMCR).....	351
12.5.2.22	UART2 输出调制控制寄存器 (UART2_TXMCR).....	352
<b>第 13 章</b>	<b>独立看门狗(IWDT).....</b>	<b>353</b>
13.1	概述.....	353
13.2	特性.....	353
13.3	功能描述.....	353
13.3.1	硬件看门狗.....	354
13.3.2	软件看门狗.....	354
13.3.3	调试模式.....	355
13.3.4	寄存器访问保护.....	355
13.4	特殊功能寄存器.....	355
13.4.1	寄存器列表.....	355

13.4.2	寄存器描述.....	356
13.4.2.1	IWDT 计数器装载值寄存器 (IWDT_LOAD) .....	356
13.4.2.2	IWDT 计数器当前值寄存器 (IWDT_VALUE) .....	356
13.4.2.3	IWDT 控制寄存器 (IWDT_CON) .....	357
13.4.2.4	IWDT 中断标志清除寄存器 (IWDT_INTCLR) .....	358
13.4.2.5	IWDT 中断标志寄存器 (IWDT_RIS) .....	358
13.4.2.6	IWDT 锁定寄存器 (IWDT_LOCK) .....	359
<b>第 14 章</b>	<b>窗口看门狗(WWDT).....</b>	<b>360</b>
14.1	概述 .....	360
14.2	特性 .....	360
14.3	功能描述.....	360
14.3.1	窗口看门狗.....	360
14.3.2	调试模式.....	362
14.3.3	寄存器访问保护.....	362
14.4	特殊功能寄存器 .....	363
14.4.1	寄存器列表.....	363
14.4.2	寄存器描述.....	364
14.4.2.1	WWDT 计数器装载值寄存器 (WWDT_LOAD) .....	364
14.4.2.2	WWDT 计数器当前值寄存器 (WWDT_VALUE) .....	364
14.4.2.3	WWDT 控制寄存器 (WWDT_CON) .....	365
14.4.2.4	WWDT 中断标志清除寄存器 (WWDT_INTCLR) .....	366
14.4.2.5	WWDT 中断标志寄存器 (WWDT_RIS) .....	366
14.4.2.6	WWDT 锁定寄存器 (WWDT_LOCK) .....	367
<b>第 15 章</b>	<b>通用 IO 及端口控制(GPIO).....</b>	<b>368</b>
15.1	概述 .....	368
15.2	特性 .....	368
15.3	结构框图.....	369
15.4	功能描述.....	369
15.4.1	端口控制寄存器.....	369
15.4.2	端口数据寄存器.....	370
15.4.3	外部端口中断 .....	370
15.4.4	通用 GPIO 配置.....	370
15.4.5	外部中断与唤醒.....	371
15.4.6	外设功能端口复用 .....	372
15.4.7	GPIO 锁定.....	372
15.4.8	GPIO 输入配置 .....	373
15.4.9	GPIO 输出配置 .....	373
15.4.10	模拟输入配置 .....	373
15.5	特殊功能寄存器 .....	374
15.5.1	寄存器列表.....	374
15.5.2	寄存器描述.....	375
15.5.2.1	GPIO 端口输入数据寄存器 (GPIO_DIN) .....	375
15.5.2.2	GPIO 端口输出数据寄存器 (GPIO_DOUT) .....	375
15.5.2.3	GPIO 端口置位和复位寄存器 (GPIO_BSRR) .....	376

15.5.2.4	GPIO 端口翻转寄存器 (GPIO_BIR)	376
15.5.2.5	GPIO 端口模式寄存器 (GPIO_MODE)	377
15.5.2.6	GPIO 端口开漏和开源寄存器 (GPIO_ODOS)	377
15.5.2.7	GPIO 端口上拉和下拉寄存器 (GPIO_PUPD)	378
15.5.2.8	GPIO 端口输出驱动寄存器 (GPIO_ODRV)	378
15.5.2.9	GPIO 端口滤波寄存器 (GPIO_FLT)	379
15.5.2.10	GPIO 端口类型寄存器 (GPIO_TYPE)	379
15.5.2.11	GPIO 端口复用功能寄存器 0 (GPIO_FUNC0)	380
15.5.2.12	GPIO 端口复用功能寄存器 1 (GPIO_FUNC1)	381
15.5.2.13	GPIO 端口锁定寄存器 (GPIO_LOCK)	382
15.5.2.14	外部中断上升沿触发使能寄存器 (GPIO_EXTIRER)	383
15.5.2.15	外部中断下降沿触发使能寄存器 (GPIO_EXTIFER)	383
15.5.2.16	外部中断使能寄存器 (GPIO_EXTIEN)	384
15.5.2.17	外部中断标志寄存器 (GPIO_EXTIFLAG)	384
15.5.2.18	外部中断标志置位寄存器 (GPIO_EXTISFR)	385
15.5.2.19	外部中断标志清零寄存器 (GPIO_EXTICFR)	385
15.5.2.20	外部中断端口选择寄存器 0 (GPIO_EXTIPSR0)	386
15.5.2.21	外部中断端口选择寄存器 1 (GPIO_EXTIPSR1)	388
15.5.2.22	外部中断滤波控制寄存器 (GPIO_EXTIFLTCR)	390
<b>第 16 章</b>	<b>循环冗余校验(CRC)</b>	<b>391</b>
16.1	概述	391
16.2	特性	391
16.3	结构框图	391
16.4	功能描述	392
16.4.1	常规操作	392
16.4.2	DMA 请求	392
16.5	特殊功能寄存器	394
16.5.1	寄存器列表	394
16.5.2	寄存器描述	395
16.5.2.1	CRC 控制寄存器 (CRC_CR)	395
16.5.2.2	CRC 写数据寄存器 (CRC_DATA)	396
16.5.2.3	CRC 种子寄存器 (CRC_SEED)	397
16.5.2.4	CRC 校验值寄存器 (CRC_CHECKSUM)	397
<b>第 17 章</b>	<b>加密处理(CRYPT)</b>	<b>398</b>
17.1	概述	398
17.2	特性	398
17.3	AES 功能描述	399
17.3.1	AES-ECB 模式加密	399
17.3.2	AES-ECB 模式解密	400
17.3.3	AES-CBC 模式加密	401
17.3.4	AES-CBC 模式解密	402
17.3.5	AES-CTR 模式加解密	404
17.3.6	AES-GCM 模式加解密	406
17.4	DES 功能描述	411

17.4.1	DES/TDES-ECB 模式加密	411
17.4.2	DES/TDES-ECB 模式解密	412
17.4.3	DES/TDES-CBC 模式加密	413
17.4.4	DES/TDES-CBC 模式解密	414
17.5	SM4 功能描述	416
17.5.1	SM4-ECB 模式加密	416
17.5.2	SM4-ECB 模式解密	417
17.5.3	SM4-CBC 模式加密	418
17.5.4	SM4-CBC 模式解密	418
17.6	FIFO 使用	419
17.7	使用 DMA 传输数据	419
17.8	特殊功能寄存器	420
17.8.1	寄存器列表	420
17.8.2	寄存器描述	421
17.8.2.1	CRYPT 数据寄存器 X (CRYPT_DATAx) (x = 0..3)	421
17.8.2.2	CRYPT 密钥寄存器 X (CRYPT_KEYX) (x = 0..7)	421
17.8.2.3	CRYPT 初始向量寄存器 X (CRYPT_IVx) (x = 0..3)	422
17.8.2.4	CRYPT 结果寄存器 X (CRYPT_RESx) (x = 0..3)	422
17.8.2.5	CRYPT 控制寄存器 (CRYPT_CON)	423
17.8.2.6	CRYPT 中断标志寄存器 (CRYPT_IF)	425
17.8.2.7	CRYPT 状态清零寄存器 (CRYPT_IFC)	426
17.8.2.8	CRYPT 缓存寄存器 (CRYPT_FIFO)	426
<b>第 18 章</b>	<b>HASH 硬件加速模块(HASH)</b>	<b>427</b>
18.1	概述	427
18.2	工作原理	427
18.3	HASH 结构框图	428
18.4	特殊功能寄存器	429
18.4.1	寄存器列表	429
18.4.2	寄存器描述	430
18.4.2.1	HASH 消息输入寄存器 (HASH_DIN)	430
18.4.2.2	HASH 控制寄存器 (HASH_CTRL)	430
18.4.2.3	HASH 状态寄存器 (HASH_STAT)	431
18.4.2.4	HASH 结果输出寄存器 0 (HASH_RES0)	432
18.4.2.5	HASH 结果输出寄存器 1 (HASH_RES1)	432
18.4.2.6	HASH 结果输出寄存器 2 (HASH_RES2)	432
18.4.2.7	HASH 结果输出寄存器 3 (HASH_RES3)	433
18.4.2.8	HASH 结果输出寄存器 4 (HASH_RES4)	433
18.4.2.9	HASH 结果输出寄存器 5 (HASH_RES5)	433
18.4.2.10	HASH 结果输出寄存器 6 (HASH_RES6)	434
18.4.2.11	HASH 结果输出寄存器 7 (HASH_RES7)	434
<b>第 19 章</b>	<b>椭圆曲线密码单元(ECC)</b>	<b>435</b>
19.1	概述	435
19.2	ECC 结构框图	435
19.3	ECC 缓存地址映射	436

19.4	特殊功能寄存器 .....	436
19.4.1	寄存器列表 .....	436
19.4.2	寄存器描述 .....	437
19.4.2.1	ECC 控制寄存器 (ECC_CON) .....	437
19.4.2.2	ECC 状态寄存器 (ECC_STA) .....	438
19.4.2.3	ECC 运算参数寄存器 (ECC_PARA) .....	439
19.4.2.4	ECC 标量数据寄存器 (ECC_SCALAR <sub>x</sub> ) (x = 0..7) .....	439
19.4.2.5	ECC 标量数据寄存器 (ECC_YCO <sub>x</sub> ) (x = 0..7) .....	439
<b>第 20 章</b>	<b>真随机发生器(TRNG).....</b>	<b>440</b>
20.1	概述 .....	440
20.2	特性 .....	440
20.3	结构框图 .....	440
20.4	功能描述 .....	440
20.4.1	初始化时间 .....	440
20.4.2	错误管理 .....	440
20.4.3	随机数生成时间 .....	441
20.4.4	随机数种子 .....	441
20.4.5	操作流程 .....	441
20.5	特殊功能寄存器 .....	442
20.5.1	寄存器列表 .....	442
20.5.2	寄存器描述 .....	443
20.5.2.1	TRNG 控制寄存器 (TRNG_CR) .....	443
20.5.2.2	TRNG 状态寄存器 (TRNG_SR) .....	444
20.5.2.3	TRNG 数据寄存器 (TRNG_DR) .....	445
20.5.2.4	TRNG 种子寄存器 (TRNG_SEED) .....	445
20.5.2.5	TRNG 初始化时间寄存器 (TRNG_CFGR) .....	446
20.5.2.6	TRNG 中断使能寄存器 (TRNG_IER) .....	446
20.5.2.7	TRNG 中断事件寄存器 (TRNG_IFR) .....	447
20.5.2.8	TRNG 中断事件清零寄存器 (TRNG_IFCR) .....	448
20.5.2.9	TRNG 状态事件寄存器 (TRNG_ISR) .....	448
<b>第 21 章</b>	<b>运算加速器(CALC).....</b>	<b>450</b>
21.1	概述 .....	450
21.2	特性 .....	450
21.3	结构框图 .....	450
21.4	平方根运算 .....	450
21.4.1	算法概述 .....	450
21.4.2	使用说明 .....	450
21.4.3	完成时间 .....	451
21.5	除法运算 .....	453
21.5.1	算法概述 .....	453
21.5.2	特例说明 .....	453
21.5.3	使用说明 .....	453
21.5.4	完成时间 .....	454
21.6	特殊功能寄存器 .....	455

21.6.1	寄存器列表	455
21.6.2	寄存器描述	456
21.6.2.1	平方根运算状态寄存器 (CALC_SQRTSR)	456
21.6.2.2	被开方数寄存器 (CALC_RDCND)	456
21.6.2.3	平方根运算结果寄存器 (CALC_SQRTRES)	457
21.6.2.4	被除数寄存器 (CALC_DIVDR)	457
21.6.2.5	除数寄存器 (CALC_DIVSR)	457
21.6.2.6	商寄存器 (CALC_DIVQR)	458
21.6.2.7	余数寄存器 (CALC_DIVRR)	458
21.6.2.8	除法运算控制状态寄存器 (CALC_DIVCSR)	459
<b>第 22 章</b>	<b>模数转换器(ADC)</b>	<b>460</b>
22.1	概述	460
22.2	特性	460
22.3	结构框图	461
22.4	功能描述	461
22.4.1	ADC 控制	461
22.4.2	ADC 时钟	461
22.4.3	通道选择	462
22.4.4	单次转换模式	462
22.4.5	连续转换模式	462
22.4.6	时序图	463
22.4.7	模拟看门狗	463
22.4.8	通道扫描	464
22.4.9	插入通道控制	464
22.4.10	不连续采样控制	465
22.4.11	数据对齐	466
22.4.12	可独立设置各通道采样时间	466
22.4.13	外部触发转换和触发极性	466
22.4.14	快速转换模式 (不支持)	467
22.4.15	软件补偿	467
22.5	数据管理	467
22.5.1	使用 DMA	467
22.5.2	在不使用 DMA 的情况下管理转换序列	467
22.5.3	在不使用 DMA 和溢出检测情况下进行转换	468
22.6	ADC 中断	468
22.7	特殊功能寄存器	469
22.7.1	寄存器列表	469
22.7.2	寄存器描述	470
22.7.2.1	ADC 状态寄存器 (ADC_STAT)	470
22.7.2.2	ADC 清零寄存器 (ADC_CLR)	471
22.7.2.3	ADC 控制寄存器 0 (ADC_CON0)	472
22.7.2.4	ADC 控制寄存器 1 (ADC_CON1)	473
22.7.2.5	ADC 采样时间寄存器 1 (ADC_SMPT1)	475
22.7.2.6	ADC 采样时间寄存器 2 (ADC_SMPT2)	475

22.7.2.7	ADC 采样时间寄存器 3 (ADC_SMPT3)	476
22.7.2.8	ADC 插入通道数据偏移寄存器 1 (ADC_ICHOFF1)	476
22.7.2.9	ADC 插入通道数据偏移寄存器 2 (ADC_ICHOFF2)	477
22.7.2.10	ADC 插入通道数据偏移寄存器 3 (ADC_ICHOFF3)	477
22.7.2.11	ADC 插入通道数据偏移寄存器 4 (ADC_ICHOFF4)	477
22.7.2.12	ADC 标准通道序列寄存器 1 (ADC_NCHS1)	478
22.7.2.13	ADC 标准通道序列寄存器 2 (ADC_NCHS2)	479
22.7.2.14	ADC 标准通道序列寄存器 3 (ADC_NCHS3)	479
22.7.2.15	ADC 标准通道序列寄存器 4 (ADC_NCHS4)	480
22.7.2.16	ADC 插入通道序列寄存器 (ADC_ICHS)	481
22.7.2.17	ADC 通道序列长度寄存器 (ADC_CHSL)	481
22.7.2.18	ADC 看门狗高阈值寄存器 (ADC_WDTH)	482
22.7.2.19	ADC 看门狗低阈值寄存器 (ADC_WDTL)	482
22.7.2.20	ADC 插入通道数据寄存器 1 (ADC_ICHDR1)	483
22.7.2.21	ADC 插入通道数据寄存器 2 (ADC_ICHDR2)	483
22.7.2.22	ADC 插入通道数据寄存器 3 (ADC_ICHDR3)	483
22.7.2.23	ADC 插入通道数据寄存器 4 (ADC_ICHDR4)	484
22.7.2.24	ADC 标准通道数据寄存器 (ADC_NCHDR)	484
22.7.2.25	ADC 通用控制寄存器 1 (ADC_CCR1)	485
22.7.2.26	ADC 通用控制寄存器 2 (ADC_CCR2)	487
<b>第 23 章</b>	<b>实时时钟(RTC)</b>	<b>488</b>
23.1	概述	488
23.2	特性	488
23.3	结构框图	489
23.4	功能描述	489
23.4.1	时钟和预分频	489
23.4.2	时钟和日历	490
23.4.3	可编程闹钟	490
23.4.4	周期性唤醒	490
23.4.5	数字校准	491
23.4.6	时间戳功能	491
23.4.7	侵入检测功能	492
23.4.8	时钟输出	492
23.4.9	自动温度补偿 (本设计中不支持 RTC 自动温度补偿)	492
23.5	基本配置	492
23.5.1	RTC 写保护	492
23.5.2	RTC 校准写保护	493
23.5.3	时间和日历初始化	493
23.5.4	夏令时	494
23.5.5	亚秒调整	494
23.6	RTC 中断	494
23.7	特殊功能寄存器	495
23.7.1	寄存器列表	495
23.7.2	寄存器描述	496



23.7.2.1	RTC 写保护寄存器 (RTC_WPR)	496
23.7.2.2	RTC 控制寄存器 (RTC_CON)	496
23.7.2.3	RTC 预分频寄存器 (RTC_PSR)	499
23.7.2.4	RTC 侵入控制寄存器 (RTC_TAMPCON)	499
23.7.2.5	RTC 时间寄存器 (RTC_TIME)	500
23.7.2.6	RTC 日期寄存器 (RTC_DATE)	501
23.7.2.7	RTC 亚秒寄存器 (RTC_SSEC)	502
23.7.2.8	RTC 唤醒匹配寄存器 (RTC_WUMAT)	502
23.7.2.9	RTC 闹钟 A 寄存器 (RTC_ALMA)	502
23.7.2.10	RTC 闹钟 B 寄存器 (RTC_ALMB)	504
23.7.2.11	RTC 闹钟 A 亚秒寄存器 (RTC_ALMASSEC)	505
23.7.2.12	RTC 闹钟 B 亚秒寄存器 (RTC_ALMBSSEC)	506
23.7.2.13	RTC 时间戳时间寄存器 (RTC_TSTIME)	506
23.7.2.14	RTC 时间戳日期寄存器 (RTC_TSDATE)	507
23.7.2.15	RTC 时间戳亚秒寄存器 (RTC_TSSSEC)	508
23.7.2.16	RTC 亚秒调整寄存器 (RTC_SSECTR)	508
23.7.2.17	RTC 中断使能寄存器 (RTC_IER)	509
23.7.2.18	RTC 中断标志寄存器 (RTC_IFR)	510
23.7.2.19	RTC 中断标志清零寄存器 (RTC_IFCR)	512
23.7.2.20	RTC 中断状态寄存器 (RTC_ISR)	513
23.7.2.21	RTC 校准写保护寄存器 (RTC_CALWPR)	515
23.7.2.22	RTC 校准控制寄存器 (RTC_CALCON)	515
23.7.2.23	RTC 校准寄存器 (RTC_CALDR)	517
23.7.2.24	RTC 备份寄存器 (RTC_BKPRx) (x = 0..31)	517
<b>第 24 章</b>	<b>高级定时器(AD16C4T)</b>	<b>519</b>
24.1	概述	519
24.2	特性	519
24.3	结构框图	520
24.4	功能描述	520
24.4.1	预分频器	520
24.4.2	重复计数器	522
24.4.3	时钟源	523
24.4.3.1	内部时钟源 (INT_CLK)	523
24.4.3.2	外部时钟源 1	523
24.4.3.3	外部时钟源 2	524
24.4.3.4	内部触发输入 (ITn)	525
24.4.4	计数模式	525
24.4.4.1	递增计数模式	525
24.4.4.2	递减计数模式	527
24.4.4.3	中心对齐模式	528
24.4.5	捕获/比较通道	529
24.4.6	输入捕获模式	531
24.4.6.1	PWM 输入模式	532
24.4.7	PWM 模式	532

24.4.7.1	PWM 边沿对齐模式 .....	533
24.4.7.2	PWM 中心对齐模式 .....	533
24.4.8	输出比较模式 .....	534
24.4.8.1	外部事件清除比较输出 .....	535
24.4.8.2	强制输出模式 .....	535
24.4.9	单脉冲模式 .....	536
24.4.10	互补输出与死区时间 .....	537
24.4.11	刹车功能 .....	538
24.4.12	编码器接口模式 .....	540
24.4.13	输入异或功能 .....	542
24.4.14	霍尔传感器接口 .....	542
24.4.15	外部触发的同步 .....	542
24.4.15.1	复位模式 .....	542
24.4.15.2	门控模式 .....	543
24.4.15.3	触发模式 .....	544
24.4.15.4	选择外部时钟源 2 的触发模式 .....	545
24.4.16	调试模式 .....	546
24.4.17	6 步 PWM 生成 .....	546
24.5	特殊功能寄存器 .....	547
24.5.1	寄存器列表 .....	547
24.5.2	寄存器描述 .....	548
24.5.2.1	控制寄存器 1 (AD16C4Tn_CON1) .....	548
24.5.2.2	控制寄存器 2 (AD16C4Tn_CON2) .....	551
24.5.2.3	从模式控制寄存器 (AD16C4Tn_SMCON) .....	553
24.5.2.4	中断使能寄存器 (AD16C4Tn_IER) .....	556
24.5.2.5	中断禁止寄存器 (AD16C4Tn_IDR) .....	557
24.5.2.6	中断有效状态寄存器 (AD16C4Tn_IVS) .....	559
24.5.2.7	原始中断标志寄存器 (AD16C4Tn_RIF) .....	561
24.5.2.8	中断标志屏蔽寄存器 (AD16C4Tn_IFM) .....	564
24.5.2.9	中断清零寄存器 (AD16C4Tn_ICR) .....	567
24.5.2.10	软件生成事件寄存器 (AD16C4Tn_SGE) .....	569
24.5.2.11	通道捕获模式寄存器 1 (AD16C4Tn_CHMR1) .....	571
24.5.2.12	通道捕获模式寄存器 2 (AD16C4Tn_CHMR2) .....	576
24.5.2.13	捕获/比较使能极性寄存器 (AD16C4Tn_CCEP) .....	579
24.5.2.14	计数寄存器 (AD16C4Tn_COUNT) .....	581
24.5.2.15	预分频寄存器 (AD16C4Tn_PRES) .....	582
24.5.2.16	计数器自动装载寄存器 (AD16C4Tn_AR) .....	582
24.5.2.17	重复计数寄存器 (AD16C4Tn_REPAR) .....	583
24.5.2.18	通道捕获/比较寄存器 1 (AD16C4Tn_CCVAL1) .....	584
24.5.2.19	通道捕获/比较寄存器 2 (AD16C4Tn_CCVAL2) .....	584
24.5.2.20	通道捕获/比较寄存器 3 (AD16C4Tn_CCVAL3) .....	585
24.5.2.21	通道捕获/比较寄存器 4 (AD16C4Tn_CCVAL4) .....	585
24.5.2.22	刹车和死区配置寄存器 (AD16C4Tn_BDCFG) .....	586
24.5.2.23	DMA 使能寄存器 (AD16C4Tn_DMAEN) .....	589

<b>第 25 章</b>	<b>通用定时器(GP16C4T)</b> .....	<b>590</b>
25.1	概述 .....	590
25.2	特性 .....	590
25.3	结构框图.....	591
25.4	功能描述.....	591
25.4.1	预分频器.....	591
25.4.2	时钟源 .....	593
25.4.2.1	内部时钟源 (INT_CLK) .....	593
25.4.2.2	外部时钟源 1.....	593
25.4.2.3	外部时钟源 2.....	594
25.4.2.4	内部触发输入 (ITn) .....	595
25.4.3	计数器模式.....	596
25.4.3.1	递增计数模式.....	596
25.4.3.2	递减计数模式.....	598
25.4.3.3	中心对齐模式.....	599
25.4.4	捕获/比较通道 .....	600
25.4.5	输入捕获模式 .....	602
25.4.5.1	PWM 输入模式 .....	603
25.4.6	PWM 模式.....	604
25.4.6.1	PWM 边沿对齐模式.....	604
25.4.6.2	PWM 中心对齐模式.....	605
25.4.7	输出比较模式.....	606
25.4.7.1	外部事件清除比较输出 .....	606
25.4.8	单脉冲模式.....	607
25.4.9	编码器接口模式.....	608
25.4.10	输入异或功能 .....	610
25.4.11	定时器和外部触发的同步 .....	610
25.4.11.1	复位模式.....	610
25.4.11.2	门控模式.....	611
25.4.11.3	触发模式.....	611
25.4.11.4	选择外部时钟源 2 的触发模式.....	612
25.4.12	调试模式.....	613
25.5	特殊功能寄存器 .....	613
25.5.1	寄存器列表 .....	613
25.5.2	寄存器描述 .....	615
25.5.2.1	控制寄存器 1 (GP16C4Tn_CON1) .....	615
25.5.2.2	控制寄存器 2 (GP16C4Tn_CON2) .....	617
25.5.2.3	从模式控制寄存器 (GP16C4Tn_SMCON) .....	619
25.5.2.4	中断使能寄存器 (GP16C4Tn_IER) .....	622
25.5.2.5	中断禁止寄存器 (GP16C4Tn_IDR) .....	623
25.5.2.6	中断有效状态寄存器 (GP16C4Tn_IVS) .....	624
25.5.2.7	原始中断标志寄存器 (GP16C4Tn_RIF) .....	625
25.5.2.8	中断标志屏蔽寄存器 (GP16C4Tn_IFM) .....	628
25.5.2.9	中断清零寄存器 (GP16C4Tn_ICR) .....	630

	25.5.2.10	软件生成事件寄存器 (GP16C4Tn_SGE)	631
	25.5.2.11	捕获/比较模式寄存器 1 (GP16C4Tn_CHMR1)	632
	25.5.2.12	捕获/比较模式寄存器 2 (GP16C4Tn_CHMR2)	637
	25.5.2.13	捕获/比较使能寄存器 (GP16C4Tn_CCEP)	640
	25.5.2.14	计数器寄存器 (GP16C4Tn_COUNT)	641
	25.5.2.15	预分频寄存器 (GP16C4Tn_PRES)	642
	25.5.2.16	自动重载寄存器 (GP16C4Tn_AR)	642
	25.5.2.17	捕获/比较寄存器 1 (GP16C4Tn_CCVAL1)	643
	25.5.2.18	捕获/比较寄存器 2 (GP16C4Tn_CCVAL2)	643
	25.5.2.19	捕获/比较寄存器 3 (GP16C4Tn_CCVAL3)	644
	25.5.2.20	捕获/比较寄存器 4 (GP16C4Tn_CCVAL4)	644
	25.5.2.21	DMA 使能寄存器 (GP16C4Tn_DMAEN)	645
<b>第 26 章</b>		<b>基本定时器(BS16T)</b>	<b>646</b>
26.1		概述	646
26.2		特性	646
26.3		结构框图	646
26.4		功能描述	646
	26.4.1	预分频器	646
	26.4.2	时钟源	647
	26.4.3	递增计数模式	647
	26.4.4	调试模式	648
26.5		特殊功能寄存器	648
	26.5.1	寄存器列表	648
	26.5.2	寄存器描述	649
	26.5.2.1	控制寄存器 1 (BS16Tn_CON1)	649
	26.5.2.2	中断使能寄存器 (BS16Tn_IER)	650
	26.5.2.3	中断禁止寄存器 (BS16Tn_IDR)	650
	26.5.2.4	中断有效状态寄存器 (BS16Tn_IVS)	651
	26.5.2.5	原始中断标志寄存器 (BS16Tn_RIF)	651
	26.5.2.6	中断标志屏蔽寄存器 (BS16Tn_IFM)	652
	26.5.2.7	中断清零寄存器 (BS16Tn_ICR)	652
	26.5.2.8	软件生成事件寄存器 (BS16Tn_SGE)	653
	26.5.2.9	计数器寄存器 (BS16Tn_COUNT)	653
	26.5.2.10	预分频寄存器 (BS16Tn_PRES)	654
	26.5.2.11	自动重载寄存器 (BS16Tn_AR)	654
	26.5.2.12	DMA 使能寄存器 (BS16Tn_DMAEN)	655
<b>第 27 章</b>		<b>低功耗定时器(LP16T)</b>	<b>656</b>
27.1		概述	656
27.2		特性	656
27.3		结构框图	657
27.4		功能描述	657
	27.4.1	数字滤波器	657
	27.4.2	预分频器	658
	27.4.3	触发源选择	658

27.4.4	计数模式.....	659
27.4.5	输出波形.....	660
27.4.6	寄存器更新.....	660
27.4.7	中断.....	661
27.5	特殊功能寄存器.....	661
27.5.1	寄存器列表.....	661
27.5.2	寄存器描述.....	662
27.5.2.1	LP16T 控制寄存器 0 (LP16T_CON0) .....	662
27.5.2.2	LP16T 控制寄存器 1 (LP16T_CON1) .....	664
27.5.2.3	LP16T 自动加载寄存器 (LP16T_ARR) .....	665
27.5.2.4	LP16T 计数寄存器 (LP16T_CNT) .....	665
27.5.2.5	LP16T 比较值寄存器 (LP16T_CMP) .....	666
27.5.2.6	LP16T 中断使能寄存器 (LP16T_IER) .....	667
27.5.2.7	LP16T 中断状态寄存器 (LP16T_ISR) .....	668
27.5.2.8	LP16T 中断状态清零寄存器 (LP16T_IFC) .....	669
27.5.2.9	LP16T 更新控制寄存器 (LP16T_UPDATE) .....	669
27.5.2.10	LP16T 写同步状态寄存器 (LP16T_SYNCSTAT) .....	670
<b>第 28 章</b>	<b>串行总线(I2C).....</b>	<b>671</b>
28.1	概述.....	671
28.2	特性.....	671
28.3	结构框图.....	672
28.4	功能描述.....	672
28.4.1	I2C 总线协议.....	672
28.4.1.1	START 和 STOP 条件协议.....	673
28.4.1.2	应答位.....	674
28.4.1.3	I2C 寻址协议.....	675
28.4.1.4	I2C 发送和接收协议.....	676
28.4.2	I2C 时钟要求.....	677
28.4.3	数据传输.....	677
28.4.4	I2C 主机模式.....	679
28.4.5	I2C 从机模式.....	683
28.4.6	I2Cx_TIMINGR 寄存器的配置的例子.....	688
28.4.7	SMBus 具体功能.....	690
28.4.8	SMBus 初始化.....	692
28.4.9	SMBus: I2Cx_TIMEOUTR 寄存器配置的例子.....	693
28.4.10	DMA 请求.....	694
28.4.11	错误情况.....	694
28.4.12	I2C 中断.....	695
28.5	特殊功能寄存器.....	697
28.5.1	寄存器列表.....	697
28.5.2	寄存器描述.....	697
28.5.2.1	I2C 控制寄存器 1 (I2Cx_CON1) .....	697
28.5.2.2	I2C 控制寄存器 2 (I2Cx_CON2) .....	700
28.5.2.3	I2C 本机地址寄存器 1 (I2Cx_ADDR1) .....	703

28.5.2.4	I2C 本机地址寄存器 2 (I2Cx_ADDR2)	704
28.5.2.5	I2C 时钟寄存器 (I2Cx_TIMINGR)	705
28.5.2.6	I2C 超时寄存器 (I2Cx_TIMEOCTR)	707
28.5.2.7	I2C 状态寄存器 (I2Cx_STAT)	709
28.5.2.8	I2C FIFO 控制寄存器 (I2Cx_FCON)	711
28.5.2.9	I2C PEC 寄存器 (I2Cx_PECR)	712
28.5.2.10	I2C 接收数据寄存器 (I2Cx_RXDATA)	712
28.5.2.11	I2C 发送数据寄存器 (I2Cx_TXDATA)	713
28.5.2.12	I2C 中断使能寄存器 (I2Cx_IER)	714
28.5.2.13	I2C 中断禁止寄存器 (I2Cx_IDR)	716
28.5.2.14	I2C 中断有效状态寄存器 (I2Cx_IVS)	718
28.5.2.15	I2C 原始中断标志寄存器 (I2Cx_RIF)	720
28.5.2.16	I2C 中断标志屏蔽寄存器 (I2Cx_IFM)	722
28.5.2.17	I2C 中断清除寄存器 (I2Cx_ICR)	724
<b>第 29 章</b>	<b>串行外设接口(SPI)</b>	<b>726</b>
29.1	概述	726
29.2	特性	726
29.3	结构框图	727
29.4	功能描述	728
29.4.1	通信模式	728
29.4.1.1	时钟相位和极性控制	729
29.4.1.2	数据帧格式	729
29.4.2	从机选择 (NSS) 引脚管理	730
29.4.3	单对单应用	730
29.4.3.1	全双工通信	730
29.4.3.2	半双工通信	731
29.4.4	数据发送和接收	731
29.4.5	DMA 请求	737
29.4.6	SPI 兼容模式	738
29.4.7	CRC 计算	739
29.4.7.1	CRC 功能描述	739
29.4.7.2	CRC 传输管理	740
29.4.7.3	复位 CRC 方式	740
29.4.8	SPI 状态标志	740
29.4.8.1	发送 FIFO 空标志 (TXE)	740
29.4.8.2	发送 FIFO 满标志 (TXF)	740
29.4.8.3	接收 FIFO 空标志 (RXE)	740
29.4.8.4	接收 FIFO 满标志 (RXF)	741
29.4.8.5	通信忙 (BUSY)	741
29.4.9	SPI 错误标志	742
29.4.9.1	发送 FIFO 溢出标志 (TXOV)	742
29.4.9.2	接收 FIFO 溢出标志 (RXOV)	742
29.4.9.3	接收 FIFO 下溢标志 (RXUD)	742
29.4.9.4	主机模式故障 (MODF)	742

29.4.9.5	CRC 错误 (CRCERR)	742
29.4.9.6	SPI 兼容模式帧格式错误 (FRE)	743
29.5	I2S 结构图	743
29.6	I2S 功能描述	744
29.6.1	音频协议	744
29.6.1.1	I2S 飞利浦标准	745
29.6.1.2	MSB 对齐标准	747
29.6.1.3	LSB 对齐标准	748
29.6.1.4	PCM 标准	750
29.6.2	时钟产生器	751
29.6.3	I2S 主机模式	752
29.6.3.1	设置流程	752
29.6.3.2	传输序列	753
29.6.3.3	接收序列	753
29.6.4	I2S 状态标志	753
29.6.4.1	发送 FIFO 空标志 (TXE)	753
29.6.4.2	发送 FIFO 满标志 (TXF)	753
29.6.4.3	接收 FIFO 空标志 (RXE)	753
29.6.4.4	接收 FIFO 满标志 (RXF)	754
29.6.4.5	BUSY 标志 (BUSY)	754
29.6.4.6	声道标志 (CHSIDE)	754
29.6.5	I2S 错误标志	754
29.6.5.1	发送 FIFO 溢出标志 (TXOV)	754
29.6.5.2	发送 FIFO 下溢标志 (TXUD)	754
29.6.5.3	接收 FIFO 溢出标志 (RXOV)	754
29.6.5.4	接收 FIFO 下溢标志 (RXUD)	755
29.7	特殊功能寄存器	755
29.7.1	寄存器列表	755
29.7.2	寄存器描述	756
29.7.2.1	SPI 控制寄存器 1 (SPI_CON1)	756
29.7.2.2	SPI 控制寄存器 2 (SPI_CON2)	759
29.7.2.3	SPI 状态寄存器 (SPI_STAT)	761
29.7.2.4	SPI 数据寄存器 (SPI_DATA)	762
29.7.2.5	SPI CRC 多项式寄存器 (SPI_CRCPOLY)	763
29.7.2.6	SPI RX CRC 寄存器 (SPI_RXCRC)	763
29.7.2.7	SPI TX CRC 寄存器 (SPI_TXCRC)	764
29.7.2.8	SPI I2S 配置寄存器 (SPI_I2SCFG)	765
29.7.2.9	SPI I2S 预分频寄存器 (SPI_I2SPR)	767
29.7.2.10	SPI 中断启用寄存器 (SPI_IER)	768
29.7.2.11	SPI 中断禁用寄存器 (SPI_IDR)	770
29.7.2.12	SPI 中断有效状态寄存器 (SPI_IVS)	772
29.7.2.13	SPI 原始中断标志状态寄存器 (SPI_RIF)	774
29.7.2.14	SPI 中断标志屏蔽状态寄存器 (SPI_IFM)	776
29.7.2.15	SPI 中断清除寄存器 (SPI_ICR)	778

<b>第 30 章</b>	<b>通用异步收发器(UART)</b> .....	<b>780</b>
30.1	概述 .....	780
30.2	特性 .....	780
30.3	结构框图 .....	781
30.4	功能描述 .....	782
30.4.1	数据长度 .....	783
30.4.2	发送器 .....	784
30.4.3	接收器 .....	787
30.4.3.1	防抖电路 .....	787
30.4.3.2	起始位检测 .....	788
30.4.4	状态寄存器 .....	791
30.4.5	波特率生成器 .....	793
30.4.6	自动波特率检测 .....	795
30.4.7	自动流控制 .....	798
30.4.7.1	RTSn 控制 .....	798
30.4.7.2	CTSn 控制 .....	799
30.4.7.3	RS485 驱动使能 (DE) .....	799
30.4.8	校验控制 .....	800
30.4.9	多处理器通信 .....	801
30.4.10	LIN 模式 .....	802
30.4.11	单线半双工通讯 .....	804
30.4.12	智能卡模式 .....	805
30.4.13	IrDA SIR 模块 .....	807
30.4.14	使用 DMA 连续通讯 .....	809
30.4.15	中断配置 .....	811
30.5	特殊功能寄存器 .....	812
30.5.1	寄存器列表 .....	812
30.5.2	寄存器描述 .....	813
30.5.2.1	UART 接收缓冲寄存器 (UART_RXBUF) .....	813
30.5.2.2	UART 发送缓冲寄存器 (UART_TXBUF) .....	813
30.5.2.3	UART 波特率寄存器 (UART_BRR) .....	814
30.5.2.4	UART 线控寄存器 (UART_LCON) .....	815
30.5.2.5	UART 模式控制寄存器 (UART_MCON) .....	818
30.5.2.6	UART RS485 控制寄存器 (UART_RS485) .....	820
30.5.2.7	UART 智能卡控制寄存器 (UART_SCARD) .....	821
30.5.2.8	UART LIN 控制寄存器 (UART_LIN) .....	822
30.5.2.9	UART 接收超时寄存器 (UART_RTOR) .....	823
30.5.2.10	UART FIFO 控制寄存器 (UART_FCON) .....	824
30.5.2.11	UART 状态寄存器 (UART_STAT) .....	825
30.5.2.12	UART 中断使能寄存器 (UART_IER) .....	828
30.5.2.13	UART 中断禁止寄存器 (UART_IDR) .....	830
30.5.2.14	UART 中断有效状态寄存器 (UART_IVS) .....	832
30.5.2.15	UART 原始中断标志寄存器 (UART_RIF) .....	834
30.5.2.16	UART 中断标志屏蔽寄存器 (UART_IFM) .....	837



	30.5.2.17	UART 中断清除寄存器 (UART_ICR) .....	840
<b>第 31 章</b>		<b>QSPI Flash 控制器 (QSPIC) .....</b>	<b>842</b>
31.1		概述 .....	842
31.2		特性 .....	842
31.3		结构框图 .....	843
31.4		功能描述 .....	844
	31.4.1	AHB 控制接口 .....	844
	31.4.1.1	AHB 接口 .....	844
	31.4.1.2	AHB 地址重映射 .....	844
	31.4.1.3	写保护 .....	844
	31.4.1.4	访问转发 .....	845
	31.4.1.5	顺序访问检测和迸发长度 .....	845
	31.4.1.6	AHB 地址解码器 .....	845
	31.4.2	直接访问控制器 (DAC) .....	845
	31.4.3	间接控制器 (INDAC) .....	847
	31.4.3.1	间接读控制器 .....	847
	31.4.3.2	间接写控制器 .....	849
	31.4.3.3	间接访问队列 .....	851
	31.4.3.4	间接传输: 连续写读 .....	851
	31.4.3.5	访问 SRAM .....	851
	31.4.4	DMA 外设控制器 .....	853
	31.4.4.1	操作顺序 .....	853
	31.4.5	软件触发指令生成器 (STIG) .....	855
	31.4.5.1	响应 STIG 请求 .....	855
	31.4.6	直接/间接访问控制器和 STIG 间的仲裁 .....	856
	31.4.7	SPI 命令转换 .....	856
	31.4.8	Flash 指令类型选择 .....	857
	31.4.9	APB 接口与寄存器模块 .....	858
	31.4.10	捕捉读数据 .....	859
	31.4.10.1	Tap 机制 .....	859
	31.4.11	BOOT 特性描述 .....	860
	31.4.12	示例: 8 字节读传输 .....	860
31.5		编程指引 .....	862
	31.5.1	复位后配置 QSPI 控制器 .....	862
	31.5.2	QSPI 控制器配置优化 .....	862
	31.5.3	Flash 命令控制寄存器的使用 (STIG 操作) .....	863
	31.5.4	SPI 传统模式 .....	863
	31.5.5	进入和退出 XIP 模式 .....	864
	31.5.5.1	从 POR 进入 XIP 模式 .....	864
	31.5.5.2	其他进入 XIP 模式的情况 .....	864
	31.5.5.3	退出 XIP 模式 .....	865
	31.5.6	间接数据传输模式 .....	865
	31.5.7	AHB 地址重映射 .....	865
	31.5.8	中断响应 .....	865

31.5.9	AHB 保护寄存器 .....	865
31.6	特殊功能寄存器 .....	866
31.6.1	寄存器列表 .....	866
31.6.2	寄存器描述 .....	868
31.6.2.1	QSPI 配置寄存器 (QSPI_CR) .....	868
31.6.2.2	QSPI 器件读指令寄存器 (QSPI_DRIR) .....	872
31.6.2.3	QSPI 器件写指令配置寄存器 (QSPI_DWIR) .....	873
31.6.2.4	QSPI 器件延时寄存器 (QSPI_DDLR) .....	874
31.6.2.5	QSPI 读数据捕捉寄存器 (QSPI_RDCCR) .....	875
31.6.2.6	QSPI 器件大小配置寄存器 (QSPI_DSCR) .....	876
31.6.2.7	QSPI SRAM 分块配置寄存器 (QSPI_SPR) .....	877
31.6.2.8	QSPI 间接 AHB 地址触发寄存器 (QSPI_IATR) .....	878
31.6.2.9	QSPI DMA 外设配置寄存器 (QSPI_DMACR) .....	878
31.6.2.10	QSPI 地址重映射寄存器 (QSPI_RAR) .....	879
31.6.2.11	QSPI 模式位配置寄存器 (QSPI_MBR) .....	879
31.6.2.12	QSPI SRAM 数据深度状态寄存器 (QSPI_SFLR) .....	880
31.6.2.13	QSPI 发送阈值寄存器 (QSPI_TXHR) .....	880
31.6.2.14	QSPI 接收阈值寄存器 (QSPI_RXHR) .....	881
31.6.2.15	QSPI 写完成控制寄存器 (QSPI_WCR) .....	881
31.6.2.16	QSPI 轮询结束寄存器 (QSPI_PER) .....	882
31.6.2.17	QSPI 中断状态寄存器 (QSPI_IFR) .....	883
31.6.2.18	QSPI 中断屏蔽寄存器 (QSPI_IMR) .....	884
31.6.2.19	QSPI 写保护低位寄存器 (QSPI_WPLR) .....	885
31.6.2.20	QSPI 写保护高位寄存器 (QSPI_WPHR) .....	885
31.6.2.21	QSPI 写保护寄存器 (QSPI_WPCR) .....	886
31.6.2.22	QSPI 间接读传输控制寄存器 (QSPI_IRTR) .....	887
31.6.2.23	QSPI 间接读传输数据深度阈值寄存器 (QSPI_IRTWR) .....	888
31.6.2.24	QSPI 间接读传输起始寄存器 (QSPI_IRTSAR) .....	888
31.6.2.25	QSPI 间接读传输字节数寄存器 (QSPI_IRTNR) .....	888
31.6.2.26	QSPI 间接写传输控制寄存器 (QSPI_IWTR) .....	889
31.6.2.27	QSPI 间接写传输数据深度阈值寄存器 (QSPI_IWTWR) .....	890
31.6.2.28	QSPI 间接写传输起始寄存器 (QSPI_IWTSAR) .....	890
31.6.2.29	QSPI 间接写传输字节数寄存器 (QSPI_IWTNR) .....	890
31.6.2.30	QSPI 间接触发地址范围寄存器 (QSPI_ITARR) .....	891
31.6.2.31	QSPI Flash 命令控制寄存器 (QSPI_FCR) .....	892
31.6.2.32	QSPI Flash 命令地址寄存器 (QSPI_FCAR) .....	893
31.6.2.33	QSPI Flash 命令读数据低位寄存器 (QSPI_FCRLR) .....	894
31.6.2.34	QSPI Flash 命令读数据高位寄存器 (QSPI_FCRHR) .....	894
31.6.2.35	QSPI Flash 命令写数据低位寄存器 (QSPI_FCWLR) .....	895
31.6.2.36	QSPI Flash 命令写数据高位寄存器 (QSPI_FCWHR) .....	895
31.6.2.37	QSPI 轮询 Flash 状态寄存器 (QSPI_PFSR) .....	896
31.6.2.38	QSPI 模块 ID 寄存器 (QSPI_MIDR) .....	896
<b>第 32 章</b>	<b>自适应差分脉冲编码调制(ADPCM).....</b>	<b>897</b>
32.1	概述 .....	897

32.2	特性 .....	897
32.3	结构框图 .....	897
32.4	功能描述 .....	897
32.5	特殊寄存器 .....	898
32.5.1	寄存器列表 .....	898
32.5.2	寄存器描述 .....	898
32.5.2.1	ADPCM 译码器输入数据和控制寄存器 (ADPCM_DINCTL) .....	898
32.5.2.2	ADPCM 通道 0 和 1 译码结果寄存器 (ADPCM_DO10) .....	899
32.5.2.3	ADPCM 通道 2 和 3 译码结果寄存器 (ADPCM_DO32) .....	900
32.5.2.4	ADPCM 中断使能寄存器 (ADPCM_IER) .....	900
32.5.2.5	ADPCM 中断禁止寄存器 (ADPCM_IDR) .....	900
32.5.2.6	ADPCM 中断屏蔽寄存器 (ADPCM_IMR) .....	901
32.5.2.7	ADPCM 中断标志寄存器 (ADPCM_RIS) .....	901
32.5.2.8	ADPCM 中断标志清除寄存器 (ADPCM_ISC) .....	901
32.5.2.9	译码状态寄存器 (ADPCM_STATUS) .....	902
32.5.2.10	DMA 控制寄存器 (ADPCM_DMACTRL) .....	902
<b>第 33 章</b>	<b>Keyboard_Scanner (KEYS) .....</b>	<b>904</b>
33.1	概述 .....	904
33.2	特性 .....	904
33.3	结构框图 .....	904
33.4	功能描述 .....	905
33.5	特殊功能寄存器 .....	906
33.5.1	寄存器列表 .....	906
33.5.2	寄存器描述 .....	907
33.5.2.1	KEY_SCANNER 控制寄存器 0 (KSCAN_CTRL0) .....	907
33.5.2.2	KEY_SCANNER 控制寄存器 1 (KSCAN_CTRL1) .....	908
33.5.2.3	KEY_SCANNER MODKEY0 寄存器 (KSCAN_MODKEY0) .....	908
33.5.2.4	KEY_SCANNER MODKEY1 寄存器 (KSCAN_MODKEY1) .....	909
33.5.2.5	KEY_SCANNER MODKEY2 寄存器 (KSCAN_MODKEY2) .....	909
33.5.2.6	KEY_SCANNER MODKEY3 寄存器 (KSCAN_MODKEY3) .....	910
33.5.2.7	KEY_SCANNER MODKEY4 寄存器 (KSCAN_MODKEY4) .....	910
33.5.2.8	KEY_SCANNER MODKEY5 寄存器 (KSCAN_MODKEY5) .....	910
33.5.2.9	KEY_SCANNER MODKEY6 寄存器 (KSCAN_MODKEY6) .....	911
33.5.2.10	KEY_SCANNER MODKEY7 寄存器 (KSCAN_MODKEY7) .....	911
33.5.2.11	KEY_SCANNER FIFO 控制寄存器 (KSCAN_FIFOCON) .....	911
33.5.2.12	KEY_SCANNER 中断使能寄存器 (KSCAN_IER) .....	912
33.5.2.13	KEY_SCANNER 中断标志寄存器 (KSCAN_RIF) .....	913
33.5.2.14	KEY_SCANNER 中断上报寄存器 (KSCAN_IFM) .....	914
33.5.2.15	KEY_SCANNER 中断清除寄存器 (KSCAN_ICR) .....	915
<b>第 34 章</b>	<b>正交译码器(QDEC) .....</b>	<b>917</b>
34.1	概述 .....	917
34.2	特性 .....	917
34.3	结构框图 .....	917
34.4	功能描述 .....	917

34.4.1	机械模式.....	918
34.4.2	光电模式.....	919
34.4.3	RAMBOL 模式.....	920
34.4.4	Debounce 滤波.....	922
34.4.5	输入输出管脚配置.....	922
34.5	特殊功能寄存器.....	923
34.5.1	寄存器列表.....	923
34.5.2	寄存器描述.....	924
34.5.2.1	QDEC 控制寄存器 0 (QDEC_CTRL0).....	924
34.5.2.2	QDEC 计数值输出寄存器 (QDEC_RD_ZCNT).....	925
<b>第 35 章</b>	<b>触控按键 (TK).....</b>	<b>926</b>
35.1	概述.....	926
35.2	内部结构图.....	926
35.3	触控按键扫描工作原理.....	926
35.4	触控按键工作流程.....	929
35.5	特殊功能寄存器.....	930
35.5.1	寄存器列表.....	930
35.5.2	寄存器描述.....	931
35.5.2.1	触控按键功能选择寄存器 (TK_SEL).....	931
35.5.2.2	触控调整寄存器 (TK_TUN).....	932
35.5.2.3	触控控制寄存器 (TK_CTL).....	933
35.5.2.4	扫描结果寄存器 (TK_DA).....	934
35.5.2.5	放大系数寄存器 (TK_MOD).....	935
35.5.2.6	内部参考电压控制寄存器 (TK_VRC).....	935
35.5.2.7	TK 开关电平输出控制寄存器 (TK_OUTS0).....	936
35.5.2.8	TK 中断寄存器 (TK_INT).....	937
35.5.2.9	TK 充放电扫描计数值寄存器 (TK_SCT).....	937
<b>第 36 章</b>	<b>调试配置控制 (DBGC).....</b>	<b>939</b>
36.1	概述.....	939
36.2	特性.....	939
36.3	结构框图.....	939
36.4	功能描述.....	940
36.4.1	调试端口.....	940
36.4.2	调试冻结.....	940
36.4.3	调试复位.....	940
36.4.4	MEM-AP 访问端口.....	941
36.5	特殊功能寄存器.....	941
36.5.1	寄存器列表.....	941
36.5.2	寄存器描述.....	942
36.5.2.1	DBG 器件识别码寄存器 (DBGC_IDCODE).....	942
36.5.2.2	DBG 控制寄存器 (DBGC_CR).....	942
36.5.2.3	APB1 外设调试冻结寄存器 (DBGC_APB1FZ).....	943
36.5.2.4	APB2 外设调试冻结寄存器 (DBGC_APB2FZ).....	944
36.5.2.5	定时器输出调试冻结寄存器 (DBGC_TIMOFZ).....	945

<b>第 37 章</b>	<b>PDM2PCM 转换器 (PDM2PCM)</b>	<b>946</b>
37.1	概述	946
37.2	特性	946
37.3	结构框图	947
37.4	功能描述	947
37.4.1	PDMPCM_PROC	947
37.4.2	PDPCM_REG	948
37.4.3	PDPCM_FIFO	948
37.5	特殊功能寄存器	949
37.5.1	寄存器列表	949
37.5.2	寄存器描述	951
37.5.2.1	PDM2PCM 接收数据缓存寄存器 (PDM2PCM_RXBR)	951
37.5.2.2	PDM2PCM 控制寄存器 (PDM2PCM_CTRL)	951
37.5.2.3	PDM2PCM FIFO 控制寄存器 (PDM2PCM_FCR)	952
37.5.2.4	PDM2PCM 数字补偿滤波器系数 0 寄存器 (PDM2PCM_0FILT)	953
37.5.2.5	PDM2PCM 数字补偿滤波器系数 1 寄存器 (PDM2PCM_1FILT)	953
37.5.2.6	PDM2PCM 数字补偿滤波器系数 2 寄存器 (PDM2PCM_2FILT)	954
37.5.2.7	PDM2PCM 数字补偿滤波器系数 3 寄存器 (PDM2PCM_3FILT)	954
37.5.2.8	PDM2PCM 数字补偿滤波器系数 4 寄存器 (PDM2PCM_4FILT)	955
37.5.2.9	PDM2PCM 数字补偿滤波器系数 5 寄存器 (PDM2PCM_5FILT)	955
37.5.2.10	PDM2PCM 数字补偿滤波器系数 6 寄存器 (PDM2PCM_6FILT)	955
37.5.2.11	PDM2PCM 数字补偿滤波器系数 7 寄存器 (PDM2PCM_7FILT)	956
37.5.2.12	PDM2PCM 数字补偿滤波器系数 8 寄存器 (PDM2PCM_8FILT)	956
37.5.2.13	PDM2PCM 数字补偿滤波器系数 9 寄存器 (PDM2PCM_9FILT)	957
37.5.2.14	PDM2PCM 数字补偿滤波器系数 10 寄存器 (PDM2PCM_10FILT)	957
37.5.2.15	PDM2PCM 数字补偿滤波器系数 11 寄存器 (PDM2PCM_11FILT)	957
37.5.2.16	PDM2PCM 数字补偿滤波器系数 12 寄存器 (PDM2PCM_12FILT)	958
37.5.2.17	PDM2PCM 数字补偿滤波器系数 13 寄存器 (PDM2PCM_13FILT)	958
37.5.2.18	PDM2PCM 数字补偿滤波器系数 14 寄存器 (PDM2PCM_14FILT)	959
37.5.2.19	PDM2PCM 数字补偿滤波器系数 15 寄存器 (PDM2PCM_15FILT)	959
37.5.2.20	PDM2PCM 数字补偿滤波器系数 16 寄存器 (PDM2PCM_16FILT)	960
37.5.2.21	PDM2PCM 数字补偿滤波器系数 17 寄存器 (PDM2PCM_17FILT)	960
37.5.2.22	PDM2PCM 数字补偿滤波器系数 18 寄存器 (PDM2PCM_18FILT)	961
37.5.2.23	PDM2PCM 数字补偿滤波器系数 19 寄存器 (PDM2PCM_19FILT)	961
37.5.2.24	PDM2PCM 数字补偿滤波器系数 20 寄存器 (PDM2PCM_20FILT)	962
37.5.2.25	PDM2PCM 数字补偿滤波器系数 21 寄存器 (PDM2PCM_21FILT)	962
37.5.2.26	PDM2PCM 数字补偿滤波器系数 22 寄存器 (PDM2PCM_22FILT)	963
37.5.2.27	PDM2PCM 数字补偿滤波器系数 23 寄存器 (PDM2PCM_23FILT)	963
37.5.2.28	PDM2PCM 数字补偿滤波器系数 24 寄存器 (PDM2PCM_24FILT)	964
37.5.2.29	PDM2PCM 数字补偿滤波器系数 25 寄存器 (PDM2PCM_25FILT)	964
37.5.2.30	PDM2PCM 数字补偿滤波器系数 26 寄存器 (PDM2PCM_26FILT)	965
37.5.2.31	PDM2PCM 数字补偿滤波器系数 27 寄存器 (PDM2PCM_27FILT)	965
37.5.2.32	PDM2PCM 数字补偿滤波器系数 28 寄存器 (PDM2PCM_28FILT)	966
37.5.2.33	PDM2PCM 数字补偿滤波器系数 29 寄存器 (PDM2PCM_29FILT)	966

37.5.2.34	PDM2PCM 数字补偿滤波器系数 30 寄存器 (PDM2PCM_30FILT)	.....967
37.5.2.35	PDM2PCM 数字补偿滤波器系数 31 寄存器 (PDM2PCM_31FILT)	.....967
37.5.2.36	PDM2PCM 数字补偿滤波器系数 32 寄存器 (PDM2PCM_32FILT)	.....968
37.5.2.37	PDM2PCM 数字补偿滤波器系数 33 寄存器 (PDM2PCM_33FILT)	.....968
37.5.2.38	PDM2PCM 数字补偿滤波器系数 34 寄存器 (PDM2PCM_34FILT)	.....969
37.5.2.39	PDM2PCM 数字补偿滤波器系数 35 寄存器 (PDM2PCM_35FILT)	.....969
37.5.2.40	PDM2PCM 数字补偿滤波器系数 36 寄存器 (PDM2PCM_36FILT)	.....970
37.5.2.41	PDM2PCM 数字补偿滤波器系数 37 寄存器 (PDM2PCM_37FILT)	.....970
37.5.2.42	PDM2PCM 数字补偿滤波器系数 38 寄存器 (PDM2PCM_38FILT)	.....970
37.5.2.43	PDM2PCM 数字补偿滤波器系数 39 寄存器 (PDM2PCM_39FILT)	.....971
37.5.2.44	PDM2PCM 数字补偿滤波器系数 40 寄存器 (PDM2PCM_40FILT)	.....971
37.5.2.45	PDM2PCM 数字补偿滤波器系数 41 寄存器 (PDM2PCM_41FILT)	.....972
37.5.2.46	PDM2PCM 数字补偿滤波器系数 42 寄存器 (PDM2PCM_42FILT)	.....972
37.5.2.47	PDM2PCM 数字补偿滤波器系数 43 寄存器 (PDM2PCM_43FILT)	.....973
37.5.2.48	PDM2PCM 数字补偿滤波器系数 44 寄存器 (PDM2PCM_44FILT)	.....973
37.5.2.49	PDM2PCM 数字补偿滤波器系数 45 寄存器 (PDM2PCM_45FILT)	.....974
37.5.2.50	PDM2PCM 状态寄存器 (PDM2PCM_SR)	.....974
37.5.2.51	PDM2PCM 中断使能寄存器 (PDM2PCM_IER)	.....975
37.5.2.52	PDM2PCM 中断不使能寄存器 (PDM2PCM_IDR)	.....976
37.5.2.53	PDM2PCM 中断使能状态寄存器 (PDM2PCM_IVS)	.....977
37.5.2.54	PDM2PCM 中断标志位寄存器 (PDM2PCM_RIF)	.....977
37.5.2.55	PDM2PCM 中断标志位 masked 寄存器 (PDM2PCM_IFM)	.....978
37.5.2.56	PDM2PCM 中断清除寄存器 (PDM2PCM_ICR)	.....979
<b>第 38 章</b>	<b>低功耗蓝牙 (BLE)</b>	<b>.....981</b>
38.1	概述	.....981
38.2	特性	.....981
38.3	结构框图	.....981
38.4	功能描述	.....982
38.4.1	Exchange Memory	.....982
38.4.2	低功耗睡眠及唤醒	.....982
38.4.2.1	工作模式切换至睡眠模式	.....982
38.4.2.2	睡眠模式切换至工作模式	.....983
38.4.2.3	睡眠模式自动超时唤醒	.....983
38.4.2.4	睡眠模式异步事件唤醒	.....984
38.5	特殊功能寄存器	.....985
38.5.1	寄存器列表	.....985
38.5.2	寄存器描述	.....987
38.5.2.1	AFE 接口信号使能寄存器 (AFE_ENABLE)	.....987
38.5.2.2	AFE 测试使能控制寄存器 (AFE_TEST_ENABLE)	.....989
38.5.2.3	AFE PLL 模拟参数控制寄存器 1 (AFE_PLLANACR1)	.....990
38.5.2.4	AFE PLL 模拟参数控制寄存器 2 (AFE_PLLANACR2)	.....990
38.5.2.5	AFE PLL 模拟参数控制寄存器 3 (AFE_PLLANACR3)	.....991
38.5.2.6	AFE TXDAC 控制寄存器 (AFE_TXDAC)	.....991
38.5.2.7	AFE Rx 控制寄存器 1 (AFE_RX_CTRL1)	.....992

38.5.2.8	AFE Rx 控制寄存器 2 (AFE_RX_CTRL2)	993
38.5.2.9	AFE RX RCCAL 控制寄存器 (AFE_RX_RCCAL)	994
38.5.2.10	AFE PLL 静态控制寄存器 (AFE_PLLSTCR)	995
38.5.2.11	AFE PLL 载波分频比配置寄存器 (AFE_PLLCHDIVCR)	996
38.5.2.12	AFE PLL 环路控制寄存器 (AFE_PLLCR)	997
38.5.2.13	AFE PLL 两点调制控制寄存器 (AFE_PLLTPMODCR)	998
38.5.2.14	AFE PLL 环路 Bypass 模式控制寄存器 (AFE_PLLSPICR)	999
38.5.2.15	AFE DAC 模块 Bypass 模式控制寄存器 (AFE_DACSPICR)	1000
38.5.2.16	AFE Glpf 中 FIFO 控制和状态寄存器 (AFE_GLPFFIFO CR)	1000
38.5.2.17	AFE 接口信号控制模式寄存器 0 (AFE_CTRL_MODE0)	1001
38.5.2.18	AFE 接口信号控制模式寄存器 1 (AFE_CTRL_MODE1)	1002
38.5.2.19	AFE RAMPGEN 控制寄存器 (AFE_RAMPGEN_CNTL)	1002
38.5.2.20	AFE RPLL_DLY 控制寄存器 (AFE_RPLL_DLY)	1003
38.5.2.21	AFE TX_DLY0 控制寄存器 (AFE_TX_DLY0)	1003
38.5.2.22	AFE TX_DLY1 控制寄存器 (AFE_TX_DLY1)	1003
38.5.2.23	AFE RX_DLY0 控制寄存器 (AFE_RX_DLY0)	1004
38.5.2.24	AFE RFDIG 配置寄存器 (AFE_RFDIG_CFG)	1004
38.5.2.25	AFE IOMUX 选择控制寄存器 (AFE_IOMUX_CTRL)	1006
38.5.2.26	AFE DAC_GAINCAL 分段校准控制寄存器 (AFE_DAC_GAINCTRL)	1006
38.5.2.27	AFE DAC_GAINCAL_2M 分段校准控制寄存器 (AFE_DAC_GAINCTRL_2M)	1007
38.5.2.28	AFE 保留扩展寄存器 (AFE_RESERVED)	1007
<b>第 39 章</b>	<b>Flash 信息区</b>	<b>1008</b>
39.1	概述	1008
39.2	特性	1008
39.3	功能描述	1009
39.3.1	Flash 信息区只读信息	1009
39.3.2	Flash 信息区配置信息	1009
39.3.2.1	芯片配置字 CFG_WORD	1009
39.3.2.2	写保护区域配置字 CFG_WRPT	1010
39.3.2.3	数据区配置字 CFG_DAFLS	1011
39.3.2.4	晶振频率校准配置字 CFG_XTAL	1011
39.3.2.5	全局读保护配置字 CFG_GBRDP	1011
39.3.2.6	用户程序校验码 CHKSUM	1012
39.3.2.7	私有代码读出保护区域配置字 CFG_PCROP	1013
<b>附录 1</b>	<b>ARM Cortex-M3 附录</b>	<b>1014</b>
附录 1.1	ARM Cortex-M3 用户指南: 简介	1014
附录 1.1.1	关于处理器和内核外设	1014
附录 1.1.1.1	系统级接口	1015
附录 1.1.1.2	集成的可配置调试功能	1015
附录 1.1.1.3	Cortex-M3 处理器特性和优点汇总	1016
附录 1.1.1.4	Cortex-M3 内核外设	1016
附录 1.2	ARM Cortex-M3 用户指南: 指令集	1016
附录 1.2.1	指令集汇总	1016
附录 1.2.2	内在函数	1022

附录 1.2.3	关于指令描述 .....	1023
附录 1.2.3.1	操作数 .....	1023
附录 1.2.3.2	使用 PC 或 SP 时的限制 .....	1023
附录 1.2.3.3	灵活的第二操作数 .....	1023
附录 1.2.3.4	移位操作 .....	1024
附录 1.2.3.5	地址对齐 .....	1027
附录 1.2.3.6	相对 PC 的表达式 .....	1027
附录 1.2.3.7	条件执行 .....	1029
附录 1.2.3.8	指令宽度选择 .....	1031
附录 1.2.4	内存访问指令 .....	1031
附录 1.2.4.1	ADR .....	1032
附录 1.2.4.2	LDR 和 STR (直接偏移量) .....	1033
附录 1.2.4.3	LDR 和 STR (寄存器偏移量) .....	1035
附录 1.2.4.4	LDR 和 STR (非特权) .....	1036
附录 1.2.4.5	LDR (相对 PC) .....	1037
附录 1.2.4.6	LDM 和 STM .....	1039
附录 1.2.4.7	PUSH 和 POP .....	1041
附录 1.2.4.8	LDREX 和 STREX .....	1042
附录 1.2.4.9	CLREX .....	1043
附录 1.2.5	通用数据处理指令 .....	1044
附录 1.2.5.1	ADD、ADC、SUB、SBC 和 RSB .....	1046
附录 1.2.5.2	AND, ORR, EOR, BIC 和 ORN .....	1048
附录 1.2.5.3	ASR, LSL, LSR, ROR 和 RRX .....	1049
附录 1.2.5.4	CLZ .....	1050
附录 1.2.5.5	CMP 和 CMN .....	1051
附录 1.2.5.6	MOV 和 MVN .....	1052
附录 1.2.5.7	MOVT .....	1053
附录 1.2.5.8	REV, REV16, REVSH 和 RBIT .....	1054
附录 1.2.5.9	TST 和 TEQ .....	1055
附录 1.2.6	乘法和除法指令 .....	1056
附录 1.2.6.1	MUL、MLA 和 MLS .....	1056
附录 1.2.6.2	UMULL, UMLAL, SMULL 和 SMLAL .....	1057
附录 1.2.6.3	SDIV 和 UDIV .....	1058
附录 1.2.7	饱和指令 .....	1059
附录 1.2.7.1	SSAT 和 USAT .....	1059
附录 1.2.8	位域指令 .....	1061
附录 1.2.8.1	BFC 和 BFI .....	1061
附录 1.2.8.2	SBFX 和 UBFX .....	1062
附录 1.2.8.3	SXT 和 UXT .....	1062
附录 1.2.9	跳转和控制指令 .....	1064
附录 1.2.9.1	B、BL、BX 和 BLX .....	1064
附录 1.2.9.2	CBZ 和 CBNZ .....	1066
附录 1.2.9.3	IT .....	1067
附录 1.2.9.4	TBB 和 TBH .....	1068



附录 1.2.10	其他指令 .....	1070
附录 1.2.10.1	BKPT .....	1070
附录 1.2.10.2	CPS .....	1071
附录 1.2.10.3	DMB .....	1072
附录 1.2.10.4	DSB .....	1072
附录 1.2.10.5	ISB .....	1073
附录 1.2.10.6	MRS .....	1073
附录 1.2.10.7	MSR .....	1074
附录 1.2.10.8	NOP .....	1074
附录 1.2.10.9	SEV .....	1075
附录 1.2.10.10	SVC .....	1075
附录 1.2.10.11	WFE .....	1076
附录 1.2.10.12	WFI .....	1076
附录 1.3	ARM Cortex-M3 用户指南: 处理器 .....	1077
附录 1.3.1	编程模型 .....	1077
附录 1.3.1.1	处理器模式和软件执行的特权等级 .....	1077
附录 1.3.1.2	堆栈 .....	1077
附录 1.3.1.3	内核寄存器 .....	1078
附录 1.3.1.4	异常和中断 .....	1084
附录 1.3.1.5	数据类型 .....	1085
附录 1.3.1.6	Cortex 微控制器软件接口标准 .....	1085
附录 1.3.2	存储器模型 .....	1086
附录 1.3.2.1	存储区、类型和属性 .....	1087
附录 1.3.2.2	存储器系统访问的排序 .....	1087
附录 1.3.2.3	存储器访问行为 .....	1088
附录 1.3.2.4	存储器访问的软件排序 .....	1089
附录 1.3.2.5	位段 .....	1090
附录 1.3.2.6	存储器字节序 .....	1092
附录 1.3.2.7	同步原语 .....	1092
附录 1.3.2.8	同步原语的编程提示 .....	1093
附录 1.3.3	异常模型 .....	1094
附录 1.3.3.1	异常状态 .....	1094
附录 1.3.3.2	异常类型 .....	1094
附录 1.3.3.3	异常处理程序 .....	1096
附录 1.3.3.4	向量表 .....	1097
附录 1.3.3.5	异常优先级 .....	1098
附录 1.3.3.6	中断优先级分组 .....	1098
附录 1.3.3.7	异常进入和返回 .....	1099
附录 1.3.4	故障处理 .....	1101
附录 1.3.4.1	故障类型 .....	1101
附录 1.3.4.2	故障升级和 HardFault .....	1102
附录 1.3.4.3	故障状态寄存器和故障地址寄存器 .....	1102
附录 1.3.4.4	锁定 .....	1103
附录 1.3.5	电源管理 .....	1103

附录 1.3.5.1	进入睡眠模式.....	1103
附录 1.3.5.2	从睡眠模式唤醒.....	1105
附录 1.3.5.3	唤醒中断控制器.....	1105
附录 1.3.5.4	电源管理编程提示.....	1105
附录 1.4	ARM Cortex-M3 用户指南：外设.....	1106
附录 1.4.1	关于 Cortex-M3 外设.....	1106
附录 1.4.2	可嵌套向量中断控制器.....	1107
附录 1.4.2.1	Cortex-M3 NVIC 寄存器的 CMSIS 映射.....	1108
附录 1.4.2.2	中断置位使能寄存器.....	1108
附录 1.4.2.3	中断清零使能寄存器.....	1109
附录 1.4.2.4	中断置位挂起寄存器.....	1109
附录 1.4.2.5	中断清零挂起寄存器.....	1110
附录 1.4.2.6	中断有效位寄存器.....	1110
附录 1.4.2.7	中断优先级寄存器.....	1112
附录 1.4.2.8	软件触发中断寄存器.....	1113
附录 1.4.2.9	电平触发和脉冲中断.....	1113
附录 1.4.2.10	NVIC 设计提示和建议.....	1114
附录 1.4.3	系统控制模块.....	1115
附录 1.4.3.1	Cortex-M3 SCB 寄存器的 CMSIS 映射.....	1115
附录 1.4.3.2	辅助控制寄存器 (ACTLR).....	1115
附录 1.4.3.3	CPUID 基址寄存器.....	1116
附录 1.4.3.4	中断控制和状态寄存器.....	1116
附录 1.4.3.5	向量表偏移量寄存器.....	1119
附录 1.4.3.6	应用中断和复位控制寄存器.....	1119
附录 1.4.3.7	系统控制寄存器.....	1120
附录 1.4.3.8	配置和控制寄存器.....	1121
附录 1.4.3.9	系统处理程序优先级寄存器.....	1122
附录 1.4.3.10	系统处理程序控制和状态寄存器.....	1124
附录 1.4.3.11	可配置故障状态寄存器.....	1125
附录 1.4.3.12	HardFault 状态寄存器.....	1130
附录 1.4.3.13	存储器管理故障地址寄存器.....	1131
附录 1.4.3.14	总线故障地址寄存器.....	1131
附录 1.4.3.15	系统控制模块设计提示和建议.....	1131
附录 1.4.3.16	系统定时器 SysTick.....	1132
附录 1.4.3.17	SysTick 控制和状态寄存器.....	1132
附录 1.4.3.18	SysTick 重载值寄存器.....	1133
附录 1.4.3.19	SysTick 当前值寄存器.....	1133
附录 1.4.3.20	SysTick 校准值寄存器.....	1133
附录 1.4.3.21	SysTick 设计提示和建议.....	1134
附录 1.4.4	存储器保护单元.....	1134
附录 1.4.4.1	MPU 类型寄存器.....	1136
附录 1.4.4.2	MPU 控制寄存器.....	1136
附录 1.4.4.3	MPU 区号寄存器.....	1137
附录 1.4.4.4	MPU 区基址寄存器.....	1138

附录 1.4.4.5 MPU 区的属性和大小寄存器 .....	1138
附录 1.4.4.6 MPU 访问权限属性 .....	1140
附录 1.4.4.7 MPU 不匹配 .....	1141
附录 1.4.4.8 更新一个 MPU 区 .....	1141
附录 1.4.4.9 MPU 设计提示和建议 .....	1144
附录 1.5 ARM Cortex-M3 用户指南：术语表 .....	1145
<b>第 40 章 修订历史 .....</b>	<b>1148</b>

## 图目录

图 2-1 系统框图 .....	48
图 3-1 系统节拍定时器时钟 .....	56
图 3-2 DMA 请求列表 .....	64
图 3-3 DMA 多路复用器与 DMA 连接图 .....	64
图 3-4 独立看门狗计数时钟 .....	67
图 3-5 窗口看门狗计数时钟 .....	68
图 3-6 低功耗定时器计数时钟 .....	74
图 3-7 RTC 计数时钟 .....	75
图 4-1 系统总线矩阵 .....	77
图 4-2 Flash 映射示意图 .....	80
图 4-3 Flash 组织架构示意图 .....	80
图 5-1 结构图 .....	86
图 5-2 FLASH 操作请求流程 .....	89
图 5-3 FLASH 全擦除模式流程图 .....	90
图 5-4 FLASH 页擦除模式流程图 .....	91
图 5-5 FLASH 双字编程模式流程图 .....	92
图 5-6 FLASH 快速编程模式流程图 .....	94
图 7-1 电源示意图 .....	120
图 7-2 POR/PDR 示意图 .....	121
图 7-3 BOR 示意图 .....	122
图 7-4 LVD 示意图 .....	122
图 9-1 复位结构图 .....	150
图 9-2 上电复位示意图 .....	154
图 9-3 低电压复位示意图 .....	154
图 10-1 时钟结构图 .....	170
图 10-2 HOSC 结构图 .....	171
图 11-1 DMA 结构框图 .....	210
图 11-2 轮询流程图 .....	213
图 11-3 乒乓示例 .....	215
图 11-4 存储器分散-聚集示例 .....	219
图 11-5 外设交替-聚集示例 .....	222
图 11-6 16 通道存储器映射（包括交替数据结构） .....	224
图 12-1 PIS 结构框图 .....	289
图 12-2 高电平调制输出波形图 .....	295
图 12-3 低电平调制输出波形图 .....	295
图 13-1 独立看门狗时序图 .....	353
图 14-1 窗口看门狗中断和溢出复位产生时序图（WWDTWIN 设定为 25%） .....	361
图 14-2 错误的喂狗时序图（WWDTWIN 设定为 25%） .....	362
图 15-1 GPIO 结构框图 .....	369
图 15-2 外中断 GPIO 映像 .....	372
图 16-1 结构框图 .....	391
图 17-1 AES-ECB 模式加密 .....	399
图 17-2 AES-ECB 模式解密 .....	400

图 17-3	AES-CBC 模式加密 .....	401
图 17-4	AES-CBC 模式解密 .....	403
图 17-5	CTR 加密流程 .....	404
图 17-6	CTR 解密流程 .....	405
图 17-7	计数器模式下的初始计数器块结构 .....	405
图 17-8	GCM 加密流程 .....	407
图 17-9	GCM 解密流程 .....	408
图 17-10	DES/TDES-ECB 模式加密 .....	411
图 17-11	DES/TDES-ECB 模式解密 .....	412
图 17-12	DES/TDES-CBC 模式加密 .....	413
图 17-13	DES/TDES-CBC 模式解密 .....	415
图 17-14	SM4-ECB 模式加密 .....	417
图 17-15	SM4-ECB 模式解密 .....	417
图 17-16	SM4-CBC 模式加密 .....	418
图 17-17	SM4-CBC 模式解密 .....	419
图 18-1	SM3 结构框图 .....	428
图 19-1	ECC 结构框图 .....	435
图 19-2	ECC 缓存地址映射 .....	436
图 20-1	TRNG 结构框图 .....	440
图 20-2	操作流程示意图 .....	442
图 21-1	CALC 结构框图 .....	450
图 22-1	ADC 结构框图 .....	461
图 22-2	ADC 转换时序图 .....	463
图 22-3	右对齐数据示意图 .....	466
图 22-4	左对齐数据示意图 .....	466
图 23-1	电路结构框图 .....	489
图 24-1	高级定时器电路结构框图 .....	520
图 24-2	预分频值计数时序图 .....	521
图 24-3	重复计数器工作模式 .....	522
图 24-4	采用内部时钟计数 .....	523
图 24-5	I1 外部时钟连接 .....	523
图 24-6	外部触发输入模块 .....	524
图 24-7	ITn 外部时钟连接 .....	525
图 24-8	计数器递增计数时序图 .....	526
图 24-9	当 ARPEN=0 时计数器时序图 .....	526
图 24-10	当 ARPEN=1 时计数器时序图 .....	527
图 24-11	定时器递减计数时序图 .....	528
图 24-12	增减计数器时序图 .....	529
图 24-13	捕获/比较通道 .....	529
图 24-14	捕获/比较通道 1 结构图 .....	530
图 24-15	捕获/比较信道的输出部分 .....	530
图 24-16	PWM 输入模式时序 .....	532
图 24-17	边沿对齐 PWM 波形 (AR=8) .....	533
图 24-18	中心对齐 PWM 波形 (AR=8) .....	534

图 24-19	单脉冲模式.....	536
图 24-20	互补输出含死区时间插入.....	537
图 24-21	刹车输出行为.....	539
图 24-22	编码器接口模式下的计数操作.....	541
图 24-23	I1 滤波后极性反相时编码器接口例子.....	541
图 24-24	复位模式控制电路.....	543
图 24-25	门控模式控制电路.....	544
图 24-26	触发模式控制电路.....	544
图 24-27	外部时钟源 2+触发模式下的控制电路.....	545
图 24-28	6 步 PWM 波形示例.....	546
图 25-1	通用定时器电路结构框图.....	591
图 25-2	预分频值计数时序图.....	592
图 25-3	采用内部时钟计数.....	593
图 25-4	I1 外部时钟连接.....	593
图 25-5	外部触发输入模块.....	594
图 25-6	ITn 外部时钟连接.....	595
图 25-7	计数器时序图, 内部时钟除以 1.....	596
图 25-8	当 ARPEN=0 时计数器时序图.....	597
图 25-9	当 ARPEN=1 时计数器时序图.....	598
图 25-10	定时器递减计数时序图.....	599
图 25-11	增减计数器时序图.....	600
图 25-12	捕获/比较通道.....	600
图 25-13	捕获/比较信道 1 主电路.....	601
图 25-14	捕获/比较通道的输出阶段.....	601
图 25-15	PWM 输入模式时序.....	603
图 25-16	边沿对齐 PWM 波形 (AR=8).....	604
图 25-17	边沿对齐 PWM 波形 (AR=0x3F).....	605
图 25-18	单脉冲模式.....	607
图 25-19	编码器接口模式下的计数操作.....	609
图 25-20	滤波后极性反相时编码器接口.....	609
图 25-21	复位模式控制电路.....	610
图 25-22	门控模式控制电路.....	611
图 25-23	触发模式控制电路.....	612
图 25-24	外部时钟源 2+触发模式下的控制电路.....	613
图 26-1	基本定时器电路结构框图.....	646
图 26-2	预分频值计数时序图.....	647
图 27-1	低功耗定时器电路结构框图.....	657
图 28-1	I2C 电路结构框图.....	672
图 28-2	START 和 STOP 条件.....	673
图 28-3	I2C 总线上的应答.....	674
图 28-4	7 位地址格式.....	675
图 28-5	10 位地址格式.....	675
图 28-6	主机-发送协议.....	676
图 28-7	主机-接收协议.....	676

图 28-8 I2C 总线上的数据传输 .....	677
图 28-9 主时钟产生 .....	679
图 28-10 SCL 主时钟同步 .....	680
图 28-11 主机发送的传输序列图 .....	681
图 28-12 主机接收的传输序列图 .....	682
图 28-13 从机初始化流程图 .....	685
图 28-14 从机发送的传输序列图 .....	686
图 28-15 从机接收的传输序列图 .....	687
图 28-16 I2C 中断映射图 .....	696
图 29-1 SPI 电路结构框图 .....	728
图 29-2 SPI 模式 .....	729
图 29-3 全双工通信 .....	730
图 29-4 全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、RXE、BUSY 行为(直接存取操作模式在连续传输的情况下) .....	733
图 29-5 全双工通信(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、RXTH、TXFLV、RXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下) .....	734
图 29-6 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXE、BUSY 行为(直接存取操作模式在连续传输的情况下) .....	735
图 29-7 单工通信-只发送模式(SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=0)的 TXTH、TXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下) .....	735
图 29-8 单工通信-只接收模式 (SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1) 的 RXE 行为(直接存取操作模式在连续传输的情况下) .....	736
图 29-9 单工通信-只接收模式 (SPI_CON1.BIDEN=0 且 SPI_CON1.RXO=1) 的 RXTH、RXFLV 行为(FIFO 缓存操作模式在连续传输的情况下) .....	736
图 29-10 发送时 (SPI_CON1.BIDEN =0 且 SPI_CON1.RXO=0) 的 TXE/BUSY 行为 (在间断传输的情况下) .....	737
图 29-11 使用 DMA 进行发送 .....	738
图 29-12 使用 DMA 进行接收 .....	738
图 29-13 SPI 兼容模式通信波形 .....	739
图 29-14 I2S 电路结构框图 .....	743
图 29-15 I2S 飞利浦标准波形 (16/32 位数据帧, CPOL = 0) .....	745
图 29-16 I2S 飞利浦标准波形 (24 位数据帧, CPOL = 0) .....	745
图 29-17 发送 0x123456 .....	745
图 29-18 接收 0x123456 .....	746
图 29-19 I2S 飞利浦标准波形 (16 位数据帧扩展到 32 位通道帧, CPOL = 0) .....	746
图 29-20 16 位数据帧扩展到 32 位通道帧的示例 .....	746
图 29-21 MSB 对齐标准波形 (16/32 位数据帧, CPOL = 0) .....	747
图 29-22 MSB 对齐标准波形 (24 位数据帧, CPOL = 0) .....	747
图 29-23 MSB 对齐标准波形 (16 位数据帧扩展到 32 位通道帧, CPOL = 0) .....	747
图 29-24 LSB 对齐标准波形 (16/32 位数据帧, CPOL = 0) .....	748
图 29-25 LSB 对齐标准波形 (24 位数据帧, CPOL = 0) .....	748
图 29-26 传输 0x123456 所需的操作 .....	748
图 29-27 接收 0x123456 所需的操作 .....	749
图 29-28 LSB 对齐标准波形 (16 位数据帧扩展到 32 位通道帧, CPOL = 0) .....	749

图 29-29	16 位数据帧扩展到 32 位通道帧的示例 .....	749
图 29-30	PCM 标准波形 (16 位) .....	750
图 29-31	PCM 标准波形 (16 位数据帧扩展到 32 位通道帧) .....	750
图 29-32	音频采样频率定义 .....	751
图 30-1	UART 框图 .....	781
图 30-2	数据宽度设置 .....	784
图 30-3	配置停止位 .....	785
图 30-4	防抖动波形 .....	787
图 30-5	防抖动输出 .....	787
图 30-6	起始位检测 .....	788
图 30-7	数值采样 .....	789
图 30-8	自动波特率检测模式 0 .....	796
图 30-9	自动波特率检测模式 1 .....	796
图 30-10	自动波特率检测模式 2 .....	797
图 30-11	自动流控制框图 .....	798
图 30-12	自动 RTSn 控制 .....	798
图 30-13	自动 CTSn 控制 .....	799
图 30-14	驱动使能当 AADINV=0 .....	799
图 30-15	使用地址标示检测模式 .....	801
图 30-16	LIN 模式下断开信号检测 (11 位断开长度-LBDL 位为 1) .....	803
图 30-17	LIN 模式下的断开检测与帧错误的检测 .....	804
图 30-18	ISO 7816-3 异步协定 .....	805
图 30-19	用 1.5 位停止位时检测校验错误 .....	806
图 30-20	红外收发框图 .....	807
图 30-21	IrDA 数据调制 (3/16) -正常模式 .....	808
图 31-1	Flash QSPI Controller 结构框图 .....	843
图 31-2	Flash QSPI Controller 时钟结构 .....	843
图 32-1	ADPCM 译码功能框图 .....	897
图 33-1	结构框图 .....	904
图 33-2	4x4 Keyboard Matrix 示意图 .....	906
图 34-1	结构框图 .....	917
图 34-2	QDEC 采集的输入信号 .....	918
图 34-3	采样译码 .....	918
图 34-4	前沿采样模式 (正向移动) .....	919
图 34-5	前沿采样模式 (反向移动) .....	919
图 34-6	双沿采样模式 (正向移动) .....	919
图 34-7	双沿采样模式 (反向移动) .....	919
图 34-8	后沿采样模式 (正向移动) .....	920
图 34-9	后沿采样模式 (反向移动) .....	920
图 34-10	Rambol 正向采样示意图 .....	921
图 34-11	Rambol 反向采样示意图 .....	922
图 34-12	光电模式管脚输出示意图 .....	922
图 35-1	触控按键内部结构示意图 .....	926
图 35-2	无触摸状态示意图 .....	927



图 35-3	有触摸状态示意图 .....	928
图 35-4	触摸按键操作流程 .....	929
图 36-1	电路结构框图 .....	939
图 36-3	MEM-AP 地址映射 .....	941
图 37-1	电路结构框图 .....	947
图 37-2	PDM2PCM 模块示意图 .....	947
图 37-3	CIC 滤波器示意图 .....	948
图 38-1	BLE Core 结构框图 .....	982
图 38-2	Core 进入 Deep Sleep Mode .....	983
图 38-3	BLE Core Deep Sleep Mode 自动超时唤醒 (extwkupdsb=1) .....	983
图 38-4	BLE Core Deep Sleep Mode 自动超时唤醒 (extwkupdsb=0) .....	984
图 38-5	BLE Core Deep Sleep Mode 异步事件唤醒 .....	985
附录图 39-1	Cortex-M3 的典型应用 .....	1014
附录图 39-2	Cortex-M3 指令 .....	1021
附录图 39-3	用来生成一些 Cortex-M3 指令的 CMSIS 内在函数 .....	1022
附录图 39-4	用来访问专用寄存器的 CMSIS 内在函数 .....	1023
附录图 39-5	ASR #3 .....	1025
附录图 39-6	LSR #3 .....	1025
附录图 39-7	LSL #3 .....	1026
附录图 39-8	ROR #3 .....	1026
附录图 39-9	RRX .....	1027
附录图 39-10	条件代码后缀 .....	1030
附录图 39-11	内存访问指令 .....	1031
附录图 39-12	偏移量范围 .....	1034
附录图 39-13	偏移量范围 .....	1038
附录图 39-14	数据处理指令 .....	1045
附录图 39-15	乘法和除法指令 .....	1056
附录图 39-16	组合和分离指令 .....	1061
附录图 39-17	跳转和控制指令 .....	1064
附录图 39-18	跳转范围 .....	1065
附录图 39-19	其他指令 .....	1070
附录图 39-20	处理器模式、执行特权级别和堆栈使用选择汇总 .....	1078
附录图 39-21	内核寄存器 .....	1078
附录图 39-22	内核寄存器集汇总 .....	1079
附录图 39-23	APSR、IPSR 和 EPSR .....	1079
附录图 39-24	组合 xPSR .....	1080
附录图 39-25	PSR 寄存器组合 .....	1080
附录图 39-26	APSR 位分配 .....	1081
附录图 39-27	IPSR 位分配 .....	1082
附录图 39-28	EPSR 位分配 .....	1082
附录图 39-29	PRIMASK 寄存器的位分配 .....	1083
附录图 39-30	FAULTMASK 寄存器的位分配 .....	1083
附录图 39-31	BASEPRI 寄存器的位分配 .....	1083
附录图 39-32	控制寄存器的位分配 .....	1084

附录图 39-33	Cortex-M3 处理器预定义的存储器映射 .....	1086
附录图 39-34	存储器排序限制 .....	1087
附录图 39-35	存储器访问行为 .....	1088
附录图 39-36	SRAM 存储器位段区 .....	1090
附录图 39-37	外设存储器位段区 .....	1090
附录图 39-38	位段映射 .....	1091
附录图 39-39	小端格式 .....	1092
附录图 39-40	独占访问指令的 C 编译器内在函数 .....	1093
附录图 39-41	不同异常类型的属性 .....	1096
附录图 39-42	向量表 .....	1097
附录图 39-43	异常返回行为 .....	1101
附录图 39-44	故障 .....	1101
附录图 39-45	故障状态和故障地址寄存器 .....	1102
附录图 39-46	内核外设寄存器区 .....	1106
附录图 39-47	NVIC 寄存器汇总 .....	1107
附录图 39-48	中断到中断变量的映射 .....	1108
附录图 39-49	ISER 位分配 .....	1109
附录图 39-50	ICER 位分配 .....	1109
附录图 39-51	ISPR 位分配 .....	1110
附录图 39-52	ICPR 位分配 .....	1110
附录图 39-53	IABR 位分配 .....	1111
附录图 39-54	中断优先级寄存器 .....	1112
附录图 39-55	IPR 位分配 .....	1112
附录图 39-56	STIR 位分配 .....	1113
附录图 39-57	用于 NVIC 控制的 CMSIS 函数 .....	1114
附录图 39-58	系统控制模块寄存器汇总 .....	1115
附录图 39-59	ACTLR 位分配 .....	1116
附录图 39-60	CPUID 寄存器的位分配 .....	1116
附录图 39-61	ICSR 位分配 .....	1118
附录图 39-62	VTOR 位分配 .....	1119
附录图 39-63	AIRCR 位分配 .....	1120
附录图 39-64	优先级分组 .....	1120
附录图 39-65	SCR 位分配 .....	1121
附录图 39-66	CCR 位分配 .....	1122
附录图 39-67	系统故障处理程序优先级域 .....	1123
附录图 39-68	SHPR1 寄存器的位分配 .....	1123
附录图 39-69	SHPR2 寄存器的位分配 .....	1123
附录图 39-70	SHPR3 寄存器的位分配 .....	1124
附录图 39-71	SHCSR 位分配 .....	1124
附录图 39-72	可配置故障状态寄存器 .....	1125
附录图 39-73	MMFSR 位分配 .....	1127
附录图 39-74	BFSR 位分配 .....	1128
附录图 39-75	UFSR 位分配 .....	1130
附录图 39-76	HFSR 位分配 .....	1130

附录图 39-77	MMFAR 位分配.....	1131
附录图 39-78	BFAR 位分配 .....	1131
附录图 39-79	系统定时器寄存器汇总 .....	1132
附录图 39-80	SysTick 控制寄存器的位分配 .....	1132
附录图 39-81	加载寄存器的位分配.....	1133
附录图 39-82	VAL 寄存器的位分配.....	1133
附录图 39-83	CALIB 寄存器的位分配.....	1134
附录图 39-84	存储器属性汇总 .....	1135
附录图 39-85	MPU 寄存器汇总.....	1135
附录图 39-86	类型寄存器的位分配.....	1136
附录图 39-87	MPU 控制寄存器的位分配.....	1137
附录图 39-88	RNR 位分配 .....	1137
附录图 39-89	RBAR 位分配.....	1138
附录图 39-90	RASR 位分配.....	1139
附录图 39-91	SIZE 域值示例 .....	1140
附录图 39-92	TEX、C、B 和 S 的编码.....	1141
附录图 39-93	存储器属性编码的缓存策略 .....	1141
附录图 39-94	AP 编码.....	1141
附录图 39-95	SRD 使用示例.....	1144
附录图 39-96	微控制器的存储器属性 .....	1144

## 表目录

表 2-1	系统模块功能 .....	50
表 2-2	ARM 32 位 Cortex-M3 内核模块 .....	50
表 2-3	存储器及存储接口 .....	50
表 2-4	系统模块 .....	51
表 2-5	时钟管理 .....	51
表 2-6	外部接口 .....	51
表 2-7	安全管理 .....	52
表 2-8	模拟 .....	52
表 2-9	定时器 .....	54
表 2-10	通信 .....	55
表 3-1	ARM Cortex-M3 内核配置的参考信息链接 .....	56
表 3-2	嵌套向量中断控制器的参考信息链接 .....	57
表 3-3	中断向量分配 .....	60
表 3-4	异步唤醒中断的参考信息链接 .....	60
表 3-5	STOP1 低功耗模式的中断唤醒源 .....	61
表 3-6	STOP2 低功耗模式的中断唤醒源 .....	61
表 3-7	STANDBY 低功耗模式的中断唤醒源 .....	61
表 3-8	事件唤醒源 .....	62
表 3-9	总线架构和存储器的参考信息链接 .....	62
表 3-10	存储器系统控制的参考信息链接 .....	62
表 3-11	扩展外部存储器的参考信息链接 .....	63
表 3-12	系统配置控制器的参考信息链接 .....	63
表 3-13	电源管理及低功耗模式的参考信息链接 .....	63
表 3-14	复位管理的参考信息链接 .....	64
表 3-15	电源管理及低功耗模式的参考信息链接 .....	64
表 3-16	DMA 请求列表 .....	66
表 3-17	外设互联(PIS)的参考信息链接 .....	67
表 3-18	独立看门狗定时器的参考信息链接 .....	67
表 3-19	窗口看门狗定时器的参考信息链接 .....	68
表 3-20	时钟管理的参考信息链接 .....	68
表 3-21	通用 IO 及端口控制的参考信息链接 .....	69
表 3-22	循环冗余校验(CRC)的参考信息链接 .....	70
表 3-24	真随机数发生器的参考信息链接 .....	70
表 3-25	真随机数发生器的参考信息链接 .....	70
表 3-26	ADC 控制的参考信息链接 .....	71
表 3-27	高级定时器的参考信息链接 .....	72
表 3-28	通用定时器的参考信息链接 .....	72
表 3-29	基本定时器的参考信息链接 .....	73
表 3-31	RTC 的参考信息链接 .....	74
表 3-32	I2C 接口的参考信息链接 .....	75
表 3-34	通用异步收发器(UART)的参考信息链接 .....	76
表 4-1	系统存储器映射 .....	79
表 4-2	外设存储映射 .....	82

表 4-3	私有外设存储器映射 .....	83
表 4-4	启动引导控制说明 .....	85
表 5-1	Flash 写保护对应区域表 .....	87
表 5-2	Flash 全局读保护级别 .....	87
表 5-3	Flash 全局读保护控制权限 .....	88
表 5-4	数据 Flash 配置 .....	88
表 7-1	低功耗模式下各模块操作 .....	127
表 7-2	低功耗模式下唤醒源 .....	129
表 9-1	POR/BOR 复位与寄存器关系 .....	151
表 9-2	MRSTn/WDT 复位与寄存器关系 .....	152
表 9-3	系统复位与寄存器关系 .....	153
表 11-1	仲裁设置 .....	211
表 11-2	DMA 通道优先级 .....	212
表 11-3	DMA 周期类型 .....	214
表 11-4	存储器分散-聚集数据结构 .....	218
表 11-5	外设分散-聚集数据结构 .....	221
表 11-6	控制信息格式描述 .....	228
表 11-7	传输过程中指针变化 .....	230
表 11-8	传输过程中指针变化 .....	230
表 12-1	生产端信号 .....	290
表 12-2	消费端信号 .....	291
表 12-3	消费端的 PIS 通道分配 .....	293
表 15-1	端口配置表 .....	371
表 15-2	端口拉电流驱动表 .....	371
表 15-3	端口灌电流驱动表 .....	371
表 20-1	随机数生成时间系数表 .....	441
表 21-1	平方根运算误差示例 .....	451
表 21-2	平方根运算时间表 .....	452
表 21-3	除法运算时间表 .....	454
表 22-1	模拟看门狗通道选择 .....	464
表 22-3	ADC 中断 .....	468
表 23-1	小时格式对照表 .....	494
表 24-1	计数方向与编码器信号的关系 .....	540
表 25-1	计数方向与编码器信号的关系 .....	608
表 28-1	10 位地址格式第一个字节中位的定义 .....	675
表 28-2	FI2CCLK = 8 MHz 的时序设置示例 .....	688
表 28-3	FI2CCLK = 16 MHz 的时序设置示例 .....	688
表 28-4	FI2CCLK = 48 MHz 的时序设置示例 .....	689
表 28-5	SMBus 超时规格 .....	691
表 28-6	各种 I2CCLK 频率的 TIMEOUTA 设置示例 (最大值 TTIMEOUT = 25 ms) .....	693
表 28-7	各种 I2CCLK 频率的 TIMEOUTB 设置示例 (最大值 TTIMEOUT = 8 ms) .....	693
表 28-8	各种 I2CCLK 频率的 TIMEOUTA 设置示例 (最大 TIDLE = 50μs) .....	693
表 29-1	不同时钟配置的示例精度值 .....	752
表 30-1	来自采样数据的噪音检测 .....	791

---

表 30-2	时钟为 48MHz 下，设置波特率时的误差计算 .....	794
表 30-3	帧格式 .....	800
表 30-4	中断配置表 .....	811
表 31-1	SRAM 访问优先级 .....	852

## 第1章 文档约定

### 1.1 寄存器读写权限的设定

缩写词	说明	描述
R/W	读/写 ( __IO )	软件可以读写这些位
R	只读 ( __I )	软件只能读取这些位
W	只写 ( __O )	软件只能写入该位，读取该位时将返回复位值
W1	只写 ( 写 1 )	软件只能写入该位，写 1 有效，写 0 无作用。
R/C_W1	读取/清零 ( 写 1 )	软件可以读取该位，也可以通过写入 1 将该位清零。写入 “0” 对该位的值无影响
R/C_W0	读取/清零 ( 写 0 )	软件可以读取该位，也可以通过写入 0 将该位清零。写入 “1” 对该位的值无影响
R/C_R	读取/清零 ( 读取 )	软件可以读取该位。读取该位时，将自动清零。写入 “0” 对该位的值无影响
C_W1	清零 ( 写 1 )	通过写入 1 将该位清零。写入 “0” 对该位的值无影响
S_W1	置位 ( 写 1 )	通过写入 1 将该位置位。写入 “0” 对该位的值无影响
C_W0	清零 ( 写 0 )	通过写入 0 将该位清零。写入 “1” 对该位的值无影响
T_W1	触发 ( 写 1 )	通过写入 1 将触发硬件动作。写入 “0” 对该位的值无影响
Reserved	保留	保留位，必须保持复位值。

## 第2章 系统概述

### 2.1 概述

该章节从系统层描述 ES32W3120 系列微控制器所涵盖的功能。

### 2.2 系统框图

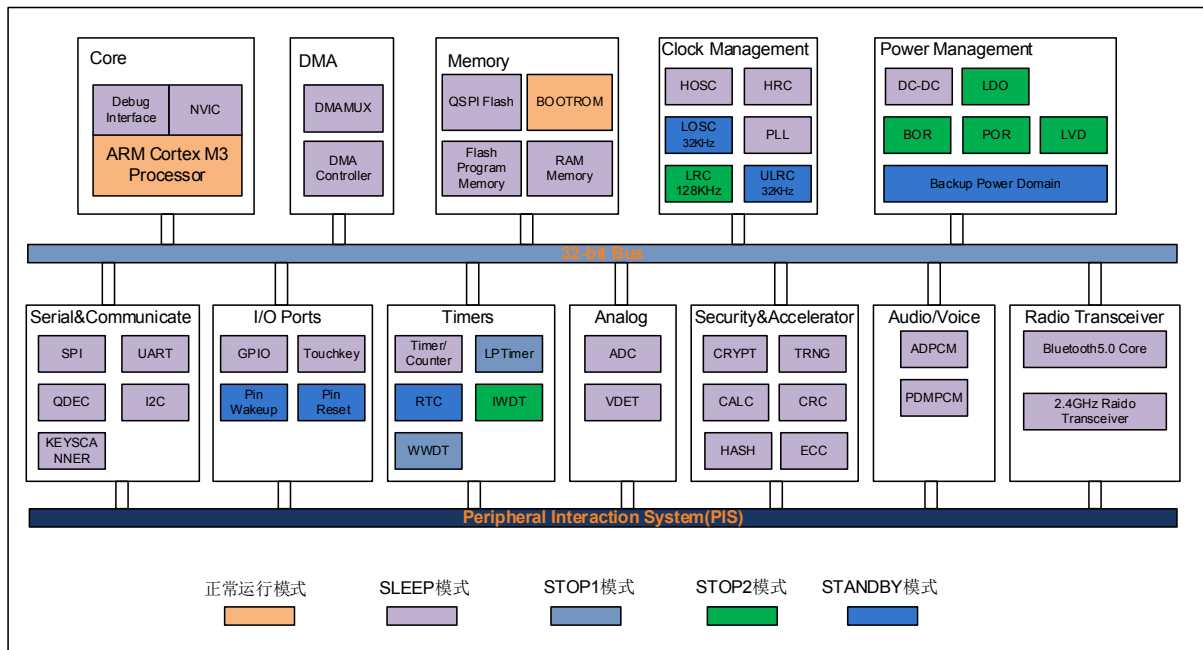


图 2-1 系统框图

注：数据手册提供了更为详尽的系统结构框图，可通过数据手册了解外设资源的具体数量和芯片全局的系统总线结构等。



## 2.3 模块功能类别

类别	描述
ARM 32 位 Cortex-M3 CPU	<ul style="list-style-type: none"> <li>◆ ARM Cortex M 系列 32 位 MCU 内核, 最高频率可达 48MHz</li> <li>◆ 调试</li> <li>◆ 中断和事件</li> </ul>
存储	<ul style="list-style-type: none"> <li>◆ 内部存储:                             <ul style="list-style-type: none"> <li>◇ FLASH 存储器(大小 512KB)</li> <li>◇ SRAM 数据存储(大小 64KB)</li> </ul> </li> <li>◆ 支持外部扩展:                             <ul style="list-style-type: none"> <li>◇ QSPI 可灵活扩展各种外部存储(QSPI Flash)</li> </ul> </li> <li>◆ 系统总线和存储器</li> <li>◆ 存储器系统控制</li> </ul>
系统管理	<ul style="list-style-type: none"> <li>◆ 系统配置控制器</li> <li>◆ 电源管理及低功耗模式                             <ul style="list-style-type: none"> <li>◇ 可支持多种低功耗模式: 运行模式, SLEEP 模式, STOP 模式和 STANDBY 模式</li> </ul> </li> <li>◆ 复位控制</li> <li>◆ DMA                             <ul style="list-style-type: none"> <li>◇ 支持多个 DMA 通道, DMAMUX 为每个 DMA 通道对应多个 DMA 请求选择</li> </ul> </li> <li>◆ 外设互联(PIS)</li> <li>◆ 看门狗定时器                             <ul style="list-style-type: none"> <li>◇ 独立看门狗定时器</li> <li>◇ 窗口看门狗定时器</li> </ul> </li> </ul>
时钟管理	<ul style="list-style-type: none"> <li>◆ 提供外部和内部多种时钟源选择</li> <li>◆ PLL 倍频满足高速应用</li> <li>◆ 可灵活选择内核, 系统及外设时钟</li> <li>◆ 可灵活设置外设时钟门控及时钟分频, 以满足低功耗应用需求</li> </ul>
外部接口	<ul style="list-style-type: none"> <li>◆ 通用 IO(GPIO)</li> </ul>
安全管理及运算加速	<ul style="list-style-type: none"> <li>◆ 循环冗余校验模块</li> <li>◆ 加密处理</li> <li>◆ 真随机数发生器</li> <li>◆ 运算加速器</li> </ul>
模拟	<ul style="list-style-type: none"> <li>◆ 模数转换 ADC</li> <li>◆ 温度传感器</li> <li>◆ 触控按键</li> <li>◆ 内部参考源</li> </ul>
定时器	<ul style="list-style-type: none"> <li>◆ 高级定时器</li> <li>◆ 普通定时器</li> <li>◆ 基本定时器</li> <li>◆ 低功耗定时器</li> <li>◆ RTC</li> </ul>

类别	描述
通信	<ul style="list-style-type: none"> <li>◆ I2C 接口</li> <li>◆ 串行外设接口(SPI)</li> <li>◆ 通用异步收发器(UART)</li> <li>◆ BLE5.0</li> <li>◆ QUAD</li> <li>◆ KEYSKAN</li> </ul>
音频	<ul style="list-style-type: none"> <li>◆ ADPCM</li> <li>◆ PDMPCM</li> </ul>

表 2-1 系统模块功能

### 2.3.1 ARM 32 位Cortex-M3 内核模块

ES32W3120 系列微控制器内核模块包含以下功能：

模块	描述
ARM 32 位 Cortex-M3 内核	<ul style="list-style-type: none"> <li>◆ 支持 Thumb 指令集；</li> <li>◆ 带 ICode, DCode 和 System 总线接口；</li> <li>◆ 支持快速中断响应；</li> <li>◆ 支持硬件除法指令 SDIV 和 UDIV；</li> <li>◆ 支持位带操作</li> </ul>
NVIC	<ul style="list-style-type: none"> <li>◆ 中断使能控制；</li> <li>◆ 中断优先级设置；</li> <li>◆ 支持末尾连锁和迟来；</li> <li>◆ 支持 32 个外部中断向量</li> </ul>
WIC	<ul style="list-style-type: none"> <li>◆ 中断唤醒控制模块</li> </ul>
PMU	<ul style="list-style-type: none"> <li>◆ 内核功耗管理模块，请结合 ARM 技术参考手册和本文档的章节“电源管理及低功耗模式”阅读，这两部分共同组成芯片整体的低功耗模式控制。</li> </ul>
调试接口	<ul style="list-style-type: none"> <li>◆ SWD 协议调试接口</li> </ul>
系统节拍定时器	<ul style="list-style-type: none"> <li>◆ 内核系统用定时器</li> </ul>

表 2-2 ARM 32 位 Cortex-M3 内核模块

### 2.3.2 存储器及存储器接口

ES32W3120 系列微控制器包含以下存储器及存储器接口模块：

模块	描述
系统总线和存储器	<ul style="list-style-type: none"> <li>◆ 系统总线类型及总线架构，存储器类型及地址空间映射。</li> </ul>
存储器系统控制	<ul style="list-style-type: none"> <li>◆ FLASH 存储器的访问控制</li> </ul>
FLASH 存储	<ul style="list-style-type: none"> <li>◆ FLASH 存储</li> </ul>
SRAM	<ul style="list-style-type: none"> <li>◆ SRAM 存储</li> </ul>
QSPI FLASH	<ul style="list-style-type: none"> <li>◆ 扩展外部串行存储器</li> </ul>

表 2-3 存储器及存储接口

### 2.3.3 系统模块

ES32W3120 系列微控制器包含以下系统模块：

模块	描述
系统配置控制器	◆ 系统的相关配置
电源管理及低功耗模式	◆ 系统电源的管理，及低功耗模式控制
复位控制	◆ 系统所有复位的管理
DMA 多路复用(DMAMUX)	◆ DMA 请求源的多路复用选择器
DMA 控制器	◆ DMA 控制器可减少 CPU 负荷，提高系统运行效率。在低功耗场合也可代替 CPU 的部分工作而不必唤醒整个系统，节省功耗。
外设互联(PIS)	◆ 外设互联系统为外设提供互联接口，可减少软件负担，提高了系统响应的及时性，同时为扩展应用场景提供了便利和灵活性。
看门狗定时器	◆ 包含了独立看门狗和窗口看门狗

表 2-4 系统模块

### 2.3.4 时钟管理

ES32W3120 系列微控制器包含以下时钟管理模块：

模块	描述
时钟管理	<ul style="list-style-type: none"> <li>◆ 时钟源配置</li> <li>◆ 系统和外设时钟的选择及切换</li> <li>◆ 外设时钟门控</li> <li>◆ 系统和外设时钟分频</li> <li>◆ PLL 倍频时钟</li> </ul>

表 2-5 时钟管理

### 2.3.5 外部接口

ES32W3120 系列微控制器包含以下外部接口模块：

模块	描述
通用 IO 及端口控制	<ul style="list-style-type: none"> <li>◆ 通用 IO 的输入输出</li> <li>◆ 对端口的控制包括：上、下拉选择，开漏/开源选择，驱动能力选择，端口模拟滤波器使能等。</li> </ul>

表 2-6 外部接口

### 2.3.6 安全管理及运算加速

ES32W3120 系列微控制器包含以下安全管理模块：

模块	描述
循环冗余校验模块(CRC)	<ul style="list-style-type: none"> <li>◆ 支持四个常用的多项式：                             <ul style="list-style-type: none"> <li>◇ CRC-CCITT</li> <li>◇ CRC-8</li> <li>◇ CRC-16</li> <li>◇ CRC-32</li> </ul> </li> </ul>
加密处理(CRYPT)	◆ 支持的标准有 AES、DES
真随机数发生器(TRNG)	◆ 可生产 1 位串行真随机数或 8/16/32 位并行真随机数。
伪随机数发生器	◆ 不支持
运算加速器(CALC)	◆ 用于执行平方根的运算加速。
HASH 硬件加速器	◆ 支持 SHA256、SM3 算法

表 2-7 安全管理

### 2.3.7 模拟

ES32W3120 系列微控制器包含以下模拟模块：

模块	描述
模数转换 (ADC)	◆ 12 位逐次逼近型模数转换器
温度传感器 (TEMP)	◆ 可生成与温度成线性关系的电压，并由 ADC 转换成数字量输出。
内部参考电压源 (AVREF)	

表 2-8 模拟

### 2.3.8 定时器

ES32W3120 系列微控制器包含以下定时器模块：

模块	描述
高级定时器 (AD16C4T)	<ul style="list-style-type: none"> <li>◆ 16 位递增、递减、递增/递减自动重载计数器</li> <li>◆ 16 位可编程预分频器,用于对计数器时钟频率进行分频(即运行时修改),分频系数 介于 1 到 65536 之间</li> <li>◆ 多达 4 个独立通道,可用于:                             <ul style="list-style-type: none"> <li>◇ 输入捕获</li> <li>◇ 输出比较</li> <li>◇ PWM 生成 (边沿和中心对齐模式)</li> <li>◇ 单脉冲模式输出</li> </ul> </li> <li>◆ 带可编程死区的互补输出</li> <li>◆ 使用外部信号控制定时器且可实现多个定时器互联的同步电路。</li> <li>◆ 重复计数器,用于仅在给定数目的计数器周期后更新定时器</li> </ul>

模块	描述
	<p>寄存器</p> <ul style="list-style-type: none"> <li>◆ 用于将定时器的输出信号置于复位状态或已知状态的刹车输入</li> <li>◆ 发生如下事件时生成中断/DMA 请求： <ul style="list-style-type: none"> <li>◇ 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）</li> <li>◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）</li> <li>◇ 输入捕获</li> <li>◇ 输出比较</li> <li>◇ 刹车输入</li> </ul> </li> <li>◆ 支持定位用增量（正交）编码及霍尔传感器电路</li> </ul>
通用定时器（GP16C4T）	<ul style="list-style-type: none"> <li>◆ 16 位递增、递减、递增/递减自动重载计数器。</li> <li>◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数 介于 1 到 65536 之间。</li> <li>◆ 多达 4 个独立通道，可用于： <ul style="list-style-type: none"> <li>◇ 输入捕获</li> <li>◇ 输出比较</li> <li>◇ PWM 生成（边沿和中心对齐模式）</li> <li>◇ 单脉冲模式输出</li> </ul> </li> <li>◆ 使用外部信号控制定时器且可实现多个定时器互连的同步电路。</li> <li>◆ 发生如下事件时生成中断/DMA 请求： <ul style="list-style-type: none"> <li>◇ 更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）</li> <li>◇ 触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）</li> <li>◇ 输入捕获</li> <li>◇ 输出比较</li> </ul> </li> <li>◆ 支持定位用增量（正交）编码器和霍尔传感器电路。</li> <li>◆ 外部时钟触发输入</li> </ul>
基本定时器（BS16T）	<ul style="list-style-type: none"> <li>◆ 16 位递增自动重载计数器。</li> <li>◆ 16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数 介于 1 到 65536 之间。</li> <li>◆ 发生如下事件时生成中断/DMA 请求： <ul style="list-style-type: none"> <li>◇ 更新：计数器上溢，计数器初始化（透过软件）</li> </ul> </li> </ul>

模块	描述
低功耗定时器 (LP16T)	<ul style="list-style-type: none"> <li>◆ 16 位向上计数</li> <li>◆ 3 位预分频器支持 8 种分频系数 (1,2,4,8,16,32,64,128)</li> <li>◆ 时钟源                             <ul style="list-style-type: none"> <li>◇ 内部时钟源: PCLK2,HRC,LRC,LOSC,HOSC,HSCO,LOSC M</li> <li>◇ 外部时钟源: 外部端口输入</li> </ul> </li> <li>◆ 16 位 ARR 自动加载寄存器</li> <li>◆ 16 位比较寄存器</li> <li>◆ 连续或单发模式可选</li> <li>◆ 软件或硬件触发可选</li> <li>◆ 可编程滤波器</li> <li>◆ 可配置输出: 脉冲, PWM, 翻转</li> <li>◆ 输出极性可配</li> </ul>
实时时钟 (RTC)	<ul style="list-style-type: none"> <li>◆ 时间计数 (实现小时、分钟、秒和亚秒) 和日历计数 (实现年、月、日和星期), 采用 BCD 格式</li> <li>◆ 闹钟可输出</li> <li>◆ 闹钟支持掩码功能</li> <li>◆ 支持时间戳功能</li> <li>◆ 发生入侵检测事件时, 将复位备份寄存器</li> </ul>

表 2-9 定时器

### 2.3.9 通信

ES32W3120 系列微控制器包含以下通信模块:

模块	描述
内部集成电路总线 (I2C)	<ul style="list-style-type: none"> <li>◆ 支持多主模式和总线仲裁</li> <li>◆ 可编程 I2C 地址检测</li> <li>◆ 最高通信速率为 1MHz</li> <li>◆ 可配置时钟延长</li> <li>◆ 16 级深度 FIFO</li> <li>◆ 兼容 SMBus2.0 协议</li> <li>◆ 兼容 PMBus</li> </ul>
串行外设接口(SPI)	<ul style="list-style-type: none"> <li>◆ 支持半双工/全双工的同步串行通信</li> <li>◆ 主模式或从模式操作</li> <li>◆ 8 位或 16 位传输帧格式选择</li> <li>◆ 16 级深度 FIFO</li> <li>◆ I2S 主机模式通信</li> </ul>
通用异步收发器(UART)	<ul style="list-style-type: none"> <li>◆ 支持与外部设备进行全双工数据通信和单线半双工通信</li> <li>◆ 支持波特率自动测量功能</li> </ul>

模块	描述
	<ul style="list-style-type: none"> <li>◆ 提供 16 级深度接收和发送 FIFO</li> <li>◆ 支持多点通信 (RS-485)</li> <li>◆ LIN (局域互连网络)</li> <li>◆ 红外通信协议</li> <li>◆ 自动硬件流控制 (CTS/RTS)</li> </ul>

表 2-10 通信

## 第3章 芯片配置指引

### 3.1 概述

本章节主要说明以下内容：

- ◇ 芯片顶层相关模块的连接及信号路径
- ◇ 阅读各模块时可参考的相关信息链接
- ◇ 芯片顶层连接资源的相关配置
- ◇ 模块之间特殊的交互，该部分内容在单独的模块说明章节中不再赘述。

### 3.2 ARM Cortex-M3 内核配置

关于 ARM Cortex-M3 内核配置的相关参考信息链接如下表：

主题	相关模块	参考链接
ARM Cortex-M3 内核说明	ARM Cortex-M3 core	<a href="#">ARM Cortex-M3 技术参考资料链接</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
低功耗模式		<a href="#">电源管理及低功耗模式</a>
系统总线矩阵		<a href="#">总线矩阵</a>
调试	串行单线调试	<a href="#">ARM Cortex-M3 技术参考资料链接</a>
中断	可嵌套中断控制器(NVIC)	<a href="#">ARM Cortex-M3 技术参考资料链接</a>

表 3-1 ARM Cortex-M3 内核配置的参考信息链接

#### 3.2.1 ARM Cortex-M3 内核

关于 ARM Cortex-M3 内核技术细节可参考 ARM 官网

#### 3.2.2 总线

32 位 AMBA3 AHB 总线矩阵。Cortex-M3 内核提供了 ICode, DCode 和 System 三条总线，可进行高效的指令预取和对外设访问，并且快速响应中断。

#### 3.2.3 系统节拍定时器

系统节拍定时器为递减计数器，计数时钟可选择内核时钟或内核时钟的 8 分频。具体可参考系统节拍控制和状态寄存器(SysTick Control and Status Register)相关说明。

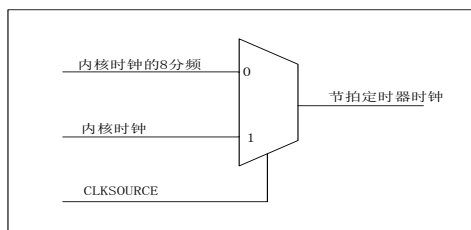


图 3-1 系统节拍定时器时钟



### 3.2.4 调试器件

支持标准 SWD 协议的调试接口。详细使用请参考 DBGC 章节

## 3.3 嵌套向量中断控制器

关于嵌套向量中断控制器的相关参考信息链接如下表：

主题	相关模块	参考链接
NVIC 说明	嵌套向量中断控制器	<a href="#">ARM Cortex-M3 技术参考资料链接</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
私有外设总线	ARM Cortex-M3 core	<a href="#">ARM Cortex-M3 技术参考资料链接</a>

表 3-2 嵌套向量中断控制器的参考信息链接

### 3.3.1 中断优先级

中断优先级寄存器(Interrupt Priority Register)每个 byte 的高 4 位为有效位，支持 16 个中断优先级设置。

### 3.3.2 中断向量分配

中断向量分配如下表所示：

编号	优先级	名称	说明	地址
0	-	-	保留	0x0000_0000
1	-3	Reset	复位	0x0000_0004
2	-2	NMI	不可屏蔽中断。 配置字取反错误	0x0000_0008
3	-1	HardFault	所有类型的错误	0x0000_000C
4	可设置	MemManage	存储器管理	0x0000_0010
5	可设置	BusFault	预取指失败，存 储器访问失败	0x0000_0014
6	可设置	UsageFault	未定义的指令或 非法状态	0x0000_0018
7~10	-	-	-	0x0000_001C~ 0x0000_002B
11	-	SVCall	通过 SWI 指令 调用的系统服务	0x0000_002C
12	-	DebugMonitor	调试监视器	0x0000_0030
13	-	-	保留	0x0000_0034
14	-	PendSV	可挂起的系统服 务	0x0000_0038
15	可设置	Systick	系统定时器	0x0000_003C
16	可设置	WWDG	WWDG 全局	0x0000_0040
17	可设置	IWDG	IWDG 全局	0x0000_0044

编号	优先级	名称	说明	地址
18	可设置	LVD	LVD 全局	0x0000_0048
19	可设置	RTC_WAKE	RTC 事件	0x0000_004C
20	可设置	RTC_ALARM	RTC 闹钟	0x0000_0050
21	可设置	-	预留	0x0000_0054
22	可设置	CMU	CMU 全局	0x0000_0058
23	可设置	ADC	ADC 全局	0x0000_005C
24	可设置	LP16T	低功耗定时器中 断	0x0000_0060
25	可设置	-	预留	0x0000_0064
26	可设置	BS16T0_IRQ	基本定时器 0 中 断	0x0000_0068
27	可设置	BS16T1_IRQ	基本定时器 1 中 断	0x0000_006C
28	可设置	GP16C4T0_IRQ	普通定时器 0 中 断	0x0000_0070
29	可设置	-	预留	0x0000_0074
30	可设置	-	预留	0x0000_0078
31	可设置	-	预留	0x0000_007C
32	可设置	GP16C4T1_IRQ	普通定时器 1 中 断	0x0000_0080
33	可设置	-	预留	0x0000_0084
34	可设置	-	预留	0x0000_0088
35	可设置	-	预留	0x0000_008C
36	可设置	AD16C4T0_BRK	高级定时器 0 刹 车中断	0x0000_0090
37	可设置	AD16C4T0_UP	高级定时器 0 更 新中断	0x0000_0094
38	可设置	AD16C4T0_TRIG_COM	高级定时器 0 触 发、通信中断	0x0000_0098
39	可设置	AD16C4T0_CC	高级定时器 0 捕 获/比较中断	0x0000_009C
40	可设置	AD16C4T1_BRK	高级定时器 0 刹 车中断	0x0000_00A0
41	可设置	AD16C4T1_UP	高级定时器 1 更 新中断	0x0000_00A4
42	可设置	AD16C4T1_TRIG_COM	高级定时器 1 触 发、通信中断	0x0000_00A8
43	可设置	AD16C4T1_CC	高级定时器 1 捕 获/比较中断	0x0000_00AC
44	可设置	I2C0_EV	I2C0 事件	0x0000_00B0
45	可设置	I2C0_ERR	I2C0 错误	0x0000_00B4
46	可设置	I2C1_EV	I2C1 事件	0x0000_00B8

编号	优先级	名称	说明	地址
47	可设置	I2C1_ERR	I2C1 错误	0x0000_00BC
48	可设置	SPI0	全局	0x0000_00C0
49	可设置	SPI1	全局	0x0000_00C4
50	可设置	UART0	全局	0x0000_00C8
51	可设置	UART1	全局	0x0000_00CC
52	可设置	UART2	全局	0x0000_00D0
53	可设置	-	预留	0x0000_00D4
54	可设置	PDM2PCM	全局	0x0000_00D8
55	可设置	TK	全局	0x0000_00DC
56	可设置	KSCAN	全局	0x0000_00E0
57	可设置	QSPI	全局	0x0000_00E4
58	可设置	CRYPT	全局	0x0000_00E8
59	可设置	ADPCM	全局	0x0000_00EC
60	可设置	-	预留	0x0000_00F0
61	可设置	SM3(HASH)	全局	0x0000_00F4
62	可设置	-	预留	0x0000_00F8
63	可设置	ECC	全局	0x0000_00FC
64	可设置	TRNG	全局	0x0000_0100
65	可设置	-	预留	0x0000_0104
66	可设置	EXTI0	全局	0x0000_0108
67	可设置	EXTI1	全局	0x0000_010C
68	可设置	EXTI2	全局	0x0000_0110
69	可设置	EXTI3	全局	0x0000_0114
70	可设置	EXTI4	全局	0x0000_0118
71	可设置	EXTI5	全局	0x0000_011C
72	可设置	EXTI6	全局	0x0000_0120
73	可设置	EXTI7	全局	0x0000_0124
74	可设置	EXTI8	全局	0x0000_0128
75	可设置	EXTI9	全局	0x0000_012C
76	可设置	EXTI10	全局	0x0000_0130
77	可设置	EXTI11	全局	0x0000_0134
78	可设置	EXTI12	全局	0x0000_0138
79	可设置	EXTI13	全局	0x0000_013C
80	可设置	EXTI14	全局	0x0000_0140
81	可设置	EXTI15	全局	0x0000_0144
82	可设置	DMA	DMA 全局	0x0000_0148
83	可设置	-	预留	0x0000_014C
84	可设置	-	预留	0x0000_0150
85	可设置	-	预留	0x0000_0154
86	可设置	-	预留	0x0000_0158
87	可设置	-	预留	0x0000_015C
88	可设置	-	预留	0x0000_0160

编号	优先级	名称	说明	地址
89	可设置	-	预留	0x0000_0164
90	可设置	-	预留	0x0000_0168
91	可设置	-	预留	0x0000_016C
92	可设置	-	预留	0x0000_0170
93	可设置	-	预留	0x0000_0174
94	可设置	-	预留	0x0000_0178
95	可设置	-	预留	0x0000_017C
96	可设置	BLE_ERROR	全局	0x0000_0180
97	可设置	BLE_HSL0T	全局	0x0000_0184
98	可设置	BLE_RX	全局	0x0000_0188
99	可设置	BLE_TX	全局	0x0000_018C
100	可设置	BLE_START	全局	0x0000_01A0
101	可设置	BLE_END	全局	0x0000_01A4
102	可设置	BLE_SKIP	全局	0x0000_01A8
103	可设置	BLE_SLP	全局	0x0000_01AC
104	可设置	BLE_CRYPT	全局	0x0000_01B0
105	可设置	BLE_GROSSTGTIM	全局	0x0000_01B4
106	可设置	BLE_FINETGTIM	全局	0x0000_01B8
107	可设置	BLE_TIMESTAMP_TGT	全局	0x0000_01BC
108	可设置	BLE_SW	全局	0x0000_01C0
109	可设置	BLE_AUDIO0	全局	0x0000_01C4
110	可设置	BLE_AUDIO1	全局	0x0000_01C8
111	可设置	BLE_AUDIO2	全局	0x0000_01CC

表 3-3 中断向量分配

### 3.4 异步唤醒中断和事件

关于异步唤醒中断的相关参考信息链接如下表：

主题	相关模块	参考链接
NVIC 说明	嵌套向量中断控制器	<a href="#">ARM Cortex-M3 技术参考资料链接</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
异步中断唤醒源		

表 3-4 异步唤醒中断的参考信息链接

#### 3.4.1 异步中断唤醒源

在芯片 STOP1 低功耗模式下的中断唤醒源如下表：

唤醒源	描述
外部端口中断	外部端口输入上升沿或下降沿中断
LVD 中断	LVD 有效边沿或电平产生的中断可唤醒芯片

唤醒源	描述
独立看门狗中断	使用低速时钟 (ULRC/LOSC/LRC) 计数时, 中断可唤醒芯片
窗口看门狗中断	使用低速时钟 (ULRC/LOSC/LRC) 计数时, 中断可唤醒芯片
低功耗定时器	使用低速时钟 (ULRC/LOSC/LRC) 计数时, 中断可唤醒芯片
RTC 中断	RTC 各中断源可唤醒芯片
复位	系统复位(不包含软件复位)

表 3-5 STOP1 低功耗模式的中断唤醒源

在芯片 STOP2 低功耗模式下的中断唤醒源如下表:

唤醒源	描述
外部端口中断	外部端口输入上升沿或下降沿中断 (PA9~PA2)
LVD 中断	LVD 有效边沿或电平产生的中断可唤醒芯片
独立看门狗中断	使用低速时钟 (ULRC/LOSC/LRC) 计数时, 中断可唤醒芯片
-	-
RTC 中断	RTC 各中断源可唤醒芯片
复位	系统复位(不包含软件复位)

表 3-6 STOP2 低功耗模式的中断唤醒源

在芯片 STANDBY 模式下的中断唤醒源如下表:

唤醒源	描述
RTC 中断	RTC 各中断源可唤醒芯片
WAKEUP 端口中断	WAKEUP 端口 (PA2~PA9) 中断, 可唤醒芯片。
MRSTN 复位	端口复位
POR	上电复位

表 3-7 STANDBY 低功耗模式的中断唤醒源

### 3.4.2 事件唤醒

ES32W3120 系列微控制器支持事件唤醒机制: 通过配置外设的中断控制寄存器使能一个中断, 但在 NVIC 中不使能该中断 (可通过设置 PRIMASK 和 BASEPRI 来禁止), 并将 Cortex-M3 内核的系统控制寄存器中的 SEVONPEND 位使能以允许中断事件唤醒 WFE。当外设中断产生后, 芯片从 WFE 唤醒。芯片唤醒后, 软件需要清除相应外设的中断标志位和外设在 NVIC 中断通道上的挂起位。芯片 STOP 模式下的事件唤醒源如下表所示:

事件唤醒源	描述
EXTI_0	选择 GPIOA0,GPIOB0 之一作为唤醒源
EXTI_1	选择 GPIOA1,GPIOB1 之一作为唤醒源
EXTI_2	选择 GPIOA2,GPIOB2 之一作为唤醒源
EXTI_3	选择 GPIOA3,GPIOB3 之一作为唤醒源
EXTI_4	选择 GPIOA4,GPIOB4 之一作为唤醒源
EXTI_5	选择 GPIOA5,GPIOB5 之一作为唤醒源
EXTI_6	选择 GPIOA6,GPIOB6 之一作为唤醒源
EXTI_7	选择 GPIOA7,GPIOB7 之一作为唤醒源
EXTI_8	选择 GPIOA8,GPIOB8 之一作为唤醒源

事件唤醒源	描述
EXTI_9	选择 GPIOA9,GPIOB9 之一作为唤醒源
EXTI_10	选择 GPIOA10,GPIOB10 之一作为唤醒源
EXTI_11	选择 GPIOA11,GPIOB11 之一作为唤醒源
EXTI_12	选择 GPIOA12,GPIOB12 之一作为唤醒源
EXTI_13	选择 GPIOA13,GPIOB13 之一作为唤醒源
EXTI_14	选择 GPIOA14,GPIOB14 之一作为唤醒源
EXTI_15	选择 GPIOA15,GPIOB15 之一作为唤醒源
IWDG	独立看门狗中断事件
WWDG	窗口看门狗中断事件
LVD	LVD 中断事件
RTC	RTC 中断事件
RTC Alarm	RTC 闹钟事件
LP16T	低功耗定时器中断事件

表 3-8 事件唤醒源

### 3.5 存储器及存储器接口

#### 3.5.1 系统总线和存储器

关于系统总线和存储器的相关参考信息链接如下表：

主题	相关模块	参考链接
系统总线和存储器		<a href="#">系统总线和存储器</a>
存储器系统控制	存储器系统控制	<a href="#">存储器系统控制</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
系统总线矩阵		<a href="#">系统总线矩阵</a>
ARM Cortex-M3 内核说明	ARM Cortex-M3 core	<a href="#">ARM Cortex-M3 技术参考资料链接</a>
DMA		<a href="#">DMA 控制器</a>
扩展外部存储		<a href="#">QSPI 控制器</a>

表 3-9 总线架构和存储器的参考信息链接

#### 3.5.2 存储器系统控制配置

关于存储器系统控制的相关参考信息链接如下表：

主题	相关模块	参考链接
存储器系统控制	存储器系统控制	<a href="#">存储器系统控制</a>
总线架构和存储器		<a href="#">总线架构和存储器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
系统总线矩阵		<a href="#">系统总线矩阵</a>

表 3-10 存储器系统控制的参考信息链接

### 3.5.3 扩展外部存储器配置

关于扩展外部存储器的相关参考信息链接如下：

主题	相关模块	参考链接
扩展外部存储控制器		<a href="#">QSPI 控制器</a>
存储器系统控制	存储器系统控制	<a href="#">存储器系统控制</a>
总线架构和存储器		<a href="#">总线架构和存储器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理</a>
系统总线矩阵		<a href="#">系统总线矩阵</a>
管脚分配		PINMUX

表 3-11 扩展外部存储器的参考信息链接

## 3.6 系统模块配置

### 3.6.1 系统配置控制器配置

关于系统配置控制器的相关参考信息链接如下：

主题	相关模块	参考链接
系统配置控制器		<a href="#">系统配置控制器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
电源管理		<a href="#">电源管理及低功耗模式</a>

表 3-12 系统配置控制器的参考信息链接

### 3.6.2 电源管理及低功耗模式配置

关于电源管理及低功耗模式的相关参考信息链接如下：

主题	相关模块	参考链接
电源管理及低功耗模式		<a href="#">电源管理及低功耗模式</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
复位管理		<a href="#">复位管理</a>
时钟管理		<a href="#">时钟管理</a>

表 3-13 电源管理及低功耗模式的参考信息链接

### 3.6.3 复位管理配置

关于复位管理的相关参考信息链接如下：

主题	相关模块	参考链接
复位管理		<a href="#">复位管理</a>
电源管理及低功耗模式		<a href="#">电源管理及低功耗模式</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>

主题	相关模块	参考链接
时钟管理		<a href="#">时钟管理</a>

表 3-14 复位管理的参考信息链接

### 3.6.4 DMA控制器配置

关于 DMA 控制器的相关参考信息链接如下：

主题	相关模块	参考链接
DMA 控制器		<a href="#">DMA 控制器</a>
电源管理及低功耗模式		<a href="#">电源管理及低功耗模式</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>
系统总线矩阵		<a href="#">系统总线矩阵</a>
DMA 请求		DMA 请求列表

表 3-15 电源管理及低功耗模式的参考信息链接

DMA 控制器包含 12 个通道，每个 DMA 通道对应一个 DMA 多路复用器，每个多路复用器包含了微控制器所有的 DMA 申请源，由 DMA\_CHx\_SELCON(x=0,1...11)配置选择。多路复用器和 DMA 之间连接图如下：

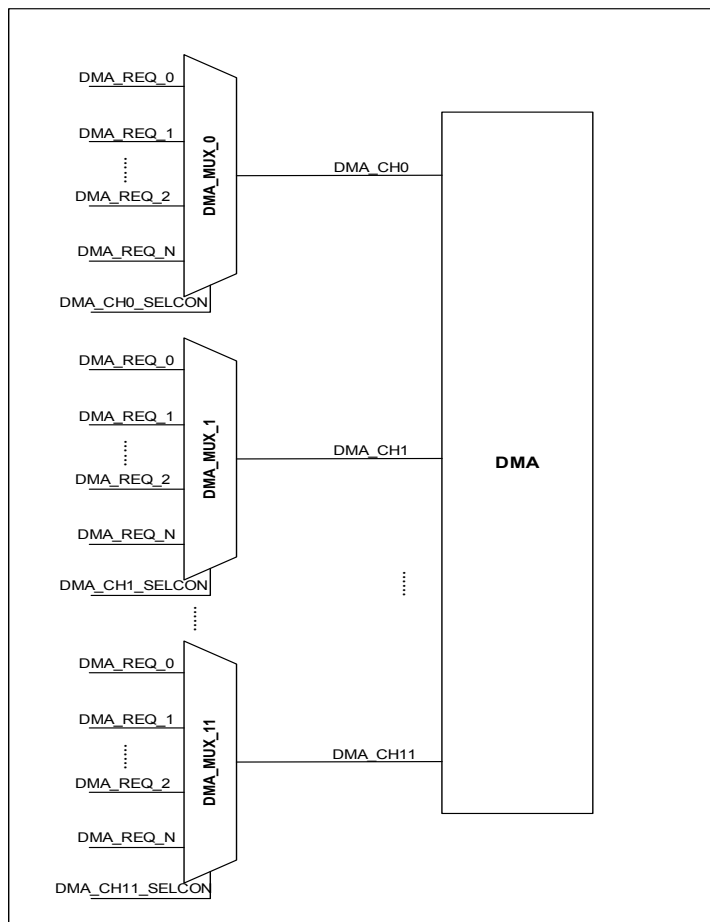


图 3-2 DMA 多路复用器与 DMA 连接图



每个 DMA 多路复用器可选择的 DMA 请求如下表所示：

编号	模块名	DMA 请求源
0	ADC0	ADC0 转换结束
1	TIM0 (GP16C4T0)	TIM0_CH1
2		TIM0_CH2
3		TIM0_CH3
4		TIM0_CH4
5		TIM0_TRIG
6		-
7		TIM0_UP
8	TIM1 (GP16C4T1)	TIM1_CH1
9		TIM1_CH2
10		TIM1_CH3
11		TIM1_CH4
12		TIM1_TRIG
13		-
14		TIM1_UP
15	TIM2 (AD16C4T0)	TIM2_CH1
16		TIM2_CH2
17		TIM2_CH3
18		TIM2_CH4
19		TIM2_TRIG
20		TIM2_COM
21		TIM2_UP
22	TIM3 (AD16C4T1)	TIM3_CH1
23		TIM3_CH2
24		TIM3_CH3
25		TIM3_CH4
26		TIM3_TRIG
27		TIM3_COM
28		TIM3_UP
29	UART0	UART0_RX
30		UART0_TX
31	UART1	UART1_RX
32		UART1_TX
33	UART2	UART2_RX
34		UART2_TX
35	SPI0	SPI0_TX
36		SPI0_RX
37	SPI1	SPI1_TX
38		SPI1_RX
39	I2C0	I2C0_TX

编号	模块名	DMA 请求源
40		I2C0_RX
41	I2C1	I2C1_TX
42		I2C1_RX
43	CRC	
44	CRYPT	CRYPT_WR
45		CRYPT_RD
46	ECC	
47		
48	GPIO	GPIO_EXTI0
49		GPIO_EXTI1
50		GPIO_EXTI2
51		GPIO_EXTI3
52		GPIO_EXTI4
53		GPIO_EXTI5
54		GPIO_EXTI6
55		GPIO_EXTI7
56		GPIO_EXTI8
57		GPIO_EXTI9
58		GPIO_EXTI10
59		GPIO_EXTI11
60		GPIO_EXTI12
61		GPIO_EXTI13
62		GPIO_EXTI14
63		GPIO_EXTI15
64	MSC (预留)	
65	PIS	PIS 通道 0~15
66	QSPI	QSPI_INDRD
67		QSPI_INDWR
68	ADPCM	
69	PDMPDM	
70	HASH	
71	TIM4 (BS16T0)	
72	TIM5 (BS16T1)	
73		
74		

表 3-16 DMA 请求列表

### 3.6.5 外设互联(PIS)配置

关于外设互联(PIS)的相关参考信息链接如下：

主题	相关模块	参考链接
外设互联(PIS)		<a href="#">外设互联(PIS)</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟分配</a>

表 3-17 外设互联(PIS)的参考信息链接

### 3.6.6 独立看门狗定时器配置

关于独立看门狗定时器的相关参考信息链接如下：

主题	相关模块	参考链接
独立看门狗定时器		<a href="#">独立看门狗定时器</a>
系统配置控制器		<a href="#">系统配置控制器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>
电源管理		<a href="#">电源管理及低功耗模式</a>

表 3-18 独立看门狗定时器的参考信息链接

#### 3.6.6.1 独立看门狗定时器的时钟

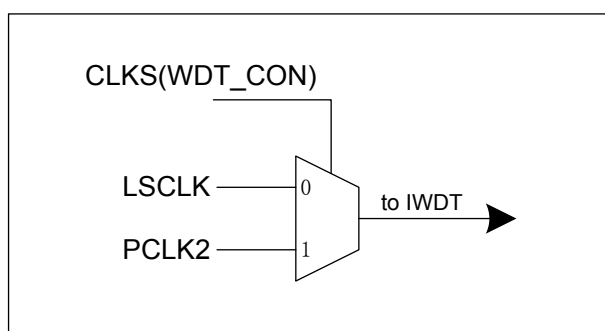


图 3-3 独立看门狗计数时钟

#### 3.6.6.2 独立看门狗定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	可工作
STANDBY	掉电

### 3.6.7 窗口看门狗定时器配置

关于窗口看门狗定时器的相关参考信息链接如下：

主题	相关模块	参考链接
窗口看门狗定时器		<a href="#">窗口看门狗定时器</a>
系统配置控制器		<a href="#">系统配置控制器</a>

主题	相关模块	参考链接
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>
电源管理		<a href="#">电源管理及低功耗模式</a>

表 3-19 窗口看门狗定时器的参考信息链接

### 3.6.7.1 窗口看门狗定时器的时钟

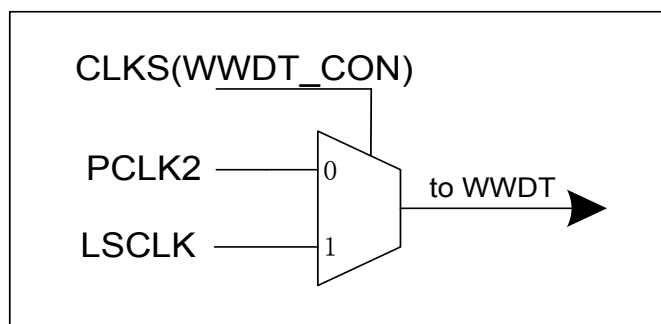


图 3-4 窗口看门狗计数时钟

### 3.6.7.2 窗口看门狗定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	掉电
STANDBY	掉电

### 3.6.8 时钟管理配置

关于时钟管理的相关参考信息链接如下：

主题	相关模块	参考链接
时钟管理		<a href="#">时钟管理</a>
系统配置控制器		<a href="#">系统配置控制器</a>
存储器映射图		<a href="#">存储器映射图</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
GPIO 及端口控制		<a href="#">GPIO 及端口控制</a>

表 3-20 时钟管理的参考信息链接

#### 3.6.8.1 LOSC的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

### 3.6.8.2 HOSC的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	掉电
STOP2	掉电
STANDBY	掉电

### 3.6.8.3 HRC的低功耗动作模式

芯片低功耗模式	模块动作模式
RUN,SLEEP	可通过配置 CMU 配置寄存器 (CMU_CFGR) 的 HRCFSW 位选择输出 24MHz 或 2MHz 时钟。
STOP1	关闭
STOP2	关闭
STANDBY	掉电

### 3.6.8.4 LRC的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	可工作
STANDBY	掉电

### 3.6.8.5 ULRC的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	开启
STOP2	开启
STANDBY	可工作

## 3.7 外部接口配置

### 3.7.1 通用IO及端口控制配置

关于通用 IO 及端口控制配置的相关参考信息链接如下：

主题	相关模块	参考链接
通用 IO 及端口控制		<a href="#">通用 IO 及端口控制</a>
系统配置控制器		<a href="#">系统配置控制器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>
电源管理		<a href="#">电源管理及低功耗模式</a>
外设互联		<a href="#">外设互联(PIS)</a>

表 3-21 通用 IO 及端口控制的参考信息链接

## 3.8 安全管理及运算加速配置

### 3.8.1 循环冗余校验(CRC)配置

关于循环冗余校验(CRC)配置的相关参考信息链接如下：

主题	相关模块	参考链接
循环冗余校验(CRC)		<a href="#">循环冗余校验(CRC)</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>

表 3-22 循环冗余校验(CRC)的参考信息链接

### 3.8.2 加密处理配置

关于加密处理配置的相关参考信息链接如下：

主题	相关模块	参考链接
加密处理		<a href="#">加密处理</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>

表 3-23 加密处理的参考信息链接

### 3.8.3 真随机数发生器配置

关于真随机数发生器配置的相关参考信息链接如下：

主题	相关模块	参考链接
真随机数发生器		<a href="#">真随机数发生器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>

表 3-24 真随机数发生器的参考信息链接

### 3.8.4 运算加速器配置

关于运算加速器配置的相关参考信息链接如下：

主题	相关模块	参考链接
运算加速器		<a href="#">运算加速器</a>
存储器映射图		<a href="#">存储器映射图</a>
时钟		<a href="#">时钟管理</a>

表 3-25 真随机数发生器的参考信息链接

## 3.9 模拟配置

### 3.9.1 ADC控制配置

关于 ADC 控制配置的相关参考信息链接如下：

主题	相关模块	参考链接
ADC		<a href="#">模数转换(ADC)</a>

主题	相关模块	参考链接
存储器映射图		存储器映射图
时钟		时钟分配
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-26 ADC 控制的参考信息链接

### 3.9.1.1 ADC转换通道配置

ADC 支持 8 个通道选择，仅支持单端模式，具体分配如下：

寄存器 ADC_RSQR*/ADC_JSQR	ADC 通道	信号分配
00000	ADC 通道 0	ADC_IN0
00001	ADC 通道 1	ADC_IN1
00010	ADC 通道 2	ADC_IN2
00011	ADC 通道 3	ADC_IN3
00100	ADC 通道 4	ADC_IN4
00101	ADC 通道 5	ADC_IN5
00110	ADC 通道 6	ADC_IN6
00111	ADC 通道 7	BT50 模拟前端 VTUNE 电压
01000	ADC 通道 8	温度传感器
01001	ADC 通道 9	电压检测器
保留		

### 3.9.1.2 ADC电源及参考电压

ADC 的正向参考电压为：内部参考、电源电压参考、外部参考 3 种模式；

ADC 的负向参考电压为：不提供负向参考；

### 3.9.1.3 ADC的时钟

PCLK2

### 3.9.1.4 ADC的低功耗动作模式

芯片低功耗模式	模块动作模式
SLEEP	可工作

## 3.10 定时器配置

### 3.10.1 高级定时器配置

关于高级定时器的相关参考信息链接如下：

主题	相关模块	参考链接
高级定时器		高级定时器

主题	相关模块	参考链接
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-27 高级定时器的参考信息链接

### 3.10.1.1 高级定时器例化说明

TIM2,TIM3 为高级定时器

### 3.10.1.2 高级定时器的时钟

PCLK1

### 3.10.1.3 高级定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

## 3.10.2 通用定时器配置

关于通用定时器的相关参考信息链接如下：

主题	相关模块	参考链接
通用定时器		通用定时器
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-28 通用定时器的参考信息链接

### 3.10.2.1 通用定时器例化说明

TIM0,TIM1 为通用定时器

### 3.10.2.2 通用定时器的时钟

PCLK1



### 3.10.2.3 通用定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

### 3.10.3 基本定时器配置

关于基本定时器的相关参考信息链接如下：

主题	相关模块	参考链接
基本定时器		基本定时器
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-29 基本定时器的参考信息链接

#### 3.10.3.1 基本定时器例化说明

TIM4,TIM5 为基本定时器

#### 3.10.3.2 基本定时器的时钟

PCLK1

#### 3.10.3.3 基本定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

### 3.10.4 低功耗定时器配置

关于低功耗定时器的相关参考信息链接如下：

主题	相关模块	参考链接
低功耗定时器		低功耗定时器
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-30 低功耗定时器的参考信息链接

### 3.10.4.1 低功耗定时器例化说明

LP16T 为低功耗定时器

### 3.10.4.2 低功耗定时器的时钟

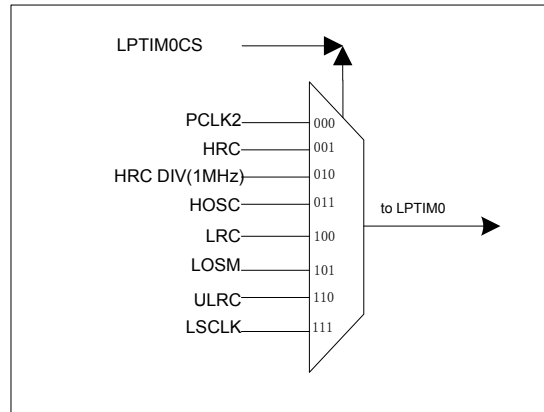


图 3-5 低功耗定时器计数时钟

### 3.10.4.3 低功耗定时器的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	不工作
STANDBY	无

### 3.10.5 RTC定时器配置

关于 RTC 定时器的相关参考信息链接如下：

主题	相关模块	参考链接
RTC		RTC
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-31 RTC 的参考信息链接

### 3.10.5.1 RTC的时钟

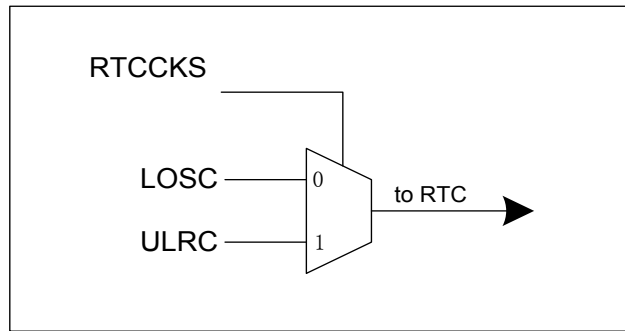


图 3-6 RTC 计数时钟

### 3.10.5.2 RTC的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	可工作
STOP2	可工作
STANDBY	可工作

## 3.11 通信配置

### 3.11.1 I2C接口配置

关于 I2C 接口的相关参考信息链接如下：

主题	相关模块	参考链接
I2C 接口		I2C 接口
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-32 I2C 接口的参考信息链接

#### 3.11.1.1 I2C接口的时钟

PCLK1

#### 3.11.1.2 I2C接口的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

### 3.11.2 串行外设接口(SPI)配置

关于串行外设接口(SPI)的相关参考信息链接如下:

主题	相关模块	参考链接
串行外设接口(SPI)		串行外设接口(SPI)
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-33 串行外设接口(SPI)的参考信息链接

#### 3.11.2.1 串行外设接口(SPI)的时钟

PCLK1

#### 3.11.2.2 串行外设接口(SPI)的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

### 3.11.3 通用异步收发器(UART)

关于通用异步收发器(UART)的相关参考信息链接如下:

主题	相关模块	参考链接
通用异步收发器(UART)		通用异步收发器(UART)
系统配置控制器		系统配置控制器
存储器映射图		存储器映射图
时钟		时钟管理
电源管理		电源管理
管脚分配		PINMUX
外设互联		外设互联(PIS)

表 3-34 通用异步收发器(UART)的参考信息链接

#### 3.11.3.1 通用异步收发器(UART)的时钟

PCLK1(目前 SOC 层只提供了 PCLK 时钟, 后续建议时钟源及预分频灵活可配)

#### 3.11.3.2 通用异步收发器(UART)的低功耗动作模式

芯片低功耗模式	模块动作模式
STOP1	不工作
STOP2	不工作
STANDBY	掉电

## 第4章 系统总线和存储器

### 4.1 概述

主系统由 32 位多层 AHB 总线矩阵构成，可实现以下部分的互连。

五条主控总线：

- ◇ Cortex-M3 内核 I 总线、D 总线和 S 总线
- ◇ DMA 存储器总线

七条被控总线：

- ◇ 内部 Flash 总线
- ◇ 内部 SRAM1 总线
- ◇ 辅助内部 SRAM2 总线 AHB1 外设
- ◇ APB1 外设
- ◇ APB2 外设
- ◇ FSMC 外部存储器接口

借助总线矩阵，可以实现主控总线到被控总线的访问，这样即使在多个高速外设同时运行期间，系统也可以实现并发访问和高效运行。

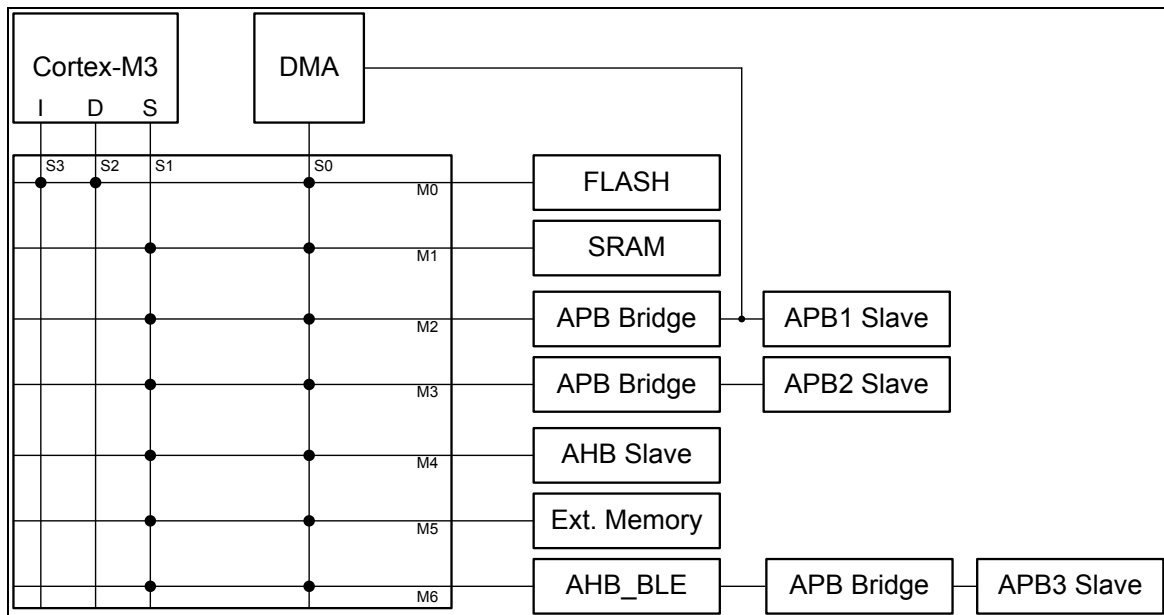


图 4-1 系统总线矩阵

### 4.2 系统总线

#### 4.2.1 S0: I总线

此总线用于将 Cortex-M3 内核的指令总线连接到总线矩阵。内核通过此总线获取指令。此总线访问的对象是包含代码的存储器（内部 Flash/SRAM 或通过 FSMC 的外部存储器）。

#### 4.2.2 S1: D总线

此总线用于将 Cortex-M3 内核的数据总线连接到总线矩阵。内核通过此总线进行立即数加载和调试访问。此总线访问的对象是包含代码或数据的存储器（内部 Flash 或通过 EMI 的外部存储器）。

#### 4.2.3 S2: S总线

此总线用于将 Cortex-M3 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据。也可通过此总线获取指令（效率低于 ICode）。此总线访问的对象是内部 SRAM、AHB1 外设、APB1 外设、APB2 外设以及通过 EMI 的外部存储器。

#### 4.2.4 S3: DMA总线

此总线用于将 DMA 存储器总线主接口连接到总线矩阵。DMA 通过此总线访问外设或执行存储器间的数据传输。此总线访问的对象是 AHB 和 APB 外设以及数据存储器：内部 SRAM 以及通过 FSMC 的外部存储器。

#### 4.2.5 总线矩阵

总线矩阵用于主控总线之间的访问仲裁管理。仲裁采用循环调度算法。

#### 4.2.6 AHB/APB总线桥

借助两个 AHB/APB 总线桥 APB1 和 APB2，可在 AHB 总线与两个 APB 总线之间实现完全同步的连接，从而可灵活选择外设频率。

### 4.3 存储器的组织结构

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个顺序的 4 GB 地址空间内。

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

有关外设寄存器映射的详细信息，请参见相关章节。

可寻址的存储空间分为 8 个主要块，每个块为 512 MB。

未分配给片上存储器和外设的所有存储区域均视为“保留区”。请参见产品数据手册中的存储器映射图。

#### 4.3.1 系统存储器映射

系统存储器映射图如下表所示

地址范围	存储	备注
0x0000_0000~0x0007_FFFF	FLASH MAIN 存储器	内嵌 eFLASH 主程序存储区
0x0008_0000~0x0008_3FFF	FLASH INFO 存储器	内嵌 eFLASH 信息区 0/1/2/3
0x0008_0000~0x0FFF_FFFF	Reserved	
0x1000_0000~0x1000_0FFF	IAP ROM Table	内置的 eFlash 编程/擦除函数

地址范围	存储	备注
0x1000_1000~0x13FF_FFFF	Reserved	
0x1400_0000~0x1BFF_FFFF	QSPI FLASH 映射区	外部扩展串行存储器
0x2000_0000~0x2000_7FFF	SRAM0	
0x2000_8000~0x2000_FFFF	SRAM1	
0x2001_0000~0x3FFF_FFFF	Reserved	
0x4000_0000~0x4003_FFFF	APB 外设 1	
0x4004_0000~0x4007_FFFF	APB 外设 2	
0x4008_0000~0x400B_FFFF	AHB 外设 1	
0x400C_0000~0x400F_FFFF	AHB 外设 2	AHB 外设 2 主要分配给 BLE5.0
0x4010_0000~0x5FFF_FFFF	Reserved	
0x6000_0000~0xA000_03FF	FSMC (保留区间)	外部扩展存储
0xA000_0400~0xDFFF_FFFF	Reserved	
0xE000_0000~0xE00F_FFFF	私有外设	
0xE010_0000~0xFFFF_FFFF	Reserved	

表 4-1 系统存储器映射

### 4.3.2 FLASH存储器映射

Flash 接口可管理 CPU 通过 AHB I-Code 和 D-Code 对 Flash 进行的访问。该接口可针对 Flash 执行擦除和编程操作，并实施读写保护机制。Flash 接口通过指令预取和缓存机制加速代码执行。

Flash 结构如下：

- ◇ 主存储区共 512K Bytes，分为 128 页，每页 4K Bytes，支持预取模式，一次可读 8 Bytes
- ◇ 信息区共 16K Bytes，分为 4 页，每页 4K Bytes，支持预取模式，一次可读 8 Bytes
- ◇ 芯片配置字，存放在信息区第 1 页中，用于配置读写保护、BOR 级别、软件/硬件看门狗以及配置器件处于待机或停止模式下的复位模式等。

Flash 映射关系如图所示。

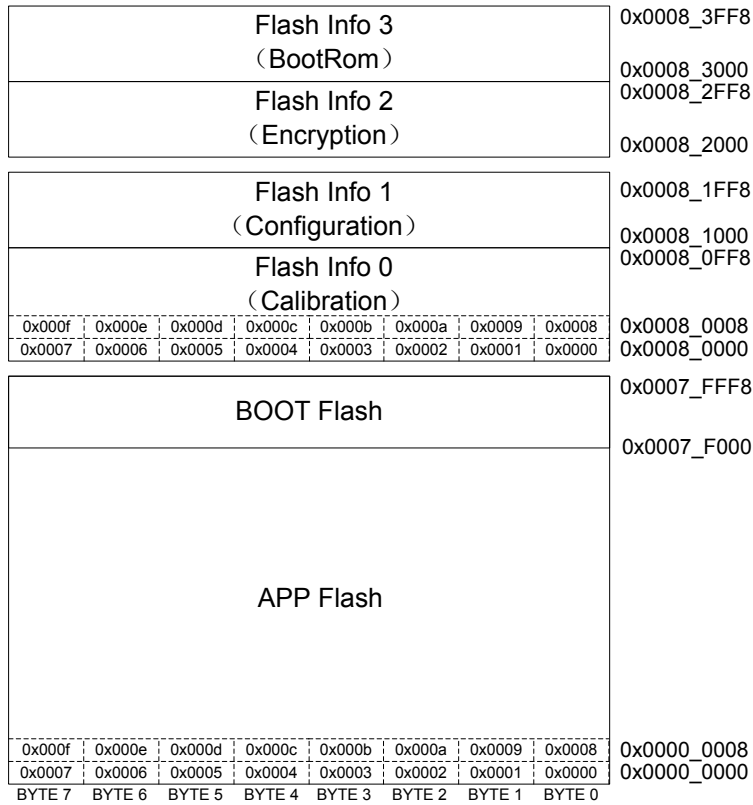


图 4-2 Flash 映射示意图

Flash 接口可支持 AHB 读取、IAP 编程、ISP 编程、配置字读取，其组织架构模式如图所示。其控制优先级从高到低依次为配置字、ISP、IAP、AHB。

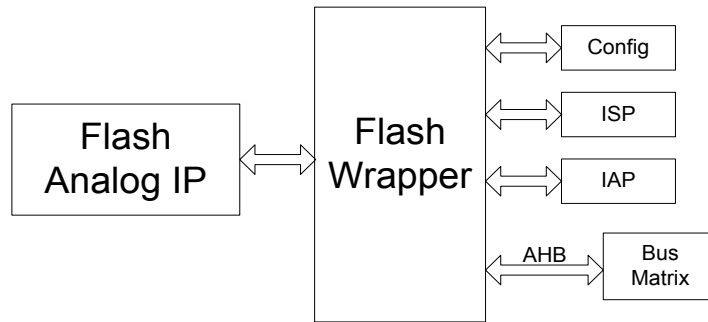


图 4-3 Flash 组织架构示意图

### 4.3.3 SRAM 存储器映射

SRAM 容量为 64K Bytes，由 SRAM0/SRAM1 组成，地址为 0x2000\_0000~0x2000\_FFFF，单周期访问时间。

### 4.3.4 外设存储器映射

ES32W3120 系列产品外设存储器映射如下表所示：

总线	边界地址	外设
APB1	0x4000 0000 - 0x4000 03FF	TIM0 (GP16C4T0)



总线	边界地址	外设
	0x4000 0400 - 0x4000 07FF	TIM1 (GP16C4T1)
	0x4000 0800 - 0x4000 0BFF	TIM2 (AD16C4T0)
	0x4000 0C00 - 0x4000 0FFF	TIM3 (AD16C4T1)
	0x4000 1000 - 0x4000 13FF	TIM4 (BS16T0)
	0x4000 1400 - 0x4000 17FF	TIM5 (BS16T1)
	0x4000 1800 - 0x4000 3FFF	—
	0x4000 4000 - 0x4000 43FF	UART0
	0x4000 4400 - 0x4000 47FF	UART1
	0x4000 4800 - 0x4000 4BFF	UART2
	0x4000 4C00 - 0x4000 5FFF	—
	0x4000 6000 - 0x4000 63FF	SPI0
	0x4000 6400 - 0x4000 67FF	SPI1
	0x4000 6800 - 0x4000 7FFF	—
	0x4000 8000 - 0x4000 83FF	I2C0
	0x4000 8400 - 0x4000 87FF	I2C1
	0x4000 8800 - 0x4000 8BFF	ADPCM
	0x4000 8C00 - 0x4000 8FFF	TK
	0x4000 9000 - 0x4000 97FF	—
	0x4000 9800 - 0x4000 9BFF	ECC
	0x4000 9C00 - 0x4000 BFFF	—
	0x4000 C000 - 0x4000 D3FF	DMA
	0x4000 D3FF - 0x4000 DFFF	—
	0x4000_E000 - 0x4000_E3FF	QSPI Controller Register
	0x4000 E400 - 0x4000 EFFF	—
	0x4000 F000 - 0x4000 F3FF	Key_scanner
	0x4000 F400 - 0x4000 F7FF	QDEC
0x4000 F800 - 0x4003 FFFF	—	
APB2	0x4004 0000 - 0x4004 03FF	LP16T
	0x4004 0400 - 0x4004 1FFF	—
	0x4004 2000 - 0x4004 23FF	ADC0
	0x4004 2400 - 0x4004 2FFF	—
	0x4004 3000 - 0x4004 33FF	—
	0x4004 3400 - 0x4004 3FFF	—
	0x4004 4000 - 0x4004 43FF	—
	0x4004 4400 - 0x4004 5FFF	—
	0x4004 6000 - 0x4004 63FF	WWDT
	0x4004 6400 - 0x4004 6FFF	—
	0x4004 7000 - 0x4004 73FF	—
	0x4004 7400 - 0x4004 7FFF	—
0x4004 8000 - 0x4004 83FF	IWDT	

总线	边界地址	外设
	0x4004 8400 - 0x4004 87FF	RTC
	0x4004 8800 - 0x4004 8BFF	—
	0x4004 8C00 - 0x4004 8FFF	BKP1
	0x4004 9000 - 0x4004 93FF	BKPC
	0x4004 9400 - 0x4004 97FF	BKPRAM
	0x4004 9800 - 0x4004 9BFF	BKP2
	0x4004 9C00 - 0x4004 9FFF	BKP3
	0x4004 A000 - 0x4004 A3FF	DBGC
	0x4004 A400 - 0x4007 FFFF	—
AHB1	0x4008 0000 - 0x4008 03FF	SYSCFG
	0x4008 0400 - 0x4008 07FF	CMU
	0x4008 0800 - 0x4008 0BFF	RMU
	0x4008 0C00 - 0x4008 0FFF	PMU
	0x4008 1000 - 0x4008 13FF	MSC
	0x4008 1400 - 0x4008 3BFF	—
	0x4008 3C00 - 0x4008 3FFF	—
	0x4008 4000 - 0x4008 4FFF	GPIO
	0x4008 4300 - 0x4008 4FFF	GPIOEXTI
	0x4008 5000 - 0x4008 53FF	CRC
	0x4008 5400 - 0x4008 57FF	CALC
	0x4008 5800 - 0x4008 5BFF	CRYPT
	0x4008 5C00 - 0x4008 5FFF	TRNG
	0x4008 6000 - 0x4008 63FF	PIS
	0x4008 6400 - 0x4008 67FF	—
	0x4008 6800 - 0x4008 6BFF	HASH
0x4008 6800 - 0x400B FFFF	—	
AHB2 (BLE5.0)	0x400C 0000 - 0x400C 03FF	BT50_Controller
	0x400C 4000 - 0x400C 7FFF	—
	0x400C 8000 - 0x400C DFFF	BT50_EM (Ex Memory)
	0x400C E000 - 0x400C FFFF	—
	0x400D 0000 - 0x400D FFFF	—
	0x400E 0000 - 0x400E 03FF	BT50_MDM
	0x400E 0400 - 0x400E 07FF	BT50_RFDIG
	0x400E 0800 - 0x400E 0BFF	—
	0x400E 0C00 - 0x400E FFFF	—
	0x400F 0000 - 0x400F FFFF	—

表 4-2 外设存储映射

### 4.3.5 私有外设存储器映射

地址范围	私有外设	备注
0xE000_0000~0xE000_0FFF	ITM	
0xE000_1000~0xE000_1FFF	DWT	
0xE000_2000~0xE000_2FFF	FPB	
0xE000_3000~0xE000_DFFF	Reserved	
0xE000_E000~0xE000_EFFF	内核系统控制	NVIC 寄存器空间
0xE000_F000~0xE003_FFFF	Reserved	
0xE004_0000~0xE004_0FFF	TPIU	
0xE004_1000~0xE004_1FFF	Reserved	
0xE004_2000~0xE00F_EFFF	外部扩展私有外设	
0xE00F_F000~0xE00F_FFFF	Reserved	

表 4-3 私有外设存储器映射

### 4.3.6 位带Bitband

位带采用 Cortex-M3 内核自带的逻辑,分为读-修改-写回三个阶段,内核在读阶段会将 AHB 总线锁定,写回完成后将解除锁定,锁定期间其他主控总线无法访问位带所操作的从动总线。

#### 4.3.6.1 SRAM位带扩展

SRAM 支持位带扩展,可使用普通的加载和存储指令对单比特进行读写操作。当 SRAM 存储空间小于 1MB 时(地址范围: 0x2000\_0000 ~ 0x2010\_0000),通过位带扩展,除可在起始地址为 0x2000\_0000 的空间访问 SRAM 外,还可在起始地址为 0x2200\_0000 的位带扩展区以单比特方式访问 SRAM。

位带扩展区把每个比特扩展为一个 32-bit 的字,即占用 4 个字节地址;一个 byte 占用  $8 \times 4 = 32$  个地址。通过访问这些字可达到访问原始比特的目的。对于 SRAM 的某个 bit,如果它所在字节地址为 A,位序号为 N ( $0 \leq N \leq 7$ ),则该 bit 在 SRAM 位带扩展后的地址为:

$$\text{AliasAddress\_A\_N} = 0x2200\_0000 + (A - 0x2000\_0000) \times 32 + N \times 4$$

例如,字节地址 A 为 0x2000\_0001,访问该地址的 bit1,地址为:

$$\text{AliasAddress\_A\_N} = 0x2200\_0000 + 1 \times 32 + 1 \times 4 = 0x2200\_0024$$

#### 4.3.6.2 外设位带扩展

外设支持位带扩展,可使用普通的加载和存储指令对单比特进行读写操作。通过位带扩展,除可在 0x4000\_0000 起始的空间访问外设外,还可在起始地址为 0x4200\_0000 的位带扩展区以单比特方式访问外设。

位带扩展区把每个比特扩展为一个 32-bit 的字,通过访问这些字可达到访问原始比特的目的。对于外设寄存器的某一位,如果它所在字节地址为 A,位序号为 N ( $0 \leq N \leq 31$ ),则该位在外设位带扩展后的地址为:

$$\text{AliasAddress\_A\_N} = 0x4200\_0000 + (A - 0x4000\_0000) \times 32 + N \times 4$$

利用外设位带访问对寄存器位置 1 和清 0 的例子。

```

LDR    R0, = AliasAddress_A_N

MOVS  R1, #1

STR    R1, [R0]           ; 对该位置 1

LDR    R0, = AliasAddress_A_N

MOVS  R1, #0

STR    R1, [R0]           ; 对该位清 0
    
```

### 4.4 启动引导

芯片上电或发生 MRSTn 复位时，程序固定从 Boot Rom 中启动，Boot Rom 为固定烧录的一段启动程序代码，地址为 0x0008\_3000(存储在 FLASH INFO3 中)，共 4K 字节。

芯片发生软件复位后，由配置字 CFG\_WORD0 的 BOOT 位决定程序是从 Boot Flash 还是 App Flash 中启动。BOOT 为 0 时固定从 App Flash 启动；BOOT 为 1 时从 App Flash 或 Boot Flash 位置启动，启动位置由 SYSCFG\_MEMRMP 中的 BTF 控制位决定。Boot Flash 为用户烧录的 Boot 程序，起始地址为 0x0007\_F000，共 4K 字节。App Flash 为用户系统运行程序，起始地址为 0x0000\_0000。

启动引导如下图所示：

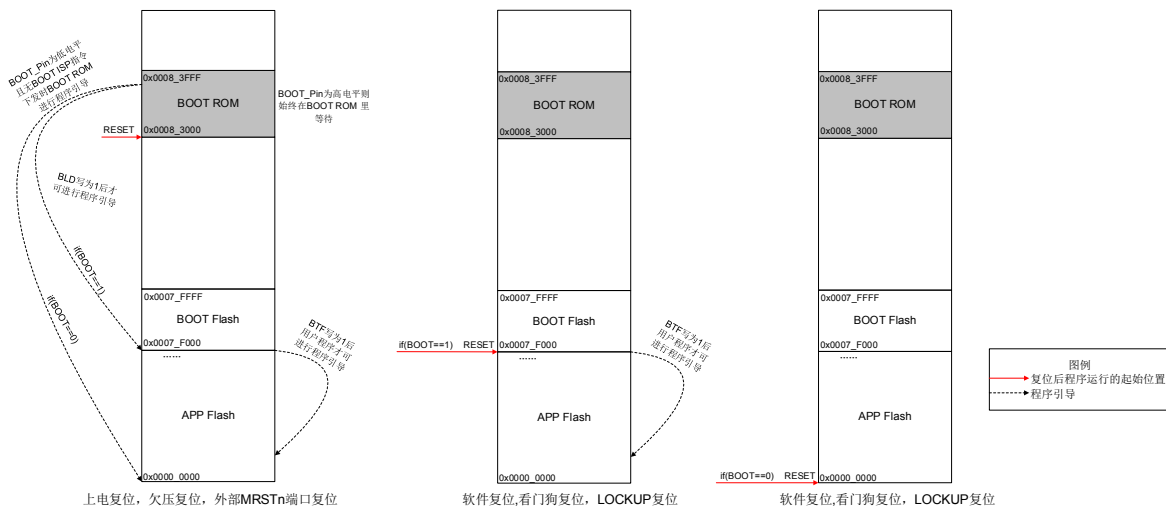


图 4-4 启动引导

复位源								控制位			复位启动位置		
POR	BOR	MRSTn	WDT	CHIPRST	LOCKUP	MCURST	CPURST	BOOT	BTF	BLD	BootRom	BootFlash	AppFlash
√								x	0	0	√		
	√							x	0	0	√		
		√						x	0	0	√		
			√					0	0	1			√
				√				0	0	1			√
					√			0	0	1			√
						√		0	0	1			√
							√	0	0	1			√
			√					1	0	1		√	
				√				1	0	1		√	
					√			1	0	1		√	
						√		1	0	1		√	
							√	1	0	1		√	

表 4-4 启动引导控制说明

注 1: BOOT 控制位对应于 CFG\_WORD0[12]; BTF 控制位对应于控制寄存器 SYSCFG\_MEMRMP[8]; BLD 控制位对应于控制寄存器 SYSCFG\_MEMRMP[0]; (BLD 的复位受 POR、BOR、MRSTn 复位源的影响)

注 2: 复位状态下, BTF 的默认值为 0, 只有当芯片正常运行后才可以修改 BTF 的值; (BTF 的复位受上述所有复位源的影响)

## 第5章 存储器系统控制

### 5.1 概述

存储器系统控制器（MSC）主要作用是控制存储器的运行状态，包括对 Flash 的擦除、编程和加密控制。

### 5.2 特性

- ◆ 支持对 Flash 存储器的写保护：防止意外干扰导致的非预期擦写，保证关键代码的高可靠性；
- ◆ 支持对 Flash 存储器的全加密和分区加密保护，防止用户代码被窃取和破坏；
- ◆ 支持对 Flash 的编程和擦除（全擦和页擦）控制；

### 5.3 结构图

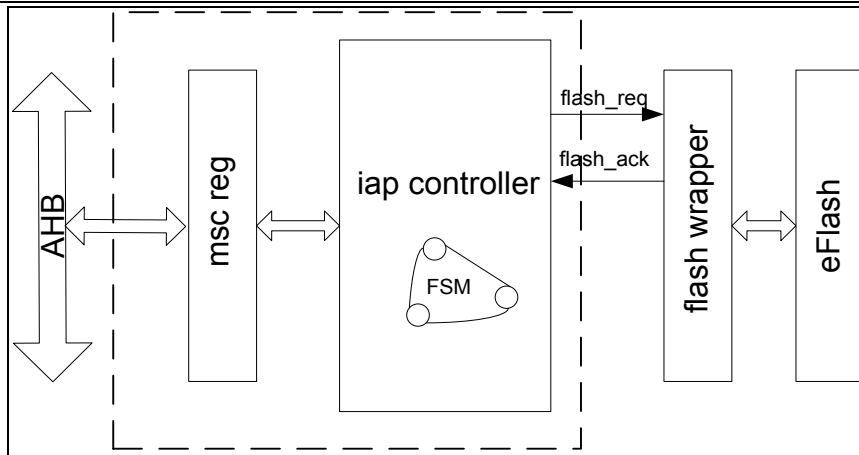


图 5-1 结构图

### 5.4 功能描述

#### 5.4.1 Flash写保护

Flash 存储器可以设置两段写保护区域，分别存储在芯片配置信息区 CFG\_WRPT0、CFG\_WRPT1 中。其中 WRP0\_ST、WRP0\_ED 和 WRP1\_ST、WRP1\_ED 配置两段写保护区域的起始页号和结束页号，WRP0\_EN 和 WRP1\_EN 分别配置两段区域是否使能写保护。当段写保护区域有效时，对写保护区域进行页擦除或编程时会置起访问 Flash 错误标示（标志位：MSC\_IAPSR->WRP\_F）。分区写保护区仅对主程序区有保护作用，对信息区无保护作用。分区保护区有效时，是可以执行芯片主程序区全擦除的动作，执行完芯片主程序区全擦动作之后，分区写保护失效。（如果未更改写保护区配置，下次启动 MSC，写保护区仍然有效）

WRP0_EN	WRP1_EN	起始页和结束页设置	保护区
0	0	XX	所有页处于非写包括状态
1	0	WRP0_ST < WRP0_ED	WRP0_ST 和 WRP0_ED 之间的页处

WRP0_EN	WRP1_EN	起始页和结束页设置	保护区域
			于写保护状态，其它页处于非保护状态
1	0	WRP0_ST>WRP0_ED	WRP0_ST和WRP0_ED之间的页处于非保护状态，其它页处于写保护状态
1	0	WRP0_ST=WRP0_ED	页 WRP0_ST 处于写保护状态，其它页处于非写保护状态
0	1	WRP1_ST<WRP1_ED	WRP1_ST和WRP1_ED之间的页处于写保护状态，其它页处于非保护状态
0	1	WRP1_ST>WRP1_ED	WRP1_ST和WRP1_ED之间的页处于非保护状态，其它页处于写保护状态
0	1	WRP1_ST=WRP1_ED	页 WRP1_ST 处于写保护状态，其它页处于非写保护状态
1	1	情况为上述组合（注 1）	（注 1）

表 5-1 Flash 写保护对应区域表

注 1: 两段分区保护可以同时有效，保护区域为两段保护区域的和。

#### 5.4.2 Flash私有代码读保护

系统中支持 Flash 的一段私有代码读保护区，存储器系统控制器（MSC）对私有代码读保护区的原则同“4.4.1 Flash 写保护”。当私有代码读保护区有效时，对私有代码读保护区进行页擦除或编程时会置起访问 Flash 错误标示（标志位：MSC\_IAPSR->PCRDP\_F）。私有代码读保护仅对主程序区有保护作用，对信息区无保护作用。私有代码读保护区有效时，是可以执行芯片主程序区全擦除的动作，执行完芯片主程序区全擦动作之后，私有代码读保护区擦写保护失效。（如果未更改私有代码读保护区配置，下次启动 MSC，擦写保护仍然有效）。其中私有代码读保护信息存储在加密信息区 CFG\_PCROP0 中。其中 ROP0\_ST、ROP0\_ED 配置一段私有代码读保护域的起始页号和结束页号。ROP0\_EN 配置两段区域是否使能私有代码读保护。

#### 5.4.3 Flash全局读保护

Flash 存储器可以进行全局读保护设置，全局读保护等级分为 Level0、Level1、Level2。全局读保护信息存储在 CFG\_GBRDP 中，处于配置信息区 INFO1 中。

加密级别	CFG_GBRDP	说明
Level0	0xFF	Flash 全局读保护无效
Level 1	0x00~0xFF	Flash 全局读保护级别 1
Level 2	0x00	Flash 全局读保护级别 2

表 5-2 Flash 全局读保护级别

存储区	保护级别	调试功能、 在 BOOTROM 或 SRAM 运行程序			在 Flash 运行程序		
		读	写	擦除	读	写	擦除
eFlash 主程序区	Level1	否	否	全擦	是	是	页擦
	Level2	否	否	否	是	是	页擦
校准信息区 INFO0	Level1	是	否	否	是	否	否
	Level2	是	否	否	是	否	否
配置信息区 INFO1	Level1	是	是	页擦	是	是	页擦
	Level2	是	否	否	是	否	否
加密信息区 INFO2	Level1	是	是	全擦+页擦	是	是	否
	Level2	是	否	否	是	否	否
保留信息区 INFO3	Level1	是	否	否	是	否	否
	Level2	是	否	否	是	否	否

表 5-3 Flash 全局读保护控制权限

注 1: 保留信息区 INFO3 用于存储 BOOTROM 程序, 在芯片量产 CP/FT 阶段烧录到芯片信息区 3;  
 注 2: 全局读保护为 Level2 时, 禁止从 BOOTROM 启动、在线调试功能禁止;  
 注 3: 全局读保护为 Level1 时, 程序运行在配置信息区 INFO1 和加密信息区 INFO2 不可访问主程序区;  
 注 4: 当对“配置信息区 INFO1”进行页擦(将全局读保护级别从 Level1 降到 Level0), 会自动对主 Flash 区域中非私有代码保护区域进行擦除(处于写保护区间的页也会被擦除, 页写保护失效);

#### 5.4.4 Data Flash划分

Flash 存储器可以配置一段数据 Flash 区域用于存储数据, 存储在芯片配置信息区 CFG\_DAFLS 中。其中 DFLS\_ST、DFLAS\_ED 配置数据 Flash 区域的起始页号和结束页号, DFLS\_EN 控制数据 Flash 是否有效。

DFLS_EN	起始页和结束页设置	保护区域
0	XXX	Flash 所有页为 Code Flash
1	DFLS_ST<DFLS_ED	DFLS_ST 和 DFLS_ED 之间的页为 Data Flash, 其它页为 Code Flash
1	DFLS_ST>DFLS_ED	DFLS_ST 和 DFLS_ED 之间的页为 Code Flash, 其它页为 Data Flash
1	DFLS_ST=DFLS_ED	页 DFLS_ST 为 Data Flash, 其它页为 Code Flash

表 5-4 数据 Flash 配置

#### 5.4.5 Flash操作请求流程

1. 首先通过 MSC\_CR 控制寄存器, 置位访问 FLASH 的请求信号 FLASH\_REQ, 查询得到允许操作应答信号 FLASH\_ACK;



2. 再进行对应的 FLASH 操作，包括 FLASH 全擦除、页擦除和编程三种操作；
3. 操作完成后，清除 FLASH 请求信号 FLASH\_REQ，查询应答信号 FLASH\_ACK 也被清零后，结束本次 FLASH 的访问操作；

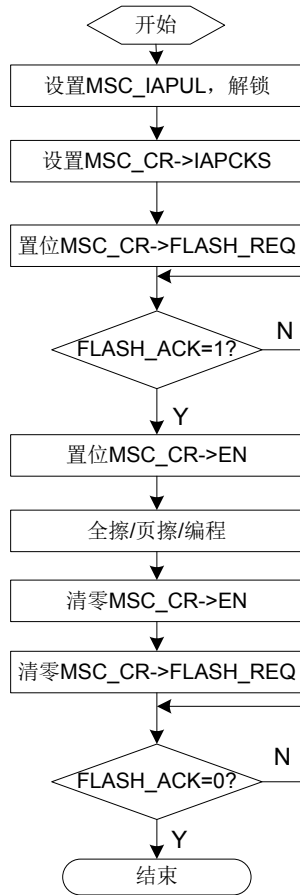


图 5-2 FLASH 操作请求流程

注：对 Flash 进行擦除、编程之前，需要禁止所有的中断，避免擦除、编程操作期间产生中断。

#### 5.4.6 Flash程序区全擦除模式

程序区全擦除可擦除全部程序区空间，一次全擦除耗时约 15ms。具体步骤如下：

1. 对 FLASH 操作进行解锁，设置寄存器 MSC\_UL；
2. 设置全擦命令，触发 FLASH 全擦除动作，设置寄存器 MSC\_IAPT=0x0000\_51AE；
3. 判断全擦除状态，判断寄存器 MSC\_SR-> BUSY 和 MSC\_SR-> MERASE\_END 状态；
4. 完成全擦除操作；

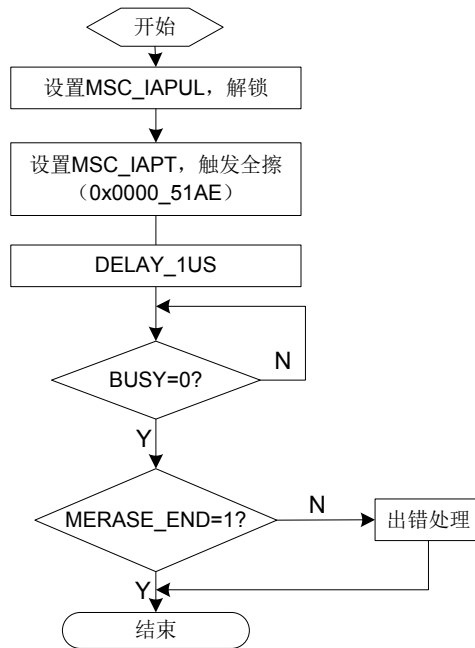


图 5-3 FLASH 全擦除模式流程图

#### 5.4.7 Flash页擦除模式

页擦除可擦除固定一页空间，其中程序区一页大小为 4KB，一页擦除耗时约 15ms。具体步骤如下：

1. 对 FLASH 操作进行解锁，设置寄存器 MSC\_UL；
2. 写入需擦除页的首地址；
3. 读取标志位 MSC\_SR->PCRDP\_F 和 MSC\_SR->WRP\_F，判断操作的页是否处于写保护区或私有代码读保护区；
4. 设置页擦命令，触发 FLASH 页擦除动作，设置寄存器 MSC\_IAPT=0x0000\_5EA1 或 0x0000\_AEA1；
5. 判断页擦除状态，判断寄存器 MSC\_SR->BUSY 和 MSC\_SR->PERASE\_END 状态；
6. 完成页擦除操作。

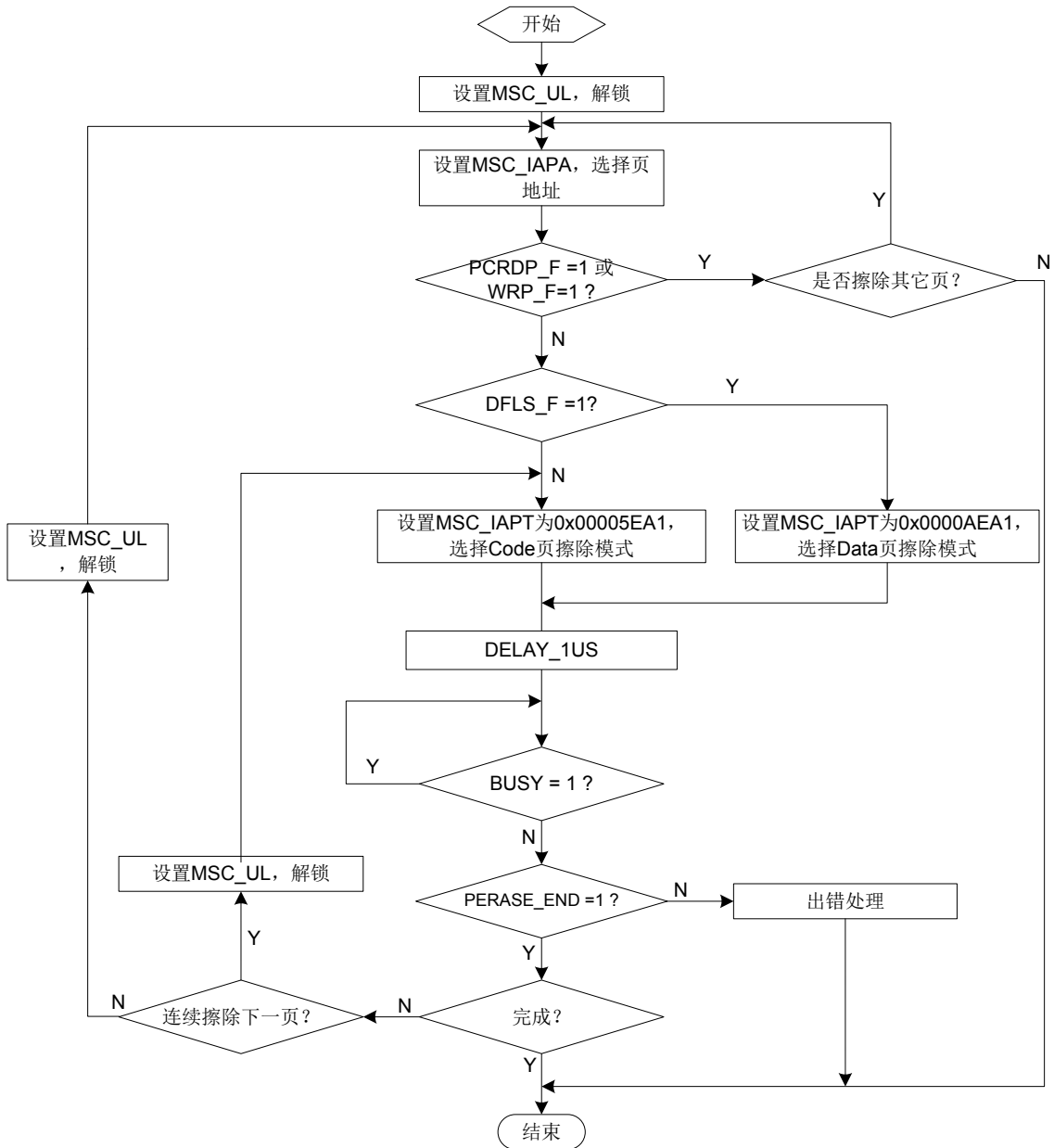


图 5-4 FLASH 页擦除模式流程图

#### 5.4.8 Flash双字编程模式

程序区双字编程可一次编程 8 Bytes 空间，一次双字编程耗时约 12us。具体步骤如下：

1. 对 FLASH 操作进行解锁，设置寄存器 MSC\_UL；
2. 写入需编程地址；
3. 读取标志位 MSC\_SR->PCRPD\_F 和 MSC\_SR->WRP\_F，判断操作的页是否处于写保护区或私有代码读保护区；
4. 写入需编程数据，程序区需同时写入 MSC\_IAPDL 和 MSC\_IAPDH；
5. 设置双字编程命令，触发 FLASH 双字编程动作，设置寄存器

- MSC\_IAPT=0x0000\_5DA2 或 0x0000\_ADA2;
6. 等待 BUSY 标志再次变为空闲状态;
  7. 判断 WPROG\_END 标志位是否置起。

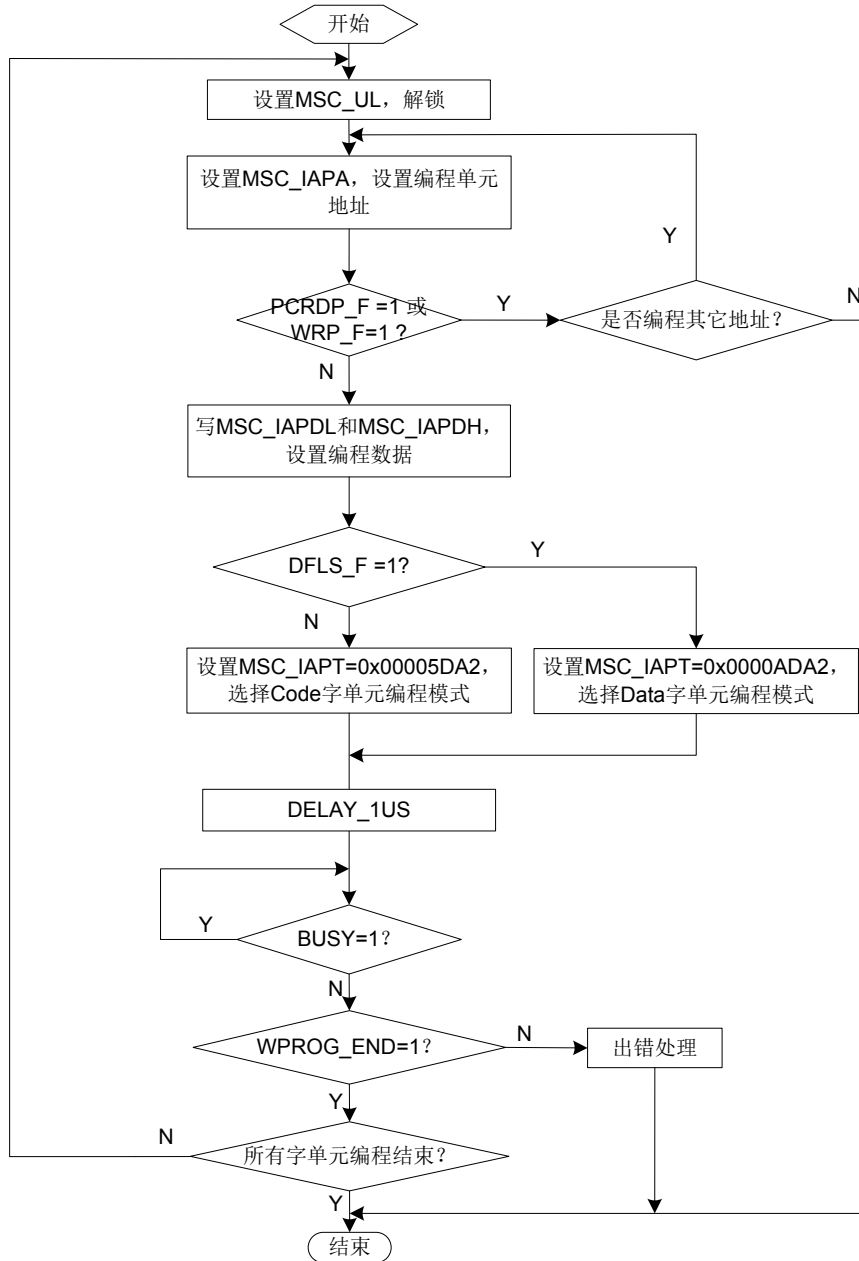


图 5-5 FLASH 双字编程模式流程图

#### 5.4.9 Flash快速编程模式

快速编程模式，可以连续进行一行编程（行大小：256Bytes），进行编程数据寄存器写之前不需要解锁，具体步骤如下：

1. 对 FLASH 操作进行解锁，设置寄存器 MSC\_UL;

2. 写入需编程首地址；
3. 读取标志位 MSC\_SR->PCRDP\_F 和 MSC\_SR->WRP\_F，判断操作的页是否处于写保护区或私有代码读保护区；
4. 设置 MSC\_IAPT 为 0x00005BA4 或 0x0000ABA4，进入快速编程模式；
5. 等待 BUSY 标志变为空闲状态；
6. 判断 PPROGS\_END 标志位是否置起；
7. 对 FLASH 操作进行解锁，设置寄存器 MSC\_UL；
8. 写入需编程数据，程序区需同时写入 MSC\_IAPDL 和 MSC\_IAPDH；
9. 设置快速编程命令，触发 FLASH 编程动作，设置寄存器 MSC\_IAPT=0x0000\_59A6 或 0x0000\_A9A6；
10. 等待 BUSY 标志再次变为空闲状态；
11. 判断 PPROGD\_END 标志位是否置起；
12. 判断行是否结束；
13. 设置 MSC\_IAPT 为 0x000058A7 或 0x0000A8A7，退出快速编程模式；
14. 等待 BUSY 标志再次变为空闲状态；
15. 判断 PPROGE\_END 标志位是否置起。

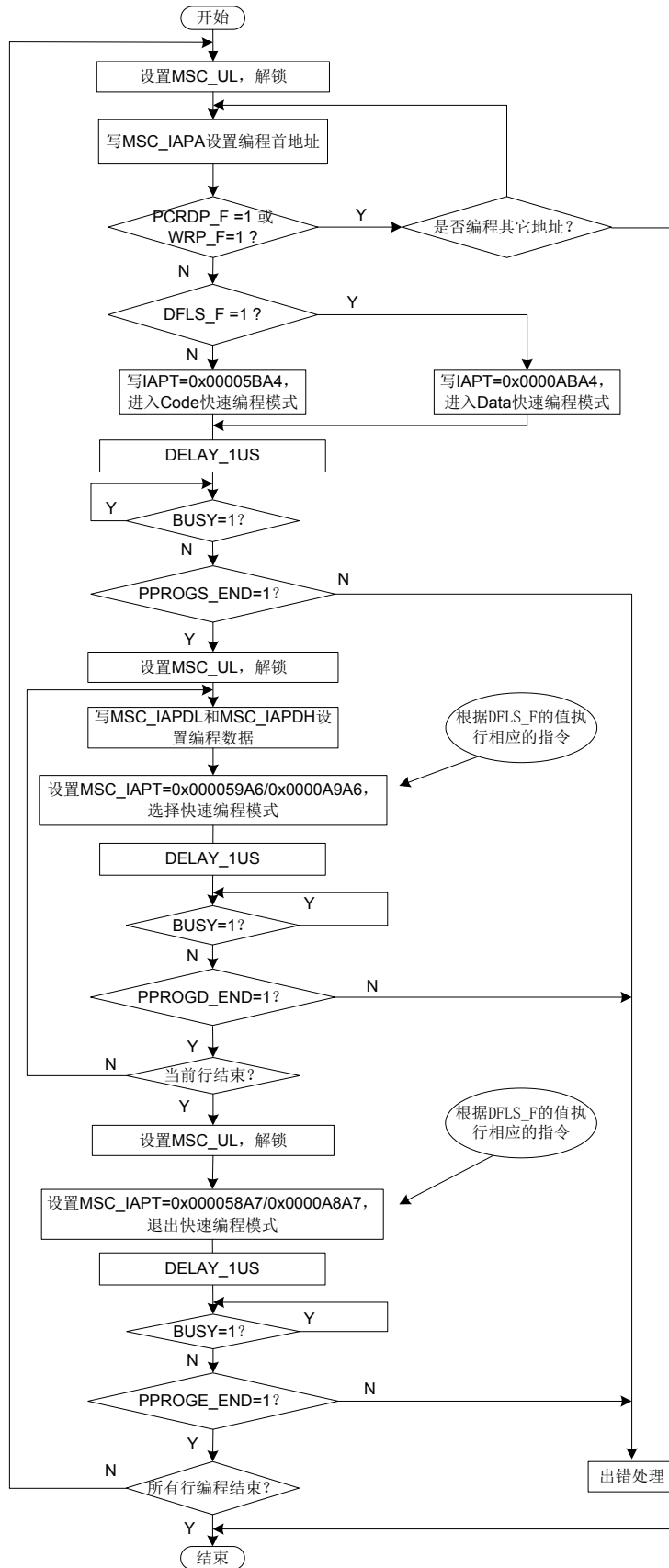


图 5-6 FLASH 快速编程模式流程图

## 5.4.10 自编程硬件固化函数

芯片内置自编程固化函数，由硬件电路实现，在自编程操作程序中可以调用这些自编程固化函数，以减少 SRAM 中的自编程操作代码量。

### 5.4.10.1 IAP\_PageErase

Code Flash 单页擦除函数：

```
uint32_t IAP_PageErase (uint32_t adr, uint32_t clk)
```

参数说明：

- ◇ 入口地址：保存在 0x10000000 单元内
- ◇ 参数输入：adr-擦除页内地址， clk-工作频率选择
- ◇ 参数输出：1 为成功，0 为失败

失败情况举例：

- ◇ 传参出错
- ◇ 擦除 DATA FLASH
- ◇ 擦除 MAIN 区加密页
- ◇ 擦除 INFO0、INFO2（全擦前）、擦除 INFO3
- ◇ 擦除写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时，仍对 FLASH 进行访问

### 5.4.10.2 IAP\_DWordProgram

Code Flash 双字编程函数：

```
uint32_t IAP_DWordProgram (uint32_t adr, uint32_t data0, uint32_t data1, uint32_t clk)
```

参数说明：

- ◇ 入口地址：保存在 0x10000004 单元内
- ◇ 参数输入：adr-编程首地址， data0-低地址编程数据， data1-高地址编程数据， clk-工作频率选择
- ◇ 参数输出：1 为成功，0 为失败

传参注意事项：

- ◇ adr 必须双字对齐即低三位必须为 0

失败情况举例：

- ◇ 传参出错
- ◇ 编程 DATA FLASH
- ◇ 编程 MAIN 区加密页

- ◇ 编程 INFO0、编程 INFO3
- ◇ 编程写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时，仍对 FLASH 进行访问

#### 5.4.10.3 IAP\_FastProgram

Code Flash 快速编程函数：

```
uint32_t IAP_FastProgram(uint32_t adr, uint8_t *buf, uint32_t byte_size, uint32_t epif,
uint32_t clk)
```

参数说明：

- ◇ 入口地址：保存在 0x10000008 单元内
- ◇ 参数输入：**adr**-编程首地址， **buf**-编程数据首地址， **byte\_size**-编程数据长度（以字节为单位）， **epif** -当编程到页首时是否先进行页擦除（1 为擦除， 0 为不擦除）， **clk**-工作频率选择
- ◇ 参数输出：1 为成功，0 为失败

传参注意事项：

- ◇ **adr** 必须双字对齐即低三位必须为 0
- ◇ **buf** 必须指向字节单元
- ◇ **byte\_size** 必须双字对齐即低三位必须为 0

失败情况举例：

- ◇ 传参出错
- ◇ 编程 DATA FLASH
- ◇ 编程 MAIN 区加密页
- ◇ 编程 INFO0、编程 INFO3
- ◇ 编程写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时，仍对 FLASH 进行访问

注：采用 IAP 行编程命令，支持从行中间地址开始编程。

#### 5.4.10.4 IAP\_PageErase\_DF

Data Flash 单页擦除函数：

```
uint32_t IAP_PageErase_DF (uint32_t adr, uint32_t clk)
```

参数说明：

- ◇ 入口地址：保存在 0x1000000C 单元内
- ◇ 参数输入：**adr**-擦除页内地址， **clk**-工作频率选择
- ◇ 参数输出：1 为成功，0 为失败



失败情况举例:

- ◇ 传参出错
- ◇ 擦除 CODE FLASH
- ◇ 擦除 MAIN 区加密页
- ◇ 擦除 INFO0~3
- ◇ 擦除写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时, 仍对 FLASH 进行访问

#### 5. 4. 10. 5 IAP\_DWordProgram\_DF

Data Flash 双字编程函数:

```
uint32_t IAP_DWordProgram_DF (uint32_t adr, uint32_t data0, uint32_t data1,  
uint32_t clk)
```

参数说明:

- ◇ 入口地址: 保存在 0x10000010 单元内
- ◇ 参数输入: adr-编程首地址, data0-低地址编程数据, data1-高地址编程数据, clk-工作频率选择
- ◇ 参数输出: 1 为成功, 0 为失败
- ◇ 传参注意事项:
- ◇ Adr 必须双字对齐即低三位必须为 0

失败情况举例:

- ◇ 传参出错
- ◇ 编程 CODE FLASH
- ◇ 编程 MAIN 区加密页
- ◇ 编程 INFO0~3
- ◇ 编程写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时, 仍对 FLASH 进行访问

#### 5. 4. 10. 6 IAP\_FastProgram\_DF

Data Flash 快速编程函数:

```
uint32_t IAP_FastProgram_DF (uint32_t adr, uint8_t *buf, uint32_t byte_size, uint32_t  
epif, uint32_t clk)
```

参数说明:

- ◇ 入口地址: 保存在 0x10000014 单元内
- ◇ 参数输入: adr-编程首地址, buf-编程数据首地址, byte\_size-编程数据长度 (以

字节为单位), epif -当编程到页首时是否先进行页擦除 (1 为擦除, 0 为不擦除),  
clk-工作频率选择

- ◇ 参数输出: 1 为成功, 0 为失败

传参注意事项:

- ◇ Adr 必须双字对齐即低三位必须为 0
- ◇ buf 必须指向字节单元

失败情况举例:

- ◇ 传参出错
- ◇ 编程 CODE FLASH
- ◇ 编程 MAIN 区加密页
- ◇ 编程 INFO0~3
- ◇ 编程写保护页
- ◇ 调用 IAPROM 的代码执行域的访问权限被禁止时, 仍对 FLASH 进行访问

注: 采用 IAP 行编程命令, 支持从行中间地址开始编程。

#### 5.4.10.7 IAP\_EraseAll

Flash 全擦除函数, 擦除区间包括主程序区和加密信息区 INFO2 两块:

uint32\_t IAP\_EraseAll (uint32\_t clk)

参数说明:

- ◇ 入口地址: 保存在 0x10000018 单元内
- ◇ 参数输入: clk-工作频率选择
- ◇ 参数输出: 1 为成功, 0 为失败

注: 对加密信息区 INFO2 擦除时, 可以调用此固化函数;

## 5.5 特殊功能寄存器

### 5.5.1 寄存器列表

MSC 寄存器列表			
名称	偏移地址	类型	描述
MSC 基地址: 4008_1000 <sub>H</sub>			
MSC_IAPCR	000 <sub>H</sub>	RW	FLASH 控制状态寄存器
MSC_IAPA	004 <sub>H</sub>	RW	FLASH 擦除编程地址寄存器

MSC 寄存器列表			
名称	偏移地址	类型	描述
MSC 基地址: 4008_1000 <sub>H</sub>			
-	008 <sub>H</sub>	-	-
MSC_IAPT	00C <sub>H</sub>	R/W	FLASH 控制命令寄存器
MSC_IAPUL	010 <sub>H</sub>	RW	FLASH 控制解锁寄存器
MSC_IAPSR	014 <sub>H</sub>	RW	FLASH 状态寄存器
MSC_IAPDL	018 <sub>H</sub>	R/W	FLASH 编程数据低位寄存器
MSC_IAPDH	01C <sub>H</sub>	R/W	FLASH 编程数据高位寄存器
-	020 <sub>H</sub> ~024 <sub>H</sub>	-	-
MSC_MEMWAIT	028 <sub>H</sub>	RW	存储器读取等待时间寄存器
-	02C <sub>H</sub>	-	-
MSC_FREPSR	030 <sub>H</sub>	R	FLASH 修复信息状态寄存器

## 5.5.2 寄存器描述

### 5.5.2.1 Flash控制状态寄存器 (MSC\_IAPCR)

FLASH 控制状态寄存器 (MSC_IAPCR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00010000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														IAPCKS			FLASH_ACK	FLASH_REQ	IAPRST	IAPEN											

Reserved	Bit 31-7	—	保留
IAPCKS	Bit 6~4	R/W	自编程模块工作频率 (自编程模块工作主频) 3'b000: 48.00MHz 3'b001: 24.00MHz 3'b010: 12.00MHz 3'b011: 6.00MHz 3'b100: 3.00MHz 3'b101: 16.00MHz 3'b110: 8.00MHz 3'b111: 4.00MHz 注: 主要用于模块内部产生 1MHz 的参考时钟
FLASH_ACK	Bit 3	R	<b>FLASH 操作许可状态位</b> 0: 禁止操作 1: 允许操作
FLASH_REQ	Bit 2	R/W	<b>FLASH 操作请求使能位</b> 0: 禁止

			1: 使能
IAPRST	Bit 1	C_W1	自编程复位 0: 无效 1: 复位自编程 读操作时始终为 0
IAPEN	Bit 0	R/W	自编程使能 0: 禁止 1: 使能

注：对该寄存器写之前需要先设置寄存器 MSC\_IAPUL，去除寄存器的写保护。

### 5.5.2.2 Flash擦除编程地址寄存器 (MSC\_IAPA)

FLASH 擦除编程地址寄存器 (MSC_IAPA)																																
偏移地址: 04 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												IFREN	IAPPA						IAPCA													

Reserved	Bit 31-20	—	保留
IFREN	Bit 19	R/W	信息区使能位 0: 禁止 1: 使能
IAPPA	Bit 18~12	R/W	FLASH 页地址 0x00~0x7F:第 0 页~第 127 页
IAPCA	Bit 11~0	R/W	FLASH 页内字节地址 1): 编程最小单位是 8 字节,地址为双字对齐方式; 2): 擦除最小单位为 4K 字节; 3): 双字编程/快速编程, 每次操作完成地址+8 字节; 4): IAPCA[11:8]为页内行地址, 每一页有 16 行; 5): IAPCA[ 7:0]为行内单元地址, 快速编程时仅在 IAPCA[ 7:0]有效, 超出将循环;

注:

- 1、对该寄存器写之前需要先设置寄存器 MSC\_IAPUL，去除寄存器的写保护；
- 2、在进行程序区页擦除时，完成页擦除后，IAPPA 自动+1；但是在进行 FLASH INFO 区页擦除时，IAPPA 不会自动加 1，需要重新填写下一个 IAPPA 页地址，触发页擦除动作；
- 3、芯片具有 4 个信息区 INFO0~INFO3，每个信息区的大小为 4KB；  
INFO0 信息区存放的是芯片校准值，自编程模块无法访问此信息区；（只可通过 ISP 访问）INFO0 对应的地址寄存

器 MSC\_IAPA 设置值为: 0x0008\_0000~0x0008\_0FFF;  
 INFO1 存放的是芯片配置字, 自编程模块可以对 INFO1 进行页擦除和编程操作; IAP 访问 INFO1 时, 地址寄存器 MSC\_IAPA 设置值为: 0x0008\_1000~0x0008\_1FFF;  
 INFO2 信息区存放的是芯片页加密信息 CFG\_PCROP0, 自编程模块在对 INFO2 进行页擦除之前, 首先必须先执行全擦除, 否则对 INFO2 页擦除无效; 对应地址寄存器 MSC\_IAPA 设置值为: 0x0008\_2000~0x0008\_2FFF;  
 INFO3 信息区用于存储 BOOTROM 程序 (在量产 CP/FT 阶段烧录), 自编程模块无法访问此信息区 (只可通过 ISP 访问); 对应地址寄存器 MSC\_IAPA 设置值为: 0x0008\_3000~0x0008\_3FFF;

### 5.5.2.3 Flash控制命令寄存器 (MSC\_IAPT)

FLASH 控制命令寄存器 (MSC_IAPT)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIG																															

TRIG	Bit 31-0	R/W	<p><b>FLASH控制命令</b>                  写寄存器:                  0x0000_51AE: 全擦除</p> <p><b>Code and INFO1/2/3 Flash 操作命令:</b>                  0x0000_5EA1: 页擦除                  0x0000_5DA2: 双字编程模式                  0x0000_5BA4: 进入快速编程模式                  0x0000_59A6: 进行快速编程模式                  0x0000_58A7: 退出快速编程模式</p> <p><b>Data Flash 操作命令:</b>                  0x0000_AEA1: 页擦除                  0x0000_ADA2: 双字编程                  0x0000_ABA4: 进入快速编程模式                  0x0000_A9A6: 进行快速编程模式                  0x0000_A8A7: 退出快速编程模式</p> <p>其它: 无操作                  读寄存器返回值:                  0x0000_0000: 空闲状态                  0x0000_0001: 页擦除操作状态                  0x0000_0002: 双字编程操作状态                  0x0000_0003: 进入快速编程操作状态</p>
------	----------	-----	--

			<p>0x0000_0004: 保留</p> <p>0x0000_0005: 快速编程操作状态</p> <p>0x0000_0006: 退出快速编程模式操作状态</p> <p>0x0000_0007: 全擦除操作状态</p> <p>0x0000_0008: DataFlash页擦除状态</p> <p>0x0000_0009: DataFlash双字编程状态</p> <p>0x0000_000A: DataFlash进入快速编程操作状态</p> <p>0x0000_000B: 保留</p> <p>0x0000_000C: DataFlash快速编程操作状态</p> <p>0x0000_000D: DataFlash退出快速编程模式操作状态</p>
--	--	--	--

注:

- 1、对该寄存器写之前需要先设置寄存器 MSC\_IAPUL，去除寄存器的写保护；
- 2、此寄存器写操作之后，寄存器写保护状态重新有效；
- 3、清零方式：1) 自编程操作完成后，硬件自动清零；2) 写自编程复位清零寄存器 IAPRST；
- 4、快速编程模式一次最多可以连续对一行数据进行编程，进行下一行数据编程时需要重新执行进入快速编程模式

### 5.5.2.4 Flash控制解锁寄存器 (MSC\_IAPUL)

FLASH 控制解锁寄存器 (MSC_IAPUL)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
≡																															

UL	Bit 31-0	R/W	<p><b>FLASH控制解锁</b></p> <p>写入0x0000_00A5，自编程模块解锁；</p> <p>进行以下任一操作，自编程模块上锁：</p> <ol style="list-style-type: none"> <li>1) 写入其他值</li> <li>2) 写寄存器 MSC_IAPT</li> <li>3) 写保留地址</li> <li>4) 写 IAPRST 软件复位</li> </ol> <p>读寄存器返回值：</p> <p>0x0000_0000: 解锁状态</p> <p>0x0000_0001: 非解锁状态</p>
----	----------	-----	--

5.5.2.5 Flash状态寄存器 (MSC\_IAPSR)

FLASH 状态寄存器 (MSC_IAPSR)																																										
偏移地址: 14 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved																DFLS_F	GBRDP_F	WRP_F	PCRDP_F	Reserved																MERASE_END	PPROGE_END	PPROGD_END	PPROGS_END	WPROG_END	PERASE_END	BUSY

Reserved	Bits 31-20	—	保留
DFLS_F	Bit 19	R	<b>Data/Code Flash 区操作指示位</b> 0: 操作区域为 Code Flash 1: 操作区域为 Data Flash
GBRDP_F	Bit 18	R	<b>全局读保护有效操作指示位</b> 0: 全局保护区域无效 1: 全局保护区域有效
WRP_F	Bit 17	R	<b>写保护区域操作指示位</b> 0: 未操作写保护区域 1: 操作写保护区域
PCRDP_F	Bit 16	R	<b>私有代码读保护区域操作指示位</b> 0: 未操作私有代码保护区域 1: 操作私有代码保护区域
Reserved	Bit 15~8	-	保留
Reserved	Bit 7	-	保留
MERASE_END	Bit 6	R	<b>FLASH 全擦除结束指示位</b> 0: 未进行或正在进行中 1: 当前全擦除完成
PPROGE_END	Bit 5	R	<b>FLASH 行编程模式退出指示位</b> 0: 未进行或正在进行中 1: 已经退出行编程模式
PPROGD_END	Bit 4	R	<b>FLASH 行编程模式下字编程结束指示位</b> 0: 未进行或正在进行中 1: 行编程模式下, 当前字编程完成
PPROGS_END	Bit 3	R	<b>FLASH 行编程模式启动指示位</b> 0: 未进行或正在进行中 1: 已经进入行编程模式
WPROG_END	Bit 2	R	<b>FLASH 双字编程结束指示位</b> 0: 未进行或正在进行中 1: 当前双字编程完成
PERASE_END	Bit 1	R	<b>FLASH 页擦除结束指示位</b>

			0: 未进行或正在进行 1: 当前页擦除完成
BUSY	Bit 0	R	自编程模块工作状态指示位 0: 空闲, 无编程/擦除操作 1: 自编程模块操作正在进行

注:

全局读保护有效操作指示位 (MSC\_IAPSR[18]) 说明:

- 1) FLASH 全局读保护级别为 Level1 时, 当程序运行在 SRAM 或 BootROM 中操作区域为主程序区时, 此标志位置 1, 自编程控制命令无效 (页擦除和编程无效);
- 2) FLASH 全局读保护级别为 Level1 时, 当程序运行在 SRAM 或 BootROM 中, FLASH 全擦除命令始终有效。当执行一次全擦除命令后, 此标志位清零, 后续自编程命令有效。
- 3) FLASH 全局读保护级别为 Level2 时, 当程序运行在 SRAM 中, 此标志位置 1, 自编程命令无效;
- 4) FLASH 全局读保护级别为 Level2 时, 当程序运行在 FLASH 中, 操作信息区时, 此标志位置 1, 操作信息区的自编程命令无效。

### 5.5.2.6 Flash编程数据低位寄存器 (MSC\_IAPDL)

FLASH 编程数据低位寄存器 (MSC_IAPDL)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAL																															

DATAL	Bit 31-0	R/W	FLASH 编程数据 Bit31~0
-------	----------	-----	--------------------

注:

- 1、对该寄存器写之前需要先设置寄存器 MSC\_IAPUL, 去除寄存器的写保护;



### 5.5.2.7 Flash编程数据低位寄存器 (MSC\_IAPDH)

FLASH 编程数据高位寄存器 (MSC_IAPDH)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAH																															

DATAH	Bit 31-0	R/W	FLASH 编程数据 Bit63~32
-------	----------	-----	---------------------

注:

- 1、对该寄存器写之前需要先设置寄存器 MSC\_IAPUL，去除寄存器的写保护;

### 5.5.2.8 存储器读取等待时间寄存器 (MSC\_MEMWAIT)

存储器读取等待时间寄存器 (MSC_MEMWAIT)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000001_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SRAM_W		Reserved				FLASH_W					

Reserved	Bits 31-10	—	保留
SRAM_W	Bits 9-8	R/W	<b>SRAM 读取等待时间位</b> 00: 无等待 01: 1 个 SYSCLK 10: 2 个 SYSCLK 11: 3 个 SYSCLK 注 1: 当系统时钟大于等于 48MHz 时, 等待时间必须大于等于 1 个 SYSCLK。
Reserved	Bits 7-4	—	保留
FLASH_W	Bits 3-0	R/W	<b>FLASH 读取等待时间选择位</b> 0000: 无等待 0001: 1 个 SYSCLK 0010: 2 个 SYSCLK ..... 1111: 15 个 SYSCLK

			<p>注 1: 当系统时钟高于 25MHz 时, 等待时间必须大于 1 个 SYSCLK 周期</p> <p>注 2: 当有低功耗需求时, 也可设置较大的等待时间可降低运行功耗</p>
--	--	--	--

### 5.5.2.9 Flash修复信息状态寄存器 (MSC\_FREPSR)

FLASH 修复信息状态寄存器 (MSC_FREPSR)																															
偏移地址: 30H																															
复位值: 00000000_00000001_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							REP1LS	REP1								Reserved							REP0LS	REP0							

Reserved	Bits 31-25	—	保留
REP1LS	Bit 24	R	<p><b>FLASH 冗余存储区 1, Repair 信息加载指示位</b></p> <p>0: 正确</p> <p>1: 错误</p> <p>当此标志位为 1 时, 表明在读取冗余信息区存储的 Repair 信息错误, 系统自动设置 REP1 的值为 0xFF。</p>
REP1	Bits 23-16	R	<p><b>FLASH 冗余存储区 1, 用于 Repair 状态</b></p> <p>[7:1]: 有缺陷 FLASH 存储区页地址</p> <p>[0]: 冗余存储区 1 是否用于 Repair 有缺陷的页。</p> <p>1'b1: 冗余存储区 1 不用于 Repair</p> <p>1'b0: 冗余存储区 1 用于 Repair</p>
Reserved	Bits 15-9	—	保留
REP0LS	Bit 8	R	<p><b>FLASH 冗余存储区 0, Repair 信息加载指示位</b></p> <p>0: 正确</p> <p>1: 错误</p> <p>当此标志位为 1 时, 表明在读取冗余信息区存储的 Repair 信息错误, 系统自动设置 REP0 的值为 0xFF。</p>
REP0	Bits 7-0	R	<p><b>FLASH 冗余存储区 0, 用于 Repair 状态</b></p> <p>[7:1]: 有缺陷 FLASH 存储区页地址</p> <p>[0]: 冗余存储区 0 是否用于 Repair 有缺陷的页。</p> <p>1'b1: 冗余存储区 0 不用于 Repair</p> <p>1'b0: 冗余存储区 0 用于 Repair</p>

## 第6章 系统配置控制器

### 6.1 概述

系统配置模块（SYSCFG）用于芯片的系统级功能配置。

### 6.2 特性

- ◇ 支持寄存器写保护功能
- ◇ 支持 Boot Rom 重映射

### 6.3 功能描述

#### 6.3.1 系统寄存器写保护

软件在寄存器 SYSCFG\_PROT 中写入 0x55AA6996 去除写保护。该寄存器保护范围为 SYSCFG、PMU、CMU、RMU 模块所有寄存器。

#### 6.3.2 存储器重映射

存储器重映射应用于系统的 3 种启动模式：

- ◇ Boot ROM 启动
- ◇ Boot Flash 启动
- ◇ 用户 Flash 启动

系统在发生上电复位、低电压复位或外部端口复位时，会固定从 Boot ROM 启动。Boot ROM 为系统固化，用户不可进行更改。通过对 BOOT 管脚的电平配置，决定是否需要启动，详见章节《存储器系统控制》中“启动引导”的介绍。

当 BOOT 管脚被配置为低电平时，进入 BootROM 程序后 20ms 内还没有接收到启动数据帧那么程序将被直接引导到用户 Flash 中运行，如果启动数据帧成功接收后连续 300ms 没有执行命令那么程序也将被直接引导到用户 Flash 执行用户程序。通过配置芯片配置字中的 Flash 启动地址选择位，来决定跳转至 Boot Flash 还是用户 Flash。用户根据实际需要编写 Boot Flash 的程序内容，在启动完成后置高 SYSCFG\_MEMRMP.BTF 位，并自行跳转至用户 Flash 区域。

当 BOOT 管脚被配置为高电平时，程序始终保持在 Boot ROM 中等待接收命令。

#### 6.3.3 定时器刹车源配置

通过配置 SYSCFG\_TBKCFG 寄存器可选择相应的定时器刹车事件。

- ◇ CPU 锁死
- ◇ LVD 事件
- ◇ 时钟安全事件

## 6.4 特殊功能寄存器

### 6.4.1 寄存器列表

SYSCFG 寄存器列表			
名称	偏移地址	类型	描述
SYSCFG 基地址: 4008_0000 <sub>H</sub>			
SYSCFG_PROT	000 <sub>H</sub>	R/W	系统写保护寄存器
SYSCFG_MEMRMP	004 <sub>H</sub>	R/W	存储器重映射寄存器
SYSCFG_RDPF	008 <sub>H</sub>	R/W	Flash 读保护区数据访问标志寄存器
SYSCFG_TIMBRK	00C <sub>H</sub>	R/W	TIMx 刹车触发控制寄存器
-	0F0 <sub>H</sub>	R/W	-
-	0F4 <sub>H</sub>	R/W	-
-	0F8 <sub>H</sub>	R/W	-

注：对上述寄存器写操作之前需要去除寄存器写保护，即写 0x55AA6996 数值到寄存器 SYSCFG\_PROT；

CFG 配置寄存器：

CFG 配置寄存器列表			
名称	偏移地址	类型	描述
CFG 基地址: 4008_3C00 <sub>H</sub>			
Reserved	000 <sub>H</sub>	-	保留
CFG_CHIP	004 <sub>H</sub>	R	芯片出厂配置寄存器
CFG_BOOT	008 <sub>H</sub>	R	芯片启动配置寄存器
CFG_RESV0	00C <sub>H</sub>	R	芯片保留配置寄存器 0
-	010 <sub>H</sub>	R	-
-	014 <sub>H</sub>	R	-
Reserved	018 <sub>H</sub> -02C <sub>H</sub>	-	保留
CFG_WORD0	100 <sub>H</sub>	R	芯片配置寄存器
CFG_WRPT0	104 <sub>H</sub>	R	写保护区 0 配置寄存器
CFG_WRPT1	108 <sub>H</sub>	R	写保护区 1 配置寄存器
CFG_DAFLS	10C <sub>H</sub>	R	数据 eFlash 配置寄存器
CFG_GBRDP	110 <sub>H</sub>	R	全局读保护配置寄存器
CFG_XTAL	114 <sub>H</sub>	R	晶振频率调校配置寄存器
Reserved	118 <sub>H</sub> -1FC <sub>H</sub>	-	保留
CFG_PENCRY0	200 <sub>H</sub>	R	私有代码读保护区 0 配置寄存器
CFG_PENCRY1	204 <sub>H</sub>	R	私有代码读保护区 1 配置寄存器

## 6.4.2 寄存器描述

### 6.4.2.1 系统写保护寄存器 (SYSCFG\_PROT)

系统写保护寄存器 (SYSCFG_PROT)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[31:1]																															PROT /KEY[0]

KEY	Bits 31-1	W	<b>保护关键码</b> 0x55AA6996: 去除写保护 其他: 开启写保护
PROT	Bit 0	R	<b>保护状态位</b> 0: 无写保护 1: 写保护

### 6.4.2.2 存储器重映射寄存器 (SYSCFG\_MEMRMP)

存储器重映射寄存器 (SYSCFG_MEMRMP)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							BTF	Reserved				BLD			

Reserved	Bits 31-9	—	保留
BTF	Bit 8	R/W	<b>Flash 程序区重映射选择位</b> 0: 程序从 BootFlash 启动 (地址: 0x0007_F000 ~ 0x0007_FFFF) 1: 程序从 AppFlash 启动 (地址: 0x0000_0000 ~ 0x0007_FFFF) 注: 1) 当芯片配置字 CFG_WORD0[12]的值为 0 时, 固定从 AppFlash 启动; 2) 当芯片配置字 CFG_WORD0[12]的值为 1 时, 程序启动位置受 BTF 的值控制;

			3) 用户从 BootFlash 退出进入 AppFlash 之前, 需要将 BTF 控制位置 1;															
			<table border="1"> <thead> <tr> <th>CFG_WORD0[12]</th> <th>BTF</th> <th>程序启动位置</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>AppFlash</td> </tr> <tr> <td>0</td> <td>1</td> <td>AppFlash</td> </tr> <tr> <td>1</td> <td>0</td> <td>BootFlash</td> </tr> <tr> <td>1</td> <td>1</td> <td>AppFlash</td> </tr> </tbody> </table>	CFG_WORD0[12]	BTF	程序启动位置	0	0	AppFlash	0	1	AppFlash	1	0	BootFlash	1	1	AppFlash
CFG_WORD0[12]	BTF	程序启动位置																
0	0	AppFlash																
0	1	AppFlash																
1	0	BootFlash																
1	1	AppFlash																
-	Bit 7 ~1	-	-															
BLD	Bit 0	R/W	<b>重映射模式选择位</b> 0: 程序从 Boot Rom 启动 1: 程序从 Flash 启动 (AppFlash 或 BootFlash) 注: BLD 的优先级高于 BTF, 即系统先执行 BLD 的配置后执行 BTF 的配置															

### 6.4.2.3 Flash读保护区数据访问标志寄存器 (SYSCFG\_RDPF)

Flash 读保护区数据访问标志寄存器 (SYSCFG_RDPF)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												PCROP_F	GBRDP_F		

Reserved	Bits 31-2	—	保留
PCROP_F	Bit 1	C_W1	<b>私有代码读保护区数据访问标志位</b> 0: 未访问 Flash 私有代码区数据 1: 访问 Flash 私有代码区数据 (读出数据全 0)
GBRDP_F	Bit 0	C_W1	<b>全局读保护区数据访问标志位</b> 0: 未访问 Flash 程序区数据 1: 访问 Flash 程序区数据 (读出数据全 0)

注:

- 全局读保护 (CFG\_GBRDP) 有效时:
  - 全局读保护级别为 Level1: 程序运行在 SRAM 或 BootLoader 中时, 当访问 Flash 的程序区时, GBRDP\_F 标志位置 1, 同时从 Flash 中读出数据为全 0, 访问 Flash 信息区时, 此标志位不会置 1;
  - 全局读保护级别为 Level2: 程序运行在 SRAM 或 BootLoader 中时, 当访问 Flash 的程序区和信息区时, GBRDP\_F 标志位置 1, 同时从 Flash 中读出数据为全 0; 程序运行在 Flash 中时, 当访问信息区时, GBRDP\_F 标志位置 1, 从 Flash 信息区中读出数据为全 0;
- 私有代码读保护 (CFG\_PCROPx) 有效时: 读取私有代码保护区的数据时, PCROP\_F 标志位置 1, 同时读出的数据为全 0; (数据读取包括 Cortexm3 内核和 DMA 的读取操作);

### 6.4.2.4 TIMx刹车触发控制寄存器 (SYSCFG\_TIMBRK)

TIMx 刹车触发控制寄存器 (SYSCFG_TIMBRK)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										LOCKBRKE	LVD BRKE	CSSBRKE			

Reserved	Bits 31-3	—	保留
LOCKBRKE	Bit2	R/W	<b>系统 LOCKUP 刹车触发控制位</b> 0: 无效 1: 有效 注: 当内核产生 lockup 时, 会产生刹车信号
LVD BRKE	Bit1	R/W	<b>LVD 刹车触发控制位</b> 0: 无效 1: 有效 注: 当工作电压低于 LVD 设置的阈值电压时, 会产生刹车信号
CSSBRKE	Bit 0	R/W	<b>系统时钟失效刹车触发使能位</b> 0: 无效 1: 有效 注: 当系统时钟选择为 PLL 或 HOSC 时, 系统时钟停振时会产生刹车信号

### 6.4.2.5 芯片出厂配置寄存器 (CFG\_CHIP)

芯片出厂配置寄存器 (CFG_CHIP)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_01100111_11111111_10011000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved										SRAM_SIZE	Reserved	FLASH_SIZE			

Reserved	Bits 31-16	—	保留
Reserved	Bits 15-7	—	保留
SRAM_SIZE	Bits 6-4	R	<b>SRAM 容量大小指示位:</b> 000: 96KB

			001: 64KB (默认) 010: 48KB 011: 32KB 100: 24KB 1x1: 16KB
Reserved	Bit 3	—	保留
FLASH_SIZE	Bits 2-0	R	<b>FLASH 容量大小指示位:</b> 000: 512KB (默认) 001: 384KB 010: 256KB 011: 192KB 1xx: 128KB

#### 6.4.2.6 芯片启动配置寄存器 (CFG\_BOOT)

芯片启动配置寄存器 (CFG_BOOT)																															
偏移地址: 008H																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																BOOT															

BOOT	Bits 31-0	R	<b>芯片上电复位启动位置选择:</b> 高低 16 位不满足取反逻辑: 芯片上电默认从 Flash 启动 高低 16 位满足取反逻辑: 芯片上电默认从 BOOTLOADER 启动。
------	-----------	---	---

#### 6.4.2.7 芯片配置字寄存器 (CFG\_WORD0)

芯片配置字寄存器 (CFG_WORD0)																																			
偏移地址: 100H																																			
复位值: 00010000_00001000_11101111_11110111 <sub>B</sub>																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																BOR09VS	WWDTEN	BOOT	BOR33VS	PWRTEN	XTAL	Reserved										IWDTEN	TM_ENB	LOSMEN	LOSCEN



Reserved	Bits 31-16	—	保留
BOR09VS	Bits 15-14	R	<b>BOR09 电压点选择位</b> 00: 0.75V 01: 0.65V 10: 0.55V 11: 由软件控制
WWDTEN	Bit 13	R	<b>WWDT 使能位</b> 0: WWDT 禁止工作 1: WWDT 可工作 注: 该位置 0 时, 软件使能 WWDT 无效, WWDT 禁止工作; 该位置 1 时, 可通过软件使能 WWDT 工作。
BOOT	Bit 12	R	<b>eFlash 启动地址选择位</b> 0: 0x0000_0000 1: 0x0007_F000
BOR33VS	Bits 11-10	R	<b>BOR33 电压点选择位</b> 00: 2.5V 01: 2.0V 10: 1.5V 11: 由软件控制
PWRTEN	Bit 9	R	<b>上电延时使能位</b> 0: 禁止 1: 使能
XTAL	Bit 8	R	<b>外部振荡器模式选择位</b> 0: 1~8MHz 1: 8~24MHz
Reserved	Bits 7-4	R	保留
IWDTEN	Bit 3	R	<b>IWDT 使能位</b> 0: 由软件使能 1: 硬件强制使能 注: 硬件强制使能后, 软件无法关闭; 中断强制使能, 软件无法关闭; 复位强制使能, 软件无法关闭; 时钟源固定为 ULRC, 软件无法切换。
-	Bit 2	R	-
LOSMEN	Bit 1	R	<b>LOSC 安全管理使能位</b> 0: 禁止 1: 使能
LOSCEN	Bit 0	R	<b>LOSC 使能位</b> 0: 由软件使能 1: 硬件强制使能 注: 硬件使能后, 软件无法关闭

### 6.4.2.8 写保护区 0 配置寄存器 (CFG\_WRPT0)

写保护区 0 配置寄存器 (CFG_WRPT0)																															
偏移地址: 104 <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WRP0_EN	WRP0_ED						Reserved	WRP0_ST							

Reserved	Bits 31-16	—	保留
WRP0_EN	Bits 15	R	<b>eFlash 写保护区 0 使能信号</b> 1'b0: 写保护使能 1'b1: 写保护禁止
WRP0_ED	Bits 14-8	R	<b>eFlash 写保护区 0, 结束页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
Reserved	Bits 7	R	保留
WRP0_ST	Bits 6-0	R	<b>eFlash 写保护区 0, 起始页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

### 6.4.2.9 写保护区 1 配置寄存器 (CFG\_WRPT1)

写保护区 1 配置寄存器 (CFG_WRPT1)																															
偏移地址: 108 <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WRP1_EN	WRP1_ED						Reserved	WRP1_ST							

Reserved	Bits 31-16	—	保留
WRP1_EN	Bits 15	R	<b>eFlash 写保护区 1 使能信号</b> 1'b0: 写保护使能 1'b1: 写保护禁止
WRP1_ED	Bits 14-8	R	<b>eFlash 写保护区 1, 结束页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
Reserved	Bits 7	R	保留
WRP1_ST	Bits 6-0	R	<b>eFlash 写保护区 1, 起始页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

#### 6.4.2.10 数据eFlash配置寄存器 (CFG\_DAFLS)

数据 eFlash 配置寄存器 (CFG_DAFLS)																															
偏移地址: 10C <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DFLS_EN	DFLS_ED						Reserved	DFLS_ST							

Reserved	Bits 31-16	—	保留
DFLS_EN	Bits 15	R	<b>数据 eFlash 使能信号</b> 1'b0: 数据 FLASH 有效 1'b1: 数据 FLASH 无效
DFLS_ED	Bits 14-8	R	<b>数据 eFlash 结束页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

Reserved	Bits 7	R	保留
DFLS_ST	Bits 6-0	R	数据 eFlash 起始页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

#### 6.4.2.11 全局读保护配置寄存器 (CFG\_GBRDP)

全局读保护配置寄存器 (CFG_GBRDP)																															
偏移地址: 110 <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								GDRDP							

Reserved	Bits 31-16	—	保留
Reserved	Bits 15-8	R	保留
GDRDP	Bits 7-0	R	全局读保护控制位 0xFF: 全局读保护级别为 0 (不加密) 0x00: 全局读保护级别为 2 其它: 全局读保护级别为 1

#### 6.4.2.12 晶振频率调校配置寄存器 (CFG\_XTAL)

晶振频率调校配置寄存器 (CFG_XTAL)																															
偏移地址: 114 <sub>H</sub>																															
复位值: 00000000_01111111_11111111_10000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								CAPCAL							

Reserved	Bits 31-16	—	保留
Reserved	Bits 15-8	R	保留
CAPCAL	Bits 7-0	R	XOSC 晶振频率调校值: 0000_0000: 最低频率

		..... 1111_1111: 最高频率
--	--	--------------------------

### 6.4.2.13 私有代码读保护区 0 配置寄存器 (CFG\_PENCRY0)

私有代码读保护区 0 配置寄存器 (CFG_PENCRY0)																																
偏移地址: 200 <sub>H</sub>																																
复位值: 11100000_00011111_00011111_11100000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																ROPO_EN	ROPO_ED							Reserved	ROPO_ST							

Reserved	Bits 31-16	—	保留
ROPO_EN	Bits 15	R	主程序保护区 0 使能信号 1'b0: 分区 0 加密使能 1'b1: 分区 0 加密禁止
ROPO_ED	Bits 14-8	R	主程序保护区 0, 结束页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
Reserved	Bits 7	R	保留
ROPO_ST	Bits 6-0	R	主程序保护区 0, 起始页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

### 6.4.2.14 私有代码读保护区 1 配置寄存器 (CFG\_PENCRY1)

私有代码读保护区 1 配置寄存器 (CFG_PENCRY1)																															
偏移地址: 204 <sub>H</sub>																															
复位值: 11100000_00011111_00011111_11100000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ROP1_EN	ROP1_ED						Reserved	ROP1_ST							

Reserved	Bits 31-16	—	保留
ROP1_EN	Bits 15	R	主程序保护区 1 使能信号 1'b0: 分区 1 加密使能 1'b1: 分区 1 加密禁止
ROP1_ED	Bits 14-8	R	主程序保护区 1, 结束页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
Reserved	Bits 7	R	保留
ROP1_ST	Bits 6-0	R	主程序保护区 1, 起始页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

注: 此寄存器无效, 设计中仅支持“私有代码读保护区 0 配置寄存器”;

## 第7章 电源管理及低功耗模式(PMU)

### 7.1 概述

电源管理单元 (PMU) 管理芯片的电源以及低功耗模式。在每个低功耗模式下，都有其对应的单元模块状态 (使能、中止或掉电)。芯片共分为 5 种功耗模式 RUN, SLEEP, STOP1, STOP2, STANDBY。其中 RUN 为芯片正常运行模式，所有的外设模块均可被使能，其余为低功耗模式。STOP2 为 CPU 最低可恢复模式，在 STOP2 模式时，CPU 暂停，大部分外设被中止处于掉电状态，部分 RAM 中数据保持，唤醒之后 CPU 从暂停处继续运行，大部分外设模块由于未作数据保持处理需要重新做初始化。STANDBY 模式下除了 POR、MRST 和备份 RTC 外所有的外设都处于掉电状态。

低功耗模式通过软件操作使能。SLEEP, STOP1, STOP2 可通过一系列中断或事件唤醒回到 RUN 模式，STANDBY 仅可通过上电复位、备份域 RTC 中断、外部 WKUP 引脚上升沿/下降沿或外部 MRST 复位唤醒回到 RUN 模式。STANDBY 模式下为进一步降低功耗，可以将所有的时钟模块关闭，PMU 支持无时钟唤醒模式。

芯片上电完成后，配置初始化时需要软件将 BKPC->PCR.LDOERPD 和 BKPC->PCR.DCBGPD 全部配置成 1。上述两个控制位的操作顺序有一定的要求，首先将 BKPC->PCR.LDOERPD 置 1，然后再将 BKPC->PCR.DCBGPD 置 1。芯片运行的整个过程中需要保持控制位 BKPC->PCR.LDOERPD 一直为高，即保证低功耗 LDO 的参考电压由 LDO 模块自身提供，控制位 BKPC->PCR.DCBGPD 可以根据 DC/DC 的工作模式的不同进行不同的配置。

### 7.2 特性

- ◇ 支持多种低功耗模式配置
- ◇ 支持多种唤醒源灵活配置
- ◇ 快速的唤醒时间

### 7.3 芯片电源

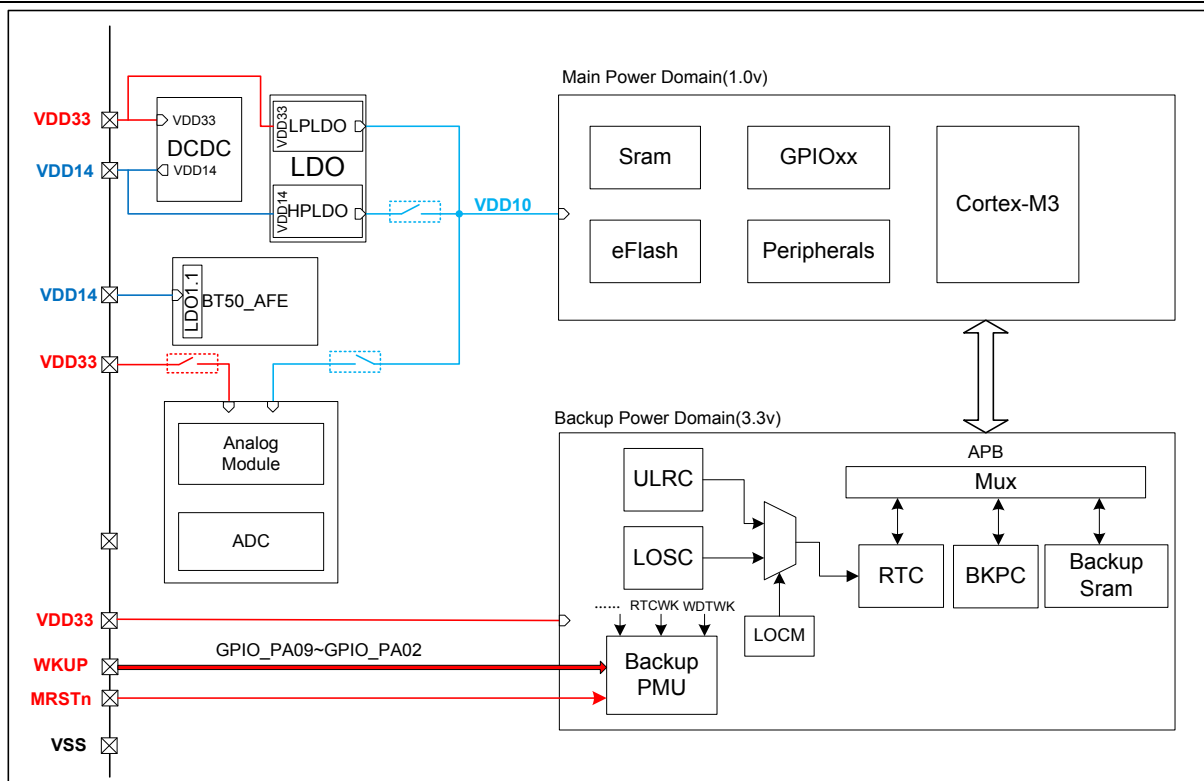


图 7-1 电源示意图

#### 7.3.1 主系统电源域

芯片工作电压 VDD33 要求介于 1.8 V 到 3.6V 之间。嵌入式 LDO 用于提供内部 1.0 V 数字电源。

备份域模块工作在高压域 3.3v，由外部 3.3v 电源直接供电。

#### 7.3.2 备份域电源

备份域电源通过外部 VDD33 电源管脚直接供电，处于常开模式，为如下模块供电：

- ◇ RTC
- ◇ LOSC
- ◇ ULRC
- ◇ PMU
- ◇ MRSTn
- ◇ GPIO\_PA09~GPIO\_PA02
- ◇ GPIO\_PB01~GPIO\_PB00（低速 32KHz 晶振输入管脚）



## 7.4 电源监视

### 7.4.1 上电复位POR

芯片内部集成 POR 产生电路。

当 VDD/VDDA 低于指定阈值 VPOR/VPDR 时，器件无需外部复位电路便会保持复位状态。

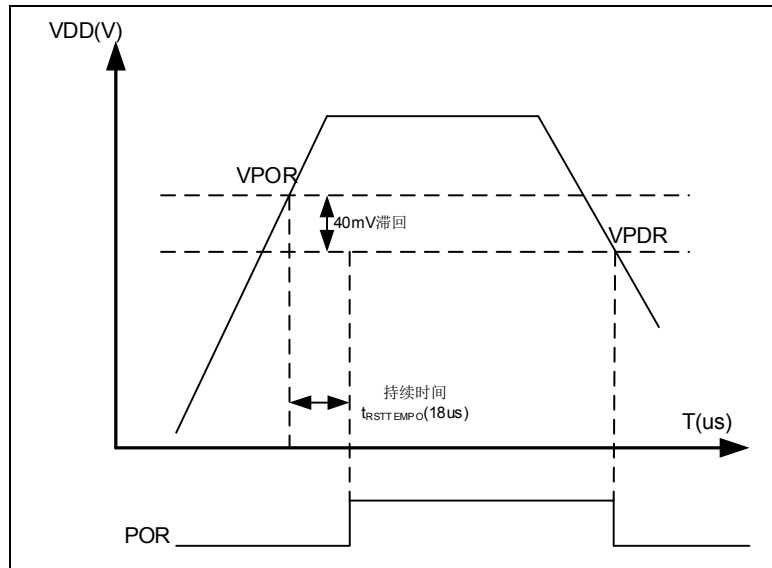


图 7-2 POR/PDR 示意图

### 7.4.2 欠压复位BOR

上电期间，欠压复位（BOR）将使器件保持复位状态，直到电源电压达到 1.6V。芯片默认 BOR 为开启状态，复位完成后，可通过软件选择 BOR 复位电压阈值 VBOR，或可将 BOR 禁止。芯片支持 8 个 VBOR 阈值选择。

当电源电压（VDD）降至所选 VBOR 阈值以下时，将使器件复位。

BOR 阈值滞回电压约为 100 mV（电源电压的上升沿与下降沿之间）。

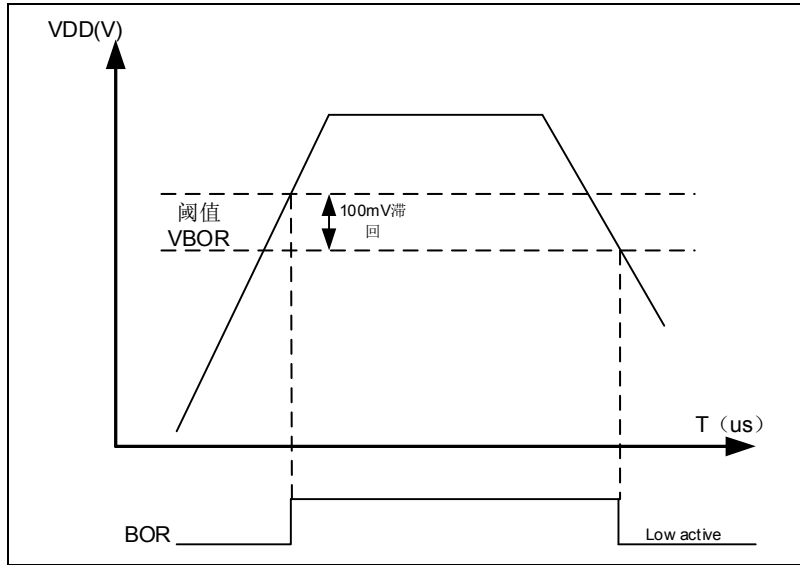


图 7-3 BOR 示意图

### 7.4.3 低电压检测LVD

LVD 可用于监视 VDD 电源，通过设置 LVDEN 使能 LVD，将 VDD 电压和 LVDS 所选择的电压值进行比较，可粗略判断当前电源 VDD 的电压值。

LVD 也可检测外部引脚输出（LVDIN）的电压值。

LVD 提供了一个状态标志位 LVDO，用于指示 VDD 是大于还是小于 LVD 阈值。通过使能 LVDIE 可使能 LVD 中断，通过选择 LVDIS 可选择 LVD 中断类型。当 VDD 降至 LVD 阈值以下以及/或者当 VDD 升至 LVD 阈值以上时，可以产生 LVD 中断，具体取决于 LVDIS 的中断类型配置。该功能的用处之一就是可以在 VDD 发生跌落时，立即进入中断服务程序中执行紧急关闭系统的任务，若外部有电池供电，则可进入低功耗模式并切换至电池供电。

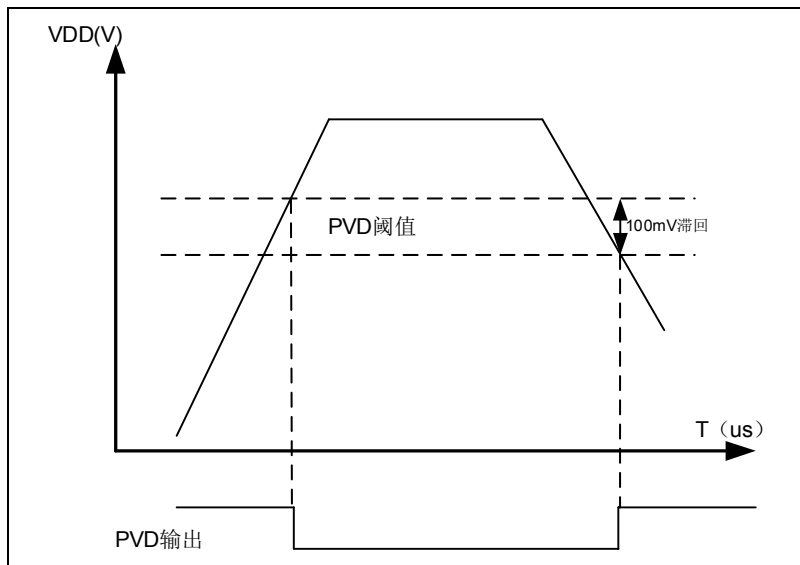


图 7-4 LVD 示意图

## 7.5 低功耗模式

### 7.5.1 低功耗模式转换

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

芯片有四个低功耗模式：

- ◇ SLEEP 模式（Cortex-M3 内核停止，外设保持运行）
- ◇ STOP1 模式（DMA 仍可动作，可配合低功耗外设，PIS 等在小系统内动作）
- ◇ STOP2 模式（DMA 关闭，仅部分低功耗外设可工作）
- ◇ STANDBY 模式（1.0V 主系统域断电,模拟 IP 3.3v 掉电）

关于低功耗模式下各模块的详细动作情况请参考 7.5.9。

此外，可通过下列方法之一降低运行模式的功耗：

- ◇ 降低系统时钟速度
- ◇ 不使用 APB 和 AHB 外设时，将对应的外设时钟关闭

进入低功耗模式的转换关系如下图所示：

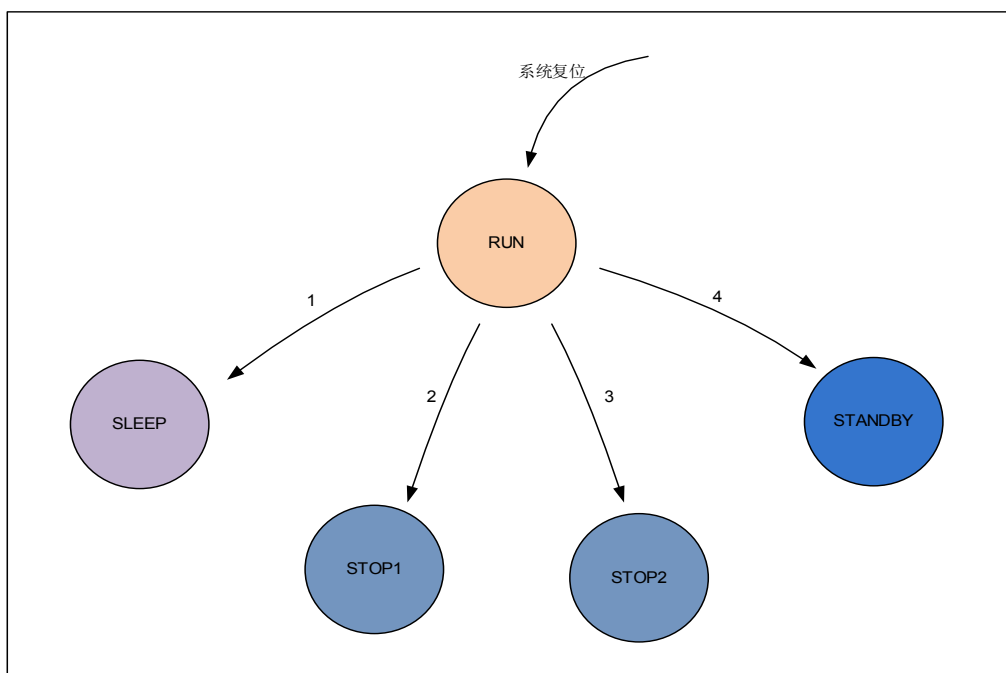


图 7-5 低功耗模式转换图

	模式	进入	唤醒	对逻辑电 路时钟的 影响	对时钟源的 影响	LDO/ DC-DC
1	SLEEP	WFI	任意中断	CPU 时钟 关闭	无	HPLDO/DC- DC 开启
		WFE	唤醒事件			
2	STOP1	LPM 位=00 +SLEEPDEEP 位 +WFI/WFE	具体请参照 表 3-5 STOP1 低功 耗模式的中 断唤醒源	具体请参 照 表 7-1 低功耗模 式下各模 块操作	PLL、HOSC 关闭 HRC 可使能	LPLDO 开启 /DC-DC 关闭
3	STOP2	LPM 位=01 +SLEEPDEEP 位 +WFI/WFE	具体请参照 表 3-6 STOP2 低功 耗模式的中 断唤醒源		PLL、HOSC、 HRC 关闭	LPLDO 开启 /DC-DC 关闭
4	STANDBY	LPM 位=1x +SLEEPDEEP 位 +WFI/WFE	具体请参照 表 3-7 STANDBY 低 功耗模式的 中断唤醒源		PLL、HOSC、 HRC 关闭	LDO/DC-DC 关闭

### 7.5.2 系统时钟速度

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（SYSCLK、HCLK、PCLK1 和 PCLK2）速度。进入睡眠模式之前，也可以使用这些预分频器降低外设速度。也可将系统时钟切换至低速时钟源并关闭高速时钟源来降低功耗。

系统时钟速度的有关详细信息，请参见[时钟管理](#)。

### 7.5.3 外设时钟门控

在运行模式下，可通过设置时钟门控来停止各外设和存储器的总线时钟或模块工作时钟以降低功耗。

要进一步降低低功耗模式的功耗，可在执行 WFI 或 WFE 指令之前可通过门控关闭外设时钟。

外设时钟门控配置的有关详细信息，请参见[时钟管理](#)。

#### 7.5.4 运行模式

- ◇ 所有高速时钟源可启用
- ◇ 所有外设可启用

#### 7.5.5 SLEEP模式

- ◇ 所有高速时钟源可启用
- ◇ CPU 时钟被关断
- ◇ 所有外设可启用

#### 7.5.6 STOP1 模式

- ◇ 所有高速时钟源被禁止
- ◇ 外设时钟和 CPU 时钟被禁止
- ◇ 低速时钟源可启用
- ◇ 可通过低速外设中断或外部管脚中断唤醒 CPU
- ◇ SRAM 和寄存器值保持

#### 7.5.7 STOP2 模式

- ◇ 高速和低速时钟源被禁止
- ◇ 可通过外部管脚中断、低速外设中断、LVD 中断唤醒
- ◇ IWDG 和使用 ULRC 时钟的低速外设可启用
- ◇ RTC 可启用
- ◇ 部分 SRAM 数据保持，部分 SRAM 数据是否保持可配置
- ◇ 大部分主系统电源域模块掉电，仅部分逻辑有电运行

#### 7.5.8 STANDBY模式

- ◇ 主系统电源域掉电
- ◇ RTC 可启用
- ◇ 备份域 SRAM 数据保持

#### 7.5.9 低功耗模式下各模块状态

下表列举了各模块在低功耗模式下的操作可能性，为低功耗应用提供参考。

	SLEEP	STOP1	STOP2	STANDBY
<b>内核</b>				
处理器系统	工作	停止	掉电（部分 Register Retention）	掉电
NVIC	工作	停止	掉电 (Retention)	掉电
调试	工作	停止	掉电	掉电
<b>存储器及存储器接口</b>				

	SLEEP	STOP1	STOP2	STANDBY
Flash	工作	停止(sleep)	掉电(shut down)	掉电
SRAM0 (0x2000_0000~0x2000_3FFF)	工作	停止(deep-sleep)	停止(deep-sleep)	掉电
SRAM1 (0x2000_4000~0x2000_7FFF)	工作/掉电	停止(deep-sleep)/掉电	停止(deep-sleep)/掉电	掉电
QSPI	可工作	停止	掉电	掉电
<b>系统模块</b>				
欠压检测	可工作	可工作	可工作	掉电
低电压检测	可工作	可工作	可工作	掉电
DMA 控制器	可工作	停止	掉电	掉电
外设互联	可工作	停止	掉电	掉电
独立看门狗定时器	可工作	可工作	可工作	掉电
窗口看门狗定时器	可工作	可工作	掉电	掉电
<b>时钟</b>				
LOSC(32KHz)	可工作	可工作	可工作	可工作
ULRC(32KHz)	可工作	可工作	可工作	可工作
LRC(128KHz)	可工作	可工作	可配置	掉电
HRC(2/24MHz)	可工作	停止	掉电	掉电
HOSC(16MHz)	可工作	掉电	掉电	掉电
PLL(48MHz)	可工作	掉电	掉电	掉电
Clkfilter	可工作	停止	掉电	掉电
内核时钟	停止	停止	掉电	掉电
系统时钟	工作	工作	工作	掉电
<b>外部接口</b>				
GPIO Function	可工作	停止	停止	掉电
GPIO PAD	工作	工作	PA2~PA9 工作	PA2~PA9 工作
<b>安全管理</b>				
CRC	可工作	停止	掉电	掉电
加密处理	可工作	停止	掉电	掉电
真随机发生器	可工作	停止	掉电	掉电
<b>模拟</b>				
ADC	可工作	停止	停止	掉电
Touchkey	可工作	停止	停止	掉电
BT50_AFE	可工作	停止	停止	掉电
HPLDO	工作	停止	停止	停止

	SLEEP	STOP1	STOP2	STANDBY
LPLDO	工作	工作	工作	停止
DC-DC	工作	停止	停止	停止
<b>定时器</b>				
高级定时器	可工作	停止	掉电	掉电
普通定时器	可工作	停止	掉电	掉电
基本定时器	可工作	停止	掉电	掉电
低功耗定时器	可工作	可工作	掉电	掉电
RTC	可工作	可工作	可工作	可工作
<b>通信</b>				
I2C 接口	可工作	停止	掉电	掉电
串行外设接口 (SPI)	可工作	停止	掉电	掉电
通用异步收发器(UART)	可工作	停止	掉电	掉电
Keyboard	可工作	可工作	掉电	掉电
正交译码器 (QDEC)	可工作	停止	掉电	掉电
<b>蓝牙(BLE5.0)</b>				
BB(BaseBand)	可工作	停止	掉电 (Control Register Retention)	掉电
BB LPM (Lower Power Module)	可工作	可工作	可工作	掉电
Exchange Memory0 (0x400C8000~0x400CBFFF)	可工作	停止(deep-sleep)	停止 (deep-sleep)	掉电
Exchange Memory1 (0x400CC000~0x400CDFFF)	可工作	停止	掉电/停止 (deep-sleep)	掉电
Modem	可工作	停止	掉电	掉电
<b>音频</b>				
PDM2PCM	可工作	停止	掉电	掉电
ADPCM	可工作	停止	掉电	掉电

表 7-1 低功耗模式下各模块操作

### 7.5.10 低功耗模式下唤醒源

下表列举了在低功耗模式下的唤醒源，为低功耗应用提供参考。

唤醒中断源	低功耗模式			
	SLEEP	STOP1	STOP2	STANDBY
WWDG	√	√		
IWDG	√	√	√	
LVD	√	√	√	
RTC_WAKE	√	√	√	√
RTC_ALARM	√	√	√	√
CMU	√			
ADC	√			
LP16T	√	√		
TIM4_IRQ (BS16T0)	√			
TIM5_IRQ (BS16T1)	√			
TIM0_IRQ (GP16C4T0)	√			
TIM1_IRQ (GP16C4T1)	√			
TIM2_BRK(AD16C4T0)	√			
TIM2_UP	√			
TIM2_TRIG_COM	√			
TIM2_CC	√			
TIM3_BRK(AD16C4T1)	√			
TIM3_UP	√			
TIM3_TRIG_COM	√			
TIM3_CC	√			
I2C0_EV	√			
I2C0_ERR	√			
I2C1_EV	√			
I2C1_ERR	√			
SPI0	√			
SPI1	√			
UART0	√			
UART1	√			
UART2	√			
PDM2PCM	√			
TK	√			
KSCAN	√			
QSPI	√			
CRYPT	√			
ADPCM	√			
SM3(HASH)	√			
ECC	√			



唤醒中断源	低功耗模式			
	SLEEP	STOP1	STOP2	STANDBY
TRNG	√			
EXTI0	√	√		
EXTI1	√	√		
EXTI2	√	√	√	√
EXTI3	√	√	√	√
EXTI4	√	√	√	√
EXTI5	√	√	√	√
EXTI6	√	√	√	√
EXTI7	√	√	√	√
EXTI8	√	√	√	√
EXTI9	√	√	√	√
EXTI10	√	√		
EXTI11	√	√		
EXTI12	√	√		
EXTI13	√	√		
EXTI14	√	√		
EXTI15	√	√		
DMA	√			
BLE_ERROR	√			
BLE_HSLOT	√			
BLE_RX	√			
BLE_TX	√			
BLE_START	√			
BLE_END	√			
BLE_SKIP	√			
BLE_SLP	√	√	√	
BLE_CRYPT	√			
BLE_GROSSTGTIM	√			
BLE_FINETGTIM	√			
BLE_TIMESTAMP_TGT	√			
BLE_SW	√			
BLE_AUDIO0	√			
BLE_AUDIO1	√			
BLE_AUDIO2	√			

表 7-2 低功耗模式下唤醒源

### 7.5.11 低功耗模式设置

低功耗模式主要支持 4 种模式：SLEEP、STOP1、STOP2、STANDBY。在进入低功耗模式时是否可以考虑 BLE 也可以不考虑 BLE，根据是否考虑 BLE，进入和退出低功耗模式的设置方式会有所不同。进入低功耗模式时如果不考虑 BLE，那么唤醒之后蓝牙将无法保

正正常连接态，当不使用用蓝牙时进入低功耗模式时可以不考虑 BLE。具体的配置方式见下文描述。

#### 1) SLEEP 模式，不考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 0，即浅睡模式，使能唤醒源的中断使能位 (NVIC\_ISER\*);
- ◇ 配置唤醒源模块相关控制寄存器，并且使能模块内部的中断使能寄存器;
- ◇ 配置 BLE 相关控制寄存器 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 0，禁止 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后，进入相应的中断函数子程序，清除中断标志位，执行其它相关子程序;

#### 2) SLEEP 模式，考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 0，即浅睡模式，使能唤醒源的中断使能位 (NVIC\_ISER\*);
- ◇ 配置唤醒源模块相关控制寄存器，并且使能模块内部的中断使能寄存器;
- ◇ 配置 BLE 相关控制寄存器：当采用外部唤醒模式时需要将 DEEPSLCNTL->EXTWKUPDSB 配置为 0；当采用 BLE 内部低功耗模块唤醒时，可选择的将 DEEPSLCNTL->EXTWKUPDSB 配置为 1，同时配置唤醒时间寄存器 DEEPSLWKUP 寄存器；上述配置完成后将 DEEPSLCNTL->DEEP\_SLEEP\_ON 配置为 1，使能 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后，进入相应的中断函数子程序，清除中断标志位，执行其它相关子程序；从中断唤醒之后需要将 DEEPSLCNTL->DEEP\_SLEEP\_CORR\_EN 配置为 1 进行 BB 的唤醒后的计算工作，否则下次将无法正常的进入低功耗模式。

#### 3) STOP1 模式，不考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 1，即深度睡眠模式，使能唤醒源的中断使能位 (NVIC\_ISER\*); 将系统低功耗控制寄存器 PMU\_CR->LPM 配置为 2'b00,使能低功耗模式 STOP1;
- ◇ 配置唤醒源模块相关控制寄存器，并且使能模块内部的中断使能寄存器;
- ◇ 配置 DC/DC 转换器的工作模式到 BYPASS 模式;
- ◇ 配置 BLE 相关控制寄存器 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 0，禁止 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后，进入相应的中断函数子程序，清除中断标志位，执行其它相关子程序;

#### 4) STOP1 模式，考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 1，即深度睡眠模式，

使能唤醒源的中断使能位 (NVIC\_IUSER\*); 将系统低功耗控制寄存器 PMU\_CR->LPM 配置为 2'b00,使能低功耗模式 STOP1;

- ◇ 配置唤醒源模块相关控制寄存器, 并且使能模块内部的中断使能寄存器;
- ◇ 配置 DC/DC 转换器的工作模式到 BYPASS 模式;
- ◇ 配置 BLE 相关控制寄存器: 当采用外部唤醒模式时需要将 DEEPSLCNTL->EXTWKUPDSB 配置为 0; 当采用 BLE 内部低功耗模块唤醒时, 可选择的将 DEEPSLCNTL->EXTWKUPDSB 配置为 1, 同时配置唤醒时间寄存器 DEEPSLWKUP 寄存器; 上述配置完成后将 DEEPSLCNTL->DEEP\_SLEEP\_ON 配置为 1, 使能 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后, 进入相应的中断函数子程序, 清除中断标志位, 执行其它相关子程序; 从中断唤醒之后需要将 DEEPSLCNTL->DEEP\_SLEEP\_CORR\_EN 配置为 1 进行 BB 的唤醒后的计算工作, 否则下次将无法正常的进入低功耗模式。

#### 5) STOP2 模式, 不考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 1, 即深度睡眠模式, 使能唤醒源的中断使能位 (NVIC\_IUSER\*); 将系统低功耗控制寄存器 PMU\_CR->LPM 配置为 2'b01,使能低功耗模式 STOP2;
- ◇ 配置唤醒源模块相关控制寄存器, 并且使能模块内部的中断使能寄存器;
- ◇ 配置 DC/DC 转换器的工作模式到 BYPASS 模式;
- ◇ 配置 BLE 相关控制寄存器 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 0, 禁止 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后, 进入相应的中断函数子程序, 清除中断标志位, 执行其它相关子程序;

#### 6) STOP2 模式, 考虑 BLE 进入 Deep Sleep 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 1, 即深度睡眠模式, 使能唤醒源的中断使能位 (NVIC\_IUSER\*); 将系统低功耗控制寄存器 PMU\_CR->LPM 配置为 2'b01,使能低功耗模式 STOP2;
- ◇ 配置唤醒源模块相关控制寄存器, 并且使能模块内部的中断使能寄存器;
- ◇ 配置 DC/DC 转换器的工作模式到 BYPASS 模式;
- ◇ 配置 BLE 相关控制寄存器: 当采用外部唤醒模式时需要将 DEEPSLCNTL->EXTWKUPDSB 配置为 0; 当采用 BLE 内部低功耗模块唤醒时, 可选择的将 DEEPSLCNTL->EXTWKUPDSB 配置为 1, 同时配置唤醒时间寄存器 DEEPSLWKUP 寄存器; 上述配置完成后将 DEEPSLCNTL->DEEP\_SLEEP\_ON 配置为 1, 使能 BLE 进入睡眠模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后, 进入相应的中断函数子程序, 清除中断标志位, 执行其它相关子程序;

从中断唤醒之后需要将 DEEPSLCNTL->DEEP\_SLEEP\_CORR\_EN 配置为 1 进行 BB 的唤醒后的计算工作，否则下次将无法正常的进入低功耗模式。

## 7) STANDBY 模式

- ◇ 配置 Cortex-M3 内核控制寄存器 SCB\_SCR->SLEEPDEEP 为 1, 即深度睡眠模式, 使能唤醒源的中断使能位 (NVIC\_IUSER\*); 将系统低功耗控制寄存器 PMU\_CR->LPM 配置为 2'b10, 使能低功耗模式 STANDBY;
- ◇ 配置唤醒源模块相关控制寄存器, 并且使能模块内部的中断使能寄存器;
- ◇ 配置 DC/DC 转换器的工作模式到 BYPASS 模式;
- ◇ 执行 WFI 进入睡眠模式;
- ◇ 睡眠唤醒后, 系统复位, 从 32'h00000000 地址开始运行

注 1: DEEPSLWKUP 和 DEEPSLCNTL 为 BLE 内部的寄存器, 具体的描述可以参照 BLE 相关章节;

注 2: DEEPSLWKUP->DEEPSLTIME 的设置值必须大于等于如下设置值:

(43+CMU\_CKLDOT[4:0]+CMU\_HOSCT[9:0]+CMU\_PLLT[4:0])

如果不满足要求, 内部电路强制将 DEEPSLWKUP->DEEPSLTIME 的值设置为上述值;

## 7.6 电源管理

芯片内部提供以下两种电源供电方式:

- ◇ 内部 LDO 模式;
- ◇ 内部 DC/DC 转换模式;

### 7.6.1 内部LDO模式

内部 LDO 工作模式时, 不使用内部 DC/DC 转换器, 可以将 DC/DC 转换器旁路, 此时只有内部 LDO 是有效的。如上述“图 6.1 电源连接示意图”所示, DC/DC 处于旁路状态时, DC/DC 转换器的输出管脚 VDD14 输出的电压值为 DC/DC 转换器的输入电压 VDD33 的值。

### 7.6.2 内部DC/DC转换模式

芯片采用降压型 DC/DC 转换, 将降低电源电压 VDD33 的电压值到 1.4V, 内部 LDO 使用降压后的电压 VDD14 给系统供电, 包括芯片内部的所有 LDO。DC/DC 转换器需要外部 LC 滤波, 以及相应的控制使能信号。对于大负载的应用, 使用 DC/DC 转换器可以降低系统的功耗, 在负载较小的应用中不建议采用 DC/DC 转换的工作模式, 避免功耗的浪费。其中 DC/DC 转换器的效率在 85%以上。

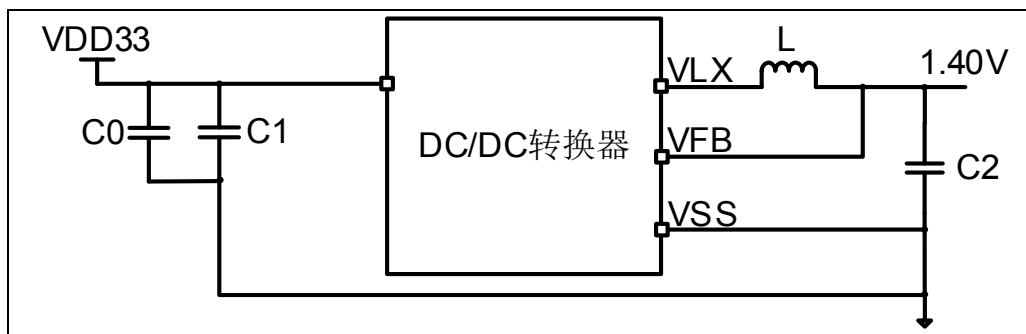


图 7-6 内部 DC/DC 转换器外部连接参考图

注 1: C0 电容值为 0.1uF, C1 电容值为 10uF, C2 电容值为 10uF; L 电感值为 33uH (建议电感 L 封装为绕线电感, 额定电流 300mA 以上);

内部 DC/DC 转换器支持 PWM、BURST、BYPASS、POWERDOWN 四种工作模式, 电源供电方式采用内部 DC/DC 转换模式时, DC/DC 转换器需要配置成 PWM 模式。芯片初始上电时, DC/DC 转换器处于 BYPASS 模式。

Control Bits	MODE			
	PWM	BURST	BYPASS	POWER DOWN
DCBGPD	0	0	1	1
DCBURST	0	1	0	0
DCPWM	1	0	0	1
DCPDC	1	1	0	1
DCBP	1	1	0	0
DCOPD	0	0	x	1
MODE	0	1	x	x

表 7-3 DC/DC 转换器工作模式

注: Control Bits 的具体描述可参照寄存器 BKPC\_PCR 和 PMU\_DCCR 描述;

### 7.6.3 供电模式切换

芯片初始上电时, DC/DC 转换器处于 BYPASS 模式, 即电源供电方式采用内部 LDO 模式。当芯片工作于蓝牙收发状态或其它高功耗的模式时, 从能效的角度考虑, 需要采用内部 DC/DC 转换模式的供电方式, 即需要将 DC/DC 转换器的工作模式从 BYPASS 模式切换到 PWM 模式。当 DC/DC 转换器处于 PWM 的工作模式时, 芯片进入到低功耗模式 STOP1/STOP2/STANDBY 时, 需要通过软件的方式将 DC/DC 转换器切换到 BYPASS 模式, 然后将 PWCTSEN 置 0, 关闭电源控制软件使能位, 将控制权交给内部硬件电路。

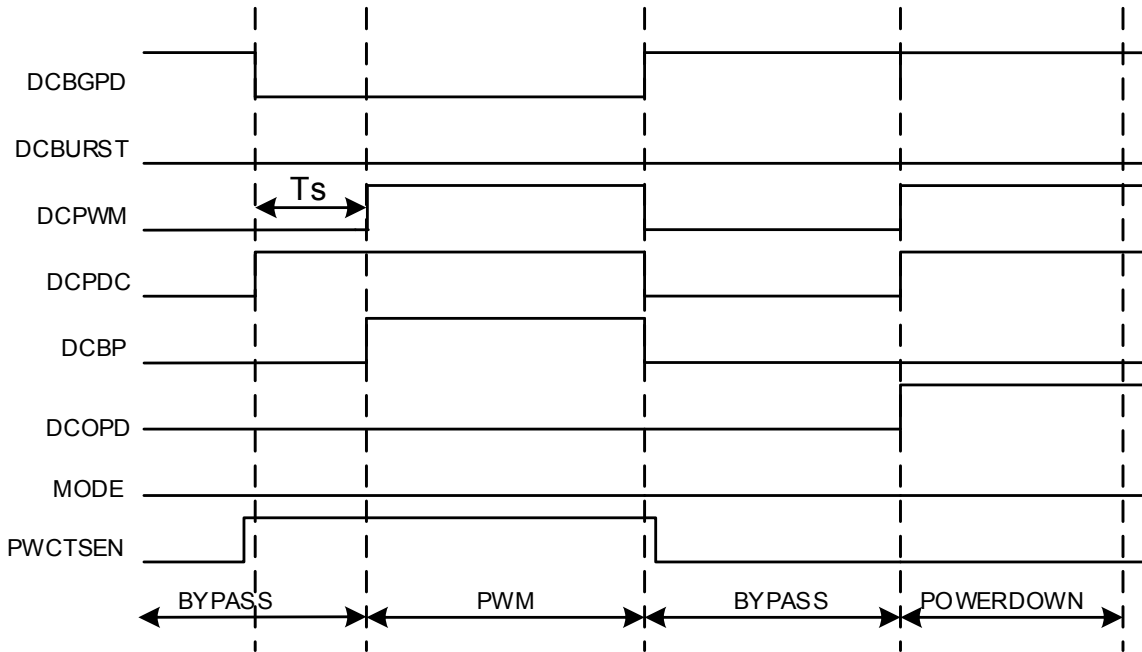


图 7-7 内部 DC/DC 转换器模式切换控制时序图

注 1:  $T_s$  为 DC/DC 内部参考电压稳定时间, 稳定时间的范围要求为: 50us~100us;  
 注 2: 在低功耗模式 STOP1/STOP2/STANDBY 模式, DC/DC 转换器处于 POWERDOWN 工作模式, 实际应用时不建议用软件的方式将 DC/DC 转换器的工作模式切到 POWERDOWN 工作模式;

- ◇ 内部 DC/DC 转换器 BYPASS 工作模式切换到 PWM 工作模式
  - ◇ Step1: 配置寄存器 BKPC->PROT.KEY=0x9669aa55, 去除备份域寄存器写保护;
  - ◇ Step2: 配置寄存器 BKPC\_PCR, 低 16 位控制位的配置值为: BKPC\_PCR<15:0>=16'h001E;
  - ◇ Step3: 配置寄存器 BKPC->PCR.PWCTSEN=1, 关闭内部硬件电路控制模式, 使能软件控制模式;
  - ◇ Step4: 配置寄存器 BKPC->PCR.DCBGPD=0、BKPC->PCR.DCPDC=1, 打开内部参考电压;
  - ◇ Step5: 延时等待 50us~100us, 等待内部参照电压工作稳定;
  - ◇ Step6: 配置寄存器 BKPC->PCR.DCBY=1, 然后配置寄存器 BKPC->PCR.DCPWM=1 进入 PWM 工作模式;
  - ◇ Step7: 配置寄存器 BKPC->PROT.KEY=0x00000000, 使能备份域寄存器写保护;
  
- ◇ 内部 DC/DC 转换器 PWM 工作模式切换到 BYPASS 工作模式
  - ◇ Step1: 配置寄存器 BKPC->PROT.KEY=0x9669aa55, 去除备份域寄存器写保护;
  - ◇ Step2: 配置寄存器 BKPC->PCR.DCBGPD=1、BKPC->PCR.DCPDC=0、BKPC->PCR.DCPWM=0、BKPC->PCR.DCBY=0, BKPC->PCR.LDOERP=1 完成 BYPASS 工作模式配置;
  - ◇ Step3: 配置寄存器 BKPC->PCR.PWCTSEN=0, 使能内部硬件电路控制模式, 关

闭软件控制模式，DC/DC 转换器进入 BYPASS 工作模式；（如果后续所有控制由软件控制，不进入低功耗模式 STOP1/STOP2/STANDBY，可以选择不执行这一步）

- ◇ Step4: 配置寄存器 BKPC->PROT.KEY=0x0000000, 使能备份域寄存器写保护;

## 7.7 特殊功能寄存器

### 7.7.1 寄存器列表

PMU 寄存器列表			
名称	偏移地址	类型	描述
PMU 基地址: 4008_0C00H			
PMU_CR	000 <sub>H</sub>	R/W	PMU 控制寄存器
PMU_SR	004 <sub>H</sub>	R	PMU 状态寄存器
PMU_LVDCR	008 <sub>H</sub>	R/W	LVD 控制寄存器
PMU_PWRCCR	00C <sub>H</sub>	R/W	电源寄存器
PMU_TWUR	010 <sub>H</sub>	R/W	唤醒时间控制寄存器
PMU_VDETCR	014 <sub>H</sub>	R/W	VDET 控制寄存器
PMU_DCCR	018 <sub>H</sub>	R/W	DC-DC 控制寄存器

## 7.7.2 寄存器描述

### 7.7.2.1 PMU控制寄存器 (PMU\_CR)

PMU 控制寄存器 (PMU_CR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000001_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															CKSWEN	Reserved										CSTANDBYF	CWUF	LPM			

Reserved	Bits 31-17	—	保留
CKSWEN	Bit 16	R/W	- (无效)
Reserved	Bits 15-4	—	保留
CSTANDBYF	Bit 3	T_W1	<b>STANDBY 标志清除位</b> 0: 无操作 1: 清除STANDBY标志
CWUF	Bit 2	T_W1	<b>WUF 标志清除位</b> 0: 无操作 1: 清除WUF标志
LPM	Bits 1-0	R/W	<b>低功耗模式选择位 (CPU 进入 Deepsleep 时有效)</b> 00: STOP1 01: STOP2 1x: STANDBY 注: 系统工作时钟选择为低速时钟时(SYS_CMD=3'b110) 无法进入STOP1/STOP2低功耗模式



### 7.7.2.2 PMU状态寄存器 (PMU\_SR)

PMU 状态寄存器 (PMU_SR)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													STANDBYF	WUF	

Reserved	Bits 31-2	—	保留
STANDBYF	Bit 1	R	<b>STANDBY 标志位</b> 0: 芯片未进入 STANDBY 模式 1: 芯片复位之前已进入STANDBY模式 注: 该位通过CSTANDBYF位来清零
WUF	Bit 0	R	<b>唤醒标志位</b> 0: 未发生唤醒事件 1: 使能 注: 该位通过 CWUF 位来清零

### 7.7.2.3 LVD控制寄存器 (PMU\_LVDCR)

LVD 控制寄存器 (PMU_LVDCR)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															
<div style="display: flex; justify-content: space-between;"> <span>LVDO</span> <span>Reserved</span> <span>LVDFLT</span> <span>LVIFS</span> <span>LVDS</span> <span>LVDCFIF</span> <span>LVDIF</span> <span>LVDIE</span> <span>LV DEN</span> </div>																															

Reserved	Bits 31-16	-	保留
LVDO	Bit 15	R	<b>LVD 状态标志位</b> 0: 小于阈值 1: 大于阈值
Reserved	Bits 14-12	—	保留
LVDFLT	Bit 11	R/W	<b>LVD 滤波使能位</b> 0: 禁止 1: 使能 注: 使能后 LVDO 稳定时间小于 100us 的变化将被忽略
LVIFS	Bits 10-8	R/W	<b>LVD 中断标志产生模式选择位</b> 000: LVDO 上升沿产生中断 001: LVDO 下降沿产生中断 010: LVDO 高电平产生中断 011: LVDO 低电平产生中断 1xx: LVDO 变化 (上升或下降沿) 产生中断
LVDS	Bits 7-4	R/W	<b>LVD 电压阈值选择位</b> 0000: 1.8V 0001: 1.9V 0010: 2.0V 0011: 2.1V 0100: 2.2V 0101: 2.3V 0110: 2.4V 0111: 2.5V 1000: 2.6V 1001: 2.7V 1010: 2.8V 1011: 2.9V 1100: 3.0V 1101: 1.8V 1110: 1.8V

			1111: 1.8V 其他: 保留
LVDCIF	Bit 3	T_W1	<b>LVD 中断标志清除位</b> 0: 无操作 1: 清除LVD中断标志
LVDIF	Bit 2	R	<b>LVD 中断标志位</b> 0: LVDO 状态未变化 1: LVDO状态发生变化 注: 该位由LVDCIF清除
LVDIE	Bit 1	R/W	<b>LVD 中断使能位</b> 0: 禁止 1: 使能
LVDEN	Bit 0	R/W	<b>LVD 使能位</b> 0: 禁止 1: 使能

#### 7.7.2.4 电源控制寄存器 (PMU\_PWRCR)

电源控制寄存器 (PMU_PWRCR)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_0000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LDO1P1S	SRAMPDS	Reserved										BBSRAMPD	SRAMPD	Reserved	LDO1PIEN	IBIASEN	

Reserved	Bits 31-18	—	保留
LDO1P1S	Bit17	R	<b>1.1V LDO 工作状态指示位 (BT50_AFE 内部 LDO)</b> 0: 未工作或处于非稳定状态 1: 处于稳定工作状态 注: 此控制位无效
SRAMPDS	Bit16	R	<b>系统存储器 SRAM1 工作状态指示位</b> 0: 正常模式 1: 掉电模式 当 SRAMPD 控制位设置完成后, 需要通过读取 SRAMPDS 状态位去判断 SRAM1 所处的工作状态之后进行 SRAM1 读写访问
Reserved	Bits 15-6	—	保留
BBSRAMPD	Bit 5	R/W	<b>BB Exchange Memory1(8KB SRAM)掉电使能位</b>

			0: 正常模式 1: 掉电模式 注: 仅在低功耗模式STOP1/STOP2模式有效
SRAMPD	Bit 4	R/W	系统存储器SRAM1掉电使能位 0: 正常模式 1: 掉电模式
Reserved	Bits 3-2	—	保留
LDO1P1EN	Bit 1	R/W	1.1V LDO 使能位 (BT50_AFE 内部 LDO) 0: 禁止 1: 使能 注: 此控制位无效
IBIASEN	Bit 0	R/W	偏置电流控制使能位 0: 禁止 1: 使能 注: 为内部ADC、LRC128K、Touchkey提供基准或参考电流。

### 7.7.2.5 唤醒时间控制寄存器 (PMU\_TWUR)

电源控制寄存器 (PMU_TWUR)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											WKT																				

Reserved	Bits 31-12	—	保留
WKT	Bits 11-0	R/W	唤醒时间控制位 唤醒时间=Tsyclk*WKT

### 7.7.2.6 VDET控制寄存器 (PMU\_VDETCR)

电源电压检测控制寄存器 (PMU_VDETCR)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00010000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SEL			Reserved			EN	

Reserved	Bits 31-8	—	保留
SEL	Bit 7-4	R/W	<b>VDET Divided voltage select control</b> 0001: VOUT=1/6 VIN 0010: VOUT=1/3 VIN 0100: VOUT=1/2 VIN 1000: VOUT=2/3 VIN EN=0,SEL[3:0]=0000 VOUT=Z EN=0,SEL[3:0]=~0000 VOUT=VIN
Reserved	Bits 3-1	—	保留
EN	Bit 0	R/W	<b>VDET 使能位</b> 0: 禁止 1: 使能

### 7.7.2.7 DCDC控制寄存器 (PMU\_DCCR)

DCDC 控制寄存器 (PMU_DCCR)																																		
偏移地址: 18 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved								VCTL				Reserved	FI				Reserved	HYSCTL				Reserved	FC				Reserved	OVCCEN	PUPWM	MODE	PUBURST	DELAY	PDDC	Reserved

Reserved	Bits 31-24	—	保留												
VCTL	Bits 23-20	R/W	<b>DC-DC输出电压控制位 (单位: V)</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>VCTL[3:0]</th> <th>VREF09</th> <th>DC-DC 输出</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>0.903</td> <td>1.405</td> </tr> <tr> <td>0010</td> <td>0.941</td> <td>1.460</td> </tr> <tr> <td>0011</td> <td>0.954</td> <td>1.480</td> </tr> </tbody> </table>	VCTL[3:0]	VREF09	DC-DC 输出	0000	0.903	1.405	0010	0.941	1.460	0011	0.954	1.480
VCTL[3:0]	VREF09	DC-DC 输出													
0000	0.903	1.405													
0010	0.941	1.460													
0011	0.954	1.480													

			0100	0.984	1.530
			0110	0.984	1.530
			0111	0.993	1.545
			1000	1.030	1.600
			1010	1.030	1.600
			1100	1.030	1.600
			1110	1.030	1.600
			0101	1.000	1.556
			1011	1.040	1.620
			1111	1.040	1.620
			1101	1.050	1.633
			1001	1.050	1.633
Reserved	Bit 19	—	保留		
FI	Bit 18-16	R/W	PWM模式OSC频率控制电流控制位 注：参见FC控制位的描述		
Reserved	Bit 15	—	保留		
HYSCTL	Bits 14-12	R/W	BURST Hys控制 000: no hys ..... 111: big hys 注：应用时此控制位的建议配置值为3'b111		
Reserved	Bit 11	—	保留		
FC	Bits 10-8	R/W	PWM模式OSC频率控制电容控制位		
			FC[2:0]	FI[2:0]	频率 (MHz)
			000	000	2.12
			.....	.....	.....
			100	010	1.72
			111	111	5.58
			注：应用时建议采用如下配置： FC[2:0]: 100 FI[2:0] : 010		
Reserved	Bit 7	—	保留		
OVCCEN	Bit 6	R/W	过流检测电路使能控制位 0: 禁止 1: 使能 注：应用时保持此控制位为0，不建议设置成1。		
PUPWM	Bit 5	R/W	注：此控制位无效，有效控制位参见BKPC_PCR[9]描述		
MODE	Bit 4	R/W	工作模式选择控制位 0: PWM 1: BURST		
PUBURST	Bit 3	R/W	注：此控制位无效，有效控制位参见		

			BKPC_PCR[8]描述
DELAY	Bit 2	R/W	<b>Nonoverlap CLK 控制位</b> 0: narrow 1: wide
PDDC	Bit 1	R/W	注：此控制位无效，有效控制位参见BKPC_PCR[10]描述
Reserved	Bit 0	—	保留

## 第8章 备份电源域控制（BKPC）

### 8.1 概述

本模块控制备份电源域（如 ULRC、LOSC、LOSM、RTC 等）的工作状态，备份域供电电压的选择，standby 模式下唤醒源的选择，STOP2 模式下部分模块掉电控制等。通过对寄存器的配置，可以达到在功耗和可靠性上对备份域的工作状态进行灵活控制与选择。

### 8.2 特性

- ◇ 受保护的寄存器访问，防止误操作
- ◇ Standby 模式下唤醒源可选择
- ◇ RTC 工作时钟可选择

### 8.3 备份域电源管理

在系统低功耗工作模式中，通过将主域电源关闭以最大化节约功耗；在备份域 RTC 计时或 GPIO 管脚外部触发，采用动态打开其电路的工作电源方式，进一步降低备份域的功耗。

在系统低功耗电源控制中，复位初始是处于上电状态 POWERED\_UP，当工作模式切换至低功耗模式中时，进入掉电准备状态 POWERING\_DOWN，随后进入掉电状态 POWERED\_DOWN，同时将主域电源切掉，并一直保持该状态，直至接收到唤醒信号，进入上电准备状态 POWERING\_UP，最终再回到上电状态 POWERED\_UP。

### 8.4 特殊功能寄存器

#### 8.4.1 寄存器列表

BKPC 寄存器列表			
名称	偏移地址	类型	描述
BKPC 基地址：4004_9000H			
BKPC_PROT	000 <sub>H</sub>	R/W	BKPC 保护控制寄存器
BKPC_WKR	004 <sub>H</sub>	R/W	BKPC STANDBY 唤醒控制寄存器
BKPC_WKSR	008 <sub>H</sub>	R	BKPC STANDBY 唤醒状态寄存器
BKPC_PCCR	00C <sub>H</sub>	R/W	BKPC 时钟选择寄存器
BKPC_PCR	010 <sub>H</sub>	R/W	BKPC 电源控制寄存器
-	014 <sub>H</sub>	-	-
-	018 <sub>H</sub>	-	-



### 8.4.1.1 备份电源域写保护寄存器 (BKPC\_PROT)

备份电源域写保护寄存器 (BKPC_PROT)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[31:1]																															PROT /KEY[0]

KEY	Bits 31-1	W	<b>保护关键码</b> 0x9669aa55: 去除写保护 其他: 开启写保护
PROT	Bit 0	R	<b>保护状态位</b> 0: 无写保护 1: 写保护

### 8.4.1.2 STANDBY唤醒控制寄存器 (BKPC\_WKR)

STANDBY 唤醒控制寄存器 (BKPC_WKR)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000001_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKSLR	Reserved					WKEG										Reserved					WKEN										

WKSLR	Bit 31	W1	<b>STANDBY 模式唤醒源指示位清零</b> 0: 无操作 1: 清零唤醒源指示位
Reserved	Bits 30-26	-	保留
WKEG	Bit 25-16	R/W	<b>STANDBY 唤醒源有效沿控制位</b> 0: 上升沿平唤醒 1: 下降沿唤醒 Bit 7-0: PA9~PA2 Bit 8 : 保留 Bit 9 : RTC <i>注: RTC 固定为上升沿唤醒</i>

Reseverd	Bits 15-10	-	保留
WKEN	Bit 9-0	R/W	<b>STANDBY 唤醒源使能位</b> 0: 禁止 1: 使能 Bit 7-0: PA9~PA2 Bit 8 : 保留 Bit 9 : RTC

### 8.4.1.3 STANDBY唤醒状态寄存器 (BKPC\_WKSR)

STANDBY 唤醒状态寄存器 (BKPC_WKSR)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WKF															

Reserved	Bits 31-10	-	保留
WKF	Bit 9-0	R	<b>STANDBY 唤醒事件标志位</b> 0: 无唤醒事件 1: 产生唤醒事件 Bit 7-0: PA9~PA2 Bit 8 : 保留 Bit 9 : RTC 通过 BKPC_WKR-> WKSLCLR 位清零

### 8.4.1.4 备份电源域时钟控制寄存器 (BKPC\_PCCR)

备份电源域时钟控制寄存器 (BKPC_PCCR)																																					
偏移地址: 0C <sub>H</sub>																																					
复位值: 00000000_00000001_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved																RTCKS	Reserved				ULRCEN	LOSCPS	LOSCPM	LOSMEN	LOSCEN	Reseverd										RTCKEN	RTCK

Reserved	Bits 31-17	-	保留
----------	------------	---	----

RTCKS	Bit 16	R	<b>RTC 模块时钟状态标志位</b> 0: LOSM 1: ULRC
Reserved	Bits 15-13	-	保留
ULRCEN	Bit 12	R/W	<b>ULRC 内部低速时钟软件使能位</b> 0: 禁止 1: 使能
LOSCPS	Bit 11	R/W	<b>LOSC 外部低速晶振 PMOS 二极管开关选择位</b> 1: 关闭 0: 打开
LOSCPM	Bit 10	R/W	<b>LOSC 外部低速晶振电路高功耗模式控制位</b> 0: 关闭, 内部 LDO 供电 1: 打开, VDD3V 供电
LOSMEN	Bit 9	R/W	<b>LOSC 安全管理使能位</b> 0: 禁止 1: 使能
LOSCEN	Bit 8	R/W	<b>LOSC 外部低速晶振软件使能位</b> 0: 禁止 1: 使能
Reseverd	Bits 7-2	-	保留
RTCKEN	Bit 1	R/W	<b>RTC 模块时钟源使能位</b> 0: 禁止 1: 使能
RTCK	Bit 0	R/W	<b>RTC 模块和备份域主状态机时钟源选择位</b> 0: ULRC 1: LOSM <i>注: 此控制位也表示备份电源域的系统主时钟</i>

注:

1、STANDBY 低功耗模式 LOSCEN 和 ULRCEN 为低电平时, 备份电源域无时钟, 此时支持 GPIO/MRST 唤醒;

### 8.4.1.5 备份电源域电源控制寄存器 (BKPC\_PCR)

备份电源域电源控制寄存器 (BKPC_PCR)																																		
偏移地址: 10 <sub>H</sub>																																		
复位值: 00000000_00010011_00000000_00001010 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved								TSUDC				TSULDO				Reserved								DCPDC	DCPWM	DCBURST	DCOPD	DCBY	AFEPD	DCBGPD	LDOSEL	LDOERPD	HPLDOEN	PWCTSEN

Reserved	Bits 31-24	-	保留
TSUDC	Bits 23-20	R/W	<b>DC-DC 稳定时间设置</b> T <sub>sudc</sub> =TSUDC * T <sub>ulrc</sub> 注: 由于硬件状态机, 对 DC/DC 切换时只在 POWERDOWN 和 BYPASS 模式之间切换, 此控制位无实际作用, 建议配置为 0, 减小唤醒时间。
TSULDO	Bits 19-16	R/W	<b>LDO 稳定时间设置</b> T <sub>suldo</sub> =TSULDO * T <sub>ulrc</sub>
Reserved	Bits 15-11	-	保留
DCPDC	Bit 10	R/W	<b>DC-DC 软件控制使能位</b> 0: 禁止 1: 使能
DCPWM	Bit 9	R/W	<b>PWM 模式软件控制使能位</b> 0: 禁止 1: 使能
DCBURST	Bit 8	R/W	<b>BURST 模式软件控制使能位</b> 0: 使能 1: 禁止
DCOPD	Bit 7	R/W	<b>DC-DC 输出软件控制使能位</b> 0: 使能 1: 禁止 (DC-DC 输出电压为 0V)
DCBY	Bit 6	R/W	<b>DC-DC bypass 软件控制使能位</b> 0: 输出 3.30v 1: 输出 1.40v
AFEPD	Bit 5	R/W	<b>BLE_AFE 软件控制使能位</b> 0: 上电 1: 掉电
DCBGPD	Bit 4	R/W	<b>DC-DC BandGap 软件控制使能位</b> 0: 使能 1: 禁止
LDOSEL	Bit 3	R/W	<b>高低功耗 LDO 软件控制选择位</b>

			0: LPLDO 1: HPLDO
LDOERPD	Bit 2	R/W	低功耗 LDO 外部参考电压控制开关使能位 0: 打开 1: 关闭 注: 芯片上电完成后需要软件将此控制位配置成“1”, 即关闭低功耗 LDO 的外部参考电压开关, 参考电压有 LDO 内部自身提供。
HPLDOEN	Bit 1	R/W	高功耗 LDO 软件控制使能位 0: 禁止 1: 使能
PWCTSEN	Bit 0	R/W	电源控制软件使能位 0: 禁止 (硬件电路控制模式) 1: 使能

## 第9章 复位管理(RMU)

### 9.1 概述

系统复位可以由下面列出的任一事件触发。这些复位事件标志可以通过读取 RMU\_RSTSR 寄存器来判断复位源。

### 9.2 特性

- ◇ 备份域支持 POR
- ◇ 主电源域支持 POR/BOR
- ◇ 支持外部端口复位 MRSTN
- ◇ 支持看门狗溢出复位
- ◇ 内核锁死 (LOCKUP) 复位
- ◇ 读取配置字错误复位 (CFG\_RST)
- ◇ 支持三种软件复位
  - ◇ 复位整个主电源域
  - ◇ 复位除启动配置外的整个主电源域
  - ◇ 复位内核
- ◇ 支持各外设模块的单独复位

### 9.3 结构框图

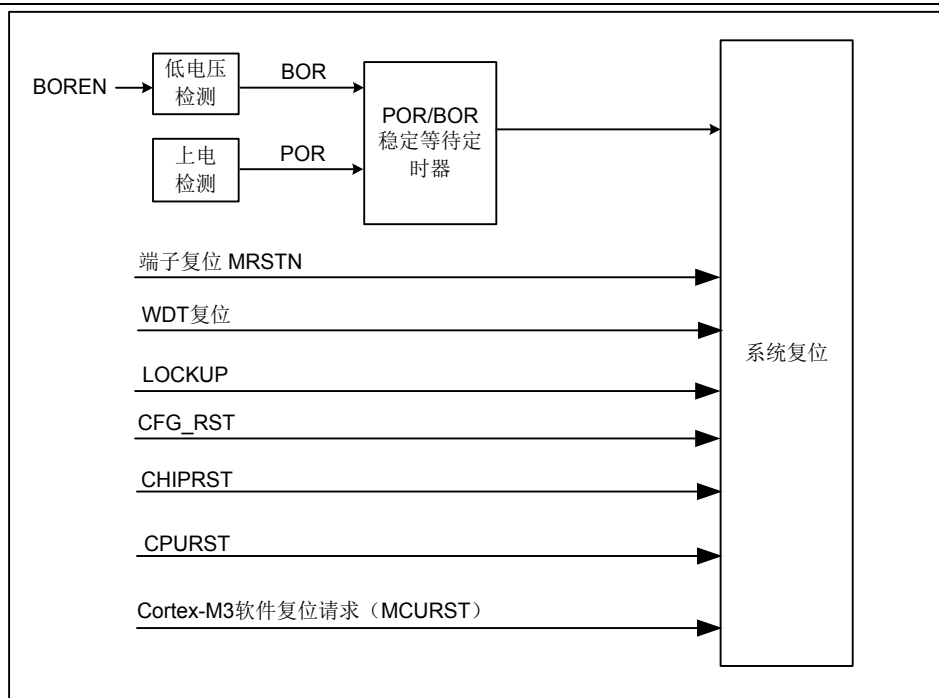


图 9-1 复位结构图

## 9.4 系统复位

ES32W3120 系列微控制器支持 8 种复位源。CPURST 只复位 CPU 内核(不包含调试部分)；其他复位源则复位 CPU 内核和所有外设。各个复位源及寄存器关系如下表所示：

RMU_RSTSR	POR	BOR
	POR=1 WAKEUP=1	BOR=1
BOR33EN (RMU_CSR[0])	0x1	0x1
BOR09EN (RMU_CSR[8])	0x1	0x1
LV DEN (PMU_LVD CR[0])	0x0	0x0
BLD (SYSCFG_MEMRMP[0])	0x0	0x0
BTF (SYSCFG_MEMRMP[8])	0x0	0x0
SYS_STU (CMU_CSR[10:8])	0x1	0x1
CFT_STU (CMU_CSR[24])	0x0	0x0
LSCK_STU (CMU_CSR[31:30])	0x1	0x1
HRCFSW (CMU_CFGR[24])	0x0	0x0
系统和外设时钟分频	1 分频	1 分频
HOSC_EN (CMU_CLKENR[0])	0x0	0x0
LOSCEN (BKPC_PCCR[8])	0x0	-
HRCEN (CMU_CLKENR[2])	0x1	0x1
LRCEN (CMU_CLKENR[3])	0x1	0x1
ULRCEN (BKPC_PCCR[12])	0x1	-
PLL1EN (CMU_CLKENR[8])	0x0	0x0
PLL1EN_96M (CMU_CLKENR[9])	0x0	0x0
CMU_AHB1ENR	0xFFFF	0xFFFF
CMU_APB1ENR	0xFFFF_FFFF	0xFFFF_FFFF
CMU_APB2ENR	0xFFFF_FFFF	0xFFFF_FFFF
CMU_BLEENR	0xF	0xF
CMU_BLECFG	0x3	0x3
CMU_BLECKS	6'd12	6'd12
CMU_PLLCFG	0x8101_0430	0x8101_0430
CPU 内核调试模块	复位值	复位值
备份域寄存器	备份域电源上电复位可复位； 主电源域上电复位不影响备份域	-
其他外设	复位值	

表 9-1 POR/BOR 复位与寄存器关系

	MRSTn	WDT
RMU_RSTSR	NMRST=1	WWDT=1 或 IWDT=1
BOR33EN (RMU_CSR[0])	0x1	0x1
BOR09EN (RMU_CSR[8])	0x1	0x1
LVDEN (PMU_LVDCR[0])	0x0	0x0
BLD (SYSCFG_MEMRMP[0])	0x0	-
BTF (SYSCFG_MEMRMP[8])	0x0	0x0
SYS_STU (CMU_CSR[10:8])	0x1	0x1
CFT_STU (CMU_CSR[24])	0x0	0x0
LSCK_STU (CMU_CSR[31:30])	0x1	0x1
HRCFSW (CMU_CFGR[24])	0x0	0x0
系统和外设时钟分频	1 分频	1 分频
HOSC_EN (CMU_CLKENR[0])	0x0	0x0
LOSCEN (BKPC_PCCR[8])	-	-
HRCEN (CMU_CLKENR[2])	0x1	0x1
LRCEN (CMU_CLKENR[3])	0x1	0x1
ULRCEN (BKPC_PCCR[12])	-	-
PLL1EN (CMU_CLKENR[8])	0x0	0x0
PLL1EN_96M (CMU_CLKENR[9])	0x0	0x0
CMU_AHB1ENR	0xFFFF	0xFFFF
CMU_APB1ENR	0xFFFF_FFFF	0xFFFF_FFFF
CMU_APB2ENR	0xFFFF_FFFF	0xFFFF_FFFF
CMU_BLEENR	0xF	0xF
CMU_BLECFG	0x3	0x3
CMU_BLECKS	6'd12	6'd12
CMU_PLLCFG	0x8101_0430	0x8101_0430
CPU 内核调试模块	-	-
备份域寄存器	复位值	-
其他外设	复位值	

表 9-2 MRSTn/WDT 复位与寄存器关系

RMU_RSTSR	LOCKUP	CHIPRST	SYSRSTREQ	CPURST
	LOCKUP=1	CHIP=1	MCU=1	CPU=1
BOR33EN (RMU_CSR[0])	0x1	0x1	0x1	-
BOR09EN (RMU_CSR[8])	0x1	0x1	0x1	-
LVDEN (PMU_LVDCR[0])	0x0	0x0	0x0	-
BLD (SYSCFG_MEMRMP[0])	-	0x0	-	-
BTF (SYSCFG_MEMRMP[8])	0x0	0x0	0x0	-
SYS_STU (CMU_CSR[10:8])	0x1	0x1	-	-
CFT_STU (CMU_CSR[24])	0x0	0x0	-	-



	LOCKUP	CHIPRST	SYSRSTREQ	CPURST
LCK_STU (CMU_CSR[31:30])	0x1	0x1	-	-
HRCFSW (CMU_CFGR[24])	0x0	0x0	-	-
系统和外设时钟分频	-	1 分频	-	-
HOSC_EN (CMU_CLKENR[0])	0x0	0x0	0x0	-
LOSCEN (BKPC_PCCR[8])	-	-	-	-
HRCEN (CMU_CLKENR[2])	0x1	0x1		
LRCEN (CMU_CLKENR[3])	0x1	0x1	-	-
ULRCEN (BKPC_PCCR[12])	-	-	-	-
PLL1EN (CMU_CLKENR[8])	0x0	0x0	0x0	-
PLL1EN_96M (CMU_CLKENR[9])	0x0	0x0	0x0	-
CMU_AHB1ENR	0xFFFF	0xFFFF	0xFFFF	-
CMU_APB1ENR	0xFFFF_FFFF	0xFFFF_FFFF	0xFFFF_FFFF	-
CMU_APB2ENR	0xFFFF_FFFF	0xFFFF_FFFF	0xFFFF_FFFF	-
CMU_BLEENR	0xF	0xF	0xF	-
CMU_BLECFG	0x3	0x3	0x3	-
CMU_BLECKS	6'd12	6'd12	6'd12	-
CMU_PLLCFG	0x8101_0430	0x8101_0430	0x8101_0430	-
CPU 内核调试模块	-	复位值	-	-
备份域寄存器	-	-	-	-
其他外设		复位值		-

表 9-3 系统复位与寄存器关系

### 9.4.1 硬件复位

硬件复位包括上电复位，欠压复位，外部端口复位，LOCKUP 复位，WDT 复位和读取配置字错误复位。

#### 9.4.1.1 上电复位

芯片内部集成 POR 产生电路，分别用于主电源域和备份电源域。备份域的 POR 通常监测的是电池供电电压(VBAT)，若无电池供电，则备份域 POR 与主电源域一致。

当 VDD/VDDA 低于指定阈值 VPOR/VPDR 时，器件无需外部复位电路便会保持复位状态。

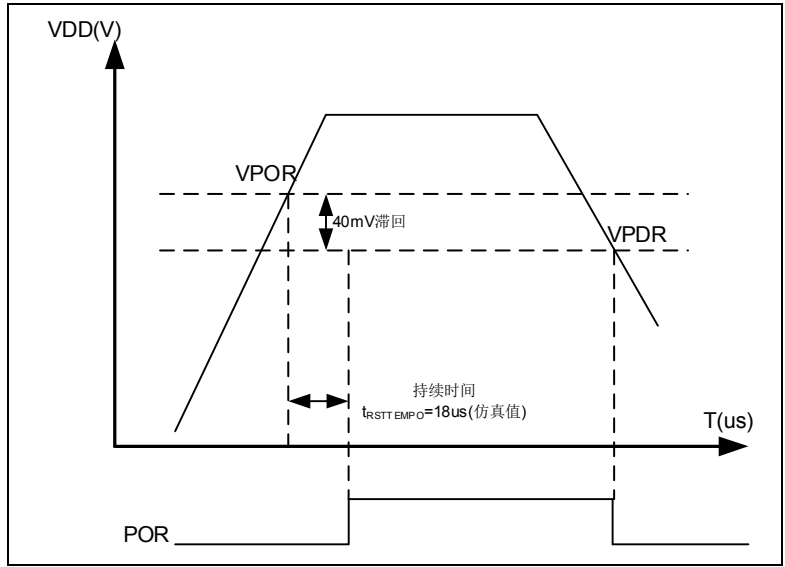


图 9-2 上电复位示意图

#### 9.4.1.2 欠压复位

上电期间，欠压复位（BOR）将使器件保持复位状态，直到电源电压达到 1.6V。芯片默认 BOR 为开启状态，复位完成后，可通过软件选择 BOR 复位电压阈值 VBOR，或可将 BOR 禁止。芯片支持 16 个 VBOR 阈值选择。

当电源电压（VDD）降至所选 VBOR 阈值以下时，将使器件复位。

通过设置芯片配置字也可以禁止 BOR。当上电期间电源电压达到 1.6V 后，BOR 自动被禁止。

BOR 阈值滞回电压约为 100 mV（电源电压的上升沿与下降沿之间）。

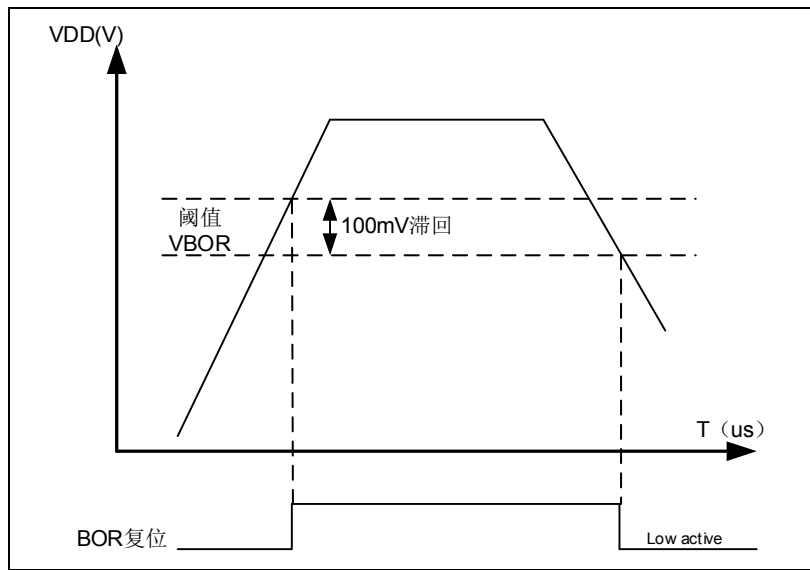


图 9-3 低电压复位示意图

### 9.4.1.3 端口复位

可复位除内核调试模块以外的芯片整体，复位解除后，芯片从 BOOT ROM 启动。

### 9.4.1.4 看门狗复位

详细描述请参照[独立看门狗](#)和[窗口看门狗](#)的说明。

可复位除内核调试模块以外的芯片整体，复位解除后，芯片正常从 FLASH 启动。

### 9.4.1.5 LOCKUP复位

由不可恢复的异常导致的内核锁死，此时将产生复位信号来重新启动内核及系统。详细说明可参考 ARM 技术手册。

### 9.4.1.6 读取配置字错误复位

在配置字加载时，如果加载的配置字错误（配置字高低 16 位不满足取反逻辑），配置字加载时会加载系统默认值而不是用户设置值，同时将 RMU\_RSTSR[16]置高。当系统程序运行过程中产生了看门狗复位请求时（IWDT/WWDT），系统产生复位会重新加载配置字。如果未产生配置字加载错误，那么当产生看门狗复位时，系统不会重新加载配置字。

## 9.4.2 软件复位

---

### 9.4.2.1 芯片复位(CHIPRST)

芯片复位由寄存器 RMU\_AHB2RSTR 的 CHIPRST 控制，可复位整体芯片。复位后芯片从 FLASH 空间启动。

### 9.4.2.2 CPU复位(CPURST)

CPU 复位由寄存器 RMU\_AHB2RSTR 的 CPURST 控制，可复位内核（不包含调试部分）。

### 9.4.2.3 MCU复位(MCURST)

MCU 复位从内核产生。由应用中断和复位控制寄存器(Application Interrupt and Reset Control Register)的 SYSRESETREQ 位控制，将该位置 1 可对系统复位。详细可参考 ARM 相关技术手册。

### 9.4.2.4 外设软件复位

对应每个外设分别分配了一个软件复位。

AHB1 外设复位寄存器（RMU\_AHB1RSTR）为 GPIO,CRC,CALC,CRYPT,TRNG,PIS 等模块的提供软件复位。

AHB2 外设复位寄存器（RMU\_AHB2RSTR）为 FSMC 等提供软件复位。CPU 复位和芯片复位的控制位也在该寄存器中。

APB1 外设复位寄存器（RMU\_APB1RSTR）为 TIM0~3, UART0~2, SPI0~2, I2C0~1 等 APB1 模块提供软件复位。

APB2 外设复位寄存器（RMU\_APB2RSTR）为 LP16T, ADC0, WWDT, IWDT, 以及备份域相关寄存器等提供软件复位。

APB3 外设复位寄存器 (RMU\_APB3RSTR) 为 BLE5.0 MDM 提供软件复位。

AHB2 BLE 复位寄存器 (RMU\_AHB2BLERST) 为 BLE5.0 BB 提供软件复位。

## 9.5 特殊功能寄存器

### 9.5.1 寄存器列表

RMU 寄存器列表			
名称	偏移地址	类型	描述
RMU 基地址: 4008_0800 <sub>H</sub>			
RMU_CSR	000 <sub>H</sub>	R/W	RMU 控制状态寄存器
RMU_RSTSR	010 <sub>H</sub>	R	RMU 复位状态寄存器
RMU_CRSTSR	014 <sub>H</sub>	W	RMU 清复位状态寄存器
RMU_AHB1RSTR	020 <sub>H</sub>	T_W1	AHB1 外设复位寄存器
RMU_AHB2RSTR	024 <sub>H</sub>	T_W1	AHB2 外设复位寄存器
RMU_APB1RSTR	030 <sub>H</sub>	T_W1	APB1 外设复位寄存器
RMU_APB2RSTR	034 <sub>H</sub>	T_W1	APB2 外设复位寄存器
RMU_APB3RSTR	038 <sub>H</sub>	T_W1	APB3 外设复位寄存器
RMU_AHB2BLERSTR	03C <sub>H</sub>	T_W1	AHB2 BLE 复位寄存器

## 9.5.2 寄存器描述

### 9.5.2.1 RMU控制状态寄存器 (RMU\_CSR)

RMU 控制状态寄存器 (RMU_CSR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000001_00001111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BOR09VS		Reserved		BOR09EN	BOR33VS			BORFLT		BOR33EN					

Reserved	Bits 31-16	—	保留
BOR09VS	Bits 15-12	R/W	<b>BOR11 电压点选择位</b> 0000: 0.55V 0001: 0.55V 0010: 0.35V 0011: 0.45V 0100: 0.55V 0101: 0.65V 0110: 0.75V 0111: 0.55V 1xxx: 保留
Reserved	Bits 11-9	—	保留
BOR09EN	Bit 8	R/W	<b>BOR11 使能位</b> 0: 禁止 1: 使能
BOR33VS	Bits 7-4	R/W	<b>BOR33 电压点选择位</b> 0000: 1.3V 0001: 1.4V 0010: 1.5V 0011: 1.6V 0100: 1.7V 0101: 2.0V 0110: 2.5V 0111: 2.7V 1xxx: 保留
BORFLT	Bits 3-1	R/W	<b>BOR 滤波时钟选择位</b> 00x: 1 个 ULRC 周期 010: 2 个 ULRC 周期 011: 3 个 ULRC 周期 100: 4 个 ULRC 周期 101: 5 个 ULRC 周期

			110: 6个ULRC周期 111: 7个ULRC周期 注: 对BOR33和BOR09同时有效
BOR33EN	Bit 0	R/W	<b>BOR33 使能位</b> 0: 禁止 1: 使能

### 9.5.2.2 RMU复位状态寄存器 (RMU\_RSTSR)

RMU 复位状态寄存器 (RMU_RSTSR)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000100_00000011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CFGERR	Reserved				CFG	CPU	MCU	CHIP	LOCKUP	WWDT	IWDT	NMRST	BOR	WAKEUP	POR

Reserved	Bits 31-17	—	保留
CFGERR	Bit 16	R	<b>配置字错误状态标志位</b> 0: 无配置字错误 1: 产生配置字错误 注: 当程序配置字加载错误时, 软件可以触发看门狗复位或芯片全局复位来重新加载配置字
Reserved	Bits 15-11	—	保留
CFG_WORD	Bit 10	R	<b>配置字复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
CPU	Bit 9	R	<b>软件 CPU 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
MCU	Bit 8	R	<b>软件 MCU 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生 注: 该复位从内核产生。由应用中断和复位控制寄存器(Application Interrupt and Reset Control Register)的SYSRESETREQ位控制, 将SYSRESETREQ位置1可对系统复位。详细可参考ARM相关技术手册。
CHIP	Bit 7	R	<b>软件 CHIP 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
LOCKUP	Bit 6	R	<b>Lockup 复位状态标志位</b> 0: 无复位发生或标志位已被清除

			1: 有复位发生
WWDTF	Bit 5	R	<b>WWDT 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
IWDTF	Bit 4	R	<b>IWDT 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
NMRST	Bit 3	R	<b>NMRST 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
BOR	Bit 2	R	<b>BOR 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
WAKEUP	Bit 1	R	<b>唤醒复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生
POR	Bit 0	R	<b>POR 复位状态标志位</b> 0: 无复位发生或标志位已被清除 1: 有复位发生

9.5.2.3 RMU清复位状态寄存器 (RMU\_CRSTSR)

RMU 清复位状态寄存器 (RMU_CRSTSR)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					CFG_WORD	CPU	MCU	CHIP	LOCKUP	WWDTC	IWDTC	NMRST	BOR	WAKEUP	POR

Reserved	Bits 31-11	—	保留
CFG_WORD	Bit 10	W	配置字复位标志清除位 0: 无操作 1: 清除标志
CPU	Bit 9	W	软件 CPU 复位标志清除位 0: 无操作 1: 清除标志
MCU	Bit 8	W	软件 MCU 复位标志清除位 0: 无操作 1: 清除标志
CHIP	Bit 7	W	软件 CHIP 复位标志清除位 0: 无操作 1: 清除标志
LOCKUP	Bit 6	W	LOCKUP 复位标志清除位 0: 无操作 1: 清除标志
WWDTC	Bit 5	W	WWDTC 复位标志清除位 0: 无操作 1: 清除标志
IWDTC	Bit 4	W	IWDTC 复位标志清除位 0: 无操作 1: 清除标志
NMRST	Bit 3	W	NMRST 复位标志清除位 0: 无操作 1: 清除标志
BOR	Bit 2	W	BOR 复位标志清除位 0: 无操作 1: 清除标志
WAKEUP	Bit 1	W	唤醒复位标志清除位 0: 无操作 1: 清除标志
POR	Bit 0	W	POR 复位标志清除位



			0: 无操作 1: 清除标志
--	--	--	-------------------

9.5.2.4 AHB1 外设复位寄存器 (RMU\_AHB1RSTR)

AHB1 外设复位寄存器 (RMU_AHB1RSTR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved						SM3RST	Reserved		PISRST	TRNGRST	CRYPTRST	CALCRST	CRCRST	GPIORST	

Reserved	Bits 31-16	—	保留
Reserved	Bits 15-9	T_W1	保留
SM3RST	Bit8	T_W1	<b>SM3 复位</b> 0: 无操作 1: 复位
Reserved	Bit7-6	T_W1	保留
PISRST	Bit5	T_W1	<b>PIS 复位</b> 0: 无操作 1: 复位
TRNGRST	Bit4	T_W1	<b>TRNG 复位</b> 0: 无操作 1: 复位
CRYPTRST	Bit3	T_W1	<b>CRYPT 复位</b> 0: 无操作 1: 复位
CALCRST	Bit2	T_W1	<b>CALC 复位</b> 0: 无操作 1: 复位
CRCRST	Bit1	T_W1	<b>CRC 复位</b> 0: 无操作 1: 复位
GPIORST	Bit0	T_W1	<b>GPIO 复位</b> 0: 无操作 1: 复位

### 9.5.2.5 AHB2 外设复位寄存器 (RMU\_AHB2RSTR)

AHB2 外设复位寄存器 (RMU_AHB2RSTR)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CPURST	CHIPRST		

Reserved	Bits 31-2	—	保留
CPURST	Bit 1	T_W1	<b>处理器内核复位</b> 0: 无操作 1: 复位 注: 该复位只复位处理器内核 (不包括 DEBUG 逻辑)
CHIPRST	Bit 0	T_W1	<b>芯片全局复位</b> 0: 无操作 1: 复位

9.5.2.6 APB1 外设复位寄存器 (RMU\_APB1RSTR)

APB1 外设复位寄存器 (RMU_APB1RSTR)																																
偏移地址: 30 <sub>H</sub>																																
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	QUADRST	KSCANRST	QSPIRST	PDPCMRST	ECCRST	Reserved	TKRST	ADPCMRST	I2C1RST	I2C0RST	Reserved	SPI1RST	SPI0RST	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Reserved	Bits 31	T_W1	保留
QUADRST	Bit30	T_W1	<b>QUAD</b> 复位 0: 无操作 1: 复位
KSCANRST	Bit29	T_W1	<b>KSCAN</b> 复位 0: 无操作 1: 复位
QSPIRST	Bit28	T_W1	<b>QSPI</b> 复位 0: 无操作 1: 复位
PDPCMRST	Bit27	T_W1	<b>PDMPCM</b> 复位 0: 无操作 1: 复位
ECCRST	Bit26	T_W1	<b>ECC</b> 复位 0: 无操作 1: 复位
Reserved	Bits 25-24	T_W1	保留
TKRST	Bit23	T_W1	<b>TK</b> 复位 0: 禁止 1: 使能
ADPCMRST	Bit22	T_W1	<b>ADPCM</b> 复位 0: 禁止 1: 使能
I2C1RST	Bit21	T_W1	<b>I2C1</b> 复位 0: 无操作 1: 复位
I2C0RST	Bit20	T_W1	<b>I2C0</b> 复位 0: 无操作 1: 复位
Reserved	Bits19-18	T_W1	保留
SPI1RST	Bit17	T_W1	<b>SPI1</b> 复位

			0: 无操作 1: 复位
SPI0RST	Bit16	T_W1	<b>SPI0 复位</b> 0: 无操作 1: 复位
Reserved	Bits15-11	T_W1	保留
UART2RST	Bit10	T_W1	<b>UART2 复位</b> 0: 无操作 1: 复位
UART1RST	Bit9	T_W1	<b>UART1 复位</b> 0: 无操作 1: 复位
UART0RST	Bit8	T_W1	<b>UART0 复位</b> 0: 无操作 1: 复位
Reserved	Bits7-6	T_W1	保留
TIM5RST	Bit 5	T_W1	<b>TIM5 复位</b> 0: 无操作 1: 复位
TIM4RST	Bit 4	T_W1	<b>TIM4 复位</b> 0: 无操作 1: 复位
TIM3RST	Bit 3	T_W1	<b>TIM3 复位</b> 0: 无操作 1: 复位
TIM2RST	Bit 2	T_W1	<b>TIM2 复位</b> 0: 无操作 1: 复位
TIM1RST	Bit 1	T_W1	<b>TIM1 复位</b> 0: 无操作 1: 复位
TIM0RST	Bit 0	T_W1	<b>TIM0 复位</b> 0: 无操作 1: 复位

9.5.2.7 APB2 外设复位寄存器 (RMU\_APB2RSTR)

APB2 外设复位寄存器 (RMU_APB2RSTR)																																		
偏移地址: 34 <sub>H</sub>																																		
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved																RTCRST	IWDTRST	Reserved	WWDTRST	Reserved											ADC0RST	Reserved		LP16TRST

Reserved	Bits 31-16	T_W1	保留
RTCRST	Bit15	T_W1	<b>RTC 复位</b> 0: 无操作 1: 复位
IWDTRST	Bit14	T_W1	<b>IWDTRST 复位</b> 0: 无操作 1: 复位
Reserved	Bit13	T_W1	保留
WWDTRST	Bit12	T_W1	<b>WWDTRST 复位</b> 0: 无操作 1: 复位
Reserved	Bits 11-5	T_W1	保留
ADC0RST	Bit4	T_W1	<b>ADC0 复位</b> 0: 无操作 1: 复位
Reserved	Bits 3-1	T_W1	保留
LP16TRST	Bit 0	T_W1	<b>LP16T复位</b> 0: 无复位 1: 复位

9.5.2.8 APB3 外设复位寄存器 (RMU\_APB3RSTR)

APB3 外设复位寄存器 (RMU_APB3RSTR)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											Reserved	BLEMDMRST			

Reserved	Bits 31-2	—	保留
Reserved	Bit1	T_W1	保留
BLEMDMRST	Bit0	T_W1	<b>BLE MODEM 复位</b> 0: 无操作 1: 复位

9.5.2.9 AHB2 BLE复位寄存器 (RMU\_AHB2BLERSTR)

APB2 BLE 复位寄存器 (RMU_AHB2BLERSTR)																															
偏移地址: 3C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												Reserved	LPRST	CRYPTRST	MASTRST

Reserved	Bits 31-4	—	保留
Reserved	Bits3	T_W1	保留
LPRST	Bit 2	T_W1	<b>BLE5.0 内 LOW POWER 复位</b> 0: 无操作 1: 复位
CRYPTRST	Bit 1	T_W1	<b>BLE5.0 内 CRYPT 复位</b> 0: 无操作 1: 复位
MASTRST	Bit 0	T_W1	<b>BLE5.0 内 MASTER 复位</b> 0: 无操作 1: 复位



## 第10章 时钟管理(CMU)

### 10.1 概述

时钟管理单位 (CMU) 主要作用是控制时钟和振荡器。CPU 可对内部各模块的时钟进行使能或关闭。软件可灵活地通过配置时钟或关闭不必要的时钟，来降低系统的总功耗。

### 10.2 特性

- ◇ 支持多种时钟源
  - ◇ 1~48MHz 外部高速晶体振荡器 (HOSC)
  - ◇ 24MHz 或 2MHz 可配置内部高速 RC 振荡器 (HRC)
  - ◇ 32.768KHz 外部低速晶体振荡器 (LOSC)
  - ◇ 128KHz 内部低速振 RC 荡器 (LRC)
  - ◇ 32.768KHz 内部超低速 RC 振荡器 (ULRC)
  - ◇ 内部锁相环倍频时钟 (PLL)
- ◇ 支持低功耗配置
- ◇ 快速的启动时间
- ◇ AHB 外设、APB 外设和 CPU 均可独立预分频
- ◇ 低功耗外设时钟可特别再预分频
- ◇ 内核和外设均支持独立的时钟门控
- ◇ 支持时钟从两个引脚输出，输出源可选择

### 10.3 结构框图

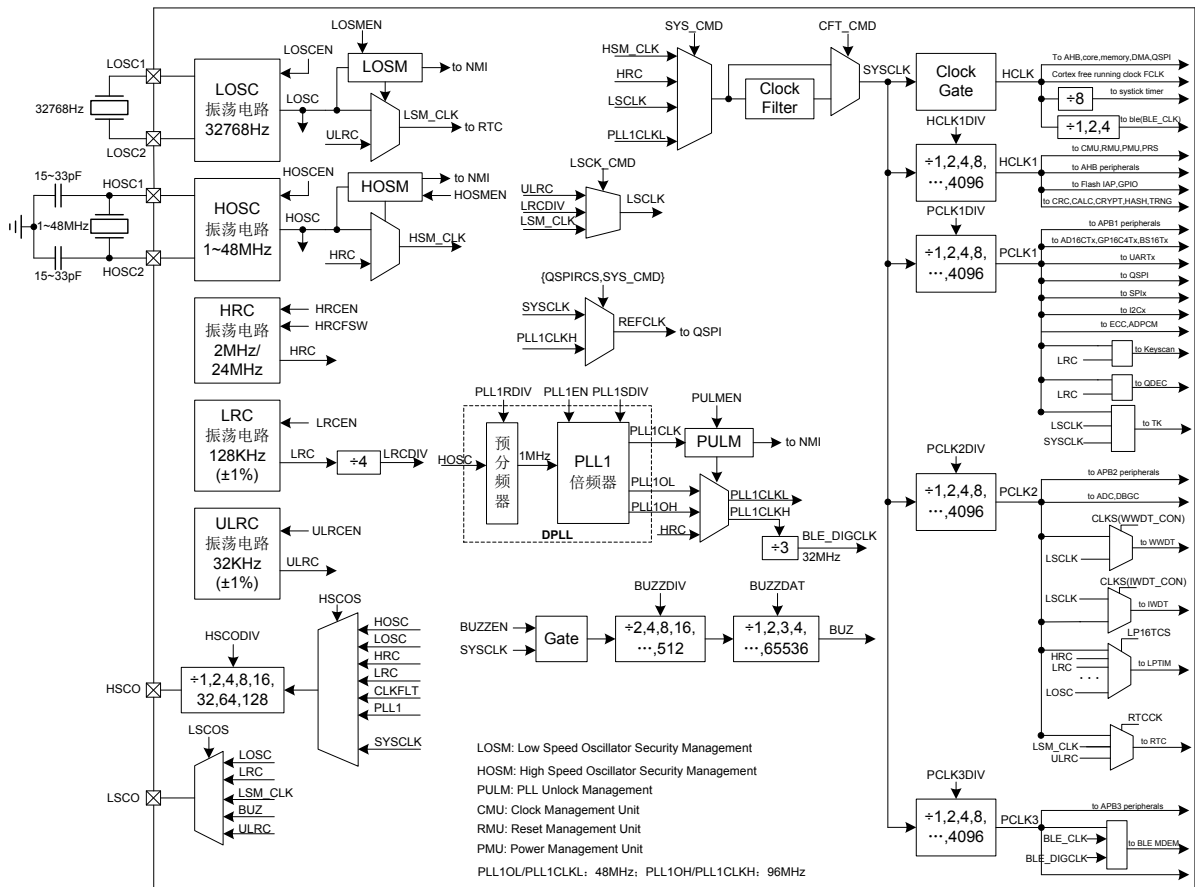


图 10-1 时钟结构图

- 注 1: BKPC\_PCCR[8]->LOSCEN, BKPC\_PCCR[9]->LOSMEN;
- 注 2: BKPC\_PCCR[12]->ULRCEN;
- 注 3: CMU\_HOSMCR[0]->HOSMEN;
- 注 4: CMU\_PULMCR[0]->PULMEN;
- 注 5: BKPC\_PCCR[0]->RTCCK.

### 10.4 系统时钟

#### 10.4.1 外部高速振荡器时钟HOSC

外部高速时钟频率范围为 1~48MHz, 实际应用时建议 HOSC 采用 32MHz 的晶体振荡器。

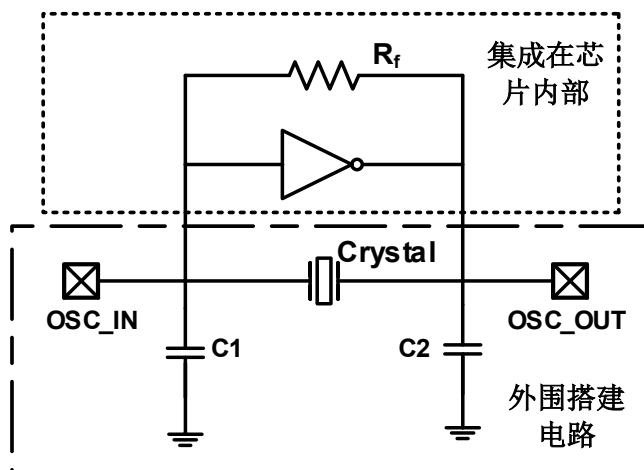


图 10-2 HOSC 结构图

#### 10.4.2 内部高速RC振荡器时钟HRC

提供 24MHz 和 2MHz 内部 RC 振荡频率时钟

#### 10.4.3 外部低速振荡器时钟LOSC

32.768KHz 的晶体振荡器

#### 10.4.4 内部低速RC振荡器时钟LRC

可校准有使能选择的 128KHz 方波振荡器

#### 10.4.5 内部超低速RC振荡器时钟ULRC

可校准有使能选择的 32.768KHz 方波振荡器

#### 10.4.6 内部倍频时钟 (PLL)

内部倍频器 PLL 的时钟源仅支持外部高速振荡器 HOSC，外部时钟源 HOSC 输入到 PLL 时首先需要进行预分频到 1MHz,然后倍频至 48MHz 和 96MHz。

当蓝牙通信模块处于工作状态时，系统时钟需要选择 PLL，同时 PLL 的 96MHz 时钟需要使能。

#### 10.4.7 系统时钟选择

系统时钟可从 HOSC,HRC,LOSC,LRC,ULRC 以及 PLL 时钟等时钟源中进行选择和切换。可对 CMU 控制状态寄存器 (CMU\_CSR) 的 SYS\_STU 位进行配置选择和切换时钟。

#### 10.4.8 高速振荡器安全管理HOSM

主要对外部高速晶振进行管理，当使能高速振荡器安全管理模块后，当检测到外部高速晶振停振时电路会将时钟自动切换到内部 HRC 时钟。具体的控制方式描述参照寄存器 CMU\_HOSMCR。

---

#### 10.4.9 低速振荡器安全管理LOSM

---

主要对外部低速晶振（32.768KHz）进行管理，当使能低速振荡器安全管理模块后，当检测到外部低速晶振停振时电路会将时钟自动切换到内部 ULRC 时钟。具体的控制方式描述参照寄存器 CMU\_LOSMCR。

---

#### 10.4.10 RTC时钟

---

RTC 的时钟源可以在 LOSC、ULRC 之间进行选择。

---

#### 10.4.11 IWDT时钟

---

IWDT 的时钟源可以在 SYSCLK、ULRC、LOSC、LRCDIV 之间进行选择。

---

#### 10.4.12 时钟输出选择

---

参见时钟结构图中的 HSCO 和 LSCO，主要用于时钟的测试。

## 10.5 特殊功能寄存器

### 10.5.1 寄存器列表

CMU 寄存器列表			
名称	偏移地址	类型	描述
CMU 基地址: 4008_0400 <sub>H</sub>			
CMU_CSR	000 <sub>H</sub>	R/W	CMU 控制状态寄存器
CMU_CFGR	004 <sub>H</sub>	R/W	CMU 配置寄存器
CMU_CLKENR	010 <sub>H</sub>	R/W	CMU 时钟使能寄存器
CMU_CLKSR	014 <sub>H</sub>	R	CMU 时钟状态寄存器
CMU_PLLCFG	018 <sub>H</sub>	R/W	PLL 配置寄存器
Reserved	01C <sub>H</sub>	-	保留
CMU_HOSMCR	020 <sub>H</sub>	R/W	HOSC 安全管理控制寄存器
CMU_LOSMCR	024 <sub>H</sub>	R/W	LOSC 安全管理控制寄存器
CMU_PULMCR	028 <sub>H</sub>	R/W	PLL 失锁管理控制寄存器
Reserved	02C <sub>H</sub>	-	保留
CMU_CLKOCR	030 <sub>H</sub>	R/W	CMU 时钟输出控制寄存器
CMU_BUZZCR	034 <sub>H</sub>	R/W	BUZZ 控制寄存器
Reserved	038 <sub>H</sub> -03C <sub>H</sub>	-	保留
CMU_AHB1ENR	040 <sub>H</sub>	R/W	AHB1 外设时钟使能寄存器
Reserved	044 <sub>H</sub> -04C <sub>H</sub>	-	保留
CMU_APB1ENR	050 <sub>H</sub>	R/W	APB1 外设时钟使能寄存器
CMU_APB2ENR	054 <sub>H</sub>	R/W	APB2 外设时钟使能寄存器
Reserved	058 <sub>H</sub> -05C <sub>H</sub>	-	保留
CMU_AHB1LPENR	060 <sub>H</sub>	R/W	时钟模块低功耗模式使能寄存器
Reserved	064 <sub>H</sub> -07C <sub>H</sub>	-	保留
CMU_PERICR	080 <sub>H</sub>	R/W	外设时钟控制寄存器
CMU_CKLDOT	084 <sub>H</sub>	R/W	时钟模块 LDO 稳定时间控制寄存器
CMU_HOSCT	088 <sub>H</sub>	R/W	HOSC 稳定时间控制寄存器
CMU_PLLT	08C <sub>H</sub>	R/W	PLL 稳定时间控制寄存器
CMU_CKSEQR	090 <sub>H</sub>	R/W	时钟开启模式控制寄存器
Reserved	094 <sub>H</sub> -0AC <sub>H</sub>	-	保留
CMU_PDPCMCFG	0B0 <sub>H</sub>	R/W	PDMPM 时钟配置
Reserved	0B4 <sub>H</sub> -0BC <sub>H</sub>	-	保留
CMU_BLEENR	0C0 <sub>H</sub>	R/W	BLE 时钟使能寄存器
CMU_BLECFG	0C4 <sub>H</sub>	R/W	BLE 时钟配置寄存器
CMU_BLECKS	0C8 <sub>H</sub>	R/W	BLE 时钟选择寄存器
Reserved	0CC <sub>H</sub> -0DC <sub>H</sub>	-	保留
CMU_ULRCUMC	0E0 <sub>H</sub>	R/W	ULRC 时钟用户校准模式寄存器
CMU_LRCUMC0	0E4 <sub>H</sub>	R/W	LRC 时钟用户校准模式寄存器 0
CMU_LRCUMC1	0E8 <sub>H</sub>	R/W	LRC 时钟用户校准模式寄存器 1

CMU 寄存器列表			
名称	偏移地址	类型	描述
CMU 基地址: 4008_0400 <sub>H</sub>			
CMU_LOSCUMC0	0EC <sub>H</sub>	R/W	LOSC 时钟用户校准模式寄存器 0
CMU_LOSCUMC1	0F0 <sub>H</sub>	R/W	LOSC 时钟用户校准模式寄存器 1
CMU_HOSCUMC	0F4 <sub>H</sub>	R/W	HOSC 时钟用户校准模式寄存器

## 10.5.2 寄存器描述

### 10.5.2.1 CMU控制状态寄存器 (CMU\_CSR)

CMU 控制状态寄存器 (CMU_CSR)																															
偏移地址: 000H																															
复位值: 01000000_00000000_00000001_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSCK_STU		LSCK_CMD		Reserved		CFT_RDYN	CFT_STU	CFT_CMD								Reserved	LSCLK_RDYN	SYS_RDYN	Reserved	SYS_STU			Reserved					SYS_CMD			

LSCK_STU	Bits 31-30	R	<p><b>当前选择低速时钟状态位</b></p> <p>00: 保留</p> <p>01: 选择ULRC</p> <p>10: 选择LRC/4</p> <p>11: 选择LOSC</p>
LSCK_CMD	Bits 29-28	W	<p><b>低速时钟切换命令</b></p> <p>00: 无操作</p> <p>01: 选择ULRC</p> <p>10: 选择LRC/4</p> <p>11: 选择LOSC</p> <p>注1: 切换后的时钟为IWWDT/MWDT/TK/BLE以及系统控制单元提供低速时钟源;</p> <p>注2: 低速时钟选择ULRC/LOSC时, 需要注意同时变更BKPC_PCCR[0]控制位;</p>
Reserved	Bits 27-26	—	保留
CFT_RDYN	Bit 25	R	<p><b>系统时钟滤波器切换状态位</b></p> <p>0: 系统时钟滤波器切换完成或未发生切换动作</p> <p>1: 系统时钟滤波器正在切换</p>
CFT_STU	Bit 24	R	<p><b>系统时钟滤波器激活状态位</b></p> <p>0: 系统时钟滤波器被选择</p> <p>1: 系统时钟滤波器被旁路</p>
CFT_CMD	Bits 23-16	W	<p><b>系统时钟滤波切换命令位</b></p> <p>0x55: 选择系统时钟滤波器</p> <p>0xAA: 旁路系统时钟滤波器</p> <p>其他: 无操作</p> <p>注1: 系统默认为选择系统时钟滤波器。</p> <p>注2: 当系统时钟滤波器正在切换时, 该位写入无效。该位读出始终为0。</p>
Reserved	Bits 15-14	—	保留
LSCLK_RDYN	Bit 13	R	<b>低速时钟切换状态位</b>

			0: 低速时钟切换完成或未发生切换动作 1: 低速时钟正在切换
SYS_RDYN	Bit 12	R	<b>系统时钟切换状态位</b> 0: 系统时钟切换完成或未发生切换动作 1: 系统时钟正在切换
Reserved	Bit 11	—	保留
SYS_STU	Bits 10-8	R	<b>当前系统时钟状态位</b> 000: 保留 001: 选择HRC 010: 保留 011: 保留 100: 选择PLL1 101: 选择HOSC 110: 选择LSCLK (ULRC/LOSC/LRCDIV) 111: 保留
Reserved	Bits 7-3	—	保留
SYS_CMD	Bits 2-0	W	<b>系统时钟切换命令位</b> 000: 无操作 001: 选择HRC 010: 无操作 011: 无操作 100: 选择PLL1 101: 选择HOSC (HOSC时钟管理器输出) 110: 选择LSCLK 111: 保留 注1: 系统默认为选择HRC。 注2: 当系统时钟正在切换时, 该位写入无效。该位读出始终为0。



10.5.2.2 CMU配置寄存器 (CMU\_CFGR)

CMU 配置寄存器 (CMU_CFGR)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 01000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSPIRCS	QSPIRAEN	PCLK3DIV				HRCFST	HRCFSW	PCLK2DIV				PCLK1DIV				HCLK4DIV				HCLK3DIV				HCLK2DIV				HCLK1DIV			

QSPIRCS	Bits 31	R/W	<p><b>QSPI REF CLK 频率选择控制位</b></p> <p>0: 选择系统时钟</p> <p>1: 选择 PLL1 的高频时钟 (96MHz)</p> <p>注 1: 当系统时钟选择为 PLL1 (CMU_CSR[10:8]=3'b100), QSPIRAEN=1'b1 时, QSPI REF CLK 自动选择 PLL1 的高频时钟;</p> <p>注2: 当系统时钟选择非PLL1时, 设置此控制位为 1, 同时设置CMU_CLKENR[9:8]=2'b11时, 可以选择PLL1的高频时钟作为QSPI REF CLK, 否则选择系统时钟。</p>
QSPIRAEN	Bit 30	R/W	<p><b>QSPI REF CLK 自动切换为 96MHz 使能位</b></p> <p>0: 自动切换禁止</p> <p>1: 自动切换使能</p>
PCLK3DIV	Bits 29-26	R/W	<p><b>PCLK3 分频选择位</b></p> <p>0000: 1分频</p> <p>0001: 2分频</p> <p>0010: 4分频</p> <p>0011: 8分频</p> <p>.....</p> <p>1100: 4096分频</p> <p>其他: 保留</p>
HRCFST	Bit 25	R	<p><b>HRC 频率选择状态位</b></p> <p>0: 24MHz</p> <p>1: 2MHz</p>
HRCFSW	Bit 24	R/W	<p><b>HRC 频率软件选择位</b></p> <p>0: 24MHz</p> <p>1: 2MHz</p>
PCLK2DIV	Bits 23-20	R/W	<p><b>PCLK2 分频选择位</b></p> <p>0000: 1分频</p> <p>0001: 2分频</p> <p>0010: 4分频</p> <p>0011: 8分频</p> <p>.....</p>

			1100: 4096分频 其他: 保留
PCLK1DIV	Bits 19-16	R/W	<b>PCLK1 分频选择位</b> 0000: 1分频 0001: 2分频 0010: 4分频 0011: 8分频 ..... 1100: 4096分频 其他: 保留
HCLK4DIV	Bits 15-12	R/W	<b>HCLK4 分频选择位 (保留未用)</b> 0000: 1分频 0001: 2分频 0010: 4分频 0011: 8分频 ..... 1100: 4096分频 其他: 保留
HCLK3DIV	Bits 11-8	R/W	<b>HCLK3 分频选择位 (保留未用)</b> 0000: 1分频 0001: 2分频 0010: 4分频 0011: 8分频 ..... 1100: 4096分频 其他: 保留
HCLK2DIV	Bits 7-4	R/W	<b>HCLK2 分频选择位 (保留未用)</b> 0000: 1分频 0001: 2分频 0010: 4分频 0011: 8分频 ..... 1100: 4096分频 其他: 保留
HCLK1DIV	Bits 3-0	R/W	<b>HCLK1 分频选择位</b> 0000: 1分频 0001: 2分频 0010: 4分频 0011: 8分频 ..... 1100: 4096分频 其他: 保留

10.5.2.3 CMU时钟使能寄存器 (CMU\_CLKENR)

CMU 时钟使能寄存器 (CMU_CLKENR)																																													
偏移地址: 010 <sub>H</sub>																																													
复位值: 00000000_00000000_00010000_00011100 <sub>B</sub>																																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Reserved																Reserved	HOSCETCEN	HOSCHPS	BIASEN	LDOEN	PLL1EN_96M	PLL1EN	Reserved	HOSCBEN	ULRCEN	LRCEN	HRCEN	LOSCEN	HOSCEN																

Reserved	Bits 31-16	—	保留
Reserved	Bits 15-14	—	保留
HOSCETCEN	Bit 13	R/W	<b>HOSC 外灌时钟使能位</b> 0: 禁止 1: 使能 注: 仅用于测试模式
HOSCHPS	Bit 12	R/W	<b>HOSC 高低功耗选择模式</b> 0: 高功耗模式 1: 低功耗模式
BIASEN	Bit 11	R/W	<b>HOSC 和 PLL1 基准软件使能位</b> 0: 禁止 1: 使能 注: 此控制位无效
LDOEN	Bit 10	R/W	<b>HOSC 和 PLL1 数字 LDO 软件使能位</b> 0: 禁止 1: 使能 注: 此控制位无效
PLL1EN_96M	Bit 9	R/W	<b>PLL1_96M 软件使能位</b> 0: 禁止 1: 使能 注: 在 PLL1 的测试模式下即 CMU_CKSEQR->CLSTM=1 时, 此控制位无效
PLL1EN	Bit 8	R/W	<b>PLL1_48M 软件使能位</b> 0: 禁止 1: 使能
Reserved	Bits 7-6	—	保留
HOSCBEN	Bit 5	R/W	<b>HOSC 输出 BUFFER 软件使能位</b> 0: 禁止 1: 使能 注: 此控制位当 CMU_CKSEQR->CLSTM=1 时 有效

ULRCEN	Bit 4	R/W	<b>ULRC 软件使能位（此控制位无效）</b> 0: 禁止 1: 使能 注：此控制位无效，ULRC 的软件使能位参照寄存器 BKPC_PCCR 描述
LRCEN	Bit 3	R/W	<b>LRC 软件使能位</b> 0: 禁止 1: 使能
HRCEN	Bit 2	R/W	<b>HRC 软件使能位</b> 0: 禁止 1: 使能
LOSCEN	Bit 1	R/W	<b>LOSC 软件使能位（此控制位无效）</b> 0: 禁止 1: 使能 注：此控制位无效，LOSC 的软件使能位参照寄存器 BKPC_PCCR 描述
HOSCEN	Bit 0	R/W	<b>HOSC 软件使能位</b> 0: 禁止 1: 使能

10.5.2.4 CMU时钟状态寄存器 (CMU\_CLKSR)

CMU 时钟状态寄存器 (CMU_CLKSR)																																
偏移地址: 014 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00011100 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved							PLL1RDY	Reserved				ULRCRDY	LRCRDY	HRCRDY	LOSCRDY	HOSCRDY	Reserved							PLL1ACT	Reserved		Reserved	ULRCACT	LRCACT	HRCACT	LOSCACT	HOSCACT

Reserved	Bits 31-25	—	保留
PLL1RDY	Bit 24	R	<b>PLL1 稳定标志位</b> 0: 未稳定 1: 稳定
Reserved	Bits 23-21	—	保留
ULRCRDY	Bit 20	R	<b>ULRC 稳定标志位</b> 0: 未稳定 1: 稳定
LRCRDY	Bit 19	R	<b>LRC 稳定标志位</b> 0: 未稳定 1: 稳定
HRCRDY	Bit 18	R	<b>HRC 稳定标志位</b> 0: 未稳定 1: 稳定
LOSCRDY	Bit 17	R	<b>LOSC 稳定标志位</b> 0: 未稳定 1: 稳定
HOSCRDY	Bit 16	R	<b>HOSC 稳定标志位</b> 0: 未稳定 1: 稳定
Reserved	Bits 15-9	—	保留
PLL1ACT	Bit 8	R	<b>PLL1 激活状态位</b> 0: 未激活 1: 激活
Reserved	Bits 7-6	—	保留
Reserved	Bit 5	—	保留
ULRCACT	Bit 4	R	<b>ULRC 激活状态位</b> 0: 未激活 1: 激活
LRCACT	Bit 3	R	<b>LRC 激活状态位</b> 0: 未激活 1: 激活

HRACT	Bit 2	R	<b>HRC 激活状态位</b> 0: 未激活 1: 激活
LOSCACT	Bit 1	R	<b>LOSC 激活状态位</b> 0: 未激活 1: 激活
HOSCACT	Bit 0	R	<b>HOSC 激活状态位</b> 0: 未激活 1: 激活

10.5.2.5 PLL配置寄存器 (CMU\_PLLCFG)

PLL 配置寄存器 (CMU_PLLCFG)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 10000001_00000001_00000100_00110000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1LCKN	Reserved					PLL1OS	Reserved	PLL1BYPS	PLL1RSV	Reserved	PLL1RDIV					PLL1ICP	Reserved	PLL1SDIV													

PLL1LCKN	Bit 31	R	<b>PLL1 锁定标志位</b> 0: 锁定 1: 未锁定
Reserved	Bits 30-26	—	保留
PLL1OS	Bits 25-24	R/W	<b>PLL1 输出时钟选择位(输出到测试端口)</b> x0: ×6 (96MHz) x1: ×3 (48MHz)
Reserved	Bit 23	—	保留
PLL1BYPS	Bit22	R/W	<b>PLL1 输出 48MHz 时钟来源选择:</b> 0: PLL1 倍频时钟 1: HOSC 注: 晶振为32MHz时, 只能选择PLL1; 晶振为48MHz, 同时PLL1异常时可选择HOSC;
PLL1RSV	Bits 21-20	R/W	<b>PLL1保留控制位</b>
Reserved	Bits 19-18	—	保留
PLL1RDIV	Bits 17-12	R/W	<b>PLL1输入时钟预分频控制位</b> Fref=HOSC_CLK/PLL1RDIV 其中Fref时钟为1MHz
PLL1ICP	Bits 11-8	R/W	<b>PLL1 CP模块电流控制位</b> 0.25uA*PLL1ICP
Reserved	Bit 7	—	保留
PLL1SDIV	Bits 6-0	R/W	<b>PLL1 分频数控控制位</b> 6'h30: 48MHz

10.5.2.6 HOSC安全管理控制寄存器 (CMU\_HOSMCR)

HOSC 安全管理控制寄存器 (CMU_HOSMCR)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRPIE	STRIF	STPIF	STRIE	STPIE	Reserved						FRQS		Reserved						CLKS	EN

Reserved	Bits 31-21	—	保留
STRPIE	Bit 20	R/W	<b>HOSC起振和停振不可屏蔽中断使能位(NMI)</b> 0: 禁止 1: 使能
STRIF	Bit 19	R/C_W1	<b>HOSC起振标志位</b> 0: 未发生 HOSC 起振或标志位已被清除 1: 发生HOSC起振 注: 该位写1清除, 写0无效
STPIF	Bit 18	R/C_W1	<b>HOSC停振标志位</b> 0: 未发生 HOSC 停振或标志位已被清除 1: 发生HOSC停振 注: 该位写1清除, 写0无效
STRIE	Bit 17	R/W	<b>HOSC起振中断使能位</b> 0: 禁止 1: 使能
STPIE	Bit 16	R/W	<b>HOSC停振中断使能位</b> 0: 禁止 1: 使能
Reserved	Bits 15-11	—	保留
FRQS	Bits 10-8	R/W	<b>HOSC 频率配置位</b> 000: 1~2MHz 001: 2~4MHz 010: 4~8MHz 011: 8~16MHz 1xx: 16~32MHz 注: 当频率处在两个相邻档位的临界值时, 这两个档位可任意选择
Reserved	Bits 7-2	—	保留
CLKS	Bit 1	R	<b>当前选择时钟状态位</b> 0: 选择 HOSC 1: 选择HRC
EN	Bit 0	R/W	<b>HOSC 安全管理使能位</b>



			0: 禁止 1: 使能
--	--	--	----------------

### 10.5.2.7 LOSC安全管理控制寄存器 (CMU\_LOSMCR)

LOSC 安全管理控制寄存器 (CMU_LOSMCR)																																	
偏移地址: 024 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved													STRIF	STPIF	STRIE	STPIE	Reserved								Reserved	Reserved						CLKS	EN

Reserved	Bits 31-20	—	保留
STRIF	Bit 19	R/C_W1	<b>LOSC起振标志位</b> 0: 未发生 LOSC 起振或标志位已被清除 1: 发生LOSC起振 注: 该位写1清除, 写0无效
STPIF	Bit 18	R/C_W1	<b>LOSC停振标志位</b> 0: 未发生 LOSC 停振或标志位已被清除 1: 发生LOSC停振 注: 该位写1清除, 写0无效
STRIE	Bit 17	R/W	<b>LOSC起振中断使能位</b> 0: 禁止 1: 使能 注: 此控制位无效
STPIE	Bit 16	R/W	<b>LOSC停振中断使能位</b> 0: 禁止 1: 使能 注:此控制位无效
Reserved	Bits 15-10	—	保留
Reserved	Bits 9-8	—	保留
Reserved	Bits 7-2	—	保留
CLKS	Bit 1	R	<b>当前选择时钟状态位</b> 0: 选择 LOSC 1: 选择ULRC
EN	Bit 0	R	<b>LOSC 安全管理使能位</b> 0: 禁止 1: 使能 注: 芯片配置字CFG_WORD0配置值

10.5.2.8 PLL失锁管理控制寄存器 (CMU\_PULMCR)

PLL 失锁管理控制寄存器 (CMU_PULMCR)																															
偏移地址: 028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											LCULKIE	LCKIF	ULKIF	LCKIE	ULKIE	Reserved						MODE	Reserved						CLKS	EN	

Reserved	Bits 31-21	—	保留
LCULKIE	Bit 20	R/W	<b>PLL锁定和失锁不可屏蔽中断使能位 (NMI)</b> 0: 禁止 1: 使能
LCKIF	Bit 19	R/C_W1	<b>PLL锁定标志位</b> 0: 未发生 PLL 失锁或标志位已被清除 1: 发生PLL失锁 注: 该位写1清除, 写0无效
ULKIF	Bit 18	R/C_W1	<b>PLL失锁标志位</b> 0: 未发生 PLL 失锁或标志位已被清除 1: 发生PLL失锁 注: 该位写1清除, 写0无效
LCKIE	Bit 17	R/W	<b>PLL锁定中断使能位</b> 0: 禁止 1: 使能
ULKIE	Bit 16	R/W	<b>PLL失锁定中断使能位</b> 0: 禁止 1: 使能
Reserved	Bits 15-10	—	保留
MODE	Bits 9-8	R/W	<b>PLL失锁管理模式选择位</b> xx: 失锁后切换至HRC, 待重新后锁定切回
Reserved	Bits 7-2	—	保留
CLKS	Bit 1	R	<b>当前选择时钟状态位</b> 0: 选择 PLL 1: 选择HRC
EN	Bit 0	R/W	<b>PLL 失锁管理使能位</b> 0: 禁止 1: 使能

10.5.2.9 CMU时钟输出控制寄存器 (CMU\_CLKOCR)

CMU 时钟输出控制寄存器 (CMU_CLKOCR)																																		
偏移地址: 030 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved				LSCOS			Reserved									LSCOEN	Reserved	HSCODIV			Reserved	HSCOS			Reserved									HSCOEN

Reserved	Bits 31-27	—	保留
LSCOS	Bits 26-24	R/W	低速时钟输出源选择位 000: LOSC 001: LRC 010: LOSM 011: BUZZ 100: ULRC 1xx: 保留
Reserved	Bits 23-17	—	保留
LSCOEN	Bit 16	R/W	低速时钟输出使能位 0: 禁止 1: 使能
Reserved	Bit 15	—	保留
HSCODIV	Bits 14-12	R/W	高速时钟输出分频选择位 000: 1 分频 001: 2 分频 010: 4 分频 011: 8 分频 100: 16 分频 101: 32 分频 110: 64 分频 111: 128 分频
Reserved	Bit 11	—	保留
HSCOS	Bits 10-8	R/W	高速时钟输出源选择位 000: HOSC 001: LOSC 010: HRC 011: LRC 100: CLKFLT 101: PLL1 (48MHz/96MHz) 110: 保留 111: SYSClk

			注: 当配置为 3'b101 时根据 PLL1OS 的不同配置输出不同的时钟频率, 具体如下: 1) CMU_PLLCFG->PLL1OS = 1, 输出 48MHz; 2) CMU_PLLCFG->PLL1OS = 0, 输出 96MHz;
Reserved	Bits 7-1	—	保留
HSCOEN	Bit 0	R/W	高速时钟输出使能位 0: 禁止 1: 使能

### 10.5.2.10 BUZZ控制寄存器 (CMU\_BUZZCR)

BUZZ 控制寄存器 (CMU_BUZZCR)																															
偏移地址: 034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUZZDAT																Reserved				BUZZDIV			Reserved						BUZZEN		

BUZZDAT	Bits 31-16	R/W	<b>BUZZ频率数据值</b> $Freq_{BUZZ} = \frac{Freq_{SYSCLK}}{2^{(DIV+1)} \times (DAT+1)}$
Reserved	Bits 15-11	—	保留
BUZZDIV	Bits 10-8	R/W	<b>BUZZ 时钟分频选择位</b> 000: 2 分频 001: 4 分频 010: 8 分频 011: 16 分频 100: 32 分频 101: 64 分频 110: 128 分频 111: 256 分频
Reserved	Bits 7-1	—	保留
BUZZEN	Bit 0	R/W	<b>BUZZ 使能位</b> 0: 禁止 1: 使能

10.5.2.11 AHB1 外设时钟使能寄存器 (CMU\_AHB1ENR)

AHB1 外设时钟使能寄存器 (CMU_AHB1ENR)																															
偏移地址: 040 <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							SM3EN	Reserved		PISEN	TRNGEN	CRYPTEN	CALCEN	CRCEN	GPIOEN

Reserved	Bits 31-9	—	保留
SM3EN	Bit8	R/W	<b>SM3 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bit7-6	—	保留
PISEN	Bit5	R/W	<b>PIS 时钟使能位</b> 0: 禁止 1: 使能
TRNGEN	Bit4	R/W	<b>TRNG 时钟使能位</b> 0: 禁止 1: 使能
CRYPTEN	Bit3	R/W	<b>CRYPT 时钟使能位</b> 0: 禁止 1: 使能
CALCEN	Bit 2	R/W	<b>CALC 时钟使能位</b> 0: 禁止 1: 使能
CRCEN	Bit 1	R/W	<b>CRC 时钟使能位</b> 0: 禁止 1: 使能
GPIOEN	Bit 0	R/W	<b>GPIO 时钟使能位</b> 0: 禁止 1: 使能

10.5.2.12 APB1 外设时钟使能寄存器 (CMU\_APB1ENR)

APB1 外设时钟使能寄存器 (CMU_APB1ENR)																																
偏移地址: 050 <sub>H</sub>																																
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	QUADEN	KSCANEN	QSPIEN	PDPCMEN	ECCEN	Reserved	TKEN	ADPCMEN	I2C1EN	I2C0EN	Reserved	SPI1EN	SPI0EN	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Reserved	Bit 31	—	保留
QUADEN	Bit30	R/W	<b>QUAD 时钟使能位</b> 0: 禁止 1: 使能
KSCANEN	Bit29	R/W	<b>KSCAN 时钟使能位</b> 0: 禁止 1: 使能
QSPIEN	Bit28	R/W	<b>QSPI 时钟使能位</b> 0: 禁止 1: 使能
PDPCMEN	Bits27	R/W	<b>PDPCM 时钟使能位</b> 0: 禁止 1: 使能
ECCEN	Bit26	R/W	<b>ECC 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bit25-24	-	保留
TKEN	Bit23	R/W	<b>TK 时钟使能位</b> 0: 禁止 1: 使能
ADPCMEN	Bit22	R/W	<b>ADPCM 时钟使能位</b> 0: 禁止 1: 使能
I2C1EN	Bit21	R/W	<b>I2C1 时钟使能位</b> 0: 禁止 1: 使能
I2C0EN	Bit20	R/W	<b>I2C0 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bits19-18	—	保留
SPI1EN	Bit17	R/W	<b>SPI1 时钟使能位</b> 0: 禁止

			1: 使能
SPI0EN	Bit16	R/W	<b>SPI0 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bits15-11	—	保留
UART2EN	Bit10	R/W	<b>UART2 时钟使能位</b> 0: 禁止 1: 使能
UART1EN	Bit9	R/W	<b>UART1 时钟使能位</b> 0: 禁止 1: 使能
UART0EN	Bit8	R/W	<b>UART0 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bits7-6	—	保留
TIM5EN	Bit 5	R/W	<b>TIM5 时钟使能位</b> 0: 禁止 1: 使能
TIM4EN	Bit 4	R/W	<b>TIM4 时钟使能位</b> 0: 禁止 1: 使能
TIM3EN	Bit 3	R/W	<b>TIM3 时钟使能位</b> 0: 禁止 1: 使能
TIM2EN	Bit 2	R/W	<b>TIM2 时钟使能位</b> 0: 禁止 1: 使能
TIM1EN	Bit 1	R/W	<b>TIM1 时钟使能位</b> 0: 禁止 1: 使能
TIM0EN	Bit 0	R/W	<b>TIM0 时钟使能位</b> 0: 禁止 1: 使能

10.5.2.13 APB2 外设时钟使能寄存器 (CMU\_APB2ENR)

APB2 外设时钟使能寄存器 (CMU_APB2ENR)																																						
偏移地址: 054 <sub>H</sub>																																						
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Reserved												DBGEN	Reserved			BKPCEN	RTCEN	IWDTEN	Reserved			WWDTEN	Reserved											ADC0EN	Reserved			LP16TEN

Reserved	Bits 31-20	—	保留
DBGEN	Bit19	R/W	<b>DBG 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bit18-17	—	保留
BKPCEN	Bit16	R/W	<b>BKPC 时钟使能位</b> 0: 禁止 1: 使能
RTCEN	Bit15	R/W	<b>RTC 时钟使能位</b> 0: 禁止 1: 使能
IWDTEN	Bit14	R/W	<b>IWDT 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bit13	—	保留
WWDTEN	Bit12	R/W	<b>WWDT 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bit11-5	—	保留
ADC0EN	Bit4	R/W	<b>ADC0 时钟使能位</b> 0: 禁止 1: 使能
Reserved	Bits3-1	—	保留
LP16TEN	Bit 0	R/W	<b>LP16T 时钟使能</b> 0: 禁止 1: 使能



10. 5. 2. 14 时钟模块低功耗模式控制寄存器 (CMU\_AHBLPENR)

时钟模块低功耗模式控制寄存器 (CMU_AHBLPENR)																															
偏移地址: 060 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												Reserved	HRCEN	LOSCEN	LRCEN

Reserved	Bits 31-4	-	保留
Reserved	Bits 3	R/W	保留
HRCEN	Bit 2	R/W	保留 (此控制位无效)
LOSCEN	Bit 1	R/W	保留 (此控制位无效)
LRCEN	Bit 0	R/W	<b>STOP3低功耗模式LRC使能位</b> 0: 禁止 1: 使能

### 10.5.2.15 外设时钟控制寄存器 (CMU\_PERICR)

外设时钟控制寄存器 (CMU_PERICR)																															
偏移地址: 080 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													Reserved		Reserved					LP16TCS			Reserved								

Reserved	Bits 31-19	—	保留
Reserved	Bits 18-16	R/W	保留
Reserved	Bits 15-11	—	保留
LP16TCS	Bits 10-8	R/W	<b>LP16T 时钟选择位</b> 000: PCLK2 001: HRC (2MHz或24MHz) 010: HRC的分频时钟 (固定为1MHz) 011: HOSC 100: LRC 101: LOSM 110: ULRC 111: LS_CLK (由CMU_CSR->LSCK_CMD控制位决定)
Reserved	Bits 7-0	—	保留

10.5.2.16 时钟模块LDO稳定时间控制寄存器 (CMU\_CKLDOT)

时钟模块 LDO 稳定时间控制寄存器 (CMU_CKLDOT)																															
偏移地址: 084 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDOF		Reserved																			LDOT										

LDOF	Bit31	R	时钟 LDO 稳定时间计数完成标志位 0: 未完成 1: 完成
Reserved	Bits30-12	-	保留
LDOT	Bits11-0	R/W	等待时钟模块LDO稳定时间控制位 Time=LDOT*TIscck  注: 主要针对外部高速晶振和PLL1时钟模块, 当LDO稳定后使能外部高速晶振或PLL1 (由内部硬件电路控制)

### 10.5.2.17 HOSC稳定时间控制寄存器 (CMU\_HOSCT)

HOSC 稳定时间控制寄存器 (CMU_HOSCT)																															
偏移地址: 088 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00010000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HOSCF	Reserved																			HOSCT											

HOSCF	Bit31	R	外部高速晶振 HOSC 稳定时间计数完成标志位 0: 未完成 1: 完成
Reserved	Bits30-20	-	保留
Reserved	Bits19-10	R/W	保留
HOSCT	Bits9-0	R/W	等待外部高速晶振 HOSC 稳定时间控制位 Time=HOSCT* <i>T</i> <sub>Isclk</sub>  注: 当外部高速晶振稳定后可使能内部 PLL1 或切换系统时钟到外部高速晶振 (由内部硬件电路控制)

10.5.2.18 PLL稳定时间控制寄存器 (CMU\_PLLT)

PLL 稳定时间控制寄存器 (CMU_PLLT)																															
偏移地址: 08C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1F		Reserved																		PLL1T											

PLL1F	Bit31	R	内部 <b>PLL1</b> 稳定时间计数完成标志位 0: 未完成 1: 完成
Reserved	Bits30-12	-	保留
Reserved	Bits11-5	R/W	保留
PLL1T	Bits4-0	R/W	等待内部 <b>PLL1</b> 稳定时间控制位 Time=PLL1T*TIsclock  注: 当 <b>PLL1</b> 稳定后可将系统时钟切换到 <b>PLL1</b> (由内部硬件电路控制)

10.5.2.19 时钟开启模式控制寄存器 (CMU\_CKSEQR)

时钟开启模式控制寄存器 (CMU_CKSEQR)																															
偏移地址: 090 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												LOCK_BYP		CKSTM	

Reserved	Bit 31-2	-	保留
LOCK_BYP	Bit 1	R/W	<p><b>PLL的LOCK指示位软件控制</b></p> <p>0: LOCK指示位由PLL模块产生</p> <p>1: LOCK常有效</p> <p>注: 如果PLL模块产生LOCK信号异常时, 需要将此控制位置1</p>
CKSTM	Bit 0	R/W	<p><b>HOSC和PLL的开启模式控制位</b></p> <p>0: 由内部硬件电路自动控制</p> <p>1: 由软件控制</p> <p>注: 由于HOSC和PLL的开启有时序要求:</p> <p>1) 当CKSTM=0时, HOSC和PLL的开启时序由内部硬件电路根据寄存器CMU_CKLDOT、CMU_HOSCT、CMU_PLLT的设置值, 依次开启相应的模块;</p> <p>2) 当CKSTM=1时, HOSC和PLL的开启时序完全由软件控制, 软件通过依次配置相应模块的使能信号开启相应的时钟模块; 但是当需要同时使能PLL输出96MHz时钟频率时, 需要同时设置CMU_CFGR[31]为1, 即“CMU_CFGR[31]=1 和 CMU_CLKENR[8]=1” (在这种测试模式下对于PLL模块而言CMU_CLKENR[9]控制位无效, 仅CMU_CLKENR[8]控制位有效)</p>

注: 当系统运行时钟为 HOSC 或 PLL1 时, 对此寄存器写无效; (此寄存器仅用于测试)

10.5.2.20 PDMP2PCM时钟配置寄存器 (CMU\_PDPCMCFG)

PDMP2PCM 时钟配置寄存器 (CMU_PDPCMCFG)																															
偏移地址: 0B0 <sub>H</sub>																															
复位值: 00000000_00001110_00000010_10101011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FDIV				FW											

Reserved	Bit 31-20	-	保留
FDIV	Bit 19-16	R/W	<p><b>PDM2PCM模块工作时钟DCLK分频比系数设置 (DCLK时钟频率要求固定为8MHz, 时钟源为SYSCLK)</b></p> <p>4'b000_0: 1.0分频                      4'b000_1: 1.0分频                      4'b001_0: 1.0分频                      4'b001_1: 1.5分频                      4'b010_0: 2.0分频                      4'b010_1: 2.5分频                      4'b011_0: 3.0分频                      4'b011_1: 3.5分频                      4'b100_0: 4.0分频                      4'b100_1: 4.5分频                      4'b101_0: 5.0分频                      4'b101_1: 5.5分频                      4'b110_0: 6.0分频                      4'b110_1: 6.5分频                      4'b111_x: 系统自动配置分频比</p> <p>注: 根据系统工作的主频SYSCLK, 配置相应的分频系数产生8MHz工作时钟。当FDIV设置为4'b111_x时, 分频系数根据选择的系统时钟自动配置。</p>
FW	Bit15- 0	R/W	<p><b>PDM2PCM输入采样时钟SCLK频率字 (SCLK: 0.500MHz~5.000MHz)</b></p> <p><math>F_{sclk} = (F_{sys} * FW) / 2^{16}</math></p> <p>注: 其中Fsys为系统时钟SYSCLK工作频率, Fsclk为输入采样时钟SCLK的工作频率</p>

### 10.5.2.21 BLE时钟使能寄存器 (CMU\_BLEENR)

BLE 时钟使能寄存器 (CMU_BLEENR)																															
偏移地址: 0C0 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												MDMEN1	MDMEN0	BBEN1	BBEN0

Reserved	Bit31-4	—	保留
MDMEN1	Bit3	R/W	保留
MDMEN0	Bit2	R/W	<b>BLE Modem时钟使能位</b> 0: 禁止 1: 使能
BBEN1	Bit1	R/W	<b>BLE Baseband时钟hgclk/emclk门控使能</b> 0: 禁止 1: 使能
BBEN0	Bit 0	R/W	<b>BLE Baseband时钟使能位</b> 0: 禁止 1: 使能



10.5.2.22 BLE时钟配置寄存器 (CMU\_BLECFG)

BLE 时钟配置寄存器 (CMU_BLECFG)																															
偏移地址: 0C4 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved				BBMCKS			

Reserved	Bit31-8	—	保留
Reserved	Bit7-2	—	保留
BBMCKS	Bit1- 0	R/W	<b>BLE Baseband工作基准时钟选择位</b> 00: 系统时钟 1 分频 01: 系统时钟 2 分频 10: 系统时钟 4 分频 11: 系统时钟 4 分频

### 10.5.2.23 BLE时钟选择寄存器 (CMU\_BLECKS)

BLE 时钟选择寄存器 (CMU_BLECKS)																															
偏移地址: 0C8 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							Reserved		BBSCKS						

Reserved	Bit31-8	—	保留
Reserved	Bit7-6	—	保留
BBSCKS	Bit5- 0	R/W	<b>BLE Baseband时钟频率选择位</b> 注: 默认值为6'd12, 即时钟频率为12MHz。需要根据CMU_BLECFG的配置, 相应的配置此寄存器。

### 10.5.2.24 ULRC时钟用户校准模式寄存器 (CMU\_ULRCUMC)

ULRC 时钟用户校准模式寄存器 (CMU_ULRCUMC)																															
偏移地址: 0E0 <sub>H</sub>																															
复位值: 00000000_00000000_00101001_11110000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST																Reserved		IBISEL		ISEL				CAPSEL							

UMCST	Bits31-16	R/W	<b>ULRC用户校准模式使能位</b> 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
Reserved	Bits15-14	R/W	保留
IBISEL	Bits 13-12	R/W	<b>IBIAS 电流选择控制位</b> 00: 01: 10: 10nA 11:
ISEL	Bits11-8	R/W	<b>电容充电电流控制位</b> 0000: MIN (0nA) ..... 1111: MAX (37.5nA) 注: $I=2.5nA*ISEL$
CAPSEL	Bits7- 0	R/W	<b>电容选择控制位</b> 00000: frequency max ..... 11111: frequency min

10.5.2.25 LRC时钟用户校准模式寄存器 0 (CMU\_LRCUMC0)

LRC 时钟用户校准模式寄存器 0 (CMU_LRCUMC0)																															
偏移地址: 0E4 <sub>H</sub>																															
复位值: 00000000_00000000_01011000_11110000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST0																Reserved	VDDSEL	BUF_ISEL	ISEL				CAPSEL								

UMCST0	Bits31-16	R/W	<b>LRC用户校准模式使能位</b> 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
Reserved	Bit15	R/W	保留
VDDSEL	Bit14	R/W	<b>VDD 选择控制位</b> 1: 外部 LDO 0.9V 0: 内部 LDO 0.9V
BUF_ISEL	Bits13-12	R/W	<b>BIAS 电流选择控制位</b> 00: 0nA 01: 50nA 10: 100nA 11: 150nA
ISEL	Bits11-8	R/W	<b>电流控制位</b> 0000: MIN (0nA) ..... 1111: MAX (187.5nA) 注: $I=12.5nA*ISEL$
CAPSEL	Bits7- 0	R/W	<b>电容选择控制位</b> 00000: frequency max ..... 11111: frequency min

### 10.5.2.26 LRC时钟用户校准模式寄存器 1 (CMU\_LRCUMC1)

LRC 时钟用户校准模式寄存器 1 (CMU_LRCUMC1)																															
偏移地址: 0E8 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_10000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST1																Reserved		LDO_CAL													

UMCST1	Bits31-16	R/W	<b>LRC用户校准模式使能位</b> 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
Reserved	Bits15-14	R/W	保留
LDO_CAL	Bits13- 0	R/W	<b>LDO 输出电压选择控制位</b> 00_0000_0000_0000: 0.50V ..... 10_0000_0000_0000: 1.20V

10.5.2.27 LOSC时钟用户校准模式寄存器0 (CMU\_LOSCUMC0)

LOSC 时钟用户校准模式寄存器 0 (CMU_LOSCUMC0)																															
偏移地址: 0EC <sub>H</sub>																															
复位值: 00000000_00000000_10000010_10001010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST0																IBSEL		AMP		Reserved	HS_LDO			FBRESSEL				CAPSEL		IRESSEL	

UMCST0	Bits31-16	R/W	<b>LOSC</b> 用户校准模式使能位 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
IBSEL	Bits15-14	R/W	<b>IBIAS</b> 电流控制位 00: 17nA 01: 21nA 10: 25nA 11: 33nA
AMP	Bits13-12	R/W	驱动电流控制 00: min 11: max
Reserved	Bit11	R/W	保留
HS_LDO	Bits10-8	R/W	<b>BUF</b> 电流控制位 001: 50nA 010: 100nA 100: 200nA
FBRESSEL	Bits7-4	R/W	反馈电阻选择 0000 : 320k, 1000 : 8M 1111 : 15.3M
CAPSEL	Bits3-2	R/W	晶振电容选择 00: 6pf, 10: 10pf 11: 14pf
IRESSEL	Bits1- 0	R/W	晶振电流控制 00: 360K 10: 240K 01: 120K 11: 120k

10.5.2.28 LOSC时钟用户校准模式寄存器 1 (CMU\_LOSCUMC1)

LOSC 时钟用户校准模式寄存器 1 (CMU_LOSCUMC1)																															
偏移地址: 0F0 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00011111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST1																Reserved										VSEL					

UMCST1	Bits31-16	R/W	<b>LOSC用户校准模式使能位</b> 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
Reserved	Bits15-5	R/W	保留
VSEL	Bits4-0	R/W	<b>内部 LDO 输出供电选择控制位</b> 00000: min vr=1.367 ..... 11111: max vr=2.0v

### 10.5.2.29 HOSC时钟用户校准模式寄存器 (CMU\_HOSCUMC)

HOSC 时钟用户校准模式寄存器 (CMU_HOSCUMC)																															
偏移地址: 0F4 <sub>H</sub>																															
复位值: 00000000_00000000_00001000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMCST																Reserved				CAPCAL								BIAS			

UMCST	Bits31-16	R/W	<b>LOSC</b> 用户校准模式使能位 0x6996: 用户校准模式使能 其它: 用户校准模式禁止
Reserved	Bits15-12	R/W	保留
CAPCAL	Bits11-4	R/W	<b>XOSC</b> 晶振频率调校控制位 0000_0000: 最低频率 ..... 1111_1111: 最高频率
BIAS	Bits 3-0	R/W	<b>XOSC</b> 外部晶振电流配置位 0000: min ..... 1111: max



## 第11章 DMA控制器(DMA)

### 11.1 概述

DMA 控制器可独立于 CPU 对内部存储进行操作，利用 DMA 可很好的减轻 CPU 的负担并且节省功耗。每个 DMA 通道可选择芯片上的任意外设资源，实现数据的搬运、传输及控制。

### 11.2 特性

该模块的控制逻辑提供以下特性：

- ◇ 支持 12 个独立的 DMA 通道
- ◇ 仲裁到达的请求
- ◇ 指示有效通道
- ◇ 指示通道完成
- ◇ 指示当 AHB-Lite 接口上发生错误
- ◇ 使能低速外设，用来拖延 DMA 周期
- ◇ 完成一个 DMA 周期前，等待清零请求
- ◇ 进行多个或单个 DMA 传输
- ◇ 进行以下不同类型的 DMA 传输
  - ◇ 存储器到存储器
  - ◇ 存储器到外设
  - ◇ 外设到存储器

注：因每个通道只提供单个 DMA 请求接口，因此不支持外设到外设传输。

### 11.3 结构框图

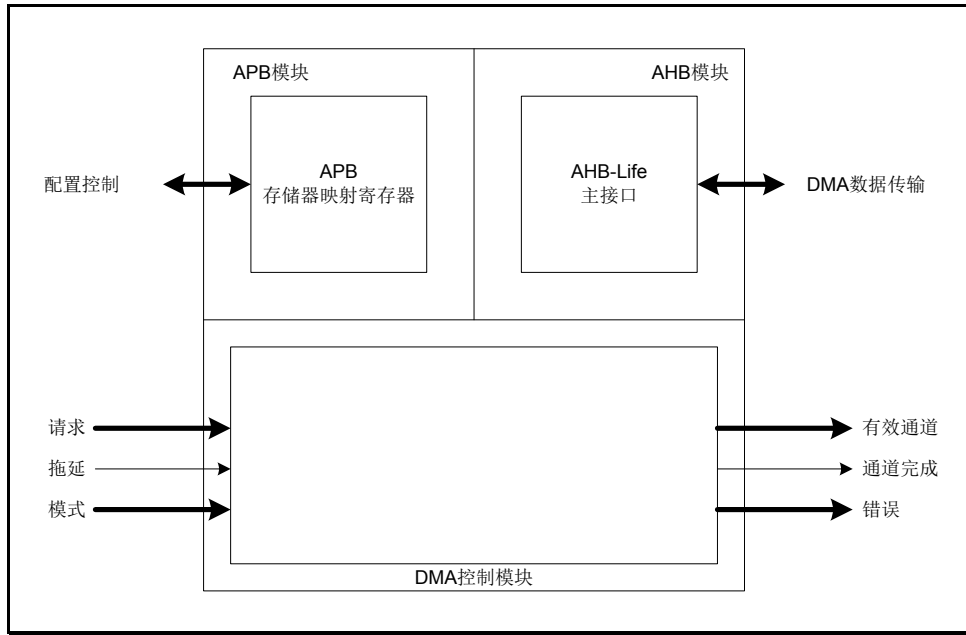


图 11-1 DMA 结构框图

### 11.4 功能描述

#### 11.4.1 通道选择配置

通道选择模块可以选择要连接到每个 DMA 通道的外设请求线路。

该配置是由软件通过控制寄存器 DMA\_CH0\_SELCON-DMA\_CH11\_SELCON 完成的。这些控制寄存器都是由 MSEL 和 MSIGSEL 位组成的。MSEL 位选择要监听的外设，MSIGSEL 位从所选外设中选择要使用的输出信号。

#### 11.4.2 DMA控制

##### 11.4.2.1 DMA仲裁率

在 DMA 传输过程中，用户可配置控制器何时进行仲裁。该动作可缩短响应高优先通道的延时。

控制器提供 4 个比特位，R<sub>power</sub>，用来配置在重新进行仲裁前，AHB 总线传输的次数。其中 R 为 2 的次方，2<sup>R</sup>，决定了仲裁率。当 R = 4，则仲裁率为 2<sup>4</sup> = 16，即每进行 16 次 DMA 传输就进行一次仲裁。

R <sub>power</sub>	x 次 DMA 传输后进行仲裁
b0000	x = 1
b0001	x = 2
b0010	x = 4
b0011	x = 8
b0100	x = 16

R_power	x 次 DMA 传输后进行仲裁
b0101	x = 32
b0110	x = 64
b0111	x = 128
b1000	x = 256
b1001	x = 512
b1010-b1111	x = 1024

表 11-1 仲裁设置

注: 不要给优先级低的通道赋一个较大的 R\_power 值, 这样会导致在重新进行仲裁前, 控制器不会响应优先级高的请求。

需要完成的 DMA 传输次数  $N$  是由用户指定的。当  $N > 2^R$  且  $N$  不是  $2^R$  的整数倍, 控制器会依次先完成  $2^R$  次传输直到剩余的次数  $N$  小于  $2^R$ 。剩余的次数  $N$  会在 DMA 周期的最后完成。用户可将 R\_power 的值保存在通道控制数据结构中。

### 11.4.2.2 优先级

当控制器进行仲裁的时候，下一个进行仲裁的通道可由以下信息决定：

- 通道数
- 通道优先级

每个通道都可以通过 CHNL\_PRIORITY\_SET 寄存器设置为默认优先级或者高优先级。

通道 0 的优先级最高。随着通道数增高，优先级随之降低。

通道数	优先级设置	优先级(由高到低)
<b>0</b>	高	最高优先级
<b>1</b>	高	-
<b>2</b>	高	-
-	高	-
-	高	-
-	高	-
<b>30</b>	高	-
<b>31</b>	高	-
<b>0</b>	默认	-
<b>1</b>	默认	-
<b>2</b>	默认	-
-	默认	-
-	默认	-
-	默认	-
<b>30</b>	默认	-
<b>31</b>	默认	最低优先级

表 11-2 DMA 通道优先级

当完成一个 DMA 传输，控制器将轮询所有的 DMA 通道。下方为轮询流程图。

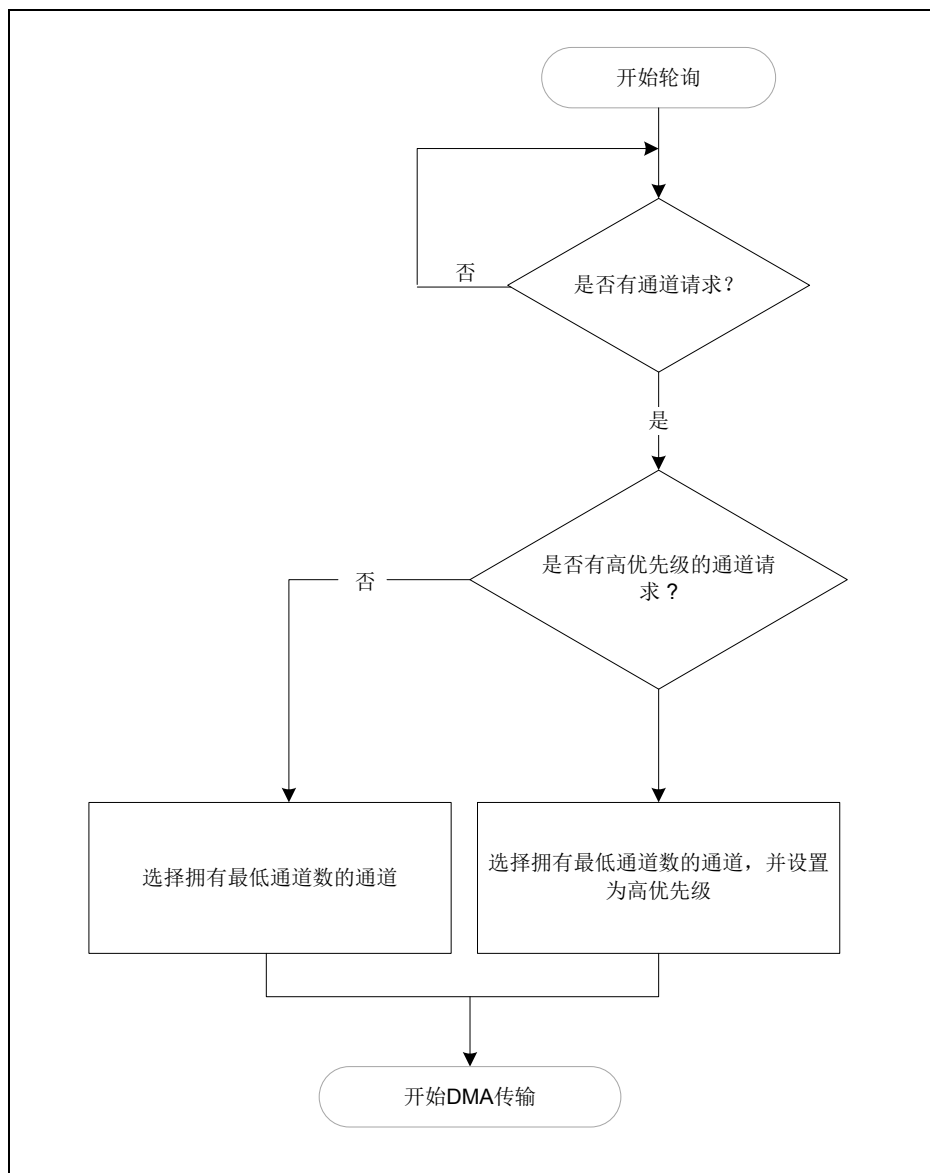


图 11-2 轮询流程图

### 11.4.2.3 DMA周期类型

控制器可通过 `cycle_ctrl` 来选择 DMA 周期的类型，请参考下表。

cycle_ctrl	功能描述
b000	通道控制数据结构无效
b001	基础 DMA 传输
b010	自动请求
b011	乒乓
b100	存储器分散-聚集(主要数据结构)
b101	存储器分散聚集(交替数据结构)
b110	外设分散-聚集(主要数据结构)
b111	外设分散聚集(交替数据结构)

表 11-3 DMA 周期类型

注: `cycle_ctrl` 位于 `channel_cfg` 存储地址。

在所有的周期模式下，控制器在完成  $2^R$  次传输后进行仲裁。若一个低优先级通道被赋予了一个较大的  $2^R$  值，则到该通道完成 DMA 传输之前，所有其它的通道都不会进行 DMA 传输。因此，用户需谨慎设置 `R_power` 的值，避免增加高优先级通道的延时。

#### 无效模式

当完成一个 DMA 传输后，控制器会将周期类型设置为无效，避免控制器发送同一个 DMA 周期。

#### 基础模式

在此模式下，用户可以设置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到请求信号，接下来的流程如下

1. 控制器展开  $2^R$  次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器仲裁如下：
  - 若高优先级通道有请求信号，则控制器先响应该通道
  - 若外设或者软件发出请求信号，则继续进行步骤 1
3. 控制器将 `dma_done[C]` 置高一个 `hclk` 周期，向主控处理器表明该 DMA 周期已完成。

#### 自动请求

在自动请求模式下，仅需要收到一次请求信号就可以完成整个 DMA 周期。无须大幅度增加回应高优先级请求的延时，也无须多次向处理器或者外设发出请求，就可以完成大数据的传输。

用户可以配置控制器使用主要数据结构或者交替数据结构。当通道使能后且控制器接收到该通道上的请求，则流程如下：

1. 控制器展开  $2^R$  次传输。如果剩余的传输次数为 0，则继续步骤 3。
2. 控制器进行仲裁。当通道 C 的优先级最高，则继续进行步骤 1。
3. 控制器将 `dma_done[C]` 置高一个 `hclk` 周期，向主控处理器表明该 DMA 周期已完成。

### 乒乓

在乒乓模式下，控制器先使用其中一种数据结构完成一个 DMA 周期，接着再使用另外一种数据结构完成一次 DMA 周期。控制器将会继续转换这两种数据结构直到读到无效的数据结构或者通道被主控处理器禁止。

下图显示为在乒乓模式下的 DMA 传输。

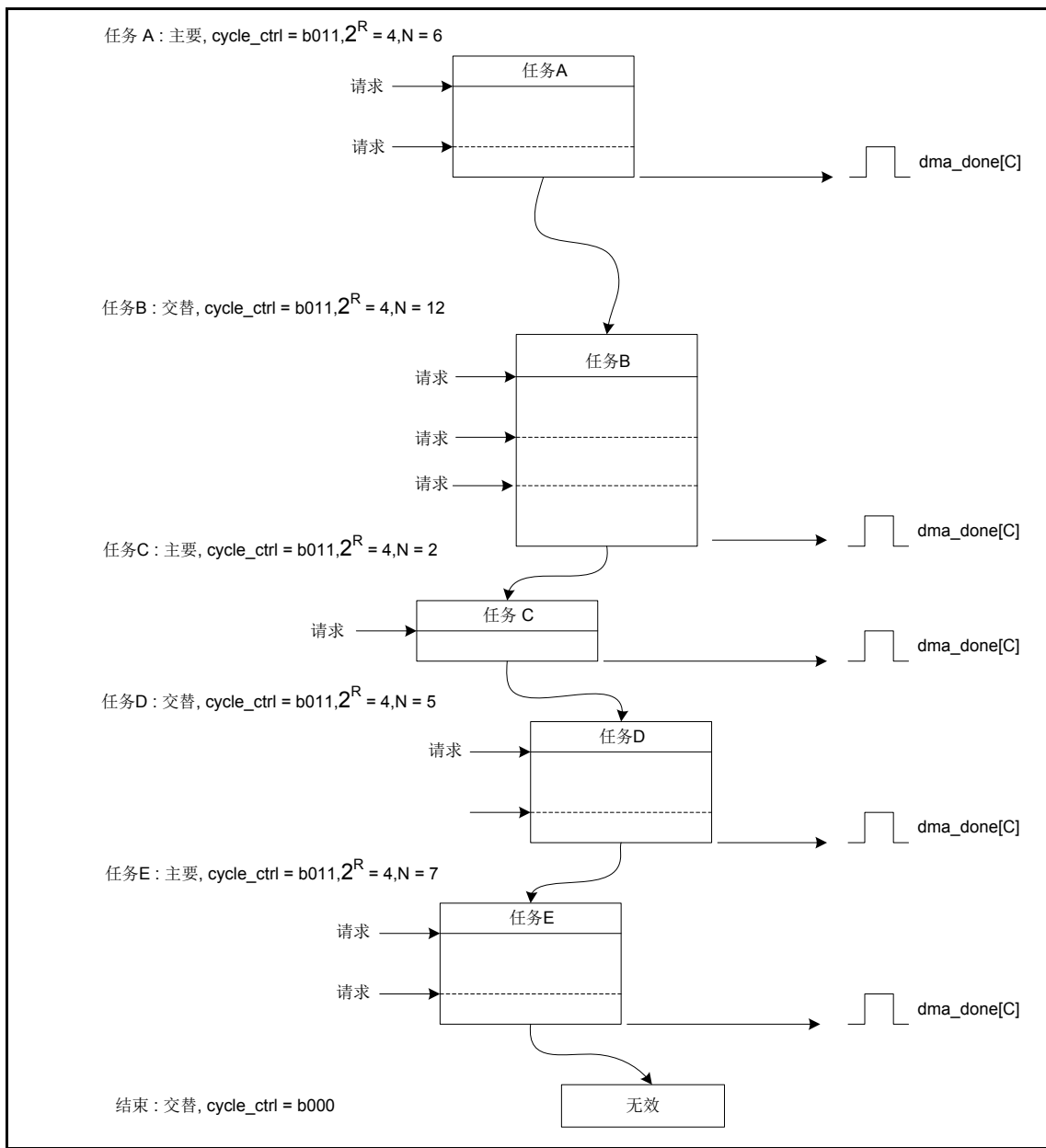


图 11-3 乒乓示例

**任务 A**

1. 主控处理器配置任务 A 为主要数据结构。
2. 主控处理器配置任务 B 为交替数据结构。当任务 A 完成时，控制器会立刻转换去任务 B，前提是没有高优先级通道提出请求。
3. 控制器接收到一个请求并完成 4 次 DMA 传输。
4. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
5. 控制器进行剩余的 2 次 DMA 传输。
6. 控制器将 dma\_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 A 完成后，主控处理器可配置任务 C 为主要数据结构。在任务 B 完成之后，控制器可立即转换去任务 C，前提是没有高优先级通道提出请求。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 B 开始执行：

**任务 B**

7. 控制器进行 4 次 DMA 传输。
8. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
9. 控制器进行 4 次 DMA 传输。
10. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
11. 控制器进行剩余的 4 次 DMA 传输。
12. 控制器将 dma\_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 B 完成后，主控处理器可将任务 D 配置为交替数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 C 开始执行：

**任务 C**

13. 控制器进行 2 次 DMA 传输。
14. 控制器将 dma\_done[C]置高一个 hclk 周期，并进入仲裁流程。

当任务 C 完成后，主控处理器可将任务 E 配置为主要数据结构。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 D 开始执行：

**任务 D**

15. 控制器进行 4 次 DMA 传输。
16. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。
17. 控制器展开剩余的 DMA 传输。



18. 控制器将 dma\_done[C]置高一个 hclk 周期，并进入仲裁流程。

当控制器接收到该通道上新的请求且拥有最高优先级，则任务 E 开始执行：

### 任务 E

19. 控制器进行 4 次 DMA 传输。

20. 控制器进行仲裁。当控制器接收到该通道上的请求，如果该通道具有最高优先级，则继续流程。

21. 控制器进行剩余的 3 次 DMA 传输。

22. 控制器将 dma\_done[C]置高一个 hclk 周期，并进入仲裁流程。

若控制器接收到该通道上的新请求且拥有最高优先级，则控制器会企图开始下一个任务。但由于主控处理器尚未配置交替数据结构，且当任务 D 完成的时候 cycle\_ctrl 被设置为 b000，则会终止乒乓 DMA 传输。

注：用户可通过设置 cycle\_ctrl 为 3' b001 将任务 E 配置为基础 DMA 周期，以用来终止乒乓 DMA 周期。

### 存储器分散-聚集

在存储器分散-聚集模式下，控制器先接收到一个初始请求，接着采用主要数据结构进行 4 次 DMA 传输。当传输完成时，会以交替数据结构开始一个新的 DMA 周期。当该周期完成时，控制器将会采用主要数据结构开始新一轮 4 次 DMA 传输。若发生以下任意一种情况，则控制器将停止主要数据结构和交替数据结构的转换：

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注：当完成 N 次主要数据结构的传输后，可设置 cycle\_ctrl 为 b000 使其变为无效的数据结构。

当完成一个基础周期的分散-聚集模式传输，控制器仅将 dma\_done[C]置为有效。

在分散-聚集模式下，控制器利用主要数据结构来编程交替数据结构。下表主要数据结构的 channel\_cfg 配置，分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31:30]	dst_inc	b10	配置控制器使用字作为地址增量
[29:28]	dst_size	b10	配置控制器使用字传输
[27:26]	src_int	b10	配置控制器使用字作为地址增量
[25:24]	src_size	b10	配置控制器使用字传输
[17:14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[3]	next_usebusrt	0	当配置为存储器分散-聚集模式时，该位必须设置为 0
[2:0]	cycle_ctrl	b100	配置控制器进行分散-聚集 DMA 周期
用户定义			

位	域	值	功能描述
[23:21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20:18]	src_prot_ctrl	-	当写入源数据后，配置 HPROT 的状态
[13:4]	n_minus_1	N <sup>a</sup>	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数

表 11-4 存储器分散-聚集数据结构

注 a: 由于 R<sub>power</sub> 设计为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为存储器分散-聚集模式的示例:

初始化:

1. 配置主要数据结构来使能复制 A, B, C 的操作:  $\text{cycle\_ctrl} = \text{b}100, 2^R = 4, N = 16$
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b101, 2 <sup>R</sup> = 4, N = 3	0XXXXXXXXXX
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b101, 2 <sup>R</sup> = 2, N = 8	0XXXXXXXXXX
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b101, 2 <sup>R</sup> = 8, N = 5	0XXXXXXXXXX
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b001, 2 <sup>R</sup> = 4, N = 4	0XXXXXXXXXX

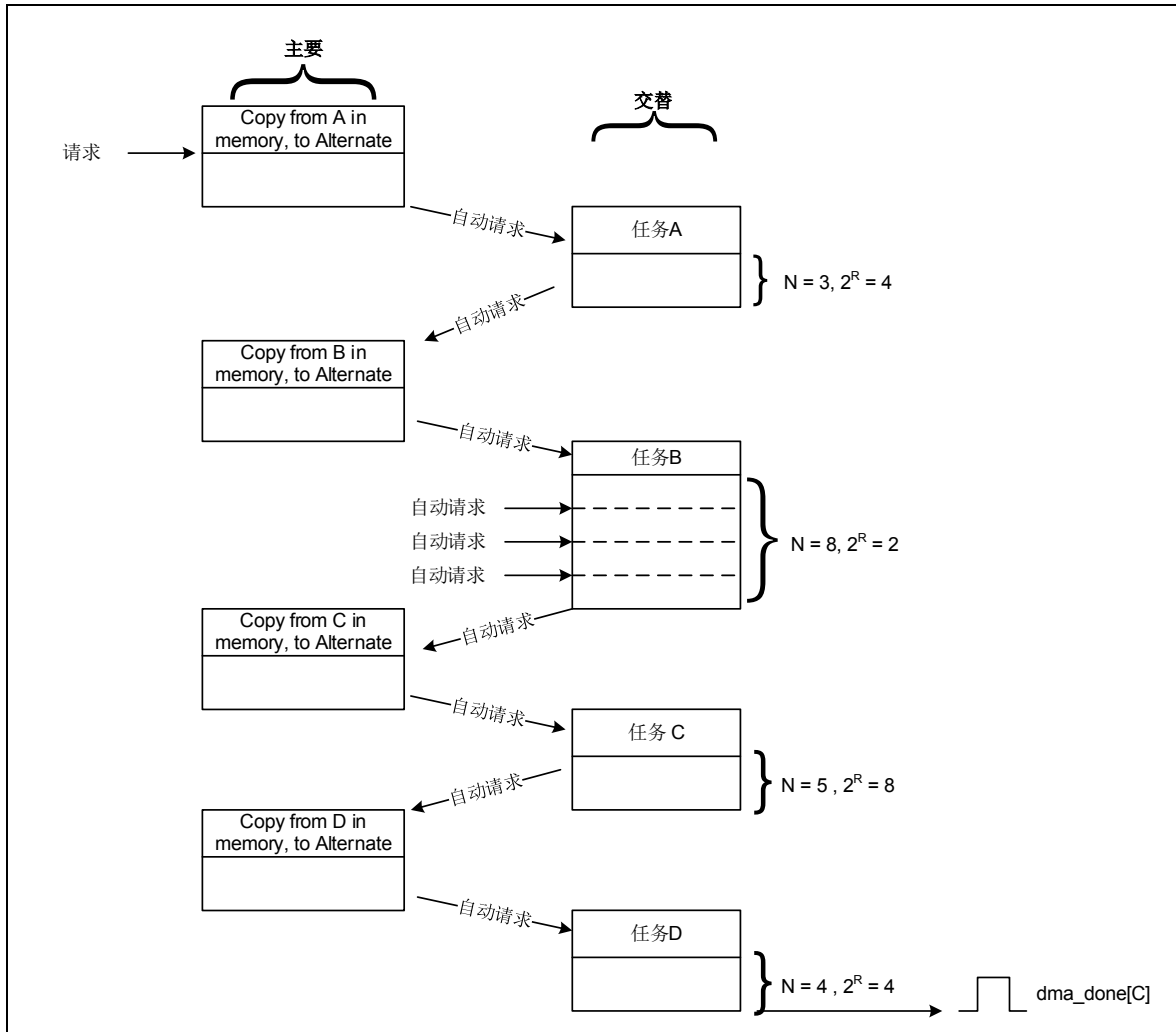


图 11-4 存储器分散-聚集示例

### 初始化

1. 主控处理器通过设置 `cycle_ctrl` 为 `b100`，使主要数据结构运行于存储器分散-聚集模式。由于单个通道的数据结构包含 4 个字，所以  $2^R$  必须设置为 4。在该示例中，有 4 个任务，因此 N 设为 16。
2. 主控处理器将任务 A, B, C 和 D 的数据结构写入由主要 `src_data_end_ptr` 指定的存储器地址中。
3. 主控处理器使能该通道。

当控制器接收到 `dma_req[C]` 请求或者来自主控处理器的手动请求，则存储器分散-聚集传输开始执行，流程如下：

### 主要，复制 A

1. 在接收到请求后，控制器进行 4 次 DMA 传输，并且任务 A 写为交替数据结构。
2. 控制器生成一个自动请求接着进行仲裁。

### 任务 A

3. 控制器进行任务 A。当任务 A 完成，控制器生成一个自动请求，接着进行仲裁。

**主要, 复制 B**

4. 控制器进行 4 次 DMA 传输, 且任务 B 写为交替数据结构。
5. 控制器生成一个自动请求接着进行仲裁。

**任务 B**

6. 控制器进行任务 B。当任务 B 完成, 控制器生成一个自动请求, 接着进行仲裁。

**主要, 复制 C**

7. 控制器进行 4 次 DMA 传输, 且任务 C 写为交替数据结构。
8. 控制器生成一个自动请求接着进行仲裁。

**任务 C**

9. 控制器进行任务 C。当任务 C 完成, 控制器生成一个自动请求, 接着进行仲裁。

**主要, 复制 D**

10. 控制器进行 4 次 DMA 传输, 且任务 D 写为交替数据结构。
11. 控制器设置主要数据结构的 `cycle_ctrl` 为 `b000`, 表明该数据结构为无效。
12. 控制器生成一个自动请求接着进行仲裁。

**任务 D**

13. 控制器使用基础 DMA 周期执行任务 D。
14. 控制器将 `dma_done[C]`置高一个 `hclk` 周期, 并进入仲裁流程。

**外设分散-聚集**

在外设分散-聚集模式下, 控制器接收到一个来自外设的初始请求, 接着使用主要数据结构执行 4 次 DMA 传输, 然后立即使用交替数据结构启动新的 DMA 周期, 无须重新仲裁也无须将 `dma_active[C]`置为低电平。

注: 仅在该状况下, 当完成主要数据结构的传输后, 控制器无须进入仲裁流程。

在该 DMA 周期完成后, 控制器重新仲裁。如果控制器接收到外设请求且拥有最高优先级, 则执行 4 次主要数据结构的 DMA 传输。接着立即启动交替数据结构的 DMA 周期, 无须重新仲裁也无须将 `dma_active[C]`置为低电平。若发生以下任意一种情况, 则控制器将停止主要数据结构和交替数据结构的转换:

- 主控处理器将交替数据结构配置为一个基础周期
- 读取到一个无效数据结构

注: 当完成 N 次主要数据结构的传输后, 可设置 `cycle_ctrl` 为 `b000` 使其变为无效的数据结构。

当完成一个基础周期的分散-聚集模式传输, 控制器将 `dma_done[C]`置为有效。

在分散-聚集模式下, 控制器利用主要数据结构来编程交替数据结构。下表主要数据结构的 `channel_cfg` 配置, 分为固定值配置和用户配置。

位	域	值	功能描述
固定值			
[31:30]	dst_inc	b10	配置控制器使用字作为地址增量
[29:28]	dst_size	b10	配置控制器使用字传输
[27:26]	src_int	b10	配置控制器使用字作为地址增量
[25:24]	src_size	b10	配置控制器使用字传输
[17:14]	R_power	b1101	配置控制器进行 4 次 DMA 传输
[2:0]	cycle_ctrl	b100	配置控制器进行外设分散-聚集 DMA 周期
用户定义			
[23:21]	dst_prot_ctrl	-	当写入目标数据后，配置 HPROT 的状态
[20:18]	src_prot_ctrl	-	当写入源数据后，配置 HPROT 的状态
[13:4]	n_minus_1	N <sup>a</sup>	配置控制器进行 N 次 DMA 传输，其中 N 为 4 的倍数
[3]	next_usebusrt	-	当设置为 1，在交替传输完成后，控制器会将 CHNL_USEBURST_SET[C]置 1。

表 11-5 外设分散-聚集数据结构

注 a: 由于 R\_power 设计为 4，因此 N 必须设置为 4 的倍数。N/4 的值为配置交替数据结构的次数。

下图为外设分散-聚集模式的示例:

初始化:

1. 配置主要数据结构来使能复制 A, B, C 的操作:  $\text{cycle\_ctrl} = \text{b}110, 2^R = 4, N = 16$
2. 利用下表中的结构，将主要源数据写入存储器。

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
任务 A 数据	0x0A000000	0x0AE00000	Cycle_ctrl = b101, $2^R = 4, N = 3$	0xFFFFFFFF
任务 B 数据	0x0B000000	0x0BE00000	Cycle_ctrl = b101, $2^R = 2, N = 8$	0xFFFFFFFF
任务 C 数据	0x0C000000	0x0CE00000	Cycle_ctrl = b101, $2^R = 8, N = 5$	0xFFFFFFFF
任务 D 数据	0x0D000000	0x0DE00000	Cycle_ctrl = b001, $2^R = 4, N = 4$	0xFFFFFFFF

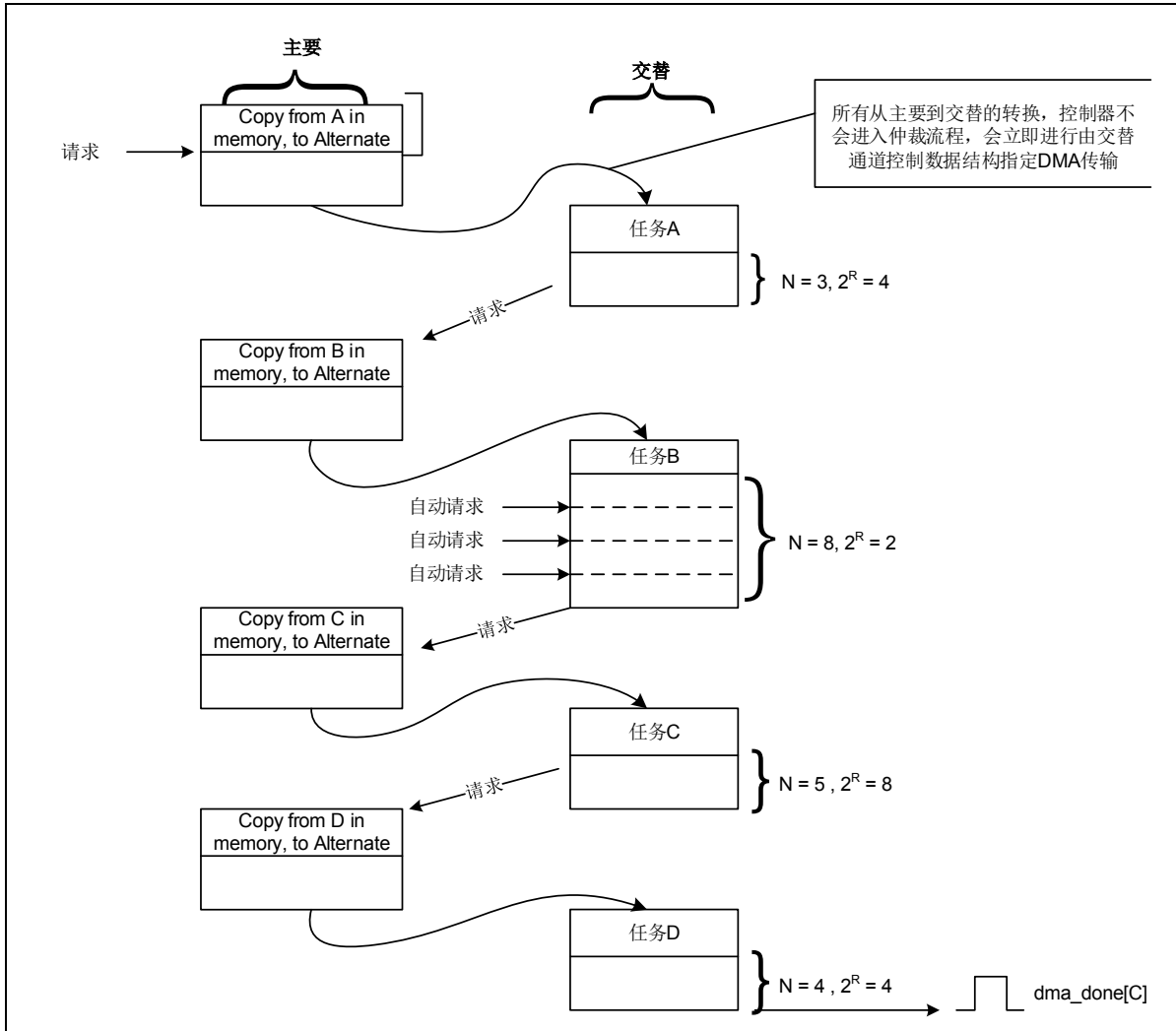


图 11-5 外设交替-聚集示例

### 初始化

1. 主控处理器通过设置 `cycle_ctrl` 为 `b110`, 使主要数据结构运行于外设分散-聚集模式。由于单个通道的数据结构包含 4 个字, 所以  $2^R$  必须设置为 4。在该示例中, 有 4 个任务, 因此  $N$  设为 16。
2. 主控处理器将任务 A, B, C 和 D 的数据结构写入由主要 `src_data_end_ptr` 指定的存储器地址中。
3. 主控处理器使能该通道。

当控制器接收到 `dma_req[C]` 请求, 则外设分散-聚集传输开始执行, 流程如下:

### 主要, 复制 A

1. 在接收到请求后, 控制器进行 4 次 DMA 传输, 并且任务 A 写为交替数据结构。

### 任务 A

2. 控制器进行任务 A。
3. 当完成任务 A 后, 控制器进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

#### 主要, 复制 B

4. 控制器进行 4 次 DMA 传输，且任务 B 写为交替数据结构。

#### 任务 B

5. 控制器执行 4 次 DMA 传输。为使控制器完成该任务，外设还必须再发出 3 次请求。

6. 当任务 B 完成，进度仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

#### 主要, 复制 C

7. 控制器进行 4 次 DMA 传输，且任务 C 写为交替数据结构。

#### 任务 C

8. 控制器执行任务 C。

9. 当任务 C 完成，进入仲裁流程。

在外设发出一个新的请求且拥有最高优先级，则继续以下流程：

#### 主要, 复制 D

10. 控制器进行 4 次 DMA 传输，且任务 D 写为交替数据结构。

11. 控制器设置主要数据结构的 `cycle_ctrl` 为 `b000`，表明该数据结构为无效。

#### 任务 D

12. 控制器使用基础 DMA 周期执行任务 D。

13. 控制器将 `dma_done[C]` 置高一个 `hclk` 周期，并进入仲裁流程。

#### 错误信号

如果控制器在 AHB-Lite 主机接口上检测到一个错误回应，则：

- ◇ 禁止该错误信号对应的通道
- ◇ 置 `dma_err` 为高电平

1. 读取 `CHNL_ENABLE_SET` 寄存器，获取一张禁止通道的列表。

当某通道将 `dma_done[C]` 置为有效，则控制器将禁止该通道。主控处理器上的程序必须记录哪些通道曾置高 `dma_done[C]` 输出。

2. 将步骤 1 中得到的列表和主程序上的记录作比较。若某通道没有 `dma_done[C]` 被置高的记录，则就是该通道发生了错误信号。

### 11.4.3 通道控制数据结构

用户必须提供系统存储器空间用来包含通道控制数据结构。该系统存储器必须:

- ◇ 提供一个连续的存储空间，以便控制器和主控处理器可以访问
- ◇ 其基地址为通道控制数据结构总容量的整数倍

下图为当使用 16 个通道和交替数据结构时，控制器所需要的通道控制数据的存储器映射图。

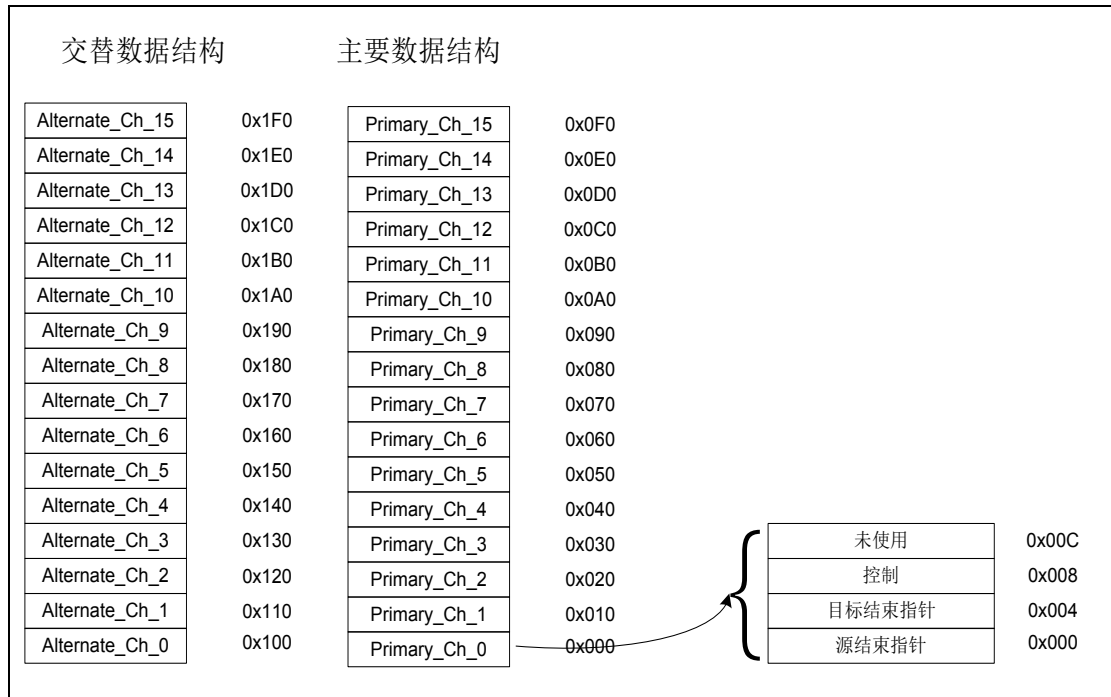


图 11-6 16 通道存储器映射（包括交替数据结构）

图中的通道数据结构使用了 448Bytes 的系统存储器。该示例中，控制器用地址低 9 位来访问数据结构中的所有地址，因此基地址必须为 0XXXXXX000，0XXXXXX200，0XXXXXX400 或者 0XXXXXX800。

通过准确设置 CTRL\_BASE\_PTR 寄存器，用户可配置主要数据结构的基地址。

控制器利用系统存储器来访问两个指针及每个通道的控制信息。以下内容详细描述了 32 位存储地址及 DMA 传输地址的计算方法。

- ◇ 源数据结束指针
- ◇ 目标数据结束指针
- ◇ 控制数据配置
- ◇ 地址计算

#### 11.4.3.1 源数据结束指针

src\_data\_end\_ptr 存储地址包含一个指针，指向源数据的最后一个地址。

在执行 DMA 传输前，该存储地址写入源数据的结束地址。当启动 2<sup>R</sup> 次 DMA 传输时，控制器读取 src\_data\_end\_ptr。注意控制器不能写该存储器地址。



### 11.4.3.2 目标数据结束指针

dst\_data\_end\_ptr 存储地址包含一个指针，指向目标数据的最后一个地址。

在执行 DMA 传输前，该存储地址写入目标数据的结束地址。当启动  $2^R$  次 DMA 传输时，控制器读取 dst\_data\_end\_ptr。注意控制器不能写该存储器地址。

### 11.4.3.3 控制数据配置

channel\_cfg 会向控制器提供每一次 DMA 传输的控制信息。

通道配置 (channel_cfg)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dst_inc		dst_size		src_inc		src_size		dst_prot_ctrl		src_prot_ctrl		R_power		n_minus_1												next_useburst	cycle_ctrl				

dst_inc	Bit 31-30	R/W	<p><b>目标地址增量</b> 地址增量取决于源数据的宽度： 源数据宽度 = 字节 b00 = 字节 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。</p> <p>源数据宽度 = 半字 b00 = 保留 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。</p> <p>源数据宽度 = 字 b00 = 保留 b01 = 保留 b10 = 字 b11 = 无增量。地址仍然是 dst_data_end_ptr 中包含的地址。</p>
dst_size	Bit 29-28	R/W	<p><b>目标数据大小</b> 需要注意的是 dst_size 和 src_size 的值必须一致。</p>
src_inc	Bit 27-26	R/W	<p><b>源地址增量</b> 地址增量取决于源数据的宽度： 源数据宽度 = 字节 b00 = 字节</p>

			<p>b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。 源数据宽度 = 半字 b00 = 保留 b01 = 半字 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。 源数据宽度 = 字 b00 = 保留 b01 = 保留 b10 = 字 b11 = 无增量。地址仍然是 src_data_end_ptr 中包含的地址。</p>
src_size	Bit 25-24	R/W	<p>设置该位段用来匹配源数据大小 b00 = 字节 b01 = 半字 b10 = 字 b11 = 保留</p>
dst_prot_ctrl	Bit 23-21	R/W	<p>当控制器写目标数据的时候，HPROT 状态控制位 bit[23] 该位对 DMA 无效 bit[22] 该位对 DMA 无效 bit[21] HPROT 状态控制位  0=HPROT 为低，访问为无特权模式 1=HPROT 为高，访问为特权模式</p>
src_prot_ctrl	Bit 20-18	R/W	<p>当控制器读取源数据的时候，HPROT 状态控制位 bit[20] 该位对 DMA 无效 bit[19] 该位对 DMA 无效 bit[18] HPROT 状态控制位  0=HPROT 为低，访问为无特权模式 1=HPROT 为高，访问为特权模式</p>
R_power	Bit 17-14	R/W	<p>在控制器重新仲裁前，该位段决定了 DMA 传输可以发生的次数 b0000: 发生 1 次 DMA 传输后仲裁 b0001: 发生 2 次 DMA 传输后仲裁 b0010: 发生 4 次 DMA 传输后仲裁 b0011: 发生 8 次 DMA 传输后仲裁 b0100: 发生 16 次 DMA 传输后仲裁</p>

			<p>b0101: 发生 32 次 DMA 传输后仲裁          b0110: 发生 64 次 DMA 传输后仲裁          b0111: 发生 128 次 DMA 传输后仲裁          b1000: 发生 256 次 DMA 传输后仲裁          b1001: 发生 512 次 DMA 传输后仲裁          b1010-b1111: 发生 1024 次 DMA 传输后仲裁。由于最大的传输次数为 1024, 由此表明 DMA 传输中无仲裁发生。</p>
n_minus_1	Bit 13-4	R/W	<p>在 DMA 周期开始前, 该位段代表的是 DMA 周期中包含的 DMA 传输的总次数。用户须根据所需要的 DMA 周期的大小来设置该位段。该 10 位值为 DMA 传输次数减 1。</p> <p>b000000000: 1 次 DMA 传输          b000000001: 2 次 DMA 传输          b000000010: 3 次 DMA 传输          b000000011: 4 次 DMA 传输          b000000100: 5 次 DMA 传输          .....          b111111111: 1024 次 DMA 传输</p> <p>在进入仲裁流程前, 控制器会立即更新该位段, 可使控制器保存该 DMA 周期中还需完成的剩下的 DMA 传输次数。</p>
next_useburst	Bit 3	R/W	<p>当控制器在外设分散-聚集模式下, 且使用交替数据结构, 该位段控制 CHNL_USEBURST_SET[C] 是否置 1。</p> <p>需要注意的是, 在完成由交替数据结构指定的 DMA 周期前, 如果剩余的传输次数小于 <math>2^R</math>, 控制器会将 CHNL_USEBURST_SET[C] 设置为 0。          Next_useburst 的设定控制了是否需要再次修改 CHNL_USEBURST_SET[C]。</p> <p>在外设分散-聚集 DMA 周期模式下, 当使用交替数据结构的 DMA 周期完成后, 会发生以下任一情况:</p> <p>0: 控制器不改变 CHNL_USEBURST_SET[C] 的值。当进行使用交替数据结构的 DMA 周期时, 如果 CHNL_USEBURST_SET[C] 为 0, 则对于所有在外设分散-聚集模式下的剩余的 DMA 周期, 控制器将回应 dma_req[ ] 和 dma_sreq[ ] 的请求。</p> <p>1: 控制器将 CHNL_USEBURST_SET[C] 置 1。因此, 当进行使用交替数据结构的 DMA 周期时, 对于所有在外设分散-聚集模式下的剩余的 DMA 周期, 控制器仅回应 dma_req[ ] 的请求。</p>

<p>cycle_ctrl</p>	<p>Bit 2-0</p>	<p>RW</p>	<p><b>DMA 周期的工作模式:</b></p> <p><b>b000:</b> 停止。表明该数据结构无效。</p> <p><b>b001:</b> 基础。在进入仲裁流程前, 控制器必须接收到一个新的请求才能完成 DMA 周期。</p> <p><b>b010:</b> 自动请求。在仲裁过程中, 控制器在合适的通道上自动插入一个请求。这就意味着最初的请求已足够完成 DMA 周期。</p> <p><b>b011:</b> 乒乓。控制器使用一种数据结构完成一个 DMA 周期。当该周期完成后, 控制器使用另一种数据结构完成一个 DMA 周期。在该周期完成后, 且如果主控处理器已更新了原始数据结构, 则控制器使用原始数据结构执行一次 DMA 周期。控制器将会继续进行 DMA 周期, 直到读取到无效数据结构或者主控处理器将 cycle_ctrl 改为 b001 或 b010。</p> <p><b>b100:</b> 存储器分散-聚集。当控制器运行在存储器分散-聚集模式下, 用户仅可在主要数据结构下使用该值。</p> <p><b>b101:</b> 存储器分散-聚集。当控制器运行在存储器分散-聚集模式下, 用户仅可在交替数据结构下使用该值。</p> <p><b>b110:</b> 外设分散-聚集。当控制器运行在外设分散-聚集模式下, 用户仅可在主要数据结构下使用该值。</p> <p><b>b111:</b> 外设分散-聚集。当控制器运行在外设分散-聚集模式下, 用户仅可在交替数据结构下使用该值。</p>
-------------------	----------------	-----------	---

表 11-6 控制信息格式描述

在 DMA 周期或者  $2^R$  DMA 传输开始的时候, 控制器会从系统存储器中获取 channel\_cfg 的值。当完成  $2^R$  或 N 次传输后, 新的 channel\_cfg 值会被存储到系统存储器中。

控制器不支持 dst\_size 和 src\_size 拥有两个不同的值。如果检测到两个值不匹配, src\_size 的值会被作为源数据和目标数据的大小。当下一次 n\_minus\_1 更新时, dst\_size 将被设置为 src\_size 的值。

当完成 N 次传输后, 控制器会将 cycle\_ctrl 设置为 b000, 以此表明 channel\_cfg 数据为无效, 用来防止控制器重复相同的 DMA 传输。

#### 11.4.3.4 地址计算

为了计算 DMA 传输的源地址, 控制器必须将 n\_minus\_1 的值向左移, 移动值由 src\_inc 定义。接着从源数据结束指针中减去移动后的 n\_minus\_1 的值。同样地, 为了计算 DMA 传输的目标地址, 控制器也必须将 n\_minus\_1 的值向左移, 移动量由 dst\_inc 定义, 接着从目标数据结束指针中减去移动后的 n\_minus\_1 的值。

根据 `src_inc` 和 `dst_inc` 的值，源地址与目标地址可用以下等式计算：

`Src_inc = b00, dst_inc = b00`

源地址 = `src_data_end_ptr - n_minus_1`

目标地址 = `dst_data_end_ptr - n_minus_1`

`Src_inc = b01, dst_inc = b01`

源地址 = `src_data_end_ptr - (n_minus_1<<1)`

目标地址 = `dst_data_end_ptr - (n_minus_1<<1)`

`Src_inc = b10, dst_inc = b10`

源地址 = `src_data_end_ptr - (n_minus_1<<2)`

目标地址 = `dst_data_end_ptr - (n_minus_1<<2)`

`Src_inc = b11, dst_inc = b11`

源地址 = `src_data_end_ptr`

目标地址 = `dst_data_end_ptr`

下表为 6 个字的 DMA 周期，地址增量为 1 个字。

channel_cfg 初始值 (DMA 周期前)				
src_size=b10, dst_inc=b10, n_minus_1=b101, cycle_ctrl=1				
DMA 传输	结束指针	次数	差值	地址
	0x2AC	5	0x14	0x298
	0x2AC	4	0x10	0x29C
	0x2AC	3	0xC	0x2A0
	0x2AC	2	0x8	0x2A4
	0x2AC	1	0x4	0x2A8
	0x2AC	0	0x0	0x2AC
channel_cfg 最终值 (DMA 周期后)				
src_size=b10, dst_inc=b10, n_minus_1=0, cycle_ctrl=0				

表 11-7 传输过程中指针变化

注：上表中的“差值”为“次数”向左移动 dst\_inc 对应位数后的结果。

channel_cfg 初始值 (DMA 周期前)				
src_size=b00, dst_inc=b01, n_minus_1=b1011, cycle_ctrl=1, R_power=b11				
DMA 传输	结束指针	次数	差值	地址
	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
0x5E7	4	0x8	0x5DF	
2 <sup>R</sup> 次 DMA 传输完成后, channel_cfg 的值				
src_size=b00, dst_inc=b01, n_minus_1=b011, cycle_ctrl=1, R_power=b11				
DMA 传输	结束指针	次数	差值	地址
	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
0x5E7	0	0x0	0x5E7	
channel_cfg 最终值 (DMA 周期后)				
src_size=b00, dst_inc=b01, n_minus_1=0, cycle_ctrl=0, R_power=b11				

表 11-8 传输过程中指针变化

注 1：上表中的“差值”为“次数”向左移动 dst\_inc 对应位数后的结果。

注 2：当控制器完成 DMA 周期后，通过将 cycle\_ctrl 清零使 channel\_cfg 无效。

## 11.5 特殊功能寄存器

该章节描述了 uDMAC 寄存器及提供了编程控制器的信息。

- ◇ 编程器模型介绍
- ◇ 寄存器描述

### 11.5.1 编程器模型介绍

下列条件适用于控制器提供的寄存器:

- ◇ 控制器的基地址并非固定不变的，但任何特定寄存器的偏移地址都是固定的。
- ◇ 用户禁止访问任何保留的地址，否则可导致控制器产生不可预期的行为。
- ◇ 除非有其他相关说明，用户必须将保留的和未使用的寄存器位写 0，无视读取值。
- ◇ 除非有其他相关说明，系统复位或者上电复位会将所有寄存器位写 0。
- ◇ 除非有其他相关说明，所有寄存器都支持读写访问。写操作可更新寄存器内容，读操作则返回寄存器内容。

## 11.6 特殊功能寄存器

### 11.6.1 寄存器列表

DMA 寄存器列表			
名称	偏移地址	类型	描述
DMA 基地址: 4000_C000 <sub>H</sub>			
DMA_STATUS	0000 <sub>H</sub>	R	DMA 状态寄存器
DMA_CFG	0004 <sub>H</sub>	W	DMA 配置寄存器
DMA_CTRLBASE	0008 <sub>H</sub>	R/W	DMA 通道控制数据基指针寄存器
DMA_ALTCTRLBASE	000C <sub>H</sub>	R	DMA 通道交替控制数据基指针寄存器
DMA_CHWAITSTATUS	0010 <sub>H</sub>	R	DMA 通道等待请求状态寄存器
DMA_CHSWREQ	0014 <sub>H</sub>	T_W1	DMA 通道软件请求寄存器
DMA_CHUSEBURSTSET	0018 <sub>H</sub>	R/C_W1	DMA 通道使用突发设置寄存器
DMA_CHUSEBURSTCLR	001C <sub>H</sub>	T_W1	DMA 通道使用突发清除寄存器
DMA_CHREQMASKSET	0020 <sub>H</sub>	R/C_W1	DMA 通道请求屏蔽设置寄存器
DMA_CHREQMASKCLR	0024 <sub>H</sub>	T_W1	DMA 通道请求屏蔽清除寄存器
DMA_CHENSET	0028 <sub>H</sub>	R/C_W1	DMA 通道使能设置寄存器
DMA_CHENCLR	002C <sub>H</sub>	T_W1	DMA 通道使能清除寄存器
DMA_CHPRIALTSET	0030 <sub>H</sub>	R/C_W1	DMA 通道主要-交替设置寄存器
DMA_CHPRIALTCLR	0034 <sub>H</sub>	T_W1	DMA 通道主要-交替清除寄存器
DMA_CHPRSET	0038 <sub>H</sub>	R/C_W1	DMA 通道优先级设置寄存器
DMA_CHPRCLR	003C <sub>H</sub>	T_W1	DMA 通道优先级清除寄存器
Reserved	0040 <sub>H</sub> ~0048 <sub>H</sub>	-	保留
DMA_ERRCLR	004C <sub>H</sub>	R/C_W1	DMA 总线错误清除寄存器
Reserved	0050 <sub>H</sub> ~0FFC <sub>H</sub>	-	保留
DMA_IFLAG	1000 <sub>H</sub>	R/C_W1	DMA 中断状态寄存器
Reserved	1004 <sub>H</sub>	-	保留
Reserved	1008 <sub>H</sub>	-	保留
DMA_IER	100C <sub>H</sub>	R/W	DMA 中断使能控制寄存器
Reserved	1010 <sub>H</sub> ~10FC <sub>H</sub>	-	保留
DMA_CH0_SELCON	1100 <sub>H</sub>	R/W	DMA 通道 0 复用选择寄存器
DMA_CH1_SELCON	1104 <sub>H</sub>	R/W	DMA 通道 1 复用选择寄存器
DMA_CH2_SELCON	1108 <sub>H</sub>	R/W	DMA 通道 2 复用选择寄存器
DMA_CH3_SELCON	110C <sub>H</sub>	R/W	DMA 通道 3 复用选择寄存器
DMA_CH4_SELCON	1110 <sub>H</sub>	R/W	DMA 通道 4 复用选择寄存器
DMA_CH5_SELCON	1114 <sub>H</sub>	R/W	DMA 通道 5 复用选择寄存器
DMA_CH6_SELCON	1118 <sub>H</sub>	R/W	DMA 通道 6 复用选择寄存器
DMA_CH7_SELCON	111C <sub>H</sub>	R/W	DMA 通道 7 复用选择寄存器
DMA_CH8_SELCON	1120 <sub>H</sub>	R/W	DMA 通道 8 复用选择寄存器
DMA_CH9_SELCON	1124 <sub>H</sub>	R/W	DMA 通道 9 复用选择寄存器
DMA_CH10_SELCON	1128 <sub>H</sub>	R/W	DMA 通道 10 复用选择寄存器



DMA_CH11_SELCON	112C <sub>H</sub>	R/W	DMA 通道 11 复用选择寄存器
-----------------	-------------------	-----	-------------------

## 11.6.2 寄存器描述

### 11.6.2.1 DMA状态寄存器 (DMA\_STATUS)

该寄存器可返回控制器的状态，为只读寄存器。当控制器处于复位状态时，用户不能读取该寄存器。

DMA 状态寄存器 (DMA_STATUS)																															
偏移地址: 000 <sub>H</sub>																															
复位值: 00010000_00001011_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								STATUS			Reserved		MASTER_ENABLE		

Reserved	Bit 31-8	-	保留
STATUS	Bit 7-4	R	<b>控制状态机的当前状态</b> b0000: 空闲 b0001: 读取通道控制器数据 b0010: 读取源数据结束指针 b0011: 读取目标数据结束指针 b0100: 读取源数据 b0101: 写目标数据 b0110: 等待 DMA 请求清 0 b0111: 写通道控制器数据 b1000: 延迟 b1001: 完成 b1010: 外设分散-聚集转换 b1011-b1111: 未定义
			Reserved
MASTER_ENABLE	Bit 0	R	<b>使能控制器</b> 0: 禁止控制器 1: 使能控制器

### 11.6.2.2 DMA配置寄存器 (DMA\_CFG)

该寄存器为只写寄存器，可对控制器进行配置。

DMA 配置寄存器 (DMA_CFG)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CHNL_PROT_CTRL			Reserved			MASTER_ENABLE	

Reserved	Bit 31-8	-	未定义，写0。
CHNL_PROT_CTRL	Bit 7-5	W	<p>通过控制 HPROT[3:1]的信号电平对 AHB-Lite 做保护设置</p> <p>Bit[7]: 控制 HPROT[3]来表明是否正在发生可缓存访问</p> <p>Bit[6]: 控制 HPROT[2]来表明是否正在发生可缓冲访问</p> <p>Bit[5]: 控制 HPROT[1]来表明是否正在发生特殊权限访问</p> <p>注意: 当 bit[n] =1, 相应 HPROT 为高电平。 当 bit[n] =0, 相应 HPROT 为低电平。</p>
Reserved	Bit 4-1	-	保留
MASTER_ENABLE	Bit 0	W	<p>使能控制器</p> <p>0: 禁止控制器</p> <p>1: 使能控制器</p>

### 11.6.2.3 DMA通道控制数据基指针寄存器 (DMA\_CTRLBASE)

该寄存器为读写寄存器。用户须配置该寄存器用来指向系统存储器中的地址。

需要指出的是，控制器不提供内部存储器用来存储通道控制数据结构。

控制器中用户所需指定的系统存储器的大小取决于以下两个条件: DMA 通道的个数和是否使用交替数据结构。因此，该基指针的设置取决于不同系统实现，可参考下图。

当控制器处于复位状态时，用户不可读取该寄存器。

DMA 通道控制数据基指针寄存器 (DMA_CTRLBASE)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRL_BASE_PTR																								Reserved							

CTRL_BASE_PTR	Bit 31- 9	R/W	主要数据结构基地址指针
Reserved	Bit 8-0	R/W	未定义，写 0。

### 11.6.2.4 DMA通道交替控制数据基指针寄存器 (DMA\_ALTCTRLBASE)

该寄存器为只读寄存器，可返回交替数据结构的基地址，因此用户无需在应用软件中计算交替数据结构的基地址。当控制器处于复位状态时，用户不可读取该寄存器。

DMA 通道交替控制数据基指针寄存器 (DMA_ALTCTRLBASE)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALT_CTRL_BASE_PTR																															

ALT_CTRL_BASE_PTR	Bit 31-0	R	交替数据结构的基地址
-------------------	----------	---	------------

### 11.6.2.5 DMA通道等待请求状态寄存器 (DMA\_CHWAITSTATUS)

该寄存器为只读寄存器，可返回 dma\_waitonreq[C]的状态。当控制器处于复位状态时，用户不可读取该寄存器。Bit0 返回的为 dma\_waitonreq[0]的状态，Bit1 返回的为 dma\_waitonreq[1]的状态，以此类推，Bit31 返回的为 dma\_waitonreq[31]的状态。

DMA 通道等待请求状态寄存器 (DMA_CHWAITSTATUS)																															
偏移地址: 010 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA_WAITONREQ_STATUS																															

DMA_WAITONREQ_STATUS	Bit 31-0	R	通道等待状态 0: dma_waitonreq[n]为低电平 1: dma_waitonreq[n]为高电平 注：此寄存器读出的值为常1
----------------------	----------	---	---

### 11.6.2.6 DMA通道软件请求寄存器 (DMA\_CHSWREQ)

该寄存器为只写寄存器，每个对应的 bit 都可在相应的通道上生产软件 DMA 请求。

DMA 通道软件请求寄存器 (DMA_CHSWREQ)																															
偏移地址: 014 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHSWREQ																															

CHSWREQ	Bit 31-0	T_W1	<b>通道软件请求</b> 0: 通道n上不生成DMA请求。 1: 在通道n上生成一个DMA请求。
---------	----------	------	---

### 11.6.2.7 DMA通道使用突发设置寄存器 (DMA\_CHUSEBURSTSET)

该寄存器为可读写寄存器，可禁止单次请求 dma\_sreq[C]产生请求信号，因此仅有 dma\_req[C]产生请求。读取该寄存器可返回冲突的使用状态。每个 bit 都有相对应的通道，bit0 对应通道 0，bit1 对应通道 1，以此类推。

当完成倒数第二次  $2^R$  传输，如果剩余的次数 N 小于  $2^R$ ，控制器会将 CHNL\_USEBURST\_SET 复位成 0。剩余的传输次数可通过 dma\_req[C]或 dma\_sreq[C]完成。

注：当 channel\_cfg 的设置值 N 小于  $2^R$ ，如果外设没有将 dma\_req[C]置为有效，则不应该将 CHNL\_USEBURST\_SET 置 1。

在外设分散-聚集模式下，当使用交替数据结构的 DMA 周期完成的时候，如果 channel\_cfg 中的 next\_useburst 已被设置为 1，控制器会将 CHNL\_USEBURST\_SET[C]置 1。

DMA 通道使用突发设置寄存器 (DMA_CHUSEBURSTSET)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_USEBURST_SET																															

寄存器名称	位范围	读写属性	描述
CHNL_USEBURST_SET	Bit 31-0	R/W	<p>返回冲突的使用状态，或禁止dma_sreq[C]产生DMA请求</p> <p>读取：</p> <p>0: 通道C对dma_req[C]或dma_sreq[C]请求做出回应。控制器执行<math>2^R</math>或单次总线传输。</p> <p>1: 通道C不回应dma_sreq[C]请求。控制器只回应dma_req[C]请求，执行<math>2^R</math>次传输。</p> <p>写入：</p> <p>0: 无效。使用CHNL_USEBURST_CLR寄存器将bit[C]写0。</p> <p>1: 禁止dma_sreq[C]产生DMA请求。控制器执行<math>2^R</math>次传输。</p>

### 11.6.2.8 DMA通道使用突发清除寄存器 (DMA\_CHUSEBURSTCLR)

DMA 通道使用突发清除寄存器(DMA_CHUSEBURSTCLR)																															
偏移地址: 01C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_USEBURST_CLR																															

CHNL_USEBURST_CLR	Bit 31-0	W	<p>使能dma_sreq[C]产生请求</p> <p>写入:</p> <p>0: 无效。使用CHNL_USEBURST_SET寄存器禁止dma_sreq[C]产生请求。</p> <p>1: 使能dma_sreq[C]产生DMA请求。</p>
-------------------	----------	---	---

### 11.6.2.9 DMA通道请求屏蔽设置寄存器 (DMA\_CHREQMASKSET)

该寄存器为读写寄存器，可禁止 dma\_req[C]和 dma\_sreq[C]产生请求。读取时可返回 dma\_req[C]和 dma\_sreq[C]的屏蔽状态。

DMA 通道请求屏蔽设置 寄存器(DMA_CHREQMASKSET)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_REQ_MASK_SET																															

CHNL_REQ_MASK_SET	Bit 31-0	R/W	<p>返回dma_req[C]和dma_sreq[C]的屏蔽状态,或禁止相应通道产生DMA请求</p> <p>读取:</p> <p>0: 通道C上的外部请求已使能。</p> <p>1: 通道C上的外部请求已禁止。</p> <p>写入:</p> <p>0: 无效。使用CHNL_REQ_MASK_CLR寄存器来使能DMA请求。</p>
-------------------	----------	-----	--

			1: 禁止dma_req[C]和dma_sreq[C]产生DMA请求。
--	--	--	-------------------------------------

### 11.6.2.10 DMA通道请求屏蔽清除寄存器 (DMA\_CHREQMASKCLR)

DMA 通道请求屏蔽清除寄存器 (DMA_CHREQMASKCLR)																																	
偏移地址: 024 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CHNL_REQ_MASK_CLR																																	

CHNL_REQ_MASK_CLR	Bit 31-0	W	<p>使能通道上的dma_req[C]和dma_sreq[C]请求</p> <p>写入:</p> <p>0: 无效。使用chnl_req_mask_set寄存器禁止dma_req[C]和dma_sreq[C]产生请求。</p> <p>1: 使能dma_sreq[C]或dma_sreq[C]产生DMA请求。</p>
-------------------	----------	---	---



### 11.6.2.11 DMA通道使能设置寄存器 (DMA\_CHENSET)

该寄存器为读写寄存器，设置该寄存器可使能通道。读取时，可返回该通道的状态。

DMA 通道使能设置寄存器(DMA_CHENSET)																															
偏移地址: 028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_ENABLE_SET																															

CHNL_ENABLE_SET	Bit 31-0	R/W	<p>返回通道的状态，或使能相应的通道</p> <p>读取：</p> <p>0: 通道C禁止。</p> <p>1: 通道C使能。</p> <p>写入：</p> <p>0: 无效。设置CHNL_ENABLE_CLR用来禁止某通道。</p> <p>1: 使能相应通道。</p>
-----------------	----------	-----	--

### 11.6.2.12 DMA通道使能清除寄存器 (DMA\_CHENCLR)

该寄存器为只写寄存器，可用来禁止相应的通道。

DMA 通道使能清除寄存器 (DMA_CHENCLR)																															
偏移地址: 02C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_ENABLE_CLR																															

CHNL_ENABLE_CLR	Bit 31-0	W	<p><b>禁止相应DMA通道</b></p> <p>写入:</p> <p>0: 无效。设置CHNL_ENABLE_SET寄存器来使能DMA通道。</p> <p>1: 禁止相应通道。</p>
-----------------	----------	---	---

注: 当发生以下任意一种情况时, 控制器可通过设置相应 CHNL\_ENABLE\_CLR 来禁止通道:

1. 控制器完成 DMA 周期
2. 控制器读取 channel\_cfg, 其 cycle\_ctrl 位段为 b000
3. AHB-Lite 总线上发生错误

### 11.6.2.13 DMA通道主要-交替设置寄存器 (DMA\_CHPRIALTSET)

该寄存器为读写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用交替数据结构。读取时，可返回相应通道的数据结构使用状态。

DMA 通道主要-交替设置寄存器 (DMA_CHPRIALTSET)																															
偏移地址: 030 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRI_ALT_SET																															

CHNL_PRI_ALT_SET	Bit 31-0	R/W	<p>返回通道的控制数据结构的状态，或为相应的<b>DMA通道选择交替数据结构</b></p> <p>读取： 0: 相应DMA通道使用的是主要数据结构。 1: 相应DMA通道使用的是交替数据结构。</p> <p>写入： 0: 无效。设置CHNL_PRI_ALT_CLR中相应的比特位为0。 1: 为相应DMA通道选择交替数据结构。</p>
------------------	----------	-----	---

注：当发生以下情况时，控制器将翻转 CHNL\_PRI\_ALT\_SET 中相应的比特位的值：

1. 在存储器/外设分散-聚集 DMA 周期模式下，完成由主要数据结构指定的 4 次传输。
2. 在乒乓 DMA 周期模式下，完成由主要数据结构指定的所有传输。
3. 在乒乓/存储器分散-聚集/外设分散-聚集 DMA 周期模式下，完成由交替数据结构指定的所有传输。

### 11.6.2.14 DMA通道主要-交替清除寄存器 (DMA\_CHPRIALTCLR)

该寄存器为只写寄存器。通过该寄存器，用户可将 DMA 通道设置为使用主要数据结构。

DMA 通道主要-交替清除寄存器 (DMA_CHPRIALTCLR)																															
偏移地址: 034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRI_ALT_CLR																															

CHNL_PRI_ALT_CLR	Bit 31-0	W	<p>设置相应的比特位，为相应的DMA通道选择主要数据结构</p> <p>写入:</p> <p>0: 无效。设置CHNL_PRI_ALT_SET寄存器用来选择交替数据结构</p> <p>1: 为相应DMA通道选择主要数据结构。</p>
------------------	----------	---	---

注: 当发生以下情况时，控制器将翻转 CHNL\_PRI\_ALT\_CLR 中相应的比特位的值:

1. 在存储器/外设分散-聚集 DMA 周期模式下，完成由主要数据结构指定的 4 次传输。
2. 在乒乓 DMA 周期模式下，完成由主要数据结构指定的所有传输。
3. 在乒乓/存储器分散-聚集/外设分散-聚集 DMA 周期模式下，完成由交替数据结构指定的所有传输。

### 11.6.2.15 DMA通道优先级设置寄存器 (DMA\_CHPRSET)

该寄存器为读写寄存器。通过该寄存器，用户可配置 DMA 通道为高优先级。读取时，可返回通道优先级的屏蔽状态。

DMA 通道优先级设置寄存器(DMA_CHPRSET)																															
偏移地址: 038 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRIORITY_SET																															

CHNL_PRIORITY_SET	Bit 31-0	R/W	<p>返回通道的优先级屏蔽状态，或设置通道优先级为高</p> <p>读取：</p> <p>0: 相应DMA通道为默认优先级。</p> <p>1: 相应DMA通道为高优先级。</p> <p>写入：</p> <p>0: 无效。设置CHNL_PRIORITY_CLR寄存器用将相应通道设置为默认优先级。</p> <p>1: 设置相应DMA通道为高优先级。</p>
-------------------	----------	-----	--

### 11.6.2.16 DMA通道优先级清除寄存器 (DMA\_CHPRCLR)

该寄存器为只写寄存器。通过该寄存器，用户可将DMA通道设置为默认优先级。

DMA 通道优先级清除寄存器 (DMA_CHPRCLR)																															
偏移地址: 03C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHNL_PRIORITY_SET																															

CHNL_PRIORITY_CLR	Bit 31-0	W	<p>设置相应的比特位，为相应的DMA通道选择默认优先级</p> <p>写入:</p> <p>0: 无效。设置CHNL_PRIORITY_SET寄存器，将相应DMA通道设置为高优先级。</p> <p>1: 将相应DMA通道设置为默认优先级。</p>
-------------------	----------	---	--

### 11.6.2.17 DMA总线错误清除寄存器 (DMA\_ERRCLR)

该读写寄存器可返回 dma\_err 的状态，可将 dma\_err 设置为低电平。

DMA 总线错误清除寄存器 (DMA_ERRCLR)																															
偏移地址: 04C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															ERR_CLR

Reserved	Bit 31-1	-	未定义，写0。
ERR_CLR	Bit 0	R/W	返回dma_err的状态，或设置该信号为低电平 读取： 0: dma_err为低电平 1: dma_err为高电平 写入： 0: 无效。dma_err状态不变。 1: 设置dma_err为低电平。 若以测试为目的，设置dma_err为高电平。

注：当 AHB-Lite 总线上发生错误的同时，若 dma\_err 被置为无效，则错误条件先发生，dma\_err 保持有效。

11.6.2.18 DMA中断状态寄存器 (DMA\_IFLAG)

DMA 中断状态寄存器(DMA_IFLAG)																																
偏移地址: 1000 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DMAERRIF		Reserved																			CH11DONEIF	CH10DONEIF	CH9DONEIF	CH8DONEIF	CH7DONEIF	CH6DONEIF	CH5DONEIF	CH4DONEIF	CH3DONEIF	CH2DONEIF	CH1DONEIF	CH0DONEIF

DMAERRIF	Bit31	R/C_W1	<b>DMA 错误中断标志</b> 0: DMA 未发生错误 1: DMA 发生错误 对该位写 1 清零。
Reserved	Bit30-12	-	保留
CH11DONEIF	Bit11	R/C_W1	<b>DMA 通道 11 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH10DONEIF	Bit10	R/C_W1	<b>DMA 通道 10 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH9DONEIF	Bit9	R/C_W1	<b>DMA 通道 9 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH8DONEIF	Bit8	R/C_W1	<b>DMA 通道 8 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH7DONEIF	Bit7	R/C_W1	<b>DMA 通道 7 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH6DONEIF	Bit6	R/C_W1	<b>DMA 通道 6 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH5DONEIF	Bit5	R/C_W1	<b>DMA 通道 5 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。



CH4DONEIF	Bit4	R/C_W1	<b>DMA 通道 4 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH3DONEIF	Bit3	R/C_W1	<b>DMA 通道 3 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH2DONEIF	Bit2	R/C_W1	<b>DMA 通道 2 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH1DONEIF	Bit1	R/C_W1	<b>DMA 通道 1 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。
CH0DONEIF	Bit0	R/C_W1	<b>DMA 通道 0 结束中断标志</b> 0: DMA 未结束 1: DMA 结束 对该位写 1 清零。

11.6.2.19 DMA中断使能控制寄存器 (DMA\_IER)

DMA 中断使能控制寄存器(DMA_IER)																																	
偏移地址: 100C <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DMAERRIE											Reserved											CH11DONEIE	CH10DONEIE	CH9DONEIE	CH8DONEIE	CH7DONEIE	CH6DONEIE	CH5DONEIE	CH4DONEIE	CH3DONEIE	CH2DONEIE	CH1DONEIE	CH0DONEIE

DMAERRIE	Bit 31	R/W	<b>DMA 错误中断使能</b> 0: 中断禁止 1: 中断使能
Reserved	Bit 30-12	-	保留
CH11DONEIE	Bit 11	R/W	<b>DMA 通道 11 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH10DONEIE	Bit10	R/W	<b>DMA 通道 10 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH9DONEIE	Bit9	R/W	<b>DMA 通道 9 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH8DONEIE	Bit8	R/W	<b>DMA 通道 8 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH7DONEIE	Bit7	R/W	<b>DMA 通道 7 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH6DONEIE	Bit6	R/W	<b>DMA 通道 6 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH5DONEIE	Bit 5	R/W	<b>DMA 通道 5 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH4DONEIE	Bit 4	R/W	<b>DMA 通道 4 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH3DONEIE	Bit 3	R/W	<b>DMA 通道 3 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH2DONEIE	Bit 2	R/W	<b>DMA 通道 2 结束中断使能</b>

			0: 中断禁止 1: 中断使能
CH1DONEIE	Bit 1	R/W	<b>DMA 通道 1 结束中断使能</b> 0: 中断禁止 1: 中断使能
CH0DONEIE	Bit 0	R/W	<b>DMA 通道 0 结束中断使能</b> 0: 中断禁止 1: 中断使能

11.6.2.20 DMA通道 0 复用选择寄存器 (DMA\_CH0\_SELCON)

DMA 通道 0 复用选择寄存器 (DMA_CH0_SELCON)																															
偏移地址: 1100 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
MSEL	Bit13-8	R/W	001110: SPI0
			001111: SPI1
			010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: TIM4
			011100: TIM5
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010(SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111(SRC=SPI0,SPI1,SPI2)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

**11.6.2.21 DMA通道 1 复用选择寄存器 (DMA\_CH1\_SELCON)**

DMA 通道 1 复用选择寄存器 (DMA_CH1_SELCON)																															
偏移地址: 1104 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved						MSIGSEL			

Reserved	Bit31-14	—	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010(SRC=UART0, UART1, UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111(SRC=SPI0, SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000, 010001(I2C0, I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0, TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2, TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4, TIM5)</b></p>



			<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPDM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--	--

**11.6.2.22 DMA通道 2 复用选择寄存器 (DMA\_CH2\_SELCON)**

DMA 通道 2 复用选择寄存器 (DMA_CH2_SELCON)																															
偏移地址: 1108 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	—	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

11.6.2.23 DMA通道3 复用选择寄存器 (DMA\_CH3\_SELCON)

DMA 通道3 复用选择寄存器 (DMA_CH3_SELCON)																															
偏移地址: 110C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPDM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

**11.6.2.24 DMA通道4 复用选择寄存器 (DMA\_CH4\_SELCON)**

DMA 通道4 复用选择寄存器 (DMA_CH4_SELCON)																															
偏移地址: 1110 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved						MSIGSEL			

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留



			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

			<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--	--

11.6.2.25 DMA通道5 复用选择寄存器 (DMA\_CH5\_SELCON)

DMA 通道5 复用选择寄存器 (DMA_CH5_SELCON)																															
偏移地址: 1114 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPDM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

**11. 6. 2. 26 DMA通道 6 复用选择寄存器 (DMA\_CH6\_SELCON)**

DMA 通道 6 复用选择寄存器 (DMA_CH6_SELCON)																															
偏移地址: 1118 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved						MSIGSEL			

Reserved	Bit31-14	—	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
MSEL	Bit13-8	R/W	001110: SPI0
			001111: SPI1
			010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--



**11.6.2.27 DMA通道7 复用选择寄存器 (DMA\_CH7\_SELCON)**

DMA 通道7 复用选择寄存器 (DMA_CH7_SELCON)																															
偏移地址: 111C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

11.6.2.28 DMA通道8复用选择寄存器 (DMA\_CH8\_SELCON)

DMA 通道8复用选择寄存器 (DMA_CH8_SELCON)																															
偏移地址: 1120 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	—	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

11.6.2.29 DMA通道9 复用选择寄存器 (DMA\_CH9\_SELCON)

DMA 通道9 复用选择寄存器 (DMA_CH9_SELCON)																															
偏移地址: 1124 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	-	保留
			输入源选择
			000000: 无输入
			000001: 预留
			000010: CRYPT
			000011: TRNG
			000100: 预留
			000101: ECC
			000110: ADC
			000111: CRC
			001000: UART0
			001001: UART1
			001010: UART2
			001011: 预留
			001100: 预留
			001101: 预留
			001110: SPI0
			001111: SPI1
MSEL	Bit13-8	R/W	010000: I2C0
			010001: I2C1
			010010: TIM0
			010011: TIM1
			010100: TIM2
			010101: TIM3
			010110: ADPCM
			010111: PDMPCM
			011000: 预留
			011001: QSPI
			011010: 预留
			011011: 预留
			011100: 预留
			011101: 预留
			011110: 预留

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>



			<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPDM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--	--

**11.6.2.30 DMA通道 10 复用选择寄存器 (DMA\_CH10\_SELCON)**

DMA 通道 10 复用选择寄存器 (DMA_CH10_SELCON)																															
偏移地址: 1128 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved						MSIGSEL			

Reserved	Bit31-14	—	保留
			<b>输入源选择</b> 000000: 无输入 000001: 预留 000010: CRYPT 000011: TRNG 000100: 预留 000101: ECC 000110: ADC 000111: CRC 001000: UART0 001001: UART1 001010: UART2 001011: 预留 001100: 预留 001101: 预留 001110: SPI0 001111: SPI1 010000: I2C0 010001: I2C1 010010: TIM0 010011: TIM1 010100: TIM2 010101: TIM3 010110: ADPCM 010111: PDMPCM 011000: 预留 011001: QSPI 011010: 预留 011011: 预留 011100: 预留 011101: 预留 011110: 预留
MSEL	Bit13-8	R/W	

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

		<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--

**11.6.2.31 DMA通道 11 复用选择寄存器 (DMA\_CH11\_SELCON)**

DMA 通道 11 复用选择寄存器 (DMA_CH11_SELCON)																															
偏移地址: 112C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MSEL						Reserved				MSIGSEL					

Reserved	Bit31-14	—	保留
			<b>输入源选择</b> 000000: 无输入 000001: 预留 000010: CRYPT 000011: TRNG 000100: 预留 000101: ECC 000110: ADC 000111: CRC 001000: UART0 001001: UART1 001010: UART2 001011: 预留 001100: 预留 001101: 预留 001110: SPI0 001111: SPI1 010000: I2C0 010001: I2C1 010010: TIM0 010011: TIM1 010100: TIM2 010101: TIM3 010110: ADPCM 010111: PDMPCM 011000: 预留 011001: QSPI 011010: 预留 011011: 预留 011100: 预留 011101: 预留 011110: 预留
MSEL	Bit13-8	R/W	

			011111: 预留 100000: PIS 100001: HASH
Reserved	Bit7-4	-	保留
MSIGSEL	Bit3-0	R/W	<p><b>MSEL=000000</b> MSIGS 无效</p> <p><b>MSEL=000010(SRC=CRYPT)</b>时 0000: CRYPT DMA 写数据申请 0001: CRYPT DMA 读数据申请</p> <p><b>MSEL=000011(SRC=TRNG)</b> TRNG DMA 申请</p> <p><b>MSEL=000101(SRC=ECC)</b> ECC DMA 申请</p> <p><b>MSEL=000110(SRC=ADC)</b> ADC DMA 申请</p> <p><b>MSEL=000111(SRC=CRC)</b> CRC DMA 申请</p> <p><b>MSEL=001000, 001001, 001010 (SRC=UART0,UART1,UART2)</b> 0000: 发送保持寄存器空申请 0001: 接收可用数据申请</p> <p><b>MSEL=001110, 001111 (SRC=SPI0,SPI1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010000,010001(I2C0,I2C1)</b> 0000: 接收缓冲器非空申请 0001: 发送缓冲器空申请</p> <p><b>MSEL=010010, 010011(SRC=TIM0,TIM1)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: 无效 0110: TIM 更新事件申请</p> <p><b>MSEL=010100, 010101(SRC=TIM2,TIM3)</b> 0000: 捕捉比较通道 1 申请 0001: 捕捉比较通道 2 申请 0010: 捕捉比较通道 3 申请 0011: 捕捉比较通道 4 申请 0100: TIM 触发申请 0101: TIM 比较匹配申请 0110: TIM 更新事件申请</p> <p><b>MSEL=011011, 011100(SRC=TIM4,TIM5)</b></p>

			<p>TIM 更新事件申请</p> <p><b>MSEL=011001(SRC=QSPI)</b>                  QSPI 间接模式操作访问请求                  0000: 间接操作模式数据读请求                  0001: 间接操作模式数据写请求</p> <p><b>MSEL=010110(SRC=ADPCM)</b>                  ADPCM 解压缩操作访问请求                  0000: 解压缩完成下一数据传输 DMA 请求</p> <p><b>MSEL=01011(SRC=PDMPCM)</b>                  0000:</p> <p><b>MSEL=100000 (SRC=PIS)</b>                  0000: PIS 通道 0                  0001: PIS 通道 1                  0010: PIS 通道 2                  0011: PIS 通道 3                  0100: PIS 通道 4                  0101: PIS 通道 5                  0110: PIS 通道 6                  0111: PIS 通道 7                  1000: PIS 通道 8                  1001: PIS 通道 9                  1010: PIS 通道 10                  1011: PIS 通道 11                  1100: PIS 通道 12                  1101: PIS 通道 13                  1110: PIS 通道 14                  1111: PIS 通道 15</p> <p><b>MSEL=100000 (SRC=HASH)</b>                  HASH DMA 请求</p>
--	--	--	--

## 第12章 外设互联(PIS)

### 12.1 概述

---

PIS(Peripheral Interaction System)在微控制器中作为外设互联的桥接口使用，利用 PIS 可实现外设之间的相互触发，控制及自动化工作，提高系统的实时性和快速响应能力，可避免占用过多的 CPU 资源并简化软件工作，为各种应用提供便捷。

### 12.2 特性

---

- ◆ 最多支持 16 个 PIS 通道选择
- ◆ 支持同步和异步通道选择
- ◆ 支持信号有效边缘选择
- ◆ 支持通道输出到管脚
- ◆ UART 输出调制



### 12.3 结构框图

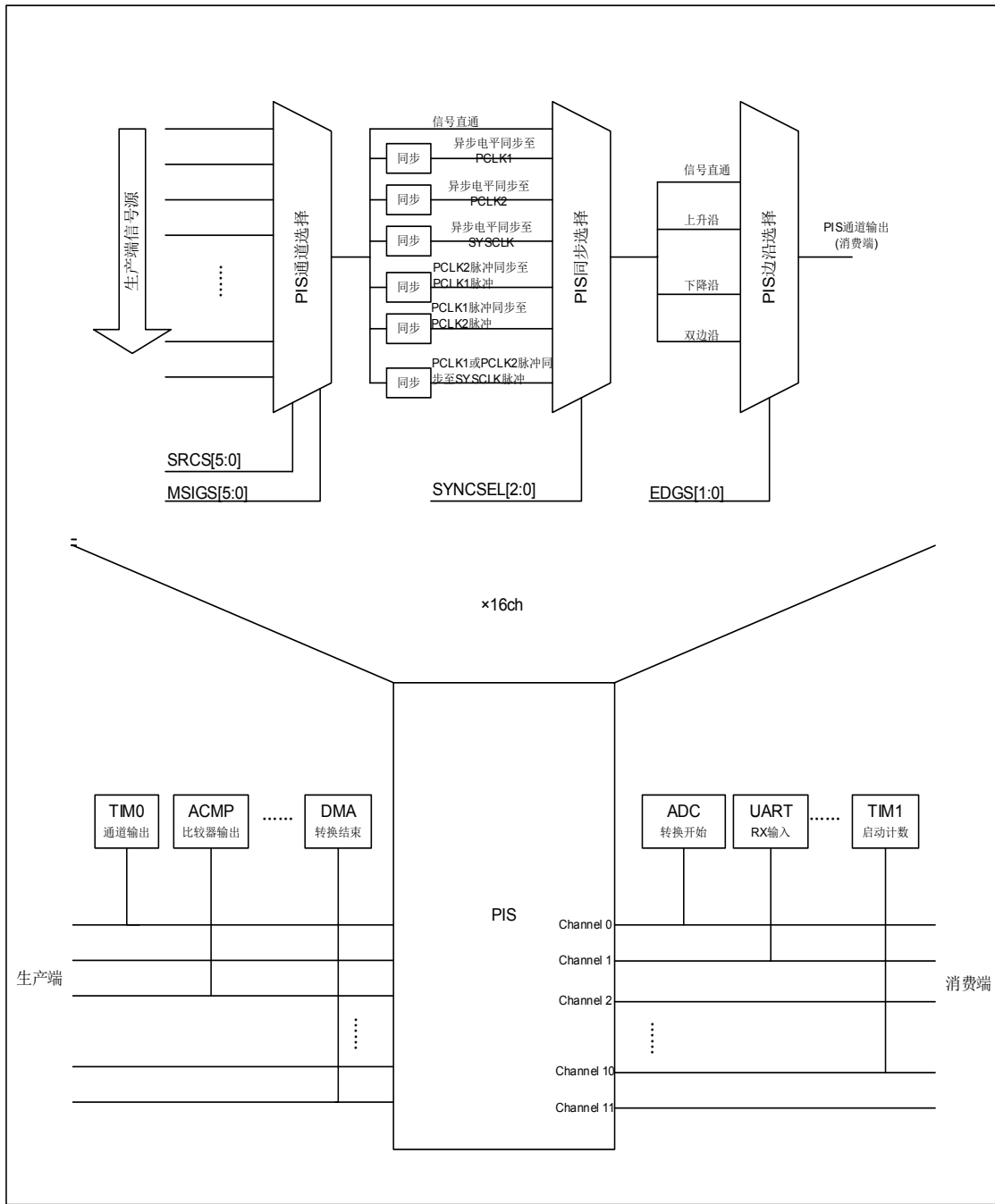


图 12-1 PIS 结构框图

### 12.4 功能描述

ES32W3120 系列产品的外设互联可支持 16 个通道资源, 每个通道均可对生产端信号进行多路复用。针对不同应用可灵活配置。

### 12.4.1 生产端信号

外设互联的生产端信号如下表所示：

	生产端	输出形式	异步支持	位置 (APB1, APB2 或 AHB)
<b>GPIO</b>	PB0~PB15 输入	电平	是	AHB 外设
<b>UART</b>	发送保持寄存器空 (TBEMPIF)	脉冲	—	APB1 外设
	接收可用数据中断 (RBRIF)	脉冲	—	
	IrDA 解码器输出	电平	—	
	RTS 输出	电平	—	
	TX 输出	电平	—	
<b>SPI</b>	接收缓冲器非空	脉冲	—	APB1 外设
	发送缓冲器空	脉冲	—	
	“片选”输出	电平	—	
<b>I2C</b>	接收缓冲器非空	电平	—	APB1 外设
	发送缓冲器空	电平	—	
<b>TIM</b>	更新事件	电平	—	APB1 外设
	触发事件	电平	—	
	输入捕获	电平	—	
	输出比较	电平	—	
<b>RTC</b>	亚秒、秒、分、时、日、月、年	脉冲	是	APB2 外设
	闹钟 A 和闹钟 B	脉冲	是	
<b>ADC</b>	标准组转换结束	脉冲	—	APB2 外设
	插入组转换结束	脉冲	—	
	窗口比较(模拟看门狗)触发 (不支持)	脉冲	—	
<b>LVD</b>	比较器输出	电平	是	—
<b>LP16T</b>	更新事件	电平	是	APB2 外设
<b>DMA</b>	DMA 通道完成	电平	是	AHB 外设

表 12-1 生产端信号

### 12.4.2 消费端信号

外设互联的消费端信号如下表所示：

	消费端	输入形式	异步支持	位置 (APB1, APB2 或 AHB)
UART	TX/RX 使能 (不支持)	电平	—	APB1 外设
	RX 输入	电平	—	
	IrDA 编码器输入	电平	—	
SPI	TX/RX 使能 (不支持)	电平	—	APB1 外设
	RX 输入	电平	—	
	CLK 输入	电平	—	
TIM	启动	脉冲	—	APB1 外设
	停止	脉冲	—	
	清零	脉冲	—	
	比较捕捉通道输入	电平或脉冲	—	
	断路输入	电平	—	
ADC	启动标准组转换	脉冲	—	APB2 外设
	启动插入组转换	脉冲	—	
LP16T	启动	脉冲	是	APB2 外设
	停止 (不支持)	脉冲	—	
	清零 (不支持)	脉冲	—	
DMA	DMA 通道请求	脉冲	是	AHB 外设

表 12-2 消费端信号

消费端信号的 PIS 通道分配如下表所示:

	消费端	源通道	备注
UART0	RX 输入或 IrDA 编码器输入	PIS 通道 9	由 UART0_RXD_SEL 设定
UART1	RX 输入或 IrDA 编码器输入	PIS 通道 10	由 UART1_RXD_SEL 设定
UART2	RX 输入或 IrDA 编码器输入	PIS 通道 11	由 UART2_RXD_SEL 设定
SPI0	RX 输入	PIS 通道 5	由 SPI0_RX_SEL 设定
	CLK 输入	PIS 通道 6	由 SPI0_CLK_SEL 设定
SPI1	RX 输入	PIS 通道 7	由 SPI1_RX_SEL 设定
	CLK 输入	PIS 通道 8	由 SPI1_CLK_SEL 设定
TIM0 (GP16C4T0)	ITR0	PIS 通道 12	
	ITR1	PIS 通道 13	
	ITR2	PIS 通道 14	
	ITR3	PIS 通道 15	
	捕捉通道 1	PIS 通道 1	由 TIM0_CH1IN_SEL 设定

	消费端	源通道	备注
	捕捉通道 2	PIS 通道 2	由 TIM0_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 TIM0_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 TIM0_CH4IN_SEL 设定
	断路输入	PIS 通道 0	由 TIM0_BRKIN_SEL 设定
	通道输出清除源 0	PIS 通道 6	由 TIM0_CMPIN_SEL 设定
	通道输出清除源 1	PIS 通道 7	
	通道输出清除源 2	PIS 通道 8	
通道输出清除源 3	PIS 通道 9		
<b>TIM1 (GP16C4T1)</b>	ITR0	PIS 通道 12	
	ITR1	PIS 通道 13	
	ITR2	PIS 通道 14	
	ITR3	PIS 通道 15	
	捕捉通道 1	PIS 通道 1	由 TIM1_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 2	由 TIM1_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 TIM1_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 TIM1_CH4IN_SEL 设定
	断路输入	断路输入	由 TIM1_BRKIN_SEL 设定
	通道输出清除源 0	PIS 通道 6	由 TIM1_CMPIN_SEL 设定
	通道输出清除源 1	PIS 通道 7	
	通道输出清除源 2	PIS 通道 8	
	通道输出清除源 3	PIS 通道 9	
<b>TIM2 (AD16C4T0)</b>	ITR0	PIS 通道 12	
	ITR1	PIS 通道 13	
	ITR2	PIS 通道 14	
	ITR3	PIS 通道 15	
	捕捉通道 1	PIS 通道 1	由 TIM2_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 2	由 TIM2_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 TIM2_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 TIM2_CH4IN_SEL 设定
	断路输入	断路输入	由 TIM2_BRKIN_SEL 设定
	通道输出清除源 0	PIS 通道 6	由 TIM2_CMPIN_SEL 设定
	通道输出清除源 1	PIS 通道 7	

	消费端	源通道	备注
	通道输出清除源 2	PIS 通道 8	
	通道输出清除源 3	PIS 通道 9	
TIM3 (AD16C4T1)	ITR0	PIS 通道 12	
	ITR1	PIS 通道 13	
	ITR2	PIS 通道 14	
	ITR3	PIS 通道 15	
	捕捉通道 1	PIS 通道 1	由 TIM3_CH1IN_SEL 设定
	捕捉通道 2	PIS 通道 2	由 TIM3_CH2IN_SEL 设定
	捕捉通道 3	PIS 通道 3	由 TIM3_CH3IN_SEL 设定
	捕捉通道 4	PIS 通道 4	由 TIM3_CH4IN_SEL 设定
	断路输入	断路输入	由 TIM3_BRKIN_SEL 设定
	通道输出清除源 0	PIS 通道 6	由 TIM3_CMPIN_SEL 设定
	通道输出清除源 1	PIS 通道 7	
	通道输出清除源 2	PIS 通道 8	
	通道输出清除源 3	PIS 通道 9	
ADC	启动标准组转换	PIS 通道 6	—
	启动插入组转换	PIS 通道 7	—
LP16T	EXT_TRIG0	PIS 通道 0	—
	EXT_TRIG1	PIS 通道 1	—
	EXT_TRIG2	PIS 通道 2	—
	EXT_TRIG3	PIS 通道 3	—
	EXT_TRIG4	PIS 通道 4	—
	EXT_TRIG5	PIS 通道 5	—
	EXT_TRIG6	PIS 通道 6	—
	EXT_TRIG7	PIS 通道 7	—
DMA	DMA 申请	PIS 通道 15~0	参照 DMA 控制器章节描述

表 12-3 消费端的 PIS 通道分配

### 12.4.3 PIS通道选择

PIS 的源端定义为生产端信号，PIS 的输出信号用于消费端。消费端可根据应用需要，来选择合适的生产端信号，并通过选择同步路径以保证正确采样到生产端信号。PIS 的生产端信号选择由 PIS 通道控制寄存器（PIS\_CHx\_CON,x=0~15）配置。PIS 通道控制寄存器的 SRCS 位用来选择生产端模块，配置 MSIGS 位则可从生产端模块的多路信号中选择一路作为生产端信号。

以下从几种情况来举例说明 PIS 的配置。

### 12.4.3.1 同一时钟域互联

以 LP16T 和 ADC 为例，在 LP16T 使用 PCLK2 作为计数时钟的情况下，产生与 PCLK2 同步的更新事件去触发 ADC 转换动作。可参照如下配置：

1. 通过 **错误！未找到引用源。** 可知 ADC 规则组转换的启动信号为 PIS 通道 6。
2. 设定寄存器 PIS\_CH6\_CON 的 SRCS 为 010111 选择 LP16T 为生产端模块。
3. 设定寄存器 PIS\_CH6\_CON 的 MSIGS 为 0000，选择 LP16T 的同步更新事件 (PCLK2 同步)作为生产端信号。
4. 设定寄存器 PIS\_CH6\_CON 的 SYNCSEL 为 000 (信号直通)，并将 EDGS 设为 00 (不输出边沿)，此时 PULCK 可任意设定。
5. 配置 ADC 选择外部触发方式进行规则组转换。
6. 配置 LP16T 进行计数，产生更新事件后将触发 ADC 开始 AD 转换。

对采用异步时钟 (非 PCLK1, PCLK2 等) 工作的两个模块之间相互触发，也可参照以上流程进行配置。

### 12.4.3.2 APB1 和 APB2 外设之间互联

以 TIM0 (位于 APB1) 和 ADC (位于 APB2) 为例，TIM0 使用 PCLK1 作为计数时钟，产生与 PCLK1 同步的更新事件去触发 ADC 转换动作。可参照如下配置：

1. 通过 **错误！未找到引用源。** 可知 ADC 规则组转换的启动信号为 PIS 通道 6。
2. 设定寄存器 PIS\_CH6\_CON 的 SRCS 为 010010 选择 TIM0 为生产端模块。
3. 设定寄存器 PIS\_CH6\_CON 的 MSIGS 为 0000, 选择 TIM0 的同步更新事件 (PCLK1 同步)作为生产端信号。
4. 设定寄存器 PIS\_CH6\_CON 的 SYNCSEL 为 101 (PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲)，并将 EDGS 设为 00 (不输出边沿)，此时 PULCK 可任意设定。
5. 配置 ADC 选择外部触发方式进行规则组转换。
6. 配置 TIM0 进行计数，产生更新事件后将触发 ADC 开始 AD 转换。

### 12.4.3.3 生产端信号为异步信号

以 RTC 和 ADC (位于 APB2) 为例，RTC 使用 LOSC 作为计数时钟，产生 32.768KHz 的事件脉冲去触发 ADC 转换动作。可参照如下配置：

1. 通过 **错误！未找到引用源。** 可知 ADC 规则组转换的启动信号为 PIS 通道 6。
2. 设定寄存器 PIS\_CH6\_CON 的 SRCS 为 010110 选择 RTC 为生产端模块。
3. 设定寄存器 PIS\_CH6\_CON 的 MSIGS 为 0000，选择 RTC 的亚秒、秒、分、时、日、月、年事件脉冲作为生产端信号。
4. 设定寄存器 PIS\_CH6\_CON 的 SYNCSEL 为 010 (异步电平信号经同步后输出到 PCLK2 时钟域)，并将 EDGS 设为 01 (上升沿)，此时 PULCK 可任意设定。
5. 配置 ADC 选择外部触发方式进行规则组转换。
6. 配置 RTC 进行计数，产生事件脉冲触发 ADC 开始 AD 转换。

### 12.4.4 UART 输出调制

UART 的输出调制功能利用定时器 PWM 波或 BUZ 信号对 UART 的 TX 调制后发送到端口。

调制方式如下图所示：

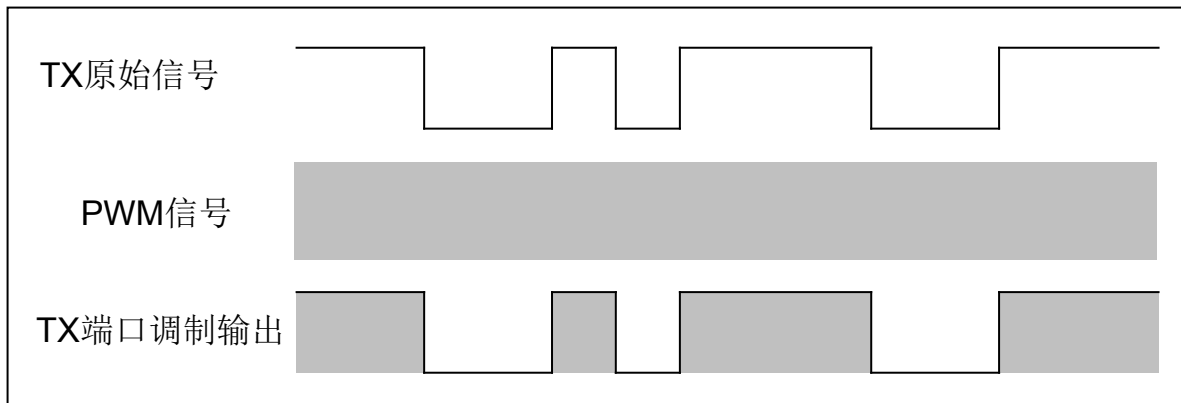


图 12-2 高电平调制输出波形图

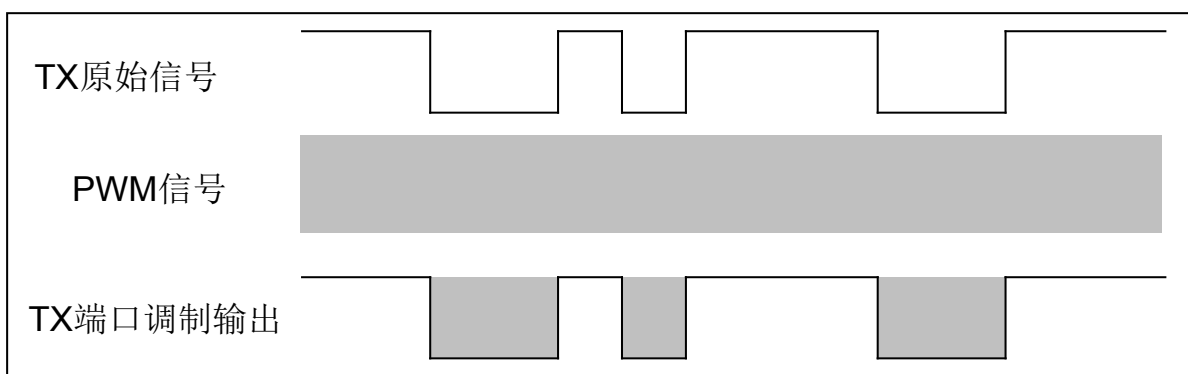


图 12-3 低电平调制输出波形图

以下为 UART 输出调制的参考配置流程（以 TIM0 调制 UART0 为例）：

1. 设置寄存器 UART0\_TXMCR 的 TXMSS 为 0001，选择 TIM0 作为调制源。
2. 设置寄存器 UART0\_TXMCR 的 TXSIGS 为 0000，选择 TIM0 通道 1 输出作为调制信号。
3. 设置 TXMLVLS 选择调制电平。
4. 配置 TIM0 进行计数。
5. 配置 UART 发送数据。

## 12.5 特殊功能寄存器

### 12.5.1 寄存器列表

PIS 寄存器列表			
名称	偏移地址	类型	描述
PIS 基地址: 4008_6000 <sub>H</sub>			
PIS_CH0_CON	0000 <sub>H</sub>	R/W	PIS 通道 0 控制寄存器
PIS_CH1_CON	0004 <sub>H</sub>	R/W	PIS 通道 1 控制寄存器
PIS_CH2_CON	0008 <sub>H</sub>	R/W	PIS 通道 2 控制寄存器
PIS_CH3_CON	000C <sub>H</sub>	R/W	PIS 通道 3 控制寄存器
PIS_CH4_CON	0010 <sub>H</sub>	R/W	PIS 通道 4 控制寄存器
PIS_CH5_CON	0014 <sub>H</sub>	R/W	PIS 通道 5 控制寄存器
PIS_CH6_CON	0018 <sub>H</sub>	R/W	PIS 通道 6 控制寄存器
PIS_CH7_CON	001C <sub>H</sub>	R/W	PIS 通道 7 控制寄存器
PIS_CH8_CON	0020 <sub>H</sub>	R/W	PIS 通道 8 控制寄存器
PIS_CH9_CON	0024 <sub>H</sub>	R/W	PIS 通道 9 控制寄存器
PIS_CH10_CON	0028 <sub>H</sub>	R/W	PIS 通道 10 控制寄存器
PIS_CH11_CON	002C <sub>H</sub>	R/W	PIS 通道 11 控制寄存器
PIS_CH12_CON	0030 <sub>H</sub>	R/W	PIS 通道 12 控制寄存器
PIS_CH13_CON	0034 <sub>H</sub>	R/W	PIS 通道 13 控制寄存器
PIS_CH14_CON	0038 <sub>H</sub>	R/W	PIS 通道 14 控制寄存器
PIS_CH15_CON	003C <sub>H</sub>	R/W	PIS 通道 15 控制寄存器
PIS_CH_OER	0040 <sub>H</sub>	R/W	PIS 通道端口输出使能寄存器
PIS_TAR_CON0	0044 <sub>H</sub>	R/W	PIS 消费端通道控制寄存器 0
PIS_TAR_CON1	0048 <sub>H</sub>	R/W	PIS 消费端通道控制寄存器 1
Reserved	004C <sub>H</sub> ~005C <sub>H</sub>	-	保留
UART0_TXMCR	0060 <sub>H</sub>	R/W	UART0 输出调制控制寄存器
UART1_TXMCR	0064 <sub>H</sub>	R/W	UART1 输出调制控制寄存器
UART2_TXMCR	0068 <sub>H</sub>	R/W	UART2 输出调制控制寄存器
UART3_TXMCR	006C <sub>H</sub>	R/W	UART3 输出调制控制寄存器



## 12.5.2 寄存器描述

### 12.5.2.1 PIS通道 0 控制寄存器 (PIS\_CH0\_CON)

PIS 通道 0 控制寄存器 (PIS_CH0_CON)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL			Reserved				PULCK		EDGS		Reserved			SRCS					Reserved			MSIGS					

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通: 在一些低功耗应用的场合需采用异步的电平或脉冲;生产端信号源与消费端处于同一时钟域,无需同步,选择信号直通即可。</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域 010:异步电平信号经同步后输出到 PCLK2 时钟域 011:异步电平信号同步后输出到 SYSCLK 时钟域 100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲 101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲 110: PCLK1 或 PCLK2 时钟域脉冲信号同步为 SYSCLK 脉冲 111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置,比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下,边沿采样时钟是确定的,该设定无效。</p> <p>00: PCLK1 01: PCLK2 10: SYSCLK 11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用,低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿 01: 上升沿</p>

			10: 下降沿 11: 双边沿
Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b></p> <p><b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p>

		<p>0011: RTS 输出  0100: TX 输出  0101: 发送保持寄存器空同步脉冲  0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>  0000: 接收缓冲器非空脉冲  0001: 发送缓冲器空脉冲  0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>  0000: 接收缓冲器非空 LEVEL  0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>  0000: 更新事件脉冲  0001: 触发事件脉冲  0010: 通道 1 输入捕获脉冲  0011: 通道 1 输出比较脉冲  0100: 通道 2 输入捕获脉冲  0101: 通道 2 输出比较脉冲  0110: 通道 3 输入捕获脉冲  0111: 通道 3 输出比较脉冲  1000: 通道 4 输入捕获脉冲  1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>  0000: 亚秒、秒、分、时、日、月、年脉冲  0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>  0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)  00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>  DMA 通道完成脉冲</p>
--	--	---

12.5.2.2 PIS通道 1 控制寄存器 (PIS\_CH1\_CON)

PIS 通道 1 控制寄存器 (PIS_CH1_CON)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

### 12.5.2.3 PIS通道 2 控制寄存器 (PIS\_CH2\_CON)

PIS 通道 2 控制寄存器 (PIS_CH2_CON)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通: 在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域 010:异步电平信号经同步后输出到 PCLK2 时钟域 011:异步电平信号同步后输出到 SYSCLK 时钟域 100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲 101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲 110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲 111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1 01: PCLK2 10: SYSCLK 11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿 01: 上升沿 10: 下降沿 11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>



		<p>0101: 发送保持寄存器同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	---

12.5.2.4 PIS通道3控制寄存器 (PIS\_CH3\_CON)

PIS 通道 3 控制寄存器 (PIS_CH3_CON)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

### 12.5.2.5 PIS通道 4 控制寄存器 (PIS\_CH4\_CON)

PIS 通道 4 控制寄存器 (PIS_CH4_CON)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.6 PIS通道 5 控制寄存器 (PIS\_CH5\_CON)

PIS 通道 5 控制寄存器 (PIS_CH5_CON)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>



Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.7 PIS通道 6 控制寄存器 (PIS\_CH6\_CON)

PIS 通道 6 控制寄存器 (PIS_CH6_CON)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          0001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	---

12.5.2.8 PIS通道 7 控制寄存器 (PIS\_CH7\_CON)

PIS 通道 7 控制寄存器 (PIS_CH7_CON)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--



12.5.2.9 PIS通道 8 控制寄存器 (PIS\_CH8\_CON)

PIS 通道 8 控制寄存器 (PIS_CH8_CON)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.10 PIS通道 9 控制寄存器 (PIS\_CH9\_CON)

PIS 通道 9 控制寄存器 (PIS_CH9_CON)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          0001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	---

12.5.2.11 PIS通道 10 控制寄存器 (PIS\_CH10\_CON)

PIS 通道 10 控制寄存器 (PIS_CH10_CON)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>



		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.12 PIS通道 11 控制寄存器 (PIS\_CH11\_CON)

PIS 通道 11 控制寄存器 (PIS_CH11_CON)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.13 PIS通道 12 控制寄存器 (PIS\_CH12\_CON)

PIS 通道 12 控制寄存器 (PIS_CH12_CON)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.14 PIS通道 13 控制寄存器 (PIS\_CH13\_CON)

PIS 通道 13 控制寄存器 (PIS_CH13_CON)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>



Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.15 PIS通道 14 控制寄存器 (PIS\_CH14\_CON)

PIS 通道 14 控制寄存器 (PIS_CH14_CON)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--

12.5.2.16 PIS通道 15 控制寄存器 (PIS\_CH15\_CON)

PIS 通道 15 控制寄存器 (PIS_CH15_CON)																															
偏移地址: 3C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SYNCSEL				Reserved				PULCK		EDGS		Reserved		SRCS						Reserved				MSIGS			

Reserved	Bit31-27	—	保留
SYNCSEL	Bit26-24	R/W	<p><b>信号同步选择:</b></p> <p>000:信号直通。在以下场合可设置信号直通:                      在一些低功耗应用的场合需采用异步的电平或脉冲; 触发信号源与触发目标处于同一时钟域, 无需同步, 比如 TIM 和 SPI 之间的触发控制</p> <p>001:异步电平信号经同步后输出到 PCLK1 时钟域                      010:异步电平信号经同步后输出到 PCLK2 时钟域                      011:异步电平信号同步后输出到 SYSCLK 时钟域                      100:PCLK2 时钟域脉冲信号同步为 PCLK1 脉冲                      101: PCLK1 时钟域脉冲信号同步为 PCLK2 脉冲                      110: PCLK2 或 PCLK1 时钟域脉冲信号同步为 SYSCLK 脉冲                      111:预留</p>
Reserved	Bit23-20	—	保留
PULCK	Bit19-18	R/W	<p><b>边沿采样时钟选择</b></p> <p>在 SYNCSEL=000, 且触发信号源与触发目标处于同一时钟域时请根据触发目标时钟域进行设置, 比如触发目标为 ADC, 请将 PULCK 设为 01 选择 PCLK2。其余情况下, 边沿采样时钟是确定的, 该设定无效。</p> <p>00: PCLK1                      01: PCLK2                      10: SYSCLK                      11: 预留</p>
EDGS	Bit17-16	R/W	<p><b>边沿选择:</b></p> <p>在输入信号为电平时使用, 低功耗应用场合下的异步通道请设置为 00。</p> <p>00: 不输出边沿                      01: 上升沿                      10: 下降沿                      11: 双边沿</p>

Reserved	Bit15-14	—	保留
SRCS	Bit13-8	R/W	<p>输入源选择</p> <p>000000: 无输入</p> <p>000001: GPIO</p> <p>000110: ADC</p> <p>000111: LVD</p> <p>001000: UART0</p> <p>001001: UART1</p> <p>001010: UART2</p> <p>001110: SPI0</p> <p>001111: SPI1</p> <p>010000: I2C0</p> <p>010001: I2C1</p> <p>010010: TIM0</p> <p>010011: TIM1</p> <p>010100: TIM2</p> <p>010101: TIM3</p> <p>010110: RTC</p> <p>010111: LP16T</p> <p>011001: DMA</p> <p>其他 : 无输入</p>
Reserved	Bit7-4	—	保留
MSIGS	Bit3-0	R/W	<p><b>SRCS=000000</b></p> <p>MSIGS 无效</p> <p><b>SRCS=000001(SRC=GPIO)</b></p> <p>0000: PB0</p> <p>0001: PB1</p> <p>0010: PB2</p> <p>.....</p> <p>1110: PB14</p> <p>1111: PB15</p> <p><b>SRCS=000110(SRC=ADC)</b></p> <p>0000: 标准组转换结束</p> <p>0001: 插入组转换结束</p> <p>0010: 窗口比较(模拟看门狗)触发 (不支持)</p> <p><b>SRCS=000111(SRC=LVD)</b></p> <p>LVD 比较器输出</p> <p><b>SRCS=001000, 001001, 001010</b> <b>(SRC=UART0,UART1,UART2)</b></p> <p>0000: 保留</p> <p>0001: 保留</p> <p>0010: IrDA 输出电平</p> <p>0011: RTS 输出</p> <p>0100: TX 输出</p>

		<p>0101: 发送保持寄存器空同步脉冲          0110: 接收可用数据同步脉冲  <b>SRCS=001110, 001111(SRC=SPI0,SPI1)</b>          0000: 接收缓冲器非空脉冲          0001: 发送缓冲器空脉冲          0010: “片选”输出  <b>SRCS=010000,010001(I2C0,I2C1)</b>          0000: 接收缓冲器非空 LEVEL          0001: 发送缓冲器空 LEVEL  <b>SRCS=010010, 010011,010100, 010101</b>  <b>(SRC= TIM0,TIM1,TIM2,TIM3)</b>          0000: 更新事件脉冲          0001: 触发事件脉冲          0010: 通道 1 输入捕获脉冲          0011: 通道 1 输出比较脉冲          0100: 通道 2 输入捕获脉冲          0101: 通道 2 输出比较脉冲          0110: 通道 3 输入捕获脉冲          0111: 通道 3 输出比较脉冲          1000: 通道 4 输入捕获脉冲          1001: 通道 4 输出比较脉冲  <b>SRCS=010110(SRC=RTC)</b>          0000: 亚秒、秒、分、时、日、月、年脉冲          0001: 闹钟 A 和闹钟 B  <b>SRCS=010111(SRC=LP16T)</b>          0000: 同步更新事件脉冲 (LP16T 采用 APB 时钟计数时使用该设定, 其余情况作为异步更新事件) (不支持)          00001: 异步更新事件脉冲  <b>SRCS=011001(SRC=DMA)</b>          DMA 通道完成脉冲</p>
--	--	--



### 12.5.2.17 PIS通道端口输出使能寄存器 (PIS\_CH\_OER)

PIS 通道端口输出使能寄存器 (PIS_CH_OER)																															
偏移地址: 40 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CH3OE	CH2OE	CH1OE	CH0OE

Reserved	Bit31-4	—	保留
CH3OE	Bit3	R/W	<b>PIS 通道 3 输出至端口使能</b> 0: 输出到端口禁止 1: 输出到端口使能
CH2OE	Bit2	R/W	<b>PIS 通道 2 输出至端口使能</b> 0: 输出到端口禁止 1: 输出到端口使能
CH1OE	Bit1	R/W	<b>PIS 通道 1 输出至端口使能</b> 0: 输出到端口禁止 1: 输出到端口使能
CH0OE	Bit0	R/W	<b>PIS 通道 0 输出至端口使能</b> 0: 输出到端口禁止 1: 输出到端口使能

12.5.2.18 PIS消费端通道控制寄存器 0 (PIS\_TAR\_CON0)

PIS 消费端通道控制寄存器 0 (PIS_TAR_CON0)																															
偏移地址: 44 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM3_CSSBRK_SEL	TIM3_CMPIN_SEL	Reserved	TIM3_BRKIN_SEL	TIM3_CH4IN_SEL	TIM3_CH3IN_SEL	TIM3_CH2IN_SEL	TIM3_CH1IN_SEL	TIM2_CSSBRK_SEL	TIM2_CMPIN_SEL	Reserved	TIM2_BRKIN_SEL	TIM2_CH4IN_SEL	TIM2_CH3IN_SEL	TIM2_CH2IN_SEL	TIM2_CH1IN_SEL	TIM1_CSSBRK_SEL	TIM1_CMPIN_SEL	Reserved	TIM1_BRKIN_SEL	TIM1_CH4IN_SEL	TIM1_CH3IN_SEL	TIM1_CH2IN_SEL	TIM1_CH1IN_SEL	TIM0_CMPIN_SEL	Reserved	TIM0_BRKIN_SEL	TIM0_BRKIN_SEL	TIM0_CH4IN_SEL	TIM0_CH3IN_SEL	TIM0_CH2IN_SEL	TIM0_CH1IN_SEL

TIM3_CSSBRK_SEL	Bit31	R/W	<b>TIM3 CSS BRK 输入选择 (此控制位无效)</b> 0: 输入 0 1: 将 PIS 通信 5 输出作为输入 注: 此控制位无效
TIM3_CMPIN_SEL	Bit30	R/W	<b>TIM3 CMP IN 输入选择</b> 0: 输入 0 1: 将 PIS 通道 6~9 输出作为输入 TIM3_CMP_IN[0]: PIS 通道 6 输出作为输入 TIM3_CMP_IN[1]: PIS 通道 7 输出作为输入 TIM3_CMP_IN[2]: PIS 通道 8 输出作为输入 TIM3_CMP_IN[3]: PIS 通道 9 输出作为输入
-	Bit29	-	保留
TIM3_BRKIN_SEL	Bit28	R/W	<b>TIM3 断路输入选择</b> 0: 从端口输入 1: 将 PIS 通道 0 输出作为输入
TIM3_CH4IN_SEL	Bit27	R/W	<b>TIM3 输入捕捉通道 4 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 4 输出作为输入
TIM3_CH3IN_SEL	Bit26	R/W	<b>TIM3 输入捕捉通道 3 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 3 输出作为输入
TIM3_CH2IN_SEL	Bit25	R/W	<b>TIM3 输入捕捉通道 2 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 2 输出作为输入
TIM3_CH1IN_SEL	Bit24	R/W	<b>TIM3 输入捕捉通道 1 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 1 输出作为输入
TIM2_CSSBRK_SEL	Bit23	R/W	<b>TIM2 CSS BRK 输入选择 (此控制位无效)</b> 0: 输入 0 1: 将 PIS 通信 5 输出作为输入

			注：此控制位无效
TIM2_CMPIN_SEL	Bit22	R/W	<b>TIM2 CMP IN 输入选择</b> 0: 输入 0 1: 将 PIS 通道 6~9 输出作为输入 TIM2_CMP_IN[0]: PIS 通道 6 输出作为输入 TIM2_CMP_IN[1]: PIS 通道 7 输出作为输入 TIM2_CMP_IN[2]: PIS 通道 8 输出作为输入 TIM2_CMP_IN[3]: PIS 通道 9 输出作为输入
-	Bit21	-	保留
TIM2_BRKIN_SEL	Bit20	R/W	<b>TIM2 断路输入选择</b> 0: 从端口输入 1: 将 PIS 通道 0 输出作为输入
TIM2_CH4IN_SEL	Bit19	R/W	<b>TIM2 输入捕捉通道 4 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 4 输出作为输入
TIM2_CH3IN_SEL	Bit18	R/W	<b>TIM2 输入捕捉通道 3 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 3 输出作为输入
TIM2_CH2IN_SEL	Bit17	R/W	<b>TIM2 输入捕捉通道 2 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 2 输出作为输入
TIM2_CH1IN_SEL	Bit16	R/W	<b>TIM2 输入捕捉通道 1 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 1 输出作为输入
TIM1_CSSBRK_SEL	Bit15	R/W	<b>TIM1 CSS BRK 输入选择（此控制位无效）</b> 0: 输入 0 1: 将 PIS 通信 5 输出作为输入 注：此控制位无效
TIM1_CMPIN_SEL	Bit14	R/W	<b>TIM1 CMP IN 输入选择</b> 0: 输入 0 1: 将 PIS 通道 6~9 输出作为输入 TIM1_CMP_IN[0]: PIS 通道 6 输出作为输入 TIM1_CMP_IN[1]: PIS 通道 7 输出作为输入 TIM1_CMP_IN[2]: PIS 通道 8 输出作为输入 TIM1_CMP_IN[3]: PIS 通道 9 输出作为输入
-	Bit13	-	保留
TIM1_BRKIN_SEL	Bit12	R/W	<b>TIM1 断路输入选择（此控制位无效）</b> 0: 从端口输入 1: 将 PIS 通道 0 输出作为输入
TIM1_CH4IN_SEL	Bit11	R/W	<b>TIM1 输入捕捉通道 4 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 4 输出作为输入
TIM1_CH3IN_SEL	Bit10	R/W	<b>TIM1 输入捕捉通道 3 输入选择</b>

			0: 从端口输入 1: 将 PIS 通道 3 输出作为输入
TIM1_CH2IN_SEL	Bit9	R/W	<b>TIM1 输入捕捉通道 2 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 2 输出作为输入
TIM1_CH1IN_SEL	Bit8	R/W	<b>TIM1 输入捕捉通道 1 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 1 输出作为输入
TIM0_CSSBRK_SEL	Bit7	R/W	<b>TIM0 CSS BRK 输入选择 (此控制位无效)</b> 0: 输入 0 1: 将 PIS 通信 5 输出作为输入 注: 此控制位无效
TIM0_CMPIN_SEL	Bit6	R/W	<b>TIM0 CMP IN 输入选择</b> 0: 输入 0 1: 将 PIS 通道 6~9 输出作为输入 TIM0_CMP_IN[0]: PIS 通道 6 输出作为输入 TIM0_CMP_IN[1]: PIS 通道 7 输出作为输入 TIM0_CMP_IN[2]: PIS 通道 8 输出作为输入 TIM0_CMP_IN[3]: PIS 通道 9 输出作为输入
-	Bit5	-	保留
TIM0_BRKIN_SEL	Bit4	R/W	<b>TIM0 断路输入选择 (此控制位无效)</b> 0: 从端口输入 1: 将 PIS 通道 0 输出作为输入
TIM0_CH4IN_SEL	Bit3	R/W	<b>TIM0 输入捕捉通道 4 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 4 输出作为输入
TIM0_CH3IN_SEL	Bit2	R/W	<b>TIM0 输入捕捉通道 3 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 3 输出作为输入
TIM0_CH2IN_SEL	Bit1	R/W	<b>TIM0 输入捕捉通道 2 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 2 输出作为输入
TIM0_CH1IN_SEL	Bit0	R/W	<b>TIM0 输入捕捉通道 1 输入选择</b> 0: 从端口输入 1: 将 PIS 通道 1 输出作为输入

12.5.2.19 PIS消费端通道控制寄存器 1 (PIS\_TAR\_CON1)

PIS 消费端通道控制寄存器 1 (PIS_TAR_CON1)																																					
偏移地址: 48 <sub>H</sub>																																					
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved																SPI1_CLK_SEL	SPI1_RX_SEL	SPI0_CLK_SEL	SPI0_RX_SEL	Reserved								LPUART0_RXD_SEL	USART1_RXD_SEL	USART0_RXD_SEL	Reserved			UART3_RXD_SEL	UART2_RXD_SEL	UART1_RXD_SEL	UART0_RXD_SEL

Reserved	Bit31-16	—	保留
SPI1_CLK_SEL	Bit15	R/W	<b>SPI1 CLK 输入选择</b> 0: 从端子输入 1: 将 PIS 通道 8 作为输入
SPI1_RX_SEL	Bit14	R/W	<b>SPI1 RX 输入选择</b> 0: 从端子输入 1: 将 PIS 通道 7 作为输入
SPI0_CLK_SEL	Bit13	R/W	<b>SPI0 CLK 输入选择</b> 0: 从端子输入 1: 将 PIS 通道 6 作为输入
SPI0_RX_SEL	Bit12	R/W	<b>SPI0 RX 输入选择</b> 0: 从端子输入 1: 将 PIS 通道 5 作为输入
Reserved	Bit11-3	—	保留
UART2_RXD_SEL	Bit2	R/W	<b>UART2 RXD 输入选择</b> 0: 从端子 RXD 输入 1: 将 PIS 通道 11 作为输入
UART1_RXD_SEL	Bit1	R/W	<b>UART1 RXD 输入选择</b> 0: 从端子 RXD 输入 1: 将 PIS 通道 10 作为输入
UART0_RXD_SEL	Bit0	R/W	<b>UART0 RXD 输入选择</b> 0: 从端子 RXD 输入 1: 将 PIS 通道 9 作为输入

12.5.2.20 UART0 输出调制控制寄存器 (UART0\_TXMCR)

UART0 输出调制控制寄存器 (UART0_TXMCR)																															
偏移地址: 60 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TXMLVLS		TXMSS			TXSIGs			

Reserved	Bit31-9	—	保留
TXMLVLS	Bit8	R/W	<b>TX 调制电平选择</b> 0: 低电平调制 (TX 与所选取的调制信号进行硬件或操作) 1: 高电平调制 (TX 与所选取的调制信号进行硬件与操作)
TXMSS	Bit7-4	R/W	<b>TX 调制源选择</b> 0000: 调制禁止 0001: TIM0 0010: TIM1 0011: TIM2 0100: TIM3 0111: LP16T 1000: BUZ 其余: 无调制
TXSIGs	Bit3-0	R/W	<b>TX 调制信号选择</b> <b>TXMSS=0000</b> TXSIGs 无效 <b>TXMSS=0001,0010,0011,0100 (调制源=TIM0,TIM1,TIM2,TIM3)</b> 0000: TIMx_CH1 0001: TIMx_CH2 0010: TIMx_CH3 0011: TIMx_CH4 <b>TXMSS=1000(调制源=BUZ)</b> 调制信号为 BUZ 输出

12.5.2.21 UART1 输出调制控制寄存器 (UART1\_TXMCR)

UART1 输出调制控制寄存器 (UART1_TXMCR)																															
偏移地址: 64 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TXMLVLS		TXMSS			TXSIGs			

Reserved	Bit31-9	—	保留
TXMLVLS	Bit8	R/W	<b>TX 调制电平选择</b> 0: 低电平调制 (TX 与所选取的调制信号进行硬件或操作) 1: 高电平调制 (TX 与所选取的调制信号进行硬件与操作)
TXMSS	Bit7-4	R/W	<b>TX 调制源选择</b> 0000: 调制禁止 0001: TIM0 0010: TIM1 0011: TIM2 0100: TIM3 0111: LP16T 1000: BUZ 其余: 无调制
TXSIGs	Bit3-0	R/W	<b>TX 调制信号选择</b> <b>TXMSS=0000</b> TXSIGs 无效 <b>TXMSS=0001,0010,0011,0100 (调制源=TIM0,TIM1,TIM2,TIM3)</b> 0000: TIMx_CH1 0001: TIMx_CH2 0010: TIMx_CH3 0011: TIMx_CH4 <b>TXMSS=1000(调制源=BUZ)</b> 调制信号为 BUZ 输出

12.5.2.22 UART2 输出调制控制寄存器 (UART2\_TXMCR)

UART2 输出调制控制寄存器 (UART2_TXMCR)																															
偏移地址: 68 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TXMLVLS		TXMSS			TXSIGs			

Reserved	Bit31-9	—	保留
TXMLVLS	Bit8	R/W	<b>TX 调制电平选择</b> 0: 低电平调制 (TX 与所选取的调制信号进行硬件或操作) 1: 高电平调制 (TX 与所选取的调制信号进行硬件与操作)
TXMSS	Bit7-4	R/W	<b>TX 调制源选择</b> 0000: 调制禁止 0001: TIM0 0010: TIM1 0011: TIM2 0100: TIM3 0111: LP16T 1000: BUZ 其余: 无调制
TXSIGs	Bit3-0	R/W	<b>TX 调制信号选择</b> <b>TXMSS=0000</b> TXSIGs 无效 <b>TXMSS=0001,0010,0011,0100 (调制源=TIM0,TIM1,TIM2,TIM3)</b> 0000: TIMx_CH1 0001: TIMx_CH2 0010: TIMx_CH3 0011: TIMx_CH4 <b>TXMSS=1000(调制源=BUZ)</b> 调制信号为 BUZ 输出



## 第13章 独立看门狗(IWDT)

### 13.1 概述

独立看门狗 IWDT，当硬件使能时，时钟强制为独立的 32KHz 低速时钟源，可用于检测软件和硬件异常，如主时钟停振，程序跑飞不再喂狗等。

当硬件使能独立看门狗时，IWDT 时钟强制变为独立的 32.768KHz 的低速时钟，且用户无法通过软件来关闭。

独立看门狗最适合独立于主程序之外，并且对时间精度要求较低场合。

### 13.2 特性

独立看门狗 IWDT 功能概述：

- ◇ 支持硬件使能和关闭看门狗
  - ◇ 芯片配置位 CFG\_WDTEN 位配置为 1，则硬件使能 IWDT
  - ◇ 芯片配置位 CFG\_WDTEN 位配置为 0，则通过软件可使能 IWDT
  - ◇ 硬件使能后不可通过软件关停
  - ◇ 硬件使能后 IWDT 时钟强制为 32KHz 低速时钟源
- ◇ IWDT 溢出时间可设定
  - ◇ 写入 IWDT\_LOAD 寄存器将重新加载看门狗
  - ◇ 溢出时产生 IWDT 复位
- ◇ IWDT 中断可唤醒 STOP1、2 模式
- ◇ 在 Debug 时进入 halting mode 时，由 IWDT\_DBGEN 关停或使能 IWDT（默认为使能），此时原来的控制信号 EN 不起作用

### 13.3 功能描述

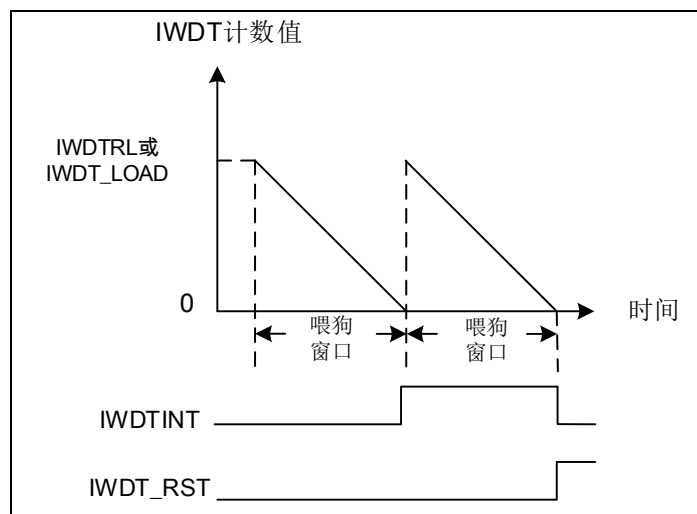


图 13-1 独立看门狗时序图

### 13.3.1 硬件看门狗

IWDT 看门狗可用于检测软件和硬件异常。可由 CFG\_IWDTEN 配置位硬件强制使能，提高系统健壮性。硬件使能时，时钟强制为独立的 32KHz 低速时钟源，保证即使系统时钟失效，IWDT 仍然工作。

当配置字 CFG\_WORD 配置为“硬件看门狗使能”时，即 CFG\_IWDTEN 位配置为 1，上电复位后看门狗立即启动(时钟固定为 32KHz 低速时钟)，IWDT 载入 IWDT\_LOAD 值(默认值为 0x0008\_0000，约 16s)，并进行递减计数，计数到 0 时，IWDT 产生中断标志，并在下一计数时钟到来时，计数器再次载入 IWDT\_LOAD 参数对应的计数初值，并继续递减计数。当计数器再次计数到 0 时，如果 IWDT 中断标志仍没有被软件清零，则 IWDT 模块将产生复位信号。

IWDT\_INTCLR 写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

#### 操作流程

1. 开启 IWDG\_IRQn 中断服务，并使能 IWDT 中断 (IWDT\_CON.IE=1)；
2. 修改 IWDT\_LOAD.LOAD 值，以确定喂狗间隔；
3. 在中断服务中检查中断标志位 (IWDT\_RIS.WDTIF) 是否被置起；
4. 如果中断标志位置起，则执行喂狗操作；

### 13.3.2 软件看门狗

当配置字配置为“看门狗禁止”时，上电看门狗不启动，但软件可配置看门狗使能。软件配置 EN=1，IWDT 计数器载入 IWDT\_LOAD 寄存器值，开始递减计数，当计数到 0 时，IWDT 产生中断标志，并在下一个计数时钟到来时，计数器再次载入 IWDT\_LOAD 寄存器值，并继续递减计数。当计数器再次计数到 0 时，如果 IWDT 中断标志仍没有被软件清零，则 IWDT 模块将产生复位信号。

IWDT\_INTCLR 写入任意值，则清除中断标志位，并重新载入计数初值，进行递减计数。

#### 操作流程

1. 开启 IWDT 中断服务，并使能 IWDT 中断 (IWDT\_CON.IE=1)；
2. 修改 IWDT\_LOAD.LOAD 值，以确定喂狗间隔；
3. 配置 IWDT\_CON.CLKS, 选择 IWDT 时钟源；
4. 配置 IWDT\_CON.EN=1,使能软件看门狗；
5. 在中断服务中检查中断标志位 (IWDT\_RIS.WDTIF) 是否被置起；
6. 如果中断标志位置起，则执行喂狗操作；

### 13.3.3 调试模式

当系统进入调试模式时，通过配置 `DBG_APB2FZ_IWDT_STOP`，可以在内核停止时暂停 IWDT 计数，以便查询系统内部状态，查询完成后，恢复程序运行。

### 13.3.4 寄存器访问保护

IWDT 模块支持寄存器写保护，配置寄存器 `IWDT_LOCK=0x1ACCE551`，可去除 IWDT 寄存器的写保护状态，此后可进行被保护的 IWDT 寄存器的写操作，否则无法写 IWDT 寄存器。

## 13.4 特殊功能寄存器

### 13.4.1 寄存器列表

IWDT 寄存器列表		
寄存器名称	偏移地址	寄存器描述
IWDT 基地址: 4004_8000 <sub>H</sub>		
IWDT_LOAD	0000 <sub>H</sub>	IWDT 计数器装载值寄存器
IWDT_VALUE	0004 <sub>H</sub>	IWDT 计数器当前值寄存器
IWDT_CON	0008 <sub>H</sub>	IWDT 控制寄存器
IWDT_INTCLR	000C <sub>H</sub>	IWDT 中断标志清除寄存器
IWDT_RIS	0010 <sub>H</sub>	IWDT 中断标志寄存器
IWDT_LOCK	0100 <sub>H</sub>	IWDT 锁定寄存器

### 13.4.2 寄存器描述

#### 13.4.2.1 IWDT计数器装载值寄存器 (IWDT\_LOAD)

IWDT 计数器装载值寄存器 (IWDT_LOAD)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00001000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															

LOAD<31:0>	bit31-0	W	IWDT 计数器重载值 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0, IWDT 不计数。
------------	---------	---	--

#### 13.4.2.2 IWDT计数器当前值寄存器 (IWDT\_VALUE)

IWDT 计数器当前值寄存器 (IWDT_VALUE)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	bit31-0	R	IWDT 计数器当前值 读取时返回 IWDT 计数器的当前计数值
-------	---------	---	-------------------------------------

### 13.4.2.3 IWDT控制寄存器 (IWDT\_CON)

IWDT 控制寄存器 (IWDT_CON)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CLKS	RSTEN	IE	EN

Reserved	bit31-4	—	保留
CLKS	bit3	R/W	<b>IWDT 计数时钟选择位</b> 0: PCLK 1: 32.768KHz 时钟 注: 时钟源配置参考时钟分配说明
RSTEN	bit2	R/W	<b>IWDT 复位使能位</b> 0: 禁止 1: 使能, IWDT 计数到 0 时, 产生复位信号, 将芯片复位
IE	bit1	R/W	<b>IWDT 中断使能位</b> 0: 禁止 1: 使能, IWDT 计数到 0 时, 产生中断标志
EN	bit0	R/W	<b>IWDT 模块使能位</b> 0: 禁止 1: 使能

### 13.4.2.4 IWDT中断标志清除寄存器 (IWDT\_INTCLR)

IWDT 中断标志清除寄存器 (IWDT_INTCLR)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCLR[31:0]																															

INTCLR<31:0>	bit31-0	W	<p>IWDT 中断标志清 0 位</p> <p>对 IWDT_INTCLR 寄存器进行任意写操作, IWDT 中断标志位均被清零, 计数器重载 IWDT_LOAD 寄存器值, 继续递减计数</p>
--------------	---------	---	---

### 13.4.2.5 IWDT中断标志寄存器 (IWDT\_RIS)

IWDT 中断标志寄存器 (IWDT_RIS)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															WDTIF

Reserved	bit31-1	—	保留
WDTIF	bit0	R	<p>IWDT 中断标志位</p> <p><b>0:</b> 未产生中断</p> <p><b>1:</b> IWDT 计数器计数到 0, 产生中断</p> <p>写寄存器 IWDT_INTCLR, 可清除 IWDT 中断标志位</p>

### 13.4.2.6 IWDT锁定寄存器 (IWDT\_LOCK)

IWDT 锁定寄存器 (IWDT_LOCK)																															
偏移地址: 100 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															LOCK

Reserved	bit31-1	—	保留
LOCK	bit0	R	IWDT 寄存器保护状态位 <b>0: IWDT 寄存器处于未保护状态</b> <b>1: IWDT 寄存器处于保护状态</b> 对 IWDT_LOCK 寄存器写入 0x1ACCE551, 被保护的寄存器处于未保护状态; 写入其它值, 处于保护状态

## 第14章 窗口看门狗(WWDT)

### 14.1 概述

窗口看门狗通常用来监视由外部干扰或不可预见的逻辑条件造成的应用程序背离运行序列而产生的软件故障。

窗口看门狗 WWDT，对于过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件没有喂狗或在禁止喂狗区内喂狗行为，防止程序跑至不可控状态。

### 14.2 特性

窗口看门狗 WWDT 功能概述：

- ◇ 支持设定喂狗禁止区
  - ◇ 通过 WDTWIN 设置喂狗禁止区
    - WDTWIN 寄存器设定为 11 时，任何区域喂狗不产生复位
  - ◇ 喂狗禁止区喂狗产生 WWDT 复位
  - ◇ 喂狗禁止区后产生 WWDT 中断
    - WWDT 中断可用作喂狗请求
- ◇ 安全可靠
  - ◇ 当 CFG\_WWDTEN 为 1 时，一旦软件使能，则只能通过复位关断
- ◇ WWDT 溢出长度可设定
  - ◇ 可通过 WWDT\_LOAD 寄存器设定
  - ◇ 溢出时产生 WWDT 复位
- ◇ WWDT 中断可用作喂狗请求，不能唤醒 STOPx 模式
- ◇ 在 Debug 时进入 halting mode 时，由 WWDT\_DBGGEN 关停或使能 WWDT（默认为使能），此时原来的控制信号 EN 不起作用

### 14.3 功能描述

#### 14.3.1 窗口看门狗

对于 WWDT 看门狗，过早或过晚喂狗都将产生 WWDT 复位，可用于检测软件的过晚或过早行为，防止程序跑至不可控状态。如中断异常，程序不断进入一个带喂狗指令的子程序的情况，可通过 WWDT 复位消除。

用户可根据程序正常执行的时间设定喂狗窗口，可检测程序未按正常次序执行，跳过某些程序的异常情况。当选用 PCLK2 作为 WWDT 时钟时，可精确地检测异常情况。

#### 窗口看门狗时钟源

用户可根据自己喂狗精度的需要，通过配置 WWDT\_CON.CLKS 来选择合适的时钟源。

#### 操作流程

1. 系统上电后，窗口看门狗不启动，配置 WWDT\_LOAD 寄存器设置计数初值，配置



- WWDT\_CON.CLKS 选择时钟源，配置 WWDT\_CON.IE 使能中断，配置 WWDT\_CON.EN 使能窗口看门狗；
- WWDT 计数器载入 WWDT\_LOAD 寄存器值的 1/4，并开始递减计数，当计数到 0 时，窗口计数器加 1；
  - 在下一个计数时钟到来时，计数器再次载入 WWDT\_LOAD 寄存器值的 1/4，并开始递减计数，当计数到 0 时，窗口计数器加 1，重复此过程直到窗口计数器计到 4；
  - 若 WWDT\_CON.WWDTWIN 设置为 00，则窗口计数器计到 1 时，WWDT 产生中断标志；
  - 若 WWDT\_CON.WWDTWIN 设置为 01，则窗口计数器计到 2 时，WWDT 产生中断标志；
  - 若 WWDT\_CON.WWDTWIN 设置为 10，则窗口计数器计到 3 时，WWDT 产生中断标志；
  - 若 WWDT 产生中断后，直至窗口计数器计数到 4 之前(即累计计数等于 WWDT\_LOAD 寄存器的值)，如果没有在喂狗窗口内进行喂狗，则 WWDT 模块将产生复位信号，如下图所示；
  - 若喂狗窗口内寄存器 WWDT\_INTCLR 写入任意值，将清除中断标志位，WWDT 计数器重新载入计数初值并进行递减计数；
  - 若在喂狗禁止区内进行喂狗操作，则会产生 WWDT 复位，如下图所示。

注：若配置子中 CFG\_WWDTEN 位配置为 1，则软件使能窗口看门狗之后，不可再通过软件关闭窗口看门狗

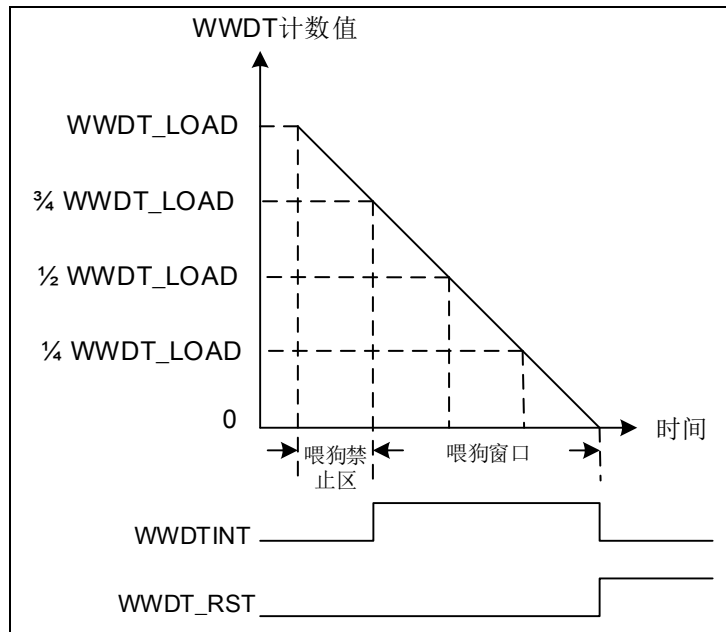


图 14-1 窗口看门狗中断和溢出复位产生时序图 (WWDTWIN 设定为 25%)

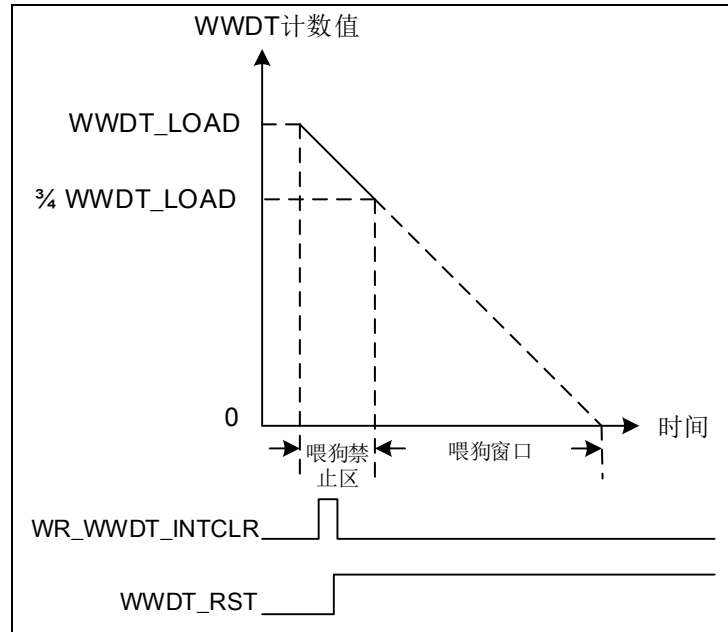


图 14-2 错误的喂狗时序图 (WWDTWIN 设定为 25%)

### 14.3.2 调试模式

当系统进入调试模式时，通过配置 `DBG_APB2FZ.WWDT_STOP`，可以在内核停止时，暂停 WWDT 计数，以便查询系统内部状态，查询完成后，恢复程序运行。

### 14.3.3 寄存器访问保护

`WWDT_LOAD`、`WWDT_CON`、`WWDT_INTCLR` 这三个寄存器具有写保护功能，对其修改时，必须先向 `WWDT_LOCK` 寄存器写入 `0x1ACCE551` 来移除写保护；当 `WWDT_LOCK` 写入其它任意值时，寄存器重新被保护。

## 14.4 特殊功能寄存器

### 14.4.1 寄存器列表

WWDT 寄存器列表		
寄存器名称	偏移地址	寄存器描述
WWDT 基地址: 4004_6000 <sub>H</sub>		
WWDT_LOAD	0000 <sub>H</sub>	WWDT 计数器装载值寄存器
WWDT_VALUE	0004 <sub>H</sub>	WWDT 计数器当前值寄存器
WWDT_CON	0008 <sub>H</sub>	WWDT 控制寄存器
WWDT_INTCLR	000C <sub>H</sub>	WWDT 中断标志清除寄存器
WWDT_RIS	0010 <sub>H</sub>	WWDT 中断标志寄存器
WWDT_LOCK	0100 <sub>H</sub>	WWDT 锁定寄存器

## 14.4.2 寄存器描述

### 14.4.2.1 WWDT计数器装载值寄存器 (WWDT\_LOAD)

WWDT 计数器装载值寄存器 (WWDT_LOAD)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000010_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD[31:0]																															

LOAD<31:0>	bit31-0	W	WWDT 计数器重载值 计数范围 0x0000_0001~0xFFFF_FFFF。如果为 0, WWDT 不计数。
------------	---------	---	--

### 14.4.2.2 WWDT计数器当前值寄存器 (WWDT\_VALUE)

WWDT 计数器当前值寄存器 (WWDT_VALUE)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00111111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VALUE	bit31-0	R	WWDT 计数器当前值 读取时返回 WWDT 计数器的当前计数值, 其中高两位为窗口计数器当前值
-------	---------	---	---

### 14.4.2.3 WWDT控制寄存器 (WWDT\_CON)

WWDT 控制寄存器 (WWDT_CON)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								WWDTWIN		CLKS	RSTEN	IE	EN		

Reserved	bit31-6	—	保留
WWDTWIN	bit5-4	R/W	<b>WWDT 禁止喂狗窗口选择位</b> 00: 25%窗口内禁止喂狗, 窗口内喂狗产生复位 01: 50%窗口内禁止喂狗, 窗口内喂狗产生复位 10: 75%窗口内禁止喂狗, 窗口内喂狗产生复位 11: 不禁止喂狗, 喂狗将使看门狗计数器重载
CLKS	bit3	R/W	<b>WWDT 计数时钟选择位</b> 0: PCLK 1: LRC 时钟 (约 32KHz)
RSTEN	bit2	R/W	<b>WWDT 复位使能位</b> 0: 禁止 1: 使能, WWDT 计数到 0 时, 产生复位信号, 将芯片复位
IE	bit1	R/W	<b>WWDT 中断使能位</b> 0: 禁止 1: 使能, WWDT 计数到 0 时, 产生中断标志
EN	bit0	W1	<b>WWDT 模块使能位</b> 0: 无效 1: 使能

#### 14.4.2.4 WWDT中断标志清除寄存器 (WWDT\_INTCLR)

WWDT 中断标志清除寄存器 (WWDT_INTCLR)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCLR[31:0]																															

INTCLR<31:0>	bit31-0	W	WWDT 中断标志清 0 位 对 WWDT_INTCLR 寄存器进行任意写操作, WWDT 中断标志位均被清零, 计数器重载 WWDT_LOAD 寄存器值, 继续递减计数
--------------	---------	---	--

#### 14.4.2.5 WWDT中断标志寄存器 (WWDT\_RIS)

WWDT 中断标志寄存器 (WWDT_RIS)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															WWDTIF

Reserved	bit31-1	—	保留
WWDTIF	bit0	R	WWDT 中断标志位 <b>0</b> : 未产生中断 <b>1</b> : WWDT 计数器计数到 0, 产生中断 写寄存器 WWDT_INTCLR, 可清除 WWDT 中断标志位

### 14.4.2.6 WWDT锁定寄存器 (WWDT\_LOCK)

WWDT 锁定寄存器 (WWDT_LOCK)																															
偏移地址: 100 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															LOCK

Reserved	bit31-1	—	保留
LOCK	bit0	R	WWDT 寄存器保护状态位 <b>0: WWDT 寄存器处于未保护状态</b> <b>1: WWDT 寄存器处于保护状态</b> 对 WWDT_LOCK 寄存器写入 0x1ACCE551, 被保护的寄存器处于未保护状态; 写入其它值, 处于保护状态

## 第15章 通用IO及端口控制(GPIO)

### 15.1 概述

每个通用 GPIO 端口包含 16 个独立的引脚。这些引脚可单独配置为输入或输出。每个引脚可额外地可配置为开漏输出或带滤波输入模式，配置为输出时可选择每个引脚的驱动强度。

GPIO 引脚可复用为外设功能端口，例如 PWM 输出口或 UART 通信口，每个外设均支持复用到多个引脚上。GPIO 端口支持最多 16 个异步外部中断，可被配置到任何一个 IO 引脚上。并且，GPIO 端口支持通过 PIS 触发其他外设。

### 15.2 特性

- ◇ 可配置为输入或输出
- ◇ 输出模式可配置
  - ◇ 推挽/开源/开漏
  - ◇ 上拉/下拉
- ◇ 输入模式
  - ◇ 端口浮空
  - ◇ 上拉/下拉
  - ◇ 模拟端口
- ◇ 支持端口输出数据的复位、置位或取反，可按位操作
- ◇ 支持复用为外设功能端口
- ◇ 输出驱动能力可配置：四种驱动能力选择
- ◇ 支持 16 个外部输入中断
- ◇ 支持端口配置写保护功能



### 15.3 结构框图

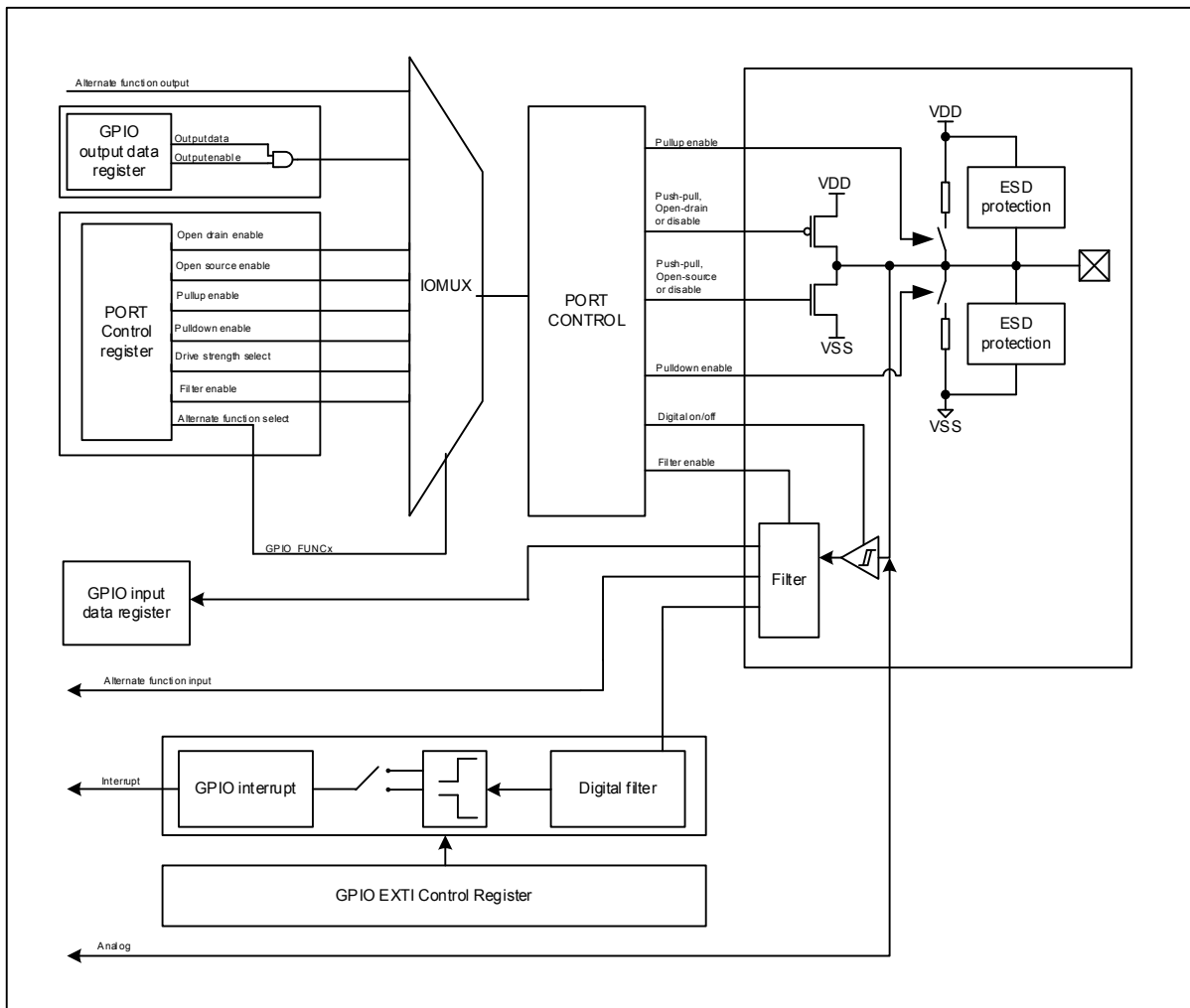


图 15-1 GPIO 结构框图

### 15.4 功能描述

#### 15.4.1 端口控制寄存器

每个 GPIO 有 16 个相对独立的引脚，每个引脚都可以通过寄存器自由配置。

通过配置 GPIO\_MODE，可选择相应端口的模式，可配置为输入模式，输出模式和端口关闭模式。选择输出模式时，端口输入同样有效。选择端口关闭模式时，相应端口可被复用为模拟功能。

通过配置 GPIO\_PUPD，可启用相应端口的上拉或/和下拉电阻。

通过配置 GPIO\_ODOS，可将相应端口输出方式配置为推挽、源极开路 and 漏极开路三种模式。在配置为源极开路或漏极开路模式时，可启用相应端口的上拉或下拉，以保证正常输出，也可在端口外部连接上拉或下拉电阻。

通过配置 GPIO\_ODRV，可选择相应端口的输出驱动能力，已满足不同的负载要求。

通过配置 GPIO\_FLT，可启用相应端口的输入滤波功能，可滤除外部引线上高频信号干扰或毛刺。若输入需要较高的实时性，建议关闭输入滤波功能。

通过配置 GPIO\_TYPE，可选择相应端口的输入类型，可选择 TTL 或 SMIT 两种模式。

通过配置 GPIO\_FUNC，可选择相应 GPIO 的复用功能，可选择 FUNC\_ALT0 ~ FUNC\_ALT7。复用功能 GPIO 为输入时必须配置为输入模式，且需外部驱动；复用功能 GPIO 为输出时必须配置为输出模式，推挽或开漏；复用功能 GPIO 为双向模式时，端口必须配置为输出模式，推挽或开漏。

通过配置 GPIO\_LOCK，可锁定相应端口的控制寄存器数值。直到下一次 CPU 复位锁定才可被解除。端口数据寄存器不受锁定的控制。

#### 15.4.2 端口数据寄存器

软件可通过读取 GPIO\_DIN 才获知端口的电平状态，若相应端口输入滤波被启用，则读到的是端口滤波滤波之后的状态。

通过配置 GPIO\_DOUT，可选择端口输出电平值，若端口模式已配置为输出，则该值所对应的电平会在管脚上立即生效。

通过配置 GPIO\_BSRR，可按位改写端口输出电平值。若同时将某位置位和复位，则置位的优先级更高。

通过配置 GPIO\_BSRR，可按位改写端口输出电平值。对置位寄存器某些位进行写入 1 可置位相应端口，写入 0 的位不会影响相应端口的输出电平。对复位寄存器某些位进行写入 1 可复位相应端口，写入 0 的位不会影响相应端口的输出电平。若同时将某位置位和复位，则置位的优先级更高。

通过配置 GPIO\_BIR，可按位翻转端口输出电平值。对翻转寄存器某些位进行写入 1 可将相应端口电平值翻转，写入 0 的位不会影响相应端口的输出电平。

#### 15.4.3 外部端口中断

通过配置 GPIOEXTI\_SER，可将外部端口中断使能位置位，其中低半字为上升沿中断使能位，高半字为下降沿中断使能位。通过配置 GPIOEXTI\_CER，可将外部端口中断使能位清除。

通过配置 GPIO\_EXTISFR，可将外部端口中断标志位置位，其中低半字为上升沿中断标志位，高半字为下降沿中断标志位。通过配置 GPIO\_EXTICFR，可将外部端口中断标志位清除。

通过配置 GPIO\_EXTIPSR0 和 GPIO\_EXTIPSR1 可选择外部端口中断的端口，每个外部中断可选择 8 个不同的 GPIO 口。

通过配置 GPIO\_EXTIFLTCR 中 FLTEN 位可启用相应外部端口中断滤波功能，配置 GPIO\_EXTIFLTCR 中 FLTCKS 位可选滤波时钟，配置 GPIO\_EXTIFLTCR 中 FLTSEL 位可选滤波时间。

#### 15.4.4 通用GPIO配置

每个 GPIO 都可以由软件设置为输出模式或输入模式，也可以复用为外设功能接口。所有

GPIO 端口都有内部上拉或下拉，应用中可以激活也可以断开。

端口配置表				
配置模式		MODEn[1:0]	PUPDn[1:0]	ODOSn[1:0]
输出	推挽输出	1x	xx	0x/11
	开漏输出	1x	xx	10
输入	浮空输入	01	00	xx
	上拉输入	01	01	xx
	下拉输入	01	10	xx
	模拟输入	01	禁止	xx

表 15-1 端口配置表

在输出模式中，还可以软件方式设置每个 GPIO 的驱动能力。输入模式中，可以软件方式设置每个 GPIO 的滤波特性。还可以软件选择 GPIO 管脚类型，有 CMOS 和 TTL 两种选择。

ODRVn[1:0]	意义
00	普通驱动电流（10mA）
01	强电流驱动（12mA）
10	普通电流驱动（10mA）
11	强电流驱动（12mA）

表 15-2 端口拉电流驱动表

ODRVn[1:0]	意义
00	普通驱动电流（10mA）
01	强电流驱动（20mA）
10	普通电流驱动（10mA）
11	强电流驱动（20mA）

表 15-3 端口灌电流驱动表

#### 15.4.5 外部中断与唤醒

外部中断可以配达每个 GPIO，且配置为外部中断的 GPIO 必须设置为输入模式。

每个外部中断通道可以独立的配置输入类型和对应的触发事件（上升沿触发、下降沿触发或双边触发），都可以独立的设置触发或禁止。

**EXTI 主要特性如下：**

- ◇ 每个外部中断都有独立的触发或禁止设置
- ◇ 每个中断通道有独立的状态标识位
- ◇ 支持多达 16 个外部中断请求
- ◇ 独立设置外部中断滤波特性

外部中断映像如下：

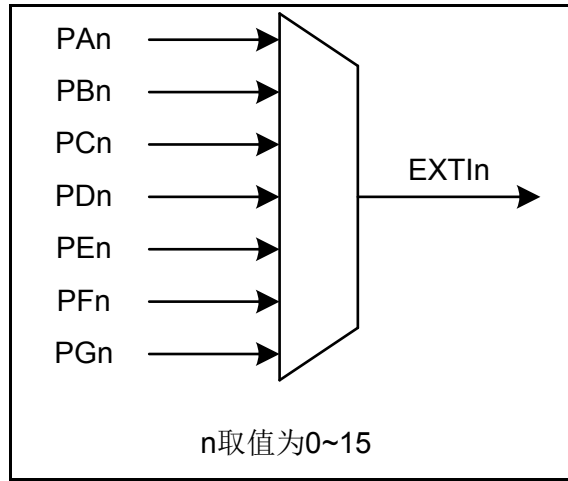


图 15-2 外中断 GPIO 映像

**应用说明:**

要产生中断，必须配置好中断端口并使能，然后根据需要通过触发事件寄存器来设置边沿检测，另外根据应用需求可以设置输入滤波。当外部端口发生了预期的边沿时将产生一个中断请求，外部中断标志寄存器中对应的位随之被置 1，此时在外部中断标志清零寄存器对应标志位置 1 可以清楚中断请求及对应标志位。

外部中断配置步骤如下:

1. 将对应 GPIO 端口配置为输入模式。
2. 配置外部中断端口选择寄存器 0 和外部中断端口选择寄存器 1 选择相应的 GPIO 端口。
3. 配置外部中断上升沿触发使能寄存器设置上升沿触发事件，配置外部中断下降沿触发使能寄存器设置下降沿触发事件。
4. 配置外部中断滤波控制寄存器设置需求对应的滤波特性。
5. 配置外部中断使能寄存器对应的中断使能位，使得外部中断可以有效响应。

**15.4.6 外设功能端口复用**

使用复用功能时必须先对 GPIO 端口复用功能寄存器 0 或 GPIO 端口复用功能寄存器 1 进行配置，对应的配置参数请查看数据手册管脚功能复用表。

- ◇ 当复用功能为输入时，端口必须配置为输入模式（浮空、上拉或下拉），输入引脚由外部驱动。
- ◇ 当复用功能为输出时，端口必须配置为输出模式（推挽、开漏或开源）。在配置开漏模式和开源模式时，GPIO 输入功能有效，此时可以用作双向模式。

如果端口复用功能为输出，则引脚会和片上外设的输出信号连接，如果对应外设没有被激活，GPIO 的输出信号将不确定。

**15.4.7 GPIO 锁定**

芯片 GPIO 锁定机制的处理是将端口配置冻结。当一个端口执行了锁定程序后，对应 GPIO 控制寄存器数值将被锁定，在下次复位之前，不能再被更改。

### 15.4.8 GPIO输入配置

---

当 GPIO 端口配置为输入模式时

- ◇ 输出缓冲器被禁止。
- ◇ 根据输入模式配置参数的不同，连接内部上拉、下拉。
- ◇ 输入数据采样到 GPIO 端口输入数据寄存器
- ◇ 访问 GPIO 端口输入数据寄存器即可得到 GPIO<sub>n</sub> 状态数据

### 15.4.9 GPIO输出配置

---

当 GPIO 端口配置为输出模式时：

- ◇ 输出缓冲器被激活
- ◇ 根据配置参数的不同，连接内部上拉、下拉
- ◇ 输入数据采样到 GPIO 端口输入数据寄存器
- ◇ 在开漏模式下，访问输入数据寄存器得到当前 GPIO<sub>n</sub> 的状态数据。
- ◇ 在推挽模式下，访问输出数据寄存器得到最后一次写的值。

### 15.4.10 模拟输入配置

---

当 GPIO 端口配置为模拟输入模式时：

- ◇ 输出缓冲器被禁止
- ◇ 施密特输入触发禁止
- ◇ 内部上拉或下拉禁止
- ◇ 访问输入数据寄存器得到的数据为全 0

## 15.5 特殊功能寄存器

### 15.5.1 寄存器列表

GPIO 寄存器列表		
名称	偏移地址	描述
GPIO0 基地址: 4008_4000 <sub>H</sub>		
GPIO1 基地址: 4008_4040 <sub>H</sub>		
GPIO_DIN	000 <sub>H</sub>	GPIO 端口输入数据寄存器
GPIO_DOUT	004 <sub>H</sub>	GPIO 端口输出数据寄存器
GPIO_BSRR	008 <sub>H</sub>	GPIO 端口置位和复位寄存器
GPIO_BIR	00C <sub>H</sub>	GPIO 端口翻转寄存器
GPIO_MODE	010 <sub>H</sub>	GPIO 端口模式寄存器
GPIO_ODOS	014 <sub>H</sub>	GPIO 端口开漏和开源寄存器
GPIO_PUPD	018 <sub>H</sub>	GPIO 端口上拉和下拉寄存器
GPIO_ODRV	01C <sub>H</sub>	GPIO 端口输出驱动寄存器
GPIO_FLT	020 <sub>H</sub>	GPIO 端口滤波寄存器
GPIO_TYPE	024 <sub>H</sub>	GPIO 端口类型寄存器
GPIO_FUNC0	028 <sub>H</sub>	GPIO 端口复用功能寄存器 0
GPIO_FUNC1	02C <sub>H</sub>	GPIO 端口复用功能寄存器 1
GPIO_LOCK	030 <sub>H</sub>	GPIO 端口锁定寄存器

GPIOEXTI 寄存器列表		
名称	偏移地址	描述
GPIOEXTI 基地址: 4008_4300 <sub>H</sub>		
GPIO_EXTIRER	000 <sub>H</sub>	外部中断上升沿触发使能寄存器
GPIO_EXTIFER	008 <sub>H</sub>	外部中断下降沿触发使能寄存器
GPIO_EXTIEN	010 <sub>H</sub>	外部中断使能寄存器
GPIO_EXTIFLAG	018 <sub>H</sub>	外部中断标志寄存器
GPIO_EXTISFR	020 <sub>H</sub>	外部中断标志置位寄存器
GPIO_EXTICFR	028 <sub>H</sub>	外部中断标志清零寄存器
GPIO_EXTIPSR0	030 <sub>H</sub>	外部中断端口选择寄存器 0
GPIO_EXTIPSR1	034 <sub>H</sub>	外部中断端口选择寄存器 1
GPIO_EXTIFLTCR	040 <sub>H</sub>	外部中断滤波控制寄存器

## 15.5.2 寄存器描述

### 15.5.2.1 GPIO端口输入数据寄存器 (GPIO\_DIN)

GPIO 端口输入数据寄存器 (GPIO_DIN)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIN15	DIN14	DIN13	DIN12	DIN11	DIN10	DIN9	DIN8	DIN7	DIN6	DIN5	DIN4	DIN3	DIN2	DIN1	DIN0

Reserved	Bit 31-16	—	保留
DIN<y>	Bit 15-0	R	GPIO 输入数据

### 15.5.2.2 GPIO端口输出数据寄存器 (GPIO\_DOUT)

GPIO 端口输出数据寄存器 (GPIO_DOUT)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DOUT15	DOUT14	DOUT13	DOUT12	DOUT11	DOUT10	DOUT9	DOUT8	DOUT7	DOUT6	DOUT5	DOUT4	DOUT3	DOUT2	DOUT1	DOUT0

Reserved	Bit 31-16	—	保留
DOUT<y>	Bit 15-0	R/W	GPIO 输出数据

### 15.5.2.3 GPIO端口置位和复位寄存器 (GPIO\_BSRR)

GPIO 端口置位和复位寄存器 (GPIO_BSRR)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR15	BRR14	BRR13	BRR12	BRR11	BRR10	BRR9	BRR8	BRR7	BRR6	BRR5	BRR4	BRR3	BRR2	BRR1	BRR0	BSR15	BSR14	BSR13	BSR12	BSR11	BSR10	BSR9	BSR8	BSR7	BSR6	BSR5	BSR4	BSR3	BSR2	BSR1	BSR0

BRR<y>	Bit 31-16	W	<b>GPIO 复位控制位</b> 0: 无操作 1: 相应位复位 注: 如果同时对 GPIO 置位和复位, 置位的优先级更高
BSR<y>	Bit 15-0	W	<b>GPIO 置位控制位</b> 0: 无操作 1: 相应位置位

### 15.5.2.4 GPIO端口翻转寄存器 (GPIO\_BIR)

GPIO 端口翻转寄存器 (GPIO_BIR)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIR15	BIR14	BIR13	BIR12	BIR11	BIR10	BIR9	BIR8	BIR7	BIR6	BIR5	BIR4	BIR3	BIR2	BIR1	BIR0

Reserved	Bit 31-16	—	保留
BIR<y>	Bit 15-0	W	<b>GPIO 翻转控制位</b> 0: 无操作 1: 相应位翻转



### 15.5.2.5 GPIO端口模式寄存器 (GPIO\_MODE)

GPIO 端口模式寄存器 (GPIO_MODE)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE15	MODE14	MODE13	MODE12	MODE11	MODE10	MODE9	MODE8	MODE7	MODE6	MODE5	MODE4	MODE3	MODE2	MODE1	MODE0																

MODE<y>	Bit 31-0	RW	<b>GPIO 模式控制位</b> 00: 预留 01: 端口输入 1x: 端口输出
---------	----------	----	---

### 15.5.2.6 GPIO端口开漏和开源寄存器 (GPIO\_ODOS)

GPIO 端口置位和复位寄存器 (GPIO_ODOS)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODOS15	ODOS14	ODOS13	ODOS12	ODOS11	ODOS10	ODOS9	ODOS8	ODOS7	ODOS6	ODOS5	ODOS4	ODOS3	ODOS2	ODOS1	ODOS0																

ODOS<y>	Bit 31-0	RW	<b>GPIO 开漏和开源控制位</b> 0x: 推挽输出 10: 开漏输出 11: 推挽输出
---------	----------	----	--

### 15.5.2.7 GPIO端口上拉和下拉寄存器 (GPIO\_PUPD)

GPIO 端口上拉和下拉寄存器 (GPIO_PUPD)																																
偏移地址: 18 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PUPD15	PUPD14	PUPD13	PUPD12	PUPD11	PUPD10	PUPD9	PUPD8	PUPD7	PUPD6	PUPD5	PUPD4	PUPD3	PUPD2	PUPD1	PUPD0																	

PUPD<y>	Bit 31-0	R/W	<b>GPIO 上拉和下拉控制位</b> 00: 无上拉和下拉 01: 上拉 10: 下拉 11: 同时上拉和下拉
---------	----------	-----	---

### 15.5.2.8 GPIO端口输出驱动寄存器 (GPIO\_ODRV)

GPIO 端口 PMOS 输出驱动寄存器 (GPIO_ODRV)																																
偏移地址: 1C <sub>H</sub>																																
复位值: 10101010_10101010_10101010_10101010 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PODRV15	PODRV14	PODRV13	PODRV12	PODRV11	PODRV10	PODRV9	PODRV8	PODRV7	PODRV6	PODRV5	PODRV4	PODRV3	PODRV2	PODRV1	PODRV0																	

ODRV<y>	Bits 31-0	R/W	<b>GPIO 输出驱动控制位</b> x0: 普通电流驱动 x1: 强电流驱动)
---------	-----------	-----	---

### 15.5.2.9 GPIO端口滤波寄存器 (GPIO\_FLT)

GPIO 端口滤波寄存器 (GPIO_FLT)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FLT15	FLT14	FLT13	FLT12	FLT11	FLT10	FLT9	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	FLT0

Reserved	Bit 31-16	—	保留
FLT<y>	Bit 15-0	R/W	<b>GPIO 滤波控制位</b> 0: 2ns 输入滤波 1: 20ns 输入滤波

### 15.5.2.10 GPIO端口类型寄存器 (GPIO\_TYPE)

GPIO 端口类型寄存器 (GPIO_TYPE)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															

Reserved	Bit 31-16	—	保留
Reserved	Bits 15-0	—	保留

### 15.5.2.11 GPIO端口复用功能寄存器 0 (GPIO\_FUNC0)

GPIO 端口复用功能寄存器 0 (GPIO_FUNC0)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO7				FSEL_IO6				FSEL_IO5				FSEL_IO4				FSEL_IO3				FSEL_IO2				FSEL_IO1				FSEL_IO0			

FSEL_IO7	Bits 31-28	R/W	<b>GPIO&lt;7&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO6	Bits 27-24	R/W	<b>GPIO&lt;6&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO5	Bits 23-20	R/W	<b>GPIO&lt;5&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO4	Bits 19-16	R/W	<b>GPIO&lt;4&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO3	Bits 15-12	R/W	<b>GPIO&lt;3&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO2	Bits 11-8	R/W	<b>GPIO&lt;2&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO1	Bits 7-4	R/W	<b>GPIO&lt;1&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO0	Bits 3-0	R/W	<b>GPIO&lt;0&gt;功能复用选择位</b> 请查看管脚功能复用表

### 15.5.2.12 GPIO端口复用功能寄存器 1 (GPIO\_FUNC1)

GPIO 端口复用功能寄存器 1 (GPIO_FUNC1)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSEL_IO15				FSEL_IO14				FSEL_IO13				FSEL_IO12				FSEL_IO11				FSEL_IO10				FSEL_IO9				FSEL_IO8			

FSEL_IO15	Bits 31-28	R/W	<b>GPIO&lt;15&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO14	Bits 27-24	R/W	<b>GPIO&lt;14&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO13	Bits 23-20	R/W	<b>GPIO&lt;13&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO12	Bits 19-16	R/W	<b>GPIO&lt;12&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO11	Bits 15-12	R/W	<b>GPIO&lt;11&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO10	Bits 11-8	R/W	<b>GPIO&lt;10&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO9	Bits 7-4	R/W	<b>GPIO&lt;9&gt;功能复用选择位</b> 请查看管脚功能复用表
FSEL_IO8	Bits 3-0	R/W	<b>GPIO&lt;8&gt;功能复用选择位</b> 请查看管脚功能复用表

### 15.5.2.13 GPIO端口锁定寄存器 (GPIO\_LOCK)

GPIO 端口锁定寄存器 (GPIO_LOCK)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY																LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0

KEY	Bit 31-16	W	<b>GPIO 锁定寄存器关键码</b> 注: 检测到写入关键码 0x55AA, 才可对 LOCK<y> 进行写操作, 读该位始终为 0
LOCK<y>	Bit 15-0	RW	<b>GPIO&lt;y&gt;锁定控制位</b> 0: 相应端口未锁定 1: 相应端口锁定 注: 被锁定相应端口只允许改变输出数据, LOCK<y>一旦被置位, 必须等到下一次 CPU 复位才被清除。

### 15.5.2.14 外部中断上升沿触发使能寄存器 (GPIO\_EXTIRER)

外部中断上升沿触发使能寄存器 (GPIO_EXTIRER)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIRER15	EXTIRER14	EXTIRER13	EXTIRER12	EXTIRER11	EXTIRER10	EXTIRER9	EXTIRER8	EXTIRER7	EXTIRER6	EXTIRER5	EXTIRER4	EXTIRER3	EXTIRER2	EXTIRER1	EXTIRER0

Reserved	Bit 31-16	—	保留
EXTIRER<y>	Bit 15-0	R/W	EXTI<y>上升沿触发使能位 0: 禁止 1: 使能

### 15.5.2.15 外部中断下降沿触发使能寄存器 (GPIO\_EXTIFER)

外部中断下降沿触发使能寄存器 (GPIO_EXTIFER)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFER15	EXTIFER14	EXTIFER13	EXTIFER12	EXTIFER11	EXTIFER10	EXTIFER9	EXTIFER8	EXTIFER7	EXTIFER6	EXTIFER5	EXTIFER4	EXTIFER3	EXTIFER2	EXTIFER1	EXTIFER0

Reserved	Bit 31-16	—	保留
EXTIFER<y>	Bit 15-0	R/W	EXTI<y>下降沿触发使能位 0: 禁止 1: 使能

### 15.5.2.16 外部中断使能寄存器 (GPIO\_EXTIEN)

外部中断使能寄存器 (GPIO_EXTIEN)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIEN15	EXTIEN14	EXTIEN13	EXTIEN12	EXTIEN11	EXTIEN10	EXTIEN9	EXTIEN8	EXTIEN7	EXTIEN6	EXTIEN5	EXTIEN4	EXTIEN3	EXTIEN2	EXTIEN1	EXTIEN0

Reserved	Bit 31-16	—	保留
EXTIEN<y>	Bit 15-0	R/W	<b>EXTI&lt;y&gt;中断使能位</b> 0: 禁止 1: 使能

### 15.5.2.17 外部中断标志寄存器 (GPIO\_EXTIFLAG)

外部中断标志寄存器 (GPIO_EXTIFLAG)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTIFLAG15	EXTIFLAG14	EXTIFLAG13	EXTIFLAG12	EXTIFLAG11	EXTIFLAG10	EXTIFLAG9	EXTIFLAG8	EXTIFLAG7	EXTIFLAG6	EXTIFLAG5	EXTIFLAG4	EXTIFLAG3	EXTIFLAG2	EXTIFLAG1	EXTIFLAG0

Reserved	Bit 31-16	—	保留
EXTIFLAG<y>	Bit 15-0	R	<b>EXTI&lt;y&gt;中断状态位</b> 0: 未检测到有效中断 1: 检测到有效中断



### 15.5.2.18 外部中断标志置位寄存器 (GPIO\_EXTISFR)

外部中断标志置位寄存器 (GPIO_EXTISFR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTISFR15	EXTISFR14	EXTISFR13	EXTISFR12	EXTISFR11	EXTISFR10	EXTISFR9	EXTISFR8	EXTISFR7	EXTISFR6	EXTISFR5	EXTISFR4	EXTISFR3	EXTISFR2	EXTISFR1	EXTISFR0

Reserved	Bit 31-16	—	保留
EXTISFR<y>	Bit 15-0	W1	<b>EXTI&lt;y&gt;中断标志位置位</b> 0: 无操作 1: 置位中断标志位 注: 对该位写 1 将中断标志置位, 写 0 无效

### 15.5.2.19 外部中断标志清零寄存器 (GPIO\_EXTICFR)

外部中断标志清零寄存器 (GPIO_EXTICFR)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTICFR15	EXTICFR14	EXTICFR13	EXTICFR12	EXTICFR11	EXTICFR10	EXTICFR9	EXTICFR8	EXTICFR7	EXTICFR6	EXTICFR5	EXTICFR4	EXTICFR3	EXTICFR2	EXTICFR1	EXTICFR0

Reserved	Bit 31-16	—	保留
EXTICFR<y>	Bit 15-0	W1	<b>EXTI&lt;y&gt;中断标志位清零</b> 0: 无操作 1: 清零中断标志位 注: 对该位写 1 将中断标志复位, 写 0 无效

15.5.2.20 外部中断端口选择寄存器 0 (GPIO\_EXTIPSR0)

外部中断端口选择寄存器 0 (GPIO_EXTIPSR0)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS7			Reserved	EXTIS6			Reserved	EXTIS5			Reserved	EXTIS4			Reserved	EXTIS3			Reserved	EXTIS2			Reserved	EXTIS1			Reserved	EXTIS0		

Reserved	Bit 31	—	保留
EXTIS7	Bits 30-28	R/W	<b>EXTI&lt;7&gt;端口选择位</b> 000: PA7 001: PB7 其它: 无效
Reserved	Bit 27	—	保留
EXTIS6	Bits 26-24	R/W	<b>EXTI&lt;6&gt;端口选择位</b> 000: PA6 001: PB6 其它: 无效
Reserved	Bit 23	—	保留
EXTIS5	Bits 22-20	R/W	<b>EXTI&lt;5&gt;端口选择位</b> 000: PA5 001: PB5 其它: 无效
Reserved	Bit 19	—	保留
EXTIS4	Bits 18-16	R/W	<b>EXTI&lt;4&gt;端口选择位</b> 000: PA4 001: PB4 其它: 无效
Reserved	Bit 15	—	保留
EXTIS3	Bits 14-12	R/W	<b>EXTI&lt;3&gt;端口选择位</b> 000: PA3 001: PB3 其它: 无效
Reserved	Bit 11	—	保留
EXTIS2	Bits 10-8	R/W	<b>EXTI&lt;2&gt;端口选择位</b> 000: PA2 001: PB2 其它: 无效
Reserved	Bit 7	—	保留
EXTIS1	Bits 6-4	R/W	<b>EXTI&lt;1&gt;端口选择位</b>

			000: PA1 001: PB1 其它: 无效
Reserved	Bit 3	—	保留
EXTIS0	Bits 2-0	R/W	<b>EXTI&lt;0&gt;端口选择位</b> 000: PA0 001: PB0 其它: 无效

15.5.2.21 外部中断端口选择寄存器 1 (GPIO\_EXTIPSR1)

外部中断端口选择寄存器 1 (GPIO_EXTIPSR1)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	EXTIS15			Reserved	EXTIS14			Reserved	EXTIS13			Reserved	EXTIS12			Reserved	EXTIS11			Reserved	EXTIS10			Reserved	EXTIS9			Reserved	EXTIS8		

Reserved	Bit 31	—	保留
EXTIS15	Bits 30-28	R/W	<b>EXTI&lt;15&gt;端口选择位</b> 000: PA15 001: PB15 其它: 无效
Reserved	Bit 27	—	保留
EXTIS14	Bits 26-24	R/W	<b>EXTI&lt;14&gt;端口选择位</b> 000: PA14 001: PB14 其它: 无效
Reserved	Bit 23	—	保留
EXTIS13	Bits 22-20	R/W	<b>EXTI&lt;13&gt;端口选择位</b> 000: PA13 001: PB13 其它: 无效
Reserved	Bit 19	—	保留
EXTIS12	Bits 18-16	R/W	<b>EXTI&lt;12&gt;端口选择位</b> 000: PA12 001: PB12 其它: 无效
Reserved	Bit 15	—	保留
EXTIS11	Bits 14-12	R/W	<b>EXTI&lt;11&gt;端口选择位</b> 000: PA11 001: PB11 其它: 无效
Reserved	Bit 11	—	保留
EXTIS10	Bits 10-8	R/W	<b>EXTI&lt;10&gt;端口选择位</b> 000: PA10 001: PB10 其它: 无效
Reserved	Bit 7	—	保留
EXTIS9	Bits 6-4	R/W	<b>EXTI&lt;9&gt;端口选择位</b>

			000: PA9 001: PB9 其它: 无效
Reserved	Bit 3	—	保留
EXTIS8	Bits 2-0	R/W	<b>EXTI&lt;8&gt;端口选择位</b> 000: PA8 001: PB8 其它: 无效

15.5.2.22 外部中断滤波控制寄存器 (GPIO\_EXTIFLTCR)

外部中断滤波控制寄存器 (GPIO_EXTIFLTCR)																															
偏移地址: 40 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						FLTCKS		FLTSEL								FLTEN15	FLTEN14	FLTEN13	FLTEN12	FLTEN11	FLTEN10	FLTEN9	FLTEN8	FLTEN7	FLTEN6	FLTEN5	FLTEN4	FLTEN3	FLTEN2	FLTEN1	FLTEN0

Reserved	Bits 31-26	—	保留
FLTCKS	Bit 25-24	R/W	EXTI 端口中断去抖滤波时钟选择 保留 注: 滤波时钟由系统低速时钟决定。 (CMU_CSR->LSCK_CMD)
FLTSEL	Bit 23-16	R/W	EXTI 端口中断去抖滤波选择 滤波时间 = (FLTSEL<7:0> + 1) × 2 个时钟周期
FLTEN<y>	Bit 15-0	R/W	EXTI<y>端口中断去抖滤波使能 0: 禁止 1: 使能 注: 该位需在中断使能之前配置, 使能之后禁止更改

## 第16章 循环冗余校验(CRC)

### 16.1 概述

循环冗余校验 (CRC) 发生器可以执行带可编程多项式设定的 CRC 计算, 用于对数据传输的完整性和正确性进行校验。

### 16.2 特性

- ◇ 支持四个常用的多项式: CRC-CCITT, CRC-8, CRC-16 和 CRC-32
  - ◇ CRC-CCITT:  $X^{16} + X^{12} + X^5 + 1$
  - ◇ CRC-8:  $X^8 + X^2 + X + 1$
  - ◇ CRC-16:  $X^{16} + X^{15} + X^2 + 1$
  - ◇ CRC-32:  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- ◇ 支持可编程的种子值
- ◇ 支持对输入数据和 CRC 校验值的可编程的反序设定
- ◇ 支持对输入数据和 CRC 校验值的可编程的反码设定
- ◇ 支持 8/16/32 位数据宽度
  - ◇ 8-bit 写模式: 1 个 AHB 时钟周期操作
  - ◇ 16-bit 写模式: 2 个 AHB 时钟周期操作
  - ◇ 32-bit 写模式: 4 个 AHB 时钟周期操作
- ◇ 支持使用 DMA 写数据执行 CRC 操作

### 16.3 结构框图

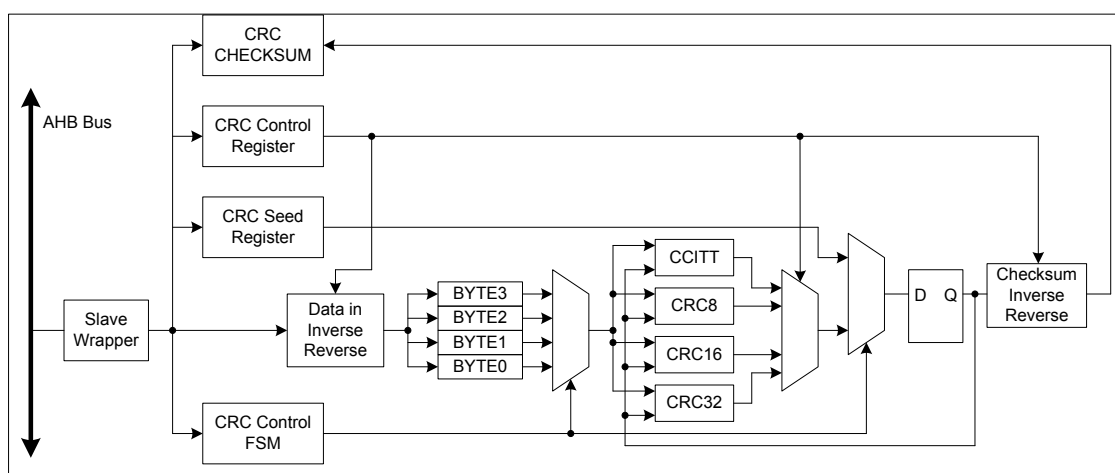


图 16-1 结构框图

## 16.4 功能描述

### 16.4.1 常规操作

CRC 发生器可以执行带可编程多项式设定的 CRC 运算。多项式操作包括 CRC-CCITT, CRC-8, CRC-16 和 CRC-32。用户可以通过设置 MODE 选择 CRC 多项式操作模式。

操作示例:

1. 通过设置 CRC\_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
  - a. 通过设置 CRC\_CR.CHSINV 配置 CRC 校验值反码
  - b. 通过设置 CRC\_CR.CHSREV 配置 CRC 校验值位反序
  - c. 通过设置 CRC\_CR.DATINV 配置 CRC 写入数据反码
  - d. 通过设置 CRC\_CR.DATREV 配置 CRC 写入数据位反序
  - e. 通过设置 CRC\_CR.MODE 配置 CRC 校验模式
  - f. 通过设置 CRC\_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC\_CR.RST 执行 CRC 复位, CRC 复位将装载初始种子值到 CRC 运算电路
4. 写数据到 CRC\_DATA 寄存器来计算 CRC 校验值
5. 通过读 CRC\_CHECKSUM 寄存器来获得 CRC 校验结果

### 16.4.2 DMA请求

DMA 请求 (在使能后) 仅用于数据传输, 当需要减少 MCU 负荷时可以使用 DMA 功能, 在计算完 CRC 后, DMA 模块将 CRC\_CHECKSUM 里面的计算值传输到用户提供的存储区中。

操作示例:

1. 通过设置 CRC\_CR.EN 使能 CRC 发生器
2. CRC 运算初始化设置
  - a. 通过设置 CRC\_CR.CHSINV 配置 CRC 校验值反码
  - b. 通过设置 CRC\_CR.CHSREV 配置 CRC 校验值位反序
  - c. 通过设置 CRC\_CR.DATINV 配置 CRC 写入数据反码
  - d. 通过设置 CRC\_CR.DATREV 配置 CRC 写入数据位反序
  - e. 通过设置 CRC\_CR.MODE 配置 CRC 校验模式
  - f. 通过设置 CRC\_CR.DATLEN 配置 CRC 写入数据长度
3. 通过设置 CRC\_CR.RST 执行 CRC 复位, CRC 复位将装载初始种子值到 CRC 运算



## 电路

4. 将 CRC\_DATA 寄存器的地址填入 DMA 模块的目标地址，表示每发生一次 CRC 的 DMA 申请事件，数据搬运到目标地址，此配置参考 DMA 章节
5. 将用户定义的数据存储区的地址填入 DMA 的原地址，表示每发生一次 CRC 的 DMA 申请事件后数据搬运到源地址，此配置参考 DMA 章节
6. 用户需要传输的总字节数需要在 DMA 中配置好，每发生一次 CRC 的 DMA 申请事件后，该值都会递减，注意最大字节数为 1024 个字节，此配置参考 DMA 章节
7. 设置 DMA 的触发源，选择触发 DMA 的输入源为 CRC
8. DMA 通道的优先级需要正确配置
9. 以上步骤都执行完毕后，初始化阶段完成了，需要激活该 DMA 通道
10. 使能 CRC\_CR.DMAEN 位。

## 16.5 特殊功能寄存器

### 16.5.1 寄存器列表

CRC 寄存器列表			
名称	偏移地址	类型	描述
CRC 基地址: 4008_5000 <sub>H</sub>			
CRC_CR	000 <sub>H</sub>	R/W	CRC 控制寄存器
CRC_DATA	004 <sub>H</sub>	R/W	CRC 写数据寄存器
CRC_SEED	008 <sub>H</sub>	R/W	CRC 种子寄存器
CRC_CHECKSUM	00C <sub>H</sub>	R	CRC 校验值寄存器

## 16.5.2 寄存器描述

### 16.5.2.1 CRC控制寄存器 (CRC\_CR)

CRC 控制寄存器 (CRC_CR)																															
偏移地址: 00H																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BYTORD	DATLEN	MODE	CHSINV	DATINV	CHSREV	DATREV	Reserved											DMAEN	CWERR	WERR	RST	EN		

Reserved	Bit 31-25	—	保留
BYTORD	Bit 24	RW	<b>校验值字节顺序选择位</b> 0: 优先校验低字节 1: 优先校验高字节 注: 优先校验低字节即优先校验Bit 7-0; 优先校验高字节, 16-bit模式即优先校验Bit 15-8, 32-bit模式优先校验Bit 31-24; 8-bit模式校验顺序无区别
DATLEN	Bit 23-22	RW	<b>数据长度选择位</b> 00: 通过写 CRC_DATA 寄存器方式自动判断 01: 数据为 8-bit (CRC_DATA[7:0]有效) 10: 数据为 16-bit (CRC_DATA[15:0]有效) 11: 数据为 32-bit (CRC_DATA[31:0]有效)
MODE	Bit 21-20	RW	<b>模式选择位</b> 00: CRC-CCITT 多项式模式 01: CRC-8 多项式模式 10: CRC-16 多项式模式 11: CRC-32 多项式模式
CHSINV	Bit 19	RW	<b>校验值反码使能位</b> 0: 禁止 1: 使能
DATINV	Bit 18	RW	<b>写数据反码使能位</b> 0: 禁止 1: 使能
CHSREV	Bit 17	RW	<b>校验值反序使能位</b> 0: 禁止 1: 使能
DATREV	Bit 16	RW	<b>写数据反序使能位</b> 0: 禁止 1: 使能
Reserved	Bit 15-5	—	保留
DMAEN	Bit 4	RW	<b>DMA 使能位</b> 0: 禁止

			1: 使能
CWERR	Bit 3	W1	<b>CRC 写数据错误标志清除位</b> 0: 无操作 1: 清除标志位
WERR	Bit 2	R	<b>CRC 写数据错误标志位</b> 0: 无错误 1: 发生写数据错误 注: 写入格式与 DATLEN 所选择不相符时会将标志位置位, 未在最低字节或低半字写数据时也会将标志位置位。
RST	Bit 1	W1	<b>CRC 复位位</b> 0: 无操作 1: 复位 注: 该位复位CRC内部状态机、DMAEN和缓存以及初始化种子值, 但不会复位寄存器值
EN	Bit 0	R/W	<b>CRC 使能位</b> 0: 禁止 1: 使能

### 16.5.2.2 CRC写数据寄存器 (CRC\_DATA)

CRC 写数据寄存器 (CRC_DATA)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

DATA	Bit 31-0	R/W	<b>CRC 写入数据位</b> 用户看可以通过 CPU 或 DMA 直接写数据到该位来执行 CRC 操作。 注 1: 当写数据长度是 8-bit 模式, 该位有效数据为 DATA[7:0], 如果数据长度是 16-bit 模式, 该位有效数据为 DATA[15:0]。如果自动检测数据长度, 则可通过最低字节、低半字或字写入方式任意写入数值。 注 2: 当写入到错误的字节或半字时, 硬件会置位写数据错误标志位 WERR。
------	----------	-----	---

### 16.5.2.3 CRC种子寄存器 (CRC\_SEED)

CRC 种子寄存器 (CRC_SEED)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															

SEED	Bit 31-0	R/W	<b>CRC 种子值</b> 该位表示 CRC 校验种子值
------	----------	-----	----------------------------------

### 16.5.2.4 CRC校验值寄存器 (CRC\_CHECKSUM)

CRC 校验值寄存器 (CRC_CHECKSUM)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM																															

CHECKSUM	Bit 31-0	R	<b>CRC 校验值</b> 该位表示 CRC 校验结果
----------	----------	---	---------------------------------

## 第17章 加密处理(CRYPT)

### 17.1 概述

硬件加密模块主要用于由硬件对数据进行加密或解密操作，支持的标准有 AES、DES、SM4。

AES (Advanced Encryption Standard) 是最新的分组对称密码算法，兼容联邦信息处理标准出版物 (FIPS PUB 197, 2001 年 11 月 26 日) 规定的高级加密标准 (AES)。

DES (Data Encryption Standard) 是应用非常广泛的对称密码算法，兼容联邦信息处理标准出版物 (FIPS PUB 46-3, 1999 年 10 月 25 日) 规定的加密标准 (DES) 和三重 DES (TDES)，遵循美国国家标准协会 (ANSI) X9.52 标准。

SM4 是国密局批准的对称加密密码算法。

### 17.2 特性

- ◇ 适用于 AES、DES 和 TDES 加密和解密操作
- ◇ AES
  - ◇ 支持 128、192 和 256 位的密钥
  - ◇ 支持 ECB、CBC、CTR 和 GCM 模式
  - ◇ 支持 1、8、16 或 32 位数据交换
- ◇ DES/TDES
  - ◇ 支持 ECB、CBC 模式
  - ◇ 支持 64、128 和 192 位密钥
  - ◇ 支持在 CBC 模式下使用的 2×32 位初始化向量 (IV)
  - ◇ 支持 1、8、16 或 32 位数据交换
- ◇ SM4
  - ◇ 支持 128 的密钥
  - ◇ 支持 ECB、CBC 模式
  - ◇ 支持 1、8、16 或 32 位数据交换
- ◇ 支持直接存储器访问 (DMA) (用于传入数据和读出已处理数据)
- ◇ 支持产生 CPU 中断请求

## 17.3 AES功能描述

### 17.3.1 AES-ECB模式加密

AES 是以 128bit 作为一个数据单元进行加密，下图介绍了 AES 电子密码本 (AES-ECB) 模式加密。一个数据单元 (128bit 明文 P) 经过位/字节/半字交换后作为一个输入单元 (I)。输入单元通过 AES 算法 (AEA) 在加密状态下使用 128 位、192 位或 256 位密钥进行处理。得到的处理结果再执行位/字节/半字交换后，得到 128bit 输出密文单元 (C)。

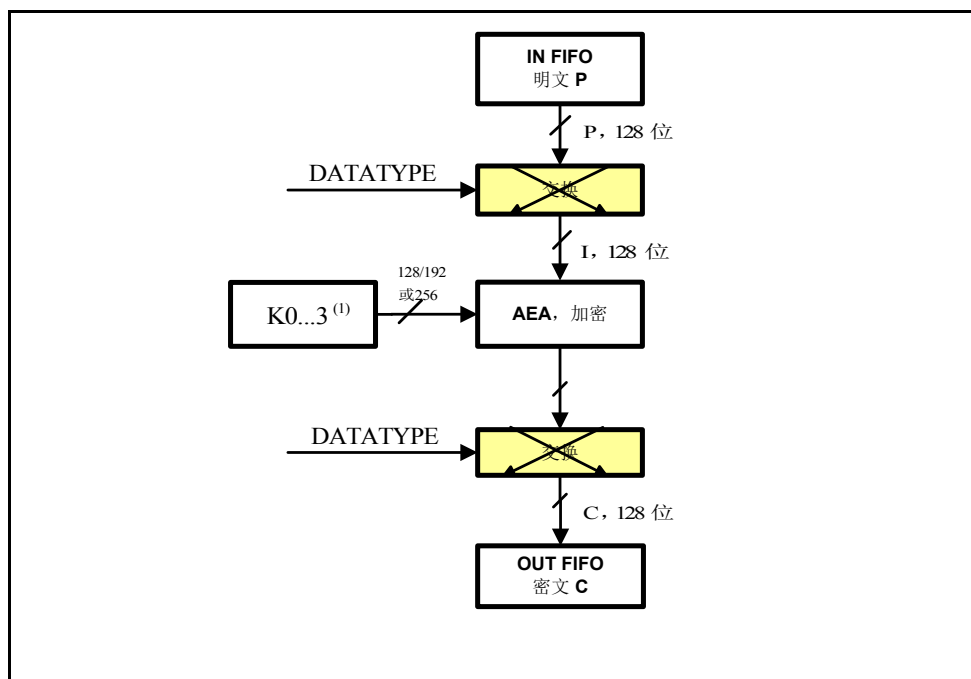


图 17-1 AES-ECB 模式加密

注 1: K = 密钥; C = 密文; I = 输入块; O = 输出块; P = 明文。

注 2: 如果密钥大小 = 128: 密钥 = [K3 K2];

如果密钥大小 = 192: 密钥 = [K3 K2 K1];

如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例:

1. 设置 CRYPT\_CON.CRYSEL = 0, 选择算法类型为 AES
2. 设置 CRYPT\_CON.ENCS = 1, 选择加密
3. 设置 CRYPT\_CON.AESKS, 选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT\_CON.MODE = 0, 选择 ECB 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥, 128bit 长度填入 KEY3/2/1/0, 192bit 长度填入 KEY5/4/3/2/1/0, 256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT\_DATA0/1/2/3 中写入需要加密的明文
7. 设置 CRYPT\_CON.GO = 1, 启动加密

8. 通过 CRYPT\_IF.DONE 判断加密是否完成，如果 CRYPT\_IF.DONE = 0，则加密完成
9. 从 CRYPT\_DATA0/1/2/3 中读出已经加密的密文

### 17.3.2 AES-ECB模式解密

AES 是以 128bit 作为一个数据单元进行解密，下图介绍了 AES 电子密码本 (AES-ECB) 模式解密。一个数据单元 (128bit 密文 C) 经过位/字节/半字交换后作为一个输入单元 (I)。输入单元通过 AES 算法 (AEA) 在解密状态下使用 128 位、192 位或 256 位密钥进行处理。得到的处理结果再执行位/字节/半字交换后，得到 128bit 输出明文单元 (P)。

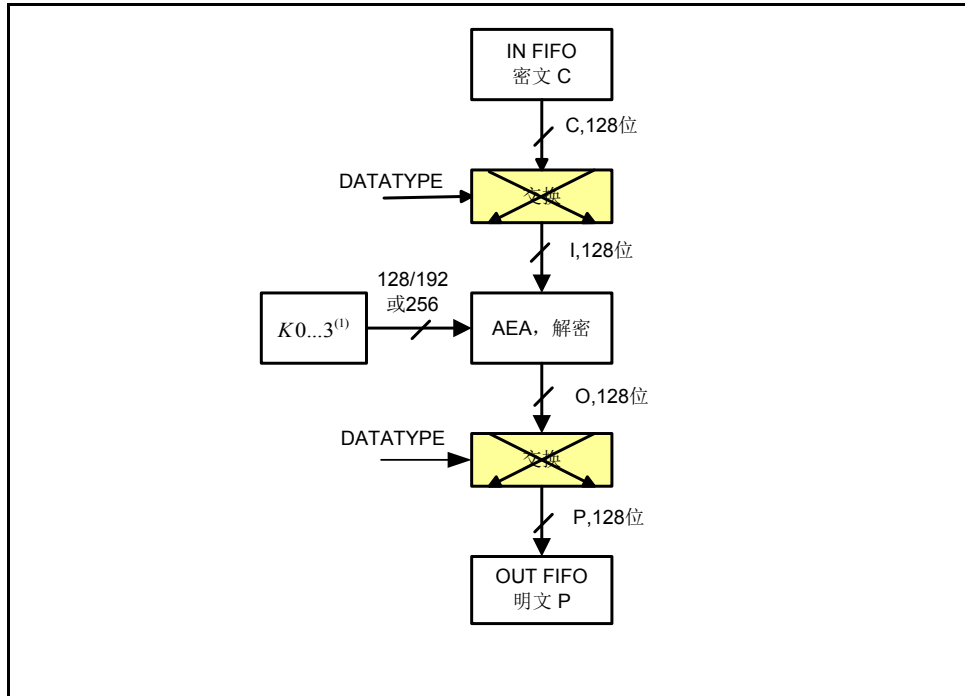


图 17-2 AES-ECB 模式解密

注 1: K = 密钥; C = 密文; I = 输入块; O = 输出块; P = 明文。

注 2: 如果密钥大小 = 128: 密钥 = [K3 K2];

如果密钥大小 = 192: 密钥 = [K3 K2 K1];

如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例:

1. 设置 CRYPT\_CON.CRYSSEL = 0，选择算法类型为 AES
2. 设置 CRYPT\_CON.ENCS = 0，选择解密
3. 设置 CRYPT\_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT\_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥，128bit 长度填入 KEY3/2/1/0, 192bit 长度填入 KEY5/4/3/2/1/0, 256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT\_DATA0/1/2/3 中写入需要解密的密文



7. 设置 CRYPT\_CON.GO = 1，启动解密
8. 通过 CRYPT\_IF.DONE 判断解密是否完成，如果 CRYPT\_IF.DONE = 0，则解密完成
9. 从 CRYPT\_DATA0/1/2/3 中读出已经解密的明文

### 17.3.3 AES-CBC模式加密

AES 是以 128 位作为一个数据单元进行加密，下图介绍了 AES 加密分组链接(AES-CBC) 模式加密。该模式首先将明文消息分成多个 128 位数据明文单元。在 CBC 加密过程中，一个数据单元经过执行位/字节/半字交换后作为明文单元 (P1)，通过与一个 128 位初始化向量 (IV) 进行异或运算 ( $IV \oplus P1$ )，作为第一个输入单元 (I1)。该输入单元通过 AES 算法 (AEA) 在加密状态下使用 128 位、192 位或 256 位密钥进行加密处理。生成的 128 位输出单元 (O1) 将直接用作密文 (C1)，即  $C1 = O1$ 。然后，第一个密文单元与第二个明文数据单元进行异或运算 ( $C1 \oplus P2$ )，从而生成第二个输入单元 (I2)。第二个输入单元通过以上 AES 处理而生成第二个密文单元。加密处理会不断将后续密文单元和明文单元链接到一起，直到消息中所有的明文单元都加密完成为止。如果消息中最后的数据单元不是一个 128 位数据单元，则由应用程序按照一定规则对不完整数据单元进行加密。

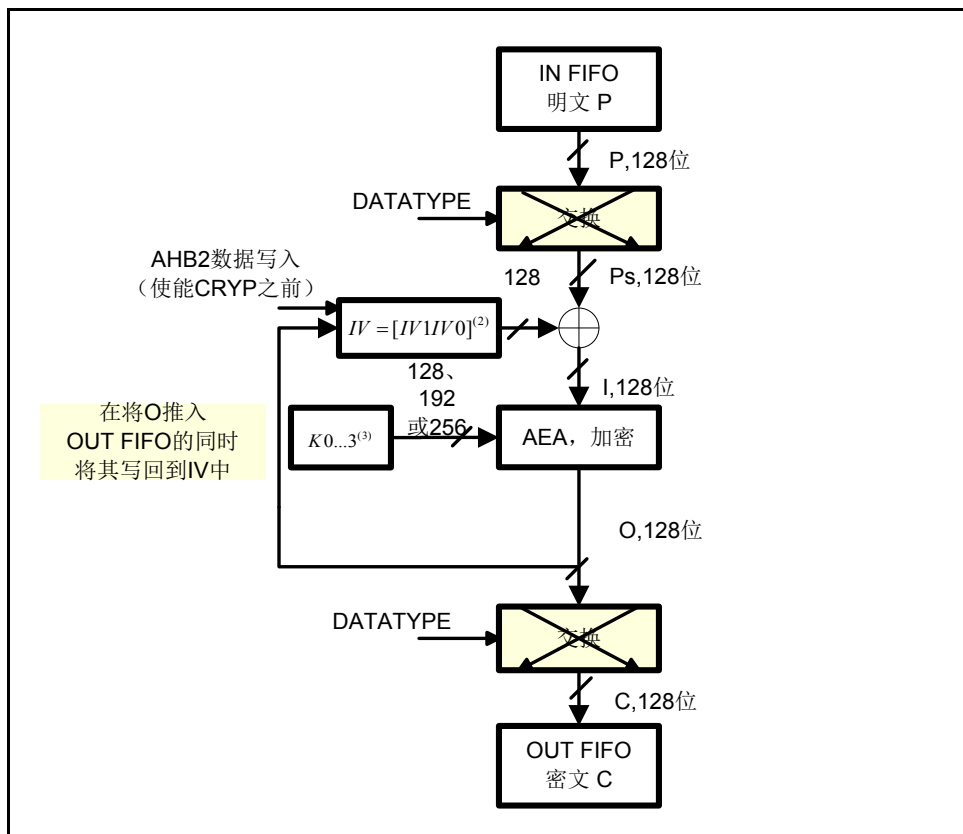


图 17-3 AES-CBC 模式加密

注 1: K = 密钥; C = 密文; I = 输入块; Ps = 交换前 (解码时) 或 交换后 (编码时) 的明文; P = 明文; IV = 初始化向量。  
 注 2:  $IVx = [IVxR \ IVxL]$ , R = 右, L = 左。  
 注 3: 如果密钥大小 = 128: 密钥 = [K3 K2];  
 如果密钥大小 = 192: 密钥 = [K3 K2 K1];

如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例:

1. 设置 CRYPT\_CON.CRYSEL = 0, 选择算法类型为 AES
2. 设置 CRYPT\_CON.ENCS = 1, 选择加密
3. 设置 CRYPT\_CON.AESKS, 选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT\_CON.MODE = 1, 选择 CBC 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥, 128bit 长度填入 KEY3/2/1/0, 192bit 长度填入 KEY5/4/3/2/1/0, 256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT\_IV0/1/2/3 填入初始向量
7. 向 CRYPT\_DATA0/1/2/3 中写入需要加密的明文
8. 设置 CRYPT\_CON.GO = 1, 启动加密
9. 通过 CRYPT\_IF.DONE 判断加密是否完成, 如果 CRYPT\_IF.DONE = 0, 则加密完成
10. 从 CRYPT\_DATA0/1/2/3 中读出已经加密的密文
11. 回到步骤 7 继续进行加密
12. 所有加密完成

#### 17.3.4 AES-CBC模式解密

AES 是以 128 位作为一个数据单元进行解密, 下图介绍了 AES 密码分组链接(AES-CBC) 模式解密。该模式首先将密文消息分成多个 128 位数据密文单元, 第一个密文单元 (C1) 经过位/字节/半字交换后作为一个输入单元 (I1), 该输入单元 (I1) 通过 AES 算法 (AEA) 在解密状态下使用 128 位、192 位或 256 位密钥进行解密处理, 生成的 128 位输出单元 (O1), 再与 IV 进行异或运算 ( $O1 \oplus IV$ ), 从而生成第一个明文单元。然后, 第二个密文单元作为下一个输入单元, 经过以上 AES 解密处理, 生成的输出单元再与第一个密文单元进行异或运算 ( $O2 \oplus C1$ ), 从而生成第二个明文数据单元 (P2)。依次进行解密, 直到消息中所有的密文单元都解密完成为止。

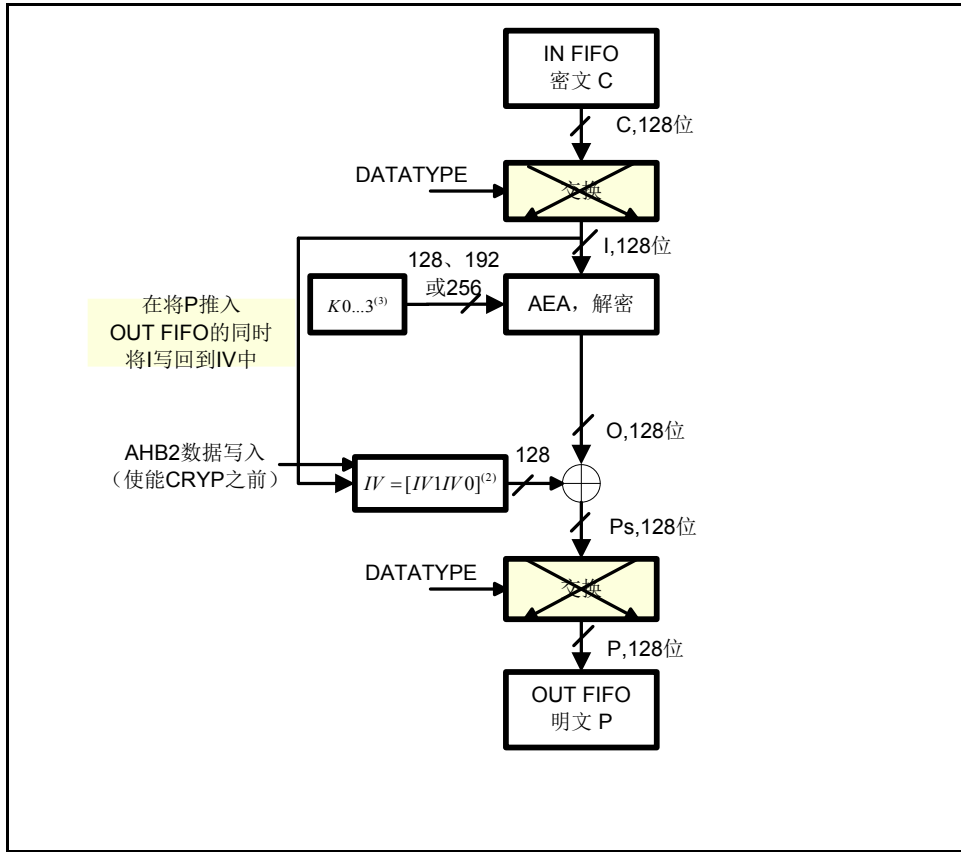


图 17-4 AES-CBC 模式解密

注 1: K = 密钥; C = 密文; I = 输入块; Ps = 交换前 (解码时) 或 交换后 (编码时) 的明文; P = 明文; IV = 初始化向量。

注 2:  $IVx = [IVxR \ IVxL]$ , R = 右, L = 左。

注 3: 如果密钥大小 = 128: 密钥 = [K3 K2];  
如果密钥大小 = 192: 密钥 = [K3 K2 K1];  
如果密钥大小 = 256: 密钥 = [K3 K2 K1 K0]。

操作示例:

1. 设置 CRYPT\_CON.CRYSSEL = 0, 选择算法类型为 AES
2. 设置 CRYPT\_CON.ENCS = 0, 选择解密
3. 设置 CRYPT\_CON.AESKS, 选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPTCON.MODE = 1, 选择 CBC 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥, 128bit 长度填入 KEY3/2/1/0, 192bit 长度填入 KEY5/4/3/2/1/0, 256bit 长度填入 KEY7/6/5/4/3/2/1/0
6. 向 CRYPT\_IV0/1/2/3 填入初始向量
7. 向 CRYPT\_DATA0/1/2/3 中写入需要解密的密文
8. 设置 CRYPT\_CON.GO = 1, 启动解密

9. 通过 CRYPT\_IF.DONE 判断解密是否完成，如果 CRYPT\_IF.DONE = 0，则解密完成
10. 从 CRYPT\_DATA0/1/2/3 中读出已经解密的明文
11. 回到步骤 7 继续进行解密
12. 所有解密完成

### 17.3.5 AES-CTR模式加解密

AES 计数模式 (AES-CTR) 即把一个 32bit 的计数器和一个随机数作为 AES 加解密模块的输入，输出的密文流作为密钥流，对明文或密文进行异或从而得到相应的密文或明文。在此模式下，解密操作与加密操作的操作方式完全相同。图 17-5 和图 17-6 分别给出了加密流程和解密流程。

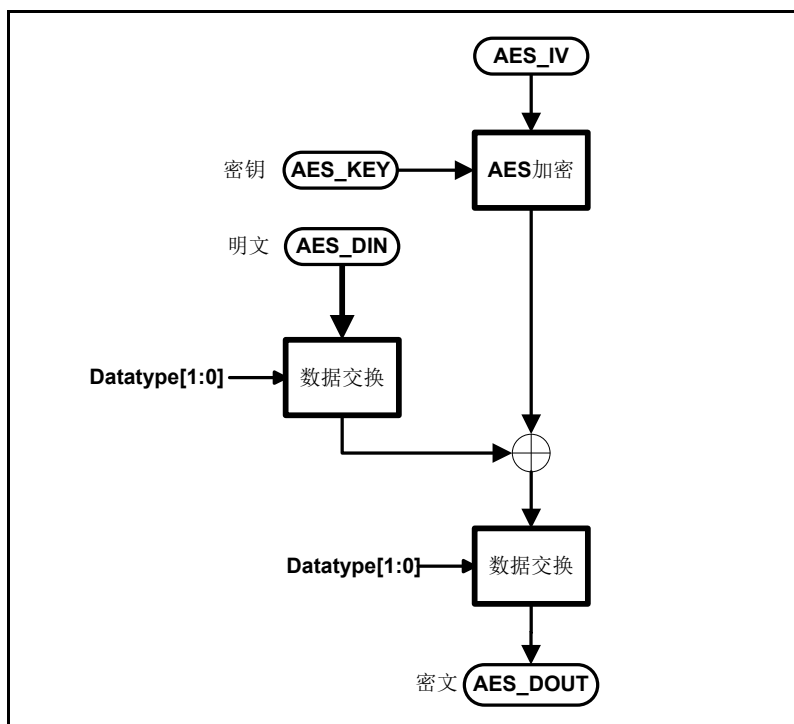


图 17-5 CTR 加密流程

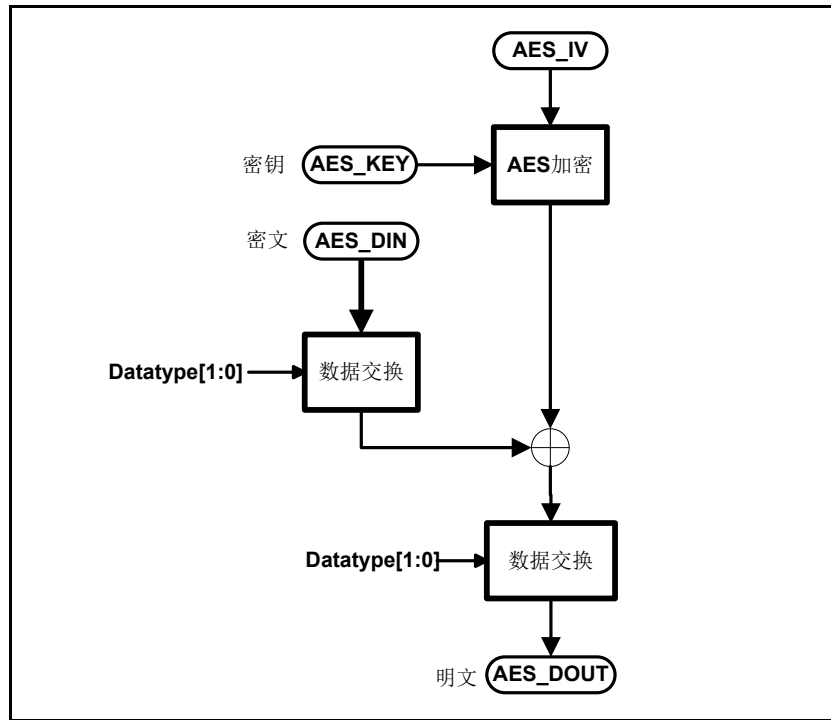


图 17-6 CTR 解密流程

随机数(nonce)和 32 位计数器存储在 IV 寄存器中，如图 17-7 所示：

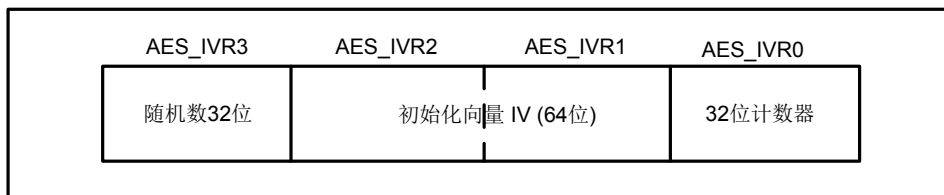


图 17-7 计数器模式下的初始计数器块结构

- ◇ 随机数是一个 32 位的一次性值。每个不同的通信都应得到最新的随机值。
- ◇ 初始化向量 (IV) 是一个 64 位的值，为确保一个给定值只用于一个给定的密钥，必须选择 IV。
- ◇ 32 位的计数器是一个大端模式的整数，随着块的加密次数增加而增加。计数器的初始值应设置为“1”。
- ◇ 为保持计数器块的最高有效 96 位不变，其使用最低有效 32 位进行增量的计数。

操作示例：

1. 设置 CRYPT\_CON.CRYSEL = 0，选择算法类型为 AES
2. 设置 CRYPT\_CON.ENCS = 1/0，选择加密/解密
3. 设置 CRYPT\_CON.AESKS，选择密钥长度 128bit、192bit 或者 256bit
4. 设置 CRYPT\_CON.MODE = 2，选择 CTR 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥，128bit 长度填入 KEY3/2/1/0,192bit 长度填入 KEY5/4/3/2/1/0，256bit 长度填入 KEY7/6/5/4/3/2/1/0

6. 向 CRYPT\_IV0/1/2/3 填入初始向量为初始计数器块
7. 向 CRYPT\_DATA0/1/2/3 中写入需要加密的明文/解密的密文
8. 设置 CRYPT\_CON.GO = 1, 启动加密/解密
9. 通过 CRYPT\_IF.DONE 判断加密/解密是否完成, 如果 CRYPT\_IF.DONE = 0, 则加密/解密完成
10. 从 CRYPT\_DATA0/1/2/3 中读出已经加密/的密文/解密的明文
11. 回到步骤 7 继续进行加密/解密
12. 所有加密/解密完成

### 17.3.6 AES-GCM模式加解密

AES Galois/计数器模式 (GCM) 不仅可以加密和验证明文, 还能够生成对应的密文(消息验证码)和标记(消息完整性检查)。该算法基于 AES 计数器模式, 可确保保密性。它针对固定有限的字段使用乘法器来生成标记。开始执行算法时需要使用初始化向量。

要处理的消息分为 2 个部分:

- ◇ 信息头 (附加认证数据): 需要验证但不被保护的数据 (类似路由数据包的信息)
- ◇ 信息 (有效负载): 经过验证和加密的消息本身

注: 附加认证数据必须在信息的前面, 并且两部分不能混合。

依据 GCM 标准, 在消息结束时, 必须传递完由信息头大小 (64 位) 和信息大小 (64 位) 构成的特定 128 位块。在计算过程中, 必须将信息头块与信息块区分开。

GCM 的加密按照以下公式定义:

$$H = E(K, 0^{128})$$
$$Y_0 = \begin{cases} IV \parallel 0^{31} & \text{if } \text{len}(IV) = 96 \\ GHASH(H, \{\}, IV) & \text{otherwise} \end{cases}$$
$$Y_i = \text{incr}(Y_{i-1}) \text{ for } i = 1, \dots, n$$
$$C_i = P_i \oplus E(K, Y_i) \text{ for } i = 1, \dots, n-1$$
$$C_n^* = P_n^* \oplus MSB_u(E(K, Y_n))$$
$$T = MSB_t(GHASH(H, A, C) \oplus E(K, Y_0))$$

其中 GHASH 函数的定义为  $GHASH(H, A, C) = X_{m+n+1}$ , 其中  $X$  的定义为

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \bullet H & \text{for } i = 1, \dots, m-1 \\ (X_{m-1} \oplus (A_m^* \parallel 0^{128-v})) \bullet H & \text{for } i = m \\ (X_{i-1} \oplus C_i) \bullet H & \text{for } i = m+1, \dots, m+n-1 \\ (X_{m+n-1} \oplus (C_m^* \parallel 0^{128-u})) \bullet H & \text{for } i = m+n \\ (X_{m+n} \oplus (\text{len}(A) \parallel \text{len}(C))) \bullet H & \text{for } i = m+n+1. \end{cases}$$

GCM 模式的加解密流程如图 17-8、图 17-9 所示。

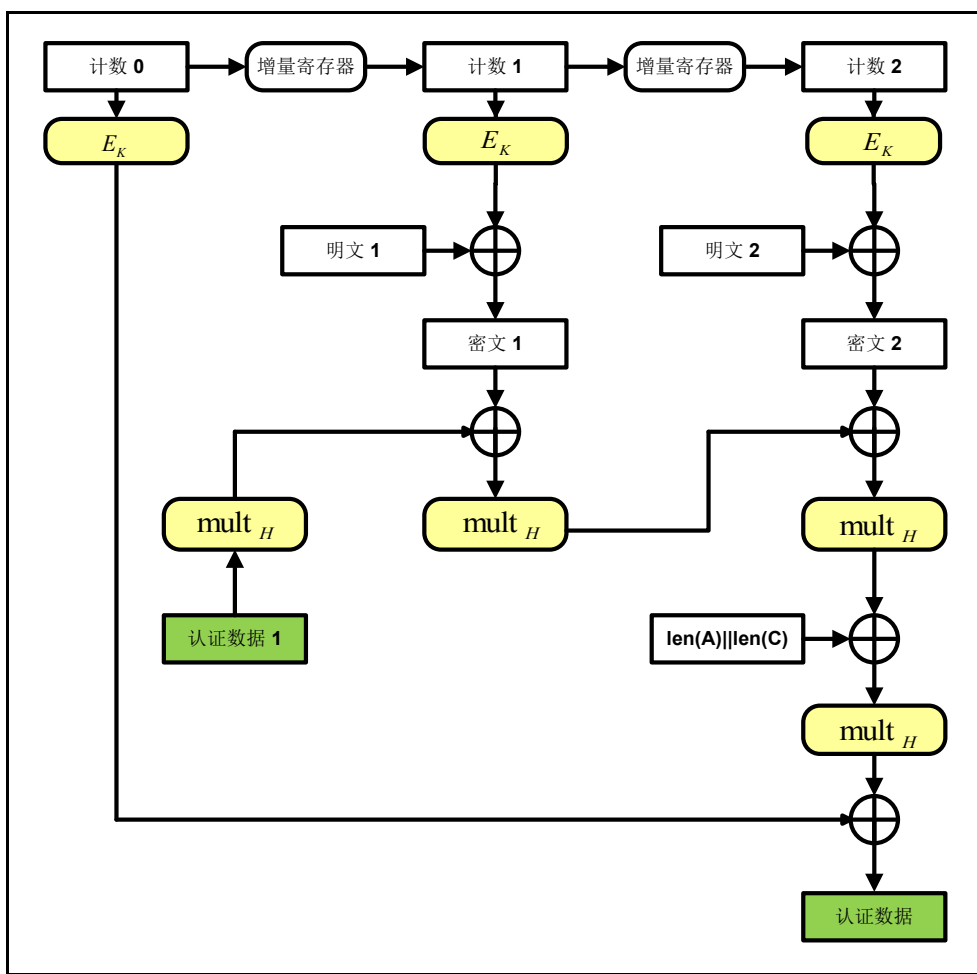


图 17-8 GCM 加密流程

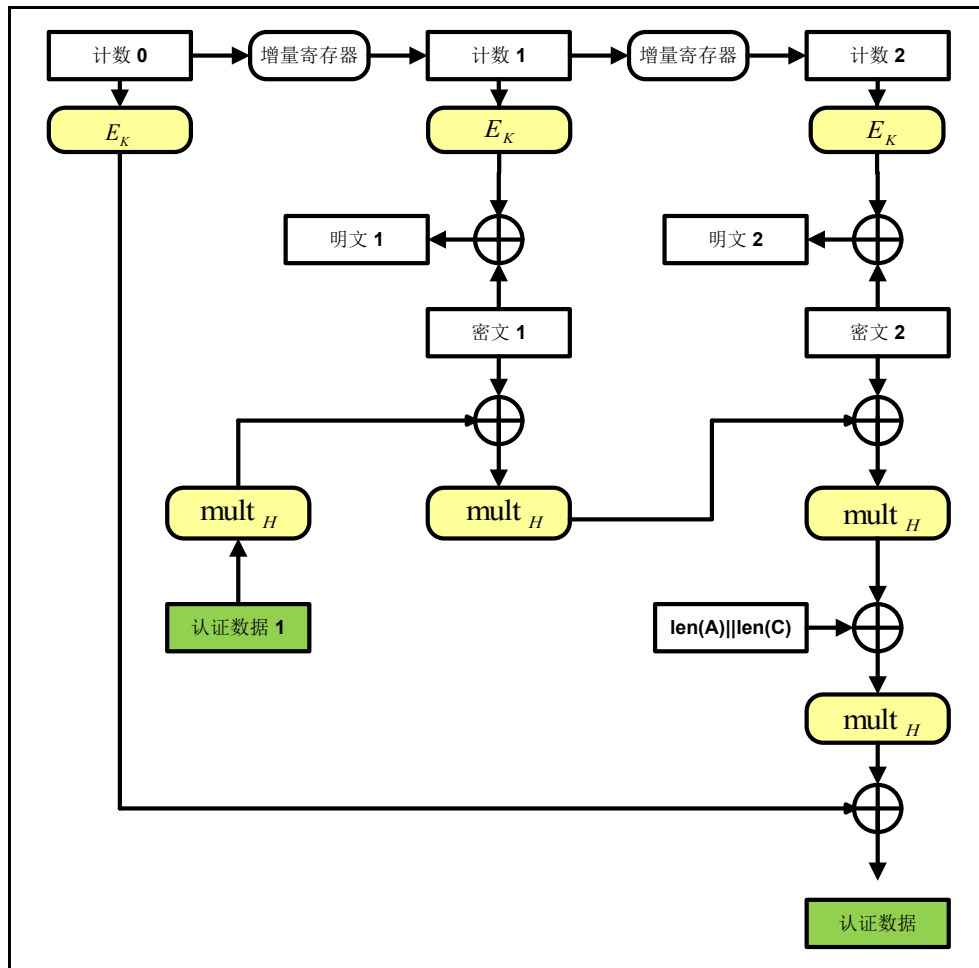


图 17-9 GCM 解密流程

图中 EK 表示 AES 加密模块。MULTH 模块是一个  $GF(2^{128})$  域上的乘法。增量寄存器表示计数器加 1。GCM 模式由软件配合实现，硬件提供一个 AES 模块和 MULTH 模块供软件调度。GCM 模式加解密的过程与 CTR 模式相同。认证过程需执行 GHASH 算法，可通过软件调度硬件模块（MUL 模式）实现。

### MUL 模式使用说明

算法描述

Algorithm 1:  $X \cdot Y$

Input:

blocks X, Y

Output:

block  $X \cdot Y$

Steps:

- 1). Let  $x_0x_1\dots x_{127}$  denote the sequence of bits in X.
- 2). Let  $Z_0 = 0^{128}$  and  $V_0 = Y$ .
- 3). For  $i = 0$  to 127, calculate blocks  $Z_{i+1}$  and  $V_{i+1}$  as follows:



$$Z_{i+1} = \begin{cases} Z_i & \text{if } x_i = 0; \\ Z_i \oplus V_i & \text{if } x_i = 1; \end{cases}$$

$$V_{i+1} = \begin{cases} V_i \gg 1 & \text{if } LSB_1(V_i) = 0; \\ V_i \gg 1 \oplus R_i & \text{if } LSB_1(V_i) = 1; \end{cases}$$

#### 4). Return Z128。

##### 实现方式

在 MUL 模式下，执行加密/解密需要以下 3 个步骤：

第 1 步：CRYPT\_CON.MODE=3 选择 MUL 模式选择)，CRYPT\_CON.RESCLR=1，清空 CRYPT\_RESx 内容，CRYPT\_CON.GO = 1 使能 MUL 运算；

第 2 步：将 X 放入 CRYPT\_DATAx 寄存器，Y 放入 CRYPT\_IVx 寄存器，Z 存放在 CRYPT\_RESx 寄存器；

第 3 步：读取输出结果：在 CRYPT\_IF.MULTHIF 为高时，读取 CRYPT\_RESx 的值，读取后，置 CRYPT\_IFC.MULTHIFC = 1。

在 GCM 模式下，执行加密/解密需要以下 3 个步骤

##### 第 1 步：GCM 初始化

操作示例：

1. 将 CRYPT\_CON.GO 清零来禁止加密处理器。
2. 初始化密钥寄存器 CRYPT\_KEYx(128/192/256 位)及初始向量寄存器 CRYPT\_IVx。

##### 第 2 步：GCM 加密/解密

操作示例：

1. 将明文/密文写入数据寄存器 CRYPT\_DATAx，并使用 CRYPT\_CON 中的 FIFOEN 位判断 FIFO 是否使能，且通过 FIFOODR 位确定 FIFO 的顺序。
2. 配置 CRYPT\_CON.ENCS 位选择加密或解密。设置 CRYPT\_CON.MODE = 2，设置 CRYPT\_CON.GO = 1 启动运算。
3. 等待运算结束后，从 CRYPT\_RESx 中读取结果。
4. 重复之前的步骤，直到所有的明文/密文加解密完成，此过程可以使用 DMA。

##### 第 3 步：GCM 签名验证

该阶段需要用到 HASH 函数。HASH 函数的输入数据包括需要验证但不受保护的数据 A（128 位的整数倍，不足以 0 补充）、密文 C（128 位的整数倍，不足以 0 补充）、数据 A 的长度（64 位）和密文 C 的长度（64 位）。

操作示例：

1. 配置 CRYPT\_CON.MODE = 0，计算 H 值。
2. 配置 CRYPT\_CON.RESCLR = 1，将寄存器 CRYPT\_RESx 清 0。
3. 将 HASH 函数的输入数据以 4 字一组与 CRYPT\_RESx 异或后写入寄存器 CRYPT\_DATAx，将 H 值写入寄存器 CRYPT\_IVx。
4. 配置 CRYPT\_CON.MODE = 3，设置 CRYPT\_CON.GO = 1 启动运算。
5. 重复 2)-5)步，直到所有数据运算完成。读取 CRYPT\_RESx 为 HASH 值。
6. 对初始向量进行一次 ECB 运算，将得到的结果与 HASH 值异或，以此得到签名。

## 17.4 DES功能描述

### 17.4.1 DES/TDES-ECB模式加密

DES 或 TDES 是以 64bit 作为一个数据单元进行加密，下图介绍 DES 和 TDES 电子密码本 (DES/TDES-ECB) 模式中的加密。一个数据单元 (64bit 明文 P) 经过位/字节/半字交换后作为一个输入单元 (I)。该输入单元通过 DES 算法 (DEA) 在加密状态下使用 K1 进行加密处理。在 DES 算法处理后的输出会作为下一次 DES 算法 (DEA) 的输入，在解密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法 (DEA) 的输入，在加密状态下使用 K3 再次执行 DES 算法。最后生成的 64bit 输出块 (O)，再执行位/字节/半字交换之后，以密文 (C) 形式输出到数据寄存器。

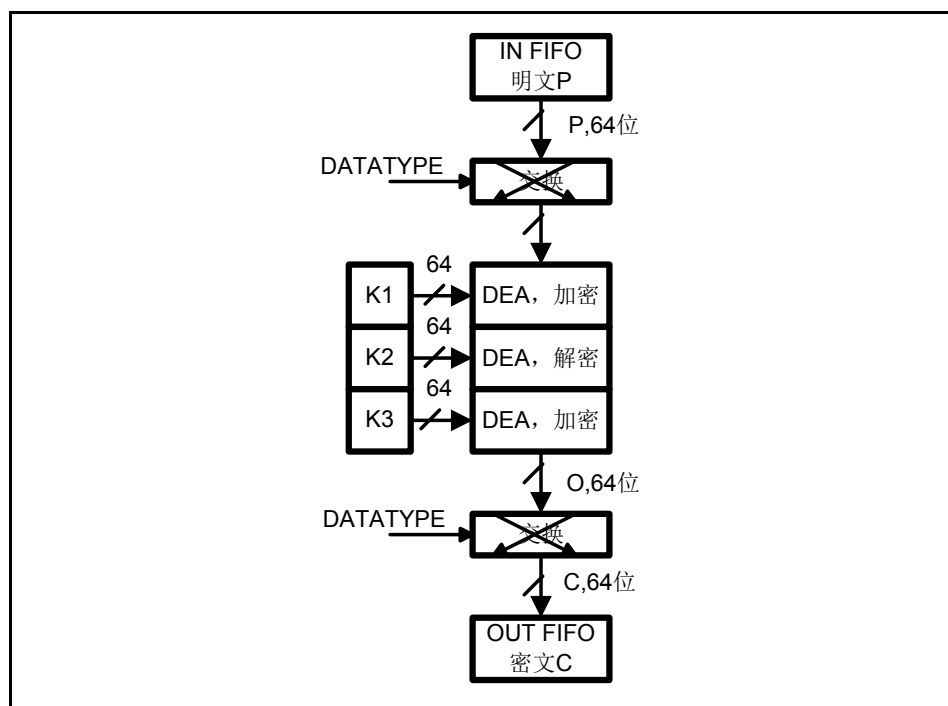


图 17-10 DES/TDES-ECB 模式加密

注：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文。

操作示例：

1. 设置 `CRYPT_CON.CRYSEL = 1`，选择算法类型为 DES/TDES
2. 设置 `CRYPT_CON.ENCS = 1`，选择加密
3. 设置 `CRYPT_CON.TDES = 0` 选择 DES，设置 `CRYPT_CON.TDES = 1` 选择 TDES，用户还可以设置 `CRYPT_CON.DESKS` 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 `CRYPT_CON.MODE = 0`，选择 ECB 模式
5. 向 `CRYPT_KEY0...CRYPT_KEY7` 中填入密钥，其中 `KEY0/1 = K1`, `KEY2/3 = K2`, `KEY4/5 = K3`。
6. 向 `CRYPT_DATA0/1` 中写入需要加密的明文

7. 设置 CRYPT\_CON.GO = 1，启动加密
8. 通过 CRYPT\_IF.DONE 判断加密是否完成，如果 CRYPT\_IF.DONE = 0，则加密完成
9. 从 CRYPT\_DATA0/1 中读出已经加密的密文

### 17.4.2 DES/TDES-ECB模式解密

DES 或 TDES 是以 64bit 作为一个数据单元进行解密，下图介绍 DES 和 TDES 电子密码本 (DES/TDES-ECB) 模式中的解密。一个数据单元 (64bit 密文 C) 经过位/字节/半字交换后作为一个输入单元 (I)，作为解密的密钥顺序是与加密过程中使用的密钥顺序是相反。该输入单元通过 DES 算法 (DEA) 在解密状态下使用 K3 进行解密处理。在 DES 算法处理后的输出会作为下一次 DES 算法 (DEA) 的输入，在加密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法 (DEA) 的输入，在解密状态下使用 K1 再次执行 DES 算法。最后生成的 64bit 输出块 (O)，再执行位/字节/半字交换之后，以明文 (P) 形式输出到数据寄存器。

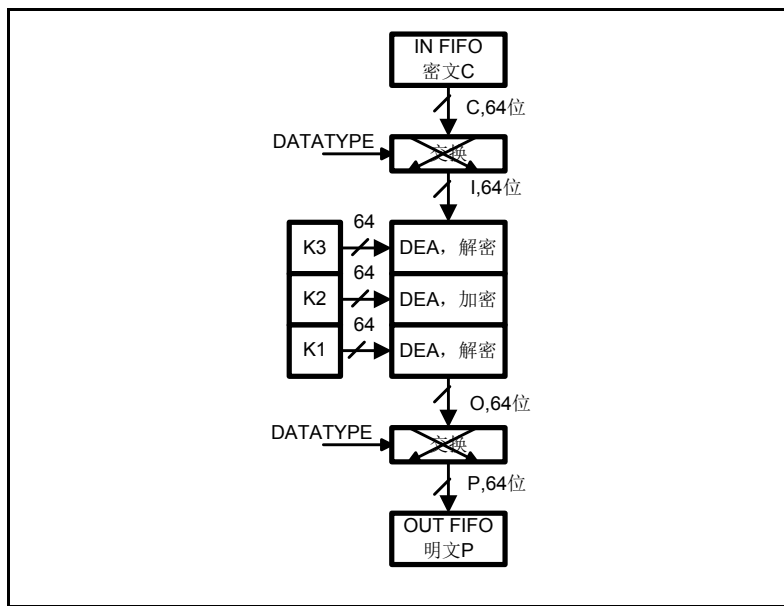


图 17-11 DES/TDES-ECB 模式解密

注：K = 密钥；C = 密文；I = 输入块；O = 输出块；P = 明文

操作示例：

1. 设置 CRYPT\_CON.CRYSEL = 1，选择算法类型为 DES/TDES
2. 设置 CRYPT\_CON.ENCS = 0，选择解密
3. 设置 CRYPT\_CON.TDES = 0 选择 DES, 设置 CRYPT\_CON.TDES = 1 选择 TDES, 用户还可以设置 CRYPT\_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT\_CON.MODE = 0，选择 ECB 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥，其中 KEY0/1 = K1, KEY2/3 = K2, KEY4/5 = K3
6. 向 CRYPT\_DATA0/1 中写入需要解密的密文

7. 设置 CRYPT\_CON.GO = 1, 启动解密
8. 通过 CRYPT\_IF.DONE 判断解密是否完成, 如果 CRYPT\_IF.DONE = 0, 则解密完成
9. 从 CRYPT\_DATA0/1 中读出已经解密的明文

### 17.4.3 DES/TDES-CBC模式加密

DES 或 TDES 是以 64 位作为一个数据单元进行加密, 下图介绍了 DES 和三重 TDES 密码分组链接 (DES/TDES-CBC) 模式加密。该模式首先将明文消息分成多个 64bit 数据明文单元。在 CBC 加密过程中, 一个数据单元经过执行位/字节/半字交换后作为明文单元 (P1), 通过与一个 64 位初始化向量 (IV) 进行异或运算 ( $IV \oplus P1$ ), 作为第一个输入单元 (I1)。该输入单元通过 DES 算法 (DEA) 在加密状态下使用 K1 进行加密处理。在 DES 算法处理后的输出会作为下一次 DES 算法 (DEA) 的输入, 在解密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法 (DEA) 的输入, 在加密状态下使用 K3 再次执行 DES 算法。生成的 64 位输出单元 (O1) 将直接用作密文 (C1), 即  $C1 = O1$ 。然后, 第一个密文单元与第二个明文数据单元进行异或运算 ( $C1 \oplus P2$ ), 从而生成第二个输入单元 (I2)。第二个输入单元通过以上 CBC 处理而生成第二个密文单元。加密处理会不断将后续密文单元和明文单元链接到一起, 直到消息中所有的明文单元都加密完成为止。如果消息中最后的数据单元不是一个 64 位数据单元, 则由应用程序按照一定规则对不完整数据单元进行加密。

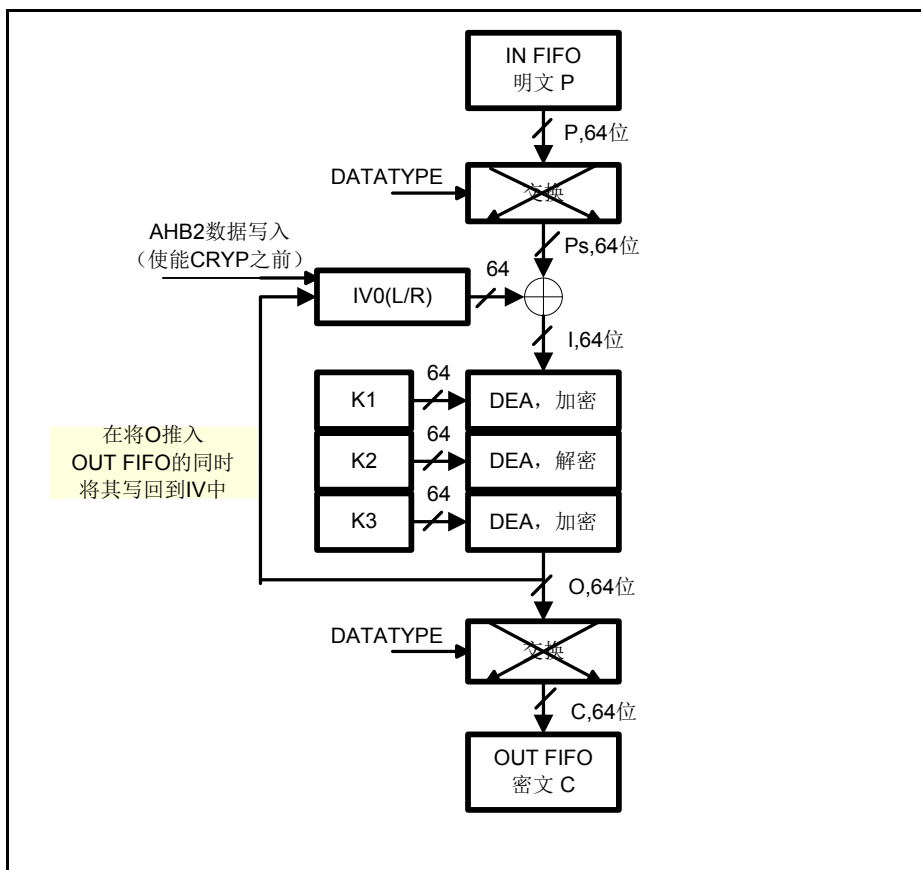


图 17-12 DES/TDES-CBC 模式加密

注: K = 密钥; C = 密文; I = 输入块; Ps = 交换前 (解码时) 或交换后 (编码时) 的明文; P = 明文; IV = 初始化向量。

操作示例:

1. 设置 CRYPT\_CON.CRYSEL = 1, 选择算法类型为 DES/TDES
2. 设置 CRYPT\_CON.ENCS = 1, 选择加密
3. 设置 CRYPT\_CON.TDES = 0 选择 DES, 设置 CRYPT\_CON.TDES = 1 选择 TDES, 用户还可以设置 CRYPT\_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT\_CON.MODE = 1, 选择 CBC 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥, 其中 KEY0/1 = K1, KEY2/3 = K2, KEY4/5 = K3.
6. 向 CRYPT\_IV0/1 填入初始向量
7. 向 CRYPT\_DATA0/1 中写入需要加密的明文
8. 设置 CRYPT\_CON.GO = 1, 启动加密
9. 通过 CRYPT\_IF.DONE 判断加密是否完成, 如果 CRYPT\_IF.DONE = 0, 则加密完成
10. 从 CRYPT\_DATA0/1 中读出已经加密的密文
11. 回到步骤 7 继续进行加密
12. 所有加密完成

#### 17.4.4 DES/TDES-CBC模式解密

DES 或 TDES 是以 64 位作为一个数据单元进行解密, 下图介绍了 DES 和三重 TDES 密码分组链接 (DES/TDES-CBC) 模式解密。该模式首先将密文消息分成多个 64 位数据密文单元, 第一个密文单元 (C1) 经过位/字节/半字交换后作为一个输入单元 (I1), 而作为解密的密钥顺序是与加密过程中使用的密钥顺序是相反。该输入单元 (I1) 通过 DES 算法 (DEA) 在解密状态下使用 K3 进行解密处理。在 DES 算法处理后的输出会作为下一次 DES 算法 (DEA) 的输入, 在加密状态下使用 K2 执行 DES 算法。在 DES 算法处理后的输出再次作为 DES 算法 (DEA) 的输入, 在解密状态下使用 K1 再次执行 DES 算法。最后生成的 64 位输出单元 (O1), 再与 IV 进行异或运算 ( $O1 \oplus IV$ ), 从而生成第一个明文单元。然后, 第二个密文单元作为下一个输入单元, 经过以上 DES 解密处理, 生成的输出单元再与第一个密文单元进行异或运算 ( $O2 \oplus C1$ ), 从而生成第二个明文数据单元 (P2)。依次进行解密, 直到消息中所有的密文单元都解密完成为止。

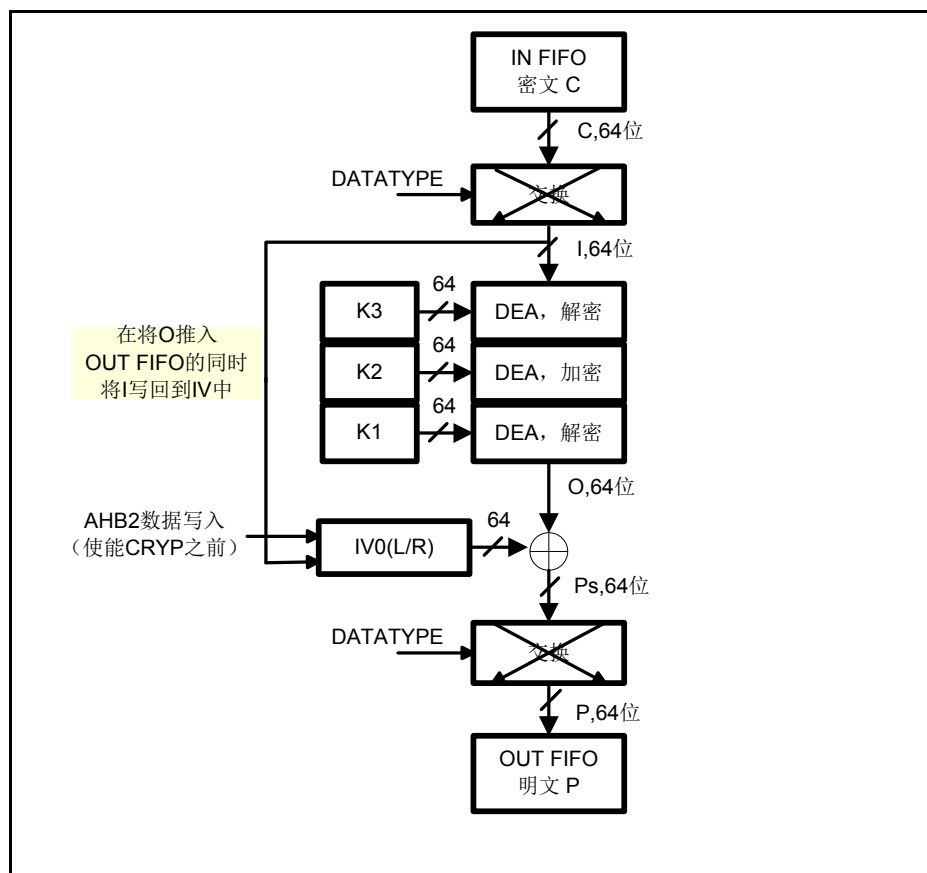


图 17-13 DES/TDES-CBC 模式解密

注：K = 密钥；C = 密文；I = 输入块；Ps = 交换前（解码时）或交换后（编码时）的明文；P = 明文；IV = 初始化向量。

操作示例:

1. 设置 CRYPT\_CON.CRYSEL = 1, 选择算法类型为 DES/TDES
2. 设置 CRYPT\_CON.ENCS = 0, 选择解密
3. 设置 CRYPT\_CON.TDES = 0 选择 DES, 设置 CRYPT\_CON.TDES = 1 选择 TDES, 用户还可以设置 CRYPT\_CON.DESKS 选择 TDES 使用 2 或者 3 个密钥。
4. 设置 CRYPT\_CON.MODE = 0, 选择 ECB 模式
5. 向 CRYPT\_KEY0...CRYPT\_KEY7 中填入密钥, 其中 KEY0/1 = K1, KEY2/3 = K2, KEY4/5 = K3
6. 向 CRYPT\_IV0/1 填入初始向量
7. 向 CRYPT\_DATA0/1 中写入需要解密的密文
8. 设置 CRYPT\_CON.GO = 1, 启动解密
9. 通过 CRYPT\_IF.DONE 判断解密是否完成, 如果 CRYPT\_IF.DONE = 0, 则解密完成
10. 从 CRYPT\_DATA0/1 中读出已经解密的明文
11. 回到步骤 7 继续进行解密
12. 所有解密完成

## 17.5 SM4 功能描述

SM4 密码算法的分组长度为 128 比特, 密钥长度为 128 比特。加密算法与密钥扩展算法都采用 32 轮非线性迭代结构。解密算法与加密算法的结构相同, 只是轮密钥的使用顺序相反, 解密轮密钥是加密轮密钥的逆序。

### 17.5.1 SM4-ECB 模式加密

下图介绍了 SM4 电子密码本 (SM4-ECB) 模式加密。

在 SM4-ECB 加密中, 执行位/字节/半字交换后, 128 位明文数据块 (P) 将用作输入块 (I)。输入块在加密状态下使用 128 位密钥进行处理。执行位/字节/半字交换后, 生成的 128 位输出块 (O) 将用作密文 (C)。该密文随后给到结果寄存器。



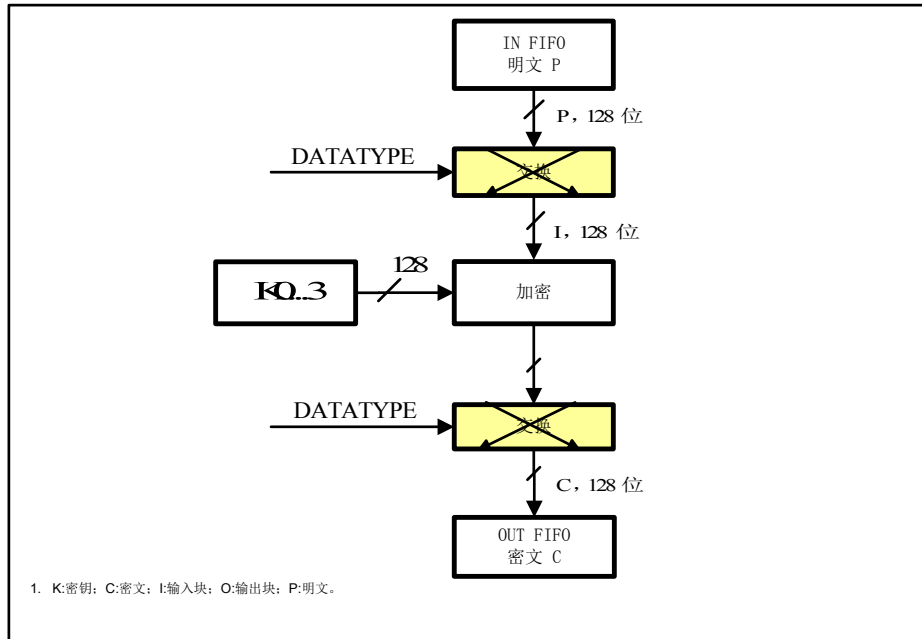


图 17-14 SM4-ECB 模式加密

### 17.5.2 SM4-ECB模式解密

下图介绍了 SM4 电子密码本 (SM4-ECB) 模式解密。

在 SM4-ECB 解密中, 执行位/字节/半字交换后, 128 位密文块 (C) 将用作输入块 (I)。执行位/字节/半字交换, 生成的 128 位输出块 (O) 将产生明文 (P)。

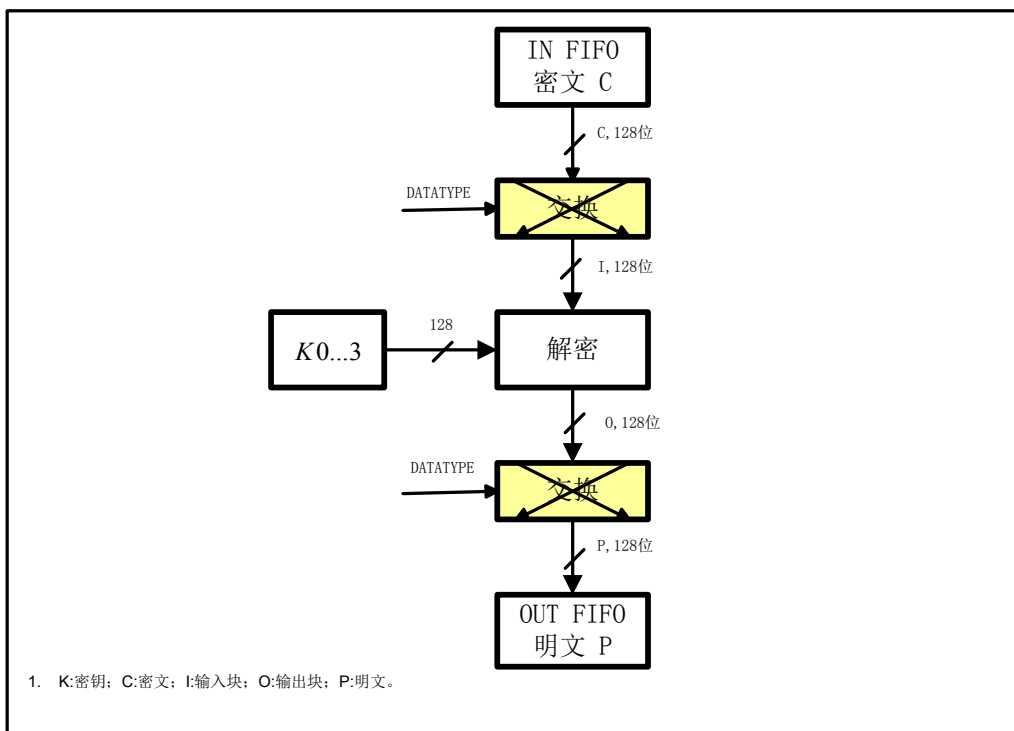


图 17-15 SM4-ECB 模式解密

### 17.5.3 SM4-CBC模式加密

SM4 加密分组链接 (SM4-CBC) 模式解密如下图所示。

在 SM4-CBC 加密中, 执行位/字节/半字交换后所获得的第一个输入块 (I1) 是通过一个 128 位初始化向量 IV 与第一个明文数据块 (P1) 进行异或运算形成的 ( $I1 = IV \oplus P1$ )。输入块加密状态下使用 128 位密钥进行处理。生成的 128 位输出块 (O1) 将直接用作密文 (C1), 即  $C1 = O1$ 。然后, 第一个密文块与第二个明文数据块进行异或运算, 从而生成第二个输入块 ( $I2 = (C1 \oplus P2)$ )。注意, 此时的 I2 和 P2 指的是第二个块。第二个输入块加密处理而生成第二个密文块。此加密处理会不断将后续密文块和明文块链接到一起, 直到消息中最后一个明文块得到加密为止。如果消息中包含的数据块数不是整数, 则应按照应用程序指定的方式对最后的不完整数据块进行加密。

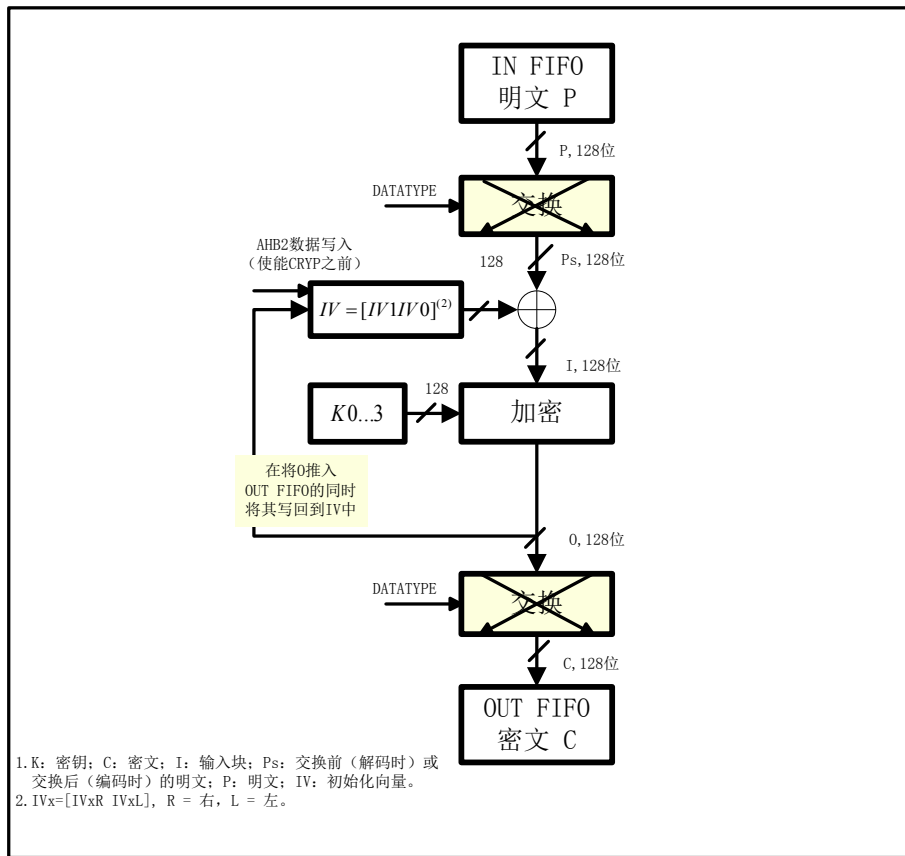


图 17-16 SM4-CBC 模式加密

### 17.5.4 SM4-CBC模式解密

在 SM4-CBC 解密 (请参见图 17-17) 中, 第一个 128 位密文块 (C1) 将直接用作输入块 (I1)。输入块在解密状态下使用 128 位密钥进行处理。生成的输出块与 128 位初始化向量 IV (必须与加密期间使用的相同) 进行异或运算, 从而生成第一个明文块 ( $P1 = O1 \oplus IV$ )。然后, 第二个密文块将用作下一个输入块, 并进行解密处理。生成的输出块与第一个密文块进行异或运算, 从而生成第二个明文数据块 ( $P2 = O2 \oplus C1$ )。(注意, P2 和 O2 指的是第二个数据块。) SM4-CBC 解密过程将以此方式继续进行, 直到最后一个完整密文块得到解密为止。必须按照应用程序指定的方式对不完整数据块密文进行解密。

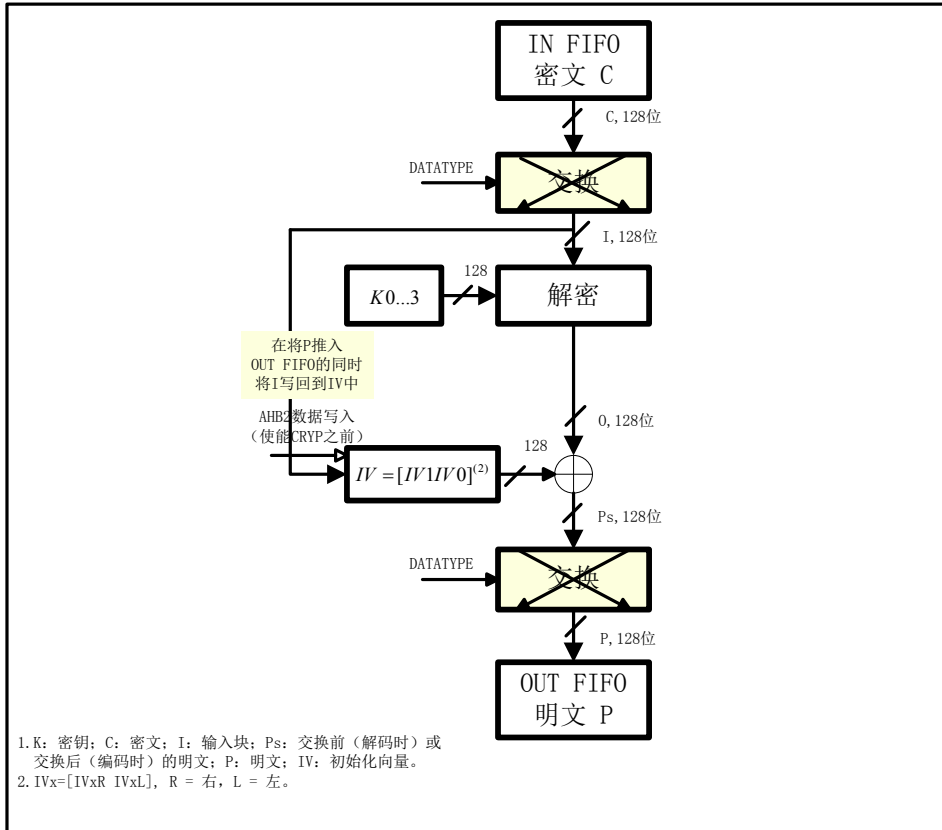


图 17-17 SM4-CBC 模式解密

## 17.6 FIFO使用

为了用户使用方便以及在加解密过程中使用 DMA，在 CRYPT 模块中添加了 FIFO 模式，在配置成 FIFO 模式后，可以通过寄存器 CRYPT\_FIFO 进行数据输入输出操作，当 AES 运算时，通过 FIFO 写入 4 个 Word 后自动触发 AES 运算；当 DES 运算时，FIFO 写入 2 个 Word 后自动触发 DES/TDES 运算。

另外配置 FIFOODR 可选择 FIFO 写入和读出的顺序。AES 模式时，FIFOODR 为 0 则优先读出 RES0 或写入 DATA0，FIFOODR 为 1 则优先读出 RES3 或写入 DATA3。DES 和 TDES 模式时，FIFOODR 为 0 则优先读出 RES0 或写入 DATA0，FIFOODR 为 1 则优先读出 RES1 或写入 DATA1。

## 17.7 使用DMA传输数据

在使用 DMA 传输数据前需配置成 FIFO 模式，并且在将 DMAEN 置 1 的同时触发 DMA 传输。DMA 传输需将读数据和写数据配置在不同的通道上，硬件自动分别控制读数据和写数据的 DMA 请求信号，直至所需数据数量全部转换完成。

使用 DMA 传输必须配置数据总量为单次转换字数的整数倍。

## 17.8 特殊功能寄存器

### 17.8.1 寄存器列表

CRYPT 寄存器列表		
名称	偏移地址	描述
CRYPT 基地址: 4008_5800 <sub>H</sub>		
CRYPT_DATA0	000 <sub>H</sub>	CRYPT 数据寄存器 0
CRYPT_DATA1	004 <sub>H</sub>	CRYPT 数据寄存器 1
CRYPT_DATA2	008 <sub>H</sub>	CRYPT 数据寄存器 2
CRYPT_DATA3	00C <sub>H</sub>	CRYPT 数据寄存器 3
CRYPT_KEY0	010 <sub>H</sub>	CRYPT 密钥寄存器 0
CRYPT_KEY1	014 <sub>H</sub>	CRYPT 密钥寄存器 1
CRYPT_KEY2	018 <sub>H</sub>	CRYPT 密钥寄存器 2
CRYPT_KEY3	01C <sub>H</sub>	CRYPT 密钥寄存器 3
CRYPT_KEY4	020 <sub>H</sub>	CRYPT 密钥寄存器 4
CRYPT_KEY5	024 <sub>H</sub>	CRYPT 密钥寄存器 5
CRYPT_KEY6	028 <sub>H</sub>	CRYPT 密钥寄存器 6
CRYPT_KEY7	02C <sub>H</sub>	CRYPT 密钥寄存器 7
CRYPT_IVR0	030 <sub>H</sub>	CRYPT 初始向量寄存器 0
CRYPT_IVR1	034 <sub>H</sub>	CRYPT 初始向量寄存器 1
CRYPT_IVR2	038 <sub>H</sub>	CRYPT 初始向量寄存器 2
CRYPT_IVR3	03C <sub>H</sub>	CRYPT 初始向量寄存器 3
CRYPT_RES0	040 <sub>H</sub>	CRYPT 结果寄存器 0
CRYPT_RES1	044 <sub>H</sub>	CRYPT 结果寄存器 1
CRYPT_RES2	048 <sub>H</sub>	CRYPT 结果寄存器 2
CRYPT_RES3	04C <sub>H</sub>	CRYPT 结果寄存器 3
CRYPT_CON	050 <sub>H</sub>	CRYPT 控制寄存器
CRYPT_IF	054 <sub>H</sub>	CRYPT 中断标志寄存器
CRYPT_IFC	058 <sub>H</sub>	CRYPT 状态清零寄存器
CRYPT_FIFO	05C <sub>H</sub>	CRYPT 缓存寄存器

## 17.8.2 寄存器描述

### 17.8.2.1 CRYPT数据寄存器X (CRYPT\_DATAx) (x = 0..3)

CRYPT 数据寄存器 X (CRYPT_DATAx) (x = 0..3)																															
偏移地址: 000 <sub>H</sub> ~ 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAx																															

DATAx	Bit 31-0	R/W	<b>CRYPT 待处理数据</b> AES 数据寄存器共 4 个字 (16 个字节); SM4 数据寄存器共 4 个字 (16 个字节); DES 数据寄存器共 2 个字 (8 个字节); 加密前, 写入明文数据; 解密前, 写入密文数据
-------	----------	-----	--

### 17.8.2.2 CRYPT密钥寄存器X (CRYPT\_KEYx) (x = 0..7)

CRYPT 密钥寄存器 X (CRYPT_KEYx) (x = 0..7)																															
偏移地址: 010 <sub>H</sub> ~ 02C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYx																															

KEYx	Bit 31-0	R/W	<b>CRYPT 密钥</b> AES 密钥寄存器可以是 4 个字 (128 位) SM4 密钥寄存器可以是 4 个字 (128 位) AES 密钥寄存器可以是 6 个字 (192 位) AES 密钥寄存器可以是 8 个字 (256 位) DES/3DES 密钥寄存器可以是 2 个字 (56 位) 3DES 2Keys 密钥寄存器可以是 4 个字 (112 位) 3DES 3Keys 密钥寄存器可以是 6 个字 (168 位)
------	----------	-----	--

17.8.2.3 CRYPT初始向量寄存器X (CRYPT\_IVx) (x = 0..3)

CRYPT 初始向量寄存器 X (CRYPT_IVx) (x = 0..3)																															
偏移地址: 030 <sub>H</sub> ~03C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVx																															

IVx	Bit 31-0	R/W	<p><b>CRYPT 初始向量</b>                  AES、SM4 在 CBC 模式时用 4 个初始向量寄存器，共 4 个字，DES 在 CBC 模式时用 2 个初始向量寄存器，共 2 个字                  AES 在 CTR、GCM 模式用作累加器。</p>
-----	----------	-----	---

17.8.2.4 CRYPT结果寄存器X (CRYPT\_RESx) (x = 0..3)

CRYPT 结果寄存器 X (CRYPT_RESx) (x = 0..3)																															
偏移地址: 040 <sub>H</sub> ~04C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESx																															

RESx	Bit 31-0	R	<p><b>CRYPT 结果</b>                  AES 结果寄存器共 4 个字 (16 个字节);                  SM4 结果寄存器共 4 个字 (16 个字节);                  DES 结果寄存器共 2 个字 (8 个字节);                  加密后，读出密文数据；解密后，读出明文数据</p>
------	----------	---	---

17.8.2.5 CRYPT控制寄存器 (CRYPT\_CON)

CRYPT 控制寄存器 (CRYPT_CON)																																							
偏移地址: 50 <sub>H</sub>																																							
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
CRYSEL														Reserved														DMAEN	FIFOODR	FIFOEN	DESKS	TDES	TYPE	IE	IVEN	MODE	AESKS	ENCS	GO

CRYSEL	Bit 31-30	R/W	<b>算法标准选择位</b> 00: AES 10: DES/TDES 01: SM4 11: 保留
Reserved	Bit 29-18	—	保留
SM4INVALID	Bit 17	W	<b>SM4 数据写入控制位</b> 0: 禁止 1: 使能
SM4SOFTTRST	Bit 16	R/W	<b>SM4 软件复位控制位</b> 0: 禁止 1: 使能
RESCLR	Bit 15	W1	<b>结果寄存器清零位</b> 0: 无操作 1: 结果寄存器清零
DMAEN	Bit 14	R/W	<b>DMA 使能位</b> 0: 禁止 1: 使能
FIFOODR	Bit 13	R/W	<b>FIFO 顺序选择位</b> 0: DATA0/RES0 优先 1: DATA3/RES3 优先 (AES); DATA1/RES1 优先 (DES/TDES)
FIFOEN	Bit 12	R/W	<b>FIFO 使能位</b> 0: 禁止 1: 使能 注1: 使用FIFO后当写入数据量达到预定数量时可自动触发加解密, 同时处理完成后可从FIFO中读出结果。 注2: FIFO写入的数据可在DATA寄存器中体现, 读出的数据可在RES寄存器中体现
DESKS	Bit 11	R/W	<b>TDES 密钥选择位</b> 0: 2 个密钥

			1: 3 个密钥
TDES	Bit 10	R/W	<b>TDES 使能位</b> 0: 禁止 1: 使能
TYPE	Bit 9-8	R/W	<b>数据类型选择位</b> 00: 32 位数据, 不交换 01: 16 位数据或半字交换 10: 8 位数据或字节交换 11: 1 位数据或位串交换
IE	Bit 7	R/W	<b>中断使能位</b> 0: 禁止 1: 使能
IVEN	Bit 6	R/W	<b>IV 使能位</b> 0: 禁止 1: 使能 注: 使能后应用于CBC模式的首次运算, 完成后硬件自动清0
MODE	Bits 5-4	R/W	<b>模式选择位</b> 00: ECB 01: CBC 10: CTR (基于AES) 11: GCM (使用MULTH)
AESKS	Bit 3-2	R/W	<b>AES 密钥选择位</b> 00: 128 位 01: 192位 10: 256位 11: 保留
ENCS	Bit 1	R/W	<b>加解密选择位</b> 0: 解密 1: 加密
GO	Bit 0	W1	<b>CRYPT 启动触发位</b> 0: 无操作 1: 启动加密或解密



17.8.2.6 CRYPT中断标志寄存器 (CRYPT\_IF)

CRYPT 中断标志寄存器 (CRYPT_IF)																															
偏移地址: 54 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							DONE	Reserved	Reserved			SM4IF	MULTHIF	DESIF	AESIF

Reserved	Bit 31-9	—	保留
DONE	Bit 8	R	<b>加解密完成标志位</b> 0: 未开始或已完成 1: 未完成
Reserved	Bit 7-4	—	保留
SM4IF	Bit 3	R	<b>SM4 中断标志位</b> 0: 未完成 SM4 运算或标志位已被清除 1: 完成SM4运算 注: 该位由硬件自动置1, 软件操作 CRYPT_ICFR 清零。
MULTHIF	Bit 2	R	<b>MULTH 中断标志位</b> 0: 未完成 MULTH 运算或标志位已被清除 1: 完成MULTH运算 注: 该位由硬件自动置1, 软件操作MULTHIFC清零。
DESIF	Bit 1	R	<b>DES 中断标志位</b> 0: 未完成 DES 运算或标志位已被清除 1: 完成DES运算 注: 该位由硬件自动置1, 软件操作DESIFC清零。
AESIF	Bit 0	R	<b>AES 中断标志位</b> 0: 未完成 AES 运算或标志位已被清除 1: 完成AES运算 注: 该位由硬件自动置1, 软件操作AESIFC清零。

### 17.8.2.7 CRYPT状态清零寄存器 (CRYPT\_IFC)

CRYPT 中断标志清零寄存器 (CRYPT_IFC)																															
偏移地址: 58 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												SM4IFC	MULTHIF	DESIF	AESIF

Reserved	Bit 31-4	—	保留
SM4IFC	Bit 3	W1	<b>SM4 中断标志位清零</b> 0: 无操作 1: SM4IF清零操作
MULTHIFC	Bit 2	W1	<b>MULTH 中断标志位清零</b> 0: 无操作 1: MULTHIF清零操作
DESIFC	Bit 1	W1	<b>DES 中断标志位清零</b> 0: 无操作 1: DESIF清零操作
AESIFC	Bit 0	W1	<b>AES 中断标志位清零</b> 0: 无操作 1: AESIF清零操作

### 17.8.2.8 CRYPT缓存寄存器 (CRYPT\_FIFO)

CRYPT FIFO 寄存器 (CRYPT_FIFO)																															
偏移地址: 5C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFO																															

FIFO	Bit 31-0	R/W	数据缓冲 需通过FIFOEN位使能该寄存器的读写功能, 运算前写入待处理数据, 运算后读取结果

## 第18章 HASH硬件加速模块(HASH)

### 18.1 概述

HASH 杂凑算法运算单元通过硬件实现 SHA256、SM3 算法标准。

SHA256 是国际通用的杂凑算法标准，符合 FIPS180-2 标准。

SM3 算法是国密局认证的商用密码杂凑算法，适用于商用密码应用中的数字签名和验证、消息认证码的生成与验证以及随机数的生成，可满足多种密码应用的安全需求。

SM3 杂凑算法定义为对长度为  $l(l < 264)$  比特的消息  $m$ ，经过填充和迭代压缩，生成杂凑值，杂凑值长度为 256 比特。

SM3 硬件算法引擎 (SM3 Engine) 特别用于 SM3 算法中的压缩算法实现。

### 18.2 工作原理

芯片内置 SM3 硬件加速模块，完成 SM3 算法中压缩算法的硬件操作。设置 SM3 控制寄存器 CTRL 的 EN=1 时，使能压缩功能，根据扩展出的消息  $W_0 \sim W_{67}$ 、 $W'_0 \sim W'_{63}$  ( $W'_i = W_i \wedge W_{i+4}$ )，计算出压缩值。

操作流程为：通过 SM3 数据输入寄存器 MIDDATA 输入 8 个 32 位的初始数据，然后进行压缩。SM3 的压缩过程包含 64 轮运算。在每轮运算中，分别将  $W_i$  和  $W'_i$  按先后顺序 ( $W_i$  先输入， $W'_i$  后输入) 输入到 SM3 消息输入寄存器 DATAIN 中，然后使能 SM3 压缩功能。因为硬件模块会对寄存器 CTRL 的 EN 位清 0，所以需要在每轮运算中使能 SM3 压缩功能。最后通过 SM3 压缩结果输出寄存器 DATAOUT 读出 8 个字，即为压缩值。

### 18.3 HASH结构框图

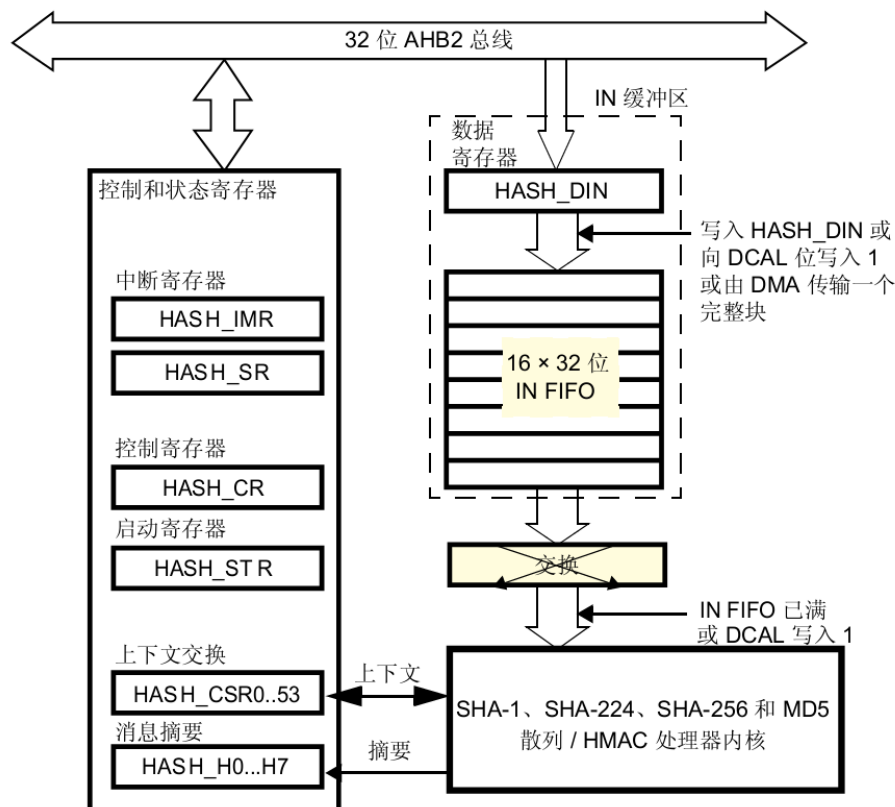


图 18-1 SM3 结构框图

硬件模块包括 4 个部分，分别是接口逻辑、寄存器组、FIFO、算法引擎。

线接口用来系统总线连接，提供寄存器接口，实现程序访问功能。

算法引擎主要实现 SM3 或 SHA256 的压缩算法，具体流程如下：

令 $A, B, C, D, E, F, G, H$ 为字寄存器,  $SS1, SS2, TT1, TT2$ 为中间变量, 压缩函数 $V^{i+1} = CF(V^{(i)}, B^{(i)})$ ,  $0 \leq i \leq n-1$ 。计算过程描述如下:

```

ABCDEF $GH \leftarrow V^{(i)}$ 
FOR  $j=0$  TO 63
     $SS1 \leftarrow ((A \lll 12) + E + (T_j \lll j)) \lll 7$ 
     $SS2 \leftarrow SS1 \oplus (A \lll 12)$ 
     $TT1 \leftarrow FF_j(A, B, C) + D + SS2 + W'_j$ 
     $TT2 \leftarrow GG_j(E, F, G) + H + SS1 + W_j$ 
     $D \leftarrow C$ 
     $C \leftarrow B \lll 9$ 
     $B \leftarrow A$ 
     $A \leftarrow TT1$ 
     $H \leftarrow G$ 
     $G \leftarrow F \lll 19$ 
     $F \leftarrow E$ 
     $E \leftarrow P_0(TT2)$ 
ENDFOR
 $V^{(i+1)} \leftarrow ABCDEF $GH \oplus V^{(i)}$$ 

```

其中, 字的存储为大端(big-endian)格式。

## 18.4 特殊功能寄存器

### 18.4.1 寄存器列表

HASH 寄存器列表		
名称	偏移地址	描述
SM3 基地址: 4008_6800 <sub>H</sub>		
HASH_DIN	000 <sub>H</sub>	HASH 消息输入寄存器
HASH_CTRL	004 <sub>H</sub>	HASH 控制寄存器
HASH_STAT	008 <sub>H</sub>	HASH 状态寄存器
HASH_RES0	00C <sub>H</sub>	HASH 结果输出寄存器 0
HASH_RES1	010 <sub>H</sub>	HASH 结果输出寄存器 1
HASH_RES2	014 <sub>H</sub>	HASH 结果输出寄存器 2
HASH_RES3	018 <sub>H</sub>	HASH 结果输出寄存器 3
HASH_RES4	01C <sub>H</sub>	HASH 结果输出寄存器 4
HASH_RES5	020 <sub>H</sub>	HASH 结果输出寄存器 5
HASH_RES6	024 <sub>H</sub>	HASH 结果输出寄存器 6
HASH_RES7	028 <sub>H</sub>	HASH 结果输出寄存器 7

## 18.4.2 寄存器描述

### 18.4.2.1 HASH消息输入寄存器 (HASH\_DIN)

HASH 消息输入寄存器 (HASH_DIN)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN																															

DIN	Bit31-0	W	输入每个分组的消息（读返回 0）（若采用 TRIG 触发 HASH 运算，除了最后一次写入需字节方式写入，其他写入必须以 32 位字写入）
-----	---------	---	---

### 18.4.2.2 HASH控制寄存器 (HASH\_CTRL)

HASH 控制寄存器 (HASH_CTRL)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TRIG	DATATYP	OUTTYP	ALGO	DMAEN	INTEN	EN		

Reserved	Bit31-9	—	保留
TRIG	Bit8	R/W	HASH 组运算触发位（本组数据不满 512bit 时可通过该位写 1 触发 HASH 运算） 0: 禁止 1: 使能（使能后自动清 0）
DATATYP	Bit7-6	R/W	HASH 数据类型选择位（若采用 TRIG 触发 HASH 运算，最后一次写入需字节方式写入） 00: 32bit, 无交换 01: 16bit, 半字交换 10: 8bit, 字节交换 11: 1bit, 比特交换
OUTTYP	Bit5	R/W	HASH 结果输出大小端选择位 0: 大端 1: 小端
ALGO	Bit4-3	R/W	HASH 算法选择位

			00: SM3 01: SHA256 1x: 保留
DMAEN	Bit2	R/W	<b>HASH DMA 使能位</b> 0: 禁止 1: 使能
INTEN	Bit1	R/W	<b>HASH 中断使能位</b> 0: 禁止 1: 使能
EN	Bit0	R/W	<b>HASH 功能使能位</b> （每次新的杂凑运算需重新使能该位） 0: 禁止 1: 使能（使能后自动清 0）

### 18.4.2.3 HASH状态寄存器（HASH\_STAT）

HASH 状态寄存器（HASH_STAT）																															
偏移地址：08 <sub>H</sub>																															
复位值：00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GRPNUM																Reserved			USED FIFO			Reserved						RND_DONE			

GRPNUM	Bit31-16	R	已完成杂凑运算的分组个数（每个分组 512bit），只可写 0。
Reserved	Bit15-12	—	保留
USED FIFO	Bit11-8	R	FIFO 中已经存入的有效字数
Reserved	Bit7-1	—	保留
RND_DONE	Bit0	R/W	单个分组 HASH 完成标志位（写 1 清 0） 0: 未完成 1: 已完成

### 18.4.2.4 HASH结果输出寄存器 0 (HASH\_RES0)

HASH 结果输出寄存器 0 (HASH_RES0)																															
偏移地址: 0c <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0																															

RES0	Bit31-0	R	输出 HASH 最后的运算结果 0
------	---------	---	-------------------

注: HASH 最后的运算结果, 共 256bit。

### 18.4.2.5 HASH结果输出寄存器 1 (HASH\_RES1)

HASH 结果输出寄存器 1 (HASH_RES1)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES1																															

RES1	Bit31-0	R	输出 HASH 最后的运算结果 1
------	---------	---	-------------------

### 18.4.2.6 HASH结果输出寄存器 2 (HASH\_RES2)

HASH 结果输出寄存器 2 (HASH_RES2)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES2																															

RES2	Bit31-0	R	输出 HASH 最后的运算结果 2
------	---------	---	-------------------



### 18.4.2.7 HASH结果输出寄存器 3 (HASH\_RES3)

HASH 结果输出寄存器 3 (HASH_RES3)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES3																															

RES3	Bit31-0	R	输出 HASH 最后的运算结果 3
------	---------	---	-------------------

### 18.4.2.8 HASH结果输出寄存器 4 (HASH\_RES4)

HASH 结果输出寄存器 4 (HASH_RES4)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES4																															

RES4	Bit31-0	R	输出 HASH 最后的运算结果 4
------	---------	---	-------------------

### 18.4.2.9 HASH结果输出寄存器 5 (HASH\_RES5)

HASH 结果输出寄存器 5 (HASH_RES5)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES5																															

RES5	Bit31-0	R	输出 HASH 最后的运算结果 5
------	---------	---	-------------------

### 18.4.2.10 HASH结果输出寄存器 6 (HASH\_RES6)

HASH 结果输出寄存器 6 (HASH_RES6)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES6																															

RES6	Bit31-0	R	输出 HASH 最后的运算结果 6
------	---------	---	-------------------

### 18.4.2.11 HASH结果输出寄存器 7 (HASH\_RES7)

HASH 结果输出寄存器 7 (HASH_RES7)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES7																															

RES7	Bit31-0	R	输出 HASH 最后的运算结果 7
------	---------	---	-------------------

## 第19章 椭圆曲线密码单元(ECC)

### 19.1 概述

ECC 算法模块是一款高性能、低成本的椭圆曲线密码算法硬件加速模块，支持素数域 P256 的椭圆曲线，可实现 ECDSA (NIST)、ECDH (NIST)、SM2 (签名、加解密、密钥交换) 等多种算法。

ECC 算法模块底层是模加、模减、模乘、模逆等运算，在此基础上，硬件可根据 ECC 标量乘(ECSM)算法特点实现多种运算组合。考虑到算法灵活性和便捷性，ECC 算法引擎并未直接完成全部 ECSM 运算，而是通过软件调用 ECC 算法引擎的运算功能实现 ECSM。

### 19.2 ECC结构框图

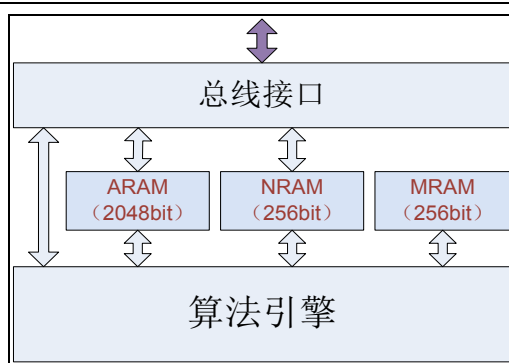


图 19-1 ECC 结构框图

硬件模块包括 3 个部分，分别是接口逻辑、算法引擎及专用存储器。

线接口用来系统总线连接，提供寄存器接口，实现程序访问功能。

算法引擎主要实现 ECSM 的底层运算，包括模加、模减、模乘、模拟以及 Montgomery 域转换；其中硬件自动实现各种组合运算以支持 Montgomery ladder 改进算法，包括 ZACAU'、DBLU'、ZADDC'、ZADDU'。

专用存储器包括 2048bits (64x32bit) 的 ARAM、256bits(8x32bits)的 NRAM 和 256bits(8x32bits)的 MRAM。其中 ARAM 和 NRAM 可供外部 CPU 访问，MRAM 不对外开放。

### 19.3 ECC缓存地址映射

0x000	ARAM
0x0FF	NRAM
0x100	
0x11F	保留
0x120	
0x2FF	REG_CON
0x300	REG_STA
0x304	REG_PARA
0x308	REG_SCAx
0x320~0x33F	REG_YCOx
0x340~0x35F	保留
0x3FF	

图 19-2 ECC 缓存地址映射

### 19.4 特殊功能寄存器

#### 19.4.1 寄存器列表

ECC 寄存器列表			
名称	偏移地址	类型	描述
ECC 基地址: 4000_9800 <sub>H</sub>			
ECC_CON	0300 <sub>H</sub>	R/W	ECC 控制寄存器
ECC_STA	0304 <sub>H</sub>	R/W	ECC 状态寄存器
ECC_PARA	0308 <sub>H</sub>	R/W	ECC 运算参数寄存器
ECC_SCALAR0	0320 <sub>H</sub>	R/W	ECC 点乘运算标量值 0
ECC_SCALAR1	0324 <sub>H</sub>	R/W	ECC 点乘运算标量值 1
ECC_SCALAR2	0328 <sub>H</sub>	R/W	ECC 点乘运算标量值 2
ECC_SCALAR3	032C <sub>H</sub>	R/W	ECC 点乘运算标量值 3
ECC_SCALAR4	0330 <sub>H</sub>	R/W	ECC 点乘运算标量值 4
ECC_SCALAR5	0334 <sub>H</sub>	R/W	ECC 点乘运算标量值 5
ECC_SCALAR6	0338 <sub>H</sub>	R/W	ECC 点乘运算标量值 6
ECC_SCALAR7	033C <sub>H</sub>	R/W	ECC 点乘运算标量值 7
ECC_YCO0	0340 <sub>H</sub>	R/W	ECC 点 Y 坐标值 0
ECC_YCO1	0344 <sub>H</sub>	R/W	ECC 点 Y 坐标值 1
ECC_YCO2	0348 <sub>H</sub>	R/W	ECC 点 Y 坐标值 2
ECC_YCO3	034C <sub>H</sub>	R/W	ECC 点 Y 坐标值 3
ECC_YCO4	0350 <sub>H</sub>	R/W	ECC 点 Y 坐标值 4
ECC_YCO5	0354 <sub>H</sub>	R/W	ECC 点 Y 坐标值 5
ECC_YCO6	0358 <sub>H</sub>	R/W	ECC 点 Y 坐标值 6
ECC_YCO7	035C <sub>H</sub>	R/W	ECC 点 Y 坐标值 7

## 19.4.2 寄存器描述

### 19.4.2.1 ECC控制寄存器 (ECC\_CON)

ECC 控制寄存器 (ECC_CON)																															
偏移地址: 300 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RUN	Comb	Remap	ComMod	Reserved													OPL	OPR	RES	Mode	INTEN										

寄存器名称	位域	读写属性	功能描述
RUN	Bit 31	R/W	<b>ECC 引擎单个运算启动位</b> 0: 不启动 1: 启动
COMB	Bit 30	W	<b>ECC 引擎组合运算启动位 (只写, 读为 0)</b> 0: 单个运算 1: 组合运算, 并启动 ECC 引擎
REMAP	Bit 29	R/W	<b>ARAM 中部分数据重映射</b> 0: 不重映射 1: 重映射 (T1<->T4, T2<->T5)
COMBMD	Bits 28-27	R/W	<b>专用组合运算选择位: (不对用户开放)</b> 00: ZACAU 01: DBLU 10: ZADDC 11: ZADDU
Reserved	Bits 26-13	—	保留
OPL	Bits 12-10	R/W	<b>操作数 L 在 ARAM 的区地址选择位</b> 000~111: T0~T7
OPR	Bits 9-7	R/W	<b>操作数 R 在 ARAM 的区地址选择位</b> 000~111: T0~T7
RES	Bits 6-4	R/W	<b>运算结果在 ARAM 的区地址选择位</b> 000~111: T0~T7
MODE	Bits 3-1	R/W	<b>运算模式设置位</b> 000: 参数 H 计算 001: Mont 模乘 010: 模加 011: 模减 100: 模逆 101: Mont 域转出 110: Mont 域转入 111: ECC 点乘运算

INTEN	Bit 0	R/W	<b>ECC 模块中断使能位</b> 0: 禁止 1: 使能
-------	-------	-----	--------------------------------------

注: Comb 置 1 后, 部分寄存器位会在组合运算中动态变化, 包括 RUN、OPL、OPR、RES、MODE。

### 19.4.2.2 ECC状态寄存器 (ECC\_STA)

ECC 状态寄存器 (ECC_STA)																															
偏移地址: 304 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											MI_ZERO	ECC_OVER	ERR	DONE	WK

Reserved	Bits 31-5	—	保留
MI_ZERO	Bit 4	R/C_W1	<b>模逆运算 0 参数标志 (软件写 1 清零)</b> 0: 模拟运算参数为非 0 1: 模拟运算参数为 0 (若在 ECC 运算中发生, 可认为是无穷远点)
ECC_OVER	Bit 3	R/C_W1	<b>ECC 运算结束标志 (软件写 1 清零)</b> 0: ECC 运算任务未完成 1: ECC 运算任务完成
ERR	Bit 2	R/C_W1	<b>出错状态位 (软件写 1 清零)</b> 0: 无错误发生 1: 有错误发生
DONE	Bit 1	R/C_W1	<b>运算结束标志 (软件写 1 清零)</b> 0: 当前运算任务未完成 1: 当前运算任务完成
WK	Bit 0	R	<b>运算进行标志</b> 0: 无正在进行运算 1: 正在进行运算

### 19.4.2.3 ECC运算参数寄存器 (ECC\_PARA)

ECC 运算参数寄存器 (ECC_PARA)																															
偏移地址: 308 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PARA																															

PARA	Bits 31~0	R/W	ECC模乘运算参数(根据模数N低32位算得)
------	-----------	-----	------------------------

### 19.4.2.4 ECC标量数据寄存器 (ECC\_SCALARx) (x = 0..7)

ECC 标量数据寄存器 (ECC_SCALARx) (x = 0..7)																															
偏移地址: 320 <sub>H</sub> ~ 33C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCALARx																															

SCALARx	Bits 31~0	R/W	ECC点乘运算标量值
---------	-----------	-----	------------

### 19.4.2.5 ECC标量数据寄存器 (ECC\_YCOx) (x = 0..7)

ECC 标量数据寄存器 (ECC_YCOx) (x = 0..7)																															
偏移地址: 340 <sub>H</sub> ~ 35C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
YCOx																															

YCOx	Bits 31~0	R/W	点Y坐标值 (256bit)
------	-----------	-----	----------------

## 第20章 真随机发生器(TRNG)

### 20.1 概述

真随机数发生器 (TRNG) 可生产 1 位串行真随机数或 8/16/32 位并行真随机数。

### 20.2 特性

- ◇ 支持可编程的随机数位宽
- ◇ 支持可编程的种子值
- ◇ 支持随机性修正模式
- ◇ 支持随机序列错误检测

### 20.3 结构框图

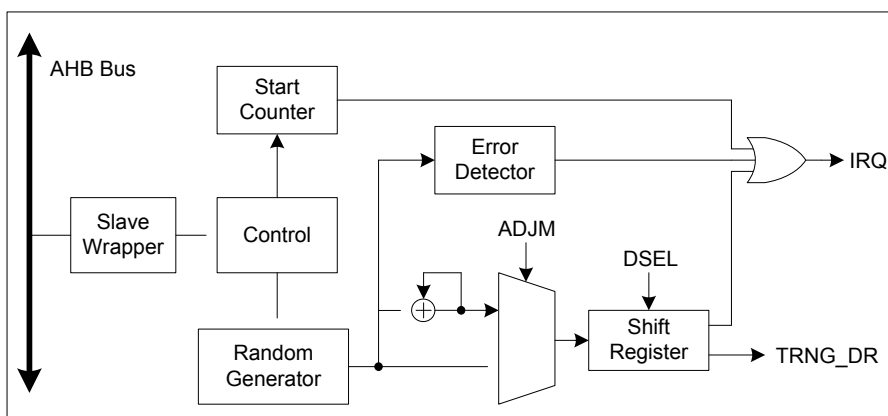


图 20-1 TRNG 结构框图

### 20.4 功能描述

#### 20.4.1 初始化时间

TRNG 使能后需要经过一段时间初始化才可产生新的随机数据，初始化时间在 TRNG\_START 寄存器配置。根据当前 HCLK 时钟频率，计算出 TRNG\_START 的值，使初始化时间不低于 1ms。

计算公式为：

$$T_{START} = T_{HCLK1} \times 2^{TSTART+1}$$

#### 20.4.2 错误管理

出现随机序列错误时，SERR 被硬件置 1，随机数产生被中断。若使能了序列错误中断，则产生该中断。如果 TRNG\_DR 寄存器中有新产生的随机数，不能使用该随机数。

应执行以下操作：置位 TRNG\_IFCR 寄存器的 SERR 位，将 TRNG\_EN 位清零并置 1，



以便重新初始化和重新启动 TRNG。

### 20.4.3 随机数生成时间

随机数生成时间由三者共同决定: TRNG\_CR 寄存器中的 DSEL 位、ADJM 位和 ADJC 位。

随机数生成时间公式为:

$$T_{\text{RNG}} \approx 5 \times D \times C^M \text{ (单位 us) (误差范围+66.66\%/-33.33\%)}$$

DSEL	系数 D	ADJM	系数 M	ADJC	系数 C
00	1	0	0	00	2
01	8			01	4
10	16	1	1	10	6
11	32			11	8

表 20-1 随机数生成时间系数表

### 20.4.4 随机数种子

因为随机序列发生器来自模拟信号, 电源和地上的干扰或环境温度的变化等都会导致随机性变差、输出序列的相邻位的自相关系数升高。通过设置一种随机的种子值, 可能使随机序列的随机性能更加优化, 降低自相关性。

种子的类型可选择为使用上一次产生的随机数或使用 TRNG\_SEED 寄存器值, 若使用 TRNG\_SEED 寄存器值, 则在每次读取新的随机数时同时更新 TRNG\_SEED 寄存器。种子的类型也可选择固定为 0 或 1。

### 20.4.5 操作流程

要运行 TRNG, 请按以下步骤操作:

1. 如果需要, 使能中断 (TRNG\_IER 相应位置 1)。准备好随机数时或出现错误时生成中断。
2. 通过将 TRNG\_CR 寄存器中的 TRNGEN 位置 1 使能随机数产生。这会激活随机数生成器、移位寄存器和错误检测器。若果需要随机性更高的数据, 可以将 ADJM 位置 1 使能修正模式, 但带来随机数生成时间会变长。配置 ADJC 可改变修正模式系数, 系数越大, 随机性越高, 但随机数生成时间也越长。
3. 每次中断时, 检查确认未出现错误 (TRNG\_SR 寄存器中的 SERR 位应为 0, TRNG\_IFR 寄存器中的 SERR 也为 0), 并且随机数已准备就绪 (TRNG\_SR 寄存器中的 DAVLD 位为 1)。然后即可读取 TRNG\_DR 寄存器中的内容。

按照 FIPS PUB (联邦信息处理标准出版物) 140-2 的要求, 将 TRNGEN 位置 1 后产生的第一个随机数不应使用, 但应保存起来, 与产生的下一个随机数进行比较。随后产生的每个随机数都需要与产生的上一个随机数进行比较。如果任何一对进行比较的数字相等, 则测试失败 (连续随机数发生器测试)。

操作流程示意图如下所示。

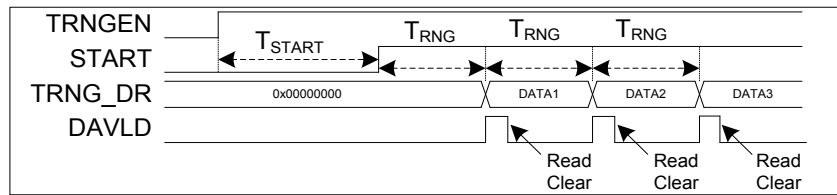


图 20-2 操作流程示意图

## 20.5 特殊功能寄存器

### 20.5.1 寄存器列表

TRNG 寄存器列表			
名称	偏移地址	类型	描述
TRNG 基地址: 4008_5C00 <sub>H</sub>			
TRNG_CR	000 <sub>H</sub>	R/W	TRNG 控制寄存器
TRNG_SR	004 <sub>H</sub>	R	TRNG 状态寄存器
TRNG_DR	008 <sub>H</sub>	R	TRNG 数据寄存器
TRNG_SEED	00C <sub>H</sub>	R/W	TRNG 种子寄存器
TRNG_CFGR	010 <sub>H</sub>	R/W	TRNG 配置寄存器
TRNG_IER	014 <sub>H</sub>	R/W	TRNG 中断使能寄存器
TRNG_IFR	018 <sub>H</sub>	R	TRNG 中断事件寄存器
TRNG_IFCR	01C <sub>H</sub>	W	TRNG 中断事件清零寄存器
TRNG_ISR	020 <sub>H</sub>	R	TRNG 中断状态寄存器

## 20.5.2 寄存器描述

### 20.5.2.1 TRNG控制寄存器 (TRNG\_CR)

TRNG 控制寄存器 (TRNG_CR)																																					
偏移地址: 00H																																					
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved																ADJC		Reserved						SDSEL		DSEL		Reserved						POSTEN	TRNGSEL	ADJM	TRNGEN

Reserved	Bits 31-18	—	保留
ADJC	Bits 17-16	R/W	<b>TRNG 修正系数</b> 注: 仅在修正模式使能时有效, 修正系数越大, 数据的随机性越高, 但随机数据生成时间会约长
Reserved	Bits 15-12	—	保留
SDSEL	Bits 11-10	R/W	<b>TRNG 种子类型选择位</b> 00: 固定为 0 01: 固定为 1 10: 使用上一次产生随机数 11: 使用 SEED 寄存器值
DSEL	Bits 9-8	R/W	<b>TRNG 数据类型选择位</b> 00: 1 位 01: 8 位 10: 16 位 11: 32 位 注: 数据位宽越短, 随机数生成时间越短, 但累积生成相同 Bit 量的时间是固定的
Reserved	Bits 7-4	—	保留
POSTEN	Bit 3	R/W	<b>数据后处理使能位</b> 0: 禁止 1: 使能
TRNGSEL	Bit 2	R/W	<b>TRNG 源选择位</b> 0: TRNG0 (不支持 TRNG0) 1: TRNG1 注: 仅可设置为1, 不可设置为0
ADJM	Bit 1	R/W	<b>TRNG 修正模式使能位</b> 0: 禁止 1: 使能 注: 使能修正模式可提高数据的随机性, 但相应地随机数据的生成时间会变长

TRNGEN	Bit 0	R/W	TRNG 使能位 0: 禁止 1: 使能
--------	-------	-----	----------------------------

### 20.5.2.2 TRNG状态寄存器 (TRNG\_SR)

TRNG 状态寄存器 (TRNG_SR)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TRNG1_SEED	TRNG1_CLK	TRNG1_SO	TRNG0_CLK	TRNG0_SO								Reserved								OVER	SERR	DAVLD	START			

Reserved	Bits 31-29	—	保留
TRNG1_SEED	Bit 28	R	TRNG1 种子
TRNG1_CLK	Bit 27	R	TRNG1 时钟
TRNG1_SO	Bit 26	R	TRNG1 数据
TRNG0_CLK	Bit 25	R	TRNG0 时钟 (不支持 TRNG0)
TRNG0_SO	Bits 24-16	R	TRNG0 数据 (不支持 TRNG0)
Reserved	Bits 15-4	—	保留
OVER	Bit 3	R	序列溢出状态位 (仅用于 TRNG0) 0: 当前序列无溢出 1: 当前序列有溢出
SERR	Bit 2	R	序列错误状态位 0: 当前序列无错误 1: 当前序列有错误 注: 当硬件检测到超过64个连续位具有相同值 (0或1), 或者超过32个连续交替的0和1 (01010101……01) 则表示序列出错
DAVLD	Bit 1	R	当前数据有效状态位 0: 数据未生成或已被读取 1: 数据有效 注: 该位在模块关闭或随机数被读取时清零
START	Bit 0	R	初始化状态位 0: 初始化未完成 1: 初始化完成 注: 该位在模块关闭时被清零

### 20.5.2.3 TRNG数据寄存器 (TRNG\_DR)

TRNG 数据寄存器 (TRNG_DR)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

DATA	Bits 31-0	R	<p><b>生成的随机数</b></p> <p>注1: 当状态寄存器中数据有效位为有效时才可读取</p> <p>注2: 该数据实际有效宽度为数据类型选择的宽度</p>
------	-----------	---	---

### 20.5.2.4 TRNG种子寄存器 (TRNG\_SEED)

TRNG 种子寄存器 (TRNG_SEED)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															

SEED	Bits 31-0	R/W	<p><b>种子值</b></p> <p>注1: 种子值可在每次生成随机数时进行更新, 也可固定为某值</p> <p>注2: 该数据实际有效宽度为数据类型选择的宽度</p>
------	-----------	-----	--

### 20.5.2.5 TRNG初始化时间寄存器 (TRNG\_CFGR)

TRNG 配置寄存器 (TRNG_CFGR)																															
偏移地址: 010 <sub>H</sub>																															
复位值: 00000001_11111111_00000111_00000111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					TOPLMT											Reserved				CKDIV				Reserved				TSTART			

Reserved	Bits 31-25	—	保留
TOPLMT	Bits 24-16	R/W	随机数上限配置 (仅用于 TRNG0, 支持 TRNG0) 默认值 0x1FF
Reserved	Bits 15-12	—	保留
CKDIV	Bits 11-8	R/W	时钟分频配置位 (仅用于 TRNG0, 支持 TRNG0) $Freq_{CK} = Freq_{HCLK1} / 2^{CKDIV+1}$
Reserved	Bits 7-3	—	保留
TSTART	Bits 2-0	R/W	初始化时间配置位 $T_{START} = T_{RNG} \times 2^{TSTART+1}$ 注: 根据当前TRNG数据流的频率值配置初始化时间, 需保证初始化时间不少于1ms

### 20.5.2.6 TRNG中断使能寄存器 (TRNG\_IER)

TRNG 中断使能寄存器 (TRNG_IER)																															
偏移地址: 014 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										SERR	DAVLD	START			

Reserved	Bits 31-3	—	保留
SERR	Bit 2	R/W	序列错误中断使能位 0: 禁止 1: 使能
DAVLD	Bit 1	R/W	数据有效中断使能位

			0: 禁止 1: 使能
START	Bit 0	R/W	初始化完成中断使能位 0: 禁止 1: 使能

### 20.5.2.7 TRNG中断事件寄存器 (TRNG\_IFR)

TRNG 中断事件寄存器 (TRNG_IFR)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										SERR	DAVLD	START			

Reserved	Bits 31-3	—	保留
SERR	Bit 2	R	<b>序列错误状态位</b> 0: 未检测到序列错误或标志位已被清除 1: 已检测到序列错误 注1: 当硬件检测到超过64个连续位具有相同值(0或1), 或者超过32个连续交替的0和1 (01010101……01) 则表示序列出错 注2: 该位通过操作TRNG_ICFR清零。
DAVLD	Bit 1	R	<b>数据有效标志位</b> 0: 数据无效或标志位已被清除 1: 数据有效 注: 该位通过操作 TRNG_ICFR 清零。若使用该位的 DMA 功能, DMA 应答后自动将该位清零。
START	Bit 0	R	<b>初始化完成标志位</b> 0: 初始化未完成或标志位已被清除 1: 初始化完成 注: 该位通过操作TRNG_ICFR清零。

### 20.5.2.8 TRNG中断事件清零寄存器 (TRNG\_IFCR)

TRNG 中断事件清零寄存器 (TRNG_IFCR)																															
偏移地址: 01C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											SERR	DAVLDC	STARTC		

Reserved	Bits 31-3	—	保留
SERRC	Bit 2	W1	<b>序列错误状态位清零</b> 0: 无操作 1: 序列错误状态位清零 注: 该位写1清除, 写0无效
DAVLDC	Bit 1	W1	<b>数据有效标志位清零</b> 0: 无操作 1: 数据有效标志位清零 注: 该位写1清除, 写0无效
STARTC	Bit 0	W1	<b>初始化完成标志位清零</b> 0: 无操作 1: 初始化完成标志位清零 注: 该位写1清除, 写0无效

### 20.5.2.9 TRNG状态事件寄存器 (TRNG\_ISR)

TRNG 中断状态寄存器 (TRNG_ISR)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											SERR	DAVLDC	STARTC		

Reserved	Bits 31-3	—	保留
SERR	Bit 2	R	<b>序列错误中断状态位</b> 0: 未发送中断 1: 已发送中断
DAVLDC	Bit 1	R	<b>数据有效中断状态位</b>



			0: 未发送中断 1: 已发送中断。
START	Bit 0	R	<b>初始化完成中断状态位</b> 0: 未发送中断 1: 已发送中断

## 第21章 运算加速器(CALC)

### 21.1 概述

运算加速器 (CALC) 可以执行平方根和除法的运算加速。

### 21.2 特性

- ◇ 支持最大 32 位无符号数平方根运算
- 支持最大 32 位有符号数或无符号数除法运算

### 21.3 结构框图

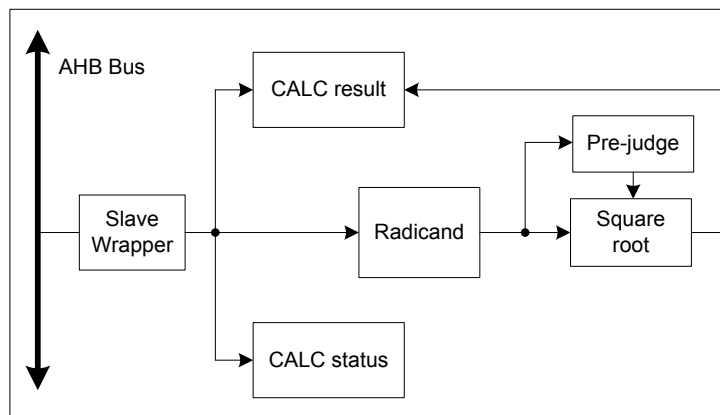


图 21-1 CALC 结构框图

### 21.4 平方根运算

#### 21.4.1 算法概述

无符号数平方根算法可对最大 32 位的无符号数进行平方根运算，运算结果最大为 16 位无符号数。若理论平方根值中含有小数部分，则硬件在计算时会舍去小数，即向 0 的方向取最大整数。

硬件运算电路中带有预判决功能，可根据被开方数 RCND 的数量级，自动选取最小的计算时间。

#### 21.4.2 使用说明

当被开方数寄存器 RDCND 发生写入动作时，平方根运算即开始，开始时平方根运算标志位 BUSY 被置位。当软件检测到 BUSY 被硬件清零时，表示平方根运算已经完成。通过读取平方根运算结果寄存器 SQRTRES 可获得被开方数的平方根近似值。

若当运算还未完成，被开方数寄存器 RDCND 发生写入动作时，硬件会重新开始新的运算，原先的运算结果将被丢弃。

为使得运算的结果更精确，在操作上可采取移位的方式来减小运算的误差。

例如，需要运算 X 的平方根，由于 X 较小，若直接写入 RDCND 中，产生的结果误差较大。可以先将 X 进行左移 n 位(n 需为偶数)，将 X 左移后可得新的被开方数即  $X' = X * 2^n$ ，将 X' 写入被开方数寄存器 RDCND，得到运算结果 Y'，可知  $Y' = \sqrt{X'} = \sqrt{X * 2^n} = 2^{n/2} * \sqrt{X}$ ，可将 SQRTRES 中的结果右移 n/2 位后，即得到 X 的平方根  $Y = \sqrt{X} = Y' / 2^{n/2}$ 。

原先 X 允许的位数 m 最大可至 32 位，当 X 的实际位数没有 32 位时，可适当的调制 n 的位数以最大程度的利用平方根运算器的性能。n 值越大，运算结果越精确。

以计算 2 的平方根为例。  $\sqrt{2} = 1.4142135623731$ 。

Radicand (Hex)	Radicand 格式	Result(Hex)	Result(Dec)	误差 (%)
0x0000 0002	m=32, n=0	0x0000 0001	1.0	-29.289
0x0000 0020	m=28, n=4	0x0000 0005	2.50	-11.612
0x0000 0200	m=24, n=8	0x0000 0016	1.3750	-2.773
0x0000 2000	m=20, n=12	0x0000 005A	1.406250	-0.563
0x0002 0000	m=16, n=16	0x0000 016A	1.41406250	-0.011
0x0020 0000	m=12, n=20	0x0000 05A8	1.41406250	-0.011
0x0200 0000	m=8, n=24	0x0000 16A0	1.41406250	-0.011
0x2000 0000	m=4, n=28	0x0000 5A82	1.4141845703	-0.002
0x8000 0000	m=2, n=30	0x0000 B504	1.4141845703	-0.002

表 21-1 平方根运算误差示例

### 21.4.3 完成时间

平方根算法可根据被开方数 RDCND 中输入数值的进行预判决，硬件针对不同量级自动判定运算时间，不同量级的运算时间可根据下表所示。

RDCND[31:0]	运算时间 (BUSY=1 的时钟个数)
1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	17
01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	16
001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	15
0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	14
0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	13
0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx	12
0000_0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx	10
0000_0000_0000_001x_xxxx_xxxx_xxxx_xxxx	9

RDCND[31:0]	运算时间 (BUSY=1 的时钟个数)
0000_0000_0000_0000_001x_xxxx_xxxx_xxxx 0000_0000_0000_0000_0001_xxxx_xxxx_xxxx	8
0000_0000_0000_0000_0000_1xxx_xxxx_xxxx 0000_0000_0000_0000_0000_01xx_xxxx_xxxx	7
0000_0000_0000_0000_0000_001x_xxxx_xxxx 0000_0000_0000_0000_0000_0001_xxxx_xxxx	6
0000_0000_0000_0000_0000_0000_1xxx_xxxx 0000_0000_0000_0000_0000_0000_01xx_xxxx	5
0000_0000_0000_0000_0000_0000_001x_xxxx 0000_0000_0000_0000_0000_0000_0001_xxxx	4
0000_0000_0000_0000_0000_0000_0000_1xxx 0000_0000_0000_0000_0000_0000_0000_01xx	3
0000_0000_0000_0000_0000_0000_0000_00xx	2

表 21-2 平方根运算时间表

## 21.5 除法运算

### 21.5.1 算法概述

除法算法可对最大 32 位的有符号数或无符号数进行除法运算，运算结果最大为 32 位有符号数或无符号数。有符号运算中 Bit31 位为符号位，负数使用二进制补码的方式表示。

商的符号由被除数和除数共同决定。当被除数和除数符号相同时，商为正数；当被除数和除数符号不同时，商为负数。余数的符号由被除数的符号决定。

硬件运算电路中带有预判决功能，可根据被除数的数量级，自动选取最小的计算时间。

### 21.5.2 特例说明

#### 溢出

在 32 位有符号除法运算中，当被除数为 0x8000\_0000，除数为 0xFFFF\_FFFF 时，即  $-2^{31}/-1$ ，得到的结果应为  $2^{31}$ ，但 32 位有符号数最大可表示的正整数为  $2^{31}-1$ ，该次运算结果将溢出。此时硬件计算所得的商为 0x8000\_0000，余数为 0x0000\_0000。硬件并无标识位指示运算结果是否为溢出。

#### 除数零

在 32 位有符号数或无符号数除法中，若除数设置为 0，则硬件计算所得的商为 0x0000\_0000，余数为 0x0000\_0000。该次计算硬件将标志位 DZ 置 1。

### 21.5.3 使用说明

通过设置 CALC\_DIVCSR.SIGN 位来选择运算的是有符号数还是无符号数。

通过设置 CALC\_DIVCSR.TRM 位来选择触发源。根据操作习惯，一般选择最后一个操作数的写入操作作为除法运算的触发源。

当被除数寄存器 DIVDR 和除数寄存器 DIVSR 写入值时，除法运算即触发开始，此时除法运算标志位 CALC\_DIVCSR.BUSY 被置位。当软件检测到 CALC\_DIVCSR.BUSY 被硬件清零时，表示除法运算已经完成。

通过读取商寄存器 DIVQR 和余数寄存器 DIVRR 可获得此次除法运算的结果。

若当运算还未完成，CALC\_DIVCSR.TRM 位所选择的触发动作发生时，硬件会重新开始的运算，原先的运算结果将被丢弃。

### 21.5.4 完成时间

除法算法可根据被除数 DIVDR 中输入数值的进行预判决，硬件针对被除数不同量级自动决定运算时间，不同量级的运算时间可根据下表所示。

被除数的绝对值 (abs(DIVD))	运算时间 (BUSY=1 的时钟个数)
1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	33
01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	32
001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	31
0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	30
0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	29
0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	28
0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	27
0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx	26
0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx	25
0000_0000_01xx_xxxx_xxxx_xxxx_xxxx_xxxx	24
0000_0000_001x_xxxx_xxxx_xxxx_xxxx_xxxx	23
0000_0000_0001_xxxx_xxxx_xxxx_xxxx_xxxx	22
0000_0000_0000_1xxx_xxxx_xxxx_xxxx_xxxx	21
0000_0000_0000_01xx_xxxx_xxxx_xxxx_xxxx	20
0000_0000_0000_001x_xxxx_xxxx_xxxx_xxxx	19
0000_0000_0000_0001_xxxx_xxxx_xxxx_xxxx	18
0000_0000_0000_0000_1xxx_xxxx_xxxx_xxxx	17
0000_0000_0000_0000_01xx_xxxx_xxxx_xxxx	16
0000_0000_0000_0000_001x_xxxx_xxxx_xxxx	15
0000_0000_0000_0000_0001_xxxx_xxxx_xxxx	14
0000_0000_0000_0000_0000_1xxx_xxxx_xxxx	13
0000_0000_0000_0000_0000_01xx_xxxx_xxxx	12
0000_0000_0000_0000_0000_001x_xxxx_xxxx	11
0000_0000_0000_0000_0000_0001_xxxx_xxxx	10
0000_0000_0000_0000_0000_0000_1xxx_xxxx	9
0000_0000_0000_0000_0000_0000_01xx_xxxx	8
0000_0000_0000_0000_0000_0000_001x_xxxx	7
0000_0000_0000_0000_0000_0000_0001_xxxx	6
0000_0000_0000_0000_0000_0000_0000_1xxx	5
0000_0000_0000_0000_0000_0000_0000_01xx	4
0000_0000_0000_0000_0000_0000_0000_001x	3
0000_0000_0000_0000_0000_0000_0000_000x	2

表 21-3 除法运算时间表

## 21.6 特殊功能寄存器

### 21.6.1 寄存器列表

CALC 寄存器列表			
名称	偏移地址	类型	描述
CALC 基地址: 4008_5400 <sub>H</sub>			
CALC_SQRTSR	000 <sub>H</sub>	R	平方根运算状态寄存器
CALC_RDCND	004 <sub>H</sub>	R/W	被开方数寄存器
CALC_SQRTRES	008 <sub>H</sub>	R	平方根运算结果寄存器
Reserved	00C <sub>H</sub> ~01C <sub>H</sub>	-	保留
CALC_DIVDR	020 <sub>H</sub>	R/W	被除数寄存器
CALC_DIVSR	024 <sub>H</sub>	R/W	除数寄存器
CALC_DIVQR	028 <sub>H</sub>	R	商寄存器
CALC_DIVRR	02C <sub>H</sub>	R	余数寄存器
CALC_DIVCSR	030 <sub>H</sub>	R/W	除法运算控制状态寄存器

## 21.6.2 寄存器描述

### 21.6.2.1 平方根运算状态寄存器 (CALC\_SQRTSR)

平方根运算状态寄存器 (CALC_SQRTSR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															BUSY

Reserved	Bits 31-1	—	保留
BUSY	Bit 0	R	平方根运算状态位 0: 完成 1: 进行中

### 21.6.2.2 被开方数寄存器 (CALC\_RDCND)

被开方数寄存器 (CALC_RDCND)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RADICAND																															

RADICAND	Bits 31-0	R/W	被开方数 32 位无符号被开方数
----------	-----------	-----	---------------------



### 21.6.2.3 平方根运算结果寄存器 (CALC\_SQRTRES)

平方根运算结果寄存器 (CALC_SQRTRES)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RESULT															

Reserved	Bits 31-16	—	保留
RESULT	Bits 15-0	R	平方根运算结果值 该位平方根运算结果

### 21.6.2.4 被除数寄存器 (CALC\_DIVDR)

被除数寄存器 (CALC_DIVDR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVD																															

DIVD	Bit 31-0	R/W	被除数 32 位被除数
------	----------	-----	----------------

### 21.6.2.5 除数寄存器 (CALC\_DIVSR)

除数寄存器 (CALC_DIVSR)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVS																															

DIVS	Bit 31-0	R/W	除数 32 位除数
------	----------	-----	--------------

### 21.6.2.6 商寄存器 (CALC\_DIVQR)

商寄存器 (CALC_DIVQR)																																
偏移地址: 28 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DIVQ																																

DIVQ	Bit 31-0	R	商 32 位商
------	----------	---	------------

### 21.6.2.7 余数寄存器 (CALC\_DIVRR)

余数寄存器 (CALC_DIVRR)																																
偏移地址: 2C <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DIVS																																

DIVS	Bit 31-0	R	余数 32 位余数
------	----------	---	--------------

### 21.6.2.8 除法运算控制状态寄存器 (CALC\_DIVCSR)

除法运算控制状态寄存器 (CALC_DIVCSR)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					TRM	SIGN	Reserved						DZ	BUSY	

Reserved	Bit 31-10	—	保留
TRM	Bit 9	R/W	除法运算触发模式选择 0: 写入被除数后触发 1: 写入除数后触发
SIGN	Bit 8	R/W	除法运算符号选择 0: 无符号数 1: 有符号数
Reserved	Bit 7-2	—	保留
DZ	Bit 1	R	除数为零状态位 0: 上一次操作除数不为0 1: 上一次操作除数为0
BUSY	Bit 0	R	除法运算状态位 0: 完成 1: 进行中

## 第22章 模数转换器(ADC)

### 22.1 概述

ADC 模块是 12 位精度的逐次逼近型模数转换器。它具有 10 个复用通道，可测量来自 7 个外部信号、两个内部信号(BT50 模拟前端 VTUNE 电压、温度传感器)和 VDET 电源电压检测信号。这些通道的 A/D 转换可在单次、连续、扫描或不连续 采样模式下进行。ADC 的结果存储在一个左对齐或右对齐的 16 位数据寄存器中。

ADC 模块具有模拟看门狗特性，允许应用检测输入电压是否超过了用户自定义的阈值上限或下限。

### 22.2 特性

- ◇ 可配置的转换精度 (6/8/10/12 位)
- ◇ 在标准转换、插入转换结束后以及发生模拟看门狗或溢出事件时产生中断
- ◇ 支持单次或连续转换模式
- ◇ 用于自动将通道 0 转换为通道 “n” 的扫描模式
- ◇ 可配置的数据对齐方式
- ◇ 可独立设置各通道采样时间
- ◇ 可配置外部触发器选项，可为标准转换和插入转换配置极性
- ◇ 支持不连续采样模式
- ◇ 可配置的参考源选择
- ◇ 可配置的转换时钟分频
- ◇ 标准通道转换期间可产生 DMA 请求

## 22.3 结构框图

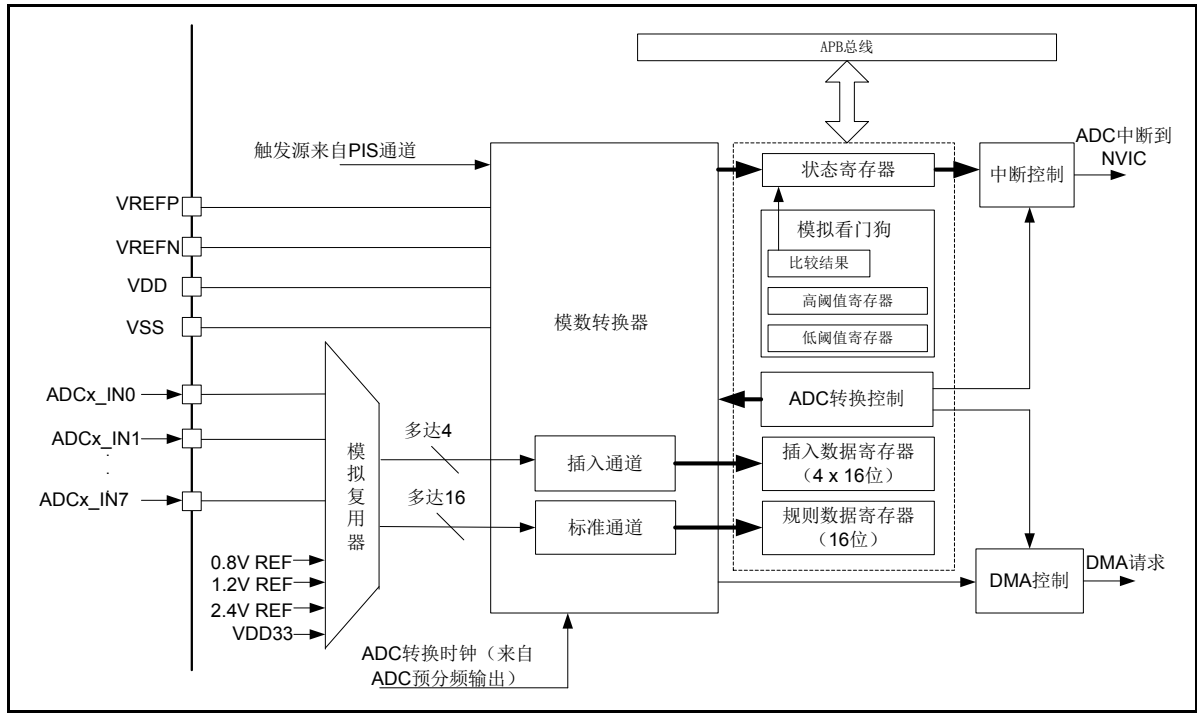


图 22-1 ADC 结构框图

## 22.4 功能描述

### 22.4.1 ADC控制

通过将 ADC\_CON1 寄存器中的 ADCEN 位置 1 来使能 ADC。

通过将 ADC\_CON1 寄存器中的 NCHTRG 或 ICHTRG 位置 1 来启动 AD 转换。

通过将 ADC\_CON1 寄存器中的 ADCEN 位清零来关闭 ADC。

### 22.4.2 ADC时钟

ADC 具有两个时钟方案：

- ◇ 用于模拟电路的时钟：ADCCLK

此时钟来自于经可编程预分频器分频的 APB 时钟，该预分频器可将 APB 时钟产生 2~256 分频时钟供 ADC 模拟电路使用。

- ◇ 用于数字接口的时钟（用于寄存器读/写访问）

此时钟为 APB 时钟。

### 22.4.3 通道选择

有 10 条复用通道。可以将转换分为两组：标准转换和插入转换。每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。例如，可按以下顺序对序列进行转换：ADC\_IN3、ADC\_IN7、ADC\_IN2、ADC\_IN2、ADC\_IN0、ADC\_IN2、ADC\_IN2、ADC\_IN8。

- ◇ 一个标准转换组最多由 16 个转换构成。必须在 ADC\_NCHSx 寄存器中选择转换序列的标准通道及其顺序。标准转换组中的转换总数必须写入 ADC\_CHSL 寄存器中的 NSL 位。
- ◇ 一个插入转换组最多由 4 个转换构成。必须在 ADC\_ICHS 寄存器中选择转换序列的插入通道及其顺序。插入转换组中的转换总数必须写入 ADC\_CHSL 寄存器中的 ISL 位。

如果在转换期间修改 ADC\_NCHSx 或 ADC\_ADC\_ICHS 寄存器，将复位当前转换并向 ADC 发送一个新的启动脉冲，以转换新选择的组。

温度传感器、BT50 模拟前端 VTUNE 电压和 VDET 电源检测内部通道：

- ◇ BT50 模拟前端 VTUNE 电压连接到通道 ADC\_IN7；
- ◇ 内部温度传感器连接到通道 ADC\_IN8；
- ◇ VDET 电源电压检测连接到通道 ADC\_IN9；

### 22.4.4 单次转换模式

单次工作模式是指 ADC 执行一次转换，其工作流程如下：

- ◇ 将 CON1.CM 位清 0，然后通过后续方式来启动此模式：
- ◇ 将 ADC\_CON1 寄存器中的 NCHTRG 位置 1（仅适用于标准通道）
- ◇ 将 ADC\_CON1 寄存器中的 ICHTRG 位置 1（仅适用于插入通道）
- ◇ 外部触发（适用于标准通道或插入通道）

完成所选通道的转换之后：

- ◇ 使用标准通道：
  - 转换数据存储在 16 位 ADC\_NCHDR 寄存器中
  - 标准转换结束标志 ADC\_STAT.NCHE 置 1
  - 若标准转换完成中断使能位 ADC\_CON0.NCHEIE 置 1，将产生中断
- ◇ 使用插入通道：
  - 转换数据存储在 16 位 ADC\_ICHDR 寄存器中
  - 插入转换结束标志 ADC\_STAT.ICHE 置 1
  - 若插入转换完成中断使能位 ADC\_CON0.ICHEIE 置 1，将产生中断

然后，ADC 停止。

### 22.4.5 连续转换模式

连续工作模式是指 ADC 执行一次转换任务后，立即执行下一个转换任务，其工作流程如下：

- ◇ 将 CON1.CM 位置 1
  - ◇ 通过外部触发或 ADC\_CON1.NCHTRG 置 1，来启动此模式（仅适用于标准通道）
- 每次转换之后：

- ◇ 如果转换了标准通道组：
  - 上次转换的数据存储在 16 位 ADC\_NCHDR 寄存器中
  - 标准转换结束标志 ADC\_STAT.NCHE 置 1
  - 若标准转换完成中断使能位 ADC\_CON0.NCHEIE 置 1，将产生中断

注 1：连续转换模式不适用于插入组通道。  
注 2：连续模式下唯一的例外情况是，在使能 ADC\_CON0.IAUTO 时，插入通道组在标准通道之后自动转换。

### 22.4.6 时序图

在使能 ADC 后，需要一段稳定时间  $t_{STAB}$ ，然后再启动 ADC 转换，并经过 15 个时钟周期后 CHE 标志置 1，如下图所示。

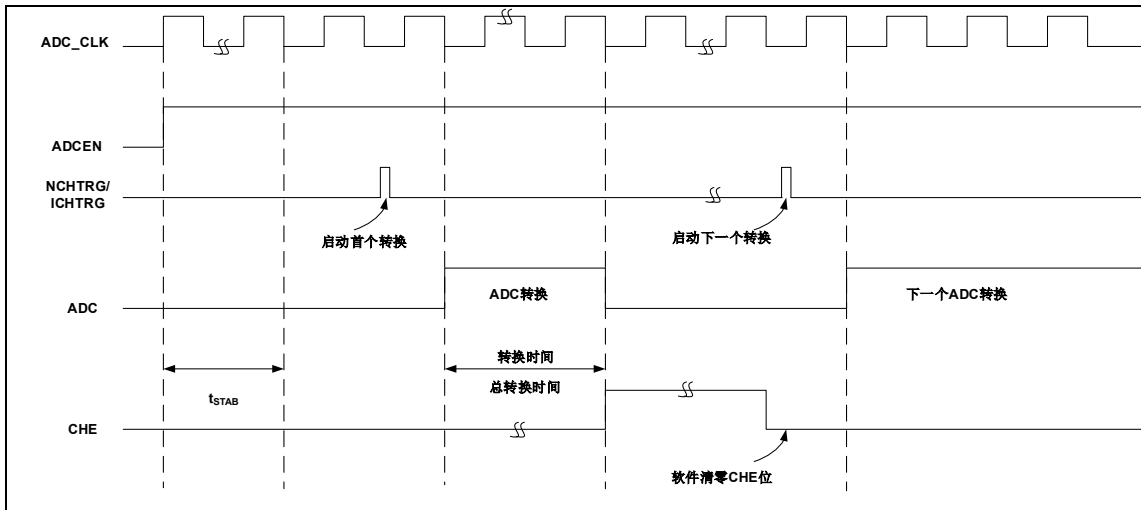


图 22-2 ADC 转换时序图

### 22.4.7 模拟看门狗

如果 ADC 转换的模拟电压低于阈值下限（寄存器 ADC\_WDTL）或高于阈值上限（寄存器 ADC\_HTR），则 AWD 模拟看门狗状态位 ADC\_STAT.AWDF 位置 1。使能 ADC\_CON0.AWDIE 位可以打开模拟看门狗中断。

上下限阈值寄存器是低 12 位有效的寄存器，而 ADC 结果寄存器可以左对齐和右对齐，但看门狗是在对齐之前将模拟电压与阈值上限和下限进行比较。

下表介绍了应如何配置 ADC\_CON0 寄存器才能在一个或多个通道上使能模拟看门狗。

模拟看门狗保护通道	AWDSGL	NCHWDEN	ICHWDTEN
无	X	0	0
所有插入通道	0	0	1
所有标准通道	0	1	0

模拟看门狗保护通道	AWDSGL	NCHWDEN	ICHWDTEN
所有标准通道和插入通道	0	1	1
单个插入通道(AWDCH 选择)	1	0	1
单个标准通道(AWDCH 选择)	1	1	0
单个标准/插入通道(AWDCH 选择)	1	1	1

表 22-1 模拟看门狗通道选择

### 使用模拟看门狗流程

1. 设置 ADC\_CON0.AWDSGL 位来选择看门狗作用于单个通道或一组通道，清 0 选择一组通道，置 1 选择单个通道。选择单个通道通过 AWDCH 选择哪个通道。
2. 置位 ADC\_CON0.NCHWDEN 或 ADC\_CON0.ICHWDTEN 选择使能标准组或插入组通道。

## 22.4.8 通道扫描

扫描模式是指连续依次转换标准组所有通道或者插入组所有通道。通过将 ADC\_CON0.SCANEN 位置 1 来选择扫描模式。ADC 会扫描在 ADC\_NCHSx 寄存器（对于标准通道）或 ADC\_ICHS 寄存器（对于插入通道）中选择的所有通道，为组中的每个通道依次执行一次转换。如果将 CM 位置 1，并且是标准组扫描，ADC 会循环转换标准组所有通道。

如果将 ADC\_CON1.DMA 位置 1，则在每次通道转换结束标志置位后通过 DMA 控制器将转换自标准通道组的数据寄存器（ADC\_NCHDR）传输到 SRAM。

在以下情况下，ADC\_STAT 寄存器中的 NCHE 位置 1：

- ◇ 如果 CON1.NCHESEL 位清零，在每个标准组序列转换结束时
- ◇ 如果 CON1.NCHESEL 位置 1，在每个标准通道转换结束时

从插入通道转换的数据始终存储在 ADC\_ICHDRx 寄存器中。

## 22.4.9 插入通道控制

### 触发插入

要使用触发插入，必须将 ADC\_CON0.IAUTO 位清零。使用触发插入时，必须确保触发事件之间的间隔长于插入序列。

在插入转换期间出现标准通道转换事件，插入转换不会被中断，标准转换在插入转换结束时执行。

在标准组转换期间触发插入：

1. 通过外部触发或将 ADC\_CON1.NCHTRG 位置 1 来启动标准通道组转换。
2. 如果在标准通道组转换期间出现外部插入触发或者 ADC\_CON1.ICHTRG 位置 1，则当前的转换会打断，并且插入通道序列会切换为单次扫描模式。
3. 然后，标准通道组的标准转换会从上上次中断的标准转换处恢复。

在插入转换期间触发插入



### 自动插入

如果将 IAUTO 位置 1, 则插入组中的通道会在标准组通道之后自动转换。这可用于转换最多由 20 个 AD 转换构成的序列, 这些转换在 ADC\_NCHSx 和 ADC\_ICHS 寄存器中编程。

在此模式下, 必须禁止插入通道上的外部触发。

如果 CM 位和 IAUTO 位均已置 1, 则在转换标准通道之后会继续转换插入通道。

## 22.4.10 不连续采样控制

### 标准组

可将 ADC\_CON0.NCHDCEN 位置 1 来使能此模式。该模式可用于转换含有  $n$  ( $n \leq 8$ ) 个转换的短序列, 该短序列是在 ADC\_NCHSx 寄存器中选择的转换序列的一部分。可通过写入 ADC\_CON0.ETRGN 位来指定  $n$  的值。

出现外部触发时, 将启动在 ADC\_NCHSx 寄存器中选择的  $n$  个转换, 直到序列中的所有转换均完成为止。通过 ADC\_CHSL.NSL 位定义总序列长度。

示例:

$n = 4$ , ADC\_CHSL.NSL=9, 要转换的通道 = 0、2、3、5、6、7、8、10、11

第 1 次触发: 转换序列 0、2、3、5

第 2 次触发: 转换序列 6、7、8、10

第 3 次触发: 转换序列 11 并生成 NCHE 事件

第 4 次触发: 转换序列 0、2、3、5

在不连续采样模式下转换标准组时, 最后一次触发剩余的通道不足 ADC\_CON0.ETRGN 指定的通道数时, 不会接着从头转换, 转换到序列最后一个通道就会停止, 生成 NCHE 时间。转换完所有序列通道后, 下一个触发信号将启动序列的第一个通道。在上述示例中, 第 4 次触发 重新转换了序列的 0、2、3、5 通道。

### 插入组

可将 ADC\_CON0 寄存器中的 ICHDCEN 位置 1 来使能此模式。在出现外部触发事件之后, 可使用该模式逐通道 ( $n=1$ ) 转换在 ADC\_ICHS 寄存器中选择的序列。

出现外部触发时, 将启动在 ADC\_ICHS 寄存器中选择的下一个通道转换, 直到序列中的所有转换均完成为止。通过 ADC\_CHSL 寄存器中的 ISL 位定义总序列长度。

示例:

$n = 1$ , ADC\_CHSL.ISL=3, 要转换的通道 = 1、3、4

第 1 次触发: 转换通道 1

第 2 次触发: 转换通道 3

第 3 次触发: 转换通道 4 并生成 CHE 和 ICHE 事件

第 4 次触发: 通道 1

转换完所有插入通道后，下一个触发信号将启动第一个插入通道的转换。在上述示例中，第 4 次触发重新转换了第 1 个插入通道。

不能同时使用自动插入和不连续采样模式。

不得同时为标准组和插入组设置不连续采样模式。只能选择其一进行不连续采样模式。

### 22.4.11 数据对齐

ADC\_CON1.ALIGN 位用于选择转换后存储的数据的对齐方式,可选择左对齐和右对齐两种方式，如下图所示。

插入通道组的转换数据将加上 ADC\_ICHOFFx 寄存器中写入的用户自定义偏移量,因此结果可以是一个负值,图中 EXS 位表示扩展的符号值。

对于标准组中的通道，不会减去任何偏移量，因此只有十二个位有效。

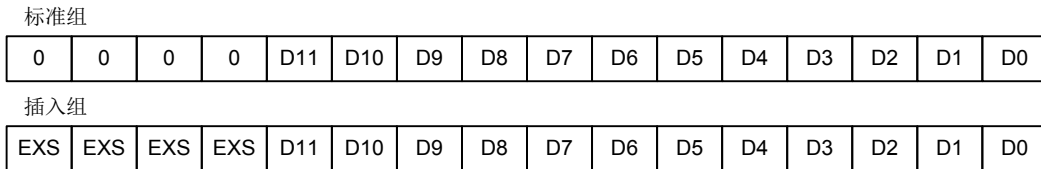


图 22-3 右对齐数据示意图

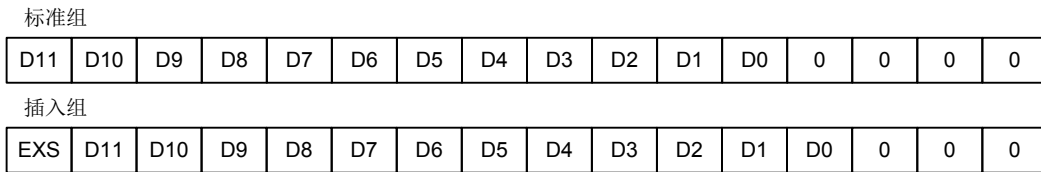


图 22-4 左对齐数据示意图

### 22.4.12 可独立设置各通道采样时间

ADC 会在数个 ADCCLK 周期内对输入电压进行采样，可使用 ADC\_SMPT1 和 ADC\_SMPT2 寄存器中的 CHTx 位修改周期数。每个通道均可以使用不同的采样时间进行采样。

总转换时间的计算公式如下：

$$T_{conv} = \text{采样时间} + 14 \text{ 个周期}$$

示例：

ADCCLK = 24MHz 且采样时间 = 3 个周期时：

$$T_{conv} = 3 + 14 = 17 \text{ 个周期} = 0.708 \mu s \text{ (APB 为 48MHz 时)}$$

### 22.4.13 外部触发转换和触发极性

可以通过配置 PIS 通道选择相应信号触发 ADC 转换，PIS 的配置请查看 PIS 相应章节描述。当 PIS 某一通道配置成 ADC 外部触发时，若相应的触发源事件发生，将自动触发标准序列或插入序列转换。

### 22.4.14 快速转换模式（不支持）

可通过降低 ADC 分辨率来执行快速转换。RSEL 位用于选择数据寄存器中可用的位数。每种分辨率的最小转换时间如下：

- ◇ 12 位：3 + 12 = 15 ADCCLK 周期
- ◇ 10 位：3 + 10 = 13 ADCCLK 周期
- ◇ 8 位：3 + 8 = 11 ADCCLK 周期
- ◇ 6 位：3 + 6 = 9 ADCCLK 周期

### 22.4.15 软件补偿

由于 ADC 转换输出的 A/D 码和理论值偏差比较大，需要采用软件补偿的方式对 ADC 输出的码值进行软件补偿，软件补偿参数 Gain 和 Offset 的存放位置如下表所示：

寄存器名称	访问地址	参数	说明
CFG_ADCI1P2	0x0008_00F0	Gain	内部 1.20v 参考电压的 Gain 和 Offset 参数
	0x0008_00F2	Offset	
CFG_ADCI2P4	0x0008_0100	Gain	内部 2.40v 参考电压的 Gain 和 Offset 参数
	0x0008_0102	Offset	
CFG_ADCV3P3	0x0008_0110	Gain	电源电压 3.30v 参考电压的 Gain 和 Offset 参数
	0x0008_0112	Offset	
CFG_ADCE3P3	0x0008_0120	Gain	外部 3.30v 参考电压的 Gain 和 Offset 参数
	0x0008_0122	Offset	

表 22-2 ADC 软件补偿参数

注 1：其中 Gain 为 16 位的无符号数；Offset 为 16 位的有符号数；

软件补偿的计算公式如下：

$$ADC_{RES} = (C - Offset) \times \frac{Gain}{32768.0} \quad (\text{公式 1})$$

其中“C”表示 ADC 模块输出的 ADC 码。

## 22.5 数据管理

### 22.5.1 使用 DMA

标准组只有一个数据寄存器（ADC\_NCHDR）用于存储 AD 转换结果值，所以，对于多个标准通道的转换，使用 DMA 可以快速存储数据，避免在上一次转换结果的值还未读出时新的 ADC 结果值又写入 ADC\_CHDR 寄存器，造成数据丢失。

每完成标准通道组中的一个通道转换后，都会生成一个 DMA 请求。这样便可将转换的数据从 ADC\_NCHDR 寄存器传输到软件指定的目标内存位置。

### 22.5.2 在不使用 DMA 的情况下管理转换序列

如果转换过程足够慢，则可使用软件来处理转换序列。在这种情况下，必须将 ADC\_CON1

寄存器中的 NCHESEL 位置 1，才能使 NCHE 状态位在每次转换结束时置 1，而不仅是在序列结束时置 1。当 NCHESEL = 1 时，会自动使能溢出检测。因此，每当转换结束时，NCHE 都会置 1，并且可以读取 ADC\_NCHDR 寄存器。如果数据丢失（溢出），则会将 ADC\_STAT 寄存器中的 OVR 位置 1 并生成一个中断（如果 ADC\_CON0.OVRIE 位已置 1）。

要在 NCHESEL 位置 1 时将 ADC 从 OVR 状态中恢复，请按以下步骤操作：

1. 将 ADC\_STAT 寄存器中的 OVR 位清零
2. 触发 ADC 以开始转换

### 22.5.3 在不使用DMA和溢出检测情况下进行转换

当 ADC 存在转换一个或多个通道时不需要每次读取数据的情况时，例如使用模拟看门狗，可将 ADC\_CON1.OVRDIS 置 1，并且仅在序列结束 (NCHESEL = 0) 时才将 NCHE 位置 1。这样溢出检测被禁止。

## 22.6 ADC中断

当模拟看门狗状态位和溢出状态位分别置 1 时，标准组和插入组在转换结束时可能会产生中断。可以使用单独的中断使能位以实现灵活性。

中断事件	事件标志位	使能控制位
结束标准组的转换	NCHE	NCHEIE
结束插入组的转换	ICHE	ICHEIE
发生模拟看门狗事件	AWDF	AWDIE
溢出	OVR	OVRIE

表 22-3 ADC 中断

## 22.7 特殊功能寄存器

### 22.7.1 寄存器列表

ADC 寄存器列表			
名称	偏移地址	类型	描述
ADC 基地址: 4004_2000 <sub>H</sub>			
ADC_STAT	000 <sub>H</sub>	R	ADC 状态寄存器
ADC_CLR	004 <sub>H</sub>	W	ADC 清零寄存器
ADC_CON0	008 <sub>H</sub>	R/W	ADC 控制寄存器 0
ADC_CON1	00C <sub>H</sub>	R/W	ADC 控制寄存器 1
ADC_SMPT1	010 <sub>H</sub>	R/W	ADC 采样时间寄存器 1
ADC_SMPT2	014 <sub>H</sub>	R/W	ADC 采样时间寄存器 2
ADC_SMPT3	018 <sub>H</sub>	R/W	ADC 采样时间寄存器 3
Reserved	01C <sub>H</sub>	-	保留
ADC_ICHOFF1	020 <sub>H</sub>	R/W	ADC 插入通道数据偏移寄存器 1
ADC_ICHOFF2	024 <sub>H</sub>	R/W	ADC 插入通道数据偏移寄存器 2
ADC_ICHOFF3	028 <sub>H</sub>	R/W	ADC 插入通道数据偏移寄存器 3
ADC_ICHOFF4	02C <sub>H</sub>	R/W	ADC 插入通道数据偏移寄存器 4
ADC_NCHS1	030 <sub>H</sub>	R/W	ADC 标准通道序列寄存器 1
ADC_NCHS2	034 <sub>H</sub>	R/W	ADC 标准通道序列寄存器 2
ADC_NCHS3	038 <sub>H</sub>	R/W	ADC 标准通道序列寄存器 3
ADC_NCHS4	03C <sub>H</sub>	R/W	ADC 标准通道序列寄存器 3
ADC_ICHS	040 <sub>H</sub>	R/W	ADC 插入通道序列寄存器
ADC_CHSL	044 <sub>H</sub>	R/W	ADC 通道序列长度寄存器
ADC_WDTH	048 <sub>H</sub>	R/W	ADC 看门狗高阈值寄存器
ADC_WDTL	04C <sub>H</sub>	R/W	ADC 看门狗低阈值寄存器
ADC_ICHDR1	050 <sub>H</sub>	R	ADC 插入通道数据寄存器 1
ADC_ICHDR2	054 <sub>H</sub>	R	ADC 插入通道数据寄存器 2
ADC_ICHDR3	058 <sub>H</sub>	R	ADC 插入通道数据寄存器 3
ADC_ICHDR4	05C <sub>H</sub>	R	ADC 插入通道数据寄存器 4
ADC_NCHDR	060 <sub>H</sub>	R	ADC 标准通道数据寄存器
ADC_CCR1	064 <sub>H</sub>	R/W	ADC 通用控制寄存器 1
ADC_CCR2	068 <sub>H</sub>	R/W	ADC 通用控制寄存器 2
-	070 <sub>H</sub>	-	-
-	074 <sub>H</sub>	-	-

## 22.7.2 寄存器描述

### 22.7.2.1 ADC状态寄存器 (ADC\_STAT)

ADC 状态寄存器 (ADC_STAT)																																			
偏移地址: 00 <sub>H</sub>																																			
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																						ICHS		NCHS		Reserved						OVR	ICHE	NCHE	AWDF

Reserved	Bit 31-10	—	保留
ICHS	Bit 9	R	<b>插入通道转换开始标志位</b> 0: 未开始插入转换或标志位已被清除 1: 已开始插入转换 注: 该位由硬件置1, 通过操作ADC_CLR清零
NCHS	Bit 8	R	<b>标准通道转换开始标志位</b> 0: 未开始标准转换或标志位已被清除 1: 已开始标准转换 注: 该位由硬件置1, 通过操作ADC_CLR清零
Reserved	Bit 7-4	—	保留
OVR	Bit 3	R	<b>转换溢出标志位</b> 0: 未发生溢出或标志位已被清除 1: 发生溢出 注 1: 溢出检测仅在 DMA=1 或 NCHESEL=1 时使能 注 2: 该位由硬件置 1, 通过操作 ADC_CLR 清零
ICHE	Bit 2	R	<b>插入通道转换结束标志位</b> 0: 所有插入转换未完成或标志位已被清除 1: 所有插入转换已完成 注: 该位由硬件置 1, 通过操作 ADC_CLR 清零
NCHE	Bit 1	R	<b>标准通道转换结束标志位</b> NCHESEL = 0时 0: 标准转换序列未完成或标志位已被清除 1: 标准转换序列已完成 NCHESEL = 1 时 0: 单次标准转换未完成或标志位已被清除 1: 单次标准转换已完成 注: 该位由硬件置1, 通过操作ADC_CLR清零
AWDF	Bit 0	R	<b>模拟看门狗标志位</b> 0: 未发生看门狗事件或标志位已被清除 1: 已发生看门狗事件

			注：该位由硬件置1，通过操作ADC_CLR清零
--	--	--	-------------------------

### 22.7.2.2 ADC清零寄存器 (ADC\_CLR)

ADC 清零寄存器 (ADC_CLR)																																				
偏移地址: 04 <sub>H</sub>																																				
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Reserved																						ICHS		NCHS		Reserved							OVR	ICHE	NCHE	AWDF

Reserved	Bit 31-10	—	保留
ICHS	Bit 9	W1	插入通道开始标志位清零 0: 无操作 1: 插入转换开始标志位清零
NCHS	Bit 8	W1	标准通道开始标志位清零 0: 无操作 1: 标准转换开始标志位清零
Reserved	Bit 7-4	—	保留
OVR	Bit 3	W1	转换溢出标志位清零 0: 无操作 1: 转换溢出标志位清零
ICHE	Bit 2	W1	插入转换结束标志位清零 0: 无操作 1: 插入转换结束标志位清零
NCHE	Bit 1	W1	标准转换结束标志位清零 0: 无操作 1: 标准转换结束标志位清零
AWDF	Bit 0	W1	模拟看门狗标志位清零 0: 无操作 1: 模拟看门狗标志位清零

### 22.7.2.3 ADC控制寄存器0 (ADC\_CON0)

ADC 控制寄存器 0 (ADC_CON0)																																
偏移地址: 08 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved					OVRIE	RSEL			NCHWDEN	ICHWDTEN	Reserved			CONWAIT			ETRGN			ICHDCEN	NCHDCEN	IAUTO	AWDSGL	SCANEN	ICHEIE	AWDIE	NCHIEIE	AWDCH				

Reserved	Bit 31-27	—	保留
OVRIE	Bit 26	R/W	溢出中断使能位 0: 禁止 1: 使能
RSEL	Bit 25-24	R/W	ADC 转换精度选择位 00: 6 位 01: 8 位 10: 10 位 11: 12 位
NCHWDEN	Bit 23	R/W	标准通道看门狗使能位 0: 禁止 1: 使能
ICHWDTEN	Bit 22	R/W	插入通道看门狗使能位 0: 禁止 1: 使能
Reserved	Bit 21-19	—	保留
CONWAIT	Bits 18-16	R/W	连续转换模式通道间隔采样等待时间选择位 000: 无等待 001: 等待1个ADCCLK 010: 等待2个ADCCLK ..... 111: 等待7个ADCCLK 注: 通道采样时间寄存器ADC_SMPT*使用时, 可不配置此控制位, 作用同ADC_SMPT*
ETRGN	Bit 15-13	R/W	外部触发不连续转换通道数 000: 1 个通道 001: 2 个通道 ..... 111: 8 个通道
ICHDCEN	Bit 12	R/W	插入通道不连续转换使能位 0: 禁止 1: 使能



NCHDCEN	Bit 11	R/W	标准通道不连续转换使能位 0: 禁止 1: 使能
IAUTO	Bit 10	R/W	插入组自动转换使能位 0: 禁止 1: 使能
AWDSGL	Bit 9	R/W	扫描模式单一通道模拟看门狗使能位 0: 禁止 1: 使能
SCANEN	Bit 8	R/W	扫描模式使能位 0: 禁止 1: 使能
ICHEIE	Bit 7	R/W	插入通道转换完成中断使能位 0: 禁止 1: 使能
AWDIE	Bit 6	R/W	模拟看门狗中断使能位 0: 禁止 1: 使能
NCHEIE	Bit 5	R/W	标准通道转换完成中断使能位 0: 禁止 1: 使能
AWDCH	Bit 4-0	R/W	模拟看门狗通道选择位 00000: ADC 输入通道 0 00001: ADC 输入通道 1 ..... 10010: ADC 输入通道 8 其他: 保留

#### 22.7.2.4 ADC控制寄存器 1 (ADC\_CON1)

ADC 控制寄存器 1 (ADC_CON1)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NCHTRG	NETS		Reserved				ICHTRG	IETS	Reserved											ALIGN	NCHESEL	Reserved	DMA	Reserved				CM	ADCEEN	

Reserved	Bit 31	—	保留
NCHTRG	Bit 30	W1	标准通道触发位 0: 无操作 1: 触发开始标准通道转换

NETS	Bit 29-28	R/W	标准转换外部触发极性选择位 00: 外部触发禁止 01: 上升沿触发 10: 下降沿触发 11: 上升沿和下降沿触发
Reserved	Bit 27-23	—	保留
ICHTRG	Bit 22	W1	插入通道触发位 0: 无操作 1: 触发开始插入通道转换
IETS	Bit 21-20	R/W	插入转换外部触发极性选择位 00: 外部触发禁止 01: 上升沿触发 10: 下降沿触发 11: 上升沿和下降沿触发
Reserved	Bit 19-12	—	保留
ALIGN	Bit 11	R/W	数据对齐方式位 0: 右对齐 1: 左对齐
NCHSEL	Bit 10	R/W	标准转换结束标志选择位 0: 每个标准转换序列结束时将 STAT.RCHE 位置 1 1: 每个标准转换结束时将 STAT.RCHE 位置 1
Reserved	Bit 9	—	保留
DMA	Bit 8	R/W	DMA 访问使能位 0: 禁止 1: 使能
Reserved	Bit 7-2	—	保留
CM	Bit 1	R/W	转换模式 0: 单次转换 1: 连续转换
ADCEN	Bit 0	R/W	ADC 使能位 0: 禁止 1: 使能

### 22.7.2.5 ADC采样时间寄存器 1 (ADC\_SMPT1)

ADC 采样时间寄存器 1 (ADC_SMPT1)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT7				CHT6				CHT5				CHT4				CHT3				CHT2				CHT1				CHT0			

CHT<y>	Bit 31-0	R/W	<b>通道 y 采样时间选择位 (y=0..7)</b> 0000: 1 个周期 0001: 2 个周期 0010: 4 个周期 0011: 15 个周期 其他: 预留
--------	----------	-----	---

### 22.7.2.6 ADC采样时间寄存器 2 (ADC\_SMPT2)

ADC 采样时间寄存器 2 (ADC_SMPT2)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHT15				CHT14				CHT13				CHT12				CHT11				CHT10				CHT8				CHT8			

CHT<y>	Bit 31-0	R/W	<b>通道 y 采样时间选择位 (y=8..15)</b> 0000: 1 个周期 0001: 2 个周期 0010: 4 个周期 0011: 15 个周期 其他: 预留
--------	----------	-----	--

### 22.7.2.7 ADC采样时间寄存器 3 (ADC\_SMPT3)

ADC 采样时间寄存器 3 (ADC_SMPT3)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CHT19		CHT18		CHT17		CHT16									

SMP<y>	Bits 31-0	R/W	<b>通道 y 采样时间选择位 (y=16..19)</b> 0000: 1 个周期 0001: 2 个周期 0010: 4 个周期 0011: 15 个周期 其他: 预留
--------	-----------	-----	---

### 22.7.2.8 ADC插入通道数据偏移寄存器 1 (ADC\_ICHOFF1)

ADC 插入通道数据偏移寄存器 1 (ADC_ICHOFF1)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																IOFF1															

Reserved	Bit 31-12	—	保留
IOFF1	Bit 11-0	R/W	<b>插入通道 1 数据偏移量</b> ADC_ICHDR1 中的数据为原始转换数据加上偏移量

### 22.7.2.9 ADC插入通道数据偏移寄存器 2 (ADC\_ICHOFF2)

ADC 插入通道数据偏移寄存器 2 (ADC_ICHOFF2)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											IOFF2																				

Reserved	Bit 31-12	—	保留
IOFF2	Bit 11-0	R/W	插入通道 2 数据偏移量 ADC_ICHDR2 中的数据为原始转换数据加上偏移量

### 22.7.2.10 ADC插入通道数据偏移寄存器 3 (ADC\_ICHOFF3)

ADC 插入通道数据偏移寄存器 3 (ADC_ICHOFF3)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											IOFF3																				

Reserved	Bit 31-12	—	保留
IOFF3	Bit 11-0	R/W	插入通道 3 数据偏移量 ADC_ICHDR3 中的数据为原始转换数据加上偏移量

### 22.7.2.11 ADC插入通道数据偏移寄存器 4 (ADC\_ICHOFF4)

ADC 插入通道数据偏移寄存器 4 (ADC_ICHOFF4)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											IOFF4																				

Reserved	Bit 31-12	—	保留
IOFF4	Bit 11-0	R/W	插入通道 4 数据偏移量 ADC_ICHDR4 中的数据为原始转换数据加上偏移量

### 22.7.2.12 ADC标准通道序列寄存器 1 (ADC\_NCHS1)

ADC 标准通道序列寄存器 1 (ADC_NCHS1)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				NS4				Reserved				NS3				Reserved				NS2				Reserved				NS1			

Reserved	Bit 31-29	—	保留
NS4	Bit 28-24	R/W	标准序列第 4 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 23-21	—	保留
NS3	Bit 20-16	R/W	标准序列第 3 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 15-13	—	保留
NS2	Bit 12-8	R/W	标准序列第 2 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 7-5	—	保留
NS1	Bit 4-0	R/W	标准序列第 1 次转换通道编号 00000~10010: 通道 0~18 其他: 预留

### 22.7.2.13 ADC标准通道序列寄存器 2 (ADC\_NCHS2)

ADC 标准通道序列寄存器 2 (ADC_NCHS2)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				NS8				Reserved				NS7				Reserved				NS6				Reserved				NS5			

Reserved	Bit 31-29	—	保留
NS8	Bit 28-24	R/W	标准序列第 8 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 23-21	—	保留
NS7	Bit 20-16	R/W	标准序列第 7 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 15-13	—	保留
NS6	Bit 12-8	R/W	标准序列第 6 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 7-5	—	保留
NS5	Bit 4-0	R/W	标准序列第 5 次转换通道编号 00000~10010: 通道 0~18 其他: 预留

### 22.7.2.14 ADC标准通道序列寄存器 3 (ADC\_NCHS3)

ADC 标准通道序列寄存器 3 (ADC_NCHS3)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				NS12				Reserved				NS11				Reserved				NS10				Reserved				NS9			

Reserved	Bit 31-29	—	保留
NS12	Bit 28-24	R/W	标准序列第 12 次转换通道编号 00000~10010: 通道 0~18

			其他：预留
Reserved	Bit 23-21	—	保留
NS11	Bit 20-16	R/W	标准序列第 11 次转换通道编号 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	保留
NS10	Bit 12-8	R/W	标准序列第 10 次转换通道编号 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	保留
NS9	Bit 4-0	R/W	标准序列第 9 次转换通道编号 00000~10010：通道 0~18 其他：预留

### 22. 7. 2. 15 ADC标准通道序列寄存器 4 (ADC\_NCHS4)

ADC 标准通道序列寄存器 4 (ADC_NCHS4)																															
偏移地址：3C <sub>H</sub>																															
复位值：00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS16				Reserved			NS15				Reserved			NS14				Reserved			NS3							

Reserved	Bit 31-29	—	保留
NS16	Bit 28-24	R/W	标准序列第 16 次转换通道编号 00000~10010：通道 0~18 其他：预留
Reserved	Bit 23-21	—	保留
NS15	Bit 20-16	R/W	标准序列第 15 次转换通道编号 00000~10010：通道 0~18 其他：预留
Reserved	Bit 15-13	—	保留
NS14	Bit 12-8	R/W	标准序列第 14 次转换通道编号 00000~10010：通道 0~18 其他：预留
Reserved	Bit 7-5	—	保留
NS13	Bit 4-0	R/W	标准序列第 13 次转换通道编号 00000~10010：通道 0~18 其他：预留



### 22.7.2.16 ADC插入通道序列寄存器 (ADC\_ICHS)

ADC 插入通道序列寄存器 (ADC_ICHS)																															
偏移地址: 40 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IS4				Reserved				IS3				Reserved				IS2				Reserved				IS1			

Reserved	Bit 31-29	—	保留
IS4	Bit 28-24	R/W	插入序列第 4 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 23-21	—	保留
IS3	Bit 20-16	R/W	插入序列第 3 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 15-13	—	保留
IS2	Bit 12-8	R/W	插入序列第 2 次转换通道编号 00000~10010: 通道 0~18 其他: 预留
Reserved	Bit 7-5	—	保留
IS1	Bit 4-0	R/W	插入序列第 1 次转换通道编号 00000~10010: 通道 0~18 其他: 预留

### 22.7.2.17 ADC通道序列长度寄存器 (ADC\_CHSL)

ADC 通道序列长度寄存器 (ADC_CHSL)																															
偏移地址: 44 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				ISL		Reserved				NSL					

Reserved	Bit 31-10	—	保留
ISL	Bit 9-8	R/W	插入通道序列长度 00: 1 次转换

			01: 2 次转换 10: 3 次转换 11: 4 次转换
Reserved	Bit 7-4	—	保留
NSL	Bit 3-0	R/W	标准通道列长度 0000: 1 次转换 0001: 2 次转换 ..... 1111: 16 次转换

### 22.7.2.18 ADC看门狗高阈值寄存器 (ADC\_WDTH)

ADC 看门狗高阈值寄存器 (ADC_WDTH)																															
偏移地址: 48 <sub>H</sub>																															
复位值: 00000000_00000000_00001111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																H															

Reserved	Bit 31-12	—	保留
HT	Bit 11-0	R/W	模拟看门狗高阈值 当原始数据大于高阈值时产生看门狗事件

### 22.7.2.19 ADC看门狗低阈值寄存器 (ADC\_WDTL)

ADC 看门狗低阈值寄存器 (ADC_WDTL)																															
偏移地址: 4C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																H															

Reserved	Bit 31-12	—	保留
LT	Bit 11-0	R/W	模拟看门狗低阈值 当原始数据小于低阈值时产生看门狗事件

### 22.7.2.20 ADC插入通道数据寄存器 1 (ADC\_ICHDR1)

ADC 插入通道数据寄存器 1 (ADC_ICHDR1)																															
偏移地址: 50 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	插入通道 1 转换数据 该数据为对齐之后的数据

### 22.7.2.21 ADC插入通道数据寄存器 2 (ADC\_ICHDR2)

ADC 插入通道数据寄存器 2 (ADC_ICHDR2)																															
偏移地址: 54 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	插入通道 2 转换数据 该数据为对齐之后的数据

### 22.7.2.22 ADC插入通道数据寄存器 3 (ADC\_ICHDR3)

ADC 插入通道数据寄存器 3 (ADC_ICHDR3)																															
偏移地址: 58 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
----------	-----------	---	----

VAL	Bit 15-0	R	插入通道 3 转换数据 该数据为对齐之后的数据
-----	----------	---	----------------------------

### 22.7.2.23 ADC插入通道数据寄存器 4 (ADC\_ICHDR4)

ADC 插入通道数据寄存器 4 (ADC_ICHDR4)																															
偏移地址: 5C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	插入通道 4 转换数据 该数据为对齐之后的数据

### 22.7.2.24 ADC标准通道数据寄存器 (ADC\_NCHDR)

ADC 标准通道数据寄存器 (ADC_NCHDR)																															
偏移地址: 60 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	标准通道转换数据 该数据为对齐之后的数据

### 22.7.2.25 ADC通用控制寄存器 1 (ADC\_CCR1)

ADC 通用控制寄存器 1 (ADC_CCR1)																																	
偏移地址: 64 <sub>H</sub>																																	
复位值: 00000000_00000000_11000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CHPBF_EN	CHNBF_EN	CHIBF_CALEN	TRMEN	Reserved	GAINCALEN	OFFCALEN	Reserved	VREFOEN	VRNSEL	VRPSEL	Reserved	DIFFEN	IREFEN	VRBUFEN	VCMBUFEN	VREFEN	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CKDIV

CHPBF_EN	Bits 31	R/W	ADC 通道正相输入 buffer 使能位 使用 ADC 时必须置一
CHNBF_EN	Bits 30	R/W	ADC 通道负相输入 buffer 使能位 使用 ADC 时必须置一
CHIBF_CALEN	Bits 29	R/W	ADC 输入 buffer 校准使能位 0: 禁止 1: 使能
TRMEN	Bit 28	R/W	ADC 数据修调使能位 0: 禁止 1: 使能
Reserved	Bit 27-26	—	保留
GAINCALEN	Bit 25	R/W	增益校准使能位 0: 禁止 1: 使能 注: 此控制位无效
OFFCALEN	Bit 24	R/W	偏置校准使能位 0: 禁止 1: 使能 注: 此控制位无效
Reserved	Bit 23-20	—	保留
VREFOEN	Bit 19	R/W	内部参考电压输出至引脚VREFP使能 0: 禁止 1: 使能 注: 此控制位无效
VRNSEL	Bit 18	R/W	负向参考电压选择位 0: VSS 1: VREFN 管脚 注: 不支持负向参考电压选择, 此控制位无效
VRPSEL	Bit 17-16	R/W	正向参考电压选择位 00: 电源电压 VDD33 01: 内部 2.4V/1.2V/0.8V 参考电压 10: 外部参考电压 (不经过 Buffer)

			<p>11: 外部参考电压 (经过 Buffer)</p> <p>真值表:</p> <table border="1"> <thead> <tr> <th>VRPSEL</th> <th>VRBUFEN</th> <th>参考电压</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>0</td> <td>电源电压 VDD33</td> </tr> <tr> <td>01</td> <td>0</td> <td>内部 2.4V/1.2V/0.8V 参考电压</td> </tr> <tr> <td>10</td> <td>0</td> <td>外部参考电压 (不经过 Buffer)</td> </tr> <tr> <td>11</td> <td>1</td> <td>外部参考电压 (经过 Buffer)</td> </tr> </tbody> </table> <p>注: 内部 2.4V/1.2V/0.8V 参考电压的配置可参照 ADC_CCR2[1:0]的描述;</p>	VRPSEL	VRBUFEN	参考电压	00	0	电源电压 VDD33	01	0	内部 2.4V/1.2V/0.8V 参考电压	10	0	外部参考电压 (不经过 Buffer)	11	1	外部参考电压 (经过 Buffer)
VRPSEL	VRBUFEN	参考电压																
00	0	电源电压 VDD33																
01	0	内部 2.4V/1.2V/0.8V 参考电压																
10	0	外部参考电压 (不经过 Buffer)																
11	1	外部参考电压 (经过 Buffer)																
Reserved	Bit 15-13	—	保留															
DIFFEN	Bit 12	R/W	<p>差分模式使能位</p> <p>0: 禁止</p> <p>1: 使能</p> <p>注: 不支持差分模式, 需要保证此控制位为 0</p>															
IREFEN	Bit 11	R/W	<p>Buffer 参考电流使能位</p> <p>0: 禁止</p> <p>1: 使能</p> <p>注: 此控制位无效</p>															
VRBUFEN	Bit 10	R/W	<p>参考电压 Buffer 使能位</p> <p>0: 禁止</p> <p>1: 使能</p>															
VCMBUFEN	Bit 9	R/W	<p>共模电压 Buffer 使能位</p> <p>0: 禁止</p> <p>1: 使能</p> <p>注: ADC 使能时此控制位必须有效</p>															
VREFEN	Bit 8	R/W	<p>内部参考电压使能位</p> <p>0: 禁止</p> <p>1: 使能</p>															
Reserved	Bit 7-3	—	保留															
CKDIV	Bit 2-0	R/W	<p>ADC 时钟分频选择位</p> <p>000: 1 分频</p> <p>001: 2 分频</p> <p>.....</p> <p>111: 128 分频</p>															

### 22. 7. 2. 26 ADC通用控制寄存器 2 (ADC\_CCR2)

ADC 通用控制寄存器 2 (ADC_CCR2)																															
偏移地址: 68 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										ADRLP		VREF_SEL			

Reserved	Bits31-3	—	保留																								
ADRLP	Bit 2	R/W	<b>ADC 工作频率选择位</b> 0: High speed mode(大于1MHz) 1: Low speed mode(小于1MHz)																								
VREF_SEL	Bit 1-0	R/W	<b>ADC 内部参考选择位:</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">VREF_SEL</th> <th colspan="2">ADC内部参考电压值</th> </tr> <tr> <th>Bit1</th> <th>Bit0</th> <th>ADC_VREFP</th> <th>ADC_VCM</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0.8V</td> <td>0.6V</td> </tr> <tr> <td>0</td> <td>1</td> <td>/</td> <td>/</td> </tr> <tr> <td>1</td> <td>0</td> <td>1.2V</td> <td>0.6V</td> </tr> <tr> <td>1</td> <td>1</td> <td>2.4V</td> <td>1.2V</td> </tr> </tbody> </table>	VREF_SEL		ADC内部参考电压值		Bit1	Bit0	ADC_VREFP	ADC_VCM	0	0	0.8V	0.6V	0	1	/	/	1	0	1.2V	0.6V	1	1	2.4V	1.2V
			VREF_SEL		ADC内部参考电压值																						
			Bit1	Bit0	ADC_VREFP	ADC_VCM																					
			0	0	0.8V	0.6V																					
			0	1	/	/																					
1	0	1.2V	0.6V																								
1	1	2.4V	1.2V																								

## 第23章 实时时钟(RTC)

### 23.1 概述

实时时钟 (RTC) 是一个独立的 BCD 定时器/计数器。RTC 提供一个日历时钟、两个可编程闹钟中断, 以及一个具有中断功能的周期性可编程唤醒标志。RTC 还包含用于管理低功耗模式的自动唤醒单元。

两个 32 位寄存器包含二进制十进制格式 (BCD) 的秒、分钟、小时 (12 或 24 小时制)、星期、日期、月份和年份。此外, 还可提供二进制格式的亚秒值。

系统可以自动将月份的天数补偿为 28、29 (闰年)、30 和 31 天。并且还可以进行夏令时补偿。

其它 32 位寄存器还包含可编程的闹钟亚秒、秒、分钟、小时、星期和日期。

此外, 还可以使用数字校准功能对晶振精度的偏差进行补偿。

上电复位后, 所有 RTC 寄存器都会受到保护, 以防止可能的非正常写访问。

无论器件状态如何 (运行模式、低功耗模式或处于复位状态), 只要电源电压保持在工作范围内, RTC 便不会停止工作。

### 23.2 特性

- ◇ 仅上电复位有效, 支持寄存器写保护, 有效避免软件误操作
- ◇ 时钟源支持 LOSC、ULRC
- ◇ 提供时钟和日历功能: 年、月、日、时、分、秒、星期
- ◇ 自动闰年识别, 有效期 100 年 (00-99)
- ◇ 12 小时和 24 小时模式设置可选
- ◇ 支持可编程的夏令时调整功能
- ◇ 两个可编程闹钟, 支持闹钟匹配字段配置
- ◇ 一个可编程的定时器, 并支持定时唤醒功能
- ◇ 可进行高精度数字校准, 最高精度  $\pm 0.0254$  ppm
- ◇ 支持硬件自动温度补偿算法, 可自动提供高精度计时
- ◇ 支持时间戳功能, 在发生时间戳事件时保持时间戳时间和日期
- ◇ 支持两路侵入检测功能
- ◇ 支持 128Bytes 备份寄存器, 在侵入事件发生时复位所有备份寄存器



### 23.3 结构框图

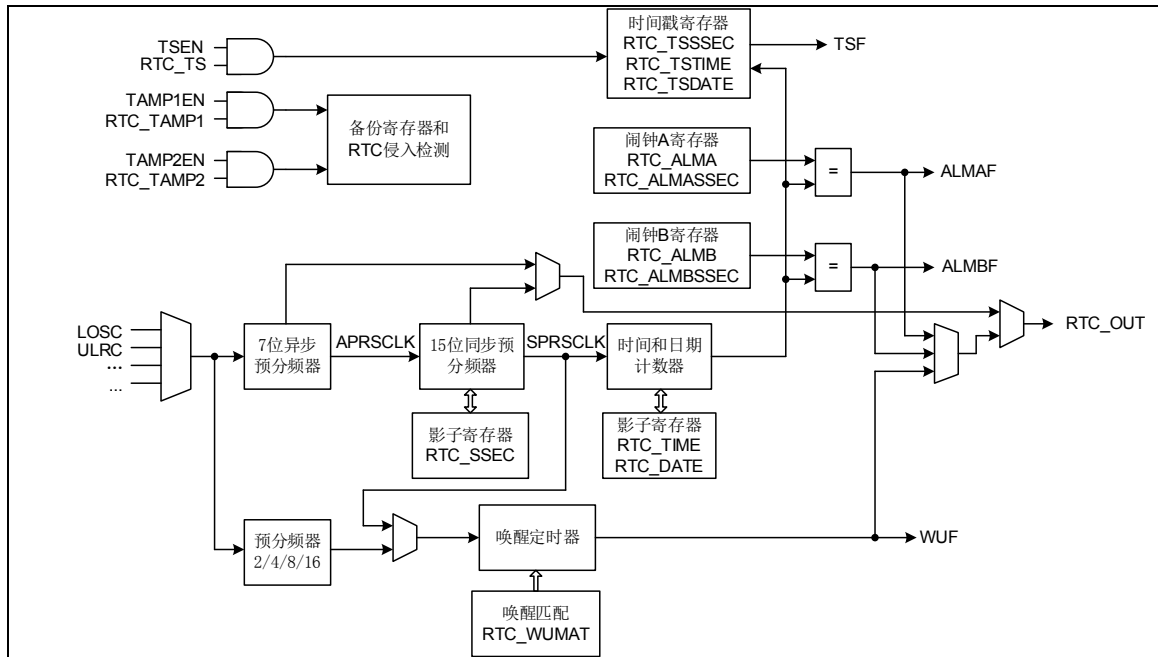


图 23-1 电路结构框图

### 23.4 功能描述

#### 23.4.1 时钟和预分频

RTC 时钟源通过备份域外设时钟控制寄存器 BKPC\_PCCR.RTCCK 进行选择, 可选择为:

- ◇ 32768Hz 低速时钟 ULRC
- ◇ 32768Hz 低速时钟 LOSC (停振自动切换至 ULRC)

RTC 支持工作时钟预分频, 预分频器分为两个可编程的预分频器:

- ◇ 7 位异步预分频器
- ◇ 15 位同步预分频器

两个预分频器可灵活使用, 使用较高的异步分频系数可降低 RTC 运行功耗, 但使用较高的同步分频系数可提升数字校准的精度。

APRSCLK 时钟频率计算公式为:

$$F_{APRSCLK} = \frac{F_{RTCCLK}}{APRS + 1}$$

APRSCLK 时钟用于亚秒寄存器 RTC\_SSEC 计数器提供时钟。当该计数器计数匹配同步分频数时会清零并重新开始计数。

SPRSCLK 时钟频率计算公式为:

$$F_{SPRSCLK} = \frac{F_{RTCCLK}}{(APRS + 1) \times (SPRS + 1)}$$

SPRSCLK 时钟用于更新日历，也可以用作 16 位唤醒定时器的计数时钟。

使用 32768Hz 频率的时钟获得频率为 1Hz 的时钟 SPRSCLK，可将异步分频系数设置为 1，并将同步分频系数设置为 32768。使用 1MHz 频率的时钟获得频率为 1Hz 的时钟 SPRSCLK，可将异步分频系数设置为 32，并将同步分频系数设置为 31250。

### 23.4.2 时钟和日历

RTC 时间和日历寄存器可通过对应的影子寄存器进行访问，或可设置为直接访问时间和日历寄存器以避免时钟同步造成的延时。

时间和日历寄存器包括：

- ◇ 亚秒寄存器 RTC\_SSEC
- ◇ 时间寄存器 RTC\_TIME
- ◇ 日期寄存器 RTC\_DATE

每两个 RTCCLK 时钟周期便将时间和日历寄存器复制至相应的影子寄存器中。在 STOP 和 STANDBY 模式下不会执行该操作，直至退出这两种模式后最迟两个 RTCCLK 时钟周期后更新。

在默认情况下，当读取时间和日历寄存器时，会访问到影子寄存器内容。也可通过将 RTC\_CON.SHDBP 置 1 来旁路影子寄存器而直接访问到时间和日历寄存器。读取影子寄存器时，APB2 时钟频率必须大于 RTCCLK 时钟频率的 4 倍以上。

### 23.4.3 可编程闹钟

RTC 提供两个可编程闹钟 A 和 B。

通过将寄存器 RTC\_CON.ALMAEN 和寄存器 RTC\_CON.ALMBEN 置 1 来使能闹钟 A 和闹钟 B。如果时间和日历的值分别与闹钟寄存器 RTC\_ALMA/RTC\_ALMASSEC 和 RTC\_ALMB/RTC\_ALMBSSEC 中的配置值相匹配，则标志位 RTC\_IFR.ALMAF 和 RTC\_IFR.ALMBF 会相应的置起。

通过闹钟寄存器 RTC\_ALMA 和 RTC\_ALMB 中的 xMSK 位，以及 RTC\_ALMASSEC 和 RTC\_ALMBSSEC 中的 SSMSK 位单独选择屏蔽相应的字段。

闹钟 A 和闹钟 B 可连接到 RTCO 端口输出，用户可通过配置 RTC\_CON.OSEL 进行使能，通过配置 RTC\_CON.POL 选择输出的极性。

### 23.4.4 周期性唤醒

RTC 提供一个 16 位周期性唤醒定时器，并可通过配置扩展至 17 位。可通过将 RTC\_CON.WUTE 置 1 使能唤醒功能。

唤醒定时器的时钟输入可选择为两种：

- ◇ RTCCLK 的分频时钟，可选择为 2/4/8/16 分频

- 当 RTCCLK 选择为 32768Hz 时，可配置的唤醒周期范围可选择为 122us 至 32s，分辨率为 61us

◇ SPRSCLK 时钟（通常为 1Hz 内部时钟）

- 当 SPRSCLK 为 1Hz，并且 RTC\_CON.WUSEL[2:1]配置为 0b10 时，可配置的唤醒周期范围可选择为 1s 至 18h，分辨率为 1s
- 当 SPRSCLK 为 1Hz，并且 RTC\_CON.WUSEL[2:1]配置为 0b11 时，可配置的唤醒周期范围可选择为 18h 至 36h，分辨率为 1s

如果唤醒定时器计数与 RTC\_WUMAT 寄存器值匹配后，标志位 RTC\_IFR.WUF 被置起，并且定时器清零后重新开始计数。系统复位和低功耗模式（SLEEP、STOP 和 STANDBY）对唤醒定时器均没有任何影响。

定时器溢出标志可连接到 RTCO 端口输出，用户可通过配置 RTC\_CON.OSEL 进行使能，通过配置 RTC\_CON.POL 选择输出的极性。

### 23.4.5 数字校准

RTC 提供了一种数字校准的方法，通过增加或减少同步分频器的系数，可对 RTC 时钟周期的偏差进行补偿。

通过将寄存器 RTC\_CALCON.CALEN 置 1 使能 RTC 数字校准功能，通过配置 RTC\_CALCON.CALP 选择数字校准的间隔周期。数字校准将在所选间隔周期的最后一秒进行补偿。

通过配置寄存器 RTC\_CALDR.CAL 来选择数字校准时增加或减少同步分频器的系数值，寄存器 RTC\_CALDR.CAL 为 16 位补码形式存放，其中 Bit15 为符号位，为 0 时会增加同步分频器的系数，RTC 时间会相应的变慢，为 1 时会减少同步分频器的系数，RTC 时间会相应的变快。

**例如：RTC 时钟源选择 32768Hz 时钟，每秒比标准时间慢 150us，选择每隔 20s 校准**

$$\text{RTC 实际周期为 } T_{\text{RTCCLK}} = \frac{10^6 + 150}{32768} \approx 30.522156 \text{ us}$$

$$\text{校准的周期数 } T = -\frac{150 \times 20}{T_{\text{RTCCLK}}} = -98.3 \approx -98 \text{ (对应补码为 } 0\text{xFF9D)}$$

校准寄存器 RTC\_CALDR.CAL 需要配置为 0xFF9D

$$\text{校准前偏差 } e = \frac{T_{\text{RTCCLK}} \times 32768 \times 20}{20 \times 10^6} - 1 = +150\text{ppm}$$

$$\text{校准后偏差 } e' = \frac{T_{\text{RTCCLK}} \times (32768 \times 20 + T)}{20 \times 10^6} - 1 = +0.441\text{ppm}$$

### 23.4.6 时间戳功能

RTC 提供了时间和日历的时间戳功能，通过将寄存器 RTC\_CON.TSEN 置 1 可启用时间戳功能。

当时间戳功能复用端口上检测到时间戳事件时，实时的时间和日历（包括亚秒、时间和日期寄存器）可被保存到时间戳寄存器中。时间戳寄存器包括：

- ◇ 时间戳亚秒寄存器 RTC\_TSSSEC
- ◇ 时间戳时间寄存器 RTC\_TSTIME
- ◇ 时间戳日期寄存器 RTC\_TSDATE

发生时间戳事件时，标志位 RTC\_IFR.TSF 将被置起，通过软件可将该标志位清零。若该标志位为 1 期间又检测到新的时间戳事件时，时间戳溢出标志位 RTC\_IFR.TSOVF 将被置起。

侵入事件的发送也可将时间和日历记录到时间戳寄存器中，同时也会置起时间戳标志位。

#### 23.4.7 侵入检测功能

RTC 提供了侵入检测的功能，通过对侵入检测复用的端口电平边沿或带滤波的电平检测，可产生侵入事件。将寄存器 RTC\_TAMPCON.TAMPxEN 置 1 可使能侵入检测的功能，配置寄存器 RTC\_TAMPCON.TAMPxLV 选择侵入电平的极性。

可通过配置寄存器 RTC\_TAMPCON.TAMPFLT 选择是否需要侵入电平进行滤波，并且选择滤波的采样事件，通过配置寄存器 RTC\_TAMPCON.TAMPCKS 选择采样时钟的频率。

侵入检测事件也可配置为同时触发时间戳事件，可通过将寄存器 RTC\_TAMPCON.TAMPTS 置 1 来使能。

发生侵入检测事件时，标志位 RTC\_IFR.TAMPxF 将被置起，通过软件可将该标志位清零。

侵入检测事件可同时将 RTC 备份寄存器 RTC\_BKPxR 全部清零。

#### 23.4.8 时钟输出

RTC 可提供 RTC 时钟分频连接到 RTCO 端口输出，用户可将寄存器 RTC\_CON.COE 置 1 进行使能，同时需要将寄存器 RTC\_CON.OSEL 配置为 0。通过配置寄存器 RTC\_CON.COSEL 选择输出时钟的频率。

#### 23.4.9 自动温度补偿（本设计中不支持RTC自动温度补偿）

RTC 自动温度补偿可由硬件自动根据当前温度计算出数字校准值，以实现 RTC 硬件自动调校功能。

RTC 自动温度补偿方法可查看相应的应用笔记。

### 23.5 基本配置

#### 23.5.1 RTC写保护

为避免程序的异常运行对 RTC 的误操作，RTC 写保护寄存器 RTC\_WPR 用于阻止程序对 RTC 其它寄存器的误写入。该寄存器保护范围为除 RTC\_WPR 寄存器外的 RTC 模块所有寄存器。

RTC\_WPR 寄存器为虚拟寄存器。要对 RTC 其它寄存器进行写操作时，需先对 RTC\_WPR 寄存器写 0x55AAAA55，之后可对 RTC 其它寄存器进行写操作。对 RTC\_WPR 寄存器写入其他值重新进入写保护状态，写保护状态下对 RTC 寄存器进行的写操作将被忽略。

可以通过读取 RTC\_WPR 寄存器确认 RTC 是否处于写保护状态，读出值为 0x00000000，

表示当前可对 RTC 寄存器进行写操作；读出值为 0x00000001 表示 RTC 处于写保护状态。RTC\_WPR 寄存器无其它读出值。

### 23.5.2 RTC校准写保护

为避免程序的异常运行对 RTC 校准的误操作，RTC 校准写保护寄存器 RTC\_CALWPR 用于阻止程序对 RTC 校准寄存器的误写入。该寄存器保护范围为除 RTC\_CALWPR 寄存器外的 RTC 校准相关的所有寄存器。

RTC\_CALWPR 寄存器为虚拟寄存器。要对 RTC 其它寄存器进行写操作时，需先对 RTC\_CALWPR 寄存器写 0x699655AA，之后可对 RTC 其它寄存器进行写操作。对 RTC\_CALWPR 寄存器写入其他值重新进入校准写保护状态，校准写保护状态下对 RTC 寄存器进行的写操作将被忽略。

可以通过读取 RTC\_CALWPR 寄存器确认 RTC 是否处于校准写保护状态，读出值为 0x00000000，表示当前可对 RTC 寄存器进行写操作；读出值为 0x00000001 表示 RTC 处于校准写保护状态。RTC\_CALWPR 寄存器无其它读出值。

注：校准相关寄存器需要同时解除 RTC 写保护和 RTC 校准写保护后才可进行正常写入操作

### 23.5.3 时间和日历初始化

由于 APB2 总线时钟与 RTC 时钟异步，因此 RTC 时间和日历计数器需要通过影子寄存器进行读写操作。

时间和日历初始化配置步骤如下：

1. 配置异步预分频系数和同步预分频系数
2. 将寄存器 RTC\_CON.GO 置 1 使能 RTC 计数器和分频器
3. 配置寄存器 RTC\_CON.HFM 选择时间格式（12 或 24 小时制）
4. 在时间和日期影子寄存器（RTC\_TIME 和 RTC\_DATE）中加载初始的时间和日期值
5. 通过判断标志位 RTC\_CON.BUSY 等待时间和日期寄存器写入同步完成

若 RTC 运行期间需要修改时间和日期，可重复以上 3~5 步骤操作。

时间和日期寄存器数据格式采用 BCD 编码。其计数范围为：

- ◇ 秒计数范围从 00 到 59，进位到分钟后从 59 变为 00。
- ◇ 分钟计数范围从 00 到 59，进位到小时后从 59 变为 00。
- ◇ 小时计数范围根据配置寄存器 RTC\_CON.HFM 的设置选择 12 或 24 小时制。
  - 选择 12 小时制时，分钟进位后小时从 PM11 更新到 AM12 或 AM11 更新到 PM12。
  - 选择 24 小时制时，分钟进位后小时从 23 到更新 00。
- ◇ 星期计数器为 3 位计数器，数值为 0-6，初始值可配置。
- ◇ 日计数按照每月最后一天加 1 进位到下月，日计数范围按月分为：
  - 一、三、五、七、八、十、十二月从 1 到 31；
  - 四、六、九、十一月从 1 到 30；

- 二月（普通年份）从 1 到 28；二月（闰年）从 1 到 29；
- ◇ 月计数范围从 1 到 12，进位到年后从 12 变为 1；
- ◇ 年计数范围从 00 到 99（00，04，08，…，92，96 为闰年），99 后进位到 00。

12 或 24 小时制小时格式对照表如下：

24 小时模式	12 小时模式	24 小时模式	12 小时模式
00	12(AM12)	12	12(PM12)
01	01(AM1)	13	01 (PM1)
02	02(AM2)	14	02 (PM2)
03	03(AM3)	15	03 (PM3)
04	04(AM4)	16	04 (PM4)
05	05(AM5)	17	05 (PM5)
06	06(AM6)	18	06 (PM6)
07	07(AM7)	19	07 (PM7)
08	08(AM8)	20	08 (PM8)
09	09(AM9)	21	09 (PM9)
10	10(AM10)	22	10 (PM10)
11	11(AM11)	23	11 (PM11)

表 23-1 小时格式对照表

#### 23.5.4 夏令时

RTC 支持夏令时的软件触发调整。通过将寄存器 `RTC_CON.SUB1H` 和 `RTC_CON.ADD1H` 置 1 可触发对当前时间增加或减少 1 小时进行调整，而无需进行重新初始化时间操作。寄存器 `RTC_CON.SUB1H` 和 `RTC_CON.ADD1H` 触发后硬件自动清零。

软件可实现配置夏令时选择寄存器 `RTC_CONT.DSTS` 选择当前是否为夏令时时间。触发完成后需要更新夏令时选择寄存器，以便之后软件可查询该寄存器值判断是否操作过夏令时时间。

当选择为冬季时间时，软件可触发时间增加 1 小时，触发时间减少 1 小时功能被禁止。当选择为夏季时间时，软件可触发时间减少 1 小时，触发时间增加 1 小时功能被禁止。

若不需要启用夏令时，则无需对这些寄存器进行操作。

#### 23.5.5 亚秒调整

RTC 可实现与远程高精度的时钟同步。在读取亚秒寄存器 `RTC_SSEC` 后，即可计算 RTC 与远程时钟的时间偏差。亚秒调整寄存器使用亚秒调整寄存器 `RTC_SSECTR` 可对 RTC 进行亚秒级修正，修正的最大偏差为 1s，修正的精度为 `ASPRCLK` 周期值。

亚秒寄存器 `RTC_SSEC` 的值实际为同步预分频器的计数值，当对亚秒调整寄存器 `RTC_SSECTR` 进行写入操作时触发亚秒调整。

### 23.6 RTC 中断

RTC 模块共有 11 个中断源，即

- ◇ 6个周期中断：年、月、日、时、分、秒中断
- ◇ 2个闹钟中断：闹钟 A 和闹钟 B 中断
- ◇ 时间戳和时间戳溢出中断
- ◇ 2个侵入检测中断：侵入检测 1 和侵入检测 2
- ◇ 周期性唤醒中断
- ◇ 寄存器同步完成中断和亚秒调整完成中断
- ◇ 自动温度补偿完成中断和自动温度补偿错误中断

每个中断源都有独立的使能位，使能位影响该中断是否产生 IRQ 中断请求，而不影响中断功能。即关闭相应中断使能，标志位仍可用于相应功能查询。当有多个中断使能时，各中断经过“或”逻辑产生 IRQ 中断请求。即任何一个被使能的中断产生中断事件时，均产生 IRQ 中断请求，且只有将所有的产生中断事件的中断标志清零后，IRQ 中断请求才解除。

## 23.7 特殊功能寄存器

### 23.7.1 寄存器列表

RTC 寄存器列表			
名称	偏移地址	类型	描述
RTC 基地址：4004_8400 <sub>H</sub>			
RTC_WPR	000 <sub>H</sub>	R/W	RTC 写保护寄存器
RTC_CON	004 <sub>H</sub>	R/W	RTC 控制寄存器
RTC_PSR	008 <sub>H</sub>	R/W	RTC 预分频寄存器
RTC_TAMPCON	00C <sub>H</sub>	R/W	RTC 侵入控制寄存器
RTC_TIME	010 <sub>H</sub>	R/W	RTC 时间寄存器
RTC_DATE	014 <sub>H</sub>	R/W	RTC 日期寄存器
RTC_SSEC	018 <sub>H</sub>	R	RTC 亚秒寄存器
RTC_WUMAT	01C <sub>H</sub>	R/W	RTC 唤醒匹配寄存器
RTC_ALMA	020 <sub>H</sub>	R/W	RTC 闹钟 A 寄存器
RTC_ALMB	024 <sub>H</sub>	R/W	RTC 闹钟 B 寄存器
RTC_ALMASSEC	028 <sub>H</sub>	R/W	RTC 闹钟 A 亚秒寄存器
RTC_ALMBSSEC	02C <sub>H</sub>	R/W	RTC 闹钟 B 亚秒寄存器
RTC_TSTIME	030 <sub>H</sub>	R	RTC 时间寄存器
RTC_TSDATE	034 <sub>H</sub>	R	RTC 日期寄存器
RTC_TSSSEC	038 <sub>H</sub>	R	RTC 亚秒寄存器
RTC_SSECTR	03C <sub>H</sub>	W	RTC 亚秒调整寄存器
RTC_IER	040 <sub>H</sub>	R/W	RTC 中断使能寄存器
RTC_IFR	044 <sub>H</sub>	R	RTC 中断标志寄存器
RTC_IFCR	048 <sub>H</sub>	W	RTC 中断状态清零寄存器
RTC_ISR	04C <sub>H</sub>	R	RTC 中断状态寄存器
RTC_CALWPR	050 <sub>H</sub>	R/W	RTC 校准写保护寄存器
RTC_CALCON	054 <sub>H</sub>	R/W	RTC 校准控制寄存器

RTC 寄存器列表			
名称	偏移地址	类型	描述
RTC 基地址: 4004_8400 <sub>H</sub>			
RTC_CALDR	058 <sub>H</sub>	R/W	RTC 校准值寄存器
Reserved	05C <sub>H</sub> ~0FC <sub>H</sub>	—	—
RTC_BKPRx	100 <sub>H</sub> ~17C <sub>H</sub>	R/W	RTC 备份寄存器 0~31

## 23.7.2 寄存器描述

### 23.7.2.1 RTC写保护寄存器 (RTC\_WPR)

RTC 写保护寄存器 (RTC_WPR)																																
偏移地址: 000 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																															WP	

Reserved	Bit 31-1	—	保留
WP	Bit 0	R	写保护状态位 0: 写保护解除 1: 写保护有效

注: 对该寄存器写入 0x55AAAA55 解除写保护, 写入其他值开启写保护。该寄存器保护除自身外的 RTC 所有区域。

### 23.7.2.2 RTC控制寄存器 (RTC\_CON)

RTC 控制寄存器 (RTC_CON)																																
偏移地址: 004 <sub>H</sub>																																
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved						SSEC	BUSY	Reserved	POL	OSEL	COSEL			COE	WUSEL			WUTE	REFCKEN	DSTS	SUB1H	ADD1H	TSPIN	TSSEL	TSEN	SHDBP	HFM	ALMBEN	ALMAEN	GO		

Reserved	Bit 31-26	—	保留
SSEC	Bit 25	R	亚秒调整状态位 0: 亚秒调整完成 1: 亚秒调整正在进行



BUSY	Bit 24	R	寄存器同步状态位 0: 寄存器同步完成 1: 寄存器同步正在进行
Reserved	Bit 23	—	保留
POL	Bit 22	RW	输出极性选择位 0: 高电平有效 (正常极性) 1: 低电平有效 (相反极性) 注: 极性选择只影响于闹钟和唤醒输出, 对校准输出无效
EOS	Bit 21-20	RW	事件输出选择位 00: 禁止输出 01: 使能闹钟 A 输出 10: 使能闹钟 B 输出 11: 使能唤醒输出
CKOS	Bit 19-17	RW	时钟输出选择位 000: 32768Hz 001: 1024Hz 010: 32Hz 011: 1Hz 100: 数字校准后 1Hz 101: 精确 1Hz 其他: 保留 注 1: RTCCLK 使用 LOSC (32768Hz), 且 APRS=0x0, SPRS=0x7FFF 注 2: 选择精确 1Hz 前需使能 PLL2 并等待其稳定
CKOE	Bit 16	RW	时钟输出使能位 0: 禁止 1: 使能 注: 当 EOS 配置为非 00 时, 校准输出被硬件强制禁止
WUCKS	Bit 15-13	RW	唤醒定时器时钟选择位 000: RTCCLK/16 001: RTCCLK/8 010: RTCCLK/4 011: RTCCLK/2 10x: 选择 SPRSCLK 11x: 选择 SPRSCLK 并将 WUT 计数值增加 $2^{16}$
WUTE	Bit 12	RW	唤醒定时器使能位 0: 禁止 1: 使能
Reserved	Bit 11	—	保留
DSTS	Bit 10	RW	夏令时选择位 0: 当前为冬季或不启用夏令时 1: 当前为夏季

SUB1H	Bit 9	W1	<p><b>冬季时间更改位</b></p> <p>0: 无操作 1: 当前时钟减少1小时 注: 当DST=0时, 硬件强制禁止</p>
ADD1H	Bit 8	W1	<p><b>夏季时间更改位</b></p> <p>0: 无操作 1: 当前时钟增加1小时 注: 当DST=1时, 硬件强制禁止</p>
TSPIN	Bit 7	R/W	<p><b>时间戳信号管脚选择位</b></p> <p>0: TAMPER0 管脚 1: TAMPER1 管脚</p>
TSSEL	Bit 6	R/W	<p><b>时间戳边沿选择位</b></p> <p>0: 上升沿 1: 下降沿</p>
TSEN	Bit 5	R/W	<p><b>时间戳使能位</b></p> <p>0: 禁止 1: 使能</p>
SHDBP	Bit 4	R/W	<p><b>影子寄存器模式选择位</b></p> <p>0: 读取时间和日期时直接从影子寄存器读取 1: 读取时间和日期时旁路影子寄存器 注 1: 涉及到的寄存器有 RTC_TIME、RTC_DATE 和 RTC_SSEC 注 2: 如果 APB 时钟频率小于 RTCCLK 的 8 倍时, 必须将此位配置为 1</p>
HFM	Bit 3	R/W	<p><b>小时格式选择位</b></p> <p>0: 24 小时制 1: 12 小时制 (AM/PM)</p>
ALMBEN	Bit 2	R/W	<p><b>闹钟 B 使能位</b></p> <p>0: 禁止 1: 使能</p>
ALMAEN	Bit 1	R/W	<p><b>闹钟 A 使能位</b></p> <p>0: 禁止 1: 使能</p>
GO	Bit 0	R/W	<p><b>RTC 运行配置位</b></p> <p>0: 停止 1: 运行 注: 该位可由软件置 1, RTC 开始运行, 软件无法对其清 0。该位仅在上电时被复位。</p>

### 23.7.2.3 RTC预分频寄存器 (RTC\_PSR)

RTC 预分频寄存器 (RTC_PSR)																															
偏移地址: 008 <sub>H</sub>																															
上电复位值: 00000000_01111111_00000000_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																APRS						Reserved	SPRS								

Reserved	Bit 31-23	—	保留
APRS	Bit 22-16	R/W	RTC 异步预分频系数
Reserved	Bit 15	—	保留
SPRS	Bit 14-0	R/W	RTC 同步预分频系数

### 23.7.2.4 RTC侵入控制寄存器 (RTC\_TAMPCON)

RTC 侵入控制寄存器 (RTC_TAMPCON)																																											
偏移地址: 00C <sub>H</sub>																																											
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved																TAMPFLT		TAMPCKS			TAMPPTS	Reserved									TAMP2LV		TAMP2EN		Reserved						TAMP1LV		TAMP1EN

Reserved	Bit 31-22	—	保留
TAMPFLT	Bit 21-20	R/W	<b>侵入电平滤波选择位</b> 00: 有效电平直接激活侵入事件 01: 在有效电平上连续 2 次采样后激活侵入事件 10: 在有效电平上连续 4 次采样后激活侵入事件 11: 在有效电平上连续 8 次采样后激活侵入事件
TAMPCKS	Bit 19-17	R/W	<b>侵入采样时钟选择位</b> 000: RTCCLK/32768 001: RTCCLK/16384 010: RTCCLK/8192 011: RTCCLK/4096 100: RTCCLK/2048 101: RTCCLK/1024 110: RTCCLK/512 111: RTCCLK/256

TAMPTS	Bit 16	R/W	侵入事件触发时间戳选择位 0: 发生侵入事件时不触发时间戳 1: 发生侵入事件时触发时间戳
Reserved	Bit 15-10	—	保留
TAMP2LV	Bit 9	R/W	侵入 2 有效电平选择位 0: 低电平 1: 高电平
TAMP2EN	Bit 8	R/W	侵入 2 检测使能位 0: 禁止 1: 使能
Reserved	Bit 7-2	—	保留
TAMP1LV	Bit 1	R/W	侵入 1 有效电平选择位 0: 低电平 1: 高电平
TAMP1EN	Bit 0	R/W	侵入 1 检测使能位 0: 禁止 1: 使能

### 23. 7. 2. 5 RTC时间寄存器 (RTC\_TIME)

RTC 时间寄存器 (RTC_TIME)																															
偏移地址: 010 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										PM	HRT	HRU				Reserved	MINT			MINU				Reserved	SECT		SECU				

Reserved	Bit 31-23	—	保留
PM	Bit 22	R/W	AM/PM 符号位 0: AM 或 24 小时制 1: PM
HRT	Bit 21-20	R/W	小时十位 注: 表示为 BCD 格式
HRU	Bit 19-16	R/W	小时个位 注: 表示为 BCD 格式
Reserved	Bit 15	—	保留
MINT	Bit 14-12	R/W	分钟十位 注: 表示为 BCD 格式
MINU	Bit 11-8	R/W	分钟个位 注: 表示为 BCD 格式

Reserved	Bit 7	—	保留
SECT	Bit 6-4	R/W	秒十位 注：表示为 BCD 格式
SECU	Bit 3-0	R/W	秒个位 注：表示为 BCD 格式

### 23.7.2.6 RTC日期寄存器 (RTC\_DATE)

RTC 日期寄存器 (RTC_DATE)																															
偏移地址: 014 <sub>H</sub>																															
复位值: 00000000_00000000_00000001_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				WD			YRT				YRU				Reserved		MONT	MONU				Reserved		DAYT	DAYU						

Reserved	Bit 31-27	—	保留
WD	Bit 26-24	R/W	星期 000: 周日 001: 周一 ..... 110: 周六 111: 禁止
YRT	Bit 23-20	R/W	年十位 注：表示为 BCD 格式
YRU	Bit 19-16	R/W	年个位 注：表示为 BCD 格式
Reserved	Bit 15-13	—	保留
MONT	Bit 12	R/W	月十位 注：表示为 BCD 格式
MONU	Bit 11-8	R/W	月个位 注：表示为 BCD 格式
Reserved	Bit 7-6	—	保留
DAYT	Bit 5-4	R/W	日十位 注：表示为 BCD 格式
DAYU	Bit 3-0	R/W	日个位 注：表示为 BCD 格式

### 23.7.2.7 RTC亚秒寄存器 (RTC\_SSEC)

RTC 亚秒寄存器 (RTC_SSEC)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R	亚秒值 该位表示同步预分频计数器的值

### 23.7.2.8 RTC唤醒匹配寄存器 (RTC\_WUMAT)

RTC 唤醒匹配寄存器 (RTC_WUMAT)																															
偏移地址: 01C <sub>H</sub>																															
上电复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VAL															

Reserved	Bit 31-16	—	保留
VAL	Bit 15-0	R/W	RTC 唤醒定时器匹配值

### 23.7.2.9 RTC闹钟A寄存器 (RTC\_ALMA)

RTC 闹钟 A 寄存器 (RTC_ALMA)																																					
偏移地址: 020 <sub>H</sub>																																					
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
WDS	DAWD							HRMSK	PM	HRT	HRU					MINMSK	MINT					MINU					SECMSK	SECT					SECU				

WDS	Bit 31	R/W	闹钟匹配星期选择位 0: 闹钟与日期匹配, Bits29-24 (DAYU/DAYT)
-----	--------	-----	--

			表示日期, Bit30 (DAYMSK) 表示日期掩码 1: 闹钟与星期匹配, Bits30-24 (WDE) 表示星期匹配使能位
DAWD	Bit 30-24	R/W	<p><b>闹钟日期匹配 (当 WDS=0 时)</b>                      Bit 27-24 DAYU: 日期个位 (表示为 BCD 格式)                      Bit 29-28 DAYT: 日期十位 (表示为 BCD 格式)                      Bit 30 DAYMSK: 闹钟日期掩码                      0: 闹钟匹配时日期有效                      1: 闹钟匹配时与日期无关</p> <p><b>闹钟星期匹配 (当 WDS=1 时)</b>                      Bit 30 WDE0: 星期日匹配使能位                      Bit 29 WDE1: 星期一匹配使能位                      Bit 28 WDE2: 星期二匹配使能位                      Bit 27 WDE3: 星期三匹配使能位                      Bit 26 WDE4: 星期四匹配使能位                      Bit 25 WDE5: 星期五匹配使能位                      Bit 24 WDE6: 星期六匹配使能位                      注: 当相应位为 1 时, 闹钟将匹配相应的星期</p>
HRMSK	Bit 23	R/W	<p><b>闹钟小时掩码</b>                      0: 闹钟匹配时小时有效                      1: 闹钟匹配时与小时无关</p>
PM	Bit 22	R/W	<p><b>AM/PM 符号位</b>                      0: AM 或 24 小时制                      1: PM</p>
HRT	Bit 21-20	R/W	<p><b>小时的十位</b>                      注: 表示为 BCD 格式</p>
HRU	Bit 19-16	R/W	<p><b>小时的个位</b>                      注: 表示为 BCD 格式</p>
MINMSK	Bit 15	R/W	<p><b>闹钟分钟掩码</b>                      0: 闹钟匹配时分钟有效                      1: 闹钟匹配时与分钟无关</p>
MINT	Bit 14-12	R/W	<p><b>分钟的十位</b>                      注: 表示为 BCD 格式</p>
MINU	Bit 11-8	R/W	<p><b>分钟的个位</b>                      注: 表示为 BCD 格式</p>
SECMSK	Bit 7	R/W	<p><b>闹钟秒掩码</b>                      0: 闹钟匹配时秒有效                      1: 闹钟匹配时与秒无关</p>
SECT	Bit 6-4	R/W	<p><b>秒的十位</b>                      注: 表示为 BCD 格式</p>
SECU	Bit 3-0	R/W	<p><b>秒的个位</b>                      注: 表示为 BCD 格式</p>

### 23.7.2.10 RTC闹钟B寄存器 (RTC\_ALMB)

RTC 闹钟 B 寄存器 (RTC_ALMB)																															
偏移地址: 024 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDS		DAWD						HRMSK		PM		HRT			HRU			MINMSK		MINT		MINU			SECMASK		SECT		SECU		

WDS	Bits 31	R/W	<p><b>闹钟匹配星期选择位</b></p> <p>0: 闹钟与日期匹配, Bits29-24 (DAYU/DAYT) 表示日期, Bit30 (DAYMSK) 表示日期掩码</p> <p>1: 闹钟与星期匹配, Bits30-24 (WDE) 表示星期匹配使能位</p>
DAWD	Bits 30-24	R/W	<p><b>闹钟日期匹配 (当 WDS=0 时)</b></p> <p>Bit 27-24 DAYU: 日期个位 (表示为 BCD 格式)</p> <p>Bit 29-28 DAYT: 日期十位 (表示为 BCD 格式)</p> <p>Bit 30 DAYMSK: 闹钟日期掩码</p> <p>0: 闹钟匹配时日期有效</p> <p>1: 闹钟匹配时与日期无关</p> <p><b>闹钟星期匹配 (当 WDS=1 时)</b></p> <p>Bit 30 WDE0: 星期日匹配使能位</p> <p>Bit 29 WDE1: 星期一匹配使能位</p> <p>Bit 28 WDE2: 星期二匹配使能位</p> <p>Bit 27 WDE3: 星期三匹配使能位</p> <p>Bit 26 WDE4: 星期四匹配使能位</p> <p>Bit 25 WDE5: 星期五匹配使能位</p> <p>Bit 24 WDE6: 星期六匹配使能位</p> <p>注: 当相应位为 1 时, 闹钟将匹配相应的星期</p>
HRMSK	Bit 23	R/W	<p><b>闹钟小时掩码</b></p> <p>0: 闹钟匹配时小时有效</p> <p>1: 闹钟匹配时与小时无关h</p>
PM	Bit 22	R/W	<p><b>AM/PM 符号位</b></p> <p>0: AM 或 24 小时制</p> <p>1: PM</p>
HRT	Bits 21-20	R/W	<p><b>小时的十位</b></p> <p>注: 表示为 BCD 格式</p>
HRU	Bits 19-16	R/W	<p><b>小时的个位</b></p> <p>注: 表示为 BCD 格式</p>
MINMSK	Bit 15	R/W	<p><b>闹钟分钟掩码</b></p> <p>0: 闹钟匹配时分钟有效</p>



			1: 闹钟匹配时与分钟无关
MINT	Bit 14-12	R/W	分钟的十位 注: 表示为 BCD 格式
MINU	Bit 11-8	R/W	分钟的个位 注: 表示为 BCD 格式
SECMSK	Bit 7	R/W	闹钟秒掩码 0: 闹钟匹配时秒有效 1: 闹钟匹配时与秒无关
SECT	Bit 6-4	R/W	秒的十位 注: 表示为 BCD 格式
SECU	Bit 3-0	R/W	秒的个位 注: 表示为 BCD 格式

### 23. 7. 2. 11 RTC闹钟A亚秒寄存器 (RTC\_ALMASSEC)

RTC 闹钟 A 亚秒寄存器 (RTC_ALMASSEC)																															
偏移地址: 028 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SSMSK				Reserved														SSEC									

Reserved	Bit 31-28	—	保留
SSECM	Bit 27-24	R/W	亚秒匹配掩码 0000: 亚秒不参与闹钟匹配 0001: 亚秒值 Bit0 匹配 SSEC[0] 0010: 亚秒值 Bit1-0 匹配 SSEC[1:0] 0011: 亚秒值 Bit2-0 匹配 SSEC[2:0] ..... 1101: 亚秒值 Bit12-0 匹配 SSEC[12:0] 1110: 亚秒值 Bit13-0 匹配 SSEC[13:0] 1111: 亚秒值 Bit14-0 匹配 SSEC[14:0] 注: 亚秒的其他位均与 0 匹配
Reserved	Bit 23-15	—	保留
SSEC	Bit 14-0	R/W	亚秒值 该值与同步预分频计数器的值匹配产生闹钟

### 23. 7. 2. 12 RTC闹钟B亚秒寄存器 (RTC\_ALMBSSEC)

RTC 闹钟 B 亚秒寄存器 (RTC_ALMBSSEC)																															
偏移地址: 02C <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SSMSK				Reserved												SSEC											

Reserved	Bit 31-28	—	保留
SSECM	Bit 27-24	RW	<b>亚秒匹配掩码</b> 0000: 亚秒不参与闹钟匹配 0001: 亚秒值 Bit0 匹配 SSEC[0] 0010: 亚秒值 Bit1-0 匹配 SSEC[1:0] 0011: 亚秒值 Bit2-0 匹配 SSEC[2:0] ..... 1101: 亚秒值 Bit12-0 匹配 SSEC[12:0] 1110: 亚秒值 Bit13-0 匹配 SSEC[13:0] 1111: 亚秒值 Bit14-0 匹配 SSEC[14:0] 注: 亚秒的其他位均与 0 匹配
Reserved	Bit 23-15	—	保留
SSEC	Bit 14-0	RW	<b>亚秒值</b> 该值与同步预分频计数器的值匹配产生闹钟

### 23. 7. 2. 13 RTC时间戳时间寄存器 (RTC\_TSTIME)

RTC 时间戳时间寄存器 (RTC_TSTIME)																															
偏移地址: 030 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										PM	HRT	HRU				Reserved	MINT	MINU				Reserved	SECT	SECU							

Reserved	Bit 31-23	—	保留
PM	Bit 22	R	<b>AM/PM 符号位</b> 0: AM 或 24 小时制 1: PM
HRT	Bit 21-20	R	小时计数十位

			注：表示为 BCD 格式
HRU	Bit 19-16	R	小时计数个位 注：表示为 BCD 格式
Reserved	Bit 15	—	保留
MINT	Bit 14-12	R	分钟计数十位 注：表示为 BCD 格式
MINU	Bit 11-8	R	分钟计数个位 注：表示为 BCD 格式
Reserved	Bit 7	—	保留
SECT	Bit 6-4	R	秒计数十位 注：表示为 BCD 格式
SECU	Bit 3-0	R	秒计数个位 注：表示为 BCD 格式

### 23.7.2.14 RTC时间戳日期寄存器 (RTC\_TSDATE)

RTC 时间戳日期寄存器 (RTC_TSDATE)																															
偏移地址: 034 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				WD			YRT				YRU				Reserved		MONT	MONU			Reserved		DAYT		DAYU						

Reserved	Bit 31-27	—	保留
WD	Bit 26-24	R	星期的个位 000: 周日 001: 周一 ..... 110: 周六 111: 禁止
YRT	Bit 23-20	R	年份的十位 注：表示为 BCD 格式
YRU	Bit 19-16	R	年份的个位 注：表示为 BCD 格式
Reserved	Bit 15-13	—	保留
MONT	Bit 12	R	月份的十位 注：表示为 BCD 格式
MONU	Bit 11-8	R	月份的个位 注：表示为 BCD 格式
Reserved	Bit 7-6	—	保留

DAYT	Bit 5-4	R	日期的十位 注：表示为 BCD 格式
DAYU	Bit 3-0	R	日期的个位 注：表示为 BCD 格式

### 23.7.2.15 RTC时间戳亚秒寄存器 (RTC\_TSSSEC)

RTC 时间戳亚秒寄存器 (RTC_TSSSEC)																															
偏移地址: 038 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SSEC															

Reserved	Bit 31-16	—	保留
SSEC	Bit 15-0	R	亚秒值 该位表示发送时间戳事件时同步预分频计数器的值

### 23.7.2.16 RTC亚秒调整寄存器 (RTC\_SSECTR)

RTC 亚秒调整寄存器 (RTC_SSECTR)																															
偏移地址: 03C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INC	Reserved															TRIM															

INC	Bit 31	W	秒修正位 0: 无修正 1: 增加 1 秒 注：该位读出始终为 0
Reserved	Bit 30-15	—	保留
TRIM	Bit 14-0	W	亚秒修正位 该位表示减小相应的亚秒数，该位与 INC 配合使用可任意对时钟进行 1 秒之内的调整 注：该位读出始终为 0

注：该寄存器不支持字节操作。

### 23.7.2.17 RTC中断使能寄存器 (RTC\_IER)

RTC 中断使能寄存器 (RTC_IER)																																		
偏移地址: 040 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved						TCE	TCC	Reserved						WU	SSTC	RSC	Reserved			TAMP2	TAMP1	TSOV	TS	ALMB	ALMA	Reserved			YR	MON	DAY	HR	MIN	SEC

Reserved	Bit 31-26	—	保留
TCE	Bit 25	R/W	温度补偿错误中断使能位 0: 禁止 1: 使能
TCC	Bit 24	R/W	温度补偿完成中断使能位 0: 禁止 1: 使能
Reserved	Bit 23-19	—	保留
WU	Bit 18	R/W	唤醒中断使能位 0: 禁止 1: 使能
SSTC	Bit 17	R/W	亚秒调整完成中断使能位 0: 禁止 1: 使能
RSC	Bit 16	R/W	寄存器同步完成中断使能位 0: 禁止 1: 使能
Reserved	Bit 15-14	—	保留
TAMP2	Bit 13	R/W	侵入检测 2 中断使能位 0: 禁止 1: 使能
TAMP1	Bit 12	R/W	侵入检测 1 中断使能位 0: 禁止 1: 使能
TSOV	Bit 11	R/W	时间戳溢出中断使能位 0: 禁止 1: 使能
TS	Bit 10	R/W	时间戳中断使能位 0: 禁止 1: 使能
ALMB	Bit 9	R/W	闹钟 B 中断使能位

			0: 禁止 1: 使能
ALMA	Bit 8	R/W	闹钟 A 中断使能位 0: 禁止 1: 使能
Reserved	Bit 7-6	—	保留
YR	Bit 5	R/W	年份中断使能位 0: 禁止 1: 使能
MON	Bit 4	R/W	月份中断使能位 0: 禁止 1: 使能
DAY	Bit 3	R/W	日中断使能位 0: 禁止 1: 使能
HR	Bit 2	R/W	小时中断使能位 0: 禁止 1: 使能
MIN	Bit 1	R/W	分钟中断使能位 0: 禁止 1: 使能
SEC	Bit 0	R/W	秒中断使能位 0: 禁止 1: 使能

### 23. 7. 2. 18 RTC中断标志寄存器 (RTC\_IFR)

RTC 中断标志寄存器 (RTC_IFR)																																		
偏移地址: 044 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved						TCEF	TCCF	Reserved						WUF	SSTCF	RSCF	Reserved			TAMP2F	TAMP1F	TSOVF	TSF	ALMBF	ALMAF	Reserved			YRF	MONF	DAYF	HRF	MINF	SECF

Reserved	Bit 31-26	—	保留
TCEF	Bit 25	R	温度补偿错误状态标志 0: 未发生事件 1: 已发生事件
TCCF	Bit 24	R	温度补偿完成中断状态标志 0: 未发生事件 1: 已发生事件

Reserved	Bit 23-19	—	保留
WUF	Bit 18	R	唤醒中断状态标志 0: 未发生事件 1: 已发生事件
SSTCF	Bit 17	R	亚秒调整完成中断状态标志 0: 未发生事件 1: 已发生事件
RSCF	Bit 16	R	寄存器同步完成中断状态标志 0: 未发生事件 1: 已发生事件
Reserved	Bit 15-14	—	保留
TAMP2F	Bit 13	R	侵入检测 2 中断状态标志 0: 未发生事件 1: 已发生事件
TAMP1F	Bit 12	R	侵入检测 1 中断状态标志 0: 未发生中断 1: 已发生中断
TSOVF	Bit 11	R	时间戳溢出中断状态标志 0: 未发生事件 1: 已发生事件
TSF	Bit 10	R	时间戳中断状态标志 0: 未发生事件 1: 已发生事件
ALMBF	Bit 9	R	闹钟 B 中断状态标志 0: 未发生事件 1: 已发生事件
ALMAF	Bit 8	R	闹钟 A 中断状态标志 0: 未发生事件 1: 已发生事件
Reserved	Bit 7-6	—	保留
YRF	Bit 5	R	年份中断状态标志 0: 未发生事件 1: 已发生事件
MONF	Bit 4	R	月份中断状态标志 0: 未发生事件 1: 已发生事件
DAYF	Bit 3	R	日中断状态标志 0: 未发生事件 1: 已发生事件
HRF	Bit 2	R	小时中断状态标志 0: 未发生事件 1: 已发生事件
MINF	Bit 1	R	分钟中断状态标志

			0: 未发生事件 1: 已发生事件
SECF	Bit 0	R	秒中断状态标志 0: 未发生事件 1: 已发生事件

注：通过操作 RTC\_IFCR 寄存器清零以上标志

### 23. 7. 2. 19 RTC中断标志清零寄存器 (RTC\_IFCR)

RTC 中断标志清零寄存器 (RTC_IFCR)																																		
偏移地址: 048 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved						TCEFC	TCCFC	Reserved						WUFC	SSTCFC	RSCFC	Reserved			TAMP2FC	TAMP1FC	TSOVFC	TSSTC	ALMBFC	ALMAFC	Reserved			YRFC	MONFC	DAYFC	HRFC	MINFC	SECF

Reserved	Bit 31-26	—	保留
TCEFC	Bit 25	W1	温度补偿错误状态标志清零 0: 无操作 1: 中断状态标志清零
TCCFC	Bit 24	W1	温度补偿完成中断状态标志清零 0: 无操作 1: 中断状态标志清零
Reserved	Bit 23-19	—	保留
WUFC	Bit 18	W1	唤醒中断状态标志清零 0: 无操作 1: 中断状态标志清零
SSTCFC	Bit 17	W1	亚秒调整完成中断状态标志清零 0: 无操作 1: 亚秒调整完成中断状态标志清零
RSCFC	Bit 16	W1	寄存器同步完成中断状态标志清零 0: 无操作 1: 寄存器同步完成中断状态标志清零
Reserved	Bit 15-14	—	保留
TAMP2FC	Bit 13	W1	侵入检测 2 中断状态标志清零 0: 无操作 1: 侵入检测 2 中断状态标志清零
TAMP1FC	Bit 12	W1	侵入检测 1 中断状态标志清零 0: 无操作



			1: 侵入检测 1 中断状态标志清零
TSOVFC	Bit 11	W1	时间戳溢出中断状态标志清零 0: 无操作 1: 时间戳溢出中断状态标志清零
TSSTC	Bit 10	W1	时间戳中断状态标志清零 0: 无操作 1: 时间戳中断状态标志清零
ALMBFC	Bit 9	W1	闹钟 B 中断状态标志清零 0: 无操作 1: 闹钟 B 中断状态标志清零
ALMAFC	Bit 8	W1	闹钟 A 中断状态标志清零 0: 无操作 1: 闹钟 A 中断状态标志清零
Reserved	Bit 7-6	—	保留
YRFC	Bit 5	W1	年份中断状态标志清零 0: 无操作 1: 年份中断状态标志清零
MONFC	Bit 4	W1	月份中断状态标志清零 0: 无操作 1: 月份中断状态标志清零
DAYFC	Bit 3	W1	日中断状态标志清零 0: 无操作 1: 日中断状态标志清零
HRFC	Bit 2	W1	小时中断状态标志清零 0: 无操作 1: 小时中断状态标志清零
MINFC	Bit 1	W1	分钟中断状态标志清零 0: 无操作 1: 分钟中断状态标志清零
SECFC	Bit 0	W1	秒中断状态标志清零 0: 无操作 1: 秒中断状态标志清零

### 23.7.2.20 RTC中断状态寄存器 (RTC\_ISR)

RTC 中断状态寄存器 (RTC_ISR)																																
偏移地址: 04C <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved						TCEF	TCCF	Reserved						WUF	SSTCF	RSCF	Reserved		TAMP2F	TAMP1F	TSOVF	TSF	ALMBF	ALVAF	Reserved		YRF	MONF	DAYF	HRF	MINF	SECF

Reserved	Bit 31-26	—	保留
TCEF	Bit 25	R	温度补偿错误状态标志 0: 未发生中断 1: 已发生中断
TCCF	Bit 24	R	温度补偿完成中断状态标志 0: 未发生中断 1: 已发生中断
Reserved	Bit 23-19	—	保留
WUF	Bit 18	R	唤醒中断状态标志 0: 未发生中断 1: 已发生中断
SSTCF	Bit 17	R	亚秒调整完成中断状态标志 0: 未发生中断 1: 已发生中断
RSCF	Bit 16	R	寄存器同步完成中断状态标志 0: 未发生中断 1: 已发生中断
Reserved	Bit 15-14	—	保留
TAMP2F	Bit 13	R	侵入检测 2 中断状态标志 0: 未发生中断 1: 已发生中断
TAMP1F	Bit 12	R	侵入检测 1 中断状态标志 0: 未发生中断 1: 已发生中断
TSOVF	Bit 11	R	时间戳溢出中断状态标志 0: 未发生中断 1: 已发生中断
TSF	Bit 10	R	时间戳中断状态标志 0: 未发生中断 1: 已发生中断
ALMBF	Bit 9	R	闹钟 B 中断状态标志 0: 未发生中断 1: 已发生中断
ALMAF	Bit 8	R	闹钟 A 中断状态标志 0: 未发生中断 1: 已发生中断
Reserved	Bit 7-6	—	保留
YRF	Bit 5	R	年份中断状态标志 0: 未发生中断 1: 已发生中断
MONF	Bit 4	R	月份中断状态标志 0: 未发生中断

			1: 已发生中断
DAYF	Bit 3	R	日中断状态标志 0: 未发生中断 1: 已发生中断
HRF	Bit 2	R	小时中断状态标志 0: 未发生中断 1: 已发生中断
MINF	Bit 1	R	分钟中断状态标志 0: 未发生中断 1: 已发生中断
SECF	Bit 0	R	秒中断状态标志 0: 未发生中断 1: 已发生中断

### 23. 7. 2. 21 RTC校准写保护寄存器 (RTC\_CALWPR)

RTC 校准写保护寄存器 (RTC_CALWPR)																															
偏移地址: 050 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															WP

Reserved	Bit 31-1	—	保留
WP	Bit 0	R	写保护状态位 0: 写保护解除 1: 写保护有效

注: 对该寄存器写入 0x699655AA 解除写保护, 写入其他值开启写保护。该寄存器保护 054<sub>H</sub>~088<sub>H</sub> 所有地址区域。

### 23. 7. 2. 22 RTC校准控制寄存器 (RTC\_CALCON)

RTC 校准控制寄存器 (RTC_CALCON)																															
偏移地址: 054 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DCMACC	ALG	TCP				ERR	BUSY	TCM	Reserved											CALP	CALEN			

Reserved	Bit 31-25	—	保留
DCMACC	Bit 24	RW	补偿值小数累积选择位 0: 不累积 1: 累积
ALG	Bit 23	RW	自动温度补偿算法选择位 0: 四次多项式法 1: 二次抛物线法
TCP	Bit 22-20	RW	自动温度补偿值计算周期选择位 000: 每隔10秒计算一次 001: 每隔20秒计算一次 010: 每隔1分钟计算一次 011: 每隔2分钟计算一次 100: 每隔5分钟秒计算一次 101: 每隔10分钟秒计算一次 110: 每隔20分钟秒计算一次 111: 每隔1小时计算一次
ERR	Bit 19	R	自动温度补偿异常状态位 0: 自动温度补偿计算正常 1: 自动温度补偿发生异常
BUSY	Bit 18	R	自动温度补偿模式状态位 0: 自动温度补偿计算完成或未开始 1: 自动温度补偿正在计算
TCM	Bit 17-16	RW	自动温度补偿模式选择位 00: 禁止自动温度补偿 01: 使能自动温度补偿, 硬件自动获取温度 10: 使能自动温度补偿, 温度由寄存器 RTC_TEMPR写入, 写入后自动触发自动校准 11: 使能自动温度补偿, 硬件自动获取温度, 对 寄存器RTC_TEMPR任意写入操作触发自动校准
Reserved	Bit 15-4	—	保留
CALP	Bit 3-1	RW	校准模式选择位 000: 每隔10秒校准一次 001: 每隔20秒校准一次 010: 每隔1分钟校准一次 011: 每隔2分钟校准一次 100: 每隔5分钟校准一次 101: 每隔10分钟校准一次 110: 每隔20分钟校准一次 111: 每隔 1 秒校准一次 注: RTC 时钟输出选择精确 1Hz 时不可选择 CALP=111
CALEN	Bit 0	RW	校准使能位 0: 禁止

			1: 使能
--	--	--	-------

### 23.7.2.23 RTC校准寄存器 (RTC\_CALDR)

RTC 校准值寄存器 (RTC_CALDR)																															
偏移地址: 058 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																VAL															

DATA	Bit 31-16	R	RTC 实时校准值 注: 读该寄存器时需连续读两次, 两次值相同时该值有效
VAL	Bit 15-0	R/W	<b>RTC 数字校准值</b> 0x7FFF: 正向补偿 32767 ..... 0x0001: 正向补偿 1 0x0000: 无补偿 0xFFFF: 负向补偿 1 ..... 0x8001: 负向补偿 32767 0x8000: 预留 注 1: 使能自动温度补偿后, 该寄存器写入无效 注 2: 写入该寄存器之前需先使能 CALEN, 否则写入后补偿无效 注 3: 写入校准值后最多延时 1 秒后生效

### 23.7.2.24 RTC备份寄存器 (RTC\_BKPRx) (x = 0..31)

RTC 备份寄存器 X (RTC_BKPRx) (x = 0..31)																															
偏移地址: 100 <sub>H</sub> ~ 17C <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
系统复位: 不受影响																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKPRx																															

BKPRx	Bit 31-0	RW	<p><b>备份寄存器</b></p> <p>可以对该寄存器进行读取或写入操作。</p> <p>该寄存器由备份域电源供电，当主电源域电源掉电时，寄存器的值仍可保持，并且系统复位不会影响寄存器的值。</p> <p>发生侵入检测事件时该寄存器会被复位，并且只要RTC_IFR.TAMPxF=1，该寄存器就一直保持复位。</p>
-------	----------	----	---

## 第24章 高级定时器(AD16C4T)

### 24.1 概述

高级控制定时器 (AD16C4T) 是一个功能强大、配置灵活的定时器模块, 它包含一个 16-bit, 定时器, 具有定时、计数、脉冲输入信号测量 (输入捕获)、产生特定 PWM 波 (输出比较) 等功能。

### 24.2 特性

- ◇ 16 位递增, 递减, 递增/递减自动加载计数器
- ◇ 16 位可编程预分频器, 可在定时器运行中对计数器工作时钟进行 1 到 65536 间的任意分频
- ◇ 带有四个独立通道, 每个通道支持以下功能
  - ◇ 输入捕获
  - ◇ 输出比较
  - ◇ PWM 输出
  - ◇ 单脉冲输出
- ◇ 通道 1~3 支持互补输出, 死区时间可配
- ◇ 在给定数目的计数周期之后更新重复计数寄存器
- ◇ 支持刹车功能, 刹车后定时器输出状态可控
- ◇ 支持中断/DMA:
  - ◇ 更新事件: 计数器上溢/下溢, 计数器初始化
  - ◇ 触发事件
  - ◇ 换相事件
  - ◇ 输入捕获
  - ◇ 输出比较
  - ◇ 刹车输入
- ◇ 支持增量 (正交) 编码及霍尔电路
- ◇ 通过外设互联 (PIS) 可支持与片上其他定时器的互联工作

## 24.3 结构框图

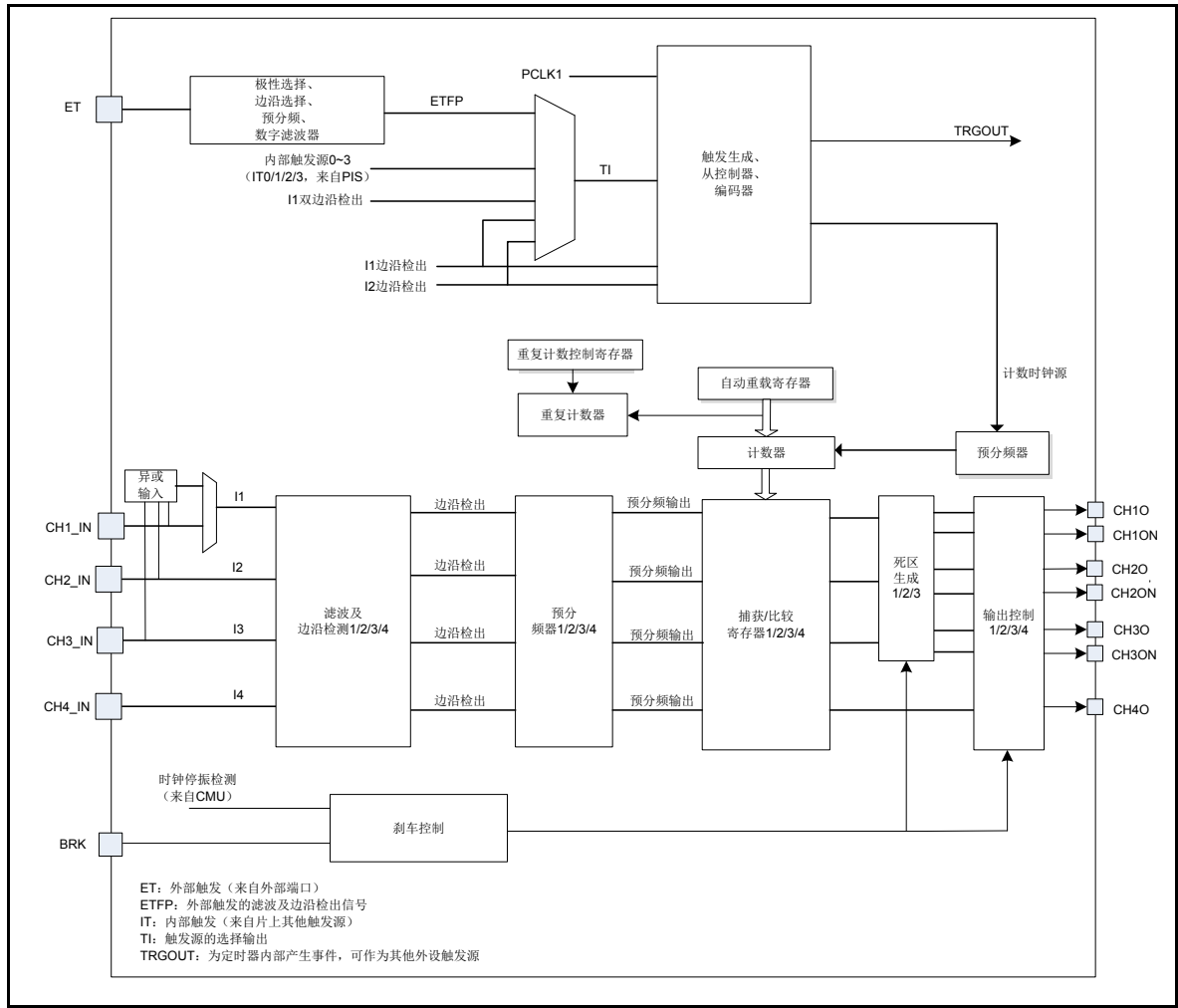


图 24-1 高级定时器电路结构框图

## 24.4 功能描述

### 24.4.1 预分频器

定时器包含一个 16-bit 的计数器 (AD16C4Tn\_COUNT)，计数时钟由预分频寄存器 (AD16C4Tn\_PRES) 进行分频。计数周期由自动重载计数器 (AD16C4Tn\_AR) 设定。重复计数寄存器则可指定计数周期数目 (AD16C4Tn\_REPAR)。

自动重载寄存器 (AD16C4Tn\_AR) 是一个可缓存的寄存器。当 AD16C4Tn\_CON1 寄存器的 ARPEN 位复位时，AD16C4Tn\_AR 寄存器重载功能失效，AD16C4Tn\_AR 就是有效寄存器；ARPEN 置位时，AD16C4Tn\_AR 寄存器具有重载功能，产生更新事件 (UEV) 时，加载值 (AD16C4Tn\_AR 寄存器值) 更新到影子寄存器。

当 AD16C4Tn\_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢 (或递减下溢) 时会产生更新事件 (UEV)。同样，软件方式也可产生更新事件。AD16C4Tn\_CON1 寄存器的 CNTEN 置位时，计数器开始计数。



注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 AD16C4Tn\_PRES 寄存器值+1 次分频。由于 AD16C4Tn\_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

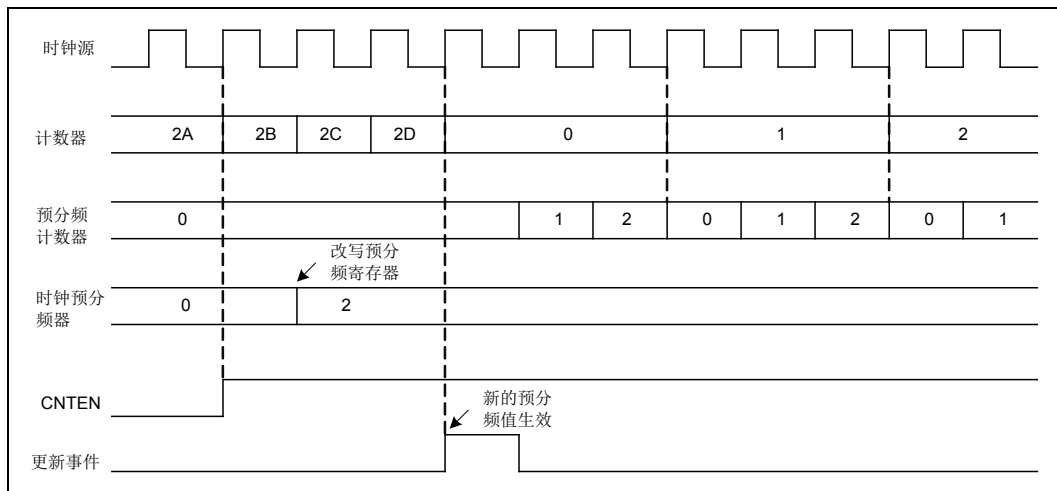


图 24-2 预分频值计数时序图

### 24.4.2 重复计数器

重复计数器用于控制发生多少次上溢或下溢次后产生更新事件。

重复计数器递减：

- ◇ 递增模式的每次上溢
- ◇ 递减模式的每次下溢
- ◇ 中心对齐模式时，计数器上溢与下溢。中心对齐模式限制了最大重复次数为 128 个 PWM 周期，每个 PWM 周期内可更新两次占空比。

AD16C4Tn\_REPAR 寄存器是一个可缓存寄存器。软件（置位 AD16C4Tn\_SGE 寄存器中的 SGU 位）或硬件从机模式控制方式产生更新事件时，无论重复计数器为何值，AD16C4Tn\_REPAR 寄存器中值会立即更新到重复计数器的影子寄存器中。

中心对齐模式下，REPAR 中值为奇数时，更新事件是在上溢或下溢时产生，取决于何时写 REPAR 寄存器及何时开始计数。若在启动计数器前写 REPAR，则上溢时产生 UEV，反之则在下溢时产生 UEV。

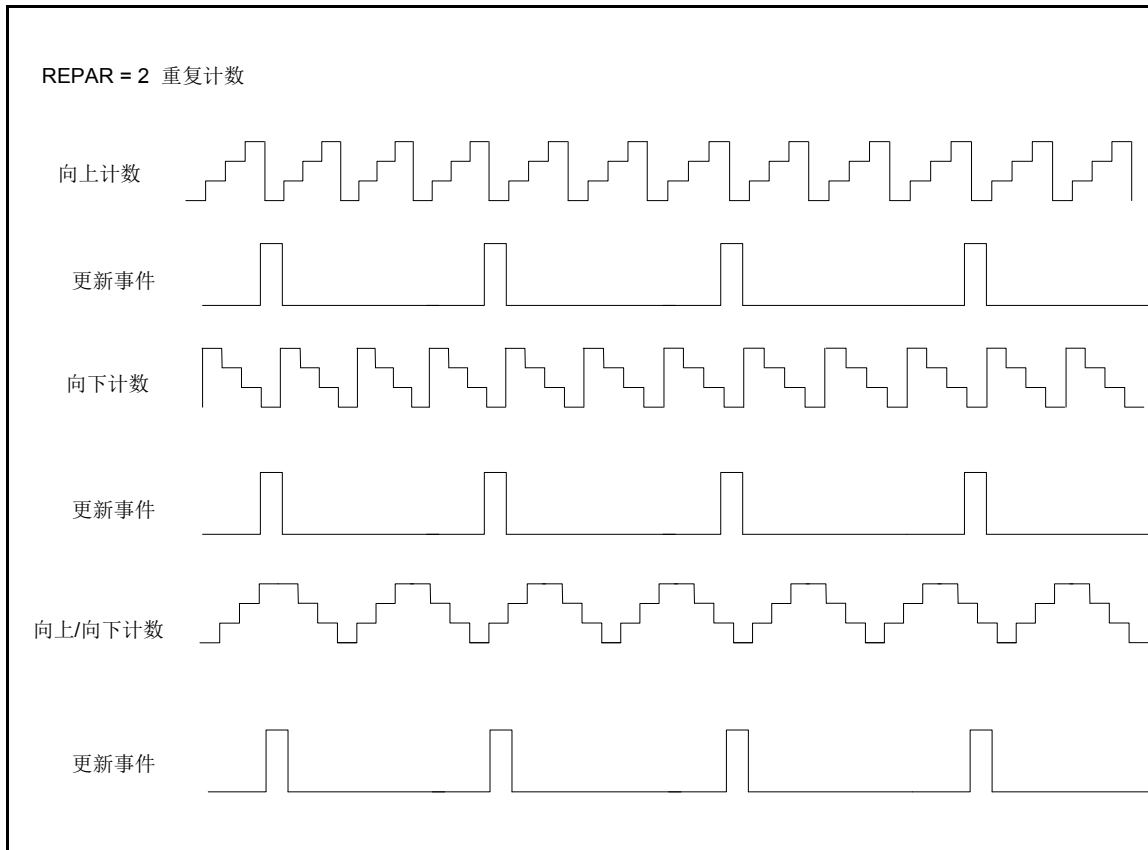


图 24-3 重复计数器工作模式

注意：置位 AD16C4Tn\_SGE 寄存器中的 SGU 位也可以产生更新事件。

### 24.4.3 时钟源

计数器工作时钟可以选择内部时钟(INT\_CLK)、外部时钟源 1 (I1、I2、I3、I4)、外部时钟源 2 (ET)，内部触发输入 (IT1、IT2、IT3、IT4)

#### 24.4.3.1 内部时钟源 (INT\_CLK)

若从模式控制器被关闭(AD16C4Tn\_SMCON 寄存器内, SMODS= "000"), 则 CNTEN, AD16C4Tn\_CON1.DIRSEL 与 AD16C4Tn\_SGE.SGU 位为实际控制位, 这些位只能软件修改 (SGU 位除外, 仍硬件自动清除)。一旦 CNTEN 位被写为'1', 预分频器就由内部 INT\_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况, 没有分频。

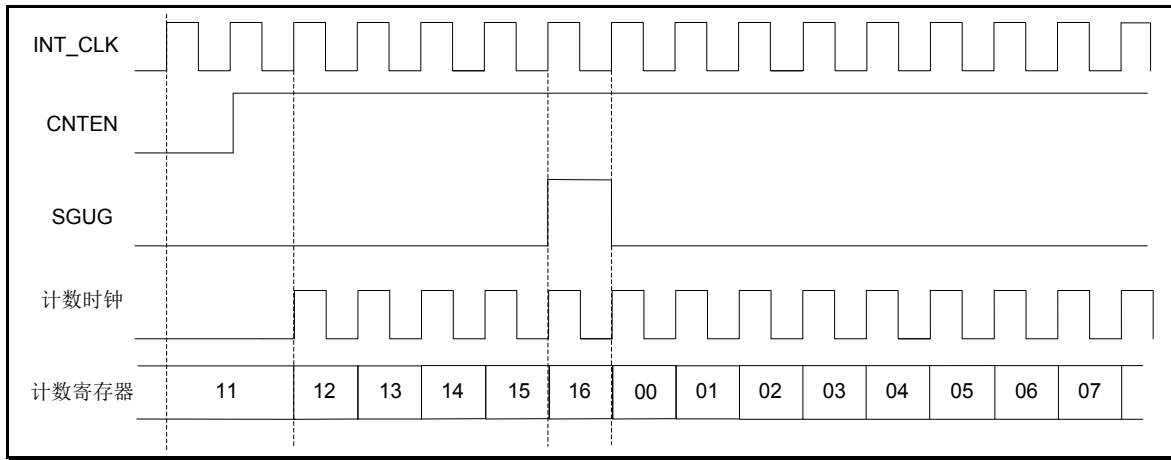


图 24-4 采用内部时钟计数

#### 24.4.3.2 外部时钟源 1

AD16C4Tn\_SMCON 寄存器的 SMODS= "111"时, 可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

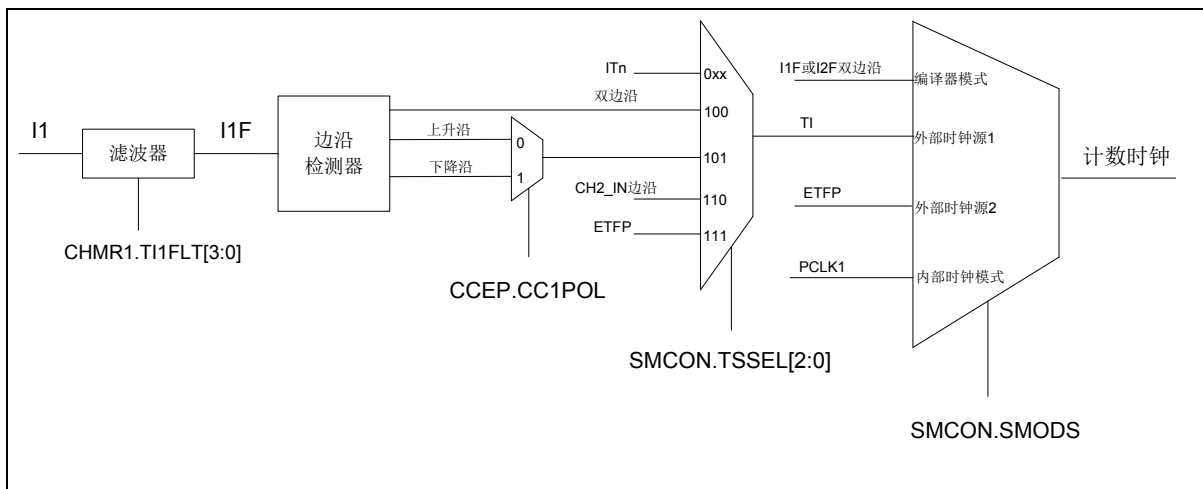


图 24-5 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

1. AD16C4Tn\_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
2. 设置 AD16C4Tn\_SMCON 寄存器中的 TSSEL 选择外部时钟源。
3. 如外部时钟源为 I1，可配置 AD16C4Tn\_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 AD16C4Tn\_CCEP 寄存器中 CC1POL = '0'，选择极性位上升沿。
4. 写 AD16C4Tn\_CHMR1 寄存器的 I1FLT[3: 0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
5. AD16C4Tn\_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

### 24.4.3.3 外部时钟源 2

置位 AD16C4Tn\_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部输入输入模块的概况。

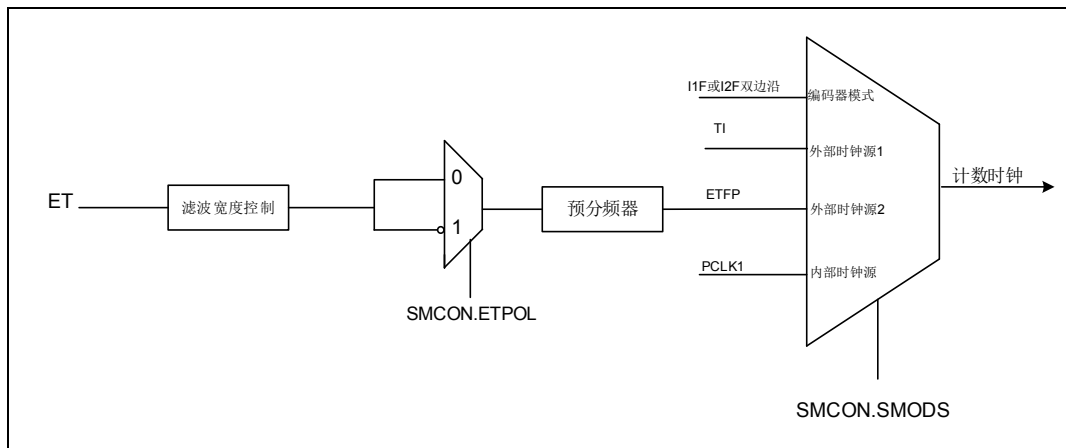


图 24-6 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

1. 设置 AD16C4Tn\_SMCON 寄存器的 ETFLT[3: 0]，配置输入滤波时间。
2. 设置 AD16C4Tn\_SMCON 寄存器中 ETPSEL[1: 0]，设置预分频器。
3. 设置 AD16C4Tn\_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
4. 设置 AD16C4Tn\_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
5. 设置 AD16C4Tn\_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

### 24.4.3.4 内部触发输入 (ITn)

当 AD16C4Tn\_SMCON 寄存器的 SMODS = "111", 选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

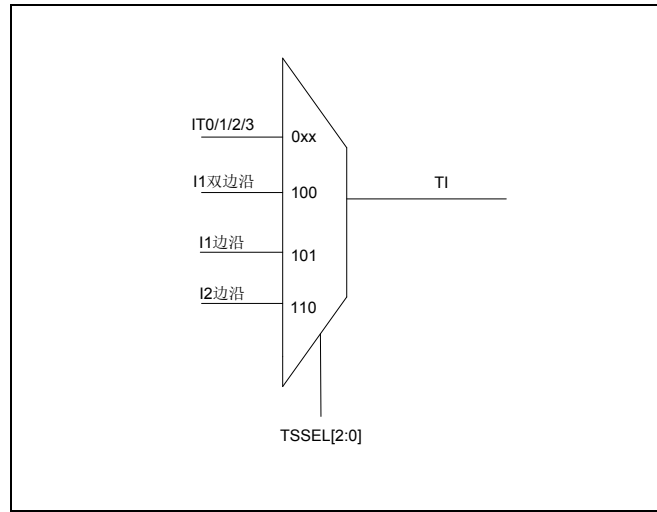


图 24-7 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

1. AD16C4Tn\_SMCON 寄存器中 SMODS = "111", 配置外部时钟模式 1。
2. AD16C4Tn\_SMCON 寄存器的 TSSEL = "0xx", 选定 ITn 作为触发输入源。
3. AD16C4Tn\_CON1 寄存器的 CNTEN = '1', 使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路。

## 24.4.4 计数模式

### 24.4.4.1 递增计数模式

当 AD16C4Tn\_REPAR 寄存器值为 0 时，定时器配置为递增模式，计数器从 0 开始递增，直至 AD16C4Tn\_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件 (UEV)。当 AD16C4Tn\_REPAR 寄存器不为 0 时，则在 AD16C4Tn\_REPAR+1 次计数后产生更新事件。

当有更新事件 (UEV) 产生时，预装载寄存器会更新到影子寄存器，更新标志位 (AD16C4Tn\_RIF 寄存器中的 UEVTIF 位) 置位 (取决于 UERSEL 位)：

- ◇ 更新 AD16C4Tn\_REPAR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn\_AR 寄存器的值到影子寄存器
- ◇ 更新 AD16C4Tn\_PRES 寄存器的值到影子寄存器

下图为 AD16C4Tn\_AR = 0x16，预分频设为 2 分频时的计数器时序。

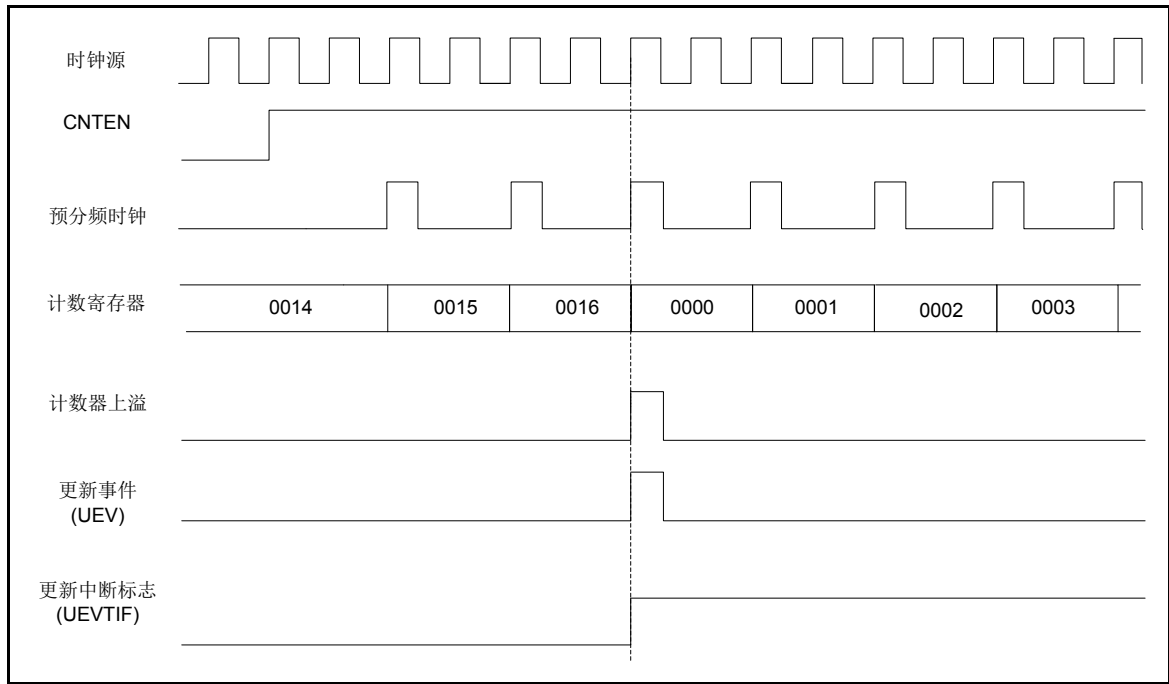


图 24-8 计数器递增计数时序图

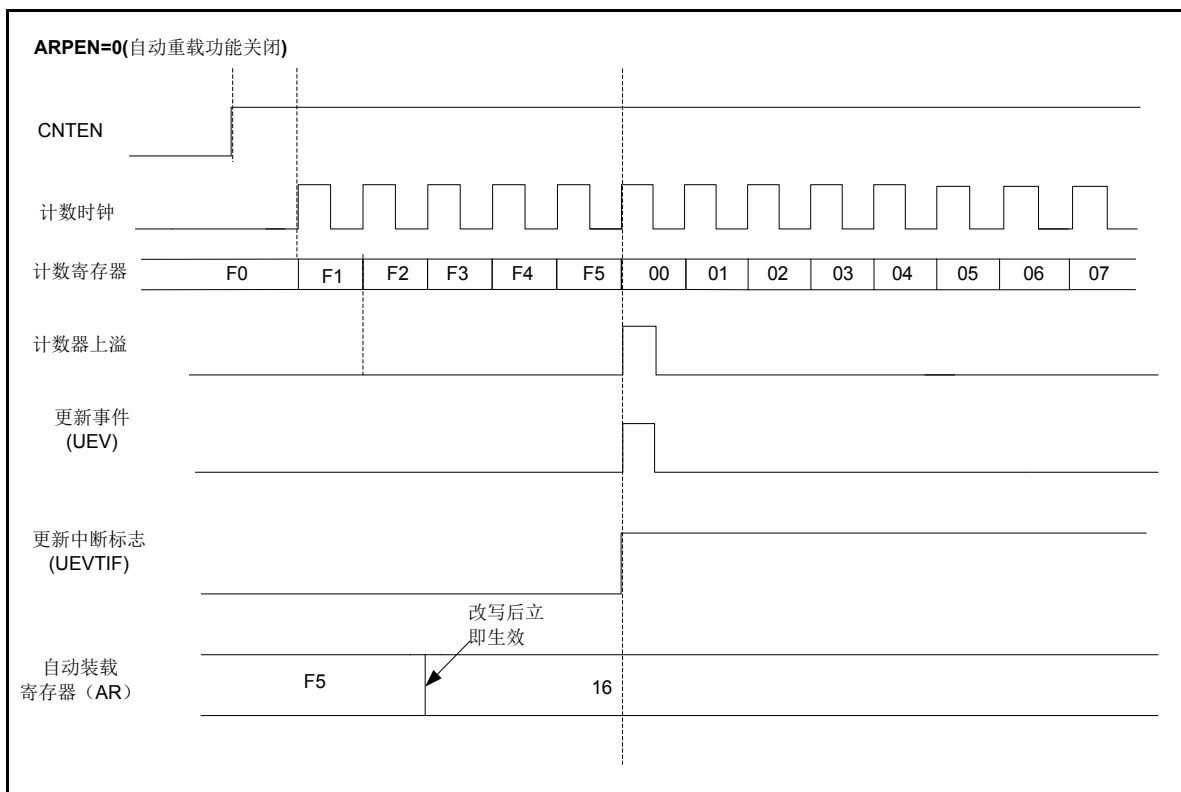


图 24-9 当 ARPEN=0 时计数器时序图

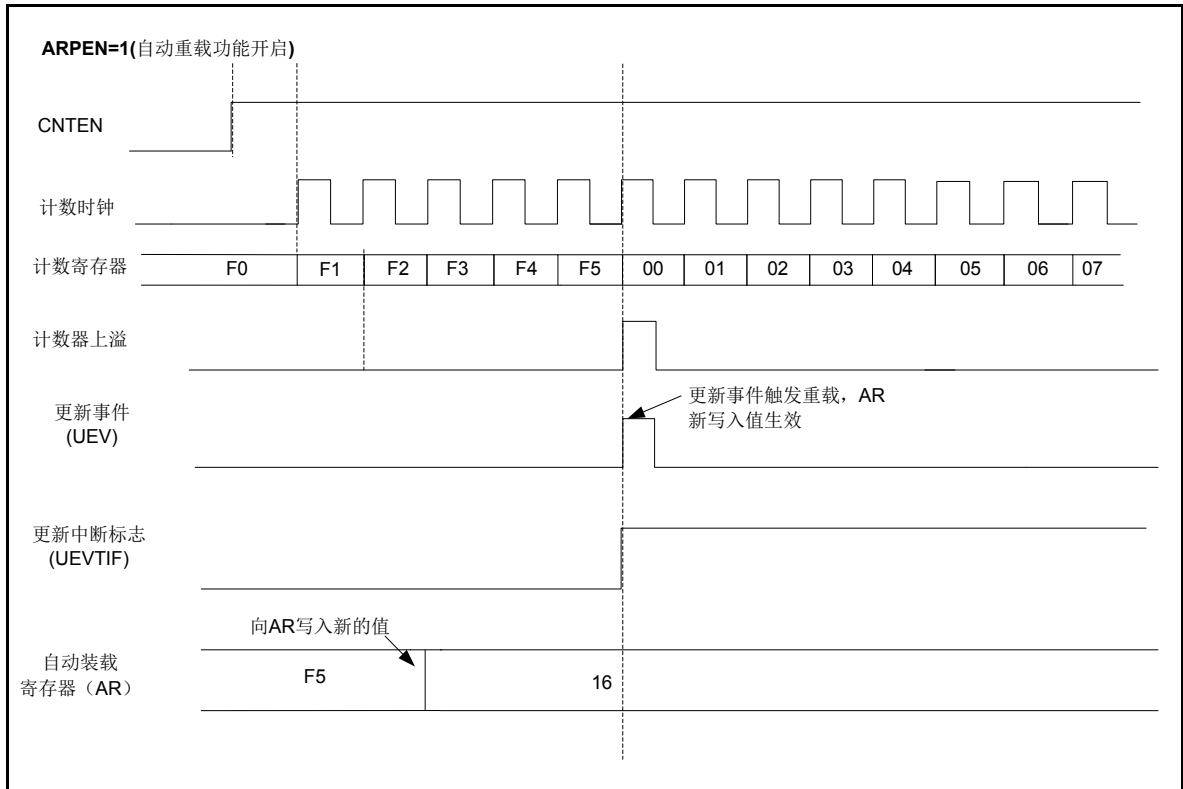


图 24-10 当 ARPEN=1 时计数器时序图

#### 24.4.4.2 递减计数模式

当 AD16C4Tn\_REPAR 寄存器值为 0 时，定时器配置为递减模式，计数器从 AD16C4Tn\_AR 寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 AD16C4Tn\_REPAR 寄存器不为 0 时，则在 AD16C4Tn\_REPAR+1 次后产生更新事件。

置位 AD16C4Tn\_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（AD16C4Tn\_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 AD16C4Tn\_AR = 0x27，预分频设为 1 分频时的计数器时序。

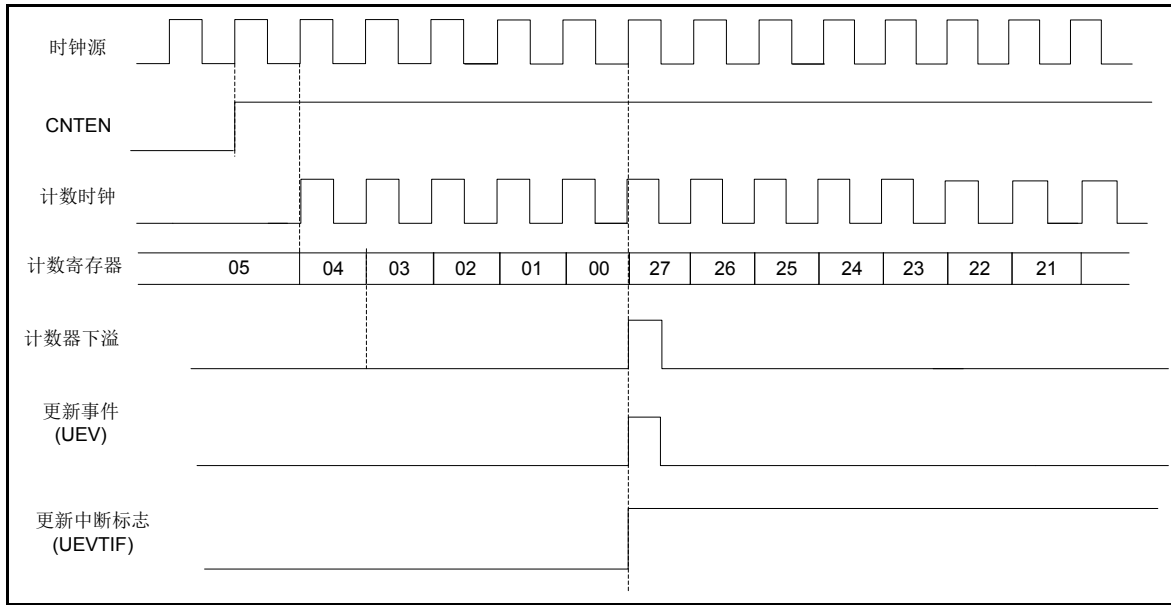


图 24-11 定时器递减计数时序图

#### 24.4.4.3 中心对齐模式

当 AD16C4Tn\_CON1 寄存器的 CMSEL 位的值不等于“00”时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 AD16C4Tn\_AR 寄存器值减 1，并产生更新事件 (UEV)；接着计数器从 AD16C4Tn\_AR 寄存器值递减至 1，并产生更新事件。如此循环计数。计数器递减计数（中心对称模式 1，CMSEL=“01”）、计数器递增计数（中心对称模式 2，CMSEL=“10”）、计数器递增和递减计数（中心对称模式 3，CMSEL=“11”），每个通道的输出比较中断标志位都会置位。

在中心对齐模式下，AD16C4Tn\_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 AD16C4Tn\_EGR 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 AD16C4Tn\_CON1 寄存器中的 DISUE 位可关闭更新事件 (UEV) 的产生。更新事件 (UEV) 关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始(但预分频值没有改变)。此外，若置位 AD16C4Tn\_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件 (UEV)，但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件 (UEV) 产生时，预载寄存器值会更新到影子寄存器，更新标志位 (AD16C4Tn\_RIF 寄存器中的 UEVTIF 位) 置位（取决于 UERSEL 位）。

注：若更新源为计数器上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。



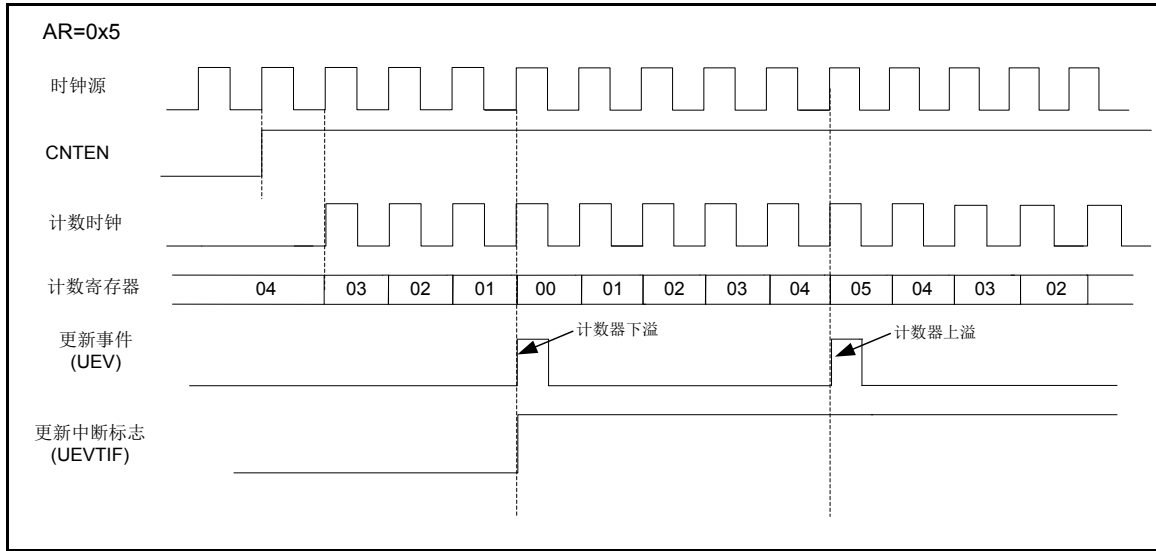


图 24-12 增减计数器时序图

### 24.4.5 捕获/比较通道

下方 3 张图为捕获/比较通道的概述。

输入电路对 In 输入端的信号进行采样，产生一个经过滤波的信号 InF。之后，一个可极性选择的边沿检测器产生 In 边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

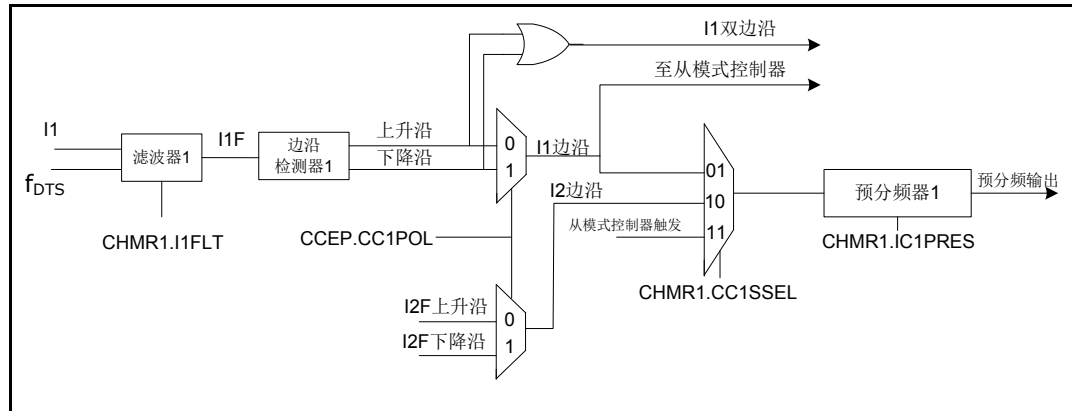


图 24-13 捕获/比较通道

输出部分产生一个中间波形（高有效）作为基准，在输出末端决定最终输出信号的极性。

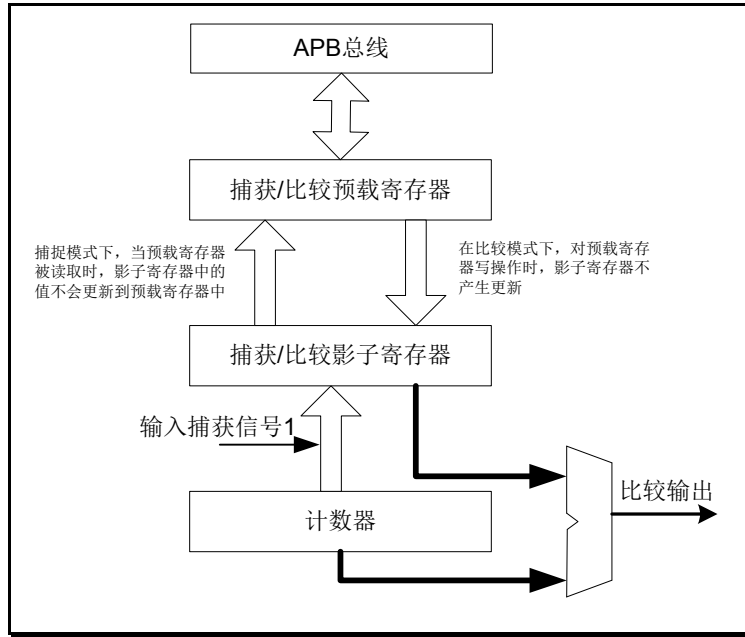


图 24-14 捕获/比较通道 1 结构图

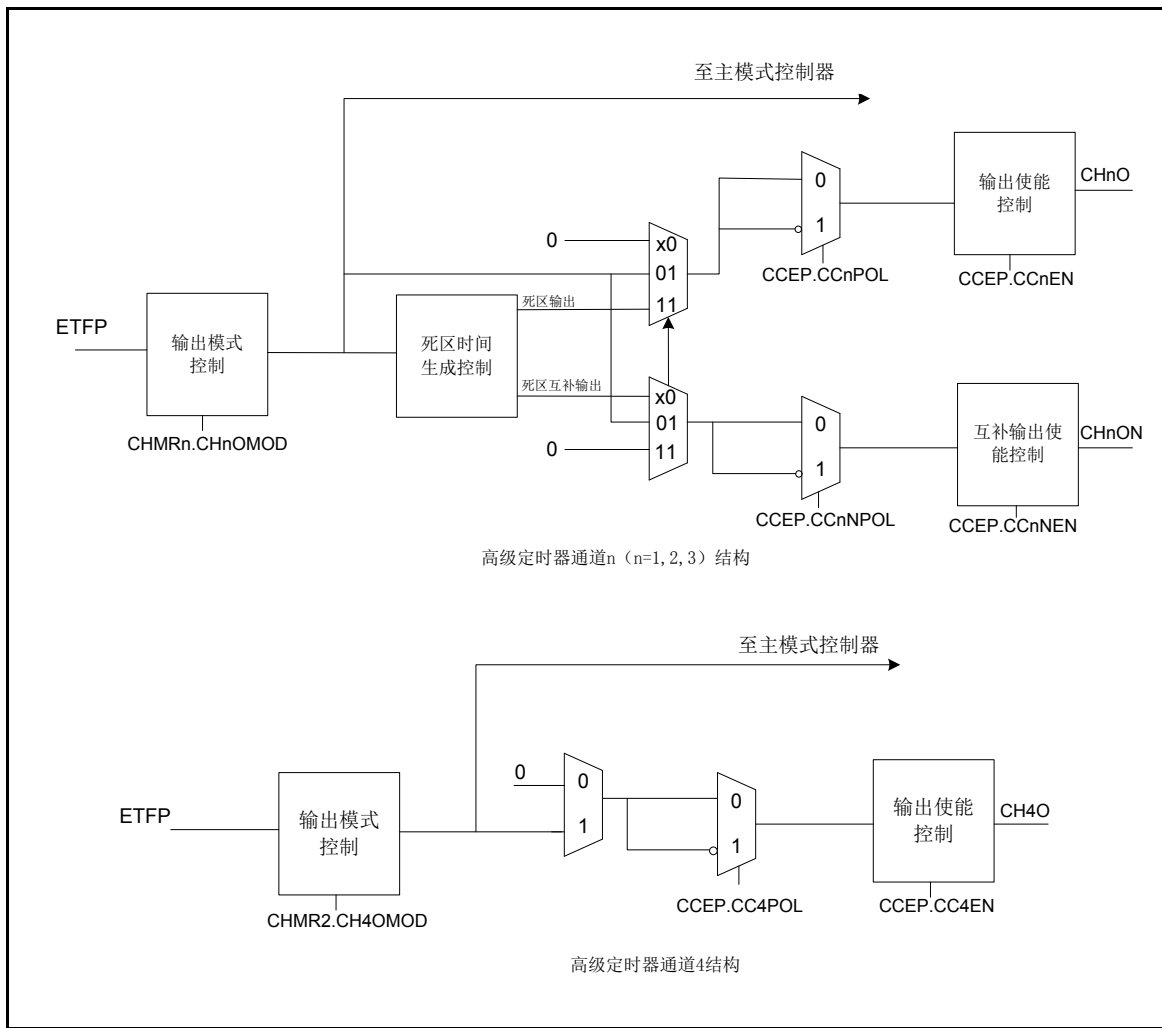


图 24-15 捕获/比较信道的输出部分

#### 24.4.6 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（AD16C4Tn\_CCVALn）寄存器中。当捕获发生时，相应的 CHnIF 标志位（AD16C4Tn\_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnIF 标志位已经置位，则过捕获 CHnOVIF 标志位（AD16C4Tn\_RIF）置位。软件写'0'或读取 AD16C4Tn\_CCVALn 寄存器中的捕获值都可以复位 CHnIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

1. 选择有效输入端：AD16C4Tn\_CCVAL1 必须连接到 I1 输入端，因此需将 AD16C4Tn\_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 AD16C4Tn\_CCVAL1 寄存器为只读。
2. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号内信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
3. 选择 I1 信道的有效边沿变换。AD16C4Tn\_CCEP 寄存器中的 CC1POL 写'0'(上升沿)。
4. 配置输入预分频器。
5. 置位 AD16C4Tn\_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
6. 如有需要，置位 AD16C4Tn\_DIER 寄存器中的 CC1IT 位，使能中断请求。置位 AD16C4Tn\_DIER 寄存器中的 CC1DMA 位，使能 DMA 请求。

当发生输入捕获时：

1. 有效边沿产生，AD16C4Tn\_CCVAL1 寄存器获取计数器的值。
2. CH1IF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
3. 中断的产生取决于 CC1IT 位。
4. DMA 请求的产生取决于 CC1DMA。

为了处理捕获溢出，建议在读取过捕获标志位前先读取捕获数据。这可以避免错过读过捕获标志位之后，读数据之前产生的捕获。

注：捕获中断请求可由软件设置 AD16C4Tn\_SGE 寄存器中 SGCCnE 位产生。

### 24.4.6.1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 AD16C4Tn\_CCVAL1 选择有效的输入：AD16C4Tn\_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。
2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 AD16C4Tn\_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。
3. 为 AD16C4Tn\_CCVAL2 选择有效输入：AD16C4Tn\_CHMR1 寄存器的 CC2SEL 位写"10"（I1 被选择）。
4. 为 I1 边沿检出选择有效极性（用于捕获数据到 AD16C4Tn\_CCVAL2）：CC2POL 位写'1'。
5. 选择有效的触发输入：AD16C4Tn\_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。
6. 配置从机模式控制器为复位模式：AD16C4Tn\_SMCON 寄存器的 SMODS 位写"100"。
7. 使能捕获：AD16C4Tn\_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

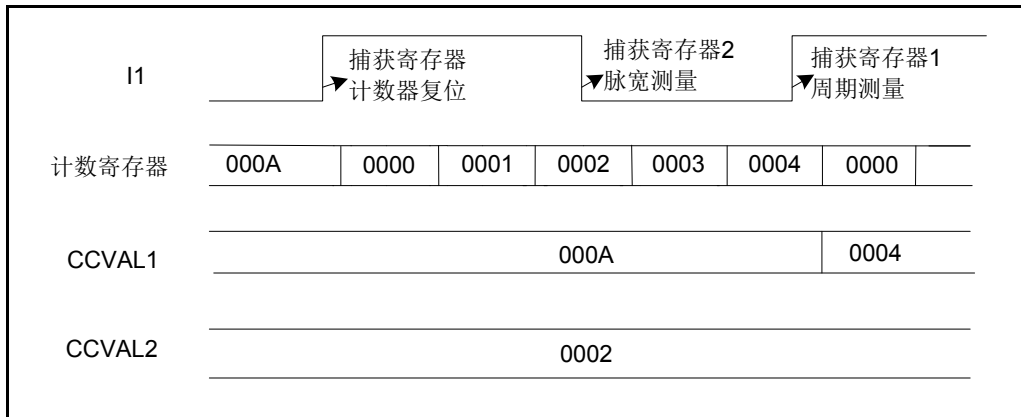


图 24-16 PWM 输入模式时序

### 24.4.7 PWM模式

脉宽调制模式可以产生一个 AD16C4Tn\_AR 寄存器值确定频率，AD16C4Tn\_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），AD16C4Tn\_CHMRn 寄存器的 CHnOCM 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 AD16C4Tn\_CHMRn 寄存器的 OCnPEN 位来使能相应的预载寄存器，最后还需置位 AD16C4Tn\_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 AD16C4Tn\_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 AD16C4Tn\_CCEP 寄存器的 CHnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN、CCnNEN、GOEN、OFFSSI 和 OFFSSR 位

(AD16C4Tn\_CCEP 和 AD16C4Tn\_BDCFG 寄存器) 组合控制。

在 PWM 模式 (1 或 2) 中, AD16C4Tn\_COUNT 和 AD16C4Tn\_CCVALn 寄存器的值会持续的比较, 确定  $AD16C4Tn\_CCVALn \leq AD16C4Tn\_COUNT$  或  $AD16C4Tn\_CCVALn \geq AD16C4Tn\_COUNT$  (取决于计数器的计数方向)。

定时器产生 PWM 波形是边沿对齐或中心对齐, 取决于 AD16C4Tn\_CON1 寄存器的 CMSEL 位。

#### 24.4.7.1 PWM边沿对齐模式

##### 1. 递增计数配置

当 AD16C4Tn\_CON1 寄存器的 DIRSEL 位为低时, 计数器递增计数。

下图给出了 AD16C4Tn\_AR = 8 时的边沿对齐 PWM 波形。

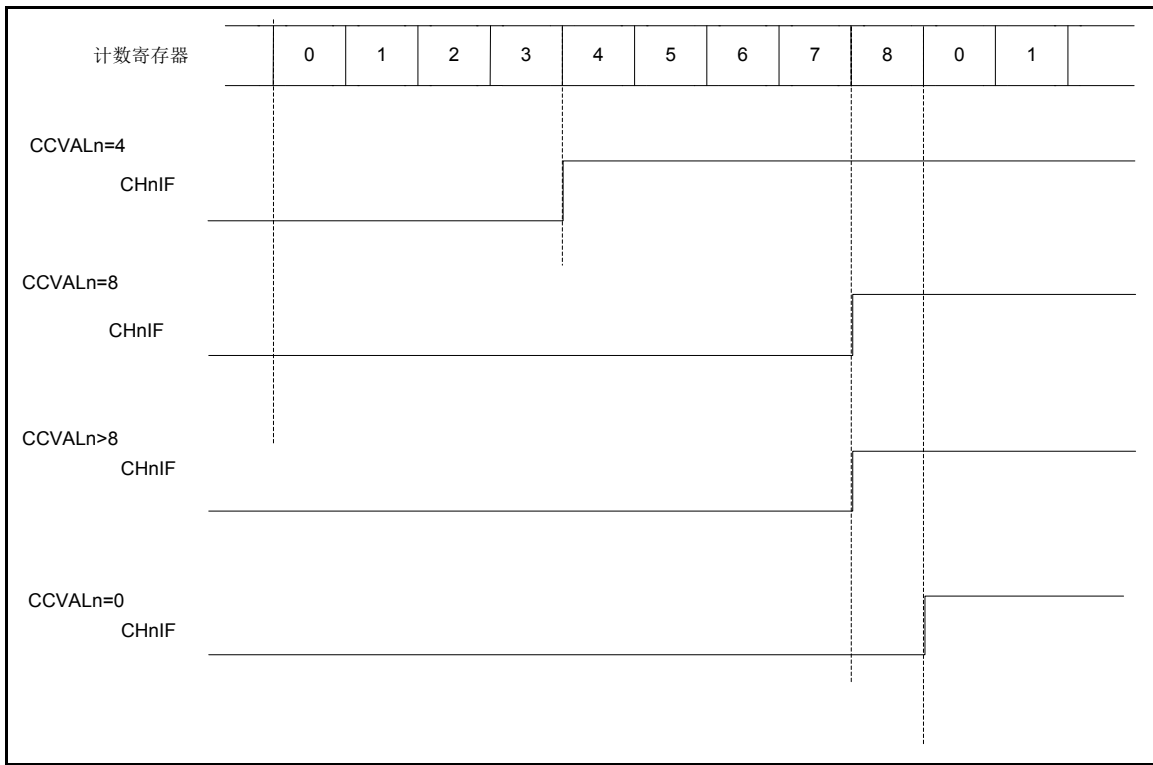


图 24-17 边沿对齐 PWM 波形 (AR=8)

##### 2. 递减计数配置

当 AD16C4Tn\_CON1 寄存器的 DIRSEL 位为高时, 计数器递减计数。

#### 24.4.7.2 PWM中心对齐模式

当 AD16C4Tn\_CON1 寄存器中的 CMSEL 位不为"00"时, 中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位, 取决于 CMSEL 位的配置。AD16C4Tn\_CON1 寄存器的方向位 (DIRSEL) 是由硬件更新的, 软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子:

- ◇ AD16C4Tn\_AR=8
- ◇ PWM 模式 1
  - AD16C4Tn\_CON1 寄存器的 CMSEL= "01", 在中心对齐模式 1 下, 计数器向下计数时会置位比较标志位。

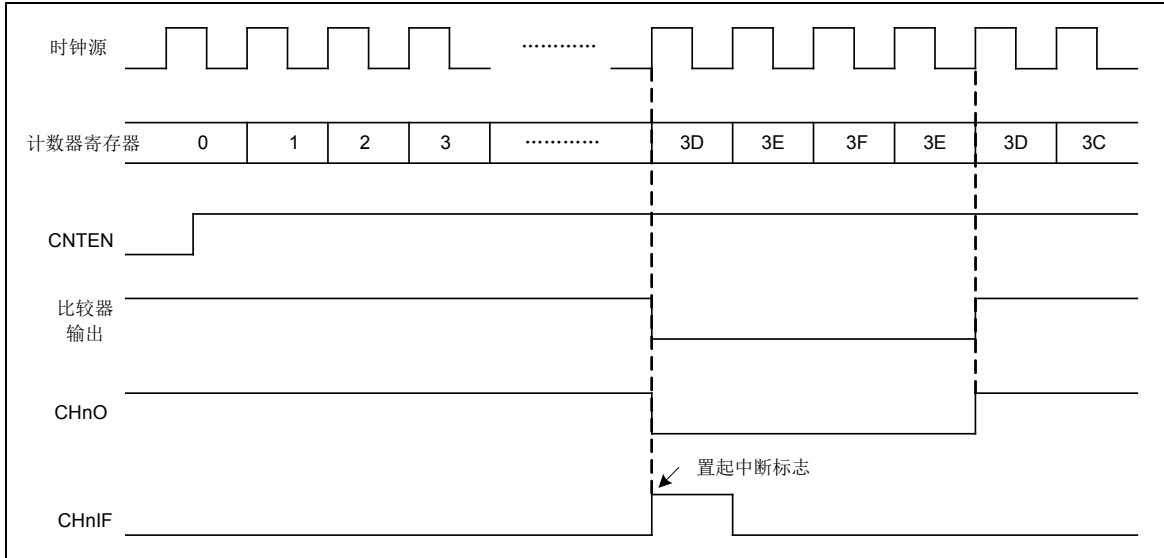


图 24-18 中心对齐 PWM 波形 (AR=8)

中心对齐模式的使用技巧:

- ◇ 当进入中心对齐模式后, 当前递增或递减配置生效。计数器递增或递减计数取决于 AD16C4Tn\_CON1 寄存器的 DIRSEL 位的值。此外, 软件无法对 DIRSEL 和 CMSEL 位同时进行修改。
- ◇ 计数器在中心对齐模式下运行时, 对计数器写操作可能导致不可预知的结果。特别是:
  - 若向计数器入的值大于自动重载值 (AD16C4Tn\_COUNT>AD16C4Tn\_AR), 计数方向不更新。例如, 如果计数器递增计数, 写入值后仍旧递增计数。
  - 若向计数器写 0 或 AD16C4Tn\_AR 中的重载值, 则计数方向更新, 但并没有产生 UEV。
- ◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件 (置位 AD16C4Tn\_SGE 寄存器中的 SGU 位) 且在计数器运行过程中不对计数器写值。

#### 24.4.8 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时, 输出比较功能:

- ◇ 输出比较模式 (AD16C4Tn\_CHMRn 寄存器中的 CHnOCM 位) 和输出极性 (AD16C4Tn\_CCEP 寄存器中的 CHnPOL 位) 的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位 (AD16C4Tn\_RIF 寄存器的 CHnIF 位)。
- ◇ 若相应的中断掩码置位, 则产生中断 (AD16C4Tn\_DIEN 寄存器的 CCnIT 位)。
- ◇ 若相应的使能位置位 (AD16C4Tn\_DIER 寄存器的 CCnDMA 位, AD16C4Tn\_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择), 则发送

DMA 请求。

AD16C4Tn\_CHMRn 寄存器中 OCnPEN 位的值可决定 AD16C4Tn\_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. AD16C4Tn\_AR 与 AD16C4Tn\_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 AD16C4Tn\_DIER 寄存器中的 CCnIT 位。
4. 选择输出模式，例如：
  - CHnOCM = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
  - OCnPEN = '0'，关闭预载寄存器。
  - CHnPOL = '0'，选择有效极性为高。
  - CCnEN = '1'，使能输出。
5. AD16C4Tn\_CON1 寄存器中的 CNTEN 位置位，使能计数器。

假设预载寄存器没有使能（OCnPEN = '0'，否则 AD16C4Tn\_CCVALn 影子寄存器只有在下次更新事件发生时才更新）。通过软件方式，AD16C4Tn\_CCVALn 寄存器的值可随时更新控制输出波形。

#### 24.4.8.1 外部事件清除比较输出

ETFP 输入端（AD16C4Tn\_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

1. 外部触发预分频器应该关闭：AD16C4Tn\_SMCON 寄存器的 ETPSEL[1: 0]位应该写"00"
2. 外部时钟源 2 关闭：AD16C4Tn\_SMCON 寄存器的 ECM2EN 位写'0'
3. 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

#### 24.4.8.2 强制输出模式

在输出模式中（AD16C4Tn\_CHMRn 寄存器中 CCnSSEL = "00"），软件可强制将每个输出比较信号（CHnON/CHnON）改为有效或无效状态，这种修改独立于输出比较寄存器和计数器的比较结果。

为了将某输出比较信号（CHnO）强制为有效状态，需将相应的 AD16C4Tn\_CHMRn 寄存器中 CHnOCM 位写"101"。因此，比较输出被强制为高（高时为有效状态）且 CHnO 的值为 CHnPOL 极性位的相反值。

例如：CCnPOL = '0'（CHnO 高电平有效），则 CHnO 被强制为高电平。

对 AD16C4Tn\_CHMRn 寄存器的 CHnOCM 位写"100", 比较输出可被置低。

无论怎样, AD16C4Tn\_CCVALn 影子寄存器和计数器之间的比较仍然进行, 相应的标志位仍可置位。

### 24.4.9 单脉冲模式

单脉冲模式 (SPMEN) 下, 响应某个触发后, 定时器的输出通道在可配置的延迟时间后产生一个脉冲, 脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 AD16C4Tn\_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时, 单脉冲才可以正确的产生。计数器开始计数前 (定时器等待触发), 必须如下配置:

- ◇ 递增计数:  $CNT < CCVALn \leq AR$  (特别地,  $0 < CCVALn$ )
- ◇ 递减计数:  $CNT > CCVALn$

基于 PWM 模式设置单脉冲输出波形的步骤如下:

- ◇ 设置 AD16C4Tn\_CHMRn 寄存器的 CHnOMOD 位, 选择 PWM 模式 1 或 2;
- ◇ 设置 AD16C4Tn\_CCEPn 寄存器的 CCnPOL 位, 选择通道端口 CHnO 的输出极性;
- ◇ 设置 AD16C4Tn\_CON1 寄存器的 DIRSEL, CMSEL, SPMEN 位, 配置为递增或递减计数, PWM 普通波形模式, 单脉冲模式使能;
- ◇ 设置 AD16C4Tn\_CHMR 寄存器的 CH1OPREN = 1, AD16C4Tn\_CON1 寄存器的 ARPEN = 1, 使能比较寄存器和计数重载寄存器的缓冲功能 (也可以根据实际情况不使能缓冲);
- ◇ 设置 AD16C4Tn\_CCVALn 寄存器和 AD16C4Tn\_AR 寄存器, 配置单脉冲输出延时和脉宽时间;
- ◇ 设置 AD16C4Tn\_SGE 寄存器的 SGU 位来产生一个更新事件;
- ◇ 设置 AD16C4Tn\_CON1 寄存器的 CNTEN=1 来启动计数器, 也可以在触发模式下, 通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

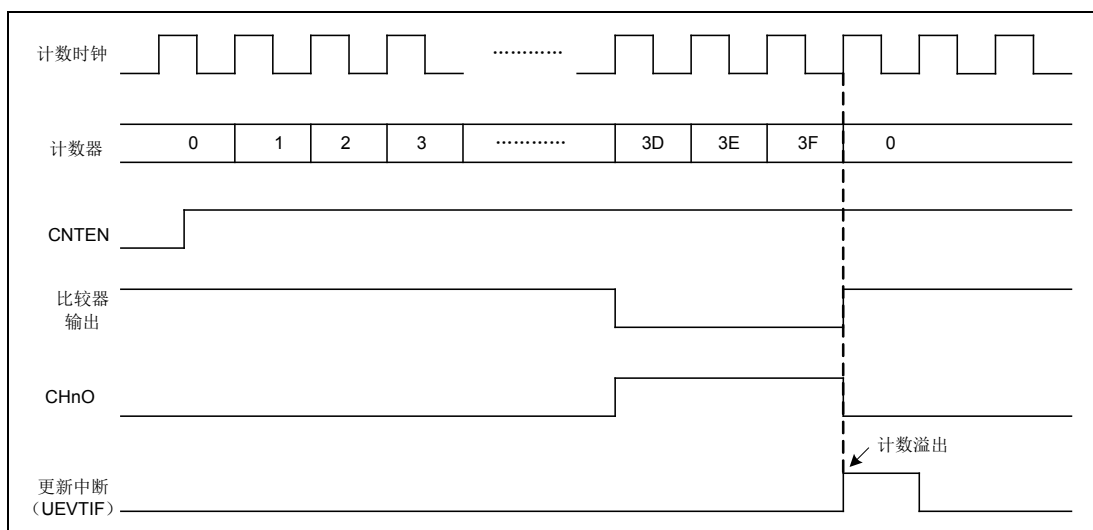


图 24-19 单脉冲模式



### 24. 4. 10 互补输出与死区时间

两个互补的通道输出信号，可以用来控制输出的瞬时开关。死区时间可配置。

每个输出可独立选择输出极性（主输出 CHnO 或互补输出 CHnON），该操作可通过写 AD16C4Tn\_CCEP 寄存器的 CHnPOL 和 CCnNP 位完成。

互补信号 CHnO 和 CHnON 由几个控制位共同控制，分别是 AD16C4Tn\_CCEP 寄存器中的 CCnEN 和 CCnNEN 位, AD16C4Tn\_BDCFG 和 AD16C4Tn\_CON2 寄存器中的 GOEN、OISn、OISnN、OFFSSI 及 OFFSSR 位。特别是死区时间使能后的空闲状态的切换(GOEN 变为 0)。

置位 CCnEN 和 CCnNEN 位，使能死区时间插入，若有刹车电路，同样需要置位 GOEN 位。AD16C4Tn\_BDCFG 寄存器的 DT[7: 0]可以控制所有通道的死区时间的产生。根据比较输出波形，产生 CHnO 和 CHnON 两路输出。若 CHnO 和 CHnON 有效电平为高：

- ◇ CHnO 的输出信号与参考信号一致。上升沿除外，相对参考信号的上升沿，CHnO 输出会有延迟。
- ◇ CHnON 的输出信号与参考信号相反。上升沿除外，相对参考信号的下降沿，CHnON 输出会有延迟。

若延迟时间大于有效输出的宽度（CHnO 或 CHnON），则相应的脉冲不会产生。

下图给出了死区时间输出信号和比较输出波形之间的关系。假设 CHnPOL = 0, CCnNP = 0, GOEN = 1, CCnEN = 1, 和 CCnNEN = 1

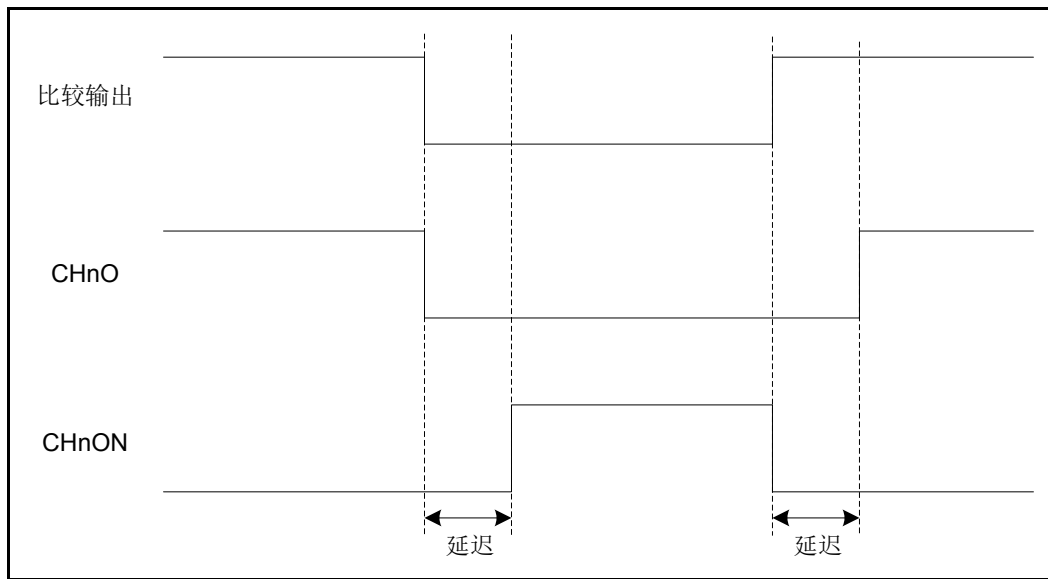


图 24-20 互补输出含死区时间插入

当 PWM 通道配置为互补输出时，如下寄存器控制位都会有缓冲：CHnOMOD、CCnE 和 CCnNE。发生互补通道更新事件时，这些寄存器位才会真正生效，这样就可以预先设置好下一步的配置，并同时对所有互补通道的配置进行更新。互补通道更新事件可以通过设置 AD16C4Tn\_SGE 寄存器的 SGCOM=1 产生，或由触发信号产生（由 AD16C4Tn\_SMCON 寄存器的 TSSEL 位选择触发信号）。

#### 24.4.11 刹车功能

刹车功能模式由以下几个控制位进行设置：AD16C4Tn\_BDCFG 寄存器中的 GOEN、OFFSSI 和 OFFSSR 位，AD16C4Tn\_CON2 寄存器中的 OISn 和 OISnN 位，输出使能信号和无效电平都会被修改。

刹车源可以是刹车输入引脚、时钟失败事件以及软件控制 AD16C4Tn\_SGE 寄存器的 SGBRK 位。时钟失败事件由时钟控制器（CMU）中，时钟安全系统（CSS）产生。时钟安全系统（CSS）详细信息可参考时钟安全系统章节。

系统复位后，刹车电路被禁止且 GOEN 位被复位。置位 AD16C4Tn\_BDCFG 寄存器的 BRKEN 位可使能刹车功能，同样寄存器中，BRKEN 位可选择刹车输入信号的极性。BRKEN 和 BRKP 位可同时修改。对 BRKEN 和 BRKP 位写操作后，1 个 APB 时钟周期延时后写入值才会生效。因此，写操作后，需等待 1 个 APB 时钟周期后才能正确读回写入值。

由于 GOEN 的下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（AD16C4Tn\_BDCFG 寄存器中）之间插入了一个同步电路。这也导致了异步和同步信号之间会产生一些延迟。特别是 GOEN 之前为低时对 GOEN 写 1 操作后，要读取正确值，必须先插入一个延时（空指令）。这是因为写入的是异步信号，而读取的是同步信号。

当发生刹车请求时（刹车输入端有刹车电平）：

- ◇ GOEN 位被异步清除，输出端进入无效状态，空闲状态或复位状态（OFFSSI 位选择）。即使 MCU 的振荡器关闭，该功能仍然有效。
- ◇ 一旦 GOEN=0，每个通道输出预先配置的电平。AD16C4Tn\_CON2 寄存器中的 OISn 位配置该电平。如果 OFFSSI=0，则定时器释放使能输出，否则使能输出一直为高。
- ◇ 当使用互补输出时：
  - 输出端首先被置于复位状态，无效状态（取决于极性）。这个过程异步的，即使定时器没有时钟也有效。
  - 如果定时器时钟仍然存在，则死区时间生成器会重新生效，这样在死区时间后，OISSn 和 OISSnN 位的配置电平可驱动输出。这种情况下，CHnO 和 CHnON 无法驱动输出端都为有效电平。
  - 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。注，由于对 GOEN 的重新同步，死区时间的周期会比通常情况下长一些（大约 2 个 TIMER 模块时钟周期）。
  - 如果 OFFSSI = 0，则定时器释放使能输出，否则使能输出保持或变高（一旦 CCnEN 和 CCnNEN 有一个变高时）。
- ◇ 当刹车状态标志位（AD16C4Tn\_RIF 寄存器中的 BRKIF 位）置位时，若 AD16C4Tn\_DIER 寄存器中的 BRKIT 位置位，可触发中断；若 AD16C4Tn\_DIER 寄存器中的 BDE 位置位，可触发 DMA 请求。
- ◇ 当 AD16C4Tn\_BDCFG 寄存器中的 AOEN 位置位时，在下次更新事件（UEV）发生时，GOEN 位会自动置位。例如，该功能可用来整形。否则，GOEN 位会保持为低，直到对其写'1'操作，该特性可用于安全方面的应用，可以将刹车输入端接到一个电源驱动的报警端、热敏传感器或其他安全器件上。

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

除刹车输入和输出管理，为保证应用程序的安全，内部刹车电路具有写保护功能。用户可冻结几个配置参数（死区时间，CHnO/CHnON 极性和失能时状态，CHnOCM 配置，刹车使能和极性）。通过 AD16C4Tn\_BDCFG 寄存器中的 LOCKLVL 位，可从三个保护等级中选择一种保护等级。MCU 复位后，LOCKLVL 位只能写一次。

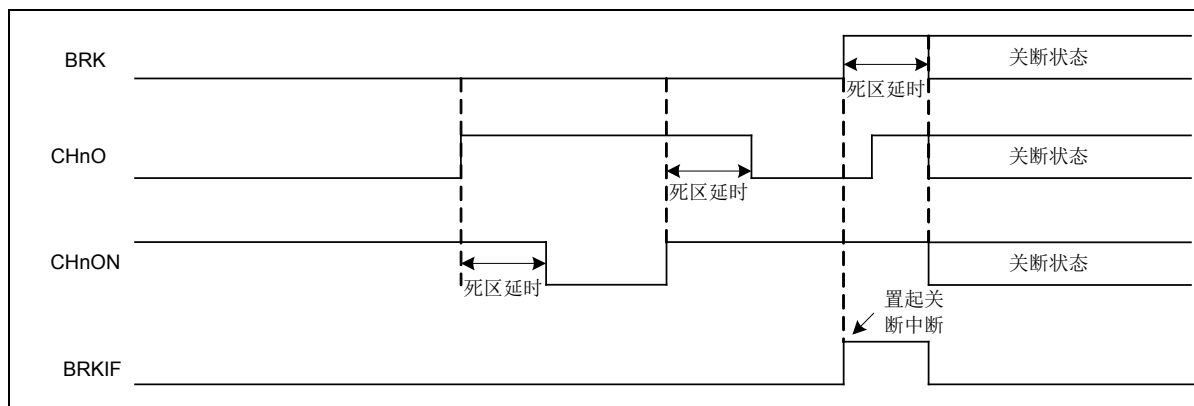


图 24-21 刹车输出行为

### 24. 4. 12 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 AD16C4Tn\_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 AD16C4Tn\_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 AD16C4Tn\_SMCON 寄存器中的 SMODS = "011"。

配置 AD16C4Tn\_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1\_IN 和 CH2\_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，AD16C4Tn\_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 AD16C4Tn\_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 AD16C4Tn\_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平 (I1 滤波信号对应 I2,I2 滤波信号对应 I1)	I1 滤波信号		I2 滤波信号	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 24-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部逻辑接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号了产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。

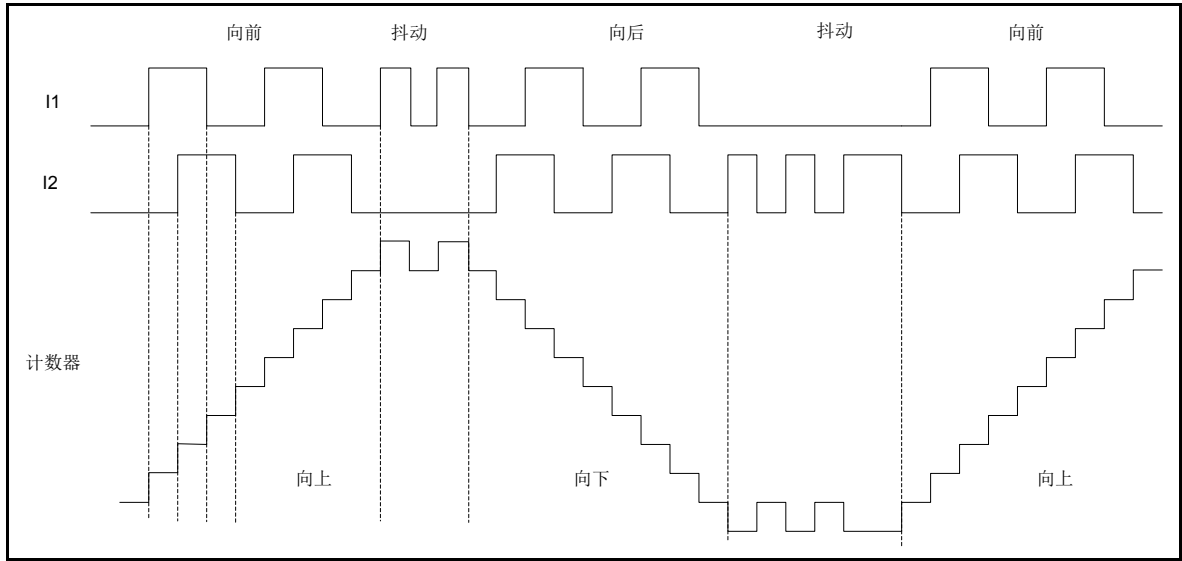


图 24-22 编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

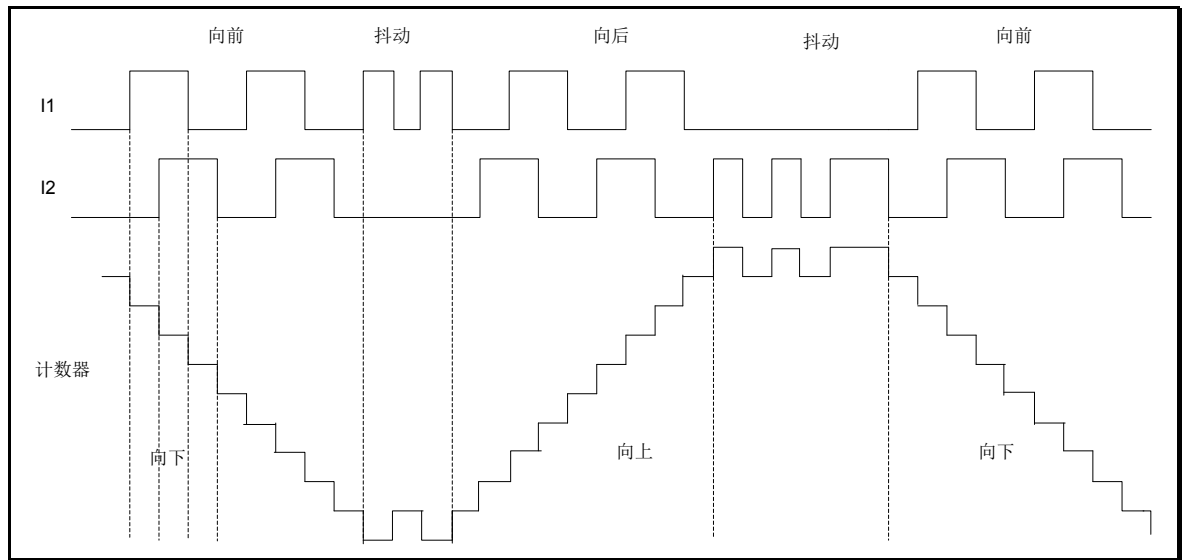


图 24-23 I1 滤波后极性反相时编码器接口例子

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时常获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

### 24.4.13 输入异或功能

通过 AD16C4Tn\_CON2 寄存器中 I1SEL 位，可将通道 1 的输入滤波器连接到 XOR 门的输出端，XOR 门联合了 CH1\_IN、CH2\_IN 和 CH3\_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能，如触发或输入捕获。该功能参见下节的霍尔传感器接口。

### 24.4.14 霍尔传感器接口

高级控制定时器产生 PWM 信号驱动电机，另外一个定时器作为“接口定时器”。“接口定时器”捕获 3 个引脚输入 (CH1\_IN, CH2\_IN, CH3\_IN)，引脚信号经过一个异或门后连接到 I1 的输入通道 (置位 AD16C4Tn\_CON2 寄存器中的 I1SEL 位选择)。

从模式控制器配置为复位模式，3 个输入后的异或信号作为从输入。因此，只要 3 个输入信号有一个变化，计数器重新从 0 开始计数。霍尔输入端的任何变化都可触发创建一个时钟基准。

### 24.4.15 外部触发的同步

AD16C4Tn 定时器可在多种模式下与外部触发同步：复位模式、门控模式及触发模式。

#### 24.4.15.1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外，若 AD16C4Tn\_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器 (AD16C4Tn\_AR, AD16C4Tn\_CCVALn) 都会因更新事件 UEV 而被更新。

在下面例子中，I1 输入端的上升沿让递增计数被清空：

- ◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期 (本例无需滤波器，故 I1FLT = "0000")。触发捕获分频器没有使用，无需配置。CC1SSEL 位只选择输入捕获源，AD16C4Tn\_CHMR1 寄存器中 CC1SSEL = "01"。AD16C4Tn\_CCEP 寄存器中 CC1POL = 0 以确定极性 (只检测上升沿)。
- ◇ 定时器配置位复位模式：AD16C4Tn\_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源：AD16C4Tn\_SMCON 寄存器中 TSSEL = "101"。
- ◇ 启动计数器：AD16C4Tn\_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数，正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时，计数器会被清零且从 0 重新开始计数。同时，标志位置位 (AD16C4Tn\_RIF 寄存器中 TRGIF 位)，如果中断及 DMA 使能 (取决于 AD16C4Tn\_DIER 寄存器中的 TRGIT 和 TRGDMA 位)，会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 AD16C4Tn\_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路，I1 上的上升沿和计数器实际复位之间会存在延时。

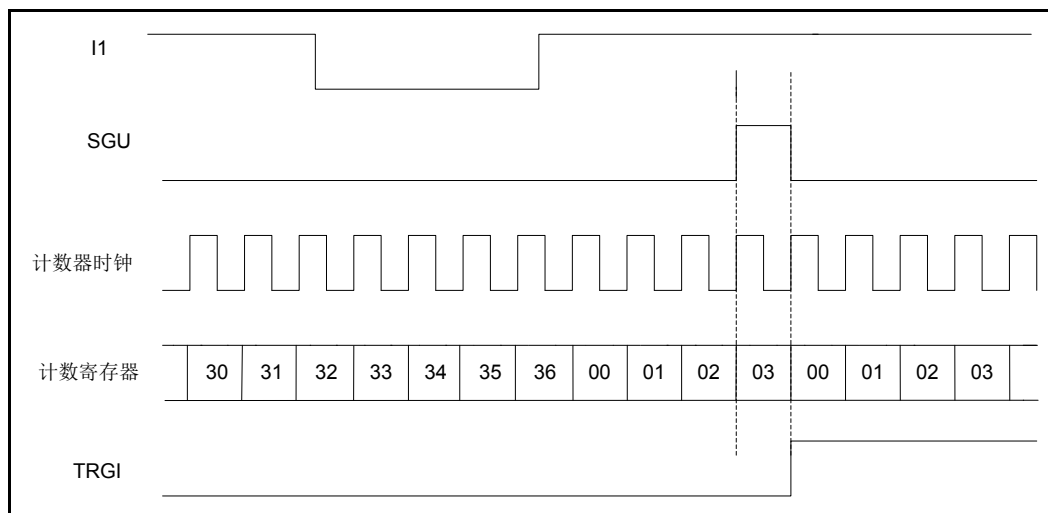


图 24-24 复位模式控制电路

#### 24.4.15.2 门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

- ◇ 配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn\_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。AD16C4Tn\_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。
- ◇ 配置定时器为门控模式：AD16C4Tn\_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：AD16C4Tn\_SMCON 寄存器中 TSSEL = "101"。
- ◇ 使能计数器：AD16C4Tn\_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

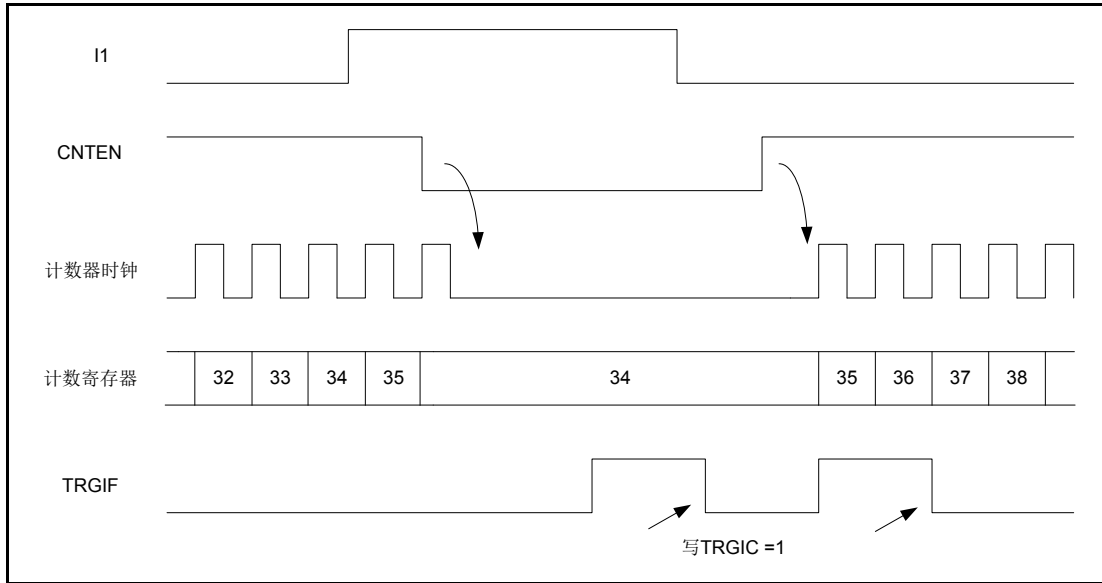


图 24-25 门控模式控制电路

### 24.4.15.3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

- ◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。AD16C4Tn\_CHMR1 寄存器中 CC2SEL = "01",用于选择捕获源。AD16C4Tn\_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。
- ◇ 配置定时器为触发模式：AD16C4Tn\_SMCON 寄存器中 SMODS = "110"。AD16C4Tn\_SMCON 寄存器中 TSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因，I2 上出现上升沿和计数器实际停止之间会有一定的延时。

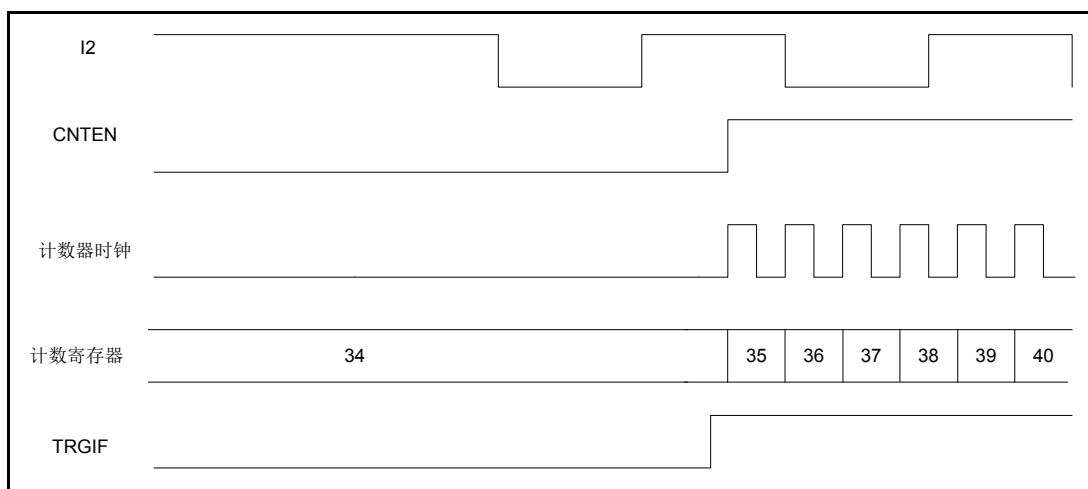


图 24-26 触发模式控制电路



#### 24.4.15.4 选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 AD16C4Tn\_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

- ◇ 通过 AD16C4Tn\_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000": 无滤波

ETPSEL = "00": 禁止分频

ETPOL = '0': 检测 ET 的上升沿，ECM2EN = '1'使能外部时钟模式 2

- ◇ 配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000": 无滤波。

触发捕获分频器没有使用，无需配置。

AD16C4Tn\_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，

AD16C4Tn\_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

- ◇ 配置定时器为触发模式：AD16C4Tn\_SMCON 寄存器中 SMODS = "110"。

AD16C4Tn\_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETRP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

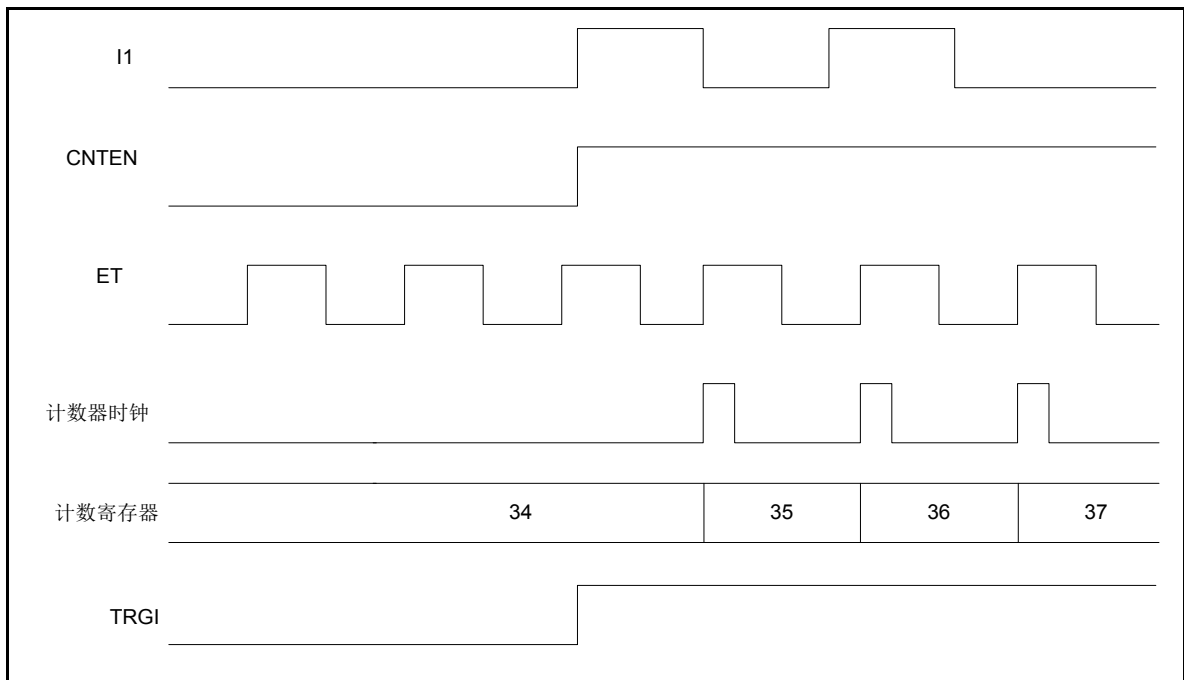


图 24-27 外部时钟源 2+触发模式下的控制电路

### 24.4.16 调试模式

当微控制器进入调试模式（Cortex™-M 内核终止），AD16C4Tn 计数器可停止计数。

### 24.4.17 6 步PWM生成

当通道配置为互补输出时，CHnOMOD, CCnEN 及 CCnNEN 位都是预装载位。发生换相事件时，预装载位传递给寄存器的影子位。因此，可预先配置下一步，同时也可修改所有通道的配置。软件置位 AD16C4Tn\_SGE 寄存器的 COM 位可以产生 COM 换相事件，硬件 TRGI 上升沿也可产生。

COM 事件发生时标志位（AD16C4Tn\_IFM 寄存器的 COMIM 位）置位可触发中断（AD16C4Tn\_IVS 寄存器的 COMIT 位置位）或 DMA 请求（AD16C4Tn\_DMAEN 寄存器的 COMDMA 位置位）。

下图描述了当发生 COM 事件时，3 种不同配置，CHnO 和 CHnON 输出情况。

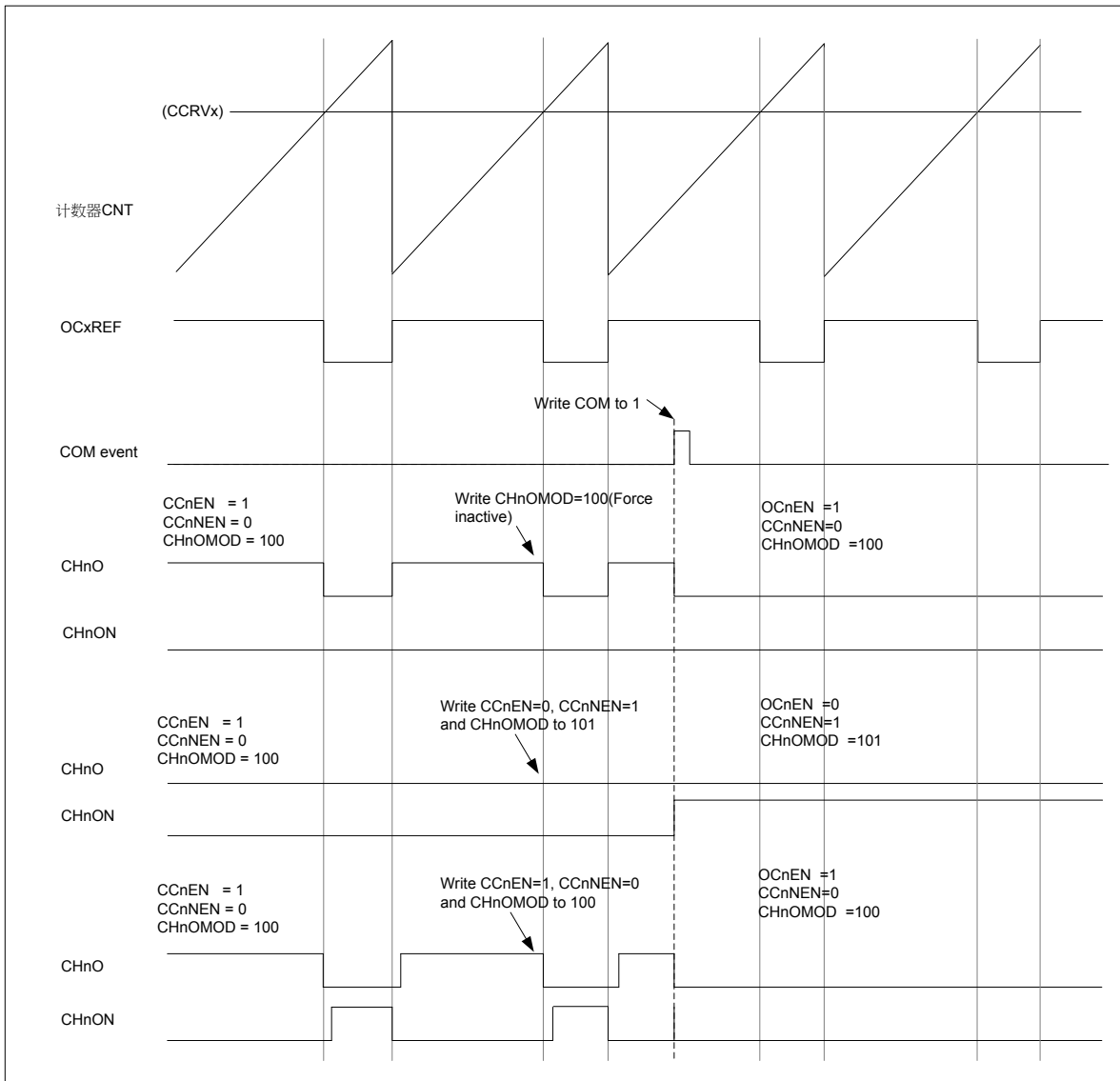


图 24-28 6 步 PWM 波形示例

## 24.5 特殊功能寄存器

### 24.5.1 寄存器列表

TIM2_3 寄存器列表		
寄存器名称	偏移地址	寄存器描述
TIM2 基地址: 4000_0800 <sub>H</sub>		
TIM3 基地址: 4000_0C00 <sub>H</sub>		
AD16C4Tn_CON1	000 <sub>H</sub>	控制寄存器 1
AD16C4Tn_CON2	004 <sub>H</sub>	控制寄存器 2
AD16C4Tn_SMCON	008 <sub>H</sub>	从模式控制寄存器
AD16C4Tn_IER	00C <sub>H</sub>	中断使能寄存器
AD16C4Tn_IDR	010 <sub>H</sub>	中断禁止寄存器
AD16C4Tn_IVS	014 <sub>H</sub>	中断有效状态寄存器
AD16C4Tn_RIF	018 <sub>H</sub>	原始中断标志寄存器
AD16C4Tn_IFM	01C <sub>H</sub>	中断标志屏蔽寄存器
AD16C4Tn_ICR	020 <sub>H</sub>	中断清零寄存器
AD16C4Tn_SGE	024 <sub>H</sub>	事件生成寄存器
AD16C4Tn_CHMR1	028 <sub>H</sub>	捕获/比较模式寄存器 1
AD16C4Tn_CHMR2	02C <sub>H</sub>	捕获/比较模式寄存器 2
AD16C4Tn_CCEP	030 <sub>H</sub>	捕获/比较使能寄存器
AD16C4Tn_COUNT	034 <sub>H</sub>	计数器
AD16C4Tn_PRES	038 <sub>H</sub>	预分频器
AD16C4Tn_AR	03C <sub>H</sub>	自动重载寄存器
AD16C4Tn_REPAR	040 <sub>H</sub>	重复计数寄存器
AD16C4Tn_CCVAL1	044 <sub>H</sub>	捕获/比较寄存器 1
AD16C4Tn_CCVAL2	048 <sub>H</sub>	捕获/比较寄存器 2
AD16C4Tn_CCVAL3	04C <sub>H</sub>	捕获/比较寄存器 3
AD16C4Tn_CCVAL4	050 <sub>H</sub>	捕获/比较寄存器 4
AD16C4Tn_BDCFG	054 <sub>H</sub>	刹车和死区时间寄存器
AD16C4Tn_DMAEN	058 <sub>H</sub>	DMA 使能寄存器

## 24.5.2 寄存器描述

### 24.5.2.1 控制寄存器 1 (AD16C4Tn\_CON1)

控制寄存器 1 (AD16C4Tn_CON1)																																
偏移地址: 000 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																DBGSEL	Reserved	OCCISS				OCCISP	DFCKSEL		ARPEN	CMSEL		DIRSEL	SPMEN	URSEL	DISUE	CNTEN

Reserved	Bit 31-16	-	保留, 必须保持为复位值
DBGSEL	Bit 15	RW	调试时通道输出状态选择: 0: 通道输出为高阻态 1: 通道输出保持
Reserved	Bit 14	-	保留, 必须保持为复位值
OCCISS	Bit 13-11	RW	通道输出清除内部触发源选择: 000: 无效 001: 清除触发源通道 0 010: 清除触发源通道 1 011: 清除触发源通道 2 100: 清除触发源通道 3
OCCISP	Bit 10	RW	通道输出清除内部触发源极性: 0: 低电平有效 1: 高电平有效
DFCKSEL	Bit 9-8	RW	死区发生器和数字滤波器工作时钟频率 <b>Fdfck</b> 选择位 该位为计数器时钟 (INT_CLK) 频率, 显示了死区时间和由死区发生器与数字滤波器 (ET, In) 所用的采样时钟 ( $t_{DTS}$ ) 之间的分频关系。 00: $t_{DTS}=t_{INT\_CLK}$ 01: $t_{DTS}=2*t_{INT\_CLK}$ 10: $t_{DTS}=4*t_{INT\_CLK}$ 11: 预留, 不允许编程该值。
ARPEN	Bit 7	RW	计数器自动重载寄存器预载 0: AD16C4Tn_AR 寄存器未缓冲 1: AD16C4Tn_AR 寄存器被装入缓冲器
CMSEL	Bit 6-5	RW	中心对齐模式选择 00: 边沿对齐模式。计数器根据方向为 (DIRSEL) 来向上或向下计数。 01: 中央对齐模式 1。计数器以交替方式向上或向下计数。仅当计数器向下计数时, 配置为输出的通道 (AD16C4Tn_CHMRn 寄存器中

			<p>CCnSSEL=00) 的输出比较中断标志位才会被设置。</p> <p><b>10: 中央对齐模式 2。</b>计数器以交替方式向上或向下计数。仅当计数器向上计数时, 配置为输出的通道 (AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00) 的输出比较中断标志位才会被设置。</p> <p><b>11: 中央对齐模式 3。</b>计数器以交替方式向上或向下计数。当计数器向上或向下计数时, 配置为输出的通道 (AD16C4Tn_CHMRn 寄存器中 CCnSSEL=00) 的输出比较中断标志位均会被设置。</p> <p>注意: 当计数器使能时 (CNTEN=1), 不允许从边沿对齐模式转换到中央对齐模式。</p>
DIRSEL	Bit 4	R/W	<p><b>计数器方向选择</b></p> <p>0: 计数器向上计数</p> <p>1: 计数器向下计数</p> <p>注意: 当计数器配置为中央对齐模式或者编码器模式时, 该位只读。</p>
SPMEN	Bit 3	R/W	<p><b>使能单脉冲模式</b></p> <p>0: 当发生更新事件时, 计数器不停止。</p> <p>1: 当发生下一次更新事件 (CNTEN 位清零) 时, 计数器停止。</p>
UERSEL	Bit 2	R/W	<p><b>选择更新事件请求</b></p> <p>该位由软件置 1 或清零, 来选择 UEV 事件源。</p> <p>0: 如果更新中断或 DMA 请求使能, 则下述任一时间都可产生更新中断或 DMA 请求:</p> <ul style="list-style-type: none"> <li>- 计数器上溢/下溢</li> <li>- 设置 SGU 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果更新中断或 DMA 请求使能, 仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断</p>
DISUE	Bit 1	R/W	<p><b>禁止更新事件</b></p> <p>该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。</p> <p>0: UEV 使能。更新事件 (UEV) 由下列任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器上溢/下溢</li> <li>- 设置 SGU 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>缓冲寄存器载入他们的预载值。</p> <p>1: UEV 禁止。不产生更新事件, 影子寄存器保持他们的值 (ARRV, PSCV, CCRVx)。如果从模式控制器接收到硬件复位, 计数器和预分频器将</p>

			被重新初始化。
CNTEN	Bit 0	R/W	<p><b>使能计数器</b></p> <p>0: 计数器禁止</p> <p>1: 计数器使能</p> <p>注意：如果软件设置了 CNTEN 位，外部时钟，门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。</p>

### 24.5.2.2 控制寄存器 2 (AD16C4Tn\_CON2)

控制寄存器 2 (AD16C4Tn_CON2)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OISS4	OISS3N	OISS3	OISS2N	OISS2	OISS1N	OISS1	I1FSEL	TRGOSEL			CCDMASEL		CCUSEL	Reserved	CCPCEN		

Reserved	Bit 31-15	-	保留, 必须保持为复位值
OISS4	Bit 14	R/W	通道 4 输出的空闲状态选择位 参考 OISS1 描述
OISS3N	Bit 13	R/W	通道 3 互补输出的空闲状态选择位 参考 OISS1N 描述
OISS3	Bit 12	R/W	通道 3 输出的空闲状态 3 选择位 参考 OISS1 描述
OISS2N	Bit 11	R/W	通道 2 互补输出的空闲状态选择位 参考 OISS1N 描述
OISS2	Bit 10	R/W	通道 2 输出的空闲状态选择位 参考 OISS1 描述
OISS1N	Bit 9	R/W	通道 1 互补输出的空闲状态选择位 0: 当 GOEN=0, 在一段死区时间后, CH1ON=0 1: 当 GOEN=0, 在一段死区时间后, CH1ON=1 注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后, OISS1N 不可更改。
OISS1	Bit 8	R/W	通道 1 输出的空闲状态选择位 0: 当 GOEN=0, 如果 CH1ON 已实现, 在一段死区时间后, CH1O=0 1: 当 GOEN=0, 如果 CH1ON 已实现, 在一段死区时间后, CH1O=1 注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 1,2,或 3 后, OISS1N 不可更改。
I1FSEL	Bit 7	R/W	选择 I1 引脚功能 0: AD16C4Tn_CH1 引脚与 I1 输入连接 1: AD16C4Tn_CH1, CH2 和 CH3 引脚与 I1 输入 (XOR) 连接。
TRGOSEL	Bit 6-4	R/W	选择主模式 TRGOUT 输出 为同步 (TRGOUT), 该位可选择在主模式下发送至从计数器的信息。 000: 复位-AD16C4Tn_SGE 寄存器中的 SGU 位

			<p>被采用为触发输出 (TRGOUT)。如果复位由触发输入生成 (从模式控制器配置复位模式), 则相较于实际复位, TRGOUT 上的信号将会延迟。</p> <p><b>001: 使能</b>-计数器使能信号被用作触发输出 (TRGOUT)。在从计数器使能的情况下, 该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制, TRGOUT 上会产生延迟, 除非被选为主/从模式 (参考 AD16C4Tn_SMCON 寄存器中的 MSCFG 位的描述)。</p> <p><b>010: 更新</b>-更新事件被选为触发输出 (TRGOUT)。举例, 主计数器可被用作从计数器的预分频器。</p> <p><b>011: 比较脉冲</b>-一旦捕获或者比较匹配发生, 当 CH1CCIF 标志位被置起 (即便已为高电平), 触发输出会发送一个正脉冲。</p> <p><b>100: 比较</b>- 通道 1 比较输出信号用作触发输出 TRGOUT</p> <p><b>101: 比较</b>- 通道 2 比较输出信号用作触发输出 TRGOUT</p> <p><b>110: 比较</b>- 通道 3 比较输出信号用作触发输出 TRGOUT</p> <p><b>111: 比较</b>- 通道 4 比较输出信号用作触发输出 TRGOUT</p>
CCDMASEL	Bit 3	R/W	<p><b>使能捕获/比较 DMA</b></p> <p>0: 当 CCn 事件发生, 会发出 CCn DMA 请求。 1: 当发生更新时间, 会发出 CCn DMA 请求。</p>
CCUSEL	Bit 2	R/W	<p><b>选择捕获比较更新</b></p> <p>0: 当捕获/比较控制位为预载值 (CCPCEN=1), 则当设置 SGCOM 位时才会被更新。 1: 当捕获/比较控制位为预载值 (CCPCEN=1), 则当设置 SGCOM 位或者当 TI 边沿上升时, 均会被更新。 注意: 该位只有用作于通道时才有互补输出。</p>
Reserved	Bit 1	-	<p><b>保留, 必须保持为复位值</b></p>
CCPCEN	Bit 0	R/W	<p><b>使能捕获比较预载控制</b></p> <p>0: CCnEN, CCnNEN 和 CHnOMOD 不为预载值。 1: CCnEN, CCnNEN 和 CHnOMOD 为预载值。 当写入预载值后, 仅当发生换相事件 (COM) 时 (SGCOM 位设置或者 TI 上检测到上升沿由 CCUSEL 位决定), 才会被更新。 注意: 该位只有用作于通道时才有互补输出。</p>



### 24.5.2.3 从模式控制寄存器 (AD16C4Tn\_SMCON)

从模式控制寄存器 (AD16C4Tn_SMCON)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL	ETFLT				MSCFG	TSSEL			OCCS	SMODS			

Reserved	Bit 31-16	-	保留, 必须保持为复位值
ETPOL	Bit 15	R/W	<b>选择外部触发信号边沿</b> 0: 正向 ET, 高电平有效或上升沿有效。 1: 反正 ET, 低电平有效或下降沿有效。
ECM2EN	Bit 14	R/W	<b>使能外部时钟模式 2</b> 该位使能外部时钟模式 2 0: 禁止外部时钟模式 2 1: 使能外部时钟模式 2。计数器由 ETFP 信号上的有效边沿计数。 注意: 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETFP 相连接 (SMODS=111 和 TSSEL=111) 具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式: 复位模式, 门控模式和除法模式。在这种情况下, TI 不能与 ETFP 相连接 (TSSEL 不能设置为 111)。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能, 外部时钟输入为 ETFP。
ETPSEL	Bit 13-12	R/W	<b>选择外部触发分频</b> 外部触发信号频率最大为 AD16C4Tn CLK 频率的 1/4。可使能预分频器来减小 ETFP 频率。该位有效用于输入高速外部时钟的情况。 00: 预分频器关闭 01: ETFP 频率 2 分频 10: ETFP 频率 4 分频 11: ETFP 频率 8 分频
ETFLT	Bit 11-8	R/W	<b>选择外部触发滤波采样时钟</b> 该位定义了对 ET 信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成, 每 N 个连续事件才视为一个有效边沿。 0000: 无滤波器, 采样频率为 $f_{DTS}$ 0001: $f_{SAMPLING} = f_{INT\_CLK}, N = 2$

			<p>0010: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}}, N = 4</math>          0011: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}}, N = 8</math>          0100: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 2, N = 6</math>          0101: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 2, N = 8</math>          0110: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 4, N = 6</math>          0111: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 4, N = 8</math>          1000: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 8, N = 6</math>          1001: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 8, N = 8</math>          1010: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 16, N = 5</math>          1011: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 16, N = 6</math>          1100: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 16, N = 8</math>          1101: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 5</math>          1110: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 6</math>          1111: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 8</math>          注意: 当 <math>\text{ETFLT}[3: 0] = 1, 2 \text{ or } 3</math> 时, 公式中的 <math>f_{\text{DTS}}</math> 由 <math>\text{INT\_CLK}</math> 取代。</p>
MSCFG	Bit 7	RW	<p><b>主/从模式配置</b>          0: 无动作          1: 延迟触发输入 (In) 上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。</p>
TSSEL	Bit 6-4	RW	<p><b>选择从模式触发源</b>          该位用来选择不同的触发输入来同步计数器。          000: 内部触发 0 (IT0)          001: 内部触发 1 (IT1)          010: 内部触发 2 (IT2)          011: 内部触发 3 (IT3)          100: I1 双边沿边沿检出 (I1F_ED)          101: I1 滤波输入 1          110: I2 滤波输入 2          111: 外部触发输入          注意: 为了避免错误边沿检测, 该位在不使用时 (<math>\text{SMODS}=000</math>) 才能改变。</p>
OCCS	Bit 3	RW	<p><b>输出通道清除源选择</b>          0: OCCISS 选择的内部通道          1: ETFP</p>
SMODS	Bit 2-0	RW	<p><b>选择从模式功能</b>          当选择外部信号, 触发信号 TI 的有效边沿与外部输入的极性有关系 (详见输入控制寄存器和控制寄存器描述)          000: <b>禁止从模式</b>—如果 <math>\text{CNTEN} = '1'</math>, 则预分频器直接由内部时钟计数。          001 <b>编码器模式 1</b>—计数器向上/向下计数 I2 边沿, 取决于 I1 电平。</p>

		<p><b>010: 编码器模式 2</b> -计数器向上/向下计数 I1 边沿, 取决于 I2 电平</p> <p><b>011 :编码器模式 3</b> -计数器向上/向下计数 I1 和 I2 边沿, 取决于另一个输入的电平。</p> <p><b>100: 复位模式</b>-选中的触发输入的上升沿重新初始化计数器, 生成寄存器的更新</p> <p><b>101: 门控模式</b>-当触发输入 TI 为高电平, 计数器时钟使能。一旦触发变为低电平, 计数器停止计数 (并非复位)。计数器的启动和停止均受控制。</p> <p><b>110: 触发模式</b>-计数器在触发信号 TI 的上升沿处启动 (不复位)。仅寄存器的启动受控制。</p> <p><b>111: 外部时钟源 1</b>-计数器在 TI 的上升沿计数</p> <p>注意: 如果 I1 双边沿检出被选为触发输入 (TSSEL='100'), 不能使用门控模式。I1 每一次转换, I1 双边沿检出就会输出 1 个脉冲, 而门控模式则是检查触发信号的电平。</p> <p>注意: 在发生来自自主计时器的接收事件之前, 从计时器的时钟必须先使能, 且在接收来自自主计时器的触发过程中, 从计数器时钟不能即时更改。</p>
--	--	--

24.5.2.4 中断使能寄存器 (AD16C4Tn\_IER)

中断使能寄存器 (AD16C4Tn_IER)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT

Reserved	Bit 31-13	-	保留, 必须保持复位值。
CC4OIT	Bit 12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit 11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit 10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit 9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8	-	保留, 必须保持为复位值
BRKIT	Bit 7	W	使能刹车中断 0: 无效 1: 使能
TRGIT	Bit 6	W	使能触发中断 0: 无效 1: 使能
COMIT	Bit 5	W	使能 COM 中断 0: 无效 1: 使能
CC4IT	Bit 4	W	使能捕获/比较 4 中断 0: 无效 1: 使能
CC3IT	Bit 3	W	使能捕获/比较 3 中断 0: 无效 1: 使能
CC2IT	Bit 2	W	使能捕获/比较 2 中断 0: 无效 1: 使能
CC1IT	Bit 1	W	使能捕获/比较 1 中断 0: 无效 1: 使能
UIT	Bit 0	W	使能更新事件中中断 0: 无效 1: 使能

### 24.5.2.5 中断禁止寄存器 (AD16C4Tn\_IDR)

中断禁止寄存器 (AD16C4Tn_IDR)																																										
偏移地址: 0010 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT																		

Reserved	Bit 31-13	-	保留, 必须保持复位值。
CC4OIT	Bit 12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit 11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit 10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit 9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8	-	保留, 必须保持为复位值
BRKIT	Bit 7	W	禁止刹车中断 0: 无效 1: 禁止
TRGIT	Bit 6	W	禁止触发中断 0: 无效 1: 禁止
COMIT	Bit 5	W	禁止 COM 中断 0: 无效 1: 禁止
CC4IT	Bit 4	W	禁止捕获/比较 4 中断 0: 无效 1: 禁止
CC3IT	Bit 3	W	禁止捕获/比较 3 中断 0: 无效 1: 禁止
CC2IT	Bit 2	W	禁止捕获/比较 2 中断 0: 无效 1: 禁止
CC1IT	Bit 1	W	禁止捕获/比较 1 中断

			0: 无效 1: 禁止
UIT	Bit 0	W	禁止更新中断 0: 无效 1: 禁止

### 24.5.2.6 中断有效状态寄存器 (AD16C4Tn\_IVS)

中断有效状态寄存器 (AD16C4Tn_IVS)																																					
偏移地址: 0014 <sub>H</sub>																																					
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved												CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	BRKIT	TRGIT	COMIT	CC4IT	CC3IT	CC2IT	CC1IT	UIT													

Reserved	Bit 31-13	-	保留
CC4OIT	Bit 12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit 11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit 10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit 9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8	-	保留, 必须保持为复位值
BRKIT	Bit 7	R	刹车中断状态 0: 禁止 1: 使能
TRGIT	Bit 6	R	触发中断状态 0: 禁止 1: 使能
COMIT	Bit 5	R	COM中断状态 0: 禁止 1: 使能
CC4IT	Bit 4	R	通道 4 捕获/比较中断状态 0: 禁止 1: 使能
CC3IT	Bit 3	R	通道 3 捕获/比较中断状态 0: 禁止 1: 使能
CC2IT	Bit 2	R	通道 2 捕获/比较中断状态 0: 禁止 1: 使能
CC1IT	Bit 1	R	通道 1 捕获/比较中断状态

			0: 禁止 1: 使能
UIT	Bit 0	R	更新事件中断状态 0: 禁止 1: 使能



### 24.5.2.7 原始中断标志寄存器 (AD16C4Tn\_RIF)

中断标志寄存器 (AD16C4Tn_RIF)																																					
偏移地址: 0018 <sub>H</sub>																																					
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved												CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved	BRKIF	TRGIF	COMIF	CH4CCIF	CH3CCIF	CH2CCIF	CH1CCIF	UEVTIF													

Reserved	Bit 31-13	-	保留
CH4OVIF	Bit 12	R	<p><b>通道 4 捕获/比较捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH4CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器</p>
CH3OVIF	Bit 11	R	<p><b>通道 3 捕获/比较捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH3CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器</p>
CH2OVIF	Bit 10	R	<p><b>通道 2 捕获/比较捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH2CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器</p>
CH1OVIF	Bit 9	R	<p><b>通道 1 捕获/比较捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH1CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器</p>
Reserved	Bit 8	R	保留
BRKIF	Bit 7	R	<p><b>刹车中断标志</b></p> <p>如果刹车中断使能, 一旦刹车输入有效, 该标志</p>

			<p>位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生刹车事件。</p> <p>1: 在刹车输入上检测到有效电平</p>
TRGIF	Bit 6	R	<p><b>触发中断标志</b></p> <p>如果触发中断使能，当从模式控制器在门控模式以外的所有模式下使能，发生触发事件时（TI 上检测到有效边沿），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生触发事件</p> <p>1: 触发中断被挂起</p>
COMIF	Bit 5	R	<p><b>COM 中断标志</b></p> <p>如果 COM 中断使能，当发生 COM 事件（当捕获/比较控制位 CCnEN, CCnNEN, CHnOMOD 更新后），该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生 COM 事件</p> <p>1: COM 中断被挂起</p>
CH4CCIF	Bit 4	R	<p><b>通道 4 捕获/比较中断标志</b></p> <p>参考 CH1CCIF 描述</p>
CH3CCIF	Bit 3	R	<p><b>通道 4 捕获/比较中断标志</b></p> <p>参考 CH1CCIF 描述</p>
CH2CCIF	Bit 2	R	<p><b>通道 2 捕获/比较中断标志</b></p> <p>参考 CH1CCIF 描述</p>
CH1CCIF	Bit 1	R	<p><b>通道 1 捕获/比较 1 中断标志</b></p> <p><b>如果 CC1 通道配置为输出：</b></p> <p>如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 不匹配。</p> <p>1: AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。</p> <p><b>如果 CC1 通道配置为输入：</b></p> <p>发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。</p> <p>0: 未发生输入捕获</p> <p>1: 计数值捕获至 AD16C4Tn_CCVAL1 寄存器（I1 上检测到与选中极性匹配的边沿）</p>
UEVTIF	Bit 0	R	<p><b>更新事件中断标志</b></p>

		<p>如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生更新。</p> <p>1: 更新中断被挂起。当寄存器更新时，该位被硬件置起：</p> <ul style="list-style-type: none"> <li>-当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0</li> <li>-当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0</li> <li>-当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0</li> </ul>
--	--	---

### 24.5.2.8 中断标志屏蔽寄存器 (AD16C4Tn\_IFM)

中断标志屏蔽寄存器 (AD16C4Tn_IFM)																																												
偏移地址: 001C <sub>H</sub>																																												
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Reserved																			CH4OVIM	CH3OVIM	CH2OVIM	CH1OVIM	Reserved	BRKIM	TRGIM	COMIM	CH4CCIM	CH3CCIM	CH2CCIM	CH1CCIM	UEVTIM													

Reserved	Bit 31-13	-	保留
CH4OVIM	Bit 12	R	屏蔽通道 4 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH4CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH3OVIM	Bit 11	R	屏蔽通道 3 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH3CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH2OVIM	Bit 10	R	屏蔽通道 2 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH2CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
CH1OVIM	Bit 9	R	屏蔽通道 1 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH1CCIF 标志为置起时, 捕获计数器值至 AD16C4Tn_CCVAL1 寄存器
Reserved	Bit 8	R	保留
BRKIM	Bit 7	R	刹车中断标志屏蔽 如果刹车中断使能, 一旦刹车输入有效, 该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0: 未发生刹车事件。 1: 在刹车输入上检测到有效电平
TRGIM	Bit 6	R	屏蔽触发中断标志 如果触发中断使能, 当从模式控制器在门控模式以外的所有模式下使能, 发生触发事件时 (TI 上检测到有效边沿), 该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。 0: 未发生触发事件 1: 触发中断被挂起
COMIM	Bit 5	R	屏蔽 COM 中断标志 如果 COM 中断使能, 当发生 COM 事件 (当捕获

			<p>/比较控制位 CCnEN, CCnNEN, CHnOMOD 更新后), 该标志位被硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生 COM 事件</p> <p>1: COM 中断被挂起</p>
CH4CCIM	Bit 4	R	<p>屏蔽通道 4 捕获/比较中断标志</p> <p>参考 CH1CCIM 描述</p>
CH3CCIM	Bit 3	R	<p>屏蔽通道 3 捕获/比较中断标志</p> <p>参考 CH1CCIM 描述</p>
CH2CCIM	Bit 2	R	<p>屏蔽通道 2 捕获/比较中断标志</p> <p>参考 CH1CCIM 描述</p>
CH1CCIM	Bit 1	R	<p>屏蔽通道 1 捕获/比较中断标志</p> <p>如果通道 1 配置为输出：</p> <p>如果中断使能，除去中央对齐模式的情况（参考 AD16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 不匹配。</p> <p>1: AD16C4Tn_COUNT 计数值与 AD16C4Tn_CCVAL1 值匹配。当 AD16C4Tn_CCVAL1 寄存器值大于 AD16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。</p> <p>如果通道配置为输入：</p> <p>发生捕获时，该位由硬件置起。该位可通过软件或者读取 AD16C4Tn_CCVAL1 寄存器来清零。</p> <p>0: 未发生输入捕获</p> <p>1: 计数值捕获至 AD16C4Tn_CCVAL1 寄存器 (I1 上检测到与选中极性匹配的边沿)</p>
UEVTIM	Bit 0	R	<p>屏蔽更新事件中中断标志</p> <p>如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 AD16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生更新。</p> <p>1: 更新中断被挂起。当寄存器更新时，该位被硬件置起：</p> <ul style="list-style-type: none"> <li>-当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 AD16C4Tn_CON1 寄存器中 DISUE=0</li> <li>-当使用 AD16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0</li> </ul>

			-当 CNT 由触发事件来重新初始化，如果 AD16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0
--	--	--	---

### 24.5.2.9 中断清零寄存器 (AD16C4Tn\_ICR)

中断清零寄存器 (AD16C4Tn_ICR)																															
偏移地址: 0020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CH4OVIC	CH3OVIC	CH2OVIC	CH1OVIC	Reserved	BRKIC	TRGIC	COMIC	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEVTIC	

Reserved	Bit 31-13	-	保留
CH4OVIC	Bit 12	R	通道 4 捕获/比较捕获溢出中断标志清除 0: 无效 1: CH4OVIF 清除
CH3OVIC	Bit 11	R	通道 3 捕获/比较捕获溢出中断标志 0: 无效 1: CH3OVIF 清除
CH2OVIC	Bit 10	R	通道 2 捕获/比较捕获溢出中断标志 0: 无效 1: CH2OVIF 清除
CH1OVIC	Bit 9	R	通道 1 捕获/比较捕获溢出中断标志 0: 无效 1: CH1OVIF 清除
Reserved	Bit 8	R	保留
BRKIC	Bit 7	C_W1	清刹车中断 0: 无效 1: 清零 (AD16C4Tn_RIF)
TRGIC	Bit 6	C_W1	清触发中断 0: 无效 1: 清零 (AD16C4Tn_RIF)
COMIC	Bit 5	C_W1	清 COM 中断清零 0: 无效 1: 清零 (AD16C4Tn_RIF)
CH4CCIC	Bit 4	C_W1	通道 4 捕获/比较中断清零 参考 CH1CCIC 描述
CH3CCIC	Bit 3	C_W1	通道 3 捕获/比较中断清零 参考 CH1CCIC 描述
CH2CCIC	Bit 2	C_W1	通道 2 捕获/比较中断清零 参考 CH1CCIC 描述
CH1CCIC	Bit 1	C_W1	通道 1 捕获/比较中断清零 0: 无效 1: 清零 (AD16C4Tn_RIF)
UEVTIC	Bit 0	C_W1	更新事件中中断清零

			0: 无效 1: 清零 (AD16C4Tn_RIF)
--	--	--	-------------------------------



24. 5. 2. 10 软件生成事件寄存器 (AD16C4Tn\_SGE)

软件生成事件寄存器 (AD16C4Tn_SGE)																															
偏移地址: 0024 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SGBRK	SGTRG	SGCOM	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU

Reserved	Bit 31-8	-	保留
SGBRK	Bit 7	W	<p><b>软件生成刹车事件</b> 该位由软件设置来生成刹车事件，可由硬件自动清零。 0: 无动作 1: 产生刹车事件。GOEN 清零，BRKIF 标志位置起，产生相关中断或 DMA 传输。</p>
SGTRG	Bit 6	W	<p><b>软件生成触发事件</b> 该位由软件设置来生成触发事件，可由硬件自动清零。 0: 无动作 1: AD16C4Tn_RIF 寄存器中的 TRGIF 被置起，产生相关中断或 DMA 传输</p>
SGCOM	Bit 5	W	<p><b>软件生成换相事件捕获</b> 该位由软件设置，由硬件自动清零。 0: 无动作 1: 当 CCPCEN 被置 1，则可更新 CCnEN, CCnNEN 和 CHnOMOD 注意: 该位只有用作于通道时才有互补输出</p>
SGCC4E	Bit 4	W	<p><b>软件触发通道 4 捕获/比较事件</b> 参考 SGCC1E 描述</p>
SGCC3E	Bit 3	W	<p><b>软件触发通道 3 捕获/比较事件</b> 参考 SGCC1E 描述</p>
SGCC2E	Bit 2	W	<p><b>软件触发通道 2 捕获/比较事件</b> 参考 SGCC1E 描述</p>
SGCC1E	Bit 1	W	<p><b>软件触发通道 1 捕获/比较事件</b> 该位由软件设置来生成事件，可由硬件自动清零。 0: 无动作 1: 通道 1 上产生捕获/比较事件: <b>如果通道 1 配置为输出:</b> CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。 <b>如果通道 1 配置为输入:</b></p>

			<p>当前计数值捕获至 AD16C4Tn_CCVAL1 寄存器。CH1CCIF 标志位被置起，产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起如果 CH1CCIF 标志位为高电平。</p>
SGU	Bit 0	W	<p><b>软件触发更新事件</b>                  该位由软件设置，可由硬件自动清零。  <b>0:</b> 无动作  <b>1:</b> 重新初始化计数器，更新寄存器。注意，预分频器也会被清零（但预分频比不会受到影响）。如果使用中央对齐模式或者 DIRSEL=0（递增），则计数器将清零；否则如果 DIRSEL=1（递减），则将使用自动重载入值。</p>

### 24.5.2.11 通道捕获模式寄存器 1 (AD16C4Tn\_CHMR1)

◇ 输出比较模式

通道捕获模式寄存器 1 (AD16C4Tn_CHMR1)																															
偏移地址: 0028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPREN	CH2OHSEN	CC2SSEL	CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL		

Reserved	Bit 31-16	-	保留
CH2OCLREN	Bit 15	R/W	使能通道 2 输出比较清零 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	通道 2 输出比较模式 参考 CH1OMOD 描述
CH2OPREN	Bit 11	R/W	使能通道 2 输出比较预载入 参考 CH1OPREN 描述
CH2OHSEN	Bit 10	R/W	使能通道 2 输出比较高速模式 参考 CH1OHSEN 描述
CC2SSEL	Bit 9-8	R/W	选择通道 2 输出比较 该位定义了通道以及使用的输入的方向（输入/输出） 00: 通道配置为输出 01: 通道配置为输入，捕获源为 I2 10: 通道配置为输入，捕获源为 I1 11: 通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 仅当内部触发输入通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（AD16C4Tn_CCEP 中 CC2EN = '0'），CC2SSEL 为只写。
CH1OCLREN	Bit 7	R/W	使能通道 1 输出比较清零 0: 通道 1 比较输出不会受到 ETFP 输入影响 1: 当 ETFP 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	输出比较 1 模式 该位定义了输出参考信号通道 1 比较输出的行为。 通道 1 比较输出为高有效，CH1O 和 CH1ON 的有效电平由 CC1POL 和 CC1NPOL 位决定。 000:冻结—输出比较寄存器 AD16C4Tn_CCVAL1

		<p>寄存器和 AD16C4Tn_COUNT 计数器之间的比较对输出无效。</p> <p><b>001：发生匹配时设置通道 1 为有效电平</b>-当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。</p> <p><b>010：发生匹配时设置通道 1 为无效电平</b>-当计数器 AD16C4Tn_COUNT 与捕获/比较寄存器 1AD16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。</p> <p><b>011：翻转</b> -当 AD16C4Tn_COUNT=AD16C4Tn_CCVAL1，通道 1 比较输出发生翻转。</p> <p><b>100：强制为无效电平</b> - 通道 1 比较输出强制为低电平。</p> <p><b>101：强制为有效电平</b>- 通道 1 比较输出强制为高电平。</p> <p><b>110：PWM 模式 1</b> -在递增模式下，当 AD16C4Tn_COUNT&lt;AD16C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 AD16C4Tn_COUNT&gt;AD16C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。</p> <p><b>111：PWM 模式 2</b> -在递增模式下，当 AD16C4Tn_COUNT&lt;AD16C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 AD16C4Tn_COUNT&gt;AD16C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。</p> <p>注意：</p> <p><b>1：</b>当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3，且 CC1SSEL=00（通道为输出模式），该位将不能更改。</p> <p><b>2：</b>在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。</p> <p><b>3：</b>对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPEN 位设置为 1，则只有当 COM 事件发生时，CH1OMOD 有效位才会设置为预载值中新的值。</p>
CH1OPREN	Bit 3	<p><b>输出比较 1 预载使能</b></p> <p>0: AD16C4Tn_CCVAL1 的预载寄存器禁止。 AD16C4Tn_CCVAL1 在任何时候都可写，新写入</p>

			<p>的值将立刻生效。</p> <p>1: AD16C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件, AD16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。</p> <p>注意:</p> <p>1: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 3, 且 CC1SSEL=00 (通道为输出模式), 该位将不能更改。</p> <p>2: 仅在单脉冲模式下 (AD16C4Tn_CON1 寄存器中的 SPMEN 设置为 1), PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。</p>
CH1OHSEN	Bit 2	R/W	<p><b>使能通道 1 输出比较高速</b></p> <p>该位用来加速在 CC 输出上的输入触发事件的效应。</p> <p>0: 当触发开启, 通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时, 至少需要 5 个时钟周期来激活通道 1 输出。</p> <p>1: 触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平, 采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式, CH1OHSEN 才会起作用。</p>
CC1SSEL	Bit 1-0	R/W	<p><b>捕获/比较 1 选择</b></p> <p>该位定义了通道和使用的输入的方向。</p> <p>00: 通道配置为<b>输出</b></p> <p>01: 通道配置为输入, 捕获源为 <b>I1</b></p> <p>10: 通道配置为输入, 捕获源为 <b>I2</b></p> <p>11: 通道配置为输入, 捕获源为 <b>ITn 或 I1 的双边沿检出</b>。</p> <p>只有当内部触发输入是通过 TSSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才运行。</p> <p>注意: 当通道关闭 (AD16C4Tn_CCEP 寄存器中的 CC1EN = '0'), CC1SSEL 为只写。</p>

◇ 输入捕获模式

通道捕获模式寄存器 1 (AD16C4Tn_CHMR1)																															
偏移地址: 0028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT			IC2PRES		CC2SSEL		I1FLT			IC1PRES		CC1SSEL			

Reserved	Bit 31-16	-	保留
I2FLT	Bit 15-12	R/W	通道 2 输入捕获滤波器 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	通道 2 输入捕获预分频器 参考 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	选择通道 2 输入捕获源 该位定义了通道和使用的输入的方向。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 I2 10: 通道配置为输入, 捕获源为 I1 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才运行 注意: 当通道关闭 (AD16C4Tn_CCEP 寄存器中的 CC2EN = '0'), CC2SSEL 为只写。
I1FLT	Bit 7-4	R/W	I1 滤波器 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成, 每 N 个连续事件才视为一个有效边沿: 0000: 无滤波器, 采样频率为 $f_{DTS}$ 0001: $f_{SAMPLING} = f_{INT\_CLK}, N = 2$ 0010: $f_{SAMPLING} = f_{INT\_CLK}, N = 4$ 0011: $f_{SAMPLING} = f_{INT\_CLK}, N = 8$ 0100: $f_{SAMPLING} = f_{DTS} / 2, N = 6$ 0101: $f_{SAMPLING} = f_{DTS} / 2, N = 8$ 0110: $f_{SAMPLING} = f_{DTS} / 4, N = 6$ 0111: $f_{SAMPLING} = f_{DTS} / 4, N = 8$ 1000: $f_{SAMPLING} = f_{DTS} / 8, N = 6$ 1001: $f_{SAMPLING} = f_{DTS} / 8, N = 8$ 1010: $f_{SAMPLING} = f_{DTS} / 16, N = 5$ 1011: $f_{SAMPLING} = f_{DTS} / 16, N = 6$

			<p>1100: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 16, N = 8</math>          1101: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 5</math>          1110: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 6</math>          1111: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 8</math>          注意: 当 ICxF [3: 0] = 1, 2 or 3 时, 公式中的 <math>f_{\text{DTS}}</math> 由 INT_CLK 取代。</p>
IC1PRES	Bit 3-2	R/W	<p><b>捕获比较器 1 输入预分频器</b>          该位定义了作用在 CC1 输入 (I1) 上的预分频比。当 CC1EN='0' (AD16C4Tn_CCEP 寄存器), 预分频器将复位。          00: 无预分频器。每当捕获输入上检测到边沿时, 发生捕获动作。          01: 每发生 2 次事件, 执行一次捕获          10: 每发生 4 次事件, 执行一次捕获          11: 每发生 8 次事件, 执行一次捕获</p>
CC1SSEL	Bit 1-0	R/W	<p><b>捕获比较器 1 输入源选择</b>          该位定义了通道和使用的输入的方向。          00: CC1 通道配置为<b>输出</b>          01: CC1 通道配置为输入, 捕获源为 <b>I1</b>          10: CC1 通道配置为输入, 捕获源为 <b>I2</b>          11: CC1 通道配置为输入, 捕获源为 <b>ITn 或 I1 的双边沿检出</b>。          只有当内部触发输入是通过 TSSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才运行          注意: 当通道关闭 (AD16C4Tn_CCEP 寄存器中的 CC1EN = '0'), CC1SSEL 为只写。</p>

### 24.5.2.12 通道捕获模式寄存器 2 (AD16C4Tn\_CHMR2)

◇ 输出比较模式

通道捕获模式寄存器 2 (AD16C4Tn_CHMR2)																															
偏移地址: 002C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPREN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPREN	CH3OHSEN	CC3SSEL	

Reserved	Bit 31-16	-	保留
CH4OCLREN	Bit 15	R/W	通道 4 输出比较清零使能 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	通道 4 输出比较模式 参考 CH1OMOD 描述
CH4OPREN	Bit 11	R/W	通道 4 输出比较预载入使能 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	通道 4 输出比较高速使能 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	通道 4 输出比较选择 该位定义了通道以及使用的输入的方向 (输入/输出)。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 I4 10: 通道配置为输入, 捕获源为 I3 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 仅当内部触发输入通过 TSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才能工作。 注意: 当通道为关闭状态时 (AD16C4Tn_CCEP 中 CC4EN = '0'), CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	通道 3 输出比较清零使能 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	通道 3 输出比较模式 参考 CH1OMOD 描述
CH3OPREN	Bit 3	R/W	通道 3 输出比较预载使能 参考 CH1OPREN 描述
CH3OHSEN	Bit 2	R/W	通道 3 输出比较高速使能 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	通道 3 捕获/比较选择



		<p>该位定义了通道和使用的输入的方向。</p> <p>00：通道配置为<b>输出</b></p> <p>01：通道配置为输入，捕获源为 <b>I3</b></p> <p>10：通道配置为输入，捕获源为 <b>I4</b></p> <p>11：通道配置为输入，捕获源为 <b>ITn 或 I1 的双边沿检出</b>。</p> <p>只有当内部触发输入是通过 TSSEL 位（AD16C4Tn_SMCON 寄存器）选择时，该模式才运行。</p> <p>注意：当通道关闭（AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写</p>
--	--	--

◇ 输入捕获模式

通道捕获模式寄存器 2 (AD16C4Tn_CHMR2)																															
偏移地址: 002C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT			IC4PRES		CC4SSEL		I3FLT			IC3PRES		CC3SSEL			

Reserved	Bit 31-16	-	保留
I4FLT	Bit 15-12	R/W	通道 4 输入捕获滤波器 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	通道 4 输入捕获预分频器 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	选择通道 4 输入捕获源 该位定义了通道和使用的输入的方向。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 I4 10: 通道配置为输入, 捕获源为 I3 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才运行 注意: 当通道关闭 (AD16C4Tn_CCEP 寄存器中的 CC4EN = '0'), CC4SSEL 为只写。
I3FLT	Bit 7-4	R/W	通道 3 输入捕获滤波器 参考 I1FLT 描述
IC3PRES	Bit 3-2	R/W	通道 3 输入捕获预分频器 参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	选择通道 3 输入捕获源 该位定义了通道和使用的输入的方向。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 I3 10: 通道配置为输入, 捕获源为 I4 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位 (AD16C4Tn_SMCON 寄存器) 选择时, 该模式才运行 注意: 当通道关闭 (AD16C4Tn_CCEP 寄存器中的 CC3EN = '0'), CC3SSEL 为只写。

24. 5. 2. 13 捕获/比较使能极性寄存器 (AD16C4Tn\_CCEP)

通道捕获/比较使能寄存器 (AD16C4Tn_CCEP)																																						
偏移地址: 0030 <sub>H</sub>																																						
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Reserved													CC4POL	CC4EN	CC3NPOL	CC3NE	CC3POL	CC3EN	CC2NPOL	CC2NE	CC2POL	CC2EN	CC1NPOL	CC1NEN	CC1POL	CC1EN												

Reserved	Bit 31-14	-	保留
CC4POL	Bit 13	R/W	通道 4 捕获/比较输出极性 参考 CC1POL 描述
CC4EN	Bit 12	R/W	使能通道 4 捕获/比较输出 参考 CC1EN 描述
CC3NPOL	Bit 11	R/W	通道 3 捕获/比较互补输出极性 参考 CC1NPOL 描述
CC3NEN	Bit 10	R/W	使能通道 3 捕获/比较互补输出 参考 CC1NEN 描述
CC3POL	Bit 9	R/W	通道 3 捕获/比较输出极性 参考 CC1POL 描述
CC3EN	Bit 8	R/W	使能通道 3 捕获/比较输出 参考 CC1EN 描述
CC2NPOL	Bit 7	R/W	通道 2 捕获/比较 2 互补输出极性 参考 CC1NPOL 描述
CC2NEN	Bit 6	R/W	使能通道 2 捕获/比较互补输出 参考 CC1NEN 描述
CC2POL	Bit 5	R/W	通道 2 捕获/比较输出极性 参考 CC1POL 描述
CC2EN	Bit 4	R/W	使能通道 2 捕获/比较输出 参考 CC1EN 描述
CC1NPOL	Bit 3	R/W	通道 1 捕获/比较互补输出极性 通道配置为输出： 0: CH1ON 高有效。 1: CH1ON 低有效。 通道配置为输入： 该位需和 CC1POL 一起使用来定义输入边沿的极性。参考 CC1POL 描述。 注意：对于有互补输出的通道，该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1，则只有当 COM 事件发生时，CC1NPOL 有效位才会设置为预载值中新的值。 注意：当 AD16C4Tn_BDCFG 寄存器中的

			LOCKLVL 位被设置为锁定级别 2 或 3, 且 CC1SSEL=00(通道为输出模式), 该位将不可写。
CC1NEN	Bit 2	R/W	<p><b>使能通道 1 捕获/比较互补输出</b></p> <p>0: 关闭 - CH1ON 无效。CH1ON 电平取决于 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能</p> <p>1: 开启 - CH1ON 为对应输出引脚上的输出信号, 由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定。</p> <p>注意: 对于有互补输出的通道, 该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPEN 位设置为 1, 则只有当 COM 事件发生时, CC1NEN 有效位才会设置为预载值中新的值</p>
CC1POL	Bit 1	R/W	<p><b>通道 1 捕获/比较输出极性</b></p> <p><b>通道配置为输出:</b></p> <p>0: CH1O 高有效</p> <p>1: CH1O 低有效</p> <p><b>通道配置为输入:</b></p> <p>CC1NPOL/CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。</p> <p>00: 正向/上升沿</p> <p>电路对 In 边沿检出的上升沿敏感 (在复位, 外部时钟或触发模式下, 进行捕获或触发), In 边沿检出不反向 (门控模式或编码器模式下, 进行触发)。</p> <p>01: 反向/下降沿</p> <p>电路对 In 边沿检出的下降沿敏感 (在复位, 外部时钟或触发模式下, 进行捕获或触发), In 边沿检出反向 (门控模式或编码器模式下, 进行触发)。</p> <p>10: 保留, 请勿使用该配置</p> <p>11: 正向/双边沿</p> <p>电路对 In 边沿检出的上升沿和下降沿均敏感 (在复位, 外部时钟或触发模式下, 进行捕获或触发), In 边沿检出不反向 (门控模式下, 进行触发)。在编码器接口模式下勿使用该配置。</p> <p>注意: 对于有互补输出的通道, 该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPEN 位设置为 1, 则只有当 COM 事件发生时, CC1POL 有效位才会设置为预载值中新的值。</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位被设置为锁定级别 2 或 3, 且 CC1SSEL=00(通道为输出模式), 该位将不可写。</p>
CC1EN	Bit 0	R/W	<p><b>使能通道 1 捕获/比较输出</b></p> <p><b>通道配置为输出:</b></p> <p>0: 关闭 - CH1O 无效。CH1ON 电平取决于</p>

		<p>GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 的功能</p> <p>1: 开启 - CH1O 为对应输出引脚上的输出信号, 由 GOEN, OFFSSI, OFFSSR, OISS1, OISS1N 和 CC1EN 决定</p> <p><b>通道配置为输入:</b></p> <p>该位决定了计数值是否能捕获到输入捕获/比较寄存器 1 (AD16C4Tn_CCVAL1)。</p> <p>0: 禁止捕获。</p> <p>1: 使能捕获。</p> <p>注意: 对于有互补输出的通道, 该位设置为预载值。如果 AD16C4Tn_CON2 寄存器中的 CCPCEN 位设置为 1, 则只有当 COM 事件发生时, CC1EN 有效位才会设置为预载值中新的值。</p>
--	--	---

#### 24.5.2.14 计数寄存器 (AD16C4Tn\_COUNT)

计数寄存器 (AD16C4Tn_COUNT)																															
偏移地址: 0034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CNTV															

Reserved	Bit 31-16	-	保留
CNTV	Bit 15-0	R/W	计数值

### 24.5.2.15 预分频寄存器 (AD16C4Tn\_PRES)

预分频寄存器 (AD16C4Tn_PRES)																															
偏移地址: 0038 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	保留
PSCV	Bit 15-0	R/W	<p><b>时钟预分频器值</b></p> <p>计数器时钟频率 (CK_CNT) = <math>f_{CK\_PSC} / (PSCV[15: 0] + 1)</math>。</p> <p>每发生一次更新事件 (包括当计数器由 AD16C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时, 通过触发控制器清零), PSCV 包含的值需填入到有效的预分频寄存器内。</p>

### 24.5.2.16 计数器自动装载寄存器 (AD16C4Tn\_AR)

计数器自动装载寄存器 (AD16C4Tn_AR)																															
偏移地址: 003C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	保留
ARRV	Bit 15-0	R/W	<p><b>计数器自动装载值</b></p> <p>ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空, 计数器被屏蔽。</p>

### 24.5.2.17 重复计数寄存器 (AD16C4Tn\_REPAR)

重复计数寄存器 (AD16C4Tn_REPAR)																															
偏移地址: 0040 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REPV							

Reserved	Bit 31-8	-	保留
REPV	Bit 7-0	RW	<p><b>重复计数值</b></p> <p>当预载寄存器使能，该位允许用户设置比较寄存器的更新率（例如：预载到有效寄存器的周期性传输），同样也可以设置更新中断生成率。每次当 REPV_CNT 的相关递减计数器递减至 0，会产生更新事件，会从 REPV 值重新计数。因为只有当发生重复更新事件 U_RC 时，REPV_CNT 才会重新载入 REPV 值，所以只有在发生下一次重复更新事件时，写入 AD16C4Tn_REPAR 寄存器的值才会生效。</p> <p>即，在 PWM 模式下，(REPV+1) 相当于：</p> <ul style="list-style-type: none"> <li>-在边沿对齐模式下，(REPV+1) 对应的是 PWM 的周期数</li> <li>-在中央对齐模式下，(REPV+1) 对应的是 1/2 PWM 的周期数</li> </ul>

24.5.2.18 通道捕获/比较寄存器 1 (AD16C4Tn\_CCVAL1)

通道捕获/比较寄存器 1 (AD16C4Tn_CCVAL1)																															
偏移地址: 0044 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bit 31-16	-	保留
CCRV1	Bit 15-0	R/W	<p><b>捕获/比较值 1</b></p> <p>如果通道 CCn 配置为输出： CCRVn 中的值将被载入实际的捕获/比较寄存器中（预载值）。 如果在 AD16C4Tn_CHMRn 寄存器中的预载功能没有选中，CCRVn 中的值将被永久载入；否则，每当发生更新事件，预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 AD16C4Tn_COUNT 中的值进行比较，并在 OCn 上输出。</p> <p>如果通道 CCn 配置为输入： CCRVn 为由上一个输入捕获事件 (ICn) 传输的计数值。</p>

24.5.2.19 通道捕获/比较寄存器 2 (AD16C4Tn\_CCVAL2)

通道捕获/比较寄存器 2 (AD16C4Tn_CCVAL2)																															
偏移地址: 0044 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bit 31-16	-	保留
CCRV2	Bit 15-0	R/W	<p><b>通道捕获/比较值 2</b></p> <p>参考 CCRV1 描述</p>



### 24.5.2.20 通道捕获/比较寄存器 3 (AD16C4Tn\_CCVAL3)

捕获/比较寄存器 3 (AD16C4Tn_CCVAL3)																															
偏移地址: 004C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV3															

Reserved	Bit 31-16	-	保留
CCRV3	Bit 15-0	R/W	捕获/比较值 3 参考 CCRV1 描述

### 24.5.2.21 通道捕获/比较寄存器 4 (AD16C4Tn\_CCVAL4)

通道捕获/比较寄存器 4 (AD16C4Tn_CCVAL4)																															
偏移地址: 0050 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV4															

Reserved	Bit 31-16	-	保留
CCRV4	Bit 15-0	R/W	通道捕获/比较值 4 参考 CCRV1 描述

24. 5. 2. 22 刹车和死区配置寄存器 (AD16C4Tn\_BDCFG)

刹车和死区配置寄存器 (AD16C4Tn_BDCFG)																																					
偏移地址: 0054 <sub>H</sub>																																					
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved																GOEN	AOEN	BRKP	BRKEN	OFFSSR	OFFSSI	LOCKLVL	DT														

Reserved	Bit 31-16	-	保留
GOEN	Bit 15	RW	<p><b>通道主要输出使能</b></p> <p>一旦刹车输入有效, 该位会由硬件异步清零。该位可由软件置 1 或自动置 1, 取决于 AOEN 位。该位仅作用于配置为输出的通道。</p> <p>0: OC 和 OCN 输出禁止或强制为空闲状态。</p> <p>1: 如果 OC 和 OCN 各自的使能位都置 1 (AD16C4Tn_CCEP 寄存器中的 CCnEN, CCnNEN), 则 OC 和 OCN 输出使能。</p>
AOEN	Bit 14	RW	<p><b>通道自动输出使能</b></p> <p>0: GOEN 仅可由软件置位</p> <p>1: 在下一个更新事件发生时 (如果刹车输入无效), GOEN 可由软件或自动置位。</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1, 则该位不可更改。</p>
BRKP	Bit 13	RW	<p><b>选择通道刹车极性</b></p> <p>0: 刹车输入 BRKP 为低有效</p> <p>1: 刹车输入 BRKP 为高有效</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1, 则该位不可更改</p> <p>注意: 任何对该位的写操作都要延时 1 APB 时钟周期后才变为有效。</p>
BRKEN	Bit 12	RW	<p><b>使能刹车</b></p> <p>0: 刹车输入 (BRKP 和 CCS 时钟失效事件) 禁止</p> <p>1: 刹车输入 (BRKP 和 CCS 时钟失效事件) 使能</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1, 则该位不可更改</p> <p>注意: 任何对该位的写操作都要延时 1 APB 时钟周期后才变为有效。</p>

OFFSSR	Bit 11	R/W	<p>运行模式下的无效状态选择位</p> <p>该位使用于, 当 GOEN=1 时, 被配置为输出并使用互补输出的通道。如果计时器中没有使用互补输出, 则 OFFSSR 不使用。</p> <p>0: 无效状态时, OC/OCN 输出禁止 (OC/OCN 使能输出信号=0)。</p> <p>1: 无效状态时, 当 CCnEN=1 或 CCnNEN=1 时, 便使能 OC/OCN 输出并将其设为无效电平。(OC/OCN 使能输出信号=1)</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2, 则该位不可更改。</p>
OFFSSI	Bit 10	R/W	<p>空闲模式下的空闲状态选择位该位使用于, 当 GOEN=0 时, 被配置为输出通道</p> <p>0: 无效状态时, OC/OCN 输出禁止 (OC/OCN 使能输出信号=0)。</p> <p>1: 无效状态时, 当 CCnEN=1 或 CCnNEN=1, 便将 OC/OCN 输出首先强制为其空闲电平。(OC/OCN 使能输出信号=1)</p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 2, 则该位不可更改。</p>
LOCKLVL	Bit 9-8	R/W	<p><b>锁定级别配置</b></p> <p>针对软件错误, 该位提供写保护。</p> <p>00: 锁定关闭-不提供写保护</p> <p>01: 锁定级别 1 = AD16C4Tn_BDCFG 寄存器中的 DT, AD16C4Tn_CON2 寄存器中的 OISSn 和 OISSnN, 和 AD16C4Tn_BDCFG 寄存器中的 BRKEN/BRKP/AOEN 不再可写。</p> <p>10: 锁定级别 2 = 锁定级别 1 + CC 极性位 (AD16C4Tn_CCEP 寄存器中的 CCnPOL/CCnNPOL, 只要相关通道由 CCnSSEL 配置为输出) 以及 OFFSSR 和 OFFSSI 都不再可写。</p> <p>11: 锁定级别 3 = 锁定级别 2 + CC 控制位 (AD16C4Tn_CHMRn 寄存器中的 CHnOMOD 和 CHnOPREN, 只要相关通道由 CCnSSEL 配置为输出) 都不再可写。</p> <p>注意: 锁定配置为仅在复位后可写。一旦 AD16C4Tn_BDCFG 已写, 其设置内容在下一个复位前都处于冻结状态。</p>
DT	Bit 7-0	R/W	<p><b>死区延时设置值</b></p> <p>该位定义了互补输出之间插入的死区时间。DT 对应的就是该时间段。</p>

		<p><b>DT[7: 5]=0xx =&gt; DT=DT[7:0]x t<sub>dtg</sub></b>, 式中 t<sub>dtg</sub>=t<sub>DTS</sub></p> <p><b>DT[7: 5]=10x =&gt; DT= (64+DT[5:0]) x t<sub>dtg</sub></b>, 式中 t<sub>dtg</sub>=2x t<sub>DTS</sub></p> <p><b>DT[7: 5]=110=&gt; DT= (32+DT[4:0]) x t<sub>dtg</sub></b>, 式中 t<sub>dtg</sub>=8x t<sub>DTS</sub></p> <p><b>DT[7: 5]=111 =&gt; DT= (32+DT[4: 0]) x t<sub>dtg</sub></b>, 式中 t<sub>dtg</sub>=16x t<sub>DTS</sub></p> <p>注意: 当 AD16C4Tn_BDCFG 寄存器中的 LOCKLVL 位已被设置为锁定级别 1, 2 或 3, 则该位不可更改</p>
--	--	---

### 24. 5. 2. 23 DMA使能寄存器 (AD16C4Tn\_DMAEN)

DMA 使能寄存器 (AD16C4Tn_DMAEN)																															
偏移地址: 058 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TRGDMA	COMDMA	CC4DMA	CC3DMA	CC2DMA	CC1DMA	UDMA	

Reserved	Bit 31-7	-	保留, 必须保持为复位值
TRGDMA	Bit 6	R/W	触发DMA请求使能 0: DMA请求禁止 1: DMA请求使能
COMDMA	Bit 5	R/W	COM DMA访问使能 0: DMA请求禁止 1: DMA请求使能
CC4DMA	Bit 4	R/W	捕获/比较值 4 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC3DMA	Bit 3	R/W	捕获/比较值 3 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC2DMA	Bit 2	R/W	捕获/比较值 2 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC1DMA	Bit 1	R/W	捕获/比较值 1 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
UDMA	Bit 0	R/W	更新 DMA 请求使能 0: DMA 请求禁止 1: DMA 请求使能

## 第25章 通用定时器(GP16C4T)

### 25.1 概述

通用定时器 (GP16C4T) 包含一个 16 位自动重载计数器, 该计数器由可配置的预分频器驱动。

通用定时器 (GP16C4T) 的用途广泛, 可测量信号脉冲长度 (输入捕获) 或输出脉冲波形 (比较输出、PWM)。

### 25.2 特性

- ◇ 16 位递增, 递减, 递增/递减自动加载计数器
- ◇ 16 位可编程预分频器, 可对计数器工作时钟进行 1 到 65536 间的任意分频
- ◇ 多达四个独立信道
  - ◇ 输入捕获
  - ◇ 输出比较
  - ◇ PWM 产生 (边沿与中央对齐模式)
  - ◇ 单脉冲输出模式
- ◇ 同步电路用于外部信号控制定时器及内部互联多个定时器
- ◇ 以下事件中产生中断/DMA:
  - ◇ 更新事件: 计数器上溢/下溢, 计数器初始化 (通过软件或内/外部触发)
  - ◇ 触发事件 (计数器起始、停止、初始化或内/外触发计数)
  - ◇ 输入捕获
  - ◇ 输出比较
- ◇ 支持增量 (正交) 编码及霍尔电路进行定位
- ◇ 触发输入可对外部时钟或周期性电流管理

### 25.3 结构框图

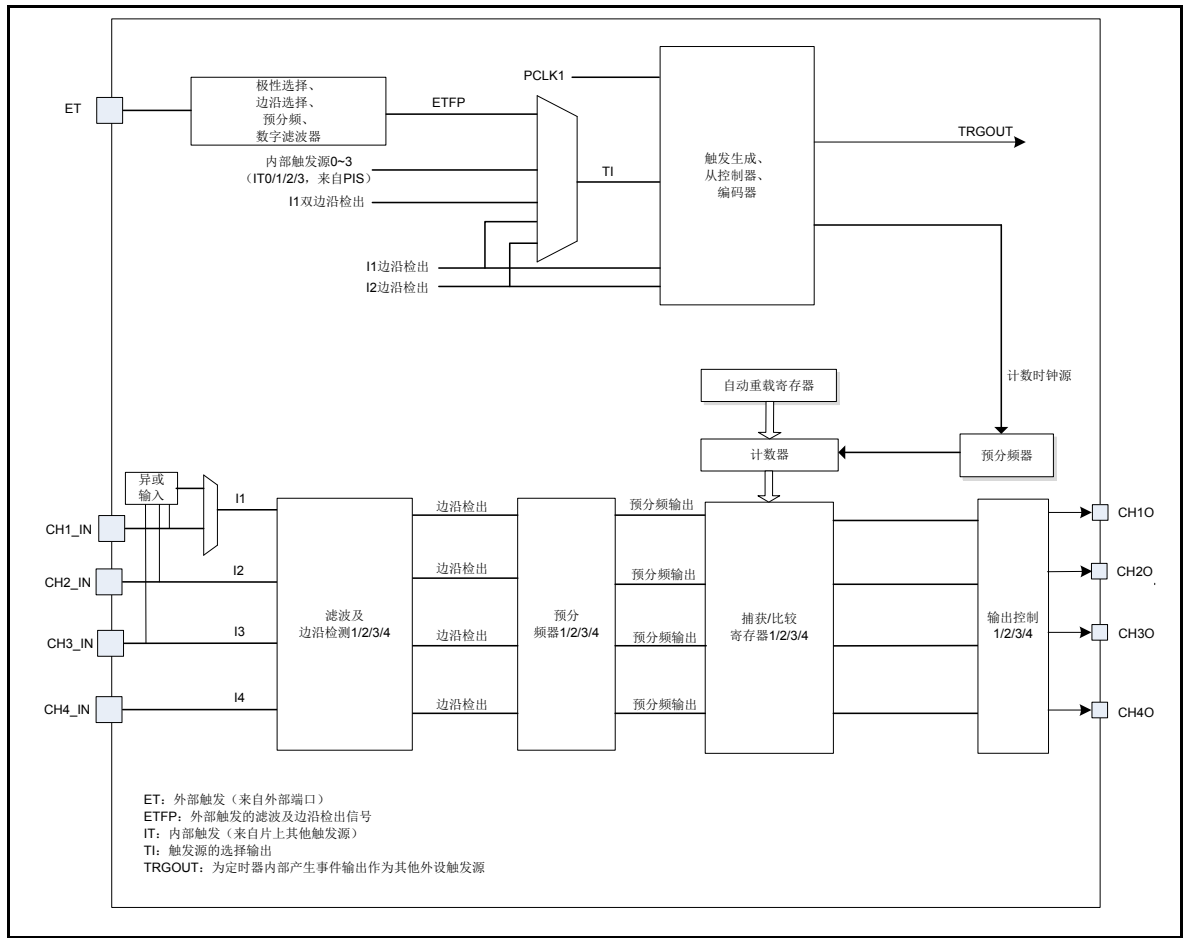


图 25-1 通用定时器电路结构框图

### 25.4 功能描述

#### 25.4.1 预分频器

定时器包含一个 16-bit 的计数器 (GP16C4Tn\_COUNT)，计数时钟由预分频寄存器 (GP16C4Tn\_PRES) 进行分频。计数周期由自动重载计数器 (GP16C4Tn\_AR) 设定。

自动重载寄存器 (GP16C4Tn\_AR) 是一个可缓存的寄存器。当 GP16C4Tn\_CON1 寄存器的 ARPEN 位复位时，GP16C4Tn\_AR 寄存器重载功能失效，GP16C4Tn\_AR 就是有效寄存器；ARPEN 置位时，GP16C4Tn\_AR 寄存器具有重载功能，产生更新事件 (UEV) 时，加载值 (GP16C4Tn\_AR 寄存器值) 更新到影子寄存器后才生效。

当 GP16C4Tn\_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢 (或递减下溢) 时会产生更新事件 (UEV)。同样，软件方式也可产生更新事件。GP16C4Tn\_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 GP16C4Tn\_PRES 寄存器值+1 次分频。由于

GP16C4Tn\_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

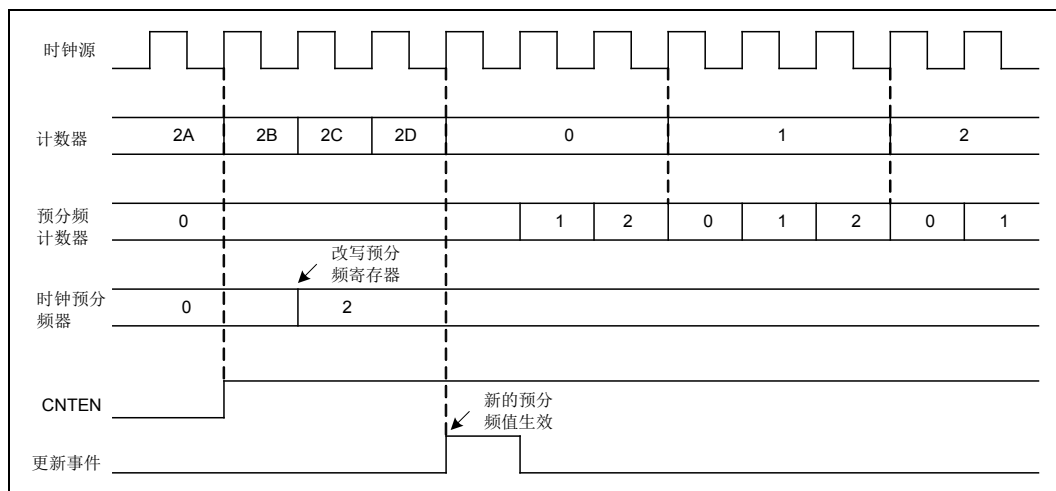


图 25-2 预分频值计数时序图



### 25.4.2 时钟源

计数器工作时钟可以选择内部时钟(INT\_CLK)、外部时钟源 1 (I1、I2、I3、I4)、外部时钟源 2 (ET)，内部触发输入 (IT1、IT2、IT3、IT4)

#### 25.4.2.1 内部时钟源 (INT\_CLK)

若从模式控制器被关闭(GP16C4Tn\_SMCON 寄存器内, SMODS= "000"), 则 CNTEN, GP16C4Tn\_CON1.DIRSEL 与 GP16C4Tn\_SGE.SGU 位为实际控制位, 这些位只能软件修改 (SGU 位除外, 仍硬件自动清除)。一旦 CNTEN 位被写为'1', 预分频器就由内部 INT\_CLK 提供时钟。

下图给出了通常模式下控制电路和递增计数的情况, 没有分频。

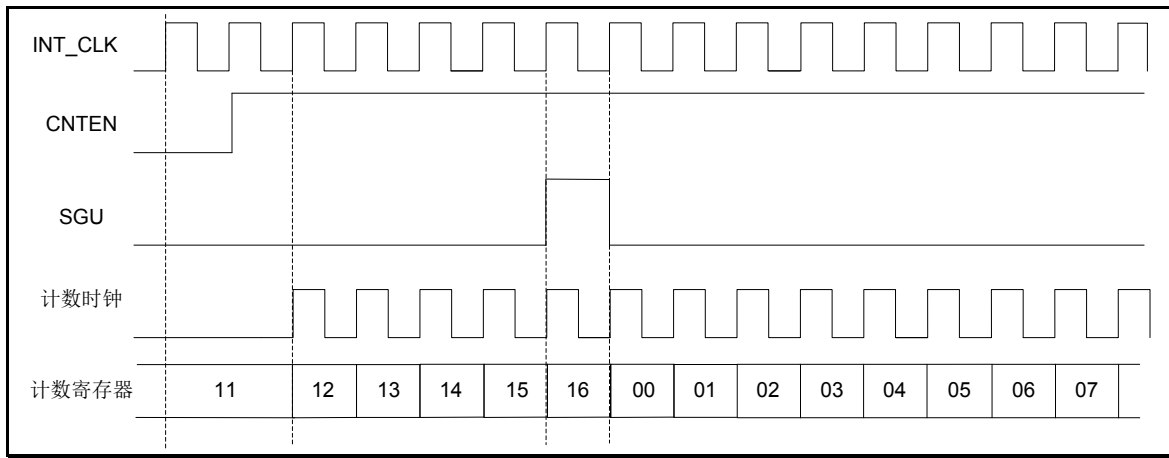


图 25-3 采用内部时钟计数

#### 25.4.2.2 外部时钟源 1

GP16C4Tn\_SMCON 寄存器的 SMODS= "111"时, 可选择外部时钟源 1。计数器可根据选定的上升沿或下降沿计数。

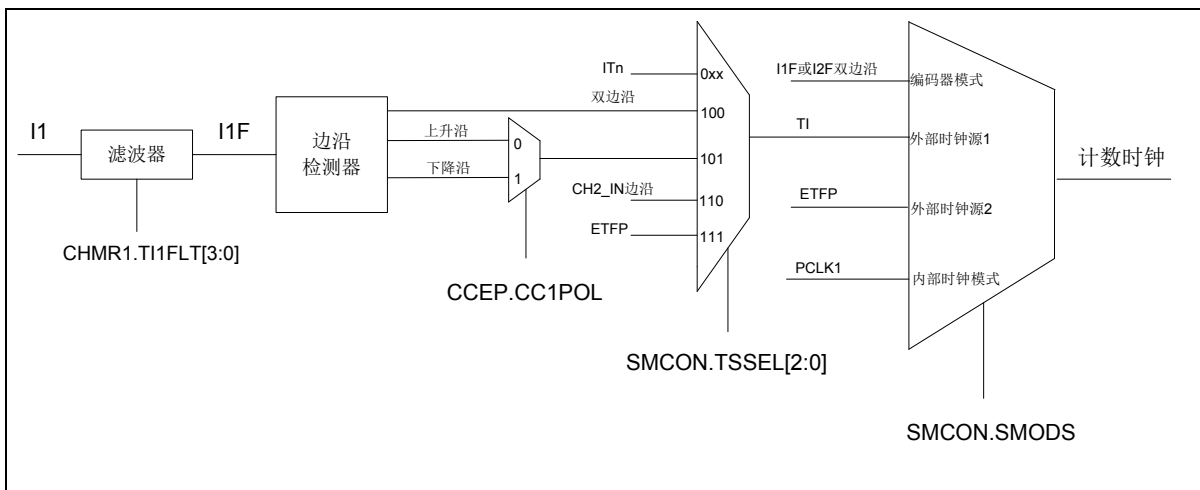


图 25-4 I1 外部时钟连接

配置计数器为外部时钟源 1，步骤如下：

6. GP16C4Tn\_SMCON 寄存器中 SMODS = "111"，配置定时器外部时钟模式 1。
7. 设置 GP16C4Tn\_SMCON 寄存器中的 TSSEL 选择外部时钟源。
8. 如外部时钟源为 I1，可配置 GP16C4Tn\_CHMR1 寄存器 CC1SSEL = "01"，配置通道 1 检测 I1 输入的上升沿；设置 GP16C4Tn\_CCEP 寄存器中 CC1POL = '0'，选择极性为上升沿。
9. 写 GP16C4Tn\_CHMR1 寄存器的 I1FLT[3: 0]位，配置输入滤波器时间（若没有滤波器需求，维持 I1FLT = "0000"）。
10. GP16C4Tn\_CON1 寄存器中 CNTEN = '1'，使能计数器。

当 I1 上出现一次上升沿时，计数器计数一次且 TRGIF 标志位置位。

### 25.4.2.3 外部时钟源 2

置位 GP16C4Tn\_SMCON 寄存器的 ECM2EN 位选定外部时钟源 2。

计数器可对外部触发输入 ET 进行上升沿或下降沿计数。

下图给出了外部输入输入模块的概况。

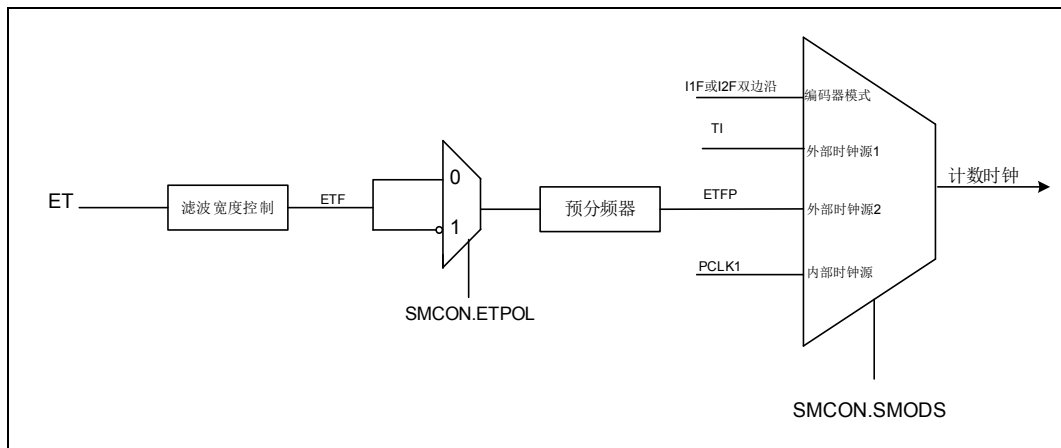


图 25-5 外部触发输入模块

配置计数器为外部时钟源 2，配置过程如下：

6. 设置 GP16C4Tn\_SMCON 寄存器的 ETFLT[3: 0]，配置输入滤波时间。
7. 设置 GP16C4Tn\_SMCON 寄存器中 ETPSEL[1: 0]，设置预分频器。
8. 设置 GP16C4Tn\_SMCON 寄存器中 ETPOL，检测 ET 引脚上升沿或下降沿。
9. 设置 GP16C4Tn\_SMCON 寄存器中 ECM2EN = '1'，使能外部时钟模式 2。
10. 设置 GP16C4Tn\_CON1 寄存器的 CNTEN = '1'，使能计数器。

计数器每两个上升沿计一次数。

#### 25.4.2.4 内部触发输入 (ITn)

当 GP16C4Tn\_SMCON 寄存器的 SMODS = "111", 选定内部触发模式。计数器根据选定的内部输入端的上升或下降沿计数。

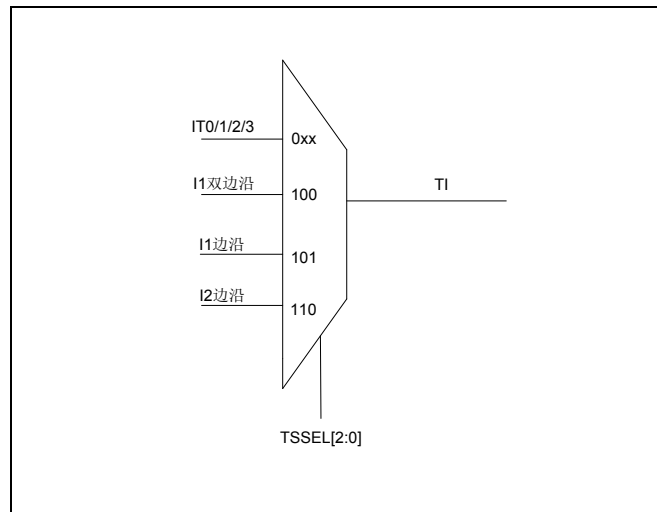


图 25-6 ITn 外部时钟连接

配置计数器在 ITn 输入端的上升沿递增计数，步骤如下：

4. GP16C4Tn\_SMCON 寄存器中 SMODS = "111", 配置外部时钟模式 1。
5. GP16C4Tn\_SMCON 寄存器的 TSSEL = "0xx", 选定 ITn 作为触发输入源。
6. GP16C4Tn\_CON1 寄存器的 CNTEN = '1', 使能计数器。

ITn 产生上升沿时，计数器计数一次。ITn 上升沿与实际时钟间的延时，取决于 ITn 输入的再同步电路，一般为 2~3 个定时器模块时钟周期。

### 25.4.3 计数器模式

#### 25.4.3.1 递增计数模式

在递增模式下，当 GP16C4Tn\_REPAR 寄存器值为 0 时，计数器从 0 开始递增，直至 GP16C4Tn\_AR 寄存器值；然后从 0 重新开始计数并产生一个更新事件（UEV）。当 GP16C4Tn\_REPAR 寄存器不为 0 时，则在 GP16C4Tn\_REPAR+1 次计数后产生更新事件。

当有更新事件（UEV）产生时，预装载寄存器会更新到影子寄存器，更新标志位（GP16C4Tn\_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）：

- ◇ 更新 GP16C4Tn\_REPAR 寄存器的值到影子寄存器
- ◇ 更新 GP16C4Tn\_AR 寄存器的值到影子寄存器
- ◇ 更新 GP16C4Tn\_PRES 寄存器的值到影子寄存器

下图为 GP16C4Tn\_REPAR=0x0，GP16C4Tn\_AR = 0x16，预分频设为 2 分频时的计数器时序。

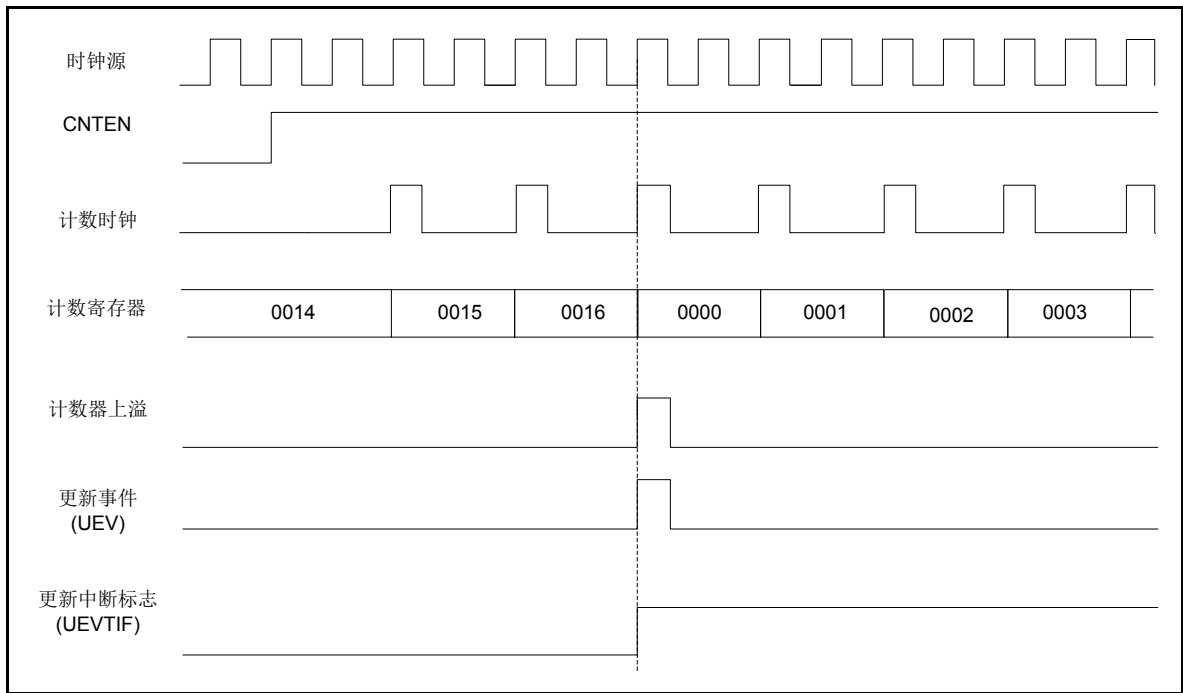


图 25-7 计数器时序图，内部时钟除以 1

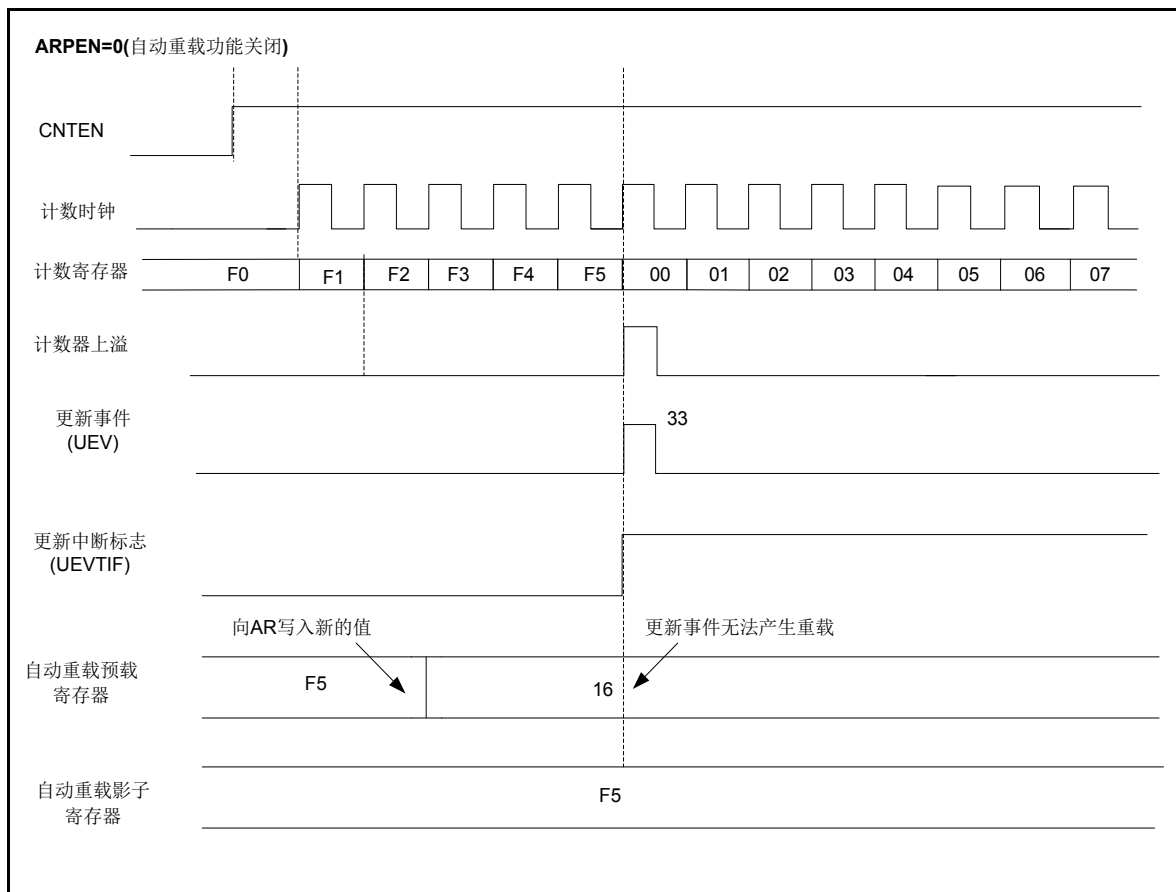


图 25-8 当 ARPEN=0 时计数器时序图

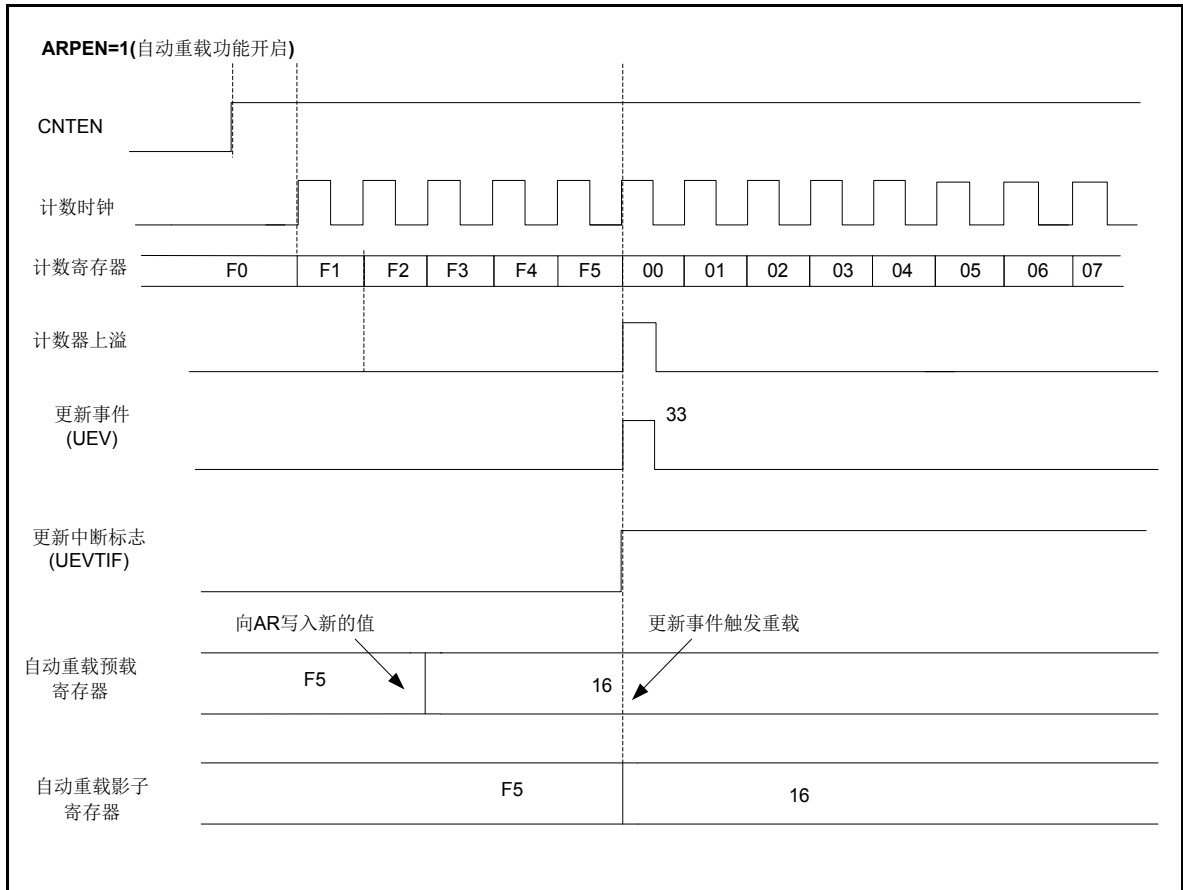


图 25-9 当 ARPEN=1 时计数器时序图

### 25.4.3.2 递减计数模式

在递增模式下，当 GP16C4Tn\_REPAR 寄存器值为 0 时，计数器从 GP16C4Tn\_AR 寄存器值开始递减至 0；然后重复递减并产生更新事件（UEV）。当 GP16C4Tn\_REPAR 寄存器不为 0 时，则在 GP16C4Tn\_REPAR+1 次后产生更新事件。

置位 GP16C4Tn\_SGE 寄存器中的 SGU 位（通过软件或使用从机模式控制器）同样会产生更新事件。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP16C4Tn\_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

下图为 GP16C4Tn\_REPAR=0x0，GP16C4Tn\_AR = 0x27，预分频设为 1 分频时的计数器时序。

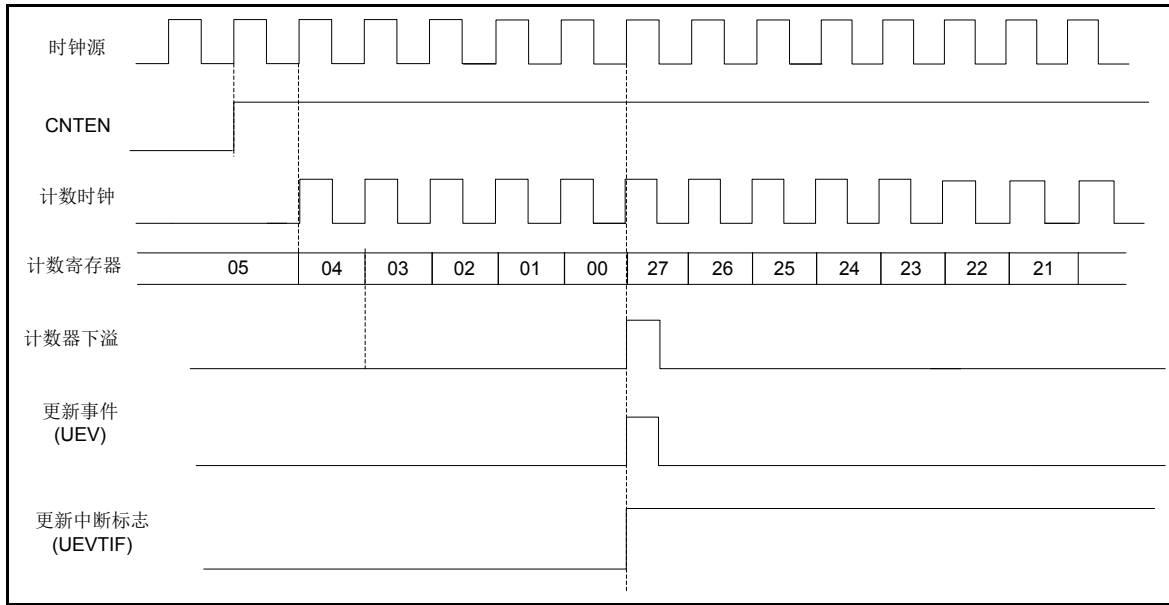


图 25-10 定时器递减计数时序图

### 25.4.3.3 中心对齐模式

当 GP16C4Tn\_CON1 寄存器的 CMSEL 位的值不等于“00”时，定时器工作在中心对齐模式。定时器配置为中心对齐模式时，计数器先从 0 开始递增至 GP16C4Tn\_AR 寄存器值-1，并产生更新事件（UEV）；接着计数器从 GP16C4Tn\_AR 寄存器值递减至 1，并产生下溢事件。如此循环计数。在计数器递减计数（中心对称模式 1，CMSEL=“01”）、计数器递增计数（中心对称模式 2，CMSEL=“10”）、计数器递增和递减计数（中心对称模式 3，CMSEL=“11”）模式下，当通道配置为输出模式时，其输出比较中断标志位会置位。

在中心对齐模式下，GP16C4Tn\_CON1 寄存器的 DIRSEL 位无法进行写操作，该位由硬件自动更新指示当前计数方向。

计数上溢、下溢或者置位 GP16C4Tn\_EGR 寄存器的 SGU 位（通过软件或使用从模式控制器）都会产生更新事件。因此，计数器和预分频器都会从 0 开始计数。

软件置位 GP16C4Tn\_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭时，可避免向预载寄存器写新值时更新影子寄存器。DISUE 复位之前都不会产生更新事件。而在正常产生更新事件时，计数器仍然从 0 开始，同样预分频计数也是从 0 开始（但预分频值没有改变）。此外，若置位 GP16C4Tn\_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。这就避免了在捕获事件时，清除计数器值时产生更新和捕获中断。

当有更新事件（UEV）产生时，预载寄存器值会更新到影子寄存器，更新标志位（GP16C4Tn\_RIF 寄存器中的 UEVTIF 位）置位（取决于 UERSEL 位）。

注：若更新源为计数上溢，自动重载会在计数器重载前更新。因此，下一周期即为预期值（计数器载入新值）。

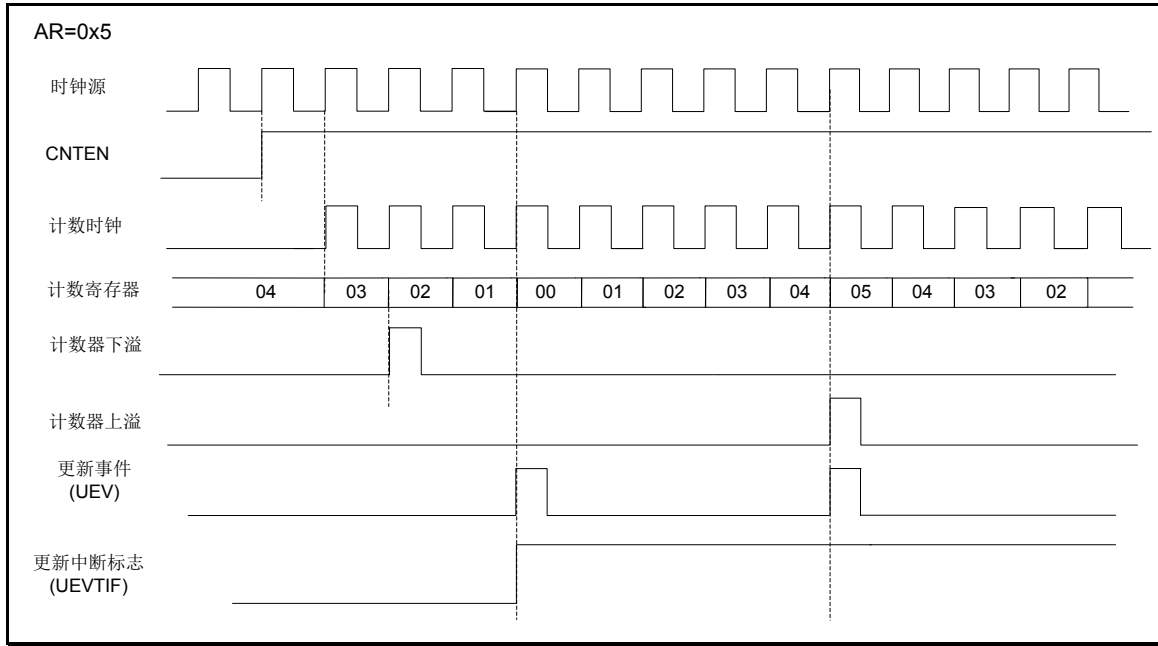


图 25-11 增减计数器时序图

### 25.4.4 捕获/比较通道

以下各图为捕获/比较通道的概述。

输入电路对  $I_n$  输入端的信号进行采样，产生一个经过滤波的信号  $I_nF$ 。之后，一个可极性选择的边沿检测器产生  $I_n$  边沿检出信号，该信号可作为从模式控制器的触发输入或作为捕获控制命令，且信号经过分频后进入捕获寄存器。

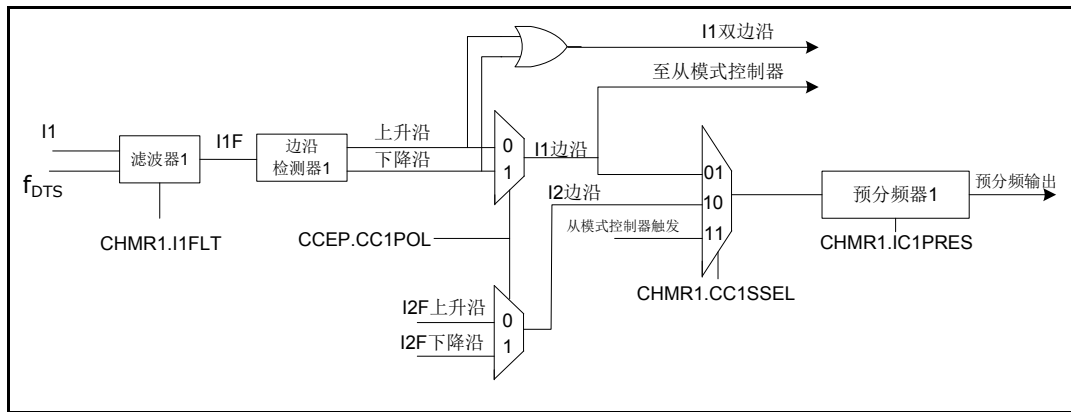


图 25-12 捕获/比较通道

输出部分产生一个中间波形（高有效）作为基准，在输出末端决定最终输出信号的极性。



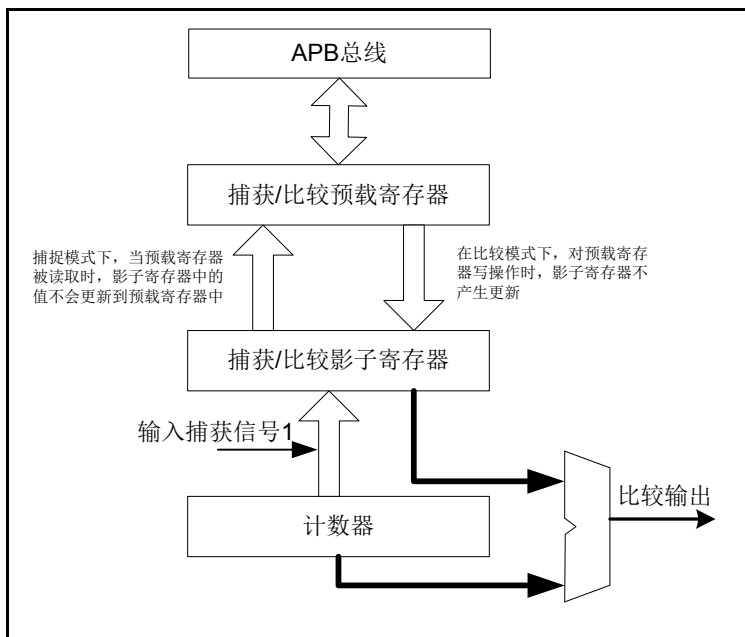


图 25-13 捕获/比较信道 1 主电路

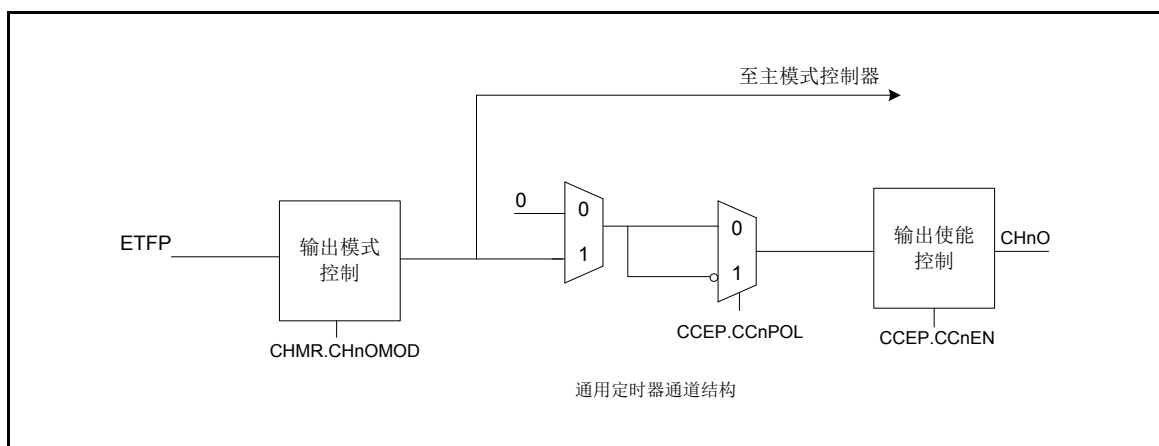


图 25-14 捕获/比较通道的输出阶段

### 25.4.5 输入捕获模式

在输入捕获模式下，当检测到 In 上相应信号变化时，计数器的值就会被锁存到捕获/比较寄存器（GP16C4Tn\_CCVALn）寄存器中。当捕获发生时，相应的 CHnCCIF 标志位（GP16C4Tn\_RIF）会置位，同时会触发中断或 DMA（如果使能）请求。若发生捕获时，CHnCCIF 标志位已经置位，则过捕获 CHnOVIF 标志位（GP16C4Tn\_OVIF）置位。软件写'0'或读取 GP16C4Tn\_CCVALn 寄存器中的捕获值都可以复位 CHnCCIF 标志位。对 CHnOVIF 位写'0'可清空该标志位。

以下为以 I1 输入上升沿作为捕获输入时的流程：

7. 选择有效输入端：GP16C4Tn\_CCVAL1 必须连接到 I1 输入端，因此需将 GP16C4Tn\_CHMR1 寄存器中的 CC1SSEL 位写"01"。只要 CC1SSEL 不为"00"，通道被配置为输入且 GP16C4Tn\_CCVAL1 寄存器为只读。
8. 根据定时器连接的输入信号，配置输入滤波器的持续时间。当输入信号翻转时，前 5 个内部时钟信号是不稳定的，因此必须配置滤波器的时间大于 5 个时钟周期。当 I1 检测到新的电平，连续 8 次采样可确认电平变化有效。
9. 选择 I1 信道的有效边沿变换。GP16C4Tn\_CCEP 寄存器中的 CC1POL 写'0'(上升沿)。
10. 配置输入预分频器。
11. 置位 GP16C4Tn\_CCEP 寄存器中的 CC1EN 位，使能捕获计数器的值到捕获寄存器。
12. 如有需要，置位 GP16C4Tn\_IER 寄存器中的 CC1IT 位，使能中断请求。置位 GP16C4Tn\_DMAEN 寄存器中的 CC1DMA 位，使能 DMA 请求。

当发生输入捕获时：

5. 有效边沿产生，GP16C4Tn\_CCVAL1 寄存器获取计数器的值。
6. CH1CCIF 标志位置位（中断标志）。若至少 2 个连续的捕获发生，但标志位没有及时清除，则 CH1OVIF 也会置位。
7. 中断的产生取决于 GP16C4Tn\_IVS 寄存器的 CC1IT 位。
8. DMA 请求的产生取决于 CC1DMA。

为了处理捕获溢出，建议在读出捕获溢出标志位之前先读取捕获数据。这可以避免丢失在读出捕获标志位之后与读取数据之前可能重复产生的捕获信息。

注：捕获中断请求可由软件设置 GP16C4Tn\_SGE 寄存器中 SGCCnE 位产生。

### 25.4.5.1 PWM输入模式

测量 I1 上 PWM 信号的周期和占空比的过程如下：

1. 为 GP16C4Tn\_CCVAL1 选择有效的输入：GP16C4Tn\_CHMR1 寄存器中的 CC1SSEL 位写"01"（I1 被选择）。
2. 为 I1 边沿检出选择有效的极性（用于捕获数据到 GP16C4Tn\_CCVAL1 寄存器和计数器清零）：CC1POL 位写'0'（上升沿有效）。
3. 为 GP16C4Tn\_CCVAL2 选择有效输入：GP16C4Tn\_CHMR1 寄存器的 CC2SSEL 位写"10"（I1 被选择）。
4. 为 I1 边沿检出选择有效极性（用于捕获数据到 GP16C4Tn\_CCVAL2）：CC2POL 位写'1'（下降沿有效）。
5. 选择有效的触发输入：GP16C4Tn\_SMCON 寄存器的 TSSEL 位写"101"（I1 边沿检出被选择）。
6. 配置从机模式控制器为复位模式：GP16C4Tn\_SMCON 寄存器的 SMODS 位写"100"。
7. 使能捕获：GP16C4Tn\_CCEP 寄存器的 CC1EN 位和 CC2EN 位写'1'。

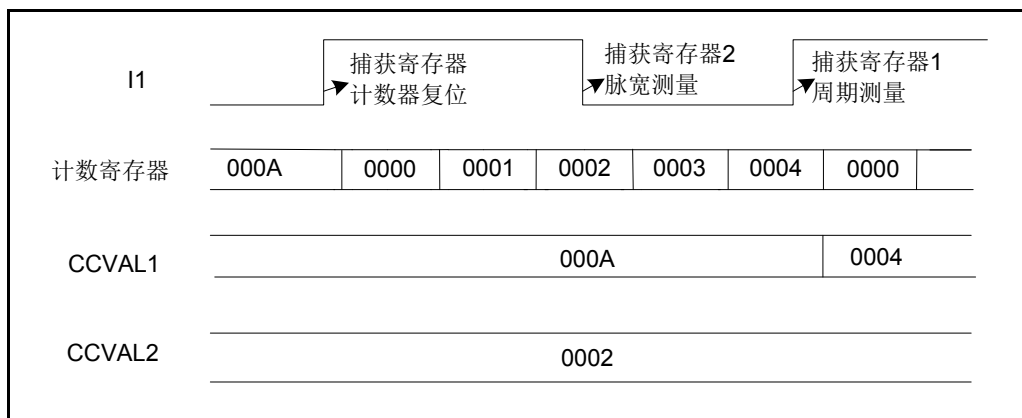


图 25-15 PWM 输入模式时序

### 25. 4. 6 PWM模式

脉宽调制模式可以产生一个 GP16C4Tn\_AR 寄存器值确定频率，GP16C4Tn\_CCVALn 寄存器值确定占空比的信号。

每个通道的 PWM 模式是相互独立的（每个 CHnO 输出一个 PWM），GP16C4Tn\_CHMRn 寄存器的 CHnOMOD 位写"110"（PWM 模式 1）或写"111"（PWM 模式 2）。必须通过置位 GP16C4Tn\_CHMRn 寄存器的 CHnOPREN 位来使能相应的预载寄存器，最后还需置位 GP16C4Tn\_CON1 寄存器的 ARPEN 位来使能自动重装预载功能。

只有当更新事件发生时预载寄存器中的值才会传到影子寄存器，因此，在使能计数前，必须通过置位 GP16C4Tn\_SGE 寄存器的 SGU 位来初始化所有的寄存器。

CHnO 的极性可通过 GP16C4Tn\_CCEP 寄存器的 CCnPOL 位配置，有效极性可配置为高或低。CHnO 的输出使能由 CCnEN 位（GP16C4Tn\_CCEP 寄存器）控制。

在 PWM 模式（1 或 2）中，GP16C4Tn\_COUNT 和 GP16C4Tn\_CCVALn 寄存器的值会持续的比较，确定 GP16C4Tn\_CCVALn <= GP16C4Tn\_COUNT 或 GP16C4Tn\_CCVALn >= GP16C4Tn\_COUNT（取决于计数器的计数方向）。

定时器产生 PWM 波形是边沿对齐或中心对齐，取决于 GP16C4Tn\_CON1 寄存器的 CMSEL 位。

#### 25. 4. 6. 1 PWM边沿对齐模式

##### 1. 递增计数配置

当 GP16C4Tn\_CON1 寄存器的 DIRSEL 位为低时，计数器递增计数。

下图给出了 GP16C4Tn\_AR = 8 时的边沿对齐 PWM 波形。

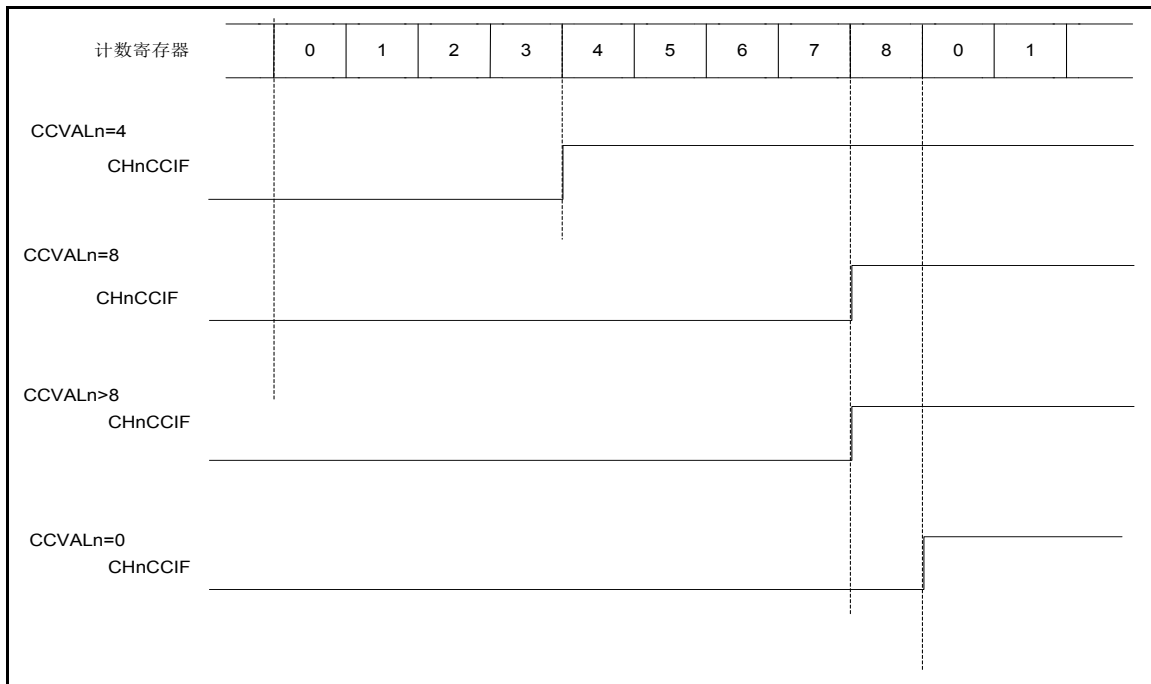


图 25-16 边沿对齐 PWM 波形 (AR=8)

## 2. 递减计数配置

当 GP16C4Tn\_CON1 寄存器的 DIRSEL 位为高时，计数器递减计数。

### 25.4.6.2 PWM中心对齐模式

当 GP16C4Tn\_CON1 寄存器中的 CMSEL 位不为"00"时，中心对齐模式有效。计数器是递增、递减计数分别置比较标志位或递增递减都置比较标志位，取决于 CMSEL 位的配置。GP16C4Tn\_CON1 寄存器的方向位 (DIRSEL) 是由硬件更新的，软件无法修改。

下图为中心对齐方式产生的 PWM 波形的例子：

- ◇ GP16C4Tn\_AR=0x3F, GP16C4Tn\_CCVALn=0x3D
- ◇ PWM 模式 1
  - GP16C4Tn\_CON1 寄存器的 CMSEL= "01", 在中心对齐模式 1 下，计数器向下计数时会置位比较标志位。

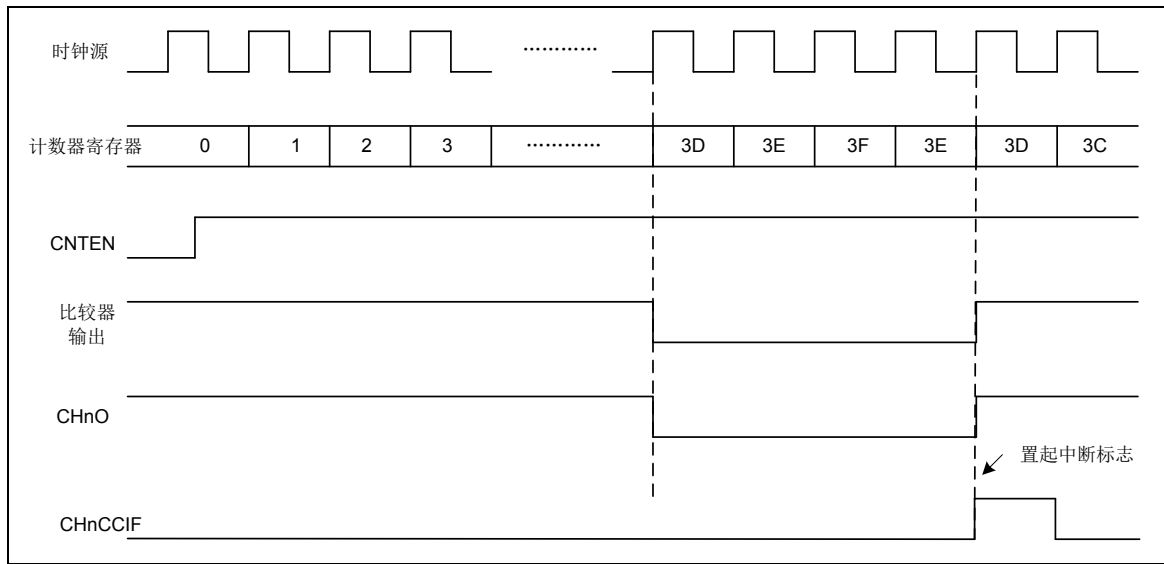


图 25-17 边沿对齐 PWM 波形 (AR=0x3F)

中心对齐模式的使用技巧：

- ◇ 当进入中心对齐模式后，当前递增或递减配置生效。计数器递增或递减计数取决于 GP16C4Tn\_CON1 寄存器的 DIRSEL 位的值。
- ◇ 计数器在中心对齐模式下运行时，对计数器写操作可能导致不可预知的结果。特别是：
  - 若向计数器入的值大于自动重载值 (GP16C4Tn\_COUNT>GP16C4Tn\_AR)，计数方向不更新。例如，如果计数器递增计数，写入值后仍旧递增计数。
  - 若向计数器写 0 或 GP16C4Tn\_AR 中的重载值，则计数方向更新，但并没有产生 UEV。
- ◇ 使用中心对齐模式最安全的方式是计数器开始计数前通过软件产生更新事件（置位 GP16C4Tn\_SGE 寄存器中的 SGU 位）且在计数器运行过程中不对计数器写值。

## 25.4.7 输出比较模式

该功能用于控制输出波形或指示周期时间的结束。

当捕获/比较寄存器和计数器值匹配时，输出比较功能：

- ◇ 输出比较模式（GP16C4Tn\_CHMRn 寄存器中的 CHnOMOD 位）和输出极性（GP16C4Tn\_CCEP 寄存器中的 CCnPOL 位）的配置值输出到对应的引脚上。
- ◇ 中断状态寄存器中的标志位置位（GP16C4Tn\_RIF 寄存器的 CHnCCIF 位）。
- ◇ 若相应的中断掩码置位，则产生中断（GP16C4Tn\_IER 寄存器的 CCnIT 位）。
- ◇ 若相应的使能位置位（GP16C4Tn\_DMAEN 寄存器的 CCnDMA 位，GP16C4Tn\_CON2 寄存器的 CCDMASEL 位用于 DMA 请求的选择），则发送 DMA 请求。

GP16C4Tn\_CHMRn 寄存器中 CHnOPREN 位的值可决定 GP16C4Tn\_CCVALn 寄存器是否带有预装载寄存器。

在输出比较模式中，更新事件 UEV 对 CHnO 的输出没有影响。计时分辨率为计数器的一次计数。输出比较模式同样可以用来输出单个脉冲（单脉冲模式）。

输出比较的配置过程：

1. 选定计数器时钟（内部，外部，预分频）。
2. GP16C4Tn\_AR 与 GP16C4Tn\_CCVALn 寄存器中写入预期值。
3. 若需要产生中断请求，置位 GP16C4Tn\_IER 寄存器中的 CCnIT 位。
4. 选择输出模式，例如：
  - CHnOMOD = "011"，当 CNTV 与 CCRVn 匹配时，CHnO 输出翻转。
  - CHnOPREN = '0'，关闭预载寄存器。
  - CCnPOL = '0'，选择有效极性为高。
  - CCnEN = '1'，使能输出。
5. GP16C4Tn\_CON1 寄存器中的 CNTEN 位置位，使能计数器。

通过配置 GP16C4Tn\_CHMR1 寄存器的 CHnOPREN 位可将 GP16C4Tn\_CCVALn 配置为是否带预装载寄存器。通过软件方式，GP16C4Tn\_CCVALn 寄存器的值可随时更新控制输出波形。

### 25.4.7.1 外部事件清除比较输出

ETFP 输入端（GP16C4Tn\_CHMRn 寄存器的 CHnOCLREN 位写'1'）上的高电平，可将给定通道的比较输出信号拉低。在下次更新事件（UEV）发生前，比较输出会一直保持为低。该功能只能应用在输出比较和 PWM 模式中，强制输出模式中不起作用。

ET 信号可以接到电流控制比较器的输出端。该例中，ET 须按如下流程配置：

1. 外部触发预分频器应该关闭：GP16C4Tn\_SMCON 寄存器的 ETPSEL[1: 0]位应该写"00"
2. 外部时钟源 2 关闭：GP16C4Tn\_SMCON 寄存器的 ECM2EN 位写'0'
3. 外部触发极性（ETPOL）和外部触发滤波器（ETFLT）可根据用户需要配置

### 25.4.8 单脉冲模式

单脉冲模式下，响应某个触发后，定时器的输出通道在可配置的延迟时间后产生一个脉冲，脉冲长度可配。从模式控制器可控制计数器的启动。脉冲波形可在输出比较模式和 PWM 模式下产生。置位 GP16C4Tn\_CON1 寄存器的 SPMEN 位可选择单脉冲模式。计数器会在下次更新事件 UEV 产生时自动停止。

只有比较值不同于计数器初始值时，单脉冲才可以正确的产生。计数器开始计数前（定时器等待触发），必须如下配置：

- ◇ 递增计数：CNT < CCVALn ≤ AR（特别地，0 < CCVALn）
- ◇ 递减计数：CNT > CCVALn

基于 PWM 模式设置单脉冲输出波形的步骤如下：

- ◇ 设置 GP16C4Tn\_CHMRn 寄存器的 CHnOMOD 位，选择 PWM 模式 1 或 2；
- ◇ 设置 GP16C4Tn\_CCEP 寄存器的 CCnPOL 位，选择通道端口 CHnO 的输出极性；
- ◇ 设置 GP16C4Tn\_CON1 寄存器的 DIRSEL，CMSEL，SPMEN 位，配置为递增或递减计数，PWM 普通波形模式，单脉冲模式使能；
- ◇ 设置 GP16C4Tn\_CHMR1 寄存器的 CH1OPREN =1，GP16C4Tn\_CON1 寄存器的 ARPEN =1，使能比较寄存器和计数重载寄存器的缓冲功能（也可以根据实际情况不使能缓冲）；
- ◇ 设置 GP16C4Tn\_CCVALn 寄存器和 GP16C4Tn\_AR 寄存器，配置单脉冲输出延时和脉宽时间；
- ◇ 设置 GP16C4Tn\_SGE 寄存器的 SGU=1 来产生一个更新事件；
- ◇ 设置 GP16C4Tn\_CON1 寄存器的 CNTEN=1 来启动计数器，也可以在触发模式下，通过外部触发输入信号来触发硬件自动设置 CNTEN=1。

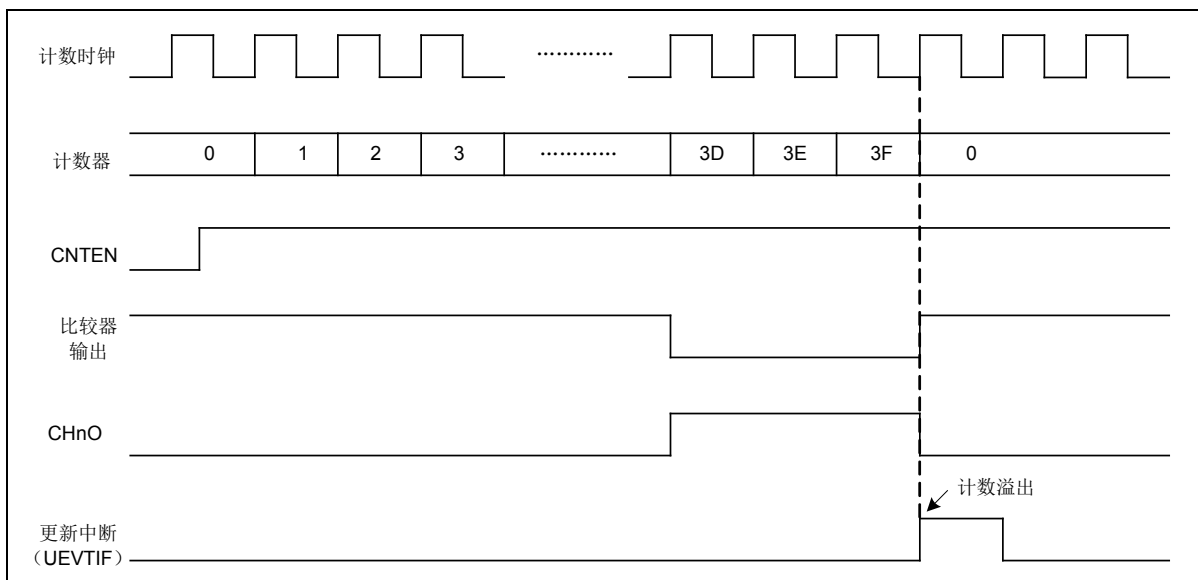


图 25-18 单脉冲模式

### 25.4.9 编码器接口模式

编码器接口模式的三种配置：若计数器只根据 I2 上的边沿计数，则 GP16C4Tn\_SMCON 寄存器中的 SMODS = "001"；若计数器只根据 I1 上的边沿计数，则 GP16C4Tn\_SMCON 寄存器中的 SMODS = "010"；若计数器同时根据 I1 和 I2 上的边沿计数，则 GP16C4Tn\_SMCON 寄存器中的 SMODS = "011"。

配置 GP16C4Tn\_CCEP 寄存器中的 CC1POL 和 CC2POL 位的值可选择 I1 和 I2 的极性。如果需要，也可以配置输入滤波器。

CH1\_IN 和 CH2\_IN 端口作为增量编码器的接口。当计数器使能时，计数器根据 I1 或 I2 上滤波后的有效电平变化时钟计数。I1 和 I2 滤波后的有效信号顺序会产生计数脉冲及方向信号。计数器是递增或递减计数由信号的跳变顺序决定，GP16C4Tn\_CON1 寄存器中的 DIRSEL 计数方向位由自动硬件更新。

编码器接口模式的工作方式类似于一个带有方向选择的外部时钟。计数器在 0 到 GP16C4Tn\_AR 寄存器中的自动重载值之间连续计数。因此，必须在开始计数前配置 GP16C4Tn\_AR 寄存器。同样的，捕获器、预分频器、重复计数器、触发输出的特性正常工作。设定编码模式和选择外部时钟源 2 不兼容，不可以同时选择。

该模式下，计数器会根据增量式编码器的速度和方向自动修改，计数器的值反应的是编码器的位置。计数方向对应着连接传感器的旋转方向。

下表列出了所有的可能组合，假设 I1 和 I2 不同时变换。

有效边沿	有效边沿相对信号的电平 (I1 滤波信号对应 I2, I2 滤波信号对应 I1)	I1 滤波信号边沿		I2 滤波信号边沿	
		上升	下降	上升	下降
仅在 I1 计数	高	下降	上升	不计数	不计数
	低	上升	下降	不计数	不计数
仅在 I2 计数	高	不计数	不计数	上升	下降
	低	不计数	不计数	下降	上升
在 I1 和 I2 上计数	高	下降	上升	上升	下降
	低	上升	下降	下降	上升

表 25-1 计数方向与编码器信号的关系

外部增量编码器可直接与 MCU 连接，无需外部逻辑接口逻辑。而比较器通常用于将编码器的差分输出转换为数字信号，这极大地增加了抗噪声能力。编码器的第三个输出端用于指示机械零点，可以连接到外部中断输入引脚以触发一次计数复位。

下图给出了计数器操作的例子，给出了计数信号了产生和方向控制。同样给出了选择双边沿时，输入抖动如何被补偿。输入抖动可能发生在传感器靠近切换点处。



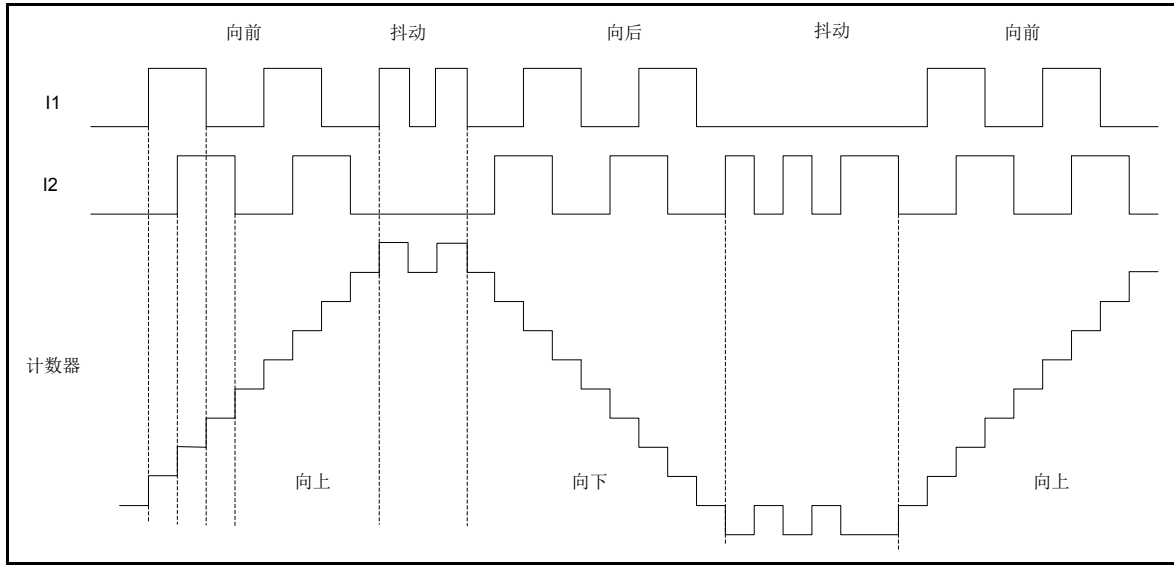


图 25-19 编码器接口模式下的计数操作

下图给出了计数器在 I1 滤波信号极性反相时的计数过程（除了 CC1POL = '1'，其他配置与上面一致）

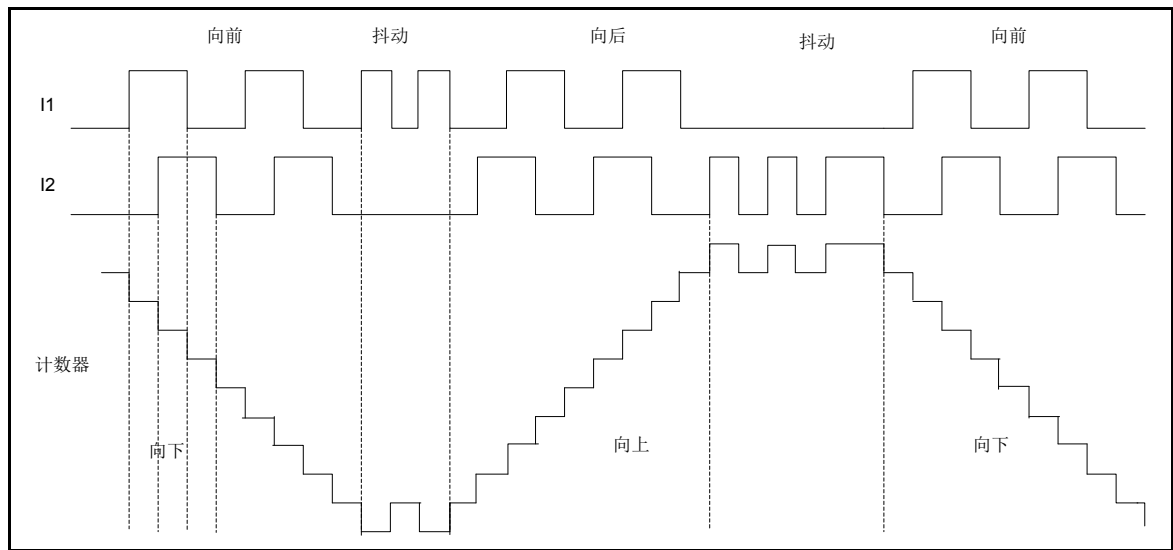


图 25-20 滤波后极性反相时编码器接口

当配置为编码器接口模式时，定时器可提供传感器的当前位置信息。配置一个额外定时器为捕获模式，用于测量两个编码器事件的间隔，根据间隔时长获取动态信息（速度、加速度、减速度）。编码器用于指示机械零点的输出就是此用处。根据编码器两个事件间隔，可以周期性的读取计数器的值。如果允许，可以将计数器值锁存到第三个输入捕获寄存器（捕获信号必须是周期性的且可由其它定时器产生）。条件允许时，可通过实时时钟产生 DMA 请求的方式读取计数器值。

### 25. 4. 10 输入异或功能

通过 GP16C4Tn\_CON2 寄存器中 I1FSEL 位, 可将通道 1 的输入滤波器连接到 XOR 门的输出端, XOR 门联合了 CH1\_IN、CH2\_IN 和 CH3\_IN 三个输入引脚。

XOR 输出用于定时器的所有输入功能, 如触发或输入捕获。

### 25. 4. 11 定时器和外部触发的同步

GP16C4Tn 定时器可在多种模式下与外部触发同步: 复位模式、门控模式及触发模式。

#### 25. 4. 11. 1 复位模式

计数器及其预分频器可以在响应触发输入事件时重新初始化。此外, 若 GP16C4Tn\_CON1 寄存器的 UERSEL 位为低时会产生一次更新事件 UEV。所有预载寄存器 (GP16C4Tn\_AR, GP16C4Tn\_CCVALn) 都会因更新事件 UEV 而被更新。

在下面例子中, I1 输入端的上升沿让递增计数被清空:

- ◇ 配置通道 1 上检测 I1 上的上升沿。配置输入滤波周期 (本例无需滤波器, 故 I1FLT = "0000")。触发捕获分频器没有使用, 无需配置。CC1SSEL 位只选择输入捕获源, GP16C4Tn\_CHMR1 寄存器中 CC1SSEL = "01"。GP16C4Tn\_CCEP 寄存器中 CC1POL = 0 以确定极性 (只检测上升沿)。
- ◇ 定时器配置为复位模式: GP16C4Tn\_SMCON 寄存器中 SMODS = "100"。选择 I1 作为输入源: GP16C4Tn\_SMCON 寄存器中 TSSEL = "101"。
- ◇ 启动计数器: GP16C4Tn\_CON1 寄存器中 CNTEN = '1'。

计数器依据内部时钟开始计数, 正常计数直到 I1 上出现上升沿。当 I1 上出现上升沿时, 计数器会被清零且从 0 重新开始计数。同时, 标志位置位 (GP16C4Tn\_RIF 寄存器中 TRGIF 位), 如果中断及 DMA 使能 (取决于 GP16C4Tn\_IER 寄存器中的 TRGIT 和 GP16C4Tn\_DMAEN 的 TRGDMA 位), 会发送中断及 DMA 请求。

下图给出了当自动重载寄存器 GP16C4Tn\_AR = 0x36 时的信号变化。由于 I1 输入的再同步电路, I1 上的上升沿和计数器实际复位之间存在延时 (包含 2~3 个模块时钟周期的同步延时)。

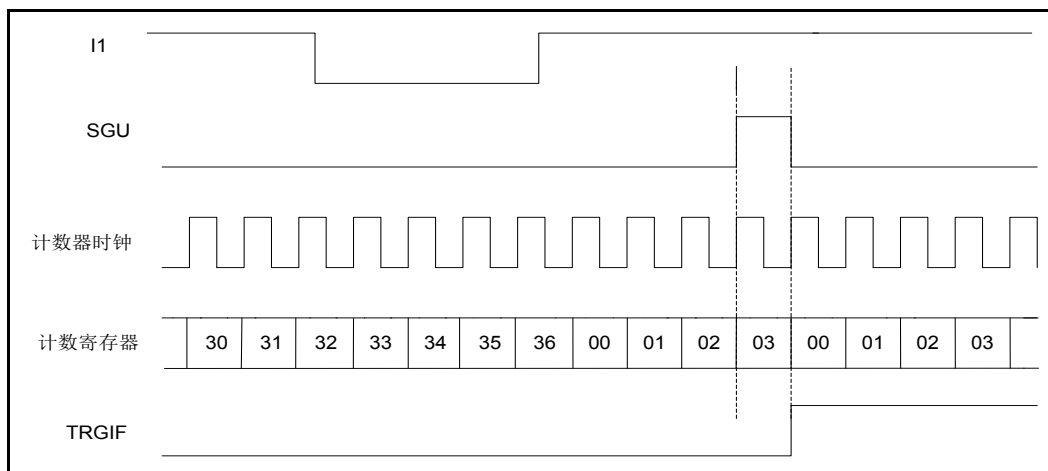


图 25-21 复位模式控制电路

### 25.4.11.2 门控模式

计数器根据选中的输入电平被使能。

下面的例子中，计数器只在 I1 输入为低电平时才递增计数：

- ◇ 配置通道 1 在 I1 上检测低电平。配置输入滤波周期（本例不需要滤波器，I1FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C4Tn\_CHMR1 寄存器中的 CC1SSEL = "01"，选择输入捕获源。GP16C4Tn\_CCEP 寄存器中 CC1POL = '1'，确认极性（只检测低电平）。
- ◇ 配置定时器为门控模式：GP16C4Tn\_SMCON 寄存器中 SMODS = "101"。选择 I1 作为输入源：GP16C4Tn\_SMCON 寄存器中 TSSEL = "101"。
- ◇ 使能计数器：GP16C4Tn\_CON1 寄存器中 CNTEN = '1'（门控模式中，如果 CNTEN = '0'，无论触发输入为何电平，计数器都不会启动）。

只要 I1 为低电平，计数器依据内部时钟开始计数，一旦 I1 为高则停止计数。由于 I1 输入端再同步电路的原因，I1 上出现上升沿和计数器实际停止之间会有一定的延时。

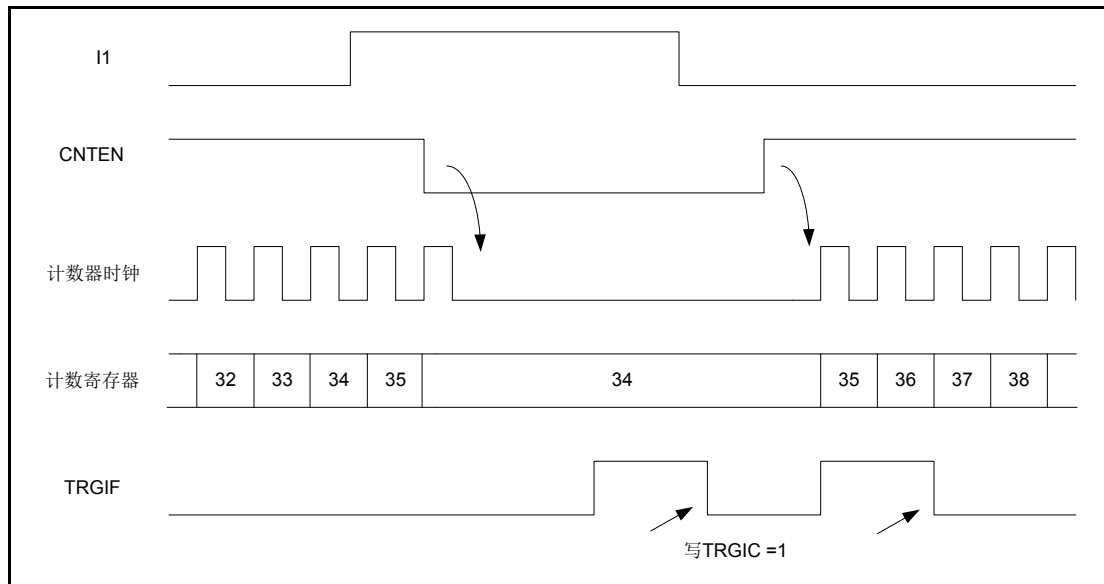


图 25-22 门控模式控制电路

### 25.4.11.3 触发模式

输入端选中的事件可以使能计数器。

下面的例子中，I2 输入端上的上升沿可以启动递增计数：

- ◇ 配置通道 2 可以检测 I2 上的上升沿。配置滤波时间（本例不需要滤波，I2FLT = "0000"）。触发捕获分频器没有使用，无需配置。GP16C4Tn\_CHMR1 寄存器中 CC2SSEL = "01"，用于选择捕获源。GP16C4Tn\_CCEP 寄存器中 CC2POL = '1'，确认极性（只检测低电平）。
- ◇ 配置定时器为触发模式：GP16C4Tn\_SMCON 寄存器中 SMODS = "110"。GP16C4Tn\_SMCON 寄存器中 TSSEL = "110"，用于选择输入源。

I2 上出现上升沿时，计数器开始依据内部时钟计数并置位 TRGIF 标志位。

由于 I2 输入的再同步原因, I2 上出现上升沿和计数器实际停止之间会有一定的延时。

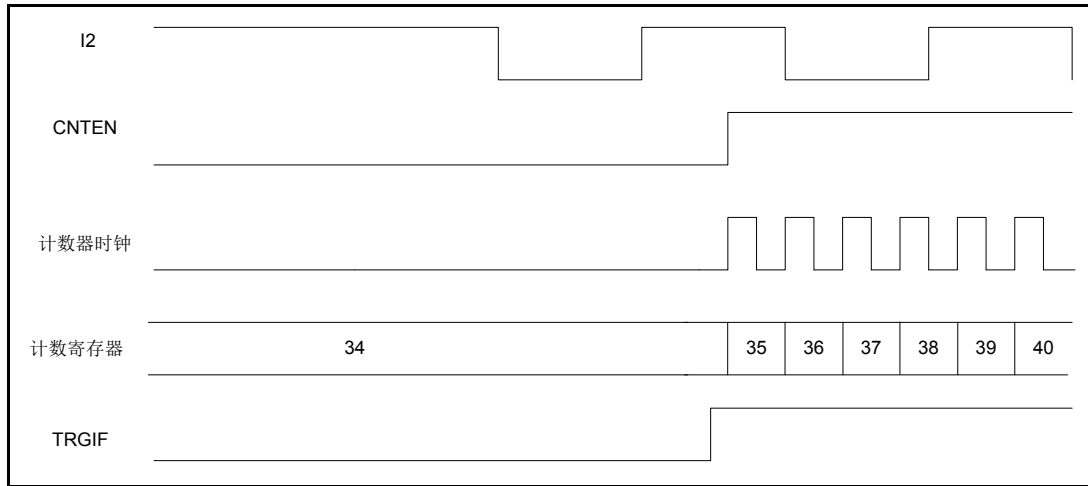


图 25-23 触发模式控制电路

#### 25.4.11.4 选择外部时钟源 2 的触发模式

外部时钟源 2 可和其他模式一起使用（除编码模式）。ET 信号可作为外部时钟输入，另一个输入可选择为触发输入（复位模式、门控模式或触发模式）。不推荐用 GP16C4Tn\_SMCON 寄存器的 TSSEL 位选择 ET 作为 TI。

下面的例子中，一旦 I1 上出现上升沿时，计数器会依据 ET 信号的每个上升沿递增计数。

- ◇ 通过 GP16C4Tn\_SMCON 寄存器，配置外部触发输入电路，过程如下：

ETFLT = "000": 无滤波

ETPSEL = "00": 禁止分频

ETPOL = '0': 检测 ET 的上升沿, ECM2EN = '1'使能外部时钟模式 2

- ◇ 配置通道 1 检测 I 的上升沿，过程如下：

I1FLT = "0000": 无滤波。

触发捕获分频器没有使用，无需配置。

GP16C4Tn\_CHMR1 寄存器中 CC1SSEL = "01"选择输入捕获源，

GP16C4Tn\_CCEP 寄存器的 CC1POL = '0'确认极性（只检测上升沿）。

- ◇ 配置定时器为触发模式：GP16C4Tn\_SMCON 寄存器中 SMODS = "110"。

GP16C4Tn\_SMCON 寄存器中 TSSEL = "101"选择 I1 作为输入源。

I1 上出现上升沿时，计数器使能且 TRGIF 标志位置位，然后计数器根据 ET 上的上升沿开始计数。

由于 ETFP 输入再同步电路的原因，ET 信号的上升沿和实际计数器的复位会有延时。

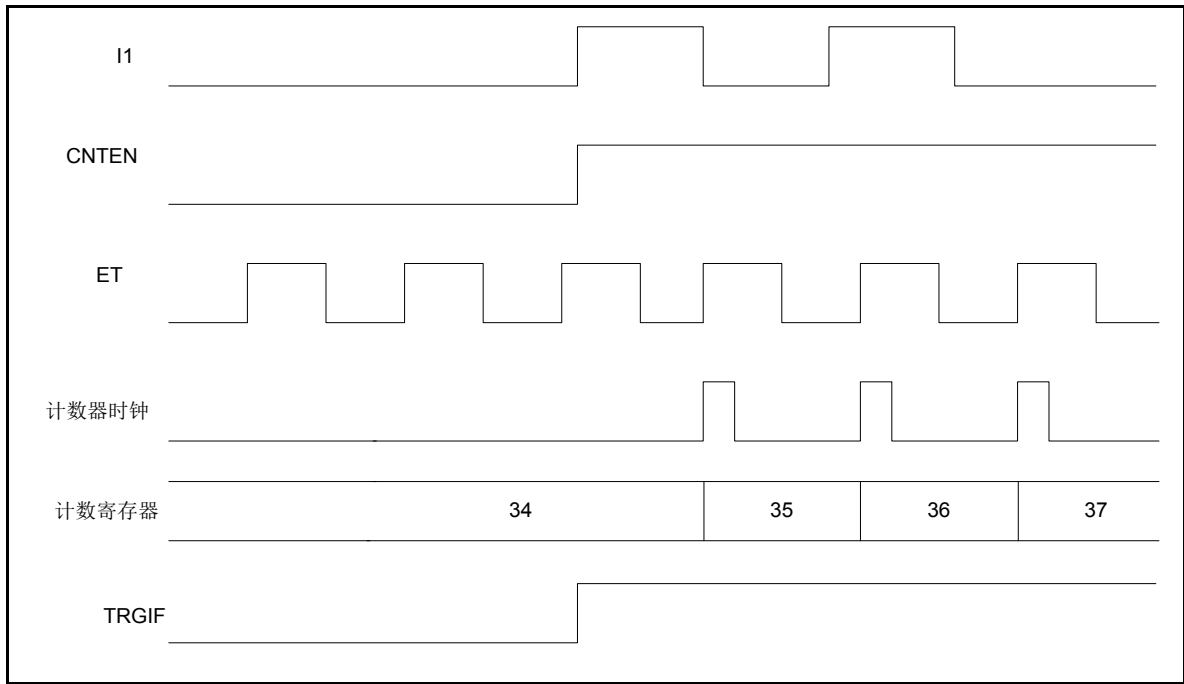


图 25-24 外部时钟源 2+触发模式下的控制电路

### 25.4.12 调试模式

当微控制器进入调试模式（Cortex™-M3 内核停止），GP16C4Tn 计数器停止计数。

## 25.5 特殊功能寄存器

### 25.5.1 寄存器列表

TIM0_1 寄存器列表		
寄存器名称	偏移地址	寄存器描述
TIM0 基地址: 4000_0000 <sub>H</sub>		
TIM1 基地址: 4000_0400 <sub>H</sub>		
GP16C4Tn_CON1	000 <sub>H</sub>	控制寄存器 1
GP16C4Tn_CON2	004 <sub>H</sub>	控制寄存器 2
GP16C4Tn_SMCON	008 <sub>H</sub>	从模式控制寄存器
GP16C4Tn_IER	00C <sub>H</sub>	中断使能寄存器
GP16C4Tn_IDR	010 <sub>H</sub>	中断禁止寄存器
GP16C4Tn_IVS	014 <sub>H</sub>	中断有效状态寄存器
GP16C4Tn_RIF	018 <sub>H</sub>	原始中断标志寄存器
GP16C4Tn_IFM	01C <sub>H</sub>	中断标志屏蔽寄存器
GP16C4Tn_ICR	020 <sub>H</sub>	中断清零计算器
GP16C4Tn_SGE	024 <sub>H</sub>	事件生成计算器

TIM0_1 寄存器列表		
寄存器名称	偏移地址	寄存器描述
TIM0 基地址: 4000_0000 <sub>H</sub>		
TIM1 基地址: 4000_0400 <sub>H</sub>		
GP16C4Tn_CHMR1	028 <sub>H</sub>	捕获/比较模式寄存器 1
GP16C4Tn_CHMR2	02C <sub>H</sub>	捕获/比较模式寄存器 2
GP16C4Tn_CCEP	030 <sub>H</sub>	捕获/比较使能寄存器
GP16C4Tn_COUNT	034 <sub>H</sub>	计数器
GP16C4Tn_PRES	038 <sub>H</sub>	预分频器
GP16C4Tn_AR	03C <sub>H</sub>	自动重载寄存器
GP16C4Tn_CCVAL1	044 <sub>H</sub>	捕获/比较寄存器 1
GP16C4Tn_CCVAL2	048 <sub>H</sub>	捕获/比较寄存器 2
GP16C4Tn_CCVAL3	04C <sub>H</sub>	捕获/比较寄存器 3
GP16C4Tn_CCVAL4	050 <sub>H</sub>	捕获/比较寄存器 4
GP16C4Tn_DMAEN	058 <sub>H</sub>	DMA 使能寄存器

## 25.5.2 寄存器描述

### 25.5.2.1 控制寄存器 1 (GP16C4Tn\_CON1)

控制寄存器 1 (GP16C4Tn_CON1)																															
偏移地址: 000 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						DFCKSEL	ARPEN	CMSEL	DIRSEL	SPMIEN	UIRSEL	DISUE	CNTEN		

Reserved	Bit 31-10	-	保留, 必须保持为复位值
DFCKSEL	Bit 9-8	RW	<p><b>时钟分频</b> 该时钟分频为定时器 (INT_CLK) 频率与死区时间生成器和数字滤波器 (ET, In) 采用的死区时间和采样时钟 (tDTS) 之间的分频比。</p> <p>00: <math>t_{DTS}=t_{INT\_CLK}</math> 01: <math>t_{DTS}=2*t_{INT\_CLK}</math> 10: <math>t_{DTS}=4*t_{INT\_CLK}</math> 11: 保留</p>
ARPEN	Bit 7	RW	<p><b>自动重载预载使能</b> 0: GP16C4Tn_AR 寄存器未缓冲 1: GP16C4Tn_AR 寄存器被装入缓冲器</p>
CMSEL	Bit 6-5	RW	<p><b>中央对齐模式选择</b> 00: 边沿对齐模式。计数器根据方向为 (DIRSEL) 来向上或向下计数。 01: <b>中央对齐模式 1</b>。计数器以交替方式向上或向下计数。仅当计数器向下计数时, 配置为输出的通道 (GP16C4T_CHMRn 寄存器中 CCnSSEL=00) 的输出比较中断标志位才会被设置。 10: <b>中央对齐模式 2</b>。计数器以交替方式向上或向下计数。仅当计数器向上计数时, 配置为输出的通道 (GP16C4T_CHMRn 寄存器中 CCnSSEL=00) 的输出比较中断标志位才会被设置。 11: <b>中央对齐模式 3</b>。计数器以交替方式向上或向下计数。当计数器向上或向下计数时, 配置为输出的通道 (GP16C4T_CHMRn 寄存器中 CCnSSEL=00) 的输出比较中断标志位均会被设置。 注意: 当计数器使能时 (CNTEN=1), 不允许从边沿对齐模式转换到中央对齐模式</p>

DIRSEL	Bit 4	R/W	<p><b>计数器方向选择</b></p> <p>0: 计数器向上计数 1: 计数器向下计数</p> <p>注意: 当计数器配置为中央对齐模式或者编码器模式时, 该位只读。</p>
SPMEN	Bit 3	R/W	<p><b>单脉冲模式</b></p> <p>0: 当发生更新事件时, 计数器不停止。 1: 当发生下一次更新事件 (CNTEN 位清零) 时, 计数器停止。</p>
UERSEL	Bit 2	R/W	<p><b>更新请求源</b></p> <p>该位由软件置 1 或清零, 来选择 UEV 事件源。</p> <p>0: 如果更新中断或 DMA 请求使能, 则下述任一事件都可产生更新中断或 DMA 请求:</p> <ul style="list-style-type: none"> <li>- 计数器上溢/下溢</li> <li>- 设置 SGU 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>1: 如果更新中断或 DMA 请求使能, 仅计数器上溢/下溢才能产生更新中断或 DMA 请求中断</p>
DISUE	Bit 1	R/W	<p><b>更新禁止</b></p> <p>该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。</p> <p>0: UEV 使能. 更新事件 (UEV) 由下列任一事件产生:</p> <ul style="list-style-type: none"> <li>- 计数器上溢/下溢</li> <li>- 设置 SGU 位</li> <li>- 从模式控制器产生的更新</li> </ul> <p>缓冲寄存器载入他们的预载值。</p> <p>1: UEV 禁止. 不产生更新事件, 影子寄存器保持他们的值 (ARRV, PSCV, CCRVx). 如果从从模式控制器接收到硬件复位, 计数器和预分频器将被重新初始化。</p>
CNTEN	Bit 0	R/W	<p><b>计数器使能</b></p> <p>0: 计数器禁止 1: 计数器使能</p> <p>注意: 如果软件设置了 CNTEN 位, 外部时钟, 门控模式和编码器模式才能工作。触发模式可由硬件自动设置 CNTEN 位。</p>



### 25.5.2.2 控制寄存器 2 (GP16C4Tn\_CON2)

控制寄存器 2 (GP16C4Tn_CON2)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							I1FSEL	TRGOSEL			CCDMASEL	Reserved			

Reserved	Bit 31-8	-	保留, 必须保持为复位值
I1FSEL	Bit 7	RW	<p><b>I1 选择</b></p> <p>0: GP16C4Tn_CH1 引脚与 I1 输入连接</p> <p>1: GP16C4Tn_CH1, CH2 和 CH3 引脚与 I1 输入 (XOR) 连接</p>
TRGOSEL	Bit 6-4	RW	<p><b>选择主模式 TRGOUT 输出</b></p> <p>为同步 (TRGOUT), 该位可选择在主模式下发送至从计数器的信息。</p> <p><b>000: 复位</b>-GP16C4Tn_SGE 寄存器中的 SGU 位被采用为触发输出 (TRGOUT)。如果复位由触发输入生成 (从模式控制器配置复位模式), 则相较于实际复位, TRGOUT 上的信号将会延迟。</p> <p><b>001: 使能</b>-计数器使能信号被用作触发输出 (TRGOUT)。在从计数器使能的情况下, 该设置用于在同一时间启动数次或者用来控制窗口。计数器使能信号是由 CNTEN 控制位与配置为门控模式的触发输入进行 OR 操作产生的。当计数器使能信号由触发输入控制, TRGOUT 上会产生延迟, 除非被选为主/从模式 (参考 GP16C4Tn_SMCON 寄存器中的 MSCFG 位的描述)。</p> <p><b>010: 更新</b>-更新事件被选为触发输出 (TRGOUT)。举例, 主计数器可被用作从计数器的预分频器。</p> <p><b>011: 比较脉冲</b>-一旦捕获或者比较匹配发生, 当 CH1CCIF 标志位被置起 (即便已为高电平), 触发输出会发送一个正脉冲。</p> <p><b>100: 比较</b>- 通道 1 比较输出信号用作触发输出 TRGOUT</p> <p><b>101: 比较</b>- 通道 2 比较输出信号用作触发输出 TRGOUT</p> <p><b>110: 比较</b>- 通道 3 比较输出信号用作触发输出 TRGOUT</p>

			111: 比较- 通道 4 比较输出信号用作触发输出 TRGOUT
CCDMASEL	Bit 3	RW	捕获/比较 DMA 选择 0: 当 CCn 事件发生, 会发出 CCn DMA 请求。 1: 当发生更新时间, 会发出 CCn DMA 请求。
Reserved	Bit 2-0	-	保留, 必须保持为复位值

25.5.2.3 从模式控制寄存器 (GP16C4Tn\_SMCON)

从模式控制寄存器 (GP16C4Tn_SMCON)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ETPOL	ECM2EN	ETPSEL	ETFLT				MSCFG	TSSEL			Reserved	SMODS			

Reserved	Bit 31-16	-	保留, 必须保持为复位值
ETPOL	Bit 15	R/W	外部触发极性 0: 正向 ET, 高电平有效或上升沿有效 1: 反正 ET, 低电平有效或下降沿有效
ECM2EN	Bit 14	R/W	使能外部时钟模式 2 该位使能外部时钟模式 2 0: 禁止外部时钟模式 2 1: 使能外部时钟模式 2。计数器由 ETFP 信号上的有效边沿计数。 注意: 1. 设置 ECM2EN 位与选择外部时钟模式 1 且 TI 与 ETFP 相连接 (SMODS=111 和 TSSEL=111) 具有相同的效果。 2. 可同时使用外部时钟模式 2 与下列从模式: 复位模式, 门控模式和除法模式。在这种情况下, TI 不能与 ETFP 相连接 (TSSEL 不能设置为 111)。 3. 如果外部时钟模式 1 和外部时钟模式 2 同时使能, 外部时钟输入为 ETFP。
ETPSEL	Bit 13-12	R/W	外部触发预分频器 外部触发信号频率最大为 GP16C4TnCLK 频率的 1/4。可使能预分频器来减小 ETFP 频率。该位有效用于输入高速外部时钟的情况。 00: 预分频器关闭 01: ETFP 频率 2 分频 10: ETFP 频率 4 分频 11: ETFP 频率 8 分频
ETFLT	Bit 11-8	R/W	外部触发滤波器 该位定义了 ETFP 信号的采样频率和数字滤波器的滤波长度。 数字滤波器由一个事件计数器组成, 每 N 个连续事件才视为一个有效边沿。 0000: 无滤波器, 采样频率为 $f_{DTS}$ 0001: $f_{SAMPLING} = f_{INT\_CLK}$ , $N = 2$

			<p>0010: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}}, N = 4</math>          0011: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}}, N = 8</math>          0100: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 2, N = 6</math>          0101: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 2, N = 8</math>          0110: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 4, N = 6</math>          0111: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 4, N = 8</math>          1000: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 8, N = 6</math>          1001: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 8, N = 8</math>          1010: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 16, N = 5</math>          1011: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 16, N = 6</math>          1100: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 16, N = 8</math>          1101: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 32, N = 5</math>          1110: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 32, N = 6</math>          1111: <math>f_{\text{SAMPLING}} = f_{\text{INT\_CLK}} / 32, N = 8</math>          注意: 当ETFLT[3:0] = 1, 2或3时, 公式中的<math>f_{\text{DTS}}</math>由INT_CLK 取代。</p>
MSCFG	Bit 7	RW	<p><b>主/从模式</b>          0: 无动作          1: 延迟触发输入 (In) 上的事件来允许当前计时器和其从器件之间的同步。该设置有效用于使用单个外部事件来同步多个计时器。</p>
TSSEL	Bit 6-4	RW	<p><b>触发选择</b>          该位用来选择不同的触发输入来同步计数器。          000: 内部触发 0 (IT0)          001: 内部触发 1 (IT1)          010: 内部触发 2 (IT2)          011: 内部触发 3 (IT3)          100: I1 边沿检测器 (I1F_ED)          101: 滤波计时器输入 1          110: 滤波计时器输入 2          111: 外部触发输入          注意: 为了避免错误边沿检测, 该位在不使用时 (SMODS=000) 才能改变。</p>
Reserved	Bit 3	-	保留, 必须保持为复位值
SMODS	Bit 2-0	RW	<p><b>选择从模式功能</b>          当选择外部信号, 触发信号TI的有效边沿与外部输入的极性有关系 (详见输入控制寄存器和控制寄存器描述)          000: <b>禁止从模式</b>—如果CNTEN = '1', 则预分频器直接由内部时钟计数。          001: <b>编码器模式1</b>—计数器向上/向下计数I2边沿, 取决于I1电平。          010: <b>编码器模式2</b>—计数器向上/向下计数I1边沿, 取决于I2电平</p>

		<p><b>011 :编码器模式3</b> -计数器向上/向下计数I1边沿检出和I2边沿检出边沿，取决于另一个输入的电平。</p> <p><b>100 : 复位模式</b>-选中的触发输入的上升沿重新初始化计数器，生成寄存器的更新</p> <p><b>101 : 门控模式</b>-当触发输入TI为高电平，计数器时钟使能。一旦触发变为低电平，计数器停止计数（并非复位）。计数器的启动和停止均受控制。</p> <p><b>110 : 触发模式</b>-计数器在触发信号TI的上升沿处启动（不复位）。仅寄存器的启动受控制。</p> <p><b>111 : 外部时钟模式1</b>-计数器在TI的上升沿计数</p> <p>注意：如果I1双边沿检出被选为触发输入（TSSEL='100'），不能使用门控模式。I1每一次转换，I1双边沿检出就会输出1个脉冲，而门控模式则是检查触发信号的电平。</p> <p>注意：在发生来自自主计时器的接收事件之前，从计时器的时钟必须先使能，且在接收来自自主计时器的触发过程中，从计数器时钟不能即时更改。</p>
--	--	--

25.5.2.4 中断使能寄存器 (GP16C4Tn\_IER)

中断使能寄存器 (GP16C4Tn_IER)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT		

Reserved	Bit31-13	-	保留, 必须保持复位值。
CC4OIT	Bit12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8-7	-	保留, 必须保持为复位值
TRGIT	Bit6	W	使能触发中断 0: 无效 1: 使能
Reserved	Bit 5	-	保留, 必须保持为复位值
CC4IT	Bit4	W	使能捕获/比较 4 中断 0: 无效 1: 使能
CC3IT	Bit3	W	使能捕获/比较 3 中断 0: 无效 1: 使能
CC2IT	Bit2	W	使能捕获/比较 2 中断 0: 无效 1: 使能
CC1IT	Bit1	W	使能捕获/比较 1 中断 0: 无效 1: 使能
UIT	Bit0	W	使能更新事件中断 0: 无效 1: 使能

### 25.5.2.5 中断禁止寄存器 (GP16C4Tn\_IDR)

中断禁止寄存器 (GP16C4Tn_IDR)																															
偏移地址: 0010 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT								

Reserved	Bit 31-13	-	保留, 必须保持复位值。
CC4OIT	Bit12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8-7	-	保留, 必须保持为复位值
TRGIT	Bit 6	W	禁止触发中断 0: 无效 1: 禁止
Reserved	Bit 5	-	保留, 必须保持为复位值
CC4IT	Bit 4	W	禁止捕获/比较 4 中断 0: 无效 1: 禁止
CC3IT	Bit 3	W	禁止捕获/比较 3 中断 0: 无效 1: 禁止
CC2IT	Bit 2	W	禁止捕获/比较 2 中断 0: 无效 1: 禁止
CC1IT	Bit 1	W	禁止捕获/比较 1 中断 0: 无效 1: 禁止
UIT	Bit 0	W	禁止更新中断 0: 无效 1: 禁止

### 25.5.2.6 中断有效状态寄存器 (GP16C4Tn\_IVS)

中断有效状态寄存器 (GP16C4Tn_IVS)																															
偏移地址: 0014 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC4OIT	CC3OIT	CC2OIT	CC1OIT	Reserved	TRGIT	Reserved	CC4IT	CC3IT	CC2IT	CC1IT	UIT								

Reserved	Bit 31-13	-	保留
CC4OIT	Bit12	W	使能捕获/比较 4 捕获溢出中断 0: 无效 1: 使能
CC3OIT	Bit11	W	使能捕获/比较 3 捕获溢出中断 0: 无效 1: 使能
CC2OIT	Bit10	W	使能捕获/比较 2 捕获溢出中断 0: 无效 1: 使能
CC1OIT	Bit9	W	使能捕获/比较 1 捕获溢出中断 0: 无效 1: 使能
Reserved	Bit 8-7	-	保留, 必须保持为复位值
TRGIT	Bit 6	R	触发中断状态 0: 禁止 1: 使能
Reserved	Bit 5	-	保留, 必须保持为复位值
CC4IT	Bit 4	R	通道 4 捕获/比较中断状态 0: 禁止 1: 使能
CC3IT	Bit 3	R	通道 3 捕获/比较中断状态 0: 禁止 1: 使能
CC2IT	Bit 2	R	通道 2 捕获/比较中断状态 0: 禁止 1: 使能
CC1IT	Bit 1	R	通道 1 捕获/比较中断状态 0: 禁止 1: 使能
UIT	Bit 0	R	更新事件中断状态 0: 禁止 1: 使能



25.5.2.7 原始中断标志寄存器 (GP16C4Tn\_RIF)

原始中断标志寄存器 (GP16C4Tn_RIF)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OVIF	CH3OVIF	CH2OVIF	CH1OVIF	Reserved	TRGIF	Reserved	CH4CCIF	CH3CCIF	CH2CCIF	CH1CCIF	UEVTIF				

Reserved	Bit 31-13	-	保留, 必须保持为复位值
CH4OVIF	Bit 12	R	<p><b>捕获/比较 4 捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH4CCIF 标志位置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器</p>
CH3OVIF	Bit 11	R	<p><b>捕获/比较 3 捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH3CCIF 标志位置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器</p>
CH2OVIF	Bit 10	R	<p><b>捕获/比较 2 捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH2CCIF 标志位置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器</p>
CH1OVIF	Bit 9	R	<p><b>捕获/比较 1 捕获溢出中断标志</b></p> <p>仅当相应的通道配置为捕获输入状态时, 该标志位才由硬件设置。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未检测到捕获溢出</p> <p>1: 当 CH1CCIF 标志位置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器</p>
Reserved	Bit 8-7	-	保留, 必须保持为复位值
TRGIF	Bit 6	R	<p><b>触发中断标志</b></p> <p>如果触发中断使能, 当从模式控制器在门控模式</p>

			<p>以外的所有模式下使能，发生触发事件时（In 上检测到有效边沿），该标志位被硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生触发事件 1: 触发中断被挂起</p>
Reserved	Bit 5	-	保留，必须保持为复位值
CH4CCIF	Bit 4	R	<p>捕获/比较 4 中断标志</p> <p>参考 CH1CCIF 描述</p>
CH3CCIF	Bit 3	R	<p>捕获/比较 3 中断标志</p> <p>参考 CH1CCIF 描述</p>
CH2CCIF	Bit 2	R	<p>捕获/比较 2 中断标志</p> <p>参考 CH1CCIF 描述</p>
CH1CCIF	Bit 1	R	<p>捕获/比较 1 中断标志</p> <p>如果 CC1 通道配置为输出： 如果中断使能，除去中央对齐模式的情况（参考 GP16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0: 不匹配。 1: GP16C4Tn_COUNT 计数值与 GP16C4Tn_CCVAL1 值匹配。当 GP16C4Tn_CCVAL1 寄存器值大于 GP16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起</p> <p>如果 CC1 通道配置为输入： 发生捕获时，该位由硬件置起。该位可通过软件或者读取 GP16C4Tn_CCVAL1 寄存器来清零。 0: 未发生输入捕获 1: 计数值捕获至 GP16C4Tn_CCVAL1 寄存器（IC1 上检测到与选中极性匹配的边沿）</p>
UEVTIF	Bit 0	R	<p>更新中断标志</p> <p>如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。</p> <p>0: 未发生更新。 1: 更新中断被挂起。当寄存器更新时，该位被硬件置起： -当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 GP16C4Tn_CON1 寄存器中 DISUE=0 -当使用 GP16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和</p>

			DISUE=0 -当 CNT 由触发事件来重新初始化，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0
--	--	--	--

### 25.5.2.8 中断标志屏蔽寄存器 (GP16C4Tn\_IFM)

中断标志屏蔽寄存器 (GP16C4Tn_IFM)																																											
偏移地址: 001C <sub>H</sub>																																											
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved																		CH4OVIM	CH3OVIM	CH2OVIM	CH1OVIM	Reserved	TRGIM	Reserved	CH4CCIM	CH3CCIM	CH2CCIM	CH1CCIM	UEVTIM														

Reserved	Bit 31-13	-	保留
CH4OVIM	Bit 12	R	屏蔽通道 4 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH4CCIF 标志为置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH3OVIM	Bit 11	R	屏蔽通道 3 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH3CCIF 标志为置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH2OVIM	Bit 10	R	屏蔽通道 2 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH2CCIF 标志为置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
CH1OVIM	Bit 9	R	屏蔽通道 1 捕获/比较捕获溢出中断标志 0: 未检测到捕获溢出 1: 当 CH1CCIF 标志为置起时, 捕获计数器值至 GP16C4Tn_CCVAL1 寄存器
Reserved	Bit 8-7	-	保留
TRGIM	Bit 6	R	屏蔽触发中断标志 如果触发中断使能, 当从模式控制器在门控模式以外的所有模式下使能, 发生触发事件时 (TI 上检测到有效边沿), 该标志位被硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0: 未发生触发事件 1: 触发中断被挂起
Reserved	Bit 5	-	保留
CH4CCIM	Bit 4	R	屏蔽通道 4 捕获/比较中断标志 参考 CH1CCIM 描述
CH3CCIM	Bit 3	R	屏蔽通道 3 捕获/比较中断标志 参考 CH1CCIM 描述
CH2CCIM	Bit 2	R	屏蔽通道 2 捕获/比较中断标志 参考 CH1CCIM 描述
CH1CCIM	Bit 1	R	屏蔽通道 1 捕获/比较中断标志如果通道 1 配置为

			<p><b>输出：</b> 如果中断使能，除去中央对齐模式的情况（参考 GP16C4Tn_CON1 寄存器中 CMSEL 的描述），当计数值与比较值匹配，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：不匹配。 1：GP16C4Tn_COUNT 计数值与 GP16C4Tn_CCVAL1 值匹配。当 GP16C4Tn_CCVAL1 寄存器值大于 GP16C4Tn_AR 值，发生计数器上溢时（递增模式和递增/递减模式）或下溢时（递减模式），CH1CCIF 为被置起。</p> <p><b>如果通道配置为输入：</b> 发生捕获时，该位由硬件置起。该位可通过软件或者读取 GP16C4Tn_CCVAL1 寄存器来清零。 0：未发生输入捕获 1：计数值捕获至 GP16C4Tn_CCVAL1 寄存器(I1 上检测到与选中极性匹配的边沿)</p>
UEVTIM	Bit 0	R	<p><b>屏蔽更新事件中中断标志</b> 如果更新中断使能，当发生更新事件，该标志位由硬件置起。对 GP16C4Tn_ICR 写 1 来清除原始中断。 0：未发生更新。 1：更新中断被挂起。当寄存器更新时，该位被硬件置起： -当重复计数器值发生上溢或者下溢（若重复计数器=0，则更新）和当 GP16C4Tn_CON1 寄存器中 DISUE=0 -当使用 GP16C4Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 -当 CNT 由触发事件来重新初始化，如果 GP16C4Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0</p>

25. 5. 2. 9 中断清零寄存器 (GP16C4Tn\_ICR)

中断清零寄存器 (GP16C4Tn_ICR)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		CH4OVIC	CH3OVIC	CH2OVIC	CH1OVIC	Reserved	TRGIC	Reserved	CH4CCIC	CH3CCIC	CH2CCIC	CH1CCIC	UEVTIC		

Reserved	Bit 31-13	-	保留, 必须保持为复位值
CH4OVIC	Bit 12	R	通道 4 捕获/比较捕获溢出中断标志清除 0: 无效 1: CH4OVIF 清除
CH3OVIC	Bit 11	R	通道 3 捕获/比较捕获溢出中断标志 0: 无效 1: CH3OVIF 清除
CH2OVIC	Bit 10	R	通道 2 捕获/比较捕获溢出中断标志 0: 无效 1: CH2OVIF 清除
CH1OVIC	Bit 9	R	通道 1 捕获/比较捕获溢出中断标志 0: 无效 1: CH1OVIF 清除
Reserved	Bit 8-7	-	保留, 必须保持为复位值
TRGIC	Bit 6	C_W1	触发中断清零 0: 无效 1: 触发中断清零 (GP16C4Tn_RIF)
Reserved	Bit 5	-	保留, 必须保持为复位值
CH4CCIC	Bit 4	C_W1	捕获/比较 4 中断清零 参考 CH1CCIC 描述
CH3CCIC	Bit 3	C_W1	捕获/比较 3 中断清零 参考 CH1CCIC 描述
CH2CCIC	Bit 2	C_W1	捕获/比较 2 中断清零 参考 CH1CCIC 描述
CH1CCIC	Bit 1	C_W1	捕获/比较 1 中断清零 0: 无效 1: 捕获/比较中断清零 (GP16C4Tn_RIF)
UEVTIC	Bit 0	C_W1	更新中断清零 0: 无效 1: 更新中断清零 (GP16C4Tn_RIF)

25.5.2.10 软件生成事件寄存器 (GP16C4Tn\_SGE)

软件生成事件寄存器 (GP16C4Tn_SGE)																															
偏移地址: 024 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SGTRG	Reserved	SGCC4E	SGCC3E	SGCC2E	SGCC1E	SGU	

Reserved	Bit 31-7	-	保留, 必须保持为复位值
SGTRG	Bit 6	W	<p><b>触发生成</b> 该位由软件设置来生成触发事件, 可由硬件自动清零。 0: 无动作 1: GP16C4Tn_RIF 寄存器中的 TRGIF 被置起, 产生相关中断或 DMA 传输</p>
Reserved	Bit 5	-	保留, 必须保持为复位值
SGCC4E	Bit 4	W	<p><b>捕获/比较 4 生成</b> 参考 SGCC1E 描述</p>
SGCC3E	Bit 3	W	<p><b>捕获/比较 3 生成</b> 参考 SGCC1E 描述</p>
SGCC2E	Bit 2	W	<p><b>捕获/比较 2 生成</b> 参考 SGCC1E 描述</p>
SGCC1E	Bit 1	W	<p><b>捕获/比较 1 生成</b> 该位由软件设置来生成事件, 可由硬件自动清零。 0: 无动作 1: 通道 1 上产生捕获/比较事件: <b>如果通道 1 配置为输出:</b> CH1CCIF 标志位被置起, 产生相应中断或 DMA 请求发送 <b>如果通道 1 配置为输入:</b> 当前计数值捕获至 GP16C4Tn_CCVAL1 寄存器。 CH1CCIF 标志位被置起, 产生相应中断或 DMA 请求发送。CH1OVIF 标志位置起如果 CH1CCIF 标志位为高电平。</p>
SGU	Bit 0	W	<p><b>更新生成</b> 该位由软件设置, 可由硬件自动清零。 0: 无动作 1: 重新初始化计数器, 更新寄存器。注意, 预分频器也会被清零 (但预分频比不会受到影响)。如果使用中央对齐模式或者 DIRSEL=0 (递增), 则计数器将清零; 否则如果 DIRSEL=1 (递减), 则将使用自动重载入值。</p>

### 25.5.2.11 捕获/比较模式寄存器 1 (GP16C4Tn\_CHMR1)

◇ 输出比较模式

捕获/比较模式寄存器 1 (GP16C4Tn_CHMR1)																															
偏移地址: 028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH2OCLREN	CH2OMOD			CH2OPREN	CH2OHSEN	CC2SSEL		CH1OCLREN	CH1OMOD			CH1OPREN	CH1OHSEN	CC1SSEL	

Reserved	Bit 31-16	-	保留，必须保持为复位值
CH2OCLREN	Bit 15	R/W	输出比较 2 清零使能 参考 CH1OCLREN 描述
CH2OMOD	Bit 14-12	R/W	输出比较 2 模式 参考 CH1OMOD 描述
CH2OPREN	Bit 11	R/W	输出比较 2 预载使能 参考 CH1OPREN 描述
CH2OHSEN	Bit 10	R/W	输出比较 2 高速使能 参考 CH1OHSEN 描述
CC2SSEL	Bit 9-8	R/W	输出比较 2 选择 该位定义了通道以及使用的输入的方向（输入/输出） 00：通道配置为输出 01：通道配置为输入，捕获源为 I2 10：通道配置为输入，捕获源为 I1 11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。 仅当内部触发输入通过 TSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才能工作。 注意：当通道为关闭状态时（GP16C4Tn_CCEP 中 CC2EN = '0'），CC2SSEL 为只写。
CH1OCLREN	Bit 7	R/W	输出比较 1 清零使能 0：通道 1 比较输出不会受到 ETFP 输入影响 1：当 ETFP 输入上检测到高电平时，通道 1 比较输出将被清零
CH1OMOD	Bit 6-4	R/W	输出比较 1 模式 该位定义了输出参考信号通道 1 比较输出的行为。 通道 1 比较输出为高有效，CH1O 的有效电平由 CC1POL 位决定。 000：冻结—输出比较寄存器 GP16C4Tn_CCVAL1



		<p>寄存器和 GP16C4Tn_COUNT 计数器之间的比较对输出无效。</p> <p>001：发生匹配时设置通道 1 为有效电平-当计数器 GP16C4Tn_COUNT 与捕获/比较寄存器 1GP16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为高电平。</p> <p>010：发生匹配时设置通道 1 为无效电平-当计数器 GP16C4Tn_COUNT 与捕获/比较寄存器 1GP16C4Tn_CCVAL1 发生匹配后，通道 1 比较输出信号强制为低电平。</p> <p>011：翻转 -当 GP16C4Tn_COUNT=GP16C4Tn_CCVAL1，通道 1 比较输出发生翻转。</p> <p>100：强制为无效电平 - 通道 1 比较输出强制为低电平。</p> <p>101：强制为有效电平- 通道 1 比较输出强制为高电平。</p> <p>110：PWM 模式 1 -在递增模式下，当 GP16C4Tn_COUNT&lt;GP16C4Tn_CCVAL1，通道 1 为有效电平，否则，通道 1 为无效电平。在递减模式下，当 GP16C4Tn_COUNT&gt;GP16C4Tn_CCVAL1，通道 1 为无效电平（通道 1 比较输出='0'），否则通道 1 为有效电平（通道 1 比较输出='1'）。</p> <p>111：PWM 模式 2 -在递增模式下，当 GP16C4Tn_COUNT&lt;GP16C4Tn_CCVAL1，通道 1 为无效电平，否则，通道 1 为有效电平。在递减模式下，当 GP16C4Tn_COUNT&gt;GP16C4Tn_CCVAL1，通道 1 为有效电平，否则通道 1 为无效电平。</p> <p>注意： 在 PWM 模式 1 和 2 中，仅当比较结果更改或当输出比较模式从冻结模式转换成 PWM 模式，比较输出电平才会更改。</p>
CH1OPREN	Bit 3	<p><b>输出比较 1 预载使能</b></p> <p>0：GP16C4Tn_CCVAL1 的预载寄存器禁止。GP16C4Tn_CCVAL1 在任何时候都可写，新写入的值将立刻生效。</p> <p>1：GP16C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件，GP16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。</p> <p>注意： 仅在单脉冲模式下（GP16C4Tn_CON1 寄存器中</p>

			的 SPMEN 设置为 1), PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。
CH1OHSEN	Bit 2	RW	<p><b>输出比较 1 高速使能</b> 该位用来加速在 CC 输出上的输入触发事件的效应。</p> <p>0: 当触发开启, 通道 1 运作正常取决于计数器和 CCRV1 的值。当触发输入上发现边沿时, 至少需要 5 个时钟周期来激活通道 1 输出。</p> <p>1: 触发输入上的有效沿类似于通道 1 输出上的比较匹配。设置 OC 为 1 用来比较电平, 采样触发输入和激活通道 1 输出的延时将会减少至 3 个时钟周期。只有当通道配置为 PWM1 或 PWM2 模式, CH1OHSEN 才会起作用。</p>
CC1SSEL	Bit 1-0	RW	<p><b>捕获/比较 1 选择</b> 该位定义了通道和使用的输入的方向。</p> <p>00: 通道配置为<b>输出</b></p> <p>01: 通道配置为输入, 捕获源为 I1</p> <p>10: 通道配置为输入, 捕获源为 I2</p> <p>11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出。</p> <p>只有当内部触发输入是通过 TSSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时, 该模式才运行。</p> <p>注意: 当通道关闭 (GP16C4Tn_CCEP 寄存器中的 CC1EN = '0'), CC1SSEL 为只写。</p>

◇ 输入捕获模式

捕获/比较模式寄存器 1 (GP16C4Tn_CHMR1)																															
偏移地址: 028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2FLT			IC2PRES		CC2SSEL		I1FLT			IC1PRES		CC1SSEL			

Reserved	Bit 31-16	-	保留，必须保持为复位值
I2FLT	Bit 15-12	R/W	<b>输入捕获 2 滤波器</b> 参考 I1FLT 描述
IC2PRES	Bit 11-10	R/W	<b>输入捕获 2 预分频器</b> 参考 IC1PRES 描述
CC2SSEL	Bit 9-8	R/W	<b>输入捕获 2 选择</b> 该位定义了通道和使用的输入的方向。 00: 通道配置为 <b>输出</b> 01: 通道配置为输入，捕获源为 <b>I2</b> 10: 通道配置为输入，捕获源为 <b>I1</b> 11: 通道配置为输入，捕获源为 <b>ITn 或 I1 的双边沿检出</b> 只有当内部触发输入是通过 TSSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时，该模式才运行 注意：当通道关闭 (GP16C4Tn_CCEP 寄存器中的 CC2EN = '0')，CC2SSEL 为只写。
I1FLT	Bit 7-4	R/W	<b>输入捕获 1 滤波器</b> 该位定义了 I1 输入的采样频率和数字滤波器的长度。 数字滤波器由一个事件计数器组成，每 N 个连续事件才视为一个有效边沿： 0000: 无滤波器，采样频率为 $f_{DTS}$ 0001: $f_{SAMPLING} = f_{INT\_CLK}, N = 2$ 0010: $f_{SAMPLING} = f_{INT\_CLK}, N = 4$ 0011: $f_{SAMPLING} = f_{INT\_CLK}, N = 8$ 0100: $f_{SAMPLING} = f_{DTS} / 2, N = 6$ 0101: $f_{SAMPLING} = f_{DTS} / 2, N = 8$ 0110: $f_{SAMPLING} = f_{DTS} / 4, N = 6$ 0111: $f_{SAMPLING} = f_{DTS} / 4, N = 8$ 1000: $f_{SAMPLING} = f_{DTS} / 8, N = 6$ 1001: $f_{SAMPLING} = f_{DTS} / 8, N = 8$ 1010: $f_{SAMPLING} = f_{DTS} / 16, N = 5$ 1011: $f_{SAMPLING} = f_{DTS} / 16, N = 6$

			<p>1100: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 16, N = 8</math>          1101: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 5</math>          1110: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 6</math>          1111: <math>f_{\text{SAMPLING}} = f_{\text{DTS}} / 32, N = 8</math>          注意: 当 ICxF [3: 0] = 1, 2 or 3 时, 公式中的 <math>f_{\text{DTS}}</math> 由 INT_CLK 取代。</p>
IC1PRES	Bit 3-2	RW	<p><b>输入捕获 1 预分频器</b>          该位定义了作用在 CC1 输入 (I1) 上的预分频比。当 CC1EN='0' (GP16C4Tn_CCEP 寄存器), 预分频器将复位。          00: 无预分频器。每当捕获输入上检测到边沿时, 发生捕获动作。          01: 每发生 2 次事件, 执行一次捕获          10: 每发生 4 次事件, 执行一次捕获          11: 每发生 8 次事件, 执行一次捕获</p>
CC1SSEL	Bit 1-0	RW	<p><b>输入捕获 1 选择</b>          该位定义了通道和使用的输入的方向          00: CC1 通道配置为<b>输出</b>          01: CC1 通道配置为输入, IC1 映射到 <b>I1</b>          10: CC1 通道配置为输入, IC1 映射到 <b>I2</b>          11: CC1 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出。只有当内部触发输入是通过 TSSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时, 该模式才运行          注意: 当通道关闭 (GP16C4Tn_CCEP 寄存器中的 CC1EN = '0'), CC1SSEL 为只写。</p>

### 25.5.2.12 捕获/比较模式寄存器 2 (GP16C4Tn\_CHMR2)

◇ 输出比较模式

捕获/比较模式寄存器 2 (GP16C4Tn_CHMR2)																															
偏移地址: 02C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CH4OCLREN	CH4OMOD			CH4OPREN	CH4OHSEN	CC4SSEL		CH3OCLREN	CH3OMOD			CH3OPREN	CH3OHSEN	CC3SSEL	

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CH4OCLREN	Bit 15	R/W	输出比较 4 清零使能 参考 CH1OCLREN 描述
CH4OMOD	Bit 14-12	R/W	输出比较 4 模式 参考 CH1OMOD 描述
CH4OPREN	Bit 11	R/W	输出比较 4 预载使能 参考 CH1OPREN 描述
CH4OHSEN	Bit 10	R/W	输出比较 4 高速使能 参考 CH1OHSEN 描述
CC4SSEL	Bit 9-8	R/W	输出比较 4 选择 该位定义了通道以及使用的输入的方向 (输入/输出)。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 I4 10: 通道配置为输入, 捕获源为 I3 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 仅当内部触发输入通过 TSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时, 该模式才能工作。 注意: 当通道为关闭状态时 (GP16C4Tn_CCEP 中 CC4EN = '0'), CC4SSEL 为只写。
CH3OCLREN	Bit 7	R/W	输出比较 3 清零使能 参考 CH1OCLREN 描述
CH3OMOD	Bit 6-4	R/W	输出比较 3 模式 参考 CH1OMOD 描述
CH3OPREN	Bit 3	R/W	输出比较 3 预载使能 参考 CH1OPREN 描述
CH3OHSEN	Bit 2	R/W	输出比较 3 高速使能 参考 CH1OHSEN 描述
CC3SSEL	Bit 1-0	R/W	捕获/比较 3 选择

		<p>该位定义了通道和使用的输入的方向。</p> <p>00：通道配置为<b>输出</b></p> <p>01：通道配置为输入，捕获源为 <b>I3</b></p> <p>10：通道配置为输入，捕获源为 <b>I4</b></p> <p>11：通道配置为输入，捕获源为 ITn 或 I1 的双边沿检出。</p> <p>只有当内部触发输入是通过 TSSEL 位（GP16C4Tn_SMCON 寄存器）选择时，该模式才运行。</p> <p>注意：当通道关闭（GP16C4Tn_CCEP 寄存器中的 CC3EN = '0'），CC3SSEL 为只写</p>
--	--	---

◇ 输入捕获模式

捕获/比较模式寄存器 2 (GP16C4Tn_CHMR2)																															
偏移地址: 02C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I4FLT			IC4PRES		CC4SSEL		IC3FLT			IC3PRES		CC3SSEL			

Reserved	Bit 31-16	-	保留，必须保持为复位值
I4FLT	Bit 15-12	R/W	输入捕获 4 滤波器 参考 I1FLT 描述
IC4PRES	Bit 11-10	R/W	输入捕获 4 预分频器 参考 IC1PRES 描述
CC4SSEL	Bit 9-8	R/W	输入捕获 4 选择 该位定义了通道和使用的输入的方向。 00: CC4 通道配置为输出 01: CC4 通道配置为输入, IC4 映射到 <b>TI4</b> 10: CC4 通道配置为输入, IC4 映射到 <b>TI3</b> 11: CC4 通道配置为输入, IC4 映射到 <b>TRC</b> 只有当内部触发输入是通过 TSSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时, 该模式才运行 注意: 当通道关闭 (GP16C4Tn_CCEP 寄存器中的 CC4EN = '0'), CC4SSEL 为只写。
IC3FLT	Bit 7-4	R/W	输入捕获 3 滤波器 参考 I1FLT 描述
IC3PRES	Bit 3-2	R/W	输入捕获 3 预分频器 参考 IC1PRES 描述
CC3SSEL	Bit 1-0	R/W	输入捕获 3 选择 该位定义了通道和使用的输入的方向。 00: 通道配置为输出 01: 通道配置为输入, 捕获源为 <b>I3</b> 10: 通道配置为输入, 捕获源为 <b>I4</b> 11: 通道配置为输入, 捕获源为 ITn 或 I1 的双边沿检出 只有当内部触发输入是通过 TSSEL 位 (GP16C4Tn_SMCON 寄存器) 选择时, 该模式才运行 注意: 当通道关闭 (GP16C4Tn_CCEP 寄存器中的 CC3EN = '0'), CC3SSEL 为只写。

25. 5. 2. 13 捕获/比较使能寄存器 (GP16C4Tn\_CCEP)

捕获/比较使能寄存器 (GP16C4Tn_CCEP)																															
偏移地址: 030 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CC4POL	CC4EN	Reserved			CC3POL	CC3EN	Reserved			CC2POL	CC2EN	Reserved			CC1POL	CC1EN		

Reserved	Bit 31-14	-	保留, 必须保持为复位值
CC4POL	Bit 13	R/W	捕获/比较 4 输出极性 参考 CC1POL 描述
CC4EN	Bit 12	R/W	捕获/比较 4 输出使能 参考 CC1EN 描述
Reserved	Bit 11-10	-	保留, 必须保持为复位值
CC3POL	Bit 9	R/W	捕获/比较 3 输出极性 参考 CC1POL 描述
CC3EN	Bit 8	R/W	捕获/比较 3 输出使能 参考 CC1EN 描述
Reserved	Bit 7-6	-	保留, 必须保持为复位值
CC2POL	Bit 5	R/W	捕获/比较 2 输出极性 参考 CC1POL 描述
CC2EN	Bit 4	R/W	捕获/比较 2 输出使能 参考 CC1EN 描述
Reserved	Bit 3-2	-	保留, 必须保持为复位值
CC1POL	Bit 1	R/W	捕获/比较 1 输出极性 通道配置为输出: 0: CH10 高有效 1: CH10 低有效 通道配置为输入: CC1POL 为触发和捕获操作选择 I1 边沿检出和 I2 边沿检出的有效极性。 0: 正向/上升沿 电路对 In 边沿检出的上升沿敏感 (在复位, 外部时钟或触发模式下, 进行捕获或触发), In 边沿检出不反向 (门控模式或编码器模式下, 进行触发)。 1: 反向/下降沿 电路对 In 边沿检出的下降沿敏感 (在复位, 外部时钟或触发模式下, 进行捕获或触发), In 边沿检出反向 (门控模式或编码器模式下, 进行触发)。
CC1EN	Bit 0	R/W	捕获/比较 1 输出使能 通道配置为输出:



			<p>0: 关闭 - CH1O 无效。 1: 开启 - CH1O 为对应输出引脚上的输出信号, 由 CC1EN 决定</p> <p><b>通道配置为输入:</b> 该位决定了计数值是否能捕获到输入捕获/比较寄存器 1 (GP16C4Tn_CCVAL1)。</p> <p>0: 禁止捕获。 1: 使能捕获。</p>
--	--	--	--

### 25. 5. 2. 14 计数器寄存器 (GP16C4Tn\_COUNT)

计数器寄存器 (GP16C4Tn_COUNT)																															
偏移地址: 034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CNTV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CNTV	Bit 15-0	R/W	计数值

### 25.5.2.15 预分频寄存器 (GP16C4Tn\_PRES)

预分频寄存器 (GP16C4Tn_PRES)																															
偏移地址: 038 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
PSCV	Bit 15-0	RW	<p><b>预分频器值</b></p> <p>计数器时钟频率 (CK_CNT) = <math>f_{CK\_PSC} / (PSCV [15:0] + 1)</math></p> <p>每发生一次更新事件 (包括当计数器由 GP16C4Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时, 通过触发控制器清零), PSCV 包含的值将被填入到有效的预分频寄存器内。</p>

### 25.5.2.16 自动重载寄存器 (GP16C4Tn\_AR)

自动重载寄存器 (GP16C4Tn_AR)																															
偏移地址: 03C <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
ARRV	Bit 15-0	RW	<p><b>自动重载值</b></p> <p>ARRV 中的值将被载入实际的自动重载寄存器中。当自动重载值为空, 计数器被屏蔽。</p>

### 25.5.2.17 捕获/比较寄存器 1 (GP16C4Tn\_CCVAL1)

捕获/比较寄存器 1 (GP16C4Tn_CCVAL1)																															
偏移地址: 044 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV1															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CCRV1	Bit 15-0	R/W	<p><b>捕获/比较值 1</b></p> <p><b>如果通道 CCn 配置为输出:</b> CCRVn 中的值将被载入实际的捕获/比较寄存器中 (预载值)。 如果在 GP16C4Tn_CHMRn 寄存器中的预载功能没有选中, CCRVn 中的值将被永久载入; 否则, 每当发生更新事件, 预载值将会复制到有效的捕获/比较寄存器中。有效捕获/比较寄存器中包含的值将会与 GP16C4Tn_COUNT 中的值进行比较, 并在 OCn 上输出。</p> <p><b>如果通道 CCn 配置为输入:</b> CCRVn 为由上一个输入捕获事件 (ICn) 传输的计数值。</p>

### 25.5.2.18 捕获/比较寄存器 2 (GP16C4Tn\_CCVAL2)

捕获/比较寄存器 2 (GP16C4Tn_CCVAL2)																															
偏移地址: 048 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCRV2															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CCRV2	Bit 15-0	R/W	<p><b>捕获/比较值 2</b></p> <p>参考 CCRV1 描述</p>

### 25.5.2.19 捕获/比较寄存器 3 (GP16C4Tn\_CCVAL3)

捕获/比较寄存器 3 (GP16C4Tn_CCVAL3)																															
偏移地址: 04C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCR3															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CCR3	Bit 15-0	R/W	捕获/比较值 3 参考 CCRV1 描述

### 25.5.2.20 捕获/比较寄存器 4 (GP16C4Tn\_CCVAL4)

捕获/比较寄存器 4 (GP16C4Tn_CCVAL4)																															
偏移地址: 050 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CCR4															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CCR4	Bit 15-0	R/W	捕获/比较值 4 参考 CCRV1 描述

25. 5. 2. 21 DMA使能寄存器 (GP16C4Tn\_DMAEN)

DMA 使能寄存器 (GP16C4Tn_DMAEN)																															
偏移地址: 058 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TRGDMA	COMDMA	CC4DMA	CC3DMA	CC2DMA	CC1DMA	UDMA	

Reserved	Bit 31-7	-	保留, 必须保持为复位值
TRGDMA	Bit 6	R/W	触发DMA请求使能 0: DMA请求禁止 1: DMA请求使能
COMDMA	Bit 5	R/W	COM DMA访问使能 0: DMA请求禁止 1: DMA请求使能
CC4DMA	Bit 4	R/W	捕获/比较值 4 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC3DMA	Bit 3	R/W	捕获/比较值 3 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC2DMA	Bit 2	R/W	捕获/比较值 2 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
CC1DMA	Bit 1	R/W	捕获/比较值 1 DMA 访问使能 0: DMA 请求禁止 1: DMA 请求使能
UDMA	Bit 0	R/W	更新 DMA 请求使能 0: DMA 请求禁止 1: DMA 请求使能

## 第26章 基本定时器(BS16T)

### 26.1 概述

基本定时器（BS16T）包含一个 16 位自动重载计数器，该计数器由可配置的预分频器驱动。

通过使用定时器的分频器和 APB 时钟控制器的预分频功能，可对脉冲长度和波形周期进行数微妙到几毫秒的调整。

### 26.2 特性

- ◇ 16 位自动加载递增计数器
- ◇ 16 位可编程预分频器，可对计数器工作时钟进行 1 到 65536 的任意分频(运行中也可以)
- ◇ 计数上溢更新事件产生中断/DMA 请求

### 26.3 结构框图

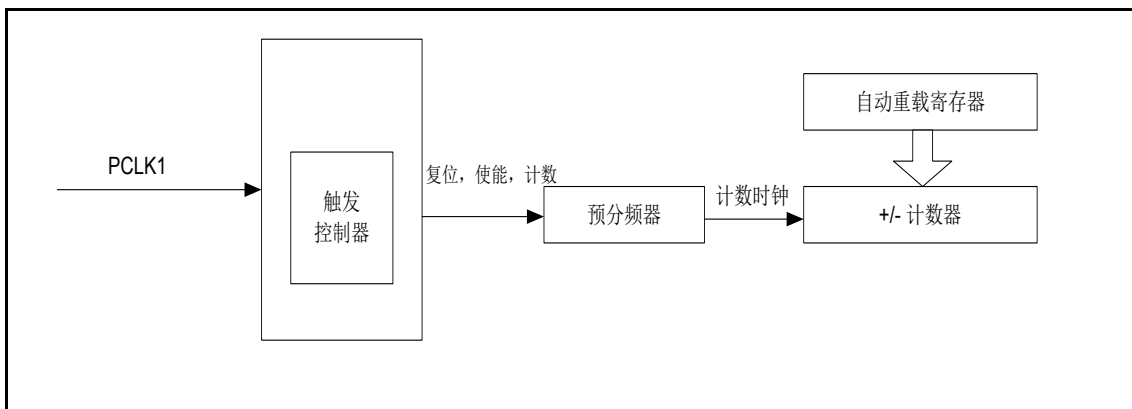


图 26-1 基本定时器电路结构框图

### 26.4 功能描述

#### 26.4.1 预分频器

定时器包含一个 16-bit 的计数器（BS16Tn\_COUNT），计数时钟由预分频寄存器（BS16Tn\_PRES）进行分频。计数周期由自动重载计数器（BS16Tn\_AR）设定。

自动重载寄存器（BS16Tn\_AR）是一个可缓存的寄存器。当 BS16Tn\_CON1 寄存器的 ARPEN 位复位时，BS16Tn\_AR 寄存器重载功能失效，BS16Tn\_AR 就是有效寄存器；ARPEN 置位时，BS16Tn\_AR 寄存器具有重载功能，产生更新事件（UEV）时，加载值（BS16Tn\_AR 寄存器值）更新到影子寄存器。

当 BS16Tn\_CON1 寄存器中 DISUE 位为 0 时，计数器计数上溢（或递减下溢）时会产生更新事件（UEV）。同样，软件方式也可产生更新事件。BS16Tn\_CON1 寄存器的 CNTEN 置位时，计数器开始计数。

注：计数器在 CNTEN 位置位 1 个时钟周期后开始计数。

预分频器可对定时器工作时钟进行 BS16Tn\_PRES 寄存器值+1 次分频。由于 BS16Tn\_PRES 是一个可重载寄存器，因此，定时器工作时可以对该寄存器进行修改，修改值在下次更新事件（UEV）后有效。

下图给出了定时器运行过程中改变预分频值时计数器的计数情况。

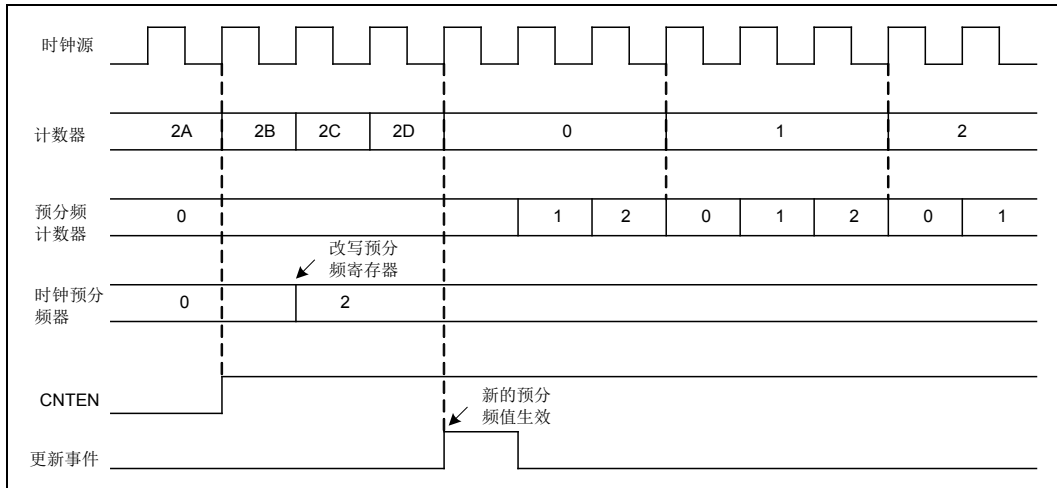


图 26-2 预分频值计数时序图

### 26.4.2 时钟源

计数时钟由内部时钟源（PCLK1）提供。

CNTEN 位（BS16Tn\_CON1 寄存器）与 SGU 位（BS16Tn\_SGE 寄存器）为实际控制位，这两个位只能软件修改（SGU 位除外，仍硬件自动清除）。一旦 CNTEN 位被写为'1'，预分频器就由内部 PCLK1 提供时钟。

### 26.4.3 递增计数模式

在递增计数模式中，计数器由 0 开始计数至自动重载值（BS16Tn\_AR 寄存器中的值），然后从 0 开始重新计数并产生一个计数溢出事件。

软件置位 BS16Tn\_CON1 寄存器中的 DISUE 位可关闭更新事件（UEV）的产生。更新事件（UEV）关闭，可避免向预载寄存器写新值的过程中更新影子寄存器。这种情况下，DISUE 位在写'0'之前都不会产生更新事件。正常产生更新事件后，计数器和预载计数器都是从 0 重新开始（但预分频值没有改变）。此外，若置位 BS16Tn\_CON1 寄存器中的 UERSEL 位（更新请求选择），置位 SGU 位时会产生一次更新事件（UEV），但 UEVTIF 标志位不会置位（因此，不会触发中断或 DMA 请求）。

当更新事件发生时，所有寄存器都会被更新且更新标志位（BS16Tn\_RIF 寄存器中的 UIF 位）置位（取决于 UERSEL 位）：

- ◇ 自动重载影子寄存器加载 BS16Tn\_AR 寄存器中的值
- ◇ 预分频器之缓冲器加载 BS16Tn\_PRES 寄存器中的值

### 26.4.4 调试模式

当微控制器进入调试模式（Cortex™-M 内核终止），计数器可被设定停止计数。

## 26.5 特殊功能寄存器

### 26.5.1 寄存器列表

TIM4_5 寄存器列表		
寄存器名称	偏移地址	寄存器描述
TIM4 基地址: 4000_1000 <sub>H</sub> TIM5 基地址: 4000_1400 <sub>H</sub>		
BS16Tn_CON1	000 <sub>H</sub>	控制寄存器 1
Reserved	004 <sub>H</sub>	保留
BS16Tn_IER	00C <sub>H</sub>	中断使能寄存器
BS16Tn_IDR	010 <sub>H</sub>	中断禁止寄存器
BS16Tn_IVS	014 <sub>H</sub>	中断有效状态寄存器
BS16Tn_RIF	018 <sub>H</sub>	原始中断标志寄存器
BS16Tn_IFM	01C <sub>H</sub>	中断标志屏蔽寄存器
BS16Tn_ICR	020 <sub>H</sub>	中断清零寄存器
BS16Tn_SGE	024 <sub>H</sub>	事件生成寄存器
BS16Tn_COUNT	034 <sub>H</sub>	计数器
BS16Tn_PRES	038 <sub>H</sub>	预分频器
BS16Tn_AR	03C <sub>H</sub>	自动重载寄存器
BS16Tn_DMAEN	058 <sub>H</sub>	DMA 使能寄存器



## 26.5.2 寄存器描述

### 26.5.2.1 控制寄存器 1 (BS16Tn\_CON1)

控制寄存器 1 (BS16Tn_CON1)																															
偏移地址: 000 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								APREN	Reserved			SPMEN	UERSEL	DISUE	CNTEN

Reserved	Bit 31-8	-	保留, 必须保持为复位值
APREN	Bit7	R/W	<b>自动重载预载使能</b> 0: BS16Tn_AR 寄存器未缓冲 1: BS16Tn_AR 寄存器被装入缓冲器
Reserved	Bit 6-4	-	保留, 必须保持为复位值
SPMEN	Bit 3	R/W	<b>单脉冲模式</b> 0: 当发生更新事件时, 计数器不停止。 1: 当发生下一次更新事件 (CNTEN 位清零) 时, 计数器停止。
UERSEL	Bit 2	R/W	<b>更新请求源</b> 该位由软件置 1 或清零, 来选择 UEV 事件源。 0: 如果更新中断或 DMA 请求使能, 则下述任一事件都可产生更新中断或 DMA 请求: -计数器上溢 -设置 SGU 位 1: 如果更新中断或 DMA 请求使能, 仅计数器上溢才能产生更新中断或 DMA 请求中断
DISUE	Bit1	R/W	<b>更新禁止</b> 该位由软件置 1 或清零来使能/禁止 UEV 事件的产生。 0: UEV 使能。更新事件 (UEV) 由下列任一事件产生: - 计数器上溢 -设置 SGU 位 缓冲寄存器载入他们的预载值。 1: UEV 禁止。不产生更新事件, 影子寄存器保持他们的值 (ARRV, PSCV)。如果从从模式控制器接收到硬件复位, 计数器和预分频器将被重新初始化。
CNTEN	Bit0	R/W	<b>计数器使能</b> 0: 计数器禁止 1: 计数器使能

### 26.5.2.2 中断使能寄存器 (BS16Tn\_IER)

中断使能寄存器 (BS16Tn_IER)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UIT

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UIT	Bit 0	W	更新中断使能 0: 无效 1: 使能

### 26.5.2.3 中断禁止寄存器 (BS16Tn\_IDR)

中断禁止寄存器 (BS16Tn_IDR)																															
偏移地址: 010 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UI

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UI	Bit 0	W	更新中断禁止 0: 无效 1: 禁止

### 26.5.2.4 中断有效状态寄存器 (BS16Tn\_IVS)

中断有效状态寄存器 (BS16Tn_IVS)																																
偏移地址: 014 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																											UEI					

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UEI	Bit 0	R	更新中断有效状态 0: 禁止更新中断 1: 使能更新中断 IER/IDR 写 1 来使能或禁止该位。

### 26.5.2.5 原始中断标志寄存器 (BS16Tn\_RIF)

原始中断标志寄存器 (BS16Tn_RIF)																																
偏移地址: 018 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																											UEVTIF					

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UEVTIF	Bit 0	R	更新中断标志 如果更新中断使能, 当发生更新事件, 该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断 0: 未发生更新 1: 更新中断被挂起。当寄存器更新时, 该位被硬件置起: -当计数器值发生上溢 (若计数器=0, 则更新) 和当 BS16Tn_CON1 寄存器中 DISUE=0 -当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时, 如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0 -当 CNT 由触发事件来重新初始化, 如果 BS16Tn_CON1 寄存中的 UERSEL=0 和 DISUE=0。

### 26.5.2.6 中断标志屏蔽寄存器 (BS16Tn\_IFM)

中断标志屏蔽寄存器 (BS16Tn_IFM)																																
偏移地址: 01C <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																															UEI	

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UEI	Bit 0	R	<p><b>更新中断标志屏蔽</b></p> <p>如果更新中断使能, 当发生更新事件, 该标志位由硬件置起。对 BS16Tn_ICR 写 1 来清除原始中断</p> <p>0: 未发生更新</p> <p>1: 更新中断被挂起。当寄存器更新时, 该位被硬件置起:</p> <ul style="list-style-type: none"> <li>-当计数器值发生上溢(若计数器=0, 则更新)和当 BS16Tn_CON1 寄存器中 DISUE=0</li> <li>-当使用 BS16Tn_SGE 寄存器中的 SGU 位来由软件重新初始化 CNT 时, 如果 BS16Tn_CON1 寄存器中的 UERSEL=0 和 DISUE=0</li> <li>-当 CNT 由触发事件来重新初始化, 如果 BS16Tn_CON1 寄存器中的 UERSEL=0 和 DISUE=0。</li> </ul>

### 26.5.2.7 中断清零寄存器 (BS16Tn\_ICR)

中断清零寄存器 (BS16Tn_ICR)																																
偏移地址: 020 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																															UEIC	

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UEIC	Bit 0	C_W1	<p><b>更新中断清零</b></p> <p>0: 无效</p> <p>1: 更新中断清零 (BS16Tn_RIF)</p>

### 26.5.2.8 软件生成事件寄存器 (BS16Tn\_SGE)

软件生成事件寄存器 (BS16Tn_SGE)																															
偏移地址: 024 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															SGU

Reserved	Bit 31-1	-	保留, 必须保持为复位值
SGU	Bit 0	W	<b>更新生成</b> 该位由软件设置, 可由硬件自动清零. 0: 无动作 1: 重新初始化计数器, 更新寄存器。注意, 预分频器也会被清零 (但预分频比不会受到影响)。

### 26.5.2.9 计数器寄存器 (BS16Tn\_COUNT)

计数器寄存器 (BS16Tn_COUNT)																															
偏移地址: 034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CNTV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
CNTV	Bit 15-0	R/W	计数值

### 26.5.2.10 预分频寄存器 (BS16Tn\_PRES)

预分频寄存器 (BS16Tn_PRES)																															
偏移地址: 038 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PSCV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
PSCV	Bit 15-0	RW	<p><b>预分频器值</b></p> <p>计数器时钟频率 (CK_CNT) = fCK_PSC / (PSCV[15:0] + 1)</p> <p>每发生一次更新事件 (包括当计数器由 BS16Tn_SGE 寄存器中的 SGU 位清零或当配置为复位模式时, 通过触发控制器清零), PSCV 包含的值将被填入到有效的预分频寄存器内。</p>

### 26.5.2.11 自动重载寄存器 (BS16Tn\_AR)

自动重载寄存器 (BS16Tn_AR)																															
偏移地址: 03C <sub>H</sub>																															
复位值: 00000000_00000000_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARRV															

Reserved	Bit 31-16	-	保留, 必须保持为复位值
ARRV	Bit 15-0	RW	<p><b>自动重载值</b></p> <p>AR 中的值将被载入实际的自动重载寄存器中。当自动重载值为空, 计数器被屏蔽。</p>

### 26. 5. 2. 12 DMA使能寄存器 (BS16Tn\_DMAEN)

DMA 使能寄存器 (BS16Tn_DMAEN)																															
偏移地址: 058 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															UDEN

Reserved	Bit 31-1	-	保留, 必须保持为复位值
UDEN	Bit 0	R/W	<b>DMA 访问使能</b> 0: DMA 访问禁止 1: DMA 访问使能

## 第27章 低功耗定时器(LP16T)

### 27.1 概述

LP16T 是为低功耗应用而设计的 16 位计数器。得益于多种时钟源选择，LP16T 能够在多种低功耗模式下（SLEE,STOPx）运行，STANDBY 除外。LP16T 支持使用外部输入作为时钟源，可以在内部时钟停止的情况下计数。另外，LP16T 可以将系统从 SLEE,STOPx 模式唤醒。

### 27.2 特性

- ◇ 16 位向上计数
- ◇ 3 位预分频器支持 8 种分频系数 (1,2,4,8,16,32,64,128)
- ◇ 时钟源
  - ◇ 内部时钟源: PCLK2,HRC,LRC,LOSC,HOSC,ULRC,LOSCM
  - ◇ 外部时钟源: 外部端口输入
- ◇ 16 位 ARR 自动加载寄存器
- ◇ 16 位比较寄存器
- ◇ 连续或单发模式可选
- ◇ 软件或硬件触发可选
- ◇ 可编程滤波器
- ◇ 可配置输出: 脉冲, PWM, 翻转
- ◇ 输出极性可配



## 27.3 结构框图

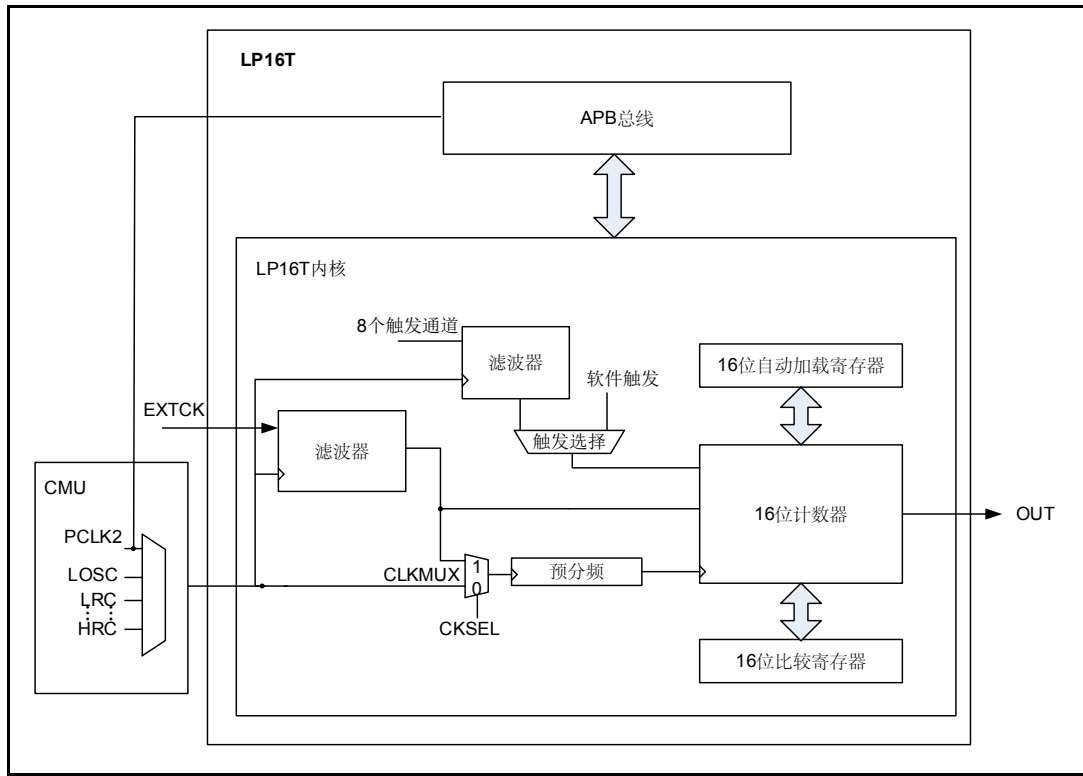


图 27-1 低功耗定时器电路结构框图

LP16T 的计数时钟源可选择 PCLK2,HRC,LRC,LOSC,HOSC,ULRC 等,具体可参考 CMU(时钟管理单元)寄存器说明。另外 LP16T 可选择外部端口输入 (input1) 作为计数时钟,在低功耗模式下内部时钟停止时作脉冲计数器使用。

通过设置 CKSEL 可选择使用内部时钟或外部时钟作为计数时钟。当使用外部时钟时,CKPOL 可选择时钟的有效极性。

## 27.4 功能描述

### 27.4.1 数字滤波器

LP16T 的内部或外部输入,可通过数字滤波器以阻止任何毛刺或噪声传导至 LP16T 内部。这样可以防止错误的计数或触发。

在使能数字滤波器前,需先选择 LP16T 内部时钟源,以保证数字滤波器的正常工作。

数字滤波器分为两组:

一组用来保护 LP16T 的外部端口输入。滤波灵敏度由 CKFLT 控制。

一组用来保护 LP16T 的内部触发输入。滤波灵敏度由 TRGFLT 控制。

注意: 数字滤波器的滤波灵敏度控制时按组分别控制的,但不对每一个滤波器进行单独控制。

滤波器的灵敏度是对所选择通道信号进行连续采样，由采样值的次数来确定信号的变化是否为合法。

下图表示 2 次连续采样时的滤波器行为

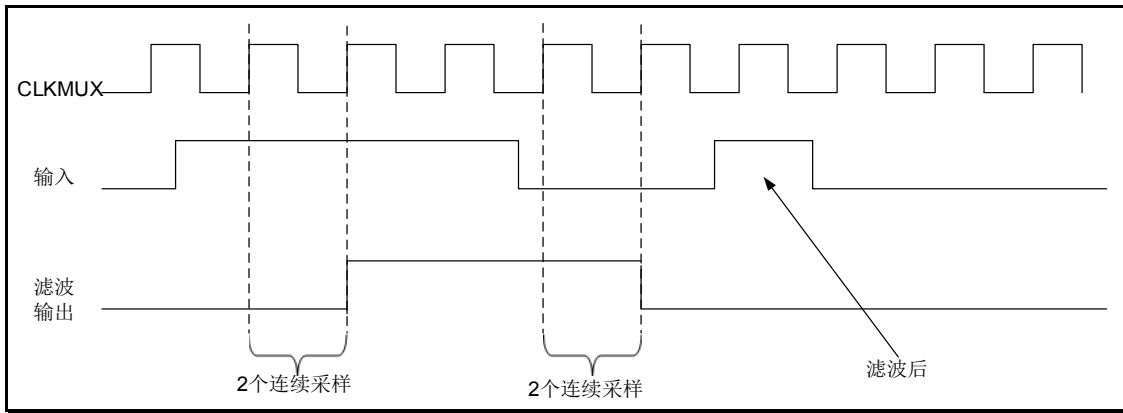


图 27-2 数值滤波器

### 27.4.2 预分频器

LP16T 的 16 位计数器前置一个可配置 2 的 N 次方分频的预分频器。预分频器的分频系数由 PRESC[2:0]控制。下表列出了所有可设置系数

编程	分频比
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

表 27-1 预分频器分频系数

### 27.4.3 触发源选择

LP16T 计数器可由软件启动，也可选择 8 个触发通道中的一个作为触发源，在检测到该通道信号的有效边缘之后，启动计数器。

TRIGEN[1: 0]用来配置选择 LP16T 的触发源。

当 TRIGEN=00 时，软件置位 CNTSTRT 或者 SNGSTRT 可马上启动 LP16T 计数器。

余下各种情况分别配置触发信号的有效边缘。LP16T 计数器将在检测到有效触发边缘后开始计数。

当 TRIGEN 不为 00 时，可设置 TRIGSEL[2:0]来选择触发通道作为 LP16T 的启动信号。

外部触发信号对于 LP16T 来说是异步的。所以在检测到触发后，至少需要 2 个计数时钟的延迟时间来保证同步。待同步完成，计数器被启动。

如果计数器已经启动，此时再次检测到新的触发事件，该事件将被忽略。

*注意：在置位 SNGSTRT/CNTSTRT 之前，必须使能计数器。在计数器禁止期间，对 SNGSTRT/CNTSTRT 操作时无效的。*

### 27.4.4 计数模式

LP16T 有两种工作模式：

连续模式：计数器自由计数，计数器由触发事件启动后便一直计数，除非设定关闭计数器。

单发模式：计数器由触发事件启动计数后，计数到 ARR 值后停止。检测到新的触发事件会重启计数器。在计数器启动之后，直到计数值达到 ARR 这段期间产生的触发事件都被忽略。

若要进行单发计数，应先置位 SNGSTRT。

如果选择了外部触发，每一个在 SNGSTRT 置位之后和计数器停止之后到达的触发事件，将启动新的单发计数。如下图所示：

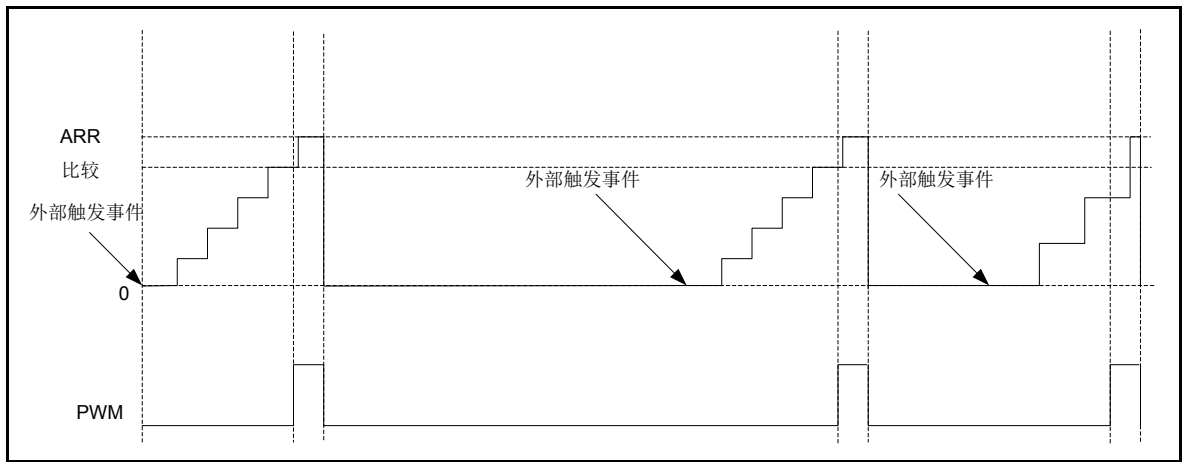


图 27-3 单发计数

若要进行连续计数，需软件置位 CNTSTRT。

选择外部触发的情况下，置位 CNTSTRT 后，触发事件将启动计数器开始连续计数。之后的触发事件将被忽略。如下图所示：

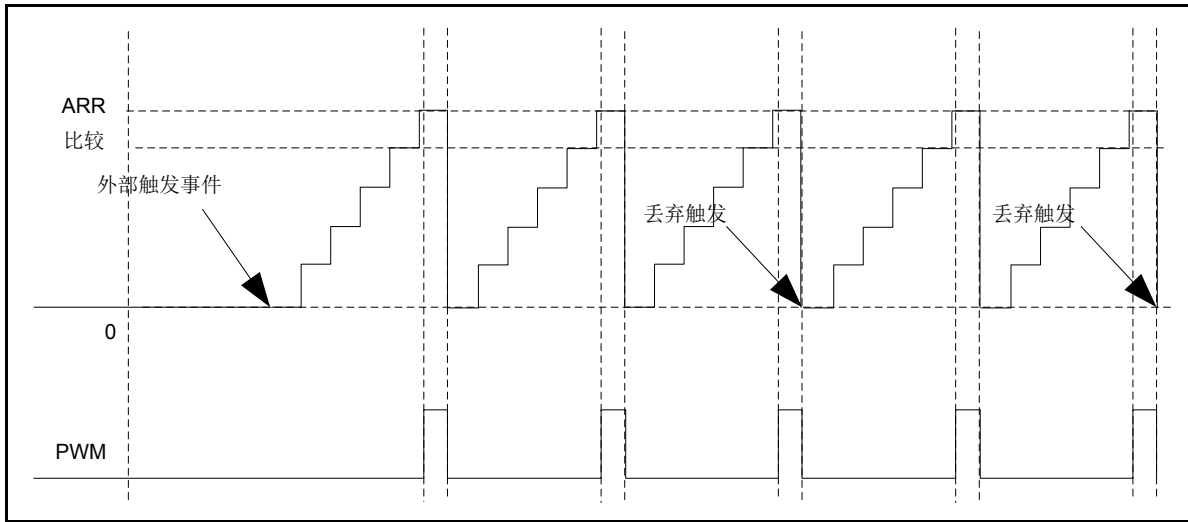


图 27-4 连续计数

如果采用软件启动的方式，置位 CTNSTRT 将启动计数器进行连续计数。

SNTSTRT 和 CNTSTRT 位只能在计数器被使能后（ENABLE 位为 1）才能置位。单发模式和连续模式之间可以相互切换。

在连续模式计数期间，置位 SNGSTRT 可将 LP16T 切换至单发模式。计数器在计数至 ARR 后将停止。

在单发模式计时期间，置位 CNTSTRT 将 LP16T 切换至连续计数模式。计数器计数至 ARR 后将重启计数。

### 27.4.5 输出波形

2 个 16 位寄存器，LP16T\_ARR（自动加载寄存器）和 LP16T\_CMP（比较寄存器），用于生成各种不同的波形，并输出到端口。

PWM: LP16T 输出在 LP16T\_CNT 和 LP16T\_CMP 匹配时置位，在 LP16T\_CNT 和 LP16T\_ARR 匹配时清零。

脉冲: LP16T 输出在 LP16T\_CNT 和 LP16T\_ARR 匹配时将置位一个计数时钟周期，在下次匹配之前一直处于清零状态。

翻转: LP16T 输出在 LP16T\_CNT 和 LP16T\_ARR 匹配时将当前输出翻转。

### 27.4.6 寄存器更新

LP16T\_ARR 和 LP16T\_CMP 寄存器可在 APB 总线写操作之后马上更新或者在计数器计完当前周期后更新。PRELOAD 可控制 LP16T\_ARR 和 LP16T\_CMP 的更新方式。

当 PRELOAD 为 0 时，LP16T\_ARR 和 LP16T\_CMP 在写操作之后马上更新。

当 PRELOAD 为 1 时，若计数器已经启动，LP16T\_ARR 和 LP16T\_CMP 将在当前计数周期结束后更新。

APB 总线和 LP16T 使用不同的时钟，所以在 APB 总线的写操作和设定值真正写到计数器

的比较寄存器之间是有一些延迟的。在这期间，须避免对同一个寄存器的写操作。

LP16T\_SYNCSTAT 的 ARRWBSY 和 CMPWBSY 标志位分别表示对 LP16T\_ARR 和 LP16T\_CMP 的写操作是否完成。

对 LP16T\_ARR 和 LP16T\_CMP 写之后，新的写命令必须在前一次写操作结束之后才能进行。连续的写操作可导致不可预期的结果。

### 27.4.7 中断

如果 LP16T\_IER 使能，以下事件可产生中断或唤醒事件

计数比较匹配

自动加载匹配

外部触发事件

## 27.5 特殊功能寄存器

### 27.5.1 寄存器列表

LP16T 寄存器列表		
名称	偏移地址	描述
LP16T 基地址: 4004_0000 <sub>H</sub>		
LP16T_CON0	000 <sub>H</sub>	LP16T 控制寄存器 0
LP16T_CON1	004 <sub>H</sub>	LP16T 控制寄存器
LP16T_ARR	008 <sub>H</sub>	LP16T 自动加载寄存器
LP16T_CNT	00C <sub>H</sub>	LP16T 计数寄存器
LP16T_CMP	010 <sub>H</sub>	LP16T 通道比较值
Reserved	014 <sub>H</sub>	保留
LP16T_IER	018 <sub>H</sub>	LP16T 中断使能寄存器
LP16T_ISR	01C <sub>H</sub>	LP16T 中断状态寄存器
LP16T_IFC	020 <sub>H</sub>	LP16T 中断状态清零寄存器
Reserved	024 <sub>H</sub>	保留
Reserved	028 <sub>H</sub>	保留
Reserved	02C <sub>H</sub>	保留
LP16T_UPDATE	030 <sub>H</sub>	LP16T 更新控制寄存器
LP16T_SYNCSTAT	034 <sub>H</sub>	LP16T 写同步状态寄存器

## 27.5.2 寄存器描述

### 27.5.2.1 LP16T控制寄存器 0 (LP16T\_CON0)

LP16T 控制寄存器 0 (LP16T_CON0)
偏移地址: 00 <sub>H</sub>
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>

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

Reserved	PRELOAD	WAVEPOL	WAVE	TRIGEN	Reserved	TRIGSEL	Reserved	PRESC	Reserved	TRGFLT	Reserved	CKFLT	Reserved	CKPOL	CKSEL
----------	---------	---------	------	--------	----------	---------	----------	-------	----------	--------	----------	-------	----------	-------	-------

Reserved	Bits 31-23	—	保留
PRELOAD	Bit 22	R/W	寄存器更新模式 控制LP16T_ARR和LP16T_CMP寄存器的更新 0: 寄存器在APB总线写操作后更新 1: 寄存器在LP16T计数周期结束后更新
WAVEPOL	Bit 21	R/W	输出波形极性 0: 输出不取反 1: 输出取反
WAVE	Bits 20-19	R/W	输出波形格式选择 00: 输出禁止 01: LP16T计数溢出时输出翻转 10: LP16T计数溢出时输出脉冲 11: LP16T输出PWM波
TRIGEN	Bits 18-17	R/W	触发使能及极性 TRIGEN控制LP16T是否由外部触发启动。如果选择外部触发, 有3中极性可供选择 00: 软件触发 01: 上升沿有效 10: 下降沿有效 11: 双边沿有效
Reserved	Bit 16	—	保留
TRIGSEL	Bits 15-13	R/W	触发选择 TRIGSEL在如下8个通道中选择触发源 000: ext_trig0 001: ext_trig1 010: ext_trig2 011: ext_trig3 100: ext_trig4 101: ext_trig5 110: ext_trig6

			111: ext_trig7
Reserved	Bit12	—	保留
PRESC	Bits 11-9	R/W	<p><b>时钟预分频</b> PRESC配置预分频系数。</p> <p>000: /1 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128</p>
Reserved	Bit8	R/W	保留
TRGFLT	Bits 7-6	R/W	<p><b>配置触发信号的数字滤波器</b> TRGFLT的值设定了滤波宽度（连续采样1或0的次数超过TRGFLT设定的个数才能被认为是合法的信号电平变化）</p> <p>00: 外部触发信号任何变化都是合法变化 01: 外部触发信号电平须维持2个时钟周期以上 10: 外部触发信号电平须维持4个时钟周期以上 11: 外部触发信号电平须维持8个时钟周期以上</p>
Reserved	Bit5	—	保留
CKFLT	Bits 4-3	R/W	<p><b>配置外部时钟的数字滤波器</b> CKFLT的值设定了滤波宽度（连续采样1或0的次数超过CKFLT设定的个数才能被认为是合法的信号电平变化）</p> <p>00: 外部时钟任何变化都是合法变化 01: 外部时钟信号电平须维持2个时钟周期以上 10: 外部时钟信号电平须维持4个时钟周期以上 11: 外部时钟信号电平须维持8个时钟周期以上</p>
Reserved	Bit2	—	保留
CKPOL	Bit1	R/W	<p><b>时钟极性</b> 如果LP16T采用外部时钟源，CKPOL用于配置计数的有效边沿</p> <p>0: 上升沿为计数有效边沿 1: 下降沿为计数有效边沿</p>
CKSEL	Bit0	R/W	<p><b>时钟选择</b> CKSEL用于选择哪个时钟源作为LP16T的计数时钟</p> <p>0: LP16T采用内部时钟源（APB时钟及其他片上时钟） 1: LP16T采用外部时钟源（从input1输入）</p>

### 27.5.2.2 LP16T控制寄存器 1 (LP16T\_CON1)

LP16T 控制寄存器 1 (LP16T_CON1)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CNTSTRT	SNGSTRT	ENABLE	

Reserved	Bits 31-3	—	保留
CNTSTRT	Bit 2	RW	<p><b>LP16T启动连续模式</b></p> <p>该位由软件置位，由硬件清零</p> <p>在软件启动方式 (TRIGEN[1:0]='00') 下，置位CNTSTRT将启动LP16T进入连续模式。</p> <p>在软件启动方式禁止时 (TRIGEN[1:0]不为00)，置位CNTSTRT后，LP16T将在检测到外部触发后启动连续计数。</p> <p>如果LP16T在单发计数模式下，置位CNTSTRT将使LP16T计数到LP16T_ARR后也会继续计数。</p> <p>该位只能在LP16T使能后进行设定。读取该位始终为0。</p>
SNGSTRT	Bit 1	RW	<p><b>LP16T启动单发模式</b></p> <p>该位由软件置位，由硬件清零</p> <p>在软件启动方式 (TRIGEN[1:0]='00') 下，置位SNGSTRT将启动LP16T进入单发模式。</p> <p>在软件启动方式禁止时 (TRIGEN[1:0]不为00)，置位SNGSTRT后，LP16T将在检测到外部触发后启动单发计数。</p> <p>如果LP16T在连续计数模式下，置位SNGSTRT将使LP16T计数到LP16T_ARR后停止计数。</p> <p>该位只能在LP16T使能后进行设定。读取该位始终为0。</p>
ENABLE	Bit 0	RW	<p><b>LP16T使能</b></p> <p>ENABLE位由软件置位或清零</p> <p>0:LP16T禁止</p> <p>1:LP16T使能</p>



### 27.5.2.3 LP16T自动加载寄存器 (LP16T\_ARR)

LP16T 自动加载寄存器 (LP16T_ARR)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ARR															

Reserved	Bits 31-16	—	保留
ARR	Bits 15-0	R/W	<b>自动加载值</b> ARR是LP16T的自动加载值。 ARR必须严格大于CMP[15: 0]的值。 ARR的值必须在LP16T使能后改写 (ENABLE设为‘1’)

### 27.5.2.4 LP16T计数寄存器 (LP16T\_CNT)

LP16T 计数寄存器 (LP16T_CNT)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CNT															

Reserved	Bits 31-16	—	保留
CNT	Bits 15-0	R	<b>计数值</b> 当LP16T正在以异步时钟计数, 读取LP16T_CNT可能会返回不可信赖值。所以, 在这种情况下有必要进行2次连续的读操作, 并且如果两次读取的值相同, 说明当前读到的计数值是可靠的。

### 27.5.2.5 LP16T比较值寄存器 (LP16T\_CMP)

LP16T 比较值寄存器 (LP16T_CMP)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CMP															

Reserved	Bits 31-16	—	保留
CMP	Bits 15-0	R/W	比较值 CMP作为LP16T的比较值 LP16T_CMP的值只能在LP16T使能后被改写 (ENABLE设为‘1’)

### 27.5.2.6 LP16T中断使能寄存器 (LP16T\_IER)

LP16T 中断使能寄存器 (LP16T_IER)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												EXTTRIGIE	ARRMIE	CMPMIE	

Reserved	Bits 31-3	—	保留
EXTTRIGIE	Bit 2	R/W	外部触发有效边沿中断 0: EXTTRIG中断禁止 1: EXTTRIG中断使能
ARRMIE	Bit 1	R/W	自动加载中断使能 0: ARRM中断禁止 1: ARRM中断使能
CMPMIE	Bit 0	R/W	比较匹配中断使能 0: CMPM中断禁止 1: CMPM中断使能

### 27.5.2.7 LP16T中断状态寄存器 (LP16T\_ISR)

LP16T 中断状态寄存器 (LP16T_ISR)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000XXX <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											EXTTRIG	ARRM	CMPM		

Reserved	Bits 31-3	—	保留
EXTTRIG	Bit 2	R	外部触发有效边沿事件 EXTTRIG由硬件置位，表示在所选择的外部触发通路上检测到有效边沿。该位通过软件操作LP16T_IFC清零。
ARRM	Bit 1	R	自动加载值匹配 ARRM由硬件置位，表示LP16T_CNT计数值达到LP16T_ARR寄存器的设定值。该位通过软件操作LP16T_IFC清零。
CMPM	Bit 0	R	比较匹配 CMPM由硬件置位，表示LP16T_CNT计数值达到LP16T_CMP寄存器的设定值。该位通过软件操作LP16T_IFC清零。

### 27.5.2.8 LP16T中断状态清零寄存器 (LP16T\_IFC)

LP16T 中断状态清零寄存器 (LP16T_IFC)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											EXTTRIG	ARRM	CMPM		

Reserved	Bits 31-3	—	保留
EXTTRIG	Bit 2	W1	外部触发有效边沿事件标志清零 对该位写1有效。
ARRM	Bit 1	W1	自动加载值匹配标志清零 对该位写1有效。
CMPM	Bit 0	W1	比较匹配标志清零 对该位写1有效。

### 27.5.2.9 LP16T更新控制寄存器 (LP16T\_UPDATE)

LP16T 更新控制寄存器 (LP16T_UPDATE)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											UDIS				

Reserved	Bits 31-1	R/W	保留
UDIS	Bit 0	R/W	寄存器更新 0: 寄存器写入后新值将被更新到低速时钟域 1: 寄存器写入后新值不被更新 置 1 后, 寄存器新写值将不被更新直到该位被清零。 用该位可控制多个寄存器同时更新。

### 27.5.2.10 LP16T写同步状态寄存器 (LP16T\_SYNCSTAT)

LP16T 写同步状态寄存器 (LP16T_SYNCSTAT)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												CMPWBSY	ARRWBSY	CRWBSY	Reserved

Reserved	Bits 31-4	—	保留
CMPWBSY	Bit 3	R	寄存器 <b>CMP</b> 写同步状态 0: 同步未开始或同步已结束 1: 同步中
ARRWBSY	Bit 2	R	寄存器 <b>ARR</b> 写同步状态 0: 同步未开始或同步已结束 1: 同步中
CON1WBSY	Bit 1	R	寄存器 <b>CON1</b> 写同步状态 0: 同步未开始或同步已结束 1: 同步中
Reserved	Bit 0	—	保留

## 第28章 串行总线(I2C)

### 28.1 概述

I2C 是两线双向的串行传输总线，提供了一种简单有效的方法来实现设备之间的数据交换。

I2C 是一种包括冲突检测与仲裁机制的多主机总线，如果两个或两个以上的主机试图同时控制总线时，其仲裁可以防止数据损坏。在此提供了标准模式(Sm)、快速模式(Fm)与极快速模式(Fm<sup>+</sup>)供用户选择。并且也提供 SMBus(系统管理总线)与 PMBus(电源管理总线)。

### 28.2 特性

- ◆ 可配置为主机或从机
- ◆ 多主机模式(Multimaster capability)
- ◆ 标准模式(up to 100 kHz)
- ◆ 快速模式(up to 400 kHz)
- ◆ 极快速模式(up to 1 MHz)
- ◆ 7 位与 10 位地址模式
- ◆ 提供 2 组 7 位从机地址(2 个地址，其中一个包括屏蔽)
- ◆ 提供所有 7 位地址应答模式
- ◆ 提供广播模式(General call)
- ◆ 可编程的设置时间和保持时间
- ◆ 可选择时钟延长(Optional clock stretching)
- ◆ 可配置数字滤波器
- ◆ 提供深度为 16 字节的 TX / RX FIFOs
- ◆ 提供 DMA 传输
- ◆ 提供 SMBus 标准
- ◆ 硬件 PEC(封包错误检查)产生与 ACK 控制
- ◆ 命令与数据应答控制
- ◆ 提供地址解析协议(Address resolution protocol(ARP) support)
- ◆ 提供可选择为主控者或设备(Host and Device support)
- ◆ 提供 SMBus 警报
- ◆ 提供侦测超时与闲置功能
- ◆ 提供 PMBus rev 1.1 标准

### 28.3 结构框图

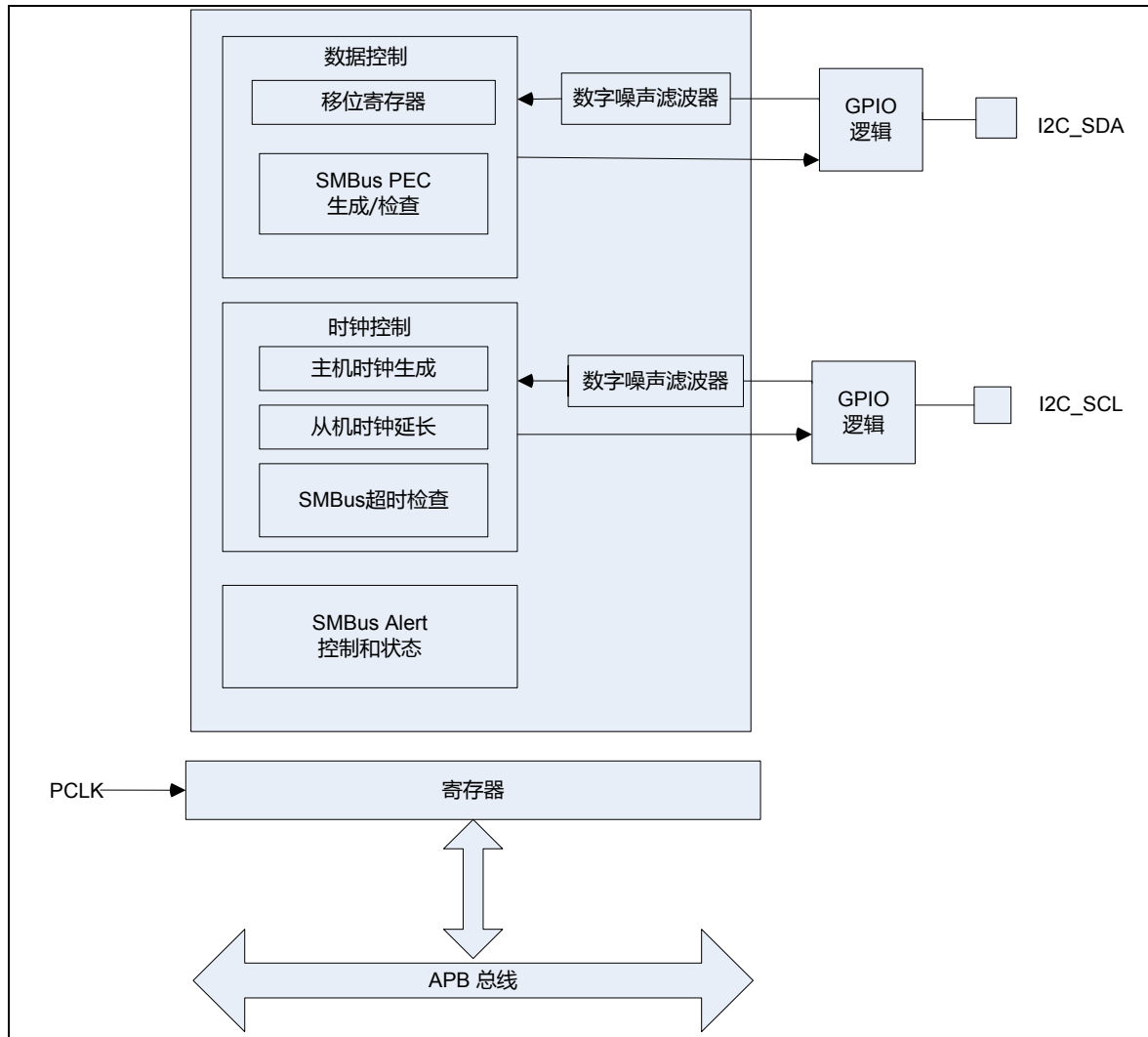


图 28-1 I2C 电路结构框图

### 28.4 功能描述

除了接收和发送数据外，I2C 接口还完成接收和发送数据的串并转换。I2C 接口可以通过数据引脚（SDA）和时钟引脚（SCL）连接标准模式（最高 100 kHz），快速模式（最高 400 kHz）或极快速模式（最高 1 MHz）的 I2C 总线。I2C 的中断由软件开启和关闭。

I2C 接口也可通过数据引脚（SDA）和时钟引脚（SCL）连接到 SMBus，如果支持 SMBus 功能，还可以使用可选的 SMBus Alert 引脚（SMBA）。

#### 28.4.1 I2C 总线协议

I2C 是一种双线双向串行总线，可在设备之间提供简单有效的数据交换方法。I2C 标准是真正的多主机总线，包含冲突检测和仲裁机制，如果两个或多个主机同时尝试控制总线，则可防止数据损坏。数据在主机和从机之间逐字节同步传输到串行数据（SDA）和串行时钟（SCL）线上，每个数据字节长度为 8 位。



### 28.4.1.1 START和STOP条件协议

I2C 规范将启动条件定义为在 SCL 时钟线为高电平时 SDA 数据线从高电平到低电平的转换。启动条件始终由主机生成，表示总线从空闲状态转换为活动状态。每个数据位对应一个 SCL 时钟脉冲，数据传输时先传输 MSB。每个传输的字节后面都有一个对应的应答位。当 SCL 为高电平时，SDA 线上的电平转换被解释为命令（START 或 STOP）。规定在 SCL 的高电平期间对每个数据位进行采样，因此 SDA 线上的数据只可以在 SCL 的低电平期间改变，并且必须在 SCL 的高电平期间保持稳定。

当总线空闲时，SCL 和 SDA 信号都通过总线上的外部上拉电阻拉高。当主机想要在总线上开始传输时，主机发出 START 信号，即 SCL 为 1 时，SDA 从高到低转换。当主机想要终止传输时，主机发出 STOP 条件，即 SCL 为 1 时，SDA 从低到高转换。下图为 START 和 STOP 条件的产生时序。当数据在总线上传输时，SCL 为 1 期间，SDA 线上的数据必须稳定。

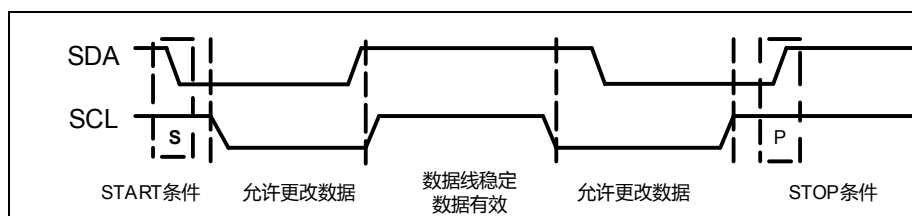


图 28-2 START 和 STOP 条件

### 28.4.1.2 应答位

数据传输时要求带有应答位，应答位相关的时钟脉冲由主机产生。主机在应答时钟脉冲期间释放 SDA 线（HIGH）。从机必须在应答时钟脉冲低电平期间下拉 SDA 线，以便在时钟脉冲的高电平期间保持稳定的低电平。同时还必须考虑建立和保持时间。通常情况下，除非消息以 CBUS 地址开始，已经被寻址的从机必须在接收到每个字节后生成应答位。

当从机不应答从机地址时（例如从机正在执行某些实时功能而无法接收或发送），从机必须将数据线保持为高电平。然后，主机可以生成 STOP 条件以中止传输，或者重复 START 条件以启动新传输。

如果从机应答了从机地址，但是，传输后的某个时间不能再接收数据字节，则主机必须中止传输。这由从机在随后的第一个字节上产生非应答来指示。从机使数据线保持高电平，主机产生 STOP 或重复 START 条件。

如果主机进行接收传输，它必须通过在从机输出的最后一个字节上产生非应答来向从机发送数据结束信号。从机必须在应答时钟脉冲期间释放数据线，以允许主机产生 STOP 或重复 START 条件。

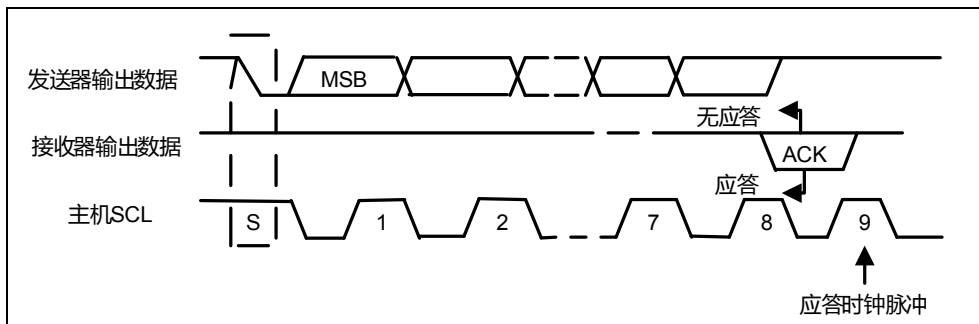


图 28-3 I2C 总线上的应答

### 28.4.1.3 I2C寻址协议

从机有两种地址格式：7 位地址格式和 10 位地址格式。7 位地址格式下，第一个字节的前 7 位（bit7~bit1）设置从地址，LSB 位（bit0）是 R/W 位。当 bit8 置 0 时，主机向从机写数据。当 bit8 置 1 时，主机从从机读数据。数据首先传输最高有效位（MSB）。10 位地址格式下，主机传输两个字节以设置 10 位地址。第一个字节传输时，前 5 位（bit7~bit3）通知从机这是一个 10 位传输，接着是后 2 位（bit2~bit1），它为从机地址的第 9 位和第 8 位，LSB 位（bit8）是 R/W 位。传输的第二个字节设置从地址的 bit7~bit0。

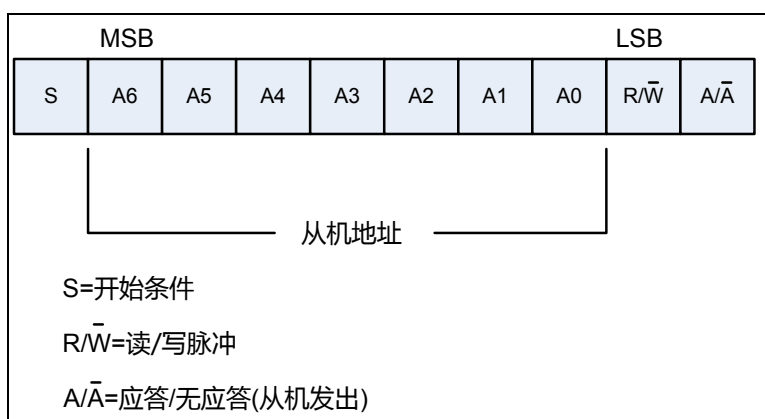


图 28-4 7 位地址格式

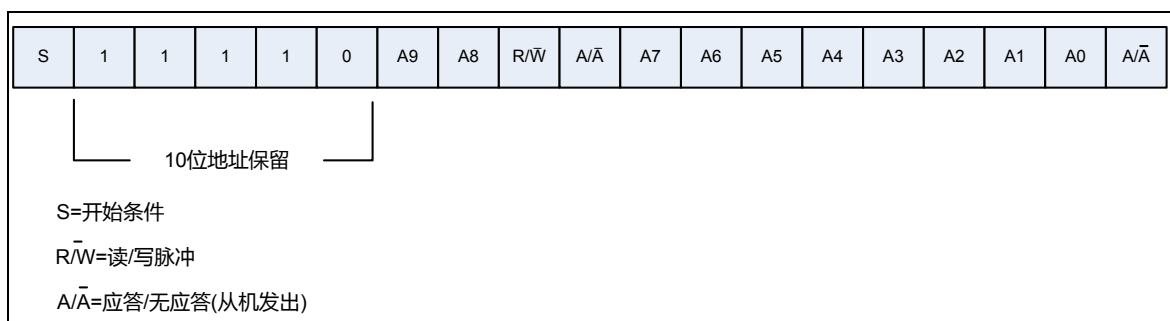


图 28-5 10 位地址格式

1、 从地址	2、 R/W位	3、 描述
4、 0000 000	5、 0	6、 广播地址
7、 0000 000	8、 1	9、 START字节
10、 0000 001	11、 X	12、 CBUS地址
13、 1111 0XX	14、 X	15、 10位从机寻址

表 28-1 10 位地址格式第一个字节中位的定义

### 28. 4. 1. 4 I2C发送和接收协议

所有数据都以字节格式传输，每次传输的字节数没有限制。在主机发送地址和 RW 位或主机向从机发送一个字节数据后，接收的从机必须响应应答脉冲。当接收的从机没有响应应答脉冲时，主机通过发出 STOP 条件来中止传输。从机应使 SDA 线保持高电平，以便主机可以中止传输。如果主机正在发送数据，则接收的从机在接收到每个数据字节后都应该响应应答脉冲。传输格式如下：

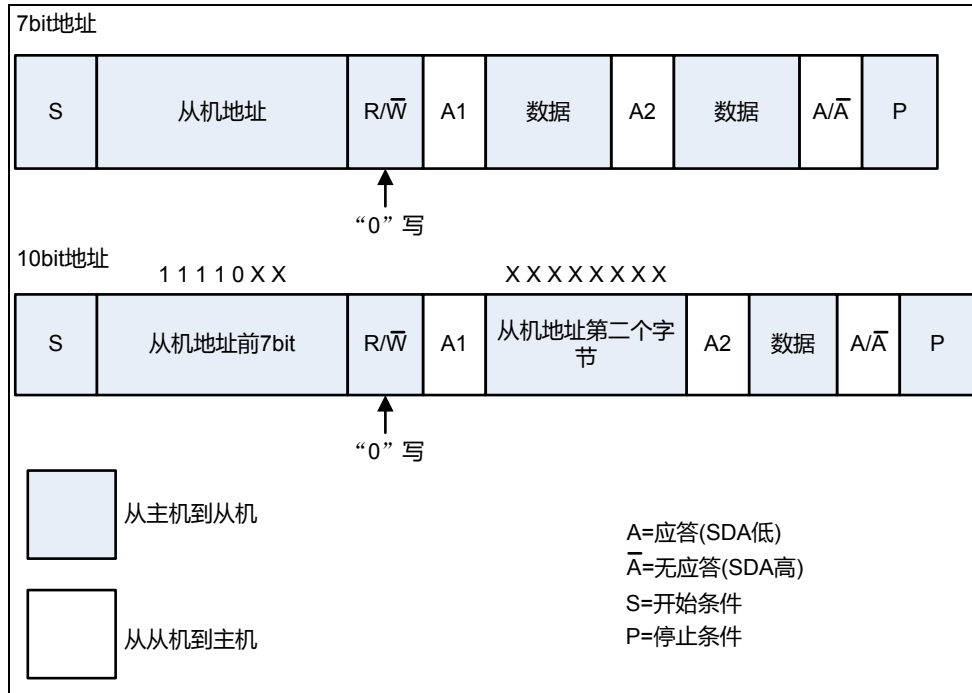


图 28-6 主机-发送协议

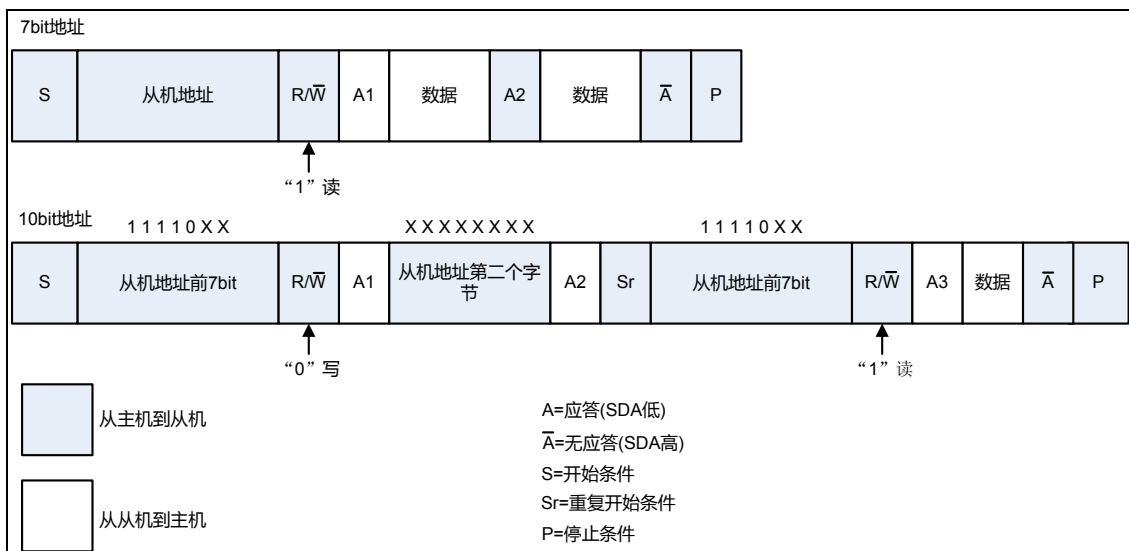


图 28-7 主机-接收协议

### 28.4.2 I2C时钟要求

I2C 内核由 I2CCLK 提供时钟。

I2CCLK 周期  $T_{I2CCLK}$  必须符合以下条件：

$$T_{I2CCLK} < (T_{LOW} - T_{Filter}) / 4 \text{ 且 } T_{I2CCLK} < T_{HIGH}$$

$T_{LOW}$ : SCL 低电平时间。

$T_{HIGH}$ : SCL 高电平时间。

$T_{Filter}$ : 使能时，数字滤波器带来的延迟总和。数字滤波器延迟为  $DNF * T_{I2CCLK}$ 。

PCLK 时钟周期  $T_{PCLK}$  必须符合以下条件：

$$T_{PCLK} < 4/3 T_{SCL}$$

$T_{SCL}$ : SCL 周期。

### 28.4.3 数据传输

SDA 线上的每个字节必须为 8 位长。每次传输可以传输的字节数不受限制。每个字节后面都必须有一个应答位。首先使用最高有效位 (MSB) 传输数据。如果从机不能接收或发送另一个完整的数据字节，则它可以将时钟线 SCL 保持为低电平以强制主机进入等待状态，直到它执行完了某些其他功能，例如服务于内部中断。当从机准备好接收另一个数据字节并释放时钟线 SCL 时，数据传输继续。

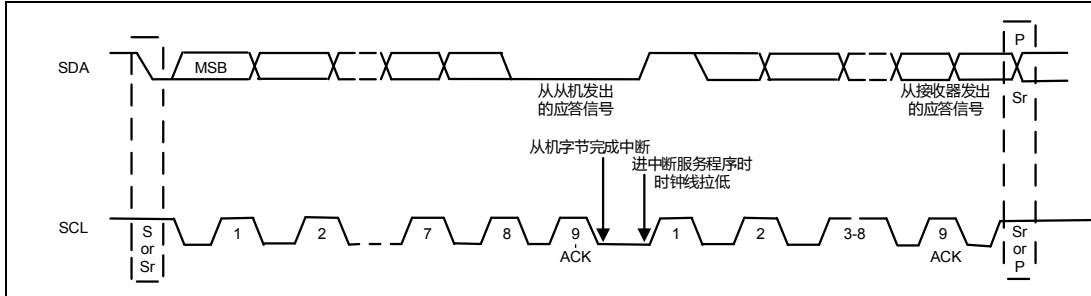


图 28-8 I2C 总线上的数据传输

#### 接收

SDA 的输入填充移位寄存器。在第 8 个 SCL 脉冲之后（当接收到完整的数据字节时），如果接收 FIFO 尚未满 ( $RXF = 0$ )，则将其复制到接收 FIFO 中。如果  $RXF = 1$ ，意味着尚未读取先前接收的数据字节，此时 SCL 线会在第 9 个 SCL 脉冲之后延长其低电平，直到读取 I2Cx\_RXDATA，使得接收 FIFO 未滿 ( $RXF = 0$ )。

#### 传输

如果发送 FIFO 不为空 ( $TXE = 0$ )，则第 9 个 SCL 脉冲（包括应答脉冲）之后将其内容复制到移位寄存器中。然后移位寄存器的内容会在 SDA 线上发送。如果  $TXE = 1$ ，意味着发送 FIFO 尚未写入数据，SCL 线将被拉低，直到有数据通过 I2Cx\_TXDATA 写入到发送 FIFO。

## 硬件传输管理

I2C 具有嵌入在硬件中的字节计数器,以便管理字节传输并在各种模式下关闭通信,例如:在主模式下生成 NACK、STOP 和 RESTART,从接收器模式下的 ACK 控制,支持 SMBus 功能时的 PEC 生成和检查。

在主模式下,字节计数器始终被使能。在从模式下,字节计数器默认被禁止,但可以通过软件设置 I2Cx\_CON1 寄存器中的 SBC(从机字节控制)位来使能。要传输的字节数在 I2Cx\_CON2 寄存器的 NBYTES<7:0>位字段中设置。如果要传输的字节数(NBYTES)大于 255,或者接收时想要控制接收数据字节的应答值,则必须通过将 I2Cx\_CON2 寄存器中的 RELOAD 位置 1 来选择重载模式。在此模式下,当达到 NBYTES 中所设置的字节数时,TCR 标志置位,如果 TCRIE 置位,则产生中断。只要设置了 TCR 标志,SCL 就会被延长。当软件对 NBYTES 写入非零值时,会清除 TCR。

当 NBYTES 计数器重新加载最后一个字节数时,必须清除 RELOAD 位。

当主模式下 RELOAD = 0 时,计数器可用于 2 种模式:

- ◇ 自动结束模式(I2Cx\_CON2 寄存器中的 AUTOEND =1)。在此模式下,一旦传输了 NBYTES<7:0>位字段中编程的字节数,主机就会自动发送 STOP 条件。
- ◇ 软件结束模式(I2Cx\_CON2 寄存器中的 AUTOEND ='0')。在此模式下,一旦传输了 NBYTES<7:0>位字段中编程的字节数,就会产生软件操作;如果 TCIE 位置 1,则 TC 标志置 1,产生中断。只要 TC 标志置位,SCL 信号就会被拉长。当 I2Cx\_CON2 寄存器中的 START 或 STOP 位置 1 时,TC 标志由软件清零。当主机要发送 RESTART 条件时,必须使用此模式。

### 28.4.4 I2C主机模式

#### I2C 主机初始化

在启用外设之前，必须通过将 I2Cx\_TIMINGR 寄存器中的 SCLH 和 SCLL 位置 1 来配置 I2C 主时钟,实现时钟同步机制以支持多主机环境和从机时钟延长。

为了允许时钟同步:

- ◇ 从 SCL 低电平内部检测开始，使用 SCLL 计数器计数低电平时钟。
- ◇ 从 SCL 高电平内部检测开始，使用 SCLH 计数器计数高电平时钟。

根据 SCL 下降沿，主机在 T<sub>SYNC</sub> 延迟后检测到自己的 SCL 低电平。一旦 SCLL 计数器达到 I2Cx\_TIMINGR 寄存器中 SCLL<7:0>位中设置的值，主机就会将 SCL 释放为高电平，SCL 输入数字噪声滤波器且与 I2Cx 时钟同步。

在 T<sub>SYNC</sub> 产生延迟后，主机会检测自己的 SCL 高电平，具体取决于 SCL 上升沿，SCL 输入数字噪声滤波器且与 I2Cx 时钟同步。

一旦 SCLH 计数器达到 I2Cx\_TIMINGR 寄存器中的 SCLH<7:0>位编程的值，主机就会将 SCL 置为低电平。

因此，主时钟周期为:

$$TSCL = TSYNC1 + TSYNC2 + \{[(SCLH+1) + (SCLL+1)] * (PRESC+1) * TI2CCLK\}$$

T<sub>SYNC1</sub> 的持续时间取决于以下参数:

- ◇ SCL 下降斜率
- ◇ 启用时，数字滤波器引起的输入延迟: DNF \* T<sub>I2CCLK</sub>
- ◇ 由于 SCL 与 I2CCLK 时钟同步 (2 至 3 个 I2CCLK 周期) 导致的延迟

T<sub>SYNC2</sub> 的持续时间取决于以下参数:

- ◇ SCL 上升斜率
- ◇ 启用时，数字滤波器引起的输入延迟: DNF \* T<sub>I2CCLK</sub>
- ◇ 由于 SCL 与 I2CCLK 时钟同步 (2 至 3 个 I2CCLK 周期) 导致的延迟

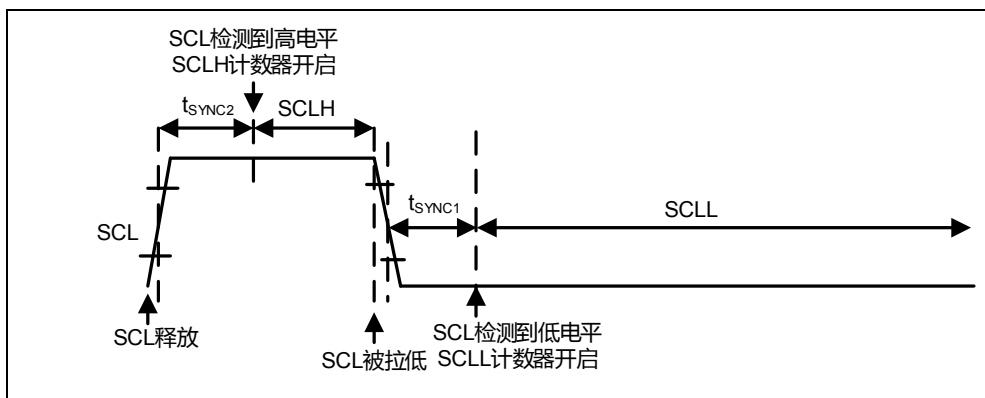


图 28-9 主时钟产生

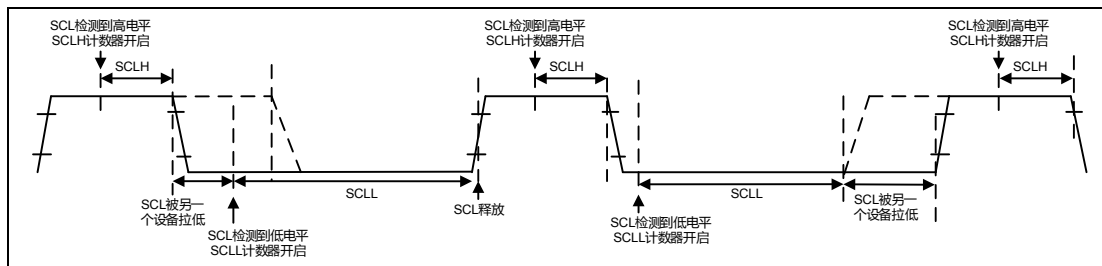


图 28-10 SCL 主时钟同步

### 主机通信初始化（地址阶段）

为了启动通信，用户必须为 I2Cx\_CON2 寄存器中寻址的从机编程以下参数：

- ◇ 寻址模式（7 位或 10 位）：ADD10
- ◇ 要发送的从站地址：SADD<9:0>
- ◇ 传输方向：RD\_WRN
- ◇ 如果是 10 位地址读取：HEAD10R 位。必须配置 HEAD10R 以指示是否必须发送完整的地址序列，或者仅在方向改变时发送地址标头。
- ◇ 要传输的字节数：NBYTES<7:0>。如果字节数等于或大于 255 个字节，则 NBYTES<7:0>最初必须填充 0xFF。

然后，用户必须将 I2Cx\_CON2 寄存器中的 START 位置 1。当 START 位置 1 时，不允许更改所有上述位。一旦检测到总线空闲（BUSY = 0）后，主机就会自动发送 START 条件，再发送从机地址。

在仲裁丢失的情况下，主机自动切换回从机模式，如果它作为寻址从机，则可以应答自己的地址。

注 1：无论接收到的应答值是什么，当总线上的从机地址发送时，START 位由硬件复位。如果发生仲裁丢失，则 START 位也由硬件复位。在 10 位寻址模式下，当从机地址前 7 位被从机 NACK 时，主机将自动重新启动从机地址传输，直到收到 ACK。如果在 START 位置 1 时 I2C 被寻址为从机（ADDRRI = 1），则 I2C 切换到从机模式，当 ADDRRI 位置 1 时，START 位清零。

注 2：对重复启动条件应用相同的过程。在这种情况下，BUSY = 1。

### 初始化主机 10 位从地址寻址接收模式

- ◇ 如果从机地址采用 10 位格式，用户可以通过清零 I2Cx\_CON2 寄存器中的 HEAD10R 位来选择发送完整的读序列。在这种情况下，主机在 START 位置 1 后自动发送以下完整序列：启动+从机地址 10 位标头写+从机地址第 2 字节+重新启动+从机地址 10 位标头读取；

如果主机寻址 10 位地址从机，将数据发送到该从机，然后从同一从机读取数据，则必须先完成主机传输流程，然后使用 HEAD10R = 1 配置的 10 位从地址设置重复启动。在这种情况下，主机发送此序列：重新启动+从机地址 10 位标头读取。



### 主机传送

在主机开始传送之前，需先配置 **I2C\_CON2** 寄存器中 **NBYTES** 位，当传输字节大于 255 时，需要额外配置 **RELOAD** 位。在此配置下，当 **NBYTES** 配置的笔数传送完成后，**I2C\_RIF** 寄存器中 **TCRRI** 位会置位，此时 **SCL** 时钟会保持低电平，直到 **NBYTES** 位重新写入新的数值以及对 **I2C\_TXDATA** 寄存器写入发送数据后，才会继续进行传输。

当从机回应 **NACK**，**I2C\_RIF** 寄存器中 **NACKRI** 位会置位，并且接下来主机会自动发送停止信号 **STOP**。

当从机响应 **ACK**，且 **RELOAD = 0**、**NBYTES** 所配置的笔数都已经传完时，会有以下情况：

- ◇ 若 **AUTOEND = 1**，此时会自动发送停止讯号 **STOP**。
- ◇ 若 **AUTOEND = 0**，此时主机会将 **SCL** 时钟保持低电平，等待软件控制后续操作：

**RESTART**：对 **I2C\_CON2** 寄存器中 **START** 置位，此时会发送重启信号。

**STOP**：对 **I2C\_CON2** 寄存器中 **STOP** 置位，此时会发送停止信号。

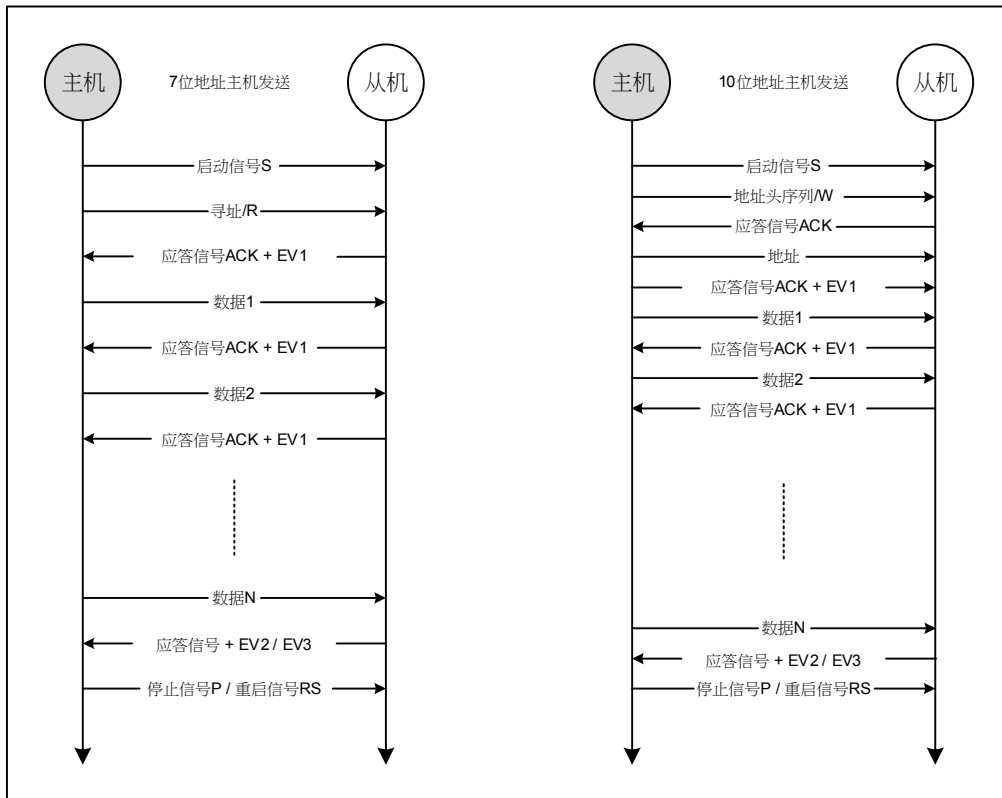


图 28-11 主机发送的传输序列图

**注 1：** S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答  
**注 2：** EV1 = 传输尚未完成事件，判断 **I2C\_STAT.TXTH**，若低于水平则对 **I2C\_TXDATA** 写入数据。  
**注 3：** EV2 = 当应答信号为 ACK 时，若传输已完成，后续操作为停止信号 **STOP** 或重启信号 **RESTART**。  
**注 4：** EV3 = 当应答信号为 NACK 时，后续操作为停止信号 **STOP**。  
**注 5：** 如果软件列在当前传送字节传输结束之前尚未写入下一个字节，导致 TX FIFO 为空时，EV1 事件将会延长 SCL 时钟低电平时间。

### 主机接收

在主机开始接收之前，需先配置 **I2C\_CON2** 寄存器中 **NBYTES** 位，当传输字节大于 255 时，需要额外配置 **RELOAD** 位。在此配置下，当 **NBYTES** 配置的笔数传送完成后，**I2C\_RIF** 寄存器中 **TCRRI** 位会置位，此时 **SCL** 时钟会保持低电平，直到 **NBYTES** 位重新写入新的数值后，才会继续进行传输。

当 **RELOAD = 0** 并且 **NBYTES** 所配置的笔数都已经传完时，会有以下情况：

- ◇ 若 **AUTOEND = 1**，此时会自动发送非应答信号 **NACK** 与停止讯号 **STOP**。
- ◇ 若 **AUTOEND = 0**，此时会自动发送非应答信号 **NACK**，并将 **SCL** 时钟保持低电平，等待软件控制后续操作：

**RESTART**：对 **I2C\_CON2** 寄存器中 **START** 置位，此时会发送重启信号。

**STOP**：对 **I2C\_CON2** 寄存器中 **STOP** 置位，此时会发送停止信号。

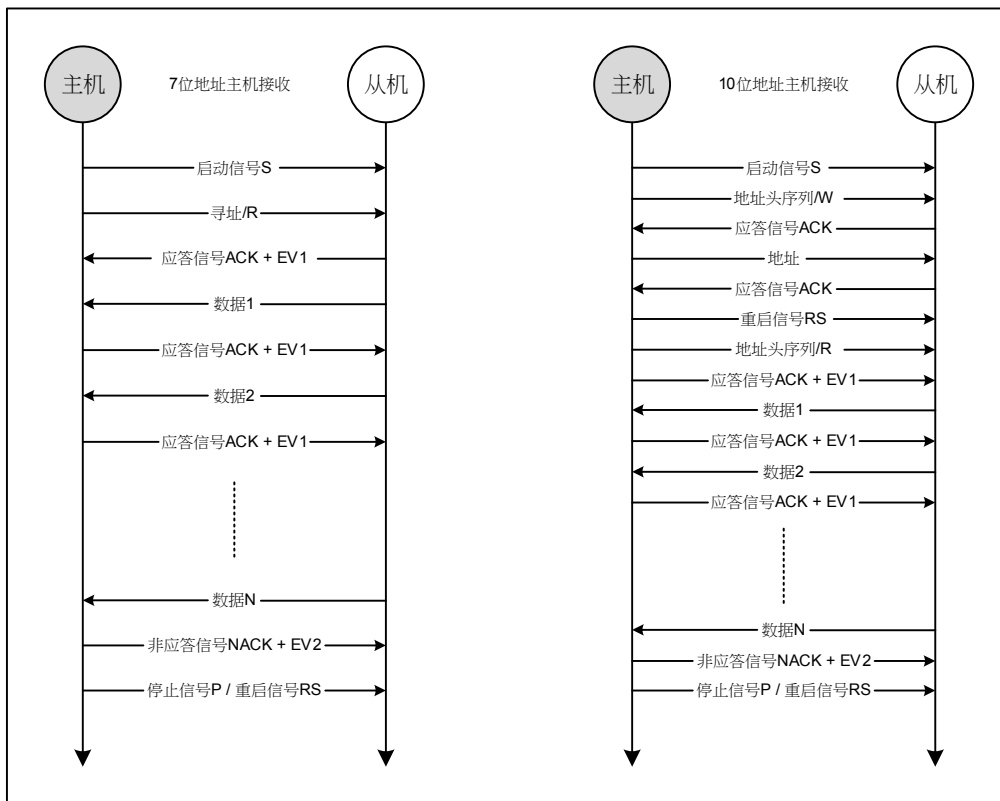


图 28-12 主机接收的传输序列图

**注 1：** S=起始位，RS=重复起始位，P=停止位，ACK=应答，NACK=非应答

**注 2：** EV1 = 传输尚未完成事件，判断 **I2C\_STAT.RXTH**，若高于水平则对 **I2C\_RXDATA** 读出数据。

**注 3：** EV2 = 传输完成事件，此时会自动发送非应答信号 **NACK**，后续根据软件配置来决定是发送停止信号 **STOP** 还是重启信号 **RESTART**。

**注 4：** 如果软件列在当前传送字节传输结束之前尚读出数据，导致 **RX FIFO** 满时，EV1 事件将会延长 **SCL** 时钟低电平时间。

## 28.4.5 I2C从机模式

### I2C 从机初始化

为了在从机模式下工作，用户必须至少启用一个从机地址。两个寄存器 I2Cx\_ADDR1 和 I2Cx\_ADDR2 可用于设置从机自身地址 OA1 和 OA2。

- ◇ 通过将 I2Cx\_ADDR1 寄存器中的 OA1MODE 位置 1，可以在 7 位模式（默认情况下）或 10 位寻址模式下配置 OA1。通过将 I2Cx\_ADDR1 寄存器中的 OA1EN 位置 1 来启用 OA1。
- ◇ 如果需要额外的从地址，可以配置第二个从地址 OA2。通过配置 I2Cx\_ADDR2 寄存器中的 OA2MSK<2:0>位，最多可以屏蔽 OA2<6:0>。因此，对于配置为 1 至 6 的 OA2MSK，仅 OA2<6:1>，OA2<6:2>，OA2<6:3>，OA2<6:4>，OA2<6:5>或 OA2<6>与收到的地址相比较。一旦 OA2MSK 不等于 0，OA2 的地址比较器就会排除未被应答的 I2C 保留地址（0000XXX 和 1111XXX）。如果 OA2MSK = 7，则应答所有接收的 7 位地址（保留地址除外）。OA2 始终是 7 位地址。

当 OA2MSK = 0 的情况下且在 I2Cx\_ADDR1 或 I2Cx\_ADDR2 寄存器中设置了特定启用位置，则可应答这些保留地址。通过将 I2Cx\_ADDR2 寄存器中的 OA2EN 位置 1 启用 OA2。

- ◇ 通过将 I2Cx\_CON1 寄存器中的 GCEN 位置 1 使能广播地址。

当启用地址选择时，地址匹配中断标志状态置 1，如果 ADDRIE 位置 1，则产生中断。

默认情况下，从机使用其时钟延长功能，这意味着它在需要时将 SCL 信号拉到低电平，以执行软件操作。如果主机不支持时钟延长，则必须在 I2Cx\_CON1 寄存器中将 I2C 配置为 NOSTRETCH = 1。

收到地址匹配中断后，如果启用了多个地址，用户必须读取 I2Cx\_STAT 寄存器中的 ADDCODE<6:0>位，以检查匹配的地址。还必须检查 DIR 标志以了解传输方向。

### 从机时钟延长（NOSTRETCH = 0）

在默认模式下，I2C 从机在以下情况下延长 SCL 时钟：

- ◇ 在传输过程中，如果先前的数据传输完成且没有在 I2Cx\_TXDATA 寄存器中写入新数据。当数据写入 I2Cx\_TXDATA 寄存器时，将释放此延长。
- ◇ 在接收时，I2Cx\_RXDATA 寄存器尚未读取且接收 FIFO 已满。读取 I2Cx\_RXDATA 且接收 FIFO 未滿时，将释放此延长。
- ◇ 从机字节控制模式下，当发生 TCR = 1 时，重载模式（SBC = 1 且 RELOAD = 1），表示数据字节已被传输。此时通过在 NBYTES<7:0>字段中写入非零值清除 TCR 时，释放该延长。
- ◇ 从机应答控制模式下，当收到地址或数据时，每个字节的第 8 和第 9 个 SCL 脉冲之间会将 SCL 延长。当设置应答更新时，将会释放该延长并回应应答脉冲。
- ◇ 在 SCL 下降沿检测后，I2C 拉低 SCL 延长单位为
$$[(SDADEL+SCLDEL+1) * (PRESC+1) + 1] * T_{I2CCLK}$$

### 从机没有时钟延长（NOSTRETCH = 1）

当 I2Cx\_CON1 寄存器中的 NOSTRETCH = 1 时，I2C 从机不会延长 SCL 信号。

- ◇ 发送时，必须在与传输相对应的第一个 SCL 脉冲之前将数据写入 I2Cx\_TXDATA 寄存器。如果不是，则发生欠载运算，在 I2Cx\_STAT 寄存器中设置 TXUD 标志，如果 I2Cx\_IER 寄存器中的 TXUDIE 位置 1，则会产生中断。
- ◇ 接收时，如果接收 FIFO 满时，必须在下一个数据字节的第 9 个 SCL 脉冲（应答脉冲）之前从 I2Cx\_RXDATA 寄存器中读取数据。如果发生溢出，则在 I2Cx\_STAT 寄存器中设置 RXOV 标志，如果 I2Cx\_IER 寄存器中的 RXOVIE 位置 1，则会产生中断。

### 从机字节控制模式

为了在从机接收模式下允许字节 ACK 控制，必须通过将 I2Cx\_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。这需要符合 SMBus 标准。

必须选择重载模式，以便在从机接收模式（RELOAD = 1）中允许字节控制。要控制每个字节，必须在 ADDR 中断子程序中将 NBYTES 初始化为 0x1，并在每个接收到的字节后重新加载到 0x1。当接收到该字节时，TCR 位置 1，在第 9 个 SCL 脉冲之后将 SCL 信号拉低。用户可以从 I2Cx\_RXDATA 寄存器读取数据，然后通过配置 I2Cx\_CON2 寄存器中的 NACK 位决定下一个数据是否应答。通过将 NBYTES 设置为非零值来释放 SCL 延伸：发送应答或不应答。

NBYTES 可以加载大于 0x1 的值，在这种情况下，接收流程在 NBYTES 数据接收期间是连续的。

注释：当禁止 I2C 时，才可配置 SBC 位。当 TCR = 1 时，此时可以更改 RELOAD 位值。

警告：从机字节控制模式与 NOSTRETCH 模式不兼容。不允许在 NOSTRETCH = 1 时设置 SBC。

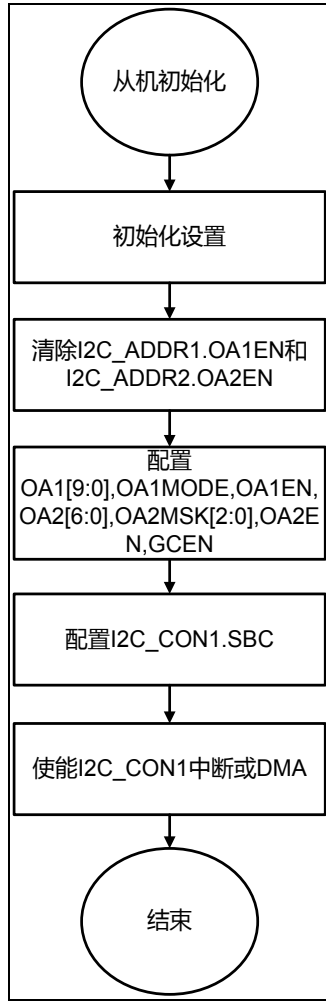


图 28-13 从机初始化流程图

### 从机传送

在接收到匹配的地址时，**I2C\_RIF** 寄存器中 **ADDRRI** 位会置位，此时若 TX FIFO 中已准备好要发送的数据时，从机会通过内部移位寄存器将 TX FIFO 中的字节发送到 SDA 线。若 TX FIFO 此时为空，从机会延长 SCL 低电平时间，直到 TX FIFO 透过 **I2C\_TXDATA** 寄存器写入发送数据为止。

当发送成功，主机会回应答信号，当主机回应 **NACK** 时，此时 **I2C\_RIF** 寄存器中 **NACKRI** 位会置位，此时从机会自动释放 SCL 与 SDA 总线让主机能够发送后续的 **STOP** 或 **RESTART** 命令。

当主机回应 **STOP** 时，此时 **I2C\_RIF** 寄存器中 **STOPRI** 位会置位，并结束通信，等待下一次收到匹配的地址。然而，若 TX FIFO 内还有尚未传送的字节，使用者可以选择下一次地址匹配时，继续传送 FIFO 中的数据，或是对 **I2C\_FCON** 寄存器中 **TXFRST** 位置位，来清除 FIFO 中的字节，下一次传送新的字节。

当从机字节控制启动时，需要传送多少字节需在 **I2C\_CON2** 寄存器中 **NBYTES** 内配置。

**警告：**当 **NOSTRETCH** 模式启动时，由于 SCL 时钟无法被延长，因此需要传送的字节需要被提前写入 TX FIFO 内。

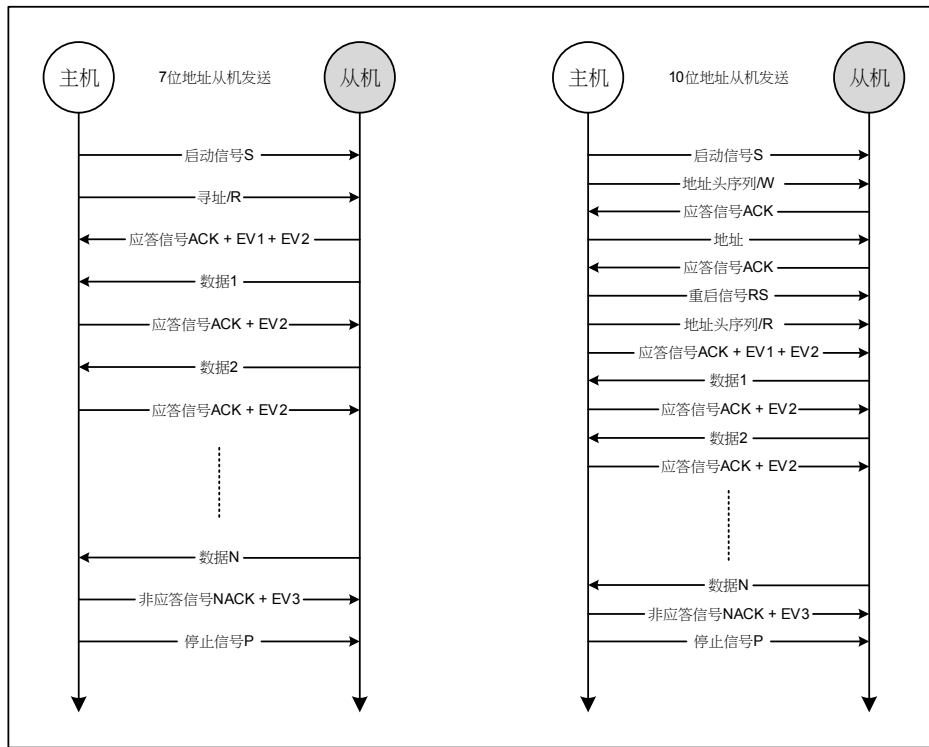


图 28-14 从机发送的传输序列图

- 注 1: S=起始位, RS=重复起始位, P=停止位, ACK=应答, NACK=非应答
- 注 2: EV1 = 当收到匹配地址时, `I2C_RIF.ADDRRI = 1`, 对 `I2C_ICR.ADDRIC` 置位来清除中断。
- 注 3: EV2 = 判断 `I2C_STAT.TXTH`, 若低于水平则对 `I2C_TXDATA` 写入数据。
- 注 4: EV3 = 当收到 NACK 时, `I2C_RIF.NACKRI = 1`, 对 `I2C_ICR.NACKIC` 置位来清除中断。此时还需判断 TX FIFO 内是否还有尚未发送的字节, 若有则软件需要额外处理。
- 注 5: 如果软件列在当前传送字节传输结束之前尚未写入下一个字节, 导致 TX FIFO 为空时, EV2 事件将会延长 SCL 时钟低电平时间。

### 从机接收

在接收到匹配的地址时, `I2C_RIF` 寄存器中 `ADDRRI` 位会置位, 从机此时会通过内部移位寄存器将 SDA 总在线的字节保存到 RX FIFO 中, 无论 RX FIFO 是否满, 从机都会自动发送应答信号, 差异仅在应答信号发送后, FIFO 满时会延长 SCL 时钟低电平时间, 等待软件将 RX FIFO 中的字节取出后, 才会释放 SCL 时钟。

当主机发送 STOP 时, 此时 `I2C_RIF` 寄存器中 `STOPRI` 位会置位, 并结束通信, 等待下一次收到匹配的地址。

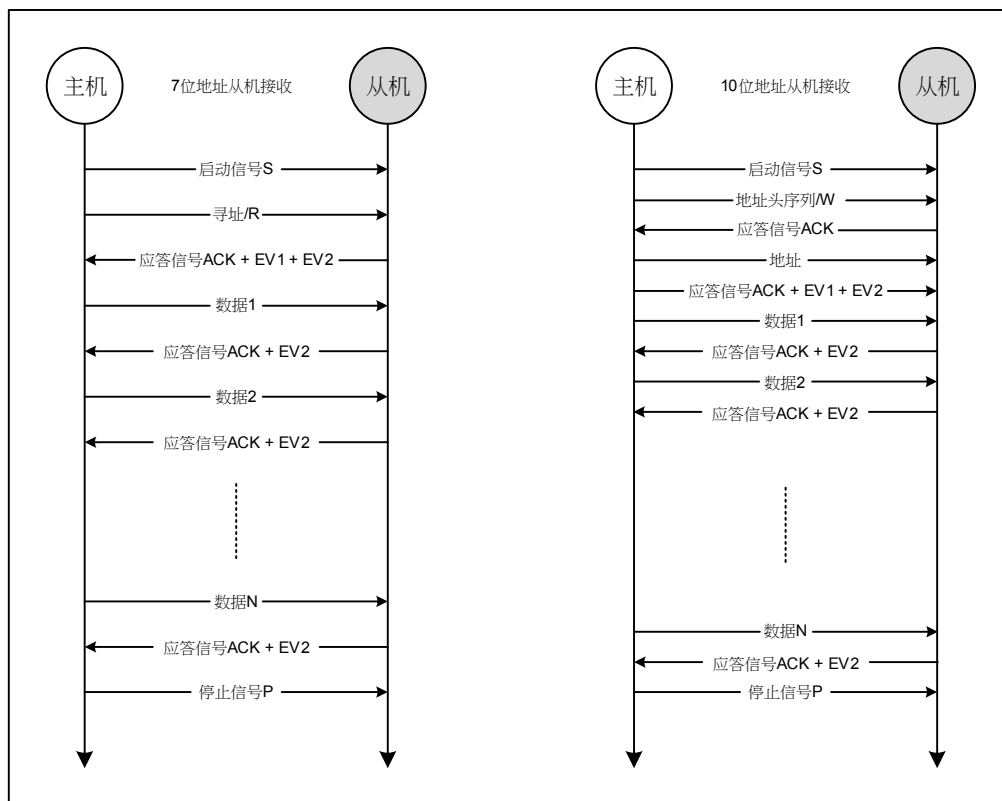


图 28-15 从机接收的传输序列图

- 注 1: S=起始位, RS=重复起始位, P=停止位, ACK=应答, NACK=非应答
- 注 2: EV1 = 当收到匹配地址时, `I2C_RIF.ADDRRI = 1`, 对 `I2C_ICR.ADDRIC` 置位来清除中断。
- 注 3: EV2 = 判断 `I2C_STAT.RXTH`, 若高于水平则对 `I2C_RXDATA` 取出数据。
- 注 4: 如果软件列在当前传送字节传输结束时, 导致 RX FIFO 满, EV2 事件将会延长 SCL 时钟低电平时间。

### 28.4.6 I2Cx\_TIMINGR寄存器的配置的例子

下表提供了如何对 I2Cx\_TIMINGR 进行编程以获得符合 I2C 规范的时序的示例。

16、参数	17、标准模式 (Sm)		18、快速模式 (Fm)	19、极快速模式 (Fm+)
	20、10 kHz	21、100 kHz	22、400 kHz	23、500 kHz
24、PRES <sub>C</sub>	25、1	26、1	27、0	28、0
29、SCLL	30、0xC7	31、0x13	32、0x9	33、0x6
34、T <sub>SCLL</sub>	35、200 * 250 ns = 50 μs	36、20 * 250 ns = 5.0 μs	37、10 * 125 ns = 1250 ns	38、7 * 125 ns = 875 ns
39、SCLH	40、0xC3	41、0xF	42、0x3	43、0x3
44、T <sub>SCLH</sub>	45、196 * 250 ns = 49 μs	46、16 * 250 ns = 4.0 μs	47、4 * 125 ns = 500 ns	48、4 * 125 ns = 500 ns
49、T <sub>SCL</sub> <sup>(1)</sup>	50、~100 μs <sup>(2)</sup>	51、~10 μs <sup>(2)</sup>	52、~2500 ns <sup>(3)</sup>	53、~2000 ns <sup>(4)</sup>
54、SDAD <sub>EL</sub>	55、0x2	56、0x2	57、0x1	58、0x0
59、T <sub>SDADE<sub>L</sub></sub>	60、2 * 250 ns = 500 ns	61、2 * 250 ns = 500 ns	62、1 * 125 ns = 125 ns	63、0 ns
64、SCLD <sub>EL</sub>	65、0x4	66、0x4	67、0x3	68、0x1
69、T <sub>SCLDE<sub>L</sub></sub>	70、5 * 250 ns = 1250 ns	71、5 * 250 ns = 1250 ns	72、4 * 125 ns = 500 ns	73、2 * 125 ns = 250 ns

表 28-2 FI2CCLK = 8 MHz 的时序设置示例

注 1: 由于 SCL 内部检测延迟, SCL 周期 T<sub>SCL</sub> 大于 T<sub>SCLL</sub> + T<sub>SCLH</sub>。为 T<sub>SCL</sub> 提供的值仅为示例。

注 2: T<sub>SYNC1</sub> + T<sub>SYNC2</sub> 最小值为 4 \* T<sub>I2CCLK</sub> = 500 ns。T<sub>SYNC1</sub> + T<sub>SYNC2</sub> = 1000 ns 的示例。

注 3: T<sub>SYNC1</sub> + T<sub>SYNC2</sub> 最小值为 4 \* T<sub>I2CCLK</sub> = 500 ns。T<sub>SYNC1</sub> + T<sub>SYNC2</sub> = 750 ns 的示例。

注 4: T<sub>SYNC1</sub> + T<sub>SYNC2</sub> 最小值为 4 \* T<sub>I2CCLK</sub> = 500 ns。T<sub>SYNC1</sub> + T<sub>SYNC2</sub> = 655 ns 的示例。

74、参数	75、标准模式 (Sm)		76、快速模式 (Fm)	77、极快速模式 (Fm+)
	78、10 kHz	79、100 kHz	80、400 kHz	81、1000 kHz
82、PRESC	83、3	84、3	85、1	86、0
87、SCLL	88、0xC7	89、0x13	90、0x9	91、0x4
92、T <sub>SCLL</sub>	93、200 * 250 ns = 50 μs	94、20 * 250 ns = 5.0 μs	95、10 * 125 ns = 1250 ns	96、6 * 62.5 ns = 312.5 ns
97、SCLH	98、0xC3	99、0xF	100、0x3	101、0x2
102、T <sub>S<sub>CLH</sub></sub>	103、196 * 250 ns = 49 μs	104、16 * 250 ns = 4.0 μs	105、4 * 125 ns = 500 ns	106、3 * 62.5 ns = 187.5 ns
107、T <sub>S<sub>CL</sub></sub> <sup>(1)</sup>	108、~100 μs <sup>(2)</sup>	109、~10 μs <sup>(2)</sup>	110、~2500 ns <sup>(3)</sup>	111、~1000 ns <sup>(4)</sup>
112、S <sub>DADEL</sub>	113、0x2	114、0x2	115、0x2	116、0x0
117、T <sub>S<sub>DADEL</sub></sub>	118、2 * 250 ns = 500 ns	119、2 * 250 ns = 500 ns	120、2 * 125 ns = 250 ns	121、0 ns
122、S <sub>CLDEL</sub>	123、0x4	124、0x4	125、0x3	126、0x2
127、T <sub>S<sub>CLDEL</sub></sub>	128、5 * 250 ns = 1250 ns	129、5 * 250 ns = 1250 ns	130、4 * 125 ns = 500 ns	131、3 * 62.5 ns = 187.5 ns

表 28-3 FI2CCLK = 16 MHz 的时序设置示例



注 1: 由于 SCL 内部检测延迟, SCL 周期  $T_{SCL}$  大于  $T_{SCLL} + T_{SCLH}$ 。为  $T_{SCL}$  提供的值仅为示例。

注 2:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 250 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 1000 \text{ ns}$  的示例。

注 3:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 250 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 750 \text{ ns}$  的示例。

注 4:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 250 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 500 \text{ ns}$  的示例。

132、 参数	133、 标准模式 (Sm)		134、 快速模式 (Fm)	135、 极快速模式 (Fm+)
	136、 10 kHz	137、 100 kHz	138、 400 kHz	139、 1000 kHz
140、 P RESC	141、 0xB	142、 0xB	143、 5	144、 5
145、 S CLL	146、 0xC7	147、 0x13	148、 0x9	149、 0x3
150、 $T_S$ CLL	151、 $200 * 250 \text{ ns} = 50 \mu\text{s}$	152、 $20 * 250 \text{ ns} = 5.0 \mu\text{s}$	153、 $10 * 125 \text{ ns} = 1250 \text{ ns}$	154、 $4 * 125 \text{ ns} = 500 \text{ ns}$
155、 S CLH	156、 0xC3	157、 0xF	158、 0x3	159、 0x1
160、 $T_S$ CLH	161、 $196 * 250 \text{ ns} = 49 \mu\text{s}$	162、 $16 * 250 \text{ ns} = 4.0 \mu\text{s}$	163、 $4 * 125 \text{ ns} = 500 \text{ ns}$	164、 $2 * 125 \text{ ns} = 250 \text{ ns}$
165、 $T_S$ CL <sup>(1)</sup>	166、 $\sim 100 \mu\text{s}$ <sup>(2)</sup>	167、 $\sim 10 \mu\text{s}$ <sup>(2)</sup>	168、 $\sim 2500 \text{ ns}$ <sup>(3)</sup>	169、 $\sim 875 \text{ ns}$ <sup>(4)</sup>
170、 S DADEL	171、 0x2	172、 0x2	173、 0x3	174、 0x0
175、 $T_S$ DADEL	176、 $2 * 250 \text{ ns} = 500 \text{ ns}$	177、 $2 * 250 \text{ ns} = 500 \text{ ns}$	178、 $3 * 125 \text{ ns} = 375 \text{ ns}$	179、 0 ns
180、 S CLDEL	181、 0x4	182、 0x4	183、 0x3	184、 0x1
185、 $T_S$ CLDEL	186、 $5 * 250 \text{ ns} = 1250 \text{ ns}$	187、 $5 * 250 \text{ ns} = 1250 \text{ ns}$	188、 $4 * 125 \text{ ns} = 500 \text{ ns}$	189、 $2 * 125 \text{ ns} = 250 \text{ ns}$

表 28-4 FI2CCLK = 48 MHz 的时序设置示例

注 1: 由于 SCL 内部检测延迟, SCL 周期  $T_{SCL}$  大于  $T_{SCLL} + T_{SCLH}$ 。为  $T_{SCL}$  提供的值仅为示例。

注 2:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 83.3 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 1000 \text{ ns}$  的示例。

注 3:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 83.3 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 750 \text{ ns}$  的示例。

注 4:  $T_{SYNC1} + T_{SYNC2}$  最小值为  $4 * T_{I2CCLK} = 83.3 \text{ ns}$ 。  $T_{SYNC1} + T_{SYNC2} = 250 \text{ ns}$  的示例。

### 28.4.7 SMBus具体功能

系统管理总线（SMBus）是一个双线接口，各种设备可以通过该接口相互通信并与系统的其余部分通信。它基于 I2C 操作原理。SMBus 为系统和电源管理相关任务提供控制总线。

该外设与 SMBUS 规范 rev 2.0 (<http://smbus.org>) 兼容。

系统管理总线规范指的是三种类型的设备。

- ◇ 从机是接收或响应命令的设备。
- ◇ 主设备是发出命令，生成时钟并终止传输的设备。
- ◇ 主控者是专用主机，为系统的 CPU 提供主接口。主控者必须可当作是主机或从机，并且必须支持 SMBus 主机通信协议。

系统中只允许一个主控者。该外设可以配置为主设备或从设备，也可以配置为主控者。

SMBUS 基于 I2C 规范 2.1 版。

#### 总线协议

对于任何给定的设备，有 11 种可能的命令协议。设备可以使用 11 个协议中的任何一个或全部来进行通信。协议包括快速命令，发送字节，接收字节，写入字节，写入字，读取字节，读取字，进程调用，块读取，块写入和块写入块读取进程调用。这些协议应由用户软件实现。有关这些协议的更多详细信息，请参阅 SMBus 规范 2.0 版 (<http://smbus.org>)。

#### 地址解析协议（ARP）

可以通过为每个从设备动态分配新的唯一地址来解决 SMBus 从地址冲突。为了提供隔离每个设备以便进行地址分配的机制，每个设备必须实现唯一的设备标识符（UDID）。这个 128 位数字由软件实现。该外设支持地址解析协议（ARP）。通过将 I2Cx\_CON1 寄存器中的 SMBDEN 位置 1 启用 SMBus 从设备默认地址（0b1100001）。ARP 命令应由用户软件实现。仲裁也在从属模式下执行以支持 ARP。

有关 SMBus 地址解析协议的更多详细信息，请参阅 SMBus 规范 2.0 版 (<http://smbus.org>)。

#### 命令接收和数据应答控制

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。为了在从机模式下允许 ACK 控制，必须通过将 I2Cx\_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。

#### 主控者通知协议

该外设通过将 I2Cx\_CON1 寄存器中的 SMBHEN 位置 1 来支持主控者通知协议。在这种情况下，主控者将应答 SMBus 主控者地址（0b0001 000）。使用此协议时，设备充当主机，主控者充当从机。

#### SMBus 警报

支持 SMBus ALERT 可选信号。仅从设备可以通过它想要通话的 SMBALERT# 引脚向主控者发送信号。主控者处理中断并同时通过警报响应地址（0b0001100）访问所有 SMBALERT# 设备。只有拉低 SMBALERT# 的设备才会应答警报响应地址。当配置为从设备（SMBHEN = 0）时，通过将 I2Cx\_CON1 寄存器中的 ALERTEN 位置 1，可将 SMBA 引脚拉低。警报响应地址同时启用。当配置为主控者（SMBHEN = 1）时，如果在 SMBA 引脚上检测到下降沿且 ALERTEN = 1，则在 I2Cx\_RIF 寄存器中设置 ALERTRI 标志。如果 I2Cx\_IER 寄存器中的 ALERTIE 位置 1，则会产生中断。当 ALERTEN = 0 时，即使外

部 SMBA 引脚为低电平,ALERT 线也会被视为高电平。如果不需要 SMBus ALERT 引脚,若 ALERTEN = 0, 则 SMBA 引脚可用作标准 GPIO。

### 数据包错误检查

SMBus 规范中引入了一种数据包错误检查机制,以提高可靠性和通信稳健性。通过在每次消息传输结束时附加分组错误代码(PEC)来实现分组错误检查。通过在所有消息字节(包括地址和读/写位)上使用  $C(x) = X^8 + X^2 + X + 1$  CRC-8 多项式来计算 PEC。外设嵌入了硬件 PEC 计算器,当接收到的字节与硬件计算的 PEC 不匹配时,允许自动发送非应答。

### 超时

该外设嵌入了硬件定时器,以符合 SMBus 规范 2.0 版中定义的 3 个超时

标记	参数	范围		单位
		最小	最大	
T <sub>TIMEOUT</sub>	检测时钟低超时	25	35	ms
T <sub>LOW:SEXT</sub> <sup>(1)</sup>	累积时钟低延长时间(从设备)	-	25	ms
T <sub>LOW:MEXT</sub> <sup>(2)</sup>	累积时钟低延长时间(主设备)	-	10	ms

表 28-5 SMBus 超时规格

注 1: T<sub>LOW:SEXT</sub> 是允许给定从设备在从初始 START 到 STOP 的一条消息中延长时钟周期的累积时间。另一个从设备或主设备也可能延长时钟,导致组合时钟低延长时间大于 T<sub>LOW:SEXT</sub>。因此,该参数是在从设备作为全速主设备的唯一目标的情况下测量的。

注 2: T<sub>LOW:MEXT</sub> 是允许主设备在从 START 到 ACK, ACK 到 ACK 或 ACK 到 STOP 定义的消息的每个字节内延长其时钟周期的累积时间。从设备或另一个主设备也可能延长时钟,导致组合时钟低电平时间大于给定字节上的 T<sub>LOW:MEXT</sub>。因此,使用全速从设备作为主设备的唯一目标来测量该参数。

### 总线空闲检测

如果总线检测到时钟和数据信号已经持续高电平并且 T<sub>IDLE</sub> 大于 T<sub>HIGH.MAX</sub>, 则主设备可以假设总线是空闲的。

该时序参数涵盖了主机已动态添加到总线并且可能未检测到 SMBCLK 或 SMBDAT 线路上的状态转换的情况。在这种情况下,主设备必须等待足够长的时间以确保当前没有进行传输。外设支持硬件总线空闲检测。

## 28.4.8 SMBus初始化

除了 I2C 初始化之外，还必须进行一些其它特定的初始化以进行 SMBus 通信：

### 接收命令和数据应答控制（从机模式）

SMBus 接收器必须能够 NACK 每个接收到的命令或数据。为了在从机模式下允许 ACK 控制，必须通过将 I2Cx\_CON1 寄存器中的 SBC 位置 1 来启用从机字节控制模式。

### 特定地址（从机模式）

如果需要，应启用特定的 SMBus 地址。

通过将 I2Cx\_CON1 寄存器中的 SMBDEN 位置 1 启用 SMBus 设备从机地址(0b1100 001)。

通过将 I2Cx\_CON1 寄存器中的 SMBHEN 位置 1 启用 SMBus 主控者从机地址 (0b0001 000)。

通过将 I2Cx\_CON1 寄存器中的 ALERTEN 位置 1 启用报警响应地址 (0b0001100)。

### 数据包错误检查

通过将 I2Cx\_CON1 寄存器中的 PECEN 位置 1 启用 PEC 计算。然后在硬件字节计数器的帮助下管理 PEC 传输：I2Cx\_CON2 寄存器中的 NBYTES<7:0>。必须先配置 PECEN 位，然后再启用 I2C。

PEC 传输由硬件字节计数器管理，因此在从机模式下连接 SMBus 时必须设置 SBC 位。在 PECBYTE 位置 1 且 RELOAD 位清零后，在传输 NBYTES-1 数据后传输 PEC。如果设置了 RELOAD，则 PECBYTE 无效。

注意：启用 I2C 时，不允许更改 PECEN 配置。

### 超时检测

通过将 I2Cx\_TIMEOUTR 寄存器中的 TIMEOUTEN 和 TEXTEN 位置 1 来启用超时检测。定时器必须以这样的方式编程，即它们在 SMBus 规范 2.0 版中给出的最大时间之前检测到超时。

#### ◇ T<sub>TIMEOUT</sub> 检查

为了启用 T<sub>TIMEOUT</sub> 检查，必须对 12 位 TIMEOUTA 定时器进行编程，以检查 T<sub>TIMEOUT</sub> 参数。必须将 TIDLE 位配置为“0”才能检测 SCL 低电平超时。然后通过设置 I2Cx\_TIMEOUTR 寄存器中的 TIMEOUTEN 来启用定时器。如果 SCL 在大于 (TIMEOUTA + 1) \* 2048 \* T<sub>I2CCLK</sub> 的时间内被拉低，则在 I2Cx\_RIF 寄存器中设置 TOUTRI 标志。

注意：当 TIMEOUTEN 位置 1 时，不允许更改 TIMEOUTA<11:0>位和 TIDLE 位配置。

#### ◇ T<sub>LOW:SEXT</sub> 和 T<sub>LOW:MEXT</sub> 检查

根据外设是配置为主机还是从机，必须配置 12 位 TIMEOUTB 定时器，以便检查从机的 T<sub>LOW:SEXT</sub> 和主机的 T<sub>LOW:MEXT</sub>。由于标准仅指定最大值，因此用户可以为两者选择相同的值。然后，通过将 I2Cx\_TIMEOUTR 寄存器中的 TEXTEN 位置 1 来启用定时器。

注意：设置 TEXTEN 位时，不允许更改 TIMEOUTB 配置。

### 总线空闲检测

为了启用 TIDLE 检查，必须对 12 位 TIMEOUTA 定时器进行编程，以获得 TIDLE 参数。必须将 TIDLE 位配置为 1 才能检测 SCL 和 SDA 高电平超时。

然后，通过将 I2Cx\_TIMEOUTR 寄存器中的 TIMEOUTEN 位置 1 来启用定时器。如果 SCL 和 SDA 线都保持高电平的时间大于  $(TIMEOUTA + 1) * 4 * T_{I2CCLK}$ ，则在 I2Cx\_RIF 寄存器中设置 TOUTRI 标志。

注意：设置 TIMEOUTEN 时，不允许更改 TIMEOUTA 和 TIDLE 配置。

#### 28.4.9 SMBus: I2Cx\_TIMEOUTR 寄存器配置的例子

◇ 将  $T_{TIMEOUT}$  的最大持续时间配置为 25 ms

$F_{I2CCLK}$	TIMEOUTA<11:0>	TIDLE	TIMEOUTEN	$T_{TIMEOUT}$
8 MHz	0x61	0	1	$98 * 2048 * 125 \text{ ns} = 25 \text{ ms}$
16 MHz	0xC3	0	1	$196 * 2048 * 62.5 \text{ ns} = 25 \text{ ms}$
32 MHz	0x186	0	1	$391 * 2048 * 31.25 \text{ ns} = 25 \text{ ms}$

表 28-6 各种 I2CCLK 频率的 TIMEOUTA 设置示例（最大值  $T_{TIMEOUT} = 25 \text{ ms}$ ）

◇ 将  $T_{LOW:SEXT}$  和  $T_{LOW:MEXT}$  的最大持续时间配置为 8 ms

$F_{I2CCLK}$	TIMEOUTB<11:0>	TIMEOUTEN	$T_{LOW:SEXT}$
8 MHz	0x1F	1	$32 * 2048 * 125 \text{ ns} = 8 \text{ ms}$
16 MHz	0x3F	1	$64 * 2048 * 62.5 \text{ ns} = 8 \text{ ms}$
32 MHz	0x7C	1	$125 * 2048 * 31.25 \text{ ns} = 8 \text{ ms}$

表 28-7 各种 I2CCLK 频率的 TIMEOUTB 设置示例（最大值  $T_{TIMEOUT} = 8 \text{ ms}$ ）

◇ 将  $T_{IDLE}$  的最大持续时间配置为 50 $\mu\text{s}$

$F_{I2CCLK}$	TIMEOUTA<11:0>	TIDLE	TIMEOUTEN	$T_{IDLE}$
8 MHz	0x63	1	1	$100 * 4 * 125 \text{ ns} = 50 \mu\text{s}$
16 MHz	0xC7	1	1	$200 * 4 * 62.5 \text{ ns} = 50 \mu\text{s}$
32 MHz	0x18F	1	1	$400 * 4 * 31.25 \text{ ns} = 50 \mu\text{s}$

表 28-8 各种 I2CCLK 频率的 TIMEOUTA 设置示例（最大 TIDLE = 50 $\mu\text{s}$ ）

## 28.4.10 DMA请求

### 使用 DMA 发送

通过将 I2Cx\_CON1 寄存器中的 TXDMAEN 位置 1, 可以启用 DMA 进行发送。只要 TXTH 位置 1, 就会从使用 DMA 外设置的 SRAM 区域加载数据到 I2Cx\_TXDATA 寄存器。只有数据通过 DMA 发送。

### 使用 DMA 接收

通过将 I2Cx\_CON1 寄存器中的 RXDMAEN 位置 1, 可以启用 DMA 以进行接收。只要 RXTH 位置 1, 就会将数据从 I2Cx\_RXDATA 寄存器加载到使用 DMA 外设置的 SRAM 区域。仅使用 DMA 传输数据 (包括 PEC)。

## 28.4.11 错误情况

以下是可能导致通信失败的错误情况。

### 总线错误 (BERR)

总线错误是指在总线上不在 9 个 SCL 时钟脉冲的倍数之后检测到 START 或 STOP 条件。仅当 I2C 作为主机或寻址从机进行传输时 (不在从机模式下的地址阶段), 才会设置总线错误标志。

如果在从机模式下检测到错误的 START 或 RESTART, 则 I2C 会进入地址识别状态, 就像正确的 START 条件一样。

检测到总线错误时, BERRRI 标志位在 I2Cx\_RIF 寄存器中, 如果 I2Cx\_IER 寄存器中的 BERRIE 位置 1, 则产生中断。

### 仲裁丢失 (ARLO)

仲裁丢失为当在 SDA 线上发送高电平时, 但在 SCL 上升沿上采样到低电平。

- ◇ 在主机模式下, 会在地址阶段, 数据阶段和数据应答阶段检测仲裁丢失。在这种情况下, SDA 和 SCL 线被释放, START 控制位由硬件清零, 主机自动切换到从机模式。
- ◇ 在从机模式下, 会在数据阶段和数据应答阶段检测仲裁丢失。在这种情况下, 传输停止, SCL 和 SDA 线被释放。当检测到仲裁丢失时, ARLORI 标志位在 I2Cx\_RIF 寄存器中, 并且如果在 I2Cx\_IER 寄存器中设置了 ARLOIE 位, 则会产生中断。

### 接收溢出/ 发送欠载错误 (RXOV / TXUD)

当 NOSTRETCH = 1 时, 在从机模式下检测到溢出或欠载错误:

- ◇ 当接收到新字节且接收 FIFO 满时。新接收的字节将会丢失, 并且自动发送 NACK 作为对新字节的应答。
- ◇ 在传输中: 当应发送新字节且尚未写入发送 FIFO 时, 将发送 0xFF。

当检测到接收溢出或发送欠载错误时, TXUD 与 RXOV 标志位在 I2Cx\_STAT 寄存器中, 如果 I2Cx\_IER 寄存器中的 TXUDIE 或 RXOVIE 位置 1, 则会产生中断

## 28. 4. 12 I2C中断

I2C 中的中断由一组六个寄存器控制。

◇ 中断控制 (IER, IDR, IVS)

I2C 中断使能寄存器 (I2Cx\_IER) 通过写 1 来启用中断请求线。同样, I2C 中断禁止寄存器 (I2Cx\_IDR) 通过写 “1” 来禁止中断。IER 和 IDR 是只写寄存器, 上述寄存器的总结果可由中断有效状态寄存器 (I2Cx\_IVS) 显示。IVS 是一个只读寄存器, 使用 “1” 或 “0” 来指示中断请求线是否有效。

◇ 原始中断标志状态读取 (RIF)

I2C 原始中断状态 (I2Cx\_RIF) 是一个只读寄存器, 用于读取模块的中断状态。该寄存器中的位显示 I2C 中断的真实状态。当观察到以下条件时, I2C 可以产生中断:

- SMBus 警报
- 发生超时
- PEC 错误
- 仲裁丢失
- 总线错误
- 传送完成并重新加载
- 传送完成
- 检测检测 STOP
- 收到非应答
- 地址匹配
- 接收 FIFO 不为空
- 发送 FIFO 为空
- 发生接收溢出或发送欠载

◇ 中断标志屏蔽状态 (IFM)

I2C 中断标志屏蔽状态寄存器 (I2Cx\_IFM) 用于读取模块的屏蔽中断状态, 显示屏蔽了哪个中断。IFM 中的每个位是 IVS 和 RIF 中各个位的逻辑 AND。

◇ 中断清除 (ICR)

向该寄存器的位写 “1” 可以清除相应的中断。

I2C 中断可根据通信状态分为事件中断和错误中断两种类型, 用户可参考中断向量分配表的入口地址进行中断服务程序划分和编写。如果产品只为 I2C 分配 1 个中断向量, 则 I2C 所有中断类型须在同一中断服务程序中处理。每个中断源对应的中断类型请参考下图。

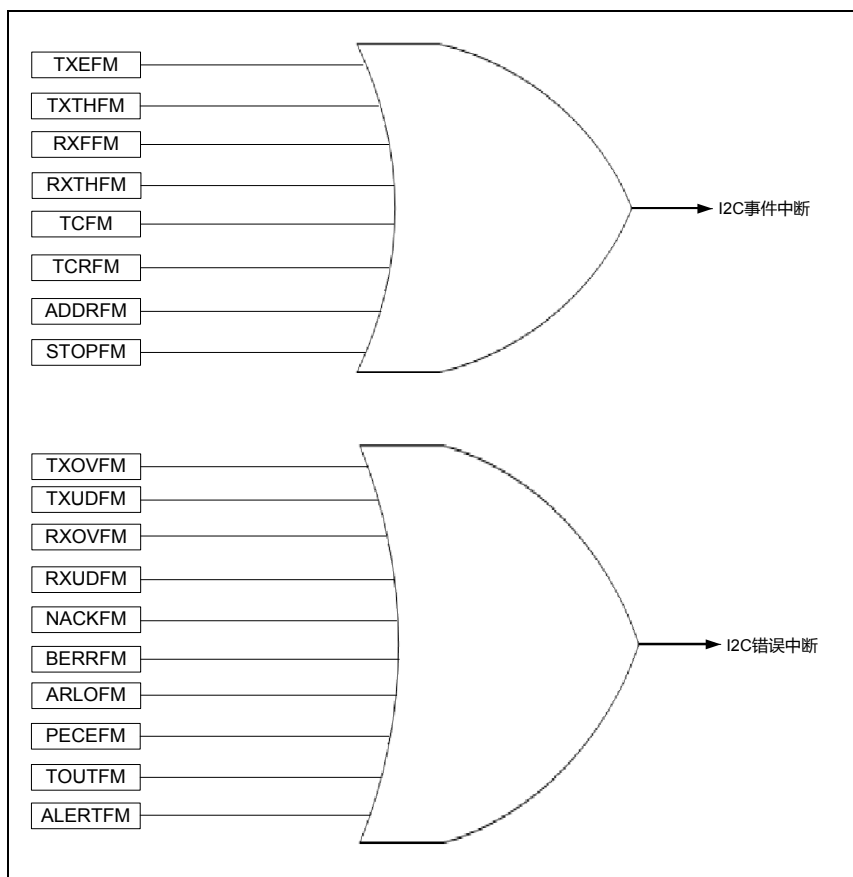


图 28-16 I2C 中断映射图



## 28.5 特殊功能寄存器

### 28.5.1 寄存器列表

I2C 寄存器列表			
寄存器名称	偏移地址	类型	寄存器描述
I2C0 基地址: 4000_8000 <sub>H</sub>			
I2C1 基地址: 4000_8400 <sub>H</sub>			
I2Cx_CON1	0000 <sub>H</sub>	R/W	I2C 控制寄存器 1
I2Cx_CON2	0004 <sub>H</sub>	R/W	I2C 控制寄存器 2
I2Cx_ADDR1	0008 <sub>H</sub>	R/W	I2C 本机地址寄存器 1
I2Cx_ADDR2	000C <sub>H</sub>	R/W	I2C 本机地址寄存器 2
I2Cx_TIMINGR	0010 <sub>H</sub>	R/W	I2C 时序寄存器
I2Cx_TIMEOCTR	0014 <sub>H</sub>	R/W	I2C 超时寄存器
I2Cx_STAT	0018 <sub>H</sub>	R	I2C 状态寄存器
I2Cx_FCON	001C <sub>H</sub>	R/W	I2C FIFO 控制寄存器
I2Cx_PECR	0020 <sub>H</sub>	R	I2C PEC 寄存器
I2Cx_RXDATA	0024 <sub>H</sub>	R	I2C 接收数据寄存器
I2Cx_TXDATA	0028 <sub>H</sub>	W	I2C 发送数据寄存器
I2Cx_IER	002C <sub>H</sub>	W1	I2C 中断启用寄存器
I2Cx_IDR	0030 <sub>H</sub>	W1	I2C 中断禁用寄存器
I2Cx_IVS	0034 <sub>H</sub>	R	I2C 中断有效状态寄存器
I2Cx_RIF	0038 <sub>H</sub>	R	I2C 原始中断标志状态寄存器
I2Cx_IFM	003C <sub>H</sub>	R	I2C 中断标志屏蔽状态寄存器
I2Cx_ICR	0040 <sub>H</sub>	C_W1	I2C 中断清除寄存器

### 28.5.2 寄存器描述

#### 28.5.2.1 I2C控制寄存器 1 (I2Cx\_CON1)

I2C 控制寄存器 1 (I2Cx_CON1)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	Reserved	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Reserved	DNF					Reserved					PE		

Reserved	Bit 31-24	—	保留
PECEN	Bit 23	R/W	PEC 使能 0: 禁止 PEC 计算

			<p>1: 使能 PEC 计算</p> <p>注意: 如果不支持 SMBus 功能, 则该位保留并由硬件强制为“0”。</p>
ALERTEN	Bit 22	R/W	<p><b>SMBus 警报使能</b></p> <p>设备模式 (SMBHEN = 0):</p> <p>0: 释放 SMBA 引脚为高电平且禁止警报响应地址标头: 0001100x 并回应 NACK。</p> <p>1: 将 SMBA 引脚驱动为低电平且使能警报响应地址标头: 0001100x 并回应 ACK。</p> <p>主控者模式 (SMBHEN = 1):</p> <p>0: 不支持 SMBus 警报引脚 (SMBA)。</p> <p>1: 支持 SMBus 警报引脚 (SMBA)。</p> <p>注: 当 ALERTEN = 0 时, SMBA 引脚可用作标准 GPIO。如果不支持 SMBus 功能, 则该位保留并由硬件强制为“0”。</p>
SMBDEN	Bit 21	R/W	<p><b>SMBus 设备从机地址使能</b></p> <p>0: 禁止设备从机地址。地址 0b1100001x 无应答。</p> <p>1: 使能设备从机地址。地址 0b1100001x 被应答。</p> <p>注意: 如果不支持 SMBus 功能, 则该位保留并由硬件强制为“0”。</p>
SMBHEN	Bit 20	R/W	<p><b>SMBus 主控者地址使能</b></p> <p>0: 禁止主控者地址。地址 0b0001000x 无应答。</p> <p>1: 使能主控者地址。地址 0b0001000x 被应答。</p> <p>注意: 如果不支持 SMBus 功能, 则该位保留并由硬件强制为“0”。</p>
GCEN	Bit 19	R/W	<p><b>广播呼叫使能</b></p> <p>0: 禁止广播呼叫。地址 0b00000000 无应答。</p> <p>1: 使能广播呼叫。地址 0b00000000 被应答。</p>
Reserved	Bit 18	—	保留
NOSTRETCH	Bit 17	R/W	<p><b>时钟延长禁止</b></p> <p>该位用于禁止从机模式下的时钟延长。必须在主机模式下保持清除状态。</p> <p>0: 使能时钟延长</p> <p>1: 禁止时钟延长</p> <p>注: 只有在 I2C 被禁止 (PE = 0) 时才能对该位进行编程。</p>
SBC	Bit 16	R/W	<p><b>从机字节控制</b></p> <p>该位用于在从机模式下使能硬件字节控制。</p>

			0: 禁止从机字节控制 1: 使能从机字节控制
RXDMAEN	Bit 15	R/W	<b>使能 DMA 接收请求</b> 0: 禁止 DMA 模式进行接收 1: 使能 DMA 模式进行接收
TXDMAEN	Bit 14	R/W	<b>使能 DMA 发送请求</b> 0: 禁止 DMA 模式进行发送 1: 使能 DMA 模式进行发送
Reserved	Bit 13-12	—	保留
DNF	Bit 11-8	R/W	<b>数字噪声滤波器</b> 这些位用于配置 SDA 和 SCL 输入上的数字噪声滤波器。数字滤波器将过滤长度高达 $DNF<3:0> * T_{I2CCLK}$ 的尖峰 0000: 禁止数字滤波器 0001: 使能数字滤波器, 滤波能力高达 $1 T_{I2CCLK}$ ... 1111: 使能数字滤波器, 滤波能力高达 $15 T_{I2CCLK}$ 注意: 只有在禁止 I2C (PE = 0) 时才能对该滤波器进行编程。
Reserved	Bit 7-1	—	保留
PE	Bit 0	R/W	<b>I2C 使能</b> 0: I2C 禁止 1: I2C 使能 注: 当 PE = 0 时, I2C SCL 和 SDA 线被释放。内部状态机和状态位将恢复为其复位值。清零后, PE 必须保持低电平至少 3 个 APB 时钟周期。

### 28.5.2.2 I2C控制寄存器 2 (I2Cx\_CON2)

I2C 控制寄存器 2 (I2Cx_CON2)																																			
偏移地址: 04 <sub>H</sub>																																			
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved	ACK_UPD	HOLDACK	Reserved	PECBYTE	AUTOEND	RELOAD	NBYTES									NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD													

Reserved	Bit 31-30	—	保留
ACK_UPD	Bit 29	R/C_W1	<b>ACK 更新</b> 0: 不更新 ACK 状态。 1: 更新 ACK 状态并释放时钟。
HOLDACK	Bit 28	R/W	<b>等待回应 ACK</b> 使能时, 第 8 和第 9 个 SCL 脉冲之间会将 SCL 延长, 并等待用户设置 ACK / NACK。在设置 ACK / NACK 之后, 设置 ACK_UPD 以发出 ACK / NACK 并释放 SCL。 0: 自动应答 ACK。 1: 使能手动回应 ACK。
Reserved	Bit 27	—	保留
PECBYTE	Bit 26	R/C_W1	<b>数据包错误检查字节</b> 该位由软件置 1, 当传输 PEC 时, 或者当接收到 STOP 条件或匹配的地址时, 由硬件清零, 当 PE = 0 时也是如此。 0: 没有 PEC 传输。 1: 请求 PEC 发送/接收 注意: 向该位写“0”无效。 设置 RELOAD 时, 该位无效。 当 SBC = 0 时, 该位对从机模式无效。 如果不支持 SMBus 功能, 则该位保留并由硬件强制为“0”。
AUTOEND	Bit 25	R/W	<b>自动结束模式 (主机模式)</b> 该位由软件置位和清除。 0: 软件结束模式: 发送完 NBYTES 数据时设置 TC 标志, 将 SCL 拉低。 1: 自动结束模式: 发送完 NBYTES 数据时自动发送 STOP 条件。

			注：该位在从机模式或 RELOAD 位置 1 时无效。
RELOAD	Bit 24	R/W	<p><b>NBYTES 重新加载模式</b></p> <p>该位由软件置位和清除。</p> <p>0：在 NBYTES 数据传输之后完成传输（接下来是 STOP 或 RESTART）。</p> <p>1：NBYTES 数据传输后将继续传输（NBYTES 将重新加载）。发送完 NBYTES 数据后设置 TCR 标志，将 SCL 拉低。</p>
NBYTES	Bit 23-16	R/W	<p><b>字节数</b></p> <p>在此编程要发送/接收的字节数。</p> <p>在 SBC = 0 的从机模式下，该字段无关紧要。</p> <p>注：不允许在设置 START 位时更改这些位。</p>
NACK	Bit 15	R/C_W1	<p><b>NACK 生成（从机模式）</b></p> <p>该位由软件置位，在发送 NACK 时由硬件清零，或者在接收到 STOP 条件或地址匹配时，或者当 PE = 0 时由硬件清零。</p> <p>0：在下一个接收的字节发送 ACK。</p> <p>1：在下一个接收的字节发送 NACK。</p> <p>注意：向该位写“0”无效。</p> <p>该位仅用于从机模式：在主机接收模式下，无论 NACK 位值如何，在 STOP 或 RESTART 条件之前的最后一个字节之后自动生成 NACK。当从机接收 NOSTRETCH 模式发生溢出时，无论 NACK 位值如何，都会自动生成 NACK。当使能硬件 PEC 检查（PECBYTE = 1）时，PEC 确认值不依赖于 NACK 值。</p>
STOP	Bit 14	R/C_W1	<p><b>STOP 生成（主机模式）</b></p> <p>该位由软件置位，当检测到停止条件时，或者当 PE = 0 时由硬件清零。</p> <p>在主模式下：</p> <p>0：无 STOP 生成。</p> <p>1：当前字节传输后 STOP 生成。</p> <p>注意：向该位写“0”无效。</p>
START	Bit 13	R/C_W1	<p><b>START 生成</b></p> <p>该位由软件置位，并在启动后跟随地址序列发送，仲裁丢失，超时错误检测或 PE = 0 时由硬件清零。</p> <p>0：无 START 生成。</p> <p>1：RESTART / START 生成：</p>

			<p>- 如果 I2C 已经处于主模式且 AUTOEND = 0, 则在 NBYTES 传输结束后, 当 RELOAD = 0 时, 将该位置 1 会产生重复启动条件。</p> <p>- 否则, 一旦总线空闲, 将该位置 1 产生 START 条件。</p> <p>注意: 向该位写“0”无效。</p> <p>即使总线忙, 也可以设置 START 位。</p> <p>在 10 位寻址模式下, 如果在地址的第一部分接收到 NACK, 则 START 位不会被硬件清零, 主机将重新发送地址序列, 除非 START 位被软件清零</p>
HEAD10R	Bit 12	R/W	<p><b>10 位地址标头仅读取方向 (主机接收模式)</b></p> <p>0: 主机发送完整的 10 位从机地址读取序列: 在写入方向上启动+ 2 字节 10 位地址+ 在读取方向上重新启动+ 10 位地址的前 7 位, 然后发送读取方向。。</p> <p>1: 主机仅发送 10 位地址的前 7 位, 然后发送读取方向。</p> <p>注: 不允许在设置 START 位时更改该位。</p>
ADD10	Bit 11	R/W	<p><b>10 位寻址模式 (主机模式)</b></p> <p>0: 主机工作在 7 位寻址模式</p> <p>1: 主机工作在 10 位寻址模式</p> <p>注: 不允许在设置 START 位时更改该位。</p>
RD_WRN	Bit 10	R/W	<p><b>传输方向 (主机模式)</b></p> <p>0: 主机请求写入传输。</p> <p>1: 主机请求读取传输。</p> <p>注: 不允许在设置 START 位时更改该位。</p>
SADD	Bit 9-0	R/W	<p><b>从机地址位 0 (主机模式)</b></p> <p>在 7 位寻址模式下 (ADD10 = 0): 此位不在乎</p> <p>在 10 位寻址模式下 (ADD10 = 1): 该位应写入要发送的从地址的第 0 位</p> <p><b>从机地址位 7: 1 (主机模式)</b></p> <p>在 7 位寻址模式下 (ADD10 = 0): 应使用要发送的 7 位从地址写入这些位</p> <p>在 10 位寻址模式下 (ADD10 = 1): 应使用要发送的从地址的位 7: 1 写入这些位。</p> <p><b>从机地址位 9: 8 (主机模式)</b></p> <p>在 7 位寻址模式下 (ADD10 = 0): 这些位都不在乎</p>

			在 10 位寻址模式下 (ADD10 = 1): 应使用要发送的从地址的第 9: 8 位写入这些位 注: 不允许在设置 START 位时更改这些位。
--	--	--	--

### 28.5.2.3 I2C本机地址寄存器 1 (I2Cx\_ADDR1)

I2C 本机地址寄存器 1 (I2Cx_ADDR1)																																	
偏移地址: 08 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved																OA1EN	Reserved						OA1MODE	OA1									

Reserved	Bit 31-16	—	保留
OA1EN	Bit 15	R/W	<b>本机地址 1 使能</b> 0: 禁止本机地址 1。接收到的从机地址 OA1 无应答。 1: 使能本机地址 1。接收到的从机地址 OA1 被应答。
Reserved	Bit 14-11	—	保留
OA1MODE	Bit 10	R/W	<b>本机地址 1, 10 位地址模式使能</b> 0: 本机地址 1 是 7 位地址。 1: 本机地址 1 是 10 位地址。 注: 仅当 OA1EN = 0 时才能写入该位。
OA1	Bit 9-0	R/W	<b>OA1&lt;0&gt;: 本机地址 1</b> 7 位寻址模式: 不在乎 10 位寻址模式: 地址的第 0 位 <b>OA1&lt;7:1&gt;: 本机地址 1</b> 7 位寻址模式: 7 位地址 10 位寻址模式: 10 位地址的 7: 1 位 <b>OA1&lt;9:8&gt;: 本机地址 1</b> 7 位寻址模式: 不在乎 10 位寻址模式: 地址的 9: 8 位 注: 仅当 OA1EN = 0 时才能写入该位。

### 28.5.2.4 I2C本机地址寄存器 2 (I2Cx\_ADDR2)

I2C 本机地址寄存器 2 (I2Cx_ADDR2)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															OA2EN	Reserved					OA2MSK	OA2							Reserved		

Reserved	Bit 31-16	—	保留
OA2EN	Bit 15	R/W	<b>本机地址 2 使能</b> 0: 禁止本机地址 2。接收到的从机地址 OA2 无应答。 1: 使能本机地址 2。接收到的从机地址 OA2 被应答。
Reserved	Bit 14-11	—	保留
OA2MSK	Bit 10-8	R/W	<b>本机地址 2 屏蔽</b> 000: 没有屏蔽 001: OA2<0>被屏蔽并且忽略。仅比较 OA2<6:1>。 010: OA2<1:0>被屏蔽并且忽略。仅比较 OA2<6:2>。 011: OA2<2:0>被屏蔽并且忽略。仅比较 OA2<6:3>。 100: OA2<3:0>被屏蔽并且忽略。仅比较 OA2<6:4>。 101: OA2<4:0>被屏蔽并且忽略。仅比较 OA2<6:5>。 110: OA2<5:0>被屏蔽并且忽略。仅比较 OA2<6>。 111: OA2<6:0>被屏蔽并且忽略。不进行比较, 并且应答所有 (除了保留的) 7 位接收地址。 注意: 仅当 OA2EN = 0 时才能写入这些位。 一旦 OA2MSK 不等于 0, 则 I2C 即使比较匹配, 也不会应答保留地址 (0b0000xxx 和 0b1111xxx)。
OA2	Bit 7-1	R/W	<b>本机地址 2</b> 7 位寻址模式: 7 位地址 注意: 仅当 OA2EN = 0 时才能写入这些位。
Reserved	Bit 0	—	保留



28.5.2.5 I2C时钟寄存器 (I2Cx\_TIMINGR)

I2C 时序寄存器 (I2Cx_TIMINGR)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESC				Reserved				SCLDEL				SDADEL				SCLH								SCLL							

PRESC	Bit 31-28	R/W	<p><b>时钟预分频器</b></p> <p>该字段用于预分频 I2CCLK，以产生用于数据设置和保持计数器以及 SCL 高电平和低电平计数器的时钟周期 T<sub>PRESC</sub>。</p> $T_{PRESC} = (PRESC + 1) * T_{I2CCLK}$ <p>禁止 I2C (PE = 0) 时，必须配置该寄存器。</p>
Reserved	Bit 27-24	—	保留
SCLDEL	Bit 23-20	R/W	<p><b>数据设置时间</b></p> <p>该字段用于在 SDA 边沿和 SCL 上升沿之间生成延迟 T<sub>SCLDEL</sub>。在主机和从机模式下，NOSTRETCH = 0 时，SCL 线在 T<sub>SCLDEL</sub> 期间拉低。</p> $T_{SCLDEL} = (SCLDEL + 1) * T_{PRESC}$ <p>注意：T<sub>SCLDEL</sub> 用于生成 T<sub>SU: DAT</sub> 时序。</p> <p>禁止 I2C (PE = 0) 时，必须配置该寄存器。</p>
SDADEL	Bit 19-16	R/W	<p><b>数据保持时间</b></p> <p>该字段用于在 SCL 下降沿和 SDA 边沿之间生成延迟 T<sub>SDADEL</sub>。在主机和从机模式下，NOSTRETCH = 0 时，SCL 线在 T<sub>SDADEL</sub> 期间拉低。</p> $T_{SDADEL} = SDADEL * T_{PRESC}$ <p>注意：SDADEL 用于生成 T<sub>HD: DAT</sub> 时序。</p> <p>禁止 I2C (PE = 0) 时，必须配置该寄存器。</p>
SCLH	Bit 15-8	R/W	<p><b>SCL 高电平期间 (主机模式)</b></p> <p>该字段用于在主机模式下生成 SCL 高电平周期。</p> $T_{SCLH} = (SCLH + 1) * T_{PRESC}$ <p>注意：SCLH 还用于生成 T<sub>SU: STO</sub> 和 T<sub>HD: STA</sub> 时序。</p> <p>禁止 I2C (PE = 0) 时，必须配置该寄存器。</p>
SCLL	Bit 7-0	R/W	<p><b>SCL 低电平周期 (主机模式)</b></p> <p>该字段用于在主机模式下生成 SCL 低电平周期。</p> $T_{SCLL} = (SCLL + 1) * T_{PRESC}$

			注意：SCLL 还用于生成 $T_{BUF}$ 和 $T_{SU: STA}$ 时序。 禁止 I2C (PE = 0) 时，必须配置该寄存器。
--	--	--	--

28. 5. 2. 6 I2C 超时寄存器 (I2Cx\_TIMEOUTR)

I2C 超时寄存器 (I2Cx_TIMEOUTR)																																	
偏移地址: 14 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
TEXTEN		Reserved			TIMEOUTB												TIMEOUTEN		Reserved			TIDLE	TIMEOUTA										

TEXTEN	Bit 31	R/W	<p><b>累积时钟延长超时使能</b></p> <p>0: 禁止累积时钟延长超时检测</p> <p>1: 使能累积时钟延长超时检测。</p> <p>当 I2C 接口完成累积 SCL 延长超过 <math>T_{LOW:EXT}</math> 时, 会检测到超时错误 (TOUTRI = 1)。</p>
Reserved	Bit 30-28	—	保留
TIMEOUTB	Bit 27-16	R/W	<p><b>总线超时 B</b></p> <p>该字段用于配置累积时钟延长超时:</p> <p>在主机模式下, 检测到主机累积时钟低延长时间 (<math>T_{LOW:MEXT}</math>)。</p> <p>在从机模式下, 检测到从机累积时钟低延长时间 (<math>T_{LOW:SEXT}</math>)。</p> <p><math>T_{LOW:EXT} = (TIMEOUTB + 1) * 2048 * T_{I2CCCLK}</math></p> <p>注意: 仅当 TEXTEN = 0 时才能写入这些位。</p>
TIMEOUTEN	Bit 15	R/W	<p><b>时钟超时使能</b></p> <p>0: 禁止 SCL 超时检测</p> <p>1: 使能 SCL 超时检测: 当 SCL 为低电平超过 <math>T_{TIMEOUT}</math> (TIDLE = 0) 或高电平超过 <math>T_{IDLE}</math> (TIDLE = 1) 时, 检测到超时错误 (TOUTRI = 1)。</p>
Reserved	Bit 14-13	—	保留
TIDLE	Bit 12	R/W	<p><b>空闲时钟超时检测</b></p> <p>0: TIMEOUTA 用于检测 SCL 低超时</p> <p>1: TIMEOUTA 用于检测 SCL 和 SDA 高超时 (总线空闲状态)</p> <p>注: 仅当 TIMEOUTEN = 0 时才能写入该位。</p>
TIMEOUTA	Bit 11-0	R/W	<p><b>总线超时 A</b></p> <p>该字段用于配置:</p> <ul style="list-style-type: none"> <li>- 当 TIDLE = 0 时, SCL 低超时条件 <math>T_{TIMEOUT}</math></li> </ul>

		<p> <math>T_{\text{TIMEOUT}} = (\text{TIMEOUTA} + 1) * 2048 * T_{\text{I2CCLK}}</math>                      - 当 TIDLE = 1 时，总线空闲状态（SCL 和 SDA 均为高电平）  <math>T_{\text{IDLE}} = (\text{TIMEOUTA} + 1) * 4 * T_{\text{I2CCLK}}</math>                      注意：仅当 TIMEOUTEN = 0 时才能写入这些位。                 </p>
--	--	---

### 28.5.2.7 I2C状态寄存器 (I2Cx\_STAT)

I2C 状态寄存器 (I2Cx_STAT)																																	
偏移地址: 18 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00110001 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved								ADDPCODE								DIR	BUSY	Reserved				TCR	TC	RXTH	RXUD	RXOV	RXF	RXE	TXTH	TXUD	TXOV	TXF	TXE

Reserved	Bit 31-24	—	保留
ADDPCODE	Bit 23-17	R	<b>地址匹配代码 (从机模式)</b> 当发生地址匹配事件 (ADDR = 1) 时, 使用接收的地址更新这些位。在 10 位地址的情况下, ADDPCODE 提供 10 位标头与地址的 2 个 MSB。
DIR	Bit 16	R	<b>传输方向 (从机模式)</b> 发生地址匹配事件 (ADDRRI = 1) 时更新此标志。 0: 写入传输, 从机进入接收模式。 1: 读取传输, 从机进入发送模式。
BUSY	Bit 15	R	<b>总线忙</b> 该标志表示总线上正在进行通信。检测到 START 条件时由硬件设置。当检测到停止条件或 PE = 0 时, 它由硬件清零。
Reserved	Bit 14-12	—	保留
TCR	Bit 11	R	<b>传输完成并重新加载</b> 当 RELOAD = 1 且已将 NBYTES 数据传输完成时, 此标志由硬件设置。当 NBYTES 写入非零值时, 它由硬件清零。 注: 当 PE = 0 时, 该位由硬件清零。 该标志仅用于主模式, 或者用于从模式 SBC 位置 1 时。
TC	Bit 10	R	<b>传输完成 (主机模式)</b> 当 RELOAD = 0, AUTOEND = 0 且已将 NBYTES 数据传输完成时, 该标志由硬件设置。当 START 位或 STOP 位置 1 时, 由硬件清零。 注: 当 PE = 0 时, 该位由硬件清零。
RXTH	Bit 9	R	<b>接收 FIFO 水平超出阈值</b> 该位提供接收 FIFO 水平超过阈值。

			0: 接收 FIFO 水平没有超过阈值。 1: 接收 FIFO 水平超过阈值。
RXUD	Bit 8	R	<b>接收 FIFO 下溢</b> 0: 接收 FIFO 没有下溢 1: 接收 FIFO 下溢
RXOV	Bit 7	R	<b>接收 FIFO 溢出</b> 0: 接收 FIFO 没有溢出 1: 接收 FIFO 溢出
RXF	Bit 6	R	<b>接收 FIFO 满</b> 0: 接收 FIFO 没有满 1: 接收 FIFO 满
RXE	Bit 5	R	<b>接收 FIFO 空</b> 0: 接收 FIFO 没有空 1: 接收 FIFO 空
TXTH	Bit 4	R	<b>发送 FIFO 水平低于阈值</b> 该位提供发送 FIFO 水平低于阈值。 0: 发送 FIFO 水平高于阈值。 1: 发送 FIFO 水平低过阈值。
TXUD	Bit 3	R	<b>发送 FIFO 下溢</b> 0: 发送 FIFO 没有下溢 1: 发送 FIFO 下溢
TXOV	Bit 2	R	<b>发送 FIFO 溢出</b> 0: 发送 FIFO 没有溢出 1: 发送 FIFO 溢出
TXF	Bit 1	R	<b>发送 FIFO 满</b> 0: 发送 FIFO 没有满 1: 发送 FIFO 满
TXE	Bit 0	R	<b>发送 FIFO 空</b> 0: 发送 FIFO 没有空 1: 发送 FIFO 空

28. 5. 2. 8 I2C FIFO控制寄存器 (I2Cx\_FCON)

I2CFIFO 控制寄存器 (I2Cx_FCON)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXFTH	RXFRST	RXFLV						TXFTH	TXFRST	TXFLV					

Reserved	Bit 31-16	—	保留
RXFTH	Bit 15-14	R/W	<p><b>接收 FIFO 触发阈值</b> 这用于选择接收 FIFO 生成接收 FIFO 触发阈值标志的阈值电平。</p> <p>00: 当接收 FIFO 中有 1 个字符时触发 01: 当接收 FIFO 中有 4 个字符时触发 10: 当接收 FIFO 中有 8 个字符时触发 11: 当接收 FIFO 中有 14 个字符时触发</p>
RXFRST	Bit 13	C_W1	<p><b>接收 FIFO 复位</b> 当 FIFO 置 1 时, 接收 FIFO 中的所有字节都被清除, 并将 FIFO 视为空。请注意, 该位将在下一个时钟周期内返回 0。</p> <p>0: 没有 FIFO 接收复位 1: 重置接收指针 该位由硬件自动清除。</p>
RXFLV	Bit 12-8	R	<p><b>接收 FIFO 电平</b> 该位由硬件设置。这用于指示接收 FIFO 中的数据条目数。 它从 0 到 16。</p>
TXFTH	Bit 7-6	R/W	<p><b>发送 FIFO 触发阈值</b> 这用于选择发送 FIFO 产生发送 FIFO 触发阈值标志的阈值电平。</p> <p>00: 当发送 FIFO 空时触发 01: 当发送 FIFO 中有 2 个字符时触发 10: 当发送 FIFO 中有 4 个字符时触发 11: 当发送 FIFO 中有 8 个字符时触发</p>
TXFRST	Bit 5	C_W1	<p><b>发送 FIFO 复位</b> 当 FIFO 置 1 时, 发送 FIFO 中的所有字节都被清除, 并将 FIFO 视为空。请注意, 该位将在下一个时钟周期内返回 0。</p> <p>0: 无 FIFO 发送复位</p>

			1: 重置发送指针 该位由硬件自动清除。
TXFLV	Bit 4-0	R	<b>发送 FIFO 电平</b> 该位由硬件设置。这用于指示发送 FIFO 中的数据条目数。 它从 0 到 16。

### 28.5.2.9 I2C PEC寄存器 (I2Cx\_PECR)

I2CPEC 寄存器 (I2Cx_PECR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PEC							

Reserved	Bit 31-8	—	保留
PEC	Bit 7-0	R	<b>数据包错误检查寄存器</b> 当 PECEN = 1 时, 该字段包含内部 PEC。 当 PECEN = 0 时, PEC 由硬件清零。

### 28.5.2.10 I2C接收数据寄存器 (I2Cx\_RXDATA)

I2C 接收数据寄存器 (I2Cx_RXDATA)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								RXDATA							

Reserved	Bit 31-8	—	保留
RXDATA	Bit 7-0	R	<b>8 位接收数据</b> 从 I2C 总线接收的数据字节。



### 28. 5. 2. 11 I2C发送数据寄存器 (I2Cx\_TXDATA)

I2C 发送数据寄存器 (I2Cx_TXDATA)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TXDATA							

Reserved	Bit 31-8	—	保留
TXDATA	Bit 7-0	W	<b>8 位发送数据</b> 要传输到 I2C 总线的数据字节。

28. 5. 2. 12 I2C中断使能寄存器 (I2Cx\_IER)

I2C 中断使能寄存器 (I2Cx_IER)																																										
偏移地址: 01C <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												ALERTIE	TOUTIE	PECEIE	ARLOIE	BERRIE	Reserved	STOPIE	NACKIE	ADDRIE	TCRIE	TCIE	RXTHIE	RXUDIE	RXOVIE	RXFIE	Reserved	TXTHIE	TXUDIE	TXOVIE	Reserved	TXEIE										

Reserved	Bit 31-21	—	保留
ALERTIE	Bit 20	W1	<b>SMBus 报警中断使能</b> 0: 无效 1: 使能中断
TOUTIE	Bit 19	W1	<b>超时中断使能</b> 0: 无效 1: 使能中断
PECEIE	Bit 18	W1	<b>PEC 错误中断使能</b> 0: 无效 1: 使能中断
ARLOIE	Bit 17	W1	<b>仲裁丢失中断使能</b> 0: 无效 1: 使能中断
BERRIE	Bit 16	W1	<b>总线错误中断使能</b> 0: 无效 1: 使能中断
Reserved	Bit 15	—	保留
STOPIE	Bit 14	W1	<b>停止检测中断使能</b> 0: 无效 1: 使能中断
NACKIE	Bit 13	W1	<b>NACK 接收中断使能</b> 0: 无效 1: 使能中断
ADDRIE	Bit 12	W1	<b>地址匹配中断使能</b> 0: 无效 1: 使能中断
TCRIE	Bit 11	W1	<b>传输完成并重新加载中断使能</b> 0: 无效 1: 使能中断
TCIE	Bit 10	W1	<b>传输完成中断使能</b> 0: 无效 1: 使能中断

RXTHIE	Bit 9	W1	接收 FIFO 超过阈值中断使能 0: 无效 1: 使能中断
RXUDIE	Bit 8	W1	接收 FIFO 欠载中断使能 0: 无效 1: 使能中断
RXOVIE	Bit 7	W1	接收 FIFO 溢出中断使能 0: 无效 1: 使能中断
RXFIE	Bit 6	W1	接收 FIFO 满中断使能 0: 无效 1: 使能中断
Reserved	Bit 5	—	保留
TXTHIE	Bit 4	W1	发送 FIFO 低于阈值中断使能 0: 无效 1: 使能中断
TXUDIE	Bit 3	W1	发送 FIFO 欠载中断使能 0: 无效 1: 使能中断
TXOVIE	Bit 2	W1	发送 FIFO 溢出中断使能 0: 无效 1: 使能中断
Reserved	Bit 1	—	保留
TXEIE	Bit 0	W1	发送 FIFO 空中断使能 0: 无效 1: 使能中断

### 28.5.2.13 I2C中断禁止寄存器 (I2Cx\_IDR)

I2C 中断禁止寄存器 (I2Cx_IDR)																																										
偏移地址: 30 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												ALERTID	TOUTID	PECEID	ARLOID	BERRID	Reserved	STOPID	NACKID	ADDRID	TCRID	TCID	RXTHID	RXUDID	RXOVID	RXFID	Reserved	TXTHID	TXUDID	TXOVID	Reserved	TXEID										

Reserved	Bit 31-21	—	保留
ALERTID	Bit 20	W1	<b>SMBus 警报中断禁止</b> 0: 无效 1: 禁止中断
TOUTID	Bit 19	W1	<b>超时中断禁止</b> 0: 无效 1: 禁止中断
PECEID	Bit 18	W1	<b>PEC 错误中断禁止</b> 0: 无效 1: 禁止中断
ARLOID	Bit 17	W1	<b>仲裁丢失中断禁止</b> 0: 无效 1: 禁止中断
BERRID	Bit 16	W1	<b>总线错误中断禁止</b> 0: 无效 1: 禁止中断
Reserved	Bit 15	—	保留
STOPID	Bit 14	W1	<b>停止检测中断禁止</b> 0: 无效 1: 禁止中断
NACKID	Bit 13	W1	<b>NACK 接收中断禁止</b> 0: 无效 1: 禁止中断
ADDRID	Bit 12	W1	<b>地址匹配中断禁止</b> 0: 无效 1: 禁止中断
TCRID	Bit 11	W1	<b>传输完成并重新加载中断禁止</b> 0: 无效 1: 禁止中断
TCID	Bit 10	W1	<b>传输完成中断禁止</b> 0: 无效 1: 禁止中断
RXTHID	Bit 9	W1	<b>接收 FIFO 超过阈值中断禁止</b>

			0: 无效 1: 禁止中断
RXUDID	Bit 8	W1	接收 FIFO 欠载中断禁止 0: 无效 1: 禁止中断
RXOVID	Bit 7	W1	接收 FIFO 溢出中断禁止 0: 无效 1: 禁止中断
RXFID	Bit 6	W1	接收 FIFO 满中断禁止 0: 无效 1: 禁止中断
Reserved	Bit 5	—	保留
TXTHID	Bit 4	W1	发送 FIFO 低于阈值中断禁止 0: 无效 1: 禁止中断
TXUDID	Bit 3	W1	发送 FIFO 欠载中断禁止 0: 无效 1: 禁止中断
TXOVID	Bit 2	W1	发送 FIFO 溢出中断禁止 0: 无效 1: 禁止中断
Reserved	Bit 1	—	保留
TXEID	Bit 0	W1	发送 FIFO 空中断禁止 0: 无效 1: 禁止中断

28. 5. 2. 14 I2C中断有效状态寄存器 (I2Cx\_IVS)

I2C 中断有效状态寄存器 (I2Cx_IVS)																																										
偏移地址: 34 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												ALERTIV	TOUTIV	PECEIV	ARLOIV	BERRIV	Reserved	STOPIV	NACKIV	ADDRIV	TCRIV	TCIV	RXTHIV	RXUDIV	RXOVIV	RXFIV	Reserved	TXTHIV	TXUDIV	TXOVIV	Reserved	TXEIV										

Reserved	Bit 31-21	—	保留
ALERTIV	Bit 20	R	<b>SMBus 警报中断有效</b> 0: 无效 1: 中断有效
TOUTIV	Bit 19	R	<b>超时中断有效</b> 0: 无效 1: 中断有效
PECEIV	Bit 18	R	<b>PEC 错误中断有效</b> 0: 无效 1: 中断有效
ARLOIV	Bit 17	R	<b>仲裁丢失中断有效</b> 0: 无效 1: 中断有效
BERRIV	Bit 16	R	<b>总线错误中断有效</b> 0: 无效 1: 中断有效
Reserved	Bit 15	—	保留
STOPIV	Bit 14	R	<b>停止检测中断有效</b> 0: 无效 1: 中断有效
NACKIV	Bit 13	R	<b>NACK 接收中断有效</b> 0: 无效 1: 中断有效
ADDRIV	Bit 12	R	<b>地址匹配中断有效</b> 0: 无效 1: 中断有效
TCRIV	Bit 11	R	<b>传输完成并重新加载中断有效</b> 0: 无效 1: 中断有效
TCIV	Bit 10	R	<b>传输完成中断有效</b> 0: 无效 1: 中断有效

RXTHIV	Bit 9	R	接收 FIFO 超过阈值中断有效 0: 无效 1: 中断有效
RXUDIV	Bit 8	R	接收 FIFO 欠载中断有效 0: 无效 1: 中断有效
RXOVIV	Bit 7	R	接收 FIFO 溢出中断有效 0: 无效 1: 中断有效
RXFIV	Bit 6	R	接收 FIFO 满中断有效 0: 无效 1: 中断有效
Reserved	Bit 5	—	保留
TXTHIV	Bit 4	R	发送 FIFO 低于阈值中断有效 0: 无效 1: 中断有效
TXUDIV	Bit 3	R	发送 FIFO 欠载中断有效 0: 无效 1: 中断有效
TXOVIV	Bit 2	R	发送 FIFO 溢出中断有效 0: 无效 1: 中断有效
Reserved	Bit 1	—	保留
TXEIV	Bit 0	R	发送 FIFO 空中断有效 0: 无效 1: 中断有效

28. 5. 2. 15 I2C原始中断标志寄存器 (I2Cx\_RIF)

I2C 原始中断标志寄存器 (I2Cx_RIF)																																										
偏移地址: 38 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												ALERTRI	TOUTRI	PECERI	ARLORI	BERRRI	Reserved	STOPRI	NACKRI	ADDRRI	TCRRI	TCRI	RXTHRI	RXUDRI	RXOVRI	RXFRI	Reserved	TXTHRI	TXUDRI	TXOVRI	Reserved	TXERI										

Reserved	Bit 31-21	—	保留
ALERTRI	Bit 20	R	<b>SMBus 警报中断标志状态</b> 0: 无效 1: 原始中断标志状态
TOUTRI	Bit 19	R	<b>超时中断标志状态</b> 0: 无效 1: 原始中断标志状态
PECERI	Bit 18	R	<b>PEC 错误中断标志状态</b> 0: 无效 1: 原始中断标志状态
ARLORI	Bit 17	R	<b>仲裁丢失中断标志状态</b> 0: 无效 1: 原始中断标志状态
BERRRI	Bit 16	R	<b>总线错误中断标志状态</b> 0: 无效 1: 原始中断标志状态
Reserved	Bit 15	—	保留
STOPRI	Bit 14	R	<b>停止检测中断标志状态</b> 0: 无效 1: 原始中断标志状态
NACKRI	Bit 13	R	<b>NACK 接收中断标志状态</b> 0: 无效 1: 原始中断标志状态
ADDRRI	Bit 12	R	<b>地址匹配中断标志状态</b> 0: 无效 1: 原始中断标志状态
TCRRI	Bit 11	R	<b>传输完成并重新加载中断标志状态</b> 0: 无效 1: 原始中断标志状态
TCRI	Bit 10	R	<b>传输完成中断标志状态</b> 0: 无效 1: 原始中断标志状态



RXTHRI	Bit 9	R	接收 FIFO 超过阈值中断标志状态 0: 无效 1: 原始中断标志状态
RXUDRI	Bit 8	R	接收 FIFO 欠载中断标志状态 0: 无效 1: 原始中断标志状态
RXOVRI	Bit 7	R	接收 FIFO 溢出中断标志状态 0: 无效 1: 原始中断标志状态
RXFRI	Bit 6	R	接收 FIFO 满中断标志状态 0: 无效 1: 原始中断标志状态
Reserved	Bit 5	—	保留
TXTHRI	Bit 4	R	发送 FIFO 低于阈值中断标志状态 0: 无效 1: 原始中断标志状态
TXUDRI	Bit 3	R	发送 FIFO 欠载中断标志状态 0: 无效 1: 原始中断标志状态
TXOVRI	Bit 2	R	发送 FIFO 溢出中断标志状态 0: 无效 1: 原始中断标志状态
Reserved	Bit 1	—	保留
TXERI	Bit 0	R	发送 FIFO 空中断标志状态 0: 无效 1: 原始中断标志状态

28.5.2.16 I2C中断标志屏蔽寄存器 (I2Cx\_IFM)

I2C 中断标志屏蔽寄存器 (I2Cx_IFM)																																
偏移地址: 3C <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												ALERTFM	TOUTFM	PECEFM	ARLOFM	BERRFM	Reserved	STOPFM	NACKFM	ADDRFM	TCRFM	TCFM	RXTHFM	RXUDFM	RXOVFM	RXFFM	Reserved	TXTHFM	TXUDFM	TXOVFM	Reserved	TXEFM

Reserved	Bit 31-21	—	保留
ALERTFM	Bit 20	R	<b>SMBus 报警中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
TOUTFM	Bit 19	R	<b>超时中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
PECEFM	Bit 18	R	<b>PEC 错误中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
ARLOFM	Bit 17	R	<b>仲裁丢失中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
BERRFM	Bit 16	R	<b>总线错误中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
Reserved	Bit 15	—	保留
STOPFM	Bit 14	R	<b>停止检测中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
NACKFM	Bit 13	R	<b>NACK 接收中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
ADDRFM	Bit 12	R	<b>地址匹配中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
TCRFM	Bit 11	R	<b>传输完成并重新加载中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态
TCFM	Bit 10	R	<b>传输完成中断标志屏蔽状态</b> 0: 无效 1: 中断标志屏蔽状态

RXTHFM	Bit 9	R	接收 FIFO 超过阈值中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXUDFM	Bit 8	R	接收 FIFO 欠载中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXOVFM	Bit 7	R	接收 FIFO 溢出中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXFFM	Bit 6	R	接收 FIFO 满中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
Reserved	Bit 5	—	保留
TXTHFM	Bit 4	R	发送 FIFO 低于阈值中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
TXUDFM	Bit 3	R	发送 FIFO 欠载中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
TXOVFM	Bit 2	R	发送 FIFO 溢出中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
Reserved	Bit 1	—	保留
TXEFM	Bit 0	R	发送 FIFO 空中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态

28. 5. 2. 17 I2C中断清除寄存器 (I2Cx\_ICR)

I2C 中断清除寄存器 (I2Cx_ICR)																																										
偏移地址: 40 <sub>H</sub>																																										
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
Reserved												ALERTIC	TOUTIC	PECEIC	ARLOIC	BERRIC	Reserved	STOPIC	NACKIC	ADDRIC	TCRIC	TCIC	RXTHIC	RXUDIC	RXOVIC	RXFIC	Reserved	TXTHIC	TXUDIC	TXOVIC	Reserved	TXEIC										

Reserved	Bit 31-21	—	保留
ALERTIC	Bit 20	C_W1	<b>SMBus 警报中断清除</b> 0: 无效 1: 中断清除
TOUTIC	Bit 19	C_W1	<b>超时中断清除</b> 0: 无效 1: 中断清除
PECEIC	Bit 18	C_W1	<b>PEC 错误中断清除</b> 0: 无效 1: 中断清除
ARLOIC	Bit 17	C_W1	<b>仲裁丢失中断清除</b> 0: 无效 1: 中断清除
BERRIC	Bit 16	C_W1	<b>总线错误中断清除</b> 0: 无效 1: 中断清除
Reserved	Bit 15	—	保留
STOPIC	Bit 14	C_W1	<b>停止检测中断清除</b> 0: 无效 1: 中断清除
NACKIC	Bit 13	C_W1	<b>NACK 接收中断清除</b> 0: 无效 1: 中断清除
ADDRIC	Bit 12	C_W1	<b>地址匹配中断清除</b> 0: 无效 1: 中断清除
TCRIC	Bit 11	C_W1	<b>传输完成并重新加载中断清除</b> 0: 无效 1: 中断清除
TCIC	Bit 10	C_W1	<b>传输完成中断清除</b> 0: 无效 1: 中断清除

RXTHIC	Bit 9	C_W1	接收 FIFO 超过阈值中断清除 0: 无效 1: 中断清除
RXUDIC	Bit 8	C_W1	接收 FIFO 欠载中断清除 0: 无效 1: 中断清除
RXOVIC	Bit 7	C_W1	接收 FIFO 溢出中断清除 0: 无效 1: 中断清除
RXFIC	Bit 6	C_W1	接收 FIFO 满中断清除 0: 无效 1: 中断清除
Reserved	Bit 5	—	保留
TXTHIC	Bit 4	C_W1	发送 FIFO 低于阈值中断清除 0: 无效 1: 中断清除
TXUDIC	Bit 3	C_W1	发送 FIFO 欠载中断清除 0: 无效 1: 中断清除
TXOVIC	Bit 2	C_W1	发送 FIFO 溢出中断清除 0: 无效 1: 中断清除
Reserved	Bit 1	—	保留
TXEIC	Bit 0	C_W1	发送 FIFO 空中断清除 0: 无效 1: 中断清除

## 第29章 串行外设接口(SPI)

### 29.1 概述

SPI /I2S 接口可用于使用 SPI 协议或 I2S 音频协议来与外部设备通信的情形。SPI 或 I2S 模式可通过软件选择。IP 复位后，默认选择 SPI 模式。

串行外设接口 (SPI) 可与外部器件进行半双工/全双工的同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟 (SCK)。该接口还能够得多主模式配置下工作。

此接口能运行多种模式，如基于双线的单工同步传输，其中一条线用于双向数据传输，同时可用 CRC 校验提高通信可靠性。

Inter-IC sound (I2S) 协议也是同步串行通信接口。它可以通过半双工通信在从机或主机模式下运行。通过组合两个 I2S 块可以实现全双工操作。它可以满足四种不同的音频标准，包括 Philips I2S 标准，MSB 和 LSB 对齐标准以及 PCM 标准。

### 29.2 特性

#### SPI 的主要特点

- ◆ 主机或从机操作
- ◆ 基于三条线的全双工同步传输
- ◆ 基于双线的半双工同步传输 (使用双向数据线)
- ◆ 基于双线的单工同步传输 (使用单向数据线)
- ◆ 8 位或 16 位传输帧格式选择
- ◆ 支持多主机模式功能
- ◆ 8 个主机模式波特率预分频器，最高可达  $F_{PCLK} / 2$
- ◆ 对于主机模式和从机模式都可通过硬件或软件进行 NSS 控制：动态切换主机/从机操作
- ◆ 时钟极性和相位可编程
- ◆ 数据顺序可编程，如最先移位 MSB 或 LSB
- ◆ 具有专用的发送和接收中断标志功能
- ◆ 支持 SPI 总线忙状态标志
- ◆ 支持摩托罗拉 SPI 协议
- ◆ 支持 TI SPI 协议
- ◆ 用于确保可靠通信的硬件 CRC 功能：
  - ◇ 在发送模式下可将 CRC 值作为最后一个字节数据发送
  - ◇ 根据收到的最后一个字节自动进行 CRC 错误校验
- ◆ 提供发送和接收 FIFO，深度为 16
- ◆ 支持 DMA 传输

### I2S 的主要特点

- ◆ 半双工通信（仅发送器或接收器）
- ◆ 仅支持主机操作
- ◆ 8 位可编程线性预分频器，可达到精确的音频采样频率（从 8 kHz 到 96 kHz）
- ◆ 数据格式可以是 16 位，24 位或 32 位
- ◆ 通过音频通道将通道帧固定为 16 位（16 位数据帧）或 32 位（16 位，24 位，32 位数据帧）
- ◆ 可编程时钟极性（稳态）
- ◆ 提供接收与发送的下溢与溢出标志
- ◆ 16 位寄存器，用于发送和接收
- ◆ 支持的 I2S 协议：
  - ◇ I2S 飞利浦标准
  - ◇ MSB 对齐标准（左对齐）
  - ◇ LSB 对齐标准（右对齐）
  - ◇ PCM 标准（16 位通道帧上的短帧和长帧同步或扩展到 32 位通道帧的 16 位数据帧）
- ◆ 数据方向始终为 MSB 优先
- ◆ 发送和接收的 DMA 功能（16 位宽）
- ◆ 可输出主时钟以驱动外部音频组件。比率固定为  $256 \times F_S$ （ $F_S$  为音频采样频率）
- ◆ 提供发送和接收 FIFO，深度为 16
- ◆ 提供两个可选的外部时钟输入引脚

## 29.3 结构框图

SPI 允许 MCU 与外部设备之间进行同步串行通信。应用软件可以通过轮询状态标志或使用专用 SPI 中断来管理通信。SPI 的主要模块及其相互关系如下面的框图所示。

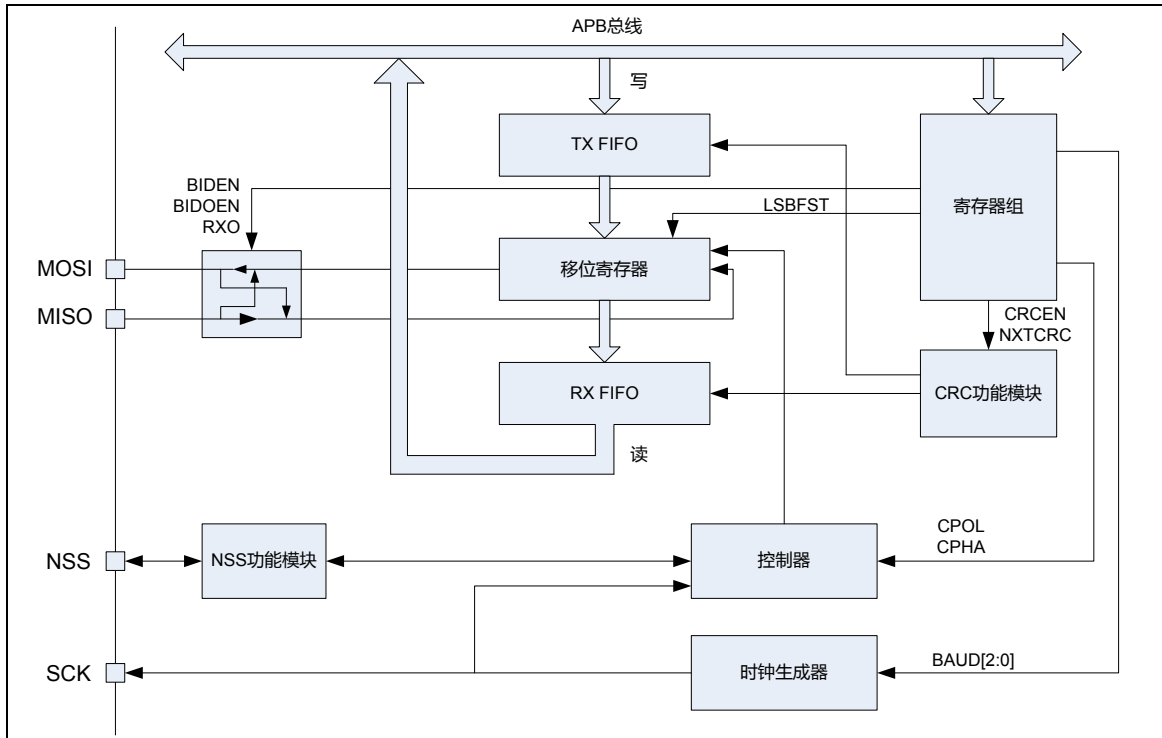


图 29-1 SPI 电路结构框图

通常，SPI 使用四个 I/O 引脚来与外部设备进行通信。

- ◇ MISO:主机输入/从机输出数据
- ◇ MOSI:主机输出/从机输入数据
- ◇ SCK:用于 SPI 主机的串行时钟输出以及 SPI 从机的串行时钟输入。
- ◇ NSS:从机选择引脚。这是用于选择从机的可选引脚。此引脚用作“片选”，可让 SPI 主机与从机进行单独通信，从而避免数据线上的竞争。从机的 NSS 输入可由主机上的标准 IO 端口驱动。NSS 引脚在使能（SPI\_CON2.NSSOE 位）时还可用作输出，并可在 SPI 处于主模式配置时驱动为低电平。通过这种方式，只要器件配置成 NSS 硬件控制模式，所有连接到该主机 NSS 引脚的其它器件 NSS 引脚都将呈现低电平，并因此而作为从机。当配置为主模式，且 NSS 配置为输入（SPI\_CON1.MSTREN=1 且 SPI\_CON2.NSSOE=0）时，如果 NSS 拉至低电平，SPI 将进入模式错误状态：SPI\_CON1.MSTREN 位自动清零，并且器件配置为从模式（参见章节 SPI 错误标志）

## 29.4 功能描述

### 29.4.1 通信模式

在 SPI 通信期间，接收和发送操作同时执行。串行时钟（SCK）同步数据线上信息的移位和采样。通信格式取决于时钟相位，时钟极性和数据帧格式。为了能够同时通信，主机和从机必须遵循相同的通信格式。

注 1: 在作为从机通讯时，从机所使用的系统频率需大于总线上的传输频率 16 倍以上才可正常工作。

注 2: 在作为从机通讯时，需要在主机发起传输前写入数据。



### 29.4.1.1 时钟相位和极性控制

通过配置 SPI\_CON1.CPOL 和 SPI\_CON1.CPHA 位，可以用软件选择四种可能的时序关系。SPI\_CON1.CPOL（时钟极性）位控制空闲时时钟线上的电平状态，此位对主机和从机都有作用。如果复位 SPI\_CON1.CPOL 位，SCK 引脚在空闲状态时处于低电平。如果将 SPI\_CON1.CPOL 位置 1，SCK 引脚在空闲状态时处于高电平。

如果将 SPI\_CON1.CPHA 位置 1，则 SCK 引脚上的第二个边沿（如果 SPI\_CON1.CPOL 位配置为 0，则为下降沿；如果 SPI\_CON1.CPOL 位配置为 1，则为上升沿）对 MSB 采样。即，在第二个时钟边沿锁存数据。如果复位 CPHA 位，则 SCK 引脚上的第一个边沿（如果 SPI\_CON1.CPOL 位配置为 0，则为上升沿；如果 SPI\_CON1.CPOL 位配置为 1，则为下降沿）对 MSB 采样。即在第一个时钟边沿锁存数据。

用户通过组合 SPI\_CON1.CPOL 和 SPI\_CON1.CPHA 位来选择数据捕获的时钟边沿。

下图显示了在 SPI\_CON1.CPHA 和 SPI\_CON1.CPOL 位的四种组合下的 SPI 传输。可以将该图解释为主机或从机时序图，其中 SCK 引脚、MISO 引脚、MOSI 引脚直接连接在主机和从机之间。

注：在切换 SPI\_CON1.CPOL/SPI\_CON1.CPHA 位之前，必须通过复位 SPI\_CON1.SPIEN 位来关闭 SPI。

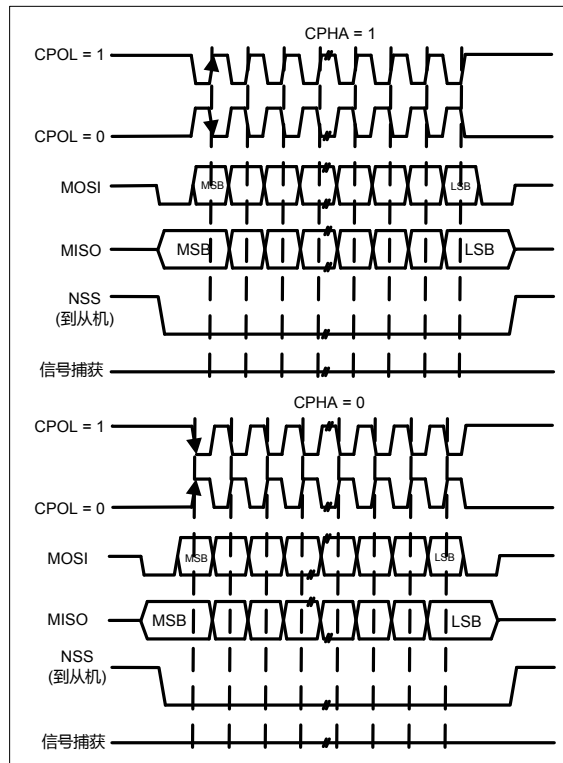


图 29-2 SPI 模式

注：数据位的顺序取决于 LSBFST 位的设置。

### 29.4.1.2 数据帧格式

SPI 移位寄存器可以设置为移出 MSB 优先或 LSB 优先，具体取决于 SPI\_CON1.LSBFST

位的值。每个数据帧为 8 或 16 位长，具体取决于 SPI\_CON1.FLEN 位的配置。所选的数据帧长度适用于发送和/或接收。

### 29.4.2 从机选择 (NSS) 引脚管理

可以使用 SPI\_CON1.SSEN 位设置硬件或软件控制从机选择。

- ◇ 软件控制 NSS (SPI\_CON1.SSEN = 1) 从机选择信息在内部由 SPI\_CON1.SSOUT 位的值驱动。
- ◇ 硬件管理 NSS (SPI\_CON1.SSEN = 0) 根据 NSS 输出配置 (SPI\_CON2.NSSOE 位)，硬件管理 NSS 有两种模式。
  - NSS 输出使能 (SPI\_CON1.SSEN = 0, SPI\_CON2.NSSOE = 1) 仅当器件在主机模式下工作时才使用此配置。当主机开始传输数据时，NSS 信号驱动为低电平，并保持到数据传输结束为止。
  - NSS 输出禁止 (SPI\_CON1.SSEN = 0, SPI\_CON2.NSSOE = 0) 对于在主机模式下工作的器件，此配置允许多主模式功能。对于设置为从机模式的器件，NSS 引脚用作传统 NSS 输入：在 NSS 为低电平时片选该从机，在 NSS 为高电平时取消对它的片选。

### 29.4.3 单对单应用

SPI 允许 MCU 使用不同的配置进行通信，具体取决于所针对的设备和应用要求。这些配置使用 2 或 3 线（使用软件 NSS 管理）或者是 3 或 4 线（使用硬件 NSS 管理）。通信始终由主机启动。

#### 29.4.3.1 全双工通信

默认情况下，SPI 配置为全双工通信。在此配置中，MOSI 引脚连接在一起，MISO 引脚连接在一起。通过这种方式，主机和从机之间以串行方式传输数据（最高有效位在前）。

通信始终由主机发起。当主机通过 MOSI 引脚向从机发送数据时，从机同时通过 MISO 引脚发出准备好的数据。这是一个数据输出和数据输入都由同一时钟进行同步的全双工通信过程，时钟信号由主机的 SCK 引脚发出提供给从机。

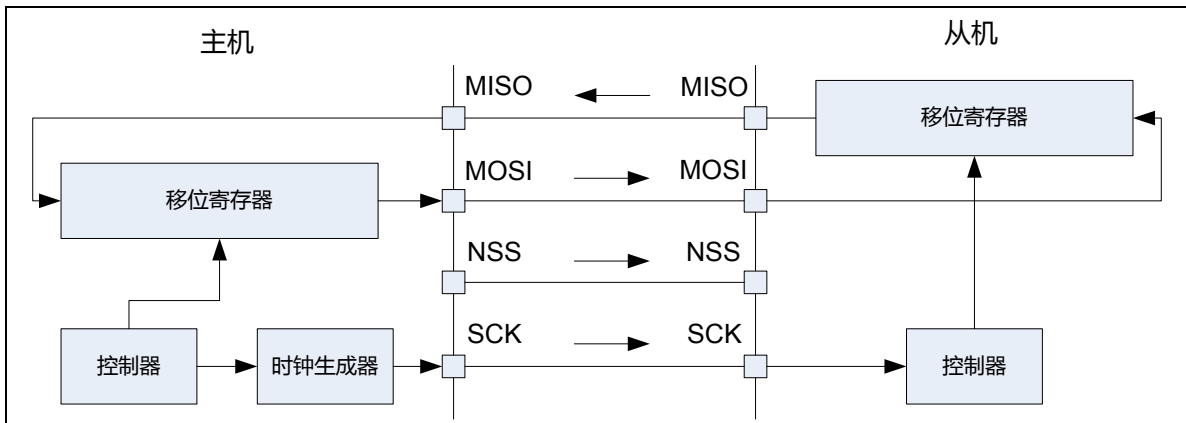


图 29-3 全双工通信

### 29.4.3.2 半双工通信

SPI 能够在以下两种配置中以半双工模式工作。

#### 1 个时钟和 1 条双向数据线 (SPI\_CON1.BIDEN=1)

可将 SPI\_CON1.BIDEN 位置 1 来使能此模式。在此模式下, SCK 作为时钟信号输出引脚, MOSI (主机模式下) 或 MISO (从机模式下) 作为数据通信引脚。通过 SPI\_CON1.BIDOEN 位来选择传输方向 (输入/输出)。当该位置 1 时, 数据线为输出, 否则为输入。

#### 1 个时钟和 1 条单向数据线 (SPI\_CON1.BIDEN=0)

在此模式下, 应用程序可使用 SPI 的只发送或只接收功能。

- ◇ 只发送模式类似于全双工模式: 在发送引脚 (主机模式下的 MOSI 或从机模式下的 MISO) 上发送数据, 不再接收数据。
- ◇ 只接收模式下, 应用程序可将 SPI\_CON1.RXO 位置 1 来关闭 SPI 输出功能。

当 SPI 进入只读模式后:

- ◇ 一旦在主机模式下使能 SPI 后, 主机会等待用户写入数据之后开始从 SCK 引脚发送时钟, 需要读回多少个数据由写入无效数据的个数来决定, 当 TX FIFO 中的无效的数据发送完毕后, 通信也立即停止。
- ◇ 在从机模式下, 只要 NSS 引脚被拉低 (或在 NSS 软件模式下将 SPI\_CON1.SSOUT 位清零), 意味着从机被选中, 同时一直有来自主机的 SCK 输入, SPI 就会继续接收。

### 29.4.4 数据发送和接收

#### 接收和发送 FIFO 缓存

所有 SPI 数据传输都通过嵌入式 16 级深度的 FIFO 缓存。使 SPI 能够连续传输工作, 并在数据帧长度较短时防止接收溢出。发送和接收都有自己的 FIFO 缓存。

对 SPI\_DATA 寄存器的读访问将返回存储在接收 FIFO 缓存中但尚未读取的最旧的值。对 SPI\_DATA 的写访问将已写数据存储在发送队列末尾的发送 FIFO 缓存中。SPI\_STAT 寄存器中 RXFLV [4:0] 和 TXFLV [4:0] 位域指示两个 FIFO 缓存的有效数据个数。

对 SPI\_DATA 寄存器的读访问必须由 RXTH 事件管理。当数据存储在接收 FIFO 缓存中并且达到阈值 (由 SPI\_CON2 寄存器中 RXFTH 位域定义) 时, 触发此事件。当 RXTH 被清除时, 表示接收 FIFO 缓存中的有效数据个数小于阈值。以类似的方式, 要发送的数据帧的写访问由 TXTH 事件管理。当发送 FIFO 缓存有效数据个数小于或等于阈值 (由 SPI\_CON2 寄存器中 TXFTH 位域定义) 时将触发此事件。

#### 在主机模式下启动通信序列

- ◇ 在全双工通信 (SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=0)
  - 将数据写入到 SPI\_DATA 寄存器 (发送 FIFO 缓存) 后, 启动通信序列。
  - 随后在第一个位的发送期间, 将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MOSI 引脚。
  - 同时, 将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器, 然后并行加载到 SPI\_DATA 寄存器 (接收 FIFO 缓存) 中。

- ◇ 在单工通信-只接收模式 (**SPI\_CON1.BIDEN=0** 且 **SPI\_CON1.RXO=1**)
  - 将无效数据写入到 **SPI\_DATA** 寄存器 (发送 FIFO 缓存) 后, 启动通信序列。
  - 随后在第一个位的发送期间, 将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MOSI 引脚。
  - 同时, 将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器, 然后并行加载到 **SPI\_DATA** 寄存器 (接收 FIFO 缓存) 中。
- ◇ 在半双工通信-发送模式 (**SPI\_CON1.BIDEN=1** 且 **SPI\_CON1.BIDOEN=1**)
  - 将数据写入到 **SPI\_DATA** 寄存器 (发送 FIFO 缓存) 时, 通信序列启动。
  - 随后在第一个位的发送期间, 将数据从发送缓冲区并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MOSI 引脚。
  - 不接收任何数据。
- ◇ 在半双工通信-接收模式 (**SPI\_CON1.BIDEN=1** 且 **SPI\_CON1.BIDOEN=0**)
  - 只要 **SPI\_CON1.SPIEN=1** 且 **SPI\_CON1.BIDOEN=0**。
  - 将无效数据写入到 **SPI\_DATA** 寄存器 (发送 FIFO 缓存) 后, 启动通信序列。
  - 随后在第一个位的发送期间, 将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MOSI 引脚。
  - 同时, 将 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器, 然后并行加载到 **SPI\_DATA** 寄存器 (接收 FIFO 缓存) 中。

#### 在从机模式下启动通信序列

- ◇ 在全双工模式 (**SPI\_CON1.BIDEN=0** 且 **SPI\_CON1.RXO=0**)
  - 当从机在其 MOSI 引脚上收到时钟信号和数据的第一位时, 通信序列开始。其余 7 个位将加载到移位寄存器中。
  - 同时, 在第一个位的发送期间, 将数据从发送缓冲区并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MISO 引脚。在 SPI 主机启动传输前, 软件必须已把要从机发送的数据写入发送 FIFO 缓存。
- ◇ 在单工通信-只接收模式 (**SPI\_CON1.BIDEN=0** 且 **SPI\_CON1.RXO=1**)
  - 当从机在其 MOSI 引脚上收到时钟信号和数据的第一位时, 通信序列开始。其余 7 个位将加载到移位寄存器中。
  - 由于发送器没有激活, 因此不会有数据以串行方式移出 MISO 引脚。
- ◇ 在半双工通信-发送模式 (**SPI\_CON1.BIDEN=1** 且 **SPI\_CON1.BIDOEN=1**)
  - 当从机收到时钟信号, 并且 MISO 引脚上发出发送 FIFO 缓存中的第一位数据时, 通信序列开始。
  - 随后在第一个位的发送期间, 将数据从发送 FIFO 缓存并行加载到 8 位移位寄存器中, 然后以串行方式将其移出到 MISO 引脚。在 SPI 主机启动传输前, 软件必须已把要从机发送的数据写入发送 FIFO 缓存。
  - 不接收任何数据。
- ◇ 在半双工通信-接收模式 (**SPI\_CON1.BIDEN=1** 且 **SPI\_CON1.BIDOEN=0**)
  - 当从机在其 MISO 引脚上收到时钟信号和数据的第一位时, 通信序列开始。

- 在 MISO 引脚上接收的数据以串行方式移入 8 位移位寄存器，然后并行加载到 **SPI\_DATA** 寄存器（接收 FIFO 缓存）中。
- 由于发送器没有被激活，因此不会有数据以串行方式移出 MISO 引脚。

**处理数据发送与接收**

全双工通信（**SPI\_CON1.BIDEN=0** 且 **SPI\_CON1.RXO=0**），发送和接收数据的处理过程直接存取操作模式：

1. 通过将 **SPI\_CON1.SPIEN** 位置 1 来使能 SPI，将第一个要发送的数据项写入 **SPI\_DATA** 寄存器（此操作会将 **SPI\_STAT.TXE** 位清零）。
2. 等待 **SPI\_STAT.TXE=1**，然后写入要发送的第二个数据项。然后等待 **SPI\_STAT.RXE=0**，读取 **SPI\_DATA** 以获取第一个接收到的数据（此操作会将 **SPI\_STAT.RXE** 位置 1）。对每个要发送和接收的数据项重复此操作，直到发送并接收完最后的数据。
3. 检查 **SPI\_STAT.TXE=1**，然后等待至 **SPI\_STAT.BUSY=0**，再关闭 SPI。
4. 此外，还可以使用 TXE 或 RXTH(将 **SPI\_CON2.RXFTH** 位域置 1)中断事件对应的各个中断子程序来实现该过程。

FIFO 缓存操作模式：

1. 通过将 **SPI\_CON1.SPIEN** 位置 1 来使能 SPI。
2. 配置 **SPI\_CON2.TXFTH** 與 **SPI\_CON2.RXFTH**。
3. 当 **SPI\_STAT.TXTH=1**，将要发送的数据写入 **SPI\_DATA** 寄存器（写入的数据个数必须大于 **SPI\_CON2.TXFTH** 设定的阈值），当 **SPI\_STAT.RXTH=1**，读取 **SPI\_DATA** 寄存器以获取接收到的数据（读取的数据个数必须为 **SPI\_CON2.RXFTH** 设定的阈值），重复此操作直到写入最后要发送的数据。
4. 等待至 **SPI\_STAT.BUSY=0**，读取 **SPI\_DATA** 寄存器以获取接收到的数据直到 **SPI\_STAT.RXFLV** 位置 0，再关闭 SPI。
5. 此外，还可以使用 TXTH 或 RXTH 中断事件对应的各个中断子程序来实现该过程。

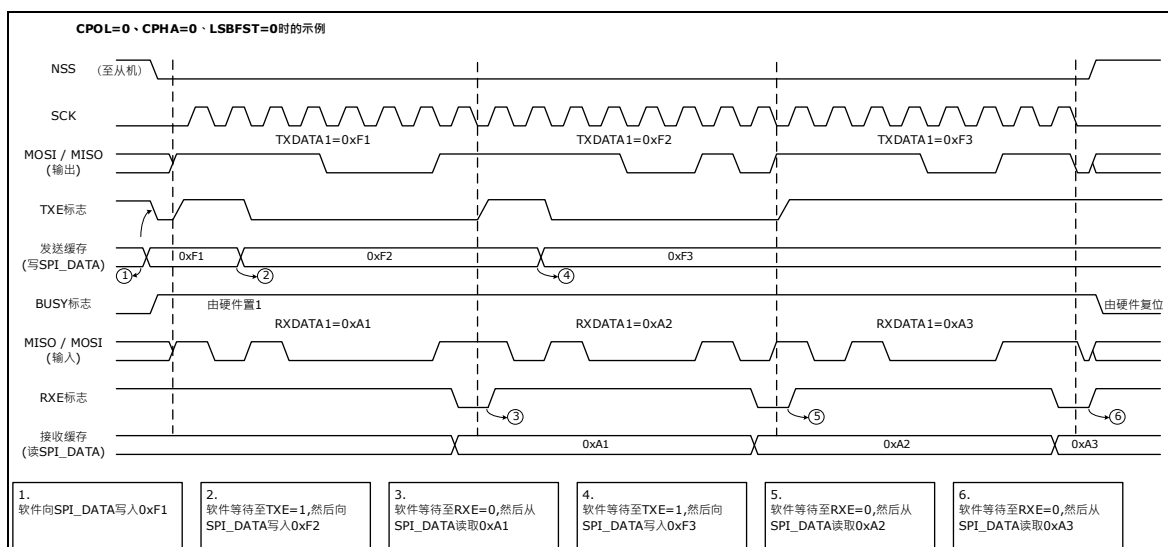


图 29-4 全双工通信(**SPI\_CON1.BIDEN=0** 且 **SPI\_CON1.RXO=0**)的 TXE、RXE、BUSY 行为(直接存取操作模式在连续传输的情况下)

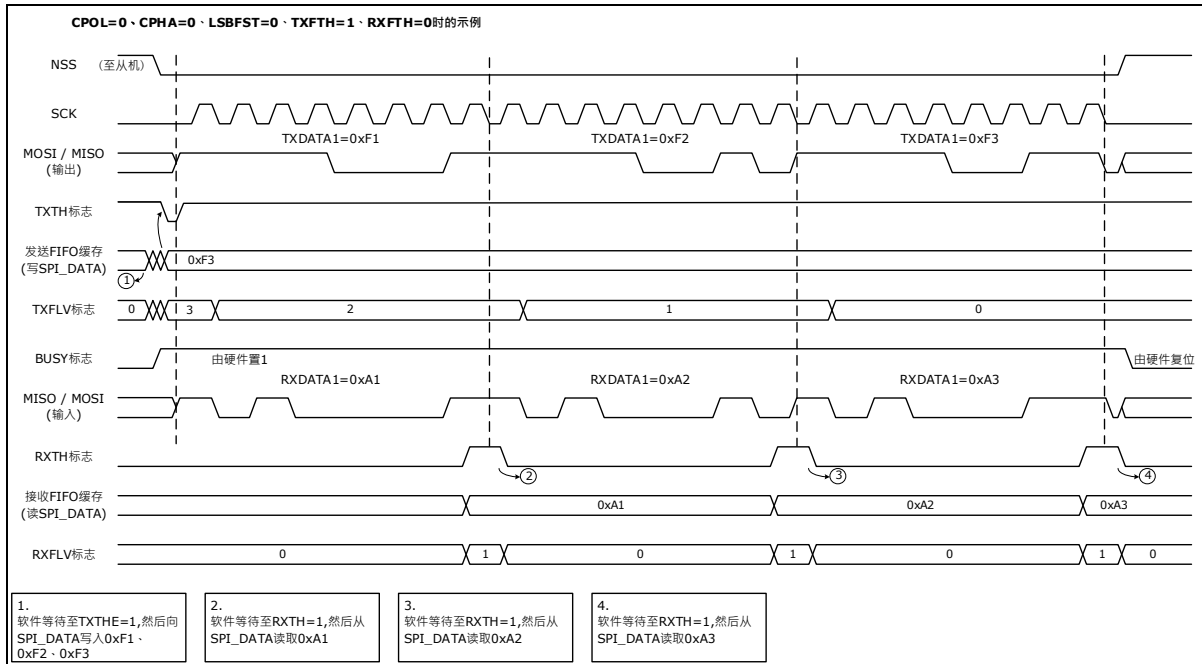


图 29-5 全双工通信(SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=0)的 TXTH、RXTH、TXFLV、RXFLV、BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

单工通信-只发送模式 (SPI\_CON1.BIDEN=0、SPI\_CON1.RXO=0)，发送数据的处理过程直接存取操作模式:

1. 通过将 SPI\_CON1.SPIEN 位置 1 来使能 SPI。
2. 等待 SPI\_STAT.TXE=1 然后写入要发送的数据。对每个要发送的数据项重复此步骤。
3. 将最后一个数据写入 SPI\_DATA 寄存器后，等待至 SPI\_STAT.TXE=1，然后等待至 SPI\_STAT.BUSY=0 再关闭 SPI，这表示最后的数据发送完成。
4. 此外，还可以使用在 TXE 中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式:

1. 通过将 SPI\_CON1.SPIEN 位置 1 来使能 SPI。
2. 配置 SPI\_CON2.TXFTH。
3. 当 SPI\_STAT.TXTH=1，将要发送的数据写入 SPI\_DATA 寄存器（写入的数据个数必须大于 SPI\_CON2.TXFTH 设定的阈值），重复此操作直到写入最后要发送的数据。
4. 等待至 SPI\_STAT.BUSY=0 再关闭 SPI。
5. 此外，还可以使用 TXTH 中断事件对应的中断子程序来实现该过程。

注意:

注 1: 在不连续通信期间，在对 SPI\_DATA 寄存器执行写操作与 SPI\_STAT.BUSY 位置 1 之间有延迟。因此在只发送模式下，写入最后的数据后，必须先等待 SPI\_STAT.TXE 位置 1，然后等待 SPI\_STAT.BUSY 位清零。

注 2: 在只发送模式下，发送 17 个数据项后，SPI\_STAT.RXOV 标志将置 1，因为始终不会读取接收的数据。

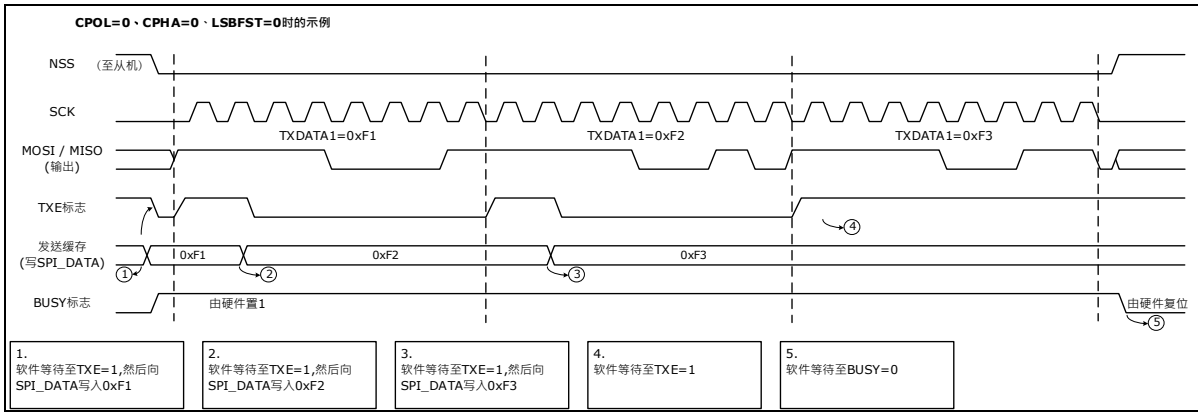


图 29-6 单工通信-只发送模式(SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=0)的 TXE、BUSY 行为  
(直接存取操作模式在连续传输的情况下)

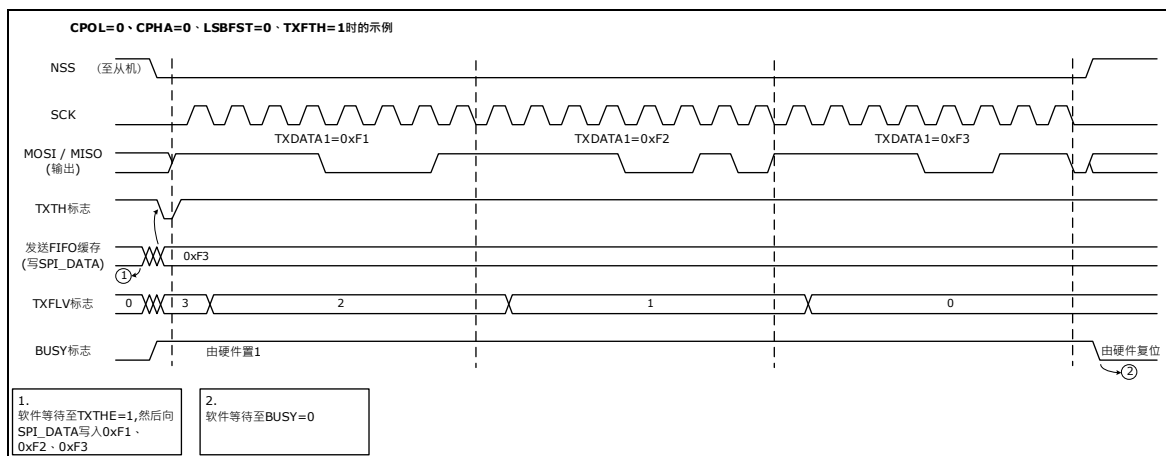


图 29-7 单工通信-只发送模式(SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=0)的 TXTH、TXFLV、  
BUSY 行为(FIFO 缓存操作模式在连续传输的情况下)

### 半双工通信-发送模式 (SPI\_CON1.BIDEN=1 且 SPI\_CON1.BIDOEN=1)，发送数据的处理过程

此模式与单工通信-只发送模式数据的处理过程相似，但是在 SPI 模块使能前，必须将 SPI\_CON1.BIDEN 位和 SPI\_CON1.BIDOEN 位置 1。

单工通信-只接收模式 (SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=1)，接收数据的处理过程直接存取操作模式：

1. 将 SPI\_CON2.RXO 位置 1。
2. 通过将 SPI\_CON1.SPIEN 位置 1 使能 SPI。
3. 在主机模式下。等待 SPI\_STAT.TXE=1，然后写入无效数据写入到 SPI\_DATA 寄存器。等待 SPI\_STAT.RXE=0，然后读取 SPI\_DATA 寄存器以获取接收的数据（此操作会将 SPI\_STAT.RXE 位置 1）。对每个要接收的数据项重复此操作。
4. 在从机模式下。等待 SPI\_STAT.RXE=0，然后读取 SPI\_DATA 寄存器以获取接收的数据（此操作会将 SPI\_STAT.RXE 位置 1）。对每个要接收的数据项重复此操作。
5. 此外，还可以使用 RXTH(将 SPI\_CON2.RXFTH 位域置 1)中断事件对应的中断子程序来实现该过程。

FIFO 缓存操作模式：

1. 将 SPI\_CON2.RXO 位置 1。

- 配置 **SPI\_CON2.RXFTH**。
- 通过将 **SPI\_CON1.SPIEN** 位置 1 使能 SPI。
- 在主机模式下。当 **SPI\_STAT.TXTH=1**，将无效数据写入 **SPI\_DATA** 寄存器（写入的数据个数必须大于 **SPI\_CON2.TXFTH** 设定的阈值），当 **SPI\_STAT.RXTH=1**，读取 **SPI\_DATA** 寄存器以获取接收到的数据（读取的数据个数必须为 **SPI\_CON2.RXFTH** 设定的阈值），重复此操作直到获取最后要接收的数据。
- 在从机模式下。当 **SPI\_STAT.RXTH=1**，读取 **SPI\_DATA** 寄存器以获取接收到的数据（读取的数据个数必须为 **SPI\_CON2.RXFTH** 设定的阈值），重复此操作直到获取最后要接收的数据。
- 此外，还可以使用 **RXTH** 中断事件对应的中断子程序来实现该过程。

注意：

- 在主机模式下，一旦 SPI 使能后 **SCK** 会立即发送时钟，从机接收到时钟后会发送数据，直到主机关闭 SPI 功能结束通信。
- 在从机模式下，当 **NSS** 被拉低并且接收到 **SCK** 时钟后开始接收数据。

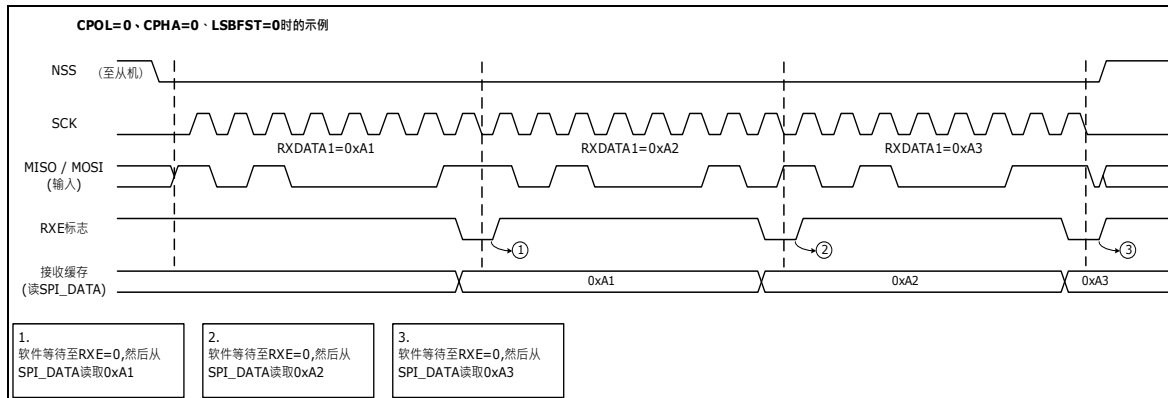


图 29-8 单工通信-只接收模式 (SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=1) 的 RXE 行为(直接存取操作模式在连续传输的情况下)

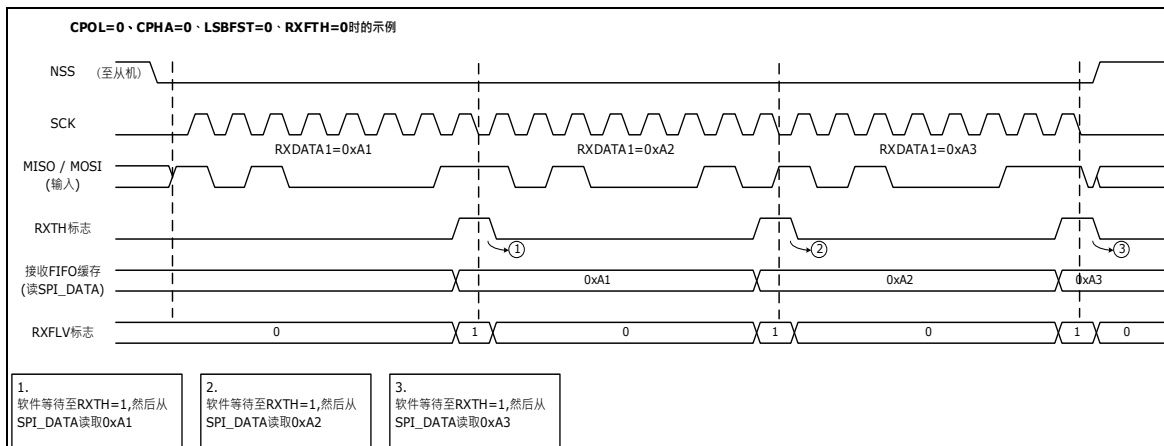


图 29-9 单工通信-只接收模式 (SPI\_CON1.BIDEN=0 且 SPI\_CON1.RXO=1) 的 RXTH、RXFLV 行为(FIFO 缓存操作模式在连续传输的情况下)



半双工通信-接收模式 (**SPI\_CON1.BIDEN=1** 和 **SPI\_CON1.BIDOEN=0**), 接收数据的处理过程

此模式与单工通信-只接收模式数据的处理过程相似, 但是在 SPI 模块使能之前, 需要将 **SPI\_CON1.BIDEN** 位置 1, 并将 **SPI\_CON1.BIDOEN** 与 **SPI\_CON1.RXO** 位清 0。

**连续传输和间断传输**

在主机模式下发送数据时, 如果软件处理速度足够快, 可以在检测到 **SPI\_STAT.TXE=1** (或发生 TXE 中断事件), 并且当前数据传输未结束, 立即将下一次的数据写入 **SPI\_DATA** 寄存器, 则能实现连续的通信。或者配置 **SPI\_CON2.TXFTH** 检测当 **SPI\_STAT.TXTH=1** (或发生 TXTH 中断事件), 将要发送的数据写入 **SPI\_DATA** 寄存器 (写入的数据个数必须大于 **SPI\_CON2.TXFTH** 设定的阈值), 实现连续的通信。观察到的现象是 **SPI\_STAT.BUSY** 位一直为 1 不被清除, 并且每个数据的 SPI 时钟保持连续。

相反, 如果软件速度不够快, 则可能导致通信中断。在这种情况下, 各数据传输之间会清零 **SPI\_STAT.BUSY** 位。

在主机或从机模式下的单工通信-只接收模式 (**SPI\_CON1.RXO=1**), 通信始终是连续的, 且 **SPI\_STAT.BUSY** 位始终为 1。

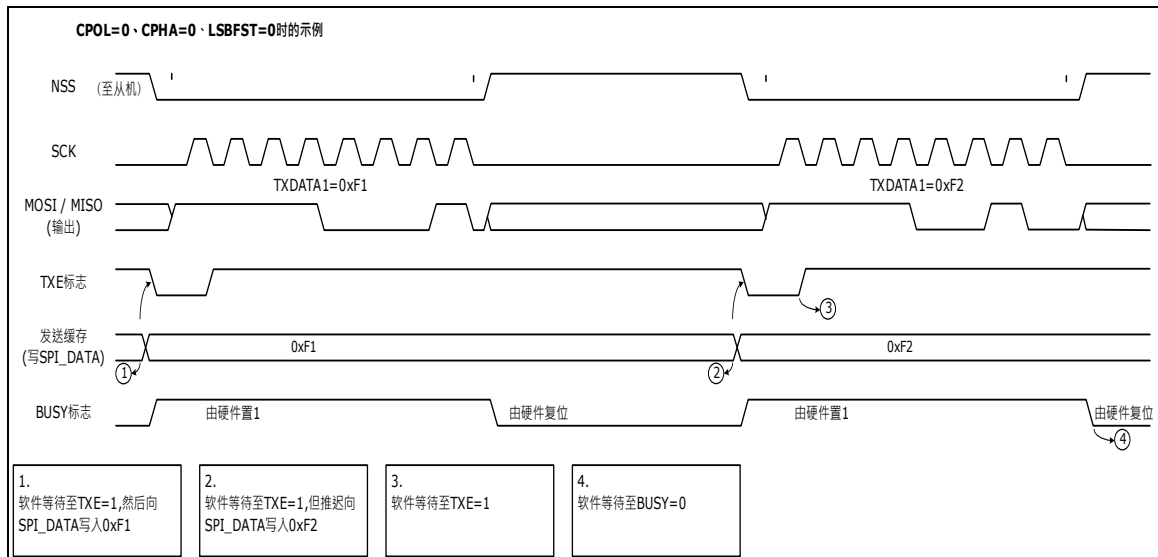


图 29-10 发送时 (**SPI\_CON1.BIDEN =0** 且 **SPI\_CON1.RXO=0**) 的 TXE/BUSY 行为 (在间断传输的情况下)

**29.4.5 DMA请求**

为了更方便的实现高速通信, SPI 提供了 DMA 功能。DMA 请求条件是根据 **SPI\_CON2** 寄存器中的 TXFTH 与 RXFTH 位域配置, 当使能 **SPI\_CON2** 寄存器中相应的 DMA 使能位时, 将请求 DMA 访问。发送 FIFO 缓存和接收 FIFO 缓存会发出各自的 DMA 请求 (参见图 26-11 和图 26-12):

- ◇ 在发送过程中, 当 **SPI\_STAT.TXTH** 位置 1 时会发出 DMA 请求。DMA 随后对 **SPI\_DATA** 寄存器执行写操作 (此操作会将 **SPI\_STAT.TXTH** 位清零)。
- ◇ 在接收过程中, 当 **SPI\_STAT.RXTH** 位置 1 时会发出 DMA 请求。DMA 随后对 **SPI\_DATA** 寄存器执行读操作 (此操作会将 **SPI\_STAT.RXTH** 位清零)。

当 SPI 仅用于只发送数据时, 可以只使能 SPI Tx DMA 通道。在这种情况下,

SPI\_STAT.RXOV 位会置 1，因为未读取接收的数据。

在发送模式下，DMA 完成了所有要发送数据的传输后，DMA\_RIF 会产生相对应通道的传输完成标志，使用者可以对 BUSY 标志进行监视，以确保 SPI 通信已完成。在关闭 SPI 或进入停止模式前必须等待 SPI\_STAT.BUSY=0。

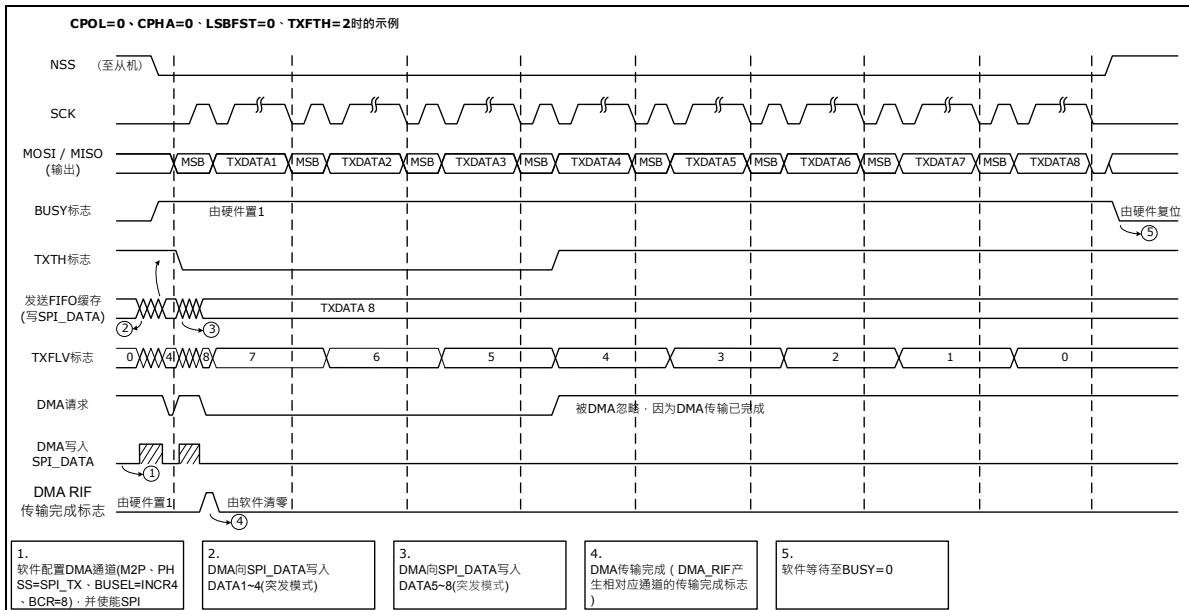


图 29-11 使用 DMA 进行发送

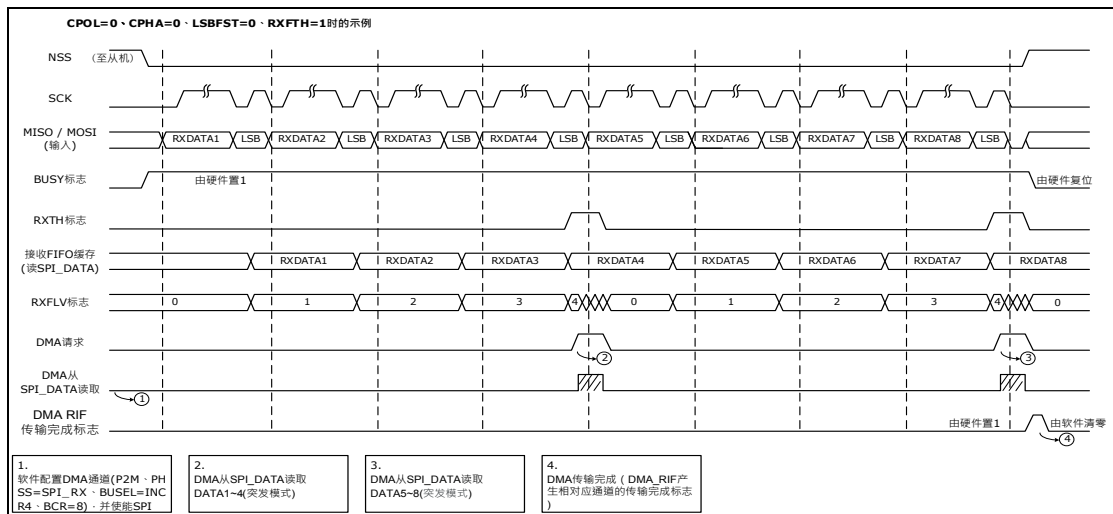


图 29-12 使用 DMA 进行接收

### 29.4.6 SPI兼容模式

该模式 SPI 接口与 TI SPI 通信协议兼容。SPI\_CON2 寄存器的 FRF 位可用于配置 SPI 以符合此协议。

无论 SPI\_CON1 寄存器中设置的值如何，时钟极性和相位都必须符合 TI 协议要求。NSS 管理也特定于 TI 协议，在这种情况下，通过 SPI\_CON1 和 SPI\_CON2 寄存器 (SSEN, SSOUT, NSSOE) 无法配置 NSS 管理。

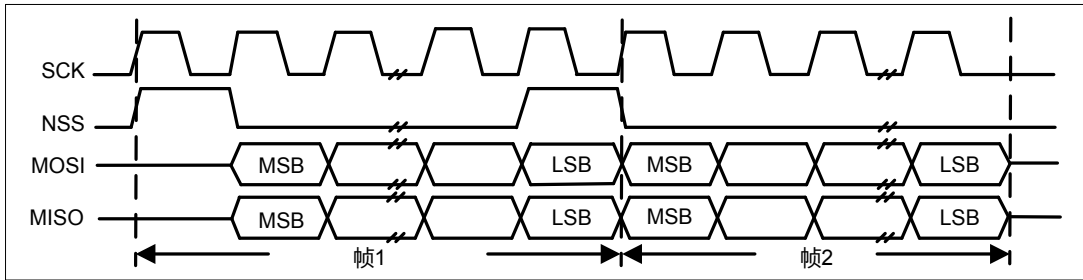


图 29-13 SPI 兼容模式通信波形

## 29.4.7 CRC 计算

### 29.4.7.1 CRC 功能描述

为确保通信的可靠性，SPI 模块实现了硬件 CRC 功能。针对发送或接收的数据帧宽度有 8 位和 16 位的选择，硬件 CRC 计算也提供了两种计算标准，分别为 8 位数据的 CRC8 和 16 位数据的 CRC16，具体取决于通过 FLEN 位选择的数据格式。使用 SPI\_CRCPOLY 寄存器中编程的多项式串行计算 CRC。目前 CRC 功能不支持 BAUD 设置为 0 与 1 的情况。

通过 SPI\_CON1.CRCEN 位置 1 来使能 CRC 计算功能，此操作会复位 CRC 寄存器（SPI\_RXCRC 和 SPI\_TXCRC）。使用每个位上的奇数可编程多项式计算 CRC 值。计算在 SPI\_CON1 寄存器中 CPHA 和 CPOL 位定义的采样时钟边沿处理。在数据块的末尾自动检查计算的 CRC 值以及由 CPU 管理的传输。当在接收数据内部计算的 CRC 与发送器发送的 CRC 之间检测到不匹配时，设置 CRCERRRI 位以指示数据损坏错误。处理 CRC 计算的正确过程取决于 SPI 配置和所选的传输管理。

注：多项式值应该只是奇数。不支持偶数值。

### 29.4.7.2 CRC传输管理

在此可分为 2 种方式来传送 CRC 帧：

◇ 跟随数据之后传输 CRC 帧

可在写入数据后设置 SPI\_CON1.NXTCRC 位，以指示在当前处理的数据帧传输完成之后将跟随 CRC 帧传输。在传输 CRC 帧时，CRC 会停止计算，并同时检测接收数据的 CRC 是否正确。当传输完全结束后，可在 SPI\_TXCRC 和 SPI\_RXCRC 寄存器中确认发送与接收的 CRC 值。

◇ 单独传输 CRC 帧

若数据帧已传输结束，此时需要单独发送 CRC 帧时，可直接设置 SPI\_CON1.NXTCRC 位，此时会直接发送已计算好的 CRC 数值，并同时检测接收数据的 CRC 是否正确。

注：接收的 CRC 帧会如同一般数据一样存入接收 FIFO 中。软件必须检查 SPI\_RIF 寄存器中的 CRCERRRI 位，以确定数据传输是否已损坏。软件通过向 SPI\_ICR 写入'1'来清除 CRCERRRI 位。

### 29.4.7.3 复位CRC方式

启用 CRC 计算时，SPI\_TXCRC 和 SPI\_RXCRC 值将自动清零。

在从机禁用（NSS 上的高电平）和新的从机使能（NSS 上的低电平）之间，应在主机和从机侧清除 CRC 值，以重新同步主机和从机各自的 CRC 计算。

要清除 CRC，请遵循以下顺序：

1. 禁用 SPI
2. 清零 CRCEN 位
3. 启用 CRCEN 位
4. 启用 SPI

### 29.4.8 SPI状态标志

为应用程序提供了五个状态标志，以完全监视 SPI 总线的状态。

#### 29.4.8.1 发送FIFO空标志（TXE）

设置时，TXE 标志指示发送 FIFO 为空，并且下一个要发送的数据可以加载到 FIFO 中。通过写 SPI\_DATA 寄存器清零 TXE 标志。

#### 29.4.8.2 发送FIFO满标志（TXF）

设置时，TXF 标志指示发送 FIFO 已满，并且下一个要发送的数据无法加载到 FIFO 中。TXF 标志由要发送的下一个数据清除。

#### 29.4.8.3 接收FIFO空标志（RXE）

设置时，RXE 标志指示接收 FIFO 为空，并且下一个要接收的数据可以加载到 FIFO 中。RXE 标志由下一个数据接收到 FIFO 中清除。

#### 29.4.8.4 接收FIFO满标志 (RXF)

设置后, RXF 标志指示接收 FIFO 已满, 并且下一个要接收的数据无法加载到 FIFO 中。通过对 SPI\_DATA 寄存器的读访问来清除 RXF 标志。

#### 29.4.8.5 通信忙 (BUSY)

BUSY 标志用于指示 SPI 通信的状态。此标志由硬件置 1 和清零。

SPI\_STAT.BUSY 位置 1 时, 表示 SPI 正在通信中。在通信结束前, 用户可检测 SPI\_STAT.BUSY 位是否为 0, 表示通信已结束, 此时关闭 SPI 模块, 停止通信。

BUSY 标志还可用于避免在多主模式系统中发生写冲突。

在以下情况硬件将清零该标志:

- ◇ 传输完成时 (主机模式下的连续通信除外)
- ◇ 关闭 SPI 时
- ◇ 发生模式错误时 (SPI\_RIF.MODFRI=1)

当通信不连续时, BUSY 标志在各通信之间处于低电平。

当通信连续时:

- ◇ 在主机模式下, BUSY 标志在所有传输期间均保持高电平
- ◇ 在从机模式下, BUSY 标志在各传输之间的一个 SPI 时钟周期内变为低电平

注: 建议始终使用 TXTH 和 RXTH 标志 (而不是 BUSY 标志) 来处理数据传输或接收操作。

## 29.4.9 SPI错误标志

如果设置了以下错误标志之一并通过将该位置 1 启用中断，则会产生 SPI 中断。

### 29.4.9.1 发送FIFO溢出标志 (TXOV)

发生发送 FIFO 溢出的条件是发送 FIFO 已满，但用户正在执行写操作。在这种情况下，新写入的数据不会加载到发送 FIFO 中。通过对 SPI\_ICR 寄存器的写访问来清除 TXOV 位，或者在新传输开始时它将自动清零。

### 29.4.9.2 接收FIFO溢出标志 (RXOV)

当主机或从机完成下一个数据帧的接收而接收 FIFO 的前 16 帧的读操作尚未完成时（设置了 RXF 标志的情况），发生接收溢出条件。

在这种情况下，接收 FIFO 的内容不会将接收的新数据存入。SPI\_DATA 寄存器的读操作返回先前接收的帧。所有其他在 RXF 标志设置的情况下，接收的新数据都将丢失。

通过对 SPI\_ICR 寄存器的写访问来清除 RXOV 位，或者在新的传输开始时它将自动清零。

### 29.4.9.3 接收FIFO下溢标志 (RXUD)

当接收 FIFO 为空时，但用户正在执行读操作，会发生接收下溢条件。在这种情况下，读取操作不会从接收 FIFO 中读到有效的数据。

通过对 SPI\_ICR 寄存器的写访问来清除 RXUD 位，或者在新的传输开始时它将自动清零。

### 29.4.9.4 主机模式故障 (MODF)

当主机模式的内部 NSS 信号清零（NSS 硬件模式下的 NSS 引脚拉低或 NSS 软件模式下的 SSOUT 位为 0）时，发生主机模式故障，这会 自动将 MODFRI 位置 1。主机模式故障通过以下方式影响 SPI 接口：

- ◇ MODFRI 位自动置 1，如果 MODFIV 位为 1，则产生 SPI 中断请求。
- ◇ MSTREN 位清零，强制 SPI 进入从机模式。

通过软件对 SPI\_ICR.MODFIC 写 1 来清除 MODFRI 位。

为避免在包含多个 MCU 的系统中发生任何从机冲突，必须在 MODFRI 位清零前将 NSS 引脚拉高。在 MODFRI 位清零之后，MSTREN 位若要恢复到其原始状态需软件重新对其写入。

在从机模式中，MODFRI 位不会被置 1。但发生主机模式故障时，器件会自动切换至从机模式，MODFRI 位被置 1，此时表示系统可能存在多主模式冲突。

### 29.4.9.5 CRC错误 (CRCERR)

该标志用于验证 SPI\_CON1 寄存器中的 CRCEN 位置 1 时接收到的值的有效性。如果移位寄存器中接收的值与接收器 SPI\_RXCRC 值不匹配，则 SPI\_RIF 寄存器中的 CRCERRRI 标志置 1。该标志由软件清除。

### 29.4.9.6 SPI兼容模式帧格式错误 (FRE)

当 SPI 在从机模式下工作并配置为兼容 TI 模式协议时,在正在进行的通信期间发生 NSS 脉冲时,会检测到兼容模式帧格式错误。发生此错误时,将在 SPI\_RIF 寄存器中设置 FRERI 标志。发生错误时,SPI 不会被禁用,NSS 脉冲被忽略,SPI 在开始新的传输之前等待下一个 NSS 脉冲。数据可能被破坏,因为错误检测可能导致两个数据字节的丢失。

如果 FREIE 位置 1,则在 NSS 错误检测时产生中断。在这种情况下,应禁用 SPI,因为不再保证数据一致性,并且当再次启用从 SPI 时,主机应重新启动通信。

## 29.5 I2S结构图

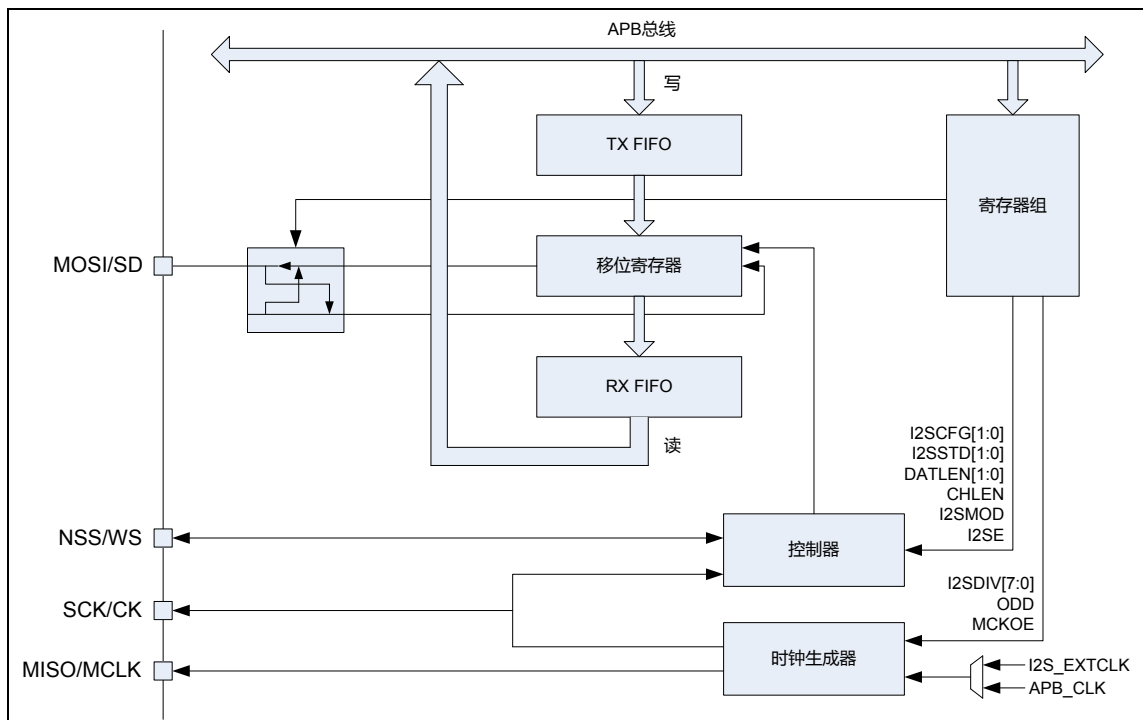


图 29-14 I2S 电路结构框图

当 I2S 功能使能时(通过设置 SPI\_I2SCFG.I2SMOD 位),SPI 可用作音频 I2S 接口。该接口主要使用与 SPI 相同的引脚,标志和中断。I2S 与 SPI 共享四个公共引脚:

- ◇ SD: 串行数据(映射在 MOSI 引脚上),用于发送或接收两个时间复用数据通道。
- ◇ WS: 声道选择(映射在 NSS 引脚上)是主机输出的数据控制信号。
- ◇ CK: 串行时钟(映射在 SCK 引脚上)是主机的串行时钟输出。
- ◇ MCLK: 当 SPI\_I2SPR.MCKOE 位置 1 时,使用主时钟(映射在 MISO 引脚上),输出以预先配置的频率等于  $256 \times FS$  生成的附加时钟,其中 FS 是音频采样频率。

I2S 使用自己的时钟发生器产生通信时钟。该时钟发生器也是主时钟输出的源。在 I2S 模式下有两个额外的寄存器。一个是 I2S 预分频寄存器 SPI\_I2SPR,另一个是通用 I2S 配置寄存器 SPI\_I2SCFG(音频标准,数据帧,通道帧,时钟极性等)。

在 I2S 模式下不使用 SPI\_CON1 寄存器和所有 CRC 寄存器。同样,不使用 SPI\_CON2 寄存器中的 NSSOE 位、中断相关寄存器中的 MODF 和 CRCERR 位。

I2S 在 16/32 位宽模式下使用相同的 SPI 寄存器进行数据传输 (SPI\_DATA)。

## 29.6 I2S功能描述

### 29.6.1 音频协议

三线总线通常在处理两个通道上进行时间复用的音频数据：右声道和左声道。但是，只有一个 16 位寄存器用于发送或接收。因此，由软件向数据寄存器写入与每个通道侧相对应的适当值，或者从数据寄存器读取数据。始终先发送左声道，然后右声道 (CHSIDE 对 PCM 协议没有意义)。

有四个数据和通道帧可供使用。数据可以采用以下格式发送：

- ◇ 16 位数据帧装在一个 16 位通道帧
- ◇ 16 位数据帧装在一个 32 位通道帧
- ◇ 24 位数据帧装在一个 32 位通道帧
- ◇ 32 位数据帧装在一个 32 位通道帧

当使用 16 位数据帧装在一个 32 位通道帧上时，前 16 位 (MSB) 是有效位，后 16 位 LSB 被强制为 0，无需任何软件操作或 DMA 请求 (只有一个读/写操作)。

如果 DMA 是应用程序的首选，则 24 位和 32 位数据帧需要对 SPI\_DATA 寄存器进行两次 CPU 读或写操作，或者需要两次 DMA 操作。具体而言，对于 24 位数据帧，8 个非有效位通过 0 位扩展为 32 位 (通过硬件)。

对于所有数据格式和通信标准，始终首先发送最高有效位 (MSB 优先)。

I2S 接口支持四种音频标准，可使用 SPI\_I2SCFG.I2SSTD 进行配置。



### 29.6.1.1 I2S飞利浦标准

对于该标准，WS 信号用于指示正在传输哪个信道。它在第一位（MSB）可用之前的一个 CK 时钟周期被激活。

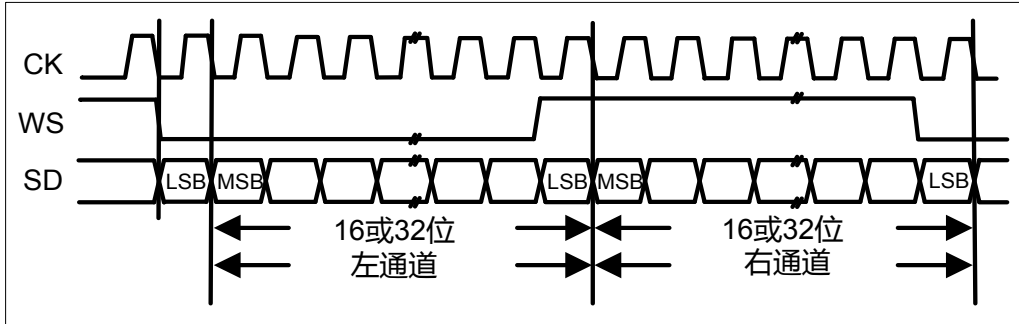


图 29-15 I2S 飞利浦标准波形（16/32 位数据帧，CPOL = 0）

数据在 CK 的下降沿（对于发送器）被锁存，并在上升沿（对于接收器）被读取。WS 信号也在 CK 的下降沿锁存。

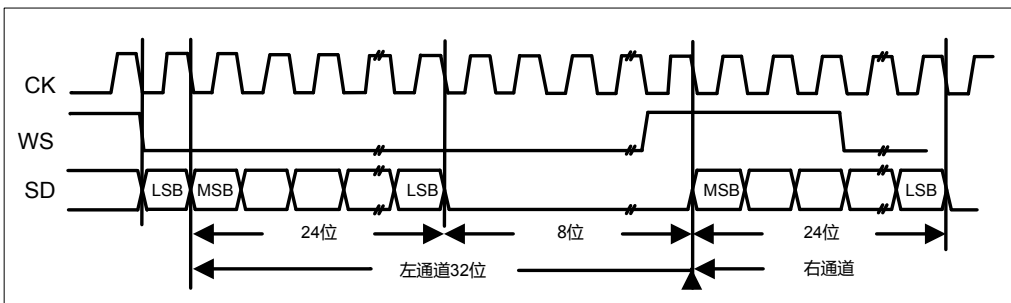


图 29-16 I2S 飞利浦标准波形（24 位数据帧，CPOL = 0）

该模式需要对 SPI\_DATA 寄存器进行两次写或读操作。

◇ 在传输模式下

如果必须发送 0x123456（24 位）

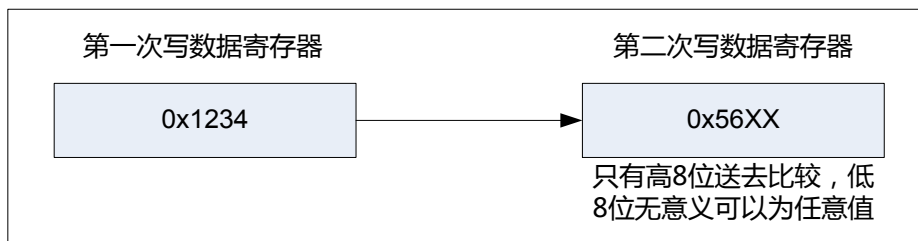


图 29-17 发送 0x123456

◇ 在接收模式下  
如果收到数据 0x123456

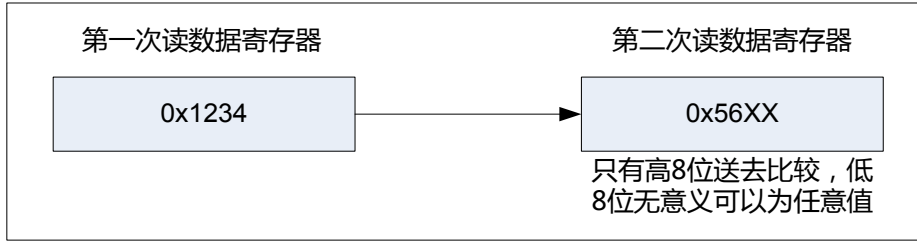


图 29-18 接收 0x123456

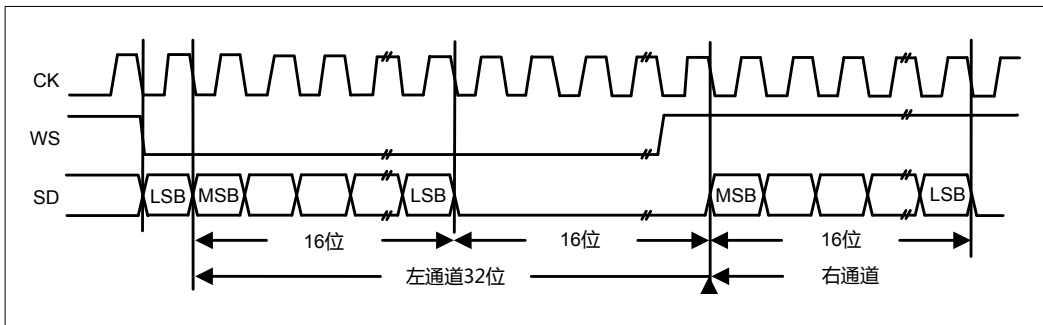


图 29-19 I2S 飞利浦标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

当在 I2S 配置阶段选择将 16 位数据帧扩展到 32 位通道帧时，只需要访问一次 SPI\_DATA 寄存器。其余 16 位由硬件强制为 0x0000，以将数据扩展为 32 位格式。

如果要传输的数据或接收的数据是 0x4567（0x4567 0000 扩展到 32 位），则需要执行下图所示的操作。



图 29-20 16 位数据帧扩展到 32 位通道帧的示例

### 29.6.1.2 MSB对齐标准

对于该标准，WS 信号与第一个数据位（MSB）同时生成。

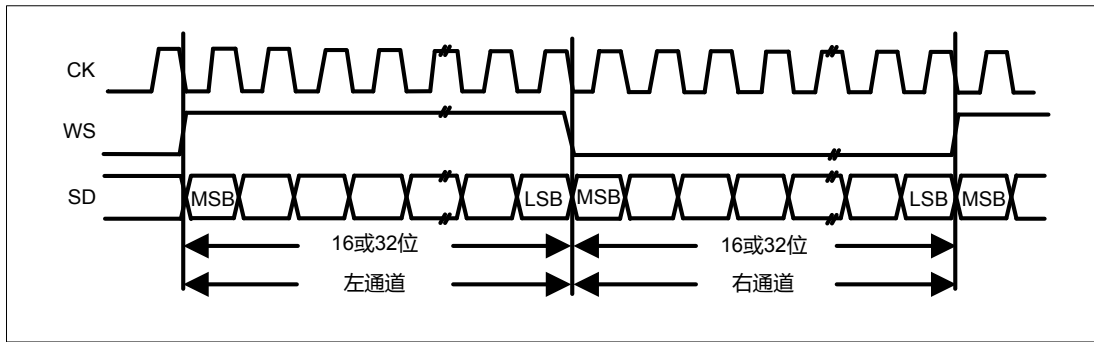


图 29-21 MSB 对齐标准波形（16/32 位数据帧，CPOL = 0）

数据在 CK 的下降沿（发送器）被锁存，并在上升沿（对于接收器）读取。

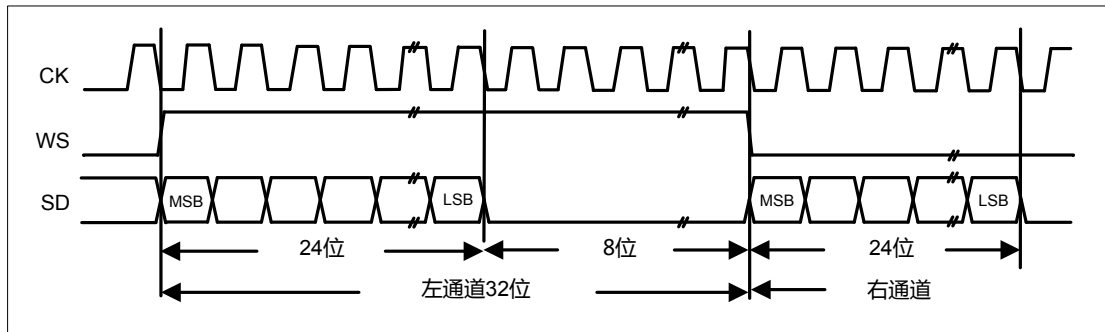


图 29-22 MSB 对齐标准波形（24 位数据帧，CPOL = 0）

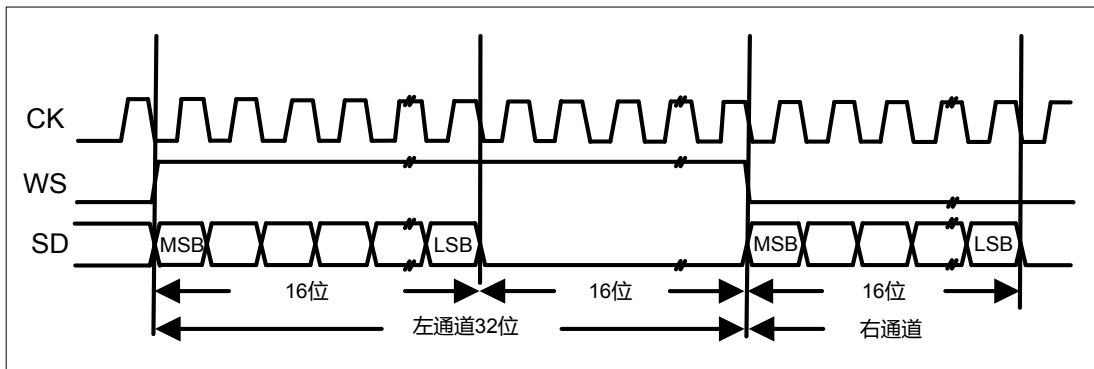


图 29-23 MSB 对齐标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

### 29.6.1.3 LSB对齐标准

该标准类似于 MSB 对齐标准（16 位和 32 位通道帧格式没有区别）。

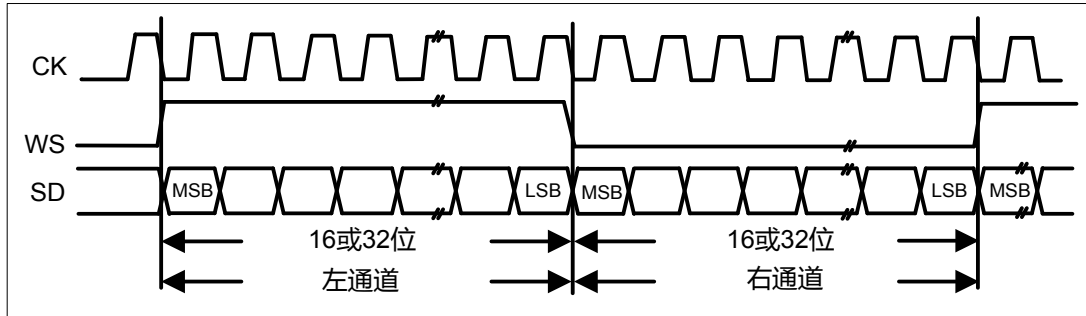


图 29-24 LSB 对齐标准波形（16/32 位数据帧，CPOL = 0）

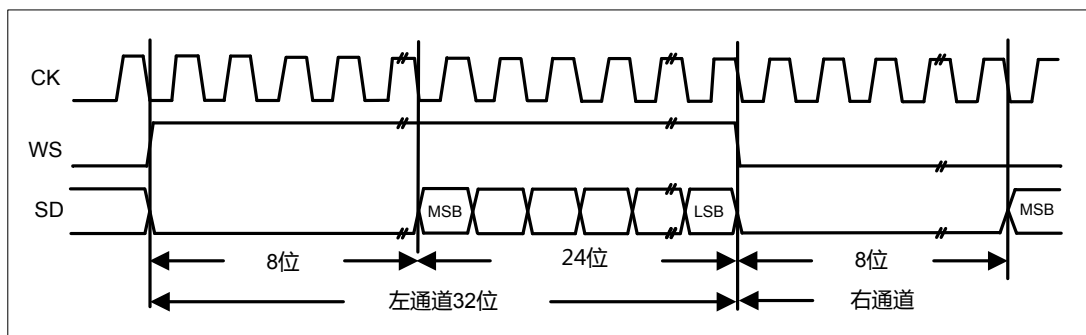


图 29-25 LSB 对齐标准波形（24 位数据帧，CPOL = 0）

◇ 在传输模式下

如果必须发送数据 0x123456，则软件或 DMA 需要对 SPI\_DATA 寄存器进行两次写操作。操作如下所示。

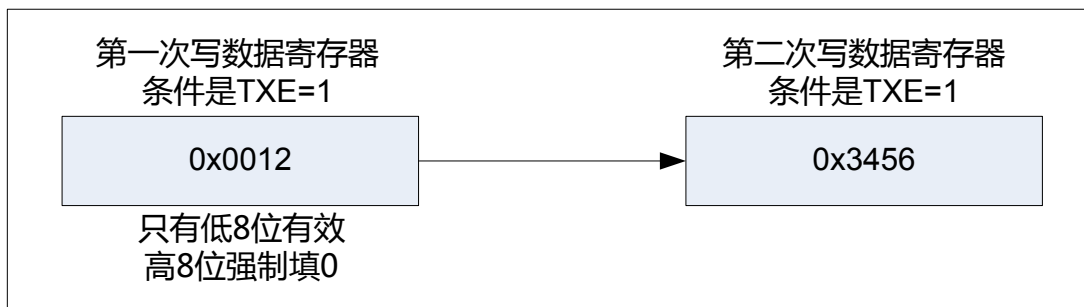


图 29-26 传输 0x123456 所需的操作

◇ 在接收模式下

如果接收到数据 0x123456，则每个 RXE 事件都需要从 SPI\_DATA 寄存器执行两次连续的读操作。

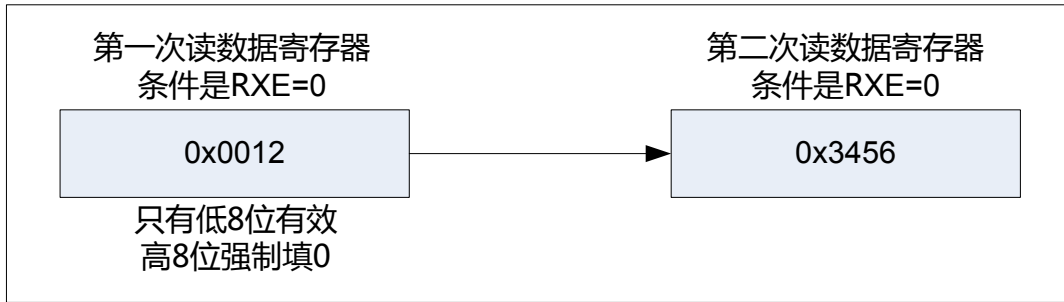


图 29-27 接收 0x123456 所需的操作

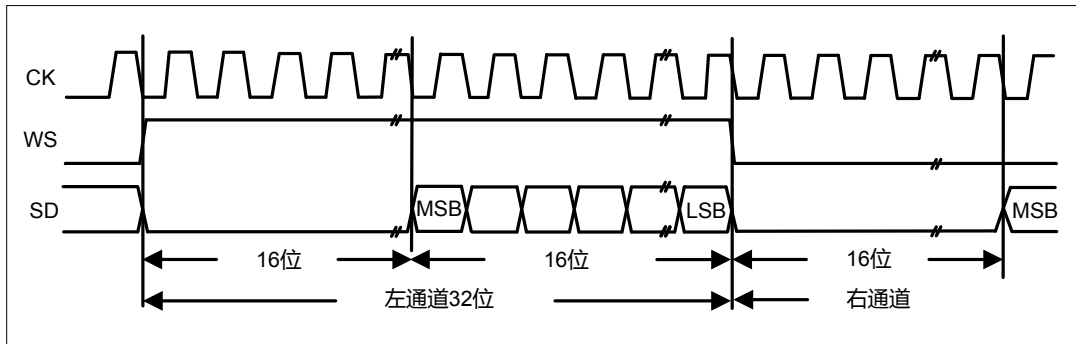


图 29-28 LSB 对齐标准波形（16 位数据帧扩展到 32 位通道帧，CPOL = 0）

在 I2S 配置阶段选择扩展到 32 位通道帧的 16 位数据帧时，只需要访问一次 SPI\_DATA 寄存器。其余 16 位由硬件强制为 0x0000，以将数据扩展为 32 位格式。在这种情况下，它对应于半字 MSB。

如果要发送的数据或接收的数据是 0x4567（0x0000 4567 扩展到 32 位），则需要执行下图所示的操作。



图 29-29 16 位数据帧扩展到 32 位通道帧的示例

在传输模式下，当发生 TXE 事件时，应用程序必须写入要传输的数据（在本例中为 0x4567）。首先发送 0x0000 字段（32 位扩展）。

在接收模式下，一旦接收到有效的半字（而不是 0x0000 字段），RXE 就会被清零。

以这种方式，在两次写入或读取操作之间提供更多时间以防止下溢或溢出条件。

### 29.6.1.4 PCM标准

对于 PCM 标准，不需要使用信道侧信息。有两种 PCM 模式（短帧和长帧）可用，使用 SPI\_I2SCFG.PCMSYNC 位进行配置。

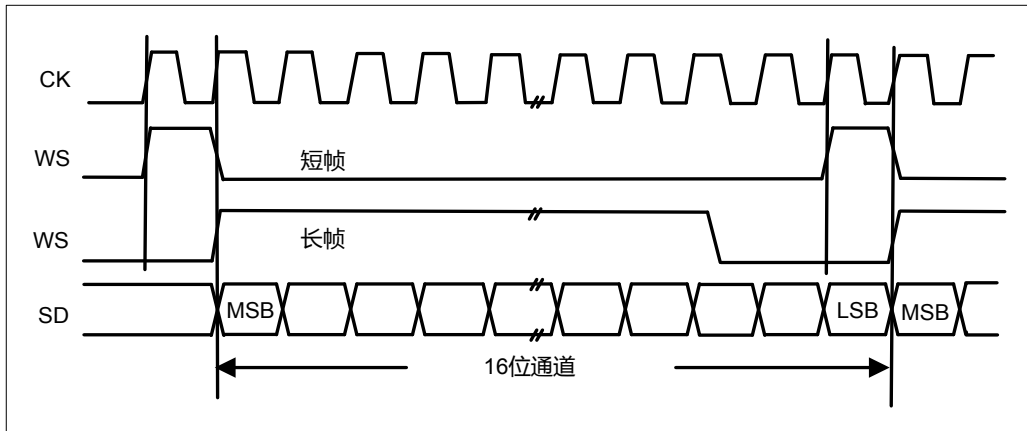


图 29-30 PCM 标准波形（16 位）

对于长帧同步，WS 信号置入时间固定为 13 位。

对于短帧同步，WS 同步信号只有一个周期。

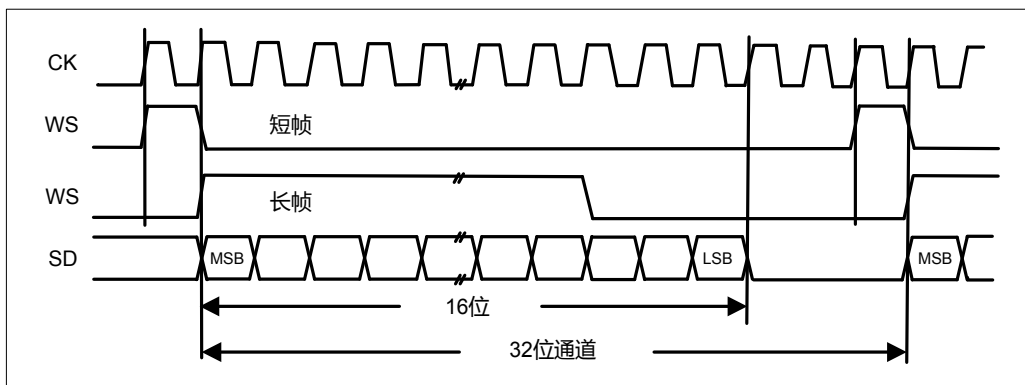


图 29-31 PCM 标准波形（16 位数据帧扩展到 32 位通道帧）

注：对于两种同步（短帧和长帧），需要指定两个连续数据（以及两个同步信号）之间的位数（SPI\_I2SCFG.DATLEN 和 SPI\_I2SCFG.CHLEN 位）。

### 29.6.2 时钟产生器

I2S 比特率决定 I2S 数据线上的数据流和 I2S 时钟信号频率。

I2S 比特率=每通道的比特数×通道数×采样音频

对于 16 位音频，左右声道，I2S 比特率计算如下：

$$\text{I2S 比特率} = 16 \times 2 \times F_s$$

如果数据包长度为 32 位宽，则 I2S 比特率=32×2×F<sub>s</sub>。

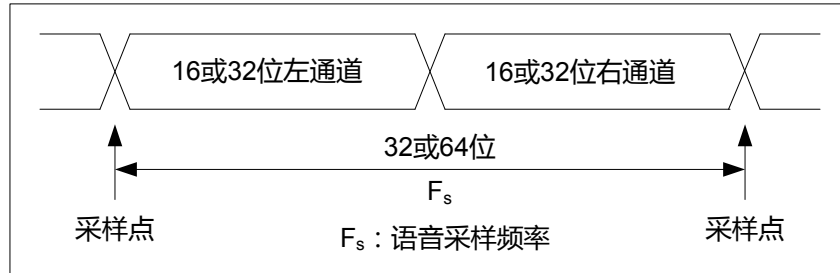


图 29-32 音频采样频率定义

音频采样频率可以是 192kHz, 96kHz, 48kHz, 44.1kHz, 32kHz, 22.05kHz, 16kHz, 11.025kHz 或 8kHz (或该范围内的任何其他值)。为了达到所需频率，需要根据以下公式对线性分频器进行编程：

在 I2S, MSB, LSB 模式中，CH 为 2。

在 PCM 模式下，CH 为 1。

当产生主时钟时 (SPI\_I2SPR 寄存器中的 MCKOE 位置 1)：

$$\text{当通道帧为 16 位宽时, } FS = \text{I2S\_CLK} / [ (16 * 2) * ((CH * \text{I2SDIV}) + \text{ODD}) * 8 ]$$

$$\text{当通道帧为 32 位宽时, } FS = \text{I2S\_CLK} / [ (32 * 2) * ((CH * \text{I2SDIV}) + \text{ODD}) * 4 ]$$

禁用主时钟时 (SPI\_I2SPR 寄存器中的 MCKOE 位清零)：

$$\text{当通道帧为 16 位宽时, } FS = \text{I2S\_CLK} / [ (16 * 2) * ((CH * \text{I2SDIV}) + \text{ODD}) ]$$

$$\text{当通道帧为 32 位宽时, } FS = \text{I2S\_CLK} / [ (32 * 2) * ((CH * \text{I2SDIV}) + \text{ODD}) ]$$

下表提供了不同时钟配置的示例精度值。

APB_CLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs(Hz)	Real fs (Hz)	Error
32	16	5	0	No	96000	100000	4.1667%
32	32	2	0	No	96000	100000	4.1667%
32	16	10	1	No	48000	47619	0.7937%
32	32	5	0	No	48000	50000	4.1667%
32	16	11	1	No	44100	43478	1.4098%
32	32	5	1	No	44100	45454	3.0715%

APB_CLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs(Hz)	Real fs (Hz)	Error
32	16	15	1	No	32000	32258	0.8065%
32	32	8	0	No	32000	31250	2.3430%
32	16	22	1	No	22050	22222	0.7811%
32	32	11	1	No	22050	21739	1.4098%
32	16	31	1	No	16000	15873	0.7937%
32	32	15	1	No	16000	16129	0.8065%
32	16	45	1	No	11025	10989	0.3264%
32	32	22	1	No	11025	11111	0.7811%
32	16	62	1	No	8000	8000	0.0000%
32	32	31	1	No	8000	7936	0.7937%
32	16	2	0	Yes	32000	31250	2.3430%
32	32	2	0	Yes	32000	31250	2.3430%
32	16	3	0	Yes	22050	20833	5.5170%
32	32	3	0	Yes	22050	20833	5.5170%
32	16	4	0	Yes	16000	15625	2.3428%
32	32	4	0	Yes	16000	15625	2.3428%
32	16	5	1	Yes	11025	11363	3.0715%
32	32	5	1	Yes	11025	11363	3.0715%
32	16	8	0	Yes	8000	7812	2.3428%
32	32	8	0	Yes	8000	7812	2.3428%

表 29-1 不同时钟配置的示例精度值

注：其他配置可能允许最佳时钟精度。

### 29.6.3 I2S主机模式

使用 I2S 功能时只能工作在主机模式。这意味着在 CK 引脚上生成串行时钟以及字选择信号 WS。主时钟 (MCK) 可以输出与否，由 SPI\_I2SPR.MCKOE 位控制。

#### 29.6.3.1 设置流程

1. 选择 SPI\_I2SPR.I2SDIV 位，定义串行时钟波特率，以达到正确的音频采样频率。同时必须定义 SPI\_I2SPR.ODD 位。
2. 选择 CKPOL 位以定义通信时钟的稳定电平。如果需要将主时钟 MCK 提供给外部 DAC/ADC 音频组件，则将 SPI\_I2SPR.MCKOE 位置 1 (I2SDIV 和 ODD 值应根据 MCK 输出的状态进行计算，更多详细信息，请参见：时钟产生器)。
3. 将 SPI\_I2SCFG.I2SMOD 位置 1 以激活 I2S 功能，并通过 SPI\_I2SCFG.I2SSTD 和 SPI\_I2SCFG.PCMSYNC 位选择 I2S 标准，通过 SPI\_I2SCFG.DATLEN 位选择数据长度，通过配置 SPI\_I2SCFG.CHLEN 位来选择通道长度。通过 SPI\_I2SCFG.I2SCFG 位选择方向 (发送器或接收器)。



4. 如果需要，通过写 SPI\_CON2 寄存器选择 DMA 功能。
5. 如果需要，通过写 SPI\_IER 寄存器选择所有可能的中断源。
6. 必须设置 SPI\_I2SCFG.I2SE 位。WS 和 CK 配置为输出模式。如果 SPI\_I2SPR.MCKOE 位置 1，MCK 也是输出。

### 29.6.3.2 传输序列

当半字写入 TX FIFO 时，传输序列开始。

写入 TX FIFO 的第一个数据应对应于左声道数据。当左声道的数据写入 TX FIFO 后必须紧跟着写入对应于右声道数据。CHSIDE 标志指示当前发送的声道。

必须将全帧视为左右声道数据传输。不会有仅发送左声道的部分帧。

有关写入操作的更多详细信息，具体取决于所选的 I2S 标准模式，请参见：音频协议。

为了确保连续的音频数据传输，必须在当前传输结束之前将对 SPI\_DATA 寄存器写入下一个要传输的数据。若要禁用 I2S，必须等待 TXE = 1 且 BUSY = 0 时，通过清除 SPI\_I2SCFG.I2SE 来关闭 I2S。

### 29.6.3.3 接收序列

操作模式与传输模式相同，除了第 3 点（参见：I2S 主机模式中描述的过程）之外，其中配置应通过 SPI\_I2SCFG.I2SCFG 位设置接收模式。

RX FIFO 满时，RXF 标志置 1，如果 SPI\_IER.RXFIE 位置 1，则产生中断。根据数据和通道长度配置，右声道或左声道接收的音频值可能是由于进入 RX FIFO 的一次或两次接收所致。

通过读 SPI\_DATA 寄存器来执行清零 RXF 位。

有关根据所选 I2S 标准模式进行读取操作的更多详细信息，请参见：音频协议。

如果在尚未读取先前接收的数据的同时接收到新的数据，则生成溢出并设置 RXOV 标志。如果 SPI\_IER 寄存器中的 RXOVIE 位置 1，则会产生中断以指示错误。

### 29.6.4 I2S 状态标志

为应用程序提供了六个状态标志，以完全监视 I2S 总线的状态。

#### 29.6.4.1 发送 FIFO 空标志 (TXE)

设置时，TXE 标志指示发送 FIFO 为空，并且下一个要发送的数据可以加载到 FIFO 中。通过写 SPI\_DATA 寄存器清零 TXE 标志。

#### 29.6.4.2 发送 FIFO 满标志 (TXF)

设置时，TXF 标志指示发送 FIFO 已满，并且下一个要发送的数据无法加载到 FIFO 中。TXF 标志由要发送的下一个数据清除。

#### 29.6.4.3 接收 FIFO 空标志 (RXE)

设置时，RXE 标志指示接收 FIFO 为空，并且下一个要接收的数据可以加载到 FIFO 中。

RXE 标志由下一个数据接收到 FIFO 中清除。

#### 29.6.4.4 接收FIFO满标志 (RXF)

设置后, RXF 标志指示接收 FIFO 已满, 并且下一个要接收的数据无法加载到 FIFO 中。通过对 SPI\_DATA 寄存器的读访问来清除 RXF 标志。

#### 29.6.4.5 BUSY标志 (BUSY)

BUSY 标志由硬件置位和清除 (写入此标志无效)。当 BUSY 标志置 1 时, 表示 I2S 上正在进行数据传输 (I2S 总线忙)。BUSY 标志可以用于检测传输的结束, 从而防止最后一次传输损坏。

在以下任一条件下清除 BUSY 标志:

- ◇ 当 I2S 正确地禁用
- ◇ 传输完成时
- ◇ 在发送模式下, BUSY 标志在所有传输期间保持高电平

注: 建议始终使用 TXTH 和 RXTH 标志 (而不是 BUSY 标志) 来处理数据传输或接收操作。

#### 29.6.4.6 声道标志 (CHSIDE)

此标志是提供使用者判断此时总线上正在传输的声道为左声道还右声道使用, 只有在 BUSY 标志被设置时有效。该标志在 PCM 标准中没有意义 (短帧和长帧模式)。

#### 29.6.5 I2S错误标志

I2S 单元有四个错误标志。

##### 29.6.5.1 发送FIFO溢出标志 (TXOV)

发生发送溢出条件的情况是发送 FIFO 已满, 但用户正在执行写操作。在这种情况下, 新写入的数据不会加载到发送 FIFO 中。通过对 SPI\_ICR 寄存器的写访问来清除 TXOV 位, 或者在新传输开始时它将自动清零。

##### 29.6.5.2 发送FIFO下溢标志 (TXUD)

发生发送下溢运行条件是当发送 FIFO 为空时, 但主机想要从发送 FIFO 拿取数据。在这种情况下, 发送 FIFO 的内容不会被发送至总线上。通过对 SPI\_ICR 寄存器的写访问来清除 TXUD 位, 或者在新的传输开始时它将自动清零。

##### 29.6.5.3 接收FIFO溢出标志 (RXOV)

当完成下一个数据帧的接收而接收 FIFO 的前 16 帧的读操作尚未完成时 (设置了 RXF 标志的情况), 发生接收溢出条件。

在这种情况下, 接收 FIFO 的内容不会将接收的新数据存入。SPI\_DATA 寄存器的读操作返回先前接收的帧。所有其他在 RXF 标志设置的情况下, 接收的新数据都将丢失。

通过对 SPI\_ICR 寄存器的写访问来清除 RXOV 位, 或者在新的传输开始时它将自动清零。

#### 29.6.5.4 接收FIFO下溢标志 (RXUD)

当接收 FIFO 为空时，但用户正在执行读操作，会发生接收下溢条件。在这种情况下，读取操作不会从接收 FIFO 中读到有效的数据。

通过对 SPI\_ICR 寄存器的写访问来清除 RXUD 位，或者在新的传输开始时它将自动清零。

### 29.7 特殊功能寄存器

#### 29.7.1 寄存器列表

外设寄存器可支持半字（16 位）或字（32 位）访问。

SPI 寄存器列表			
寄存器名称	偏移地址	类型	寄存器描述
SPI0 基地址: 4000_6000 <sub>H</sub> SPI1 基地址: 4000_6400 <sub>H</sub>			
SPI_CON1	0000 <sub>H</sub>	R/W	SPI 控制寄存器 1
SPI_CON2	0004 <sub>H</sub>	R/W	SPI 控制寄存器 2
SPI_STAT	0008 <sub>H</sub>	R	SPI 状态寄存器
SPI_DATA	000C <sub>H</sub>	R/W	SPI 数据寄存器
SPI_CRCPOLY	0010 <sub>H</sub>	R/W	SPI CRC 多项式寄存器
SPI_RXCRC	0014 <sub>H</sub>	R	SPI RX CRC 寄存器
SPI_TXCRC	0018 <sub>H</sub>	R	SPI TX CRC 寄存器
SPI_I2SCFG	001C <sub>H</sub>	R/W	SPI I2S 配置寄存器
SPI_I2SPR	0020 <sub>H</sub>	R/W	SPI I2S 预除频寄存器
SPI_IER	0024 <sub>H</sub>	W1	SPI 中断启用寄存器
SPI_IDR	0028 <sub>H</sub>	W1	SPI 中断禁用寄存器
SPI_IVS	002C <sub>H</sub>	R	SPI 中断有效状态寄存器
SPI_RIF	0030 <sub>H</sub>	R	SPI 原始中断标志状态寄存器
SPI_IFM	0034 <sub>H</sub>	R	SPI 中断标志屏蔽状态寄存器
SPI_ICR	0038 <sub>H</sub>	C_W1	SPI 中断清除寄存器

## 29.7.2 寄存器描述

### 29.7.2.1 SPI控制寄存器 1 (SPI\_CON1)

SPI 控制寄存器 1 (SPI_CON1)																															
偏移地址: 00H																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BIDEN	BIDOEN	CRCEN	NXTCRC	FLEN	RXO	SSEN	SSOUT	LSBFST	SPIEN	BAUD			MSTREN	CPOL	CPHA

Reserved	Bit 31-16	—	保留
BIDEN	Bit 15	R/W	<p><b>启用双向数据模式</b></p> <p>该位使用通用单双向数据线实现半双工通信。双向模式激活时，保持 RXO 位清零。</p> <p>0: 选择双线单向通信数据模式</p> <p>1: 选择单线双向通信数据模式</p> <p>注：该位不用于 I2S 模式。</p>
BIDOEN	Bit 14	R/W	<p><b>双向模式下的输出使能</b></p> <p>该位与 BIDEN 位组合在一起选择双向模式下的传输方向</p> <p>0: 禁用输出（仅接收模式）</p> <p>1: 启用输出（仅发送模式）</p> <p>注：在主机模式下，使用 MOSI 引脚，在从机模式，使用 MISO 引脚。在 I2S 模式下不使用该位。</p>
CRCEN	Bit 13	R/W	<p><b>硬件 CRC 计算使能</b></p> <p>0: 禁用 CRC 计算</p> <p>1: 启用 CRC 计算</p> <p>注：为确保正确操作，只应在禁止 SPI (SPIEN =“0”) 时对此位执行写操作。I2S 模式中不使用该位。</p>
NXTCRC	Bit 12	R/W	<p><b>下一次传输 CRC</b></p> <p>0: 数据传输结束时不传输 CRC</p> <p>1: 数据传输结束时传输 CRC</p> <p>注：若当前传输需要传送 CRC 时，在写入数据后即可将该位置 1。若为接收时，则在写入无效数据后将该位置 1。I2S 模式中不使用该位。</p>
FLEN	Bit 11	R/W	<p><b>数据帧长度</b></p> <p>0: 选择 8 位数据帧格式进行发送/接收</p> <p>1: 选择 16 位数据帧格式进行发送/接收</p> <p>注：为确保正确操作，只应在禁止 SPI (SPIEN =“0”) 时对此位执行写操作。I2S 模式中不使用该</p>

			位。
RXO	Bit 10	R/W	<p><b>启用仅接收模式</b> 该位允许使用单个单向线进行单工通信，以专门接收数据。当仅接收模式处于活动状态时，保持 <b>BIDEN</b> 位清零。该位在多区域系统中也很有用，在该系统中不访问此特定从机，来自被访问从机的输出不会被破坏。 0: 全双工（发送和接收） 1: 禁用输出（仅接收模式） 注：该位不用于 I2S 模式。</p>
SSEN	Bit 9	R/W	<p><b>软件从机选择管理</b> 当 <b>SSEN</b> 位置 1 时，<b>NSS</b> 引脚输入将被 <b>SSOUT</b> 位的值替换。 0: 禁用软件从机选择管理 1: 启用软件从机选择管理 注：该位不用于 I2S 模式和 SPI 兼容模式。</p>
SSOUT	Bit 8	R/W	<p><b>软件从机选择</b> 0: <b>NSS</b> 引脚输入为 0 1: <b>NSS</b> 引脚输入为 1 仅当 <b>SSEN</b> 位置 1 时，该位才有效。此位的值将作用到 <b>NSS</b> 引脚上，并忽略 <b>NSS</b> 引脚的 IO 值。 注：该位不用于 I2S 模式和 SPI 兼容模式。</p>
LSBFST	Bit 7	R/W	<p><b>先发送 LSB</b> 0: 首先使用 <b>MSB</b> 发送/接收数据 1: 首先使用 <b>LSB</b> 发送/接收数据 注 1: 通信正在进行时，不应更改此位。 注 2: 该位不用于 I2S 模式和 SPI 兼容模式。</p>
SPIEN	Bit 6	R/W	<p><b>SPI 启用</b> 0: 禁用 <b>SPI</b> 外设 1: 启用 <b>SPI</b> 外设 注：该位不用于 I2S 模式。</p>
BAUD	Bit 5-3	R/W	<p><b>波特率控制</b> 000: <math>F_{PCLK} / 2</math> 001: <math>F_{PCLK} / 4</math> 010: <math>F_{PCLK} / 8</math> 011: <math>F_{PCLK} / 16</math> 100: <math>F_{PCLK} / 32</math> 101: <math>F_{PCLK} / 64</math> 110: <math>F_{PCLK} / 128</math> 111: <math>F_{PCLK} / 256</math> 注：通信正在进行时，不应更改这些位。在 I2S 模式下不使用该位。</p>
MSTREN	Bit 2	R/W	<p><b>主机选择</b> 0: 从机模式</p>

			<p>1: 主机模式</p> <p>注: 通信正在进行时, 不应更改此位。在 I2S 模式下不使用该位。</p>
CPOL	Bit 1	R/W	<p><b>时钟极性</b></p> <p>0: 在空闲的状态下, SCK 引脚保持低电平输出</p> <p>1: 在空闲的状态下, SCK 引脚保持高电平输出</p> <p>注: 通信正在进行时, 不应更改此位。该位不用于 I2S 模式和 SPI 兼容模式。</p>
CPHA	Bit 0	R/W	<p><b>时钟相位</b></p> <p>0: 从第一个时钟边沿开始采样数据</p> <p>1: 从第二个时钟边沿开始采样数据</p> <p>注: 通信正在进行时, 不应更改此位。该位不用于 I2S 模式和 SPI 兼容模式。</p>

### 29.7.2.2 SPI控制寄存器 2 (SPI\_CON2)

SPI 控制寄存器 2 (SPI_CON2)																																
偏移地址: 04 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																RXFTH	TXFTH	Reserved										FRF	NSSP	NSSOE	TXDMA	RXDMA

Reserved	Bit 31-16	—	保留
RXFTH	Bit15-14	R/W	<p><b>接收 FIFO 触发阈值</b> 这用于选择接收 FIFO 生成接收 FIFO 触发阈值标志的阈值。 00: 当接收 FIFO 中有 1 个字符时触发 01: 当接收 FIFO 中有 4 个字符时触发 10: 当接收 FIFO 中有 8 个字符时触发 11: 当接收 FIFO 中有 14 个字符时触发 注: 如果 RX DMA 使能, 该位也提供 RX DMA 请求的条件。</p>
TXFTH	Bit 13-12	R/W	<p><b>发送 FIFO 触发阈值</b> 这用于选择发送 FIFO 产生发送 FIFO 触发阈值标志的阈值。 00: 当发送 FIFO 空时触发 01: 当发送 FIFO 中有 2 个字符时触发 10: 当发送 FIFO 中有 4 个字符时触发 11: 当发送 FIFO 中有 8 个字符时触发 注: 如果 TX DMA 使能, 该位也提供 TX DMA 请求的条件。</p>
Reserved	Bit 11-5	—	保留
FRF	Bit 4	R/W	<p><b>通信模式选择</b> 0: SPI 标准模式 (摩托罗拉通信协议) 1: SPI 兼容模式 (兼容 TI 通信协议) 注: 只有在禁止 SPI (SPIEN = 0) 时才能写入该位。在 I2S 模式下不使用该位。</p>
NSSP	Bit 3	R/W	<p><b>NSS 脉冲管理</b> 该位仅用于主机模式。它允许 SPI 在进行连续传输时在两个连续数据之间产生 NSS 脉冲。在单次数据传输的情况下, 它会在传输后强制 NSS 引脚为高电平。如果 CPHA = 1 或 FRF = 1, 则没有意义。 0: 没有 NSS 脉冲 1: 产生 NSS 脉冲 注 1: 只有在禁止 SPI (SPIEN = 0) 时才能写入</p>

			该位。 注 2: 该位不用于 I2S 模式和 SPI 兼容模式。
NSSOE	Bit 2	R/W	<b>NSS 引脚输出使能</b> 0: 在主机模式下禁止 NSS 输出, 可在多主模式配置下工作 1: 在主机模式下使能 NSS 输出, 不能在多主模式环境下工作 注: 该位不用于 I2S 模式和 SPI 兼容模式。
TXDMA	Bit 1	R/W	<b>TX DMA 使能</b> 该位置 1 时, 只要 TXTH 标志置 1, 就会产生 DMA 请求。 0: 禁用 TX DMA 1: 启用 TX DMA
RXDMA	Bit 0	R/W	<b>RX DMA 使能</b> 该位置 1 时, 只要 RXTH 标志置 1, 就会产生 DMA 请求。 0: 禁用 RX DMA 1: 启用 RX DMA



29.7.2.3 SPI状态寄存器 (SPI\_STAT)

SPI 状态寄存器(SPI_STAT)																																
偏移地址: 08 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_01010001 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			RXFLV				Reserved			TXFLV						BUSY	CHSIDE	Reserved	RXTH	RXUD	RXOV	RXF	RXE	Reserved				TXTH	TXUD	TXOV	TXF	TXE

Reserved	Bit 31-29	—	保留
RXFLV	Bit 28-24	R	<b>RX FIFO 的水平</b> 该位提供 RX FIFO 的水平值。
Reserved	Bit 23-21	—	保留
TXFLV	Bit 20-16	R	<b>TX FIFO 水平</b> 该位提供 TX FIFO 的水平值。
BUSY	Bit 15	R	<b>忙标志</b> 0: SPI (或 I2S) 不繁忙 1: SPI (或 I2S) 忙于通信 该标志由硬件设置和清除。
CHSIDE	Bit 14	R	<b>声道侧</b> 0: 当前正在发送或接收左声道 1: 当前正在传输或接收右声道 注: 该位不用于 SPI 模式。它在 PCM 模式下没有意义。
Reserved	Bit 13	—	保留
RXTH	Bit 12	R	<b>接收 FIFO 水平超出阈值</b> 该位提供接收FIFO水平超过阈值。 0: 接收 FIFO 水平没有超过阈值。 1: 接收FIFO水平超过阈值。
RXUD	Bit 11	R	<b>接收 FIFO 下溢</b> 0: 接收 FIFO 没有下溢 1: 接收FIFO下溢
RXOV	Bit 10	R	<b>接收 FIFO 溢出</b> 0: 接收 FIFO 没有溢出 1: 接收FIFO溢出
RXF	Bit 9	R	<b>接收 FIFO 满</b> 0: 接收 FIFO 没有满 1: 接收FIFO满
RXE	Bit 8	R	<b>接收 FIFO 空</b> 0: 接收 FIFO 没有空

			1: 接收FIFO空
Reserved	Bit 7-5	—	保留
TXTH	Bit 4	R	发送 FIFO 水平低于阈值 该位提供发送FIFO水平低于阈值。 0: 发送 FIFO 水平高于阈值。 1: 发送FIFO水平低过阈值。
TXUD	Bit 3	R	发送 FIFO 下溢 0: 发送 FIFO 没有下溢 1: 发送FIFO下溢
TXOV	Bit 2	R	发送 FIFO 溢出 0: 发送 FIFO 没有溢出 1: 发送FIFO溢出
TXF	Bit 1	R	发送 FIFO 满 0: 发送 FIFO 没有满 1: 发送FIFO满
TXE	Bit 0	R	发送 FIFO 空 0: 发送 FIFO 没有空 1: 发送FIFO空

#### 29.7.2.4 SPI数据寄存器 (SPI\_DATA)

SPI 数据寄存器(SPI_DATA)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DATA															

Reserved	Bit 31-16	—	保留
DATA	Bit 15-0	R/W	数据寄存器 收到或将要传输的数据 数据寄存器用作RX FIFO和TX FIFO之间的接口。 读取数据寄存器时, 访问RX FIFO, 而写入数据寄存器访问TX FIFO。 注: 数据始终是右对齐的。写入寄存器时忽略未使用的位, 读取寄存器时读取为0。RX阈值设置必须始终与当前使用的读取访问权限相对应。

### 29.7.2.5 SPI CRC多项式寄存器 (SPI\_CRCPOLY)

SPI CRC 多项式寄存器(SPI_CRCPOLY)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000111																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CRCPOLY															

Reserved	Bit 31-16	—	保留
CRCPOLY	Bit 15-0	R/W	<p><b>CRC多项式寄存器</b></p> <p>该寄存器包含CRC计算的多项式。</p> <p>CRC多项式 (0007h) 是该寄存器的复位值。可以根据需要配置另一个多项式。</p> <p>注1: 多项式值应仅为奇数。没有支持偶数。</p> <p>注2: 当使用I2S时, 这些位应配置为默认值。</p>

### 29.7.2.6 SPI RX CRC寄存器 (SPI\_RXCRC)

SPI RX CRC 寄存器(SPI_RXCRC)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXCRC															

Reserved	Bit 31-16	—	保留
RXCRC	Bit 15-0	R	<p><b>接收 CRC 值</b></p> <p>使能 CRC 计算后, RXCRC&lt;15:0&gt;位将包含后续接收字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时, 此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。数据帧长度设置为 8 位数据 (SPI_CON1 的 FLEN 位清零) 时, 仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。选择 16 位数据帧长度 (SPI_CON1 寄存器的 FLEN 位置 1) 时, 考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。</p> <p>注 1: 当 BUSY 标志置 1 时, 读取此寄存器可能返</p>

			回一个不正确的值。 注 2: 当使用 I2S 时, 这些位应配置为默认值。
--	--	--	--

29.7.2.7 SPI TX CRC寄存器 (SPI\_TXCRC)

SPI TX CRC 寄存器(SPI_TXCRC)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXCRC															

Reserved	Bit 31-16	—	保留
TXCRC	Bit 15-0	R	<p><b>发送 CRC 值</b></p> <p>使能 CRC 计算后, TXCRC&lt;15:0&gt;位将包含后续发送字节在计算后所得到的 CRC 值。当 SPI_CON1 寄存器中的 CRCEN 位写入 1 时, 此寄存器复位。CRC 通过 SPI_CRCPOLY 寄存器中编程的多项式连续计算。数据帧长度设置为 8 位数据 (SPI_CON1 的 FLEN 位清零) 时, 仅考虑 8 个 LSB 位。CRC 计算依据任意 CRC8 标准进行。选择 16 位数据帧长度 (SPI_CON1 寄存器的 FLEN 位置 1) 时, 考虑此寄存器的全部 16 个位。CRC 计算依据任意 CRC16 标准进行。</p> <p>注 1: 当 BUSY 标志置 1 时, 读取此寄存器可能返回一个不正确的值。</p> <p>注 2: 当使用 I2S 时, 这些位应配置为默认值。</p>

### 29.7.2.8 SPI I2S配置寄存器 (SPI\_I2SCFG)

SPI I2S 配置寄存器 (SPI_I2SCFG)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											I2SMOD		I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN	CHLEN									

Reserved	Bit 31-12	—	保留
I2SMOD	Bit 11	R/W	<b>I2S 模式选择</b> 0: 选择 SPI 模式 1: 选择 I2S 模式 注: 当 SPI 或 I2S 禁止时, 应配置该位为默认值。
I2SE	Bit 10	R/W	<b>I2S 启用</b> 0: 禁用 I2S 外设 1: 启用 I2S 外设 注: 该位不用于 SPI 模式。
I2SCFG	Bit 9-8	R/W	<b>I2S 模式配置</b> 0x: 保留 10: 发送 11: 接收 注: 当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。
PCMSYNC	Bit 7	R/W	<b>PCM 帧同步</b> 0: 短帧同步 1: 长帧同步 注: 仅当 I2SSTD = 11 (使用 PCM 标准) 时, 该位才有意义。它不用于 SPI 模式。
Reserved	Bit 6	—	保留
I2SSTD	Bit 5-4	R/W	<b>I2S 标准选择</b> 00: I2S 飞利浦标准 01: MSB 对齐标准 (左对齐) 10: LSB 对齐标准 (右对齐) 11: PCM 标准 注: 为了正确操作, 当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。
CKPOL	Bit 3	R/W	<b>非活动状态时钟极性</b> 0: I2S 时钟非活动状态为低电平 1: I2S 时钟非活动状态是高电平 注: 为了正确操作, 当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。CKPOL 位不会影响用

			于接收或发送 SD 和 WS 信号的 CK 边沿灵敏度。
DATLEN	Bit 2-1	R/W	<p><b>要传输的数据长度</b></p> <p>00: 16 位数据长度</p> <p>01: 24 位数据长度</p> <p>10: 32 位数据长度</p> <p>11: 不允许</p> <p>注: 为了正确操作, 当 I2S 禁止时应配置这些位为默认值它们不用于 SPI 模式。</p>
CHLEN	Bit 0	R/W	<p><b>通道长度</b> (每个音频通道的位数)</p> <p>0: 16 位宽</p> <p>1: 32 位宽</p> <p>当 DATLEN 大于 00 时, 通道长度需固定为 32 位。</p> <p>注: 为了正确操作, 当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。</p>

### 29.7.2.9 SPI I2S预分频寄存器 (SPI\_I2SPR)

SPI I2S 预分频寄存器 (SPI_I2SPR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EXTCKEN	MCKOE	ODD	I2SDIV																		

Reserved	Bit 31-11	—	保留
EXTCKEN	Bit 10	R/W	<b>外部 I2S 时钟使能</b> 0: 选择 APB 时钟 1: 选择外部时钟 注: 当 I2S 禁止时应配置该位为默认值。该位不用于 SPI 模式。
MCKOE	Bit 9	R/W	<b>主时钟输出使能</b> 0: 禁用主时钟输出 1: 启用主时钟输出 注: 当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。
ODD	Bit 8	R/W	<b>预分频器的奇数系数</b> 0: 实际分频器值 = I2SDIV * 2 1: 实际分频器值 = (I2SDIV * 2) + 1 注: 当 I2S 禁止时应配置该位为默认值。它不用于 SPI 模式。
I2SDIV	Bit 7-0	R/W	<b>I2S 线性预分频器</b> I2SDIV<7:0> = 0 是禁用值。 注: 当 I2S 禁止时应配置这些位为默认值。它们不用于 SPI 模式。

### 29.7.2.10 SPI中断启用寄存器 (SPI\_IER)

SPI 中断启用寄存器 (SPI_IER)																																
偏移地址: 24 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FREIE	MODFIE	CRCERRIE	Reserved				RXTHIE	RXUDIE	RXOVIE	RXFIE	Reserved				TXTHIE	TXUDIE	TXOVIE	Reserved	TXEIE

Reserved	Bit 31-19	—	保留
FREIE	Bit 18	W1	帧格式错误中断启用 0: 无效 1: 启用中断
MODFIE	Bit 17	W1	模式故障中断启用 0: 无效 1: 启用中断
CRCERRIE	Bit 16	W1	<b>CRC 错误中断启用</b> 0: 无效 1: 启用中断
Reserved	Bit 15-13	—	保留
RXTHIE	Bit 12	W1	接收 <b>FIFO</b> 超过阈值中断启用 0: 无效 1: 启用中断
RXUDIE	Bit 11	W1	接收 <b>FIFO</b> 欠载中断启用 0: 无效 1: 启用中断
RXOVIE	Bit 10	W1	接收 <b>FIFO</b> 溢出中断启用 0: 无效 1: 启用中断
RXFIE	Bit 9	W1	接收 <b>FIFO</b> 满中断启用 0: 无效 1: 启用中断
Reserved	Bit 8-5	—	保留
TXTHIE	Bit 4	W1	发送 <b>FIFO</b> 低于阈值中断启用 0: 无效 1: 启用中断
TXUDIE	Bit 3	W1	发送 <b>FIFO</b> 欠载中断启用 0: 无效 1: 启用中断
TXOVIE	Bit 2	W1	发送 <b>FIFO</b> 溢出中断启用 0: 无效



			1: 启用中断
Reserved	Bit 1	—	保留
TXEIE	Bit 0	W1	发送 <b>FIFO</b> 空中断启用 0: 无效 1: 启用中断

### 29.7.2.11 SPI中断禁用寄存器 (SPI\_IDR)

SPI 中断禁用寄存器(SPI_IDR)																																
偏移地址: 28 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FREID	MODFID	CRCERRID	Reserved				RXTHID	RXUDID	RXOVID	RXFID	Reserved				TXTHID	TXUDID	TXOVID	Reserved	TXEID

Reserved	Bit 31-19	—	保留
FREID	Bit 18	W1	帧格式错误中断禁用 0: 无效 1: 禁用中断
MODFID	Bit 17	W1	模式故障中断禁用 0: 无效 1: 禁用中断
CRCERRID	Bit 16	W1	<b>CRC 错误中断禁用</b> 0: 无效 1: 禁用中断
Reserved	Bit 15-13	—	保留
RXTHID	Bit 12	W1	接收 <b>FIFO</b> 超过阈值中断禁用 0: 无效 1: 禁用中断
RXUDID	Bit 11	W1	接收 <b>FIFO</b> 欠载中断禁用 0: 无效 1: 禁用中断
RXOVID	Bit 10	W1	接收 <b>FIFO</b> 溢出中断禁用 0: 无效 1: 禁用中断
RXFID	Bit 9	W1	接收 <b>FIFO</b> 满中断禁用 0: 无效 1: 禁用中断
Reserved	Bit 8-5	—	保留
TXTHID	Bit 4	W1	发送 <b>FIFO</b> 低于阈值中断禁用 0: 无效 1: 禁用中断
TXUDID	Bit 3	W1	发送 <b>FIFO</b> 欠载中断禁用 0: 无效 1: 禁用中断
TXOVID	Bit 2	W1	发送 <b>FIFO</b> 溢出中断禁用 0: 无效

			1: 禁用中断
Reserved	Bit 1	—	保留
TXEID	Bit 0	W1	发送 <b>FIFO</b> 空中断禁用 0: 无效 1: 禁用中断

29. 7. 2. 12 SPI中断有效状态寄存器 (SPI\_IVS)

SPI 中断有效状态寄存器 (SPI_IVS)																																
偏移地址: 2C <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FREIV	MODFIV	CRCERRIV	Reserved				RXTHIV	RXUDIV	RXOVIV	RXFIV	Reserved				TXTHIV	TXUDIV	TXOVIV	Reserved	TXEIV

Reserved	Bit 31-19	—	保留
FREIV	Bit 18	R	帧格式错误中断有效 0: 无效 1: 中断有效
MODFIV	Bit 17	R	模式故障中断有效 0: 无效 1: 中断有效
CRCERRIV	Bit 16	R	<b>CRC 错误中断有效</b> 0: 无效 1: 中断有效
Reserved	Bit 15-13	—	保留
RXTHIV	Bit 12	R	接收 <b>FIFO 超过阈值</b> 中断有效 0: 无效 1: 中断有效
RXUDIV	Bit 11	R	接收 <b>FIFO 欠载</b> 中断有效 0: 无效 1: 中断有效
RXOVIV	Bit 10	R	接收 <b>FIFO 溢出</b> 中断有效 0: 无效 1: 中断有效
RXFIV	Bit 9	R	接收 <b>FIFO 满</b> 中断有效 0: 无效 1: 中断有效
Reserved	Bit 8-5	—	保留
TXTHIV	Bit 4	R	发送 <b>FIFO 低于阈值</b> 中断有效 0: 无效 1: 中断有效
TXUDIV	Bit 3	R	发送 <b>FIFO 欠载</b> 中断有效 0: 无效 1: 中断有效
TXOVIV	Bit 2	R	发送 <b>FIFO 溢出</b> 中断有效 0: 无效

			1: 中断有效
Reserved	Bit 1	—	保留
TXEIV	Bit 0	R	发送 <b>FIFO</b> 空中断有效 0: 无效 1: 中断有效

29. 7. 2. 13 SPI原始中断标志状态寄存器 (SPI\_RIF)

SPI 原始中断标志状态寄存器 (SPI_RIF)																																
偏移地址: 30 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FRERI	MODFRI	CRCERRRI	Reserved				RXTHRI	RXUDRI	RXOVRI	RXFRI	Reserved				TXTHRI	TXUDRI	TXOVRI	Reserved	TXERI

Reserved	Bit 31-19	—	保留
FRERI	Bit 18	R	帧格式错误中断标志状态 0: 无效 1: 原始中断标志状态
MODFRI	Bit 17	R	模式故障中断标志状态 0: 无效 1: 原始中断标志状态
CRCERRRI	Bit 16	R	<b>CRC 错误中断标志状态</b> 0: 无效 1: 原始中断标志状态
Reserved	Bit 15-13	—	保留
RXTHRI	Bit 12	R	接收 <b>FIFO</b> 超过阈值中断标志状态 0: 无效 1: 原始中断标志状态
RXUDRI	Bit 11	R	接收 <b>FIFO</b> 欠载中断标志状态 0: 无效 1: 原始中断标志状态
RXOVRI	Bit 10	R	接收 <b>FIFO</b> 溢出中断标志状态 0: 无效 1: 原始中断标志状态
RXFRI	Bit 9	R	接收 <b>FIFO</b> 满中断标志状态 0: 无效 1: 原始中断标志状态
Reserved	Bit 8-5	—	保留
TXTHRI	Bit 4	R	发送 <b>FIFO</b> 低于阈值中断标志状态 0: 无效 1: 原始中断标志状态
TXUDRI	Bit 3	R	发送 <b>FIFO</b> 欠载中断标志状态 0: 无效 1: 原始中断标志状态
TXOVRI	Bit 2	R	发送 <b>FIFO</b> 溢出中断标志状态 0: 无效

			1: 原始中断标志状态
Reserved	Bit 1	—	保留
TXERI	Bit 0	R	发送 <b>FIFO</b> 空中断标志状态 0: 无效 1: 原始中断标志状态

29. 7. 2. 14 SPI中断标志屏蔽状态寄存器 (SPI\_IFM)

SPI 中断标志屏蔽状态寄存器 (SPI_IFM)																																
偏移地址: 34 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FREFM	MODFFM	CRCERRFM	Reserved				RXTHFM	RXUDFM	RXOVFM	RXFFM	Reserved				TXTHFM	TXUDFM	TXOVFM	Reserved	TXEFM

Reserved	Bit 31-19	—	保留
FREFM	Bit 18	R	帧格式错误中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
MODFFM	Bit 17	R	模式故障中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
CRCERRFM	Bit 16	R	<b>CRC 错误</b> 中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
Reserved	Bit 15-13	—	保留
RXTHFM	Bit 12	R	接收 <b>FIFO</b> 超过阈值中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXUDFM	Bit 11	R	接收 <b>FIFO</b> 欠载中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXOVFM	Bit 10	R	接收 <b>FIFO</b> 溢出中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
RXFFM	Bit 9	R	接收 <b>FIFO</b> 满中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
Reserved	Bit 8-5	—	保留
TXTHFM	Bit 4	R	发送 <b>FIFO</b> 低于阈值中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
TXUDFM	Bit 3	R	发送 <b>FIFO</b> 欠载中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态
TXOVFM	Bit 2	R	发送 <b>FIFO</b> 溢出中断标志屏蔽状态 0: 无效



			1: 中断标志屏蔽状态
Reserved	Bit 1	—	保留
TXEFM	Bit 0	R	发送 <b>FIFO</b> 空中断标志屏蔽状态 0: 无效 1: 中断标志屏蔽状态

29. 7. 2. 15 SPI中断清除寄存器 (SPI\_ICR)

SPI 中断清除寄存器 (SPI_ICR)																																
偏移地址: 38 <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													FREIC	MODFIC	CRCERRIC	Reserved				RXTHIC	RXUDIC	RXOVIC	RXFIC	Reserved				TXTHIC	TXUDIC	TXOVIC	Reserved	TXEIC

Reserved	Bit 31-19	—	保留
FREIC	Bit 18	C_W1	帧格式错误中断清除 0: 无效 1: 中断清除
MODFIC	Bit 17	C_W1	模式故障中断清除 0: 无效 1: 中断清除
CRCERRIC	Bit 16	C_W1	<b>CRC 错误</b> 中断清除 0: 无效 1: 中断清除
Reserved	Bit 15-13	—	保留
RXTHIC	Bit 12	C_W1	接收 <b>FIFO</b> 超过阈值中断清除 0: 无效 1: 中断清除
RXUDIC	Bit 11	C_W1	接收 <b>FIFO</b> 欠载中断清除 0: 无效 1: 中断清除
RXOVIC	Bit 10	C_W1	接收 <b>FIFO</b> 溢出中断清除 0: 无效 1: 中断清除
RXFIC	Bit 9	C_W1	接收 <b>FIFO</b> 满中断清除 0: 无效 1: 中断清除
Reserved	Bit 8-5	—	保留
TXTHIC	Bit 4	C_W1	发送 <b>FIFO</b> 低于阈值中断清除 0: 无效 1: 中断清除
TXUDIC	Bit 3	C_W1	发送 <b>FIFO</b> 欠载中断清除 0: 无效 1: 中断清除
TXOVIC	Bit 2	C_W1	发送 <b>FIFO</b> 溢出中断清除 0: 无效

			1: 中断清除
Reserved	Bit 1	—	保留
TXEIC	Bit 0	C_W1	发送 <b>FIFO</b> 空中断清除 0: 无效 1: 中断清除

## 第30章 通用异步收发器(UART)

### 30.1 概述

通用异步收发器 (UART) 提供了一个灵活的方式, 使 MCU 可以与外部设备通过工业标准 NRZ 的形式实现全双工异步串行数据通讯。UART 可以使用分数波特率发生器, 提供了超宽的波特率设置范围。

UART 支持异步通讯模式和半双工单线通讯。也支持 LIN(本地互连网络), 智能卡协议和 IrDA (红外数据协会) SIR ENDEC 规范和 modem 流控操作 (CTS<sub>n</sub>/RTS<sub>n</sub>) 同时还支持多机通讯方式。

可以使用 DMA 实现多缓冲区设置, 从而能够支持高速数据通讯

### 30.2 特性

- ◆ 全双工异步通信
- ◆ 16-byte 接收和发送 FIFO
- ◆ 兼容 16C550 标准
- ◆ 可软件控制接收 FIFO 触发点
- ◆ 通信波特率可设置
- ◆ 十七个中断源
- ◆ 可与 DMA 使用
  - ◇ 利用 DMA 功能将收/发字节缓冲到保留的 SRAM 空间
- ◆ 内置小数波特率发生器, 覆盖范围广, 不需要特定值的外部晶体
  - ◇ 在时钟频率为 48 MHz 下, 可编程收发波特率高达 3Mbps, 最低可达 732.4bps
  - ◇ 在时钟频率为 4 MHz 下, 可编程收发波特率高达 250Kbps, 最低可达 61bps
- ◆ 支持自动硬件流控制/流控制功能 (CTS<sub>n</sub>、RTS<sub>n</sub>), RTS<sub>n</sub> 控制流触发点可程序设计
  - ◇ Modem 硬件自动控制
  - ◇ RS485 发送使能控制
- ◆ 支持 CTS<sub>n</sub> 唤醒功能
- ◆ 支持 IrDA SIR 模式
  - ◇ 支持 3/16 位周期调制
- ◆ 支持 RS-485
  - ◇ 支持 9-位模式
  - ◇ 多处理器通信
- ◆ 完全可程序设计的串行接口特性
  - ◇ 可程序设计数据位个数, 即 5, 6, 7, 8, 9 位, 9 位用于 RS485 模式
  - ◇ 校验位, 奇、偶、无校验
  - ◇ 停止位长度可程序设计: 1, 2 位, 在智能卡模式中支持 0.5, 1.5 位

- ◇ 可设置高位在前或低位在前
- ◆ 单线半双工通讯
- ◆ 交换 TX/RX pin 配置
- ◆ LIN 主机的断开信号发送能力和 LIN 从机的断开信号检测能力
  - ◇ 将 UART 设置为 LIN 模式时, 有 13 位的断开信号发生器与断开信号检测功能
- ◆ 智能卡模式
  - ◇ 支持 ISO/IEC7816-3 标准定义的 T=0 和 T=1 智能卡异步协议
  - ◇ 智能卡使用的 1.5 停止位长度
- ◆ 噪声检测
- ◆ 支持 Modbus 协议

### 30.3 结构框图

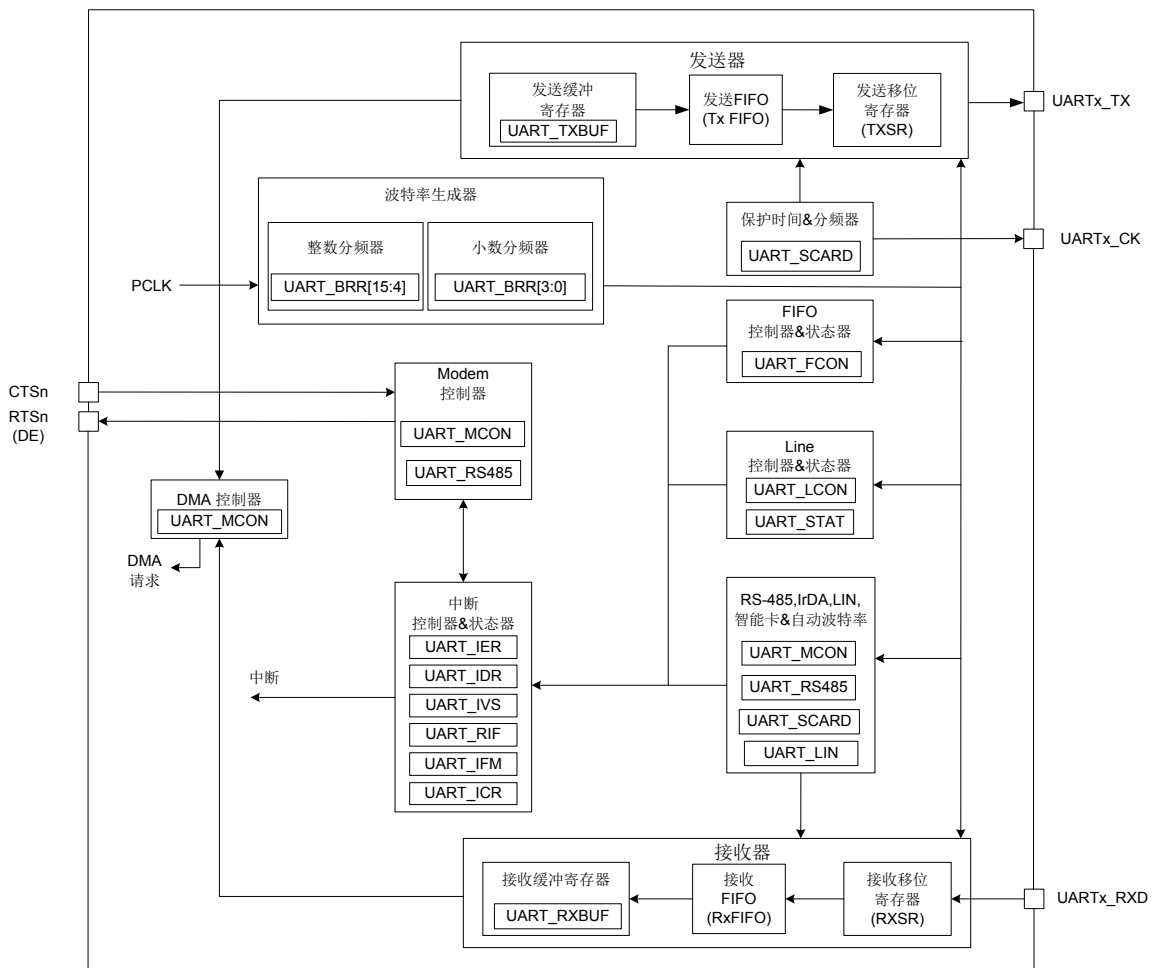


图 30-1 UART 框图

## 30.4 功能描述

接口与外部设备通过三个引脚相连。任何 UART 双向通讯要求最少有两个引脚：接收数据输入 (RX) 和发送数据输出 (TX)。

**RX:** 接收数据输入,是串行数据的输入口。使用过采样技术来完成数据采集,以区别输入数据和噪声。

**TX:** 数据发送输出。当发送器被禁止,输出脚回到其 I/O 口配置状态。当发送器被使能,但不发送数据时, TX 脚为高电平输出。在单线和智能卡模式中,这个引脚既用于发送数据也用于接收数据。

通过这些引脚, 串行数据用数据帧的形式发送和接收:

- ◇ 在发送和接收之前为空闲状态
- ◇ 起始位
- ◇ 数据可通过 UART\_LCON 寄存器的 MSB 位设定
- ◇ 1, 2 个停止位表明帧的结束 (0.5, 1.5 个停止位用于智能卡模式)
- ◇ 一个状态寄存器 (UART\_STAT)
- ◇ 分开的接收和发送数据寄存器 (UART\_RXBUF, UART\_TXBUF)
- ◇ 一个波特率寄存器 (UART\_BRR) 12 位整数和 4 位小数。
- ◇ 一个智能卡寄存器 (UART\_SCARD) 用于智能卡模式。
- ◇ 一个接收超时寄存器 (UART\_RTOR) 检测输入信号时间并产生中断

下面的引脚在智能卡模式中会用到:

**CK:** 时钟输出。智能卡模式中, CK 引脚会向智能卡提供时钟。

下列引脚用于支持硬件流控制模式:

**CTS<sub>n</sub>:** 低电平发送, 当高电平时作为发送阻塞信号。

**RTS<sub>n</sub>:** 请求发送, 表明 UART 已经准备好接收数据 (低电平的时候)。

下列引脚在 RS485 驱动使能控制的时候会用到:

**DE:** 驱动使能将外部收发器的发送模式激活。

注: DE 和 RTS<sub>n</sub> 共享同一个外部引脚。

### 30.4.1 数据长度

配置 LCON 寄存器中的 DLS 位可选择 8-5 位字长。

默认设置中，发送和接收的起始位都是低电平。而停止位都是高电平。

这个逻辑可以在 LCON 寄存器的 TXINV 与 RXINV 位设置为反向。

- ◇ 8 位字符宽度：DLS<1:0>=00
- ◇ 7 位字符宽度：DLS<1:0>=01
- ◇ 6 位字符宽度：DLS<1:0>=10
- ◇ 5 位字符宽度：DLS<1:0>=11

注：第 9 位使用于 RS485 多处理器模式。

空闲符号被视为完全由'1'组成的完整的数据帧，后面跟着包含了数据的下一帧的开始位('1'的位数也包括了停止位的位数)。

断开符号被视为在一个帧周期内全部收到'0'（包括停止位期间，也是'0'）。在断开帧结束时，发送器会再插入 2 个停止位。

发送和接收由一个共享的波特率发生器驱动，当发送器和接收器的使能位分别置 1 时，分别为其产生时钟。

下面是每个字长的详细说明。

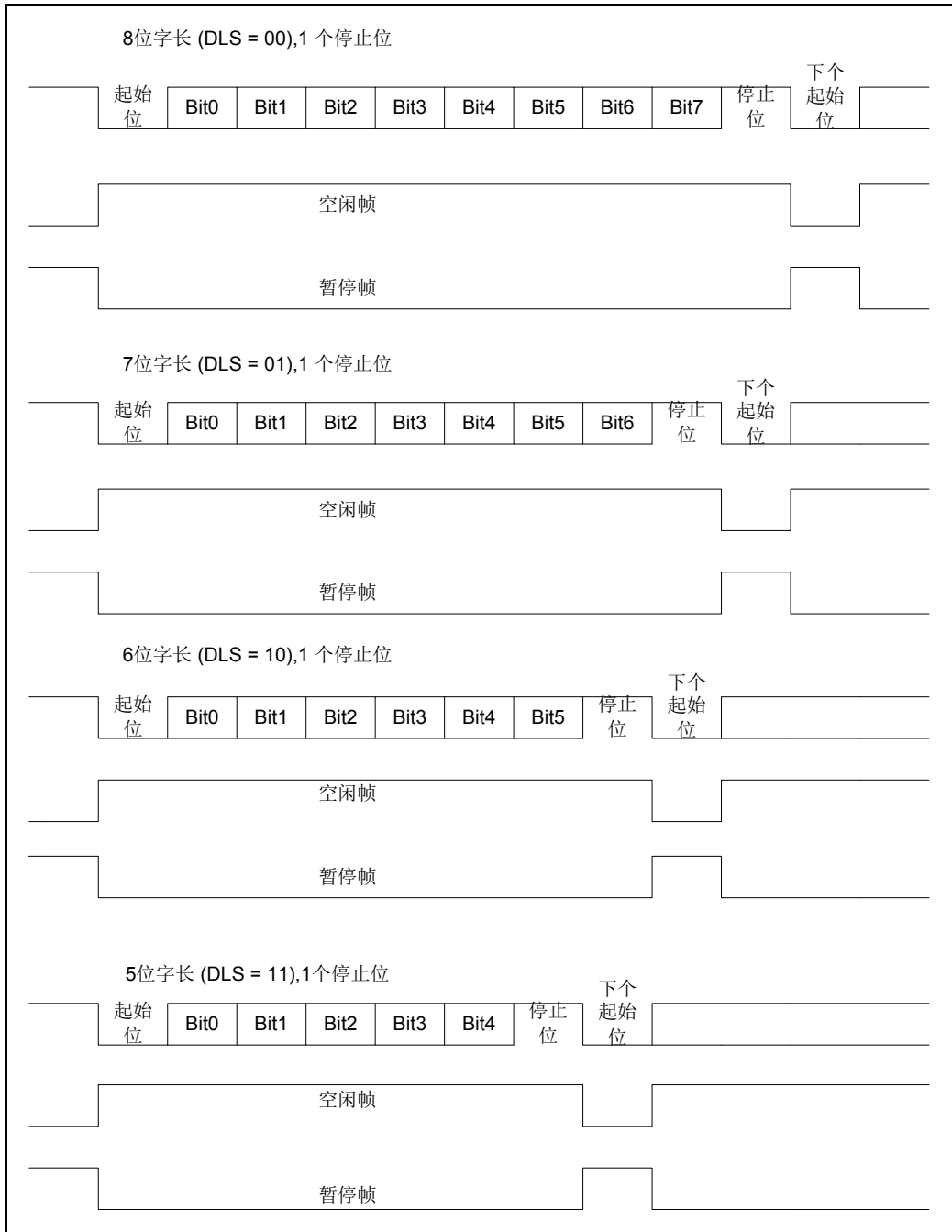


图 30-2 数据宽度设置

### 30.4.2 发送器

发送器根据 LCON 寄存器中的 DLS 位的状态发送 8-5 位的数据字。

当写入 UART\_TXBUF 后，发送移位寄存器中的数据在 TX 脚上输出。

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，TXBUF 寄存器充当了一个内部总线和发送移位寄存器之间的缓冲器（TXSR）。

每个字符之前都有一个低电平的起始位，字符结束有停止位，停止位的数目可配置。

UART 支持多种停止位的选择：0.5, 1, 1.5 和 2 个停止位。



注 1: 在写入 TXBUF 寄存器数据前必须先令 STAT 寄存器中的 TFFULL 位为 0。  
 注 2: 打开 TX 开关后, 数据才能在 TX 脚上输出可配置的停止位, 随每个字符发送的停止位的位数可以通过 LCON 寄存器中的 STOP 位进行编程。

- ◇ 0.5 个停止位: 在智能卡模式下发送和接收数据时使用。
- ◇ 1 个停止位: 停止位的位数的默认值。
- ◇ 1.5 个停止位: 在智能卡模式下发送和接收数据时使用。
- ◇ 2 个停止位: 可用于常规 UART 模式、以及调制解调器模式。

空闲帧包括了停止位。

断开帧可通过 MCON 寄存器的 BKREQ 位产生 10 位低电平 (当 DLS=00 时), 9 位低电平 (当 DLS=01 时), 8 位低电平 (当 DLS=10 时) 或者 7 位低电平 (当 DLS=11 时), 后跟 2 个停止位。

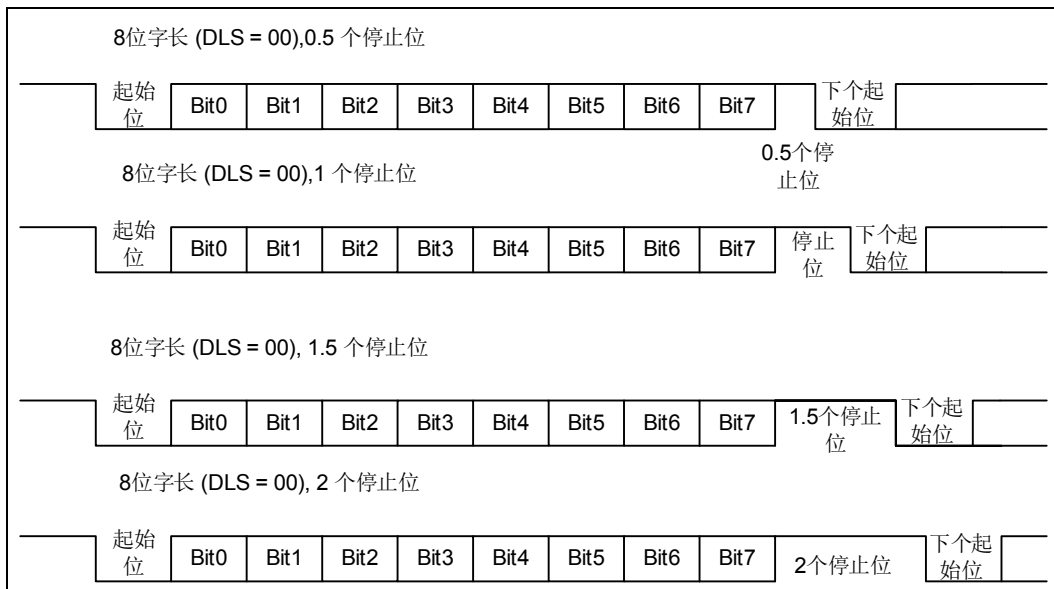


图 30-3 配置停止位

### TX FIFO 阈值设置

设置 FCON 寄存器的 TXTH 位, 可设定 FIFO 的阈值为 0,2,4,8, 当 FIFO 内的数据个数小于阈值会使得 STAT 寄存器的 TFTH 位为 1, 告知用户需要再填入数据, 以避免数据传送中断。

开启 IER 寄存器的 TFTH 位为 1, UART 会判断 FIFO 内的个数是否小于阈值, 使得 RIF 寄存器的 TFTH 位为 1, 产生中断。在产生中断的期间设置 ICR 寄存器的 TFTH 位为 1, 使得 RIF 寄存器的 TFTH 位清除, 若使用者未写入新的数据至 FIFO 中, FIFO 内的数据个数仍然小于阈值则不会再次产生中断, 需要重新开启 IER 寄存器的 TFTH 位。

注: 一开始 RIF 寄存器的 TFTH 与 TFEMPTY 位为 0, 当产生 FIFO 内的数据个数小于阈值事件与 FIFO 空事件时, 使得 TFTH 与 TFEMPTY 位为 1。STAT 寄存器的 TFTH 与 TFEMPTY 位则是反映 TX FIFO 状态。

配置步骤:

1. 设置 LCON 寄存器中的 DLS 位来定义字长。
2. 设置 LCON 寄存器中的 STOP 位设置停止位的位数。
3. 设置 LCON 寄存器中的 PE 与 PS 位设置校验控制开关与极性。
4. 设置 BRR 寄存器选择希望的波特率。
5. 如果采用多缓冲器通信, 配置 MCON 寄存器中的 TXDMAEN 位为 1。按多缓冲器通信中的描述配置 DMA 寄存器。
6. 设置 LCON 寄存器中的 TXEN 位, 使能发送器。
7. 把要发送的数据写进 TXBUF 寄存器(此动作将清除 STAT 寄存器中的 TFEMPTY 位)。
8. 在 TXBUF 寄存器中写入数据字时, 要等待 STAT 寄存器中的 TFFULL 位为 0, 它表示 FIFO 中未满足 16byte。当需要关闭 UART, 需要确认传输结束 STAT 寄存器中的 TSBUSY 位为 0, 避免破坏最后一次传输。

注: 当 LCON 寄存器的 TXEN 与 RXEN 位为 1 时, 无法写入 LCON 寄存器与 BRR 寄存器。

### 30.4.3 接收器

#### 30.4.3.1 防抖电路

在 UART\_RX 引脚上配置了一个防抖电路, 设置 LCON 寄存器中的 DBCEN 位开启功能, 输入信号须维持至少 8 个高电平或低电平, 才能使得信号反映至 UART 中, 反之则被忽略, 如下叙述。

在下图中, SYNC0 是指输入信号由模块时钟一次采样; SYNC1 是指 SYNC0 被模块时钟一次采样; SYNC2 表示 SYNC1 由模块时钟一次采样。SYNC0、SYNC1 和 SYNC2 可以表示成  $x[n] \times Ts$ ,  $x[n+1] \times Ts$  和  $x[n+2] \times Ts$ 。Ts 是模块时钟周期, n 是采样时间。如果关闭防抖模块, 时间就可以表示为  $x[n+1] \times Ts$ 。

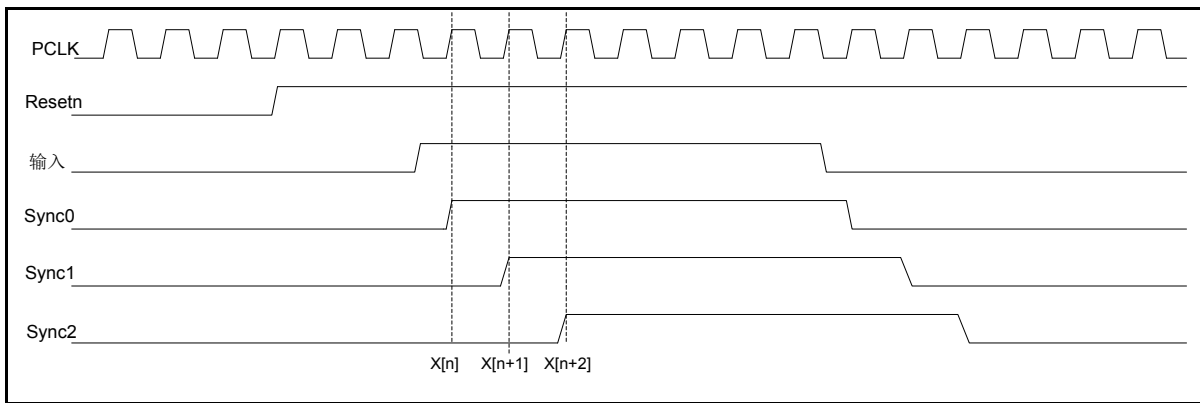


图 30-4 防抖动波形

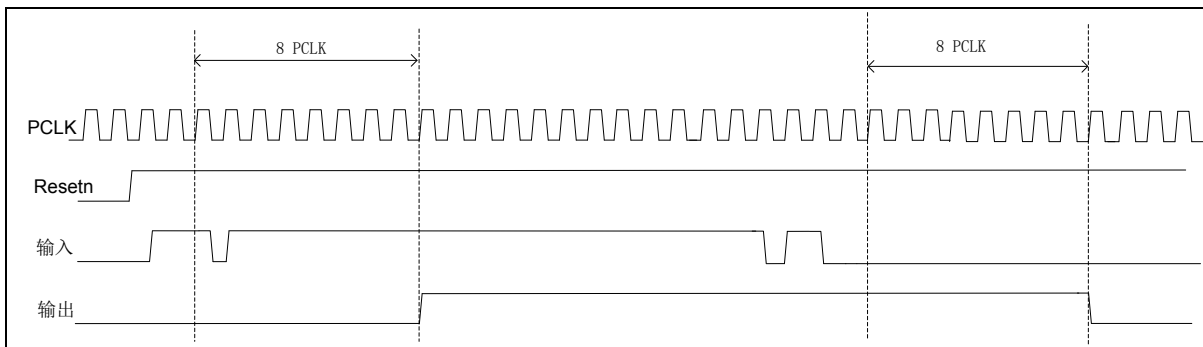


图 30-5 防抖动输出

### 30.4.3.2 起始位检测

接收器根据 LCON 寄存器中 DLS 位的状态接收 8-5 位的数据字。

#### 起始位检测

在 UART 中，如果辨认出一个特殊的采样序列，那么就认为检测到一个起始位。

该序列为：1 1 1 0 X 0 X 0 X 0 0 0 0 X X X X X X X

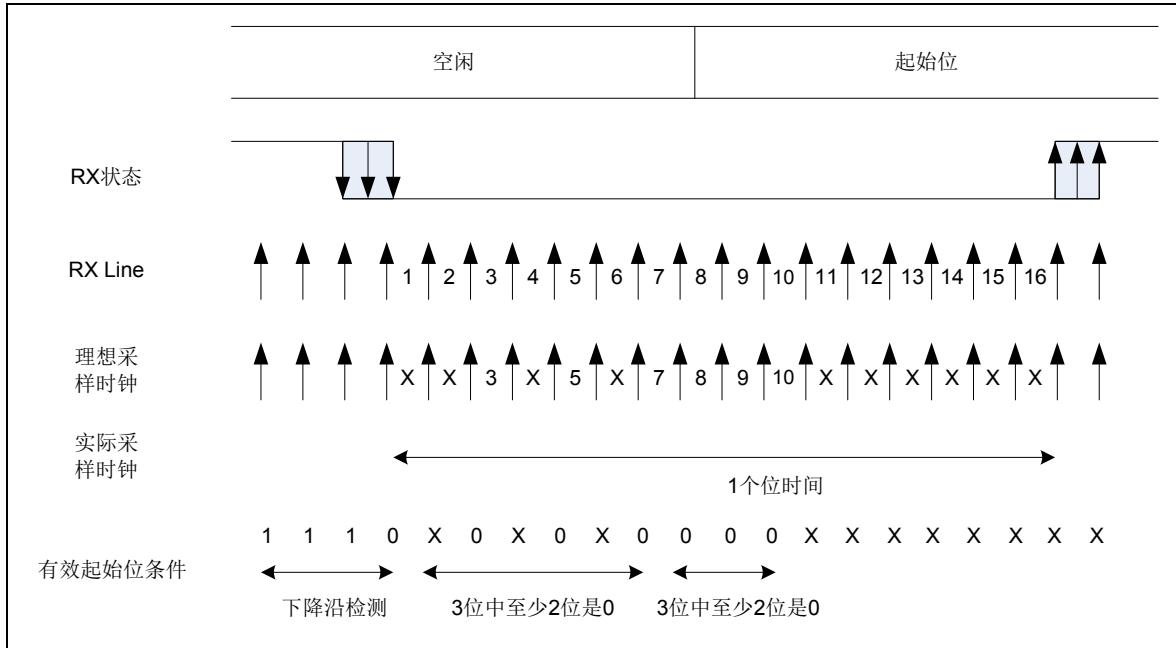


图 30-6 起始位检测

- 注 1: 如果该序列不完整，那么接收端将退出起始位检测并回到空闲状态（不设置标志位）开始等待下降沿。
- 注 2: 如果 3 个采样点都为 '0'（在第 3、5、7 位的第一次采样，和在第 8、9、10 的第二次采样都为 '0'），则确认收到起始位。
- 注 3: 如果两次 3 个采样点上有 2 个是 '0'（第 3、5、7 位的采样点和第 8、9、10 位的采样点），那么起始位仍然是有效的。如果不能满足这个条件，则中止起始位的检测过程，接收器会回到空闲状态。
- 注 4: 如果两次 3 个采样点上有 2 个是 '1'（第 3、5、7 位的采样点和第 8、9、10 位的采样点），那么起始位是无效的，将退出起始位检测并回到空闲状态，并会设置噪声标志位。

#### RX FIFO 阈值设置

设置 FCON 寄存器的 RXTH 位，可设定 FIFO 的阈值为 1,4,8,14，当 FIFO 内的数据个数大于等于阈值会使得 STAT 寄存器的 RFTH 位为 1，告知用户需要读取数据，以避免数据遗失。

开启 IER 寄存器的 RFTH 位为 1，UART 会判断 FIFO 内的个数是否大于等于阈值，使得 RIF 寄存器的 RFTH 位为 1，产生中断。在产生中断的期间设置 ICR 寄存器的 RFTH 位为 1，使得 RIF 寄存器的 RFTH 位清除，若用户未读取数据，FIFO 内的数据个数仍然大于等于阈值则不会再次产生中断，需要重新开启 IER 寄存器的 RFTH 位。

注：一开始 RIF 寄存器的 RFTH 位为 0，当产生 FIFO 内的数据个数大于等于阈值事件时，使得 RFTH 位为 1。STAT 寄存器的 RFTH 位则是反映 RX FIFO 状态。

配置步骤:

1. 设置 LCON 寄存器中的 DLS 位来定义字长。
2. 设置 LCON 寄存器中的 STOP 位设置停止位的位数。
3. 设置 LCON 寄存器中的 PE 与 PS 位设置校验控制开关与极性。
4. 设置 BRR 寄存器选择希望的波特率。
5. 如果采用多缓冲器通信, 配置 MCON 寄存器中的 RXDMAEN 位为 1。按多缓冲器通信中的描述配置 DMA 寄存器。
6. 设置 LCON 寄存器中的 RXEN 位, 这将启动接收器, 使它开始寻找起始位。

当一个字符被接收到时, STAT 寄存器的 RFEMPTY 位被置 0。它表明移位寄存器的内容被转移到 RX FIFO 中。换句话说, 数据已经被接收并且可以被读出 (包括与之有关的错误标志)。

这时如果 IER 寄存器的 RFTH 位是 1, 且 RX FIFO 阈值为 1, 将会引起中断请求。

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。RXBERR 标志也会和 RFTH 一起被置 1。

在多缓冲器通信时, RFEMPTY 在每个字节接收后被清零, 并由 DMA 对数据寄存器的读操作而置起。

RFFULL 位必须在下一字符接收结束前被清零, 以避免溢出错误。

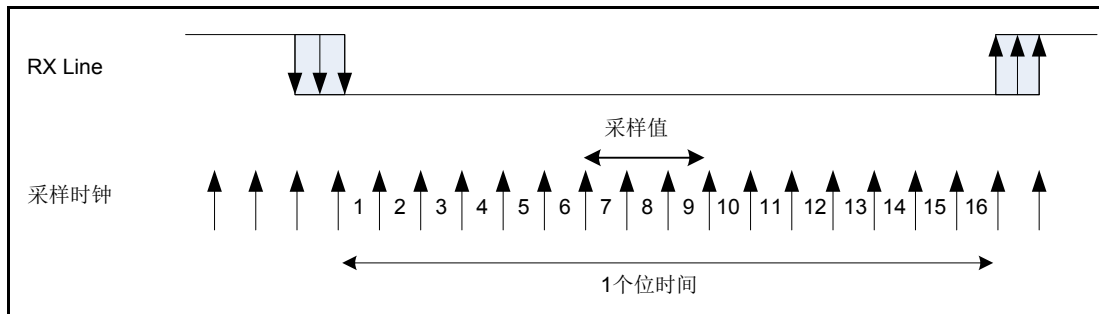


图 30-7 数值采样

### 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:

1. FERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 FERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为帧错误。
4. 寄存器 RIF 中的 RXBERR 位则会在接收的过程中被置起, 若 IER 中的 RXBERR 位

为 1 则会产生中断。

### 奇偶位错误

当以下情况发生时检测到奇偶性错误：

由于没有同步上或大量噪音的原因，奇偶位没有在预期的时间上接收和识别出来。

当奇偶位错误被检测到时：

1. PERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 PERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为校验错误。
4. 寄存器 RIF 中的 RXBERR 位则会在接收的过程中被置起，若 IER 中的 RXBERR 位为 1 则会产生中断。

### 断开错误

当以下情况发生时检测到断开错误：

由于没有同步上或大量噪音的原因，字符与字符停止位为 0 时没有在预期的时间上接收和识别出来。

当断开错误被检测到时：

1. BKERR 位被硬件置 1
2. 此时读取的 RXBUF 寄存器数据可能有错。
3. 寄存器 STAT 中的 BKERR 位显示当前从 FIFO 读取的 RXBUF 寄存器是否为断开错误。
4. 这个错误并不会产生中断。

### 溢出错误

当以下情况发生时检测到溢出错误：

由于 FIFO 已满 16byte 还没有被读取，而又接收到一个字符，则发生溢出错误。

当溢出错误被检测到时：

1. RFOERR 位被硬件置 1
2. RXBUF 内容将不会丢失，读取 RXBUF 寄存器仍能得到先前的数据。
3. 移位寄存器中以前的内容将被覆盖，随后接收的数据将丢失。
4. 寄存器 RIF 中的 RFOERR 位则会在接收的过程中被置起，若 IER 中的 RFOERR 位为 1 则会产生中断。

采样值	接收到的值
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

表 30-1 来自采样数据的噪音检测

### 30.4.4 状态寄存器

在 UART 中配置了 15 种 UART 状态提供使用者使用，叙述如下

- ◇ PERR (Parity error)
  - 当所接收的字符没有正确的校验字节，将生成一个奇偶校验错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。
- ◇ FERR (Framing error)
  - 当接收到的字符停止位为 0 时，产生帧错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。
- ◇ BKERR (Break error)
  - 当接收到的字符与字符停止位为 0 时，产生断开错误。该错误与 FIFO 顶部的字符有关，即读取 RXBUF 寄存器的字符。
- ◇ CTSSTA (CTS<sub>n</sub> status error)
  - 清除发送。此位为 CTS<sub>n</sub> pin 上的状态。当 CTSSTA 位为 0，这是一个迹象表明调制解调器和数据设备已准备好与 UART 进行数据交换。
- ◇ RSBUSY (RX shift register busy)
  - 当此位为 1 表示接收器正在接收字符，为 0 则为接收完成。
- ◇ RFTH (RX FIFO 阈值)
  - 当此位为 1 表示 RX FIFO 内数据大于等于阈值（设置 FCON 寄存器的 RXTH，可

- 设定阈值为 1,4,8,14), 告知使用者需要读取 RXBUF 寄存器。
- ◇ RFEMPTY (RX FIFO empty)
    - 当此位为 1 表示 RX FIFO 内无任何字符, 为 0 则为已接收 1 个以上的字符。
  - ◇ RFFULL (RX FIFO full)
    - 当此位为 1 表示 RX FIFO 内已有 16 个字符, 需要被读取。
  - ◇ RFOERR (RX FIFO overrun)
    - 当此位为 1 表示 RX FIFO 内已有 16 个字符, 且又再接收 1 个字符, 此时 FIFO 内字符不会丢失, 接收的字符则会被丢失。
  - ◇ RFUERR (RX FIFO underrun)
    - 当此位为 1 表示 RX FIFO 内无任何字符, 且又被读取。
  - ◇ TSBUSY (TX shift register busy)
    - 当此位为 1 表示发送器正在传送字符, 为 0 则为传送完成, 当写入第一个数据至 TXBUF 寄存器就会使得 TSBUSY 位为 1。
  - ◇ TFTH (TX FIFO 阈值)
    - 当此位为 1 表示 TX FIFO 内数据小于阈值 (设置 FCON 寄存器的 TXTH, 可设定阈值为 0,2,4,8), 告知使用者需要写入 TXBUF 寄存器。
  - ◇ TFEMPTY (TX FIFO empty)
    - 当此位为 1 表示 TX FIFO 内无任何字符, 为 0 则为准备发送 1 个以上的字符。
  - ◇ TFFULL (TX FIFO full)
    - 当此位为 1 表示 TX FIFO 内已有 16 个字符, 准备发送。
  - ◇ TFOERR (TX FIFO overrun)
    - 当此位为 1 表示 TX FIFO 内已有 16 个字符, 且又在写入 1 个字符, 此时 FIFO 内字符不会丢失, 写入的字符则会被丢失。



### 30.4.5 波特率生成器

接收器和发送器的波特率在 UARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{UARTDIV} = \text{UART\_BRR.BRR.}$$
$$\text{TX/RX baud} = \text{PCLK} / \text{UARTDIV}$$

注 1: 当 LCON 寄存中的 RXEN 与 TXEN 为 1 时, BRR 寄存器无法被写入。

注 2: 当  $\text{BRR}\langle 15:4 \rangle = 0$  时, 无法运行。

从 UART\_BRR 寄存器值中推断出 UART 波特率

- ◇ 在 4 MHz 下, 为了得到 115200 波特率
  - $\text{UARTDIV} = 4000000/115200 = 34.7$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 35\text{d} = 23\text{h}$  (四舍五入)
- ◇ 在 8 MHz 下, 为了得到 9600 波特率
  - $\text{UARTDIV} = 8000000/9600 = 833.33$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 833\text{d} = 341\text{h}$  (四舍五入)
- ◇ 在 16 MHz 下, 为了得到 1200 波特率
  - $\text{UARTDIV} = 16000000/1200 = 13333.33$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 13333.33\text{d} = 3415\text{h}$  (四舍五入)
- ◇ 在 24 MHz 下, 为了得到 460800 波特率
  - $\text{UARTDIV} = 24000000/460800 = 52.08$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 52\text{d} = 34\text{h}$  (四舍五入)
- ◇ 在 48 MHz 下, 为了得到 115200 波特率
  - $\text{UARTDIV} = 48000000/115200 = 416.66$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 417\text{d} = 1\text{A}1\text{h}$  (四舍五入)
- ◇ 在 48 MHz 下, 为了得到 921600 波特率
  - $\text{UARTDIV} = 48000000/921600 = 52.08$
  - $\text{BRR}\langle 15:0 \rangle = \text{UARTDIV} = 52\text{d} = 34\text{h}$  (四舍五入)

波特率		16 倍过采样		
序号	预期值	实际值	BRR	%误差
1	734Bps	733KBps	0xFFCC	0
2	1.2KBps	1.2KBps	0x9C40	0
3	2.4KBps	2.4KBps	0x4E20	0
4	4.8KBps	4.8KBps	0x2710	0
5	9.6KBps	9.6KBps	0x1388	0
6	19.2KBps	19.2KBps	0x9C4	0
7	38.4KBps	38.4KBps	0x4E2	0
8	57.6KBps	57.62KBps	0x341	0.03
9	115.2KBps	115.11KBps	0x1A1	0.08
10	230.4KBps	230.77KBps	0xD0	0.16
11	460.8KBps	461.54KBps	0x68	0.16
12	921.6KBps	923.07KBps	0x34	0.16
13	1.5MBps	1.5MBps	0x20	0
14	2MBps	2MBps	0x18	0
15	3MBps	3MBps	0x10	0

表 30-2 时钟为 48MHz 下，设置波特率时的误差计算

### 30.4.6 自动波特率检测

UART 可以根据接收到的一个字符来检测和自动设置 BRR 寄存器的值。自动波特率检测在两种情况下有用：

- ◇ 通讯速度不可知的情况下
- ◇ 使用低精度时钟源，需要在不测量时钟偏差的条件下纠正波特率的时候。

时钟源的频率必须和预期的波特率保持相对的稳定（过采样率为 16，并且波特率处于 PCLK/65535 和 PCLK/16 之间）。

在打开自动波特率检测之前，字符的内容必须先确认。有三个可能的字符内容，能够通过 MCON 寄存器中的 ABRMOD 位进行选择。具体是：

- ◇ 模式 0 (0x00)：波特率在 UART 的 RX 引脚的两个连续的下降沿上测量（起始位的下降沿和最低数据位的下降沿 (LSB)）
- ◇ 模式 1 (0x01)：波特率在 UART 的 RX 引脚下降沿和后续上升沿之间（起始位的长度）测量。
- ◇ 模式 2 (0x10)：波特率在 UART 的 RX 引脚下降沿和后续上升沿之间（起始位的长度加上 bit<0>的长度）测量（e.g. 0xFE）。

将 MCON 寄存器中的 ABREN 位置 1，开启自动波特率检测功能。UART 在 RX 线上等待第一个字符过来。当自动波特率操作结束后，RIF 寄存器中的 ABEND 标志会被硬件自动设置 1，若 IER 寄存器中的 ABEND 位为 1 则会产生中断。

如果线路噪声严重，不能保证得到的波特率是准确的。这时 BRR 值可能是错的或者 ABEND 错误标志会被置 1。在通讯速度超出自动波特率检测范围（位长度不在 0x10 到 0x1FFFF 个时钟周期之间）时也会发生这种情况。

在此模式中配置了两个中断，分别为检测超时与检测结束。在软件并未清除 MCON 寄存器中的 ABREN 的情况下，UART 计数器会持续计数直到 0x1FFFF 后，硬件会自动将 ABREN 清除，并将 RIF 寄存器中的 ABTO 位置起，如果开启中断，则会产生 ABTO 中断。

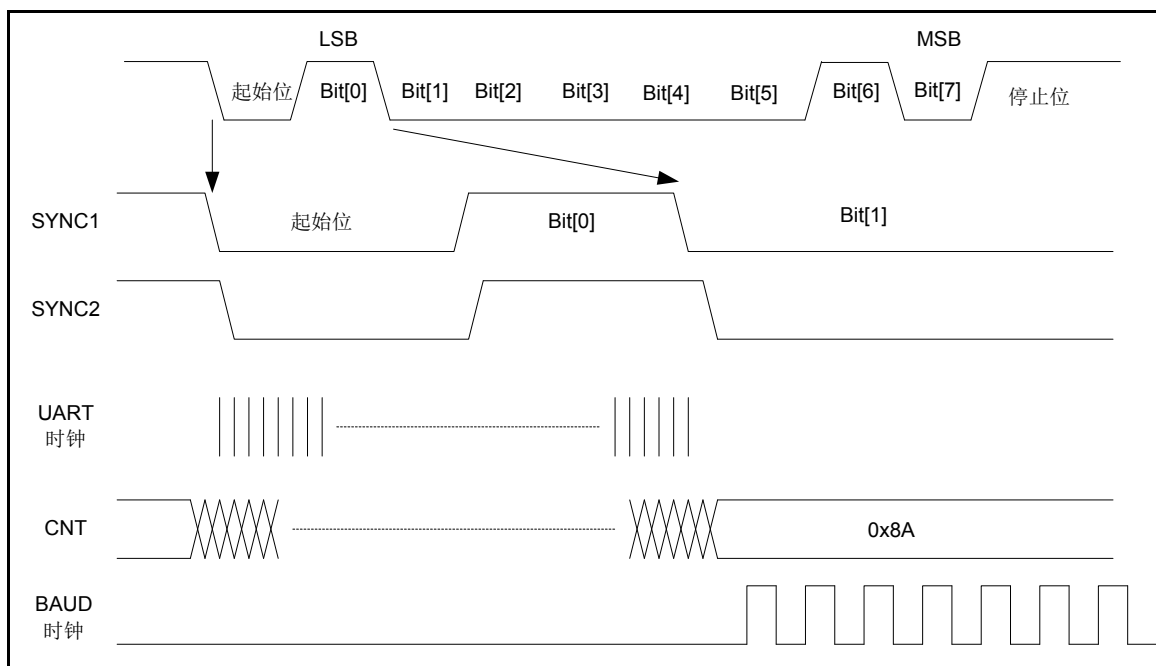


图 30-8 自动波特率检测模式 0

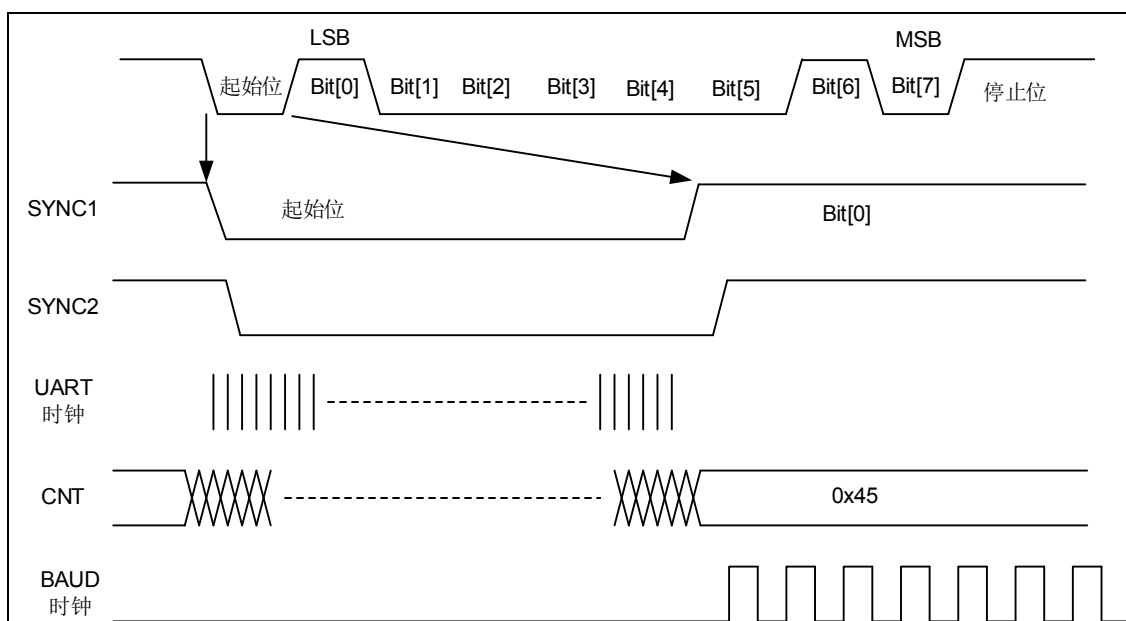


图 30-9 自动波特率检测模式 1

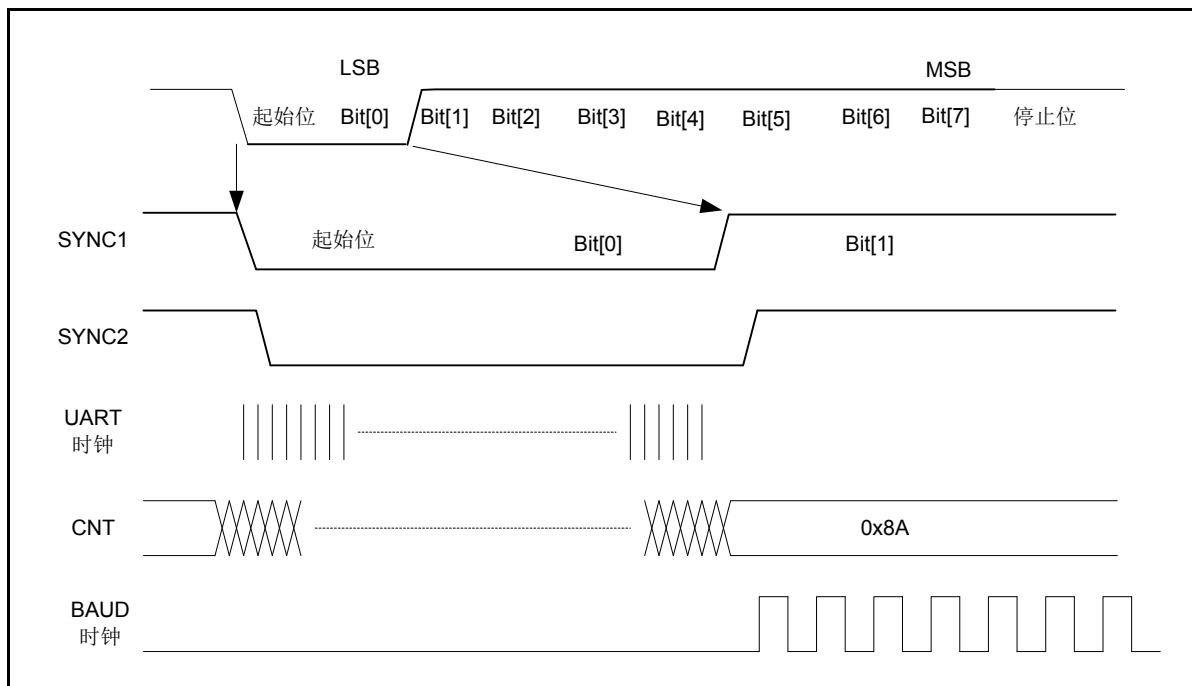


图 30-10 自动波特率检测模式 2

### 30.4.7 自动流控制

如果使能自动流控制，接收 FIFO 和发送 FIFO 会通过 UARTx\_RTSn 和 UARTx\_CTSn 引脚去控制 UART 的接收 (RX) 和发送 (TX)。

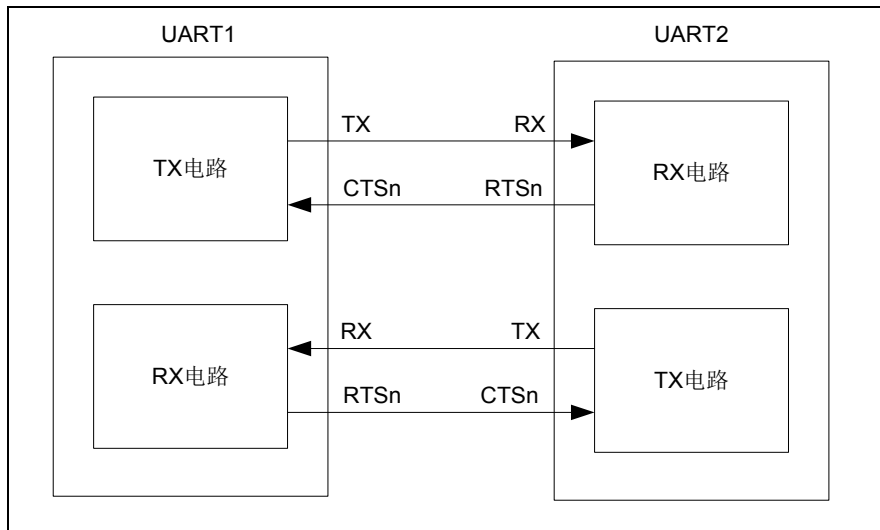


图 30-11 自动流控制框图

#### 30.4.7.1 RTSn 控制

当自动 RTS 被使能时，接收 FIFO 达到由 UART\_FCON 寄存器设置的阈值，UART\_RTSn 输出会拉为高电平。当 UART\_RTSn 连接到另一个 UART 设备的 UART\_CTSn 输入，另一个 UART 会停止发送串行数据，直到接收 FIFO 完全是空的。

可选择接收 FIFO 阈值的值是：1, 4, 8, 14 个字符。一个额外的字符有可能会在 UART\_RTSn 成为无效后传送到 UART (由于从 UART 进入发送器的数据尚未发送完成)，阈值设置为 14 个字符能最大限度的使用 FIFO。

硬件通过读取接收缓冲寄存器 UART\_RXBUF，一旦得知接收 FIFO 完全为空时，UART\_RTSn 变低电平，通知其他 UART 继续发送数据。

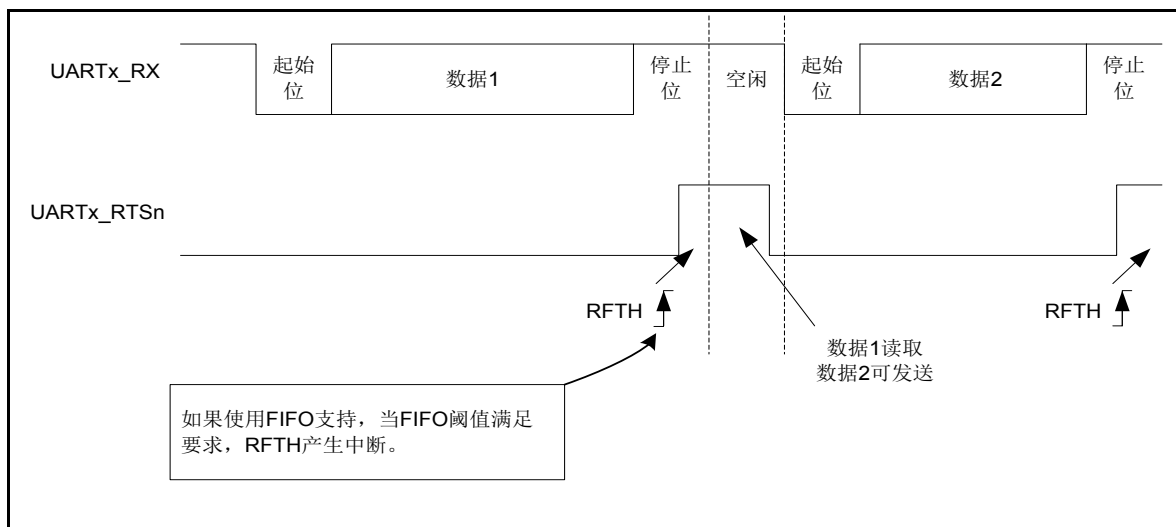


图 30-12 自动 RTSn 控制

### 30.4.7.2 CTSn控制

当自动 CTS 启用，每当 UART\_CTSn 输入变为高电平时，UART 发送器被禁用。这可以防止接收端 UART 的 FIFO 溢出。在最后一个停止位发出时，假设 UART\_CTSn 依然为低电平，则发送器会在禁用前继续传输一个字符。在发送器被禁用时，发送 FIFO 仍然可以被写入，甚至溢出。

只要 UART\_CTSn 输入一变换状态，硬件就自动设置 RIF 寄存器中的 DCTS 位。它表明接收器是否准备好进行通信。如果开启中断，则会产生 DCTS 中断。

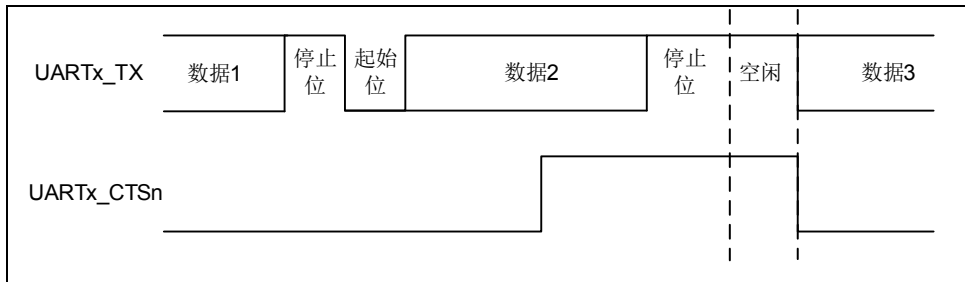


图 30-13 自动 CTSn 控制

自动流控制可以减少系统中断，当自动流控制使能，CTS<sub>n</sub> 状态不会触发系统中断。因为是设备自动地控制其收发器。

### 30.4.7.3 RS485 驱动使能 (DE)

当 RS485 驱动使能功能启用。它允许用户通过 DE (驱动使能) 信号来激活外部收发器的控制端。滞后时间是一个发送消息的最后一个字节的停止位和释放 DE 信号之间的时间间隔，这个时间可以在 RS485 寄存器中的 DLY 位设置。DE 信号的极性则可以通过 RS485 寄存器中的 AADINV 位中设置并进行选择。

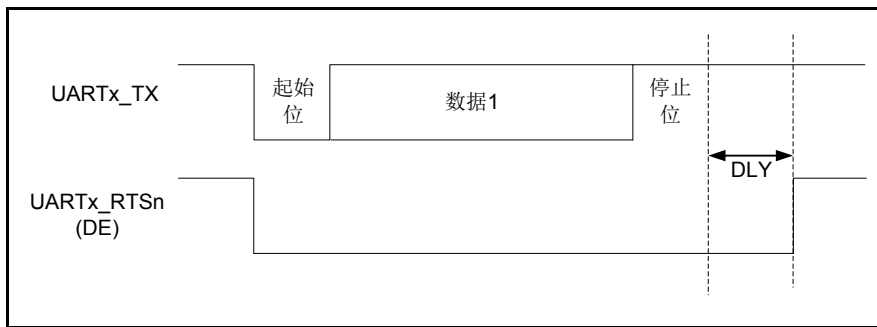


图 30-14 驱动使能当 AADINV=0

### 30.4.8 校验控制

设置 LCON 寄存器中的 PE 位，可以使能校验控制（发送时生成一个校验位，接收时进行校验检查）。根据 DLS 位定义的帧长度，可能的 UART 帧格式列在下表中。

DLS<1:0>	PE	UART 帧
00	0	起始位→8 位数据→停止位
00	1	起始位→8 位数据→校验位→停止位
01	0	起始位→7 位数据→停止位
01	1	起始位→7 位数据→校验位→停止位
10	0	起始位→6 位数据→停止位
10	1	起始位→6 位数据→校验位→停止位
11	0	起始位→5 位数据→停止位
11	1	起始位→5 位数据→校验位→停止位

表 30-3 帧格式

#### ◇ 奇校验

校验位的内容使得一帧中的 8, 7, 6 或 5 个 LSB 数据以及校验位中 1 的个数为奇数。

例如：数据=00110101，有 4 个 1，如果选择奇校验（在 LCON 寄存器中的 PS=0），校验位将是 1。

#### ◇ 偶校验

校验位的内容使得一帧中的 8, 7, 6 或 5 个 LSB 数据以及校验位中 1 的个数为偶数。

例如：数据=00110101，有 4 个 1，如果选择偶校验（在 LCON 寄存器中的 PS=1），校验位将是 0。

#### ◇ 接收时的校验检查

如果校验检查失败，STAT 寄存器中的 PERR 标志会被置 1，如果 IER 寄存器中的 RXBERR 为 1，将引发相应中断。

#### ◇ 发送时的校验生成

如果 LCON 寄存器的 PE 位被置 1，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去（如果选择奇校验奇数个 1，如果选择偶校验偶数个 1）。



### 30.4.9 多处理器通信

设置 DLS 位为 8 位字长（第 9 位为判断地址或数据）

设置 RS485 寄存器的 AADEN 位为 1 以进入模式。

设置 RS485 寄存器的 ADDR 位配置匹配地址

可以将多个 UART 连接成一个网络来实现多机通讯。例如某个 UART 设备可以是主，它的 TX 输出和其他 UART 从设备的 RX 输入相连接；UART 从设备各自的 TX 输出逻辑地与在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，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进入静默模式。要使用静默模式功能，RS485 寄存器的 AADEN 位必须被置 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 RS485 寄存器的 ADDR 位。

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。当 UART 进到静默模式后，接收字节时既不会改变 RIF 寄存器的 RFTH 位标志也不会产生中断或发出 DMA 请求。

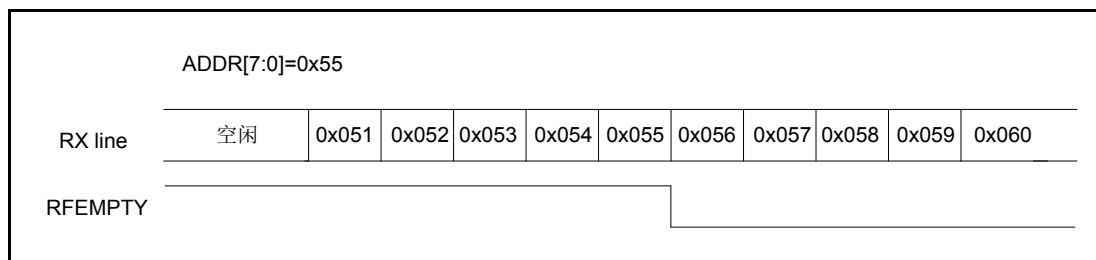


图 30-15 使用地址标示检测模式

### 30.4.10 LIN模式

#### ◇ LIN 发送

和常规的 UART 发送相同，但包含下列区别：

- ◇ 设置 DLS 位为 8 位字长
- ◇ 设置 LIN 寄存器的 LINEN 位为 1 以进入 LIN 模式。这时，置 LIN 寄存器的 LINBKREQ 位为 1 将发送 13 位 0 作为断开符号。然后发两位 1，以允许对下一个开始位的检测。

#### ◇ LIN 接收

当 LIN 模式被使能时（LIN 寄存器的 LINEN 位为 1），断开符号检测电路被激活。该检测完全独立于 UART 接收器。不管是在总线空闲时还是在发送某数据帧期间，断开符号只要一出现就能检测到。

一旦接收器被激活（LCON 寄存器的 RXEN 位为 1），电路就开始监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8, 9, 10 个过采样时钟点上进行采样，就像针对数据一样。如果 10 个（当 LIN 寄存器中的 LINBDL=0）或 11 个（当 LIN 寄存器中的 LINBDL=1）连续位都是 0，并且又跟着一个定界符，RIF 寄存器的 LINBK 位标志就会被置 1。如果 IER 寄存器的 LINBK 位为 1，还会产生中断。在确认定界符前，要检查定界符，因为它表示 RX 线已经回到高电平。如果在第 10 或 11 个采样点之前采样到了 1，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如正常 UART 那样工作，不再考虑检测断开符号。如果 LIN 模式被激活（LINEN=1），只要一发生帧错误（例如：停止位检测到 0，这种情况出现在断开符号被接收到的时候），接收器就停止，直到断开符号检测电路接收到一个 1（这种情况发生于断开符号没有完整的发出来），或一个定界符（这种情况发生于已经检测到一个完整的断开符号）。

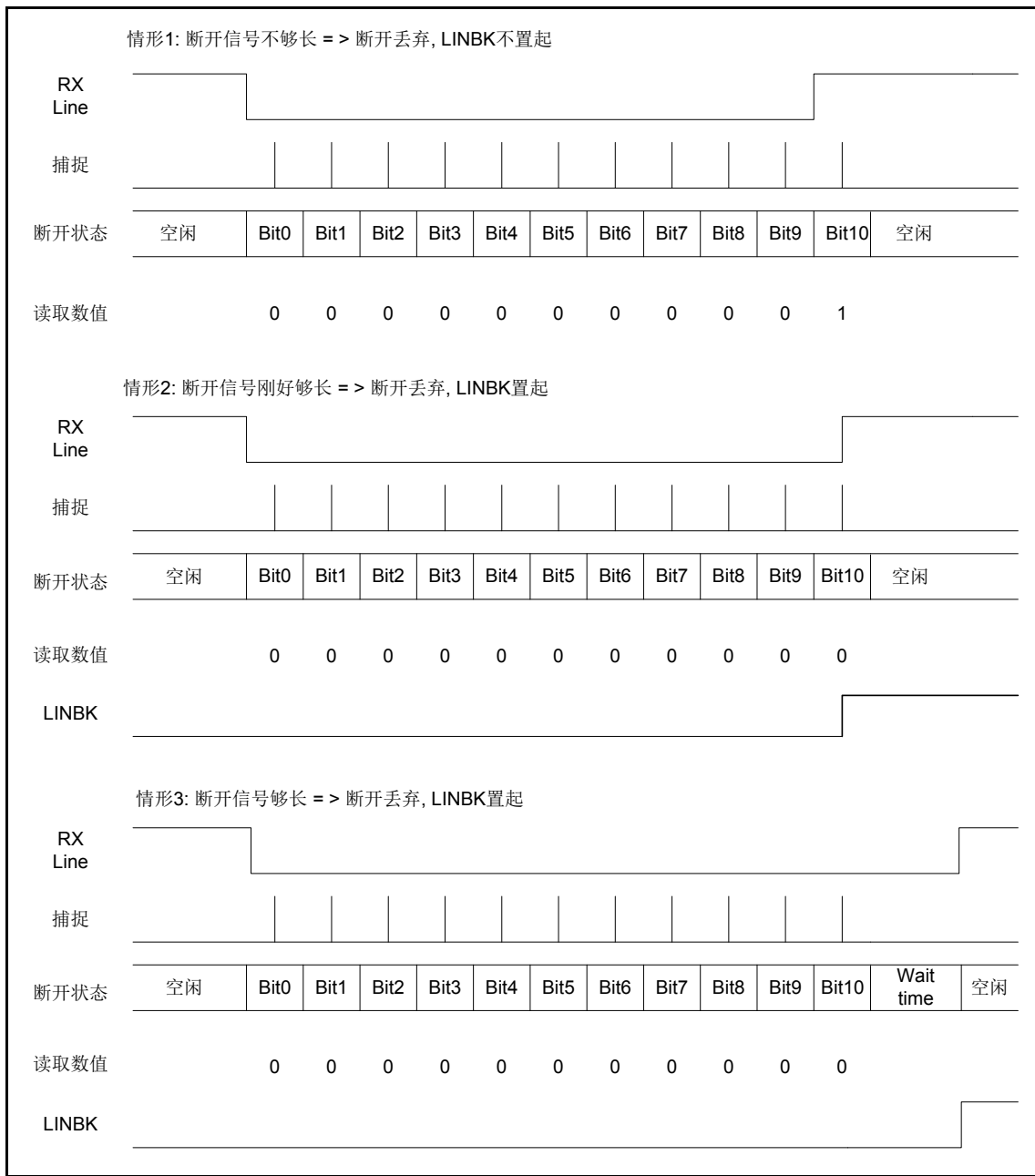


图 30-16 LIN 模式下断开信号检测 (11 位断开长度-LBDL 位为 1)

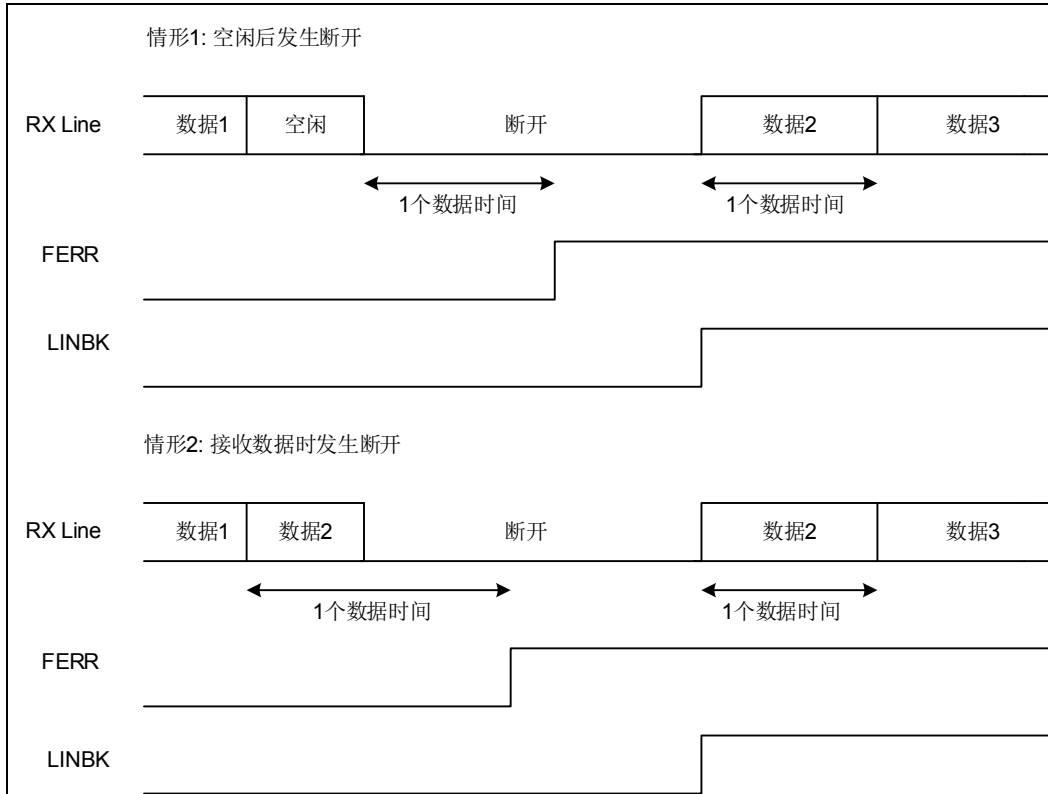


图 30-17 LIN 模式下的断开检测与帧错误的检测

### 30.4.11 单线半双工通讯

UART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部是连在一起的。使用控制位（MCON 寄存器中的 HDEN 位）选择半双工或全双工通信。

◇ 当 HDEN 为 1 时

- TX 和 RX 引脚在芯片内部是连在一起的
- RX 不再被使用
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态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。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 LCON 寄存器的 TXEN 为 1，只要数据一写到数据寄存器中，发送就会开始。

### 30.4.12 智能卡模式

设置 8 位数据位加校验位：即 LCON 寄存器中 DLS=00, PE=1

设置 0.5/1.5 个停止位：即 LCON 寄存器的 STOP=0/1

设置 SCARD 寄存器的 SCEN 为 1 以进入 Smart card 模式

在 T=0（字符）模式中，保护时间内，校验错误在字符发送完毕后被提出。

所示为在数据线上有校验错误和没有校验错误时的情形。

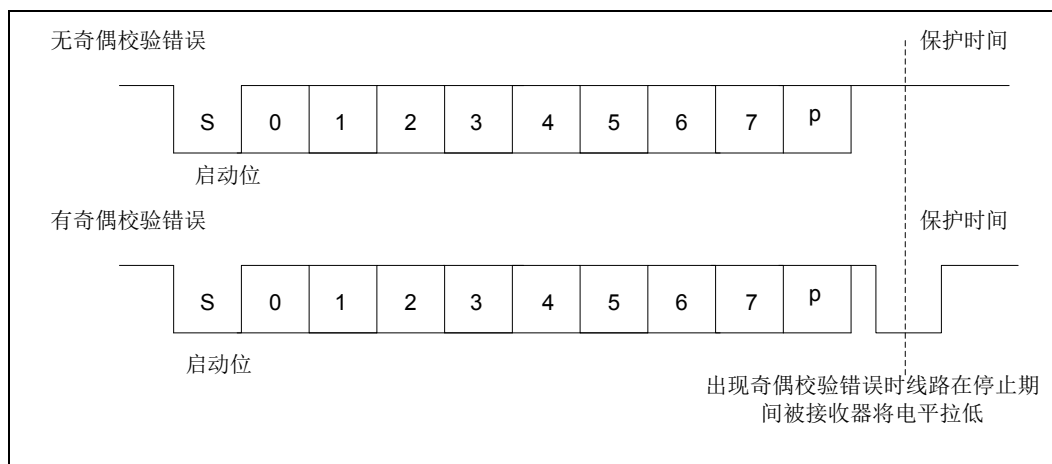


图 30-18 ISO 7816-3 异步协定

当连接到智能卡时，UART 的 TX 管脚和智能卡数据管脚通过同一根双向数据线进行通讯。所以 TX 引脚必须配置成开漏状态。

智能卡是一个单线半双工通信协议：

- ◇ 从发送移位寄存器发送数据会经过至少 1/2 个时钟周期的延迟。正常工作时，已满的发送移位寄存器会在下一个时钟边沿开始移位。在智能卡模式下，此发送过程还会进一步经过 1/2 波特时钟周期的延迟。
- ◇ 如果在接收一个使用 0.5 或 1.5 个停止位编程的帧期间检测到奇偶校验错误，则在完成接收帧后，发送线会被拉低一个时钟周期。这是为了向智能卡指出发送到 UART 的数据尚未正确接收。此 NACK 信号（将发送线拉低 1 个时钟周期）会导致发送器端（配置为 1.5 个停止位）出现帧错误。应用程序可根据协议重新发送数据。如果 NACK 控制位置 1，则接收器会发送“NACK”信号；否则不会发送 NACK 信号。
- ◇ 通过对保护时间寄存器进行编程，可以延迟 RIF 寄存器的 TBC 标志的置位。正常工作时，当发送移位寄存器为空时，会对 TBC 标志进行置位。在智能卡模式下，空的发送移位寄存器会触发保护时间计数器，使其递增计数至保护时间寄存器中的值。在此期间，TBC 标志被强制为低电平。当保护时间计数器达到设置值时，TBC 置位为高电平。
- ◇ 对 TBC 标志的释放不受智能卡模式的影响。
- ◇ 如果在发送端检测到帧错误（由来自接收器的 NACK 信号引起），则发送端的接收器不会将 NACK 作为起始位进行检测。根据 ISO 协议，接收到的 NACK 信号的持

续时间可以是 1 或 2 个时钟周期。

- ◇ 在接收端，如果检测到奇偶校验错误并发送了 NACK 信号，则接收端不会将 NACK 作为起始位进行检测。

注：在智能卡模式下带有帧错误的 0x00 数据将被视为数据，而非中断。

下图详细介绍了 UART 如何对 NACK 信号采样。在本例中，UART 正在发送数据并配置了 1.5 个停止位。UART 的接收部分已被使能，以检查数据的完整性和 NACK 信号。

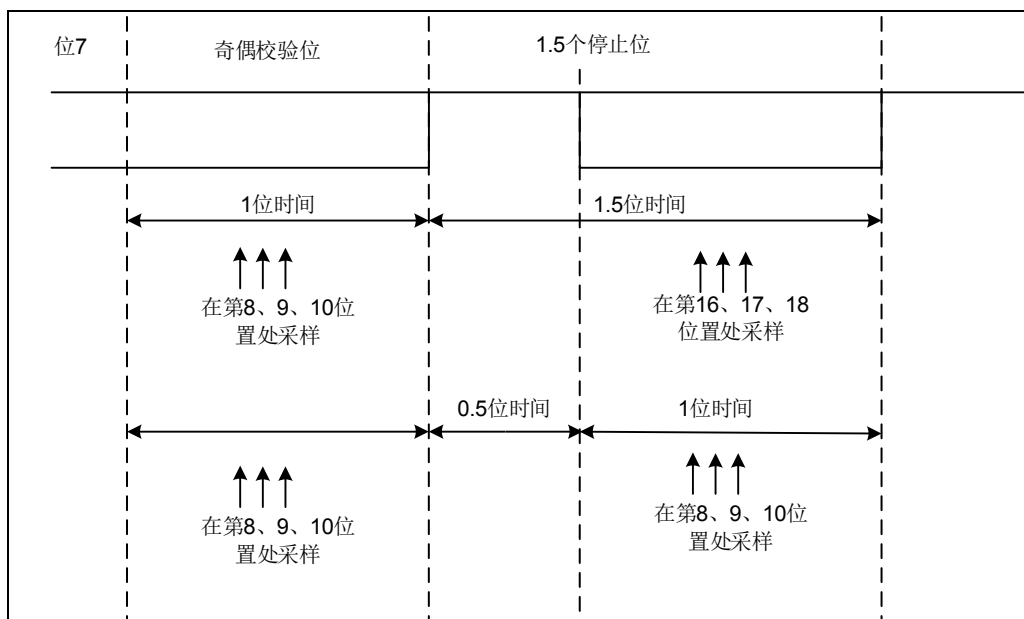


图 30-19 用 1.5 位停止位时检测校验错误

UART 可以通过 CK 脚向智能卡提供时钟。智能卡模式中，CK 和通讯没有关系，只是通过一个 5 位的预分频器从内部外设时钟源得到时钟信号。这个分频系数在 SCARD 寄存器的 PSC 位设置。CK 频率可以设置在  $f_{ck}/2$  到  $f_{ck}/64$  之间， $f_{ck}$  指外设输入时钟。

### 30.4.13 IrDA SIR 模块

设置 MCON 寄存器的 IREN 为 1 以进入 IrDA 模式。

IrDA SIR 物理层规定使用反相归零 (RZI) 调制方案, 它以红外光脉冲表示逻辑 0。

SIR 发送编码器用于调制 UART 发出的非归零 (NRZ) 位流。输出脉冲流会发送到外部输出驱动器和红外线 LED。UART 支持的 SIR 编码比特率最高为 115.2Kbps。在正常模式下, 所发送的脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器用于解调由红外探测器发出的归零位流, 并将接收到的 NRZ 串行位流输出到 UART。在空闲状态下, 解码器输入通常为高电平 (标记状态)。发送编码器输出的极性与解码器输入相反。当解码器输入为低电平时, 会检测到起始位。

- ◇ IrDA 是一个半双工通信协议。如果发送器忙, 例如 UART 正在向 IrDA 编码器发送数据, 则 IrDA 解码器会忽略 IrDA 接收线上的所有数据; 如果接收器忙, 例如 UART 正在接收来自 RX 引脚上的数据, 则 IrDA 不会对 UART 发送到 IrDA 的 TX 数据进行编码。在接收数据时, 应避免同时进行发送, 因为这样做可能会破坏要发送的数据。
- ◇ SIR 发送逻辑把 0 作为高脉冲发送, 把 1 作为低电平发送。脉冲的宽度规定为所选位周期的 3/16
- ◇ SIR 解码器用于将兼容 IrDA 的接收信号转换为 UART 的位流。
- ◇ SIR 接收逻辑把高电平状态解释为 1, 把低脉冲解释为 0。
- ◇ 发送编码器输出的极性与解码器输入相反。SIR 输出在空闲时处于低电平状态。
- ◇ 在 IrDA 模式里, LCON 寄存器中的 STOP 位必须配置成 1 个停止位。

接收器的建立时间应由软件进行管理。IrDA 物理层规范规定发送和接收之间至少要经过 10ms 的延迟 (IrDA 是一个半双工协议)。

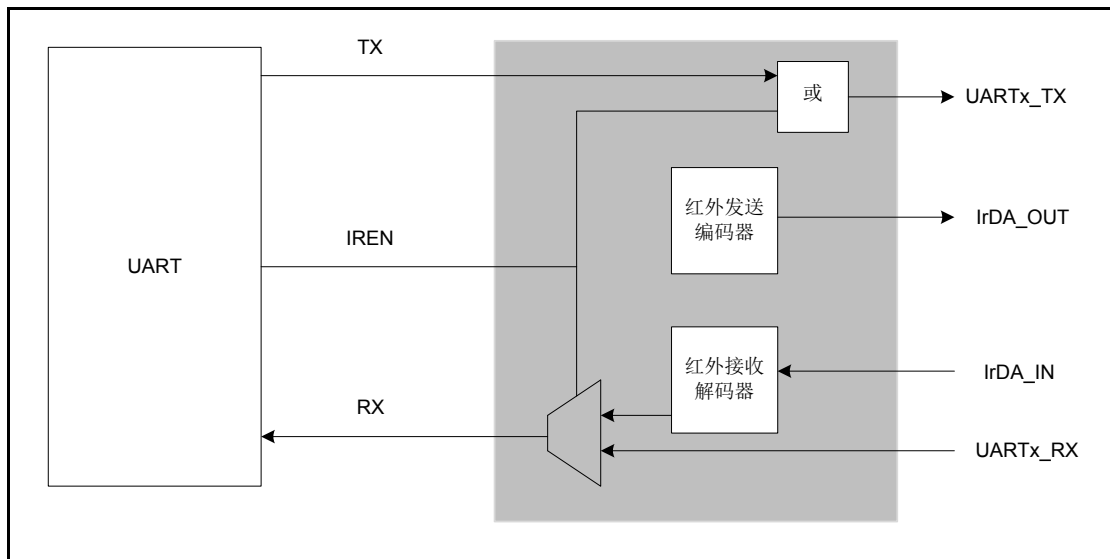


图 30-20 红外收发框图

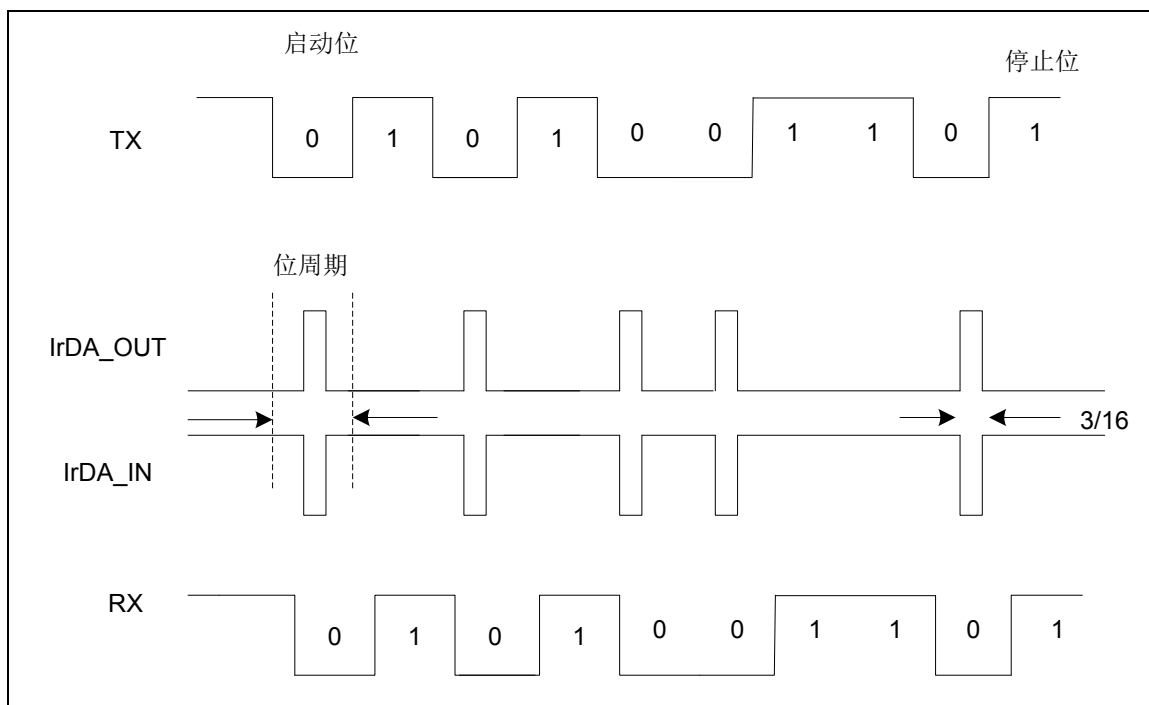


图 30-21 IrDA 数据调制 (3/16) - 正常模式



### 30.4.14 使用DMA连续通讯

设置 MCON 的 RXDMAEN 为 1 使能 RX DMA 或 TXDMAEN 为 1 使能 TX DMA。

UART 可以利用 DMA 连续通信。RX 缓冲器和 TX 缓冲器的 DMA 请求是分别产生的。

#### 利用 DMA 发送

使用 DMA 进行发送，可以通过设置 MCON 寄存器中的 TXDMAEN 位使能。在数据被预先放到 DMA 外设所设定的 SRAM 区域，设置一个 DMA 通道给 UART 发送，要使用下列步骤（x 指通道号）：

- ◇ 在 DMA 控制寄存器上将 TXBUF 寄存器地址配置成 DMA 传输的目的地址。在每个发送事件后，数据将被传送到这个地址。
- ◇ 在 DMA 控制寄存器上将内存地址配置成 DMA 传输的源地址。在每个 TTFH 事件后，将从此存储器区读出数据并传送到 TXBUF 寄存器中。
- ◇ 在 DMA 控制寄存器中配置要传输的总的字节数。
- ◇ 在 DMA 控制寄存器上配置通道优先级。
- ◇ 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- ◇ 将 ICR 寄存器的 TTFH 位置 1 以清除 RIF 寄存器的 TTFH 标志。
- ◇ 在 DMA 控制寄存器上激活该通道。

注：若使用 FIFO，可一次写入多个传输字节数，并设定 FCON 寄存器中的 TXTH 位，可设定 FIFO 深度为 0,2,4,8 个字符时，新的数据将被传送到 TXBUF 寄存器中。

在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 IFM 寄存器的 TFEMPTY 标志；监视 RIF 寄存器的 TFEMPTY 标志可以确认 UART 通信是否结束。这样可以在关闭 UART 或进入停机模式之前避免破坏最后一次传输的数据。软件必须等待 TBC 被置 1。TBC 标志在全部数据发送期间会是零，并且在最后一帧数据发送出去之后会由硬件置 1。

### 利用 DMA 接收

使用 DMA 进行接收，可以通过设置 MCON 寄存器中的 RXDMAEN 使能。当收到一个字节数据时，从 RXBUF 寄存器取出来的数据会被转移到 DMA 外设中指向的 SRAM 区域，将一个 DMA 通道设置给 UART 接收，要按照下列步骤：

- ◇ 在 DMA 控制寄存器上将 RXBUF 寄存器地址配置成 DMA 传输的源地址。在每个 RFTH 事件后，数据将从这个地址取走。
- ◇ 在 DMA 控制寄存器上将内存地址配置成 DMA 传输的目标地址。在每个 RFTH 事件后，数据将从 RXBUF 寄存器取出，放入这个目标地址。
- ◇ 在 DMA 控制寄存器中配置要传输的总的字节数。
- ◇ 在 DMA 控制寄存器上配置通道优先级。
- ◇ 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
- ◇ 在 DMA 控制寄存器上激活该通道。

需要注意的是，若使用 FIFO，可一次接收多笔传输字节数，并设定 FCON 寄存器中的 RXTH，可设定 FIFO 深度为 1,4,8,14 个字符时，将新的数据将被传送到 RXBUF 寄存器中。

当传输完成时，若使能了 DMA 中断，DMA 控制器会产生中断。

多缓冲器通信中的错误标志和中断产生：在多缓冲器通信的情况下，通信期间如果发生任何错误，会在当前字节传输后将错误标志置 1。如果中断使能位被置 1，将产生中断。在单个字节接收的情况下，和 IFM 寄存器中的 RXBERR 和 RFOERR 一起被置起的帧错误和溢出错误，有单独的错误标志中断使能位；如果被置 1 了，会在当前字节传输结束后，产生中断。

### 30.4.15 中断配置

◇ UART\_IER 中断使能寄存器

中断使能寄存器，此位设定 1 时，表示开启中断功能，并且同时反映在 IVS 寄存器。此寄存器属于只写，并且只允许写入 1，无法写 0 取消使能中断设定。

◇ UART\_IDR 中断禁止寄存器

中断禁止寄存器，此位设定 1 时，表示关闭中断功能，并且同时反映在 IVS 寄存器。此寄存器属于只写，并且只允许写入 1，无法写 0 取消禁止中断设定。

◇ UART\_IVS 中断有效状态寄存器

中断有效状态寄存器，反映 IER 与 IDR 寄存器所设定的结果。0：中断禁止 1：中断使能

◇ UART\_RIF 原始中断标志寄存器

原始中断标志寄存器，反映所有发生中断事件的状态，无论 IVS 是否有使能中断，皆会反映在此寄存器中，主要提供用户监控无屏蔽的中断位，是否有错误事件发生。

◇ UART\_IFM 中断标志屏蔽寄存器

中断标志屏蔽寄存器，记录中断使能位所发生中断事件。0：无中断事件 1：发生中断事件。

◇ UART\_ICR 中断清除寄存器

中断清除寄存器，此位设定 1 时，清除中断标志 RIF 与 IFM，此寄存器属于写一清零，并且只允许写入 1 清除，无法写 0 清除。

在 UART 中，配置了 17 种中断，分别为下表。

中断事件	中断标志
接收器字节格式错误	RXBERR
自动波特率检测结束	ABEND
自动波特率检测超时	ABTO
CTS <sub>n</sub> 引脚电平改变	DCTS
接收超时	RXTO
地址匹配	ADDRM
LIN 断开检测	LINBK
块结束	EOB
噪声位检测	NOISE
接收器 FIFO 触发阈值	RFTH
接收器 FIFO 满	RFFULL
接收器 FIFO 溢出	RFOERR
接收器 FIFO 下溢	RFUERR
发送器字节完成	TBC
发送器 FIFO 触发阈值	TFTH
发送器 FIFO 空	TFEMPTY
发送器 FIFO 溢出	TFOVER

表 30-4 中断配置表

## 30.5 特殊功能寄存器

### 30.5.1 寄存器列表

UART 寄存器列表			
寄存器名称	偏移地址	类型	寄存器描述
UART0 基地址: 4000_4000 <sub>H</sub>			
UART1 基地址: 4000_4400 <sub>H</sub>			
UART2 基地址: 4000_4800 <sub>H</sub>			
UART_RXBUF	00 <sub>H</sub>	R	接收缓冲寄存器
UART_TXBUF	04 <sub>H</sub>	R/W	发送缓冲寄存器
UART_BRR	08 <sub>H</sub>	R/W	波特率寄存器
UART_LCON	0C <sub>H</sub>	R/W	格式控制寄存器
UART_MCON	10 <sub>H</sub>	R/W	模式控制寄存器
UART_RS485	14 <sub>H</sub>	R/W	RS485 控制寄存器
UART_SCARD	18 <sub>H</sub>	R/W	智能卡控制寄存器
UART_LIN	1C <sub>H</sub>	R/W	LIN 控制寄存器
UART_RTOR	20 <sub>H</sub>	R/W	接收超时寄存器
UART_FCON	24 <sub>H</sub>	R/W	FIFO 控制寄存器
UART_STAT	28 <sub>H</sub>	R	状态寄存器
UART_IER	2C <sub>H</sub>	W1	中断使能寄存器
UART_IDR	30 <sub>H</sub>	W1	中断禁止寄存器
UART_IVS	34 <sub>H</sub>	R	中断有效位状态寄存器
UART_RIF	38 <sub>H</sub>	R	无屏蔽中断标志寄存器
UART_IFM	3C <sub>H</sub>	R	屏蔽中断禁止寄存器
UART_ICR	40 <sub>H</sub>	C_W1	中断清除寄存器

### 30.5.2 寄存器描述

#### 30.5.2.1 UART接收缓冲寄存器 (UART\_RXBUF)

UART 接收缓冲寄存器 (UART_RXBUF)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXBUF															

Reserved	Bit 31-9	—	保留
RXBUF	Bit 8-0	R	<b>接收缓冲寄存器</b> 包含接收到的字节。 RXBUF寄存器提供接收移位寄存器和内部总线间的并行接口。 注: Bit 8用于RS485寻址模式

#### 30.5.2.2 UART发送缓冲寄存器 (UART\_TXBUF)

UART 发送缓冲寄存器 (UART_TXBUF)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXBUF															

Reserved	Bit 31-9	—	保留
TXBUF	Bit 8-0	R/W	<b>发送缓冲寄存器</b> 用于写入要发送的数据字节。 TXBUF寄存器提供发送移位寄存器与内部总线间的并行接口。 注: Bit 8用于RS485寻址模式

### 30.5.2.3 UART波特率寄存器 (UART\_BRR)

UART 波特率寄存器 (UART_BRR)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																BRR															

Reserved	Bit 31-16	—	保留
BRR	Bit 15-0	R/W	<b>波特率寄存器</b> 整数部分BRR<15:4> 小数部分BRR<3:0> 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 注：使用自动波特率功能时则可自动写入

### 30.5.2.4 UART线控寄存器 (UART\_LCON)

UART 线控寄存器 (UART_LCON)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXEN	RXEN	DBCEN	Reserved	BREAK	Reserved	TXINV	RXINV	DATAINV	MSB	PS	PE	STOP	DLS		

Reserved	Bit 31-16	—	保留
TXEN	Bit 15	R/W	<b>发送器使能</b> 使能发送器，此位由软件设置1和清除。 0: 发送器禁止 1: 发送器使能
RXEN	Bit 14	R/W	<b>接收器使能</b> 使能接收器，此位由软件设置1和清除。 0: 接收器禁止 1: 接收器使能
DBCEN	Bit 13	R/W	<b>防抖动使能</b> 使能防抖动功能，此位由软件设置1和清除。 0: 防抖动禁止 1: 防抖动使能，在RX线上须维持8个模块时钟 (PCLK) 的电平
Reserved	Bit 12-11	—	保留
BREAK	Bit 10	R/W	<b>断开使能</b> 使能断开功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: 断开禁止 1: 断开使能，会使得TX输出为0
Reserved	Bit 9	—	保留
TXINV	Bit 8	R/W	<b>TX引脚电平反向</b> TX引脚反向功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: TX引脚信号工作于标准逻辑电平 (VDD=1/idle, Gnd=0/mark) 1: TX引脚信号反向 (VDD=0/mark, Gnd=1/idle)。 此功能可用于TX线上带有外部反向器时
RXINV	Bit 7	R/W	<b>RX引脚电平反向</b> RX引脚反向功能，此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时

			才可以写入。 0: RX引脚信号工作于标准逻辑电平 (VDD=1/idle, Gnd=0/mark) 1: RX引脚信号反向 (VDD=0/mark, Gnd=1/idle)。此功能可用于RX线上带有外部反向器时
DATAINV	Bit 6	R/W	<b>二进制反向使能</b> 二进制反向功能, 此位由软件设置1和清除。此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: 缓冲寄存器中的逻辑数据在接收的时候, 采用正/直接逻辑。(1=H, 0=L) 1: 缓冲寄存器中的逻辑数据在接收的时候, 采用负/反向逻辑。(1=L, 0=H)
MSB	Bit 5	R/W	<b>高位在前使能</b> 高位在前功能, 此位由软件设置1和清除。此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: 数据在发送和接收的时候, 采用起始位后面跟着第0位的顺序 1: 数据在发送和接收的时候, 采用起始位后面跟着的最高位的顺序
PS	Bit 4	R/W	<b>校验位奇偶选择</b> 当使能校验功能时, 选择校验位为奇校验或偶校验, 此位由软件设置1和清除。此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: 奇校验 1: 偶校验
PE	Bit 3	R/W	<b>校验使能</b> 使能校验功能, 计算好的校验位被插入到最高位, 并检测接收数据的校验位 (接收与发送功能), 此位由软件设置1和清除。此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 0: 校验位禁止 1: 校验位使能
STOP	Bit 2	R/W	<b>停止位选择</b> 此位由软件设置1和清除。此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 普通模式下 0: 1个停止位 1: 2个停止位 (在5字长模式为1.5个停止位) 智能卡模式



			0: 0.5个停止位 1: 1.5个停止位
DLS	Bit 1-0	R/W	<b>数据字长选择</b> 此位由软件设置1和清除。 此位在LCON寄存器中的RXEN与TXEN位为0时才可以写入。 00: 8位字长 01: 7位字长 10: 6位字长 11: 5位字长

### 30.5.2.5 UART模式控制寄存器 (UART\_MCON)

UART 模式控制寄存器 (UART_MCON)																																												
偏移地址: 10 <sub>H</sub>																																												
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Reserved																TXDMAEN	RXDMAEN	Reserved	ABRREPT	ABRMOD	ABREN	Reserved	BKREQ	HDEN	IREN	AFCEN	RTSSET	LPBKEN																

Reserved	Bit 31-16	—	保留
TXDMAEN	Bit 15	R/W	<b>发送器DMA使能</b> 此位由软件设置1和清除。 0: 发送器DMA通讯禁止 1: 发送器DMA通讯使能
RXDMAEN	Bit 14	R/W	<b>接收器DMA使能</b> 此位由软件设置1和清除。 0: 接收器DMA通讯禁止 1: 接收器DMA通讯使能
Reserved	Bit 13-12	—	保留
ABRREPT	Bit 11	R/W	<b>重复自动波特率检测</b> 在使能自动波特率检测时, 当产生波特率超时并不会关闭自动波特率检测功能, 并在下一个下降沿时重复自动波特率检测, 此位由软件设置1和清除。 0: 重复自动波特率检测禁止 1: 重复自动波特率检测使能
ABRMOD	Bit 10-9	R/W	<b>自动波特率模式选择</b> 此位由软件设置1和清除。 00: 模式0, 检测第一个下降沿到第二个下降沿时间 (检测2位) 01: 模式1, 检测第一个下降沿到第一个上升沿时间 (检测1位) 10: 模式2, 检测第一个下降沿到第一个上升沿时间 (检测2位) 11: 保留
ABREN	Bit 8	R/W	<b>自动波特率使能</b> 此位在使能后并完成自动波特率检测后会自动清除, 也可由软件设置1和清除。 0: 自动波特率禁止 1: 自动波特率使能
Reserved	Bit 7-6	—	保留
BKREQ	Bit 5	W	<b>断开请求</b> 此位在写入后的下一个时钟会自动清除。

			<p>0: 断开请求禁止 1: 断开请求使能, 根据设定的N位长 (8/7/6/5) 产生N个低脉冲信号</p>
HDEN	Bit 4	R/W	<p><b>单线半双工使能</b> 此位由软件设置1和清除。 0: 单线半双工禁止 1: 单线半双工使能</p>
IREN	Bit 3	R/W	<p><b>IrDA红外线模式使能</b> 此位由软件设置1和清除。 0: IrDA红外线模式禁止 1: IrDA红外线模式使能</p>
AFCEN	Bit 2	R/W	<p><b>自动流量控制使能</b> 此位由软件设置1和清除。 0: 自动流量控制禁止 1: 自动流量控制使能</p>
RTSSET	Bit 1	R/W	<p><b>RTSn设置控制</b> 此位由软件设置1和清除。 0: 自动流量控制禁止时, RTSn引脚输出高电平 1: 自动流量控制禁止时, RTSn引脚输出低电平</p>
LPBKEN	Bit 0	R/W	<p><b>回送模式使能</b> 此模式是用于UART测试项目的诊断模式, 在UART普通模式下运行, TX引脚输出为高电平, 串行的数据在内部回送至RX。在此模式下, 所有中断都是正常运行的, 此位由软件设置1和清除。 0: 回送模式禁止 1: 回送模式使能</p>

### 30.5.2.6 UART RS485 控制寄存器 (UART\_RS485)

UART RS485 控制寄存器 (UART_RS485)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DLY								ADDR								Reserved				AADINV	AADACEN	AADNEN	AADEN

Reserved	Bit 31-24	—	保留
DLY	Bit 23-16	R/W	<b>延迟数值</b> 用于设置延迟RTSn的输出时间, 是由一个8位计数器计数, 此位由软件设置和清除
ADDR	Bit 15-8	R/W	<b>地址匹配值</b> 用于多机通讯时地址标记的检测。 接收器在RS485自动检测模式时, 当接收数据的最高位为1且匹配ADDR, 数据才允许接收至FIFO中, 否则舍弃此数值, 此位由软件设置和清除。
Reserved	Bit 7-4	—	保留
AADINV	Bit 3	R/W	<b>驱动使能反向</b> 在自动流量控制模式时, 设置驱动使能引脚 (RTSn/DE) 的输出电平, 此位由软件设置和清除。 0: 当发送器开始发送数据时, 驱动使能引脚输出0, 发送完成且FIFO内无数据时, 驱动使能引脚输出1 1: 当发送器开始发送数据时, 驱动使能引脚输出1, 发送完成且FIFO内无数据时, 驱动使能引脚输出0
AADACEN	Bit 2	R/W	<b>自动流量控制模式使能</b> 此位由软件设置1和清除。 0: 自动流量控制模式禁止 1: 自动流量控制模式使能
AADNEN	Bit 1	R/W	<b>普通模式使能</b> 此位由软件设置1和清除。 0: 普通模式禁止 1: 普通模式使能, 接收数据的地址位为第8位 (UART_RXBUF)
AADEN	Bit 0	R/W	<b>自动地址检测模式使能</b> 在普通模式时, 设置此位无效, 此位由软件设置1和清除。 0: 自动地址检测模式禁止 1: 自动地址检测模式使能, 当接收数据的地址位为1且匹配ADDR时, 才会将数据接收至FIFO中

### 30.5.2.7 UART智能卡控制寄存器 (UART\_SCARD)

UART 智能卡控制寄存器 (UART_SCARD)																																	
偏移地址: 18 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
BLEN								GT								PSC								Reserved		SCCNT			SCLKEN		SCNACK		SCEN

BLEN	Bit 31-24	R/W	<p><b>块长度</b></p> <p>设置了智能卡模式T=1的接收时的块长度，此位由软件设置1和清除。</p> <p>例如</p> <p>BLEN = 0 -&gt; 0个信号字符</p> <p>BLEN = 1 -&gt; 1个信号字符</p> <p>BLEN = 255 -&gt; 255个信号字符</p> <p>这个功能也可以在其他模式中使用，当LCON寄存器的RXEN位清除时，块长度计数器会重新计数</p>
GT	Bit 23-16	R/W	<p><b>保护时间</b></p> <p>设置保护时间长度，是使用波特时钟为单位。</p> <p>在智能卡模式中使用，完成标志（RIF寄存器TBC位）在保护时间过后置起，此位由软件设置1和清除</p>
PSC	Bit 15-8	R/W	<p><b>分频器数值</b></p> <p>此位由软件设置1和清除。</p> <p>智能卡模式：</p> <p>PSC&lt;4:0&gt;：输出时钟分频数值</p> <p>用于设定UART时钟的分频数，得到智能卡输出时钟，由五个有效位组成，乘以2得到的数值作为分频</p> <p>00000：保留</p> <p>00001：2分频</p> <p>00010：4分频</p> <p>00011：6分频</p>
Reserved	Bit 7-6	—	—
SCCNT	Bit 5-3	R/W	<p><b>智能卡重试计数器</b></p> <p>设置智能卡模式中接收和发送的重试次数。此位由软件设置1和清除。</p> <p>在发送模式下，在产生帧错误前重试发送的次数</p> <p>在接收模式下，在接收到NACK后重试接收的次数</p> <p>0x0：重试功能关闭，在发送与接收模式下不进行自动重试</p>

			0x1至0x7: 在产生错误前自动重试的次数
SCLKEN	Bit 2	R/W	<b>智能卡时钟使能</b> 此位由软件设置1和清除。 0: CK引脚禁止 1: CK引脚使能
SCNACK	Bit 1	R/W	<b>智能卡NACK发送使能</b> 此位由软件设置1和清除。 0: 出现校验错误时禁止发送NACK信号 1: 出现校验错误时使能发送NACK信号
SCEN	Bit 0	R/W	<b>智能卡模式使能</b> 此位由软件设置1和清除。 0: 智能卡模式禁止 1: 智能卡模式使能

### 30.5.2.8 UART LIN控制寄存器 (UART\_LIN)

UART LIN 控制寄存器 (UART_LIN)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										LINBKREQ		LINBDL		LINEN	

Reserved	Bit 31-3	—	保留
LINBKREQ	Bit 2	W1	<b>LIN模式断开请求</b> 在LIN模式下, 发送器将发送13位0作为断开符号后, 发送2位1用于对下一个开始位的检测。此位由软件设置1并在下一个时钟后自动清除。 0: LIN模式断开请求禁止 1: LIN模式断开请求使能
LINBDL	Bit 1	R/W	<b>LIN模式断开字长</b> 此位由软件设置1和清除。 0: 10位断开字符检测 1: 11位断开字符检测
LINEN	Bit 0	R/W	<b>LIN模式使能</b> 此位由软件设置1和清除。 0: LIN模式禁止 1: LIN模式使能

### 30.5.2.9 UART接收超时寄存器 (UART\_RTOR)

UART 超时接收寄存器 (UART_RTOR)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							RTOEN	RTO																							

Reserved	Bit 31-25	—	保留
RTOEN	Bit 24	R/W	<b>接收器超时使能</b> 此位由软件设置1和清除。 0: 接收器超时禁止 1: 接收器超时使能
RTO	Bit 23-0	R/W	<b>接收器超时数值</b> 设置接收超时时间，使用波特率时钟的字长为单 位。在标准模式下，接收最后一个字节后，在超时 时间内未检测到新的起始位，将置起 RIF 寄存器的 RXTO 位，此位由软件设置和清除。

### 30.5.2.10 UART FIFO控制寄存器 (UART\_FCON)

UART FIFO 控制寄存器 (UART_FCON)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TXFL				TXTH		TFRST	RXFL				RXTH		RFRST		

Reserved	Bit 31-16	—	保留
TXFL	Bit 15-11	R	<b>发送器FIFO中数据个数</b> 显示发送器FIFO内准备发送的个数 (0-16), 此位由硬件设置且只能读取。
TXTH	Bit 10-9	R/W	<b>发送器触发阈值</b> 设置发送器触发阈值, 当TXFL个数小于TXTH设定的个数时, 将置起RIF寄存器的TFTH位, 与STAT寄存器的TFTH位 00: TX FIFO内无字符 01: TX FIFO有2个字符 10: TX FIFO有4个字符 11: TX FIFO有8个字符
TFRST	Bit 8	W1	<b>发送器FIFO重置使能</b> 设置清除TX FIFO内字符, 此位由软件设置1且在下一个时钟自动清除。 0: 发送器FIFO重置禁止 1: 发送器FIFO重置使能
RXFL	Bit 7-3	R	<b>接收器FIFO中数据个数</b> 显示接收器FIFO内准备接收的个数 (0-16), 此位由硬件设置且只能读取
RXTH	Bit 2-1	R/W	<b>接收器触发阈值</b> 设置接收器触发阈值, 当RXFL个数大于等于RXTH设定的个数时, 将置起RIF寄存器的RFTH位, 与STAT寄存器的RFTH位 00: RX FIFO有1字符 01: RX FIFO有4个字符 10: RX FIFO有8个字符 11: RX FIFO有14个字符
RFRST	Bit 0	W1	<b>接收器FIFO重置</b> 设置清除RX FIFO内字符, 此位由软件设置1且在下一个时钟自动清除。 0: 接收器FIFO重置禁止 1: 接收器FIFO重置使能



### 30.5.2.11 UART状态寄存器 (UART\_STAT)

UART 状态寄存器 (UART_STAT)																																
偏移地址: 28 <sub>H</sub>																																
复位值: 00000000_00000001_10000100_00001000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved													TFOERR	TFFULL	TFEMPTY	TFTH	TSBUSY	RFUERR	RFOERR	RFFULL	RFEMPTY	RFTH	RSBUSY	Reserved					CTSSTA	BKERR	FERR	PERR

Reserved	Bit 31-19	—	保留
TFOERR	Bit 18	R	<b>发送器FIFO溢出错误</b> 当发送器FIFO内已有16个数据时, 有新数据再次写入FIFO中时, 将会置起此位并舍弃新数据。此位由硬件设置1, 在发送数据时清除或设置ICR寄存器的TFOVER位清除 0: 发送器FIFO溢出错误未产生 1: 发送器FIFO溢出错误产生
TFFULL	Bit 17	R	<b>发送器FIFO满</b> 当发送器FIFO内有16个数据时, 此位由硬件设置1, 在FIFO内未满足16个数据时清除。 0: 发送器FIFO未满足16个数据 1: 发送器FIFO满足16个数据
TFEMPTY	Bit 16	R	<b>发送器FIFO空</b> 当发送器FIFO内无任何数据时, 此位由硬件设置1, 在FIFO内写入数据时清除。 0: 发送器FIFO有数据 1: 发送器FIFO无数据
TFTH	Bit 15	R	<b>发送器FIFO触发阈值</b> 当FCON寄存器的TXFL位所指示的FIFO字节数小于FCON寄存器的TXTH设定的阈值, 将会置起此位。此位由硬件设置1和在未满足阈值水平时清除 0: 发送器FIFO未小于阈值水平 1: 发送器FIFO小于阈值水平
TSBUSY	Bit 14	R	<b>发送器移位寄存器忙碌</b> 当写入数据至FIFO中由硬件设置1在发送最后一个数据完成后清除。 0: 发送器FIFO内无数据等待传送 1: 发送器FIFO内有数据等待传送且未发送完最后一个数据
RFUERR	Bit 13	R	<b>接收器FIFO下溢错误</b> 当接收器FIFO内无数据时, 又再次读取接收器FIFO时, 将会置起此位。此位由硬件设置1, 在接

			收数据时清除或设置ICR寄存器的RFUERR位清除 0: 接收器FIFO下溢错误未产生 1: 接收器FIFO下溢错误产生
RFOERR	Bit 12	R	<b>接收器FIFO溢出错误</b> 当接收器FIFO内已有16个数据时, 有新数据再次接收至FIFO中时, 将会置起此位并舍弃新数据。此位由硬件设置1, 在读取数据时清除或设置ICR寄存器的RFOERR位清除 0: 接收器FIFO溢出错误未产生 1: 接收器FIFO溢出错误产生
RFFULL	Bit 11	R	<b>接收器FIFO满</b> 当接收器FIFO内有16个数据时, 此位由硬件设置1, 在FIFO内未满足16个数据时清除。 0: 接收器FIFO未满足16个数据 1: 接收器FIFO满足16个数据
RFEMPTY	Bit 10	R	<b>接收器FIFO空</b> 当接收器FIFO内无任何数据时, 此位由硬件设置1, 接收数据至FIFO内时清除。 0: 接收器FIFO有数据 1: 接收器FIFO无数据
RFTH	Bit 9	R	<b>接收器FIFO触发阈值</b> 当FCON寄存器的RXFL位所指示的FIFO字节数大于等于FCON寄存器的RXTH设定的阈值, 将会置起此位。此位由硬件设置1, 在未满足阈值水平时清除 0: 接收器FIFO未大于等于阈值水平 1: 接收器FIFO大于等于阈值水平
RSBUSY	Bit 8	R	<b>接收移位寄存器忙碌</b> 当接收数据时, 由硬件设置1在完成接收数据后清除 0: 接收器未接收数据 1: 接收器正在接收数据
Reserved	Bits 7-4	—	保留
CTSSTA	Bit 3	R	<b>CTS<sub>n</sub>状态</b> 此位显示CTS <sub>n</sub> 输入引脚状态, 由硬件设置1和清除。 0: CTS <sub>n</sub> 输入引脚为0 1: CTS <sub>n</sub> 输入引脚为1
BKERR	Bit 2	R	<b>断开错误</b> 当接收数据与停止位皆为0时, 将由硬件设置1。此位为显示当前读取接收器FIFO的数值。 0: 断开错误未产生 1: 断开错误产生

FERR	Bit 1	R	<p><b>帧错误</b>                      当接收数据的停止位为0时，将由硬件设置1。此位为显示当前读取接收器FIFO的数值。                      0：帧错误未产生                      1：帧错误产生</p>
PERR	Bit 0	R	<p><b>校验错误</b>                      当接收数据的校验位接收错误时，将由硬件设置1。此位为显示当前读取接收器FIFO的数值。                      0：校验错误未产生                      1：校验错误产生</p>

### 30.5.2.12 UART中断使能寄存器 (UART\_IER)

UART 中断使能寄存器 (UART_IER)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	W1	发送器FIFO溢出中断使能 0: 写入0无效 1: 发送器FIFO溢出中断使能
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	W1	发送器FIFO空中断使能 0: 写入0无效 1: 发送器FIFO空中断使能
TFTH	Bit 15	W1	发送器FIFO触发阈值中断使能 0: 写入0无效 1: 发送器FIFO触发阈值中断使能
TBC	Bit 14	W1	发送器字节完成中断使能 0: 写入0无效 1: 发送器字节完成中断使能
RFUERR	Bit 13	W1	接收器FIFO下溢中断使能 0: 写入0无效 1: 接收器FIFO下溢中断使能
RFOERR	Bit 12	W1	接收器FIFO溢出中断使能 0: 写入0无效 1: 接收器FIFO溢出中断使能
RFFULL	Bit 11	W1	接收器FIFO满中断使能 0: 写入0无效 1: 接收器FIFO满中断使能
Reserved	Bit 10	—	保留
RFTH	Bit 9	W1	接收器FIFO触发阈值中断使能 0: 写入0无效 1: 接收器FIFO触发阈值中断使能
NOISE	Bit 8	W1	噪声位检测中断使能 0: 写入0无效 1: 噪声位检测中断使能
EOB	Bit 7	W1	块结束中断使能 0: 写入0无效

			1: 块结束中断使能
LINBK	Bit 6	W1	<b>LIN断开检测中断使能</b> 0: 写入0无效 1: LIN断开检测中断使能
ADDRM	Bit 5	W1	<b>地址匹配中断使能</b> 0: 写入0无效 1: 地址匹配中断使能
RXTO	Bit 4	W1	<b>接收超时中断使能</b> 0: 写入0无效 1: 接收超时中断使能
DCTS	Bit 3	W1	<b>CTS<sub>n</sub>引脚电平改变中断使能</b> 0: 写入0无效 1: CTS <sub>n</sub> 引脚电平改变中断使能
ABTO	Bit 2	W1	<b>自动波特率检测超时中断使能</b> 0: 写入0无效 1: 自动波特率检测超时中断使能
ABEND	Bit 1	W1	<b>自动波特率检测结束中断使能</b> 0: 写入0无效 1: 自动波特率检测结束中断使能
RXBERR	Bit 0	W1	<b>接收器字节格式错误中断使能</b> 0: 写入0无效 1: 接收器字节格式错误中断使能

### 30.5.2.13 UART中断禁止寄存器 (UART\_IDR)

UART 中断禁止寄存器 (UART_IDR)																																											
偏移地址: 30 <sub>H</sub>																																											
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR												

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	W1	发送器FIFO溢出中断禁止 0: 写入0无效 1: 发送器FIFO溢出中断禁止
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	W1	发送器FIFO空中断禁止 0: 写入0无效 1: 发送器FIFO空中断禁止
TFTH	Bit 15	W1	发送器FIFO触发阈值中断禁止 0: 写入0无效 1: 发送器FIFO触发阈值中断禁止
TBC	Bit 14	W1	发送器字节完成中断禁止 0: 写入0无效 1: 发送器字节完成中断禁止
RFUERR	Bit 13	W1	接收器FIFO下溢中断禁止 0: 写入0无效 1: 接收器FIFO下溢中断禁止
RFOERR	Bit 12	W1	接收器FIFO溢出中断禁止 0: 写入0无效 1: 接收器FIFO溢出中断禁止
RFFULL	Bit 11	W1	接收器FIFO满中断禁止 0: 写入0无效 1: 接收器FIFO满中断禁止
Reserved	Bit 10	—	保留
RFTH	Bit 9	W1	接收器FIFO触发阈值中断禁止 0: 写入0无效 1: 接收器FIFO触发阈值中断禁止
NOISE	Bit 8	W1	噪声位检测中断禁止 0: 写入0无效 1: 噪声位检测中断禁止
EOB	Bit 7	W1	块结束中断禁止 0: 写入0无效

			1: 块结束中断禁止
LINBK	Bit 6	W1	<b>LIN断开检测中断禁止</b> 0: 写入0无效 1: LIN断开检测中断禁止
ADDRM	Bit 5	W1	<b>地址匹配中断禁止</b> 0: 写入0无效 1: 地址匹配中断禁止
RXTO	Bit 4	W1	<b>接收超时中断禁止</b> 0: 写入0无效 1: 接收超时中断禁止
DCTS	Bit 3	W1	<b>CTS<sub>n</sub>引脚电平改变中断禁止</b> 0: 写入0无效 1: CTS <sub>n</sub> 引脚电平改变中断禁止
ABTO	Bit 2	W1	<b>自动波特率检测超时中断禁止</b> 0: 写入0无效 1: 自动波特率检测超时中断禁止
ABEND	Bit 1	W1	<b>自动波特率检测结束中断禁止</b> 0: 写入0无效 1: 自动波特率检测结束中断禁止
RXBERR	Bit 0	W1	<b>接收器字节格式错误中断禁止</b> 0: 写入0无效 1: 接收器字节格式错误中断禁止

### 30.5.2.14 UART中断有效状态寄存器 (UART\_IVS)

UART 中断有效状态寄存器 (UART_IVS)																																											
偏移地址: 34 <sub>H</sub>																																											
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR												

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	R	发送器FIFO溢出中断有效位 0: 发送器FIFO溢出中断禁止 1: 发送器FIFO溢出中断使能
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	R	发送器FIFO空中断有效位 0: 发送器FIFO空中断禁止 1: 发送器FIFO空中断使能
TFTH	Bit 15	R	发送器FIFO触发阈值中断有效位 0: 发送器FIFO触发阈值中断禁止 1: 发送器FIFO触发阈值中断使能
TBC	Bit 14	R	发送器字节完成中断有效位 0: 发送器字节完成中断禁止 1: 发送器字节完成中断使能
RFUERR	Bit 13	R	接收器FIFO下溢中断有效位 0: 接收器FIFO下溢中断禁止 1: 接收器FIFO下溢中断使能
RFOERR	Bit 12	R	接收器FIFO溢出中断有效位 0: 接收器FIFO溢出中断禁止 1: 接收器FIFO溢出中断使能
RFFULL	Bit 11	R	接收器FIFO满中断有效位 0: 接收器FIFO满中断禁止 1: 接收器FIFO满中断使能
Reserved	Bit 10	—	保留
RFTH	Bit 9	R	接收器FIFO触发阈值中断有效位 0: 接收器FIFO触发阈值中断禁止 1: 接收器FIFO触发阈值中断使能
NOISE	Bit 8	R	噪声位检测中断有效位 0: 噪声位检测中断禁止 1: 噪声位检测中断使能
EOB	Bit 7	R	块结束中断有效位 0: 块结束中断禁止



			1: 块结束中断使能
LINBK	Bit 6	R	<b>LIN断开检测中断有效位</b> 0: LIN断开检测中断禁止 1: LIN断开检测中断使能
ADDRM	Bit 5	R	<b>地址匹配中断有效位</b> 0: 地址匹配中断禁止 1: 地址匹配中断使能
RXTO	Bit 4	R	<b>接收超时中断有效位</b> 0: 接收超时中断禁止 1: 接收超时中断使能
DCTS	Bit 3	R	<b>CTS<sub>n</sub>引脚电平改变中断有效位</b> 0: CTS <sub>n</sub> 引脚电平改变中断禁止 1: CTS <sub>n</sub> 引脚电平改变中断使能
ABTO	Bit 2	R	<b>自动波特率检测超时中断有效位</b> 0: 自动波特率检测超时中断禁止 1: 自动波特率检测超时中断使能
ABEND	Bit 1	R	<b>自动波特率检测结束中断有效位</b> 0: 自动波特率检测结束中断禁止 1: 自动波特率检测结束中断使能
RXBERR	Bit 0	R	<b>接收器字节格式错误中断有效位</b> 0: 接收器字节格式错误中断禁止 1: 接收器字节格式错误中断使能

### 30.5.2.15 UART原始中断标志寄存器 (UART\_RIF)

UART 原始中断标志寄存器 (UART_RIF)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	R	<p><b>发送器FIFO溢出中断标志</b> 当发送器FIFO内已有16个数据时，有新数据再次写入FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，设置ICR寄存器的TFOVER位清除</p> <p>0: 无中断产生 1: 发送器FIFO溢出中断产生</p>
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	R	<p><b>发送器FIFO空中断标志</b> 发送器FIFO由一个数值转为空时置起，或设置IER寄存器的TFEMPTY位时，UART会判断当前发送器FIFO是否为空将此位置起。此位由硬件设置1，设置ICR寄存器的TFEMPTY位清除</p> <p>0: 无中断产生 1: 发送器FIFO空中断产生</p>
TFTH	Bit 15	R	<p><b>发送器FIFO触发阈值中断标志</b> 发送器FIFO个数达到发送器设定的阈值时置起，或设置IER寄存器的TFTH位时，UART会判断当前发送器FIFO个数达到发送器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的TFTH位清除</p> <p>0: 无中断产生 1: 发送器FIFO触发阈值中断产生</p>
TBC	Bit 14	R	<p><b>发送器字节完成中断标志</b> 发送器完成单个字节时置起。此位由硬件设置1，设置ICR寄存器的TBC位清除</p> <p>0: 无中断产生 1: 发送器字节完成中断产生</p>
RFUERR	Bit 13	R	<p><b>接收器FIFO下溢中断标志</b> 当接收器FIFO内无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除</p>

			0: 无中断产生 1: 接收器FIFO下溢中断产生
RFOERR	Bit 12	R	<b>接收器FIFO溢出中断标志</b> 当接收器FIFO内已有16个数据时, 有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1, 设置ICR寄存器的RFOERR位清除 0: 无中断产生 1: 接收器FIFO溢出中断产生
RFFULL	Bit 11	R	<b>接收器FIFO满中断标志</b> 当接收器FIFO内由15个数据接收至16个数据时置起, 或设置IER寄存器的RFFULL位时, UART会判断当前接收器FIFO是否为16个数据而置起。此位由硬件设置1, 设置ICR寄存器的RFFULL位清除 0: 无中断产生 1: 接收器FIFO满中断产生
Reserved	Bit 10	—	保留
RFTH	Bit 9	R	<b>接收器FIFO触发阈值中断标志</b> 接收器FIFO个数达到接收器设定的阈值时置起, 或设置IER寄存器的RFTH位时, UART会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1, 设置ICR寄存器的RFTH位清除 0: 无中断产生 1: 接收器FIFO触发阈值中断产生
NOISE	Bit 8	R	<b>噪声位检测中断标志</b> 此位由硬件设置1, 设置ICR寄存器的NOISE位清除 0: 无中断产生 1: 噪声位检测中断产生
EOB	Bit 7	R	<b>块结束中断标志</b> 接收器接收个数等于SCARD寄存器的BLEN位时置起。此位由硬件设置1, 设置ICR寄存器的EOB位清除 0: 无中断产生 1: 块结束中断产生
LINBK	Bit 6	R	<b>LIN断开检测中断标志</b> 在LIN模式中, 当接收器检测到断开字符时置起。此位由硬件设置1, 设置ICR寄存器的LINBK位清除 0: 无中断产生 1: LIN断开检测中断产生
ADDRM	Bit 5	R	<b>地址匹配中断标志</b> 接收器接收到的字符匹配RS485寄存器的ADDR位时置起。此位由硬件设置1, 设置ICR寄存器的

			<p>ADDRM位清除</p> <p>0: 无中断产生</p> <p>1: 地址匹配中断产生</p>
RXTO	Bit 4	R	<p><b>接收超时中断标志</b></p> <p>接收最后一个字节后, 在超时时间内未检测到新的起始位时置起。此位由硬件设置1, 设置ICR寄存器的RXTO位清除</p> <p>0: 无中断产生</p> <p>1: 接收超时中断产生</p>
DCTS	Bit 3	R	<p><b>CTS<sub>n</sub>引脚电平改变中断标志</b></p> <p>CTS<sub>n</sub>引脚电平改变时置起。此位由硬件设置1, 设置ICR寄存器的DCTS位清除</p> <p>0: 无中断产生</p> <p>1: CTS<sub>n</sub>引脚电平改变中断产生</p>
ABTO	Bit 2	R	<p><b>自动波特率检测超时中断标志</b></p> <p>自动波特率在检测到下降沿后, 在超时时间内未检测到上升沿时置起。此位由硬件设置1, 设置ICR寄存器的ABTO位清除</p> <p>0: 无中断产生</p> <p>1: 自动波特率检测超时中断产生</p>
ABEND	Bit 1	R	<p><b>自动波特率检测结束中断标志</b></p> <p>自动波特率检测完成时置起。此位由硬件设置1, 设置ICR寄存器的ABEND位清除</p> <p>0: 无中断产生</p> <p>1: 自动波特率检测结束中断产生</p>
RXBERR	Bit 0	R	<p><b>接收器字节格式错误中断标志</b></p> <p>接收器接收到的字符发生校验错误与帧错误时置起。此位由硬件设置1, 设置ICR寄存器的RXBERR位清除</p> <p>0: 无中断产生</p> <p>1: 接收器字节格式错误中断产生</p>

### 30.5.2.16 UART中断标志屏蔽寄存器 (UART\_IFM)

UART 中断标志屏蔽寄存器 (UART_IFM)																																											
偏移地址: 3C <sub>H</sub>																																											
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR												

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	R	<p><b>发送器FIFO溢出中断屏蔽标志</b></p> <p>当IVS寄存器的TFOVER位为1时，发送器FIFO内已有16个数据时，有新数据再次写入FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，设置ICR寄存器的TFOVER位清除</p> <p>0: 无中断产生 1: 发送器FIFO溢出中断产生</p>
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	R	<p><b>发送器FIFO空中断屏蔽标志</b></p> <p>当IVS寄存器的TFEMPTY位为1时，发送器FIFO由一个数值转为空时置起，或设置IER寄存器的TFEMPTY位时，UART会判断当前发送器FIFO是否为空将此位置起。此位由硬件设置1，设置ICR寄存器的TFEMPTY位清除</p> <p>0: 无中断产生 1: 发送器FIFO空中断产生</p>
TFTH	Bit 15	R	<p><b>发送器FIFO触发阈值中断屏蔽标志</b></p> <p>当IVS寄存器的TFTH位为1时，发送器FIFO个数达到发送器设定的阈值时置起，或设置IER寄存器的TFTH位时，UART会判断当前发送器FIFO个数达到发送器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的TFTH位清除</p> <p>0: 无中断产生 1: 发送器FIFO触发阈值中断产生</p>
TBC	Bit 14	R	<p><b>发送器字节完成中断屏蔽标志</b></p> <p>当IVS寄存器的TBC位为1时，发送器完成单个字节时置起。此位由硬件设置1，设置ICR寄存器的TBC位清除</p> <p>0: 无中断产生 1: 发送器字节完成中断产生</p>
RFUERR	Bit 13	R	<p><b>接收器FIFO下溢中断屏蔽标志</b></p> <p>当IVS寄存器的RFUERR位为1时，接收器FIFO内</p>

			无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除 0：无中断产生 1：接收器FIFO下溢中断产生
RFOERR	Bit 12	R	<b>接收器FIFO溢出中断屏蔽标志</b> 当IVS寄存器的RFOERR位为1时，接收器FIFO内已有16个数据时，有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1，设置ICR寄存器的RFOERR位清除 0：无中断产生 1：接收器FIFO溢出中断产生
RFFULL	Bit 11	R	<b>接收器FIFO满中断屏蔽标志</b> 当IVS寄存器的RFFULL位为1时，接收器FIFO内由15个数据接收至16个数据时置起，或设置IER寄存器的RFFULL位时，UART会判断当前接收器FIFO是否为16个数据而置起。此位由硬件设置1，设置ICR寄存器的RFFULL位清除 0：无中断产生 1：接收器FIFO满中断产生
Reserved	Bit 10	—	保留
RFTH	Bit 9	R	<b>接收器FIFO触发阈值中断屏蔽标志</b> 当IVS寄存器的RFTH位为1时，接收器FIFO个数达到接收器设定的阈值时置起，或设置IER寄存器的RFTH位时，UART会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的RFTH位清除 0：无中断产生 1：接收器FIFO触发阈值中断产生
NOISE	Bit 8	R	<b>噪声位检测中断屏蔽标志</b> 当IVS寄存器的NOISE位为1时，如噪声位检测中断产生该位置起，此位由硬件设置1，设置ICR寄存器的NOISE位清除 0：无中断产生 1：噪声位检测中断产生
EOB	Bit 7	R	<b>块结束中断屏蔽标志</b> 当IVS寄存器的EOB位为1时，接收器接收个数等于SCARD寄存器的BLEN位时置起。此位由硬件设置1，设置ICR寄存器的EOB位清除 0：无中断产生 1：块结束中断产生
LINBK	Bit 6	R	<b>LIN断开检测中断屏蔽标志</b> 当IVS寄存器的LINBK位为1时，LIN模式中，当接收器检测到断开字符时置起。此位由硬件设置1，设置ICR寄存器的LINBK位清除

			<p>0: 无中断产生 1: LIN断开检测中断产生</p>
ADDRM	Bit 5	R	<p><b>地址匹配中断屏蔽标志</b> 当IVS寄存器的ADDRM位为1时, 接收器接收到的字符匹配RS485寄存器的ADDR位时置起。此位由硬件设置1, 设置ICR寄存器的ADDRM位清除 0: 无中断产生 1: 地址匹配中断产生</p>
RXTO	Bit 4	R	<p><b>接收超时中断屏蔽标志</b> 当IVS寄存器的RXTO位为1时, 接收最后一个字节后, 在超时时间内未检测到新的起始位时置起。此位由硬件设置1, 设置ICR寄存器的RXTO位清除 0: 无中断产生 1: 接收超时中断产生</p>
DCTS	Bit 3	R	<p><b>CTS<sub>n</sub>引脚电平改变中断屏蔽标志</b> 当IVS寄存器的DCTS位为1时, CTS<sub>n</sub>引脚电平改变时置起。此位由硬件设置1, 设置ICR寄存器的DCTS位清除 0: 无中断产生 1: CTS<sub>n</sub>引脚电平改变中断产生</p>
ABTO	Bit 2	R	<p><b>自动波特率检测超时中断屏蔽标志</b> 当IVS寄存器的ABTO为1时, 自动波特率在检测到下降沿后, 在超时时间内未检测到上升沿时置起。此位由硬件设置1, 设置ICR寄存器的ABTO位清除 0: 无中断产生 1: 自动波特率检测超时中断产生</p>
ABEND	Bit 1	R	<p><b>自动波特率检测结束中断屏蔽标志</b> 当IVS寄存器的ABEND位为1时, 自动波特率检测完成时置起。此位由硬件设置1, 设置ICR寄存器的ABEND位清除 0: 无中断产生 1: 自动波特率检测结束中断产生</p>
RXBERR	Bit 0	R	<p><b>接收器字节格式错误中断屏蔽标志</b> 当IVS寄存器的RXBERR位为1时, 接收器接收到的字符发生校验错误与帧错误时置起。此位由硬件设置1, 设置ICR寄存器的RXBERR位清除 0: 无中断产生 1: 接收器字节格式错误中断产生</p>

### 30.5.2.17 UART中断清除寄存器 (UART\_ICR)

UART 中断清除寄存器 (UART_ICR)																																						
偏移地址: 40 <sub>H</sub>																																						
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Reserved													TFOVER	Reserved	TFEMPTY	TFTH	TBC	RFUERR	RFOERR	RFFULL	Reserved	RFTH	NOISE	EOB	LINBK	ADDRM	RXTO	DCTS	ABTO	ABEND	RXBERR							

Reserved	Bit 31-19	—	保留
TFOVER	Bit 18	C_W1	发送器FIFO溢出中断清除 0: 写入0无效 1: 发送器FIFO溢出中断清除
Reserved	Bit 17	—	保留
TFEMPTY	Bit 16	C_W1	发送器FIFO空中断清除 0: 写入0无效 1: 发送器FIFO空中断清除
TFTH	Bit 15	C_W1	发送器FIFO触发阈值中断清除 0: 写入0无效 1: 发送器FIFO触发阈值中断清除
TBC	Bit 14	C_W1	发送器字节完成中断清除 0: 写入0无效 1: 发送器字节完成中断清除
RFUERR	Bit 13	C_W1	接收器FIFO下溢中断清除 0: 写入0无效 1: 接收器FIFO下溢中断清除
RFOERR	Bit 12	C_W1	接收器FIFO溢出中断清除 0: 写入0无效 1: 接收器FIFO溢出中断清除
RFFULL	Bit 11	C_W1	接收器FIFO满中断清除 0: 写入0无效 1: 接收器FIFO满中断清除
Reserved	Bit 10	—	保留
RFTH	Bit 9	C_W1	接收器FIFO触发阈值中断清除 0: 写入0无效 1: 接收器FIFO触发阈值中断清除
NOISE	Bit 8	C_W1	噪声位检测中断清除 0: 写入0无效 1: 噪声位检测中断清除
EOB	Bit 7	C_W1	块结束中断清除 0: 写入0无效



			1: 块结束中断清除
LINBK	Bit 6	C_W1	<b>LIN断开检测中断清除</b> 0: 写入0无效 1: LIN断开检测中断清除
ADDRM	Bit 5	C_W1	<b>地址匹配中断清除</b> 0: 写入0无效 1: 地址匹配中断清除
RXTO	Bit 4	C_W1	<b>接收超时中断清除</b> 0: 写入0无效 1: 接收超时中断清除
DCTS	Bit 3	C_W1	<b>CTS<sub>n</sub>引脚电平改变中断清除</b> 0: 写入0无效 1: CTS <sub>n</sub> 引脚电平改变中断清除
ABTO	Bit 2	C_W1	<b>自动波特率检测超时中断清除</b> 0: 写入0无效 1: 自动波特率检测超时中断清除
ABEND	Bit 1	C_W1	<b>自动波特率检测结束中断清除</b> 0: 写入0无效 1: 自动波特率检测结束中断清除
RXBERR	Bit 0	C_W1	<b>接收器字节格式错误中断清除</b> 0: 写入0无效 1: 接收器字节格式错误中断清除

## 第31章 QSPI Flash控制器（QSPIC）

### 31.1 概述

Flash QSPI Controller 通过 AMBA 的 AHB 和 APB 总线连接到 SOC 中。AHB 总线接口主要用于数据的传输，包括在存储器映射的直接模式或者间接模式。间接模式下，系统存储器和外部 Flash 存储器之间数据的传输是通过内部 SRAM 做为中间媒介完成传输，这样可以完成低延时的数据传输，中断或状态寄存器可以用于触发何时去访问内部 SRAM。DMA 接口支持外部 DMA 的访问，在间接模式下可以优化数据的传输，减轻内核负担提高系统的运行效率。

### 31.2 特性

- ◇ 支持存储器映射的直接模式，可以通过 AHB 总线直接从外部串行 Flash 读写数据和运行程序；
- ◇ 支持软件触发的间接模式，完成低延时、无需内核实时参与的数据传输；
- ◇ 间接模式下支持外部 DMA 操作；
- ◇ 支持 1KB 大小的 SRAM，用于减轻 AHB 负担缓存数据在间接模式下；
- ◇ 支持任何设备的时钟频率，包括市面上的 133MHz SDR（Single Date Rate）或 80MHz DDR（Double Data Rate）设备；
- ◇ 支持 XIP（Execute in Place）模式；
- ◇ 支持 DDR（Double Data Rate）模式和 DTR（Double Transfer Rate）协议；
- ◇ 支持 Single、Dual、Quad I/O 命令；
- ◇ 支持可编程的设备大小；
- ◇ 支持可编程的写保护区域，防止系统对外部设备的误写操作；
- ◇ 支持可编程的事务处理之间的间隔；
- ◇ 支持独立的 reference clock 用于去耦系统 AHB Clock 和 SPI Clock；
- ◇ 支持 SPI 串行时钟的极性可编程；
- ◇ 支持可编程的 SPI Clock 波特率产生器；
- ◇ 支持增强型的高速模式的数据捕捉机制；
- ◇ 支持可编程的中断；
- ◇ 支持可编程的 AHB 地址译码，使能每一个外部设备的连续地址模式和自动检测不同设备间的边界；
- ◇ 支持 BOOT 模式；

### 31.3 结构框图

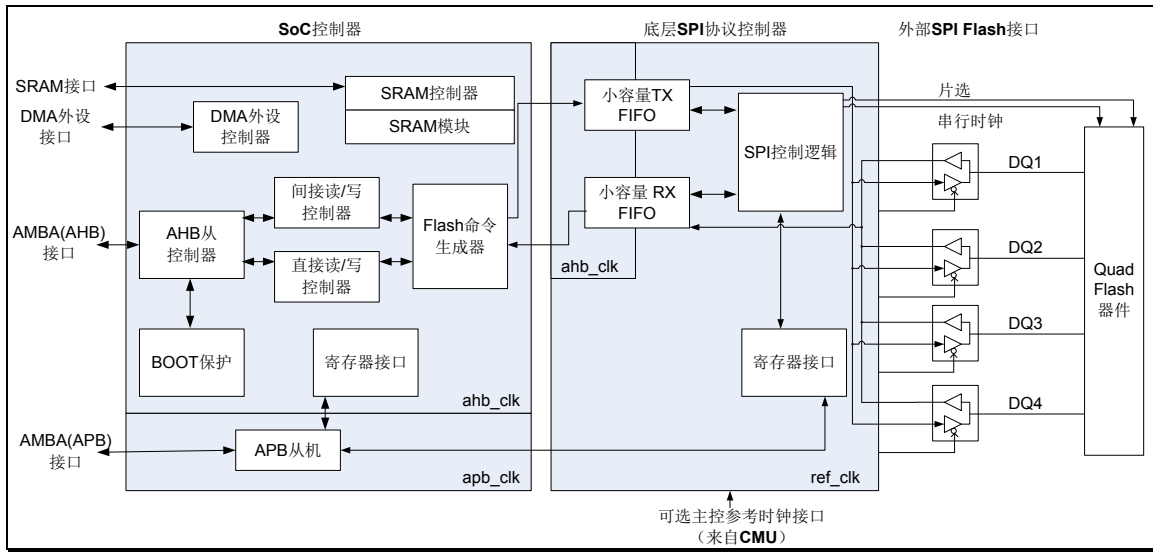


图 31-1 Flash QSPI Controller 结构框图

- ◇ SRAM 大小为 1KB;
- ◇ 支持 DMA Peripheral 接口;
- ◇ AHB 寻址范围: 0x1400\_0000 ~ 0x1BFF\_FFFF
- ◇ APB 寻址范围: 0x4000\_E000 ~ 0x4000\_E3FF
- ◇ 支持单独的 Reference clock, SPI Clock 的 Master Clock 是系统时钟 Reference clock;
- ◇ 支持 1 个外部 Flash 设备;

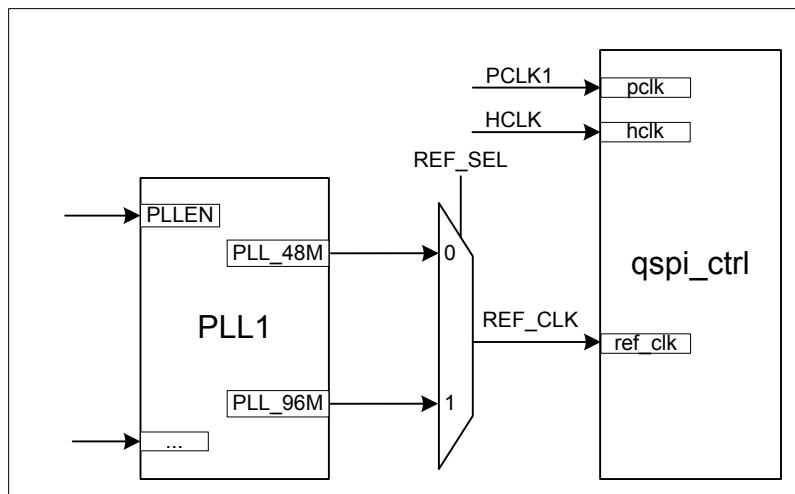


图 31-2 Flash QSPI Controller 时钟结构

- ◇ 当系统时钟选择为 PLL1 (CMU\_CSR[10:8] ->SYS\_STU = 3'b100), 同时 CMU\_CLKENR->PLL1EN\_96M、CMU\_CFGR->QSPIRAEN = 1, REF\_SEL 自动被置高, REF\_CLK 为 96MHz;
- ◇ 当系统时钟选择非 PLL1 (CMU\_CSR[10:8] ->SYS\_STY != 3'b100) 时, 设置

CMU\_CFGR[31]->QSPIRCS = 1, 同时设置 CMU\_CLKENR>PLL1EN\_96M = 1 时,REF\_SEL 自动被置高, REF\_CLK 为 96MHz; 其它设置情况下 REF\_SEL 被置低, REF\_CLK 为系统时钟 sysclk。

## 31.4 功能描述

### 31.4.1 AHB控制接口

AHB 从控制器可验证接收到的 AHB 访问; 对无效请求作出响应; 进行任意字节或者半字的重排序; 屏蔽违反写保护规则的写操作 (仅限直接访问); 向直接访问控制器或者间接访问控制器转发传输请求。

#### 31.4.1.1 AHB接口

AHB 接口遵循 ARM 的 AMBA 3 AHB-Lite 协议规格。不支持传输锁定 (HMASLOCK) 和采用保护控制信号 (HPROT) 的传输。

AHB 的数据位宽为 32 位。因此, 只允许字节, 半字和字访问。关于写操作, 只支持递增迸发 (incrementing bursts), 接收到的 wrapping 写迸发将会产生错误。支持 INCR16, INCR8, INCR4, INCR 和 SINGLE 的迸发类型。关于读操作, 所有迸发类型, 包括 WRAPS, 均支持。如果迸发传输因互连迸发终止而造成提前终止, 或又如果在突发传输内从机发生错误访问, 从机仍将正确运行。

#### 31.4.1.2 AHB地址重映射

QSPI 控制器不会对接收到的错误地址进行具体的地址解码, 但有 AHB 地址解码器。如果使能, QSPI Flash 控制器能够检测到每一个独立器件的地址范围, 基于 AHB 的地址, 做出有效器件自动切换的选择。接收到的 AHB 地址默认直接映射到串行发送至 Flash 器件上的地址。如果 Flash 器件有 24 位地址, 将会发送 AHB 地址的低 24 位。

重映射功能可将接收到的 AHB 地址重映射到地址+N, 其中 N 为重映射寄存器 (0x24) 中的值。重映射功能可通过 QSPI 配置寄存器使能。当软件需要将 BOOT 代码移动至另一个 Flash 区域时, 需要用到重映射功能。

#### 31.4.1.3 写保护

为保护 Flash 器件, 写保护功能可通过硬件实现, 也可通过软件控制。任何检测到的 AHB 写操作, 指向受保护的 Flash 区域, 将通过 HRESP 错误输出产生一个错误信号。

Flash 器件上的区域, 若由多个 Flash “块” 定义而成, 且起始 “块” 有特定编号, 则该区域可受到写保护。提供 3 个可编程寄存器, 0x50~0x58。第一个寄存器定义了需要保护区域底部的 Flash 块。第二个寄存器定义了需要保护区域顶部的 Flash 块。第三个寄存器为一个 2 位控制寄存器。Bit0 可将受保护的区域进行反转, 使得该区域成为 Flash 存储器中唯一不受保护的区域。Bit1 为写保护使能位。当 Bit1 为低电平, Flash 器件不受保护。

AHB 控制器需将接收到的 AHB 地址映射到相关 Flash 块。一个块的大小可通过器件容量配置寄存器设置, 范围为 1~65K 字节。

#### 31.4.1.4 访问转发

为了合法访问，AHB 控制器会将所有的 AHB 访问转发给 2 个访问控制器中的一个：直接访问控制器或间接访问控制器。假设直接访问控制器已通过 QSPI 配置寄存器(0x00)使能，则默认地，所有 AHB 访问将会被转发至直接访问控制器。在转发任何访问至间接访问控制器前，必须由软件先配置。该过程在间接访问控制器章节进行了全面的讲解。如果直接访问寄存器禁能，任何不能转发至间接访问控制器的 AHB 访问将被立即停止，并产生 HRESP 错误。如果直接访问控制器使能，同样的访问将由直接访问控制器转发和服务。

#### 31.4.1.5 顺序访问检测和迸发长度

为了使性能最大化，将不使用 AHB 接口信号 htrans 来识别顺序和非顺序访问。Htrans 的问题在于它将一个新的 AHB 迸发传输启动时的所有访问都识别为非顺序访问，1K 边界以后的所有访问也识别为非顺序访问。取而代之的方法是，通过比较当前地址和上一个地址来检测非顺序访问。如果访问方向(写/读)已改变，则视为非顺序访问。如果读访问的大小(字节/半字/字)已改变，则同样视为非顺序访问。如果 AHB 地址解码器使能并检测到器件之间的切换，则需生成新的 SPI 传输，因为需要初始化当前选中的器件。该情况需视为非顺序访问，尽管地址为连续地址。否则，如果当前传输地址与上一个传输地址为连续地址，则视为顺序访问。

另外，控制器不采用 AHB 的 hburst 信号来识别突发长度。迸发传输将继续，直到收到非顺序访问。

#### 31.4.1.6 AHB地址解码器

控制器可连接最多 4 个 Flash 器件。新兴的存储器已达到 2Gb 的容量并仍在上升。连接多个 Flash 器件时，为确保高性能的连续数据传输，可使能 AHB 地址解码器。AHB 总线发出访问的地址范围被整合成连续的格式。无需利用额外步骤来检测器件之间的边界，就可使软件驱动来管理访问。基于 AHB 接口的实际地址，控制器可自动进行切换。地址边界由硬件进行计算。每个连接器件的大小为 512Mb 到 4Gb。

AHB 地址解码器仅限直接读传输时使用。如果之前有写访问发出，为避免切换后轮询各个器件，要求需读取的数据必须稳定。为确保数据的稳定性，推荐在对每个器件进行第一次读操作时不要使用 AHB 解码器。

### 31.4.2 直接访问控制器 (DAC)

直接访问是指 AHB 访问直接触发对 Flash 存储器进行读或写操作。该操作为存储器映射，可访问并直接执行外部 Flash 存储器中的编码。任何接收到的不在可编程接触发区域内的 AHB 访问都被假想为直接访问，并由直接访问控制器响应。注意，不使用直接访问控制器的访问将不使用嵌入 SRAM。当进行读或写迸发操作时，AHB 将会节流，等待状态的数量取决于控制器的延时。延时已设计的尽可能小。当 XIP 读指令使能时，延时将保持最小值。

当响应 AHB 读操作时，DAC 将会发送一个额外的下游访问。该访问将不会显现于系统接口。该操作作为一个预读操作，确保底层 SPI 核运行于最大带宽。这里定义的 AHB 迸发不同于由 AMBA 定义的 AHB 迸发。

这里的 AHB 定义如下：

1. AHB 迸发的第一次访问由以下定义：
  - a) 非连续 AHB 访问（基于地址比较来判断是否为非连续，而非基于 htrans）
  - b) 非连续或连续 AHB 访问（当下游模块处于空闲状态）
2. AHB 迸发的最后一次访问由连续 AHB 访问定义，优先于上面 1 中定义的新迸发。
3. AHB 迸发的大小为 AHB 访问的数量，从第一次访问到最后一次访问。
4. 每个 AHB 迸发的 DAC 请求的数量 = AHB 迸发大小 + 1

关于 AHB 写操作，直接访问控制器会触发一系列的写命令（类似于读操作的处理方式），DAC 写请求的数量等于接收到的 AHB 写请求的数量。写操作时，AHB 控制器将确保 Flash 迸发不会超出 Flash 页边界。当检测到页边界时，只有到边界为止的字节访问数会被转发。一个跨页边界的连续写请求必须检测为到下游模块的非连续请求，使得下游控制器强制 Flash 器件进入自定时页程序周期。

内核支持跨页分开写，仅限于字对齐的地址。

如果系统发送连续写操作时延迟太久，Flash 写周期可能提前启动，减少器件的有效寿命。注意，如若不能保证，则采用间接写操作来避免这个问题。

Flash 擦操作由软件使用编程接口触发。

一旦页编程周期启动，在允许后续 AHB 访问完成之前，QSPI Flash 控制器将自动轮询写周期直到完成。该操作通过将后续 AHB 直接访问控制在等待状态来实现。

### 31.4.3 间接控制器 (INDAC)

#### 31.4.3.1 间接读控制器

间接操作的目的在于，无需 AHB 访问触发的情况下，从 Flash 存储器读取字节数。间接操作由软件通过特定的 APB 控制/配置寄存器 (0x60 → 0x6C) 控制和触发。该模块与底层 SPI 协议状态机模块通信，展开有效优化的 Flash 读进发操作，将读数据放在本地 SRAM 模块中，为和外部 AHB 主机进行快速及低延时传输做准备。

默认状态下，间接读控制器禁能。在使能前，软件必须配置读取数据的多少和起始地址。0x68 寄存器定义起始地址，0x6C 寄存器定义字节数。一次可同时配置两个间接操作。第二个间接操作可在第一个操作过程中被触发。上一次间接操作完成与下一次间接操作开始之间允许一段短时间的周转。

间接操作读取的字节总数不受 SRAM 大小限制。SRAM 的大小仅会限制发送至 DMA 的请求数量（在 DMA 外设接口模块使能的情况下）。发生 SRAM 上溢时，控制器将压回 Flash 读直到 SRAM 有空余空间。通过完成当前读进发，在 SPI 接口处回压读操作，等待直到 SRAM 有剩余空间，在上一个进发操作终止的地址发送新的读进发操作。

通过发送 AHB 读到 QSPI 控制器，外部主机能够获取控制器从外部 Flash 存储器读取的数据。接收的读访问 AHB 地址必须在 AHB 间接触发地址（通过间接 AHB 触发地址寄存器配置）到 (AHB 间接触发地址 + 2\*\* (间接 AHB 触发地址范围) - 1) 的范围内。范围的默认值等于 16，可有效地处理更小的进发操作。实际的 AHB 地址必须在间接范围内同意 SRAM 为源。每个有效的 AHB 间接读将会引起内部 SRAM 出栈，从而将 AHB 地址从 Flash 地址中分离出来（非直接映射）。因此，AHB 间接触发地址与 Flash 地址毫无任何关系。触发任何有效间接读之后，SRAM 被视为源，而非 Flash 存储器阵列。间接读的 Flash 地址由间接传输起始寄存器 (0x68) 获取。假设请求读取的数据已经在 SRAM 时，QSPI 控制器一旦收到 AHB 访问地址，SRAM 中数据将被取走，这就实现了读进发的最小延时响应。当数据从 SRAM 中读取，QSPI 控制器将释放 SRAM 中的相关资源。如果接收到的 AHB 读访问的地址不在上述范围内，则该访问将不会由间接控制器完成，而是由直接访问控制器完成。

如果接收到的 AHB 读访问在上述范围内，但读取的数据不存在于 SRAM 中，则 AHB 将处于等待状态直到从 Flash 读取到数据并压栈至 SRAM 中。

如果接收到的 AHB 读进发的访问元素横跨 AHB 间接触发范围，则间接触发范围内的访问将由间接控制器处理，其余部分由直接访问控制器处理。注意，这可能是软件配置错误。

外部主机仅允许发送 32-位 AHB 读，直到间接传输结束，这可以减少 SRAM 控制逻辑的复杂程度。在读最后非 32 位对齐数据时，外部主机可发送 16 位（半字）或者字节地址来完成此次传输。同样外部主机可以始终发送 32 位读，末尾非 32 位对齐的数据高位用 0 填充。

理想状态下，在执行读操作时，SRAM 将保持满的状态。SRAM 的数据深度状态 (Fill level) 可直接由软件直接读取 (0x2C)。如果 DMA 外设接口控制器使能，将自动请求外部 DMA 通过 AHB 以有效数据块 (chunks of data) 的形式从 SRAM 获取数据，每个数据块皆是整体间接传输的一部分。

通过设置间接传输控制寄存器的 bit1，可随时取消间接操作。

任何总线主机可发起间接访问。DMA 总线外设接口可分担部分软件开销，有效管理数据传输。另外一个选择是：通过 APB 寄存器，软件直接访问 SRAM 数据深度状态，决定何时从 SRAM 获取数据。当 DMA 外设接口禁能，数据深度阈值寄存器可通过间接传输数据深度阈值寄存器（0x64）提供。当 SRAM 数据深度状态超过该数据深度阈值（watermark），会产生一个中断。如果数据深度阈值大于 0，当数据的最后一个字节被 QSPI 控制器读取并放置于 SRAM 中，即便实际的 SRAM 数据深度状态还未超数据深度阈值，同样会产生数据深度阈值中断。该功能有助于避免软件追踪已被读取的数据量和复位间接读传输中最后几个字节的数据深度阈值。注意，该数据深度阈值寄存器是一个两用寄存器。当 DMA 外设接口使能，由硬件来控制 DMA 请求发送的速率。当 DMA 外设接口禁能，其行为如上文描述。

为了解间接操作的状态，另外还提供两个中断源。第一个：当完成一个间接操作时，会产生一个中断。第二个：如果发出间接读操作请求，但由于 QSPI 控制器中已缓存 2 个间接操作，而未被接受时，会产生中断。

设置间接传输控制寄存器（0x60）中的 bit0 来启动间接读操作，bit2 可用来检查状态。

### 间接读传输流程

◇ 当 DMA 外设控制器使能，需遵循下列流程：

1. 设置 QSPI 配置寄存器（0x00）
2. 通过间接传输数据深度阈值寄存器（0x64）设置 SRAM 数据深度阈值
3. 通过间接传输起始地址寄存器（0x68）设置间接传输的 Flash 起始地址
4. 在间接传输字节数寄存器（0x6C）中设置需传输字节的数量
5. 在间接 AHB 触发地址寄存器（0x1C）中设置间接传输 AHB 触发地址
6. 在间接 AHB 触发地址范围寄存器（0x80）中设置间接传输 AHB 触发地址范围
7. 在 DMA 外设配置寄存器（0x20）中设置
8. 设置间接传输控制寄存器[0] = 1 来触发间接读访问
9. 通过间接传输控制寄存器（0x60）来轮询间接读操作的完成状态。注意，该位为写清零（write-to-clear）。或者通过产生间接完成中断来判断间接读操作是否完成。
10. 在相同的寄存器中读取已完成的间接读操作数量。

◇ 当 DMA 外设控制器禁能，需遵循下列流程：

1. 设置 QSPI 配置寄存器（0x00）
2. 通过间接传输起始地址寄存器（0x68）设置间接传输的 Flash 起始地址
3. 在间接传输字节数寄存器（0x6C）中设置需传输字节的数量
4. 在间接 AHB 触发地址寄存器（0x1C）中设置间接传输 AHB 触发地址
5. 在间接 AHB 触发地址范围寄存器（0x80）中设置间接传输 AHB 触发地址范围
6. 如果使用数据深度阈值中断功能，当数据深度状态超出 SRAM 数据深度阈值寄存器（0x64）设置的阈值，会产生中断。设置数据深度阈值有助于提示软件何时读取间接读传输的下一个部分。注意，如果数据深度阈值设置为一个非 0 值，即便数据深度阈值高于实际的数据深度状态，一旦获取间接传输的最后一个字节并存放于 SRAM 中，就会产生数据深度阈值中断。
7. 设置间接传输控制寄存器[0] = 1 来触发间接读访问
8. 如果使用数据深度阈值中断功能，则等待数据深度阈值中断发生。否则，轮询 SRAM



fill level 来决定什么时候 SRAM 中有足够的数据来触发获取 AHB 数据。

9. 从 SRAM 中读取期望的数据量。如果还要获取更多数据才能完成间接读传输，则回到步骤 8。否则，继续到 10。
10. 通过间接传输控制寄存器 (0x60) 来轮询间接读操作的完成状态。
11. 当间接读操作完成，将产生间接完成中断。

### 31.4.3.2 间接写控制器

间接写操作的目的是以最有效的方式将大量的数据从处理器或 DMA 写到 Flash 存储器。间接传输在 Flash 器件内部执行尽可能少的写周期，因此可最大化器件的寿命。

间接写操作在软件上可视为间接读的相反过程，通过特定的 APB 控制/配置寄存器 (0x70→0x7C) 来控制 and 触发。该模块通过外部 AHB 主机来等待写数据的传输，在与现有的传统 SPI IP 内核进行通信前，写数据会存放在 SRAM 中。

默认状态下，间接写控制器禁能。在使能前，软件必须配置数据的多少和起始地址。0x78 寄存器定义起始地址，0x7C 寄存器定义字节数。一次性可同时编程两个间接操作。第二个间接操作可在第一个操作过程中被触发。上一次间接操作完成与下一次间接操作开始之间允许一段短时间的周转。间接写序列类似于间接读序列。

间接操作中，写的字节总数不受 SRAM 的大小限制。SRAM 的大小仅会限制从外部 AHB 主机接收的数据数量。当外部主机为 DMA，控制器通过 DMA 外设接口要求的数据量不会超出 SRAM 的现有 Fill level。但是，这并不保证 DMA 或者其他外部主机不会试图发送更多的数据，超出 SRAM 所能接收的量。当发生 SRAM 上溢时，控制器会在等待状态下回压 AHB。注意，通过可配置寄存器 (0x2C) 可读取 SRAM 的 Fill level，避免发生上溢。

外部主机会提供写数据，并通过发送 AHB 写操作发送至 QSPI。接收的写访问 AHB 地址必须在 AHB 间接触发地址（通过间接 AHB 触发地址寄存器配置）到（AHB 间接触发地址 + 2\*\*（间接 AHB 触发地址范围）- 1）的范围内。该范围的默认值等于 16，可有效地处理更小的迸发操作。另外，对于压栈连续地址序列也没有严格要求。实际的 AHB 地址必须在间接范围内同意 SRAM 为源。每个有效的 AHB 间接写将会引起内部 SRAM 出栈，从而将 AHB 地址从 Flash 地址中分离出来。因此，AHB 间接触发地址与 Flash 地址毫无任何关系。触发任何有效间接写之后，SRAM 被视为源，而非 Flash 存储器阵列。间接写的 Flash 地址由间接传输起始寄存器 (0x78) 获取。假设在 QSPI 控制器接收 AHB 访问的时候，SRAM 的状态为非满，则会以最小延时向 SRAM 中压栈数据。

如果接收到的 AHB 写访问的地址不在上述范围内，则该访问将不会由间接控制器完成，而是由直接访问控制器完成。

如果接收到的 AHB 写访问在上述范围内，但 SRAM 状态为满，则 AHB 将处于等待状态直到一部分数据或者全部数据从 SRAM 压栈到 SRAM 中。

如果接收到的 AHB 写迸发的访问元素横跨 AHB 间接触发范围，则间接触发范围内的访问将由间接控制器处理，其余部分由直接访问控制器处理。注意，这可能是软件配置错误。

外部主机仅允许发送 32-位 AHB 写，直到间接传输结束，用来减少 SRAM 控制逻辑的复杂程度。在写最后一个字时，外部主机可发送一个 32 位，16 位（半字）或者一个字节来结束此次传输。如果最后一次传输少于 4 个字节，主机依旧可发送 32 位的数据传输。

多余的字节将被丢弃。

当 SRAM 中的字节数量等于或大于一页 Flash 的容量（默认 256 字节）或当 SRAM 中的字节为当前执行的间接传输的所有剩余字节，控制器会向命令发生器发起写迸发操作。

通过设置间接传输控制寄存器的 bit1，可随时取消间接操作。

任何总线主机可发起间接访问。DMA 总线外设接口可分担部分软件开销，有效管理数据传输。另外一个选择是：通过 APB 寄存器，软件直接访问 SRAM 数据深度状态，决定何时向 SRAM 写数据。当 DMA 外设接口禁能，数据深度阈值可通过间接传输数据深度阈值寄存器（0x64）提供。当 SRAM 数据深度状态低于该深度阈值，会产生一个中断。注意，该数据深度阈值寄存器为一个两用寄存器。当 DMA 外设接口使能，由硬件来控制 DMA 请求发送的速率。在该模式下，因为 QSPI 不会发起写操作除非 SRAM 中有至少一页 Flash，或者间接传输的剩余字节在 SRAM 中，DMA 模式下的数据深度阈值需设置为  $\geq 1$  个 Flash 页。

为了解间接操作的状态，另外还提供两个中断源。第一个：当完成一个间接操作时，会产生一个中断。第二个：如果发出间接写操作请求，但由于 QSPI 控制器中已缓存 2 个间接操作，而未被接受时，会产生中断。

设置间接传输控制寄存器（0x70）中的 bit0 来启动间接读操作，bit2 可用来检查状态。

### 间接写传输流程

◇ 当 DMA 外设控制器使能，需遵循下列流程：

1. 通过间接传输起始地址寄存器（0x78）设置间接传输的 Flash 起始地址
2. 在间接传输字节数寄存器（0x7C）中设置需传输字节的数量
3. 在间接 AHB 触发地址寄存器（0x1C）中设置间接传输 AHB 触发地址
4. 在间接 AHB 触发地址范围寄存器（0x80）中设置间接传输 AHB 触发地址范围
5. 在 DMA 外设配置寄存器（0x20）中设置
6. 可选：设置寄存器（0x74）寄存器来控制 DMA 请求的发送速率
7. 设置间接传输控制寄存器[0] = 1 来触发间接写访问
8. QSPI 控制器将利用 DMA 请求接口来要求 DMA 传输数据
9. 通过间接传输控制寄存器（0x70）来轮询间接写操作的完成状态。注意，该位为写清零（write-to-clear）。在相同的寄存器中读取已完成的间接写操作数量
10. 当间接传输写操作完成，会产生间接完成中断。

◇ 当 DMA 外设控制器禁能，需遵循下列流程：

1. 通过间接传输起始地址寄存器（0x78）设置间接传输的 Flash 起始地址
2. 在间接传输字节数寄存器（0x7C）中设置需传输字节的数量
3. 在间接 AHB 触发地址寄存器（0x1C）中设置间接传输 AHB 触发地址
4. 在间接 AHB 触发地址范围寄存器（0x80）中设置间接传输 AHB 触发地址范围
5. 功能上，软件可简单的在一个块内将所有数据写到 SRAM 中。但是，如果写字节数超过 SRAM 的大小，SRAM 极有可能会满，导致 QSPI 花大量时间回压系统 AHB 总线。该时间基于 Flash 的数据速率和器件的页写时间。为避免所有数据在一个块内发送，软件可利用数据深度阈值中断在合适的时间一次发送一页数据。也可选择，软件直接轮询 SRAM 数据深度状态寄存器来识别 SRAM 的空状态，再决定下一部分传输 发送的最合适时间。

6. 如果使用数据深度阈值中断，当数据深度状态低于深度阈值，将产生中断。数据深度阈值的设置范围为 0 ~ 页容量。举例：如果页容量为 256 字节，数据深度阈值的合理设置为 10 到 250 之间。当数据深度状态下降到设定值以下，将触发中断。设置数据深度阈值有助于提示软件何时读取间接读传输的下一个部分。
7. 设置间接传输控制寄存器[0] = 1 来触发间接写访问
8. 如果当前间接传输剩余的字节数大于一个 Flash 页，则向 SRAM 中写入一个 Flash 页数据；否则，将所有剩余数据发送至 SRAM。
9. 如果间接传输中所有的数据都已发送给 SRAM，跳至步骤 11，等待完成状态。如果还有更多的数据需要传输，则：
  - a) 如果使用数据深度阈值中断，则等到中断发生
  - b) 使用 SRAM 数据深度状态来判断发送数据的合适时间
10. 循环回步骤 8
11. 可选：间接写操作的完成状态可通过间接传输控制寄存器（0x70）查询。
12. 当间接写操作完成，会产生间接完成中断。

### 31.4.3.3 间接访问队列

软件允许间接写控制器和间接读控制器有 2 个间接传输排列等候。支持前一个间接操作完成和后一个操作开始前有短暂周转时间。如果等候队列超出 2 个，则会产生中断。

软件上，间接访问队列通过短时间内触发两次间接传输控制寄存器（0x60 或 0x70）的 bit0 来实现。在每一次触发 bit0 前，间接字节数寄存器和间接 Flash 起始地址寄存器必须完成设置。因这些寄存器将进行周期性更新，在间接传输的周期过程中，硬件需要对这些寄存器保持采样的状态。。

内部寄存器块一次只发送一个间接起始触发给关键数据路径块。间接访问控制器中有 2 个独立的数据路径块，可接收和独立采样间接起始触发信息。第一个数据路径块存在于 SRAM 的 AHB 端。当进行间接读时，为读接口；当进行间接写时，为写接口。第二个数据路径在 SRAM 的 Flash 端。当进行间接读时，为写接口；当进行间接写时，为读接口。这两个块可在不同的时间处理间接传输。举例：进行间接读操作时，当第一个传输中的最后一个字节已写入 SRAM，SRAM 中的 Flash 上的数据路径即可开始处理第二个队列传输。开始第二个传输前，必须对字节数和间接 Flash 起始地址寄存器进行重采样。同样地，当第一个间接传输的所有的 Flash 数据都已从 SRAM 发送至 AHB，AHB 上的数据路径块也将重采样相同的寄存器。

### 31.4.3.4 间接传输：连续写读

间接写操作过程中，软件允许触发间接读操作。相似地，间接读操作过程中，也允许触发间接写操作。间接写操作优先级更高。

### 31.4.3.5 访问SRAM

物理上，SRAM 为一个单个端口模块。SRAM 深度可配置。SRAM 被划分为两块区域，低区域部分预留给间接读使用。高区域部分仅为间接写使用。两部分的容量可通过 SRAM 分块寄存器（0x18）配置，用户可选择分配 SRAM 地址总线中多少的 bit 给间接读操作。默认地，该寄存器设置 SRAM 的一半为间接读控制器使用。为确保 AHB 读数据总线不是直接由 SRAM 读数据通过组合逻辑提供，间接读数据路径中包含了另一组寄存器。

以下示例展示了在间接读和间接写之间，SRAM（和另一组寄存器）的分配方式。以下示例中，SRAM 深度为 8 位，等同于 256 地址。

- 如果 SRAM 分块寄存器设置为 0x00，256 个地址分配给间接写，1 个地址分配给间接读
- 如果 SRAM 分块寄存器设置为 0x01，255 个地址分配给间接写，2 个地址分配给间接读
- 如果 SRAM 分块寄存器设置为 0x02，254 个地址分配给间接写，3 个地址分配给间接读
- 依次类推，直到.....
- 如果 SRAM 分块寄存器设置为 0xfd，3 个地址分配给间接写，254 个地址分配给间接读
- 如果 SRAM 分块寄存器设置为 0xfe，2 个地址分配给间接写，255 个地址分配给间接读
- 如果 SRAM 分块寄存器设置为 0xff，1 个地址分配给间接写，256 个地址分配给间接读

注：避免将 SRAM 分块寄存器设置为 0xFF 或者 0x00，因为仅 SRAM fill level 底部的 8 位才能通过软件访问。如果间接读或间接写的 fill level 达到 256，当读取 fill level 时，会显示为 0。

有 4 个 SRAM 源，它们都在单个 SRAM 端口上进行仲裁和复用。高达 3 个源可在任意时间访问该端口。

- 间接写，写源，位于 SRAM 的 AHB 上。
- 间接写，读源，位于 SRAM 的 Flash 上。
- 间接读，写源，位于 SRAM 的 Flash 上。
- 间接读，读源，位于 SRAM 的 AHB 上。

优先级的仲裁方案如下：

		SRAM 访问优先级
间接写	写入 SRAM (从系统 AHB)	第 3 优先级 (不包括 AHB 读请求)
	读取 SRAM (从 QSPI 控制器)	第 2 优先级
间接读	写入 SRAM (从 QSPI 控制器)	第 1 优先级
	读取 SRAM (从系统 AHB)	第 3 优先级 (不包括 AHB 写请求)

表 31-1 SRAM 访问优先级

在读操作过程中（SRAM 中的 Flash）除写端口外，驱动 4 个源的逻辑不能被认为是单个周期完成。为避免数据丢失，在间接读过程中，必须允许写 SRAM 可以立即完成。因此，该端口拥有最大优先级。

### 31.4.4 DMA外设控制器

外设接口用来触发外部 DMA 进行短延时 AHB 数据迸发访问。DMA 外设接口指的是 ARM 的 DMA 控制器 DMA-230。DMA 外设接口仅用于间接操作模式，数据缓存于 SRAM 可更快速的响应 AHB 请求，并使内核在一个较长的周期内进行底层 Flash 传输。支持 2 个 DMA 请求，一个用于间接读控制器，另一个用于间接写控制器。关于间接读控制器，当数据已从 Flash 获取并写入自有 SRAM 后，QSPI 控制器仅发送 DMA 请求。关于间接写控制器，当触发传输后，QSPI 控制器会立即发送 DMA 请求并一直持续发送直到整个间接写传输完成。Watermark 寄存器可改变发送请求的速率。

#### 31.4.4.1 操作顺序

当间接操作被触发，DMA 外设控制器可见所有 Flash 存储器的传输数据(以字节为单位)。控制器会将这些数据分为 DMA 迸发请求和单次请求：所有字节数除以迸发请求中设置的字节数，余下的除以单次请求中的字节数。软件需确保在这些除法之后无余数。举例，如果从 Flash 要读取的数据总字节为 512，SRAM 固定容量为 256 字节，软件配置的迸发传输字节数为 256，则当前 256 个字节已缓冲，SPI 控制器将会触发 DMA 迸发请求。当又有 256 个字节在 SRAM 中缓冲完成后才会触发第二次迸发请求。由于 SRAM 本身容量为 256 字节，意味着在发送下一个请求前，DMA 会从 SRAM 中获取所有内容。

SRAM 数据深度状态对 DMA 外设控制器可设计为可见。关于间接读，会基于以下状态机发送请求给外部 DMA。

1. 等待 START 触发
2. 如果下一个需发送 BURST 请求，等待直到下面两个条件均为 TRUE。
  - a) SRAM 数据深度状态大于或等于 burst\_size 寄存器中的值
  - b) SRAM 数据深度状态大于或等于 watermark 寄存器中的值当 TRUE 时，执行以下：
  - 发送 BURST
  - 计算数据深度状态期望值 = SRAM fill level – num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 BURST 请求，跳至 4
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1
3. 如果下一个需发送的为 SINGLE 请求，等待直到下面条件均为 TRUE。
  - a) SRAM 数据深度状态大于或等于 single\_size 寄存器中的值当 TRUE 时，执行以下：
  - 发送 SINGLE
  - 计算数据深度状态期望值 = SRAM fill level – num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1
4. 下一个需发送的为 BURST 请求。如果 (SRAM fill level – num\_bytes\_requested\_to\_DMA) 小于 burst\_size 寄存器中的值，跳至 2。其他，执行以下：

- 发送 BURST
  - 计算数据深度状态期望值 = SRAM fill level – num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 BURST 请求，跳至 4
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1
5. 下一个需发送的为 SINGLE 请求。如果 (SRAM fill level – num\_bytes\_requested\_to\_DMA) 小于 single\_size 寄存器中的值，跳至 3。其他，执行以下：
- 发送 SINGLE
  - 计算数据深度状态期望值 = SRAM fill level – num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1

关于间接写，会基于以下状态机发送请求给外部 DMA。

1. 等待 START 触发
2. 如果下一个需发送的为 BURST 请求，等待直到下面两个条件均为 TRUE。
  - a) SRAM 的剩余空间大于或等于 burst\_size 寄存器中的值
  - b) SRAM 数据深度状态小于或等于 watermark 寄存器中的值当 TRUE 时，执行以下：
  - 发送 BURST
  - 计算数据深度状态期望值 = SRAM fill level + num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 BURST 请求，跳至 4
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1
3. 如果下一个需发送的为 SINGLE 请求，等待直到下面条件均为 TRUE。
  - a) SRAM 的剩余空间大于或等于 single\_size 寄存器中的值当 TRUE 时，执行以下：
  - 发送 SINGLE
  - 计算数据深度状态期望值 = SRAM fill level + num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1
4. 下一个需发送的为 BURST 请求。如果 (SRAM 剩余空间 – num\_bytes\_requested\_to\_DMA) 小于 burst\_size 寄存器中的值，跳至 2。其他，执行以下：
  - 发送 BURST
  - 计算数据深度状态期望值 = SRAM fill level + num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 BURST 请求，跳至 4
  - 如果下一个需发送的为 SINGLE 请求，跳至 5

- 其他，跳至 1
5. 下一个需发送的为 SINGLE 请求。如果 (SRAM 剩余空间 - num\_bytes\_requested\_to\_DMA) 小于 single\_size 寄存器中的值，跳至 3。其他，执行以下：
- 发送 SINGLE
  - 计算 fill level 期望值 = SRAM fill level + num\_bytes\_requested\_to\_DMA
  - 如果下一个需发送的为 SINGLE 请求，跳至 5
  - 其他，跳至 1

关于间接读操作，SRAM 数据深度阈值寄存器 (0x64) 可定义控制器发送的第一个 DMA 请求的最小数据深度状态，设置的值越大，在 DMA 获取之前，缓存在 SRAM 中的数据就越多。关于间接写操作，SRAM 数据深度阈值寄存器 (0x64) 可定义控制器发送的第一个 DMA 进发/单次请求的最大数据深度状态。数据深度阈值寄存器可使系统在短周期内选择性的集中于 AHB 传输。默认地，数据深度阈值寄存器被复位为 0，意味着外设控制器可尽快发送 DMA 请求。在间接读的情况下，意味着 SRAM 中有足够的数据进行进发请求或单次请求 (如果剩余字节小于进发请求的设置值)。注意，如果 SRAM 为空，DMA 外设不可发送间接读 DMA 请求；如果 SRAM 为满，DMA 外设不可发送间接写请求。

DMA 外设接口可通过软件禁能。当接口禁能，将不会发送任何通道请求。注意，如果除了 DMA 以外的 AHB 主机将进行间接数据传输，DMA 外设接口必须被禁能。

#### 31.4.5 软件触发指令生成器 (STIG)

直接和间接访问控制器用来进行数据传输。为进行擦除和访问易失性和非易失性配置寄存器，传统 SPI 状态寄存器，其他的状态/保护寄存器，需要有一个独立的软件控制器。软件触发指令生成器 STIG 由 Flash 命令控制寄存器 (0x90) 通过设置命令发送去 Flash 器件来控制。这是一个通用寄存器，可用来执行 Flash 器件支持的扩展 SPI 协议的任何指令。指令如果不符合 Flash 器件规格可导致未知的控制器行为。读时，当命令已响应 (bit1 从 1 翻转为 0)，最多 8 个数据可放置于 Flash 命令读数据寄存器 (0xA0 和 0xA4)。写时，写数据应该放置在 Flash 命名写数据寄存器 (0xA8 和 0xAC)。

实现以上功能的代码保存在 QSPI 的 APB 寄存器块内。

##### 31.4.5.1 响应 STIG 请求

STIG 请求会致使 QSPI Flash 控制器询问 Flash 命令控制寄存器 (0x90)，决定该以什么方式发送多少字节去 Flash 器件。寄存器中的 Bits[31:24] 指示要发送的指令，始终首先压栈。地址发送紧接在指令后面，地址大小同样在此寄存器中配置。地址本身保存在 Flash 命令地址寄存器 (0x94)。Dummy 字节此后发送 (字节数量仍在 0x90 中配置)。写/读数据的地字节数同样在 0x90 中配置。写时，最多可发送 8 个字节 (保存于 Flash 命令写数据寄存器 0xA8 和 0xAC)。读时，当读数据已从 Flash 器件获取，QSPI Flash 控制器会将读数据保存在 Flash 命令读数据寄存器 (0xA0 和 0xA4)。

当 QSPI Flash 控制器开始响应 STIG 请求，Flash 命令控制寄存器的 bit1 置 1，表示正在执行命令中。

当 QSPI Flash 控制器处于自动轮询状态，响应 STIG 请求会有一些不同。在一个程序操

作过后，大部分器件都不可访问直到写操作完成。它们中的一些器件还有可能会停止编程页。轮询 Flash 状态寄存器的 bit8 指示了有效自动轮询状态。当发生 STIG 请求后，QSPI Flash 控制器会立即发送合适的 OPCODE 去存储器。在响应 STIG（自动轮询中）过重，命令执行的状态位保持，其他位（比如 ADDRESS 或 DUMMY 位）都被禁止。

另外还有一个可配置选项：在两个重复的轮询操作之间加入延时（延时由写完成控制寄存器得 MSB 位定义）。该功能的作用是用来释放 SPI 带宽。

#### 31.4.6 直接/间接访问控制器和STIG间的仲裁

当多个寄存器同时有效，一个简单的固定优先级仲裁方案可用来对每个接口进行仲裁，访问外部 Flash。固定优先级定义如下，1 为最高优先级。

1. 间接访问写
2. 直接访问写
3. STIG
4. 直接访问读
5. 间接访问读

等待响应时，每个控制器都会被回压。

#### 31.4.7 SPI命令转换

由直接访问控制器，间接访问控制器或 STIG 发送的请求会转换成字节传输序列，发送去下游。下例显示的是一个字节无序列读：

- INSTRUCTION OPCODE → ADDRESS → Mode Byte → Dummy Bytes → 1 byte of don't care

当进行序列访问时，每一次读，都会在上面的序列最后向 Flash 器件压栈一个额外的字节。目的是为了每个传输字节之间没有空隙。

实际的发送去 Flash 器件的序列由要求的传输决定，是否该传输是非序列的还是序列的，是否器件已在 XIP 模式下配置完成，以及主要器件指令类型寄存器（地址偏移 0x4 和 0x8）的状态。

写时，在发送写序列前，Flash 器件内的写使能锁存（WEL）必须为高电平。在通过直接或间接访问控制器（DAC/INDAC）触发写命令前，QSPI Flash 控制器将自动发送写使能锁存命令。为提高灵活性和性能，用户可设置器件写指令配置寄存器中的 bit8 来关闭该功能。WREN 的 OPCODE 为 0x06，在器件之间共用。

当不再接收来自直接或间接访问控制器的写请求，并且所有请求都已发出，则 Flash 器件将自动开始页编程写周期。此时接收到的请求将会保持在等待状态，直到写周期完成。QSPI Flash 控制器会自动轮询 Flash 器件的传统 SPI 状态寄存器，来判断写周期是否完成：发送 RDSR OPCODE 去 Flash 器件，一直等待直到器件本身显示写周期已完成（直到 Write In Progress bit[0]已清零，写使能锁存位也已清零）。WREN 和 RDSR 器件指令为仅有的由两个由控制器在底层发送的指令。任何其他的具体指令都应发送去器件，通过 STIG 发送 Flash 命令来单独处理。



### 31.4.8 Flash指令类型选择

为了发送正确的读写 OPCODE，软件应初始化器件读指令类型寄存器（地址偏移 0x4）和器件写指令类型寄存器（地址偏移 0x8）。这两个寄存器中，可设置指令 OPCODE，指令类型，边沿模式（DRR 或 SDR），和选择单个，双个还是 4 个引脚来进行数据传输。为确保控制器可在复位状态中运行，寄存器会复位到兼容 SIO 器件的 OPCODE。

器件读指令寄存器中包含指令类型字段，器件写指令寄存器无指令类型字段。如果软件将指令类型设置为非 0 值，器件读指令类型寄存器和器件写指令类型寄存器中的地址传输类型和数据传输类型位无关。软件支持不太常见的 Flash 指令，在 2 或 4 通道发送 OPCODE，地址和数据。注意，对于那些支持 2 或 4 通道发送 OPCODE 的指令的器件，这些指令的名字与 Flash 数据手册上的命名不一致。其中一个支持这些指令的器件为 Numonyx（Micron）N25Q128。另外的读指令被称为 DCFR 和 QCFR。写指令为 DCPP 和 QCPP。下表以 N25Q128 作为参考，展示了软件该如何配置 QSPI。

◇ 读

OPCODE	OPCODE 发送通道数	ADDRESS/DUMMY MODE 发送通道数	DATA Bytes 发送字节数	QSPI 指令类型（器件读指令配置寄存器）	QSPI 地址 Xfer 类型（器件读指令配置寄存器）	QSPI 数据 Xfer 类型（器件读指令配置寄存器）
READ	1	1	1	0	0	0
FAST_READ	1	1	1	0	0	0
DOFR (Dual O/P Fast Read)	1	1	2	0	0	1
DIOFR (Dual O/P Fast Read)	1	2	2	0	1	1
QOFR (Quad O/P Fast Read)	1	1	4	0	0	2
QIOFR (Quad O/P Fast Read)	1	4	4	0	2	2
DCFR (Dual Command Fast Read)	2	2	2	1	Don't Care	Don't Care
QCFR (Quad Command Fast Read)	4	4	4	2	Don't Care	Don't Care

◇ 写

OPCODE	OPCODE 发送通道 数	ADDRESS/ DUMMY MODE 发送 通道数	DATA Bytes 发送字 节数	QSPI指 令类型 (器件 读指令 配置寄 存器)	QSPI地 址 Xfer 类型(器 件读指 令配置 寄存器)	QSPI数 据 Xfer 类型(器 件读指 令配置 寄存器)
PP	1	1	1	0	0	0
DIFP (Dual Input Fast Program)	1	1	2	0	0	1
DIEFP (Dual Input Extended Fast Program)	1	2	2	0	1	1
QIFP (Quad Input Fast Program)	1	1	4	0	0	2
QIEFP (Quad Input Extended Fast Program)	1	4	4	0	2	2
DCPP (Dual Command Fast Program)	2	2	2	1	Don't Care	Don't Care
QCPP (Quad Command Fast Program)	4	4	4	2	Don't Care	Don't Care

有几组器件（例如：Numonyx（Micro）N25Q512A）能够在双数据速率（DDR，也可叫做双传输速率模式（DTR））模式下处理读操作。在专有命令类型工作过程中，这些器件均可在上升沿和下降沿发送数据。这使得控制器可在  $1/2 \text{ spi\_clk}$  的情况下保持吞吐量。器件读指令类型寄存器中的 DDR 使能位可告知 QSPI Flash 控制器，写入读 OPCODE 位段的 OPCODE 可处理 DDR 命令类型。读数据捕捉寄存器能在 DDR 模式下移动传输数据。默认地，数据移位 1 个时钟周期可确保在 DDR 传输中保持时间大于 0。对于高  $\text{ref\_clk}$  频率来说，移位 1 个时钟可能不够。DDR 命令的配置与读表中列出的 SDR 命令兼容。

Micron 器件中的 MT25 系列还加入了 DTR 协议，可处理 DTR 模式中的所有命令。根据专门的 OPCODE，DTR 读命令可检测 DTR 模式。因此，OPCODE 必须作为 STR 发送。当 DTR 协议使能，器件无需 OPCODE 来检测边沿模式，因为可从配置寄存器中的易失性和非易失性位来识别。

### 31.4.9 APB接口与寄存器模块

APB 接口利用 Flash 命令控制寄存器来进行由软件控制的 Flash 访问。APB 接口提供单个寄存器块，包含了可配置的寄存器组。该寄存器组由 APB 时钟定时。

### 31.4.10 捕捉读数据

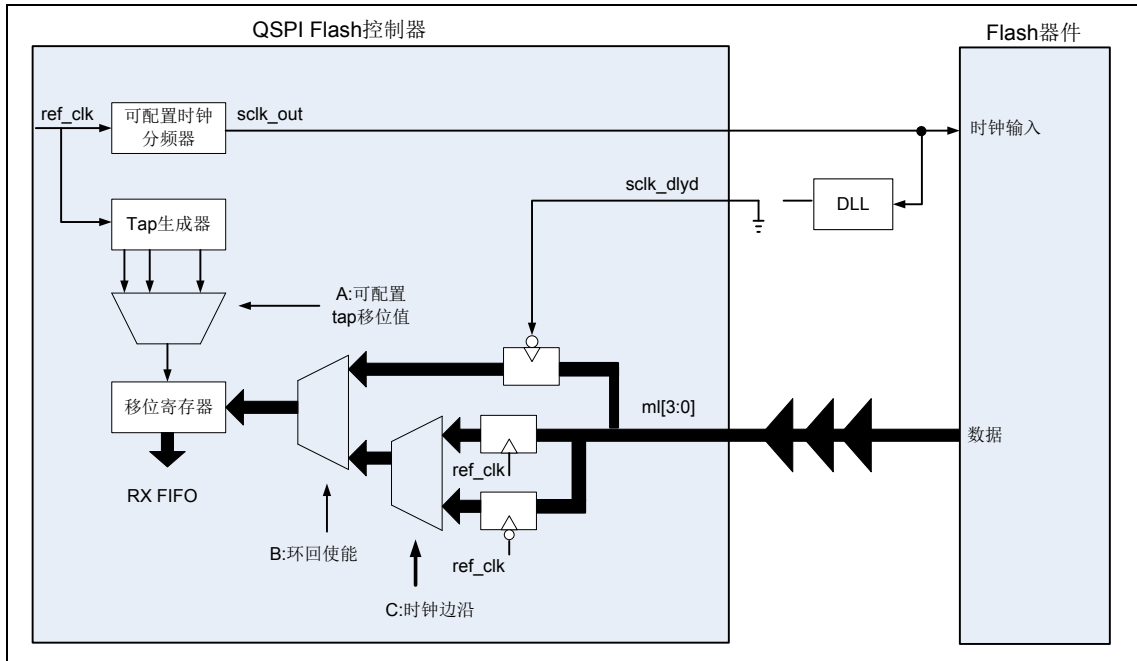


图 31-2 数据采样框图

- 注 1: 如上图所示设计中不支持时钟 loopback 的功能, 所以在产品应用时 loopback enable 信号应该一直保持为 1, 即 QSPI\_RDCR->BYLPC 为常 1;
- 注 2: A 点 configurable tap shift value 由寄存器控制位 QSPI\_RDCR->DLYR 控制;
- 注 3: C 点 clock edge 由寄存器控制位 QSPI\_RDCR->SMES 控制;

#### 31.4.10.1 Tap机制

有三个功能允许软件对读数据捕捉进行调节。POR 后, 更新后的回环时钟电路和 `ref_clk` 延时寄存器线在禁能状态下唤醒。这在大部分的应用下都有效, 并可用来做器件计算。读数据捕捉寄存器 (0x10) 可控制这些功能。

Bit5 选择 `ref_clk` 的边沿, 在选中的边沿上对数据输出进行采样。另外, 提高 2 倍的采样精度仅限于高性能的读操作。

Bits4:1 控制读数据捕捉周期的额外数量 (`fast ref_clk`, 至少是器件时钟的 4 倍), 应用于内部读数据捕捉电路。Flash 存储器的 clock-to-out 延时和跟踪延时, 以及其他器件延时, 可强加一个最大的 flash 时钟频率 (小于 flash 存储器本身的运行频率)。该寄存器的设置值需能确保稳健的数据捕捉。

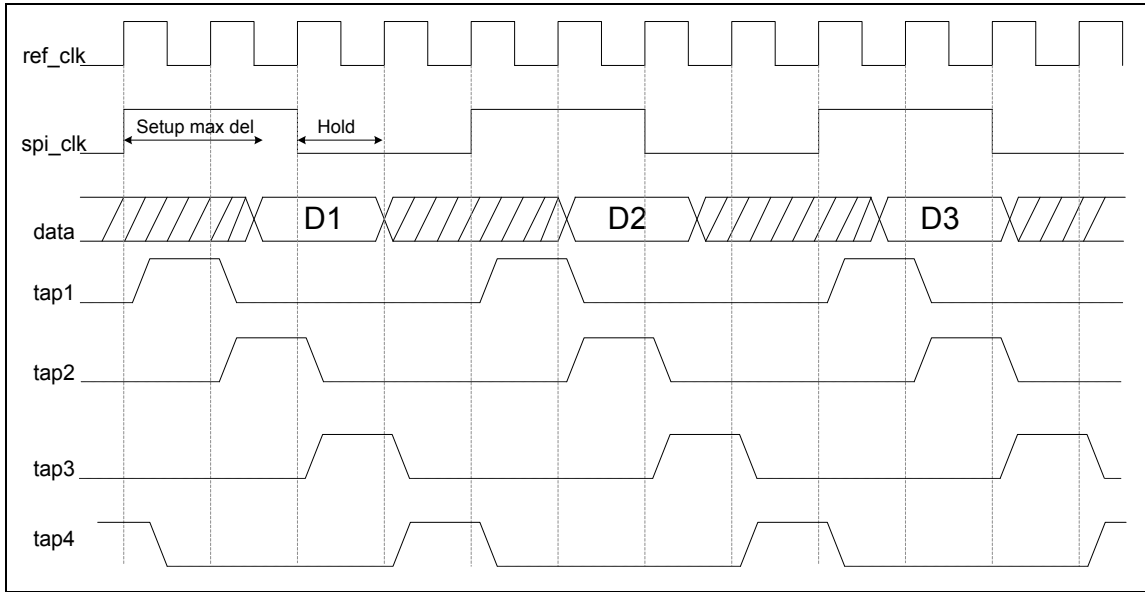


图 31-3 过采样时序图

### 31. 4. 11 BOOT特性描述

默认 BOOT 功能进行以下假定：

- 波特率分频 = /32 (sclk = ref\_clk/32)
- 仅 CS[0]有效 (默认 BOOT 功能仅能从单个 Flash 器件中获取数据)
- 简易 BOOT 仅在扩展 SPI 协议下工作 (Single 模式)
- 简易 BOOT 功能采用简易读数据命令 (opcode: 0x03)

### 31. 4. 12 示例：8 字节读传输

为了更好的理解 QSPI 接口的高层行为，下图展示了在读传输过程中的初始阶段。注意，该示例仅为读指令中的一种。

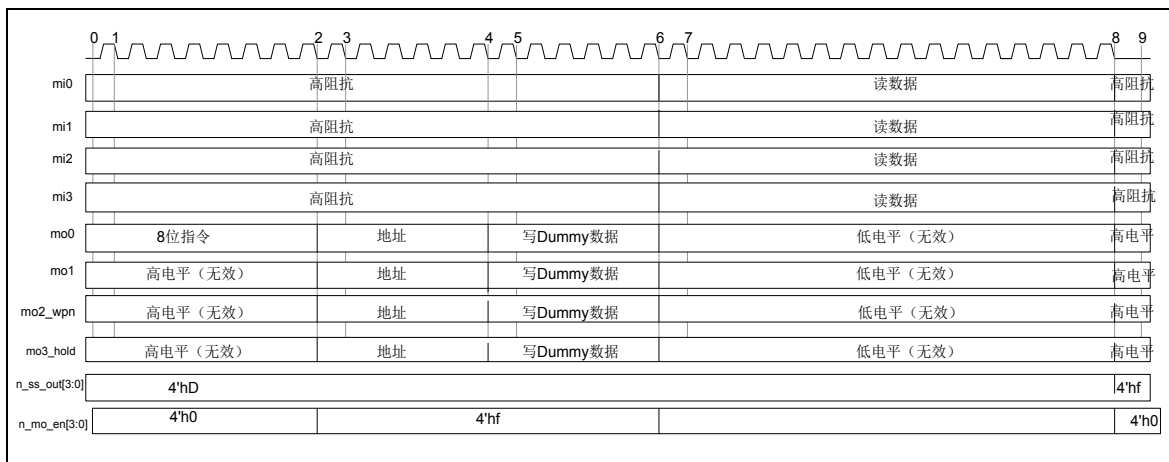


图 31-4 QSPI 读传输时序

- 0: 传输开始, 片选 (`n_ss_out`) 激活 `slave1` (`4'hf` 转变为 `4'hd`)。
- 1: 指令阶段起始。首 8 位仅从 `mo0` 上输出。其他 `mo` 引脚在该阶段未使用。
- 2: 指令阶段结束。输出最后 8 个指令位。
- 3: 地址阶段起始。该例使用 3 个字节的地址在 4 个 `mo` 引脚上输出。半个字节输出需要一个时钟周期, 所有三个字节需花六个时钟周期。
- 4: 地址阶段结束。最后的半个字节地址从 4 个 `mo` 引脚上输出。
- 5: 读传输要求 3 个字节的 `dummy` 写数据。就是在此边沿处, 前 6 个半字节在 4 个 `mo` 引脚上输出。
- 6: `dummy` 写数据的最后半字节。这是读传输的最后一个写阶段, 因此输出使能 (`n_mo_en`) 从 `4'h0` 转变为 `4'hf`。
- 7: 输入 `mi` 引脚上的首个读数据周期。该示例中, 在 4 个 `mi` 引脚上读取 8 个字节用来 16 个周期。当读数据被接收时, `mo` 信号无效, 为低电平。
- 8: 读数据输入的最后周期。`mi` 输入引脚恢复为高阻抗输入。`Mo` 输出引脚转换为高电平。芯片使能变为无效。传输完成。
- 9: 输出使能恢复为 `4'h0`。

## 31.5 编程指引

在与 Flash 器件通信前，软件负责配置 QSPI 控制器。

在 QSPI 控制器通过 QSPI 配置寄存器的 bit0 使能前，控制器静态配置位需设置完成。该目的是：在未实施亚稳定保护的路径上避免时钟边界问题。如果用户希望更改控制器配置，在重新配置前，建议将 QSPI 的使能为设置为禁能。

### 31.5.1 复位后配置QSPI控制器

QSPI 控制器可在合适的状态唤醒后，使用直接访问控制器进行基本读写。所有目标器件均支持基本读（opcode 0x03）和基本写（opcode 0x02）操作。控制器唤醒时，波特率分频为 32。假设复位后，参考时钟为 400MHz，意味着有效 SPI 时钟仅 12.5MHz。该时钟必须足够慢才能满足所有目标器件的时序要求。

如果目标器件不使用 3 个地址字节，器件容量配置寄存器必须更改为一个合适的值。所有在该项目中测试的器件，其页容量为 256 字节，无需再更新寄存器。

如果软件要进行写入器件操作，而每个器件页的字节数不等于 256 字节，则该器件容量配置寄存器也必须更改。

虽不是硬性要求，但建议在使能 QSPI 控制前，软件先使能写保护功能。需设置保护寄存器（0x50, 0x54 和 0x58），器件容量配置寄存器中的每个器件块的字节数也需设置。

初始化后，当控制器和 DAC 使能后，软件可读取和写入 Flash 器件。简单对 QSPI 配置寄存器（地址偏移 0x00）就可使能控制器。注意，不要更改该寄存器中波特率分频和 CPOL/CPHA 的默认值。建议写入值为 0x00780081。

### 31.5.2 QSPI控制器配置优化

为更优化地访问 Flash，软件需准确配置控制器。

1. 等待直到挂起的 STIG 或 INDAC 操作都已完成或轮询 QSPI 配置寄存器的 bit31
2. 禁能 DAC（QSPI 配置寄存器 bit7）。不必要完全禁能 QSPI 控制器（QSPI 配置寄存器 bit0）
3. 根据用户希望使用的指令类型来更新器件指令配置寄存器
4. 如果器件指令配置寄存器中的模式位使能，更新模式位配置寄存器
5. 如果内容不正确，更新器件容量配置寄存器。注意，初始化后，部分或整个寄存器都有可能已更新。进行读写时，地址字节数是一个关键配置。进行任何写操作时，需要设置每页的字节数。只有再写保护功能使用的情况下，才需要每个器件块的字节数。如果目标器件的默认值正确，或者如果某些值（不包括地址字节数）不正确，但不允许器件写操作。
6. 更新 QSPI 器件延时寄存器。该寄存器允许用户在每次 Flash 访问后，对如何驱动片选进行调整。原因是每个器件可能有不同的时序要求。当串行时钟频率增加，时序要求将变得越来越重要。注意，该寄存器中设置的值基于 ref\_clk 周期。举例：在 CS 无效后重新设置为有效前，ATMEL 器件需要至少 50ns 的时间。默认情况下，控制器只会提供最小 1 SCLK 周期。当器件运行于 100MHz，SCLK 周期仅为 10ns，

因此需要额外的 40ns。当 `ref_clk` 运行于 400MHz (2.5ns 周期)，该寄存器中的 `d_nss` 位至少要设置为 16。在自动轮询阶段，可增加该延时。写完成寄存器中可定义轮询重复延时。

7. 如有需要，重新映射地址寄存器。仅影响 DAC 路径。
8. 如有需要，如果在过往的初始化中还未设置，则设置和使能写保护寄存器。
9. 通过中断屏蔽寄存器使能需要的中断。
10. 在 QSPI 配置寄存器中设置波特率分频，定义目标器件所需的时钟频率。
11. 更新读数据捕捉寄存器。当捕捉读数据时，该寄存器可延时。当由器件到控制器的读数据路径很长且器件时钟频率很多，该寄存器的设置也会有所帮助。
12. 通过 QSPI 配置寄存器使能 QSPI 控制器和 DAC。

### 31.5.3 Flash命令控制寄存器的使用 (STIG操作)

Flash 命令控制寄存器提供了一种灵活可编程的方式访问 Flash 器件，被称为 STIG 操作。指令 `OPCODE`，地址字节数，地址本身，`dummy` 字节数，写数据字节数，写数据本书和读数据字节数，均可设置。一旦设置完成，软件可通过 `bit0` 触发命令，轮询 `bit1` 等待被接受。当该位变为无效，就可触发另一个 STIG。软件可用此种访问 Flash 的典型方法来访问 Flash 器件寄存器，也可进行擦除才做。尽管最多 8 个数据字节（由 Flash 命令写和读数据寄存器定义，`0xA0`, `0xA4`, `0xA8`, `0xAC`）可一次读或写，该方法同样可用来访问 Flash 阵列本身。

对比 AHB 的所有其他读访问，该接口发送的命令拥有更高优先级。

### 31.5.4 SPI传统模式

SPI 传统模式允许软件直接访问内部 TX-FIFO 和 RX-FIFO，因此可绕过直接，间接和 STIG 控制器。

传统模式允许用户发送任意 Flash 指令，但不会花费太多的软件开销，确保了 FIFO fill level 的有效管理。传统 SPI 内核是双向传输，当片选使能，数据可在任一方向连续传输。即使驱动器只希望从 Flash 器件读取数据，`dummy` 数据必须写出确保片选保持有效。对于写传输，反之亦然。这意味着，向一个器件（3 个地址字节）读 4 个字节，软件需向 TX FIFO 中写入一共 8 个字节。第一个字节为指令 `OPCODE`，接下来的 3 个字节为地址，最后 4 个字节为 `dummy` 数据。类似的，因 TX-FIFO 中写入了 8 个字节，软件会期望有 8 个字节返回至 RX-FIFO 中。前 4 个字节会被丢弃，后 4 个字节为所需的读数据。

由于 TX-FIFO 和 RX-FIFO 深度有限，软件有责任保持 FIFO 的深度，确保在指令执行过程中 TX-FIFO 不会下溢，RX-FIFO 不会上溢。这可能会增加需要软件开销。当数据深度超过深度阈值时，将会产生中断。

当传统模式使能，软件可访问 TX-FIFO，通过 AHB 接口向 QSPI 控制器的任意地址内写入任意值。当传统模式使能，软件可通过 AHB 接口向 QSPI 控制器读取任意地址。

## 31.5.5 进入和退出XIP模式

### 31.5.5.1 从POR进入XIP模式

如果 XIP 以非易失性配置的使能，则 XIP 模式可以非易失性方式进入。Flash 器件中仅一小部分支持该功能。由于 Flash 器件唯一能识别的操作为 XIP 读操作，软件无法通过 Flash 状态寄存器发现 POR 后 XIP 的状态。

如果已得知器件会从 POR 进入 XIP，模式位配置寄存器和 QSPI 配置寄存器中的 bit18 应在初始 BOOT 中设置完成。

如果不知道器件会从 POR 进入 XIP，且连接的 Flash 器件支持从 POR 进入 XIP，则软件通过 STIG 命令发送 XIP 退出命令来尝试退出 XIP 模式。软件需了解器件的模式位要求，因为每个器件的 XIP 进入和退出都不同。Micron N25Q 和 MT25 支持 XIP 从 POR 进入/退出。退出 XIP 模式时，需设置 OPCODE 为 8'h00，地址字节数为 3，dummy 周期数为 16，读字节数为 1。地址需配置为{8 模式位，16'h0000}。Micron 器件模式位需设置为 8'b10000000。

注意，通过 Flash 命令控制寄存器发送 NVCR 命令可使能从 POR 进入 XIP。直到下一个 POR 序列时，才会生效。

### 31.5.5.2 其他进入XIP模式的情况

大部分 Flash 器件支持 XIP 模式。但，Flash 生厂商并没有一致的标准。大部分是在地址字节后直接发送 signature bits。其他，像 Micron 器件，采用 signature bits 的同时，也要求协 Flash 器件配置寄存器来使能 XIP。对于必须服从该控制器的 Flash 器件，需遵循以下步骤进入 XIP 模式。

#### Micron N25Q 和 MT25（不支持基本 XIP）

通过设置 Flash 器件内的 VCR[3]来使能 XIP 模式。利用 Flash 命令控制寄存器写 VCR 来发送 VCR 写命令，步骤如下：

- 禁能直接访问控制器和简介访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 配置 Flash 命令控制寄存器，发送 VCR 写至 Flash 存储器。
- 设置模式位配置寄存器（0x28）中的 XIP 模式位为 8'b00000000。
- 通过设置 QSPI 配置寄存器中的 bit17 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，简介访问控制器。

#### 其他 Micron 器件（支持基本 XIP）

- 禁能直接访问控制器和简介访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位配置寄存器（0x28）中的 XIP 模式位为 8'b10000000。
- 通过设置 QSPI 配置寄存器中的 bit17 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，简介访问控制器。

#### Winbond 器件

- 禁能直接访问控制器和简介访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位配置寄存器（0x28）中的 XIP 模式位为 8'b00100000。



- 通过设置 QSPI 配置寄存器中的 bit17 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，简介访问控制器。

#### Spansion 器件

- 禁能直接访问控制器和简介访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。
- 设置模式位配置寄存器（0x28）中的 XIP 模式位为 8'b1010000。
- 通过设置 QSPI 配置寄存器中的 bit17 来使能自有控制器的 XIP 模式。
- 如有需要，重新使能直接访问控制器，简介访问控制器。

#### 31.5.5.3 退出XIP模式

为退出 XIP 模式，软件需禁止直接访问控制器和简介访问控制器，确保没有新的 AHB 读访问发送到 Flash 器件。然后将模式位设置为其他任何值，而不是相应 Flash 器件规格尚的特定模式位。接着软件需复位 QSPI 配置寄存器中的 bit17。

注意，在禁能内部 XIP 模式状态前，Flash 器件必须见到读指令，意味着，XIP 模式将一直保持有效直到下一个读指令被响应。需确保 XIP 模式在读序列结束前禁能。

#### 31.5.6 间接数据传输模式

当使用间接读控制器时，必须遵循软件序列，包括 DMA 外设接口的设置（可参考间接写控制器和间接读控制器章节。）

#### 31.5.7 AHB地址重映射

当访问经过直接访问控制器时，QSPI 的重映射功能定义了如何转换接收到的 AHB 地址。默认为 1:1 映射。当 QSPI 配置寄存器的 bit16 使能，接收到的 AHB 地址重映射至 address + N，其中 N 的值保存在重映射地址寄存器（0x24）中。建议在配置重映射寄存器前，先禁能 QSPI。

#### 31.5.8 中断响应

QSPI 提供一个高有效的中断引脚。关于所有中断源的信息，请参考中断状态寄存器（0x40）。

当软件读中断状态寄存器是，中断源清零。QSPI 控制器也可配置成，写中断状态寄存器时，中断清零。默认的清零方式为：写清零（write-to-clear）。

中断屏蔽寄存器（0x44）可屏蔽所有中断。写 0 使能，写 1 清零。

#### 31.5.9 AHB保护寄存器

发生 POR 时，AHB 保护机制唤醒禁能状态。软件可利用保护寄存器来组织 AHB 写特性寄存器。在配置保护寄存器前，建议禁能 QSPI。

## 31.6 特殊功能寄存器

### 31.6.1 寄存器列表

QSPI 控制寄存器列表			
名称	偏移地址	类型	描述
QSPI 基地址: 4000_E000 <sub>H</sub>			
QSPI_CR	0000 <sub>H</sub>	R/W	QSPI 配置寄存器
QSPI_DRIR	0004 <sub>H</sub>	R/W	QSPI 器件读指令寄存器
QSPI_DWIR	0008 <sub>H</sub>	R/W	QSPI 器件写指令寄存器
QSPI_DDLR	000C <sub>H</sub>	R/W	QSPI 器件延时寄存器
QSPI_RDCLR	0010 <sub>H</sub>	R/W	QSPI 读数据捕捉寄存器
QSPI_DSCR	0014 <sub>H</sub>	R/W	QSPI 器件大小寄存器
QSPI_SPR	0018 <sub>H</sub>	R/W	QSPI SRAM 分块配置寄存器
QSPI_IATR	001C <sub>H</sub>	R/W	QSPI 间接 AHB 地址触发寄存器
QSPI_DMACR	0020 <sub>H</sub>	R/W	QSPI DMA 外设配置寄存器
QSPI_RAR	0024 <sub>H</sub>	R/W	QSPI 地址重映射寄存器
QSPI_MBR	0028 <sub>H</sub>	R/W	QSPI 模式位寄存器
QSPI_SFLR	002C <sub>H</sub>	R/W	QSPI SRAM 数据深度状态寄存器
QSPI_TXHR	0030 <sub>H</sub>	R/W	QSPI 发送阈值寄存器
QSPI_RXHR	0034 <sub>H</sub>	R/W	QSPI 接收阈值寄存器
QSPI_WCR	0038 <sub>H</sub>	R/W	QSPI 写完成控制寄存器
QSPI_PER	003C <sub>H</sub>	R/W	QSPI 轮询结束寄存器
QSPI_IFR	0040 <sub>H</sub>	R/C_W1	QSPI 中断状态寄存器
QSPI_IMR	0044 <sub>H</sub>	R/W	QSPI 中断屏蔽寄存器
Reserved	0048 <sub>H</sub>	-	保留
Reserved	004C <sub>H</sub>	-	保留
QSPI_WPLR	0050 <sub>H</sub>	R/W	QSPI 写保护低位寄存器
QSPI_WPHR	0054 <sub>H</sub>	R/W	QSPI 写保护高位寄存器
QSPI_WPCR	0058 <sub>H</sub>	R/W	QSPI 写保护寄存器
Reserved	005C <sub>H</sub>	-	保留
QSPI_IRTR	0060 <sub>H</sub>	R/W	QSPI 间接读传输控制寄存器
QSPI_IRTWR	0064 <sub>H</sub>	R/W	QSPI 间接读传输深度阈值寄存器
QSPI_IRTSAR	0068 <sub>H</sub>	R/W	QSPI 间接读传输起始寄存器
QSPI_IRTNR	006C <sub>H</sub>	R/W	QSPI 间接度传输字节数寄存器
QSPI_IWTR	0070 <sub>H</sub>	R/W	QSPI 间接写传输控制寄存器
QSPI_IWTWR	0074 <sub>H</sub>	R/W	QSPI 间接写传输深度阈值寄存器
QSPI_IWTSAR	0078 <sub>H</sub>	R/W	QSPI 间接写传输起始寄存器
QSPI_IWTNR	007C <sub>H</sub>	R/W	QSPI 间接写传输字节数寄存器
QSPI_ITARR	0080 <sub>H</sub>	R/W	QSPI 间接触发地址范围寄存器
Reserved	0084 <sub>H</sub> ~008C <sub>H</sub>	-	保留

QSPI 控制寄存器列表			
名称	偏移地址	类型	描述
QSPI 基地址: 4000_E000 <sub>H</sub>			
QSPI_FCR	0090 <sub>H</sub>	R/W	QSPI Flash 命令控制寄存器
QSPI_FCAR	0094 <sub>H</sub>	R/W	QSPI Flash 命令地址寄存器
Reserved	0098 <sub>H</sub> ~009C <sub>H</sub>	-	保留
QSPI_FCRLR	00A0 <sub>H</sub>	R	QSPI Flash 命令读数据低位寄存器
QSPI_FCRHR	00A4 <sub>H</sub>	R	QSPI Flash 命令读数据高位寄存器
QSPI_FCWLR	00A8 <sub>H</sub>	R/W	QSPI Flash 命令写数据低位寄存器
QSPI_FCWHR	00AC <sub>H</sub>	R/W	QSPI Flash 命令写数据高位寄存器
QSPI_PFSR	00B0 <sub>H</sub>	R	QSPI 轮询 Flash 状态寄存器
Reserved	00B4 <sub>H</sub> ~00F8 <sub>H</sub>	-	保留
QSPI_MIDR	00FC <sub>H</sub>	R	QSPI 模块 ID 寄存器

### 31.6.2 寄存器描述

#### 31.6.2.1 QSPI配置寄存器 (QSPI\_CR)

QSPI 配置寄存器 (QSPI_CR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 10000000_01111000_00000000_10000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLES	Reserved					DTRM	ADEN	BAUD				XIPIM	XIPNX	AREN	DMAEN	SWPP	PSL					PSD	LIMEN	DACEN	Reserved			CPHA	CPOL	EN	

IDLES	Bit 31	R	<p>串行接口和 <b>QSPI PIPELINE</b> 处于空闲</p> <p>该位是一个重新计时的信号，所以生产该状态位信号时会有一些延时。</p> <p>在设置 IDLE 位有效和切换配置域之间，建议等待至少 4 个 ref_clk 周期。该延时可确保底层的同步，FIFO 为空，控制器可引入任何操作模式</p>
Reserved	Bit 30-25	R	保留，读作 0。
DTRM	Bit 24	RW	<p><b>DTR 协议使能</b></p> <p>如果器件使用 DTR 协议，则置位该位。</p> <p>注意，DTR 协议在 DTR 模式下提供所有命令。有些 DTR 读命令可在 STR 协议中处理（在 DTR 模式下）。将该位设置为 0 来执行这些命令。STR 协议中的 DTR 命令由器件读配置寄存器中的第 10 位来控制。</p>
ADEN	Bit 23	RW	<p><b>AHB 解码器使能（仅限直接访问模式）</b></p> <p>当设置为 1 时，bits[13:10]为无关位。</p> <p>有效从机取决于实际的 AHB 地址。</p> <p>每个器件的配置基于器件大小配置寄存器中的 bits[28:21]来计算。</p>
BAUD	Bit 22-19	RW	<p><b>主模式波特率分频</b></p> <p>SPI 波特率 = (主参考时钟) / BD</p> <p>其中 BD 可配置为以下：</p> <p>0000 = /2</p> <p>0001 = /4</p> <p>0010 = /6</p> <p>0011 = /8</p> <p>0100 = /10</p> <p>0101 = /12</p> <p>0110 = /14</p> <p>0111 = /16</p>

			<p>1000 = /18</p> <p>...</p> <p>1111 = /32</p> <p>使能 QSPI 控制器前，优先设置好该寄存器。</p> <p>注：DTR 模式，SPI 波特率不支持"0000"配置。</p>
XIPIM	Bit 18	R/W	<p><b>即刻进入 XIP 模式</b></p> <p>0：如果 XIP 使能，当下一个读指令完成后，控制器将退出 XIP 模式。</p> <p>1：即刻进入 XIP 模式</p> <p>当外部器件从 XIP 模式唤醒时，使用该寄存器。控制器假定下一个读指令将作为 XIP 指令传递给器件，因此不会要求传输读操作码。</p> <p>注意：为退出 XIP 模式，该位需设置为 0。只有在下一个读指令执行后，才能在连接的器件中生效。因此，确定退出 XIP 模式前，在复位该位后，软件必须确保至少有一个读指令。</p> <p>该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。</p>
XIPNX	Bit 17	R/W	<p><b>在下一个读指令进入 XIP 模式</b></p> <p>0：如果 XIP 使能，当下一个读指令完成后，控制器将退出 XIP 模式。</p> <p>1：如果 XIP 禁能，该位置 1 可通知控制器，器件已就绪，可在下一个读指令进入 XIP 模式。接着，控制器将发送合适的命令序列，包括让器件进入 XIP 模式的模式位。</p> <p>当控制器已确保接连的 Flash 器件已准备就绪可进入 XIP 模式后，使用该寄存器。</p> <p>注意：为退出 XIP 模式，该位需设置为 0。只有在下一个读指令执行后，才能在连接的器件中生效。因此，确定退出 XIP 模式前，在复位该位后，软件必须确保至少有一个读指令。</p> <p>该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。</p>
AREN	Bit 16	R/W	<p><b>AHB 地址重映射使能（仅限直接访问模式）</b></p> <p>当该位置 1，接收到的 AHB 地址将更新，并将 (address+N) 发送至 Flash 器件，其中 N 的值保存在重映射地址寄存器中。</p>

			该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。												
DMAEN	Bit 15	R/W	<p><b>DMA 外设接口使能</b></p> <p>0: 禁能 1: 使能 DMA 握手逻辑。使能后，QSPI 会通过 DMA 外设接口触发 DMA 传输请求。</p> <p>该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。</p>												
SWPP	Bit 14	R/W	<p><b>写保护引脚设置</b></p> <p>设置该位来驱动 Flash 器件的写保护引脚，与生成的存储器时钟重新同步。</p> <p>注意，WP 引脚仅在 SINGLE 或 DUAL 传输模式下有效。在 QUAD 传输模式下，WP 引脚用来传输数据，因此该寄存器的设置将被无视。</p> <p>该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。</p>												
PSL	Bit 13-10	R/W	<p><b>外设片选线</b></p> <p>如果寄存器中的 bit9 设置为 0，ss[3:0]为输出：</p> <table border="0"> <tr> <td>ss[3:0]</td> <td>n_ss_out[3:0]</td> </tr> <tr> <td>xxx0</td> <td>1110</td> </tr> <tr> <td>xx01</td> <td>1101</td> </tr> <tr> <td>x011</td> <td>1011</td> </tr> <tr> <td>0111</td> <td>0111</td> </tr> <tr> <td>1111</td> <td>1111 (无外设选中)</td> </tr> </table> <p>其他情况下，ss[3:0]直接驱动 n_ss_out[3:0]</p>	ss[3:0]	n_ss_out[3:0]	xxx0	1110	xx01	1101	x011	1011	0111	0111	1111	1111 (无外设选中)
ss[3:0]	n_ss_out[3:0]														
xxx0	1110														
xx01	1101														
x011	1011														
0111	0111														
1111	1111 (无外设选中)														
PSD	Bit 9	R/W	<p><b>外设选择解码</b></p> <p>0 = only 1 of 4 selects n_ss_out[3:0] is active 1 = allow external 4-to-16 decode (n_ss_out = ss)</p>												
LIMEN	Bit 8	R/W	<p><b>传统 IP 模式使能</b></p> <p>0: 使用直接访问控制器/简介访问控制器或 STIG 接口用作数据传输 1: 使能传统模式。在传统模式下，任何 AHB 接口写操作都是串行发送至 Flash 器件。任何有效的 AHB 读会从内部 RX-FIFO 出栈，在 SPI 线上获取有外部 Flash 器件转发的数据，通过 HSIZE 输入控制和传输 4,2 或 1 个字节。</p> <p>该位由硬件同步。控制器运行过程中，可对该位进行设置或清零。</p>												

DACEN	Bit 7	R/W	<p><b>直接访问控制器使能</b></p> <p>0: 一旦当前数据字 (FF_W) 传输完成, 禁能直接访问控制器。</p> <p>1: 使能直接访问控制器</p> <p>当直接访问控制器和间接访问控制器均禁能, 所有 AHB 请求完成时都会产生错误反应。</p> <p>该位由硬件同步。控制器运行过程中, 可对该位进行设置或清零。</p>
Reserved	Bit 6-3	R	<b>保留, 读作 0。</b>
CPHA	Bit 2	R/W	<p><b>时钟相位</b></p> <p>时钟相位映射到标准 SPI 时钟相位传输格式。</p> <p>注意: 当运行于 DDR 模式或者 DDR 协议时, 保持该位为低电平。</p>
CPOL	Bit 1	R/W	<p><b>SPI 字外的时钟极性</b></p> <p>映射到标准 SPI 时钟极性传输格式。</p> <p>注意: 当运行于 DDR 模式或者 DDR 协议时, 保持该位为低电平。</p>
EN	Bit 0	R/W	<p><b>QSPI 使能</b></p> <p>0: 一旦当前数据字 (FF_W) 传输完成, 禁能 QSPI。</p> <p>1: 使能 QSPI</p> <p>当 spi_enable=0, 所有的输出使能为无效, 且所有引脚设置为输入模式。</p> <p>该位由硬件同步。</p>

### 31.6.2.2 QSPI器件读指令寄存器 (QSPI\_DRIR)

QSPI 器件读指令寄存器 (QSPI_DRIR)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DCYC				Reserved		MBEN	Reserved		DMODE		Reserved		ADMODE		Reserved		DDRM		IMODE		RINST								

Reserved	Bit 31-29	R	保留, 读作 0。
DCYC	Bit 28-24	R/W	读指令所需的 Dummy 时钟周期数
Reserved	Bit 23-21	R	保留, 读作 0。
MBEN	Bit 20	R/W	模式位使能 置 1 确保模式位配置寄存器中定义的模式位跟在地址字节后发出。
Reserved	Bit 19-18	R	保留, 读作 0。
DMODE	Bit 17-16	R/W	标准 SPI 模式下的数据传输类型 (如 bits[9:8]所定义) 0: SIO 模式 – 在 DQ0 上发送数据去器件, DQ1 上接收器件上的数据 1: 用作 DUAL 输入/输出指令。数据传输时, DQ0 和 DQ1 均用做为输入和输出。 2: 用作 QUAD 输入/输出指令。数据传输时, DQ0, DQ1, DQ2 和 DQ4 均用做为输入和输出。
Reserved	Bit 15-14	R	保留, 读作 0。
ADMODE	Bit 13-12	R/W	标准 SPI 模式下的地址传输类型 (如 bits[9:8]所定义) 0: 地址仅可在 DQ0 上发送去器件。 1: 地址仅可在 DQ0 和 DQ1 上发送去器件。 2: 地址可在 DQ0, DQ1, DQ2 和 DQ3 上发送去器件。
Reserved	Bit 11	R	保留, 读作 0。
DDRM	Bit 10	R/W	DDR 位使能 当 OPCODE 的 bit7~0 符合 DDR 命令, 该位置 1。
IMODE	Bit 9-8	R/W	指令类型 0: 标准 SPI 模式 (始终在 DQ0 上发送指令去器件) 1: DIO-SPI 模式 (始终在 DQ0 和 DQ1 上发送指令, 地址和数据) 2: QIO-SPI 模式 (始终在 DQ0, DQ1, DQ2 和 DQ3 上发送指令, 地址和数据)



			注意：该位不仅仅只与读传输相关。该位为全局设置，将影响 DAC 和 INDAC 读，写和任何 STIG 传输。
RINST	Bit 7-0	R/W	在非 XIP 模式下，读操作码

### 31.6.2.3 QSPI器件写指令配置寄存器 (QSPI\_DWIR)

QSPI 器件写指令配置寄存器 (QSPI_DWIR)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DCYC				Reserved						DMODE		Reserved		ADMODE		Reserved			WELD		WINST							

Reserved	Bit 31-29	R	保留，读作 0。
DCYC	Bit 28-24	R/W	写指令所需的 Dummy 时钟周期数
Reserved	Bit 23-18	R	保留，读作 0。
DMODE	Bit 17-16	R/W	标准 SPI 模式下的数据传输类型 (如 bits[9:8]所定义) 0: SIO 模式 – 在 DQ0 上发送数据去器件，DQ1 上接收器件上的数据 1: 用作 DUAL 输入/输出指令。数据传输时，DQ0 和 DQ1 均用做为输入和输出。 2: 用作 QUAD 输入/输出指令。数据传输时，DQ0, DQ1, DQ2 和 DQ4 均用做为输入和输出。
Reserved	Bit 15-14	R	保留，读作 0。
ADMODE	Bit 13-12	R/W	标准 SPI 模式下的地址传输类型 (如 bits[9:8]所定义) 0: 地址仅可在 DQ0 上发送去器件。 1: 地址仅可在 DQ0 和 DQ1 上发送去器件。 2: 地址可在 DQ0, DQ1, DQ2 和 DQ3 上发送去器件。
Reserved	Bit 11-9	R	保留，读作 0。
WELD	Bit 8	R/W	<b>WEL 禁能</b> 在写操作前，该位允许关闭自动发送 WEL 命令
WINST	Bit 7-0	R/W	写操作码

### 31.6.2.4 QSPI器件延时寄存器 (QSPI\_DDLR)

QSPI 器件延时寄存器 (QSPI_DDLR)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSDA								CADADS								CSEOT								CSSOT							

CSDA	Bit 31-24	R/W	<p><b>片选设为无效</b></p> <p>在两个传输之间，主模式片选输出设为无效的时间（主参考时钟 <b>ref_clk</b> 上的延时）</p> <p>片选设为无效的最小延时：：</p> <p><b>1 sclk_out + 1 ref_clk</b></p> <p>为确保片选不会在 1 个 <b>sclk_out</b> 周期内再次变为有效。</p> <p>如果 <b>CSDA=X</b>，则片选无效时间为：</p> <p><b>1 sclk_out + 1 ref_clk + X ref_clks</b></p>
CSDADS	Bit 23-16	R/W	<p><b>不同从机的片选设为无效</b></p> <p>一个片选设为无效和另一个片选设为有效，其之间的延时（主参考时钟 <b>ref_clk</b> 上的延时）。该延时确保了两个不同从机切换之间的安静周期。当在两个外部 <b>Flash</b> 器件切换时，该位才有效。</p> <p>举例：有 2 个 <b>Flash</b> 器件 <b>A</b> 和 <b>B</b>，作为不同的片选与控制器相连。用户有需求要在每个连续的非序列的传输之间，在特定时间段内，将片选设为无效。此时，用户可使用 <b>CSDA</b> 来进行控制。<b>A</b> 器件上传输的结束和 <b>B</b> 器件上传输的开始之间，将片选设为无效的时间要求可能不同。此时，用户可使用 <b>CSDADS</b> 来控制。</p> <p>最小延时 (<b>CSDADS=0</b>):</p> <p><b>1 sclk_out + 3 ref_clks.</b></p> <p>如果 <b>CSDADS=X</b>，则延时为：</p> <p><b>1 sclk_out + 3 ref_clks + X ref_clks</b></p>
CSEOT	Bit 15-8	R/W	<p><b>片选传输结束</b></p> <p>当前传输的最后一位与将片选设 (<b>n_ss_out</b>) 为无效之间的延时（主参考时钟 <b>ref_clk</b> 上的延时）</p> <p>默认，当 <b>CSEOT=0</b>，在当前传输结束时，片选将在 <b>sclk_out</b> 的最后一个下降沿上设为无效。</p>
CSSOT	Bit 7-0	R/W	<p><b>片选传输起始</b></p>

			<p>设置 n_ss_out 为低电平和传输第一个比特位之间的延时。</p> <p>默认，当 CSSOT=0，在 sclk_out 的第一个上升沿之间的半个 SCLK 周期，片选将被设为有效。</p> <p>如果 CSSOT=X，在 sclk_out + X ref_clks 的第一个上升沿之间的半个 sclk_out K 周期，片选将被设为有效。</p>
--	--	--	---

### 31.6.2.5 QSPI读数据捕捉寄存器 (QSPI\_RDCR)

QSPI 数据读捕捉寄存器 (QSPI_RDCR)																																	
偏移地址: 010 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved												DLYT				Reserved												SMES	DLYR				BYLPC

Reserved	Bit 31-20	R	保留，读作 0。
DLYT	Bit 19-16	R/W	<p><b>传输数据延时</b></p> <p>(目的为在 DDR 模式中提高传输过程中的保持时间)</p> <p>当 DDR 读命令执行时，该位才有效；否则，无视。</p> <p>delay_value = 1 x ref_clk + &lt;field_value&gt;*ref_clk</p>
Reserved	Bit 15-6	R	保留，读作 0。
SMES	Bit 5	R/W	<p><b>采样边沿选择 (Flash 存储器数据输出)</b></p> <p>0: 在 ref_clk 的下降沿对 Flash 存储器的数据输出进行采样</p> <p>1: 在 ref_clk 的上升沿沿对 Flash 存储器的数据输出进行采样</p>
DLYR	Bit 4-1	R/W	<p><b>读数据捕捉延时</b> (延时时间为设定的 ref_clk 周期数)</p>
BYLPC	Bit 0	R/W	<p><b>旁路适配后的回环时钟电路 Bypass of the adapted loopback clock circuit</b></p> <p>0: 使能适配回环时钟电路</p> <p>1: 禁能适配回环时钟电路</p>

### 31.6.2.6 QSPI器件大小配置寄存器 (QSPI\_DSCR)

QSPI 器件大小配置寄存器 (QSPI_DSCR)																																		
偏移地址: 014 <sub>H</sub>																																		
复位值: 00000000_00010000_00010000_00000010 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved				CS3SIZE			CS2SIZE			CS1SIZE			CS0SIZE			BKSIZE					PASIZE										ADSIZE			

Reserved	Bit 31-29	R	保留, 读作 0。
CS3SIZE	Bit 28-27	R/W	连接 <b>CS[3]</b> 引脚的 <b>Flash</b> 器件大小 当 AHB 解码器使能时, 该位有效。 00: 512Mb 01: 1Gb 10: 2Gb 11: 4Gb
CS2SIZE	Bit 26-25	R/W	连接 <b>CS[2]</b> 引脚的 <b>Flash</b> 器件大小 当 AHB 解码器使能时, 该位有效。 00: 512Mb 01: 1Gb 10: 2Gb 11: 4Gb
CS1SIZE	Bit 24-23	R/W	连接 <b>CS[1]</b> 引脚的 <b>Flash</b> 器件大小 当 AHB 解码器使能时, 该位有效。 00: 512Mb 01: 1Gb 10: 2Gb 11: 4Gb
CS0SIZE	Bit 22-21	R/W	连接 <b>CS[0]</b> 引脚的 <b>Flash</b> 器件大小 当 AHB 解码器使能时, 该位有效。 00: 512Mb 01: 1Gb 10: 2Gb 11: 4Gb
BKSIZE	Bit 20-16	R/W	每块的字节数 控制器需要该位来执行写保护逻辑。每块的字节数必须为 2 的 N 次方。 0: 1 byte 1: 2 bytes 2: 4 bytes 3: 8 bytes 16: 65535 bytes

PASIZE	Bit 15-4	R/W	每个器件页的字节数 控制器需要该位来执行跨页的 Flash 写操作。
ADSIZE	Bit 3-0	R/W	地址字节数 0: 1 byte ...

### 31.6.2.7 QSPI SRAM分块配置寄存器 (QSPI\_SPR)

QSPI SRAM 分块配置寄存器 (QSPI_SPR)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00100000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SRAMPS							

Reserved	Bit 31-8	R	保留, 读作 0。
SRAMPS	Bit 7-0	R/W	定义了 SRAM 中间接读区域的大小。默认地, SRAM 中一半的大小预留给间接读操作, 另一个分配给间接写操作。该寄存器大小可随着 SRAM 深度改变。 分配给间接读的地址数量= SRAM_PARTITION_REG+1  分配给间接写的地址数量= (2**8)-SRAM_PARTITION_REG  不要将 SRAM_PARTITION_REG 设置为 0x0000 或 (2**8)-1

### 31.6.2.8 QSPI间接AHB地址触发寄存器 (QSPI\_IATR)

QSPI 间接 AHB 地址触发寄存器 (QSPI_IATR)																															
偏移地址: 01C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INDTAD																															

INDTAD	Bit 31-0	R/W	<b>间接触发地址</b> 该地址为基地址。当接收到的 AHB 读访问地址与 (从触发地址到触发地址+<配置范围) 内的地址匹配, 则通过从间接控制器 SRAM 中获取数据来完成 AHB 请求。
--------	----------	-----	--

### 31.6.2.9 QSPI DMA外设配置寄存器 (QSPI\_DMACR)

QSPI DMA 外设配置寄存器 (QSPI_DMACR)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BNUMB				Reserved				SNUMB											

Reserved	Bit 31-12	R	保留, 读作 0。
BNUMB	Bit 11-8	R/W	<b>DMA 外设请求中 Burst 类型的字节数</b> 设置为 0 时, 代表一个字节。该位需在间接读/写开始前设置完成。实际的字节数为 2**(寄存器设置值)
Reserved	Bit 7-4	R	保留, 读作 0。
SNUMB	Bit 3-0	R/W	<b>DMA 外设请求中 Single 类型的字节数</b> 设置为 0 时, 代表一个字节。该位需在间接读/写开始前设置完成。实际的字节数为 2**(寄存器设置值)

### 31.6.2.10 QSPI地址重映射寄存器 (QSPI\_RAR)

QSPI 地址重映射寄存器 (QSPI_RAR)																															
偏移地址: 024 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READDR																															

READDR	Bit 31-0	R/W	该寄存器将接收到的 AHB 地址重映射到一个不同的地址。
--------	----------	-----	------------------------------

### 31.6.2.11 QSPI模式位配置寄存器 (QSPI\_MBR)

QSPI 模式位配置寄存器 (QSPI_MBR)																															
偏移地址: 028 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								MODEB							

Reserved	Bit 31-8	R	保留, 读作 0。
MODEB	Bit 7-0	R/W	模式位 在地址字节后, 向外部器件发送这 8 位模式位

### 31.6.2.12 QSPI SRAM数据深度状态寄存器 (QSPI\_SFLR)

QSPI SRAM 数据深度状态寄存器 (QSPI_SFLR)																															
偏移地址: 02C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INDWSFL				Reserved								INDRSFL			

Reserved	Bit 31-24	R	保留
INDWSFL	Bit 23-16	R	SRAM 深度状态 (间接写区域), 单位为字 (4 个字节)
Reserved	Bit 15-8	R	保留
INDRSFL	Bit 7-0	R	SRAM 深度状态 (间接读区域), 单位为字 (4 个字节)

### 31.6.2.13 QSPI发送阈值寄存器 (QSPI\_TXHR)

QSPI 发送阈值寄存器 (QSPI_TXHR)																															
偏移地址: 030 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TXTH							

Reserved	Bit 31-5	R	保留, 读作 0。
TXTH	Bit 4-0	R/W	小容量 TX FIFO 不满阈值设置值 只有在传统模式下访问 Flash 器件, 该位才有效。 其他情况下可无视。



### 31.6.2.14 QSPI接收阈值寄存器 (QSPI\_RXHR)

QSPI 接收阈值寄存器 (QSPI_RXHR)																															
偏移地址: 034 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												RXTH			

Reserved	Bit 31-5	R	保留, 读作 0。
RXTH	Bit 4-0	R/W	小容量 RX FIFO 不空阈值设置值 只有在传统模式下访问 Flash 器件, 该位才有效。 其他情况下可无视。

### 31.6.2.15 QSPI写完成控制寄存器 (QSPI\_WCR)

QSPI 写完成控制寄存器 (QSPI_WCR)																															
偏移地址: 038 <sub>H</sub>																															
复位值: 00000000_00000001_00000000_00000101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREPD				PCNT								Reserved	PDIS	PPLT	Reserved	PBIND			OPCODE												

PREPD	Bit 31-24	R/W	<b>轮询重复延时</b> 在自动轮询阶段, 片选保持无效的额外延时时间
PCNT	Bit 23-16	R/W	<b>轮询次数</b> 定义轮询次数。当控制器吞吐量高时, 有必要添加额外的周期。
Reserved	Bit 15	R	保留, 读作 0。
PDIS	Bit 14	R/W	<b>轮询禁能</b> 关闭自动轮询功能
PPLT	Bit 13	R/W	<b>轮询极性</b> 0: 如果轮询位为 0, 则写传输完成。 1: 如果轮询位为 1, 则写传输完成。
Reserved	Bit 12-11	R	保留, 读作 0。
PBIND	Bit 10-8	R/W	<b>轮询位检索</b> 设置为 010 时, 表示轮询返回数据的 bit2。 设置为 111 时, 表示轮询返回数据的 bit7。
OPCODE	Bit 7-0	R/W	<b>操作码</b> 当自动轮询器件编程的完成状态时, 控制器必须

			<p>发送操作码。该命令在所有的器件写操作后发送。</p> <p>默认使用操作码 0x05 来轮询标准器件状态寄存器。</p>
--	--	--	---

### 31.6.2.16 QSPI轮询结束寄存器 (QSPI\_PER)

QSPI 轮询结束寄存器 (QSPI_PER)																															
偏移地址: 03C <sub>H</sub>																															
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCYCN																															

PCYCN	Bit 31-0	R/W	<p>轮询周期数</p> <p>轮询周期数后，产生轮询结束中断。</p>
-------	----------	-----	--------------------------------------

### 31.6.2.17 QSPI中断状态寄存器 (QSPI\_IFR)

QSPI 中断状态寄存器 (QSPI_IFR)																															
偏移地址: 040 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>E</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													POLLF	INDRSFF	SRFFF	SRFNEF	STFFF	STFNFF	ROVF	INDTWF	AHBAEF	WPAF	INDRRF	INDCF	UDFF	MODFF					

Reserved	Bit 31-14	R	保留, 读作 0。
POLLF	Bit 13	R/W_C1	最大的轮询周期数
INDRSFF	Bit 12	R/W_C1	SRAM 中的间接读区域满, 无法立即完成间接操作。
SRFFF	Bit 11	R/W_C1	小容量 RX FIFO 满 (当前 FIFO 状态) 在非 SPI 传统模式下, 可无视。 0: FIFO 非满 1: FIFO 满
SRFNEF	Bit 10	R/W_C1	小容量 RX FIFO 非空 (当前 FIFO 状态) 在非 SPI 传统模式下, 可无视。 0: FIFO < RX 阈值 1: FIFO ≥ 阈值
STFFF	Bit 9	R/W_C1	小容量 TX FIFO 满 (当前 FIFO 状态) 在非 SPI 传统模式下, 可无视。 0: FIFO 非满 1: FIFO 满
STFNFF	Bit 8	R/W_C1	小容量 TX FIFO 非满 (当前 FIFO 状态) 在非 SPI 传统模式下, 可无视。 0: FIFO > 阈值 1: FIFO ≤ 阈值
ROVF	Bit 7	R/W_C1	接收上溢 (仅在传统模式下发生) 0: 未检测到上溢 1: 发生上溢
INDTWF	Bit 6	R/W_C1	Indirect Transfer Watermark Level Breached
AHBAEF	Bit 5	R/W_C1	检测到非法 AHB 访问 AHB 写 Wrapping Burst 和使用 SPLIT/RETRY 访问将触发该中断。 如果 DAC 禁能状态下进行 AHB 访问, 也将触发该中断。
WPAF	Bit 4	R/W_C1	企图写保护区域被拒绝
INDRRF	Bit 3	R/W_C1	不接收间接操作请求 已存在两个间接操作。
INDCF	Bit 2	R/W_C1	控制器已完成上一个间接操作

UDFF	Bit 1	R/W_C1	<b>下溢检测</b> 0: 未检测到下溢 1: 检测到下溢: 当小容量 TX FIFO 为空时, 企图发送数据。当 AHB 写数据太慢, 跟不上写请求时, 有可能发生下溢。
MODFF	Bit 0	R/W_C1	<b>模式失效</b> 不再使用传统模式。

### 31.6.2.18 QSPI中断屏蔽寄存器 (QSPI\_IMR)

QSPI 中断屏蔽寄存器 (QSPI_IMR)																															
偏移地址: 044 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														INTEN																	

Reserved	Bit 31-14	R	保留, 读作 0。
INTEN	Bit 13-0	R/W	0: 中断状态寄存器中的相关中断位禁止 1: 中断状态寄存器中的相关中断位使能

### 31.6.2.19 QSPI写保护低位寄存器 (QSPI\_WPLR)

QSPI 写保护低位寄存器 (QSPI_WPLR)																															
偏移地址: 050 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LBLKNUM																															

LBLKNUM	Bit 31-0	R/W	定义了写保护块范围的起始块，通过器件大小配置寄存器可配置块的字节数。 当写保护功能（写保护寄存器中 bit1）为低电平，可更改该位。
---------	----------	-----	---

### 31.6.2.20 QSPI写保护高位寄存器 (QSPI\_WPHR)

QSPI 写保护高位寄存器 (QSPI_WPHR)																															
偏移地址: 054 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBLKNUM																															

HBLKNUM	Bit 31-0	R/W	定义了写保护块范围的结束块，通过器件大小配置寄存器可配置块的字节数。 当写保护功能（写保护寄存器中 bit1）为低电平，可更改该位。
---------	----------	-----	---

### 31.6.2.21 QSPI写保护寄存器 (QSPI\_WPCR)

QSP 写保护配置寄存器 (QSPI_WPCR)																															
偏移地址: 058 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													WPEN		

Reserved	Bit 31-2	R	保留, 读作 0。
WPEN	Bit 1	R/W	<b>写保护使能</b> 1: 拒绝访问任何 AHB 写访问 (其写访问地址在写保护地位寄存器和写保护高位寄存器范围内)。产生一个 AHB 错误响应, 触发中断源。 0: 写保护禁能 该位由硬件内部同步。当改变该位状态时, 无特别软件要求。
WPINV	Bit 0	R/W	<b>写保护取反</b> 1: 系统允许写由写保护地位寄存器和写保护高位寄存器定义的保护区域的地址。 0: 系统不允许写由写保护地位寄存器和写保护高位寄存器定义的保护区域的地址。 当写保护功能 (写保护寄存器中 bit1) 为低电平, 可更改该位。

### 31.6.2.22 QSPI间接读传输控制寄存器 (QSPI\_IRTR)

QSP 间接读传输控制寄存器 (QSPI_IRTR)																															
偏移地址: 060 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								INDRNUM	RDCS	RDQS	SRAMFS	RDPS	RDDIS	RDST	

Reserved	Bit 31-8	R	保留, 读作 0。
INDRNUM	Bit 7-6	R	<b>间接操作的完成数</b> 该位与 bit5 联合使用。 当完成一个间接操作时, 由硬件递增。 当 bit5 写入 1 时, 则递减。
RDCS	Bit 5	RC_W1	<b>间接完成状态</b> 当完成一个间接操作, 该位由硬件置 1, 写 1 清零。
RDQS	Bit 4	R	<b>已有两个间接读操作排队等候</b> 该位由硬件置 1 和清零。
SRAMFS	Bit 3	RC_W1	<b>SRAM 满, 无法立即完成间接操作</b> 硬件置 1, 写 1 清零。
RDPS	Bit 2	R	<b>间接读操作进行中</b> 仅由硬件清零。
RDDIS	Bit 1	W1	<b>取消间接读</b> 写 1 取消进行中的间接读操作。可取消当前所有执行中的间接操作。 该位由硬件内部同步。
RDST	Bit 0	W1	<b>开始间接读</b> 写 1 触发间接读操作。触发间接读操作前, 先设置好间接起始地址和间接字节数寄存器。 该位由硬件内部同步。

### 31.6.2.23 QSPI间接读传输数据深度阈值寄存器 (QSPI\_IRTWR)

QSP 间接读传输数据深度阈值寄存器 (QSPI_IRTWR)																															
偏移地址: 064 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VAULE	Bit 31-0	R/W	<b>数据深度</b> 在 DMA 外设访问前, 该值代表的是 SRAM 的最小数据深度。当 SRAM 数据深度超越该值, 则触发中断源。写全 0 可禁能。 单位为字节。
-------	----------	-----	---

### 31.6.2.24 QSPI间接读传输起始寄存器 (QSPI\_IRTSAR)

QSP 间接读传输起始地址设置寄存器 (QSPI_IRTSAR)																															
偏移地址: 068 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

ADDR	Bit 31-0	R/W	间接访问起始地址
------	----------	-----	----------

### 31.6.2.25 QSPI间接读传输字节数寄存器 (QSPI\_IRTNR)

QSPI 间接读传输数据字节数寄存器 (QSPI_IRTNR)																															
偏移地址: 06C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM																															

NUM	Bit 31-0	R/W	<b>间接字节数</b> 该设置值可大于 SRAM 大小。
-----	----------	-----	----------------------------------



### 31.6.2.26 QSPI间接写传输控制寄存器 (QSPI\_IWTR)

QSP 间接写传输控制寄存器 (QSPI_IWTR)																															
偏移地址: 070 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														INDWNUM	WRCS	WRQS	Reserved	WRPS	WRDIS	WRST											

Reserved	Bit 31-8	R	保留, 读作 0。
INDWNUM	Bit 7-6	R	间接操作的完成数 该位与 bit5 联合使用。 当完成一个间接操作时, 由硬件递增。 当 bit5 写入 1 时, 则递减。
WRCS	Bit 5	RC_W1	间接完成状态 当完成一个间接操作, 该位由硬件置 1, 写 1 清零。
WRQS	Bit 4	R	已有两个间接写操作排队等候 该位由硬件置 1 和清零。
Reserved	Bit 3	RC_W1	保留, 读作 0
WRPS	Bit 2	R	间接写操作进行中 仅由硬件清零。
WRDIS	Bit 1	W1	取消间接写 写 1 取消进行中的间接写操作。可取消当前所有执行中的间接操作。 该位由硬件内部同步。
WRST	Bit 0	W1	开始间接写 写 1 触发间接写操作。触发间接写操作前, 先设置好间接起始地址和间接字节数寄存器。 该位由硬件内部同步。

### 31.6.2.27 QSPI间接写传输数据深度阈值寄存器 (QSPI\_IWTWR)

QSP 间接写传输数据深度阈值寄存器 (QSPI_IWTWR)																															
偏移地址: 074 <sub>H</sub>																															
复位值: 11111111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															

VAULE	Bit 31-0	R/W	<b>数据深度</b> 在 DMA 外设访问前, 该值代表的是 SRAM 的最大数据深度。当 SRAM 数据深度低于该值, 则触发中断源。写全 1 可禁能。 单位为字节。
-------	----------	-----	---

### 31.6.2.28 QSPI间接写传输起始寄存器 (QSPI\_IWTSAR)

QSP 间接读传输起始地址设置寄存器 (QSPI_IWTSAR)																															
偏移地址: 078 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

ADDR	Bit 31-0	R/W	间接访问起始地址
------	----------	-----	----------

### 31.6.2.29 QSPI间接写传输字节数寄存器 (QSPI\_IWTNR)

QSPI 间接读传输数据字节数寄存器 (QSPI_IWTNR)																															
偏移地址: 07C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM																															

NUM	Bit 31-0	R/W	<b>间接字节数</b> 该设置值可大于 SRAM 大小。
-----	----------	-----	----------------------------------

### 31.6.2.30 QSPI间接触发地址范围寄存器 (QSPI\_ITARR)

QSP 间接地址范围设置寄存器 (QSPI_ITARR)																															
偏移地址: 080 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											RNGW				

Reserved	Bit 31-4	R	保留, 读作 0
RNGW	Bit 3-0	R/W	<p><b>间接范围宽度</b></p> <p>该位为间接触发地址寄存器的地址偏移。当触发间接访问且 AHB 地址在此范围内, 访问请求就会发送至间接写/读控制器。</p> <p>该值为 2 的 N 次方。默认 2**4=16, 允许进行 16 字节的 Burst。该位段反映了范围宽度, 有效地址的个数 (单个地址 32 位) 从间接触发地址到间接触发地址 + &lt;间接范围宽度-1&gt;(间接触发地址-&gt; 间接触发地址 + 15</p>

### 31.6.2.31 QSPI Flash命令控制寄存器 (QSPI\_FCR)

QSP Flash 命令控制寄存器 (QSPI_FCR)																															
偏移地址: 090 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE								RDEN	RDNUM			ADDREN	MOBBEN	ADNUM		WREN	WDNUM			DUMNUM			Reserved					CMD5	CMDT		

OPCODE	Bit 31-24	R/W	<p><b>命令操作码</b></p> <p>命令操作码需在触发命令前设置。举例：0x20 映射到 SubSector 擦除，写 bit0 可发送命令。注意，利用该方式发送命令，将用到器件指令配置寄存器中的指令类型。如果该位设置为 2'b00，则会连续传输命令操作码，命令地址，命令 dummy 字节，命令数据。如果该位设置为 2'b01，则会通过 DQ0 和 DQ1 引脚并行传输命令操作码，命令地址，命令 dummy 字节和命令数据。如果该位设置为 2'b10，则会通过 DQ0, DQ1, DQ2, DQ3 引脚并行传输命令操作码，命令地址，命令 dummy 字节和命令数据。</p>
RDEN	Bit 23	R/W	<p><b>读数据使能</b></p> <p>如果 bit31-24 中的命令要求接收读数据字节，将该位设置为 1。</p>
RDNUM	Bit 22-20	R/W	<p><b>读数据字节数</b></p> <p>高达 8 个数据字节可读取</p> <p>0: 1 字节 1: 2 字节 ... 7: 8 字节</p>
ADDREN	Bit 19	R/W	<p><b>命令地址使能</b></p> <p>如果 bit31-24 中的命令需要地址，将该位设置为 1。</p> <p>通过 bit0 触发命令前，先设置好该位。</p>
MOBBEN	Bit 18	R/W	<p><b>模式位使能</b></p> <p>该位置 1 确保模式位配置寄存器中的模式位在地址字节后发送。</p>
ADNUM	Bit 17-16	R/W	<p><b>地址字节数</b></p> <p>设置所需的地址字节数（地址本身在 Flash 命令地址寄存器中设置）</p> <p>通过 bit0 触发命令前，先设置好该位</p> <p>00: 1 地址字节</p>

			01: 2 地址字节 10: 3 地址字节 11: 4 地址字节
WREN	Bit 15	R/W	<b>写数据使能</b> 如果 bit31-24 中的命令要求发送写数据字节, 将该位设置为 1。
WDNUM	Bit 14-12	R/W	<b>写数据字节数</b> 高达 8 个数据字节可写 0: 1 字节 1: 2 字节 ... 7: 8 字节
DUMNUM	Bit 11-7	R/W	<b>Dummy 字节数</b> 设置所需的 dummy 字节。 通过 bit0 触发命令前, 先设置好该位。 注意, 该位得设置不能大于 5'h03。设置过多的 dummy 字节有可能触发 TX FIFO 满或忽视掉读命令的一些数据, 无法预知内核行为。 不要混淆 dummy 周期和 dummy 字节, 仔细察看 Flash 器件规格中 dummy 周期的定义。
Reserved	Bit 6-2	R	保留, 读作 0
CMDS	Bit 1	R	命令执行进行中
CMDT	Bit 0	W1	<b>执行命令</b> 该位内部同步。

### 31. 6. 2. 32 QSPI Flash命令地址寄存器 (QSPI\_FCAR)

QSP Flash 命令地址寄存器 (QSPI_FCAR)																															
偏移地址: 094 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDADR																															

CMDADR	Bit 31-0	R/W	<b>命令地址</b> 在触发 Flash 命令控制寄存器 bit0 前, 先设置好该位。该地址为 Flash 命令控制寄存器中 bit31-24 命令的地址。
--------	----------	-----	---

### 31.6.2.33 QSPI Flash命令读数据低位寄存器 (QSPI\_FCRLR)

QSP Flash 命令读数据低位寄存器 (QSPI_FCRLR)																															
偏移地址: 0A0 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDL																															

CMDDL	Bit 31-0	R	<p><b>命令读数据 (低字节)</b></p> <p>通过触发控制寄存器中的事件, 由 Flash 器件返回的任何状态或配置读操作的数据。</p> <p>当控制寄存器中的轮询位为低电平, 该寄存器有效。</p>
-------	----------	---	---

### 31.6.2.34 QSPI Flash命令读数据高位寄存器 (QSPI\_FCRHR)

QSP Flash 命令读数据高位寄存器 (QSPI_FCRHR)																															
偏移地址: 0A4 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDH																															

CMDDH	Bit 31-0	R	<p><b>命令读数据 (高字节)</b></p> <p>通过触发控制寄存器中的事件, 由 Flash 器件返回的任何状态或配置读操作的数据。</p> <p>当控制寄存器中的轮询位为低电平, 该寄存器有效。</p>
-------	----------	---	---

### 31.6.2.35 QSPI Flash命令写数据低位寄存器 (QSPI\_FCWLR)

QSP Flash 命令写数据低位寄存器 (QSPI_FCWLR)																															
偏移地址: 0A8 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDL																															

CMDDL	Bit 31-0	R	<p><b>命令写数据 (低字节)</b></p> <p>在触发 Flash 命令控制寄存器 bit0 前, 先设置好该位。</p> <p>通过触发控制寄存器中的事件, 由 Flash 器件写入的任何状态或配置写操作的数据。</p>
-------	----------	---	--

### 31.6.2.36 QSPI Flash命令写数据高位寄存器 (QSPI\_FCWHR)

QSP Flash 命令读数据高位寄存器 (QSPI_FCWHR)																															
偏移地址: 0AC <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDDH																															

CMDDH	Bit 31-0	R	<p><b>命令写数据 (高字节)</b></p> <p>在触发 Flash 命令控制寄存器 bit0 前, 先设置好该位。</p> <p>通过触发控制寄存器中的事件, 由 Flash 器件写入的任何状态或配置写操作的数据。</p>
-------	----------	---	--

### 31.6.2.37 QSPI轮询Flash状态寄存器 (QSPI\_PFSR)

QSPI 轮询 Flash 状态寄存器 (QSPI_PFSR)																															
偏移地址: 0B0 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							PSV		FLSS						

Reserved	Bit 31-9	R	保留, 读作 0
PSV	Bit 8	R	轮询状态有效 当 bit7-0 有效, 该位置 1。
FLSS	Bit 7-0	R	Flash 状态 定义了器件的实际状态寄存器。

### 31.6.2.38 QSPI模块ID寄存器 (QSPI\_MIDR)

QSPI 模块 ID 寄存器 (QSPI_MIDR)																															
偏移地址: 0FC <sub>H</sub>																															
复位值: 00000000_00000001_00000100_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID1								ID0																							

ID1	Bit 31-24	R	固定号码 与版本号 (由该寄存器的 3 LSB 定义) 相关
ID0	Bit 23-0	R	模块/版本 ID 号



## 第32章 自适应差分脉冲编码调制(ADPCM)

### 32.1 概述

ADPCM(自适应差分脉冲编码调制)是一种广泛使用于音频和音乐数据压缩的数字压缩技术。ADPCM 译码器支持 4 路相互独立译码的声道, 支持 IMA (交互式多媒体协会) 标准。

### 32.2 特性

- ◇ 支持 4 路相互独立译码通道;
- ◇ 支持 IMAADPCM (交互式多媒体协会) 标准;
- ◇ 仅支持 ADPCM 译码功能;

### 32.3 结构框图

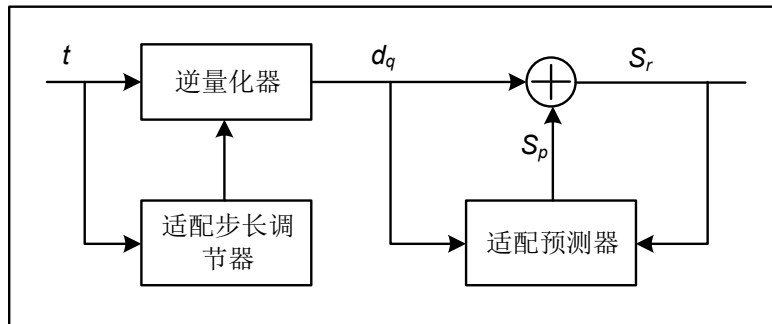


图 32-1 ADPCM 译码功能框图

### 32.4 功能描述

ADPCM 算法要求连续采样得到的音频或音乐数据具有很高的相关性。这使得相邻样本的差值很小, 因此可以很可靠的对未采样样本进行预测。ADPCM 不是直接的编码 PCM 数据, 而是编码预测数据和实际数据的差值。因为差值较小, 因此表示差值需要更少的位数, 从而实现更有效的数据压缩。流程图说明了 ADPCM 译码器的主要的功能模块。

ADPCM 的编码数据流向译码器。编码数据为 4 位有符号数, 数值大小在 7 和 -7 之间。首先, 适配步长调节器进行步长调整并预测样本, spare 设为 0。译码过程是通过使用步长调整表逆量化 4 位二进制输入数的过程。步长调整表是将输入映射到逆量化差值 dq 的编码尺度。将 dq 和预测样本加起来重建信号 Sr。最后保存重建信号, 用于产生下一个样本值。

## 32.5 特殊寄存器

### 32.5.1 寄存器列表

ADPCM 控制寄存器列表			
名称	偏移地址	类型	描述
ADPCM 基地址: 4000_8800 <sub>H</sub>			
ADPCM_DINCTL	0000 <sub>H</sub>	R/W	ADPCM 译码器输入数据和控制寄存器
ADPCM_DO10	0004 <sub>H</sub>	R	ADPCM 通道 0 和 1 译码结果寄存器
ADPCM_DO32	0008 <sub>H</sub>	R	ADPCM 通道 2 和 3 译码结果寄存器
ADPCM_IER	000C <sub>H</sub>	T_W1	ADPCM 中断使能寄存器
ADPCM_IDR	0010 <sub>H</sub>	R/W	ADPCM 中断禁能寄存器
ADPCM_IMR	0014 <sub>H</sub>	R	ADPCM 中断屏蔽寄存器
ADPCM_RIS	0018 <sub>H</sub>	R	ADPCM 中断标志寄存器
ADPCM_ISC	001C <sub>H</sub>	R/C_W1	ADPCM 中断标志清除寄存器
ADPCM_STATUS	0020 <sub>H</sub>	R	ADPCM 译码状态寄存器
ADPCM_DMACTRL	0024 <sub>H</sub>	R/W	ADPCM DMA 控制寄存器

### 32.5.2 寄存器描述

#### 32.5.2.1 ADPCM译码器输入数据和控制寄存器 (ADPCM\_DINCTL)

ADPCM 译码器输入数据控制寄存器 (ADPCM_DINCTL)																															
偏移地址: 000 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CH3INIT	CH3LOAD	CH3Data				Reserved	CH2INIT	CH2LOAD	CH2Data				Reserved	CH1INIT	CH1LOAD	CH1Data				Reserved	CH0INIT	CH0LOAD	CH0Data							

Reserved	Bit31-30	-	保留
CH3INIT	Bit29	R/W	<b>通道 3 译码状态机初始化控制位</b> 在译码时, 在第一个样本期间“CH3INIT”必须置为“1”, 用来初始化译码器内部状态。在随后的采样期间设置 CH3INIT = 0。
CH3LOAD	Bit28	R/W	<b>通道 3 有效控制位</b> 0: 通道无效 1: 通道有效 1) 为 0 时, 译码时不进行此通道的译码处理。 2) 为 1 时, 译码时进行此通道的译码处理。
CH3Data	Bit27-24	R/W	<b>通道 3, 4-bit ADPCM 输入数据</b>
Reserved	Bit23-22	-	保留
CH2INIT	Bit21	R/W	<b>通道 2 译码状态机初始化控制位</b>

			在译码时，在第一个样本期间“CH2INIT”必须置为“1”，用来初始化译码器内部状态。在随后的采样期间设置 CH2INIT = 0。
CH2LOAD	Bit20	R/W	<b>通道 2 有效控制位</b> 0: 通道无效 1: 通道有效 1) 为 0 时，译码时不进行此通道的译码处理。 2) 为 1 时，译码时进行此通道的译码处理。
CH2Data	Bit19-16	R/W	<b>通道 2, 4-bit ADPCM 输入数据</b>
Reserved	Bit15-14	-	保留
CH1INIT	Bit13	R/W	<b>通道 1 译码状态机初始化控制位</b> 在译码时，在第一个样本期间“CH1INIT”必须置为“1”，用来初始化译码器内部状态。在随后的采样期间设置 CH1INIT = 0。
CH1LOAD	Bit12	R/W	<b>通道 1 有效控制位</b> 0: 通道无效 1: 通道有效 1) 为 0 时，译码时不进行此通道的译码处理。 2) 为 1 时，译码时进行此通道的译码处理。
CH1Data	Bit11-8	R/W	<b>通道 1, 4-bit ADPCM 输入数据</b>
Reserved	Bit7-6	-	保留
CH0INIT	Bit5	R/W	<b>通道 0 译码状态机初始化控制位</b> 在译码时，在第一个样本期间“CH0INIT”必须置为“1”，用来初始化译码器内部状态。在随后的采样期间设置 CH0INIT = 0。
CH0LOAD	Bit4	R/W	<b>通道 0 有效控制位</b> 0: 通道无效 1: 通道有效 1) 为 0 时，译码时不进行此通道的译码处理。 2) 为 1 时，译码时进行此通道的译码处理。
CH0Data	Bit 3-0	R/W	<b>通道 0, 4-bit ADPCM 输入数据</b>

### 32.5.2.2 APDCM通道 0 和 1 译码结果寄存器 (ADPCM\_DO10)

ADPCM 通道 0 和 1 译码结果寄存器 (ADPCM_DO10)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1RD																CH0RD															

CH1RD	Bit31-16	R	Channel 1, 16-bit 译码结果
-------	----------	---	------------------------

CH0RD	Bit15-0	R	Channel 0, 16-bit 译码结果
-------	---------	---	------------------------

### 32.5.2.3 ADPCM通道 2 和 3 译码结果寄存器 (ADPCM\_DO32)

ADPCM 通道 2 和 3 译码结果寄存器 (ADPCM_DO32)																															
偏移地址: 008 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3RD																CH2RD															

CH3RD	Bit31-16	R	Channel 3, 16-bit 译码结果
CH2RD	Bit15-0	R	Channel 2, 16-bit 译码结果

### 32.5.2.4 ADPCM中断使能寄存器 (ADPCM\_IER)

ADPCM 中断使能寄存器 (ADPCM_IER)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															IER

Reserved	Bit31-1	-	保留
IER	Bit0	T_W1	中断使能控制位 0: 无效 1: 使能中断

### 32.5.2.5 ADPCM中断禁止寄存器 (ADPCM\_IDR)

ADPCM 中断禁止寄存器 (ADPCM_IDR)																															
偏移地址: 010 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															IDR

Reserved	Bit31-1	-	保留
----------	---------	---	----

IDR	Bit0	T_W1	中断禁止控制位 0: 无效 1: 禁止中断
-----	------	------	-----------------------------

### 32.5.2.6 ADPCM中断屏蔽寄存器 (ADPCM\_IMR)

ADPCM 中断屏蔽寄存器 (ADPCM_IMR)																																	
偏移地址: 014 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved																																	IMR

Reserved	Bit31-1	-	保留
IMR	Bit0	R	中断控制位状态指示位 0: 中断处于禁止状态 1: 中断处于使能状态

### 32.5.2.7 ADPCM中断标志寄存器 (ADPCM\_RIS)

ADPCM 中断标志寄存器 (ADPCM_RIS)																																	
偏移地址: 018 <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved																																	RIS

Reserved	Bit31-1	-	保留
RIS	Bit0	R	中断标志指示位 0: 未产生中断 1: 产生中断

### 32.5.2.8 ADPCM中断标志清除寄存器 (ADPCM\_ISC)

ADPCM 中断标志清除寄存器 (ADPCM_ISC)																																	
偏移地址: 01C <sub>H</sub>																																	

复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															ISC

Reserved	Bit31-1	-	保留
ISC	Bit0	R/C_W1	<b>中断标志清除位</b> 0: 无效 1: 清除中断标志 1) 当 IER=1 时, 读此控制位的值为 ADPCM_RIS->RIS 的值; 2) 当 IER=0 时, 读此控制位的值为 0;

### 32.5.2.9 译码状态寄存器 (ADPCM\_STATUS)

ADPCM 译码状态寄存器 (ADPCM_STATUS)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															REC_READY

Reserved	Bit31-1	-	保留
DEC_READY	Bit0	R	<b>译码状态指示位</b> 0: 译码进行中 1: 译码完成或未启动译码 此寄存器返回内部译码逻辑执行状态。当译码器处于空闲状态时返回“1”, 译码器处于忙碌状态返回“0”。向 ADPCM_DINCTL 中写入数据时, 会自动触发译码器进行工作。其中译码器工作时, 只处理 CHxLOAD 控制位为“1”的通道。当所有有效通道译码完成后, DEC_READY 置高, 表明译码完成, 可以接收新的数据进行新数据的译码处理。 如果 IER = 1, 会产生中断请求。

### 32.5.2.10 DMA控制寄存器 (ADPCM\_DMACTRL)

ADPCM DMA 控制寄存器 (ADPCM_DMACTRL)																															
偏移地址: 024 <sub>H</sub>																															

复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															DECDMAEN

Reserved	Bit31-1	-	保留
DECDMAEN	Bit0	R/W	<b>ADPCM 译码器 DMA 请求使能位</b> 1: 使能 0: 禁能 一旦译码器工作, 当 ADPCM 完成时, DMA 请求将会产生。

## 第33章 Keyboard\_Scanner (KEYS)

### 33.1 概述

Keyboard\_Scanner 在没有主机控制的情况下,可以自动检测按键事件,并且存储在缓存中。

### 33.2 特性

- ◇ 支持在没有检测到按键事件时自动关闭时钟;
- ◇ 最大支持按顺序扫描 8 行 20 列共 160 个按键;
- ◇ 支持列数可配置,配置范围 (1-20);
- ◇ 支持行数可配置,配置范围 (1-8);
- ◇ 支持 16 个 byte 的按键 index 缓存;
- ◇ 支持 128K 时钟,支持 1.2ms 内扫描 152 个按键;
- ◇ 支持 ghost 功能,最大支持 2 键锁定 (N-key roll-over);
- ◇ 按键事件存储在缓存中,在缓存没有溢出前,主机可以读取,最大支持 16 个事件存储;
- ◇ 支持硬件防弹跳和噪声以及毛刺过滤;
- ◇ 低功耗控制,在睡眠模式下电流为  $\mu\text{A}$  级;

### 33.3 结构框图

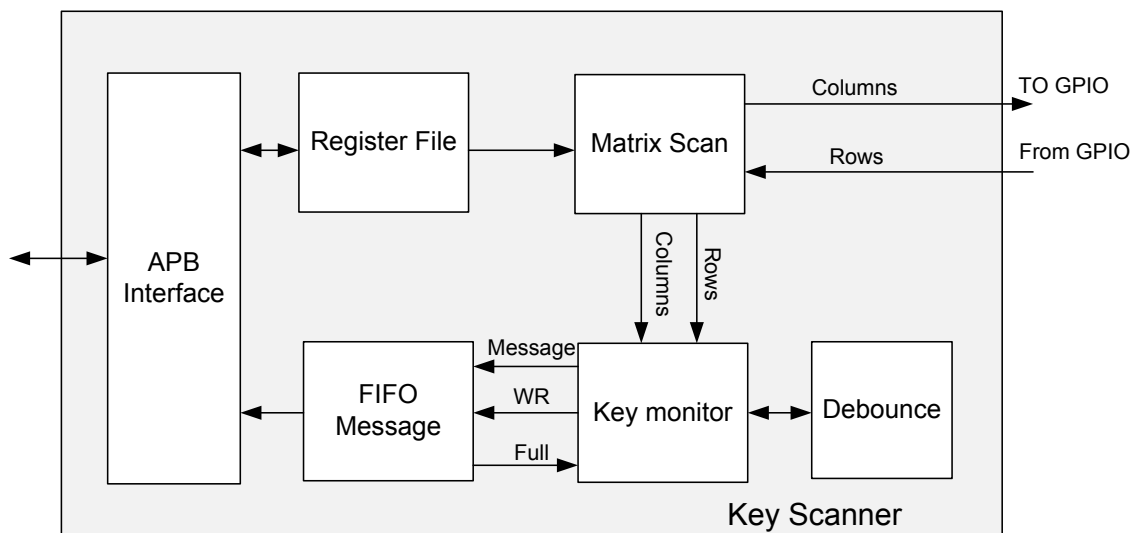


图 33-1 结构框图



## 33.4 功能描述

整个扫描过程由状态机控制实现。状态机主要分三个主状态：IDLE、SCAN、SCAN\_END。

### ◇ IDLE:

空闲状态下，如果任何按键被按下，行输入会有翻转。128K 时钟会被唤醒，状态机进入 SCAN 状态。在这个状态 key\_index，以及行命中寄存器保持为 0

### ◇ SCAN:

(1) 扫描状态下，行列计数启动轮询计数，扫描完所有的行，列计数器递增 1（即在列保持的情况下，先进行所有的行扫描，当所有的行扫描完成后，列计数器递增），所有行列轮询结束后进入 SCAN\_END 状态。

(2) 在扫描过程中，如果有按键被检测到 key\_index 被存放到 buf 中，共后续主机程序读取和查表，生成最终的 HID 报告，如果按键 key\_index 与辅助按键匹配，上报辅助按键检测标志。

(3) 在扫描过程中，支持 debounce，过滤干扰噪声，通过检测的识别为有效按键。

(4) 每检测到新的 key\_index 后，进入 scan\_down 子状态，对于满足 debounce 的 index 将存放在寄存器缓存中。

(5) 如果连续多轮扫描过程中，如果检测到的按键与寄存器缓存中 index 多次匹配到 md\_dbounce，触发 key\_down 事件，将缓存中的 key\_index 记录到事件 fifo 中。在没有对应的 key\_up 出现之前，此 index 不会响应新的 key\_down 事件。

(6) 每次检测到新的 key\_index 与缓存中的按键状态发生改变时，进入 dif 子状态，满足 debounce 后，修改当前 index 的状态为等待弹起。

(7) 如果连续多轮扫描过程中，如果检测到的按键与寄存器缓存中 index 多次匹配到 mu\_dbounce，触发 key\_up 事件，将缓存中的 key\_index 记录到事件 fifo 中。同时清空此 index 对应寄存器缓存。

(8) 扫描过程中，如果检测到 ghost 现象（检测到先后两列有两个相同的按键），做上报处理。

### ◇ SCAN\_END:

每次扫描结束后，如果有效按键，则继续扫描，如果没有有效按键，返回 IDLE 状态，时钟关闭。

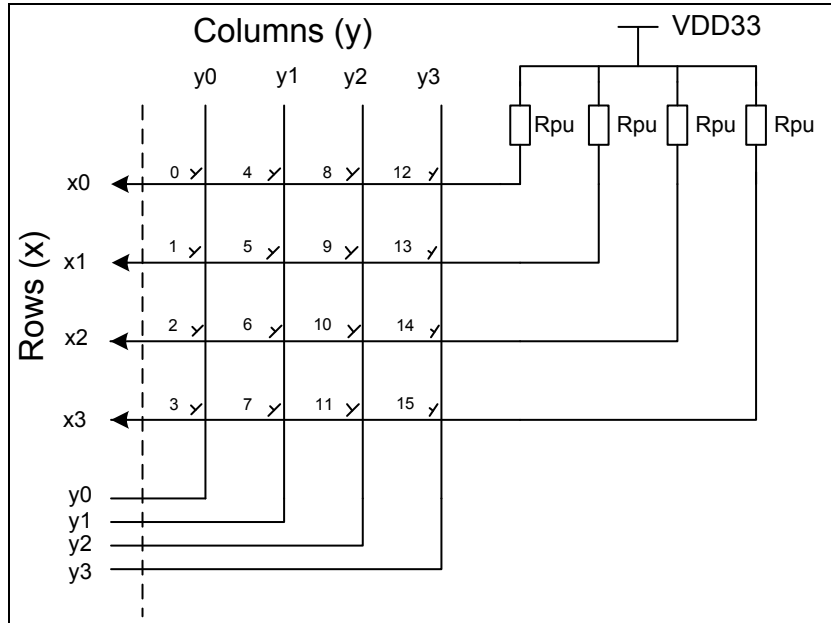


图 33-2 4x4 Keyboard Matrix 示意图

其中行 (Row) 输入线外接上拉电阻到电源 VDD33，也可以使用 GPIO 的上拉功能从而省去外部的上拉电阻简化外围电路。“图 31-2” 中的 0~15 的编号表示每一个按键的 key\_index。

### 33.5 特殊功能寄存器

#### 33.5.1 寄存器列表

KSCAN 控制寄存器列表			
名称	偏移地址	类型	描述
KSCAN 基地址: 4000_F000 <sub>H</sub>			
KSCAN_CTRL0	0000 <sub>H</sub>	R/W	Key_scanner 控制寄存器 0
KSCAN_CTRL1	0004 <sub>H</sub>	R/W	Key_scanner 控制寄存器 1
KSCAN_MODKEY0	0008 <sub>H</sub>	R/W	Key_scanner 辅助按键 0
KSCAN_MODKEY1	000C <sub>H</sub>	R/W	Key_scanner 辅助按键 1
KSCAN_MODKEY2	0010 <sub>H</sub>	R/W	Key_scanner 辅助按键 2
KSCAN_MODKEY3	0014 <sub>H</sub>	R/W	Key_scanner 辅助按键 3
KSCAN_MODKEY4	0018 <sub>H</sub>	R/W	Key_scanner 辅助按键 4
KSCAN_MODKEY5	001C <sub>H</sub>	R/W	Key_scanner 辅助按键 5
KSCAN_MODKEY6	0020 <sub>H</sub>	R/W	Key_scanner 辅助按键 6
KSCAN_MODKEY7	0024 <sub>H</sub>	R/W	Key_scanner 辅助按键 7
KSCAN_FIFOCON	0028 <sub>H</sub>	R/W	Key_scanner FIFO 控制寄存器
KSCAN_IER	002C <sub>H</sub>	R/W	Key_scanner 中断使能寄存器
KSCAN_RIF	0030 <sub>H</sub>	R	Key_scanner 中断标志寄存器
KSCAN_IFM	0034 <sub>H</sub>	R	Key_scanner 中断上报寄存器
KSCAN_ICR	0038 <sub>H</sub>	WO	Key_scanner 中断清除寄存器

### 33.5.2 寄存器描述

#### 33.5.2.1 KEY\_SCANNER控制寄存器 0 (KSCAN\_CTRL0)

KEY_SCANNER 控制寄存器 0 (KSCAN_CTRL0)																															
偏移地址: 00H																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CLK_FROZEN	KSCLK_STAYON	KS_IRQ_EN	KS_GHOST_EN	KS_FREEZE	KS_EN	Reserved										COLUMN_NUM	ROW_NUM				

Reserved	Bit31-22	—	保留
CLK_FROZEN	Bit21	R	<b>时钟关闭指示</b> 0: 时钟未关闭 1: 时钟关闭
KSCLK_STAYON	Bit20	R/W	<b>Key_scanner 时钟使能</b> 0: 时钟受门控和 FREEZE 控制 1: 时钟始终保持
KS_IRQ_EN	Bit19	R/W	<b>Key_scanner 检测上报使能</b> 0:关闭. 1:使能 注: 此控制位无效
KS_GHOST_EN	Bit18	R/W	<b>Key_scanner ghost 使能</b> 0:关闭. 1:使能
KS_FREEZE	Bit17	R/W	<b>Key_scanner 时钟关闭使能</b> 0: 时钟打开 1: 时钟关闭
KS_EN	Bit16	R/W	<b>Key_scanner 使能</b> 0:关闭 1:使能
Reserved	Bit15-8	—	保留
COLUMN_NUM	Bit7-3	R/W	列数配置选择 <b>0-19</b>
ROW_NUM	Bit2-0	R/W	行数配置选择 <b>0-7</b>

### 33. 5. 2. 2 KEY\_SCANNER控制寄存器 1 (KSCAN\_CTRL1)

KEY_SCANNER 控制寄存器 1 (KSCAN_CTRL1)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				MU_DEBOUNCE				MD_DEBOUNCE				Reserved				U_DEBOUNCE				SCCYCLE_TIMER											

Reserved	Bit31-28	—	保留
MU_DEBOUNCE	Bit27-24	R/W	<b>按键弹起 debounce 配置 0-15</b> 同一按键弹起被连续扫描到的次数大于 (16-MU_DEBOUNCE+2) 时, 识别为有效按键弹起记录到事件 FIFO 中。
MD_DEBOUNCE	Bit23-20	R/W	<b>按键按下 debounce 配置 0-15</b> 同一按键按下被连续扫描到的次数大于 (MD_DEBOUNCE+2) 时, 识别为有效按键按下记录到事件 FIFO 中。
Reserved	Bit19-14	—	保留
U_DEBOUNCE	Bit13-12	R/W	<b>Debounce 配置 0-3</b>
SCCYCLE_TIMER	Bit11-0	R/W	<b>两次扫描间隔 (计数时钟频率 128KHz)</b>

### 33. 5. 2. 3 KEY\_SCANNER MODKEY0 寄存器 (KSCAN\_MODKEY0)

KEY_SCANNER MODKEY0 寄存器 (KSCAN_MODKEY0)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_0MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY0	Bit7-0	R/W	<b>Key_scanner 辅助按键 0 索引</b> 当识别的有效按键索引和辅助按键寄存器的索

			引相同时，会触发产生中断信号。
--	--	--	-----------------

### 33. 5. 2. 4 KEY\_SCANNER MODKEY1 寄存器 (KSCAN\_MODKEY1)

KEY_SCANNER MODKEY1 寄存器 (KSCAN_MODKEY1)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_1MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY1	Bit7-0	R/W	Key_scanner 辅助按键 1 索引

### 33. 5. 2. 5 KEY\_SCANNER MODKEY2 寄存器 (KSCAN\_MODKEY2)

KEY_SCANNER MODKEY2 寄存器 (KSCAN_MODKEY2)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_2MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY2	Bit7-0	R/W	Key_scanner 辅助按键 2 索引

### 33. 5. 2. 6 KEY\_SCANNER MODKEY3 寄存器 (KSCAN\_MODKEY3)

KEY_SCANNER MODKEY3 寄存器 (KSCAN_MODKEY3)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_3MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY3	Bit7-0	R/W	Key_scanner 辅助按键 3 索引

### 33. 5. 2. 7 KEY\_SCANNER MODKEY4 寄存器 (KSCAN\_MODKEY4)

KEY_SCANNER MODKEY4 寄存器 (KSCAN_MODKEY4)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_4MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY4	Bit7-0	R/W	Key_scanner 辅助按键 4 索引

### 33. 5. 2. 8 KEY\_SCANNER MODKEY5 寄存器 (KSCAN\_MODKEY5)

KEY_SCANNER MODKEY5 寄存器 (KSCAN_MODKEY5)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_5MODKEY							

Reserved	Bit31-8	—	保留
----------	---------	---	----

KS_MODKEY5	Bit7-0	R/W	Key_scanner 辅助按键 5 索引
------------	--------	-----	-----------------------

### 33. 5. 2. 9 KEY\_SCANNER MODKEY6 寄存器 (KSCAN\_MODKEY6)

KEY_SCANNER MODKEY6 寄存器 (KSCAN_MODKEY6)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_6MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY6	Bit7-0	R/W	Key_scanner 辅助按键 6 索引

### 33. 5. 2. 10 KEY\_SCANNER MODKEY7 寄存器 (KSCAN\_MODKEY7)

KEY_SCANNER MODKEY7 寄存器 (KSCAN_MODKEY7)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								KS_7MODKEY							

Reserved	Bit31-8	—	保留
KS_MODKEY7	Bit7-0	R/W	Key_scanner 辅助按键 7 索引

### 33. 5. 2. 11 KEY\_SCANNER FIFO控制寄存器 (KSCAN\_FIFOCON)

KEY_SCANNER FIFO 控制寄存器 (KSCAN_FIFOCON)																																	
偏移地址: 28 <sub>H</sub>																																	
复位值: 00000000_00000010_00000000_00000000 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved														FIFO_EMPTY		FIFO_FULL		FIFO_EVENT_NUM						FIFO_DOUT						FIFO_RD		FIFO_CLR	

Reserved	Bit31-18	—	保留
FIFO_EMPTY	Bit17	R	<b>FIFO 空指示</b> 0: FIFO 非空 1: FIFO 空
FIFO_FULL	Bit16	R	<b>FIFO 满指示</b> 0: FIFO 非满 1: FIFO 满 当 FIFO 满时, 新的事件会将之前存储的最后一个事件的值覆盖。
FIFO_EVENT_NUM	Bit15-11	R	<b>FIFO 中剩余事件个数</b>
FIFO_DOUT	Bit10-2	R	<b>FIFO 事件输出</b> Bit10: 事件类型 0: key_down 1: key_up Bit 9-2 : key index
FIFO_RD	Bit1	T_W1	<b>FIFO 读使能</b> 此 bit 写 1 触发读使能, 后自动清 0
FIFO_CLR	Bit0	T_W1	<b>FIFO 复位</b> 此 bit 写 1 触发清空 FIFO, 后自动清 0

### 33. 5. 2. 12 KEY\_SCANNER 中断使能寄存器 (KSCAN\_IER)

KEY_SCANNER 中断使能寄存器 (KSCAN_IER)
偏移地址: 2C <sub>H</sub>



复位值: 00000000\_00000000\_00000000\_00000000<sub>B</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									FIFO_FULL	GHOST_DETECTED	MODKEY_DETECTED	KEY_DETECTED	FIFO_OVERRUN	FIFO_UNDERRUN	

Reserved	Bit31-6	—	保留
FIFO_FULL	Bit5	R/W	<b>FIFO 满指示中断使能</b> 0: 中断关闭 1: 中断使能
GHOST_DETECTED	Bit4	R/W	<b>检测 ghost 事件中断使能</b> 0: 中断关闭 1: 中断使能
MODKEY_DETECTED	Bit3	R/W	<b>检测辅助按键中断使能</b> 0: 中断关闭 1: 中断使能
KEY_DETECTED	Bit2	R/W	<b>检测按键事件中断使能</b> 0: 中断关闭 1: 中断使能
FIFO_OVERRUN	Bit1	R/W	<b>FIFO 满写中断使能</b> 0: 中断关闭 1: 中断使能
FIFO_UNDERRUN	Bit0	R/W	<b>FIFO 空读中断使能</b> 0: 中断关闭 1: 中断使能

### 33. 5. 2. 13 KEY\_SCANNER中断标志寄存器 (KSCAN\_RIF)

KEY_SCANNER 中断标志寄存器 (KSCAN_RIF)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	FIFO_FULL	GHOST_DETECTED	MODKEY_DETECTED	KEY_DETECTED	FIFO_OVERRUN	FIFO_UNDERRUN
----------	-----------	----------------	-----------------	--------------	--------------	---------------

Reserved	Bit31-6	—	保留
FIFO_FULL	Bit5	R	<b>FIFO 满指示</b> 0: FIFO 非满 1: FIFO 满
GHOST_DETECTED	Bit4	R	<b>检测 ghost 事件指示</b> 0: 未检测到 ghost 事件 1: 检测到 ghost 事件
MODKEY_DETECTED	Bit3	R	<b>检测辅助按键指示</b> 0: 未检测到辅助按键 1: 检测到辅助按键
KEY_DETECTED	Bit2	R	<b>检测按键事件指示</b> 0: 未检测到按键事件 1: 检测到按键事件
FIFO_OVERRUN	Bit1	R	<b>FIFO 满写指示</b> 0: FIFO 未发生满写 1: FIFO 发生满写
FIFO_UNDERRUN	Bit0	R	<b>FIFO 空读指示</b> 0: FIFO 未发生空读 1: FIFO 发生空读

### 33. 5. 2. 14 KEY\_SCANNER中断上报寄存器 (KSCAN\_IFM)

KEY_SCANNER 中断寄存器 (KSCAN_IFM)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	FIFO_FULL	GHOST_DETECTED	MODKEY_DETECTED	KEY_DETECTED	FIFO_OVERRUN	FIFO_UNDERRUN
----------	-----------	----------------	-----------------	--------------	--------------	---------------

Reserved	Bit31-6	—	保留
FIFO_FULL	Bit5	R	<b>FIFO 满指示</b> 0: FIFO 非满 1: FIFO 满
GHOST_DETECTED	Bit4	R	<b>检测 ghost 事件指示</b> 0: 未检测到 ghost 事件 1: 检测到 ghost 事件
MODKEY_DETECTED	Bit3	R	<b>检测辅助按键指示</b> 0: 未检测到辅助按键 1: 检测到辅助按键
KEY_DETECTED	Bit2	R	<b>检测按键事件指示</b> 0: 未检测到按键事件 1: 检测到按键事件
FIFO_OVERRUN	Bit1	R	<b>FIFO 满写指示</b> 0: FIFO 未发生满写 1: FIFO 发生满写
FIFO_UNDERRUN	Bit0	R	<b>FIFO 空读指示</b> 0: FIFO 未发生空读 1: FIFO 发生空读

### 33. 5. 2. 15 KEY\_SCANNER中断清除寄存器 (KSCAN\_ICR)

KEY_SCANNER 中断寄存器 (KSCAN_ICR)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	FIFO_FULL	GHOST_DETECTED	MODKEY_DETECTED	KEY_DETECTED	FIFO_OVERRUN	FIFO_UNDERRUN
----------	-----------	----------------	-----------------	--------------	--------------	---------------

Reserved	Bit31-6	—	保留
FIFO_FULL	Bit5	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除 FIFO 满中断标志位
GHOST_DETECTED	Bit4	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除 ghost 中断标志位
MODKEY_DETECTED	Bit3	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除辅助按键中断标志位
KEY_DETECTED	Bit2	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除到按键中断标志位
FIFO_OVERRUN	Bit1	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除 FIFO 满写中断标志位
FIFO_UNDERRUN	Bit0	W1	清除中断标志寄存器 0: 无操作 1: 写 1 清除 FIFO 空读中断标志位

## 第34章 正交译码器(QDEC)

### 34.1 概述

正交译码器(QDEC)支持自动解析由光机械鼠标生成的两路正交信号或单路信号。QDEC 是一种运动增量译码器，通过记录来自光机械鼠标的计数脉冲信号来确定鼠标移动的方向和距离。

### 34.2 特性

- ◇ 支持在将正交信号解析为位移偏移量 8bit Delta Z (DZ)；
- ◇ 支持多种模式：机械鼠标 rambol 鼠标（单路正交信号）光电鼠标（正向正交信号）光电鼠标（反向正交信号）；
- ◇ 支持多种正交信号采样模式：双沿 上升沿 下降沿；
- ◇ 支持通过 GPIO 控制外部 LED/光感应器件；

### 34.3 结构框图

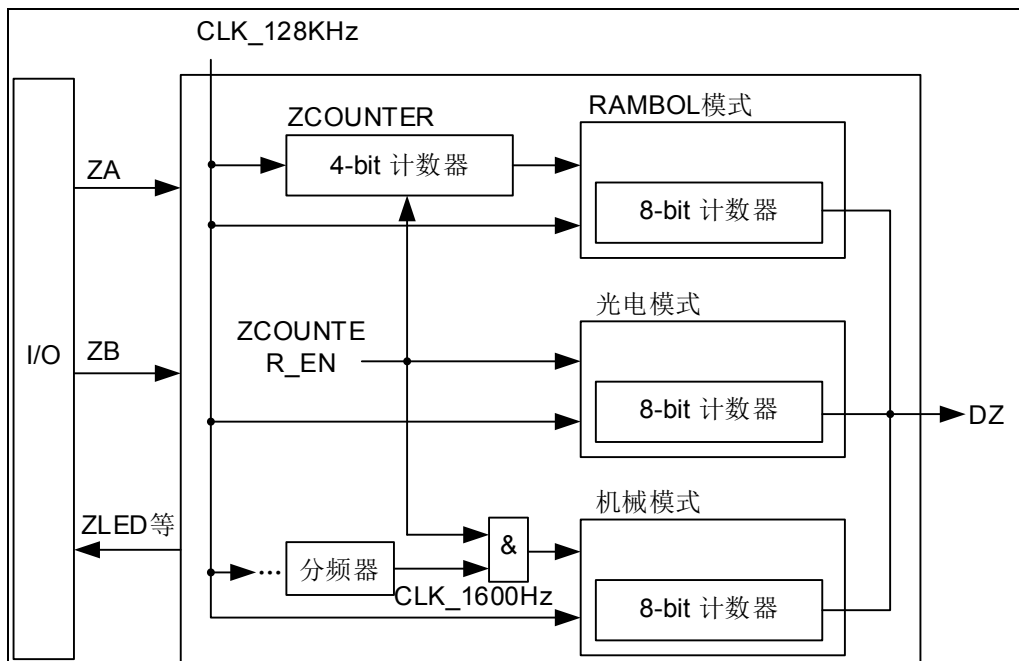


图 34-1 结构框图

### 34.4 功能描述

光电鼠标和机械鼠标在移动的过程中，内置的传感器会将其位移信息转换为两路周期性的正交脉冲信号：信号 ZA 和 ZB。ZA 和 ZB 同时产生，且相位总相差 90°。所以在这两路信号中，总有一路信号较另一路信号先发生变化，当 ZA 先于 ZB 变化时称为正向，当 ZB 先于 ZA 变化时称为反向。图 32-2 所示为鼠标正向和反向移动时 ZA 和 ZB 的脉冲波形及电平翻转情况，ZA 与 ZB 的波形相同但相位相差 1/4 个波形周期，当正向移动时 ZA 先 ZB 在 1/4

一个波形周期前发生变化，之后 ZB 总延迟 1/4 个波形周期跟随 ZA 变化，同理，当反向移动时 ZA 总延迟 1/4 个波形周期跟随 ZB 变化。当跟随信号出现上升沿时称采样边沿为前沿，当跟随信号出现下降沿时称采样边沿为后沿。

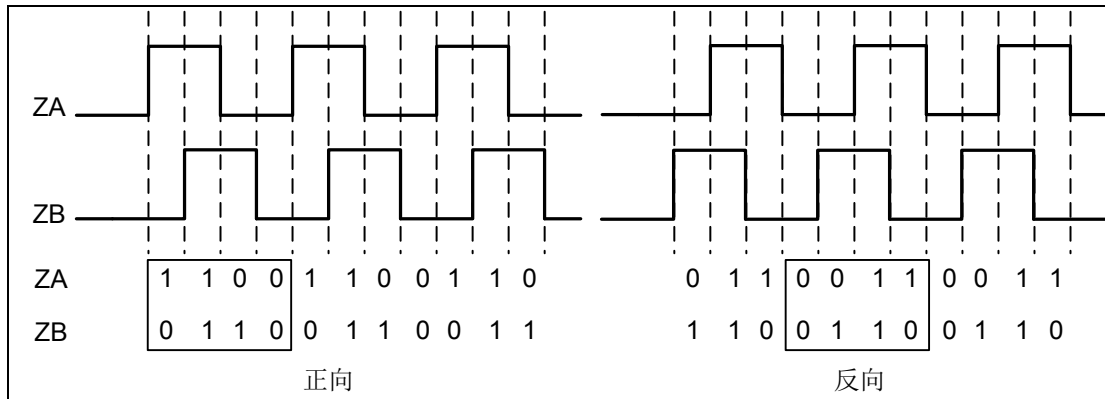


图 34-2 QDEC 采集的输入信号

变化前		变化后		状态描述	采样边沿
ZA	ZB	ZA	ZB		
0	0	0	0	未移动	-
0	0	0	1	反向移动+1	前沿
0	0	1	0	正向移动	-
0	1	0	0	正向移动+1	后沿
0	1	0	1	未移动	-
0	1	1	1	反向移动	-
1	0	0	0	反向移动	-
1	0	1	0	未移动	-
1	0	1	1	正向移动+1	前沿
1	1	0	1	正向移动	-
1	1	1	0	反向移动+1	后沿
1	1	1	1	未移动	-

图 34-3 采样译码

正交译码器，通过在多种模式下，检测正交信号的相位变化以及变化沿个数，解析出正向或是反相信号，通过 8bit 二进制有符号计数器来记录。当检测到正向时计数器值增加，当检测到反向时计数器值减小。计数器的正数表示正向，负数表示反向，计数范围为-128~+127，计数器的绝对值在同等采样率下，可以表示移动的速度和距离。

累计的变化沿个数存储在 DZ 计数器中，系统通过定时读取 DZ 计数器获得鼠标移动数据，读取后 DZ 计数器会清 0。

### 34.4.1 机械模式

通过配置 QDEC\_CTRL0 寄存器的 CFG 控制位可以使 QDEC 工作在机械模式，机械模式

的工作时钟频率为 1.6KHz。

机械模式不支持后沿采样。

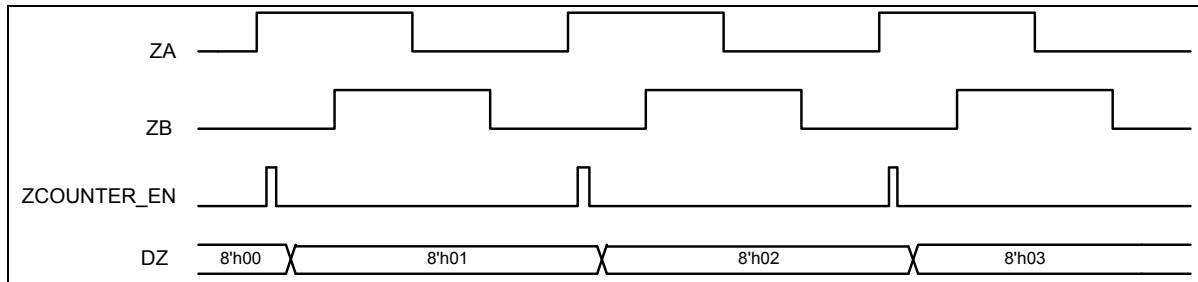


图 34-4 前沿采样模式（正向移动）

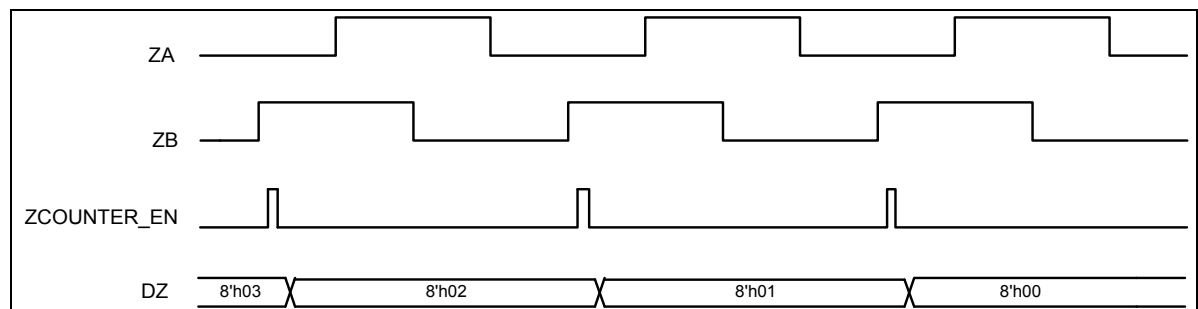


图 34-5 前沿采样模式（反向移动）

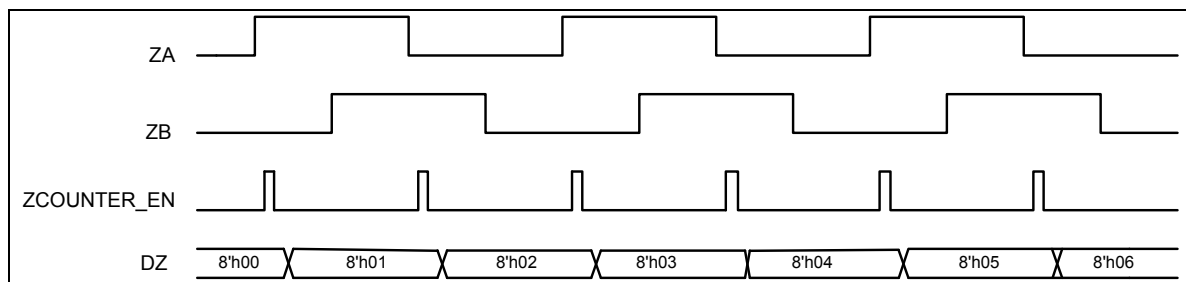


图 34-6 双沿采样模式（正向移动）

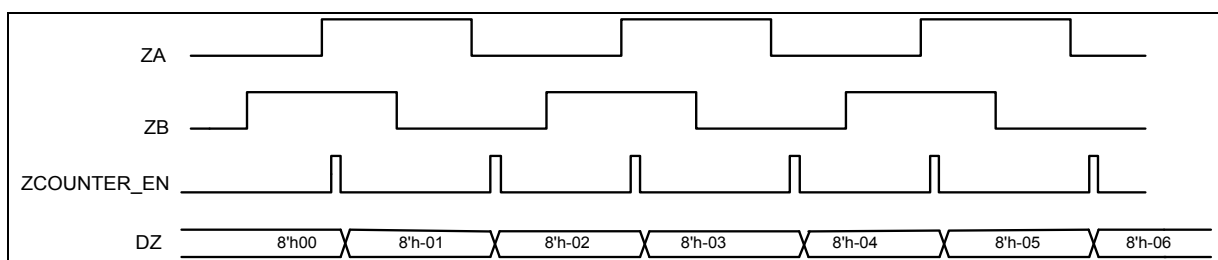


图 34-7 双沿采样模式（反向移动）

### 34.4.2 光电模式

通过配置 QDEC\_CTRL0 寄存器的 CFG 控制位可以使 QDEC 工作在光电模式。相比于另两种模式，光电模式只需 ZCOUNTER\_EN=1 即可对输入信号进行采样，通过配置

SAMP\_MODE 控制位可以选择采样边沿。ZCOUNTER\_EN 脉冲信号由硬件电路产生，通过读取 QDEC\_RD\_ZCNT 寄存器的 ZCOUNTER\_EN 状态位可查看其电平状态。

QDEC 通过比较当前读取的正交信号及上一次读取的正交信号电平关系，确定信号变化的方向为正向或反向，当 ZCOUNTER\_EN 使能后，QDEC 在设定的边沿处决定 DZ 计数器是增大、减小还是不变。DZ 计数器每次处理单位 1 的数据，符合表 32-1 的运算规律，当变化前后的数据出现表以外的情况时，DZ 保持不变。

当 DZ 增加到 127 后检测到处理为正向+1 时，DZ 保持 127 不变。当 DZ 降低到-128 后检测到处理状态为反向+1 时，DZ 保持-128 不变。

直接利用采样得到的信号进行信号变化方向判断的模式为正向正交信号模式，利用采样得到信号的反值来判断信号变化方向的模式为反向正交信号模式，即对输入的 ZA、ZB 信号做取反操作然后判断移动方向。

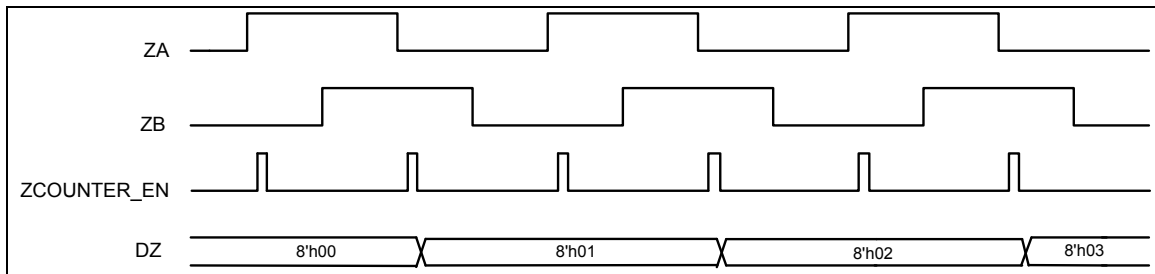


图 34-8 后沿采样模式（正向移动）

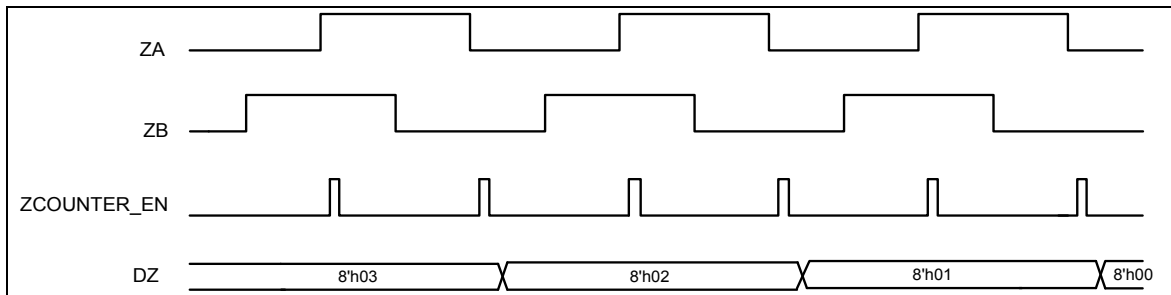


图 34-9 后沿采样模式（反向移动）

前沿采样模式和双沿采样模式同机械模式。

### 34.4.3 RAMBOL 模式

RAMBOL 为单路输入模式，只对输入的 ZA 信号进行采样，继而判断正向还是反向。Rambol 模式的采样周期固定为 281.25us（ZLED 输出周期表示采样周期），在一个采用周期内会对 ZA 信号进行两次采样，当两次采样的信号为“01”时为正向移动，当两次采样的信号为“10”时为反向移动，根据移动的方向改变 ZCOUNTER 的计数值。

通过配置 QDEC\_CTRL0 寄存器的 CFG 控制位可以使 QDEC 工作在 RAMBOL 模式。RAMBOL 模式下 DZ 的计数使能信号由一个 4bit 有符号计数器 ZCOUNTER 决定，当 ZCOUNTER 计数达到上限（4'b0100），如果此时 DZ 处理状态为正向+1 则 DZ+1；当



ZCOUNTER 计数达到下限 (4'b1100)，如果此时 DZ 处理状态为反向+1 则 DZ-1。其他处理过程同光电模式相同。

ZCOUNTER 的计数使能信号为 ZCOUNTER\_EN，增减控制信号为输入正交信号的方向信号，这同光电模式下的 DZ 计数器是一样的。当 ZCOUNTER 计数达到上限后硬件会自动赋值 4'b1101；当 ZCOUNTER 计数达到下限后硬件会自动赋值 4'b0011。

RAMBOL 模式支持 64ms 滤波功能，从正向移动变为反向移动或者从反向移动变为正向移动时，当维持变换方向一方的时间大于 64ms 时，DZ 的计数值才进行“+1”和“-1”计数，否则 DZ 计数值保持不变。例如当从反向移动方向变为正向移动方向，假设此时 DZ 的计数值为 8'h09，对 ZA 信号进行周期性的采样，ZCOUNTER 在-3~4 之间进行周期性计数，当 ZA 维持正向移动方向的时间小于 64ms，ZCOUNTER 计数到 4'b0100 时，DZ 保持当前值 8'h09 不变，直到 ZA 维持正向移动方向的时间大于 64ms 后，每当 ZCOUNTER 计数到 4'b0100 时，DZ 自动加 1；

ZCOUNTER 计数值可以通过读取 QDEC\_RD\_ZCNT 寄存器的 ZCOUNTER 状态位查看。超过 1s 没有检测到鼠标移动，ZCOUNTER 会自动清 0。

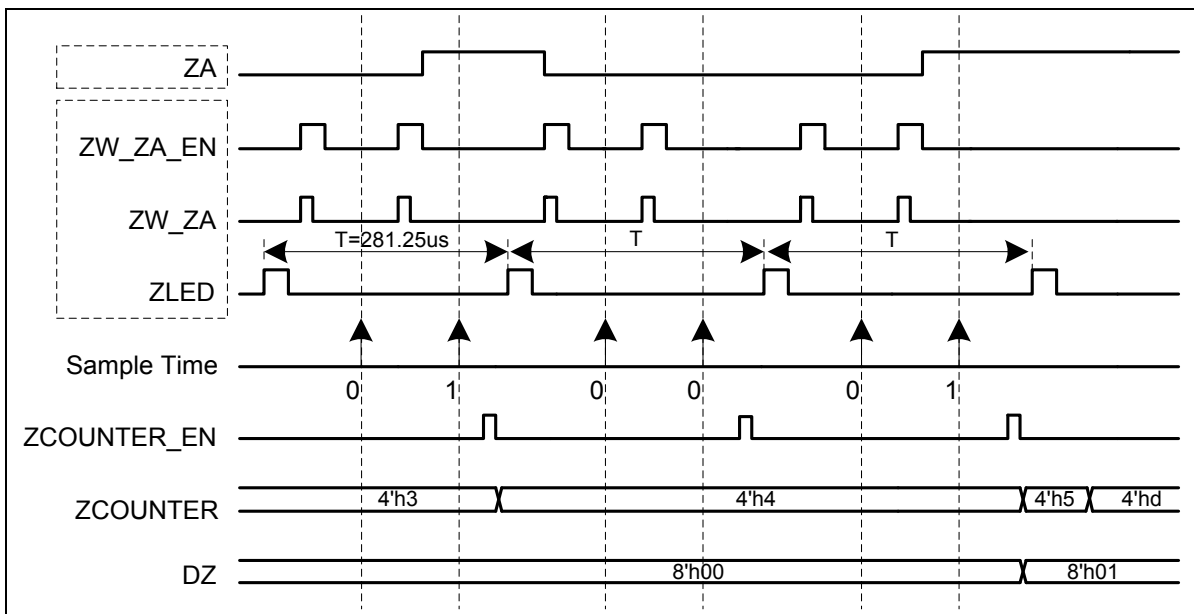


图 34-10 Rambol 正向采样示意图

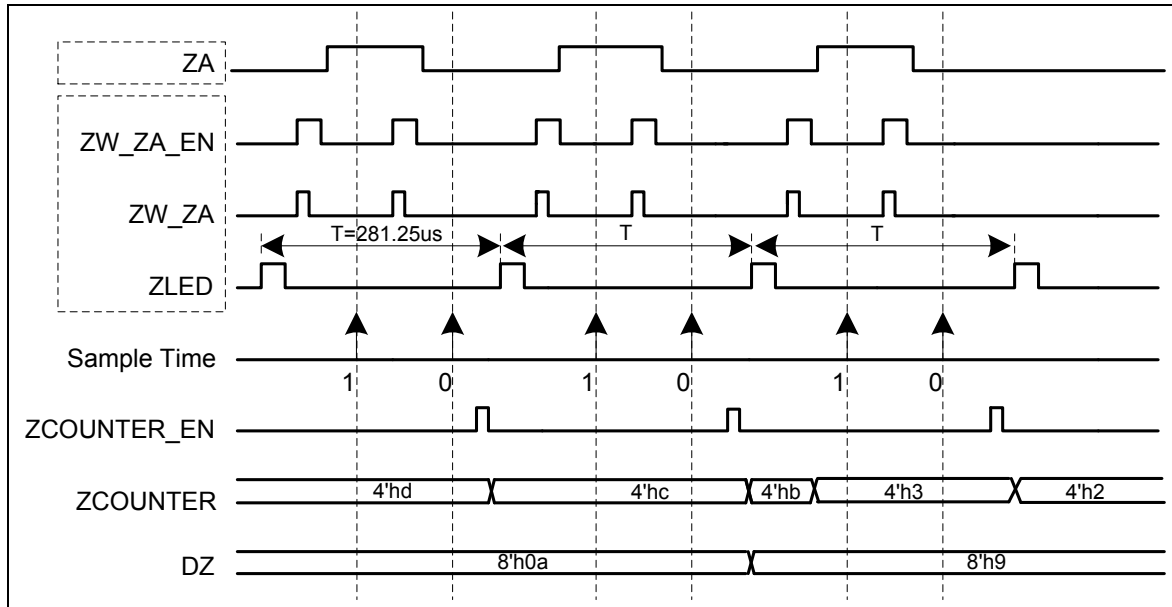


图 34-11 Rambol 反向采样示意图

### 34.4.4 Debounce 滤波

两路正交脉冲输入信号 ZA、ZB 支持数字滤波。其中机械模式的采样时钟固定为 1.6KHz，光电模式和 RAMBOL 模式采样时钟固定为 128KHz。外部输入信号稳定时间必须大于三个采样时钟周期，否则信号被滤除。正交输入信号 ZA、ZB 输入到模块内部会被延时四个采样时钟周期。

### 34.4.5 输入输出管脚配置

外部鼠标设备产生的正交信号 ZA 和 ZB 通过 GPIO 传送至 QDEC。ZA 信号复用 GPIO\_PB14 端口，ZB 信号复用 GPIO\_PB15 端口，且 FUNCTION=4。

在光电模式下 QDEC 的输出信号和工作状态可以通过 GPIO 送出。ZW\_ZA 和 ZW\_ZB 表示输出驱动状态，当选择输出驱动为高电平时 ZW\_ZA 和 ZW\_ZB 为 1，否则 ZW\_ZA 和 ZW\_ZB 为 0。ZW\_ZA\_EN 和 ZW\_ZB\_EN 表示输出使能状态，当选择输出使能时 ZW\_ZA\_EN 和 ZW\_ZB\_EN 为 1，否则 ZW\_ZA\_EN 和 ZW\_ZB\_EN 为 0。当 GPIO 选择 FUNCTION=4 时，GPIO\_PB9~GPIO\_PB13 依次输出 ZW\_ZA、ZW\_ZA\_EN、ZW\_ZB、ZW\_ZB\_EN 和 ZLED，其中 ZLED、ZW\_ZA\_EN 和 ZW\_ZB\_EN 的输出周期固定为 281.25us。

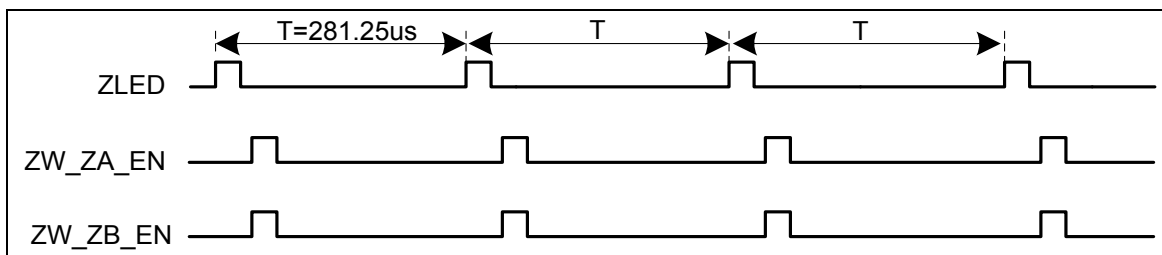


图 34-12 光电模式管脚输出示意图

RAMBOL 模式下支持 ZW\_ZA、ZW\_ZA\_EN、ZLED 状态通过 GPIO 的输出，具体的输出波形图见 RAMBOL 模式章节的相关描述。

机械模式下不支持 ZW\_ZA、ZW\_ZB、ZLED 的输出控制；

## 34.5 特殊功能寄存器

### 34.5.1 寄存器列表

QDEC 控制寄存器列表			
名称	偏移地址	类型	描述
QDEC 基地址: 4000_F400 <sub>H</sub>			
QDEC_CTRL0	0000 <sub>H</sub>	R/W	QDEC 控制寄存器 0
QDEC_RD_ZCNT	0020 <sub>H</sub>	R	QDEC 计数值输出寄存器

### 34.5.2 寄存器描述

#### 34.5.2.1 QDEC控制寄存器 0 (QDEC\_CTRL0)

QDEC 控制寄存器 0 (QDEC_CTRL0)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SAMP_MODE		DRV_POL_DIS	DRV_POL	CFG			

Reserved	Bit31-6	—	保留
SAMP_MODE	Bit5-4	R/W	<b>光电模式:</b> 00: 双沿采样 01: 后沿采样 10: 前沿采样 11: 非法值 <b>机械模式:</b> 0x: 双沿采样 1x: 前沿采样 <b>RAMBOL 模式:</b> 无效
DRV_POL_DIS	Bit3	R/W	<b>输出使能关闭使能</b> 0: 输出使能打开 1: 输出使能关闭
DRV_POL	Bit2	R/W	<b>输出驱动选择</b> 0: 输出低电平 1: 输出高电平
CFG	Bit1-0	R/W	<b>模式选择 0-3</b> 2'b00: 机械模式 2'b01: 光电模式 (正向正交信号) 2'b10: RAMBOL 模式 2'b11: 光电模式 (反向正交信号)

### 34. 5. 2. 2 QDEC计数值输出寄存器 (QDEC\_RD\_ZCNT)

QDEC 计数值输出寄存器 (QDEC_RD_ZCNT)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DZ										ZCOUNTER_EN	ZCOUNTER				

Reserved	Bit31-13	—	保留
DZ	Bit12-5	R	<b>QDEC DZ 计数器 -128 ~ +127</b> 注: 两种清零情况 1、超过 1s 中没有检测到鼠标移动, 会硬件自动清 0 2、软件读取此计数器后也会清 0
ZCOUNTER_EN	Bit4	R	<b>ZCOUNTER 计数使能</b> 0: ZCOUNTER 计数停止 1: ZCOUNTER 计数使能
ZCOUNTER	Bit3-0	R	<b>QDEC COUNTER</b> 注: 超过 1s 中没有检测到鼠标移动, 会自动清 0

## 第35章 触控按键 (TK)

### 35.1 概述

- ◆ 支持最多 8 个触摸按键，一路按键补偿通道
- ◆ 支持 4 种工作频率可配置
- ◆ 支持可选基准电压
- ◆ 支持触摸按键扫描中断
- ◆ 支持 1 个模拟比较器 ACP，负端参考电压可选

### 35.2 内部结构图

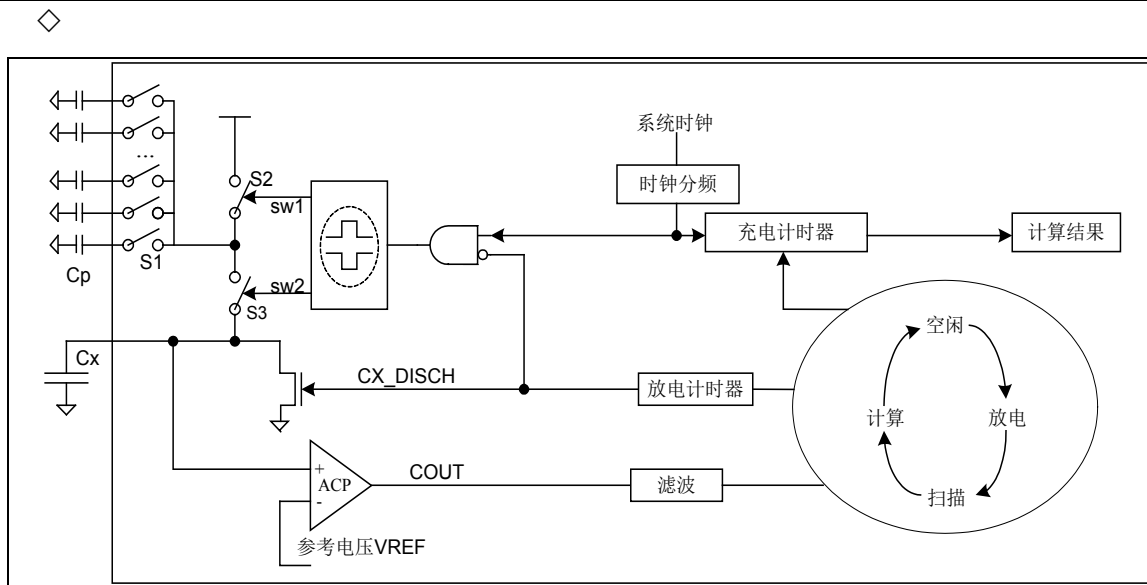


图 35-1 触控按键内部结构示意图

注 1: 由于 Touchkey 模块的基准电压和电流是由 ADC 模块相关的 VREF 提供，所以在使用 Touchkey 模块时需要先将 ADC 模块相关的 VREF 使能，即将 ADC\_CCR1->VREFEN 配置为 1;

### 35.3 触控按键扫描工作原理

本芯片实现一种电容式感应按键检测，电容式感应按键检测方法采用了“电容电荷转移”的工作原理。

在上图中，开关 S1 由按键选通寄存器控制；开关 S2 由脉宽调制后的系统时钟分频信号控制；开关 S3 由脉宽调制后的系统时钟分频信号的互补信号控制。当有手指触摸按键时，相当于增加了一个到地的电容 Cf，实际电容变成 Cp+Cf，相对于未触摸时，电容 Cx 充电的时间将会变短，如下面的触摸状态示意图所示。根据电容 Cx 充电时间的长短，来实现触摸按键的扫描。具体的工作原理如下：

- 1、S1，S2 闭合，S3 断开，系统开始对电容 Cp 充电。
- 2、S1，S3 闭合，S2 断开，电容 Cp 对电容 Cx 放电。

- 3、循环往复 1、2 步骤给电容  $C_x$  充电，直到电容  $C_x$  上的电压大于比较器 ACP 负端的参考电压  $V_{REF}$  时，模拟比较器 COUT 输出高电平，滤波器开始采样处理。
- 4、滤波器采样处理后，放电计时器输出高电平，并驱动 N 管导通，此时 S2, S3 断开，电容  $C_x$  开始放电。（为了保证电容  $C_x$  放电完全，可增加放电时间，即用户可根据电容  $C_x$  的容量大小，软件配置不同的放电时间。）
- 5、滤波器输出为低电平并且电容  $C_x$  放电结束时，充电计时器将开始统计电容  $C_x$  充电的时间。每统计一次电容  $C_x$  充电的时间即为一次采样结果，软件可配置多次采样，硬件会自动计算多次采样的平均值。
- 6、当滤波器输出高电平时，充电计时器把统计的数值送到运算放大器放大，以便增加当前状态值的噪声容限。同时放电计时器开始电容  $C_x$  放电的计时。如果没有错误启动和计算溢出，硬件会自动产生运算结束中断，并加载 24 位的采样值，从而完成一次按键扫描。

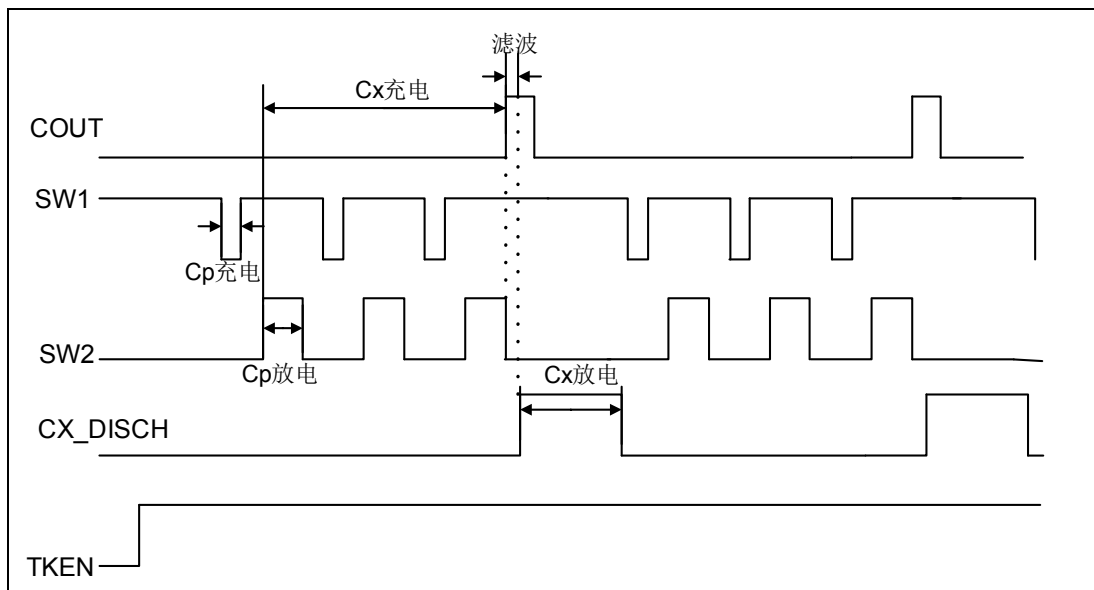


图 35-2 无触摸状态示意图

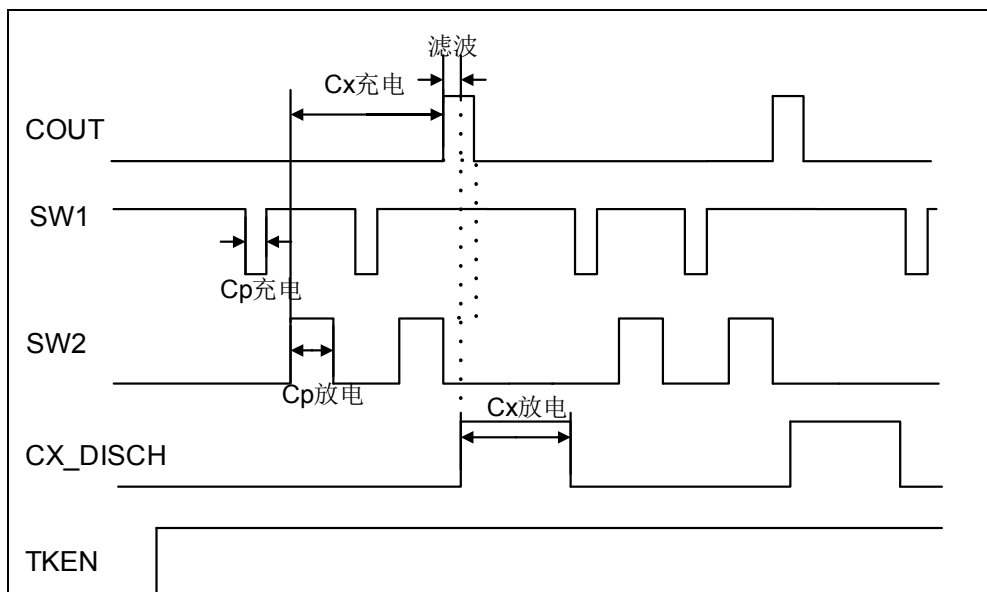


图 35-3 有触摸状态示意图



### 35.4 触控按键工作流程

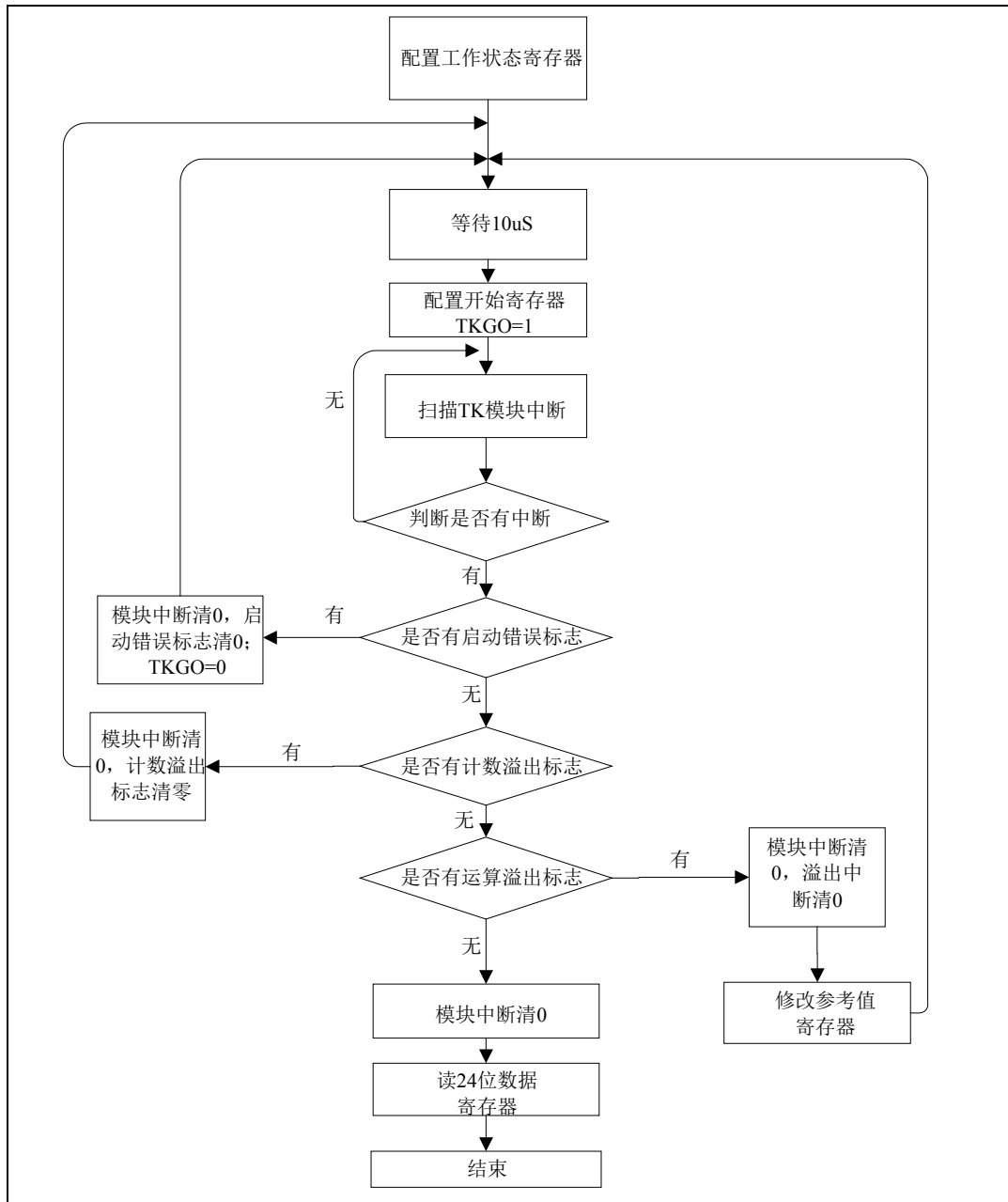


图 35-4 触摸按键操作流程

注 1: 当启动错误标志 TKERR 有效, 发生扫描启动错误时, 可尝试将 TKGO 清零, 设置 TKDIST 寄存器将放电时间加大, 然后重新将 TKGO 置 1 启动扫描。若即使将放电时间加至最大, 仍然发生扫描启动错误, 则可能所使用 Cx 电容过大, 可考虑减小该电容值。亦可如上图操作流程所示, 不修改放电时间, 通过多次启动, 完成对 Cx 电容的放电。用户可软件设定重复次数, 达到某一设定次数后, 若仍然产生 TKERR 错误, 需检查 Cx 电容等系统设计问题。

## 35.5 特殊功能寄存器

### 35.5.1 寄存器列表

TK 控制寄存器列表			
名称	偏移地址	类型	描述
TK 基地址: 4000_8C00 <sub>H</sub>			
TK_SEL	0000 <sub>H</sub>	R/W	触控按键功能选择寄存器
TK_TUN	0004 <sub>H</sub>	R/W	触控调整寄存器
TK_CTL	0008 <sub>H</sub>	R/W	触控控制寄存器
TK_DA	000C <sub>H</sub>	R	扫描结果寄存器
TK_MOD	0010 <sub>H</sub>	R/W	放大系数寄存器
TK_VRC	0014 <sub>H</sub>	R/W	内部参考电压控制寄存器
TK_OUTS0	0018 <sub>H</sub>	R/W	TK 开关电平输出控制寄存器
-	001C <sub>H</sub>	-	-
TK_INT	0020 <sub>H</sub>	R/W	TK 中断寄存器
TK_SCT	0024 <sub>H</sub>	R	TK 充放电扫描计数值寄存器

### 35.5.2 寄存器描述

#### 35.5.2.1 触控按键功能选择寄存器 (TK\_SEL)

触控按键功能选择寄存器 (TK_SEL)																															
偏移地址: 000 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							TKDUS		TKTS		Reserved		TKCHS		

—	bit31-17	—	—
TKCHCPEN	Bit16	R/W	<b>补偿通道使能位</b> 0: 禁止 1: 使能
TKCH7EN	Bit15	R/W	<b>按键通道 7 使能位</b> 0: 禁止 1: 使能
TKCH6EN	Bit14	R/W	<b>按键通道 6 使能位</b> 0: 禁止 1: 使能
TKCH5EN	Bit13	R/W	<b>按键通道 5 使能位</b> 0: 禁止 1: 使能
TKCH4EN	Bit12	R/W	<b>按键通道 4 使能位</b> 0: 禁止 1: 使能
TKCH3EN	Bit11	R/W	<b>按键通道 3 使能位</b> 0: 禁止 1: 使能
TKCH2EN	Bit10	R/W	<b>按键通道 2 使能位</b> 0: 禁止 1: 使能
TKCH1EN	bit9	R/W	<b>按键通道 1 使能位</b> 0: 禁止 1: 使能
TKCH0EN	bit8	R/W	<b>按键通道 0 使能位</b> 0: 禁止 1: 使能
TKDUS	bit7	R/W	<b>模拟充放电开关占空比控制位</b> 0: 1/4 占空比 1: 1/2 占空比

TKFS<1:0>	bit6-5	R/W	控制电路工作频率选择位 00: Fsys/8 01: Fsys/16 10: Fsys/32 11: Fsys/64
—	bit4	—	—
—	bit3-0	—	—

注 1: TK0~TK7 对应 PA2~PA8、PA11

### 35.5.2.2 触控调整寄存器 (TK\_TUN)

触控调整寄存器 (TK_TUN)																															
偏移地址: 004 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TKTMS		TKCFT		TKDIST		TKAVS	

—	bit31-8	—	—
TKTMS<1:0>	bit 7~6	R/W	采样次数选择位 00: 采样 1 次, 结果为 1 次采样的平均值 01: 采样 4 次, 结果为 4 次采样的平均值 10: 采样 8 次, 结果为 8 次采样的平均值 11: 采样 16 次, 结果为 16 次采样的平均值
TKCFT<1:0>	bit5-4	R/W	比较器输出滤波时间选择位 00: 无滤波 01: 2*Tdiv (Tdiv 为系统时钟分频后时钟) 10: 4*Tdiv 11: 8*Tdiv
TKDIST<2:0>	bit3-1	R/W	Cx 放电时间选择位 000: 32*Tdiv (Tdiv 为系统时钟分频后时钟) 001: 64*Tdiv 010: 128*Tdiv 011: 256*Tdiv 100: 384*Tdiv 101: 512*Tdiv 110: 1024*Tdiv 111: 2048*Tdiv
TKAVS	bit0	R/W	TK 通道选择 AMUX 供电电源配置位 0: 选择 VDD 电源 (默认值) 1: 选择内部增压泵电源

35.5.2.3 触控控制寄存器 (TK\_CTL)

触控控制寄存器 (TK_CTL)																																																	
偏移地址: 008 <sub>H</sub>																																																	
复位值: 00000000_00010000_00000001_00000000 <sub>B</sub>																																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
Reserved								CMPF_VCNT								Reserved								CMPF_TH								Reserved								TK_OE_VREF	TK_CHARGE_SEL	TKCHGC	TKCHGS	TKCTR	SCANOV	TKERR	TKOV	TKEN	TKGO

—	bit31-29	—	—
CMPF_VCNT	bit28-24	R	<p><b>比较器滤波输出信号有效次数</b></p> <p>1) 当 TKERR 为 1 时, 不用关心此控制位;</p> <p>2) 当 TKERR 为 0 时, 此控制位为 0, 表示模拟比较器滤波输出信号无效, 此次触控无效。</p> <p>3) 当 TKERR 为 0 时, 此控制位为非 0, 表示模拟比较器滤波输出信号有效, 值表示产生的有效次数。</p> <p>TKTMS=2'b00: 理论值应该为 5'h01;</p> <p>TKTMS=2'b01: 理论值应该为 5'h04;</p> <p>TKTMS=2'b10: 理论值应该为 5'h08;</p> <p>TKTMS=2'b11: 理论值应该为 5'h10;</p>
—	bit23-22	—	—
CMPF_TH	bit21-16	R/W	<p><b>比较器滤波输出信号是否有效阈值设置</b></p> <p>6'h00: 1* Tdiv (Tdiv 为系统时钟分频后时钟)</p> <p>6'h01: 1* Tdiv</p> <p>6'h02: 2* Tdiv</p> <p>.....</p> <p>6'hFF: 63* Tdiv</p> <p>注: 模拟比较器 COUT 输出高电平, 滤波器采样输出后, COUT 输出维持高电平的时间多久认为有效, 可以通过此控制位设置。当 COUT 输出高电平的时间大于 CMPF_TH 设置值, 认为比较器 COUT 输出有效, 同时设置 CMPF_VCNT 标志位; 主要用于避免系统受到外部干扰, 误产生 COUT 输出信号;</p> <p>建议设置此值小于等于 6'h1c, 如果设置值大于 6'h1c, 那么在中断程序中读取 CMPF_VCNT 之前需要延时一段时间, 因为扫描完成后进行运算用时内部电路固定为 28*Tdiv, 运算完成后会产生扫描完成中断;</p>
—	bit15-10	—	—

TK_OE_VREF	bit9	R/W	充电电压 1.7V 和比较基准电压测试输出使能控制位（仅用于测试模式） 0: 输出关闭 1: 输出使能
TK_CHARGE_SEL	bit8	R/W	充电电压选择位 0: VDD33V 电源供电 1: 内部基准电压 1.7v 供电
TKCHGC	bit7	R/W	充电补偿使能位（禁用） 0: 禁止 1: 使能
TKCHGS	bit6	R/W	TK 充电增强控制位 0: 充电电流不增强 1: 充电电流增强
TKCTR	bit5	R/W	未扫描按键输出使能位 0: 禁止 1: 使能
SCANOV	bit4	R	扫描计数器溢出标志位 0: 未发生扫描计数器溢出 1: 发生扫描计数器溢出
TKERR	bit3	R	扫描启动错误标志位 0: 未发生扫描启动错误 1: 发生扫描启动错误
TKOV	bit2	R	运算结果溢出标志位 0: 本次运算结果未溢出 1: 本次运算结果溢出
TKEN	bit1	R/W	触摸按键使能控制位 0: 关闭触摸按键模块 1: 使能触摸按键模块
TKGO	bit0	R/W	按键启动扫描控制位 0: 未启动按键扫描 1: 按键扫描进行中

### 35.5.2.4 扫描结果寄存器 (TK\_DA)

扫描结果寄存器 (TK_DA)																															
偏移地址: 00C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TKDA															

—	bit31-24	—	—
---	----------	---	---

TKDA<23:0>	bit23-0	R	扫描结果 扫描结果为：放大系数/TK 充放电的计数值 注：其中 TK 充放电计数值为寄存器 TK_SCT
------------	---------	---	--

### 35.5.2.5 放大系数寄存器 (TK\_MOD)

放大系数寄存器 (TK_MOD)																															
偏移地址：010 <sub>H</sub>																															
复位值：00001111_11111111_11111111_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TKMOD															

—	bit31-28	—	—
TKMOD	bit27-0	R/W	放大系数

### 35.5.2.6 内部参考电压控制寄存器 (TK\_VRC)

内部参考电压控制寄存器 (TK_VRC)																															
偏移地址：014 <sub>H</sub>																															
复位值：00000000_00000000_00000000_01100001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								VRCS		Reserved	HPEN	HYSEN			

-	bit31-7	-	-
VRCS<2:0>	bit6-4	R/W	内部参考电压选择位 000: 0.3V 001: 0.6V 010: 0.8V 011: 1.0V 100: 1.2V 101: 1.3V 110: 1.5V 111: 1.7V
-	bit3-2	-	-
HPEN	bit1	R/W	比较器功耗使能位 0: 低功耗 1: 高功耗
HYSEN	bit0	R/W	比较器迟滞使能

			0: 禁止 1: 使能
--	--	--	----------------

### 35.5.2.7 TK开关电平输出控制寄存器 (TK\_OUTS0)

TK 开关电平输出控制寄存器 (TK_OUTS0)																															
偏移地址: 018 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																TKOUT7		TKOUT6		TKOUT5		TKOUT4		TKOUT3		TKOUT2		TKOUT1		TKOUT0	

-	bit31-16	-	-
TKOUT7<1:0>	bit15-14	R/W	TK7 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT6<1:0>	bit13-12	R/W	TK6 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT5<1:0>	bit11-10	R/W	TK5 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT4<1:0>	bit9-8	R/W	TK4 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT3<1:0>	bit7-6	R/W	TK3 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT2<1:0>	bit5-4	R/W	TK2 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号



			11: 未扫描按键输出为 SW1 的反向信号
TKOUT1<1:0>	bit3-2	R/W	TK1 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号
TKOUT0<1:0>	bit1-0	R/W	TK0 开关电平输出控制位 00: 未扫描按键输出低电平 01: 未扫描按键输出高电平 10: 未扫描按键输出为 SW1 信号 11: 未扫描按键输出为 SW1 的反向信号

### 35.5.2.8 TK中断寄存器 (TK\_INT)

TK 中断寄存器 (TK_INT)																															
偏移地址: 020 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												TKIE	TKIF		

—	bit31-2	—	—
TKIE	bit1	R/W	<b>TK 中断使能位</b> 0: 禁止 1: 使能
TKIF	bit0	R/C_W1	<b>TK 中断标志位</b> 0: 未发生中断 1: 发生按键扫描中断 (由硬件置 1, 软件写 1 清除, 软件写 0 无效)

### 35.5.2.9 TK充放电扫描计数值寄存器 (TK\_SCT)

TK 充放电扫描计数值寄存器 (TK_SCT)																															
偏移地址: 024 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SCCNT															

—	bit31-20	—	—
SCCNT	bit19-0	R	<p><b>TK 充放电扫描计数值</b></p> <p>1) 当 TKTMS=2'b00 时, 表示采样 1 次的计数值;</p> <p>2) 当 TKTMS=2'b01 时, 表示采样 4 次的累加计数值;</p> <p>3) 当 TKTMS=2'b10 时, 表示采样 8 次的累加计数值;</p> <p>4) 当 TKTMS=2'b11 时, 表示采样 16 次的累加计数值;</p>

## 第36章 调试配置控制 (DBGC)

### 36.1 概述

ES32W3120 使用的的内核是 Cortex™-M3，该内核包含用于高级调试功能的硬件。利用这些调试功能，可以在取指（指令断点）或取访问数据（数据断点）时停止内核。内核停止时，可以查询内核的内部状态和系统的外部状态。查询完成后，将恢复内核和系统并恢复程序执行。

当调试器与 MCU 相连并进行调试时，将使用内核的硬件调试模块。

MCU 提供串行 SWD 调试接口。

参考文档：

ARM\_debug\_interface\_v5.pdf (ADIV5 Architecture specification)

ARM\_debug\_interface\_v5\_supplement.pdf (ADIV5.1 Spec Supplement)

DDI0419C\_arm\_architecture\_v6m\_reference\_manual.pdf (ARMv6 Architecture)

### 36.2 特性

- ◇ 支持 SW-DP：调试端口电路，实现 DAP 电路和外部调试主机的通讯
- ◇ 支持 MEM-AP：访问端口电路，实现 DAP 电路与被调试单元的通讯
- ◇ 支持断点 (Breakpoint)：4 个断点
- ◇ 支持数据观测和追踪 (DWT)：2 个数据观测点

### 36.3 结构框图

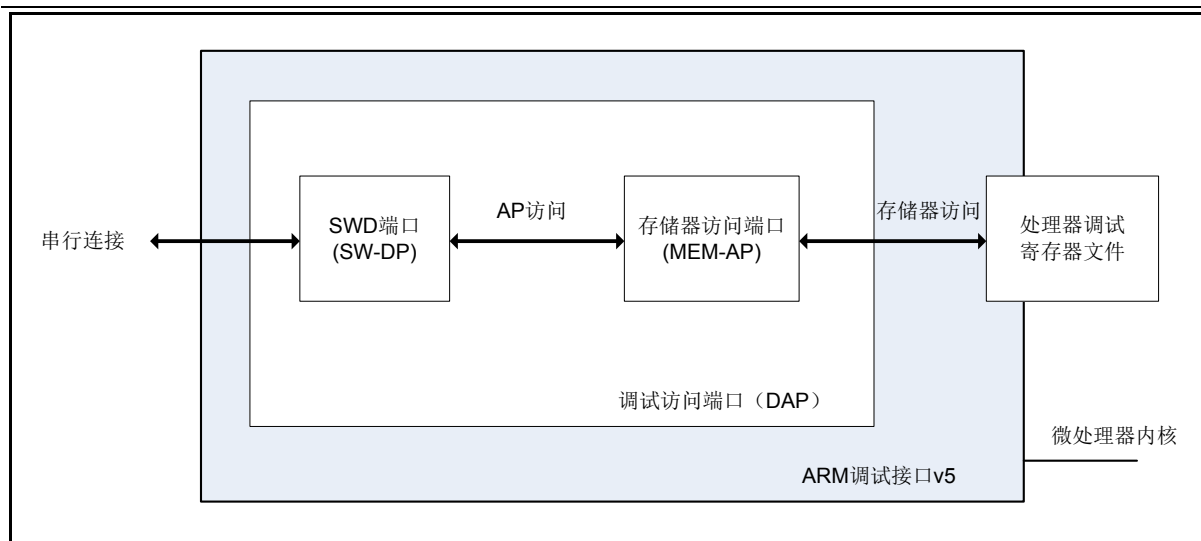


图 36-1 电路结构框图

## 36.4 功能描述

### 36.4.1 调试端口

下表为 SWD 调试用端口，芯片上电后默认作为调试功能使用。

端口功能	输入/输出	说明
SWCLK	输入	调试时钟。该端口在调试模式下为调试电路提供通信时钟。内部默认上拉。
SWDIO	输入/输出	调试数据输入/输出端口。用于 SW-DP 与外部调试主机的数据交互。内部默认为上拉。

图 36-2 SWD 端口描述

### 36.4.2 调试冻结

在程序最终完成之前，我们会经历反复调试，通过调试工具对运行程序进行暂停时，实际上我们不仅仅需要暂停内核，很多硬件外设仍然在工作（如：定时器、RTC、看门狗等），有时候我们需要在暂停内核时，同时要求硬件也进行暂停，通过 DBGMC，我们能够达到要求。

操作示例：

1. 按照需求正确初始化使用的外设，如定时器 TIM1
2. 为保证程序在调试过程中暂停时 TIM1 计数同时暂停，设置 `DBG_APB1FZ.TIM1_STOP = 1`
3. 如果未设置 `DBG_APB1FZ.TIM1_STOP = 1`，程序在调试过程中暂停时，实际 TIM 仍然在计数。

### 36.4.3 调试复位

内核调试电路和调试控制寄存器只可被上电复位、欠压复位及软件复位中的芯片全局复位（`RMU_AHB2RSTR.CHIPRST`）所复位。

### 36.4.4 MEM-AP访问端口

MEM-AP 端口，用于访问被调试单元的存储器映射区域。

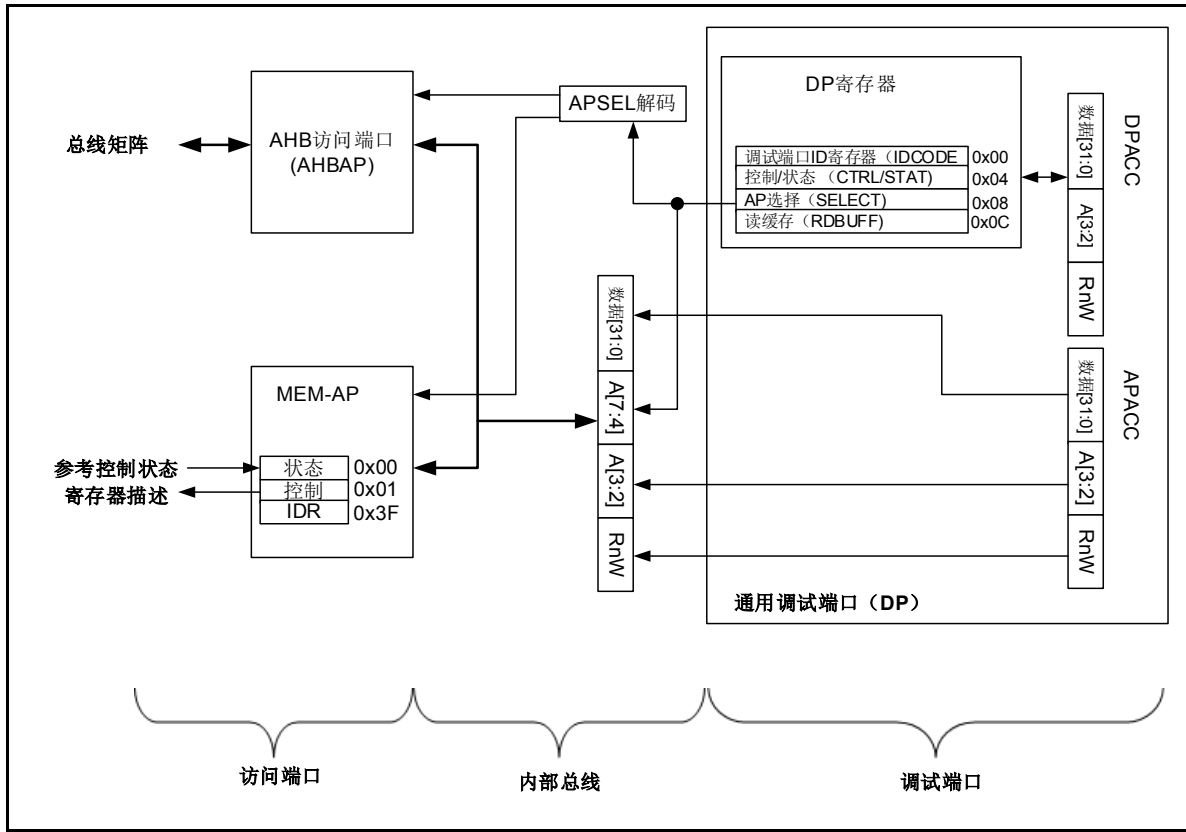


图 36-3 MEM-AP 地址映射

## 36.5 特殊功能寄存器

### 36.5.1 寄存器列表

DBGC 寄存器列表			
名称	偏移地址	类型	描述
DBGC 基地址: 4004_A000 <sub>H</sub>			
DBGC_IDCODE	000 <sub>H</sub>	R	DBG 器件识别码寄存器
DBGC_CR	004 <sub>H</sub>	RW	DBG 控制寄存器
DBGC_APB1FZ	008 <sub>H</sub>	RW	APB1 外设调试冻结寄存器
DBGC_APB2FZ	00C <sub>H</sub>	RW	APB1 外设调试冻结寄存器
DBGC_TIMOFZ	010 <sub>H</sub>	RW	定时器输出调试冻结寄存器

### 36.5.2 寄存器描述

#### 36.5.2.1 DBG器件识别码寄存器 (DBGC\_IDCODE)

DBG 器件识别码寄存器 (DBGC_IDCODE)																															
偏移地址: 000 <sub>H</sub>																															
上电复位值: 00010000_00000000_00110010_01100101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV_ID																CORE_ID				DEV_ID											

REV_ID	Bits 31-16	R	版本识别码 0x1000: 版本 A 0x1001: 版本 B
CORE_ID	Bits 15-12	R	内核识别码 0x0: Cortex-M0 0x3: Cortex-M3
DEV_ID	Bits 11-0	R	器件识别码 0x265: MCU 识别码

#### 36.5.2.2 DBG控制寄存器 (DBGC\_CR)

DBG 控制寄存器 (DBGC_CR)																																			
偏移地址: 004 <sub>H</sub>																																			
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																												DBG_STANDBY		DBG_STOP2		DBG_STOP1		DBG_SLEEP	

—	Bits 31-4	—	—
DBG_STANDBY	Bit 3	R/W	调试 <b>STANDBY</b> 模式选择位 0: FCLK 关闭, HCLK 关闭, 主电源域掉电 1: FCLK 开启, HCLK 开启, 主电源域不掉电
DBG_STOP2	Bit 2	R/W	调试 <b>STOP2</b> 模式选择位 0: FCLK 关闭, HCLK 关闭 1: FCLK 开启, HCLK 开启
DBG_STOP1	Bit 1	R/W	调试 <b>STOP1</b> 模式选择位

			0: FCLK 关闭, HCLK 关闭 1: FCLK 开启, HCLK 开启
DBG_SLEEP	Bit 0	R/W	调试 <b>SLEEP</b> 模式选择位 0: FCLK 开启, HCLK 关闭 1: FCLK 开启, HCLK 开启

注 1: 该寄存器仅支持按字写入;  
注 2: 此控制寄存器所有控制位无效;

### 36.5.2.3 APB1 外设调试冻结寄存器 (DBGC\_APB1FZ)

APB1 外设调试冻结寄存器 (DBGC_APB1FZ)																															
偏移地址: 008 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											I2C1_SMBUS_TO		I2C0_SMBUS_TO		TIM7_STOP		TIM6_STOP		TIM5_STOP		TIM4_STOP		TIM3_STOP		TIM2_STOP		TIM1_STOP		TIM0_STOP		

—	Bits 31-13	—	—
—	Bit 12	R/W	—
—	Bits 11-10	—	—
I2C1_SMBUS_TO	Bit 9	R/W	<b>I2C1 SMBUS 超时调试暂停选择位 (无效)</b> 0: 内核停止时, 超时定时器仍正常工作 1: 内核停止时, 超时定时器暂停
I2C0_SMBUS_TO	Bit 8	R/W	<b>I2C0 SMBUS 超时调试暂停选择位 (无效)</b> 0: 内核停止时, 超时定时器仍正常工作 1: 内核停止时, 超时定时器暂停
TIM7_STOP	Bit7	R/W	<b>TIM7 调试暂停选择位 (无效)</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM6_STOP	Bit6	R/W	<b>TIM6 调试暂停选择位 (无效)</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM5_STOP	Bit5	R/W	<b>TIM5 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM4_STOP	Bit4	R/W	<b>TIM4 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数

TIM3_STOP	Bit3	R/W	<b>TIM3 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM2_STOP	Bit2	R/W	<b>TIM2 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM1_STOP	Bit1	R/W	<b>TIM1 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
TIM0_STOP	Bit0	R/W	<b>TIM0 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数

注 1: 该寄存器仅支持按字写入;

### 36. 5. 2. 4 APB2 外设调试冻结寄存器 (DBGC\_APB2FZ)

APB2 外设调试冻结寄存器 (DBGC_APB2FZ)																															
偏移地址: 00C <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTC_STOP			WWDT_STOP			IWDT_STOP			Reserved			LP16T0_STOP									

—	Bits 31-11	—	—
RTC_STOP	Bit 10	R/W	<b>RTC 调试暂停选择位 (无效)</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
WWDT_STOP	Bit 9	R/W	<b>WWDT 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
IWDT_STOP	Bit 8	R/W	<b>IWDT 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数
—	Bits 7-1	—	—
LP16T0_STOP	Bit 0	R/W	<b>LP16T0 调试暂停选择位</b> 0: 内核停止时, 仍正常计数 1: 内核停止时, 暂停计数

注 1: 该寄存器仅支持按字写入;



### 36.5.2.5 定时器输出调试冻结寄存器 (DBGC\_TIMOFZ)

定时器输出调试冻结寄存器 (DBGC_TIMOFZ)																															
偏移地址: 010 <sub>H</sub>																															
上电复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TIM7_OFZ	TIM6_OFZ	TIM5_OFZ	TIM4_OFZ	TIM3_OFZ	TIM2_OFZ	TIM1_OFZ	TIM0_OFZ

—	Bits 31-8	—	—
TIM7_OFZ	Bit7	R/W	<b>TIM7 调试输出禁止选择位 (无效)</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM6_OFZ	Bit6	R/W	<b>TIM6 调试输出禁止选择位 (无效)</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM5_OFZ	Bit5	R/W	<b>TIM5 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM4_OFZ	Bit4	R/W	<b>TIM4 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM3_OFZ	Bit3	R/W	<b>TIM3 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM2_OFZ	Bit2	R/W	<b>TIM2 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM1_OFZ	Bit1	R/W	<b>TIM1 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止
TIM0_OFZ	Bit0	R/W	<b>TIM0 调试输出禁止选择位</b> 0: 内核停止时, 定时器所有通道输出使能 1: 内核停止时, 定时器所有通道输出禁止

注 1: 该寄存器仅支持按字写入;

## 第37章 PDM2PCM转换器（PDM2PCM）

### 37.1 概述

---

PDM2PCM 转换器主要将高采样率、单比特脉冲密度调制（PDM）数字音频信号转换为低采样率、16 比特精度的脉冲编码调制（PCM）数字音频信号。

### 37.2 特性

---

PDM2PCM 主要有以下特性：

- ◆ 采样速率：0.5MHz~5MHz
- ◆ 输出采样率：16KHz/48KHz/96KHz 可配
- ◆ 支持 16bit PCM output
- ◆ CIC 滤波器阶数支持 4/5/6 可配
- ◆ 增益控制：-12dB~+12dB
- ◆ 支持 DC removal
- ◆ 支持 PDM output bypass
- ◆ 支持双声道输入输出
- ◆ 降采样支持：32 倍，64 倍，128 倍
- ◆ 支持 8 级深度的 FIFO
- ◆ 支持 DMA 操作模式

### 37.3 结构框图

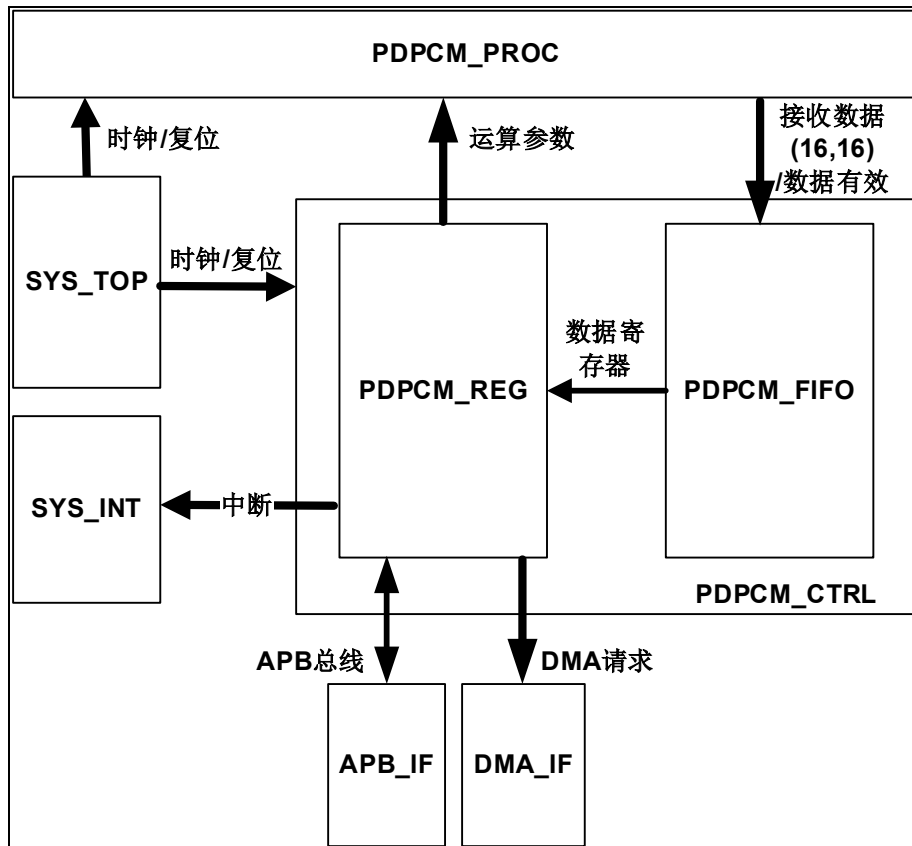


图 37-1 电路结构框图

### 37.4 功能描述

#### 37.4.1 PDMPCM\_PROC

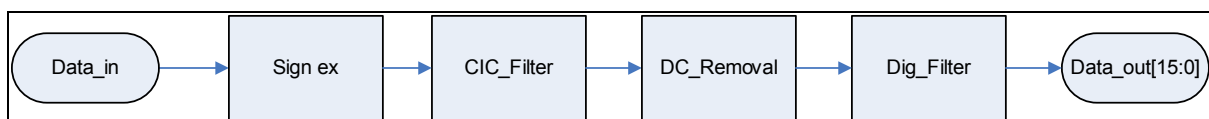


图 37-2 PDM2PCM 模块示意图

- ◇ PDM 数据输入为 1bit
- ◇ 位扩展后为 2bit (1,2,0)
- ◇ Dig\_Filter 为 91 阶 FIR 滤波器
- ◇ CIC 滤波器

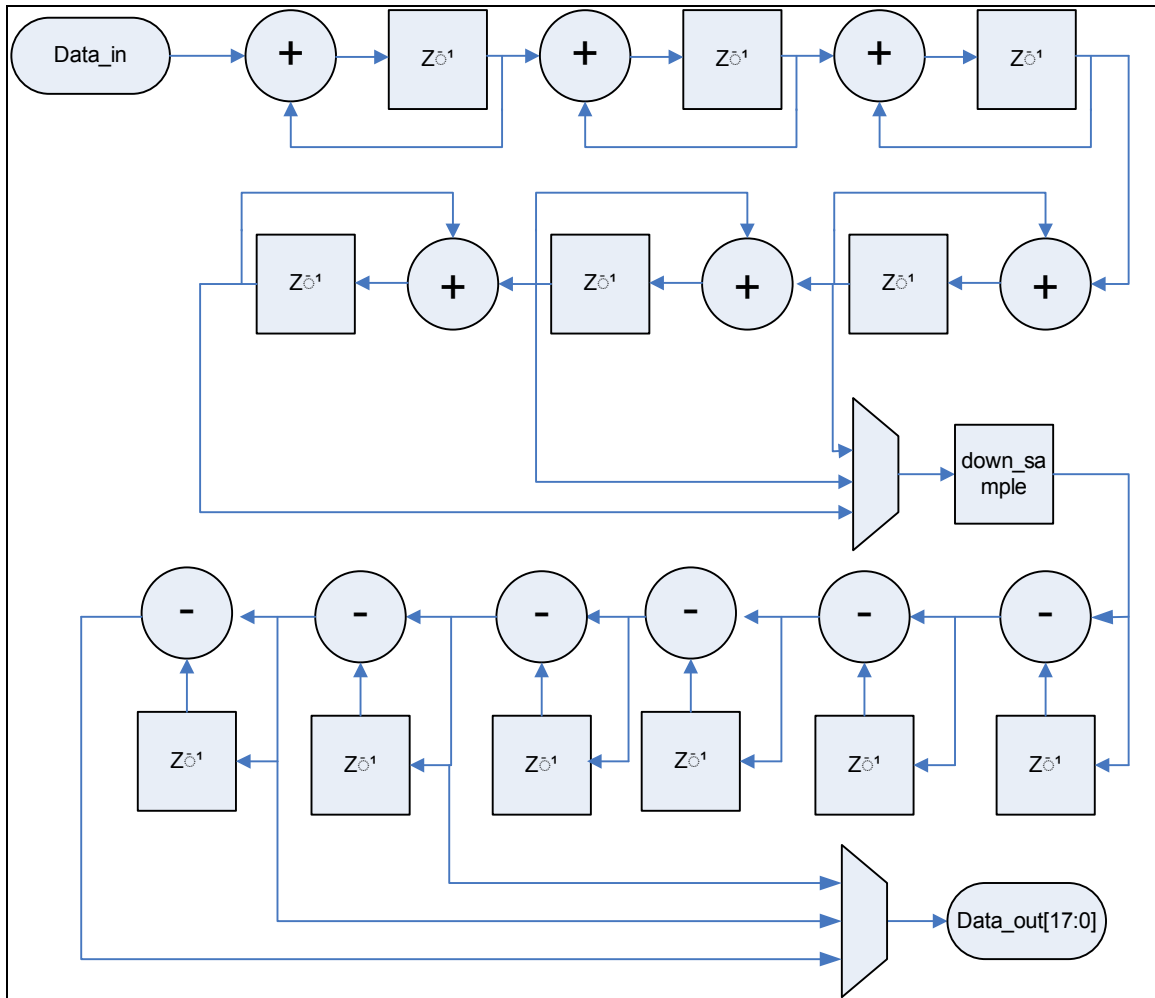


图 37-3 CIC 滤波器示意图

### 37.4.2 PDPCM\_REG

- 1) 完成 APB 接口参数的配置和回读，状态上报
- 2) 完成与 PDPCM\_FIFO 的数据读取控制
- 3) 完成中断标志维护和上报
- 4) 通过 FIFO 水线控制产生 DMA 请求
- 5) 参数列表和地址分配

### 37.4.3 PDPCM\_FIFO

- 1) 接收 PROC 输出的 16bit 低速数据，采样到 pclk 时钟域
- 2) 缓存左右声道数据
- 3) 上报 FIFO 状态和异常恢复
- 4) FIFO 位宽 32bit，深度 (8)

## 37.5 特殊功能寄存器

### 37.5.1 寄存器列表

名称	偏移地址	类型	描述
<b>PDM2PCM 基地址: 4000_9000<sub>H</sub></b>			
PDM2PCM_RXBR	0000 <sub>H</sub>	R	PDM2PCM 接收数据缓存寄存器
PDM2PCM_CTRL	0004 <sub>H</sub>	R/W	PDM2PCM 控制寄存器
PDM2PCM_FCR	0008 <sub>H</sub>	R/W	PDM2PCM FIFO 控制寄存器
PDM2PCM_0FILT	000C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 0 寄存器
PDM2PCM_1FILT	0010 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 1 寄存器
PDM2PCM_2FILT	0014 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 2 寄存器
PDM2PCM_3FILT	0018 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 3 寄存器
PDM2PCM_4FILT	001C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 4 寄存器
PDM2PCM_5FILT	0020 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 5 寄存器
PDM2PCM_6FILT	0024 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 6 寄存器
PDM2PCM_7FILT	0028 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 7 寄存器
PDM2PCM_8FILT	002C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 8 寄存器
PDM2PCM_9FILT	0030 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 9 寄存器
PDM2PCM_10FILT	0034 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 10 寄存器
PDM2PCM_11FILT	0038 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 11 寄存器
PDM2PCM_12FILT	003C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 12 寄存器
PDM2PCM_13FILT	0040 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 13 寄存器
PDM2PCM_14FILT	0044 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 14 寄存器
PDM2PCM_15FILT	0048 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 15 寄存器
PDM2PCM_16FILT	004C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 16 寄存器
PDM2PCM_17FILT	0050 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 17 寄存器
PDM2PCM_18FILT	0054 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 18 寄存器
PDM2PCM_19FILT	0058 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 19 寄存器
PDM2PCM_20FILT	005C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 20 寄存器
PDM2PCM_21FILT	0060 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 21 寄存器
PDM2PCM_22FILT	0064 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 22 寄存器
PDM2PCM_23FILT	0068 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 23 寄存器
PDM2PCM_24FILT	006C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 24 寄存器
PDM2PCM_25FILT	0070 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 25 寄存器
PDM2PCM_26FILT	0074 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 26 寄存器
PDM2PCM_27FILT	0078 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 27 寄存器
PDM2PCM_28FILT	007C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 28 寄存器
PDM2PCM_29FILT	0080 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 29 寄存器
PDM2PCM_30FILT	0084 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 30 寄存器
PDM2PCM_31FILT	0088 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 31 寄存器
PDM2PCM_32FILT	008C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 32 寄存器

PDM2PCM_33FILT	0090 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 33 寄存器
PDM2PCM_34FILT	0094 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 34 寄存器
PDM2PCM_35FILT	0098 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 35 寄存器
PDM2PCM_36FILT	009C <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 36 寄存器
PDM2PCM_37FILT	00A0 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 37 寄存器
PDM2PCM_38FILT	00A4 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 38 寄存器
PDM2PCM_39FILT	00A8 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 39 寄存器
PDM2PCM_40FILT	00AC <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 40 寄存器
PDM2PCM_41FILT	00B0 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 41 寄存器
PDM2PCM_42FILT	00B4 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 42 寄存器
PDM2PCM_43FILT	00B8 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 43 寄存器
PDM2PCM_44FILT	00BC <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 44 寄存器
PDM2PCM_45FILT	00C0 <sub>H</sub>	R/W	PDM2PCM 数字补偿滤波器系数 45 寄存器
PDM2PCM_SR	00C4 <sub>H</sub>	R	PDM2PCM 状态寄存器
PDM2PCM_IER	00C8 <sub>H</sub>	W1	PDM2PCM 中断使能寄存器
PDM2PCM_IDR	00CC <sub>H</sub>	W1	PDM2PCM 中断禁止寄存器
PDM2PCM_IVS	00D0 <sub>H</sub>	R	PDM2PCM 中断使能状态寄存器
PDM2PCM_RIF	00D4 <sub>H</sub>	R	PDM2PCM 中断标志位寄存器
PDM2PCM_IFM	00D8 <sub>H</sub>	R	PDM2PCM 中断标志位 (after mask) 寄存器
PDM2PCM_ICR	00DC <sub>H</sub>	W1	PDM2PCM 中断清零寄存器

### 37.5.2 寄存器描述

#### 37.5.2.1 PDM2PCM接收数据缓存寄存器 (PDM2PCM\_RXBR)

PDM2PCM 接收数据缓存寄存器 (PDM2PCM_RXBR)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCM_L_DATA																PCM_R_DATA															

PCM_L_DATA	Bit31-16	R	PCM 输出左声道数据
PCM_R_DATA	Bit15-0	R	PCM 输出右声道数据

#### 37.5.2.2 PDM2PCM控制寄存器 (PDM2PCM\_CTRL)

PDM2PCM 控制寄存器 (PDM2PCM_CTRL)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00100011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													EDGE_MODE	PDM2PCMenable	DMA_EN	GAIN_DB	DC_NUM	ORDER	FACTOR	DIG_FILTER_EN	DC_RM_EN										

Reserved	Bit31-14	—	保留
EDGE_MODE	Bit13	R/W	输入数据在 PDM CLK 采样模式选择 1: 下降沿采样右声道, 上升沿采样左声道 0: 下降沿采样左声道, 上升沿采样右声道
PDM2PCM_EN	Bit12	R/W	PDM2PCM 模块使能位 1: 使能 0: 不使能
DMA_EN	Bit11	R/W	DMA 使能位 1: 使能 0: 不使能
GAIN_DB	Bit10-8	R/W	增益控制位

			<p>000:1 倍</p> <p>001:2 倍</p> <p>010:4 倍</p> <p>011:8 倍</p> <p>100:16 倍</p> <p>101:32 倍</p> <p>110:64 倍</p>
			111:128 倍
DC_NUM	Bit7-6	R/W	<p>数据采样个数配置位</p> <p>00:1024</p> <p>01:2048</p> <p>10:4096</p> <p>11:8192</p>
ORDER	Bit5-4	R/W	<p><b>CIC</b> 滤波器阶数配置位</p> <p>00: 4 阶</p> <p>01: 5 阶</p> <p>1x: 6 阶</p>
FACTOR	Bit3-2	R/W	<p>降采样系数设置</p> <p>00: 32 倍</p> <p>01: 64 倍</p> <p>1x: 128倍</p>
DIG_FILTER_EN	Bit1	R/W	<p>数字滤波器补偿使能位</p> <p>1: 使能</p> <p>0: 不使能</p>
DC_RM_EN	Bit0	R/W	<p><b>DC</b> 电平去除使能位</p> <p>1: 使能</p> <p>0: 不使能</p>

### 37. 5. 2. 3 PDM2PCM FIFO控制寄存器 (PDM2PCM\_FCR)

PDM2PCM FIFO 控制寄存器 (PDM2PCM_FCR)	
偏移地址: 08 <sub>H</sub>	



复位值: 00000000\_00000000\_00000000\_00001000<sub>B</sub>

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

Reserved	Bit31-8	—	保留
FIFO_NUM	Bit7-4	R	FIFO内存数据个数上报
AF_THRESH	Bit3-1	R/W	FIFO 触发门限: 达到触发门限后触发中断或是 DMA 请求
FIFO_RST	Bit0	W	FIFO RST: 写 1 产生 FIFO 复位脉冲, 回读为 0

### 37. 5. 2. 4 PDM2PCM 数字补偿滤波器系数 0 寄存器 (PDM2PCM\_0FILT)

**PDM2PCM 数字补偿滤波器系数 0 寄存器 (PDM2PCM\_0FILT)**

偏移地址: 0C<sub>H</sub>

复位值: 00000000\_00000011\_11111111\_10110100<sub>B</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_0FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_0FILT	Bit17-0	R/W	PDM2PCM 数字补偿滤波器系数寄存器 0

### 37. 5. 2. 5 PDM2PCM 数字补偿滤波器系数 1 寄存器 (PDM2PCM\_1FILT)

**PDM2PCM 数字补偿滤波器系数 1 寄存器 (PDM2PCM\_1FILT)**

偏移地址: 10<sub>H</sub>

复位值: 00000000\_00000011\_11111111\_11010111<sub>B</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_1FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_1FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 1</b>

### 37.5.2.6 PDM2PCM 数字补偿滤波器系数 2 寄存器 (PDM2PCM\_2FILT)

PDM2PCM 数字补偿滤波器系数 2 寄存器 (PDM2PCM_2FILT)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001110 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_2FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_2FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 2</b>

### 37.5.2.7 PDM2PCM 数字补偿滤波器系数 3 寄存器 (PDM2PCM\_3FILT)

PDM2PCM 数字补偿滤波器系数 3 寄存器 (PDM2PCM_3FILT)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_01001000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_3FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_3FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 3</b>

### 37.5.2.8 PDM2PCM 数字补偿滤波器系数 4 寄存器 (PDM2PCM\_4FILT)

PDM2PCM 数字补偿滤波器系数 4 寄存器 (PDM2PCM_4FILT)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_01101110 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_4FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_4FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 4</b>

### 37.5.2.9 PDM2PCM 数字补偿滤波器系数 5 寄存器 (PDM2PCM\_5FILT)

PDM2PCM 数字补偿滤波器系数 5 寄存器 (PDM2PCM_5FILT)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_01101001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_5FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_5FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 5</b>

### 37.5.2.10 PDM2PCM 数字补偿滤波器系数 6 寄存器 (PDM2PCM\_6FILT)

PDM2PCM 数字补偿滤波器系数 6 寄存器 (PDM2PCM_6FILT)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00110010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_6FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_6FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 6</b>

### 37.5.2.11 PDM2PCM 数字补偿滤波器系数 7 寄存器 (PDM2PCM\_7FILT)

PDM2PCM 数字补偿滤波器系数 7 寄存器 (PDM2PCM_7FILT)																															
偏移地址: 28 <sub>H</sub>																															
复位值: 00000000_00000011_11111111_11010000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_7FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_7FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 7</b>

### 37.5.2.12 PDM2PCM 数字补偿滤波器系数 8 寄存器 (PDM2PCM\_8FILT)

PDM2PCM 数字补偿滤波器系数 8 寄存器 (PDM2PCM_8FILT)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000011_11111111_01100110 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_8FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_8FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 8</b>

### 37.5.2.13 PDM2PCM 数字补偿滤波器系数 9 寄存器 (PDM2PCM\_9FILT)

PDM2PCM 数字补偿滤波器系数 9 寄存器 (PDM2PCM_9FILT)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000000_00000011_11111111_00100110 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_9FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_9FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 9</b>

### 37.5.2.14 PDM2PCM 数字补偿滤波器系数 10 寄存器 (PDM2PCM\_10FILT)

PDM2PCM 数字补偿滤波器系数 10 寄存器 (PDM2PCM_10FILT)																															
偏移地址: 34 <sub>H</sub>																															
复位值: 00000000_00000011_11111111_00111101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_10FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_10FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 10</b>

### 37.5.2.15 PDM2PCM 数字补偿滤波器系数 11 寄存器 (PDM2PCM\_11FILT)

PDM2PCM 数字补偿滤波器系数 11 寄存器 (PDM2PCM_11FILT)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000011_11111111_10111100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	PDM2PCM_11FILT
----------	----------------

Reserved	Bit31-18	—	保留
PDM2PCM_11FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 11</b>

### 37.5.2.16 PDM2PCM 数字补偿滤波器系数 12 寄存器 (PDM2PCM\_12FILT)

PDM2PCM 数字补偿滤波器系数 12 寄存器 (PDM2PCM_12FILT)																																	
偏移地址: 3C <sub>H</sub>																																	
复位值: 00000000_00000000_00000000_10000011 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved																	PDM2PCM_12FILT																

Reserved	Bit31-18	—	保留
PDM2PCM_12FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 12</b>

### 37.5.2.17 PDM2PCM 数字补偿滤波器系数 13 寄存器 (PDM2PCM\_13FILT)

PDM2PCM 数字补偿滤波器系数 13 寄存器 (PDM2PCM_13FILT)																																	
偏移地址: 40 <sub>H</sub>																																	
复位值: 00000000_00000000_00000001_01000110 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved																	PDM2PCM_13FILT																

Reserved	Bit31-18	—	保留
PDM2PCM_13FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 13</b>

### 37. 5. 2. 18 PDM2PCM 数字补偿滤波器系数 14 寄存器 (PDM2PCM\_14FILT)

PDM2PCM 数字补偿滤波器系数寄存器 14 (PDM2PCM_14FILT)																															
偏移地址: 44 <sub>H</sub>																															
复位值: 00000000_00000000_00000001_10100010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_14FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_14FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 14</b>

### 37. 5. 2. 19 PDM2PCM 数字补偿滤波器系数 15 寄存器 (PDM2PCM\_15FILT)

PDM2PCM 数字补偿滤波器系数寄存器 15 (PDM2PCM_15FILT)																															
偏移地址: 48 <sub>H</sub>																															
复位值: 00000000_00000000_00000001_01001100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_15FILT																	

Reserved	Bit31-18	—	保留
----------	----------	---	----

PDM2PCM_15FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 15</b>
----------------	---------	-----	--------------------------------

### 37.5.2.20 PDM2PCM 数字补偿滤波器系数 16 寄存器 (PDM2PCM\_16FILT)

PDM2PCM 数字补偿滤波器系数 16 寄存器 (PDM2PCM_16FILT)																															
偏移地址: 4C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00111100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_16FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_16FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 16</b>

### 37.5.2.21 PDM2PCM 数字补偿滤波器系数 17 寄存器 (PDM2PCM\_17FILT)

PDM2PCM 数字补偿滤波器系数 17 寄存器 (PDM2PCM_17FILT)																															
偏移地址: 50 <sub>H</sub>																															
复位值: 00000000_00000011_11111110_11000101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_17FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_17FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 17</b>



### 37. 5. 2. 22 PDM2PCM 数字补偿滤波器系数 18 寄存器 (PDM2PCM\_18FILT)

PDM2PCM 数字补偿滤波器系数 18 寄存器 (PDM2PCM_18FILT)																															
偏移地址: 54 <sub>H</sub>																															
复位值: 00000000_00000011_11111101_10000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_18FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_18FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 18</b>

### 37. 5. 2. 23 PDM2PCM 数字补偿滤波器系数 19 寄存器 (PDM2PCM\_19FILT)

PDM2PCM 数字补偿滤波器系数 19 寄存器 (PDM2PCM_19FILT)																															
偏移地址: 58 <sub>H</sub>																															
复位值: 00000000_00000011_11111101_00011110 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_19FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_19FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 19</b>

### 37.5.2.24 PDM2PCM 数字补偿滤波器系数 20 寄存器 (PDM2PCM\_20FILT)

PDM2PCM 数字补偿滤波器系数 20 寄存器 (PDM2PCM_20FILT)																															
偏移地址: 5C <sub>H</sub>																															
复位值: 00000000_00000011_11111101_11111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		PDM2PCM_20FILT													

Reserved	Bit31-18	—	保留
PDM2PCM_20FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 20</b>

### 37.5.2.25 PDM2PCM 数字补偿滤波器系数 21 寄存器 (PDM2PCM\_21FILT)

PDM2PCM 数字补偿滤波器系数 21 寄存器 (PDM2PCM_21FILT)																															
偏移地址: 60 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		PDM2PCM_21FILT													

Reserved	Bit31-18	—	保留
PDM2PCM_21FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 21</b>

### 37.5.2.26 PDM2PCM 数字补偿滤波器系数 22 寄存器 (PDM2PCM\_22FILT)

PDM2PCM 数字补偿滤波器系数 22 寄存器 (PDM2PCM_22FILT)																															
偏移地址: 64 <sub>H</sub>																															
复位值: 00000000_00000000_00000010_10001011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_22FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_22FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 22</b>

### 37.5.2.27 PDM2PCM 数字补偿滤波器系数 23 寄存器 (PDM2PCM\_23FILT)

PDM2PCM 数字补偿滤波器系数 23 寄存器 (PDM2PCM_23FILT)																															
偏移地址: 68 <sub>H</sub>																															
复位值: 00000000_00000000_00000100_01101111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_23FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_23FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 23</b>

### 37.5.2.28 PDM2PCM 数字补偿滤波器系数 24 寄存器 (PDM2PCM\_24FILT)

PDM2PCM 数字补偿滤波器系数 24 寄存器 (PDM2PCM_24FILT)																															
偏移地址: 6C <sub>H</sub>																															
复位值: 00000000_00000000_00000100_10101101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_24FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_24FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 24</b>

### 37.5.2.29 PDM2PCM 数字补偿滤波器系数 25 寄存器 (PDM2PCM\_25FILT)

PDM2PCM 数字补偿滤波器系数 25 寄存器 (PDM2PCM_25FILT)																															
偏移地址: 70 <sub>H</sub>																															
复位值: 00000000_00000000_00000010_11001001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_25FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_25FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 25</b>

### 37.5.2.30 PDM2PCM 数字补偿滤波器系数 26 寄存器 (PDM2PCM\_26FILT)

PDM2PCM 数字补偿滤波器系数 26 寄存器 (PDM2PCM_26FILT)																															
偏移地址: 74 <sub>H</sub>																															
复位值: 00000000_00000011_11111111_00101100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_26FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_26FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 26</b>

### 37.5.2.31 PDM2PCM 数字补偿滤波器系数 27 寄存器 (PDM2PCM\_27FILT)

PDM2PCM 数字补偿滤波器系数 27 寄存器 (PDM2PCM_27FILT)																															
偏移地址: 78 <sub>H</sub>																															
复位值: 00000000_00000011_11111011_00100011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_27FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_27FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 27</b>

### 37. 5. 2. 32 PDM2PCM 数字补偿滤波器系数 28 寄存器 (PDM2PCM\_28FILT)

PDM2PCM 数字补偿滤波器系数 28 寄存器 (PDM2PCM_28FILT)																															
偏移地址: 7C <sub>H</sub>																															
复位值: 00000000_00000011_11111000_01111000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_28FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_28FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 28</b>

### 37. 5. 2. 33 PDM2PCM 数字补偿滤波器系数 29 寄存器 (PDM2PCM\_29FILT)

PDM2PCM 数字补偿滤波器系数 29 寄存器 (PDM2PCM_29FILT)																															
偏移地址: 80 <sub>H</sub>																															
复位值: 00000000_00000011_11111000_10110100 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_29FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_29FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 29</b>

### 37.5.2.34 PDM2PCM 数字补偿滤波器系数 30 寄存器 (PDM2PCM\_30FILT)

PDM2PCM 数字补偿滤波器系数 30 寄存器 (PDM2PCM_30FILT)																															
偏移地址: 84 <sub>H</sub>																															
复位值: 00000000_00000011_11111100_01100001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_30FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_30FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 30</b>

### 37.5.2.35 PDM2PCM 数字补偿滤波器系数 31 寄存器 (PDM2PCM\_31FILT)

PDM2PCM 数字补偿滤波器系数 31 寄存器 (PDM2PCM_31FILT)																															
偏移地址: 88 <sub>H</sub>																															
复位值: 00000000_00000000_00000010_10010010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_31FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_31FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 31</b>

**37. 5. 2. 36 PDM2PCM 数字补偿滤波器系数 32 寄存器 (PDM2PCM\_32FILT)**

PDM2PCM 数字补偿滤波器系数 32 寄存器 (PDM2PCM_32FILT)																															
偏移地址: 8C <sub>H</sub>																															
复位值: 00000000_00000000_00001001_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_32FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_32FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 32</b>

**37. 5. 2. 37 PDM2PCM 数字补偿滤波器系数 33 寄存器 (PDM2PCM\_33FILT)**

PDM2PCM 数字补偿滤波器系数 33 寄存器 (PDM2PCM_33FILT)																															
偏移地址: 90 <sub>H</sub>																															
复位值: 00000000_00000000_00001100_11000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_33FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_33FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 33</b>



### 37.5.2.38 PDM2PCM 数字补偿滤波器系数 34 寄存器 (PDM2PCM\_34FILT)

PDM2PCM 数字补偿滤波器系数 34 寄存器 (PDM2PCM_34FILT)																															
偏移地址: 94 <sub>H</sub>																															
复位值: 00000000_00000000_00001011_10000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_34FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_34FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 34</b>

### 37.5.2.39 PDM2PCM 数字补偿滤波器系数 35 寄存器 (PDM2PCM\_35FILT)

PDM2PCM 数字补偿滤波器系数 35 寄存器 (PDM2PCM_35FILT)																															
偏移地址: 98 <sub>H</sub>																															
复位值: 00000000_00000000_00000100_10000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_35FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_35FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 35</b>

### 37.5.2.40 PDM2PCM 数字补偿滤波器系数 36 寄存器 (PDM2PCM\_36FILT)

PDM2PCM 数字补偿滤波器系数 36 寄存器 (PDM2PCM_36FILT)																															
偏移地址: 9C <sub>H</sub>																															
复位值: 00000000_00000011_11111001_01110101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_36FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_36FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 36</b>

### 37.5.2.41 PDM2PCM 数字补偿滤波器系数 37 寄存器 (PDM2PCM\_37FILT)

PDM2PCM 数字补偿滤波器系数 37 寄存器 (PDM2PCM_37FILT)																															
偏移地址: A0 <sub>H</sub>																															
复位值: 00000000_00000011_11101110_00111000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_37FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_37FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 37</b>

### 37.5.2.42 PDM2PCM 数字补偿滤波器系数 38 寄存器 (PDM2PCM\_38FILT)

**PDM2PCM 数字补偿滤波器系数 38 寄存器 (PDM2PCM\_38FILT)**

偏移地址: A4 <sub>H</sub>																															
复位值: 00000000_00000011_11100111_11010100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_38FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_38FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 38</b>

**37. 5. 2. 43 PDM2PCM 数字补偿滤波器系数 39 寄存器 (PDM2PCM\_39FILT)**

**PDM2PCM 数字补偿滤波器系数 39 寄存器 (PDM2PCM\_39FILT)**

偏移地址: A8 <sub>H</sub>																															
复位值: 00000000_00000011_11101010_11010001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_39FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_39FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 39</b>

**37. 5. 2. 44 PDM2PCM 数字补偿滤波器系数 40 寄存器 (PDM2PCM\_40FILT)**

**PDM2PCM 数字补偿滤波器系数 40 寄存器 (PDM2PCM\_40FILT)**

偏移地址: AC <sub>H</sub>																															
-----------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

复位值: 00000000_00000011_11111001_01111100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_40FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_40FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 40</b>

### 37.5.2.45 PDM2PCM 数字补偿滤波器系数 41 寄存器 (PDM2PCM\_41FILT)

<b>PDM2PCM 数字补偿滤波器系数 41 寄存器 (PDM2PCM_41FILT)</b>																															
偏移地址: B0 <sub>H</sub>																															
复位值: 00000000_00000000_00010010_10111111 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_41FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_41FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 41</b>

### 37.5.2.46 PDM2PCM 数字补偿滤波器系数 42 寄存器 (PDM2PCM\_42FILT)

<b>PDM2PCM 数字补偿滤波器系数 42 寄存器 (PDM2PCM_42FILT)</b>																															
偏移地址: B4 <sub>H</sub>																															
复位值: 00000000_00000000_00110010_00001001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	PDM2PCM_42FILT
----------	----------------

Reserved	Bit31-18	—	保留
PDM2PCM_42FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 42</b>

### 37.5.2.47 PDM2PCM 数字补偿滤波器系数 43 寄存器 (PDM2PCM\_43FILT)

PDM2PCM 数字补偿滤波器系数 43 寄存器 (PDM2PCM_43FILT)																																	
偏移地址: B8 <sub>H</sub>																																	
复位值: 00000000_00000000_01010000_01100010 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved														PDM2PCM_43FILT																			

Reserved	Bit31-18	—	保留
PDM2PCM_43FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 43</b>

### 37.5.2.48 PDM2PCM 数字补偿滤波器系数 44 寄存器 (PDM2PCM\_44FILT)

PDM2PCM 数字补偿滤波器系数 44 寄存器 (PDM2PCM_44FILT)																																	
偏移地址: BC <sub>H</sub>																																	
复位值: 00000000_00000000_01100110_01100001 <sub>B</sub>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

Reserved	PDM2PCM_44FILT
----------	----------------

Reserved	Bit31-18	—	保留
PDM2PCM_44FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 44</b>

### 37.5.2.49 PDM2PCM 数字补偿滤波器系数 45 寄存器 (PDM2PCM\_45FILT)

PDM2PCM 数字补偿滤波器系数 45 寄存器 (PDM2PCM_45FILT)																															
偏移地址: C0 <sub>H</sub>																															
复位值: 00000000_00000000_01101110_01101001 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PDM2PCM_45FILT																	

Reserved	Bit31-18	—	保留
PDM2PCM_45FILT	Bit17-0	R/W	<b>PDM2PCM 数字补偿滤波器系数寄存器 45</b>

### 37.5.2.50 PDM2PCM状态寄存器 (PDM2PCM\_SR)

PDM2PCM 状态寄存器 (PDM2PCM_SR)																															
偏移地址: C4 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	RFUERR	RFOERR	RF_FULL	RF_EMPTY	RF_TH
----------	--------	--------	---------	----------	-------

Reserved	Bit31-5	—	保留
RFUERR	Bit4	R	<p><b>接收器FIFO下溢错误</b> 当接收器FIFO内无数据时，又再次读取接收器FIFO时，将会置起此位。此位由硬件设置1，在接收数据时清除或设置ICR寄存器的RFUERR位清除</p> <p>0: 接收器FIFO下溢错误未产生 1: 接收器 FIFO 下溢错误产生</p>
RFOERR	Bit3	R	<p><b>接收器FIFO溢出错误</b> 当接收器FIFO内已有8个数据时，有新数据再次接收至FIFO中时，将会置起此位并舍弃新数据。此位由硬件设置1，在读取数据时清除或设置ICR寄存器的RFOERR位清除</p> <p>0: 接收器FIFO溢出错误未产生 1: 接收器 FIFO 溢出错误产生</p>
RF_FULL	Bit2	R	<p><b>接收器FIFO满</b> 当接收器FIFO内有8个数据时，此位由硬件设置1，在FIFO内未满足8个数据时清除。</p> <p>0: 接收器FIFO未满足16个数据 1: 接收器FIFO满足16个数据</p>
RF_EMPTY	Bit1	R	<p><b>接收器FIFO满</b> 当接收器FIFO内有16个数据时，此位由硬件设置1，在FIFO内未满足16个数据时清除。</p> <p>0: 接收器FIFO未满足16个数据 1: 接收器 FIFO 满足 16 个数据</p>
RF_TH	Bit0	R	<p><b>接收器FIFO触发阈值</b> 当FCR寄存器的FIFO_NUM位所指示的FIFO字节数大于等于FCR寄存器的AF_THRESH设定的阈值，将会置起此位。此位由硬件设置1，在未满足阈值水平时清除</p> <p>0: 接收器FIFO未大于等于阈值水平 1: 接收器 FIFO 大于等于阈值水平</p>

### 37. 5. 2. 51 PDM2PCM中断使能寄存器 (PDM2PCM\_IER)

PDM2PCM 中断使能寄存器 (PDM2PCM_IER)																															
偏移地址: C8 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	RFUERR	RFOERR	RF_FULL	Reserved	RF_TH
----------	--------	--------	---------	----------	-------

Reserved	Bit31-5	—	保留
RFUERR	Bit4	W1	接收器 FIFO 下溢中断使能 0: 写入0无效 1: 接收器 FIFO 下溢中断使能
RFOERR	Bit3	W1	接收器FIFO溢出中断使能 0: 写入0无效 1: 接收器 FIFO 溢出中断使能
RF_FULL	Bit2	W1	接收器FIFO满中断使能 0: 写入0无效 1: 接收器FIFO满中断使能
Reserved	Bit1	—	保留
RF_TH	Bit0	W1	接收器FIFO触发阈值中断使能 0: 写入0无效 1: 接收器 FIFO 触发阈值中断使能

### 37. 5. 2. 52 PDM2PCM中断不使能寄存器 (PDM2PCM\_IDR)

PDM2PCM 中断不使能寄存器 (PDM2PCM_IDR)																																
偏移地址: CC <sub>H</sub>																																
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																												RFUERR	RFOERR	RF_FULL	Reserved	RF_TH

Reserved	Bit31-5	—	保留
RFUERR	Bit4	W1	接收器FIFO下溢中断禁止 0: 写入0无效 1: 接收器 FIFO 下溢中断禁止
RFOERR	Bit3	W1	接收器FIFO溢出中断禁止 0: 写入0无效 1: 接收器 FIFO 溢出中断禁止
RF_FULL	Bit2	W1	接收器FIFO满中断禁止 0: 写入0无效 1: 接收器FIFO满中断禁止
Reserved	Bit1	—	保留
RF_TH	Bit0	W1	接收器FIFO触发阈值中断禁止



			0: 写入0无效 1: 接收器 FIFO 触发阈值中断禁止
--	--	--	----------------------------------

### 37. 5. 2. 53 PDM2PCM中断使能状态寄存器 (PDM2PCM\_IVS)

PDM2PCM 中断使能状态寄存器 (PDM2PCM_IVS)																															
偏移地址: D0 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											RFUERR	RFOERR	RF_FULL	Reserved	RF_TH

Reserved	Bit31-5	—	保留
RFUERR	Bit4	R	接收器FIFO下溢中断有效位 0: 接收器FIFO下溢中断禁止 1: 接收器 FIFO 下溢中断使能
RFOERR	Bit3	R	接收器FIFO溢出中断有效位 0: 接收器FIFO溢出中断禁止 1: 接收器 FIFO 溢出中断使能
RF_FULL	Bit2	R	接收器 FIFO 满中断有效位 0: 接收器FIFO满中断禁止 1: 接收器 FIFO 满中断使能
Reserved	Bit1	—	保留
RF_TH	Bit0	R	接收器FIFO触发阈值中断有效位 0: 接收器FIFO触发阈值中断禁止 1: 接收器 FIFO 触发阈值中断使能

### 37. 5. 2. 54 PDM2PCM中断标志位寄存器 (PDM2PCM\_RIF)

PDM2PCM 中断标志位寄存器 (PDM2PCM_RIF)																															
偏移地址: D4 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	RFUERR	RFOERR	RF_FULL	Reserved	RF_TH
----------	--------	--------	---------	----------	-------

Reserved	Bit31-5	—	保留
RFUERR	Bit4	R	<p><b>接收器FIFO下溢中断标志</b> 当接收器FIFO内无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除</p> <p>0：无中断产生 1：接收器 FIFO 下溢中断产生</p>
RFOERR	Bit3	R	<p><b>接收器FIFO溢出中断标志</b> 当接收器FIFO内已有16个数据时，有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1，设置ICR寄存器的RFOERR位清除</p> <p>0：无中断产生 1：接收器 FIFO 溢出中断产生</p>
RF_FULL	Bit2	R	<p><b>接收器FIFO满中断标志</b> 当接收器FIFO内由15个数据接收至16个数据时置起，或设置IER寄存器的RFFULL位时，UART会判断当前接收器FIFO是否为16个数据而置起。此位由硬件设置1，设置ICR寄存器的RFFULL位清除</p> <p>0：无中断产生 1：接收器 FIFO 满中断产生</p>
Reserved	Bit1	—	保留
RF_TH	Bit0	R	<p><b>接收器FIFO触发阈值中断标志</b> 接收器FIFO个数达到接收器设定的阈值时置起，或设置IER寄存器的RFTH位时，UART会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的RFTH位清除</p> <p>0：无中断产生 1：接收器 FIFO 触发阈值中断产生</p>

### 37. 5. 2. 55 PDM2PCM中断标志位masked寄存器 (PDM2PCM\_IFM)

PDM2PCM 中断标志位 masked 寄存器 (PDM2PCM_IFM)																															
偏移地址: D8 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	RFUERR	RFOERR	RF_FULL	Reserved	RF_TH
----------	--------	--------	---------	----------	-------

Reserved	Bit31-5	—	保留
RFUERR	Bit4	R	<p><b>接收器FIFO下溢中断屏蔽标志</b></p> <p>当IVS寄存器的RFUERR位为1时，接收器FIFO内无数据时，又再次读取接收器FIFO时置起。此位由硬件设置1，设置ICR寄存器的RFUERR位清除</p> <p>0：无中断产生 1：接收器 FIFO 下溢中断产生</p>
RFOERR	Bit3	R	<p><b>接收器 FIFO 溢出中断屏蔽标志</b></p> <p>当IVS寄存器的RFOERR位为1时，接收器FIFO内已有8个数据时，有新数据再次接收至FIFO中时置起并舍弃新数据。此位由硬件设置1，设置ICR寄存器的RFOERR位清除</p> <p>0：无中断产生 1：接收器 FIFO 溢出中断产生</p>
RF_FULL	Bit2	R	<p><b>接收器 FIFO 满中断屏蔽标志</b></p> <p>当IVS寄存器的RF_FULL位为1时，接收器FIFO内由7个数据接收至8个数据时置起，或设置IER寄存器的RF_FULL位时，内部电路会判断当前接收器FIFO是否为8个数据而置起。此位由硬件设置1，设置ICR寄存器的RF_FULL位清除</p> <p>0：无中断产生 1：接收器 FIFO 满中断产生</p>
Reserved	Bit1	—	保留
RF_TH	Bit0	R	<p><b>接收器FIFO触发阈值中断屏蔽标志</b></p> <p>当IVS寄存器的RF_TH位为1时，接收器FIFO个数达到接收器设定的阈值时置起，或设置IER寄存器的RF_TH位时，内部电路会判断当前接收器FIFO个数达到接收器设定的阈值将此位置起。此位由硬件设置1，设置ICR寄存器的RF_TH位清除</p> <p>0：无中断产生 1：接收器 FIFO 触发阈值中断产生</p>

### 37. 5. 2. 56 PDM2PCM中断清除寄存器 (PDM2PCM\_ICR)

PDM2PCM 中断清除寄存器 (PDM2PCM_ICR)																															
偏移地址: DC <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	RFUERR	RFOERR	RF_FULL	Reserved	RF_TH
----------	--------	--------	---------	----------	-------

Reserved	Bit31-5	—	保留
RFUERR	Bit4	W1	接收器FIFO下溢中断清除 0: 写入0无效 1: 接收器 FIFO 下溢中断清除
RFOERR	Bit3	W1	接收器 FIFO 溢出中断清除 0: 写入0无效 1: 接收器 FIFO 溢出中断清除
RF_FULL	Bit2	W1	接收器FIFO满中断清除 0: 写入0无效 1: 接收器 FIFO 满中断清除
Reserved	Bit1	—	保留
RF_TH	Bit0	W1	接收器FIFO触发阈值中断清除 0: 写入0无效 1: 接收器 FIFO 触发阈值中断清除

## 第38章 低功耗蓝牙（BLE）

### 38.1 概述

低功耗蓝牙（Bluetooth Low Energy, BLE）由蓝牙技术联盟（Bluetooth SIG）设计、授权认证销售的一种个人局域网技术。工作在 2.4GHz ISM 频段，标准号 IEEE 802.15.1。低功耗蓝牙使用 40 个无线信道进行信息传输，分别为 3 个广播信道和 37 个数据信道。载波频偏为  $(2402+2k)$  MHz, ( $k=0,1,2, \dots, 39$ )，载波频偏间隔为 2MHz。采用调制指数为 0.45~0.55 的 GFSK 调制方式。

蓝牙 5.0 由蓝牙技术联盟于 2016 年 6 月 16 日在伦敦正式发布，相比蓝牙 4.2 版本，性能得到很大的改善：

- 1) 两倍的数据传输速度，从 1Mbps 提高到 2Mbps；
- 2) 四倍的传输距离，理论上有效工作距离可以达到 300 米；
- 3) 八倍的广播数据容量，由 31 Byte 提高到 255 Byte，从而减少因 2.4GHz 频段干扰而造成的传输速率损失。

### 38.2 特性

主要特性如下：

- ◆ 蓝牙 5.0 核心规范
- ◆ 所有 device types(Broadcaster/Central/Observer/Peripheral)
- ◆ 所有包格式 (Broadcasting/Advertising/Data/Control)
- ◆ 蓝牙 5.0 Long Range
- ◆ 加解密 AES-CCM
- ◆ 数据流处理 (CRC/Whitening)
- ◆ FDMA / TDMA /events formatting and synchronization
- ◆ 跳频计算
- ◆ 低功耗模式支持 32.768 kHz

### 38.3 结构框图

图 38-1 给出了 BLE Core 模块的结构框图。BLE Core 即蓝牙基带部分，负责管理物理信道以及报文帧结构处理、编解码等链路控制。

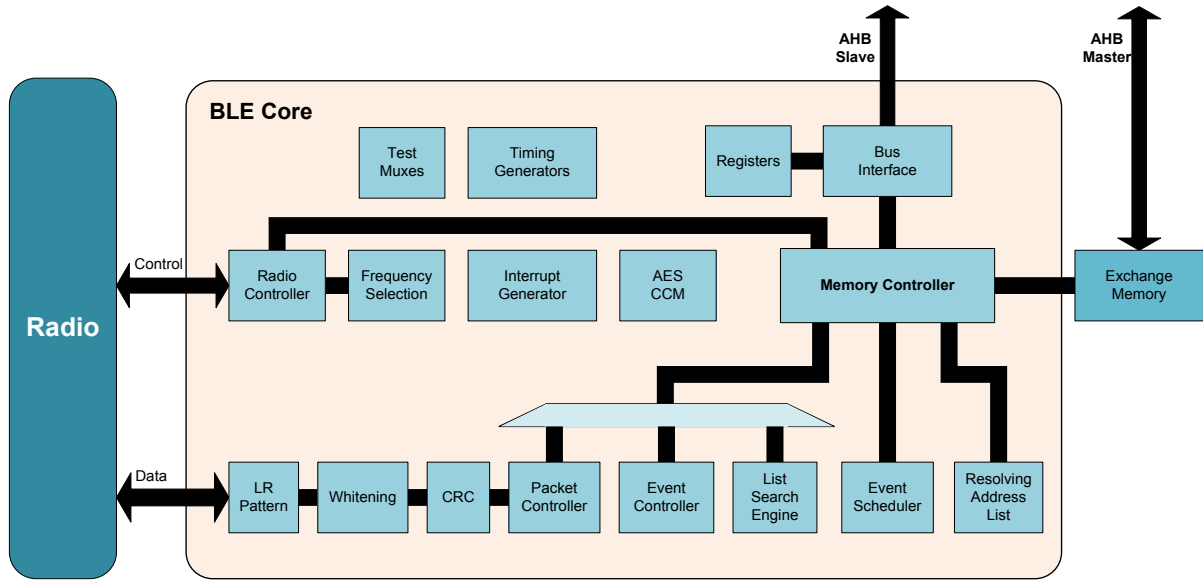


图 38-1 BLE Core 结构框图

## 38.4 功能描述

### 38.4.1 Exchange Memory

芯片内部集成 2 个 Exchange Memory(数据交换存储器, 简称 EM), 作为 BLE Baseband Core 的数据交换使用。当 BLE 连接数不同时, 所需要的 EM Size 不同, 可通过配置选择使用相应大小 EM。

注: 该 EM 被分为 2 个独立的 SRAM 来实现, Power Down 功能实现低功耗设计。

EM 地址映射:

Address	Description	AMBA
0x400C_8000~0x400C_BFFF	16KB retention RAM0	AHB
0x400C_C000~0x400C_DFFF	8KB retention RAM1	AHB

### 38.4.2 低功耗睡眠及唤醒

BLE Core 睡眠及唤醒功能由 Timing Generators 模块实现。

#### 38.4.2.1 工作模式切换至睡眠模式

可通过软件配置寄存器 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 1, 使能 BLE Core 从工作模式 (Active Mode) 切换至睡眠模式 (Deep Sleep Mode), 参见章节 6.5.11。

进入睡眠模式后, BLE Core 系统时钟 master1\_gclk 关闭, 采用 32.768 kHz 低速时钟 low\_power\_clk 作为内部 625us 时序参考。

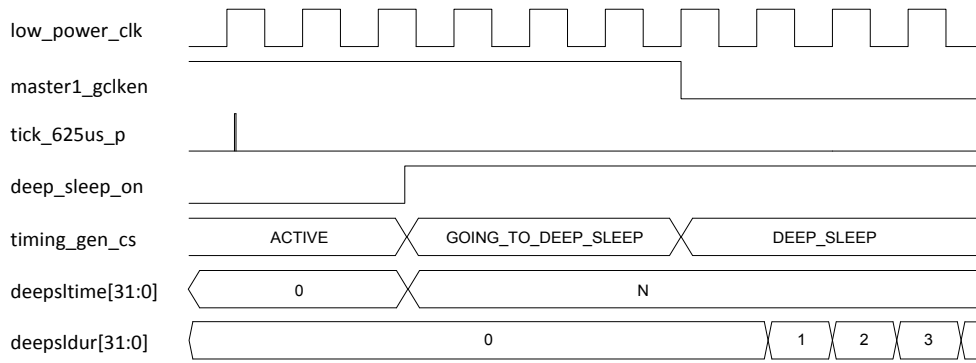


图 38-2 Core 进入 Deep Sleep Mode

### 38.4.2.2 睡眠模式切换至工作模式

BLE Core 从睡眠模式（Deep Sleep Mode）唤醒有两种方式：1）自动超时唤醒；2）异步事件唤醒。

### 38.4.2.3 睡眠模式自动超时唤醒

自动超时唤醒方式下，需预设 BLE Core 低功耗睡眠模式唤醒时间，硬件以 32.768 kHz 低速时钟 low\_power\_clk 进行睡眠时间计数，达到预设唤醒时间时，硬件将自动唤醒 BLE Core 模块，打开 BLE Core 系统时钟 master1\_gclk，相关寄存器配置描述如下：

#### 1) 异步事件唤醒方式禁止

寄存器 DEEPSLCNTL->EXTWKUPDSB 配置为 1，此时仅支持自动超时唤醒方式。

DEEPSLWKUP 配置预设的唤醒时间（此时，DEEPSLWKUP 需配置非 0 值）；最后配置 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 1，使能 BLE Core 进入睡眠模式。

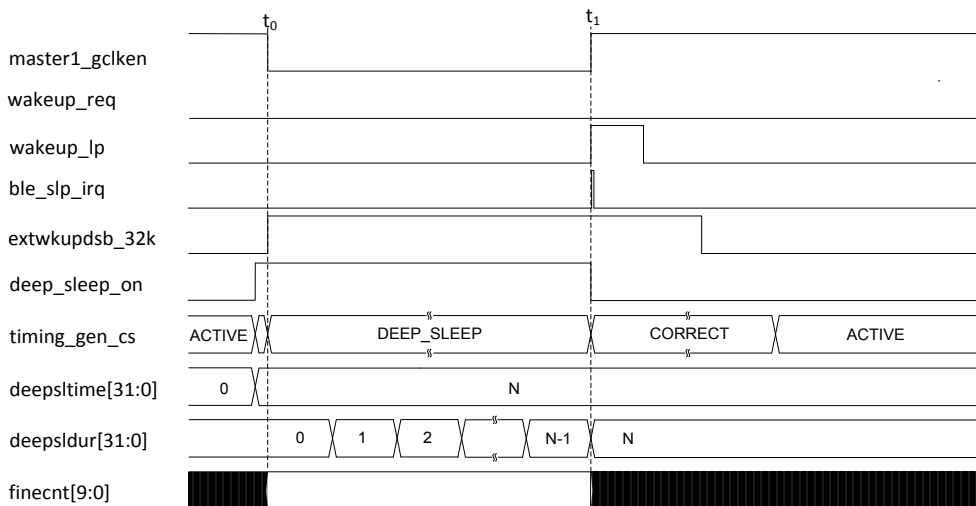


图 38-3 BLE Core Deep Sleep Mode 自动超时唤醒（extwkupdsb=1）

注：当异步事件唤醒方式禁止时，若寄存器 DEEPSLWKUP 配置为 0，则进入无限睡眠模式，两种唤醒模式均无法唤醒 BLE Core。

#### 2) 异步事件唤醒方式使能且异步事件延时触发

寄存器 DEEPSLCNTL->EXTWKUPDSB 配置为 0，异步事件唤醒使能。

寄存器 DEEPSLWKUP 配置为非 0 值，因异步事件触发时刻点在 DEEPSLWKUP 预设唤醒时间之后，则 BLE Core 将由硬件提前自动唤醒。

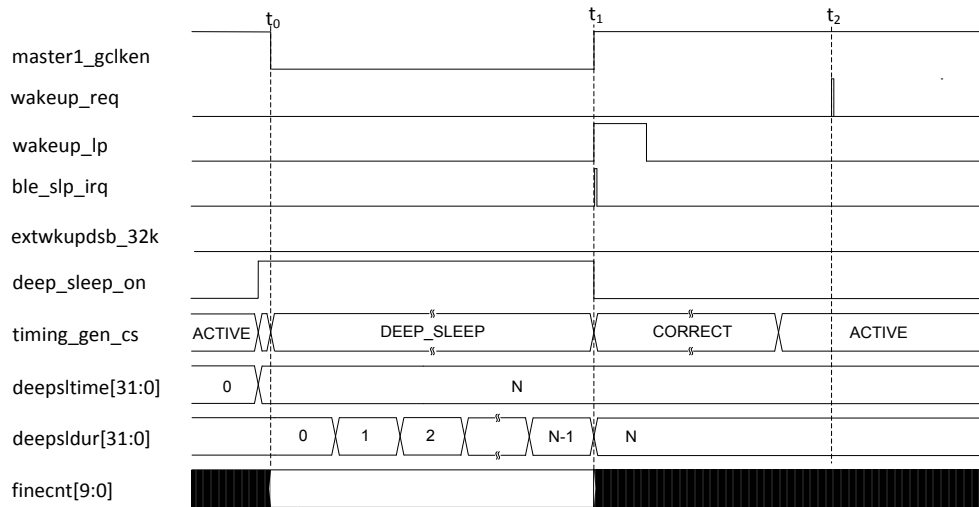


图 38-4 BLE Core Deep Sleep Mode 自动超时唤醒 (extwkupdsb=0)

#### 38.4.2.4 睡眠模式异步事件唤醒

异步事件唤醒，即通过外部的异步事件将 BLE Core 从低功耗模式唤醒（中断唤醒源详见章节 6.5.10），相关寄存器配置描述如下：

配置寄存器 DEEPSLCNTL->EXTWKUPDSB 配置为 0，异步事件唤醒使能。

寄存器 ENBPRESET->TWEXT[10:0]用于外部中断源触发后、唤醒 BLE Core 之前，高速晶振需要的稳定时间配置；最后配置 DEEPSLCNTL->DEEP\_SLEEP\_ON 为 1，使能 BLE Core 进入睡眠模式。

当前配置情况下，若异步事件触发时刻点在 DEEPSLWKUP 预设唤醒时间之前，则 BLE Core 将由异步事件唤醒。



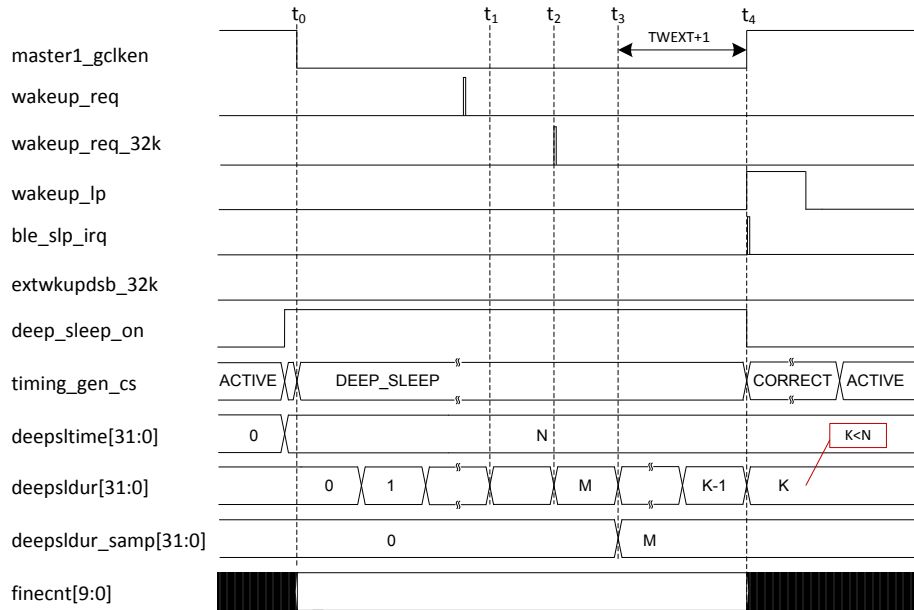


图 38-5 BLE Core Deep Sleep Mode 异步事件唤醒

注：当异步事件唤醒方式使能时，当寄存器 DEEPSLWKUP 配置为 0 时，仅支持异步事件唤醒。

### 38.5 特殊功能寄存器

此部分寄存器仅对蓝牙模拟前端控制寄存器进行描述，蓝牙 Baseband 和 Modem 部分寄存器通过其它文件提供。

#### 38.5.1 寄存器列表

名称	偏移地址	类型	描述
AFE 基地址: 400E_0400 <sub>H</sub>			
AFE_ENABLE	00 <sub>H</sub>	R/W	AFE 接口信号使能寄存器
AFE_TEST_ENABLE	04 <sub>H</sub>	R/W	AFE 测试使能控制寄存器
AFE_PLLANACR1	08 <sub>H</sub>	R/W	AFE PLL 模拟参数控制寄存器 1
AFE_PLLANACR2	0C <sub>H</sub>	R/W	AFE PLL 模拟参数控制寄存器 2
AFE_PLLANACR3	10 <sub>H</sub>	R/W	AFE PLL 模拟参数控制寄存器 3
AFE_TXDAC	14 <sub>H</sub>	R/W	AFE TXDAC 控制寄存器
AFE_RX_CTRL1	18 <sub>H</sub>	R/W	AFE RX 控制寄存器 1
AFE_RX_CTRL2	1C <sub>H</sub>	R/W	AFE RX 控制寄存器 2
AFE_RX_RCCAL	20 <sub>H</sub>	R/W	AFE RX RCCAL 控制寄存器
AFE_PLLSTCR	24 <sub>H</sub>	R/W	AFE PLL 静态控制寄存器
AFE_PLLCHDIVCR	28 <sub>H</sub>	R/W	AFE PLL 载波分频比配置寄存器
AFE_PLLCR	2C <sub>H</sub>	R/W	AFE PLL 环路控制寄存器
AFE_PLLTPMODCR	30 <sub>H</sub>	R/W	AFE PLL 两点调制控制寄存器

AFE_PLLSPICR	34 <sub>H</sub>	R/W	AFE PLL 环路 Bypass 模式控制寄存器
AFE_DACSPICR	38 <sub>H</sub>	R/W	AFE DAC 模块 Bypass 模式控制寄存器
AFE_GLPFFIFO CR	3C <sub>H</sub>	R/W	AFE Glpf 中 FIFO 控制和状态寄存器
AFE_CTRL_MODE0	40 <sub>H</sub>	R/W	AFE 接口信号控制模式寄存器 0
AFE_CTRL_MODE1	44 <sub>H</sub>	R/W	AFE 接口信号控制模式寄存器 1
AFE_RAMPGEN_CN TL	48 <sub>H</sub>	R/W	AFE RAMPGEN 控制寄存器
AFE_RPLL_DLY	4C <sub>H</sub>	R/W	AFE RPLL_DLY 控制寄存器
AFE_TX_DLY0	50 <sub>H</sub>	R/W	AFE TX_DLY0 控制寄存器
AFE_TX_DLY1	54 <sub>H</sub>	R/W	AFE TX_DLY1 控制寄存器
AFE_RX_DLY0	58 <sub>H</sub>	R/W	AFE RX_DLY0 控制寄存器
AFE_RFDIG_CFG	5C <sub>H</sub>	R/W	AFE RFDIG 配置寄存器
AFE_IOMUX_CTRL	60 <sub>H</sub>	R/W	AFE IOMUX 选择控制寄存器
AFE_DAC_GAIN CTRL	64 <sub>H</sub>	R/W	AFE DAC_GAINCAL 分段校准控制寄存器
AFE_DAC_GAIN CTRL_2M	68 <sub>H</sub>	R/W	AFE DAC_GAINCAL_2M 分段校准控制寄存器
AFE_RESERVED	6C <sub>H</sub>	R/W	AFE 保留扩展寄存器

## 38.5.2 寄存器描述

### 38.5.2.1 AFE接口信号使能寄存器 (AFE\_ENABLE)

AFE 接口信号使能寄存器 (AFE_ENABLE)																															
偏移地址: 00 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_LDOCP	EN_LDOVCO	Reserved	EN_LDOANA	EN_LDOADDA	EN_LDO_PA	Reserved		EN_FILTI	EN_FILTQ	EN_FMTX2P	EN_IF	Reserved	DaPFDEN	DaCPEN	DaPSCntEN	DaLO_Div_En	DaAFCCNTEN	DaGainCalEN	DaVCOEN	EN_RXADC	EN_LNA	EN_MIXER	EN_RXPATH	EN_RXPGA	EN_TXDAC	EN_PA	Reserved			EN_DCOCI	EN_DCOcq

EN_LDOCP	Bits 31	R/W	<b>LDO_CP</b> 软件使能位 0: 关闭 1: 使能
EN_LDOVCO	Bits 30	R/W	<b>LDO_VCO</b> 软件使能位 0: 关闭 1: 使能
Reserved	Bits 29	—	保留
EN_LDOANA	Bits 28	R/W	<b>LDO_ANA</b> 软件使能位 0: 关闭 1: 使能
EN_LDOADDA	Bits 27	R/W	<b>LDO_ADDA</b> 软件使能位 0: 关闭 1: 使能
EN_LDO_PA	Bits 26	R/W	<b>LDO_PA</b> 软件使能位 0: 关闭 1: 使能
EN_VZ2ADC	Bits 25	R/W	将VLPF送至SAR ADC进行采样的控制开关 0: 关闭 1: 工作
Reserved	Bits 24	—	保留
EN_FILTI	Bits 23	R/W	<b>FILT Channel I</b> 软件使能位 0: 关闭 1: 使能
EN_FILTQ	Bits 22	R/W	<b>FILT Channel Q</b> 软件使能位 0: 关闭 1: 使能
EN_FMTX2P	Bits 21	R/W	两点调制 (发送模式下) 软件使能位 0: 关闭 1: 使能
EN_IF	Bits 20	R/W	<b>RX mode</b> 软件控制位

			0: real mode 1: complex mode
Reserved	Bits 19	—	保留
DaPFDEN	Bits 18	R/W	<b>RFPLL PFD软件使能位</b> 0: 关闭 1: 工作
DaCPEN	Bits 17	R/W	<b>RFPLL CP软件使能位</b> 0: 关闭 1: 工作
DaPSCntEN	Bits 16	R/W	<b>RFPLL分频器软件使能位</b> 0: 关闭 1: 工作
DaLO_Div_En	Bits 15	R/W	<b>LO软件使能位</b> 0: 关闭 1: 工作
DaAFCCNTEN	Bits 14	R/W	<b>AFC 计数器软件使能位</b> 0: 关闭 1: 工作
DaGainCalEN	Bits 13	R/W	<b>增益校准软件使能位</b> 0: 关闭 1: 工作
DaVCOEN	Bits 12	R/W	<b>VCO软件使能位</b> 0: 关闭 1: 工作
EN_RXADC	Bits 11	R/W	<b>RX ADC软件使能位</b> 0: 关闭 1: 工作
EN_LNA	Bits 10	R/W	<b>RX LNA软件使能位</b> 0: 关闭 1: 工作
EN_MIXER	Bits 9	R/W	<b>RX MIXER软件使能位</b> 0: 关闭 1: 工作
EN_RXPATH	Bits 8	R/W	<b>RX模式软件使能位</b> 1: RX模式 0: TX模式
EN_RXPGA	Bits 7	R/W	<b>RX PGA软件使能位</b> 0: 关闭 1: 使能
EN_TXDAC	Bits 6	R/W	<b>TX DAC软件使能位</b> 0: 关闭 1: 使能
EN_PA	Bits 5	R/W	<b>TX PA软件使能位</b>

			0: 关闭 1: 使能
Reserved	Bits 4-2	—	保留
EN_DCOC1	Bits 1	R/W	<b>DCO for I way开关控制位</b> 0: 关闭 1: 使能
EN_DCOCQ	Bits 0	R/W	<b>DCO for Q way开关控制位</b> 0: 关闭 1: 使能

### 38.5.2.2 AFE测试使能控制寄存器 (AFE\_TEST\_ENABLE)

AFE 测试使能控制寄存器 (AFE_TEST_ENABLE)																															
偏移地址: 04 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TSTEN_GPIO	TSTEN_RSV			TSTEN_DPLL_VZ		TSTEN_XOSC_CLK		TEST_SEL												

Reserved	Bits 31-12	—	保留
TSTEN_GPIO	Bits 11	R/W	<b>GPIO测试使能控制</b> 0: 关闭 1: 使能 (将TST_P和TST_N信号输到GPIO)
TSTEN_RSV	Bits 10-7	R/W	TSTEN保留
TSTEN_RPLL_VZ	Bits 6	R/W	保留位
TSTEN_XOSC_CLK	Bits 5	R/W	<b>XOSC 时钟测试使能控制位</b> 0: 关闭 1: 使能
TEST_SEL	Bits 4-0	R/W	模拟测试端口控制

### 38.5.2.3 AFE PLL模拟参数控制寄存器 1 (AFE\_PLLANACR1)

AFE PLL 模拟参数控制寄存器 1 (AFE_PLLANACR1)																															
偏移地址: 08 <sub>H</sub>																															
复位值: 00000000_00000001_00000001_00010010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															IBCPOpt			CPloffsetDNEN <12:11>		CPloffsetUPEN		CPCtrlOpt			LDDelayOpt		PFDDelayOpt				

Reserved	Bits 31-17	—	保留
DaIBCPOpt	Bits 16-13	R/W	CP电荷泵偏置电流trimming
DaCPloffsetDNEN	Bits 12-11	R/W	CP的放电offset电流控制位
DaCPloffsetUPEN	Bits 10-9	R/W	CP的充电offset电流控制位
DaPCtrlOpt	Bits 8-5	R/W	CP电荷泵控制参数
DaLDDelayOpt	Bits 4-2	R/W	LD锁定检测电路延迟可调
DaPFDDelayOpt	Bits 1-0	R/W	PFD相位延迟可调

### 38.5.2.4 AFE PLL模拟参数控制寄存器 2 (AFE\_PLLANACR2)

AFE PLL 模拟参数控制寄存器 2 (AFE_PLLANACR2)																															
偏移地址: 0C <sub>H</sub>																															
复位值: 00010000_10001010_11011011_01100010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				VafcTrim			GainCtrlCapOpt	GainCtrlResOpt	IBVCOOpt				VCOVdacOpt			VCOVtuneOpt										VCOAmOpt					

Reserved	Bits 31-29	—	保留
VafcTrim	Bits 28-26	R/W	AFC校准时VCO BIAS偏置电压控制
DaGainCtrlCapOpt	Bits 25	R/W	增益校正滤波电容控制
DaGainCtrlResOpt	Bits 24	R/W	增益校正滤波电阻控制
DaIBVCOOpt	Bits 23-20	R/W	VCO偏置电流trimming
DaVCOVdacOpt	Bits 19-16	R/W	提高KDAC (DAC输出电压引起VCO压控灵敏度) 曲线的线性度的电压控制位
DaVCOVtuneOpt	Bits 15-4	R/W	KVCO曲线提高线性度的电压控制位
DaVCOAmOpt	Bits 3-0	R/W	LC振荡器幅度控制

### 38.5.2.5 AFE PLL模拟参数控制寄存器3 (AFE\_PLLANACR3)

AFE PLL 模拟参数控制寄存器3 (AFE_PLLANACR3)																															
偏移地址: 10 <sub>H</sub>																															
复位值: 00001100_00110000_11000011_00011000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DaLPFCapResOpt															

PLL_RSV	Bits 31-28	R/W	<3:2>保留位 <1> 0: R3维持; 1: R3电阻增加125K <0> 0: R2维持; 1: R2电阻增加47.5K
DaLPFCapResOpt	Bits 27-0	R/W	LPF电容电阻trimming

### 38.5.2.6 AFE TXDAC控制寄存器 (AFE\_TXDAC)

AFE TXDAC 控制寄存器 (AFE_TXDAC)																															
偏移地址: 14 <sub>H</sub>																															
复位值: 00001110_00000000_00000100_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TXDAC_CLK_INV	TXDAC_DC				Reserved										TXDAC_GAIN				Reserved	TXDAC_DATA				NRST_TXDAC					

GFSK_BAND5M6_SEL	Bits 31	R/W	<b>GFSK 双转单带宽选择</b> 0: 2.9MHz 1: 5.8MHz
GFSK1M_AMP_SEL	Bits 30	R/W	<b>GFSK1M幅度选择控制</b> 0: 250K (幅度为双端的1/4) 1: 500K (幅度为双端的1/2)
TXDAC_CLK_INV	Bits 29	R/W	无用 (DAC CLK的取反操作由AFE_RFDIG_CFG的bit10 txdac_clk_inv控制)
TXDAC_DC	Bits 28-24	R/W	发送DAC DC Offset校正 (2's补码)
Reserved	Bits 23-15	—	保留
TXDAC_GAIN	Bits 14-8	R/W	发送DAC可编程增益控制位 +/-1.5dB, 步长0.5dB(2's补码)
Reserved	Bits 7	—	保留
TXDAC_DATA	Bits 6-1	R/W	发送DAC数据 (2's补码)
NRST_TXDAC	Bits 0	R/W	TXDAC复位信号(状态机控制, 软件控制)

38. 5. 2. 7 AFE Rx控制寄存器 1 (AFE\_RX\_CTRL1)

AFE Rx 控制寄存器 1 (AFE_RX_CTRL1)																															
偏移地址: 18 <sub>H</sub>																															
复位值: 10111011_10100000_00000100_00100101 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GC_PA				GC_LNA				GC_MIXER,	FILT_GAIN				RXPGA_GAIN		Reserved	RES_TRIM_LNA		FILT_BAND	PGA_BYPASS	GFSK_INV	FILT_ICTL		Reserved		RXPGA_ICTL		VCM_CTL_MIXER	CUR_CTL_MIXER			

GC_PA	Bits 31-28	R/W	PA增益控制
GC_LNA	Bits 27-24	R/W	LNA增益控制
GC_MIXER	Bits 23	R/W	MIXER增益控制
FILT_GAIN	Bits 22-20	R/W	FILT增益控制; 共8档, 步长6dB(状态机控制, 软件控制) Gain <2:0>    Gain (dB) 3'b000:            5 3'b001:            11 3'b010:            17 3'b011:            23 3'b100:            29 3'b101:            35 3'b110:            41 3'b111:            47
RXPGA_GAIN	Bits 19-18	R/W	<b>RXPGA增益控制</b> 2'b00: 0 2'b01: 6 2'b10: 12 2'b 11: 18 共4档, 步长6dB(动态,测试模式下可控)
Reserved	Bits 17	—	保留
RES_TRIM_LNA	Bits 16-14	R/W	LNA 输入阻抗控制
FILT_BAND	Bits 13	R/W	<b>带宽选择</b> (状态机控制, 软件控制) rx mode: 1'b0 for 1MHz. 1'b1 for 2MHz tx mode: 1'b0 for 1.33MHz. 1'b1 for 2.7MHz
PGA_BYPASS	Bits 12	R/W	PGA Bypass控制位 1: 使能 0: 不使能
GFSK_INV	Bits 11	R/W	GFSK信号反相控制位 1: 反相 0: 保持
FILT_ICTL	Bits 10-9	R/W	FILTER OPA偏置电流控制; 00最小



Reserved	Bits 8-6	—	保留
RXPGA_ICTL	Bits 5-4	R/W	PGA电流档位控制
VCM_CTL_MIXER	Bits 3-2	R/W	MIXER共模电压控制
CUR_CTL_MIXER	Bits 1-0	R/W	MIXER偏置电流控制

### 38.5.2.8 AFE Rx控制寄存器 2 (AFE\_RX\_CTRL2)

AFE Rx 控制寄存器 2 (AFE_RX_CTRL2)																															
偏移地址: 1C <sub>H</sub>																															
复位值: 00000000_00000000_01010000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TRX_RSV				RX_RSV				RCCAL_RSV		FILT_VCM_CTL		RXADC_CLK12M_INV		RXADC_REFSEL		DCOC_I				DCOC_Q							

Reserved	Bits 31-28	—	保留																					
TRX_RSV	Bits 27-24	R/W	Reserved																					
RX_RSV	Bits 23-18	R/W	<p>&lt;2:0&gt;ppf dcoc偏置电流控制位，电流随配置字递减变化，步长1uA,111时电流最小（1uA），电流越大失调校准范围越大；</p> <p>&lt;3&gt;ppf低通模式校准控制位；1: enable；0: disable</p> <p>&lt;4&gt;保留位</p> <p>&lt;5&gt;ECO到REV1（分频器输出占空比配置）</p>																					
RCCAL_RSV	Bits 17-16	R/W	Reserved																					
FILT_VCM_CTL	Bits 15-14	R/W	FILT 输出共模电压调节位																					
RXADC_CLK12M_INV	Bits 13	R/W	<p>RX ADC 12M时钟信号沿选择</p> <p>0: 默认</p>																					
RXADC_REFSEL	Bits 12	R/W	<p>RX ADC参考电压范围选择</p> <p>1: 默认，1.2v（差分范围）；</p> <p>0: 2.4v</p>																					
DCOC_I	Bits 11-6	R/W	<p>I way DCO修正控制位</p> <table border="0"> <tr> <td>&lt;5~0&gt;</td> <td>Inp</td> <td>Dec</td> </tr> <tr> <td>6'b0xxxxx:</td> <td>0</td> <td>0-&gt;31</td> </tr> <tr> <td>6'b111111:</td> <td>250n</td> <td>-1</td> </tr> <tr> <td>6'b111110:</td> <td>500n</td> <td>-2</td> </tr> <tr> <td>6'b111101:</td> <td>750n</td> <td>-3</td> </tr> <tr> <td>.....</td> <td>.....</td> <td>.....</td> </tr> <tr> <td>6'b100000:</td> <td>8uA</td> <td>-32</td> </tr> </table> <p>&lt;5~0&gt; Inn Dec</p>	<5~0>	Inp	Dec	6'b0xxxxx:	0	0->31	6'b111111:	250n	-1	6'b111110:	500n	-2	6'b111101:	750n	-3	.....	.....	.....	6'b100000:	8uA	-32
<5~0>	Inp	Dec																						
6'b0xxxxx:	0	0->31																						
6'b111111:	250n	-1																						
6'b111110:	500n	-2																						
6'b111101:	750n	-3																						
.....	.....	.....																						
6'b100000:	8uA	-32																						

			6'b1xxxxx: 0 0->31 6'b00000: 0 -1 6'b000001: 250n -2 6'b000010: 500n -3 ..... 6'b011111: 7.75uA -32
DCOC_Q	Bits 5-0	R/W	Q way DCO修正控制位 <b>&lt;5~0&gt; Inp Dec</b> 6'b0xxxxx: 0 0->31 6'b111111: 250n -1 6'b111110: 500n -2 6'b111101: 750n -3 ..... 6'b100000: 8uA -32  <b>&lt;5~0&gt; Inn Dec</b> 6'b1xxxxx: 0 0->31 6'b00000: 0 -1 6'b000001: 250n -2 6'b000010: 500n -3 ..... 6'b011111: 7.75uA -32

38. 5. 2. 9 AFE RX RCCAL控制寄存器 (AFE\_RX\_RCCAL)

AFE RX RCCAL 控制寄存器 (AFE_RX_RCCAL)																															
偏移地址: 20 <sub>H</sub>																															
复位值: 00000000_00000000_11001011_00110000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				EN_RCCAL	NRST_RCCAL	RCCAL_FILT_DONE	RCCAL_FILT_RESULT									FILT_RCVAL				RCCAL_TRIM_FILT				AFE_RES_CAL				EN_PLL_RESCAL			

Reserved	Bits 31-30	—	保留
EN_RCCAL	Bits 29	R/W	<b>RC校准使能控制位</b> 0: 关闭 1: 使能
NRST_RCCAL	Bits 28	R/W	RC校准复位信号
RCCAL_FILT_DONE	Bits 27	R	<b>RC校准结束标志位</b>
RCCAL_FILT_RESULT	Bits 26-17	R	<b>RC校准计数器输出</b>
FILT_RCVAL	Bits 16-12	R/W	<b>FILTER RC校准输入</b>

			默认由RC calibration电路校准得到，手动输入时： 12: 典型值 0: 带宽最大 15: 带宽最小
RCCAL_TRIM_FILT	Bits 11-6	R/W	rc校准电容控制位
AFE_RES_CAL	Bits 5-1	R/W	<b>PLL中电阻校准控制</b> 需要在机台上进行校准，需在配置区
EN_PLL_RESCAL	Bits 0	R/W	<b>PLL电阻校准使能控制位</b> 0: 关闭 1: 校准使能

### 38.5.2.10 AFE PLL静态控制寄存器 (AFE\_PLLSTCR)

AFE PLL 静态控制寄存器 (AFE_PLLSTCR)																															
偏移地址: 24 <sub>H</sub>																															
复位值: 00000000_00101000_11010001_10000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											VcoDelay_Afc			VcoDelay_Afc									GlpfModeSel	DacCtrlMode	DacEnLogicMode	NotchENn	OSSDMENn	DitherSel	DitherPower	DitherENn	AfcCntTime

Reserved	Bits 31-22	—	保留
VcoDelay_Afc	Bits 21-18	R/W	需要执行AFC时的VCO使能延时等待时间 (单位 us)
VcoDelay_NoAfc	Bits 17-12	R/W	—
GlpfModeSel	Bits 11	R/W	<b>Gauss滤波器工作速率选择</b> 0: 1Mbps的数据速率 1: 2Mbps的数据速率
DacCtrlMode	Bits 10	R/W	<b>DAC控制模式</b> 0: 逻辑控制模式 1: 逻辑Bypass模式
DacEnLogicMode	Bits 9	R/W	<b>增益校正期间DAC使能模式控制</b> 0: GainCtrl和PllLock同时为高则使能DAC 1: GainCtrl为高即使能DAC
NotchENn	Bits 8	R/W	<b>Notch滤波器使能信号</b> 0: 使能 1: 不使能
OSSDMENn	Bits 7	R/W	<b>Offset SDM使能信号</b> 0: 使能 1: 不使能

DitherSel	Bits 6-5	R/W	<b>Dither类型选择信号</b> 00: PN Dither 01: Preset Dither 10: Period $\pm 1$ Sequence Dither 11: 无效
DitherPower	Bits 4-3	R/W	<b>Dither能量控制</b> 00: 1 01: 3 10: 7 11: 31
DitherENn	Bit2	R/W	<b>Dither功能使能信号</b> 0: 使能 1: 不使能
AfcCntTime	Bits 1-0	R/W	<b>AFC 算法小数分频比精度控制</b> 00: 保留 3bit 小数分频比 01: 保留 4bit 小数分频比 10: 保留 5bit 小数分频比 11: 保留6bit小数分频比

### 38. 5. 2. 11 AFE PLL载波分频比配置寄存器 (AFE\_PLLCHDIVCR)

AFE PLL 载波分频比配置寄存器 (AFE_PLLCHDIVCR)																																
偏移地址: 28 <sub>H</sub>																																
复位值: 00000100_11000100_00000000_00000000 <sub>B</sub>																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				PIIDivNSrcSel	PIIDivNInt							PIIDivNFrac																				

Reserved	Bits 31-28	—	保留
PIIDivNSrcSel	Bits 27	R/W	<b>PLL环路载波分频比来源选择</b> 0: 来自BLE Baseband 1: 来自软件配置的{PIIDivNInt, PIIDivNFrac}
PIIDivNInt	Bits 26-20	R/W	<b>PLL载波整数分频比控制字</b> 如, 若要实现75.25的分频比, 则75即整数分频控制字
PIIDivNFrac	Bits 19-0	R/W	<b>PLL载波小数分频比控制字</b> 如, 若要实现75.25的小数分频比, 则0.25*(2 <sup>20</sup> )即得到载波对应的小数分频控制字

### 38.5.2.12 AFE PLL环路控制寄存器 (AFE\_PLLCR)

AFE PLL 环路控制寄存器 (AFE_PLLCR)																															
偏移地址: 2C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLLockedFlag		AfcFinish	Reserved				AfcResultMinus										AfcResultPlus										HDAfcCtrl	SoftPIIEn		PIIEnMode	

PLLLockedFlag	Bits 31	R	<b>Pll环路锁定状态标志</b> 0: Pll环路未锁定; 1: Pll环路锁定
AfcFinish	Bits 30	R	<b>Afc算法完成状态标志</b> 0: Afc算法正在执行, 未完成 1: Afc算法完成 Afc算法完成后, 可通过读取bit[25:4]来获得最终的AFC结果
Reserved	Bits 29-26	—	保留位
AfcResultMinus	Bits 25-15	R/W	<b>AFC电容阵列控制字</b> 注: 写操作时, 直接配置到该寄存器中 (并在软件模式下作用于VCO); 读操作时, 读到的结果为Afc算法计算结果
AfcResultPlus	Bits 14-4	R/W	<b>AFC电容阵列控制字</b> 注: 写操作时, 直接配置到该寄存器中 (并在软件模式下作用于VCO); 读操作时, 读到的结果为Afc算法计算结果;
HDAfcCtrl	Bits 3	R/W	<b>AFC控制位</b> 0: PLL使能处于硬件模式时, 执行AFC时的VCO使能延时等待时间, 根据频点是否变化选择VcoDelay_Afc或VcoDelay_NoAfc; 1: PLL使能处于硬件模式时, 执行AFC时的VCO使能延时等待时间, 固定选择VcoDelay_Afc;
SoftPIIEn	Bits 2-1	R/W	<b>软件模式下PLL使能控制</b> 00: 关闭AFC及PLL 01: 不进行AFC, 直接执行PLL锁定 10: 只进行AFC, 不执行PLL锁定 11: 进行AFC, 并执行PLL锁定过程
PIIEnMode	Bits 0	R/W	<b>PLL使能控制模式</b> 0: 硬件模式, 根据TX/RX自动启动Pll 1: 软件模式, 具体的Pll动作由SoftPIIEn决定

38. 5. 2. 13 AFE PLL两点调制控制寄存器 (AFE\_PLLTPMODCR)

AFE PLL 两点调制控制寄存器 (AFE_PLLTPMODCR)																															
偏移地址: 30 <sub>H</sub>																															
复位值: 00000010_01110011_00110100_11001100 <sub>b</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GainCtrlAlg			GainCtrlMode	TxFreqOffset	DlyCal_Dac				DlyCal_Sdm				PIIGainCtrlEn	PIIFreqCal						PIIGainCtrlDataCnt				PIIGainCtrlCarCnt			

Reserved	Bits 31-27	—	保留
GainCtrlAlg	Bits 26	R/W	<b>增益校正算法选择</b> 0: 采用二分法模式 1: 采用+1/-1模式
GainCtrlMode	Bits 25	R/W	<b>增益校正电路模式选择</b> 0: GPADC采样校准模式 1: 直接采样模式;
TxFreqOffset	Bits 24	R/W	<b>两点调制发射数据最大频偏控制</b> 0: 实现±250KHz频偏 1: 实现±500KHz频偏
DlyCal_Dac	Bits 23-20	R/W	两点调制DAC通路延时校正控制字, 可配置为1~12
DlyCal_Sdm	Bits 19-16	R/W	两点调制SDM通路延时校正控制字, 可配置为0~12
PIIGainCtrlEn	Bits 15	R/W	<b>两点调制增益校正使能寄存器</b> 0: 增益校正结束 1: 增益校正过程中 软件写1启动增益校正, 硬件在校正结束后自动清零
PIIFreqCal	Bits 14-8	R	两点调制增益校正结果, 可以在增益校正结束后读取校正结果
PIIGainCtrlDataCnt	Bits 7-4	R/W	增益校正过程中数据稳定时间, 为 (PIIGainCtrlDataCnt+4) us
PIIGainCtrlCarCnt	Bits 3-0	R/W	增益校正过程中载波稳定时间, 为 (PIIGainCtrlCarCnt +4) us

### 38. 5. 2. 14 AFE PLL环路Bypass模式控制寄存器 (AFE\_PLLSPICR)

AFE PLL 环路 Bypass 模式控制寄存器 (AFE_PLLSPICR)																																		
偏移地址: 34 <sub>H</sub>																																		
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
SpiPIICtrlMode		Reserved		SpiPIICapMinus														SpiPIICapPlus										SpiPIISDMEn	SpiPIIPscEn	SpiPIILoopOpen	SpiPIIPFDEn	SpiPIICPEn	SpiPIIVCOEn	SpiPIIPre2En

SpiPIICtrlMode	Bits 31	R/W	<b>Pll环路控制模式</b> 0: 逻辑联动控制模式 1: Bypass控制模式
Reserved	Bits 30-29	—	保留位
SpiPIICapMinus	Bits 28-18	R/W	Byapss模式下VCO电容阵列负端控制字
SpiPIICapPlus	Bits 17-7	R/W	Byapss模式下VCO电容阵列正端控制字
SpiPIISDMEn	Bits 6	R/W	<b>Byapss模式下SDM模块使能控制</b> 0: 不使能 1: 使能
SpiPIIPscEn	Bits 5	R/W	<b>Byapss模式下Psc模块使能控制</b> 0: 不使能 1: 使能
SpiPIILoopOpen	Bits 4	R/W	<b>Byapss模式下Pll环路开环控制</b> 0: Pll不开环 1: Pll开环
SpiPIIPFDEn	Bits 3	R/W	<b>Byapss模式下PFD使能控制</b> 0: 不使能 1: 使能
SpiPIICPEn	Bits 2	R/W	<b>Byapss模式下CP使能控制</b> 0: 不使能 1: 使能
SpiPIIVCOEn	Bits 1	R/W	<b>Byapss模式下VCO使能控制</b> 0: 不使能 1: 使能
SpiPIIPre2En	Bits 0	R/W	<b>Byapss模式下Div2使能控制</b> 0: 不使能 1: 使能

### 38.5.2.15 AFE DAC模块Bypass模式控制寄存器 (AFE\_DACSPICR)

AFE DAC 模块 Bypass 模式控制寄存器 (AFE_DACSPICR)																															
偏移地址: 38 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_01101000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															SpiDacFrqCal										SpiDacEn						

Reserved	Bits 31-8	—	保留
SpiDacFrqCal	Bits 7-1	R/W	Bypass模式下DAC增益校正控制字
SpiDacEn	Bits 0	R/W	<b>Bypass模式下DAC使能控制</b> 0: 不使能 1: 使能

### 38.5.2.16 AFE Glpf中FIFO控制和状态寄存器 (AFE\_GLPFFIFO CR)

AFE Glpf 中 FIFO 控制和状态寄存器 (AFE_GLPFFIFO CR)																															
偏移地址: 3C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000010 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											FlpfFifoClr	FiFoDatNum							FifoUnderFlow	FifoOverFlow	FifoRdEmpty	FifoWrFull									

Reserved	Bits 31-12	—	保留
FlpfFifoClr	Bits 11	R/W	<b>FIFO复位</b> 软件写1, 触发清空FIFO, 后硬件自动清0
FiFoDatNum	Bits 10-4	R	<b>FIFO剩余数据量</b>
FifoUnderFlow	Bits 3	R	FIFO向下溢出标志
FifoOverFlow	Bits 2	R	FIFO向上溢出标志
FifoRdEmpty	Bits 1	R	<b>FIFO空指示</b> 0: FIFO非空 1: FIFO空
FifoWrFull	Bits 0	R	<b>FIFO满指示</b> 0: FIFO非满 1: FIFO满



### 38.5.2.17 AFE接口信号控制模式寄存器0 (AFE\_CTRL\_MODE0)

AFE 接口信号控制模式寄存器0 (AFE_CTRL_MODE0)																															
偏移地址: 40 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
afe_ctrl_mode0																															

afe_ctrl_mode0	Bits 31-0	R/W	<p><b>AFE接口信号控制模式</b></p> <p>0: 逻辑联动控制模式</p> <p>1: Bypass测试模式</p> <p>映射关系如下:</p> <p>[0]en_idocp</p> <p>[1]en_idovco</p> <p>[2]en_idoana</p> <p>[3]en_idoadda</p> <p>[4]en_ido_pa</p> <p>[5]en_filtl</p> <p>[6]en_filtq</p> <p>[7]en_fmrx2p</p> <p>[8]en_if</p> <p>[9]dapfdn</p> <p>[10]dacpen</p> <p>[11]dapscnten</p> <p>[12]dalo_div_en</p> <p>[13]daafcnten</p> <p>[14]dagaincalen</p> <p>[15]davcoen</p> <p>[16]en_rxadc</p> <p>[17]en_lna</p> <p>[18]en_mixer</p> <p>[19]en_rxpath</p> <p>[20]en_rxpga</p> <p>[21]en_txdac</p> <p>[22]en_pa</p> <p>[23]nrst_txdac</p> <p>[24]gc_pa</p> <p>[25]gc_lna</p> <p>[26]gc_mixer</p> <p>[27]filt_gain</p>
----------------	-----------	-----	--

			[28]rxpga_gain [29]txdac_gain [30]txdac_data [31]gfsk1m_amp_sel
--	--	--	--

### 38.5.2.18 AFE接口信号控制模式寄存器 1 (AFE\_CTRL\_MODE1)

AFE 接口信号控制模式寄存器 1 (AFE_CTRL_MODE1)																															
偏移地址: 44 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																afe_ctrl_mode1															

Reserved	Bits 31-16	—	保留
afe_ctrl_mode1	Bits 15-0	R/W	<b>AFE接口信号控制模式</b> 0: 逻辑联动控制模式 1: Bypass测试模式 映射关系如下: [0]filt_band [1]dcoci [2]dcocq [3]glpfmodelsel [4]plltxfreqoffset

### 38.5.2.19 AFE RAMPGEN控制寄存器 (AFE\_RAMPGEN\_CNTL)

AFE RAMPGEN 控制寄存器 (AFE_RAMPGEN_CNTL)																															
偏移地址: 48 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_11111011 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ramp_gain_step								ramp_gain_num							

Reserved	Bits 31-12	—	保留
ramp_gain_step	Bits 11-4	R/W	RAMPUP/RAMDOWN的步进 0.5~2us可配置 时钟32M 例如: 16:0.5us 32:1us 64:2us ... 默认值:

ramp_gain_num	Bits 3-0	R/W	RAMPUP/RAMDOWN的次数配置位, 支持 0~15
---------------	----------	-----	-------------------------------

### 38.5.2.20 AFE RPLL\_DLY控制寄存器 (AFE\_RPLL\_DLY)

AFE RPLL_DLY 控制寄存器 (AFE_RPLL_DLY)																															
偏移地址: 4C <sub>H</sub>																															
复位值: 00000000_00000000_00000101_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																rpll_dly															

Reserved	Bits 31-16	—	保留
rpll_dly	Bits 15-0	R/W	10us~90us, 默认40us 默认值: 40*32 = 0x500

### 38.5.2.21 AFE TX\_DLY0 控制寄存器 (AFE\_TX\_DLY0)

AFE TX_DLY0 控制寄存器 (AFE_TX_DLY0)																															
偏移地址: 50 <sub>H</sub>																															
复位值: 00000000_10000000_00000000_01000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tx_path_dly																tx_rf_dly															

tx_path_dly	Bits 31-16	R/W	Tx Path Delay
tx_rf_dly	Bits 15-0	R/W	2us~12us, 默认2us    64~384 默认值: 2*32=0x40

### 38.5.2.22 AFE TX\_DLY1 控制寄存器 (AFE\_TX\_DLY1)

AFE TX_DLY1 控制寄存器 (AFE_TX_DLY1)																															
偏移地址: 54 <sub>H</sub>																															
复位值: 00000000_01000000_00000000_00010000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ramp_up_dly																ramp_dw_dly															

ramp_up_dly	Bits 31-16	R/W	1us~5us, 默认2us 32~160 默认值:2*32=0x40
ramp_dw_dly	Bits 15-0	R/W	0.5~2us,: 默认 16~64 默认值

### 38.5.2.23 AFE RX\_DLY0 控制寄存器 (AFE\_RX\_DLY0)

AFE R X_DLY0 控制寄存器 (AFE_RX_DLY0)																															
偏移地址: 58 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_01000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																rx_ana_dly															

Reserved	Bits 31-16	—	保留
rx_ana_dly	Bits 15-0	R/W	2us~12us,默认2us 32~160 默认值:2*32=0x40

### 38.5.2.24 AFE RFDIG配置寄存器 (AFE\_RFDIG\_CFG)

AFE RFDIG 配置寄存器 (AFE_RFDIG_CFG)																																													
偏移地址: 5C <sub>H</sub>																																													
复位值: 00111100_00000111_00010001_01010110 <sub>B</sub>																																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
Reserved		filt_band_tx				Reserved		lo_txdac_data_en	lomGaussTxEn		lpf_sel	SelXO16M		PmuPIIDaDacEn		Reserved		ind_sel_mode				Reserved		BBRx_Lo_sel		pll_lock_mode		txdac_data_inv		txdac_clk_inv		txdacclk_sel		filt_band_val				gfsk1m_amp_val		en_if_val		en_rxpga_val		en_ldopa_val	

Reserved	Bits 31-30	—	保留
filt_band_tx	Bits 29-26	R/W	<b>FILT BAND TX配置参数</b> [0]: 发送状态, 1M速率的配置值 [1]: 发送状态, 2M速率的配置值 [2]: 发送状态, 125k速率的配置值 [3]: 发送状态, 500k速率的配置值
Reserved	Bits 25	—	保留
lo_txdac_data_en	Bits 24	R/W	
lomGaussTxEn	Bits 23	R/W	<b>GLPF输入数据选择</b> 0: 来自BB 1: 来自IO
lpf_sel	Bits 22	R/W	<b>GLPF选择</b> 0: LPF1 1: LPF2 (CEVA)
SelXO16M	Bits 21	R/W	<b>AFCcntTime</b> 计时时钟

			0: 32M 1: 16M
PmuPIIDaDacEn	Bits 20	R/W	保留
Reserved	Bits 19	—	保留
ind_sel_mode	Bits 18-16	R/W	00: DaPFDEN启动状态机进入等待rpII锁定状态 01: DaCPEN启动状态机进入等待rpII锁定状态 10: DaPSCntEN启动状态机进入等待rpII锁定状态
dac_sdm_data_dis	Bits 15	R/W	0: 打开低通通道的TxDataFrac数据 1: 关闭低通通道的TxDataFrac数据;
BBRx_Lo_sel	Bit 14	R/W	接收频率设置: 0: -2M 1: +2M
pll_lock_mode	Bits 13-12	R/W	状态机判定rpII锁定模式 00: 收到rpII锁定信号有效且达到rpII锁定超时门限 01: 达到rpII锁定超时门限 10: 收到rpII锁定信号 11: 收到rpII锁定信号或达到rpII锁定超时门限
txdac_data_inv	Bits 11	R/W	<b>TXDAC输入数据取反控制位</b> 0: 不取反 1: 取反
txdac_clk_inv	Bits 10	R/W	<b>TXDAC时钟反向控制位</b> 0: 不取反 1: 取反
txdacclk_sel	Bits 9	R/W	<b>Txdac时钟选择</b> 0: 选择clksdm 1: 选择clkosc32m
filt_band_val	Bits 8-5	R/W	<b>FILT BAND RX配置参数</b> [0]: 接收状态, 1M速率的配置值 [1]: 接收状态, 2M速率的配置值 [2]: 接收状态, 125k速率的配置值 [3]: 接收状态, 500k速率的配置值
gfsk1m_amp_val	Bits 4-3	R/W	<b>gfsk1m_amp_val 配置参数</b> [0]: 发送状态, 非2M速率的配置值 [1]: 发送状态, 2M速率的配置值
en_if_val	Bits 2	R/W	en_if_val接收模式下配置值
en_rxpga_val	Bits 1	R/W	en_rxpga_val接收模式下配置值
en_ldopa_val	Bits 0	R/W	en_ldopa_val接收模式下配置值

### 38.5.2.25 AFE IOMUX选择控制寄存器 (AFE\_IOMUX\_CTRL)

AFE IOMUX 选择控制寄存器 (AFE_IOMUX_CTRL)																															
偏移地址: 60 <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00000000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rf_digital_dbg								io_mux1_ctrl								io_mux0_ctrl							

Reserved	Bits 31-24	—	保留
rf_digital_dbg	Bits23-16	R	RF Digital Interface Debug
io_mux1_ctrl	Bits 15-8	R/W	iomux选择控制寄存器1
io_mux0_ctrl	Bits 7-0	R/W	iomux选择控制寄存器0

### 38.5.2.26 AFE DAC\_GAINCAL分段校准控制寄存器 (AFE\_DAC\_GAIN\_CTRL)

AFE DAC_GAINCAL 分段校准控制寄存器 (AFE_DAC_GAIN_CTRL)																															
偏移地址: 64 <sub>H</sub>																															
复位值: 00000100_00000100_00000100_00000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	dac_gaincal3							Reserved	dac_gaincal2							Reserved	dac_gaincal1							Reserved	dac_gaincal0						

Reserved	Bits31	—	保留
dac_gaincal3	Bits 30-24	R/W	Ch39,Ch28-36 增益校准
Reserved	Bits23	—	保留
dac_gaincal2	Bits 22-16	R/W	Ch18-27 增益校准
Reserved	Bits15	—	保留
dac_gaincal1	Bits 14-8	R/W	Ch38,Ch9-17 增益校准值
Reserved	Bits7	—	保留
dac_gaincal0	Bits 6-0	R/W	Ch37,Ch0-8 增益校准值

### 38.5.2.27 AFE DAC\_GAINCAL\_2M分段校准控制寄存器 (AFE\_DAC\_GAIN\_CTRL\_2M)

AFE DAC_GAINCAL_2M 分段校准控制寄存器 (AFE_DAC_GAIN_CTRL_2M)																															
偏移地址: 68 <sub>H</sub>																															
复位值: 00000100_00000100_00000100_00000100 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				dac_gaincal3_2m				Reserved				dac_gaincal2_2m				Reserved				dac_gaincal1_2m				Reserved				dac_gaincal0_2m			

Reserved	Bits31	—	保留
dac_gaincal3_2m	Bits 30-24	R/W	Ch39,Ch28-36 增益校准
Reserved	Bits23	—	保留
dac_gaincal2_2m	Bits 22-16	R/W	Ch18-27 增益校准
Reserved	Bits15	—	保留
dac_gaincal1_2m	Bits 14-8	R/W	Ch38,Ch9-17 增益校准值
Reserved	Bits7	—	保留
dac_gaincal0_2m	Bits 6-0	R/W	Ch37,Ch0-8 增益校准值

### 38.5.2.28 AFE保留扩展寄存器 (AFE\_RESERVED)

AFE 保留扩展寄存器 (AFE_RESERVED)																															
偏移地址: 6C <sub>H</sub>																															
复位值: 00000000_00000000_00000000_00001000 <sub>B</sub>																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCCAL_RSVA				BG_RSVA				TX_RSVA				RX_RSVA				DPLL_RSVA				PLL_RSVA				TXFILT_RCVAL							

RCCAL_RSVA	Bits 31-26	R/W	保留位
BG_RSVA	Bits 25-24	R/W	保留位
TX_RSVA	Bits 23-16	R/W	保留位
RX_RSVA	Bits 15-10	R/W	保留位
DPLL_RSVA	Bits 9-8	R/W	保留位
PLL_RSVA	Bits 7-5	R/W	保留位
TXFILT_RCVAL	Bits 4-0	R/W	TX时FILTER RC校准输入: 默认由RC calibration 电路校准得到; 手动输入时, 12, 典型值; 0, 带宽最大; 15, 带宽最小;

## 第39章 Flash信息区

### 39.1 概述

---

Flash 信息区用来存储芯片的只读信息和配置信息。

用户在程序中只能对只读信息和配置信息进行读操作。可以使用芯片烧录工具对配置信息进行修改，但无法改变只读信息。

### 39.2 特性

---

- ◇ Flash 信息区存储的配置信息包括
  - ◇ 芯片配置字
  - ◇ 写保护区域配置字
  - ◇ 数据区配置字
  - ◇ 全局读保护配置字
  - ◇ 私有代码读出保护区域配置字
  - ◇ 用户程序校验码



### 39.3 功能描述

#### 39.3.1 Flash信息区只读信息

Flash 信息区的基地址为 0x0008\_0000，可以字读取。

#### 39.3.2 Flash信息区配置信息

Flash 信息区的配置信息在芯片程序运行前生效。用户程序不能对其进行修改。

Flash 信息区可以字读取。只能使用芯片烧录工具对配置信息进行修改。

##### 39.3.2.1 芯片配置字CFG\_WORD

芯片的很多特性需要配置，而且这些配置需要在程序运行前生效，这就需要芯片配置字。

寄存器名称	芯片配置字 0 (CFG_WORD0)	
地址偏移	0000_1000 <sub>H</sub>	
低 16 位复位值	11101111_11110111 <sub>B</sub>	
—	Bit 31-16	Bit 15-0 取反值 (不满足取反时 Bit 15-0 强制为默认值)
BOR09VS	Bit 15-14	<b>BOR09 电压点选择位</b> 00: 0.75V 01: 0.65V 10: 0.55V 11: 由软件控制
WWDTEN	Bit 13	<b>WWDT 使能位</b> 0: WWDT 禁止工作 1: WWDT 可工作 注: 该位置 0 时, 软件使能 WWDT 无效, WWDT 禁止工作; 该位置 1 时, 可通过软件使能 WWDT 工作。
BOOT	Bit 12	<b>Flash 启动地址选择位</b> 0: 0x0000_0000 1: 0x0000_0000 或 0x0007_F000
BOR33VS	Bit 11-10	<b>BOR33 电压点选择位</b> 00: 2.5V 01: 2.0V 10: 1.5V 11: 由软件控制
PWRTEN	Bit 9	<b>上电延时使能位</b> 0: 禁止 1: 使能
XTAL	Bit 8	<b>外部振荡器模式选择位</b> 0: 1~8MHz 1: 8~24MHz
—	Bit 7-4	—
IWDTEN	Bit 3	<b>IWDT 使能位</b> 0: 由软件使能

寄存器名称	芯片配置字 0 (CFG_WORD0)	
—	—	1: 硬件强制使能 注: 硬件强制使能后, 软件无法关闭; 中断强制使能, 软件无法关闭; 复位强制使能, 软件无法关闭; 时钟源固定为 ULRC, 软件无法切换。
—	Bit 2	—
LOSMEN	Bit 1	<b>LOSC 安全管理使能位</b> 0: 禁止 1: 使能 注: 硬件使能后, 软件无法关闭
LOSCEN	Bit 0	<b>LOSC 使能位</b> 0: 由软件使能 1: 硬件强制使能 注: 硬件使能后, 软件无法关闭

注 1: 复位值是指芯片配置字被从信息区读出之前的值。

### 39.3.2.2 写保护区域配置字CFG\_WRPT

芯片支持 2 个保护区域, 分别通过 CFG\_WRP0 和 CFG\_WRP1 来配置。设置为写保护的区域, 用户程序将不能通过 IAP 对其进行擦写。

寄存器名称	写保护区域 x 配置字 (CFG_WRPx) (x=0..1)	
地址偏移	0000_1004 <sub>H</sub> ~ 0000_1008 <sub>H</sub>	
低 16 位复位值	11111111_11111111 <sub>B</sub>	
—	Bit 31-16	<b>Bit 15-0 取反值 (不满足取反时 Bit 15-0 强制为默认值)</b>
ENB	Bit 15	<b>FLASH 写保护区 0/1 使能信号</b> 1'b1: 写保护禁止 其它: 写保护使能
END	Bit 14-8	<b>FLASH 写保护区 0/1, 结束页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
—	Bit 7	保留
START	Bit 6-0	<b>FLASH 写保护区 0/1, 起始页选择位</b> 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

### 39.3.2.3 数据区配置字CFG\_DAFLS

芯片支持 1 个数据区域，通过 CFG\_DAFLS 来配置。通过其配置可以将 Flash 空间分为程序区和数据区。程序区和数据区的 IAP 擦写命令不同。

寄存器名称	数据 Flash 配置字 (CFG_DAFLS)	
地址偏移	0000_100C <sub>H</sub>	
低 16 位复位值	11111111_11111111 <sub>B</sub>	
—	Bit 31-16	Bit 15-0 取反值 (不满足取反时 Bit 15-0 强制为默认值)
ENB	Bit 15	数据 eFlash 使能信号 1'b0: 数据 FLASH 有效 1'b1: 数据 FLASH 无效
END	Bit 14-8	数据 Flash 结束页配置位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
—	Bit 7	—
START	Bit 6-0	数据 Flash 起始页配置位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

### 39.3.2.4 晶振频率校准配置字CFG\_XTAL

寄存器名称	晶振频率校准配置字 (CFG_XTAL)	
地址偏移	0000_1010 <sub>H</sub>	
低 16 位复位值	11111111_11111111 <sub>B</sub>	
—	Bit 31-16	Bit 15-0 取反值 (不满足取反时 Bit 15-0 强制为默认值)
—	Bit 15-8	—
CAPCAL	Bit 7-0	XOSC 晶振频率调校值: 0000_0000: 最低频率 ..... 1111_1111: 最高频率

### 39.3.2.5 全局读保护配置字CFG\_GBRDP

全局读保护分为 Level0~2 三个等级，Level0 为不保护，Level1 和 Level2 的详细说明请参考“存储器系统控制 (MSC)”中“Flash 全局读保护”章节的描述。

寄存器名称		全局读保护配置字 (CFG_GBRDP)	
地址偏移	0000_1018 <sub>H</sub>		
低 16 位复位值	1111_1111_1111_1111 <sub>B</sub>		
—	Bit 31-16	Bit 15-0 取反值 (不满足取反时 Bit 15-0 强制为默认值)	
—	Bit 15-8	—	
GBRDP	Bit 7-0	全局读保护控制位 0xFF: 全局读保护级别为 0 (不加密) 0x00: 全局读保护级别为 2 其它: 全局读保护级别为 1	

### 39.3.2.6 用户程序校验码CHKSUM

烧录工具将用户程序校验码写入此区域。

寄存器名称		用户程序校验码 (CHKSUM)	
地址偏移	0000_1FE8 <sub>H</sub>		
CHKSUM	Bit 31-0	用户程序校验码	

寄存器名称		用户程序校验码反码 (CHKSUMN)	
地址偏移	0000_1FEC <sub>H</sub>		
CHKSUMN	Bit 31-0	用户程序校验码反码	

### 39.3.2.7 私有代码读出保护区域配置字CFG\_PCROP

芯片支持 1 个私有代码读出保护区域 CFG\_PCROP0。被设置为私有代码读出保护区域后，其中的代码可以被运行但无法读出。详细参考“存储器系统控制（MSC）”中“Flash 私有代码读出保护区域”章节的描述。

寄存器名称	私有代码读出保护区域配置字（CFG_PCROP0）	
地址偏移	0000_2008 <sub>H</sub>	
低 16 位复位值	1111_1111_1111_1111 <sub>B</sub>	
—	Bit 31-16	—
ENB	Bit 15	主程序保护区使能信号 1'b0: 分区 0 加密使能 1'b1: 分区 0 加密禁止
END	Bit 14-8	主程序保护区，结束页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页
—	Bit 7	—
START	Bit 6-0	主程序保护区，起始页选择位 0x00: 第 0 页 0x01: 第 1 页 0x02: 第 2 页 0x03: 第 3 页 ..... 0x7F: 第 127 页

## 附录1 ARM Cortex-M3 附录

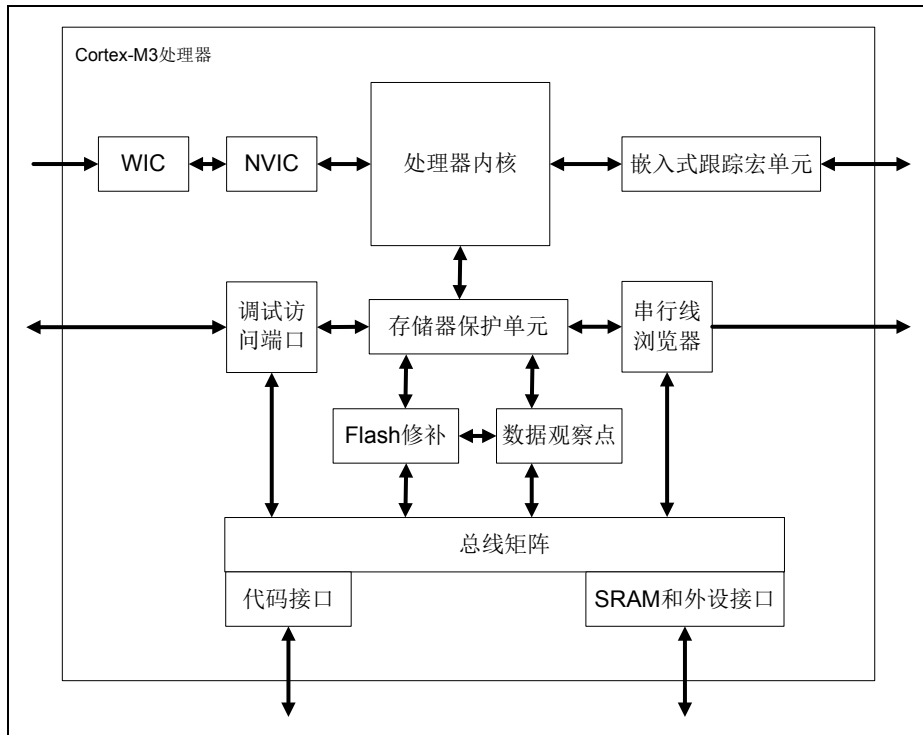
### 附录1.1 ARM Cortex-M3 用户指南：简介

ARM Limited 提供本附录中的资料，并包括在含 Cortex-M3 CPU 的设备的用户手册中。为了反映只针对 ES32 微控制器的实现选项以及其他的差异，内容有轻微的改动。

#### 附录1.1.1 关于处理器和内核外设

Cortex-M3 处理器是一种为微控制器市场设计的高性能 32 位处理器。它为开发人员提供了显著的益处，包括：

- ◇ 杰出的处理性能，以及快速的中断处理能力；
- ◇ 大量断点和跟踪能力，增强了系统调试能力；
- ◇ 高效的处理器内核、系统和存储器；
- ◇ 集成睡眠模式，具有超低功耗；
- ◇ 有可选的集成存储器保护单元（MPU），保证平台的安全性。



附录图 39-1 Cortex-M3 的典型应用

Cortex-M3 处理器建立在一个高性能处理器内核上，采用一个 3 级流水线哈佛架构，非常适合于高要求的嵌入式应用。该处理器使用了一个高效指令集和广泛优化的设计，具备了优异的能效，提供高端的处理硬件，包括单周期 32x32 乘法器和专用的硬件除法器。

为了降低设备的设计成本，Cortex-M3 处理器实现了紧密耦合的系统部件，在降低处理器面积的同时，还显著提高了中断处理和系统调试能力。Cortex-M3 处理器实现了一个 Thumb 指令集版本，从而确保了高代码密度，降低了对程序存储器的要求。Cortex-M3 指令集具有 8 位和 16 位微控制器的高代码密度，提供了现代 32 位架构理想的优异性能。

Cortex-M3 处理器紧密地集成了一个可配置的可嵌套向量中断控制器 (NVIC)，提供了业界领先的中断性能。NVIC 包括一个不可屏蔽中断 (NMI)，并提供高达 256 个中断优先级。处理器内核和 NVIC 的紧密耦合实现了中断服务程序 (ISR) 的快速执行，大大缩短了中断延迟。其实现方式是通过寄存器的硬件堆栈，以及挂起多个加载 (load-multiple) 和多个存储 (store-multiple) 操作的能力。中断处理程序不需要任何的汇编程序插桩 (stub)，消除了所有来自 ISR 的代码开销。末尾连锁 (Tail-chaining) 优化也显著降低了从一个 ISR 切换到另一个 ISR 时所产生的开销。

为了优化低功耗设计，NVIC 集成了睡眠模式，包括一个深度睡眠功能，能够使整个设备快速进入掉电模式。

ES32 还支持其他的低功耗模式，详细内容参见“电源管理及低功耗模式”章节。

### 附录1.1.1.1 系统级接口

Cortex-M3 处理器采用 AMBA 技术提供了多个接口，以实现高速、低等待时间的存储器访问。它支持非对齐的数据访问，实现了原子位操作 (atomic bit manipulation)，能够实现更快的外设控制、系统自旋锁 (spinlock)，以及线程安全的 Boolean 数据处理。

Cortex-M3 处理器有一个可选的存储器保护单元 (MPU)，提供细粒度的存储器控制，使应用程序能够实现安全特权级别，在不同任务基础上划分代码、数据和堆栈。在很多嵌入式应用 (如汽车) 中，这类需求正变得越来越重要。ES32 中包括了 MPU。

### 附录1.1.1.2 集成的可配置调试功能

Cortex-M3 处理器实现了完全的硬件调试解决方案。通过传统的 JTAG 端口或一个双管脚的串行线调试 (SWD) 端口提供了对处理器和存储器的系统高可视性，SWD 非常适用于微控制器和其它小封装设备。MCU 厂商决定了调试特性的配置，因此它在不同设备和产品系列上可能存在差异。

在系统跟踪方面，除数据观察点和一个分析单元外，处理器中还集成了一个指令跟踪宏单元 (ITM)。为了能够简单而高效地分析这些生成的系统事件，串行线浏览器 (SWV) 可通过一个管脚输出一个软件生成消息、数据跟踪和数据分析信息流。

可选的嵌入式跟踪宏单元 (ETM) 的面积远小于传统跟踪单元的面积，但提供了无与伦比的指令跟踪捕获能力，从而使很多低成本 MCU 第一次实现了完全的指令跟踪。

### 附录1.1.1.3 Cortex-M3 处理器特性和优点汇总

- ◇ 系统外设的紧密集成不但节约了空间还降低了开发成本；
- ◇ Thumb 指令集兼备了高代码密度和 32 位性能；
- ◇ 具备代码修补能力，支持系统更新；
- ◇ 系统部件的功率控制优化；
- ◇ 集成睡眠模式，实现低功耗；
- ◇ 快速的代码执行允许使用较慢的处理器时钟或增加睡眠模式的时间；
- ◇ 硬件除法器 and 快速乘法器；
- ◇ 确定性的高性能中断处理，可用于对时间要求严格的应用中；
- ◇ 可选的存储器保护单元（MPU），用于对安全性要求严格的应用中；
- ◇ 全面的调试和跟踪能力：
  - ◇ 串行线调试和串行线跟踪，减少了调试和跟踪所需的管脚数。

### 附录1.1.1.4 Cortex-M3 内核外设

Cortex-M3 内核外设：

- ◇ 可嵌套向量中断控制器

可嵌套向量中断控制器（NVIC）是一种支持低等待时间中断处理的嵌入式中断控制器。

- ◇ 系统控制模块

系统控制模块（SCB）是编程模型与处理器之间的接口。它提供了系统实现信息和系统控制，包括配置、控制以及系统异常的报告。

- ◇ 系统定时器

系统定时器 SysTick 是一个 24 位的递减计数定时器。它可以用作实时操作系统（RTOS）的节拍定时器或一个普通计数器。

- ◇ 存储器保护单元

存储器保护单元（MPU）通过为不同存储区定义存储器属性，提高了系统的可靠性。它最多可以提供 8 个不同的区域，还有一个可选的预定义背景区。

## 附录1.2 ARM Cortex-M3 用户指南：指令集

### 附录1.2.1 指令集汇总

该处理器实现了一个 Thumb 指令集版本。下表列出了支持的指令。

下表中：

- ◇ 尖括号<>中为操作数的代替形式
- ◇ 花括号{}中为可选操作数
- ◇ “操作数”栏并未列出全部
- ◇ Op2 是一个灵活的第二操作数，可以是一个寄存器或是一个常数
- ◇ 多数指令可使用一个可选的条件代码后缀。



有关指令和操作数的更多信息，请参见指令描述部分。

助记符	操作数	简要描述	标志	章节（已改为对应名称）
ADC, ADCS	{Rd,} Rn, Op2	带进位加法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
ADD, ADDS	{Rd,} Rn, Op2	加法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
ADD, ADDW	{Rd,} Rn, #imm12	加法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
ADR	Rd, label	加载相对 PC 地址	-	ADR
AND, ANDS	{Rd,} Rn, Op2	逻辑“与”	N, Z, C	AND, ORR, EOR, BIC 和 ORN
ASR, ASRS	Rd, Rm, <Rs   #n>	算术右移	N, Z, C	ASR, LSL, LSR, ROR 和 RRX
B	label	跳转	-	B、BL、BX 和 BLX
BFC	Rd, #lsb, #width	位域清零	-	BFC 和 BFI
BFI	Rd, Rn, #lsb, #width	位域插入	-	BFC 和 BFI
BIC, BICS	{Rd,} Rn, Op2	位清零	N, Z, C	AND, ORR, EOR, BIC 和 ORN
BKPT	#imm	断点	-	BKPT
BL	label	带链接的跳转	-	B、BL、BX 和 BLX
BLX	Rm	带链接的跳转并切换指令集	-	B、BL、BX 和 BLX
BX	Rm	跳转并切换指令集	-	B、BL、BX 和 BLX
CBNZ	Rn, label	比较，结果非零则跳转	-	CBZ 和 CBNZ
CBZ	Rn, label	比较，结果为零则跳转	-	CBZ 和 CBNZ
CLREX	-	独占清零	-	CLREX
CLZ	Rd, Rm	计算前导零数目	-	CLZ
CMN, CMNS	Rn, Op2	与负值比较	N, Z, C, V	CMP 和 CMN
CMP, CMPS	Rn, Op2	比较	N, Z, C, V	CMP 和 CMN
CPSID	iflags	更改处理器状态，禁能中断	-	CPS
CPSIE	iflags	更改处理器状态，	-	CPS

助记符	操作数	简要描述	标志	章节（已改为对应名称）
		使能中断		
DMB	-	数据内存屏障	-	DMB
DSB	-	数据同步屏障	-	DSB
EOR, EORS	{Rd,} Rn, Op2	异或	N, Z, C	AND, ORR, EOR, BIC 和 ORN
ISB	-	指令同步屏障	-	ISB
IT	-	If-Then 条件模块	-	IT
LDM	Rn{!}, reglist	加载多个寄存器， 之后递增	-	LDM 和 STM
LDMDB, LDMEA	Rn{!}, reglist	加载多个寄存器， 之前递减	-	LDM 和 STM
LDMFD, LDMIA	Rn{!}, reglist	加载多个寄存器， 之后递增	-	LDM 和 STM
LDR	Rt, [Rn, #offset]	加载字至寄存器	-	内存访问指令
LDRB, LDRBT	Rt, [Rn, #offset]	加载字节至寄存器	-	内存访问指令
LDRD	Rt, Rt2, [Rn, #offset]	加载双字节至寄存 器	-	LDR 和 STR(直接偏移量)
LDREX	Rt, [Rn, #offset]	独占加载寄存器	-	LDREX 和 STREX
LDREXB	Rt, [Rn]	独占加载字节至寄 存器	-	LDREX 和 STREX
LDREXH	Rt, [Rn]	独占加载半字至寄 存器	-	LDREX 和 STREX
LDRH, LDRHT	Rt, [Rn, #offset]	加载半字至寄存器	-	内存访问指令
LDRSB, LDRSBT	Rt, [Rn, #offset]	加载经有符号扩展 后的字节至寄存器	-	内存访问指令
LDRSH, LDRSHT	Rt, [Rn, #offset]	加载经有符号扩展 后的半字至寄存器	-	内存访问指令
LDRT	Rt, [Rn, #offset]	加载字至寄存器	-	内存访问指令
LSL, LSLS	Rd, Rm, <Rs   #n>	逻辑左移	N, Z, C	ASR, LSL, LSR, ROR 和 RRX
LSR, LSRS	Rd, Rm, <Rs	逻辑右移	N, Z, C	ASR, LSL, LSR, ROR

助记符	操作数	简要描述	标志	章节（已改为对应名称）
	#n>			和 RRX
MLA	Rd, Rn, Rm, Ra	乘加, 结果为 32 位	-	MUL、MLA 和 MLS
MLS	Rd, Rn, Rm, Ra	乘减, 结果为 32 位	-	MUL、MLA 和 MLS
MOV, MOVS	Rd, Op2	移动	N, Z, C	MOV 和 MVN
MOVT	Rd, #imm16	移动到顶部	-	MOVT
MOVW, MOV	Rd, #imm16	移动 16 位常数	N, Z, C	MOV 和 MVN
MRS	Rd, spec_reg	将专用寄存器的内容移到通用寄存器中	-	MRS
MSR	spec_reg,Rm	将通用寄存器的内容移到专用寄存器中	N, Z, C, V	MSR
MUL, MULS	{Rd,} Rn, Rm	乘法, 结果为 32 位	N, Z	MUL、MLA 和 MLS
MVN, MVNS	Rd, Op2	取反移动	N, Z, C	MOV 和 MVN
NOP	-	不执行任何操作	-	NOP
ORN, ORNS	{Rd,} Rn, Op2	逻辑“或非”	N, Z, C	AND, ORR, EOR, BIC 和 ORN
ORR, ORRS	{Rd,} Rn, Op2	逻辑“或”	N, Z, C	AND, ORR, EOR, BIC 和 ORN
POP	reglist	从堆栈弹出寄存器	-	PUSH 和 POP
PUSH	reglist	将寄存器推入堆栈	-	PUSH 和 POP
RBIT	Rd, Rn	反转位	-	REV, REV16, REVSH 和 RBIT
REV	Rd, Rn	反转字中的字节顺序	-	REV, REV16, REVSH 和 RBIT
REV16	Rd, Rn	反转每个半字中的字节顺序	-	REV, REV16, REVSH 和 RBIT
REVSH	Rd, Rn	反转低半字中的字节顺序, 并将符号扩展	-	REV, REV16, REVSH 和 RBIT
ROR, RORS	Rd, Rm, <Rs   #n>	向右循环移	N, Z, C	ASR, LSL, LSR, ROR 和 RRX
RRX, RRXS	Rd, Rm	带扩展向右循环移	N, Z, C	ASR, LSL, LSR, ROR

助记符	操作数	简要描述	标志	章节（已改为对应名称）
				和 RRX
RSB, RSBS	{Rd,} Rn, Op2	反向减法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
SBC, SBCS	{Rd,} Rn, Op2	带进位减法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
SBFX	Rd, Rn, #lsb, #width	有符号位域提取	-	SBFX 和 UBFX
SDIV	{Rd,} Rn, Rm	有符号除法	-	SDIV 和 UDIV
SEV	-	发送事件	-	SEV
SMLAL	RdLo, RdHi, Rn, Rm	有符号乘加 (32x32+64), 结果 为 64 位		UMULL, UMLAL, SMULL 和 SMLAL
SMULL	RdLo, RdHi, Rn, Rm	有符号乘法 (32x32), 结果为 64 位	-	UMULL, UMLAL, SMULL 和 SMLAL
SSAT	Rd, #n, Rm {,shift #s}	有符号饱和	Q	SSAT 和 USAT
STM	Rn{!}, reglist	存储多个寄存器, 之后递增	-	LDM 和 STM
STMDB, STMEA	Rn{!}, reglist	存储多个寄存器, 之前递减	-	LDM 和 STM
STMTD, STMIA	Rn{!}, reglist	存储多个寄存器, 之后递增	-	LDM 和 STM
STR	Rt, [Rn, #offset]	存储寄存器字	-	内存访问指令
STRB, STRBT	Rt, [Rn, #offset]	存储寄存器字节	-	内存访问指令
STRD	Rt, Rt2, [Rn, #offset]	存储寄存器双字	-	LDR 和 STR(直接偏移量)
STREX	Rd, Rt, [Rn, #offset]	独占存储寄存器	-	LDREX 和 STREX
STREXB	Rd, Rt, [Rn]	独占存储寄存器, 字节	-	LDREX 和 STREX
STREXH	Rd, Rt, [Rn]	独占存储寄存器, 半字	-	LDREX 和 STREX
STRH,	Rt, [Rn,	存储寄存器, 半字	-	内存访问指令

助记符	操作数	简要描述	标志	章节（已改为对应名称）
STRHT	#offset]			
STRT	Rt, [Rn, #offset]	存储寄存器，字	-	内存访问指令
SUB, SUBS	{Rd,} Rn, Op2	减法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
SUB, SUBW	{Rd,} Rn, #imm12	减法	N, Z, C, V	ADD、ADC、SUB、SBC 和 RSB
SVC	#imm	超级用户调用	-	BKPT0
SXTB	{Rd,} Rm {,ROR #n}	有符号扩展一个字节	-	SXT 和 UXT
SXTH	{Rd,} Rm {,ROR #n}	有符号扩展一个半字	-	SXT 和 UXT
TBB	[Rn, Rm]	表跳转字节	-	TBB 和 TBH
TBH	[Rn, Rm, LSL #1]	表跳转半字	-	TBB 和 TBH
TEQ	Rn, Op2	相等测试	N, Z, C	TST 和 TEQ
TST	Rn, Op2	测试	N, Z, C	TST 和 TEQ
UBFX	Rd, Rn, #lsb, #width	无符号位域提取	-	SBFX 和 UBFX
UDIV	{Rd,} Rn, Rm	无符号除法	-	SDIV 和 UDIV
UMLAL	RdLo, RdHi, Rn, Rm	无符号乘加 (32x32+64), 结果为 64 位		UMULL, UMLAL, SMULL 和 SMLAL
UMULL	RdLo, RdHi, Rn, Rm	无符号乘法 (32x32), 结果为 64 位	-	UMULL, UMLAL, SMULL 和 SMLAL
USAT	Rd, #n, Rm {,shift #s}	无符号饱和	Q	SSAT 和 USAT
UXTB	{Rd,} Rm {,ROR #n}	用零扩展一个字节	-	SXT 和 UXT
UXTH	{Rd,} Rm {,ROR #n}	用零扩展一个半字	-	SXT 和 UXT
WFE	-	等待事件	-	BKPT1
WFI	-	等待中断	-	BKPT2

附录图 39-2 Cortex-M3 指令

### 附录1.2.2 内在函数

ANSI 不能直接访问某些 Cortex-M3 指令。本节描述了可生成这些指令的内在函数，这些函数由 CMSIS 提供，也可以由 C 编译器提供。如果 C 编译器不支持合适的内在函数，则可能必须要用内联汇编器来访问这些指令。

CMSIS 提供以下内在函数，用以生成 ANSI 不能直接访问的指令：

指令	CMSIS 内在函数
CPSIE I	void__enable_irq(void)
CPSID I	void__disable_irq(void)
CPSIE F	void__enable_fault_irq(void)
CPSID F	void__disable_fault_irq(void)
ISB	void__ISB(void)
DSB	void__DSB(void)
DMB	void__DMB(void)
REV	uint32_t__REV(uint32_t int value)
REV16	uint32_t__REV16(uint32_t int value)
REVSH	uint32_t__REVSH(uint32_t int value)
RBIT	uint32_t__RBIT(uint32_t int value)
SEV	void__SEV(void)
WFE	void__WFE(void)
WFI	void__WFI(void)

附录图 39-3 用来生成一些 Cortex-M3 指令的 CMSIS 内在函数

CMSIS 还提供了一些函数，用于访问使用 MRS 和 MSR 指令的专用寄存器：

专用寄存器	访问	CMSIS 函数
PRIMASK	读	uint32_t__get_PRIMASK(void)
	写	void__set_PRIMASK(uint32_t value)
FAULTMASK	读	uint32_t__get_FAULTMASK(void)
	写	void__set_FAULTMASK(uint32_t value)
BASEPRI	读	uint32_t__get_BASEPRI(void)
	写	void__set_BASEPRI(uint32_t value)
CONTROL	读	uint32_t__get_CONTROL(void)
	写	void__set_CONTROL(uint32_t value)

专用寄存器	访问	CMSIS 函数
MSP	读	uint32_t __get_MSP(void)
	写	void __set_MSP(uint32_t TopOfMainStack)
PSP	读	uint32_t __get_PSP(void)
	写	Void __set_PSP(uint32_t TopOfProcStack)

附录图 39-4 用来访问专用寄存器的 CMSIS 内在函数

### 附录1.2.3 关于指令描述

以下章节提供了与使用指令相关的更多信息：

#### 附录1.2.3.1 操作数

指令操作数可以是一个 ARM 寄存器、一个常数或是其他指令特有的参数。指令按操作数执行，并通常将结果保存在一个目标寄存器中。当指令中有一个目标寄存器时，它通常在操作数前指定。

某些指令中的操作数比较灵活，因为它即可以是一个寄存器，也可以是一个常数，参见小节“灵活的第二操作数”。

#### 附录1.2.3.2 使用 PC 或 SP 时的限制

很多指令对操作数或目标寄存器是否可使用程序计数器（PC）或堆栈指针（SP）有限制。更多信息请参见指令描述。

注：当使用 BX、BLX、LDM、LDR 或 POP 指令写 PC 时，任意写入地址的位[0]必须为 1 才能正确执行指令，因为此位表示所需的指令集，而 Cortex-M3 处理器只支持 Thumb 指令。

#### 附录1.2.3.3 灵活的第二操作数

很多通用数据处理指令都有一个灵活的第二操作数，在各个指令的语法描述中表示为操作数 2（Operand2）。

Operand2 可以是“常数”或“带可选移位的寄存器”。

##### 常数

以下列形式指定一个 Operand2 常数：

#constant

其中“constant”可以是：

- ◇ 在一个 32 位的字内，将一个 8 位值左移任意位数可以得到的任何常数；
- ◇ 任何 0x00XY00XY 形式的常数；
- ◇ 任何 0xXY00XY00 形式的常数；
- ◇ 任何 0xXYXYXYXY 形式的常数。

注：在上述常数中，X 和 Y 均为 16 进制数字。





的影响方式。在这些描述中，Rm 为包含待移位值的寄存器，n 为移位长度。

**ASR**

算术右移 n 位，是将寄存器 Rm 左边 32-n 位向右移 n 个位置，成为结果中的右边 32-n 位。并且，它还将寄存器原来的位[31]复制到结果左边的 n 个位上。参见下图。

可使用 ASR #n 操作来将寄存器 Rm 中的值除以  $2^n$ ，得到的结果向负无穷方向舍入。

当指令为 ASRS，或指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 中将 ASR #n 用于 Operand2 时，进位标志更新为寄存器 Rm 的最后一个移出位，即位[n-1]。

- ◇ 如果 n 为 32 或更大，则结果中的所有位都设置为 Rm 位[31]的值。
- ◇ 如果 n 为 32 或更大且进位标志被更新，则其更新为 Rm 位[31]的值。



附录图 39-5 ASR #3

**LSR**

逻辑右移 n 位，是将寄存器 Rm 左边 32-n 位向右移 n 个位置，成为结果中的右边 32-n 位。并且，将结果的左边 n 位设置为 0。参见下图。

如果 Rm 中的值被视为一个无符号整数，则可以使用 LSR #n 操作将寄存器 Rm 中的值除以  $2^n$ 。

当指令为 LSRS，或将 LSR #n 用于指令 MOVS、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时，进位标志更新为寄存器 Rm 的最后一个移出位，即位[n-1]。

- ◇ 如果 n 为 32 或更大，则结果中的全部位都清零。
- ◇ 如果 n 为 33 或更大且进位标志被更新，则其更新为 0。



附录图 39-6 LSR #3

**LSL**

逻辑左移  $n$  位, 是将寄存器  $Rm$  右边  $32-n$  位向左移  $n$  个位置, 成为结果的左边  $32-n$  位。并且将结果的右边  $n$  位设置为 0。参见下图。

如果寄存器  $Rm$  中的值被视为一个无符号整数或一个 2 的补码有符号整数, 则可使用  $LSL \#n$  操作将寄存器  $Rm$  中的值乘以  $2^n$ 。这可能会引发不带报警的溢出。

当指令为  $LSLS$ , 或将  $LSL \#n$  ( $n$  非零) 用于指令  $MOVS$ 、 $MVNS$ 、 $ANDS$ 、 $ORRS$ 、 $ORNS$ 、 $EORS$ 、 $BICS$ 、 $TEQ$  或  $TST$  的  $Operand2$  时, 进位标志更新为寄存器  $Rm$  的最后一个移出位, 即位 $[32-n]$ 。当使用  $LSL \#0$  时, 这些指令不影响进位标志。

- ◇ 如果  $n$  为 32 或更大, 则结果中全部位都清零。
- ◇ 如果  $n$  为 33 或更大且进位标志被更新, 则其更新为 0。



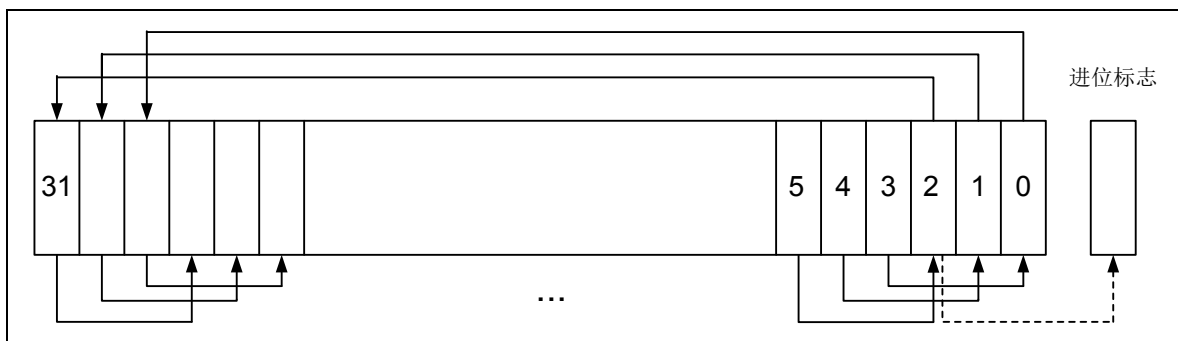
附录图 39-7 LSL #3

**ROR**

向右循环移  $n$  位, 是将寄存器  $Rm$  左边  $32-n$  位向右移  $n$  个位置, 成为结果的右边  $32-n$  位。并且将寄存器的右边  $n$  位移到结果的左边  $n$  位。参见下图。

当指令为  $RORS$ , 或将  $ROR \#n$  用于指令  $MOVS$ 、 $MVNS$ 、 $ANDS$ 、 $ORRS$ 、 $ORNS$ 、 $EORS$ 、 $BICS$ 、 $TEQ$  或  $TST$  的  $Operand2$  时, 进位标志更新为寄存器  $Rm$  的最后一个循环位, 即位 $[n-1]$ 。

- ◇ 如果  $n$  为 32, 则结果的值与  $Rm$  中的值相同, 并且如果进位标志被更新, 则它更新为  $Rm$  的位 $[31]$ 。
- ◇ 移位长度  $n$  大于 32 的  $ROR$  与移位长度为  $n-32$  的  $ROR$  相同。



附录图 39-8 ROR #3

### RRX

带扩展的向右循环移是将寄存器 Rm 中的位向右移一位。它还将进位标志复制到结果的位[31]中。参见下图。

当指令为 RRXS, 或将 RRX 用于指令 MOVs、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 的 Operand2 时, 进位标志更新为寄存器 Rm 的位[0]。



附录图 39-9 RRX

#### 附录1.2.3.5 地址对齐

对齐访问是将一个按字对齐的地址用于一个字、双字和多字的访问中, 或将一个按半字对齐的地址用于半字访问中。字节访问总是对齐的。

Cortex-M3 处理器仅对下列指令支持非对齐的访问:

- ◇ LDR, LDRT
- ◇ LDRH, LDRHT
- ◇ LDRSH, LDRSHT
- ◇ STR, STRT
- ◇ STRH, STRHT

如果执行非对齐访问, 则所有其他的加载和存储指令都会产生使用故障异常, 因此, 这些指令的访问都必须是地址对齐访问。有关使用故障的更多信息, 见“故障处理”小节。

非对齐访问通常慢于对齐访问。另外, 某些存储区可能不支持非对齐访问。因此, ARM 建议, 程序员们应确保每次访问都是对齐访问。为了避免非对齐访问的偶然发生, 应在配置和控制寄存器中使用 UNALIGN\_TRP 位来捕获所有的非对齐访问, 见“配置和控制寄存器”小节。

#### 附录1.2.3.6 相对 PC 的表达式

相对 PC 的表达式或标签是一个代表指令地址或立即数 (literal data) 的符号。在指令中, 它表示为: PC 值加上或减去一个数值偏移量。汇编器根据标签和当前指令的地址计算出所需的偏移量。如果偏移量过大, 则汇编器产生一个错误。

- ◇ 对于 B、BL、CBNZ 和 CBZ 指令, PC 值为当前指令地址加上 4 字节。
- ◇ 对于使用标签的所有其他指令, PC 值为当前指令地址加上 4 字节, 结果的位[1]清零, 使之按字对齐。

- ◇ 所用汇编器可能允许相对 PC 表达式的其他语法, 例如一个标签加上或减去一个数字, 或一个[PC, #number]形式的表达式。

### 附录1.2.3.7 条件执行

大多数数据处理指令都具有一个选项，该选项可以根据操作结果来更新应用程序状态寄存器（SPAR）中的条件标志，见“程序状态寄存器”小节。有些指令会更新全部标志，有些指令则只更新一个子集。如果某个标志没有更新，则会保留其原始值。受影响的标志参见指令描述部分。

可以根据其他指令中设置的条件标志有条件地执行一条指令，执行时间为：

- ◇ 在更新标志的指令后立即执行；
- ◇ 在没有更新标志的任意数目的插入指令之后执行。

当使用条件跳转或为指令添加条件代码后缀时，可以实现有条件地执行。表“条件代码后缀”是一份后缀清单，这些后缀都可以添加到指令中使其可以有条件地执行。条件代码后缀使处理器能够根据标志来测试一个条件。如果某个条件指令的条件测试失败，则指令会：

- ◇ 不执行；
- ◇ 不向其目标寄存器写入任何值；
- ◇ 不影响任何标志；
- ◇ 不产生任何异常。

除了条件跳转外，条件指令必须处于一个 **If-Then** 指令模块中。有关使用 **IT** 指令的更多信息和限制，请参见“**IT**”小节。如果在 **IT** 块外存在条件指令，汇编器可能自动插入一条 **IT** 指令（取决于厂商）。

使用 **CBZ** 和 **CBNZ** 指令将寄存器值与零比较，并根据结果执行跳转（**branch**）。

本节介绍以下内容：“条件标志”和“条件代码后缀”。

#### 条件标志

**APSR** 包括以下条件标志：

- N**—操作结果为负值时设置为 1，否则清零。
- Z**—操作结果为零时设置为 1，否则清零。
- C**—操作导致一个进位出现时设置为 1，否则清零。
- V**—操作引起溢出时设置为 1，否则清零。

有关 **APSR** 的更多信息，请参见“程序状态寄存器”小节。以下情况中出现进位：

- ◇ 如果加法结果大于或等于  $2^{32}$ ；
- ◇ 如果减法结果为正值或零；
- ◇ 在一个移动或逻辑指令中，内联桶式（**inline barrel**）移位器操作的结果。

如果一个加法、减法或比较的结果大于或等于  $2^{31}$ ，或小于  $-2^{31}$ ，则发生溢出。

注：多数指令仅在指定后缀 **S** 的情况下才更新状态标志。更多信息请参见指令描述。

### 条件代码后缀

可有条件地执行的指令具有一个可选条件代码，如{cond}中的语法描述所示。条件执行时要求之前有一条 IT 指令。仅当 APSR 中的条件代码标志满足指定的条件时，才会执行带有条件代码的指令。下表显示了可使用的条件代码。

可以通过 IT 指令使用有条件地执行来减少代码中跳转指令的数量。

下表还显示了条件代码后缀和 N、Z、C 和 V 标志之间的关系。

后缀	标志	含义
EQ	Z = 1	等于
NE	Z = 0	不等于
CS/HS	C = 1	进位置位/无符号大于或等于
CC/LO	C = 0	进位清零/无符号小于
MI	N = 1	减/负
PL	N = 0	加/正或零
VS	V = 1	溢出
VC	V = 0	无溢出
HI	C = 1 且 Z = 0	无符号大于
LS	C = 0 或 Z = 1	无符号小于或等于
GE	N = V	有符号大于或等于
LT	N! = V	有符号小于
GT	Z = 0 且 N = V	有符号大于
LE	Z = 1 且 N! = V	有符号小于或等于
AL	任意值	始终。无指定后缀时的缺省值

附录图 39-10 条件代码后缀

示例：绝对值 展示了使用一个条件指令来得到某个数的绝对值。R0 = ABS (R1)。

示例：比较和更新值 展示了当符号值 R0 大于 R1 且 R2 大于 R3 时，使用条件指令来更新 R4 的值。

示例：绝对值：

MOVS R0, R1 ; R0 = R1, setting flags

ITMI ; IT instruction for the negative condition

RSBMI R0, R1, #0 ; If negative, R0 = -R1

示例：比较和更新值：

CMP R0, R1 ; Compare R0 and R1, setting flags

ITT GT ; IT instruction for the two GT conditions

CMPGT R2, R3 ; If 'greater than', compare R2 and R3, setting flags

MOVGT R4, R5 ; If still 'greater than', do R4 = R5

### 附录1.2.3.8 指令宽度选择

很多指令既可产生 16 位编码，也可产生 32 位编码，这取决于指定的操作数和目标寄存器。对其中的一些指令，可使用一个指令宽度后缀来强制设置一个特定的指令尺寸。“*.W*” 后缀会强制使用 32 位指令编码。“*.N*” 后缀会强制使用 16 位指令编码。

如果指定了一个指令宽度后缀，但汇编器无法生成所要求宽度的指令编码，则产生一个错误。

注：某些情况下指定 *.W* 后缀是非常有必要的，例如，当操作数为一个指令的标签或立即数时，如跳转指令的情况。这是因为汇编器可能无法自动生成大小正确的编码。

当使用指令宽度后缀时，将其紧跟在指令助记符和条件代码（如果有）后面。如下列出了带指令宽度后缀的指令。

#### 示例：指令宽度选择

```
BCS.W Label           ; creates a 32-bit instruction even for a short branch
ADDS.W R0, R0, R1     ; creates a 32-bit instruction even though the same
                       ; operation can be done by a 16-bit instruction
```

### 附录1.2.4 内存访问指令

下表中列出了内存访问指令：

	简要描述	参见（已改为对应章节名称）
ADR	加载相对 PC 地址	ADR
CLREX	独占清零	CLREX
LDM{mode}	加载多个寄存器	LDM 和 STM
LDR{type}	利用直接偏移量加载寄存器	LDR 和 STR（直接偏移量）
LDR{type}	利用寄存器偏移量加载寄存器	LDR 和 STR（寄存器偏移量）
LDR{type}T	通过非特权访问加载寄存器	LDR 和 STR（非特权）
LDR	利用相对 PC 地址加载寄存器	LDR（相对 PC）
LDREX{type}	独占加载寄存器	LDREX 和 STREX
POP	从堆栈弹出寄存器	PUSH 和 POP
PUSH	将寄存器推入堆栈	PUSH 和 POP
STM{mode}	存储多个寄存器	LDM 和 STM
STR{type}	利用直接偏移量存储寄存器	LDR 和 STR（直接偏移量）
STR{type}	利用寄存器偏移量存储寄存器	LDR 和 STR（寄存器偏移量）
STR{type}T	通过非特权访问存储寄存器	LDR 和 STR（非特权）
STREX{type}	独占存储寄存器	LDREX 和 STREX

附录图 39-11 内存访问指令

### 附录1.2.4.1 ADR

加载相对 PC 地址。

#### 语法

ADR{cond} Rd, label

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为目标寄存器。

label 为一个相对 PC 表达式。参见“相对 PC 的表达式”小节。

#### 操作

ADR 通过将一个立即值与 PC 值相加来确定地址，并将结果写入目标寄存器。

ADR 生成与位置无关的代码，因为地址相对于 PC。

如果使用 ADR 为 BX 或 BLX 指令生成目标地址，则必须确保将所生成地址的位[0] 设为 1，才能保证正确执行。

label 的值必须在 PC 中地址的-4095 到+4095 范围内。

注：要得到最大偏移范围或生成未按字对齐的地址时可能必须使用到“.W”后缀。参见“指令宽度选择”小节。

#### 限制

Rd 不得为 SP 且不得为 PC。

#### 条件标志

此指令不改变标志。

#### 示例

```
ADR R1, TextMessage ; Write address value of a location labelled as  
; TextMessage to R1
```



## 附录1.2.4.2 LDR 和 STR（直接偏移量）

带有直接偏移量、前变址直接偏移量或后变址直接偏移量的加载和存储。

### 语法

op{type}{cond} Rt, [Rn {, #offset}] ; 直接偏移量  
op{type}{cond} Rt, [Rn, #offset]! ; 前变址  
op{type}{cond} Rt, [Rn], #offset ; 后变址  
opD{cond} Rt, Rt2, [Rn {, #offset}] ; 直接偏移量, 双字  
opD{cond} Rt, Rt2, [Rn, #offset]! ; 前变址, 双字  
opD{cond} Rt, Rt2, [Rn], #offset ; 后变址, 双字

其中:

Op 可为以下指令之一:

LDR: 加载寄存器。

STR: 存储寄存器。

Type 可以是下列项之一:

B: 无符号字节, 加载时零扩展到 32 位。

SB: 有符号字节, 符号扩展到 32 位 (仅 LDR)。

H: 无符号半字, 加载时零扩展到 32 位。

SH: 有符号半字, 符号扩展到 32 位 (仅 LDR)。

—: 如果是字, 则省略。

cond 为一个可选的条件代码, 见 “条件执行” 小节。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为一个 Rn 的偏移量。如果省略 offset, 则该地址为 Rn 的内容。

Rt2 为附加寄存器, 在双字操作时使用, 用于加载或存储。

### 操作

LDR 指令将存储器中的一个值加载到一个或两个寄存器中。

STR 指令将一个或两个寄存器值存储到存储器中。

带直接偏移量的加载和存储指令可用于以下寻址模式:

#### ◇ 偏移量寻址

在寄存器 Rn 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址。寄存器 Rn 不改变。此模式的汇编语言语法为:

[Rn, #offset]

#### ◇ 前变址寻址

在寄存器 Rn 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址，并写回寄存器 Rn 中。此模式的汇编语言语法为：

[Rn, #offset]!

◇ 后变址寻址

使用来自寄存器 Rn 的地址作为存储器的访问地址。给地址加上或减去一个偏移量，并写回寄存器 Rn 中。此模式的汇编语言语法为：

[Rn], #offset

要加载或存储的值可以是一个字节、半字、字或双字。字节和半字可以有符号的，也可以是无符号的。参见“地址对齐”小节。

下表显示了直接、前变址和后变址形式的偏移量范围。

指令类型	直接偏移量	前变址偏移量	后变址偏移量
字、半字、有符号半字、字节或有符号字节	-255~4095	-255~255	-255~255
双字	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数

附录图 39-12 偏移量范围

### 限制

对于加载指令：

- ◇ 仅对字加载，Rt 可以是 SP 或 PC；
- ◇ 对双字加载，Rt 必须与 Rt2 不同；
- ◇ 前变址或后变址形式中，Rn 必须与 Rt 和 Rt2 不同。
- ◇ 当字加载指令中的 Rt 为 PC 时：
- ◇ 加载值的位[0]必须为 1 才能正确执行；
- ◇ 发生到当被加载值位[0]变为 0 时生成的一个地址的跳转；
- ◇ 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。
- ◇ 对于存储指令：
- ◇ 仅对字存储，Rt 可以是 SP 或 PC；
- ◇ Rt 不得为 PC；
- ◇ Rn 不得为 PC；
- ◇ 前变址或后变址形式中，Rn 必须与 Rt 和 Rt2 不同。

### 条件标志

这些指令不改变标志。

### 示例

```
LDR    R8, [R10]           ; Loads R8 from the address in R10.
LDRNE  R2, [R5, #960]!    ; Loads (conditionally) R2 from a word
```

			; 960 bytes above the address in R5, and ; increments R5 by 960.
STR	R2, [R9,#const-struct]		; const-struct is an expression evaluating ; to a constant in the range 0-4095.
STRH	R3, [R4], #4		; Store R3 as halfword data into address in ; R4, then increment R4 by 4
LDRD	R8, R9, [R3, #0x20]		; Load R8 from a word 32 bytes above the ; address in R3, and load R9 from a word 36 ; bytes above the address in R3
STRD	R0, R1, [R8], #-16		; Store R0 to address in R8, and store R1 to ; a word 4 bytes above the address in R8, ; and then decrement R8 by 16.

### 附录1.2.4.3 LDR 和 STR（寄存器偏移量）

带有寄存器偏移量的加载与存储。

#### 语法

op{type}{cond} Rt, [Rn, Rm {, LSL #n}]

其中：

op 为以下指令之一：

LDR: 加载寄存器。

STR: 存储寄存器。

type 为下列项之一：

B: 无符号字节，加载时零扩展到 32 位。

SB: 有符号字节，符号扩展到 32 位（仅 LDR）。

H: 无符号半字，加载时零扩展到 32 位。

SH: 有符号半字，符号扩展到 32 位（仅 LDR）。

—: 如果是字，则省略。

cond 为一个可选的条件代码，见“条件执行”小节。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

Rm 为一个寄存器，包含要用作偏移量的值。

LSL #n 为一个可选的移位，n 的范围为 0~3。

#### 操作

LDR 指令将存储器的一个值加载到寄存器。

STR 指令将一个寄存器值存入存储器。

加载或存入的存储器地址为来自寄存器 Rn 的一个偏移量。偏移量由寄存器 Rm 指定，

并可使用 LSL 左移最多 3 位。

要加载或存储的值可以是一个字节、半字，或字。对于加载指令，字节和半字可以是有符号的，也可以是无符号的。参见“地址对齐”小节。

### 限制

在这些指令中：

- ◇ Rn 不得为 PC；
- ◇ Rm 不得为 SP 且不得为 PC；
- ◇ Rt 仅在字加载或字存储时可以是 SP；
- ◇ Rt 仅在字加载时可以是 PC。

当字加载指令中的 Rt 为 PC 时：

- ◇ 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址的跳转；
- ◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

### 条件标志

这些指令不改变标志。

### 示例

```
STR R0, [R5, R1] ; Store value of R0 into an address equal to
                  ; sum of R5 and R1
LDRSB R0, [R5, R1, LSL #1] ; Read byte value from an address equal to
                             ; sum of R5 and two times R1, sign extended it
                             ; to a word value and put it in R0
STR R0, [R1, R2, LSL #2] ; Stores R0 to an address equal to sum of R1
                          ; and four times R2
```

## 附录1.2.4.4 LDR 和 STR（非特权）

带非特权访问的加载和存储。

### 语法

op{type}T{cond} Rt, [Rn {, #offset}]；直接偏移量

其中：

op 为下列指令之一：

LDR：加载寄存器。

STR：存储寄存器。

type 为下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

一：如果是字，则省略。

cond 为一个可选条件代码，见“条件执行”小节。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为 Rn 的偏移量，取值范围为 0~255。如果省略 offset，则该地址为 Rn 中的值。

### 操作

这些加载和存储指令完成与带直接偏移量的存储器访问指令相同的功能。差别是，即使在特权软件中，这些指令也只能进行非特权访问。

当在非特权软件中使用时，这些指令与带直接偏移量的正常存储器访问指令完全一样。

### 限制

在这些指令中：

- ◇ Rn 不得为 PC；
- ◇ Rt 不得为 SP 且不得为 PC。

### 条件标志

这些指令不改变标志。

### 示例

```
STRBTEQ    R4, [R7]    ;   Conditionally store least significant byte in
                ;   R4 to an address in R7, with unprivileged access
LDRHT     R2, [R2, #8]  ;   Load halfword value from an address equal to
                ;   sum of R2 and 8 into R2, with unprivileged access
```

## 附录1.2.4.5 LDR（相对 PC）

从存储器加载寄存器。

### 语法

```
LDR{type}{cond} Rt, label
LDRD{cond} Rt, Rt2, label ; 加载双字
```

type 为下列项之一：

- B：无符号字节，加载时零扩展到 32 位。
- SB：有符号字节，符号扩展到 32 位（仅 LDR）。
- H：无符号半字，加载时零扩展到 32 位。
- SH：有符号半字，符号扩展到 32 位（仅 LDR）。

一：如果是字，则省略。

cond 为一个可选条件代码，见“条件执行”小节。

Rt 为要加载或存储的寄存器。

Rt2 为第二个要加载或存储的寄存器。

label 为一个相对 PC 表达式。参见“相对 PC 的表达式”小节。

### 操作

LDR 用一个相对 PC 存储器地址的值加载寄存器。存储器地址由标签指定，或由一个 PC 偏移量指定。

加载或存储的值可以是一个字节、半字或字。对于加载指令，字节和半字可以是有符号的，也可以是无符号的。参见“地址对齐”小节。

label 必须在当前指令的限制范围内。下表显示了标签和 PC 之间的可能的偏移量。

指令类型	偏移量范围
字、半字、有符号半字、字节、有符号字节	-4095~4095
双字	-1020~1020

附录图 39-13 偏移量范围

注：要获得最大偏移量范围可能需要使用“.W”后缀。见“指令宽度选择”小节。

### 限制

在这些指令中：

- ◇ 仅对字加载，Rt 可以是 SP 或 PC；
- ◇ Rt2 不得为 SP 且不得为 PC；
- ◇ Rt 必须与 Rt2 不同。

当一个字加载指令中的 Rt 为 PC 时：

- ◇ 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址上的跳转。
- ◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

### 条件标志

这些指令不改变标志。

### 示例

```
LDR R0, LookUpTable ; Load R0 with a word of data from an address
                    ; labelled as LookUpTable
LDRSB R7, localdata ; Load a byte value from an address labelled
                    ; as localdata, sign extend it to a word
                    ; value, and put it in R7
```

## 附录1.2.4.6 LDM 和 STM

加载和存储多个寄存器。

### 语法

`op{addr_mode}{cond} Rn{!}, reglist`

其中:

`op` 为下列指令之一:

**LDM**: 加载多个寄存器。

**STM**: 存储多个寄存器。

`addr_mode` 为下列项之一:

**IA**: 每次访问后递增地址。这是默认设置。

**DB**: 每次访问前递减地址。

`cond` 为一个可选条件代码, 见“条件执行”小节。

`Rn` 为存储器地址所基于的寄存器。

“!” 为一个可选的写回后缀。如果有“!”, 则要加载或存储到的最终地址被写回到 `Rn`。

`reglist` 为要加载或存储的一个或多个寄存器列表, 括在大括号内。它可以包含寄存器范围。如果包含一个以上的寄存器或寄存器范围, 则必须以逗号隔开。

**LDM** 和 **LDMFD** 与 **LDMIA** 同义。**LDMFD** 指它用于从满降序堆栈 (full descending stack) 弹出 (pop) 数据。

**LDMEA** 与 **LDMDB** 同义, 表示它用于从空升序堆栈 (empty ascending stack) 弹出数据。

**STM** 和 **STMEA** 与 **STMIA** 同义, **STMEA** 表示它用于向空升序堆栈推入 (push) 数据。

**STMFD** 与 **STMDB** 同义, **STMFD** 表示它用于向满降序堆栈推入数据。

### 操作

**LDM** 指令将基于 `Rn` 的存储器地址的字值加载到 `reglist` 中的寄存器。

**STM** 指令将 `reglist` 中寄存器的字值存储到基于 `Rn` 的存储器地址。

对于 **LDM**、**LDMIA**、**LDMFD**、**STM**、**STMIA** 和 **STMEA**, 用于访问的存储器地址以 4 个字节为间隔, 范围从 `Rn` 到 `Rn + 4x (n-1)`, 其中 `n` 为 `reglist` 中的寄存器序号。访问按照寄存器序号递增的顺序进行, 最低序号的寄存器使用最低的存储器地址, 最高序号的寄存器使用最高的存储器地址。如果指定了写回后缀, 则 `Rn + 4x (n-1)` 的值被写回到 `Rn`。

对于 **LDMDB**、**LDMEA**、**STMDB** 和 **STMFD**, 用于访问的存储器地址以 4 个字节为间隔, 范围为从 `Rn` 到 `Rn - 4x (n-1)`, 其中 `n` 为 `reglist` 中的寄存器序号。访问是按照寄存器序号递减的顺序进行, 最高序号的寄存器使用最高的存储器地址, 最低序号的

寄存器使用最低的存储器地址。如果指定了写回后缀，则  $Rn - 4x(n-1)$  的值被写回到  $Rn$ 。

PUSH 和 POP 指令可以用此形式表示。详情参见“PUSH 和 POP”小节。

### 限制

在这些指令中：

- ◇  $Rn$  不得为 PC；
- ◇ reglist 不得包括 SP；
- ◇ 在任何 STM 指令中，reglist 不得包括 PC；
- ◇ 在任何 LDM 指令中，如果 reglist 包括 LR，则它不得包括 PC；
- ◇ 如果指定了写回后缀，则 reglist 不得包括  $Rn$ 。

STM 指令中，当 PC 包括在 reglist 中时：

- ◇ 加载到 PC 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。
- ◇ 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。

### 条件标志

这些指令不改变标志。

### 示例

```
LDM R8,{R0,R2,R9} ; LDMIA is a synonym for LDM
STMDB R1!,{R3-R6,R11,R12}
```

### 不正确示例

```
STM R5!,{R5,R4,R9} ; Value stored for R5 is unpredictable
LDM R2, {} ; There must be at least one register in the list
```



### 附录1.2.4.7 PUSH 和 POP

向一个满降序堆栈推入和弹出寄存器。

#### 语法

PUSH{cond} reglist POP{cond} reglist

其中：

cond 为一个可选条件代码，见“条件执行”小节。

reglist 为一个非空的寄存器列表，括在大括号内。它可以包括寄存器范围。如果包含了一个以上寄存器或寄存器范围，则必须以逗号隔开。

PUSH 和 POP 与 STMDB 和 LDM(或 LDMIA)同义，以存储器地址进行基于 SP 的访问，访问的最终地址被写回 SP。在这些情况下，PUSH 和 POP 是首选助记符。

#### 操作

PUSH 按照寄存器序号递减的顺序将寄存器存储到堆栈上，最高序号的寄存器使用最高的存储器地址，最低序号的寄存器使用最低的存储器地址。

POP 按照寄存器序号递增的顺序从堆栈加载寄存器，最低序号的寄存器使用最低的存储器地址，最高序号的寄存器使用最高的存储器地址。

更多信息请参见“LDM 和 STM”小节。

#### 限制

在这些指令中：

- ◇ reglist 不得包括 SP；
- ◇ 对于 PUSH 指令，reglist 不得包括 PC；
- ◇ 对于 POP 指令，如果 reglist 包括 LR，则它不得包括 PC。

在 POP 指令中，当 PC 包括在 reglist 中时：

- ◇ 被加载到 PC 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。
- ◇ 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

#### 条件标志

这些指令不改变标志。

#### 示例

```
PUSH    {R0,R4-R7}
PUSH    {R2,LR}
POP     {R0,R10,PC}
```

### 附录1.2.4.8 LDREX 和 STREX

独占加载和存储寄存器。

#### 语法

```
LDREX{cond} Rt, [Rn {, #offset}]  
STREX{cond} Rd, Rt, [Rn {, #offset}]  
LDREXB{cond} Rt, [Rn]  
STREXB{cond} Rd, Rt,  
[Rn] LDREXH{cond} Rt,  
[Rn] STREXH{cond} Rd, Rt, [Rn]
```

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为存放返回状态的目标寄存器。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为应用于 Rn 中的值的可选偏移量。如果省略 offset，则该地址为 Rn 中的值。

#### 操作

LDREX、LDREXB 和 LDREXH 分别从一个存储器地址加载一个字、字节和半字。

STREX、STREXB 和 STREXH 分别尝试将一个字、字节和半字存储到某个存储器地址。独占存储指令中所用的地址必须与刚刚执行的独占加载指令中的地址相同。独占存储指令存储值的数据大小也必须与之前独占加载指令加载值的数据大小相同。这就是说，软件一定要用一条独占加载指令和一条匹配的独占存储指令来执行同步操作，见“同步原语”小节。

如果一条独占存储指令执行存储，则它将 0 写入其目标寄存器。如果它未执行存储，则将 1 写入其目标寄存器。如果独占存储指令将 0 写入目标寄存器，则可保证在独占加载和独占存储指令之间，系统中没有其他进程访问过此存储单元。

出于性能考虑，尽量将相应独占加载和独占存储指令之间的指令数量降到最低。

注：如果执行一个独占存储指令的地址不同于之前独占加载指令的地址，则结果不可预知。

#### 限制

在这些指令中：

- ◇ 不要使用 PC；
- ◇ Rd 和 Rt 不要使用 SP；
- ◇ 对于 STREX，Rd 必须与 Rt 和 Rn 都不同；
- ◇ offset 的值必须为 0~1020 范围内 4 的倍数。

#### 条件标志

这些指令不改变标志。

**示例**

```
MOV R1, #0x1 ; Initialize the „lock taken“ value  
LDREX R0, [LockAddr] ; Load the lock value  
CMP R0, #0 ; Is the lock free?  
ITT EQ ; IT instruction for STREXEQ and CMPEQ  
STREXEQ R0, R1, [LockAddr] ; Try and claim the lock  
CMPEQ R0, #0 ; Did this succeed?  
BNE try ; No – try again  
.... ; Yes – we have the lock
```

**附录1.2.4.9 CLREX**

独占清零。

**语法**

CLREX{cond}

其中：cond 为一个可选的条件代码，见“条件执行”小节。

**操作**

使用 CLREX 使下一条 STREX、STREXB 或 STREXH 指令把 1 写入其目标寄存器且不执行存储。在异常处理程序代码中，当一个同步操作中独占加载指令和相应的独占存储指令之间发生异常时，它可用于强制独占存储失败。

更多信息请参见“同步原语”小节。

**条件标志**

这些指令不改变标志。

**示例**

CLREX

## 附录1.2.5 通用数据处理指令

下表列出了数据处理指令：

助记符	简要描述	参见（已改为对应章节名称）
ADC	带进位加法	ADD、ADC、SUB、SBC 和 RSB
ADD	加法	ADD、ADC、SUB、SBC 和 RSB
ADDW	加法	ADD、ADC、SUB、SBC 和 RSB
AND	逻辑“与”	AND, ORR, EOR, BIC 和 ORN
ASR	算术右移	ASR, LSL, LSR, ROR 和 RRX
BIC	位清零	AND, ORR, EOR, BIC 和 ORN
CLZ	计算前导零数目	CLZ
CMN	与负值比较	CMP 和 CMN
CMP	比较	CMP 和 CMN
EOR	异或	AND, ORR, EOR, BIC 和 ORN
LSL	逻辑左移	ASR, LSL, LSR, ROR 和 RRX
LSR	逻辑右移	ASR, LSL, LSR, ROR 和 RRX
MOV	移动	MOV 和 MVN
MOVT	移动到顶部	MOVT
MOVW	移动 16 位常数	MOV 和 MVN
MVN	取反移动	MOV 和 MVN
ORN	逻辑“或非”	AND, ORR, EOR, BIC 和 ORN
ORR	逻辑“或”	AND, ORR, EOR, BIC 和 ORN
RBIT	反转位	REV, REV16, REVSH 和 RBIT
REV	反转字中的字节顺序	REV, REV16, REVSH 和 RBIT
REV16	反转每个半字中的字节顺序	REV, REV16, REVSH 和 RBIT
REVSH	反转低半字中的字节顺序，并将符号扩展	REV, REV16, REVSH 和 RBIT
ROR	向右循环移	ASR, LSL, LSR, ROR 和 RRX
RRX	带扩展向右循环移	ASR, LSL, LSR, ROR 和 RRX
RSB	反向减法	ADD、ADC、SUB、SBC 和 RSB
SBC	带进位减法	ADD、ADC、SUB、SBC 和 RSB
SUB	减法	ADD、ADC、SUB、SBC 和 RSB
SUBW	减法	ADD、ADC、SUB、SBC 和 RSB
TEQ	相等测试	TST 和 TEQ

助记符	简要描述	参见（已改为对应章节名称）
TST	测试	TST 和 TEQ

附录图 39-14 数据处理指令

### 附录1.2.5.1 ADD、ADC、SUB、SBC 和 RSB

加法、带进位加法、减法、带进位减法和反向减法。

#### 语法

op{S}{cond} {Rd,} Rn, Operand 2

op{cond} {Rd,} Rn, #imm12 ; 仅 ADD 和 SUB

其中:

op 为下列项之一:

ADD: 加法。

ADC: 带进位加法。

SUB: 减法。

RSB: 反向减法。

S: 是一个可选后缀。如果指定了 S, 则会更新操作结果的条件代码标志, 见“条件执行”小节。

cond: 是一个可选条件代码, 见“条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略, 则目标寄存器为 Rn。

Rn 为存放第一个操作数的寄存器。

Operand2 为一个灵活的第二操作数。此选项详细说明请参见“灵活的第二操作数”小节。

imm12 可为 0~4095 范围内的任意值。

#### 操作

ADD 指令将 Rn 中的值与 Operand2 或 imm12 中的值相加。

ADC 指令将 Rn 中的值与 Operand2 相加 (带进位标志)。

SUB 指令从 Rn 中的值减去 Operand2 或 imm12 的值。

SBC 指令从 Rn 中的值减去 Operand2 的值。如果进位标志被清除, 则结果减 1。

RSB 指令将 Operand2 的值减去 Rn 中的值。这是很有用的, 因为有了该指令, Operand2

的选项范围就会更大。

可以使用 ADC 和 SBC 进行合成多字运算, 请参见“多字算术操作示例”小节。

亦可参见“ADR”小节。

注: ADDW 与使用 imm12 操作数的 ADD 语法等效。SUBW 与使用 imm12 操作数的 SUB 语法等效。

## 限制

- ◇ Operand2 不得为 SP 且不得为 PC;
- ◇ 仅在 ADD 和 SUB 中, Rd 可以是 SP, 并有附加的限制:
  - Rn 必须也是 SP;
  - Operand2 中的任何移位都限于用 LSL 最多移 3 位;
- ◇ 仅在 ADD 和 SUB 中, Rn 可以是 SP;
- ◇ 仅在 cond 指令中, Rd 可以是 PC, 且其中:
  - 不得指定 S 后缀;
  - Rm 不得为 PC 且不得为 SP;
  - 如果指令是有条件的, 则必须是 IT 块中的最后一条指令。
- ◇ cond 指令有异常时, 仅在 ADD 和 SUB 中, Rn 可以是 SP, 且仅在下列附加限制下:
  - 不得指定 S 后缀;
  - 第二操作数必须是 0~4095 范围内的一个常数。
- ◇ 当使用 PC 进行加法或减法时, 计算前要将 PC 的位[1:0]舍入到 b00, 从而让计算的基址按字对齐。
- ◇ 如果想产生某条指令的地址, 则必须根据 PC 的值调整常数。ARM 建议使用 ADR 指令代替 Rn 等于 PC 的 ADD 或 SUB 指令, 因为编译器会自动为 ADR 指令计算正确的常数

当 cond 指令中的 Rd 为 PC 时:

- ◇ 写入 PC 的值的位[0]被忽略;
- ◇ 发生到将该值的位[0]强制为 0 生成的地址的跳转。

## 条件标志

如果指定了 S, 这些指令根据结果更新 N、Z、C 和 V 标志。

## 示例

```
ADD R2, R1, R3
SUBS R8, R6, #240 ;Sets the flags on the result
RSB R4, R4, #1280 ;Subtracts contents of R4 from 1280
ADCHI R11, R0, R3 ; Only executed if C flag set and Z
; flag clear
```

## 多字算术操作示例

64 位加法中的两条指令将 R2 和 R3 中包含的一个 64 位整数与 R0 和 R1 中包含的另一个 64 位整数相加, 并将结果存入 R4 和 R5。

多字值不一定非要使用连续的寄存器。96 位减法中的指令从 R6、R2 和 R8 中包含的一个 96 位整数中减去 R9、R1 和 R11 中包含的一个 96 位整数。示例将结果存储在 R6、R9 和 R2 中。

64 位加法:

```
ADDS R4, R0, R2 ; add the least significant words
ADC R5, R1, R3 ; add the most significant words with carry
```

96 位减法:

SUBS R6, R6, R9 ; subtract the least significant words

SBCS R9, R2, R1 ; subtract the middle words with carry

SBC R2, R8, R11 ; subtract the most significant words with carry

#### 附录1.2.5.2 AND, ORR, EOR, BIC 和 ORN

逻辑“与”、“或”、“异或”、位清零和“或非”。

##### 语法

op{S}{cond} {Rd,} Rn, Operand2

其中:

op 为下列项之一:

AND: 逻辑“与”。

ORR: 逻辑“或”，或置位。

EOR: 逻辑“异或”。

BIC: 逻辑“与非”，或位清零。

ORN: 逻辑“或非”。

S: 是一个可选后缀。如果指定了 S, 则会更新操作结果的条件代码标志, 见“条件执行”小节。

cond 为一个可选条件代码, 见“条件执行”小节。

Rd 为目标寄存器。

Rn 为保存第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见“灵活的第二操作数”小节。

##### 操作

AND、EOR 和 ORR 指令可对 Rn 和 Operand2 中的值按位进行“与”、“异或”和“或”操作。

BIC 指令对 Rn 中的位和 Operand2 值中相应位的补码进行“与”操作。

ORN 指令对 Rn 中的位和 Operand2 值中相应位的补码进行“或”操作。

##### 限制

不使用 SP 且不使用 PC。

##### 条件标志

如果指定了 S, 则这些指令:

- ◇ 根据结果更新 N 和 Z 标志;
- ◇ 可以在 Operand2 运算期间更新 C 标志, 见“灵活的第二操作数”小节;
- ◇ 不影响 V 标志。



**示例**

```
AND R9, R2, #0xFF00
ORREQ R2, R0, R5
ANDS R9, R8, #0x19
EORS R7, R11, #0x18181818
BIC R0, R1, #0xab
ORN R7, R11, R14, ROR #4
ORNS R7, R11, R14, ASR #32
```

**附录1.2.5.3 ASR, LSL, LSR, ROR 和 RRX**

算术右移、逻辑左移、逻辑右移、向右循环移和带扩展向右循环移。

**语法**

```
op{S}{cond} Rd, Rm, Rs
op{S}{cond} Rd, Rm, #n
RRX{S}{cond} Rd, Rm
```

其中：

op 为下列项之一：

ASR: 算术右移。

LSL: 逻辑左移。

LSR: 逻辑右移。

ROR: 向右循环移。

S: 是一个可选后缀。如果指定了 S, 则会更新操作结果的条件代码标志, 见“条件执行”小节。

Rd 为目标寄存器。

Rm 为存储将被移位数值的寄存器。

Rs 为存放移位长度的寄存器, 所存放的移位长度应用于 Rm 中的值。仅使用最低有效字节, 范围为 0~255。

n 为移位长度。不同指令的移位长度的范围为：

ASR: 移位长度为 1~32;

LSL: 移位长度为 0~31;

LSR: 移位长度为 1~32;

ROR: 移位长度为 1~31。

对于 LSL{S}{cond} Rd, Rm, #0, 首选语法为 MOV{S}{cond} Rd, Rm。

**操作**

ASR、LSL、LSR 和 ROR 将寄存器 Rm 中的位左移或右移, 移位的数量由常数 n 或寄存器 Rs 指定。

RRX 将寄存器 Rm 中的位向右移 1 位。

在所有这些指令中，结果写到 Rd，但寄存器 Rm 中的值保持不变。欲详细了解不同指令产生的结果，请参见“移位操作”小节。

#### 限制

不使用 SP 且不使用 PC。

#### 条件标志

如果指定了 S，则：

- ◇ 这些指令根据结果更新 N 和 Z 标志；
- ◇ 如果移位长度为 0，则不会影响 C 标志。否则，C 标志会更新为移出的最后一位。见“移位操作”小节。

#### 示例

```
ASR R7, R8, #9 ; Arithmetic shift right by 9 bits
LSLS R1, R2, #3 ; Logical shift left by 3 bits with flag update
LSR R4, R5, #6 ; Logical shift right by 6 bits
ROR R4, R5, R6 ; Rotate right by the value in the bottom byte of R6
RRX R4, R5 ; Rotate right with extend
```

### 附录1.2.5.4 CLZ

计算前导零数目。

#### 语法

```
CLZ{cond} Rd, Rm
```

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rm 为操作数寄存器。

#### 操作

CLZ 指令对 Rm 中值中的前导零数目进行计算，并将结果返回到 Rd 中。如果未在源寄存器中设置任何位，则该结果值为 32，如果位[31]被置位，则结果值为零。

#### 限制

不使用 SP 且不使用 PC。

#### 条件标志

此指令不改变标志。

#### 示例

```
CLZ R4,R9
CLZNE R2,R3
```

### 附录1.2.5.5 CMP 和 CMN

比较和与负值比较。

#### 语法

CMP{cond} Rn, Operand2

CMN{cond} Rn, Operand2

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rn 为存放第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见关于灵活的第二操作数的内容。

#### 操作

这些指令可比较寄存器中的值与 Operand2 比较。它们更新结果的条件标志，但不将结果写到寄存器。

CMP 指令从 Rn 中的值减去 Operand2 的值。这与 SUBS 指令功能相同，只是结果被丢弃。

CMN 指令给 Rn 中的值加上 Operand2 的值。这与 ADDS 指令功能相同，只是结果被丢弃。

#### 限制

在这些指令中：

- ◇ 不使用 PC；
- ◇ Operand2 不得为 SP。

#### 条件标志

这些指令根据结果更新 N、Z、C 和 V 标志。

#### 示例

```
CMP R2, R9
```

```
CMN R0, #6400
```

```
CMPGT SP, R7, LSL #2
```

### 附录1.2.5.6 MOV 和 MVN

移动和取反移动。

#### 语法

MOV{S}{cond} Rd, Operand2

MOV{cond} Rd, #imm16

MVN{S}{cond} Rd, Operand2

其中：

S 为一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见关于灵活的第二操作数的内容。

imm16 可为 0~65535 范围内的任意值。

#### 操作

MOV 指令可将 Operand2 的值复制到 Rd 中。

当 MOV 指令中的 Operand2 为一个带移位而非 LSL #0 的寄存器时，首选语法为相应的移位指令：

- ◇ 对于 MOV{S}{cond} Rd, Rm, ASR #n, ASR{S}{cond} Rd, Rm, #n 为首选语法
- ◇ 对于 MOV{S}{cond} Rd, Rm, LSL #n, LSL{S}{cond} Rd, Rm, #n 为首选语法 (n if !=0)
- ◇ 对于 MOV{S}{cond} Rd, Rm, LSR #n, LSR{S}{cond} Rd, Rm, #n 为首选语法
- ◇ 对于 MOV{S}{cond} Rd, Rm, ROR #n, ROR{S}{cond} Rd, Rm, #n 为首选语法
- ◇ 对于 MOV{S}{cond} Rd, Rm, RRX, RRX{S}{cond} Rd, Rm 为首选语法。

另外，作为移位指令的同义词，MOV 指令还允许其他形式的 Operand2：

- ◇ MOV{S}{cond} Rd, Rm, ASR Rs 与 SR{S}{cond} Rd, Rm, Rs 同义
- ◇ MOV{S}{cond} Rd, Rm, LSL Rs 与 LSL{S}{cond} Rd, Rm, Rs 同义
- ◇ MOV{S}{cond} Rd, Rm, LSR Rs 与 LSR{S}{cond} Rd, Rm, Rs 同义
- ◇ MOV{S}{cond} Rd, Rm, ROR Rs 与 ROR{S}{cond} Rd, Rm, Rs 同义

见“ASR, LSL, LSR, ROR 和 RRX”小节。

MVN 指令先获取 Operand2 的值，然后对该值按位进行逻辑“非”操作，并将结果存入 Rd。

注：MOVW 指令提供与 MOV 相同的功能，但仅限于使用 imm16 操作数。

### 限制

仅在 MOV 指令中可使用 SP 和 PC，并有下列限制：

- ◇ 第二操作数必须为一个未带移位的寄存器；
- ◇ 不得指定 S 后缀。

当 MOV 指令中 Rd 为 PC 时：

- ◇ 写入 PC 的值的位[0]被忽略；
- ◇ 发生到强制使该值的位[0]为零产生的地址的跳转。

注：尽管可以将 MOV 作为一条跳转指令，但 ARM 强烈建议使用 BX 或 BLX 指令进行跳转，以保证软件对 ARM 指令集的可移植性。

### 条件标志

如果指定了 S，则这些指令：

- ◇ 根据结果更新 N 和 Z 标志；
- ◇ Operand2 运算过程中可以更新 C 标志，见“灵活的第二操作数”小节；
- ◇ 不影响 V 标志。

### 示例

```
MOVS    R11, #0x000B    ; Write value of 0x000B to R11, flags get updated
MOV     R1, #0xFA05     ; Write value of 0xFA05 to R1, flags are not updated
MOVS    R10, R12        ; Write value in R12 to R10, flags get updated
MOV     R3, #23         ; Write value of 23 to R3
MOV     R8, SP          ; Write value of stack pointer to R8
MVNS    R2, #0xF        ; Write value of 0xFFFFFFFF0 (bitwise inverse of 0xF)
                          ; to the R2 and update flags
```

## 附录1.2.5.7 MOV T

移动到顶部。

### 语法

```
MOV T{cond} Rd, #imm16
```

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

imm16 为一个 16 位立即常数。

### 操作

MOV T 可将一个 16 位立即值 imm16 写入其目标寄存器的高半字，Rd[31:16]中。该写操作不会影响 Rd[15:0]。

可使用 MOV、MOV T 指令对生成任意的 32 位常数。

### 限制

Rd 不得为 SP 且不得为 PC。

#### 条件标志

此指令不改变标志。

#### 示例

```
MOVT R3, #0xF123 ; Write 0xF123 to upper halfword of R3, lower halfword  
; and APSR are unchanged
```

### 附录1.2.5.8 REV, REV16, REVSH 和 RBIT

在字或半字内反转字节或位的顺序。

#### 语法

op{cond} Rd, Rn

其中：

op 为下列项之一：

- REV 反转字中的字节顺序。
- REV16 独立地反转每个半字中的字节顺序。
- REVSH 反转低半字中的字节顺序，并将符号扩展
- RBIT 反转 32 位字中的位的顺序。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为存放操作数的寄存器。

#### 操作

可使用这些指令来更改数据的字节序（endianness）：

REV：将 32 位大端数据转换为小端数据，或将 32 位小端数据转换为大端数据。

REV16 将 16 位大端数据转换为小端数据，或将 16 位小端数据转换为大端数据。

REVSH 可完成以下任一转换：

将 16 位带符号大端数据转换为 32 位带符号小端数据；

将 16 位带符号小端数据转换为 32 位带符号大端数据。

#### 限制

不使用 SP 且不使用 PC。

#### 条件标志

这些指令不改变标志。

#### 示例

```
REV R3, R7 ; Reverse byte order of value in R7 and write it to R3  
REV16 R0, R0 ; Reverse byte order of each 16-bit halfword in R0
```

```

REVSH  R0, R5      ; Reverse Signed Halfword
REVHS  R3, R7      ; Reverse with Higher or Same condition
RBIT   R7, R8      ; Reverse bit order of value in R8 and write the result to R7

```

### 附录1.2.5.9 TST 和 TEQ

位测试和相等测试。

#### 语法

```

TST{cond} Rn, Operand2
TEQ{cond} Rn, Operand2

```

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rn 为存放第一个操作数的寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见“灵活的第二操作数”小节。

#### 操作

这些指令可利用 Operand2 来测试寄存器中的值。它们会更新结果的条件标志，但不会将结果写入任何寄存器中。

TST 指令对 Rn 中的值和 Operand2 的值按位进行“与”操作。除了结果会被丢弃以外，这与 ANDS 指令功能相同。

如要测试 Rn 的某个位是否为 0 或 1，使用带一个 Operand2 常数的 TST 指令，会使该位置位为 1，并清零所有其他位。

TEQ 指令对 Rn 中的值和 Operand2 的值按位进行“异或”操作。除了结果会被丢弃以外，这与 EORS 指令功能相同。

使用 TEQ 指令可在不影响 V 或 C 标志的情况下，测试两个值是否相等。

TEQ 也可用来测试值的符号。比较完毕后，两个操作数的符号位的逻辑“异或”运算结果将成为 N 标志。

#### 限制

不得使用 SP 且不得使用 PC。

#### 条件标志

这些指令：

- ◇ 根据结果更新 N 和 Z 标志；
- ◇ 在 Operand2 运算过程中能够更新 C 标志；
- ◇ 不影响 V 标志。

#### 示例

```

TST   R0, #0x3F8      ; Perform bitwise AND of R0 value to 0x3F8,
                       ; APSR is updated but result is discarded
TEQEQ R10, R9         ; Conditionally test if value in R10 is equal to
                       ; value in R9, APSR is updated but result is discarded

```

## 附录1.2.6 乘法和除法指令

下表列出了乘法和除法指令：

助记符	简要描述	参见（已改为对应章节名称）
MLA	乘加，结果为 32 位	MUL、MLA 和 MLS
MLS	乘减，结果为 32 位	MUL、MLA 和 MLS
MUL	乘法，结果为 32 位	MUL、MLA 和 MLS
SDIV	有符号除法	SDIV 和 UDIV
SMLAL	有符号乘加（32x32+64），结果为 64 位	UMULL, UMLAL, SMULL 和 SMLAL
SMULL	有符号乘法（32 x 32），结果为 64 位	UMULL, UMLAL, SMULL 和 SMLAL
UDIV	无符号除法	SDIV 和 UDIV
UMLAL	无符号乘加（32x32+64），结果为 64 位	UMULL, UMLAL, SMULL 和 SMLAL
UMULL	无符号乘法（32 x 32），结果为 64 位	UMULL, UMLAL, SMULL 和 SMLAL

附录图 39-15 乘法和除法指令

### 附录1.2.6.1 MUL、MLA 和 MLS

使用 32 位操作数的乘法、乘加和乘减，并产生一个 32 位的结果。

#### 语法

MUL{S}{cond} {Rd,} Rn, Rm ; 乘法

MLA{cond} Rd, Rn, Rm, Ra ; 乘加

MLS{cond} Rd, Rn, Rm, Ra ; 乘减

其中：

cond 为一个可选条件代码，见“条件执行”小节。

S 是一个可选后缀。如果指定了 S，则会更新操作结果的条件代码标志，见“条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn, Rm 为存放乘数的寄存器。

Ra 为存放被加数或被减数的寄存器。

#### 操作

MUL 指令可将 Rn 和 Rm 中的值相乘，并将所得结果的低 32 位存入 Rd。

MLA 指令可将 Rn 和 Rm 中的值相乘，然后再将乘积与 Ra 的值相加，最后将所得结果的低 32 位存入 Rd

MLS 指令可将 Rn 和 Rm 中的值相乘，然后再从 Ra 中的值减去乘积，最后将所得结果的低 32 位存入 Rd。

这些指令的结果与操作数是有符号还是无符号无关。



### 限制

在这些指令中，不得使用 SP 且不得使用 PC。如果 MUL 指令使用 S 后缀：

- ◇ Rd、Rn 和 Rm 必须全部都在 R0~R7 范围内；
- ◇ Rd 必须与 Rm 相同；
- ◇ 不得使用 cond 后缀。

### 条件标志

如果指定了 S，则 MUL 指令：

- ◇ 根据结果更新 N 和 Z 标志；
- ◇ 不影响 C 和 V 标志。

### 示例

```
MUL R10, R2, R5 ; Multiply, R10 = R2 x R5
MLA R10, R2, R1, R5 ; Multiply with accumulate, R10 = (R2 x R1) + R5
MULS R0, R2, R2 ; Multiply with flag update, R0 = R2 x R2
MULLT R2, R3, R2 ; Conditionally multiply, R2 = R3 x R2
MLS R4, R5, R6, R7 ; Multiply with subtract, R4 = R7 - (R5 x R6)
```

## 附录1.2.6.2 UMULL, UMLAL, SMULL 和 SMLAL

使用 32 位操作数的有符号长乘法和无符号长乘法，产生一个 64 位结果。

### 语法

op{cond} RdLo, RdHi, Rn, Rm

其中：

op 为下列项之一：

- UMULL：无符号长乘法。
- UMLAL：无符号长乘法，带累加。
- SMULL：有符号长乘法。
- SMLAL：有符号长乘法，带累加。

cond 为一个可选条件代码，见“条件执行”小节。

RdHi、RdLo 为目标寄存器。对于 UMLAL 和 SMLAL，它们还用于存放累加值。

Rn、Rm 为存放操作数的寄存器。

### 操作

UMULL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将结果的低 32 位存入 RdLo，高 32 位存入 RdHi。

UMLAL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将 64 位结果与 RdHi 和 RdLo 中的 64 位无符号整数相加，并将结果写回到 RdHi 和 RdLo。

SMULL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个

整数的乘积，然后将结果的低 32 位存入 RdLo，高 32 位存入 RdHi。

SMLAL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个整数的乘积，然后将 64 位结果与 RdHi 和 RdLo 中的 64 位有符号整数相加，并将结果写回到 RdHi 和 RdLo。

#### 限制

在这些指令中：

- ◇ 不得使用 SP 且不得使用 PC；
- ◇ RdHi 和 RdLo 必须为不同的寄存器。

#### 条件标志

这些指令不影响条件代码标志。

#### 示例

```
UMULL    R0, R4, R5, R6    ; Unsigned (R4,R0) = R5 x R6
SMLAL    R4, R5, R3, R8    ; Signed (R5,R4) = (R5,R4) + R3 x R8
```

### 附录1.2.6.3 SDIV 和 UDIV

有符号除法和无符号除法。

#### 语法

```
SDIV{cond} {Rd,} Rn, Rm
UDIV{cond} {Rd,} Rn, Rm
```

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn 为存放被除数的寄存器。

Rm 为存放除数的寄存器。

#### 操作

SDIV 对 Rn 中的值用 Rm 中的值进行有符号的整数除法。

UDIV 对 Rn 中的值用 Rm 中的值进行无符号的整数除法。

对于这两条指令来说，如果 Rn 中的值不能被 Rm 中的值整除，则结果向零舍入。

#### 限制

不得使用 SP 且不得使用 PC。

#### 条件标志

这些指令不改变标志。

#### 示例

```
SDIV    R0, R2, R4    ; Signed divide, R0 = R2/R4
UDIV    R8, R8, R1    ; Unsigned divide, R8 = R8/R1
```

## 附录1.2.7 饱和指令

本节介绍饱和指令，SSAT 和 USAT。

### 附录1.2.7.1 SSAT 和 USAT

有符号饱和到任何位位置和无符号饱和到任何位位置，可选择在饱和前进行移位。

#### 语法

op{cond} Rd, #n, Rm {, shift #s}

其中：

op 为下列项之一：

SSAT 可将一个有符号值饱和到一个有符号范围内。

USAT 可将一个有符号值饱和到一个无符号范围内。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

n 指定要饱和到的位位置：

- ◇ 对于 SSAT，n 的范围为 1~32。
- ◇ 对于 USAT，n 的范围为 0~31。

Rm 为存放待饱和值的寄存器。

shift #s 为一个可选的饱和前的移位，应用于 Rm。它必须为下列项之一：

ASR #s: 其中，s 的范围为 1~31；

LSL #s: 其中，s 的范围为 0~31。

#### 操作

这些指令饱和到一个有符号或无符号的 n 位值。

SSAT 指令会先进行指定的移位，然后将结果饱和到有符号范围  $-2^{n-1} \leq x \leq 2^{n-1}-1$ 。

USAT 指令会先进行指定的移位，然后将结果饱和到无符号范围  $0 \leq x \leq 2^n-1$ 。

对于使用 SSAT 的有符号 n 位饱和，这意味着：

- ◇ 如果待饱和的值小于  $-2^{n-1}$ ，则返回结果为  $2^{n-1}$ ；
- ◇ 如果待饱和的值大于  $2^{n-1}-1$ ，则返回结果为  $2^{n-1}-1$ ；
- ◇ 否则，返回结果与待饱和的值相同。

对于使用 USAT 的无符号 n 位饱和，这意味着：

- ◇ 如果待饱和的值小于 0，则返回结果为 0；
- ◇ 如果待饱和的值大于  $2^n-1$ ，则返回结果为  $2^n-1$ ；
- ◇ 否则，返回结果与待饱和的值相同。

如果返回结果与待饱和的值不同，则称为饱和。如果发生了饱和，该指令在 APSR 中

将 Q 标志设为 1。否则，它将保持 Q 标志不变。如要清零 Q 标志，必须使用 MSR 指令，见“MSR”小节。

如要读取 Q 标志的状态，可使用 MRS 指令，见“MRS”小节。

#### 限制

不得使用 SP 且不得使用 PC。

#### 条件标志

这些指令不影响条件代码标志。 如果发生饱和，这些指令将 Q 标志设置为 1。

#### 示例

```
SSAT    R7, #16, R7, LSL #4    ; Logical shift left value in R7 by 4, then  
                                                ; saturate it as a signed 16-bit value and  
                                                ; write it back to R7  
USATNE  R0, #7, R5             ; Conditionally saturate value in R5 as an  
                                                ; unsigned 7 bit value and write it to R0
```

## 附录1.2.8 位域指令

下表列出了对寄存器中位的相邻集（或称位域）操作的指令：

助记符	简要描述	参见（已改为对应章节名称）
BFC	位域清零	BFC 和 BFI
BFI	位域插入	BFC 和 BFI
SBFX	有符号位域提取	SBFX 和 UBFX
SXTB	有符号扩展一个字节	SXT 和 UXT
SXTH	有符号扩展一个半字	SXT 和 UXT
UBFX	无符号位域提取	SBFX 和 UBFX
UXTB	用零扩展一个字节	SXT 和 UXT
UXTH	用零扩展一个半字	SXT 和 UXT

附录图 39-16 组合和分离指令

### 附录1.2.8.1 BFC 和 BFI

位域清零和位域插入。

#### 语法

BFC{cond} Rd, #lsb, #width

BFI{cond} Rd, Rn, #lsb, #width

其中：

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域最低有效位的位置。lsb 必须在 0~31 范围之内。

width 为位域宽度，必须在 1~32-lsb 范围内。

#### 操作

BFC 清零寄存器中的一个位域。它从低位位置 lsb 开始，清零 Rd 中的 width 位。Rd 中的其他位不变。

BFI 将一个位域从一个寄存器复制到另一个寄存器。它用 Rn 从位[0]开始的 width 位替换 Rd 中从低位位置 lsb 开始的 width 位。Rd 中的其他位不变。

#### 限制

不得使用 SP 且不得使用 PC。

#### 条件标志

这些指令不影响标志。

#### 示例

BFC R4, #8, #12 ; Clear bit 8 to bit 19 (12 bits) of R4 to 0  
BFI R9, R2, #8, #12 ; Replace bit 8 to bit 19 (12 bits) of R9 with  
; bit 0 to bit 11 from R2

### 附录1.2.8.2 SBFX 和 UBFX

有符号位域提取和无符号位域提取。

#### 语法

SBFX{cond} Rd, Rn, #lsb, #width

UBFX{cond} Rd, Rn, #lsb, #width

其中:

cond 为一个可选条件代码, 见“条件执行”小节。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域中的最低有效位的位置。lsb 必须在 0~31 范围内。

width 为位域宽度, 必须在 1~32-lsb 范围内。

#### 操作

SBFX 从一个寄存器中提取一个位域, 用符号将其扩展到 32 位, 并将结果写到目标寄存器。

UBFX 从一个寄存器中提取一个位域, 用零将其扩展到 32 位, 并将结果写到目标寄存器。

#### 限制

不得使用 SP 且不得使用 PC。

#### 条件标志

这些指令不影响标志。

#### 示例

SBFX R0, R1, #20, #4 ; Extract bit 20 to bit 23 (4 bits) from R1 and sign  
; extend to 32 bits and then write the result to R0.

UBFX R8, R11, #9, #10 ; Extract bit 9 to bit 18 (10 bits) from R11 and zero  
; extend to 32 bits and then write the result to R8

### 附录1.2.8.3 SXT 和 UXT

符号扩展和零扩展。

#### 语法

SXTextend{cond} {Rd,} Rm {, ROR #n}

UXTextend{cond} {Rd}, Rm {, ROR #n}

其中:

extend 为下列项之一:

B: 将一个 8 位值扩展为 32 位值。

H: 将一个 16 位值扩展为 32 位值。

cond 为一个可选条件代码，见“条件执行”小节。

Rd 为目标寄存器。

Rm 为存放待扩展值的寄存器。

ROR #n 为下列项之一：

ROR #8: 将 Rm 中的值向右循环移 8 位。

ROR #16: 将 Rm 中的值向右循环移 16 位。

ROR #24: 将 Rm 中的值向右循环移 24 位。

如果 ROR #n 被省略，则不执行循环移位。

### 操作

这些指令执行以下操作：

1. 将 Rm 中的值向右循环移 0、8、16 或 24 位。
2. 从结果值中提取位：
  - ◇ SXTB 提取位[7:0]，并用符号扩展到 32 位。
  - ◇ UXTB 提取位[7:0]，并用零扩展到 32 位。
  - ◇ SXTH 提取位[15:0]，并用符号扩展到 32 位。
  - ◇ UXTH 提取位[15:0]，并用零扩展到 32 位。

### 限制

不得使用 SP 且不得使用 PC。

### 条件标志

这些指令不影响标志。

### 示例

SXTH	R4, R6, ROR #16	; Rotate R6 right by 16 bits, then obtain the lower ; halfword of the result and then sign extend to ; 32 bits and write the result to R4.
UXTB	R3, R10	; Extract lowest byte of the value in R10 and zero ; extend it, and write the result to R3.

## 附录1.2.9 跳转和控制指令

下表列出了跳转和控制指令：

助记符	简要描述	参见（已改为对应章节名称）
B	跳转	B、BL、BX 和 BLX
BL	带链接的跳转	B、BL、BX 和 BLX
BLX	带链接的跳转并切换指令集	B、BL、BX 和 BLX
BX	跳转并切换指令集	B、BL、BX 和 BLX
CBNZ	比较，结果非零则跳转	CBZ 和 CBNZ
CBZ	比较，结果为零则跳转	CBZ 和 CBNZ
IT	If-Then	IT
TBB	表跳转字节	TBB 和 TBH
TBH	表跳转半字	TBB 和 TBH

附录图 39-17 跳转和控制指令

### 附录1.2.9.1 B、BL、BX 和 BLX

跳转指令。

#### 语法

B{cond} lable  
BL{cond} lable  
BX{cond} Rm  
BLX{cond} Rm

其中：

B 为跳转（立即）。

BL 为带链接跳转（立即）。

BX 为跳转并切换指令集（寄存器）。

BLX 为带链接的跳转并切换指令集（寄存器）。

cond 为一个可选条件代码，见“条件执行”小节。

label 为一个相对 PC 表达式。见“相对 PC 的表达式”小节。

Rm 为一个寄存器，指示要跳转到的目标地址。Rm 中值的位[0]必须为 1，但跳转到的目标地址是通过将位[0]变为 0 生成的。

#### 操作

所有这些指令都会引发跳转，或跳转到 lable，或跳转到 Rm 所指示的地址处。此外：

- ◇ BL 和 BLX 指令可将下一条指令的地址复制到链接寄存器（LR）R14 中。
- ◇ 如果 Rm 位[0]为 0，BX 和 BLX 指令会导致 UsageFault 异常。



Bcond 标签是唯一一个既可在 IT 块内又可在 IT 块外的条件指令。所有其他跳转指令在 IT 块内都必须是有条件的，在 IT 块外必须是无条件的。

下表列出了各种跳转指令的范围。

指令	跳转范围
B label	-16MB~+16MB
Bcond label (IT 块外)	-1MB~+1MB
Bcond label (IT 块内)	-16MB~+16MB
BL{cond} label	-16MB~+16MB
BX{cond} Rm	寄存器中的任意值
BLX{cond} Rm	寄存器中的任意值

附录图 39-18 跳转范围

注：要获得最大跳转范围，可能需要使用.W 后缀。参见“指令宽度选择”小节。

### 限制

限制如下：

- ◇ BLX 指令中不使用 PC；
- ◇ 对于 BX 和 BLX，Rm 的位[0]必须为 1 才能正确执行指令，但会发生到将位[0]变为 0 所生成的目标地址的跳转。
- ◇ 如果这些指令中的任意一个处于一个 IT 块内，则它必须是 IT 块中的最后一条指令。

Bcond 是唯一一条不需要在 IT 块内的条件指令。但是，当它在 IT 块内时，具有较大的跳转范围。

### 条件标志

这些指令不改变标志。

### 示例

```

B    loopA      ; Branch to loopA
BLE  ng         ; Conditionally branch to label ng
B.W  target     ; Branch to target within 16MB range
BEQ  target     ; Conditionally branch to target
BEQ.W target    ; Conditionally branch to target within 1MB
BL   funC       ; Branch with link (Call) to function funC, return address
                        ; stored in LR
BX   LR         ; Return from function call
BXNE R0         ; Conditionally branch to address stored in R0
BLX  R0         ; Branch with link and exchange (Call) to a address stored
                        ; in R0
    
```

### 附录1.2.9.2 CBZ 和 CBNZ

比较，结果为零则跳转；比较，结果非零则跳转。

#### 语法

CBZ Rn, lable

CBNZ Rn, lable

其中：

Rn 为存放操作数的寄存器。

lable 为跳转目标。

#### 操作

可以使用 CBZ 或 CBNZ 指令避免改变条件代码标志并减少指令数目。

除了不改变条件标志外，CBZ Rn, lable 与下列指令序列功能相同：

CMP Rn, #0

BEQ lable

除了不改变条件标志外，CBNZ Rn, lable 与下列指令序列功能相同：

CMP Rn, #0

BNE lable

#### 限制

限制如下：

- ◇ Rn 必须在 R0~R7 范围内；
- ◇ 跳转目标必须在指令之后的 4 到 130 个字节之内；
- ◇ 这些指令一定不能在一个 IT 块内使用。

#### 条件标志

这些指令不改变标志。

#### 示例

CBZ R5, target ; Forward branch if R5 is zero

CBNZ R0, target ; Forward branch if R0 is not zero

### 附录1.2.9.3 IT

If-Then (IT) 条件指令。

#### 语法

IT{x{y{z}}} cond

其中：

- ◇ x 指定 IT 块中第二条指令的条件开关。
- ◇ y 指定 IT 块中第三条指令的条件开关。
- ◇ z 指定 IT 块中第四条指令的条件开关。
- ◇ cond 指定 IT 块中第一条指令的条件。

IT 块中第二、第三和第四条指令的条件开关可以是下列项之一：

- ◇ T: Then。将条件 cond 应用于指令。
- ◇ E: Else。将 cond 的相反条件应用于指令。

注：在 IT 指令中，cond 可以使用 AL (“总是 (满足)” 条件)。如果这样，则 IT 块中所有指令都必须是无条件的，且每个 x、y 和 z 都必须为 T (不能是 E) 或省略。

#### 操作

IT 指令最多可以构成 4 条后续条件指令。条件可以全部相同，或部分条件可以是其他条件的逻辑反相。IT 指令之后的条件指令构成了 IT 块。

IT 块中的指令，包括所有跳转，都必须在其语法的{cond}部分指定条件。

注：汇编器也可以自动地为条件指令生成所需 IT 指令，从而不需要自己编写。详细情况请参见所用汇编器的文档。

IT 块中的 BKPT 指令总会得到执行，即使无法满足其条件也是如此。

在一条 IT 指令和相应的 IT 块之间(或一个 IT 块内)都会发生异常。如果发生此异常，则会进入到相应的异常处理程序，同时将适用的返回信息存入 LR 和压入堆栈的 PSR 中。

可像平常一样利用异常返回指令从异常中返回，IT 块将会继续正常执行。这是 PC 修改指令跳转到一个 IT 块中的指令的唯一方法。

#### 限制

在 IT 块中不允许使用下列指令：

- ◇ IT
- ◇ CBZ 和 CBNZ
- ◇ CPSID 和 CPSIE。

使用 IT 块的其他限制有：

- ◇ 跳转指令或修改 PC 的任何指令或者必须处于 IT 块外，或者必须是 IT 块中的最后一条指令。这些指令有：

- ADD PC、PC、Rm
  - MOV PC、Rm
  - B、BL、BX、BLX
  - LDM、LDR 或任何写入 PC 的 POP 指令
  - TBB 和 TBH
- ◇ 无法跳转到 IT 块内的任何指令，除非是从一个异常处理程序返回时。
  - ◇ 除 Bcond 外，所有其他条件指令都必须在 IT 块内。Bcond 可以在 IT 块外，也可以在 IT 块内，但它在 IT 块内时具有较大的跳转范围。
  - ◇ IT 块内每条指令都必须指定一个条件代码后缀。这些条件代码后缀可以是相同的，也可以是与块中其他指令的条件的逻辑反相。

注：所用的汇编器可能对 IT 块的使用有额外的限制，例如禁止在块内使用汇编器指令。

### 条件标志

此指令不改变标志。

### 示例

```

ITTE      NE      ; Next 3 instructions are conditional
ANDNE     R0, R0, R1 ; ANDNE does not update condition flags
ADDSNE    R2, R2, #1 ; ADDSNE updates condition flags
MOVEQ     R2, R3    ; Conditional move
CMP       R0, #9    ; Convert R0 hex value (0 to 15) into ASCII
                ; ('0'-'9', 'A'-'F')

ITE       GT      ; Next 2 instructions are conditional
ADDGT     R1, R0, #55 ; Convert 0xA -> 'A'
ADDLE     R1, R0, #48 ; Convert 0x0 -> '0'
IT        GT      ; IT block with only one conditional instruction
ADDGT     R1, R1, #1 ; Increment R1 conditionally
ITTEE     EQ      ; Next 4 instructions are conditional
MOVEQ     R0, R1    ; Conditional move
ADDEQ     R2, R2, #10 ; Conditional add
ANDNE     R3, R3, #1 ; Conditional AND
BNE.W     dloop    ; Branch instruction can only be used in the last
                ; instruction of an IT block

IT        NE      ; Next instruction is conditional
ADD       R0, R0, R1 ; Syntax error: no condition code used in IT block
    
```

### 附录1.2.9.4 TBB 和 TBH

表跳转字节和表跳转半字。

### 语法

TBB [Rn, Rm]

TBH [Rn, Rm, LSL #1]

其中：

Rn 为寄存器，包含跳转长度表的地址。

如果 Rn 为 PC，则表格地址为紧跟在 TBB 或 TBH 指令后的字节的地址。

Rm 为索引寄存器。用于存放到表格的索引。对于半字表，LSL #1 将 Rm 中的值加倍，以形成表中的正确偏移量。

### 操作

这些指令可使用一个单字节偏移表 (TBB) 或半字偏移表 (TBH) 来产生一个相对 PC 的向前跳转。Rn 可提供一个表的指针，而 Rm 可提供表的索引。对于 TBB，跳转偏移量是从表返回的无符号字节值的两倍；对于 TBH，跳转偏移量是从表返回的无符号半字值的两倍。跳转到的地址是紧跟 TBB 或 TBH 指令后字节地址偏移量的地址。

### 限制

限制如下：

- ◇ Rn 不得为 SP；
- ◇ Rm 不得为 SP 且不得为 PC；
- ◇ 这些指令中的任意一个在 IT 块中使用时，它必须是 IT 块中的最后一条指令。

### 条件标志

这些指令不改变标志。

### 示例

```
ADR.W    R0, BranchTable_Byte
TBB      [R0, R1]      ; R1 is the index, R0 is the base address of the
                        ; branch table
```

Case1; an instruction sequence follows

Case2; an instruction sequence follows

Case3; an instruction sequence follows

```
BranchTable_Byte
```

```
DCB      0              ; Case1 offset calculation
```

```
DCB      ((Case2-Case1)/2) ; Case2 offset calculation
```

```
DCB      ((Case3-Case1)/2) ; Case3 offset calculation
```

```
TBH      [PC, R1, LSL #1] ; R1 is the index, PC is used as base of the
                        ; branch table
```

```
BranchTable_H
```

```
DCI      ((CaseA - BranchTable_H)/2) ; CaseA offset calculation
```

```
DCI      ((CaseB - BranchTable_H)/2) ; CaseB offset calculation
```

```
DCI      ((CaseC - BranchTable_H)/2) ; CaseC offset calculation
```

CaseA; an instruction sequence follows

CaseB; an instruction sequence follows

CaseC; an instruction sequence follows

## 附录1. 2. 10 其他指令

下表列出了 Cortex-M3 的其他指令：

助记符	简要描述	参见（已改为对应章节名称）
BKPT	断点	BKPT
CPSID	更改处理器状态，禁能中断	CPS
CPSIE	更改处理器状态，使能中断	CPS
DMB	数据内存屏障	DMB
DSB	数据同步屏障	DSB
ISB	指令同步屏障	ISB
MRS	将专用寄存器的内容移到通用寄存器中	MRS
MSR	将通用寄存器的内容移到专用寄存器中	MSR
NOP	不执行任何操作	NOP
SEV	发送事件	SEV
SVC	超级用户调用	BKPT0
WFE	等待事件	BKPT1
WFI	等待中断	BKPT2

附录图 39-19 其他指令

### 附录1. 2. 10. 1 BKPT

断点。

#### 语法

BKPT #imm

其中：

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

#### 操作

BKPT 指令可使处理器进入调试状态。当指令到达某个特定地址时，调试工具可以使用此指令来检查系统的状态。

imm 被处理器忽略。如果需要，调试器可用它来存储关于断点的附加信息。

BKPT 指令可放置在一个 IT 块内，但它会无条件执行，不受 IT 指令指定条件的影响。

#### 条件标志

此指令不改变标志。

#### 示例

```
BKPT 0xAB ; Breakpoint with immediate value set to 0xAB (debugger can
; extract the immediate value by locating it using the PC)
```

## 附录1. 2. 10. 2 CPS

更改处理器状态。

### 语法

CPSeffect iflags

其中：

effect 为下列项之一：

IE 清零专用寄存器。

ID 置位专用寄存器。

iflags 为一个或多个标志组成的序列：

i 置位或清零 PRIMASK。

f 置位或清零 FAULTMASK。

### 操作

CPS 会更改 PRIMASK 和 FAULTMASK 专用寄存器值。有关这些寄存器的更多信息，请参见“异常屏蔽寄存器”小节。

### 限制

限制如下：

- ◇ 仅可通过特权软件使用 CPS，在非特权软件中使用该指令无效。
- ◇ CPS 不能是有条件指令，因此无法用于 IT 块内。

### 条件标志

此指令不改变条件标志。

### 示例

```
CPSID i      ; Disable interrupts and configurable fault handlers (set PRIMASK)
CPSID f      ; Disable interrupts and all fault handlers (set FAULTMASK)
CPSIE i      ; Enable interrupts and configurable fault handlers (clear PRIMASK)
CPSIE f      ; Enable interrupts and fault handlers (clear FAULTMASK)
```

**附录1.2.10.3 DMB**

数据内存屏障。

**语法**

DMB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

**操作**

DMB 用作一种数据内存屏障。它可确保会先检测到程序中位于 DMB 指令前的所有显式内存访问指令，然后再检测到程序中位于 DMB 指令后的显式内存访问指令。DMB 不影响其他不访问存储器的指令的执行顺序。

**条件标志**

此指令不改变标志。

**示例**

DMB ; Data Memory Barrier

**附录1.2.10.4 DSB**

数据同步屏障。

**语法**

DSB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

**操作**

DSB 是一种特殊的数据同步内存屏障。只有当 DSB 指令执行完毕后，才会执行程序位于此指令后的指令。当 DSB 指令前的所有显式内存访问完成时，DSB 指令才会完成。

**条件标志**

此指令不改变标志。

**示例**

DSB ; Data Synchronisation Barrier



## 附录1.2.10.5 ISB

指令同步屏障。

### 语法

ISB{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

### 操作

ISB 用作指令同步屏障。它可刷新处理器中的流水线，以便在 ISB 指令完成后，才从缓存或存储器中提取位于该指令后的所有其他指令。

### 条件标志

此指令不改变标志。

### 示例

```
ISB ; Instruction Synchronisation Barrier
```

## 附录1.2.10.6 MRS

将一个专用寄存器的内容移到通用寄存器。

### 语法

MRS{cond} Rd, spec\_reg

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

Rd 为目标寄存器。

spec\_reg 可以是以下任意一个：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI\_MAX、FAULTMASK 或 CONTROL。

### 操作

可将 MRS 与 MSR 配合使用作为一个更新 PSR 的读取-修改-写入序列的一部分，例如清除 Q 标志。

在进程交换代码中，必须保存换出进程的编程模型状态，包括 PSR 的相关内容。同样，也必须恢复换入进程的状态。这些操作将 MRS 用于状态保存指令序列，将 MSR 用于状态恢复指令序列。

注：与 MRS 指令共同使用时，BASEPRI\_MAX 为 BASEPRI 的一个别名。参见“MSR”小节。

### 限制

Rd 不得为 SP 且不得为 PC。

### 条件标志

此指令不改变标志。

### 示例

```
MRS R0, PRIMASK ; Read PRIMASK value and write it to R0
```

## 附录1.2.10.7 MSR

将一个通用寄存器的内容移到指定的专用寄存器。

### 语法

MSR{cond} spec\_reg, Rn

其中:

cond 为一个可选的条件代码, 见“条件执行”小节。

Rn 为源寄存器。

spec\_reg 可以是以下任意一个: APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI\_MAX、FAULTMASK 或 CONTROL。

### 操作

MSR 中寄存器访问操作取决于特权级别。非特权软件只能访问 APSR, 见表“APSR 位分配”。特权软件可访问所有专用寄存器。

在非特权软件中, 对 PSR 中未分配状态位或执行状态位的写入被忽略。

当写入 BASEPRI\_MAX 时, 指令仅在以下情况之一时才写入 BASEPRI:

- ◇ Rn 为非零值且当前 BASEPRI 值为 0;
- ◇ Rn 为非零值且小于当前 BASEPRI 值。参见“MRS”小节。

### 限制

Rn 不得为 SP 且不得为 PC。

### 条件标志

此指令根据 Rn 中的值显式更新标志。

### 示例

MSR CONTROL, R1 ; Read R1 value and write it to the CONTROL register

## 附录1.2.10.8 NOP

不执行任何操作。

### 语法

NOP{cond}

其中:

cond 为一个可选条件代码, 见“条件执行”小节。

### 操作

NOP 不执行任何操作。NOP 未必一定是消耗时间的 NOP。也许在该指令执行前, 处理器就会将其从流水线中删除。

可以使用 NOP 进行填充 (padding), 例如将后续指令置于 64 位边界上。

### 条件标志

此指令不改变标志。

### 示例

NOP ; No operation

### 附录1.2.10.9 SEV

发送事件。

#### 语法

SEV{cond}

其中：

cond 为一个可选的条件代码，见“条件执行”小节。

#### 操作

SEV 是一条提示指令，向一个多处理器系统中的所有处理器发送事件信号。它还会将本地事件寄存器设置为 1，见“电源管理”小节。

#### 条件标志

此指令不改变标志。

#### 示例

SEV ; Send Event

### 附录1.2.10.10 SVC

超级用户调用。

#### 语法

SVC{cond} #imm

其中：

cond 为一个可选条件代码，见“条件执行”小节。

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

#### 操作

SVC 指令会引发一个 SVC 异常。

Imm 会被处理器忽略。如果需要，异常处理程序会将其恢复，借以确定所请求的服务。

#### 条件标志

此指令不改变标志。

#### 示例

SVC 0x32 ; Supervisor Call (SVC handler can extract the immediate value  
; by locating it via the stacked PC)

**附录1.2.10.11 WFE**

等待事件。

**语法**

WFE{cond}

其中：

cond 为一个可选条件代码，见“条件执行”小节。

**操作**

WFE 为一条提示指令。

如果事件寄存器为 0，WFE 会挂起程序执行，直到发生任一以下事件后再恢复执行：

- ◇ 一个异常，除非被异常屏蔽寄存器或当前优先级所屏蔽；
- ◇ 一个异常进入挂起状态，前提是系统控制寄存器中的 SEVONPEND 位置位；
- ◇ 一个调试进入请求，前提是调试已使能。
- ◇ 一个外设或多处理器系统中的另一个处理器使用 SEV 指令发出了一个事件信号。

如果事件寄存器为 1，WFE 将其清零并立即返回。

更多信息参见“电源管理”小节。

**条件标志**

此指令不改变标志。

**示例**

WFE ; Wait for event

**附录1.2.10.12 WFI**

等待中断。

**语法**

WFI{cond}

其中：

cond 为一个可选条件代码，见“条件执行”小节。

**操作**

WFI 是一条提示指令，它会挂起程序执行，直到发生以下事件之一：

- 一个异常；
- 一个调试进入请求，无论是否使能了调试。

**条件标志**

此指令不改变标志。

**示例**

WFI ; Wait for interrupt

## 附录1.3 ARM Cortex-M3 用户指南：处理器

### 附录1.3.1 编程模型

本节描述了 Cortex-M3 的编程模型。除了对各个内核寄存器的描述外，它还包括关于处理器模式以及软件执行和堆栈的特权等级的信息。

#### 附录1.3.1.1 处理器模式和软件执行的特权等级

处理器模式包括：

◇ 线程模式

用于执行应用软件。处理器在退出复位时进入线程模式。

◇ 处理模式

用于处理异常。处理器在完成异常处理后返回线程模式。

软件执行的特权等级有：

◇ 非特权

软件：

- 对 MSR 和 MRS 指令的有限访问权限，且不能使用 CPS 指令；
- 不能访问系统定时器、NVIC 或系统控制模块；
- 可能限制对存储器或外设的访问。

非特权软件在非特权等级执行。

◇ 特权

软件可使用所有指令，并可访问全部资源。

特权软件在特权等级执行。

在线程模式中，控制寄存器控制着软件执行是有特权的还是非特权的，见表“控制寄存器的位分配”。在处理模式中，软件执行总是有特权的。

只有特权软件才可写入控制寄存器，以改变线程模式下软件执行的特权等级。非特权软件可使用 SVC 指令执行超级用户调用，以将控制权转移给特权软件。

#### 附录1.3.1.2 堆栈

处理器使用满降序堆栈。也就是说，堆栈指针指示最后推入堆栈存储器的项。当处理器将一个新项推入堆栈时，它会递减堆栈指针，然后将该项写入新的存储单元。处理器实现了两个堆栈：主堆栈和进程堆栈，它们有独立的堆栈指针副本，见“堆栈指针”小节。

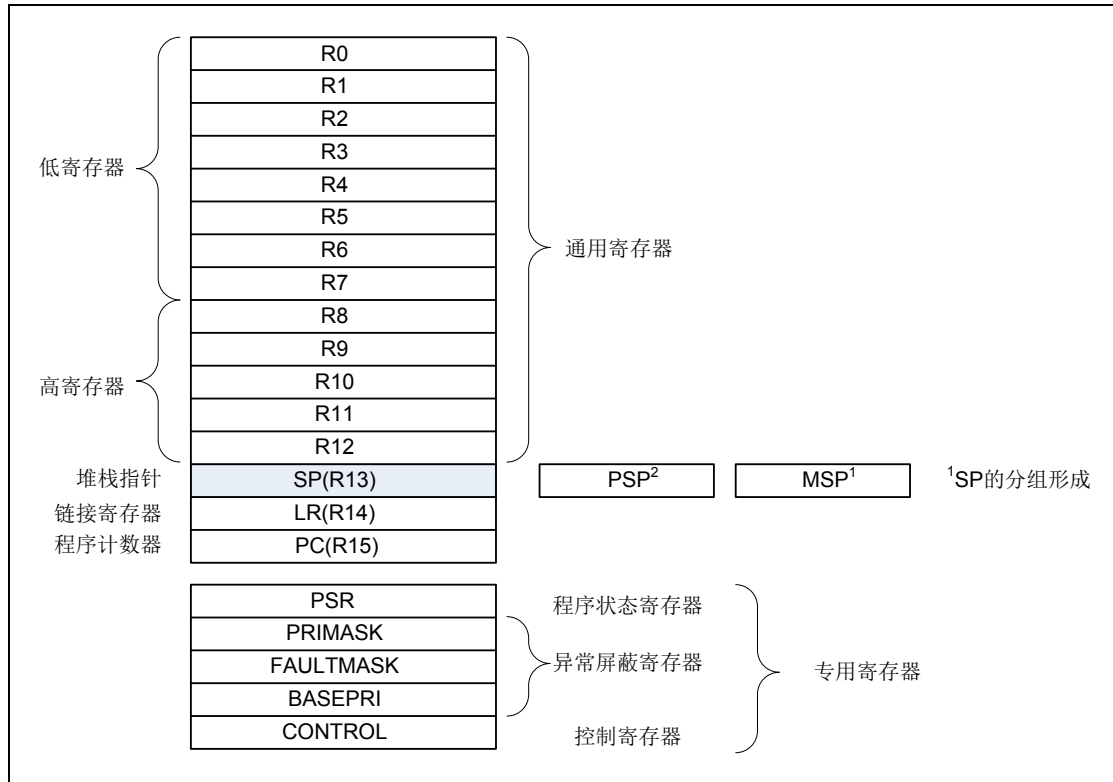
在线程模式下，控制寄存器控制着处理器是使用主堆栈还是进程堆栈，见“控制寄存器的位分配”。在处理模式下，处理器总是使用主堆栈。处理器工作的选项有：

处理器模式	用于执行	软件执行的特权级别	使用的堆栈
线程	应用程序	特权或非特权 <sup>[1]</sup>	主堆栈或进程堆栈 <sup>[1]</sup>
处理	异常处理程序	总是特权	主堆栈

附录图 39-20 处理器模式、执行特权级别和堆栈使用选择汇总

### 附录1.3.1.3 内核寄存器

处理器的内核寄存器有：



附录图 39-21 内核寄存器

名称	类型 <sup>[1]</sup>	所需特权 <sup>[2]</sup>	复位值	描述 (已改为对应章节名称)
R0-R12	RW	两者皆可	未定义	通用寄存器
MSP	RW	特权	参见描述章节	主堆栈指针
PSP	RW	两者皆可	未定义	进程堆栈指针
LR	RW	两者皆可	0xFFFFFFFF	链接寄存器
PC	RW	两者皆可	参见描述章节	程序计数器
PSR	RW	特权	0x01000000	程序状态寄存器
ASPR	RW	两者皆可	0x00000000	见表“APSR 位分配”
IPSR	R	特权	0x00000000	见表“IPSR 位分配”
EPSR	R	特权	0x01000000	见表“EPSR 位分配”
PRIMASK	RW	特权	0x00000000	见表“PRIMASK 寄存器的位分配”
FAULTMASK	RW	特权	0x00000000	见表“FAULTMASK 寄存器的位分配”
BASEPRI	RW	特权	0x00000000	见表“BASEPRI 寄存器的位分配”

名称	类型 <sup>[1]</sup>	所需特权 <sup>[2]</sup>	复位值	描述（已改为对应章节名称）
CONTROL	RW	特权	0x00000000	见表“控制寄存器的位分配”

附录图 39-22 内核寄存器集汇总

注[1]: 描述线程模式和处理模式下程序执行过程中的访问类型。调试访问可以不同。

注[2]: “两者皆可”意味着特权或非特权软件都可以访问该寄存器。

### 通用寄存器

R0-R12 为用于数据操作的 32 位通用寄存器。

### 堆栈指针

堆栈指针（SP）为寄存器 R13。在线程模式下，控制寄存器中的位[1]表示所使用的堆栈指针：

复位时，处理器将地址 0x00000000 的值加载到 MSP。

- ◇ 0 = 主堆栈指针（MSP）。这是复位值。
- ◇ 1 = 进程堆栈指针（PSP）。

### 链接寄存器

链接寄存器（LR）为寄存器 R14。它存储子程序、函数调用和异常的返回信息。复位时，处理器加载 LR 值 0xFFFFFFFF。

### 程序计数器

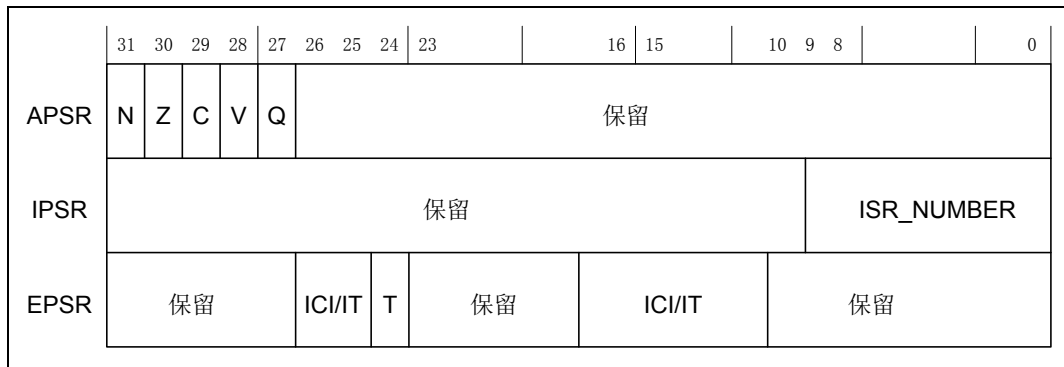
程序计数器（PC）为寄存器 R15。它包含当前的程序地址。位[0]总是为 0，因为指令的取指必须按半字对齐。复位时，处理器用复位向量的值加载 PC，复位向量在地址 0x00000004。

### 程序状态寄存器

程序状态寄存器（PSR）组合了：

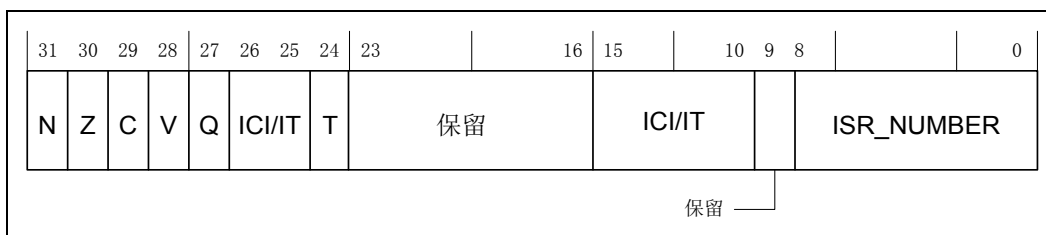
- ◇ 应用程序状态寄存器（APSR）
- ◇ 中断程序状态寄存器（IPSR）
- ◇ 执行程序状态寄存器（EPSR）。

这些寄存器在 32 位 PSR 中为互斥的位域。位的分配如下：



附录图 39-23 APSR、IPSR 和 EPSR

PSR 位分配如下:



附录图 39-24 组合 xPSR

单独地访问这些寄存器，或访问任意两或三个寄存器的组合，可用寄存器名作为 MSR 或 MRS 指令的一个参数。例如：

- ◇ 使用带有 MRS 指令的 PSR 读取全部寄存器；
- ◇ 使用带有 MSR 指令的 APSR 写入 APSR。

PSR 组合和属性如下：

寄存器	类型	组合
PSR	RW <sup>[1][2]</sup>	APSR、EPSR 与 IPSR
IEPSR	R	EPSR 与 IPSR
IAPSR	RW <sup>[1]</sup>	APSR 与 IPSR
EAPSR	RW <sup>[2]</sup>	APSR 与 EPSR

附录图 39-25 PSR 寄存器组合

注[1]: 处理器忽略 IPSR 位的写入值。

注[2]: 对 EPSR 位进行读操作返回结果为零且处理器忽略这些位的写入值。

关于如何访问程序状态寄存器的更多信息，请参见指令描述“MRS”和“MSR”小节。

应用程序状态寄存器：APSR 包含了前面指令执行的条件标志的当前状态。其属性请参见“内核寄存器集汇总”表中寄存器汇总。位分配如下：

位	名称	功能
[31]	N	负数或小于标志：
		0: 操作结果为正数、零、大于或等于
		1: 操作结果为负数或小于
[30]	Z	零标志：
		0: 操作结果非零
		1: 操作结果为零
[29]	C	进位或借位标志：
		0: 加法操作未产生进位或减法操作产生了一个借位



位	名称	功能
		1: 加法操作产生了一个进位或减法操作未产生借位
[28]	V	溢出标志
		0: 操作未导致溢出发生
		1: 操作导致溢出发生
[27]	Q	粘着饱和 (sticky saturation) 标志:
		0: 复位或该位被最终清零后没有出现饱和
		1: SSAT 或 USAT 指令导致了饱和的出现
		该位由软件使用 MRS 指令清零。
[26:0]	-	保留

附录图 39-26 APSR 位分配

中断程序状态寄存器: IPSR 包含当前中断服务程序 (ISR) 的异常类型号。其属性请参见“内核寄存器集汇总”表中寄存器汇总。位分配如下:

位	名称	功能
[31:9]	-	保留
[8:0]	ISR_NUMBER	<p>该位为当前异常的序号。</p> <p>0 = 线程模式</p> <p>1 = 保留</p> <p>2 = NMI</p> <p>3 = HardFault</p> <p>4 = 存储器管理故障</p> <p>5 = 总线故障</p> <p>6 = 使用故障</p> <p>7~10 = 保留</p> <p>11 = SVCcall</p> <p>12 = 保留供调试使用</p> <p>13 = 保留</p> <p>14 = PendSV</p> <p>15 = SysTick</p> <p>16 = IRQ0</p> <p>17 = IRQ1 (第一个设备专用中断)</p> <p>.....</p> <p>255 = IRQ243 (最后执行的中断, 取决于设备)</p>

		更多信息请参见 “异常类型” 小节。
--	--	--------------------

附录图 39-27 IPSR 位分配

执行程序状态寄存器：EPSR 包含以下所列项之一的 Thumb 状态位和执行状态位：

- ◇ If-Then (IT) 指令
- ◇ 用于一条中断多个加载或多个存储指令的可中断-可继续指令 (ICI) 域。

EPSR 属性，请参见“内核寄存器集汇总”表中寄存器汇总。位分配为：

位	名称	功能
[31:27]	-	保留。
[26:25], [15:10]	ICI	可中断-可继续的指令位，参见“Cortex 微控制器软件接口标准”小节。
[26:25], [15:10]	IT	IT 指令的执行状态位，参见“IT”小节。
[24]	T	一直设置为 1。
[23:16]	-	保留。
[9:0]	-	保留。

附录图 39-28 EPSR 位分配

如下图用 MSR 指令，直接通过应用软件读取 EPSR，则永远返回零。试图在应用软件中使用 MSR 指令写 EPSR，则会被忽略。故障处理程序可检查推入堆栈的 PSR 中 EPSR 的值，以指示出有故障的操作。参见异常进入和返回”小节。

可中断-可继续指令：当执行 LDM 或 STM 指令期间发生中断时，处理器执行下列操作：

处理中断以后，处理器：

- ◇ 暂时停止多个加载或多个存储指令的运行。
- ◇ 将多个操作中的下一个寄存器操作数保存到 EPSR 的位[15:12]。
- ◇ 返回位[15:12]指向的寄存器。
- ◇ 恢复执行多个加载或存储指令。

当 EPSR 保存有 ICI 执行状态时，位[26:25, 11:10]为零。

If-Then 块：If-Then 块在一个 16 位 IT 指令后可跟随最多 4 条指令。块中的每条指令都是有条件的。指令条件既可以全部相同，也可以是其它条件的逻辑反。更多信息请参见 “IT” 小节。

#### 异常屏蔽寄存器

异常屏蔽寄存器禁止由处理器处理异常。当异常可能影响到时序关键任务时禁用异常。

用 **use MSR** 和 **MRS** 指令访问异常屏蔽寄存器，或使用 **CPS** 指令来改变 **PRIMASK** 或 **FAULTMASK** 的值。更多信息请参见“**MRS**”、“**MSR**”和“**CPS**”小节。

优先级屏蔽寄存器：**PRIMASK** 寄存器能阻止所有可配置优先级的异常的激活。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:1]	-	保留
[0]	PRIMA	0: 无影响
	SK	1: 阻止所有可配置优先级的异常的激活

附录图 39-29 PRIMASK 寄存器的位分配

故障屏蔽寄存器：故障屏蔽寄存器阻止除不可屏蔽中断（**NMI**）外所有异常的激活。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:1]	-	保留
[0]	FAULTMASK	0: 无影响
		1: 阻止除 <b>NMI</b> 外其他异常的激活

附录图 39-30 FAULTMASK 寄存器的位分配

当从任意异常处理程序退出时（**NMI** 处理程序除外），处理器将 **FAULTMASK** 位清零。

基本优先级屏蔽寄存器：**BASEPRI** 寄存器定义了针对异常处理的最低优先级。当 **BASEPRI** 设置为一个非零值时，它阻止激活所有与 **BASEPRI** 值相同或更低优先级的异常。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:8]	-	保留
[7:0]	<b>BASEPRI</b> <sup>[1]</sup>	优先级屏蔽位： 0x0000: 无影响 非零值：定义了异常处理的基本优先级 处理器不处理那些优先级值大于或等于 <b>BASEPRI</b> 值的异常。

附录图 39-31 BASEPRI 寄存器的位分配

注[1]: 该域与中断优先级寄存器中的优先级域相似。处理器仅执行该域的位[7:M]，位[M-1:0]被读为 0 且忽略写入值。M 值取决于所选设备。更多信息参见“中断优先级寄存器”小节。请记住，较高优先级域值对应较低异常优先级。

### 控制寄存器

当处理器处于线程模式时，控制寄存器控制所用堆栈和软件执行的特权等级。其属性见“内核寄存器集汇总”表中寄存器汇总。位分配见下表。

位	名称	功能
[31:2]	-	保留
[1]	有效堆栈指针	定义了当前堆栈： 0: MSP 是当前堆栈指针 1: PSP 是当前堆栈指针在处理模式下，该位读取为 0 且忽略写入值。
[0]	线程模式特权级别	定义了线程模式下的特权级别： 0: 特权 1: 非特权

附录图 39-32 控制寄存器的位分配

处理模式始终使用 MSP，因此，当在处理模式下时，处理器会忽略对控制寄存器有效堆栈指针位的显式写入。异常进入和返回机制会更新控制寄存器。

在一个 OS 环境中，ARM 建议线程模式下运行的线程使用进程堆栈与内核，而异常处理程序使用主堆栈。

默认情况下，线程模式使用 MSP。如要将线程模式下使用的堆栈指针切换到 PSP，使用 MSR 指令将“有效”堆栈指针位设置为 1，见“MSR”小节。

注：改变堆栈指针时，软件必须在 MSR 指令后立即使用一条 ISB 指令。这样可确保 ISB 后的指令使用新的堆栈指针执行操作。参见“ISB”小节。

#### 附录1.3.1.4 异常和中断

Cortex-M3 处理器支持中断和系统异常。处理器和可嵌套向量中断控制器 (NVIC) 确定所有异常的优先级并进行处理。一个异常会改变软件控制的正常流程。处理器使用处理模式来处理除复位外的所有异常。更多信息参见“异常进入”和“异常返回”小节。

NVIC 寄存器控制中断处理。更多信息参见“可嵌套向量中断控制器”小节。

### 附录1.3.1.5 数据类型

处理器:

- ◇ 支持以下数据类型:
  - 32 位字
  - 16 位半字
  - 8 位字节
- ◇ 支持 64 位数据传输指令。
- ◇ 以小端形式管理所有的数据内存访问。更多信息参见“存储区、类型和属性”小节。

### 附录1.3.1.6 Cortex 微控制器软件接口标准

对于 Cortex-M3 微控制器系统，Cortex 微控制器软件接口标准（CMSIS）定义了：

- ◇ 一种通用方式，用来：
  - 访问外设寄存器；
  - 定义异常向量。
- ◇ 以下项的名称：
  - 内核外设的寄存器；
  - 内核异常向量。
- ◇ 用于 RTOS 内核的设备独立接口，包括一个调试通道。

CMSIS 包括 Cortex-M3 处理器中内核外设的地址定义和数据结构。它还包括可选的中间件组件接口，中间件组件包括 TCP/IP 堆栈和 Flash 文件系统。

CMSIS 能够重用各个中间件厂商的模板代码，以及符合 CMSIS 要求的软件组件组合，从而简化了软件开发工作。软件厂商可扩充 CMSIS，包括加入其外设定义，以及这些外设的访问函数。

本文件包括了 CMSIS 中定义的寄存器名称，并简要描述了用于访问处理器内核和内核外设的 CMSIS 函数。

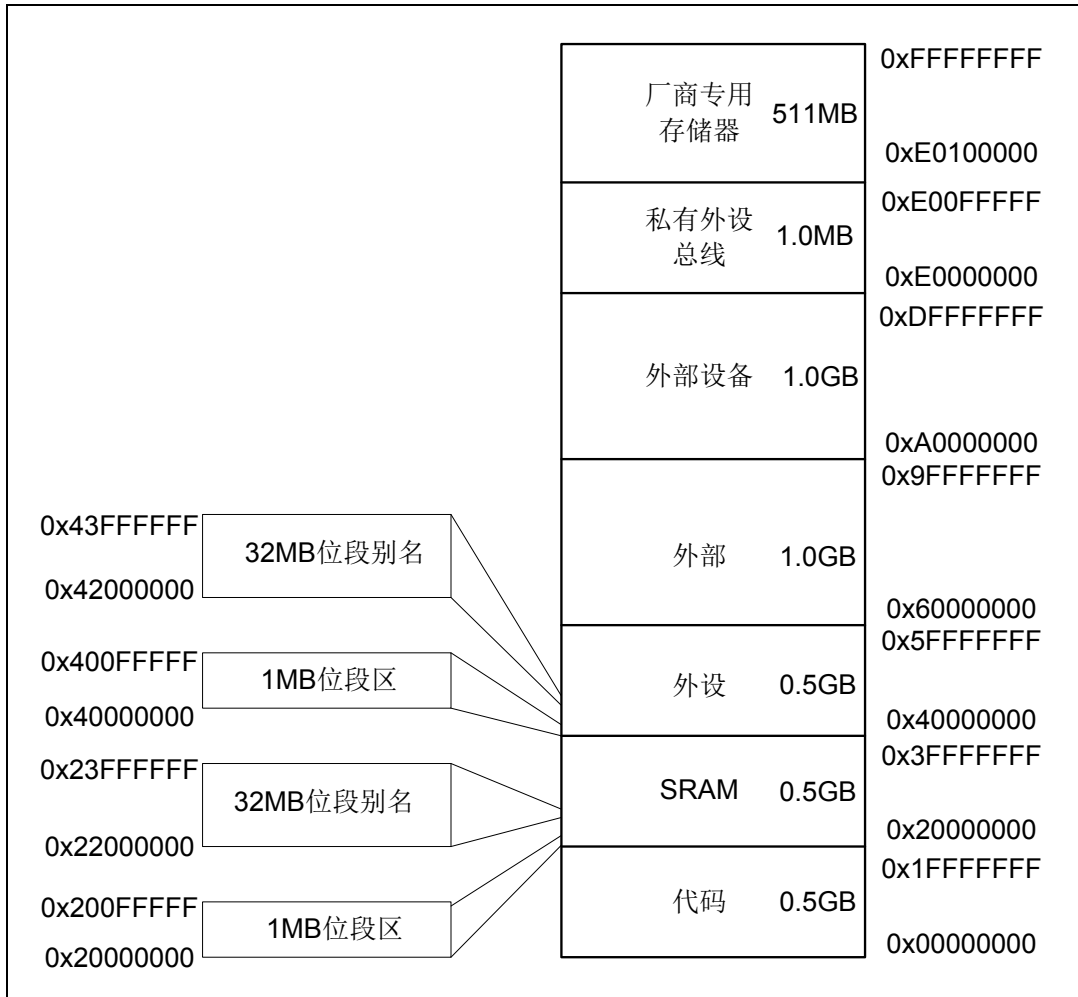
注：本文件使用了 CMSIS 定义的寄存器的简称。在个别情况下，这些名称可能不同于其他文件中使用的结构上的简称。

以下各节给出了关于 CMSIS 的更多信息：

- ◇ “电源管理编程提示”
- ◇ “内在函数”
- ◇ “Cortex-M3 NVIC 寄存器的 CMSIS 映射”
- ◇ “NVIC 编程提示”。

### 附录1.3.2 存储器模型

本节描述了处理器存储器映射、存储器访问行为，以及位段（bit-banding）特性。处理器具有固定存储器映射，可提供高达 4GB 可寻址存储器。存储器映射为：



附录图 39-33 Cortex-M3 处理器预定义的存储器映射

SRAM 和外设区包括了位段区。位段提供了对位数据的原子操作，见“位段”小节。

处理器将私有外设总线（PPB）区保留用于内核外设寄存器，见“关于 Cortex-M3 外设”小节。

### 附录1.3.2.1 存储区、类型和属性

MPU 的存储器映射和编程将存储器映射分隔为区。每个区都有一个确定的存储器类型，有些区有附加的存储器属性。存储器类型和属性决定了对该区访问的行为。

存储器类型有：

- ◇ 正常型：处理器可将事务重新排序以改善效率，或执行猜读（speculative read）。
- ◇ 设备型：处理器保持相对于其他设备或非常有序型存储器的事务顺序。
- ◇ 非常有序型：处理器保持相对于全部其他事务的事务顺序。

设备型和非常有序型存储器有不同的排序要求，这意味着存储器系统可以缓冲对设备型存储器的写入，但不得缓冲对非常有序型存储器的写入。

其他存储器属性包括：

- ◇ 可共享

对于一个可共享存储区，存储器系统可在一个系统的总线主机与多个总线主机之间（例如处理器与 DMA 控制器之间）提供数据同步。

非常有序型存储器总是可共享的。

如果多个总线主机可访问一个不可共享的存储区，则软件必须保证多个总线主机之间的数据一致性。

- ◇ 从不执行（XN）

表示处理器阻止指令的访问。任何从 XN 区获取指令的尝试都会导致一个存储器管理故障异常。

### 附录1.3.2.2 存储器系统访问的排序

对于多数显式存储器访问指令引起的存储器访问，只要不影响指令序列的行为，存储器系统就不会保证访问完成的顺序与指令的程序顺序相匹配。通常，如果程序正确执行依赖于两个存储器访问按程序的顺序完成，则软件必须在存储器访问指令之间插入一条存储器屏障指令，见“存储器访问的软件排序”小节。

但是，存储器系统一定要保证对设备型和非常有序型存储器的一定访问次序。对于两条存储器访问指令 A1 和 A2，如果在程序顺序中，A1 在 A2 之前发生，则两条指令产生的存储器访问次序为：

A2 \ A1	正常访问	设备访问		非常有序访问
		非共享	可共享	
正常访问	-	-	-	-
设备访问，非共享	-	<	-	<
设备访问，可共享	-	-	<	<
非常有序访问	-	<	<	<

附录图 39-34 存储器排序限制

其中：

“-”表示存储器系统不保证访问次序。

“<”表示访问依照了程序顺序，即，A1 总在 A2 之前执行。

### 附录1.3.2.3 存储器访问行为

存储器映射中各区的访问行为如下：

地址范围	存储区	存储器类型	XN	描述
0x00000000~ 0x1FFFFFFF	代码	正常 <sup>[1]</sup>	-	编程代码的可执行区。也可在此放置数据。
0x20000000~ 0x3FFFFFFF	SRAM	正常 <sup>[1]</sup>	-	数据的可执行区。也可在此放置代码。该区包含位段和位段别名区，参见表“SRAM 存储器位段区”
0x40000000~ 0x5FFFFFFF	外设	设备 <sup>[1]</sup>	XN <sup>[1]</sup>	该区包含位段区和位段别名区，参见表“外设存储器位段区”
0x60000000~ 0x9FFFFFFF	外部 RAM	正常 <sup>[1]</sup>	-	数据的可执行区。
0xA0000000~ 0xDFFFFFFF	外部设备	设备 <sup>[1]</sup>	XN <sup>[1]</sup>	外部设备存储器。
0xE0000000~ 0xE0FFFFFF	私有外设总线	非常有序 <sup>[1]</sup>	XN <sup>[1]</sup>	该区包含 NVIC、系统定时器和系统控制模块。
0xE0100000~ 0xFFFFFFFF	厂商专用设备	设备 <sup>[1]</sup>	XN <sup>[1]</sup>	不用于 ES32 设备。

附录图 39-35 存储器访问行为

注[1]：详细信息请参见“存储区、类型和属性”小节。

代码、SRAM 和外部 RAM 区可保存程序。然而，最有效的程序访问是从代码区。这是因为处理器有单独的总线，能够同时进行指令获取和数据访问。

MPU 可以覆盖本节所述的默认存储器访问行为。更多信息见存储器保护单元”小节。



#### 附录1.3.2.4 存储器访问的软件排序

程序流中指令顺序并不是总能保证对应存储器事务的顺序。这是因为：

- ◇ 只要不影响指令序列的行为，处理器可将一些存储器访问重新排序来提高效率。
- ◇ 处理器有多个总线接口。
- ◇ 存储器映射中的存储器或设备具有不同的等待状态。
- ◇ 有些存储器访问是缓冲的或者是猜测的。

“存储器系统访问的排序”小节描述了存储器系统保证存储器访问顺序的情况。否则，如果存储器访问顺序很关键，软件必须包括存储器屏障指令以强制排序。处理器提供以下存储器屏障指令：

- ◇ **DMB**

数据内存屏障(DMB)指令确保未完成的存储器事务先于后续存储器事务完成。见“DMB”小节。

- ◇ **DSB**

数据同步屏障(DSB)指令确保未完成的存储器事务在执行后续指令前完成。请参见“DSB”小节。

- ◇ **ISB**

指令同步屏障 (ISB) 确保全部完成的存储器事务的效果可以被后续指令识别。请参见“ISB”小节。

在下列示例情况下可使用存储器屏障指令：

对非常有序型存储器（例如系统控制模块）的存储器访问不需要使用 **DMB** 指令。

- ◇ **MPU 编程：**

- 使用一个 **DSB** 指令，确保在上下文切换结束时 **MPU** 立即生效。
- 如果用一个跳转或调用访问了 **MPU** 配置代码，则使用一个 **ISB** 指令，以确保在一个或多个 **MPU** 区编程后，新的 **MPU** 设置立即生效。如果是通过异常机制输入 **MPU** 配置代码，则不需要 **ISB** 指令。

- ◇ **向量表。**如果程序改变了一个向量表项，然后使能了对应的异常，则要在两个操作间使用 **DMB** 指令。这样可以确保如果异常在使能后被立即获取，处理器使用新的异常向量。

- ◇ **自修改代码。**如果程序包含自修改代码，则在程序中的代码修改之后要立即使用一条 **ISB** 指令。这样可确保后续指令执行时使用更新后的程序。

- ◇ **存储器映射切换。**如果系统包含一个存储器映射切换机制，则在切换程序中的存储器映射后使用 **DSB** 指令。这样可确保后续指令执行时使用更新后的存储器映射。

- ◇ **动态异常优先级更改。**如果在异常挂起或有效期间，必须更改一个异常的优先级，则在更改后使用 **DSB** 指令。这样可确保更改在 **DSB** 指令完成时生效。

- ◇ **在多主机系统中使用一个信号量。**如果系统包含一个以上的总线主机（例如系统中有另一个处理器），则每个处理器都必须要在任何信号量指令后使用一个 **DMB** 指令，才能确保其他总线主机能按照执行的顺序看到存储器事务。

### 附录1.3.2.5 位段

位段区将一个位段别名区中的每个字映射到位段区中的单个体。位段区占用 SRAM 和外设存储区的最低 1MB。

存储器映射中有两个 32MB 的别名区，它们映射到两个 1MB 的位段区：

- ◇ 对 32MB SRAM 别名区的访问映射到 1MB SRAM 位段区，如下表“SRAM 存储器位段区”所示；
- ◇ 对 32MB 外设别名的访问映射到 1MB 外设位段区，如下表“外设存储器位段区”所示。

地址范围	存储区	指令与数据访问
0x20000000-0x200FFFFFF	SRAM 位段区	直接访问此存储区与访问 SRAM 存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x22000000-0x23FFFFFFF0	SRAM 位段别名区	对此区的数据访问被重映射到位段区。写操作的效果相当于读取-修改-写入。指令访问不能重映射。

附录图 39-36 SRAM 存储器位段区

地址范围	存储区	指令与数据访问
0x40000000~0x400FFFFFF	外设位段别名区	对此存储区的直接访问与访问外设存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x42000000~0x44FFFFFFF	外设位段区	对此区的数据访问被重映射到位段区。写操作的效果相当于 读取-修改-写入。不允许指令访问。

附录图 39-37 外设存储器位段区

注：对 SRAM 或外设位段别名区的字访问映射到 SRAM 或外设位段区中的单个体。

以下公式给出了别名区映射到位段区的方式：

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

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + \text{bit\_word\_offset}$$

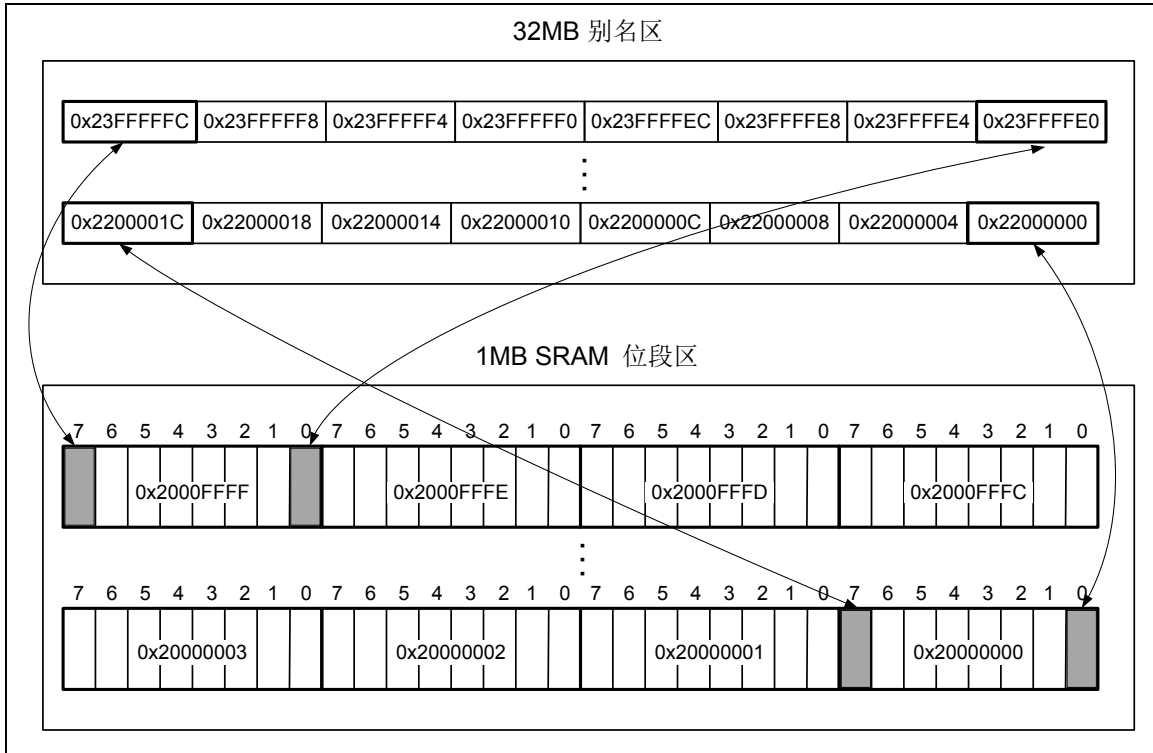
其中：

- ◇ bit\_word\_offset 是位段存储区中目标位的位置。
- ◇ bit\_word\_addr 是映射到目标位的字在别名存储区中的地址。
- ◇ bit\_band\_base 是别名区的起始地址。
- ◇ byte\_offset 是包含目标位的位段区的字节号。
- ◇ bit\_number 是目标位的位置 0-7。

下图给出了 SRAM 位段别名区和 SRAM 位段区之间的位段映射示例：

- ◇ 0x23FFFFE0 处的别名字映射到 0x200FFFFFF 处位段字节的位[0]：

- ◇  $0x23FFFFE0 = 0x22000000 + (0xFFFF*32) + (0*4)$ 。
- ◇  $0x23FFFFFC$  处的别名字映射到  $0x200FFFFF$  处位段字节的位[7]:
- ◇  $0x23FFFFFC = 0x22000000 + (0xFFFF*32) + (7*4)$ 。
- ◇  $0x22000000$  处的别名字映射到  $0x20000000$  处位段字节的位[0]:
- ◇  $0x22000000 = 0x22000000 + (0*32) + (0*4)$ 。
- ◇  $0x2200001C$  处的别名字映射到  $0x20000000$  处位段字节的位[7]:
- ◇  $0x2200001C = 0x22000000 + (0*32) + (7*4)$ 。



附录图 39-38 位段映射

### 直接访问一个别名区

向别名区中一个字执行写操作会更新位段区中的单个位。

向别名区中一个字写入值的位[0]决定了写入位段区中目标位的值。写入一个位[0]设为 1 的值，则会向位段位写入一个 1，写入一个位[0]设为 0 的值，则会向位段位写入一个 0。

别名字的位[31:1]对位段位无影响。写入 0x01 与写入 0xFF 效果相同。写入 0x00 与写入 0x0E 效果相同。

读取别名区中一个字：

- ◇ 0x00000000 表示位段区中目标位被设为 0；
- ◇ 0x00000001 表示位段区中目标位被设为 1。

### 直接访问一个位段区

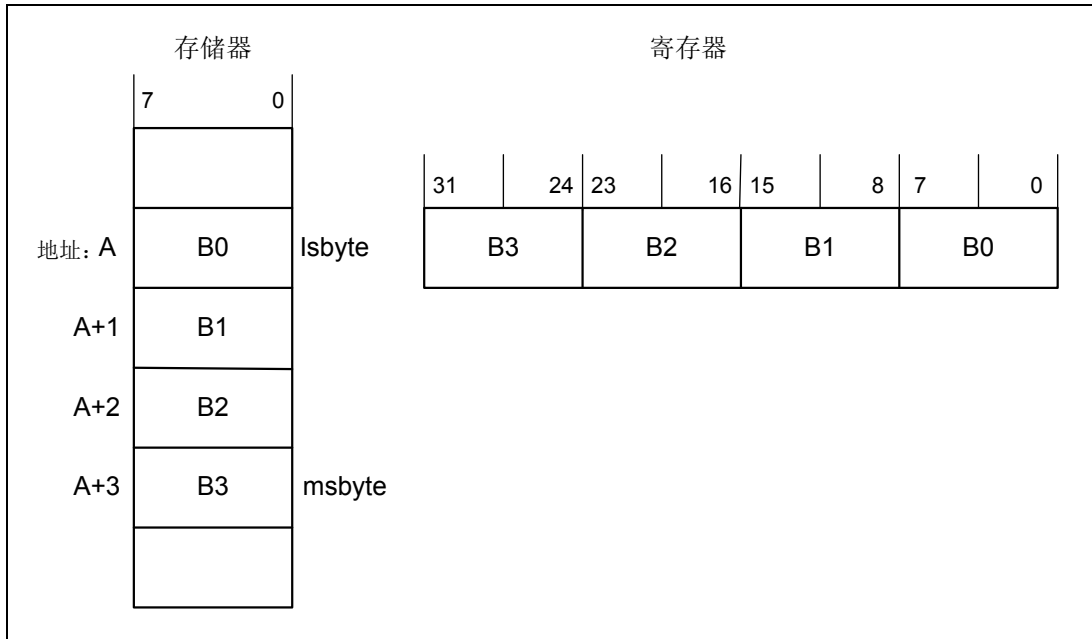
“存储器访问行为”小节描述了对位段区的直接字节、半字或字访问的行为。

### 附录1.3.2.6 存储器字节序

处理器将存储器视为一个线性的字节集合，从零开始递增序号。例如，字节 0-3 保存最先存储的字，字节 4-7 保存第二个存储的字。“小端（Little-endian）格式”小节描述了数据字如何存储在存储器中。

#### 小端（Little-endian）格式

在小端格式中，处理器将一个字的最低（有效）位字节存储在序号最小的字节，最高位（有效）位字节存储在序号最大的字节。例如：



附录图 39-39 小端格式

### 附录1.3.2.7 同步原语

Cortex-M3 指令集包括多对同步原语。这些原语提供了一种可以为线程或进程使用的非阻塞机制，从而获得某个存储单元的独占访问权限。软件可用它们来执行有保障的读取-修改-写入存储器更新序列，或实现一个信号量（semaphore）机制。一个同步原语对包括：

- ◇ 一条“独占加载”指令

用于读取某个存储单元的值，申请对该单元的独占访问。

- ◇ 一条“独占存储”指令

用于尝试写入相同的存储单元，返回一个状态位到寄存器。如果此位为：

- 0：表示线程或进程获得对存储器的独占访问权限，写入成功；
- 1：表示线程或进程未获得对存储器的独占访问权限，未进行写入。

“独占加载”和“独占存储”指令对有：

- ◇ 字指令 LDREX 和 STREX
- ◇ 半字指令 LDREXH 和 STREXH

◇ 字节指令 LDREXB 和 STREXB。

对于“独占存储”指令，软件必须使用相应的“独占加载”指令。

为了对某个存储单元进行有保障的读取-修改-写入，软件必须：

1. 使用一条“独占加载”指令，读取某个单元的值。
2. 按需要更新值。
3. 使用一条“独占存储”指令，尝试将新的值写回存储单元，并测试返回的状态位。  
如果 此位为：
  - 0：读取-修改-写入成功完成；
  - 1：未完成读取-修改-写入。这表示步骤 1 中返回的值可能已过期。软件必须重试读取-修改-写入序列。

软件可使用同步原语来实现一个信号量，如下：

1. 使用一条“独占加载”指令从信号量地址读取，检查信号量是否空闲。
2. 如果信号量空闲，则使用“独占存储”将声明值（claim value）写入信号量地址。
3. 如果步骤 2 返回的状态位指示“独占存储”成功，则软件已获得了信号量。但是，如果“独占存储”失败，则可能在软件执行步骤 1 后，有其他进程获得了信号量。

Cortex-M3 包括一个独占访问监控器，它会标注处理器已执行“独占加载”指令的情况。

如有以下情况，处理器会删除独占访问标签：

- ◇ 执行了一条 CLREX 指令。
- ◇ 无论写入是否成功，执行了一条“独占存储”指令。
- ◇ 发生一个异常。这表示处理器可解决不同线程间的信号量冲突。

有关同步原语指令的更多信息，请“LDREX 和 STREX”和“CLREX”小节。

### 附录1.3.2.8 同步原语的编程提示

ANSI C 不能直接产生独占访问指令。有些 C 编译器提供了可产生这些指令的内在函数：

指令	内在函数
LDREX, LDREXH 或 LDREXB	unsigned int ldrex(volatile void *ptr)
STREX, STREXH 或 STREXB	int strex(unsigned int val, volatile void *ptr)
CLREX	void clrex(void)

附录图 39-40 独占访问指令的 C 编译器内在函数

实际产生的独占访问指令取决于传递给内在函数的指针数据类型。例如，以下 C 代码产生所需的 LDREXB 操作：

```
__ldrex ((volatile char *) 0xFF) ;
```

### 附录1.3.3 异常模型

本节描述了异常模型。

#### 附录1.3.3.1 异常状态

每个异常都是以下几个状态之一：

- ◇ 无效  
异常无效且未被挂起。
- ◇ 挂起  
异常等待处理器的处理。来自外设或软件的中断请求可将相应中断的状态改为挂起。
- ◇ 有效  
异常正在由处理器处理，但尚未完成。

注：一个异常处理程序可以中断其他异常处理程序的执行。在此情况下，两个异常都处于有效状态。

- ◇ 有效和挂起  
处理器正在处理一个异常，且有一个相同来源的处于挂起状态的异常。

#### 附录1.3.3.2 异常类型

异常类型如下：

- ◇ 复位  
上电或热复位时，复位启动。异常模型将复位作为一种特殊的异常形式。当复位使能时，处理器可能在一条指令的任何位置停止操作。当复位禁能时，由向量表中的复位表项提供重新开始执行的地址。在线程模式下，重新执行是特权执行。
- ◇ NMI  
不可屏蔽中断（NMI）可由一个外设发出信号或由软件触发。这是除复位外的最高优先级异常。它被永久性使能，具有固定的优先级-2。NMI 不能：
  - 因任何其他异常的激活而被屏蔽或阻止
  - 被除复位外的其他任何异常抢占。
- ◇ HardFault  
HardFault 是一种异常，发生原因是异常处理期间出错，或异常无法被任何其他异常机制管理。HardFault 具有固定优先级-1，意味着它们的优先级高于任何具有可配置优先级的异常。
- ◇ 存储器管理故障  
存储器管理故障是一种由存储器保护相关故障而引发的异常。对于指令和数据存储事务，MPU 或固定的存储器保护约束条件决定此故障。此故障用于中止对从不执行(XN)存储区的指令访问，即使 MPU 被禁能也是如此。
- ◇ 总线故障  
总线故障是一种异常，由一个指令或数据存储器事务的存储器相关故障引发。这可能源

于从一个存储器系统总线上检测出的错误。

◇ 使用故障

使用故障是指令执行相关故障导致的一种异常。这包括：

- ◇ 未定义的指令
- ◇ 非法的非对齐访问
- ◇ 指令执行时的无效状态
- ◇ 异常返回时的错误。

当内核被配置为报告使用故障时，以下情况可以导致一个使用故障：

- ◇ 字和半字存储器访问时的非对齐地址；
- ◇ 除以零。

◇ SVCall

一次超级用户调用（SVC）是由 SVC 指令触发的异常。在 OS 环境中，应用程序可使用 SVC 指令来访问 OS 内核函数和设备驱动程序。

◇ PendSV

PendSV 是一个对系统级服务的中断驱动请求。在 OS 环境中，当无其他有效的异常时，使用 PendSV 进行上下文切换。

◇ SysTick

SysTick 异常是当系统定时器达到零时产生的异常。软件也可以产生一个 SysTick 异常。在 OS 环境中，处理器可将此异常用作系统节拍。

◇ 中断（IRQ）

中断（IRQ）是由一个外设发出信号，或由一个软件请求而产生的异常。所有中断都与指令执行异步。在系统中，外设使用中断与处理器进行通信。

异常序号 <sup>[1]</sup>	IRQ 序号 <sup>[1]</sup>	异常类型	优先级	向量地址或偏移量 <sup>[2]</sup>	激活
1	-	复位	-3, 最高	0x00000004	异步
2	-14	NMI	-2	0x00000008	异步
3	-13	HardFault	-1	0x0000000C	-
4	-12	存储器管理故障	可配置 <sup>[3]</sup>	0x00000010	同步
5	-11	总线故障	可配置 <sup>[3]</sup>	0x00000014	精确时是同步, 不精确时是异步
6	-10	使用故障	可配置 <sup>[3]</sup>	0x00000018	同步
7-10	-	-	-	保留	-
11	-5	SVCall	可配置 <sup>[3]</sup>	0x0000002C	同步
12-13	-	-	-	保留	-
14	-2	PendSV	可配置 <sup>[3]</sup>	0x00000038	异步
15	-1	SysTick	可配置 <sup>[3]</sup>	0x0000003C	异步
16 及以上	0 及以上	中断 (IRQ)	可配置 <sup>[4]</sup>	0x00000040 及以上 <sup>[5]</sup>	异步

附录图 39-41 不同异常类型的属性

注[1]: 为简化软件层级, CMSIS 仅使用 IRQ 序号, 因此针对异常 (而非中断) 使用负值。IPSR 返回异常序号, 参见表“IPSR 位分配”。

注[2]: 详细信息请参见“向量表”小节。

注[3]: 参见“系统处理程序优先级寄存器”小节。

注[4]: 参见“中断优先级寄存器”小节。

注[5]: 以 4 为单位递增。

对于异步的异常, 除复位外, 在异常被触发和处理器进入异常处理程序之间, 处理器也可以执行其他指令。

特权软件可以禁能上表“不同异常类型的属性”中列出的具有可配置优先级的异常, 见:

- ◇ “系统处理程序控制和状态寄存器”
- ◇ “中断清零使能寄存器”。

有关 HardFault、存储器管理故障、总线故障和使用故障的更多信息, 请参见“故障处理”小节。

### 附录1.3.3.3 异常处理程序

处理器使用以下程序来处理异常:

- ◇ 中断服务程序 (ISR)

IRQ0 和更高级别的中断是由 ISR 处理的异常。



◇ 故障处理程序

HardFault、存储器管理故障、使用故障、总线故障是由故障处理程序处理的异常。

◇ 系统处理程序

NMI、PendSV、SVCall SysTick 和故障异常都是系统异常，由系统处理程序处理。

附录1.3.3.4 向量表

向量表包含了堆栈指针复位值和全部异常处理程序的起始地址，又称异常向量。如下图“向量表”列出了向量表中异常向量的顺序。每个向量的最低有效位必须为 1，这表示异常处理程序为 Thumb 代码。请注意 IRQ 序号的上限因设备而异。

异常序号	IRQ序号	偏移量	向量
127	111	0x1FC	IRQ111
.	.	.	.
:	:	:	:
18		0x004C	IRQ2
17	2	0x0048	IRQ1
16	1	0x0044	IRQ0
15	0	0x0040	Systick
14	-1	0x003C	PendSV
13	-2	0x0038	保留 保留供调试使用
12			
11			SVCall
10	-5	0x002C	保留 使用
9			
8			
7			
6	-10	0x0018	
5	-11	0x0014	
4	-12	0x0010	
3	-13	0x000C	
2	-14	0x0008	
1		0x0004	
		0x0000	

附录图 39-42 向量表

系统复位时，向量表固定在地址 0x00000000。特权软件可通过写入 VTOR，将向量表起始地址重新定位到不同的存储单元，范围为 0x00000080 至 0x3FFFFFF80，见“向量表偏移量寄存器”小节。

### 附录1.3.3.5 异常优先级

如表“不同异常类型的属性”所示，所有异常都有一个相关的优先级，且：

- ◇ 用一个较低的优先级值表示一个较高的优先级；
- ◇ 除复位、HardFault 和 NMI 外的其他异常都有可配置的优先级。

如果软件未配置任何优先级，则所有具有可配置优先级的异常都具有 0 优先级。有关异常优

先级配置的信息，请参见：

- ◇ “系统处理程序优先级寄存器”
- ◇ “中断优先级寄存器”。

注：可配置的优先级值范围为 0 到 31。这就是说，具有固定的负数优先级值的复位、HardFault 和 NMI 异常总是具有比其他异常更高的优先级。

例如，如果给 IRQ[0] 赋以较高的优先级值并给 IRQ[1] 赋以较低的优先级值，那么这就意味着 IRQ[1] 的优先级高于 IRQ[0]。如果 IRQ[1] 和 IRQ[0] 都有效，则 IRQ[1] 先于 IRQ[0] 处理。

如果多个挂起的异常具有相同的优先级，则最先处理具有最低异常序号的异常。例如，如果 IRQ[0] 和 IRQ[1] 都挂起并具有相同的优先级，则 IRQ[0] 先于 IRQ[1] 被处理。

当处理器正在执行一个异常处理程序时，如果发生更高优先级的异常，则异常处理程序被抢占。如果发生了与正在处理的异常优先级相同的异常，无论异常序号如何，处理程序不会被抢占。但是，新中断的状态变为挂起。

### 附录1.3.3.6 中断优先级分组

为了对具有中断的系统加强优先级控制，NVIC 支持优先级分组。这将每个中断优先级寄存器项划分为两个域：

- ◇ 上区域定义了组优先级；
- ◇ 下区域定义了组内的子优先级。

只有组优先级才能决定中断异常的抢占。当处理器正在执行一个中断异常处理程序时，另一个与正在处理中的中断具有相同组优先级的中断不会抢占处理程序。

如果多个挂起的中断具有相同的组优先级，子优先级域决定了它们的处理顺序。如果多个挂起的中断具有相同的组优先级和子优先级，则具有最低 IRQ 序号的中断最先被处理。

有关中断优先级域拆分为组优先级和子优先级的相关信息，见“应用中断和复位控制寄存器”小节。

### 附录1.3.3.7 异常进入和返回

使用以下术语来描述异常处理：

#### ◇ 抢占

当处理器正在执行一个异常处理程序时，如果某个异常的优先级高于正在处理中异常的优先级，则这个异常可抢占异常处理程序。有关中断抢占的更多信息，请参见“中断优先级分组”小节。

当某个异常抢占另一个异常的处理程序时，这些异常被称为嵌套异常。更多信息请参见“异常进入”小节。

#### ◇ 返回

当异常处理程序完成且满足下列条件时，发生返回：

- 不存在挂起的异常，并且它们有被处理的足够优先级；
- 完成的异常处理程序并未在处理一个迟来（late-arriving）的异常。

处理器弹出堆栈，并将处理器状态恢复为中断发生前的状态。更多信息，请参见“异常返回”小节。

#### ◇ 末尾连锁（Tail-chaining）

此机制可提高异常服务的速度。当一个异常处理程序完成时，如果有一个挂起的异常符合异常进入要求，则跳过堆栈弹出将控制权转移给新的异常处理程序。

#### ◇ 迟来（late-arriving）

此机制可提高抢占速度。如果在前一异常状态保存期间发生了较高优先级的异常，处理器就会切换去处理较高优先级的异常，并启动对此异常的向量获取。状态保存不受迟来异常的影响，因为对这两个异常来说，保存的状态都是一样的。因此，状态保存可以不中断地继续进行。处理器可以接受迟来异常，直到初始异常的异常处理程序中的第一条指令进入处理器的执行阶段。在迟来异常的异常处理程序返回时，采用正常的末尾连锁规则。

### 异常进入

当存在一个有足够优先级的挂起异常，且满足以下条件之一时，发生异常进入：

- ◇ 处理器处于线程模式；
- ◇ 新的异常比正在处理中的异常具有更高优先级，在此情况下新的异常抢占原来的异常。

当一个异常抢占另一个异常时，异常被嵌套。

足够优先级指异常具有高于屏蔽寄存器所设置限值的优先级，见“异常屏蔽寄存器”小节。优先级低于它的异常被挂起，但不被处理器处理。

当处理器取得一个异常时，除非异常为末尾连锁或迟来异常，否则，处理器会将信息推入当前堆栈。此操作被称为入栈（stacking），而 8 个数据字的结构被称为堆栈帧。堆栈帧包含以下信息：

- ◇ R0-R3, R12；
- ◇ 返回地址；

- ◇ PSR;
- ◇ LR。

紧接着入栈之后，堆栈指针表示堆栈帧中的最低地址。除非禁能了堆栈对齐，否则，堆栈帧按双字地址对齐。如果配置控制寄存器（CCR）的 STKALIGN 位被设为 1，则在入栈期间要进行堆栈对齐调节。

堆栈帧包括返回地址。这是中断的程序中下一条指令的地址。在异常返回时，此值被恢复到 PC，以便让中断的程序继续运行。

入栈操作的同时，处理器执行向量取指，即从向量表读取异常处理程序的起始地址。当入栈完成时，处理器开始执行异常处理程序。同时，处理器将一个 EXC\_RETURN 值写入 LR。用以指示哪个堆栈指针对应于堆栈帧，以及在异常进入发生前，处理器处于何种操作模式。

如果异常进入期间没有发生更高优先级的异常，则处理器开始执行异常处理程序，并自动地将相应挂起中断的状态改变为有效。

如果异常进入期间发生了更高优先级的异常，则处理器开始执行此异常的异常处理程序，且

不改变前一异常的挂起状态。这就是迟来异常的情况。

#### 异常返回

当处理器处于处理模式，并执行下列指令之一将 EXC\_RETURN 值加载到 PC 时，发生异常返回：

- ◇ 一条包括 PC 的 POP 指令；
- ◇ 一条带任意寄存器的 BX 指令；
- ◇ 一条以 PC 为目标的 LDR 或 LDM 指令。

EXC\_RETURN 为异常进入时加载到 LR 中的值。异常机制靠此值来检测处理器何时完成一个异常处理程序。此值的最低 4 位提供关于返回堆栈和处理器模式的信息。下表列出了 EXC\_RETURN[3:0] 的值以及异常返回行为的描述。

处理器将 EXC\_RETURN 的位[31:4]设为 0xFFFFFFFF。当此值被加载到 PC 中时，它向处理器指示异常已完成，处理器就会启动异常返回序列操作。

EXC_RETURN[3:0]	描述
bXXX0	保留。
b0001	返回处理模式。异常返回，获得来自 MSP 的状态。返回后指令执行使用 MSP。
b0011	保留。
b01X1	保留。
b1001	返回线程模式。异常返回，获得来自 MSP 的状态。返回后指令执行使用 MSP。
b1101	返回线程模式。异常返回，获得来自 PSP 的状态。返回后指令执行使用 PSP。
b1X11	保留。

附录图 39-43 异常返回行为

### 附录1.3.4 故障处理

故障为异常的一个子集，见“异常模型”小节。以下情况下会产生一个故障：

- ◇ 以下情况中的一个总线错误：
  - 指令取指或向量表加载
  - 数据访问
- ◇ 一个内部检测到的错误，例如未定义的指令，或试图用 BX 指令改变状态；
- ◇ 试图从标记为不可执行（XN）的存储区执行一个指令；
- ◇ 因特权违犯或试图访问一个未管理（unmanaged）区所引起的 MPU 故障。

#### 附录1.3.4.1 故障类型

下表列出了故障类型、故障的处理程序、相应的故障状态寄存器和指示发生故障的寄存器位。关于故障状态寄存器的更多信息，请参见“可配置故障状态寄存器”小节。

故障		处理程序	位名称	故障状态寄存器
读向量总线错误		HardFault	VECTTBL	HardFault 状态寄存器
升级为 HardFault 的故障			FORCED	
MPU 不匹配	由于指令访问	存储器管理故障	IACCVIOL <sup>[1]</sup>	存储器管理故障地址寄存器
	由于数据访问		DACCVIOL	
	异常入栈期间		MSTKERR	
	异常出栈期间		MUNSKERR	
总线错误	异常入栈期间	总线故障	STKERR	总线故障地址寄存器
	异常出栈期间		UNSTKERR	
	指令预取期间		IBUSERR	
	精确数据总线错误		PRECISERR	
	非精确数据总线错误		IMPRECISERR	
试图访问一个协处理器		使用故障	NOCP	使用故障状态寄存器
未定义指令			UNDEFINSTR	
试图进入一个无效的指令集状态 <sup>[2]</sup>			INVSTATE	
无效的 EXC_RETURN 值			INVPC	
非法的未对齐加载或存储			UNALIGNED	
除以 0			DIVBYZERO	

附录图 39-44 故障

注[1]: 即使 MPU 被禁能, 该操作也会在每次访问 XN 区时发生。

注[2]: 试图使用一个非 Thumb 指令集。

### 附录1.3.4.2 故障升级和 HardFault

除了 HardFault, 所有故障异常都具有可配置的异常优先级, 见“系统处理程序优先级寄存器”小节。软件可禁止执行这些故障的处理程序, 见“系统处理程序控制和状态寄存器”小节。

通常, 异常优先级与异常屏蔽寄存器的值共同决定了处理器是否进入故障处理程序, 以及一个故障处理程序是否可抢占另一个故障处理程序。如“异常模型”小节所述。

在某些情况下, 有可配置优先级的故障可作为一个 HardFault。这被称为优先级升级, 该故障被描述为升级到 HardFault。出现以下情况时发生升级到 HardFault:

- ◇ 故障处理程序会产生与它正在处理的故障相同类型的故障。升级到 HardFault 的原因是故障处理程序不能抢占自身, 因为它必须与当前优先级具有相同的优先级。
- ◇ 故障处理程序导致一个与它正在处理的故障具有相同或更低优先级的故障。这是因为新故障的处理程序不能抢占当前正在执行的故障处理程序。
- ◇ 异常处理程序导致一个与正在执行的异常具有相同或更低优先级的故障。
- ◇ 发生了一个故障, 但此故障的处理程序未被使能。

如果在进入总线故障处理程序时, 堆栈推入期间发生了一个总线故障, 则总线故障不会升级到 HardFault。这就是说, 如果被破坏的堆栈导致一个故障发生, 则即使处理程序的堆栈推入失败, 故障处理程序也会执行。此故障处理程序可以工作, 但堆栈内容被破坏。

注: 只有复位和 NMI 可抢占固定优先级的 HardFault。HardFault 可抢占除复位、NMI 或另一 HardFault 外的任何异常。

### 附录1.3.4.3 故障状态寄存器和故障地址寄存器

故障状态寄存器指示故障的起因。对于总线故障和存储器管理故障来说, 故障地址寄存器指示导致故障发生的操作所访问的地址, 如下表所示。

处理程序	状态寄存器名称	地址寄存器名称	寄存器描述
HardFault	HFSR	-	“HardFault 状态寄存器”小节
存储器管理故障	MMFSR	MMFAR	“存储器管理故障地址 寄存器”小节 “存储器管理故障状态 寄存器”小节
总线故障	BFSR	BFAR	“总线故障状态寄存器 ”小节 “总线故障地址寄存器 ”小节
使用故障	UFSR	-	“使用故障状态寄存器 ”小节

附录图 39-45 故障状态和故障地址寄存器

#### 附录1.3.4.4 锁定

执行 NMI 或 HardFault 处理程序时,如果发生 HardFault,则处理器进入一个锁定状态。当处理器处于锁定状态时,它不执行任何指令。处理器保持在锁定状态,直到发生以下情况之一:

- ◇ 处理器被复位;
- ◇ 发生一个 NMI 故障。

注:如果 NMI 处理程序使处理器进入锁定状态,则后续 NMI 也不会改变处理器的锁定状态。

#### 附录1.3.5 电源管理

注:基于 Cortex-M3 处理器的 ES32 微控制器都支持额外的降功率模式。有关全部可用的低功率模式的信息,请参见“功率控制”小节。

Cortex-M3 处理器的睡眠模式可降低功耗:

- ◇ 睡眠模式会停止处理器时钟;
- ◇ 深度睡眠模式会停止系统时钟,并关闭 PLL 和 Flash 存储器的电源。

SCR 的 SLEEPDEEP 位用来选择使用哪种睡眠模式,见“系统控制寄存器”小节。更多有关睡眠模式行为的信息,请参见“功率控制”小节。

本节叙述了进入睡眠模式的机制和从睡眠模式唤醒的条件。

##### 附录1.3.5.1 进入睡眠模式

本节叙述了软件用于将处理器置于睡眠模式的机制。

系统可能发生伪唤醒事件,例如,一个调试操作唤醒处理器。因此,软件必须能够在该类事件发生后,将处理器返回睡眠模式。程序可以有一个空闲循环(idle loop)来将处理器置回睡眠模式。

##### 等待中断

等待中断指令 WFI 会使处理器立即进入睡眠模式。当处理器执行 WFI 指令时,处理器会停止执行指令并进入睡眠模式。更多信息,请参见“WFI”小节。

##### 等待事件

注:基于 Cortex-M3 处理器的 ES32 微控制器不执行外部事件。

等待事件指令 WFE 使处理器根据一个单一事件寄存器的值,有条件地进入睡眠模式。当处理器执行 WFE 指令时,它会检查该寄存器的下列内容:

- ◇ 如果该寄存器为 0,处理器停止执行指令并进入睡眠模式;
- ◇ 如果该寄存器为 1,处理器将该寄存器清零,不进入睡眠模式并继续执行指令。

更多信息,请参见“WFE”小节。

如果事件寄存器为 1,就表示当执行 WFE 指令时,处理器不得进入睡眠模式。通常,这是因为使能了一个外部事件信号,或系统中的一个处理器执行了一条 SEV 指令,请

参见“SEV”小节。软件不能直接访问此寄存器。

#### **退出时睡眠 (Sleep-on-exit)**

如果 SCR 的 SLEEPONEXIT 位设为 1，则当处理器执行完一个异常处理程序时会返回到线程模式，并立即进入睡眠模式。在发生异常时，只在那些需要处理器运行的应用中使用此机制。



### 附录1.3.5.2 从睡眠模式唤醒

处理器唤醒的条件取决于导致处理器进入睡眠模式的机制。

#### 从“WFI”或“退出时睡眠”唤醒

正常情况下，只有在处理器检测到一个具有足够优先级且可造成异常进入的异常时，处理器才会唤醒。

某些嵌入式系统可能必须在处理器唤醒后和执行一个中断处理程序前，先执行系统恢复任务。为了实现该操作，将 PRIMASK 位设为 1，并将 FAULTMASK 位设为 0。如果有一个中断到达，该中断被使能且具有比当前异常优先级更高的优先级，则处理器唤醒，但不执行中断处理程序，直到处理器将 PRIMASK 设为零。有关 PRIMASK 和 FAULTMASK 的更多信息，请参见“异常屏蔽寄存器”小节。

#### 从“WFE”唤醒

如有以下情况，处理器被唤醒：

- ◇ 处理器探测到一个有足够优先级的异常，可产生异常进入。

另外，如果 SCR 中 SEVONPEND 位被设为 1，则任何新挂起的中断都会触发一个事件并唤醒处理器，哪怕该中断被禁止或没有足够高的优先级来产生异常进入。更多关于 SCR 的信息，请参见“系统控制寄存器”小节。

### 附录1.3.5.3 唤醒中断控制器

唤醒中断控制器 (WIC) 是一个外设，它可以检测一个中断并将处理器从深度睡眠模式、掉电或深度掉电模式唤醒。只有在 SCR 中 DEEPSLEEP 位被设为 1 时，WIC 才被使能。见“系统控制寄存器”小节。

注：ES32 微控制器扩展了降功率模式的数量，使之超出了 Cortex-M3 直接支持的模式数量。关于全部可用降功率模式和对 ES32 微控制器唤醒可能性的详细说明，请参见“功率控制”小节。

WIC 不可编程，且没有任何寄存器或用户接口。它完全依靠来自硬件的信号工作。

当 WIC 被使能，且处理器进入深度睡眠模式或掉电模式时，系统中电源管理单元可能会关闭 Cortex-M3 处理器的大部分电源。产生的副作用是会停止 SysTick 定时器。当 WIC 收到一个中断时，它要消耗一些时钟周期来唤醒处理器并恢复其状态，然后才能处理中断。这就意味着，在深度睡眠模式下，中断等待时间变长。从掉电模式唤醒需要启动设备的很多其他部分，需要消耗更长的时间。从深度掉电模式唤醒还要加上重新建立片上稳压器电压的时间。

注：如果处理器检测到一个与调试器的连接，它会禁能 WIC。

### 附录1.3.5.4 电源管理编程提示

ANSI C 不能直接产生 WFI 和 WFE 指令。CMSIS 为这些指令提供了以下内在函数：

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
```

## 附录1.4 ARM Cortex-M3 用户指南：外设

### 附录1.4.1 关于 Cortex-M3 外设

私有外设总线（PPB）的地址映射如下：

地址	内核外设	描述
0xE000E008~0xE000E00F	系统控制模块	见表“系统控制模块寄存器汇总”
0xE000E010~0xE000E01F	系统定时器	见表“系统定时器寄存器汇总”
0xE000E100~0xE000E4EF	可嵌套向量中断控制器	见表“NVIC 寄存器汇总”
0xE000ED00~0xE000ED3F	系统控制模块	见表“系统控制模块寄存器汇总”
0xE000ED90~0xE000EDB8	存储器保护单元	见表“MPU 寄存器汇总”
0xE000EF00~0xE000EF03	可嵌套向量中断控制器	见表“NVIC 寄存器汇总”

附录图 39-46 内核外设寄存器区

在寄存器描述中：

- ◇ 寄存器类型描述如下：
  - ◇ RW：读写。
  - ◇ R：只读。
  - ◇ W：只写。
- ◇ 所需特权给出访问寄存器所需的下列特权等级：
  - ◇ 特权：仅特权软件可访问寄存器。
  - ◇ 非特权：非特权和特权软件均可访问寄存器。

## 附录1.4.2 可嵌套向量中断控制器

本节叙述了可嵌套向量中断控制器（NVIC）和它所使用的寄存器。NVIC 支持：

- ◇ 多达 112 个中断。实现的中断数量因设备而异。
- ◇ 每个中断有可编程的 0 到 31 优先等级。较高等级对应于较低的优先级，因此 0 级为最高的中断优先级。
- ◇ 中断信号的电平和脉冲检波。
- ◇ 中断优先级的动态重设。
- ◇ 优先级值分组为组优先级和子优先级域。
- ◇ 中断末尾连锁。
- ◇ 一个外部不可屏蔽中断（NMI）。

处理器在异常进入时将其状态自动入栈，并在异常退出时将其状态自动出栈，没有指令开销。这样就提供了低延迟的异常处理。NVIC 寄存器的硬件实现为：

地址	名称	类型	所需特权	复位值	描述
0xE000E100~0xE000E10C	ISER0 - ISER3	RW	特权	0x00000000	见表“ISER 位分配”
0xE000E180~0xE000E18C	ICER0 - ICER3	RW	特权	0x00000000	见表“ICER 位分配”
0xE000E200~0xE000E20C	ISPR0 - ISPR3	RW	特权	0x00000000	见表“ISPR 位分配”
0xE000E280~0xE000E28C	ICPR0 - ICPR3	RW	特权	0x00000000	见表“ICPR 位分配”
0xE000E300~0xE000E30C	IABR0 - IABR3	R	特权	0x00000000	见表“IABR 位分配”
0xE000E400~0xE000E46C	IPR0 - IPR27	RW	特权	0x00000000	见表“IPR 位分配”
0xE000EF00	STIR	W	可配置 <sup>[1]</sup>	0x00000000	见表“STIR 位分配”

附录图 39-47 NVIC 寄存器汇总

注[1]: 每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

### 附录1.4.2.1 Cortex-M3 NVIC 寄存器的 CMSIS 映射

为了提高软件效率，CMSIS 简化了 NVIC 寄存器的表达。在 CMSIS 中：

- ◇ 置位使能、清零使能、置位挂起、清零挂起和有效位寄存器映射到 32 位整数的数组，从而使：
  - ◇ 数组 ISER[0]到 ISER[3]对应于寄存器 ISER0 - ISER3；
  - ◇ 数组 ICER[0]到 ICER[3]对应于寄存器 ICER0 - ICER3；
  - ◇ 数组 ISPR[0]到 ISPR[3]对应于寄存器 ISPR0 - ISPR3；
  - ◇ 数组 ICPR[0]到 ICPR[3]对应于寄存器 ICPR0 - ICPR3；
  - ◇ 数组 IABR[0]到 IABR[3]对应于寄存器 IABR0 - IABR3。
- ◇ 8 位的“中断优先级寄存器”域映射到一个 8 位整数数组，从而使数组 IP[0]到 IP[112]对应于寄存器 IPR0 - IPR59，数组表项 IP[n]保存中断 n 的中断优先级。

CMSIS 提供线程安全代码，它实现了对“中断优先级寄存器”的原子访问。更多信息，请参见“NVIC 编程提示”小节中 NVIC\_SetPriority 函数的描述。下表列出了中断、或 IRQ 序号是如何映射到中断寄存器和对相应的 CMSIS 变量，这些变量对每个中断有一个位。

中断	CMSIS 数组元 <sup>[1]</sup>				
	置位-使能	清零-使能	置位-挂起	清零-挂起	有效位
0-31	ISER[0]	ICER[0]	ISPR[0]	ICPR[0]	IABR[0]
32-63	ISER[1]	ICER[1]	ISPR[1]	ICPR[1]	IABR[1]
64-95	ISER[2]	ICER[2]	ISPR[2]	ICPR[2]	IABR[2]
96-127	ISER[3]	ICER[3]	ISPR[3]	ICPR[3]	IABR[3]

附录图 39-48 中断到中断变量的映射

注[1]: 每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

### 附录1.4.2.2 中断置位使能寄存器

ISER0-ISER3 寄存器使能中断，并显示哪些中断被使能。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
  - ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。
- 位分配见下表。

位	名称	功能
[31:0]	SETENA	中断置位-使能位。
		写：
		0: 无影响
		1: 使能中断。

		读:
		0: 中断被禁能
		1: 中断被使能。

附录图 39-49 ISER 位分配

如果某个挂起中断被使能，NVIC 根据其优先级激活中断。如果一个中断没有被使能，则这个中断信号的发出会将中断状态变为挂起，但无论其优先级如何，NVIC 永远不会激活此中断。

#### 附录1.4.2.3 中断清零使能寄存器

ICER0-ICER3 寄存器禁止中断，并显示哪些中断被使能。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
- ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	CLRENA	中断清零-使能位。
		写:
		0: 无影响
		1: 禁能中断。
		读:
		0: 中断被禁能
		1: 中断被使能。

附录图 39-50 ICER 位分配

#### 附录1.4.2.4 中断置位挂起寄存器

ISPR0-ISPR3 寄存器强制中断进入挂起状态，并显示哪些中断正在挂起。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
- ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	SETPEND	中断置位-挂起位。
		写:
		0: 无影响
		1: 改变中断状态为挂起状态
		读:

		0: 中断未挂起
		1: 中断挂起

附录图 39-51 ISPR 位分配

注：将 1 写到对应于以下项的 ISPR 位：

- ◇ 一个挂起的中断，无效果；
- ◇ 一个被禁用的中断，设置该中断的状态为挂起。

#### 附录1.4.2.5 中断清零挂起寄存器

ICPR0-ICPR3 寄存器移除中断的挂起状态，并显示有哪些中断在挂起。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
- ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

位	名称	功能
[31:0]	CLRPEND	中断清零-挂起位。
		写：
		0: 无影响
		1: 清除挂起状态的一个中断
		读：
		0: 中断未挂起
		1: 中断挂起

附录图 39-52 ICPR 位分配

注：将 1 写到一个 ICPR 位不影响相应中断的行为状态。

#### 附录1.4.2.6 中断有效位寄存器

IABR0-IABR3 寄存器指示哪些中断是有效的。请参见：

- ◇ “NVIC 寄存器汇总”表中寄存器汇总的寄存器属性；
- ◇ “中断到中断变量的映射”表中了解每个寄存器都控制哪些中断。

位分配见下表。

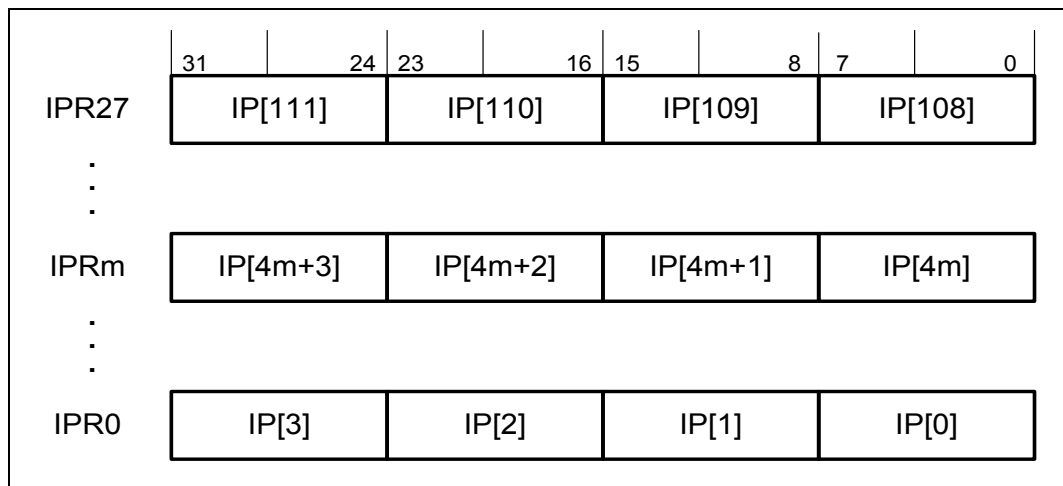
位	名称	功能
[31:0]	ACTIVE	中断有效标志：
		0: 中断无效
		1: 中断有效。

附录图 39-53 IABR 位分配

如果相应中断的状态为有效或有效并挂起，则该位读取值为 1。

### 附录1.4.2.7 中断优先级寄存器

IPR0-IPR27 寄存器为每个中断提供一个 5 位的优先级域。这些寄存器可按字节访问。其属性请参见“NVIC 寄存器汇总”表中寄存器汇总。每个寄存器保存 4 个优先级域，它映射到 CMSIS 中断优先级数组 IP[0]到 IP[111]中的 4 个元素，如下所示：



附录图 39-54 中断优先级寄存器

位	名称	功能
[31:24]	优先级，字节偏移量 3	每个优先级域承载一个优先级值，0 至 31。这个值越低，相应中断的优先级就越高。处理器仅执行每个域的位[7:3]，位 [2:0]读为零且忽略写入值。
[23:16]	优先级，字节偏移量 2	
[15:8]	优先级，字节偏移量 1	
[7:0]	优先级，字节偏移量 0	

附录图 39-55 IPR 位分配

更多关于 IP[0]到 IP[111]中断优先级数组的信息，请参见“中断到中断变量的映射”表，该数组提供了中断优先级的软件视图。

按以下方式查找中断 N 的 IPR 号和字节偏移量：

- ◇ 相应的 IPR 号 M，用  $M = N$  除以 4 得到；
- ◇ 此寄存器中所需“优先级”域的字节偏移量为  $N \text{ MOD } 4$ ，其中：
  - 字节偏移量 0 引用寄存器位[7:0]；
  - 字节偏移量 1 引用寄存器位[15:8]；
  - 字节偏移量 2 引用寄存器位[23:16]；
  - 字节偏移量 3 引用寄存器位[31:24]。



### 附录1.4.2.8 软件触发中断寄存器

写入 STIR 产生一个软件产生的中断 (SGI)。请参见“NVIC 寄存器汇总”表中寄存器汇总了解 STIR 属性。

当 CCR 中 USERSETMPEND 位设置为 1 时, 非特权软件可访问 STIR, 见“配置和控制寄存器”小节。

注: 只有特权软件才能对 STIR 的非特权访问。

位分配见下表。

位	域	功能
[31:9]	-	保留。
[8:0]	INTID	所需 SGI 的中断 ID 范围 (0~111)。例如, b000000011 指定的是中断 IRQ3。

附录图 39-56 STIR 位分配

### 附录1.4.2.9 电平触发和脉冲中断

处理器同时支持电平触发中断和脉冲中断。脉冲中断也称为边沿触发中断。

电平触发中断会保持有效状态, 直到外设撤销中断信号。通常这是因为 ISR 访问外设, 导致它清除中断请求。脉冲中断是一个在处理器时钟上升沿同步采样的中断信号。为了确保 NVIC 检测到中断, 外设必须使中断信号保持有效至少一个时钟周期, 在此期间 NVIC 可检测到脉冲并锁存中断。

当处理器进入 ISR 时, 它自动从中断移除挂起状态, 见“中断的软/硬件控制”小节。对于电平触发中断, 如果在处理器从 ISR 返回前信号没有撤销, 中断重新变为挂起状态, 则处理器必须再次执行它的 ISR。这就是说, 外设可以保持中断一直有效, 直到不再需要处理为止。

#### 中断的软/硬件控制

Cortex-M3 锁存全部中断。一个外设中断会因以下原因之一而变为挂起状态:

- ◇ NVIC 检测到中断信号为高电平且中断为无效状态;
- ◇ NVIC 检测到一个中断信号的上升沿;
- ◇ 软件写入相应的中断置位挂起寄存器位, 见表“ISPR 位分配”, 或写入 STIR, 造成一个 SGI 挂起, 见表“STIR 位分配”。

一个挂起中断会保持在挂起状态, 直到发生以下情况之一:

- ◇ 处理器进入该中断的 ISR。使中断的状态从挂起变为有效。然后:
  - 对于电平触发中断, 当处理器从 ISR 返回时, NVIC 采样中断信号。如果信号被使能, 则中断变为挂起状态, 这可能使处理器立即重新进入 ISR。否则, 中断状态变为无效。
  - 对于脉冲中断, NVIC 持续监测中断信号, 如果有信号脉冲, 则中断状态变为挂起和有效。在此情况下, 当处理器从 ISR 返回时, 中断状态变为挂起, 这可能使处理器立即重新进入 ISR。

- 如果处理器在 ISR 中时没有中断信号脉冲，则当处理器从 ISR 返回时，中断状态变为无效。
- ◇ 软件写入相应的中断清零挂起寄存器位。对于电平触发中断，如果中断信号仍然有效，则中断状态不变。否则中断状态变为无效。

对于脉冲中断，中断状态变为：

- 无效，如果原状态为挂起；
- 有效，如果原状态为有效和挂起。

#### 附录1.4.2.10 NVIC 设计提示和建议

确保软件使用了正确对齐的寄存器访问。处理器不支持对 NVIC 寄存器的非对齐访问。请参见各个寄存器的描述了解它所支持的访问大小。

即使某个中断被禁用，它也可以进入挂起状态。

在对 VTOR 编程以重新定位向量表以前，要确保对故障处理程序、NMI 和所有使能的异常（如中断）都建立了新向量表的向量表项。更多信息，请参见表“VTOR 位分配”。

#### NVIC 编程提示

软件使用 CPSIE I 和 CPSID I 指令来使能和禁能中断。CMSIS 提供为这些指令提供了以下内在函数：

```
void __disable_irq(void) // Disable Interrupts
void __enable_irq(void) // Enable Interrupts
```

另外，CMSIS 提供了一些 NVIC 控制函数，包括：

CMSIS 中断控制函数	描述
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)	设置优先级分组
void NVIC_EnableIRQ(IRQn_t IRQn)	使能 IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	禁能 IRQn
uint32_t NVIC_GetPendingIRQ(IRQn_t IRQn)	如果 IRQn 挂起，返回结果为真 (IRQ-Number 即 IRQ 序号)
void NVIC_SetPendingIRQ(IRQn_t IRQn)	设置 IRQn 挂起
void NVIC_ClearPendingIRQ(IRQn_t IRQn)	清除 IRQn 挂起状态
uint32_t NVIC_GetActive(IRQn_t IRQn)	返回有效中断的 IRQ 序号
void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority)	设置 IRQn 优先级
uint32_t NVIC_GetPriority(IRQn_t IRQn)	读取 IRQn 优先级
void NVIC_SystemReset(void)	复位系统

附录图 39-57 用于 NVIC 控制的 CMSIS 函数

有关这些函数的更多信息，请参见 CMSIS 文档。

### 附录1.4.3 系统控制模块

系统控制模块（SCB）提供系统执行信息和系统控制。这包括对系统异常的配置、控制和报告。系统控制模块寄存器有：

地址	名称	类型	所需特权	复位值
0xE00E008	ACTLR	RW	特权	0x00000000
0xE00ED00	CPUID	R	特权	0x412FC230
0xE00ED04	ICSR	RW <sup>[1]</sup>	特权	0x00000000
0xE00ED08	VTOR	RW	特权	0x00000000
0xE00ED0C	AIRCR	RW <sup>[1]</sup>	特权	0xFA050000
0xE00ED10	SCR	RW	特权	0x00000000
0xE00ED14	CCR	RW	特权	0x00000200
0xE00ED18	SHPR1	RW	特权	0x00000000
0xE00ED1C	SHPR2	RW	特权	0x00000000
0xE00ED20	SHPR3	RW	特权	0x00000000
0xE00ED24	SHCRS	RW	特权	0x00000000
0xE00ED28	CFSR	RW	特权	0x00000000
0xE00ED28	MMSR <sup>[2]</sup>	RW	特权	0x00
0xE00ED29	BFSR <sup>[2]</sup>	RW	特权	0x00
0xE00ED2A	UFSR <sup>[2]</sup>	RW	特权	0x0000
0xE00ED2C	HFSR	RW	特权	0x00000000
0xE00ED34	MMFAR	RW	特权	未定义
0xE00ED38	BFAR	RW	特权	未定义

附录图 39-58 系统控制模块寄存器汇总

注[1]: 更多信息, 参见寄存器描述。

注[2]: CFSR 的一个子寄存器。

#### 附录1.4.3.1 Cortex-M3 SCB 寄存器的 CMSIS 映射

为了提高软件效率, CMSIS 简化了 SCB 寄存器的表述。在 CMSIS 中, 字节数组 SHP[0] 到 SHP[12]对应于寄存器 SHPR1 至 SHPR3。

#### 附录1.4.3.2 辅助控制寄存器 (ACTLR)

ACTLR 提供了以下处理器功能的禁用位:

- ◇ IT 折叠 (folding);
- ◇ 用于访问默认存储器映射的写缓冲;

◇ 多周期指令的中断。

请参见表“系统控制模块寄存器汇总”中寄存器汇总了解 ACTLR 属性。位分配见表“ACTLR 位分配”。

位	名称	功能
[31:3]	-	保留
[2]	DISFOLD	该位为 1 时，禁能 IT 折叠。更多信息请参见“关于 IT 折叠”小节。
[1]	DISDEFWBUF	该位为 1 时，在默认存储器映射访问期间禁能写缓冲使用。这使得所有的总线故障都是精确总线故障，但却降低了性能，因为存储器的所有存储操作都必须在处理器能够执行下一指令前完成。 注：该位仅影响在 Cortex-M3 处理器中执行的写缓冲。
[0]	DISMCYCINT	该位为 1 时，禁能多个加载和多个存储指令的中断。这增加了处理器的中断延迟，因为所有的 LDM 或 STM 都必须在处理器能够堆栈当前状态并进入中断处理程序前完成。

附录图 39-59 ACTLR 位分配

#### 关于 IT 折叠

在某些情况下，处理器可以在执行 IT 指令时，开始执行某个 IT 块中的第一条指令。此行为称为 IT 折叠，可提高性能，然而，IT 折叠可能导致循环中的抖动（jitter）。如果某个任务必须避免抖动，则要在执行任务前将 DISFOLD 位设为 1，禁用 IT 折叠。

#### 附录1.4.3.3 CPUID 基址寄存器

CPUID 寄存器包括处理器部件号、版本和执行信息。其属性请参见表“系统控制模块寄存器汇总”中的寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	Implementer	实现者代码：0x41 = ARM
[23:20]	Variant	变量号，rnpn 产品修订标识符中的 r 值：0x2 = r2p0 读为 0xF
[19:16]	Constant	
[15:4]	PartNo	处理器的部件序号：0xC23 = Cortex-M3
[3:0]	Revision	修订号，rnpn 产品修订标识符中的 p 值：0x0 = r2p0

附录图 39-60 CPUID 寄存器的位分配

#### 附录1.4.3.4 中断控制和状态寄存器

##### ICSR:

◇ 提供:

- 用于不可屏蔽中断（NMI）异常的置位挂起位；
- 用于 PendSV 和 SysTick 异常的置位挂起位和清零挂起位。

◇ 指示:

- 正在被处理的异常的异常号；

- 有无被抢占的有效异常；
- 具有最高优先级的挂起异常的异常号；
- 是否有挂起中断。

ICSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总和表“ICSR 位分配”中的类型描述。位分配见下表。

位	名称	类型	功能
[31]	NMIPENDSET	RW	NMI 置位-挂起位。
			写:
			0: 无影响
			1: 改变 NMI 异常状态为挂起。
			读:
			0: NMI 异常未挂起
			1: NMI 异常挂起。
			因为 NMI 是最高优先级的异常, 通常情况下处理器在该位被写入 1 后马上进入 NMI 异常处理程序, 进入处理程序将该位清零。当处理器执行某个 NMI 异常处理程序时, 只有当 NMI 信号再次有效时, 该处理程序对该位的读取结果才返回为 1。
[30:29]	-	-	保留。
[28]	PENDSVSET	RW	PendSV 置位-挂起位。
			写:
			0: 无影响
			1: 改变 PendSV 异常状态为挂起。
			读:
			0: PendSV 异常未挂起
			1: PendSV 异常挂起。
			向该位写入 1 是将 PendSV 异常状态设为挂起的唯一途径。
[27]	PENDSVCLR	W	PendSV 清零-挂起位。
			写:
			0: 无影响
			1: 移除 PendSV 异常中的挂起状态
[26]	PENDSTSET	RW	SysTick 异常置位-挂起位。
			写:
			0: 无影响
			1: 改变 SysTick 异常状态为挂起。

位	名称	类型	功能
			读： 0: SysTick 异常未挂起 1: SysTick 异常挂起。
[25]	PENDSTCLR	W	SysTick 异常清零-挂起位。 写： 0: 无影响 1: 移除 SysTick 异常中的挂起状态。该位为只写位。该位在寄存器上读取值未知。
[24]	-	-	保留。
[23]	Reserved for Debug use	R	该位保留用于调试，当处理器不在调试模式时该位读取为零。
[22]	ISRPENDING	R	中断挂起标志，不包括 NMI 和故障： 0: 中断未挂起 1: 中断挂起。
[21:18]	-	-	保留。
[17:12]	VECTPENDING	R	该位表示最高优先级挂起使能异常的异常序号： 0: 无挂起异常 非零值：最高优先级挂起使能异常的异常序号。 该域值包括 BASEPRI 和 FAULTMASK 寄存器的效果，但不包括 PRIMASK 寄存器的任何效果。
[11]	RETTOBASE	R	该位表示有无被抢占的有效异常： 0: 有待执行的被抢占的有效异常 1: 无有效异常，或当前执行的异常是唯一有效异常。
[10:9]	-	-	保留。
[8:0]	VECTACTIVE <sup>[1]</sup>	R	包含有效异常序号： 0: 线程模式 非零值：当前有效异常的异常序号 <sup>[1]</sup> 。 注：该值减 16 可以得到用来索引到中断清零使能、置位使能、清零挂起、置位挂起或优先级寄存器所需的 IRQ 序号，见表“IPSR 位分配”。

附录图 39-61 ICSR 位分配

注[1]: 这与 IPSR 位[8:0]的值相同，见表“IPSR 位分配”。

当写入 ICSR 时，如有以下情况，则后果不可预知：

- ◇ 写入 1 到 PENDSVSET 位并写入 1 到 PENDSVCLR 位；
- ◇ 写入 1 到 PENDSTSET 位并写入 1 到 PENDSTCLR 位。

#### 附录1.4.3.5 向量表偏移量寄存器

VTOR 表示从存储器地址 0x00000000 开始的向量表基址偏移量。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

位分配见下表。

位	名称	功能
[31:30]	-	保留。
[29:8]	TBLOFF	向量表的基址偏移域。它包含向量表基址与存储器映射底部的偏移量的位 [29:8]。
		注：位[29]决定向量表是否在编码或 SRAM 存储区：
		位[29]有时被称为位 TBLBASE。
		0：编码 1：SRAM。
[7:0]	-	保留。

附录图 39-62 VTOR 位分配

当置位 TBLOFF 时，必须使偏移对准向量表中异常进入的序号。建议的对齐方式为 256 个字，可用于 128 个中断。

注：表对齐要求的含义是表偏移的位[7:0]始终为零。

#### 附录1.4.3.6 应用中断和复位控制寄存器

AIRCR 提供了异常模型的优先级分组控制、数据访问的字节序状态，以及系统的复位控制。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总和表“AIRCR 位分配”。

如要对此寄存器执行写操作，必须将 0x5VA 写入 VECTKEY 域，否则处理器会忽略写入值。位分配见下表。

	名称	类型	功能
[31:16]	写：VECTKEYSTAT	RW	寄存器密钥（register key）：读为 0x05FA 执行写操作时要求向
	读：VECTKEY		VECTKEY 写入 0x5FA，否则写入值被忽略。
[15]	ENDIANESS	R	数据字节序位： 0：小端。
[14:11]	-	-	保留

	名称	类型	功能
[10:8]	PRIGROUP	R/W	中断优先级分组域。该域决定从子优先级中拆分组优先级，见“二进制点”小节。
[7:3]	-	-	保留。
[2]	SYSRESETREQ	W	系统复位请求： 0: 无系统复位请求 1: 让信号在外部系统有效，表示请求一个复位。该位用于强制对调试设备之外的所有主要设备进行一次大的系统复位。 注：ES32 微控制器支持 SYSRESETREQ。该位读为 0。
[1]	VECTCLRACTIVE	W	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。
[0]	VECTRESET	W	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。

附录图 39-63 AIRCR 位分配

### 二进制点

PRIGROUP 域指示了二进制点的位置，该点将“中断优先级寄存器”中 PRI\_n 域分割为独立的组优先级和子优先级域。见下表列出了 PRIGROUP 值如何控制此分割。

	中断优先级级别值, PRI_N[7:0]			数目	
	二进制点 <sup>[1]</sup>	组优先级位	子优先级位	组优先级	子优先级
b010	bxxxxxx.000	[7:3]	无	32	1
b011	bxxxx.y000	[7:4]	[3]	16	2
b100	bxxx.yy000	[7:5]	[4:3]	8	4
b101	bxx.yyy000	[7:6]	[5:3]	4	8
b110	bx.yyyy000	[7]	[6:3]	2	16
b111	b.yyyyy000	无	[7:3]	1	32

附录图 39-64 优先级分组

注[1]: PRI\_n[7:0]域显示二进制点。X 指示一个组优先级域位，y 指示一个子优先级域位。位[3:0]不在 ES32 微控制器中使用。

备注：确定一个异常是否抢占时，只使用组优先级域，见“中断优先级分组”小节。

### 附录1.4.3.7 系统控制寄存器

SCR 控制着进入和离开低功耗状态的特性。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配见下表。



位	名称	功能
[31:5]	-	保留。
[4]	SEVONPEND	<p>在挂起位上发送事件：</p> <p>0：仅使能的中断或事件可以唤醒处理器，禁能的中断不能唤醒处理器。</p> <p>1：使能的事件和所有中断，包括禁能的中断，都可以唤醒处理器。当一个事件或中断进入挂起状态，事件信号将处理器从 WFE 中唤醒。如果处理器没有在等待一个事件，该事件被记录并影响下一 WFE。该处理器也可以在执行 SEV 指令或一个外部事件时被唤醒。</p>
[3]	-	保留。
[2]	SLEEPDEEP	<p>控制处理器是否使用睡眠或深度睡眠作为其低功率模式：</p> <p>0：睡眠</p> <p>1：深度睡眠。</p>
[1]	SLEEPONEXIT	<p>表示当从处理模式返回到线程模式时开始“退出时睡眠”：</p> <p>0：返回到线程模式时不进入到睡眠模式</p> <p>1：从 ISR 返回时进入睡眠或深度睡眠模式。</p> <p>将该位设为 1 会使能一个中断驱动应用程序以避免返回到空的主应用程序中。</p>
[0]	-	保留。

附录图 39-65 SCR 位分配

#### 附录1.4.3.8 配置和控制寄存器

CCR 控制着线程模式的进入，并使能：

- ◇ 通过 FAULTMASK 升级的 NMI、HardFault 和故障处理程序，以忽略总线故障；
- ◇ 被零除和非对齐访问的捕获；
- ◇ 非特权软件对 STIR 的访问，见表“STIR 位分配”。

CCR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

位分配见下表。

位	名称	功能
[31:10]	-	保留。
[9]	STKALIGN	<p>表示进入异常时的堆栈对齐。</p> <p>0：4 字节对齐</p> <p>1：8 字节对齐</p> <p>进入异常时，处理器使用压入堆栈的 PSR 位[9]来指示堆栈对齐。从异常返回时，这个堆栈位被用来恢复正确的堆栈对齐。</p>

位	名称	功能
[8]	BFHFNMIGN	使能时，使得以优先级位-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。它用于 HardFault、NMI 和 FAULTMASK 升级处理程序中：
		0：加载和存储指令引起的数据总线故障会引起锁定。
		1：以优先级-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。
		仅在处理程序和其数据处于绝对安全的存储器时将该位设为 1。一般将该位用于探测系统设备和桥接器以检测并纠正控制路径问题。
[7:5]	-	保留。
[4]	DIV_0_TRP	当处理器进行除 0 操作（SDIV 或 UDIV 指令）时，会导致故障或停止。
		0：不捕获除以零故障
		1：捕获除以零故障。
		当该位设为 0 时，除以零返回的商数为 0。
[3]	UNALIGN_TRP	使能非对齐访问捕获：
		0：不捕获非对齐半字和字访问
		1：捕获非对齐半字和字访问。
		如果该位设为 1，非对齐访问产生一个使用故障。无论 UNALIGN_TRP 是否设为 1，非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错。
[2]	-	保留。
[1]	USERSETMP END	使能对 STIR 的无特权软件访问，参见表“STIR 位分配”。
		0：禁能
		1：使能
		指示处理器如何进入线程模式：
[0]	NONEBASE THRDENA	0：处理器仅在没有有效异常时才能够进入线程模式。
		1：处理器可以从 EXC_RETURN 值控制下的任何级别进入线程模式，参见“异常返回”小节。

附录图 39-66 CCR 位分配

#### 附录1.4.3.9 系统处理程序优先级寄存器

SHPR1-SHPR3 寄存器用于设置有可配置优先级异常处理程序的优先级 0 到 31。

SHPR1-SHPR3 可按字节访问。这些寄存器的属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

系统故障处理程序，以及各处理程序的优先级域和寄存器如下：

处理程序	域	寄存器描述
存储器管理故障	PRI_4	见表“SHPR1 寄存器的位分配”
总线故障	PRI_5	
使用故障	PRI_6	
SVCall	PRI_11	见表“SHPR2 寄存器的位分配”
PendSV	PRI_14	见表“SHPR3 寄存器的位分配”
SysTick	PRI_15	

附录图 39-67 系统故障处理程序优先级域

每个 PRI\_N 域为 8 位宽，但是处理器仅实现每个域的位[7:3]，位[2:0]读取值为零并忽略写入值。

#### 系统处理程序优先级寄存器 1

位分配见下表。

位	名称	功能
[31:24]	PRI_7	保留
[23:16]	PRI_6	系统处理程序 6 的优先级，使用故障
[15:8]	PRI_5	系统处理程序 5 的优先级，总线故障
[7:0]	PRI_4	系统处理程序 4 的优先级，存储器管理故障

附录图 39-68 SHPR1 寄存器的位分配

#### 系统处理程序优先级寄存器 2

位分配见下表。

位	名称	功能
[31:24]	PRI_11	系统处理程序 11 的优先级，SVCall
[23:0]	-	保留

附录图 39-69 SHPR2 寄存器的位分配

### 系统处理程序优先级寄存器 3

位分配见下表。

位	名称	功能
[31:24]	PRI_15	系统处理程序 15 的优先级, SysTick 异常
[23:16]	PRI_14	系统处理程序 14 的优先级, PendSV
[15:0]	-	保留

附录图 39-70 SHPR3 寄存器的位分配

#### 附录1.4.3.10 系统处理程序控制和状态寄存器

SHCSR 使能系统处理程序, 并指示:

- ◇ 总线故障、存储器管理故障和 SVC 异常的挂起状态;
- ◇ 系统处理程序的有效状态。

SHCSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:19]	-	保留
[18]	USGFAULTENA	使用故障使能位, 设为 1 时使能 <sup>[1]</sup>
[17]	BUSFAULTENA	总线故障使能位, 设为 1 时使能 <sup>[1]</sup>
[16]	MEMFAULTENA	存储器管理故障使能位, 设为 1 时使能 <sup>[1]</sup>
[15]	SVCALLPENDEDED	SVC 调用挂起位, 如果异常挂起, 该位读为 1 <sup>[2]</sup>
[14]	BUSFAULTPENDEDED	总线故障异常挂起位, 如果异常挂起, 该位读为 1 <sup>[2]</sup>
[13]	MEMFAULTPENDEDED	存储器管理故障异常挂起位, 如果异常挂起, 该位读为 1 <sup>[2]</sup>
[12]	USGFAULTPENDEDED	使用故障异常挂起位, 如果异常挂起, 该位读为 1 <sup>[2]</sup>
[11]	SYSTICKACT	SysTick 异常有效位, 如果异常有效, 该位读为 1 <sup>[3]</sup>
[10]	PENDSVACT	PendSV 异常有效位, 如果异常有效, 该位读为 1
[9]	-	保留
[8]	MONITORACT	调试监控有效位, 如果调试监控有效, 该位读为 1
[7]	SVCALLACT	SVC 调用有效位, 如果 SVC 调用有效, 该位读为 1
[6:4]	-	保留
[3]	USGFAULTACT	使用故障异常有效位, 如果异常有效, 该位读为 1
[2]	-	保留
[1]	BUSFAULTACT	总线故障异常有效位, 如果异常有效, 该位读为 1
[0]	MEMFAULTACT	存储器管理故障异常有效位, 如果异常有效, 该位读为 1

附录图 39-71 SHCSR 位分配

注[1]: 使能位, 设为 1 使能异常, 或设为 0 禁能异常。  
 注[2]: 挂起位, 如果异常挂起, 该位读为 1, 或如果异常未挂起, 该位读为 0。可以向这些位写入值以改变异常的挂起状态。  
 注[3]: 有效位, 如果异常有效, 该位读为 1, 或如果异常无效, 该位读为 0。可以向这些位写入值以改变异常的有效状态, 但需关注本章节中的“注意”部分。  
 备注: 如果禁用某个系统处理程序, 并发生了对应的故障, 则处理器将该故障作为 HardFault。

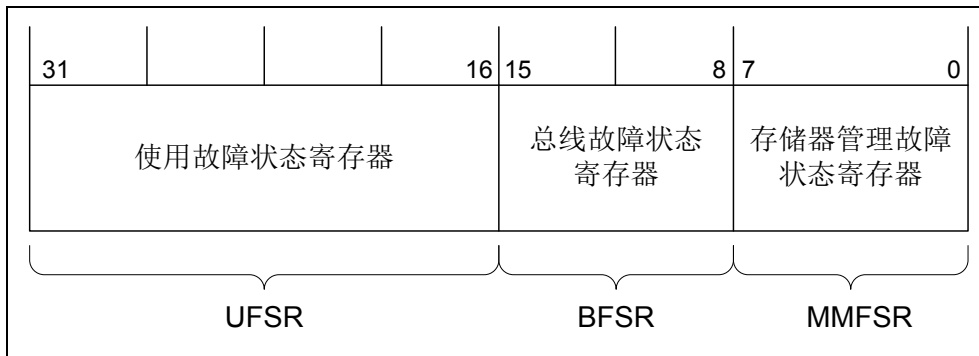
可写入此寄存器来改变系统异常的挂起或有效状态。OS 内核可以写入有效位进行上下文切换, 从而改变当前异常的类型。

注意

- ◇ 如果软件改变了此寄存器的某个有效位的值, 而未正确调整入栈的内容, 则可能会使处理器产生一个故障异常。要确保写入此寄存器的软件继续并在随后恢复当前的有效状态。
- ◇ 使能了系统处理程序后, 如果必须改变此寄存器中某个位的值, 则必须采用读取-修改-写入步骤, 以确保只改变需要的位。

附录1.4.3.11 可配置故障状态寄存器

CFSR 指示存储器管理故障、总线故障或使用故障的原因。CFSR 属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为:



附录图 39-72 可配置故障状态寄存器

以下各小节描述了构成 CSFR 的子寄存器:

- ◇ 表“MMFSR 位分配”
- ◇ 表“BFAR 位分配”
- ◇ 表“UFSR 位分配”

CFSR 可按字节访问。可如下访问 CSFR 或其子寄存器:

- ◇ 通过到 0xE00ED28 的字访问来访问整个 CSFR;
- ◇ 通过到 0xE00ED28 的字节访问来访问 MMFSR;
- ◇ 通过到 0xE00ED28 的半字访问来访问 MMFSR 和 BFSR;
- ◇ 通过到 0xE00ED29 的字节访问来访问 BFSR;
- ◇ 通过到 0xE00ED2A 的半字访问来访问 UFSR。

存储器管理故障状态寄存器

MMFSR 中标志指示存储器访问故障的原因。位分配见下表。

位	名称	功能
[7]	MMARVALID	存储器管理故障地址寄存器 (MMAR) 有效标志:
		0: MMAR 中的值不是一个有效故障地址
		1: MMAR 中保留一个有效故障地址。
		如果发生了一个存储器管理故障, 并由于优先级的原因升级成一个 HardFault, 那么 HardFault 处理程序必须将该位设为 0。 这样可以避免在返回到 MMAR 值已被覆写的压入堆栈的有效存储器管理故障处理程序时出现问题。
[6:5]	-	保留。
[4]	MSTKERR	进入异常时的入栈操作引起的存储器管理故障:
		0: 无入栈故障
		1: 进入异常时的入栈操作引起了一个或一个以上的访问违犯。
		当该位设为 1 时, 依然要对 SP 进行调节, 并且堆栈的上下文区域的值可能不正确。处理器没有向 MMAR 中写入故障地址。
[3]	MUNSTKERR	异常返回时的出栈操作引起的存储器管理故障:
		0: 无出栈故障
		1: 异常返回时的出栈操作已引起一个或一个以上的访问违犯
		该故障与处理程序相连, 这意味着当该位为 1 时, 原始的返回堆栈仍然存在。处理器不能对返回失败的 SP 进行调节, 并且不会执行新的存储操作。处理器没有向 MMAR 中写入故障地址。
[2]	-	保留
[1]	DACCVIOL	数据访问违犯标志:
		0: 无数据访问违犯故障
		1: 处理器试图在不允许执行操作的位置上进行加载和存储。
		当该位为 1 时, 异常返回的压入堆栈的 PC 值指向出错指令。 处理器已在 MMAR 中加载了目标访问的地址。
[0]	IACCVIOL	指令访问违犯标志:
		0: 无指令访问违犯错误
		1: 处理器试图从不允许执行操作的位置上进行指令获取。
		即使 MPU 被禁能, 这一故障也会在每次访问 XN 区时发生。 当该位为 1 时, 异常返回的压入堆栈的 PC 值指向出错指令。处理器没有向 MMAR 中写入故障地址。

附录图 39-73 MMFSR 位分配

总线故障状态寄存器

BFSR 中标志指示总线访问故障原因。位分配见下表。

位	名称	功能
[7]	BFARVALID	总线故障地址寄存器 (BFAR) 有效标志:
		0: BFAR 中的值不是有效故障地址
		1: BFAR 中保留一个有效故障地址。
		在地址已知的总线故障发生后处理器将该位设为 1。该位可以被其他故障清零, 例如之后发生的存储器管理故障。 如果发生总线故障, 并由于优先级原因升级为一个硬错误, 那么 HardFault 处理程序必须将该位设为 0。当返回至一个 BFAR 值已被覆写的压入堆栈的有效总线故障处理程序时, 这样做可以防止问题的出现。
[6:5]	-	保留。
[4]	STKERR	进入异常时的入栈操作引起的总线故障:
		0: 无入栈故障
		1: 进入异常时的入栈操作已引起一个或一个以上的总线故障。 当处理器将该位设为 1 时, 依然要对 SP 进行调节, 并且堆栈的上下文区域的值可能不正确。处理器没有向 BFAR 中写入故障地址。
[3]	UNSTKERR	异常返回时的出栈操作引起的总线故障:
		0: 无出栈故障
		1: 异常返回时的出栈操作已引起一个或一个以上的总线故障。 该故障与处理程序相连, 这意味着当处理器将该位设为 1 时, 原始的回堆栈仍然存在。处理器不能对返回失败的 SP 进行调节, 并且不会执行新的存储操作, 也未向 BFAR 中写入故障地址。
[2]	IMPRECISERR	非精确数据总线错误:
		0: 无非精确数据总线错误
		1: 已发生一个数据总线错误, 但是堆栈帧中的返回地址与引起错误的指令无关。 当处理器将该位设为 1 时, 不向 BFAR 中写入故障地址。 这是一个异步故障。因此, 如果在当前进程的优先级高于总线故障优先级时检测到该故障, 总线故障被挂起并仅在处理器从所有更高优先级进程中返回时开始变为有效。如果在处理器进入非精确总线故障的处理程序前发生一个精确故障, 那么处理程序同时对 IMPRECISERR 和其中一个精确故障状态位进行检测, 判断它们是否置位为 1。
[1]	PRECISERR	精确数据总线错误:

位	名称	功能
		0: 非精确数据总线错误
		1: 已发生一个数据总线错误, 且异常返回的压入堆栈的 PC 值指向引起故障的指令。
		当处理器将该位设为 1 时, 向 BFAR 中写入故障地址。
[0]	IBUSERR	指令总线错误:
		0: 无指令总线错误
		1: 指令总线错误。
		处理器检测到预取指令时的指令总线错误, 但仅在其试图签发故障指令时才将 IBUSERR 标志设为 1。
		当处理器将该位设为 1 时, 不向 BFAR 中写入故障地址。

附录图 39-74 BFSR 位分配



### 使用故障状态寄存器

UFSR 指示使用故障的原因。位分配见下表。

位	名称	功能
[15:10]	-	保留。
[9]	DIVBYZERO	<p>除以零使用故障：</p> <p>0：无除以零故障或除以零捕获未使能</p> <p>1：处理器已执行 SDIV 或 UDIV 指令（除以零）。</p> <p>当处理器将该位设为 1 时，异常返回的压入堆栈的 PC 值指向执行除以零的指令。</p> <p>通过将 CCR 中的 DIV_0_TRP 位设为 1 使能除以零捕获，参见表“CCR 位分配”。</p>
[8]	UNALIGNED	<p>非对齐访问使用故障：</p> <p>0：无非对齐访问故障，或非对齐访问捕获未使能</p> <p>1：处理器已进行了一次非对齐的存储器访问。</p> <p>通过将 CCR 中的 UNALIGN_TRP 位设为 1 来使能非对齐访问捕获，参见表“CCR 位分配”。</p> <p>非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错，与 UNALIGN_TRP 的设置无关。</p>
[7:4]	-	保留。
[3]	NOCP	<p>无协处理器使用故障。处理器不支持协处理器指令：</p> <p>0：试图访问一个协处理器未引起使用故障</p> <p>1：处理器已试图访问一个协处理器。</p>
[2]	INVPC	<p>EXC_RETURN 的无效 PC 加载引起的无效 PC 加载使用故障：</p> <p>0：没有发生无效 PC 加载使用故障</p> <p>1：处理器已试图将 EXC_RETURN 非法载入 PC，作为一个无效的上下文或一个无效的 EXC_RETURN 值。</p> <p>当该位被设为 1 时，异常返回的压入堆栈的 PC 值指向尝试执行非法 PC 加载的指令。</p>
[1]	INVSTATE	<p>无效状态使用故障：</p> <p>0：未发生无效状态使用故障</p> <p>1：处理器已试图执行一个非法使用 EPSR 的指令。</p> <p>当该位设为 1 时，异常返回的压入堆栈的 PC 值指向一个尝试非法使用 EPSR 的指令。</p> <p>如果一个未定义的指令使用了 EPSR，则该位不被置位为 1。</p>
[0]	UNDEFINSTR	未定义的指令使用故障：

位	名称	功能
		0: 无未定义的指令使用故障
		1: 处理器已试图执行一个未定义的指令。当该位设为 1 时, 异常返回的压入堆栈的 PC 值指向未定义的指令。 未定义的指令是一条不能被处理器译码的指令。

附录图 39-75 UFSR 位分配

注: UFSR 位是粘性 (sticky) 位。这就是说, 当一个或多个故障发生时, 相关的位被设为 1。被设为 1 的位只能通过向该位写入 1 或复位才能清零。

#### 附录1. 4. 3. 12 HardFault 状态寄存器

HFSR 提供关于激活 HardFault 处理程序的事件的信息。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。

此寄存器为可读并通过写入清零。也就是说, 此寄存器中的位可正常读取, 但向任何位写入 1 会将该位清零。位分配见下表。

位	名称	功能
[31]	DEBUGEVT	保留供调试使用。对寄存器执行写操作时, 必须向该位写入 0; 否则, 该行为不可预知。
[30]	FORCED	指示一个强制的 HardFault, 该 HardFault 由一个优先级可配置且无法处理的故障升级而来, 这一故障无法处理是由于优先级或可配置故障被禁能的原因: 0: 无强制的 HardFault 1: 强制的 HardFault。 当该位设为 1 时, HardFault 处理程序必须读其他故障状态寄存器以找出故障原因。
[29:2]	-	保留。
[1]	VECTTBL	指示一个在异常处理过程中读向量表而引起的总线故障: 0: 读向量表未引起总线故障 1: 读向量表引起了总线故障。 这一错误通常情况下都由 HardFault 处理程序来处理。 当该位设为 1 时, 异常返回的压入堆栈的 PC 值指向被异常抢占的指令。
[0]	-	保留。

附录图 39-76 HFSR 位分配

注: HFSR 位是粘性 (sticky) 位。这就是说, 当一个或多个故障发生时, 相关的位被设为 1。被设为 1 的位, 只能通过向该位写入 1 或复位才能清零。

#### 附录1.4.3.13 存储器管理故障地址寄存器

MMFAR 包括产生存储器管理故障的位置的地址。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为：

位	名称	功能
[31:0]	ADDRESS	当 MMFSR 的位 MMARVALID 设为 1 时，该域中保存的是产生存储器管理故障的单元地址。

附录图 39-77 MMFAR 位分配

当出现非对齐访问故障时，地址就是实际出错的地址。由于单个读或写指令可能被分割为多个对齐的访问，故障地址可能是请求访问尺寸范围内的任何地址。

MMFSR 中的标志指示故障原因以及 MMFAR 中的值是否有效。请参见表“MMFSR 位分配”。

#### 附录1.4.3.14 总线故障地址寄存器

BFAR 包括产生总线故障的位置的地址。其属性请参见表“系统控制模块寄存器汇总”中寄存器汇总。位分配为：

位	名称	功能
[31:0]	ADDRESS	当 BFSR 的位 BFARVALID 设为 1 时，该域中保存的是产生总线故障的单元地址。

附录图 39-78 BFAR 位分配

当发生非对齐访问故障时，BFAR 中地址为指令所请求的地址，即使该地址不是故障的地址。

BFSR 中的标志用于指示故障原因以及 BFAR 中的值是否有效。请参见表“BFSR 位分配”。

#### 附录1.4.3.15 系统控制模块设计提示和建议

要确保软件使用正确尺寸的对齐访问来访问系统控制模块寄存器：

- ◇ 除了 CFSR 和 SHPR1-SHPR3，必须使用对齐的字访问；
- ◇ 对于 CFSR 和 SHPR1-SHPR3，可使用字节或对齐的半字或字访问。

处理器不支持对系统控制模块寄存器的非对齐访问。

在故障处理程序中，为了确定真实的出错地址：

1. 读取并保存 MMFAR 或 BFAR 值。
2. 读取 MMFSR 中 MMARVALID 位，或 BFSR 中 BFARVALID 位。仅在此位为 1 时，MMFAR 或 BFAR 地址才有效。

软件必须按照此顺序，因为可能会有另一个更高优先级的异常改变 MMFAR 或 BFAR

的值。例如，如果更高优先级的处理程序抢占了当前故障处理程序，则其他故障可能会改变 MMFAR 或 BFAR 值。

### 附录1.4.3.16 系统定时器 SysTick

处理器有一个 24 位系统定时器 SysTick，它从重载值递减计数至零，在下一时钟边沿重载（绕回）加载寄存器中的值，然后在后续的时钟中递减计数。

注：当处理器停止调试时，计数器不会递减计数。

系统定时器寄存器有：

地址	名称	类型	所需特权	复位值
0xE000E010	CTRL	RW	特权	0x00000004
0xE000E014	LOAD	RW	特权	0x00000000
0xE000E018	VAL	RW	特权	0x00000000
0xE000E01C	CALIB	R	特权	0x000F423F <sup>[1]</sup>

附录图 39-79 系统定时器寄存器汇总

注[1]：SysTick 校准值。该值特用于 ES32 微控制器。

### 附录1.4.3.17 SysTick 控制和状态寄存器

SysTick CTRL 寄存器用于使能 SysTick 特性。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:17]	-	保留。
[16]	COUNTFLAG	从上次读取定时器开始，如果定时器计数到 0，则返回 1。
[15:3]	-	保留。
[2]	CLKSOURCE	指示时钟源： 0：外部时钟 1：处理器时钟。
[1]	TICKINT	使能 SysTick 异常请求： 0：向下计数至 0 不会使能 SysTick 异常请求 1：向下计数至 0 会使能 SysTick 异常请求。软件可以使用 COUNTFLAG 来判断 SysTick 是否曾计数到 0。
[0]	ENABLE	使能计数器： 0：计数器禁能 1：计数器使能。

附录图 39-80 SysTick 控制寄存器的位分配

当 ENABLE 设为 1 时,计数器从加载寄存器加载重载(RELOAD)值,然后递减计数。达到 0 时,它将 COUNTFLAG 设为 1,并可选择根据 TICKINT 的值使能 SysTick。然后它再次加载重载值并开始计数。

#### 附录1.4.3.18 SysTick 重载值寄存器

加载寄存器规定了载入 VAL 寄存器的起始值。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:0]	RELOAD	当计数器使能且计数器值到 0 时要载入 VAL 寄存器的值,参见“计算重载值”小节。

附录图 39-81 加载寄存器的位分配

#### 计算重载值

重载值可以是 0x00000001 至 0x00FFFFFF 范围内的任意值。可以以 0 为起始值,但这样没有效果,因为 SysTick 异常请求和 COUNTFLAG 在从 1 计数到 0 时激活。

根据其用途,计算得到重载值:

- ◇ 如要产生一个具有 N 个处理器时钟周期的连拍式(multi-shot)定时器,使用重载值 N-1。例如,如果每 100 时钟脉冲需要有一个 SysTick 中断,则将重载设到 99。
- ◇ 如要提供一个经过了 N 个处理器时钟周期延迟后的单次触发,则使用重载值 N。例如,如果要求 400 时钟脉冲后发生 SysTick 中断,则将重载设到 400。

#### 附录1.4.3.19 SysTick 当前值寄存器

VAL 寄存器包含 SysTick 计数器的当前值。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:0]	CURRENT	读取返回 SysTick 计数器的当前值。
		向该寄存器写入任意值都可以将该域清零,还会导致 SysTick CTRL.COUNTFLAG 位清零。

附录图 39-82 VAL 寄存器的位分配

#### 附录1.4.3.20 SysTick 校准值寄存器

CALIB 寄存器指示 SysTick 的校准性质。其属性请参见表“系统定时器寄存器汇总”中寄存器汇总。位分配见下表。

位	名称	功能
[31]	NOREF	是否有一个可用的单独参考时钟。该位的值在出厂时已预设,固定为 0。
[30]	SKEW	TENMS 的值是否精确。这会影响 SysTick 作为一个软件实时时钟的适合性。该

		位的值在出厂时已预设，固定为 1。
[29:24]	-	保留。
[23:0]	TENMS	该位的值在出厂时已预设，固定为 120000。

附录图 39-83 CALIB 寄存器的位分配

如果所用频率与工厂预设值的频率不同，则需要根据处理器时钟或外部时钟频率计算校准值。

#### 附录1.4.3.21 SysTick 设计提示和建议

SysTick 计数器使用处理器时钟运行。如果 SysTick 计数器靠其运行时，为了低功耗模式而停止了此时钟信号，则 SysTick 计数器将停止。

要确保软件使用对齐的字访问来访问 SysTick 寄存器。

#### 附录1.4.4 存储器保护单元

本节介绍存储器保护单元 (MPU)。

MPU 将存储器映射划分为多个区，并定义了每个区的位置、大小、访问权限和存储器属性。

它支持：

- ◇ 每个区独立的属性设置；
- ◇ 重叠区；
- ◇ 存储器属性导出到系统。

存储器属性影响对区的存储器访问的行为。Cortex-M3 MPU 定义了：

- ◇ 8 个单独的存储区 0-7；
- ◇ 1 个背景区。

当存储区重叠时，存储器访问受序号最大的区的属性影响。例如，区 7 的属性要优先于任何与区 7 重叠的区。

背景区具有与默认存储器映射相同的存储器访问属性，但只能从特权软件访问。

Cortex-M3 MPU 存储器映射是统一的。这就是说指令访问和数据访问具有相同的区设置。如果某个程序访问被 MPU 禁止的存储单元，则处理器产生一个存储器管理故障。这会导致故障异常，并可能导致 OS 环境中的进程终止。

在 OS 环境中，内核可根据要执行的进程，动态地更新 MPU 区设置。通常，嵌入式 OS 使用 MPU 进行存储器保护。

MPU 区的配置基于存储器的类型，见“存储区、类型和属性”小节。

下表列出了可能的 MPU 区属性。它们包括与多数微控制器实现无关的“可共享性”和缓存行为属性。该类实现的编程指南见表“微控制器的存储器属性”。

存储器类型	可共享性	其他属性	描述
非常有序	-	-	所有对非常有序存储器的访问都按照 程序顺序进行。所有非常有序区都假设是可共享的。
设备	可共享	-	一些处理器共享的存储器映射的外设。
	非共享	-	仅供一个处理器使用的存储器映射的外设。
正常	可共享	不可缓存	几个处理器共享的正常存储器。
		直写可缓存	
		写回可缓存	
	非共享	不可缓存	仅供一个处理器使用的正常存储器。
		直写可缓存	
		写回可缓存	

附录图 39-84 存储器属性汇总

使用 MPU 寄存器来定义 MPU 区及其属性。MPU 寄存器有：

地址	名称	类型	所需特权	复位值	描述
0xE000ED90	TYPE	R	特权	0x00000800	见表“类型寄存器的位分配”
0xE000ED94	CTRL	RW	特权	0x00000000	见表“MPU 控制寄存器的位分配”
0xE000ED98	RNR	RW	特权	0x00000000	见表“RNR 位分配”
0xE000ED9C	RBAR	RW	特权	0x00000000	见表“RBAR 位分配”
0xE000EDA0	RASR	RW	特权	0x00000000	见表“RASR 位分配”
0xE000EDA4	RBAR_A1	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDA8	RASR_A1	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”
0xE000EDAC	RBAR_A2	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDB0	RASR_A2	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”
0xE000EDB4	RBAR_A3	RW	特权	0x00000000	RBAR 的别名，参见表“RBAR 位分配”
0xE000EDB8	RASR_A3	RW	特权	0x00000000	RASR 的别名，参见表“RASR 位分配”

附录图 39-85 MPU 寄存器汇总

#### 附录1.4.4.1 MPU 类型寄存器

类型 (TYPE) 寄存器指示是否存在 MPU，如果存在，它支持多少个区。该寄存器属性请参见表“MPU 寄存器汇总”中的寄存器汇总。位分配见下表。

位	名称	功能
[31:24]	-	保留。
[23:16]	IREGION	指示支持的 MPU 指令区数目。 总是包含 0x00。MPU 存储器映射是统一的，描述参见 DREGION 域。
[15:8]	DREGION	指示支持的 MPU 数据区数目：
		0x08: 8 个 MPU 区。
[7:0]	-	保留。
[0]	SEPARATE	指示支持统一的或单独的指令和数据存储器映射：
		0: 统一的。

附录图 39-86 类型寄存器的位分配

#### 附录1.4.4.2 MPU 控制寄存器

MPU CTRL 寄存器用于：

- ◇ 使能 MPU；
- ◇ 使能默认存储器映射的背景区；
- ◇ 在 HardFault、不可屏蔽中断 (NMI) 和 FAULTMASK 升级的处理程序中时，使能 MPU 的使用。

MPU CTRL 的属性请参见表“MPU 寄存器汇总”中的寄存器汇总。位分配见下表。

位	名称	功能
[31:3]	-	保留。
[2]	PRIVDEFENA	使能供特权软件访问使用的默认存储器映射：
		0: 如果 MPU 使能，禁能默认存储器映射的使用。当任意存储器访问一个未被任何使能区覆盖的单元时会导致故障产生。
		1: 如果 MPU 使能，使能默认存储器映射的使用，将其作为背景区供特权软件访问使用。
		使能时，背景区充当区号-1。所有已定义且使能的区的优先级都高于这个缺省映射。 如果 MPU 禁能，处理器忽略该位的写入值。
[1]	HFNMIENA	在处于 HardFault、NMI 和 FAULTMASK 处理程序时，该位使能 MPU。
		当 MPU 使能时：
		0: 在处于 HardFault、NMI 和 FAULTMASK 处理程序时，MPU 被禁能，



位	名称	功能
		与 ENABLE 位的值无关。
		1: 在处于 HardFault、NMI 和 FAULTMASK 处理程序时, MPU 被使能。
		当 MPU 禁能时, 如果该位设为 1, 那么行为不可预知。
[0]	ENABLE	使能 MPU:
		0: MPU 禁能
		1: MPU 使能。

附录图 39-87 MPU 控制寄存器的位分配

当 ENABLE 和 PRIVDEFENA 都设置为 1 时:

- ◇ 对于特权访问, 默认存储器映射“存储器模型”小节所述。特权软件对任何不以使能的存储区为地址的访问, 其行为都如默认存储器映射的定义。
- ◇ 非特权软件对任何不以使能的存储区为地址的访问都会导致一个存储器管理故障。
- ◇ 无论 ENABLE 位的值如何, XN 和非常有序规则总是适用于“系统控制空间”。

当 ENABLE 位设为 1 时, 存储器映射的至少一个区必须被使能, 以便让系统能工作, 除非 PRIVDEFENA 位设为 1。如果 PRIVDEFENA 位设为 1, 且没有区被使能, 则只有特权软件可工作。

当 ENABLE 位设为 0 时, 系统使用默认存储器映射。这与没有实施 MPU 的情况具有相同的存储器属性, 见表“存储器访问行为”。默认存储器映射适用于特权软件和非特权软件的访问。

当使能 MPU 时, 对“系统控制空间”和向量表的访问总是被允许。其他区域的访问许可要看存储区的情况, 以及 PRIVDEFENA 是否设为 1。

除非 HFNMIENA 设为 1, 否则当处理器正为一个优先级为-1 或-2 的异常执行处理程序时, MPU 不被使能。只有在处理一个 HardFault 或 NMI 异常, 或当 FAULTMASK 被使能时, 才可能有这样的优先级。设置 HFNMIENA 位到 1, 会使能运行在这两个优先级的 MPU。

#### 附录1.4.4.3 MPU 区号寄存器

RNR 选择哪个存储区被 RBAR 和 RASR 寄存器引用。其属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

位分配见下表。

位	名称	功能
[31:8]	-	保留。
[7:0]	REGION	指示被 RBAR 和 RASR 寄存器引用的 MPU 区。 MPU 支持 8 个存储区, 因此, 该域的允许值范围为 0~7。

附录图 39-88 RNR 位分配

通常，在访问 RBAR 或 RASR 前，要将所需的区号写入该寄存器。但是，可通过写入 RBAR 将 VALID 置位为 1 而改变区号，见表“RBAR 位分配”。此写入更新 REGION 域的值。

#### 附录1.4.4.4 MPU 区基址寄存器

RBAR 定义了 RNR 所选择 MPU 区的基址，并可以更新 RNR 的值。其属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

将 VALID 置位为 1 写入 RBAR，可改变当前区号和更新 RNR。位分配见下表。

位	名称	功能
[31:N]	ADDR	区基址域。N 值取决于区域的大小。更多详情请参见“ADDR 域”小节。
[(N-1):5]	-	保留。
[4]	VALID	MPU 区号有效位：
		写：
		0：RNR 保持不变且处理器：
		更新 RNR 中规定区域的基址
		忽略 REGION 域的值
		1：处理器：
		将 RNR 的值更新为 REGION 域的值
		更新 REGION 域规定区域的基址。
[3:0]	REGION	读取值总是为 0。
		MPU 区的域：
		有关写操作行为，参见 VALID 域的描述。
		读取该位返回由 RNR 设定的当前区号。

附录图 39-89 RBAR 位分配

#### ADDR 域

ADDR 域为 RBAR 的位[31:N]。这个区域的大小由 RASR 中的 SIZE 域指定，它定义了 N 的值：

$N = \text{Log}_2$ （单位为字节的区大小），

如果 RASR 中区的大小被配置为 4GB，则没有有效的 ADDR 域。在此情况下，区占用了全部的存储器映射，基址为 0x00000000。

基址按区的大小调整。例如一个 64 kB 的区必须对齐 64 kB 的倍数，例如，在 0x00010000 或 0x00020000。

#### 附录1.4.4.5 MPU 区的属性和大小寄存器

RASR 定义了区的大小和 RNR 所指定的 MPU 区的存储器属性，并使能该区及其子区。此寄存器属性请参见表“MPU 寄存器汇总”中的寄存器汇总。

RASR 可使用字或半字访问来访问：位分配见下表。

- ◇ 高位半字保存区的属性；
- ◇ 低位半字保存区的大小以及区和子区的使能位。

位	名称	功能
[31:29]	-	保留。
[28]	XN	指令访问禁能位：
		0：指令获取使能
		1：指令获取禁能。
[27]	-	保留。
[26:24]	AP	访问许可域，参见表“AP 编码”。
[23:22]	-	保留。
[21:19, 17, 16]	TEX, C, B	存储器访问属性，参见表“TEX、C、B 和 S 的编码”。
[18]	S	可共享位，参见表“TEX、C、B 和 S 的编码”。
[15:8]	SRD	子区禁能位。该域的几个位：
		0：相应子区被使能
		1：相应子区被禁能 更多详情请参见“子区”小节。
		不支持 128 字节及更小的子区。对该域写属性时，向 SRD 域写入 0x00。
[7:6]	-	保留。
[5:1]	SIZE	指示 MPU 保护区的大小。最小值为 3 (b00010)，更多详情参见“SIZE 域的值”小节。
[0]	ENABLE	区域使能位。

附录图 39-90 RASR 位分配

有关访问权限的信息，见“MPU 访问权限属性”小节。

### SIZE 域的值

SIZE 域定义了由 RNR 指定的 MPU 存储区的大小。定义如下：

$$(\text{单位为字节的区的大小}) = 2 (\text{SIZE} + 1)$$

允许的最小区为 32B，对应的 SIZE 值是 4。下表给出了 SIZE 值示例，并列出了相应的区的大小和 RBAR 中 N 的值。

SIZE 值	区域大小	N 值 <sup>[1]</sup>	注
b00100 (4)	32 B	5	最小区域值
b01001 (9)	1 kB	10	-
b10011 (19)	1 MB	20	-

SIZE 值	区域大小	N 值 <sup>[1]</sup>	注
b11101 (29)	1 GB	30	-
b11111 (31)	4 GB	b01100	最大区域值

附录图 39-91 SIZE 域值示例

注[1]: RBAR 中, 参见表“AR 位分配”。

#### 附录1.4.4.6 MPU 访问权限属性

本节描述了 MPU 访问权限属性。RASR 的访问权限位 TEX、C、B、S、AP 和 XN 控制对相应存储区的访问。如果对没有所需权限的存储区域进行访问, 则 MPU 产生一个许可故障。

下表列出了 TEX、C、B、S 访问权限位的编码。

TEX	C	B	S	存储器类型	可共享性	其他属性
b000	0	0	x <sup>[1]</sup>	非常有序	可共享	-
		1	x <sup>[1]</sup>	设备	可共享	-
	1	0	0	正常	非共享	外部和内部直写, 无写分配。
			1		可共享	
		1	0	正常	非共享	外部和内部写回, 无写分配。
			1		可共享	
b001	0	0	0	正常	非共享	外部和内部不可缓存。
			1		可共享	
		1	x <sup>[1]</sup>	保留编码		
	1	0	x <sup>[1]</sup>	实现定义的属性。		
		1	0	正常	非共享	外部和内部写回, 写和读分配。
			1		可共享	
b010	0	0	x <sup>[1]</sup>	设备	非共享	非共享设备。
		1	x <sup>[1]</sup>	保留编码		-
	1	x <sup>[1]</sup>	x <sup>[1]</sup>	保留编码		-
b1BB	A	A	0	正常	非共享	缓存的存储器 <sup>[2]</sup> , BB=外部策略, AA=内部策略。
			1		可共享	

附录图 39-92 TEX、C、B 和 S 的编码

注[1]: MPU 忽略该位的值。

注[2]: 有关 AA 和 BB 位的编码, 参见下表“存储器属性编码的缓存策略”。

下表列出了存储器属性编码的缓存策略, TEX 值在 4~7 范围内。

编码, AA 或 BB	相应缓存策略
00	不可缓存
01	写回, 写和读分配
10	直写, 无写分配
11	写回, 无写分配

附录图 39-93 存储器属性编码的缓存策略

下表定义了特权和非特权软件访问权限的 AP 编码。

AP[2:0]	特权许可	非特权许可	描述
000	不可访问	不可访问	所有的访问都会产生一个许可故障
001	RW	不可访问	只可进行特权软件访问
010	RW	R	非特权软件进行写操作时会产生一个许可故障
011	RW	RW	完全访问
100	不可预知	不可预知	保留
101	R	不可访问	只能进行特权软件读操作
110	R	R	只能进行特权或非特权软件读操作
111	R	R	只能进行特权或非特权软件读操作

附录图 39-94 AP 编码

#### 附录1.4.4.7 MPU 不匹配

当一个访问违反 MPU 权限时, 处理器产生一个存储器管理故障, 见“异常和中断”小节。MMFSR 指示故障的原因。更多信息见表“MMFSR 位分配”

#### 附录1.4.4.8 更新一个 MPU 区

如要更新一个 MPU 区的属性, 需更新 RNR、RBAR 和 RASR 寄存器。可以对每个寄存器单独编程, 也可使用一个多字写入, 对全部这些寄存器编程。可使用 RBAR 和 RASR 别名利用一条 STM 指令同时对 4 个区编程。

##### 使用单独的字更新一个 MPU 区

配置一个区的简单代码:

```
; R1 = region number
; R2 = size/enable
```

```
; R3 = attributes
; R4 = address
LDR R0,=MPU_RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]        ; Region Number
STR R4, [R0, #0x4]        ; Region Base Address
STRH R2, [R0, #0x8]       ; Region Size and Enable
STRH R3, [R0, #0xA]       ; Region Attribute
```

如果已使能了要更改的区，则在写入新的区设置到 MPU 前要先禁用该区。例如：

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPU_RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]        ; Region Number
BIC R2, R2, #1            ; Disable
STRH R2, [R0, #0x8]       ; Region Size and Enable
STR R4, [R0, #0x4]        ; Region Base Address
STRH R3, [R0, #0xA]       ; Region Attribute
ORR R2, #1                ; Enable
STRH R2, [R0, #0x8]       ; Region Size and Enable
```

以下情况下，软件必须使用存储器屏障指令：

- ◇ 如果可能有未完成的存储器传输(如缓冲写入)，可能受到 MPU 设置更改的影响，则要在 MPU 设置前使用；
- ◇ 如果存储器传输必须使用新的 MPU 设置，则要在 MPU 设置后使用。
- ◇ 但是，如果 MPU 设置进程通过进入一个异常处理程序启动，或后边跟一个异常返回，则不需要存储器屏障指令，因为异常进入和异常返回机制会引起存储器屏障行为。

MPU 设置期间，软件不需要任何存储器屏障指令，因为它通过 PPB 访问 MPU，而 PPB 为一个非常有序型存储区。

例如，如果需要在编程序列后，让全部存储器访问行为立即生效，可以使用一条 DSB 指令和一条 ISB 指令。改变 MPU 设置后，例如上下文切换后，需要使用一个 DSB。如果利用跳转或调用进入一个或多个 MPU 区的编程代码，则需要使用 ISB。如果利用异常返回或是获取异常进入一个编程序列，则不需要 ISB。

#### 使用多字写入更新一个 MPU 区

根据信息的分割方式，可使用多字写入直接编程。考虑以下再编程 (reprogramming)：

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPU_RNR          ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]        ; Region Number
STR R2, [R0, #0x4]        ; Region Base Address
STR R3, [R0, #0x8]        ; Region Attribute, Size and Enable
```

使用一条 STM 指令来优化它：

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPU_RNR ; 0xE000ED98, MPU region number register
STM R0, {R1-R3} ; Region Number, address, attribute, size and enable
```

对于预封装的信息，可用双字实现此操作。这就是说，RBAR 包含所需的区号，并将 VALID 位设为 1，当数据是静态封装时，例如在一个启动加载程序中时，使用此方法：

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPU_RBAR ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0] ; Region base address and
; region number combined with VALID (bit 4) set to 1
STR R2, [R0, #0x4] ; Region Attribute, Size and Enable
```

使用一条 STM 指令来优化它：

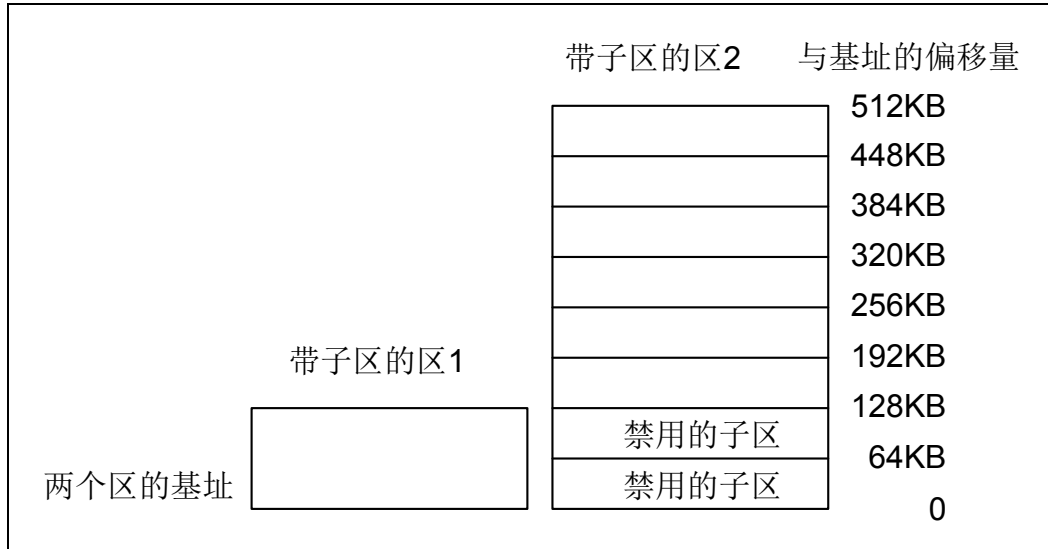
```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPU_RBAR ; 0xE000ED9C, MPU Region Base register
STM R0, {R1-R2} ; Region base address, region number and VALID bit,
; and Region Attribute, Size and Enable
```

### 子区

256 或更多字节的区可划分为 8 个相同大小的子区。可设置 RASR 中 SRD 域内相应的位来禁用某个子区，见表“RASR 位分配”。SRD 的最低位控制第一个子区，最高位控制最后一个子区。禁用某个子区就意味着，另一个与禁用区重叠的区取代了它实现了匹配。如果没有其他使能的区与禁用的子区重叠，MPU 发出一个故障。

32、64 和 128 个字节的区不支持子区，使用这些大小的区时，必须将 SRD 域设置为 0x00，否则 MPU 行为不可预知。

SRD 使用示例：两个具有相同基址的区重叠。区 1 为 128 kB，区 2 为 512 kB。为了保证区 1 的属性应用到第一个 128 kB 区，将区 2 的 SRD 域设置为 b00000011，禁用前两个子区，如下图所示。



附录图 39-95 SRD 使用示例

#### 附录1.4.4.9 MPU 设计提示和建议

为了避免不期望的行为，在更新某个中断处理程序可能访问的区属性前，禁用中断。确保软件使用大小正确的对齐访问来访问 MPU 寄存器：

- ◇ 除 RASR 外，必须使用对齐字访问；
- ◇ 对于 RASR，可使用字节、对齐半字或字访问。

处理器不支持对 MPU 寄存器的非对齐访问。

当设置 MPU 时，如果 MPU 已经编程，则禁用未使用的区，以防止之前任何区的设置影响到新的 MPU 设置。

#### 微控制器的 MPU 配置

通常，一个微控制器系统只有单个处理器且没有缓存。在该类系统中，MPU 编程如下：

存储区	TEX	C	B	S	存储器类型和属性
Flash 存储器	b000	1	0	0	正常存储器，非共享，直写
内部 SRAM	b000	1	0	1	正常存储器，可共享，直写
外部 SRAM	b000	1	1	1	正常存储器，可共享，写回，写分配
外设	b000	0	1	1	设备存储器，可共享

附录图 39-96 微控制器的存储器属性

在大多数微控制器执行中，可共享性和缓存策略的属性都不影响系统行为。然而，如果 MPU 区使用了这些设置，可使应用代码具有更好的可移植性。所列出的值是针对典型情况。在特殊系统中，例如多处理器设计或具有单独 DMA 引擎的设计中，可共享性属性可能会很重要。在这些情况下，请参考存储设备制造商的建议。



## 附录1.5 ARM Cortex-M3 用户指南：术语表

**中止**——向处理器表示一个存储器访问相关的值无效的一种机制。中止可能由外部或内部存储器系统引起，作为试图访问无效指令或数据存储器的结果。

**对齐**——如果保存在一个地址的数据项能够被定义数据大小的字节数整除，则此数据项被称为是对齐的。对齐的字和半字地址分别可被 4 和 2 整除。因此，术语字对齐和半字对齐规定，其地址能够分别被 4 和 2 整除。

**分组寄存器**——一个寄存器有多个物理副本，处理器的状态决定了使用哪个副本。堆栈指针 SP (R13) 就是一个分组寄存器。

**基址寄存器**——在指令描述中，加载或存储指令指定一个寄存器，用于保持指令地址计算的基值。根据指令及其寻址模式，可在基址寄存器值上加或减一个偏移量，就形成了要发送到存储器的地址。亦见索引寄存器。

**大端 (big-endian)**——字节定序方案，其中递减有效字节保存在存储器的递增地址中。亦见字节不变、字节序、小端。

**大端存储器**——具有以下特点的存储器：

- ◇ 字对齐地址上的字节或半字为此地址的字中的最高有效字节或半字。
- ◇ 半字对齐地址的字节为此地址的半字中的最高有效字节。

**断点**——断点是调试器提供的一个机制，用来识别程序执行被停止的指令。断点由程序员插入，以便能够检查程序执行中的寄存器内容，存储单元和某个固定点的变量值，从而检查程序工作是否正常。成功完成程序测试后，断点被移除。

**字节不变 (byte-invariant)**——在字节不变系统中，当在小端和大端操作间切换时，存储器每个字节的地址保持不变。当把大于 1 个字节的数据项从存储器加载或存储到存储器时，根据存储器访问的字节序，将构成该数据项的字节安排为正确的顺序。

ARM 的字节不变实现还支持非对齐的半字和字存储器访问。多字访问按字对齐。

**缓存**——片上或片外快速访问存储单元模块，位于处理器和主存储器之间，用于存储和检索常用指令、数据或同时包括指令和数据的副本。这样可大大增加存储器访问的平均速度，改善处理器性能。

**条件域**——指令中一个 4 位的域，它指定了指令可以执行的条件。

**上下文**——对于一个多任务操作系统，各进程运行的环境。在 ARM 处理器中，它仅限于表示可访问的存储器物理地址范围，以及相关的存储器访问权限。

**协处理器**——协助主处理器的一种处理器。Cortex-M3 不支持任何协处理器。

**调试器**——一种包含某个程序的调试系统，用于检测、定位和纠正软件故障，还包括支持软件调试的定制硬件。

**直接存储器访问 (DMA)**——无需处理器对相关数据执行任何访问，而直接访问主存的操作。

**双字**——一个 64 位数据项。除非另外说明，其内容被视为一个无符号整数。

**双字对齐**——存储器地址可被 8 整除的数据项。

**字节序 (endianness)** —— 字节定序。该机制用来确定存储在存储器中的一个数据字的连续字节的顺序。是系统的存储器映射的一个方面。亦见小端和大端。

**异常** —— 中断程序执行的一个事件。当发生异常时，处理器挂起正常的程序流，并开始从相应异常向量所指示的地址执行。指示的地址存有异常处理程序的第一条指令。

一个异常可以是一个中断请求、一个故障，或软件产生的系统异常。故障包括试图进行非法的存储器访问、试图在非法的处理器状态下执行某条指令，以及试图执行一条未定义的指令。

**异常服务程序** —— 亦见中断处理程序。

**异常向量** —— 亦见中断向量。

**平板地址映射 (flat address mapping)** —— 一种存储器组织体系，其中存储器空间中每个物理地址都与对应的虚拟地址相同。

**半字** —— 一个 16 位的数据项。

**非法指令** —— 结构上未定义的指令。

**实现定义的 (Implementation-defined)** —— 结构上未定义，但在单独实现中定义并证明的。

**实现特定的 (Implementation-specific)** —— 结构上未定义，且不必在单独实现中证明。

当有多个实现选项可供选择，且所选选项不影响软件兼容性时，使用该“实现特定的”。

**索引寄存器** —— 在某些加载和存储指令描述中，此寄存器的值被作为一个偏移量，将其加入基址寄存器值或从基址寄存器中减去，以形成发送到存储器的地址。某些寻址模式可以选择在加法或减法前，使能被移位的索引寄存器值。亦见基址寄存器。

**指令周期数** —— 指令占用流水线执行阶段所消耗的周期数。

**中断处理程序** —— 发生中断时，处理器的控制权传递到这个程序。

**中断向量** —— 存储器低地址空间或存储器高地址空间（如果配置了高位向量）内一些固定地址，存有相应中断处理程序的第一条指令。

**小端 (little-endian)** —— 字节定序机制，其中一个数据字中的递增有效的字节保存在存储器的递增地址中。亦见大端、字节不变、字节序。

**小端存储器** —— 具有以下特点的存储器：

- ◇ 字对齐地址中的字节或半字为此地址字中的最低有效字节或半字。
- ◇ 半字对齐地址中的字节为此地址半字中的最低有效字节。

亦见大端存储器。

**加载/存储架构** —— 一种处理器架构，数据处理操作只针对寄存器的内容，而不直接对存储器内容。

**存储器保护单元 (MPU)** —— 控制对存储器模块访问权限的硬件。MPU 不进行任何地址转换。

**预取** —— 在流水线处理器中，当前面的指令执行完成前，从存储器获取指令以填充流水线的过程。预取一条指令并不意味着该指令一定会执行。

**读**——读定义为具有一种加载语义的存储器操作。读包括 Thumb 指令 LDM、LDR、LDRSH、LDRH、LDRSB、LDRB 和 POP。

**区**——存储器空间的分区。

**保留**——在控制寄存器或指令格式中，如果某个域被实现定义了，或者在该域不为零时会产生不可预知的结果，则该域是保留域。这些域保留用于将来的结构扩展，或是由实现特定使用。所有未被实现使用的保留位都必须写为 0，且读取为 0。

**应为 1 (Should Be One, SBO)**——软件写 1 或位域全写 1。写入 0 会产生不可预知的结果。

**应为 0 (Should Be Zero, SBZ)**——软件写 0 或位域全写 0。写入 1 会产生不可预知的结果。

**应为 0 或保留 (Should Be Zero or Preserved, SBZP)**——软件写 0 或位域全写 0，或通过前面读出的相同处理器上相同域的值写回来保留。

**线程安全 (Thread-safe)**——在多任务环境中，线程安全功能在访问共享资源时使用保护机制，以确保操作正确，而不会有共享访问冲突的风险。

**Thumb 指令**——指定处理器完成一个操作的一个或两个半字。Thumb 指令必须按半字对齐。

**非对齐**——存储在不能被定义的数据大小的字节数整除的地址的数据项被称为“非对齐”。例如，存储在不能被 4 整除的地址的一个字。

**不可预知 (UNP)**——不能确知的行为。不可预知的行为不代表安全漏洞。不可预知的行为不得让处理器或系统的任何部分停止或挂起。

**热复位**——又称内核复位。它对处理器的大部分元件初始化，调试控制器和调试逻辑除外。

这种类型的复位在使用处理器的调试功能时非常有用。

**WA**——见写分配。

**WB**——见写回。

**字**——一个 32 位的数据项。

**写**——写定义为具有加载语义的存储器操作。写包括 Thumb 指令 STM、STR、STRH、STRB 和 PUSH。

**写分配 (write-allocate)**——在一个写分配缓存中，存储数据的缓存未命中会使一个缓存线被分配到缓存中。

**写回 (write-back)**——在写回缓存中，仅在缓存未命中后行替换时，数据被强制清出缓存，才将数据写入主存储器。否则，处理器的写入只更新缓存。这又称回拷 (copyback)。

**写缓冲**——位于数据缓存和主存之间的高速存储器模块，用作一个 FIFO 缓冲区，目的是优化到存储器的存储。

**直写 (write-through)**——在直写缓存中，数据在缓存被更新的同时写入主存储器。

## 第40章 修订历史

版本	修订日期	修订内容
V1.0	2020.05.06	初版发布