

WS8100

芯片数据手册

修订记录

修订 版本	描述	作者
1.0	初稿完成	Lhx
1.1	芯片特性、外设描述更新	Lhx、Zhlj
1.3	检查更新外设，整理格式	Lhx, Lkk
1.4	增加 ESD、晶振、回流焊电气特性 增加温度、电压传感器曲线	Lhx
1.5	删除简介中信息	Lhx
1.6	修复几处错误点	Lkk
1.7	标注出了低功耗唤醒源	Zhlj
1.8	更新内容	Lhx

目录

1	芯片简介.....	10
1.1	特性	10
2	存储器和总线架构.....	11
2.1	系统架构	11
2.2	存储器映射	11
2.2.1	外设地址映射	12
2.2.2	嵌入式RAM.....	13
2.2.3	位段访问	13
2.2.4	Flash特性	14
3	芯片特性说明	15
3.1	电气特性	15
3.1.1	极限参数	15
3.1.2	功耗	15
3.1.3	射频	16
3.1.4	ESD	16
3.1.5	晶振	17
3.1.6	回流焊曲线	17
4	时钟与复位管理（CRM）	18
4.1	外设时钟管理.....	18
4.2	软复位.....	18
4.3	寄存器描述	18
4.3.1	地址映射表	18
4.3.2	时钟门控控制寄存器1（CG_CTRL）	18
4.3.3	软件复位寄存器（SOFT_RST）	20
5	系统控制（SYSCTRL）	23
5.1	时钟	23
5.1.1	外部时钟源	23
5.1.2	HFCLK时钟	23
5.1.3	HCLK时钟	23
5.1.4	PCLK时钟	23
5.1.5	BTCORECLK时钟	24
5.2	低功耗控制	24
5.2.1	电源域划分	24
5.2.2	低功耗模式	24
5.2.3	低功耗模式进入	25
5.2.4	低功耗模式退出	25
5.2.5	模式切换状态图	26
5.3	外设控制	26
5.4	寄存器描述	27
5.4.1	地址映射表	27
5.4.2	高速时钟源选择寄存器（HFCLK_SEL）	28
5.4.3	低速时钟源选择寄存器（LFCLK_SEL）	29
5.4.4	时钟频率选择寄存器（FREQ_SEL）	29
5.4.5	时钟校准控制寄存器（CALIB_EN）	30
5.4.6	时钟校准值寄存器（CALIB_VAL）	31

5.4.7	系统休眠模式配置寄存器 (SLEEP_CTRL)	31
5.4.8	系统休眠唤醒类型配置寄存器 (WKUP_TYPE)	32
5.4.9	系统休眠唤醒源配置寄存器 (WKUP_SRC)	34
5.4.10	MCU休眠唤醒配置寄存器 (WAKE_MCU_CTRL)	35
5.4.11	DMA硬件握手外设分配寄存器 (DMA_CH_HS)	36
5.4.12	Timer挂起控制寄存器 (TIMER_PAUSE)	38
5.4.13	SPI工作模式配置寄存器 (SPI_SLAVE_SEL)	39
5.4.14	RTC预分频装载寄存器 (RTC_DIV_VAL)	39
5.4.15	RTC预分频器当前值寄存器 (RTC_DIV_CNT)	40
6	通用输入输出 (GPIO)	41
6.1	GPIO功能描述	41
6.1.1	通用I/O (GPIO)	41
6.1.2	单独的位设置或清除	42
6.1.3	外部中断	42
6.1.4	外部唤醒事件	42
6.1.5	I/O功能复用	42
6.2	GPIO寄存器	44
6.2.1	地址映射表	44
6.2.2	数据寄存器 (Px_IOD) (x=A..D)	45
6.2.3	置位/复位寄存器 (Px_BSR) (x=A..D)	45
6.2.4	控制寄存器 (Px_CON) (x=A..D)	46
6.2.5	GPIO A-D [3:0] 模式寄存器 (Px_L_MODE) (x=A..D)	46
6.2.6	GPIO A-D [7:4] 模式寄存器 (Px_H_MODE) (x=A..D)	47
6.2.7	中断类型配置寄存器 (INTP_TYPE)	48
6.2.8	中断源配置寄存器 (INTP_SRC)	49
6.2.9	中断状态寄存器 (INTP_STA)	50
7	RTC简介	52
7.1	RTC操作说明	52
7.2	RTC寄存器	52
7.2.1	地址映射表	52
7.2.2	RTC当前计数寄存器 (RTC_CCVR)	53
7.2.3	RTC计数匹配寄存器 (RTC_CMR)	53
7.2.4	RTC计数器装载寄存器 (RTC_CLR)	54
7.2.5	RTC计数器控制寄存器 (RTC_CCR)	54
7.2.6	RTC中断状态寄存器 (RTC_STAT)	55
7.2.7	RTC中断原始状态寄存器 (RTC_RSTAT)	55
7.2.8	RTC中断结束寄存器 (RTC_EOI)	56
8	看门狗 (WDT)	57
8.1	看门狗外设时钟	57
8.2	计数器 (Counter)	57
8.3	计数器预设值 (Timeout Period Values)	57
8.4	启用看门狗 (WatchDog Enable)	57
8.5	系统复位/中断 (System Resets)	57
8.6	寄存器描述	58
8.6.1	地址映射表	58
8.6.2	看门狗控制寄存器 (WDT_CR)	59
8.6.3	看门狗计数器 (WDT_CCVR)	59
8.6.4	看门狗计数器重置寄存器 (WDT_CRR)	60
8.6.5	看门狗中断状态寄存器 (WDT_STAT)	60
8.6.6	看门狗中断清除寄存器 (WDT_EOI)	61
8.6.7	看门狗匹配值寄存器 (WDT_RLD)	62

9	定时器 (TIMER)	63
9.1	定时器简介	63
9.2	定时器外设时钟	63
9.3	通用定时器	63
9.3.1	通用定时器的两种模式	63
9.3.2	中断处理	63
9.4	PWM模式	64
9.4.1	PWM工作模式	64
9.4.2	PWM周期及占空比设定	64
9.5	寄存器描述	64
9.5.1	地址映射表	64
9.5.2	自动重载计数器 (TimerNLoadCount) (N=0...5)	65
9.5.3	自动重载计数器2 (TimerNLoadCount2) (N=0...5)	65
9.5.4	当前计数器值 (TimerNCurrentValue) (N=0...5)	66
9.5.5	控制寄存器 (TimerNControlReg) (N=0...5)	66
9.5.6	中断清除寄存器 (TimerNEOI) (N=0...5)	67
9.5.7	中断状态寄存器 (TimerNIntStatus) (N=0...5)	68
9.5.8	全局中断清除寄存器 (TimersEOI)	68
9.5.9	全局原中断状态寄存器 (TimersRawIntStatus)	69
10	UART简介	71
10.1	接收/发送FIFO	71
10.1.1	接收/发送FIFO介绍	71
10.1.2	接收/发送FIFO	71
10.1.3	接收/发送FIFO中断使用	71
10.1.4	接收/发送FIFO访问模式	71
10.2	UART外设时钟	72
10.3	中断 (Interrupt)	72
10.4	可编程THRE中断 (Programmable THRE Interrupt)	72
10.5	DMA支持	73
10.6	寄存器描述	73
10.6.1	地址映射表	73
10.6.2	接收缓存寄存器 (RBR)	75
10.6.3	发送保持寄存器 (THR)	76
10.6.4	分频寄存器_高 (DLH)	76
10.6.5	分频寄存器_低 (DLL)	77
10.6.6	中断使能寄存器 (IER)	78
10.6.7	中断标志寄存器 (IIR)	80
10.6.8	FIFO控制寄存器 (FCR)	82
10.6.9	Line控制寄存器 (LCR)	84
10.6.10	Modem控制寄存器 (MCR)	86
10.6.11	Line状态寄存器 (LSR)	89
10.6.12	Modem状态寄存器 (MSR)	93
10.6.13	暂存器寄存器 (SCR)	97
10.6.14	低功率除数低位锁存器寄存器 (LPDLL)	98
10.6.15	低功率除数高位锁存器寄存器 (LPDLH)	99
10.6.16	影子接收缓存寄存器 (SRBR)	100
10.6.17	影子发送缓存寄存器 (STHR)	101
10.6.18	FIFO访问使能寄存器 (FAR)	102
10.6.19	读发送FIFO寄存器 (TFR)	103
10.6.20	写接收FIFO寄存器 (RFW)	104
10.6.21	UART状态寄存器 (USR)	105

10.6.22	发送FIFO数据量 (TFL)	107
10.6.23	接收FIFO数据量 (RFL)	107
10.6.24	软复位寄存器 (SRR)	108
10.6.25	发送请求影子寄存器 (SRTS)	109
10.6.26	Break信号控制影子寄存器 (SBCR)	111
10.6.27	DMA模式影子寄存器 (SDMAM)	112
10.6.28	FIFO使能影子寄存器 (SFE)	113
10.6.29	接收FIFO满阈值设置影子寄存器 (SRT)	113
10.6.30	发送FIFO空阈值设置影子寄存器 (STET)	114
10.6.31	发送暂停寄存器 (HTX)	115
10.6.32	DMA软应答 (DMASA)	116
11	串行外设接口(SPI)	117
11.1	SPI简介	117
11.2	SPI主要特点	117
11.3	SPI功能描述	117
11.3.1	SPI外设时钟及要求	117
11.3.2	接收/发送FIFO	118
11.3.3	中断类型	118
11.3.4	Motorola SPI通行协议	119
11.3.5	SPI主/从模式选择	120
11.3.6	SPI主模式配置	120
11.3.7	SPI主模式数据收发	121
11.3.8	SPI从模式配置	121
11.3.9	SPI从模式数据收发	121
11.3.10	DMA操作	122
11.4	SPI寄存器描述	122
11.4.1	地址映射表	122
11.4.2	控制寄存器0 (CTRLR0)	123
11.4.3	控制寄存器1 (CTRLR1)	126
11.4.4	使能寄存器 (SSIENR)	126
11.4.5	Microwire控制寄存器 (MWCR)	127
11.4.6	从设备选择寄存器 (SER)	129
11.4.7	波特率寄存器 (BAUDR)	129
11.4.8	发送FIFO阈值寄存器 (TXFTLR)	130
11.4.9	接收FIFO阈值寄存器 (RXFTLR)	131
11.4.10	发送FIFO数据量寄存器 (TXFLR)	133
11.4.11	接收FIFO数据量寄存器 (RXFLR)	133
11.4.12	状态寄存器 (SR)	134
11.4.13	中断屏蔽寄存器 (IMR)	135
11.4.14	中断状态寄存器 (ISR)	136
11.4.15	原中断状态寄存器 (RISR)	137
11.4.16	发送FIFO上溢中断清除寄存器 (TXOICR)	138
11.4.17	接收FIFO上溢中断清除寄存器 (RXOICR)	139
11.4.18	接收FIFO下溢中断清除寄存器 (RXUICR)	139
11.4.19	多主机竞争中断清除寄存器 (MSTICR)	140
11.4.20	全局中断清除寄存器 (ICR)	140
11.4.21	DMA控制寄存器 (DMACR)	141
11.4.22	DMA发送数据阈值寄存器 (DMATDLR)	141
11.4.23	DMA接收数据阈值寄存器 (DMARDLR)	143
11.4.24	数据寄存器 (DR)	144
11.4.25	接收采样延迟寄存器 (RX_SAMPLE_DLY)	144

11.4.26 其他相关寄存器	145
12 两线式串行总线 (IIC)	146
12.1 IIC简介	146
12.2 I2C总线时序	146
12.2.1 I2C总线传输过程	146
12.2.2 I2C总线地址传输格式	147
12.2.3 I2C总线地址的reserved	147
12.2.4 I2C总线的速度类型	147
12.2.5 I2C总线的restart	148
12.2.6 I2C总线的占用	148
12.2.7 I2C总线的占用	149
12.2.8 I2C总线的竞争	150
12.3 寄存器描述	150
12.3.1 地址映射表	150
12.3.2 I2C控制寄存器 (IC_CON)	152
12.3.3 I2C目标地址寄存器 (IC_TAR)	156
12.3.4 I2C从机地址寄存器 (IC_SAR)	158
12.3.5 I2C高速主机模式地址码寄存器 (IC_HS_MADDR)	159
12.3.6 I2C数据命令寄存器 (IC_DATA_CMD)	160
12.3.7 I2C标准时钟高速计数寄存器 (IC_SS_SCL_HCNT)	162
12.3.8 I2C标准时钟低速计数寄存器 (IC_SS_SCL_LCNT)	163
12.3.9 I2C快速时钟高速计数寄存器 (IC_FS_SCL_HCNT)	164
12.3.10 I2C快速时钟低速计数寄存器 (IC_FS_SCL_LCNT)	166
12.3.11 I2C高速时钟高速计数寄存器 (IC_HS_SCL_HCNT)	167
12.3.12 I2C高速时钟低速计数寄存器 (IC_HS_SCL_LCNT)	168
12.3.13 I2C中断状态寄存器 (IC_INTR_STAT)	169
12.3.14 I2C屏蔽状态寄存器 (IC_INTR_MASK)	171
12.3.15 I2C原中断状态寄存器 (IC_RAW_INTR_STAT)	173
12.3.16 I2C接收阈值寄存器 (IC_RX_TL)	177
12.3.17 I2C发送阈值寄存器 (IC_TX_TL)	178
12.3.18 I2C中断清除寄存器 (IC_CLR_INTR)	179
12.3.19 I2CRX_UNDER中断清除寄存器 (IC_CLR_RX_UNDER)	179
12.3.20 I2CRX_OVER中断清除寄存器 (IC_CLR_RX_OVER)	180
12.3.21 I2CTX_OVER中断清除寄存器 (IC_CLR_TX_OVER)	180
12.3.22 I2CTX_OVER中断清除寄存器 (IC_CLR_RD_REQ)	181
12.3.23 I2CTX_ABRT中断清除寄存器 (IC_CLR_TX_ABRT)	181
12.3.24 I2CTX_ABRT中断清除寄存器 (IC_CLR_RX_DONE)	182
12.3.25 I2CACTIVITY中断清除寄存器 (IC_CLR_ACTIVITY)	182
12.3.26 I2CSTOP_DET中断清除寄存器 (IC_CLR_STOP_DET)	183
12.3.27 I2CSTART_DET中断清除寄存器 (IC_CLR_START_DET)	183
12.3.28 I2CGEN_CALL中断清除寄存器 (IC_CLR_GEN_CALL)	184
12.3.29 I2C使能寄存器 (IC_ENABLE)	184
12.3.30 I2C状态寄存器 (IC_STATUS)	186
12.3.31 I2C发送等级寄存器 (IC_TXFLR)	188
12.3.32 I2C接收等级寄存器 (IC_RXFLR)	189
12.3.33 I2CSDA保持时长寄存器 (IC_SDA_HOLD)	189
12.3.34 I2C发送中断源寄存器 (IC_TX_ABRT_SOURCE)	190
12.3.35 I2C从机响应寄存器 (IC_SLV_DATA_NACK_ONLY)	194
12.3.36 I2CDMA控制寄存器 (IC_DMA_CR)	195
12.3.37 I2CDMA写控制寄存器 (IC_DMA_TDLR)	196
12.3.38 I2CDMA读控制寄存器 (IC_DMA_RDLR)	197

12.3.39	I2CSDA启动寄存器 (IC_SDA_SETUP)	197
12.3.40	I2CACK生成调用寄存器 (IC_ACK_GENERAL_CALL)	198
12.3.41	I2C使能状态寄存器 (IC_ENABLE_STATUS)	199
12.3.42	I2CSS/FS尖峰抑制极限寄存器 (IC_FS_SPKLEN)	201
12.3.43	I2C HS尖峰抑制极限寄存器 (IC_HS_SPKLEN)	202
13	通用ADC(GPADC).....	203
13.1	ADC简介	203
13.2	ADC特性	203
13.3	电压传感器	203
13.4	温度传感器	204
13.5	ADC寄存器	204
13.5.1	地址映射表	204
13.5.2	GPADC配置寄存器 (GPADC_CONF)	205
13.5.3	GPADC控制寄存器 (GPADC_CTRL)	206
13.5.4	GPADC FIFOLEVEL寄存器 (GPADC_FIFOLEVEL)	207
13.5.5	GPADC Threshold寄存器 (GPADC_Threshold)	208
13.5.6	GPADC FIFO FLUSH寄存器 (GPADC_FIFO_FLUSH)	208
13.5.7	GPADC FIFO DATA寄存器 (GPADC_FIFO_DATA)	209
13.5.8	中断状态寄存器 (GPADC_STAT)	210
13.5.9	中断控制寄存器 (GPADC_INT_CTRL)	211
13.5.10	GPADC数据寄存器 (GPADC_DATA)	212
13.5.11	DMA控制寄存器 (DMA_CR)	213
13.5.12	DMA接收FIFOLEVEL寄存器 (DMA_RDLR)	213
14	正交解码器(QuadDec).....	215
14.1	QuadDec简介	215
14.2	QuadDec特性	215
14.2.1	索引输入	215
14.2.2	数据过滤	215
14.2.3	中断输出	216
14.2.4	状态跃变	216
14.3	使用说明	216
14.4	QuadDec寄存器	216
14.4.1	地址映射表	216
14.4.2	Quadrature Control寄存器 (QCR)	217
14.4.3	Quadrature Status寄存器 (QSR)	218
14.4.4	Quadrature Interrupt enable寄存器 (QIR)	220
14.4.5	Quadrature Count Read / Write寄存器 (QRW)	221
14.4.6	Quadrature Sample Clock Frequency Selection寄存器 (QCK)	222
15	梯形键盘扫描(KeyBoard).....	223
15.1	KeyBoard简介	223
15.2	KeyBoard特性	223
15.3	KeyBoard寄存器	225
15.3.1	地址映射表	225
15.3.2	KPC使能寄存器 (KPC_CTRL)	225
15.3.3	KPC配置寄存器1 (KPC_CFG1)	226
15.3.4	KPC配置寄存器2 (KPC_CFG2)	226
15.3.5	中断状态寄存器 (KPC_STAT)	228
15.3.6	中断控制寄存器 (KPC_INT_CTRL)	229
15.3.7	KPC数据寄存器 (KPC_DATA)	230
16	直接存储器存取(DMA).....	231
16.1	DMA简介	231

16.2	DMA特性	231
16.3	DMA 外设类型	231
	存储器外设	231
	非存储器外设	231
16.4	DMA握手接口	232
16.5	DMA中断	233
16.5.1	中断类型	233
16.5.2	中断寄存器	233
16.6	DMA数据传输规则	234
16.7	DMA寄存器	237
16.7.1	地址映射表	237
16.7.2	DMAC配置寄存器 (DmaCfgReg_H/ DmaCfgReg_L)	239
16.7.3	DMAC通道使能寄存器 (ChEnReg)	240
16.7.4	通道x源地址寄存器 (SARx) (x=0)	240
16.7.5	通道x目标寄存器 (DARx) (x=0)	241
16.7.6	通道x控制寄存器 (CTLx_H/ CTLx_L) (x=0)	241
16.7.7	通道x配置寄存器 (CFGx) (x=0)	244
16.7.8	原中断状态寄存器	247
16.7.9	状态寄存器	248
16.7.10	中断屏蔽寄存器	249
16.7.11	中断状态寄存器	250
16.7.12	组合中断状态寄存器 (ChEnReg)	251
16.7.13	软握手接口寄存器	252
17	快速循环冗余校验 (FASTCRC)	254
17.1	FASTCRC简介	254
17.2	FASTCRC主要特性	254
17.3	FASTCRC功能描述	254
17.4	FASTCRC寄存器	255
17.4.1	地址映射表	255
17.4.2	CRC控制状态寄存器 (CRC_CSR)	255
17.4.3	初始值寄存器 (CRC_INI)	257
17.4.4	数据寄存器 (CRC_DATA)	258

1 芯片简介

1.1 特性

- 微控制器
 - 32位高性能RISC核心
 - 16MHz / 32MHz时钟
 - 512KB/1MB Flash
 - 40KB缓存静态RAM (SRAM)
 - 2 引脚JTAG 和JTAG 调试
 - 支持无线升级(OTA)
- 外设
 - 所有数字外设引脚均可连接任意GPIO
 - 2个UART接口, 硬件支持流控(CTS/RTS)
 - 2个同步串行接口 (SSI) (SPI、M ICROWIRE 和TI)
 - I2C
 - 四个通用定时器模块
 - 实时时钟 (RTC)
 - 正交解码器
 - AES-128 安全模块
 - 集成电压检测
 - 集成温度传感器
 - 10位模数转换器(ADC)、1.3MSPS、8通道模拟多路复用器
 - 支持16Mhz/12Mhz IO时钟输出
 - 高精度32kRC振荡器
- 射频部分
 - 2.4GHz RF 收发器, 符合Bluetooth 低功耗(BLE) 5.0 规范
 - -97dBm接收灵敏度
 - -20dBm~+7dBm 的可编程输出功率
 - 单端RF 接口
- 低功耗
 - 工作电压范围: 1.8至3.6V
 - 芯片内部DC-DC转换器
 - MCU 工作电流: 1.4 mA @ 16Mhz
 - MCU休眠电流:
 - 0.6 uA (IO唤醒)
 - 1.0uA (32Khz on、8K RAM on)
 - 1.1uA (32Khz on、24K RAM on)
 - 接收电流: 8.5mA
 - 发送电流: 9.5mA@+0dbm
16mA@+7dbm
- 休眠平均功耗
 - 1S 间隔不可连接广播: 15uA
 - 1S 间隔可连接广播: 24uA
 - 1S 间隔连接保持: 12uA
- 封装
 - 7mm×7mm QFN48封装 (32个GPIO)
 - 4mm×4mm QFN32封装 (17个GPIO)
- 工具和开发环境
 - Keil编译器
 - JLINK调试器
- 软件特性
 - 集成Host和Controller协议栈
- 典型应用
 - 鼠标键盘
 - 蓝牙遥控器
 - Mesh 智能灯
 - 防丢器
 - 可穿戴设备
 - 无线玩具
 - 医疗设备
 - 工业控制
 - Pos、蓝牙 key
 - ETC 车载单元

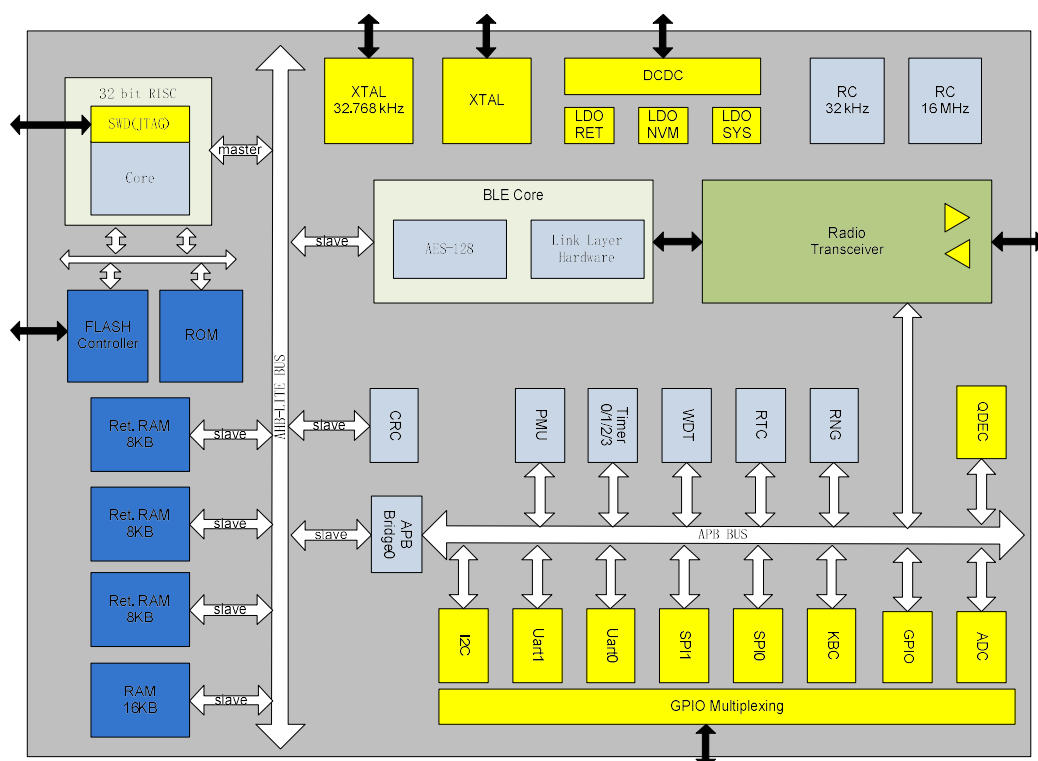
2 存储器和总线架构

2.1 系统架构

芯片系统有以下单元组成：

系统部分：

- 32位高性能RISC
- 40KBytes SRAM(8K*3 掉电保持RAM, 16K*1 掉电丢失RAM)
- 系统外设



2.2 存储器映射

存储器地址划分如下所示：

	0xFFFF_FFFF
reserved	0xE010_0000
MCU	0xE000_0000
reserved	0x6000_0000

APB0	0x5000_0000
AHB	0x4000_0000
SRAM	0x2000_0000
Flash	0x0100_0000
reserved	0x0001_4000
ROM	0x0000_0000

2.2.1 外设地址映射

下表为芯片总体地址映射表，外设地址映射见相应章节。

总线	地址范围	外设
AHB	0x4000_0000-0x4000_1FFF	BT_REG
	0x4000_2000-0x4000_2FFF	BT_EM
	0x4000_3000-0x4000_3FFF	Reserved
	0x4000_4000-0x4000_4FFF	DMA
	0x4000_5000-0x4000_5FFF	Reserved
	0x4000_6000-0x4000_6FFF	Clock and Reset Management
	0x4000_7000-0x4000_7FFF	Reserved
	0x4000_8000-0x4000_8FFF	FASTCRC
	0x4000_9000-0x4000_9FFF	Reserved
APB0	0x5000_0000-0x5000_0FFF	I2C
	0x5000_1000-0x5000_1FFF	UART0
	0x5000_2000-0x5000_2FFF	UART1
	0x5000_3000-0x5000_3FFF	SPIM/S0
	0x5000_4000-0x5000_4FFF	SPIM/S1
	0x5000_5000-0x5000_5FFF	Reserved
	0x5000_6000-0x5000_6FFF	Qdec
	0x5000_7000-0x5000_7FFF	KeyBoard
	0x5000_8000-0x5000_8FFF	SYSCFG
	0x5000_9000-0x5000_9FFF	RNG
	0x5000_A000-0x5000_AFFF	Timer0/1
	0x5000_B000-0x5000_BFFF	Timer2/3
	0x5000_C000-0x5000_CFFF	Watchdog
	0x5000_D000-0x5000_DFFF	RTC

	0x5000_E000-0x5000_EFFF	GPIO
	0x5000_F000-0x5000_FFFF	GADC
	0x5001_0000-0x5FFF_FFFF	Reserved

2.2.2 嵌入式RAM

SOC内置40KB的静态SRAM。SRAM可以以字节、半字（16位）或字（32位）访问。SRAM的起始地址：0x2000_0000。

前24K为休眠保留区，由3个8K组成，可分别控制保持区域，由SYSCTRL.SLEEP_CTRL.sram_ret_sel控制。后16K为掉电丢失RAM，休眠时数据清0。

外设	地址范围	容量
SRAM	0x2000_0000-0x2000_A000	40KB

2.2.3 位段访问

存储器映像包括两个位段(bit-band)区。这两个位段区将别名存储器区中的每个字映射到位段存储器区的一个位，在别名存储区写入一个字具有对位段区的目标位执行读-改-写操作的效果。

外设寄存器和SRAM都被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位带区的相应位的：

$$\text{bit_word_addr} = \text{bit_band_base} + (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$$

其中：

bit_word_addr是别名存储器区中字的地址，它映射到某个目标位。

bit_band_base是别名区的起始地址。

byte_offset是包含目标位的字节在位段里的序号

bit_number是目标位所在位置(0-31)

例子：

下面的例子说明如何映射别名区中SRAM地址为0x20000300的字节中的位2：

$$0x22006008 = 0x22000000 + (0x300 \times 32) + (2 \times 4).$$

对0x22006008地址的写操作与对SRAM中地址0x20000300字节的位2执行读-改-写操作有着相同的效果。

读0x22006008地址返回SRAM中地址0x20000300字节的位2的值(0x01 或 0x00)。

2.2.4 Flash特性

芯片内置Flash 主要特性:

- 512K bytes Flash 存储器。
- Page 编程单位是 256 Bytes
- Sector 擦除（4KB）为最小单位
- Block 擦除（32KB）
- 支持至少 100000 次擦除，数据至少可以保存 20 年

3 芯片特性说明

3.1 电气特性

3.1.1 极限参数

参数	说明	范围	单位
VDD	稳态电源电压	-0.3 to 3.6	V
Iddpd	关机电流	--	nA
Tamb	工作温度	-40~+105	°C
Tstg	储藏温度	-40~+150	°C
Ground	地	-0.3~0.3	V
Voh	数字输出高电平	VDD -0.3 ~ VDD+0.3	V
Vol	数字输出低电平	<0.4	V
Ioh	拉电流	15	mA
Iol	灌电流	11	mA
Vih	数字输入高电平	$\geq 0.7 \times VDD$	V
ViL	数字输入低电平	$\leq 0.3 \times VDD$	V

3.1.2 功耗

工作模式	说明	功耗	单位
RUN	● 所有外设全关		mA
	■ @16MHz	1.4	
	■ @32MHz	2.2	
	● 所有外设全开		
	■ @16MHz	1.6	
	■ @32MHz	2.2	
CPU Sleep	● 开启高速时钟，所有外设全关		mA
	■ @16MHz	1.0	
	■ @32MHz	1.2	
Sleep	● 关闭高速时钟	550	uA
Deep	关闭高速时钟、关闭外设（支持IO、RTC、KeyBoard、BLE唤醒）		uA

Sleep (常用模式)	● 内部 32K RC		
	■ 8K ram retention	1.0	
	■ 16K ram retention	1.1	
	■ 24K ram retention	1.1	
	● 外部 32K 晶振		
	■ 8K ram retention	1.2	
	■ 16K ram retention	1.3	
	■ 24K ram retention	1.3	
Deep Sleep+	● 关闭高速时钟、关闭外设、RAM 掉电（支持 IO、RTC、KeyBoard 唤醒）	0.9	uA
ShutDown	● 关断状态	50	nA

3.1.3 射频

名称	参数(Condition)	最小	典型	最大	单位
FOP	Operating Frequency	240 0		248 0	MHz
Transmitter Characteristics					
PRF	RF output power	-20	0	+7	dBm
CD	Carrier Drift Rate		5		kHz/ 50us
BW	20dB bandwidth		0. 9		MHz
Receiver Characteristics					
SEN	High Gain mode, Sensitivity @0.1%		-9 7		dBm
MaxIn	Maximum Input Power		0		dBm
C/ICO	Co-channel C/I, Basic Rate, GFSK		7		dBm
C/I1ST	ACS C/I 1MHz, Basic Rate, GFSK		5. 5	7	dBm
C/I1ST I	ACS C/I Image channel, Basic Rate, GFSK		-3 4		dBm
C/I2ND I	C/I 1 MHz adjacent to image channel, Basic Rate, GFSK		-2 8		dB

3.1.4 ESD

名称	参数	最小	典型	最大	单位
ESD					
HBM	TAMB=25℃	-	-	± 300 0	V

3.1.5 晶振

16/32M高速晶振

名称	参数	最小	典型	最大	单位
频率		16	-	32	Mhz
频率精度	@-40 to +85°C	±10		±30	ppm
温度范围		-40		+85	degree
驱动能力			100		uW
ESR	等效串联电阻	40		85	Ω
Cload	负载电容	8	15	20	pF
Cshut	寄生并联电容		5		pF

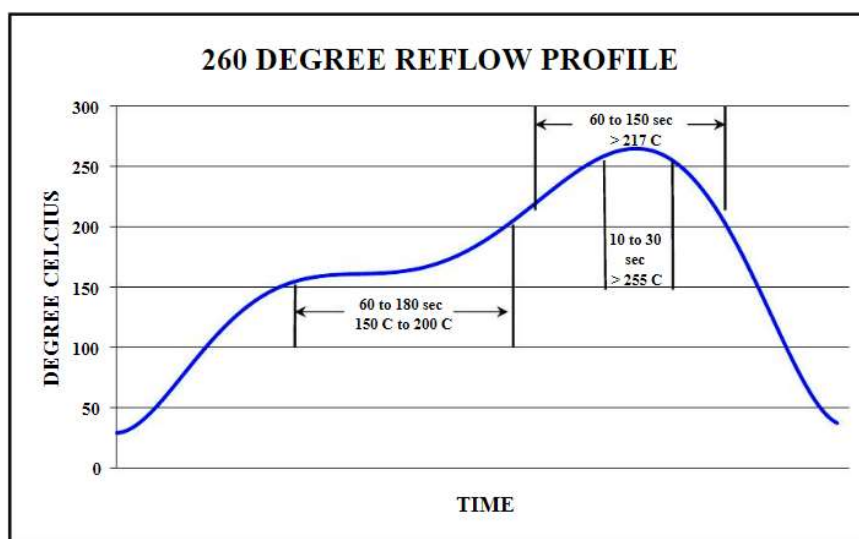
32K低速晶振

名称	参数 (Condition)	最小	典型	最大	单位
频率			32		KHz
精度		±10		±30	ppm

内部低速RC

名称	参数 (Condition)	最小	典型	最大	单位
频率			32		KHz
精度				±300	ppm

3.1.6 回流焊曲线



4 时钟与复位管理（CRM）

4.1 外设时钟管理

系统提供时钟门控控制寄存器（CG_CTRL）管理外设时钟。用户可以通过该寄存器打开和关闭对应外设时钟。外设时钟关闭后外设将不在运行，通过该寄存器可以更灵活的控制系统功耗。

4.2 软复位

系统提供软件复位操作，对软件复位寄存器（SOFT_RST）对应操作位写“1”实现相应模块复位操作，写“1”后由硬件清“0”。

4.3 寄存器描述

4.3.1 地址映射表

CRM基地址表

地址范围	基地址	外设	总线
0x4000_6000-0x4000_6FFF	0x4000_6000	CRM	AHB

CRM寄存器偏移地址表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	CG_CTRL	32	0x00000000
0x04	SOFT_RST	32	0x00000000

4.3.2 时钟门控控制寄存器1（CG_CTRL）

偏移地址：0x0000

复位值：0x00040000

31	30	29	28	27	26	25	24
预留							
							rw
23	22	21	20	19	18	17	16
预留					qspi_en	rc_en	bt_en
							rw
15	14	13	12	11	10	9	8

预留	gpa dc_en	qdec_en	gpio_en	rtc_en	wdt_en	timer 23_en	time r01_en
	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
rng_en	syscfg_en	kpc_en	spi1_en	spi0_en	uart1_en	uart0_en	i2c_en
rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	W/R	Description
31-19	rsvd	RO	
18	qspi_en	RW	QSPI 模块 HCLK 门控时钟使能： 0: 不使能； 1: 使能
17	crc_en	RW	CRC 模块 HCLK 门控时钟使能： 0: 不使能； 1: 使能
16	bt_en	RW	BT-BLE 模块门控时钟使能： 0: 不使能； 1: 使能
15	dmac_en	RW	DMAC 模块门控时钟使能： 0: 不使能； 1: 使能
14	gpadc_en	RW	GPADC 模块时钟门控使能： 0: 不使能； 1: 使能
13	qdec_en		QDEC 模块门控时钟使能： 0: 不使能； 1: 使能
12	gpio_en	RW	GPIO 模块门控时钟使能： 0: 不使能； 1: 使能
11	rtc_en	RW	RTC 模块门控时钟使能： 0: 不使能； 1: 使能
10	wdt_en	RW	Watchdog 模块门控时钟使能： 0: 不使能； 1: 使能
9	timer23_en	RW	Timer2/3 模块门控时钟使能： 0: 不使能； 1: 使能
8	timer01_en	RW	Timer0/1 模块门控时钟使能： 0: 不使能； 1: 使能

7	rng_en	RW	RNG模块门控时钟使能： 0: 不使能； 1: 使能
6	syscfg_en	RW	系统配置模块门控时钟使能： 0: 不使能； 1: 使能
5	kpc_en	RW	Keypad模块门控时钟使能： 0: 不使能； 1: 使能
4	spi1_en	RW	SPI1模块门控时钟使能： 0: 不使能； 1: 使能
3	spi0_en	RW	SPI0模块门控时钟使能： 0: 不使能； 1: 使能
2	uart1_en	RW	UART1模块门控时钟使能： 0: 不使能； 1: 使能
1	uart0_en	RW	UART0模块门控时钟使能： 0: 不使能； 1: 使能
0	i2c_en	RW	I2C模块门控时钟使能： 0: 不使能； 1: 使能

4.3.3 软件复位寄存器（SOFT_RST）

偏移地址：0x0004

复位值：0x00000000

31	30	29	28	27	26	25	24
							glb_srst
							wc
23	22	21	20	19	18	17	16
预留	预留	master_srst	crypt_srst	crc_srst	cpu_srst	crm_srst	bt_srst
ro	wc	wc	wc	wc	wc	wc	wc
15	14	13	12	11	10	9	8
dma_c_srst	gpadc_srst	qdec_srst	gpio_srst	rtc_srst	wdt_srst	timer23_srst	timer01_srst
wc	wc	wc	wc	wc	wc	wc	wc
7	6	5	4	3	2	1	0

rng_s rst	syscf g_srst	kpc_s rst	spi1_ srst	spi0_ srst	uart1 _srst	uart0_ srst	srst_i 2c
wc	wc	wc	wc	wc	wc	wc	wc

注意：对于所有的软复位信号，对应的bit写1执行软复位操作（写1后硬件自动恢复为0，不需要软件清0）。

Bit	Name	W/R	Description
31-25	rsvd[6:0]	RO	
24	glb_srst	WC	芯片数字部分全局软复位信号
23	rsvd	RO	
22	qspi_cache_ srst	RW	QSPI/Cache模块软复位信号
21	master_srst	WC	BT-BLE模块master控制逻辑软复位信号
20	crypt_srst	WC	BT-BLE模块crypt_clk控制逻辑软复位信号
19	crc_srst	WC	CRC模块软复位信号
18	cm3_srst	WC	CPU内核软复位信号
17	crm_srst	WC	CRM模块软复位信号
16	bt_srst	WC	BT-BLE/2.4G模块HCLK寄存器软复位信号
15	dmac_srst	WC	DMAC模块软复位信号
14	gpadc_srst	WC	GPADC模块软复位信号
13	qdec_srst	WC	QDEC模块软复位信号
12	gpio_srst	WC	GPIO模块软复位信号
11	rtc_srst	WC	RTC模块软复位信号
10	wdt_srst	WC	Watchdog模块软复位信号
9	timer23_srst	WC	Timer2/3模块软复位信号
8	timer01_srst	WC	Timer1/0模块软复位信号
7	rng_srst	WC	RNG模块软复位信号
6	syscfg_srst	WC	系统配置模块软复位信号
5	kpc_srst	WC	Keypad模块软复位信号
4	spi1_srst	WC	SPI1模块软复位信号
3	spi0_srst	WC	SPI0模块软复位信号
2	uart1_srst	WC	UART1软复位信号

1	uart0_srst	WC	UART0软复位信号
0	srst_i2c	WC	I2C软复位信号

5 系统控制（SYSCTRL）

5.1 时钟

芯片支持内部16MHz振荡器和外置16Mhz或32MHz晶体，内部集成的16MHz晶体的精度为 $\pm 2\%$ 。内部振荡器或外部晶振为系统时钟，系统时钟通过hclk_div_val分频后作为HCLK时钟，HCLK经过pclk_div_val分频逻辑单元后作为PCLK时钟；HCLK经过bt_div_val分频后作为BTCORE的时钟。

芯片上电复位后使用内部16M RC，在bootloader中切换成外部晶体，程序运行到用户程序的时候工作在外部晶体。

5.1.1 外部时钟源

芯片外部系统时钟要求为16MHz或32MHz，外部低速时钟要求为32.768KHz。

5.1.2 HFCLK时钟

由内部16MHz RC或外部16/32MHz晶振直接提供。

5.1.3 HCLK时钟

HFCLK通过hclk_div_val分频后作为HCLK时钟。

通过寄存器对PLL时钟进行分频：

- $HCLK = HFCLK \text{ 时钟}$
- $HCLK = HFCLK \text{ 时钟} / 2$
- $HCLK = HFCLK \text{ 时钟} / 4$

5.1.4 PCLK时钟

HCLK时钟通过时钟分频单元分频后作为PCLK时钟。

通过寄存器对PLL时钟进行分频：

- $PCLK = HCLK$
- $PCLK = HCLK / 2$
- $PCLK = HCLK / 4$

5.1.5 BTCORECLK时钟

HCLK时钟通过时钟分频单元分频后作为BTCORECLK时钟。

通过寄存器对PLL时钟进行分频：

- PCLK = HCLK
- PCLK = HCLK / 2
- PCLK = HCLK / 4

5.2 低功耗控制

系统或电源复位后，CPU处于运行状态。当CPU不需要继续运行时，可以利用多种低功耗模式来节省功耗，例如等待某个外部事件的产生。

可以通过下几种方式降低功耗：

- 关闭 AHB 和 APB 总线上不使用的外设时钟（CG_CTRL）
- 睡眠模式
- 深度睡眠模式

5.2.1 电源域划分

常开电源域：包括RTC、 KPC、 GPIO、 BLE LP Timing Generaion Unit。

外设电源域：除AON Power Domain以外的其它逻辑。

5.2.2 低功耗模式

<i>Power Domain</i>	<i>Peri</i>	<i>Ram</i>	<i>AON</i>	<i>Flash</i>
ShutDown	Off	Off	Off	Off
Deep Sleep+	Off	Off	On	Off
Deep Sleep	Off	Ret.	On	Off
Sleep(HOSC off)	On	On	On	On

在Deep Sleep模式，Ram数据Retention，CPU及外设寄存器由软件在RAM区进行备份，从Deep Sleep唤醒后恢复，PC指向休眠之前的下一条指令，然后进行CPU和外设备份寄存器的恢复。

5.2.3 低功耗模式进入

系统进入Sleep模式的方式是WFI指令，并且根据NVIC.(addr:0xE000ED10).SLEEPDEEP和SYSCFG.(addr:0x50008020).deep_slp_sel的不同配置来分别进入不同的Sleep模式。

<i>Power Domain</i>	<i>NVIC.(addr:0xE000ED10).SLEEPDEEP</i>	<i>SYSCFG.(addr:0x50008020).deep_slp_sel</i>
Deep Sleep+	1	10
Deep Sleep	1	01
Sleep	1	00
CPU_Sleep	0	xx

注意：当CRM.(addr:0x40006000).bt_en=1，即BT模块工作时，如下bit分别控制是否允许关闭OSC。

BT.(addr:0x40000030).osc_sleep_en

当CRM.(addr:0x40006000).bt_en=0，OSC始终允许被关闭

5.2.4 低功耗模式退出

<i>唤醒方式</i>	<i>SYSCFG (ADDR:0X50008020) GPIO_WK_EN</i>	<i>SYSCFG (ADDR:0X50008020) KPC_WK_EN</i>	<i>SYSCFG (ADDR:0X50008020) RTC_WK_EN</i>	<i>SYSCFG (ADDR:0X50008020) BLE_EXTWKUPDSB</i>	<i>BT (ADDR:0X40000030) DEEPSLTIME</i>
仅GPIO唤醒	1	0	0	1	0
仅KPC唤醒	0	1	0	1	0
仅RTC唤醒	0	0	1	1	0
仅BLE唤醒	0	0	0	0 (0表示唤醒)	0
全部	1	1	1	0 (0表示唤醒)	0

唤醒					
----	--	--	--	--	--

注意:

GPIO唤醒是指其高电平直接唤醒系统, KPC、RTC唤醒是指其产生的中断直接唤醒系统, 但都不唤醒BLE

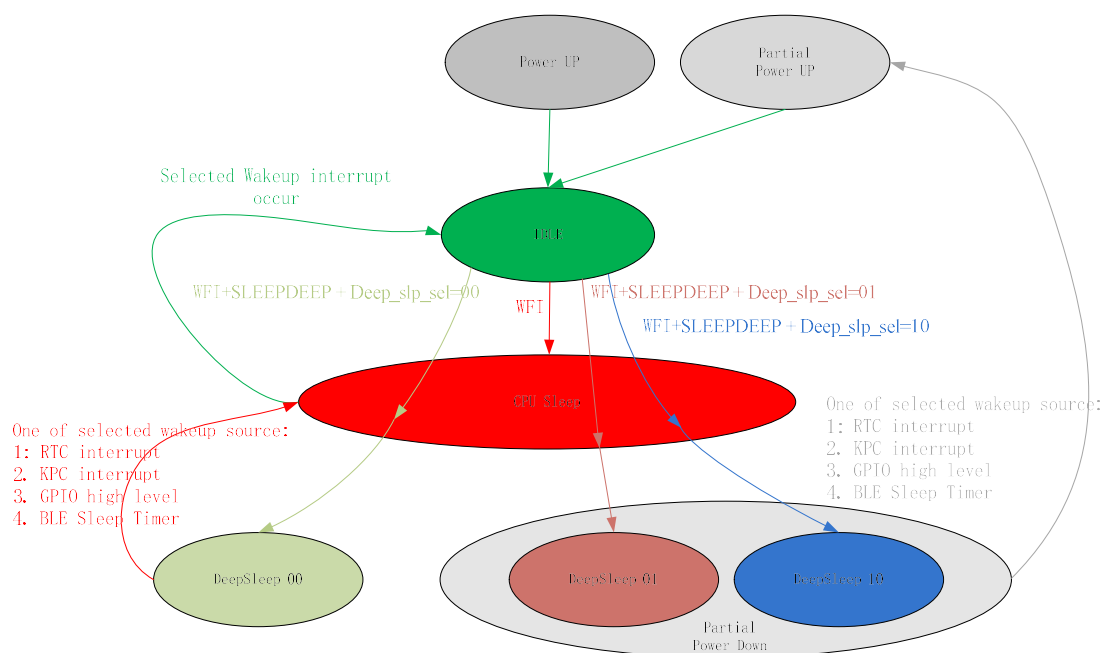
BLE唤醒是指GPIO高电平和KPC、RTC产生的中断通过BLE模块, 使产生BLE唤醒中断, 进而唤醒系统

唤醒方式可以任意组合, 多个唤醒源同时使能

两种推荐的唤醒方式:

- 当不使用BLE时, 设置CRM.(addr:0x40006000).bt_en=0, 系统可通过GPIO、KPC、RTC唤醒 (可以任意单独使能一个或几个)
- 当使用BLE时, 设置CRM.(addr:0x40006000).bt_en=1, SYSCFG(addr:0x50008020).ble_extwkupdsb=0, 系统可通过GPIO、KPC、RTC及BLE定时唤醒 (可以任意单独使能一个或几个)

5.2.5 模式切换状态图



DeepSleep模式唤醒源: RTC中断唤醒、KPC中断唤醒、GPIO高电平唤醒、BLE定时唤醒

在执行WFI指令之前必须先配置SYSCFG.SLEEP_CTRL.deep_slp_sel和

NVIC.(addr:0xE000ED10)SLEEPDEEP寄存器。

5.3 外设控制

系统控制区域包含几个外设控制寄存器, 该寄存器实现对部分外设的控制。

控制内容如下：

- TIMER_PAUSE: Timer0-1挂起使能
- SPI_SLAVE_SEL: Spi0-1主从模式选择
- RTC_DIV_VAL: RTC预分频装载寄存器
- RTC_DIV_CNT: RTC时钟分频器的当前值

5.4 寄存器描述

5.4.1 地址映射表

SYSCTRL基地址表

地址范围	基地址	外设	总线
0x5000_8000-0x5000_8FFF	0x5000_8000	SYSCTRL	APB0

SYSCTRL寄存器偏移地址表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	HFCLK_SEL	32	0x00000100
0x04	LFCLK_SEL	32	0x00000100
0x08	FREQ_SEL	32	0x00000000
0x0C	CALIB_EN	32	0x00000000
0x10	CALIB_VAL	32	0x00020820
0x14~0x1C	预留	32	0x00000000
0x20	SLEEP_CTRL	32	0x00000000
0x24	WKUP_TYPE	32	0x00000000
0x28	WKUP_SRC	32	0x00000000
0x40	WAKE_MCU_CTRL	32	0x00040000
0x50	DMA_CH_HS	32	0x00000000
0x60	TIMER_PAUSE	32	0x00000000
0x64	SPI_SLAVE_SEL	32	0x00000003
0x70	RTC_DIV_VAL	32	0x00000000
0x74	RTC_DIV_CNT	32	0x00000000
0x80~0xF0	预留	32	预留
0xF4	RESET_FLAG	32	0x00000001

5.4.2 高速时钟源选择寄存器 (HFCLK_SEL)

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	1	9	8
					0		
预留					xtal_en	rc_en	
7	6	5	4	3	2	1	0
预留					xtal_freq	clk_es	
						1	

Bit	Name	W/R	Description
31:10	rsvd	RO	预留
9	xtal_en	RW	外部高速晶体振荡器使能 0: 关闭; 1: 开启 注意: 只有当 HFCLK_SEL.rc_en=1 和 HFCLK_SEL.clk_sel=0 时, 才能关闭外部高速晶体振荡器
8	rc_en	RW	内部高速 RC 振荡器使能 0: 关闭; 1: 开启 注意: 只有当 HFCLK_SEL.xtal_en=1 和 HFCLK_SEL.clk16m_sel=1 时, 才能关闭内部高速 RC 振荡器
7:2	rsvd	RO	预留
1	xtal_freq	RW	外部晶振频率选择. This register bit has to match the actual crystal used in design to enable correct behavior. 0: 使用 16Mhz 晶振 1: 使用 32Mhz 晶振
0	clk_sel	RW	高速时钟源选择(HFCLK): 0: 选择内部 RC 振荡器时钟作为系统时钟; 1: 选择外部晶体振荡器时钟作为系统时钟;

注: 本寄存器中各个Bit主要功能为选择高速时钟源, 既内外部的16M晶振或振荡器。

5.4.3 低速时钟源选择寄存器 (LFCLK_SEL)

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留						xtal32k_e n	rc32k _en
7	6	5	4	3	2	1	0
预留							clk32 k_sel

Bit	Name	W/R	Description
31:10	rsvd	RO	预留
9	xtal32k_en	RW	外部 32.768kHz 晶体振荡器使能 0: 关闭; 1: 开启
8	rc32k_en	RW	内部 32kHz RC 振荡器使能 0: 关闭; 1: 开启
7:1	rsvd	RO	预留
0	clk32k_sel	RW	低速时钟源选择(LFCLK): 0: 选择内部 32kHz RC 振荡器时钟作为 RTC 或者休眠唤醒时钟; 1: 选择外部 32.768kHz 晶体振荡器时钟作为 RTC 或者休眠唤醒时钟;

注：本寄存器中各个Bit主要功能为选择低速时钟源，既内外部的32K晶振或振荡器。

5.4.4 时钟频率选择寄存器 (FREQ_SEL)

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

预留	bt_div_val	pclk_div_val	hclk_div_val
----	------------	--------------	--------------

Bit	Name	W/R	Description
31:6	rsvd	RO	预留
5:4	bt_div_val	RW	BT-BLE Core 时钟分频值（基于 HCLK）： 00：CLK 不对输入时钟分频； 01：CLK 对输入时钟 2 分频； 10：CLK 对输入时钟 4 分频 注：建议保持 BLE_CORE 频率为 16Mhz
3:2	pclk_div_val	RW	PCLK 分频值（基于 HCLK）： 00：PCLK 不对输入时钟分频； 01：PCLK 对输入时钟 2 分频； 10：PCLK 对输入时钟 4 分频
1:0	hclk_div_val	RW	HCLK 分频值（基于系统时钟） 00：HCLK 不对输入时钟分频； 01：HCLK 对输入时钟 2 分频； 10：HCLK 对输入时钟 4 分频

注：本寄存器中各个Bit主要功能为选择分频时钟，分别将时钟分频为各个所需的时钟频率，使用时请按照系统时钟HCLK-PCLK/BT-BLE计算。

5.4.5 时钟校准控制寄存器（CALIB_EN）

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留					caldev_en	cal16_en	cal32_k_en

Bit	Name	W/R	Description
31:3	srst_trng	RO	预留
2	caldev_en	RW	1：频偏校准使能，当校准完成时，自动清零。

			校准之前请置位 xtal_en 和 clk_sel
1	cal16_en	RW	1: 内部 16MHz RC 振荡器时钟校准使能, 当校准完成时, 自动清零。 校准之前请置位 xtal_en 和 clk_sel
0	cal32k_en	RW	1: 内部 32kHz RC 振荡器时钟校准使能, 当校准完成时, 自动清零。 校准之前请置位 xtal_en 和 clk_sel

注：本寄存器用于校准时钟使能，校准使能顺序请勿打乱。

5.4.6 时钟校准值寄存器（CALIB_VAL）

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留						caldev_val	
15	14	13	12	11	10	9	8
caldev_val				cal16_val			
7	6	5	4	3	2	1	0
cal16_val		cal32k_val					预留

Bit	Name	W/R	Description
31:18	rsvd	RO	预留
17:12	caldev_val	RW	频偏校准值
21:6	cal16_val	RW	内部 16MHz RC 振荡器时钟校准值
5:1	cal32k_val	RW	内部 32kHz RC 振荡器时钟校准值
0	rsvd	RO	预留

注：本寄存器用于存放校准值，根据需要设置校准值。

5.4.7 系统休眠模式配置寄存器（SLEEP_CTRL）

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留						sram_ret_sel	
15	14	13	12	11	10	9	8

预留				ble_e xtwk updsb	gpio_ wk_e n	kpc_ wk_e n	rtc_w k_en
7	6	5	4	3	2	1	0
预留						deep_slp_sel	

Bit	Name	W/R	Description
31:18	rsvd	RO	预留
17:16	sram_ret_sel	RW	当deep_slp_sel=01模式下，数据区SRAM Retention状态： 00: 24K SRAM Retention 01: 16K SRAM Retention 10: 8K SRAM Retention 11: reserved
15:12	rsvd	RO	预留
13	jtag_wkup_type	RW	JTAG TMS信号唤醒类型 0: 低电平唤醒 1: 高电平唤醒
12	jtag_wkup_type	RW	JTAG系统休眠唤醒使能
11	ble_extwkupdsb	RW	BLE外部唤醒禁止
10	gpio_wk_en	RW	GPIO系统休眠唤醒使能
9	kpc_wk_en	RW	Keypad中断系统休眠唤醒使能
8	rtc_wk_en	RW	RTC中断系统休眠唤醒使能
7:2	rsvd	RO	预留
1:0	deep_slp_sel	RW	当系统进入休眠时，选择休眠模式的深度： 00: 关闭高速时钟振荡器 01: 关闭高速时钟振荡器，并且关闭外设供电。 10: 关闭高速时钟振荡器，并且关闭外设和SRAM供电。

注：本寄存器存放配置系统休眠DEEP_SLEEP时的设置，包括唤醒使能以及SRAM大小。

5.4.8 系统休眠唤醒类型配置寄存器（WKUP_TYPE）

31	30	29	28	27	26	25	24
预留						gpio_wkup3_type	

23	22	21	20	19	18	17	16
预留						gpio_wkup2_type	
15	14	13	12	11	10	9	8
预留						gpio_wkup1_type	
7	6	5	4	3	2	1	0
预留						gpio_wkup0_type	

Bit	Name	W/R	Description
31:2 6	rsvd	RO	预留
25:2 4	gpio_wkup3_type	RW	GPIO系统休眠唤醒源3类型及使能控制位： 00：不唤醒；01：直接唤醒； 10：过滤1个32K时钟周期毛刺后唤醒；11：保留
23:1 8	rsvd	RO	预留
17:1 6	gpio_wkup2_type	RW	GPIO系统休眠唤醒源2类型及使能控制位： 00：不唤醒；01：直接唤醒； 10：过滤1个32K时钟周期毛刺后唤醒；11：保留
15:1 0	rsvd	RO	预留
9:8	gpio_wkup1_type	RW	GPIO系统休眠唤醒源1类型及使能控制位： 00：不唤醒；01：直接唤醒； 10：过滤1个32K时钟周期毛刺后唤醒；11：保留
7:2	rsvd	RO	预留
1:0	gpio_wkup0_type	RW	GPIO系统休眠唤醒源0类型及使能控制位： 00：不唤醒；01：直接唤醒； 10：过滤1个32K时钟周期毛刺后唤醒；11：保留

注：本寄存器选择使用GPIO系统唤醒时选择的中断触发时的反应。

5.4.9 系统休眠唤醒源配置寄存器 (WKUP_SRC)

31	30	29	28	27	26	25	24
预留			gpio_wkup3_src				
23	22	21	20	19	18	17	16
预留			gpio_wkup2_src				
15	14	13	12	11	10	9	8
预留			gpio_wkup1_src				
7	6	5	4	3	2	1	0
预留			gpio_wkup0_src				

Bit	Name	W / R	Description
31-2 9	rsvd	R O	预留
28-2 4	gpio_wkup3_src	R W	GPIO系统休眠唤醒源3来源选择寄存器: 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
23-2 1	rsvd	R O	预留
20-1 6	gpio_wkup2_src	R W	GPIO系统休眠唤醒源2来源选择寄存器: 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
15-1 3	rsvd	R O	预留
12-8	gpio_wkup1_src	R W	GPIO系统休眠唤醒源1来源选择寄存器: 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
7-5	rsvd	R O	预留

4-0	gpio_wkup0_src	R W	GPIO系统休眠唤醒源0来源选择寄存器： 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
-----	----------------	--------	--

注：本寄存器选择使用GPIO系统唤醒时选择的中断触发的引脚。

5.4.10 MCU休眠唤醒配置寄存器（WAKE_MCU_CTRL）

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							qdec_wk_mcu
15	14	13	12	11	10	9	8
bt_wk_mcu	gpio_int3_wk_mcu	gpio_int2_wk_mcu	gpio_int1_wk_mcu	gpio_int0_wk_mcu	rtc_wk_mcu	timer3_wk_mcu	timer2_wk_mcu
7	6	5	4	3	2	1	0
timer1_wk_mcu	timer0_wk_mcu	kpc_wk_mcu	spi1_mk_mcu	spi0_wk_mcu	uart1_wk_mcu	uart0_wk_mcu	i2c_wk_mcu
Bit	Name		W/R	Description			
31:17	rsvd		RO	预留			
18	all_wk_en		RW	所有 Exception, External Interrupt 都能唤醒 MCU			
17	dmac_wk_mcu		RW	DMAC 中断唤醒 MCU 使能			
16	qdec_wk_mcu		RW	QDEC中断唤醒MCU使能			
15	bt_wk_mcu		RW	BT-BLE中断唤醒MCU使能			
14	gpio_int3_wk_mc		R	GPIO中断3唤醒MCU使能			

	u	W	
13	gpio_int2_wk_mcu	R	GPIO中断2唤醒MCU使能
	u	W	
12	gpio_int1_wk_mcu	R	GPIO中断1唤醒MCU使能
	u	W	
11	gpio_int0_wk_mcu	R	GPIO中断0唤醒MCU使能
	u	W	
10	rtc_wk_mcu	R	RTC中断唤醒MCU使能
		W	
9	timer3_wk_mcu	R	Timer3中断唤醒MCU使能
		W	
8	timer2_wk_mcu	R	Timer2中断唤醒MCU使能
		W	
7	timer1_wk_mcu	R	Timer1中断唤醒MCU使能
		W	
6	timer0_wk_mcu	R	Timer0中断唤醒MCU使能
		W	
5	kpc_wk_mcu	R	Keypad中断唤醒MCU使能
		W	
4	spi1_mk_mcu	R	SPI1中断唤醒MCU使能
		W	
3	spi0_wk_mcu	R	SPI0中断唤醒MCU使能
		W	
2	uart1_wk_mcu	R	UART1中断唤醒MCU使能
		W	
1	uart0_wk_mcu	R	UART0中断唤醒MCU使能
		W	
0	i2c_wk_mcu	R	I2C中断唤醒MCU使能
		W	

注：本寄存器使能MCU处于SLEEP模式时的唤醒使能。

5.4.11 DMA硬件握手外设分配寄存器（DMA_CH_HS）

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

预留							
15	14	13	12	11	10	9	8
dma_ch3_hs				dma_ch2_hs			
7	6	5	4	3	2	1	0
dma_ch1_hs				dma_ch0_hs			
Bit	Name	W/ R	Description				
31:16	rsvd	RO	预留				
15:12	dma_ch3_hs	R W	DMA Channel3 硬件握手外设分配: 0: Uart0 TX 1: Uart0 RX 2: Uart1 TX 3: Uart1 RX 4: SPI0 TX 5: SPI0 RX 6: SPI1 TX 7: SPI1 RX 8: I2C TX 9: I2C RX 10: GPADC_RX 11: QSPI				
11:8	dma_ch2_hs	R W	DMA Channel2 硬件握手外设分配: 0: Uart0 TX 1: Uart0 RX 2: Uart1 TX 3: Uart1 RX 4: SPI0 TX 5: SPI0 RX 6: SPI1 TX 7: SPI1 RX 8: I2C TX 9: I2C RX 10: GPADC_RX 11: QSPI				

7:4	dma_ch1_hs	R W	DMA Channel1 硬件握手外设分配: 0: Uart0 TX 1: Uart0 RX 2: Uart1 TX 3: Uart1 RX 4: SPI0 TX 5: SPI0 RX 6: SPI1 TX 7: SPI1 RX 8: I2C TX 9: I2C RX 10: GPADC_RX 11: QSPI
3:0	dma_ch0_hs	R W	DMA Channel0 硬件握手外设分配: 0: Uart0 TX 1: Uart0 RX 2: Uart1 TX 3: Uart1 RX 4: SPI0 TX 5: SPI0 RX 6: SPI1 TX 7: SPI1 RX 8: I2C TX 9: I2C RX 10: GPADC_RX 11: QSPI

5.4.12 Timer挂起控制寄存器 (TIMER_PAUSE)

31:2	1	0
预留	timer1_pa use	timer0_pa use

Bit	Name	W/R	Description
-----	------	-----	-------------

31:2	rsvd	RO	预留
1	timer1_pause	RW	Timer1挂起使能： 0：不挂起； 1：挂起
0	timer0_pause	RW	Timer0挂起使能： 0：不挂起； 1：挂起

注：本寄存器控制定时器的挂起，不挂起可恢复。

5.4.13 SPI工作模式配置寄存器（SPI_SLAVE_SEL）

31:2	1	0
预留	spi1_slv_sel	spi0_slv_sel

Bit	Name	W/R	Description
31:2	rsvd	RO	预留
1	spi1_slv_sel	RW	SPI1工作模式选择： 0: master模式； 1: slave模式
0	spi0_slv_sel	RW	SPI0工作模式选择： 0: master模式； 1: slave模式

注：本寄存器控制SPI模式下主从机的设置。

5.4.14 RTC预分频装载寄存器（RTC_DIV_VAL）

31:20	19:0
预留	rtc_div_cnt

Bit	Name	W/R	Description
31:20	rsvd	RO	预留
19:0	rtc_div_cnt	RW	根据以下公式，来定义计数器的时钟频率： $F(\text{rtc_clk}) = F(\text{LFCLK}) / (\text{rtc_div_val}[19:0] + 1)$ 注：如果输入时钟频率是32.768kHz（LFCLK）， 这个寄存器中写入7FFFh可获得周期为1秒钟的信号。

			注：推荐配置rtc_div_cnt=0x01，通过16.384k的频率作为RTC的计数周期，否则RTC容易产生不能唤醒问题
--	--	--	---

注：本寄存器控制RTC外设的时钟输入分频。

5.4.15 RTC预分频器当前值寄存器（RTC_DIV_CNT）

31:20	19:0
预留	rtc_div_cnt

Bit	Name	W/R	Description
31:20	rsvd	RO	预留
19:0	rtc_div_cnt	RO	RTC时钟分频器的当前值

注：本寄存器保存当前RTC外设的时钟计数器的值。

6 通用输入输出（GPIO）

6.1 GPIO功能描述

芯片共分四个GPIO组（A-D），每组GPIO为[0:7]编号8个引脚共32个GPIO，任意引脚均可复用任意外设功能。

芯片每组GPIO包含5个32位寄存器：

- 数据寄存器（Px_IODR）
- 置位/复位寄存器（Px_BSRR）
- 控制寄存器（Px_CON）
- 复用选择寄存器（Px_H_MODE）
- 复用选择寄存器（Px_L_MODE）

Px_IODR[15~8]位作为输入寄存器（Px_IDR）（只读），[7~0]位作为输出寄存器（Px_ODR）（读写）

Px_BSRR高8位作为reset寄存器，低8位作为set寄存器

GPIO端口的每个引脚可以配置为多种工作方式：

- 输入模式（输入高阻、输入下拉）
- 推挽输出(没有开漏输出模式)

GPIO工作模式配置图：

工作模式	Px_IODR	Px_BSRR	Px_CON	Px_H/L_MODE
输入浮空	data_r	不使用	00	0
输入下拉	data_r	不使用	10	0
推挽输出	data_rw	data_w	11	0

注：

data_r：数据读操作

data_w：数据写操作

data_rw：数据读写操作

6.1.1 通用I/O（GPIO）

作为输出配置时，写到输出数据寄存器上的值将输出到对于I/O上。输入数据寄存器显示APB上捕捉I/O上的数据。

所有GPIO引脚上都有一个下拉电阻，可以通过下拉使能寄存器（Px_CON）控制是否有效。

6.1.2 单独的位设置或清除

通过置位/复位寄存器（Px_BSRR）可以实现对单独I/O的置位/复位操作。

6.1.3 外部中断

通过配置 GPIO 使其对引脚上状态变化对 CPU 产生中断请求。

GPIO 外部中断响应类型：

- 不产生中断
- 上升沿中断
- 下降沿中断
- 双边沿中断

6.1.4 外部唤醒事件

芯片所有GPIO管脚均支持超低功耗唤醒，通过gpio_wk_en开启GPIO系统休眠唤醒使能，通过gpio_intx_wk_mcu选择外部中断X（0~3）作为唤醒源。通过gpio_wkupx_src选择GPIO管脚。

注：芯片最多同时支持4个IO配置为外部唤醒管脚，如果希望更多的IO可唤醒芯片请使用Keyboard模式。

6.1.5 I/O功能复用

外设与I/O复用可通过复用控制寄存器（Px_H/L_MODE）进行配置。

- 根据需要进行 I/O 复用
- 复用为功能外设后，无需配置 I/O 工作模式（输入高阻、输入下拉、开漏输出、推挽输出），复用配置完成后系统会自动进入对应 I/O 工作模式。

管脚复用映射表（MODE设定对应GPIO功能）

GPIO	功能描述
Px7_mode	GPIO的功能复用选择： 0:GPIO 1:UART0_RX 2:UART0_TX 3:UART0_CTSN

	4:UART0_RTSN
	5:UART1_RX
	6:UART1_TX
	7:UART1_CTSN
	8: UART1_RTSN
	9: SPI0_CSN
	10: SPI0_CLK
	11: SPI0_MOSI
	12: SPI0_MISO
	13: SPI1_CSN
	14: SPI1_CLK
	15: SPI1_MOSI
	16: SPI1_MISO
	17: I2C_SCL
	18: I2C_SDA
	19: PWM0
	20: PWM1
	21: QDEC_CHA
	22: QDEC_CHB
	23: QDEC_IDX
	26-24: Reserved
	27: Reserved
	28: JTAG_TDO
	29: JTAG_TDI
	30: J Reserved
	31: JTAG_TRST
	32: KPC_PIO[0]
	33: KPC_PIO[1]
	34: KPC_PIO[2]
	35: KPC_PIO[3]
	36: KPC_PIO[4]
	37: KPC_PIO[5]
	38: KPC_PIO[6]
	39: KPC_PIO[7]
	40: KPC_PIO[8]
	41: KPC_PIO[9]
	42: KPC_PIO[10]
	43: KPC_PIO[11]

	44: KPC_PIO[12] 45: KPC_PIO[13] 46: KPC_PIO[14] 47: KPC_PIO[15] 48: ADC (only for PA[7:0]) 50-49: Reserved 51: 12M CLOCK 53-52: Reserved 54: 16M/32M OSC/16M RC CLOCK 55-63: Reserved
Px6_mode	同Px7_mode
Px5_mode	同Px7_mode
Px4_mode	同Px7_mode
Px3_mode	同Px7_mode
Px2_mode	同Px7_mode
Px1_mode	同Px7_mode
Px0_mode	同Px7_mode

注:

- x 代表 A/B/C/D
- 7~0 代表 PA/PB/PC/PD 组的第 n 个 IO 管脚

6.2 GPIO寄存器

6.2.1 地址映射表

GPIO基地址表

地址范围	基地址	外设	总线
0x5000_E000-0x5000_EFFF	0x5000_E000	GPIO	APB0

GPIO寄存器偏移地址表

偏移地址	寄存器名称	宽度 (bit)	复位值	注释
0x00	PA_IOD	32	0x00000000	GPIOA
0x04	PA_BSR	32	0x00000000	
0x08	PA_CON	32	0x0000AAAA	
0x10	PB_IOD	32	0x00000000	GPIOB
0x14	PB_BSR	32	0x00000000	
0x18	PB_CON	32	0x0000AAAA	

0x20	PC_IOD	32	0x00000000	GPIOC
0x24	PC_BSR	32	0x00000000	
0x28	PC_CON	32	0x0000AAAA	
0x30	PD_IOD	32	0x00000000	GPIOD
0x34	PD_BSR	32	0x00000000	
0x38	PD_CON	32	0x0000AAAA	
0x40	PA_L_MODE	32	0x00000000	PX_MODE
0x44	PA_H_MODE	32	0x00000000	
0x48	PB_L_MODE	32	0x00000000	
0x4C	PB_H_MODE	32	0x00000000	
0x50	PC_L_MODE	32	0x001F1D1C	
0x54	PC_H_MODE	32	0x00000000	
0x58	PD_L_MODE	32	0x00000000	
0x5C	PD_H_MODE	32	0x00000000	
0x80	INTP_TYPE	32	0x00000000	INTP
0x84	INTP_SRC	32	0x00000000	
0x88	INTP_STAT	32	0x00000000	

6.2.2 数据寄存器（Px_IOD）（x=A..D）

31:16	15:8	7:0
预留	px_datin	px_datout

Bit	Name	W/R	Description
31:16	rsvd	RO	预留
15:8	px_datin	RO	GPIO A-D 输入数据寄存器
7:0	px_datout	RW	GPIO A-D 输出数据寄存器

注：该寄存器低八位以及高八位分别对应输出输入的引脚电平状态，[15:8]为输入只读，可通过[7:0]设置相对应引脚输出电平。

6.2.3 置位/复位寄存器（Px_BSR）（x=A..D）

31:16	15:8	7:0
预留	px_bit_clr	px_bit_set

Bit	Name	W/ R	Description
31:16	rsvd	RO	预留
15:8	px_bit_clr		GPIO A-D 对应 bit 复位寄存器，当向对应的 bit 写 1 时，pa_datout 对应的 bit 位被清零
7:0	px_bit_set	WO	GPIO A-D 对应 bit 置位寄存器，当向对应的 bit 写 1 时，Px_ODR 对应的 bit 位被置 1

注：相对应的Px_IOD的低八位输出控制，分别为置位和复位。

6.2.4 控制寄存器（Px_CON）（x=A..D）

31:16	15:0
预留	px_con

Bit	Name	W/ R	Description
31:16	rsvd	RO	预留
15:0	px_con	RW	GPIO A-D 对应 bit 输入/输出使能，下拉电阻使能控制寄存器： 00：输入，不选择下拉电阻； 10：输入，选择下拉电阻； 11：输出。 注意，作为 ADC 输入口时，不受该控制位控制

注：该寄存器是对整组GPIO的控制，既2Bit控制一个引脚的输入输出属性。

※设置为输入时，请勿悬空引脚，建议添加下拉电阻，避免电源不稳引起引脚电平波动。

6.2.5 GPIO A-D [3:0] 模式寄存器（Px_L_MODE）（x=A..D）

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留	px3_mod					预留					px2_mod				
1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0										
预留	px1_mod					预留					px0_mod				

Bit	Name	W/ R	Description
31:3 0	rsvd	R O	预留
29:2 4	px3_mod	R W	详见 6.1.5 I/O 功能复用
23:2 2	rsvd	R O	预留
21:1 6	px2_mod	R W	详见 6.1.5 I/O 功能复用
15:1 4	rsvd	R O	预留
13:8	px1_mod	R W	详见 6.1.5 I/O 功能复用
7:6	rsvd	R O	预留
5:0	px0_mod	R W	详见 6.1.5 I/O 功能复用

注：该寄存器是针对GPIO部分低四位引脚初始化复用的选项。

6.2.6 GPIO A-D [7:4] 模式寄存器 (Px_H_MODE) (x=A..D)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留	px7_mode					预留	px6_mode								
1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0										
预留	px5_mode					预留	px4_mode								

Bit	Name	W/ R	Description
31:3 0	rsvd	R O	预留
29:2 4	px7_mod	R W	详见 6.1.5 I/O 功能复用
23:2	rsvd	R	预留

2		O	
21:1 6	px6_mod	R W	详见 6.1.5 I/O 功能复用
15:1 4	rsvd	R O	预留
13:8	px5_mod	R W	详见 6.1.5 I/O 功能复用
7:6	rsvd	R O	预留
5:0	px4_mod	R W	详见 6.1.5 I/O 功能复用

注：该寄存器是对于GPIO部分高四位引脚初始化复用的选项。

6.2.7 中断类型配置寄存器 (INTP_TYPE)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6
预留						int3_t ype		预留						int2_t ype	
1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
预留						int1_t ype		预留						int0_t ype	

Bit	Name	W/ R	Description
31:2 6	rsvd	R O	预留
25:2 4	int3_type	R W	中断 3 类型及使能控制位： 00：不产生中断；01：上升沿中断； 10：下降沿中断；11：双沿中断
23:1 8	rsvd	R O	预留
17:1 6	int2_type	R W	中断 2 类型及使能控制位： 00：不产生中断；01：上升沿中断； 10：下降沿中断；11：双沿中断
15:1	rsvd	R	预留

0		O	
9:8	int1_type	R W	中断 1 类型及使能控制位： 00 : 不产生中断；01 : 上升沿中断； 10 : 下降沿中断；11 : 双沿中断
7:2	rsvd	R O	预留
1:0	int0_type	R W	中断 0 类型及使能控制位： 00 : 不产生中断；01 : 上升沿中断； 10 : 下降沿中断；11 : 双沿中断

注：该寄存器用于选择触发中断的方式。

6.2.8 中断源配置寄存器 (INTP_SRC)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留		int3_src				预留				int2_src					
1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0										
预留		int1_src				预留				int0_src					

Bit	Name	W/R	Description
31:29	rsvd	RO	预留
28:24	int3_src	RW	中断 3 来源选择寄存器： 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
23:21	rsvd	RO	预留
20:16	int2_src	RW	中断 2 来源选择寄存器： 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
15:13	rsvd	RO	预留
12:8	int1_src	RW	中断 1 来源选择寄存器： 5'h1f-5'h18 : PD[7]-PD[0]

			5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]
7:5	rsvd	RO	预留
4:0	int0_src	RW	中断 0 来源选择寄存器: 5'h1f-5'h18 : PD[7]-PD[0] 5'h17-5'h10 : PC[7]-PC[0] 5'h0f-5'h08 : PB[7]-PB[0] 5'h07-5'h00 : PA[7]-PA[0]

注：该寄存器用于选择触发中断的引脚，每5个位控制一路GPIO中断的选择。

0x1f-0x00对于4组32个引脚。

6.2.9 中断状态寄存器 (INTP_STA)

3 3 2 2 2 2 2 24 2 2 2 2 1 1 1 16
 1 0 9 8 7 6 5 3 2 1 0 9 8 7

保留	int 3_s tat	保留	int2_ stat
1 1 1 1 1 1 9 8 7 6 5 4 3 2 1 0			
5 4 3 2 1 0			

保留	int 1_s tat	保留	int0_ stat
----	-------------------	----	---------------

Bit	Name	W/ R	Description
31:1 6	rsvd	R O	预留
24	int3_stat	W C	中断 3 状态寄存器,当向该位写 1 时清除中断 3 状态: 0 : 未产生中断; 1: 产生中断
23:1 7	rsvd	R O	预留
16	int2_stat	W	中断 2 状态寄存器,当向该位写 1 时清除中断

		C	3 状态： 0 : 未产生中断； 1: 产生中断
15:9	rsvd	R O	预留
8	int1_stat	W C	中断 1 状态寄存器, 当向该位写 1 时清除中断 3 状态： 0 : 未产生中断； 1: 产生中断
7:1	rsvd	R O	预留
0	int0_stat	W C	中断 0 状态寄存器, 当向该位写 1 时清除中断 3 状态： 0 : 未产生中断； 1: 产生中断

注：该寄存器保存当前的中断状态，也可由该寄存器写入控制中断清除，既兼替了中断清除寄存器的功能。

实时时钟（RTC）

7 RTC简介

实时时钟是一个独立的定时器。RTC模块拥有一组连续计数的计数器。RTC模块和RTC相关配置寄存器都处于AON电源域。

注：必须等待RTC内部计数器(RTC->CCVR)值大于2时才能配置芯片进入休眠，否则可能出现RTC无法从休眠状态唤醒的问题。所以SYSCFG->RTC_DIV_VAL越小越好，推荐0x01。

7.1 RTC操作说明

RTC预分频装载寄存器RTC_DIV_VAL（推荐设1），通过以下公式来定义计数器的时钟频率：

$$F(\text{rtc_clk}) = F(\text{LFCLK}) / (\text{RTC_DIV_VAL} + 1)$$

如果输入时钟频率是32.768kHz（LFCLK），这个寄存器写入0x01可获得周期(1+1)/32K秒钟的信号。

操作示例：

```
SYSCFG->RTC_DIV_VAL = 0x1
```

```
RTC->RTC_CMR = 16384; //1s产生一次中断 //
```

RTC->RTC_CLR = 0x0; //初始值寄存器，CCVR在此CLR值的基础上递增，可以用于设置当前时间的秒数

```
RTC->RTC_CCR = 0x05;
```

```
time = RTC->RTC_CCVR; //读取RTC的计数秒值
```

7.2 RTC寄存器

7.2.1 地址映射表

RTC基地址表

地址范围	基地址	外设	总线
0x5000_D000 - 0x5000_DFFF	0x5000_D000	RTC	APB0

RTC寄存器偏移地址表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	RTC_CCVR	32	0x00000000
0x04	RTC_CMR	32	0x00000000
0x08	RTC_CLR	32	0x00000000
0x0C	RTC_CCR	32	0x00000000
0x10	RTC_STAT	32	0x00000000
0x14	RTC_RSTAT	32	0x00000000
0x18	RTC_EOI	32	0x00000000

7.2.2 RTC当前计数寄存器 (RTC_CCVR)

31:0

rtc_cmr

Bit	Name	W/ R	Description
31:0	rtc_cmr	R O	When read, this register is the current value of the internal counter. This value always is read coherently. Bits from RTC_CNT_WIDTH to 31 are read as 0 when RTC_CNT_WIDTH is less than 31.

7.2.3 RTC计数匹配寄存器 (RTC_CMR)

31:0

rtc_cmr

Bit	Name	W/ R	Description
31:0	rtc_cmr	R W	Interrupt Match Register. When the internal counter matches this register, an interrupt is generated, provided interrupt generation is enabled. When appropriate, this value is written coherently. Only when all the bytes are written is the register used by the interrupt detection logic. Bits from RTC_CNT_WIDTH

			and above are read and written as 0 when RTC_CNT_WIDTH is less than 31.
--	--	--	--

7.2.4 RTC计数器装载寄存器 (RTC_CLR)

31:0

rtc_clr

Bit	Name	W/ R	Description
31:0	rtc_clr	R W	Loaded into the counter as the loaded value, which is written coherently. Bits from RTC_CNT_WIDTH and above are read and written as 0 when RTC_CNT_WIDTH is less than 31.

7.2.5 RTC计数器控制寄存器 (RTC_CCR)

31:4

3

2

1

0

预留	rtc_ we n	rtc_ en	rtc_ mas k	rtc_ _ie n
----	-----------------	------------	------------------	------------------

Bit	Name	W/ R	Description
31:4	rsvd	R O	预留
3	rtc_wen	R W	<i>Optional.</i> Allows the user to force the counter to wrap when a match occurs instead of waiting until the maximum count is reached. 0 = Wrap disabled 1 = Wrap enabled This bit is writable only when RTC_WRAP_MODE = 1.
2	rtc_en	R W	<i>Optional.</i> Allows the user to control counting in the counter.

			0 = Counter disabled 1 = Counter enabled This bit does not exist if RTC_EN_MODE = 0. Internally, the counter always is enabled.
1	rtc_mask	R W	Allows the user to mask a generated interrupt. 0 = Interrupt unmasked 1 = Interrupt masked
0	rtc_ien	R W	Allows the user to disable interrupt generation. 0 = Interrupt disabled 1 = Interrupt enabled

7.2.6 RTC中断状态寄存器 (RTC_STAT)

31:1	0
预留	rtc_stat

Bit	Name	W/ R	Description
31:1	rsvd	R O	预留
0	rtc_stat	R O	This register is the masked raw status 0 = Interrupt is inactive 1 = Interrupt is active (regardless of polarity)

7.2.7 RTC中断原始状态寄存器 (RTC_RSTAT)

31:1	0
预留	rtc_rstat

Bit	Name	W/ R	Description
31:1	rsvd	R O	预留
0	rtc_rstat	R O	0 = Interrupt is inactive 1 = Interrupt is active (regardless of polarity)

7.2.8 RTC中断结束寄存器 (RTC_EOI)

31:1	0
预留	rtc_eoi

Bit	Name	W/ R	Description
31:1	rsvd	R O	预留
0	osc32k_cal_word	R O	By reading this location, the match interrupt is cleared. Performing read-to-clear on interrupts, the interrupt is cleared at the end of the read.

8 看门狗 (WDT)

8.1 看门狗外设时钟

看门狗外设时钟由PCLK提供，即看门狗外设时钟频率等于PCLK时钟频率。

8.2 计数器 (Counter)

看门狗计数器 (DWT_CCVR: Watchdog Timer Current Counter Value Register) 为递减计数器，即计数器值由预设值递减直至数值为0。当计数器计数到0时，看门狗根据设定模式产生系统复位或中断。

工作中断模式下的看门狗，在看门狗计数器第1次计数到0时，会产生看门狗中断，并重置看门狗计数器到预设值，但不会产生系统复位。此后看门狗计数器会进入下一轮递减计数，用户必须在此次计数过程中进行喂狗或清中断处理操作，否则在此次计数至0后，系统发生复位。

用户可以在发生复位前任意时刻，通过对计数器重置寄存器 (WDT_CRR: Counter Restart Register) 写0x76，来重置计数器为预设值。完成“喂狗”操作。

8.3 计数器预设值 (Timeout Period Values)

看门狗计数器预设值由WDT_RLD (Watchdog Timer Reload Value Register) 保存，用户可以通过该寄存器设定看门狗计数器超时时间。对WDT_CRR寄存器写0x76将对DWT_CCVR寄存器的置重置为此预设值，完成“喂狗”操作。

8.4 启用看门狗 (WatchDog Enable)

看门狗开启由看门狗控制寄存器 (WDT_CR: Watchdog Timer Control Register) 控制，当WDT_CR[0] = 1时看门狗开启。看门狗使能开启后将无法关闭，可通过复位看门狗模块来关闭看门狗。

8.5 系统复位/中断 (System Resets)

看门狗包含2中模式：

WDT_CR[1] = 0: 系统复位模式

WDT_CR[1] = 1: 中断模式

系统复位模式：

看门狗计数器计数到0后，系统立即产生复位。

中断模式：

在看门狗计数器第1次计数到0时，会产生看门狗中断（中断源为不可屏蔽中断NMI），并重置看门狗计数器到预设值，但不会产生系统复位。此后看门狗计数器会进入下1轮递减计数，用户必须在此次计数过程中进行喂狗或清中断处理操作，否则在此次计数至0后，系统发生复位。

运行在中断模式中的看门狗，除了使用普通喂狗方式（重置看门狗计数器）外，还可以通过清除看门狗中断标记完成喂狗。

看门狗中断可以通过以下两种方式清除：

1、重置看门狗计数器（喂狗）

对WDT_CRR寄存器写0x76后，硬件自动将WDT_RLD寄存器的值加载到DWT_CCVR寄存器，完成“喂狗”操作。

2、读看门狗中断清除寄存器（WDT_EOI）

对WDT_EOI寄存器进行读操作清除看门狗中断标记。

8.6 寄存器描述

8.6.1 地址映射表

WDT基地址列表

地址范围	基地址	外设	总线
0x5000_C000-0x5000_CFFF	0x5000_C000	WDT	APB0

WDT寄存器偏移地址列表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	WDT_CR	32	0x00000000
0x04	预留	32	0x00000000
0x08	WDT_CCVR	32	0x0000FFFF
0x0C	WDT_CCR	32	0x00000000
0x10	WDT_STAT	32	0x00000000
0x14	WDT_EOI	32	0x00000000
0x18	预留	32	0x00000000
0x1C	WDT_RLD	32	0x00000000

8.6.2 看门狗控制寄存器 (WDT_CR)

- **Name:** Control Register
- **Size:** 32bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

31:2	1	0
预留	RM OD	WD T_E N

Bit	Name	RW	Description
31:2	预留	-	读操作返回 0
1	RMOD	RW	Response mode 选择看门狗超时应答模式 0 = 产生系统复位 1 = 第 1 次看门狗超时产生系统中断, 第 2 次看门狗超时产生前未清除中断, 则发生系统复位。 复位值: 0x00
0	WDT_EN	RW	WDT enable 看门狗使能操作位。看门狗使能开启后将无法关闭, 系统复位不影响看门狗使能。 0 = 看门狗关闭 1 = 看门狗开启 复位值: 0x00

8.6.3 看门狗计数器 (WDT_CCVR)

- **Name:** Current Counter Value Register
- **Size:** 32bits
- **Address Offset:** 0x08
- **Read/write access:** read

31:0

WDT_CCVR

Bit	Name	RW	Description
31:0	WDT_CC VR	R	对该寄存器读操作，读取的值为读取时刻对应的计数值。 复位值：0xFFFF

8.6.4 看门狗计数器重置寄存器（WDT_CRR）

- **Name:** Counter Restart Register
- **Size:** 32 bits
- **Address Offset:** 0x0c
- **Read/write access:** write

31:8	7:0
预留	WDT_CRR

Bit	Name	RW	Description
31:8	预留	-	读操作返回 0
7:0	WDT_CR R	W	该寄存器用于重置看门狗计数器值，为防止意外操作，写入值必须是0x76。对该寄存器读返回值为0 复位值：0x00

8.6.5 看门狗中断状态寄存器（WDT_STAT）

- **Name:** Interrupt Status Register
- **Size:** 32 bit
- **Address Offset:** 0x10
- **Read/write access:** read

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留															S

	T A T
--	-------------

Bit	Name	RW	Description
31:1	预留	-	读操作返回 0
0	STAT	R	Interrupt Status 该位用于表示看门狗的中断状态 1 = 看门狗中断产生 0 = 看门狗中断未产生 复位值: 0x00

8.6.6 看门狗中断清除寄存器 (WDT_EOI)

■ **Name:** Interrupt Clear Register

■ **Size:** 32 bit

■ **Address Offset:** 0x14

■ **Read/write access:** read

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留															E O I
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------------

Bit	Name	RW	Description
31:1	预留	-	读操作返回 0
0	EOI	R	Clears the watchdog interrupt 清除看门狗中断,并重置看门狗计数器(WDT_CCVR)。 复位值: 0x00

8.6.7 看门狗匹配值寄存器 (WDT_RLD)

- **Name:** RestartLoad Register
- **Size:** 32 bits
- **Address Offset:** 0x0c
- **Read/write access:** write

31:0

WDT_RLD

Bit	Name	RW	Description
31:0	WDT_RLD	RW	写入看门狗匹配值，看门狗计数器达到该值时启动复位。

9 定时器 (TIMER)

9.1 定时器简介

- 2 个 Timer 单元，每个单元包含 2 个独立定时器 (Timer0, Timer1)，共 4 个定时器。
- 4 个定时器中断源独立，每个定时器单独占 1 个中断源
- 定时器采用向下计数方式
- 第一个 Timer 单元的定时器支持 PWM 模式
- 使用 PCLK 时钟频率作为定时器计时钟源

9.2 定时器外设时钟

定时器外设时钟由PCLK提供，即定时器时钟频率等于PCLK外设时钟频率

9.3 通用定时器

9.3.1 通用定时器的两种模式

在自由运行 (free-running) 和用户定义 (user-defined) 模式下，当定时器使能后计数值由 TimerNLoadCount 寄存器载入。

根据选择的模式选择载入值：

用户定义模式 (user-defined)：

定时器计数值载入 TimerNLoadCount 寄存器设定值。使用用户模式可以产生固定时间的定时器中断。

自由运行模式 (free-running)：

定时器计数值会载入其允许的最大值，即 0xFFFFFFFF。在定时器产生中断（定时器计数器计数到 0）前用户可以再编程或禁止定时器中断。使用该模式，定时器只产生 1 次中断。中断产生后计数值重置为 0xFFFFFFFF 并向下计数，但不会再产生中断。

9.3.2 中断处理

定时器产生中断后，用户可以通过对 TimerNEOI 或 TimersEOI 进行读操作清除定时器中断状态。

TimerNEOI：只清除对应定时器中断源上的中断状态，既各个定时器的中断清除位。

TimersEOI: 清除该定时器单元中的定时器中断状态，既定时器单元的全局中断清除位。

9.4 PWM模式

Timer0_M单元定时器Timer0和Timer1可编程产生PWM信号。

当用户设定TimerNControlReg中PWM比特位为“1”且定时器处于用户定义模式后，定时器进入PWM工作模式。此时PWM 由TimerNLoadCount2和TimerNLoadCount寄存器分别控制高电平及低电平周期翻转输出。

9.4.1 PWM工作模式

设定TimerNControlReg中PWM位为“1”并设定Mode位为“1”，定时器使能后工作在PWM模式。

9.4.2 PWM周期及占空比设定

PWM信号频率及占空比可通过以下方式进行配置：

- Width of HIGH period = (TimerNLoadCount2 + 1) * PCLK_Period
- Width of LOW period = (TimerNLoadCount + 1) * PCLK_Period

9.5 寄存器描述

9.5.1 地址映射表

TIMER基地址列表

地址范围	基地址	外设	总线
0x5000_A000-0x5000_AFFF	0x5000_A000	Timer0/1	APB0
0x5000_B000-0x5000_BFFF	0x5000_B000	Timer2/3	APB0

TIMERx模块寄存器列表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	Timer1LoadCount	32	0x00000000
0x04	Timer1CurrentValue	32	0xFFFFFFFF
0x08	Timer1ControlReg	32	0x00000000
0x0C	Timer1EOI	32	0x00000000
0x10	Timer1IntStatus	32	0x00000000

0x14	Timer2LoadCount	32	0x00000060
0x18	Timer2CurrentValue	32	0xFFFFFFFF
0x1C	Timer2ControlReg	32	0x00000000
0x20	Timer2EOI	32	0x00000000
0x24	Timer2IntStatus	32	0x00000000
0x28 - 0x9F	预留 x 30	32 x 30	0x00000000
0xA0	TimersIntStatus	32	0x00000000
0xA4	TimersEOI	32	0x00000000
0xA8	TimersRawIntStatus	32	0x00000000
0xAC	预留	32	0x00000000
0xB0	Timer1LoadCount2	32	0x00000000
0xB4	Timer2LoadCount2	32	0x00000000

9.5.2 自动重载计数器（TimerNLoadCount）（N=0...5）

■ **Name: TimerN Load Count Register**

■ **Size:** 32 bits

■ **Address Offset:**

for N = 1, 0x00

for N =2, 0x14

■ **Read/write access:** read/write

31:0

TimerN Load Count

Bit	Name	R/W	Description
31:0	TimerN Load Count	R/W	该值被自动加载到 TimerN 中计数。

9.5.3 自动重载计数器2（TimerNLoadCount2）（N=0...5）

■ **Name: TimerN Load Count Register 2**

■ **Size:** 32 bits

■ **Address Offset:**

for N = 1, 0xb0

for N = 2, 0xb4

■ **Read/write access:** read/write

31: 0

TimerN Load Count2

Bit	Name	R/W	Description
31:0	TimerN Load Count2	R/W	当定时器工作在 PWM 模式中时, 该值被自动加载到 TimerN 中计数。当该值被加载到 TimerN 中, 在此计数期间 PWM 输出保持高电平。

9.5.4 当前计数器值 (TimerNCurrentValue) (N=0...5)

■ **Name:** TimerN Current Value Register

■ **Size:** 32 bits

■ **Address Offset:**

for N = 1, 0x04

for N = 2, 0x18

■ **Read/write access:** read only

31:0

TimerNCurrentValue

Bit	Name	R/W	Description
31:0	TimerN CurrentVa lue	R	TimerN 当前计数值。

9.5.5 控制寄存器 (TimerNControlReg) (N=0...5)

■ **Name:** TimerN Control Register

■ **Size:** 32 bits

■ **Address Offset:**

for N = 1, 0x08

for N = 2, 0x1C

■ **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1		1		16
1	0	9	8	7	6	5	4	3	2	1	0	9	18	7		
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留												P	Int	M	En	
												W	Ma	o	ab	
												M	sk	d	le	
														e		

Bit	Name	R/W	Description
31:4	预留	-	
3	PWM	R/W	PWM 使能位： 0-PWM 模式关闭 1-PWM 模式打开
2	IntMask	R/W	中断屏蔽位： 0-中断打开 1-中断屏蔽
1	Mode	R/W	通用定时器工作模式： 0-自由运行模式（free-running）： 1-用户定义模式（user-defined）：
0	Enable	R/W	定时器使能位： 0-关闭 1-打开

9.5.6 中断清除寄存器（TimerNEOI）（N=0...5）

■ **Name:** TimerN End-of-Interrupt Register

■ **Size:** 32 bits

■ **Address Offset:**

for N = 1, 0x0C

for N = 2, 0x20

■ **Read/write access:** read only

31:1	0
预留	Timer NEOI

Bit	Name	R/W	Description
31:1	预留	-	
0	TimerNE OI	R	对该位读操作清除定时器中断状态。返回值为 0

9.5.7 中断状态寄存器（TimerNIntStatus）（N=0...5）

- **Name:** TimerN Interrupt Status Register
- **Size:** 32bits
- **Address Offset:**
 - for N = 1, 0x10
 - for N = 2, 0x24
- **Read/write access:** read only

31:1	0
预留	TimerNIntStat us

Bit	Name	R/W	Description
31:1	预留	-	
0	TimerNInt Status	R	定时器中断标志位，定时器产生中断该位为“1”。

9.5.8 全局中断清除寄存器（TimersEOI）

- **Name:** Timers End-of-Interrupt Register
- **Size:** 32 bits
- **Address Offset:** 0xa4
- **Read/write access:** read only

该寄存器状态不受TimerNControlReg中IntMask位影响，当定时器计数值计数为0后对应位置“1”。

31:2	1	0
预留	Time rN_1	Time rN_0

Bit	Name	R/W	Description
31:2	预留	-	
1	TimerN_1	R	定时器 TimerN_1 中断标志位，定时器产生中断该位为“1”。
0	TimerN_0	R	定时器 TimerN_0 中断标志位，定时器产生中断该位为“1”。

9.5.9 全局原中断状态寄存器（TimersRawIntStatus）

- **Name: Timers Interrupt Status Register**
- **Size:**32 bits
- **Address Offset:**0xa8
- **Read/write access:** read only

该寄存器中断标志值不受寄存器TimerNControlReg中IntMask影响。

寄存器TimersIntStatus的值是寄存器TimersRawIntStatus经IntMask后的值。当TimersIntStatus有效位置“1”，Timer向CPU发出中断请求。

31:2	1	0
预留	Time rN_1	Time rN_0

Bit	Name	R/W	Description
31:2	预留	-	
1	TimerN_1	R	定时器 TimerN_1 原中断标志位，定时器产生中断该位为“1”。
0	TimerN_0	R	定时器 TimerN_0 原中断标志位，定时器产生中断该位为“1”。

通用异步收发器（UART）

10 UART简介

通用异步收发器(UART)提供了一种灵活的方法与使用工业标准NRZ异步串行数据格式的外部设备之间进行全双工数据交换。UART利用波特率发生器提供宽范围的波特率选择。

通用异步收发器(UART)支持单向通信、双工通信，以及CTS/RTS操作。

10.1 接收/发送FIFO

10.1.1 接收/发送FIFO介绍

UART外设可配置使用FIFO进行收发数据。每个UART外设均包括16bytes的独立的接收和发送FIFO。

10.1.2 接收/发送FIFO

当用户使能FIFO后，CPU写THR寄存器的数据被保存到发送FIFO中，UART外设接收到的数据被保存到接收FIFO中。用户可以通过UART状态寄存器（USR）等相关寄存器获取FIFO中有效数据数量或FIFO状态。也可以配置UART中断，设定在特定条件下向CPU产生中断请求后处理FIFO中数据。

10.1.3 接收/发送FIFO中断使用

触发阈值配置：

用户可使用FIFO控制寄存器（FCR）或其对应的影子寄存器配置接收/发送FIFO数据中断触发的阈值。当接收/发送FIFO中的数据满足设定阈值后会触发对应使能的FIFO中断。

接收FIFO中断：

FIFO模式启用，ERBFI中断使能，当接收FIFO中接收的数据量满足设定阈值后触发“接收数据有效中断”。

发送FIFO中断：

FIFO模式启用，ETBEI中断使能，并且可编程THRE中断模式使能(IER[7] = 1)，当发送FIFO中剩余的数据量满足设定阈值后触发“发送寄存器空中断”。在数据超时未取的情况下触发“字符超时中断”。

10.1.4 接收/发送FIFO访问模式

UART外设提供FIFO访问模式，该模式用于程序调试。在FIFO访问模式中，CPU可对接收FIFO进行写操作，对发送FIFO进行读操作。

进入访问模式：

FAR[0] = 1, FIFO访问模式（FIFO Access mode）使能打开，使能打开后发送和接收FIFO被硬件清空。

发送FIFO测试：

在FIFO访问模式下，写入到发送FIFO中的数据不会被移位发送，而是一直预留在发送FIFO。用户可以通过对TFR寄存器进行读取FIFO中数据，用于测试数据的正确性。

接收FIFO测试：

在FIFO访问模式下，用户通过对RFW(Receive FIFO Write) 寄存器写操作向接收FIFO中写入数据，其中RFW[9]用于测试帧错误的产生，RFW[8]用于测试校验错误产生。数据可以由接收FIFO正常的回读。由于在FIFO访问模式下正常的操作被禁止，所以数据必人为写入到接收FIFO，无法有外部接收。

10.2 UART外设时钟

UART外设时钟由内部PCLK提供，即UART外设时钟频率等于PCLK时钟频率。

10.3 中断（Interrupt）

UART外设产生中后，用户可以通过IIR寄存器获取中断类型。

UART外设可产生中断类型如下：

- Modem 状态中断
- 发送寄存器空中断
- 接收数据有效中断
- Line 状态中断
- UART 忙中断
- 字符超时中断

10.4 可编程THRE中断（Programmable THRE Interrupt）

UART外设可以通过编程THRE中断模式来实现。

THRE中断模式设定关系如下：

THRE中断	可编程THRE中断模式 使能 (IER[7])	FIFO使能	THRE中断使能 (IER[1])
无中断	-	-	×
THR寄存器为空时 产生中断	-	×	√
THR和发送FIFO均 为空时产生中断	×	√	√
发送FIFO数据到达 或低于设定阈值时 产生中断	√	√	√

发送FIFO可设置的空阈值（FCR[5:4]）如下：

- empty
- 2chars
- ¼ Full
- ½ Full

10.5 DMA支持

UART外设使用DMA功能可以有效的减少系统中断，提高数据传输效率。每个UART外设可以使用2个DMA通道，分别用来接收和发送数据。

UART外设使用DMA功能时可以选择是否使用UART FIFO。在不使用FIFO情况下，UART在每次发完成或接收到数据后向DMA发出请求。使用FIFO情况下，UART在每次接收/发送FIFO中数据量满足触发阈值后向DMA发出请求。由此UART提供了2中DMA模式中用户选择。

UART提供了2中可选的DMA模式：

DMA模式0：FIFO控制寄存器（FCR）bit3置0

DMA模式1：FIFO控制寄存器（FCR）bit3置1

10.6 寄存器描述

10.6.1 地址映射表

UARTx（x=0...1）基地址列表

地址范围	基地址	外设	总线
0x5000_1000-0x5000_1FFF	0x5000_1000	UART0	APB0
0x5000_2000-0x5000_2FFF	0x5000_2000	UART1	

UART寄存器偏移地址列表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	DLL/THR/RBR	32	0x00000000
0x04	DLH/IER	32	0x00000000
0x08	IIR/FCR	32	IIR = 0x00000001 FCR = 0x00000000
0x0C	LCR	32	0x00000000
0x10	MCR	32	0x00000000
0x14	LSR	32	0x00000060
0x18	MSR	32	0x00000000
0x1C	SCR	32	0x00000000
0x20	LPDLL	32	0x00000000
0x24	LPDLH	32	0x00000000
0x28	预留	32	0x00000000
0x2C	预留	32	0x00000000
0x30-0x6C	SRBR/STHR	32	0x00000000
0x70	FAR	32	0x00000000
0x74	TFR	32	0x00000000
0x78	RFW	32	0x00000000
0x7C	USR	32	0x00000006
0x80	TFL	32	0x00000000
0x84	RFL	32	0x00000000
0x88	SRR	32	0x00000000
0x8C	SRTS	32	0x00000000
0x90	SBCR	32	0x00000000
0x94	SDMAM	32	0x00000000
0x98	SFE	32	0x00000000
0x9C	SRT	32	0x00000000
0xA0	STET	32	0x00000000
0xA4	HTX	32	0x00000000
0xA8	DMASA	32	0x00000000

10.6.2 接收缓存寄存器（RBR）

- **Name:** Receive Buffer Register
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** read-only

RBR寄存器只有当DLAB比特位清0后才可以访问。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留								Receive Buffer Register							

Bit	Name	R/W	Description
7:0	Receive Buffer Register	R	接收数据寄存器 UART外设 在串口模式下用于接收数据。 UART外设成功接收数据后，LSR寄存器中DR位置“1”。 在非FIFO模式下，未及时读取的数据会被下次接收的数据覆盖，并产生溢出错误标志。 在FIFO模式下，新接收数据被保存在FIFO头部。若FIFO以满，后续接收的数据会被丢弃，并产生溢出错误标志。 复位值：0x00

10.6.3 发送保持寄存器（THR）

- **Name:** Transmit Holding Register
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** write-only

THR寄存器只有当DLAB比特位清0后才可以访问。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留								Transmit Holding Register								

Bit	Name	R/W	Description
7:0	Transmit Holding Register	W	发送数据寄存器 UART外设在串口模式下用于发送数据。 在非FIFO模式，数据发送保持位（THRE）置“1”，发送数据寄存器为空，数据发送保持位（THRE）清“0”，发送数据寄存器不为空。 在FIFO模式下，在FIFO未满的情况下可以连续写入数据，当FIFO已满继续写入的数据将丢失。 复位值：0x00

10.6.4 分频寄存器_高（DLH）

- **Name:** Divisor Latch High
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

DLH寄存器只有在LCR寄存器的DLAB位置1并且UART不是忙状态（USRT bit0 为0）时才能够访问。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留								Divisor Latch(High)							

Bit	Name	R/W	Description
7:0	Divisor Latch (High)	R/W	波特率分频寄存器高位。 波特率=输入时钟/(16*分频波特率寄存器值), UART输入时钟即PCLK时钟周期 当波特率分频寄存器(DLL和DLH)的值设置为0, 波特率时钟关闭, 串口不可通信。 设定DLH寄存器后, UART经过8个波特率时钟后, 进行数据接收或发送操作。 复位值: 0x00

10.6.5 分频寄存器_低 (DLL)

- **Name:** Divisor Latch Low
- **Size:** 32 bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

DLL寄存器只有在LCR寄存器的DLAB位置1并且UART不是忙状态（USRT bit0 为0）时才能够访问。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0

预留	Divisor Latch(Low)
----	--------------------

Bit	Name	R/W	Description
7:0	Divisor Latch (Low)	R/W	波特率分频寄存器低位。 波特率=输入时钟/(16*分频波特率寄存器值), UART输入时钟即PCLK时钟周期 当波特率分频寄存器(DLL和DLH)的值设置为0, 波特率时钟关闭, 串口不可通信。 设定DLH寄存器后, UART经过8个波特率时钟后, 进行数据接收或发送操作。 复位值: 0x00

10.6.6 中断使能寄存器 (IER)

- **Name:** Interrupt Enable Register
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

IER寄存器只有在LCR寄存器的DLAB位清0后才可以访问。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0										

预留				P T I M E	预留				E D S S I	E L S I	E T B E I	E R B F I
----	--	--	--	-----------------------	----	--	--	--	-----------------------	------------------	-----------------------	-----------------------

Bit	Name	R/W	Description
7	PTIME	R/	Programmable THRE Interrupt Mode Enable

		W	用于设定THRE中断模式，用来控制是否产生THRE中断 0 = disabled 1 = enabled 复位值：0x00
6:4	预留	-	-
3	EDSSI	R/ W	Enable Modem Status Interrupt 使能Modem状态中断，用来控制是否产生Modem状态中断。中断优先级为4（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
2	ELSI	R/ W	Enable Receiver Line Status Interrupt 使能Line状态中断，用来控制是否产生Line状态中断。中断优先级为1（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
1	ETBEI	R/ W	Enable Transmit Holding Register Empty Interrupt 使能发送寄存器空中断，用来控制是否产生发送寄存器空中断。 中断优先级为3（优先相应高优先级中断）。 0 = disabled 1 = enabled 复位值：0x00
0	ERBFI	R/ W	Enable Received Data Available Interrupt 使能接收数据有效中断，用来控制是否产生接收数据有效中断和接收字节超时中断（在FIFO模式下或FIFO使能）。 中断优先级为2（优先相应高优先级中断）。。 0 = disabled 1 = enabled 复位值：0x00

10.6.7 中断标志寄存器（IIR）

- **Name:** Interrupt Identity Register
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留	FIFO SE	预留	IID
----	------------	----	-----

Bit	Name	R/W	Description
7:6	FIFOSE	R	FIFO State Enabled FIFO使能状态。 00 = disabled 11 = enabled 复位值: 0x00
4:5	预留	-	-
3:0	IID	R	Interrupt ID UART中断标志，对应中断类型（比特位表示） 0000 – Modem状态中断 0001 – 未发生中断 0010 – 发送寄存器空中断 0100 – 接收数据有效中断 0110 – Line状态中断 0111 – 忙中断 1100 – 字符超时中断（接收FIFO使能） 复位值: 0x01

中断号	中断描述			
二进制	响应优	中断类型	中断源	中断清除操作

表示	优先级			
0001	-	无	无	
0110	1	Line状态中断	接收过载错误、校验错误、帧错误或接收到break中断	读Line中断状态寄存器（LSR）
0100	2	接收数据有效中断	①非FIFO模式中： 接收到有效数据后 ②FIFO模式中：接收FIFO中的数据量达到设定阈值后	①非FIFO模式中：读取接收缓冲寄存器（RBR） ②FIFO模式中：读取接收FIFO中的数据，使接收FIFO中的数据量减少到触发阈值以下
1100	2	字节超时中断	在FIFO模式中，接收FIFO中有有效数据且在上个有效数据接收后的4单位时间内未接收到数据。 1单位时间：1个字节数据接收的用时	读取接收数据寄存器（RBR）
0010	3	发送寄存器空中断	①IRE寄存器PTIME位置0（IRE[7] = 0）： 发送保持寄存器为空时 ②IRE寄存器PTIME位置1（IRE[7] = 1）： 发送FIFO的数据低于设定的阈值时触发中断（IRE寄存器	如果中断源为①则给THR寄存器写数据或关闭发送空中断使能 如果中断源为②则写发送FIFO直到数据高于设定的触发阈值或关闭发送空中断使能

			中PTIME置1)	
0000	4	Modem状态 中断	Modem状态寄存器 相关位置1。 如果自动控制流使 能，则CTS的改变 不会引起中断	读Modem状态寄 存器
0111	5	忙中断	UART处于忙状态 时，尝试对LCR寄 存器进行写操作	读USR寄存器

10.6.8 FIFO控制寄存器（FCR）

- **Name:** FIFO Control Register
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** write-only

3 3 2 2 2 2 2 2 2 2 2 1 18 17 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1 1 1 1 1 1 9 8 7 6 5 4 3 2 1 0
 5 4 3 2 1 0

预留				RC VR	TE T	D M A M	X FI F O R	R FI F O R	F I F O E
----	--	--	--	----------	---------	------------------	------------------------	------------------------	-----------------------

Bit	Name	R/ W	Description
7:6	RCVR	W	Receiver Trigger 接收FIFO满触发的阈值设定。 当接收FIFO中的数据超过设定的阈值后，会触发接收 数据有效中断。 在自动流控模式下，该阈值用于决定接收rts_n信号状态

			<p>(在RTC_FCT使能关闭的状态下)。</p> <p>在DMA模式下, FIFO中数据量触发阈值后UART发出DMA请求。</p> <p>以下为支持的阈值类型。</p> <p>00 – 1 character in the FIFO</p> <p>01 – FIFO ¼ full</p> <p>10 – FIFO ½ full</p> <p>11 – FIFO 2 less than full</p> <p>复位值: 0x00</p>
5:4	TET	W	<p>TX Empty Trigger</p> <p>发送FIFO空的阈值设定。</p> <p>当PTIME使能打开(IER[7] = 1), 发送FIFO中数据等于或低于该阈值将触发THRE中断产生。</p> <p>在DMA模式下, FIFO中数据量触发阈值后UART发出DMA请求。</p> <p>以下为支持的阈值类型。</p> <p>00 – FIFO empty</p> <p>01 – 2 characters in the FIFO</p> <p>10 – FIFO ¼ full</p> <p>11 – FIFO ½ full</p> <p>复位值: 0x00</p>
3	DMAM	W	<p>DMA Mode</p> <p>在额外DMA握手信号没有选择 (DMA_EXTRE = NO) 的情况下用于确定DMA发送请求和接收请求信号的模式。</p> <p>0 = mode0</p> <p>1 = modd1</p> <p>复位值: 0x00</p>
2	XFIFOR	W	<p>XMIT FIFO Reset</p> <p>发送FIFO复位。</p> <p>复位发送FIFO相关状态信息并清空发送FIFO。复位会使DMA TX请求信号无效。</p> <p>置“1”后, 由硬件清“0”。</p> <p>复位值: 0x00</p>
1	RFIFOR	W	<p>RCVR FIFO Reset</p>

			接收FIFO复位。 复位接收FIFO相关状态信息并清空接收FIFO。复位会使DMA RX请求信号无效。 置“1”后，由硬件清“0”。 复位值：0x00
0	FIFOE	W	FIFO Enable 使能接收和发送FIFO。 使能操作会导致接收和发送FIFO复位。 复位值：0x00

10.6.9 Line控制寄存器（LCR）

- **Name:** Line Control Register
- **Size:** 32 bits
- **Address Offset:** 0x0C
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留								D L A B	B C	S P	E P S	P E N	S T O P	DLS	
----	--	--	--	--	--	--	--	------------------	--------	--------	-------------	-------------	------------------	-----	--

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	DLAB	R/W	Divisor Latch Access Bit DLL 和 DLH 读写操作使能位。 DLL/DLH 与其他寄存器地址复用，该为作为操作切换开关。

			0-对相应复用寄存器操作 1-对 DLL/DLH 操作 置“1”后，需软件清“0”。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 复位值：0x00
6	BC	R/W	Break Control Bit 产生输出break信号到接收设备，该位置1，UART外设输出引脚会强行输出逻辑0信号。 在非回环模式中（MCR[4] = 0），UART模式（MCR[4] = 0），sout输出将强制拉低直。该位清0后停止输出。 在回环模式下（MCR[4] = 1），输出的break信号由内部回环到接收器，sout输出被强制拉低。 复位值：0x00
5	SP	R/W	Stick Parity 强制产生校验位值 当 PEN、EPS、SP 设置 1，校验位将输出逻辑 0。 当 PEN、SP 设置 1，EPS 置 0，校验位将输出逻辑 1。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = disable 1 = enable 复位值：0x00
4	EPS	R/W	Even Parity Select 选择奇偶校验方式。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = Odd 1 = Even 复位值：0x00
3	PEN	R/W	Parity Enable 奇偶校验使能 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = disable 1 = enable 复位值：0x00
2	STOP	R/W	STOP Bits 停止位个数选择。

			写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 0 = 1 stop bit 1 = 当 DLS 中 LCR[1:0]=0 时为 1.5 stop bit，其他值时为 2 stop bit 复位值：0x00
1:0	DLS	R/W	Data Length Select 数据位个数。 写操作，需要 USR[0]为 0，即 UART 外设不在忙状态。 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits 复位值：0x00

10.6.10 Modem控制寄存器（MCR）

- **Name:** Modem Control Register
- **Size:** 32 bits
- **Address Offset:** 0x10
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留	S	A	L	O	O	R	D
	I	F	B	U	U	T	T
	R	C	E	T	T	S	R
	E	E		2	1		

Bit	Name	R/W	Description
31:7	预留	-	读操作返回 0

6	SIRE	R/W	SIR Mode Enable 红外模式使能(不使用)。 复位值: 0x00
5	AFCE	R/W	Auto Flow Control Enable 自动流控使能。 0 = disable 1 = enable 复位值: 0x00
4	LB	R/W	LookBack Bit 诊断模式使能。该模式用于测试。 在 UART 模式中(MCR[6] = 0),, sout 输出上的数据保持高电平, sout 输出的数据内部回环到 sin 输入。在此模式中所有中断都是可用的。 在回环模式中, modem 控制输入(dsr_n, cts_n, ri_n, dcd_n)不连接, modem 控制输出(dtr_n, rts_n, out1_n, out2_n)均内部回环到 modem 控制输入。 复位值: 0x00
3	OUT2	R/W	OUT2 Output2 引脚(out2_n)的输出。 0 = 输出逻辑 1 1 = 输出逻辑 0 在 LookBack 模式中(MCR[4] = 1), out2_n 保持高电平, 此时该位置时内部环路的一个输入。 复位值: 0x00
2	OUT1	R/W	OUT1 Output1 引脚(out1_n)的输出。 0 = 输出逻辑 1 1 = 输出逻辑 0 在 LookBack 模式中(MCR[4] = 1), out1_n 保持高电平, 此时该位置时内部环路的一个输入。 复位值: 0x00
1	RTS	R/W	Request to Send RTC 输出。

			<p>具体控制方式如下：</p> <p>1、自动流控未使能（MCR[5]=0）：</p> <p>该位直接控制 RTS 引脚（Request to Send）的输出。置 1 时 RTS 引脚输出有效电平（低电平），清 0 时 RTS 引脚输出失效电平（高电平）。</p> <p>2、自动流控使能（MCR[5]=1）且 FIFO 使能（FCR[0]=1）：</p> <p>该位作为 RTC 使能位。置 1 时 RTS 引脚输出使能打开，清 0 时 RTS 引脚输出使能关闭。</p> <p>RTS 引脚输出电平由接收 FIFO 阈值触发控制，当接收数据低于阈值时 RTS 引脚输出有效电平（低电平），当接收数据等于或高于阈值时 RTS 引脚输出无效电平（高电平）。</p>			
			自动流控	FIFO	RTS	注
			使能关闭 MCR[5]=0	使能关闭 MCR[5]=0	直接控制	
			使能关闭 MCR[5]=0	使能打开 FCR[0]=1	直接控制	
			使能打开 MCR[5]=1	使能关闭 MCR[5]=0	--	自动流控模式必须有 FIFO 支持
			使能打开 MCR[5]=1	使能打开 FCR[0]=1	使能控制	
			复位值：0x00			

0	DTR	R/W	Data Terminal Ready 数据端就绪态输出 写入寄存器的置与引脚输出的值的逻辑相反 0 = dtr_n 逻辑 1 1 = dtr_n 逻辑 0 数据端就绪的输出用于通知 modem，UART 通信建立完成。 复位值：0x00
---	-----	-----	---

10.6.11 Line状态寄存器（LSR）

- **Name:** Line Status Register
- **Size:** 32 bits
- **Address Offset:** 0x14
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留								R F E	T E M T	T H R E	B I	F E	P E	O E	D R
----	--	--	--	--	--	--	--	-------------	------------------	------------------	--------	--------	--------	--------	--------

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	RFE	RC_ R	Receiver FIFO Error bit 接收 FIFO 错误标志位 在 FIFO 使能的情况下（FCR[0] = 1），该位用于表示接收 FIFO 中至少有一个字节数据存在校验错误，帧错误或 break 状态。

			<p>0 = RX FIFO 没有错误</p> <p>1 = RX FIFO 有错误</p> <p>在错误字节处于接收 FIFO 顶部且接收 FIFO 中不再有错误字节的数据的情况下，读 LSR 寄存器该位清 0。</p> <p>复位值：0x00</p>
6	TEMT	R	<p>Transmitter Empty bit</p> <p>①非 FIFO 模式下（FCR[0] = 0）：</p> <p>该位为 1 表示发送移位寄存器（TSR）和发送保持寄存器（THR）同时为空。</p> <p>②FIFO 模式（FCR[0] = 1）：</p> <p>该位为 1 表示发送移位寄存器（TSR）和 FIFO 同时为空。</p> <p>其中：</p> <p>名词缩写</p> <p>TSR：Transmitter Shift Register</p> <p>THR：Transmitter Holding Register</p> <p>不满足以上置位条件，硬件清 0。</p> <p>复位值：0x01</p>
5	THRE	R	<p>Transmit Holding Register Empty bit.</p> <p>THRE 中断使能，当 THRE 置 1 时会触发 THRE 中断。</p> <p>①THRE 中断模式清 0（IER[7] = 0）：</p> <p>该位用来表示 THR 或发送 FIFO 为空（FIFO 使能有效）。当数据由 THR 或发送 FIFO 输出到 TSR 并且没有新的数据写入到 THR 或发送 FIFO 中时都会导致该位置 1。</p> <p>②THRE 中断模式置 1（IER[7] = 1）且 FIFO 使能有效（FCR[0] = 1）：</p> <p>该位的功能转变为表示发送 FIFO 阈值触发的状态，不再用于表示 THR 为空，发送 FIFO 阈值由 FCR[5:4]设定，当发送 FIFO 中的数据量触发阈值后该位置 1。</p>

			<p>不满足以上置位条件，硬件清 0。</p> <p>复位值：0x01</p>
4	BI	RC_ R	<p>Break Interrupt bit</p> <p>Break 中断标志位</p> <p>该位用来指示 UART 外设接收到对方设备的 break 序列信号数据。如果 UART 外设接收到了 break 信号，break 信号会被认为是每个比特都为 0 的字节接收。若 break 信号保持时间超过 1 次以上传输时间也仅作为 1 个字节数据接收。</p> <p>在 UART 模式中（MCR[6] = 0），如果输入引脚保持逻辑 0 的时长大于起始位、数据位、校验位和停止位的时间之和，该位置 1。</p> <p>在 SIR 模式中（MCR[6] = 1），如果输入引脚连续输入的逻辑 0 脉冲时长大于起始位、数据位、校验位和停止位之和，该位置 1。</p> <p>在非 FIFO 模式中（FCR[0] = 0）： 当 break 信号数据字节被接收后该位马上置 1</p> <p>在 FIFO 模式中（FCR[0] = 1）： 当 break 信号数据字节被接收且处于队列顶部位置时，该位置 1。</p> <p>注： 在接收 FIFO 已经满的情况下接收到一个 break 信号，会产生 FIFO 过载。此时，该字节数据的其他附加信息（校验信息，帧错误信息以及 break 信息）都将被丢弃。读 LSR 寄存器该位清 0。</p> <p>复位值：0x00</p>
3	FE	RC_ R	<p>Framing Error bit</p> <p>帧错误标志位</p> <p>该位用于指示数据接收中发生的帧错误。当 UART 外设接收器没接收到有效的停止位时会产生帧错误。</p>

			<p>在 FIFO 模式中，每个字节数据被接收时都会附带一个帧错误信息。产生帧错误的字节数据处于接收 FIFO 顶部时，帧错误标志置 1。</p> <p>当产生帧错后，UART 外设会尝试重现同步。UART 外设会假设该错误是由于下个字节的起始位引起的并继续接收后续比特位。</p> <p>需要注意的是，当接收到一个 break 信号数据后帧错误标志会被置 1，也就是说 break 标志置位的同时也会导致帧错误标志置位。因为 break 信号是由连续的逻辑 0 电平表示，所以必定会产生帧错误。</p> <p>0 = 无帧错误 1 = 有帧错误</p> <p>读 LSR 寄存器该位清 0。</p> <p>复位值：0x00</p>
2	PE	RC_R	<p>Parity Error bit 奇偶校验错误标志位</p> <p>校验使能有效情况下（LCR[3] = 1），该为用于指示数据接收中接收中发生的校验错误。</p> <p>在 FIFO 模式中，每个字节数据被接收时都会附带一个校验错误信息。产生校验错误的字节数据处于接收 FIFO 顶部时，校验错误标志置 1。</p> <p>需要注意的是，在校验使能有效（LCR[3] = 1）且校验类型为 Odd（LCR[4] = 0）情况下，当接收到一个 break 信号数据后校验错误标志会被置 1，也就是说 break 标志置位的同时也会导致校验错误标志置位。</p> <p>0 = 无校验错误 1 = 有校验错误</p> <p>读 LSR 寄存器该位清 0。</p> <p>复位值：0x00</p>
1	OE	RC_R	<p>Overrun error bit 过载错误标志位</p> <p>该位用于指示过载错误。如果新数据被接收而先前数据</p>

			<p>未及时读取时会产生过载错误。</p> <p>①非 FIFO 模式下 (FCR[0] = 0) :</p> <p>1 个新字节数据被接收而在先前在 BRB 中的数据未被及时读取, 标志位置 1。如果发生过载, 则之前在 BRB 中的数据被覆盖。</p> <p>②FIFO 模式下 (FCR[0] = 1) :</p> <p>在接收 FIFO 已满的情况下接收到 1 个新字节数据, 会发生过载错误, 标志位置 1。如果发生过载, UART 外设会预留先前 FIFO 中接收的数据而丢弃当前接收的数据。</p> <p>0 = 无过载错误 1 = 有过载错误</p> <p>读 LSR 寄存器该位清 0。</p> <p>复位值: 0x00</p>
0	DR	R	<p>Data Ready bit</p> <p>数据有效标志位</p> <p>该位用于指示在 BRB 或接收 FIFO 中至少接收到 1 个有效数据。</p> <p>0 = 没有有效数据 1 = 有有效数据</p> <p>进行以下操作后, 硬件清 0:</p> <p>①非 FIFO 模式下 (FCR[0] = 0), 对 BRB 进行读操作。</p> <p>②FIFO 模式下 (FCR[0] = 1), 对接收 FIFO 进行读操作直到接收 FIFO 为空。</p> <p>复位值: 0x00</p>

10.6.12 Modem状态寄存器 (MSR)

- **Name:** Modem Status Register
- **Size:** 32 bits
- **Address Offset:** 0x18
- **Read/write access:** read-only

0、1、2、3比特用来指示modem控制输入变化，输入发生变化对应位置1。如果IER中modem状态中断使能打开，则会产生相应中断，否则忽略产生的中断。因为在同步modem信号版本的过程中会重置modem控制输入，复位值为0，重置完成后值变为1，所以即使modem中各信号时无效，以上比特位在复位后也会被置1。在复位后对MSR寄存器进行读操作，可以防止不必要的中断产生。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留								D	R	D	C	D	T	D	D	D
								C		S	T	D	E	D	D	
								D		R	S	C	R	S	S	

Bit	Name	R/W	Description
31:8	预留	-	读操作返回 0
7	DCD	R	Data Carrier Detect DCD 状态标志位 该位用于指示当前 modem 控制线 DCD 的状态。当该位置 1,表示 Modem 已经感知到“数据载体”（Data Carrier）。 0 = DCD 输入脚无效（高电平） 1 = DCD 输入脚有效（低电平） 在回环模式中 (MCR[4] = 1), DCD 与 OUT2 (MCR[3]) 状态相同。 复位值: 0x00
6	RI	R	Ring Indicator RI 状态标志位 该位用于指示当前 modem 控制线 RI 的状态。当该位置

			<p>1, 表示 Modem 已经接收到“电话响铃信号” (telephone ringing signal) 。</p> <p>0 = RI 输入脚无效 (高电平)</p> <p>1 = RI 输入脚有效 (低电平)</p> <p>在回环模式中 (MCR[4] = 1), RI 与 OUT1 (MCR[2]) 状态相同。</p> <p>复位值: 0x00</p>
5	DSR	R	<p>Data Set Ready</p> <p>DSR 状态标志位</p> <p>该位用于指示当前 modem 控制线 DSR 的状态。当该位置 1, 表示 Modem 向 UART 外设发送的数据已经准备就绪。</p> <p>0 = DSR 输入脚无效 (高电平)</p> <p>1 = DSR 输入脚有效 (低电平)</p> <p>在回环模式中 (MCR[4] = 1), DSR 与 DTR (MCR[0]) 状态相同。</p> <p>复位值: 0x00</p>
4	CTS	R	<p>Clear to Send</p> <p>CTS 状态标志位</p> <p>该位用于指示当前 modem 控制线 CTS 的状态。当该位置 1, 表示 UART 外设接收到 Modem 的请求数据信号 (RTS)。</p> <p>0 = CTS 输入脚无效 (高电平)</p> <p>1 = CTS 输入脚有效 (低电平)</p> <p>在回环模式中 (MCR[4] = 1), CTS 与 RTS (MCR[1]) 状态相同。</p> <p>复位值: 0x00</p>
3	DDCD	RC_ R	<p>Delta Data Carrier Detect</p> <p>DDCD 状态标志位</p> <p>该位用于指示当前 modem 控制线 DCD 的状态发生变化。该位置 1, 表示上次 MSR 读操作后, DCD 状态发</p>

			<p>生变化。</p> <p>0 = 上次 MSR 读操作后, DCD 的状态未发生变化</p> <p>1 = 上次 MSR 读操作后, DCD 的状态发生变化</p> <p>对 MSR 读操作清 0 该位, 在回环模式中 (MCR[4] = 1), DDCD 指示 OUT2 (MCR[3]) 的状态变化。</p> <p>注, 以下情况 DDCD 会置 1:</p> <p>DDCD 状态为 0, DCD 信号有效且复位产生, 如果 DCD 一直保持有效, 直到复位完成, 则 DDCD 会置 1。</p> <p>复位值: 0x00</p>
2	TERI	RC_ R	<p>Trailing Edge of Ring Indicator</p> <p>TERI 状态标志位</p> <p>该位用于指示当前 modem 控制线 RI 的状态发生变化 (由低电平有效状态变为高电平无效状态)。该位置 1, 表示上次 MSR 读操作后, RI 状态发生变化。</p> <p>0 = 上次 MSR 读操作后, RI 的状态未发生变化</p> <p>1 = 上次 MSR 读操作后, RI 的状态发生变化</p> <p>对 MSR 读操作清 0 该位, 在回环模式中 (MCR[4] = 1), TERI 指示 OUT1 (MCR[2]) 的状态变化 (由低电平有效状态变为高电平无效状态)。</p> <p>复位值: 0x00</p>
1	DDSR	RC_ R	<p>Delta Data Set Ready</p> <p>DDSR 状态标志位</p> <p>该位用于指示当前 modem 控制线 DSR 的状态发生变化。该位置 1, 表示上次 MSR 读操作后, DSR 的状态发生变化。</p> <p>0 = 上次 MSR 读操作后, DSR 的状态未发生变化</p> <p>1 = 上次 MSR 读操作后, DSR 的状态发生变化</p> <p>对 MSR 读操作清 0 该位, 在回环模式中 (MCR[4] = 1), DDSR 指示 DTR (MCR[0]) 的状态变化。</p> <p>注, 以下情况 DDSR 会置 1:</p> <p>DDSR 状态为 0, DSR 信号有效且复位产生, 如果 DSR 一直保持有效, 直到复位完成, 则 DDSR 会置 1。</p>

			复位值: 0x00
0	DCTS	RC_ R	Delta Clear to Send DCTS 状态标志位 该位用于指示当前 modem 控制线 CTS 的状态发生变化。该位置 1, 表示上次 MSR 读操作后, CTS 的状态发生变化。 0 = 上次 MSR 读操作后, CTS 的状态未发生变化 1 = 上次 MSR 读操作后, CTS 的状态发生变化 对 MSR 读操作清 0 该位, 在回环模式中 (MCR[4] = 1), DCTS 指示 CTS (MCR[1]) 的状态变化。 注, 以下情况 DCTS 会置 1: DCTS 状态为 0, CTS 信号有效且复位产生, 如果 CTS 一直保持有效, 直到复位完成, 则 DCTS 会置 1。 复位值: 0x00

10.6.13 暂存器寄存器 (SCR)

- **Name** Scratchpad Register
- **Size:** 32 bits
- **Address Offset:** 0x1C
- **Read/write access:** read/write

31:8	7:0
预留	Scratchpad Register

Bit	Name	R/W	Description
31:1	rsvd	RO	-
7:0	Scratchpad Register	RW	This register is for programmers to use as a temporary storage space. It has no defined purpose in the DW_apb_uart.

10.6.14 低功率除数低位锁存器寄存器 (LPDLL)

- **Name** Low Power Divisor Latch Low Register
- **Size:** 32 bits
- **Address Offset:** 0x20
- **Read/write access:** read/write

This register is only valid when the DW_apb_uart is configured to have SIR low-power reception capabilities implemented (SIR_LP_RX = Yes). If SIR low-power reception capabilities are not implemented,

this register does not exist and reading from this register address returns 0.

If UART_16550_COMPATIBLE = No, then this register can be accessed only when the DLAB bit (LCR[7]) is

set and the UART is not busy—that is, USR[0] is 0; otherwise this register can be accessed only when the

DLAB bit (LCR[7]) is set.

31:8

7:0

预留	Low Power Divisor Latch Low Register
----	--------------------------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	-
7:0	Low Power Divisor Latch Low Register	RW	<p>This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver.</p> <p>The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> <p>Low power baud rate = (serial clock frequency)/(16* divisor)</p> <p>Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to</p>

			0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLL is set, at least eight clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.
--	--	--	--

10.6.15 低功率除数高位锁存器寄存器 (LPDLH)

- **Name** Low Power Divisor Latch High Register
- **Size:** 32 bits
- **Address Offset:** 0x24
- **Read/write access:** read/write

This register is valid only when the DW_apb_uart is configured to have SIR low-power reception capabilities implemented (SIR_LP_RX = Yes). If SIR low-power reception capabilities are not implemented,

this register does not exist and reading from this register address returns 0.

If UART_16550_COMPATIBLE = No, then this register can be accessed only when the DLAB bit (LCR[7]) is

set and the UART is not busy—that is, USR[0] is 0; otherwise this register can be accessed only when the

DLAB bit (LCR[7]) is set.

31:8

7:0

预留	Low Power Divisor Latch High Register
----	---------------------------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	-
7:0	Low Power Divisor Latch High Register	RW	This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. The output low-power baud rate is equal to the serial

			<p>clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> <p>Low power baud rate = (serial clock frequency)/(16* divisor)</p> <p>Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLH is set, at least eight clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p>
--	--	--	---

10.6.16 影子接收缓存寄存器 (SRBR)

- **Name:** Shadow Receive Buffer Register
- **Size:** 32 bits
- **Address Offset:** 0x30 - 0x6C
- **Read/write access:** read-only

31:8	7:0
预留	Shadow Receive Buffer Register

Bit	Name	R/W	Description
31:8	rsvd	RO	-
7:0	Shadow Receive Buffer Register	RO	<p>This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register</p>

			<p>(LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE = NONE) or FIFOs are disabled (FCR[0] set to 0), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An overrun error also occurs.</p>
--	--	--	---

10.6.17 影子发送缓存寄存器（STHR）

- **Name:** Shadow Transmit Holding Register
- **Size:** 32 bits
- **Address Offset:** 0x30 - 0x6C
- **Read/write access:** read-only

31:8	7:0
预留	Shadow Transmit Holding Register

Bit	Name	R/W	Description
31:8	rsvd	RO	-
7:0	Shadow Transmit Holding Register	WO	<p>This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p>

			<p>If in non-FIFO mode or FIFOs are disabled (FCR[0] set to 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFOs are enabled (FCR[0] set to 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p>
--	--	--	---

10.6.18 FIFO访问使能寄存器（FAR）

- **Name:** FIFO Access Register
- **Size:** 32 bits
- **Address Offset:** 0x70
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留															F A R

Bit	Name	R/W	Description
31:1	预留	-	-
0	FAR	R/W	FIFO Access Register FIFO 访问使能

			<p>该位用于用于 FIFO 测试，控制 FIFO 是否可以被用户访问。当使能打开后，用户可以写接收 FIFO，读发送 FIFO。使能关闭，用户只能通过 RBR 和 THR 来访问 FIFO。】</p> <p>0 = FIFO 访问使能关闭</p> <p>1 = FIFO 访问使能打开</p> <p>注：当 FIFO 访问使能进行打开/关闭操作时，接收和发送 FIFO 的控制部分会复位并且 FIFO 也被视为空。</p> <p>复位值：0x00</p>
--	--	--	--

10.6.19 读发送FIFO寄存器（TFR）

- **Name:** Transmit FIFO Read
- **Size:** 32 bits
- **Address Offset:** 0x74
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留								TFRD								

Bit	Name	R/W	Description
31:8	预留	-	-
7:0	TFRD	R	<p>Transmit FIFO Read</p> <p>读发送 FIFO 数据</p> <p>该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。</p> <p>当 FIFO 功能打开,对该位进行读操作将返回发送 FIFO 队列顶部数据。进行连续的读操作将依次取出 FIFO 中</p>

			<p>的数据，每次读操作读取的数据均是当前发送 FIFO 队列顶部数据。</p> <p>当 FIFO 功能关闭，对该位的读操作将返回 THR 内的数据。</p> <p>复位值：0x00</p>
--	--	--	--

10.6.20 写接收FIFO寄存器（RFW）

- **Name:** Receive FIFO Write
- **Size:** 32 bits
- **Address Offset:** 0x78
- **Read/write access:** write-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

预留						R	R	RFWD							
						F	F								
						F	P								
						E	E								

Bit	Name	R/W	Description
31:1 0	预留	-	-
9	RFFE	W	<p>Receive FIFO Framing Error. 接收 FIFO 帧错误命令</p> <p>该位只有当 FIFO 访问使能打开后操作才有效（FAR[0] = 1）。</p> <p>当 FIFO 功能打开，该位用于写帧错误信息到接收 FIFO。</p> <p>当 FIFO 功能关闭，该位用于写帧错误信息到 RBR。</p> <p>复位值：0x00</p>

8	RFPE	W	Receive FIFO Parity Error. 接收 FIFO 校验错误命令 该位只有当 FIFO 访问使能打开后操作才有效 (FAR[0] = 1)。 当 FIFO 功能打开, 该位用于写校验错误信息到接收 FIFO。 当 FIFO 功能关闭, 该位用于写校验错误信息到 RBR。 复位值: 0x00
7:0	RFWD	W	Receive FIFO Write Data 写接收 FIFO 数据 该位只有当 FIFO 访问使能打开后操作才有效 (FAR[0] = 1)。 当 FIFO 功能打开, 写入该位的数据将压到接收 FIFO 中。每次写入该位的数据将在下次写该位时被压到接收 FIFO 中。 当 FIFO 功能关闭, 写入该位的数据将压到 RBR 中。 复位值: 0x00

10.6.21 UART状态寄存器 (USR)

- **Name:** UART Status Register
- **Size:** 32 bits
- **Address Offset:** 0x7C
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留												R	R	T	T
												F	F	F	F
												F	N	E	N
														B	B
														U	U
														S	S

		E		F	Y
--	--	---	--	---	---

Bit	Name	R/W	Description
31:5	预留	-	-
4	RFF	R	Receive FIFO Full 接收 FIFO 满（completely full）标志。 0 = 接收 FIFO 未滿 1 = 接收 FIFO 滿 当接收 FIFO 不再是满的时候该位被清空。 复位值：0x00
3	RFNE	R	Receive FIFO Not Empty 接收 FIFO 非空标志。 0 = 接收 FIFO 空 1 = 接收 FIFO 非空 当接收 FIFO 为空时该位被清空。 复位值：0x00
2	TFE	R	Transmit FIFO Empty 发送 FIFO 为空（completely empty）标志。 0 = 发送 FIFO 非空 1 = 发送 FIFO 空 当发送 FIFO 为非空时该位被清空。 复位值：0x00
1	TFNF	R	Transmit FIFO Not Full 发送 FIFO 未滿标志。 0 = 发送 FIFO 已滿 1 = 发送 FIFO 未滿 当发送 FIFO 为已滿时该位被清空。 复位值：0x00
0	BUSY	R	UART Busy UART 忙标志位 该位为 1 串行传输正在进行，为 0 表示 UART 外设空闲或不活动。 0 = UART 外设空闲或不活动

			<p>1 = UART 外设忙（正在进行串行数据传输）</p> <p>以下任意一种情况将导致该位置 1，即 UART 外设忙：</p> <p>1、串口接口正在进行发送</p> <p>2、FIFO 访问使能未打开（FAR[0] = 0），波特率分频不为 0（DLH, DLL ≠ 0），分频访问使能关闭（LCR.DLAB = 0）且 THR 中有发送数据</p> <p>3、串口正在进行数据接收</p> <p>4、FIFO 访问使能未打开（FAR[0] = 0）且 RBR 中有接收的数据</p> <p>复位值：0x00</p>
--	--	--	---

10.6.22 发送FIFO数据量（TFL）

- **Name:** Transmit FIFO Level
- **Size:** 32bits
- **Address Offset:** 0x80
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留												Transmit FIFO Level			

Bit	Name	R/W	Description
31:4	预留	-	-
3:0	TFL	R	Transmit FIFO Level 当前发送 FIFO 中数据的个数。 复位值：0x00

10.6.23 接收FIFO数据量（RFL）

- **Name:** Receive FIFO Level

- **Size:** 32bits
- **Address Offset:** 0x84
- **Read/write access:** read-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留												Receive FIFO Level			

Bit	Name	R/W	Description
31:4	预留	-	-
3:0	RFL	R	Receive FIFO Level 当前接收 FIFO 中数据的个数。 复位值: 0x00

10.6.24 软复位寄存器（SRR）

- **Name:** Software Reset Register
- **Size:** 32 bits
- **Address Offset:** 0x88
- **Read/write access:** write-only

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留												X F R	R F R	U R	

Bit	Name	R/W	Description
31:3	预留	-	-
2	XFR	W	<p>XMIT FIFO Reset. 发送 FIFO 复位</p> <p>该操作位是 FCR[2]的影子操作位。当用户只复位发送 FIFO 时，通过 FCR 寄存器复位发送 FIFO 会覆盖之前的 FCR 寄存器设置，使用该影子操作位可以避免上述情况。该操作会复位发送 FIFO 的控制部分并初始发送 FIFO 为空。复位会使 DMA TX 请求信号无效。该位硬件清 0。</p> <p>复位值：0x00</p>
1	RFR	W	<p>RCVR FIFO Reset 接收 FIFO 复位</p> <p>该操作位是 FCR[1]的影子操作位。当用户只复位接收 FIFO 时，通过 FCR 寄存器复位接收 FIFO 会覆盖之前的 FCR 寄存器设置，使用该影子操作位可以避免上述情况。该操作会复位接收 FIFO 的控制部分并初始接收 FIFO 为空。复位会使 DMA TX 请求信号无效。该位硬件清 0。</p> <p>复位值：0x00</p>
0	UR	W	<p>UART Reset UART 复位</p> <p>该位操作复位 UART 外设，需要经过 2 个执行时钟周期，等待 pclk 和 sclk 都完成复位。复位完成后立即清除复位标志。</p> <p>复位值：0x00</p>

10.6.25 发送请求影子寄存器（SRTS）

- **Name:** Shadow Request to Send
- **Size:** 32 bits
- **Address Offset:** 0x8C

■ Read/write access: read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留														S	R
														T	S

Bit	Name	R/W	Description				
31:1	预留	-	-				
0	SRTS	R/W	<p>Shadow Request to Send</p> <p>RTS 输出影子寄存器</p> <p>该操作位是 MCR[1](RTS)的影子操作位。</p> <p>该位用于控制 UART 外设 RTC 输出，具体控制方式如下：</p> <p>1、自动流控未使能（MCR[5]=0）：</p> <p>该位直接控制 RTS 引脚（Request to Send）的输出。置 1 时 RTS 引脚输出有效电平（低电平），清 0 时 RTS 引脚输出失效电平（高电平）。</p> <p>2、自动流控使能（MCR[5]=1）且 FIFO 使能（FCR[0]=1）：</p> <p>该位作为 RTC 使能位。置 1 时 RTS 引脚输出使能打开，清 0 时 RTS 引脚输出使能关闭。</p> <p>RTS 引脚输出电平由接收 FIFO 阈值触发控制，当接收数据低于阈值时 RTS 引脚输出有效电平（低电平），当接收数据等于或高于阈值时 RTS 引脚输出无效电平（高电平）。</p>				
			<table border="1"> <tr> <td>自动流</td><td>FIFO</td><td>RTS</td><td>注</td></tr> </table>	自动流	FIFO	RTS	注
自动流	FIFO	RTS	注				

			控			
			使能关 闭 MCR[5] =0	使能关 闭 MCR[5] =0	直接控 制	
			使能关 闭 MCR[5] =0	使能打 开 FCR[0] = 1	直接控 制	
			使能打 开 MCR[5] =1	使能关 闭 MCR[5] =0	--	自动流 控模式 必须有 FIFO 支 持
			使能打 开 MCR[5] =1	使能打 开 FCR[0] = 1	使能控 制	
复位值： 0x00						

10.6.26 Break信号控制影子寄存器 (SBCR)

- **Name:** Shadow Break Control Register
- **Size:** 32 bits
- **Address Offset:** 0x90
- **Read/write access:** read/write

31:1			0
预留			SBCR
Bit	Name	R/W	Description
31:1	预留	-	-
0	SBCR	R/W	Shadow Break Control Bit Break控制影子寄存器

			<p>该操作位是 LCR[6](Break Control bit)的影子操作位。</p> <p>通过该位可以省去对 LCR 的“读-修改-写”操作。</p> <p>该位用于产生输出break信号到接收设备，该位置1，UART外设输出引脚会强行输出逻辑0信号。</p> <p>当在非回环模式中（MCR[4] = 0），UART模式（MCR[4] = 0），sout输出将强制拉低直。该位清0后停止输出。</p> <p>当在回环模式下（MCR[4] = 1），输出的break信号由内部回环到接收器，sout输出被强制拉低。</p> <p>复位值：0x00</p>
--	--	--	---

10.6.27 DMA模式影子寄存器（SDMAM）

- **Name:** Shadow DMA Mode
- **Size:** 32 bits
- **Address Offset:** 0x94
- **Read/write access:** read/write

31:1	0
预留	SDMAM

Bit	Name	R/W	Description
31:1	预留	-	-
0	SDMAM	R/W	<p>Shadow DMA Mode.</p> <p>DMA 模式影子寄存器</p> <p>该操作位是 FCR[3](DMA mode bit)的影子操作位。</p> <p>该位可以避免在只修改 DMA 模式位的情况下，影响 FCR 之前设置的内容。</p> <p>该位在额外DMA握手信号没有选择（DMA_EXTRE = NO）的情况下用于确定DMA发送请求和接收请求信号的模式。</p> <p>0 = mode0</p> <p>1 = modd1</p> <p>复位值：0x00</p>

10.6.28 FIFO使能影子寄存器（SFE）

- **Name:** Shadow FIFO Enable
- **Size:** 32 bits
- **Address Offset:** 0x98
- **Read/write access:** read/write

31:1	0
预留	SFE

Bit	Name	R/W	Description
31:1	预留	-	-
0	SFE	R/W	Shadow FIFO Enable FIFO 使能影子寄存器 该操作位是 FCR[0](FIFO enable bit)的影子操作位。 该位可以避免在只修改 FIFO 使能位的情况下，影响 FCR 之前设置的内容。 用于使能接收和发送FIFO。当该位发送变化时接收和发送FIFO都将复位。 复位值：0x00

10.6.29 接收FIFO满阈值设置影子寄存器（SRT）

- **Name:** Shadow RCVR Trigger
- **Size:** 32 bits
- **Address Offset:** 0x9C
- **Read/write access:** read/write

31:2	1 0
预留	SRT

Bit	Name	R/W	Description
-----	------	-----	-------------

31:2	预留	-	-
1:0	SRT	R/W	<p>Shadow RCVR Trigger</p> <p>接收FIFO满触发的阈值设定影子寄存器。</p> <p>该操作位是 FCR[7:6](RCVR Trigger)的影子操作位。</p> <p>该位可以避免在只修改接收 FIF 阈值的情况下，影响 FCR 之前设置的内容。</p> <p>当接收FIFO中的数据超过设定的阈值后，会触发接收数据有效中断。</p> <p>在自动流控模式下，该阈值用于决定接收rts_n信号状态（在RTC_FCT使能关闭的状态下）。</p> <p>在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。</p> <p>以下为支持的阈值类型。</p> <p>00 – 1 character in the FIFO</p> <p>01 – FIFO ¼ full</p> <p>10 – FIFO ½ full</p> <p>11 – FIFO 2 less than full</p> <p>复位值：0x00</p>

10.6.30 发送FIFO空阈值设置影子寄存器（STET）

- **Name:** Shadow TX Empty Trigger
- **Size:** 32 bits
- **Address Offset:** 0xA0
- **Read/write access:** read/write

		31:2	1	0
预留		STET		
位	Name	R/W	Description	
31:2	预留	-	-	
1:0	STET	R/W	Shadow TX Empty Trigger	

			<p>发送FIFO空的阈值设定影子寄存器。</p> <p>该操作位是 FCR[5:4](TX Empty Trigger)的影子操作位。</p> <p>该位可以避免在只修改发送 FIF 阈值的情况下，影响 FCR 之前设置的内容。</p> <p>当PTIME使能打开(IER[7] = 1)，发送FIFO中数据等于或低于该阈值将触发THRE中断产生。</p> <p>在DMA模式下，FIFO中数据量触发阈值后UART发出DMA请求。</p> <p>以下为支持的阈值类型。</p> <p>00 – FIFO empty</p> <p>01 – 2 characters in the FIFO</p> <p>10 – FIFO ¼ full</p> <p>11 – FIFO ½ full</p> <p>复位值：0x00</p>
--	--	--	---

10.6.31 发送暂停寄存器（HTX）

- **Name:** Halt TX
- **Size:** 32 bits
- **Address Offset:** 0xA4
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1
5	4	3	2	1	0										0
预留															H
															T
															X

Bit	Name	R/W	Description
31:1	预留	-	-

0	HTX	R/W	Halt TX 发送暂停寄存器 该寄存器用于 UART 测试，对其操作来停止 UART 发送。这样在 FIFO 使能打开的状态下，发送 FIFO 可以被用户写满。 0 = 停止 TX 使能关闭 1 = 停止 TX 使能打开 复位值：0x00
---	-----	-----	---

10.6.32 DMA软应答（DMASA）

- **Name:** DMA Software Acknowledge
- **Size:** 32 bits
- **Address Offset:** 0xA8
- **Read/write access:** write

31:1	0
预留	DM ASA

Bit	Name	R/W	Description
31:1	预留	-	-
0	DMASA	W	DMA Software Acknowledge DMA软应答 如果在DMA传输过程中产生了错误需要终止，那么可以用是该寄存器产生一个DMA软应答。 例如，如果DMA通道使能关闭，UART需要清除它的请求信号，那么对该寄存器的操作将使TX request, TX single, RX request 和 RX single 信号无效。对该寄存器的操作无需用户清0，硬件自行清0。 复位值：0x00

11 串行外设接口(SPI)

11.1 SPI简介

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

11.2 SPI主要特点

- SPI时钟由PCLK提供，即SPI_CLK = PCLK
- 支持协议Motorola Serial Peripheral Interface (SPI)
- 支持协议Texas Instruments Serial Protocol (SSP)
- 支持协议National Semiconductor Microwire
- 包含硬件实现收发FIFO，深度为16
- 独立硬件收发FIFO，可配收发FIFO中断阈值
- 主或从操作（主/从地址不同）
- 4到16位数据帧格式选择
- 支持全双工、半双工模式
- 主模式包含CSN0、CSN1、CSN2、CSN3独立的四个从设备片选信号
- 从模式中使用CSN0片选信号
- 多种收发、错误中断检测
- DMA支持

11.3 SPI功能描述

11.3.1 SPI外设时钟及要求

SPI时钟由PCLK提供，即SPI_CLK = PCLK

SPI_CLK: SPI接入时钟频率

SPI_M_CLK: SPI主模式总线时钟频率

SPI_S_CLK: SPI从模式总线时钟频率

理论时钟要求:

$SPI_CLK \geq 2 \times SPI_M_CLK$

$SPI_CLK \geq 10 \times SPI_S_CLK$

11.3.2 接收/发送FIFO

SPI外设包含2个独立的深度为16的接收和发送FIFO。

CPU对寄存器DR写操作，数据写入发送FIFO，CPU对寄存器DR读操作，数据由接收FIFO中取出。

接收和发送FIFO有独立的中断阈值设定，当数据符合设定阈值时SPI向CPU发出中断请求。

接收和发送FIFO有独立的DMA阈值设定，当数据符合设定阈值时SPI发出相应DMA请求。

设定的阈值需结合DMA Burst (MSIZE)值进行设定，相见DMA“SRC_MSIZE/DEST_MSIZE参数置设定”章节

11.3.3 中断类型

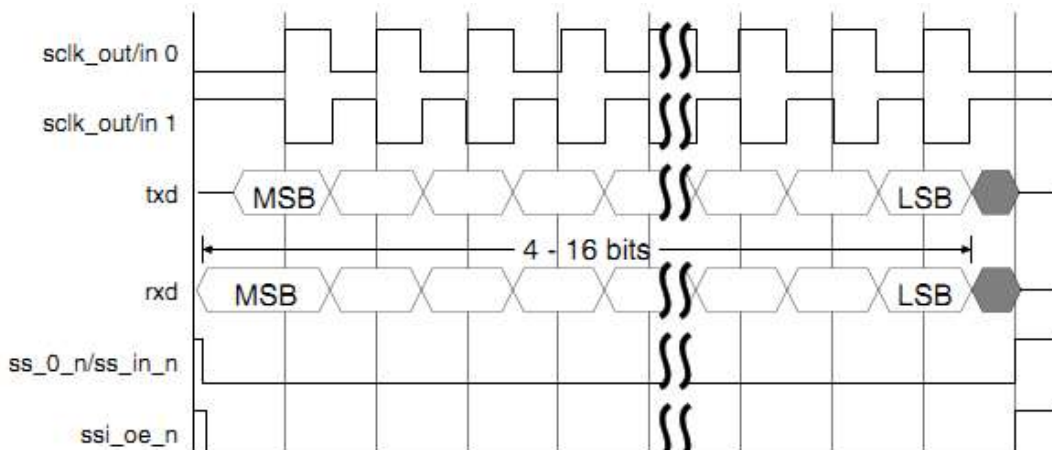
SPI外设支持以下中断类型：

- **Transmit FIFO Empty Interrupt**
发送 FIFO 空中断，当发送 FIFO 中数据量满足设定阈值时产生中断
- **Transmit FIFO Overflow Interrupt**
发送 FIFO 上溢中断，在发送 FIFO 数据已经满的情况下对 DR 进行写操作引起发送 FIFO 上溢中断
- **Receive FIFO Full Interrupt**
接收 FIFO 慢中断，当接收 FIFO 中数据量满足设定阈值时产生中断
- **Receive FIFO Overflow Interrupt**
接收 FIFO 上溢中断，在接收 FIFO 数据已经满的情况下 SPI 外设接收新数据引起接收 FIFO 上溢中断
- **Receive FIFO Underflow Interrupt**
接收 FIFO 下溢中断，接收 FIFO 已经为空，对接收 FIFO 进行读操作会触发接收 FIFO 下溢中断
- **Multi-Master Contention Interrupt**
多主机总线仲裁中断，SPI 工作在主模式下并占用总先，此时有其他主设备选中当前 SPI 外设并传输数据。
- **Combined Interrupt Request**
以上中断类型经过中断屏蔽寄存器后或运算值

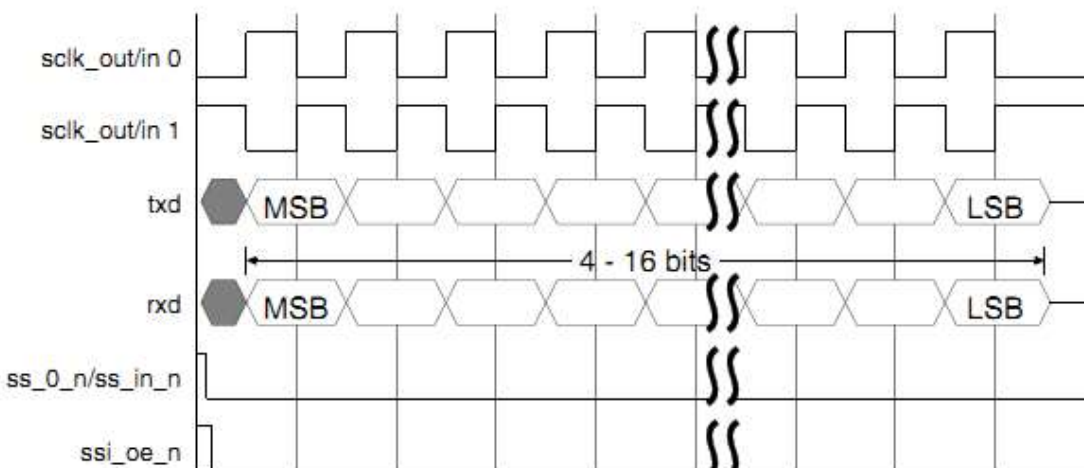
11.3.4 Motorola SPI通行协议

常用Motorola SPI通讯协议支持的四种通讯模式，能够实现全双工通讯。系统上电默认采用模式0工作方式。

SCPH = 0:



SCPH = 1:



sclk_out/ in: 总线时钟，out: SPI为主设备输出CLK。in: SPI为从设备输入CLK

sclk_out/ in = 0: (CPHA) = 0

sclk_out/ in = 1: (CPHA) = 1

ss_0_n/ss_in_n: 片选信号，s_0_n: SPI为主设备时输出片选。s_in_n: SPI为主设备时输入片选

ss_oe_n: SPI为从模式时输出使能选项。

SPI协议规定的4中通讯格式说明如下:

- 模式0: 时钟极性 (CPOL) = 0, 时钟相位 (CPHA) = 0, 该模式下串行同步时钟的空闲状态为低电平, 芯片将在串行同步时钟的第一个跳变沿 (上升沿) 采样, 芯片默认为该模式;
- 模式1: 时钟极性 (CPOL) = 0, 时钟相位 (CPHA) = 1, 该模式下串行同步时钟的空闲状态为低电平, 芯片将在串行同步时钟的第二个跳变沿 (下降沿) 采样;

- 模式2：时钟极性（CPOL）=1，时钟相位（CPHA）=0，该模式下串行同步时钟的空闲状态为高电平，芯片将在串行同步时钟的第一个跳变沿（下降沿）采样；
- 模式3：时钟极性（CPOL）=1，时钟相位（CPHA）=1，该模式下串行同步时钟的空闲状态为高电平，芯片将在串行同步时钟的第二个跳变沿（上升沿）采样。

11.3.5 SPI主/从模式选择

在SPCU中每个SPI外设包括2组寄存器组，分别用于实现主模式（SPIMx）和从模式（SPISx），2组寄存器组结构相同，地址不同。SPI外设工作模式使用SYSCTRL寄存器中CG_CTRL相应位切换。

当工作在主模式下，SPI相应初始化及数据收发操作由SPIMx完成。当工作在从模式下，SPI相应初始化及数据接收操作由SPISx完成

使能SPI0时钟门控并使其处在主模式下，具体流程如下：

```
// CG_CTRL为时钟门控寄存器详见系统控制（SYSCTL）章节  
CG_CTRL |= 1<<8  
// PHER_CTRL为外设控制寄存器详见系统控制（SYSCTL）章节  
PHER_CTRL&= ~(1<<24)
```

使能SPI0时钟门控并使其处在从模式下，具体流程如下：

```
// CG_CTRL为时钟门控寄存器详见系统控制（SYSCTL）章节  
CG_CTRL |= 1<<8  
// PHER_CTRL为外设控制寄存器详见系统控制（SYSCTL）章节  
PHER_CTRL |= 1<<24
```

11.3.6 SPI主模式配置

主模式使用SPIMx寄存器组进行配置

配置流程如下：

- 1、 配置CG_CTRL使SPI处于主模式下
- 2、 配置SPIMx->SSIENR = 0，SPI使能关闭配置SPIMx->IMR= 0，关中断配置SPIMx->CTRLR0，配置传输模式，帧模式，时钟极性，时钟相位，及通讯数据宽度
- 3、 配置SPIMx->BAUDR，配置需要的波特率
- 4、 配置SPIMx->SER，初始化片选信号

- 5、 配置SPIMx-> RXFTLR和SPIMx-> TXFTLR，配置收发FIFO中断触发阈值
- 6、 配置SPIMx-> IMR，开启相关中断
- 7、 配置SPIMx-> SSIENR = 1，SPI使能打开

11.3.7 SPI主模式数据收发

主模式使用SPIMx寄存器组进行数据发送/接收

数据发送/接收流程：

判断发送FIFO是否已满。

如果发送FIFO未滿，向SPIMx->DR寄存器中写数据。数据将发送到对应片选从设备。

如果SPIMx-> SER没有使能任何从设备，则在SPIMx-> SER是使能后数据被发送。

当发生发送FIFO空中断时，向SPIMx-> DR写数据到发送FIFO中。当发生接收FIFO满时，读SPIMx-> DR由发送FIFO读取数据。

数据传输完成，SPI回到非忙状态

11.3.8 SPI从模式配置

从模式使用SPISx寄存器组进行配置

配置流程如下：

- 1、 配置CG_CTRL使SPI处于从模式下
- 2、 配置SPISx-> SSIENR = 0，SPI使能关闭
- 3、 配置SPISx-> IMR= 0，关中断
- 4、 配置SPISx-> CTRLR0，配置传输模式，帧模式，时钟极性，时钟相位，从输出使能
- 5、 配置SPISx-> RXFTLR和SPISx-> TXFTLR，配置收发FIFO中断触发阈值
- 6、 配置SPISx-> IMR，开启相关中断
- 7、 配置SPISx-> SSIENR = 1，SPI使能打开

11.3.9 SPI从模式数据收发

从模式使用SPISx寄存器组进行数据发送/接收

数据发送/接收流程：

- 1、 当SPI片选有效后，数据开始传输。
- 2、 当发生发送FIFO空中断时，向SPISx-> DR写数据到发送FIFO中。当发生接收FIFO满时，

读SPISx->DR由发送FIFO读取数据。

3、数据传输完成，SPI回到非忙状态

11.3.10 DMA操作

SPI外设支持DMA数据操作，以减少CPU使用。

DMACR寄存器：

该寄存器用于使能SPI收发DMA功能，分别有2bit控制位操作。

DMATDLR/DMARDLR寄存器：

DMATDLR/DMARDLR寄存器用于控制SPI对DMA产生请求条件。

当发送FIFO中数据满足DMATDLR设定值，则触发SPI对DMA请求，DMA相应请求后，根据对应通道配置向发送FIFO中1次性写入Burst（MSIZE）数据量的数据。

当接收FIFO中数据满足DMARDLR设定值，则触发SPI对DMA请求，DMA相应请求后，根据对应通道配置由发送FIFO中1次性取出Burst（MSIZE）数据量的数据。

具体设定可参考DMA中“SRC_MSIZ/DEST_MSIZ参数置设定”章节。

11.4 SPI寄存器描述

11.4.1 地址映射表

SPIx（x=0...2）基地址列表

地址范围	基地址	外设	总线
0x5000_3000-0x5000_3FFF	0x5000_3000	SPIM/S0	APB0
0x5000_4000-0x5000_4FFF	0x5000_4000	SPIM/S1	

SPI寄存器偏移地址列表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	CTRLR0	16	0x00000007
0x02	保留	16	
0x04	CTRLR1	16	0x00000000
0x06	保留		
0x08	SSIENR	32	0x00000000
0x0C	MWCR	32	0x00000000
0x10	SER	32	0x00000000
0x14	BAUDR	32	0x00000000
0x18	TXFTLR	32	0x00000000

0x1C	RXFTLR	32	0x00000000
0x20	TXFLR	32	0x00000000
0x24	RXFLR	32	0x00000000
0x28	SR	32	0x00000006
0x2C	IMR	32	0x0000003F
0x30	ISR	32	0x00000000
0x34	RISR	32	0x00000000
0x38	TXOICR	32	0x00000000
0x3C	RXOICR	32	0x00000000
0x40	RXUICR	32	0x00000000
0x44	MSTICR	32	0x00000000
0x48	ICR	32	0x00000000
0x4C	DMACR	32	0x00000000
0x50	DMATDLR	32	0x00000000
0x54	DMARDLR	32	0x00000000
0x58-0x5C	预留	32	0xFFFFFFFF
0x60-0xE	DR	32	0x00000000
0xF0	RX_SAMPLE_DLY	32	0x00000000

11.4.2 控制寄存器0（CTRLR0）

- **Name:** Control Register 0
- **Size:** 32 bits
- **Address Offset:** 0x0
- **Read/write access:** read/write

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1											
CFS		S R L	S L V	TM OD		S C P	S C P	FRF		DFS					

		— O E		O L	H		
--	--	-------------	--	--------	---	--	--

Bit	Name	W/R	Description																																
31:1 6	rsvd	RO	预留																																
15:1 2	CFS	W/R	<div>控制帧大小</div> <div>Microwire 帧格式，设定值参见下表：</div> <table><tr><td>0000</td><td>1-bit control word</td></tr><tr><td>0001</td><td>2-bit control word</td></tr><tr><td>0010</td><td>3-bit control word</td></tr><tr><td>0011</td><td>4-bit control word</td></tr><tr><td>0100</td><td>5-bit control word</td></tr><tr><td>0101</td><td>6-bit control word</td></tr><tr><td>0110</td><td>7-bit control word</td></tr><tr><td>0111</td><td>8-bit control word</td></tr><tr><td>1000</td><td>9-bit control word</td></tr><tr><td>1001</td><td>10-bit control word</td></tr><tr><td>1010</td><td>11-bit control word</td></tr><tr><td>1011</td><td>12-bit control word</td></tr><tr><td>1100</td><td>13-bit control word</td></tr><tr><td>1101</td><td>14-bit control word</td></tr><tr><td>1110</td><td>15-bit control word</td></tr><tr><td>1111</td><td>16-bit control word</td></tr></table>	0000	1-bit control word	0001	2-bit control word	0010	3-bit control word	0011	4-bit control word	0100	5-bit control word	0101	6-bit control word	0110	7-bit control word	0111	8-bit control word	1000	9-bit control word	1001	10-bit control word	1010	11-bit control word	1011	12-bit control word	1100	13-bit control word	1101	14-bit control word	1110	15-bit control word	1111	16-bit control word
0000	1-bit control word																																		
0001	2-bit control word																																		
0010	3-bit control word																																		
0011	4-bit control word																																		
0100	5-bit control word																																		
0101	6-bit control word																																		
0110	7-bit control word																																		
0111	8-bit control word																																		
1000	9-bit control word																																		
1001	10-bit control word																																		
1010	11-bit control word																																		
1011	12-bit control word																																		
1100	13-bit control word																																		
1101	14-bit control word																																		
1110	15-bit control word																																		
1111	16-bit control word																																		
11	SRL	W/R	<div>移位寄存器环</div> <div>该位仅用于测试，该位置“1”则输出移位寄存器自动连接到输入移位寄存器上。</div> <div>0-正常操作模式</div> <div>1-测试操作模式</div>																																
10	SLV_OE	W/R	<div>从输出使能</div> <div>该位仅用于模块工作在从模式，当工作在主模式是对该位操作无效</div> <div>0-从发送打开</div> <div>1-从发送关闭</div>																																
9:8	TMOD	W/R	传输模式																																

			00-发送和接收模式 01-只发送 10-只接收 11-EEPROM 模式																																
7	SCPOL	W/R	时钟极性 0-空闲状态时，SCK 保持低电平； 1-空闲状态时，SCK 保持高电平。																																
6	SCPH	W/R	时钟相位 0-数据采样从第一个时钟边沿开始； 1-数据采样从第二个时钟边沿开始。																																
5:4	FRF	W/R	协议帧格式 00-Motorola SPI 01-Texas Instruments SSP 10-National Semiconductors Microwire 11-Reserved																																
3:0	DFS	W/R	数据帧大小 设定值参考下表： <table><tr><td>0000</td><td>预留</td></tr><tr><td>0001</td><td>预留</td></tr><tr><td>0010</td><td>预留</td></tr><tr><td>0011</td><td>4-bit data size</td></tr><tr><td>0100</td><td>5-bit data size</td></tr><tr><td>0101</td><td>6-bit data size</td></tr><tr><td>0110</td><td>7-bit data size</td></tr><tr><td>0111</td><td>8-bit data size</td></tr><tr><td>1000</td><td>9-bit data size</td></tr><tr><td>1001</td><td>10-bit data size</td></tr><tr><td>1010</td><td>11-bit data size</td></tr><tr><td>1011</td><td>12-bit data size</td></tr><tr><td>1100</td><td>13-bit data size</td></tr><tr><td>1101</td><td>14-bit data size</td></tr><tr><td>1110</td><td>15-bit data size</td></tr><tr><td>1111</td><td>16-bit data size</td></tr></table>	0000	预留	0001	预留	0010	预留	0011	4-bit data size	0100	5-bit data size	0101	6-bit data size	0110	7-bit data size	0111	8-bit data size	1000	9-bit data size	1001	10-bit data size	1010	11-bit data size	1011	12-bit data size	1100	13-bit data size	1101	14-bit data size	1110	15-bit data size	1111	16-bit data size
0000	预留																																		
0001	预留																																		
0010	预留																																		
0011	4-bit data size																																		
0100	5-bit data size																																		
0101	6-bit data size																																		
0110	7-bit data size																																		
0111	8-bit data size																																		
1000	9-bit data size																																		
1001	10-bit data size																																		
1010	11-bit data size																																		
1011	12-bit data size																																		
1100	13-bit data size																																		
1101	14-bit data size																																		
1110	15-bit data size																																		
1111	16-bit data size																																		

11.4.3 控制寄存器1（CTRLR1）

- **Name:** Control Register 1
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6

预留

1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											

NDF

Bit	Name	R/W	Description
31:1 6	rsvd	-	
15:0	NDF	RW	<p>Number of Data Frames. When TMOD = 10 or TMOD = 11, this register field sets the number of data frames to be continuously received by the DW_apb_ssi.</p> <p>The DW_apb_ssi continues to receive serial data until the number of dataframes received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer.</p> <p>When the DW_apb_ssi is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the DW_apb_ssi is configured as a serial slave.</p>

11.4.4 使能寄存器（SSIENR）

- **Name:** SSI Enable Register
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	16
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	
预留															
1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0										
预留															SSI _E N

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	SSI_EN	RW	SSI Enable. Enables and disables all DW_apb_ssi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the DW_apb_ssi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

11.4.5 Microwire控制寄存器（MWCR）

- **Name:** Microwire Control Register
- **Size:** 32bits
- **Address Offset:** 0x0C
 - **Read/write access:** read/write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	16	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7		
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留													M	M	M	

	H S	D D	W M O D
--	--------	--------	------------------

Bit	Name	W/R	Description
31:3	rsvd	RO	预留
2	MHS	RW	<p>Microwire Handshaking. Relevant only when the DW_apb_ssi is configured as a serial-master device. When configured as a serial slave, this bit field has no functionality.</p> <p>Used to enable and disable the “busy/ready” handshaking interface for the Microwire protocol. When enabled, the DW_apb_ssi checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register.</p> <p>0: handshaking interface is disabled 1: handshaking interface is enabled</p>
1	MDD	RW	<p>Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the DW_apb_ssi MacroCell from the external serial device. When this bit is set to 1, the data word is transmitted from the DW_apb_ssi MacroCell to the external serial device.</p>
0	MWMOD	RW	<p>Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there</p>

			must be a control word for each data word that is transmitted or received. 0 – non-sequential transfer 1 – sequential transfer
--	--	--	--

11.4.6 从设备选择寄存器（SER）

- **Name: Slave Enable Register**
- **Size: 32bits**
- **Address Offset: 0x10**
- **Read/write access: read/write**

31:4	3:0
预留	SER

Bit	Name	W/R	Description
31:4	rsvd	RO	预留
3:0	SER	RW	Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_x_n] from the DW_apb_ssi master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set. 1: Selected 0: Not Selected

11.4.7 波特率寄存器（BAUDR）

- **Name: Baud Rate Select**

- **Size:** 32bits
- **Address Offset:** 0x14
- **Read/write access:** read/write

31:16	15:0
预留	SCKDV

Bit	Name	W/R	Description
31:16	rsvd	RO	<div style="border: 2px solid red; padding: 5px; display: inline-block;">0123456789</div>
15:0	SCKDV	W/R	

SCK / SCKDV
 2 到 65534
 开后，对该寄存器操作有效。

11.4.8 发送FIFO阈值寄存器（TXFTLR）

- **Name:** Transmit FIFO Threshold Level
- **Size:** 32bits
- **Address Offset:** 0x18
- **Read/Write Access:** Read/Write

31:4	3:0
预留	TXF TLR

Bit	Name	W/R	Description								
31:8	rsvd	RO	预留								
7:0	TXFTLR	W/R	<div>发送 FIFO 满中断阈值</div> <div>设定值参考下表：</div> <table><tr><td>0</td><td>发送 FIFO 中数据数量为 0 触发中断</td></tr><tr><td>1</td><td>发送 FIFO 中数据少于 1 个触发中断</td></tr><tr><td>2</td><td>发送 FIFO 中数据少于 2 个触发中断</td></tr><tr><td>3</td><td>发送 FIFO 中数据少于 3 个触发中断</td></tr></table>	0	发送 FIFO 中数据数量为 0 触发中断	1	发送 FIFO 中数据少于 1 个触发中断	2	发送 FIFO 中数据少于 2 个触发中断	3	发送 FIFO 中数据少于 3 个触发中断
0	发送 FIFO 中数据数量为 0 触发中断										
1	发送 FIFO 中数据少于 1 个触发中断										
2	发送 FIFO 中数据少于 2 个触发中断										
3	发送 FIFO 中数据少于 3 个触发中断										

			4	发送 FIFO 中数据少于 4 个触发中断
			5	发送 FIFO 中数据少于 5 个触发中断
			6	发送 FIFO 中数据少于 6 个触发中断
			7	发送 FIFO 中数据少于 7 个触发中断
			8	发送 FIFO 中数据少于 8 个触发中断
			9	发送 FIFO 中数据少于 9 个触发中断
			10	发送 FIFO 中数据少于 10 个触发中断
			11	发送 FIFO 中数据少于 11 个触发中断
			12	发送 FIFO 中数据少于 12 个触发中断
			13	发送 FIFO 中数据少于 13 个触发中断
			14	发送 FIFO 中数据少于 14 个触发中断
			15	发送 FIFO 中数据少于 15 个触发中断

11.4.9 接收FIFO阈值寄存器（RXFTLR）

- **Name: Receive FIFO Threshold Level**
- **Size: 32bits**
- **Address Offset: 0x1C**
- **Read/Write Access: Read/Write**

31:4	3:0
预留	RXF TLR

Bit	Name	W/R	Description
31:8	rsvd	RO	预留
7:0	RXFTLR	W/R	接收 FIFO 空中断阈值 设定值参考下表：

			0	接收 FIFO 中数据 1 个及以上触发中断
			1	接收 FIFO 中数据 2 个及以上触发中断
			2	接收 FIFO 中数据 3 个及以上触发中断
			3	接收 FIFO 中数据 4 个及以上触发中断
			4	接收 FIFO 中数据 5 个及以上触发中断
			5	接收 FIFO 中数据 6 个及以上触发中断
			6	接收 FIFO 中数据 7 个及以上触发中断
			7	接收 FIFO 中数据 8 个及以上触发中断
			8	接收 FIFO 中数据 9 个及以上触发中断
			9	接收 FIFO 中数据 10 个及以上触发中断
			10	接收 FIFO 中数据 11 个及以上触发中断
			11	接收 FIFO 中数据 12 个及以上触发中断
			12	接收 FIFO 中数据 13 个及以上触发中断
			13	接收 FIFO 中数据 14 个及以上触发中断
			14	接收 FIFO 中数据 15 个及以上触发中断
			15	接收 FIFO 中数据 16 个触发中断

11.4.10 发送FIFO数据量寄存器（TXFLR）

- **Name: Transmit FIFO Level Register**
- **Size: 32bits**
- **Address Offset: 0x20**
- **Read/Write Access: Read**

当向FIFO中写入数据时寄存器值增加，当I2C从FIFO中取数据时寄存器值减小。

31:4	3:0
预留	TXFLR

Bit	Name	W/ R	Description
31:4	rsvd	R O	预留
3:0	TXFLR	R	发送 FIFO 数据量 发送 FIFO 中包含的有效数据数

11.4.11 接收FIFO数据量寄存器（RXFLR）

- **Name: Receive FIFO Level Register**
- **Size: 32bits**
- **Address Offset: 0x20**
- **Read/Write Access: Read**

当向FIFO中写入数据时寄存器值增加，当I2C从FIFO中取数据时寄存器值减小。

31:4	3:0
预留	RXFLR

Bit	Name	W/ R	Description
31:4	rsvd	R O	预留
3:0	RXFLR	R	接收 FIFO 数据量

			接收 FIFO 中包含的有效数据数
--	--	--	-------------------

11.4.12 状态寄存器 (SR)

- **Name: Status Register**
- **Size: 32 bits**
- **Address Offset: 0x28**
- **Read/Write Access: Read**

31:7	6	5	4	3	2	1	0
预留	DC OL	TX E	RF F	RF NE	TF E	TF NF	BU SY

Bit	Name	W/ R	Description
31:7	rsvd	RO	预留
6	DCOL	RO	数据碰撞错误 0-无错误 1-传输数据碰撞错误 读寄存器清除该位。
5	TXE	RO	传输错误 0-无错误 1-传输错误 读寄存器清除该位。
4	RFF	RO	接收 FIFO 满 0-接收 FIFO 未滿 1-接收 FIFO 满 当 FIFO 未滿时，由硬件自动清“0”
3	RFNE	RO	接收 FIFO 未空 0-接收 FIFO 空 1-接收 FIFO 未空

			通过软件读 FIFO 清“0”
2	TFE	RO	发送 FIFO 空 0-发送 FIFO 未空 1-发送 FIFO 空 当 FIFO 未空时，由硬件自动清“0”
1	TFNF	RO	发送 FIFO 未空 0-发送 FIFO 满 1-发送 FIFO 未空 当 FIFO 满时，由硬件自动清“0”
0	BUSY	RO	忙状态 0-SPI 处于 EDLE 或关闭状态 1-SPI 处于 Activity 传输数据状态

11.4.13 中断屏蔽寄存器（IMR）

■ **Name: Interrupt Mask Register**

■ **Size: 32bits**

■ **Address Offset: 0x2C**

■ **Read/Write Access: read/write**

用于屏蔽或允许SPI各中断源。

31:6	5	4	3	2	1	0
预留	MSTI M	RXFI M	RXOI M	RXUI M	TXOI M	TXEI M

Bit	Name	W/R	Description
31:6	rsvd	RO	预留
5	MSTIM	RW	多主机竞争中断屏蔽 0-禁止多主机竞争中断 1-允许多主机竞争中断
4	RXFIM	RW	接收 FIFO 满中断屏蔽 0-禁止接收 FIFO 满中断

			1-允许接收 FIFO 满中断
3	RXOIM	RW	接收 FIFO 上溢中断屏蔽 0-禁止接收 FIFO 上溢中断 1-允许接收 FIFO 上溢中断
2	RXUIM	RW	接收 FIFO 下溢中断屏蔽 0-禁止接收 FIFO 下溢中断 1-允许接收 FIFO 下溢中断
1	TXOIM	RW	发送 FIFO 上溢中断屏蔽 0-禁止发送 FIFO 上溢中断 1-允许发送 FIFO 上溢中断
0	TXEIM	RW	发送 FIFO 空中断屏蔽 0-禁止发送 FIFO 空中断 1-允许发送 FIFO 空中断

11.4.14 中断状态寄存器 (ISR)

- **Name: Interrupt Status Register**
- **Size: 32bits**
- **Address Offset: 0x30**
- **Read/Write Access: read**

31:6	5	4	3	2	1	0
预留	MSTIS	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS

Bit	Name	W/R	Description
31:6	rsvd	RO	预留
5	SSTIS	RO	多主机竞争中断状态 0-未产生多主机竞争中断 1-产生多主机竞争中断
4	RXFIS	RO	接收 FIFO 满中断状态 0-未产生接收 FIFO 满中断 1-产生接收 FIFO 满中断
3	RXOIS	RO	接收 FIFO 上溢中断状态

			0-未产生接收 FIFO 上溢中断 1-产生接收 FIFO 上溢中断
2	RXUIS	RO	接收 FIFO 下溢中断状态 0-未产生接收 FIFO 下溢中断 1-产生接收 FIFO 下溢中断
1	TXOIS	RO	发送 FIFO 上溢中断状态 0-未产生发送 FIFO 上溢中断 1-产生发送 FIFO 上溢中断
0	TXEIS	RO	发送 FIFO 空中断状态 0-未产生发送 FIFO 空中断 1-产生发送 FIFO 空中断

11.4.15 原中断状态寄存器 (RISR)

- **Name: Raw Interrupt Status Register**
- **Size: 32bits**
- **Address Offset: 0x34**
- **Read/Write Access: read**

该寄存器状态值与中断状态寄存器(ISR)中不同在于,该寄存器的值不受中断屏蔽寄存器(IMR)控制。

31:6	5	4	3	2	1	0
预留	MSTI R	RXFI R	RXOI R	RXUI R	TXOI R	TXEI R

Bit	Name	W/R	Description
31:6	rsvd	RO	预留
5	MSTIR	RO	多主机竞争中断状态 0-未产生多主机竞争中断 1-产生多主机竞争中断
4	RXFIR	RO	接收 FIFO 满中断状态 0-未产生接收 FIFO 满中断

			1-产生接收 FIFO 满中断
3	RXOIR	RO	接收 FIFO 上溢中断状态 0-未产生接收 FIFO 上溢中断 1-产生接收 FIFO 上溢中断
2	RXUIR	RO	接收 FIFO 下溢中断状态 0-未产生接收 FIFO 下溢中断 1-产生接收 FIFO 下溢中断
1	TXOIR	RO	发送 FIFO 上溢中断状态 0-未产生发送 FIFO 上溢中断 1-产生发送 FIFO 上溢中断
0	TXEIR	RO	发送 FIFO 空中断状态 0-未产生发送 FIFO 空中断 1-产生发送 FIFO 空中断

11.4.16 发送FIFO上溢中断清除寄存器（TXOICR）

- **Name: Transmit FIFO Overflow Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x38**
- **Read/Write Access: read**

31:1	0
预留	TX OIC R

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	TXOICR	RO	发送 FIFO 上溢中断清除 对其读操作清除中断

11.4.17 接收FIFO上溢中断清除寄存器（RXOICR）

- **Name: Receive FIFO Overflow Interrupt Clear Register**

- **Size: 32bits**
- **Address Offset: 0x38**
- **Read/Write Access: read**

31:1	0
预留	RX OIC R

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	RXOICR	RO	接收 FIFO 上溢中断清除 对其读操作清除中断

11.4.18 接收FIFO下溢中断清除寄存器（RXUICR）

- **Name: Receive FIFO Underflow Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x40**
- **Read/Write Access: read**

31:1	0
预留	RX UIC R

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	RXUICR	RO	接收 FIFO 下溢中断清除 对其读操作清除中断

11.4.19 多主机竞争中断清除寄存器（MSTICR）

- **Name: Multi-Master Interrupt Clear Register**
- **Size: 32bits**

- **Address Offset: 0x44**
- **Read/Write Access: read**

31:1	0
预留	MS TIC R

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	MSTICR	RO	多主机竞争中断清除 对其读操作清除中断

11.4.20 全局中断清除寄存器（ICR）

- **Name: Interrupt Clear Register**
- **Size: 32bits**
- **Address Offset: 0x48**
- **Read/Write Access: read**

31:1	0
预留	I C R

Bit	Name	W/R	Description
31:1	rsvd	RO	预留
0	ICR	RO	全局中断清除 对其读操作清除以上 4 个中断状态

11.4.21 DMA控制寄存器（DMACR）

- **Name: DMA Control Register**
- **Size: 32 bits**
- **Address Offset: 0x4C**

■ **Read/Write Access: read/write**

对该寄存器的操作不受SPI使能的影响。

31:2	1	0
预留	TDMA E	RDM AE

Bit	Name	W/R	Description
31:2	rsvd	RO	预留
1	TDMAE	RW	发送 DMA 使能 0-发送 DMA 关闭 1-发送 DMA 打开
0	RDMAE	RW	接收 DMA 使能 0-接收 DMA 关闭 1-接收 DMA 打开

11.4.22 DMA发送数据阈值寄存器（DMATDLR）

■ **Name: DMA Transmit Data Level**

■ **Size:32bits**

■ **Address Offset: 0x50**

■ **Read/Write Access: Read/Write**

31:4	3:0
预留	DMAT DLR

Bit	Name	W/ R	Description
31: 4	rsvd	RO	预留
3:0	DMATDLR	W/ R	DMA 发送阈值 当发送 FIFO 中数据量等于或小于 DMA 发送阈值，SPI 会对 DMA 发出请求，请求 DMA 向 SPI

			发送 FIFO 中写入数据。																																
			设定值参考下表：																																
			<table><tr><td>0</td><td>发送 FIFO 中数据数量为 0</td></tr><tr><td>1</td><td>发送 FIFO 中数据少于等于 1 个</td></tr><tr><td>2</td><td>发送 FIFO 中数据少于等于 2 个</td></tr><tr><td>3</td><td>发送 FIFO 中数据少于等于 3 个</td></tr><tr><td>4</td><td>发送 FIFO 中数据少于等于 4 个</td></tr><tr><td>5</td><td>发送 FIFO 中数据少于等于 5 个</td></tr><tr><td>6</td><td>发送 FIFO 中数据少于等于 6 个</td></tr><tr><td>7</td><td>发送 FIFO 中数据少于等于 7 个</td></tr><tr><td>8</td><td>发送 FIFO 中数据少于等于 8 个</td></tr><tr><td>9</td><td>发送 FIFO 中数据少于等于 9 个</td></tr><tr><td>10</td><td>发送 FIFO 中数据少于等于 10 个</td></tr><tr><td>11</td><td>发送 FIFO 中数据少于等于 11 个</td></tr><tr><td>12</td><td>发送 FIFO 中数据少于等于 12 个</td></tr><tr><td>13</td><td>发送 FIFO 中数据少于等于 13 个</td></tr><tr><td>14</td><td>发送 FIFO 中数据少于等于 14 个</td></tr><tr><td>15</td><td>发送 FIFO 中数据少于等于 15 个</td></tr></table>	0	发送 FIFO 中数据数量为 0	1	发送 FIFO 中数据少于等于 1 个	2	发送 FIFO 中数据少于等于 2 个	3	发送 FIFO 中数据少于等于 3 个	4	发送 FIFO 中数据少于等于 4 个	5	发送 FIFO 中数据少于等于 5 个	6	发送 FIFO 中数据少于等于 6 个	7	发送 FIFO 中数据少于等于 7 个	8	发送 FIFO 中数据少于等于 8 个	9	发送 FIFO 中数据少于等于 9 个	10	发送 FIFO 中数据少于等于 10 个	11	发送 FIFO 中数据少于等于 11 个	12	发送 FIFO 中数据少于等于 12 个	13	发送 FIFO 中数据少于等于 13 个	14	发送 FIFO 中数据少于等于 14 个	15	发送 FIFO 中数据少于等于 15 个
0	发送 FIFO 中数据数量为 0																																		
1	发送 FIFO 中数据少于等于 1 个																																		
2	发送 FIFO 中数据少于等于 2 个																																		
3	发送 FIFO 中数据少于等于 3 个																																		
4	发送 FIFO 中数据少于等于 4 个																																		
5	发送 FIFO 中数据少于等于 5 个																																		
6	发送 FIFO 中数据少于等于 6 个																																		
7	发送 FIFO 中数据少于等于 7 个																																		
8	发送 FIFO 中数据少于等于 8 个																																		
9	发送 FIFO 中数据少于等于 9 个																																		
10	发送 FIFO 中数据少于等于 10 个																																		
11	发送 FIFO 中数据少于等于 11 个																																		
12	发送 FIFO 中数据少于等于 12 个																																		
13	发送 FIFO 中数据少于等于 13 个																																		
14	发送 FIFO 中数据少于等于 14 个																																		
15	发送 FIFO 中数据少于等于 15 个																																		

11.4.23 DMA接收数据阈值寄存器（DMARDLR）

- **Name: DMA Receive Data Level**
- **Size: 32bits**
- **Address Offset: 0x54**
- **Read/Write Access: Read/Write**

31:4

3:0

预留	DMAR DLR
----	-------------

Bit	Name	W/ R	Description
31: 4	rsvd	RO	预留
3:0	DMARDLR	W/ R	DMA 接收阈值
			当接收 FIFO 中数据量等于或大于 DMA 发送阈值，SPI 会对 DMA 发出请求，请求 DMA 取出 SPI 中接收 FIFO 的数据。
			设定值参考下表：
			0接收 FIFO 中数据为 1 个及以上
			1接收 FIFO 中数据为 2 个及以上
			2接收 FIFO 中数据为 3 个及以上
			3接收 FIFO 中数据为 4 个及以上
			4接收 FIFO 中数据为 5 个及以上
			5接收 FIFO 中数据为 6 个及以上
			6接收 FIFO 中数据为 7 个及以上
			7接收 FIFO 中数据为 8 个及以上
			8接收 FIFO 中数据为 9 个及以上
			9接收 FIFO 中数据为 10 个及以上
			10接收 FIFO 中数据为 11 个及以上
			11接收 FIFO 中数据为 12 个及以上
			12接收 FIFO 中数据为 13 个及以上
			13接收 FIFO 中数据为 14 个及以上
14接收 FIFO 中数据为 15 个及以上			
1接收 FIFO 中数据为 16 个			

			5	
--	--	--	---	--

11.4.24 数据寄存器（DR）

- **Name: Data Register**
- **Size: 32bits**
- **Address Offset: 0x60**
- **Read/Write Access: read/write**

数据寄存器是1个16bit宽的读写buffer。当对其进行读操作时，数据通过接收FIFO取出。当对其进行写操作时，数据被写入发送FIFO中。只有SSI_EN = 1时才能进行写操作。

31:16	15:0
预留	DR

Bit	Name	W/R	Description
31:16 6	rsvd	RO	预留
15:0	DR	RW	SPI 数据寄存器 写数据时数据必须为右对齐数据，读取数据时数据自动右对齐。 Read = 接收数据 FIFO Write = 发送数据 FIFO

11.4.25 接收采样延迟寄存器（RX_SAMPLE_DLY）

- **Name: Rx Sample Delay Register**
- **Size: 32bits**
- **Address Offset: 0xfc**
- **Read/Write Access: read/write**

该寄存器用于在标准接收采样时间点向后延迟采样点时间。延迟时间以PCLK周期为单位。在SPI使能后可对该寄存器进行读写操作。

31:8

7:0

预留	RSD
----	-----

Bit	Name	W/R	Description
31:8	rsvd	RO	预留
7:0	RSD	RW	采样延迟寄存器 该寄存器用于在标准接收采样时间点向后延迟采样点时间。延迟时间以 PCLK 周期为单位。

11.4.26 其他相关寄存器

SPI工作模式配置寄存器（SPI_SLAVE_SEL）寄存器相关信息详见5.4.13。

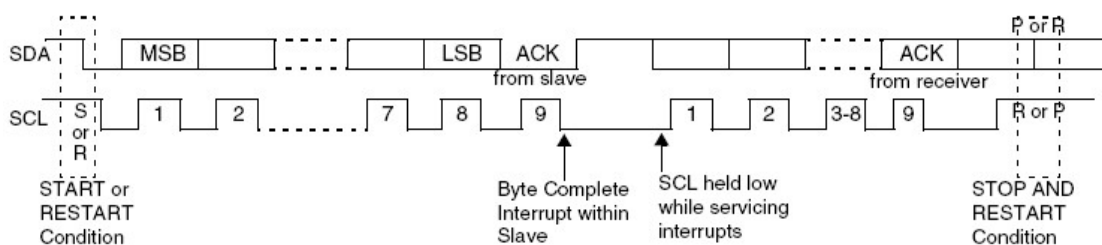
12 两线式串行总线（IIC）

12.1 IIC简介

由philips公司开发的两线式串行总线，用于连接CPU及其外设。优点：简单性和有效性。
 两线式：串行数据（SDA）和串行时钟（SCL）占用电路板空间和芯片管脚数量都非常小。支持多个master，任何能够发送或者接收的设备都可以成为master。被寻址的设备被称为slave。

12.2 I2C总线时序

-3 Data transfer on the I2C Bus



一次完整的数据传输图示：addr phase+data phase

SCL和SDA默认值为高，在SCL为高电平时，SDA不允许改变电平值

S（start）：在SCL为高时，SDA由1→0，表示master发起本次传输

P（stop）：SCL为高时，SDA由0→1，表示master终止本次传输

R（restart）：在一次start之后又开始一次start

12.2.1 I2C总线传输过程

Addr phase

地址模式分为7bit和10bit模式

传输地址从MSB开始，发送的地址即为为slave设备所分配的地址

如果地址配对正确，Slave拉低SDA，作为对master的ACK。准备开始接下来的数据传输。

如果地址接收出错，Slave不拉低SDA，即为NAK

Data phase:

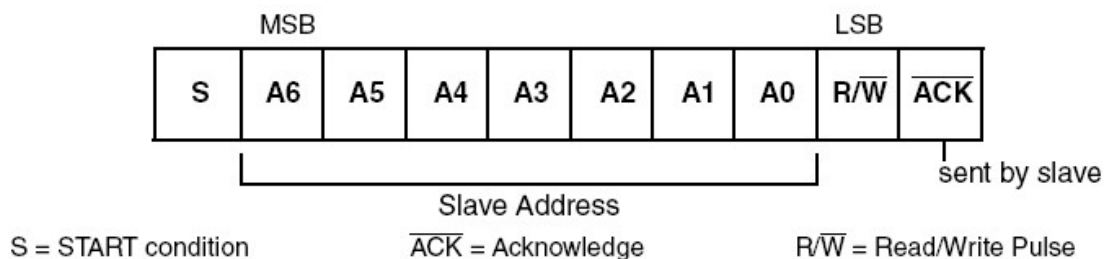
传输数据从MSB开始

每正确传送1byte， Slave拉低SDA，作为对master的ACK。

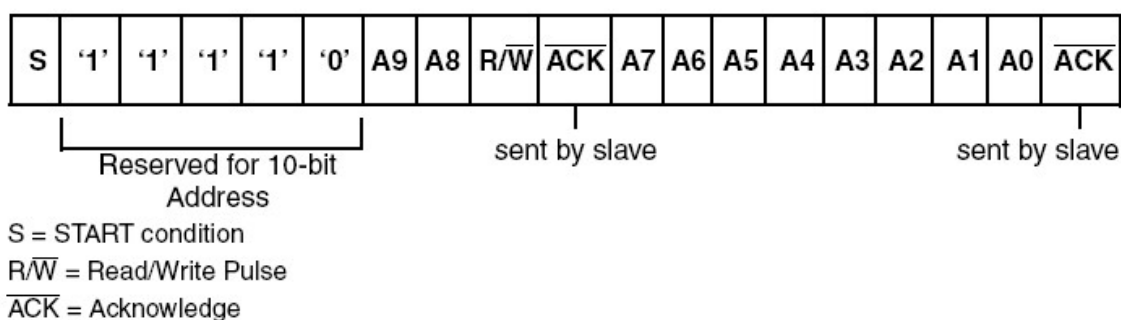
如果数据接收出错，Slave不拉低SDA，即为NAK。

12.2.2 I2C总线地址传输格式

5 7-bit Address Format



6 10-bit Address Format



注：10bit地址传输的2nd byte的开始，会有1次restart，所以10bit模式时，应将restart_en置为

1

12.2.3 I2c总线地址的reserved

General call: 7' h00

广播呼叫地址：用来寻址链接到I2C总线上的每个设备

如果设备不须从广播帧中得到数据，可以回应NAK

广播呼叫发送的数据均为写操作，不允许有rd操作

发送完会产生对应的general call中断

Start byte : 7'h01

起始字节，slave不予回应

作为慢速CPU的起始过程

发送完会产生对应的start byte中断

12.2.4 I2C总线的速度类型

Master根据自身配置的scl_lcnt和scl_hcnt产生SCL时钟，从而决定传输速度

High speed 3.4Mb/s

在传输开头传送slve的hs设备码，该设备码不可重复

Fast Speed 400kb/s

standard speed 100kb/s

12.2.5 I2c总线的restart

Figure 3-16 Master Transmitter — Restart Bit of IC_DATA_CMD Is Set (IC_EMPTYFIFO_HOLD_MASTER_EN = 1)

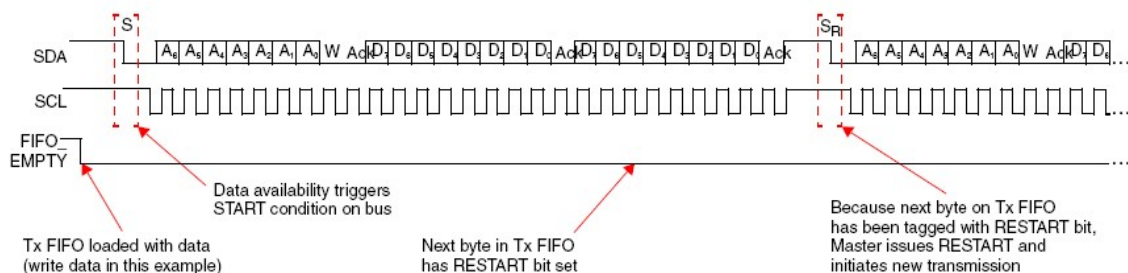
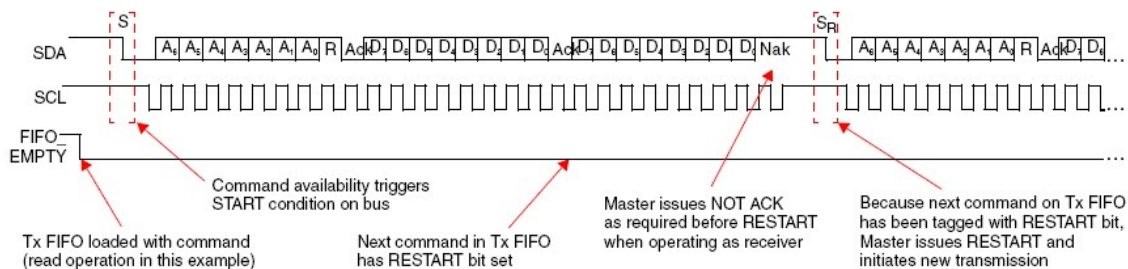


Figure 3-17 Master Receiver — Restart Bit of IC_DATA_CMD Is Set (IC_EMPTYFIFO_HOLD_MASTER_EN = 1)

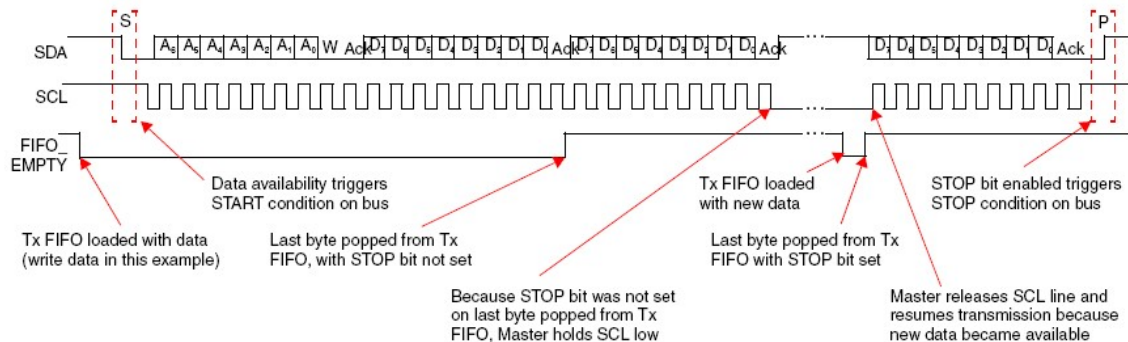


向tx fifo内写入带restart的ic_data_cmd,可以发起restart

在总线无传输且tx fifo为空的情况下，可以动态更新地址

12.2.6 I2C总线的占用

Figure 3-14 Master Transmitter — Tx FIFO Empties/STOP Generation If IC_EMPTYFIFO_HOLD_MASTER_EN = 1

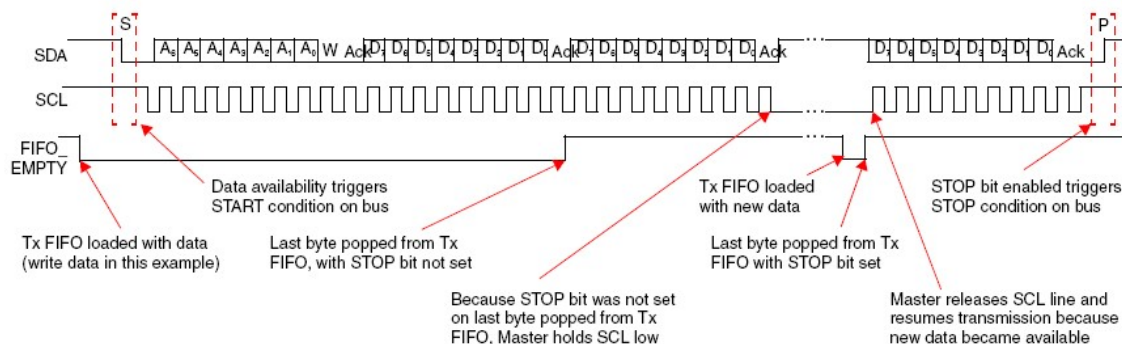


由master决定传输是否结束

当tx fifo为空且传输未结束时，master拉低SCL，继续占用i2c总线。直到FIFO再次写入数据。

12.2.7 I2C总线的占用

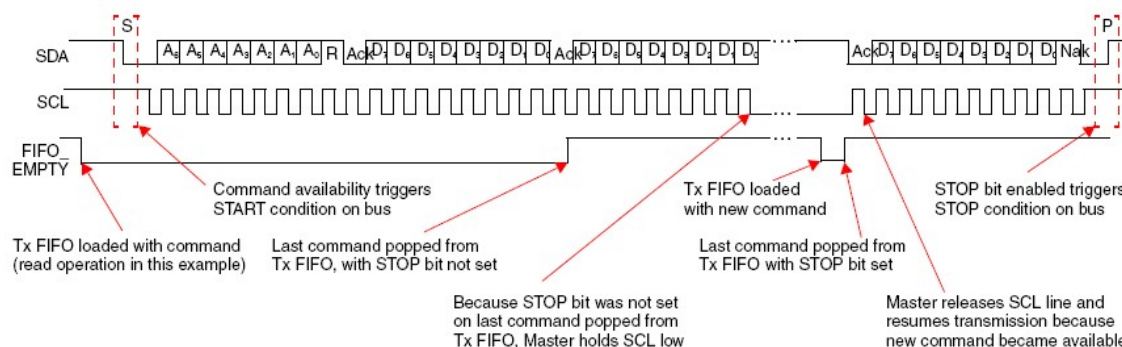
Figure 3-14 Master Transmitter — Tx FIFO Empties/STOP Generation If IC_EMPTYFIFO_HOLD_MASTER_EN = 1



由master决定传输是否结束

当tx fifo为空且传输未结束时，master拉低SCL，继续占用i2c总线。直到FIFO再次写入数据。

Figure 3-15 Master Receiver — Tx FIFO Empties/STOP Generation If IC_EMPTYFIFO_HOLD_MASTER_EN = 1

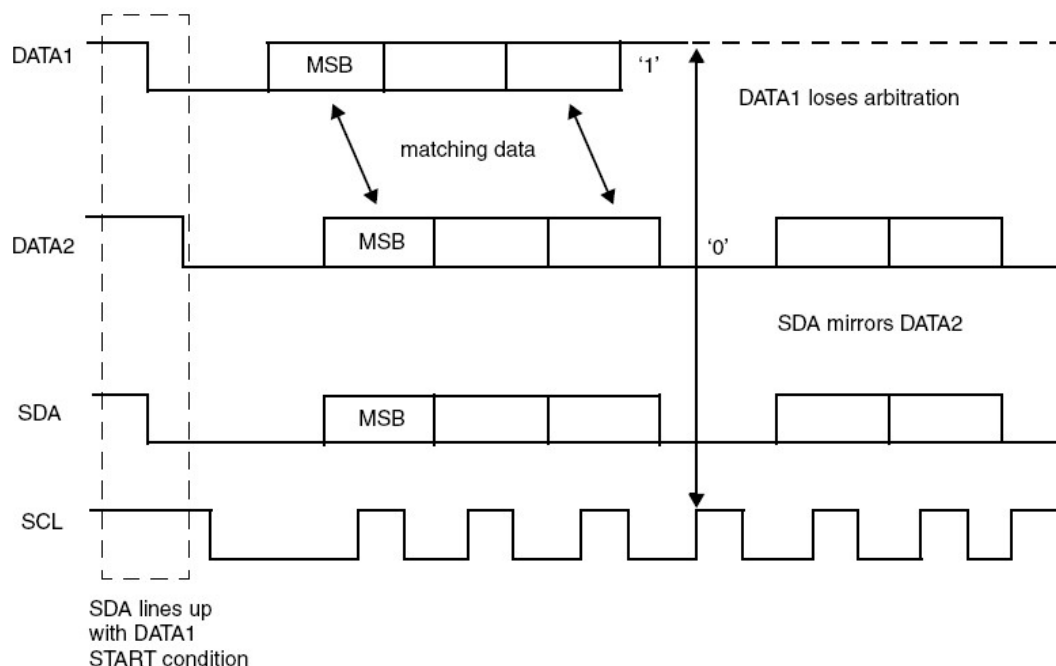


Slave传输数据，master回应ACK表示传输继续

Master回应NAK，决定传输是否结束

12.2.8 I2C总线的竞争

Figure 3-22 Multiple Master Arbitration



如果竞争在ADDR阶段未分胜负，则继续在data阶段进行
Hs设备不存在竞争现象，因为hs设备码是独一无二的

12.3 寄存器描述

12.3.1 地址映射表

I2C基地址列表

地址范围	基地址	外设	总线
0x5000_0000-0x5000_0FFF	0x5000_0000	I2C	APB0

I2C寄存器偏移地址列表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	IC_CON	32	0x00000000
0x04	IC_TAR	32	0x00000000
0x08	IC_SAR	32	0x0000FFFF

0x0C	IC_HS_MADDR	32	0x00000000
0x10	IC_DATA_CMD	32	0x00000000
0x14	IC_SS_SCL_HCNT	32	0x00000190
0x18	IC_SS_SCL_LCNT	32	0x000001d6
0x1c	IC_FS_SCL_HCNT	32	0x0000003c
0x20	IC_FS_SCL_LCNT	32	0x00000082
0x24	IC_HS_SCL_HCNT	32	0x00000006
0x28	IC_HS_SCL_LCNT	32	0x00000010
0x2c	IC_INTR_STAT	32	0x00000000
0x30	IC_INTR_MASK	32	0x000008FF
0x34	IC_RAW_INTR_STAT	32	0x00000000
0x38	IC_RX_TL	32	0x00000000
0x3C	IC_TX_TL	32	0x00000000
0x40	IC_CLR_INTR	32	0x00000000
0x44	IC_CLR_RX_UNDER	32	0x00000000
0x48	IC_CLR_RX_OVER	32	0x00000000
0x4C	IC_CLR_TX_OVER	32	0x00000000
0x50	IC_CLR_RD_REQ	32	0x00000000
0x54	IC_CLR_TX_ABRT	32	0x00000000
0x58	IC_CLR_RX_DONE	32	0x00000000
0x5C	IC_CLR_ACTIVITY	32	0x00000000
0x60	IC_CLR_STOP_DET	32	0x00000000
0x64	IC_CLR_START_DET	32	0x00000000
0x68	IC_CLR_GEN_CALL	32	0x00000000
0x6C	IC_ENABLE	32	0x00000000
0x70	IC_STATUS	32	0x00000006
0x74	IC_TXFLR	32	0x00000000
0x78	IC_RXFLR	32	0x00000000
0x7C	IC_SDA_HOLD	32	0x00000001
0x80	IC_TX_ABRT_SOURCE	32	0x00000000
0x84	IC_SLV_DATA_NACK_ONLY	32	0x00000000
0x88	IC_DMA_CR	32	0x00000000
0x8C	IC_DMA_TDLR	32	0x00000000
0x90	IC_DMA_RDLR	32	0x00000000
0x94	IC_SDA_SETUP	32	0x00000064
0x98	IC_ACK_GENERAL_CALL	32	0x00000001
0x9C	IC_ENABLE_STATUS	32	0x00000000

0xA0	IC_FS_SPKLEN	32	0xFFFFFFFF
0xA4	IC_HS_SPKLEN	32	0xFFFFFFFF
0xF4	预留	32	0x00000000
0xF8	预留	32	0x00000000
0xFC	预留	32	0x00000000

12.3.2 I2C控制寄存器（IC_CON）

- **Name:** I2C Control Register
- **Size:** 32bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							
7	6	5	4	3	2	1	0
预留	IC_SLAVE_DISABLE	IC_RESTART_EN	IC_10BIT_ADDR_MASTER_ONLY	IC_10BIT_ADDR_SLAVE	SPEED		MASTER_MODE

Bit	Name	R/W	Description
31:7	rsvd	RO	预留
6	IC_SLAVE_DISABLE	RW	This bit controls whether I2C has its slave disabled, which means once the preseln signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. You have the

			<p>choice of having the slave enabled or disabled after reset is applied,</p> <p>which means software does not have to configure the slave. By default, the slave is always enabled (in reset state as well). If you need to disable it after reset, set this bit to 1.</p> <p>If this bit is set (slave is disabled), DW_apb_i2c functions only as a master and does not perform any action that requires a slave.</p> <p>0: slave is enabled 1: slave is disabled</p> <p>Reset value: IC_SLAVE_DISABLE configuration parameter</p> <p>NOTE: Software should ensure that if this bit is written with '0,' then bit 0 should also be written with a '0'.</p>
5	IC_RESTART_EN	RW	<p>Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several DW_apb_i2c operations.</p> <p>0: disable 1: enable</p> <p>When the RESTART is disabled, the DW_apb_i2c master is incapable of performing the following functions:</p> <ul style="list-style-type: none"> ■ Sending a START BYTE ■ Performing any high-speed mode operation ■ Performing direction changes in combined format mode ■ Performing a read operation with a 10-bit address <p>By replacing RESTART condition followed by a STOP and a</p>

			<p>subsequent START condition, split operations are broken down into multiple DW_apb_i2c transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register.</p> <p>Reset value: IC_RESTART_EN configuration parameter</p>
4	IC_10BIT_ADDR_MASTER(rd_only)	RW	<p>If the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to “No” (0), this bit is named IC_10BITADDR_MASTER and controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master.</p> <p>If I2C_DYNAMIC_TAR_UPDATE is set to “Yes” (1), the function of this bit is handled by bit 12 of IC_TAR register, and becomes a read-only copy called IC_10BITADDR_MASTER_rd_only.</p> <p>0: 7-bit addressing 1: 10-bit addressing</p> <p>Dependencies: If I2C_DYNAMIC_TAR_UPDATE = 1, then this bit is read-only. If I2C_DYNAMIC_TAR_UPDATE = 0, then this bit can be read or write.</p> <p>Reset value: IC_10BITADDR_MASTER configuration parameter</p>
3	IC_10BIT_ADDR_SLAVE	RW	<p>When acting as a slave, this bit controls whether the DW_apb_i2c responds to 7- or 10-bit addresses.</p> <p>0: 7-bit addressing. The DW_apb_i2c ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the IC_SAR register are compared.</p> <p>1: 10-bit addressing. The DW_apb_i2c responds to only 10-bit</p>

			addressing transfers that match the full 10 bits of the IC_SAR register. Reset value: IC_10BITADDR_SLAVE configuration parameter
2:1	SPEED	RW	These bits control at which speed the DW_apb_i2c operates; its setting is relevant only if one is operating the DW_apb_i2c in master mode. Hardware protects against illegal values being programmed by software. This register should be programmed only with a value in the range of 1 to IC_MAX_SPEED_MODE; otherwise, hardware updates this register with the value of IC_MAX_SPEED_MODE. ■ 1: standard mode (0 to 100 kbit/s) ■ 2: fast mode (≤ 400 kbit/s) ■ 3: high speed mode (≤ 3.4 Mbit/s) Reset value: IC_MAX_SPEED_MODE configuration
0	MASTER_MODE	RW	This bit controls whether the DW_apb_i2c master is enabled. 0: master disabled 1: master enabled Reset value: IC_MASTER_MODE configuration parameter NOTE: Software should ensure that if this bit is written with '1,' then bit 6 should also be written with a '1'.

Certain combinations of the IC_SLAVE_DISABLE (bit 6) and MASTER_MODE (bit 0) result in a configuration error. Next table lists the states that result from the combinations of these two bits.

IC_SLAVE_DISABLE (IC_CON[6])	MASTER_MODE IC_CON[0]	Stat
0	1	Slave Device
0	1	Config Error
1	0	Config Error

1	1	Master Device
---	---	---------------

12.3.3 I2C目标地址寄存器（IC_TAR）

- **Name:** I2C Target Address Register
- **Size:** 32bits
- **Address Offset:** 0x00
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留			IC_1 0BIT ADD R_M AST ER	SPEC IAL	GC_ OR_S TAR T	IC_TAR	
7	6	5	4	3	2	1	0
IC_TAR							

Bit	Name	R/W	Description
31:13	rsvd	RO	预留
12	IC_10BIT ADDR_M ASTER	RW	This bit controls whether the DW_apb_i2c starts its transfers in 7-or 10-bit addressing mode when acting as a master. ■ 0: 7-bit addressing ■ 1: 10-bit addressing Dependencies: This bit exists in this register only if the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to “Yes” (1). Reset value: IC_10BITADDR_MASTER configuration

			parameter
11	SPECIAL	RW	<p>This bit indicates whether software performs a General Call or START BYTE command.</p> <p>■ 0: ignore bit 10 GC_OR_START and use IC_TAR normally</p> <p>■ 1: perform special I2C command as specified in GC_OR_START bit</p> <p>Reset value: 0x0</p>
10	GC_OR_START	RW	<p>If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the DW_apb_i2c.</p> <p>■ 0: General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The DW_apb_i2c remains in General Call mode until the SPECIAL bit value (bit 11) is cleared.</p> <p>■ 1: START BYTE</p> <p>Reset value: 0x0</p>
9:0	IC_TAR	RW	<p>This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits.</p> <p>Reset value: IC_DEFAULT_TAR_SLAVE_ADDR configuration parameter</p> <p>If the IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not</p>

			feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.
--	--	--	---

12.3.4 I2C从机地址寄存器（IC_SAR）

- **Name:** I2C Slave Address Register
- **Size:** 32bits
- **Address Offset:** 0x08
- **Read/write access:** read/write

31:13	12:0
预留	IC_SAR

Bit	Name	R/W	Description
31:13 3	rsvd	RO	预留
12:0	IC_SAR	RW	<p>The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>Note</p> <p>The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the IC_SAR or IC_TAR to a reserved value. Refer to Table 3-1 on page 37 for a complete list of these reserved values.</p> <p>Reset value: IC_DEFAULT_SLAVE_ADDR</p>

			configuration parameter
--	--	--	-------------------------

12.3.5 I2C高速主机模式地址码寄存器（IC_HS_MADDR）

- **Name:** I2C High Speed Master Mode Code Address Register
- **Size:** 32bits
- **Address Offset:** 0x0C
- **Read/write access:** read/write

31:3	2:0
预留	IC_HS_ MAR

Bit	Name	R/W	Description
31:3	rsvd	RO	预留
2:0	IC_HS_M AR	RW	<p>This bit field holds the value of the I2C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight highspeed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7. This register goes away and becomes read-only returning 0's if the IC_MAX_SPEED_MODE configuration parameter is set to either Standard (1) or Fast (2).</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>Reset value: IC_HS_MASTER_CODE configuration parameter</p>

12.3.6 I2C数据命令寄存器 (IC_DATA_CMD)

■ **Name:** I2C Rx/Tx Data Buffer and Command Register; this is the register the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO

■ **Size:** 32bits

■ **Address Offset:** 0x10

■ **Read/write access:** read/write

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

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
1			
10	RESTART	W	<p>This bit controls whether a RESTART is issued before the byte is sent or received.</p> <p>This bit is available only if IC_EMPTYFIFO_HOLD_MASTER_EN is configured to 1.</p> <p>■ 1 - If IC_RESTART_EN is 1, a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed by a START is issued instead.</p> <p>■ 0 - If IC_RESTART_EN is 1, a RESTART is issued only if the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed</p>

			by a START is issued instead.
9	STOP	W	<p>This bit controls whether a STOP is issued after the byte is sent or received. This bit is available only if IC_EMPTYFIFO_HOLD_MASTER_EN is configured to 1.</p> <p>■ 1 – STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus.</p> <p>■ 0 – STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.</p>
8	CMD	W	<p>This bit controls whether a read or a write is performed. This bit does not control the direction when the DW_apb_i2c acts as a slave. It controls only the direction when it acts as a master.</p> <p>■ 1 = Read</p> <p>■ 0 = Write</p> <p>When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a “don’t care” because writes to this register are not required. In slave-transmitter mode, a “0” indicates that the data in IC_DATA_CMD is to be transmitted.</p> <p>When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a</p>

			TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared. If a “1” is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs. Reset value: 0x0
7:0	DAT	RW	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the DW_apb_i2c. However, when you read this register, these bits return the value of data received on the DW_apb_i2c interface. Reset value: 0x0

12.3.7 I2C标准时钟高速计数寄存器 (IC_SS_SCL_HCNT)

- **Name:** Standard Speed I2C Clock SCL High Count Register
- **Size:** 32bits
- **Address Offset:** 0x14
- **Read/write access:** read/write

31:16

15:0

预留	IC_SS_SCL_HCNT
----	----------------

Bit	Name	R/W	Description
31:16	rsvd	RO	预留
15:0	IC_SS_SCL_HCNT	RW	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. For more information, refer to “IC_CLK Frequency Configuration” on page 56. This register can be written only when the I2C interface is

			<p>disabled which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>NOTE: This register must not be programmed to a value higher than 65525, because DW_apb_i2c uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.</p> <p>Reset value: IC_SS_SCL_HIGH_COUNT configuration parameter</p> <p>1Read-only if IC_HC_COUNT_VALUES = 1.</p>
--	--	--	---

12.3.8 I2C标准时钟低速计数寄存器 (IC_SS_SCL_LCNT)

- **Name:** Standard Speed I2C Clock SCL Low Count Register
- **Size:** 32bits
- **Address Offset:** 0x18
- **Read/write access:** read/write

31:16	15:0
预留	IC_SS_SCL_LCNT

Bit	Name	R/W	Description
31:1 6	rsvd	RO	预留
15:0	IC_SS_SCL_LCNT	RW	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. For more information, refer to “IC_CLK Frequency Configuration” on page 56.</p> <p>This register can be written only when the I2C interface is disabled which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set. For designs with APB_DATA_WIDTH = 8, the order of programming is important to ensure the correct operation of DW_apb_i2c. The lower byte must be programmed first, and then the upper byte is programmed.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Reset value: IC_SS_SCL_LOW_COUNT configuration parameter</p> <p>1Read-only if IC_HC_COUNT_VALUES = 1.</p>

12.3.9 I2C快速时钟高速计数寄存器（IC_FS_SCL_HCNT）

- **Name:** Fast Speed I2C Clock SCL High Count Register
- **Size:** 32bits
- **Address Offset:** 0x1C
- **Read/write access:** read/write

31:16

15:0

预留	IC_FS_SCL_HCNT
----	----------------

Bit	Name	R/W	Description
31:1 6	rsvd	RO	预留
15:0	IC_SS_SCL_HCNT	RW	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to “IC_CLK Frequency Configuration” on page 56.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH == 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Reset value: IC_FS_SCL_HIGH_COUNT configuration parameter</p> <p>1Read-only if IC_HC_COUNT_VALUES = 1.</p>

12.3.10 I2C快速时钟低速计数寄存器 (IC_FS_SCL_LCNT)

- **Name:** Fast Speed I2C Clock SCL Low Count Register
- **Size:** 32bits
- **Address Offset:** 0x20
- **Read/write access:** read/write

31:16

15:0

预留	IC_FS_SCL_LCNT
----	----------------

Bit	Name	R/W	Description
31:16 6	rsvd	RO	预留
15:0	IC_SS_SCL_LCNT	RW	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to “IC_CLK Frequency Configuration” on page 56.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the</p>

			<p>correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8.</p> <p>When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only.</p> <p>Reset value: IC_FS_SCL_LOW_COUNT configuration parameter</p> <p>1Read-only if IC_HC_COUNT_VALUES = 1.</p>
--	--	--	--

12.3.11 I2C高速时钟高速计数寄存器 (IC_HS_SCL_HCNT)

- **Name:**High Speed I2C Clock SCL High Count Register
- **Size:** 32bits
- **Address Offset:** 0x24
- **Read/write access:** read/write

31:16	15:0
预留	IC_FS_SCL_HCNT

Bit	Name	R/W	Description
31:16	rsvd	RO	预留
15:0	IC_SS_SCL_HCNT	RW	<p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high period count for high speed. For more information, refer to “IC_CLK Frequency Configuration” on page 56.</p> <p>The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns.</p> <p>This register goes away and becomes read-only returning</p>

			<p>0s if IC_MAX_SPEED_MODE != high. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the DW_apb_i2c. The lower byte must be programmed first. Then the upper byte is programmed. When the configuration parameter IC_HC_COUNT_VALUES is set to 1, this register is read only. Reset value: IC_HS_SCL_HIGH_COUNT configuration parameter 1Read-only if IC_HC_COUNT_VALUES = 1.</p>
--	--	--	---

12.3.12 I2C高速时钟低速计数寄存器 (IC_HS_SCL_LCNT)

- **Name:** High Speed I2C Clock SCL Low Count Register
- **Size:** 32bits
- **Address Offset:** 0x28
- **Read/write access:** read/write

31:16	15:0
预留	IC_FS_SCL_LCNT

Bit	Name	R/W	Description
31:16 6	rsvd	RO	预留
15:0	IC_SS_SCL_L_LCNT	RW	This register must be set before any I2C bus transaction can take place to

			<p>ensure proper I/O timing. This register sets the SCL clock low period count for high speed. For more information, refer to “IC_CLK Frequency Configuration” on page 56.</p> <p>The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns.</p> <p>This register goes away and becomes read-only returning 0s if <code>IC_MAX_SPEED_MODE != high</code>.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to <code>IC_ENABLE[0]</code> being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with <code>APB_DATA_WIDTH == 8</code> the order of programming is important to ensure the correct operation of the <code>DW_apb_i2c</code>. The lower byte must be programmed first. Then the upper byte is programmed. If the value is less than 8 then the count value gets changed to 8.</p> <p>When the configuration parameter <code>IC_HC_COUNT_VALUES</code> is set to 1, this register is read only.</p> <p>Reset value: <code>IC_HS_SCL_LOW_COUNT</code> configuration parameter</p> <p>1Read-only if <code>IC_HC_COUNT_VALUES = 1</code>.</p>
--	--	--	---

12.3.13 I2C中断状态寄存器 (IC_INTR_STAT)

■ **Name:** I2C Interrupt Status Register

■ **Size:** 32bits

■ Address Offset: 0x2C

■ Read/write access: Read

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留				R_G EN_ CAL L	R_ST ART_ DET	R_ST OP_ DET	R_A CTIV ITY
7	6	5	4	3	2	1	0
R_R X_D ONE	R_T X_A BRT	R_R D_RE Q	R_T X_E MPT Y	R_T X_O VER	R_R X_FU LL	R_R X_O VER	R_R X_U NDE R

Bit	Name	R/ W	Description
31:1 2	rsvd	R O	预留
11	R_GEN_CA LL	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
10	R_START_ DET	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
9	R_STOP_D ET	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
8	R_ACTIVIT Y	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0

7	R_RX_DON E	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
6	R_TX_ABR T	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
5	R_RD_REQ	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
4	R_TX_EMPTY	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
3	R_TX_OVERR	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
2	R_RX_FULL	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
1	R_RX_OVERR	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0
0	R_RX_UNDE R	R O	See “IC_RAW_INTR_STAT” for a detailed description of these bits. Reset value: 0x0

12.3.14 I2C屏蔽状态寄存器 (IC_INTR_MASK)

■ **Name:** I2C Interrupt Mask Register

■ **Size:** 32bits

■ **Address Offset:** 0x30

■ **Read/write access:** Read/Write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留				M_G EN_ CAL L	M_S TAR T_DE T	M_S TOP_ DET	M_A CTIV ITY
7	6	5	4	3	2	1	0
M_R X_D ONE	M_T X_A BRT	M_R D_RE Q	M_T X_E MPT Y	M_T X_O VER	M_R X_FU LL	M_R X_O VER	M_R X_U NDE R

Bit	Name	R/ W	Description
31:1 2	rsvd	R O	预留
11	M_GEN_C ALL	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
10	M_START_ DET	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
9	M_STOP_D ET	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
8	M_ACTIVI TY	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff

7	M_RX_DONE	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
6	M_TX_ABORT	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
5	M_RD_REQ	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
4	M_TX_EMPTY	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
3	M_TX_OVER	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
2	M_RX_FULL	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
1	M_RX_OVER	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff
0	M_RX_UNDER	R W	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register. Reset value: 12'h8ff

12.3.15 I2C原中断状态寄存器 (IC_RAW_INTR_STAT)

■ **Name:** I2C Raw Interrupt Status Register

■ **Size:** 32bits

■ Address Offset: 0x34

■ Read/write access: Read

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留				GEN _CA LL	STA RT_D ET	STOP _DET	ACTI VITY
7	6	5	4	3	2	1	0
RX_ DON E	TX_ ABR T	RD_ REQ	TX_E MPT Y	TX_ OVE R	RX_F ULL	RX_ OVE R	RX_ UND ER

Bit	Name	R/ W	Description
31:1 2	rsvd	R O	预留
11	GEN_CALL	R O	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling DW_apb_i2c or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. DW_apb_i2c stores the received data in the Rx buffer. Reset value: 0x0
10	START_DE T	R O	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
9	STOP_DET	R O	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether DW_apb_i2c is operating in slave or master mode. Reset value: 0x0
8	ACTIVITY	R	This bit captures DW_apb_i2c activity and stays set until

		O	<p>it is cleared. There are four ways to clear it:</p> <ul style="list-style-type: none"> ■ Disabling the DW_apb_i2c ■ Reading the IC_CLR_ACTIVITY register ■ Reading the IC_CLR_INTR register ■ System reset <p>Once this bit is set, it stays set unless one of the four methods is used to clear it.</p> <p>Even if the DW_apb_i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus.</p> <p>Reset value: 0x0</p>
7	RX_DONE	R O	<p>When the DW_apb_i2c is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.</p> <p>Reset value: 0x0</p>
6	TX_ABORT	R O	<p>This bit indicates if DW_apb_i2c, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a “transmit abort”.</p> <p>When this bit is set to 1, the IC_TX_ABORT_SOURCE register indicates the reason why the transmit abort takes places.</p> <p>NOTE: The DW_apb_i2c flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register IC_CLR_TX_ABORT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface.</p> <p>Reset value: 0x0</p>
5	RD_REQ	R O	<p>This bit is set to 1 when DW_apb_i2c is acting as a slave and another I2C master</p>

			<p>is attempting to read data from DW_apb_i2c. The DW_apb_i2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred.</p> <p>The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register.</p> <p>Reset value: 0x0</p>
4	TX_EMPTY	RO	<p>This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When IC_ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines.</p> <p>When there is no longer activity, then with ic_en=0, this bit is set to 0.</p> <p>Reset value: 0x0</p>
3	TX_OVER	RO	<p>Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Reset value: 0x0</p>
2	RX_FULL	RO	<p>Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by</p>

			<p>hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once IC_ENABLE[0] is set to 0, regardless of the activity that continues.</p> <p>Reset value: 0x0</p>
1	RX_OVER	RO	<p>Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Reset value: 0x0</p>
0	RX_UNDE R	RO	<p>Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> <p>Reset value: 0x0</p>

12.3.16 I2C接收阈值寄存器 (IC_RX_TL)

- **Name:** I2C Receive FIFO Threshold Register
- **Size:** 32bits
- **Address Offset:** 0x38
- **Read/write access:** read/write

31:8	7:0
预留	RX_TL

Bit	Name	R/W	Description
31:8	rsvd	RO	预留
7:0	RX_TL	RW	<p>Receive FIFO Threshold Level</p> <p>Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer.</p> <p>A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.</p> <p>Reset value: IC_RX_TL configuration parameter</p>

12.3.17 I2C发送阈值寄存器 (IC_TX_TL)

- **Name:** I2C Transmit FIFO Threshold Register
- **Size:** 32bits
- **Address Offset:** 0x3C
- **Read/write access:** read/write

31:8	7:0
预留	TX_TL

Bit	Name	R/W	Description
31:8	rsvd	RO	预留
7:0	TX_TL	RW	<p>Transmit FIFO Threshold Level</p> <p>Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the</p>

			maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries. Reset value: IC_TX_TL configuration parameter
--	--	--	---

12.3.18 I2C中断清除寄存器 (IC_CLR_INTR)

- **Name:** Clear Combined and Individual Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x40
- **Read/write access:** read

31:1	0
预留	CLR_I NTR

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_INT R	RO	Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Reset value: 0x0

12.3.19 I2CRX_UNDER中断清除寄存器 (IC_CLR_RX_UNDER)

- **Name:** Clear RX_UNDER Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x44

- **Read/write access:** read

31:1	0
预留	CLR_RX_UNDER

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_RX_UNDER	RO	Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.20 I2CRX_OVER中断清除寄存器 (IC_CLR_RX_OVER)

- **Name:** Clear RX_OVER Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x48
- **Read/write access:** read

31:1	0
预留	CLR_RX_OVER

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_RX_OVER	RO	Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.21 I2CTX_OVER中断清除寄存器 (IC_CLR_TX_OVER)

- **Name:** Clear TX_OVER Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x4C
- **Read/write access:** read

31:1	0
------	---

预留	CLR_TX_O ER
----	----------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_TX_ OVER	RO	Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.22 I2CTX_OVER中断清除寄存器 (IC_CLR_RD_REQ)

- **Name:** Clear RD_REQ Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x50
- **Read/write access:** read

31:1

0

预留	CLR_RD_RE Q
----	----------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_RD_ REQ	RO	Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.23 I2CTX_ABRT中断清除寄存器 (IC_CLR_TX_ABRT)

- **Name:** Clear TX_ABRT Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x54
- **Read/write access:** read

31:1

0

预留	CLR_TX_AB RT
----	-----------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_TX_ABRT	RO	Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. Reset value: 0x0

12.3.24 I2CTX_ABRT中断清除寄存器 (IC_CLR_RX_DONE)

- **Name:** Clear RX_DONE Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x58
- **Read/write access:** read

31:1	0
预留	CLR_RX_DONE

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_RX_DONE	RO	Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.25 I2CACTIVITY中断清除寄存器 (IC_CLR_ACTIVITY)

- **Name:** Clear ACTIVITY Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x5C
- **Read/write access:** read

31:1	0
------	---

预留	CLR_ ACTIVITY
----	------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_ ACTIVITY	RO	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.26 I2CSTOP_DET中断清除寄存器（IC_CLR_STOP_DET）

- **Name:** Clear STOP_DET Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x60
- **Read/write access:** read

31:10

预留	CLR_ STOP_DET
----	------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_STOP_DET	RO	Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.27 I2CSTART_DET中断清除寄存器（IC_CLR_START_DET）

- **Name:** Clear START_DET Interrupt Register
- **Size:** 32bits
- **Address Offset:** 0x64

■ **Read/write access:** read

31:1

0

预留	CLR_ START_DET
----	-------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_ START_D ET	RO	Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.28 I2CGEN_CALL中断清除寄存器 (IC_CLR_GEN_CALL)

■ **Name:** Clear GEN_CALL Interrupt Register

■ **Size:** 32bits

■ **Address Offset:** 0x68

■ **Read/write access:** read

31:1

0

预留	CLR_ GEN_CALL
----	------------------

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	CLR_ GEN_CA LL	RO	Read this register to clear the GEN_CALL interrupt (bit 11) of IC_RAW_INTR_STAT register. Reset value: 0x0

12.3.29 I2C使能寄存器 (IC_ENABLE)

■ **Name:** I2C Enable Register

■ **Size:** 32bits

■ **Address Offset:** 0x6C

■ **Read/write access:** Read/Write

31:2

1

0

预留	ABO RT	ENA BLE
----	-----------	------------

Bit	Name	R/W	Description
31:2	rsvd	RO	预留
1	ABORT	RW	<p>When set, the controller initiates the transfer abort.</p> <ul style="list-style-type: none"> ■ 0: ABORT not initiated or ABORT done ■ 1: ABORT operation in progress <p>The software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation.</p> <p>For a detailed description on how to abort I2C transfers, refer to “Aborting I2C Transfers” on page 54.</p> <p>Reset value: 0x0</p>
0	ENABLE	RW	<p>Controls whether the DW_apb_i2c is enabled.</p> <ul style="list-style-type: none"> ■ 0: Disables DW_apb_i2c (TX and RX FIFOs are held in an erased state) ■ 1: Enables DW_apb_i2c <p>Software can disable DW_apb_i2c while it is active. However, it is important that care be taken to ensure that DW_apb_i2c is disabled properly. A recommended procedure is described in “Disabling DW_apb_i2c” on page 53.</p> <p>When DW_apb_i2c is disabled, the following occurs:</p> <ul style="list-style-type: none"> ■ The TX FIFO and RX FIFO get flushed. ■ Status bits in the IC_INTR_STAT register are still active until DW_apb_i2c goes into

			<p>IDLE state.</p> <p>If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the DW_apb_i2c stops the current transfer at the end of the current byte and does not acknowledge the transfer.</p> <p>In systems with asynchronous pclk and ic_clk when IC_CLK_TYPE parameter set to asynchronous (1), there is a two ic_clk delay when enabling or disabling the DW_apb_i2c.</p> <p>For a detailed description on how to disable DW_apb_i2c, refer to “Disabling DW_apb_i2c” on page 53.</p> <p>Reset value: 0x0</p>
--	--	--	---

12.3.30 I2C状态寄存器 (IC_STATUS)

- **Name:** I2C Status Register
- **Size:** 32bits
- **Address Offset:** 0x70
- **Read/write access:** read

This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt. When the I2C is disabled by writing 0 in bit 0 of the IC_ENABLE register:

■ Bits 1 and 2 are set to 1

■ Bits 3 and 4 are set to 0

When the master or slave state machines goes to idle and ic_en=0:

■ Bits 5 and 6 are set to 0

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8

预留							
7	6	5	4	3	2	1	0
预留	SLV_ ACTI VITY	MST _AC TIVI TY	RFF	RFN E	TFE	TFNF	ACTI VITY

Bit	Name	R/W	Description
31:7	rsvd	RO	预留
6	SLV_ACT IVITY	RO	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. ■ 0: Slave FSM is in IDLE state so the Slave part of DW_apb_i2c is not Active ■ 1: Slave FSM is not in IDLE state so the Slave part of DW_apb_i2c is Active Reset value: 0x0
5	MST_AC TIVITY	RO	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. ■ 0: Master FSM is in IDLE state so the Master part of DW_apb_i2c is not Active ■ 1: Master FSM is not in IDLE state so the Master part of DW_apb_i2c is Active Note IC_STATUS[0]—that is, ACTIVITY bit—is the OR of SLV_ACTIVITY and MST_ACTIVITY bits. Reset value: 0x0
4	RFF	RO	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. ■ 0: Receive FIFO is not full ■ 1: Receive FIFO is full Reset value: 0x0
3	RFNE	RO	Receive FIFO Not Empty. This bit is set when the receive

			FIFO contains one or more entries; it is cleared when the receive FIFO is empty. ■ 0: Receive FIFO is empty ■ 1: Receive FIFO is not empty Reset value: 0x0
2	TFE	RO	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. ■ 0: Transmit FIFO is not empty ■ 1: Transmit FIFO is empty Reset value: 0x1
1	TFNF	RO	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. ■ 0: Transmit FIFO is full ■ 1: Transmit FIFO is not full Reset value: 0x1
0	ACTIVITY	RO	I2C Activity Status. Reset value: 0x0

12.3.31 I2C发送等级寄存器 (IC_TXFLR)

■ **Name:** I2C Transmit FIFO Level Register

■ **Size:** 32bits

■ **Address Offset:** 0x74

■ **Read/write access:** read

This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

- The I2C is disabled
- There is a transmit abort—that is, TX_ABRT bit is set in the IC_RAW_INTR_STAT register
- The slave bulk transmit mode is aborted

The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

31:4

3:0

预留	TXFLR
----	-------

Bit	Name	R/W	Description
31:4	rsvd	RO	预留
3:0	TXFLR	RO	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. Reset value: 0x0

12.3.32 I2C接收等级寄存器 (IC_RXFLR)

■ **Name:** I2C Receive FIFO Level Register

■ **Size:** 32bits

■ **Address Offset:** 0x78

■ **Read/write access:** read

This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

■ The I2C is disabled

■ Whenever there is a transmit abort caused by any of the events tracked in

IC_TX_ABRT_SOURCEThe register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

31:4	3:0
预留	RXFLR

Bit	Name	R/W	Description
31:4	rsvd	RO	预留
3:0	RXFLR	RO	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. Reset value: 0x0

12.3.33 I2CSDA保持时长寄存器 (IC_SDA_HOLD)

■ **Name:** I2C SDA Hold Time Length Register

■ **Size:** 32bits

■ **Address Offset:** 0x7C

■ **Read/write access:** Read/Write

This register controls the amount of hold time on the SDA signal after a negative edge of SCL in

both master and slave mode, in units of ic_clk period. The value programmed must be greater than the minimum holdtime in each mode for the value to be implemented—one cycle in master mode, seven cycles in slave mode. Writes to this register succeed only when IC_ENABLE[0]=0.

The programmed SDA hold time cannot exceed at any time the duration of the low part of scl.

Therefore the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the scl period measured in ic_clk cycles.

31:16	15:0
预留	IC_SDA_HOLD

Bit	Name	R/W	Description
31:16	rsvd	RO	预留
15:0	IC_SDA_HOLD	RW	Sets the required SDA hold time in units of ic_clk period. Reset value: IC_DEFAULT_SDA_HOLD

12.3.34 I2C发送中断源寄存器 (IC_TX_ABRT_SOURCE)

- **Name:** I2C Transmit Abort Source Register
- **Size:** 32bits
- **Address Offset:** 0x80
- **Read/write access:** Read

This register has 32 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; RESTART must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]).

Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

31	30	29	28	27	26	25	24
TX_FLUSH_CNT							
23	22	21	20	19	18	17	16
预留							ABR T_US ER_

							ABR T
15	14	13	12	11	10	9	8
ABR T_SL VRD _INT X	ABR T_SL V_A RBL OST	A BRT_ SLVF LUS H_T XFIF O	ARB _LOS T	ABR T_M AST ER_ DIS	ABR T_10 B_R D_N ORS TRT	ABR T_SB YTE_ NOR STRT	ABR T_HS _NO RST RT
7	6	5	4	3	2	1	0
ABR T_SB YTE_ ACK DET	ABR T_HS _AC KDE T	ABR T_GC ALL_ REA D	ABR T_G CAL L_N OAC K	ABR T_TX DAT A_N OAC K	ABR T_10 ADD R2_N OAC K	ABR T_10 ADD R1_N OAC K	ABR T_7B _AD DR_ NOA CK

Bit	Name	R/W	Description
31:2 4	TX_FLUS H_CNT	RO	This field preserves the TXFLR value prior to the last TX_ABRT event. It is cleared whenever I2C is disabled. Reset value: 0x0
23:1 7	rsvd	RO	预留
16	ABRT_U SER_ABR T	RO	This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1]). Reset value: 0x0
15	ABRT_SL VRD_INT X	RO	1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register. Reset value: 0x0
14	ABRT_SL V_ARBL OST	RO	■ 1: Slave lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time.

			<p>Note: Even though the slave never “owns” the bus, something could go wrong on the bus.</p> <p>This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_i2c no longer own the bus.</p> <p>Reset value: 0x0</p>
13	A BRT_SLV FLUSH_TXFIFO	RO	<p>■ 1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.</p> <p>Reset value: 0x0</p>
12	ARB_LOST	RO	<p>■ 1: Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration.</p> <p>Note: I2C can be both master and slave at the same time.</p> <p>Reset value: 0x0</p>
11	ABRT_MASTER_DISABLE	RO	<p>■ 1: User tries to initiate a Master operation with the Master mode disabled.</p> <p>Reset value: 0x0</p>
10	ABRT_10BIT_READ_ORSTRT	RO	<p>■ 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the master sends a read command in 10-bit addressing mode.</p> <p>Reset value: 0x0</p>
9	ABRT_SBYTE_NORSTRT	RO	<p>To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the</p>

			<p>ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted.</p> <p>1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to send a START Byte.</p> <p>Reset value: 0x0</p>
8	ABRT_HS_NORSTRT	RO	<p>■ 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode.</p> <p>Reset value: 0x0</p>
7	ABRT_SBYTE_ACKDET	RO	<p>■ 1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior).</p> <p>Reset value: 0x0</p>
6	ABRT_HS_ACKDET	RO	<p>■ 1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior).</p> <p>Reset value: 0x0</p>
5	ABRT_GENERAL_READ	RO	<p>■ 1: DW_apb_i2c in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1).</p> <p>Reset value: 0x0</p>
4	ABRT_GENERAL_NOACK	RO	<p>■ 1: DW_apb_i2c in master mode sent a General Call and no slave on the bus acknowledged the General Call.</p> <p>Reset value: 0x0</p>
3	ABRT_TXDATA_NOACK	RO	<p>■ 1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s).</p> <p>Reset value: 0x0</p>
2	ABRT_10	RO	<p>■ 1: Master is in 10-bit address mode and</p>

	ADDR2_ NOACK		the second address byte of the 10-bit address was not acknowledged by any slave. Reset value: 0x0
1	ABRT_10 ADDR1_ NOACK	RO	■ 1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Reset value: 0x0
0	ABRT_7B _ADDR_ NOACK	RO	■ 1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Reset value: 0x0

12.3.35 I2C从机响应寄存器 (IC_SLV_DATA_NACK_ONLY)

■ **Name:** Generate Slave Data NACK Register

■ **Size:** 32bits

■ **Address Offset:** 0x84

■ **Read/write access:** Read/Write

The register is used to generate a NACK for the data part of a transfer when DW_apb_i2c is acting as a slave-receiver. This register only exists when the IC_SLV_DATA_NACK_ONLY parameter is set to 1. When this parameter is disabled, this register does not exist and writing to the register's address has no effect.

A write can occur on this register if either of the following conditions are met:

■ DW_apb_i2c is disabled (IC_ENABLE[0] = 0)

■ Slave part is inactive (IC_STATUS[6] = 0)

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

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	NACK	RW	<p>Generate NACK. This NACK generation only occurs when DW_apb_i2c is a slavereceiver.</p> <p>If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted and the data received is not pushed to the receive buffer.</p> <p>When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria.</p> <ul style="list-style-type: none"> ■ 1 = generate NACK after data byte received ■ 0 = generate NACK/ACK normally <p>Reset value: 0x0</p>

12.3.36 I2CDMA控制寄存器 (IC_DMA_CR)

- **Name:** DMA Control Register
- **Size:** 32bits
- **Address Offset:** 0x88
- **Read/write access:** Read/Write

This register is only valid when DW_apb_i2c is configured with a set of DMA Controller interface signals(IC_HAS_DMA = 1). When DW_apb_i2c is not configured for DMA operation, this register does not exist and writing to the register's address has no effect and reading from this register address will return zero. The register is used to enable the DMA Controller interface operation. There is a separate bit for transmit and receive. This can be programmed regardless of the state of IC_ENABLE.

31:2	1	0
预留	TDM AE	RDM AE

Bit	Name	R/W	Description
31:2	rsvd	RO	预留

1	TDMAE	RW	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. ■ 0 = Transmit DMA disabled ■ 1 = Transmit DMA enabled Reset value: 0x0
0	RDMAE	RW	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel. ■ 0 = Receive DMA disabled ■ 1 = Receive DMA enabled Reset value: 0x0

12.3.37 I2CDMA写控制寄存器 (IC_DMA_TDLR)

- **Name:** DMA Transmit Data Level Register
- **Size:** 32bits
- **Address Offset:** 0x8C
- **Read/write access:** Read/Write

This register is only valid when the DW_apb_i2c is configured with a set of DMA interface signals (IC_HAS_DMA = 1). When DW_apb_i2c is not configured for DMA operation, this register does not exist; writing to its address has no effect; reading from its address returns zero.

31:4	3:0
预留	DMATDL

Bit	Name	R/W	Description
31:4	rsvd	RO	预留
3:0	DMATDL	RW	Transmit Data Level. This bit field controls the level at which a DMArequest is made by the transmit logic. It is equal to the watermarklevel; that is, the dma_tx_req signal is generated when the number ofvalid data entries in the transmit FIFO is equal to or below this fieldvalue, and TDMAE = 1. Reset value: 0x0

12.3.38 I2CDMA读控制寄存器（IC_DMA_RDLR）

- **Name:** DMA Receive Data Level Register
- **Size:** 32bits
- **Address Offset:** 0x90
- **Read/write access:** Read/Write

This register is only valid when DW_apb_i2c is configured with a set of DMA interface signals (IC_HAS_DMA = 1). When DW_apb_i2c is not configured for DMA operation, this register does not exist; writing to its address has no effect; reading from its address returns zero.

31:4	3:0
预留	DMAR DL

Bit	Name	R/W	Description
31:4	rsvd	RO	预留
3:0	DMARDL	RW	<p>Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE = 1. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.</p> <p>Reset value: 0x0</p>

12.3.39 I2CSDA启动寄存器（IC_SDA_SETUP）

- **Name:** I2C SDA Setup Register
- **Size:** 32bits
- **Address Offset:** 0x94
- **Read/write access:** Read/Write

This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the rising edge of SCL—relative to SDA changing—by holding SCL low when DW_apb_i2c services a read request while operating as a slave-transmitter. The relevant I2C

requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification. This register must be programmed with a value equal to or greater than 2.

Writes to this register succeed only when IC_ENABLE[0] = 0.

31:8	7:0
预留	SDA_SETUP

Bit	Name	R/W	Description
31:8	rsvd	RO	预留
7:0	SDA_SETUP	RW	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2. Default Reset value: 0x64, but can be hardcoded by setting the IC_DEFAULT_SDA_SETUP configuration parameter

12.3.40 I2CACK生成调用寄存器 (IC_ACK_GENERAL_CALL)

- **Name:** I2C ACK General Call Register
- **Size:** 32bits
- **Address Offset:** 0x98
- **Read/write access:** Read/Write

The register controls whether DW_apb_i2c responds with a ACK or NACK when it receives an I2C GeneralCall address.

31:1	0
预留	ACK_GEN_CALL

Bit	Name	R/W	Description
31:1	rsvd	RO	预留
0	ACK_GENERAL_CALL	RW	ACK General Call. When set to 1, DW_apb_i2c responds with a ACK (by asserting ic_data_oe) when it receives a

			<p>General Call. When set to 0, theDW_apb_i2c does not generate General Call interrupts.</p> <p>Default Reset value: 0x1, but can be hardcoded by setting the</p> <p>IC_DEFAULT_ACK_GENERAL_CALL configuration parameter.</p>
--	--	--	---

12.3.41 I2C使能状态寄存器 (IC_ENABLE_STATUS)

- **Name:** I2C Enable Status Register
- **Size:** 32bits
- **Address Offset:** 0x9C
- **Read/write access:** Read

The register is used to report the DW_apb_i2c hardware status when IC_ENABLE[0] is set from 1 to 0; that is, when DW_apb_i2c is disabled.

If IC_ENABLE[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.

If IC_ENABLE[0] has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

31:3	2	1	0
预留	SLV_RX_DATA_LOST	SLV_DISABLED_W HIL E_BUSY	IC_EN

Bit	Name	R/W	Description
31:3	rsvd	RO	预留
2	SLV_RX_DATA_LOST	RO	<p>Slave Received Data Lost. This bit indicates if a Slave-Receiveroperation has been aborted with at least one data byte received from anI2C transfer due to setting IC_ENABLE[0] from 1 to 0.</p> <p>When read as 1, DW_apb_i2c is deemed to have been actively engaged inan aborted I2C transfer (with matching address) and the data phase of theI2C transfer has been entered, even though a data byte has beenresponded with a NACK. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the DW_apb_i2c has a chance toNACK a transfer, and IC_ENABLE[0] has been set to 0, then this bit is also set to 1.</p>

			<p>When read as 0, DW_apb_i2c is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer.</p> <p>NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.</p> <p>Reset value: 0x0</p>
1	SLV_DISABLED_WHILE_BUSY	RO	<p>Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to setting bit 0 of the IC_ENABLE register from 1 to 0. This bit is set when the CPU writes a 0 to bit 0 of IC_ENABLE while: (a) DW_apb_i2c is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master.</p> <p>When read as 1, DW_apb_i2c is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in DW_apb_i2c (IC_SAR register) OR if the transfer is completed before bit 0 of IC_ENABLE is set to 0, but has not taken effect.</p> <p>NOTE: If the remote I2C master terminates the transfer with a STOP condition before the DW_apb_i2c has a chance to NACK a transfer, and bit 0 of IC_ENABLE has been set to 0, then this bit will also be set to 1. When read as 0, DW_apb_i2c is deemed to have been disabled when there is master activity, or when the I2C bus is idle.</p> <p>NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.</p> <p>Reset value: 0x0</p>
0	IC_EN	RO	<p>ic_en Status. This bit always reflects the value driven on the output port ic_en.</p> <ul style="list-style-type: none"> ■ When read as 1, DW_apb_i2c is deemed to be in an enabled state. ■ When read as 0, DW_apb_i2c is deemed completely inactive. <p>NOTE: The CPU can safely read this bit anytime. When</p>

			this bit is read as 0, the CPU can safely read SLV_RX_DATA_LOST (bit 2) and SLV_DISABLED_WHILE_BUSY (bit 1). Reset value: 0x0
--	--	--	--

12.3.42 I2CSS/FS尖峰抑制极限寄存器 (IC_FS_SPKLEN)

- **Name:** I2C SS and FS Spike Suppression Limit Register
- **Size:** 32bits
- **Address Offset:** 0xA0
- **Read/write access:** Read/Write

This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS or FS modes. The relevant I2C requirement is tSP (Table 4) as detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 1.

31:8	7:0
预留	IC_FS_SPKLEN

Bit	Name	R/W	Description
31:8	rsvd	RO	预留
7:0	IC_FS_SPKLEN	RW	<p>This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic; for more information, refer to “Spike Suppression” on page 54. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect. The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.</p> <p>Reset value: IC_DEFAULT_FS_SPKLEN configuration parameter.</p>

12.3.43 I2C HS尖峰抑制极限寄存器（IC_HS_SPKLEN）

- **Name:** I2C HS Spike Suppression Limit Register
- **Size:** 32bits
- **Address Offset:** 0xA4
- **Read/write access:** Read/Write

This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in HS mode. The relevant I2C requirement is tSP (Table 6) as detailed in the I2C Bus Specification. This register must be programmed with a minimum value of 1 and is implemented only if the component is configured to support HS mode; that is, if the IC_MAX_SPEED_MODE parameter is set to 3.

31:8	7:0
预留	IC_HS_SPKLEN

Bit	Name	R/W	Description
31:8	rsvd	RO	预留
7:0	IC_HS_SPKLEN	RW	<p>This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic; for more information, refer to “Spike Suppression” on page 54.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.</p> <p>This register is implemented only if the component is configured to support HS mode; that is, if the IC_MAX_SPEED_MODE parameter is set to 3.</p> <p>Reset value: IC_DEFAULT_HS_SPKLEN configuration parameter</p>

13 通用ADC(GPADC)

13.1 ADC简介

GPADC支持GPIO测量、内部温度检测和内部电压检测，同一时刻仅支持一路测量，如需多路测量建议分时使用。

13.2 ADC特性

通过gpadc_ch_sel选择ADC测量类别，支持GPIO检测、温度检测、电压检测。

最高采样精度为10比特，采集电压范围为0~1.2V。

ADC采样率 = $1.333333\text{Mhz}/(\text{GPADC_CTRL.sample_rate}+1)$

链路内部可配增益为0.5~4倍

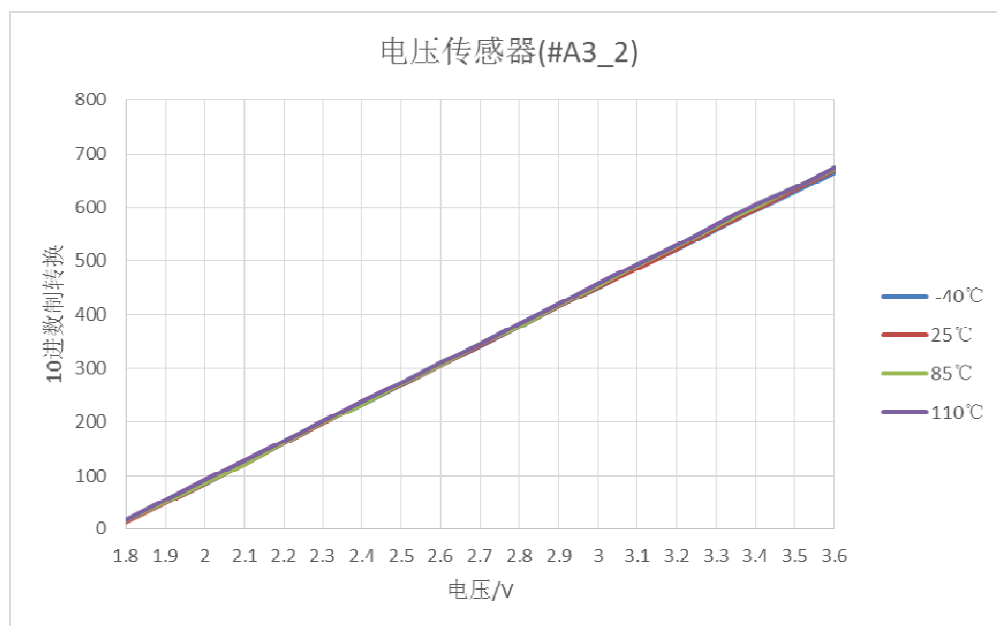
VCM（信号直流偏置）可配范围0.45V~0.80V,建议使用0.6V。

支持单端输入和差分输入，单端输入通过gpadc_sel1选择8个模拟GPIO（PA0~PA7）之一；差分输入通过gpadc_sel2和gpadc_sel1选择8个模拟GPIO（PA0~PA7）中的2个。

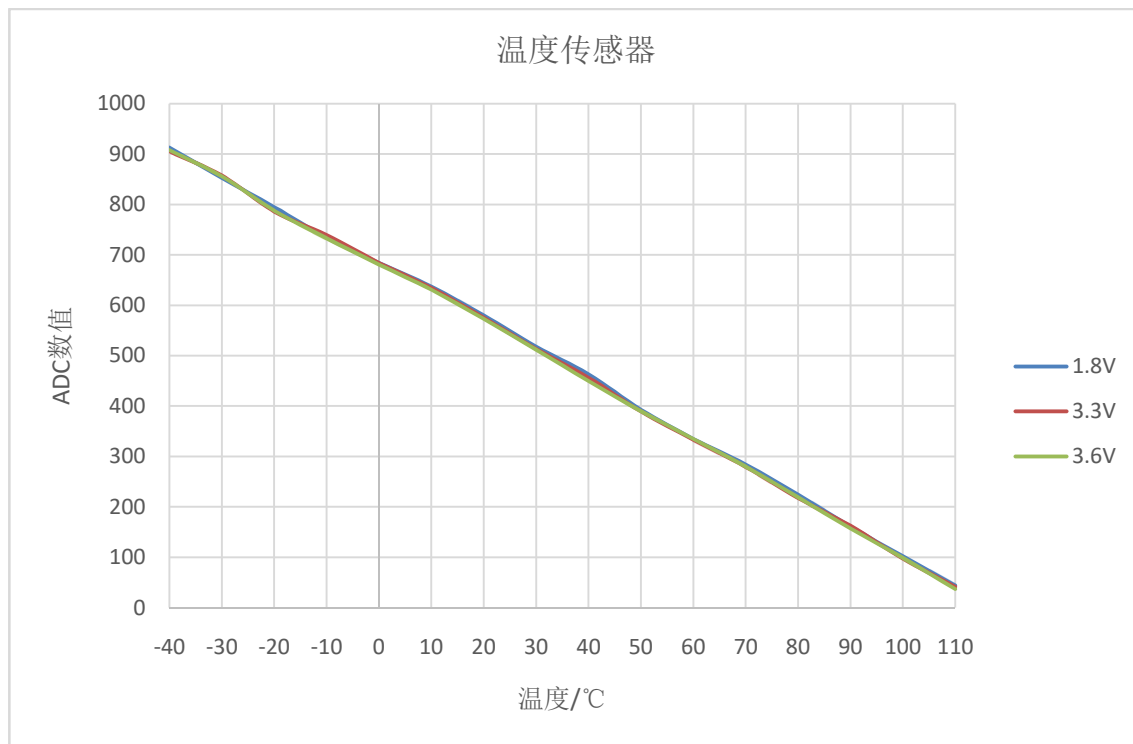
FIFO深度为16。

支持DMA操作。

13.3 电压传感器



13.4 温度传感器



13.5 ADC寄存器

13.5.1 地址映射表

ADC基地址表

地址范围	基地址	外设	总线
0x5000F000-0x5000_FFFF	0x5000_F000	GPADC	APB0

ADC寄存器偏移表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	GPADC_CONF	32	0x0000C050
0x04	GPADC_CTRL	32	0x00000000
0x08	GPADC_FIFOLEVEL	32	0x00000000
0x0C	GPADC_Threshold	32	0x00000008
0x10	GPADC_FIFOFLUSH	32	0x00000000
0x14	GPADC_FIFODATA	32	0x00000000
0x18	GPADC_STAT	32	0x00000000

0x1c	GPADC_INT_CTRL	32	0x00000000
0x20	GPADC_DATA	32	0x00000000
0x24	DMA_CR	32	0x00000000
0x28	DMA_RDLR	32	0x00000000

13.5.2 GPADC配置寄存器（GPADC_CONF）

- **Name:** Configure Register
- **Size:** 32 bits
- **Address Offset:** 0x0
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							gpadc_gain
15	14	13	12	11	10	9	8
gpadc_gain							
7	6	5	4	3	2	1	0
gpadc_sel2		gpadc_sel1		gpadc_diff_en			
gpadc_vcm_sel		gpadc_ch_sel		预留			

Bit	Name	R/W	Description
31-16	rsvd	RO	
15:4	gpadc_gain	RW	00: 检测链路的增益为0.5 01: 检测链路的增益为1 10: 检测链路的增益为2 11: 检测链路的增益为4 电压检测时增益必须设置为2 温度检测时增益必须设置为4
13:1	gpadc_sel2	RW	对8个模拟GPIO进行8选1作为负输入端

10:8	gpadc_sel 1	RW	对8个模拟GPIO进行8选1作为正输入端
7	rsvd	RO	
6:4	gpadc_vcm_sel	RW	000: VCM=0.45 001: VCM=0.50 010: VCM=0.55 011: VCM=0.60 100: VCM=0.65 101: VCM=0.70 110: VCM=0.75 111: VCM=0.80
3	gpadc_diff_en	RW	0: 检测GPIO单端输入信号 1: 检测GPIO差分输入信号 当检测温度、电池和offset时，gpadc_diff_en必须设置0
2:1	gpadc_ch_sel	RW	00: 保留。 01: 温度检测 10: 电池检测 11: GPIO检测
0	rsvd	RO	

13.5.3 GPADC控制寄存器（GPADC_CTRL）

- **Name:** Control Register
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							sample_rate
15	14	13	12	1	10	9	8
				1			

sample_rate							
7	6	5	4	3	2	1	0
预留				gpadc_mode		gpadc_en	
Bit	Name		R/W		Description		
31-16	rsvd		RO				
15-8	sample_rate		RW		GPADC FIFO Mode sample rate = 1.333333Mhz/(sample_rate+1)		
7-3	rsvd		RO				
2-1	gpadc_mode		RW		00: 单次转换，数据写入 GPADC_DATA 01: 多次转换，数据写入 GPADC_DATA 10: 连续转换，数据写入FIFO		
0	gpadc_en		RW		GPADC enable 当gpadc_mode=00时,单次转换，数据写入GPADC_DATA，完成后gpadc_en硬件自动清零，并且给出gpadc_done_int中断； 当gpadc_mode=01时，多次转换，当前实时数据写入 GPADC_DATA，不会产生 gpadc_done_int中断，需软件清零gpadc_en。 当gpadc_mode=10时,多次转换，数据写入FIFO，软件清零 gpadc_en。		

13.5.4 GPADC FIFOLEVEL寄存器 (GPADC_FIFOLEVEL)

■ Name: FIFOLEVEL Register

- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** read

3	3	2	2	2		26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7			5	4	3	2	1	0	9	8	7	6
预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												
预留												fifo level				

Bit	Name	R/W	Description
31-6	rsvd	RO	
4-0	fifo level	RO	FIFO Level, 最大值为FIFO深度16

13.5.5 GPADC Threshold寄存器（GPADC_Threshold）

- **Name:** Threshold Register
- **Size:** 32 bits
- **Address Offset:** 0x0C
- **Read/write access:** read/write

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1											
预留											fifo threshold				
Bit		Name				R/W				Description					
31-6		rsvd				RO									
4-0		fifo threshold				RW				FIFO Threshold					

13.5.6 GPADC FIFO FLUSH寄存器（GPADC FIFO FLUSH）

- **Name:** FIFOFLUSH Register

- **Size:** 32 bits
- **Address Offset:** 0x10
- **Read/write access:** write

3	3	2	2	2		26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7			5	4	3	2	1	0	9	8	7	6
预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												
预留															fifo flush	

Bit	Name	R/W	Description
31-6	rsvd	RO	
4-0	fifo flush	RW	FIFO Flush操作

13.5.7 GPADC FIFO DATA寄存器（GPADC FIFO DATA）

- **Name:** FIFODATA Register
- **Size:** 32 bits
- **Address Offset:** 0x14
- **Read/write access:** read

3	3	2	2	2		26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7			5	4	3	2	1	0	9	8	7	6
预留																
1	1	1	1	1		10	9	8	7	6		4	3	2	1	0
5	4	3	2	1												
预留							fifo_data									

Bit	Name	R/W	Description
31-6	rsvd	RO	
4-0	fifo_data	RO	FIFO DATA

13.5.8 中断状态寄存器 (GPADC_STAT)

■ **Name: Interrupt Status Register**

■ **Size: 32 bits**

■ **Address Offset: 0x18**

■ **Read/write access: write**



Bit	Name	R/W	Description
31-3	rsvd	RO	
2	int_ovfl	WC	FIFO写满溢出时产生中断 0: GPADC interrupt not occur 1: GPADC interrupt occur Write 1 to clear it. Write 0 is ignored
1	int_alert	WC	当FIFO Level>=FIFO Threshold时，产生中断 0: GPADC interrupt not occur 1: GPADC interrupt occur Write 1 to clear it. Write 0 is ignored.
0	int_done	WC	GPADC done interrupt occurs. 0: GPADC interrupt not occur 1: GPADC interrupt occur Write 1 to clear it. Write 0 is ignored.

13.5.9 中断控制寄存器 (GPADC_INT_CTRL)

■ **Name:** Interrupt Control Register

■ **Size:** 32 bits

■ **Address Offset:** 0x1C

■ **Read/write access:** read/write

31	30	29	28	27	26	25	24
预留							
23	22	21	20	19	18	17	16
预留							
15	1	13	1	11	10	9	8
	4		2				
预留				int_ovfl_ mask	int_alert_ mask	int_done_ mask	
7	6	5	4	3	2	1	0
预留				int_ovf l_en	int_aler t_en	int_don e_en	

Bit	Name	R/W	Description
31-11	rsvd	RO	
10	int_ovfl_mask	RW	Allows the user to mask a generated FIFO Overflow interrupt. 0 = Interrupt unmasked 1 = Interrupt masked
9	int_alert_mask	RW	Allows the user to mask a generated FIFO Thresholds Alert interrupt. 0 = Interrupt unmasked 1 = Interrupt masked
8	int_done_mask	RW	Allows the user to mask a generated GPADC done interrupt. 0 = Interrupt unmasked 1 = Interrupt masked

7-3	rsvd	RO	
2	int_ovfl_en	RW	Allows the user to disable FIFO Overflow interrupt generation. 0 = Interrupt disabled 1 = Interrupt enabled
1	int_alert_en	RW	Allows the user to disable FIFO Thresholds Alert interrupt generation. 0 = Interrupt disabled 1 = Interrupt enabled
0	int_done_en	RW	Allows the user to disable GPADC done interrupt generation. 0 = Interrupt disabled 1 = Interrupt enabled

13.5.10 GPADC数据寄存器 (GPADC_DATA)

- **Name:** Data Register
- **Size:** 32 bits
- **Address Offset:** 0x20
- **Read/write access:** read

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1		9	8	7	6		4	3	2	1	0
5	4	3	2	1	10										
预留								gpadc_data							

Bit	Name	R/W	Description
31-6	rsvd	RO	
4-0	gpadc_data	RO	ADC数据

13.5.11 DMA控制寄存器（DMA_CR）

■ **Name:** DMA Control Register

■ **Size:** 32 bits

■ **Address Offset:** 0x24

■ **Read/write access:** read/write



Bit	Name	R/W	Description
31-2	rsvd	RO	
1	rsvd	RW	
0	dmacr	RW	DMA使能

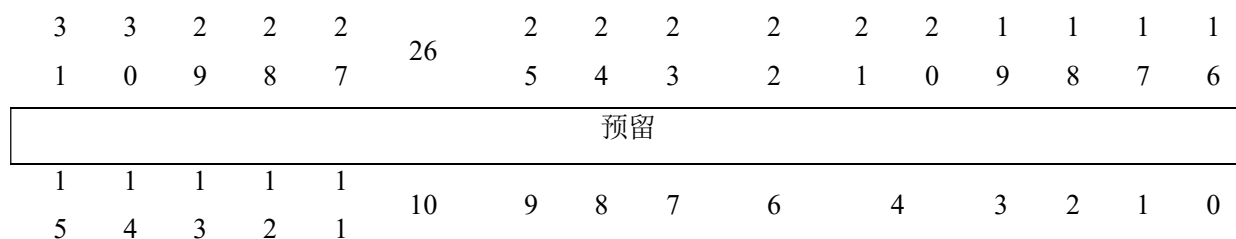
13.5.12 DMA接收FIFOLEVEL寄存器（DMA_RDLR）

■ **Name:** DMA Receive FIFOLEVEL Register

■ **Size:** 32 bits

■ **Address Offset:** 0x28

■ **Read/write access:** read/write



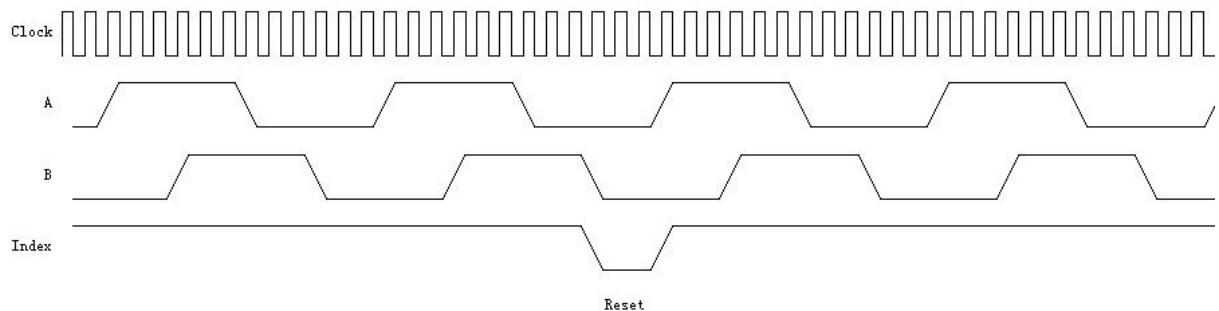
预留	dmardlr
----	---------

Bit	Name	R/W	Description
31-4	rsvd	RO	
3-0	dmardlr	RW	DMA receive fifo data level 当FIFO Level >= dmardlr + 1 时，发起DMA请求

14 正交解码器(QuadDec)

14.1 QuadDec简介

正交解码器 (QuadDec) 组件提供针对成对数字信号的跃变进行计数的功能。这些信号通常由电机或轨迹球上安装的速度/位置反馈系统来提供。



这两个信号通常称为A和B，相位上偏离90度角，使得输出呈现格雷码规律。格雷码是在每次计数时仅有一位发生变化的序列。这是避免短时脉冲发生的有效方法。此外，通过格雷码，还可以检测方向和相对位置。第三个可选信号命名为**Index**（索引），为每转一圈产生一次绝对位置的参考。

正交解码器用于解码正交编码器的输出。正交编码器感应对象（例如，鼠标、轨迹球、自动控制轴等）当前位置、速度和方向。此外，正交解码器还用于精确测量电机转子的速度，加速度和位置，并结合旋钮确定用户的输入。

14.2 QuadDec特性

14.2.1 索引输入

此输入用于检测正交解码器的参考位置。缺省状态下，使用索引输入时，如果输入A、B，并且索引全部为0，则计数器也要复位到0。索引脉冲选通，索引脉冲选通时，索引、A、B的有效电平都可以通过寄存器来配置。通过索引选通，计数器可以仅在多次旋转的某一次进行复位。

14.2.2 数据过滤

时钟信号用于采样和过滤输入上的短时脉冲。过滤输出不会发生变化，直到3个连续输入采样具有相同值。采样时钟周期应该大于短时脉冲预期发生的最大时间周期。理论上可以过滤掉小于两倍采样时钟周期的短时脉冲。

时钟输入频率应大于A或B输入频率最大值的6倍。

索引脉冲应该保持至少3个时钟周期。

索引有效脉冲时间（结合A、B有效值，共同使计数器复位的状态）应该大于一个时钟周期。

14.2.3 中断输出

在下列一种或多种情况下可产生中断：

- 计数器上溢出
- 计数器下溢出
- 索引输入时
- A和B输入上产生无效状态跃变
- 计数器达到预定值时

14.2.4 状态跃变

正交解码器使用状态机和递增/递减计数器进行解码，默认配置为32位递增/递减计数器，分辨率为4x。解码器有4种状态，对应于A和B输入所有可能的值。状态跃变图如下所示。标有“+”和“-”的状态跃变表示正交相位计数器递增和递减操作。对于正交相位信号的每一个满周期，正交相位计数器发生4次计数变化。

14.3 使用说明

配置3个io复用作为QDEC_CHA、QDEC_CHB、QDEC_IDX

QDEC_Init(ENABLE);

cnt= QDEC_GetInternalCounter();//读取数值

具体参考Qdec示例。

14.4 QuadDec寄存器

14.4.1 地址映射表

QuadDec基地址表

地址范围	基地址	外设	总线
0x5000_6000 -0x5000_6FFF	0x5000_6000	Qdec	APB0

QuadDec寄存器偏移地址表

偏移地址	寄存器名称	宽度（bit）	复位值
0x00	QCR	32	0x00000000

0x04	QSR	32	0x00000000
0x08	QIR	32	0x00000000
0x0C	QRW	32	0x00000000
0x10	QCK	32	0x00000000

14.4.2 Quadrature Control寄存器（QCR）

■ **Name: Quadrature Control Register**

■ **Size: 32 bits**

■ **Address Offset: 0x0**

■ **Read/write access: read/write**

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6
预留															
15	14		13		12	1	10		9		8				
1															
预留												INR	IDXL		
												C			
7	6	5	4	3	2		1		0						
ICH	ICH	QL	PLC	INZ	INEN		CTDR		ECNT						
B	A	AT	T	C											

Bit	Name	W/ R	Description
31:1 0	rsvd	R O	预留
9	INRC	R W	Index Read Count Bit 0 = Quadrature Read / Write register not affected. 1 = Read the value of the quad. count to the QRW reg, when index event is true.
8	IDXL	R W	Index Level configuration 0 = Index asserted when quadrature index is logic low 1 = Index asserted when quadrature index is logic high

7	ICHB	R W	Index Channel B configuration 0 = Index asserted when quadrature channel B logic low 1 = Index asserted when quadrature channel B logic high
6	ICHA	R W	Index Channel A configuration 0 = Index asserted when quadrature channel A logic low 1 = Index asserted when quadrature channel A logic high
5	QLAT	R W	Quadrature Count Latch 0 = No action. 1 = Latch and store quad count in QRW register, auto cleared
4	PLCT	R W	Pre Load Count register 0 = No action. 1 = Load value currently in pre load reg into count reg, auto clear
3	INZC	R W	Index Zero Count Bit 0 = Internal count not affected. 1 = Zero internal quad_count when quad_index is asserted.
2	INEN	R W	Index Enable Bit 0 = Index input disabled 1 = Index input enabled
1	CTDR	R W	Set Count Direction 0 = Counts positive when A leads B 1 = Counts negative when A leads B
0	ECNT	R W	Enable Counting 0 = Quadrature counting disabled 1 = Quadrature counting enabled

14.4.3 Quadrature Status寄存器（QSR）

- **Name: Quadrature Status Register**
- **Size: 32 bits**
- **Address Offset: 0x04**

■ Read/write access: read/write

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6
预留															
15		14		13		12		1		10		9		8	
								1							
预留															
7	6	5	4	3		2		1		0					
预留				CMI	INI	UNI	OVI	QEI							

Bit	Name	W/ R	Description
31:5	rsvd	R O	预留
4	CMI	R W	Count Compare Match Event, auto set, user cleared 0 = Compare match event has not occurred 1 = Compare match event occurred, interrupt genetated if enabled
3	INI	R W	Index event, auto set, user cleared 0 = Index event has not occurred 1 = Index event occured, interrupt requested if INIE set
2	UNI	R W	Counter Underflow, auto set, user cleared 0 = No underflow detected 1 = Counter underflow from 0x0000 to 0xFFFF
1	OVI	R W	Counter Overflow, auto set, user cleared 0 = No overflow detected 1 = Counter overflow from 0xFFFF to 0x0000
0	QEI	R W	Quadrature decoder error status, auto set, user cleared 0 = No error 1 = Illegal state transition detected

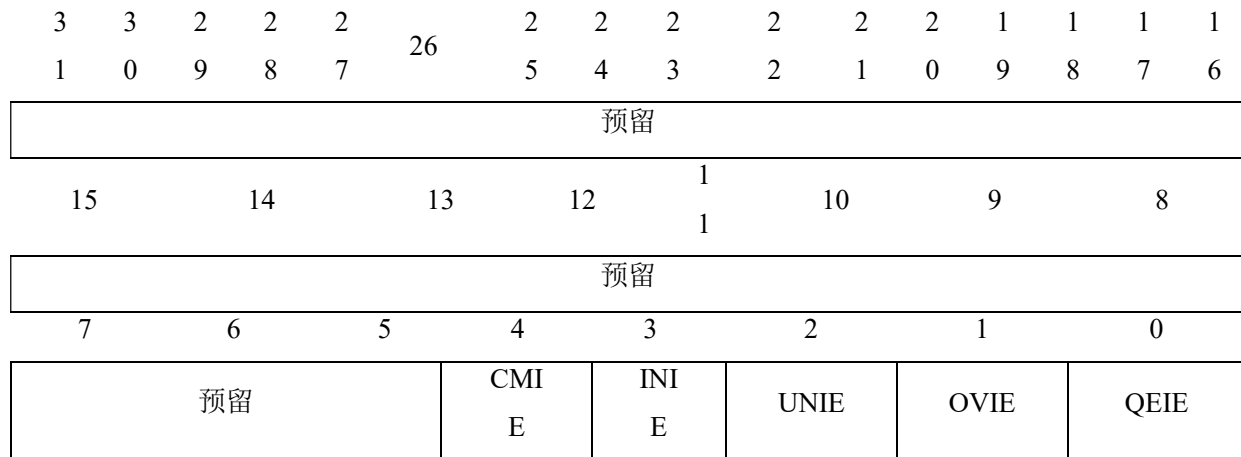
14.4.4 Quadrature Interrupt enable寄存器（QIR）

■ **Name:** Quadrature Interrupt enable Register

■ **Size:** 32 bits

■ **Address Offset:** 0x08

■ **Read/write access:** read/write



Bit	Name	W/ R	Description
31:5	rsvd	R O	预留
4	CMIE	R W	Compare Match Interrupt Enable 0 = No interrupt generated when a compare match event is asserted. 1 = An external interrupt will be generated when the compare event is asserted.
3	INIE	R W	Index Interrupt Enable 0 = Index interrupt request disabled 1 = Index interrupt request enabled
2	UNIE	R W	Underflow Interrupt Enable 0 = Underflow event will not trigger an interrupt. 1 = Underflow event will trigger an interrupt
1	OVIE	R W	Overflow Interrupt Enable 0 = Overflow event will not trigger an interrupt 1 = Overflow event will trigger an interrupt

0	QEIE	R W	Quadrature Error Interrupt enable 0 = Quadrature Error Interrupt disabled 1 = Quadrature Error Interrupt enabled
---	------	--------	--

14.4.5 Quadrature Count Read / Write寄存器 (QRW)

■ **Name: Quadrature Count Read / Write Register**

■ **Size:** 32 bits

■ **Address Offset:** 0x0C

■ **Read/write access:** read/write

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6
QRW															
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1
5	4	3	2	1											0
QRW															

Bit	Name	W/ R	Description
31:0	QRW	R W	<p>The actual quadrature count value must be latched prior to reading from this register. This may be triggered two ways:</p> <p>1) Writing a '1' to the QCR, bit 5 quadrature count latch (QLAT), -or-</p> <p>2) Asserting the external asynchronous input quad_idx_i.</p> <p>Once either event occurs, the quadrature count value will be copied to the QRW register.</p> <p>This register is also used to hold the pre-load count value.</p> <p>After writing to this register, the pre-load value is transferred to the quadrature count register by</p>

			writing a '1' to the QCR, bit 4, quadrature pre-load count (PLCT)
--	--	--	---

14.4.6 Quadrature Sample Clock Frequency Selection寄存器（QCK）

■ **Name:** Quadrature Sample Clock Frequency Selection Register

■ **Size:** 32 bits

■ **Address Offset:** 0x10

■ **Read/write access:** read/write

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6
预留															
1	1	1	1	1	10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1											
预留											QCK				

Bit	Name	W/ R	Description
31:5	rsvd	R O	预留
4:0	QCK	R W	根据以下公式，来定义 Quadrature Decoder 采样时钟的频率： $F(qdec_clk) = F(PCLK)/(QCK[4:0]+1)$

15 梯形键盘扫描(KeyBoard)

15.1 KeyBoard简介

本芯片内置梯形键盘结构，可以为产品的按键减少IO占用，使用16个IO最多可以支持136个按键。

15.2 KeyBoard特性

键盘扫描采用硬件方式实现，扫描时钟为32Khz时钟，扫描周期可通过KSPW和KSPON配置。通过GPIO复用寄存器Px_n_mode选择复用为keyboard的IO。再通过KPC_SEL独立的控制位控制其是否用于键盘。

键盘KSPW与KSPON控制的扫描时间如下表：

表15.1键盘扫描时间表

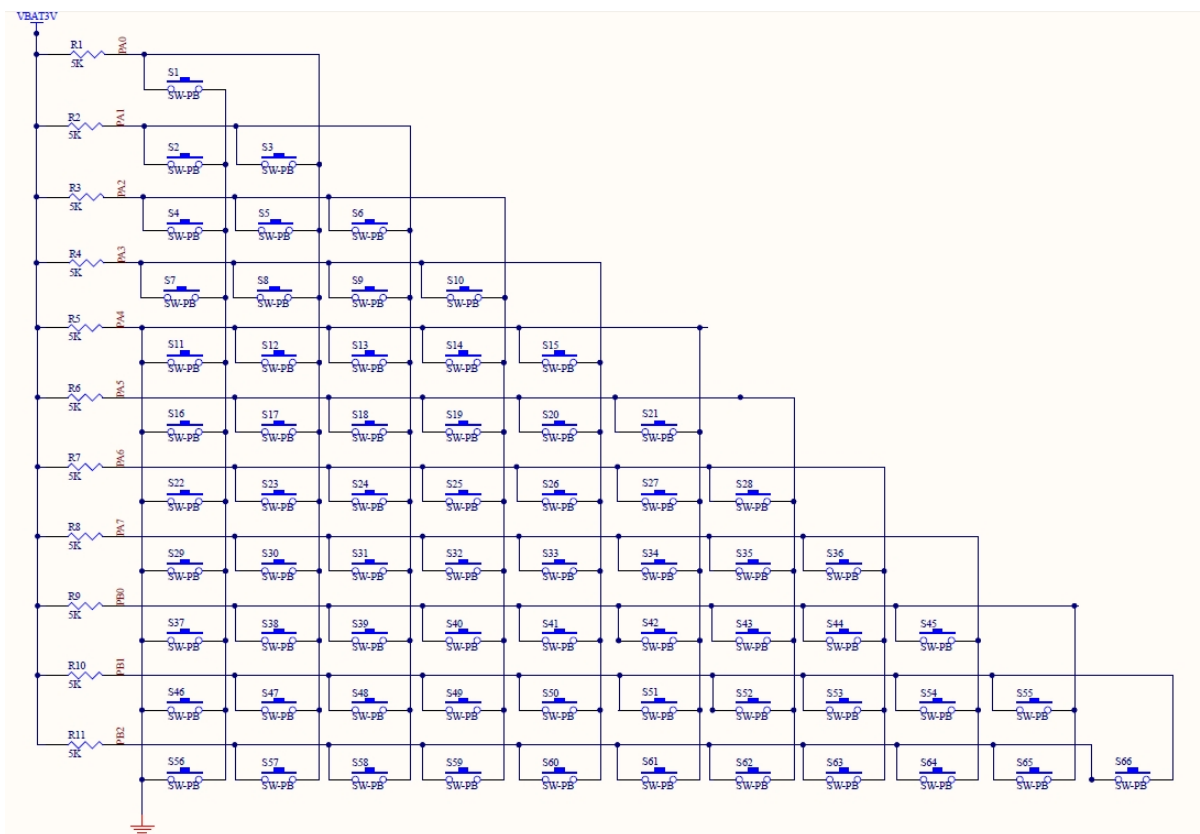
KSPW	KSPON	扫描时间	备注
2'b00	2'b00	4ms	16x4x2x31.25us
	2'b01	6ms	16x4x3x31.25us
	2'b1x	8ms	16x4x4x31.25us
2'b01	2'b00	8ms	16x8x2x31.25us
	2'b01	12ms	16x8x3x31.25us
	2'b1x	16ms	16x8x4x31.25us
2'b10	2'b00	16ms	16x16x2x31.25us
	2'b01	24ms	16x16x3x31.25us
	2'b1x	32ms	16x16x4x31.25us
2'b11	2'b00	32ms	16x32x2x31.25us
	2'b01	48ms	16x32x3x31.25us
	2'b1x	64ms	16x32x4x31.25us

当硬件检测到按键扫描时（只要有一个按键被按下），上报中断给CPU，如果第一列按键通过判断数据寄存器（Px_IOD）的输入数据寄存器确认键值，如果非第一列按键则判断KPC_DATA确认键值，去判断哪一个或者多个键被按下。

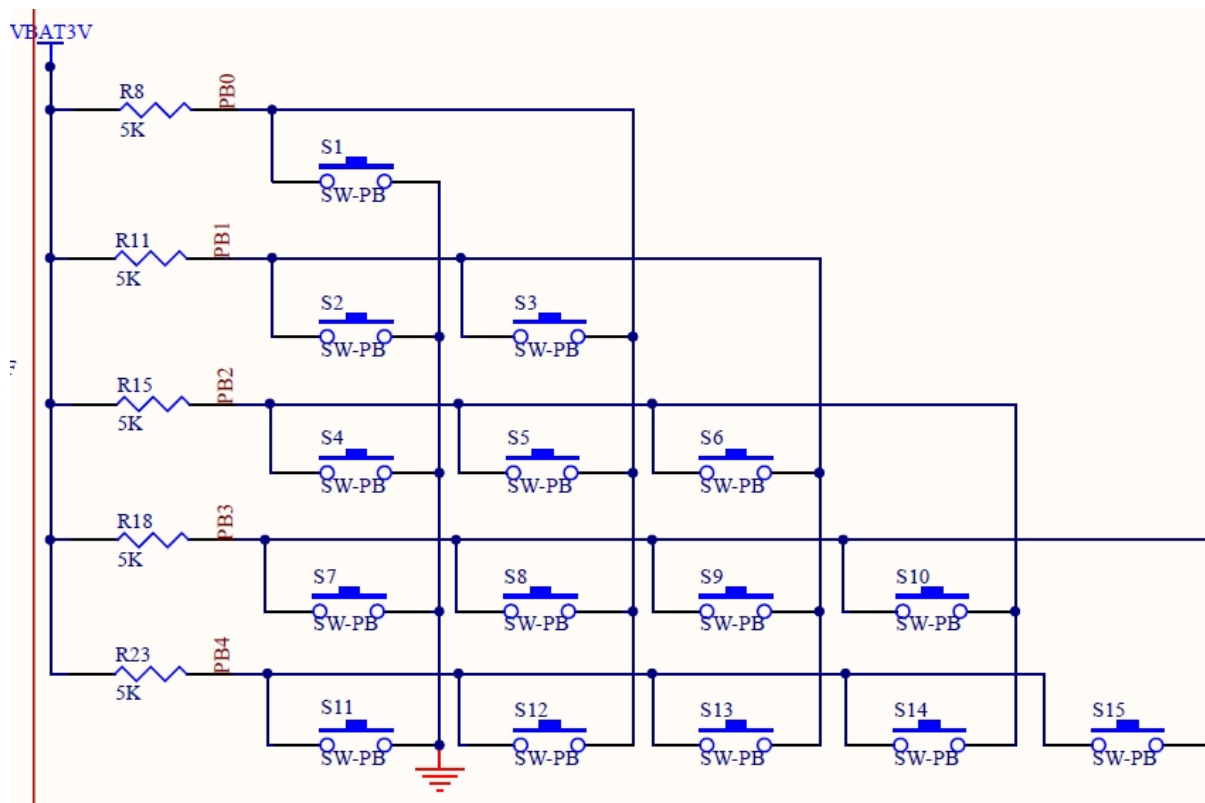
当需要支持多个按键被同时按下时，对于楼梯形的键盘结构，建议仅支持第一列或同一行中除第一列的其他多个按键的组合。

内置的键盘扫描电路（例如当使用11个IO时）支持的键盘结构如下图所示。

11个io矩阵示例:



5个io矩阵示例:



使用说明:

以5 个io矩阵为例:

1. 如果按下 S1 以后, PB0 由高变低, 产生 keyboard 中断, 寄存器 GPIOB 的 Px_IOD 寄存器由 0x1f, 变化为 0x1e, 确认 K1 按下, 键值是 0x1e。
2. 如果 S2 按下同 S1, PB1 由高变低, 键值为 0x1c。
3. 如果按下 S3, 则 PB0 和 PB1 连通, GPIOB 的 Px_IOD 寄存器保持 0x1f 不变, keyboard 的 KPC_DATA 寄存器由 0x1f 变化成 0x1c, 则判断 S3 被按下, 键值为 0x1c。

注意:

1. 第一列按键被按下,通过检测 gpio 数据寄存器确认键值,非第一列案件被按下,检测 keyboard 的 KPC_DATA 寄存器确认键值。
2. 建议选择连续的 gpio 作为 keyboard 的 GPIO。

15.3 KeyBoard寄存器

15.3.1 地址映射表

KPC基地址列表

地址范围	基地址	外设	总线
0x50007000-0x50007014	0x5000_7000	KPC	APB0

KPC寄存器偏移地址列表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	KPC_CTRL	32	0x00000000
0x04	KPC_CFG1	32	0x00000000
0x08	KPC_CFG2	32	0x00000000
0x0C	KPC_STAT	32	0x00000000
0x10	KPC_INT_CTRL	32	0x00000000
0x14	KPC_DATA	32	0x0000ffff

15.3.2 KPC使能寄存器 (KPC_CTRL)

■ **Name: Control Register**

■ **Size: 32 bits**

■ **Address Offset: 0x0**

■ **Read/write access: read/write**

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												
预留															kpc_ en	

Bit	Name	W/R	Description
31:1	预留	-	
0	kpc_en	W/R	Keypad scan enable: 0: disable; 1: enable;

15.3.3 KPC配置寄存器1 (KPC_CFG1)

- **Name:** Configure Register 1
- **Size:** 32 bits
- **Address Offset:** 0x04
- **Read/write access:** read/write

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												
KPC_CFG1																

Bit	Name	W/R	Description
31:1	预留	-	
6			
15:0	KPC_CFG1	W/R	对应的 KPC_PIO[15:0]使能位

15.3.4 KPC配置寄存器2 (KPC_CFG2)

- **Name:** Configure Register 2
- **Size:** 32 bits
- **Address Offset:** 0x08
- **Read/write access:** read/write

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6

预留															
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												

预留	kspw
----	------

Bit	Name	W/R	Description																																
31:4	预留	-																																	
3:0	kspw	W/R	键盘扫描时间表: 设定值参考下表: <table border="1" style="width: 100%; margin-top: 10px;"> <thead> <tr> <th>KSPW [3:2]</th><th>KSPW [1:0]</th><th>扫 描 时 间</th><th>备注</th></tr> </thead> <tbody> <tr> <td>2'b00</td><td>2'b00</td><td>4 m s</td><td>16x4x2x31.2 5us</td></tr> <tr> <td></td><td>2'b01</td><td>6 m s</td><td>16x4x3x31.2 5us</td></tr> <tr> <td></td><td>2'b1x</td><td>8 m s</td><td>16x4x4x31.2 5us</td></tr> <tr> <td></td><td></td><td></td><td></td></tr> <tr> <td>2'b01</td><td>2'b00</td><td>8 m s</td><td>16x16x2x31. 25us</td></tr> <tr> <td></td><td>2'b01</td><td>1 2 m s</td><td>16x16x3x31. 25us</td></tr> <tr> <td></td><td>2'b1x</td><td>1 6 m</td><td>16x16x4x31. 25us</td></tr> </tbody> </table>	KSPW [3:2]	KSPW [1:0]	扫 描 时 间	备注	2'b00	2'b00	4 m s	16x4x2x31.2 5us		2'b01	6 m s	16x4x3x31.2 5us		2'b1x	8 m s	16x4x4x31.2 5us					2'b01	2'b00	8 m s	16x16x2x31. 25us		2'b01	1 2 m s	16x16x3x31. 25us		2'b1x	1 6 m	16x16x4x31. 25us
KSPW [3:2]	KSPW [1:0]	扫 描 时 间	备注																																
2'b00	2'b00	4 m s	16x4x2x31.2 5us																																
	2'b01	6 m s	16x4x3x31.2 5us																																
	2'b1x	8 m s	16x4x4x31.2 5us																																
2'b01	2'b00	8 m s	16x16x2x31. 25us																																
	2'b01	1 2 m s	16x16x3x31. 25us																																
	2'b1x	1 6 m	16x16x4x31. 25us																																

		s	
2'b10	2'b00	1 6 m s	16x32x2x31. 25us
	2'b01	2 4 m s	16x32x3x31. 25us
	2'b1x	3 2 m s	16x32x4x31. 25us
2'b11	2'b00	3 2 m s	16x64x2x31. 25us
	2'b01	4 8 m s	16x64x3x31. 25us
	2'b1x	6 4 m s	16x64x4x31. 25us

15.3.5 中断状态寄存器 (KPC_STAT)

- **Name: Interrupt Status Register**
- **Size: 32 bits**
- **Address Offset: 0x0C**
- **Read/write access: write**

3	3	2	2	2	26	2	2	2	2	2	2	1	1	1	1
1	0	9	8	7		5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												
预留															kpc_ stat	

Bit	Name	W/R	Description
31:2	预留	-	
0	kpc_stat	W	Keypad interrupt (key press or release) occurs. 0: Keypad interrupt not occur 1: Keypad interrupt occur Write 1 to clear it. Write 0 is ignored.

15.3.6 中断控制寄存器 (KPC_INT_CTRL)

■ **Name: Interrupt Control Register**

■ **Size: 32 bits**

■ **Address Offset: 0x10**

■ **Read/write access: read/write**

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留														int_ mas k	int_ _e n	

Bit	Name	W/R	Description
31:2	预留	-	
1	int_mask	W/R	Allows the user to mask a generated interrupt. 0 = Interrupt unmasked 1 = Interrupt masked
0	int_en	W/R	Allows the user to disable interrupt

			generation. 0 = Interrupt disabled 1 = Interrupt enabled
--	--	--	--

15.3.7 KPC数据寄存器（KPC_DATA）

■ **Name: Data Register**

■ **Size: 32 bits**

■ **Address Offset: 0x14**

■ **Read/write access: read**

3	3	2	2	2		2	2	2	2	2	2	1	1	1	1
1	0	9	8	7	26	5	4	3	2	1	0	9	8	7	6

预留																
1	1	1	1	1		10	9	8	7	6	5	4	3	2	1	0
5	4	3	2	1												

kpc_data															
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Bit	Name	W/R	Description
31:1 6	预留	-	
15:0	kpc_data	R	扫描返回的数据值

16 直接存储器存取(DMA)

16.1 DMA简介

直接存储器存取(DMA)用来提供在外设和存储器之间或存储器与存储器之间的高速传输。无需CPU干预,数据可以通过DMA快速地移动,节省CPU资源来进行其他操作。

DMA控制器有4个通道,专门用来管理内存到内存、内存到外设、外设到内存的访问请求。

16.2 DMA特性

支持多种宽度数据传输

每个通道有各自独立中断信号及标记

支持内存到内存、内存到外设、外设到内存之间的传输

16.3 DMA 外设类型

使用DMA传输的外设可分为两类外设:

存储器外设

非存储器外设

存储器外设:

存储器外设与DMA控制器间不需要“DMA握手接口”,因此当DMA通道使能后,DMA不需要请求信号即可开始DMA传输。同时由于无握手接口,所以SRC_MSIZE和DEST_MSIZE对

存储器外设无限制作用。

非存储器外设:

非存储器外设,如:UART、I2C、SPI等均包含与DMA直接进行通信的请求信号,即DMA握手接口。非存储器外设进行DMA传输时,会通过DMA直接的握手接口对数据的传输进行控制。DMA接口分为两种:硬握手接口和软握手接口。

16.4 DMA握手接口

DMA提供两种DMA请求握手接口：

硬握手接口

软握手接口

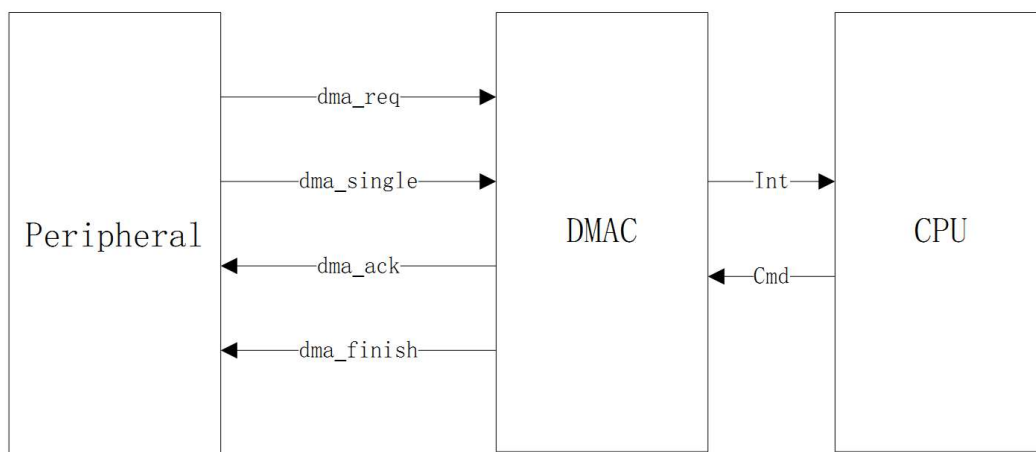
用户可以通过DMA寄存器配置各通道寄存器选择握手接口方式。

DMA硬握手接口

当用户配置对应外设DMA使能、配置相应触发条件及对应DMA通道为硬握手信号后，DMA请求信号触发将由硬件根据条件自动完成。

DMA控制器硬握手接口通过dma_req、dma_single、dma_ack和dma_finish五个信号同相应外设

进行交互。SCPU中具有DMA功能的外设均包含以下接口信号。



Name	Description
dma_req	Burst transaction request from peripheral 外设向 DMA 发出突发传输请求
dma_single	Single transfer status 外设单次传输状态
dma_ack	DMA 应答信号 DMA 完成 1 次在 AHB 总线上传输（突发（Burst）或单次（single））后对外设响应信号
dma_finish	DMA 传输完成 DMA 完成块（BLOCK）传输后对外设响应信号

16.5 DMA中断

16.5.1 中断类型

DMA每个通道均包含5种中断类型：

块传输完成中断（IntBlock – Block Transfer Complete Interrupt）

DMA 通道对目标设备的块传输数据完成后，中断置位。

目标数据传输完成中断（IntDstTran – Destination Transaction Complete Interrupt）

DMA 通道在AHB 总线上完成1次（“单次”或“突发”）对目标设备的数据传输后，中断置位。

错误中断（IntErr – Error Interrupt）

在 DMA 传输过程中，DMA 接收到来自AHB 总线上的错误应答后，中断置位。

源数据传输完成中断（IntSrcTran – Source Transaction Complete Interrupt）

DMA 通道在AHB 总线上完成1次（“单次”或“突发”）对源设备的数据传输后，中断置位。

DMA 传输完成中断（IntTfr – DMA Transfer Complete Interrupt）

DMA 完成对目标设备数据传输后，中断置位。

16.5.2 中断寄存器

5组中断寄存器对应每种中断类型，其中每个寄存器中的bit位对应1个通道。

原（Raw）中断状态寄存器：RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr

DMA 中断事件被保存在原中断状态寄存器中

中断状态寄存器：StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr

未被中断屏蔽寄存器屏蔽的中断状态保存到中断状态寄存器中

中断屏蔽寄存器：MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr

打开或关闭DMA 通道中断状态

中断清除寄存器：ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr

对中断控制寄存器写“1”清除对应通道产生中断。

16.6 DMA数据传输规则

DMA寄存器与数据类型及传输量有关的3个参数分别是：

传输数据块大小：BLOCK_TS

突发数据传输大小：SRC_MSIZE、DEST_MSIZE

数据位宽：SRC_TR_WIDTH、DST_TR_WIDTH

存储器到存储器

存储器到存储器的DMA传输由DMA BLOCK_TS和SRC_TR_WIDTH来决定传输量，DMA与存

储器外设无握手接口，数据传输过程中不会对数据传输进行控制。

总传输量Bytes = BLOCK_TS * (SRC_TR_WIDTH / 8)

存储器到外设/外设到存储器

在存储器到外设/外设到存储器情况下，存储器与DMA直接无握手接口库，存储器到DMA与存

储器到存储器间情况一样。

DMA与非存储器外设的数据传输会受到FIFO大小的影响，因此DMA会对数据传输进行控制。

DMA使用SRC_MSIZE/DEST_MSIZE来限制1次“突发”（Burst）传输的数据量大小。

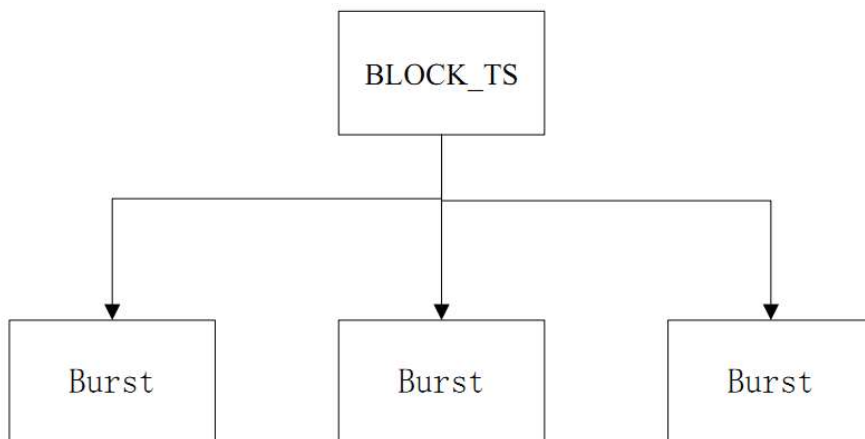
若DMA BLOCK_TS 是 SRC_MSIZE/DEST_MSIZE 的整数倍，则整个传输过程中仅产生“突发”传输。

示例参数配置：

DMA BLOCK_TS = 12

SRC_MSIZE/DEST_MSIZE = 4

根据以上配置则DMA传输控制流程如下：



若 DMA BLOCK_TS 不是 SRC_MSIZE/DEST_MSIZE 的整数倍，则在最后剩余的传输中使用“单次”（Single）传输完成。“单次”传输每次仅传输 SRC_TR_WIDTH/DST_TR_WIDTH 单个数据

示例参数配置：

DMA BLOCK_TS = 12

SRC_MSIZE/DEST_MSIZE = 5

根据以上配置则DMA传输控制流程如下：

SRC_MSIZE/DEST_MSIZE参数置设定

在非存储器外设中会使用到 SRC_MSIZE/DEST_MSIZE 参数。

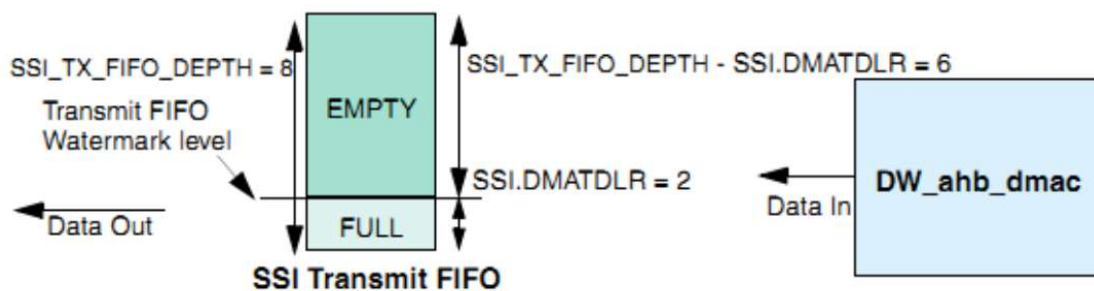
SRC_MSIZE/DEST_MSIZE 与非存储器外设接收和发送FIFO 设定相关。

当非存储器外设作为目标设备时，非存储器外设发送 FIFO 中数据达到设定阈值后，向DMA发

出请求，此时 DMA 会根据 DEST_MSIZE 中的数据量向发送 FIFO 中写数据。此时发送FIFO

剩余的空间应大于DEST_MSIZE 值，以满足DMA通道1次搬运的数据量，如果小于则会引起对应外设FIFO 上溢错误。

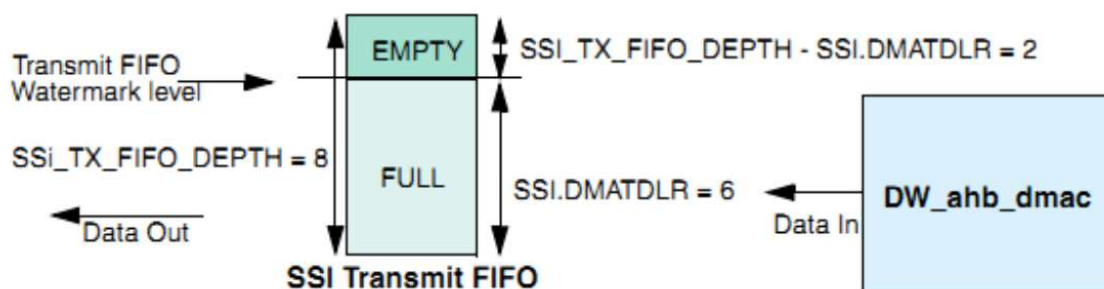
合理安全设定：



参数	注释
SSI.DMATDLR = 2	SSI外设发送FIFO 阈值
DMA.CTLx.DEST_MSIZE = 6 SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 6	DEST_MSIZE 等于 FIFO 剩余空间大小
SSI_TX_FIFO_DEPTH = 8	发送 FIFO 总深度
DMA.CTLx.BLOCK_TS = 30	块大小

不合理设定:

$DMA.CTLx.DEST_MSIZE > SSI_TX_FIFO_DEPTH - SSI.DMATDLR$



参数	注释
SSI.DMATDLR = 6	SSI外设发送FIFO 阈值
DMA.CTLx.DEST_MSIZE = 4 SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 2	DEST_MSIZE 等于 FIFO 剩余空间大小
SSI_TX_FIFO_DEPTH = 8	发送 FIFO 总深度
DMA.CTLx.BLOCK_TS = 30	块大小

当非存储器外设作为源设备时，非存储器外设接收 FIFO 中数据达到设定阈值后，向 DMA 发出请求，此时DMA 会根据 SRC_MSIZE 中的数据量由接收 FIFO 中取数据。因此接收 FIFO 中

设定的阈值应大于SRC_MSIZ值，以满足DMA通道 1次读取数据量，如果小于 1次读取数据量会导致相应外设产生 FIFO 下溢中断。

16.7 DMA寄存器

16.7.1 地址映射表

DMA基地址列表

地址范围	基地址	外设	总线
0x4000_4000-0x4000_4FFF	0x4000_4000	DMA	AHB

DMA寄存器偏移表

偏移地址	寄存器名称	宽度 (bit)	复位值	寄存器域
0x000	SAR0	32	0x00000000	DMA_Channel_0
0x004	预留	32	0x00000000	
0x008	DAR0	32	0x00000000	
0x00C - 0x017	预留 x 3	32 x 3	0x00000000	
0x018	CTL0_L	32	0x00304801	
0x01C	CTL0_H	32	0x00000002	
0x020 - 0x03F	预留 x 8	32 x 8	0x00000000	
0x040	CFG0	32	0x00000E00	
0x044 - 0x057	预留 x 5	32 x 5	0x00000000	
0x058	SAR0	32	0x00000000	DMA_Channel_1
0x05c	预留	32	0x00000000	
0x060	DAR0	32	0x00000000	
0x064 - 0x06F	预留 x 3	32 x 3	0x00000000	
0x070	CTL0_L	32	0x00304801	
0x074	CTL0_H	32	0x00000002	
0x078-0x097	预留 x 8	32 x 8	0x00000000	
0x098	CFG0	32	0x00000E00	
0x09C-0x0AF	预留 x 5	32 x 5	0x00000000	
0x0B0	SAR0	32	0x00000000	DMA_Channel_2
0x0B4	预留	32	0x00000000	
0x0B8	DAR0	32	0x00000000	
0x0BC - 0x0C7	预留 x 3	32 x 3	0x00000000	
0x0C8	CTL0_L	32	0x00304801	
0x0CC	CTL0_H	32	0x00000002	

0x0D0 - 0x0EF	预留 x 8	32 x 8	0x00000000	DMA_Channel_3
0x0F0	CFG0	32	0x00000E00	
0x0F4 - 0x107	预留 x 5	32 x 5	0x00000000	
0x108	SAR0	32	0x00000000	
0x10C	预留	32	0x00000000	
0x110	DAR0	32	0x00000000	
0x114-0x11F	预留 x 3	32 x 3	0x00000000	
0x120	CTL0_L	32	0x00304801	
0x124	CTL0_H	32	0x00000002	
0x128-0x147	预留 x 8	32 x 8	0x00000000	
0x148	CFG0	32	0x00000E00	
0x14C-0x15F	预留 x 5	32 x 5	0x00000000	
0x058 - 0x2BF	预留 x 88	32 x 88	0x00000000	预留
0x2C0	RawTfr	32	0x00000000	DMA_Interrupt
0x2C4	预留	32	0x00000000	
0x2C8	RawBlock	32	0x00000000	
0x2CC	预留	32	0x00000000	
0x2D0	RawSrcTran	32	0x00000000	
0x2D4	预留	32	0x00000000	
0x2D8	RawDstTran	32	0x00000000	
0x2DC	预留	32	0x00000000	
0x2E0	RawErr	32	0x00000000	
0x2E4	预留	32	0x00000000	
0x2E8-0x364	预留	32	0x00000000	预留
0x368	ReqSrcReg	32	0x00000000	Soft_HandShake
0x36C	预留	32	0x00000000	
0x370	ReqDstReg	32	0x00000000	
0x374	预留	32	0x00000000	
0x378	SglReqSrcReg	32	0x00000000	

0x37C	预留	32	0x00000000	
0x380	SglReqDst Reg	32	0x00000000	
0x384	预留	32	0x00000000	
0x388	LstSrcReg	32	0x00000000	
0x38C	预留	32	0x00000000	
0x390	LstDstReg	32	0x00000000	
0x394	预留	32	0x00000000	
0x398	DmaCfgR eg	32	0x00000000	DMA_Contro 1
0x39C	预留	32	0x00000000	
0x3A0	ChEnReg	32	0x00000000	
0x3A4	预留	32	0x00000000	

DMA寄存器区域划分表

寄存器域地址范围	寄存器域地址	寄存器域	外设
0x4000_4000-0x4000_4057	0x4000_4000	DMA_Channel_0	DMA
0x4000_4058-0x4000_40AF	0x4000_4058	DMA_Channel_1	
0x4000_40B0-0x4000_4107	0x4000_40B0	DMA_Channel_2	
0x4000_4108-0x4000_4015F	0x4000_4108	DMA_Channel_3	
0x4000_4160-0x4000_42BF	0x4000_0160	预留	
0x4000_42C0-0x4000_4367	0x4000_42C0	DMA_Interrupt	
0x4000_4368-0x4000_4397	0x4000_4368	Soft_HandShake	
0x4000_4398-0x4000_43B7	0x4000_4398	DMA_Control	
0x4000_43B8-0x4000_4FFF	0x4000_4FFF	预留	

16.7.2 DMAC配置寄存器（DmaCfgReg_H/ DmaCfgReg_L）

Name: DMAC Configuration Register

Size: 32 bits

Address Offset: 0x398

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20
----	----	----	----	----	----	----	----	----	----	----	----

预留											
15	14	13	12	11	10	9	8	7	6	5	4
预留											

Bit	Name	W/R	Description
63:1	预留	-	
0	DMA_EN	W/R	DMA 外设时能

16.7.3 DMAC通道使能寄存器（ChEnReg）

Name: DW_ahb_dmac Channel Enable Register

Size: 32 bits

Address Offset: 0x3a0

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20
预留											
15	14	13	12	11	10	9	8	7	6	5	4
预留							CH0_EN_WE	预留			

Bit	Name	W/R	Description
31:9	预留	-	预留
8	CH0_EN_WE	W	DMAC 通道0使能写保护位
7:1	预留	-	预留
0	CH0_EN	W/R	DMAC 通道0使能位

16.7.4 通道x源地址寄存器（SARx）（x=0）

Name: Source Address Register for Channel x

Size: 32 bits (upper 32 bits are reserved)

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20
SARx											
15	14	13	12	11	10	9	8	7	6	5	4
SARx											

Bit	Name	W/R	Description
31:0	SARx	W/R	通道 x源地址寄存器

16.7.5 通道x目标寄存器（DARx）（x=0）

Name: Destination Address Register for Channel x

Size: 32 bits (upper 32 bits are reserved)

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20
DARx											
15	14	13	12	11	10	9	8	7	6	5	4
DARx											

Bit	Name	W/R	Description
31:0	DARx	W/R	通道 x源地址寄存器

16.7.6 通道x控制寄存器（CTLx_H/ CTLx_L）（x=0）

Name: Control Register for Channel x

Size: 64 bits

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20	19
预留												

15	14	13	12	11	10	9	8	7	6	5	4	3
预留			DO NE	BLOCK_TS								

Bit	Name	W/R	Description
31:13	预留	-	
12	DONE	W/R	
11:0	BLOCK_TS	W/R	DMA 传输数据块大小，即 XXX_TR_WIDTH 单位数据数量 DMA 传输数据时是以为单位数据，1 次 DMA 传输数据块大小即以 XXX_TR_WIDTH 为单位数据的个数。 1次DMA 传输字节（Byte）量 = $BLOCK_TS * XXX_TR_WIDTH / 8$

31	30	29	28	27	26	25	24	23	22	21	20	19
预留									TT_FC			
15	14	13	12	11	10	9	8	7	6	5	4	3
SRC_MSI ZE	DEST_MSIZ			SINC		DINC		SRC_TR_WID TH			D	

Bit	Name	W/R	Description						
31:23	预留	-							
22:20	TT_FC	W/R	传输方式及流控制器选择						
			设定值与传输方式对应表：						
			<table><tr><td>设定值</td><td>传输方式</td></tr><tr><td>000</td><td>Memory to Memory</td></tr><tr><td>001</td><td>Memory to Peripheral</td></tr></table>	设定值	传输方式	000	Memory to Memory	001	Memory to Peripheral
			设定值	传输方式					
000	Memory to Memory								
001	Memory to Peripheral								

			010	Peripheral to Memory																		
19:17	预留	-																				
16:14	SRC_MSIZE	W/R	<div>源数据突发传输量（Source Burst Transaction Length）</div> <div>源数据 1 次突发传输中，传输 SRC_TR_WIDTH 的数量，即传输多少个 SRC_TR_WIDTH。</div> <div>源数据 1 次突发传输Byte = SRC_MSIZE * SRC_TR_WIDTH / 8</div> <div>设定值与数据量对应表：</div> <table><tr><td>设定值</td><td>数据量</td></tr><tr><td>000</td><td>1</td></tr><tr><td>001</td><td>4</td></tr><tr><td>010</td><td>8</td></tr><tr><td>011</td><td>16</td></tr><tr><td>100</td><td>32</td></tr><tr><td>101</td><td>64</td></tr><tr><td>110</td><td>128</td></tr><tr><td>111</td><td>256</td></tr></table>		设定值	数据量	000	1	001	4	010	8	011	16	100	32	101	64	110	128	111	256
设定值	数据量																					
000	1																					
001	4																					
010	8																					
011	16																					
100	32																					
101	64																					
110	128																					
111	256																					
13:11	DEST_MSIZE	W/R	<div>目标数据突发传输量（Destination Burst Transaction Length）</div> <div>源数据 1 次突发传输中，传输 DST_TR_WIDTH 的数量，即传输多少个DST_TR_WIDTH。</div> <div>目标数据 1次突发传输Byte = DEST_MSIZE * DST_TR_WIDTH / 8</div> <div>设定值与数据量对应表同SRC_MSIZE</div>																			
10:9	SINC	W/R	<div>源地址增量模式</div> <div>00 = 地址自加</div> <div>01 = 地址自减</div>																			

			1x = 地址不变																
8:7	DINC	W/R	目标地址增量模式 00 = 地址自加 01 = 地址自减 1x = 地址不变																
6:4	SRC_TR_WIDTH	W/R	源数据位宽 设定值与数据位宽对应表：																
			<table><tr><td>设定值</td><td>数据量</td></tr><tr><td>000</td><td>8</td></tr><tr><td>001</td><td>16</td></tr><tr><td>010</td><td>32</td></tr><tr><td>011</td><td>64</td></tr><tr><td>100</td><td>128</td></tr><tr><td>101</td><td>256</td></tr><tr><td>11x</td><td>256</td></tr></table>	设定值	数据量	000	8	001	16	010	32	011	64	100	128	101	256	11x	256
			设定值	数据量															
			000	8															
			001	16															
			010	32															
			011	64															
			100	128															
			101	256															
11x	256																		
3:1	DST_TR_WIDTH	W/R	目标数据位宽 设定值与数据位宽对应同 SRC_TR_WIDTH																
0	INT_EN	W/R	DMA 中断总控制位 0-关闭 DMA所有中断 1-打开 DMA所有中断																

16.7.7 通道x配置寄存器（CFGx）（x=0）

Name: Configuration Register for Channel x

Size: 32 bits (upper 32 bits are reserved)

Read/Write Access: Read/Write

31	30	29	28	27	26	25	24	23	22	21	20
15	14	13	12	11	10	9	8	7	6	5	4
		DEST_PER					SRC_PER				PRC

Bit	Name	W/R	Description
31:14	预留	-	
13:12	DEST_PER	W/R	指定硬握手接口 (通道与目标设备)
11:9	预留	-	
8:7	SRC_PER	W/R	指定硬握手接口 (通道与源设备)
6:5	预留	-	
4:3	PROTCTL	W/R	AHB保护控制
1	FIFO_MODE	W/R	FIFO 模式
0	FCMODE	W/R	流控模式

3	3	2	2				2	2	2	2	2	1	1	1	1
1	0	9	8	27	26	25	4	3	2	1	0	9	8	7	6

预留

 S
R
C
-
H
S
-
P
O
L

 D
S
T
-
H
S
-
P
O
L

预留

1	1	1	1												
5	4	3	2	11	10	9	8	7	6	5	4	3	2	1	0

预留

 HS
_S
EL

 HS
_S
EL

 FIF
O_
E

 C
H
-

 CH_P
RIOR

预留

	_S RC	_D ST	MP TY	S U S P		
--	----------	----------	----------	------------------	--	--

Bit	Name	W/R	Description
31:20	预留	-	
19	SRC_HS_POL	W/R	源设备握手信号 有效极性 0-高有效 1-低有效
18	DST_HS_POL	W/R	目标设备握手信号 有效极性 0-高有效 1-低有效
17:12	预留	-	
11	HS_SEL_SRC	W/R	源设备软/硬握手 接口选择 0-硬握手 1-软握手
10	HS_SEL_DST	W/R	目标设备软/硬握手 接口选择 0-硬握手 1-软握手
9	FIFO_EMPTY	R	通道 FIFO 状态 0-通道 FIFO 未 满 1-通道 FIFO 满
8	CH_SUSP	W/R	通道暂停 (Channel Suspend) 0-不暂停 1-暂定 DMA通

			道从源设备获取数据
7:5	CH_PRIOR	W/R	通道优先级 设定值范围： $0 < \text{CH_PRIOR} < 3$ (通道数 - 1) 超出设定范围会导致异常错误
4:0	预留	-	

16.7.8 原中断状态寄存器

Name: Interrupt Raw Status Registers

Size: 32 bits

Address Offset:

RawTfr – 0x2c0

RawBlock – 0x2c8

RawSrcTran – 0x2d0

RawDstTran – 0x2d8

RawErr – 0x2e0

Read/Write Access: Read/Write

原中断状态寄存器包括：RawBlock、RawDstTran、RawErr、RawSrcTran、RawTfr

具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。

原中断状态寄存器中状态值不受中断屏蔽寄存器（Mask）影响。

通过对对应中断清除寄存器写“1”，清除相应原中断状态寄存器中的标记位。

31	30	29	28	27	26	25	24	23	22	21	20
预留											
15	14	13	12	11	10	9	8	7	6	5	4
预留											

Bit	Name	W/R	Description
31:1	预留	-	
0	CH0	W/R	原中断标志位， 写操作通常用于 测试，普通

16.7.9 状态寄存器

Name: Interrupt Status Registers

Size: 32 bits

Address Offset:

StatusTfr – 0x2c0

StatusBlock – 0x2c8

StatusSrcTran – 0x2d0

StatusDstTran – 0x2d8

StatusErr – 0x2e0

Read/Write Access: Read/Write

中断状态寄存器包括：StatusBlock、StatusDstTran、StatusErr、StatusSrcTran、StatusTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。

中断状态寄存器中状态值会中断屏蔽寄存器(Mask)影响，当中断状态寄存器bit位置“1”，
外设会

对CPU产生中断信号。

通过对对应中断清除寄存器写“1”，清除相应原中断状态寄存器中的标记位。

31	30	29	28	27	26	25	24	23	22	21	20
预留											
15	14	13	12	11	10	9	8	7	6	5	4
预留											

Bit	Name	W/R	Description
31:1	预留	-	
0	CH0	R	中 断 状 态 标 志

			位。 0-未产生中断 1-产生中断
--	--	--	-------------------------

16.7.10 中断屏蔽寄存器

Name: Interrupt Mask Registers

Size: 32 bits

Address Offset:

MaskTfr – 0x310

MaskBlock – 0x318

MaskSrcTran – 0x320

MaskDstTran – 0x328

MaskErr – 0x330

中断屏蔽寄存器包括：MaskBlock、MaskDstTran、MaskErr、MaskSrcTran、MaskTfr
 具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。

中断屏蔽寄存器（Mask）用于屏蔽DMA中断，影响中断状态寄存器（Status）值。

每个中断屏蔽位对应1个写保护操作为。

例如：打开通道0中断，对MaskBlock写0x01x1，其中仅通道0操作有效，其余x对应位操作无效。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留							C	预留							C	
							H								H	
							0								0	
							—									

	W E		
--	--------	--	--

Bit	Name	W/R	Description
31:9	预留	-	
8	CH0_WE	W	中断屏蔽写保护位 0-CHx 写操作无效 1-CHx 写操作有效
7:1	预留	-	
0	CH0	W/R	中断状态标志位。 0-中断关闭 1-中断打开

16.7.11 中断状态寄存器

Name: Interrupt Clear Registers

Size: 32 bits

Address Offset:

ClearTfr – 0x2c0

ClearBlock – 0x2c8

ClearSrcTran – 0x2d0

ClearDstTran – 0x2d8

ClearErr – 0x2e0

Read/Write Access: Read/Write

中断状态寄存器包括：ClearBlock、ClearDstTran、ClearErr、ClearSrcTran、ClearTfr
具体说明见“DMA中断”章节

以上寄存器结构及对应通道相同，操作地址不同。

对各中断清除寄存器写“1”，原中断寄存器中的标志位会在同一周期被清“0”。

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6	5	4	3	2	1	0
5	4	3	2	1	0											
预留															C	
															H	
															0	

Bit	Name	W/R	Description
31:1	预留	-	
0	CH0	W	中断状态标志位。 0-无效 1-清除中断

16.7.12 组合中断状态寄存器（ChEnReg）

Name: Combined Interrupt Status Register

Size: 32 bits

Address Offset: 0x360

Read/Write Access: Read/Write

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	
预留																
1	1	1	1	1	1		9	8	7	6		4	3	2	1	0
5	4	3	2	1	0											
预留											E R R	D S T T	S R C T	B L O C K	T F R	

Bit	Name	W/R	Description
-----	------	-----	-------------

31:5	预留	-	
4	ERR	R	错误状态中断
3	DSTT	R	目标数据传输完成中断
2	SRCT	R	源数据传输完成中断
1	BLOCK	R	块传输中断
0	TFR	R	传输完成中断

16.7.13 软握手接口寄存器

Name: Interrupt Mask Registers

Size: 32 bits

Address Offset:

ReqSrcReg- 0x368

ReqDstReg-0x370

SglReqSrcReg-0x378

SglReqDstReg-0x380

LstSrcReg-0x388

LstDstReg-0x390

Read/Write Access: Read/Write

软握手接口寄存器：ReqSrcReg、ReqDstReg、SglReqSrcReg、SglReqDstReg、LstSrcReg、LstDstReg

功能说明见“DMA软握手接口”章节

以上寄存器结构及对应通道相同，操作地址不同。

每个中断屏蔽位对应1个写保护操作。

例如：对应通道0产生源数据DMA请求，对ReqSrcReg写0x0101，其中仅通道0操作有效，其余x对

应位操作无效。

31	30	29	28	27	26	25	24	23	22	21	20
预留											
15	14	13	12	11	10	9	8	7	6	5	4

预留	CH 0	预留
----	---------	----

Bit	Name	W/R	Description
31:9	预留	-	
8	CH0_WE	W	中断屏蔽写保护位 0-CHx 写操作无效 1-CHx 写操作有效
7:1	预留	-	
0	CH0	W/R	DMA 请求位。 0-无效 1-产生请求

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

17 快速循环冗余校验（FASTCRC）

17.1 FASTCRC简介

快速循环冗余校验计算单元（FASTCRC）根据选定的生产多项式得到任一16/32位的CRC计算果。在其他的应用中，CRC技术主要应用于核实数据传输或者数据存储的正确性和完整性。

17.2 FASTCRC主要特性

- 支持CRC-16/32 两种CRC计算方式
- 支持CRC-16多项式：0x8005和0x1021
- 支持CRC-32多项式：0x04C11DB7
- 数据按字节输入
- 输入数据可配置由硬件进行大小端翻转
- 输出数据可配置由硬件进行大小端翻转
- 支持指定CRC计算初始值

17.3 FASTCRC功能描述

FASTCRC单元包含一个32位的数据输入寄存器

- 对其进行写操作时，作为32位的输入寄存器
- 对其进行读操作时，作为输出寄存器，其有效位宽由 CRC 控制状态寄存器（CRC_CSR）决定。

根据需要配置完成控制状态寄存器（CRC_CSR）和 CRC 计算初始值寄存器（CRC_INI）后对数据寄存器进行连续写操作，需要校验的数据写操作完成后，对 CRC 数据寄存器进行读操作获取CRC校验值。

下图为CRC操作流程图中：

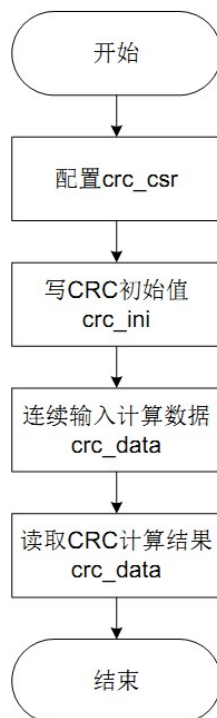


图17.1

17.4 FASTCRC寄存器

17.4.1 地址映射表

FASTCRC外设基地址表

地址范围	基地址	外设	总线
0x4000_8000-0x4000_8FFF	0x4000_8000	FASTCRC	AHB

FASTCRC寄存器表

偏移地址	寄存器名称	宽度 (bit)	复位值
0x00	CRC_CSR	32	0x000000C0
0x04	CRC_INI	32	0x00000000
0x08	CRC_DATA	32	0x00000000
0x08	CRC_DATA	32	0x00000000

17.4.2 CRC控制状态寄存器 (CRC_CSR)

31 30 29 28 27 26 25 24

预留							
23	22	21	20	19	18	17	16
预留							
15	14	13	12	11	10	9	8
预留							crc_b usy
7	6	5	4	3	2	1	0
byte_num	xor_o ut_sel	rev_o ut_sel	byte_ rev_s el	rev_i n_sel	type_ sel	poly_ sel	

Bit	Name	W/R	Description
31:9	预留	-	
8	crc_busy	RO	CRC 计算busy信号，当 busy信号为0时才能下一次 crc计算,CRC计算需要4个 HCLK
7:6	byte_num	RW	选择CRC计算的byte数量 0: 只对byte0进行CRC运算 1: byte0,byte1 2: byte0,byte1,byte2 3: byte0,byte1,byte2,byte3
5	xor_out_sel	RW	CRC 计算结果和0xffff进行异或； (此步骤在rev_out_sel之后进行) 1: 和0xffff进行异或； 0: 计算结果直接输出。
4	rev_out_sel	RW	CRC 计算结果高低位反转； 1: 反转； 0: 不反转。
3	byte_rev_sel	RW	CRC 输入byte大小端翻

			转; 1: 计算顺序 byte3->byte2->byte1->byte0 0: 计算顺序 byte0->byte1->byte2->byte3
2	rev_in_sel	RW	CRC 8-bit输入大小端反转 进行计算, 如bit7作为bit0 参与运算, bit6作为bit1, 以此类推。 1: 反转; 0: 不反转。
1	type_sel	RW	CRC类型选择; 1: CRC32; 0: CRC16。
0	poly_sel	RW	CRC16的多项式选择, 当选择CRC32时, 该位无效。 1: 0x1021; 0: 0x8005。

17.4.3 初始值寄存器 (CRC_INI)

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

Bit	Name	W/R	Description
31:0	Crc_ini[31:0]	WO	CRC计算的初始值; 计算CRC16时, 低16

			位有效。
--	--	--	------

17.4.4 数据寄存器（CRC_DATA）

CRC_DATA_IN输入（写操作）

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

Bit	Name	W/R	Description
31:24	din[24:31]	W	CRC的数据输入， byte3
23:16	din[16:23]	W	CRC的数据输入， byte2
15:8	din[8:15]	W	CRC的数据输入， byte1
7:0	din[7:0]	W	CRC的数据输入， byte0

CRC_DATA_OUT输出（读操作）

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

Bit	Name	W/R	Description
31:0	dout[31:0]	RO	CRC的计算结果输出； CRC16时，仅[15:0]位有效。